HOMEWORK 3:
LINEAR REGRESSION AND LOGISTIC REGRESSION

CMU 10601: MACHINE LEARNING (FALL 2016)
OUT: Sep. 14, 2016
DUE: 5:30 pm, Sep. 26, 2016
TAs: Simon Shaolei Du, Tianshu Ren, Sriram Vasudevan

Instructions

- **Homework Submission:** Submit your answers and results to Gradescope. You will use Autolab to submit your code in Problem 2.2. For Gradescope, you will need to specify which pages go with which question. We provide a LaTeX template which you can use to type up your solutions. Please check Piazza for updates about the homework.
- **Collaboration policy**: The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes (including code) are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved. Please refer to the course webpage.

1 Linear Regression [40 pts]

In this problem, you will derive the linear regression and ridge linear regression formulas from a statistical point of view. First let's see how frequentists approach the following inference problem:
Consider the model

$$p(y|\mathbf{x}) = \mathcal{N}(y|\mathbf{w}^\top \mathbf{x}, \sigma^2)$$

where $\mathbf{x}$ is the feature, y is the label. $\mathcal{N}(y|\mu, \sigma^2)$ is the Gaussian distribution with mean μ and variance σ^2. Throughout this question, we assume σ is known. Suppose we have the following data independently generated from this model:

$$\mathcal{D} = (\mathbf{X}, \mathbf{y})$$

where $\mathbf{X} \in \mathbb{R}^{d\times n}$, the i-th column $\mathbf{X}_i$ are the features of the i-th training sample and $y \in \mathbb{R}^n$, y_i is the label of the i-th training sample.

(a) **[10 Points]** Please write out the likelihood function $p(\mathcal{D}|\mathbf{w})$ in terms of $\mathbf{X}$,y,w and σ.

$$p(\mathcal{D}|\mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2}) = (\frac{1}{\sqrt{2\pi}\sigma})^n \exp(-\frac{\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2}) \tag{1}$$

(b) **[10 Points]** Maximum Likelihood Estimation: Please derive a formula for $\mathbf{w}$ that maximize $p(\mathcal{D}|\mathbf{w})$ (hint: consider log-likelihood).

Taking the gradient of $p(\mathcal{D}|\mathbf{w})$ in Equation (**??**) with respect to $\mathbf{w}$ and set it to 0, we get

$$\nabla_{\mathbf{w}} p(\mathcal{D}|\mathbf{w}) \propto \sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i = \sum_{i=1}^{n} y_i \mathbf{x}_i - \sum_{i=1}^{n}(\mathbf{x}_i \mathbf{x}_i^\top)\mathbf{w} = \mathbf{X}\mathbf{y} - \mathbf{X}\mathbf{X}^\top \mathbf{w} = 0 \tag{2}$$

$$\mathbf{w}_{\text{MLE}} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{y} \tag{3}$$

(c) **[10 Points]** Now suppose we have become Bayesians and we have a prior for $\mathbf{w}$:

$$p(\mathbf{w}) = \Pi_{j=1}^{d} \mathcal{N}\left(w_j|0, \gamma^2\right),$$

i.e. each coordinate of $\mathbf{w}$ is independent of other coordinates and have the mean 0 and variance γ^2. Again, we assume variance γ^2 is known. Please write out the posterior probability: $p(\mathbf{w}|\mathcal{D})$ in terms of $\mathbf{X}$, y, $\mathbf{w}$, σ and γ.

First we compute the joint probability:

$$p(\mathcal{D}, \mathbf{w}) = p(\mathbf{w})p(\mathcal{D}|\mathbf{w}) \tag{4}$$

$$= (\frac{1}{\sqrt{2\pi}\sigma})^n \exp(-\frac{\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2})(\frac{1}{\sqrt{2\pi}\gamma})^d \exp(-\frac{\mathbf{w}^\top \mathbf{w}}{2\gamma^2}) \tag{5}$$

$$= (\frac{1}{\sqrt{2\pi}\sigma})^n (\frac{1}{\sqrt{2\pi}\gamma})^d \exp(-\frac{\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{2\sigma^2} - \frac{\mathbf{w}^\top \mathbf{w}}{2\gamma^2}) \tag{6}$$

Using the Bayes rule, we have

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}, \mathbf{w})}{p(\mathcal{D})} \tag{7}$$

By marginalizing $\mathbf{w}$ from the joint probability, we get

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}, \mathbf{w})}{\int_{\mathbf{w}\in\mathbb{R}^d} p(\mathcal{D}, \mathbf{w})\mathbf{dw}} \tag{8}$$

(d) **[10 Points]** Please derive a formula for $\mathbf{w}$ that maximize the posterior (hint: use Bayes' rule to simplify calculation).

