DATA.ML.200 Pattern Recognition and Machine Learning

Exercise Set 4: November 16–November 20, 2020

- Exercises consist of both **pen&paper** and **python** assignments.
- Prepare a single PDF and return to Moodle on Friday, November 20th at 23:55 at the latest.
- Mark on 1st page which exercises you did.

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.

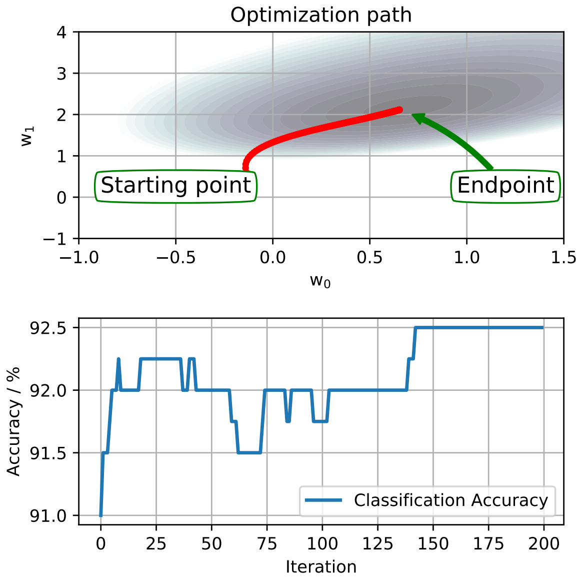
```
N = 10          # Number of feature maps
w, h = 5, 5     # Conv. window size

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).


Optimization path
Starting point
Endpoint
w_1
w_0
Accuracy / %
Classification Accuracy
Iteration

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

There are two options for this task. Either a) or b) will be enough.

a) Study from the slides and Tensorflow/Keras documentation how to instantiate a pretrained **Mobilenet V2** model with input shape `64x64x3` and 9 outputs. Last layer should be a softmax layer.

b) Implement a neural network such that tf.keras' `model.summary()` gives the following output. Last layer should be a softmax layer.

```
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, 9)                 909
=================================================================
Total params: 80,273
Trainable params: 80,273
Non-trainable params: 0
```

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

a) Compile the network of Question 4 above.

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:** Adam or 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.