EC503 Introduction to Learning from Data — Boston University
Francesco Orabona — Spring 2021

Homework 3
Due: Tuesday, March 2, 3:30pm

Important: Submit the code on Blackboard and anything else on Gradescope.

Linear Regression and Model Selection

Problem 3.1

In this question, we consider once again linear regression. The input to the algorithms is the training set $\mathcal{S} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, for $i = 1, \dots, m$. Define the design matrix $X \in \mathbb{R}^{m \times d}$ with rows the inputs $\mathbf{x}_1^\top, \dots, \mathbf{x}_m^\top$ and the response vector $\mathbf{y} = (y_1, \dots, y_m)$. We want to fit a linear function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. As we did in class, define $\tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$ and the augmented design matrix $\tilde{X} = [\mathbf{1}\ X]$, where $\mathbf{1}$ is the vector of all ones, and set $\tilde{\mathbf{w}} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$. Let me also remind you that the Least Square optimization problem is:

$$\min_{\tilde{\mathbf{w}}} \ \|\mathbf{y} - \tilde{X}\tilde{\mathbf{w}}\|^2,$$

and the consider the Regularized Least Square optimization problem

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

In class, you learned about using k-folds cross validation as a way to estimate the true error of a learning algorithm and to tune parameters. If we use k equal to the number of training samples we obtain the *Leave-One-Out Cross Validation* (LOOCV), which provides an *almost unbiased* estimate of this true error. Unfortunately, it can take a really long time to compute the LOOCV. In this problem, you will derive a formula to efficiently compute the LOOCV error for RLS.

(a) Using the same notation of Homework 2, let $C = \tilde{X}^\top \tilde{X} + Q$, $\boldsymbol{d} = \tilde{X}^\top \mathbf{y}$, and $\tilde{\mathbf{w}}$ the solution of the RLS problem. Suppose we remove $\tilde{\mathbf{x}}_i$, from the training data, let $C_{(i)}, \boldsymbol{d}_{(i)}, \tilde{\mathbf{w}}_{(i)}$ be the corresponding matrices and vectors obtained after removing $\tilde{\mathbf{x}}_i$. Express $C_{(i)}$ in terms of C and $\tilde{\mathbf{x}}_i$. Express $\boldsymbol{d}_{(i)}$ in terms of $\boldsymbol{d}$, $\mathbf{x}_i$, and y_i.

(b) Express $C_{(i)}^{-1}$ in terms of C^{-1} and $\tilde{\mathbf{x}}_i$. *Hint:* use again the Sherman-Morrison formula.

(c) Show that

$$\tilde{\mathbf{w}}_{(i)} = \tilde{\mathbf{w}} + (C^{-1}\tilde{\mathbf{x}}_i) \frac{\tilde{\mathbf{x}}_i^\top \tilde{\mathbf{w}} - y_i}{1 - \tilde{\mathbf{x}}_i^\top C^{-1} \tilde{\mathbf{x}}_i} \ . \tag{2}$$

(d) Show that the leave-one-out residual, that is predicted value minus correct value, for removing the i^{th} training sample is

$$\tilde{\mathbf{w}}_{(i)}^\top \tilde{\mathbf{x}}_i - y_i = \frac{\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}_i - y_i}{1 - \tilde{\mathbf{x}}_i^\top C^{-1} \tilde{\mathbf{x}}_i} \ . \tag{3}$$

(e) The average LOOCV error is defined as: $\frac{1}{m}\sum_{i=1}^m (\tilde{\mathbf{w}}_{(i)}^\top \tilde{\mathbf{x}}_i - y_i)^2$. What is the algorithmic complexity of computing LOOCV error using the formula given in the previous point with respect to m and d? How is it compared with the naive way of computing LOOCV?

(f) Implement a new version of RLS with prototype

```
[w,b,train_err,loo_err] = train_rls_loo(X,y,lambda,epsilon)
```

where $\lambda > 0$, $\epsilon > 0$, `train_err` is the average training error (just the average error on the samples not the average error plus regularizer!), and `loo_err` is the average Leave-one-out error calculated with the formula above.

