ЗВІТ

з навчального курсу «СТАТИСТИЧНІ АЛГОРИТМИ НАВЧАННЯ»

ВАРІАНТ – 8

студентки 2 курсу ОР «Магістр»
механіко – математичного факультету
спеціальності «Прикладна та теоретична статистика»
Дегтяр Олени

ЛАБОРАТОРНА РОБОТА № 3

«Згорткові нейронні мережі»

1) Завантажте датасет, підготуйте його для входу в нейронну мережу.

Варіант 2 `http://help.sentiment140.com/for-students`

The data is a CSV with emoticons removed. Data file format has 6 fields:
0 - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
1 - the id of the tweet (2087)
2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009)
3 - the query (lyx). If there is no query, then this value is NO_QUERY.
4 - the user that tweeted (robotickilldozr)
5 - the text of the tweet (Lyx is cool)

```
pd.read_csv("train_data.csv",encoding='latin-1', header = None)
```

	0	1	2	3	4	5
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
...	...	...	...	...	...	...
1599995	4	2193601966	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	AmandaMarie1028	Just woke up. Having no school is the best fee...
1599996	4	2193601969	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	TheWDBoards	TheWDB.com - Very cool to hear old Walt interv...
1599997	4	2193601991	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	bpbabe	Are you ready for your MoJo Makeover? Ask me f...
1599998	4	2193602064	Tue Jun 16 08:40:49 PDT 2009	NO_QUERY	tinydiamondz	Happy 38th Birthday to my boo of alll time!!! ...
1599999	4	2193602129	Tue Jun 16 08:40:50 PDT 2009	NO_QUERY	RyanTrevMorris	happy #charitytuesday @theNSPCC @SparksCharity...

1600000 rows × 6 columns

```python
def preprocessing_data(data, len_set):
    data = pd.read_csv(data,encoding='latin-1', header = None)
    lab0 = data[data[0] == 0][:len_set]
    lab4 = data[data[0] == 4][:len_set]
    data = pd.concat([lab0,lab4])
    data = data.reset_index()
    data.pop('index')
    tweets = []
    for row in range(len(data)):
        tweets.append((data[5][row], data[0][row]))
    filtered_tweets = []
    X = []
    y = []
    porter=PorterStemmer()
    sc = StandardScaler()
    for (words, sentiment) in tweets:
        words_filtered = [e.lower() for e in words.split() if len(e) >= 3]
        words_filtered = list(filter(lambda x: not x.startswith('http') and not x.startswith('@'), words_filtered))
        words_filtered = [re.sub(r"\p{P}+", "", word) for word in words_filtered ]
        stem_sentence=[]
        for word in words_filtered:
            stem_sentence.append(porter.stem(word))
        X.append(stem_sentence)
        y.append(sentiment)
    X = get_X_matrix(X)
    return sc.fit_transform(X),np.array(y)
```

```
("@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You shoulda got David Carr of Third Day to do it. ;D", 0)
```

↓

```
['awww', 'thats', 'bummer', 'you', 'shoulda', 'got', 'david', 'carr', 'third', 'day', 'it']
```

↓

```
array([ 0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
        0.        ,  0.        ,  0.        , -0.25242374, -0.29513458,
       -0.3358429 , -0.52033963, -0.4618172 , -0.52435527, -0.61334922,
       -0.54774867,  0.28358308, -0.58173595,  0.00807338, -0.65008127,
       -0.09778573, -0.63240708, -0.17972438, -0.27208858, -0.32101304,
       -0.46026206, -1.75605401])
```

