Recurrent neural networks in Keras

Max Pumperla - Skymind

skymind

Available RNNs in keras

- `SimpleRNN` - basic RNN

skymind

Available RNNs in keras

- `SimpleRNN` - basic RNN
- `GRU` - Gated Recursive Unit (2014)

skymind

Available RNNs in keras

- `SimpleRNN` - basic RNN
- `GRU` - Gated Recursive Unit (2014)
- `LSTM` - Long short-term memory (1997)

skymind

Available RNNs in keras

- `SimpleRNN` - basic RNN
- `GRU` - Gated Recursive Unit (2014)
- `LSTM` - Long short-term memory (1997)
- Will focus exclusively on LSTMs here

skymind

LSTM layers

```
from keras.layers.recurrent import LSTM

LSTM(units,
     activation='tanh',
     recurrent_activation='hard_sigmoid',
     recurrent_initializer='orthogonal',
     recurrent_regularizer=None,
     dropout=0.0, recurrent_dropout=0.0,
     return_sequences=False)
```

skymind

Embedding layers

- Embedding layers for first layer only

skymind

Embedding layers

- Embedding layers for first layer only
- Transform integers into vectors of same length

skymind

Embedding layers

- Embedding layers for first layer only
- Transform integers into vectors of same length
- Example: [3, 12] embedded as [[0.1, 0.5], [1.3, 4.2]]

skymind

Embedding layers

- Embedding layers for first layer only
- Transform integers into vectors of same length
- Example: [3, 12] embedded as [[0.1, 0.5], [1.3, 4.2]]
- Embed a vocabulary into a vector space, apply to sentences

skymind

Embedding layers

- Embedding layers for first layer only
- Transform integers into vectors of same length
- Example: [3, 12] embedded as [[0.1, 0.5], [1.3, 4.2]]
- Embed a vocabulary into a vector space, apply to sentences
- 2D input mapped to 3D output, connects to LSTMs.

skymind

Embedding layers

```
from keras.layers.embeddings import Embedding

Embedding(input_dim,              # Vocabulary size
          output_dim,             # Output vector length
          embeddings_initializer='uniform',
          mask_zero=False)        # Mask zero values
```

skymind

Sentiment classification for movie reviews

- 25.000 movie reviews from IMDB, labelled good or bad

skymind

Sentiment classification for movie reviews

- 25.000 movie reviews from IMDB, labelled good or bad
- Data available from `keras.datasets` module

skymind

Sentiment classification for movie reviews

- 25.000 movie reviews from IMDB, labelled good or bad
- Data available from `keras.datasets` module
- Data is preprocessed as sequences of integers

skymind

Sentiment classification for movie reviews

- 25.000 movie reviews from IMDB, labelled good or bad
- Data available from `keras.datasets` module
- Data is preprocessed as sequences of integers
- Task: classify sentiment from review content

skymind

Sentiment classification for movie reviews

- 25.000 movie reviews from IMDB, labelled good or bad
- Data available from `keras.datasets` module
- Data is preprocessed as sequences of integers
- Task: classify sentiment from review content
- Strategy: embed sentences, then learn structure with LSTM

skymind

Loading IMDB sentiment data

```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb

max_features = 20000
maxlen = 80

(x_train, y_train), (x_test, y_test) = \
    imdb.load_data(num_words=max_features)
```

skymind

Padding sequences and defining LSTM model

```
x_train = sequence.pad_sequences(x_train, maxlen=maxlen)
x_test = sequence.pad_sequences(x_test, maxlen=maxlen)

model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

skymind

Run and evaluate model

```
model.compile(loss='binary_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=32, epochs=15,
          validation_data=(x_test, y_test))

model.evaluate(x_test, y_test, batch_size=32)
```