CS 229, Public Course
Problem Set #3: Learning Theory and Unsupervised Learning

---

1. **Uniform convergence and Model Selection**

In this problem, we will prove a bound on the error of a simple model selection procedure.

Let there be a binary classification problem with labels $y \in \{0, 1\}$, and let $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \ldots \subseteq \mathcal{H}_k$ be k different finite hypothesis classes ($|\mathcal{H}_i| < \infty$). Given a dataset S of m iid training examples, we will divide it into a training set S_{train} consisting of the first $(1-\beta)m$ examples, and a hold-out cross validation set S_{cv} consisting of the remaining βm examples. Here, $\beta \in (0, 1)$.

Let $\hat{h}_i = \arg\min_{h \in \mathcal{H}_i} \hat{\varepsilon}_{S_{\text{train}}}(h)$ be the hypothesis in $\mathcal{H}_i$ with the lowest *training* error (on S_{train}). Thus, $\hat{h}_i$ would be the hypothesis returned by training (with empirical risk minimization) using hypothesis class $\mathcal{H}_i$ and dataset S_{train}. Also let $h_i^\star = \arg\min_{h \in \mathcal{H}_i} \varepsilon(h)$ be the hypothesis in $\mathcal{H}_i$ with the lowest *generalization* error.

Suppose that our algorithm first finds all the $\hat{h}_i$'s using empirical risk minimization then uses the hold-out cross validation set to select a hypothesis from this the $\{\hat{h}_1, \ldots, \hat{h}_k\}$ with minimum training error. That is, the algorithm will output

$$\hat{h} = \arg \min_{h \in \{\hat{h}_1, \ldots, \hat{h}_k\}} \hat{\varepsilon}_{S_{\text{cv}}}(h).$$

For this question you will prove the following bound. Let any $\delta > 0$ be fixed. Then with probability at least $1 - \delta$, we have that

$$\varepsilon(\hat{h}) \leq \min_{i=1,\ldots,k} \left(\varepsilon(h_i^*) + \sqrt{\frac{2}{(1-\beta)m} \log \frac{4|\mathcal{H}_i|}{\delta}} \right) + \sqrt{\frac{2}{2\beta m} \log \frac{4k}{\delta}}$$

(a) Prove that with probability at least $1 - \frac{\delta}{2}$, for all $\hat{h}_i$,

$$|\varepsilon(\hat{h}_i) - \hat{\varepsilon}_{S_{\text{cv}}}(\hat{h}_i)| \leq \sqrt{\frac{1}{2\beta m} \log \frac{4k}{\delta}}.$$

(b) Use part (a) to show that with probability $1 - \frac{\delta}{2}$,

$$\varepsilon(\hat{h}) \leq \min_{i=1,\ldots,k} \varepsilon(\hat{h}_i) + \sqrt{\frac{2}{\beta m} \log \frac{4k}{\delta}}.$$

(c) Let $j = \arg\min_i \varepsilon(\hat{h}_i)$. We know from class that for $\mathcal{H}_j$, with probability $1 - \frac{\delta}{2}$

$$|\varepsilon(\hat{h}_j) - \hat{\varepsilon}_{S_{\text{train}}}(h_j^\star)| \leq \sqrt{\frac{2}{(1-\beta)m} \log \frac{4|\mathcal{H}_j|}{\delta}}, \quad \forall h_j \in \mathcal{H}_j.$$

Use this to prove the final bound given at the beginning of this problem.

~~2~~. VC Dimension

Let the input domain of a learning problem be $\mathcal{X} = \mathbb{R}$. Give the VC dimension for each of the following classes of hypotheses. In each case, if you claim that the VC dimension is d, then you need to show that the hypothesis class can shatter d points, and explain why there are no $d+1$ points it can shatter.

- $h(x) = \mathbf{1}\{a < x\}$, with parameter $a \in \mathbb{R}$.
- $h(x) = \mathbf{1}\{a < x < b\}$, with parameters $a, b \in \mathbb{R}$.
- $h(x) = \mathbf{1}\{a \sin x > 0\}$, with parameter $a \in \mathbb{R}$.
- $h(x) = \mathbf{1}\{\sin(x + a) > 0\}$, with parameter $a \in \mathbb{R}$.

~~3~~. ℓ_1 regularization for least squares

In the previous problem set, we looked at the least squares problem where the objective function is augmented with an additional regularization term $\lambda\|\theta\|_2^2$. In this problem we'll consider a similar regularized objective but this time with a penalty on the ℓ_1 norm of the parameters $\lambda\|\theta\|_1$, where $\|\theta\|_1$ is defined as $\sum_i |\theta_i|$. That is, we want to minimize the objective

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(\theta^T x^{(i)} - y^{(i)})^2 + \lambda\sum_{i=1}^{n}|\theta_i|.$$

There has been a great deal of recent interest in ℓ_1 regularization, which, as we will see, has the benefit of outputting sparse solutions (i.e., many components of the resulting θ are equal to zero).

The ℓ_1 regularized least squares problem is more difficult than the unregularized or ℓ_2 regularized case, because the ℓ_1 term is not differentiable. However, there have been many efficient algorithms developed for this problem that work very well in practive. One very straightforward approach, which we have already seen in class, is the coordinate descent method. In this problem you'll derive and implement a coordinate descent algorithm for ℓ_1 regularized least squares, and apply it to test data.

