CS 4780/5780 Machine Learning

Assignment 3: Perceptron and SVM

Instructor: Thorsten Joachims

Course Policy:

Read all the instructions below carefully before you start working on the assignment, and before you make a submission

- Assignments are due at the beginning of class on the due date in hard copy.

- Write your NetIDs with submission date and time on the first page of the hard copy.

- No assignment will be accepted after the solution is publicized (about 4 days after due)

- The submission time of whatever you submit last (hard copy or CMS) is counted as the official submission timeand will determine the late penalty

- Upload only the code you wrote to CMS. Do not upload any additional libraries. Provide a README file with your submission that includes a list of all libraries and instructions needed to run your code.

- Late assignments can be submitted in class or to Ian Lenz in Upson Hall 5151. Since the fifth floor of Upson is locked on the weekends, weekend submissions (all code and answers to all questions) should be made digitally via CMS, with a hard copy delivered to Ian as soon as possible afterwards.

- All sources of material must be cited. Assignment solutions will be made available along with the graded homework solutions. The University Academic Code of Conduct will be strictly enforced, including running cheating detection software.

- No more than one submission per group.

Problem 1: Perceptron and Margins [35 points]

In this problem we will be working exclusively with a small dataset S in Figure 1. Consider all red (diamond) training examples as negative instances ($y_\diamond = -1$) and all blue (cross) training examples as positive instances ($y_\times = +1$). We will explore two variations on the standard perceptron algorithm you learned in class, and relate the perceptron to a hard-margin SVM.

(a) In class you learned the perceptron algorithm for finding an *unbiased* hyperplane that separates two classes with 0 training error. There is no such hyperplane, however, for the dataset S in Figure 1. A simple way to incorporate a bias term b is to augment the weight vector $\mathbf{w}$, such that $\mathbf{w}_{biased} = \langle \mathbf{w}, b \rangle$, and augment all training examples such that $\mathbf{x}_i := \langle \mathbf{x}_i, 1 \rangle$. Implement a biased perceptron learning algorithm and train it on S, following the order of training examples given in Figure 1. Compute $\mathbf{w}_{biased}$. Plot the resulting hyperplane. Does the resulting hyperplane maximize the margin between the two classes? (Note for all perceptron implementations in this problem:

initialize your weight vector $\mathbf{w}$ to the zero vector.)

(b) Sketch a hyperplane that achieves the largest hard margin in S *without* using an SVM package. Clearly mark all support vectors in your diagram. What is the weight vector $\mathbf{w}_{opt}$ and bias b_{opt} of the maximum margin hyperplane that achieves a functional margin of 1 for the support vectors? What is the geometric margin γ_{opt}? Also indicate which dual variables α_i are non-zero. Show all work leading you to your answer.

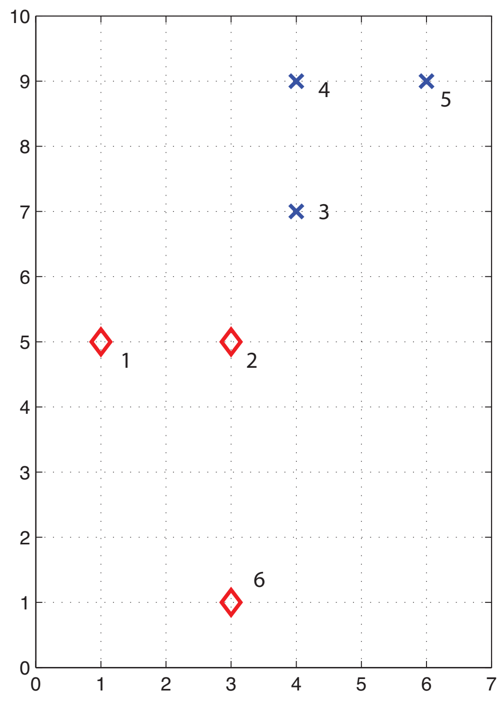

Figure 1: Training set S

(c) All of a sudden you get a strong and unstoppable urge to implement your own SVM. Unfortunately you have no background in convex optimization, so you decide to try something very simple instead. Being clever, you think up a way to modify the biased perceptron algorithm to allow you to 'tune' its margin. One simple way to accomplish this is to update the weight vector whenever a training example is within a user-specified fraction $\beta \in [0, 1)$ of the maximum margin γ.

$$\frac{y_i(\mathbf{w}^T\mathbf{x}_i + b)}{\|\mathbf{w}\|} < \beta\gamma \tag{1}$$

Set $\beta = 0.5$ and run your modified perceptron algorithm until convergence (train on the examples in the same order as indicated in Figure 1). Plot the resulting hyperplane (you found γ in part *b*). What do you observe?

