Boston University
Department of Electrical and Computer Engineering

ENG EC 414 Introduction to Machine Learning

HW 4 Solution

Issued: Fri 25 Sept 2020 **Due:** 10:00am Fri 2 Oct 2020 in Gradescope (non-code) + Blackboard

Important: Before you proceed, please read the documents pertaining to *Homework formatting and submission guidelines* in the Homeworks section of Blackboard. **In particular, for computer assignments you are prohibited from using any online code or built-in MATLAB functions except as indicated in the problem or skeleton code (when provided).**

Note: Problem difficulty = number of coffee cups ☕

Problem 4.1 [15pts] *(Hyperplanes)*
Consider two-class classification with the following training feature vectors: $\mathbf{x}_P = (2,0)^\top, \mathbf{x}_Q = (0,4)^\top, \mathbf{x}_R = (3,3)^\top, \mathbf{x}_S = (7,5)^\top$ with labels $-1, -1, +1, +1$ respectively.

(a) [2pts] Hand-plot the training set. Proper labeling of axes and key points is needed to receive full credit.

(b) [1pt] Hand-sketch the hyperplane with parameters $\mathbf{w} = (3,0)^\top$, $b = -3$. Proper labeling of axes and key points is needed to receive full credit.

(c) [2pts] Hand-compute the ℓ_2 distance of $\mathbf{x}_P$ from the hyperplane in part (b).

(d) [2pts] Determine if the hyperplane in part (b) linearly separates the training set. Explanation is needed to receive credit.

(e) [3pts] Hand-compute the parameters of the hyperplane passing through $\mathbf{x}_P$ and $\mathbf{x}_Q$.

(f) [5pts] Hand-compute the orthogonal projection of $\mathbf{x}_R$ onto the hyperplane in part (e).

Solution:

(a),(b) See Figure 1.

(c)

$$\text{dist}(\mathbf{x}_P, (\mathbf{w}, b)) = \frac{|\mathbf{w}^\top \mathbf{x}_P + b|}{\|\mathbf{w}\|} = \frac{|3 \times 2 + 0 - 3|}{\sqrt{3^2 + 0^2}} = \frac{3}{3} = 1.$$

(d) No: $\text{sign}(\mathbf{w}^\top \mathbf{x}_P + b) = +1$, $\text{sign}(\mathbf{w}^\top \mathbf{x}_Q + b) = -1$, but $\mathbf{x}_P$ and $\mathbf{x}_Q$ are in the same class.

(e) Slope $= (0-4)/(2-0) = -2$. Vertical intercept $= 4$. Therefore equation of line is $x_2 = -2x_1 + 4$. Remember that there are infinite representations of the same hyperplane, so a possible representation is:

$$\underbrace{(2,1)^\top}_{\mathbf{w}^\top} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \underbrace{-4}_{b} = 0.$$

