CS 536: Homework 3

Due: March 27, 11:59pm EST

Instructions: Same as homework 1.

1 Bayesian Regression (10 pts)

Recall Bayesian estimation from homework 1. Here we estimate a parameter θ by maximizing the weighted log likelihood

$$\theta_{new} = argmax_{\hat{\theta}} P(\hat{\theta}) P(X_1, X_2, \ldots X_n | \hat{\theta})$$

The weight function $P(\hat{\theta})$ is commonly known as the *prior*.

1. Show that the Ridge regression estimate is equivalent to doing Bayesian estimation as above by assuming a Gaussian prior on each weight value, i.e. assuming $w_i \sim N(0, \frac{\sigma^2}{\lambda})$.
2. Show that the Lasso estimate is equivalent to doing Bayesian estimation as above by assuming a Laplace prior[1] on each weight value, i.e. assuming $w_i \sim L(0, \frac{\sigma^2}{\lambda})$.

Here λ is the penalty used in Ridge/Lasso regression and the regression model is $y = \sum_{i=1}^{n} w_i x_i + \xi$, where $\xi \sim N(0, \sigma^2)$ is Gaussian noise.

2 Ridge vs Lasso (10 pts)

Consider regression in 2 dimensions, i.e. $y = w_1 x_1 + w_2 x_2 + \xi$, where $\xi \sim N(0, \sigma^2)$ is Gaussian noise.

1. Derive closed form expressions for the Ridge and Lasso estimates for the above 2-dimensional problem.
2. How do the estimates change as the co-variance between x_1 and x_2 changes.

3 VC Dimension Bounds (10 pts)

Let C be a finite class of functions from $\mathcal{R}^d \mapsto \{0, 1\}$. Show that the VC dimension of C is at most $\log |C|$, where $|C| =$ the number of functions in C.

4 VC Dimension of Majority (10 pts)

Show that if hypothesis class H has VC-dimension d, then the class $MAJ_k(H)$ has VC-dimension $O(kd \log kd)$. Here, $MAJ_k(H)$ is the class of functions achievable by taking majority votes over k functions in H. Note that we are only asking for an upper bound here, not a lower bound.

[1] https://en.wikipedia.org/wiki/Laplace_distribution

5 Ridge in RKHS (10 pts)

Given a regression problem $y = \sum_{i=1}^{n} w_i x_i + \xi$, where $\xi \sim N(0, \sigma^2)$, show how one would run ridge regression efficiently in a different feature space $\phi : \mathcal{R}^n \mapsto H$. Here H is a reproducing kernel Hilbert space with a kernel function $K : \mathcal{R}^n \times \mathcal{R}^n \mapsto \mathcal{R}$. Your algorithm should run in time polynomial in n and the size of training data and the time required to compute the kernel function on a pair of points. The run time should have no dependence on the dimensionality of the Hilbert space.

Hint: You might find the following equality from linear algebra useful:

$$(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = P B^T (B P B^T + R)^{-1}.$$

6 L_0 Regression (10 pts)

Consider a regression problem $y = \sum_{i=1}^{n} w_i x_i + \xi$, where $\xi \sim N(0, \sigma^2)$. Given $(\vec{X}_1, y_1), (\vec{X}_2, y_2), \ldots (\vec{X}_m, y_m)$, let $\boldsymbol{X}$ be the $m \times n$ data matrix where $\boldsymbol{X}_{i,j} = \vec{X}_{i,j}$ and let $\boldsymbol{Y} = [y_1, y_2, \ldots, y_m]^T$. Define

$$\hat{w} = min_w \frac{1}{2m} \sum_{i=1}^{m} (y_i - \vec{X}_i \cdot w)^2 + \lambda \|w\|_0$$

Show that if $\frac{1}{m} \boldsymbol{X^T X} = \boldsymbol{I}$ then

$$\hat{w}_j = \begin{cases} \frac{1}{m} v_j^T \mathbf{Y} & \text{if} \quad \frac{1}{m} v_j^T \mathbf{Y} > \sqrt{2\lambda} \\ 0 & \text{if} \quad \frac{1}{m} v_j^T \mathbf{Y} \in [-\sqrt{2\lambda}, \sqrt{2\lambda}] \\ \frac{1}{m} v_j^T \mathbf{Y} & \text{if} \quad \frac{1}{m} v_j^T \mathbf{Y} < -\sqrt{2\lambda} \end{cases}$$

for all $j = 1 \ldots n$. Here v_j is the jth column of $\boldsymbol{X}$ and $\|w\|_0 =$ number of non-zeros in w.

