HW Assignment 7 (Due by 1:30 pm on Nov 19)

1 Theory (140 points)

1. [**Normalized Kernels**, 50 + 25 points]
   Let $K : X \times X \to R$ be a kernel function defined over a sample space X.

   (a) Prove that the function below (a *normalized kernel*) is a valid kernel.

   $$\frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$$

   (b) Why it would not make sense to normalize a Gaussian kernel?

   (c) [**Bonus**] Which types of input data would benefit from normalizing the kernel function? Explain why, and provide real world examples.

2. [**Support Vector Machines**, 50 points]
   Prove that the sum of slacks $\sum \xi_n$ from the objective function of the SVM formulation with soft margin is an upper bound on the number of misclassified training examples.

3. [**Max Margin Hyperplanes**, 20 points]
   Consider the constrained optimization SVM problem for the separable case shown on slide 12. Show that, if the 1 on the right-hand side of the inequality constraint is replaced by some arbitrary constant $\gamma > 0$, the resulting maximum margin hyperplane is unchanged.

4. [**Kernel Techniques**, 20 + 20 points]
   (a) Show that if $k_1(\mathbf{x}, \mathbf{y})$ and $k_2(\mathbf{x}, \mathbf{y})$ are valid kernel functions, then $k(\mathbf{x}, \mathbf{y}) = k_1(\mathbf{x}, \mathbf{y})k_2(\mathbf{x}, \mathbf{y})$ is also a valid kernel.
   (b) **(*)** Show that if A is a symmetric positive semidefinite matrix, then $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T A\mathbf{y}$ is a valid kernel. *Hint: Inderjit Dhillon's Linear Algebra Background describes some useful properties of symmetric positive semidefinite matrices.*

5. [**Positive Definite Matrices (*)**, 20 points]
   Show that a diagonal matrix $\mathbf{W}$ whose elements satisfy $0 < W_{ii} < 1$ is positive definite. Show that the sum of two positive definite matrices is itself positive definite.

6. [**Large Margin Perceptron (*)**, 30 points]
   Let $\mathbf{u}$ be a current current vector of parameters and $\mathbf{x}$ and $\mathbf{y}$ two training examples such that $\mathbf{u}^T(\mathbf{x} - \mathbf{y}) < 1$. Use the technique of Lagrange multipliers to find a new vector of parameters $\mathbf{w}$ as the solution to the convex optimization problem below:

   minimize:
   $J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{u}\|^2$

   subject to:
   $\mathbf{w}^T(\mathbf{x} - \mathbf{y}) \geq 1$

2 Text Classification (100 points)

Train and test the SVM algorithm on the *Spam vs. Non-spam* and *Atheism vs. Religion* classification problems, using the datasets provided for the previous assignment. Use a linear kernel, with the cost parameter $C = 5$. Report and compare the accuracy of the trained SVM models with the perceptron and average perceptron accuracies from the previous assignment.

3 Digit Recognition (200 points)

In this exercise, you are asked to run an experimental evaluation of SVMs and the perceptron algorithm, with and without kernels, on the problem of classifying images representing digits.

1. The UCI Machine Learning Repository at *www.ics.uci.edu/~mlearn* maintains datasets for a wide variety of machine learning problems. For this assignment, you are supposed to work with the Optical Recognition of Handwritten Digits Data Set. The webpage for this dataset is at:
   http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
   The actual dataset is located at:
   http://archive.ics.uci.edu/ml/machine-learning-databases/optdigits/

   Read the description of the dataset. Download the training set `optdigits.tra` and the test set `optdigits.tes`. Use the first 1000 examples in `optdigits.tra` for *development* and the rest of 2823 examples for *training*. Use all 1797 examples in `optdigits.tes` for *testing*. Scale all the features between [0, 1], as discussed in class, using the min and max computed over the training examples. Create training files for each of the 10 digits, setting the class to 1 for instances of that digit, and to -1 for instances of other digits, i.e. *one-vs-rest* scenario.

2. Train first the linear perceptron, with the number of epochs set to $T \in \{1, 2, ..., 20\}$. After training each linear perceptron, normalize the learned weight vector. Select for T the value that obtains the best overall accuracy on the development data, and use this value for the remaining perceptron experiments.

