Instructions There are 4 questions on this assignment. The third question involves coding. Do not attach your code to the writeup. Instead, copy your implementation to

/afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW2

To write in this directory, you need a kerberos instance for andrew, or you can log into, for example, unix.andrew.cmu.edu. Please submit each problem separately with your name and userid on each problem. Refer to the webpage for policies regarding collaboration, due dates, and extensions.

1 Learning with the L1 Norm, L1 Error [Yucheng, 9 points]

Suppose you want to predict an unknown value $Y$, but you are only given a sequence of noisy observations $x_1...x_n$ of $Y$ with i.i.d. noise ($x_i = Y + \epsilon_i$).

We have seen in the last homework, that if we assume the noise is i.i.d. Gaussian ($\epsilon_i \sim N(0, \sigma^2)$), finding the maximum likelihood estimate for $Y$ is equivalent to finding the value $\hat{y}$ which minimizes the sum of least squares error to the $x$'s. That is to say:

$$\hat{y} = \arg \min_y \sum_{i=1}^{n} (y - x_i)^2$$

And there is a simple closed form solution:

$$\hat{y} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

It also was suggested that if we assume the noise is i.i.d. Laplace ($\epsilon_i \sim \text{Laplace}(0, b)$) with pdf

$$f_{\epsilon_i}(x) = \frac{1}{2b} \exp\left( -\frac{|x|}{b} \right)$$

we end up with a more robust estimate of the solution. We will show this more rigorously in this question.

1. [1 pts] Begin by showing that finding the MLE for $Y$, assuming Laplace noise, is equivalent to finding the value $\hat{y}$ that minimizes the sum of absolute errors. That is to say:

$$L(y) = \sum_{i=1}^{n} |y - x_i|$$

$$\hat{y} = \arg \min_y L(y)$$
2. [3 pts] A standard way to minimize a loss function is to take the derivative and set it to zero. This loss function is not directly differentiable. However, it is easy to see that the function is not differentiable only where \( y \) has the same value as any of the \( x \)'s.

Assume that the \( x \)'s are distinct and are sorted in ascending order (\( \forall i, \forall j > i, X_i > X_j \)). Constraining \( y \) to be in between two consecutive values of \( x \) (That is to say \( x_i < y < x_{i+1} \)), find an expression for the gradient \( \frac{dL(y)}{dy} \). (Hint: You may need to consider the \( x \)'s which are \( > y \) separately from the \( x \)'s which are \( < y \)).

3. [2 pts] Assuming that there are an even number of \( x \)'s, what are the values of \( y \) for which \( \frac{dL(y)}{dy} = 0 \)?

4. [2 pts] If I have an odd number of \( x \)'s, there is no value for \( y \) where \( \frac{dL(y)}{dy} = 0 \). However, there is a value \( y_0 \) such that \( \frac{dL(y)}{dy} < 0 \) for \( y < y_0 \) and \( \frac{dL(y)}{dy} > 0 \) for \( y > y_0 \). What is \( y_0 \)?

5. [1 pts] Your answer to the last two parts is therefore the solution to \( \hat{y} \). Give a brief explanation why this solution may be more robust against outliers in the data (as compared to least squares error).

2 L1 Norm in Regression and Feature Selection [Babis, 21 points]

You are given a set of points and the corresponding outputs: \( D = \{ (x_1, y_1), \ldots, (x_n, y_n) \} \) where \( x_i \in \mathbb{R}^d \) and \( y_i \in \mathbb{R} \) for \( i = 1 \ldots n \). You want to use this data to train a linear predictor \( y = w^T F(x) \) where \( w^T = (w_1, \ldots, w_k) \) and \( F(x)^T = (f_1(x), \ldots, f_k(x)) \), where \( k \) is finite. As we know from class, \( f_i \) is the \( i \)-th basis function/feature for our learning problem. Consider the following objective function:

\[
J(w, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - w^T F(x_i))^2 + \lambda ||w||_1
\]

where \( ||w||_1 = \sum_{i=1}^{k} |w_i| \) and \( ||w||_1 \) is the l1 norm of the vector \( w \). We use our data to learn the vector \( w \) by minimizing \( J(w, \lambda) \), i.e.,

\[
w^* = \arg \min_{w \in \mathbb{R}^k} J(w, \lambda)
\]

