Boston University
Department of Electrical and Computer Engineering

ENG EC 414 Introduction to Machine Learning

HW 3 Solution

Issued: Tue 15 Sept 2020 **Due:** 10:00am Fri 25 Sept 2020 on Gradescope + 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 ☕

Notation. In the following, we will also denote the training set as $\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ for regression and $y_i \in \{-1, 1\}$ for binary classification, for $i = 1, \dots, m$. Define the input matrix $X \in \mathbb{R}^{m\times d}$ with rows the inputs $\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_m^\top$ and the label vector $\boldsymbol{y} = (y_1, \dots, y_m)$.

As we did in class, define $\tilde{\boldsymbol{x}}_i = \begin{bmatrix} 1 \\ \boldsymbol{x}_i \end{bmatrix}$, for $i = 1, \dots, m$, and the augmented input matrix $\tilde{X} = \begin{bmatrix} \mathbf{1} & X \end{bmatrix}$, where $\mathbf{1}$ is the (column) vector of all ones, and set $\tilde{\boldsymbol{w}} = \begin{bmatrix} b \\ \boldsymbol{w} \end{bmatrix}$. In the following, all the norms are L2 norms a.k.a. Euclidean norms.

Problem 3.1 [6pts] Consider the following training set $\mathcal{S} = \{(x_1, y_1) = (-1, -1), (x_2, y_2) = (-1/2, -1/8), (x_3, y_3) = (0, 0), (x_4, y_4) = (1/2, 1/8), (x_5, y_5) = (1, 1)\}$. Hand-compute the following:

(a) [6pts] *Ordinary Least Squares*: $(w_{OLS}, b_{OLS}) = \arg\min_{w,b} \sum_{j=1}^{5} (y_j - wx_j - b)^2$.

Solution:

(a) We can solve the system of linear equations by hand, or use directly the formula to solve LS. The formula as far as you do all the calculations by hand, even the inverse. Here, we will consider the system of linear equation $(\tilde{X}^\top \tilde{X})\tilde{\boldsymbol{w}} = \tilde{X}^\top \boldsymbol{y}$. We have that $\tilde{X} = \begin{bmatrix} 1 & -1 \\ 1 & -1/2 \\ 1 & 0 \\ 1 & 1/2 \\ 1 & 1 \end{bmatrix}$, and $\tilde{X}^\top \tilde{X} = \begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix}$. Also, $\tilde{X}^\top \boldsymbol{y} = \begin{bmatrix} 0 \\ 17/8 \end{bmatrix}$. So, we have $\begin{bmatrix} 5 & 0 \\ 0 & 2.5 \end{bmatrix} \tilde{\boldsymbol{w}} = \begin{bmatrix} 0 \\ 17/8 \end{bmatrix}$, that is

$$5b = 0$$
$$2.5w = 17/8\ .$$

Hence, $b = 0$ and $w = 17/20$.

Problem 3.2 [20pts] *(Ridge Regression)* Here, you will code a Matlab (or Octave) function that implements the Least Square (LS) algorithm seen in class and a generalization called Regularized Least Squares (RLS), or Ridge Regression. We will apply RLS to a real-world 8-dimensional (8 features) prostate cancer dataset contained in the file `prostateStnd.mat`. In this dataset, 8 medically relevant features named lcavol, lweight, age, lbph, svi, lcp, gleason, and pgg45 are used to estimate lpsa (log prostate specific antigen). The training and test data are provided as `(xtrain, ytrain)` and `(xtest, ytest)` respectively. The first 8 features correspond to the first 8 entries of `names`. The ninth entry of `names` (the last one) is the label to be predicted whose values are in `(ytest, ytrain)`.

Using the notation defined above, the LS problem is:

$$\min_{\tilde{\boldsymbol{w}}} \|\boldsymbol{y} - \tilde{X}\tilde{\boldsymbol{w}}\|^2 .$$

The RLS problem is:

$$\min_{\boldsymbol{w},b} \sum_{i=1}^{m} (\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)^2 + \lambda \|\boldsymbol{w}\|^2, \tag{1}$$

where λ is the regularization parameter. We still don't know what a regularizer is and why we should use it. Yet, here we will try to gather some practical intuition on it. You can see that LS is nothing else than RLS with $\lambda = 0$. Hence, we can just implement RLS.

