МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДНІПРОВСЬКИЙ ДЕРЖАВНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Л.В. Дранишников

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни " **Інтелектуальні методи в управлінні** "
для здобувачів другого (магістерського) рівня вищої освіти зі спеціальності 121 «Інженерія програмного забезпечення» за освітньо-професійною програмою «Інженерія програмного забезпечення»

ЗАТВЕРДЖЕНО:
редакційно–видавничою секцією
науково–методичної ради ДДТУ
16.01. 2020 р., протокол №_1_

Кам'янське
2020

Конспект лекцій з дисципліни «Інтелектуальні методи в управлінні**»** для здобувачів другого (магістерського) рівня вищої освіти зі спеціальності 121 «Інженерія програмного забезпечення» за освітньо-професійною програмою «Інженерія програмного забезпечення»

/Укладач– Дранишников Л.В. Кам'янське, ДДТУ, 2020 –188с.

Укладач:
докт.техн. наук, проф. Дранишников Л.В.

Відповідальний за випуск:
докт. техн. наук, проф. Шумейко О.О.

Рецензент:
докт.техн. наук, проф. Самохвалов С.Є.

Затверджено на засіданні каф. „Програмне забезпечення автоматизованих систем”
Протокол № 12 від 09.12.2019 р.

ЗМІСТ

Основні вимоги з техніки безпеки		4
Тема 1. **Лекція 1-4**	Введення. Задачі штучного інтелекту. Структура досліджень в області штучного інтелекту. Інтелектуальний інтерфейс. Нова технологія рішення задач. Прикладні системи, орієнтовані на знання. Основні проблеми штучного інтелекту. Представлення знань.	5
Тема 2. **Лекція 5-8**	Моделі представлення знань. Продукційна модель. Модель, заснована на використанні фреймів. Модель семантичної мережі. Логічна модель.	28
Тема 3. **Лекція 9-12**	Експертні системи. Призначення експертних систем. Структура експертної системи. Функціонування ЕС. Класифікація інструментальних засобів. Етапи створення експертних систем. Автоматичне міркування. Основні механізми дедукції. Розробка ЕС, у якій реалізується зворотний ланцюжок міркувань. Стан і тенденції розвитку штучного інтелекту.	59
Тема 4. **Лекція 13-19**	Нейронні мережі. Застосування нейронних мереж. Структура штучного нейрона. Функції активації. Класифікація нейронних мереж. Робота нейромережі. Навчання нейронной мережі. Алгоритми навчання нейронних мереж. Типи нейронних мереж. Пакет Neural Networks Toolbox. Приклади створення і використання нейронних мереж. Використання Simulink при побудові НС. Система автоматичного керування з нейромережевими регулятором на основі еталонної моделі	94
Тема 5. **Лекція 20-22**	Нейро-нечітке моделювання в середовище MatLab. Моделювання і реалізація нейрон-нечіткої мережі в середовищі MatLab. Синтез нейрон-нечіткої мережі в середовищі MatLab.	131
Тема 6. **Лекція 23-27**	Генетичні алгоритми. Кодування в генетичних алгоритмах. Схрещування. Мутація. Прийоми виконання генетичних алгоритмів.	150
Література		188

Основні вимоги з техніки безпеки

При виконанні лабораторних робіт у комп’ютерних залах студент повинен дотримуватися означених нижче правил техніки безпеки:

1. Не приступати до роботи, не прослухавши інструктаж з техніки безпеки та не зареєструвавшись у журналі користувачів ПК (перед початком виконання першої роботи).
2. Не вмикати чи вимикати живлення у комп'ютерному залі або на своєму робочому місці та не перезавантажувати ПК.
3. Не змінювати налагоджень комп'ютера та його операційної системи.
4. Не встановлювати та не видаляти жодних програм.
5. Не працювати за комп’ютером більше як удвох одночасно.
6. Закінчивши виконання лабораторної роботи скопіювати результати на власний носій інформації та закрити всі відкриті програми без збереження на ПК власних результатів.

Лекції 1-2. Введення

Під інтелектуальними методами розуміються такі способи розв'язання завдань, в основі яких лежать алгоритми та дії, більшою чи меншою мірою пов'язані з інтелектуальною діяльністю людини. Термін "інтелектуальні" в області комп'ютерних технологій можна вважати усталеним і поєднання "інтелектуальні інформаційні технології" не ріже слух і сприймається фахівцями однозначно.

Клас інтелектуальних технологій (ІТ) включає наступні напрямки:

• штучні нейронні мережі (ШНМ);
• нечітка логіка (НЛ);
• генетичні алгоритми (ГА);
• нелінійна динаміка (НД).

Штучні нейронні мережі складаються з окремих обчислювальних елементів (формальних нейронів), які певною мірою подібні до біологічних нейронів мозку людини. Характерна особливість ІНС полягає в тому, що процес програмування традиційного шляхи вирішення завдання замінюється процедурою навчання мереж. Метод навчання ШНМ є одним з головних класифікаційних ознак мереж. Способи об'єднання нейронів у мережі, кількість шарів нейронів, наявність (відсутність) зворотних зв'язків визначають архітектуру ІНС. Мережі різних архітектур призначені для вирішення різних завдань, з яких найбільш важливий клас погано формалізованих завдань, де немає явних залежностей між вхідними і вихідними величинами. До проблем, які можуть бути вирішені з допомогою ШНМ, ставляться завдання класифікації і ранжування підприємств, фірм, побудови рейтингів банків, прогнозування, зміни обмінного курсу валют і т. д.

Нечітка логіка і правила, що засновані на її концепції, являють собою засіб моделювання невизначеностей природничих понять мови. Прикладами таких невизначеностей служать лінгвістичні змінні холодна, тепла, гаряча стосовно до температури води. На вхід системи з нечіткою логікою надходять чіткі змінні, які перетворюються на нечіткі підмножини. З допомогою експертів створюється база правил виду "якщо ..., то ...", включається в систему. Над вхідними змінними за допомогою бази правил виробляються необхідні

перетворення, після чого за допомогою методів дефазифікації (переходу від нечітких змінних до чітким) на виході системи формується чітка вихідна змінна. Серед управлінських завдань, розв'язуваних з допомогою систем НЛ, можна виділити клас проблем, де при нечітких вхідних змінних потрібно отримати кількісну характеристику вихідної величини.

Генетичні алгоритми являють собою алгоритми пошуку оптимальних рішень, які побудовані на принципах природного відбору і генетики. Будь-яке можливе рішення відображається у вигляді рядка (хромосоми) фіксованої довжини, популяції яких застосовуються традиційні генетичні оператори: селекція, схрещування, мутація. Селекція направляє генетичний пошук в перспективні райони простору рішень, схрещування виконує випадковий пошук поблизу локального оптимуму, мутація відшукує покращене рішення. Визначаючи в кожному поколінні кращі хромосоми, схрещуємо їх між собою для отримання потомства, яке являє близьке до оптимального вирішення завдання. До задач, що розв'язуються за допомогою ГА, можна віднести складання плану оптимальних перевезень, визначення кращої торгової стратегії, розміщення виробничих потужностей.

Нелінійна динаміка або хаос узгоджується більшою мірою щоденного досвіду людини набагато сильніший, ніж точна передбачуваність. Нелінійна динаміка — міждисциплінарна наука, в якій вивчаються властивості нелінійних динамічних систем. Нелінійна динаміка використовує для опису систем нелінійні моделі, зазвичай описуються диференціальними рівняннями і дискретними відображеннями.

Теорія хаосу — математичний апарат, що описує поведінку деяких нелінійних динамічних систем, схильних при певних умовах явищу, відомому як хаос (динамічний хаос, детермінований хаос). Поведінка такої системи здається випадковим, навіть якщо модель, що описує систему, є детермінованою.

Синергетика або теорія складних систем — міждисциплінарний напрям науки, що вивчає загальні закономірності явищ і процесів у складних нерівноважних системах (фізичних, хімічних, біологічних, екологічних, соціальних та інших) на основі властивих їм принципів самоорганізації. Синергетика є міждисциплінарним підходом, оскільки принципи, що керують процесами самоорганізації, подаються одними і тими ж незалежно від природи систем, і для їх опису повинен бути придатний загальний математичний апарат. Методи нелінійної динаміки дозволяють моделювати швидкі нерівноважні процеси (так звані «фазові переходи») в економічних системах, пов'язані з

переходом від одних стійких станів в інші. Велике значення для діяльності світового фінансового ринку має завдання прогнозування його динаміки як у періоди зростання, так і в стані кризи і падіння. Ця задача важлива як для практиків, так і для економістів і фінансистів-теоретиків. Учасникам фінансових ринків необхідно оцінювати прибутковість вкладень, оптимізувати портфель, визначати моменти входу на ринок і виходу з нього, обчислювати справедливу вартість деривативів. Ризик-менеджерам потрібно точно оцінювати ризикові характеристики активів і портфелів, такі як вартість під ризиком.

Для рішення подібних задач розроблено безліч різних підходів – від емпіричних методів технічного аналізу до агентно-орієнтованих імітаційних моделей або технологій «розкопування даних» (data mining). Таке різноманіття пояснюється тим, що динаміка прибутковості на фінансових ринках не може бути якісно описана лінійними моделями ARMA і випадкового блукання.

При цьому нелінійні методи аналізу динаміки і відповідні технології управління ризиками розроблені й успішно застосовуються для розвинених фінансових ринків США та Європи. Вони мало вивчені по відношенню до нестійким ринків.

Штучний інтелект

Штучний інтелект (ШІ) - це програмна система, що імітує на комп'ютері мислення людини. Для створення такої системи необхідно вивчити процес мислення людини, вирішальної певні задачі або приймаючого рішення в конкретній області, виділити основні кроки цього процесу і розробити програмні засоби, відтворюючі їх на комп'ютері.

Штучний інтелект (ШІ) - сукупність наукових дисциплін, що вивчають методи рішення задач інтелектуального (творчого) характеру з використанням ЕОМ. Штучний інтелект - один з напрямів інформатики, метою якого є розробка апаратно-програмних засобів, що дозволяють користувачу-непрограмісту ставити і вирішувати свої, що традиційно вважаються інтелектуальними задачі, спілкуючись з ЕОМ на обмеженій підмножині природної мови.

Системи штучного інтелекту (СШІ) — це системи, створені на базі ЕОМ, які імітують рішення людиною складних інтелектуальних задач.

Знання: у загальному випадку знання — перевірений практикою результат пізнання дійсності, вірне її віддзеркалення в мисленні людини, володіння досвідом і розумінням, які є правильними і в суб'єктивному, і в об'єктивному

відношенні, на підставі яких можна побудувати думки і висновки, уявні достатньо надійними для того, щоб розглядатися як знання. Тому в контексті ШІ термін знання - це інформація, присутня при реалізації інтелектуальних функцій. Звично це відхилення, тенденції, шаблони і залежності, знайдені в інформації Іншими словами, інтелектуальні системи є в той же час системами обробки знань.

Задачі штучного інтелекту: область ШІ має більш ніж сорокалітню історію розвитку. Із самого початку в ній розглядався ряд вельми складних задач, які, разом з іншими, і дотепер є предметом досліджень: докази теорем; розпізнавання образів; робототехніка; моделювання ігор; інженерія знань; експертні системи

Одну і ту ж задачу можна запрограмувати, використовуючи або традиційні методи, або методи штучного інтелекту. Програми ШІ володіють особливою властивістю, схожою на характерну властивість людського інтелекту - зміна будь-якої, навіть невеликій частині інформації не впливає на структуру всієї програми. Така гнучкість додає процесу програмування велику ефективність, дає можливість створювати програми, що уміють "розуміти", тобто володіючи рисами розуму.

Методи ШІ припускають високий ступінь незалежності окремих частин програми, кожна з яких реалізує певний крок рішення однієї або декількох задач. Незалежні частини програми можна порівняти з окремими з окремими блоками інформації в людській пам'яті. Вибираючи потрібну інформацію, людський мозок автоматично підключає факти, що тільки відносяться до справи, не перебираючи всі доступні йому знання.

Цілі, факти, правила. При проектуванні систем ШІ завжди слід пам'ятати про мету, для досягнення якої вони призначені. Метою називають кінцевий результат, на який направлені розумові процеси людини. Думки, що ведуть до кінцевого результату, не випадкові, а строго обгрунтовані. Кожен крок на шляху до головної мети має свою локальну мету. Мозок завжди зосереджений на меті незалежно від того, чи виконує людина просту фізичну роботу або вирішує складну інтелектуальну задачу. Мета примушує людину думати. Люди роблять що-небудь не тому, що думають, а думають, тому що повинні що-небудь зробити. Загалом, інтелект можна представити як сукупність фактів і способів їх застосування для досягнення мети. Частково цілі досягаються за допомогою правил використовування всіх відомих фактів.

Приклад 1. Факт. У часи пік на вулиці багато машин. **Правило**. ЯКЩО спробувати в часи пік перейти шосе, ТО можна потрапити під машину.
Приклад 2. **Факт 2а.** Тихі, темні вулиці небезпечні. **Факт 2б.** Літні люди звичайно не скоюють зухвалих злочинів. **Факт 2в.** Поліція захищає людей від злочинців.
Правило 2а. ЯКЩО на тихій, темній вулиці зустрінеться літня людина, ТО можна не дуже турбуватися. **Правило 2б**. ЯКЩО на тихій, темній вулиці ви побачите поліцейського, ТО можна відчувати себе в безпеці.

Помітимо, що в приведених прикладах всі правила виражені умовним відношенням ЯКЩО - ТО, тобто якщо виконується деяка умова, то виконується певна дія або яка-небудь інша реакція. Факти і правила можуть бути різній складності. Звичайно досягши мети люди зв'язують складні сукупності фактів і правил.

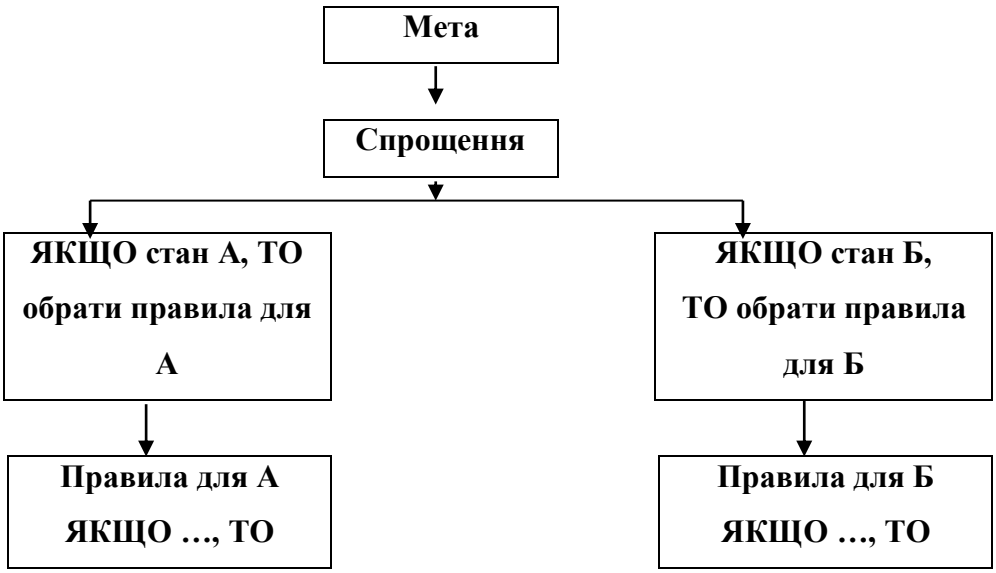
Спрощення. Яким чином людський мозок з величезної різноманітності фактів і правил швидко вибирає підмножину, відповідну до конкретної ситуації? Річ у тому, що в мозку людини існує складна система, керівна вибором правильної реакції на конкретну ситуацію. Такий вибір називається спрощенням. Механізм вибору блокує думки, що не мають відношення до вирішуваної в даний момент задачі. Механізм спрощення сприяє досягненню мети, ігноруючи всі даремні для цього факти, тобто механізм спрощення примушує мозок зосередитися тільки на фактах і правилах, потрібних для досягнення поставленої мети. На рис.1 приведена схема роботи механізму спрощення при виборі необхідних правил і фактів.

Тут механізм спрощення використовується при виборі правил для ситуації А і б. В процесі спрощення відповідно до відношення ЯКЩО - ТО перевіряється деяка умова. Якщо умова задовольняє стану А, вибираються правила для А, або навпаки.

Механізм висновку. Досягаючи мети, людина не тільки приходить до рішення поставленої перед ним задачі, але одночасно придбаває нові знання. Наприклад: 1. ІВАН і МАР’Я - батьки ДІМИ. 2. ІВАН і МАР’Я - батьки СВЄТИ. Ким доводяться один одному ДІМА і СВЄТА. Механізм спрощення примушує людину звернутися до того, що зберігається в його мозку правилу: ЯКЩО у дівчинки і хлопчика одні і ті ж батьки, ТО хлопчик і дівчинка - брат і сестра. Мета миттєво досягнута.

Частина інтелекту, яка допомагає витягувати нові факти, називається механізмом висновку. Саме механізм висновку дозволяє людині вчитися на

досвіді, оскільки він дає можливість генерувати нові факти з вже існуючих, застосовуючи наявні знання до нової ситуації.



Структура досліджень у області штучного інтелекту.

Дослідження у області **ШІ** направлені в першу чергу на рішення найважливіших проблем, що стоять на шляху до масового упровадження обчислювальних машин і роботів в різні галузі господарства, в системи управління різного рангу, в наукові дослідження, в процеси проектування і конструювання нових технічних систем і, нарешті, в наш повсякденний побут. У подальшому ми розглянемо ті методи і ті моделі, яких має в своєму розпорядженні зараз штучний інтелект для вирішення задач, що стоять перед ним.

Штучний інтелект – новий етап в розвитку людино-машинних інформаційно-розрахункових систем. Програмно-апаратні засоби штучного інтелекту дозволяють утворити інтерфейс між непрограмуючими користувачами і сучасними обчислювальними системами і комплексами. Інтелектуальний інтерфейс представляє можливість створення орієнтованих на знання інтелектуальних інформаційно-пошукових систем, інтелектуальних пакетів прикладних програм і розрахунково-логічних систем, експертних систем.

На рис.2. вказані чотири великі групи досліджень, що входять в область інтересів штучного інтелекту.

У першій групі зосереджені дослідження по відтворенню на ЕОМ процедур, що є для людини творчими. В рамках цих досліджень створюються програми для шахової гри або програми, складаючі музичні твори. Сюди ж відносяться програми для доказу теорем і програми, імітуючі здібність людини до навчання. Дослідження такого типу історично були найранішими. Вони і вплинули на вибір назви всієї наукової області, про яку йде мова. Питома вага і значущість робіт, що відносяться до імітації творчих процесів, зараз невеликі. Ці дослідження носять фундаментальний характер. В ході їх формуються і відпрацьовуються основні методи і прийоми штучного інтелекту.

У другій групі зосереджені дослідження, метою яких є рішення проблеми інтелектуалізації ЕОМ майбутніх поколінь. Якщо гранично стисло охарактеризувати цю групу досліджень, то можна сказати, що основним результатом їх є побудова обчислювальних машин, з якими можуть спілкуватися непідготовлені користувачі.

До **третьої групи** віднесені всі дослідження, направлені на створення нових принципів обробки інформації і рішення задач. Ці принципи використовують характерні особливості методів штучного інтелекту і, перш за все, те, що всі ці методи оперують із знаннями так само, як це роблять люди, коли виконують творчу роботу.

Четверта група – це дослідження, пов'язані із створенням інтелектуальних роботів, здатних автономно вирішувати складні задачі. Тут розв'язуються специфічні задачі, пов'язані з переміщенням роботів в реальному середовищі, обробкою зорової інформації, нормативною поведінкою роботів і багато чим іншим.

Дослідження, що входять до трьох останніх груп, носять ясно виражений прикладний характер. Завдяки цим дослідженням штучний інтелект привернув увагу великого числа фахівців-прикладників.

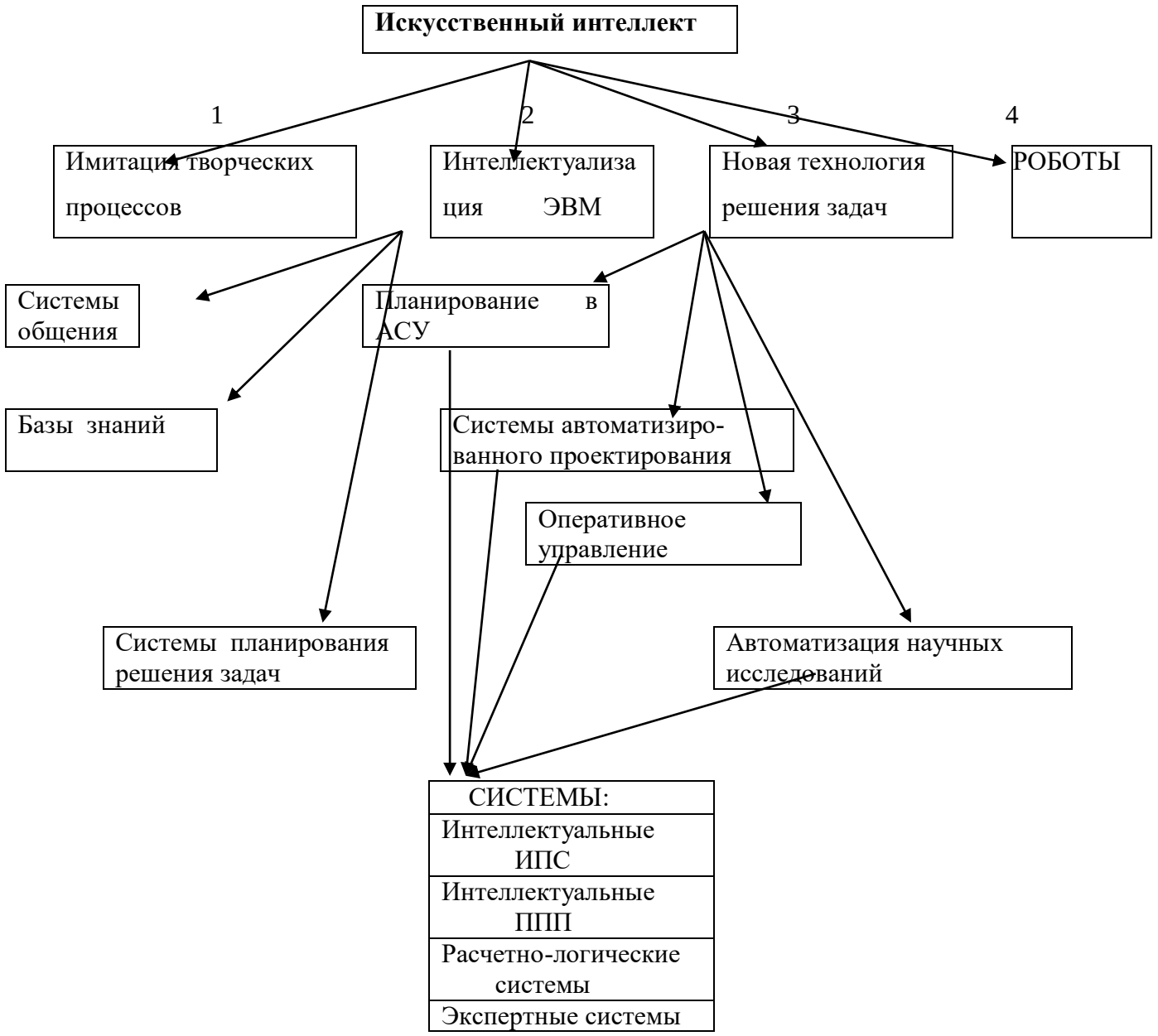


Рис.2. Структура досліджень у області штучного інтелекту.

Лекція 3. Інтелектуальний інтерфейс.

Не дивлячись на те, що програмування як спеціальність з'явилася лише 45 років тому, кількість програмістів

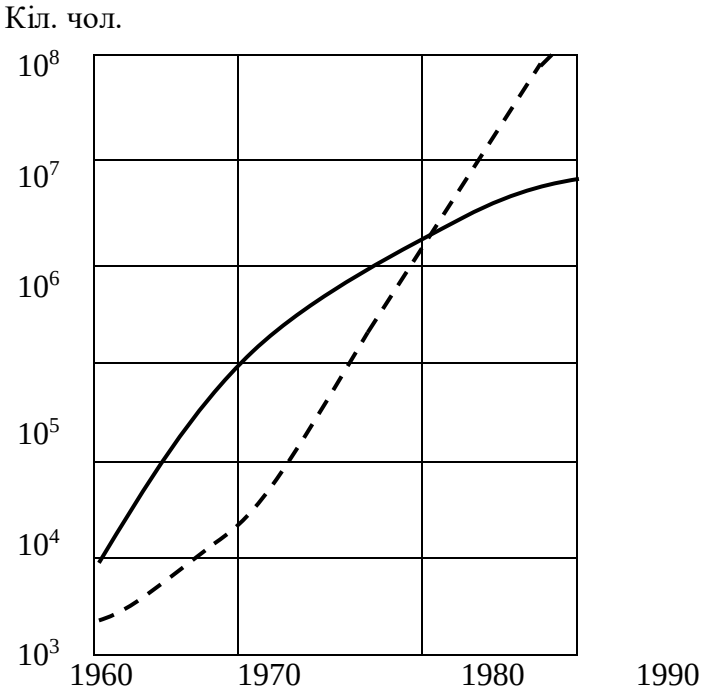


Рис.3.

На рис.3 показані два графіки, характеризуючі зростання в США числа професійних програмістів (безперервна крива) і тих, хто програмує сам (пунктирна крива). Професійних програмістів прийнято підрозділяти на системних і прикладних. Системні програмісти – це ті, хто зайнятий створенням базового математичного забезпечення для ЕОМ. Вони розробляють операційні системи, транслятори з різних мов, сервісні програми і ін. Прикладні програмісти пишуть програми, що дозволяють вирішувати конкретні задачі, що цікавлять користувачів ЕОМ. Це значить, що їх єдиною роботою є посередництво між користувачем і ЕОМ. З рис. 3 видно, що спостерігається тенденція швидкого зростання людей, зайнятих програмуванням. Якщо збережеться ця тенденція, то незабаром все доросле населення земної кулі буде зайняте програмуванням. Абсурдність цього очевидна. Одним з виходів було б

виключення програміста з ланцюга користувач – програміст – ЕОМ. Але це досить важко. Річ у тому, що рішення складних задач включає в цей ланцюг не тільки одного програміста, а цілу групу людей. Кінцевий користувач (КП) спілкується з аналітиком (програміст – аналітик). Його функція – переклад задачі КП в деяку початкову формальну модель. Кінцевий користувач (біолог, лінгвіст, хімік і т.д.), швидше за все, сформулює виниклу перед ним задачу на мові своєї професії. Мистецтво аналітика полягає в умінні по цьому формулюванню побудувати адекватну їй математичну задачу.

Те, що зуміє зробити аналітик, є початкове представлення задачі для фахівця наступного рівня – прикладного програміста (ПП). Його мета перетворити продукт аналітика в програмний продукт.

При такій технології рішення задачі на ЕОМ КП розплачується низькою ефективністю своєї праці, залежністю від інших людей, які вносять свої спотворення в його уявлення про проблему, вирішувану задачу і інтерпретацію рішення.

Першим кроком на шляху виключення з технологічного ланцюжка зайвих програмістів був перехід до термінального доступу до ЕОМ. Другим кроком з'явилося те, що КП дістав прямий доступ до ЕОМ, але для цього йому треба було навчитися виконувати роботу аналітика. По цьому шляху пішло немало фахівців. Саме ці люди поповнюють ряди тих, зростання числа яких характеризує пунктирний графік на рис.3.

Але не кожен фахівець встане на шлях навчання новій для нього справі. Для цього треба подолати неминучий психологічний бар'єр і мати смак до діяльності, що вимагає орієнтованого на ЕОМ стилю мислення. Отже, виникає наступна задача. Якщо фахівці не хочуть ”опускатися” до рівня машини, то можна спробувати підняти рівень ЕОМ, зробити спілкування з нею схожим на спілкування двох фахівців з однієї проблемної області. Іншими словами, системні програмісти повинні створити такі засоби, реалізовувані всередині ЕОМ, щоб всі функції аналітика узяла на себе ЕОМ.

Які ж функції аналітика потрібно змоделювати в машині? Перш за все, функції, пов'язані із спілкуванням кінцевого користувача з аналітиком. КП, висловлюючи аналітику суть проблеми, що виникає перед ним, користується своєю професійною мовою. Професійна мова складає не дуже велику підмножину звичної природної мови. Ця підмножина володіє рядом властивостей, що полегшують взаємне розуміння фахівцями суті

обговорюваних проблем. Фрази і тексти в професійних мовах будуються за жорсткішими правилами, ніж довільні фрази і тексти мови у всій його повноті, а для того, щоб ЕОМ могла розуміти професійну мову, формулювання тієї задачі, яку їй хоче повідомити кінцевий користувач, вона повинна мати два **сорти знань:**

- знання про мову і
- знання про ту проблемну область, в якій працює користувач.

Знання першого типу використовуються в діалоговій системі, що здійснює зв'язок між користувачем і ЕОМ на професійній мові. Часто говорять не про діалогову систему, а про лінгвістичний процесор. Це може бути система спеціальних програм або складний комплекс, що включає і апаратуру і програми. Знання про мову зберігаються прямо в **діалоговому** процесорі.

Знання про **проблемну область** зберігаються в спеціальній **базі знань**, яка, подібно діалоговому процесору, може бути фізично реалізована різними способами. Для практично цікавих проблемних областей база знань може мати великий об'єм пам'яті і спеціальні засоби для роботи з відомостями, що зберігаються в ній.

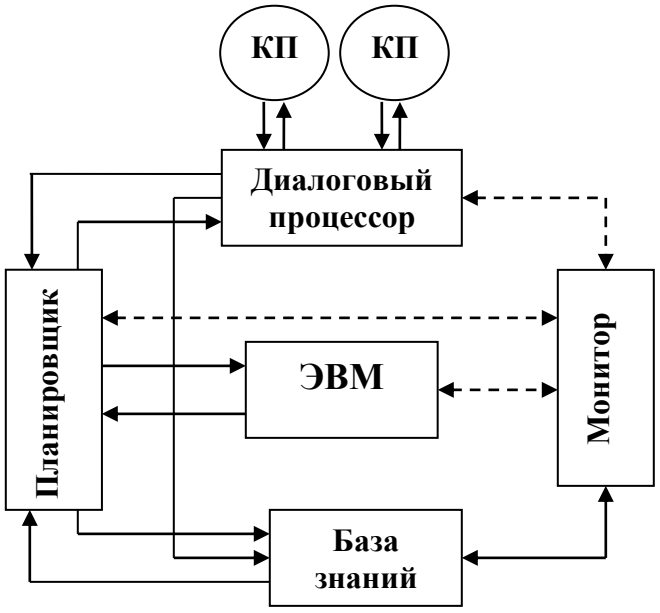


Рис.4.

Нарешті, необхідно уміти відтворювати і ще одну функцію аналітика – уміти перетворювати опис початкової задачі в робочу програму, її вирішальну. Комплекс засобів, що дозволяють робити це, називають плануючою системою або просто **планувальником**. Під час своєї роботи планувальник повинен

постійно контактувати з базою знань. З цієї бази він черпає інформацію не тільки про проблемну область і способи рішення в ній тих або інших задач, але і інформацію про те, як складаються робочі програми для ЕОМ, тобто відомості про можливості автоматичного синтезу програм з деяких базових програм, що зберігаються все в тій же базі. На рис.4. можна бачити ЕОМ в оточенні чотирьох блоків. Всі чотири блоки в сукупності утворюють ”інтелектуальний рівень” ЕОМ і називаються інтелектуальним інтерфейсом.

Таким чином, інтелектуальний інтерфейс можна створити тільки на основі методів і ідей, що розробляються в штучному інтелекті.

Нова технологія рішення задач.

Розробка засобів, створюючих інтелектуальний інтерфейс, випереджає розробку ЕОМ п'ятого покоління (швидкодія 1012 операцій в секунду; об'єм оперативної пам'яті досягати 1012 байтів; елементна база на основі надвеликих інтегральних схемах – СНІС – на одному кристалі може розміщуватися 106 логічних елементів, на яких зберігаються багато програм, що часто зустрічаються при рішенні задач з даної проблемної області – перехід від звичного програмування до програмування прямо на кристалі; істотна увага надається не тільки обчислювальним, але і логічним операціям, бо велика частина задач – це задачі інформаційного типу – продуктивність ЕОМ оцінюється через нову спеціальну одиницю ЛВС – логічні висновки в секунду – 108-109 ЛВС).

Правда, всі ці новини мало мають відносини до штучного інтелекту. Але в японському проекті передбачається інтелектуальний інтерфейс, який диктує вимоги до технічної частини нових ЕОМ, до їх елементної бази і архітектури. Головною метою фахівців Японії було забезпечення масового упровадження нових ЕОМ в повсякденну діяльність фахівців, що працюють в самих різних областях науки, техніки, економіки, політики і ін.

Багато програмних систем, що виникли ще в ”доінтелектуальну” епохи, знаходять нові якості, стають інтелектуальними. Виникають системи принципово нового типу.

Першими системами, націленими на автоматизацію інтелектуальних функцій, пов'язаних з діяльністю людей, були інформаційно-пошукові системи, які з'явилися значно раніше ЕОМ. В даний час інформаційно-довідкові системи стали інтелектуальними. Структура такої системи показана на рис.5. На ній видно вже відомий діалоговий процесор, за допомогою якого споживач ІПС

спілкується з ЕОМ. Оскільки запити споживача можуть бути класифіковані наперед по деяких певних типах, то діалоговий процесор на рис.5. більш простий, ніж той, який ми розглядали раніше. Він реалізує так званий регламентований діалог. При регламентованому діалозі запити споживача і відповіді системи кодуються у вигляді повідомлень із заданою структурою. Наприклад, споживач для формування запиту заповнює спеціальну анкету із стандартними графами. А система видає йому відповіді на заповнених конкретними значеннями стандартних бланках. При достатньо продуманій системі стандартизація форм запитів і відповідей спілкування між споживачем і системою може бути зовні надзвичайне схоже на діалог на природній мові. Ускладнюються і пошукові процедури, реалізовані в таких ІПС.

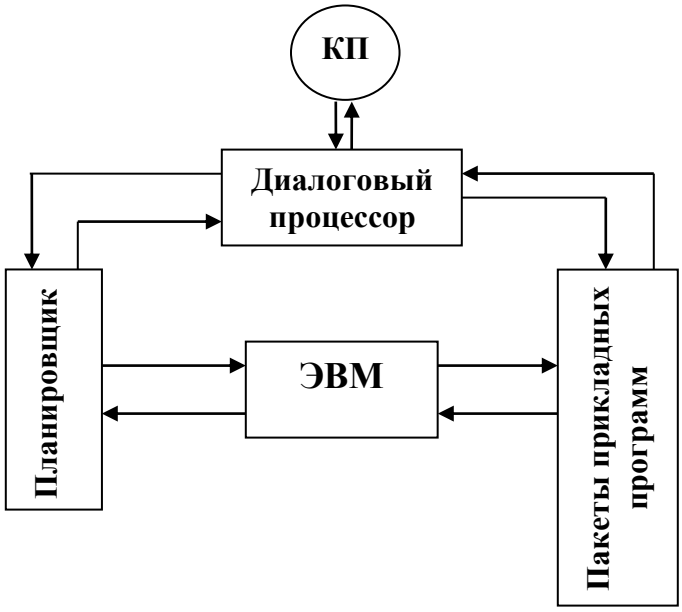


Рис. 5.

Система забезпечується базою знань про ту проблемну область або набір таких областей, інформація про факти і явища з яких складає інформаційний зміст системи. Системи подібного роду називають інтелектуальними інформаційно-пошуковими системами (ІПС).

Інший тип систем показаний на рис.6. Це так звані **інтелектуальні пакети прикладних програм (ІПП)**. Вони складаються з набору програм, що дозволяють вирішувати задачі з деякої фіксованої області, і спеціальної організуючої програми, що служить для виклику прикладних програм на вимогу користувача, постачання цих програм початковими даними, що задаються користувачем, і для організації взаємодії прикладних програм між собою через

загальні дані. В процесі діалогу користувач може одержати інформацію про склад прикладних програм, що є в пам'яті ЕОМ, про характеристики цих програм і про вимоги, які вони пред'являють до даних. У задачу користувача входить декомпозиція завдання в послідовність прикладних програм і «склеювання» їх за вхідними і вихідними даними. З'являється база знань, в якій міститься інформація про побудову математичних моделей, характерних для даної проблемної області. Це призводить до того, що доступ до пакетів прикладних програм для користувача стає можливим прямо з рівня опису задачі, що цікавить його, на мові тієї проблемної області, якої він займається.

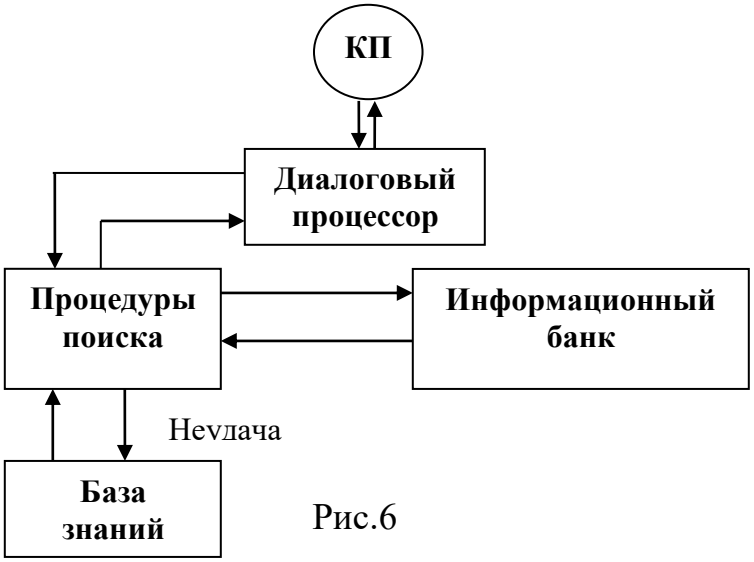


Рис.6

Подальшим розвитком ІППП є розрахунково-логічні системи (РЛС). Якщо в ІППП мають справу з одним (можливо груповим) користувачем, то в РЛС завжди є ціла група користувачів, спільно вирішальна загальну задачу, наприклад задачу планування або проектування. Крупні задачі такого рівня завжди розбиваються на ієрархічну сукупність локальних задач, вирішуваних окремими групами користувачів. Це пов'язано з тим, що знання, кваліфікація, відповідальність в організаційних системах завжди розподілені. В рамках рішення загальної задачі відбуваються багатократні ітерації як між фахівцями одного рангу, так і по ”вертикалі”, коли фахівці нижчого рангу свої результати з працівниками вищого рівня. Для організації такої взаємодії РЛС повинні мати спеціальні засоби. Вони входять до складу засобів планувальника, що бере на себе задачу обміну інформацією між ЕОМ, якими користуються фахівці, зайняті рішенням загальної задачі.

Останнім новим класом програмних систем, появі яких сприяли успіхи у області штучного інтелекту, є експертні системи. Загальна структура такої системи показана на рис.7.

З рис.7. видно, що до складу експертних систем входять вже знайомі нам блоки: **діалоговий процесор, база знань, планувальник**. Новим блоком є **підсистема пояснення**. Її поява пов'язана з характером експертних систем. Їх основною задачею є допомога фахівцям за рахунок використовування знань про проблемну область, одержаних з найрізноманітніших джерел: книг, статі, науково-технічної документації, експертів-фахівців і т. п., тобто в експертних системах зберігається колективний досвід, накопичений в даній проблемній області. Експертні системи виступають як експерти, коли фахівець звертається до них по допомогу. Задача експертної системи – пояснити користувачу незрозуміле для нього явище.

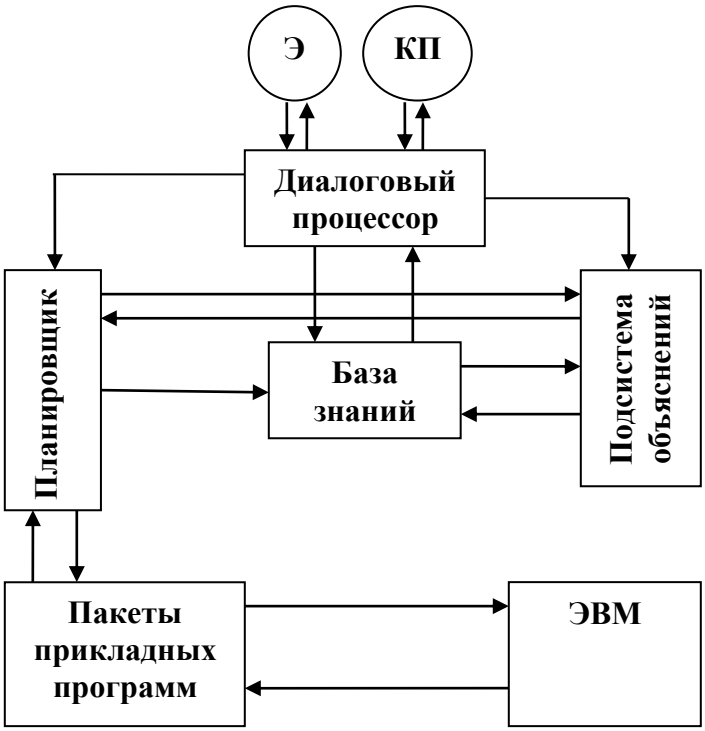


Рис.7.

Пошук інформації для консультації по запиту користувача здійснює планувальник. Але якщо фахівцю повідомити лише кінцевий результат пошуку (наприклад, передбачуваний діагноз захворювання або висновок про датування культурного шару), то він навряд цьому результату так просто повірить. Він повинен сам переконається в обгрунтованості висновків, одержаних на ЕОМ. Для розсіювання його сумнівів із цього приводу передбачена підсистема

пояснень. Вона стежить за роботою планувальника і описує його діяльність в стислій формі, зручній для аналізу людиною. Опускаючи опис не визначальних і неістотних кроків в роботі планувальника, підсистема чітко фіксує всі міркування, прийняті їм при альтернативних виборах.

На рис.7, окрім КП, показаний Э. У реальній системі це може бути і цілий колектив експертів або інформаційний вхід для введення текстів з книг і інших джерел, що зберігають потрібні відомості. Для обслуговування КП і Э діалоговий процесор в ЕС повинен мати різні засоби, оскільки і мови, якими користуються КП і Э, і їх цілі різні.

Лекція 4. Прикладні системи, орієнтовані на знання.

Популярність і значущість штучного інтелекту як основи нової інформаційної технології визначаються двома обставинами. Непрограмуючим фахівцям (кінцевим користувачам) забезпечується прямий доступ до сучасних обчислювальних комплексів для вирішення своїх задач в діалоговому режимі. У діалоговому режимі непрограмуючі користувачі можуть виробляти оцінку ситуацій і ухвалення рішень в математично слабо формалізованих галузях знань (медицина, біологія, геологія, хімія, суспільні науки, військова справа і т.п.).

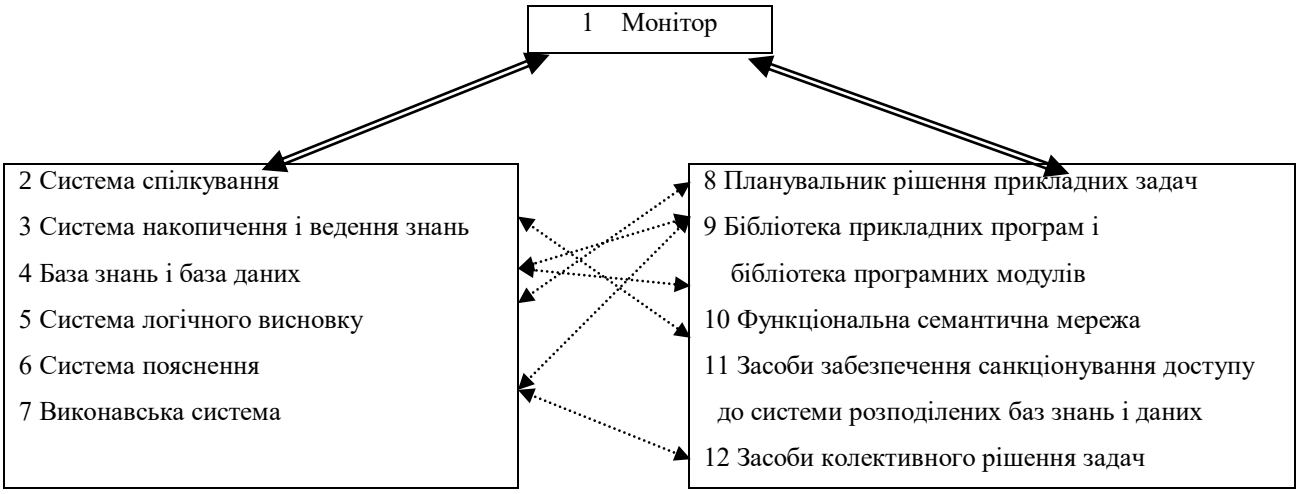


Рис.8.

1,2,3,4,5 – інтелектуальні ІПС; 1,2,4,8,9,10 – інтелектуальні ППП;
1,2,3,4,5,6,7 – ЕС; 1,2,4,8,9,10 – РЛС; 1,2,3,4,5,6,7,11 – розподілені ЕС;
1,2,3,4,5,6,7,12 – ЕС колективного користування;

1,2.3,4,5,6,7,8,9,10,11,12 – узагальнені прикладні інтелектуальні системи.

На рис.8. представлені основні функціональні підсистеми, характерні для всіх трьох типів прикладних інтелектуальних систем.

Ліва частина таблиці містить підсистеми, характерні для ЕС, а права – підсистеми, характерні для ІППП і РЛС. Пунктирні стрілки указують на відповідність підсистем в лівій і правій частинах таблиці. Об'єднання цих двох таблиць приводить до ЕС з сильно розвиненою обчислювальною компонентою або до РЛС з сильно розвиненою експертною компонентою.

Основні проблеми штучного інтелекту. Представлення знань

Термін «знання» в застосуванні до інформації з'явився не дуже давно (на ранньому етапі в ходу були терміни «програма» і «дані»). Для фахівців у області штучного інтелекту термін *знання* означає інформацію, яка необхідна програмі, щоб вона поводилася інтелектуально. Знання - це виявлені закономірності наочної області (принципи, зв'язки, закони), дозволяючі вирішувати задачі в цій області. Знання представлені в інтелектуальній системі, утворюють базу знань. Представлення знань – це угода про те, як описувати реальний мир. У природних і технічних науках прийнятий наступний традиційний спосіб представлення знань. На природній мові вводяться основні поняття і відносини між ними. Але при цьому використовуються раніше певні поняття і відносини, значення яких вже відоме. Далі встановлюється відповідність між характеристиками (найчастіше кількісними) понять знання і відповідною математичною моделлю. Основна мета представлення знань – будувати математичні моделі реального миру і його частин, для яких відповідність між системою понять проблемного знання може бути встановлене на основі збігу імен змінних моделі і імен понять без попередніх пояснень і встановлення додаткових неформальних відповідностей. Представлення знань звичайно виконується в рамках тієї або іншої системи представлення знань.

Системою представлення знань (СПЗ) називають засоби, що дозволяють описувати знання про наочну область за допомогою мови представлення знань, організувати зберігання знань в системі (накопичення, аналіз, узагальнення і організація структурованості знань), виводити нові знання і об'єднувати їх з тими, що є, виводити нові знання з тих, що є, знаходити необхідні знання, усувати застарілі знання, здійснювати інтерфейс між користувачем і знаннями.

Дані – це окремі факти, що характеризують об'єкти, процеси і явища в наочній області, а також їх властивості.

Можна вказати чотири причини, які дозволяють говорити не про дані, а про знання, використовувані в ЕОМ.

1. **Внутрішня інтерпретується.** Кожна інформаційна одиниця повинна мати унікальне ім'я, по якому інтелектуальна система (ІС) знаходить її, а також відповідає на запити, в яких це ім'я згадане. Коли дані, що зберігаються в пам'яті, були позбавлені імен, то відсутня можливість їх ідентифікації системою. Дані могла ідентифікувати лише програма, що витягує їх з пам'яті по вказівці програміста, що написав програму. Що ховається за тим або іншим двійковим кодом машинного слова, системі було невідомо. Для машини всі ці відомості однакові і однотипні. ЕОМ їх інтерпретує як двійкові числа, над якими скоюються ті або інші операції. Перший крок на шляху до знань полягає в тому, щоб повідомити ЕОМ ці відомості, дати їй можливість інтерпретувати дані і свої дії з ними на змістовному рівні.

2. Структурованість. Однією з фундаментальних особливостей людського пізнання навколишнього світу є його здібність до декомпозиції спостережуваних об'єктів, уміння виділяти в них окремі елементи і зв'язки між ними. Це дозволяє людині сприймати будь-який об'єкт як деяку структуру. Саме ця властивість дозволяє представляти об'єкти у вигляді сукупності простіших і пізнавати властивості нового об'єкту через властивості тих, що входять в його структуру. Інформаційні одиниці повинні були володіти гнучкою структурою. Для них повинен виконуватися “принцип матрьошки”, тобто рекурсивна вкладеність одних інформаційних одиниць в інші. Кожна інформаційна одиниця може бути включена до складу будь-якої іншій, і з кожної одиниці можна виділити деякі її складові. Іншими словами повинна існувати можливість довільного встановлення між окремими інформаційними одиницями відносин типу “частина – ціле”," рід – вигляд” або “елемент – клас”.

Властивість структурованості відображається у тому, що відповідні поняття виявляються сполученими між собою зв'язками типу «частина-ціле», «клас-елемент», «рід-вигляд» і т.п. Такі зв'язки утворюють в нашій пам'яті ієрархії понять і дозволяють вводити нові поняття, об'єднуючі в собі безліч понять, відповідних реальним об'єктам зовнішнього світу. Так виникають поняття типу «дерево», «ссавець», «меблі». Знання, уявні в ЕОМ, повинні володіти такою ж здібністю до утворення ієрархій, введення нових узагальнених понять на становлячі їх поняття і відносини між ними. Зробити це

можна різним і способами. Один з них – явна вказівка зберігаються в базі інформації тих її одиниць, які є для них видовими. На рис.9 показано, як організовується такий взаємозв'язок. Опис кожної інформаційної одиниці на цьому малюнку складається з двох частин: інформації, що відноситься до даної одиниці, і посилань на родові одиниці (видові одиниці).

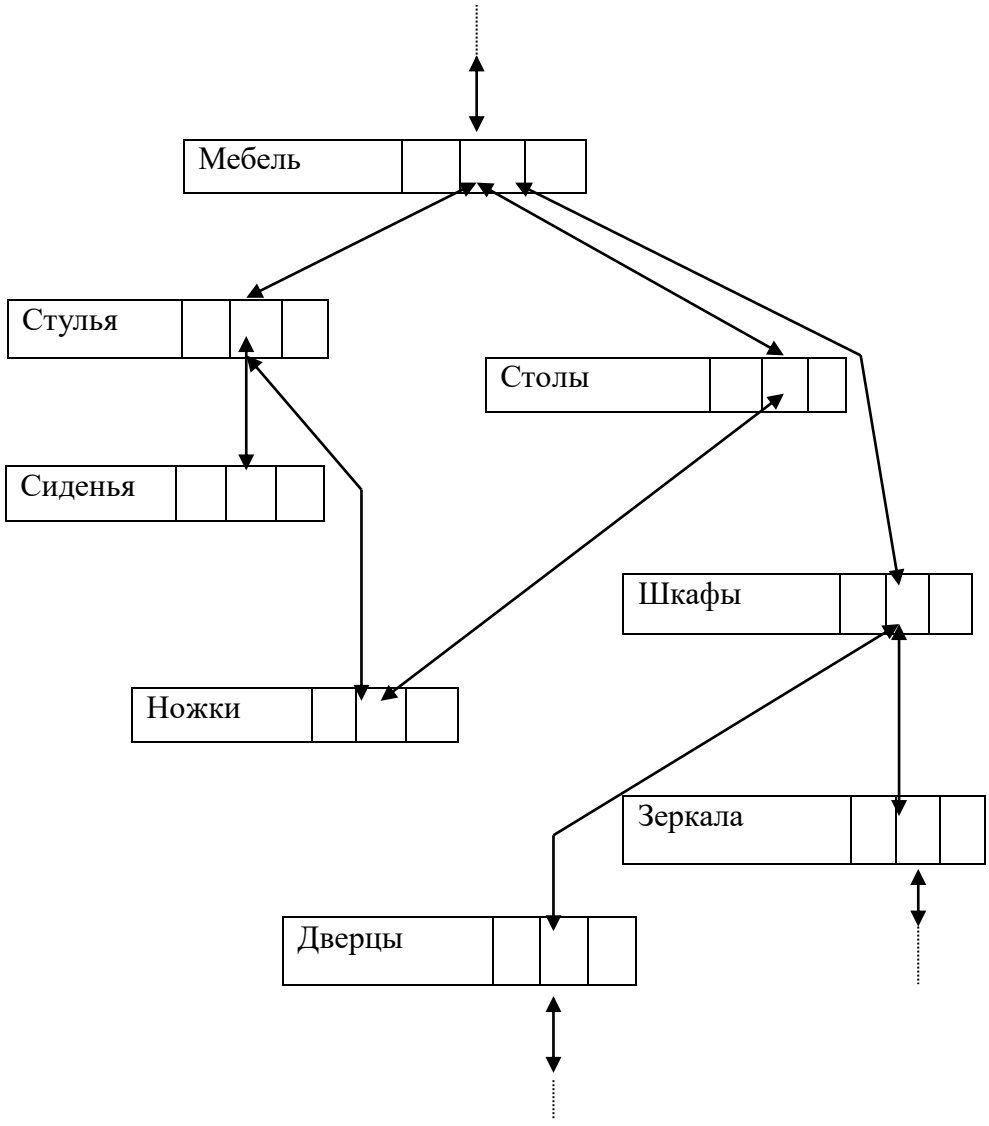


Рис.9.

Важливою особливістю такої організації є можливість одноразового запису інформації. Наприклад, вся інформація, загальна для всіх стільців, записана тільки один раз в одиниці інформації, названої стільці, а вся інформація, загальна для всіх меблів, записана також тільки один раз в одиниці

меблі і не повторюється в одиницях стільці, столи, шафи і т.д. При необхідності отримання на даному рівні деталізації всієї інформації, що відноситься до вибраної інформаційної одиниці, потрібно пройти від неї вгору до самої останню в ієрархії родової одиниці і зібрати на цьому шляху всю бракуючу інформацію. Допускаються і такі зв'язки, при яких видова одиниця пов'язана з декількома родовими одиницями. Така ситуація в нашому прикладі має місце для інформаційної одиниці ніжка.

3. Зв'язаність. Особливе значення мають для нас ті зв'язки між знаннями, які встановлюють закономірності між окремими одиницями інформації – фактами, процесами, явищами, а також визначають структуру тієї ситуації, в рамках якої спостерігається суть, якій відповідають ці інформаційні одиниці. У інформаційній базі між інформаційними одиницями повинна бути передбачена можливість встановлення зв'язків різного типу. Перш за все, ці зв'язки можуть характеризувати відносини між інформаційними одиницями. Наприклад: дві або більш інформаційні одиниці можуть бути зв'язані відношенням «одночасно», дві інформаційні одиниці - відношенням «причина – слідство» або відношенням «бути поряд».

