# Анотація

У даній дипломній кваліфікаційній роботі розглядається створення нейронної мережі для серверної частини системи оброблення статистичних даних, яка в перспективі мала би змогу адаптивно проводити навчання і коригування власних вагових коефіцієнтів згідно зміни зовнішнього середовища.

В результаті було створену систему для прогнозування стану пацієнта(користувача) та класифікації його вхідних параметрів для віднесення вектору даних до прогнозованого діагнозу та групи ризику. Результатом роботи являється набір клієнтських бібліотек на базі відкритої платформи .NET які використовуються на серверній частині системи. Також було створено прикладний програмний інтерфейс, застосовуючи фреймворк ASP.NET WEB API 2. Було спроектовано структуру класів нейронної мережі для швидкої побудови і навчання мережі яка би змогла виконувати любого роду задачі класифікації.

В першому розділі дипломної роботи проводиться аналіз можливостей нейронних мереж, типів активаційних функції, і різних типів архітектур.

В другому розділі дипломної роботи проведено аналіз машинного навчання, різних видів асоціативної пам’яті та можливості навчання різного типу. Також було проаналізовано задачі які може виконуватись мережа. Також було розглянуто проблему адаптації нейронних мереж.

В третьому розділі дипломної роботи розглянуто засоби і технології які застосовувались при реалізації даної системи.

В четвертому розділі наведено результати виконання системи.

Робота міститить 30 рисунків, 10 літературних джерел, та 2 додатка.

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, .NET, SQL, ASP NET WEB API, ADO .NET.

**Abstract**

# In this thesis qualifying work considers the creating of neural network for server side of the system processing statistical data, which in the future would give the possibility to make learning and configuring weight coefficients according to changes of the external environment in adaptive way.

As a result, created the system for prognostication the state of patient (user) and classification of input parameters for assignment of vector data to predictive diagnosis and group risk. The result of this work include a set of client libraries based on .NET open platform which will be used in server part of application.

It was also created application programming interface using ASP.NET WEB API 2 framework. It was planned the structure of future classes of neural network for faster building and learning network, which would be able to make any sort of classification tasks.

In the first chapter of the thesis, analyzes the possibilities of neural networks, types of activation functions and types of architectures.

In the second chapter of this thesis, analyzes the machine learning, types of associate memory, and possibilities of different types of machine learning’s. Also was analyzed tasks which can be handle by neural network. Also was analyzed the adaptive problem of neural networks.

In the third section of the thesis deals with tools and technologies which were applied in the implementation of the system.

# In the fourth section provides results.

The work includes: 30 drawings, 10 literary sources and 2 of Appendix.

ARTIFICIAL NEURAL NETWORKS, MACHINE LEARNING, .NET, SQL, ASP NET WEB API, ADO .NET.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

HTML – HyperText Markup Language (Мова розмітки гіпертексту)

MVC – Model-view-controller (Моде́ль-вид-контро́лер)

SQL – Structured Query Language (Структурована мова запитів)

API – Application Programming Interface (Прикладни́й програ́мний інтерфейс)

WCF - Windows Communication Foundation

ASP.NET – Active Server Pages.

IIS – Internet Information Services (Інтернет інформаційний сервіс).

CLR – Common Language Runtime (Загальне середовище виконання мов).

# ЗМІСТ

[ВСТУП 9](#_Toc359262822)

[1. Штучні нейронні мережі 11](#_Toc359262840)

[1.1 Штучний інтелект і нейронні мережі 11](#_Toc359262841)

[1.2 Модель нейрона 16](#_Toc359262842)

[1.3 Представлення нейронних мереж. Направлені графи 20](#_Toc359262843)

[1.4 Архітектура мереж 21](#_Toc359262844)

[1.5 Представлення знань 23](#_Toc359262845)

[1.6 Висновки 24](#_Toc359262846)

[2. ПРОЦЕСИ МАШИННОГО НАВЧАННЯ. 25](#_Toc359262824)

[2.1 Машинне навчання та нейронні мережі. 25](#_Toc359262825)

[2.2 Навчання на основі корекції помилок 26](#_Toc359262826)

[2.3 Навчання на основі пам’яті. 29](#_Toc359262827)

[2.4 Задача присвоєння коефіцієнтів довіри. 30](#_Toc359262828)

[2.5 Навчання з учителем. 31](#_Toc359262829)

[2.6 Навчання без учителя. 33](#_Toc359262833)

[2.7 Задачі навчання 36](#_Toc359262830)

[2.8 Адаптація нейронних мереж. 40](#_Toc359262831)

[2.9 Пам'ять 43](#_Toc359262832)

[2.10 Багатошаровий персептрон 46](#_Toc359262834)

[2.11 Інтелектуальні системи 49](#_Toc359262835)

[2.12 Висновки 51](#_Toc359262836)

[3. Вибір засобів для реалізації 52](#_Toc359262850)

[3.1 Microsoft Visual Studio 52](#_Toc359262851)

[3.2 .NET Framework 4,5 53](#_Toc359262852)

[3.3 Середовище CLR і ядро платформи 54](#_Toc359262853)

[3.4 Прикладні технології 56](#_Toc359262854)

[3.5 Технології серверної частини 57](#_Toc359262855)

[3.6 Висновки 59](#_Toc359262855)

[4. Реалізація 60](#_Toc359262850)

[4.1 Призначення і загальні вимоги 60](#_Toc359262851)

[4.2 Структура системи 61](#_Toc359262852)

[4.3 База даних. 65](#_Toc359262853)

[5. ЕКОНОМІЧНА ЧАСТИНА 66](#_Toc359262858)

[5.1 Економічна характеристика проектного рішення 66](#_Toc359262859)

[5.2 І Розрахунок витрат на розробку і впровадження проектного рішення 66](#_Toc359262860)

[5.3 Визначення комплексного показника якості 70](#_Toc359262861)

[5.4 Визначення експлуатаційних витрат 73](#_Toc359262862)

[5.6 Висновки 80](#_Toc359262864)

[ВИСНОВКИ 80](#_Toc359262864)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 81](#_Toc359262864)

[ДОДАТКИ 82](#_Toc359262864)

Вступ

Є категорії проблем, які не можуть бути сформульовані у вигляді алгоритму. Проблеми які залежать від багатьох тонких факторів, наприклад, ціна покупки нерухомого майна, яку наш мозок може (приблизно) розрахувати. Без алгоритму комп’ютер не може зробити те ж саме. Тому питання, яке слід задати: Як ми вчимося досліджувати такі проблеми?

Люди може навчатись – це та можливість, якої комп'ютерам явно не вистачає. Люди мають мозок, який може вчитися. Комп'ютери мають деякі блоки обробки і пам'ять. Вони дозволяють комп'ютеру виконувати найбільш складні чисельних розрахунки в дуже короткий проміжок часу, але вони не являються адаптивними.

Якщо порівняти комп'ютер і мозок, можна відзначити, що, теоретично, комп'ютер повинен бути більш потужним, ніж наш мозок: Він включає в себе 109  транзисторів з часом перемикання 10-9 секунд. Мозок містить 1011 нейронів, але вони мають лише час перемикання близько 10-3 секунд.

|  |  |  |
| --- | --- | --- |
|  | Людський мозок | Комп’ютер |
| Кількість блоків обробки | 1011 | 109 |
| Тип блоків обробки | Нейрони | Транзистори |
| Тип обрахунків | Масовий паралелізм | Як правило - послідовний |
| Час перемикання | 10-3с | 10-9 с |
| Максимальна можлива кількість операцій за секунду | 1013 | 1018 |

Таблиця.1 Порівняння між людським мозком і комп’ютером

Найбільша частина мозку постійно працює, в той час як найбільша частина комп'ютера використовується тільки як пасивне зберігання даних. Таким чином, мозок працює паралельно і, отже, виконує кількість операцій яка є близькою до теоретичного максимуму. (Таблиця 1). Крім того, комп'ютер є статичним - мозок як біологічна нейронна мережа може перебудуватися під час свого "життя", отже, здатний вчитися, щоб компенсувати помилки.

Таким чином, вивчення штучних нейронних мереж мотивується їхньою подібністю до успішно працюючих біологічних систем, які - в порівнянні із звичайними системами - складаються з дуже простих, але численних нервових клітин, які працюють в широкому масштабі паралельно і (що ймовірно, є одним з найбільш важливих аспектів) мають можливість вчитися. Там немає необхідності явно програмувати нейронну мережу.

Одним з результатів даної процедури навчання є здатність нейронних мереж узагальнювати і асоціювати дані: після успішного навчання нейронної мережі можна знайти розумні рішення для аналогічних завдань одного і того ж класу, що не були явно визначені в процесі навчання. Це, в свою чергу, призводить до високого ступеня відмовостійкості від змінених вхідних даних.

Стійкість до збоїв тісно пов'язана з біологічними нейронними мережами, в яких ця характеристика дуже різна: Як згадувалося раніше, людина має близько 1011 нейронів, які безперервно перебудовуються або реорганізовуються шляхом зовнішніх впливів. Проте, наші пізнавальні здібності не піддаються істотному впливу.

Отже, як підсумок, основні характеристики які намагаються адаптуватися з біології в теоріях нейронних мереж:

* Самоорганізація і здатність до навчання,
* Узагальнюючої здатності
* Відмовостійкість.

1. **ШТУЧНІ НЕЙРОННІ МЕРЕЖІ.**

**1.1 Штучний інтелект і нейронні мережі.**

Основною задачею штучного інтелекту являється розроблення парадигм або алгоритмів, які би забезпечували комп’ютерне вирішення задач, які властиві людському мозку. Варто зазначити, що дане визначення задачі штучного інтелекту не є єдиним можливим. Усі системи штучного інтелекту повинні забезпечувати вирішення наступних трьох задач: накопичення знань, застосування накопичених знань для вирішення проблеми і виділення знань з досвіду. Системи штучного інтелекту реалізують три ключові функції: представлення, міркування, навчання.



Рис.1.1.1 Ключові функції систем штучного інтелекту

1. Представлення(representation). Одною з важливих ознак систем штучного інтелекту являється використовування символьної мови для представлення загальних знань про предметну область і конкретних знань о способах вирішення задач. Символи зазвичай формуються в уже відомих термінах. Це робить символьне представлення простим і зрозумілим людині. Більш того, зрозумілість символьних систем штучного інтелекту робить їх пригодними для людино-машинного спілкування. Термін «знання», використовуваний розробниками систем штучного інтелекту, являється лиш ще одною назвою даних. Знання можуть мати процедурний і декларативний характер. В декларативному представлення знання – це статистичний набір фактів. При цьому існує відносно малий об’єм процедур, використовуваних для маніпуляцій цими фактами. Характерною властивість декларативного представлення, являється те, що в очах людини вона має сенс сама по собі, незалежно від використовування в системах штучного інтелекту. В процедурному представленні знання вставленні в процедури, які функціонують незалежно від сенсу знань. В більшості предметних областей, потрібно використовувати обидва типи представлення знань.
2. Міркування(reasoning). Під міркуванням зазвичай розуміють властивість вирішувати задачі. Для того щоб систему можна було назвати розумною, вона має відповідати наступним умовам:описувати і вирішувати широкий спектр задач, розуміти явну і не явну інформацію, повинна мати механізм управління.

Вирішення задач можна розглядати як деяку задачу пошуку. В процесі пошуку використовуються правила, дані і управляючий контроль. Правила діють на область даних, а управляючий контроль виділяються для правил. Для прикладу, якщо взяти відому «задачу комівояжера». В ній потрібно знайти найкоротший маршрут з одного з міст в інший. При цьому всі міста які розставлені по маршруту, необхідно відвідати тільки один раз. В цій задачі множина даних складається з всіх можливих маршрутів і їхньої вартості, представлена в формі зваженого графа. Правила визначають шляхи руху з одного міста в інший, а модуль контролю вирішує, коли і які з них використовувати. В багатьох практичних задачах доступний набір знань являється неповним і неточним. В таких випадках використовуються вірогідні міркування(probabalistic reasoning), дозволяючи системам штучного інтелекту працювати в умовах невизначеності.

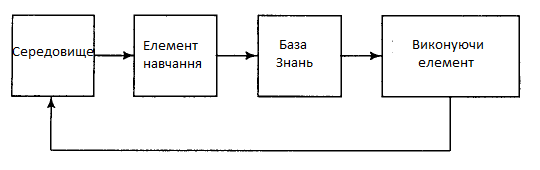


Рис.1.1.2 Проста модель машинного навчання.

1. Навчання. В найпростішій моделі машинного навчання(РИС) інформація для елементу навчання представляє саме середовище. Елемент навчання використовує отриману інформацію для модернізації бази знань, знання з якої функціональний елемент використовує для виконання поставленої задачі. Інформація яка отримується з зовнішнього середовища являється неідеальною, тому елемент навчання не знає заздалегідь як заповнити пробіли чи ігнорувати незначні деталі. Машина діє навмання, після чого отримує сигнал зворотного зв’язку від функціонального елементу. Механізм зворотного зв’язку дозволяє системі провіряти гіпотези і переглядати їх по мірі необхідності.

Машинне навчання може включати два зовсім різних підходи обробки інформації:індуктивний і дедуктивний. При індуктивній обробці інформації, загальні шаблони і правила створюються на основі практичного досвіду і потоків даних. При дедуктивній обробці інформації для визначення фактів, використовуються загальні правила. Навчання на основі подібності представляє собою індуктивний процес, а доведення теорем – дедуктивний, оскільки воно опирається на відомі аксіоми і вже доказані теореми. В навчання на основі пояснень використовується як дедукція так і індукція.

Виникаючі при навчання складнощі і накопичений при цьому досвід привели до створення різних методів і алгоритмів поповнення баз знань. Загалом, якщо в даній предметній області працюють досвідчені професіонали, простіше отримати їх загальний досвід, чим намагатись дублювати експериментальних шлях, який вони пройшли в процесі його накопичення. Ця ідея і покладена в основу експертних систем.

Виникає питання: як зрівняти моделі нейронних мереж і символьних систем штучного інтелекту? Для такого порівняння потрібно розбити проблему на три частини: рівень пояснення, стиль обробки і структуру представлення.

1. Рівень пояснення(Explanation level). Класичні системи штучного інтелекту засновані на символьному представленні. З точки зору пізнання АІ передбачається існування ментального представлення, в якому пізнання виконується як послідовна обробка символьної інформації.

В центрі уваги нейронних мереж знаходяться моделі паралельної розподіленої обробки. В цих моделях передбачається, що обробка інформації виконується за рахунок взаємодії великої кількості нейронів, кожен з яких передає сигнали збудження і гальмування іншим нейронам мережі. Більш того, в теорії нейронних мереж, велику увагу виділяється нейрон-біологічному описанню процесу пізнання.

1. Стиль обробки(processing style). В класичних системах штучного інтелекту, обробка відбувається послідовно, як і в традиційному програмуванні. Навіть якщо порядок виконання дії строго не визначений, операції все рівно виконуються по-кроково. Така послідовна обробка, скоріш за все, пояснюється послідовною природою мов і логічних знань, а також структурної машини фон Неймана. Не можна забувати про те, що класичні системи штучного інтелекту зародились практично в ту саму інтелектуальну епоху, що і машини фон Неймана.

На відмінно від них, концепція обробки інформації в нейронних мережах походить з принципу паралелізму, який являється джерелом їх гнучкості.

