DATA.ML.200 Pattern Recognition and Machine Learning

Exercise Set 3: November 6–November 13, 2020

- Exercises consist of both **pen&paper** and **python** assignments.
- Prepare a single PDF and return to Moodle on Friday, Novermber 13th at 23:55 at the latest.
- Mark on 1st page which exercises you did.

1. **pen&paper** *Design an LDA classifier manually.*

   A dataset consists of two classes, whose distributions are assumed Gaussian, and whose sample covariances and means are the following:

   $$\boldsymbol{\mu}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad \boldsymbol{\mu}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$
   $$\mathbf{C}_0 = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{C}_1 = \begin{pmatrix} 3 & -1 \\ -1 & 1 \end{pmatrix}$$

   Calculate the projection vector $\mathbf{w}$. In order to be fully manual, invert the 2×2 matrix using the rule

   $$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

2. **pen&paper** *Compute the threshold and classify.*

   In the lecture slides, we proposed to define the threshold T as the mean of the projected class means: $T = \frac{1}{2}\mathbf{w}^T(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)$. This approach does not take into account the fact that the two classes have different spreads, and the threshold should probably not be exactly at the center.

   A more appropriate approach computes the projection of the multivariate Gaussians and sets the threshold as we did in detection theory. The projected Gaussians are univariate normal: $\mathcal{N}(\mathbf{w}^T\boldsymbol{\mu}_1, \mathbf{w}^T\mathbf{C}_1\mathbf{w})$ and $\mathcal{N}(\mathbf{w}^T\boldsymbol{\mu}_2, \mathbf{w}^T\mathbf{C}_2\mathbf{w})$. Formulate the classification problem as a likelihood ratio test and choose the threshold based on that.

   Which class will be predicted for sample $\mathbf{x} = (1, 2)$?

3. **python** *Train sklearn classifiers for the GTSRB task.*

   In this exercise we will use sklearn for categorization of traffic signs. Download the following file:

   `https://www.dropbox.com/s/1tuuoe6qd9jmi3b/GTSRB.zip`

Each folder contains images from the German Traffic Sign Recognition Benchmark (GTSRB) arranged to folders by class. This is a subset of the whole dataset consisting of speed limits only[1].

Your task is to fill in the attached table. Implement your code such that you loop over the classifiers—do not copy-paste almost same code 6 times.

Model	*Accuracy Score*	*Training time (total)*	*Prediction time (per sample)*
3-Nearest Neighbor			
Linear Discriminant Analysis			
Logistic Regression			
SVM (linear kernel)			
SVM (RBF kernel)			
Random Forest (`n_estimators=20`)			

Notes:

- Feed the images directly as pixels to the classifiers (of course you will need to vectorize them first).
- The pictures are of different size and need to be resized in advance (recommended common size 32x32).
- Scale all images to the range 0...1.
- Images can be loaded using e.g., `matplotlib.pyplot.imread` or `cv2.imread`.
- Execution time can be measured like this:

```
import time

start_time = time.time()
<do something>
elapsed_time = time.time() - start_time
```

- An example output of your code is below:

```
Results for 3-NN:
-------------------
Accuracy is 87.21 %
Training time / batch [redacted] s
Test time / sample [redacted] ms
Results for LogReg:
-------------------
Accuracy is [redacted] %
Training time / batch [redacted] s
Test time / sample 0.011 ms
```

[1]The full dataset is available at `http://benchmark.ini.rub.de/?section=gtsrb`

4. **python** *Variance in model accuracy.* Since the `train_test_split` is random, you get a different result at every run. Study the use of `sklearn.model_selection.cross_val_score`, and compute the accuracies 5 times for the best model of exercise 3. What are the mean and standard deviation of these 5 numbers?

5. **python** *Train a convnet for the GTSRB data.* Extend your table with a convolutional neural network experiment. Either use the code you implemented during the intro course of take advantage of the below template.

```
layers = [tf.keras.layers.Conv2D(kernel_size = 3, filters = 32, activation='relu', input_shape=(32, 32, 3)),
          tf.keras.layers.MaxPool2D(),
          tf.keras.layers.Conv2D(kernel_size = 3, filters = 32, activation='relu'),
          tf.keras.layers.MaxPool2D(),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dense(num_classes, activation = 'softmax')]

model = tf.keras.Sequential(layers)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
model.fit(X_train, y_train, epochs = 10)

# model.predict() works like predict_proba in sklearn.
# You need to take np.argmax to get the index of largest probability.
```