An Introduction to Deep Learning With Python

[4.1] Classifying movie reviews a binary classification example

Prof. Yuzo Iano

pgs: 105 - 110

Loading the IMDB dataset

In [1]:
```
from keras.datasets import imdb
(train_data, train_labels),(test_data, test_labels) = imdb.load_data(num_words = 10000)
```

```
Using TensorFlow backend.
```

In [2]:
```
word_index = imdb.get_word_index()
reverse_word_index = dict(
    [(value, key) for (key, value) in word_index.items()])
decoded_review = ' '.join(
    [reverse_word_index.get(i - 3, '?') for i in train_data[0]])
```

Preparing the data

Encoding the Integer sequences Into a binary matrix

In [3]:
```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [4]:
```
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

Setting aside a validation set

In [5]:
```
x_val = x_train[: 10000]
partial_x_train = x_train[10000:]

y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

Building your model

Original model

In [6]:
```python
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(10000,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
WARNING:tensorflow:From C:\Users\pablo\AppData\Roaming\Python\Python36\site-packages\tensorflow\py
thon\framework\op_def_library.py:263: colocate_with (from tensorflow.python.framework.ops) is depr
ecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_1 (Dense)              (None, 16)                160016    
_________________________________________________________________
dense_2 (Dense)              (None, 16)                272       
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17        
=================================================================
Total params: 160,305
Trainable params: 160,305
Non-trainable params: 0
_________________________________________________________________
```

Training your model

In [8]:
```python
from keras import optimizers

model.compile(optimizer = 'rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['acc'])

history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs = 20,
                    batch_size = 512,
                    validation_data = (x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 5s 364us/step - loss: 0.1637 - acc: 0.9431 - val_lo
ss: 0.3212 - val_acc: 0.8728
Epoch 2/20
15000/15000 [==============================] - 5s 365us/step - loss: 0.1277 - acc: 0.9588 - val_lo
ss: 0.2897 - val_acc: 0.8887
Epoch 3/20
15000/15000 [==============================] - 5s 340us/step - loss: 0.1011 - acc: 0.9699 - val_lo
ss: 0.3131 - val_acc: 0.8835
Epoch 4/20
15000/15000 [==============================] - 5s 326us/step - loss: 0.0850 - acc: 0.9743 - val_lo
ss: 0.3744 - val_acc: 0.8720
Epoch 5/20
15000/15000 [==============================] - 5s 329us/step - loss: 0.0707 - acc: 0.9791 - val_lo
ss: 0.3604 - val_acc: 0.8767
Epoch 6/20
15000/15000 [==============================] - 5s 321us/step - loss: 0.0585 - acc: 0.9844 - val_lo
ss: 0.3745 - val_acc: 0.8818
Epoch 7/20
15000/15000 [==============================] - 5s 323us/step - loss: 0.0503 - acc: 0.9869 - val_lo
ss: 0.4004 - val_acc: 0.8790
Epoch 8/20
15000/15000 [==============================] - 5s 324us/step - loss: 0.0369 - acc: 0.9923 - val_lo
ss: 0.4389 - val_acc: 0.8748
Epoch 9/20
15000/15000 [==============================] - 5s 328us/step - loss: 0.0330 - acc: 0.9927 - val_lo
ss: 0.4694 - val_acc: 0.8741
Epoch 10/20
15000/15000 [==============================] - 5s 329us/step - loss: 0.0243 - acc: 0.9959 - val_lo
ss: 0.5208 - val_acc: 0.8698
Epoch 11/20
15000/15000 [==============================] - 5s 322us/step - loss: 0.0204 - acc: 0.9969 - val_lo
ss: 0.5304 - val_acc: 0.8711
Epoch 12/20
15000/15000 [==============================] - 5s 321us/step - loss: 0.0146 - acc: 0.9981 - val_lo
ss: 0.5892 - val_acc: 0.8680
Epoch 13/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0146 - acc: 0.9974 - val_lo
ss: 0.6086 - val_acc: 0.8697
Epoch 14/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0069 - acc: 0.9997 - val_lo
ss: 0.6485 - val_acc: 0.8646
Epoch 15/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0113 - acc: 0.9975 - val_lo
ss: 0.6717 - val_acc: 0.8669
Epoch 16/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0042 - acc: 0.9998 - val_lo
ss: 0.7049 - val_acc: 0.8636
Epoch 17/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0074 - acc: 0.9987 - val_lo
ss: 0.7562 - val_acc: 0.8645
Epoch 18/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0022 - acc: 0.9999 - val_lo
ss: 0.7770 - val_acc: 0.8633
Epoch 19/20
15000/15000 [==============================] - 5s 331us/step - loss: 0.0076 - acc: 0.9980 - val_lo
ss: 0.8096 - val_acc: 0.8625
Epoch 20/20
15000/15000 [==============================] - 5s 334us/step - loss: 0.0014 - acc: 0.9999 - val_lo
ss: 0.8258 - val_acc: 0.8621
```

In [9]:
```
original_loss = history.history['val_loss']
original_loss_training = history.history['loss']
```

Version of the model with lower capacity

In [10]:
```
low = Sequential()
low.add(Dense(4, activation='relu', input_shape=(10000,)))
low.add(Dense(4, activation='relu'))
low.add(Dense(1, activation='sigmoid'))
low.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 4)                 40004     
_________________________________________________________________
dense_5 (Dense)              (None, 4)                 20        
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 5         
=================================================================
Total params: 40,029
Trainable params: 40,029
Non-trainable params: 0
_________________________________________________________________
```

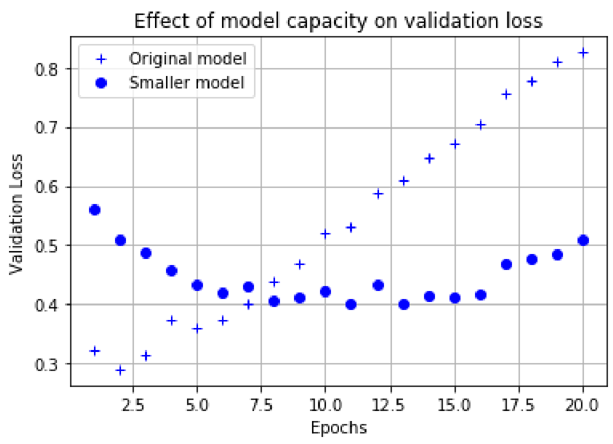
In [11]:
```
low.compile(optimizer = 'rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['acc'])

