ECE M146 | Homework 3 Solution
Introduction to Machine Learning | Monday, April 12, 2021
Instructor: Lara Dolecek | Due: Monday, April 26, 2021
TA: Zehui (Alex) Chen | chen1046@ucla.edu

Please upload your homework to Gradescope by April 26, 4:00 pm.
Please submit a single PDF directly on Gradescope
You may type your homework or scan your handwritten version. Make sure all the work is discernible.

Note: All logs in the decision tree problems are base 2.

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

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

where $w \in \mathbb{R}^M$, $x^{(n)} \in \mathbb{R}^M$, $y^{(n)} \in \{0,1\}$, $h_w(x^{(n)}) = \sigma(w^T x^{(n)})$ and the sigmoid function $\sigma(t) = \frac{1}{1+\exp(-t)}$.

(a) Find the partial derivative $\frac{\partial J}{\partial w_j}$ and derive the gradient descent update rules for the weights. In the above expression, w_j is the j-th element of w.

Solution:

$$\begin{aligned}
\frac{\partial}{\partial w_j} J(w) &= w_j - \sum_{n=1}^{N}\left[\frac{y^{(n)}}{h_w(x^{(n)})} - \frac{1-y^{(n)}}{1-h_w(x^{(n)})}\right]\frac{\partial}{\partial w_j} h_w(x^{(n)}) \\
&= w_j - \sum_{n=1}^{N}\left[\frac{y^{(n)}}{h_w(x^{(n)})} - \frac{1-y^{(n)}}{1-h_w(x^{(n)})}\right] h_w(x^{(n)})(1-h_w(x^{(n)}))\frac{\partial}{\partial w_j} w^T x \\
&= w_j - \sum_{n=1}^{N}\left[y^{(n)}(1-h_w(x^{(n)})) - (1-y^{(n)})h_w(x^{(n)})\right] x_j^{(n)} \\
&= w_j + \sum_{n=1}^{N}(h_w(x^{(n)}) - y^{(n)})x_j^{(n)}.
\end{aligned}$$

The gradient descent update rules is therefore:

$$w_j := w_j - \eta\left[w_j + \sum_{n=1}^{N}(h_w(x^{(n)}) - y^{(n)})x_j^{(n)}\right],$$

where η denotes the learning rate. You may notice that the update rule is mathematically the same as the result obtained for the gradient descent under quadratic loss and with L2 regularization. This is due to the special choice of the sigmoid function and the cross entropy loss.

(b) When the data is scalar, we have $M = 2$, $x^{(n)} = [x_1^{(n)}, x_2^{(n)}]^T$ where $x_1^{(n)} = 1$, and $w = [w_1, w_2]^T$. Find the Hessian matrix $\nabla_w^2 J(w)$.
Solution: Using results from part (a), we first find the gradient:

$$\nabla_w J(w) = \begin{bmatrix} \frac{\partial}{\partial w_1} J(w) \\ \frac{\partial}{\partial w_2} J(w) \end{bmatrix} = \begin{bmatrix} w_1 + \sum_{n=1}^N (h_w(x^{(n)}) - y^{(n)}) \\ w_2 + \sum_{n=1}^N (h_w(x^{(n)}) - y^{(n)}) x_2^{(n)} \end{bmatrix}.$$

We then find the Hessian matrix:

$$\begin{aligned}
\nabla_w^2 J(w) &= \begin{bmatrix} \frac{\partial^2 J(w)}{\partial w_1 \partial w_1}, \frac{\partial^2 J(w)}{\partial w_1 \partial w_2} \\ \frac{\partial^2 J(w)}{\partial w_2 \partial w_1}, \frac{\partial^2 J(w)}{\partial w_2 \partial w_2} \end{bmatrix} \\
&= \begin{bmatrix} 1 + \sum_{n=1}^N \frac{\partial}{\partial w_1} h_w(x^{(n)}), \sum_{n=1}^N \frac{\partial}{\partial w_2} h_w(x^{(n)}) \\ \sum_{n=1}^N \frac{\partial}{\partial w_1} h_w(x^{(n)}) x_2^{(n)}, 1 + \sum_{n=1}^N \frac{\partial}{\partial w_2} h_w(x^{(n)}) x_2^{(n)} \end{bmatrix} \\
&= \begin{bmatrix} 1 + \sum_{n=1}^N h_w(x^{(n)})(1 - h_w(x^{(n)})), \sum_{n=1}^N h_w(x^{(n)})(1 - h_w(x^{(n)})) x_2^{(n)} \\ \sum_{n=1}^N h_w(x^{(n)})(1 - h_w(x^{(n)})) x_2^{(n)}, 1 + \sum_{n=1}^N h_w(x^{(n)})(1 - h_w(x^{(n)})) x_2^{(n)} x_2^{(n)} \end{bmatrix}.
\end{aligned}$$

