УДК 519.68:007.5

Лабораторная работа 2. Сети LSTM

Цели работы: изучение возможностей LSTM сетей для решения задач классификации и регрессии. Работа с готовыми реализациями LSTM в Python.

Ключевые слова: сети долгой кратковременной памяти, комбинирование различных нейросетевых моделей.

§ 1. Применение LSTM в Python для задач классификации

Рекурректные сети предназначены для работы с последовательностями целочисленных векторов. Для адаптации сети LSTM под задачи классификации предложений нам нужно ввести словарь с полным перечнем используемых слов и представить предложение в виде последовательности кодов слов из словаря. Затем нам потребуется превратить целочисленный код в вектор признаков, который уже можно будет подать на входы первому слою LSTM. Пример с использованием библиотеки Keras приведен ниже.

```
import keras
from keras import backend as K
from keras.optimizers import Adam
from keras.metrics import categorical_crossentropy
from keras.preprocessing import image
from keras.models import Model,Sequential
from keras.layers import Dense,LSTM,Embedding

numFeatures = 12 # число признаков (число входов LSTM)
numHidden   = 20 # число скрытых блоков LSTM
numClasses  = 10 # число итоговых классов
vocabSize   = 1000 # размер словаря (количество слов)
maxLen      = 200 # максимальная длина предложения (в словах)

model = Sequential() # строим модель на основе стека слоев
model.add(Embedding(vocabSize, numFeatures, input_length=maxLen))
# перекодируем целочисленный код слова в вектор признаков
model.add(LSTM(numHidden, return_sequences=False))
# возвращает одно значение, а не последовательность
model.add(Dense(numClasses, activation='softmax'))
```

Для использования данной модели потребуется предварительно перекодировать все тексты в массивы из кодов слов (длина в словах = maxLen). Для коротких текстов придется расширить их, дополнив в конце нулевыми кодами. Затем ввести метки классов для каждого текста и собрать данные вместе с метками в обучающую и валидатационную выборки.

Замечание: *для лучшей работы алгоритма желательно упорядочить все тексты по возрастанию их длины в словах* (*до расширения нулями*).

```
maxEpochs = 40;  # число эпох для обучения
miniBatchSize = 25; # мини блоки обучающей выборки

model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(Train_data,
          validation_data=Valid_data,
          batch_size=miniBatchSize,
          verbose =2,
          shuffle=False, # не перемешивать
          # упорядоченные данные
          epochs=maxEpochs)
```

§ 2. Применение LSTM в Python для задач регрессии

Для решения задачи регрессии на основе LSTM более предпочтительным будет режим непрерывной выдачи последовательностей (return_sequences=True). Итоговое значение мы будем получать с помощью одного нейрона с линейной функцией активации, а в качестве функции потерь воспользуемся среднеквадратической ошибкой.

```
numFeatures  = 12  # число признаков (число входов сети).
numHiddenUnits  = 40 # число скрытых блоков LSTM
numResponses  = 1 # число выходов для слоя регрессии
maxLen  = 100 # максимальная длина последовательностей

model = Sequential() # строим модель на основе стека слоев
model.add(LSTM( numHiddenUnits,
return_sequences=True,
input_shape=(numFeatures,maxLen)))
model.add(Dense(numResponses,activation='linear'))

maxEpochs = 60; % число эпох для обучения
miniBatchSize = 25;% мини блоки обучающей выборки

model.compile(loss='mean_squared_error', optimizer='adam')
history = model.fit(train_X, train_y,
epochs=maxEpochs,
batch_size=miniBatchSize,
validation_data=(test_X, test_y),
verbose=2,
shuffle=False)
```

Для использования этой модели нам нужно будет подготовить последовательности для обучения train_X вместе с целевыми значениями train_Y (эталонные значения на выходе), а также аналогичную пару для тестовых данных.

§ 3. Практические задания

Задание № 1: Классификация частей речи.

1. Реализовать LSTM модель для классификации 5 типов коротких текстов на английском: научный, литературный, стихи, философский, бессвязный бред.
2. Подготовить обучающую выборку из эталонных текстов, а также расширенную выборку для валидации результатов.
3. Упорядочить обучающую выборку по длине текстов и найти оптимальное разбиение на мини блоки (приводящее к минимальному расширению нулями в пределах мини выборки).
4. Обучить сеть и проверить качество предсказаний. Сравнить с обучением без упорядочивания по длине.
5. Как можно улучшить результаты? Попробуйте использовать метод ранней остановки, dropout и регуляризацию.

Задание № 2: Предсказание времени отказа для ракетных двигателей.

1. Скачать данные с сайта НАСА https://ti.arc.nasa.gov/c/6/
2. Использовать данные из колонок 3 – 26 в качестве данных для входов.
3. Данные из второй колонки (время работы блока) использовать в качестве эталонных данных для регрессии.
4. Удалить из выборки константные данные (не менявшиеся на протяжении измерений).
5. Упорядочить обучающую выборку по длине последовательностей и найти оптимальное разбиение на мини блоки (приводящее к минимальному расширению нулями в пределах мини выборки).
6. Обучить сеть и проверить качество предсказаний. Сравнить с обучением без упорядочивания по длине.
7. Как можно улучшить результаты? Попробуйте использовать метод ранней остановки, dropout и регуляризацию.