EECS 545 HW 3

Due Wednesday, Oct. 13, by 11:59 PM Eastern Time via Gradescope

Please read carefully: When you upload your solutions to gradescope, please indicate which pages of your solution are associated with each problem. Please show all your work for all problems to receive full credit. This means that you should show all necessary steps so that we can reproduce your solution, for both coding and math/derivation problems. Also please note that your code for problems 1, 3, 4 and 5 should be uploaded to Gradescope (as a .pdf) and Canvas (as .py)

1. Ridge regression

In this problem, you will implement ridge regression on the Ames, Iowa Housing dataset (located on Canvas `hw3_iowa.zip`. The data is composed of 2000 train instances and 925 test instances. There are 58 features, and the target/response is home price in $1000. More details about the dataset can be found at `jse.amstat.org/v19n3/decock/DataDocumentation.txt`. We are using 55 of these features, as well as 3 additional features (`Total.SF, Total.Bath, Total.Porch.SF`). The file `housing_feature_names.npy` gives the feature names for corresponding indices in the data.

We will build a linear ridge regression model to predict house prices given the other features. Our objective function will be the penalized residual sum of squares (PRSS) loss function discussed in class:

$$PRSS = \sum_{j=1}^{n} (\mathbf{w}^T \mathbf{x}_j + w_0 - y_j)^2 + \lambda \|\mathbf{w}\|^2$$

where $\lambda > 0$, $\mathbf{w} = [w_1, \ldots, w_p]^T$ is the vector of weights, not including the offset term w_0, and $\mathbf{x}_j$ is the j-th feature vector of dimension d. The solution $\mathbf{w} = \mathbf{w}^*$ to the PRSS for a particular value of λ is called the optimal ridge weight vector for that λ value. The data is split into training and test samples.

Note: The above description uses notation from Section 001. In section 002, b replaces w_0, and the residual sum of squares has a leading factor of $\frac{1}{n}$. These versions of ridge regression are equivalent by substituting $\lambda \to n\lambda$. Your solution should follow the notations and conventions stated above.

(a). Sphere the feature variables in the training data by transforming the $d \times n$ feature matrix $\mathbb{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$ so that each of its rows have zero sample mean and have sample variance equal to one. Specifically, for each feature i, compute the sample mean $\hat{\mu}_i = n^{-1} \sum_{j=1}^{n} x_{ij}$ and the sample variance $\hat{\sigma}_i^2 = n^{-1} \sum_{j=1}^{n} (x_{ij} - \hat{\mu}_i)^2$, and then replace x_{ij} by $(x_{ij} - \hat{\mu}_i)/\hat{\sigma}_i$, $i = 1, \ldots, d$, $j = 1, \ldots, n$. Verify that the rows of the thus sphered $\mathbb{X}$ training data matrix have zero sample mean and unit variance.

Submit plots of the vector $\hat{\boldsymbol{\mu}}$ of sample means and the vector $\hat{\boldsymbol{\sigma}}$ of sample standard deviations.

(b). Learn the optimal ridge regression parameters on training data. Using your sphered training data compute the optimal ridge regression coefficients $\mathbf{w}$ and offset w_0 that minimize the PRSS for $\lambda = 100$. Report the offset w_0 and **the first five ridge regression coefficients** w_1**,** w_2**,** w_3**,** w_4**,** w_5.

(c). Next, we will see what happens when we vary λ. For 200 evenly spaced samples of λ ranging between 0 and 10^5, compute the d ridge regression coefficients and offset as we did in part (b). Then plot the trajectories of the d ridge coefficients $\mathbf{w}$ on a single plot as a function of λ (do not plot the offset w_0).

On a separate figure, use the training data to plot $MSE = \frac{1}{n}\sum_{j=1}^{n}(\mathbf{w}^T\mathbf{x}_j + w_0 - y_j)^2$ as λ varies over this range. Submit both the weight trajectories and training MSE plots.

(d). Finally, we will compare error on the training and test sets. First, obtain the ridge regression coefficients and offsets for 200 evenly spaced values of λ between 1 and 100. For each λ, evaluate MSE on the training and test data.