2) Натренуйте власне вкладення (embedding), проаналізуйте як якість алгоритму змінюється зі зміною розмірності латентного простору?

```python
model = tf.keras.models.Sequential([
  #tf.keras.layers.Input(shape=(500,)),
  #tf.keras.layers.Embedding(10000, 50, input_length=500),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation=tf.nn.relu),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10)
model.evaluate(X_test, y_test)
```

```
Epoch 1/10
625/625 [==============================] - 2s 2ms/step - loss: 0.8174 - accuracy: 0.5431
Epoch 2/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6766 - accuracy: 0.5605
Epoch 3/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6713 - accuracy: 0.5616
Epoch 4/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6689 - accuracy: 0.5630
Epoch 5/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6661 - accuracy: 0.5664
Epoch 6/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6648 - accuracy: 0.5637
Epoch 7/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6623 - accuracy: 0.5705
Epoch 8/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6607 - accuracy: 0.5693
Epoch 9/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6590 - accuracy: 0.5702
Epoch 10/10
625/625 [==============================] - 1s 2ms/step - loss: 0.6587 - accuracy: 0.5723
12/12 [==============================] - 0s 2ms/step - loss: 0.7854 - accuracy: 0.3677
[0.7854083180427551, 0.36768803000450134]
```

```
model = tf.keras.models.Sequential([
      #tf.keras.layers.Flatten(),
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 64, input_length=32),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      #tf.keras.layers.Dropout(0.5),
      tf.keras.layers.Dense(32, activation=tf.nn.softmax),
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=64)
```

```
Epoch 16/20
32/32 [==============================] - 0s 5ms/step - loss: 0.6809 - accuracy: 0.5505
Epoch 17/20
32/32 [==============================] - 0s 5ms/step - loss: 0.6768 - accuracy: 0.5600
Epoch 18/20
32/32 [==============================] - 0s 6ms/step - loss: 0.6716 - accuracy: 0.5605
Epoch 19/20
32/32 [==============================] - 0s 5ms/step - loss: 0.6707 - accuracy: 0.5650
Epoch 20/20
32/32 [==============================] - 0s 5ms/step - loss: 0.6729 - accuracy: 0.5505
```

```
model.evaluate(X_test, y_test, batch_size=64)
```

```
6/6 [==============================] - 0s 2ms/step - loss: 0.6962 - accuracy: 0.6407
```

Далі я обрала на роль функції втрат я вирішила використовувати binary_crossentropy, оскільки ми маємо справу з бінарною класифікацією і також саме цю функцію втрат радить низка статтей.

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 64, input_length=32),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64)
model.evaluate(X_test, y_test, batch_size=64)
```

```
Epoch 1/5
32/32 [==============================] - 1s 6ms/step - loss: 0.6952 - accuracy: 0.4860
Epoch 2/5
32/32 [==============================] - 0s 6ms/step - loss: 0.6928 - accuracy: 0.5205
Epoch 3/5
32/32 [==============================] - 0s 5ms/step - loss: 0.6929 - accuracy: 0.5080
Epoch 4/5
32/32 [==============================] - 0s 5ms/step - loss: 0.6909 - accuracy: 0.5290
Epoch 5/5
32/32 [==============================] - 0s 5ms/step - loss: 0.6885 - accuracy: 0.5275
6/6 [==============================] - 0s 2ms/step - loss: 0.6816 - accuracy: 0.6546
[0.6815520524978638, 0.6545960903167725]
```

Спроба збільшити розмірність латентного простору не дала покращення результатів:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 128, input_length=32),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64)
model.evaluate(X_test, y_test, batch_size=64)
```

```
Epoch 1/5
32/32 [==============================] - 1s 8ms/step - loss: 0.6953 - accuracy: 0.4920
Epoch 2/5
32/32 [==============================] - 0s 8ms/step - loss: 0.6945 - accuracy: 0.4855
Epoch 3/5
32/32 [==============================] - 0s 8ms/step - loss: 0.6927 - accuracy: 0.5130
Epoch 4/5
32/32 [==============================] - 0s 13ms/step - loss: 0.6920 - accuracy: 0.5445
Epoch 5/5
32/32 [==============================] - 0s 9ms/step - loss: 0.6893 - accuracy: 0.5460
6/6 [==============================] - 0s 3ms/step - loss: 0.6796 - accuracy: 0.6267
[0.6796187162399292, 0.6267409324645996]
```

Як і спроба зменшити його розмірність:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 8, input_length=32),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64)
model.evaluate(X_test, y_test, batch_size=64)
```