historylow = low.fit(partial_x_train,
                     partial_y_train,
                     epochs = 20,
                     batch_size = 512,
                     validation_data = (x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 5s 346us/step - loss: 0.6187 - acc: 0.6503 - val_lo
ss: 0.5607 - val_acc: 0.7740
Epoch 2/20
15000/15000 [==============================] - 5s 302us/step - loss: 0.5202 - acc: 0.8149 - val_lo
ss: 0.5100 - val_acc: 0.8055
Epoch 3/20
15000/15000 [==============================] - 5s 301us/step - loss: 0.4654 - acc: 0.8729 - val_lo
ss: 0.4869 - val_acc: 0.8097
Epoch 4/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.4243 - acc: 0.9039 - val_lo
ss: 0.4568 - val_acc: 0.8518
Epoch 5/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.3899 - acc: 0.9233 - val_lo
ss: 0.4342 - val_acc: 0.8748
Epoch 6/20
15000/15000 [==============================] - 5s 305us/step - loss: 0.3599 - acc: 0.9373 - val_lo
ss: 0.4189 - val_acc: 0.8790
Epoch 7/20
15000/15000 [==============================] - 5s 304us/step - loss: 0.3331 - acc: 0.9484 - val_lo
ss: 0.4307 - val_acc: 0.8599
Epoch 8/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.3087 - acc: 0.9561 - val_lo
ss: 0.4053 - val_acc: 0.8771
Epoch 9/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.2855 - acc: 0.9639 - val_lo
ss: 0.4127 - val_acc: 0.8706
Epoch 10/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.2646 - acc: 0.9695 - val_lo
ss: 0.4214 - val_acc: 0.8671
Epoch 11/20
15000/15000 [==============================] - 5s 304us/step - loss: 0.2449 - acc: 0.9739 - val_lo
ss: 0.3998 - val_acc: 0.8752
Epoch 12/20
15000/15000 [==============================] - 5s 303us/step - loss: 0.2262 - acc: 0.9781 - val_lo
ss: 0.4339 - val_acc: 0.8636
Epoch 13/20
15000/15000 [==============================] - 5s 304us/step - loss: 0.2087 - acc: 0.9818 - val_lo
ss: 0.3999 - val_acc: 0.8733
Epoch 14/20
15000/15000 [==============================] - 5s 302us/step - loss: 0.1924 - acc: 0.9843 - val_lo
ss: 0.4136 - val_acc: 0.8707
Epoch 15/20
15000/15000 [==============================] - 5s 302us/step - loss: 0.1776 - acc: 0.9867 - val_lo
ss: 0.4119 - val_acc: 0.8719
Epoch 16/20
15000/15000 [==============================] - 5s 307us/step - loss: 0.1632 - acc: 0.9875 - val_lo
ss: 0.4165 - val_acc: 0.8704
Epoch 17/20
15000/15000 [==============================] - 5s 304us/step - loss: 0.1494 - acc: 0.9892 - val_lo
ss: 0.4682 - val_acc: 0.8627
Epoch 18/20
15000/15000 [==============================] - 5s 300us/step - loss: 0.1376 - acc: 0.9901 - val_lo
ss: 0.4766 - val_acc: 0.8634
Epoch 19/20
15000/15000 [==============================] - 5s 315us/step - loss: 0.1262 - acc: 0.9911 - val_lo
ss: 0.4839 - val_acc: 0.8622
Epoch 20/20
15000/15000 [==============================] - 5s 313us/step - loss: 0.1157 - acc: 0.9918 - val_lo
ss: 0.5082 - val_acc: 0.8610
```

In [12]:
```
low_loss = historylow.history['val_loss']
```

Plotting the validation loss

In [14]:
```python
import matplotlib.pyplot as plt

