# 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 будь-які задачі «навчання з вчителем».

В першому 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.

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

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, 18 літературних джерел, та 2 додатка.

ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, .NET, SQL, ASP NET WEBAPI, ADO .NET.

**Abstract**

# In this thesis qualifying work considers the creating of neural network for server side of the systemprocessing 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 forprognostication 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 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 second chapter of this thesis, analyzes the possibilities of neural networks, types of activation functions and types of architectures.

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, 18 literary sources and 2 of Appendix.

ARTIFICIAL NEURAL NETWORKS, MACHINE LEARNING, .NET, SQL, ASP NET WEBAPI, 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)

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. НЕЙРОННІ МЕРЕЖІ. ПРОЦЕСИ МАШИННОГО НАВЧАННЯ. 11](#_Toc359262840)

[1.1 Аналіз предметної області. 11](#_Toc359262841)

[1.2 Огляд літературних джерел. 13](#_Toc359262841)

[1.3 Аналіз існуючих проектних рішень на основі нейронної мережі. 16](#_Toc359262841)

[1.4 Машинне навчання та нейронні мережі. 21](#_Toc359262842)

[1.5 Навчання з учителем. 22](#_Toc359262842)

[1.6 Навчання без учителя 24](#_Toc359262843)

[1.7 Задача присвоєння коефіцієнтів довіри. 27](#_Toc359262844)

[1.8 Задачі навчання 28](#_Toc359262845)

[1.9 Пам'ять інтелектуальних систем 32](#_Toc359262846)

[1.10 Штучний інтелект і нейронні мережі. 35](#_Toc359262846)

[1.10 Висновки 40](#_Toc359262846)

[2. КОМПОНЕНТИ НЕЙРОННОЇ МЕРЕЖІ. 43](#_Toc359262824)

[2.1 Модель нейрона. 43](#_Toc359262825)

[2.2 Архітектура мереж 47](#_Toc359262826)

[2.3 Багатошаровий персептрон. 49](#_Toc359262827)

[2.4 Навчання на основі корекції помилок.. 53](#_Toc359262828)

[2.5 Представлення знань. 57](#_Toc359262829)

[2.6 Адаптація нейронних мереж. 58](#_Toc359262833)

[2.7 Висновки 60](#_Toc359262830)

[3. ИСТЕМНИЙ АНАЛІЗ ПРОЕКТНОГО РІШЕННЯ 61](#_Toc359262850)

[3.1 Призначення системи та загальні вимоги 61](#_Toc359262851)

[3.2 Структура системи 62](#_Toc359262851)

[3.3 Обгрунтування засобів реалізації 65](#_Toc359262851)

[3.3.1 Microsoft Visual Studio 65](#_Toc359262851)

[3.3.2 .NET Framework 4,5 66](#_Toc359262852)

[3.3.3 Середовище CLR і ядро платформи 67](#_Toc359262853)

[3.3.4 Прикладні технології 69](#_Toc359262854)

[3.3.5 Технології серверної частини 71](#_Toc359262855)

[3.4 Висновки 72](#_Toc359262855)

[4. РЕЗУЛЬТАТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 73](#_Toc359262850)

[4.1 База даних 73](#_Toc359262852)

[4.2 Програмні компоненти системи 74](#_Toc359262853)

[4.3 Результати виконання 76](#_Toc359262853)

[4.4 Висновки 81](#_Toc359262853)

[5. ЕКОНОМІЧНА ЧАСТИНА 82](#_Toc359262858)

[5.1 Економічна характеристика проектного рішення 82](#_Toc359262859)

[5.2 Розрахунок витрат на розробку і впровадження проектного рішення 82](#_Toc359262860)

[5.3 Визначення комплексного показника якості 86](#_Toc359262861)

[5.4 Визначення експлуатаційних витрат 89](#_Toc359262862)

[5.5 Розрахунок ціни споживання проектного рішення 92](#_Toc359262862)

[5.6 Визначення показників економічної ефективності 93](#_Toc359262862)

[5.7 Висновки 94](#_Toc359262864)

[ВИСНОВКИ 96](#_Toc359262864)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 98](#_Toc359262864)

[ДОДАТКИ 99](#_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?

Людина може навчатись – це та можливість, якої комп'ютерам явно не вистачає. Люди мають мозок, який може вчитися. Комп'ютери містять в собі 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 секунди. Мозок містить 1011 нейронів, але вони мають 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

Більша частина мозку постійно працює, в той час як найбільша частина комп'ютера використовується тільки як пасивне сховище данних. Таким чином, мозок працює паралельно, 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.

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. **НЕЙРОННІ МЕРЕЖІ. ПРОЦЕСИ МАШИННОГО НАВЧАННЯ.**

**1.1Аналіз предметної області.**

Метою системи, основним компонентом якої є штучна нейрона мережа, є оцінка фізичного стану користувача на основі статистичних даних. Штучні нейронні мережі надають потужний механізм для допомоги лікарям розпізнавати та аналізувати клінічні дані. Проте точних методів діагностики досі немає. Так наприклад електрокардіограма деколи не містить явних ознак хвороби. Основним аргументом на користь застосування нейронних мереж є те, що існує величезна кількість параметрів, які можуть допомогти поставити діагноз і, нажаль, лікар не завжди може оцінити всі взаємозв’язки між характеристиками стану пацієнта. В дуже великій мірі, нейронна мережа може допомогти у цьому.

Статистика говорить, що лікар в стані поставити діагноз інфаркт міокарда у 88% хворих, проте у 29% цей діагноз є помилковим. Як бачимо, кількість «гіпердіагностики» є надзвичайно великий. За всю історію, було багато методів обробки даних, які застосовувались для покращення невтішної статистики, проте всі ці засоби змогли зменшити кількість неправильних діагнозів тільки на 3%.

Нейронна мережа являється прикладом нелінійної системи, вона дозволяє набагато краще класифікувати дані чим будь-які лінійні методи. Система на основі такого потужного механізму буде здатна приймати рішення засновуючись на прихованих закономірностях. Найважливішим плюсом такої системи, що вона не програмується, нейронна мережа це механізм, який навчається на основі величезної навчальної вибірки. Це відрізняє їх від любої експертної системи. Діагностика стану пацієнта являється тільки частковим випадком класифікації. Коли на карту поставлене людське здоров’я або навіть життя, проявляється найважливіша перевага нейронних мереж - здатність системи узагальнювати досвід отриманий при навчанні, і застосовувати його в випадках які до того не зустрічались.

Ще одним застосуванням нейронних мереж являється можливість прогнозування даних. Тут нейронні мережі можуть принести велику користь в обробці даних ЕКГ. Так наприклад, систему можна застосувати в виявленні так званих «шумів», на основі попереднього відрізку часу. Ще одним прикладом застосування прогнозування нейронних мереж є вибір плану лікування. На основі прогнозування динаміки окремих параметрів під час лікування можна вибрати той чи інший метод.

Нейромережі також можна застосовувати для прогнозування дії різних засобів лікування. Так вже є розробки в хімії та фармацепції по прогнозуванню властивості з’єднань на основі молекулярної структури. Так нейронні мережі можна використати в класифікації нових препаратів для визначення напрямку проведення дорогих експериментів.

Системи на основі нейронної мережі можна використовувати також при виявленні опухлі. Так наприклад встановлення раку молочної залози базується на 8 основних параметрах біопсії, і як говорить статистика від 10 до 20% результатів такої біопсії є невірними. Також діагностика деяких видів раку(наприклад меланома) ускладнюється тим що їх тяжко відрізнити від іншого типу опухлі, що часто призводить до неправильного лікування.

В наш час, серед лікарів появилась теорія, що діагностика і лікування онкологічних захворювань, а також розробка препаратів повинна бути пов’язана з дослідженням генетичних причин захворювання. Саме тому, наприклад Національний інститут здоров’я(США) виступають з рекомендаціями по збільшенню зусиль на вивчення причин які викликають рак. Тут нейромережеві структури можна використовувати для аналізу послідовностей ДНК, особливо для розпізнавання промоторів – спеціальних ділянок,зв’язаних з білком РНК-полімерази.

Система запропонована в даній роботі, слугує інструментом для відслідковування небажаної закономірності в стані користувача. Вона дозволить слідкувати за своїм станом здоров’я без походів в медичні заклади тощо.

Звичайно, перелік застосування обробки статистики в медицині є надзвичайно великим, сюди можна включити психіатрію, травматологію та багато інших розділів медицини де нейронна мережа може служити потужним помічником діагноста.

Звичайно дані інформаційні структури мають свої мінуси. І союз комп’ютерних технології і охорони здоров’я не є безхмарним. Впровадження нейронних мереж і систем на їх основі є вкрай дорогі(до десятків тисяч доларів), і як відомо, лікарі зачасту ставляться надзвичайно скептично до будь-яких інновацій, в Україні ця ситуація ще гірша чим у світі. Тому можна зробити висновок, що впровадження таких технологій мусить супроводжуватись великою кількістю пояснень і коментарів. Також такі системи, які впроваджуються в медицину повинні містити технологію пояснення своїх результатів.

Проте, освоєння та застосування нейромережевих структур є набагато простішими чим витрачати час на вивчення математичної статистики і нечіткої логіки. Для створення і впровадження такої системи потрібні не роки,а декілька місяців.

**1.2. Огляд літературних джерел.**

Дослідження нейронних мереж несуть свій початок в 60-тих роках минулого століття. Проте нейромережеві структури того часу існували в дуже примітивних видах або взагалі тільки в теорії. Це було пов’язано з недостатнім розвиток інформаційних технологій того часу. Проте роботи авторів того часу є актуальними і до сих пір, адже теоретичне обґрунтування лишилось незмінним.

Перш ніж приступати до реалізації нейромережевих структур варто звернути увагу на концепції машинного навчання.

«Машинне навчання. Наука і мистецтво побудови алгоритмів які витягують знання з даних» Пітера Флаха, являється один з найкращих підручників по машинному навчання - розділу штучного інтелекту, який вивчає методи побудови моделей, здатних навчатися, і алгоритмів для їх побудови. Автор віддав належне неймовірному багатству предмета і не випустив з уваги об'єднуючих принципів. Автор з перших сторінок занурює в машинне навчання, але без непотрібних на перших порах технічних деталей. У міру вивчення предмета представленні ретельно підібрані приклади, супроводжувані ілюстраціями, які поступово ускладнюються. У книзі описано широкий круг логічних, геометричних і статистичних моделей, зачіпаються і такі які знаходяться на передньому краї науки, такі як матрична факторизація і аналіз РХП. Особливу увагу приділено найважливішій ролі ознак. Усталена термінологія доповнюється введенням в розгляд нових корисних концепцій. В кінці кожного розділу наводяться посилання на додаткову літературу з авторськими коментарями. Книга ясно написана і добре організована. Почавши з основ, автор вміло веде читача, знайомлячи його з корисними фактами і докладно описуючи ряд методів машинного навчання. Наводиться також псевдокод ключових алгоритмів. Завдяки всьому цьому книга задає новий стандарт вивчення такої складної дисципліни як машинне навчання.

Ще однією книгою по штучному інтелекту є «Нейронні мережі, генетичні алгоритми і нечіткі системи». Книга присвячена питанням "інтелектуальних обчислень". Містить базові знання про генетичні алгоритми, еволюційне програмування, нечіткі системи, а також про зв'язки цих напрямків з нейронними мережами.

Мабуть найкращою і найповнішою книгою по нейронним мережам являється «Нейронні мережі – повний курс» Саймон Хайкін. Незважаючи на рік випуску – 2006, тут представлені актуальні концепції і алгоритми які стосуються нейронних мереж. У книзі розглядаються основні парадигми штучних нейронних мереж. Представлений матеріал містить суворе математичне обґрунтування всіх нейромережевих парадигм, ілюструється прикладами, описом комп'ютерних експериментів, містить безліч практичних завдань, а також велику бібліографію.

У книзі також аналізується роль нейронних мереж при вирішенні задач розпізнавання образів, управління і обробки сигналів. Структура книги дуже зручна для розробки курсів навчання нейронних мереж і інтелектуальним обчислень.

Книга буде корисна для інженерів, фахівців в області комп'ютерних наук, фізиків і фахівців в інших областях, а також для всіх тих, хто цікавиться штучними нейронними мережами.