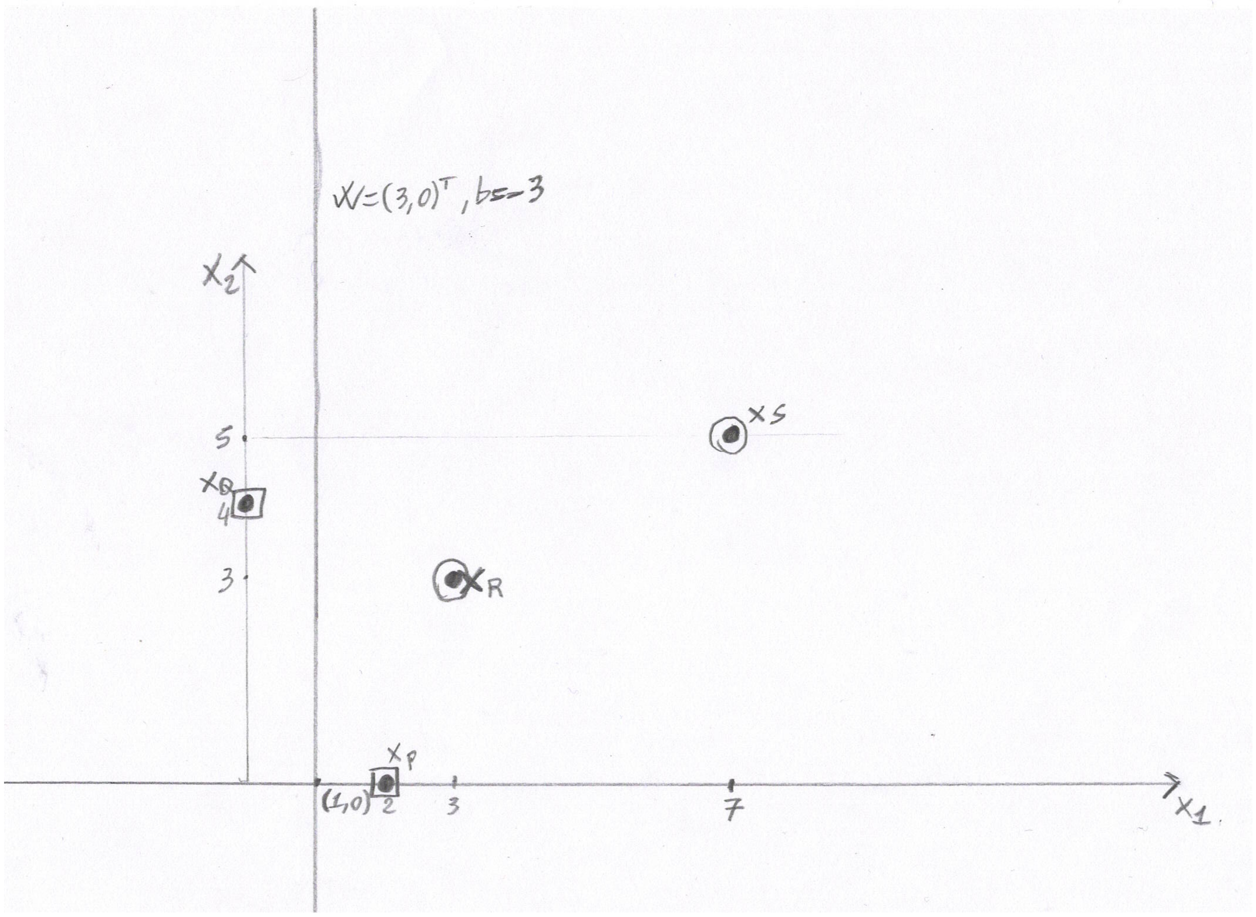


Figure 1: Figure for questions (a) and (b).

(f) If $\mathbf{x}_{\_}$ denotes the orthogonal projection, then $(\mathbf{x}_R - \mathbf{x}_{\_}) = \gamma \mathbf{w}$ for some γ since $\mathbf{w}$ is perpendicular to the hyperplane. Thus, $\mathbf{x}_{\_} = \mathbf{x}_R - \gamma \mathbf{w}$. Since $\mathbf{x}_{\_}$ lies on the hyperplane, it satisfies the equation of the hyperplane: $\mathbf{w}^\top \mathbf{x}_{\_} + b = 0$. This implies that $\gamma = \frac{\mathbf{w}^\top \mathbf{x}_R + b}{\|\mathbf{w}\|^2} = \frac{9-4}{4+1} = 1$. Hence, $\mathbf{x}_{\_} = \mathbf{x}_R - \mathbf{w} = (1,2)^\top$.

Problem 4.2 [8pts] *(Logistic regression by hand)*
Let

$$\mathcal{S} = \Big\{(x_1 = -3, y_1 = -1), (x_2 = 5, y_2 = +1)\Big\}$$

be a training set of pairs of scalar features and their labels for training a binary logistic regression classifier with class labels $-1, +1$.

(a) [2pts] Let $F(w,b) = \frac{1}{m}\sum_{i=1}^m \ln(1 + \exp(-y_i(b + wx_i)))$ be the objective function for one-dimensional logistic regression. Compute the expression of the partial derivative of this function with respect to w and b using the training set above.