epochs = range(1, len(original_loss) + 1)

plt.plot(epochs, original_loss, 'b+', label='Original model')
plt.plot(epochs, low_loss, 'bo', label='Smaller model')
plt.title('Effect of model capacity on validation loss')
plt.xlabel('Epochs')
plt.ylabel('Validation Loss')
plt.legend()
plt.grid()

plt.show()
```



Version of the model with higher capacity

In [15]:
```python
high = Sequential()
high.add(Dense(512, activation='relu', input_shape=(10000,)))
high.add(Dense(512, activation='relu'))
high.add(Dense(1, activation='sigmoid'))
high.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_7 (Dense)              (None, 512)               5120512   
_________________________________________________________________
dense_8 (Dense)              (None, 512)               262656    
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 513       
=================================================================
Total params: 5,383,681
Trainable params: 5,383,681
Non-trainable params: 0
_________________________________________________________________
```

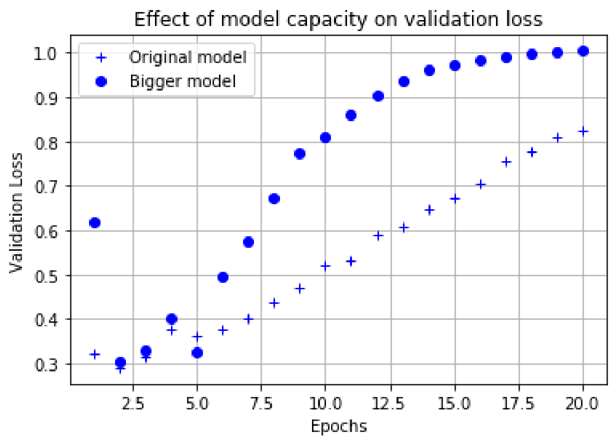
In [16]:
```
high.compile(optimizer = 'rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['acc'])