Пояснимо це на прикладі опису такої ситуації: «Серед саду в тіні розлогих кленів, стояла старенька сіренька альтанка – три сходинки вгору, замшілий поміст, низенькі стіни, шість точених пузатих стовпів і шестискатна кровелька. Марта сиділа в альтанці…». Виділимо в цій ситуації основні одиниці інформації і встановимо між ними зв'язки. Тут є цілий комплекс одиниць, зв'язаних між собою родовою одиницею «альтанка». Відносно частина-ціле до неї знаходяться «сходинки», «поміст», «стіни», «стовпи» і «кровелька». Окрім цих одиниць, можна виділити наступні: «сад». «тінь», «клени», «Марта». Ці одиниці зв'язані між собою і з одиницею «альтанка» зв'язками іншого типу, ніж родо-видовий зв'язок. Ці зв'язки носять ситуативний характер. Наприклад, між «Мартою» і «альтанкою» є просторовий зв'язок типу «знаходитися в», а між «кленами» і «тінню» – зв'язок «причина – слідство».

Повна структура взаємозв'язків показана на рис.10. За умовчанням між всіма вершинами в цій мережі є також відношення «бути одночасно».

Приведені відносини характеризують декларативні знання. Якщо між двома інформаційними одиницями встановлено відношення «аргумент – функція», то він характеризує процедурне знання, пов'язане з обчисленням певних функцій. Існують - відносини структуризації, функціональні відносини, каузальні відносини і семантичні відносини. За допомогою перших задаються

ієрархії інформаційних одиниць, другі несуть процедурну інформацію, дозволяючі обчислювати (знаходити) одні інформаційні одиниці через інші, треті задають причинно слідчі зв'язки, четверті відповідають всій решті відносин.

Перераховані три особливості знань дозволяють ввести загальну модель представлення знань, яку можна назвати семантичною мережею, що є ієрархічною мережею у вершинах якої знаходяться інформаційні одиниці.

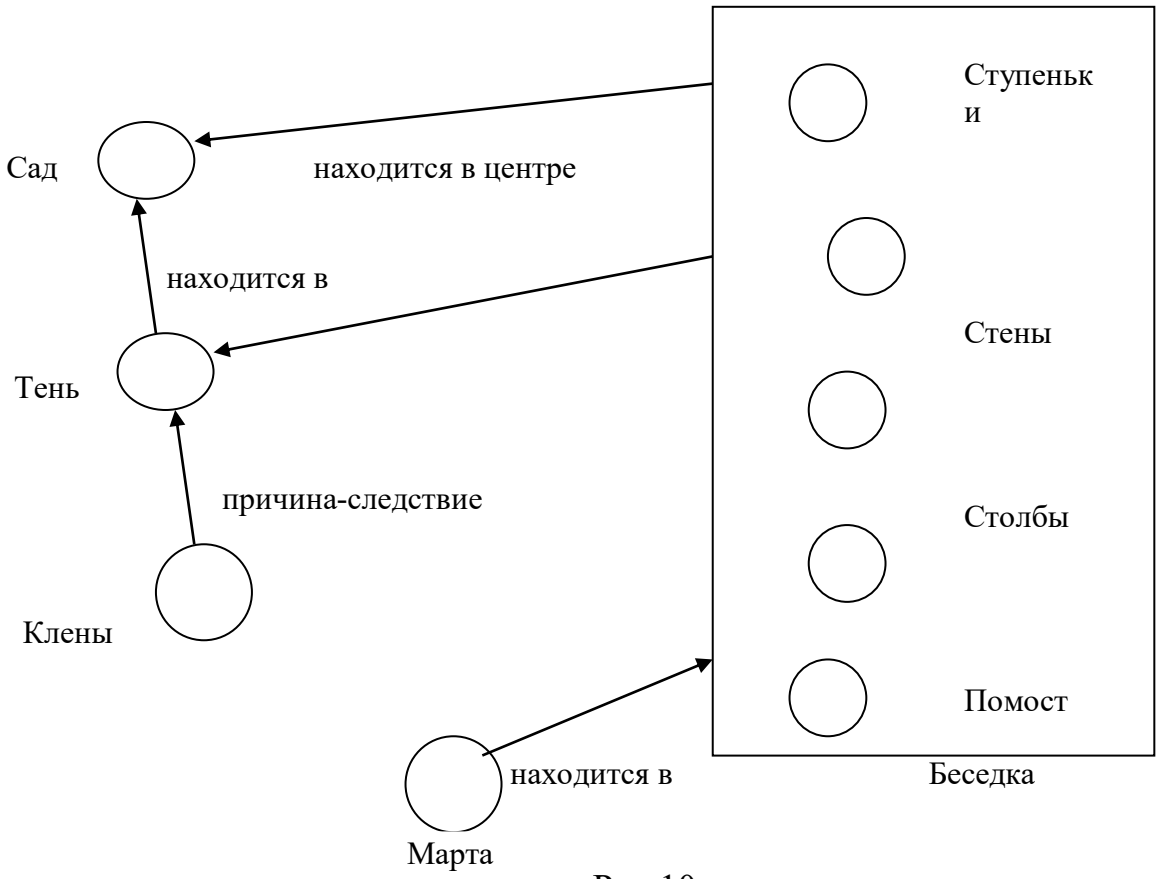


Рис.10

4. **Активність.** Четвертою важливою характеристикою того, що можна вважати справжніми знаннями, є їх активність. При роботі з ЕОМ активними вважалися програми, тобто процедури, а дані в ЕОМ опинялися пасивними (вони просто зберігалися в пам'яті і чекали, коли їх витягнуть). У людини активними виявляються саме знання, які формують цілі його діяльності. Активізовані цілі викликають реалізацію тих процедур, які можуть виявитися корисними для досягнення поставлених цілей.

Неповнота знань примушує людину шукати шляху їх поповнення, і суперечність в знаннях – прагнення до подолання її. При створенні інтелектуальних систем необхідно забезпечити відтворення цієї найважливішої властивості «бази знань» людини. Є багато способів імітації цієї характеристики знань в штучних системах. Розглянемо один з них. На мал. 11,а показані три інформаційні одиниці, що зберігаються в базі знань. Вони зв'язані між собою особливими оцінними зв'язками.

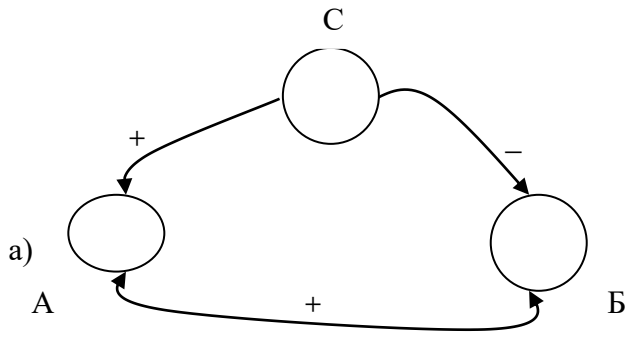


Рис.11

Вершина, відповідна самій системі (С), має позитивну оцінку користувача А. Відносно користувача б у системи думка інша, воно носить негативний характер. Хай в результаті опиту система взнала, що користувачі А і б високої думки один про одного. Цієї інформації, одержаної системою, відповідає позитивний зв'язок між користувачами А і б.

Якщо штучна система поводиться подібно людині, то структура у вигляді трикутника, повинна активізувати систему, оскільки подібна структура характеризує дисонанс в її знаннях, а всякий дисонанс вимагає свого дозволу. Як і людини, у системи є чотири можливості перетворення наявного дисонансу в консонанс.

Перша полягає в зміні своєї думки про користувача Б. Можливо, система не зовсім правильно оцінила результати взаємодії з ним, і необхідно зібрати про користувача б додаткову інформацію. Для цього слід порушити відповідні програми її збору. Якщо після їх роботи система змінить свою думку про користувача б, то вміст тієї зони її бази знань стане таким, як це показано на рис. 12 б. Тепер виконаний принцип: «друзі моїх друзів – мої друзі», і система зі стану дісонансу приходить у стан, який її повністю здовільняє. Друга можливість містить в зміненні своєї уяви про користувача А. Адже система могла і помилитися, приписавши йому позитивну характеристику. Знову

включаються процедури поповнення знань про користувачів. Тепер адресатом їх є користувач А. Якщо після додаткової перевірки зона бази знань стає такою, як це показано на рис.12 в, то дисонанс знову ліквідований. Зрозуміло, що А і б добре відносяться один до одного, але обидва вони з погляду системи не дуже хороші. Третя можливість – порушити процедури в користувачах А і б по перевірці відносин між собою. Наприклад, повідомити цим користувачам (або одному з них, наприклад А, до якого система добре відноситься) деяку додаткову, має в своєму розпорядженні систему інформацію про них. Якщо після цього користувачі (принаймні А) змінять думку один про одного і трикутник прийме вигляд, показаний на мал. 12г, то система знов буде задоволена.

На її думку «справедливість восторжествувала» і дисонанс ліквідований. Якщо ж всі три спроби не увінчалися успіхом – трикутник залишився таким, як він показаний на мал. а, - то у системи залишається в запасі єдина процедура – витіснення дисонансного знання з активного поля уваги.

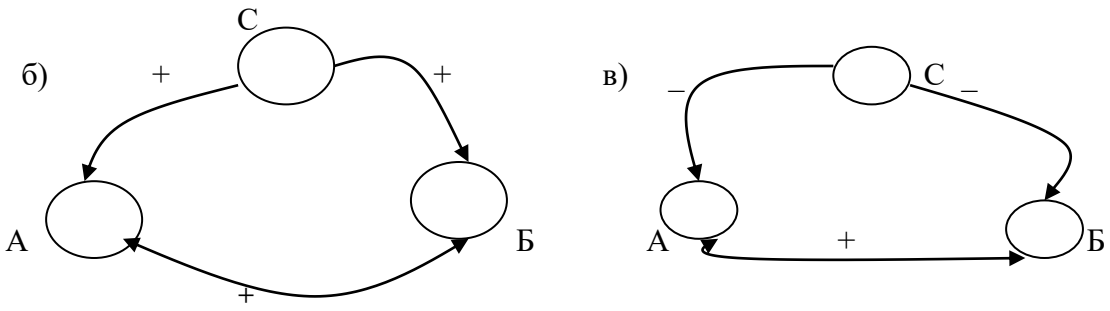


Рис.12

Можливо, що з часом з'явиться нова інформація, поки недоступна системі, яка і дозволить ліквідовувати цей неприємний фрагмент в знаннях системи.

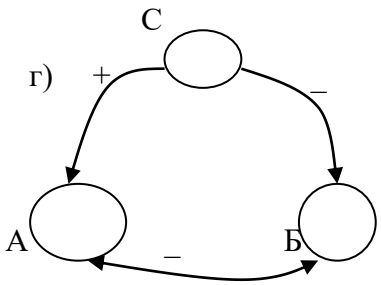


Рис.11

Лекція 5. Моделі представлення знань

Однієї з найважливіших проблем, характерних для систем, заснованих на знаннях, є проблема представлення знань. Це пояснюється тим, що форма представлення знань робить істотний вплив на характеристики і властивості системи. Для того, щоб маніпулювати всілякими знаннями з реального світу за допомогою комп'ютера, необхідно здійснювати їх моделювання. У таких випадках необхідно відрізняти знання, призначені для обробки комп'ютером, від знань, використовуваних людиною.

Для зберігання даних використовуються **бази даних** (для них характерні великий обсяг і відносно невелика питома вартість інформації), для зберігання знань – бази знань (невеликого обсягу, але виключно дорогі інформаційні масиви).

База знань – основа будь-якої інтелектуальної системи, де знання описані деякою мовою представлення знань, наближеній до природного. Сьогодні знання набули чисто декларативної форми, тобто знаннями вважаються речення, записані мовами подання знань, наближеними до природної мови і зрозумілих неспеціалістам.

Сукупність знань, потрібних для прийняття рішень, прийнято називати предметною областю або знаннями про предметну область. У будь-якій предметній області є свої поняття і зв'язки між ними, своя термінологія, свої закони, що пов'язують між собою об'єкти даних предметної області, свої процеси і події. Крім того, кожна предметна область має свої методи вирішення завдань. Вирішуючи завдання такого виду на ЕОМ, використовують інформаційні системи, ядром яких є база знань, що містить основні характеристики предметних областей.

Бази знань базуються на моделях представлення знань, подібно до баз даних, які засновані на моделях представлення даних (ієрархічній, мережевій, реляційній, післяреляційної тощо).

При поданні знань у пам'ять інтелектуальної системи традиційні мови, засновані на чисельному поданні даних, є неефективними. Для цього використовуються спеціальні мови представлення знань, засновані на символьному поданні даних. Вони діляться на типи за формальними моделям представлення знань. Найбільш часто використовується на практиці класифікація моделей подання знань, наведена на рис.1, де моделі представлення знань діляться на детерміновані (жорсткі) і м'які.

Детерміновані моделі включають в себе фрейми, логіко-алгебраїчні моделі, семантичні мережі і продукційні моделі. М'які моделі включають у себе нечіткі системи, нейронні мережі, еволюційні моделі, гібридні системи.

З моделюванням знань безпосередньо пов'язана проблема вибору мови уявлення. З метою класифікації моделей уявлення знань виділяється дев'ять ключових вимог до моделей знань:

1) спільність (універсальність);
2) наочність представлення знань;
3) однорідність;
4) реалізація в моделі властивості активності знань;
5) відкритість;
6) можливість відображення структурних відносин об'єктів предметної області;
7) наявність механізму «проектування» знань на систему семантичних шкал;
8) можливість оперування нечіткими знаннями;
9) використання багаторівневих уявлень (дані, моделі, метамоделі, метаметамоделі і т.і.).

Моделі подання знань не задовольняють повністю цим вимогам, що і пояснює їх різноманіття й активний розвиток даного напрямку.

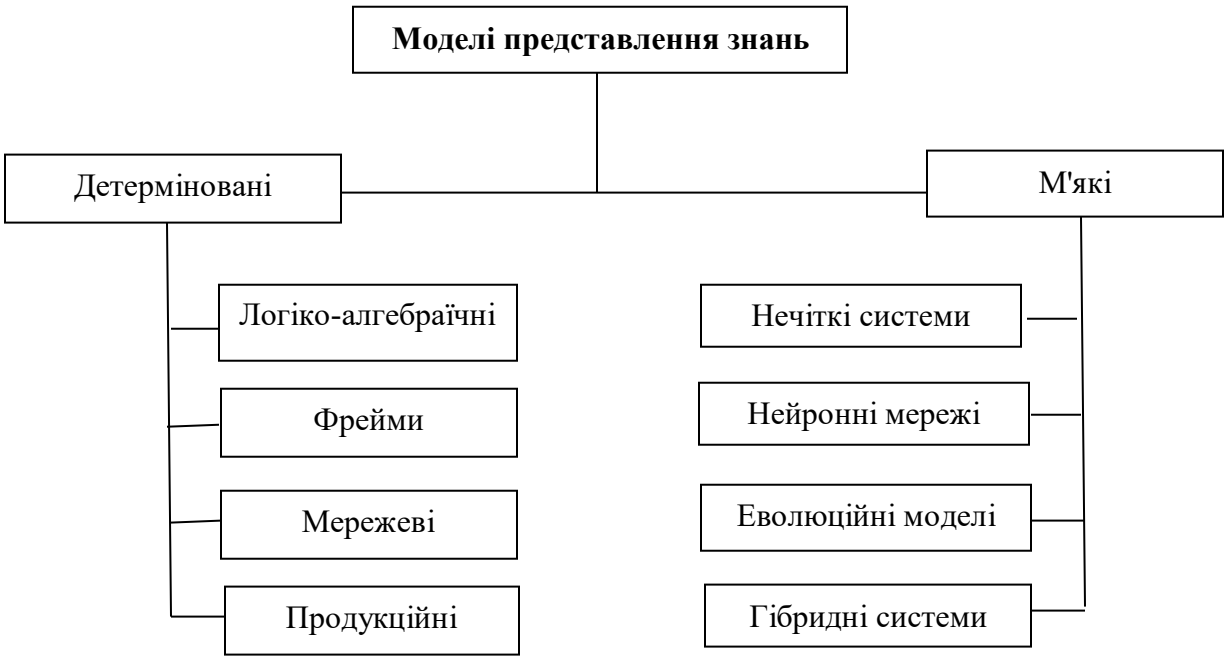


Рис. 13. Моделі представлення знань

Найбільш поширені три моделі представлення знань (таблиця 1): фреймова, продукційна і семантична. Вибір методу представлення знань залежить від особливостей предметної області (які структури знань найбільш часто зустрічаються, чи присутні ієрархічність або мережеві конструкції, характер вхідних і вихідних даних у задачах і т.і.), досвіду когнітолога, обраного інструментарію розробки.

Таблиця 1 . Основні моделі подання знань, що використовуються на практиці

Модель	Переваги	Недоліки
Продукції	Наочність, висока модульність, легкість внесення доповнень і змін, простота механізму логічного виведення, простота інтерпретації.	При накопиченні великої кількості (декількох сотень) продукцій вони починають суперечити одна одній, виникають труднощі при додаванні правил, що залежать від вже наявних в базі знань, відсутній цілісний образ знань, неясність взаємозв'язків між правилами.
Семантичні мережі	Наочність, відповідає уявленням про організацію довготривалої пам'яті людини, дозволяє знизити обсяг збережених даних.	Являють собою пасивні структури, для обробки яких необхідний спеціальний апарат формального виводу і планування, довільна структура і різні типи вершин і зв'язків ускладнюють процедуру обробки інформації, мережева модель не дає чіткого уявлення про структуру предметної області.
Фрейми	Гнучкість, наочність, зручний спосіб включення процедурних знань, зводиться до інших моделей, модульність	Відсутність універсальної процедури управління виводу крім механізму наслідування, є ідеологічною концепцією.

Продукційна модель.

Продукція – це пропозиція-зразок виду «Якщо, то», за яким здійснюється пошук у базі знань. В продукції виділяють ліву частину (починається з «якщо» і закінчується перед «то») і праву (починається після «то»). Ліва частина продукції – антецедент – умова виконання правої частина продукції. Права частина – консеквент – дія, що виконується в разі знаходження елементів, що задовольняють лівій частині. Дія може бути проміжною і

виступати потім в якості консеквента або цільовою, завершальною процедуру виведення.

Антецедент формується з фактів, вхідних даних задачі і логічних зв'язок (і, або, не). Консеквент може представляти із собою дію зі зміни фактів, даних, рекомендацію, рішення задачі. Крім цього, будь-яка продукція має ім'я і пріоритет, який визначає послідовність перевірки продукцій машиною виведення.

Продукції відображають причинно-наслідкові зв'язки, які і дозволяють людині приймати рішення, базуючись на знаннях і припущеннях про те, що є і що буде, якщо щось зробити.

При використовуванні продукційної моделі база знань складається з набору правил. Програма, що управляє перебором правив, називається машиною висновку. Найчастіший висновок буває прямий (від даних до пошуку мети) або зворотний (від мети для її підтвердження - до даних). Дані - це початкові факти, на підставі яких запускається машина висновку - програма, що перебирає правила з бази.

Прикладом продукції може служити наступний вираз:
ЯКЩО клієнт працює на одному місці більше двох років, **ТО** клієнт має постійну роботу.

Як умова, так і дія правила можуть враховувати декілька виразів, об'єднаних логічними зв'язками І, АБО, НЕ:
ЯКЩО клієнт має постійну роботу **І** клієнту більше 18 років **І** клієнт не має фінансових зобов'язань, **ТО** клієнт може претендувати на отримання кредиту.

Крім продукційних правил база знань повинна включати і прості факти, що поступають в систему через інтерфейс користувача або рішення задачі, що виводяться в процесі пошуку. Факти є простими затвердженнями типу «клієнт працює на одному місці більше двох років». І коли в процесі інтерпретації правил машиною висновку який-небудь факт узгоджується з частиною правила ЯКЩО, то виконується дія, визначувана частиною ТЕ цього правила. Нові факти, що додаються в базу знань в результаті дій, описаних в правилах, також можуть бути використані для зіставлення з частинами ЯКЩО інших правил. Послідовне зіставлення частин правив ЯКЩО з фактами породжує ланцюжок висновку. Ланцюжок висновку, одержана в результаті послідовного виконання правил П1 і П2 показана нижче (див. мал. 12. Приклад ланцюжка висновку.). Цей ланцюжок показує, як на підставі правил і початкових фактів виводиться висновок про можливість отримання кредиту. Ланцюжки висновку експертної

системи можуть бути пред'явлені користувачу і допомагають зрозуміти, як було одержане рішення.

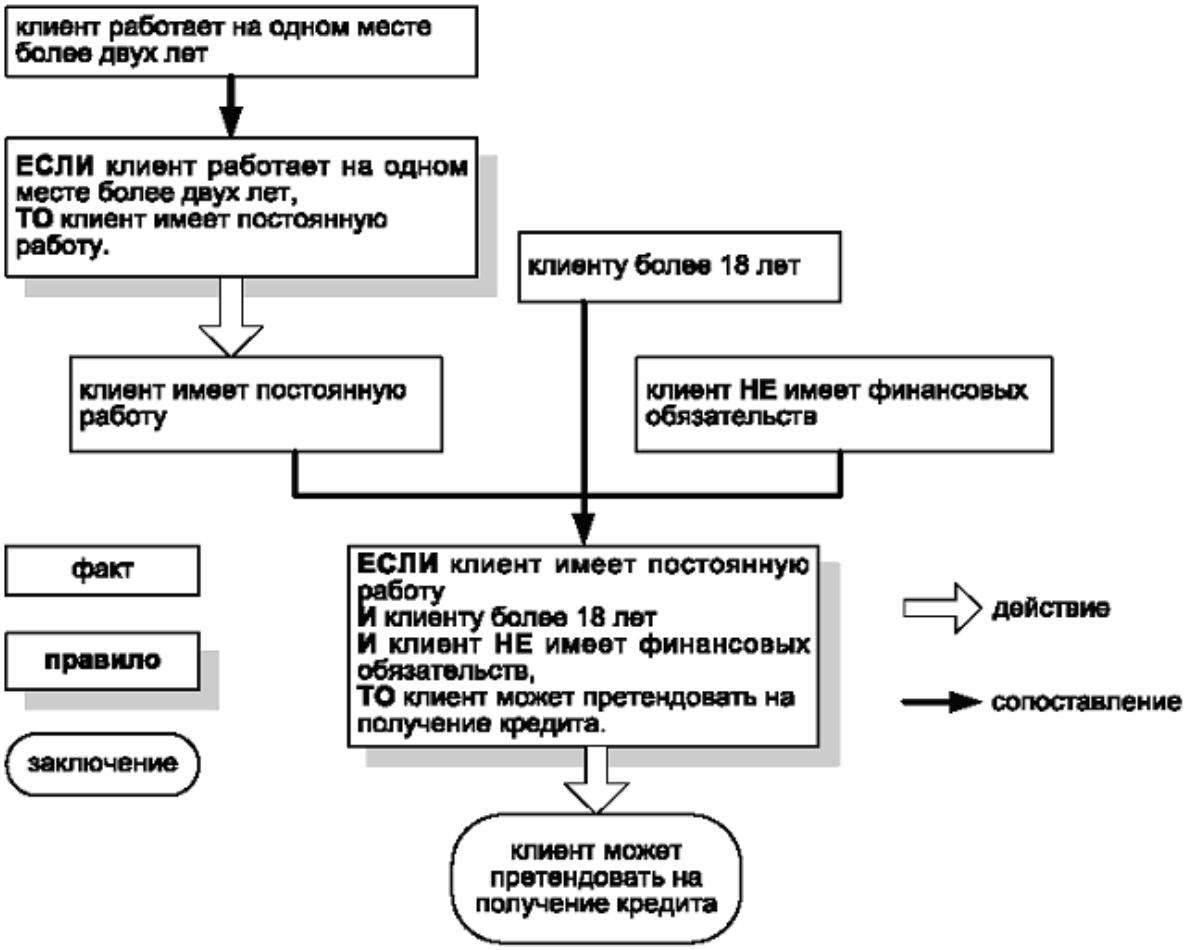


Рис. 12. Приклад ланцюжка висновку

Представлення знань у вигляді продукції найбільш поширено в експертних системах, оскільки запис знань фактично ведеться на підмножині природної мови. Продукційна модель найчастіше застосовується в промислових експертних системах, які називають продукційними. Вона привертає розробників своєю наочністю, високим модулем, легкістю внесення доповнень і змін і простотою механізму логічного висновку.

Приклад рішення задачі. Побудувати продукційну модель подання знань у предметній області «Ресторан» (відвідування ресторану). Опис процесу рішення. Для побудови продукційної моделі подання знань необхідно виконати наступні кроки:

1) Визначити цільові дії завдання (що є рішеннями).

2) Визначити проміжні дії або ланцюжок дій між початковим станом і кінцевим (між тим, що є, і цільовим дією).

3) Визначити умови для кожної дії, при якій їх доцільно і можливо виконати. Визначити порядок виконання дій.

4) Додати конкретики при необхідності, виходячи з поставленого завдання.

5) Перетворити отриманий порядок дій і відповідні їм умови в продукції.

6) Для перевірки правильності побудови продукцій записати ланцюжок продукцій, явно простеживши зв'язок між ними. Цей набір кроків передбачає рух при побудові продукційної моделі від результату до початкового стану, але можливий і рух від початкового стану до результату (кроки 1 і 2).

Рішення.

1). Обов'язкова дія, яке виконується в ресторанах – поглинання їжі та її оплата. Значить, є вже дві цільових дії «з'їсти їжу» й «оплатити», які взаємопов'язані і слідують одна за одною.

2). Перш ніж що-небудь з'їсти в ресторані, туди потрібно прийти, дочекатися офіціанта і зробити замовлення. Крім того, потрібно вибрати, до якого саме ресторану піти. Значить, ланцюжок проміжних дій: «вибір ресторану і шлях туди», «зробити замовлення офіціантові».

3). Перш ніж йти до ресторану, необхідно переконатися, що є необхідна сума грошей. Вибір ресторану може обумовлюватися багатьма причинами, виберемо територіальну ознаку – до якого ближче, до того і йдемо. У різних ресторанах працюють різні люди, тому, залежно від вибору ресторану, офіціанти будуть різні. Крім того, різні ресторани спеціалізуються на різних кухнях, тому замовлені страви будуть у різних ресторанах відрізнятися. Значить спочатку йдуть дії, що дозволяють вибрати ресторан, потім характеризують ресторани, а вже після замовлення – їжа й оплата замовлення.

4). Нехай в задачі будуть розглядатися два ресторани: «Смачна їжа» і «Смакота». Перший – паб і замовлення приносять швидше, ніж у другому, другий – піцерія. У першому працює офіціант Сергій, а в другому – офіціантка Марина. Петро – це клієнт.

5). Вище описане можна перетворити в наступні пропозиції типу «Якщо, то»:

- якщо суб'єкт хоче їсти й у суб'єкта є достатня сума грошей, то суб'єкт може піти до ресторану;

- якщо суб'єкт ближче до ресторану «Смачна їжа», ніж до ресторану «Смакота» і суб'єкт може піти до ресторану, то суб'єкт йде до ресторану «Смачна їжа»;

- якщо суб'єкт ближче до ресторану «Смакота», ніж до ресторану «Смачна їжа» і суб'єкт може піти до ресторану, то суб'єкт йде до ресторану «Смакота»;

- якщо суб'єкт йде до ресторану «Смакота» і в ресторані «Смакота» працює офіціант Марина, то у суб'єкта приймає замовлення Марина;

- якщо суб'єкт йде до ресторану «Смачна їжа» і в ресторані «Смачна їжа» працює офіціант Сергій, то у суб'єкта приймає замовлення Сергій;

- якщо суб'єкт вибрав страви й у суб'єкта приймає замовлення Марина, то замовлення принесуть через 20 хв.;

- якщо суб'єкт вибрав страви й у суб'єкта приймає замовлення Сергій, то замовлення принесуть через 10 хв.;

- якщо замовлення принесуть через 20 хв. або замовлення принесуть через 10 хв., то суб'єкт може їсти;

- якщо суб'єкт може їсти, то після їжі суб'єкт повинен оплатити замовлення.

Введемо позначення для фактів (Ф), дій (Д) і продукцій (П), тоді:

Суб'єкт = Петро;

Ф1 = суб'єкт хоче їсти;

Ф2 = у суб'єкта є достатня сума грошей;

Ф3 = суб'єкт ближче до ресторану «Смачна їжа», ніж до «Смакота»;

Ф4 = в ресторані «Смакота» працює офіціант Марина;

Ф5 = в ресторані «Смачна їжа» працює офіціант Сергій;

Ф6 = суб'єкт вибрав страви;

Д1 = суб'єкт може піти до ресторану;

Д2 = суб'єкт йде до ресторану «Смачна їжа»;

Д3 = суб'єкт йде до ресторану «Смакота»;

Д4 = у суб'єкта приймає замовлення Марина;

Д5 = у суб'єкта приймає замовлення Сергій;

Д6 = замовлення принесуть через 20 хв.

Д7 = замовлення принесуть через 10 хв.

Д8 = після їжі суб'єкт повинен оплатити замовлення.

Для продукцій встановимо пріоритет (в дужках перед комою, чим вище пріоритет, ніж раніше перевіряється правило).

П1 (4, Ф1 і Ф2) = Д1;

П2 (5, Ф3 і Д1) = Д2;
П3 (4, що не Ф3 і Д1) = Д3;
П4 (3, Д3 і Ф4) = Д4;
П5 (3, Д2 і Ф5) = Д5;
П6 (2, Д4) = Д6;
П7 (2, Д5) = Д7;
П8 (1, Д6 або Д7) = Д8;
6) Для відображення взаємозв'язку продукцій побудуємо граф (рис. 13).

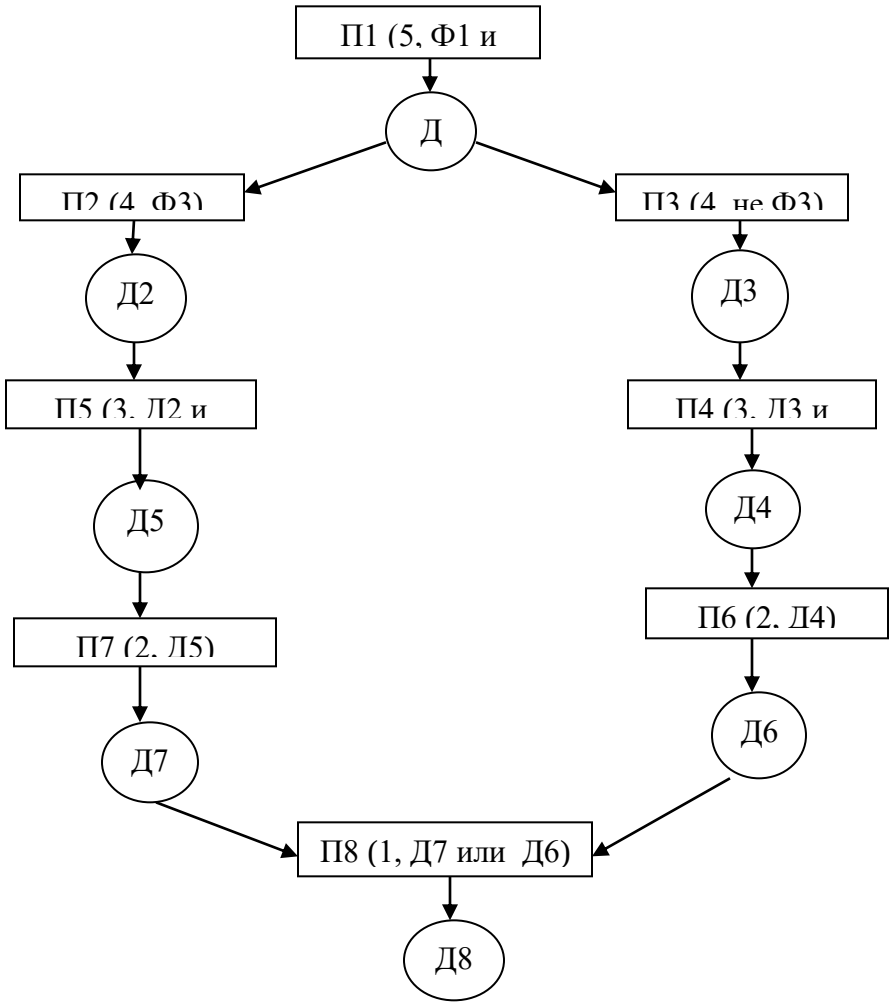


Рис. 13. Схема продукцій предметної області «Ресторан»

Лекція 6. Модель семантичної мережі.

Одним із способів представлення знань є семантична мережа. Семантична мережа – це орієнтований граф, вершини якого – поняття, а дуги – відносини між ними. Вузли у семантичної мережі зазвичай відповідають об'єктам, концепціям, подій або поняттям. Будь-фрагмент мережі, наприклад одна вершина, дві вершини і дуги, що з'єднують їх, називають під сіткою. Логічний висновок (пошук рішення) на семантичній мережі полягає в тому, щоб знайти або сконструювати підмережу, що задовольняє деяким умовам.

Спочатку семантична мережа була задумана як модель кончини структури довготривалої пам'яті в психології, але надалі стала одним з основних способів представлення знань в інженерії знань.

Таблиця 2. Основні види відносин в семантичних мережах.

Тип	Опис
Бути спадкоємцем (a-kind-of)	Задає ієрархічні зв'язки між класами
Бути екземпляром	Визначає значення, описує конкретний об'єкт, поняття
Це (are, есть)	Може використовуватися замість зв'язку a-kind-of у відносинах, що передбачають рівність або еквівалентність
Бути частиною (has-part)	Визначає структурні зв'язки, описує частини або цілі об'єкти
Функциональні	Визначаються зазвичай дієсловами, відображають різні відносини (вчити, володіти і т.і.).
Кількісні	Відображають кількісні співвідношення між вершинами (більше, менше і т.і.)
Просторові	Відображають просторові відносини між вершинами (близько, далеко і т.і.)
Тимчасові	Описують тимчасові зв'язки між вершинами (скоро, довго, зараз тощо)
Атрибутивні	Описують властивості об'єктів, понять
Логічні	Описують логічні зв'язки між вершинами (і, або, не)

У основі мережевих моделей представлення знань лежить ідея про те, що будь-які знання можна представити у вигляді сукупності об'єктів (понять) і зв'язків (відносин) між ними. На відміну від продукційних ці моделі більш наочні, оскільки будь-який приклад можна представити у вигляді орієнтованого (направленого) графа, вершини якого - поняття, а дуги - відносини між ними.

Відносини, що подаються дугами, у семантичній мережі можуть бути різними (таблиця 2). Типи відносин вибираються залежно від виду семантичної мережі (таблиця 3) і розв'язуваної задачі.

Зрозуміло, що зазвичай виступають абстрактні або конкретні об'єкти, а відносини, що з'єднують ці зв'язки, типу: «це» (АКО – A-Kind-Of, is або «елемент класу»), «має частиною» (has part), «належить», «любить» тощо.

Таблиця 3 –Типи семантичних мереж.

Тип	Опис
За типом знання	
Екстенсиональні	Описує конкретні відносини даної ситуації..
Інтенсіональні	Описують імена класів об'єктів, а не індивідуальні імена об'єктів, зв'язки відображають ті відносини, які завжди притаманні об'єктам даного класу.
За типом обмежень на дуги і вершини	
Прості	Вершини мережі не мають внутрішньої структури
Ієрархічні	Вершини мають внутрішню структуру, в ієрархічній мережі є можливість розділяти мережу на підмережі і встановлювати відносини не тільки між вершинами, а й між підмережами (різні підмережі, існуючі в мережі, можуть бути впорядковані у вигляді дерева підмереж, вершини якого – підмережі, а дуги – відносини видимості)
Динамічні (сценарії)	Мережі з подіями
За кількістю типів відносин	
Однорідні	Мають тільки одним типом відносин
Неоднорідні	Кількість типів відносин більше двох
За арності відносин	
Бінарні	Всі відносини в графі пов'язують рівно два поняття
N-арні	У мережі є відносини, що зв'язують більше двох об'єктів

Поняттями звичайно виступають абстрактні або конкретні об'єкти, а відносини - це зв'язки типу: "це" ("is"), "має частиною" ("has part"), "належить", "любить" і т.п. Характерною особливістю семантичних мереж є обов'язкова наявність трьох типів відносин: клас - елемент класу; властивість – значення; приклад елементу класу.

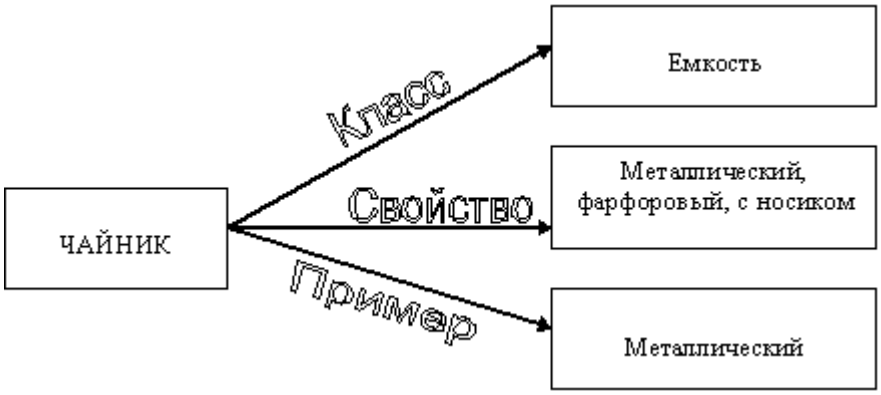


Рис. 14.

Як приклад на рис.14 показана вельми проста семантична мережа для представлення об'єкту «чайник».

Найчастіше в семантичних мережах використовуються наступні відносини:

зв'язки типу "частина-ціле" ("клас-підклас", "елемент-множина" і т.п.); функціональні зв'язки (визначувані звичайно дієсловами "виробляє", "впливає"...); кількісні (більше, менше, рівно...); просторові (далеко від, близько від, за, під, над...); тимчасові (раніше, пізніше, в течію...); атрибутивні зв'язки (мати властивість, мати значення...); логічні зв'язки (і, або, не) і ін.

Можна ввести декілька класифікацій семантичних мереж.

Наприклад, по кількості типів відносин: **однорідні** (з єдиним типом відносин); **неоднорідні** (з різними типами відносин).

По типах відносин: **бінарні** (у яких відносини зв'язують два об'єкти); **парні** (у яких є спеціальні відносини, що зв'язують більше двох понять).

Проблема пошуку рішення в базі знань типу семантичної мережі зводиться до задачі пошуку фрагмента мережі, відповідного деякій підмережі, відповідній поставленому питанню. Подібного роду задачі розв'язуються за допомогою апарату теорії графів. Слід особливо відзначити роль фундаментальних ознак зв'язків (рефлексія, симетричність і транзитивність) в процесі висновку на мережі. Так, наприклад, зв'язки вигляду "це є" або "мати частиною" транзитивні, що дозволяє говорити про встановлення за допомогою цього зв'язку властивостей ієрархії спадкоємства в мережі. Це означає, що елементи нижчого рівня в мережі можуть успадковувати властивості елементів вищого рівня в мережі. У найзагальнішому вигляді семантична мережа є орієнтований граф, в якій вершини відповідають певним об'єктам або поняттям, а дуги відображають ті відносини, які є між вершинами. Базовим функціональним

елементом семантичної мережі служить структура з двох компонентів – «вузлів» і зв'язуючих їх «дуг». Кожен вузол представляє деяке поняття, а дуга – відношення між парами понять. Дуга має спрямованість, завдяки чому між поняттями в рамках певного факту виражається відношення «суб'єкт/об'єкт». Більш того, будь-який з вузлів може бути сполучений з будь-яким числом інших вузлів; в результаті цього забезпечується формування мережі фактів. Можна вважати, що кожна з таких пар відносин представляє простий факт. Вузли позначаються ім'ям відповідного відношення. З позицій логіки базову структуру семантичної мережі можна розглядати як еквівалент предиката з двома аргументами (бінарний предикат); ці два аргументи представляються двома вузлами, а власне предикат – направленою дугою, що зв'язує ці вузли. При розумному виборі позначень відносин можна виразити дуже складні сукупності фактів. При розробці уявлення у вигляді семантичної мережі особливий практичний інтерес має зв'язок вигляду «є», що відображає приналежність до деякого класу об'єктів. До інших видів зв'язків відноситься зв'язок «має», вказуюча на те, що одне поняття представляє частину іншого, а також зв'язок «є», вказуюча на те, що одне поняття служить атрибутом іншого.

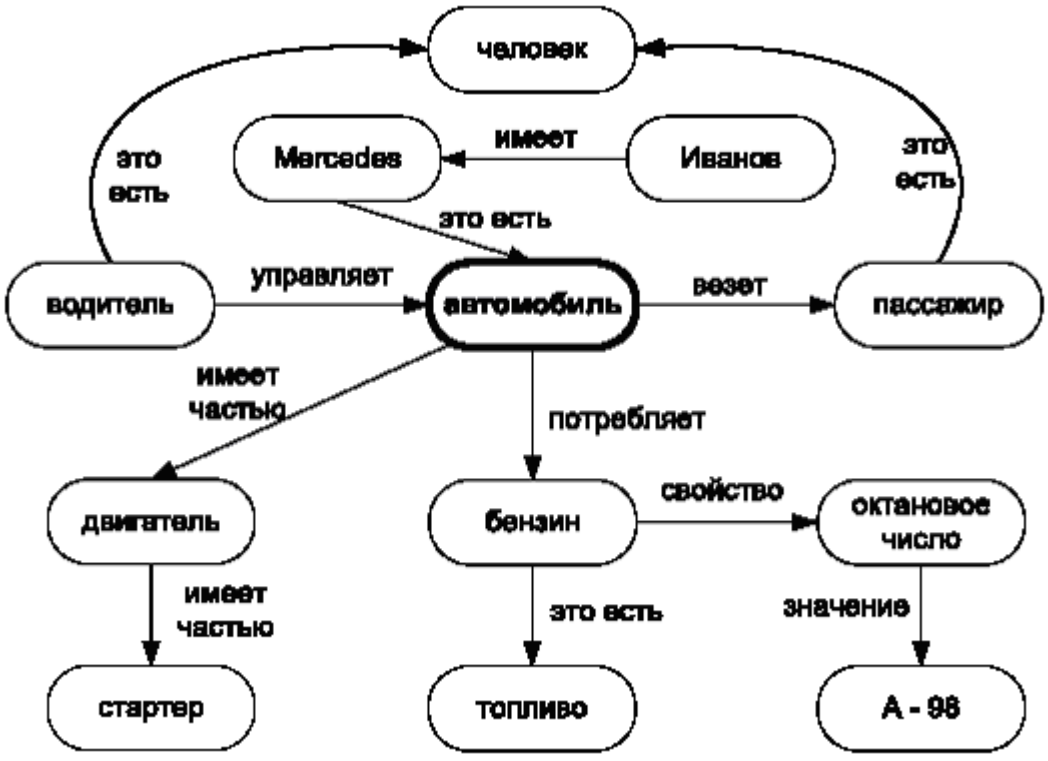


Рис.15. Приклад простої семантичної мережі

Прикладом простої семантичної мережі є опис об'єкту автомобіль і виряджаючи пов'язаних з ним понять (див. рис. 15. Приклад простої семантичної мережі.). На цій мережі присутній наступний ланцюжок понять: «автомобіль має частиною двигун», «двигун має частиною стартер». Через транзитивність відношення "мати частиною" можна вивести наступне твердження «автомобіль має частиною стартер». Аналогічно можна зробити цілком очевидний висновок, що «Іванов володіє автомобілем» або, що «Mercedes має частиною двигун і споживає паливо».

Користуючись подібними відносинами, можна представити складні сукупності фактів. За допомогою мережевих елементів можуть бути побудовані досить складні комбінації фактів. На мові Lisp (List Processing – обробка списків) базові елементи семантичної мережі можуть бути запрограмовані у вигляді спискової комбінації атом/властивість. Так, елемент **Іван працює у виробничий відділ** представляє факт – Іван працює у виробничому відділі: атом – «Іван», властивість – «працює», значення даної властивості – «виробничий відділ». Популярність семантичних мереж зобов'язана зв'язку «є», в якій закладені великі можливості для побудови ієрархії понять.

Основна перевага цієї моделі - у відповідності сучасним уявленням про організацію довготривалої пам'яті людини. Недолік моделі - складність пошуку висновку на семантичній мережі.

Підсумуємо основні властивості семантичних мереж.

1. Семантичні мережі описують відносини між об'єктами, які задаються вузлами мережі.

1. Вузли позначаються колами і мають імена.
2. Відносини між вузлами указуються зв'язуючими їх лініями.
3. Семантичну мережу можна використовувати для створення структур і об'єктів.
4. Семантичну мережу можна використовувати для створення правил бази знань.

Володіючи чималими можливостями, семантичні мережі при їх використовуванні не дуже зручні через дуже довільну структуру і різні типи вершин і зв'язків. Ця різноманітність вимагає різноманітності процедур обробки інформації, що міститься в семантичній мережі, що приводить до ускладнення програмного забезпечення ЕОМ. Так з'явилися особливі типи семантичних мереж з своїми специфічними назвами: синтагматичні **ланцюги, сценарії,** фрейми і т.п.

Приклад розв'язання задачі. Побудувати мережеву модель подання знань у предметній області «Ресторан» (відвідування ресторану).

Опис процесу рішення. Для побудови мережевої моделі подання знань необхідно виконати такі кроки:

1) Визначити абстрактні об'єкти і поняття предметної області, необхідні для вирішення поставленого завдання. Оформити їх у вигляді вершин.

2) Поставити властивості для виділених вершин, оформивши їх у вигляді вершин, пов'язаних з вихідними вершинами атрибутивними відносинами.

3) Поставити зв'язку між цими вершинами, використовуючи функціональні, просторові, кількісні, логічні, часові, атрибутивні відносини, а також відносини типу «бути спадкоємцем» і «бути частиною».

4) Додати конкретні об'єкти і поняття, що описують завдання, що розв'язується. Оформити їх у вигляді вершин, пов'язаних з уже існуючими відносинами типу «бути екземпляром», «є».

5) Перевірити правильність встановлених відносин (вершини і самовідношення при правильній побудові утворюють речення, наприклад «Двигун є частиною автомобіля»).

Розв'язок

1) Ключові поняття даної предметної області – ресторан, той, хто відвідує ресторан (клієнт) і ті, хто його обслуговують (кухарі, метрдотелі, офіціанти, для простоти обмежимося тільки офіціантами). У обслуговуючого персоналу і клієнтів є загальні характеристики, тому доцільно виділити загальне абстрактне поняття – людина. Продукцією ресторану є страви, які замовляють клієнти.

Виходячи з цього, вершини графа будуть такими: «Ресторан», «Людина», «Офіціант», «Клієнт», «Замовлення» і «Страва».

2) У цих об'єктів є певні властивості й атрибути. Наприклад, ресторани розташовуються за певними адресами, кожне страва з меню має свою ціну. Тому додамо вершини «Адреса» і «Ціна».

3) Визначимо для наявних вершин відносини і їх типи, використовуючи таблицю 2.

4) Додамо знання про конкретні факти розв'язуваної задачі. Нехай є два ресторани: «Смакота» і «Смачна їжа», у першому працює офіціантка Марина, а в другому офіціант Сергій. Петро вирішив піти до ресторану «Смачна їжа» і зробив замовлення офіціантові на 2 страви: картопля фрі за 30 р., Біфштекс за 130 р. Також відомі адреси цих ресторанів та їх специфіка.

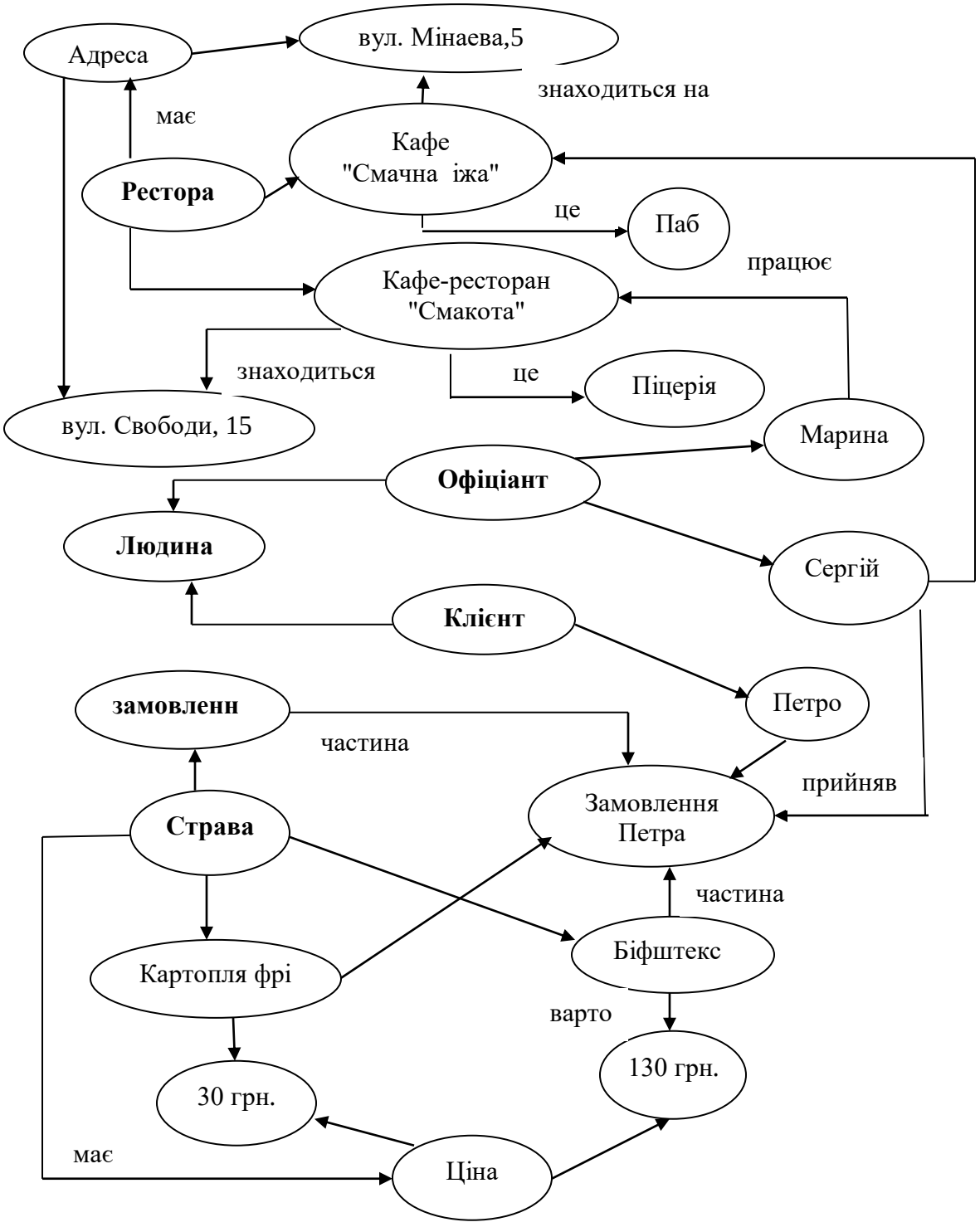


Рис. 16. Семантична мережа предметної області «Ресторан»

Виходячи з цього, додамо відповідні вершини у граф і з'єднаємо їх функціональними відносинами і відносинами типу «наприклад або бути екземпляром». Отриманий у результаті граф зображений на рис.16.

5) Здійснимо перевірку встановлених зв'язків. Наприклад, візьмемо вершину «Страва» і пройдемо за встановленими зв'язками. Отримуємо таку інформацію: страва є частиною замовлення, прикладами страв можуть служити картопля фрі і біфштекс.

Для отримання відповіді на будь-яке питання по цій задачі, необхідно знайти відповідну ділянку мережі і, використовуючи зв'язки, отримати результат.

Наприклад, питання «Яка ціна замовлення Петра (скільки Петро заплатив за замовлення)?» З запиту зрозуміло, що необхідно знайти такі вершини: «Ціна», «Петро» і «Замовлення» або «Замовлення Петра». Частина семантичної мережі, що знаходиться між цими вершинами, містить відповідь, а саме, частиною замовлення Петра є картопля фрі і біфштекс, які коштують 30 і 130 грн. відповідно. Більше інформації про замовлення Петра в моделі немає, тому робимо висновок - Петро заплатив 160 грн.

Лекція 7-8. Фреймова модель.

Фрейм – це мінімально можливий опис сутності якої-небудь події, ситуації, процесу або об'єкта. Фреймова модель представлення знань була запропонована М. Мінським у 1979 році і є розвитком семантичних мереж. Фрейм (англ. Frame) – абстрактний образ для представлення деякого стереотипу сприйняття. Кожен фрейм має власну назву і список слотів та їх значень. Іншими словами, фрейм – це форма опису знань, що окреслює рамки розглянутого (у поточній ситуації при вирішенні даного завдання) фрагмента предметної області.

«Фрейм – це структура даних, представляюча стереотипну ситуацію, на зразок знаходження усередині деякого роду житлової кімнати, або збору на вечірку з приводу дня народження дитини. До кожного фрейма приєднується декілька видів інформації. Частина цієї інформації – про те, як використовувати фрейм. Частина про те, чого можна чекати далі. Частина про те, що слід робити, якщо ці очікування не підтвердяться».

Фрейм - це мінімальний можливий опис суті якого-небудь явища, події, ситуації, процесу або об'єкту. Мінімальність означає, що при подальшому спрощенні опису втрачається його повнота, вона перестає визначати ту одиницю знань, для якої призначено. Наприклад, слово "кімната" викликає у слухаючих образ кімнати: "житлове приміщення з чотирма стінами, підлогою, стелею, вікнами і дверима, площею 6-20ì ² x0". З цього опису нічого не можна прибрати (наприклад, прибравши вікна ми одержимо вже комірку, а не кімнату), але в ньому є "дірки", - це незаповнені значення деяких атрибутів - кількість вікон, колір стін, висота стелі. покриття підлоги і ін. У теорії фреймів такий образ називається фреймом.

Фрейм має певну структуру, що складається з безлічі елементів – слотів. Кожен слот в свою чергу, представляється певною структурою даних, процедурою, або може бути пов'язаний з іншим фреймом.

Значеннями можуть бути дані будь-якого типу, а також назва іншого фрейму. Таким чином, фрейми утворюють мережу. Крім того, існує зв'язок між фреймами типу АКО (a kind of), яка вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються список і значення слотів. При цьому можливо множинне спадкування – перенесення властивостей від декількох прототипів.

Модель фрейма є досить універсальною, оскільки дозволяє відобразити все різноманіття знань про світ через:

– фрейми-структури, для позначення об'єктів і понять (позика, застава, вексель);

– фрейми-ролі (менеджер, касир, клієнт);

– фрейми-сценарії (банкрутство, збори акціонерів, святкування іменин);

– фрейми-ситуації (тривога, аварія, робочий режим пристрою) тощо.

Розрізняють фрейми-зразки, або прототипи, і фрейми-екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що надходять.

Фрейм має ім'я (назву) та складається зі слотів.

Традиційно структура фрейму може бути представлена як список властивостей:

(**Ім'я фреймау**:

(Ім'я 1-го слота: значення 1-го слота),

(Ім'я 2-го слота: значення 2-го слота),

.....

(Ім'я N-го слота: значення N-го слота)).

Табличне представлення слота виглядає таким чином (таблиця 4):

Таблиця 4. Структура фрейма.

ІМ'Я ФРЕЙМА			
Ім'я слота	Значення слота	Спосіб отримання значення	Приєднана процедура

У таблиці додаткові стовпці (3-й і 4-й) призначені для опису способу отримання слотом його значення і можливого приєднання до того чи іншого слота спеціальних процедур, що допускається в теорії фреймів. Як значення слота може виступати ім'я іншої фрейма – так утворюються мережі фреймів.

