CSE 151: Introduction to Machine Learning Spring 2015

Problem Set 4

Instructor: Kamalika Chaudhuri **Due on:** May 15, 2015

Instructions

- This is a 40 point homework.
- Problem 3 is a programming assignment. For this problem, you are free to use any programming language you wish. Please submit a printout of your code along with your homework.

Problem 1: 8 points

Alice, Bob and Carol have all been asked to implement the perceptron algorithm. They all have the same training and test data, and they make a single pass over the training and test data with all the algorithms (Alice's variant, Bob's variant, and Carol's variant). Carol implements the version of perceptron that we discussed in lecture.

1. Suppose Alice implements the following variant of the perceptron algorithm.

   (a) Initially: $w_1 = 0$.

   (b) For $t = 1, 2, 3, 4, \ldots, T$

      i. If $y_t \langle w_t, x_t \rangle \leq 0$ then $w_{t+1} = w_t + y_t x_t$.

      ii. Otherwise: $w_{t+1} = w_t$.

   (c) Output $w_{Alice} = w_{T+1}/\|w_{T+1}\|$.

   Is the test error of the classifier output by Alice's algorithm the same as the test error of Carol's algorithm, no matter what the test data is? If your answer is yes, justify your answer. If your answer is no, provide a counterexample or a brief justification.

2. Bob implements a second variant of the perceptron algorithm, as follows.

   (a) Initially: $w_1 = 0$.

   (b) For $t = 1, 2, 3, 4, \ldots, T$

      i. If $y_t \langle w_t, x_t \rangle \leq 0$ then $w_{t+1} = \frac{w_t + y_t x_t}{\|w_t + y_t x_t\|}$.

      ii. Otherwise: $w_{t+1} = w_t$.

   (c) Output $w_{Bob} = w_{T+1}$.

   Is the test error of the classifier output by Bob's algorithm the same as the test error of Carol's algorithm, no matter what the test dataset is? If your answer is yes, provide a justification for your answer; if your answer is no, provide a counterexample or a brief justification.

Solution

1. Yes. Carol's and Alice's part (a) and (b) in the training algorithm are the same. The only difference is in part (c): Carol outputs

   $$w_{Carol} = w_{T+1},$$

   while Alice outputs

   $$w_{Alice} = w_{T+1}/||w_{T+1}||.$$

w_{Alice} and w_{Carol} give the same decision boundary, and thus the same test error. This is because $\forall x$,

$$\begin{aligned}\text{sign}(\langle w_{Carol}, x\rangle) &= \text{sign}(\langle w_{T+1}, x\rangle) \\ &= \text{sign}(\langle w_{T+1}/||w_{T+1}||, x\rangle) \\ &= \text{sign}(\langle w_{Alice}, x\rangle).\end{aligned}$$

2. No. Consider a training set with two points: $x_1 = (2,0)$, $y_1 = 1$; $x_2 = (0,2)$, $y_2 = -1$.
   In Bob's algorithm:

$$\begin{aligned}w_1 &= 0 \\ w_2 &= \frac{w_1 + y_1 x_1}{||w_1 + y_1 x_1||} = \frac{(2,0)}{||(2,0)||} = (1,0) \\ w_3 &= \frac{w_2 + y_2 x_2}{||w_2 + y_2 x_2||} = \frac{(1,-2)}{||(1,-2)||} = \frac{1}{\sqrt{5}}(1,-2) \\ w_{Bob} &= w_3 = \frac{1}{\sqrt{5}}(1,-2)\end{aligned}$$

   While in Carol's algorithm:

$$\begin{aligned}w_1 &= 0 \\ w_2 &= \frac{w_1 + y_1 x_1}{||w_1 + y_1 x_1||} = (2,0) \\ w_3 &= \frac{w_2 + y_2 x_2}{||w_2 + y_2 x_2||} = (2,-2) \\ w_{Carol} &= w_3 = (2,-2)\end{aligned}$$

   w_{Bob} and w_{Carol} give different decision boundaries, and thus test error may not be the same. For example, for test point $x = (\frac{3}{2}, 1)$, $\text{sign}(\langle w_{Bob}, x\rangle) = -1$, while $\text{sign}(\langle w_{Carol}, x\rangle) = 1$.