В даній книзі детально описана проблема адаптивності нейронних мереж, а також всі інші слабкі місця нейромережевих структур. Приклади приведені в даній книзі можна легко адаптувати до сучасних мов програмування.

Однією з найповніших книжок по алгоритмам навчання нейронних мереж являється «Нейромережеві моделі і алгоритми» Дмитра Тархова. Цей підручний, який вийшов в 2014 році, містить найсучасніші підходи до нейромережевих структур. Тут розглянуто математичні моделі і алгоритми функціонування і навчання нейронних мереж, а також використовувані при їх навчанні, алгоритми побудови лінійної і нелінійної регресії, метод головних компонент, методи нелінійної оптимізації та розподілені обчислення з нейронними мережами. Викладена методологія і наведені приклади застосування нейромережевих технологій до завдань математичного моделювання, включаючи стандартні і нестандартні задачі математичної фізики. Дана методологія на порядок скорочує трудомісткість моделювання процесів і явищ в технічних системах і дозволяє інженеру-досліднику самостійно вирішувати завдання, раніше доступні тільки науковим колективам, що включає кваліфікованих фахівців з обчислювальної математики.

Ще одним посібником з хорошим математичним викладом являється «Вступ в математичну теорію систем розпізнавання і нейронних мереж». У посібнику систематично викладаються основи математичної теорії навчаючих систем розпізнавання і нейронних мереж. Поєднуючи математичну строгість викладу з змістовної мотивацією і інтерпретацією матеріалу, автори знайомлять читача з основними методами побудови навчаючих систем розпізнавання, базовими постановками задач і найважливішими типами алгоритмів. Особливу увагу приділено методам дослідження динаміки нейронних мереж як найважливішого класу навчаючих систем розпізнавання, а також досягненням Петербурзької школи математичної кібернетики В.А.Якубовича.

Питання навчання нейронних мереж,і їх застосування викладено в праці «Нейронні мережі: навчання,організація і застосування». Тут викладено математичні та алгоритмічні аспекти функціонування нейронних мереж з прямими і зворотними зв'язками; відображені питання самоорганізації, відмовостійкості та реалізації нейронних мереж на систолі стичних процесорах; велика увага приділена застосуванню і проектування нейронних мереж для вирішення різного роду завдань.

З практичного боку представлені нейронні мережі у книзі «Штучні нейронні мережі, Теорія та практика» Вадима Борисова. Книга присвячена одному з сучасних напрямків в галузі інформатики та обчислювальної техніки - нейрокомп'ютерних технологіях. Перевагою книги є те, що в ній розглянуті не тільки питання теорії штучних нейронних мереж, а й велику увагу приділено сучасним програмним оболонокам-імітатора нейронних мереж, а також рішення з їх допомогою практичних завдань розпізнавання образів, кластеризації, прогнозування, оптимізації, побудови і використання нейромережевих експертних систем. Книга містить великий довідковий матеріал.

**1.3 Аналіз існуючих проектних рішень на основі нейронної мережі.**

Вже сьогодні, нейронні мережі використовуються в багатьох областях, проте тяжко уявити застосування даного підходу в системах, де на карту поставлені людські життя або значні матеріальні ресурси. Адже перш ніж впроваджувати нейронні мережі в такого виду системи, потрібно вирішити багато питань які стосуються надійності їх роботи.

Останнім часом, робляться активні спроби об’єднання штучних нейронних мереж і експертних систем. В такій системі штучна мережа може реагувати на прості випадки, а все інше передається на розгляд до експертної системи. Нейро-мережеві прикладні програми, які розробляються компаніями, дозволяють працювати з різними видами нейронних мереж і з різними способами їх навчання. Вони можуть бути як спеціалізованими (наприклад, для передбачення курсу акцій), так і досить універсальними.

Області застосування нейронних мереж досить різноманітні - це розпізнавання тексту й мови, семантичний пошук, експертні системи і системи підтримки прийняття рішень, передбачення курсів акцій, системи безпеки, аналіз текстів.

1)Техніка і телекомунікації.

Фірма Accurate Automation Corp, Chattanooga розробила літальний апарат LoFLYTE. Цей гіперзвуковий розвідник використовував нейронні мережі. Задачею мережі було запам’ятовування дій пілота, записуючи усі прийоми в статистичну навчальну вибірку. Даний літальний апарат був розрахований на велику швидкість і часами швидкості реакції пілота було недостатньо для виходу з екстреної ситуації, проте навчена нейрона мережа могла запросто управляти літальним апаратом і адекватно та швидко реагувати на екстремальну ситуацію.

Одна з найважливіших задач в області телекомунікацій, яка полягає в знаходженні оптимального шляху пересилання трафіку між вузлами, може бути успішно вирішена за допомогою нейронних мереж. В даному випадку необхідно брати до уваги те, що, по-перше, запропоноване рішення має враховувати поточний стан мережі, якість зв'язку і наявність збійних ділянок, а по-друге, пошук оптимального рішення повинен здійснюватися в реальному часі. Нейронні мережі добре підходять для вирішення завдань такого роду. Крім управління маршрутизацією потоків, нейронні мережі можуть використовуватися і при проектуванні нових телекомунікаційних мереж, дозволяючи отримувати вельми ефективні рішення.

2)Інформаційні технології.

Визначення тематики текстових повідомлень - ще один приклад успішного використання штучних нейронних мереж. Так, сервер новин Convectis (продукт компанії Aptex Software, Inc.) був обраний в 1997 році компанією PointCast, Inc., що була лідером персоналізованої доставки новин в Інтернеті, для автоматичної рубрикації повідомлень за категоріями. Визначаючи значення ключових слів з контексту, сервер Convectis був здатний в реальному часі розпізнавати тематику і автоматично рубрикувати величезні потоки текстових повідомлень, переданих за такими інформаційних мереж, як Reuters, NBC і CBS.

Нейромережевий продукт SelectCast від Aptex Software, Inc. дозволяв визначати область інтересів користувачів Інтернету і пропонував їм рекламу відповідної тематики. Влітку 1997 року компанія Excite, Inc. ліцензувала цю розробку для використання на своїх пошукових серверах. Після установки на серверах Excite і Infoseek нейромережевою рекламою було охоплено близько третини всіх користувачів мережі на той момент. Проведені дослідження встановили, що відгук на таку тематичну рекламу була в середньому в два рази вище, ніж на звичайну, а для окремих її видів ефективність збільшувалася до п'яти разів.

Розпізнавання мови є вельми популярним застосуванням нейронних мереж, реалізованих в ряді програмних продуктів. У компанії «НейроПроект» кілька років тому була створена демонстраційна система для мовного управління вбудованим в Windows калькулятором. Система дозволяла без попереднього навчання впевнено розпізнавати кожне з 36 слів, сказаних в мікрофон будь-якою людиною. Для класифікації використовувалася ієрархічна нейронна мережа, що складається з двох каскадів: перший здійснював зразкове розпізнавання слова, відносячи його до одного з шести класів, а другий точно класифікував слово всередині кожного з класів. У навчанні цієї нейронної мережі брали участь 19 дикторів.

3)Економіка і фінанси.

Нейронні мережі активно застосовуються на фінансових ринках. Наприклад, американський Citibank використовує нейромережеві передбачення з 1990 року, і вже через два роки після їх впровадження, за свідченням журналу The Economist, автоматичний дилинг показував прибутковість 25% річних. Chemical Bank застосовує нейромережевому систему фірми Neural Data для попередньої обробки транзакцій на валютних біржах ряду країн, відстежуючи підозрілі операції. Автоматизовані системи ведення портфелів з використанням нейромереж є на озброєнні і у Deere & Co LBS Capital, причому експертна система об'єднується приблизно з 900 нейронними мережами.

У вересні 1992 року компанія HNC, яка до цього займалася виробництвом нейрокомп'ютерів, випустила програмний продукт Falcon, що дозволяє виявляти і запобігати в реальному часі підозрілі операції по краденим кредитними та дебетними картками. Штучні нейронні мережі навчалися типовому поводженню клієнтів і могли виявляти різкі зміна характеру покупок, що сигналізує про можливу крадіжку. Щорічний збиток великих банків від подібних злочинів вимірювався десятками мільйонів доларів, але завдяки впровадженню Falcon в 1994 році вперше за всю історію пластикових карт ці втрати пішли на спад. Аналогічна система була розроблена фірмою ITC для моніторингу операцій з кредитними картами Visa.

Кілька років тому великий канадський банк CIBC для управління ризиками та ідентифікації зловмисників встановив програму Knowledge Seeker фірми Angoss. З її допомогою фахівці банку вирішили з'ясувати, хто з їхніх клієнтів в майбутньому буде з високою часткою ймовірності затримувати виплати по заставних. Спочатку передбачалося, що в першу чергу ними виявляться ті, хто і перш затримував свої виплати на кілька днів. Однак дослідження показали, що в майбутньому проблеми з платежами виникнуть у тих клієнтів банку, які на тлі регулярних виплат іноді нібито забували заплатити. Як з'ясувалося, подібна «забудькуватість» була пов'язана з серйозними фінансовими труднощами.

4)Реклама й маркетинг.

Компанія Neural Innovation Ltd використовувала при роботі з маркетинговими компаніями стратегію прямої розсилки. Спочатку вона здійснювала розсилку всього 25% від загального числа пропозицій і збирала інформацію про відгуки і реакціях споживачів. Потім ці дані надходили на вхід нейронної мережі, за допомогою якої здійснювався пошук оптимального сегмента споживчого ринку для кожного товару. Після цього інші 75% пропозицій розсилалися вже з урахуванням виявлених закономірностей в зазначений сегмент, і ефективність другої розсилки значно зростала в порівнянні з початковою.

При веденні бізнесу в умовах конкуренції компаніям необхідно підтримувати постійний контакт з споживачами, забезпечуючи зворотний зв'язок. Для цього деякі компанії проводять опитування споживачів, що дозволяють з'ясувати, які чинники є вирішальними при купівлі даного товару або послуги. Аналіз результатів подібного опитування - непросте завдання, оскільки необхідно досліджувати велику кількість пов'язаних між собою параметрів і виявити фактори, що найбільший вплив на попит. Існуючі нейромережеві методи дозволяють з'ясувати це і прогнозувати поведінку споживачів при зміні маркетингової політики, а значить, знаходити оптимальні стратегії роботи компанії.

Одне велике англійське видавництво, що випускає газети, придбало у фірми Neural Innovation Ltd систему планування цін та витрат, побудовану на використанні нейронної мережі і генетичних алгоритмів. На основі накопичених даних ця система дозволяла виявляти складні залежності між витратами на рекламу, об'ємом продажів, ціною газети, цінами конкурентів, днем ​​тижня, пори року і рядом інших факторів. В результаті видавництво могло підбирати оптимальну стратегію з точки зору максимізації обсягу продажів або прибутку.

4)Охорона здоров’я.

Свого часу в США була введена в дію система виявлення шахрайств в області охорони здоров'я. Було підраховано, що втрати бюджету від такого роду фальсифікацій становлять близько 730 млн. дол. на рік. Створення спеціалізованої нейромережевої системи зайняло у фірми ITC більше року і обійшлося всього в 2,5 млн. дол. Тестування нової системи показало, що нейронна мережа дозволяє виявляти 38% випадків шахрайства, тоді як використовувалася до неї експертна система давала тільки 14%. Для настройки нейронної системи були застосовані також методи нечіткої логіки та генетичної оптимізації.

У медичній діагностиці нейронні мережі нерідко використовуються разом з експертними системами. Компанією «Нейро Проект» була створена система об'єктивної діагностики слуху у немовлят. Загальноприйнята методика діагностики полягає в тому, що в процесі обстеження реєструються відгуки мозку у відповідь на звуковий подразник, які проявляються у вигляді сплесків на електроенцефалограмі. Для діагностики слуху дитини досвідченому експерту-аудиолога необхідно провести близько 2 тис. тестів, нейронна мережа здатна з тією ж достовірністю визначити рівень слуху вже по 200 спостереженнями протягом всього декількох хвилин, причому без участі фахівця.

**1.4 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 засобами налаштування 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.