(b) [2pts] With initialization $w_1 = 0$ and $b_1 = 0$, compute w_2 and b_2, that is the value after one iteration of the gradient descent algorithm for minimizing the objective function, expressing them as functions of the step-size $\eta > 0$.

(c) [4pts] Fix $b = 0$ and find analytically the minimum of the objective function on the above training set and the optimal value of w.

Solution:

(a) [2pts] First of all, we have

$$F(w,b) = \frac{1}{m}\sum_{i=1}^m \ln(1 + \exp(-y_i(b + wx_i))) = \frac{1}{2}[\ln(1 + \exp(-3w + b)) + \ln(1 + \exp(-5w - b))] \ .$$

Hence, the partial derivatives are

$$\frac{\partial F}{\partial w} = \frac{1}{2}\left[\frac{-3\exp(-3w+b)}{1+\exp(-3w+b)} + \frac{-5\exp(-5w-b)}{1+\exp(-5w-b)}\right],$$
$$\frac{\partial F}{\partial b} = \frac{1}{2}\left[\frac{\exp(-3w+b)}{1+\exp(-3w+b)} + \frac{-\exp(-5w-b)}{1+\exp(-5w-b)}\right] .$$

(b) [2pts]

$$w_2 = w_1 - \eta \cdot \left.\frac{\partial F}{\partial w}\right|_{w=0,b=0} = 0 - \eta\frac{1}{2}\left[\frac{-3}{1+1} + \frac{-5}{1+1}\right] = 2\eta,$$
$$b_2 = b_1 - \eta \cdot \left.\frac{\partial F}{\partial b}\right|_{w=0,b=0} = 0 - \eta\frac{1}{2}\left[\frac{1}{1+1} + \frac{-1}{1+1}\right] = 0 \ .$$

(c) [4pts] Fixing $b = 0$, the objective function is

$$\frac{1}{2}[\ln(1 + \exp(-3w)) + \ln(1 + \exp(-5w))] \ .$$

It is enough to observe that this is a strictly decreasing function of w (since e^{-t} strictly decreases as t increases). Hence, the minimum is 0 at w equal to infinity. (More properly, the minimum does not exist, the infimum is 0, and the limit for w that goes to infinity is 0. I'll accept both answers.)

Problem 4.3 [21pts] *(Gradient Descent for Logistic regression)*
In this problem, we will implement the Gradient Descent (GD) Algorithm to learn the parameters of a logistic regression classifier. We will use again the Adult dataset.

Algorithm 1 Gradient Descent for Logistic Regression

```
1: Input: Training set $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, learning rate $\eta > 0$, maximum number of iterations $T$, initial hyperplane $\boldsymbol{w}_1$, initial bias $b_1$
2: Set $\tilde{\mathbf{w}}_1 = \begin{bmatrix} b_1 \\ \boldsymbol{w}_1 \end{bmatrix} \in \mathbb{R}^{d+1}$
3: Construct augmented training features: $\tilde{\boldsymbol{x}}_1, \dots, \tilde{\boldsymbol{x}}_m$
4: for $t = 1, 2, \dots, T$ do
5:     Calculate value of objective function: $obj_t = \sum_{i=1}^{m} \ln(1 + \exp(-y_i \tilde{\boldsymbol{w}}_t^\top \tilde{\boldsymbol{x}}_i))$
6:     Compute gradient: $\tilde{\boldsymbol{g}}_t = -\sum_{i=1}^{m} \frac{y_i \tilde{\boldsymbol{x}}_i}{1+\exp(y_i \tilde{\boldsymbol{w}}_t^\top \tilde{\boldsymbol{x}}_i)} \in \mathbb{R}^{d+1}$
7:     Gradient descent step: $\tilde{\boldsymbol{w}}_{t+1} = \tilde{\boldsymbol{w}}_t - \eta \tilde{\boldsymbol{g}}_t$
8: end for
9: return Output: Extract $\boldsymbol{w}_{T+1}$ and $b_{T+1}$ from $\tilde{\mathbf{w}}_{T+1}$ and return them
```