2. 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 $w \in \mathbb{R}^m$:

$$w^* = \arg\max_w \prod_{i=1}^{n} P(y_i|x_i, w). \tag{2}$$

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:

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

Assuming a standard Gaussian prior $\mathcal{N}(0, I)$ for the prior on 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 $Z \sim \mathcal{N}(0, I_m)$,

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

Solution: By assumption, $P(y_i|x_i, w) = (h_w(x_i))^{y_i}(1 - h_w(x_i))^{1-y_i}$. We can write the MAP estimator explicitly as:

$$w^* = \arg\max_w \frac{1}{(2\pi)^{\frac{m}{2}}} \exp\left(-\sum_{i=1}^{m} \frac{w_i^2}{2}\right) \prod_{i=1}^{n} (h_w(x_i))^{y_i}(1 - h_w(x_i))^{1-y_i}.$$

Taking negative log of the argument, the MAP estimator is then:

$$\begin{aligned} w^* &= \arg\min_w -\sum_{n=1}^{N} \left[y_n \log h_w(x_n) + (1 - y_n)\log(1 - h_w(x_n))\right] + \frac{1}{2}\sum_i w_i^2 \\ &= \arg\min_w J(w), \end{aligned}$$

where $J(w)$ is the loss function in Question 1.

3. You are trying to choose between two restaurants (sample 9 and sample 10) to eat at. To do this, you will use a decision tree that will be trained based on your past experiences (sample 1-8). The features for each restaurants and your judgment on the goodness of sample 1-8 are summarized by the following chart.

Sample #	HasOutdoorSeating	HasBar	IsClean	HasGoodAtmosphere	IsGoodRestaurant
1	0	0	0	1	1
2	1	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	1	1	0	0
6	1	0	1	0	1
7	1	1	0	1	1
8	0	0	1	1	1
9	0	1	0	1	?
10	1	1	1	1	?

(a) What is the entropy of IsGoodRestaurant, i.e., $H(IsGoodRestaurant)$?

Solution: Define the binary entropy function as follows:

$$H_b(p) = -p\log(p) - (1-p)\log(1-p).$$

$$H(IsGoodRestaurant) = H_b(\frac{2}{8}) = -(\frac{6}{8}\log(\frac{6}{8}) + \frac{2}{8}\log(\frac{2}{8})) \approx 0.8113$$

(b) Calculate the conditional entropy of IsGoodRestaurant conditioning on HasOutdoorSeating. To do this, first compute $H(IsGoodRestaurant|HasOutdoorSeating = 0)$ and $H(IsGoodRestaurant|HasOutdoorSeating = 1)$, then weigh each term by the probabilities $P(HasOutdoorSeating = 0)$ and $P(HasOutdoorSeating = 1)$, respectively. Namely, calculate the following:

$$\begin{aligned}&H(IsGoodRestaurant|HasOutdoorSeating)\\ =&P(HasOutdoorSeating = 0)H(IsGoodRestaurant|HasOutdoorSeating = 0)\\ +&P(HasOutdoorSeating = 1)H(IsGoodRestaurant|HasOutdoorSeating = 1).\end{aligned}$$

Solution: Using the given equation, we get:

$$\begin{aligned}&H(IsGoodRestaurant|HasOutdoorSeating)\\ =&\frac{3}{8}H_b(\frac{3}{3}) + \frac{5}{8}H_b(\frac{3}{5}) \approx 0.606844.\end{aligned}$$

(c) Similarly, calculate

$$H(IsGoodRestaurant|X), \text{for } X \in \{HasBar, IsClean, HasGoodAtmosphere\},$$

i.e., the conditional entropy of IsGoodRestaurant conditioning on the other three features.