**1.5 Навчання з учителем.**

Навчання з учителем є однією з основних парадигм навчання нейронних мереж. На рисунку 1.5.1 показана блокова діаграма, яка ілюструє цю форму навчання.

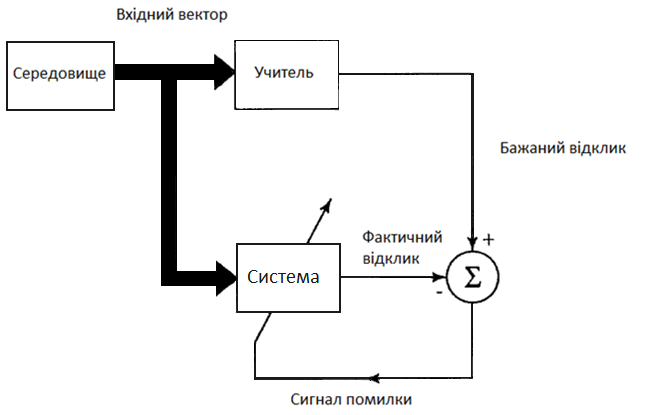


Рис.1.5.1 Блокова діаграма навчання з вчителем.

Концептуально участь учителя можна розглядати як наявність знань про оточуюче середовище, представлене у вигляді пар вхід-вихід. При цьому, саме середовище невідоме нейронній мережі яка навчається. Тепер припустимо, 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 по-кроково 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 є 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 багатовимірну 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 про градієнт 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 здатне вирішувати такі задачі як класифікація і апроксимація функцій.

**1.6 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.

В навчанні з підкріпленням, формування відображення вхідних сигналів в вихідні виконується в процесі взаємодії з зовнішнім середовищем з ціллю мінімізації скалярного індексу продуктивності. На рисунку 1.6.1 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. Ціллю навчання є мінімізація функції переходу, яка являє собою математичне очікування кумулятивної вартості дій протягом декількох кроків а, не просто теперішньої вартості.

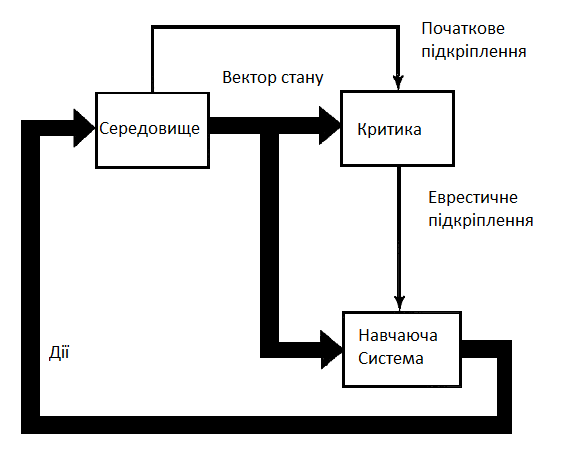


Рис.1.6.1 Блокова діаграма навчання з підкріпленням.

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 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 даних, і таким же чином автоматично створювати нові класи.

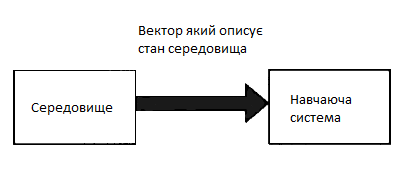


Рис.1.6.2 Блокова діаграма навчання без вчителя

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 «переможець отримує все». Як було показано при такій стратегії нейрон з найбільшим сумарним вхідним сигналом «перемагає» і переходить в активний стан.

**1.7 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.

**1.8 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 даному розділі буде розглянуто тільки ті задачі які стосуються обробки статистичних даних.

**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.1.8.1 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 (Рис. 1.8.1).

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.

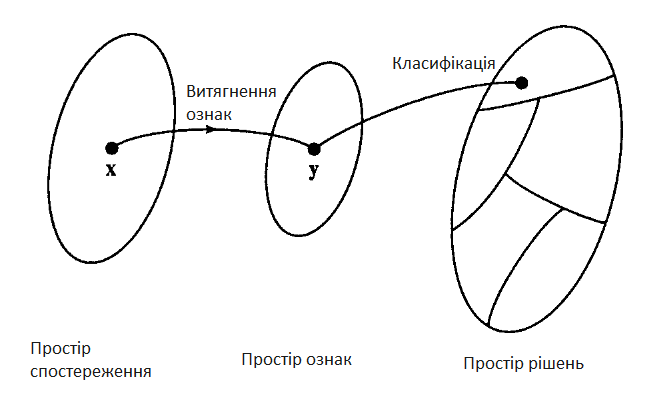


Рис.1.8.2 Ілюстрація класичного підходу до розпізнавання образів

Саме такий тип системи буде використовуватись в результатах даної роботи.

**Апроксимація функцій.**

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

(1.8.1)

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

(1.8.2)

До структури нейронної мережі, яка апроксимує невідому функцію, пред’являється наступна умова:функція F(x), яка описує відображення вхідного сигналу в вихідний, повинна бути достатньо близько до функції f(x) 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.

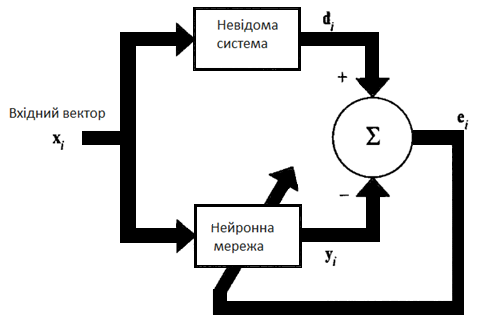


Рис.1.8.3 Блокова діаграма вирішення задачі ідентифікації системи

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).

**1.9 Пам'ять інтелектуальних систем.**

В контексті нейробіології, під пам’ятю розуміють відносно тривала по часі деформація структури нейронів, яка була викликана на організм зовнішнім середовищем. 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 1.9.1 показана мережа, яку можна розглядатись як модельний компонент нервової системи. Всі нейрони вхідного шару з’єднані зі всіма нейронами вихідного. В реальних системах зв’язки між нейронами являються тяжкими і надлишковими. В моделі на рисунку рисунку 1.9.1 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.

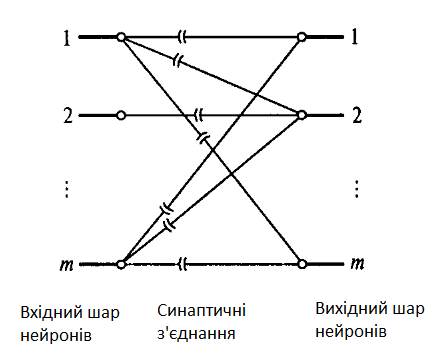


Рис.1.9.1 Компонент асоціативної пам’яті нервової системи.

Аналогічна ситуація для штучної нейронної мережі показаної на рисунку 1.9.2. В даному випадку, вузли джерел 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.

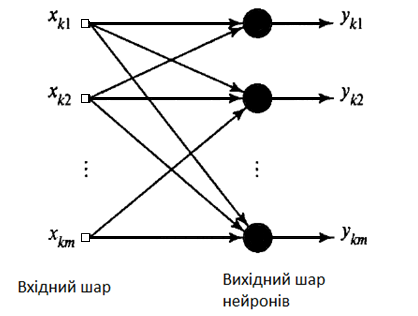


Рис.1.9.2 Модель асоціативної пам’яті з штучними нейронами.

**1.10 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.1.10.1 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 засобах 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.

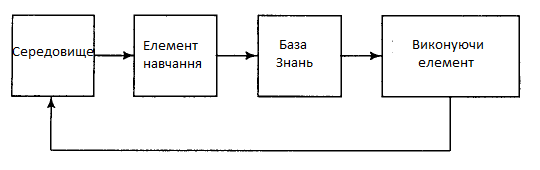


Рис.1.10.2 Проста модель машинного навчання.

1. Навчання. В найпростішій моделі машинного навчання(Рис.1.10.2) 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 нейроно-біологічному опису 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 мова 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 робото-спроможності. Для реалізації даного підходу 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.

**1.11 Висновки**

Отже, 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.

Як результат, можна створити структуру системи класифікації яка би відповідала теперішнім уявленням про використання нейронних мереж. На рисунку 1.6.1 показана структура такої системи.

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 1.7.1 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.

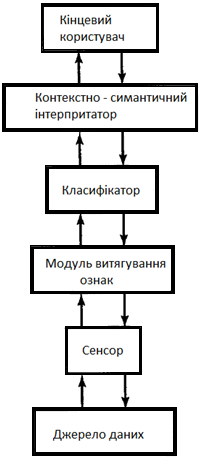


Рис.1.11.1 Функціональна архітектура інтелектуальної системи.

Наведені приклади використання нейромережевих систем показують, що технології нейронних мереж застосовні практично в будь-якій області, а в таких завданнях, як розпізнавання образів і прогнозування котирувань акцій, вони стали вже звичним і широко використовуваним інструментом. Повсюдне проникнення нейронних технологій в інші області - тільки питання часу. Звичайно, впровадження нових наукоємних технологій - процес складний, однак практика показує, що інвестиції не тільки окупаються і приносять вигоду, а й дають тим, хто їх використовує, відчутні переваги.

**2. КОМПОНЕНТИ НЕЙРОННОЇ МЕРЕЖІ.**

**2.1 Модель нейрона**

Штучний нейрон це базовий елементом обробки інформації в штучній нейронній мережі. В моделі цього елемента можна виділити такі складові елементи(Рис.2.1.1):

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.

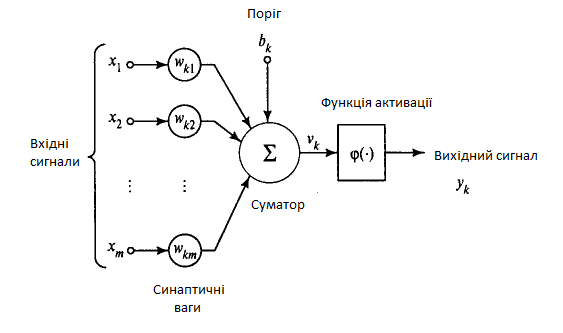


Рис.2.1.1 Модель нейрона

Математично представити штучний нейрон можна за допомогою рівнянь:

(2.1.1)

(2.1.2)

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

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

В залежності від мінусового або додатного знаку порогу *b*k потенціал активації нейрона змінюється як на рисунку 2.1.2:

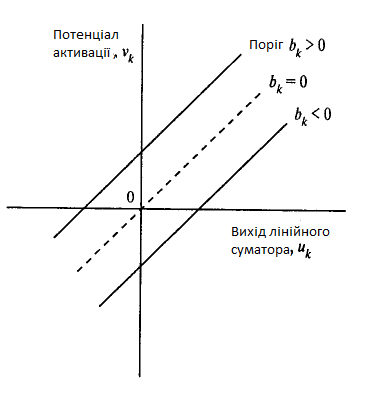


Рис.2.1.2 Афінне перетворення викликане наявністю порогу.

Як вже було сказано, поріг часто відображається як статичний вхід зі своєю вагою (Рис. 2.1.3), саме таку реалізацію нейрона було використано в даній роботі.

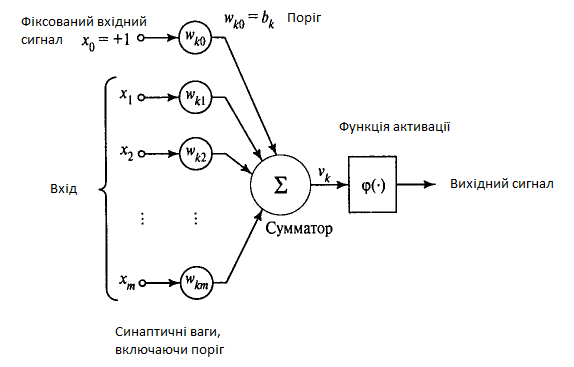


Рис.2.1.3 Модель нейрона з статичним входом

**Функція активації.**

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

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

(2.1.3)

Де *v –* потенціал активації нейрона.

(2.1.4)

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

2.Кусково лінійна функція.

