SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 4: November 18–November 22, 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** *Compute the gradient of the log-loss.*

   In the lectures we defined the *logistic loss function*:

   $$\ell(\mathbf{w}) = \sum_{n=0}^{N-1} \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)), \tag{1}$$

   and computed its gradient $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}}$. Here, $\mathbf{x}_n \in \mathbf{R}^P$ and $y_n \in \{-1, 1\}$ are the inputs and labels for the samples $n = 0, 1, \ldots, N-1$, and $\mathbf{w} \in \mathbf{R}^P$ are the model parameters to be learnt.

   The *L_2-regularized logistic loss* is defined by:

   $$\ell(\mathbf{w}) = \sum_{n=0}^{N-1} \ln(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)) + C \cdot \mathbf{w}^T \mathbf{w}, \tag{2}$$

   with $C \geq 0$ the regularization strength parameter. Compute the gradient of the regularized loss.

   Hint: For finding the gradients of vector functions, check the document at
   `http://www.kamperh.com/notes/kamper_matrixcalculus13.pdf`

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`** *Implement gradient descent for log-loss.*

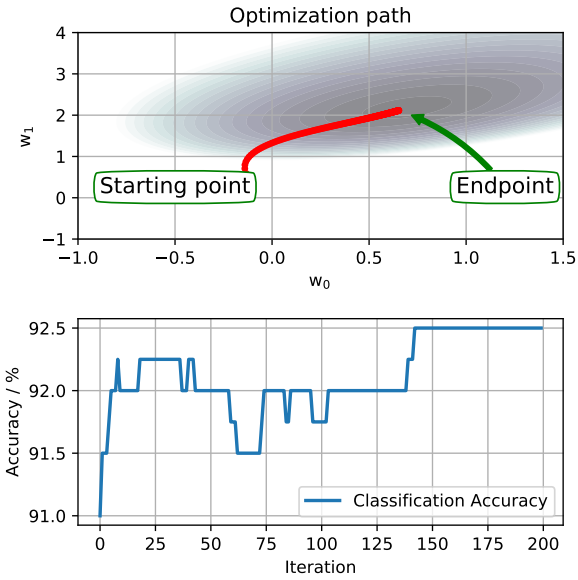
a) Implement a log-loss minimization algorithm for the loss of Equation (1). You may use the template provided by the teaching assistant.

b) Apply the code for the data downloaded from

`https://github.com/mahehu/SGN-41007/tree/master/exercises/Ex5/log_loss_data.zip`

The data is in CSV format. Load `X` and `y` using `numpy.loadtxt`.

c) Plot the path of $\mathbf{w}$ over 100 iterations and check the accuracy (see plots below).



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

   Edit the network of Question 2 such that the `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.*

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

   b) Train the model with the GTSRB dataset from last week.

      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.