10-715 Advanced Introduction to Machine Learning: Homework 5
Semi-Supervised Learning, Clustering, Active Learning

Released: Thursday, November 8, 2018

Due: 11:59 p.m. Wednesday, November 21, 2018

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.
- **Collaboration policy:** Collaboration on solving the homework is allowed. Discussions are encouraged but you should think about the problems on your own. When you do collaborate, you should list your collaborators! Also cite your resources, in case you got some inspiration from other resources (books, websites, papers). If you do collaborate with someone or use a book or website, you are expected to write up your solution independently. That is, close the book and all of your notes before starting to write up your solution. 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 (link: `https://autolab.andrew.cmu.edu/courses/10715-f18/assessments`). Please use LaTeX to typeset your solutions, and submit a single pdf called **hw5.pdf**.
- This homework involves a significant amount of coding, and requires a number of experiments to be run. **Start Early!**

1 Semi-Supervised Learning [50 points + 50 points EXTRA CREDIT]

1.1 Basic Framework [50 points]

This problem will investigate a famous semi-supervised learning algorithm [1], which is based on the assumption that similar datapoints should have similar labels. You are encouraged to read through the paper for the motivation and further details.

We suppose there are labeled points $(x_1, y_1), ..., (x_l, y_l)$, and u unlabeled points $x_{l+1}, ..., x_{l+u}$; typically $l << u$. Let $n = l + u$ be the total number of data points. To begin, we assume the labels are binary: $y \in \{0, 1\}$. Consider a connected graph $G = (V, E)$ with nodes V corresponding to the data points, with nodes $L = \{1, ..., l\}$ corresponding to the labeled points with labels $y_1, ..., y_l$, and nodes $U = \{l + 1, ..., l + u\}$ corresponding to the unlabeled points. Our task is to assign labels to nodes U. We assume an $n \times n$ symmetric weight matrix W on the edges of the graph is given. For example, when $x \in \mathbb{R}^m$, the weight matrix can be

$$w_{ij} = \exp\left(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}\right) \tag{1}$$

where x_{id} is the d-th component of instance x_i represented as a vector $x_i \in \mathbb{R}^m$, and $\sigma_1, ...\sigma_m$ are length scale hyperparameters for each dimension. Thus, nearby points in Euclidean space are assigned large edge weights.data

Our strategy is to first compute a real-valued function $f : V \to \mathbb{R}$ on G with certain nice properties, and to then assign labels based on f. Let f be the $n \times 1$ vector of predictions $f = [f(i)]_i$, $i = 1, ..., n$. We constrain f to take values $f(i) = f_l(i) = y_i$ on the labeled data $i = 1, ..., l$. Intuitively, we want unlabeled points that are nearby in the graph to have similar labels. This motivates the choice of the quadratic energy function

$$E(f) = \frac{1}{2}\sum_{i,j} w_{ij}(f(i) - f(j))^2 \tag{2}$$

We form a Gaussian field $p_\beta(f) = \frac{\exp(-\beta E(f))}{Z_\beta}$ where β is an "inverse temperature" parameter, and Z_β is the partition function $Z_\beta = \int_{f|_L = f_l} \exp(-\beta E(f))\, df$, which normalizes over all functions constrained to f_l on the labeled data.

Problem 1.1.1 [10 points] Show that the minimum energy $f = \operatorname{argmin}_{f|_L = f_l} E(f)$ is harmonic, namely, it satisfies $\Delta f = 0$ on unlabeled points U, and is equal to f_l on the labeled data points L. Here Δ is the combinatorial Laplacian, given in matrix form as $\Delta = D - W$ where $D = \text{diag}(d_i)$ is the diagonal matrix with entries $d_i = \sum_j w_{ij}$ and $W = [w_{ij}]_{ij}$ is the weight matrix.

Note: By Δf, we mean the $n \times n$ matrix Δ multiplied by the $n \times 1$ vector of predictions $f = [f(i)]_i$, $i = 1, ..., n$.

Problem 1.1.2 [10 points] Show that the harmonic property means that the value of f at each unlabeled data point is the average of f at neighboring points:

$$f(j) = \frac{1}{d_j}\sum_{i \sim j} w_{ij} f(i), \text{for } j = l + 1, ..., l + u \tag{3}$$

which is consistent with our prior notion of smoothness of f with respect to the graph.

Observe that we can also express f as $f = Pf$ where $P = D^{-1}W$.

To compute the harmonic solution explicitly in terms of matrix operations, we split the weight matrix W (and similarly D, P) into 4 blocks after the lth row and column:

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix} \tag{4}$$

Problem 1.1.3 [10 points] Let $f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}$ where f_u denotes the values on the unlabeled data points. Show that the harmonic solution $\Delta f = 0$ subject to $f|_L = f_l$ is given by

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l \tag{5}$$

The learned function f_u can now be used to predict the label of the unlabeled training datapoints. We predict 1 if $f_u > 0.5$ and 0 otherwise.