```
Epoch 1/5
32/32 [==============================] - 1s 3ms/step - loss: 0.6943 - accuracy: 0.4830
Epoch 2/5
32/32 [==============================] - 0s 2ms/step - loss: 0.6932 - accuracy: 0.5005
Epoch 3/5
32/32 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.5245
Epoch 4/5
32/32 [==============================] - 0s 2ms/step - loss: 0.6914 - accuracy: 0.5445
Epoch 5/5
32/32 [==============================] - 0s 2ms/step - loss: 0.6910 - accuracy: 0.5230
6/6 [==============================] - 0s 2ms/step - loss: 0.6861 - accuracy: 0.4596
[0.6861168742179871, 0.4596100151538849]
```

На жаль, як і спроба додати LSTM:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 64, input_length=32),
      tf.keras.layers.LSTM(8, dropout=0.2),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64)
model.evaluate(X_test, y_test, batch_size=64)
```

```
Epoch 1/5
32/32 [==============================] - 2s 19ms/step - loss: 0.6938 - accuracy: 0.4860
Epoch 2/5
32/32 [==============================] - 1s 18ms/step - loss: 0.6934 - accuracy: 0.4985
Epoch 3/5
32/32 [==============================] - 1s 17ms/step - loss: 0.6934 - accuracy: 0.4860
Epoch 4/5
32/32 [==============================] - 1s 17ms/step - loss: 0.6931 - accuracy: 0.4895
Epoch 5/5
32/32 [==============================] - 1s 18ms/step - loss: 0.6932 - accuracy: 0.5070
6/6 [==============================] - 0s 5ms/step - loss: 0.6895 - accuracy: 0.6351
[0.6894552707672119, 0.6350975036621094]
```

За допомогою блоків GRU вдалося трохи підвищити точність:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 64, input_length=32),
      tf.keras.layers.GRU(4),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=64)
model.evaluate(X_test, y_test, batch_size=64)
```

```
Epoch 1/5
32/32 [==============================] - 3s 16ms/step - loss: 0.6934 - accuracy: 0.4970
Epoch 2/5
32/32 [==============================] - 1s 18ms/step - loss: 0.6934 - accuracy: 0.4985
Epoch 3/5
32/32 [==============================] - 1s 16ms/step - loss: 0.6933 - accuracy: 0.4880
Epoch 4/5
32/32 [==============================] - 1s 17ms/step - loss: 0.6932 - accuracy: 0.5070
Epoch 5/5
32/32 [==============================] - 1s 16ms/step - loss: 0.6932 - accuracy: 0.5010
6/6 [==============================] - 0s 4ms/step - loss: 0.6926 - accuracy: 0.6713
[0.6926321983337402, 0.6713091731071472]
```

Далі я ще спробувала додати рекурентні шари і декілька нових параметрів у функцію fit():

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(100,)),
      tf.keras.layers.Embedding(1000, 32, input_length=100),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(50, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.5),
      tf.keras.layers.Dense(50, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.3),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=32, shuffle = True, verbose=1)
model.evaluate(X_test, y_test, batch_size=32,verbose=1)
```

```
Epoch 1/10
63/63 [==============================] - 1s 4ms/step - loss: 0.6957 - accuracy: 0.4920
Epoch 2/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6942 - accuracy: 0.5005
Epoch 3/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6941 - accuracy: 0.4865
Epoch 4/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 5/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6931 - accuracy: 0.5010
Epoch 6/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6918 - accuracy: 0.5285
Epoch 7/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6901 - accuracy: 0.5260
Epoch 8/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6872 - accuracy: 0.5380
Epoch 9/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6844 - accuracy: 0.5435
Epoch 10/10
63/63 [==============================] - 0s 4ms/step - loss: 0.6807 - accuracy: 0.5410
12/12 [==============================] - 0s 2ms/step - loss: 0.6588 - accuracy: 0.7047
[0.6588070392608643, 0.7047353982925415]
```