(2.1.5)

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

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

(2.1.6)

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

Оскільки результатом спроектованої нейронної мережі є певний клас, якому належить вхідний вектор, хорошим кандидатом на функцію активації являється порогова функція. Проте, якщо вхідний образ може належати одночасно декільком образам, то виходу 1 або 0 даної функції буде не достатньо. Саме тому було вибрано сигмоїдальну функцію. Таким чином результатом являється вихід(або виходи) які мають найбільше значення.

Основна складність побудови мережі, на основі сигмоїдальної функції активації, заключається в виборі параметру нахилу. Адже для швидкої збіжності, індуктивне поле нейрона повинно знаходитись близько до середніх значень функцій. Оскільки за мету даної системи, ставиться саме адаптивна нейронна мережа, то міняти коефіцієнт нахилу в реальному часі стає неможливим. Проте, якщо провести нормалізацію вхідних даних, середні значення індуктивних полів нейрона будуть в діапазоні від 0 до 1. Якщо глянути на рисунок 2.1.4 стає видно що параметр нахилу рівний одиниці задовільняє умову швидкої збіжності. Таким чином параметр нахилу можна прийняти за одиницю.

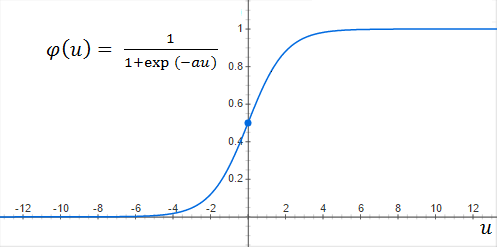


Рис.2.1.4. Сигмоїдальна функція, де параметр нахилу рівний одиниці.

**2.2. Архітектура мереж**

Архітектура мережі завжди залежить від вибраного алгоритму навчання. Зазвичай виділяють три фундаментальних класи нейро-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 2.2.1 зображена дана архітектура яка називається одношаровою.

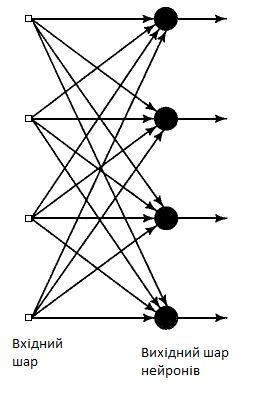


Рис.2.2.1 Одношарова мережа прямого розповсюдження.

**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 шарів 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.2.2.2)

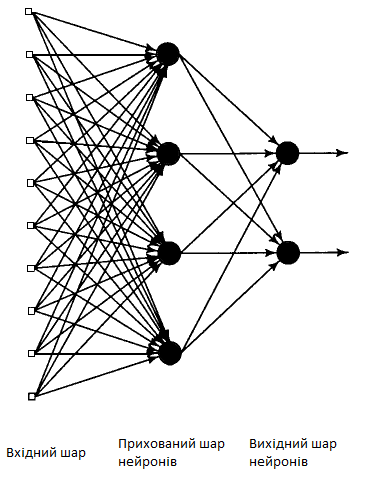


Рис.2.2.2 Мережа прямого розповсюдження з одним прихованим шаром.

**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.2.2.3) Такі мережі використовують так звані елементи затримки. Вони дозволяють виконувати нелінійну поведінку мережі. Наявність таких зворотних зв'язків має величезний вплив на навчання нейронних мереж і на їхню швидкість.

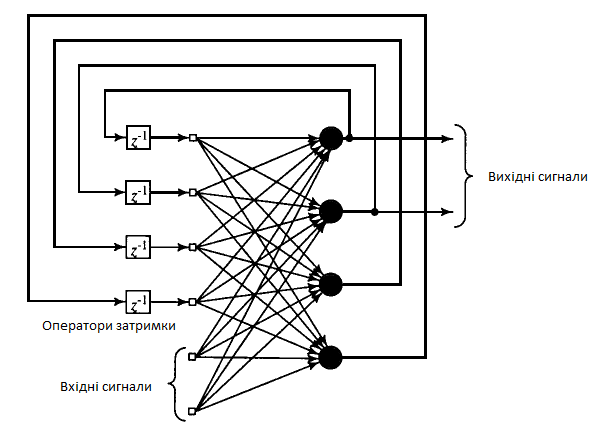


Рис.2.2.3 Рекурентна мережа

Для спроектованої системи було вибрано багатошарову мережу прямого розповсюдження. Кількість прихованих шарів та кількість нейронів в них зазвичай вибирається експериментально. При розгляді графіків швидкодії, було вибрано архітектуру в якій приховані шари містять однакову кількість нейронів, яких є в два рази більше чим нейронів у вхідному шарі мережі.

**2.3 Багатошаровий персептрон**

Вибрана архітектура для спроектованої системи являється так званим "Багатошаровим перспетроном". 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).

(2.3.1)

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 з'єднань 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.

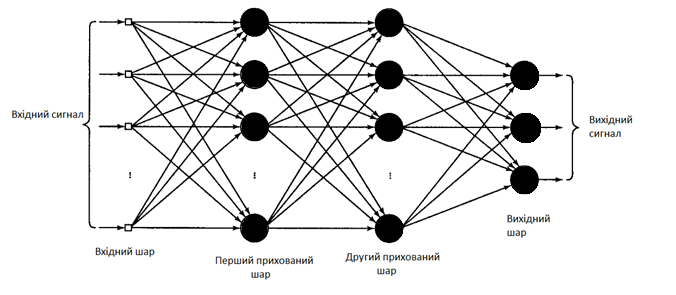


Рис.2.3.1 Архітектурний граф багатошарового перспетрона з двома прихованими шарами

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

На рисунку 2.3.2 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 мережу.

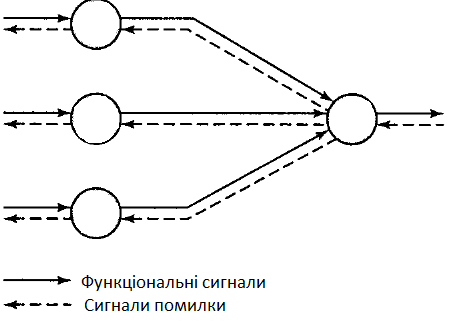


Рис.2.3.2 Напраямок двох основних потоків сигналів для багатошарового перспетрона.

**2.4 Навчання на основі корекції помилок.**

Обраним алгоритмом навчання нейронної мережі являється "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)*:

(2.4.1)

Сигнал помилки ініціалізує механізм управління, ціль якого заключається 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:

(2.4.2)

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:

(2.4.3)

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 помилки на вхідний сигнал який його викликав.

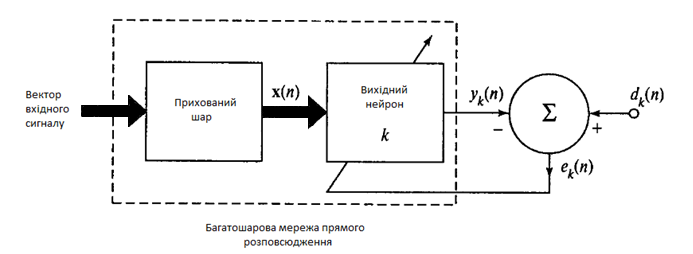


Рис.2.4.1 Блокова діаграма нейронної мережі

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.2.4.1) 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.*

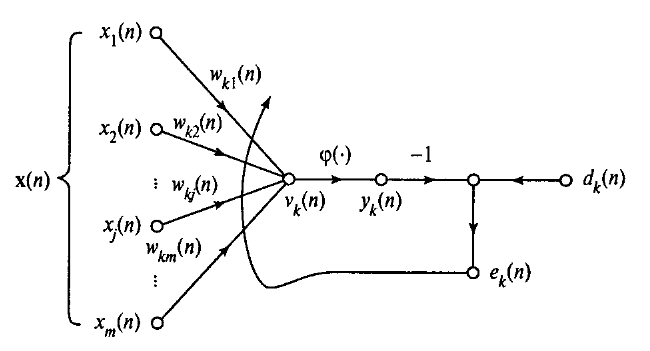


Рис.2.4.2 Граф передачі на основі корекції помилок.

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

(2.4.4)

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

(2.4.5)

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

На рисунку 2.4.2 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 збіжності 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.

**2.5 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.

В спроектованій системі навчальна вибірка зберігається в реляційній базі даних, до якої має доступ тільки адміністратор системи. Після будь яких змін в базі даних навчальної вибірки відбувається адаптація нейронної мережі до нових вхідних даних. Сам навчальний приклад являється стандартною парою "вхід - очікуваний вихід".

**2.6 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 використовувати 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.

**2.7 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.

Описаний алгоритм адаптації нейронної мережі 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.

**3.Системний аналіз проектного рішення**

**3.1 Призначення системи та загальні вимоги**

Призначення системи, реалізованої в даній роботі – в швидкій обробці статистичних даних, які представляються у вигляді щоденного збору стану користувача. Основною метою нейронної мережі, на основі якої побудована система, це визначення можливого діагнозу і групи ризику, базуючись на даних зібраних на стороні клієнта. Нейронна мережа повинна бути спроектована як клієнтська бібліотека, яку можна було б використовувати для любих задач «навчання з вчителем».

Основні вимоги:

* Швидкодія системи.
* Адаптивне довчання нейронної мережі.
* Можливість працювати з великою кількістю навчальних даних.

Доступитись до нейронної мережі можна через програмний інтерфейс, який реалізований у вигляді веб – додатка (ASP.NET WEB API 2). При публікації додатка на сервер, відбувається навчання нейронної мережі, на навчальній вибірці, яка зберігається в базі даних MS SQL Server. В базі даних також повинна зберігатись інформація про ваги нейронів.

Користувач, який бажає скористатись системою, надсилає запит через клієнтський додаток, і отримує відповідь у вигляді прогнозованого стану, та групи ризику, також разом з відповіддю відправляються поради та нормальні показники стану здоров’я.

Нейронна мережа повинна адаптуватись до зміни зовнішнього середовища. Адже з плином часу, статистичні дані зміщуються в координатному просторі. Просте доповнення бази даних навчальних прикладів можливе, проте завжди є ризик «перенавчання» нейронної мережі.

Спілкування системи з користувачем зображено на рисунку 3.1.1 у вигляді UML діаграми послідовності. Як ми бачимо, при запиті, нейронна мережа ініціалізується заново кожен раз, проте ваги відповідних нейронів збережені.

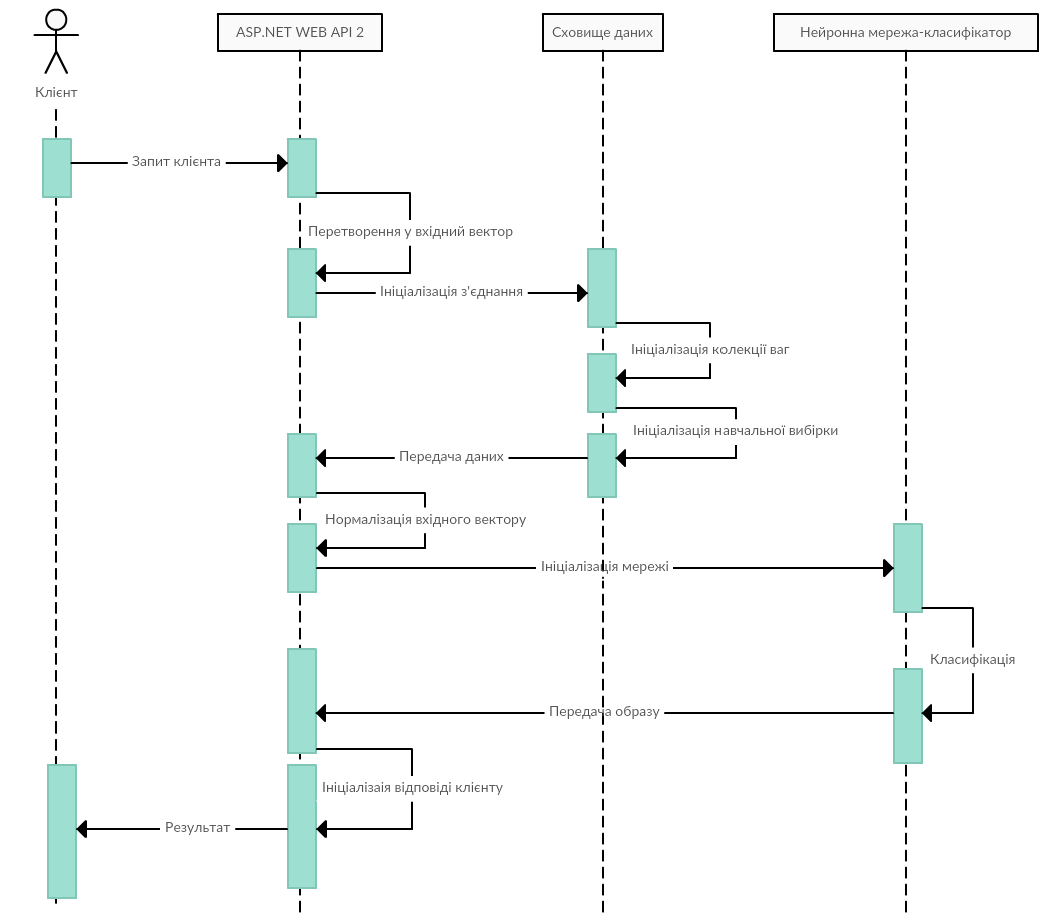


Рис.3.1.1 UML Діаграма послідовності.