Problem 2: 12 points

Suppose we are given a training data set $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where each feature vector x_i lies in d-dimensional space. For each x_i, suppose we transform it to z_i by rescaling each axis of the data by a fixed factor; that is, for every $i = 1, \ldots, n$ and every coordinate $j = 1, \ldots, d$, we write:

$$z_i^j = \alpha^j x_i^j$$

Here α^js are real, non-zero and positive constants. Thus, our original training set S is transformed after rescaling to a new data set $S' = \{(z_1, y_1), \ldots, (z_n, y_n)\}$.

A classifier C in the original space is said to be equal to a classifier C' in the rescaled space if for every $x \in \mathbb{R}^d$, $C(x) = C'(z)$, where z is obtained by transforming x. For example, if $d = 1$, and if $\alpha^1 = 2$, the classifier C in the original space:

C: *Predict* 0 *if* $|x| \leq 1$ *and* 1 *otherwise*

is equal to the classifier C' in the rescaled space:

C': *Predict* 0 *if* $|z| \leq 2$ *and* 1 *otherwise*

1. First, suppose that all the α^i values are equal; that is, $\alpha^1 = \ldots = \alpha^d$. Suppose we train a k-NN classifier C on S and a k-NN classifier C' on S'. Are these two classifiers equal? What if we trained C and C' on S and S' respectively using the ID3 Decision Tree algorithm? What if we trained C and C' on S and S' respectively using the Perceptron algorithm? If the classifiers are equal, provide a *brief* argument to justify why; if they are not equal, provide a counterexample.

2. Repeat your answers to the questions in part (1) when the α_is are different. Provide a *brief* justification for each answer if the classifiers are equal, and a counterexample if they are not.

3. From the results of parts(1)and (2), what can you conclude about how k-NN, decision trees and perceptrons behave under scaling transformations?

Solution

For this question, we assume that ties are always broken in a consistent manner for both the k-NN and ID3 decision tree algorithms.

k-NN. We will obtain equal k-NN classifiers before and after a space transform for an arbitrary data set, if and only if, the following *relative distance* conditon holds: for any three points x, x_p and x_q in the original space, $d(x, x_p) \geq d(x, x_q)$ implies $d(z, z_p) \geq d(z, z_q)$, whre z, z_p and z_q are the points after rescaling. In other words, we need to ensure that in all cases, the nearest neighbors of a point in the original space are still the nearest neighbors in the rescaled space.

In the case of a uniform scaling factor (all $\alpha^j = \alpha$), the distance between any two points z_1 and z_2 in the rescaled space is,

$$d(z_1, z_2) = \sqrt{\sum_j (z_1^j - z_2^j)^2} = \sqrt{\sum_j (\alpha x_1^j - \alpha x_2^j)^2} = \alpha \sqrt{\sum_j (x_1^j - x_2^j)^2} = \alpha d(x_1, x_2),$$

This is simply the distance in the original space scaled by a constant α. Clearly the relative distance condition holds. In particular, this means that the training points that are the nearest neighbors of x in the original space remain the nearest neighbors of z in the rescaled space, therefore prediction for x remains the same as the prediction for z.

For nonuniform scaling factors, the relative distance condition does not necessarily hold. One extreme example is if $\alpha^1 = 1, \alpha^2 = 0.0001$ (a very small quantity). The transform now essentially projects each point to the x-axis (although the point will not be exactly on the axis). Consider the training points $(1, 0)$ with label 0 and $(0, 1)$ with label 1, and a test example $(0.1, 0)$. In the original space, $(0.1, 0)$ is closer to $(1, 0)$ than $(0, 1)$ and will be assigned label 0; in the rescaled space however, it will be rescaled to be $(0.1, 0)$, will be closer to $(0, 1)$ (now rescaled as $(0, 0.001)$), and thus will be assigned label 1 by the 1-NN classifier. Therefore in this case, we are not guaranteed to get the same k-NN classisifer.

ID3 Decision Tree. The decision trees produced by the ID3 algorithm will be equal in both cases, assuming that ties are broken in a consistent manner. We can show this by induction. In what follows, we will say that a splitting rule (j, t) in the original space *is equal to* to a splitting rule $(j, \alpha^j t)$ in the rescaled space.

We run the ID3 algorithm on S and S' simultaneously, and maintain the following invariants at each step of the algorithm. If T and T' are the trees built based on S and S' respectively, then, (a) T and T' have the same structure, (b) for each internal node v in T, the splitting rule at v is equal to the splitting rule at the corresponding internal node v' in T' and (c) if D is the dataset associated with a leaf node in T, then the dataset associated with the corresponding leaf node in T' is the rescaled version of points in D.

