SGN-11000 Videomuotoinen viikkoharjoitus

22.1 – 26.1.2016

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 text **`python`** and Pen&paper questions by text **`pen&paper`**

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 **??**. 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

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

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

   Consider the traditional shallow neural network architecture of Figure **??**. Suppose our inputs are 28×28 bitmaps of digits from categories $\{0, 1, \ldots, 9\}$.

   a) Let the network structure be the following:

   - The input is $28 \times 28 = 784$-dimensional

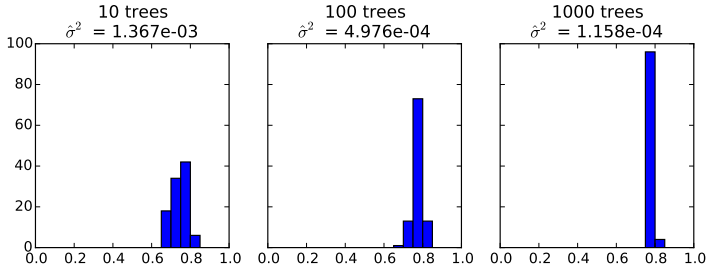

Figure 1: Variances of random forest with different number of trees.

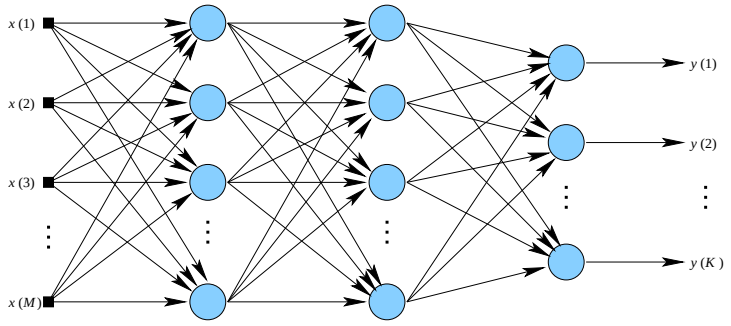

Figure 2: Vanilla neural network.

- 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.

```
model = Sequential()

w, h = 3, 3

model.add(Convolution2D(32, w, h,
          input_shape=(1, 28, 28),
          border_mode='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Convolution2D(32, w, h,
          border_mode='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(10, activation = 'softmax'))
```

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 training a Keras 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) $\rightarrow$ (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)`.

Also prepare your target array `y` to size (660, 2): the first column is 1 for class 1 items, and 0 otherwise. The second column is the complement of the first:

$$\mathbf{y} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ \vdots & \vdots \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

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

   Define the network of Question 2b in your code. However, for better performance add the following changes:

   - Add one dense layer with 100 nodes after the two convolutional ones.
   - Set the activation to all but the last layer into "relu". See Keras documentation for instructions.
   - Add a dropout regularizer after the 2nd convolutional layer with $p = 0.2$.
   - Also adjust input and output sizes to match this data.

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

   Compile and train the network as described in

   `http://keras.io/models/#sequential`

   Use the following parameters:

   - **Loss:** binary crossentropy (also called log loss; see previous exercises)
   - **Optimizer:** stochastic gradient descent
   - **Minibatch size:** 32
   - **Number of epochs:** 20

   Also set `validation_split = 0.1`, and `show_accuracy = True` for `.fit()`, so the optimizer will compute test error every epoch.