Problem 1.1.4 [15 points] We provide a subset of the MNIST dataset as the data. There are 600 labeled images of which there are 300 images for digit 5 and 300 images for digit 6. All images for 5 have label 0 and all images for 6 have label 1. We use the same σ_d for all dimensions and we find that $\sigma = 1.5$ works well in practice.

Complete the get_prediction function and submit it to Autolab for evaluation. The inputs of the functions include X, Y, σ, l. The output of the function should be the predicted label on the unlabeled data. We will sample a subset of the provided data and compare the results of your implementation with ours.

We use Python 2.7.5 for autograding. To make sure that your code works on Autolab, please use the following command to install the correct version of numpy:

```
pip install 'numpy==1.7.1'
```

Submission Instructions: Run the following command to submit your code.

```
$ tar cvf src.tar src
```

Then submit your tarfile. Note that the autograding will take several minutes to complete. Hence we recommend you to debug your code offline and see if it gets an reasonable accuracy for Problem 1.1.5 before submitting it.

Debugging Suggestions: A sanity check is to verify whether your model can overfit the labeled data. To verify this, you can remove all unlabeled samples and instead copy the labeled samples as unlabeled samples. In other words, all unlabeled samples will appear in the labeled set. Set σ to a small value so that the matrix W only contains 0 or 1 and see if your code achieves an 100% accuracy.

Problem 1.1.5 [5 points] Randomly sample l_k labeled images from each class $k \in \{0, 1\}$. This results in a total of $l = 2 \times l_k$ labeled digits and $u = 600 - l$ unlabeled digits in the training set.

Please run the semi-supervised learning basic framework with the following choices of $l_k \in \{3, 10, 50\}$ and compute the performance of digit classification on the unlabeled samples. Repeat the experiment five times, each time randomly sampling a different subset of the labeled digits to reduce variance of the estimates and obtain an average accuracy on the unlabeled set. You should be able to get an accuracy of around 97% when you set l_k to 10.

Plot a graph of average accuracy on the unlabeled set against the value of l_k and explain your observations.

1.2 Do Something Extra [50 points EXTRA CREDIT]

This section is reserved for students who wish to further investigate this algorithm for semi-supervised learning. This section is entirely optional and will count for 50 points extra credit. You are required to suggest extensions to the algorithm and evaluate their effectiveness. In more detail:

1. Clearly define and motivate the extension you are trying to pursue. What problem exists in the current algorithm, and how does your proposed extension solve it?
2. Implement the extension and clearly include the implementation details. Describe all relevant details including datasets, distance metrics, loss function, optimizer, classifier choices, etc.
3. Write a few paragraphs explaining the experimental results and discuss the effectiveness of the implemented method. If it works well, give some intuitions (or even better, proofs) of why it improves performance. If it does not work well, that's fine too. Give some explanations for the performance and some possible fixes.

The grading for this section will be determined by the difficulty of the selected task, accuracy of implementation, and quality of reporting of experimental results. Here are some suggested ideas:

1. Section 4 of [1]: Incorporating Class Prior Knowledge.
2. Section 5 of [1]: Incorporating External Classifiers.
3. Section 6 of [1]: Learning the Weight Matrix W.

4. Pick more challenging datasets (e.g. time series data, text, images video, multimodal data etc.) and test out the algorithm with your choices of distance metrics. Explain why the chosen distance metric makes sense.
5. Use the intermediate representations learned from neural networks to compute distance metrics. Explore this when the neural networks are pretrained, or when the networks are learned in an end-to-end manner. Does this improve over distance metrics on raw features?
6. Extend [1] to work for multiclass classification.

Of course you are also encouraged to propose your own extensions. Good luck!

2 k-means Clustering [35 points + 5 points EXTRA CREDIT]

Recall that given a set $S = \{x_1, ..., x_n\} \subseteq \mathbb{R}^d$ of n points in d-dimensional space, the goal of k-means clustering is to find a set of centers $c_1, ..., c_k \in \mathbb{R}^d$ that minimize the k-means objective:

$$\sum_{i=1}^{n} \min_{j \in \{1,...,k\}} \|x_i - c_j\|_2^2 \tag{6}$$

which measures the sum of squared distances from each point x_j to its nearest center. 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.

2.1 Choosing k [5 points]

Instead of holding the number of clusters k fixed, one can think of minimizing (6) over both k and c. Show that this is a bad idea. Specifically, what is the minimum possible value of (6)? what values of k and c result in this value?

2.2 Kernalized k-means [15 points]

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.