(a) Here we'll derive the coordinate descent update for a given θ_i. Given the X and $\vec{y}$ matrices, as defined in the class notes, as well a parameter vector θ, how can we adjust θ_i so as to minimize the optimization objective? To answer this question, we'll rewrite the optimization objective above as

$$J(\theta) = \frac{1}{2}\|X\theta - \vec{y}\|_2^2 + \lambda\|\theta\|_1 = \frac{1}{2}\|X\bar{\theta} + X_i\theta_i - \vec{y}\|_2^2 + \lambda\|\bar{\theta}\|_1 + \lambda|\theta_i|$$

where $X_i \in \mathbb{R}^m$ denotes the ith column of X, and $\bar{\theta}$ is equal to θ except with $\bar{\theta}_i = 0$; all we have done in rewriting the above expression is to make the θ_i term explicit in the objective. However, this still contains the $|\theta_i|$ term, which is non-differentiable and therefore difficult to optimize. To get around this we make the observation that the sign of θ_i must either be non-negative or non-positive. But if we *knew* the sign of θ_i, then $|\theta_i|$ becomes just a linear term. That, is, we can rewrite the objective as

$$J(\theta) = \frac{1}{2}\|X\bar{\theta} + X_i\theta_i - \vec{y}\|_2^2 + \lambda\|\bar{\theta}\|_1 + \lambda s_i\theta_i$$

where s_i denotes the sign of θ_i, $s_i \in \{-1, 1\}$. In order to update θ_i, we can just compute the optimal θ_i for both possible values of s_i (making sure that we restrict

the optimal θ_i to obey the sign restriction we used to solve for it), then look to see which achieves the best objective value.

For each of the possible values of s_i, compute the resulting optimal value of θ_i. [Hint: to do this, you can fix s_i in the above equation, then differentiate with respect to θ_i to find the best value. Finally, clip θ_i so that it lies in the allowable range — i.e., for $s_i = 1$, you need to clip θ_i such that $\theta_i \geq 0$.]

(b) Implement the above coordinate descent algorithm using the updates you found in the previous part. We have provided a skeleton `theta = l1ls(X,y,lambda)` function in the `q3/` directory. To implement the coordinate descent algorithm, you should repeatedly iterate over all the θ_i's, adjusting each as you found above. You can terminate the process when θ changes by less than 10^-5 after all n of the updates.

(c) Test your implementation on the data provided in the `q3/` directory. The `[X, y, theta_true] = load_data;` function will load all the data — the data was generated by `y = X*theta_true + 0.05*randn(20,1)`, but `theta_true` is sparse, so that very few of the columns of `X` actually contain relevant features. Run your `l1ls.m` implementation on this data set, ranging λ from 0.001 to 10. Comment briefly on how this algorithm might be used for feature selection.

~~4.~~ K-Means Clustering

In this problem you'll implement the K-means clustering algorithm on a synthetic data set. There is code and data for this problem in the `q4/` directory. Run `load 'X.dat';` to load the data file for clustering. Implement the `[clusters, centers] = k_means(X, k)` function in this directory. As input, this function takes the $m \times n$ data matrix `X` and the number of clusters `k`. It should output a m element vector, `clusters`, which indicates which of the clusters each data point belongs to, and a $k \times n$ matrix, `centers`, which contains the centroids of each cluster. Run the algorithm on the data provided, with $k = 3$ and $k = 4$. Plot the cluster assignments and centroids for each iteration of the algorithm using the `draw_clusters(X, clusters, centroids)` function. For each k, be sure to run the algorithm several times using different initial centroids.

~~5.~~ The Generalized EM algorithm

When attempting to run the EM algorithm, it may sometimes be difficult to perform the M step exactly — recall that we often need to implement numerical optimization to perform the maximization, which can be costly. Therefore, instead of finding the global maximum of our lower bound on the log-likelihood, and alternative is to just increase this lower bound a little bit, by taking one step of gradient ascent, for example. This is commonly known as the Generalized EM (GEM) algorithm.

Put slightly more formally, recall that the M-step of the standard EM algorithm performs the maximization

explicitly by numerical optimization

$$\theta := \arg\max_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}.$$

The GEM algorithm, in constrast, performs the following update in the M-step:

implicitly by gradient descent

$$\theta := \theta + \alpha \nabla_\theta \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i(z^{(i)})}$$

where α is a learning rate which we assume is choosen small enough such that we do not decrease the objective function when taking this gradient step.

(a) Prove that the GEM algorithm described above converges. To do this, you should show that the the likelihood is monotonically improving, as it does for the EM algorithm — i.e., show that $\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)})$.

(b) Instead of using the EM algorithm at all, suppose we just want to apply gradient ascent to maximize the log-likelihood directly. In other words, we are trying to maximize the (non-convex) function

$$\ell(\theta) = \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta)$$

so we could simply use the update

$$\theta := \theta + \alpha \nabla_\theta \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}; \theta).$$

Show that this procedure in fact gives the same update as the GEM algorithm described above. both derivative result in the same equation