КОМП'ЮТЕРНІ ЗАСОБИ, МЕРЕЖІ ТА СИСТЕМИ

УДК 004.4

Я.О. ТУПАЛО

ВИКОРИСТАННЯ МЕТОДІВ МАШИННОГО НАВЧАННЯ НА ПРАКТИЦІ

Вступ. Машинне навчання – це процес, у ході якого система обробляє велику кількість прикладів, виявляє закономірності і використовує їх, щоб прогнозувати характеристики нових даних. На практиці алгоритм машинного навчання буде виконувати таку послідовність дій. Припустимо, є мільярд користувачів і кожен з них повідомив системі назви десяти улюблених фільмів. Це група прикладів. Комп'ютер проаналізує її і визначить, що об'єднує ці фільми. Потім він знайде закономірність у виборі користувачів. Припустимо, тим кому подобаються фільми жахів не подобаються мелодрами, зате люди часто вважають за краще кіно з знайомими акторами. Тепер, якщо комп'ютер дізнається, що вам подобається "Сяйво" з Джеком Ніколсоном, він зробить висновок, що вам варто запропонувати комедію "Любов за правилами і без", в якій також знімався Джек Ніколсон, і зможе рекомендувати інші відео на YouTube. Процес пошуку закономірностей дуже складний. Система шукає їх навіть на рівні пікселів. Наприклад, що потрібно зробити Google Фото, щоб знайти фотографії на запитання "собака"? По-перше, обробити величезну кількість фотографій собак з Інтернету, стільки ж зображень з ярликом "кішка" і мільйоном інших ярликів. По-друге, потрібно знайти послідовності пікселів і поєднання кольорів, характерні для картинок з собаками, кішками та іншими об'єктами.

Спочатку система тільки припускає, які набори даних можуть бути зображеннями собак. Потім вона співвідносить свої здогадки

Y. Tupalo

USING MACHINE LEARNING METHODS IN PRACTICE

The main purpose of this article is to show how it is possible to use methods of machine learning for constructing a system of classification of text.
Key words: mashine learning, TensorFlow.

Рассмотрено как можно использовать методы машинного обучения для построения системы классификации текста.
Ключевые слова: машинное обучение, тензорный поток.

Розглянуто як можна використовувати методи машинного навчання для побудування системи класифікації тексту.
Ключові слова: машинне навчання, тензорний потік.

з фотографією собаки. Якщо виявиться, що система помилково прийняла собаку за кішку, вона виправить закономірності, які використовує для пошуку картинок з собаками і кішками. Це повторюється близько мільярда разів: аналіз зразка – перевірка відповіді – коригування закономірностей. Нарешті закономірності сформують модель – глибоку нейронну мережу, яка в більшості випадків зможе правильно розпізнавати на зображеннях собак, кішок, людей і не тільки. В статті розглядається приклад застосування методів машинного навчання для задачі класифікації текстів.

Основна частина. За майданчик для тестів був використаний суперком'ютер Інституту кібернетики імені В.М. Глушкова НАН України [1], програма написана на мові програмування Python з використанням бібліотеки TensorFlow [2]. Ця бібліотека – це бібліотека з відкритим кодом для машинного навчання, створена Google. Назва допомагає зрозуміти, як з нею працювати: тензори є багатовимірними масивами, які течуть (flow) через вузли графа. Кожне обчислення в TensorFlow представляється як граф потоку даних. У нього є два елементи:

- набір tf.Operation, який представляє одиниці обчислень.
- набір tf.Tensor, який представляє одиниці даних.

Щоб побачити, як це все працює, створимо наступний граф потоку даних (рис. 1).