Our objective is to find $\mathbf{w}$ such that $p(\mathbf{w}|\mathcal{D})$ is maximized. However note that $p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D},\mathbf{w})}{p(\mathcal{D})}$, and $\mathbf{w}$ does not appear in the denominator $p(\mathcal{D})$. Thus, $p(\mathbf{w}|\mathcal{D})$ is maximized if and only if $p(\mathbf{w},\mathcal{D})$ is maximized. Here, it sufficies that we find $\mathbf{w}$ to maximize $p(\mathbf{w},\mathcal{D})$.

Again, taking the gradient of $p(\mathbf{w},\mathcal{D})$ in Equation (**??**) with respect to $\mathbf{w}$ and set it to 0, we get

$$\nabla_{\mathbf{w}} p(\mathbf{w},\mathcal{D}) \propto \frac{\sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i}{\sigma^2} - \frac{\mathbf{w}}{\gamma^2} = 0 \tag{9}$$

$$\gamma^2 \sum_{i=1}^{n}(y_i - \mathbf{w}^\top \mathbf{x}_i)\mathbf{x}_i - \sigma^2\mathbf{w} = \gamma^2\mathbf{X}\mathbf{y} - (\gamma^2(\mathbf{X}\mathbf{X}^\top) + \sigma^2\mathbf{I})\mathbf{w} = 0 \tag{10}$$

$$\mathbf{w}_{\text{MAP}} = (\gamma^2\mathbf{X}\mathbf{X}^\top + \sigma^2\mathbf{I})^{-1}\gamma^2\mathbf{X}\mathbf{y} \tag{11}$$

2 Logistic Regression [60 pts]

You will implement a logistic regression classifier and apply it to a two-class classification problem. Your code will be autograded by a series of test scripts that run on Autolab. To get started, download the template for Homework 3 from the course website, and extract the contents of the compressed directory. In the archive, you will find one `.m` file for each of the functions that you are asked to implement, along with a file called `HW3Data.mat` that contains the data for this problem. You can load the data into Octave by executing `load("HW3Data.mat")` in the Octave interpreter. Make sure not to modify any of the function headers that are provided.

2.1 Theoretical Derivation

(a) **[5 Points]** In logistic regression, our goal is to learn a set of parameters by maximizing the conditional log likelihood of the data. Assuming you are given a dataset with n training examples and p features, write down a formula for the conditional log likelihood of the training data in terms of the the class labels $y^{(i)}$, the features $x_1^{(i)}, \ldots, x_p^{(i)}$, and the parameters $w_0, w_1, \ldots, w_p$, where the superscript (i) denotes the sample index. This will be your objective function for gradient ascent.

Taking $\mathbf{x}^{(i)}$ to be a $(p+1)$-dimensional vector where $x_0^{(i)} = 1$, the likelihood of the data $\mathcal{D}$ given the parameters $\mathbf{w}$ is:

$$L(\mathcal{D}|\mathbf{w}) = L(\mathbf{y}|\mathbf{X},\mathbf{w}) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n}\left(\frac{e^{\mathbf{w}^T\mathbf{x}^{(i)}}}{1+e^{\mathbf{w}^T\mathbf{x}^{(i)}}}\right)^{y_i}\left(\frac{1}{1+e^{\mathbf{w}^T\mathbf{x}^{(i)}}}\right)^{(1-y_i)} \tag{12}$$

$$= \prod_{i=1}^{n}\frac{\left(e^{\mathbf{w}^T\mathbf{x}^{(i)}}\right)^{y_i}}{1+e^{\mathbf{w}^T\mathbf{x}^{(i)}}} \tag{13}$$

Hence, the log-likelihood is:

$$l(\mathcal{D}|\mathbf{w}) = \log L(\mathcal{D}|\mathbf{w}) = \sum_{i=1}^{n} y_i\left(\mathbf{w}^T\mathbf{x}^{(i)}\right) - \log\left(1+e^{\mathbf{w}^T\mathbf{x}^{(i)}}\right) \tag{14}$$

(b) **[5 Points]** Compute the partial derivative of the objective function with respect to w_0 and with respect to an arbitrary w_j, i.e. derive $\partial f/\partial w_0$ and $\partial f/\partial w_j$, where f is the objective that you provided above. Please show all derivatives can be written in a finite sum form.

