ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ TENSORFLOW ДЛЯ РАЗРАБОТКИ МОДЕЛИ ПРОГНОЗИРОВАНИЯ

И.С. Власенко
Томский политехнический университет
E-mail: isv10@tpu.ru

Введение

На сегодняшний день TensorFlow – это самый популярный фреймворк для решения задач машинного обучения и создания нейронных сетей. Бэкенд его написан на C++, но для управления обычно используется Python. TensorFlow решает вычислительные задачи посредством графического представления. Такой подход позволяет ассоциировать математические операции как элементы графов данных, операторов и переменных. Так как нейронные сети по сути представляют собой графы данных, Tensorflow отлично подходит для их создания.

Подготовка данных

Для изучения библиотеки Tensorflow были выбранны биржевые данные хранящиеся в csv-файле. Этот датасет содержит n = 41266 минут данных, охватывающих торги 500 акциями в период с апреля по август 2019. Данные брались из сервиса Kaggle для специалистов по машинному обучению.

На рисунке 1 временной ряд индекса S&P, построенный с помощью pyplot.plot(data['SP500']):

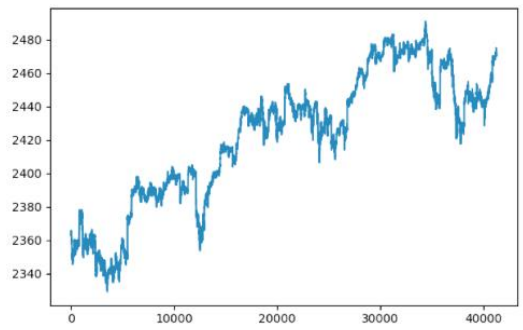

Рис. 1. Временной ряд данных

Набор данных был разбит на два — одна часть для тестирования, а вторая для обучения. При этом, данные для обучения составили 80% от всего их объема, а остальные 20% - для тестирования.

Большинство архитектур нейронных сетей используют масштабирование входных и выходных данных. В настоящее время, наиболее часто используются активации выпрямленной линейной единицей (ReLU):

```
scaler.fit(data_train)
data_train = scaler.transform(data_train)
data_test = scaler.transform(data_test)
```

Плейсхолдеры

Плейсхолдеры – инструмент Tensorflow, использующийся как константы. Инициализация плейсхолдеров происходит следующим образом:

```
X = tf.placeholder(dtype=tf.float32, shape=[None, n_stocks])
Y = tf.placeholder(dtype=tf.float32, shape=[None])
```

Помимо плейсхолдеров, в TensorFlow есть такой важный элемент как переменные. Их различие в следующем: плейсхолдеры используются для хранения неизменных входных данных в графе, а переменные служат контейнерами внутри графа, которые могут изменяться. Так как веса и смещения приходится адаптировать в процессе обучения, они представлены переменными. Перед началом обучения переменные необходимо инициализировать.

Модель состоит из четырех скрытых уровней. Первый содержит 1024 нейрона, что чуть более чем в два раза превышает объем входных данных. Последующие скрытые уровни всегда в два раза меньше предыдущего – они содержат 512, 256 и 128 нейронов. Ниже представлен фрагмент для первого уровня:

```
W_hidden_1 = tf.Variable(weight_initializer([n_stocks, n_neurons_1]))
bias_hidden_1 = tf.Variable(bias_initializer([n_neurons_1]))
```

Разработка архитектуры сети

Плейсхолдеры и переменные необходимо объединить в систему последовательных матричных умножений. Функции активации – это важный элемент сетевой инфраструктуры, которые трансформируют скрытые уровни данной сети и вносят нелинейность в систему. Существуют большое количество функций активации, и одна из самых распространенных – выпрямленная линейная единица (rectified linear unit, ReLU):

```
hidden_1 = tf.nn.relu(tf.add(tf.matmul(X, W_hidden_1), bias_hidden_1))
```

Представленное на рисунке 2 изображение иллюстрирует архитектуру сети. Модель состоит из трех главных блоков. Уровень входных данных, скрытые уровни и выходной уровень. Такая инфраструктура называется упреждающей сетью (feedforward network). Это означает что куски данных продвигаются по структуре строго слева-направо. При других реализациях, например, в случае рекуррентных нейронных сетей, данные могут перетекать внутри сети в разные стороны.