**3.2 Структура системи.**

Як вже було зазначено, система включає в себе веб-додаток ASP.NET WEB API 2 та компоненти нейронної мережі. Нейронна мережа створена як клієнтська бібліотека з модулями для конфігурації та управління(Рис.3.2.1).

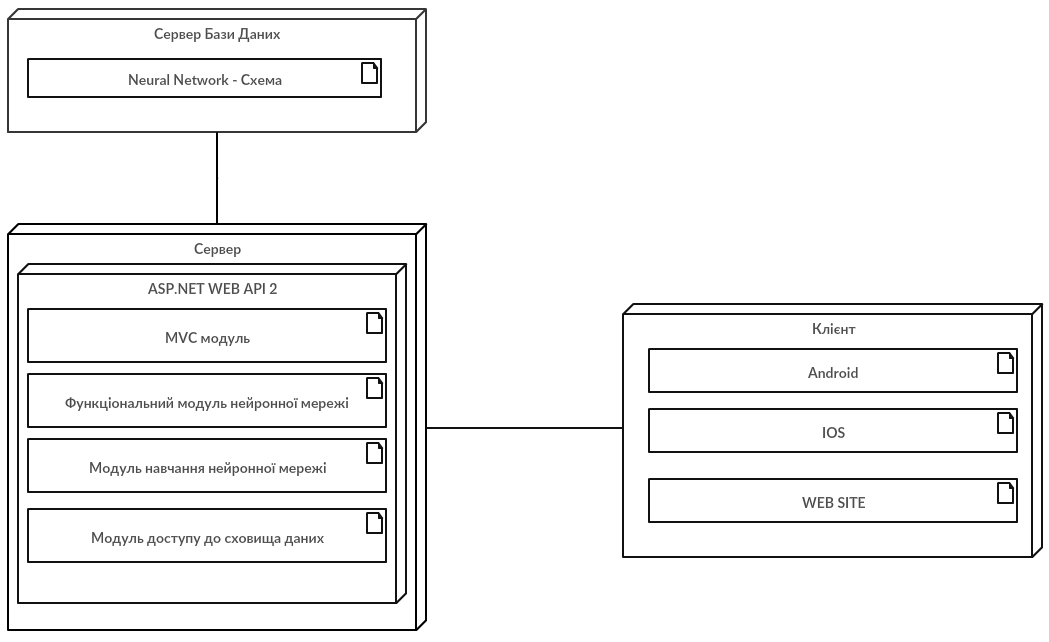


Рис.3.2.1. UML Діаграма розгортання.

Також створено окремий модуль для з’єднання з базою даних. Загальний модуль нейронної мережі зображений на рисунку 3.2.2.

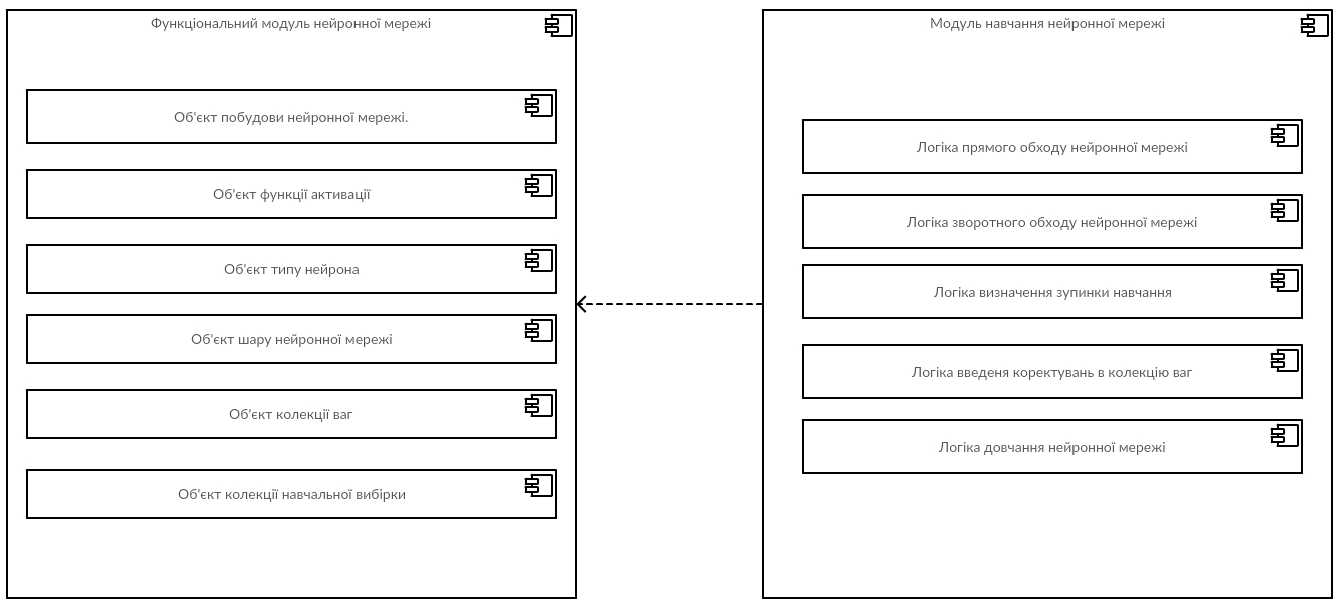


Рис.3.2.2 Компонентна діаграма модуля нейронної мережі.

Для поставленої задачі класифікації можливого діагнозу, було використано повнозв’язну нейронну мережу 21-42-42-42-n, де n це кількість хворіб занесених в навчальну вибірку(Рис.3.2.3). Також в системі використовуються допоміжні нейронні мережі для визначення групи ризику. Допоміжні нейронні мережі будуться автоматично в залежності від кількості діагнозів в навчальній вибірці і мають схему 21-15-3(На виході отримується один з 3 класів: Low, Middle, Hight). Оскільки функцією активації було вибрано сигмоїдальну функцію, то вибирається той вихід який має найбільшу величину, також можна вибирати декілька входів по бажанню користувача. Якщо ж користувач бажає проігнорувати один з класів, то вихід можна відключити на час даного запиту.

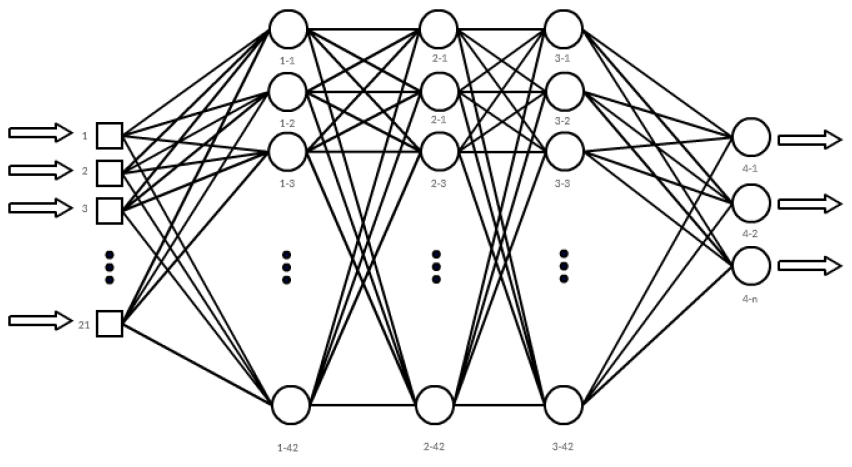


Рис.3.2.3 Схема основної нейронної мережі системи.

Кожен вихід основної нейронної мережі під’єднаний до своєї версії допоміжної мережі, також в допоміжні нейронні мережі передаються всі вхідні параметри основної мережі.

**3.3 Обгрунтування засобів реалізації**

**3.3.1 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.

**3.3.2 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).

**3.3.3 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.

**3.3.4 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.

**3.3.5 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.

**3.4 Висновки**

Отже, спроектована система дає змогу в реальному часі проводити до навчання нейронної мережі, таким чином роблячи її адаптивним фільтром. Це дає змогу застосовувати дану методику в місцях де середовище нейронної мережі є змінюваним а прикладні технології .NET дозволили зберегти швидкодію системи, хоча це і являється найбільшим мінусом нейронних мереж.

Відкрита платформа .NETдозволяє 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.РЕЗУЛЬТАТИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ**

**4.1 База даних.**

На рисунку 4.1.1 зображено схему бази даних яка використовується у даній системі.

Основною табличкою являється табличка LearningSamples. Вона містить в собі навчальну вибірку для нейронної мережі. Тут зберігаються статистичні дані пацієнтів, їхнього стану та групи ризику. Вхідний вектор включає в себе такі показники як:ріст, вагу, пройдену відстань, рівень цукру тощо.

Іншою важливою табличкою являється таблиця Weights. Вона використовується для збереження поточного стану ваг нейронної мережі. Вона в основному містить індекси шару нейронів і власне індекси позицій нейронів.

Така схема дозволяє зберігати ваги в пам’яті в C# колекції Dictionary. Таким чином отримується момементальний доступ до ваги під час обрахунку результатів та навчання.

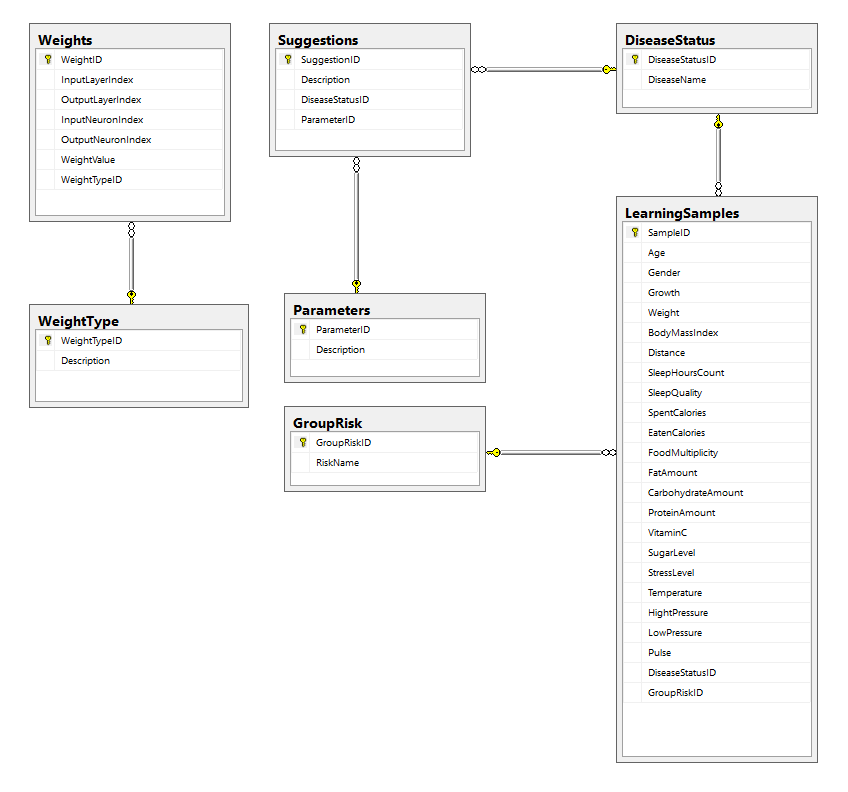


Рис.4.1.1 Схема бази даних

Також база даних містить таблиці для зберігання важливих параметрів які стосуються хвороби та порад користувачеві.