Якщо обчислення розподілені між багатьма нейронами, практично не є важливим, що стан окремого нейрона мережі відрізняється від очікуваного. Зашумлений або неповний вхідний сигнал все рівно можна розпізнати;пошкоджена мережа може продовжувати виконання своїх функції на задовільному рівні, а навчання не обов’язково може бути ідеальним. Продуктивність мережі в межах деякого діапазону знижується достатньо повільно. Крім того, можна додатково підвищити ефективність мережі, представляючи кожну властивість групою нейронів.

1. Структура представлення(representational structure). В класичних системах штучного інтелекту, в якості моделі виступає язик мислення, тому символьне представлення має квазі – лінгвістичну структуру. Подібно фразам звичайної мови, вираження класичних мереж штучного інтелекту, як правило, тяжкі і складаються шляхом систематизації простих символів. Враховуючи обмежену кількість символів, нові смислові фрази складаються на основі композиції символьних фраз і аналогій між синтаксичною структурою і семантикою.

З іншої сторони, в нейронних мережах природа і структура представлення являються ключовими проблемами. З усього вище сказаного наступні властивості притаманні власне системам штучного інтелекту і не притаманні нейронним мережам:

* Уявне представлення характеризується комбінаторною вибірковою структурою і не комбінаторною семантикою.
* Уявні процеси характеризуються чутливістю до комбінаторної структури представлення, з якими вони працюють.

Таким чином, символьні моделі штучного інтелекту – це формальні системи, засновані на використанні мови алгоритмів і представленні даних по принципу «зверху в низ», а нейронні мережі – це розподілені процесори, які мають вроджені здібності до навчанні і які працюють по принципу «знизу в верх». Тому при вирішення когнітивних задач має сенс створювати структурні моделі на основі зв’язків або гібридні системи, об’єднуючи два підходи. Це забезпечить поєднання властивостей адаптивності та роботоспроможності. Для реалізації даного підходи були розроблені методи витягування правил з навчених нейронних мереж. Ці результати не тільки дозволяють інтегрувати нейронні мережі в інтелектуальні машини, але й забезпечують вирішення наступних задач:

* Верифікація нейромережових компонентів в програмних системах. Для цього внутрішній стан нейронної мережі переводиться в форму, зрозумілу користувачеві.
* Покращення узагальнюючої здатності нейронної мережі за рахунок виявлення областей вхідного простору, не достатньо повно представлених в навчальній вибірці, а також виявлення умов при яких узагальнення не можливе.
* Виявлення прихованих залежностей на множені вхідних даних.
* Інтеграція символьного і конекціоністського підходів при розробці інтелектуальних машин
* Забезпечення безпеки систем для яких вона являється критичною.

**1.2 Модель нейрона**

Штучний нейрон являється базовим елементом обробки інформації в штучній нейронній мережі. В моделі цього елемента можна виділити такі складові елементи(Рис.1.2.1):

1. Зв’язки. Основною характеристикою зв’язку є його вага(або сила). Кожний сигнал який проходить по даному зв’язку помножується на відповідну вагу.
2. Суматор. Його завданням є сумування вхідних сигналів.
3. Функція активації. Ця функція використовується для обмеження вихідного сигналу. Зазвичай в нейронних мережах використовуються функції які обмежують амплітуду виходу в діапазоні [0,1].
4. Пороговий елемент. Цей елемент використовується для збільшення або зменшення вхідного сигналу. Часто він відображаєтся як статичний, додатковий вхід зі своєю вагою.

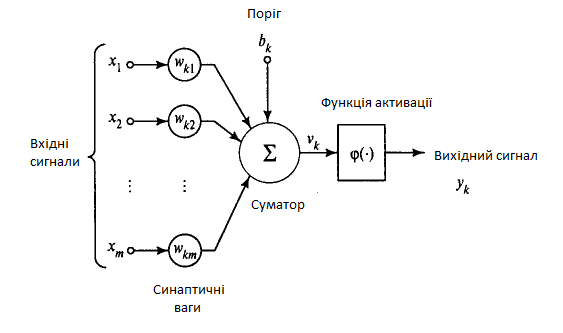


Рис.1.2.1 Модель нейрона

Математично представити штучний нейрон можна за допомогою рівнянь:

(1.2.1)

(1.2.2)

Де *x*1.. *x*m – вхідні сигнали, *wk*1…*wkm*– синаптичні ваги відповідних нейронів,

φ() - функція активації, *y*k – вихіднийсигнал нейрона.

В залежності від мінусового або додатного знаку порогу *b*k потенціал активації нейрона змінюється як на рисунку 1.2.2:

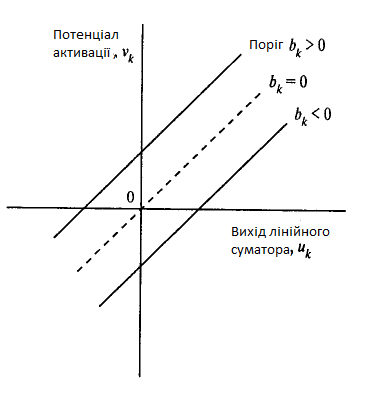


Рис.1.2.2 Афінне перетворення викликане наявністю порогу.

Як вже було сказано, поріг часто відображається як статичний вхід зі своєю вагою ( Рис. 1.2.3):

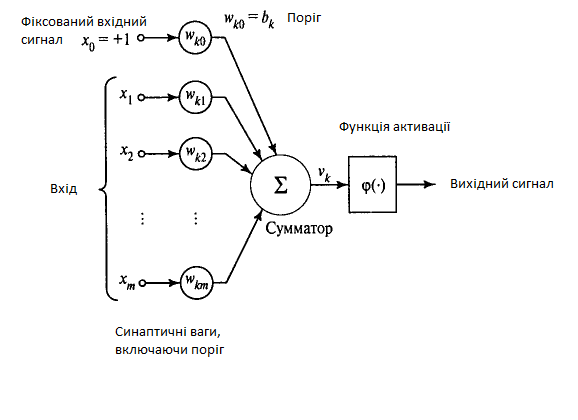


Рис.1.2.3 Модель нейрона з статичним входом

**Функція активації.**

Функція активації використовується як обмеження амплітуди вихідного сигналу. Можна виділити такі основні функції:

1. Порогова функція. Також вона відома як функція Хевісайда.

(1.2.3)

Де *v –* потенціал активації нейрона.

(1.2.4)

Ця модель називається моделлю ”все або нічого” Мак-Каллока-Пітца.

2.Кусково лінійна функція.

(1.2.5)

Тут потрібно зауважити, що дана функція використовується тільки тоді коли поріг рівний одиниці.

3.Сігмоїдальна функція. Найбільш поширена функція активації, яка є ідеальним балансом між лінійною і нелінійною поведінкою.

(1.2.6)

Тут *a* являється параметром нахилу функції, змінюючи який можна будувати функції різної крутизни.

* 1. **Представлення нейронних мереж. Направлені графи.**

Нейронна мережа – це направлений граф, який складається з вузлів які об’єднуються синаптичними і активаційними зв’язками, які характеризуються чотирма властивостями.

1)Кожний нейрон представляється множиною лінійних синаптичних зв’язків і, можливо, нелінійною функцією активації. Поріг, як вже було зазначено являється вхідним синаптичним зв’язком і в більшості випадків рахується рівним +1.

2)Синаптичні зв’язки нейрона використовуються для корегування відповідних вхідних сигналів.

3) Сума вхідних сигналів являє собою потенціал активації конкретного нейрона.

4)Функції активації модифікують потенціал активації нейрона, створюючи вихідний сигнал.

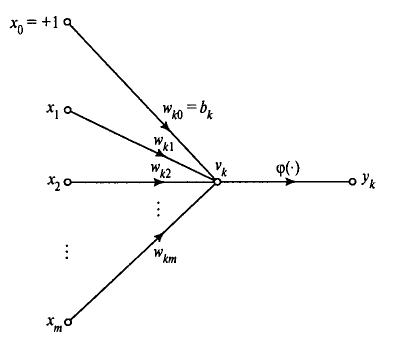


Рис.1.3.1 Граф обрахунку вихідного сигналу для нейрона.

Можна виділити три типи представлення нейронних мереж:

1. Блокова діаграма.
2. Граф обрахунку сигналу(Рис.1.3.1)
3. Архітектурний граф(Рис.1.3.2)

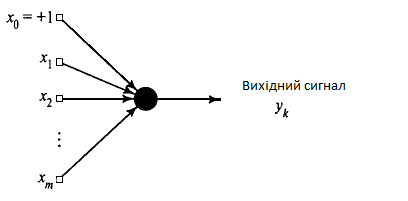


Рис.1.3.2 Архітектурний граф нейрона.

* 1. **Архітектура мереж**

Архітектура мережі завжди залежить від вибраного алгоритму навчання. Зазвичай виділяють три фундаментальних класса нейронних мережевих архітектур.

**Одношарові мережі прямого розповсюдження.**

В усіх нейронних мережах, нейрони встановлюються по шарам. В самому простому випадку існує вхідний шар, від якого інформація передається на вихідний шар нейронів. Така мережа називається мережею прямого розповсюдження. На рисунку 1.4.1 зображена дана архітектура яка називається одношаровою.

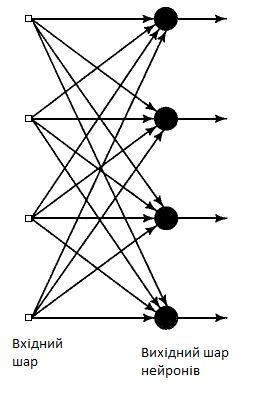


Рис.1.4.1 Одношарова мережа прямого розповсюдження.

**Багатошарові мережі прямого розповсюдження.**

Даний клас нейронних мереж прямого розповсюдження характеризується одним або більше прихованих шарів. Вони слугують посередником між вхідним і вихідним шаром нейронної мережі. Додавання таких прихованих мереж дозволяє нам виділяти глобальні властивості даних за допомогою додаткових синаптичних зв’язків .

Вузли вхідного шару формують вхідний вектор, який складає вхідний сигнал на перший прихований шар нейронів, вихідні сигнали другого шару використовуються як вхідні для третього і т.д. В результаті, вихідний сигнал останнього шару мережі формує вихідний образ даної нейронної мережі.(Рис.1.4.2)

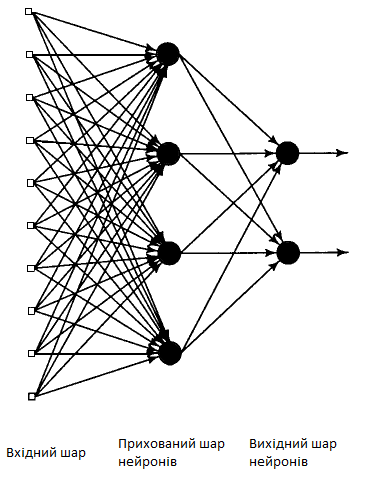


Рис.1.4.2 Мережа прямого розповсюдження з одним прихованим шаром.

**Рекурентні мережі.**

Рекурентна нейрона мережа повинна містити хоча б один зворотній звязок. Наприклад це може бути мережа де вихідний сигнал кожного нейрона є вхідним сигналом для нейронів того самого шару.(Рис.1.4.3) Такі мережі використовують так звані елементи затримки. Вони дозволяють виконувати нелінійну поведінку мережі. Наявність таких зворотніх звязків має величезний вплив на навчання нейронних мереж і на їхню швидкість.

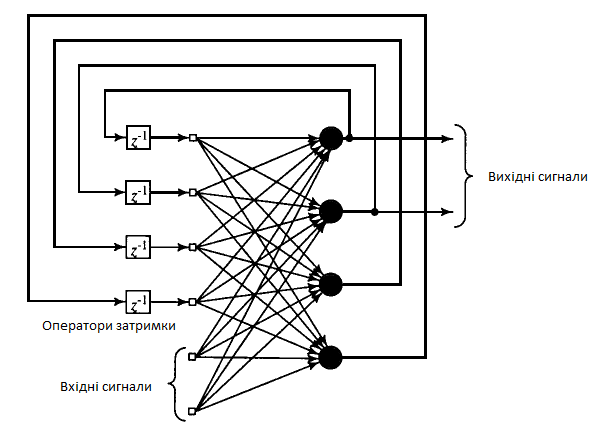


Рис.1.4.3 Рекурентна мережа

* 1. **Представлення знань**

Під знаннями розумується інформація яка зберігається людиною або машиною для інтерпретування, прогнозування і реакції на зовнішні події.

Основною задачею нейронної мережі являється якнайкраще навчання моделі зовнішнього світу для вирішення поставленої задачі. Знання про світ включають два типи інформації:

1. Відомий стан зовнішнього світу, представлені наявними достовірними фактами. Така інформація називається апріорною.
2. Спостереження за зовнішнім світом, отримані за допомогою сенсорів, адаптивних для конкретних умов. Зазвичай такі виміри в значній мірі спотворені, що потенційно може привести до помилок.

Приклади можуть бути маркованими і немаркованими. В маркованих прикладах вхідному сигналу відповідає очікуваний відклик. Немарковані приклади складаються з декількох різних реалізацій одного сигналу. В любому випадку набір прикладів представляють знання про зацікавлену предметну область, на основі яких і проводиться навчання нейронної мережі.

Множина векторів вхід – вихід, кожен з яких складається з вхідного сигналу і очікуваному йому виходу називають навчальною вибіркою. Зазвичай така множина векторів складає як позитивні так і негативні приклади.

* 1. **Висновки**

Отже, можна зробити висновок, що нейронна мережа є основною концепцією розвитку штучного інтелекту. Дана технологія застосування простих обчислювальних елементів дає змогу робити прогнозування та класифікацію на основі вхідного вектора даних. Оскільки при поставленні діагнозу, або при обробці любих інших статистичних даних, вхідний вектор містить велику кількість елементів, задача віднесення вектору до якогось образу робиться надзвичайно складною. Проте нейронні мережі за допомогою великої кількості синаптичних зв’язків дають змогу виконати дану задачу.

**2. ПРОЦЕСИ МАШИННОГО НАВЧАННЯ.**

**2.1 Машинне навчання та нейронні мережі.**

Найважливішою властивістю нейронних мереж являється їх властивість навчатися на основі даних оточуючого середовища і в результаті навчання підвищувати свою продуктивність. Підвищення продуктивності відбувається з часом у відповідності до визначених правил. Навчання нейронної мережі відбувається шляхом інтерактивного процесу коректування синаптичних ваг і порогів. В ідеальному випадку, нейронна мережа отримує знання про оточуючий світ на кожній ітерації процесу навчання.

З поняттям навчання асоціюється досить багато видів діяльності, тому тяжко дати цьому процесу однозначне визначення. Більш того, процес навчання залежить від точки зору на нього.

Навчання – це процес, в якому вільні параметри нейронної мережі настроюються шляхом моделювання середовища, в яке ця мережа встановлена. Тип навчання визначається способом настроювання цих параметрів.

Це визначення процесу навчання передбачає наступну послідовність подій:

* В нейронну мережу поступають стимули з зовнішнього середовища
* В результаті цього змінюються вільні параметри нейронної мережі.
* Після зміни внутрішньої структури, нейрона мережа відповідає на збудження вже іншим чином.

Цей список правил називається алгоритмом навчання. Насправді, не існує універсального алгоритму навчання який би підійшов до всіх архітектур нейронних мереж. Існує лиш набір засобів представлений множиною алгоритмів навчання, кожний з яких має свої переваги і недоліки. Алгоритми навчання відрізняються один від одного способом настройки синаптичних ваг нейронів. Ще одною характерною ознакою являється спосіб зв’язку навченої нейронної мережі з зовнішнім світом. В даному контексті говорять про «парадигму навчання», яка є зв’язана з моделлю навколишнього середовища, в якій функціонує дана нейронна мережа.