historyhigh = high.fit(partial_x_train,
                    partial_y_train,
                    epochs = 20,
                    batch_size = 512,
                    validation_data = (x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 10s 654us/step - loss: 0.5303 - acc: 0.7758 - val_l
oss: 0.6182 - val_acc: 0.7285
Epoch 2/20
15000/15000 [==============================] - 9s 588us/step - loss: 0.2540 - acc: 0.8991 - val_lo
ss: 0.3049 - val_acc: 0.8780
Epoch 3/20
15000/15000 [==============================] - 9s 590us/step - loss: 0.1562 - acc: 0.9409 - val_lo
ss: 0.3292 - val_acc: 0.8609
Epoch 4/20
15000/15000 [==============================] - 9s 583us/step - loss: 0.0834 - acc: 0.9723 - val_lo
ss: 0.4004 - val_acc: 0.8810
Epoch 5/20
15000/15000 [==============================] - 9s 570us/step - loss: 0.1209 - acc: 0.9769 - val_lo
ss: 0.3261 - val_acc: 0.8825
Epoch 6/20
15000/15000 [==============================] - 9s 570us/step - loss: 0.0059 - acc: 0.9998 - val_lo
ss: 0.4956 - val_acc: 0.8839
Epoch 7/20
15000/15000 [==============================] - 9s 573us/step - loss: 7.1435e-04 - acc: 1.0000 - va
l_loss: 0.5763 - val_acc: 0.8865
Epoch 8/20
15000/15000 [==============================] - 9s 572us/step - loss: 1.3496e-04 - acc: 1.0000 - va
l_loss: 0.6726 - val_acc: 0.8851
Epoch 9/20
15000/15000 [==============================] - 9s 602us/step - loss: 2.1774e-05 - acc: 1.0000 - va
l_loss: 0.7739 - val_acc: 0.8813
Epoch 10/20
15000/15000 [==============================] - 9s 574us/step - loss: 5.1310e-06 - acc: 1.0000 - va
l_loss: 0.8090 - val_acc: 0.8848
Epoch 11/20
15000/15000 [==============================] - 9s 577us/step - loss: 1.2307e-06 - acc: 1.0000 - va
l_loss: 0.8606 - val_acc: 0.8857
Epoch 12/20
15000/15000 [==============================] - 9s 590us/step - loss: 4.2933e-07 - acc: 1.0000 - va
l_loss: 0.9018 - val_acc: 0.8862
Epoch 13/20
15000/15000 [==============================] - 9s 594us/step - loss: 2.1638e-07 - acc: 1.0000 - va
l_loss: 0.9347 - val_acc: 0.8858
Epoch 14/20
15000/15000 [==============================] - 9s 598us/step - loss: 1.4739e-07 - acc: 1.0000 - va
l_loss: 0.9607 - val_acc: 0.8853
Epoch 15/20
15000/15000 [==============================] - 9s 600us/step - loss: 1.2666e-07 - acc: 1.0000 - va
l_loss: 0.9719 - val_acc: 0.8852
Epoch 16/20
15000/15000 [==============================] - 9s 593us/step - loss: 1.1897e-07 - acc: 1.0000 - va
l_loss: 0.9819 - val_acc: 0.8856
Epoch 17/20
15000/15000 [==============================] - 9s 602us/step - loss: 1.1569e-07 - acc: 1.0000 - va
l_loss: 0.9890 - val_acc: 0.8853
Epoch 18/20
15000/15000 [==============================] - 9s 574us/step - loss: 1.1386e-07 - acc: 1.0000 - va
l_loss: 0.9959 - val_acc: 0.8855
Epoch 19/20
15000/15000 [==============================] - 9s 586us/step - loss: 1.1283e-07 - acc: 1.0000 - va
l_loss: 0.9996 - val_acc: 0.8854
Epoch 20/20
15000/15000 [==============================] - 9s 578us/step - loss: 1.1193e-07 - acc: 1.0000 - va
l_loss: 1.0038 - val_acc: 0.8857
```

In [17]:
```
high_loss = historyhigh.history['val_loss']
high_loss_training = historyhigh.history['loss']
```

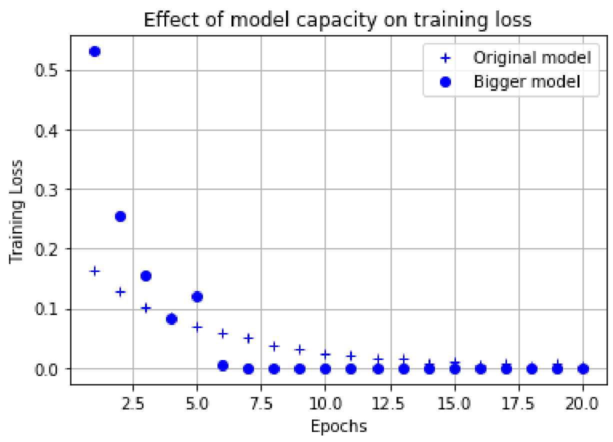
Plotting the validation loss

In [18]:
```
plt.plot(epochs, original_loss, 'b+', label='Original model')
plt.plot(epochs, high_loss, 'bo', label='Bigger model')
plt.title('Effect of model capacity on validation loss')
plt.xlabel('Epochs')
plt.ylabel('Validation Loss')
plt.legend()
plt.grid()