Problem 3.2 In this question, you have to implement linear regression from $\mathbb{R}^d$ to $\mathbb{R}$ using the square root to k-th root of a number as basis functions. In other words, for each coordinate i of the input, x_i, we generate new features $(x_i^{1/2}, x_i^{1/3}, \dots, x_i^{1/k})$ and we append them to the original features. Then, we learn a linear classifier in this space. This corresponds to learn a predictor of the form

$$\hat{y} = w_1 x_1 + \cdots + w_d x_d + w_{d+1} x_1^{1/2} + \dots w_{2d} x_d^{1/2} + \cdots + w_{d(k-1)+1} x_1^{1/k} + \cdots + w_{dk} x_d^{1/k} + b \ .$$

(a) Implement a function that generates a matrix of input samples that contains the roots of each feature, from the first root to k-th root, with prototype

```
[X_poly] = generate_poly_features(X,k)
```

(b) In the zip file there is also the "cadata" training/test data in the file "cadata_train_test.mat". It is a random train/test split of the Housing dataset from the UCI repository. The task is to predict the median house value from features describing a town. I normalized the features for you, to be in $[0, 1]$. Use the `cadata_tester.m` file to test your function to generate the roots features from $k = 1$ to $k = 10$, train 10 different RLS regressors using the function `train_rls_loo` with these features and $\lambda = 0.001$, $\epsilon = 1e-5$, and plot training error, test error, and the LOOCV error for each value of k. Plot and discuss the results: is this what you expected? Does it make sense? Why? Would the parameter picked by the LOOCV procedure give you good performance?

Perceptron

Problem 3.3

Given a set S of m samples $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ and the labels $y_i \in \{-1, 1\}$, the Perceptron algorithm runs as follows:

Algorithm 1 Perceptron pseudocode (with one pass over training set and averaging).

```
Input: Training samples $S = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$
Initialize: $\tilde{\mathbf{w}}_1 = \mathbf{0} \in \mathbb{R}^{d+1}$
for $i = 1, \ldots, m$ do
  Pick sample $(\tilde{\mathbf{x}}_i, y_i)$ from $S$
  if $y_i \langle \tilde{\mathbf{w}}_i, \tilde{\mathbf{x}}_i \rangle \leq 0$ then
    $\tilde{\mathbf{w}}_{i+1} \leftarrow \tilde{\mathbf{w}}_i + y_i \tilde{\mathbf{x}}_i$
  else
    $\tilde{\mathbf{w}}_{i+1} \leftarrow \tilde{\mathbf{w}}_i$
  end if
end for
return Last solution $\tilde{\mathbf{w}}$ and averaged solution $\frac{1}{m} \sum_{t=1}^{m+1} \tilde{\mathbf{w}}_t$
```

For simplicity, we have put the bias b into $\mathbf{w}$, that is $\tilde{\mathbf{w}} \leftarrow \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}$ and $\tilde{\mathbf{x}}_i \leftarrow \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$. So it is not necessary to consider the bias term in next two questions. Assume $\|\tilde{\mathbf{x}}_i\| \leq R$ for all i. We have shown that if the training samples are linearly separable with margin γ (that is, there exists $\tilde{\mathbf{w}}^*$ such that $\frac{y_i \langle \tilde{\mathbf{w}}^*, \tilde{\mathbf{x}}_i \rangle}{\|\tilde{\mathbf{w}}^*\|} \geq \gamma$ for all i), then the number of mistakes of the Perceptron is upper bounded by

$$\frac{R^2}{\gamma^2} \,. \tag{4}$$

(a) Consider the generalized Perceptron updates $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} + \eta y_i \tilde{\mathbf{x}}_i$ with *learning rate* $\eta > 0$. The Perceptron algorithm is the special case $\eta = 1$. Prove a bound on the number of mistakes similar to (4). How does η affect this bound?

(b) Implement the Perceptron algorithm in Algorithm 1. Note that it does only one pass over the training set and it returns both the last and average solution. Given that we don't run the algorithm till convergence, the averaged solution gives a little more stability to the algorithm. As usual $X \in \mathbb{R}^{m \times d}$ is the matrix of m input vectors in d dimension, i.e. each input $\mathbf{x}_i^\top$ is a row of X, y is a column vector of m rows containing the labels associated with the training samples. Learn `b` using the usual trick of augmenting the input data with a constant feature equal to 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) 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. Using the code in the point above, 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?

Code-submission via Blackboard: Create three dot-m files, `train_rls_loo.m`, `generate_poly_features.m`, `train_perceptron.m`. 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 3 you would submit a single directory named: `charles500_hw3.zip` which contains all the MATLAB code (and only the code). Reach-out to the TAs via Piazza and their office/discussion hours for questions related to coding.