Таблиця 5– Способи отримання значень слотів

Спосіб	**Опис**
За замовчуванням на прототипі (одного з батьків)	Слоту присвоюється значення, визначене за замовчуванням у фреймі-прототипі, деякі стандартні значення.
Через спадкування	Відрізняється від першого способу тим, що значення задано в спеціальному слоті батьківського фрейма, з'єднаного з поточним зв'язком АКО.
За формулою	Слоту призначається формула, результат обчислення якої є значенням слота.
Через приєднану процедуру	Слоту призначається процедура, що дозволяє отримати значення слота алгоритмічно.
Із зовнішніх джерел даних	При використанні моделі в інтелектуальних системах дані, які є значеннями слотів, можуть надходити з баз даних, від системи датчиків, від користувача.

При табличному поданні фрейма крім уже описаних складових фрейма вказуються і додаткові параметри. Спосіб отримання значення визначає, як саме встановлюється значення конкретного слота. Існує кілька способів (таблиця 5), вибір способу залежить від властивостей самих даних.

У теорії фреймів допускається, щоб до слотів приєднувалися різні спеціальні процедури. Для цього використовуються так звані демони. Демоном (таблиця 6) називається процедура, що автоматично запускається при виконанні деякої умови (події) при зверненні до відповідного слоту. Демонів може бути

кілька. Найбільш схожий механізм приєднаних процедур до триггерів у реляційних базах даних.

Таблиця 6. Найбільш поширені демони.

Демон	Подія	Опис
IF-REMOVED	якщо видалено	Виконується, коли інформація видаляється зі слота
IF-ADDED	якщо додано	Виконується, коли нова інформація записується в слот.
IF-NEEDED	за вимогою	Виконується, коли запитується інформація з пустого слота
IF-DEFAULT	за замовчуванням	Виконується, коли встановлюється значення за замовчуванням

Існує кілька видів фреймів, які дають можливість окреслити предметну область і вирішуване завдання. У таблиці 7 представлені найбільш поширені типи фреймів, вказані типи знань, які вони відображають, а також приклади фреймів даного типу з різних предметних областей.

Із слотом можна пов'язати будь-яку кількість процедур, але найчастіше використовуються наступні:

Процедура на подію «якщо додано» (IF-REMOVED). Виконується, коли нова інформація записується в слот.

Процедура на подію «якщо видалено» (IF-ADDED). Виконується, коли інформація віддаляється із слота.

Процедура на подію «на вимогу» (IF-NEEDED). Виконується, коли запрошується інформація з порожнього слота.

Як значення слота може виступати ім'я іншого фрейма; так утворюють мережі фреймів.

Розрізняють фрейми-зразки, або прототипи, що зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних ситуацій на основі поступаючих даних. Прототип – це вже не абстрактний образ, а найтиповіший представник свого класу, з узагальненими, але цілком конкретними, значеннями своїх властивостей. Наприклад, прототип поняття чотирикутник можна визначити як фігуру, що має чотири кути.

Таблиця 7 – Типи фреймів.

Тип фрейма	Тип знання	Опис	Приклад
За пізнавальним призначенням			
Фрейми-прототипи (шаблони, зразки)	інтенсіональні	Відображають знання про абстрактні стереотипні поняття, які є класами якихось конкретних об'єктів	людина, автомобіль
Фрейми-екземпляри (приклади)	екстенсіональні	Відображають знання про конкретні факти предметної області	Іванов І.І., ВАЗ-2110
За функціональним призначенням			
Фрейми-структури (об'єкти)	декларативні	Відображають абстрактні і конкретні предмети і поняття предметної області (містять набір характеристик, описують об'єкт або поняття)	позика, застава, вексель, людина, лекція
Фрейми-операції	процедурні	Відображають різні процеси перетворення або використання об'єктів предметної області (містять набір характеристик процесу)	процеси отримання позику, синтезу пристроїв
Фрейми-ситуації	прагматічні	Відображають типові ситуації, в яких можуть перебувати фрейми-об'єкти і фрейми-ролі (містять набір характеристик, що ідентифікують ситуацію)	аварія, тривога, робочий режим пристрою
Фрейми-сценарії	иехнологичні	Відображають розвиток ситуації, типову структуру для деякої дії, поняття, події, відображають динаміку (містять набір характеристик, дозволяють забезпечити розвиток системи за таким сценарієм)	банкрутство, святкування іменин, здача іспиту
Фрейми-ролі	функціональні	Відображають типову роль, виконувану фреймом-об'єктом у певній ситуації (містять набір характеристик ролі)	менеджер, касир, клієнт, студент, викладач

Модель фрейма є достатньо універсальною, оскільки дозволяє відобразити все різноманіття знань про світ через:
фрейми-структури, для позначення об'єктів і понять (заїм, застава, вексель);
фрейми-ролі (менеджер, касир, клієнт);
фрейми-сценарії (банкрутство, збори акціонерів, святкування іменин);
фрейми-ситуації (тривога, аварія, робочий режим пристрою) і ін.

Важнейшим свойством теории фреймов является заимствованное из теории семантических сетей наследование свойств. И во фреймах, и в семантических сетях наследование происходит по АКО-связям (A-Kind-Of =

это). Слот АКО указывает на фрейм более высокого уровня иерархии, откуда неявно наследуются, т.е. переносятся, список и значения слотов. Возможно наследование свойств от нескольких прототипов. Такой вид наследования получи название «множественное наследование».

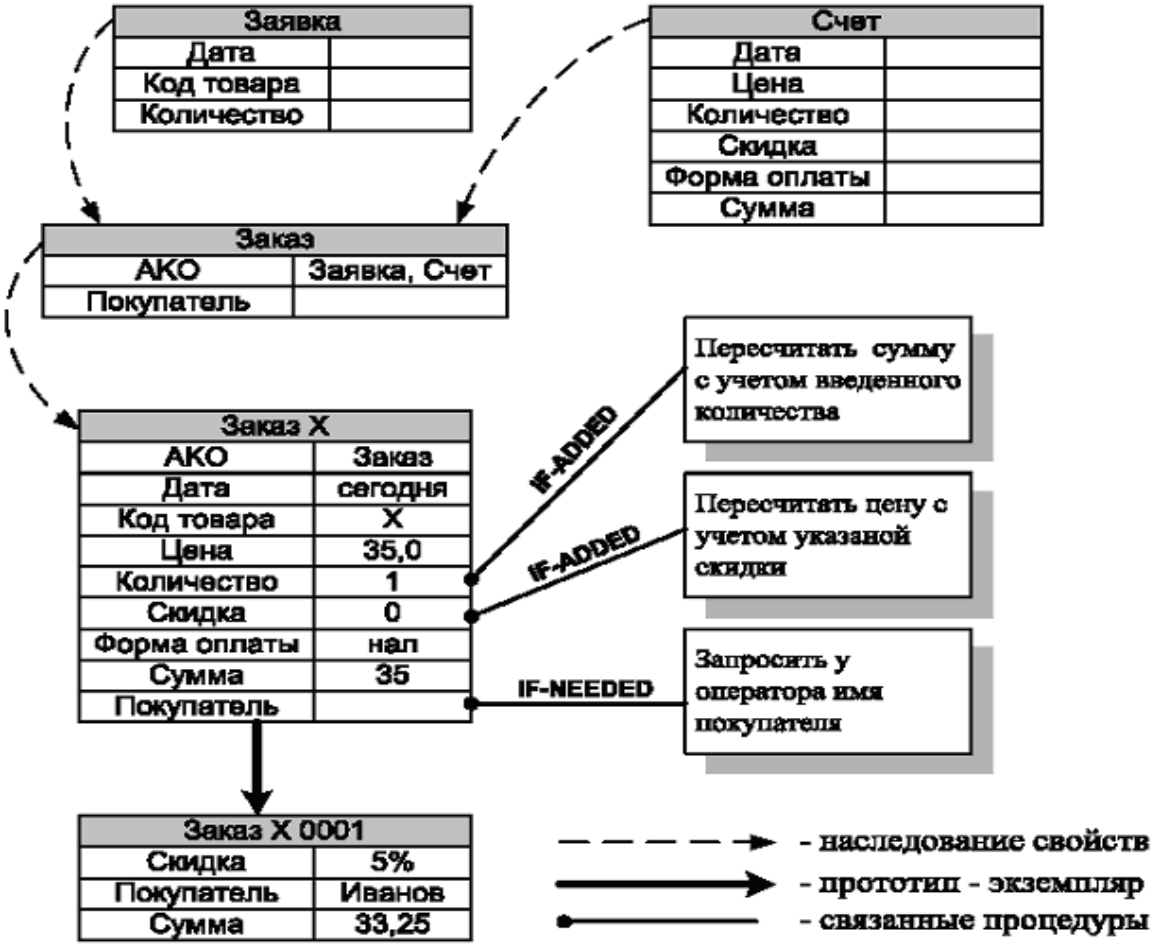


Рис.17. Приклад опису знань за допомогою фреймів

Як приклад можна розглянути формування поняття замовлення товару (див. рис.17. Приклад опису знань за допомогою фреймів.).

Основною перевагою фреймів як моделі представлення знань є здатність відображати концептуальну основу організації пам'яті людини, а також гнучкість і наочність.

Розрізняють **статичні** і **динамічні** системи **фреймів**. У системах першого типу фрейми не можуть бути змінені в процесі рішення задачі, в системах другого типу це допустимо.

Про системи програмування, заснованих на фреймах, говорять, що вони є об'єктно-орієнтованими. Кожен фрейм відповідає деякому об'єкту наочної області, а слоти містять дані, що описують цей слот, тобто в слотах знаходяться значення ознак об'єкту. Фрейм може бути представлений у вигляді списку властивостей, а якщо використовувати засоби бази даних, то у вигляді запису. Найяскравіше достоїнства фреймових систем представлення знань виявляються в тому випадку, якщо родовидові зв'язки змінюються нечасто і наочна область налічує небагато виключень. Значення будь-якого слота при необхідності може бути обчислено за допомогою відповідних процедур або знайдено евристичними методами.

Найпоширенішим типом фрейма може служити структура представлення знань наступного типу: : (Имя фрейма (<Имя слота>(Значение слота>)(<Имя слота><Значение слота>),…(<Имя слота><Значение слота>)).

Например, рассмотрим **фрейм**: «квартира» (квартира <столовая> <20кв.м>) (<стол><обеденный раскладной>)(<стулья><6, мягкие>…).

Перевагами системи, що використовує фрейми, полягає у тому, що ті елементи, які традиційно присутні в описі об'єкту або події, групуються і завдяки цьому можуть витягуватися і оброблятися як єдине ціле. Скелетний фрейм для поняття «керівник» має вигляд:

имя: **РУКОВОДИТЕЛЬ**

специальность: **СЛУЖАЩИЙ**

имя:_____

возраст:_____

адрес:_____

отдел:_____

заработная плата:_____

дата начала:_____

до:_____

Рис.18. Скелетний фрейм для поняття "КЕРІВНИК"

Фрейм має ім'я для ідентифікації, його опис складається з ряду описів, приведених зліва (слоти), за допомогою яких ідентифікуються основні структурні елементи понять. За слотами слідують шпації (проміжок), в які

поміщають деякі об'єкти, що представляють поточні значення слотів (у даному фреймі всі слоти порожні, за винятком першого). Нижче показаний той же фрейм, але із заповненими слотами.

имя: **РУКОВОДИТЕЛЬ**

специальность: **СЛУЖАЩИЙ**

имя: агрегат (фамилия, имя, отчество)

возраст: агрегат (годы)

адрес: **АДРЕС**

отдел: диапазон (производство, администрация)

заработная плата: **ЗАРПЛАТА**

дата начала: агрегат (месяц, год)

до: агрегат (месяц, год) (по умолчанию: теперь)

Частина слотів заповнена якимись об'єктами, а не простими іменами. У даному прикладі фігурують три різні типи заповнювачів. Заповнювач слота може бути або константою, або ім'ям іншого фрейма. Прості з них: **АДРЕСА**, **ЗАРПЛАТА** – це імена інших фреймів даної системи, на які робиться посилання. Позначення “агрегат” указує на те, що повинні бути задані певні об'єкти, а позначення “інтервал” – на те, що повинен бути вибраний один з безлічі об'єктів. Зрештою до заповнювача відповідних слотів приєднуються вказані в дужках значення, що маються (що приймаються за умовчанням) на увазі. Позначення “агрегат”, “ інтервал”, що “мається” на увазі називають **фасетами** слота.

Приклад фрейма для заповнювача ЗАРПЛАТА, що відноситься до фрейма КЕРІВНИК, має вигляд (містить чотири слоти, один з яких вимагає подвійного заповнювача):

имя: **ЗАРПЛАТА**

почасовая заработная плата: агрегат (грн. в час)

код налога: агрегат (код налога) (по умолчанию: непредвиденный)

налог на дату: агрегат (месяц, год)

вычислить ((**ОПЛАТА НАЛОГА**) (налог))

В даному прикладі проілюстровано поняття “процедурне приєднання”, яке дає можливість вставляти у фрейми звичні програми. Процедура, що

вставляється, викликається заповнювачем фасета “обчислити”. У слоті “податок на дату” фрейма ЗАРПЛАТА знаходимо заповнювач ”обчислити (ОПЛАТА ПОДАТКУ)(податок)”. Це указує на те, що заповнювач фрейма ЗАРПЛАТА повинен витягнути фрейм ”ОПЛАТА ПОДАТКУ”, що містить якусь процедуру. Ця процедура потім буде реалізована з використанням інформації, що міститься в системі фреймів як дані для обчислення значення ”податок на дату”, яке після цього може бути внесене в слот в полі ”податок”.

Таким чином, фрейм має «матрьошечну» структуру. Як значення слота може виступати система імен слотів глибшого рівня, а як значення в цих слотах можуть з'являтися нові послідовності імен і значень слотів. Ця властивість забезпечує фреймовим мовам задоволення вимозі структурованості знань. Наявність імен фреймів і імен слотів забезпечує можливість **внутрішньої інтерпретується** знань, що зберігаються у фреймах. Можливість мати як значення слоти посилання на інші слоти даного фрейма і на інші фрейми забезпечує властивість зв'язності в базі знань. Як значення слотів можуть стояти накази виклику тих або інших процедур для виконання, дозволяє мати засоби **активізації** програм за рахунок наявних знань.

Сукупність фреймів, моделююча яку-небудь наочну область, є ієрархічною структурою, в якій фрейми з'єднуються за допомогою родовидових зв'язків. На верхньому рівні ієрархії знаходиться фрейм, що містить найзагальнішу інформацію, істинну для всієї решти фреймів. Фрейми володіють здатність успадковувати значення характеристик своїх батьків, що знаходяться на вищому рівні ієрархії. Все це забезпечує широке розповсюдження мов такого типу в інтелектуальних системах.

Перехресні посилання фреймів один на одного створюють мережу з вельми складною геометрією. При пошуку інформації і її запису необхідно мати спеціальні процедури руху по ній.

Як недолік фреймових систем слід зазначити їх відносно високу складність, що виявляється в зниженні швидкості роботи висновку і в збільшенні трудомісткості внесення зміни в родовидову ієрархію.

Приклад розв'язання задачі. Побудувати фреймову модель подання знань у предметній області «Ресторан» (відвідування ресторану).

Опис процесу розв'язку. Для побудови фреймової моделі подання знань необхідно виконати такі кроки:

1). Визначити абстрактні об'єкти і поняття предметної області, необхідні для вирішення поставленого завдання. Оформити їх у вигляді фреймів-прототипів (фреймів-об'єктів, фреймів-ролей);
2). Поставити конкретні об'єкти предметної області. Оформити їх у вигляді фреймів-екземплярів (фреймів-об'єктів, фреймів-ролей);
3). Визначити набір можливих ситуацій. Оформити їх у вигляді фреймів-ситуацій (прототипи). Якщо існують прецеденти ситуацій у предметній області, додати фрейми-екземпляри (фрейми-ситуації);
4) Описати динаміку розвитку ситуацій (перехід від одних до інших) через набір сцен. Оформити їх у вигляді фреймів-сценаріїв;
5) Додати фрейми-об'єкти сценаріїв і сцен, які відображають дані конкретного завдання.

Розв'язання

1). Ключові поняття даної предметної області – ресторан, той, хто відвідує ресторан (клієнт) і ті, хто його обслуговують (кухарі, метрдотелі, офіціанти, для простоти обмежимося тільки офіціантами). У обслуговуючого персоналу і клієнтів є загальні характеристики, тому доцільно виділити загальне абстрактне поняття – людина. Тоді фрейми «Ресторан» і «Людина» є прототипами-зразками, а фрейми «Офіціант» і «Клієнт» – прототипами-ролями. Також потрібно визначити основні слоти фреймів – характеристики, що мають значення для розв'язуваної задачі.

ЛЮДИНА			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
пол	Чоловічий або жіночий	із зовнішніх джерел	
вік	Від 0 до 70 років	із зовнішніх джерел	

РЕСТОРАН			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Назва		із зовнішніх джерел	
Адреса		із зовнішніх джерел	
Години роботи		із зовнішніх джерел	
Спеціалізація		із зовнішніх джерел	
Клас		із зовнішніх джерел	

Фрейми-спадкоємці містять всі слоти своїх батьків, вони явно прописуються тільки в разі зміни будь-якого параметра.

ОФІЦІАНТ (АКО **ЛЮДИНА)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
вік	від 18 до 55 років	із зовнішніх джерел	
стаж роботи		із зовнішніх джерел	
зарплата		із зовнішніх джерел	
графік роботи		із зовнішніх джерел	
місце роботи	Фрейм-об'єкт	із зовнішніх джерел	

КЛІЄНТ (АКО **ЛЮДИНА)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Вид оплати	Готівкові або картка	За замовчуванням (готівкові)	
Статус	Звичайний або Vip	За замовчуванням (звичайний)	
Форма замовлення	Замовлення є чи ні	За замовчуванням (замовлення немає)	
Чайові		із зовнішніх джерел	

2) Фрейми-взірці описують конкретну ситуацію: які ресторани є в місті як саме організовується відвідування, хто є відвідувачем, хто працює в обраному ресторані і т.д. Тому визначимо наступні фрейми-зразки, які є спадкоємцями фреймів-прототипів:

КАФЕ-РЕСТОРАН " СМАКОТА" (АКО **РЕСТОРАН)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Назва	**Смакота**	із зовнішніх джерел	
Адреса	м. Одеса, вулиця Мінаева, 15	із зовнішніх джерел	
Години роботи	9:00-00:00	із зовнішніх джерел	
Спеціалізація	Паб	із зовнішніх джерел	
Клас	Фрейм-об'єкт	із зовнішніх джерел	

КАФЕ-РЕСТОРАН " СМАЧНА ІЖА " (АКО **РЕСТОРАН)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Назва	**Смачна іжа**	із зовнішніх джерел	
Адреса	м. Одеса, вулица Шілера, 5	із зовнішніх джерел	
Години роботи	9:00-00:00	із зовнішніх джерел	
Спеціалізація	Паб	із зовнішніх джерел	
Клас	середній	із зовнішніх джерел	

СЕРГІЙ (АКО **ОФІЦІАНТ)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
вік	27	із зовнішніх джерел	
пол	чоловічий	із зовнішніх джерел	
стаж роботи	5	із зовнішніх джерел	
зарплата	7000	із зовнішніх джерел	
графік роботи	Через день з 18:00 до 00:00	із зовнішніх джерел	
місце роботи	**СМАЧНА ІЖА**	із зовнішніх джерел	

МАРИНА (АКО **ОФІЦІАНТ)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
вік	24	із зовнішніх джерел	
стать	жіночий	із зовнішніх джерел	
стаж роботи	2	із зовнішніх джерел	
зарплата	8200	із зовнішніх джерел	
графік роботи	Кожен день з 9:00 до 14:00	із зовнішніх джерел	
місце роботи	**КАФЕ-РЕСТОРАН " СМАКОТА "**	із зовнішніх джерел	

ПЕТР (АКО **КЛІЄНТ)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
пол	чоловічий	із зовнішніх джерел	
вік	19	із зовнішніх джерел	
Вид оплати	готівкові	За замовчуванням (готівкові)	
Статус	звичайний	За замовчуванням (звичайний)	
Форма замовлення	замовлення немає	За замовчуванням (замовлення немає)	
Чайові	7% від суми замовлення	із зовнішніх джерел	

3) Фрейми-ситуації описують можливі ситуації. У ресторані клієнт потрапляє в декілька типових ситуацій: замовлення і оплата. Можливі й інші не типові ситуації: клієнт подавився, у клієнта немає готівки для оплати рахунку і т.ы. Розглянемо типові ситуації (їх може бути більше):

ЗАМОВЛЕННЯ			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Перелік страв		із зовнішніх джерел	IF-ADDED (змінює слот «Перелік цін»)
Перелік цін		Приєднана процедура	IF-ADDED (змінює слот «Сума замовлення»)
Сума замовлення		Приєднана процедура	
Прийняв замовлення	Фрейм-зразок	із зовнішніх джерел	
Зробив замовлення	Фрейм-зразок	із зовнішніх джерел	

ОПЛАТА			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Вид платежу		із зовнішніх джерел	IF-ADDED (змінює слот «Чайові»)
Чайові		Приєднана процедура	
Сплатив	Фрейм-зразок	Приєднана процедура	
Замовлення	Фрейм-зразок	із зовнішніх джерел	IF-ADDED (змінює слот «Сплатив»)

4). Ситуації виникають після настання якихось подій, виконання умов та можуть слідувати одна за одною. Динаміку предметної області можна відобразити в фреймах-сценаріях. Їх може бути безліч, опишемо найбільш загальний і типовий сценарій відвідування ресторану:

ВІДВІДУВАННЯ РЕСТОРАНУ			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Відвідувач	Фрейм-об'єкт	із зовнішніх джерел	
Ресторан	Фрейм-об'єкт	із зовнішніх джерел	IF-ADDED (змінює слот «Офіціант»)
Офіціант	Фрейм-об'єкт	Приєднана процедура (визначає за обраним ресторану)	
Сцена 1	Вхід, вибір	із зовнішніх джерел	
Сцена 2	Замовлення	із зовнішніх джерел	
Сцена 3	Іжа	із зовнішніх джерел	
Сцена 4	Оплата	із зовнішніх джерел	
Сцена 5	Вихід	із зовнішніх джерел	

5) Нехай в рамках нашого завдання Петро відвідав ресторан «Смачна їжа». Тоді фрейми буде заповнено так:

ВІДВІДУВАННЯ " СМАЧНА ЇЖА " (АКО **ВІДВІДУВАННЯ РЕСТОРАНА)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Відвідувач	Петр	із зовнішніх джерел	
Ресторан	**СМАЧНА ЇЖА**	із зовнішніх джерел	IF-ADDED, IF-REMOVED (змінює слот «Офіціант»)
Офіціант	**Сергій**	Приєднана процедура (визначає за обраним ресторану)	
Сцена 1	Вхід, вибір	із зовнішніх джерел	
Сцена 2	**Замовлення Петра**	із зовнішніх джерел	
Сцена 3	Іжа	із зовнішніх джерел	
Сцена 4	**Оплата Петра**	із зовнішніх джерел	
Сцена 5	Вихід	із зовнішніх джерел	

ЗАМОВЛЕННЯ ПЕТРА (АКО **ЗАМОВЛЕННЯ)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Перелік страв	Відбивна, темне пиво	із зовнішніх джерел	IF-ADDED (змінює слот «Перелік цін»)
Перелік цін	250, 75	Приєднана процедура	IF-ADDED (змінює слот «Сума замовлення»)
Сума замовлення	325	Приєднана процедура	
Прийняв замовлення	**СЕРГІЙ**	із зовнішніх джерел	
Зробив замовлення	**ПЕТР**	із зовнішніх джерел	

ОПЛАТА ПЕТРА (АКО **ОПЛАТА)**			
І'мя слота	**Значення слота**	**Спосіб отримання значення**	**Демон**
Вид платежу	Наличные	із зовнішніх джерел	IF-ADDED (змінює слот «Чайові»)
Чайові	30	Приєднана процедура	
Сплатив	**ПЕТР**	із зовнішніх джерел	
Заказ	**Замовлення ПЕТРА**	із зовнішніх джерел	IF-ADDED (змиінює слот «Сплатив»)

Взаємозв'язок різних видів фреймів відображається графічно у вигляді графа (рис. 19).

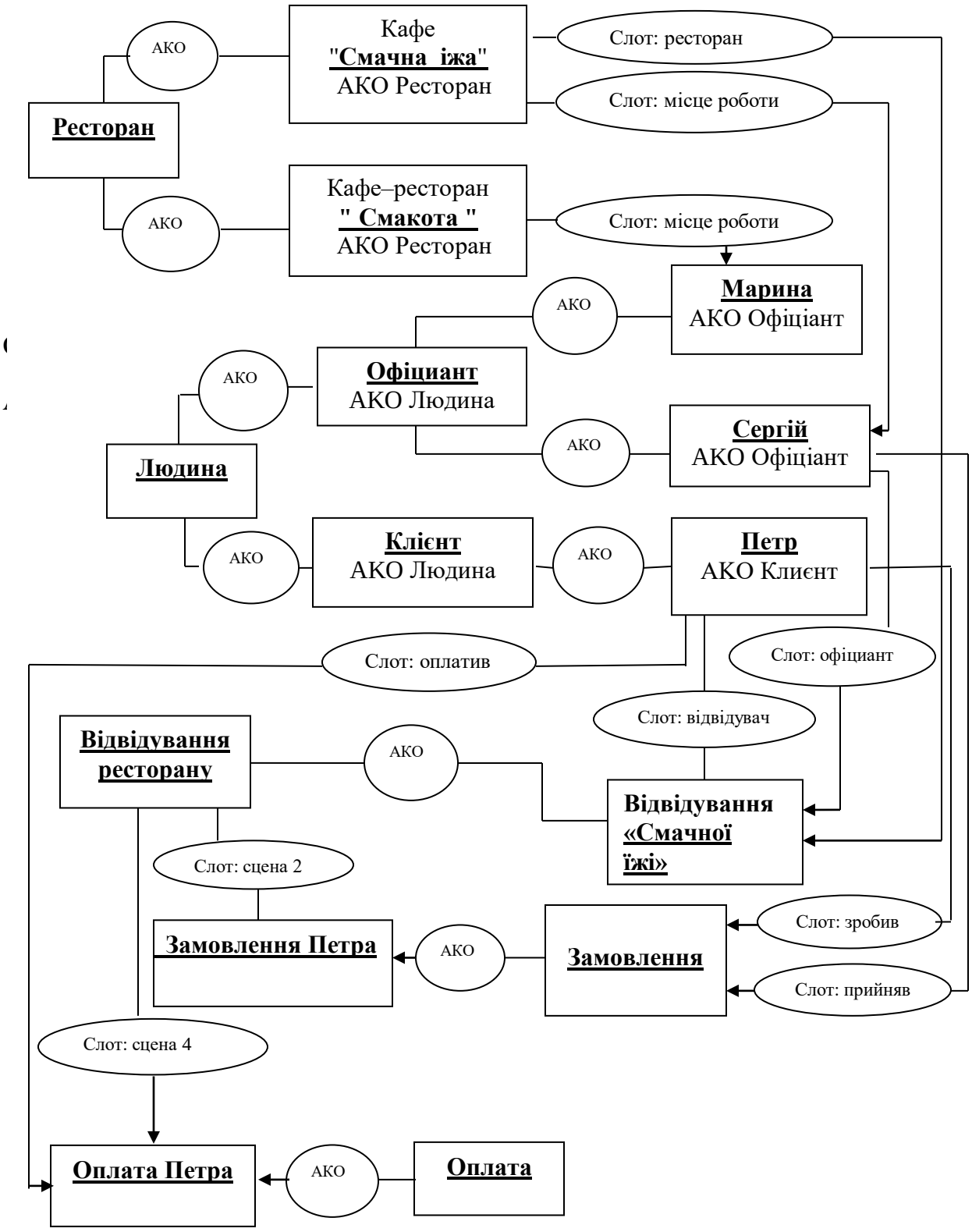


Рис. 19. Схема фреймів для предметної області «Ресторан».

Використання фреймової моделі аналогічно семантичної, тільки в процесі отримання відповіді крім вершин враховуються і слоти. Наприклад, отримати

відповідь на питання «Хто працює офіціантом в ресторані" Смачна їжа "?» Можна наступним чином: із запиту зрозуміло, що необхідно знайти фрейм «Ресторан" Смачна їжа "» і простежити зв'язок з фреймом «Сергій», яка є спадкоємцем фрейму « Офіціант ». Також можна знайти слот «Місце роботи» і перевіривши його значення у фреймах спадкоємців фрейму «Офіціант» визначити, що офіціантом в ресторані "Смачна їжа" працює Сергій.

Завдання для самостійного розв'язку

1. Побудувати продукційну (семантичну, фреймову) модель подання знань ув предметній області «Аеропорт» (диспетчерська).
2. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Залізниця» (продаж квитків).
3. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Торговий центр» (організація).
4. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Автозаправка» (обслуговування клієнтів).
5. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Автопарк» (пасажирські перевезення).
6. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Комп'ютерні мережі» (організація).
7. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Університет» (навчальний процес).
8. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Комп'ютерна безпека» (засоби і способи її забезпечення).
9. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Комп'ютерна безпека» (загрози).
10. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Інтернет-кафе» (організація й обслуговування).
11. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Розробка інформаційних систем» (ведення інформаційного проекту).
12. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Туристична агенція» (робота з клієнтами).
13. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Зоопарк» (організація).

14. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Кухня» (приготування їжі).
15. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Лікарня» (прийом хворих).
16. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Кінопрокат» (асортимент і робота з клієнтами).
17. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Прокат автомобілів» (асортимент і робота з клієнтами).
18. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Операційні системи» (функціонування).
19. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Інформаційні системи» (види і функціонування).
20. Побудувати продукційну (семантичну, фреймову) модель подання знань у предметній області «Підприємство» (структура і функціонування).

Лекція 9. Логічна модель. Експертні системи.

Традиційно в представленні знань виділяють логічні моделі, засновані на класичному численні предикатів першого порядку, коли наочна область або задача описується у вигляді набору аксіом. Основна перевага використовування логіки предикатів для представлення знань полягає у тому, що володіючий добре зрозумілими математичними властивостями могутній механізм висновку може бути безпосередньо запрограмований. За допомогою цих програм з відомих раніше знань можуть бути одержані нові знання.

Мови логічного типу спираються на числення, запозичені з логіки (найчастіше на числення предикатів). Кожен факт в базі знань описується у вигляді деякого набору предикатів. Якщо, наприклад, Р1 є предикат «добре відноситься», то P_1(С, А) може інтерпретуватися як твердження: « система добре відноситься до користувача А». Складні структури з фактів задаються формулами, що складаються з предикатів і операцій. Наприклад, формула $P_1(C, A) \& \overline{P}_1(C, \acute{A}) \& P_1(A, \acute{A})$ описує ситуацію на рис.а. У цій формулі риска над символом предиката означає заперечення відповідного твердження, а значок & (операція кон'юнкції) означає одночасність того, що всі три предикати, є істинними. Закономірності, характерні для проблемної області, в якій працює система, також задаються формулами числення предикатів. Для цього

використовується операція імплікації, що позначається стрілкою $\rightarrow$. Запис: $(P_1(C, X) \& P(X, Y)) \rightarrow P_1(C, Y)$ може характеризувати принцип типу: «друзі моїх друзів – мої друзі». В цій формулі X і Y – перемінні, замість яких можна підставляти різні обличчя, пошук необхідного факту або закономірності в базах знань, що використовують мови логічного типу, зводиться до реалізації процедури логічного виводу.

Гідністю мов логічного типу є існування могутніх процедур логічного висновку, забезпечуючих пошук висновку в більшості випадків. При використовуванні подібних мов досить просто розв'язується важлива для баз знань задача поповнення текстових описів, що поступають в базу знань після роботи системи спілкування.

Недоліком, властивим всім мовам логічного типу, є їх погана наочність для людини. Вельми нелегко описувати проблемну область у вигляді набору аксіом, що відображають фундаментальні факти і закономірності, характерні для неї. На практиці вони не набули великого поширення через малу наочність бази знань.

Експертні системи

Розглянемо декілька визначень. **Експертна система** - програмно-технічний засіб, що дозволяє користувачу в діалоговому режимі одержувати від комп'ютера допомогу консультанта в конкретній наочній області, де сконцентровані досвід і знання людей-експертів (фахівців в даній області).

Експертні системи – програми для комп'ютера, які можуть відтворювати процес рішення проблеми людиною-експертом.

Експертна система - програма, яка використовує знання фахівців (експертів) про деяку конкретну вузько спеціалізовану наочну область і в межах цієї області здатна ухвалювати рішення на рівні експерта-професіонала.

Експертні системи - прикладні програми ІІ, в яких база знань є формалізованими емпіричними знаннями висококваліфікованих фахівців (експертів) в якій-небудь вузькій наочній області.

У основі функціонування ЕС лежить використовування знань, а маніпулювання ними здійснюється на базі евристичних правил, сформульованих експертами. ЕС видають ради, проводять аналіз, виконують класифікацію, дають консультації і ставлять діагноз. Вони орієнтовані на рішення задач, що звичайно вимагають проведення експертизи людиною-

фахівцем. На відміну від машинних програм, використовуючий процедурний аналіз, ЕС вирішують задачі у вузькій наочній області (конкретної області експертизи) на основі дедуктивних міркувань. Головна гідність експертних систем - можливість накопичувати знання, зберігати їх тривалий час, обновляти і тим самим забезпечувати відносну незалежність конкретної організації від наявності в ній кваліфікованих фахівців.

Всі експертні системи є системами штучного інтелекту, але не всі системи штучного інтелекту є ЕС. Для ЕС характерна наявність мети функціонування, полягаючої в рішенні складних проблем, рішення яких під силу фахівцю високої кваліфікації – експерту. Виділимо характерні риси ЕС:

1. Алгоритми функціонування ЕС імітують підхід до рішення проблеми з боку людини.
2. Уміння пояснювати свої дії в зрозумілій для людини формі.
3. Наявність природно-мовного інтерфейсу.

Розрізняють ЕС **наочно-орієнтовані** і **ЕС-оболонки**, призначені для наповнення будь-яким наочним знанням. ЕС можуть будується на основі представлення знань у вигляді набору правил і на базі адаптивного підходу, заснованого на навчанні системи на прикладах. У ЕС першого типа логічний висновок описується великим числом правил (продукції) типу

ЯКЩО умова **ТА** дія

Умова визначає обставини, при яких правило повинне використовуватися, дія – що повинне відбуватися, коли умова спрацьовує. Дія може бути будь-якою, але, як правило, йдеться про висновок висновку як частину аргументування або доказу. В цьому випадку правила представляють наступне:

ЯКЩО посилка ТОЙ висновок

У ЕС, заснованих на правилах, наочні знання представляються набором правил, які перевіряються через набір фактів або знань про поточну ситуацію. Коли частина ЯКЩО правила задовольняє фактам, то дія, вказана в частині ТЕ, виконується. Інтерпретатор правил порівнює частину ЯКЩО правила з фактами і виконує те правило, чия частина ТЕ узгоджується з фактами.

Робота всіх експертних систем заснована строго на експертній інформації, одержаній в конкретній проблемній області. При створенні бази знань для ЕС програмна система повинна мати елементи, що становлять процес ухвалення рішення людиною - цілі, факти, правила, механізми висновку і спрощення. Основні компоненти таких систем показані на рис.20.

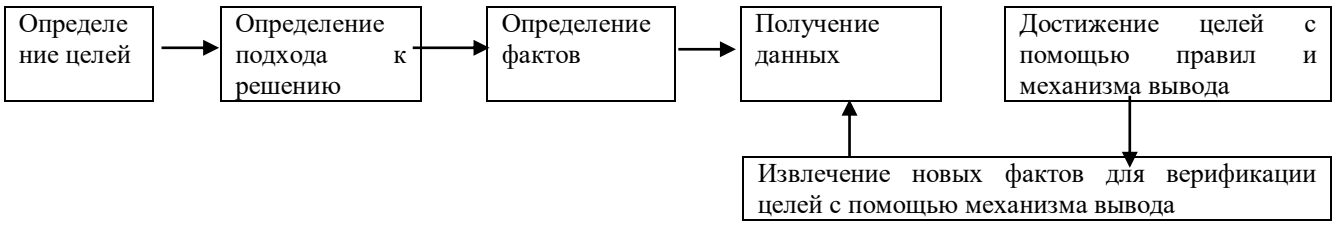


Рис.20.

При проектуванні системи ШІ перш за все потрібно визначити цілі, для досягнення яких вона призначена. Приступаючи до розробки програми, необхідної для вирішення деякої задачі, треба знати, до якого класу відноситься дана задача, і уміти описати її в потрібних термінах. Перш ніж ухвалити рішення, людина зважує різні чинники. В процесі ухвалення рішення постійно використовуються правила, у тому числі і спеціальні правила механізму спрощення. Неістотними вважаються всі факти і правила, що не відносяться до досягнення сформульованої мети.

Факти формулюються у вигляді питань, відповіді на які допомагають людині ухвалити остаточне рішення. Відносна важливість (вага) фактів оцінюється за допомогою вагових чинників (вибираються не випадково), що є знаннями, одержаними в результаті дослідження проблемної області. Ніж більше призначена факту вага, тим більше значення має цей факт при рішенні задачі.

Коли доцільне використовування експертних систем?

Експертні системи доцільно використовувати тоді, коли 1) *розробка ЕС можлива*, 2) *виправдана* і 3) *методи інженерії знань відповідають вирішуваній задачі*.

Розглянемо детальніше ці умови.

Розробка ЕС можлива, коли:

- існують експерти в даній області;
- експерти повинні сходитися в оцінці пропонованого рішення;
- експерти повинні уміти виразити на природній мові і пояснити використовувані методи;
- задача вимагає тільки міркувань, а не дій;
- задача не повинна бути дуже важкою, її рішення повинне займати у експерта до декількох годин або днів, а не тижнів або місяців;
- задача повинна відноситися до достатньо структурованої області;

- рішення не повинне використовувати значною мірою здоровий глузд (тобто широкий спектр загальних відомостей про світ і про спосіб його функціонування).

Розробка ЕС виправдана, якщо:

- рішення задачі принесе значний ефект:
- використовувати людину-експерта неможливо через обмежену кількість експертів або через необхідність виконання експертизи одночасно в багатьох місцях;
- при передачі інформації експерту відбувається значна втрата часу або інформації;
- необхідно вирішувати задачу в оточенні, ворожому людині.

Методи інженерії знань відповідають задачі, якщо задача володіє наступними характеристиками:

- може бути природним чином вирішена за допомогою маніпуляції з символами, а не з числами;
- має евристичну природу, тобто не годиться задача, яка може бути вирішена гарантований за допомогою деяких формальних процедур;
- повинна бути достатньо складною, щоб виправдати витрати, але не надмірно складною;
- повинна бути достатньо вузькою, але практично значущої.

Призначення експертних систем

Таким чином, експертні системи - це прикладні системи ШІ, в яких база знань є формалізованими емпіричними знаннями висококваліфікованих фахівців (експертів) в якій або вузької наочної області. Експертні системи призначені для заміни при рішенні задач експертів через їх недостатню кількість, недостатню оперативність в рішенні задачі або в небезпечних (шкідливих) для них умовах.

Звичайно експертні системи розглядаються з погляду їх застосування в двох аспектах: для вирішення яких задач вони можуть бути використані і в якій області діяльності. Ці два аспекти накладають свій відбиток на архітектуру експертної системи, що розробляється.

ЕС знайшли широке застосування в оцінці позик, ризиків страхування і капітальних вкладень, медичній діагностиці і лікуванні захворювань, синтезі нових хімічних з'єднань із заданими властивостями, навчанні і ін.

Можна виділити наступні основні класи задач, вирішуваних експертними системами:

- діагностика,
- прогнозування,
- ідентифікація,
- управління,
- проектування,
- моніторинг.

Області діяльності, що найбільш ширше зустрічаються, де використовуються експертні системи:

- медицина,
- обчислювальна техніка,
- військова справа,
- мікроелектроніка,
- радіоелектроніка,
- юриспруденція,
- економіка,
- екологія,
- геологія (пошук корисної копалини),
- математика.

Приклади широко відомих і ефективно використовуваних (або використаних свого часу) експертних систем:

DENDRAL *- ЕС для розпізнавання структури складних органічних молекул за наслідками їх спектрального аналізу (вважається першою в світі експертною системою),* ***MOLGEN*** *- ЕС для вироблення гіпотез про структуру ДНК на основі експериментів з ферментами;* ***MYCIN-*** *ЕС діагности кишкових захворювань;* ***PUFF*** *- ЕС діагности легеневих захворювань;* ***MACSYMA*** *- ЕС для символьних перетворень виразів алгебри;* ***YES/MVS*** *- ЕС для управління багатозадачною операційною системою MVS великих ЕОМ корпорації IBM;* ***DART*** *- ЕС для діагностики великих НМД корпорації IBM;* ***PROSPECTOR*** *- ЕС для консультацій при пошуку покладів корисної копалини;* РОММЕ ***- ЕС*** *для видачі рекомендацій по догляду за яблуневим садом; набір експертних систем для управління плануванням, запуском і польотом космічних апаратів типу "човник";* ***ЕСПЛАН*** *- ЕС для планування виробництва*

*на Бакинському нафтопереробному заводі**; МОДІС - ЕС** діагности різних форм гіпертонії.*

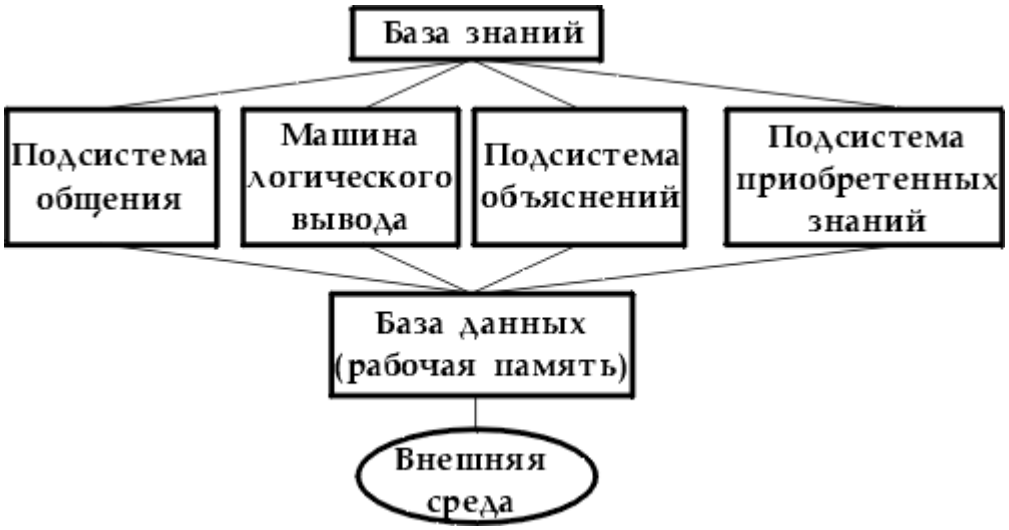


Рис.21. Структура експертної системи

На рис.21 зображена узагальнена структура експертної системи. **Підсистема придбання знань** призначена для додавання в базу знань нових правил і модифікації є. У простому випадку це - інтелектуальний редактор бази знань, в складніших експертних системах - засоби для витягання знань з баз даних, неструктурованого тексту, графічної інформації і т.д. У її задачу входить приведення правила до вигляду, що дозволяє підсистемі висновку застосовувати це правило в процесі роботи. У складніших системах передбачені ще і засоби для перевірки правил, що вводяться або модифікуються, на несуперечність з наявними правилами.

База знань - важлива компоненту експертної системи, вона призначена для зберігання довгострокових даних, що описують дану наочну область, і правил, що описують доцільні перетворення даних цієї області. Як наочна область вибирається вузька (спеціальна) прикладна область. Далі для створення ЕС у вибраній області збираються факти і правила, які поміщаються в базу знань разом з механізмами висновку і спрощення. На відміну від всіх інших компонент ЕС, база знань - "змінна " частина системи, яка може поповнюватися і модифікуватися інженерами знань і досвіду використовування ЕС, між консультаціями (а в деяких системах і в процесі консультації). Існує декілька способів представлення знань в ЕС, проте загальним для всіх них є те, що знання представлені в символьній формі (елементарними компонентами

представлення знань є тексти, списки і інші символьні структури). Тим самим, в ЕС реалізується принцип символьної природи міркувань, який полягає у тому, що процес міркування представляється як послідовність символьних перетворень. Існують динамічні і статичні бази знань. Динамічна база знань змінюється з часом. Її вміст залежить і від стану оточуючої. Нові факти, що додаються в базу знань, є результатом висновку, який полягає в застосуванні правил до наявних фактів. У системах з монотонним висновком факти, що зберігаються в базі знань, статичні, тобто не змінюються в процесі рішення задачі. У системах з немонотонним висновком допускається зміна або видалення фактів з бази знань. Як приклад системи з немонотонним висновком можна привести ЕС, призначену для складання перспективного плану капіталовкладення компанії. У такій системі по вашому бажанню можуть бути змінені навіть ті дані, які після висновку вже викликали спрацьовування яких-небудь правил. Іншими словами є можливість модифікувати значення атрибутів у складі фактів, що знаходяться в робочій пам'яті. Зміна фактів в свою чергу приводить до необхідності видалення з бази знань висновків, одержаних за допомогою згаданих правил. Тим самим висновок виконується повторно для того, щоб переглянути ті рішення, які були одержані на основі фактів, що піддалися зміні.

Лекція 10. Експертні системи (продовження)

База даних призначена для тимчасового зберігання фактів або гіпотез, що є проміжними рішеннями або результатом спілкування системи із зовнішнім середовищем, як яка звичайно виступає людина, що веде діалог з експертною системою.

Основу ЕС складає **підсистема (машина) логічного висновку**, яка використовує інформацію з бази знань (БЗ), генерує рекомендації за рішенням шуканої задачі. Найчастіше для представлення знань в ЕС використовуються системи **продукцій і семантичні мережі.** Для цього звичайно використовується програмно реалізований механізм дедуктивного логічного висновку (який-небудь його різновид) або механізм пошуку рішення в мережі фреймів або семантичної мережі. Припустимо, БЗ складається з фактів і правил (якщо <посылка> то <заключение>). Якщо ЕС визначає, що посилка вірна, то правило признається відповідним для даної консультації і воно запускається в дію. Запуск правила означає ухвалення висновку даного правила як складова частина

процесу консультації. Мета ЕС - вивести деякий заданий факт, який називається цільовим твердженням (тобто в результаті застосування правив добитися того, щоб цей факт був включений в робочу множину), або спростувати цей факт (тобто переконатися, що його вивести неможливо, отже, при даному рівні знань системи він є помилковим). Цільове твердження може бути або "закладено" наперед в базу знань системи, або витягується системою з діалогу з користувачем. Робота системи є послідовністю кроків, на кожному з яких з бази вибирається деяке правило, яке застосовується до поточного вмісту робочої множини. Цикл закінчується, коли виведене або спростоване цільове твердження. Цикл роботи експертної системи інакше називається логічним висновком Логічний висновок може відбуватися багатьма способами, з яких найпоширеніші - прямий порядок висновку і зворотний порядок висновку. Прямий порядок висновку - від фактів, які знаходяться в робочій множині, до висновку. Якщо такий висновок вдається знайти, то воно заноситься в робочу множину. Прямий висновок часто називають висновком, керованим даними.
Машина логічного висновку може реалізовувати міркування у вигляді:

1. Дедуктивного висновку (прямого, зворотного, змішаного);
2. Нечіткого висновку (апарат нечітких множин і нечіткої логіки);
3. Висновку вірогідності (використовування коефіцієнтів упевненості, виражаючих міру достовірності знання);
4. Уніфікації (подібно тому, як це реалізовано в Пролозі);
5. Пошуку рішення з розбиттям на послідовність підзадач;
6. Пошуку рішення з використанням стратегії розбиття простору пошуку з урахуванням рівнів абстрагування рішення або понять, з ними зв'язаних;
7. Монотонного або немонотонного міркування,
8. Міркувань з використанням механізму аргументування;
9. Асоціативного пошуку з використанням нейронних мереж;
10. Висновку з використанням механізму лінгвістичної змінної.

Підсистема спілкування служить для ведення діалогу з користувачем, в ході якого ЕС запрошує у користувача необхідні факти для процесу міркування, а також, що дає можливість користувачу якоюсь мірою контролювати і коректувати хід міркувань експертної системи.

Підсистема пояснень необхідна для того, щоб дати можливість користувачу контролювати хід міркувань і, можливо, вчитися у експертної системи. ЕС пояснює, як система одержала рішення задачі (або чому вона не одержала рішення) і які знання вона при цьому використовувала, що полегшує

експерту тестування і підвищує довіру користувача до одержаного результату. Для того, щоб непідготовлений користувач міг взаємодіяти з ЕС в неї вимагається включити засоби спілкування на природній мові. Якщо немає цієї підсистеми, експертна система виглядає для користувача як "річ в собі", рішенням якої можна або вірити або немає. Нормальний користувач вибирає останнє, і така ЕС не має перспектив для використовування.

Функціонування ЕС

ЕС працює в двох режимах: 1) режимі придбання знань; 2) режимі рішення задачі, званому також режимом консультації або режимом використовування ЕС

У режимі придбання знань спілкування ЕС здійснює експерт через посередництво інженера по знаннях. У цьому режимі експерт, використовуючи компонент придбання знань, наповнює систему знаннями, які дозволяють ЕС в режимі рішення самостійно (без експерта) вирішувати задачі з проблемної області. Експерт описує проблемну область у вигляді сукупності даних і правил. Дані визначають об'єкти, їх характеристики і значення, існуючі у області експертизи. Правила визначають способи маніпулювання з даними, характерні для даної області. Режиму придбання знань в традиційному підході до розробки програм відповідають етапи алгоритмізації, програмування і відладки, виконувані програмістом. Таким чином, на відміну від традиційного підходу у разі ЕС розробку програм здійснює не програміст, а експерт (за допомогою ЕС), що не володіє програмуванням.

У режимі консультації спілкування з ЕС здійснює кінцевий користувач, якого цікавить результат і (або) спосіб його отримання. Необхідно відзначити, що залежно від призначення ЕС користувач може не бути фахівцем в даній проблемній області (в цьому випадку він звертається до ЕС за результатом, не уміючи одержати його сам) або бути фахівцем (в цьому випадку користувач може сам одержати результат, але він звертається до ЕС з метою або прискорити процес отримання результату, або покласти на ЕС цю рутинну роботу). У режимі консультації дані про задачу через інтерфейс користувача поступають в робочу пам'ять (тут зберігаються проміжні дані вирішуваної у нинішній момент задачі). На основі вхідних даних з робочої пам'яті, загальних даних про проблемну область і правил бази знань за допомогою механізму логічного висновку формується рішення задачі. ЕС при рішенні задачі не тільки виконує наказану послідовність операцій, але і заздалегідь формує її.

Класифікація інструментальних засобів.

Інструментальні засоби підрозділяються на наступні категорії:
Мови програмування
Засоби автоматизації розробки експертних систем
Оболонки експертних систем

Мови програмування

Мови високого рівня є в руках досвідченого програміста засобом швидкого створення прототипу експертної системи, дозволяють забезпечити гнучкість процесу розробки при одночасному зниженні матеріальних витрат і скороченні термінів виконання проекту.

Мови опису правил, що породжують, об'єктно-орієнтовані мови і процедурні дедуктивні системи надають проектувальнику експертних систем значно велику свободу дій, ніж оболонки. Особливо це торкається програмування процедур управління і обробки невизначеності. Як наголошувалося вище, звичайно оболонка має вбудований режим управління і методи обробки невизначеності, які не можуть бути потім змінені в процесі побудови на її основі конкретної експертної системи. Та гнучкість, яку надають програмісту мови високого рівня, особливо важлива при створенні експериментальних систем, в яких наперед вибрати оптимальний режим управління навряд можливо.

По своєму призначенню і функціональним можливостям інструментальні програми, вживані при проектуванні експертних систем, можна розділити на чотири достатньо великих категорії: **Оболонки експертних систем**. Системи цього типу створюються, як правило, на основі якої-небудь експертної системи, що достатньо добре зарекомендувала себе на практиці. При створенні оболонки з системи-прототипу віддаляються компоненти, дуже специфічні для області її безпосереднього застосування, і залишаються ті, які не мають вузької спеціалізації. **Мови програмування високого рівня**. Інструментальні засоби цієї категорії позбавляють розробника від необхідності заглиблюватися в деталі реалізації системи — способи ефективного розподілу пам'яті, низькорівневі процедури доступу і маніпулювання даними. **Одним з найвідоміших представників таких мов є OPS5.** Ця мова проста у вивченні і надає

програмісту набагато ширші можливості, ніж типові спеціалізовані оболонки. Слід зазначити, що більшість подібних мов так і не була доведена до рівня комерційного продукту і є швидше інструментом для дослідників.

Середовище програмування, підтримуюча декілька парадигм. Засоби цієї категорії включають декілька програмних модулів, що дозволяє користувачу комбінувати в процесі розробки експертної системи різні стилі програмування. **Додаткові модулі**. Засоби цієї категорії є автономними програмними модулями, призначеними для виконання специфічних задач в рамках вибраної архітектури системи рішення проблем. Хорошим прикладом тут може служити модуль роботи з семантичною мережею. Цей модуль дозволяє відстежувати зв'язки між значеннями раніше встановлених і нових параметрів проектування в процесі роботи над проектом. Подібні модулі управління семантичною мережею можна використовувати для розповсюдження внесених змін на всі компоненти системи.

Об'єктно-орієнтовані мови

Формат правил добре узгоджується з представленням знань у формі "при виконанні умов C_b,...,C_n виконати дію A", але менш підходить для опису складних об'єктів і відносин між ними. Мови об'єктно-орієнтованого програмування надають в розпорядження програміста альтернативне програмне середовище для організації знань в термінах декларативного представлення об'єктів наочної області. Все, пов'язане з процедурною стороною рішення проблем, розподіляється між цими об'єктами, які у такому разі мають в своєму розпорядженні власні процедури і можуть спілкуватися один з одним за допомогою протоколів передачі повідомлень.

Іншим аспектом об'єктно-орієнтованого програмування є можливість використовування таких стилів представлення знань, які не зустрічаються в численні предикатів і в правилах, що породжують. Замість "розмивання" знань про об'єкт наочної області між множиною правив або аксіом, на які вони посилаються, ці знання концентруються в єдиному місці — в програмному описі об'єкту. Існує безліч систем, в яких обмін інформацією може бути представлений через обмін повідомленнями між їх комп'ютерними уявленнями, і такий зв'язок з технологією моделювання є дуже важливою гідністю даного підходу. Об'єктно-орієнтоване програмування інтегрує символічні обчислення в операційне середовище, що базується на засобах графічного інтерфейсу, —

меню, піктограми і т.п. Хоча саме по собі оснащення експертної системи цими засобами і не вирішує проблему її прозорості для користувача, в руках умілого програміста вони дозволяють краще представити користувачу процеси, що відбуваються в системі.

Основна складність у використовуванні засобів об'єктно-орієнтованого програмування — з'ясувати для себе, що саме повинен представляти програмний об'єкт по відношенню до наочної області. Але для того, щоб упровадити об'єктно-орієнтований стиль в проектування експертних систем, потрібно задуматися над тим, як співвіднести програмні об'єкти з абстрактними поняттями і категоріями наочної області. Об'єкти повинні представляти факти і цілі, набори правив або окремі гіпотези. Тому далеко не очевидно, якими повідомленнями повинні обмінюватися такі об'єкти і яке значення повинен вкладатися в ці повідомлення.

Smalltalk — об'єктно-орієнтована мова програмування з динамічною типізацією, продовжує активно розвиватися.