Solution:

$$H(IsGoodRestaurant|HasBar) = \frac{1}{2}H_b(\frac{4}{4}) + \frac{1}{2}H_b(\frac{2}{4}) = 0.5.$$

$$H(IsGoodRestaurant|IsClean) = \frac{1}{2}H_b(\frac{3}{4}) + \frac{1}{2}H_b(\frac{3}{4}) \approx 0.8113.$$

$$H(IsGoodRestaurant|HasGoodAtmosphere) = \frac{3}{8}H_b(\frac{1}{3}) + \frac{5}{8}H_b(\frac{5}{5}) \approx 0.34436.$$

(d) Calculate the information gain:

$$I(IsGoodRestaurant; X) = H(IsGoodRestaurant) - H(IsGoodRestaurant|X),$$

for

$$X \in \{HasOutdoorSeating, HasBar, IsClean, HasGoodAtmosphere\}.$$

Solution:

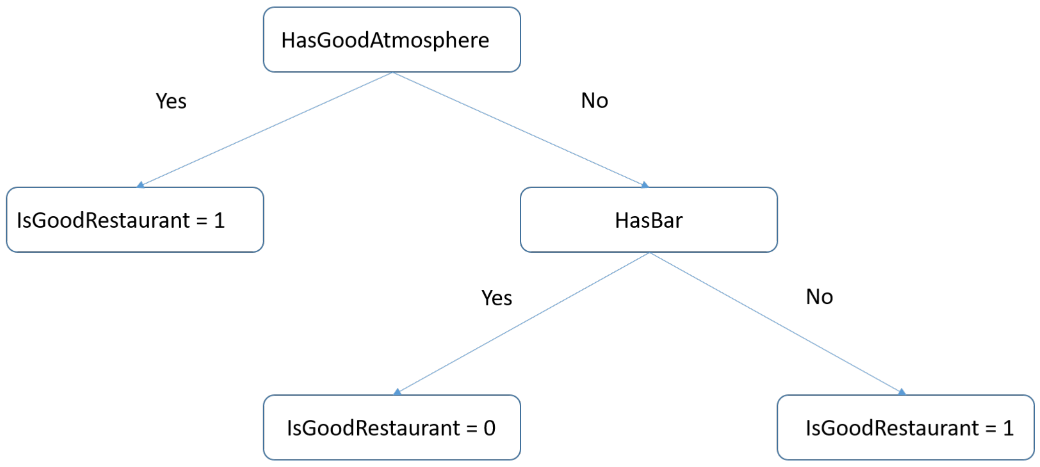
$$\begin{aligned} I(IsGoodRestaurant; HasOutdoorSeating) &= 0.8113 - 0.606844 = 0.204456; \\ I(IsGoodRestaurant; HasBar) &= 0.8113 - 0.5 = 0.3113; \\ I(IsGoodRestaurant; IsClean) &= 0.8113 - 0.8113 = 0; \\ I(IsGoodRestaurant; HasGoodAtmosphere) &= 0.8113 - 0.34436 = 0.46694. \end{aligned}$$

(e) Based on the information gain, determine the first attribute to split on.

Solution: We choose HasGoodAtmosphere which has the largest information gain.

(f) Make the full decision tree. Start at the root with first splitting attribute you found in part (e). After each split, treat the sets of samples with $X = 0$ and $X = 1$ as two separate sets and redo (b), (c), (d) and (e) on each of them. X is the feature for previous split and is thus excluded from the available features which can be split on next. Terminate splitting if after the previous split, the entropy of IsGoodRetaurant in the current set is 0. For example, if we choose HasGoodAtmosphere as our first feature to split, we get $H(IsGoodRetaurant|HasGoodAtmosphere = 1) = 0$. We thus stop splitting the tree in this branch. Draw the tree and indicate the split at each node.

Solution: Below shows the decision tree if we choose HasGoodAtmosphere as the first splitting feature.



After the first split, $H(IsGoodRetaurant|HasGoodAtmosphere = 1) = 0$ so the tree stops growing on that branch. We are left with the samples that have $HasGoodAtmosphere = 0$ which is summarized in the following table.

Sample #	HasOutdoorSeating	HasBar	IsClean	IsGoodRestaurant
2	1	1	0	0
5	1	1	1	0
6	1	0	1	1

By observation, Feature HasBar has the highest information gain. Then the next split should be HasBar. After this split, every leaf is pure, i.e., IsGoodRestaurant is either 0 or 1. Therefore, we stop growing the tree.

(g) Now, determine if restaurants 9 and 10 are good or not.