The invariant holds at the beginning of the algorithm, as the only (leaf) node is the root, which is associated with S in T and S' in T'. Suppose the invariant holds at step t of the algorithm, and at step $t + 1$ we split a node v in T such that the dataset associated with v is D. If the splitting rule used is (j, t), then, this splitting rule has the highest information gain among all the possible splitting rules. Observe that as the corresponding node v' in T' is associated with a scaled version D' of D, for any j and t, the information gain of a splitting rule $(j, \alpha^j t)$ at v' is equal to the information gain of the splitting rule (j, t) in v. Thus, assuming that ties are broken consistently, we will pick the splitting rule

$(j, \alpha^j t)$ to split node v'. Thus invariants (a) and (b) are maintained after step $t+1$. Finally, invariant (c) is also maintained as the subset of D for which feature j is $\leq t$ is exactly equal to the subset of D' for which feature j is $\leq \alpha^j t$.

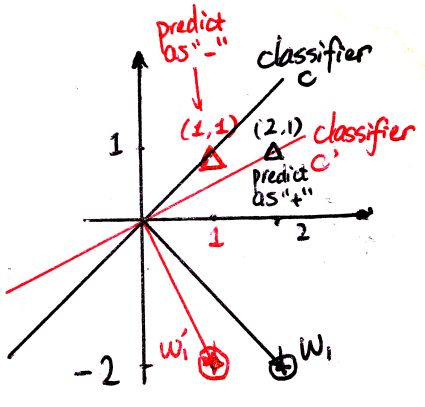
Thus, at the end of the ID3 decision tree algorithm, we arrive at two trees T and T' which have exactly the same structure, where the corresponding nodes v and v' have equal splitting rules. Thus if a test example x follows a path P in T from the root to the leaf, its rescaled version z will follow exactly the same path in T' from root to leaf and will be classified the same way. Therefore the two decision trees will be equal.

Perceptron. For a uniform scaling factor α, we claim that at any step, if the hyperplane normal in the original space is w, then the hyperplane normal in the rescaled space must be αw. If this claim is true, then the classfiers in the two spaces will be equal, because as $\alpha > 0$, $\text{sign}(\langle w, x_t \rangle) = \text{sign}(\langle \alpha w, z_t \rangle) = \text{sign}(\langle \alpha w, \alpha x_t \rangle)$.

We prove this by induction. The base case is trivial because w is initialized to 0 in both spaces. Then suppose our claim is true for step $t-1$, we show that the claim still holds at step t. At step t the algorithm predicts the label for the training data (x_t, y_t) in the original space and training data $(z_t = \alpha x_t, y_t)$ in the rescaled space. It is easy to see that the prediction result is the same for the classifiers in both spaces as $\alpha > 0$. If the result is correct, then no change is made to either w. If the result is wrong, the normal in the original space is updated to $w + y_t x_t$, while in the rescaled space, the normal is updated to $\alpha w + y_t z_t = \alpha(w + y_t x_t)$. Thus the claim still holds at this step. Therefore the Perceptron algorithm produces equal classifiers in both spaces.

For non-uniform α's, the two classifiers are not equal. One counter-example is given below. There is only one positive training data $(2, -2)$, which becomes $(1, -2)$ in the rescaled space. Consider the test data $(2, 1)$, and the rescaled version $(1, 1)$. The resulting classifier classifies them into different labels.

Behavior under scaling transformations. In case of uniform scaling transformations (same α^i) across all features/dimensions, all the 3 algorithms are equally robust. However, in case of non-uniform scaling transformations (different α^i), ID3 Decision Trees are more robust to compared to k-NN and Perceptrons.



Problem 3: Programming Assignment: 20 points

In this problem, we look at the task of classifying images of digits again, but this time we will use perceptron instead of k-nearest neighbor classification in Homework 2. Download the files `hw4atrain.txt` and `hw4atest.txt` from the class website. These files contain your training and test data sets respectively.

1. First, we will classify images of 0 vs. 6, which the k-nearest neighbor algorithm found difficult to tease apart in homework 2. For your benefit, we have already converted the images into vectors of pixel colors; each pixel color is a value between 0 and 255. The data files are in ASCII text format, and each line of the files contains a feature vector of 784 features, followed by its label (0 or 6). The coordinates of the feature vector are separated by spaces.

