EE M146 Homework 2
Introduction to Machine Learning Monday, April 8, 2019
Instructor: Lara Dolecek Due: Wednesday, April 17, 2019
TA: Zehui (Alex) Chen, Ruiyi (John) Wu

1. In class, we see that the solution of the least square problem satisfies the normal equation $X^TXw = X^Ty$, where $X \in \mathbb{R}^{N\times M}$ and $y \in \mathbb{R}^N$. Prove the following statement:

   The Gram matrix X^TX is nonsingular if and only if X has linearly independent columns.

   With $N >> M$, there will likely be M linearly independent vectors x_n. Therefore, the Gram matrix X^TX is likely to be invertible. Hint: X has linearly independent columns if $AX = 0$ only for $x = 0$. You may want to use the following properties of (non)singular matrix. The equation $Ax = 0$ has only the trivial solution $x = 0$ if and only if A is nonsingular. If A is singular, there exist $z \neq 0$ such that $Az = 0$.

2. Consider the hat matrix $H = X(X^TX)^{-1}X^T$, where $X \in \mathbb{R}^{N\times M}$ and X^TX is invertible.

   (a) Show that H is symmetric.
   (b) Show that $H^K = H$ for any positive integer K.
   (c) If I is the identity matrix of size N, show that $(I - H)^K = I - H$ for any positive integer K.
   (d) Show that $Trace(H) = M$, where the trace is the sum of diagonal elements. Hint:$Trace(AB) = Trace(BA)$.

3. Consider a linear regression problem in which we want to "weigh" different training instances differently because some of the instances are more important than others. Specifically, suppose we want to minimize

$$J(w_0, w_1) = \sum_{n=1}^{N} \alpha_n (w_0 + w_1 x_{n,1} - y_n)^2. \tag{1}$$

   Here $\alpha_n > 0$. In class we worked out what happens for the case where all weights (the $\alpha_n's$) are the same. In this problem, we will generalize some of those ideas to the weighted setting. Calculate the gradient by computing the partial derivative of J with respect to each of the parameters (w_0, w_1). Comment on how the $\alpha_n's$ affect the linear regression problem. For example, what happens if $\alpha_i = 0$ for some i? Qualitatively describe what will happen in gradient descent if α_j is much greater than other $\alpha_{j'}, j' \neq j$.

4. Consider the modified objective function of regularized logistic regression:

$$J(w) = -\sum_{n=1}^{N} [y_n \log h_w(x_n) + (1 - y_n) \log(1 - h_w(x_n))] + \frac{1}{2}\sum_i w_i^2 \tag{2}$$

   where $h_w(x) = \sigma(w^Tx)$ and the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$. Find the partial derivative $\frac{\partial J}{\partial w_j}$ and derive the gradient descent update rules for the weights.

5. In class we have seen the probabilistic interpretation of the logistic regression objective, and the derivation of the gradient descent rule for maximizing the conditional likelihood. We have this maximum likelihood (ML) formulation for $w \in \mathbb{R}^m$:

$$w^* = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w). \tag{3}$$

To prevent overfitting, we want the weights to be small. To achieve this, we consider some prior distribution of the weights $f(w)$ and use maximum a posteriori (MAP) estimation:

$$w^* = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w) f(w). \tag{4}$$

Assuming a standard Gaussian prior $\mathcal{N}(0, I)$ for the weight vector, show that the MAP estimation is equivalent to minimizing the logistic regression objective with L2 regularization as shown in the previous question. Note: For $Z \sim \mathcal{N}(0, I_m)$,

$$f_Z(z) = \frac{1}{(2\pi)^{\frac{m}{2}}} \exp\left(-\sum_{i=1}^{m} \frac{z_i^2}{2}\right).$$

6. In this exercise, you will work through linear and polynomial regression. Our data consists of inputs $x_n \in \mathbb{R}$ and outputs $y_n \in \mathbb{R}, n \in \{1, \cdots, N\}$, which are related through a target function $f(x)$. Your goal is to learn a predictor $h_w(x)$ that best approximates $f(x)$.