1. **[5 points]** Let z_{ij} be an indicator that is equal to 1 if 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.
2. **[5 points]** Given two data points x_1 and x_2, show that the square distance $\|x_1 - x_2\|_2^2$ can be computed using only (linear combinations of) inner products.
3. **[5 points]** Given the results of parts 1 and 2, show how to compute the square distance $\|x_i - c_j\|_2^2$ using only (linear combinations of) inner products between the data points $x_1, ..., x_n$.

   Note: This means that given a kernel K, we can run Lloyds algorithm. We begin with some initial data points as centers and use the answer to part 1 to find the closest center for each data point, giving us the initial z_{ij}'s. We then repeatedly use the answer to part 2 to reassign the points to centers and update the z_{ij}'s.

2.3 k-means for Single Dimensional Data [15 points + 5 points EXTRA CREDIT]

In this problem you will show that the k-means objective (6) can be minimized in polynomial time when the data points are single dimensional $(d = 1)$, despite the fact that in general finding the optimal centers is NP-hard.

1. **[3 points]** Consider the case where $k = 3$ and we have 4 data points $x_1 = 1, x_2 = 3, x_3 = 6, x_4 = 7$. What is the optimal clustering for this data? What is the corresponding value of the objective (6).
2. **[5 points]** One might be tempted to think that Lloyds method is guaranteed to converge to the global minimum when $d = 1$. Show that there exists a suboptimal cluster assignment for the data in part (1) that Lloyds 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).

3. **[7 points]** Assume we sort our data points such that $x_1 \leq x_2 \leq \ \leq x_n$. 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 exits i_1 and i_2 such that the cluster consists of $\{x_{i_1}, x_{i_{1+1}}, ..., x_{i_2}\}$.

4. **[5 points EXTRA CREDIT]** Develop an $O(kn^2)$ dynamic programming algorithm for single dimensional k-means. (Hint: from part (3), what we need to optimize are $k-1$ cluster boundaries where the ith boundary marks the largest point in the ith cluster.)

3 Disagreement-based Active Learning [15 points]

3.1 True or False [6 points]

Answer the following questions. If your answer is True, provide 1-2 sentences to justify your answer. If your answer is False, give a counter-example if possible. Recall from the lecture notes that a classification problem is realizable if the true labeling function h^* is in our hypothesis class H. Given a set of labeled examples $\{(x_t, y_t)\}_{t=1}^n$, where $y_t = h^*(x_t), \forall t = 1, ..., n$, the version space $H' \subseteq H$ is a subset of H in which the hypotheses are consistent with the labels seen so far. The region of disagreement with respect to a version space H', denoted as DIS(H'), is defined to be the part of data space about which there is still some uncertainty. More specifically, DIS$(H') = \{x \in X : \exists h_1, h_2 \in H', h_1(x) \neq h_2(x)\}$.

1. **[2 points]** Assume the problem is realizable. Given a concept class H and a sequence of training instances $\{(x_t, y_t)\}_{t=1}^n$. Let $H_t \subseteq H$ be the version space after the learner has received the t-th instance. Then $H_t \neq \varnothing, \forall t = 1, ..., n$.
2. **[2 points]** Assume the problem is realizable. Given a concept class H and a sequence of training instances $\{(x_t, y_t)\}_{t=1}^n$. Let $H_t \subseteq H$ be the version space after the learner has received the t-th instance. After all n examples have been observed we have $|H_n| = 1$.
3. **[2 points]** Let $X = \{(x_t, y_t)\}_{t=1}^n$ be a set of n training instances and H be the hypothesis space. Assume our current version space is H'. Pick $x_0 \in$ DIS(H'). Let $\hat{H}$ be the version space of X after querying x_0, i.e., the version space of $X \cup \{(x_0, y_0)\}$, where y_0 is provided by the true labeling function, then it must be that $\hat{H} \neq H'$.

3.2 An Example [9 points]

Consider a binary classification problem where the sample space is $\Omega = \{(\mathbf{x}, y) : \|\mathbf{x}\|_2 \leq 2, y \in \{+1, -1\}\}$. Let the concept class $H = \{\mathbf{w} : \mathbf{w} \in \mathbb{R}^2, \|\mathbf{w}\|_2 = 1\}$, i.e., the set of linear classifiers where the decision rule is given by:

$$h_{\mathbf{w}}(\mathbf{x}) = \begin{cases} +1 & \mathbf{w}^\top \mathbf{x} \geq 0 \\ -1 & \mathbf{w}^\top \mathbf{x} < 0 \end{cases} \tag{7}$$

In this problem we assume that H is realizable. Suppose after receiving a sequence of training instances we know that the entire sector AOB shown in Figure 1 is $+1$ and the entire sector COD is -1. Both sectors AOB and COD are symmetric with respect to the x_1-axis and $\angle AOB = \angle COD = \frac{\pi}{3}$.