   Run experiments with the linear and kernel perceptron algorithms. For the kernel perceptron, experiment with polynomial kernels $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^d$ with degrees $d \in \{2, 3, 4, 5, 6\}$, and with Gaussian kernels $k(\mathbf{x}, \mathbf{y}) = exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$ with the width $\sigma \in \{0.1, 0.5, 2, 5, 10\}$. For each hyper-parameter value, you will have trained 10 models, one for each digit. In order to compute the label for a test or development example, you will run the 10 trained models and output the label that obtains the highest score. Compute the accuracy on the development data and identify the hyper-parameter value that obtains the best accuracy. Use the tuned hyper-parameter (d for poly-kernel, σ for Gaussian) to compute the overall performance on the test data.

   For each of the three perceptrons (linear, poly kernel and Gaussian kernel) report the total training time, the overall accuracy, and the number of support vectors. Show and compare the corresponding 4 confusion matrices. Which digit seems to be the hardest to classify? Which perceptron / kernel combination achieves the best performance? Which algorithms are slower at training time, and why?

3. Run the same experiments using SVMs instead of perceptrons, i.e. linear SVMs and SVMs with polynomial and Gaussian kernels. Use the same tuning scenarios for the hyper-parameters of the polynomial and Gaussian kernels. Use $C = 1$ in all SVM experiments. Report the same types of results and analysis as above, and compare with the perceptron results.

4 Tools

You are free to use MATLAB, R, or packages written in C++/Java/Python such as SVM-LIGHT (C), LIBSVM (C++, Java), or SCIKIT-LEARN (Python) to complete the implementation part of this assignment. Their web sites contain plenty of documentation on how to use them. If you use SCIKIT-LEARN, the following functionality from the `sklearn.svm` will be useful:

1. `SVC()`: This is the main class used for SVM classification models. Its implementation is based on LIBSVM. Make sure that you properly map the SVM hyper-parameters to the parameters in the constructor of this class. For example, the *gamma* parameter in the constructor corresponds to our $1/2\sigma^2$ coefficient in the Gaussian kernel. The formulas for the kernels implemented by SVC are described in this User Guide.

2. `decision_function(x)`: Once the classifier is trained, this will compute the distance between a sample `x` and the decision hyperplance. This is the quantity that you can use to determine the highest scoring class when training the 10 $one - vs - rest$ classifiers: once a classifier is trained for all 10 digits, given a sample `x` you compute this quantity for all 10 classifiers and select the class that corresponds to the classifier with largest decision function value.

3. `fit()`: This is the function used to train the classifier.

4. `predict(x)`: This is used to calculate the (binary) label for sample `x`.

LIBSVM, and therefore SCIKIT-LEARN too, already implement the *one-vs-rest* classification scheme. In this scheme, you can directly use the training dataset with the 10 original labels, and SCIKIT-LEARN will train the 10 binary classifiers for you. You can use this capability for this assignment, however bonus points will be given if you train the 10 binary classifiers directly, as described for the perceptron algorithm above, by creating a binary training dataset for each class.

5 Submission

Turn in a hard copy of your homework report at the beginning of class on the due date. Electronically submit on Blackboard a hw07.zip file that contains the hw07 folder in which you place the code and the datasets. Make sure you include a README.txt file explaining how the code is supposed to be used to replicate the results included in the report. The screen output produced when running the code should be redirected to (saved into) an **output.txt** file.

On a Linux system, creating the archive can be done using the command:

```
> zip -r hw07.zip hw07
```

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. **Do not submit third-party ML packages on Blackboard!** Just explain in the REAMDE file how you use external packages.
4. Make sure your code runs correctly when used in the directory structure shown above.
5. **Type and nicely format the project report**, including discussion points, tables, graphs etc. so that it is presentable and easy to read.
6. Working code and/or correct answers is only one part of the assignment. The project report, including discussion of the specific issues which the assignment asks about, is also a very important part of the assignment. Take the time and space to make an adequate and clear project report. On the non-programming learning-theory assignment, clear and complete explanations and proofs of your results are as important as getting the right answer.