Оскільки швидкодія для нейронної мережі завжди було слабким місцем, було вирішено не використовувати ORM яка би значно сповільнила з’єднання з базою даних. Для доступу було використано технологію ADO.NET.

**4.2 Програмні компоненти системи**

Об’єкт побудови нейронної мережі створений у вигляді партерна Builder(Рис.4.2.1). Це дозволяє користувачу бібліотеки легко конфігурувати нейронну мережу згідно поставленої задачі.



Рис.4.2.1 Конфігурація нейронної мережі

Також бібліотека містить основні інтерфейси та базові імплементації основних компонентів нейронної мережі (Рис. 4.2.2), такі як:нейрон, шар нейронів, об’єкт функції активації тощо.

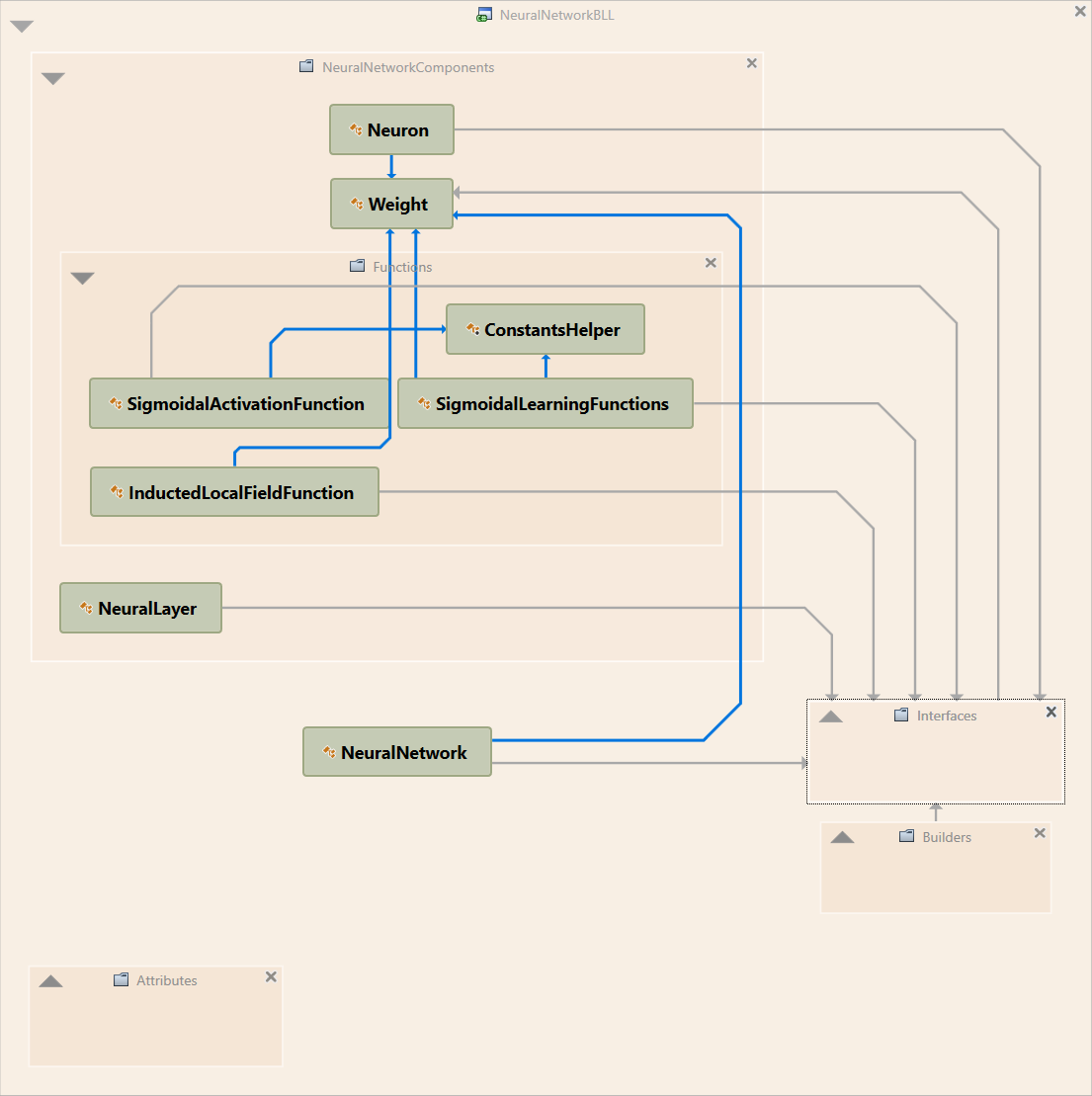


Рис.4.2.2 Основні компоненти бібліотеки нейронної мережі.

Модуль доступу до сховища даних включає в себе SQL запити та логіку утворення об’єктів які необхідні для навчання нейронної мережі. Загальна структура цієї частини системи зображена на рис.4.2.3.

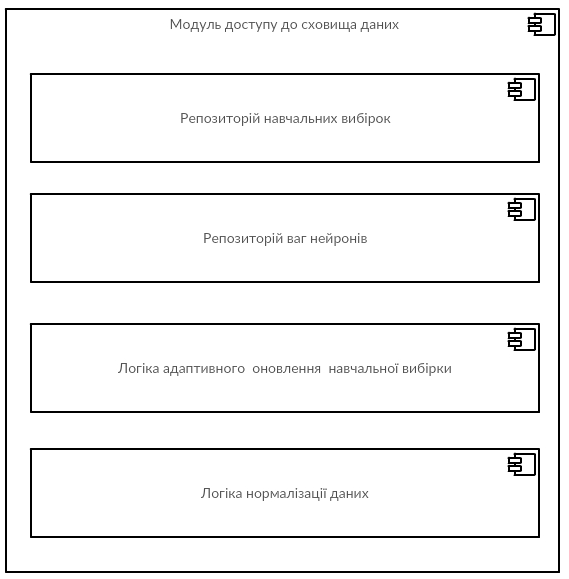


Рис.4.2.3 Модуль доступу до сховища даних.

Програмна структура логіки роботи з базою даних зображена на рисунку 4.2.4.

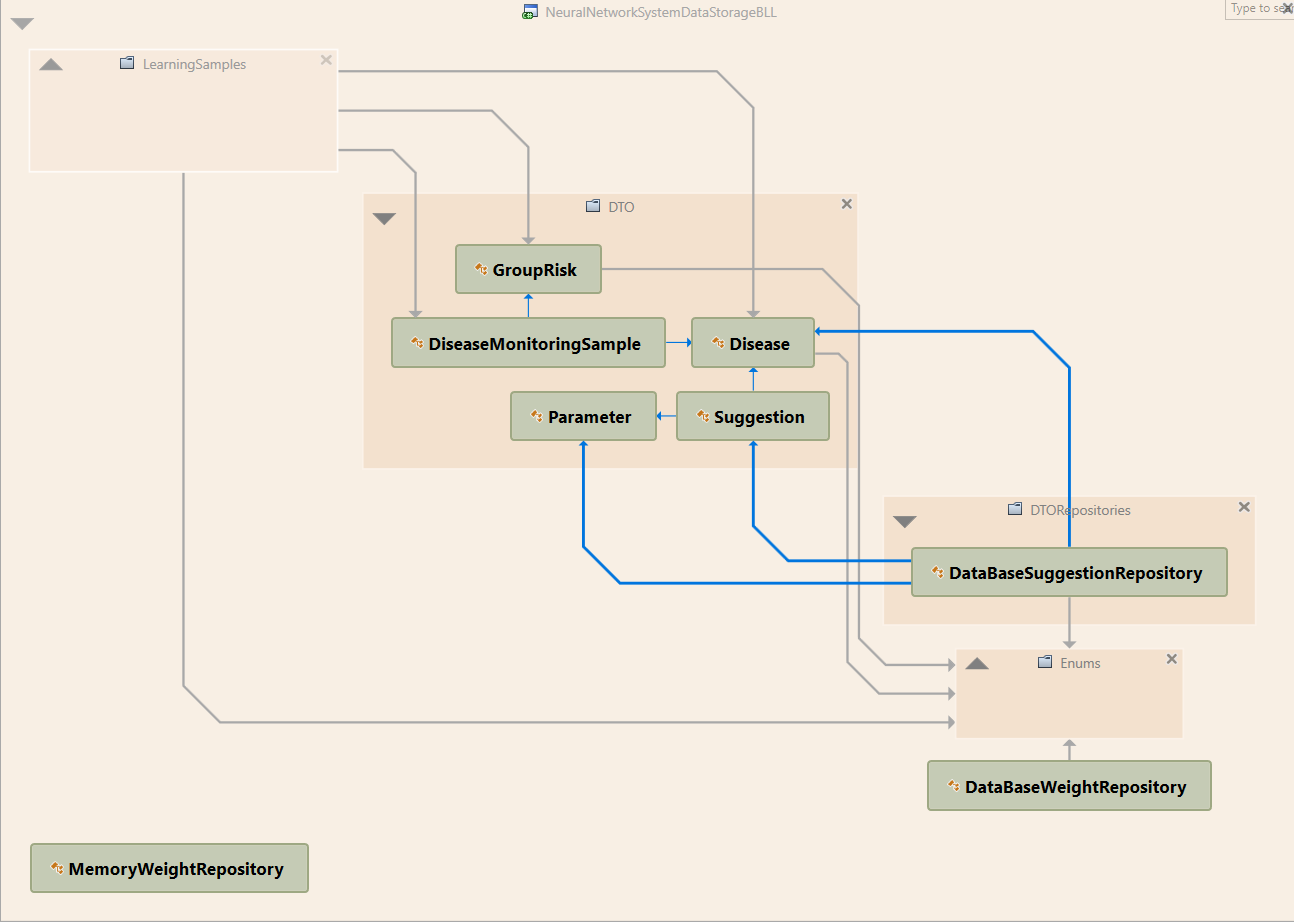


Рис.4.2.4 Складові частини системи для доступу до бази даних.

**4.3 Результати виконання.**

Звертаючись до веб – додатку системи, користувач в змозі отримати результат у вигляді класифікованого стану та групи ризику. Також в разі віднесення стану до якогось діагнозу користувачеві надсилаються поради.

Приклад запиту до системи за допомогою програми Postman зображено на рисунку 4.3.2. Як видно з рисунку, система отримує HTTP GET запит та формує з нього вхідний вектор. Після чого система нормалізує вхідні дані(Рис.4.3.1).

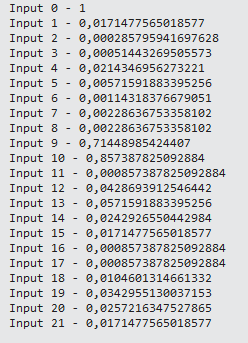


Рис.4.3.1 Нормалізовані дані.

Як бачимо, всі дані знаходяться в межах від 0 до 1. Що дає можливість швидкої збіжності при навчанні мережі.