(d) Rerun your algorithm until convergence for a range of values $\beta \in$\{*0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95*\}. Plot (on one graph) a hyperplane for each value of β (10 values total) overlaying the optimal hyperplane you found in part *b*.

(e) What do you observe? What do you expect to happen if you rerun this modified perceptron learning algorithm with different initializations of the $\mathbf{w}$ vector, and different ordering of the training examples, for different values of β, as β approaches 1? You don't need to perform the experiments, but justify your answer.

In the rest of the problem we will try to prove the mistake bound of this modified perceptron algorithm with $\beta = 0.5$. Each remaining part of the problem is designed to lead you to the final bound. If you are unable to complete any single part, you can use the provided result as a given for the parts that follow.

Assume that all datapoints are rescaled into a ball of radius 1, so that $R^2 = 1$ in all of the bounds you have seen in class. This does not affect the generality of the proofs.

(i) You have already seen in class, that we can bound the *squared length* of the weight vector in the standard perceptron on every mistake update as follows:

$$||\mathbf{w}_{k+1}||^2 \leq ||\mathbf{w}_k||^2 + 1$$

Show that this bound implies the following bound on the *length* of the weight vector in the standard perceptron:

$$||\mathbf{w}_{k+1}|| \leq ||\mathbf{w}_k|| + \frac{1}{2||\mathbf{w}_k||}$$

(ii) Now let's modify the bound above to account for a modified perceptron learning rule specified in part c of this problem. **Show that** for this modified perceptron learning algorithm, the following bound on the *length* of the weight vector holds:

$$||\mathbf{w}_{k+1}|| \leq ||\mathbf{w}_k|| + \frac{1}{2||\mathbf{w}_k||} + \frac{\gamma}{2}$$

Hint 1 It may be helpful to recall that $||\mathbf{w}_{k+1}||^2 = ||\mathbf{w}_k + y\mathbf{x}||^2$ and that a squared norm of any vector $||\mathbf{v}||^2$ can be written as $\mathbf{v} \cdot \mathbf{v}$. Additionally, the following inequality may be useful: $\sqrt{1+x} \leq 1 + \frac{x}{2}$. In contrast to the perceptron update rule you saw in class, you need to consider how the new update rule (that incorporates the margin) will affect the upper bound.

Hint 2 You may also use geometric intuition instead, and show that the growth in $||w_{k+1}||$ is also affected by a nonzero projection of $y\mathbf{x}$ on $\mathbf{w}$ under the new update rule, and use that to help you derive the upper bound.

(iii) **Show that** the following simpler bound holds if $||\mathbf{w}_k|| \geq \frac{2}{\gamma}$:

$$||\mathbf{w}_{k+1}|| \leq ||\mathbf{w}_k|| + \frac{3\gamma}{4}$$

(iv) **Finally show that** the total number of updates (mistakes) K for this modified perceptron algorithm can be bounded by $K \leq 8/\gamma^2$. Use the lower bound on $||w_k||$ together with the upper bound above to arrive to the desired upper bound on K. Notice that the lower bound on $||w_k||$ is the same as for the perceptron algorithm you analyzed in class.

(v) Using the same approach for obtaining the bound on the number of mistakes for $\beta = 0.5$, what happens to the bound as β approaches 1? You will need to rework the bound for a general user-specified β in this modified perceptron algorithm. Does this agree/disagree with your findings in part d?

Problem 2: Multi-class classification with SVMs [35 points]

In this problem, you will use binary SVMs to classify online newsgroup posts into 1 of 4 categories: AUTO, COMPUTERS, RELIGION, SPORTS. The two datafiles *groups2.train* and *groups2.test* have been slightly modified from HW2. All training examples are the same, except that all features were binarized, i.e., were set to 1 if their value was greater than zero. Again, each of the files contains 2000 examples with feature size 2000.

Since we have more than two classes, using a single binary SVM is not sufficient. Instead, we are going to combine several binary SVM classifiers to tackle this problem. Thus, our goal is to train multiple linear SVM classifiers predicting each class in isolation, and combine them to properly predict one of four classes.

We train linear SVM classifiers with SVM^{light} (http://svmlight.joachims.org). It provides you with both learning and predicting modules with instructions on the website. Please make sure that what you are using is SVM^{light} instead of SVM^{struct}, $SVM^{multiclass}$, or other variations.