plt.show()
```



Plotting the training loss

In [19]:
```
plt.plot(epochs, original_loss_training, 'b+', label='Original model')
plt.plot(epochs, high_loss_training, 'bo', label='Bigger model')
plt.title('Effect of model capacity on training loss')
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.legend()
plt.grid()

plt.show()
```



Adding L2 weight regularization to the model

In [20]:
```python
from keras import regularizers

modell2 = Sequential()
modell2.add(Dense(16, kernel_regularizer=regularizers.l2(0.001), activation='relu', input_shape=(1
0000,)))
modell2.add(Dense(16, kernel_regularizer=regularizers.l2(0.001), activation='relu'))
modell2.add(Dense(1, activation='sigmoid'))
modell2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_10 (Dense)             (None, 16)                160016    
_________________________________________________________________
dense_11 (Dense)             (None, 16)                272       
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 17        
=================================================================
Total params: 160,305
Trainable params: 160,305
Non-trainable params: 0
_________________________________________________________________
```

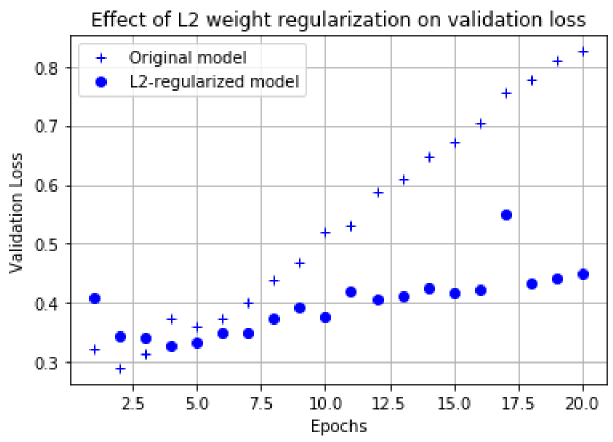
In [21]:
```
modell2.compile(optimizer = 'rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['acc'])