Smalltalk зробив великий вплив на розвиток багатьох інших мов, таких як: Objective-C, Actor, Java і Ruby. Основними ідеями Smalltalk’а є:

«Все — об'єкти». Рядки, цілі числа, логічні значення, визначення класів, блоки коду, стеки, пам'ять — все представляється у вигляді об'єктів. Виконання програми складається з посилок повідомлень між об'єктами. Будь-яке повідомлення може бути послане будь-якому об'єкту; об'єкт-одержувач визначає, чи є це повідомлення правильним, і що треба зробити, щоб його обробити.

Однією з особливостей Smalltalk’а є те, що традиційні конструкції: if-then-else, for, while, і т.д. не є частиною мови. Всі вони реалізовані за допомогою об'єктів. Наприклад, рішення ухвалюється за допомогою посилки повідомлення ifTrue: логічному об'єкту, і передає управління фрагменту коду, якщо логічне значення істинне. Є всього три конструкції:

посилка повідомлення об'єкту;

привласнення об'єкту змінної;

повернення об'єкту з методу;

і декілька синтаксичних конструкцій для визначення літеральних об'єктів і тимчасових змінних.

Мова логічного програмування PROLOG

Пролог (Prolog) — мова логічного програмування, заснований на логіці діз'юнктов Хорна, що є підмножиною логіки предикатів першого порядку. Будучи декларативною мовою програмування, Пролог сприймає як програма деякий опис задачі, і сам виробляє пошук рішення, користуючись механізмом бектрекинга і уніфікацією.

Вбудований в PROLOG режим управління використовує стратегію зворотного логічного висновку. Таблиці знань і інші дані можна представити за допомогою тверджень. Такі структури даних, як графи і дерева, можна організувати за допомогою фраз мови PROLOG, які містять комплексні терми. Мовні засоби PROLOG дозволять програмісту розробити власний механізм обробки невизначеності, причому не виключається і використовування коефіцієнтів упевненості.

З практичної точки зору, користуючись мовою PROLOG, програміст як "безкоштовний додаток" дістає в своє розпорядження наступні можливості:

- індексовану базу даних фраз, які можна використовувати для представлення правил, процедур або даних;
- універсальний механізм зіставлення, який дозволяє виконувати зіставлення даних і шаблонів, що включають змінні, і повертати підстановку, яка може забезпечити їх збіг;
- стратегію управління (пошук в глибину — depth-first search), засновану на правилах низхідного пошуку (фрази, які розміщені в базі даних ближче до "голови", обробляються першими) і обчисленні зліва направо (підмети обробляються в тому порядку, в якому вони перераховані в списку).

Дійсно, дедуктивну систему, що породжує, досить ПРОСТО емулювати на мові PROLOG (як використовується емуляція) Можна без особливих зусиль розробити і простий інтерпретатор, що реалізовує стратегію побудови прямого ланцюжка висновку. Модифікація робочої пам'яті виконується операторами assert і retract, які додають або видаляють формули з бази даних. Ви вже знаєте, як можна організувати локальне управління ходом процесу в системі, заснованій на фреймах, як організувати обробку значень за умовчанням і виключень, хоча ці методи і не вписуються в стандартну логіку.

Успішний досвід застосування ідей логічного програмування, зокрема створення програми МЕСНО, продемонстрував ряд явних відхилень від синтаксису числення предикатів першого порядку і його процедурної

інтерпретації в стандартній версії PROLOG. Деякі семантичні і синтаксичні обмеження в програмах MECHO і PLANNER дотепер не подолані в системах, що базуються на мовах логічного програмування.

Мова функціонального програмування LISP

Лісп (LISP, від англ. LISt Processing — «обробка списків») — сімейство мов програмування, заснованих на представленні програми системою лінійних списків, які притому є основною структурою даних мови. Лісп вважається другою після Фортрану старою високорівневою мовою програмування. Традиційний Лісп має строгу динамічну систему типів, містить імперативні властивості, але загалом заохочує функціональну парадигму програмування. Існує об'єктно-орієнтоване розширення мови — CLOS.

Основна особливість Ліспу — представлення програми у вигляді списків — визначає однорідність і простоту синтаксису. Зовні початковий код програми на Ліспі відрізняється великою кількістю круглих дужок; редагування програм значно спрощується використовуванням текстового редактора, що підтримує автоматичне вирівнювання коду, підсвічування відповідних пар дужок і команди роду «перейти через список управо». Добре пристосований до кодування на Ліспі редактор Emacs (велика частина якого написана на Ліспі), крім того в комерційних реалізаціях (наприклад в LispWorks) IDE містить зручних редакторів, що зберігають всі достоїнства Emacs.

Одним з найпоширеніших діалектів Ліспу є Common Lisp. Його реалізації існують для більшості платформ. Є реалізації поширювані вільно під ліцензіями, що дозволяють використовувати Лісп без обмежень в комерційних додатках, окрім цього існують комерційні реалізації, що пропонують додаткові можливості (IDE, додаткові бібліотеки і т. п.), зокрема технічну підтримку. З повним списком реалізацій можна ознайомитися тут: Common Lisp Implementation.

Лекція 11. Багатофункціональні програмні середовища

Багатофункціональні програмні середовища дозволяють досвідченому програмісту експериментувати при рішенні нових класів проблем, вибираючи відповідні поєднання різних методів, представлених в наявному модульному наборі. Оскільки не існує єдиної універсальної мови представлення знань для

довільної експертної системи, у розробників виникає бажання об'єднати декілька різних схем уявлення, особливо на етапі створення прототипу. Хоча вичерпної теорії таких гібридних систем і не існує, експерименти з різними схемами уявлення і логічного висновку показали, що кожна з них має свої слабкі сторони. Тому зрозуміле бажання об'єднати різні методики так, щоб достоїнства одних компенсували слабкості інших.

Структуровані об'єкти, наприклад фрейми, виявляються зручнішим засобом для зберігання і маніпулювання описами об'єктів наочної області, але застосування таких знань вимагає включення в програму фрагментів програмного коду (наприклад, на мові LISP), які потім важко аналізувати. Раціональне зерно в перших спробах звести разом стилі, засновані на правилах і фреймах, полягало в тому, щоб об'єднати здатність представляти об'єкти, характерні для фреймів, з можливостями зв'язувати умови і дії за допомогою правил, що породжують.

Одним з перших багатофункціональних середовищ штучного інтелекту є **LOOPS**. у якій в рамках єдиної архітектури обміну повідомленнями були об'єднані чотири парадигми програмування:

1. Процедурно-орієнтоване програмування. Ця парадигма була представлена мовою LISP, в якому активним компонентом є процедури, а пасивним — дані, не дивлячись на те, що в LISP процедури самі по собі також є даними, оскільки мають вид списків. В рамках єдиного середовища процедури можуть бути використані для обробки зовнішніх даних, зокрема зміни значень загальнодоступних змінних.
2. Програмування, орієнтоване на правила. Ця парадигма аналогічна попередній, але роль процедур виконують правила "умова-дія". У середовищі LOOPS набори правил самі по собі є об'єктами, які можна рекурсивно вкладати один в інший. Таким чином, частина "дія" одного правила, у свою чергу, може активізувати підлеглий набір правил. З безліччю правил зв'язуються управляючі компоненти, за допомогою яких в простій формі виконується дозвіл конфліктів.
3. Об'єктно-орієнтоване програмування. Структуровані об'єкти володіють властивостями і процедур, і даних, причому побічні ефекти звичайно локалізуються в межах об'єкту. Обробка поступаючих повідомлень приводить до передачі даних або зміни їх значень, але всі маніпуляції даними виконуються під управлінням того компоненту, який звернувся до

об'єкту. При цьому зухвалий об'єкт абсолютно не цікавить, як зберігаються дані і як вони модифікуються усередині об'єкту.

4. Програмування, орієнтоване на дані. Доступ до даних і оновлення даних запускає певні процедури, причому не має значення, чому змінений компонент даних, — чи то це результат побічного ефекту, чи то результат дії інших процедур. Із змінними, в яких зберігаються значення даних, зв'язуються певні процедури, подібно тому, як це робиться в слотах фрейма, причому такі змінні часто називають активними величинами. У таких додатках, як моделювання, цей стиль програмування виявляється досить продуктивним, оскільки дозволяє розповсюдити ефект зміни якого-небудь компоненту на інші, з ним зв'язані.

В рамках основної об'єктно-орієнтованої парадигми модулі середовища, підтримуючі різні стилі програмування, можна комбінувати. Звично умови в правилах, що породжують, і логічні фрази зв'язуються із значеннями слотів структурованих об'єктів, а правила модифікують значення цих слотів. Саме такий стиль об'єднання парадигм в даний час реалізований в мові CLIPS.

CLIPS як багатофункціональне середовище програмування(інженерії знань)

Окрім підтримки інтерпретатора правил CLIPS, що породжують, володіє наступними функціональними можливостями:

- для визначення стандартних функцій використовується синтаксис, подібний LISP;
- надає в розпорядження розробника родові функції, аналогічні мультиметодам CLOS;
- має в своєму розпорядженні вбудовану об'єктно-орієнтовану мову COOL, яка, на відміну від CLOS, включає і засоби підтримки обміну повідомленнями.

Звернення до стандартних функцій допускається включати в праву частину правил і в цьому випадку вони виконуються так, як якби були компонентом дій, специфікованих в правилі. Функції викликаються або з метою одержати побічний ефект, або для використовування результату, що явно повертається функцією, який може бути збережений за допомогою оператора привласнення. Для роботи із змінними в цьому випадку використовується той же синтаксис, що і в мові опису правив.

У програмі на мові CLIPS можна викликати і функції, написані на мові З, хоча це і виконується декілька незвичайно. Виконуюча система CLIPS може виступати як упроваджений додаток, тобто програма на CLIPS може бути скомпільована і укомпонована з програмою на мові З, яка викликатиме CLIPS-фрагменти як підпрограми. Це дозволяє упроваджувати функції штучного інтелекту в компоненти великих програмних комплексів.

Засоби автоматизації розробки експертних систем.

Додаткові модулі. Під додатковими модулями розуміються ті корисні програми, які можна виконувати разом з додатком. Як правило, такі програми реалізують деякі спеціальні функції, як би "знімаючи їх з полиці", причому для звернення до таких функцій не потрібне що-небудь програмувати в основному додатку або займатися його індивідуальною настройкою. Одним з прикладів такого роду додаткового модуля може служити програмний пакет Simkit з комплекту середовища КЕЕ. Цей пакет дозволяє оснастити експертну систему методами моделювання.

Тенденція використовування додаткових модулів швидше за все розвиватиметься, оскільки користувачі експертних систем часто мають потребу в різного роду додаткових функціональних можливостях, специфічних для конкретного додатку, а також в можливості інтегрувати експертну систему з програмними продуктами інших класів. На практиці експертна система часто використовується разом в базою даних або системою управління рухом робота, одержує інформацію від систем обробки сигналів або пакетів статистичної обробки.

Оболонки експертних систем.

Оболонки експертних систем - програмний продукт, що володіє засобами представлення знань для певних наочних областей. Задача користувача полягає не в безпосередньому програмуванні, а у формалізації і введенні знань з використанням наданих оболонкою можливостей. Недоліком цих систем можна рахувати неможливість обхвату однією системою всіх існуючих наочних областей. Прикладом можуть служити ІНТЕРЕКСПЕРТ, PC+, VP-Expert.

Оболонка, shell - базовий елемент операційної системи, визначаючий інтерпретацію команд і дій користувача.

CLIPS (Мова З, інтегрована Продукційна Система) - OPS-ПОДІБНА продукційна система, що використовує висновок від фактів до мети, написана на З в ANSI NASA. Механізм логічного висновку CLIPS включає супровід, динамічне додавання правив і стратегії вирішення протиріч, що настроюються. CLIPS, включаючи динамічну версію, легко вбудовується в інші прикладні програми. CLIPS включає об'єктно-орієнтовану мову, названу COOL(Об'єктно-орієнтована Мова CLIPS), який прямо інтегрований з механізмом логічного висновку.

FuzzyCLIPS 6.02 - версія CLIPS, оболонка експертної системи, заснована на правилах, використовується для уявлення і управління нечіткими фактами і правилами. На додаток до функціональних можливостей CLIPS, FuzzyCLIPS може мати справу з точними, нечіткими (або неточними) знаннями, складними міркуваннями, які можна вільно змішувати в правилах і фактах експертної системи. Система використовує дві базисні концепції об неточності, нечіткість і невизначеність.

Етапи створення експертних систем

У проектуванні експертних систем можна виділити наступні етапи:

1. Ідентифікація

1.1. Визначення учасників і їх ролей в процесі створення і експлуатації експертної системи

В процесі створення експертної системи можуть брати участь наступні фахівці: інженери по знаннях, експерти, програмісти, керівник проекту, замовники (кінцеві користувачі). При реалізації порівняно простих експертних систем програмістів може не бути. Роль інженера по знаннях - вивуджування професійних знань з експертів і проектування бази знань експертної системи і її архітектури. Програміст необхідний при розробці спеціалізованого для даної експертної системи програмного забезпечення, коли відповідного стандартного (наприклад, оболонки для створення експертних систем) не існує або його можливостей не достатньо і потрібні додаткові модулі.

В процесі експлуатації можуть брати участь кінцеві користувачі, експерти, адміністратор.

1.2. Ідентифікація проблеми

На цьому етапі розробники повинні відповісти на ряд питань, що визначають особливості вирішуваних експертами, а отже, майбутньою експертною системою, задач. Ці особливості визначать і особливості архітектури експертної системи, формованої на подальших етапах. До цих питань відносяться наступні:

- Який клас задач повинна вирішувати ЕС
- Як ці задачі можуть бути охарактеризовані або визначені
- Які можна виділити підзадачі
- Які початкові дані повинні використовуватися для вирішення
- Які поняття і взаємозв'язки між ними використовуються при рішенні задачі експертами
- Який вигляд має рішення, і які концепції використовуються в ньому
- Які аспекти досвіду експерта істотні для вирішення задачі
- Яка природа і об'їм знань, необхідних для вирішення задачі
- Які перешкоди зустрічаються при рішенні задач
- Як ці перешкоди можуть впливати на рішення задачі

1.3. Визначення необхідних ресурсів - тимчасових, людських, матеріальних

1.4. Визначення цілей

Як цілі, переслідувані при створенні експертних систем, можуть бути: підвищення швидкості ухвалення рішення, підвищення якості рішень, тиражування досвіду експертів і т.п.

2. Концептуалізація

На даному етапі проводиться змістовний аналіз проблемної області, виявляються використовувані поняття і їх взаємозв'язки, визначаються методи рішення задач. Цей етап завершується створенням моделі наочної області (ПО), що включає основні концепти і відносини. На етапі концептуалізації визначаються наступні особливості задачі: типи доступних даних; початкові і виводяться дані, підзадачі загальної задачі; використовувані стратегії і гіпотези; види взаємозв'язків між об'єктами ПЗ, типи використовуваних відносин (ієрархія, причина — слідство, частина — ціле і т.п.); процеси, використовувані в ході рішення; склад знань, використовуваних при рішенні задачі; типи обмежень, що накладаються на процеси, використовувані в ході рішення; склад знань, використовуваних для обгрунтування рішень.

3. Формалізація

Всі ключові поняття і відносини виражаються на деякій формальній мові, яка або вибирається з числа вже існуючих, або створюється наново. Іншими словами, на даному етапі визначаються склад засобів і способи представлення декларативних і процедурних знань, здійснюється це уявлення і у результаті формується опис рішення задачі ЕС на запропонованій (інженером по знаннях) формальній мові. Виходом етапу формалізації є опис того, як дана задача може

бути представлена у вибраному або розробленому формалізмі. Сюди відноситься вказівка способів представлення знань (фрейми, сценарії, семантичні мережі і т.д.) і визначення способів маніпулювання цими знаннями (логічний висновок, аналітична модель, статистична модель і ін.) і інтерпретації знань.

Лекція 11. Експертні системи ЕС

4. Реалізація прототипної версії

Мета цього етапу — створення одного або декількох прототипів ЕС, вирішальних необхідні задачі. Потім на даному етапі за наслідками тестування і досвідченої експлуатації створюється кінцевий продукт, придатний для промислового використовування. Розробка прототипу полягає в програмуванні його компонентів або виборі їх з відомих інструментальних засобів і наповненні бази знань.

Головне в створенні прототипу полягає в тому, щоб цей прототип забезпечив перевірку адекватності ідей, методів і способів представлення знань вирішуваним задачам. Створення першого прототипу повинне підтвердити, що вибрані методи рішень і способи уявлення придатні для успішного вирішення, принаймні, ряду задач з актуальної наочної області, а також продемонструвати тенденцію до отримання високоякісних і ефективних рішень для всіх задач наочної області у міру збільшення об'єму знань.

Після розробки першого прототипу ЕС-1 круг пропонованих для вирішення задач розширяється, і збираються побажання і зауваження, які повинні бути враховані в черговій версії системи ЕС-2. Здійснюється розвиток ЕС-1 шляхом додавання "дружнього" інтерфейсу, засобів для дослідження бази знань і ланцюжків висновків, що генеруються системою, а також засобів для збору зауважень користувачів і засобів зберігання бібліотеки задач, вирішених системою.

Виконання експериментів з розширеною версією ЕС-1, аналіз побажань і зауважень служать відправною крапкою для створення другого прототипу ЕС-2. Процес розробки ЕС-2 ітеративний. Він може продовжуватися від декількох місяців до декількох років залежно від складності наочної області, гнучкості вибраного представлення знань і ступеня відповідності управляючого механізму

вирішуваним задачам (можливо, потрібно розробка ЕС-3 і т.д.). При розробці ЕС-2, окрім перерахованих задач, розв'язуються наступні:

– аналіз функціонування системи при значному розширенні бази знань; дослідження можливостей системи в рішенні ширшого круга задач і вживання заходів для забезпечення таких можливостей;

–аналіз думок користувачів про функціонування ЕС;

–розробка системи введення-висновку, здійснюючої аналіз або синтез пропозицій обмеженої природної мови, дозволяючої взаємодіяти з ЕС-2 у формі, близькій до форми стандартних підручників для даної області.

Якщо ЕС-2 успішно пройшла етап тестування, то вона може класифікуватися як промислова експертна система. Етап тестування.

В ході даного етапу виробляється оцінка вибраного способу представлення знань в ЕС в цілому. Для цього інженер по знаннях підбирає приклади, що забезпечують перевірку всіх можливостей розробленої ЕС.

Розрізняють наступні джерела невдач в роботі системи: тестові приклади, введення-висновок, правила висновку, управляючі стратегії. Показові тестові приклади є найочевиднішою причиною невдалої роботи ЕС. У гіршому разі тестові приклади можуть виявитися взагалі поза наочною областю, на яку розрахована ЕС, проте частіше безліч тестових прикладів виявляється дуже однорідною і не охоплює всю наочну область. Тому при підготовці тестових прикладів слід класифікувати їх по підпроблемах наочної області, виділяючи стандартні випадки, визначаючи межі важких ситуацій і т.п.

Введення-висновок характеризується даними, придбаними в ході діалогу з експертом, і висновками, пред'явленими ЕС в ході пояснень. Методи придбання даних можуть не давати необхідних результатів, оскільки, наприклад, задавалися неправильні питання або зібрана не вся необхідна інформація. Крім того, питання системи можуть бути важкими для розуміння, багатозначними і не відповідними знанням користувача. Помилки при введенні можуть виникати також через незручну для користувача вхідну мову. У ряді додатку для користувача зручне введення не тільки в друкарській, але і в графічній або звуковій формі.

Вихідні повідомлення (висновки) системи можуть виявитися незрозумілі користувачу (експерту) з різних причин. Наприклад, їх може бути дуже багато або, навпаки, дуже мало. Також причиною помилок може бути невдала організація, впорядкованість висновків або невідповідний користувачу рівень абстракцій з незрозумілою йому лексикою.

Найпоширеніше джерело помилок в міркуваннях торкається правил висновку. Важлива причина тут часто криється у відсутності обліку взаємозалежності сформованих правил. Інша причина полягає в помилковості, суперечності і неповноті використовуваних правил. Якщо невірна посилка правила, то це може привести до вживання правила в невідповідному контексті. Якщо помилкова дія правила, то важко передбачити кінцевий результат. Правило може бути помилкове, якщо при коректності його умови і дії порушена відповідність між ними.

Нерідко до помилок в роботі ЕС приводять вживані управляючі стратегії. Зміна стратегії буває необхідна, наприклад, якщо ЕС аналізує суть в порядку, відмінному від "природного" для експерта. Послідовність, в якій дані розглядаються ЕС, не тільки впливає на ефективність роботи системи, але і може приводити до зміни кінцевого результату. Так, розгляд правила А до правила В здатне привести до того, що правило В завжди ігноруватиметься системою. Зміна стратегії буває також необхідна і у разі неефективної роботи ЕС. Крім того, недоліки в управляючих стратегіях можуть привести до надмірно складних висновків і пояснень ЕС.

Критерії оцінки ЕС залежать від точки зору. Наприклад, при тестуванні ЕС-1 головним в оцінці роботи системи є повнота і безпомилковість правил висновку. При тестуванні промислової системи превалює точка зору інженера по знаннях, якого в першу чергу цікавить питання оптимізації уявлення і маніпулювання знаннями. І, нарешті, при тестуванні ЕС після досвідченої експлуатації оцінка виробляється з погляду користувача, зацікавленого в зручності роботи і отримання практичної користі

5. Етап досвідченої експлуатації

На цьому етапі перевіряється придатність ЕС для кінцевого користувача. Придатність ЕС для користувача визначається в основному зручністю роботи з нею і її корисністю. Під корисністю ЕС розуміється її здатність в ході діалогу визначати потреби користувача, виявляти і усувати причини невдач в роботі, а також задовольняти вказані потреби користувача (вирішувати поставлені задачі). У свою чергу, зручність роботи з ЕС має на увазі природність взаємодії з нею (спілкування в звичному, не стомливому користувача вигляді), гнучкість ЕС (здатність системи настроюватися на різних користувачів, а також враховувати зміни в кваліфікації одного і того ж користувача) і стійкість системи до помилок (здатність не виходити з ладу при помилкових діях недосвідченого користувачах).

В ході розробки ЕС майже завжди здійснюється її модифікація. Виділяють наступні види модифікації системи: переформулювання понять і вимог, переконструювання представлення знань в системі і удосконалення прототипу. Експертні системи, паралельні і послідовні рішення.

Як ми можемо помітити, в більшості алгоритмів розпізнавання образів мається на увазі, що до початку роботи алгоритму вже відома вся вхідна інформація, яка переробляється паралельно. Проте її отримання часто вимагає певних зусиль. Та і наші спостереження за реальними експертами підтверджують, що часто вони ставлять два-три питання, після чого роблять правильні висновки. Уявіть собі, якби лікар (експерт у області медицини) перед постановкою діагнозу "ангіна" примушував би пацієнта пройти повне обстеження аж до кулоноскопії і пункції хребта.

Відповідно більшість алгоритмів модифікується, щоб забезпечити виконання наступних умов:
алгоритми повинні працювати в умовах неповної інформації (послідовно);
послідовність запиту інформації повинна бути оптимальна по критеріях швидкості отримання результату і (або) якнайменшої трудомісткості (хворобливості, вартості і т.д.) отримання цієї інформації.

Однією з можливих стратегій для оптимізації запитів є стратегія отримання в першу чергу тієї інформації, яка підтверджує або спростовує найвірогідніший на даний момент результат. Іншими словами ми намагаємося підтвердити або спростувати наші припущення (зворотний висновок)

6. Прототип і життєвий цикл експертної системи

При розробці практично всіх інструментальних засобів за основу приймається методологія автоматизації проектування на базі використовування прототипів. По відношенню до програмного забезпечення термін прототип означає "працюючу модель програми, яка функціонально еквівалентна підмножині кінцевого продукту". Ідея полягає в тому, щоб на ранній стадії роботи над проектом розробити спрощену версію кінцевої програми, яка могла б послужити доказом продуктивності основних ідей, встановлених в підставу проекту. Прототип повинен бути здатний вирішувати яку-небудь з нетривіальних задач, характерних для заданої області застосування. На основі аналізу досвіду роботи з прототипом розробники можуть уточнити вимоги до системи в цілому і її Основним функціональним характеристикам.

Працездатність прототипу може послужити очевидним доказом можливості рішення проблем за допомогою створюваної системи ще до того, як на її розробку будуть витрачені значні засоби.

Після всебічного аналізу прототип відкладається убік і починається розробка робочої версії програми, яка повинна вирішувати весь комплекс задач, визначених в специфікації проекту. Процес розробки експертної системи, як правило, складається з послідовності окремих етапів, на яких нарощуються можливості системи, причому кожний з етапів підрозділяється на фази проектування, реалізації, компоновки і тестування. В результаті після кожного етапу нарощування можливостей у розпорядженні користувача є система, яка здатна справлятися зі все більш складними варіантами проблеми.

Така методика проектування декілька відрізняється від методики розробки програм інших видів. При створенні більшості програмних продуктів частіше використовується інша модель процесу - спочатку розробляється специфікація продукту, потім виконується планування, проектування компонентів, їх реалізація, компоновка комплексу і тестування кінцевого варіанту. Той факт, що при розробці експертних систем є можливість спочатку побудувати і всесторонньо випробувати прототип, дозволяє уникнути безлічі переробок в процесі створення робочої версії системи. Але технологія послідовного нарощування функціональних можливостей таїть в собі і проблему інтеграції нових функцій з реалізованими в попередніх варіантах. Інструментальні засоби розробки експертних систем і створювалися, в першу чергу, з метою подолання виникаючих при цьому складнощів на основі модульного представлення знань.

При незадовільному функціонуванні прототипу експерт і інженер по знаннях мають нагоду оцінити, що саме буде включене в розробку остаточного варіанту системи.

Якщо спочатку вибрані об'єкти або властивості виявляються невідповідними, їх необхідно змінити. Можна зробити оцінку загального числа евристичних правил, необхідних для створення остаточного варіанту експертної системи. Іноді при розробці промислової системи виділяють додаткові етапи: **демонстраційний** прототип; **дослідницький** прототип; **діючий** прототип; **промислова** система.

Проте частіше реалізується плавний перехід від демонстраційного прототипу до промислової системи, при цьому, якщо програмний

інструментарій вибраний вдало, необов'язковий перепис іншими програмними засобами.

Поняття ж комерційної системи в нашій країні входить в поняття промисловий програмний продукт, або промислової ЕС в цій роботі.

Перехід від прототипу до промислової експертної системи.

Демонстраційний прототип ЕС. Система вирішує частину задач, демонструючи нездатність підходу (декілька десятків правив або понять).

Дослідницький прототип. ЕС Система вирішує більшість задач, але не стійка в роботі і не повністю перевірена [декілька сотень правив або понять).

Діючий прототип ЕС. Система надійно вирішує всі задачі на реальних прикладах, але для складної задачі вимагає багато часу і пам'яті.

Промислова система. Система забезпечує високу якість рішень при мінімізації необхідного часу і пам'яті: переписується з використанням ефективніших засобів представлення знань. **Комерційна** система Промислова система, придатна до продажу, тобто добре документована і забезпечена сервісом

Основне на цьому етапі полягає в додаванні великого числа додаткових евристик. Ці евристики звичайно збільшують глибину системи, забезпечуючи більше число правил для важковловимих аспектів окремих випадків. В той же час експерт і інженер по знаннях можуть розширити обхват системи, включаючи правила, що управляють додатковими підзадачами або додатковими аспектами експертної задачі (метазнання).

Після встановлення основної структури ЕС інженер по знаннях приступає до розробки і адаптації інтерфейсів, за допомогою яких система спілкуватиметься з користувачем і експертом. Необхідно звернути особливу увагу на мовні можливості інтерфейсів, їх простоту і зручність для управління роботою ЕС. Система повинна забезпечувати користувачу можливість легенею і природним чином питати незрозуміле, припиняти роботу і т.д. Зокрема, можуть виявитися корисними графічні уявлення.

Життєвий цикл експертної системи складається з етапів розробки і супроводу. На етапі розробки створюється програмне забезпечення і база знань експертної системи, на етапі супроводу відбувається виправлення виявлених помилок і поповнення бази знань без участі розробників (якщо останнє допускається архітектурою експертної системи).

Застосування експертної системи з базою знань, незмінної в процесі експлуатації, можливе при достатньо стабільній протягом довгого часу наочній

області, в якій розв'язуються задачі. Прикладами таких наочних областей є розділи математичного аналізу, опис правил діагностики різних захворювань. Прикладами областей застосування, вимагаючих гнучкості з боку створення і поповнення бази знань, є: планування виробництва, проектування і діагностика у області електроніки, обчислювальної техніки і машинобудування.

Лекція 12. Автоматичне міркування. Основні механізми дедукції (логічного висновку)

Примусити машину виконувати простий процес міркувань нескладно. Ця здатність складає ядро більшості ЕС. В цілому знання ЕС повинні бути введені у формі правил для того, щоб могли бути реалізовані процеси автоматичного міркування. Як вже мовилося вище, та частина програми, яка застосовує правила, а потім управляє міркуваннями, одержала назву механізму висновку.

Логічний висновок - центральне поняття в ЕС. Воно має два аспекти: 1) використовування міркувань для знаходження різних припущень, які обумовлені наявними фактами і правилами, або 2) для вивчення висновків, які представляють інтерес і можуть бути істинними. Перший метод носить назву **прямого ланцюжка міркувань,** другий - **зворотного ланцюжка міркувань**. Майже всі експертні системи побудовані або на першому, або на другому методі.

Прямий ланцюжок міркувань припускає використовування правил для дедукції (логічного висновку) нових фактів, а також фактів, які існували і раніше, але можуть бути зроблені явними за допомогою застосування правил. Механізм висновку циклічно проглядає всі правила, досліджуючи їх по черзі з'ясовувавши, чи є інформація на лівій стороні істинною. Якщо вона істинна, механізм висновку додає факт на правій стороні правила до істинних фактів, що зберігаються. Потім він переходить до наступного правила, і процес повторюється. Перевіривши всі правила, механізм починає роботу наново. Т.ч. механізм висновку працює для поповнення початкового запасу істинних фактів фактами, які маються на увазі за допомогою набору правил.

Проте, в системі з реальним числом правил є так багато фактів, що маються (прихованих) на увазі, що, будучи знайдені, вони затьмарять ті декілька фактів, які дійсно представляють інтерес і є корисними. У подібних випадках система повинна мати додатковий механізм (залежно від проблеми) для того, щоб визначити, з яким наступним правилом їй належить працювати. Замість

простого циклу, який механічно вибирає правила, механізм вибору чітко встановлює пріоритет вибору тих або інших фактів і правил. ЕС на базі прямого ланцюжка міркувань найчастіше застосовують в плануванні або проектуванні.

Приклад прямого ланцюжка міркувань.

Для ЕС фондової біржі можна було б скористатися, наприклад, такими правилами:

1 ЯКЩО ПРОЦЕНТНІ СТАВКИ - ПАДАЮТЬ, ТО РІВЕНЬ ЦІН НА БІРЖІ - РОСТЕ
2 ЯКЩО ПРОЦЕНТНІ СТАВКИ - РОСТУТЬ, ТО РІВЕНЬ ЦІН НА БІРЖІ - ПАДАЄ
3 ЯКЩО ВАЛЮТНИЙ КУРС ДОЛАРА - ПАДАЄ, ТО ПРОЦЕНТНІ СТАВКИ - РОСТУТЬ
4 ЯКЩО ВАЛЮТНИЙ КУРС ДОЛАРА - РОСТЕ, ТО ПРОЦЕНТНІ СТАВКИ - ПАДАЮТЬ

Припустимо, що створена фірма, що дає на основі цих правил консультації у області біржових операцій. Клієнт повідомив, що валютний курс долара падає по відношенню до основних валют інших країн, і попросив раду Мета полягає у виборі правильної поведінки на біржі, але чи залишиться клієнт у виграші, залежить від поки що не певних умов.

Для нашого прикладу необхідно, щоб в умовній частині якого-небудь правила містилася б умова ВАЛЮТНИЙ КУРС ДОЛАРА ПАДАЄ. Така умова міститься в правилі 3. Відповідно до цього правила можна зробити висновок про зростання процентних ставок. Але про валютний курс згадується ще в правилі 4, але умова, записана в цьому правилі, не відповідає початковому стану падіння курсу долара, і тому правило 4 в подальших міркуваннях не братиме участь. Правило 3, у свою чергу, породжує нову ситуацію: ПРОЦЕНТНІ СТАВКИ - РОСТУТЬ. Необхідно перевірити, чи не приведе вона до інших висновків. Видно, що в правилі 1 відповідної умови немає, а в правилі 2 є. Виникає нова ситуація: РІВЕНЬ ЦІН НА БІРЖІ - ПАДАЄ і міркування продовжуються.

Ще раз виконується перевірка всіх правил, але ні в одному правилі в умовній частині не згадується рівень цін на біржі, і на цьому міркування закінчуються. Клієнту можна сказати наступне: коли обмінний курс долара падає, ростуть процентні ставки і рівень цін на біржі падає.

Розглянутий приклад ілюструє роботу типової системи прямих міркувань:

1. Система містить опис ряду ситуацій.

2. Для кожної ситуації система шукає в базі знань правила, в умовній частині яких міститься відповідна умова.
3. Відповідно до констатуючої частини (частиною ТЕ) кожне правило може генерувати нові ситуації, які додаються до вже існуючих.
4. Система обробляє кожну ситуацію, що знов згенерує. За наявності хоча б однієї такої ситуації виконуються дії, починаючи з пункту 2. Міркування закінчуються, коли немає більше необроблених ситуацій.

Зворотний ланцюжок міркувань. Зворотної ланцюжок міркувань називається тому, що починається з події, що вже відбулася, і йде до його витоків. Програмні засоби, що працюють за принципом зворотного ланцюжка міркувань, призначені для пошуку причин по вже відомому результату. Ланцюжок виконується за допомогою серії питань, які система ШІ задає людині. Система, що реалізовує прямий ланцюжок міркувань, на підставі наявних умов робить можливі логічні висновки; система, що реалізовує зворотний ланцюжок міркувань по наявних висновках, шукає необхідні для них умови.

Розробка ЕС, в якій реалізується зворотний ланцюжок міркувань.

Задача: директора крупної технічної фірми дійшов відвідувач, охочий влаштується на роботу. Директор має в своєму розпорядженні відомості про його кваліфікацію, про потреби фірми у фахівцях і загальному положенні справ у фірмі. Йому потрібно вирішити, яку посаду у фірмі може посісти відвідувач.

На перший погляд задача не дуже складна, але на рішення директора впливає багато чинників. Наприклад, претендент працює в даній області недавно, але вже зробив важливе відкриття або він закінчив учбовий заклад з посередніми оцінками, але декілька років працював за фахом. У даній ситуації люди поводяться по-різному, і хоча для того, щоб одержати роботу необхідно, задовольняти певним критеріям, в біографії претендента можуть бути самі різні факти, аналіз яких допоможе підібрати для нього відповідну посаду.

Оскільки в задачі треба вибрати один з декількох можливих варіантів (посад), для її вирішення можна скористатися зворотним ланцюжком міркувань. Насправді відповідь вже існує. Перед директором сидить людина і всіма силами прагне справити на нього хороше враження. Якщо директора ця людина влаштовує, для нього потрібно підібрати відповідну посаду. Директору

необхідно поставити відвідувачу такі питання, відповіді на які дають можливість зробити правильний вибір.

Задача поставлена. Тепер потрібно її наочно уявити. Для опису подібних задач звичайно використовуються діаграми, які називаються деревами рішень. Дерева рішень дають необхідну наочність і дозволяють прослідити хід міркувань.

Діаграми називаються деревами рішень тому, що, подібно справжньому дереву, мають гілки. Гілки дерев рішень закінчуються висновками. Для нашого прикладу висновок полягає в тому, чи запропонує директор посада поступаючому на роботу, і якщо так, то яку. Багато задач складне, і їх непросто уявити. Дерево рішень допомагає подолати ці труднощі.

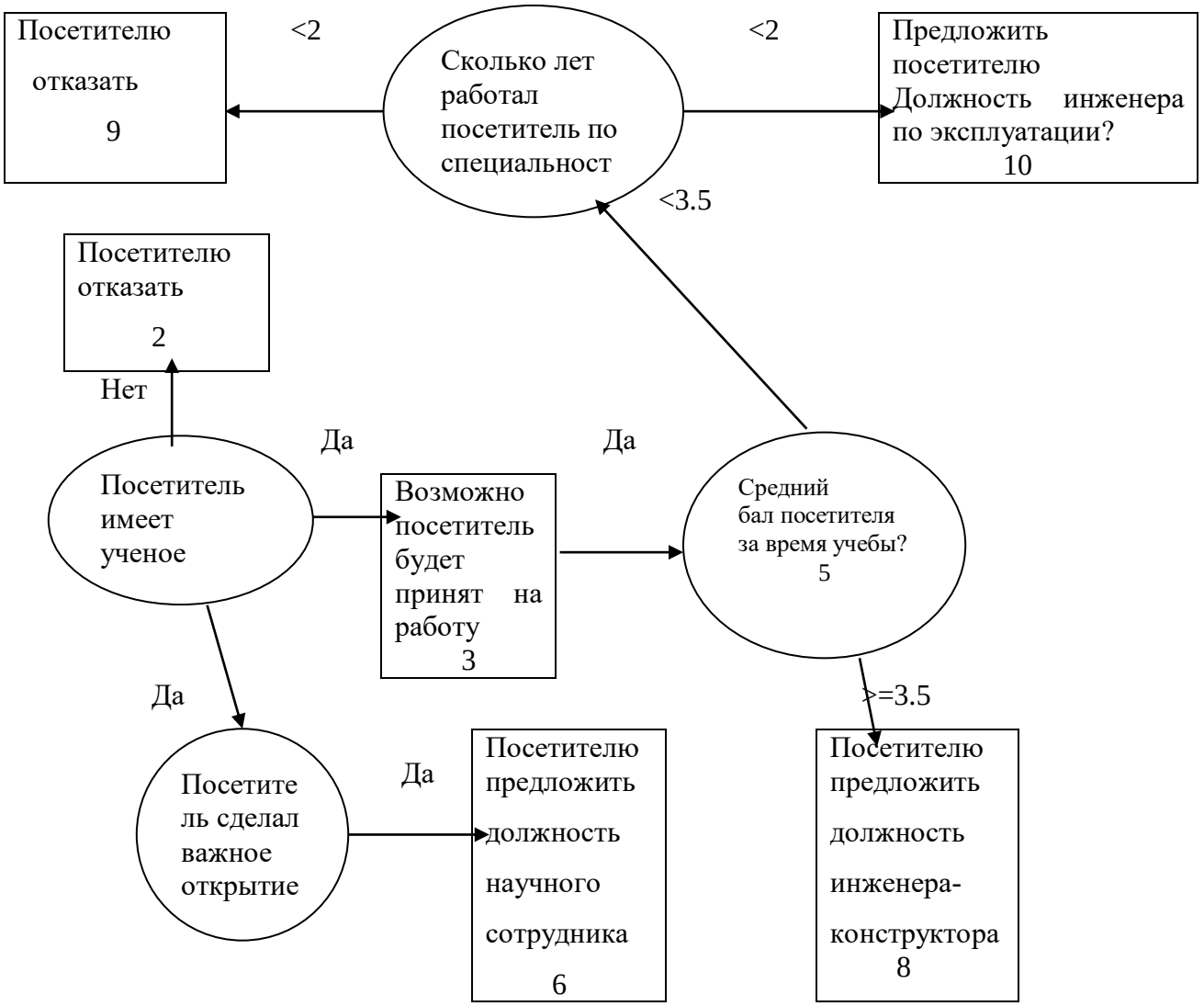


Рис.22. Дерево рішень для вибору посади.

На рис.22 показане дерево рішень для прикладу з прийомом на роботу. Видно, що діаграма складається з кружків і прямокутників, які називаються вершинами. Кожній вершині привласнюється номер. На вершини можна посилатися по цих номерах. Лінії, що сполучають вершини, називаються дугами або гілками. Кухлі, що містять питання, називаються вершинами рішень. Прямокутники містять цілі діаграми і означають логічні висновки. Лінії показують напрям діаграми. Багато вершин мають декілька гілок, що пов'язують їх з іншими вершинами. Вибір гілки, що виходить з вершини, визначається перевіркою умови, що міститься у вершині. Наприклад, вершина 5 містить питання, на який є дві можливі відповіді, і тому у неї два шляхи залежно від середнього балу відвідувача за час навчання, тобто можливий вибір однієї з двох гілок.

У програмі під середній бал спочатку відводиться змінна, а потім їй привласнюється значення. Можна сказати, що вершини містять змінні, а шляхи - це умови, відповідно до яких змінним привласнюються значення. Після того, як для проблемної області сформульовані правила, ці умови стають умовними частинами (ЯКЩО) правила. **Прямокутники** містять приватні або загальні висновки. Загальна мета системи, в якій реалізовані зворотні міркування, - одержати остаточну відповідь. Локальною метою може бути що міститься в прямокутнику з вершиною 3 відповідь на питання, чи буде відвідувачу запропонована посада. Проте ця вершина має і витікаючі гілки, і, отже, через неї може проходити шлях до наступного логічного висновку. У останньому випадку, оскільки витікаюча гілка не містить умови і вона тільки одна, говорять, що вершина містить локальний висновок для іншої мети. **Локальний висновок** - це також становляча умовній частині правила.

Зауваження по розробці системи

Система, що реалізовує зворотний ланцюжок міркувань, повинна виконувати наступні кроки:

1. Визначити змінну логічного висновку.
2. У списку логічних висновків шукати перше входження цієї змінної. Якщо змінна знайдена, в стек логічних висновків помістити номер відповідного правила і встановити номер умови рівним 1. Якщо змінна не знайдена, повідомити користувача, що відповідь знайти неможливо.

Привласнити значення всім змінним умови з даного правила.

Якщо в списку змінних вказано, що якої-небудь змінної умови не привласнено значення і її немає серед змінних логічного висновку (її немає в списку логічних висновків), запитати її значення у користувача.

Якщо яка-небудь змінна умови входить в змінні логічного висновку, помістити в стек номер правила, в логічний висновок якого вона входить, і повернутися до кроку 3.

Якщо з правила не можна визначити значення змінної, видалити відповідний йому елемент із стека і в списку логічних висновків продовжити пошук правила з цією змінною логічного висновку.

Якщо таке правило знайдене, перейти до кроку 3.

Якщо змінна не знайдена ні в одному з правил, що залишилися, в логічному висновку, правило для попереднього висновку не вірно. Якщо попереднього висновку не існує, повідомити користувача, що відповідь одержати неможливо. Якщо попередній висновок існує, повернутися до кроку 6.

Визначити значення змінної з правила, розташованого на початку стека; правило із стека видалити. Якщо є ще змінні логічного висновку, збільшити значення номера умови і для перевірки змінних, що залишилися, повернутися до кроку 3. Якщо більше немає змінних логічного вивода, повідомити користувачу остаточний висновок.

Розглянутий вище алгоритм, що реалізовує зворотний ланцюжок міркувань, можна набудувати на будь-яку базу знань, яка складається з правил.

Стан і тенденції розвитку штучного інтелекту

Програмні засоби, що базуються на технології і методах штучного інтелекту, набули значне поширення в світі. Їх важливість, і, в першу чергу, експертних систем і нейронних мереж, полягає у тому, що дані технології істотно розширюють круг практично значущих задач, які можна вирішувати на комп'ютерах, і їх рішення приносить значний економічний ефект. В той же час, технологія експертних систем є найважливішим засобом в рішенні глобальних проблем традиційного програмування: тривалість і, отже, висока вартість розробки додатків; висока вартість супроводу складних систем; повторна використовувана програм і т.п. Крім того, об'єднання технологій експертних систем і нейронних мереж з технологією традиційного програмування додає нові якості до комерційних продуктів за рахунок забезпечення динамічної модифікації додатків користувачем, а не програмістом, більшої "прозорості"

додатку (наприклад, знання зберігаються на обмеженій природній мові, що не вимагає коментарів до них, спрощує навчання і супровід), кращих графічних засобів, призначеного для користувача інтерфейсу і взаємодії.

У недалекій перспективі експертні системи виконуватимуть провідну роль у всіх фазах проектування, розробки, виробництва, розподілу, продажу, підтримки і надання послуг. Їх технологія, набувши комерційне поширення, забезпечить революційний прорив в інтеграції додатків з готових інтелектуально-взаємодіючих модулів.

Комерційний ринок продуктів штучного інтелекту в світі має декілька основних напрямів:

1) експертні системи; тепер їх часто позначають ще одним терміном - "системи, засновані на знаннях";

2) нейронні мережі і "розмиті" (fuzzy) логіки;

3) природно-мовні системи.

Ринок цей можна розділити і інакше: на системи штучного інтелекту (додатки) і інструментальні засоби, призначені для автоматизації всіх етапів існування додатку.

Успіхи систем штучного інтелекту

Використовування експертних систем і нейронних мереж приносить значний економічний ефект. Так, наприклад:

- American Express скоротила свої втрати на 27 млн. доларів в рік завдяки експертній системі, що визначає доцільність видачі або відмови в кредиті тій або іншій фірмі;
- DEC щорічно економить 70 млн. доларів в рік завдяки системі XCON/XSEL, яка за замовленням покупця складає конфігурацію обчислювальної системи VAX. Її використовування скоротило число помилок від 30% до 1%;
- Sira скоротила витрати на будівництво трубопроводу в Австралії на 40 млн. доларів за рахунок управляючої трубопроводом експертної системи.

Причини, що привели системи штучного інтелекту до **комерційного успіху**, наступні:

1. Спеціалізація. Перехід від розробки інструментальних засобів загального призначення до проблемно/предметно спеціалізованих засобів, що забезпечує скорочення термінів розробки додатків, збільшує ефективність використовування інструментарію, спрощує і прискорює роботу експерта,

дозволяє повторно використовувати інформаційне і програмне забезпечення (об'єкти, класи, правила, процедури).

2. Використовування мов традиційного програмування і робочих станцій. Перехід від систем, заснованих на мовах штучного інтелекту (Lisp, Prolog і т.п.), до мов традиційного програмування (З, С++ і т.п.) спростив "інтегрованість" і понизив вимоги додатків до швидкодії і місткості пам'яті. Використовування робочих станцій замість ПК різко збільшило круг можливих додатків методів штучного інтелекту.

3. Інтегрованість. Розроблені інструментальні засоби штучного інтелекту, що легко інтегруються з іншими інформаційними технологіями і засобами (з CASE, СУБД, контроллерами, концентраторами даних і т.п.).

4. Відвертість і переносимість. Розробки ведуться з дотриманням стандартів, що забезпечують дані характеристики .

5. Архітектура клієнт/сервер. Розробка розподіленої інформаційної системи в даній архітектурі дозволяє понизити вартість устаткування, використовуваного в додатку, децентралізувати додатки, підвищити надійність і загальну продуктивність, оскільки скорочується об'єм інформації, що пересилається між ЕОМ, і кожен модуль додатку виконується на адекватному устаткуванні.

Отже, у області штучного інтелекту найбільшого комерційного успіху досягли експертні системи і засоби для їх розробки. У свою чергу, в цьому напрямі найбільшого успіху досягли проблемно/предметно спеціалізовані засоби.

Експертні системи реального часу - основний напрям штучного інтелекту. **Серед спеціалізованих систем, заснованих на знаннях, найбільш значущі експертні системи реального часу, або динамічні експертні системи. На їх частку припадає 70 відсотків цього ринку.**

Значущість інструментальних засобів реального часу визначається не стільки їх бурхливим комерційним успіхом (хоча і це гідно ретельного аналізу), але, в першу чергу, тим, що тільки за допомогою подібних засобів створюються стратегічно значущі додатки в таких областях, як управління безперервними виробничими процесами в хімії, фармакології, виробництві цементу, продуктів харчування і т.п., аерокосмічні дослідження, транспортування і переробка нафти і газу, управління атомними і тепловими електростанціями, фінансові операції, зв'язок і багато інших.

Класи задач, вирішуваних експертними системами реального часу, такі: моніторинг в реальному масштабі часу, системи управління верхнього рівня,

системи виявлення несправностей, діагноста, складання розкладів, планування, оптимізація, системи-порадники оператора, системи проектування.

Статичні експертні системи не здатні вирішувати подібні задачі, оскільки вони не виконують вимоги, що пред'являються до систем, що працюють у реальному часі:

1. Представляти змінюються в часі дані, що поступають від зовнішніх джерел, забезпечувати зберігання і аналіз даних, що змінюються.
2. Виконувати тимчасові міркування про декілька різних асинхронних процесів одночасно (тобто планувати відповідно до пріоритетів обробку процесів, що поступили в систему).
3. Забезпечувати механізм міркування при обмежених ресурсах (час, пам'ять). Реалізація цього механізму висуває вимоги до високої швидкості роботи системи, здатності одночасно вирішувати декілька задач (тобто операційні системи UNIX, VMS, Windows NT, але не MS-DOS).
4. Забезпечувати "передбаченість" поведінки системи, тобто гарантію того, що кожна задача буде запущена і завершена в строгій відповідності з тимчасовими обмеженнями. Наприклад, дана вимога не допускає використовування в експертній системі реального часу механізму "збірки сміття", властивого мові Lisp.
5. Моделювати "навколишній світ", що розглядається в даному додатку, забезпечувати створення різних його станів.
6. Протоколювати свої дії і дії персоналу, забезпечувати відновлення після збою.
7. Забезпечувати наповнення бази знань для додатків реального ступеня складності з мінімальними витратами часу і праці (необхідне використовування об'єктно-орієнтованої технології, загальних правил, модуля і т.п.).
8. Забезпечувати настройку системи на вирішувані задачі (проблемна/наочна орієнтованість).
9. Забезпечувати створення і підтримку призначених для користувача інтерфейсів для різних категорій користувачів.
10. Забезпечувати рівень захисту інформації (по категоріях користувачів) і запобігати несанкціонованому доступу.

Підкреслимо, що окрім цих десяти вимог засобу створення експертних систем реального часу повинні задовольняти і перерахованим вище загальним вимогам.

Лекція 13. Нейронні мережі. Застосування нейронних мереж.

Як визначення нейронних мереж може бути прийнято: ШНС (штучна нейронна мережа) – паралельно розподілена структура обробки інформації, що складається з окремих елементів (нейронів), сполучених між собою зв'язками.

Нейрокомп’ютери (комп'ютери, побудовані на основі нейронних мереж) володіє цілий ряд властивостей, привабливих з погляду їх практичного використовування:

1. Надвисока швидкодія за рахунок використовування масового паралелізму обробки інформації.
2. Толерантність до помилок: працездатність зберігається при пошкодженні значного числа нейронів.
3. Здібність до навчання; програмування обчислювальної системи замінюється навчанням.
4. Здібність до розпізнавання образів в умовах сильних перешкод і спотворень.

Проте, перші 2 властивості мають місце тільки при апаратній реалізації нейронних мереж. Апаратний реалізовані нейронні мережі забезпечують рішення складних задач за часи порядку часів спрацьовування ланцюжків електронних і/або оптичних елементів. Рішення слабо залежить від несправності окремого нейрона. Це робить їх привабливими для використовування у складі бортових обчислювальних систем.

Пакет Neural Networks Toolbox системи Matlab містить засоби для проектування, моделювання, навчання і використовування безлічі відомих парадигм сучасного апарату штучних нейронних мереж: від базових моделей персептрона до найсучасніших асоціативних і самоорганізовуються мереж.

В даний час ІНС застосовуються для вирішення багатьох задач, що не формалізуються або важко формалізуються:

- розпізнавання і синтезу мови;
- розпізнавання аерокосмічних зображень;
- прогнозування котирування цінних паперів і курсу валют;
- попередження шахрайства з кредитними картками;
- оцінки вартості нерухомості;
- оцінки фінансового стану підприємств і ризику незворотності кредитів;
- обробки сигналів радіолокацій;
- контролю руху на швидкісних автомагістралях і залізницях;
- діагности в медицині;

- здобичі знань з великих об'ємів даних в бізнесі, фінансах і наукових дослідженнях.

Нейронні мережі можна використовувати за наступних умов:

- **Якщо задачу може вирішувати людина.**
- Якщо при рішенні задачі можна виділити безліч вхідних чинників (сигналів, ознак, даних і т.п.) і безліч вихідних чинників.
- Якщо зміни вхідних чинників приводить до зміни вихідних.

В той же час застосування нейронних мереж при рішенні деяких задач може виявитися ефективніше за використовування розуму людини. Це пояснюється тим, що людський розум орієнтований на рішення задач в тривимірному просторі. Багатовимірні задачі для нього характеризуються значно більшою трудомісткістю. Штучним нейронним мережам не властиве таке обмеження. Їм все одно вирішувати тривимірну або 10-мірну задачу.

Місце нейромережевої технології серед інших методів обробки даних показане на рис.23.

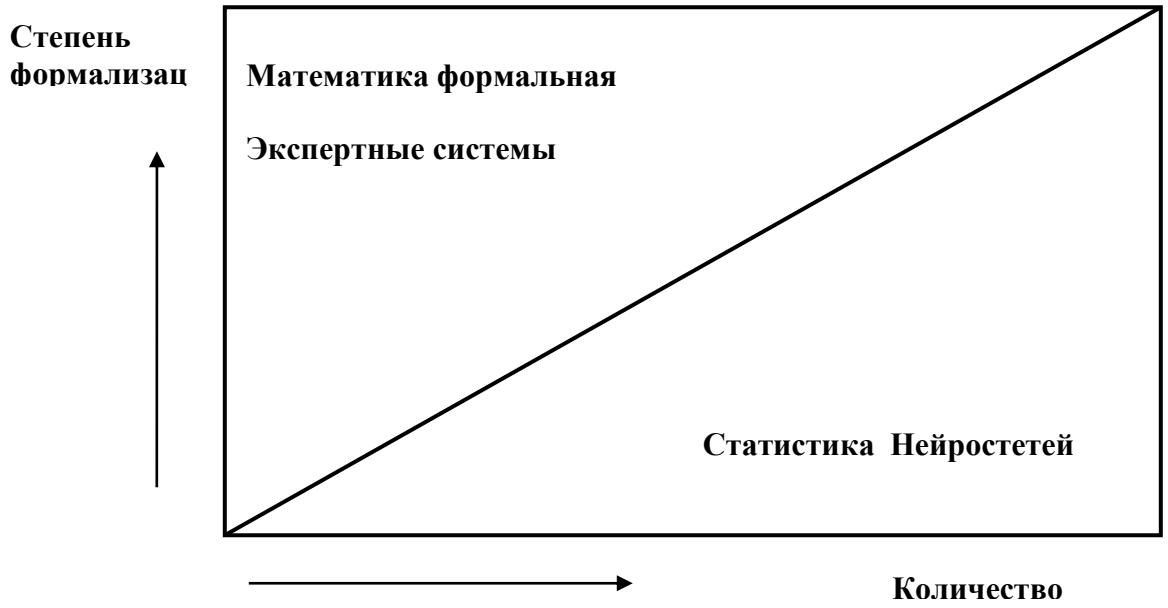


Рис.23. Область використовування нейронних мереж

Подібно біологічній нейронній системі ШНС є обчислювальною системою з величезним числом паралельно функціонуючих простих процесорів з безліччю зв'язків. Моделі ШНС в деякій мірі відтворюють "організаційні" принципи, властиві мозку людини. Моделювання біологічної нейронної системи з використанням ШНС може також сприяти кращому розумінню

біологічних функцій. Такі технології виробництва, як VLSI (надвисокий рівень інтеграції) і оптичні апаратні засоби, роблять можливим подібне моделювання.

Глибоке вивчення ШНС вимагає знання нейрофізіології, науки про пізнання, психології, фізики (статистичної механіки), теорії управління, теорії обчислень, проблем штучного інтелекту, статистики/математики, розпізнавання образів, комп'ютерного зору, паралельних обчислень і апаратних засобів (цифрових/аналогових/VLSI/оптичних).

З другого боку, ШНС також стимулюють ці дисципліни, забезпечуючи їх новими ШНС трументами і уявленнями. Цей симбіоз життєво необхідний для досліджень по нейронних мережах. Представимо деякі проблеми, вирішувані в контексті ШНС і представляючі інтерес для учених і інженерів.

