CS189: Introduction to Machine Learning

Homework 3

Due: October 6, 2015 @ 11:59PM

Homework party: October 1, 8-10pm (Wozniak Lounge).

Submission: **bCourses** (no Kaggle, no Gradescope)

Submission Instructions

In your submission, include two separate files:

1. A pdf writeup with answers to all the questions and your plots. Include in the pdf a copy of your code for each problem (code for problems 1, 2, 3).
2. A zip archive containing your code for each problem, and a README with instructions on how to run your code.

Submit **2 separate files to bCourses: a pdf and a zip of your code**.

Problem 1: Linear Regression

In this problem we will try to predict the median home value in a given Census area by using linear regression. The data is in `housing_data.mat`, and it comes from `http://lib.stat.cmu.edu/datasets/`. (`houses.zip`).There are only 8 features for each data point; you can read about the features in `housing_data_source.txt`.

1. Implement a linear regression model with least squares. Include your code in the submission.

   You should add a constant term to the training data (e.g. add another dimension to each data point, with the value of 1). This is same as adding the bias term to linear regression (see discussion 4 question 1). Note that each data point $\mathbf{x}^{(i)^T}$ is a row of the training data matrix X.

   Solution: In the code there should be some equation close to the form:

   $$X^T X\mathbf{w} = X^T\mathbf{y} \text{ or } \mathbf{w} = (X^T X)^{-1}X^T\mathbf{y}$$

   Although it would be weird if you did this, other valid solutions include linear programming and gradient descent.

2. Test your trained model on the validation set. What is the residual sum of squares (RSS) on the validation set? What is the range of predicted values (min, max)? Do they make sense?

Solution:

$$RSS = 5.7950e12.$$

The range of predicted values is $-5.6563e04$ to $7.1080e05$. They don't really make sense because negative values are predicted.

3. Plot the regression coefficients $\boldsymbol{w}$ (plot the value of each coefficient against the index of the coefficient). Be sure to exclude the coefficient corresponding to the constant offset you added earlier.

Solution: Plot should show that the 1st, 7th, and 8th coefficients are the most significant. There should not be a 9th coefficient that dominates all (this would be the bias coefficient).

4. Plot a histogram of the residuals of the training data (the residual corresponding to point i is $f(\mathbf{x}^{(i)}) - y^{(i)}$). What distribution does this resemble?

Solution: Normal distribution.

NOTE: You may not use any library routine for linear regression or least squares solving. You may use any other linear algebra routines.

Problem 2: Logistic Regression

Let $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$ be a training set, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ and $y^{(i)} \in \{-1, 1\}$. Recall that the loss function for logistic regression is the cross-entropy. Therefore our risk is:

$$R[\mathbf{w}] = \sum_{i=1}^{n} \log(1 + e^{-z^{(i)}})$$

where, $z^{(i)} = y^{(i)} f(\mathbf{x}^{(i)})$, and $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$.

In this problem, you will minimize the cross-entropy risk (also known as the *negative log likelihood* of $\mathbf{w}$) on a small training set. We have four data points in $\mathbb{R}^2$, two of class 1, and two of class -1. Here is the data (you may want to draw this on paper to see what it looks like):

$$X = \begin{bmatrix} 0 & 3 \\ 1 & 3 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

Here, X is the training data matrix; each row $\mathbf{x}^{(i)^T}$ of X is a transposed data point.

Notice that the data cannot be separated by a boundary that goes through the origin. To account for this, you should *append* 1 to the $\mathbf{x}^{(i)}$ vectors and fit a three-dimensional $\mathbf{w}$ vector that includes an offset term.

1. Derive the gradient of the cross-entropy risk with respect to $\mathbf{w}$. Show your work. Your answer should be a matrix-vector expression. Do NOT write your answer in terms of the individual components of the gradient.

   For notation, you may let $diag(\mathbf{v})$ denote the square matrix with components of vector $\mathbf{v}$ on the diagonal.