The above optimization criterion typically leads to an effective feature selection by picking a large value for parameter \( \lambda \). In other words, if we pick a large value for \( \lambda \) many \( w_i \)'s will become 0. Therefore, the corresponding basis functions/features will be unimportant for our predictor.

(1) [2 points] A real valued function \( f \) defined on an interval \([a, b]\), or in general on any convex subset of a vector space, is called convex if for any two points \( x_1, x_2 \) in \([a, b]\), we have that \( f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2) \), for \( t \in [0, 1] \). Prove that \( f(x) = |x| \) is a convex function, \( x \in \mathbb{R} \).

(2) [2 points] Define a subderivative of a convex function \( f : C \rightarrow \mathbb{R} \) at a point \( x_0 \in C \) as the value \( c \) such that the following holds: \( f(x) - f(x_0) \geq c(x - x_0) \) for all \( x \in C \). One can show that the set of subderivatives at point \( x_0 \) is a nonempty closed interval \([c_1, c_2]\) where:

\[
c_1 = \lim_{x \to x_0^-} \frac{f(x) - f(x_0)}{x - x_0}
\]

\[
c_2 = \lim_{x \to x_0^+} \frac{f(x) - f(x_0)}{x - x_0}
\]

Consider the function \( f(x) = |x| \), which you showed in question (1) to be convex. What is the set of subderivatives of \( f \) at point \( 0 \)? This set is also known as subdifferential.

(3) [4 points] Compute the subdifferential \( \partial_w J \) of function \( J(w, \lambda) \) with respect to the parameter \( w_i \).

Hint 1: Compute first the derivative of the first term of the function with respect to \( w_i \) and then the subdifferential of the second term. Make sure you consider all three cases. Write the gradient of the first term as \( a_i w_i - r_i \) where \( r_i \) is given in equation (3).

Hint 2: For differentiable functions the subgradient is simply the gradient of the function.
Hint 3: The subgradient you are asked to compute will be equal to the sum of the subgradient of the first term (its gradient actually since it is differentiable) plus the subgradient of the second term. Use your answer from part (2).

4) [6 points] Find the global optimizer \( w_i^* \) using the fact that “\( w^* \) is a global optimizer of the convex function \( f \) iff \( 0 \in \partial f(w^*) \)” in other words, find \( w_i^* \) such that \( 0 \in \partial w_i^*J \).

5) [7 points] Let’s explore more carefully the relation between the regularization parameter \( \lambda \), the weight of the \( i \)-th basis function \( w_i \) and the quantity \( r_i \) given by the following equation:

\[
r_i = \frac{1}{n} \sum_{j=1}^{n} f_i(x_j)(y_j - \sum_{m=1, m \neq i}^{k} w_m f_m(x_j))
\]

What is the meaning of \( r_i \) ([2 points])? Provide a plot of \( w_i^* \) vs. \( r_i \). Where does \( \lambda \) appear in this plot ([5 points])?

3 Boosting [Shay, 50 points]


The algorithm details of Schapire’s tutorial differ slightly from those in the textbook. Both will yield the same results, but the internal values of weights will differ. The proofs of 3.1 follow Schapire’s tutorial. Please use Schapire’s algorithm for Problem 3.1 and 3.2. You may implement either and should obtain identical results for Problem 3.3.

3.1 [20 Points] Analyzing the training error of boosting

Consider the AdaBoost algorithm you saw in class. In this question we will try to analyze the training error of Boosting.