Класифікація образів. Задача полягає у вказівці приналежності вхідного образу (наприклад, мовного сигналу або рукописного символу), представленого вектором ознак, одному або декільком заздалегідь певним класам. До відомих додатків відносяться розпізнавання букв, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація кліток крові.

Кластеризація/категоризація. При рішенні задачі кластеризації, яка відома також як класифікація образів "без вчителя", відсутня повчальна вибірка з мітками класів. Алгоритм кластеризації заснований на подібності образів і розміщує близькі образи в один кластер. Відомі випадки застосування кластеризації для витягання знань, стиснення даних і дослідження властивостей даних.

Апроксимація функцій. Припустимо, що є повчальна вибірка ((x1,y1), (x2,y2)..., (xn,yn)) (пари даних вхід-вихід), яка генерується невідомою функцією f(x), спотвореною шумом. Задача апроксимації полягає в знаходженні оцінки невідомої функції f(x). Апроксимація функцій необхідна при рішенні численних інженерних і наукових задач моделювання.

Прогноз/прогноз. Хай задані n дискретних відліків {y(t1), y(t2)..., y(tn)} в послідовні моменти часу t1, t2,..., tn . Задача полягає в прогнозі значення y(tn+1) в деякий майбутній момент часу tn+1. Прогноз/прогноз мають значний вплив на ухвалення рішень в бізнесі, науці і техніці. Прогноз цін на фондовій біржі і прогноз погоди є типовими додатками техніки прогнозу/прогнозу.

Оптимізація. Численні проблеми в математиці, статистиці, техніці, науці, медицині і економіці можуть розглядатися як проблеми оптимізації. Задачею алгоритму оптимізації є знаходження такого рішення, яке задовольняє системі

обмежень і максимізує або мінімізує цільову функцію. Задача комівояжера, що відноситься до класу NP-повних, є класичним прикладом задачі оптимізації.

Управління. Розглянемо динамічну систему, задану сукупністю {u(t), y(t)}, де u(t) є вхідною управляючою дією, а y(t) - виходом системи у момент часу t. У системах управління з еталонною моделлю метою управління є розрахунок такої вхідної дії u(t), при якому система слідує по бажаній траєкторії, диктованій еталонною моделлю. Прикладом є оптимальне управління двигуном.

При застосуванні нейронних мереж необхідно вирішити наступні задачі:

1. Постановка задачі, придатної для вирішення за допомогою нейронної мережі.
2. Вибір моделі ШНС.
3. Підготовка початкових даних для навчання ШНС.
4. Навчання ШНС.
5. Власне рішення задачі за допомогою навченої ШНС

Крім того, іноді потрібен ще один етап – інтерпретація рішення, одержаного нейронною мережею.

Найбільш трудомісткими процесами при використовуванні нейронних мереж є підготовка початкових даних для навчання і навчання нейронної мережі.

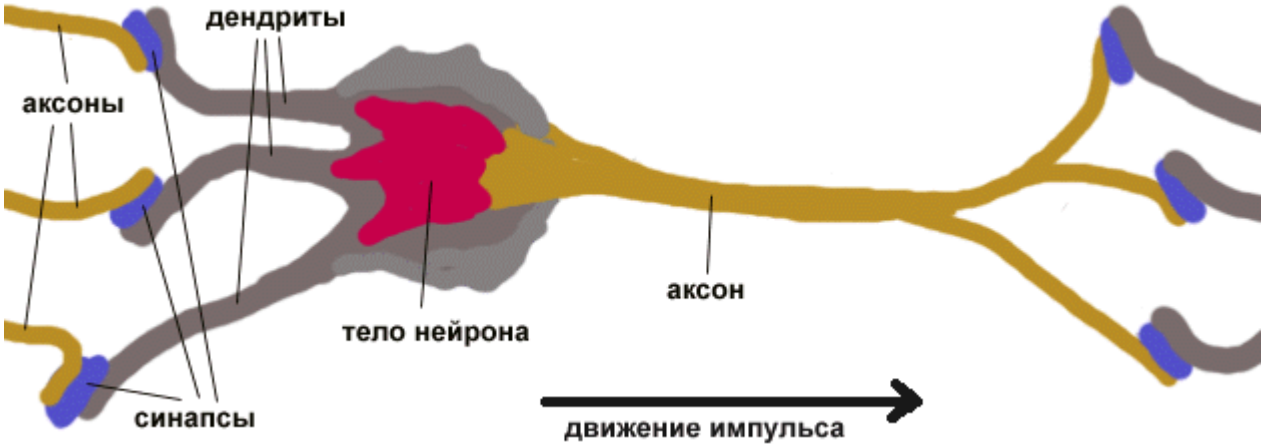


Рис. 24. Будова біологічного нейрона

Нервова система і мозок людини складаються з нейронів, сполучених між собою нервовими волокнами (рис.24). Нервові волокна здатні передавати електричні імпульси між нейронами. Всі процеси передачі роздратувань від нашої шкіри, вух і очей до мозку, процеси мислення і управління діями - все це

реалізовано в живому організмі як передача електричних імпульсів між нейронами. Розглянемо будову біологічного нейрона. Кожен нейрон має відростки нервових волокон двох типів - дендріти, по яких приймаються імпульси, і єдиний аксон, по якому нейрон може передавати імпульс. Аксон контактує з дендрітамі інших нейронів через спеціальні утворення - *синапси*, які впливають на силу імпульсу.

Можна вважати, що при проходженні синапсу сила імпульсу міняється в певне число раз, яке ми називатимемо вагою синапсу. Імпульси, що поступили до нейрона одночасне по декількох дендрітам, підсумовуються. Якщо сумарний імпульс перевищує деякий поріг, нейрон збуджується, формує власний імпульс і передає його далі по аксону. Передача інформації між нейронами здійснюється **ЕЛЕКТРИЧНИМИ ІМПУЛЬСАМИ**. Під дією цих сигналів виникає процес хімічної дифузії іонів натрію, калія, хлора і деяких інших елементів, змінюючий електричний **ПОТЕНЦІАЛ МЕМБРАНИ**. Коли цей потенціал (сумарний імпульс) досягає деякої величини, званої **ПОРОГОМ НЕЙРОНА**, нейрон збуджується і виробляє імпульс, який йде по аксону. При цьому потенціал мембрани різко падає і нейрон "розряджається". Після деякої паузи нейрон може знову сформувати імпульс. Важливо відзначити, що вага синапсів може змінюватися з часом, а значить, міняється і поведінка відповідного нейрона. Вихідний сигнал проходить по гілках аксона і досягає СИНАПСІВ**,** які сполучають аксони з дендрітнимі деревами інших нейронів. Всі нейрони виробляють сигнали тільки ОДНОГО **ЗНАКУ.** Якщо імпульси, що поступають на СИНАПС**,** приводять до підвищення мембранного потенціалу, то такий СИНАПС називається ЗБУДЛИВИМ. Інакше СИНАПС називається ГАЛЬМУЮЧИМ. Величина вхідного сигналу, що генерується синапсом, може бути різною навіть при однаковій величині сигналу, що проходить через синапс. Ці відмінності визначаються ефективністю або ВАГОЮ **СИНАПСУ.**
Важливо відзначити, що вага синапсів може змінюватися з часом, а значить, міняється і поведінка відповідного нейрона.

Під нейронними мережами маються на увазі обчислювальні структури, які моделюють прості біологічні процеси, звичайно асоційовані з процесами людського мозку. Що адаптуються і навчані, вони є розпаралелюючими системами, здібними до навчання шляхом аналізу позитивних і негативних дій. Елементарним перетворювачем в даних мережах є *штучний нейрон* або просто нейрон, названий так по аналогії з біологічним прототипом.

До теперішнього часу запропонована і вивчена велика кількість моделей нейроновидних елементів і нейронних мереж.

Структура штучного нейрона

Структура штучного нейрона показана на мал. 25. До складу нейрона входять помножувачі (синапси), суматор і нелінійний перетворювач.

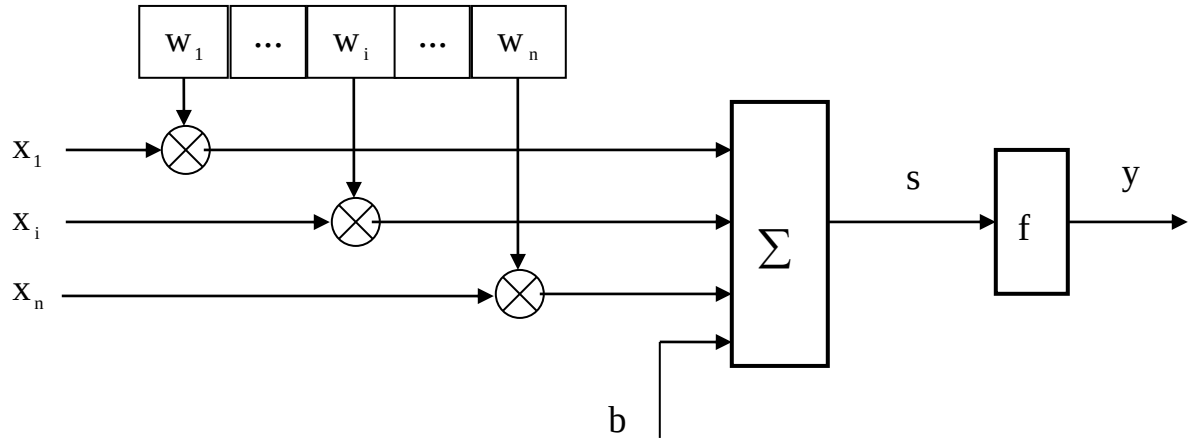


Рис.25. Структура штучного нейрона

Синапси здійснюють зв'язок між нейронами і умножають вхідний сигнал на число, що характеризує силу зв'язку, - вага синапсу. Суматор виконує складання сигналів, що поступають по синаптиічним зв'язках від інших нейронів, і зовнішніх вхідних сигналів. Нелінійний перетворювач реалізує нелінійну функцію одного аргументу – виходу суматора. Ця функція називається *функцією активації* або *передавальною функцією* нейрона. Нейрон в цілому реалізує скалярну функцію векторного аргументу. Неважко побудувати математичну модель описаного процесу. Математична модель нейрона описується співвідношеннями:

$$s = \sum_{i=1}^{n} w_i x_i + b_i, \qquad y = f(s),$$

де вага синапсу, $(i = 1,...n)$; $s -$ результат підсумовування; $x_i -$ компонент вхідного вектора (вхідний сигнал); $y -$ вихідний сигнал нейрона; $n -$ число входів нейрона;

f – нелінійне перетворення (функція активації або передавальна функція); b – значення зсуву (у нейронну мережу іноді вводячи зсув, який діє як ваговий коефіцієнт від осередку з активацією, рівною 1; зсув збільшує вхідну дію в мережу на одиницю; замість зсуву у ряді випадків застосовується фіксований поріг для функції активації).

Лекція 14. Нейронні мережі. Функції активації

У загальному випадку вхідний сигнал, вагові коефіцієнти і значення зсуву можуть приймати речовинні значення. Вихід у визначається видом функції активації і може бути як дійсним, так і цілим.

Синаптіческіє вагу з позитивними вагами називають збудливими, з негативними – гальмуючими.

Нейрон перетворить одержаний сумарний імпульс відповідно до деякої передавальної функції $f(s)$. Таким чином, нейрон повністю описується своїми вагами w_i і передавальною функцією }plain $f(s)$. Одержавши набір чисел (вектор) x_i як входи, нейрон видає деяке число у на виході.

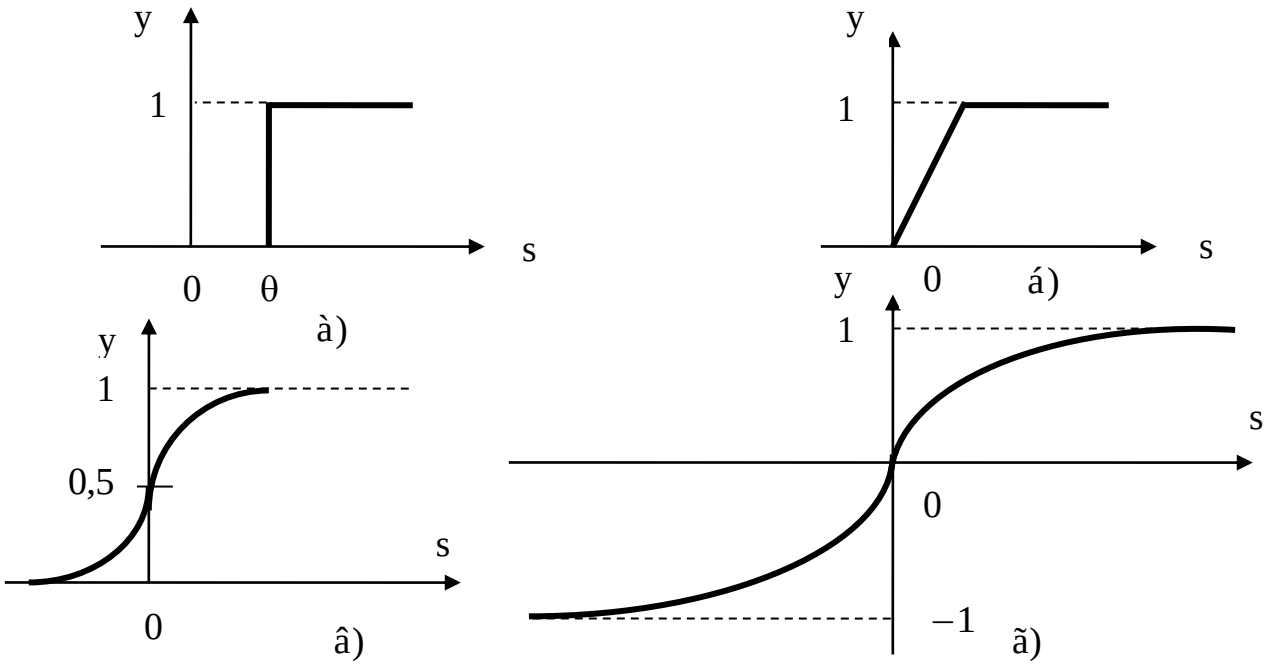


Рис. 26. Примеры активаційних функцій: а – порогова; б – напівлінійна з насиченням; у – сигмоїд (логістична); г сигмоїд (гіперболічний тангенс)

На вхідний сигнал s нелінійний перетворювач відповідає вихідним сигналом, який є виходом нейрона }plain y. Приклади активаційних функцій представлені в табл.8 і на рис.26.

Табл.8

Название	**Формула**	**Область значений**
Порогоговая	$f(s)=\begin{cases}0, & s<\theta\\ 1, & s\ge\theta\end{cases}$	(0,1)
Знаковая	$f(s)=\begin{cases}1, & s>0\\ -1, & s\le 0\end{cases}$	(-1,1)
Сигмоидальная (логистическая)	$f(s)=\frac{1}{1+e^{-s}}$	(0,1)
Полулинейная	$f(s)=\begin{cases}s, & s>0\\ 0, & s\le 0\end{cases}$	$(0,\infty)$
Линейная	$f(s)=s$	$(-\infty,+\infty)$
Радиальная базисная (гауссова)	$f(s)=\exp(-s^2)$	(0,1)
Полулинейная с насыщением	$f(s)=\begin{cases}0, & s\le 0\\ s, & 0<s<1\\ 1, & s\ge 1\end{cases}$	(0,1)
Линейная с насыщением	$f(s)=\begin{cases}-1, & s\le -1\\ s, & -1<s<1\\ 1, & s\ge 1\end{cases}$	(-1,1)
Гиперболический тангенс (сигмоидальная)	$f(s)=\frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	(-1,1)
Треугольная	$f(s)=\begin{cases}1-\lvert s\rvert, & \lvert s\rvert\le 1\\ 0, & \lvert s\rvert>1\end{cases}$	(0,1)

Однією з найпоширеніших є нелінійна функція з насиченням, так звана логістична функція або сигмоїд:

$$f(s)=\frac{1}{1+e^{-s}}$$

Одна з цінних властивостей сигмоїдальної функції – простий вираз для її похідної:

$$f'(s)=f(s)\left[1-f(s)\right],$$

яке використовується в деяких алгоритмах навчання.

Класифікація нейронних мереж. Як працює нейромережа

Штучна нейронна мережа (*ШНМ*) - це набір нейронів, сполучених між собою. Як правило, передавальні функції всіх нейронів в мережі фіксовані, а вага є параметрами мережі і може змінюватися. Деякі входи нейронів помічені як зовнішні входи мережі, а деякі виходи - як зовнішні виходи мережі. Подаючи будь-які числа на входи мережі, ми одержуємо якийсь набір чисел на виходах мережі. Таким чином, робота нейромережі полягає в перетворенні вхідного вектора X у вихідний вектор, причому це перетворення задається вагами мережі.

Нейронні мережі розрізняють по структурі мережі (зв'язків між нейронами), особливостям моделі нейрона, особливостям навчання мережі. По структурі нейронні мережі можна розділити (рис.27) на неповнозв'язні (або шаруваті) і повнозв'язні, з випадковими і регулярними зв'язками, з симетричними і несиметричними зв'язками.

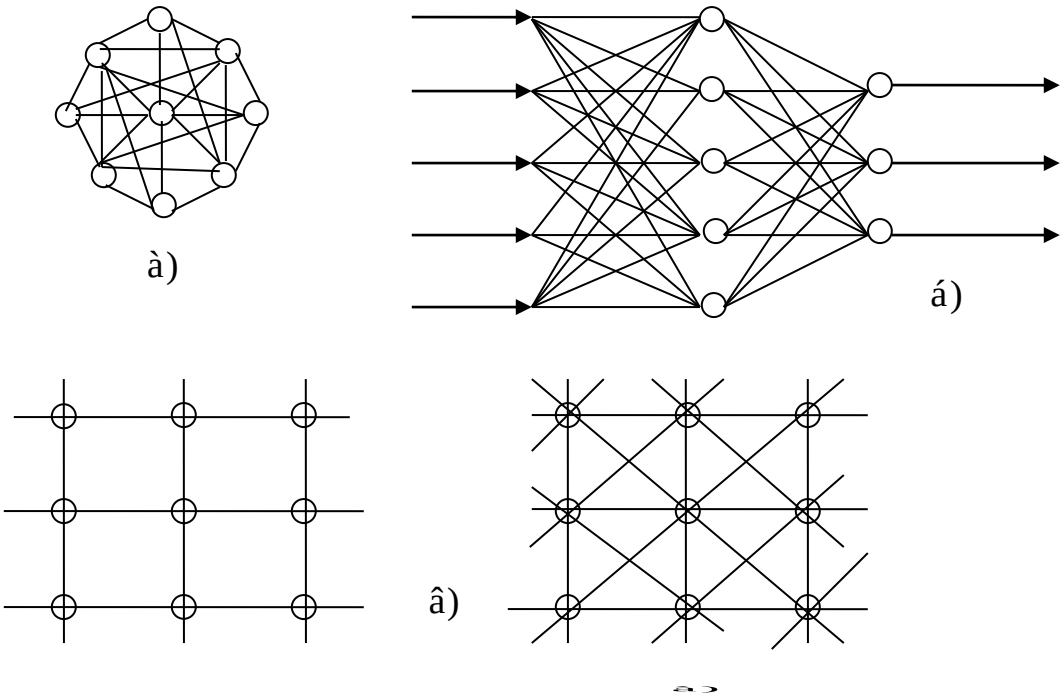


Рис.27. Архітектура нейронних мереж: а – повнозв'язна мережа; б – багатошарова мережа з послідовними зв'язками; у – слабозв'язні мережі

Неполнозв'язні нейронні мережі (описувані неповнозв'язним орієнтованим графом і звичайно звані персептронами), підрозділяються на одношарові (прості персептрони) і багатошарові, з прямими, перехресними і зворотними зв'язками. Класичним варіантом шаруватих мереж є мережі прямого розповсюдження (рис.25).

У нейронних мережах з прямими зв'язками нейрони *j-ого* шару по входах можуть з'єднуватися тільки з нейронами і-их шарів, де $j > i$, тобто з нейронами ніжележащих шарів. У нейронних мережах з перехресними зв'язками допускаються зв'язки усередині одного шару, тобто вище приведена нерівність замінюється на $j >= i$ В нейронних мережах із зворотними зв'язками використовуються і зв'язки *j-ого* шаруючи по входах з *і-им* при $j < i$. Крім того, по вигляду зв'язків розрізняють персептрони з регулярними і випадковими зв'язками.

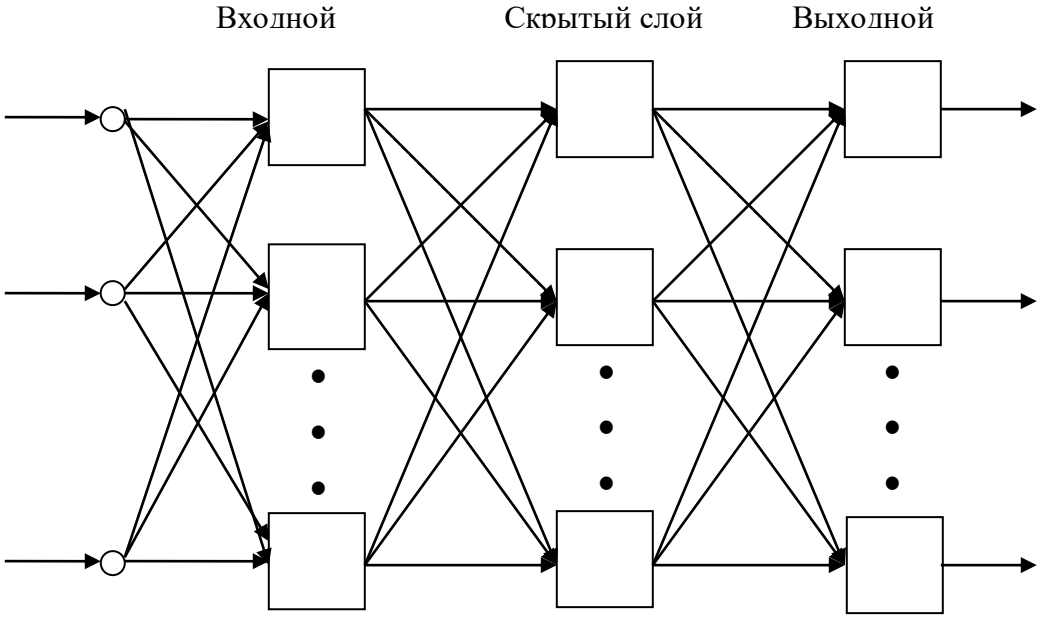


Рис.27. Багатошарова (двошарова) мережа прямого розповсюдження

За способом подачі інформації на входи нейронної мережі розрізняють:

- подачу сигналів на синапси вхідних нейронів;
- подачу сигналів на виходи вхідних нейронів;
- подачу сигналів у вигляді вагів синапсів вхідних нейронів;
- адитивну подачу на синапси вхідних нейронів.

За способом знімання інформації з виходів нейронної мережі розрізняють:

- з'їм з виходів вихідних нейронів;

- з'їм з синапсів вихідних нейронів;
- з'їм у вигляді значень вагів синапсів вихідних нейронів;
- адитивне знімання з синапсів вихідних нейронів.

По організації навчання розділяють навчання нейронних мереж з вчителем (supervised neural networks) і без вчителя (nonsupervised). При навчанні з вчителем передбачається, що є зовнішнє середовище, яке надає повчальні приклади (значення входів і відповідні їм значення виходів) на етапі навчання або оцінює правильність функціонування нейронної мережі і відповідно до своїх критеріїв міняє стан нейронної мережі або заохочує (карає) нейронну мережу, запускаючи тим самим механізм зміни її стану. Під станом нейронної мережі, який може змінюватися, звичайно розуміється:

- вага синапсів нейронів (карта вагів - map) (конекційністський підхід);
- вага синапсів і пороги нейронів (звичайно в цьому випадку поріг є легше змінним параметром, ніж вага синапсів);
- встановлення нових зв'язків між нейронами (властивість біологічних нейронів встановлювати нові зв'язки і ліквідовувати старі називається пластичністю).

За способом навчання розділяють навчання по входах і по виходах. При навчанні по входах повчальний приклад є тільки вектором вхідних сигналів, а при навчанні по виходах в нього входить і вектор вихідних сигналів, відповідний вхідному вектору.

За способом пред'явлення прикладів розрізняють пред'явлення одиночних прикладів і "сторінки" прикладів. У першому випадку зміна стану нейронної мережі (навчання) відбувається після пред'явлення кожного прикладу. У другому - після пред'явлення "сторінки" (множини) прикладів на основі аналізу відразу їх всіх.

Практично будь-яку задачу можна звести до задачі, вирішуваної нейромережею.

У табл.9 показано, яким чином слід сформулювати в термінах нейромережі задачу розпізнавання рукописних букв.

Пояснимо, навіщо вимагається вибирати вихід з максимальним рівнем сигналу. Річ у тому, що рівень вихідного сигналу, як правило, може приймати будь-які значення з якогось відрізка. Проте, в даній задачі нас цікавить не аналогова відповідь, а всього лише номер категорії (номер букви в алфавіті). Тому використовується наступний підхід - кожній категорії зіставляється свій вихід, а відповіддю мережі вважається та категорія, на чиєму виході рівень

сигналу максимальний. У певному значенні рівень сигналу на виході "А" - це достовірність того, що на вхід була подана рукописна буква "А". Задачі, в яких потрібно віднести вхідні дані до однієї з відомих категорій, називаються *задачами класифікації*. Висловлений підхід - стандартний спосіб класифікації за допомогою нейронних мереж.

Потім можна переходити до наступного питання «як будувати мережу». Це розв'язується в два етапи:

1) вибір типу (архітектура мережі);
2) підбір вагів (навчання) мережі.

На **першому етапі** слід вибрати наступне:

- які нейрони ми хочемо використовувати (число входів, передавальні функції);
- яким чином слід з'єднати їх між собою;
- що узяти як входи і виходи мережі.

Табл.9. Задача розпізнавання рукописних букв в термінах нейромережі

Задача распознавания рукописных букв	
Дано: растровое черно-белое изображение буквы размером 30x30 пикселов	**Надо**: определить, какая это буква (в алфавите 33 буквы)
Формулировка для нейросети	
Дано: входной вектор из 900 двоичных символов (900=30x30)	**Надо**: построить нейросеть с 900 входами и 33 выходами, которые помечены буквами. Если на входе сети - изображение буквы "А", то максимальное значение выходного сигнала достигается на выходе "А". Аналогично сеть работает для всех 33 букв.

Лекція 15. Нейронні мережі (продовження)

Ця задача на перший погляд здається неозорою, але необов'язково придумувати нейромережа "з нуля" - існує декілька десятків різної нейромережевих архітектури, причому ефективність багатьох з них доведена математично. Найпопулярніша і вивчена архітектура - це багатошаровий персептрон, нейромережа із загальною регресією, мережі Кохонена і інші. На **другому етапі** нам слід "навчити" вибрану мережу, тобто підібрати такі значення її вагів, щоб мережа працювала потрібним чином. Ненавчена мережа

подібна дитині - її можна навчити чому завгодно. У використовуваних на практиці нейромереж кількість вагів може складати декілька десятків тисяч, тому навчання - дійсно складний процес. Для багатьох архітектури розроблені спеціальні алгоритми навчання, які дозволяють набудувати вагу мережі певним чином. Найпопулярніший з цих алгоритмів - метод зворотного розповсюдження помилки (Error Back Propagation), використовуваний, наприклад, для навчання персептрона.

Залежно від функцій, виконуваних нейронами в мережі, можна виділити три їх типи:

1) *вхідні* нейрони – на них подається вхідний вектор, що кодує вхідну дію або образ зовнішнього середовища; обчислювальних процедур в цих нейронах не здійснюється, інформація передається з входу на вихід шляхом зміни його активації;

2) *вихідні* нейрони – вихідні значення яких представляють вихід мережі;

3) *проміжні* нейрони – складають основу ШНС, перетворення в них виконуються по вищенаведених формулах.

Вибір структури ШРС здійснюється відповідно до особливостей і складності задачі. Для вирішення деяких окремих типів задач існують оптимальні конфігурації. Якщо ж задача не може бути зведена ні до одного з відомих типів, розробнику доводиться вирішувати проблему синтезу нової конфігурації, причому необхідно керуватися наступними принципами:

1) можливості мережі зростають із збільшенням числа осередків мережі, густина зв'язку між ними і числом виділених шарів;
2) введення зворотних зв'язків разом із збільшенням можливостей мережі піднімає питання про динамічну стійкість мережі;
3) складність алгоритмів функціонування мережі (наприклад, введення декількох типів синапсів) також сприяє посиленню можливостей мережі.

В більшості випадків оптимальний варіант виходить на основі інтуїтивного підбору, хоча в літературі приведені докази того, що для будь-якого алгоритму існує нейронна мережа, яка може його реалізувати.

Багато задач: розпізнавання образів (зорових, мовних), управління, прогнозування, ідентифікації складних систем, зводяться до наступної постановки: необхідно побудувати відображення $X \to Y$ таке, щоб у відповідь на кожен можливий вхідний сигнал X формувався правильний вихідний сигнал У. Відображення задається кінцевим набором пар («вхід», «відомий вихід»). Число таких пар (повчальних прикладів) істотно менше за загальне число

можливих поєднань значень вхідних і вихідних сигналів. Сукупність всіх повчальних прикладів носить назву повчальної вибірки.

У задачах розпізнавання образів X – деяке представлення образу (зображення, вектор чисел), У – номер класу, до якого належить вхідний образ.

У **задачах управління** X – набір контрольованих параметрів керованого об'єкту, У – код, що визначає управляючу дію, відповідну поточним значенням контрольованих параметрів.

У задачах **прогнозування** як вхідні сигнали використовуються тимчасові ряди, що представляють значення контрольованих змінних на деякому інтервалі часу. Вихідний сигнал – безліч змінних, яка є підмножиною змінних вхідного сигналу.

Взагалі кажучи, велика частина прикладних задач може бути зведена до реалізації деякого складного багатовимірного функціонального перетворення $X \to Y$.

В результаті побудови такого перетворення (відображення) необхідно добиватися того, щоб забезпечувалося формування правильних вихідних сигналів у відповідності:

1) зі всіма прикладами повчальної вибірки;

2) зі всіма можливими вхідними сигналами, які не увійшли до повчальної вибірки.

Відзначимо, що теоретичною основою для побудови НС є наступне твердження: для будь-якої безлічі пар вхідних-вихідних векторів довільної розмірності існує двошарова однорідна нейронна мережа з послідовними зв'язками, з сигмоїдальними передавальними функціями і з кінцевим числом нейронів, яка для кожного вхідного вектора X^k формує відповідний йому вихідний вектор Y^k.

Таким чином, для представлення багатовимірних функцій багатьох змінних може бути використана однорідна нейронна мережа, що має всього один прихований шар, з сигмоїдальними передавальними функціями нейронів.

Для оцінки числа нейронів в прихованих шарах ОНС можна скористатися формулою для оцінки необхідного числа синаптичних вагів L_w (у багато-шаровій мережі з сигмоїдальними передавальними функціями):

$$\frac{mN}{1+\log_2 N} \le L_w \le m\left(\frac{N}{m}+1\right)(n+m+1)+m,$$

де n – розмірність вхідного сигналу, m – розмірність вихідного сигналу,

N – число елементів повчальної вибірки.

Число нейронів в двошаровій мережі складе: $L = \frac{L_w}{n+m}$ або

$$\frac{N}{10} - n - m \le L \le \frac{N}{2} - n - m$$

Теорема про повноту. Будь-яка безперервна функція на замкнутій обмеженій множині може бути рівномірно наближена функціями, обчислюваними нейронними мережами, якщо функція активації нейрона двічі безперервне диференціюєма і нелінійна.

Таким чином, НС є універсальними апроксимуючими системами. Задавшися певною структурою ШНС, якій-небудь задачі, що відповідає, розробник мережі повинен знайти оптимальні значення всіх вагових коефіцієнтів. Цей етап називається навчанням ШНС.

Навчання нейронної мережі

Таким чином, нейромережі складені з простих елементів, діючих паралель. Можна навчити нейромережа, регулюючи значення вагів між елементами. Звичайно мережа регулюється або навчається так, щоб приватний вхід вів до цільового виходу. Регульована мережа заснована на порівнянні виходу і мети, поки мережевий вихід не відповідатиме меті. Звичайно є багато повчальних пар вхід/мета. Кількість повчальних пар вибирається залежно від постановки задачі. Змінні, виступаючі в ролі вхідних і вихідних сигналів НС, можуть бути експериментальними даними, одержуваними вимірюванням вихідних параметрів об'єкту при завданні певних вхідних.

Навчити нейронну мережу - значить, повідомити її, чого ми від неї добиваємося. Цей процес дуже схожий на навчання дитини алфавіту. Показавши дитині зображення букви "А", ми питаємо його: "Яка це буква?" Якщо відповідь невірна, ми повідомляємо дитині ту відповідь, яка ми хотіли б від нього одержати: "Це буква А". Дитина запам'ятовує цей приклад разом з вірною відповіддю, тобто в його пам'яті відбуваються деякі зміни в потрібному напрямі. Ми повторюватимемо процес пред'явлення букв знову і знову доти, коли всі 33 букви твердо запам'ятають. **Такий процес називають "навчання з вчителем"** (рис.28).

При навчанні мережі ми діємо абсолютно аналогічно. У нас є деяка база даних, що містить приклади (набір рукописних зображень букв). Пред'являючи зображення букви "А" на вхід мережі, ми одержуємо від неї деяку відповідь, не обов'язково вірний. Нам відома і вірна (бажаний) відповідь - в даному випадку нам хотілося б, щоб на виході з міткою "А" рівень сигналу був максимальний. Звичайно як бажаний вихід в задачі класифікації беруть набір (1, 0, 0, ...), де 1 стоїть на виході з міткою "А", а 0 - на всій решті виходів. Обчислюючи різницю між бажаною відповіддю і реальною відповіддю мережі, ми одержуємо 33 числа - *вектор помилки*. Алгоритм зворотного розповсюдження помилки - це набір формул, який дозволяє по вектору помилки обчислити необхідні поправки для вагів мережі. Одну і ту ж букву (а також різні зображення однієї і тієї ж букви) ми можемо пред'являти мережі багато раз. У цьому значенні навчання швидше нагадує повторення вправ в спорті - тренування.

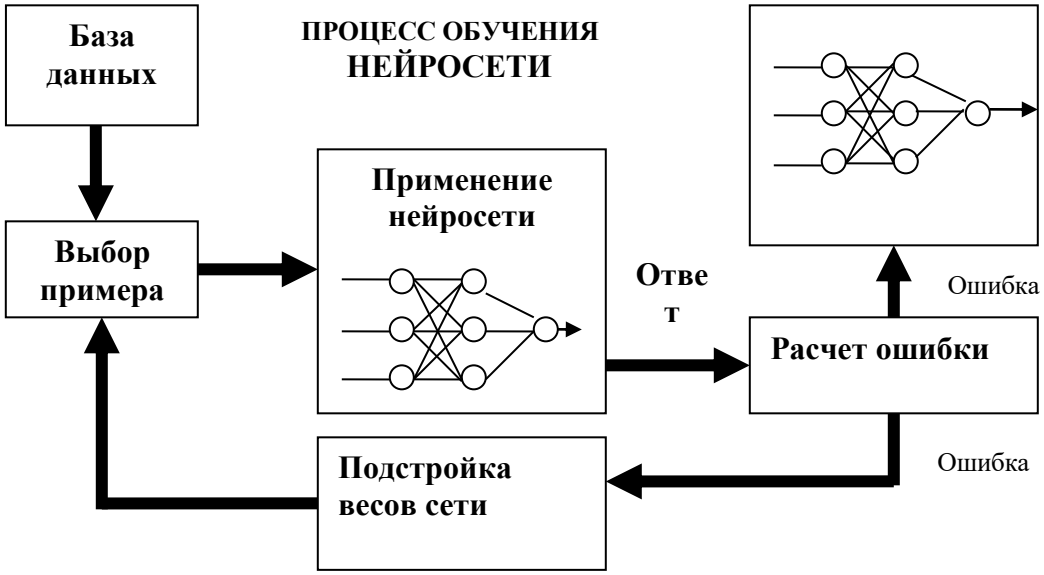


Рис.28. Ілюстрація процесу навчання НС

Виявляється, що після багатократного пред'явлення прикладів ваги мережі стабілізуються, причому мережа дає правильні відповіді на всі (або майже все) приклади з бази даних. У такому разі говорять, що "мережа вивчила всі приклади", "мережа навчена", або "мережа натренована". У програмних реалізаціях можна бачити, що в процесі навчання величина помилки (сума квадратів помилок по всіх виходах) поступово зменшується. Коли величина помилки досягає нуля або прийнятного малого рівня, тренування зупиняють, а одержану мережу вважають натренованою і готовою до застосування на нових даних. Важливо відзначити, що вся інформація, яку мережа має про задачу,

міститься в наборі прикладів. Тому якість навчання мережі напряму залежить від кількості прикладів в повчальній вибірці, а також від того, наскільки повно ці приклади описують дану задачу. Так, наприклад, безглуздо використовувати мережу для прогнозу фінансової кризи, якщо в повчальній вибірці криз не представлено. Вважається, що для повноцінного тренування потрібно хоча б декілька десятків (а краще за сотні) прикладів. Повторюваний ще раз, що навчання мережі - складний і наукомісткий процес. Алгоритми навчання мають різні параметри і настройки, для управління якими потрібне розуміння їх впливу.

Лекція 16. Алгоритми навчання нейронних мереж

Розглянемо ідею одного з найпоширеніших алгоритмів навчання – **алгоритму зворотного розповсюдження помилки**. Це ітеративний градієнтний алгоритм навчання, який використовується з метою мінімізації середньоквадратичного відхилення поточного виходу від бажаного виходу в багатошарових нейронних мережах. Цей алгоритм використовується для навчання багатошарових НС з послідовними зв'язками вигляду рис.25. Нейрони в таких мережах діляться на групи із загальним вхідним сигналом - шари. На кожен нейрон першого шару подаються всі елементи зовнішнього вхідного сигналу. Всі виходи нейронів m-го шару подаються на кожен нейрон шару m+1. Нейрони виконують зважене підсумовування елементів вхідних сигналів. До суми елементів вхідних сигналів, що множать на відповідну синаптичні вагу, додається зсув нейрона. Над результатом підсумовування виконується нелінійне перетворення - функція активації (передавальна функція). Значення функції активації є вихід нейрона.

У багатошарових мережах оптимальні вихідні значення нейронів всіх шарів, окрім останнього, як правило, невідомі, і трьох- або більш шаровий персептрон вже неможливо навчити, керуючись тільки величинами помилок на виходах НС. Найприйнятнішим варіантом навчання в таких умовах опинився градієнтний метод пошуку мінімуму функції помилки з розглядом сигналів помилки від виходів НС до її входів, тобто в напрямі, зворотному прямому розповсюдженню сигналів в звичному режимі роботи. Цей алгоритм навчання НС одержав назву процедури зворотного розповсюдження.

У даному алгоритмі функції помилки є сумою квадратів розузгодження (помилки) бажаного виходу мережі і реального. При обчисленні елементів вектора градієнта використаний своєрідний вид похідних функцій активації сигмоїдного типу. Алгоритм діє циклічно і його цикли прийнято називати епохами. На кожному етапі на вхід мережі по черзі подаються всі повчальні спостереження, вихідні значення мережі порівнюються з цільовими значеннями і обчислюється помилка. Значення помилки, а також градієнта поверхні помилок використовується для коректування вагів, після чого всі дії повторюються. Початкова конфігурація мережі вибирається випадковим чином, і процес навчання припиняється або коли пройдена певна кількість епох, або коли помилка досягне деякого певного рівня крихти, або коли помилка перестане зменшуватися.

Навчання мережі методом ОРО включає три етапи:

- Пряме розповсюдження вхідного повчального образу
- Обчислення помилки і її зворотне розповсюдження
- Регулювання вагів

Цей метод заснований на обчисленні вектора градієнта поверхні помилок, який указує напрям найкоротшого спуску по поверхні з даної крапки. Послідовність кроків приводить після ряду ітерацій до мінімуму поверхні помилок. Основна трудність тут представляє вибір довжини кроку. На практиці величина кроку приймається пропорційній крутизні схилу з постійною, званою швидкістю навчання. Розглянемо алгоритм для **багатошарового прямонаправленого персептрона** (МПП) (рис.29).

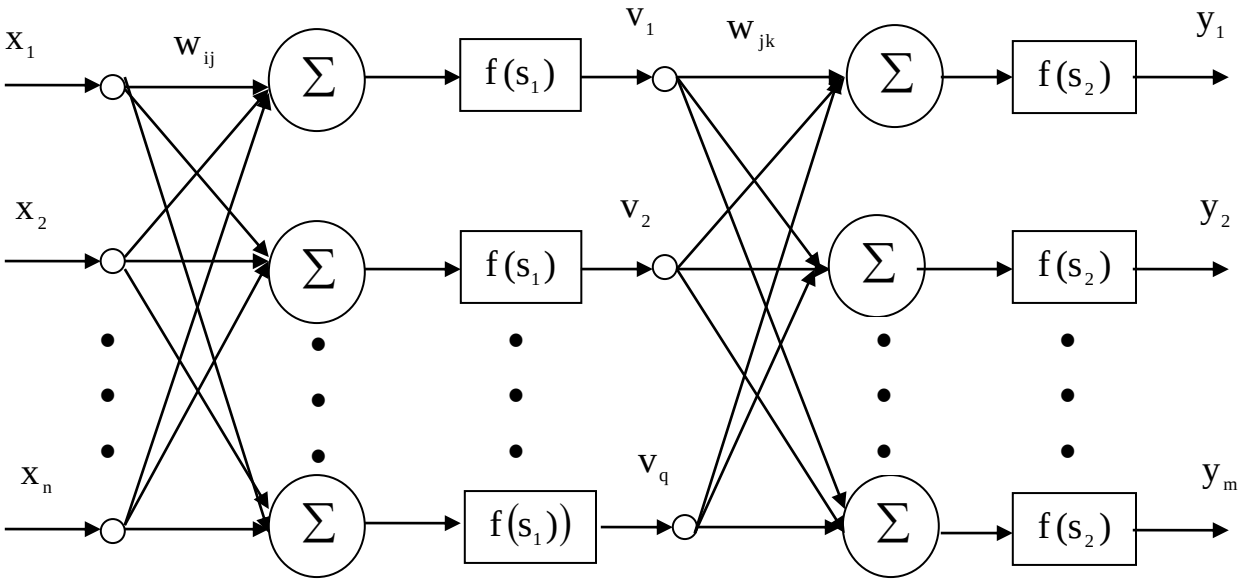


Рис. 29. Схема тришарового персептрона

Позначимо через $x_i (i=1,2,...,n)$ вхідні нейрони; $v_j (j=1,2,...,q)$ – нейрони прихованого шару; $y_k (k=1,2,...,m)$ – вихідні нейрони; w_{ij} – вага від осередків вхідного шару до нейронів прихованого шару; w_{jk} – вага від нейронів прихованого шару до вихідних нейронів. Індексом p $(p=1,2,...,P)$ позначимо різні образи, що пред'являються на вхід. Вхідні сигнали можуть бути **бінарними, біполярними** або **безперервними**.

Поведінка мережі визначається на основі ряду пар «вхід-вихід». Кожен повчальний приклад складається з n вхідних сигналів x_i і m необхідних вихідних сигналів t_k. Навчання МПП для конкретної задачі еквівалентне знаходженню таких значень всіх синаптичних вагів, при яких для відповідного входу формується необхідний вихід або навчання багатошарової мережі полягає в регулюванні всіх вагів таким чином, що помилка між необхідними вихідними t_k і дійсними вихідними y_k сигналами, усереднена по всіх повчальних прикладах, була мінімальна. При пред'явленні образу p на вхід мережі прихований осередок j приймає сигнал:

$$s_j^p = \sum_i w_{ij} x_i^p$$

і на своєму виході за допомогою функції активації виробляє такий сигнал:

$$v_j^p = f(s_j^p) = f\left(\sum_i w_{ij} x_i^p \right).$$

Вихідний осередок з номером k підсумовує сигнали від нейронів прихованого шару, утворюючи сигнал, рівний:

$$s_k^p = \sum_j w_{jk} v_j^p = \sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p \right),$$

і після дії функції активації формує вихідний сигнал (порогові значення відкинуті, оскільки вони завжди можуть бути введені в мережу):

$$y_k^p = f\left(s_k^p\right) = f\left(\sum_j w_{jk} v_j^p \right) = f\left[\sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p \right) \right].$$

якості функції помилок приймемо функцію вигляду:

$$E[w] = 0{,}5 \sum_p \sum_k \left(t_k^p - y_k^p\right)^2,$$

де необхідне значення виходу.

Підставляючи в останній вираз значення для y_k^p, одержимо:

$$E[w] = 0{,}5\sum_p \sum_k \left\{ t_k^p - f\left[\sum_j w_{jk} f\left(\sum_i w_{ij} x_i^p \right) \right] \right\}^2 .$$

Функція є функцією, що безперервно диференціюється, від кожної ваги, що входить в нього вирази, тому можна скористатися алгоритмом градієнтного спуску для знаходження вагів.

Для вагів між прихованим і вихідними шарами правило градієнтного спуску дає:

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}} = \eta \sum_p \left(t_k^p - y_k^p \right) \cdot f'\left(s_k^p \right) \cdot v_j^p = \eta \sum_p \delta_k^p v_j^p ,$$

де, коефіцієнт швидкості навчання ($0 < \eta < 10$). Для вагів між вхідним і прихованим шарами потрібно продиференціювати вираз для $E[w]$, скориставшися ланцюговим правилом:

$$\Delta w_{ij} = \eta \frac{\partial E}{\partial w_{ij}} = -\eta \sum_p \frac{\partial E}{\partial v_j^p} \cdot \frac{\partial v_j^p}{\partial w_{ij}} = \eta \sum_p \sum_k \left(t_k^p - y_k^p \right) f'\left(s_k^p \right) w_{jk} f'\left(s_j^p \right) x_i^p =$$

$$= \eta \sum_p \sum_k \delta_k^p w_{jk} f'\left(s_j^p \right) x_i^p = \sum_p \delta_j^p x_i^p ,$$

де $\delta_j^p = f'\left(s_j^p \right) \sum_k w_{jk} d_k^p .$

Відзначимо, що рівняння для Δw_{jk} і Δw_{ij} мають однакову форму, але розрізняються значеннями параметра ain δ. В загальному випадку при довільному числі шарів правило зміни вагів в методі ОРО має вигляд:

$$\Delta w_{\alpha\beta} = \eta \sum_p \delta_{out} v_{in} ,$$

де підсумовування виробляється по всіх образах, що пред'являються; вихід і вхід відносяться до двох кінців α і β синаптичного з'єднання; v_{in} – активація від прихованого осередку або реального входи. Значення δ залежать від даного шару; для останнього шару ця величина визначається по виразу, а для інших шарів – формулою подібною $\delta_j^p = f'\left(s_j^p \right) \sum_k w_{jk} d_k^p .$

Вирази, що визначають правила зміни ваги, записані у вигляді сум по образах, що пред'являються, проте звичайно образи поступають на вхід послідовно: образ p пред'являється на вхід, і після закінчення проходу «вперед-назад» по мережі вся вага змінюється перед представленням наступного образу. Це зменшує функцію помилок E на кожному кроці. Якщо образи вибираються

у випадковому порядку, то рух по простору вагів відбувається стохастично, що дозволяє ширше досліджувати поверхню помилок. Альтернативна версія зміни вагів (групове навчання) полягає в мінімізації функції помилки таким чином, що вагові зміни накопичуються по всіх повчальних прикладах і тільки тоді відбувається модифікація вагів.

Цей алгоритм є наступною послідовністю кроків.

1. Ініціювати вагу, прийнявши їх малими випадковими величинами.
2. Якщо умова зупинки не виконується, робити кроки 3-10.
3. Вибирається чергова повчальна пара (X, Y) з повчальної множини; вектор X подається на вхід мережі. Для кожної повчальної пари виконувати кроки 4-9.

Прямий прохід по мережі.

4. Кожен вхідний осередок x_i приймає вхідний сигнал і поширює його до всіх нейронів прихованого шару.
5. Кожен осередок прихованого шару $v_j, j = 1,2,..,q$ підсумовує свої зважені вхідні сигнали $s_j = \sum w_{ij} x_i$ застосовує до одержаної суми функцію активації, формуючи вихідний сигнал $v_j = f(s_j)$, який посилається до всіх осередків вихідного шару.
6. Кожен вихідний осередок $y_k, k = 1,2,...,m$ підсумовує зважені сигнали:

$$s_k = \sum w_{jk} v_j,$$

формуючи після застосування функції активації вихідний сигнал мережі:

$$y_k = f(s_k).$$

Зворотне розповсюдження помилки.

6. Кожен вихідний осередок зіставляє своє значення виходу з необхідною цільовою величиною і обчислює параметр δ_k :

$$\delta_k = (t_k - y_k) f'(s_k),$$

Після чого визначається член коректування для вагів:

$$\Delta w_{jk} = \eta \delta_k v_j,$$

а параметри δ_k посилаються в нейрони прихованого шару.

8. Кожен прихований осередок v_j підсумовує свої $\delta-$ входи від нейронів вихідного шару:

$$s_j = \sum_k \delta_k w_{jk}$$

Результат умножається на похідну від функції активації для визначення δ_j:

$$\delta_j = f'(s_j)\sum_k \delta_k w_{jk},$$

Зміна вагів.

9. Вага між прихованим і вихідними шарами модифікуються таким чином:

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}.$$

Аналогічним чином змінюється вага між вхідним і прихованими шарами:

$$w_{ij}(new) = w_{ij}(old) + \Delta w_{ij}.$$

10. Перевірка умови зупинки: мінімізація помилки між необхідним і реальним вихідними сигналами.

Схема алгоритму ОРО представлена на рис.30.

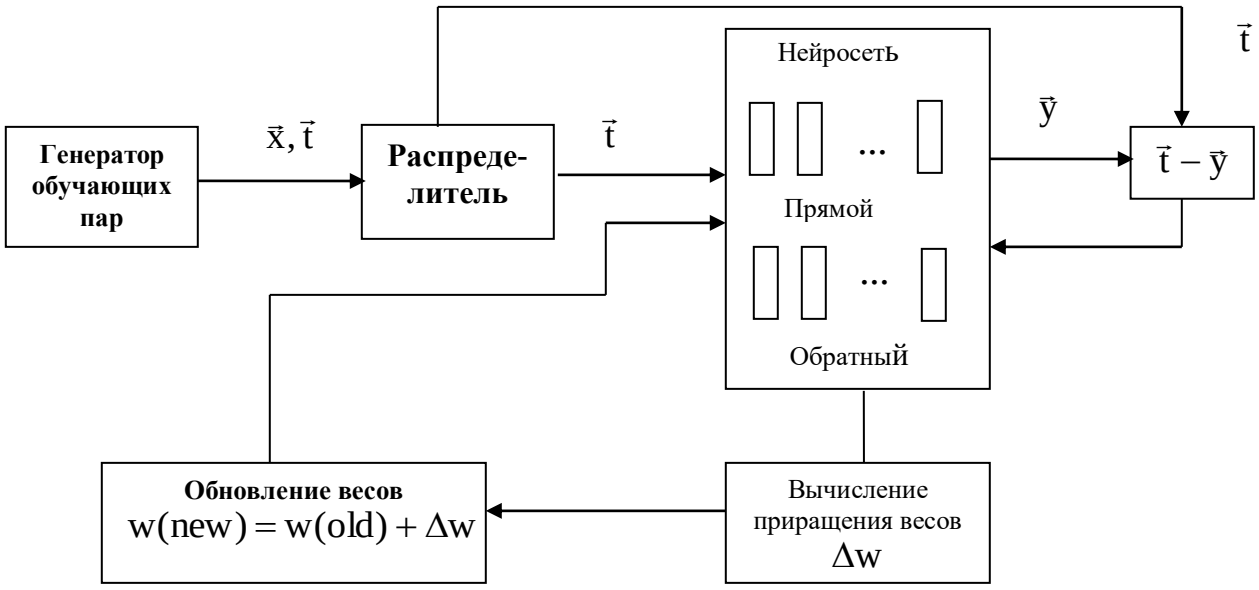


Рис.30. Обчислювальна схема методу ОРО

Лекція 17. Практичні рекомендації при використовуванні алгоритму ОРО

1. Вибір початкових значень. Цей етап робить вплив на досягнення мережею глобального мінімуму функції помилки і на швидкість сходження до мінімуму. З одного боку, значення початкових вагів не повинні бути дуже великими, інакше початкові вхідні сигнали в кожен прихований або вхідний осередок потраплять в діапазон, де похідна сигмоїдної функції активації має дуже малу величину. З другого боку, якщо початкові значення узяти достатньо малими, то мережевий вхід в прихований або вихідний нейрон буде ближчим до нуля, що приведе до дуже повільного навчання. Загальне правило ініціації початкових вагів полягає у виборі їх значень з рівномірно розподілених величин в інтервалі (-0,5…0,5) – іноді цей інтервал може бути декілька менше або більше, але не перевищувати ± 1. Значення вагів можуть позитивними або негативними, оскільки остаточна вага після навчання також може мати будь-який знак.

2. Тривалість навчання мережі. Пропонується використовувати дві серії даних під час навчання: серію повчальних образів і серію контрольних образів. Ці дві серії є роздільними. Регулювання ваги засноване на повчальних образах. Проте під час навчання помилка обчислюється з використанням контрольних образів. Доти поки помилка для контрольних образів зменшується, процес навчання продовжується. При зростанні цієї помилки мережа починає втрачати свою здібність до узагальнення, і у цей момент навчання припиняється.

3. Кількість необхідних навчаних пар. Для співвідношення між числом навчаних образів P, кількістю регульованих вагів і точністю класифікації запропоновано використовувати наступний вираз: $w/P = \varepsilon$. Наприклад, для $\varepsilon = 0{,}1$ багатошарова мережа з 80 регульованими вагами зажадає 800 навчаних образів, щоб бути упевненим в правильній класифікації 90% контрольних образів, що пред'являються.

4. Представлення даних. Для нейронної мережі легше навчитися набору різних станів, ніж відгуку з безперервним значенням.

У багатьох задачах вхідні і вихідні вектори мають складові в одному і тому ж діапазоні величин. Унаслідок того, що один з членів у виразі для коректування вагів є активацією осередку попереднього шару, нейрони, що мають нульову активацію, навчатися не будуть. Навчання може бути поліпшене в тому випадку, якщо вхідний вектор представлений в біполярній формі, а як функція

активації використовується біполярна сигмоїда (біполярна сигмоїда дуже близька до функції гіперболічного тангенса).

5. Введення інерційної поправки. Показано, що пошук мінімуму функції помилок методом градієнтного спуску виявляється достатньо повільним, якщо швидкість навчання η мала, і приводити до значних осциляцій при великій швидкості η.

6. Модифікація функції активації. Діапазон функції активації повинен відповідати діапазону цільових значень конкретної задачі. Бінарна сигмоїдна функція вигляду $f(x) = [1 + \exp(-x)]^{-1}$ з похідною $f'(x) = f(x)[1 - f(x)]$

Може бути змінена для перекриття будь-якого необхідного діапазону з центром при будь-якому значенні x і необхідному нахилі.

Сигмоїда може мати розширений діапазон, щоб відображати значення в інтервалі $[a, b]$ для будь-яких a і b. Для цього потрібно ввести параметри

$$\gamma = b - a; \qquad \rho = -a$$

Тоді сигмоїдна функція:

$$g(x) = \gamma f(x) - \rho$$

має необхідні властивості, тобто діапазон $[a, b]$. Її похідна виражається як:

$$g'(x) = \gamma^{-1}[\rho + g(x)][\gamma - \rho - g(x)].$$

Нахил сигмоїдної функції може бути змінений за допомогою введеного параметра γ.