**2.2 Навчання на основі корекції помилок.**

Для простоти ілюстрації даного алгоритму розглянемо простий випадок коли нейрон *k* – єдиний обчислювальний вузол вихідного шару нейронної мережі прямого розповсюдження. Нейрон *k* працює під управлінням вектора сигналу x(n), який виходить з одного або декількох прихованих шарів, які в свою чергу отримують інформацію з вхідного вектора представленим початковим вузлом нейронної мережі. Під n розуміється дискретний час, або, більш конкретно – номер кроку інтерактивного процесу корегування синаптичних ваг нейрона *k*. Вихідний сигнал нейрона *k* позначається як *yk(n)*. Цей сигнал являється єдиним виходом нейронної мережі. Він буде порівнюватись з очікуваним виходом, який позначається *dk(n).* В результаті отримуємо сигнал помилки *ek(n)*:

(2.2.1)

Сигнал помилки ініціалізує механізм управління, ціль якого заклечається в застосуванні послідовності коректувань до синаптичних ваг нейрона *k*. Ця зміна націлена на покрокове наближення вихідного сигналу *yk(n)* до очікуваного *dk(n).*

Ця ціль досягається за рахунок мінімізації функції вартості або індексу продуктивності E(n), який визначається в термінах сигналу помилки наступним чином:

(2.2.2)

Де E(n) – теперішнє значенні енергії помилки. Покрокове коректування синаптичних ваг нейрона *k* продовжується до тих пір поки система не дійде до рівня стабільного стану(тобто такого, при якому синаптичні вага практично стабілізуються). В цій точці процес зупиняється.

Процес описаний вище, називається навчанням, на основі корекції помилок. Мінімізація функції E(n) виконується по так званому дельта – правилу, або правилу Відроу-Хоффа, названого так в честь його засновників.

Позначимо *wkj(n)* як теперішнє значення синаптичної ваги *wkj*  нейрона *k,* який відповідає елементу *xi(n)* вектора x(n), на кроці дискретизації n. У відповідності до дельта-правилу зміна ∆*wkj(n),* яке відноситься до ваги  *wkj* на даному кроці дискретизації, задається виразом:

(2.2.3)

Де ŋ – деяка додаткова константа, яка являє собою швидкість навчання і використовується при переході від одного кроку процесу до іншого. З формули видно, що цю константу природно називати параметром швидкості навчання. Вербально дельта-правило можна визначити наступним чином:

*Коректування, яке застосовується до синаптичної ваги нейрона, пропорційна множенню сигналу помилка на вхідний сигнал який його викликав.*

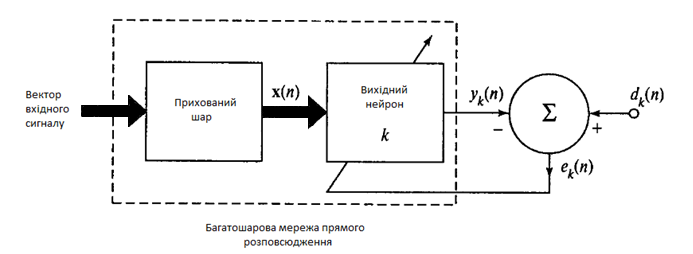


Рис.2.2.1 Блокова діаграма нейронної мережі

Для виконання дельта - правила передбачається можливість використання прямого виміру сигналу помилки. Для забезпечення такого виміру потрібно поступлення очікуваного відклику від деякого зовнішнього джерела, яке є доступним для нейрона *k.* Іншими словами, нейрон *k* повинен бути видимим для зовнішнього світу. З рисунку (Рис.2.2.1) видно що навчання на основі корекції помилок по своїй природі являється локальним. Це прямо вказує на те, що коректування синаптичних ваг по дельта – правилу може бути локалізована в окремому нейроні *k.*

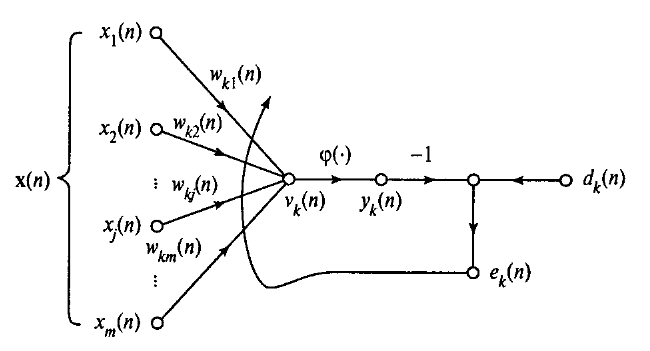


Рис.2.2.2 Граф передачі на основі корекції помилок.

Обрахувавши величину зміни синаптичної ваги ∆*wkj(n),*можна визначити його нове значення для наступного кроку дискретизації:

(2.2.4)

Таким чином, *wkj(n),* I *wkj(n+1),* можна розглядати як старе і нове значення синаптичної ваги *wkj*. В математичних термінах можна записати

(2.2.5)

Де z-1- оператор одиничної затримки. Іншими словами z-1 представляє собою елемент пам’яті.

На РИС представлений граф проходження сигналу в процесі навчання, заснованого на основі корекції помилок для нейрона *k*. Вхідний сигнал *xk* і індуктивне поле *uk* нейрона *k*  представлені у вигляді предсинаптичних і постсинаптичніх сигналів j-го синапсу нейрона *k.* На рисунку видно, що навчання на основі корекції помилок – це приклад замкнутої системи з зворотнім зв’язком. З теорії управління, відомо що стійкість такої системи визначається параметром зворотного зв’язку. В даному випадку, існує всього один зворотній зв'язок і єдиний параметр який нас цікавить швидкість навчання - ŋ. Для забезпечення стійкості або сходи мості ітеративного процесу навчання потрібно ретельно вибрати цей параметр. Вибір параметру швидкості навчання впливає також на точність і інші характеристики навчання. Іншими словами, параметр швидкості навчання грає ключову роль в забезпечення продуктивності процесу навчання на практиці.

**2.3 Навчання на основі пам’яті.**

При навчання на основі пам’яті весь минулий досвід накопичується в великому сховищі правильно класифікованих прикладів виду вхід-вихід.

:{(*xi, di*)}*N* де *xi* – вхідний вектор, а *di* – відповідний йому очікуваний вихідний сигнал. Наприклад, якщо розглядати задачу бінарного розпізнавання образів або класифікації на два класи (гіпотези) С1 і С2. В цьому прикладі, очікуваний відклик системи приймає значення 0(або – 1) для класу С1, і значення +1 для класу С2 . Якщо потрібно класифікувати деякий невідомий вектор із бази даних вибирається вихід, відповідний вхідному сигналу, який є близький до вектору.

Всі алгоритми навчання на основі пам’яті включають в себе дві складові:

* Критерій, який використовується для визначення округу вектора
* Правило навчання, яке застосовується до прикладу з округу тестового вектору.

Всі алгоритми відрізняються один від одного способом реалізації цих двух складових.

В самому простому алгоритмі навчання на основі пам’яті, який називається правилом найближчого сусіда, в округ включається приклад, який є близьким до тестового. Для цього використовується Евклідова відстань між векторами. Клас, до якого відноситься ближній сусід, являється також класом тестую чого вектора. Це правило не залежить від розподілу, який використовується при генерування прикладів навчання. При цьому аналіз ґрунтується на двух наступних гіпотезах:

* Класифікуючі приклади (*xi, di* )незалежні і рівномірно розподілені у відповідності з спільним розподілом приклада (x,d).
* Розмірність навчальної множини N безкінечно велика.

Показано, що при цих двох гіпотезах вірогідність помилки класифікації при використанні правила ближнього сусіда вдвічі перевищує баєсівську вірогідність помилки(Bayes probability error).

Варіацією класифікатора на основі ближнього сусіда являється класифікатор к-ближніх сусідів. Він описується наступним чином.

Знаходимо *k* класифікованих сусідів, ближніх до вхідного вектору, де *k –* деяке ціле число.

Вхідний вектор відносимо до того класу(гіпотезі), який частіше інших зустрічається серед *k –* ближніх сусідів тестую чого вектора.

Таким чином, класифікатор на основі *k –* ближніх сусідів працює подібно інструмента осереднювача.

**2.4 Задача присвоєння коефіцієнтів довіри.**

По суті дана задача – це задача присвоєння коефіцієнтів довіри чи недовіри всім результатам, які отримуються за допомогою машини яка, навчається.

В багатьох випадках залежність виходів від внутрішніх рішень визначається послідовністю дій, які машина виконує. Іншими словами, внутрішні рішення впливають на виконання конкретних дій, після чого, саме ці дії, а не самі рішення прямо визначають загальні результати. В такій ситуації можна виконати декомпозицію задачі присвоєння коефіцієнтів довіри на дві інші під задачі.

1. Присвоєння коефіцієнтів довіри результатам дій. Ця задача називається часовою задачею присвоєння коефіцієнтів довіри. В ній визначається проміжок часу, протягом якого реально виконуються дії, яким відпущений кредит довіри.
2. Присвоєння коефіцієнтів довіри дій внутрішнім рішенням. Це називається структурною задачею присвоєння коефіцієнтів довіри. В ній коефіцієнти довіри назначаються внутрішнім структурам дій, які генерує система.

Структурна задача присвоєння коефіцієнтів довіри має смисл в контексті багатокомпонентних навчаючих машин, коли необхідно точно визначити, поведінку якого елементу системи потрібно скоректувати і на яку величину, щоб підвищити загальну продуктивність системи. З іншої сторони, часова задача присвоєння коефіцієнтів довіри ставиться в тому випадку, коли навчаюча машина виконує достатньо багато дій, які приводять до деякого результату, і потребує визначити, які з цих дій несуть відповідальність за результат. Сукупність часової і структурної задачі присвоєння коефіцієнтів довіри потребує ускладнення поведінка розподіленою системи яка навчається.

Наприклад, задача присвоєння коефіцієнтів довіри виникає в тому випадку, коли навчання на основі корекції помилок застосовується до багатошарової нейронної мережі прямого розповсюдження. Дії кожного прихованого і вихідного нейрона такої мережі важливі для формування правильного результату в даній прикладній області. Це значить, що для рішення поставленої задачі не обхідно задати визначення форми поведінки всіх нейронів в процесі навчання на основі корекції помилок.

**2.5 Навчання з учителем.**

Навчання з учителем являється однією з основних парадигм навчання нейронних мереж. На рисунку 2.4.1 показана блокова діаграма, яка ілюструє цю форму навчання.

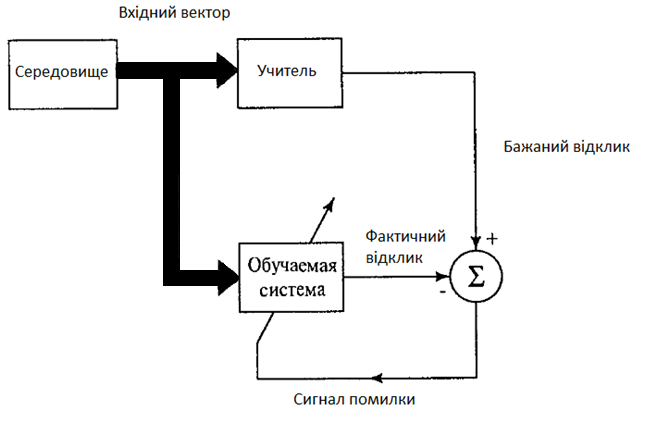


Рис.2.4.1 Блокова діаграма навчання з вчителем.

Концептуально участь учителя можна розглядати як наявність знань про оточуюче середовище представлені у вигляді пар вхід-вихід. При цьому саме середовище невідоме нейронній мережі яка навчається. Тепер допустимо, що учителю і мережі яка навчається подається навчаючий вектор з зовнішнього середовища. На основі вставлених знань, учитель може сформувати і передати навченій нейронній мережі бажаний відклик, який відповідає даному вхідному вектору. Цей бажаний результат представляє собою оптимальну дію, яку повинна виконати нейронна мережа. Параметри мережі коригуються з розрахунком навчаючого вектора і сигналом помилки. Сигнал помилки – це різниця між бажаним сигналом і теперішнім відкликом нейронної мережі. Коректування параметрів виконується покроково з метою імітації нейронною мережею поведінки учителя. Ця емуляція в деякому статистичному сенсі має бути оптимальною. Таким чином, в процесі навчання, знання учителя передаються в мережу в максимальному об’ємі. Після закінчення навчання, можна відключити і дозволити нейронній мережі працювати самостійно.

Дана форма навчання являється нічим іншим як навчання на основі корекції помилок яка була описана раніше. Це замкнута система з зворотнім зв’язком, яка не включає в себе зовнішнє середовище. Продуктивність такої системи можна оцінювати в термінах середньоквадратичної помилки або суми квадратів помилок на навчаючій вибірці, представленої у вигляді функції від вільних параметрів системи. Для такої функції можна побудувати багатовимірною поверхню помилки в координатах вільних параметрів. При цьому реальна поверхня помилки усереднюється по всім можливим параметрам, представленим у вигляді пар вхід – вихід. Люба конкретна дія системи з учителем представляється однією точкою на поверхні помилок. Для збільшення продуктивності системи по часі, значення помилки повинно зміщуватись в сторону мінімуму на поверхні помилок. Цей мінімум може бути як локальним так і глобальним. Це можна зробити, якщо система має в собі корисну інформацію о градієнті поверхні помилок. Градієнт поверхні помилок в любій точці – це вектор, який визначає напрямок найшвидшого спуску по цій поверхні. У випадку навчання з вчителем на прикладах вираховується моментальна оцінка вектора градієнта, в який вхідний вектор рахується функцією часу. При використанні результатів такої оцінки переміщення точки по поверхні помилок, зазвичай має вигляд «випадкового блукання» . Тим не менше, при використанні відповідного алгоритму мінімізації функції вартості, адекватному наборі навчальних прикладів у формі «вхід – вихід» і достатньому часі для навчання системи – навчання з учителем здатні вирішувати такі задачі як класифікація і апроксимація функцій.

**2.6 Навчання без учителя.**

Описаний вище процес навчання проходить у супроводі учителя. Альтернативною парадигмою навчання є навчання без учителя. Сама назва підкреслює відсутність керівника, який контролює процес коригування вагових коефіцієнтів. При використанні такого підходу немає ніяких маркованих прикладів, по яких проводиться навчання мережі. В цій альтернативній парадигмі можна виділити два методи.

**Навчання з підкріпленням, або нейродинамічне програмування.**

В навчанні з підкріпленням, формування відображення вхідних сигналів в вихідні виконується в процесі взаємодії з зовнішнім середовищем з цілью мінімізації скалярного індексу продуктивності. На рисунку 2.6.1 показана блокова діаграма однієї з форм системи навчання з підкріпленням.

Вона включає в себе блок «критики», який перетворює первинний сигнал підкріплення, який отримується з зовнішнього середовища, в сигнал більш високої якості, який називається евристичним сигналом підкріплення. Обидва сигнали являються скалярними.

Така система передбачає навчання з відкладеним підкріпленням. Це значить, що система отримує з зовнішнього середовища послідовність сигналів збудження, які приводять до генерації евристичного сигналу підкріплення. Цілю навчання являється мінімізація функції переходу, яка являє собою математичне очікування кумулятивної вартості дій протягом декількох кроків а, не просто теперішньої вартості.

