In [1]:

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

max_features = 10000
max_len = 500

print('데이터 로드...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
print(len(x_train), '훈련 시퀀스')
print(len(x_test), '테스트 시퀀스')

print('시퀀스 패딩 (sample 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 크기:', x_train.shape)
print('x_test 크기:', x_test.shape)
```

```
Using TensorFlow backend.

데이터 로드...
25000 훈련 시퀀스
25000 테스트 시퀀스
시퀀스 패딩 (sample x time)
x_train 크기: (25000, 500)
x_test 크기: (25000, 500)
```

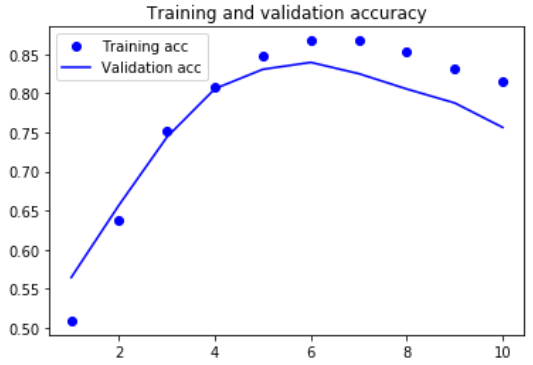
In [2]:

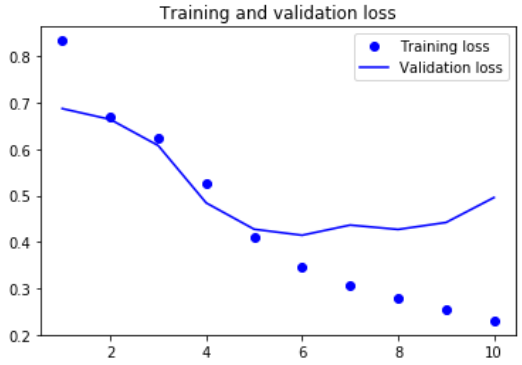
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

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))

model.summary()

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)
```

```
WARNING:tensorflow:From C:₩ProgramData₩Anaconda3₩lib₩site-packages₩tensorflow₩pyth
on₩framework₩op_def_library.py:263: colocate_with (from tensorflow.python.framewor
k.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
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
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
WARNING:tensorflow:From C:₩ProgramData₩Anaconda3₩lib₩site-packages₩tensorflow₩pyth
on₩ops₩math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is depreca
ted and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
WARNING:tensorflow:From C:₩ProgramData₩Anaconda3₩lib₩site-packages₩tensorflow₩pyth
on₩ops₩math_grad.py:102: div (from tensorflow.python.ops.math_ops) is deprecated a
nd will be removed in a future version.
Instructions for updating:
Deprecated in favor of operator or tf.math.divide.
Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 7s 355us/step - loss: 0.8337 - acc:
0.5093 - val_loss: 0.6874 - val_acc: 0.5646
Epoch 2/10
20000/20000 [==============================] - 1s 71us/step - loss: 0.6700 - acc:
0.6384 - val_loss: 0.6641 - val_acc: 0.6574
Epoch 3/10
20000/20000 [==============================] - 1s 72us/step - loss: 0.6235 - acc:
0.7524 - val_loss: 0.6078 - val_acc: 0.7442
Epoch 4/10
20000/20000 [==============================] - 1s 73us/step - loss: 0.5257 - acc:
0.8079 - val_loss: 0.4843 - val_acc: 0.8058
Epoch 5/10
20000/20000 [==============================] - 1s 73us/step - loss: 0.4112 - acc:
0.8482 - val_loss: 0.4276 - val_acc: 0.8306
Epoch 6/10
20000/20000 [==============================] - 1s 73us/step - loss: 0.3462 - acc:
0.8672 - val_loss: 0.4146 - val_acc: 0.8394
Epoch 7/10
20000/20000 [==============================] - 1s 73us/step - loss: 0.3071 - acc:
0.8669 - val_loss: 0.4365 - val_acc: 0.8250
Epoch 8/10
20000/20000 [==============================] - 1s 72us/step - loss: 0.2801 - acc:
0.8531 - val_loss: 0.4272 - val_acc: 0.8054
Epoch 9/10
20000/20000 [==============================] - 1s 74us/step - loss: 0.2539 - acc:
0.8323 - val_loss: 0.4421 - val_acc: 0.7876
Epoch 10/10
20000/20000 [==============================] - 1s 74us/step - loss: 0.2305 - acc:
0.8157 - val_loss: 0.4959 - val_acc: 0.7564
```

In [4]:

