Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет Компьютерных сетей и систем

Кафедра Информатики

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по курсу «Интеллектуальный анализ информации»

Исследование возможностей библиотеки Keras

Студент:
гр. 758641
Ярош Г.И.

Проверил:
Ивашенко В.П.

Минск, 2018

СОДЕРЖАНИЕ

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ БИБЛИОТЕКИ KERAS ДЛЯ ПОСТРОЕНИЯ МОДЕЛЕЙ НЕЙРОННЫХ СЕТЕЙ

1. Цель

Ознакомиться и проанализировать средства библиотеки Keras для построения нейросетевых моделей. Привести краткие сведения по основным методам построения моделей нейронных сетей.

2. Краткие сведения о библиотеке Keras.

Keras является высокоуровневой библиотекой для построения нейронных сетей с помощью языка программирования Python. На более низком уровне Keras использует библиотеку TensorFlow , CNTK или Theano.

Руководящие принципы:

- Удобство для пользователя. Keras предоставляет удобный высокоуровневый API для быстрого проектирования и обучения нейронных сетей удобный для пользователей.
- Модульность. Библиотека предоставляет набор модулей реализующих типы слоев нейросетей, функций активации, методов обучения и т.д. Все предоставленные модули легко модифицируются и расширяются.
- Работа с Python. Работа с Keras может производиться исключительно с помощью языка Python. Разработчику не нужно писать отдельные модули на других, более низкоуровневых языках.

3. Построение моделей нейронных сетей с помощью библиотеки

Рассмотрим пример построения линейной рециркуляционной сети из первой лабораторной работы с помощью библиотеки Keras:

```
input_layer = layers.Input((n,))
hidden_layer = layers.Dense(k, activation='linear')(input_layer)
output_layer = layers.Dense(n, activation='linear')(hidden_layer)
autoencoder = models.Model(inputs=[input_layer], outputs=[output_layer])

optimizer = optimizers.Adagrad(lr=lr)
autoencoder.compile(optimizer=optimizer, loss='mean_squared_error')
autoencoder.summary()
```

Модуль `keras.layers` предоставляет классы для создания слоев нейросети. В примере создается три полносвязных слоя. Первым параметром задается количество нейронов в слое. Вторым – функция активации. Созданный слой может быть вызван,

как функция. Параметром можно передать предыдущий слой. Таким образом формируется последовательность слоев.

В модуле `keras.layers` содержаться многие необходимые виды слоев. Некоторые из них приведены ниже:

- `keras.layers.Dense` – обычный полносвязный слой;
- `keras.layers.Flattern` – слой, преобразующий многомерные данные в одномерный вектор значений;
- `keras.layers.Permute` – слой, предназначенный для перемешивания входных даннхых;
- `leras.layers.Dropout` – слой, обнуляющий некоторый процент входных значений;
- `keras.layers.Conv1D, keras.layers.Conv2D` – сверточные слои (одномерный и двухмерный вход соответственно);
- `keras.layers.BatchNormalization` – слой, преобразующий входные данные к интервалу [0, 1] для каждого батча данных;
- `keras.layers.MaxPooling1D, keras.layers.MaxPooling2D` – слои, уменьшаяющие размер входных данный путем разбиения данных на участки и выбора максимальных значений на них.

Модуль keras.activations содержит функции активации, такие как: `softmax, elu, softplus, softsign, relu, tanh, sigmoid`. Для настройки функции активации для слоя достаточно передать параметр `activation` со строковым значением фнуикции активации. Если же нужно настроить параметры функции активации, то необходимо импортировать класс, реализующий функцию и передать его экземпляр в параметр `activation`.

Для создания модели необходимо использовать класс `keras.models.Model`. В его конструктор необходимо передать входной и выходной слои. Далее необходимо вызвать функцию `compile`. В нее нужно передать метод оптимизации и функцию подсчета ошибки. Также в метод можно передать список дополнительных метрик, подсчитываемых при обучении нейросети.

Библиотека Keras предоставляет следующие методы оптимизации:

- `keras.optimize.SGD` - оптимизация с помощью алгоритма стохастического градиетнтного спуска;
- `keras.optimize.Adagrad` – оптимизация с помощью метода градиетного спуска с адаптируемым шагом обучения. Шаг обучения настраивается относительно того, как часто меняется значение ошибки в течении одной эпохи;
- `keras.optimize.Adam` – оптимизация с помощью метода градиентного спуска с адаптивным шагом обучения с моментами.

Модуль `keras.losses` содержит функции подсчета ошибки, такие как: `mean_squared_error, mean_absolute_error, hinge, squared_hinge, binary_crossentropy, categorical_crossentropy, logcosh`.

4. *Обучение нейронный сетей в Keras*

Созданную нейронную сеть можно обучить с помощью функции `fit`. На вход она принимает входную выборку и желаемую выходную выборку. Параметр `epohs` задает количество эпох обучения. Параметр `validation_split` задает процент выборки, которая будет использоваться для процесса валидации при обучении. Также с помощью параметра `validation` можно передать саму выборку для валидации, предварительно отделив ее от тренировочной выборки. Следующий пример демонстрирует описанные возможности:

```
class MyEarlyStopping(callbacks.Callback):

    def on_epoch_end(self, epoch, logs=None):
        current = logs.get('val_loss')
        if abs(current) < e:
            self.model.stop_training = True

train, test = train_test_split(data, test_size=0.2, train_size=0.8)

rv = autoencoder.fit(
    train, train,
    epochs=500,
    validation_split=0.2,
    callbacks=[MyEarlyStopping(), callbacks.EarlyStopping(patience=6)]
)
```

Параметр `keras.callbacks` предназначен для определения различных проверок в процессе тестирования. В данном примере указан `keras.callbacks.EarlyStopping`, который останавливает обучение, если ошибка валидации не улучшается 6 эпох подряд. Также в примере реализован другой колбек, который останавливает обучение, если ошибка по валидационной выборке достигает значения, меньшего, чем допустимое.

Функция `predict` используется для предсказания выходных значений с помощью обученной модели.

5. *Вывод*

Библиотека Keras содержит достаточное количество средств для создания и обучения моделей нейросетей. Ее средства достаточно понятны и просты в использовании. При необходимости библиотека легко расширяемая и кастомизируемая.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Keras Documentation [Электронный ресурс]. – Электронные данные. – - Режим доступа: https://keras.io/.

[2] How to build a Neural Network with Keras [Электронный ресурс]. – Электронные данные. – Режим доступа: https://towardsdatascience.com/how-to-build-a-neural-network-with-keras-e8faa33d0ae4.