1. (4 points) Given a set of \( m \) examples, \((x_i, y_i)\) \( (y_i \) is the class label of \( x_i \), \( i = 1, \ldots, m \), let \( h_t(x) \) be the weak classifier obtained at step \( t \), and let \( \alpha_t \) be its weight. Recall that the final classifier is

\[
H(x) = \text{sign}(f(x)), \text{ where } f(x) = \sum_{t=1}^{T} \alpha_t h_t(x).
\]

Show that the training error of the final classifier can be bounded from above by an exponential loss function:

\[
\frac{1}{m} \sum_{i=1}^{m} I(H(x_i) \neq y_i) \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-f(x_i)y_i),
\]

where \( I(a = b) \) is the indicator function. It is equal to 1 if \( a = b \), and 0 otherwise

**Hint**: \( e^{-x} \geq 1 \iff x \leq 0 \).

2. (4 points) Remember that

\[
D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}
\]

Start with this recursive definition to prove the following.

\[
\frac{1}{m} \sum_{i=1}^{m} \exp(-f(x_i)y_i) = \prod_{t=1}^{T} Z_t,
\]
where $Z_t$ is the normalization factor for distribution $D_{t+1}$:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i)).$$

(7)

*Hint: remember that $e^{\sum_i g_i} = \prod_i e^{g_i}, D_1(i) = \frac{1}{m}$, and that $\sum_i D_{t+1}(i) = 1$.\)

3. (4 points) Equation 6 suggests that the training error can be reduced rapidly by greedily optimizing $Z_t$ at each step. You have shown that the error is bounded from above:

$$\epsilon_{\text{training}} \leq \prod_{t=1}^{T} Z_t.$$

Observe that $Z_1, \ldots, Z_{t-1}$ are determined by the first $(t-1)$ rounds of boosting, and we cannot change them on round $t$. A greedy step we can take to minimize the training error bound on round $t$ is to minimize $Z_t$.

In this question, you will prove that for binary weak classifiers, $Z_t$ from Equation 7 is minimized by picking $\alpha_t$ as:

$$\alpha^*_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right),$$

(8)

where $\epsilon_t$ is the training error of weak classifier $h_t$ for the weighted dataset:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(x_i) \neq y_i).$$

where $I$ is the indicator function. For this proof, only consider the simplest case of binary classifiers, i.e. the output of $h_t(x)$ is binary, $\{-1, +1\}$.

For this special class of classifiers, first show that the normalizer $Z_t$ can be written as:

$$Z_t = (1 - \epsilon_t) \exp(-\alpha_t) + \epsilon_t \exp(\alpha_t).$$

*Hint: consider the sums over correctly and incorrectly classified examples separately.\)

Now, prove that the value of $\alpha_t$ that minimizes this definition of $Z_t$ is given by Equation 8.

4. (4 points) Prove that for the above value of $\alpha_t$

$$Z_t = 2 \sqrt{\epsilon_t(1 - \epsilon_t)}$$

Furthermore, let $\epsilon_t = \frac{1}{2} - \gamma_t$, prove that

$$Z_t \leq \exp(-2\gamma_t^2)$$

*Hint: $\log(1 - x) \leq -x$ for $0 < x \leq 1$.\)

Therefore

$$\epsilon_{\text{training}} \leq \prod_t Z_t \leq \exp(-2 \sum_t \gamma_t^2)$$

Finally prove that, if each weak classifier is slightly better than random, so that $\gamma_t \geq \gamma$, for some $\gamma > 0$, then the training error drops exponentially fast in $T$, i.e.

$$\epsilon_{\text{training}} \leq \exp(-2T\gamma^2)$$

5. (4 points) Show that in each round of boosting, there always exists a weak classifier $h_t$ such that its training error on the weighted dataset $\epsilon_t \leq 0.5$. Also show that for $\epsilon_t = 0.5$ the training error can get "stuck" above zero.