Solution:

Restaurant 9: Good

Restaurant 10: Good

4. When training decision trees for classification, we generally use entropy or gini index to help determine good splits. These functions are examples of impurity measures. We can formally define impurity measures as follows.

   Assume we are performing binary classification. Let V be a set of data points and let $\{y_i \in \{+1, -1\}\ ,\ \forall i \in V\}$ be the set of labels. Let $V_1 \subseteq V$. We define p and q as follows:

$$p(V_1, V) = \frac{|V_1|}{|V|} \quad q(V_1) = \frac{|\{i : i \in V_1, y_i = 1\}|}{|V_1|}$$

   where $|\cdot|$ is the cardinality of a set.
   Let $i(q(V))$ measure the impurity of a set V. Two desired properties of $i(q(V))$ are:

   - $i(q(V)) = 0$ when the set has only one class
   - $i(q(V))$ reaches its maximum value for sets where the two classes are perfectly balanced.

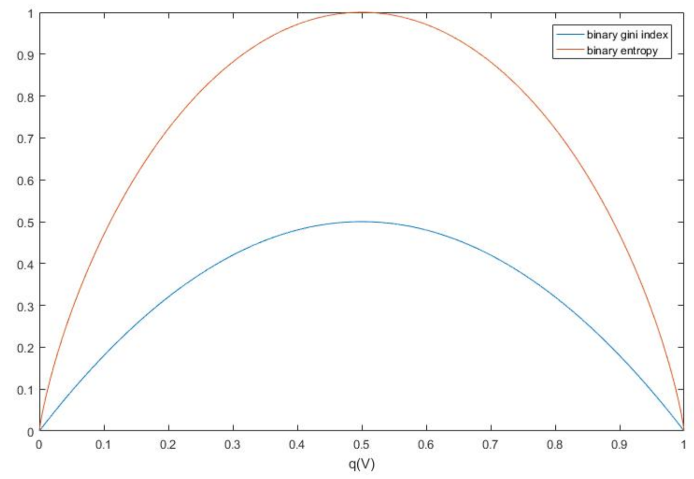
   When a split is performed on a set V, the result is two sets $V_1 \cup V_2 = V$ such that $V_1 \cap V_2 = \emptyset$. The information gain of this split is defined as

$$I(V_1, V_2, V) = i(q(V)) - (p(V_1, V)i(q(V_1)) + p(V_2, V)i(q(V_2))).$$

   Using the previous notation, we define the binary gini Index and binary entropy as:

$$\begin{aligned} gini(q(V)) &= 2q(V)(1 - q(V)) \\ H(q(V)) &= -(q(V)\log(q(V)) + (1 - q(V))\log(1 - q(V))) \end{aligned}$$

   (a) Like binary entropy, gini index is an impurity measure that is commonly used. Plot both the Gini index and the binary entropy function for $q(V)$ from 0 to 1. Comment on their similarities.
   Solution:



   They both attains the maximum value at $q(V) = \frac{1}{2}$. They are both 0 when $q(V) = 0$ or $q(V)1$.

(b) Show that if $i(q(V))$ is concave in $q(V)$ then $I(V_1, V_2, V) \geq 0 \; \forall \; V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$. This means that every split does not lose any information. Recall that a function is concave then $i(\lambda a_1 + (1-\lambda)a_2) \geq \lambda i(a_1) + (1-\lambda)i(a_2), \forall a_1, a_2, \forall \lambda \in [0, 1]$.

Solution: For simplicity, let $q = q(V), q_1 = q(V_1), q_2 = q(V_2), p_1 = p(V_1, V), p_2 = p(V_2, V)$.

$$\begin{aligned}
q &= \frac{|\{i : i \in V, y_i = 1\}|}{|V|} \\
&= \frac{|\{i : i \in V_1, y_i = 1\}|}{|V|} + \frac{|\{i : i \in V_2, y_i = 1\}|}{|V|} \\
&= \frac{|V_1|}{|V|}\frac{|\{i : i \in V_1, y_i = 1\}|}{|V_1|} + \frac{|V_2|}{|V|}\frac{|\{i : i \in V_2, y_i = 1\}|}{|V_2|} \\
&= p_1 * q_1 + p_2 * q_2
\end{aligned}$$

Due to concavity, we have

$$i(q) = i(p_1q_1 + p_2q_2) \geq p_1 i(q_1) + p_2 i(q_2).$$

Hence, $I(V_1, V_2, V) \geq 0$

(c) Show that binary entropy is concave. Hint: Show that the 2nd derivative is always non-positive.