Submit a plot which shows both the train and test MSE as a function of λ. Compare the trends for train and test, and briefly comment on similarities/differences. Report the λ which minimizes the train MSE, and the λ which minimizes test MSE.

Note that, prior to applying the ridge predictor, for each test sample feature i you will need to subtract the value $\hat{\mu}_i$ and divide by the value $\hat{\sigma}_i$, determined in part (a).

2. Optimal soft-margin hyperplane

Let $(\boldsymbol{w}^*, b^*, \boldsymbol{\xi}^*)$ denote the solution to the soft-margin hyperplane quadratic program. (Note that the parameter b is denoted w_0, and that the vectors $\boldsymbol{w}$ and $\boldsymbol{\xi}$ are a different font in Sec. 001 lectures.)

a. Show that if $\boldsymbol{x}_i$ is misclassified by the optimal soft-margin hyperplane classifier, then $\xi_i^* \geq 1$. Conclude that $\frac{1}{n}\sum_i \xi_i^*$ upper bounds the training error. Hence, the OSM objective is balancing margin maximization with minimizing a bound on the training error.

b. Show that if $\xi_i^* > 0$, then ξ_i^* is proportional to the distance from $\boldsymbol{x}_i$ to the margin hyperplane associated with class y_i (that is, the set $\{\boldsymbol{x} : (\boldsymbol{w}^*)^T\boldsymbol{x} + b^* = y_i\}$), and give the constant of proportionality.

3. Subgradient methods for the optimal soft margin hyperplane

In this problem you will implement the subgradient and stochastic subgradient methods for minimizing the convex but nondifferentiable function

$$J(\mathbf{w}, b) = \frac{1}{n}\sum_{i=1}^{n}\left(L(y_i, \mathbf{w}^T\mathbf{x}_i + b) + \frac{\lambda}{2}\|\mathbf{w}\|^2\right)$$

where $L(y, t) = \max\{0, 1 - yt\}$ is the hinge loss. As we saw in class, this corresponds to the optimal soft margin hyperplane classifier.

(a) Determine $J_i(\mathbf{w}, b)$ such that

$$J(\mathbf{w}, b) = \sum_{i=1}^{n} J_i(\mathbf{w}, b).$$

Determine a subgradient $\boldsymbol{u}_i$ of each J_i with respect to the variable $\boldsymbol{\theta} = [b\ \mathbf{w}^T]^T$. A subgradient of J is then $\sum_i \mathbf{u}_i$.

Note: Recall the chain rule for subdifferentials discussed in class: If $f(\mathbf{z}) = g(h(\mathbf{z}))$ where $g : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}$, and both g and h are differentiable, then

$$\nabla f(\mathbf{z}) = \nabla h(\mathbf{z}) \cdot g'(h(\mathbf{z})).$$

If g is convex and h is differentiable, the same formula gives a subgradient of f at $\mathbf{z}$, where $g'(h(\mathbf{z}))$ is replaced by a subgradient of g at $h(\mathbf{z})$.

Download the file `hw3_pulsars.zip` from canvas. The data for this binary classification problem consists of features which are statistics of radio signals from stars, and the binary classes denote whether the signal came from a pulsar or not. For more details on the dataset, see: `archive.ics.uci.edu/ml/datasets/HTRU2`.

`pulsar_features.npy` contains a $d \times n$ matrix of features, where $d = 2$ features and $n = 3278$ samples (note we are only using the 1st and 7th features from the original HTRU2 dataset). `pulsar_labels.npy` contains an n vector of labels where -1 denotes not a pulsar and 1 denotes a pulsar.

(b) Implement the subgradient method for minimizing J and apply it to the nuclear data. Submit two figures: One showing the data and the learned line, the other showing J as a function of iteration number. Also report the estimated hyperplane parameters, the margin, and the minimum achieved value of the objective function.

Use the starter code in `hw3_p3.py` to visualize the data and seed the random number generator.

