МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЛЕКЦІЯ 3. Нейронні мережі та глибоке навчання

Львів -- 2025

Лекція зі штучного інтелекту 2025-03

Вступ

На цьому занятті ми розглянемо нейронні мережі — один із найпотужніших та найпоширеніших підходів у сучасному машинному навчанні. Ми ознайомимося з основними концепціями нейронних мереж, їх структурою, алгоритмами навчання та практичними застосуваннями.

Теми, що розглядаються

1. Вступ до нейронних мереж
2. Структура штучного нейрона
3. Багатошарові нейронні мережі
4. Функції активації
5. Алгоритм зворотного поширення помилки (Backpropagation)
6. Глибоке навчання (Deep Learning)
7. Практичні застосування нейронних мереж

Вступ до нейронних мереж

Нейронні мережі — це обчислювальні моделі, натхненні біологічними нейронними мережами людського мозку. Вони складаються з великої кількості взаємопов'язаних штучних нейронів, які здатні навчатися та узагальнювати інформацію.

Основні переваги нейронних мереж:

- Висока здатність до навчання складних нелінійних залежностей
- Можливість роботи з великими обсягами даних
- Гнучкість та універсальність застосування

Структура штучного нейрона

Штучний нейрон — це базовий елемент нейронної мережі, який отримує вхідні сигнали, обчислює зважену суму цих сигналів, додає зміщення (bias) та застосовує функцію активації.

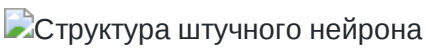
Математично нейрон описується формулою:

[y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)]

де:

- (x_i) — вхідні сигнали
- (w_i) — вагові коефіцієнти
- (b) — зміщення (bias)

- (f(\cdot)) — функція активації
- (y) — вихід нейрона



Багатошарові нейронні мережі

Багатошарові нейронні мережі (Multilayer Neural Networks) складаються з кількох шарів нейронів:

- Вхідний шар (input layer)
- Один або кілька прихованих шарів (hidden layers)
- Вихідний шар (output layer)

Кожен шар повністю або частково з'єднаний з наступним шаром. Наявність прихованих шарів дозволяє мережі навчатися складних нелінійних залежностей.

Популярні архітектури нейронних мереж

Окрім базової структури багатошарових нейронних мереж, існує кілька спеціалізованих архітектур, розроблених для вирішення конкретних задач:

Автоенкодери (Autoencoders)

Автоенкодери — це нейронні мережі, які навчаються ефективному кодуванню даних для зменшення розмірності, а потім відновлення вхідних даних з цього кодування. Вони складаються з двох основних частин:

- **Енкодер**: стискає вхідні дані до прихованого представлення (латентного простору)
- **Декодер**: відновлює вхідні дані з прихованого представлення

Автоенкодери отримали свою назву від поєднання слів "авто" (само-) та "енкодер" (кодувальник), оскільки вони автоматично кодують дані в стиснуте представлення, а потім декодують назад. Ключова особливість автоенкодерів полягає в тому, що вони навчаються без явної розмітки даних (самонавчання), використовуючи вхідні дані як цільові значення.

Принцип роботи автоенкодера

Процес роботи автоенкодера можна розділити на три основні етапи:

1. **Кодування (encoding)**: Вхідні дані (x) перетворюються енкодером у стиснуте представлення (z) (латентний вектор): [z = f_{\text{encoder}}(x)]
2. **Представлення в латентному просторі**: Дані зберігаються у стисненому вигляді (z), який зазвичай має меншу розмірність, ніж вхідні дані.
3. **Декодування (decoding)**: Латентне представлення (z) перетворюється декодером назад у реконструйовані дані (x'): [x' = f_{\text{decoder}}(z)]

Мета навчання автоенкодера — мінімізувати різницю між вхідними даними (x) та реконструйованими даними (x'), тобто мінімізувати функцію втрат: [L(x, x') = |x - x'|^2]

Типи автоенкодерів

1. **Звичайні автоенкодери**: Базова архітектура з повнозв'язними шарами
2. **Розріджені автоенкодери (Sparse Autoencoders)**: Додають регуляризацію для отримання розріджених представлень
3. **Шумопригнічуючі автоенкодери (Denoising Autoencoders)**: Навчаються відновлювати оригінальні дані з зашумлених вхідних даних
4. **Варіаційні автоенкодери (VAE)**: Генеративні моделі, які кодують дані як розподіл у латентному просторі
5. **Згорткові автоенкодери**: Використовують згорткові шари для роботи з зображеннями

Згорткові шари в автоенкодерах

Згорткові автоенкодери використовують згорткові шари (convolutional layers) замість повнозв'язних шарів, що робить їх особливо ефективними для обробки зображень та інших даних з просторовою структурою.