Summary of cost function and algorithm: Let $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$ be a training set of m examples consisting of feature vector $\mathbf{x}_j \in \mathbb{R}^d$ and label $y_j \in \{-1, 1\}$ for each j. Let $\tilde{\mathbf{x}}_j = \begin{bmatrix} 1 \\ \mathbf{x}_j \end{bmatrix} \in \mathbb{R}^{d+1}$ denote the augmented representation of the j-th feature vector. The pseudocode for implementing GD for logistic regression is as follows:

(a) [6pts] Complete the skeleton of `train_logistic_regression_gd.m` implementing Algorithm 1 in a function with prototype

```
[w, b, obj] = train_logistic_regression_gd(X, y, eta, T, w1, b1)
```

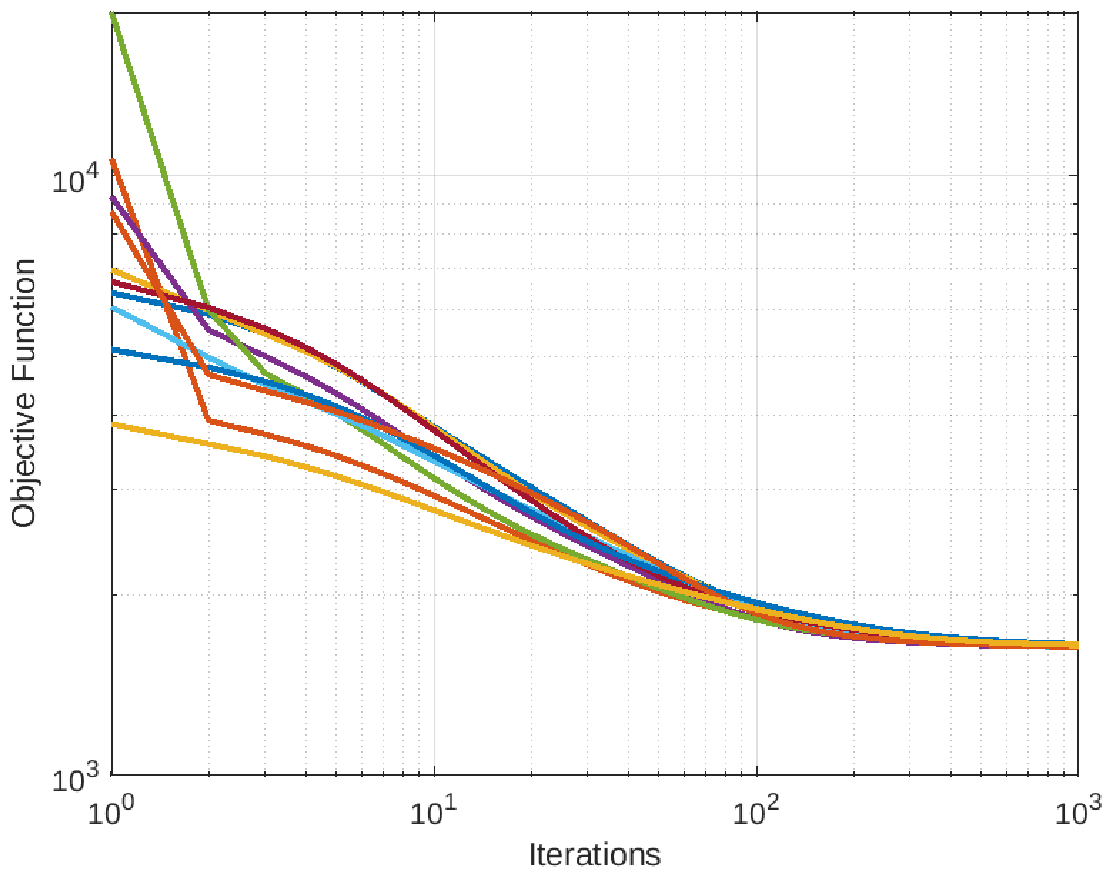
where `w` $\in \mathbb{R}^d$ and `b` $\in \mathbb{R}$ are the hyperplane vector and the bias found by GD, `obj` is the vector of objective function values, `X` is the matrix of the training samples $\in \mathbb{R}^{m \times d}$, `y` is the vector of labels $\in \mathbb{R}^m$, `eta` is the learning rate, `T` is the maximum number of iterations to run, `w1` $\in \mathbb{R}^d$ is the initialization for $\boldsymbol{w}$, and `b1` $\in \mathbb{R}$ is the initialization for b. Note: It will be faster if you use Matlab vectorized operations to calculate the gradient, but it is also fine to use for loops.