7. Число нейронів в прихованому шарі. Підхід при виборі прихованих нейронів полягає у тому, що на першому етапі число таких нейронів береться явно більшим, ніж потрібен, а далі, у міру навчання мережі, зайві нейрони забираються. Всі нейрони, які не вносять внесок в рішення або дають інформацію, не потрібну в подальшому шарі, розглядаються як зайві і віддаляються з прихованого шару. На практиці вважається, що мережа досягла збіжності, якщо різниця між необхідним і дійсним виходами не перевищує 0,1. Якщо два приховані нейрони дають приблизно однаковий вихід для всіх повчальних прикладів, то тільки один з них дійсно необхідний, оскільки обидва нейрони переносять однакову інформацію. Після видалення тих осередків, які не дають внеску в рішення, значення ваги зменшення мережі повинні бути змінені шляхом перенавчання мережі для отримання необхідних характеристик.

Класичний метод ОРО відноситься до алгоритмів з лінійною збіжністю. Його відомими **недоліками** є: невисока швидкість збіжності (велике число ітерацій), можливість сходиться не до глобального, а до локальних рішень.

Можливий також параліч мережі – більшість нейронів функціонує при дуже великих значеннях аргументу функції активації, тобто на пологій ділянці (оскільки помилка пропорційна похідною, яка на даних ділянках мала, то процес навчання практично завмирає). Для усунення цих недоліків були запропоновані численні модифікації алгоритму ОРО.

Навчання без вчителя. Головна межа, що робить навчання без вчителя привабливим, - це його «самостійність». Процес навчання, як і у випадку з вчителем, полягає в підстроюванні вагів синапсів. Очевидно, що підстроювання вагів синапсів може проводитися тільки на підставі інформації, доступної в нейроні, тобто інформації про його стан, вже наявних вагових коефіцієнтів і поданому векторі Х. Виходячи їх цього і по аналогії з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба і Кохонена. Загальна ідея даних алгоритмів полягає у тому, що в процесі самонавчання шляхом відповідної корекції вагових коефіцієнтів посилюються зв'язки між збудженими нейронами.

Типи нейронних мереж

Багатошарові нейронні мережі – ці мережі з сигмоїдними передавальними функціями є найзагальнішою, універсальною мережевою архітектурою. Є різні структури багатошарових мереж: з послідовними, перехресними і зворотними зв'язками, з фіксованою і змінною структурою. Із зростанням числа шарів зростають як складність побудови мережі, так і якість її роботи. Багатошарові нейронні мережі є універсальними апроксимаціями.

Мережа Кохонена – кластерний аналіз, розпізнавання образів, класифікація. Шар Кохонена самонавчається.

Нейронні мережі стрічного розповсюдження – розпізнавання і відновлення образів, стиснення даних (з втратою інформації), статистичний аналіз. Мережа містить два шари з послідовними зв'язками: перший шар Кохонена, другий Гроссберга. **Переваги**: 1) мережа проста; 2) мережа швидко навчається; мережа дає можливість будувати функцію і зворотну до неї функцію, що знаходить застосування при рішенні практичних задач. **Недоліки.** Слабке теоретичне опрацьовування. Мережа не дає можливість будувати точні апроксимації.

Нейронні мережі Хопфілда і Хеммінга – дозволяють просто і ефективно дозволити задачу відтворення образів за неповною і спотвореною інформацією.

Мережі з радіальними базисними функціями (RBF) – двошарова мережа без зворотних зв'язків, яка містить прихований шар радіально-симетричних прихованих нейронів (шаблонний шар). Мережі RBF моделюють довільну нелінійну функцію за допомогою всього одного проміжного шару. Параметри лінійної комбінації у вихідному шарі можна повністю оптимізувати за допомогою добре відомих методів лінійної оптимізації, які працюють швидко і не зазнають труднощів з локальними мінімумами, що так заважають при навчанні з використанням алгортма ОРО. Тому мережа RBF навчається на порядок швидше, ніж з використанням алгоритму ОРО. **Недоліками** мереж RBF є: мережі володіють поганими екстрапалюючими властивостями і виходять вельми громіздкими при великій розмірності вектора входів. Модифікацією мережі RBF є радіальна базисна *мережа з нульовою помилкою*.

Іншими модифікаціями є нейронні мережі (PNN) вірогідності і узагальнено-регресійна нейронна мережа (GRNN).

Нейронна мережа вірогідності (PNN). Призначені для вирішення задач вірогідності і, зокрема, задач класифікації. Архітектура мережі PNN базується на архітектурі базисної мережі, але як другий шар використовують так званий конкуруючий шар, який підраховує вірогідність приналежності вхідного вектора до того або іншого класу, і, кінець кінцем, зіставляє вектор з тим класом, вірогідність приналежності до якого вища. Вихідне значення має сенс вірогідності. Мережа швидко навчається. **Недоліком** таких мереж є їх об'єм – мережі вимагають багато пам'яті і можуть поволі працювати.

Узагальнено-регресійна нейронна мережа (GRNN). Дана мережа аналогічна нейронній мережі вірогідності, але вона призначена для вирішення задач регресії. Як і мережа RBF, мережа GRNN не володіє здатністю екстраполювати дані.

Лінійні нейронні мережі. Згідно загальноприйнятому в науці принципу, якщо складніша модель не дає кращих результатів, ніж простіша, то з них слід віддати перевагу другій. В термінах апроксимації відображень найпростішою моделлю буде лінійна, в якій апроксимуюча (підганяльна) функція визначається гіперплощиною. У задачі класифікації гіперплощина розміщується так, щоб вона розділяла собою два класи (лінійна дискримінантна функція); у задачі регресії гіперплощина повинна проходити через задані крапки. Лінійна модель задається рівнянням:

$$\boldsymbol{Y} = \boldsymbol{X} \cdot \boldsymbol{W} + \boldsymbol{B},$$

де $\boldsymbol{W}$ – матриця вагів мережі, вектор зсувів.

На мові нейронних мереж лінійна модель представляється мережею без проміжних шарів, яка у вихідному шарі містить тільки лінійні елементи, тобто елементи з лінійною функцією активації. Вага відповідає елементам матриці, а пороги – компонентам вектора зсуву.

Пакет Neural Networks Toolbox

Пакет Neural Networks Toolbox (нейронні мережі) містить засоби для проектування, моделювання, навчання і використовування безлічі відомих парадигм апарату штучних нейронних мереж, від базових моделей персептрона до сучасних асоціативних і самоорганізовуються мереж. Пакет може бути використаний для вирішення безлічі різноманітних задач, таких як обробка сигналів, нелінійне управління, фінансове моделювання і т.п.

Для кожного типу архітектури і повчального алгоритму ШНМ є функції ініціалізації, навчання, адаптації, створення і моделювання, демонстрації і приклади застосування.

Приклади створення і використовування нейронних мереж

1. Функції створення нейронних мереж. Функції даної групи дозволяють створити НМ заданої структури (ненавчену).

Синтаксис: **Ім'я мережі = функція** (параметри мережі)

Наприклад, запис вигляду

net = newlind (p,t),

Опис: **newlind** – функція проектування лінійної НМ. Дана функція по матрицях вхідних і вихідних векторів методом якнайменших квадратів визначає вагу і зсуви лінійної НМ.

net = newgrnn (p,t,spread) – функція створення узагальнено-регресійної мережі;

net = newrbe (p,t,spread) – функція створення мережі з радіальними базисними елементами з нульовою помилкою на повчальній вибірці;

net = newpnn (p,t,spread) - функція створення вірогідності НМ;

net = newp (pr,s,tf,lf) - функція створення персептрона (**pr** – матриця мінімальних і максимальних значень вхідних елементів, **s** – число нейронів, **tf** – функція активації, за умовчанням порогова, **lf** – функція, що реалізовує алгоритм навчання персептрона.

Функції моделювання НМ

1. Sim - функція, що моделює роботу НМ дозволяє розрахувати виходи навченої НС при заданих векторах входів.

2. Gensim – функція формує S-модель нейронної мережі для її запуску в середовищі Simulink.

Синтаксис: Gensim (net,st), де net – ім'я створеної мережі, st – інтервал дискретизації (якщо НМ не має затримок, асоційованих з її входами або шарами, значення даного аргументу встановлюється рівні -1). Функція генерує нейромережвою блок Simulink для подальшого моделювання НС засобами цього пакету.

Приклад 1. Створити узагальнено-регресійну НМ (типа GRNN) з ім'ям a, реалізовуючу функціональну залежність між входом і виходом вигляду $y = x^2$. Синтаксис: net = newgrnn(p,t,spread), де p – матриця вхідних векторів, t – матриця цільових векторів, spread – відхилення (за умовчанням 1,0). Використовувати наступні експериментальні дані:

x=[-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9 1];

y=[1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1].

Перевірку якості відновлення приведеної залежності здійснити, використовуючи дані контрольної вибірки x1=[-0.9 -0.7 -0.3 0.4 0.8].

Процедура створення і використовування даної НС описується таким чином:

```
>> x=[-1 -0.8 -0.5 -0.2 0 0.1 0.3 0.6 0.9 1]; задание входных значений
>> y=[1 0.64 0.25 0.04 0 0.01 0.09 0.36 0.81 1]; задание целевых значений
>> a=newgrnn(x,y,0.01); создание НС с отклонением 0,01
>> y1=sim(a,[-0.9 -0.7 -0.3 0.4 0.8]) опрос сети
y1 =
   0.8200    0.5049    0.0316    0.0710    0.6390
```

Як видно, точність апроксимації в даному випадку вийшла не дуже високою. Спробуємо використовувати мережу з радіальними базисними елементами типу newrbe:

```
>> a=newrbe(x,y);
>> y1=sim(a,[-0.9 -0.7 -0.3 0.4 0.8])
y1 =
   0.8100    0.4900    0.0900    0.1600    0.6400
```

Неважко бачити, що застосування мережі даного типу приводить до точного відновлення заданої залежності.

Приклад 2. Розглянемо задачу відновлення деякої, взагалі кажучи, невідомої залежності за наявними експериментальними даними з використанням лінійної НМ. Хай експериментальна інформація задана значеннями

x = [1.0 1.5 3.0 -1.2],

y = [0.5 1.1 3.0 -1.0]

Створимо вектори входу і цілей:

```
>> x=[1.0 1.5 3.0 -1.2];
>> y=[0.5 1.1 3.0 -1.0];
```

Створимо лінійну нейронну мережу:

```
>> b=newlind(x,y); Создание НС с именем b.
```

Проведемо опит мережі для значення входу, рівного 3,0 (цьому, згідно експериментальним даним, відповідає цільове значення 3,0):

```
>> y1=sim(b, 3.0) Опрос мережі
y1 =    2.7003
```

Погрішність відновлення за даними повчальної вибірки в даному випадку - 10%.

Лекція 18-19. Використовування Simulink при побудові НМ

Пакет Neural Networks містить ряд блоків, які можуть бути або безпосередньо використані для побудови НС в середовищі **Simulink, або застосовуватися разом з розглянутою функцією gensim.**

Для виклику набору блоків в командному рядку необхідно набрати команду **neural**, після виконання якої з'явиться вікно (рис.31).

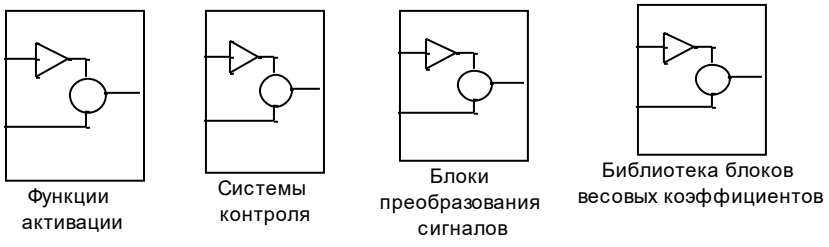


Рис.31. Основні нейромережеві блоки **Simulink**

Кожний з представлених на рис.29 блоків, у свою чергу, є набором деяких блоків.

Блоки функції активації. Подвійне клацання на блоці приводить до появи бібліотеки блоків (рис.32). Кожний з цих блоків даної бібліотеки перетворить вектор, що подається на нього, в відповідний вектор тієї ж розмірності.

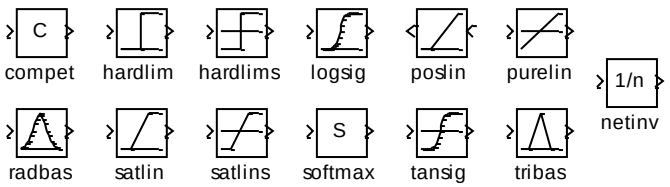


Рис.32. Бібліотека блоків функцій активації

Блоки перетворення входів мережі. Функції даної групи реалізують функції накопичення потенціалу нейрона у вигляді поелементного добутку або сум зважених входів нейрона, а також обчислюють похідні від таких добутків або сум (рис.33).

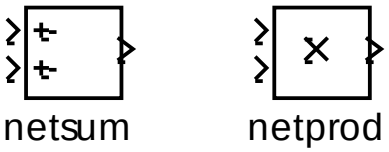


Рис.33. Вікно бібліотеки блоків перетворення сигналів

Блоки вагових коефіцієнтів. Функції цієї групи виконують наступні операції: зважування і обчислення відстаней в мережах з топологією (dotprod – функція функція додання входам P деяких вагів $WZ = W \cdot P$. Повертає матрицю ; normprod – функція обчислення нормованого скалярного твору (кожен елемент додатково ділиться на суму елементів відповідного стовпця-співмножника); dist – функція обчислення евклідова відстані; **negdist – негативна евклідово відстань. Вектори в Simulink необхідно представляти як стовпці.**

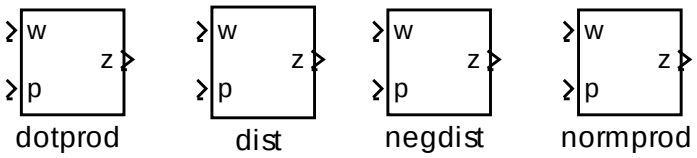


Рис.34. Бібліотека блоків вагових коефіцієнтів

Блоки нейромережевих регуляторів. На рис.35 показаний набір блоків, об'єднаних в бібліотеку системи контролю. Дані блоки реалізують нейромережеві регулятори трьох різних структур – регулятор з прогнозом, регулятор, заснований на використовуванні моделі нелінійної авторегресії з ковзаючим середнім (NARMA 1,2 Controller) і регулятор на основі еталонної моделі, які зручні при побудові і дослідженні моделей систем автоматичного управління, а також блок проглядання результатів.

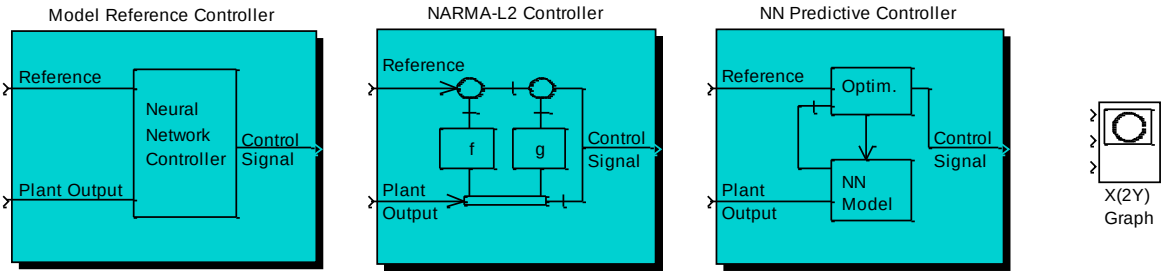


Рис.35. Вікно бібліотеки нейромережевих регуляторів

Формування нейромережевих моделей. Основною функцією для формування нейросетевих моделей в **Simulink є функція gensim, записувана у формі**

Gensim (net,st), де net – ім'я створеної мережі, st – інтервал дискретизації (якщо НС не має затримок, асоційованих з її входами або шарами, значення даного аргументу встановлюється рівні -1). Функція генерує нейроережевий блок Simulink (рис.36) для подальшого моделювання НС засобами цього пакету.

Приклад 3. Створити двошарову НМ прямої передачі сигналу (функція створення багатошарової НМ записується як newff) з вектором мінімальних і максимальних значень входів [0,1], п'ятьма нейронами в першому (прихованому) шарі і одним нейроном – у вихідному шарі, а потім сформувати S-модель такої НМ. Дана задача розв'язується таким чином:

```
>> net=newff([0 1],[5 1]); %  Здійснення нової НМ
>> gensim(net)
```

Результат виконання функції gensim відображений на рис.36.

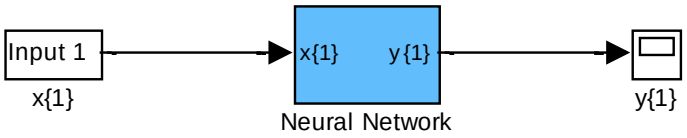
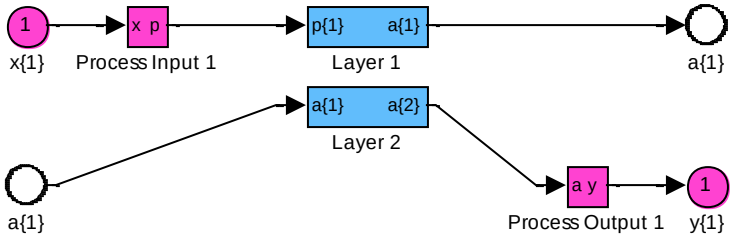
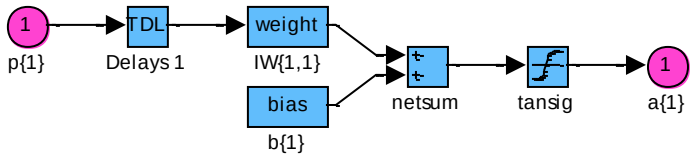


Рис. 36. Результат виконання функції gensim

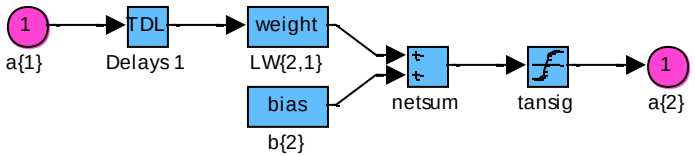
Двічі клацаючи на блоці Neural Network, а потім на блоках Layer1 і Layer2, можна одержати детальну інформацію про структуру мережі (рис.37)



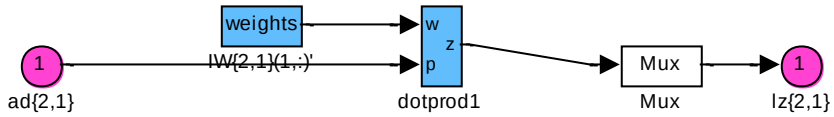
Структура здійснення НС



Структура 1-го шару

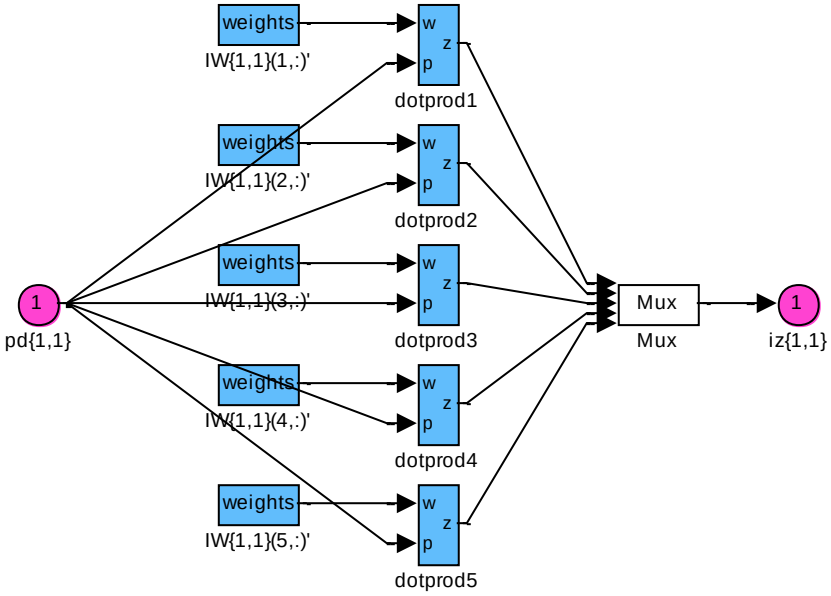


Структура 2-го шару



Структура вихідного шару

Рис.37. Структура здійснення НС з детальною графічною інформацією



Структура прихованого шару

Приклад 4. Нехай вхідною і цільовою вектори мають вигляд

p=[1 2 3 4 5];
t=[1 3 5 7 9];

Створимо линів НМ і протестуємо її за даними повчальної вибірки:

```
>> p=[1 2 3 4 5];
>> t=[1 3 5 7 9];
>> net=newlind(p,t); создание НС
>> y=sim(net,p) опрос сети
y =
   1.0000    3.0000    5.0000    7.0000    9.0000
```

Запустимо **Simulink** командою

```
>> gensim(net,-1)
```

Це приведе до відкриттю блок-діаграми, показаної на рис.34. В даному випадку блок Input1является стандартним боком завдання константи (Constant). Змінимо значення за умовчанням на 4 і натиснемо кнопку ОК, потім кнопку Start в панелі інструментів. Для висновку нового значення необхідно двічі клацнути на правому значку блоку y(1), воно буде рівне 7.

Із створеними мережами можна проводити різні експерименти, можливі в середовищі **Simulink**, наприклад, можливе вбудовування нейромережевого регулятора в систему управління і моделювання останньої.

Система автоматичного управління з нейромережевим регулятором на основі еталонної моделі

Як приклад моделювання в середовищі Simulink систем управління з використанням нейроережевих регуляторів використовуємо приклад, що ілюструє систему управління з еталонною моделлю реактора повного перемішування (рис.38).

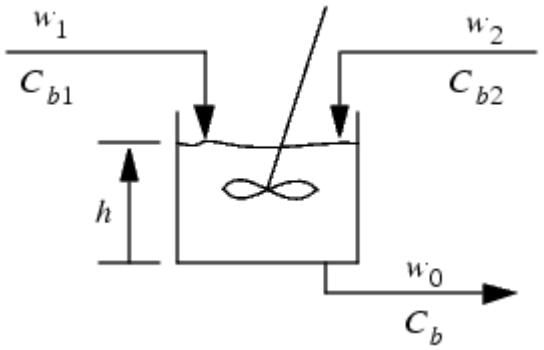


Рис.38. Реактор повного перемішування (w_1, w_2, w_0 – витрати, рівень, концентрації компонентів на вході і виході з реактора, відповідно).

Зміни концентрації в реакторі і на виході з нього, а також рівень рідини в реакторі (динамічна модель системи) представлені наступною системою диференціальних рівнянь (одержані з рівнянь матеріального балансу):

$$\frac{dC_b(t)}{dt} = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1 + k_2 C_b(t))^2}$$

$$\frac{dh(t)}{dt} = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$

де $k_1 = 1$ і $k_2 = 1$ - константи,, $C_{b2} = 0.1$.

Мета автоматичного управління - стабілізувати концентрацію на виході, регулюючи потік $w_1(t)$ (витрата $w_2(t) = 0.1$ є величиною постійною). Рівень резервуару $h(t)$ не є керованим параметром.

Відповідна динамічна модель об'єкту регулювання приведена на рис.39.

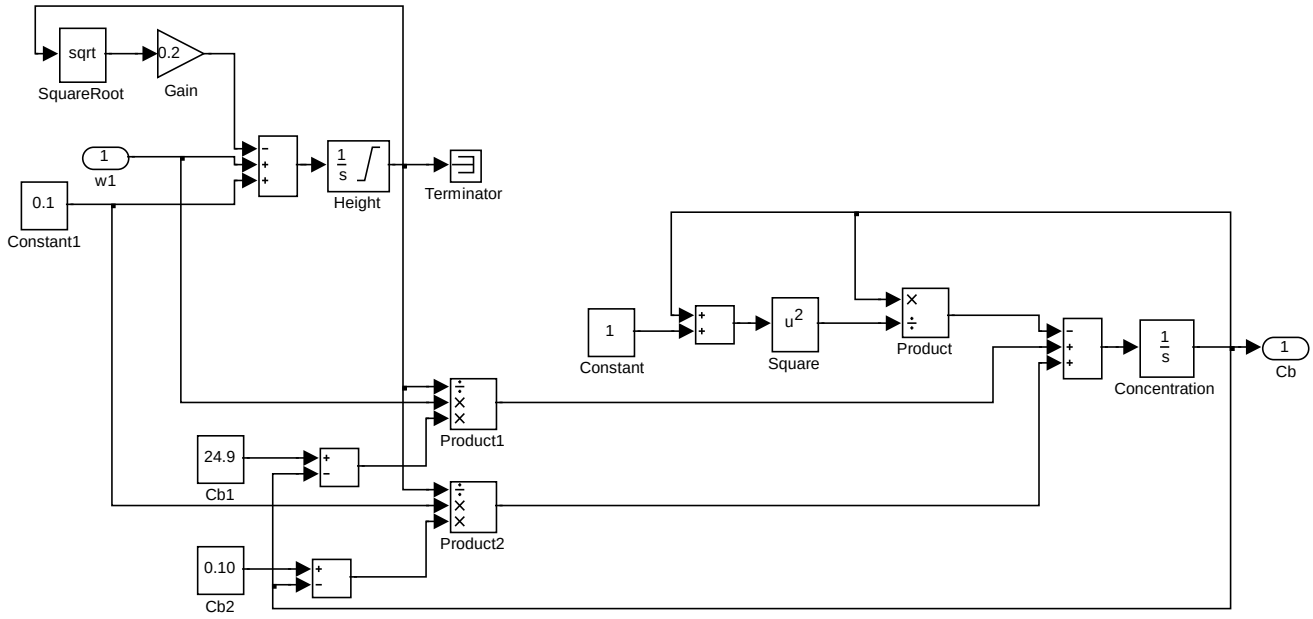


Рис.39. Структурна динамічна модель реактора

Структурна схема, що пояснює принцип побудови системи управління показана на рис.40.

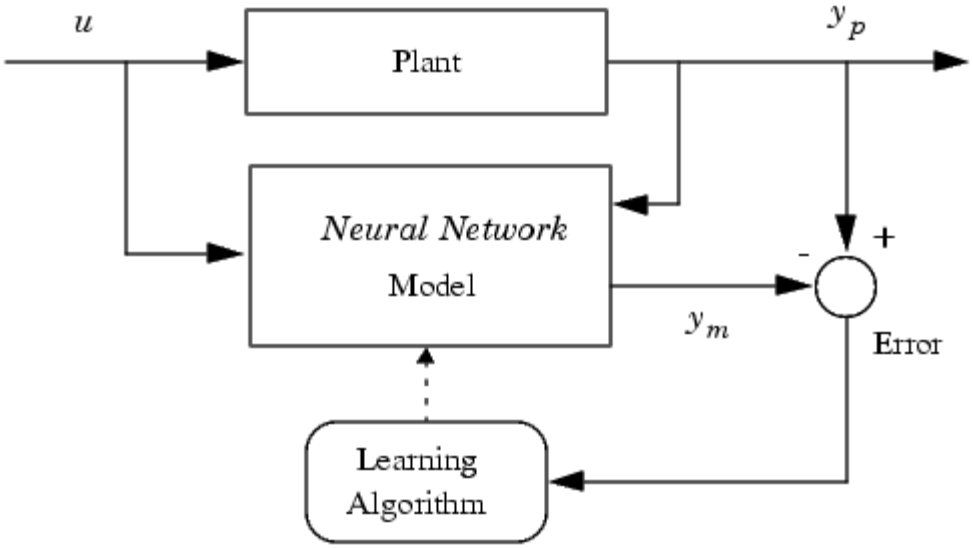


Рис.40. Система управління (Plant – об'єкт управління, Learning algorithm – повчальний алгоритм

Автоматичний регулятор складається з нейронної мережевої моделі об'єкту і блоку оптимізації рис.41). Блок оптимізації визначає оптимальне значення управляючого параметра (u), який подається на вхід об'єкту.

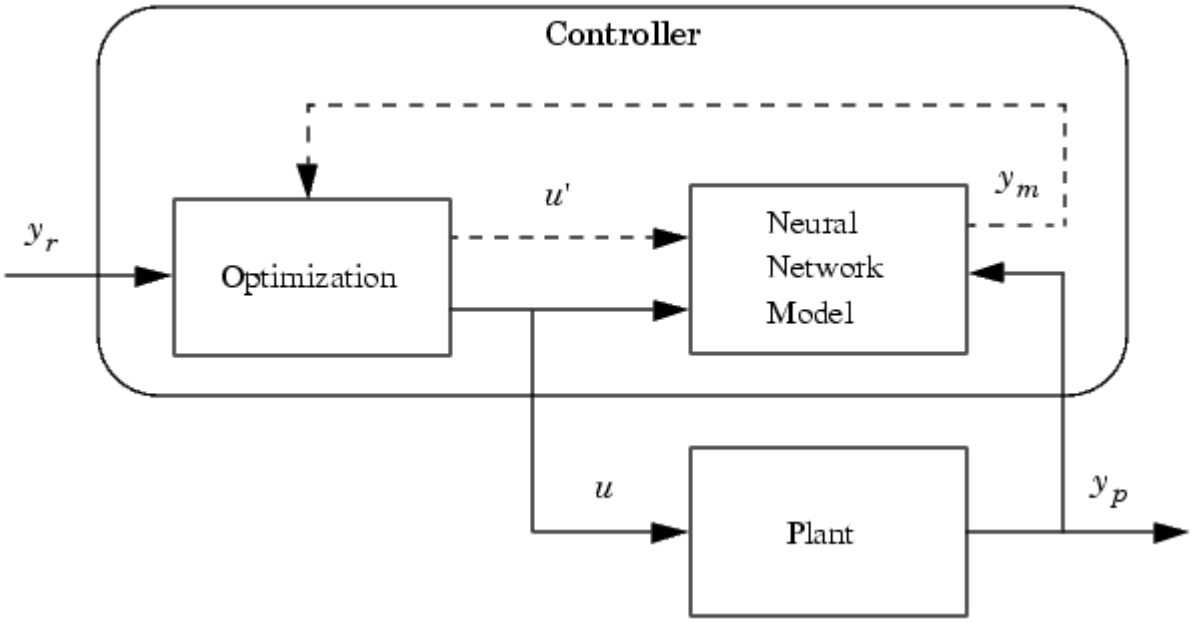


Рис.41.

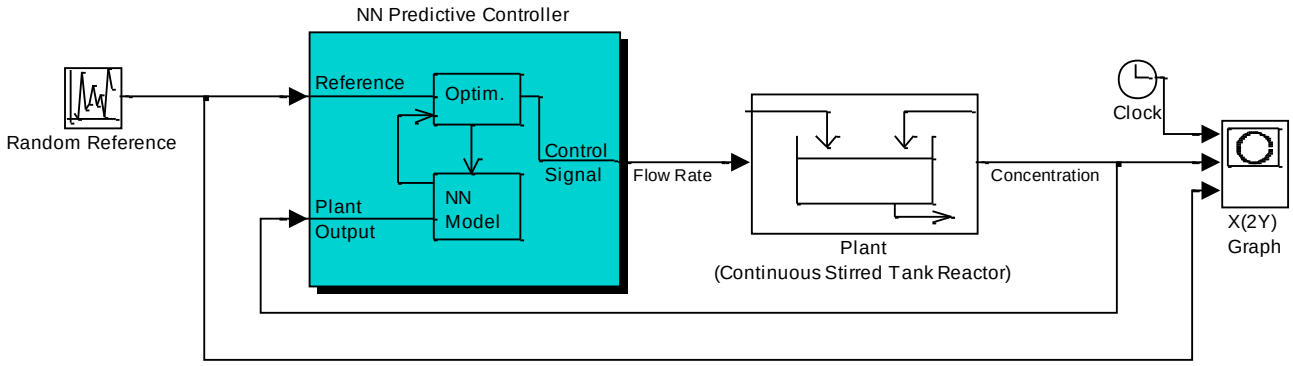


Рис. 42. Структура нейромережевої системи управління ректора повного перемішування (random reference - програма генерує дані, що навчаються, застосовуючи випадкові сигнали до Simulink моделі об'єкту).

На рис.43 показано як будується НМ моделі регулятора у вікні Neural Network Predictive Control і вікні Plant Identiffication (ρ – ваговий чинник, α – параметр оптимізації). На рис.44 показаний графік регулювання концентрації продукту на виході реактора.

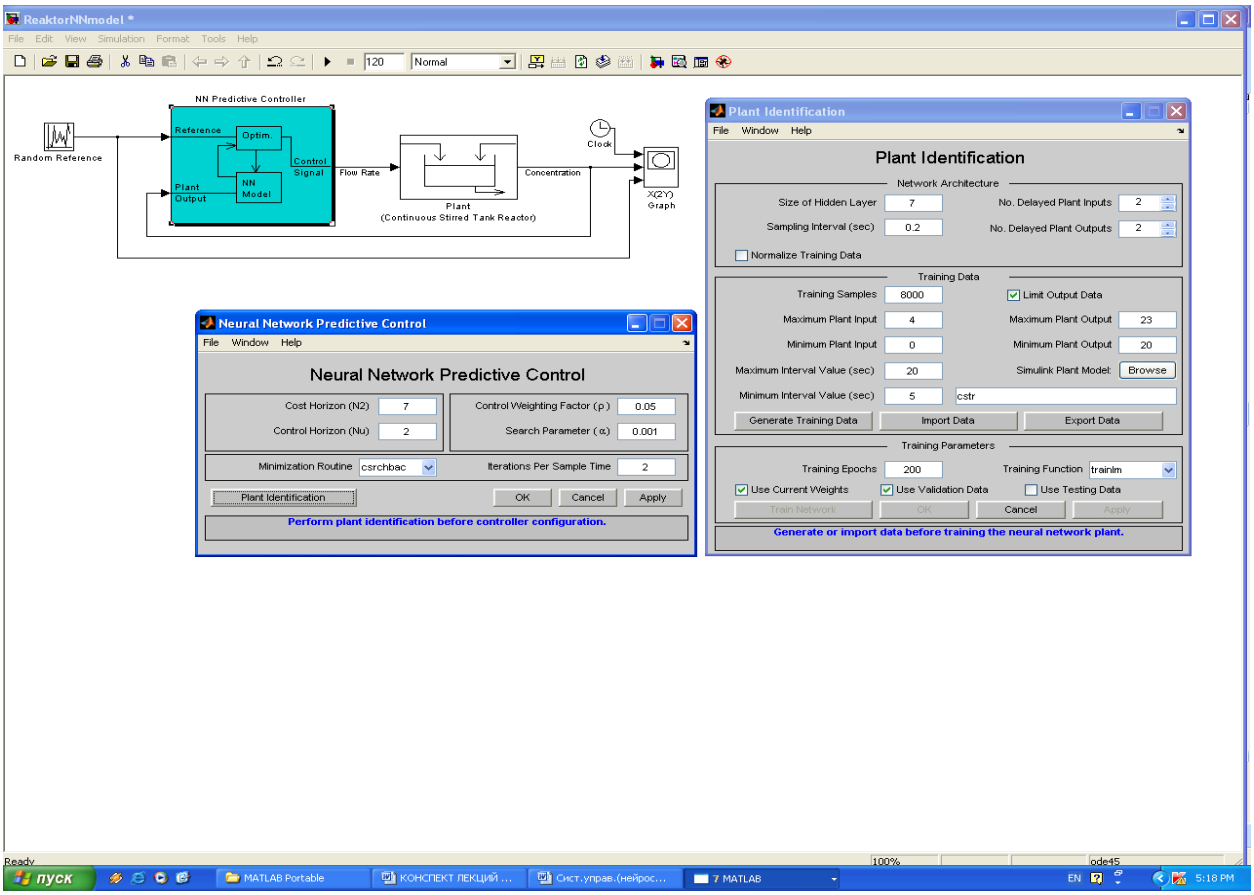


Рис.43.

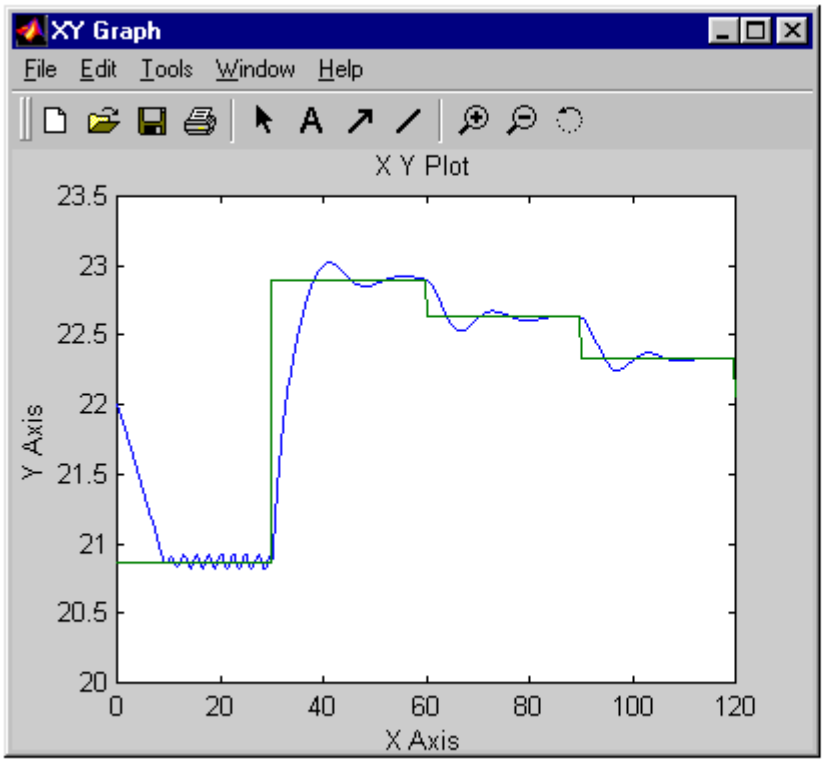


Рис.44. Графік регулювання концентрації продукту на виході реактора.

Лекція 20. Нейро-нечітке моделювання в середовище MatLab

Як правило, в стані нестабільної економіки, що характеризується частою зміною економічних умов, прийняття управлінських рішень здійснюється в умовах невизначеності, унаслідок чого завдання планування економічної діяльності та прогнозування її результатів є однією з найскладніших та неоднозначних. Існує ряд класичних методів прогнозування економічних показників, що базуються на апараті математичної статистики, серед яких виділяються методи аналізу і моделювання часових рядів, методи багатовимірного регресійного аналізу. Особливістю зазначених методів є необхідність чіткої специфікації конструйованих моделей, крім того, додаткові труднощі для використання даних методів створює нестаціонарність досліджуваних економічних процесів. Перспективним напрямком в області вирішення задач прогнозування є застосування апарату штучних нейронних мереж і нейро-нечітких мереж.

Принцип роботи нейронної мережі полягає в тому, що за наявного набору даних конструюється деяка залежність між вхідними і вихідними змінними системи, при цьому в процесі навчання мережі настроюються параметри (ваги) одержуваних функціональних відносин. (Як правило, виявлення і визначення даної залежності в явному вигляді не представляється можливим через вищенаведені причини). Модель нейронної мережі позиціонується як "чорний ящик" унаслідок того, що внутрішній алгоритм настройки "прозорий", й одержані результати і взаємозв'язки складно інтерпретувати.

Нечіткі нейронні мережі або гібридні мережі покликані об'єднати в собі переваги нейронних мереж і систем нечіткого виводу. З одного боку, вони дозволяють розробляти і представляти моделі систем у формі правил нечітких продукцій, які володіють наочністю та простотою змістовної інтерпретації. З іншого боку, для побудови правил нечітких продукцій використовуються методи нейронних мереж, що є більш зручним і менш трудомістким процесом для системних аналітиків.

Метою даного розділу є вивчення і засвоєння методів моделювання та принципів функціонування нейро-нечітких мереж, у тому числі при вирішенні задач економічного прогнозування, а також набуття навичок з конструювання нейро-нечітких мереж у середовищі MATLAB.

Для прогнозування використовуємо нечітку мережу TSK. Узагальнену схему виведення моделі ЦК при використанні M правил і N змінних x_i можна представити у вигляді:

$$\text{IF } (x_1.\text{IS}.A_1^{(1)}).\text{AND}.(x_2.\text{IS}.A_2^{(1)}).\text{AND}. \ldots .\text{AND}.(x_n.\text{IS}.A_n^{(1)}),$$
$$\text{THEN } y_1 = p_{10} + \sum_{j=1}^{N} p_{1j}x_j$$
$$\ldots \qquad \ldots \qquad \ldots$$
$$\text{IF } (x_1.\text{IS}.A_1^{(M)}).\text{AND}.(x_2.\text{IS}.A_2^{(M)}).\text{AND}. \ldots .\text{AND}.(x_n.\text{IS}.A_n^{(M)}),$$
$$\text{THEN } y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj}x_j \ .$$

Умова IF (xi.IS.Ai) реалізується функцією фазифікації, яка представляється узагальненою функцією Гаусса окремо для кожної змінної x_i:

$$\mu_A(x_i) = \frac{1}{1+\left(\frac{x_i - c_i}{\sigma_i}\right)^{2bi}}, \qquad (1)$$

де $\mu_A(x_i)$ є оператором A_i. У нечітких мережах доцільно ставити цю умову у вигляді алгебраїчного добутку, з якого випливає, що для k-го правила виводу .

$$\mu_A^{(k)}(x) = \prod_{j=1}^{N}\left[\frac{1}{1+\left(\frac{x_i - c_j^{(k)}}{\sigma_j^{(k)}}\right)^{2b_i^{(k)}}}\right] \qquad (2)$$

При M правила виводу агрегування вихідного результату мережі здійснюється за формулою

$$y = \sum_{i=1}^{M}\frac{w_i}{\sum_{j=1}^{N} w_j}\left(p_{io} + \sum_{j=1}^{N} p_{ij}x_j\right) \qquad (3)$$

яку можна представити у вигляді

$$y(x) = \frac{1}{\sum_{k=1}^{N} w_k} \sum_{k=1}^{M} w_k y_k(x), \text{ де } y_k(x) = p_{ko} + \sum_{j=1}^{N} p_{kj} x_j . \qquad (4)$$

Присутні в цьому виразі ваги w_k інтерпретуються як значущість $\mu_A^{(k)}(x)$ компонентів , визначених формулою (1). За цієї умови з формулою (4) можна зіставити багатошарову структуру мережі (рис.4). У такій мережі виділяється п'ять шарів.

• Перший шар виконує роздільну фазифікацію кожної змінної x_i (i=1,2, ... , N), визначаючи для кожного k-го правила виводу значення коефіцієнта приналежності $\mu_A^{(k)}(x_i)$ відповідно до застосовуваної функцією фазифікації. Це параметричний шар з параметрами $c_j^{(k)}$, $\sigma_j^{(k)}$, $b_j^{(k)}$, що підлягають адаптації у процесі навчання.

• Другий шар виконує агрегування окремих змінних x_i, визначаючи результуюче значення коефіцієнта приналежності $w_k = \mu_A^{(k)}(x)$ для вектора x (рівень активізації правила виводу) відповідно до формули (2). Це шар непараметричний.

• Третій шар являє собою генератор функції TSK, розраховує значення $y_k(x) = p_{ko} + \sum_{j=1}^{N} p_{kj} x_j$. У цьому шарі також проводиться множення сигналів $y_k(x)$ на значення w_k, сформовані на попередньому шарі. Це параметричний шар, в якому адаптації підлягають лінійні ваги p_{kj} для k = 1,2, ... ,M та j = 1,2, ... ,N, визначають функцію слідства моделі TSK.

• Четвертий шар складають два нейрона-суматора, один з яких розраховує зважену суму сигналів $y_k(x)$, а другий визначає суму ваг $\sum_{k=1}^{M} w_k$. Це непараметричний шар.

• Останній, п'ятий шар, що складається з єдиного вихідного нейрона,- це нормалізуючий шар, в якому піддаються нормалізації ваги відповідно до формули (4). Вихідний сигнал y(x) визначається виразом, що відповідає залежності (3),

$$y(x) = f(x) = \frac{f_1}{f_2} . \qquad (5)$$

З наведеного опису випливає, що нечітка мережа TSK містить лише два параметричних шари (перший і третій), параметри яких уточнюються в процесі навчання. Параметри першого шару будемо називати нелінійними параметрами, оскільки вони відносяться до нелінійної функції (1), а параметри третього шару – лінійними вагами, так як вони відносяться до параметрів p_{ki} лінійної функції TSK.

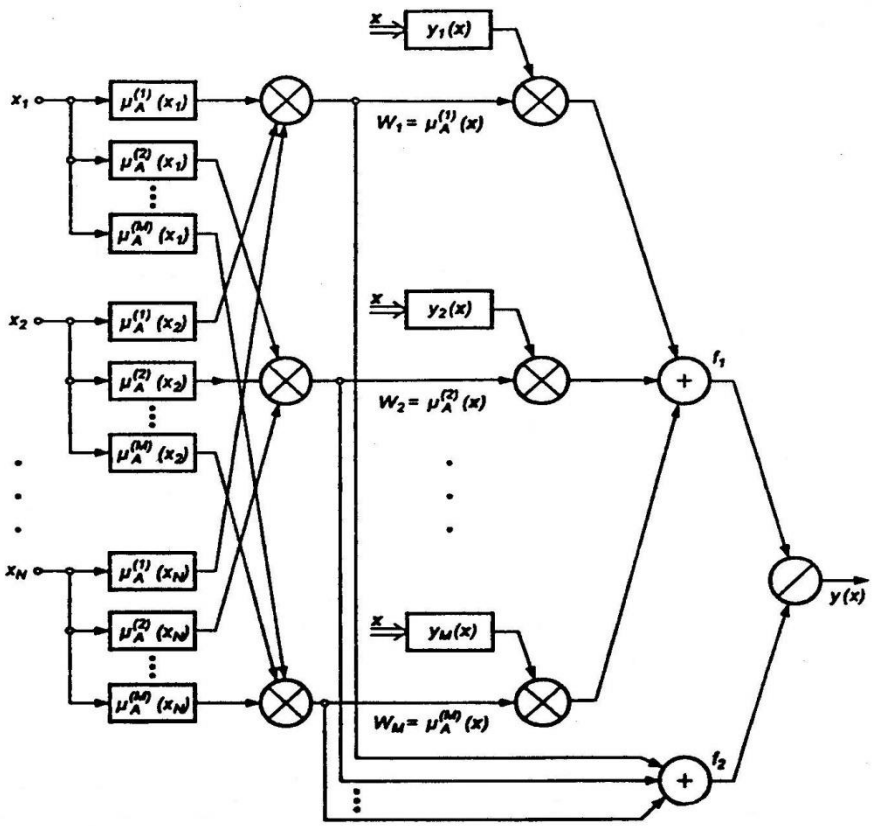


Рис. 45. Структура нечіткої нейронної мережі TSK

При уточненні функціональної залежності (4) для мережі ЦК отримуємо:

$$y(x) = \frac{1}{\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]}\sum_{k=1}^{M}\left[\prod_{j=1}^{N}\mu_A^{(k)}(x_j)\right]\left[p_{k0}+\sum_{j=1}^{N}p_{kj}x_j\right]. \qquad (6)$$

Якщо прийняти, що в конкретний момент часу параметри умови зафіксовані, то функція y(x) є лінійною відносно змінних x_i (i=1,2, ... ,N). При наявності N вхідних змінних кожне правило формує N+1 змінних $p_j^{(k)}$ лінійної залежності TSK. При M правила виводу це дає M(N+1) лінійних параметрів мережі. У свою чергу, кожна функція приналежності використовує три

параметра (c,s,b), що підлягають адаптації. Якщо прийняти, що кожна змінна x_i характеризується власною функцією приналежності, то за M правил виводу ми отримаємо 3MN нелінійних параметрів. Разом це дає M(4N+1) лінійних і нелінійних параметрів, значення яких повинні підбиратися в процесі навчання мережі. На практиці для зменшення кількості параметрів, що адаптуються, оперують меншою кількістю незалежних функцій приналежності для окремих змінних, керуючись правилами, в яких комбінуються функції приналежності змінних. Якщо прийняти, що кожна змінна xi має т різних функцій приналежності, то максимальна кількість правил, які можна створити при їх комбінуванні, складе: $M = m^N$ (при трьох функціях приналежності, що поширюються на дві змінні, це $3^2 = 9$ правил виводу). Таким чином, сумарна кількість нелінійних параметрів мережі при M правилах виведення зменшується з 3MN у загальному випадку до $3NM^{1/N}$. Кількість лінійних параметрів при подібній модифікації залишається без змін, тобто M(N+1).

Гібридна мережа як адаптивна система нейро-нечіткого виводу

Гібридна мережа являє собою багатошарову нейронну мережу спеціальної структури без зворотних зв'язків, у якій використовуються звичайні (не нечіткі) сигнали, ваги і функції активації, а виконання операції підсумовування засноване на використанні фіксованої Т-норми, Т-конорми або деякої іншої безперервної операції. При цьому значення входів, виходів і ваг гібридної нейронної мережі являють собою дійсні числа з відрізка [0, 1].

Основна ідея, покладена в основу моделі гібридних мереж, полягає в тому, щоб використовувати існуючу вибірку даних для визначення параметрів функцій приналежності, які найкраще відповідають деякій системі нечіткого виводу. При цьому для знаходження параметрів функцій приналежності використовуються відомі процедури навчання нейронних мереж.

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі реалізовані у формі так званої адаптивної системи нейро-нечіткого виводу ANFIS. З одного боку, гібридна мережа ANFIS являє собою нейронну мережу з єдиним виходом і кількома входами, які являють собою нечіткі лінгвістичні змінні. При цьому терми вхідних лінгвістичних змінних описуються стандартними для системи MATLAB функціями приналежності, а терми вихідної змінної представляються лінійної або постійною функцією приналежності. З іншого боку, гібридна мережа ANFIS являє собою систему нечіткого виводу FIS типу Сугено

нульового або першого порядку, в якій кожне з правил нечітких продукцій має постійну вагу, що дорівнює 1.

Лекція 21. Моделювання і реалізація нейро-нечіткої мережі в середовищі MATLAB

У пакеті Fuzzy Logic Toolbox системи MATLAB гібридні мережі реалізовані у формі адаптивних систем нейро-нечіткого виводу ANFIS. При цьому розробка та дослідження гібридних мереж виявляється можливою:

• в інтерактивному режимі за допомогою спеціального графічного редактора адаптивних мереж, що отримав назву редактора ANFIS;

• у режимі командного рядка за допомогою введення імен відповідних функцій з необхідними аргументами безпосередньо у вікно команд системи MATLAB.

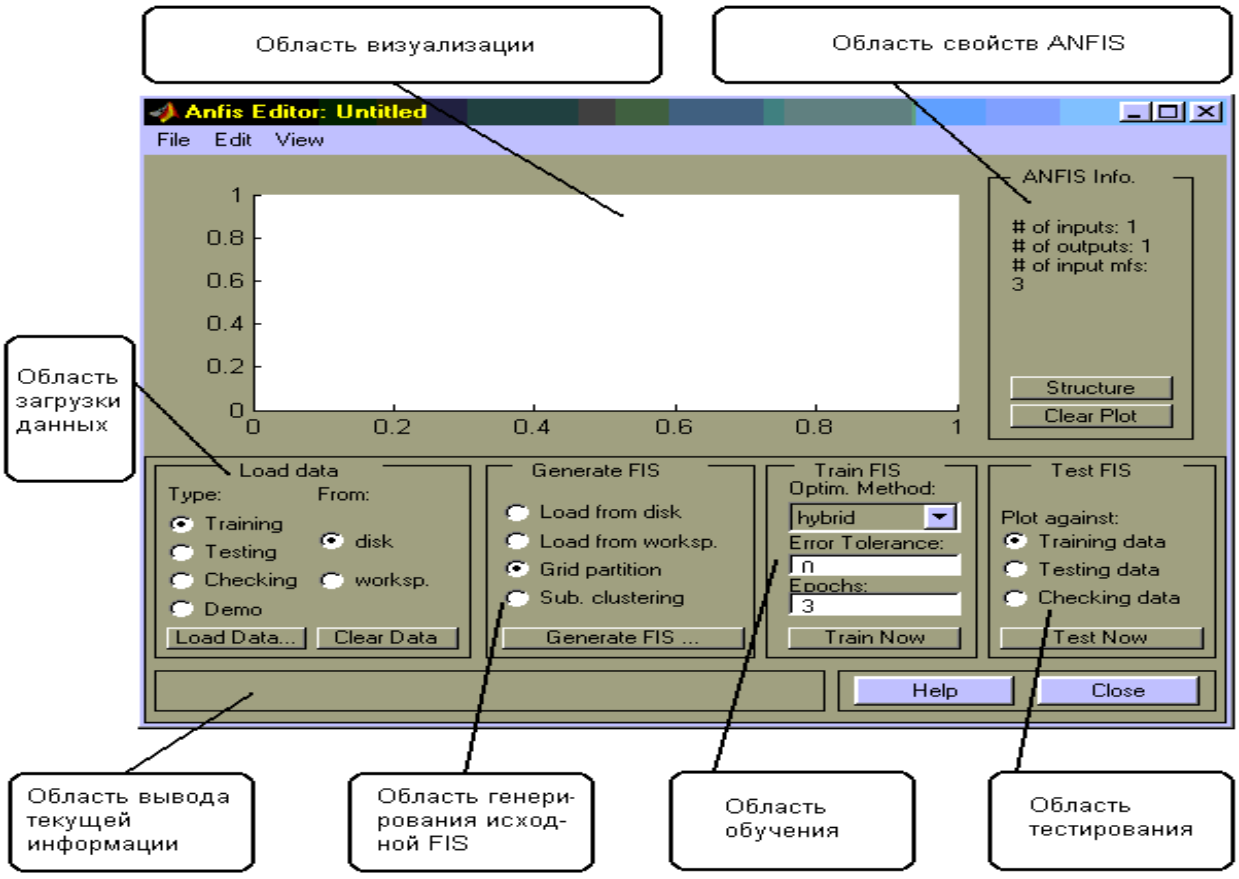


Рис. 46. Основне вікно ANFIS-редактора

Редактор ANFIS дозволяє створювати або завантажувати конкретну модель адаптивної системи нейро-нечіткого виводу, виконувати її навчання, візуалізувати її структуру, змінювати і налаштовувати її параметри, а також

використовувати налагоджену мережу для отримання результатів нечіткого виводу. Опис ANFIS-редактора ANFIS є абревіатурою Adaptive Neuro-Fuzzy Inference System – (адаптивна нейро-нечітка система). ANFIS-редактор дозволяє автоматично синтезувати з експериментальних даних нейро-нечіткі мережі. Нейро-нечітку мережу можна розглядати як одну з різновидів систем нечіткого логічного виводу типу Сугено. При цьому функції приналежності синтезованих систем налаштовані (навчені) так, щоб мінімізувати відхилення між результатами нечіткого моделювання та експериментальними даними. Завантаження ANFIS-редактора здійснюється за командою. У результаті виконання цієї команди з'явиться графічне вікно (рис. 46.). На **anfisedit** цьому ж рисунку вказані функціональні області ANFIS-редактора, опис яких наведено нижче.

ANFIS-редактор містить 3 верхніх меню **File**, **Edit** й **View** та область візуалізації, область властивостей ANFIS, область завантаження даних, область генерування вихідної системи нечіткого логічного виводу, область навчання, тестування, область відображення поточної інформації, а також кнопки **Help** і **Close**, які дозволяють викликати вікно довідки і закрити ANFIS-редактор. Меню **File й View** та однакові для всіх GUI-модулів, що використовуються з системами нечіткого логічного виводу.