Solution:

$$\begin{aligned}
\frac{dH(q)}{dq} &= \log(\frac{1}{q} - 1) \\
\frac{d^2H(q)}{dq^2} &= -\frac{1}{q - q^2}
\end{aligned}$$

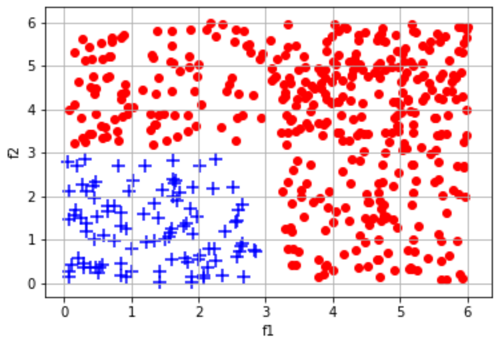
Since $q \in [0, 1]$, then $q \geq q^2$. As such, the second derivative is always non-positive.

(d) Show that the binary Gini index is concave.

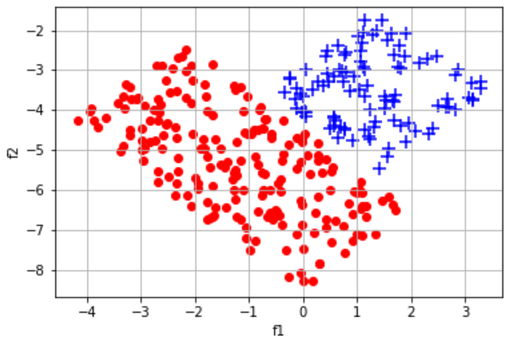
Solution: Similarly,

$$\begin{aligned}
\frac{dgini(q)}{dq} &= 2(1 - 2q) \\
\frac{d^2gini(q)}{dq^2} &= -4
\end{aligned}$$

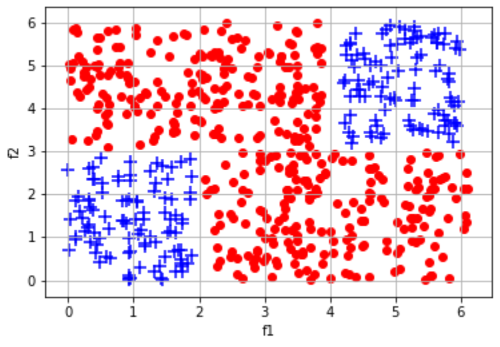
5. Consider the following plots:



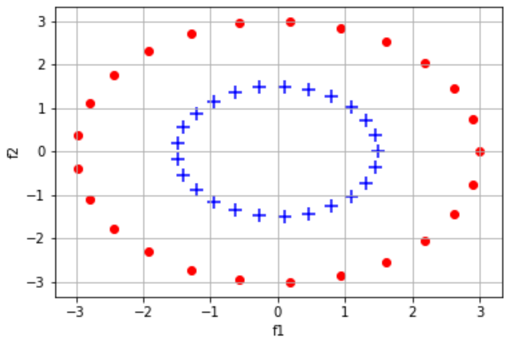
(1) Decision Tree Example 1



(2) Decision Tree Example 2



(3) Decision Tree Example 3



(4) Decision Tree Example 4

Assume that you are using a binary split decision tree to classify the points. At each node, you may ask questions like: "Is $f_1 \geq 3$?" or "Is $f_2 \leq 1.8$?". Here, $f1$ and $f2$ are the variables on the horizontal axis and vertical axis of the examples, respectively.

(a) Which examples can be fully separated using a depth 2 decision tree?

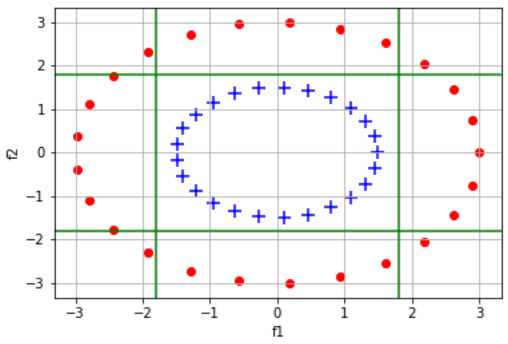
Solution: Examples 1 and 3. For example 1, the first question to ask is "Is $f1 \geq 3$?". If "No", ask the second question "Is $f2 \geq 3$?". For example 2, ask the first question "Is $f2 \geq 3$?" . If "No", ask the second question "Is $f1 \geq 2$?". If "Yes", ask the second question "Is $f1 \geq 4$?".

