AI201
Programming Assignment 4
Comparison of Boosted Perceptrons and SVM

Instructor: Pros Naval

Individual Submissions
Due: April 23, 2025 @ 12:00 noon

1. Introduction

We compare two high-performance classification methods in this programming assignment. The first one, called Adaboost, combines different weak learners whose training accuracies are slightly better than random (for a binary classifier) and transforming them into an ensemble of classifiers that can produce classification errors that are arbitrarily close to zero. This method reweights examples in the original training set as a function of whether they were misclassified in the previous iteration, thus producing a strong learner. We choose the Perceptron Classifier for our weak learner and train it using the Pocket Algorithm since our data is not linearly separable.

Support Vector Machines (SVM) are kernel-based learning algorithms that implicitly map nonlinearly separable input data into a higher dimensional feature space where an optimal separating hyperplane can be efficiently computed.

The pseudo codes for Adaboost and Pocket Algorithm are given below:

1.1. Adaboost

Given examples $S = \{(x_1, y_1), ..., (x_N, y_N)\}$ where $x \in \mathbb{R}^d, y_i \in \{+1, -1\}$, d: number of features, N: number of training examples:

1. Initialize $w_1(i) = 1/N$. (uniform distribution over data)
2. for $t = 1, ..., K$
   (a) Select new training set S_t from S with replacement according to w_t
   (b) Train weak learner L on S_t to obtain hypothesis h_t
   (c) Compute training error ϵ_t of h_t on S:

$$\epsilon_t = \sum_{j=1}^{N} w_t(j)\delta(y_j \neq h_t(x_j))$$

where

$$\delta(y_j \neq h_t(x_j)) = \begin{cases} 1 & \text{if } y_j \neq h_t(x_j) \\ 0 & \text{otherwise} \end{cases}$$

   (d) Compute coefficient α_t:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

   (e) Compute new weights on S:
   for $i = 1$ to N:

$$w_{t+1}(i) = \frac{w_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

   where Z_t is a normalization factor chosen so that w_{t+1} remains a probability distribution:

$$Z_t = \sum_{i=1}^{N} w_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

At the end of K iterations of the algorithm we have hypotheses $h_1, h_2, ..., h_K$ and $\alpha_1, \alpha_2, ..., \alpha_K$.

The ensemble classifier is then

$$H(x) = sgn \left(\sum_{t=1}^{K} \alpha_t h_t(x) \right)$$

1.2. Pocket Algorithm

Given training examples $\{(x_1, y_1), ..., (x_N, y_N)\}$ where $x \in \mathbb{R}^d, y_i \in \{+1, -1\}$, d: number of features, N: number of training examples. The Pocket Algorithm outputs a weight vector $w \in \mathbb{R}^{d+1}$ (w_0 is the bias) corresponding to a trained neuron.

Temporary Memory:
$v \in \mathbb{R}^{d+1}$: current weight vector for the perceptron
n_v: number of consecutive iterations for which weight vector v correctly classified the examples

n_w: number of consecutive iterations for which weight vector w correctly classified the examples

Pocket Algorithm:

1. Set $n_v = n_w = 0$ and $v_i = w_i = 0$ for $i = 0, 1, ..., d$; Set $itercnt = 0$
2. Randomly choose an input pattern x_j and associated label y_j from the training set
3. Compute the neuron's output given x_j:

$$\hat{y}_j = \begin{cases} +1 & \text{if } v \cdot x_j \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

4. If v correctly classifies x_j (i.e, $\hat{y}_j y_j > 0$) then
   - set $n_v = n_v + 1$

   otherwise
   - if $n_v > n_w$ then
     - set $w = v$ and $n_w = n_v$ (put current vector in the pocket)
   - change vector v by setting $v_i = v_i + y_j x_{ij}$ for $i = 0, 1, ..., d$. (note: x_{ij} is the ith element of sample x_j)
   - set $n_v = 0$
5. $itercnt = itercnt + 1$
6. Go to step 2 unless $itercnt > maxitercnt$

2. Perceptron Classifier Construction

In the first part of this PA, you are to implement a perceptron learner and test its performance on a binary classification problem. Use only the numpy library for implementing the Perceptron Classifier. The construction steps are as follows:

1. Generate data consisting of 100 two-dimensional vectors taken from a normal distribution with $\mu_1 = [0, 0]^T, \sigma_1 = I$ and label them as class "-1". Form a class "-1" training subset by setting aside 50 points and let the remaining 50 points serve as part of the test set. Do the same for class "+1" for another normal distribution with $\mu_2 = [10, 10]^T, \sigma_2 = I$. Merge the subsets to form the training and test sets.
2. Write code named `classify(···)` that implements the Pocket Algorithm for perceptron learning. Set $maxitercnt$ to 10000 iterations.
3. Write code named `predict(···)` to test the classifier and measure the sum of square errors for the test set.

3. Adaboost Construction and Evaluation

After having successfully implemented the perceptron learner, we now construct an ensemble of perceptrons that uses Adaboost. Use only the numpy library for implementing Adaboost.

1. Following the Adaboost pseudocode presented in the previous section, write code called `adabtrain` that implements the Adaboost algorithm with the Pocket Algorithm as the basic learner.
2. Form the training set by sampling 400 points from the banana dataset provided. Use the remaining 4900 points as your test set
3. Write code named `adabpredict` that classifies unknown/unseen data.
4. Using `adabtrain` and `adabpredict`, train the boosted perceptron algorithm and measure the training and test accuracies for the ensemble classifier having K learners
5. Write code that automatically plots the training and test accuracies against the number of learners K used for training. Use $K = 10, 20, 30, ...1000$.
6. Train and test your Boosted Perceptron on the splice dataset and plot the accuracy vs K curves for both training and test. Use 1000 points for training, 2175 points for training, and $K = 10, 20, 30, ...1000$

4. SVM Classifier

You have been provided LibSVM in this folder. Familiarize yourself with the codes. Compile the and test the codes.

Form a training dataset of 400 points and test set of 4900 points from banana dataset. Run the Support Vector Machine Classifier on these sets. Find the kernel and kernel parameters to obtain the best performance. Repeat for the splice dataset with 1000 training points and 2175 test points. Compare the performance of SVM with Boosted Perceptrons in terms of test accuracy, training and test speeds.

5. Deliverables

Deliverables for this Programming Assignment:

1. Jupyter Notebook with Python code for the Boosted Perceptron (40 points)
2. Plots of Training and Test Set Accuracies vs K for banana and splice datasets (20 points)
3. SVM Kernel and parameter settings for banana and splice datasets (20 points)

4. Comparison of SVM and Boosted Perceptron performance for banana and splice datasets (20 points)

The deadline for submission is **12:00 noon of April 23, 2025**. Email soft copy of report, Jupyter Notebook with Python code and files to `submit2pcnaval@gmail.com` with "[AI201: PA4 Submission] Your Name" on the subject line.