(b) [4pts] In class, we said that using a random initialization to minimize a convex function should be avoided. Here, we will test this claim. Complete the skeleton code in `problem_4_3b.m` to use your implementation of logistic regression on the Adult dataset for $T = 1000$ iterations, $\eta = 1/5000$, and 10 random initializations using Gaussian with zero mean and variance 1 on each coordinate of $\boldsymbol{w}$ and on b (You can generate Gaussian noise in Matlab with `randn`).

(c) [2pts] Plot the 10 lines of the values of the objective function during training in the point above. What do you observe? Based on these results and what we said in class, why random initialization should be avoided? (To better observe the difference among the lines, use `loglog` that uses a logarithmic scale on both axes.)

(d) [4pts] Complete the skeleton code in `problem_4_3d.m` to use your implementation of GD for logistic regression, starting from the zero vector and zero bias, $T = 1000$ iterations, $\eta = 1/5000$, and test the obtained solution on the test set. Count the number of prediction mistakes on the test set, predicting for each sample the class with the highest probability.

(e) [5pts] ☕ If the learning rate is too big, you should get a lot of "Inf". Verify it yourself using for example $\eta = 0.1$. Propose a way to fix this issue. Hint: First, locate the cause of the problem, then try to understand why we get this problem numerically. Finally, think how to fix the problem. You can propose approximations.



Solution:

(a) [6pts] See code.

(b) [4pts] See code.

(c) [2pts] The objective function is convex, so GD will converge no matter where we start. In fact, the plots of the objective functions becomes very similar after few iterations. However, the random initialization makes the optimization process stochastic, this means that it is different each time we run it. Given that we don't gain anything from this random initialization, the stochasticity should be avoided to have a deterministic and simpler-to-debug code.

(d) [4pts] See code.

(e) [5pts] ☕ The problem is due to the fact that we need to calculate $\ln(1 + \exp(-y_i \tilde{\boldsymbol{w}}_t^\top \tilde{\boldsymbol{x}}_i))$, that is a well-behaved quantity, but the exponential can quickly become too big to be handled by Matlab. Let's see how to solve this problem. Basically, we need a numerically safe way to calculate $\ln(1 + \exp(x))$ for large values of x, while for small values we can just rely on Matlab. We can observe that when x is very large we have $\ln(1 + \exp(x)) \approx \ln(\exp(x)) = x$. Hence, we can substitute the function $\ln(1 + \exp(x))$ with the safe approximation

$$\ln(1 + \exp(x)) \approx \begin{cases} \ln(1 + \exp(x)), & \text{if } x < 20, \\ x & \text{otherwise} . \end{cases}$$

Note that the threshold "20" is arbitrary: It is chosen simply to have something that Matlab can still handle, while resulting in a good approximation. Also, note that simply using the approximation $\ln(1 + \exp(x)) \approx x$ for all x would not be a good idea. For example $\ln(1 + \exp(x))$ is always positive while x is not, resulting in a very poor approximation.

Code-submission via Blackboard: You must prepare 3 files: `train_logistic_regression_gd.m` for Problem 4.3(a), `problem_4_3b.m` for Problem 4.3(b), and `problem_4_3d.m` for Problem 4.3(d). Place them in a **single** directory which should be zipped and uploaded into Blackboard. Your directory must be named as follows: `<yourBUemailID>_hwX` where `X` is the homework number. For example, if your BU email address is `charles500@bu.edu` then for homework number 4 you would submit a single directory named: `charles500_hw4.zip` which contains all the MATLAB code (and only the code).

Three corresponding skeleton code files are provided for your reference. Reach-out to the TAs via Piazza and their office/discussion hours for questions related to coding.