22.2) Sequence Processing with Convnets

Vitor Kamada

August 2019

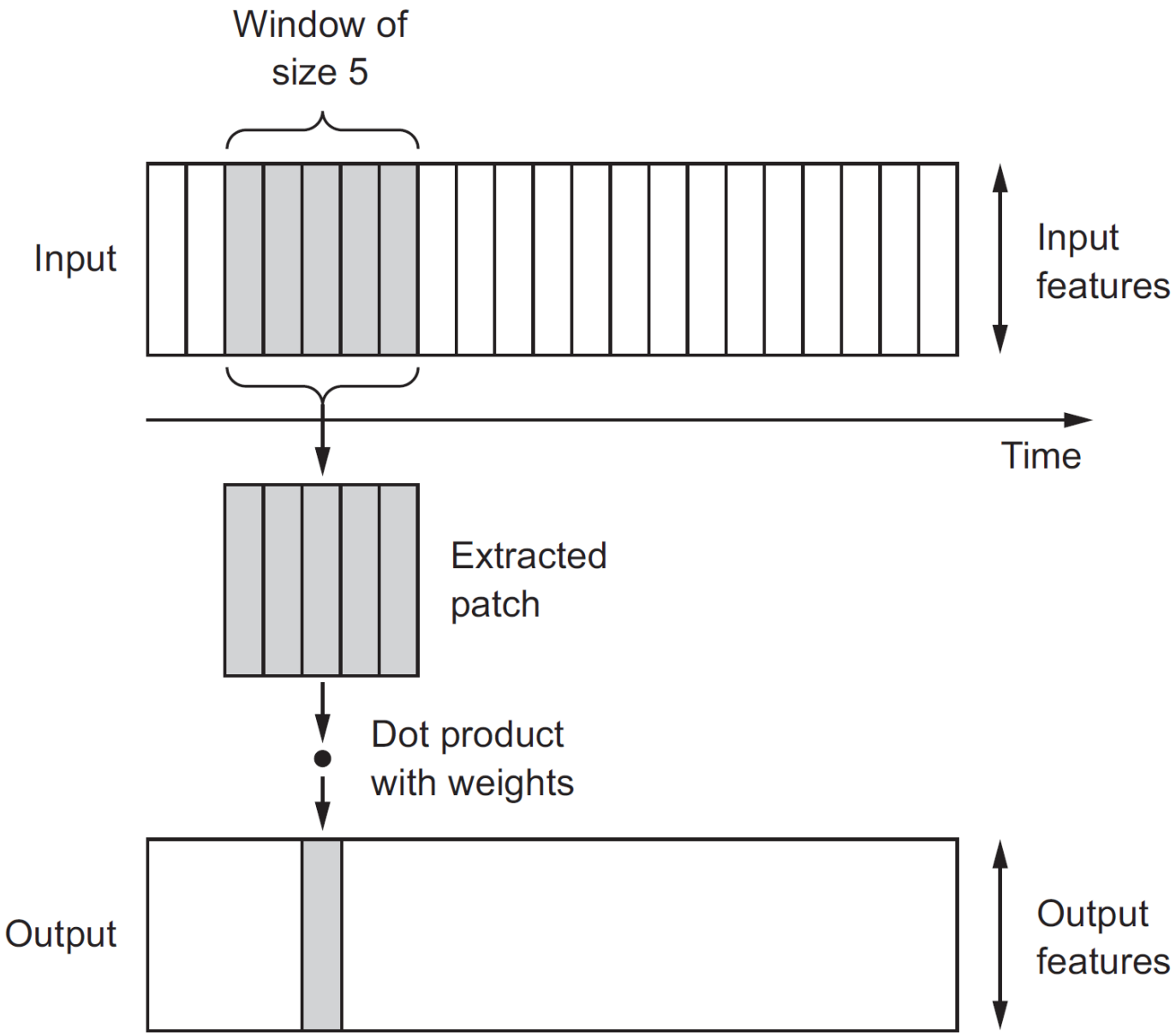
Reference

Chollet (2018): Ch 6.4

https://www.manning.com/books/deep-learning-with-python

https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/6.4-sequence-processing-with-convnets.ipynb

1D Convolution for Sequence Data



Chollet (2018: 225)

```
from keras.datasets import imdb
from keras.preprocessing import sequence
```

```
max_features = 10000  # number of words to consider as features
max_len = 500  # cut texts after this number of words (among top max_features

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), 'train sequences')
print(len(x_test), 'test sequences')

print('Pad sequences (samples x time)')
x_train = sequence.pad_sequences(x_train, maxlen=max_len)
x_test = sequence.pad_sequences(x_test, maxlen=max_len)
print('x_train shape:', x_train.shape)
print('x_test shape:', x_test.shape)
```

```
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 500)
x_test shape: (25000, 500)
```

1D Convnet

```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

model = Sequential()
model.add(layers.Embedding(max_features, 128,
                           input_length=max_len))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))
```

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 128)          1280000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 494, 32)           28704
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 98, 32)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 92, 32)            7200
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,315,937
```

```
model.compile(optimizer=RMSprop(lr=1e-4),
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/10
20000/20000 loss: 0.7713 - acc: 0.5287 - val_loss: 0.6818 - val_acc: 0.5970
Epoch 2/10
20000/20000 loss: 0.6631 - acc: 0.6775 - val_loss: 0.6582 - val_acc: 0.6646
Epoch 3/10
20000/20000 loss: 0.6142 - acc: 0.7580 - val_loss: 0.5987 - val_acc: 0.7118
Epoch 4/10
20000/20000 loss: 0.5156 - acc: 0.8124 - val_loss: 0.4936 - val_acc: 0.7736
Epoch 5/10
20000/20000 loss: 0.4029 - acc: 0.8469 - val_loss: 0.4123 - val_acc: 0.8358
Epoch 6/10
20000/20000 loss: 0.3455 - acc: 0.8653 - val_loss: 0.4040 - val_acc: 0.8382
Epoch 7/10
20000/20000 loss: 0.3078 - acc: 0.8634 - val_loss: 0.4059 - val_acc: 0.8240
Epoch 8/10
20000/20000 loss: 0.2812 - acc: 0.8535 - val_loss: 0.4147 - val_acc: 0.8098
Epoch 9/10
20000/20000 loss: 0.2554 - acc: 0.8334 - val_loss: 0.4296 - val_acc: 0.7878
Epoch 10/10
20000/20000 loss: 0.2356 - acc: 0.8052 - val_loss: 0.4296 - val_acc: 0.7600
```

Validation Accuracy 83%