(b) Which example would have the most complex decision tree in terms of the number of splits? Explain why.

Solution: Example 2 because the points are separated by a non-axis aligned hyperplane which would require a lot of binary splits to model. Decision tree works well only when the decision boundaries are axis-aligned.

(c) If you used a depth 4 decision tree, is one more example now separable? If so, show how you can separate it using a depth 4 decision tree.

Solution: Example 4. You may ask these four questions: "Is $f1 \geq -1.8$?", "Is $f1 \leq 1.8$?", "Is $f1 \geq -1.8$?" and "Is $f1 \leq 1.8$?". If all answers are "Yes", then we decide the class is the blue one. See illustration in the next figure.


f1
f2
−3
−2
−1
0
1
2
3

6. In this problem, you will implement the logistic regression algorithm and test them on our running datasets *UCLA_EE_grad_2030.csv* and *UCLA_EE_grad_2031.csv*. You can find detailed description of the datasets in HW1. Note that the labels are $-1/+1$ in the dataset instead of $0/1$ so a conversion may be needed.

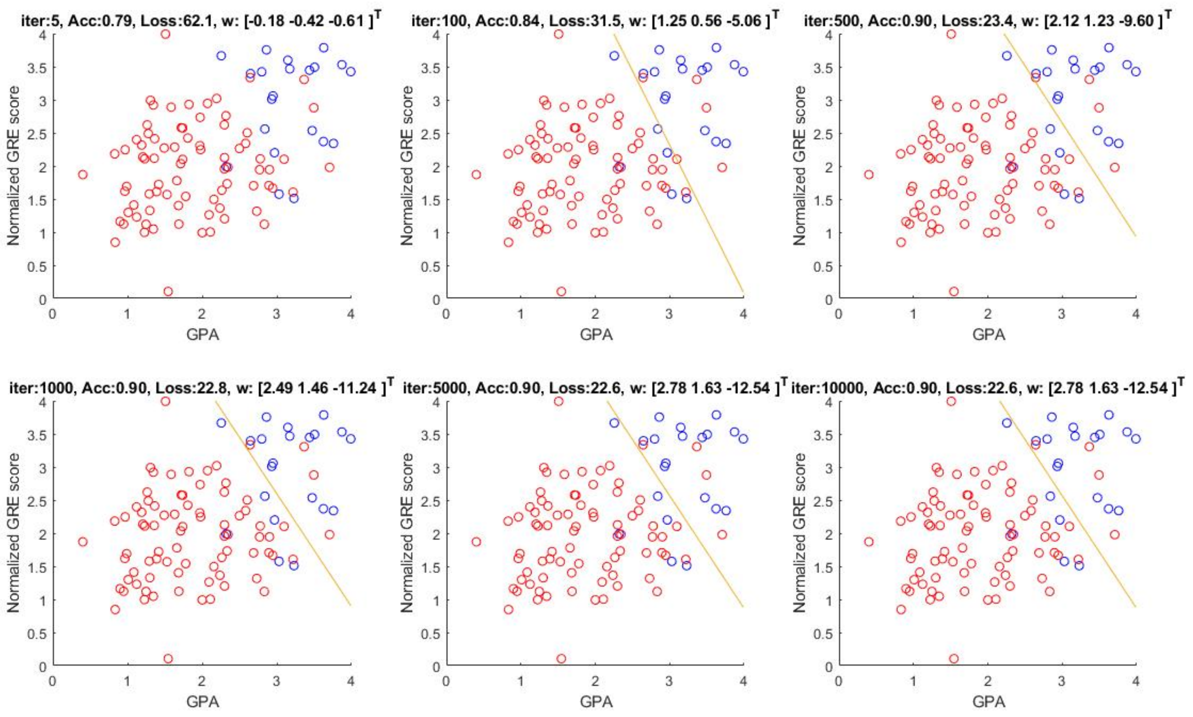
   In the logistic regression algorithm, we wish to find a vector $w = [w_1, w_2, b]^T \in \mathbb{R}^3$ such that the following logistic loss is minimized:

$$J(w) = -\sum_{n=1}^{N} \left[y^{(n)} \log h_w(x^{(n)}) + (1 - y^{(n)}) \log(1 - h_w(x^{(n)})) \right], \tag{4}$$

   where $x^{(n)} \in \mathbb{R}^3$, $y^{(n)} \in \{0, 1\}$, $h_w(x^{(n)}) = \sigma(w^T x^{(n)})$ and the sigmoid function $\sigma(t) = \frac{1}{1+\exp(-t)}$. Note that the two dimensional feature vector need to be augmented with 1 to include a bias term.