(a) [4pts] As a first step, write Matlab code to normalize the training dataset so that post-normalization, each of the 8 features and the label in the normalized training dataset has zero mean and unit variance. This requires determining a *pair* of offset and scaling parameters, one pair for each feature and one pair for the label. These parameters must be computed only from the training dataset, but they must be applied to both the training and test datasets, i.e., we normalize both the training and test data, but we are only allowed to normalize the test data using parameters derived from the training data. In other words we must apply identical operations to training and test data. Only the training data will be actually normalized by the operation. If the test data is statistically similar to the training data, it too will be approximately normalized. The prototype must be

```
[XtrainNormalized, XtestNormalized] = normalize_data(Xtrain, Xtest)
```

(b) [4pts] Let $C = \tilde{X}^\top \tilde{X} + \lambda \tilde{I}$, where

$$\tilde{I} = \begin{bmatrix} 0 & \boldsymbol{0}^\top \\ \boldsymbol{0} & I \end{bmatrix},$$

$\boldsymbol{0}$ is the vector of all zeroes, and I the identity matrix. Show that the solution to the RLS problem satisfies

$$C\tilde{\boldsymbol{w}} = \tilde{X}^\top \boldsymbol{y}. \tag{2}$$

(c) [6pts] From the above, implement RLS with prototype

```
[w, b] = train_rls(X, y, lambda)
```

The code must be robust to the case that the matrix C is not invertible.

(d) [3pts] Use the normalized data to train a ridge regression model for each of the following values of the regularization penalty parameter λ: $\{e^{-5}, e^{-4}, e^{-3}, ..., e^{10}\}$. In a single figure, plot the ridge regression coefficient of each feature (8 in total) as a function of $\ln \lambda$ (8 curves in total) for $\ln \lambda$ ranging from -5 to 10 in steps of 1. Use suitable colors and/or markers to distinguish between the 8 curves and label them appropriately in a legend. Discuss what happens to the coefficients as λ becomes larger. (No code to submit here, just plots and your comments)

(e) [3pts] In another figure, plot the mean-squared-error (MSE) of both the training and the test data as a function of $\ln \lambda$. Discuss your observations. (No code to submit here, just plots and your comments)

Solution:

(a) [4pts] See the code.

(b) [4pts] We have to show that the solution of (1) satisfies (2). Indeed, taking the gradient of (1) and putting it equal to $\mathbf{0}$, we obtain (2). Let's see how. First, as we did in class for OLS, it is useful to transform (1) in matrix form. We would proceed as for OLS, the only difference is that we need a way to express $\|\boldsymbol{w}\|^2$ using $\tilde{\boldsymbol{w}}$. However, (2) suggests how to do it. In fact, it should be easy to see that

$$\lambda\|\boldsymbol{w}\|^2 = \tilde{\boldsymbol{w}}^\top \tilde{I} \tilde{\boldsymbol{w}} \ .$$

Also, from OLS, we know that

$$\sum_{i=1}^{m} (\boldsymbol{w}^\top \boldsymbol{x}_i + b - y_i)^2 = \|\tilde{X}\tilde{\boldsymbol{w}} - \boldsymbol{y}\|^2 \ .$$

Hence, we have that (1) is equal to

$$\lambda\tilde{\boldsymbol{w}}^\top \tilde{I} \tilde{\boldsymbol{w}} + \|\tilde{X}\tilde{\boldsymbol{w}} - \boldsymbol{y}\|^2 \ .$$

The gradient of this expression is

$$2\lambda\tilde{I}\tilde{\boldsymbol{w}} + 2\tilde{X}^\top(\tilde{X}\tilde{\boldsymbol{w}} - \boldsymbol{y}) = 2(\lambda\tilde{I} + \tilde{X}^\top\tilde{X})\tilde{\boldsymbol{w}} - 2\tilde{X}^\top\boldsymbol{y} \ .$$

Equating this expression to $\mathbf{0}$, we get (2).

(c) [6pts] From the previous point, we just have to solve the equation, to get

$$\tilde{\boldsymbol{w}} = pinv(\lambda\tilde{I} + \tilde{X}^\top\tilde{X})\tilde{X}^\top\boldsymbol{y},$$

where *pinv* is the pseudoinverse. See the code for its implementation.

(d) [3pts] See Fig. 1. As the regularization parameter λ increases, it dominates the objective function of the optimization problem that ridge regression attempts to solve and causes all feature weights to tend to zero. This happens because the argmin of $\|\boldsymbol{w}\|^2$ is $\mathbf{0}$.

