COEN 4890/EECE 5890: Developments in Computing: Machine Learning for Image Processing, Fall 2018: Homework 1

Instructor: Dong Hye Ye

Due: Thursday, October 4^{th}, beginning of class

Instructions There are 3 written questions, plus 2 coding questions. Submit your written answers as papers and your implementation to D2L.

1 Parameter Estimation [15 points]

This question uses a probability distribution called the Poisson distribution. A discrete random variable X follows a Poisson distribution with parameter λ if

$$p(X|\lambda) = \frac{\lambda^X}{X!}e^{-\lambda}.$$

The Poisson distribution is a useful discrete distribution which can be used to model the number of occurrences of something per unit time. For examples, if a bank teller sits at a counter, the number of customers arriving in each time interval, say 30 minutes, is in the Poisson distribution.

Here, we will estimate the parameter λ from n observations $\{X_1, \ldots, X_i, \ldots X_n\}$ (e.g., the number of customers for the i^{th} teller in 30 minutes) which we assume are drawn i.i.d from the Poisson distribution.

1. *(3 points)* Compute the log-likelihood for observations $\{X_1, \ldots, X_n\}$.
2. *(4 points)* Compute the MLE for λ.
3. *(8 points)* Now let's be Bayesian and put a prior distribution over the parameter λ. Your extensive experience in statistics tells you that the a good prior distribution for λ is a Gamma distribution.

   $$p(\lambda|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \lambda > 0.$$

   It is well known that the mode of λ in the Gamma distribution with α and β is $(\alpha - 1)/\beta$ for $\alpha > 1$. Recall that the mode in the statistics represents the value that appears most often (i.e., the maxima of the probability mass function). Then, compute the MAP for λ. (Hint: $\lambda^{\sum X_i + \alpha - 1}e^{-\lambda(n+\beta)}$ can be represented by a Gamma distribution.)

2 Linear Regression and LOOCV [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 the best 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*. (Unfortunately, such an efficient trick 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 β.

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, β and ϵ as column vectors, we can write the matrix form of the linear regression model as:

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

where:

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

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

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

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

$$\begin{aligned} \hat{Y} &= X\hat{\beta} \\ &= X(X^T X)^{-1} X^T Y \\ &= HY \end{aligned}$$

where we define $H = X(X^T X)^{-1} X^T$ (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 (e.g., it minimizes $\sum_{j \neq i} (Y_j - \hat{Y}_j^{(-i)})^2$).

1. *(2 point)* Write $\hat{Y}_i$ in terms of H and Y.
2. *(5 points)* Show that $\hat{Y}^{(-i)}$ is also the estimator which minimizes SSE for Z where

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

3. *(3 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.
4. *(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.
5. *(5 points)* Show that

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

Note: We have the closed-form form for LOOCV of linear regression indicating that the algorithmic complexity is very low.

3 Decision Trees [15 points]

Consider the following set of training examples for the unknown target function $< X_1, X_2 > \rightarrow Y$. Each row indicates the values observed, and how many times that set of values was observed. For example, $(+, T, T)$ was observed 3 times, while $(-, T, T)$ was never observed.

Y	X_1	X_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

1. *(3 points)* What is the sample entropy $H(Y)$ for this training data (with logarithms base 2)?
2. (4 points) What are the information gains $IG(X_1) = H(Y)H(Y|X_1)$ and $IG(X_2) = H(Y)H(Y|X_2)$ for this sample of training data?
3. (8 points) Draw the decision tree based on the information gain (without postpruning) from this sample of training data.

4 Logistic Regression [25 points]

For this problem, you need to download the Breast Cancer dataset from course webpage. The description of this dataset is in `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.names`. I have removed the records with missing values for you. Here, you will obtain the learning curves (accuracy vs. training data size). Implement a logistic regression classifier with the assumption that each attribute value for a particular record is independently generated. You should submit the code electronically to D2L.

1. *(10 points)* Briefly describe how you implement it by giving the pseudocode. The pseudocode must include equations for estimating the classification parameters and for classifying a new example. Remember, this should not be a printout of your code, but a high-level outline.
2. *(15 points)* Plot a learning curve: the accuracy vs. the size of the training data. Generate six points on the curve, using [.01 .02 .03 .125 .625 1] fractions of your training set and testing on the full test set each time. Average your results over 5 random splits of the data into a training and test set (always keep 2/3 of the data for training and 1/3 for testing, but randomize over which points go to training set and which to testing). This averaging will make your results less dependent on the order of records in the file. Specify your choice of regularization parameters and keep those parameters constant for these tests. A typical choice of constants would be $\lambda = 0$ (no regularization).

5 AdaBoost [25 points]

For this problem, you need to download teh Bupa Liver Disorder dataset that is available in the course website. Here, you will 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. Implement the AdaBoost algorithm using a decision stump as the weak classifier. You should submit the code electronically to D2L.

AdaBoost trains a sequence of classifiers. Each classifier is trained on the same set of training data $(\mathbf{x}_i, y_i)$, $i = 1, ..., m$, but with the significance $D_t(i)$ of each example $\{\mathbf{x}_i, y_i\}$ weighted differently. At each iteration, a classifier, $h_t(x) \rightarrow \{-1, 1\}$, is trained to minimize the weighted classification error, $\sum_{i=1}^{m} D_t(i) \cdot I(h_t(\mathbf{x}_i) \neq y_i)$, 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, $H_T(\mathbf{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$.

A decision stump is a decision tree with a single node (a depth 1 decision tree). It corresponds to a single threshold in one of the features, and predicts the class for examples falling above and below the threshold respectively, $h_t(\mathbf{x}) = C_1 I(\mathbf{x}^j \geq c) + C_2 I(\mathbf{x}^j < c)$, where $\mathbf{x}^j$ is the j th component of the feature vector $\mathbf{x}$. 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. *(10 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 $h_t(\mathbf{x})$ used in each of the first 10 boosting iterations ($t = 1, 2, ..., 10$).

2. *(15 points)* 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