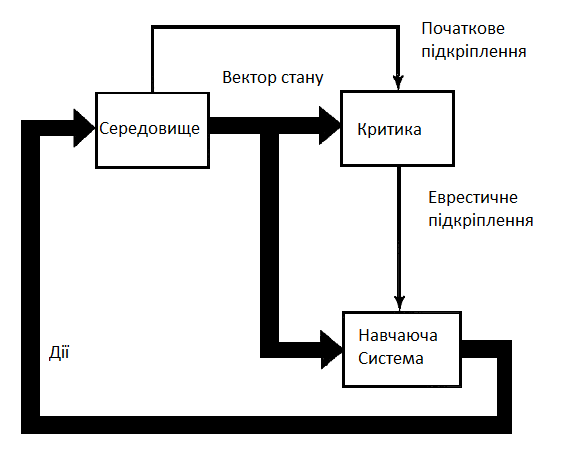


Рис.2.6.1 Блокова діаграма навчання з підкріпленням.

Практична реалізація навчання з відкладеним підкріпленням є ускладненою по двом причинам.

* Не існує учителя, який формує бажаний відклик на кожному кроці процесу навчання.
* Наявність затримки при формуванні первинного сигналу підкріплення потребує вирішення часової задачі присвоєння коефіцієнтів довіри. Це значить, що машина яка навчається, повинна бути здатна присвоювати коефіцієнти довіри і недовіри діям, які виконуються на всіх етапах які приводять до кінцевого результату, в той час як первинний сигнал з підкріпленням формується тільки на основі кінцевого результату.

Незважаючи на ці складності, системи навчання з відкладеним підкріпленням являються досить привабливими. Вони складають базис систем, які взаємодіють з зовнішнім середовищем, розвиваючи тим самим здатність самостійно вирішувати задачі на основі тільки своїх результатів

**Навчання без вчителя.**

Навчання без учителя виконується без зовнішнього вчителя, або коректора, який контролює процес навчання. Існує тільки незалежна від задачі міра якості представлення, якому повинна навчитись нейронна мережа і вільні параметри мережі оптимізуються по відношенню до цієї міри. Після навчання мережі на статичних закономірностях вхідного сигналу, вона здатна формувати внутрішнє представлення кодованих ознак вхідних данних, і таким же чином автоматично створювати нові класи.

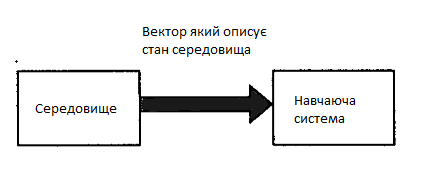


Рис.2.6.2 Блокова діаграма навчання без вчителя

Для навчання без учителя, можна використати правило конкретного навчання. Наприклад, можна використати нейронну мережу, яка складається з двох шарів – вхідного і вихідного. Вхідний шар получає доступні дані. Вихідний шар складається з нейронів, які конкурують один з одним за право відклику на ознаки, які є в вхідних даних. В найпростішому випадку нейронна мережа діє по принципу «переможить отримує все». Як було показано при такій стратегії нейрон з найбільшим сумарним вхідним сигналом «перемагає» і переходить в активний стан.

**2.7 Задачі навчання.**

Вибір конкретного алгоритму навчання залежить від задач, які нейронна мережа буде вирішувати. В даному контексті можна виділити 6 основних задач, для вирішення яких, в тому чи іншому вигляді використовуються нейронні мережі. В даній роботі буде розглянуто тільки ті задачі які стосуються спроектованої системи.

**Розпізнавання образів.**

Людський мозок добре пристосований до розпізнавання образів. Ми отримуємо дані з оточуючого світу через сенсори і здатні розпізнати джерело даних. Найчастіше це виконується моментально без всяких зусиль. Наприклад, ми можемо впізнати знайоме лице, людина може пізнати знайомий голос по телефону незважаючи на перешкоди на лінії зв’язку і т.д.

Розпізнавання образів формально визначається як процес, в якому отриманий сигнал або образ повинен бути віднесений до одного з визначених класів(категорій). Щоб нейронна мережа могла вирішувати задачі розпізнавання образів, її спочатку необхідно навчити, даючи їй послідовність вхідних образів разом з категоріями до яких вони належать. Після навчання мережі, на вхід подається раніше не використовуваний образ, який належить до того же набору категорій до якого належали образи в навчальній вибірці. Завдяки інформації виділеної під час навчання, мережа може віднести вхідний образ до конкретного класу. Розпізнавання образів, яке виконується мережею являється статистичним. При цьому образи представляються окремими точками в просторі яке називається простором рішень. Весь простір рішень розділяється на окремі області, кожна з яких асоціюється з визначеним класом. Границі цих областей якраз і формуються в процесі навчання. Побудова цих границь виконується статистично на основі дисперсії.

В цілому машини розпізнавання образів, створені на основі нейронних мереж можна розділити на два типи:

* Система складається з двох частин: мережі витягування ознак(без учителя) і мережі класифікації(з учителем).



Рис.2.7.1 Блокова діаграма системи розпізнавання образів

Такий метод відповідає традиційному підходу до статистичного розпізнавання образів. В концептуальних термінах образ представляється як набір із m спостережень, кожне з яких можна розглядати як точку x в m-мірному просторі даних. Витягнення ознак описується за допомогою перетворення, яке переводить точку х в проміжну точку у в q-мірному просторі ознак де q < m (Рис2.7.1).

Це перетворення можна розглядати як операцію зниження розмірності, яка полегшує задачу класифікації. Сама класифікація описується як перетворення, яке відображає проміжну точку у в один з класів r-мірного простору рішень.

* Система проектується як єдина, одношарова мережа прямого розповсюдження, яка використовує один з алгоритмів з учителем. При цьому підході задача витягнення ознак виконується обчислювальними вузлами прихованого шару мережі.

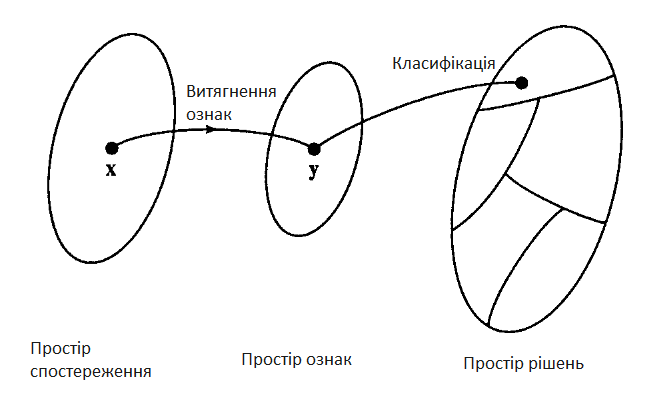


Рис.2.7.2 Ілюстрація класичного підходу до розпізнавання образів

**Апроксимація функцій.**

Наступною задачею навчання являється апроксимація функцій. Розглянемо нелінійне відображення типу «вхід-вихід», яке задається наступним чином:

(2.7.1)

де вектор x - вхід , а вектор d – вихід. Векторна функція f() рахується невідомою. Щоб за поповнити пробіл в знаннях про функцію f(), мережі надається множина маркованих прикладів:

(2.7.2)

До структури нейронної мережі, яка апроксимує невідому функцію, пред’являється наступна умова:функція F(), яка описує відображення вхідного сигналу в вихідний, повинна бути достатньо близько до функції f() в сенсі Евклідової норми на множині всіх вхідних векторів.

Властивість нейронної мережі апроксимувати невідоме відображення вхідного простору в вихідне можна використовувати для вирішення декількох задач.

* Ідентифікація систем. Нехай вище згадана формула описує відношення між входом і виходом в невідомій системі з декількома входами і виходами без пам’яті.

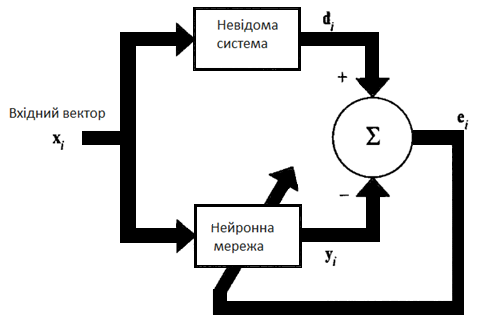


Рис.2.7.3 Блокова діаграма вирішення задачі ідентифікації системи

Термін «без пам’яті» означає інваріантність системи в часі. Тоді множина маркованих прикладів можна використовувати для навчання нейронної мережі, яка представляє модель цієї системи.

* Інверсні системи.

**Фільтрація**

Під терміном фільтр зазвичай мається на увазі пристрій, або алгоритм, який використовується для витягування корисної інформації із набору зашумлених даних. Шум може виникати по багатьох причинах. Наприклад, дані можуть бути виміряні з похибками, або шум може виникнути при передачі інформаційного сигналу через лінії зв’язку. Крім того, на корисний сигнал може бути накладений інший сигнал, який поступає з зовнішнього середовища. Фільтр можна використовувати для вирішення трьох основних задач обробки інформації.

* Фільтрація, тобто витягнення корисної інформації в дискретний момент часу із даних.
* Згладжування . Ця задача відрізняється від фільтрації тим, що інформація про корисний сигнал в момент часу не потребується, тому для витягнення інформації можна використовувати дані які отримуються пізніше. Це значить, що в згладжуванні при формуванні результату присутнє запізнення(delay). Так як при згладжуванні можна використовувати дані, не тільки до конкретного моменту але й після цього, цей процес в статистичному смислі являється більш точним чим фільтрація.
* Прогнозування. Цілю цього процесу являється прогноз відносно стану об’єкту управління в деякий момент часу, на основі даних отриманих до цього часу(включно)

**2.8 Адаптація нейронних мереж.**

При вирішенні реальних задач, часто стається так, що одним з основних вимірів процесу навчання являється простір, а іншим час. Просторово-часова структура навчання підтверджується багатьма прикладами задач навчання. Біологічні види, володіють здатністю до представлення часової структури досвіду. Таке представлення дозволяє тваринам адаптувати свою поведінку до часової структури подій в просторі поведінки.

Якщо нейронна мережа працює в стаціонарному середовищі, вона теоретично може бути навчена статистичними характеристиками середовища за допомогою учителя. Наприклад синаптичні ваги можна порахувати в процесі навчання на множині даних які представляють середовище. Після завершення процесу навчання, синаптичні ваги відображають статистичну структуру середовища, яка з цього моменту рахуватиметься незмінною або «замороженою». Таким чином, для витягнення і використовування накопиченого досвіду, система опирається на ту чи іншу форму пам’яті.

Однак часто, навколишнє середовище, являється нестаціонарним. Це значить, що статистичні параметри вхідних сигналів які генерує середовище, міняються в часі. В такому роді ситуацій, методи навчання з учителем доказали свою некомпетентність, так як мережа не володіє засобами відслідковування статичних варіацій середовища, з яким вона має діло. Щоб обійти цей мінус, потрібно постійно адаптувати вільні параметри мережі до варіацій вхідного сигналу в режимі реального часу, тобто адаптивна система, повинна відповідати на кожний наступний сигнал як на новий. Іншими словами, процес навчання в адаптивній системі не завершується, допоки в неї поступають нові сигнали для обробки. Така форма навчання називається неперервним навчанням.

Для реалізації безперервного навчання можна застосувати нелінійні адаптивні фільтри, які побудовані для лінійного суматора.

При дослідженні безперервного навчання і його застосування в теорії нейронних мереж, виникає питання. Як нейронна мережа може адаптувати свою поведінку до зміни часової структури вхідних сигналів в просторі поведінки? Одна з відповідей на це фундаментальне питання передбачає, що зміни статистичних характеристик нестаціонарних процесів протікають досить повільно, щоб процес на короткому проміжку часу, можна було розглядати як псевдо стаціонарний. Декілька прикладів:

* Синтез розмовного сигналу можна розглядати як стаціонарний процес на інтервалі приблизно 10-30 мілісекунд.
* Ехо радару від дна океану можна рахувати стаціонарним на інтервалах часу приблизно декількох секунд.
* При довготривалому прогнозуванні погоди, синоптичні дані можна розглядати як стаціонарні на інтервалах часу порядку декількох секунд.
* В контексті оцінки тенденцій біржового ринку дані можна рахувати стаціонарними на інтервалах часу приблизно декількох днів.

Використовуючи властивість псевдо стаціонарності в стохастичних процесах, можна збільшити термін ефективної роботи нейронної мережі за рахунок її періодичного перенавчання, яке дозволяє врахувати варіації вхідних даних. Такий підхід можна використовувати, наприклад, для обробки біржових даних.

Можна також використати і більш точний динамічний підхід. Для цього потрібно виконати наступну послідовність дій.

* Вибрати достатньо короткий інтервал часу, на якому дані можна рахувати псевдо стаціонарними, і викриростовувати для навчання мережі.
* Після отримання нового навчального прикладу, потрібно відкинути самий старший вектор і добавити в вибірку новий приклад.
* Використати оновлену вибірку для навчання мережі.
* Неперервно повторяти описану процедуру.

Описаний алгоритм дозволяє встановити часові властивості в архітектуру нейронної мережі, реалізуючи таким чином принцип неперервного навчання на упорядкованих по часі прикладах. При використання такого підходу нейронну мережу можна рахувати нелінійним адаптивним фільтром, який представляє собою узагальнення лінійного адаптивного фільтру. Однак для реалізації такого динамічного підходу, потребується велика швидкодія комп’ютера , яка би дозволила виконувати всі необхідні обрахунки за один інтервал дискретизації в реальному часі. Тільки в такому випадку фільтр не буде відставати від зміни даних на вході системи.

**2.9 Пам'ять**

В контексті нейробіології, під пам’ятю розуміють відносно тривала по часі деформація структури нейронів, яка була викликана на організм зовнішнього середовища. Без такої деформації пам'ять не існує. Щоб пам'ять була корисною, вони повинна бути доступною для нервової системи. Тільки тоді вона може впливати на майбутню поведінку організму. Однак для цього в пам’яті попередньо повинні бути накопичені відповідні моделі поведінки. Це накопичення виконується за допомогою процесу навчання. Пам'ять і навчання тісно пов’язані. При вивчені деякого образу він зберігається в структурі мозку, звідси його можна дістати у випадку необхідності. Формально, пам'ять можна розділити на короткотривалу і довготривалу, в залежності від часу можливого збереження інформації. Короткотривала пам'ять являється відображенням теперішнього стану навколишнього середовища. Кожний новий стан середовища, яке відрізняється від образу, який зберігається в короткотривалій пам’яті, приводить до обновленню даних в пам’яті. З іншого боку в довготривалій пам’яті зберігаються знання, які призначення для довготривалого використання. Асоціативна пам'ять характеризується наступними особливостями:

* Ця пам'ять являється розподіленою.
* Стимули і відклики асоціативної пам'яті представляють собою вектори даних
* Інформація запам’ятовується за допомогою формування просторових образів нейромережевої активності на великій кількості нейронів.
* Інформація, яка зберігається в стимулі, визначає не тільки місце її запам’ятовування, але і адрес для її витягування.
* Незважаючи на те, що нейрони не являються надійними обчислювальними елементами і працюють в умовах шуму, пам'ять володіє високою стійкістю до перешкод і спотворенню даних.
* Між окремими образами, які зберігаються в пам’яті, можуть бути внутрішні взаємозв’язки. Звідси і протікає вірогідність помилок при витягуванні інформації з пам’яті.