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

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

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [9]:

```
import os

data_dir = './jena_climate'
fname = os.path.join(data_dir, 'jena_climate_2009_2016.csv')

f = open(fname)
data = f.read()
f.close()
lines = data.split('Wn')
header = lines[0].split(',')
lines = lines[1:]

print(header)
print(len(lines))
```

```
['"Date Time"', '"p (mbar)"', '"T (degC)"', '"Tpot (K)"', '"Tdew (degC)"', '"rh
(%)"', '"VPmax (mbar)"', '"VPact (mbar)"', '"VPdef (mbar)"', '"sh (g/kg)"', '"H2OC
(mmol/mol)"', '"rho (g/m**3)"', '"wv (m/s)"', '"max. wv (m/s)"', '"wd (deg)"']
420551
```

In [10]:

```
import numpy as np

float_data = np.zeros((len(lines), len(header) - 1))
for i, line in enumerate(lines):
    values = [float(x) for x in line.split(',')[1:]]
    float_data[i, :] = values
```

In [11]:

```
mean = float_data[:200000].mean(axis=0)
float_data -= mean
std = float_data[:200000].std(axis=0)
float_data /= std
```

In [12]:

```
def generator(data, lookback, delay, min_index, max_index, shuffle=False, batch_size=128, step=6
):
    if max_index is None:
        max_index = len(data) - delay - 1
    i = min_index + lookback
    while 1:
        if shuffle:
            rows = np.random.randint(min_index + lookback, max_index, size=batch_size)
        else:
            if i + batch_size >= max_index:
                i = min_index + lookback
            rows = np.arange(i, min(i + batch_size, max_index))
            i += len(rows)

        samples = np.zeros((len(rows), lookback // step, data.shape[-1]))
        targets = np.zeros((len(rows),))
        for j, row in enumerate(rows):
            indices = range(rows[j] - lookback, rows[j], step)
            samples[j] = data[indices]
            targets[j] = data[rows[j] + delay][1]
        yield samples, targets
```

In [14]:

```
lookback = 1440
step = 6
delay = 144
batch_size = 128
train_gen = generator(float_data, lookback=lookback, delay=delay, min_index=0, max_index=200000,
shuffle=True, step=step,
                      batch_size=batch_size)

val_gen = generator(float_data, lookback=lookback, delay=delay, min_index=200001, max_index=3000
00, shuffle=True, step=step,
                    batch_size=batch_size)

test_gen = generator(float_data, lookback=lookback, delay=delay, min_index=300001, max_index=Non
e, shuffle=True, step=step,
                     batch_size=batch_size)

val_steps = (300000 - 200001 - lookback) // batch_size
test_steps = (len(float_data) - 300001 - lookback) // batch_size
```

1D 컨브넷 훈련 평가

In [15]:

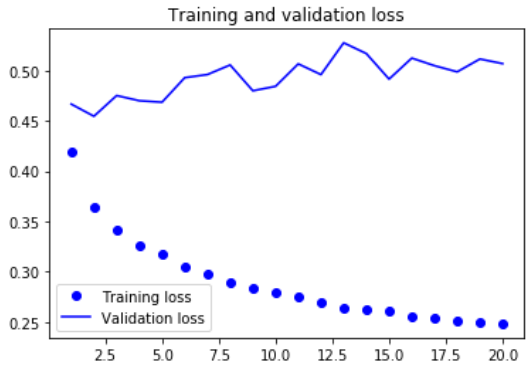
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop

model = Sequential()
model.add(layers.Conv1D(32, 5, activation='relu', input_shape=(None, float_data.shape[-1])))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 5, activation='relu'))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 5, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(), loss='mae')
history = model.fit_generator(train_gen, steps_per_epoch = 500, epochs=20, validation_data = val
_gen, validation_steps = val_steps)
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_3 (Conv1D)            (None, None, 32)          2272
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, None, 32)          0
_________________________________________________________________
conv1d_4 (Conv1D)            (None, None, 32)          5152
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, None, 32)          0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, None, 32)          5152
_________________________________________________________________
global_max_pooling1d_2 (Glob (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 12,609
Trainable params: 12,609
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
500/500 [==============================] - 14s 28ms/step - loss: 0.4197 - val_los
s: 0.4667
Epoch 2/20
500/500 [==============================] - 14s 28ms/step - loss: 0.3647 - val_los
s: 0.4546
Epoch 3/20
500/500 [==============================] - 14s 28ms/step - loss: 0.3414 - val_los
s: 0.4752
Epoch 4/20
500/500 [==============================] - 14s 28ms/step - loss: 0.3266 - val_los
s: 0.4701
Epoch 5/20
500/500 [==============================] - 14s 28ms/step - loss: 0.3173 - val_los
s: 0.4686
Epoch 6/20
500/500 [==============================] - 14s 27ms/step - loss: 0.3048 - val_los
s: 0.4930
Epoch 7/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2971 - val_los
s: 0.4961
Epoch 8/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2898 - val_los
s: 0.5057
Epoch 9/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2831 - val_los
s: 0.4800
Epoch 10/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2787 - val_los
s: 0.4844
Epoch 11/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2749 - val_los
s: 0.5068
Epoch 12/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2695 - val_los
s: 0.4961
Epoch 13/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2643 - val_los
s: 0.5277
Epoch 14/20
```