historyl2 = modell2.fit(partial_x_train,
                    partial_y_train,
                    epochs = 20,
                    batch_size = 512,
                    validation_data = (x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 6s 385us/step - loss: 0.5298 - acc: 0.7949 - val_lo
ss: 0.4098 - val_acc: 0.8655
Epoch 2/20
15000/15000 [==============================] - 5s 323us/step - loss: 0.3371 - acc: 0.9017 - val_lo
ss: 0.3439 - val_acc: 0.8892
Epoch 3/20
15000/15000 [==============================] - 5s 321us/step - loss: 0.2741 - acc: 0.9241 - val_lo
ss: 0.3400 - val_acc: 0.8853
Epoch 4/20
15000/15000 [==============================] - 5s 322us/step - loss: 0.2403 - acc: 0.9345 - val_lo
ss: 0.3273 - val_acc: 0.8893
Epoch 5/20
15000/15000 [==============================] - 5s 328us/step - loss: 0.2233 - acc: 0.9437 - val_lo
ss: 0.3339 - val_acc: 0.8877
Epoch 6/20
15000/15000 [==============================] - 5s 341us/step - loss: 0.2056 - acc: 0.9515 - val_lo
ss: 0.3485 - val_acc: 0.8849
Epoch 7/20
15000/15000 [==============================] - 5s 330us/step - loss: 0.1961 - acc: 0.9531 - val_lo
ss: 0.3487 - val_acc: 0.8840
Epoch 8/20
15000/15000 [==============================] - 5s 318us/step - loss: 0.1901 - acc: 0.9559 - val_lo
ss: 0.3745 - val_acc: 0.8796
Epoch 9/20
15000/15000 [==============================] - 5s 324us/step - loss: 0.1824 - acc: 0.9599 - val_lo
ss: 0.3915 - val_acc: 0.8770
Epoch 10/20
15000/15000 [==============================] - 5s 333us/step - loss: 0.1750 - acc: 0.9643 - val_lo
ss: 0.3766 - val_acc: 0.8801
Epoch 11/20
15000/15000 [==============================] - 5s 322us/step - loss: 0.1698 - acc: 0.9649 - val_lo
ss: 0.4184 - val_acc: 0.8671
Epoch 12/20
15000/15000 [==============================] - 5s 317us/step - loss: 0.1654 - acc: 0.9665 - val_lo
ss: 0.4054 - val_acc: 0.8747
Epoch 13/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.1621 - acc: 0.9679 - val_lo
ss: 0.4103 - val_acc: 0.8734
Epoch 14/20
15000/15000 [==============================] - 5s 316us/step - loss: 0.1608 - acc: 0.9687 - val_lo
ss: 0.4254 - val_acc: 0.8706
Epoch 15/20
15000/15000 [==============================] - 5s 322us/step - loss: 0.1536 - acc: 0.9713 - val_lo
ss: 0.4180 - val_acc: 0.8749
Epoch 16/20
15000/15000 [==============================] - 5s 325us/step - loss: 0.1587 - acc: 0.9677 - val_lo
ss: 0.4234 - val_acc: 0.8748
Epoch 17/20
15000/15000 [==============================] - 5s 345us/step - loss: 0.1528 - acc: 0.9700 - val_lo
ss: 0.5488 - val_acc: 0.8490
Epoch 18/20
15000/15000 [==============================] - 5s 339us/step - loss: 0.1505 - acc: 0.9719 - val_lo
ss: 0.4336 - val_acc: 0.8753
Epoch 19/20
15000/15000 [==============================] - 5s 326us/step - loss: 0.1457 - acc: 0.9747 - val_lo
ss: 0.4419 - val_acc: 0.8708
Epoch 20/20
15000/15000 [==============================] - 5s 326us/step - loss: 0.1397 - acc: 0.9774 - val_lo
ss: 0.4508 - val_acc: 0.8713
```

In [22]:
```
l2_loss = historyl2.history['val_loss']
```

Plotting the validation loss

In [23]:
```
plt.plot(epochs, original_loss, 'b+', label='Original model')
plt.plot(epochs, l2_loss, 'bo', label='L2-regularized model')
plt.title('Effect of L2 weight regularization on validation loss')
plt.xlabel('Epochs')
plt.ylabel('Validation Loss')
plt.legend()
plt.grid()

plt.show()
```



Adding dropout to the IMDB network

In [24]:
```
from keras.layers import Dropout

model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(10000,)))
model.add(Dropout(0.5))
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

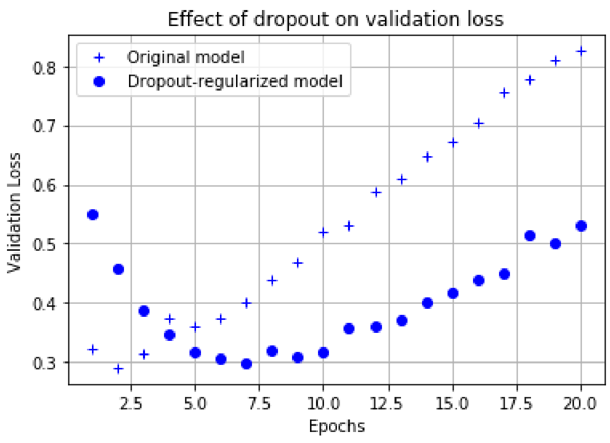
```
WARNING:tensorflow:From C:\Users\pablo\Python\envs\DAVID\lib\site-packages\keras\backend\tensorflo
w_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecate
d and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_13 (Dense)             (None, 16)                160016    
_________________________________________________________________
dropout_1 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_14 (Dense)             (None, 16)                272       
_________________________________________________________________
dropout_2 (Dropout)          (None, 16)                0         
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 17        
=================================================================
Total params: 160,305
Trainable params: 160,305
Non-trainable params: 0
_________________________________________________________________
```