   Hint: You may use $diag(\mathbf{y})X$. For notational purposes you may have in your answer a matrix A, where you define $A_{i,j}$ as some function of some other matrix's i, jth component.

   Note: We write these updates as matrix operations instead of for-loops because it's cleaner code and allows you to use optimized linear algebra routines (these routines are implemented with cache-aware multithreading in C/Fortran - which you'd rather not implement yourself!).

Solution:

$$\begin{aligned}
\nabla_{\mathbf{w}} R &= \sum_{i=1}^{n} (-y^{(i)} \mathbf{x}^{(i)}) \frac{exp(-y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)})}{1 + exp(-y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)})} \\
&= \sum_{i=1}^{n} (-y^{(i)} \mathbf{x}^{(i)}) \frac{1}{1 + exp(y^{(i)} \mathbf{w}^T \mathbf{x}^{(i)})} \\
&= -(diag(\mathbf{y})X)^T \frac{1}{1 + exp(diag(\mathbf{y})X\mathbf{w})} \\
&= -\big[diag(\mathbf{y})X\big]^T S(diag(\mathbf{y})X\mathbf{w}), \text{ where } S(\mathbf{z}) = \frac{1}{1 + exp(-\mathbf{z})}
\end{aligned}$$

2. In general, to verify that the function we minimize is convex we will want to show that the Hessian is positive semidefinite. For now, just derive the Hessian of the risk. Show your work; your answer should be a (somewhat complicated) matrix-vector expression.

Solution:

$$\mathcal{H}_{\mathbf{w}} R = \sum_{i=1}^{n} (y^{(i)}\mathbf{x}^{(i)}) \frac{exp(y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)})}{[1+exp(y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)})]^2} (y^{(i)}\mathbf{x}^{(i)})^T$$

$$= (diag(\mathbf{y})X)^T \frac{exp(diag(\mathbf{y})diag(X\mathbf{w}))}{[1+exp(diag(\mathbf{y})diag(X\mathbf{w}))]^2} diag(\mathbf{y})X$$

where $\frac{exp(diag(\mathbf{y})diag(X\mathbf{w}))}{[1+exp(diag(\mathbf{y})diag(X\mathbf{w}))]^2}$ is evaluated component-wise.

Note this is a $d \times d$ matrix, as expected.

The order matters! The component-wise exponential terms must be in the middle of the matrix expression.

3. We will now perform gradient descent for a few iterations. Set the learning rate (η) as 1. We are given that $\mathbf{w}^{(0)} = \begin{bmatrix} -2 & 1 & 0 \end{bmatrix}^T$. $\mathbf{w}^{(0)}$ is the value of $\mathbf{w}$ at the 0^{th} iteration.

   (a) State the value of $\boldsymbol{\mu}^{(0)}$. This should be an $n-$dimensional vector, with $\mu_i^{(0)} = P(Y = 1|X = \mathbf{x}^{(i)})$.

   Solution:

   $$\boldsymbol{\mu}^{(0)} = \begin{bmatrix} 0.9526 \\ 0.7311 \\ 0.7311 \\ 0.2689 \end{bmatrix}$$

   (b) State the value of $\mathbf{w}^{(1)}$ (the value of $\mathbf{w}$ after one iteration).

   Solution:

   $$\mathbf{w}^{(1)} = \begin{bmatrix} -2.0000 \\ 0.9491 \\ -0.6836 \end{bmatrix}$$

   (c) State the value of $\boldsymbol{\mu}^{(1)}$.

   Solution:

   $$\boldsymbol{\mu}^{(1)} = \begin{bmatrix} 0.8969 \\ 0.5408 \\ 0.5660 \\ 0.1500 \end{bmatrix}$$

(d) After performing a second iteration, state the value of $\mathbf{w}^{(2)}$.