(e) [3pts] As λ increases, the MSE in both training and test sets eventually increases and becomes large, see Fig. 2. This is because the ridge feature coefficients all move towards zero as we observed from part (d). The test MSE seems to attain a minimum at around $\ln \lambda = 3$ or 3.5. Recall that if $\lambda = 0$ then there is no regularization and ridge regression would reduce to ordinary least squares. Hence, RLS with a properly tuned λ can outperform OLS.

Problem 3.3 [17pts] In this problem, we will show some interesting properties of objective functions for regression with linear predictors. We consider a linear predictor parametrized by $\tilde{\boldsymbol{w}}$, that is $f_{\tilde{\boldsymbol{w}}}(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}^\top\tilde{\boldsymbol{w}}$.

(a) [5pts] Consider the squared loss on a single sample: $g_i(\tilde{\boldsymbol{w}}) = (y_i - \tilde{\boldsymbol{x}}_i^\top\tilde{\boldsymbol{w}})^2$. Prove that it is a convex function. Hint: Through the Hessian it might be the faster way.

(b) [5pts] Consider the absolute loss on a single sample: $g_i(\tilde{\boldsymbol{w}}) = |y_i - \tilde{\boldsymbol{x}}_i^\top\tilde{\boldsymbol{w}}|$. Prove that it is a convex function. Hint: observe that $|x| = \max(x, -x)$.

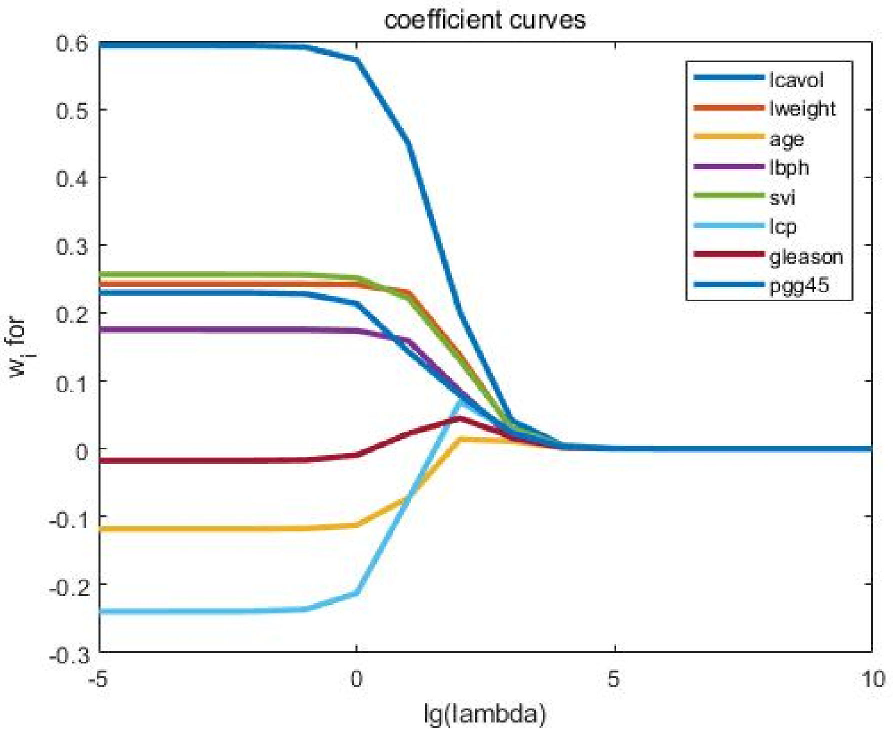

Figure 1: *Behavior of the ridge feature weights as regularization is increased.*