To do multi-class classification with SVM^{light}, first train four different soft-margin linear classifiers. Note that you have to replace the target class number to 1 and all others to -1 before using the library. For instance, if you learn a classifier for RELIGION, you use 1,000 articles about politics as positive samples and 3,000 others as negative samples for training. Once you learned four classifiers, the output label of a test example $\mathbf{x}$ is determined choosing the classifier that yields the greatest margin:

$$h(\mathbf{x}) = \underset{y \in \{1,2,3,4\}}{\text{argmax}}\ \mathbf{w_y} \cdot \mathbf{x} + b_y$$

where $\mathbf{w_y}$ is the learned weight vector and b_y is the biased term for class y. Break ties by choosing the class with the smaller number. Note also, that `svm_classify` will output exactly these values for each test example, so you do not need to compute them by hand.

a) Randomly partition the training set into 5 equal-sized folds. Then, do 5-fold cross-validation for $C \in \{0.001, 0.01, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 100.0\}$. Use the same C-value and folds for each binary SVM. Make sure that the only parameter you use with `svm_learn` is `-c`.
If the prediction $h(\mathbf{x})$ is different from the ground-truth class, we count it as an error. Plot a graph showing both training and validation accuracies (averaged across the 5 folds) together with C in log-scale (i.e., x-axis: $\log_2 C$, y-axis: accuracy). Provide also a table showing these numbers. What value of C gives the best validation accuracy?
b) Now, re-train your four binary SVMs on the complete training set using the optimal C from a). Report the resulting training and test accuracies.
c) A standard preprocessing technique often used in practice is length normalization. There, each feature vector $\mathbf{x}$ gets rescaled so that $\|\mathbf{x}\|_2 = 1$. Now, normalize all training and test instances and repeat part a) and b) for the new data. Provide the same plots and tables.
d) Compare the test error from c) to previous test error from b) without normalization, and explain why normalization makes a difference (*Hint:* Think about what happens to the slack variables.). Why did we have to reestimate the optimal value for C? Relate your answers to the SVM objective.

Problem 3: Skewed datasets [30 points]

In practice, you will often have to deal with datasets that do not contain an equal number of positive and negative training examples. These *skewed* (or *unbalanced*) datasets can, for example, occur in clinical studies where patients with a rare form of cancer are the positive examples and patients with no abnormalities are the negative examples. In such situations, it is virtually impossible to obtain more positive examples. Another example is spam classification where users usually receive more ham than spam.

In this problem, you will investigate various problems with skewed datasets are and learn a way of mitigating them.

a) Suppose you know for a binary classification problem that $P(Y = 1) = 0.07$ in both the test and training set. Describe a simple rule that would give you at least 90% classification accuracy on the test set. What is the major problem with that rule?

b) You are given two synthetic data files *boxes.train* and *boxes.test*, for training and testing respectively. The training dataset contains 200 examples, 100 positive and 100 negative ones, the test dataset ten times as many samples. The positive examples stem from a uniform distribution $f(x, y)$ with $0 \leq x, y \leq 0.5$ and the negative ones from a uniform distribution $g(x, y)$ with $0.5 < x, y \leq 1.0$.
Let S_{nneg} denote the dataset containing all positive training examples but only the first $nneg$ negative training examples (in the order they appear in the file). For example, S_{30} would contain 130 examples in total, 100 positive examples and the first 30 negative ones.
Train a soft-margin linear classifier with SVM^{light} on S_{nneg} for all combinations of $nneg \in \{100, 50, 10\}$ and $C \in \{1, 1000\}$. Make sure that the only parameter you set for `svm_learn` is C (option `-c`). For each combination, plot the resulting dataset and the solution of the SVM, i.e. the optimal hyperplane. Use plus signs and circles as markers for positive and negative instances respectively.

c) Look at the plots of part b). How does the location of the optimal hyperplane change for $C = 1$ as we decrease the number of negative instances? How does it change for $C = 1000$? Provide an explanation for both observations. Relate your answers to the primal optimization objective of soft-margin SVMs.

d) One way of overcoming this issue is using custom weights in the optimization objective:

$$\min_{\mathbf{w},\mathbf{b},\xi} \quad \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C \cdot j \sum_{i:y_i=1}^{M} \xi_i + C \sum_{i:y_i=-1}^{M} \xi_i \tag{2}$$

$$\text{subject to} \quad \forall k : y_k(\mathbf{w} \cdot \mathbf{x}_k + \mathbf{b}) \geq 1 - \xi_k \quad , \xi_k \geq 0 \tag{3}$$

Suppose we set

$$j = \frac{\#\text{ negative training examples}}{\#\text{ positive training examples}}.$$

Assume that we always have at least one positive training example so that j is well-defined. Give a quantitative argument for why this choice is reasonable given no other information about the classes.

e) Run SVM^{light} on S_{10} using $C = 1$ and $j \in \{0.5, 0.1, 0.05\}$ and give plots (same as in b)) for each value of j.
How does the position of the hyperplane change when j gets decreased? Why does that happen? Relate your answer to the optimization objective in d).

f) Now run each of the three models of part e) on *boxes.test* and report overall accuracy, as well as the number of false positives and false negatives for each model.
How does the number of false negatives / false positives behave when j gets decreased? Relate your findings to the plots in e).