AIML-15: Machine Learning, Homework 2. Support Vector Machine and Model Selection

November 23, 2015

General information. Problem solutions should be submitted in PDF format in report style (no source code listings required). All reports must be submitted before December 20 to the moodle (elearning) system. It is advised to use Python as a programming language, but you can use any language of your choice (at your own risk). In case you use Python, free `Anaconda` distribution comes with all needed packages:

`https://www.continuum.io/downloads`

In particular, you might find useful `scikit-learn` general machine learning library and `matplotlib` plotting facilities. When in doubt, read the manual and take a look at the large set of examples:

`http://scikit-learn.org/stable/documentation.html`

`http://scikit-learn.org/stable/auto_examples/`

`http://matplotlib.org/examples/`

Data preparation. In this homework you will work with the `MNIST` [1] dataset composed from 10 classes of handwritten digits. The dataset contains ≈ 70000, 28×28 images. Steps:

- If you are using Python and `scikit-learn`, you can get training data by running:

  ```
  from sklearn.datasets import fetch_mldata
  mnist = fetch_mldata('MNIST_original')
  X = mnist.data     # 70000 by 784 matrix of instances
  y = mnist.target   # 70000 vector of labels
  ```

  This might take some time when you execute it for the first time, because this command will download the dataset.

- If you are using Matlab, you can use `http://www.cs.nyu.edu/~roweis/data/mnist_all.mat`.

- Otherwise, in binary format from `http://yann.lecun.com/exdb/mnist/`.

Training Linear Support Vector Machine (SVM).

- Select the subset of $\boldsymbol{X}$ and $\boldsymbol{y}$, which belongs only to classes 1 and 7.
- Standardize, shuffle, and split selected data into the training, validation, and testing sets as $50\%, 20\%, 30\%$.
- Train linear binary SVM, varying parameter C in the range of your choice, and plot it's accuracy on the *validation* set against every choice of C. In Python you can use `scikit-learn`, e.g.,

```
from sklearn.svm import LinearSVC
cls = LinearSVC(C=1)
cls.fit(X_train, y_train)     # Train on X_train, y_train
accuracy = cls.score(X_test, y_test)     # Test on X_test, y_test
```

  and for plotting you can use `pylab` library,

```
import pylab; pylab.plot(C_range, accuracies)
```

- Which C will you use for the final classifier? Why?
- Train linear binary SVM once again, setting the best C. Test your classifier on the testing set and report obtained score.

Training Multiclass Non-Linear SVM.

- From $\boldsymbol{X}$ and $\boldsymbol{y}$ select examples which belong only to classes $0, 1, 2, 3, 4$.
- Standardize, shuffle, and split selected data into the training and testing sets as $50/50\%$ in a *stratified* way. Stratified means that all classes should be represented in both training and testing sets, according to the splitting ratio. Consider label set $y = \{1, 1, 2, 2, 3, 3, 3, 3\}$, where example of stratified splitting is , $y_{\text{train}} = \{1, 2, 3, 3\}$, and $y_{\text{test}} = \{1, 2, 3, 3\}$, and a counter-example would be $y_{\text{train}} = \{1, 3, 3, 3\}$, and $y_{\text{test}} = \{2, 2, 3, 3\}$. Why is it important to do stratified splitting?
- Train a multiclass non-linear SVM with Gaussian kernel function in One-vs-All (OVA) way. You can train and get margin values of a binary non-linear SVM like this:

```
from sklearn.svm import SVC
cls = SVC(C=0.000001, kernel='rbf', gamma=10000)
cls.fit(X_train, y_train)     # Train on X_train, y_train
# Margin (decision) values of classifier
margins = cls.decision_function(X_test)
```

- How can you use it in for multiclass OVA classification? Write down a short explanation.

- Train multiclass non-linear SVM and report the accuracy on the testing set.
- Tune C and γ values of multiclass non-linear SVM using the *grid search*, to give the best performance. Remember, you cannot touch testing set during tuning! Explain your steps for tuning these hyper-parameters. Finally, when you have tuned everything, report the accuracy of the classifier on the testing set.

References

[1] Y. LeCun, C. Cortes, and C. JC Burges. The mnist database of handwritten digits, 1998.