Solution:

$$\mathbf{w}^{(2)} = \begin{bmatrix} -1.6908 \\ 1.9198 \\ -0.8374 \end{bmatrix}$$

Problem 3: Spam classification using Logistic Regression

The spam dataset given to you as part of the homework in `spam.mat` consists of 4601 email messages, from which 57 features have been extracted as follows:

- 48 features giving the proportion (0 to 1) of words in a given message which match a given word on the list. The list contains words such as business, free, george, etc. (The data was collected by George Forman, so his name occurs quite a lot!)
- 6 features giving the proportion (0 - 1) of characters in the email that match a given character on the list. The characters are ; ([! $ # .
- Feature 55: The average length of an uninterrupted sequence of capital letters
- Feature 56: The length of the longest uninterrupted sequence of capital letters
- Feature 57: The sum of the lengths of uninterrupted sequences of capital letters

The dataset consists of a training set size 3450 and a test set of size 1151. One can imagine performing several kinds of preprocessing to this data matrix. Try each of the following separately:

i) Standardize each column so they each have mean 0 and unit variance.

ii) Transform the features using $x_j^{(i)} \leftarrow log(x_j^{(i)} + 0.1)$.

iii) Binarize the features using $x_j^{(i)} \leftarrow \mathbb{I}(x_j^{(i)} > 0)$. $\mathbb{I}$ denotes an indicator variable.

Note: You will need to tune the step size carefully to avoid numerical issues and to avoid a diverging training risk.

1. Implement logistic regression to classify the spam data. Use batch gradient *descent*.

   Plot the training risk (the cross-entropy risk of the training set) vs. the number of iterations. You should have one plot for each preprocessing method.

   Note: One batch gradient descent iteration amounts to scanning through the whole training data and computing the full gradient.

Solution: Plot should be smooth and strictly decreasing. The risk should converge.

2. Derive stochastic gradient *descent* equations for logistic regression and show your steps. Plot the training risk vs. number of iterations. You should have one plot for each preprocessing method. How are the plots different from (1)?

   Note: One stochastic gradient descent iteration amounts to computing the gradient using one data point.

Solution:

$$\nabla_{\mathbf{w}} L = -S(-z^{(i)})y^{(i)}\mathbf{x}^{(i)}, \text{ where } S(z) = \frac{1}{1+exp(-z)}$$

Plot should be rough. Risk should not converge (risk should oscillate within a band of values at the end).

3. Instead of a constant learning rate (η), repeat (2) where the learning rate decreases as $\eta \propto 1/t$ for the t^{th} iteration. Plot the training risk vs number of iterations. Is this strategy better than having a constant η? You should have one plot for each preprocessing method.

Solution: This is better, as it solves the problem of the risk oscillating within a final band of values. The risk should now converge.

4. (a) Now let's use kernel logistic regression with a polynomial kernel of degree 2. Our risk is still the same as in problem 2, but our classifier is now:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}^{(i)}, \mathbf{x}) \text{ where } K(\mathbf{x}^{(i)}, \mathbf{x}) = (\mathbf{x}^T\mathbf{x}^{(i)} + 1)^2$$

   instead of $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ as it was originally. Show that the stochastic gradient *descent* update for data point $\mathbf{x}^{(i)}$ is:

$$\alpha_i \leftarrow \alpha_i + \eta S(-z^{(i)})y^{(i)}$$

$$\text{Where } z^{(i)} = y^{(i)} f(\mathbf{x}^{(i)}), S(z^{(i)}) = \frac{1}{1+exp(-z^{(i)})}$$

   Note that this means $\boldsymbol{\alpha} \in \mathbb{R}^n$. Also derive $\frac{\partial L}{\partial \alpha_i}$ directly from the loss function. You should get a different answer than the dual algorithm shown above. Under what condition are the two updates you derived the same?