Comments:

- Use $\lambda = 0.001$.
- Use a step-size of $\eta_j = 100/j$, where j is the iteration number.
- To compute the subgradient of J, write a subroutine to find the subgradient of J_i, and then sum those results.
- Perform 10 iterations of gradient descent.
- Initialize the hyperplane parameters to zeros before training.
- Debugging goes faster if you just look at a subsample of the data. You can also use the Python debugger pdb: `https://realpython.com/python-debugging-pdb/`.

(c) Now implement the *stochastic subgradient* (SGD) method, which is like the subgradient method, except that your step direction is a subgradient of a randomly selected J_i, not J. Be sure to cycle through all data points before starting a new loop through the data.

Report/hand in the same items as for part **(b)**. In addition, comment on the (empirical) rate of convergence of the stochastic subgradient method relative to the subgradient method. Explain your findings.

More comments:

- Use the same λ, η_j, and initialization as in part **(b)**. Here j indexes the number of times you have cycled (randomly) through the data.
- Cycle through the data 10 times (i.e. perform $10n$ steps of stochastic subgradient descent).
- To save time, you do not need to compute J after every update, as that would result in too many computations. You should compute J after every iteration through the data points.
- To generate a random permutation use

  `np.random.permutation`
- Submit all code to Canvas (.py) and Gradescope (.pdf).

4. Coordinate Descent Ridge Regression

Coordinate descent (CD) cycles through successive minimizations of the objective function $F(\mathbf{w})$ with respect to each coordinate:

$$w_i = \text{argmin}_w F(w_1, \ldots, w_{i-1}, w, w_{i+1}, \ldots, w_p), i = 1, 2 \ldots, p, 1, 2, \ldots,$$

Here you will implement CD for ridge regression. The ridge regression objective function is

$$F_{ridge}(\mathbf{w}) = \sum_{j=1}^{n} (y_j - \hat{y}_j(\mathbf{w}))^2 + \lambda \sum_{i=1}^{p} |w_i|^2,$$

where $\lambda > 0$ and $\hat{y}_j(\mathbf{w}) = w_0 + \sum_{i=1}^{p} w_i x_{ij} = w_0 + \mathbf{x}^T \mathbf{w}_1$. As usual x_{ij} is the ij element of the $p \times n$ feature matrix $\mathbb{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$.

(a). Assume that the features have been adjusted so that they have zero mean, i.e., $\mathbb{X}\mathbf{1} = \sum_{j=1}^{n} \mathbf{x}_j = 0$, which will be true when you sphere the training data (see Problem 1). Show that the w_0 that minimizes F_{ridge} is $w_0^* = \frac{1}{n} \sum_{j=1}^{n} y_j$.

(b). Show that the partial derivative with respect to w_i, $i = 1, \ldots, d$, of the data-dependent term in F_{ridge} is

$$\frac{\partial}{\partial w_i} \sum_{j=1}^{n} (y_j - \hat{y}_j(\mathbf{w}))^2 = a_i w_i - c_i$$

where

$$a_i = 2 \sum_{j=1}^{n} x_{ij}^2, \quad c_i = 2 \sum_{j=1}^{n} x_{ij} (y_j - \hat{y}_j(\mathbf{w}_{-i})), \quad i \geq 1$$

with $\hat{y}_j(\mathbf{w}_{-i})$ the predictor with coefficient w_i set to zero, i.e.,

$$\hat{y}_j(\mathbf{w}_{-i}) = w_0 + \sum_{k \neq i}^{p} w_k x_{kj}, \quad i \geq 1.$$

where the summation is over $k = 1, \ldots, d$, excluding index i.

(c). The optimality condition for w_i to minimize the ith coordinate of $F_{ridge}(\mathbf{w})$ is $\frac{\partial F(\mathbf{w})}{\partial w_i} = 0$, $i = 1, \ldots, d$. By solving this equation for w_i show that the CD update of w_i is

$$\text{argmin}_{w_i} F_{ridge}(\mathbf{w}) = w_i = \frac{c_i}{a_i + 2\lambda}, \quad i \geq 1.$$

(d). Here you will implement the coordinate descent ridge regression algorithm and apply it to the **sphered** Ames Iowa Housing data you treated in Problem 1. Make sure you include the y-intercept weight $w_0 = \frac{1}{n} \sum_{j=1}^{n} y_j$ in the data dependent term but exclude it from the penalty term in $F_{ridge}(\mathbf{w})$. Using the ridge regularization parameter $\lambda = 100$ and initializing your $\mathbf{w}$ to be a vector with all elements equal to 1, run your CD ridge regression for 2900 iterations (50 cycles).

