Empirical/Programming Assignment 4

Due date: Wednesday, December 3 (electronically by 12:00 noon and in hardcopy in class)

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 5 datasets where $A$, $B$, $C$ 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 ($\text{PwM}$) from assignment 2 and its dual version as discussed in class. The algorithms (with parameters $\tau, I$) are as follows where examples $\vec{x}^i$ and the weight vector $\vec{w}$ are in $R^m$, the labels ($y^i$) are in $\{-1, +1\}$, the sign function gives a value in $\{-1, +1\}$, and the number of examples in the training set is $N$.

**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 $(\vec{x}^i, y^i)$ in training set do:
      • (Classify): $O = \text{sign}(\vec{w} \cdot \vec{x}^i)$.
      • (Update): if $y^i(\vec{w} \cdot \vec{x}^i) < \tau$
         
   For all $k \in \{1 \ldots m\}$, $w_k = w_k + y^i \vec{x}_k^i$
3. Output the last weight vector $\vec{w}$ 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 $(\vec{x}^i, y^i)$ in training set do:
      • (Classify): $O = \text{sign}(\sum_k \alpha_k y^k (\vec{x}^k \cdot \vec{x}^i))$.
      • (Update): if $y^i(\sum_k \alpha_k y^k (\vec{x}^k \cdot \vec{x}^i)) < \tau$
         
   Assign $\alpha_i = \alpha_i + 1$
3. Output the last weight vector $\vec{\alpha}$ as the final hypothesis.
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 dual perceptron the norm of $x^i$ is replaced with $\sqrt{k(x^i, x^i)}$. This needs to be calculated before training starts. 

As in assignment 2, we treat the algorithm as a batch algorithm and output a fixed vector for use in testing. Given a test set we can evaluate the error rate on this set in the usual manner. 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.

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 $d(\vec{u}, \vec{v}) = \sqrt{k((\vec{u}, \vec{u}) + k(\vec{v}, \vec{v}) - 2k(\vec{u}, \vec{v}))}$.

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 $k(\vec{u}, \vec{v}) = e^{-\frac{1}{2s^2} \| \vec{u} - \vec{v} \|^2}$. 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 5 datasets where for the kernel versions you should run Polynomial kernels with $d = 1, 2, 3, 4, 5$ and the RBF kernel with $s = 0.1, 0.5, 1$, so you should have 9*2 algorithm variants in all 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?

4 Submitting your assignment

- You should submit the following items both electronically and in hardcopy:
  - (1) All your code for data processing, algorithms, and the experiments (please write clear code and document it as needed).
  - (2) The results and the report on your observations.

- Please submit electronically using provide by 12:00 noon: 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.

- Please submit a hardcopy in class (i.e., 4:30 pm).