Home Quiz 1 - Logistic Regression

Due 9 a.m. Wednesday, March 20, 2019

Problem 1: Gradient Descent

In class we derived a gradient descent learning algorithm for simple linear regression where we assumed

$$y = w_0 + w_1 x_1 + \epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

In this question, you will do the same for the following model

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + \epsilon \qquad \text{where } \epsilon \sim \mathcal{N}(0, \sigma^2)$$

where your learning algorithm will estimate the parameters w_0, w_1, w_2, and w_3.

(a) Write down an expression for $P(y|x_1, x_2)$.

(b) Assume you are given a set of training observations $(x_1^{(i)}, x_2^{(i)}, y^{(i)})$ for $i = 1, \dots, n$. Write down the conditional log likelihood of this training data. Drop any constants that do not depend on the parameters $w_0, \dots, w_3$.

(c) Based on your answer above, write down a function $f(w_0, w_1, w_2, w_3)$ that can be *minimized* to find the desired parameter estimates.

(d) Calculate the gradient of $f(\boldsymbol{w})$ with respect to the parameter vector $\boldsymbol{w} = [w_0, w_1, w_2, w_3]^T$. Hint: The gradient can be written as

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \begin{bmatrix} \frac{\partial f(\boldsymbol{w})}{\partial w_0} & \frac{\partial f(\boldsymbol{w})}{\partial w_1} & \frac{\partial f(\boldsymbol{w})}{\partial w_2} & \frac{\partial f(\boldsymbol{w})}{\partial w_3} \end{bmatrix}^T$$

(e) Write down a gradient descent update rule for $\boldsymbol{w}$ in terms of $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$.

(f) Use the sympy python library to compute the above derivatives

Problem 2: Logistic Regression

In this question, you will implement a logistic regression classifier and apply it to a two-class classification problem. In the file of the assignment, you will find the data for this problem.

(a) 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 $x_1^{(i)}, \dots, x_p^{(i)}$, and the parameters $w_0, w_1, \dots, w_p$, where the superscript (i) denotes the sample index. This will be your objective function for gradient ascent.

(b) 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.

(c) Implement a logistic regression classifier using 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 gradient: `grad = LR_CalcGrad(XTrain,yTrain,wHat)`
- Update the parameter value: `wHat = LR_UpdateParams(wHat,grad,eta)`
- Check whether gradient ascent has converged: `hasConverged = LR_CheckConvg(oldObj,newObj,tol)`
- Complete the implementation of gradient ascent: `[wHat,objVals] = LR_GradientAscent(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
- `wHat` is a $p + 1 \times 1$ dimensional vector containing the regression parameter estimates $\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_p$
- `grad` is a $p + 1 \times 1$ dimensional vector containing the value of the 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 = 0.01`
- `obj`, `oldObj` and `newObj` are values of the objective function
- `tol` is the convergence tolerance, which you should set to `tol = 0.001`
- `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_GradientAscent` function, you should use the helper functions `LR_CalcObj`, `LR_CalcGrad`, `LR_UpdateParams`, and `LR_CheckConvg`.

(d) Train your logistic regression classifier on the data provided in `XTrain` and `yTrain` with `LR_GradientAscent`, and then use your estimated parameters `wHat` to calculate predicted labels for the data in `XTest` with `LR_PredictLabels`.

(e) Report the number of misclassified examples in the test set.

(f) Plot the value of the objective function on each iteration of gradient descent, 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.

(g) 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:

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

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, pot 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.

(h) Based on the logistic regression formula you learned in class, derive the analytical expression for the decision boundary of the classifier in terms of $w_0, w_1, \ldots, w_p$ and $x_1, \ldots, x_p$. What can you say about the shape of the decision boundary?

(i) In this part, you will plot the decision boundary produced by your classifier. First, create a two-dimensional scatter plot of your test data by choosing the two features that have highest absolute weight in your estimated parameters `wHat` (let's call them features j and k), and plotting the j-th dimension stored in `XTest(:,j)` on the horizontal axis and the k-th dimension stored in `XTest(:,k)` on the vertical axis. Color each point on the plot so that examples with true label $y = 1$ are shown in blue and label $y = 0$ are shown in red. Next, using the formula that you derived in part (h), plot the decision boundary of your classifier in black on the same figure, again considering only dimensions j and k.