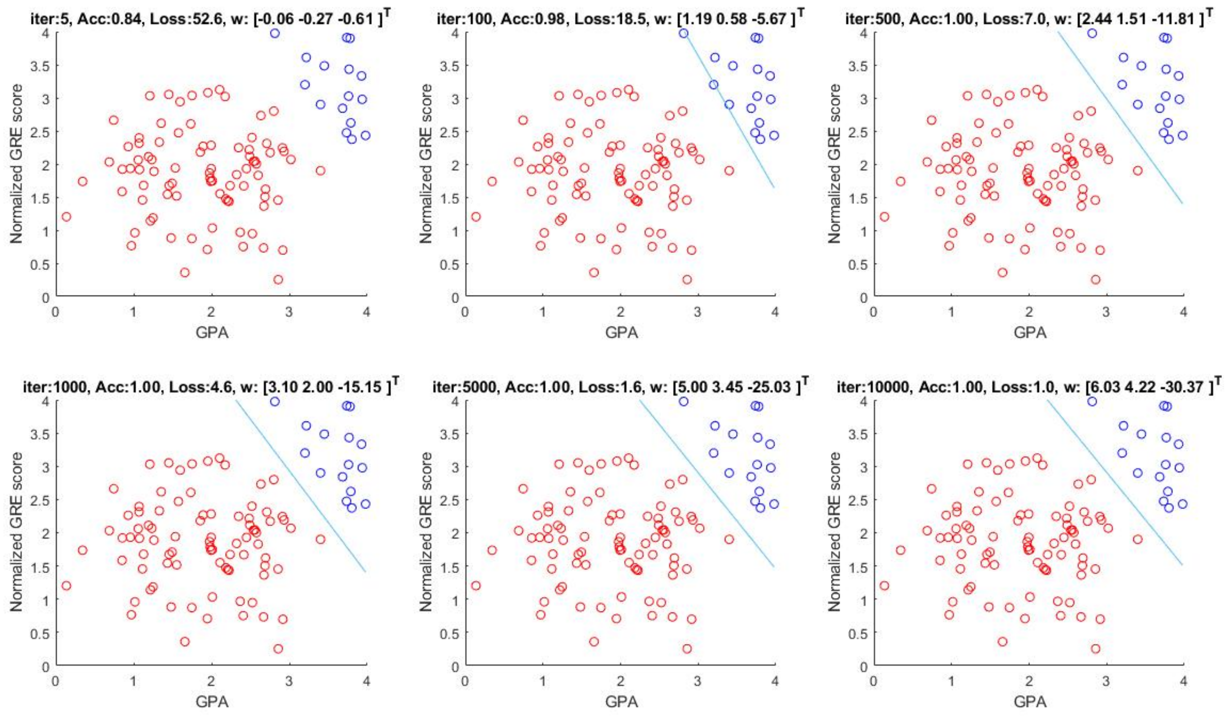
   (a) For each dataset, starting with the weights being all zeros, train the logistic regression model with the gradient descent algorithm using **learning rate 0.01** for **10000 iterations**. For iteration 5, 100, 500, 1000, 5000 and 10000, plot the data and the learned hyperplane. For these iterations, also report the weights, the loss $J(w)$, and the classification accuracy of this hyperplane.

Figure 2: UCLA_EE_grad_2030



Solution: The weights, the loss $J(w)$, and the classification accuracy are reported in the following figures. Note that for both datasets, at iteration 5, the hyperplane is not within the range of GPA/GRE and it is therefore not shown.

Figure 3: UCLA_EE_grad_2031



(b) Comment on the convergence of this algorithm on the linearly non-separable dataset. Does the algorithm converge and why?

Solution: The algorithm converges for the linearly non-separable dataset. We observe that the loss and learned weight are not changing after iteration 5000.

(c) Comment on the convergence of this algorithm on the linearly separable dataset. Does the algorithm converge? If not, does the separating hyperplane change a lot after iteration 1000? Does the loss change a lot after iteration 1000? Given a brief explanation on your observation.

