Problem Set #3

ECE 4424 / CS 4824 - Machine learning
Virginia Tech

October 11, 2020

- Feel free to collaborate with other classmates in doing the homework. Please indicate your collaborators with their student ID. You should, however, write down your solution yourself. Please try to keep the answers brief and clear.
- Whenever you need clarification, please post the related questions on Piazza under the corresponding homework folder.
- Total: 100 points + 10 bonus points
- **Due date: 10/24/2020, 11:59PM ET**
- Late submission: each student will have a total of **four** free late (calendar) days to use for homeworks. **Note: this is the total number of late days accumulated through all the homeworks so far. It's NOT reset after each homework!** Once these late days are exhausted, any assignments turned in late will be penalized 20% per late day. However, no assignment will be accepted more than three days after its due date. Each 24 hours or part thereof that a homework is late uses up one full late day.

1 Convex optimization basics

Question 1.1 (10 points): Is the matrix $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ positive semidefinite? If yes, prove it; if no, provide a counter example.

Question 1.2 (20 points): Recall that in the example of spam filtering discussed in Lecture 10 and 11, to calculate the maximum likelihood estimation for Naive Bayes, we first derive the log-likelihood function:

$$f(\theta) = n_1 \log \theta_{y=1} + (n - n_1) \log(1 - \theta_{y=1}) + \sum_{j=1}^{d} \left(r_0^{(j)} \log \theta_{j|0} + (n - n_1 - r_0^{(j)}) \log(1 - \theta_{j|0}) + r_1^{(j)} \log \theta_{j|1} + (n_1 - r_1^{(j)}) \log(1 - \theta_{j|1}) \right)$$

where $n_1 = \sum_{i=1}^{n} y_i$ is the number of spam emails (label 1), $r_0^{(j)} = \sum_{i=1}^{n} x_i^{(j)} \mathbb{1}(y_i = 0)$ (for every word $j \in \{1, ..., d\}$) is number of non-spam emails that contain word j, $r_1^{(j)} = \sum_{i=1}^{n} x_i^{(j)} \mathbb{1}(y_i = 1)$ (for every word $j \in \{1, ..., d\}$) is number of spam emails that contain word j.

The parameters of Naive Bayes θ include $\theta_{j|1}$ for every $j \in \{1, ..., d\}$ (i.e., probability that word j is present given a spam email), $\theta_{j|0}$ for every $j \in \{1, ..., d\}$ (i.e., probability that word j is present given a non-spam email), and $\theta_{y=1}$ (i.e., the prior probability of a spam email).

a) (10 pts) Explain why the Hessian of the function is a diagonal matrix, i.e., $\frac{\partial^2}{\partial \theta_p \theta_q} f(\theta) = 0$ for any two different parameters θ_p and θ_q. For example, $\frac{\partial^2}{\partial \theta_{j|1} \partial \theta_{y=1}} f(\theta) = 0$ for any $j \in \{1, ..., d\}$, and $\frac{\partial^2}{\partial \theta_{j|0} \partial \theta_{j|1}} f(\theta) = 0$ for any $j \in \{1, ..., d\}$. Hint: write out the partial derivative of the function with respect to each parameter, and observe if it depends on the other parameters.

b) (10 pts) Show that the **negative log-likelihood function**, $-f(\theta)$ is convex. Hint: Calculate the second-order derivative of the negative log-likelihood for each parameter and show that it is non-negative. Use the fact that the Hessian is diagonal to make the case that it is positive semidefinite.

Question 1.3 (20 points): Given a dataset $\mathcal{D} = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, the goal of linear least squares is to learn the parameters $\tilde{\theta} = \begin{bmatrix} \theta_0 & \theta^\top \end{bmatrix}^\top$ of a linear function $h(x) = \theta_0 + \theta \cdot x = \tilde{\theta}^\top \tilde{x}$, where $\theta_0 \in \mathbb{R}$, $\theta \in \mathbb{R}^d$, $\tilde{\theta} \in \mathbb{R}^{d+1}$, $\tilde{x} = \begin{bmatrix} 1 & x^\top \end{bmatrix}^\top \in \mathbb{R}^{d+1}$. The method aims to minimize the mean squared error:

$$L_{\mathcal{D}}(\tilde{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{\theta}^\top \tilde{x}_i)^2$$

a) (10 pts) Derive the derivative $\frac{\partial}{\partial \tilde{\theta}} L_{\mathcal{D}}(\tilde{\theta})$. Note: this should be a $d+1$-dimensional vector.

b) (10 pts) Show that the loss function $L_{\mathcal{D}}(\tilde{\theta})$ is convex. Hint: Derive the Hessian and show that it is positive semidefinite.