The partial derivate of the log-likelihood wrt $\mathbf{w}_j$, $j \in \{0, ..., p\}$ is:

$$\frac{\partial l(\mathcal{D}|\mathbf{w})}{\partial \mathbf{w}_j} = \sum_{i=1}^{n}\mathbf{x}_j^{(i)}\left[y_i - \frac{e^{\mathbf{w}^T\mathbf{x}^{(i)}}}{1+e^{\mathbf{w}^T\mathbf{x}^{(i)}}}\right] \tag{15}$$

For $j = 0$, $\mathbf{x}_j^{(i)} = 1$ by the definition in the previous solution.

2.2 Implementation

(a) **[30 Points]** Implement a logistic regression classifier using stochastic gradient ascent by filling in the missing code for the following functions:

- Calculate the value of the objective function: `obj = LR_CalcObj(XTrain,yTrain,wHat)`
- Calculate the stochastic gradient: `sg = LR_CalcSG(x,y,wHat)`
- Update the parameter value: `wHat = LR_UpdateParams(wHat,sg,eta)`
- Check whether gradient ascent has converged (whether the difference in the values is less than the tolerance): `hasConverged = LR_CheckConvg(oldObj,newObj,tol)`
- Complete the implementation of gradient ascent: `[wHat,objVals] = LR_SGA(XTrain,yTrain)`
- Predict the labels for a set of test examples: `[yHat,numErrors] = LR_PredictLabels(XTest,yTest,wHat)`

where the arguments and return values of each function are defined as follows:

- `XTrain` is an $n \times p$ dimensional matrix that contains one training instance per row
- `yTrain` is an $n \times 1$ dimensional vector containing the class labels for each training instance
- `x` is an $1 \times p$ dimensional vector that contains one training instance
- `y` is a scalar, a class labels for a single training instance
- `wHat` is a $p + 1 \times 1$ dimensional vector containing the regression parameter estimates $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_p$
- `hasConverged` is a scalar that is either true or false.
- `sg` is a $p + 1 \times 1$ dimensional vector containing the value of the a stochastic gradient of the objective function with respect to each parameter in `wHat`
- `eta` is the gradient ascent step size that you should set to `eta = 0.5/sqrt(n)` where `n` is the iteration number
- `obj`, `oldObj` and `newObj` are values of the objective function
- `tol` is the convergence tolerance, which you should set to `tol = 0.0001`
- `objVals` is a vector containing the objective value at each iteration of gradient ascent
- `XTest` is an $m \times p$ dimensional matrix that contains one test instance per row
- `yTest` is an $m \times 1$ dimensional vector containing the true class labels for each test instance
- `yHat` is an $m \times 1$ dimensional vector containing your predicted class labels for each test instance
- `numErrors` is the number of misclassified examples, i.e. the differences between `yHat` and `yTest`

To complete the `LR_SGA` function, you should use the helper functions `LR_CalcObj`, `LR_CalcSG`, `LR_UpdateParams`, and `LR_CheckConvg`, so we can give you partial credits.

See the functions `LR_CalcObj`, `LR_CalcSG`, `LR_UpdateParams`, `LR_CheckConvg`, `LR_SGA` and the prediction function `LR_PredictLabels` in the solution code.

(b) **[0 Points]** Train your logistic regression classifier on the data provided in `XTrain` and `yTrain` with `LR_SGA`, and then use your estimated parameters `wHat` to calculate predicted labels for the data in `XTest` with `LR_PredictLabels`.

See the functions `RunLR` in the solution code.

(c) **[5 Points]** Report the number of misclassified examples in the test set.

There are 12 misclassified examples in the test set.

(d) **[5 Points]** Plot the value of the objective function on each iteration of stochastic gradient ascent, with the iteration number on the horizontal axis and the objective value on the vertical axis. Make sure to include axis labels and a title for your plot. Report the number of iterations that are required for the algorithm to converge.

See Figure **??**. The algorithm converges after 393 iterations.