Plot the evolution of the weights (a single plot of all weights except for w_0 over iteration), and also plot the learning curve with respect to average squared prediction error $\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j(\mathbf{w}^{train}))^2$ ($n = 2000$) on the training data. The label on the x-axis should be number of iterations.

Hint: Your weights should converge to the exact ridge weights implemented in Problem 1 when $\lambda = 100$.

5. Coordinate Descent Lasso Regression. Here you will derive and implement the lasso "shooting method" discussed in class. The lasso objective function is:

$$F_{lasso}(\mathbf{w}) = \sum_{j=1}^{n}(y_j - \hat{y}_j(\mathbf{w}))^2 + \lambda \sum_{i=1}^{p} |w_i|$$

where $\lambda > 0$ and $\hat{y}_j(\mathbf{w}) = w_0 + \sum_{i=1}^{p} w_i x_{ij} = w_0 + \mathbf{x}^T \mathbf{w}_1$. As usual x_{ij} is the ij element of the $p \times n$ feature matrix $\mathbb{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]$.

(a). As $|w_i|$ is not differentiable, the optimality condition for w_i to minimize the ith coordinate of $F_{lasso}(\mathbf{w})$ is that $\mathbf{0} \in \partial_{w_i} F_{lasso}(\mathbf{w})$, where $\partial_{w_i} F_{lasso}(\mathbf{w})$ is the subdifferential with respect to the i-th coordinate w_i, $i = 1, \ldots, d$. We stated in class that the subdifferential is

$$\partial_{w_i} F_{lasso}(\mathbf{w}) = a_i w_i - c_i + \lambda \partial_{w_i} |w_i|, \quad i \geq 1.$$

We also stated that the solution w_i to the subdifferential equation $\mathbf{0} \in \partial_{w_i} F_{lasso}(\mathbf{w})$ is the soft thresholded w_i for $i \geq 1$

$$w_i = \text{soft}\left(\frac{c_i}{a_i}; \frac{\lambda}{a_i}\right) = \begin{cases} \frac{c_i - \lambda}{a_i}, & c_i > \lambda \\ 0, & c_i \in [-\lambda, \lambda] \\ \frac{c_i + \lambda}{a_i}, & c_i < -\lambda \end{cases}$$

where $\text{soft}\,(a; \delta) = \text{sign}(a) \max\{0, |a| - \delta\}$. By checking all three cases of c_i in the above expression, verify that if w_i is so specified it satisfies the subdifferential optimality condition

$$\mathbf{0} \in \partial_{w_i} F_{lasso}(\mathbf{w}).$$

Hint: use the definition of the subdifferential of $|w|$ (derived in class)

$$\partial_w |w| = \begin{cases} 1, & w > 0 \\ [-1, 1], & w = 0 \\ -1, & w < 0 \end{cases}$$

(b). Using the results of part (a) and the fact (recall part (a) of Problem 4) that for sphered data the optimum offset weight w_0 that minimizes F_{lasso} is $w_0^* = \frac{1}{n} \sum_{j=1}^{n} y_j$, implement the CD Lasso regression in python. Run your CD Lasso regression code on the **sphered** data you created in Problem 1 for 2900 iterations with $\lambda = 100$ and with $\mathbf{w}$ initialized to be a vector of all 1's.

As in Problem 4, plot the weight trajectories over iteration, plot the learning curve (MSE on the training data). Also report final test MSE.

Comment on the differences between the results of the ridge regression solution and the CD lasso solution. Are there any values of the lasso weight vector that have been thresholded to zero after 2900 iterations?