10-601 Machine Learning: Homework 7

Due 5 p.m. Wednesday, April 22, 2015

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that. You *must* turn in at least $n-1$ of the n homeworks to pass the class, even if for zero credit.
- **Collaboration policy:** Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online on autolab. We recommend that you use LaTeX to type your solutions to the written questions, but we will accept scanned solutions as well. On the Homework 7 autolab page, you can download the template, which is a tar archive containing a blank placeholder pdf for the written questions. Replace each pdf file with one that contains your solutions to the written questions. When you are ready to submit, create a new tar archive of the top-level directory and submit your archived solutions online by clicking the "Submit File" button. You should submit a single tar archive identical to the template, except with the blank pdfs replaced by your solutions for the written questions. You are free to submit as many times as you like. ***DO NOT*** change the name of any of the files or folders in the submission template. In other words, your submitted files should have exactly the same names as those in the submission template. Do not modify the directory structure.

Problem 1: k-means Clustering [40 pt + 10 Extra Credit]

Recall that in $k-$means clustering we attempt to find k cluster centers $c_j \in \mathbb{R}^d, j \in \{1, \ldots, k\}$ such that the total distance between each datapoint and the nearest cluster center is minimized. In other words, we attempt to find $c_1, \ldots, c_k$ that minimizes

$$\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|x_i - c_j\|^2, \tag{1}$$

where n is the number of data points. To do so, we iterate between assigning x_i to the nearest cluster center and updating each cluster center c_j to the average of all points assigned to the j^{th} cluster.

Choosing k is no easy matter

1. [10 pt] Instead of holding the number of clusters k fixed, one can think of minimizing (1) over both k and c. Show that this is a bad idea. Specifically, what is the minimum possible value of (1) ? what values of k and c result in this value ?

k-means can be kernelized too !

k-means with Euclidean distance metric assumes that each pair of clusters is linearly separable. This may not be the case. A classical example is where we have two clusters corresponding to data points on two concentric circles in the $\mathbb{R}^2$ plane. We have seen that we can use kernels to obtain a non-linear version of an algorithm that is linear by nature and k-means is no exception. Recall that there are two main aspects of kernelized algorithms: (i) the solution is expressed as a linear combination of training examples, (ii) the algorithm relies only on inner products between data points rather than their explicit representation. We will show that these two aspects can be satisfied in k-means.

2. [5 pt] Let z_{ij} be an indicator that is equal to 1 if the x_i is currently assigned to the j^{th} cluster and 0 otherwise ($1 \leq i \leq n$ and $1 \leq j \leq k$). Show that the j^{th} cluster center c_j can be updated as $\sum_{i=1}^{n} \alpha_{ij} x_i$. Specifically, show how α_{ij} can be computed given all z's.

3. [5 pt] Given two data points x_1 and x_2, show that the square distance $\|x_1 - x_2\|^2$ can be computed using only (linear combinations of) inner products.

4. [5 pt] Given the results of parts 2 and 3, show how to compute the square distance $\|x_i - c_j\|^2$ using only (linear combinations of) inner products between the data points $x_1, \ldots, x_n$.

   Note: This means that given a kernel K, we can run Lloyd's algorithm. We begin with some initial data points as centers and use the answer to part 2 to find the closest center for each data point, giving us the initial z_{ij}'s. We then repeatedly use the answer to part 3 to reassign the points to centers and update the z_{ij}'s.

k-means for single dimensional data

In general, minimizing (1) for a fixed k is an NP-hard problem. However it can be solved in polynomial time if the datapoints are single dimensional ($d = 1$). For the remaining of this question, we will focus on that case.

5. [2 pt] Consider the case where $k = 3$ and we have 4 data points $x_1 = 1, x_2 = 2, x_3 = 5, x_4 = 7$. What is the optimal clustering for this data ? What is the corresponding value of the objective (1) ?

6. [3 pt] One might be tempted to think that Lloyd's algorithm is guaranteed to converge to the global minimum when $d = 1$. Show that there exists a suboptimal cluster assignment for the data in part 5 that Lloyd's algorithm will not be able to improve (to get full credit, you need to show the assignment, show why it is suboptimal *and* explain why it will not be improved).

7. [10 pt] Assume we sort our data points such that $x_1 \leq x_2 \leq \cdots \leq x_n$, prove that an optimal cluster assignment has the property that each cluster corresponds to some interval of points. That is, for each cluster j there exists i_1, i_2 such that the cluster consists of $\{x_{i_1}, x_{i_1+1}, \ldots, x_{i_2}\}$.

8. **(Extra Credit [10 pt])** Develop an $O(kn^2)$ dynamic programming algorithm for single dimensional k-means. [**Hint**: From part 7, what we need to optimize are $k - 1$ cluster boundaries where the i^{th} boundary marks the largest data point in the i^{th} cluster.]