Принцип роботи згорткових шарів

Згорткові шари працюють за принципом ковзного вікна (фільтра), яке переміщується по вхідних даних і виконує операцію згортки:

1. **Фільтр (ядро)**: Невелика матриця ваг, яка "сканує" вхідне зображення
2. **Згортка**: Поелементне множення значень фільтра та відповідної області вхідних даних з подальшим сумуванням
3. **Активація**: Застосування нелінійної функції до результату згортки

Математично операцію згортки для 2D-зображення можна записати як:

$$(I * K)(i, j) = \sum_m \sum_n I(i+m, j+n) \cdot K(m, n)$$

де I — вхідне зображення, K — ядро згортки.

Переваги згорткових шарів в автоенкодерах

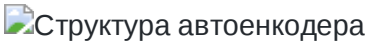
- **Зменшення кількості параметрів**: Завдяки спільному використанню ваг фільтра
- **Збереження просторової інформації**: Важливо для правильного відновлення зображень
- **Інваріантність до зсувів**: Здатність виявляти ознаки незалежно від їх положення

Архітектура згорткового автоенкодера

У згортковому автоенкодері:

- **Енкодер** використовує згорткові шари та операції об'єднання (pooling) для зменшення просторової розмірності
- **Декодер** використовує транспоновані згорткові шари (deconvolution або transposed convolution) для збільшення розмірності та відновлення оригінального зображення

Транспонована згортка виконує операцію, обернену до звичайної згортки, дозволяючи збільшувати просторову розмірність даних.



Застосування автоенкодерів:

- Зменшення розмірності даних
- Видалення шуму з даних
- Виявлення аномалій
- Генерація нових даних

Генеративно-змагальні мережі (GANs)

GANs (Generative Adversarial Networks) складаються з двох нейронних мереж, які змагаються між собою:

- **Генератор**: створює синтетичні дані, намагаючись обманути дискримінатор
- **Дискримінатор**: намагається відрізнити справжні дані від синтетичних

Під час навчання ці дві мережі покращують свої здібності, що призводить до генерації все більш реалістичних даних.

Застосування GANs:

- Генерація реалістичних зображень
- Перенесення стилю
- Створення синтетичних даних для навчання інших моделей
- Відновлення зображень високої роздільної здатності

Мережі з залишковими зв'язками (ResNets)

ResNets (Residual Networks) вирішують проблему зникаючого градієнта в глибоких нейронних мережах шляхом додавання "залишкових зв'язків" (skip connections), які дозволяють сигналу обходити один або кілька шарів.

Основна ідея ResNet полягає в тому, що шари вчаться відображати залишкову функцію F(x) = H(x) - x, замість прямого відображення H(x), де H(x) — бажане відображення.

Ця архітектура дозволяє будувати надзвичайно глибокі мережі (сотні шарів) без проблем з навчанням.

Мережі довгої короткочасної пам'яті (LSTM)

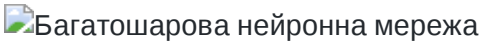
LSTM (Long Short-Term Memory) — це тип рекурентних нейронних мереж, спеціально розроблений для роботи з послідовностями даних та вирішення проблеми зникаючого градієнта в стандартних RNN.