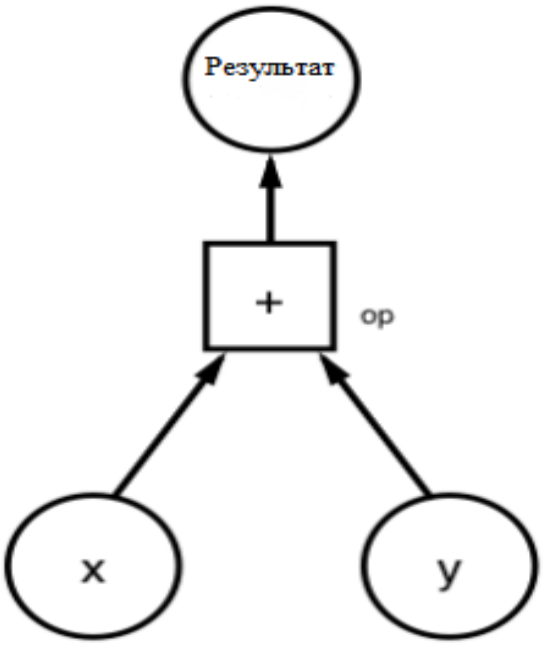

РИС. 1. Граф, що виконує x + y

Визначимо x = [1, 3, 6] і y = [1,1,1]. Так як для подання одиниць даних граф працює з tf.Tensor, створимо тензори-константи:

```
import tensorflow as tf

x = tf.constant([1,3,6])

y = tf.constant([1,1,1])
```

Тепер визначимо одиницю операції:

```
import tensorflow as tf

x = tf.constant([1,3,6])

y = tf.constant([1,1,1])

op = tf.add(x,y)
```

У нас є всі елементи графа. Час його побудувати:

```
import tensorflow as tf

my_graph = tf.Graph()

with my_graph.as_default():

    x = tf.constant([1,3,6])

    y = tf.constant([1,1,1])

    op = tf.add(x,y)
```

Так робочий процес TensorFlow і влаштований: спочатку створюється граф, а потім виконується обчислення, дійсно «запускаючи» вузли графа з операціями. Для цього необхідно створити tf.Session. Об'єкт tf.Session інкапсулює середовище, в якому виконуються об'єкти Operation і оцінюються об'єкти. Щоб зробити це, необхідно визначити, який граф буде використовуватися в сесії:

```
import tensorflow as tf

my_graph = tf.Graph()

with tf.Session(graph=my_graph) as sess:

    x = tf.constant([1,3,6])

    y = tf.constant([1,1,1])

    op = tf.add(x,y)
```

Для виконання операцій використовується метод tf.Session.run (). Він робить один «крок» обчислень TensorFlow, запускаючи необхідний фрагмент графа для виконання кожного об'єкта Operation і оцінки кожного Tensor, переданого в аргументі fetches. У нашому випадку запускається крок операції додавання:

```
import tensorflow as tf

my_graph = tf.Graph()

with tf.Session(graph=my_graph) as sess:

    x = tf.constant([1,3,6])

    y = tf.constant([1,1,1])

    op = tf.add(x,y)

    result = sess.run(fetches=op)

    print(result)

>>> [2 4 7]
```

Тепер, коли відомо, як TensorFlow працює, треба створити модель прогнозування. Процес побудови моделі прогнозування показано на рис. 2.

Можна помітити, що даний процес складається з алгоритму машинного навчання, «натренованого» на дані. З них формується модель прогнозування, далі видається відповідний результат (рис. 3).