В розподіленій пам’яті головний інтерес представляє одночасне або майже паралельне функціонування множини різних нейронів при обробці внутрішніх або зовнішніх стимулів. Нейронна активність формує в пам’яті просторові зв’язки, які зберігають в собі інформацію про стимул. Таким чином, пам'ять виконує розподілення відображених образів в просторі вхідних сигналів в інші образи вихідного простору. Деякі важливі властивості відображення розподіленою пам’яті можна проілюструвати на прикладі ідеалізованої нейронної мережі, яка складається з двох шарів нейронів. На рисунку 2.9.1 показана мережа, яку можна розглядати як модельний компонент нервової системи. Всі нейрони вхідного шару з’єднані зі всіма нейронами вихідного. В реальних системах зв’язки між являються тяжкими і надлишковими. В моделі на рисунку рисунку 2.9.1 для представлення загального результату від всіх синаптичних контактів між дендритами нейронів вхідного шару і відгалуженнями аксонів нейронів вихідного шару використовувався ідеальний стан. Рівень активності нейрона вихідного шару може викликати вплив на ступінь активності любого нейрона вихідного шару.

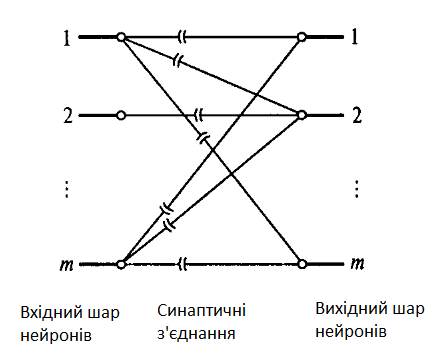


Рис.2.9.1 Компонент асоціативної пам’яті нервової системи.

Аналогічна ситуація для штучної нейронної мережі показаної на рисунку 2.9.3. В даному випадку, вузли джерела із вхідного шару і нейрони з вихідного шару працюють як обчислювальні елементи. Синаптичні ваги інтегровані в нейрони вихідного шару. Зв’язки між двома шарами мережі представляють собою прості з’єднання.

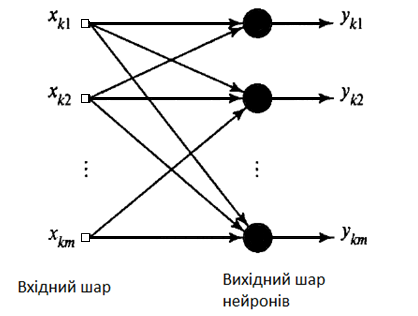


Рис.2.9.2 Модель асоціативної пам’яті з штучними нейронами.

* 1. **Багатошаровий персептрон**

Зазвичай багатошарові мережі прямого розповсюдження складається з множини сенсорних елементів(вхідних вузлів), які складають вхідний шар;одного або декількох прихованих шарів обчислювальних нейронів і одного вихідного шару нейронів. Вхідний сигнал розповсюджується по мережам в прямому напрямку, від шару до шару. Такі мережі зазвичай називаються багатошаровими персептронами.

Багатошарові персептрони успішно застосовуються для вирішення різних тяжких завдань. При цьому навчання з учителем відбувається за допомогою такого популярного алгоритму, як алгоритм зворотного розповсюдження помилки. Цей алгоритм засновується на корекції помилок. Його можна застосовувати як узагальнення такого ж популярного алгоритму адаптивної фільтрації – алгоритму мінімізації середньоквадратичної помилки.

Навчання цим методом передбачає два проходження по всім шарам мережі: прямого і зворотного. При прямому підході образ подається на сенсорні вузли мережі, після чого розповсюджуються від шару до шару. В результаті генерується набір вихідних сигналів, який і являється фактичною реакцією мережі на даний вхідний образ. Під час прямого проходження, всі синаптичні ваги фіксовані. Під час зворотного проходження всі синаптичні ваги настроюються у відповідності до правила корекції помилок, а саме: фактичний вихід мережі віднімається від бажаного відклику, в результаті чого формується сигнал помилки. Цей сигнал в результаті розповсюджується по мережі в напрямку який є зворотнім. Звідси й назва – алгоритм зворотного розповсюдження помилки. Синаптичні ваги настроюються з цілю максимального наближення вихідного сигналу мережі до бажаного в статистичному сенсі.

Багатошарові персептрони мають три основні ознаки:

* Кожен нейрон мережі, має нелінійну функцію активації. Важливо підкреслити, що дана нелінійна функція являється гладкою, на відмінну від жорсткою порогової функції яка використовується в перспетроні Розенблатта. Найбільш популярною формую функції являється сигмоїдальна функція, або як її ще називають – «логістична функція» (Logistic function).

(2.10.1)

Де *ui*- індуктивне локальне поле нейрона, *yi* – вихід нейрона. Наявність не лінійності грає дуже важливу роль, так як в протилежному випадку, відображення «вхід – вихід» мережі можна звести до звичайного одношарового персептрону. Більш того, використання логістичної функції мотивовано біологічно, так як в ній враховується відновлювальна фаза реального нейрона.

* Мережа містить один або декілька прихованих шарів нейронів, які не являються входом або виходом мережі. Ці нейрони дозволяють мережі навчатись вирішувати складні задачі, послідовно витягуючи найбільш важливі ознаки з вхідного образу.
* Мережа володіє високою ступеню зв’язності, яка реалізується шляхом синаптичних ваг. Зміна рівня зв’язності мережі потребує зміни множини синаптичних зєднань або їх вагових коефіцієнтів.

Комбінація всіх цих якостей, поряд з властивістю до навчання на власному досвіді забезпечує обчислювальну потужність багатошарового персептрона. Однак, всі ці якості являються причиною неповноти сучасних знань о поведінка такого роду мереж. По перше, розподілена форма не лінійності і висока зв’язність мережі значно ускладнюють теоретичний аналіз багатошарового персептрона. По друге, наявність прихованих нейронів робить процес навчання більш тяжким для візуалізації. Саме в процесі навчання необхідно визначити, які ознаки вхідного сигналу варто представляти прихованим нейронам. Тоді процес навчання стає ще більш тяжким, оскільки пошук повинен виконуватись в широкій області можливий функцій, а вибір повинен проводитись серед альтернативних представлень вхідних образів.

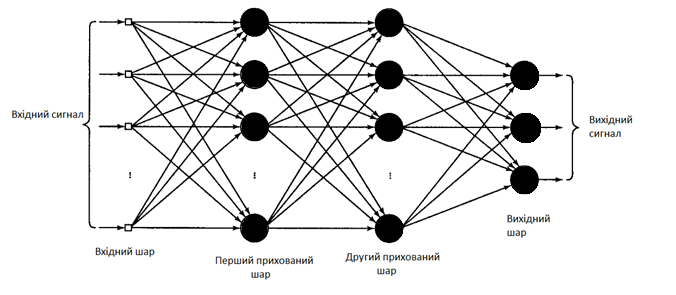


Рис.2.10.1 Архітектурний граф багатошарового перспетрона з двома прихованими шарами

Поява алгоритму зворотнього розповсюдження стало значною подією в області розвитку нейронних мереж, так як він реалізує обчислювально-ефективний метод навчання багатошарового персептрона.

На РИС показано фрагмент багатошарового персептрона. Для цього типу мережі виділяють два типи сигналів:

* Функціональний сигнал. Це вхідний сигнал, який поступає в мережу і передається вперед від нейрона до нейрона по всій мережі.

Такий сигнал доходить до кінця мережі в вигляді вихідного сигналу. Цей сигнал називається функціональним по двом причинам. По перше, він призначений для використання деяких функцій на виході мережі. По друге, в кожному нейроні, через який передається цей сигнал, вираховується деяка функція з розрахунком коефіцієнтів.

* Сигнал помилки. Сигнал помилки бере свій початок на виході мережі і розповсюджується в зворотному напрямку.

Вихідні нейрони складають вихідний шар мережі. Решту нейронів відносяться до прихованих шарів. Таким чином, приховані вузли не являються частиною входу чи виходу – звідси вони й отримали свою назву. Перший прихований шар отримує дані від вхідного шару, який складається з сенсорних елементів. Результуючий сигнал першого прихованого шару, в свою чергу передається на наступний прихований шар, і т.д.

Любий прихований або вихідний нейрон багатошарового перспетрона може виконувати два типа обчислень.

* Обчислення функціонального сигналу на виході нейрона, яке реалізується у вигляді неперервної нелінійної функції від вхідного сигналу і синаптичних ваг зв’язаних з даним нейроном.
* Обчислення оцінки вектора градієнта необхідного для зворотного проходу через мереду.

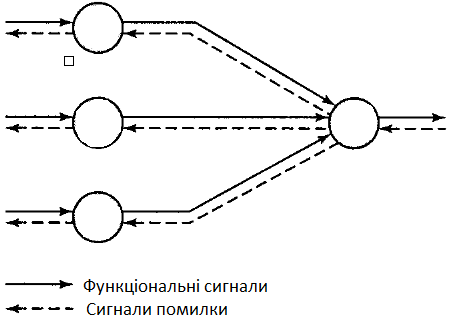


Рис.2.10.2 Напраямок двох основних потоків сигналів для багатошарового перспетрона.

**2.11 Інтелектуальні системи**

Більша частина зусиль дослідників нейронних мереж була сфокусована на розпізнаванні образів. Враховуючи практичну важливість цієї задачі і її повсюдну природу, а також той факт, що нейронні мережі виключно добре підходять для вирішення задач класифікації, така концентрація зусиль вчених направлялась на пошук засобів коректної класифікації. Розвиваючи цей напрямок, стало можливим закласти основи адаптивної класифікації образів.

На рисунку 2.11.1 показана структура гіпотетичної системи класифікації. Перший рівень такої системи отримує сенсорні дані, які генеруються деяким джерелом інформації. Другий рівень витягує множину ознак, яка характеризує отримані сенсорні дані. Третій рівень класифікує ці ознаки в одну чи декілька категорій, які помішаються в глобальний контекст на четвертому рівні.

Важливими ознаками, які характеризують дану систему являється наступні:

* Розпізнавання, яке являється результатом прямого проходження інформації від одного рівня системи до іншого в традиційній системі класифікації.
* Фокусування, при якій більш високий рівень системи здатен вибірково впливати на обробку інформації більш низьким рівнем за допомогою знань, накопичених на основі даних.

Таким чином, новаторство системи класифікації образів, показаної на рисунку 2.11.1 лежить в знаннях о цільової області і їх використанні нижнім рівнем системи для збільшення загальної продуктивності системи при наявних фундаментальних обмеженнях об’єму інформації яка обробляється.

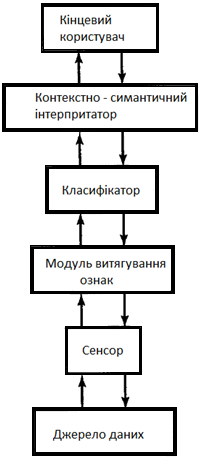


Рис.2.11.1 Функціональна архітектура інтелектуальної системи.

**2.12 Висновки**

Отже, навчання нейронної мережі надзвичайно складний і багатокроковий процес, і ніколи не має гарантій, що даний процес буде успішним. Проте навчена нейронна мережа прекрасно справляється з поставленими задачами навіть тоді, коли вектор вхідних даних відрізняється від тих, що були у навчальній вибірці. Проте коли нейронна мережа зустрічає на вході дані які кардинально відрізняються від тих, що використовувались при навчання вона може інтерпретувати їх некоректно. Це і є основна проблема адаптивності. В реалізованій системі, адаптивність досягається постійним коригуванням ваг нейронної мережі в «реальному часі».

**3. Вибір засобів для реалізації**

**3.1 Microsoft Visual Studio**

Microsoft Visual Studio Professional - засіб для незалежних розробників, що дозволяє вирішувати основні завдання розробки: створення, налагодження та розгортання багаторівневих інтернет-додатків на різних платформах, включаючи SharePoint, численні пристрої і середовища. Microsoft Visual Studio дозволяє підтримувати і модернізувати існуючі програми, а також створювати програми для нових платформ, наприклад Windows 8.

Робоче середовище Visual Studio має новий контекстно-залежний інтерфейс. Головна його особливість полягає в тому, що він пропонує розробникові тільки ті функції та інструменти, які йому потрібні на даному етапі роботи. Таким чином, панель інструментів не містить нічого зайвого і не ускладнює пошук потрібних функцій.

Середовище для розробки Visual Studio створене так, щоб допомагати розробнику і підказувати йому найкращі рішення. Для цього в продукт були включені спеціальні технології, які спрощують і прискорюють роботу. Наприклад, функція Code Clone шукає однакові по функціоналу коди і об'єднує їх у відповідні модулі для того, щоб розробник згодом міг скористатися вже готовим кодом при створенні нової програми.

Visual Studio дозволяє ефективно керувати повним життєвим циклом додатки від етапу його розробки до стадії експлуатації. Такий підхід передбачає командну роботу та участь у процесі великої кількості фахівців різного профілю: від архітекторів і розробників до дизайнерів і замовників проекту.

Консолідацію всіх циклів роботи над додатком і взаємодія робочої групи в Visual Studio забезпечує вдосконалене рішення Team Foundation Server. З його допомогою всі учасники процесу розробки можуть відстежувати стан проекту, бачити його динаміку, контролювати терміни і отримувати аналітичні звіти про кожному періоді роботи. Крім того, Visual Studio містить оновлені інструменти перевірки якості і працездатності додатка, що дозволяє тестувальникам моделювати поведінку програми в момент його використання, а також вчасно виявляти недоліки в розробці. А функція PowerPoint StoryBoarding дозволяє технічним фахівцям представляти макет майбутнього рішення замовнику в зрозумілому для нього форматі.

Visual Studio 2012 дозволяє створювати сучасні, конкурентоспроможні програми для різних пристроїв і середовищ, але найбільш ефективний продукт при інтеграції з платформою Windows, зокрема з новою операційною системою Windows 8, мобільною платформою Windows Phone і платформою Windows Azure.

**3.2 NET FRAMEWORK 4.5**

Майже всі можливості .NET Framework доступні через величезну кількість управляючих типів. Ці типи організовані в ієрархічний простір імен і упаковані в набір збірок, які разом із середовищем CLR складають платформу .NET. Деякі з типів .NET використовуються безпосередньо CLR і є критично важливими для середовища керованого розміщення. Ці типи знаходяться в збірці mscorlib.dll і включають вбудовані типи C#, а також базові класи колекцій, типи для обробки потоків даних, серіалізації, рефлексії, багато потоковості і власної можливості взаємодії (mscorlib являє собою абревіатуру від Multi-language Standard Common Object Runtime Library (стандартна багатомовна загальна об'єктна бібліотека часу виконання)).

На рівні вище цього знаходяться додаткові типи, які розширюють функціональність рівня CLR, надаючи такі засоби, як XML, робота в мережі і LINQ. Вони знаходяться в System.dll, System.Xml.dll і System.Core.dll, і разом з mscorlib.dll формують розвинене середовище для програмування, на основі якому побудовані інші частини .NET Framework.

.NET Framework складається з прикладних API-інтерфейсів, більшість з яких покривають три області функціональності:

* технології користувальницьких інтерфейсів;
* технології серверної частини;
* технології розподілених систем.

Нижче перераховані нові функціональні можливості .NET Framework 4.5.

* Покращена підтримка асинхронності через методи повернення із задач.
* Підтримка протоколу стиснення ZIP.
* Покращена підтримка HTTP через новий клас HttpClient.
* Поліпшення показників продуктивності для збірки сміття та вилучення ресурсів збірки.
* Підтримка взаємодії WinRT і API-інтерфейсів для побудови планшетних додатків в стилі Metro.