Меню **Edit**

Загальний вигляд меню наведено на рис.47.

Рис. 47. Меню Edit

Команда **Undo** скасовує раніше виконану дію. Виконується також за допомогою натискання Ctrl+Z..

Команда **FIS Properties**... відкриває FIS-редактор. Ця команда може бути також виконана натисканням Ctrl+1.

Команда **Membership Functions**... відкриває редактор функцій приналежностей. Ця команда може бути також виконана натисканням Ctrl+2.

Команда **Rules**... відкриває редактор бази знань. Ця команда може бути також виконана натисканням Ctrl+3.

Команда **Anfis**... відкриває ANFIS-редактор. Ця команда може бути також виконана натисканням Ctrl+3. Зауважимо, що дана команда, запущена з ANFIS-редактора не приводить до виконання яких-небудь дій, так як цей редактор вже відкритий. Однак, у меню **Edit** інших GUI-модулів, що використовються з системами нечіткого логічного виводу, додається команда **Anfis**..., що дозволяє відкрити ANFIS-редактор з цих модулів.

Область візуалізації

У цій області з'являється два типи інформації:

– при навчанні системи – крива навчання у вигляді графіка залежності помилки навчання від порядкового номера ітерації;

– при завантаженні даних і тестування системи – експериментальні дані та результати моделювання.

Експериментальні дані та результати моделювання виводяться у вигляді множини точок у двовимірному просторі. При цьому на осі абсцис відкладається порядковий номер рядка даних у вибірці (навчальної, тестувальної або контрольної), а на осі ординат - значення вихідної змінної для даного рядка вибірки. Використовуються такі маркери:

блакитна крапка (.) – тестувальна вибірка;

блакитна окружність (o) – навчальна вибірка;

блакитний плюс (+) – контрольна вибірка;

червона зірочка (*) – результати моделювання.

Область властивостей ANFIS

В області властивостей ANFIS (ANFIS info) виводиться інформація про кількість вхідних і вихідних змінних, про кількість функцій приналежностей для кожної вхідної змінної, а також про кількість рядків у вибірках. У цій області розташовані дві кнопки: **Structure** і **Clear Plot**. Натискання кнопки Structure відкриває нове графічне вікно, в якому система нечіткого логічного виводу представляється у вигляді нейро-нечіткої мережі. В якості ілюстрації наведена нейро-нечітка мережа, що містить чотири вхідних змінних та одну вихідну (рис. 48). У цій системі по три терми лінгвістичних використовуються

для оцінки кожної з вхідних змінних і чотири терми - для вихідної. Натискання кнопки **Clear Plot** дозволяє очистити область візуалізації.

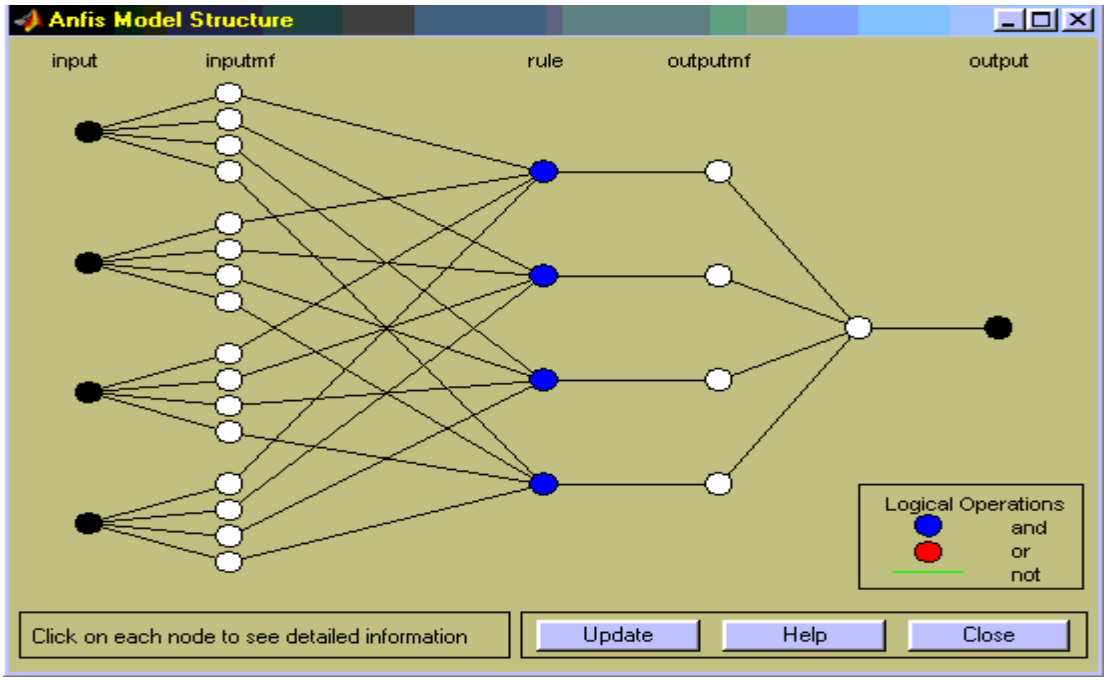


Рис. 48. Приклад структури нейро-нечіткої мережі

Область завантаження даних

В області завантаження даних (**Load data**) розташовані:
меню вибору типу даних (**Type**), що містить альтернативи:
Traning - навчальна вибірка;
Testing - тестувальна вибірка;
Checking - контрольна вибірка;
Demo - демонстраційний приклад; меню вибору джерела даних (From), що містить альтернативи:
disk – диск;
worksp. - робоча область MatLab; кнопка завантаження даних **Load Data**... при натисканні якої з'являється діалогове вікно вибору файлу, якщо завантажуються дані з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних відбувається з робочої області;
кнопка очищення даних **Clear Data**.

Примітка. Протягом одного сеансу роботи ANFIS-редактора можна завантажувати дані з одного формату, тобто кількість вхідних змінних у вибірках повинна бути однаковою.

Область генерування вихідної системи нечіткого логічного виводу

В області генерування (**Generate FIS**) розташоване меню вибору способу створення вихідної системи нечіткого логічного виводу. Меню містить такі альтернативи:

Load from disk – завантаження системи з диска;

Load from worksp. – завантаження системи з робочої області MatLab;

Grid partition - генерування системи за методом решітки (без кластеризації);

Sub. clustering – генерування системи за методом субкластеризации.

В області також розташована кнопка **Generate**, після натискання якої генерується початкова система нечіткого логічного виводу. При виборі Load from disk з'являється стандартне діалогове вікно відкриття файлу.

При виборі **Load from worksp.** з'являється стандартне діалогове вікно введення ідентифікатора системи нечіткого логічного виводу.

При виборі **Grid partition** з'являється вікно введення параметрів методу решітки (рис. 49), в якому потрібно вказати кількість термів для кожної вхідної змінної і тип функцій приналежності для вхідних та вихідної змінних.

При виборі Sub. clustering з'являється вікно введення таких параметрів методу субкластеризації (рис. 50):

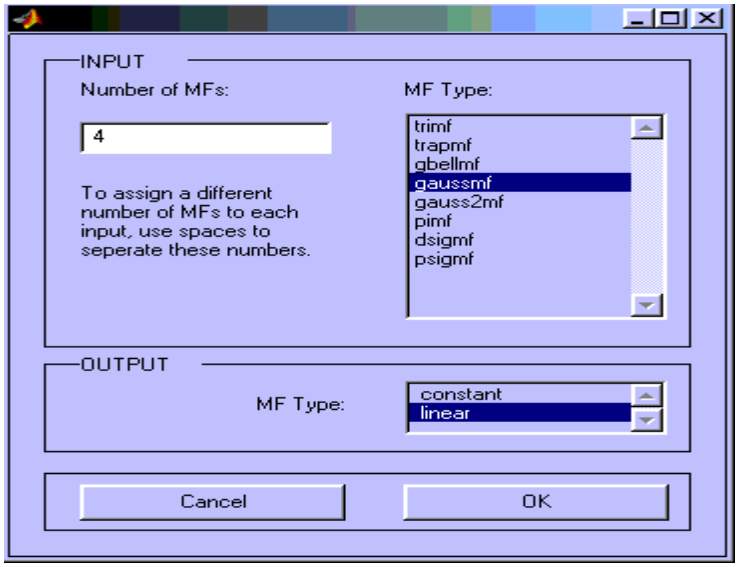


Рис. 49. Вікно введення параметрів для методу решітки

Range of influence – рівні впливу вхідних змінних; Squash factor – коефіцієнт переваження;

Accept ratio – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинен бути вищим від потенціалу центру першого кластера для того, щоб центром одного з кластерів була призначена розглянута точка;

Reject ratio – коефіцієнт, що встановлює у скільки разів потенціал даної точки повинен бути нижче від потенціалу центру першого кластера, щоб розглянута точка була виключена з можливих центрів кластерів.

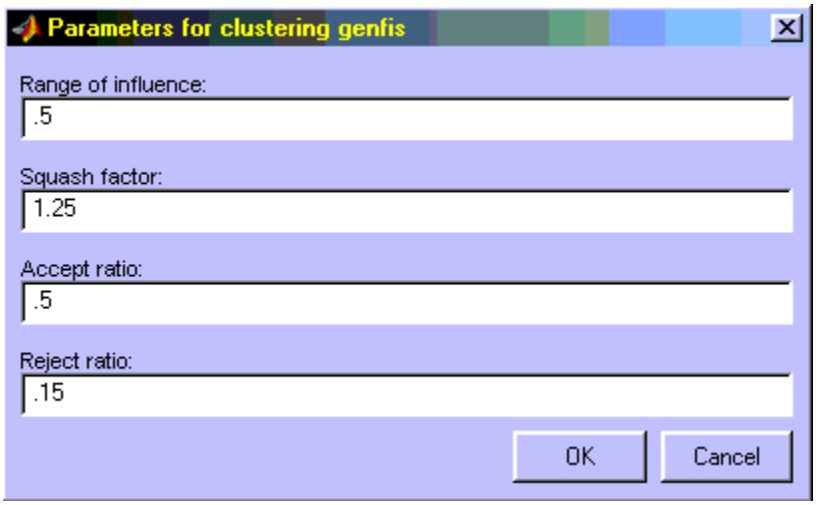


Рис.50. Вікно введення параметрів для методу субкластеризації

Область навчання

В області навчання (**Train FIS**) розташовані меню вибору методу оптимізації (Optim. method), поле завдання необхідної точності навчання (Error tolerance), поле завдання кількості ітерацій навчання (Epochs) і кнопка Train Now, натискання якої запускає режим навчання. Проміжні результати навчання виводяться в область візуалізації та в робочу область MatLab. У ANFIS-редакторі реалізовані два методи навчання:

backpropa – метод зворотного поширення помилки, заснований на ідеях методу найшвидшого спуску;

hybrid – гібридний метод, який поєднує метод зворотного поширення помилки з методом найменших квадратів.

Область тестування

В області тестування (Test FIS) розташовані меню вибору вибірки та кнопка Test Now, при натисненні якої відбувається тестування нечіткої системи з виведенням результатів в область візуалізації.

Область відображення поточної інформації

У цій області виводиться найбільш істотна поточна інформація, наприклад, повідомлення про закінчення виконання операцій, значення помилки навчання або тестування і т. і.

Лекція 22. Синтез нейро-нечіткої мережі в середовищі MATLAB

Опис завдання: Є вихідні дані індексу РТС за період з 01.03.2004 по 30.04.2004. Потрібно побудувати нейро-нечітку мережу і спрогнозувати значення індексу на 1.05.2004.

Загальна послідовність процесу розробки моделі гібридної мережі може бути представлена у такому вигляді.

1. Підготовка файлу з навчальними даними. Доцільно скористатися редактором електронних таблиць MS Excel. Навчальну вибірку необхідно зберегти у зовнішньому файлі з розширенням *.dat.

Приклад «*Фрагмент навчальної вибірки для аналізу і прогнозування індексу РТС*». Алгоритм прогнозування має на увазі те, що кожне наступне значення розраховується на основі декількох попередніх (Табл. 10).

Таблиця 10. Набір даних для навчання нейро-нечіткої мережі

Перша вхідна змінна	Друга вхідна змінна	Третя вхідна змінна	Вихідна змінна
688.72	686.21	667.27	669.26
686.21	667.27	669.26	673.25
667.27	669.26	673.25	688.68
669.26	673.25	688.68	680.86
673.25	688.68	680.86	671.33
688.68	680.86	671.33	669.55
680.86	671.33	669.55	676.20
671.33	669.55	676.20	680.57
669.55	676.20	680.57	698.70
676.20	680.57	698.70	708.59

Продовження табл.10

680.57	698.70	708.59	721.81
698.70	708.59	721.81	712.88
708.59	721.81	712.88	713.14
721.81	712.88	713.14	705.16
712.88	713.14	705.16	706.71
713.14	705.16	706.71	716.55
705.16	706.71	716.55	721.35
706.71	716.55	721.35	736.28
688.72	686.21	667.27	669.26
686.21	667.27	669.26	673.25
667.27	669.26	673.25	688.68
669.26	673.25	688.68	680.86
673.25	688.68	680.86	671.33
688.68	680.86	671.33	669.55
680.86	671.33	669.55	676.20
671.33	669.55	676.20	680.57

2. Відкрити редактор ANFIS. Завантажити файл з навчальними даними. Кнопка завантаження даних **Load Data,** при натисканні якої з'являється діалогове вікно вибору файлу, якщо завантаження даних з диска, або вікно введення ідентифікатора вибірки, якщо завантаження даних відбувається з робочої області.

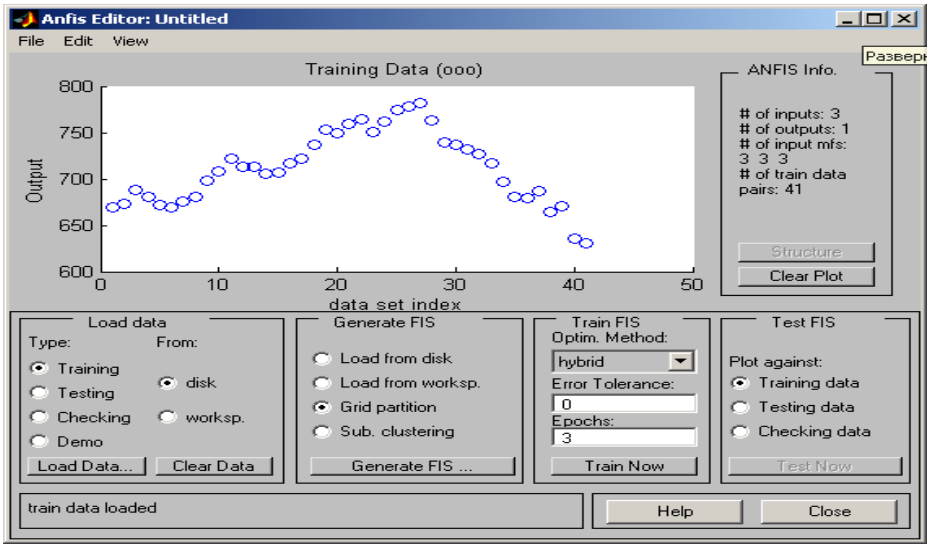


Рис. 51. Графічний інтерфейс редактора ANFIS після завантаження навчальних даних

3. Після підготовки і завантаження навчальних даних можна створити структуру системи нечіткого виводу FIS типу Сугено, яка є моделлю гібридної мережі в системі Matlab. Для цієї мети слід скористатися кнопкою **Generate FIS** у нижній частині робочого вікна редактора. При цьому перші 2 опції відносяться до попередньо створеної структури гібридної мережі, а 2 останніх – до форми розбиття вхідних змінних моделі. Перед генерацією структури системи нечіткого виводу типу Сугено після виклику діалогового вікна властивостей задамо для кожної з вхідних змінних на 3 лінгвістичних терми, а в якості типу їх функцій приналежності виберемо трикутні функції. Після натискання кнопки Generate FIS викликається діалогове вікно із зазначенням кількості та типу функцій приналежності для окремих термів вхідних змінних і вихідної змінної (рис. 52).

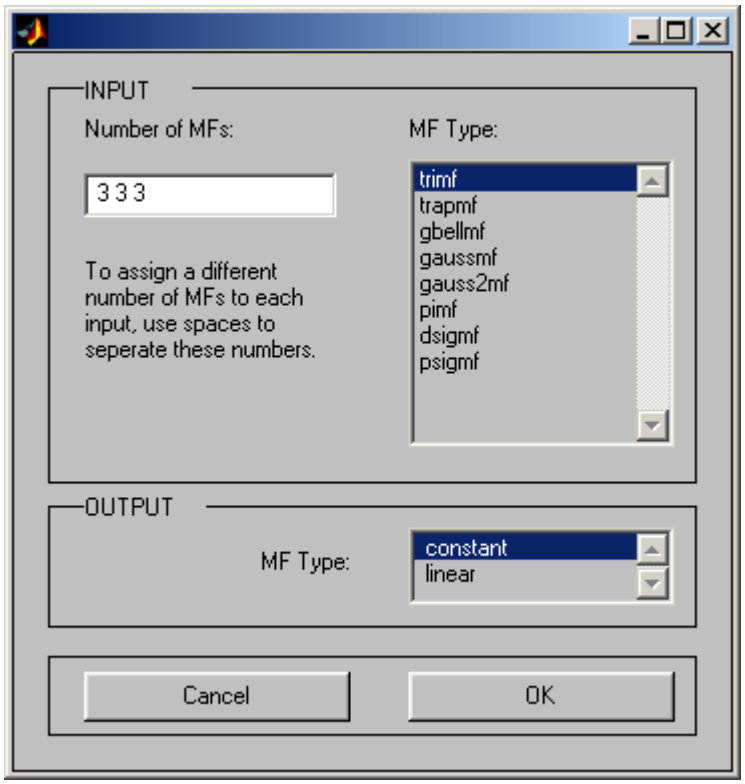


Рис. 52. Діалогове вікно для завдання кількості і типу функцій приналежності 4.

4. Після генерації структури гібридної мережі можна візуалізувати її структуру, для чого слід натиснути кнопку **Structure** у правій частині графічного вікна (рис. 53).

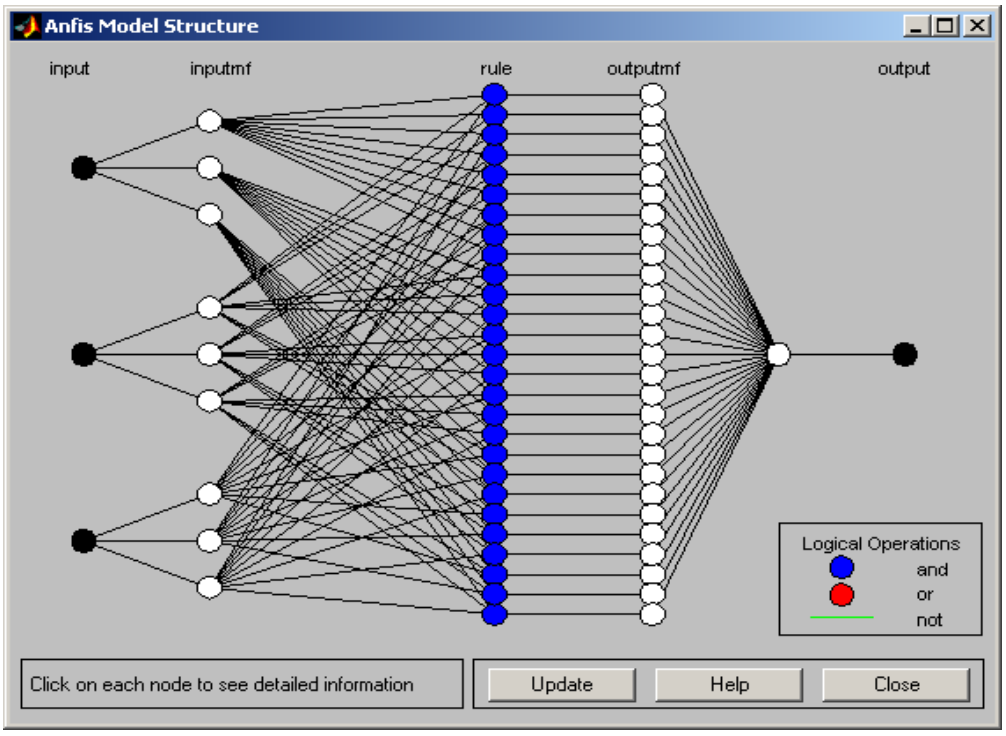


Рис. 53. Структура згенерованої системи нечіткого виводу

Для розглянутого прикладу система нечіткого виводу містить три вхідних змінних з трьома термами кожна, 27 правил нечітких продукцій, одну вихідну змінну з 27 термами.

5. Перед навчанням гібридної мережі необхідно задати параметри навчання, для чого слід скористатися такою групою опцій у правій нижній частині робочого вікна:

1) обрати метод навчання гібридної мережі зворотного поширення (backpropo) або гібридний (hybrid), що представляє собою комбінацію методу найменших квадратів та методу зменшення зворотного градієнта;

2) установити рівень помилки навчання (Error Tolerance) – за замовчуванням значення 0. (змінювати не рекомендується);

3) визначити кількість циклів навчання (Epochs) – за замовчуванням значення 3 (рекомендується збільшити для розглянутого прикладу - задати його значення рівним 40).

Для навчання мережі слід натиснути кнопку Train now. При цьому хід процесу навчання ілюструється у вікні візуалізації у формі графіка залежності помилки від кількості циклів навчання (рис. 54).

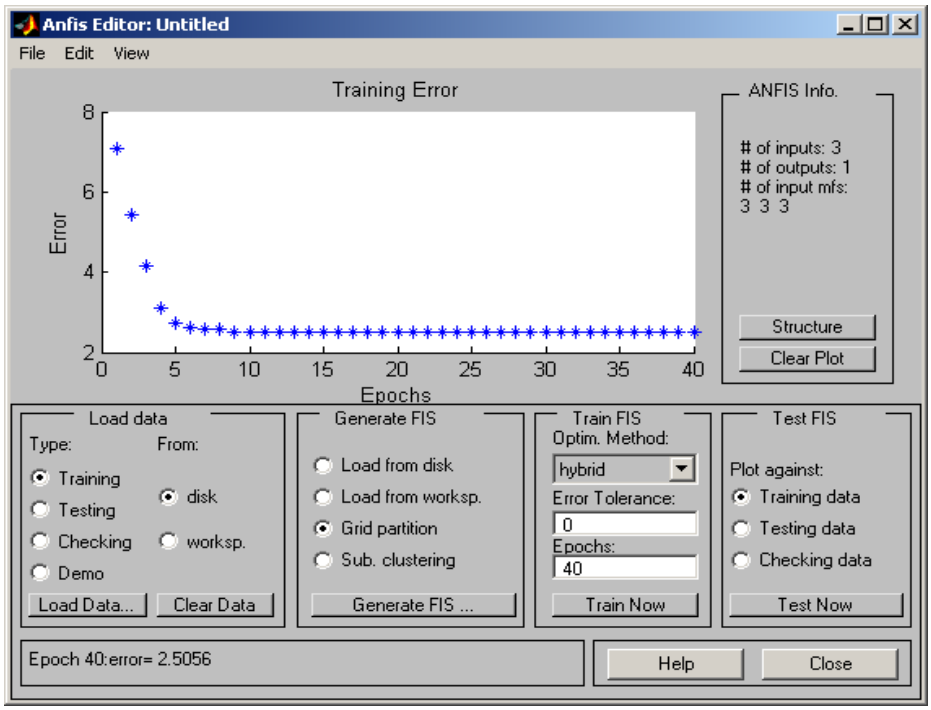


Рис. 54. Графік залежності помилок навчання від кількості циклів навчання

Подальша настройка параметрів побудованої і навченої гібридної мережі може бути виконана за допомогою стандартних графічних засобів пакету Fuzzy Logic Toolbox. Для цього рекомендується зберегти створену систему нечіткого виводу у зовнішньому файлі з розширенням *.fis, після чого слід завантажити цей файл у редактор систем нечіткого виводу FIS.

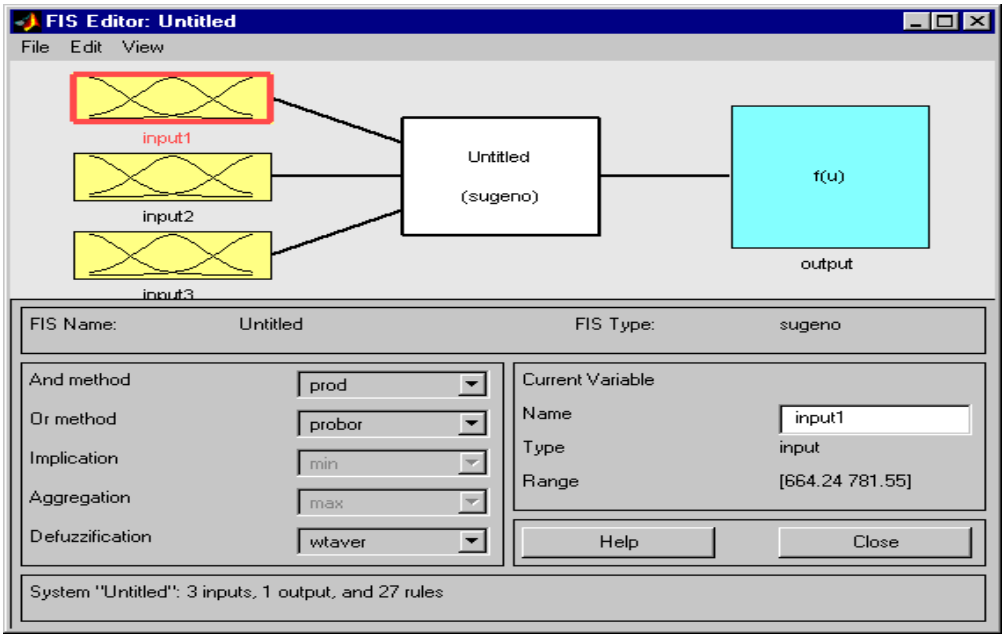


Рис. 55. Графічний інтерфейс редактора FIS для згенерованої системи нечіткого виводу

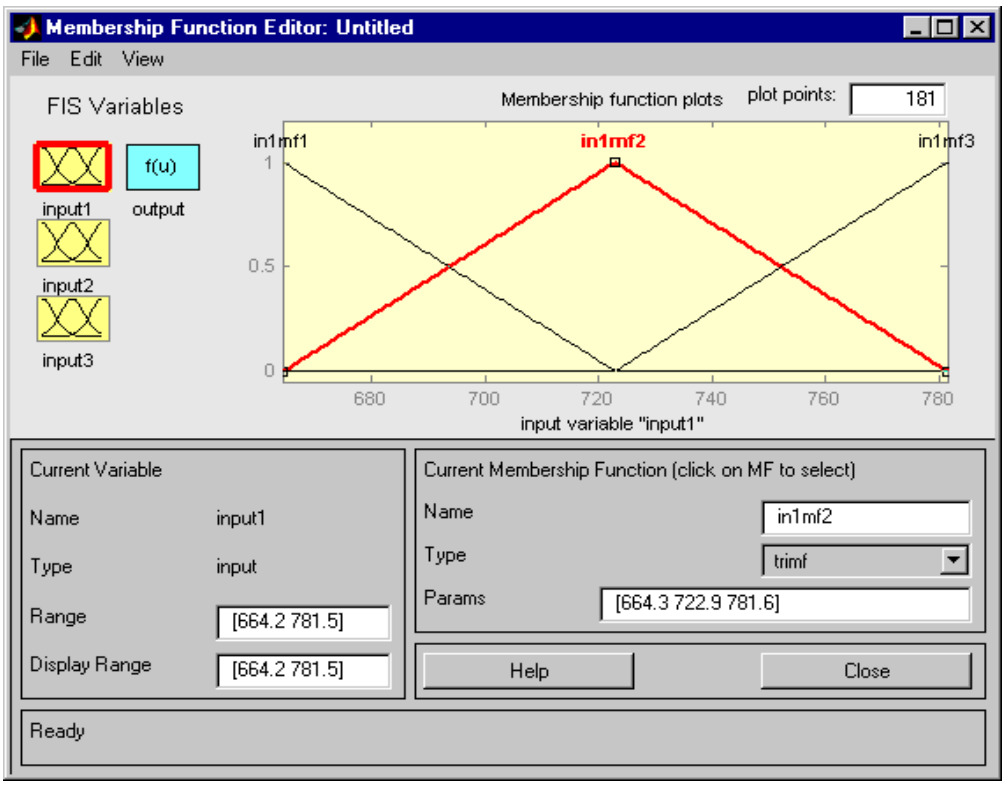


Рис. 56. Графічний інтерфейс редактора функцій приналежності побудованої системи нечіткого виводу

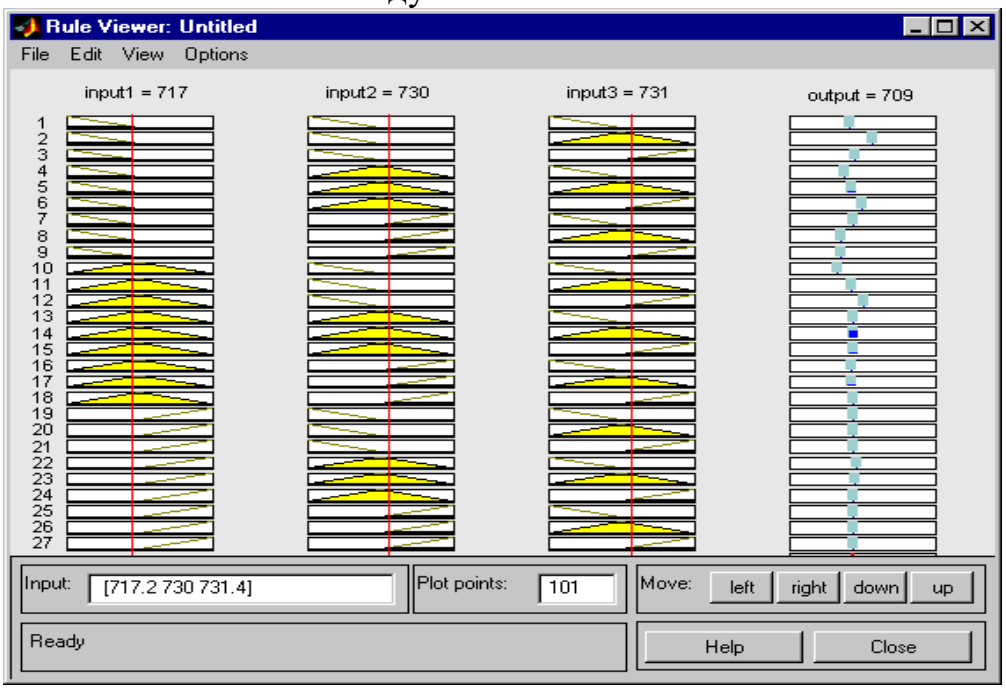


Рис. 57. Графічний інтерфейс перегляду правил згенерованої системи нечіткого виводу

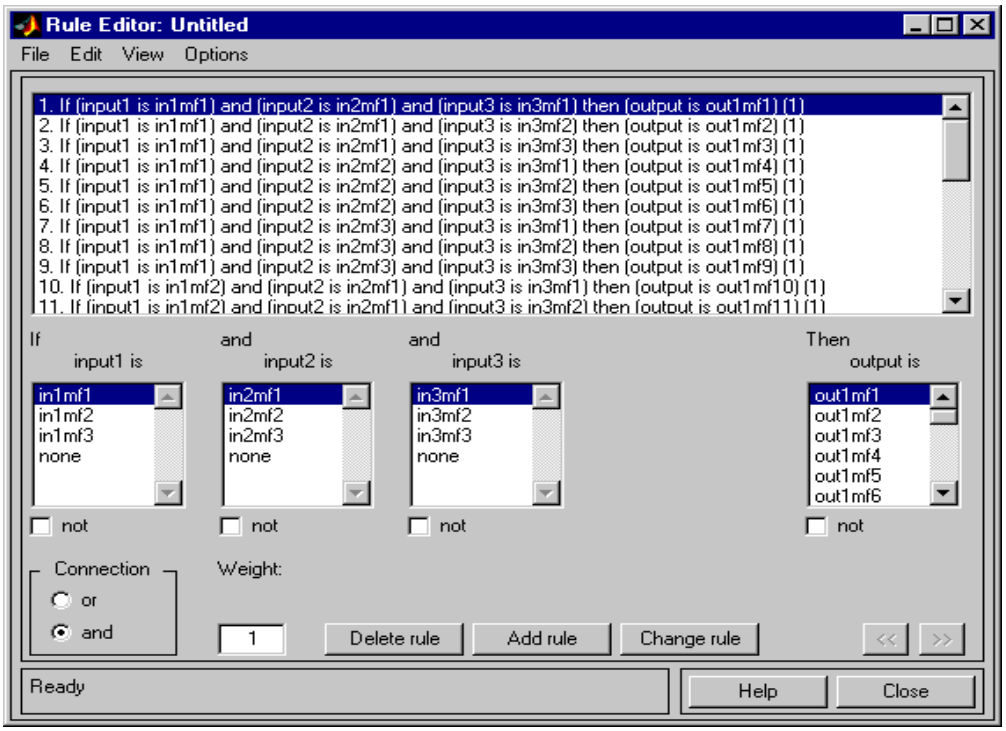


Рис. 58. Фрагмент бази нечітких правил

6. Виконаємо перевірку адекватності побудованої нечіткої моделі гібридної мережі.

Для цього можна спрогнозувати курс долара на визначений день. Для вирішення цього завдання необхідно скористатися функцією evalfis.

Лекція 23. Генетичні алгоритми

Генетичні алгоритми застосовуються для рішення оптимізаційних задач за допомогою методу еволюції, тобто шляхом добору з безлічі рішень найбільш підходящого. У типовій задачі оптимізації існує набір змінних, що впливають на процес, і формула або алгоритм, які використовують ці змінні для побудови моделі цього процесу. При цьому завдання полягає в тому, щоб знайти такі значення змінних, які певним чином оптимізують модель. У разі, якщо моделлю є формула, то зазвичай відшукують максимум або мінімум функції, яку дана формула представляє. Існує багато математичних методів, які вирішують завдання оптимізації в тому випадку, якщо це завдання з «гарною поведінкою». Однак традиційні методи терплять крах, якщо завдання не належить до цього класу. Є великий клас задач, для яких обчислювальні складності ростуть експоненціально з розмірністю задачі. . Прикладами завдань з таким «поганим поводженням» можуть служити комбінаторні задачі, а також задачі,

математичний опис яких не є гладкою безперервною функцією. Тут мінімізується цільова функція (цільова функція, функція вигоди або функція вартості), яка залежить від порядку кінцевого числа об'єктів. Кількість варіантів розміщення N об'єктів, і, отже, зусиль для знаходження мінімуму цільової функції збільшуються в геометричній прогресії із зростанням N.

Сутність еволюційних обчислень

В останні 30 років з'явився інтерес до завдань, вирішення яких грунтується на принципах еволюції та успадкування ознак. Системи подібного роду містять популяцію потенційних рішень, мають певний процес відбору, який використовує критерії придатності індивідуумів (окремих рішень), застосовують деякі оператори рекомбінації. До таких систем відноситься клас еволюційних обчислень (ЕО). Під останніми розуміється термін, використовуваний для опису алгоритмів пошуку, оптимізації або навчання, заснованих на деяких формальних ознаках природного еволюційного відбору [25]. Методи ЕО часто застосовуються для опису процесів еволюції програм або функцій (генетичне програмування), кінцевих автоматів (еволюційне програмування) і систем, заснованих на продукційних правилах (класифікаційні системи). Іноді ЕО разом з НЛ (нечітка логіка) використовуються для навчання нейронних мереж, що призвело до нового терміну «м'які» обчислення", що об'єднали ГА, НМ і ШНС.

Серед методів ЕО можна виділити наступні:

еволюційне програмування – представляє рішення задачі у вигляді універсальних кінцевих автоматів, які реагують на подразнення з зовнішнього середовища;

еволюційні стратегії – кожне рішення знаходиться у вигляді масиву числових параметрів, що визначають аргумент цільової функції;

генетичні алгоритми – кожне рішення є бітової рядком (хромосомою) певної довжини в популяції фіксованого розміру;

генетичне програмування – тут застосовуються ідеї ГА для еволюції комп'ютерних програм.

В самому загальному вигляді метод ЕО можна описати наступним чином. Метод ЕО являє собою імовірнісний алгоритм, який містить популяцію індивідуумів $P(t) = \left\{x_1^t, x_2^t, \ldots, x_n^t\right\}$ на ітерації *t*. Кожен індивідуум є потенційне

рішення розглянутої проблеми, і в будь-якому з методів ЕО реалізується як деяка (можливо, складна) структура данних *S*. Кожне рішення x_i^t оцінюється для отримання величини придатності (в англійській мові часто використовується термін «fitness»). Потім створюють-ся нова популяція (ітерація *t+1*) шляхом відбору найбільш придатних індивідуумів (етап селекції).

Деякі члени нової популяції зазнають перетворень (етап рекомбінації) за допомогою генетичних операторів для формування нових рішень. Серед операторів можна виділити оператор мутації, який створює новий індивідуум шляхом малих змін вихідного, і оператор схрещування, формує нові індивідууми допомогою комбінування частин кількох вихідних. Після ряду таких генерацій програма сходиться, тобто знаходиться кращий індивідуум, який представляє собою оптимальне рішення.

Очевидно, що багато еволюційні програми (синонім еволюційних обчислень) можуть бути сформульовані для вирішення конкретної проблеми. Такі програми можуть мати різну структуру даних для побудови окремого індивідуума, генетичні оператори для трансформації індивідуумів, методи для створення початкової популяції, параметри обчислень (розмір популяції, ймовірності застосування різних операторів і т. д.).

Слід зазначити, що методи ЕО не гарантують виявлення глобального оптимуму за прийнятний час. Практичний інтерес до них пояснюється тим, що ці методи дозволяють знайти «хороші» вирішення дуже складних завдань за менший час, ніж іншими методами.

Необхідно усвідомити різницю між еволюційним програмуванням і ГА: при першому підході використовується будь-яка структура даних (хромосомне подання), підходить для розв'язання задачі, і будь-яке безліч генетичних операторів; у другій ситуації застосовуються двійкові рядки фіксованої довжини (хромосоми) і два оператора (бінарні мутація і схрещування).

В еволюційних програмах проблема обмежень має інший характер. Тут мова йде, скоріше, про вибір «кращого» хромосомного представлення шуканого рішення разом зі значущими генетичними операторами для задоволення всіх обмежень, що накладаються цією проблемою. Будь-генетичний оператор повинен проходити через деяку характеристичну структуру від «батьків» до «нащадкам», тому структура подання відіграє важливу роль у визначенні генетичних операторів. Крім того, різні структури уявлень мають різні

характеристики придатності для обмежень, що ще більше ускладнює завдання. Ці два компоненти – представлення та оператори – впливають один на одного.

Основні поняття генетичних алгоритмів

Ідея використання ГА полягає в тому, щоб виконувати те, що робить природа. Розглянемо один приклад з популяцією кроликів, наявної в не-який момент часу t. Частина з них більш швидка і моторна, ніж інша. Такі швидкі кролики мають менше шансів бути з'їденими лисицями, і тому більшість з них виживає, щоб робити те, що кролики добре роблять: створювати ще більше кроликів. Вижила популяція кроликів дає нове потомство, яке має гарну суміш генетичного матеріалу: деякі «повільні» кролики схрещуються з «швидкими», швидкі – з швидкими, моторні – з незграбними і т. д. Результуюче потомство буде в середньому більш швидким і спритним, ніж у вихідній популяції, тому що тільки такі особини мають шанс уникнути загибелі.

Генетичні алгоритми виконують крок за кроком процедуру, яка тєм-але пов'язана з прикладом з життя кроликів. Отже, одним з можливих визначень може служити наступне: ГА – це алгоритми, методи пошуку моделюють приватні природні явища: генетична спадковість і боротьбу за виживання. Перш ніж перейти до структурі ГА, коротко висвітлимо історію генетики. Генетика – це вивчення спадковості. Генетики вивчають взаємодія генів та їх передачу від батьків до потомства. В живих організмах клітини містять важливі групи спеціальної матерії, що містить спадкову інформацію. Такі ниткоподібні структури називаються хромосомами. Гени – спеціальні елементи, які переносяться хромосомами. Основний процес кодування інформації в межах генів і хромосом досить простий, але можливі результати майже безмежні за кількістю. Генетична структура кожного організму називається генотипом, який визначає і обмежує багато аспектів розвитку і відбору. Відгук організму на зовнішні умови принципово визначається його **генотипом** . Властивості організму, які виникають з цього зіткнення з навколишнього середовища, визначають його **фенотіп**.

Основний принцип природного відбору як головного принципу еволюції був сформульований Ч. Дарвіном задовго до відкриття генетичних механізмів. Ігноруючи основні спадкові принципи, Ч. Дарвін висунув гіпотезу злиття, вважаючи, що батьківські якості змішуються разом, подібно до рідин, на слідчому організмі. Його селекційна теорія викликала серйозну критику з боку

інших науковців, проте в той час ніякої більш прийнятною генетичної теорії ще не було.

Значний крок вперед у період 1856-1863 рр. в розробці генетики зробив католицький священик з Чехії Р. Мендель (G. Mendel), який відкрив основні принципи передачі спадкових факторів від батьків до потомства. В результаті багаторічних досліджень він відкрив закони домінування ознак в першому поколінні незалежно від їх розподілу в наступних поколіннях і їх кількісного співвідношення.

Генетика була повністю розроблена американським вченим Т. Морганом (T. Morgan) в період 1909-1914-х рр. і його колегами, які експериментально довели, що хромосоми є головними носіями спадкової інформації, а гени, які представляють спадкові фактори, розташовані в лінію на хромосомах. Т. Моргану вдалося спостерігати в мікроскоп процес обміну генетичного матеріалу між різними хромосомами: дві хромосоми зближувалися і схрещувалися, обмінюючись фрагментами. Він представляв гени впорядкованими по довжині хромосом, як намистинки намисто. Експериментальні дані привели його до чудової ідеї про створення генетичних карт. Очевидно, що чим далі знаходяться два гена один від одного, тим більше ймовірність обриву їх зв'язує нитки і отримання нових поєднань генів.

На початку 1960-х рр. деякі біологи почали експериментувати з комп'ютерной імітацією генетичних систем. Зазначимо, що сучасна теорія ГА, застосовна для вирішення оптимізаційних завдань, пов'язана з ім'ям Д. Холланда.

Генетичні алгоритми використовують словник, запозичений з природною генетики. Тут необхідно відзначити все ще не усталену термінологію в цій області, і тому в різних виданнях можна зустріти різну інтерпретацію однакових понять. Нижче будемо дотримуватися наступної термінології:

бітова рядок 0101...101 (хромосома, індивід, індивідуум) – визначає точку простору пошуку та представляє потенційне рішення задачі;

гени – елементи, з яких складається хромосома (синонім генів – ознаки, букви);

популяція – набір хромосом (рядків); в класі ГА розмір (величина) популяції приймається фіксованою величиною;

покоління (генерація) – нове покоління після кожного кроку роботи ГА;

батьки – хромосоми, з яких шляхом схрещування і відбору форми-руются хромосоми "«нащадки»;

якість хромосоми – функція, що оцінює придатність даної рядки порівняно з іншими (в англ. мовою – fitness). генетичні оператори (відбір, схрещування, мутація) – перетворення, яким піддаються хромосоми в процесі еволюції і боротьби за виживання.

Підкреслимо, що кожна рядок (хромосома) представляє потенційне рішення задачі. Еволюційний процес через популяцію хромосом відповідає пошуку по простору потенційних рішень.

Популяція здійснює імітаційну еволюцію:

при кожній генерації щодо «хороші» рішення відтворюються, в той час як «погані» – помирають. Для відмінності між поганими і хорошими рішеннями використовується оцінна функція, визначальна якість кожного рядка.

Для того щоб виявити відмінність ГА при пошуку оптимальних рішень від традиційних способів вирішення оптимізаційних задач, коротко вкажемо принцип дії двох загальноприйнятих методів: **підйому на пагорб** і **імітаційний відпал**.

Метод підйому на пагорб (hillclimbing method) використовує ітераційні поліпшує підхід. Метод застосуємо до єдиної (поточної) точки в просторі пошуків. Під час однієї ітерації наступна точка вибирається з умови сусідства з поточною точкою. У випадку, якщо нова точка забезпечує кращу величину цільової функції, то вона стає поточною. В іншому випадку вибирається інша сусідня точка, яка порівнюється з поточною. Метод пошуку оптимуму завершується, якщо подальшого поліпшення в поведінці цільової функції не настає. Цей метод забезпечує лише локальний оптимум, а тривалість пошуку залежить від вибору началь-ної точки. Крім того, тут відсутня інформація про відносну помилку (по відношенню до глобального оптимуму) знайденого рішення. Для збільшення шансів на успіх метод зазвичай виконується для значного числа початкових точок.

Імітаційний відпал (simulated annealing) заснований на ідеї, заимство-ванни з статистичної фізики. Цей метод описує поведінку матеріального тіла при твердінні із застосуванням процедури відпалу (керованого охолодження) при температурі, послідовно уменьшаемой до нуля. У реальних процесах кристалізації твердих тіл температура знижується ступінчастим чином. На кожному рівні вона якийсь час підтримується постійною, що необхідно для забезпечення термічної рівноваги. Протягом усього періоду часу, коли температура залишається вище абсолютного нуля, вона може як підвищуватися, так і знижуватися. За рахунок утримання температури процесу поблизу від

значення, відповідного рівню термічної рівноваги, вдається обходити пастки локальних максимумів. Метод імітації відпалу являє собою алгоритмічний аналог фізичного процесу керованого охолодження. В даний час даний метод вважається одним з небагатьох алгоритмів, що дозволяють практично знаходити глобальний мінімум функції декількох змінних.

В описі алгоритму в якості назви параметра, що впливає на ймовірність збільшення значення цільової функції, використовується термін «температура», хоча з формальної точки зору наведена модель оптимізації є тільки математичної аналогією процесу відпалу. **Алгоритм імітації відпалу** виглядає концептуально нескладним і логічно обґрунтованим. Насправді доводиться вирішувати багато фундаментальних проблем, які впливають на його практичну придатність. Основний слід назвати проблему тривалості імітації. Для підвищення ймовірності досягнення глобального мінімуму тривалість відпалу (подається кількістю циклів, повторюваних при одному і тому ж значенні температури) повинна бути досить великий, а коефіцієнт зменшення температури – низьким. Це збільшує тривалість процесу моделювання, що може дискредитувати його з позицій практичної доцільності.

Метод імітаційного відпалу виключає більшість недоліків методу підйому на пагорб: рішення не залежить від початкової точки і зазвичай є близькою до оптимальної точки. Це досягається введенням ймовірності p заміни поточної точки новою точкою $p = 1$, якщо нова точка забезпечує краще значення цільової функції; однак $p > 0$ у протилежній ситуації. В останньому випадку ймовірність p залежить від значень цільової функції для поточної і нової точок та керуючого параметра температури. Під час роботи методу температура знижується сходами, і алгоритм завершується при деякому малому значенні температури.

Відмінності ГА від традиційних методів пошуку полягають у наступному:

– для пошуку оптимуму використовують кілька точок одночасно, а не переходять від точки до точки, як це робиться у традиційних методах, що дозволяє уникнути небезпеки попадання в **локальний екстремум** цільової функції, якщо вона не є унімодальной, тобто має кілька таких екстремумів. Використання декількох точок одночасно значно знижує таку можливість;

– в процесі роботи ГА не вимагають ніякої додаткової інформації, що збільшує швидкість роботи алгоритму (єдина інформація: область допустимих значень і функція придатності в певних точках);

– використовують і детерміновані правила для переходу від одних до інших точок, і імовірнісні правила для породження нових точок аналізу;

– працюють з кодами, в яких представлений набір параметрів, що залежать від аргументів цільової функції. Інтерпретація цих кодів відбувається тільки перед початком роботи алгоритму і після завершення її роботи для отримання результату. У процесі роботи маніпуляції з кодами відбуваються абсолютно незалежно від їх інтерпретації, код розглядається просто як бітова рядок.

Структура ГА така ж, як і в будь еволюційної програмі. Під час ітерації t ГА містить популяцію потенційних рішень (хромосом, векторів) $P(t) = \left\{x_1^t,\ldots,x_n^t\right\}$. Кожне рішення x_i^t оцінюється для того, щоб отримати деяку міру її придатності (fitness). Потім на ітерації $t+1$ формується нова популяція шляхом відбору найбільш придатних індивідуумів. Деякі члени цієї нової популяції піддаються репродукції за допомогою генетичних операторів: схрещування та мутації для утворення нових рішень. Схрещування поєднує ознаки двох батьківських хромосом для утворення двох нащадків обміном відповідних сегментів батьків. Наприклад, якщо батьки відображаються пятимерными векторами (a_1, b_1, c_1, d_1, e_1) та (a_2, b_2, c_2, d_2, e_2), то, вибравши точку схрещування після другого гена, отримаємо такі нащадки: (a_1, b_1, c_2, d_2, e_2) та (a_2, b_2, c_1, d_1, e_1).

Ясно, що оператор схрещування являє собою обмін інформацією між різними потенційними рішеннями. Мутація довільно змінює один або більше генів обраної хромосоми випадкової заміною значення, що дорівнює одиниці, на нуль або навпаки. Очевидно, що оператор мутації вводить деяку варіативність у популяції хромосом. ГА для вирішення будь-якої проблеми повинен містити, як правило, такі компоненти:

– генетичне подання потенційних рішень задачі;
– спосіб створення початкової популяції потенційних рішень;
– оцінну функцію, яка відіграє роль оточення і ранжує рішення щодо ступеня їх придатності;

– генетичні оператори, змінюють генетичний склад потомства;
– значення параметрів ГА (ймовірність схрещування і мутації, розмір популяції, кількість поколінь та ін).

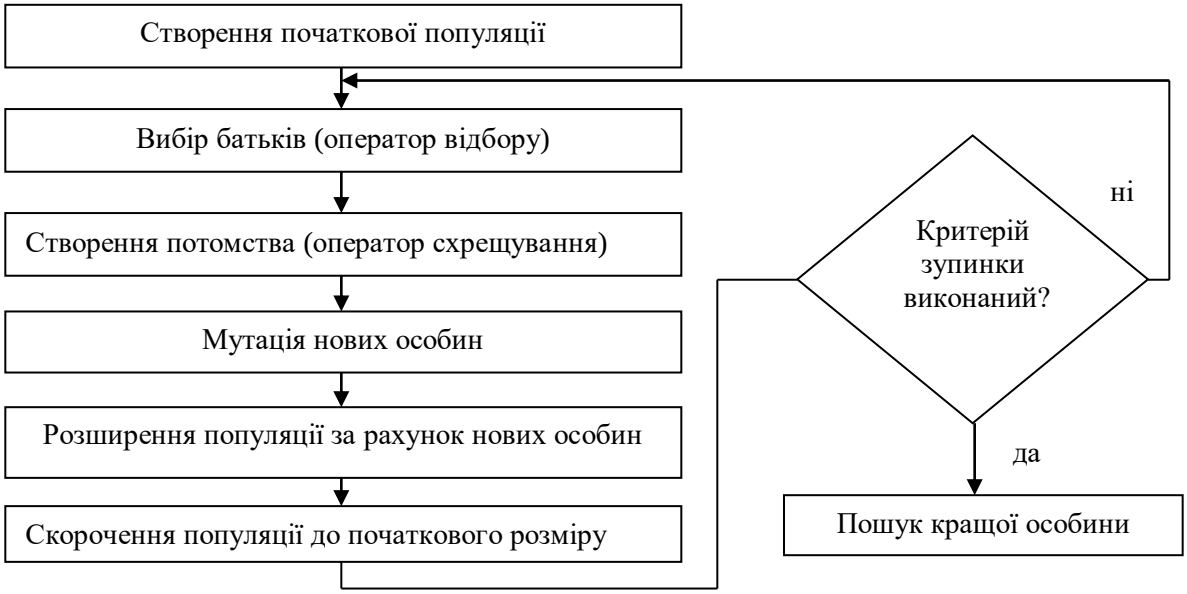


Рис.59. Схема роботи генетичного алгоритму

В якості критеріїв зупинки виконання алгоритму можуть використовуватися такі:

– сформовано задане число поколінь;

– популяція досягла заданої якості;

– досягнутий певний рівень збіжності.

Зазначимо, що останні два критерії пов'язані з заданою величиною придатності популяції або подібністю рядків у популяції. Процес роботи ГА наведено на рис. 59.

Для ілюстрації основних понять і принципу дії ГА наведемо простий приклад на пошук максимуму.

Приклад 4.1. Знайти максимум функції $f(x) = x^2$ у діапазоні $0 < x < 31$. Тут в якості функції придатності виступає сама функція. Чим більше її значення, тим краще придатність хромосоми. Встановимо розмір популяції, рівний чотирьом рядкам. Початкова популяція і оцінка придатності наведена в табл. 11.

Таблица 11. **Початкова популяція і оцінка придатності**

№ рядка	**Початкова популяція**	***x***	***f(x)***	**Відносна придатність, %**
1	01101	13	169	14,4
2	11000	24	576	**49,2**
3	01000	8	64	5,5
4	10011	19	361	30,9
			1170	100

Так як функція придатності другий рядки краща, відбираємо дві копії другого рядка і залишаємо першу і четверту рядки в батьківському пулі. Відбір партнерів виробляємо випадковим чином: партнером першого рядка служить друга, партнером четвертої – теж друга. Положення точок схрещування також випадково і вибирається таким чином: для пари з першої і другої рядків точка схрещування – після четвертого біта; для пари з другої– четвертої рядків – після другого біта (табл. 12).

Таблиця 12. **Батьківський пул і схрещування**

№ рядка	**Батьківський пул**	**Парна рядок**	**До схрещування**	**Після схрещування**
1	01101	2	0110[1]	01100
2	11000	1	1100[0]	11001
3	11000	4	11[000]	11011
4	10011	2	10[011]	10000

Друге покоління без мутації наведено в табл. 13.

Таблиця 13. **Друге покоління**

Рядок	***x***	***f(x)***	**Відносна придатність, %**
01100	12	144	8,2
11001	25	625	35,6
11011	27	729	41,2
10000	16	256	14,7
		1754	100%

З табл. 13 видно, що третій рядок є кращою у другому поколінні і значення x = 27 досить близько до отыскиваемому максимуму. Очевидно, що через кілька кроків оптимальне рішення буде знайдено навіть без використання оператора мутації.

Лекция 24. Кодування в генетичних алгоритмах

З біології відомо, що будь-який організм може бути представлений своїм фенотипом, який фактично визначає, чим є об'єкт в реальному світі (у зовнішньому середовищі). Генотип містить всю інформацію про об'єкт на рівні хромосомного набору. Для вирішення задачі необхідно представити кожну ознаку об'єкта у формі, придатній для використання в ГА. Кодування рішення задачі в хромосомі є ключовою проблемою застосування ГА. Проблема досліджена з різних сторін, наприклад, за характером відображення простору генотипів в простір фенотипів, коли окремі особини декодуються в рішення, та за властивостями перетворення, коли хромосоми регулюються за допомогою операторів.

У класичній роботі Д. Холланда кодування виконується з використанням бінарних рядків. Таке кодування для задач оптимізації має кілька недоліків внаслідок існування хеммингова зсуву для пари закодованих значень, що мають велике *хеммингово відстань* (число позицій, в яких відповідні символи двох слів однакової довжини різні), в той час, як ці величини належать до точок з мінімальною відстанню в фенотипическом просторі. Наприклад, пара 01111111111 і 10000000000 належить сусідніх точках у фенотипическом просторі (точки з мінімальним *евклідовим відстанню* – відстань між двома точками евклідового простору, обчислюване за теоремою Піфагора), але мають максимальну хеммингово відстань в генотипическом просторі. Для подолання хеммингова зсуву всі біти повинні змінюватися одночасно, але ймовірність такої події дуже мала.