Найкращий результат вийшов з такими параметрами:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(32,)),
      tf.keras.layers.Embedding(1000, 64, input_length=32),
      #tf.keras.layers.GRU(),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.5),
      tf.keras.layers.Dense(50, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.3),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, batch_size=100, shuffle = True, verbose=1)
model.evaluate(X_test, y_test, batch_size=100,verbose=1)
```

```
Epoch 1/5
20/20 [==============================] - 1s 9ms/step - loss: 0.6973 - accuracy: 0.4965
Epoch 2/5
20/20 [==============================] - 0s 9ms/step - loss: 0.6949 - accuracy: 0.4955
Epoch 3/5
20/20 [==============================] - 0s 9ms/step - loss: 0.6944 - accuracy: 0.4900
Epoch 4/5
20/20 [==============================] - 0s 6ms/step - loss: 0.6931 - accuracy: 0.5030
Epoch 5/5
20/20 [==============================] - 0s 7ms/step - loss: 0.6925 - accuracy: 0.5090
4/4 [==============================] - 0s 3ms/step - loss: 0.6897 - accuracy: 0.7326
[0.689702570438385, 0.7325905561447144]
```

Ще його звісно можна підвищити шляхом збільшення початкової вибірки з датасету:

```
model = tf.keras.models.Sequential([
      tf.keras.layers.Input(shape=(100,)),
      tf.keras.layers.Embedding(1000, 32, input_length=100),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(50, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.5),
      tf.keras.layers.Dense(50, activation=tf.nn.relu),
      tf.keras.layers.Dropout(0.3),
      tf.keras.layers.Dense(1, activation=tf.nn.sigmoid),
])
model.compile(optimizer=RMSprop(), loss='binary_crossentropy', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=100, shuffle = True, verbose=1)
model.evaluate(X_test, y_test, batch_size=100,verbose=1)
```

```
Epoch 1/10
2000/2000 [==============================] - 12s 6ms/step - loss: 0.6931 - accuracy: 0.5068
Epoch 2/10
2000/2000 [==============================] - 13s 7ms/step - loss: 0.6924 - accuracy: 0.5155
Epoch 3/10
2000/2000 [==============================] - 12s 6ms/step - loss: 0.6922 - accuracy: 0.5166
Epoch 4/10
2000/2000 [==============================] - 11s 6ms/step - loss: 0.6919 - accuracy: 0.5197
Epoch 5/10
2000/2000 [==============================] - 11s 6ms/step - loss: 0.6919 - accuracy: 0.5197
Epoch 6/10
2000/2000 [==============================] - 12s 6ms/step - loss: 0.6918 - accuracy: 0.5195
Epoch 7/10
2000/2000 [==============================] - 13s 6ms/step - loss: 0.6920 - accuracy: 0.5190
Epoch 8/10
2000/2000 [==============================] - 12s 6ms/step - loss: 0.6919 - accuracy: 0.5186
Epoch 9/10
2000/2000 [==============================] - 12s 6ms/step - loss: 0.6920 - accuracy: 0.5180
Epoch 10/10
2000/2000 [==============================] - 11s 6ms/step - loss: 0.6921 - accuracy: 0.5182
4/4 [==============================] - 0s 3ms/step - loss: 0.6654 - accuracy: 0.7855
[0.665386974811554, 0.785515308380127]
```

ВИСНОВКИ:

У процесі виконання роботи максимальна точність, яку вдалося досягнути, становила **78,6%**. Для цього було протестовано роботу моделі із різними значеннями параметрів та рекурентними шарами нейронної мережі.

За наявних умов цієї лабораторної роботи збільшення або зменшення кількості **епох** несуттєво впливало на кінцеву точність.

Зміна **batch size** несуттєво покращувала точність, але збільшення цього параметру збільшувало й перенавчання, яке простежувалося протягом виконання роботи.

Теоретично додавання **LSTM** мало б позитивно вплинути на результат і використання функції **RMSprop()** у якості оптимайзера, але практично цього зафіксувати не вдалося, хоча з їх використанням при різних запусках точність змінювалася не суттєво і перенавчання не спостерігалося, як і в мене у цьому семестрі :)