З'явилися також новий клас TypeInfo і можливість вказівки тайм-аутів при зіставленні з регулярними виразами.

В області паралельних обчислень стала доступною нова бібліотека на ім'я Dataflow, призначена для побудови мереж у стилі постачальник / споживач.

Внесено також удосконалення в бібліотеки WPF, WCF і WF (Workflow Foundation).

**3.3 Середовище CLR і ядро ​​платформи**

**Системні типи.**

Найбільш фундаментальні типи знаходяться безпосередньо в просторі імен System. У їх число входять вбудовані типи C#, базовий клас Exception, базові класи Enum, Array і Delegate, а також Nullable, Type, DateTime, TimeSpan і Guid. Простір імен System також включає типи для виконання математичних функцій (Math), генерації випадкових чисел (Random) і перетворення між різними типами (Convert і BitConverter).

**Паралелізм і асинхронність.**

Більшості сучасних додатків доводиться мати справу з більш ніж однією річчю, яка відбувається в один і той же момент часу. Мова C # 5.0 і платформа .NET Framework 4.5 роблять це простіше, ніж було раніше, через асинхронні функції і такі високорівневі конструкції, як завдання і комбінатори завдань. Типи для роботи з потоками і асинхронними операціями знаходяться в просторах імен System.Threading і System.Threading.Tasks.

**Потоки даних і введення-виведення.**

Платформа .NET Framework надає потокову модель для низькорівневого вводу - виводу. Зазвичай для читання і запису безпосередньо у файли і мережеві підключення використовуються потоки даних, які можуть бути з'єднані або поставлені всередину декорованих потоків для додавання функціональності стиснення або шифрування. Тип Stream і типи вводу-виводу .NET визначені в просторі імен System.IO і його підпросторах, а типи WinRT для файлового введення-виведення - у просторі імен Windows.Storage і його підпросторах.

**Робота з мережами.**

За допомогою типів в просторі імен System.Net можна безпосередньо працювати зі стандартними мережевими протоколами, такими як HTTP, FTP, TCP / IP і SMTP.

**Серіалізация.**

В .NET Framework надається декілька систем для збереження і відновлення об'єктів в бінарному або текстовому поданні. Такі системи є обов'язковими в технологіях розподілених додатків, таких як WCF, Web Services і Remoting, а також використовуються для збереження і відновлення об'єктів у файлі.

Типи, пов'язані з серіалізацією, знаходяться в таких просторах імен:

* System.Runtime.Serialization
* System.Xml.Serialization

**Збірки, рефлексія і атрибути.**

Збірки, в які компілюються програми на C#, складаються з виконуваних інструкцій (представлених на проміжній мові (intermediate language - IL)) і метаданих, що описують типи, члени та атрибути програми. За допомогою рефлксії можна переглядати метадані під час виконання і робити такі речі, як динамічний виклик методів. За допомогою Reflection.Emit можна конструювати новий код на льоту.

Типи, призначені для рефлексії і роботи зі збірками, знаходяться в таких просторах імен:

* System
* System.Reflection
* System.Reflection.Emit

**Розширена багатопотоковість.**

Асинхронні функції C# 5 значно спрощують паралельне програмування, оскільки вони зменшують потребу в роботі з низькорівневими технологіями. Тим не менш, все ще виникають ситуації, при яких потрібні сигнальні конструкції, локальне сховище потоку, блокування читання / запису і т.п.

Типи, що відносяться до багатопотоковості, знаходяться в просторі імен System.Threading.

**3.4 Прикладні технології**

Платформа .NET Framework пропонує чотири API-інтерфейси для застосування при побудові додатків з інтерфейсом.

* ASP.NET (System.Web.UI). Призначений для написання додатків клієнтів, які виконуються в стандартних веб-браузерах.
* Silverlight. Призначений для побудови розширених користувальницьких інтерфейсів всередині веб-браузера.
* Windows Presentation Foundation (System.Windows).
* Windows Forms (System.Windows.Forms).

**ASP.NET**

Програми, написані з використанням ASP.NET, розташовуються на сервері Windows IIS і можуть бути доступні за допомогою майже всіх веб-браузерів. Нижче перечисельні переваги ASP.NET:

* Відсутність потреби в розгортанні на клієнтській стороні.
* Клієнти можуть використовувати платформи, відмінні від Windows.
* Просте розгортання оновлень.

Крім того, оскільки більша частина коду, який доводиться писати в додатку ASP.NET, виконується на сервері, рівень доступу до даних проектується для виконання в тому ж самому домені додатка - без обмеження безпеки або масштабованості.

При написанні своїх веб-сторінок можна вибирати між традиційним API-інтерфейсом Web Forms і новим API-інтерфейсом MVC (Model-View-Controller - модель-представлення -контролер). Обидва вони побудовані на основі інфраструктури ASP.NET. Технологія Web Forms була частиною .NET Framework з самого початку, а MVC реалізована набагато пізніше як реакція на успіх Ruby on Rails і MonoRail.

Інфраструктура MVC з'явилася в .NET Framework 4.0 і з тих пір сильно розвинулась. В цілому вона надає кращу програмну абстракцію, ніж Web Forms; вона також дозволяє мати більший контроль над згенерованою HTML-розміткою. Однак є один аспект, у якому MVC програє Web Forms - візуальний конструктор. Це зберігає Web Forms в якості гарного засобу для побудови веб-сторінок з переважно статичним вмістом.

Обмеження ASP.NET:

* інтерфейс веб-браузера істотно обмежує те, що можна робити;
* підтримка стану на стороні клієнта (або від імені клієнта) є громіздкою.

Тим не менш, інтерактивність і чуйність можна поліпшити за допомогою сценаріїв на клієнтської стороні або технологій на зразок AJAX: хорошим ресурсом для цього є веб-сайт http://ajax.asp.net. Робота з AJAX спрощується за рахунок використовування таких бібліотек, як jQuery.

Типи, призначені для написання додатків ASP.NET, знаходяться в просторі імен System.Web.UI і його підпросторах; вони упаковані в збірку System.Web.dll.

**3.5 Технології серверної частини**

ADO.NET - це керований API-інтерфейс доступу до даних. Хоча назва походить від застосованої в 1990-х роках технології ADO (ActiveX Data Objects - об'єкти даних ActiveX), технологія ADO.NET зовсім інша. ADO.NET містить два основних низькорівневих компонента.

* Рівень постачальників. Модель постачальників визначає загальні класи та інтерфейси для низькорівневого доступу до постачальників баз даних. Ці інтерфейси складаються з підключень, команд, адаптерів і засобів читання (односпрямованих курсорів, призначених тільки для читання, в базі даних). Платформа .NET Framework поставляється з власною підтримкою Microsoft SQL Server і Oracle, а також має постачальники OLEDB і ODBC.
* Модель DataSet. DataSet - це структурований кеш даних. Він схожий на примітивну базу даних у пам'яті, яка визначає такі SQL-конструкції, як таблиці, рядки, стовпці, відносини і представлення. За рахунок програмування для кеша даних можна скоротити кількість звернень до сервера, покращуючи показники масштабованості сервера і чуйності інтерфейсу.

DataSet підтримує серіалізацію і можливість передачі по мережі між клієнтськими і серверними додатками.

Поверх рівня постачальників знаходяться два API-інтерфейси, які надають можливість запитуватись до бази даних за допомогою LINQ:

• LINQ to SQL (з'явився в .NET Framework 3.5);

• Entity Framework (з'явився в .NET Framework 3.5 SP1).

Обидві технології включають об'єктно-реляційні образи (object / relational mapper - ORM), які автоматично відображають об'єкти (засновані на реалізованих класах) на рядки в базі даних. Це дозволяє запитувати такі об'єкти із застосуванням LINQ (замість написання SQL-операторів select) і обновлять їх без написання вручну SQL-операторів insert / delete / update. У результаті скорочується обсяг коду на рівні доступу до даних в додатку (особливо допоміжного коду) і забезпечується сувора безпека статичних типів.

Ці технології також усувають потребу в наявності DataSet в якості сховищ даних, хоча DataSet досі пропонують унікальну можливість зберігання та серіалізації змін стану (те, що особливо корисно для багаторівневих додатків). Спільно з DataSet можна використовувати LINQ to SQL або Entity Framework, хоча цей процес дещо тяжкий зважаючи на незграбність самих DataSet. Іншими словами, поки що не існує очевидного готового рішення для написання n-рівневих додатків з ORM від Microsoft. LINQ to SQL простіше і швидше Entity Framework, до того ж виробляє найкращий SQL-код (незважаючи на удосконалення Entity Framework в останніх версіях). Технологія Entity Framework більш гнучка в тому, що дозволяє створювати точні відображення між базою даних і потребуючими класами, і пропонує модель, яка допускає підтримку від третіх сторін для баз даних, відмінних від SQL Server.

**3.6 Висновки**

Відкрита платформа .NET дозволяє досягти потрібної швидкодії, яка так важлива при створенні систем на основі нейронних мереж. Фреймворк ASP.NET WEB API 2, дозволяє створювати зручний програмний інтерфейс для використання клієнтськими додатками серверного коду системи, а технологія ADO.NET дозволяє доступатись до коефіцієнтів ваг нейронних мереж і навчальної вибірки у базі даних практично миттєво. Це все підтверджує доцільність використання даних технологій, так як швидкодія це одне з слабких місць штучних нейронних мереж.

**4. РЕАЛІЗАЦІЯ**

**4.1 Призначення і загальні вимоги**

Призначення серверної частини системи збору, обробки, класифікації та прогнозування даних на основі нейронної мережі - в динамічній обробці запитів від користувачів через програмний інтерфейс. Така система має за мету адаптивно пристосовуватись до змін зовнішнього статистичного середовища і виконувати класифікацію з завжди оновленими даними.

Областю застосування даної адаптивної нейронної мережі може бути не тільки системи прогнозування діагнозу, а й люба система для обробки великої кількості статистичних даних (біржові системи, системи ехо-локації, системи розпізнавання зображень, тощо).

Основні вимоги:

* Швидкодія.
* Система повинна адаптивно пристосовуватись до зміни зовнішнього середовища
* Система повинна представляти зручний програмний інтерфейс для доступу до нейронної мережі, та сховища даних.

В реалізованій системі, клієнтом являється Android додаток, проте для програмного інтерфейсу немає значення яка саме це платформа. Так як адміністративна частина системи реалізована за допомогою фреймоврку ASP.NET MVC, як звичайна web – сторінка.

Користувач періодично надсилає запит до програмного інтерфейсу, з зібраними даними які трасформуються у вхідний вектор для нейронної мережі. Після чого нейронна мережа класифікує його відносячи дані до нормального стану користувача або ж до відповідного діагнозу. Після чого весь вхідний вектор подається на вхід іншій нейронній мережі яка навчена під кожен окремий діагноз, для визначеня ступені ризику. Діаграма спілкування з користувачем зображена на рисунку 4.1.1.

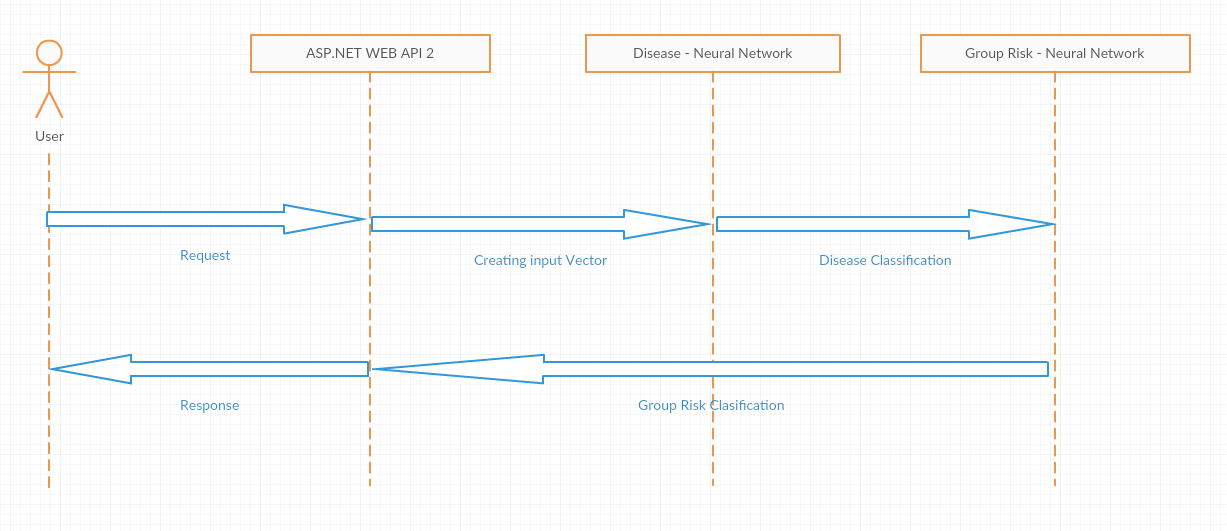


Рис.4.1.1 UML діаграма послідовності спілкування системи з користувачем

**4.2 Структура системи**

**Модель нейрона.**

Моделлю нейрона було вибрано звичайний тип нейрона з пороговим значенням рівному одиниці(Рис.4.2.1).

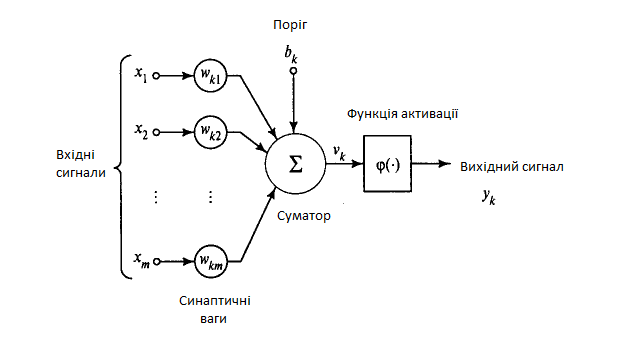


Рис.4.2.1 Вибрана модель нейрона.

**Функція активації.**

Функцією активації було вибрано сігмоїдальну логістичну функцію.(Рис.4.2.2)

(4.2.1)

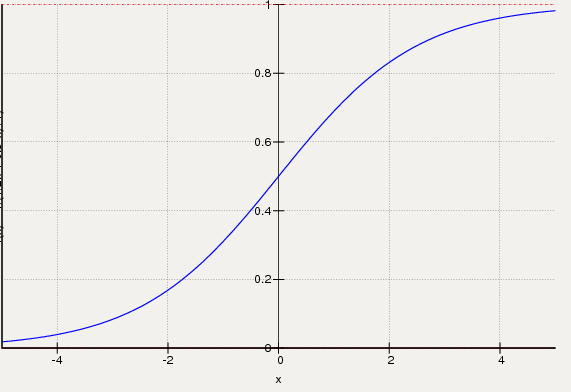


Рис.4.2.2 Функція активації.

**Архітектура мережі.**

Вхідний вектор мережі включає в себе 21 нормальнізоване значення. Тому кількість входів рівне 21, також в нейронній мережі пристуні 2 приховані шари по 42 нейрона,і кількість виходів які залежать від кількості класів.

