SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 6: February 11–February 15, 2019

Exercises consist of both pen&paper and computer assignments. Pen&paper questions are solved at home before exercises, while computer assignments are solved during exercise hours. The computer assignments are marked by **`python`** and Pen&paper questions by **`pen&paper`**

1. **`pen&paper`** Count the number of parameters in a neural network

   Consider the traditional shallow neural network architecture of Figure 1. Suppose our inputs are 64×64 RGB bitmaps of two categories of traffic signs.

   Let the network structure be the following:

   - The input is $64 \times 64 \times 3 = 12288$-dimensional
   - On the 1st layer there are 100 nodes (marked in blue)
   - On the 2nd layer there are 100 nodes (marked in blue)
   - On the 3rd (output) layer there are 10 nodes (marked in blue; one for each class)

   Compute the number of parameters (coefficients) in the net.

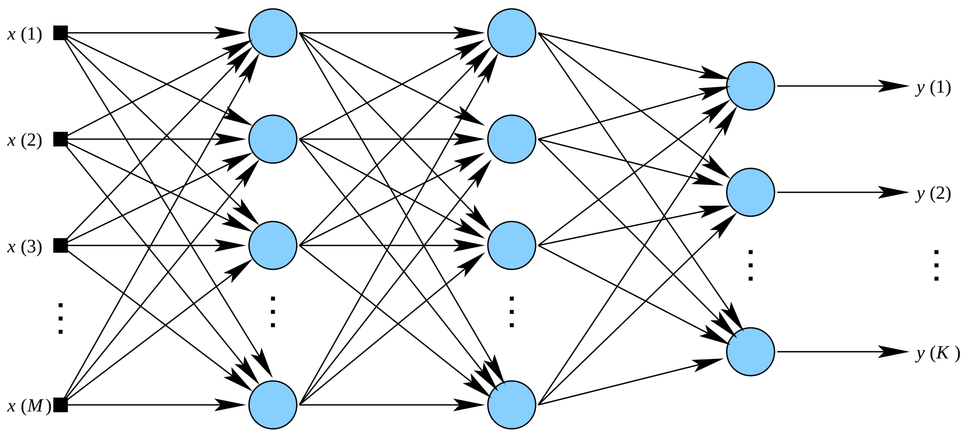

Figure 1: Vanilla neural network.

2. **pen&paper** Consider the following Keras code defining a convolutional neural network.

```
N = 10          # Number of feature maps
w, h = 5, 5     # Conv. window size

model.add(Conv2D(N, (w, h),
          input_shape=(64, 64, 3),
          activation = 'relu',
          padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(N, (w, h),
          activation = 'relu',
          padding = 'same'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(2, activation = 'sigmoid'))
```

   a) Draw a diagram of the network similar to the one at the bottom of slide 14 in `http://www.cs.tut.fi/courses/SGN-41007/slides/Lecture6.pdf`

   b) Compute the number of parameters of the network at each layer (and explain why).

3. **python** *Load Traffic sign data for deep neural network processing.*

   Download an extended version of the two class German Traffic Sign Recognition Benchmark (GTSRB) dataset from

   `http://www.cs.tut.fi/courses/SGN-41007/GTSRB_subset_2.zip`

   This time, images are in color and there are about 400 from both classes.

   After collecting the data, normalize all samples into range [0,1]; *i.e.*, subtract `numpy.min(X)` and divide the result by `numpy.max(X)`.

   Finally, split the data to training and testing (80% / 20%) using `sklearn.cross_validation.train_test_split`.

4. **python** *Define the network in Keras.*

Edit the network of Question 2 in your code such that `model.summary()` gives the following output:

```
model.summary()
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_49 (Conv2D)           (None, 64, 64, 32)        2432
_________________________________________________________________
max_pooling2d_47 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_50 (Conv2D)           (None, 16, 16, 32)        25632
_________________________________________________________________
max_pooling2d_48 (MaxPooling (None, 4, 4, 32)          0
_________________________________________________________________
flatten_15 (Flatten)         (None, 512)               0
_________________________________________________________________
dense_29 (Dense)             (None, 100)               51300
_________________________________________________________________
dense_30 (Dense)             (None, 2)                 202
=================================================================
Total params: 79,566
Trainable params: 79,566
Non-trainable params: 0
_________________________________________________________________
```

5. `python` *Compile and train the net.*

   Compile and train the network following the examples of the lecture slides and documentation at `http://keras.io/`.

   Use the following parameters:

   - **Loss:** categorical crossentropy (same thing as log loss; see previous exercises)
   - **Optimizer:** stochastic gradient descent
   - **Minibatch size:** 32
   - **Number of epochs:** 20

   Also add the parameter `metrics=['accuracy']` as an argument of `model.compile` and give the test data to training algorithm `model.fit(..., validation_data = [X_test, y_test])` Then, the optimizer will report the test error every epoch.