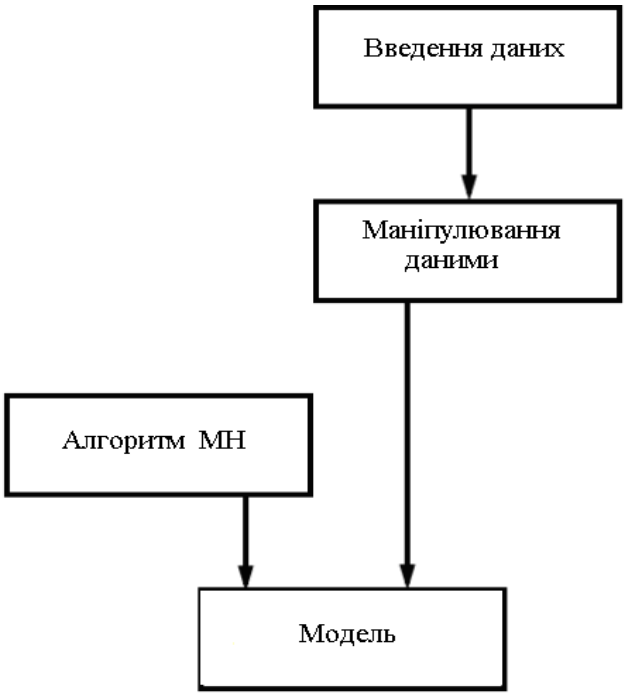

РИС. 2. Процес побудови моделі прогнозування

РИС. 3. Робочий процес прогнозування

Мета моделі, яку ми створимо, буде полягати в класифікації тексту за категоріями і можуть бути встановлені:

- введення: текст;
- результат: категорія.

У нас є набір даних для тренувань, в якому всі тексти позначені (кожна мітка вказує, до якої категорії вона належить). У машинному навчанні такий підхід називається навчанням з учителем. Класифікуються дані за категоріями, отже,

це завдання класифікації [3]. Для створення моделі використовуємо нейронну мережу. У нашій нейронній мережі буде 2 прихованих шари. Завдання кожного прихованого рівня полягає у тому, щоб перетворити вхідні дані в щось, що міг би використовувати шар виведення. Щоб отримати шар виведення будемо використовувати унітарне кодування [4]. Тут тільки 1 біт дорівнює 1, а всі інші – 0. Наприклад, ми хочемо закодувати три категорії: «спорт», «космос» і «комп'ютерна графіка»:

Категорія	Значення
спорт	001
космос	010
комп'ютерна графіка	100

Отримаємо, що число вузлів виведення дорівнює числу класів вхідного набору даних. Значення шару виведення множаться на ваги, до них додається зсув, але функція активації вже інша. Ми хочемо помітити кожен текст категорією, між собою вони є взаємовиключними, тому що текст не може належати двом категоріям одночасно. Щоб досягти мети, замість ReLu візьмемо функцію Softmax. Вона перетворює висновок для кожної категорії в значення між 0 і 1, а також перевіряє, що сума всіх значень дорівнює 1. Так висновок покаже нам ймовірність приналежності тексту до кожної категорії:

```
| 1.2                    0.46|
| 0.9   -> [softmax] ->  0.34|
| 0.4                    0.20|
```

Тепер у нас є граф потоку даних нейронної мережі. Якщо перевести все в код, то вийде наступне:

```
# параметри мережі

n_hidden_1 = 10        # кількість ознак першого шару

n_hidden_2 = 5         # кількість ознак другого шару

n_input = total_words  # слова в словнику

n_classes = 3          # категорії

def multilayer_perceptron(input_tensor, weights, biases):

    layer_1_multiplication = tf.matmul(input_tensor, weights['h1'])
```

```
    layer_1_addition = tf.add(layer_1_multiplication, biases['b1'])

    layer_1_activation = tf.nn.relu(layer_1_addition)

# Прихований шар з RELU активацією

    layer_2_multiplication = tf.matmul(layer_1_activation,
weights['h2'])

    layer_2_addition = tf.add(layer_2_multiplication, biases['b2'])

    layer_2_activation = tf.nn.relu(layer_2_addition)

# Шар виведення з лінійної активацією

    out_layer_multiplication = tf.matmul(layer_2_activation,
weights['out'])

    out_layer_addition = out_layer_multiplication + biases['out']

return out_layer_addition
```

Навчання нейронної мережі. Ваги і зміщення зберігаються в змінних tf.Variable, які містять стан у графі між викликами run (). У машинному навчанні прийнято працювати з вагами і зсувами, отриманими через нормальний розподіл [3]:

```
weights = {

    'h1': tf.Variable(tf.random_normal([n_input, n_hidden_1])),

    'h2': tf.Variable(tf.random_normal([n_hidden_1, n_hidden_2])),

    'out': tf.Variable(tf.random_normal([n_hidden_2, n_classes]))

}

biases = {

    'b1': tf.Variable(tf.random_normal([n_hidden_1])),

    'b2': tf.Variable(tf.random_normal([n_hidden_2])),

    'out': tf.Variable(tf.random_normal([n_classes]))

}
```