```
500/500 [==============================] - 14s 28ms/step - loss: 0.2619 - val_los
s: 0.5170
Epoch 15/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2603 - val_los
s: 0.4916
Epoch 16/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2558 - val_los
s: 0.5125
Epoch 17/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2540 - val_los
s: 0.5049
Epoch 18/20
500/500 [==============================] - 14s 27ms/step - loss: 0.2512 - val_los
s: 0.4988
Epoch 19/20
500/500 [==============================] - 14s 27ms/step - loss: 0.2493 - val_los
s: 0.5116
Epoch 20/20
500/500 [==============================] - 14s 28ms/step - loss: 0.2483 - val_los
s: 0.5070
```

In [16]:

```
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']

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

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



합성곱과 GRU 연결

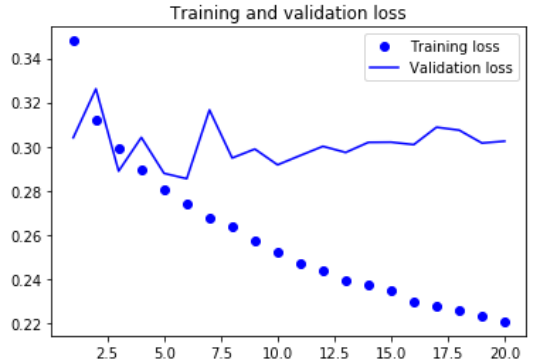
In [17]:

```
from keras.layers import GRU

model = Sequential()
model.add(layers.Conv1D(32, 5, activation='relu', input_shape=(None, float_data.shape[-1])))
model.add(layers.MaxPooling1D(3))
model.add(layers.Conv1D(32, 5, activation='relu'))
model.add(layers.GRU(32, dropout=0.1, recurrent_dropout=0.5))
model.add(layers.Dense(1))

model.summary()

model.compile(optimizer=RMSprop(), loss='mae')
history = model.fit_generator(train_gen, steps_per_epoch = 500, epochs=20, validation_data = val
_gen, validation_steps = val_steps)
```

```
WARNING:tensorflow:From C:₩ProgramData₩Anaconda3₩lib₩site-packages₩keras₩backend₩t
ensorflow_backend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) wit
h keep_prob is deprecated 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_p
rob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_6 (Conv1D)            (None, None, 32)          2272
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, None, 32)          0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, None, 32)          5152
_________________________________________________________________
gru_1 (GRU)                  (None, 32)                6240
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 13,697
Trainable params: 13,697
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
500/500 [==============================] - 67s 135ms/step - loss: 0.3479 - val_los
s: 0.3040
Epoch 2/20
500/500 [==============================] - 66s 133ms/step - loss: 0.3118 - val_los
s: 0.3261
Epoch 3/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2989 - val_los
s: 0.2889
Epoch 4/20
500/500 [==============================] - 67s 135ms/step - loss: 0.2893 - val_los
s: 0.3042
Epoch 5/20
500/500 [==============================] - 67s 134ms/step - loss: 0.2806 - val_los
s: 0.2879
Epoch 6/20
500/500 [==============================] - 67s 134ms/step - loss: 0.2741 - val_los
s: 0.2855
Epoch 7/20
500/500 [==============================] - 66s 133ms/step - loss: 0.2679 - val_los
s: 0.3166
Epoch 8/20
500/500 [==============================] - 66s 133ms/step - loss: 0.2637 - val_los
s: 0.2948
Epoch 9/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2573 - val_los
s: 0.2989
Epoch 10/20
500/500 [==============================] - 65s 130ms/step - loss: 0.2523 - val_los
s: 0.2917
Epoch 11/20
500/500 [==============================] - 65s 131ms/step - loss: 0.2473 - val_los
s: 0.2960
Epoch 12/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2438 - val_los
s: 0.3001
Epoch 13/20
500/500 [==============================] - 66s 133ms/step - loss: 0.2391 - val_los
s: 0.2974
Epoch 14/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2372 - val_los
s: 0.3019
Epoch 15/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2348 - val_los
s: 0.3020
Epoch 16/20
500/500 [==============================] - 65s 130ms/step - loss: 0.2298 - val_los
s: 0.3009
Epoch 17/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2279 - val_los
s: 0.3088
Epoch 18/20
500/500 [==============================] - 66s 131ms/step - loss: 0.2257 - val_los
s: 0.3074
Epoch 19/20
500/500 [==============================] - 66s 132ms/step - loss: 0.2236 - val_los
s: 0.3016
Epoch 20/20
500/500 [==============================] - 65s 131ms/step - loss: 0.2208 - val_los
s: 0.3025
```

In [18]:

```
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']

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

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



In [ ]:

```
```