Problem 2: Dimensionality Reduction and Representation Learning [30 pt]

In this question, we explore the relation between PCA, kernel PCA and auto encoder neural networks (trained to output the same vector they receive as input). We will use n and d to denote the number and dimensionality of the given data points respectively.

1. [10 pt] Consider an auto encoder with a single hidden layer of k nodes. Let w_{ij} denote the weight of the edge from the i^{th} input node to the j^{th} hidden node. Similarly, let v_{ij} denote the weight of the edge from the i^{th} hidden node to the j^{th} output node. Show how you can set the activation functions of hidden and output nodes as well as the weights w_{ij} and v_{ij} such that the resulting auto encoder resembles PCA.

2. [10 pt] Kernel PCA is a non-linear dimensionality reduction reduction where a principal vector u_j is computed as a linear combination of training examples in the feature space

$$u_j = \sum_{i=1}^{n} \alpha_{ij} \phi(x_i).$$

   Computing the principal component of a new point x can then be done using kernel evaluations

$$z_j(x) = \langle u_j, \phi(x) \rangle = \sum_{i=1}^{n} \alpha_{ij} \langle \phi(x_i), \phi(x) \rangle = \sum_{i=1}^{n} \alpha_{ij} k(x_i, x).$$

   You will show that kernel PCA can be represented by a neural network. First we define a *kernel node*. A kernel node with a vector w_i of incoming weights and an input vector x computes the output $y = k(x, w_i)$. Show that, given a data set $x_1, \ldots, x_n$, there exists a network with a single hidden layer and the output of the network is the kernel principal components $z_1(x), \ldots, z_k(x)$ for a given input x. Specify the number of nodes in the input, output and hidden layers, the type and activation function of hidden and output nodes , and the weights of the edges in terms of $\alpha, x_1, \ldots, x_n$.

3. [5 pt] What is the number of parameters (weights) required to store the network in part 2?

4. [5 pt] Another way to do non-linear dimensionality reduction is to train an auto encoder with non-linear activation functions (e.g. sigmoid) in the hidden layers. State one advantage and one disadvantage of that approach compared to kernel PCA.

Problem 3: Co-training Doesn't Like Groupthink [30 pt]

Consider the data set in figure 1. Each data point has two features. Circled data points are unlabeled points where the true label (shown inside the circle) is invisible to the learning algorithm. We will use this dataset to co-train two threshold based classifiers C_1 and C_2 where C_i is trained using feature i and produces a decision threshold on feature i that maximizes the margin between training examples and the threshold. The "confidence" of a classifier for a new data points is measured by how far away it is from the threshold. In particular, the farther away the point is from the decision boundary, the more confident the classifier is. We will run iterative co-training such that, in each iteration, each classifier adds the unlabeled example it is most confident about to the training data. Assume that co-training halts when, for each classifier, the unlabeled point that is farthest from the threshold (i.e. most confident) is between the largest known negative example and the smallest known positive example (at that point, the algorithm deems the unlabeled examples too uncertain to label for the other classifier).