*Hint: $D_t(i)$s do not change over $t$.\)
3.2 [5 Points] Adaboost on a toy dataset

Now we will apply Adaboost to classify a toy dataset. Consider the following dataset in Figure 1a). The dataset consists of 4 points, (X₁ : 0, -1, -), (X₂ : 1, 0, +), (X₃ : -1, 0, +) and (X₄ : 0, 1, -).

1. Use simple decision stumps as weak classifiers. (For description of decision stumps, refer to Problem 2.3) Now for T = 4, show how Adaboost works for this dataset. For each timestep remember to compute the following numbers:

\[ \epsilon_t, \alpha_t, Z_t, D_t(i) \forall i, \]

Also for each timestep draw your weak classifier. For example h₁ can be as shown in 1b).

2. What is the training error of Adaboost?

3. Is the above dataset linearly separable? Explain why Adaboost does better than a decision stump in the above dataset.

3.3 [25 Points] Implementation

Implement the AdaBoost algorithm (page 658 in the Bishop book) using a decision stump as the weak classifier.

AdaBoost trains a sequence of classifiers. Each classifier is trained on the same set of training data (xᵢ, yᵢ), i = 1, ..., m, but with the significance Dᵢ(i) of each example \{xᵢ, yᵢ\} weighted differently. At each iteration, a classifier, \( h_t(x) \rightarrow \{-1, 1\} \), is trained to minimize the weighted classification error, \( \sum_{i=1}^{m} D_t(i) \cdot I(h_t(x_i) \neq y_i) \), where \( I \) is the indicator function (0 if the predicted and actual labels match, and 1 otherwise).

The overall prediction of the AdaBoost algorithm is a linear combination of these classifiers, \( H_T(x) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(x)) \). \textit{Note:} The textbook uses \( w_i \equiv D_t(i) \).

A decision stump is a decision tree with a single node. It corresponds to a single threshold in one of the features, and predicts the class for examples falling above and below the threshold respectively, \( h_t(x) = C_1 I(x^j \geq c) + C_2 I(x^j < c) \), where \( x^j \) is the \( j \)th component of the feature vector \( x \). Unlike in class, where we split on Information Gain, for this algorithm split the data based on the weighted classification accuracy described above, and find the class assignments \( C_1, C_2 \in \{-1, 1\} \), threshold \( c \), and feature choice \( j \) that maximizes this accuracy.

1. (15 points) Submit your source code to:
   /afs/andrew.cmu.edu/course/10/701/Submit/your_andrew_id/HW2

2. Evaluate your AdaBoost implementation on the Bupa Liver Disorder dataset that is available for download from the course website. The classification problem is to predict whether an individual has a liver disorder (indicated by the selector feature) based on the results of a number of blood tests and levels of alcohol consumption. Use 90% of the dataset for training and 10% for testing. Average your results over 50 random splits of the data into training sets and test sets. Limit the number of boosting iterations to 100. In a single plot show:
• average training error after each boosting iteration
• average test error after each boosting iteration

3. (5 points) Using all of the data for training, display the selected feature component $j$, threshold $c$, and class label $C_1$ of the decision stump $h_t(x)$ used in each of the first 10 boosting iterations ($t = 1, 2, ..., 10$)

4. (5 points) Using all of the data for training, in a single plot, show the empirical cumulative distribution functions of the margins $y_T(x_i)$ after 10, 50 and 100 iterations respectively, where $f_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$. Notice that in this problem, before calculating $f_T(x)$, you should normalize the $\alpha_t$s so that $\sum_{t=1}^{T} \alpha_t = 1$. This is to ensure that the margins are between -1 and 1.