   Assume that the data is linearly separable by a hyperplane through the origin. Run one, two and three passes of perceptron, voted perceptron, and averaged perceptron on the training dataset to find classifiers that separate the two classes. What are the training errors and the test errors of perceptron, voted perceptron and averaged perceptron after one, two and three passes?

2. For the second part of the question, download `hw4btrain.txt` and `hw4btest.txt`. This will be your training and test data respectively.

   For each class $i = 0, \ldots, 9$, run a single pass of the perceptron algorithm on the training dataset to compute a linear classifier separating the training data points in class i from the training data points not in class i. Call this classifier C_i. We will now use these classifiers to construct a *one-vs-all* multiclass classifier.

   Given a test example x, the one-vs-all classifier predicts as follows. If $C_i(x) = i$ for exactly one $i = 0, \ldots, 9$, then predict label i. If $C_i(x) = i$ for more than one i in $0, \ldots, 9$, or if $C_i(x) = i$ for no i, then report *Don't Know*.

   Recall from Homework 2 that the confusion matrix is a 10×10 matrix, where each row is labelled $0, \ldots, 9$ and each column is labelled $0, \ldots, 9$. The entry of the matrix at row i and column j is C_{ij}/N_j where C_{ij} is the number of test examples that have label j but are classified as label i by the classifier, and N_j is the number of test examples that have label j. Since the one-vs-all classifier can also predict *Don't Know*, the confusion matrix will now be an 11×10 matrix – that is, it will have an extra row corresponding to the *Don't Know* predictions.

   Write down the confusion matrix for the one-vs-all classifier on the training data in `hw4btrain.txt` based on the test data in `hw4btest.txt`. Looking at this confusion matrix, and the solutions of Homework 2, what can you say about the performance of perceptron compared with the 3-nearest neighbor classifier on this dataset?

Solution

1. After one,two and three passes of the three algorithms each, we have:

	Training Error			Test Error		
	1 pass	2 passes	3 passes	1 pass	2 passes	3 passes
perceptron	0.01	0.008	0.008	0.02	0.01	0.014
voted perceptron	0.013	0.006	0.003	0.012	0.01	0.008
averaged perceptron	0.01	0.008	0.002	0.01	0.01	0.01

2. After a single pass of the perceptron algorithm, we have the confusion matrix:

prediction \ truth	0	1	2	3	4	5	6	7	8	9
0	**0.7778**	0	0	0	0	0	0.0094	0	0	0
1	0	0.2755	0.0098	0	0	0	0	0	0	0
2	0	0	**0.5784**	0.0187	0.0100	0	0.0094	0	0	0.0085
3	0	0	0.0098	0.4206	0	0.0108	0	0.0119	0	0
4	0	0	0.0098	0	**0.4500**	0.0108	0	0	0	0
5	0	0	0	0	0	0.3118	0	0	0	0
6	0	0	0	0	0	0.0108	**0.4528**	0	0	0
7	0	0	0	0	0.0100	0	0	0.3452	0	0
8	0.0202	0.0102	0.0588	0.0654	0.0300	0.1505	0.0943	0.0119	**0.8387**	0.0169
9	0	0	0	0	0.0600	0.0108	0.0094	0.0357	0.0108	**0.5847**
Unknown	0.202	**0.7143**	0.3333	**0.4953**	0.4400	**0.4946**	0.4245	**0.5952**	0.1505	0.3898

Compare this to the confusion matrix for 3-NN classifier from Homework 2:

prediction \ truth	0	1	2	3	4	5	6	7	8	9
0	**0.893**	0	0	0.033	0	0	0	0	0	0
1	0	**1**	0.053	0.1	0	0	0.054	0	0	0
2	0	0	**0.868**	0	0	0	0	0	0	0
3	0	0	0	**0.733**	0	0.038	0	0	0.036	0.037
4	0	0	0	0	**0.929**	0	0	0	0	0.037
5	0	0	0	0.033	0	**0.846**	0	0	0	0
6	0.107	0	0	0	0	0	**0.946**	0	0.036	0
7	0	0	0.053	0	0	0.038	0	**0.933**	0	0
8	0	0	0.026	0.067	0	0.038	0	0	**0.929**	0
9	0	0	0	0.033	0.071	0.038	0	0.067	0	**0.926**

The perceptron classifier in this case performs worse than the 3-NN classifier.