LSTM містять спеціальні "комірки пам'яті", які можуть зберігати інформацію протягом тривалого часу, та "ворота" (gates), які контролюють потік інформації:

- Вхідні ворота (input gate)
- Ворота забування (forget gate)
- Вихідні ворота (output gate)

Застосування LSTM:

- Обробка тексту та мовлення
- Машинний переклад
- Генерація послідовностей
- Прогнозування часових рядів



Функції активації

Функція активації визначає вихід нейрона залежно від його вхідного сигналу. Найпоширеніші функції активації:

- **Сигмоїдна (Sigmoid)**: $f(x) = \frac{1}{1 + e^{-x}}$
- **Гіперболічний тангенс (Tanh)**: $f(x) = \tanh(x)$
- **ReLU (Rectified Linear Unit)**: $f(x) = \max(0, x)$
- **Softmax** (для задач класифікації): $f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Переваги та недоліки функцій активації

Сигмоїдна функція (Sigmoid)

Переваги:

- Гладка функція з плавним градієнтом
- Обмежений вихід (від 0 до 1), що зручно для ймовірнісної інтерпретації
- Історично перша широко використовувана функція активації

Недоліки:

- Проблема зникаючого градієнта при насиченні (коли вхід дуже великий або малий)
- Не центрована відносно нуля, що може ускладнювати навчання
- Обчислювально затратна (використовує експоненту)

Гіперболічний тангенс (Tanh)

Переваги:

- Центрована відносно нуля (вихід від -1 до 1)
- Сильніші градієнти порівняно з сигмоїдною функцією
- Підходить для рекурентних нейронних мереж

Недоліки:

- Все ще схильна до проблеми зникаючого градієнта
- Обчислювально затратна

ReLU (Rectified Linear Unit)

Переваги:

- Обчислювально ефективна (проста операція максимуму)
- Не страждає від проблеми зникаючого градієнта для додатних входів
- Сприяє розрідженості активацій (sparse activations)
- Дозволяє навчати глибокі мережі значно швидше

Недоліки:

- "Проблема мертвого ReLU" — нейрони можуть "вмирати" під час навчання, якщо вхід стає від'ємним
- Не обмежена зверху, що може призводити до вибухових активацій
- Не диференційована в точці 0

Softmax

Переваги:

- Ідеальна для задач багатокласової класифікації
- Виходи інтерпретуються як ймовірності (сума дорівнює 1)
- Підкреслює найбільші значення, пригнічуючи менші

Недоліки:

- Обчислювально затратна для великої кількості класів
- Чутлива до екстремальних значень
- Не підходить для проміжних шарів мережі

Інші популярні функції активації

Leaky ReLU: $f(x) = \max(\alpha x, x)$, де α — малий коефіцієнт (зазвичай 0.01)

- **Переваги**: Вирішує проблему "мертвого ReLU"
- **Недоліки**: Додатковий гіперпараметр α потребує налаштування

ELU (Exponential Linear Unit): $f(x) = x \text{ if } x > 0 \text{ else } \alpha(e^x - 1)$

- **Переваги**: Згладжена версія ReLU з від'ємними значеннями
- **Недоліки**: Обчислювально складніша за ReLU

GELU (Gaussian Error Linear Unit): $f(x) = x \cdot P(X \leq x)$, де P — функція розподілу нормального розподілу

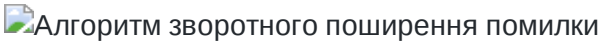
- **Переваги**: Використовується в сучасних трансформерах (BERT, GPT)
- **Недоліки**: Складніша для обчислення

Алгоритм зворотного поширення помилки (Backpropagation)

Backpropagation — це алгоритм навчання нейронних мереж, який дозволяє ефективно обчислювати градієнти функції втрат відносно вагових коефіцієнтів мережі.

Основні етапи алгоритму:

1. Пряме поширення (forward propagation): обчислення виходу мережі
2. Обчислення помилки на виході мережі
3. Зворотне поширення (backward propagation): обчислення градієнтів помилки
4. Оновлення вагових коефіцієнтів за допомогою градієнтного спуску



Глибоке навчання (Deep Learning)

Глибоке навчання — це підхід, що використовує нейронні мережі з великою кількістю прихованих шарів (глибокі нейронні мережі). Такі мережі здатні навчатися дуже складних закономірностей у великих обсягах даних.