Запустимо мережу в перший раз з вагами, отриманими за нормальним розподілом:

```
input values: x

weights: w

bias: b

output values: z

expected values: expected
```

Щоб дізнатися, чи вчиться мережа, необхідно порівняти вихідні значення z з очікуваними значеннями expected. Існує багато методів, як порахувати втрату loss. Так як ми працюємо з завданням на класифікацію, найкращим способом обчислення помилки буде перехресної ентропії [3]. Зробимо це за допомогою TensorFlow, використовуючи метод tf.nn.softmax_cross_entropy_with_logits () (функцію активації softmax), і обчислимо середню помилку tf.reduced_mean ():

```
# конструювання моделі

prediction = multilayer_perceptron(input_tensor, weights, biases)

# визначення втрат

entropy_loss =
tf.nn.softmax_cross_entropy_with_logits(logits=prediction,
labels=output_tensor)

loss = tf.reduce_mean(entropy_loss)
```

Завдання знайти найкращі значення ваг і зміщень для того, щоб мінімізувати помилки при виведенні – різницю між отриманим і правильним значеннями. Для цього скористаємося методом градієнтного спуску. А якщо бути точніше, то стохастическим градієнтним спуском [3]. Існує безліч алгоритмів для обчислення градієнтного спуску, будемо використовувати адаптивну оцінку моментів. Передамо значення learning_rate, яке визначає крок значень для знаходження кращої ваги.

```
learning_rate = 0.001

# конструювання моделі
```

```
prediction = multilayer_perceptron(input_tensor, weights, biases)

# визначення втрати

entropy_loss =
tf.nn.softmax_cross_entropy_with_logits(logits=prediction,
labels=output_tensor)

loss = tf.reduce_mean(entropy_loss)

optimizer =
tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)
```

Для тестування моделі ми наповнимо словник великого блоку, тому необхідно визначити його змінне значення. Функція get_batches () показує кількість текстів разом з розміром блоку. Тепер можна запустити модель:

```
training_epochs = 10

# запуск графа

with tf.Session() as sess:

    sess.run(init) # ініціалізація нормальним розподілом

    # тренувальний цикл

    for epoch in range(training_epochs):

        avg_cost = 0.

        total_batch = int(len(newsgroups_train.data)/batch_size)

        # цикл по всім блокам

        for i in range(total_batch):

            batch_x,batch_y =
get_batch(newsgroups_train,i,batch_size)

            # запустимо оптимізацію

            c,_ = sess.run([loss,optimizer], feed_dict={input_tensor:
batch_x, output_tensor:batch_y})
```

Тепер у нас є натренована модель. Щоб протестувати її, необхідно створити елементи графа. Будемо вимірювати точність, треба отримати індекси прогнозованого значення й індекс правильного значення, тому що ми використовуємо унітарне кодування. Потім перевірити, чи рівні вони, і обчислити середнє для всього тестового набору даних.

Висновки. В даній статі автор зробив дослідження, які показують як можна використовувати методи машинного навчання для побудови інтелектуальної системи, яка може класифікувати тексти. Етапи даного дослідження представлені фрагментами коду, які без проблем можна використовувати для задач другого типу. Даний алгоритм був протестований на суперкомп'ютері Інституту кібернетики імені В.М. Глушкова НАН України.

СПИСОК ЛІТЕРАТУРИ

1. Головинський А.Л., Маленко А.Л., Сергієнко І.В., Тульчинський В.Г. Енергоефективний суперкомп'ютер СКІТ-4. *Вісник НАН України*. 2013. № 2. С. 50–59.
2. https://www.tensorflow.org/
3. Mastering Machine Learning Algorithms: Expert techniques to implement popular machine learning algorithms and fine-tune your models Paperback – Giuseppe Bonaccorso.
4. https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D1%82%D0%B0%D1%80%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4

Одержано 24.10.2018