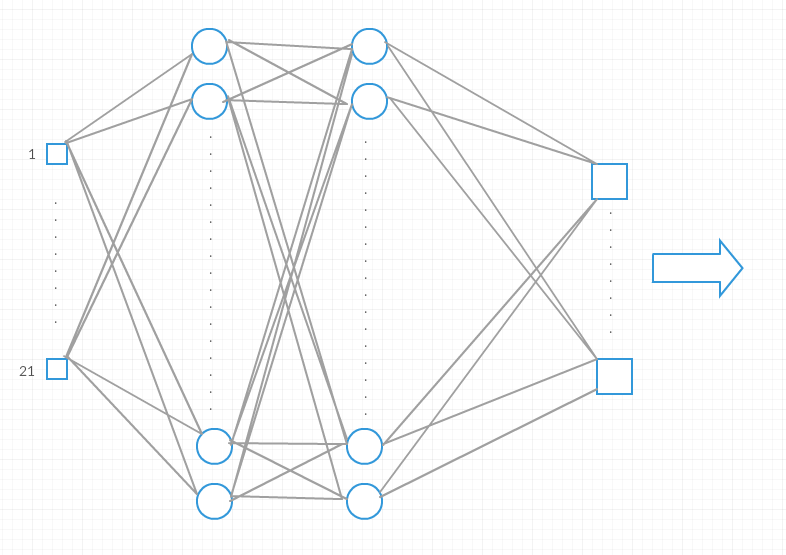


Рис.4.2.3 Вибрана архітектура нейронної мережі.

Компонентна діаграма системи зображена на рисунку 4.2.4.

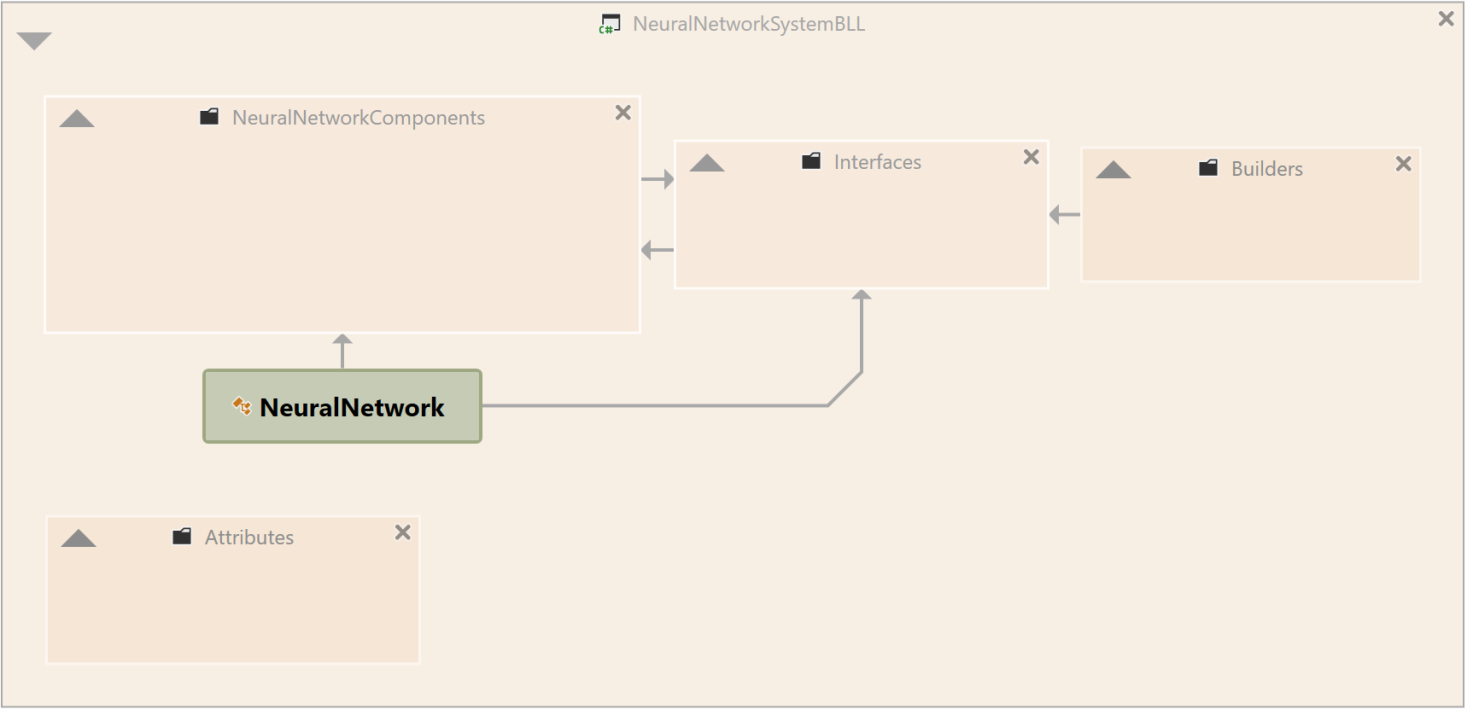


Рис.4.2.4. Компонентна діаграма системи

Структура проекту клієнтської бібліотеки для створення нейронної мережі зображено на рисунку 4.2.5.

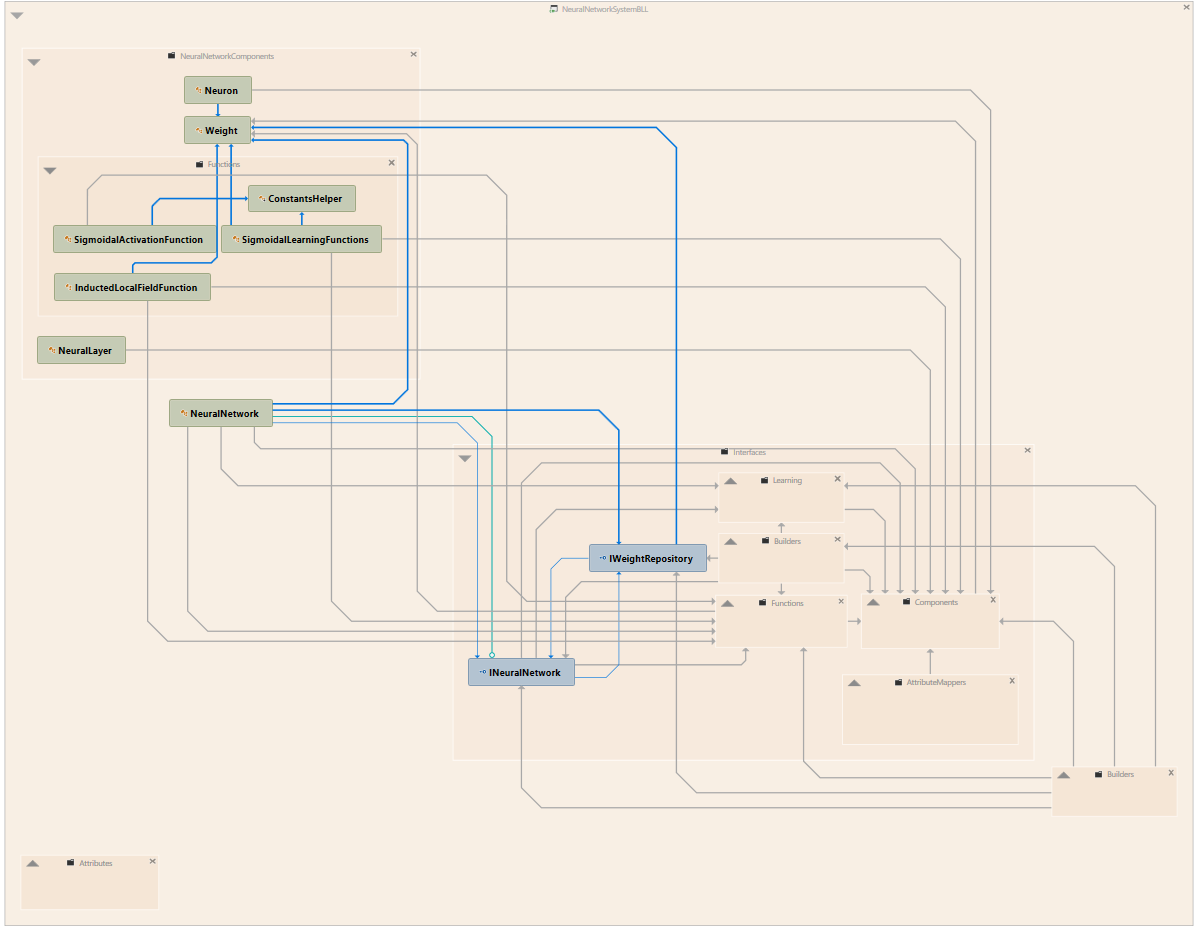


Рис.4.2.5 Структура клієнтської бібліотеки.

Клієнтська бібліотека класів включає в себе інтерфейси і компоненти для створення нейронної мережі.Також вона містить стандартні компоненти для побудови архітектури мережі. В даному випадку було використано пат терн «Будівельник».

**4.3 База даних.**

Для сховища даних було вибрано MS SQL SERVER 2014. Для доступу до сховища було вибрано технологію ADO.NET. Схема бази даних включає в себе навчальну вибірку, групи ризику, типи прогнозів, а також значення параметрів вхідного вектора для різного типу групи ризиків.Також в базі даних зберігаються поради щодо покращення результатів(Рис.4.3.1).



Рис.4.3.1 Схема бази даних.

**5. ЕКОНОМІЧНА ЧАСТИНА**

* 1. **Економічна характеристика проектного рішення**

Метою магістерської роботи є розроблення серверної частини системи збору, обробки, класифікації та прогнозування статистичних даних. Система повинна дати змогу адаптивно, без людського втручання обробляти статистичні дані, змінювати конфігурацію системи, та динамічно навчати нейронну мережу на основі якої система і будується.

Дана система розробляється для широкого кола підприємств, які в ході своєї діяльності використовують велику кількість статистичних даних.

Нейронна мережа, на основі якої будується система, буде мати змогу без переконфігурування, динамічно навчатись на основі відомих статистичних даних підприємства і таким чином пристосовуватись до середовища.

Нейронні мережі в сучасному світі використовувались для незначних завдань, проте завжди мали великий потенціал. Тепер коли комп’ютерні технології пішли вперед, і таким мережам є достатньо швидкодії, їх доцільно використовувати в побуті.

Від розробленого проекту очікується фінансовий прибуток за рахунок бажання підприємств використовувати дане середовище в своїй діяльності. Системи на основі нейронних мереж тільки почали набувати популярності, тому дана система дасть змогу отримати вагому частку на ринку збуту.

* 1. **Розрахунок витрат на розробку і впровадження проектного рішення**

Витрати на розробку і впровадження програмного засобу (*К*) визначаються як:

 (5.2.1)

де – витрати на розробку програмного засобу, грн.;

 – витрати на відлагодження і дослідну експлуатацію програмного засобу на ЕОМ, грн.

Витрати на розробку програмного засобу включають в себе:

1. витрати на оплату праці розробників ();
2. єдиний соціальний внесок ();
3. вартість додаткових виробів, що закуповуються ();
4. накладні витрати ();
5. інші витрати ().

У Таблиці 4.1 наведено розрахунок витрат на оплату праці.

*Таблиця 5.2.1*

Розрахунок витрат на оплату праці

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Спеціальність розробника | Кількість розробників, чол. | Час роботи, дні | Денна заробітна плата розробника, грн. | Витрати на оплату праці, грн. |
| Керівник МР | 1 | 20 | 120 | 2400 |
| Консультант з економіки | 1 | 2 | 105,58 | 211,16 |
| Рецензент | 1 | 1 | 100 | 100 |
| Студент | 1 | 60 | 25 | 1500 |
| Разом: | 4 | 83 | 350, 58 | 4211,16 |

Розрахунок витрат на куповані вироби представлений у Таблиці 5.2.2 У тому числі транспортно-заготівельні витрати становлять 10-15 % (у даному випадку – 10%) суми витрат на додаткові вироби, що закуповуються.

*Таблиця 5.2.2*

Розрахунок витрат на куповані вироби

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | Найменування купованих виробів | Марка, тип | Кількість на розробку, шт. | Ціна за одиницю, грн. | Сума витрат, грн. | Сума витрат з урахуванням транспортно-заготівельних витрат, грн. |
| 1 | Процесор | IntelCore i7-5500U (2.4ГГц) | 1 | 3000,00 | 3000,00 | 3300 |
| 2 | Монітор | DELL P2414H | 1 | 4200 | 4200 | 4620 |
| 3 | Flash-пам’ять USB | Transcend 16GB | 1 | 180,00 | 180,00 | 198 |
| 4 | Папір (формат А4) | Папір офісний Zoom 500 арк./уп. | 1 | 80 | 80 | 88 |
| *Всього* | | | | | | 8206 |

Слід зазначити, що витрати на оплату праці працівникам тягнуть за собою сплату єдиного соціального внеску, ставка якого залежить від класу професійного ризику розробників програмного забезпечення та визначатиметься у відсотковому співвідношенні від фонду оплати праці. В даному випадку ставка становить 36,77% (2 клас):

Накладні витрати (*)* проектних організацій включають витрати на управління, загальногосподарські, невиробничі витрати. Вони становлять 20-30 % витрат на оплату праці:

Інші витрати (*)* – це витрати, які не враховані в попередніх статтях витрат. Вони розраховуються за встановленими відсотками (5-12 %) до витрат на оплату праці. Становлять 7%:

Витрати на розробку проектного рішення обчислюються за формулою:

 (5.2.2)

Витрати на відлагодження і дослідну експлуатацію системи визначаються згідно формули:

 (5.2.3)

де – вартість однієї години роботи ПК, грн./год.

 – кількість годин роботи ПК на відлагодження програми, год.

Загальна кількість днів роботи на ЕОМ рівна 60 днів. Середній щоденний час роботи на ЕОМ - 5 год., тому:

За даними обчислювального центру НУ “Львівська Політехніка” для ЕОМ типу IBMPC / ATSMг = 3 грн.

Звідси:

.

Отже, розрахунок витрат на розробку та впровадження проектного рішення наведений в Таблиці 5.2.3

*Таблиця 5.2.3*

Кошторис витрат на розробку проектного рішення

|  |  |
| --- | --- |
| Найменування елементів витрат | Сума витрат, грн. |
| Витрати на розробку проектного рішення, у т.ч.: | 15228,95 |
| витрати на оплату праці | 4211,16 |
| сплата єдиного соціального внеску |  |
| витрати на додаткові вироби, що закуповуються | 8206 |
| накладні витрати |  |
| інші витрати |  |
| Витрати на відлагодження і дослідну експлуатацію системи | 900,00 |
| *Всього* | 16128,95 |

* 1. **Визначення комплексного показника якості**

Зростаюча складність (ПЗ) призводить до збільшення кількості помилок у ньому, а одночасне зростання кількості й критичності виконуваних ним функцій тягне за собою збільшення збитків від цих помилок. Тому завдання оцінювання його якості є актуальним.

Для вирішення проектної задачі за аналог обрано варіант ручного виконання функцій, оскільки системи на основі нейронних мереж створються для симулювання штучного інтелекту і автоматизації роботи яку виконує людина.

Вибір номенклатури показників якості програмної продукції полягає у встановленні переліку найменувань характеристик властивостей продукції, які визначають якість даного виду продукції і забезпечують можливість повної і достовірної оцінки її рівня якості. Провівши аналіз проектного рішення та його аналогу – ручного варіанту виконання функцій, обрано наступні експлуатаційні показники для порівняння:

1. Показники призначення (функціональності): функціональна повнота; правильність; сумісність.
2. Показники надійності: безвідмовність, стійкість до помилок;
3. Зручність застосування: зрозумілість; легкість навчання; оперативність.
4. Супроводжуваність: стабільність, тестованість.

