Machine Learning 2
Exercise Sheet 4

Prof. Dr. Dr. Lars Schmidt-Thieme, Brad Baker
Information Systems and Machine Learning Lab
University of Hildesheim

May 10th, 2017

Submission until May 16th, 8:00 AM by learnweb.
Please put your name in all filenames and somewhere visible in the margins of the pdf.
Non-pdf submissions for non-programming exercises will not be graded.

For this sheet, it is a good idea to look at the primary source papers for Advanced SVM methods:

- PEGASOS: Shalev-Shwartz et. al. 2007
- Adaptive Multi-Hyperplane Machine: Wang et. al. 2011
- Dual Coordinate Descent SVM: Hsieh et. al. 2008
- BudgetedSVM Toolbox: Djuric et. al. 2013

Exercise 7: Sub-Gradient SVMs / Primal Optimization (10 Points)

a) (3 points) The primal prooblem which is solved to learn a Support Vector Machine is given as

$$minimize \frac{1}{2}||\beta||^2 + \gamma \sum_{n=1}^{N} \xi_n.$$

Rewrite this in terms of a loss function by treating one of the terms as a loss. Identify the kind of loss function you have used, and justify why you have chosen this loss

b) (3 points)

- For the loss function you have just derived, identify a gradient-based optimization method for minimizing the loss, and justify why you have made this choice.
- Derive the relevant update rule for the method you have identified.

c) (4 points) "One of the main benefits of SVMs is that they can be used with *kernels* rather than with direct access to the feature vectors $\mathbf{x}$."

In section 4 of Shalev-Shwartz et al 2007, a general primal-based method for SVM optimization is derived for kernel function $K(\mathbf{x_i}, \mathbf{x_j})$.

The polynomial kernel of degree d is given as

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i}^T \mathbf{x_j} + c)^d$$

.

Using the Shalev-Shwartz paper as a guide, derive an update rule for the PEGASOS algorithm which utilizes either the general polynomial kernel, or the quadratic kernel, where $d = 1$ and $c = 1.\ 0$

Exercise 8: Dual Coordinate-Descent and Adaptive Multi Hyperplane Machines (10 Points)

a) (1 points) The dual problem for optimizing an SVM is given as

$$\textbf{minimize} f(\alpha) = 1/2\alpha^T Q\alpha - 1^T\alpha\,, Q_{n,m} = y_n y_m x_n^T x_m.$$

Derive the coordinate descent rule for α_n, and identify the bottle-neck in this initial update.

b) (6 points) Suppose we have the following data set for a multi-class classification problem.

$$X = \begin{pmatrix} 1 & 69 \\ 1 & 71 \\ 1 & 70 \\ 3 & 60 \\ 4 & 60 \\ 4 & 59 \\ 6 & 70 \\ 6 & 72 \\ 9 & 72 \end{pmatrix}, Y = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \end{pmatrix}.$$

We are trying to train a multi-class SVM, and we start with a single hyperplane for classifying the third class

$$\beta_{3,1} = (67, 1, -1).$$

- Plot the data and this given hyperplane.
- Using the update criterion for an Adaptive Multi-Hyperplane Machine, decide whether or not to initialize a new hyperplane for classifying the third class, and perform one stochastic gradient step for each of the hyperplanes for this class.
- Plot your updated hyperplanes and compare the results with the initial given hyperplane.

c) (3 points) Read the Djuric et. al. 2013 paper "BudgetedSVM: A Toolbox for Scalable SVM".

- Choose one of the methods discussed in that paper, and briefly summarize the general approach.
- Compare the results of the paper either with PEGASOS, Dual Coordinate Descent, or Adaptive Multi-Hyperplane Machines.

1 Bonus: Advanced SVMs with R (10 points)

Implement one of the scalable SVM methods discussed in the lecture, or presented in the Djuric paper. Some implementations are available online in the MATLAB programming language, so please port the implementation into R or your favorite programming langauge. MATLAB submissions will not be accepted.

If you are not famliar with the IRIS dataset, read the wikipedia article: `https://en.wikipedia.org/wiki/Iris_flower_data_set`.

Test your chosen method using the IRIS data set, which is pre-loaded into R, or which can be downloaded from the UCI repository: `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`. You should classify according to species.

Compare your method either with the vanilla R implementation of SVM from the libsvm package (`https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf`) or with a native python implementation.

For simplicity's sake, use a linear kernel, performing 5-fold cross validation. Do not forget to randomly shuffle data instances between your folds.