2 Programming assignment (50 + 10 pts)

For the following programming assignment, please download the datasets and iPython notebooks from Canvas and submit the following:

- Completed and ready-to-run iPython notebooks. Note: we will inspect the code and run your notebook if needed. If we cannot run any section of your notebook, you will not receive any points for the task related to that section.

- Responses (texts, codes, and/or figures) to the following problems/tasks

In this programming exercise, you will build a logistic regression model for sentiment analysis.

Task P1 (4 pts): Once you get the bag-of-words representation, append a '1' to the beginning of each vector to allow our linear classifier to learn a bias term. What is the size of the resulting data_mat matrix?

Task P2 (10 pts): As we have seen in the class, to learn the parameters of logistic regression, we need to perform the following optimization:

$$\tilde{\theta}_t = \underset{\tilde{\theta}}{\operatorname{argmin}}\ L_{\mathcal{D}}(\tilde{\theta}) = \underset{\tilde{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} \ln\left(1 + e^{y_i\ \tilde{\theta}_t^T \tilde{x}_i}\right)$$

where $y_i \in \{-1, +1\}$ is the label, $\tilde{\theta}$ is the vector of coefficients:

$$\tilde{\theta} = \begin{bmatrix}\theta_0 & \theta_1 & ... & \theta_d\end{bmatrix}^T,$$

and $\tilde{x}$ is the "augmented" feature vector (of $d+1$ dimensions), where we stick a 1 in the front of the original features:

$$\tilde{x} = \begin{bmatrix}1 & x_1 & ... & x_d\end{bmatrix}^T.$$

Derive the gradient of the loss $L_{\mathcal{D}}(\tilde{\theta})$ with respect to $\tilde{\theta}$, namely $\nabla L_{\mathcal{D}}(\tilde{\theta}_t)$. The answer should depend on data points (x_i, y_i) for $i = 1, ..., n$, and the model parameter $\tilde{\theta}$. Make sure you get the sign correct. Also implement the function 'weight_derivative'. Print the output of the code.

Task P3 (5 pts): Specify the initial_weights, step_size, and tolerance for the function 'gradient_descent'. Copy the outputs of the code to the solution file.

Task P4 (4 pts): Write the code to extract the y-intercept θ_0 and the rest of the parameters $\theta = \begin{bmatrix}\theta_1 & ... & \theta_d\end{bmatrix}^T$. Copy the code and print the outputs.

Task P5 (6 pts): Write the code to make the prediction for a given data matrix. Report the training error and test error.

Task P6 (8 pts): Implement the function 'margin_counts' that takes as input the learned weights $\tilde{\theta}$, the feature matrix ('feature_matrix'), and a value of 'gamma', and computes how many points in the data have margin at least 'gamma'. Copy the code and the output plot (i.e., visualization of the test set's distribution of margin values) to the solution file.

Task P7 (7 pts): Implement the function 'margin_errors' that computes the fraction of points with margin at least 'gamma' that are misclassified. Copy the code and the output plot (i.e., visualization of the relationship between margin and error rate) to the solution file. What do you observe from the plot?

Task P8 (6 pts): Report the top 10 positive words (i.e., words with the largest positive coefficients of θ_j) and the top 10 negative words (i.e., words with the most negative coefficients of θ_j).

Bonus questions (10 pts): Suppose you are building a classifier, and can tolerate an error rate of at most some value $\epsilon \in (0, 1)$. Unfortunately, every classifier you try has a higher error than this.

Therefore, you decide that the classifier is allowed to occasionally "abstain": that is, to say "don't know". When it actually makes a prediction, it is expected to have error rate at most $\epsilon \in (0, 1)$. And subject to this constraint, it should abstain as infrequently as possible.

How would you build an abstaining classifier of this kind, starting from a logistic regression model? To get the bonus score, you need to show the following:

- A general description of the method
- Your code implementation
- A case study to show how you can use it in practice (including necessary plots)