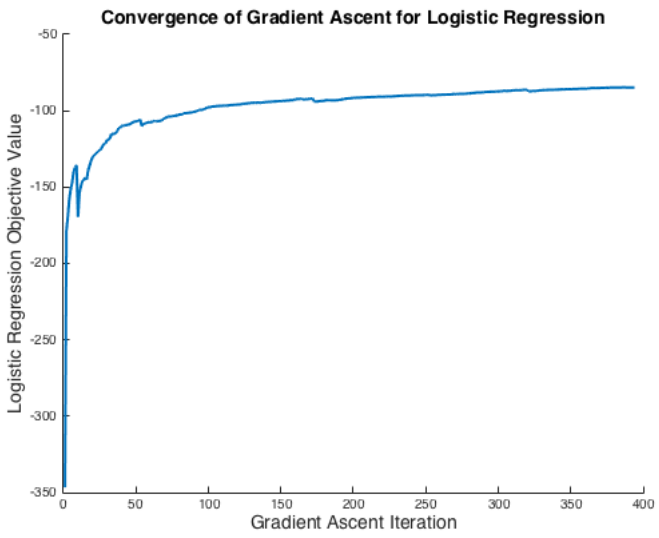

Figure 1: Solution to Problem 2(d)

(e) **[10 Points]** Next, you will evaluate how the training and test error change as the training set size increases. For each value of k in the set $\{10, 20, 30, \ldots, 480, 490, 500\}$, first choose a random subset of the training data of size k using the following code:

$$\texttt{subsetInds} = \texttt{randperm}(\texttt{n}, \texttt{k})$$
$$\texttt{XTrainSubset} = \texttt{XTrain}(\texttt{subsetInds}, :)$$
$$\texttt{yTrainSubset} = \texttt{yTrain}(\texttt{subsetInds})$$

Then re-train your classifier using `XTrainSubset` and `yTrainSubset`, and use the estimated parameters to calculate the number of misclassified examples on both the training set `XTrainSubset` and `yTrainSubset` and on the original test set `XTest` and `yTest`. Finally, generate a plot with two lines: in blue, plot the value of the training error against k, and in red, plot the value of the test error against k, where the error should be on the vertical axis and training set size should be on the horizontal axis. Make sure to include a legend in your plot to label the two lines. Describe what happens to the training and test error as the training set size increases, and provide an explanation for why this behavior occurs.

See Figure **??**. As the training set size increases, the test error decreases, but the training error increases. This trend becomes more apparent when the same experiment is repeated multiple times for each k and the averages errors are plotted instead, as in Figure **??**.

The reason behind the observed trend is that the logistic regression model is often capable of perfectly classifying the training data when the train set size is small, since there is relatively less variation. Hence, the training error is close to zero. However, such a model has poor generalization capability since the estimated parameters $\mathbf{w}$ are computed off a training sample that is not truly representative of the true population from which the data is drawn. This phenomenon, where the training error is extremely low while the test error is very high is known as *overfitting*, since the model fits the training data too closely.

As the size of the training set is increased, more variation is introduced, thus better representing the true population. This usually results in the LR model not being able to fit the training data as well as before. Further, the data in itself may not be entirely linearly separable. Nevertheless, since the larger dataset provides the model with a more complete picture of the overall population, the parameters $\mathbf{w}$ learnt in this case are estimated more accurately (lower variance in the estimated parameters). This in turn results in better generalization ability, thereby reducing the error on the test dataset.

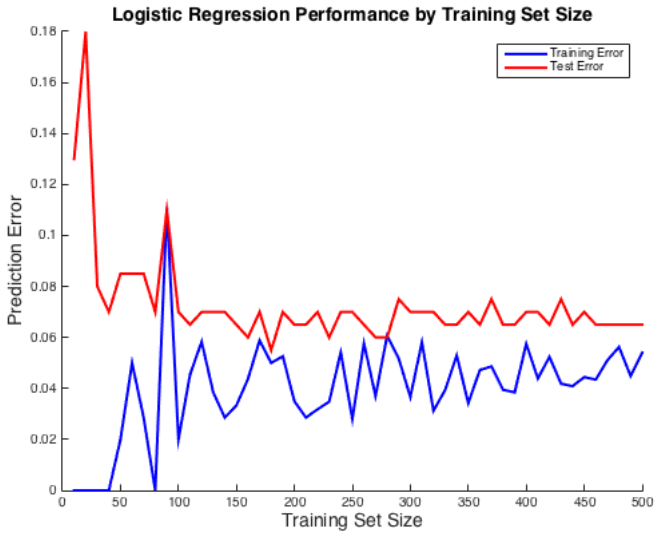

Figure 2: Solution to Problem 2(e)

Figure 3: Additional solution to Problem 2(e)