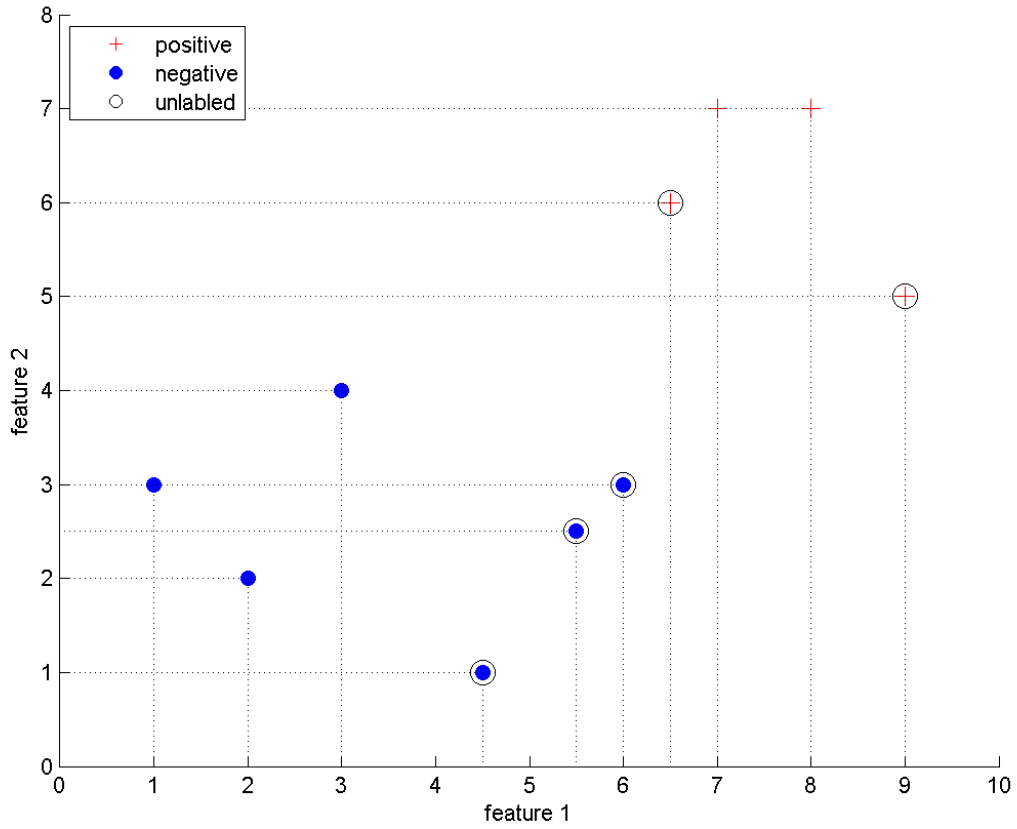

Figure 1: Data set for problem 3.

1. [10 pt] Explain what happens in a single iteration of co-training. Specifically, illustrate:
   - The initial thresholds produced by C_1 and C_2 given labeled examples.
   - The new labeled example (coordinates and label) that will be provided to C_2 by C_1 and vice versa.
   - The new thresholds after incorporating the new examples.
   - The number of data points misclassified by C_1 using the initial and updated thresholds.

2. [15 pt] Now assume that we train both C_1 and C_2 using feature 1. Therefore they share the same view of the data. What happens if we run co-training to completion ? What are the initial thresholds, which unlabeled example will be added in each iteration at what are the final thresholds ?

3. [5 pt] Based on your observations of parts 1 and 2, provide an intuitive explanation (in no more than two lines) for why having features that satisfy independence given the label helps co-training to be successful.

Extra Credit Problem: From Active Heaven to Adversarial Hell [10 pt]

In this problem we will explore how the rate by which an algorithm learns a concept can drastically change based on how labeled examples are obtained. For that we look at three settings: (i) an active learning setting where the algorithm has the luxury of specifying a data point and querying its label, (ii) a passive learning setting where labeled examples are drawn at random and (iii) an adversarial setting where training examples are given by an adversary that tries to make your life hard.

Consider a binary classification problem where each data point consists of d binary features. Let H be the hypothesis class of conjunctions of subsets of the d features and their negations. So for example one hypothesis could be $h_1(x) = x_1 \wedge x_2 \wedge \neg x_d$ (where $\wedge$ denotes the logical "and" and $\neg$ denotes the logical "not"). Another hypothesis could be $h_2(x) = \neg x_3 \wedge x_5$. A third hypothesis could be $h_3(x) = 1$ (a conjunction of zero features). A conjunction in H *cannot* contain both x_i and $\neg x_i$. We assume a consistent learning scenario where there exists a hypothesis $h^* \in H$ that is consistent with the true label for all data points.

1. [4 pt] In the active learning setting, the learning algorithm can query the label of an unlabeled example. Assume that you can query any possible example. Show that, starting with a single positive example, you can exactly learn the true hypothesis h^* using d queries.

2. [3 pt] In the passive learning setting, the examples are drawn i.i.d from an unknown distribution. According to PAC learning thoery, how many examples (in big-O notation) are required to guarantee a generalization error less than ϵ with probability $1 - \delta$? (**Hint**: the VC dimension of the class of conjunctions of d binary features is d)

   Note: The result of part 1 is much stronger that that of part 2; it guarantees that the classifier will exactly learn the true hypothesis with probability 1. PAC learning guarantees, on the other hand, would require an infinite number of examples as the error ϵ and probability of failure δ got to 0. In other words, it is "surely exactly correct" compared to "probably approximately correct".

3. [3 pt] Show that if the training data is not representative of the underlying distribution, a consistent hypothesis can perform poorly. Specifically, assume that the true hypothesis h^* is a conjunction of k out of the d features for some $k > 0$ and that all possible data points are equally likely. Show that there exists a training set of $2^{(d-k)}$ unique examples and a hypothesis $\hat{h}$ that is consistent with this training set but achieves a classification error $\geq 50\%$ when tested on all possible data points.

   Note: The result of part 3 does not contradict that of part 2; the adversarial unrepresentative sample given in part 3 could still occur with random i.i.d sampling. The probability of having such unrepresentative training sets is included in the failure probability δ.