SGN-41007 Pattern Recognition and Machine Learning

Exercise Set 5: February 8–February 10, 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`**

1. **`pen&paper`** *Derive the explicit mapping corresponding to a kernel trick.*

   In the lectures we saw that the kernel trick $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$ for $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$ corresponds to the mapping

   $$\begin{pmatrix} u \\ v \end{pmatrix} \mapsto \begin{pmatrix} u^2 \\ v^2 \\ \sqrt{2}uv \end{pmatrix}$$

   Find the explicit mapping corresponding to the inhomogeneous kernel $\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^2$ with $\mathbf{x}, \mathbf{y} \in \mathbf{R}^2$.

   Hint: Expand the kernel formula as far as you can. At that point, reformulate the result into a dot product of two 5-dimensional vectors; one composed of coordinates of $\mathbf{x}$ only and the other of coordinates of $\mathbf{y}$ only.

2. **`pen&paper`** *Compute the gradient of the log-loss.*

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

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

   a) Compute the formula for its gradient $\frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}}$.

   b) There are two alternative strategies for using the gradient.

   - **Batch gradient:** Compute the gradient from all samples and then apply the gradient descent rule $\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \ell(\mathbf{w})}{\partial \mathbf{w}}$.
   - **Stochastic gradient:** Compute the gradient from one sample and then apply the gradient descent rule. In other words, pretend $N = 1$ in formula 1.

   In the latter case, compute the next estimate for $\mathbf{w}$ when $\mathbf{x}_n = [-0.3, -1.7]^T$ and $y[n] = 1$ and $\mathbf{w} = [0.9, 0.9]^T$.

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

   a) Implement a log-loss minimization algorithm. 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 Python's own *pickle* serialization format (similar to `.mat` in Matlab)[1]. You can open the container in Python as follows:

```
import pickle
data = pickle.load(open("log_loss_data.pkl"))
```

Better yet, a more pythonic approach would use the `with` statement (this way the file pointer will not be left open after use):

```
import pickle
with open("log_loss_data.pkl", "r") as f:
    data = pickle.load(f)
```

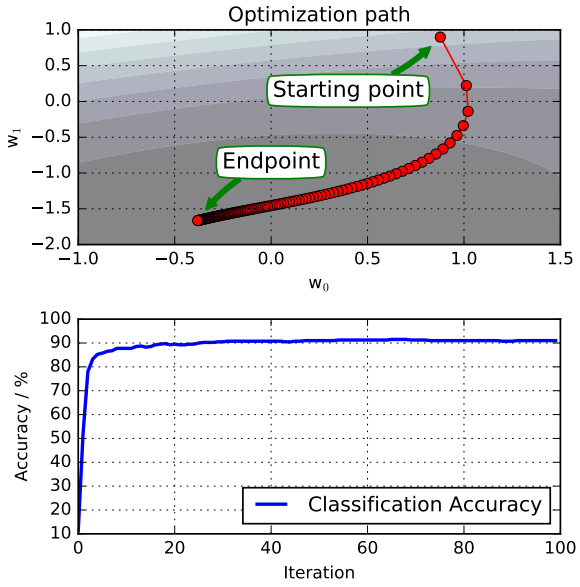
After this, `data` is a dict object (similar to `map` in C++). You can see the contents by:

```
>>> data.keys()
['y', 'X']
```

and access the contents as:

```
>>> X = data["X"]
>>> print X.shape
(400L, 2L)
```

c) Plot the path of w over 100 iterations and check the accuracy (see plots below).



[1]The name refers to *long term storage* of vegetables in vinegar.

4. **python** *Select appropriate hyperparameters for the GTSRB data.*

   Last week we trained classifiers for the German Traffic Sign Recognition Benchmark (GTSRB) dataset. It turned out that the SVM was really poor with default arguments, but changing the kernel pushed it to the top. In this exercise, we use brute force to find good hyperparameters for the classifiers (kernel, C, number of trees, etc.).

   Consider the following two classifiers

```
clf_list = [LogisticRegression(), SVC()]
clf_name = ['LR', 'SVC']
```

   Most important hyperparameters are the *regularization strength* `C` and the penalty type parameter `penalty`, which can have values "`l1`" and "`l2`".

   In order to use the same range for the two methods, you need to scale the data to zero mean and unit variance using `sklearn.preprocessing.Normalizer`.

   Implement a grid search over these two parameters along the following lines:

```
for clf,name in zip(clf_list, clf_name):
    for C in C_range:
    for penalty in ["l1", "l2"]:
            clf.C = C
            clf.penalty = penalty
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_test)
            score = accuracy_score(y_test, y_pred)
```

   A reasonable range for `C` is $C \in \{10^{-5}, ..., 10^{0}\}$.

5. **python** *Train ensemble methods with the GTSRB data.*

   a) Train a 100-tree Random Forest classifier with the GTSRB and compute the accuracy on the test set.

   b) Train a 100-tree Extremely Randomized Trees classifier with the GTSRB and compute the accuracy on the test set.

   c) Train a 100-tree AdaBoost classifier with the GTSRB and compute the accuracy on the test set.

   d) Train a 100-tree Gradient Boosted Tree classifier with the GTSRB and compute the accuracy on the test set.