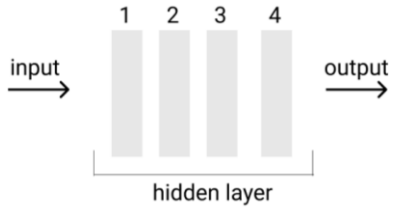

Рис. 2. Архитектура сети

Функция стоимости

Функция стоимости используется для оценки отклонения между прогнозами модели и реальными данными наблюдений. Обычно в качестве такой функции используют MSE - функцию средней квадратичной ошибки. Данная функция вычисляет среднее квадратичное отклонение между целями и предсказаниями.

```
mse = tf.reduce_mean(tf.squared_difference(out, Y))
```

Оптимизатор

Оптимизаторы необходимы для вычислений, которые нужны для адаптации весов и переменных отклонений нейросети. Результат этих вычислений - так называемые градиенты, обозначающие направление изменения отклонений и весов, необходимое для минимизации функции стоимости.

```
optim = tf.train.AdamOptimizer().minimize(mse)
```

В данном примере используется оптимизатор Adam Optimizer, который является наиболее популярным в сфере машинного обучения.

Инициализаторы

Инициализаторы выполняют функцию инициализации переменных перед началом обучения. В данном случае использован tf.variance_scaling_initializer(), который реализует одну из стандартных стратегий инициализации:

```
weight_initializer = tf.variance_scaling_initializer(mode="fan_avg", distribution="uniform", scale=sigma)
bias_initializer = tf.zeros_initializer()
```

Настройка нейросети

После того, как определены плейсхолдеры, переменные, инициализаторы, функции стоимости и оптимизаторы, модель необходимо обучить. Обычно для этого используется подход мини-партий. В ходе такого обучения из набора данных для обучения отбираются случайные комплекты данных размера n = batch_size и загружаются в нейросеть. Затем в сеть последовательно отправляются n / batch_size частей данных. Далее плейсхолдеры X и Y отправляют данные в нейросеть.

Данные X, заранее разбитые на части, до выходного уровня проходят по сети. Затем TensorFlow сопоставляет прогнозы, сгенерированные моделью с реально наблюдаемыми в текущем цикле данными Y. Затем идет этап оптимизации. На нём обновляются веса и отклонения. После обновления, процесс повторяется для нового куска данных. Данная процедура повторяется до тех пор, пока все части данных не будут отправлены в нейросеть. Полный цикл такой обработки называется «эпохой».

Критерием остановки обучения является либо достижение максимального числа эпох, либо срабатывание другого заранее определенного критерия остановки.

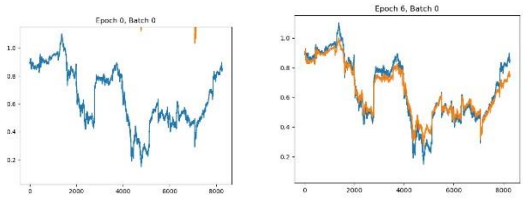

Рис. 3-4. Обучение на эпохе 0 и эпохе 6

Как видно на рисунках 3-4, нейросеть быстро адаптируется к базовой форме временного ряда и продолжает искать наилучшие паттерны данных. После прохождения 7 эпох мы получаем результаты, очень близкие к тестовым данным. Финальное значение функции MSE составляет 0.0021132957. Средняя абсолютная процентная погрешность прогноза на тестовом наборе равняется 7,93%. Важно понимать, что это лишь совпадение с тестовыми, а не реальными данными.

Заключение

В результате проведения тестирования можно сделать вывод о том, что разработанная с помощью Tensorflow модель является работоспособной. Результаты предсказания данной модели имеют погрешность 7,93%, что является хорошим результатом.

Список использованных источников

1. Руководство по Python [Электронный ресурс]. – URL: https://www.python.org (дата обращения 20.12.2019).
2. Руководство по Tensorflow [Электронный ресурс]. – URL: https://www.tensorflow.org (дата обращения 20.12.2019).
3. Социальная сеть специалистов по обработке данных и машинному обучению. [Электронный ресурс]. – URL: https://www.kaggle.com (дата обращения 20.12.2019).
4. Хайкин С. Нейронные сети: полный курс. – М.: Вильямс, 2006. – 1104 с.
5. Введение в нейросети [Электронный ресурс]. – URL: https://python-scripts.com/intro-to-neural-networks (дата обращения 20.12.2019).