\textit{hint: the empirical cumulative distribution function of a random variable $X$ at $x$ is the proportion of times $X \leq x$.}

4 Linear Regression and LOOCV [Kate, 20 points]

In class you learned about using cross validation as a way to estimate the true error of a learning algorithm. A solution that provides an almost unbiased estimate of this true error is Leave-One-Out Cross Validation (LOOCV), but it can take a really long time to compute the LOOCV error. In this problem you will derive an algorithm for efficiently computing the LOOCV error for linear regression using the Hat Matrix.\footnote{Unfortunately, such an efficient algorithm may not be easily found for other learning methods.}

Assume that there are $r$ given training examples, $(X_1, Y_1), (X_2, Y_2), \ldots, (X_r, Y_r)$, where each input data point $X_i$, has $n$ real valued features. The goal of regression is to learn to predict $Y$ from $X$. The linear regression model assumes that the output $Y$ is a linear combination of the input features plus Gaussian noise with weights given by $\beta$.

We can write this in matrix form by stacking the data points as the rows of a matrix $X$ so that $x_{ij}$ is the $j$-th feature of the $i$-th data point. Then writing $Y$, $\beta$ and $\epsilon$ as column vectors, we can write the matrix form of the linear regression model as:

$$ Y = X\beta + \epsilon $$

where:

$$ Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_r \end{bmatrix}, X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r1} & x_{r2} & \cdots & x_{rn} \end{bmatrix}, \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \text{ and } \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_r \end{bmatrix} $$

Assume that $\epsilon_i$ is normally distributed with variance $\sigma^2$. We saw in class that the maximum likelihood estimate of the model parameters $\beta$ (which also happens to minimize the sum of squared prediction errors) is given by the Normal equation:

$$ \hat{\beta} = (X^TX)^{-1}X^TY $$

Define $\hat{Y}$ to be the vector of predictions using $\hat{\beta}$ if we were to plug in the original training set $X$:

$$ \hat{Y} = X\hat{\beta} = X(X^TX)^{-1}X^TY = HY $$

where we define $H = X(X^TX)^{-1}X^T$ ($H$ is often called the Hat Matrix).

As mentioned above, $\hat{\beta}$, also minimizes the sum of squared errors:

$$ \text{SSE} = \sum_{i=1}^{r} (Y_i - \hat{Y}_i)^2 $$
Now recall that the Leave-One-Out Cross Validation score is defined to be:

$$\text{LOOCV} = \sum_{i=1}^{r} (Y_i - \hat{Y}_i^{(-i)})^2$$

where $\hat{Y}^{(-i)}$ is the estimator of $Y$ after removing the $i$-th observation (i.e., it minimizes $\sum_{j \neq i} (Y_j - \hat{Y}_j^{(-i)})^2$).

1. (3 points) What is the time complexity of computing the LOOCV score naively? (The naive algorithm is to loop through each point, performing a regression on the $r-1$ remaining points at each iteration.)

   *Hint:* The complexity of matrix inversion for a $k \times k$ matrix is $O(k^3)$.

2. (1 point) Write $\hat{Y}_i$ in terms of $H$ and $Y$.

3. (5 points) Show that $\hat{Y}^{(-i)}$ is also the estimator which minimizes SSE for $Z$ where

   $$Z_j = \begin{cases} 
   Y_j, & j \neq i \\
   \hat{Y}_i^{(-i)}, & j = i 
   \end{cases}$$

4. (1 point) Write $\hat{Y}_i^{(-i)}$ in terms of $H$ and $Z$. By definition, $\hat{Y}_i^{(-i)} = Z_i$, but give an answer that includes both $H$ and $Z$.

5. (5 points) Show that $\hat{Y}_i - \hat{Y}_i^{(-i)} = H_{ii}Y_i - H_{ii}\hat{Y}_i^{(-i)}$, where $H_{ii}$ denotes the $i$-th element along the diagonal of $H$.

6. (5 points) Show that

   $$\text{LOOCV} = \sum_{i=1}^{r} \left( \frac{Y_i - \hat{Y}_i}{1 - H_{ii}} \right)^2$$

   What is the algorithmic complexity of computing the LOOCV score using this formula?

   *Note:* We see from this formula that the diagonal elements of $H$ somehow indicate the impact that each particular observation has on the result of the regression.

---

2There are faster algorithms out there but for simplicity we’ll assume that we are using the naive $O(k^3)$ algorithm.