Популярні архітектури глибоких нейронних мереж:

- Згорткові нейронні мережі (CNN, Convolutional Neural Networks)
- Рекурентні нейронні мережі (RNN, Recurrent Neural Networks)
- Трансформери (Transformers)

Практичні застосування нейронних мереж

Нейронні мережі широко застосовуються у різних галузях:

- **Комп'ютерний зір**: розпізнавання об'єктів, сегментація зображень
- **Обробка природної мови (NLP)**: машинний переклад, аналіз текстів
- **Розпізнавання мовлення**: голосові асистенти, автоматичні субтитри
- **Рекомендаційні системи**: персоналізовані рекомендації товарів та контенту
- **Автономні транспортні засоби**: розпізнавання дорожніх ситуацій, керування автомобілем

Сучасні інструменти для роботи з нейронними мережами

Розробка та навчання нейронних мереж значно спростилися завдяки спеціалізованим бібліотекам та фреймворкам. Розглянемо найпопулярніші з них.

TensorFlow

TensorFlow — це відкрита бібліотека для машинного навчання, розроблена Google. Вона забезпечує гнучку екосистему інструментів, бібліотек та ресурсів для розробки та впровадження моделей машинного навчання.

Основні особливості TensorFlow:

- **Обчислювальні графи**: представлення обчислень у вигляді графів потоку даних
- **Автоматичне диференціювання**: автоматичне обчислення градієнтів для оптимізації моделей
- **Підтримка GPU та TPU**: прискорення обчислень на графічних та тензорних процесорах

- **TensorFlow Extended (TFX)**: платформа для повного циклу машинного навчання
- **TensorFlow.js**: реалізація для веб-середовища
- **TensorFlow Lite**: оптимізована версія для мобільних та вбудованих пристроїв

PyTorch

PyTorch — це бібліотека машинного навчання з відкритим кодом, розроблена Facebook (Meta). Вона стала надзвичайно популярною серед дослідників завдяки своїй гнучкості та інтуїтивному дизайну.

Основні особливості PyTorch:

- **Динамічні обчислювальні графи**: дозволяють змінювати структуру мережі під час виконання
- **Імперативний стиль програмування**: більш природний для Python-розробників
- **Підтримка розподілених обчислень**: ефективне навчання на кластерах
- **TorchScript**: компіляція моделей для продакшн-середовища
- **Екосистема інструментів**: TorchVision, TorchText, TorchAudio для різних типів даних
- **PyTorch Lightning**: високорівневий інтерфейс для спрощення розробки

Keras

Keras — це високорівневий API для нейронних мереж, який може працювати поверх TensorFlow, Theano або CNTK. Він був розроблений з фокусом на швидке експериментування та простоту використання.

Основні особливості Keras:

- **Простий та інтуїтивний інтерфейс**: швидке створення прототипів моделей
- **Модульність**: легке комбінування шарів та компонентів
- **Розширюваність**: можливість створення власних шарів та функцій
- **Підтримка різних типів мереж**: CNN, RNN, комбіновані архітектури
- **Інтеграція з TensorFlow**: тепер є офіційним високорівневим API для TensorFlow

JAX

JAX — це бібліотека для високопродуктивних числових обчислень, розроблена Google Research. Вона поєднує NumPy з автоматичним диференціюванням для машинного навчання.

Основні особливості JAX:

- **Сумісність з NumPy**: знайомий API для наукових обчислень
- **Автоматичне диференціювання**: обчислення градієнтів будь-якого порядку
- **JIT-компіляція**: прискорення обчислень за допомогою XLA
- **Паралелізм**: ефективне використання багатоядерних процесорів та GPU
- **Функціональний підхід**: чисті функції без побічних ефектів

Hugging Face Transformers

Hugging Face Transformers — це бібліотека, що надає готові до використання моделі трансформерів для різних задач обробки природної мови та комп'ютерного зору.

Основні особливості Hugging Face Transformers:

- **Велика колекція попередньо навчених моделей**: BERT, GPT, T5, ViT та інші
- **Єдиний інтерфейс**: спільний API для різних архітектур
- **Інтеграція з PyTorch та TensorFlow**: підтримка обох фреймворків
- **Model Hub**: платформа для обміну моделями в спільноті
- **Pipelines**: високорівневі абстракції для типових задач

Висновок

У цій лекції ми розглянули основні концепції нейронних мереж, їх структуру, алгоритми навчання та практичні застосування. Нейронні мережі є потужним інструментом для вирішення складних задач машинного навчання та лежать в основі сучасних технологій штучного інтелекту.