10-315 Intro to Machine Learning HW2

INSTRUCTIONS

- **Due: Tuesday, 4 February 2020 at 11:59 PM EDT.**
- **Format:** Complete this pdf with your work and answers. Whether you edit the latex source, use a pdf annotator, or hand write / scan, make sure that your answers (tex'ed, typed, or handwritten) are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.
- **How to submit:** Submit a pdf with your answers on Gradescope. Log in and click on our class 10-315, click on the appropriate *Written* assignment, and upload your pdf containing your answers. Don't forget to submit the associated *Programming* component on Gradescope if there is any programming required.
- **Policy:** See the course website for homework policies and Academic Integrity.

Name	
Andrew ID	
Hours to complete (both written and programming)?	

For staff use only

Q1	Q2	Q3	Total
/15	/10	/ 35	/ 60

Q1. [15 pts] Linear Regression with More Weights

In our original model for linear least squares regression, we made the assumption that each data point provided equally precise information in determining our estimate for the output. We do this often with the assumption of *homoskedasticity*, meaning that we assume all samples are drawn from a population with constant variance. However, this may not always be the case. For example, if we were trying to predict net worth from age, we would probably see a greater variance as age increases. In situations like this, when it may not be reasonable to assume that every observation should be treated equally, weighted least squares can often be used to maximize the efficiency of parameter estimation. This is done by attempting to give each data point its proper amount of influence over the parameter estimates.

With this in mind, consider observing a data set $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$ where $\mathbf{x}^{(n)} \in \mathbb{R}^M$ denotes the n-th example, and $y^{(n)}$ denotes the target response value. Assume that each data point $(\mathbf{x}^{(n)}, y^{(n)})$ comes with a weighting factor $r_n > 0$, in which case our error function becomes:

$$J_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} r_n (y^{(n)} - \mathbf{w}^T\mathbf{x}^{(n)})^2$$

We can also write this with linear algebra notation without the summation, in which case we get:

$$J_{\mathcal{D}}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T \boldsymbol{R}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

Here, $\mathbf{w}$ is the weight matrix denoting the model parameters for our linear model, $\boldsymbol{R}$ is a diagonal matrix where $R_{n,n} = r_n$, and $\mathbf{X}$ is the design matrix where the n-th row contains the features of $x^{(n)}$.

(a) [15 pts] Derive the solution $\mathbf{w}^*$ that minimizes the weighted sum of squares error defined above. You must show your work. Reasonable steps should be taken between lines in the derivation. Providing an explanation per line is optional.

Do NOT use summations, and instead use a linear algebra representation. Also, make sure to define the dimensions of any variables you use.

$\mathbf{w}^*$:

Q2. [10 pts] Linear Regression on MNIST

Suppose we would like to perform some analysis on the MNIST dataset, a dataset consisting of images of digits in the range (0-9). Given an image, we want to use the linear regression to predict the image it represents.

Putting this in the regression framework, we have some dataset $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$ where each $\mathbf{x}^{(n)} \in \mathbb{R}^M$ is a feature vector corresponding to the n-th image (you could think of the pixels being laid out in row-major format to form a vector) and each $\mathbf{y}^{(n)} \in \mathbb{R}^K$ is a vector denoting the digit that the image contains using the one-of-K encoding scheme. In this scheme, we can represent a categorical variable taking on up to K different values as a vector of size K, where value number k corresponds to the vector having a 1 at position k and zeros otherwise.

(a) [2 pts] Given the description above, and assuming we are using MSE as our loss function, write the objective function $J_{\mathcal{D}}$.

Do NOT use summations, and instead use the linear algebra form similar to the previous problem. Also, make sure to define the dimensions of any variables you use.

Hint: You will need to use the Frobenius norm.

$J(\mathbf{W})$:

(b) [4 pts] Using the objective function from part a, derive the gradient of the objective function with respect to the weights. Recall that:

$$\nabla J(\boldsymbol{W}) = \frac{\partial J}{\partial \boldsymbol{W}} = \begin{pmatrix} \frac{\partial J}{\partial W_{1,1}} & \frac{\partial J}{\partial W_{1,2}} & \cdots & \frac{\partial J}{\partial W_{1,K}} \\ \frac{\partial J}{\partial W_{2,1}} & \frac{\partial J}{\partial W_{2,2}} & \cdots & \frac{\partial J}{\partial W_{2,K}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J}{\partial W_{M,1}} & \frac{\partial J}{\partial W_{M,2}} & \cdots & \frac{\partial J}{\partial W_{M,K}} \end{pmatrix} \tag{1}$$

Do NOT use summations in your final answer, and instead use a linear algebra representation. Also, make sure to define the dimensions of any variables you use.

$\nabla J(\boldsymbol{W})$:

(c) [4 pts] Finally, provide the closed form solution for the weights.

Do NOT use summations in your final answer, and instead, use a linear algebra representation. Also, make sure to define the dimensions of any variables you use.

$\mathbf{W}^*$**:**

Q3. [35 pts] Programming

(a) [3 pts] Programming Q1

Why didn't we give you a test set? Write an explanation other than "You want to save it for the autograder on Gradescope."

Explanation:

(b) [6 pts] Programming Q2

Include your plot of data and closed form hypothesis function line. Running `python3.6 autograder.py -q Q2` should have auto generated this image and saved it to the `figures` directory within your code directory.

Plot:

Training MSE:	**Validation MSE:**

(c) [2 pts] Programming Q3

Include your plot of data and closed form hypothesis function line. Running `python3.6 autograder.py -q Q3` should have auto generated this image and saved it to the `figures` directory within your code directory.

Plot:

(d) [10 pts] Programming Q4

Include your plot of data and closed form hypothesis function line. Running `python3.6 autograder.py -q Q4` should have auto generated this image and saved it to the `figures` directory within your code directory.

Plot:

Training MSE:	**Validation MSE:**

Note: For this report, it is ok if these numbers come from a different training run than the plot above.

(e) [8 pts] Programming Q6

Include your confusion matrix on the training data. Running `python3.6 autograder.py -q Q6` should have printed this to stdout.

Confusion matrix:

How many times is an eight in the training set incorrectly labelled as a nine?

Number of times:

Training accuracy:

Validation accuracy:

(f) [6 pts] Programming Q7

Include your confusion matrix on the training data. Running `python3.6 autograder.py -q Q7` should have printed this to stdout.

Confusion matrix:

Training accuracy:

Validation accuracy:

Note: For this report, it is ok if these numbers come from a different training run than the confusion matrix.