In [25]:
```
model.compile(optimizer = 'rmsprop',
              loss = 'binary_crossentropy',
              metrics = ['acc'])

history_dropout = model.fit(partial_x_train,
                    partial_y_train,
                    epochs = 20,
                    batch_size = 512,
                    validation_data = (x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 5s 353us/step - loss: 0.6414 - acc: 0.6269 - val_lo
ss: 0.5487 - val_acc: 0.7669
Epoch 2/20
15000/15000 [==============================] - 5s 345us/step - loss: 0.5377 - acc: 0.7530 - val_lo
ss: 0.4570 - val_acc: 0.8666
Epoch 3/20
15000/15000 [==============================] - 5s 331us/step - loss: 0.4633 - acc: 0.8154 - val_lo
ss: 0.3865 - val_acc: 0.8800
Epoch 4/20
15000/15000 [==============================] - 5s 333us/step - loss: 0.4013 - acc: 0.8533 - val_lo
ss: 0.3465 - val_acc: 0.8798
Epoch 5/20
15000/15000 [==============================] - 6s 412us/step - loss: 0.3548 - acc: 0.8838 - val_lo
ss: 0.3159 - val_acc: 0.8841
Epoch 6/20
15000/15000 [==============================] - 6s 377us/step - loss: 0.3134 - acc: 0.8993 - val_lo
ss: 0.3066 - val_acc: 0.8764
Epoch 7/20
15000/15000 [==============================] - 5s 346us/step - loss: 0.2806 - acc: 0.9156 - val_lo
ss: 0.2982 - val_acc: 0.8825
Epoch 8/20
15000/15000 [==============================] - 5s 358us/step - loss: 0.2542 - acc: 0.9275 - val_lo
ss: 0.3181 - val_acc: 0.8861
Epoch 9/20
15000/15000 [==============================] - 5s 347us/step - loss: 0.2246 - acc: 0.9354 - val_lo
ss: 0.3079 - val_acc: 0.8848
Epoch 10/20
15000/15000 [==============================] - 5s 340us/step - loss: 0.2006 - acc: 0.9445 - val_lo
ss: 0.3174 - val_acc: 0.8854
Epoch 11/20
15000/15000 [==============================] - 5s 330us/step - loss: 0.1818 - acc: 0.9494 - val_lo
ss: 0.3564 - val_acc: 0.8857
Epoch 12/20
15000/15000 [==============================] - 5s 336us/step - loss: 0.1658 - acc: 0.9523 - val_lo
ss: 0.3598 - val_acc: 0.8845
Epoch 13/20
15000/15000 [==============================] - 5s 347us/step - loss: 0.1546 - acc: 0.9553 - val_lo
ss: 0.3705 - val_acc: 0.8837
Epoch 14/20
15000/15000 [==============================] - 5s 326us/step - loss: 0.1397 - acc: 0.9620 - val_lo
ss: 0.4020 - val_acc: 0.8849
Epoch 15/20
15000/15000 [==============================] - 5s 328us/step - loss: 0.1283 - acc: 0.9645 - val_lo
ss: 0.4183 - val_acc: 0.8855
Epoch 16/20
15000/15000 [==============================] - 5s 331us/step - loss: 0.1188 - acc: 0.9669 - val_lo
ss: 0.4393 - val_acc: 0.8863
Epoch 17/20
15000/15000 [==============================] - 5s 356us/step - loss: 0.1127 - acc: 0.9683 - val_lo
ss: 0.4482 - val_acc: 0.8845
Epoch 18/20
15000/15000 [==============================] - 5s 334us/step - loss: 0.1103 - acc: 0.9701 - val_lo
ss: 0.5141 - val_acc: 0.8836
Epoch 19/20
15000/15000 [==============================] - 5s 327us/step - loss: 0.0986 - acc: 0.9719 - val_lo
ss: 0.5001 - val_acc: 0.8843
Epoch 20/20
15000/15000 [==============================] - 5s 325us/step - loss: 0.0998 - acc: 0.9733 - val_lo
ss: 0.5302 - val_acc: 0.8838
```

In [26]:
```
dropout_loss = history_dropout.history['val_loss']
```

In [27]:
```python
plt.plot(epochs, original_loss, 'b+', label='Original model')
plt.plot(epochs, dropout_loss, 'bo', label='Dropout-regularized model')
plt.title('Effect of dropout on validation loss')
plt.xlabel('Epochs')
plt.ylabel('Validation Loss')
plt.legend()
plt.grid()

plt.show()
```



Pablo Minango

- pablodavid218@gmail.com