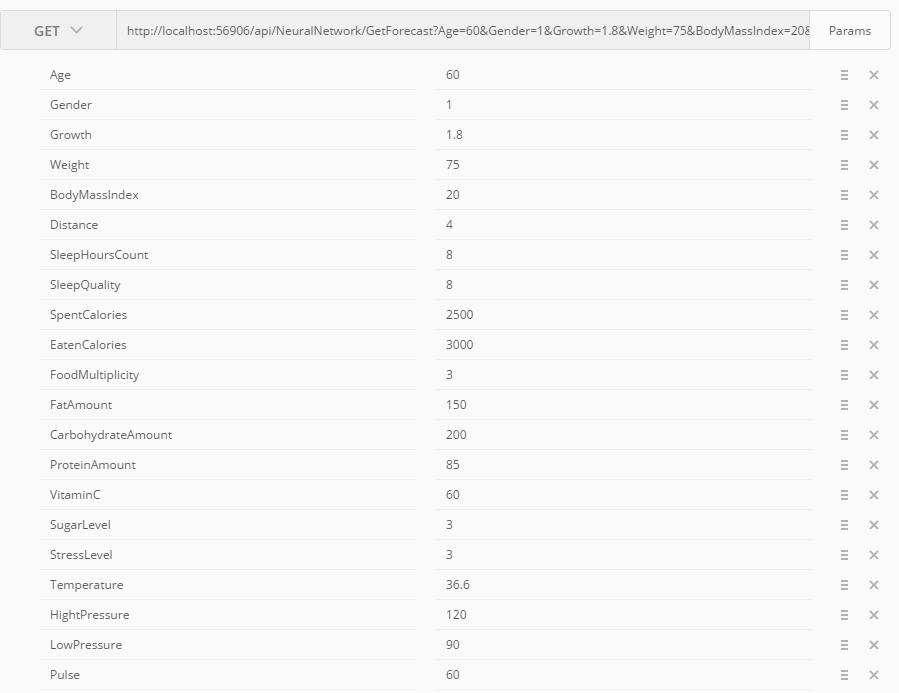


Рис.4.3.2 Приклад запиту до системи.

В результаті з усіх виходів нейронної мережі вибирається вихід з максимальним значенням і формується відповідь користувачеві у форматі JSON(Рис.4.3.2). Відповідь включає в себе ідентифікатор хвороби, її назву, ідентифікатор групи ризику, та поради щодо всіх важливих компонентів статистичних даних.

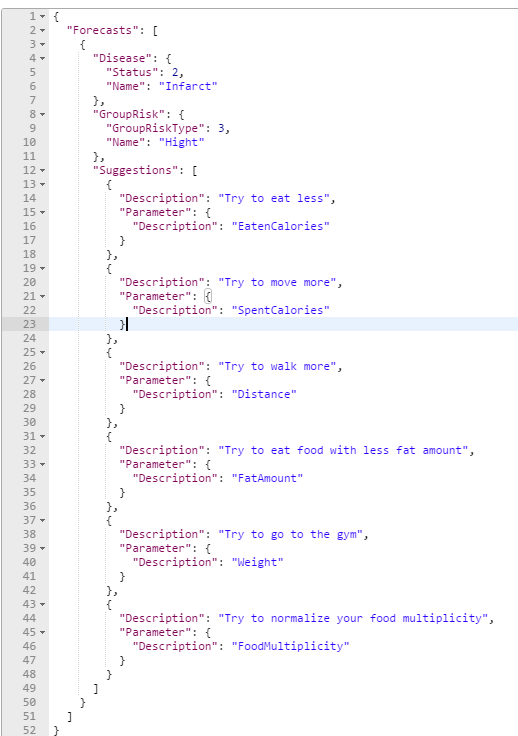


Рис.4.3.3 Відповідь користувачеві.

Також система має можливість автоматичного оновлення середовища. Системі можна відправити новий навчальний приклад. Даний запит(Рис.4.3.3) автоматично запустить процес довчання нейронної мережі.

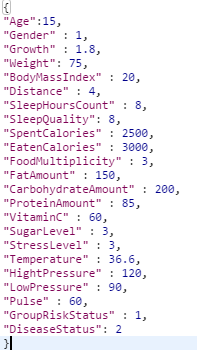


Рис.4.3.3 Тіло запиту на оновлення статистичної вибірки.

Перед тим як включити новий навчальний приклад, система вибирає один приклад для видалення з навчальної вибірки. Це зроблено для того, аби запобігти перенавчання системи. Алгоритм наступний: вибирається окіл найближчих векторів, з яких для видалення обирається найдальший. Для розрахунку околу та найдальшого вектору використовується Евклідова відстань.

Після цього, мережі достатньо однієї епохи навчання для того аби адаптувати свої ваги під змінене середовище.

Процес навчання сам по собі довготривалий і напряму залежить від навчальної вибірки. Критерієм зупинки навчання було вибрано такий алгоритм:коли процент помилок на навчальній вибірці стає меншим якогось достатньо малого числа – навчання припиняється.

Як видно з наступного графіка, основній нейронній мережі для визначення діагнозу достатньо менше десяти епох для досягнення мінімального значення помилок. Проте за рахунок великої кількості нейронів, ці епохи займають декілька десяток секунд кожна. Початкова кількість помилок є випадоковою, так як початкові ваги випадковим чином виставляються від -0.5 до + 0.5.

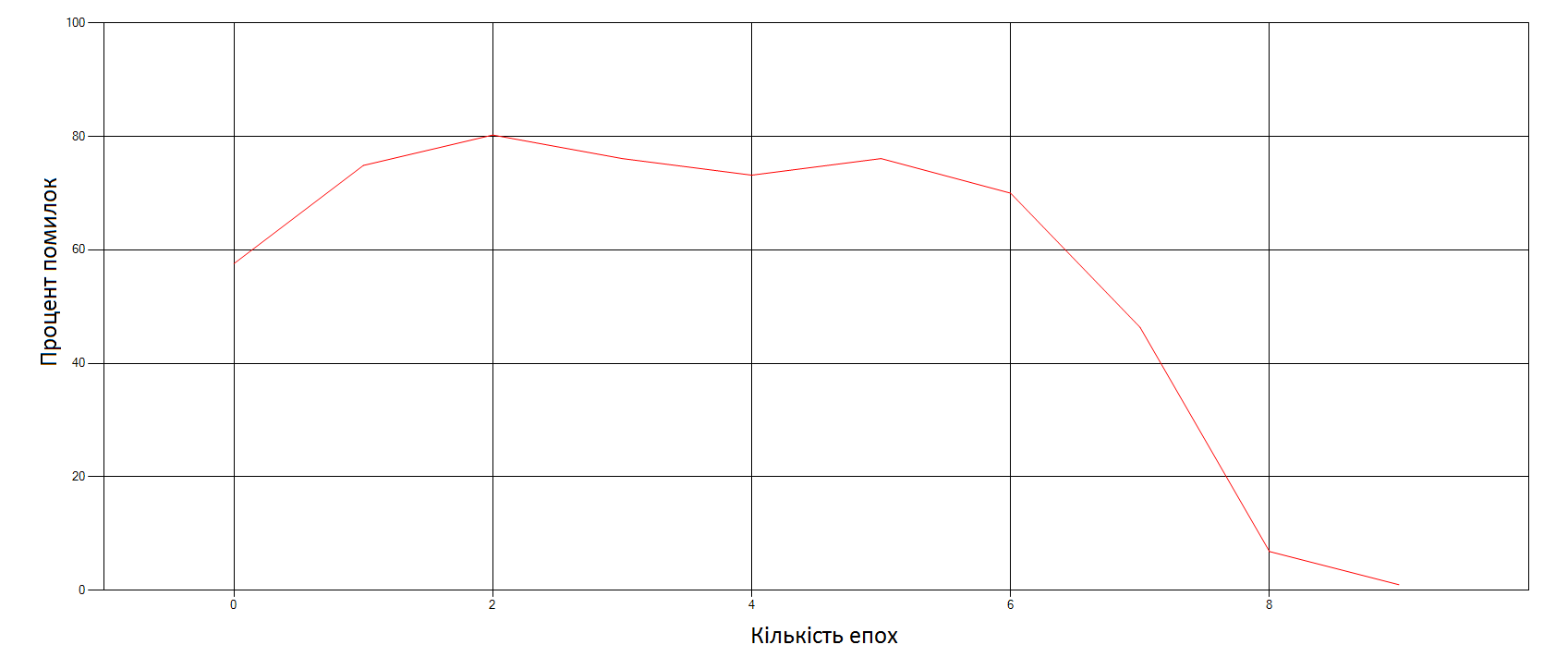


Рис.4.3.4 Процес навчання основної нейронної мережі.

Щодо додаткової нейронної мережі, яка використовується для визначення групи ризику, ми бачимо зовсім іншу картину. За рахунок меншої кількості нейронів, епоха проходить менше чим за секунду, проте за рахунок не однорідності і тісноті образів на просторі, для даної нейронної мережі потрібно значно більше епох для досягнення мінімуму помилок.

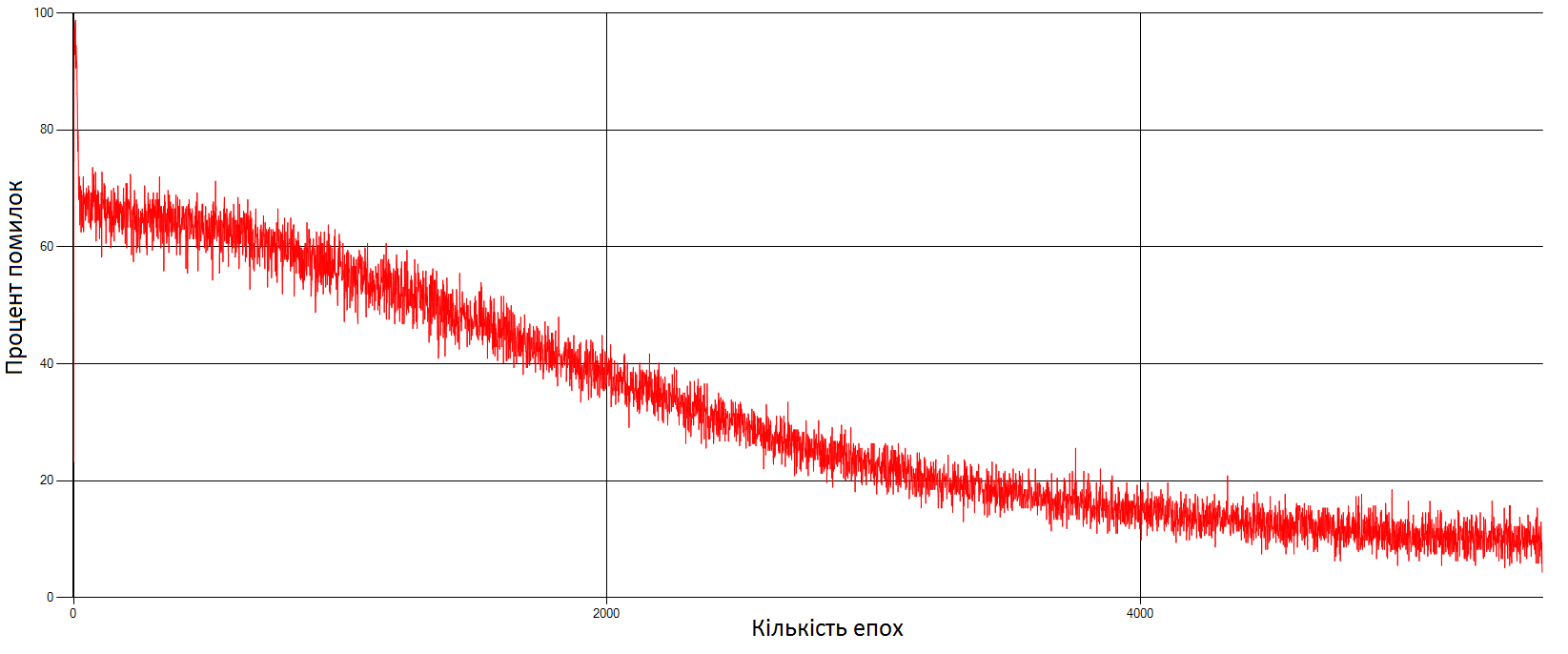


Рис.4.3.5 Процес навчення додаткової нейронної мережі.

**4.4 Висновки**

Розроблена бібліотека класів дає змогу створювати і конфігурувати нейронну мережу і використовувати її для будь-яких задач «навчання з вчитилем». Можна спостерігати, що збіжність результатів нейронної мережі сильно залежить від складеної вибірки і нормалізації даних, яка згідно експериментів дозволила пришвидшити процес навчання в сотні разів.

Розроблена система містить базу даних з великою кількістю навчальних прикладів для нейронної мережі. Також можна спостерігати що основна нейронна мережа містить більше чотирьох тисяч зв’язків що дозволяє досягнути великої точності обрахунків. А використана технологія ADO.NET дозволила ініціалізовувати дані для кожного запиту без втрати швидкодії.