Solution:

$$\frac{\partial L}{\partial \alpha_i} = -S(-z^{(i)})y^{(i)}K(\mathbf{x}^{(i)}, \mathbf{x}^{(i)})$$

The two updates are the same when $K(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}) = 1$. This is true for the Gaussian (RBF) kernel, but not for polynomial kernels, for example.

(b) Finally, repeat (2), using the best preprocessing method you found, using kernel logistic *ridge* regression. Use whichever learning rate scheme you wish. Use $\gamma = 10^{-5}$. You may optionally adjust the value of γ. Generate a plot of training risk vs. number of iterations, and another plot of validation risk vs. number of iterations (use a 2/3, 1/3 split). Use the following update equations (you do not need to derive them):
Hint: if you are getting numerical issues, feel free to clamp the values of z.

$$\alpha_i \leftarrow \alpha_i - \gamma\alpha_i + \eta S(-z^{(i)})y^{(i)}$$
$$\alpha_h \leftarrow \alpha_h - \gamma\alpha_h \text{ for } h \neq i$$

Solution: It was fairly difficult/finnicky to get well-behaved plots. A jagged plot without clear convergence is fine, so long as the risk doesn't wildly increase.

(c) Repeat the same experiment with the linear kernel $K(\mathbf{x}^{(i)}, \mathbf{x}) = \mathbf{x}^T\mathbf{x}^{(i)} + 1$. Does the quadratic kernel overfit the data? For each kernel, should you decrease or increase γ to try to improve performance?

Solution: Neither kernel overfits the data, so you could try *decreasing* γ on both to increase performance.

NOTE: You are NOT supposed to use any kind of software package for logistic regression!

Problem 4: Real World Spam Classification

Motivation: After taking CS 189, students should be able to wrestle with "real world" data and problems. These issues might be deeply technical and require theoretical background, or might demand specific domain knowledge. Here is an example that a past TA encountered.

Daniel recently interned as an anti-spam product manager for an email service provider. His company uses a linear SVM to predict whether an incoming spam message is spam or ham. He notices that the number of spam messages received tends to spike upwards a few minutes before and after midnight. Eager to obtain a return offer, he adds the timestamp of the received message, stored as number of milliseconds since the previous midnight, to each feature vector for the SVM to train on, in hopes that the ML model will identify the abnormal spike in spam volume at night. To his dismay, after testing with the new feature, Daniel discovers that the linear SVM's success rate barely improves.

Why can't the linear SVM utilize the new feature well, and what can Daniel do to improve his results? Daniel is unfortunately limited to only a quadratic kernel. This is an actual interview question Daniel received for a machine learning engineering position!

Write a short explanation. This question is open ended and there can be many correct answers.

Solution:
Original solution: Daniel unfortunately made a mistake by storing the timestamp as the number of milliseconds since the previous midnight. This raw format would have worked perfectly fine if the spam spike was at a different time of day (say 5pm) since the magnitude of the time stamps would be relatively similar. Thus, identifying the spike would be simple for a linear decision boundary. However, since the spike is at midnight, the magnitude will vary widely since 11:59pm is 86340000ms while 12:00am is 0ms. Essentially, time is stored in a modular fashion and the linear decision boundary with no regularization and kernel will do poorly.

However, we can provide our own transformations to the feature. Take the timestamp in milliseconds since the previous midnight and normalize it to range uniformly between $\theta = [0, 2\pi]$. Now, use theta to generate 2 coordinates $X = cos(\theta)$, $Y = sin(\theta)$. Notice that we have mapped our timestamp into a unit circle, similar to how an analog clock behaves. Note that in this format, 11:59pm and 12:00am are close if we compare the L2 norm between their two points on the unit circle! Now, we can apply a linear SVM and predict when the spam spike will be.

Other ways to improve performance:

1. Map times from 12am to 12pm to [-1, 1], then use a quadratic kernel.
2. Add 15 minutes to each timestamp, mod 24 hours, then use a linear SVM.
3. Hack together a binary feature indicating whether a timestamp is "close to midnight" or not, then use linear SVM.