   (a) **Visualization** We provide you two sets of data, the training data and the testing data in the two files, *regression_train.csv* and *regression_test.csv*. In each file, the first column is the input and the second column is the output. In MATLAB, visualize the training data by plotting the input v.s. the output. What do you observe? For example, can you make an educated guess on the effectiveness of linear regression in predicting the data?

   (b) **Linear Regression: closed-form solution** Let us start by considering a simple linear regression model:

$$h_w(x) = w^T x = w_0 + w_1 x.$$

Recall that linear regression attempts to minimize the objective function

$$J(w) = \sum_{n=1}^{N} (h_w(x_n) - y_n)^2 = \|Xw - y\|^2,$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}, X = \begin{bmatrix} 1, x_1 \\ 1, x_2 \\ \vdots, \vdots \\ 1, x_n \end{bmatrix}, w = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}.$$

Note that to take into account the intercept term w_0, we can add an additional "feature" to each instance and set it to one, e.g., $x_{i,0} = 1$. This is equivalent to adding an additional first column to X and setting it to all ones.

In class we learned that the closed-form solution to linear regression is:

$$w^* = (X^T X)^{-1} X^T y.$$

Implement this closed-form solution in MATLAB using the training data and report w and $J(w)$. Also generate a plot depicting your training data and the fitted line.

(c) **Linear Regression: gradient descent** Another way to solve linear regression is through gradient descent (GD). In gradient descent, each iteration performs the following update:

$$w_j := w_j - \eta \sum_{n=1}^{N} (h_w(x_n) - y_n) x_{n,j} \quad \text{(simultaneously update } w_j \text{ for all } j\text{)}.$$

With each iteration of gradient descent, we expect our updated parameters w to come closer to the optimal w^* and therefore achieve a lower value of $J(w)$.

Implement the gradient descent algorithm in MATLAB using all of the following specifications for the gradient descent algorithm:

- Initialize w to be the all 0s vector.
- Run the algorithm for 10000 iterations.
- Terminate the algorithm earlier if the value of $J(w)$ is unchanged across consecutive iterations. In this exercise, we say $J(w)$ is unchanged if $|J(w)_{t-1} - J(w)_t| < 0.0001$.
- Use a fixed learning rate η.

For different $\eta = 0.0407, 0.01, 0.001, 0.0001$, report the final $J(w)$ and the number of iterations until convergence (the number will be 10000 if the algorithm did not converge within 10000 iterations). Discuss how does the learning rate η affect the GD algorithm.

(d) **Gradient Descent Visualization** Repeat the exercise in (c) with only $\eta = 0.0407$ for only 40 iterations, with w initialized to the all 0s vector. Report w, $J(w)$ and plot the fitted line (along with the data) for iteration 0, 10, 20, 30 and 40. Compare your fitted lines and $J(w)$(s) to what you get in (b). What do you observe over different iterations?

(e) **Polynomial Regression** Next let us consider the more complicated case of polynomial regression, where our hypothesis is

$$h_w(x) = w^T \phi(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_m x^m.$$

Note that the function is linear in the coefficient w. We can therefore extend the

result of linear regression by replacing the input matrix X with

$$\Phi = \begin{bmatrix} \phi(x_1)^T \\ \phi(x_2)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

where $\phi(x)$ is a function such that $\phi_j(x) = x^j$ for $j = 0, \cdots, m$.

For $m = 0, \cdots, 10$, use the closed-form solution to determine the best-fit polynomial regression model on the training data. With this model, calculate the RMSE (Root-Mean-Square error),i.e., $E_{RMS} = \sqrt{J(w)/N}$ where N is the number of data points, on both the training data and the test data. Generate a plot depicting how RMSE (both training and testing) varies with model complexity (polynomial degree m). Which degree of polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your plot to justify you answer.