**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. | - | 16128,95 |
| 2. Ціна придбання | грн. | 0 | 20967,64 |
| 3. Річні експлуатаційні витрати | грн. | 41050,8 | 4261,32 |
| 4. Ціна споживання | грн. | 122696,92 | 33704,32 |
| 5. Економічний ефект в сфері експлуатації | грн. | - | 36789,48 |
| 6. Додатковий економічний ефект в сфері експлуатації | грн. | - | 273713,78 |
| 7. Економічний ефект в сфері проектування | грн. | - | -20967,64 |
| 8. Додатковий економічний ефект в сфері проектування | грн. | - | - |
| 9.Термін окупності витрат на проектування рішення | місяці | - | 5 |
| 10.Коефіцієнт конкурентоспроможності |  | - | 0,5 |

**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 - класифікація вхідного вектору який включає в себе дані про користувача. Також було продемонстровано як основна нейронна мережа може бути підкріплена додатковою, шо дозволяє виділити з вхідного вектора основні ознаки.

В роботі було запропоновано вирішення проблеми адаптивності нейронних мереж, та було доведено доцільність такого підходу. За допомогою нормалізації даних, та швидкодії технологій доступу до бази даних, нейронна мережа є достатньо швидкою для того щоб пристосовуватись до нових статистичних даних та до зміни зовнішнього середовища.

Розроблена клієнтська бібліотека дозволяє використовувати її для любої задачі «навчання з вчителем». Основний аспект при створені даної системи було покладено в швидкодії системи, так як це було завжди слабким місцем нейромережевих структур.

Аналіз показує що використання нейронних мереж ходь і потребує обчислювальних ресурсів і часу на навчання, проте при класифікації вхідного вектора уже на навченій мережі, швидкодія такої системи перевищує будь - який алгоритм кластеризації.

Було виявлено, що процес навчання нейронних мереж сильно залежить від адекватності навчальної вибірки і поставленої задачі. Для правильного функціонування нейромережевої структури, образи на координатній площині повинні бути явно виражені, так як в іншому випадку це призведе до втрати швидкодії.

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

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

# [Владимир Круглов](http://www.ozon.ru/person/316755/), [Вадим Борисов](http://www.ozon.ru/person/335556/) - Искусственные нейронные сети. Теория и практика. Горячая линия-Телеком. 2001р. – 382с.

# Владимир Головко - Нейронные сети: обучение, организация и применение. Горячая линия - Телеком 2002р. – 256с

# Данута Рутковская - Нейронные сети, генетические алгоритмы и нечеткие системы. Горячая линия - Телеком 2007р. - 452с.

# Саймон Хайкин - Нейронные сети. Полный курс. Вильямс. 2006р. - 1104с.

# Дмитрий Тархов - Нейросетевые модели и алгоритмы. Справочник. Радиотехника. 2014р. - 352с.

# Аркадий Гелиг - Введение в математическую теорию обучаемых распознающих систем и нейронных сетей. СПб.: Изд-во С.-Петерб. ун-та 2014 – 224с.

# Петер Флах - Наука и искусство построения алгоритмов, которые извлекают знания из данных.ДМК Пресс 2015р. – 400с.

# ASP.NETMVC 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. Искусственные нейронные сети. Теория и практика **Fundamentals of Artificial Neural Networks**. Mohamad H. Hassoun
9. [Vladimir Cherkassky](https://www.amazon.com/s/ref=dp_byline_sr_book_1?ie=UTF8&text=Vladimir+Cherkassky&search-alias=books&field-author=Vladimir+Cherkassky&sort=relevancerank) - Learning from Data: Concepts, Theory, and Methods 2nd Edition. Wiley-IEEE Press. 2007р. – 538с.
10. [Yaser S. Abu-Mostafa](https://www.amazon.com/Yaser-S.-Abu-Mostafa/e/B007K3LFWM/ref=dp_byline_cont_book_1) - Learning From Data. AMLBook. 2012р. – 213с.

**ДОДАТКИ**

**Додаток А.**

**Програмний код системи**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.IO;

using System.Linq;

using NeuralNetworkSystemBLL.Interfaces;

using NeuralNetworkSystemBLL.Interfaces.Components;

using NeuralNetworkSystemBLL.Interfaces.Functions;

using NeuralNetworkSystemBLL.Interfaces.Learning;

using NeuralNetworkSystemBLL.NeuralNetworkComponents;

namespace NeuralNetworkSystemBLL

{

public class NeuralNetwork : INeuralNetwork

{

public IWeightRepository WeightRepository { get; set; }

public ILearningSamplesRepository LeariningSamplesRepository { get; set; }

public List<ILayer> Layers { get; set; }

public ILearningFunctions LearningFunctions { get; set; }

public int MaximumEpochCount { get; set; }

public double NormalizationMinValue { get; set; }

public double NormalizationMaxValue { get; set; }

public double ErrorThreshold { get; set; }

public double ErrorsCountThreshold { get; set; }

private readonly Random \_random;

public NeuralNetwork()

{

\_random = new Random();

}

public ILayer Normalize(ILayer layer)

{

foreach (var neuron in layer.Neurons)

{

if (neuron.ElementIndex == 0)

{

continue;

}

neuron.Value = (neuron.Value - NormalizationMinValue)/

(NormalizationMaxValue - NormalizationMinValue);

}

return layer;

}

public ILayer GetOutputLayer()

{

return Layers.LastOrDefault();

}

public ILayer GetInputLayer()

{

return Layers.FirstOrDefault();

}

public void Calculate(ILayer inputLauer)

{

if (inputLauer.Neurons.Count() != GetInputLayer().Neurons.Count())

{

throw new Exception("Invalid number of input Elements");

}

var layersCount = Layers.Count();

Layers[0] = inputLauer;

int j = 0;

foreach (var neuraon in inputLauer.Neurons)

{

Debug.WriteLine($"Input {j++} - " + neuraon.Value);

}

//Select second layer(maybe output)

for (var i = 1; i < layersCount; i++)

{

var neurons = Layers[i].Neurons.Where(n => !n.IsThreshold);

foreach (var t in neurons)

{

var weights = GetWeighsForNeuronInput(t);

t.CalculateOutput(Layers[i - 1].Neurons, weights);

}

}

}

public INeuralNetwork LearnNetwork()

{

if (WeightRepository == null)

{

throw new NullReferenceException("Weight repository are empty");

}

if (LeariningSamplesRepository == null)

{

throw new NullReferenceException("Learning Samples repository are empty");

}

// NormalizeSamples();

for (var i = 0; i <= MaximumEpochCount;)

{

var errorsCount = 0;

foreach (var sample in LeariningSamplesRepository.LearningSamples)

{

//back propogation

FrontIteration(sample);

if (IsTooBigError())

{

errorsCount++;

}

BackIteration();

WeightRepository.ChangeWeightsAfterIteration();

}

if (CheckToStop(errorsCount))

{

break;

}

LeariningSamplesRepository.LearningSamples = ReorderSamples(LeariningSamplesRepository.LearningSamples);

if (MaximumEpochCount != 0)

{

i++;

}

}

WeightRepository.UpdateWeights();

return this;

}

public void NormalizeSamples()

{

NormalizationMinValue = FoundMinValueInLayer(LeariningSamplesRepository.LearningSamples);

NormalizationMaxValue = FoundMaxValueInLayer(LeariningSamplesRepository.LearningSamples);

foreach (var sample in LeariningSamplesRepository.LearningSamples)

{

sample.InputLayer = Normalize(sample.InputLayer);

}

}

public static double FoundMinValueInLayer(IEnumerable<ILearningSample> samples)

{

var allInputs = new List<double>();

foreach (var sample in samples)

{

allInputs.AddRange(sample.InputLayer.Neurons.Where(n => n.ElementIndex != 0).Select(n => n.Value));

}

var result = allInputs.Min();

return result;

}

public static double FoundMaxValueInLayer(IEnumerable<ILearningSample> samples)

{

var allInputs = new List<double>();

foreach (var sample in samples)

{

allInputs.AddRange(sample.InputLayer.Neurons.Where(n => n.ElementIndex != 0).Select(n => n.Value));

}

var result = allInputs.Max();

return result;

}

private bool IsTooBigError()

{

var outputLayer = GetOutputLayer();

return outputLayer.Neurons.Any(neuron => Math.Abs(neuron.Error) >= ErrorThreshold);

}

private bool CheckToStop(int errorsCount)

{

var samplesCount = Convert.ToDouble(LeariningSamplesRepository.LearningSamples.Count());

var errors = Convert.ToDouble(errorsCount);

var percent = (errors\*100)/samplesCount;

Debug.WriteLine("Error percentage : " + percent);

WriteErrorsToFile(percent);

return !(percent >= ErrorsCountThreshold);

}

private List<ILearningSample> ReorderSamples(List<ILearningSample> samples)

{

var samplesLenght = samples.Count();

for (var i = 0; i < samplesLenght; i++)

{

var source = \_random.Next(0, samplesLenght - 1);

var target = \_random.Next(0, samplesLenght - 1);

var sourceSample = samples[source];

var targetSample = samples[target];

samples[target] = sourceSample;

samples[source] = targetSample;

}

return samples;

}

private List<Weight> GetWeighsForNeuronInput(INeuron neuron)

{

var previousLayerIndex = neuron.LayerIndex - 1;

var currentLayerIndex = neuron.LayerIndex;

var currentNeuronIndex = neuron.ElementIndex;

if (previousLayerIndex < 0)

{

throw new Exception("Neuron should be not in input Layer");

}

var previousLayer = Layers[previousLayerIndex];

var weightList = previousLayer.Neurons

.Select(neuronInLayer =>

WeightRepository

.GetWeight(previousLayerIndex, currentLayerIndex, neuronInLayer.ElementIndex, currentNeuronIndex))

.ToList();

return weightList;

}

private List<Weight> GetWeighsForNeuronOutput(INeuron neuron)

{

var currentLayerIndex = neuron.LayerIndex;

var currentLayer = Layers[currentLayerIndex];

if (currentLayer.IsOutputLayer)

{

return new List<Weight>();

}

var nextLayerIndex = neuron.LayerIndex + 1;

var currentNeuronIndex = neuron.ElementIndex;

var nextLayer = Layers[nextLayerIndex];

return

nextLayer.Neurons.Where(n => !n.IsThreshold)

.Select(

nextNeuron =>

WeightRepository.GetWeight(currentLayerIndex, nextLayerIndex, currentNeuronIndex,

nextNeuron.ElementIndex))

.ToList();

}

private void FrontIteration(ILearningSample sample)

{

Calculate(sample.InputLayer);

var outputLayer = GetOutputLayer();

foreach (var neuron in outputLayer.Neurons)

{

neuron.Error = sample.OutputLayer.Neurons.Single(n => n.ElementIndex == neuron.ElementIndex).Value -

neuron.Value;

}

}

private void BackIteration()

{

var layersCount = Layers.Count;

for (var i = layersCount - 1; i > 0; i--)

{

var neuronsArray = Layers[i].Neurons.Where(n => !n.IsThreshold).ToList();

var neuronsCount = neuronsArray.Count();

for (var j = 0; j < neuronsCount; j++)

{

if (Layers[i].IsOutputLayer)

{

LearningFunctions.GetOutPutGradient(neuronsArray[j]);

}

else

{

LearningFunctions.GetHiddenGradient(neuronsArray[j], GetWeighsForNeuronOutput(neuronsArray[j]),

Layers[i + 1].Neurons.Where(n => !n.IsThreshold).ToList());

}

var previousLayerNeurons = Layers[i - 1].Neurons;

var previousWeights = GetWeighsForNeuronInput(neuronsArray[j]);

var previousWeightCount = previousWeights.Count();

for (var k = 0; k < previousWeightCount; k++)

{

var neuronValue = previousLayerNeurons[k].Value;

var delta = LearningFunctions.LearningSpeed\*neuronsArray[j].Gradient\*neuronValue;

previousWeights[k].NextIterationWeightValue = previousWeights[k].WeightValue + delta;

}

}

}

}

private void WriteErrorsToFile(double percentage)

{

try

{

var filePath = @"D:\\" + Layers.Count() + "Errors.txt";

using (var sw = new StreamWriter(File.Open(filePath, System.IO.FileMode.Append)))

{

sw.WriteLine(percentage);

}

}

catch (Exception)

{

// ignored

}

}

}

}