Solution: The algorithm does not converge for the linearly separable dataset. We observe that the loss and learned weight are still changing after iteration 1000 while the hyperplane is nearly unchanged.

For a linearly separable dataset, once a separating hyperplane is found, one can always multiply the weights with a factor and thus reduce the loss while keeping the separating hyperplane unchanged. See iteration 1000 and iteration 10000 for an example. This causes the algorithm to not converge. The weights will tend to positive/negative infinity and this will cause the model to over-fit; this situation is not desirable if one seeks to probabilistically make the decision on data.

7. In this section, you will consider a k-nearest neighbor classifier using Euclidean distance metric on a binary classification task. We assign the class of the test point to be the class of the majority of the k nearest neighbors.

   Consider the following two datasets:

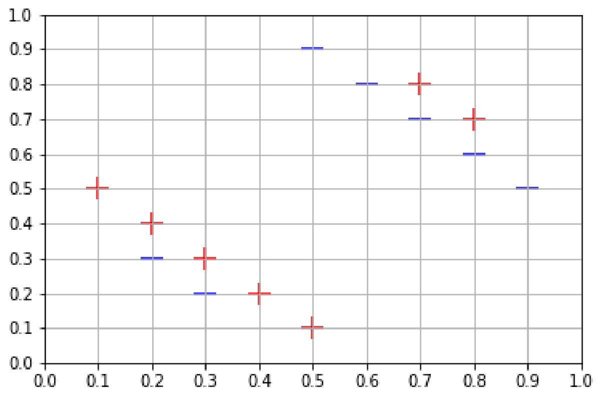


Figure 4: KNN Example 1

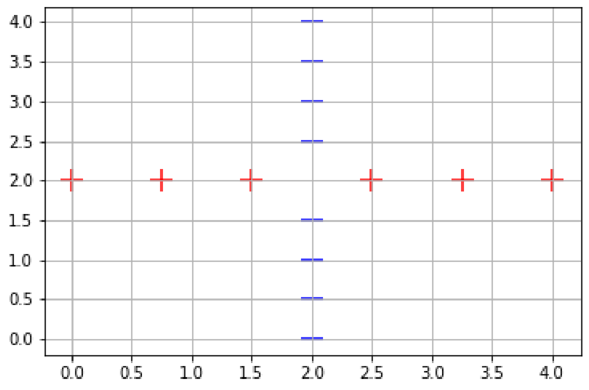


Figure 5: KNN Example 2

   (a) For $k \in \{1, 3, 5, 7\}$, what values of k minimize leave-one-out cross-validation error for each dataset? What is the resulting validation error?

   Solution:

   For the first example, the best k are 5 and 7 with a validation error of $\frac{4}{14}$. If $k = 1$, with leave-one-out cross-validation, 10 points (despite the four points at $(0.1, 0.5), (0.5, 0.1), (0.9, 0.5)and(0.5, 0.9)$) are classified erroneously, resulting in a validation error of $\frac{10}{14}$. If $k = 3$, in addition to the 4 points that are seemingly out of place, the two points at $(0.3, 0.3)$ and $(0.7, 0.7)$ are also classified erroneously, resulting in a validation error of $\frac{6}{14}$.

For the second example, the best k are 1 with a validation error of $\frac{2}{14}$. The two misclassified points are $(1.5, 2)$ and $(2.5, 2)$. For $k = 3, 5$ and 7, there are at least two additional points that are classified erroneously. For $k = 3$, these two points are $(2, 1.5)$ and $(2, 2.5)$. For $k = 5$, these two points are $(1.75, 2)$ and $(1.75, 2)$. For $k = 7$, these two points are $(0, 2)$ and $(4, 2)$.

(b) In general, what are the drawbacks of using too big a k? What about too small a k? To see this, calculate the leave-one-out cross-validation error for the dataset in Figure 4 using $k = 1$ and $k = 13$.

Solution:

In general, too big a k causes underfitting since the largest class would dominate and too small a k causes overfitting. For Example 1, the validation error for $k = 1$ is $\frac{10}{14}$. The validation error for $k = 13$ is $\frac{14}{14}$ which is really bad.
Note although this statement holds true in general, for certain cases, e.g, our synthetic example 2, small k or large k may be preferred. Moreover, the nearest neighbor classifier, i.e., $k = 1$, is one commonly used classifier.