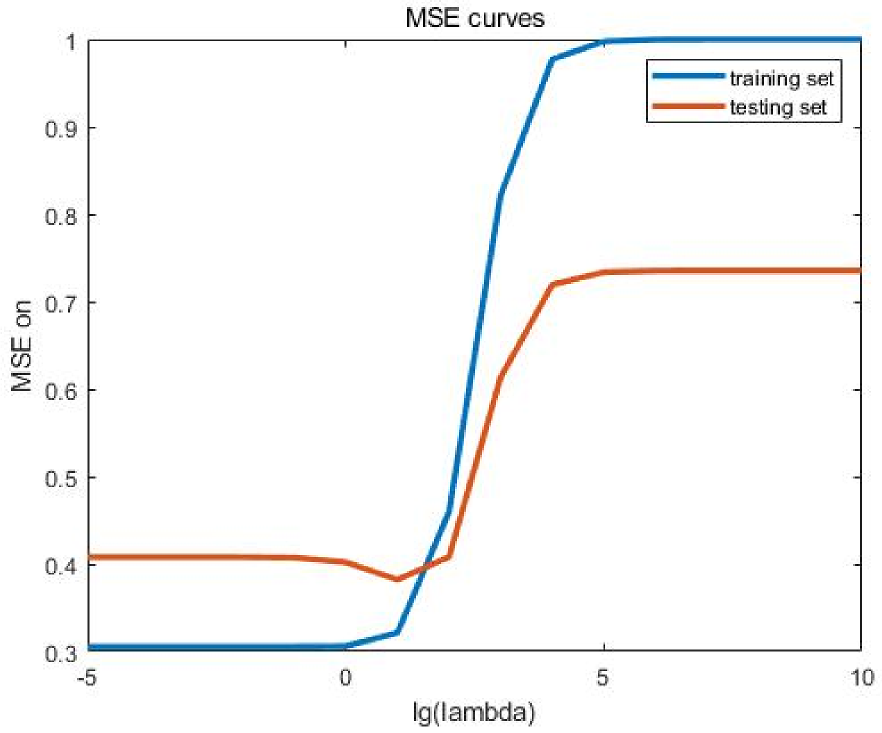

Figure 2: *Behavior of the MSE as the regularization is increased.*

(c) [5pts] ☕ Consider a generic loss function $\ell(\tilde{y}, y)$ twice differentiable and convex in the first argument. Let $g_i(\tilde{\boldsymbol{w}}) = \ell(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i)$ the loss of the linear predictor measured according to ℓ on i-th training sample. Prove that $g_i(\tilde{\boldsymbol{w}})$ is convex.

(d) [2pts] Prove that for any of the choices of loss functions above, the objective function $\frac{1}{m}\sum_{i=1}^m g_i(\tilde{\boldsymbol{w}})$ is convex.

Solution: Note that there are many valid ways to solve these problems. I'll explain some possible solutions.

(a) [5pts] Let's simplify the expression of the loss:

$$(y_i - \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}})^2 = \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}} \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}} - 2y_i \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}} + y_i^2 = \tilde{\boldsymbol{w}}^\top (\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top) \tilde{\boldsymbol{w}} - 2y_i \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}} + y_i^2 \ .$$

This is a quadratic function in $\tilde{\boldsymbol{w}}$, so the Hessian is $2\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top$. Matrices of the form $\boldsymbol{v}\boldsymbol{v}^\top$ are always PSD. To see it, it is enough to use the definition:

$$\boldsymbol{x}^\top (\boldsymbol{v}\boldsymbol{v}^\top) \boldsymbol{x} = (\boldsymbol{x}^\top \boldsymbol{v})^2 \geq 0 \ .$$

Hence, the function is convex.

(b) [5pts] Using the hint, we can write

$$g_i(\tilde{\boldsymbol{w}}) = |y_i - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{w}}| = \max(y_i - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{w}}, -(y_i - \tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{w}})) \ .$$

The two functions in the max are convex because linear (see hints on Piazza), and max of convex functions is always convex.

(c) [5pts] ☕ One possible approach for this problem is to calculate the Hessian of the loss function with respect to $\tilde{\boldsymbol{w}}$. Knowing how the Hessian work, it should be immediate to see that the Hessian is $\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top \ell''(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i)$. This Hessian is PSD because $\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top$ is PSD (see above) and $\ell'' \geq 0$ because ℓ is convex. Hence, $g_i(\tilde{\boldsymbol{w}})$ is convex.

For completeness, let's see how to calculate the Hessian in case you are not familiar with these kind of calculations. By definition, the elements (r, c) of the Hessian is $\frac{\partial^2 g}{\partial \tilde{w}_r \tilde{w}_c}$, where $\tilde{w}_r$ and $\tilde{w}_c$ are the coordinates r and c of $\tilde{\boldsymbol{w}}$ respectively. First, let's calculate the gradient using the chain rule:

$$\nabla g_i(\tilde{\boldsymbol{w}}) = \tilde{\boldsymbol{x}}_i \ell'(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i) = \begin{bmatrix} \tilde{x}_{i,1} \ell'(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i) \\ \tilde{x}_{i,2} \ell'(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i) \\ \dots \\ \tilde{x}_{i,d+1} \ell'(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i) \end{bmatrix}$$

where the derivative is with respect to the first argument of ℓ. Now, let's calculate the partial second derivatives using again the chain rule:

$$\frac{\partial^2 g}{\partial \tilde{w}_r \tilde{w}_c} = \frac{\partial}{\partial \tilde{w}_c} \left(\tilde{x}_{i,r} \ell'(\tilde{\boldsymbol{w}}_i^\top \tilde{\boldsymbol{w}}), y_i) \right) = \tilde{x}_{i,r} \ell''(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i) \tilde{x}_{i,c} \ .$$

From this is should be easy to see that the Hessian is $\tilde{\boldsymbol{x}}_i \tilde{\boldsymbol{x}}_i^\top \ell''(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{w}}, y_i)$ as we stated.