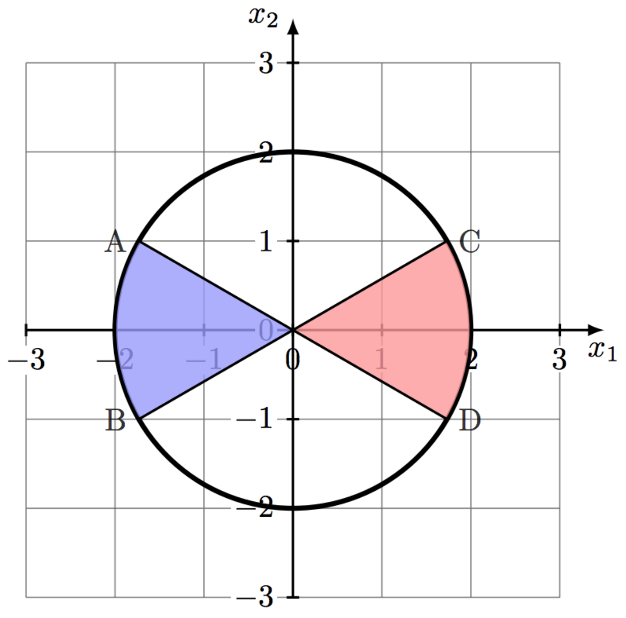
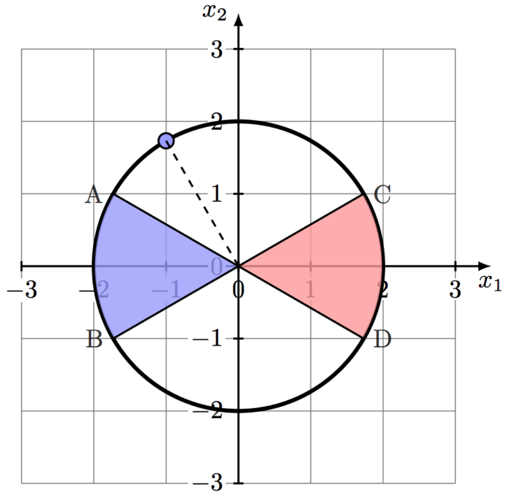
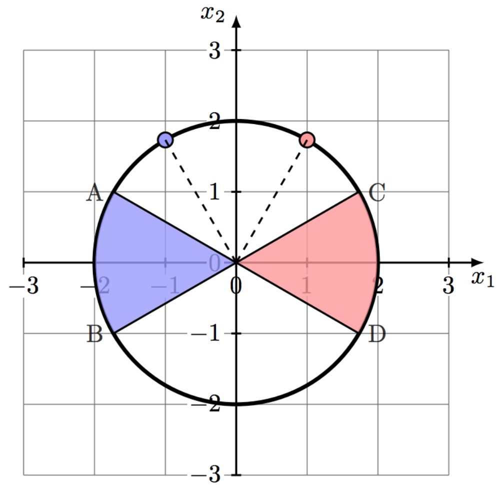


Figure 1: All hypotheses in the version space label the blue region as $+1$ and the red region as -1.

1. **[3 points]** Explicitly write down the version space $H_1 \subseteq H$ that is consistent with all the instances in sectors AOB and COD and draw H_1 in Figure 1. Specifically, in Figure 1, shade in the region that corresponds to the set of vectors $\mathbf{w}$ such that $h_{\mathbf{w}} \in H_1$.
2. **[3 points]** Suppose we receive a new instance $(\mathbf{x}, y) = ((-1, \sqrt{3}), +1)$. Write down the new version space $H_2 \subseteq H$ that is consistent with all the instances in both sectors as well as the new instance. Draw H_2 in Figure 2a.
3. **[3 points]** This time another new instance with negative label $(\mathbf{x}, y) = ((1, \sqrt{3}), -1)$ has been received. Write down the new version space $H_3 \subseteq H$ that is consistent with all the instances in both sectors along with the two new instances. Draw H_3 in Figure 2b.



(a) A new instance $(\mathbf{x}, y) = ((-1, \sqrt{3}), +1)$ is received



(b) Another new instance $(\mathbf{x}, y) = ((1, \sqrt{3}), -1)$ is received

Figure 2: New instances are received.

4 From Active Heaven to Adversarial Hell [15 points EXTRA CREDIT]

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 logical 'and' and $\neg$ denotes 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. **[5 points EXTRA CREDIT]** 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. **[5 points EXTRA CREDIT]** In the passive learning setting, the examples are drawn i.i.d from an unknown distribution. According to PAC learning theory, how many examples (in big-O) 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).

3. **[5 points EXTRA CREDIT]** 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.

References

[1] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919. AAAI Press, 2003.