Empirical/Programming Assignment 4

Due date: Thursday, 12/3 (by the beginning of class, both paper and electronically)

1 Introduction

In this assignment you will implement the primal and dual/kernel versions of $k$-nearest neighbors and the perceptron algorithm and evaluate them.

2 Data

For this assignment we provide you with data split into train and test files for 4 datasets where A, B are artificial and back and sonar are from the UCI repository. You may want to use code from previous assignments to read arff files into your learning programs.

3 Your Tasks

3.1 The Perceptron Algorithm

We recap here the code for Perceptron with Margin (PwM) and its dual version as discussed in class. The algorithms (with parameters $\tau$, $I$) are as follows where examples $\bar{x}$ and the weight vector $\bar{w}$ are in $R^m$, the labels ($y$) are in {$-1, +1$}, the sign function gives a value in {$-1, +1$}, and the number of examples in the training set is $N$. The notation $\bar{w} \cdot \bar{x}$ is a shorthand for $\sum_j w_j x_j^i$.

Primal Perceptron with Margin:

1. Initialize $w_k = 0$ for all $k \in \{1, \ldots, m\}$.
2. Repeat for $I$ Iterations
   (a) For each example $(\bar{x}, y)$ in training set do:
   - (Classify): $O = sign(\bar{w} \cdot \bar{x})$.
   - (Update):
     if $y(\bar{w} \cdot \bar{x}) < \tau$
     For all $k \in \{1 \ldots m\}$, $w_k = w_k + y x_k^i$
3. Output the last weight vector $\bar{w}$ as the final hypothesis.

Dual Perceptron with Margin:

1. Initialize $\alpha_k = 0$ for all $k \in \{1, \ldots, N\}$.
2. Repeat for $I$ Iterations
   (a) For each example $(\bar{x}, y)$ in training set do:
   - (Classify): $O = sign(\sum_k \alpha_k y_k (\bar{x}^k \cdot \bar{x})$).
   - (Update):
     if $y(\sum_k \alpha_k y_k (\bar{x}^k \cdot \bar{x})) < \tau$
     Assign $\alpha_i = \alpha_i + 1$
3. Output the last weight vector $\bar{\alpha}$ as the final hypothesis.
**Kernel Perceptron with Margin:**
Replace $\vec{x}^k \cdot \vec{x}^i$ in the Classify and Update lines of the dual perceptron with $K(x^k, x^i)$.

**How to run the algorithms:** You will be running the primal perceptron and the kernel perceptron. For parameters, use $I = 50$ and $\tau = 0.1 * A$ where $A = \frac{1}{N} \sum_{i=1}^{N} \|x^i\|$ is the average norm of training examples, where for the kernel perceptron the norm of $x^i$ is replaced with $\sqrt{K(x^i, x^i)}$. This needs to be calculated before training starts.

Your implementation of the primal algorithm should add a feature with constant value 1 to all examples to account for the threshold. This should be done inside the algorithm so that the dataset files would not require changing.

For evaluation, we treat the algorithm as a batch algorithm and output the last vector for use in testing (as in step 3 of the algorithms). Given a test set we can evaluate the error rate on this set in the usual manner.

### 3.2 Nearest Neighbors
Recall that given a dataset and a test example the $k$-NN algorithm finds the $k$ training examples closest to the test example and predicts the mode (most frequent) among their labels. The primal version of the algorithm calculates Euclidean distance $d(\vec{u}, \vec{v}) = \|\vec{u} - \vec{v}\|$. For consistency with the other algorithms (and kernels) the primal version should add a feature with constant value 1 to all examples. The dual version calculates distance via kernels as $d(\vec{u}, \vec{v}) = \sqrt{K(\vec{u}, \vec{u}) + K(\vec{v}, \vec{v}) - 2K(\vec{u}, \vec{v})}$. In this assignment we use $k = 1$, that is, 1-nearest-neighbors.

### 3.3 Kernels
For use with the kernel methods you should implement the polynomial kernel $K(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + 1)^d$, and the RBF kernel $K(\vec{u}, \vec{v}) = e^{-\frac{\|\vec{u} - \vec{v}\|^2}{2s^2}}$. Note that the linear kernel (polynomial kernel with $d = 1$) adds a 1 to the standard inner product similar to the requirement for the primal algorithms.

### 3.4 Experiments and Results to Report
Implement your own versions of the algorithms run them on the 4 datasets where for the kernel versions you should run Polynomial kernels with $d = 1, 2, 3, 4, 5$ and the RBF kernel with $s = 0.1, 0.5, 1$. This means that for each dataset you should have $2^4 * 9$ results including the primal versions. Tabulate or plot the test set accuracy results in some way that is easy to view and report on your observations.

In your report, please make sure to address at least the following questions: Are the primal and dual version of algorithms with linear kernel indeed identical? How does the kernel parameter affect the results for the polynomial and RBF kernels? Is the effect consistent across algorithms? How do Perceptron and $k$-NN compare in the experiments across the kernels?

### 3.5 Prepare a Program or Script for Testing
In addition to the programs above, please prepare a program (or bash script) that will allow us to test your code on additional data. Your program (please call it `test`) should expect the data files `additionalTraining.arff` and `additionalTest.arff` in the same directory where the program is running and should require no arguments. When run, (that is, we will run `./test` in linux) your program should output two lines, each with 9 numbers giving the accuracies for $k$-NN (first line) and perceptron (second line) in the order: primal, polynomial kernel $d = 1, 2, 3, 4, 5$, RBF kernel with $s = 0.1, 0.5, 1$.

### 4 Submitting your assignment
- You should submit the following items both electronically and in hardcopy:
  (1) All your code for data processing, learning algorithms, test program, and the experiments. Please write clear code and document it as needed.
Please include a README file with instructions how to compile and run your code both for the main experiments and for the test program.

(2) A short report with the tables or plots as requested and a discussion with your observations from these plots.

- Please submit a hardcopy in class.

- Please submit electronically using provide by 1:30 (class time). Put all the files from the previous item into a zip or tar archive, for example call it myfile.zip. Then submit using provide comp135 a4 myfile.zip.

Your assignment will be graded based on the code, its clarity, documentation and correctness, the presentation of the tables/plots, and their discussion. We will plan to test you code with additional data to help us in grading it for correctness.