7 Infinite VC Dimension (10pts)

Define a function $h_w(x) = sgn(sin(wx))$ for $w \in \mathbb{R}$. Let $H = \{h_w(x) : w \in \mathbb{R}\}$. Show that the VC dimension of H is infinite.

8 Matchmaking via Lasso (30pts)

You have just launched a new dating app. In order to gain an edge among your competitors, you would like to get a better understanding of your customer base. Towards that goal you want to group your users into different types, say based on their personality[2]. Based on a user's information and interaction history, you have represented each user as a high dimensional vector. Your goal now is to partition these vectors into k groups. Formally,

You are given a set S of n vectors in $\mathbb{R}^d$. You want to output k sets $S_1, S_2, \ldots S_k$ where $S_i \subseteq S$ and $S_i \cap S_j = \phi, \forall i \neq j$ and $\cup_{i=1}^{k} S_i = S$. In order to make this problem well defined, we need to make a reasonable assumption about the sets S_i[3]. Here is one such assumption

1. **Low Rank Assumption:** For each $i = 1 \ldots k$, let A_i be the $|S_i| \times d$ matrix where each row corresponds to the vector of one of the users in the set S_i. Then, A_i has rank $\frac{d}{k}$.

Show how you would use Lasso to get the correct partitioning under the above assumption. You can assume that all sets are of the same size and $rank(\sum_{i=1}^{k} A_i) = d$. Furthermore, for each set S_i, any subset of $\frac{d}{k}$ points in S_i are linearly independent.

[Hint: The low rank assumption implies that points within a set lie in a low dimensional space.]

[2]That way you can suggest better matches. For instance, you can recommend people of similar personality type more often to a person.

[3]These sets represent "similar" people and hence must have special structure.

9 Coordinate Gradient Descent (20pts)

Recall that Lasso solves the following loss minimization problem $min_w \frac{1}{m}\sum_{i=1}^{m}(y_i - \vec{X_i}\cdot w)^2 + \lambda\|w\|_1$. Given a vector w, suppose we want to locally improve it by just changing one coordinate, say coordinate j. To do this we will minimize $\frac{1}{m}\sum_{i=1}^{m}(y_i - \vec{X_i}\cdot \hat{w})^2 + \lambda\|\hat{w}\|_1$ over all vectors $\hat{w}$ such that $\hat{w}_i = w_i, \forall i \neq j$. So this is a one-dimensional problem.

1. Derive a close form expression for the update of $\hat{w}_j$.

Using the above observation, a practical algorithm for Lasso is the following:

- Set $w_0 = \vec{0}$.
- for $j = 1 \ldots n$
  - Update the current weight vector to a new one by optimizing over j.

The above for loop is repeated until the weight vector converges or the total loss does not decrease significantly from one step to the next.

2. Implement your solution to Problem 8 using the above algorithm as a subroutine to solve Lasso.
3. Download the CIFAR-10 dataset from here: http://www.cs.toronto.edu/ kriz/cifar.html
4. The dataset has 60,000 images belonging to 10 different classes. 50,000 images are for training and 10,000 for testing. Each image is a vector of length 3072. Use your implementation above to partition the test set into 10 categories. Report your error. Error is defined as $\frac{\text{\# pairs of images that have the same class label but belong to different sets in your solution}}{\text{total \# pairs of images}}$.

Note: Use the training data or a subset of it to tune the value of λ. Submit your code. Your algorithm for Problem 8 is probably designed to work when *all* the sets lie in a low dimensional space. You will have to modify it in a reasonable way to run it on this dataset.