SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 6: February 15–February 17, 2017

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`

Note: Python exercises can not be executed in Pinni ML1084. Try to bring your own laptop with Anaconda installed. We will form groups around those machines that have Keras.

1. `pen&paper` *Derive an estimate for the variance of the random forest.*

   The efficiency of the random forest relies on the variability of the individual weak trees. However, this results in randomness of the results, as well: If you run training 10 times, you may have 10 different results.

   Increasing the number of trees helps to reduce the variation of outputs, as seen in Figure 1. More trees mean less variation. However, the question remains: how many trees is enough.

   The number of trees can be assessed theoretically via the following theorem in a regression context (output of the RF is the average of individual trees).

   Consider the prediction of trees $1, 2, \ldots, N$ as random variables $T_1, T_2, \ldots, T_N$, each with identical variance σ^2. If the predictions were independent, their average would have variance σ^2/N. However, if the RV's have pairwise correlation coefficient ρ, show that the variance of the average is given as

$$\rho\sigma^2 + \frac{1-\rho}{N}\sigma^2$$

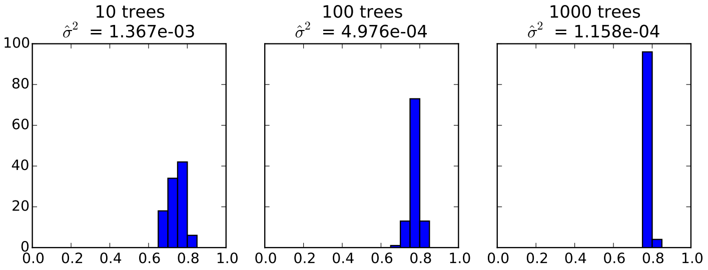

Figure 1: Variances of random forest with different number of trees.

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

Consider the traditional shallow neural network architecture of Figure 2. Suppose our inputs are 64×64 bitmaps of two categories of traffic signs.

a) Let the network structure be the following:

- The input is $3 \times 64 \times 64 = 12288$-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.

b) Consider the following code defining a convolutional architecture and compute the number of coefficients in this case.

```
N = 10        # Number of feature maps
w, h = 3, 3   # Conv. window size

model = Sequential()

model.add(Convolution2D(nb_filter = N,
                        nb_col = w,
                        nb_row = h,
                        activation = ’relu’,
                        border_mode = ’same’,
                        input_shape = (3,64,64)))
model.add(MaxPooling2D((2,2)))

model.add(Convolution2D(nb_filter = N,
                        nb_col = w,
                        nb_row = h,
                        border_mode = ’same’,
                        activation = ’relu’))
model.add(MaxPooling2D((2,2)))

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

c) An old rule of thumb states that the number of training samples should be at least 5 times the number of coefficients. Compute the desired sample size based on this rule for (a) and (b).

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

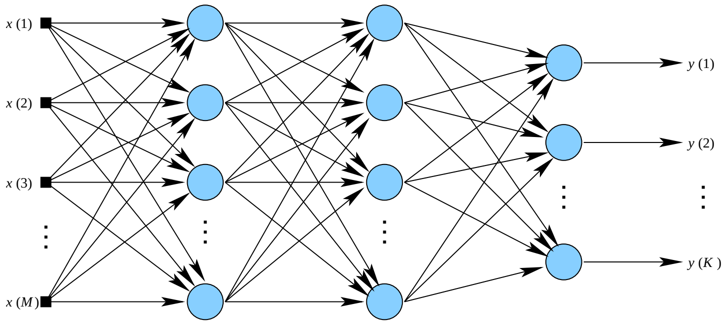

Figure 2: Vanilla neural network.

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

`http://www.cs.tut.fi/courses/SGN-41006/GTSRB_subset_2.zip`

This time, images are in color and there are about 400 from both classes. Edit your previous script (exercise set 4) such that the image array will have dimension (660, 3, 64, 64): 660 images of size 64x64 and 3 color channels. You will need `numpy.transpose` to reorder dimensions: (64, 64, 3) → (3, 64, 64).

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)`. Make sure that you are using float (not integer) division.

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

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

   Define the network of Question 2b in your code.

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

   Note: You need to set up Keras to use the Theano backend instead of Tensorflow (the default). To this aim, your config file `~/.keras/keras.json` has to look like this:

```
{
    "image_dim_ordering": "th",
    "epsilon": 1e-07,
    "floatx": "float32",
    "backend": "theano",
    "optimizer": "None"
}
```

   Compile and train the network as described in

   `https://keras.io/models/sequential/`

Use the following parameters:

- **Loss:** binary 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.