10-601 Machine Learning: Homework 4

Due 5 p.m. Monday, February 16, 2015

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that. You *must* turn in at least $n-1$ of the n homeworks to pass the class, even if for zero credit.
- **Collaboration policy:** Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.
- **Online submission:** You must submit your solutions online on autolab. We recommend that you use LaTeX to type your solutions to the written questions, but we will accept scanned solutions as well. On the Homework 4 autolab page, you can download the template, which is a tar archive containing a blank placeholder pdf for the written questions and one Octave source file for each of the programming questions. Replace each pdf file with one that contains your solutions to the written questions and fill in each of the Octave source files with your code. When you are ready to submit, create a new tar archive of the top-level directory and submit your archived solutions online by clicking the "Submit File" button. You should submit a single tar archive identical to the template, except with each of the Octave source files filled in and with the blank pdf replaced by your solutions for the written questions. You are free to submit as many times as you like (which is useful since you can see the autograder feedback immediately).

  DO NOT change the name of any of the files or folders in the submission template. In other words, your submitted files should have exactly the same names as those in the submission template. Do not modify the directory structure.

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) **[8 Points]** Write down an expression for $P(y|x_1, x_2)$.

Solution

$$P(y|x_1, x_2) = \mathcal{N}(f(x_1, x_2; W)|\sigma^2) = \mathcal{N}(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2|\sigma^2)$$

(b) **[8 Points]** Assume you are given a set of training observations $(x_1^{(i)}, x_2^{(i)}, y^{(i)})$ for $i = 1, \ldots, n$. Write down the conditional log likelihood of this training data. Drop any constants that do not depend on the parameters $w_0, \ldots, w_3$.

Solution

$$\underset{W}{\text{argmax}} \prod_i P(y|x_1, x_2; W) = \underset{W}{\text{argmax}} \sum_i \log P(y|x_1, x_2; W) = \underset{W}{\text{argmax}} \sum_i \log \mathcal{N}(f(x_1, x_2; W)|\sigma^2)$$

$$= \underset{W}{\text{argmax}} \sum_i \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2}\left(\frac{y^i - f(x_1^i, x_2^i; W)}{\sigma}\right)^2$$

$$= \underset{W}{\text{argmax}} \sum_i \left(y^i - f(x_1^i, x_2^i; W)\right)^2$$

(c) **[8 Points]** 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.

Solution

We can convert the maximization to a minimization by taking the negative. But because conditional log likelihood is quadratic, the negative sign disappears.

$$f(\boldsymbol{w}) = \underset{W}{\text{argmin}} \sum_i \left(y^i - (w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2)\right)^2$$

(d) **[8 Points]** 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

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

Solution

$$\frac{\partial f(\boldsymbol{w})}{\partial w_0} = \sum_i 2(f(x_1, x_2; W) - y^i)$$

$$\frac{\partial f(\boldsymbol{w})}{\partial w_1} = \sum_i 2x_1(f(x_1, x_2; W) - y^i)$$

$$\frac{\partial f(\boldsymbol{w})}{\partial w_2} = \sum_i 2x_2(f(x_1, x_2; W) - y^i)$$

$$\frac{\partial f(\boldsymbol{w})}{\partial w_3} = \sum_i 4x_1(f(x_1, x_2; W) - y^i)$$

(kev:) check

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

(e) **[8 Points]** Write down a gradient descent update rule for $\boldsymbol{w}$ in terms of $\nabla_{\boldsymbol{w}} f(\boldsymbol{w})$.

Solution

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla_{\boldsymbol{w}} f(\boldsymbol{w})$$

Problem 2: Logistic Regression

In this question, you will implement a logistic regression classifier and apply it to a two-class classification problem. As in the last assignment, your code will be autograded by a series of test scripts that run on Autolab. To get started, download the template for Homework 4 from the Autolab 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 `HW4Data.mat` that contains the data for this problem. You can load the data into Octave by executing `load("HW4Data.mat")` in the Octave interpreter. Make sure not to modify any of the function headers that are provided.

(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.

Solution

$$L(w) = \sum_{i=1}^{n} y^{(i)} \ln \sigma(w^T x^{(i)}) + (1 - y^{(i)}) \ln(1 - \sigma(w^T x^{(i)})))$$

(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.

Solution

We use the following derivatives:

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)), \qquad \frac{\partial w^T x^{(i)}}{\partial w} = x^{(i)}, \qquad \frac{\partial f(w)}{\partial w^T} = \frac{\partial f(w)^T}{\partial w}$$

Therefore the gradient is:

$$\frac{\partial L(w)}{\partial w} = \sum_i \left(y^{(i)} - \sigma(w^T x^{(i)}) \right) x^{(i)}$$

Letting $\mu_i \equiv \sigma(w^T x^{(i)})$, we have:

$$\frac{\partial L(w)}{\partial w} = X^T (y - \mu)$$

Note that here we have wrapped the bias term term into w by appending a 1 to the start of x.

(c) **[30 Points]** 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) **[0 Points]** 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) **[5 Points]** Report the number of misclassified examples in the test set.

Solution

There are 13 misclassified examples in the test set.

(f) **[5 Points]** 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.

Solution

Converges in 88 iterations.

(g) **[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, 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.

Solution

As the training set size increases, the training error increases and the test error decreases. This happens because it is easy for the model to overfit to small datasets, but this causes poor generalization. In contrast, as the training set size increases, it becomes harder for the model to fit it, so training error increases, but the testing error decreases because we can fit the target function better.

Extra Credit:

(h) **[5 Points]** 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?

Solution

We can write the decision rule as follows:

$$\hat{y} = \begin{cases} 0, & P(Y = 1 \mid X) < P(Y = 0 \mid X) \\ 1, & P(Y = 1 \mid X) > P(Y = 0 \mid X) \end{cases}$$

Thus, we predict class 1 if:

$$\sigma(w^T x) > 0.5$$

.

After some algebraic manipulation, we get:

$$0 < w^T x$$

Therefore we can rewrite the decision rule as:

$$\hat{y} = \begin{cases} 0, & 0 > w^T x \\ 1, & 0 < w^T x \end{cases}$$

Therefore the decision boundary is linear in w.

(i) **[5 Points]** 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.