Another approach, probably easier: use directly the definition of convexity. Hence, we have to prove that the function g_i satisfies the definition of convexity, that is

$$g_i(\alpha \tilde{\boldsymbol{u}} + (1-\alpha)\tilde{\boldsymbol{v}}) = \ell(\tilde{\boldsymbol{x}}_i^\top (\alpha \tilde{\boldsymbol{u}} + (1-\alpha)\tilde{\boldsymbol{v}}), y_i)$$

$$
\begin{aligned}
&= \ell(\alpha \tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{u}} + (1-\alpha)\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{v}}, y_i) \\
&\le \alpha \ell(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{u}}, y_i) + (1-\alpha)\ell(\tilde{\boldsymbol{x}}_i^\top \tilde{\boldsymbol{v}}, y_i) \\
&= \alpha g_i(\tilde{\boldsymbol{u}}) + (1-\alpha) g_i(\tilde{\boldsymbol{v}}),
\end{aligned}
$$

where the inequality is due to the fact that ℓ is convex in the first argument.

(d) [2pts] Sum of convex functions is always convex.

Problem 3.4 [13pts] *(Perceptron)*
Here, we will implement and test the Perceptron algorithm on a real-world binary classification dataset.

Algorithm 1 Perceptron pseudocode.

```
Initialize: $\tilde{\boldsymbol{w}}_1 = \boldsymbol{0} \in \mathbb{R}^{d+1}$
for $i = 1, \dots, m$ do
    Pick sample $(\boldsymbol{x}_i, y_i)$ from $\mathcal{S}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$
    Construct augmented feature $\tilde{\boldsymbol{x}}_i$
    if $y_i \tilde{\boldsymbol{w}}_i^\top \tilde{\boldsymbol{x}}_i \le 0$ then
        $\tilde{\boldsymbol{w}}_{i+1} = \tilde{\boldsymbol{w}}_i + y_i \tilde{\boldsymbol{x}}_i$
    else
        $\tilde{\boldsymbol{w}}_{i+1} = \tilde{\boldsymbol{w}}_i$
    end if
end for
```

(a) [4pts] ☕ Consider the alternative Perceptron updates $\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \eta y_i \tilde{\boldsymbol{x}}_i$ with learning rate $\eta > 0$. The Perceptron algorithm is the special case $\eta = 1$. Consider this alternative Perceptron and the standard Perceptron on the same sequence of training samples. Prove that these two algorithm with make exactly the same mistakes regardless of the value of $\eta > 0$.

(b) [6pts] Implement the Perceptron algorithm in Algorithm 1. The prototype of the function must be

```
[w, b, average_w, average_b] = train_perceptron(X, y)
```

where `w` and `b` are the last solutions, while `average_w` and `average_b` are the averaged solutions

(c) [3pts] Now, let's test the Perceptron algorithm on the training data of Adult UCI dataset, where the binary classification task is to determine whether a person makes over 50K a year. The Perceptron has to do only one pass over the data. The `test_adult` will run your code and report the performance of the last solution and of the average solution on the test set, shuffling the training data and repeating the above 10 times. What do you observe? (No code to submit here, just numbers and your comments)

Solution:

(a) [4pts] ☕ Observe that the updates of the Perceptron depends only on the sign of the prediction, not on its magnitude. Also, the signs of the predictions won't change if we start from 0 and use any positive learning rate η. Hence, the behavior of the Perceptron algorithm will be exactly the same for any $\eta > 0$.

(b) [6pts] See code.

(c) [3pts] The performance of the averaged solution is more stable than the one of the last solution, in the sense that the variance of the test error is much smaller.

Code-submission via Blackboard: Create three dot-m files, named `normalize_data.m` for Problem 3.2(a), named `train_rls.m` for Problem 3.2(a), and one named `train_perceptron.m` for Problem 3.4(b). All local functions and scripts pertaining to one question should appear within the single dot-m file for that problem. Reach-out to the TAs via Piazza and their office/discussion hours for questions related to coding.