Для багатьох задач, що зустрічаються в сучасних проблемах, досить важко уявити їх вирішення за допомогою бінарного кодування. Протягом останніх десяти років були розроблені різні методи кодування для забезпечення ефективного виконання ГА, які можуть бути розділені на наступні класи [4]:

– ***бінарне кодування***;
– ***кодування дійсними числами***;
– ***цілочисельне кодування***;

– ***кодування загальної структури даних***.

Більш докладно розглянемо бінарне кодування, яке використовується в класичному підході Д. Холланда. Нехай вектор допустимого рішення $x \in D$, де D – область пошуку рішень. Кожен компонент вектора $\bar{x} = (x_1, x_2, ..., x_n)$ можна закодувати за допомогою цілого ненегативного числа $\beta_i \in [0, K_i], i = 1, n; (K_i -$ число можливих дискретних значень і-ї змінної).

Введемо двійковий алфавіт $B_2 = \{0;1\}$. Для подання цілочисельного вектора $\beta = (\beta_1, ..., \beta_n)$ в алфавіті B_2 необхідно визначити максимальне число двійкових символів θ, що достатньо для відображення в двійковому коді будь-якого значення β_i з області її допустимих значень $[0, K_i]$. Неважко бачити, що параметр θ повинен задовольняти нерівності

$$K < 2^{\theta}, \tag{1}$$

де $K = \max K_i, \quad 1 \le i \le n.$

Тоді для $\beta_i \, (0 \le \beta_i \le 2^{\theta})$ можна записати

$$\beta_i = \sum_{i=1}^{n} \alpha_i \cdot 2^{\theta - 1}, \tag{2}$$

де α_i – двійкове число (0 або 1); θ – довжина двійкового слова, що кодує ціле число β_i.

Приклад 2. Представимо у вигляді хромосомної рядка кількість $\beta_i = 19$.

З урахуванням формул (4.1) і (4.2) маємо: $\theta = 5,$ так як $2^{\theta} = 2^5 = 32.$ За формулою (4.2) отримаємо:

$$19 = 1 \cdot 2^{5-1} + 0 \cdot 2^{5-2} + 0 \cdot 2^{5-3} + 1 \cdot 2^{5-4} + 1 \cdot 2^{5-5} = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

Рядок буде мати наступний вигляд:

1 0 0 1 1

В якості гена, тобто одиниці спадкового матеріалу, відповідального за формування ознак особини, приймемо бінарну комбінацію $e_0(\beta_i)$, яка визначає фіксоване значення цілочисельного коду b_i керованої змінної x_i у звичайному двоїчному коді. В цьому випадку $e_0(\beta_i)$ має вигляд:

α_1	α_2	...	α_θ

θ

Одна особина α_k^t буде характеризуватися n генами, кожен з яких відповідає за формування цілочисельного коду відповідної керованої змінної. Тоді хромосому можна визначити в наступному вигляді *E(X)*:

$\alpha_i^1 \ldots \alpha_\theta^1$ $e_\theta(\beta_1)$ *ген* 1	$\alpha_i^2 \ldots \alpha_\theta^2$ $e_\theta(\beta_2)$ *ген* 2	... ...	$\alpha_i^n \ldots \alpha_\theta^n$ $e_\theta(\beta_n)$ *ген n*
Локус 1	Локус 2	...	Локус n
ХРОМОСОМА			

Розташування певного гена в хромосомі називається локусом, а альтернативні форми одного і того ж гена, розташовані в одних і тих же локусах, називаються алелями.

Хромосома, що містить в своїх локусах конкретні значення алелей, визначає генотип (генетичний код) $E(\alpha_k^t)$, який містить всю спадкову генетичну інформацію про особини α_k^t, одержувану від предків і потім передається нащадкам. Кінцеве безліч всіх допустимих генотипів утворює генофонд.

Таким чином, у реалізації ГА хромосома являє собою бітову рядок фіксованої довжини. При цьому кожній ділянці рядка відповідає ген. Довжина гена усередині рядка може бути однаковою або різною. Наприклад, для об'єкта з п'яти ознак, кожен з яких закодований геном довжиною в чотири елемента, загальна довжина хромосоми складе $5 \times 4 = 20$ бітів:

0010 1010 1001 0100 1101

При використанні N бітів для бінарної рядка (хромосоми) перетворення від двійкового коду рядка до десятичному значення здійснюється за формулою:

$$x_i = a_i + decimal(100...010_2) \cdot \frac{b_i - a_i}{2^N - 1}$$

де a_i, b_i – нижня і верхня межі зміни і-го ознаки; $decimal(100...010_2)$ – десяткове значення бінарної рядка.

Кодування дійсними числами є кращим при вирішенні задач оптимізації функцій. Так як топологічна структура простору генотипів для цього виду кодування ідентична структурі у просторі фенотипів, то легко сформувати ефективні генетичні оператори запозиченням корисних прийомів у традиційних методів.

Цілочисельне кодування краще всього підходить для комбінаторних оптимізаційних задач.

У відповідності з кодуванням загальної структури даних методи кодування можуть бути розділені на два види:

– одновимірний;

– багатовимірний.

В більшості випадків використовується одномірне кодування, однак багато проблем потребують вирішення у багатовимірному просторі, тому природно застосовувати метод багатовимірного кодування в таких ситуаціях.

Генетичні алгоритми виконуються на двох типах просторів: кодування і рішення, іншими словами, просторах генотипу та фенотипу. Генетичні оператори працюють у просторі генотипів, а оцінка і відбір відбуваються в фенотипическом просторі. Природний відбір – це зв'язок між хромосомами і поведінкою декодованих рішень.

Генетичні оператори

Визначено три основних види генетичних операторів:

– *селекція*;

– *схрещування*;

– *мутація*.

Перед тим як безпосередньо перейти до вивчення цих операторів, зупинимося докладніше на концепції пошуку та його застосування в ГА.

Пошук – один з універсальних методів вирішення завдань, у яких не можна заздалегідь визначити послідовність кроків, що приводять до бажаного результату. Зазвичай розглядаються два види пошуку:

– *випадковий*;

– *локальний*.

Випадковий пошук досліджує рішення цілком і здатний уникнути попадання в локальний оптимум. Локальний пошук використовує найкраще рішення і здатний піднятися вгору до локального оптимуму.

Ідеальний пошук повинен поєднувати обидва типи пошуку одночасно. Але практично такий пошук неможливо здійснити традиційними методами. ГА є класом общецелевых пошукових методів, що комбінують елементи спрямованого і стохастичного типів пошуку, що призводить до хорошого балансу між дослідженням і використанням пошукового простору.

У традиційних ГА оператор схрещування застосовується як основний, і поведінка генетичної системи в значній мірі залежить від нього. Оператор мутації, який здійснює спонтанні випадкові зміни в різних хромосомах, частково виконує роль фону. По суті, генетичні оператори проводять випадковий пошук і не можуть гарантувати появи покращеного потомства. Було виконано багато емпіричних досліджень порівняння мутації та схрещування, в результаті чого з'ясувалося, що іноді мутація грає більш важливу роль, ніж схрещування.

Є дві гіпотези пояснення того, як ГА використовують розподілену інформацію для формування хороших рішень:

– гіпотеза будівельних блоків;

– гіпотеза зміни керованої збіжності.

Згідно з першою гіпотезою схрещування комбінує ознаки від двох батьків для виробництва потомства. Іноді схрещування використовує кращі ознаки від обох батьків, формуючи дуже гарне потомство. Так як придатність особини часто залежить від складної взаємодії простих ознак, то важливо, щоб оператор був в змозі передати нащадкам ті набори ознак, які вносили вклад в оцінку придатності батьків.

Друга гіпотеза припускає використання збіжності популяції для обмеження пошуку. При збіжності популяції варіація придатності стає більш помітною. У той час як перша гіпотеза посилює комбінацію ознак, які вижили батьківської популяції, друга гіпотеза посилює випадкову вибірку з розподілу, який є функцією поточної популяції.

Селекція

Основний принцип генетичних алгоритмів, по суті, являє собою дарвінівський природний відбір. Селекція забезпечує рушійну силу ГА. Чим

більше сила, тим швидше може завершитися генетичний пошук; при меншій величині цієї сили еволюційний процес буде повільніше, ніж необхідно.

Селекція направляє генетичний пошук в «обіцяють» райони пошукового простору. Основний компонент ГА – це метод, використовуваний для переходу від однієї генерації до наступної. Існує багато можливих варіацій щодо вибору потенційних батьків і способи їх комбінування для відтворення потомства. Селекція може змінюватися від дуже трудомістких підходів, які потребують значних зусиль, до дуже легких схем. Протягом останніх 20 років були запропоновані багато методи, серед яких найбільше поширення одержали наступні [4,6]:

Селекція за допомогою пропорційної рулетки. Метод рулетки – найвідоміший спосіб відбору. Основна ідея цього підходу полягає у визначенні ймовірності відбору (ймовірності виживання) для кожної хромосоми пропорційно величині її придатності. Процес відбору заснований на обертанні колеса рулетки стільки разів, який розмір популяції, щоразу обираючи одну хромосому для нової популяції. Колесо служить методом відбору як стохастична вибіркова процедура.

В якості дуже спрощеного прикладу розглянемо популяцію з 10 індивідуумів. Нехай ця популяція містить шість ідентичних хромосом, які позначимо як тип А і чотири ідентичних хромосоми типу В. Припустимо, що тип А має величину придатності, що дорівнює 12, а тип В – 7. Загальна придатність складе величину:

$$6 \times 12 + 4 \times 7 = 100.$$

Тип А займає 72% загальної придатності, а тип В – 28%. При таких вхідних даних «конструюємо» рулетку, 72% площі якої позначимо міткою А, а 28% – міткою В. Для вибору наступної популяції даємо рулетці тільки 10 обертів (такий розмір популяції) і створюємо нову комбінацію з типів А і В. У створеної нової генерації буде міститися, в середньому, близько семи (заснованому на 72%) членів типу А і трьох (заснованому на 28%) членів типу В. Оскільки обертання колеса рулетки і момент його зупинки випадкові (в комп'ютерній реалізації використовується випадковий цифровий генератор), то дійсне поділ між А і В може бути і 0/10, 10 /0, однак у середньому цей метод дає високі значення придатності.

Стохастична залишкова селекція. Розглянемо, як і вище, десятичленную популяцію типів А і В. Очікуване число копій типу А є 7,2, а для типу В – це число складе 2,8. У цьому методі селекції спочатку

зосередимося на цілих частинах очікуваних величин: 7 і 2. Сім копій типу А розміщуються в наступної генерації так само, як і дві копії типу В. Для визначення ідентичності 10-го члена залишки використовуються для «зважування» колеса рулетки.
Тут В отримує 80% всієї ваги, а тип А – тільки 20%. Цей метод буде давати внесок в нову генерацію в співвідношенні 7/3 або 8/2 між типами А і В.

Стохастична універсальна селекція. Цей метод – ще одна ва-риация попередніх двох, але вимагає трохи більшого уяви для його реалізації. Уявімо те ж саме колесо рулетки, 72% площі якій зазначено як А, а 28% – як В. Доповнимо це колесо зовнішнім кільцем з 10 рівних інтервалів (маркерів). Колесо рулетки робить один оборот. Для визначення складу наступної генерації розглядаються 10 маркерів: ті, які знаходяться навпроти площі А, позначаються як тип А; те ж саме відноситься і до В. Цей метод буде давати результати, схожі на попередній випадок: поділ в цьому випадку складе або 7/3, або 8/2.

Турнірна селекція. Турнірна селекція являє собою приклад процедури, що поєднує одночасно випадковий і детермінований підхід. Турніри схожі на маленькі «бої» між членами популяції, щоб побачити, хто візьме участь у наступному поколінні. Тут випадково визначається безліч хромосом, з якого відбираються найкращі особини для репродукції. Число хромосом у цій множині називається розміром турніру, який найчастіше приймається рівним 2 (бінарний турнір). Після отримання пари хромосоми з більш високою оцінкою придатності впроваджуються в нову популяцію. Процес продовжується до тих пір, поки не заповниться вся популяція.

Ранжированная селекція. Ця процедура починається з ранжирування популяції за величиною придатності. Далі вводиться функція присвоювання, яка дає кожній хромосомі ймовірність включення в наступну генерацію. Хромосоми з більш високим рангом мають велику ймовірність включення. Функція присвоювання може бути лінійною або нелінійною (частіше – остання). Колесо рулетки будується зі слотами, що визначаються функцією присвоювання. Наступна генерація n"розмірної популяції визначається після n обертів колеса. Ця процедура просуває селекцію до членів популяції, що володіють кращими характеристиками.

З перерахованих методів селекції найбільше поширення отримав метод пропорційної рулетки, однак потрібно вказати деякі недоліки процедури пропорційного відбору. У цій схемі на ранніх поколіннях може проявитися тенденція домінування «дуже хороших» особин у процесі відбору; на більш

пізніх генераціях конкуренція серед таких хромосом стає менш сильною, і більшою мірою домінує випадковий пошук.

Крім зазначених способів селекції, існують так звані елітні методи, які гарантують, що в процесі селекції будуть зберігатися кращі (у сенсі придатності) члени популяції.

Найбільш часто використовується процедура збереження однієї найкращої хромосоми, якщо вона не пройшла, як інші, через селекцію, схрещування і мутацію. Елітний метод може бути застосований у будь-яку із зазначених вище схем селекції.

Лекция 25. Схрещування

Схрещування – це крок, який реально підсилює генетичні алгоритми. Воно дозволяє розширити пошук у різних напрямках, оцінюючи привабливі рішення і дозволяючи двом рядкам «злучатися». Такий підхід може призвести до спадщини, яка виявиться більш привабливим, ніж батьки.

Нехай популяція $P^t = (a_1^t,...,a_v^t)$ являє собою репродукційну групу, тобто сукупність *v* особин, будь-які дві з яких a_k^t, $a_l^t \in P^t, k \neq 1$ можуть розмножуватися, виступаючи в ролі батьків (a_k^t – мати; a_i^t – батько). Тут під розмноженням розуміється властивість особин відтворювати одного або декількох собі подібних безпосередніх нащадків $b_i^t, i \geq 1$ і забезпечувати у них безперервність і спадкову спадкоємність якісних ознак батьків. Таким чином, цей чинник еволюційного розвитку популяції призводить до отримання нової генетичної інформації, яка містить різні комбінації алельних форм генів батьківських генотипів.

Розглянемо механізм розмноження двох батьківських особин a_k^t, $a_l^t \in P$ шляхом сигнамии (запліднення) їх репродуктивних клітин: материнської гамети (яйцеклітини) $E(a_k^t)$ і батьківській гамети (сперматозоїда) $E(a_l^t)$. В процесі сигнамии утворюється батьківська зигота (запліднена клітина), здатна розвиватися в нову особину з передачею спадкових ознак (генетичною інформацією) від батьків їх нащадкам. Зигота містить одну пару з двох не відрізняються одна від іншої хромосом, які відбуваються від «батьківських»

гамет: одна – від материнської гамети, а інша – від батьківської гамети. Такі хромосоми називаються гомологічними хромосомами.

В результаті акту сигнамии алелі батьківських гамет можуть помінятися місцями в алельних гени гомологічної хромосоми, що дозволяє розглядати наступні ситуація утворення зигот:

– ген з батьківської хромосоми переходить в материнську хромосому;

– ген з материнської хромосоми переходить в батьківську хромосому;

– відбувається взаємний обмін генами між материнською і батьківською хромосомами;

– батьківська і материнська хромосоми залишаються без зміни.

Таким чином, при утворенні зигот відбувається незалежне і випадкове розбіжність батьківських генів за аллельным генам гомологічних хромосом зиготи незалежно від того, у який із батьківських гамет вони були присутні до запліднення.

Заключним етапом розмноження особин є акт мейозу, під яким розуміється процес утворення гамет з батьківського зиготи шляхом незалежного розходження гомологічних хромосом дочірнім гаметам, відтворюють потомство.

Процес розмноження двох особин повинен задовольняти першому і другому законам спадковості Менделя, які формулюються наступним чином:

– два гени, що визначають той чи інший ознака, не зливаються і не розчиняються один в іншому, але залишаються незалежними одна від одної, розщеплюючись при формуванні гамет;

– батьківські гени, що визначають різні ознаки, які успадковуються незалежно один від одного.

Відповідно до першого закону гени (або відповідні їм ознаки батьків), що мають однакові алелі, зберігають свої значення в потомстві, тобто передаються з імовірністю, рівною 1, нащадку у спадок. Гени батьків, мають різні алелі, передаються нащадку у спадок з імовірністю, що дорівнює 0,5, тобто половина гамет виявляється носієм алелі батька, а інша половина – алелі матері.

У відповідності з другим законом рекомбінація (обмін) генів в акті сигнамии може відбуватися або в якомусь одному аллельном гені, або в декількох алельних гени одночасно, тобто передача алелей від батьків потомству може відбуватися у кожному аллельном гені незалежно один від одного. При цьому може виявитися, що гамети нащадків або збігаються з

батьківськими гаметами, або відрізняються від них в одному або декількох локусах.

Алгоритм схрещування може бути описаний такою послідовністю кроків.

1. З популяції випадково вибираються два потенційних батьків.

2. Виконується жеребкування (комп'ютерна) для визначення – проводити чи не проводити схрещування.

3. При позитивному варіанті відповіді на попередньому кроці випадково вибирається точка розщеплення.

4. Кожна вихідна батьківська рядок перерізується в точці розщеплення.

5. Проводиться обмін генетичним матеріалом ліворуч і праворуч від точки розщеплення і з'єднання отриманих рядків для формування потомства.

Схема схрещування показана на рис.60.

Батьки

010011000	1010

110010101	1100

Нащадки

010011000	1100

110010101	1010

Рис.60. Схема схрещування

Наведена схема називається *одноточечным схрещуванням*, так як є тільки один розріз в кожній хромосомі.

Потрібно зазначити, що процедура схрещування є досить критичною для пошуку найкращих рішень, оскільки хороші з точки зору придатності рядки можуть бути сильно спотворені. У разі, якщо схрещування відбувається досить часто, то пошук – хаотичний і неадекватно отримує користь з попередніх успіхів. Рідкісні (нечасті) схрещування утримують пошук «вузьких областях простору пошуку.

Поряд з одноточечным є *двоточкове схрещування*, яке включає два розрізи і може бути зображено так, як показано на рис. 61.

Батьки

011	001	1

110	111	1

Нащадки

011	111	1

110	001	1

Рис.61. Схема схрещування двухточечного

Для усунення небажаних ситуацій були розроблені схеми багатоточкового схрещування, зокрема рівномірного. Останнім потенційно дозволяє будь-якого образу здійснювати обмін і сформувати потомство. Тут завдання вирішується на побітовою основі (bitby"bit basis) для з'ясування того, як рядки потомства співвідносяться з батьківськими рядками.

В якості прикладу розглянемо дві пятибитовые рядка. Для створення скрещенного образу скористаємося для кожної бітової позиції жеребом (монета), замість орла і решки якого визначимо два результату: «same – такий самий» та «swap "обмін». Результат показаний в табл. 13.

Далі отриманий образ «same"swap"same"swap"same» використовуємо для формування двох нащадків. Для першої бітової позиції образ «same»; батько 1 тут має 0, тому нащадок 1 отримує також 0; батько 2 – 1, отже, нащадок 2 отримує 1. Для другої бітової позиції образ «swap», тому нащадок 1 отримує 1 від батьків 1, а нащадок 2 – 0 від батьків 2. Остаточний результат наведено в останніх двох рядках табл. 6.4.

Таблица 13. **Рівномірний схрещування**

Рядки	Положення бита				
	1	2	3	4	5
Батько 1	0	0	0	1	1
Батько 2	1	1	1	0	0
Схрещений образ	same	swap	same	swap	same
Нащадок 1	0	1	0	0	1
Нащадок 2	1	0	1	1	0

Рівномірний схрещування дає більшу гнучкість при комбінуванні рядків, що є привабливим властивістю при роботі з ГА.
Підводячи підсумок розгляду оператора схрещування, зазначимо, що схрещування підлягає не вся популяція рядків, а тільки її частину, яка визначається ймовірністю схрещування (наприклад, $p_{скр.} = 0,6$ означає, що лише 60% рядків з популяції будуть утворювати батьківські пари.

Лекция 26. Мутація

При схрещуванні генотипи нащадків містять нові сполучення алельних форм генів батьків, що ведуть до нових кількісними ознаками нащадків (ступеня придатності). Проте генетична інформація, що міститься в хромосомному наборі батьків і нащадків, не змінюється, так як в результаті розмноження особин шляхом сигнамии і мейозу частоти алелей залишаються постійними, а змінюються тільки частоти генотипів. Джерелом генетичної мінливості особин є мутація, під якою розуміється зміна якісних ознак особин в результаті появи нових алельних форм в окремих генах або цілком у хромосомі. Тим самим в кожному поколінні мутації постачають у хромосомний набір популяції безліч різних генетичних варіацій, притаманних особистостям, які в подальшому будемо називати мутантами.

Процес зміни змісту генів у хромосомі шляхом мутації називається мутагенезом. Найголовніше полягає в тому, що цей фактор еволюції популяції є джерелом нової генетичної інформації, не міститься раніше в генах батьків та їх нащадків.
Мутації відбуваються незалежно від того, чи приносять вони особини шкоду або користь. Вони не спрямовані на підвищення або зниження ступеня пристосованості особини, а тільки справляють структурні зміни в алельних формах генів, що призводить до зміни кількісних ознак особини. В принципі, комбінація мутацій може привести до виникнення нових форм алелей в деяких генах генотипу мутанта, які забезпечують збільшення його ступеня пристосованості до зовнішньої середовищі.

Найбільш простим видом мутацій є точкова мутація, пов'язана із зміною алелю батьківського гена в одному з битовгенной інформації (0 замінюється на 1 або навпаки), наприклад, у рядку виду 0 1 **1** 0 0 замінюється середній елемент, в результаті чого отримаємо такий рядок: 0 1 **0** 0 0. Крім того, мутація може

бути інверсійної, при якій відбувається зміна всіх компонентів рядка, наприклад, рядок 0 1 1 0 0 заміщається рядком 1 0 0 1 1.

Оператор мутації вводить елемент варіабельності популяцію, так як різко змінює деякий рішення. Основним параметром мутації є її вірогідність, яку задає користувач. Зазвичай її величина досить мала (близько 0,01 і нижче), щоб, з одного боку, розширити область пошуку, а з іншого – не привести до таких змін нащадків, які будуть далекі від прийнятних рішень.

Приклад 3. Розглянемо роботу ГА в задачі мінімізації функції чотирьох змінних

$$J(x_1, x_2, x_3, x_4) = (x_1 - 5)^2 + (x_2 - 6)^2 + (x_3 - 7)^2 + (x_4 - 8)^2,$$

мінімум якій досягається при $x_1 = 5, x_2 = 6, x_3 = 7, x_4 = 8.$ Задамо межі зміни змінних: $1 \le x_i \le 10$ для всіх j =1,2,3,4, і визначимо основні поняття ГА.

Особина, або хромосома P^i – це сукупність значень змінних, званих генами, $P^i = \left(x_1^i, x_2^i, x_3^i, x_4^i\right), i = 0,1,2,...,n$. Наприклад, нехай нульова особина P^0 містить наступні значення генів:

$$P^0 = \left(x_1^0, x_2^0, x_3^0, x_4^0\right) = (1,1,1,1).$$

Значення критерію буде дорівнює

$$J\left(P^0\right) = J(1,1,1,1) = (1-5)^2 + (1-6)^2 + (1-7)^2 + (1-8)^2 = 126.$$

Популяція $\Pi = \left(P^1, P^2, ..., P^n\right)$ містить кілька особин $P^1, P^2, ..., P^n$, кількість яких n обмежено умовою $n \le 5.$

Ген j характеризується величиною x_j і положенням j в особини. Можна випадково вибрати j-ї ген:

$$j = 1 + (m-1) \cdot Rnd, \qquad (3)$$

і випадковим чином змінити його величину, тобто мутувати j-ї ген:

$$x_j = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \cdot Rnd, \qquad (4)$$

де m - номер останнього гена в особини; $x_j^{\min}, x_j^{\max}$ – нижній і верхній гранич-

ли зміни змінної x_j; Rnd– генератор випадкових чисел, виробляє при кожному зверненні до нього число в інтервалі [0,1].

У нашому випадку $m = 4, x_i^{min} = 1, x_j^{max} = 10$ для всіх j =1,2,3,4, то при підстановці цих значень в формулу (4.3) і (4.4) отримаємо:

$$j = 1 + 3 \cdot Rnd \quad (5)$$

і

$$x_j = 1 + 9 \cdot Rnd \quad (6)$$

Для простоти будемо оперувати цілочисельними значеннями j і x_j. ГА складається з операторів створення початкової популяції, відбору, схрещування, мутації і редукції (обчислення значення критерію).

Вихідна популяція $\Pi^0 = (P^1, P^2, P^3)$ складається з трьох особин P^1, P^2, P^3, одержуваних з нульовою особини P^0. Нехай при першому зверненні до формули (4.5) знайдено кількість генів n=2, підлягають мутації. Виконуючи n=2 звернення до формули (5), визначаємо номери j_1=1, j_2=3, також віполняя n=2 звернення до формули (6), визначаємо значення $x_1^1 = 5, x_3^1 = 6$ мутованих генів. Замінюючи гени $x_1^0 = 1, x_3^0 = 1$ в нульовий особини P^0 на відповідні мутовані $x_1^1 = 5, x_3^1 = 6$, отримуємо першу особина:

$$P^1 = (x_1^1, x_2^0, x_3^1, x_4^0) = (5,1,6,1).$$

Аналогічним чином, при $n = 1, j_1 = 4$ і $x_4^2 = 7$ отримуємо другу особина:

$$P^2 = (x_1^0, x_2^0, x_3^0, x_4^2) = (1,1,1,7),$$

а при $n = 3, j_1 = 1, j_2 = 2, j_3 = 4$ і $x_1^3 = 2, x_2^3 = 3, x_4^3 = 4$ – третю особина

$$P^3 = (x_1^3, x_2^3, x_3^0, x_4^3) = (2,3,1,4).$$

Оператор відбору обчислює функцію *J* для кожної особи:

$$J\left(P^1\right)=J(5,1,6,1)=(5-5)^2+(1-6)^2+(6-7)^2+(1-8)^2=75$$
$$J\left(P^2\right)=J(1,1,1,7)=(1-5)^2+(1-6)^2+(1-7)^2+(7-8)^2=78$$
$$J\left(P^3\right)=J(2,3,1,4)=(2-5)^2+(3-6)^2+(1-7)^2+(4-8)^2=70,$$

і вибирає дві особини - батьків P^1 і P^3 з мінімальними значеннями критеріїв $J\left(P^1\right)=75, J\left(P^3\right)=70.$

Оператор схрещування передає нащадкам властивості батьків. Спочатку за формулою (4.5) визначається номер гена k=2, після якого обмінюються місцями підрядки з освітою нащадків – особин P^4і P^5, успадковують властивості батьків

$$\text{Батьки} \quad \begin{cases} P^1=(5,1,6,1), \\ P^3=(2,3,1,4). \end{cases} \qquad k=2 \qquad \text{Нащадки} \begin{cases} P^4=(5,1,1,4), \\ P^5=(2,3,6,1). \end{cases}$$

Оператор мутації при виконанні умови $\text{Rnd}\le\varepsilon,$ де $0<\varepsilon<1,$ за формулою (4.5) визначає номер $\lambda=2$ генів, які переставляються в нащадках - особинах P^4і P^5 з утворенням мутованих нащадків $\overline{P}^4$і $\overline{P}^5$:

$$\text{Нащадки} \begin{cases} P^4=(5,1,1,4), \\ P^3=(2,3,6,1). \end{cases} \Rightarrow \begin{cases} \overline{P}^4=(5,3,1,4), \\ \overline{P}^5=(2,1,6,1). \end{cases} \qquad \lambda=2$$

Оператор редукції обчислює значення критерію для нащадків:

$$J\left(\overline{P}^4\right)=J(5,3,1,4)=(5-5)^2+(3-6)^2+(1-7)^2+(4-8)^2=61$$
$$J\left(\overline{P}^5\right)=J(2,1,6,1)=(2-5)^2+(1-6)^2+(6-7)^2+(1-8)^2=84.$$

З п'яти особин

$$P^1=(5,1,6,1), P^2=(1,1,1,7), P^3=(2,3,1,4), \overline{P}^4=(5,3,1,4), \overline{P}^5=(2,1,6,1)$$

з критеріями

$$J\left(P^1\right)=75, J\left(P^2\right)=78, J(P^3)=70, J(\overline{P}^4)=61, J(\overline{P}^5)=84$$

вибираються три особини $P^1, P^3, \overline{P}^4$, мають мінімальні значення критеріїв і утворюють нову вихідну популяцію

$$\Pi^0 = \left(\mathrm{P}^1, \mathrm{P}^3, \overline{\mathrm{P}}^4\right).$$

Слід зазначити, що значення критеріїв $\mathrm{J}\left(\mathrm{P}^1\right) == 75, \mathrm{J}(\mathrm{P}^3) = 70, \mathrm{J}(\overline{\mathrm{P}}^4) = 61$ для особин $P^1, P^3, \overline{P}^4$ набагато менше, ніж величина критерію $J(P^0) = 126$ для нульової особини P^0. Отже, гени-змінні $\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4$ змінюються в напрямі мінімізації *J*. Перераховані дії складають один повний цикл розрахунку ГА, званої епохою. ГА завершує роботу, якщо досягнута задана кількість епох або не змінюється величина критерію *J*, що свідчить про досягнення мінімуму. В іншому випадку, з новою вихідною популяцією повторюються дії операторів відбору, схрещування, мутації, редукції і т. д.

Лекция 27. Прийоми виконання генетичних алгоритмів

Розглянемо дію ГА на проблемі оптимізації функції багатьох змінних.

Нехай завдання полягає в максимізації функції *k* змінних $\mathrm{f}(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{k})$; при цьому кожна змінна $x_i\left(i = \overline{1,k}\right)$ приймає значення з області $D_i = \left[a_i, b_i\right]$ та $\mathrm{f}(\mathrm{x}_1, \mathrm{x}_2, ..., \mathrm{x}_\mathrm{k}) > 0$ для всіх $x_i \notin D_i$. Встановимо необхідну точність оптимізації функції f: два знаки після коми. Ясно, що кожна область D_i повинна бути розділена на $(b_i - a_i)\cdot 10^2$ рівних відрізків.

Позначимо через m_i найменше число, яке задовольняє нерівності

$$(\mathrm{b_i} - \mathrm{a_i})\cdot 10^2 \le 2^{\mathrm{m_i}} - 1.$$

Тоді кожна змінна x_i кодується як бінарна рядок довжиною m_i, що задовольняє заданій точності.

Кожна хромосома (потенційне рішення) представляється бінарної рядком завдовжки $\mathrm{m} = \sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{m_i}$. У цьому рядку перші m_1 бітів відображають x_1 з діапазону $[a_1,b_1]$, другі m_2 – з діапазону $[a_2,b_2]$ і т. д.

В результаті хромосома може мати такий вигляд:

$$\underbrace{01010}_{\mathrm{m_1}}\underbrace{1110}_{\mathrm{m_2}}0011\underbrace{1100}_{\mathrm{m_k}}$$

Крім того, задамо розмір популяції M (число хромосом). Далі робота ГА видається цілком очевидною у відповідності з наведеним вище алгоритмом:

– у кожній генерації оцінюється окрема хромосома на предмет її придатності з використанням функції f на декодированном наборі змінних:

– відбирається нова популяція з урахуванням розрахованої придатності;

– з допомогою операторів схрещування і мутації хромосоми комбінуються в нову популяцію.

Після деякого числа генерацій, коли не спостерігається поліпшення популяції, краща хромосома являє оптимальне (можливо глобальне) рішення. Часто алгоритм зупиняють після фіксованого числа ітерацій.

Розглянемо деякі кроки більш докладно.

Селекція. Для процесу селекції служить рулетка з розмірами секторів, пропорційних придатності кожного рядка. Розробка такої рулетки складається з наступних кроків:

– обчислюється придатність $\mu(a_j)$ для кожної хромосоми:

$$a_j, j = \overline{1, M};$$

– знаходиться загальна функція придатності всієї популяції:

$$F = \sum_{j=1}^{M} \mu(a_j);$$

– визначається ймовірність вибору p_j для кожної хромосоми a_j:

$$p_j = \mu(a_j) / F;$$

– обчислюється кумулятивна (накопичена) ймовірність q_j для кожної хромосоми:

$$q_j = \sum_{j=1}^{j^*} p_j$$

Процес селекції заснований на обертанні колеса M раз, і кожен раз відбирається одна хромосома в нову популяцію наступним чином:

– генерується випадкове число *r* з діапазону [0...1];

– якщо $r < q_1$, то вибирається перша хромосома a_1; в іншому випадку відбирається *j*-я хромосома a_j $(2 \le j \le M)$ так, щоб $q_{j-1} < r \le q_j$.

Очевидно, що деякі хромосоми будуть обрані більше, ніж один раз. Кращі хромосоми дають більше копій, середні – залишаються незмінними, погані – помирають. Нові рішення на цьому етапі не створюються.

Схрещування. Задається ймовірність схрещування p_c. Очікуване число хромосом, які піддаються схрещуванню, становить $p_c M$.

Для кожної хромосоми (нової) популяції:

– генерується випадкове число s з діапазону [0...1];

– якщо $r < p_c$, то ця хромосома вибирається для схрещування.

Таким чином відбираються особини для схрещування. Вибір точки схрещування теж випадковий: генерується випадкове число s з діапазону [1...(*m–1*)] (*m* – довжина хромосоми). Це число *s* визначає точку схрещування.

В результаті дві хромосоми $(b_1 b_2 ... b_s b_{s+1} ... b_m)$ і $(c_1 c_2 ... c_s c_{s+1} .. c_m)$ замінюються парою нащадків $(b_1 b_2 ... b_s c_{s+1} .. c_m)$ і $(c_1 c_2 ... c_s b_{s+1} ... b_m)$.

Мутація. Задається ймовірність мутації p_m. Очікуване число змінених біт складе $p_m \cdot m \cdot M$. Кожен біт у всіх хромосомах у всій популяції має рівний шанс зазнати мутації, тобто змінитися з 0 на 1 або навпаки. Це здійснюється наступним чином:

– генерується випадкове число *r* з діапазону [0...1];

– якщо $r < p_m$,то біт змінюється.

Після відбору, схрещування і мутації нова популяція готова для подальшого оцінювання. Отримані оцінки використовуються для побудови нової рулетки з секторами, пропорційними поточним значенням функції придатності. Інша частина еволюції представляє по суті циклічне повторення процесу.

Приклад 4. Скористаємося наведеними поясненнями для вирішення наступного завдання [2]. Знайти максимум функції:

$$f(x_1, x_2) = 21{,}5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2),$$

де $-3{,}0 \le x_1 \le 12{,}1 \qquad 4{,}1 \le x_2 \le 5{,}8.$

Крім того, приймемо розмір популяції M = 20; ймовірність схрещування p_c= 0,25; імовірності мутації p_m = 0,01. Припустимо, що необхідна точність становить чотири цифри після коми для кожної змінної. Тоді діапазон для змінної x_1, якої становить 15,1, повинен бути розділений на 15,1 *10000 рівних відрізків. Це означає, що для першої частини хромосоми потрібно 18 бітів, так

як

$$2^{17} \le 15100 \le 2^{18}.$$

Для другої змінної x_2 з діапазоном рівним 1,7, умова встановленої точності вимагає, щоб весь діапазон був розділений на 1,7*10000 рівних відрізків. Таким чином, для цієї змінної необхідно 15 бітів, оскільки

$$2^{14} \le 17000 \le 2^{15}.$$

Загальна довжина хромосоми (вектор потенційного рішення) складе m = 18+15 = 33 бітів, з яких перші 18 кодують першу змінну, а решту 15 – другу. Розглянемо, наприклад, такий рядок:

(010001001011010000 111110010100010).

Перші 18 бітів визначають таке значення змінної x_1:

$$x_1 = -3{,}0 + \text{decimal}(010001001011010000_2) \cdot \frac{12{,}1 - (-3{,}0)}{2^{18} - 1} =$$

$$= -3{,}0 + 70352 \cdot \frac{15{,}1}{262143} = -3{,}0 + 4{,}05 = 1{,}05$$

Решта 15 бітів, декодированные за аналогією з вищенаведеним рівністю, дають для змінної x_2 значення, що дорівнює 5,75. Таким чином, хромосома (010001001011010000 111110010100010) відповідає $(x_1, x_2) = (1{,}05; 5{,}75)$, що визначає для оптимізованної функції наступне значення:

$$f(x_1, x_2) = f(1{,}05; 5{,}75) = 20{,}25.$$

Створимо початкову популяцію, що складається з 20 рядків, у кожному з яких значення 33 бітів ініціюються випадковим чином. Покладемо, що після ініціювання отримана популяція, наведена в табл. 14.

Таблица 14. **Початкова популяція**

Номер рядка	**Рядок-хромосома**
1	10011010000000111111110100110111111
2	11100010010011011100101010001101O
3	00001000001100100000101011101101
4	10001100010110100111100000111001O
5	00011101100101001101011111100010 5
6	00010100001001010100101011111101 1
7	00100010000011010111101101111101 1
8	10000110000111010001011010110011 1
9	01000000010110001011000000111110 0
10	00000111100011000001101000011101 1
11	01100111111011010110000110111100 0
12	11010001011110110100010101000000 0
13	11101111101000100011000000100011 0
14	01001001100000101010011110010100 1
15	11101110110111000010001111101111 0
16	11001111000001111110000110100101 1
17	01101011111100111101000110111110 1
18	01110100000000111010011111010110 1
19	00010101001111111111000011000110 0
20	10111001011001111001100010111111 0

Тепер необхідно декодувати кожну хромосому і обчислити функцію придатності кожного рядка. Ясно, що в цьому прикладі придатність визначається самою оптимізованної функцією. Після декодування отримаємо результат, показаний в табл. 15.

Таблица 15. **Придатність початкової популяції**

Номер рядка	Функція	Придатність
1	*f*(6,08;5,65)	26,01
2	*f*(10,34;4,38)	7,58
3	*f*(-2,51;4,39)	19,52
4	*f*(5,27;5,59)	17,40
5	*f*(-1,25;4,73)	25,34
6	*f*(-1,81;4,39)	18,10
7	*f*(-0,99;5,68)	16,02
8	*f*(4,91;4,70)	17,95
9	*f*(0,79;5,38)	16,12
10	*f*(-2,55;4,79)	21,27
11	*f*(3,13;4,99)	23,41
12	*f*(9,35;4,23)	15,01
13	*f*(11,13;5,37)	27,31
14	*f*(1,33;5,15)	19,87
15	*f*(11,08;5,05)	30,06
16	*f*(9,21;4,99)	23,86
17	*f*(3,36;4,57)	13,69
18	*f*(3,84;5,15)	15,41
19	*f*(-1,74;5,39)	20,09
20	*f*(7,93;4,75)	13,66

З отриманих даних видно, що друга хромосома володіє найменшою придатністю, а хромосома a_{15} – найбільшою.

Перейдемо до конструювання рулетки, необхідної для процесу селекції. Загальна придатність всієї популяції становить величину

$$F = \sum_{j=1}^{20} \mu(a_j) = 387{,}77.$$

Ймовірності вибору p_j для кожної хромосоми у відповідності з вищезазначеним правилом наведено в табл. 16.

Кумулятивні ймовірності для кожної хромосоми наведено в табл. 17.

Далі необхідно зробити 20 обертів рулетки, щоразу забираючи єдину хромосому для нової популяції. Нехай випадкова послідовність 20 чисел з діапазону [0...1] має вигляд, показаний в табл. 18.

Таблица 16. **Значення ймовірності кожної хромосоми**

Номер рядка	**Ймовірність** p_j	**Номер рядка**	**Ймовірність** p_j
1	0,067	11	0,060
2	0,019	12	0,038
3	0,050	13	0,070
4	0,044	14	0,051
5	0,065	15	0,077
6	0,046	16	0,061
7	0,041	17	0,035
8	0,046	18	0,039
9	0,041	19	0,051
10	0,054	20	0,035

Таблица 17. **Кумулятивні ймовірності кожної хромосоми**

Номер рядка	**Ймовірність** q_j	**Номер рядка**	**Ймовірність** q_j
1	0,067	11	0,538
2	0,086	12	0,577
3	0,137	13	0,647
4	0,181	14	0,698
5	0,247	15	0,776
6	0,293	16	0,837
7	0,335	17	0,873
8	0,381	18	0,912
9	0,423	19	0,964
10	0,478	20	1,000

Таблица 18. **Випадкові числа з діапазону [0,1]**

Розіграні числа				
0,513	0,175	0,308	0,534	0,947
0,171	0,702	0,226	0,494	0,424
0,703	0,389	0,227	0,368	0,983
0,005	0,765	0,646	0,767	0,780

Перше число r_1 більше, ніж q_{10} і менше, ніж q_{11}, тому для нової популяції вибирається хромосома a_{11}; друге число r_2 більше, ніж q_3 і менше, ніж q_4. Отже, другий для нової популяції вибирається рядок a_4 і т. д.

Остаточно нова популяція має вигляд, наведений в табл.19.

Таблиця 19. **Нова популяція хромосом**

Номер рядка	**Рядок-хромосома**	**Старий номер рядка**
1	01100111111011010110000110111 1000	11
2	10001100010110100111100000111 0010	4
3	00100010000011010111101101111 1011	7
4	01100111111011010110000110111 1000	11
5	00010101001111111111000011000 1100	19
6	10001100010110100111100000111 0010	4
7	11101110110111000010001111101 1110	15
8	00011101100101001101011111100 0101	5
9	01100111111011010110000110111 1000	11
10	00001000001100100000101011101 1101	3
11	11101110110111000010001111101 1110	15
12	01000000010110001011000000111 1100	9
13	00010100001001010100101011111 1011	6
14	10000110000111010001011010110 0111	8
15	10111001011001111001100010111 1110	20
16	10011010000000111111101001101 1111	1
17	00000111100011000001101000011 1011	10
18	11101111101000100011000000100 0110	13
19	11101110110111000010001111101 1110	15
20	11001111000001111110000110100 1011	16

Як видно з табл. 19, найгірша в початковій популяції рядок 2 після селекції не потрапила в наступну генерацію, а найкраща в початковій популяції рядок 15 з'явилася в новій популяції три рази. Наступним кроком у проведенні ГА є схрещування, яке застосовується до отриманої нової популяції. Задана ймовірність схрещування становить величину

$p_c = 0{,}25$, тому в середньому має зазнати схрещування 25% вихідних хромосом. Тут робимо наступним чином: для кожної хромосоми в новій популяції генеруємо випадкове число r з діапазону [0...1]; якщо $r < 0{,}25$, то вибираємо дану хромосому для схрещування.

Припустимо, що послідовність випадкових чисел діапазону [0...1] вийшла така, як показано в табл. 20.

З табл. 20 випливає, що для схрещування відбираються хромосоми з номерами 2, 11, 13 і 18, оскільки значення випадкових чисел на цих позиціях менше ніж 0,25.

Таблиця 20. **Випадкові числа з діапазону [0,1]**

Розіграні числа				
0,82	0,15	0,62	0,31	0,34
0,91	0,51	0,40	0,60	0,78
0,03	0,86	0,16	0,67	0,75
0,58	0,38	0,20	0,35	0,82

Зазначимо, що в даному випадку число відібраних хромосом вийшло парних, тому легко скласти батьківські пари. В іншому випадку необхідно додати або прибрати одну хромосому. Склад батьківських пар також випадковий, наприклад, в якості однієї такої пари виберемо рядка a_2, a_{11} і інший – рядки a_{13}, a_{18}. Для кожної з цих двох пар генеруємо випадкове число s з діапазону [1...32] (нагадаємо, що 33 – загальне число бітів в хромосомі), що визначає положення точки схрещування. Для першої пари це число складе 9, а для другої – 20.

Перша пара хромосом

a_2 = 100011000 | 101101001111000001110010

a_{11} = 111011101 | 101110000100011111011110

після схрещування дає таку пару нащадків:

a_2^* = 100011000 | 101110000100011111011110

a_{11}^* = 111011101 | 101101001111000001110010.

Друга пара хромосом

a_{13} = 00010100001001010100 | 1010111111011

a_{18} = 11101111101000100011 | 0000001000110

в результаті схрещування дає таку пару нащадків:

a_{13}^* = 00010100001001010100 | 0000001000110

a_{18} = 11101111101000100011 | 1010111111011.

Після схрещування популяція хромосом приймає вигляд, показаний в табл. 21.

Таблица 21. **Популяція хромосом після схрещування**

Номер рядка	**Рядок-хромосома**
1	01100111111011010110000110111 1000
2*	10001100010111000010001111101 1110
3	00100010000011010111101101111 1011
4	01100111111011010110000110111 1000
5	00010101001111111111000011000 1100
6	10001100010110100111100000111 0010
7	11101110110111000010001111101 1110
8	00011101100101001101011111100 0101
9	01100111111011010110000110111 1000
10	00001000001100100000101011101 1101
11*	11101110110110100111100000111 0010
12	01000000010110001011000000111 1100
13*	00010100001001010100000000100 0110
14	10000110000111010001011010110 0111
15	10111001011001111001100010111 1110
16	10011010000000111111101001101 1111
17	00000111100011000001101000011 1011
18*	11101111101000100011101011111 1011
19	11101110110111000010001111101 1110
20	11001111000001111110000110100 1011

Примітка. Схрещені хромосоми відмічені зірочкою.

Перейдемо тепер до оператора мутації, який виконується на побітовою основі. Задана ймовірність мутації p_m=0,01, тому очікуване число мутируемых біт складе 1% від загального числа бітів в популяції. В останній є 33*20 = 660 бітів. Отже, в середньому число бітів"мутантів складе 6-7 одиниць.

Кожний біт в популяції має рівний шанс зазнати мутації, тому для кожного біта генеруємо випадкове число r з діапазону [0...1]; якщо $r < 0{,}01$, то даний біт мутируется. В цілому, необхідно розіграти 660 випадкових чисел, з яких в даному випадку тільки п'ять задовольняють необхідним умовам. Положення біта і відповідне значення випадкового числа наведено в табл. 22.

Таблица 22. **Позиція мутируемого біта в популяції**

Позиція біта	**Випадкове число**
112	0,00021
349	0,00994
418	0,00880
429	0,00542
602	0,00283

Для визначення положення мутованої біта в рядках популяції скористаємося табл. 23.

З табл. 22 видно, що чотири хромосоми піддалися мутації, причому одна з рядків з номером 13 двічі змінила значення бітів.

Таблица 23. **Положення мутованої біта в рядках популяції**

Позиція біта	**Номер хромосоми**	**Номер біта в хромосомі**
112	4	13
349	11	19
418	13	22
429	13	33
602	19	8

Остаточна популяція після операторів схрещування і мутації наведена в табл. 24.

Таблица 24. **Популяція після схрещування і мутації**

Номер рядка	**Хромосома**	**Функція придатності**
1	01100111111011010110000110111100**0**	*f*(3,13;4,99)= 23,41
2*	10001100010111000010001111101111**0**	*f*(5,27;5,05)= 18,20
3	00100010000011010111101101111101**1**	*f*(-0,99;5,68)= 16,02
4**	01100111111**0**1010110000110111100**0**	*f*(3,13;4,99)= 23,41
5	00010101001111111111000011000110**0**	*f*(-1,74;5,39)= 20,09
6	10001100010110100111100000111001**0**	*f*(5,27;5,59)= 17,40
7	11101110110111000010001111101111**0**	*f*(1,08;5,05)= 30,06
8	00011101100101001101011111100010**1**	*f*(-1,25;4,73)= 25,34
9	01100111111011010110000110111100**0**	*f*(3,13;4,99)= 23,41
10	00001000001100100000101011101110**1**	*f*(-2,51;4,39)= 19,52
11**	11101110110110100**0**11000001110**0**10	*f*(11,08;4,74)= 33,35
12	01000000010110001011000000111110**0**	*f*(0,79;5,38)= 16,12
13**	00010100001001010100**1**000010001**1**1	*f*(-1,81;4,20)= 22,69
14	10000110000111010001011010110011**1**	*f*(4,91;4,70)= 17,95
15	10111001011001111001100010111111**0**	*f*(7,93;4,75)= 13,66
16	10011010000000111111101001101111**1**	*f*(6,08;5,65)= 26,01
17	00000111100011000001101000011101**1**	*f*(-2,55;4,73)= 21,27
18*	11101111101000100011101011111101**1**	*f*(11,13;5,66)= 27,59
19**	1110111**1**11011100001000111110111**0**	*f*(11,05;5,05)= 27,60
20	11001111000001111110000110100101**1**	*f*(9,21;4,99)= 23,86

В цій же таблиці в останньому стовпці наведені значення функції придатності, отримані для вихідної популяції після селекції, схрещування і мутації. Краща рядок має значення функції придатності, рівне 33,35, що перевищує найбільшу величину вихідної популяції. Крім того, і загальна придатність, рівна 447,04, набагато перевищує аналогічну величину в початку роботи ГА. Таким чином, за один крок процедури виконання ГА вдалося значно просунутися вперед на шляху пошуку максимального значення даної функції. Далі необхідно знову застосувати селекцію, схрещування і мутації, оцінити отриману генера-цію з точки зору її придатності і т. д. до тих пір, поки не буде задовольняти творяться умова зупинки.

Примітка. Мутовані рядки позначені двома зірочками, а змінені біти виділені жирним шрифтом.

Програмне забезпечення генетичних алгоритмів.

Одна з переваг використання ГА для вирішення різних завдань полягає в тому, що для реалізації ГА не потрібно створювати окремий програмний продукт. Единственне, що потрібно від користувача, - це уявити шукане рішення у вигляді хромосоми і сформувати функцію придатності. Далі реалізація ГА відбувається незалежно від конкретності розглянутої задачі. Враховуючи схему виконання ГА, розглянуту вище, процес програмування не представляє значних труднощів, так як реалізація ГА є циклічна процедура застосування генетичних операторів до початкової популяції хромосом.

ЛІТЕРАТУРА

ОСНОВНА

1. Уотермен Д. Руководство по экспертным системам.-М.: Мир, 1989.
2. Элти Дж., Кумбс М. Экспертные системы: концепции и примеры.-М.: Финансы и статистика, 1987.
3. Представление и использование знаний/ Под ред. Х.Уэно.-М.: Мир, 1989.
4. Лорьер Ж.-Л. Системы искусственного интеллекта.-М.: Мир, 1991.
5. Любарский Ю.Я. Интеллектуальные информационные системы.-М.: Наука, 1990.
6. Левин Р., Дранг Д., Эделсон Б. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на Бейсике.-М.: Финансы и статистика, 1991.
7. Хювенен Э., Сеппанен И. Мир ЛИСПа.-М.: Мир, 1989.
8. Дьяконов В. П., Круглов В. В. MATLAB 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6 в. Инструменты искуственного интеллекта и биоинформатики. – М.: Солон-Пресс, 2006.
9. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB. Спец. справочник. СПб.: Питер, 2001.
10. Таусенд К., Фохт Д. Проектирование и программная реализация экспертных систем на персональных ЭВМ.-М.: Финансы и статистика, 1990.
11. Уоссермен Ф.. Нейрокомпьютерная техника : Теория и практика. - М.: Мир. 1992.
12. Леоненков А. Нечеткое моделирование в среде MATLAB и fuzzyTECH. Санкт-Петербург, 2003,720с.
13. Барский А.Б. Нейронные сети: распознавание, управление, принятие решений. Москва. 2004.175с.
14. Кричевский М.Л. Интеллектуальные методы в менеджменте. Нейронные сети. Нечеткая логика, Генетические алгоритмы. Динамические системы. Питер, 2005, 304с.

ДОДАТКОВА

15. Сойер Б., Фостер Д.Л. Программирование экспертных систем на ПАСКАЛЕ.-М.: Финансы и статистика, 1990.
16. Дорохов И.Н., Меньшиков В.В. Системный анализ процессов химической технологии. Интеллектуальные системы и инженерное творчество в задачах интенсификации технологических процессов и производств. Москва, Наука, 2005, 582с.
17. Братко И. Программирование на языке ПРОЛОГ для искусственного интеллекта.-М.: Мир, 1990.
18. Марселлус Д. Программирование экспертных систем на Турбо-Прологе. Москва. Финансы и статистика. 1994.

19. Усков А.А., Кузьмин А.В. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика. Москва. 2004.
20. Потёмкин В.Г. Инструментальные средства MATLAB 5.x. М.: Диалог-МИФИ, 2000.
21. Тэрано Т., Асаи К., Сугено М. Прикладные нечеткие системы. М.Мир.1993.

НАВЧАЛЬНО-МЕТОДИЧНА

22. Дранишников Л.В., Інелектуальні методи в управлінні. Навчальний посібник. Кам'янське: ДДТУ, 2018, 416 с.
23. Конспект лекцій з дисципліни "Інтелектуальні методи в управлінні" для студентів 5 курсу денної та заочної форм навчання, що навчаються за напрямком 7,8.0840303 «*Програмне забезпечення автоматизованих систем*». /Укладач Л.В.Дранишников –Дніпродзержинськ, ДДТУ, 2015. - 134с.
24. Методичні вказівки до практичних робіт з дисципліни «Інтелектуальні методи в управлінні» для студентів денної та заочної форм навчання, що навчаються за напрямком 7,8.0840303 «*Програмне забезпечення систем*». /Укладач Л.В.Дранишников –Дніпродзержинськ, ДДТУ, 2015. - 50с.
25. Методичні вказівки до виконання курсової роботи з дисципліни „Інтелектуальні методи в управлінні” *для здобувачів вищої освіти освітньо-професійної програми другого (магістерського) рівнязі спеціальності 121 Інженерія програмного забезпечення.* /Укладач Л.В.Дранишников – Кам’янське: ДДТУ, 2017. – 72 с.
26. Методичні вказівки до до самостійної роботи з дисципліни «Інтелектуальні методи в управлінні» для здобувачів вищої освіти другого (магістерського) рівня зі спеціальності 121 «Інженерія програмного забезпечення»за освітньо-професійною програмою «Інженерія програмного забезпечення» /Укладач Л.В.Дранишников – Кам’янське: ДДТУ, 2018. – 14 с.
27. Конспект лекцій з дисципліни "Інтелектуальні методи в управлінні" для підготовки студентів магістрів денної та заочної форм навчання що навчаються за спеціальностю 121 Інженерія програмного забезпечення /Укладач Л.В.Дранишников – Кам’янське: ДДТУ, 2017. – 171 с.
28. Богатиков В.Н., Дранишников Л.В., Пророков А.Е. Построение систем управления на основе нейронных сетей. Учебно-методическое пособие. г. Апатиты. 2011г. .41с.
29. Богатиков В.Н., Дранишников Л.В., Пророков А.Е. Основы построения нечетких систем управления. г. Апатиты. 2011г. 72с.
30. Богатиков В.Н., Дранишников Л.В., Пророков А.Е. Моделирование систем автоматического регулирования технологических процессов. г. Апатиты. 2011г. 18с.

НАВЧАЛЬНЕ ВИДАННЯ

Конспект лекцій з дисципліни «Інтелектуальні методи в управлінні**»** для здобувачів другого (магістерського) рівня вищої освіти зі спеціальності 121 «Інженерія програмного забезпечення» за освітньо-професійною програмою «Інженерія програмного забезпечення»

Укладач:
Дранишников Леонід Васильович

Підписано до друку _____ 2020р.
Формат _А4_ Обсяг_13__ др. арк.
Наклад _____ прим. Замовлення_____
51918, м.Дніпродзержинськ,
вул. Дніпробудівська, 2.