5. Інші показники (продуктивність): ефективність використання ресурсів.

*Комплексний показник* якості проектованої системи визначаємо методом арифметичного середньозваженого з формули:

 (5.3.1)

де  - кількість одиничних показників (параметрів), прийнятих для оцінки якості проектованої системи;

- коефіцієнт вагомості кожного з параметрів щодо їхнього впливу на технічний рівень та якість проектованої системи (встановлюється експертним шляхом), причому:

 (5.3.2)

 *-* часткові показники якості, визначені порівнянням числових значень одиничних показників проектованої системи і аналога за формулами:

 або , (5.3.3)

де ,  - кількісні значення і-го одиничного показника якості відповідно проектованої системи і аналога.

У Таблиці 5.3.1. наведено визначення комплексного показника якості проектованої системи.

*Таблиця 5.3.1*

Визначення комплексного показника якості проектованої системи (аналога)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Показники | Числове значення показників, бали | | Відносний показник якості, Сі | Коефіцієнт вагомості  qi | Ci ×qi |
| Аналог | Проект. прогр. продукт |
| 1. Показники призначення |  | | | | |
| * функціональна повнота; | 2 | 8 | 4,0 | 0,05 | 0,2 |
| * правильність; | 6 | 9 | 1,5 | 0,05 | 0,075 |
| * сумісність. | 2 | 7 | 3,5 | 0,05 | 0,175 |
| 1. Показники надійності |  | | | | |
| * Безвідмовність | 6 | 6 | 1 | 0,2 | 0,2 |
| * Стійкість до помилок | 6 | 7 | 1,2 | 0,15 | 0,18 |
| 1. Зручність застосування |  | | | | |
| * зрозумілість; | 7 | 6 | 0,86 | 0,1 | 0,086 |
| * легкість навчання; | 2 | 10 | 5 | 0,05 | 0,25 |
| * оперативність. | 3 | 10 | 3,3 | 0,05 | 0,165 |
| 1. Супроводжуваність |  | | | | |
| * Стабільність. | 5 | 6 | 1,2 | 0,2 | 0,24 |
| * Тестованість. | 10 | 2 | 0,2 | 0,05 | 0,01 |
| 1. Продуктивність |  | | | | |
| * ефективність використання ресурсів. | 2 | 9 | 4,5 | 0,05 | 0,225 |
| *Всього* | – | | | 1,0 | **1,82** |

**5.4 Визначення експлуатаційних витрат**

При порівнянні програмних засобів в експлуатаційні витрати включають вартість підготовки даних () і вартість годин роботи ПК (). Одноразові експлуатаційні витрати визначаються за формулою:

(5.4.1)

де - одноразові експлуатаційні витрати на проектне рішення, грн.;

- вартість підготовки даних для експлуатації проектного рішення, грн.;

- вартість машино-годин роботи ПК для проектного рішення, грн.

Річні експлуатаційні витрати визначаються за формулою:

(5.4.2)

де – експлуатаційні річні витрати проектного рішення, грн.;

- періодичність експлуатації проектного рішення, разів/рік.

Вартість підготовки даних для експлуатації проектного рішення () визначаються за формулою:

 (5.4.3)

де  – номери категорій персоналу, які беруть участь у підготовці даних;

– кількість співробітників і-ї категорії, чол.;

– трудомісткість роботи співробітників і-ї категорії, чол.;

– середньогодинна ставка робітника і-ї категорії з врахуванням сплати єдиного соціального внеску, грн./год.

Середньогодинна ставка оператора визначається за формулою:

 (5.4.4)

де – основна місячна зарплата працівника і-ї категорії, грн.;

– коефіцієнт, який враховує сплату єдиного соціального внеску (його ставка повинна відповідати значенню, визначеному у п. 2 економічної частини);

 – місячний фонд робочого часу, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає 2000 грн.

Трудоємність працівника по підготовці даних для проектного рішення складає 1 год., для аналога – 5 год.

Підставивши відповідні значення у формули, отримаємо:

Розрахунок витрат на підготовку даних для роботи на ЕОМ представлено в Таблиці 5.4.1.

*Таблиця 5.4.1*

Розрахунок витрат на підготовку даних для роботи на ЕОМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категорія персоналу | Чисельність співробітників і-ої категорії, чол. | Час роботи співробітників і-ої категорії, год. | Середньогодинна ЗП співробітника і-ої категорії, грн. | Витрати на підготовку даних, грн. |
| Проектне рішення | | | | |
| 1-ша | 1 | 1 | 11,91 | 11,91 |
| *Всього* | 1 | 1 | 11,91 | 11,91 |

У випадку ручного виконання функцій (для аналогу) одноразові витрати на них () визначаються за формулою:

 (5.4.5)

де  - трудоємність обробки інформації ручним способом при одноразовому виконанні функцій, люд./год.;

 - коефіцієнт, який враховує додаткову трудомісткість обробки інформації на додаткових операціях (від 2 до 3; у даному випадку 2);

 – середньогодинна ставка персоналу певної категорії, грн./год.

Трудомісткість обробки інформації ручним способом розраховано в Таблиці 5.4.2.

*Таблиця 5.4.2*

Розрахунок трудомісткості виконання автоматизованих функцій традиційним методом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Операції по виконуваних фукціях | Спеціальність і розряд виконавців | Чисельність виконавців по операціях, чол. | Час, що витрачається виконавцем на операцію, год. | Трудоємність виконання операцій Dоі, люд./год. |
| Розрахунок | 1-ий розряд | 1 | 5 | 5 |

Таким чином:

**5.5 Розрахунок ціни споживання проектного рішення**

Ціна споживання () – це витрати на придбання і експлуатацію проектного рішення за весь строк його служби. Визначається за формулою:

 (5.5.1)

де – ціна придбання проектного рішення, грн.;

– теперішня вартість витрат на експлуатацію проектного рішення (за весь час його експлуатації), грн.:

(5.5.2)

де – норматив рентабельності;

– витрати на прив'язку та освоєння проектного рішення на конкретному об’єкті, грн.;

– витрати на доукомплектування технічних засобів на об'єкті, грн.

Теперішня вартість витрат на експлуатацію проектного рішення розраховується за формулою:

де - річні експлуатаційні витрати в t-ому році, грн.;

 - строк служби проектного рішення, років (5 років);

 - річна ставка проценту банків.

Виконавши відповідні розрахунки, для проектного рішення отримаємо:

Отже, ціна споживання проектного рішення:

Оскільки за аналог взято ручне виконання функцій, то .

Отже, ціна споживання аналогу:

**5.6 Визначення показників економічної ефективності**

Оскільки базою для порівняння обрано ручне виконання функцій, розраховуються такі показники:

1. Показник конкурентоспроможності:
2. Економічний ефект в сфері експлуатації:
3. Економічний ефект в сфері проектування (грн.):
4. Додатковий економічний ефект в сфері експлуатації (грн.):

Оскільки Епр<0, то додатковий економічний ефект у сфері проектування не розраховується.

1. Термін окупності витрат на проектування рішення:

Результуючі показники економічної ефективності подано в Таблиці 5.6.1.

*Таблиця 5.6.1*

Показники економічної ефективності проектного рішення

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування показників | Одиниці вимірювання | Значення показників | |
| Аналог | Проектне рішення |
| 1. Капітальні вкладення | грн. | - |  |
| 2. Ціна придбання | грн. | 0 |  |
| 3. Річні експлуатаційні витрати | грн. |  |  |
| 4. Ціна споживання | грн. |  |  |
| 5. Економічний ефект в сфері експлуатації | грн. | - |  |
| 6. Додатковий економічний ефект в сфері експлуатації | грн. | - |  |
| 7. Економічний ефект в сфері проектування | грн. | - |  |
| 8. Додатковий економічний ефект в сфері проектування | грн. | - | - |
| 9.Термін окупності витрат на проектування рішення | місяці | - | 5 |
| 10.Коефіцієнт конкурентоспроможності |  | - |  |

**5.7 Висновки**

В економічній частині магістерської роботи було проведено розрахунок витрат на розробку програмного продукту, а саме системи для збору, обробки, аналізу та прогнозування статистичних даних.. Проведено економічні розрахунки для відображення доцільності розробки програмно-апаратного комплексу. Здійснено порівняння з аналогом – варіантом ручного виконання функцій, які автоматизуються в розроблених програмних засобах.

Результати економічних обчислень показали, що дане проектне рішення має переваги в порівнянні з ручним виконанням функцій. Коефіцієнт конкурентоздатності становить 0,5<1, тому проектне рішення не конкурентоспроможне, проте має ряд переваг, таких як додатній економічний ефект, що становить 36789.48грн. та очікуваний термін окупності витрат, що становить 5 місяців. У сфері експлуатації очікується додатковий економічний ефект 273713,78 грн.

**ВИСНОВКИ.**

В даній роботі вирішені наступні завдання: виявилено всі переваги використання нейронних мереж як класифікатора в системах обробки великої кількості статистичних даних. Спроектовано та створено серверну частину системи прогнозування та моніторингу стану пацієнта.

Основної функцією нейронної мережі в даній системі цекласифікація вхідного вектору який включає в себе дані про користувача.

В роботі було запропоновано вирішення проблеми адаптивності нейронних мереж, та було доведено доцільність такого підходу.

Аналіз показує що використання нейронних мереж ходь і потребує швидкодії і часу на навчання проте при класифікації вхідного вектора швидкодія нейронної мережі перевищує будь який алгоритм кластеризації.

Дану систему рекомендовано використовувати там, де зовнішнє статистичне середовище є стохастичним і не статичним, так як дана модель системи дозволяє навчати нейронну мережу на ходу і уникнути ще однієї проблеми – перенавчення нейронної мережі.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

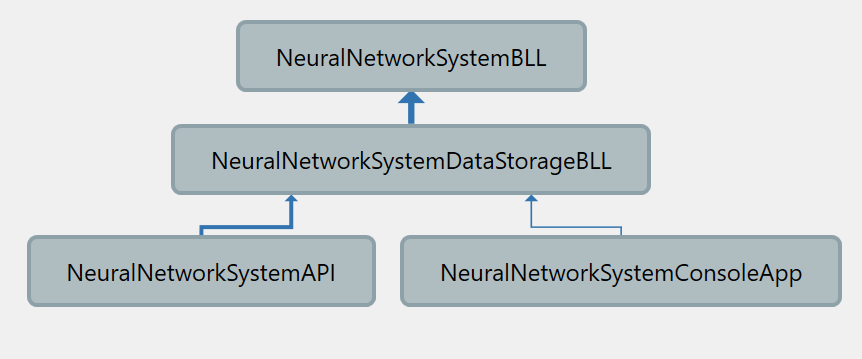
# ASP.NET MVC 4 Framework(Фрімен А., Сандерсон С.). Apress; 4 edition.

1. ASP.NET MVC 4( Джефрі Палермо). Manning Publications; Third Edition edition (June 5, 2012)
2. https://msdn.microsoft.com/-Мережа розробників Майкрософт
3. http://metanit.com/sharp/tutorial/13.3.php
4. http://www.sansys.net/2013/05/asynchronous-programming-in-csharp-using-async-and-await.html
5. http://www.dotnetperls.com/async
6. https://nesteruk.wordpress.com/2010/10/31/async-await-csharp5/
7. Concurrency in C# CookBook(Stephen Cleary). O'Reilly Media; 1 edition
8. **Нейронные сети, генетические алгоритмы и нечеткие системы.** Д. Рутковская, М. Пилиньский, Л. Рутковский
9. **Fundamentals of Artificial Neural Networks.** Mohamad H. Hassoun

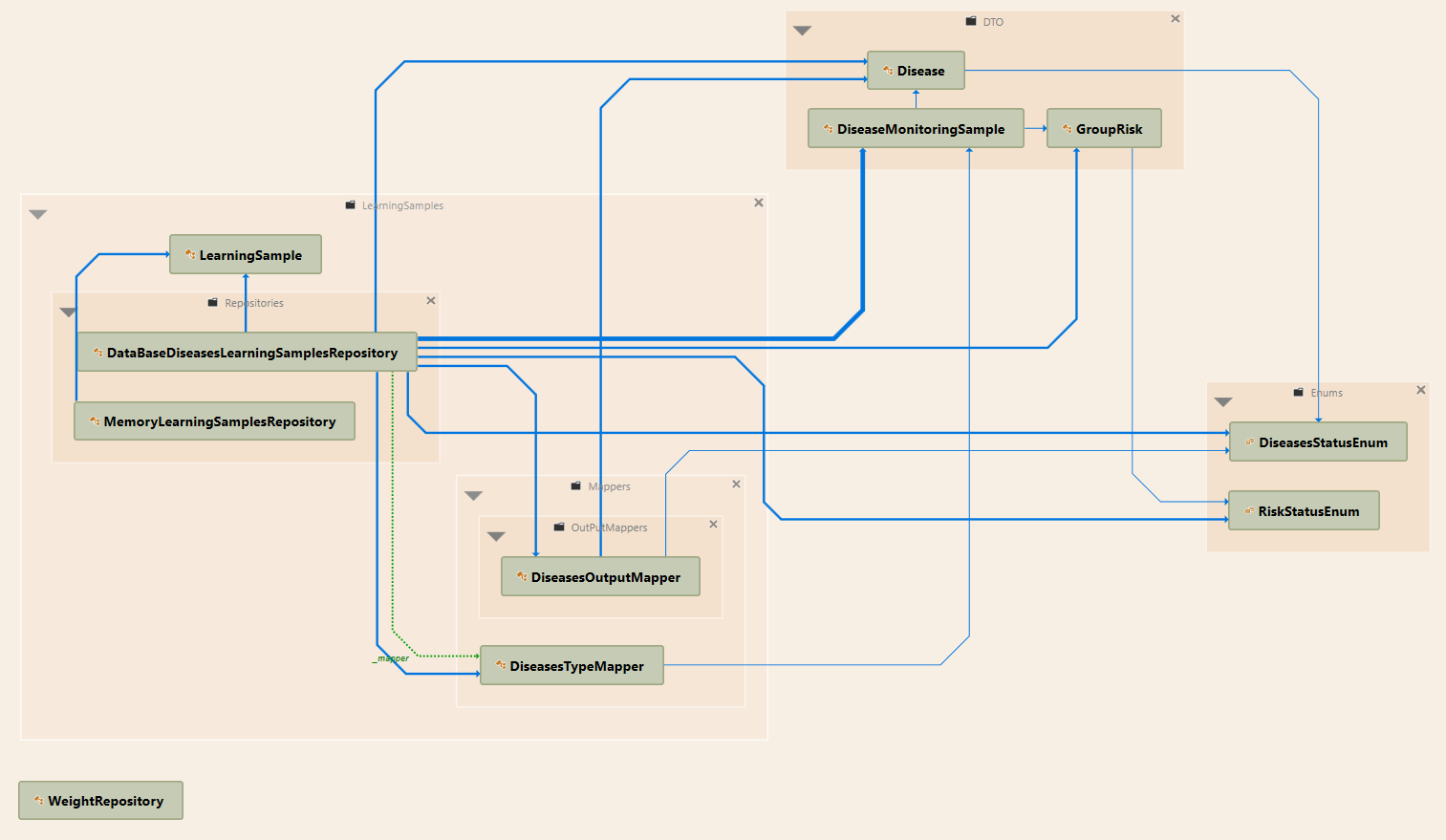
**ДОДАТКИ**

**Додаток А.**

Діаграма залежності проектів системи.

****

**Додаток Б.**

**Діаграма залежності проекту доступу до сховища даних.**