Homework II

Due date: 09/22/2024 at 11:59pm

Submission Guidelines: 1. Put all the documents into one folder, name that folder as firstnamelastname_UIN and compress it into a .zip file.

2. For coding problems, your submission should include a code file (either .py or .ipynb), and also a pdf file (in one file together with other non-coding questions) to report the results that are required in the question.

Problem 1: Regression with Ambiguous Data (25 points)

In the regression model we talked about in class, we assume that for each training data point $\mathbf{x}_i$, its output value y_i is observed. However in some situations that we can not measure the exact value of y_i. Instead we only have information about if y_i is larger or less than some value z_i. More specifically, the training data is given as a triplet $(\mathbf{x}_i, z_i, b_i)$, where

- $\mathbf{x}_i$ is represented by a vector $\phi(\mathbf{x}_i) = (\phi_0(\mathbf{x}_i), \ldots, \phi_{M-1}(\mathbf{x}_i))^\top$
- $z_i \in \mathbb{R}$ is a scalar, $b_i \in \{0, 1\}$ is a binary variable indicating that if the true output y_i is larger than z_i ($b_i = 1$) or not $b_i = 0$

Develop a regression model for the ambiguous training data $(\mathbf{x}_i, z_i, b_i), i = 1, \ldots, n$.

Hint: Define a Gaussian noise model for y and derive a log-likelihood for the observed data. You can derive the objective function using the error function given below (note that there is no closed-form solution). The error function is defined as

$$\text{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} e^{-t^2} dt$$

It is known that

$$\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right], \text{ and } \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-t^2/2} dt = \frac{1}{2}\left[1 - \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$$

Problem 2: Ridge Regression with a Prior (20 points)

In class, we interpret Ridget Regression as MAP with a Gaussian prior $\Pr(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0, \beta I)$. What if we have a prior knowledge that $\mathbf{w}$ should be concentrated around $\mathbf{w}_0$. Thus, we can assume a Gaussian prior $\Pr(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \beta I)$. Please write down the objective of MAP and derive the MAP solution in a closed form using this prior.

Problem 3: Ridge Regression (40 points)

In this problem[1], you are asked to learn regression models using Ridge regression and Lasso. The data set that we are going to use is the E2006-tfidf[2].

The first column is the target output y, and the remaining columns are features in the form of (feature_index:feature_value). You can load the data by sklearn[3]. If we let $\mathbf{x} \in \mathbb{R}^d$ denote the feature vector, the prediction is given by $\mathbf{w}^\top \mathbf{x} + w_0$, where $\mathbf{w} \in \mathbb{R}^d$ contains the coefficients for all features and w_0 is a intercept term. Denoting $\mathbf{X}^\top = (\mathbf{x}_1, ..., \mathbf{x}_n)$, the Ridge regression problem becomes

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{X}\mathbf{w} + w_0 - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

and the Lasso regression problem becomes

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{X}\mathbf{w} + w_0 - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_1,$$

You can use the Python sklearn library for Ridge regression[4].

(1) Solution of Ridge Regression and Lasso: Set the value of the regularization parameter $\lambda = 0.1$, compute the optimal solution for Ridge regression and Lasso. Report the number of nonzero coefficient in the solution $\mathbf{w}$ for both Ridge regression and Lasso. You may observe that the solutions of Ridge regression and Lasso contain very different numbers of nonzero elements. What is the cause of that? What can this imply? Justify your observation and hypothesis. (Note: If you use the sklearn Lasso, the value of alpha should be set to λ/n, where n is the number of training examples, and in the sklearn Ridge, set alpha to be λ. Same for following questions.)

(2) Training and testing error with different values of λ: (i) For each value of λ in [0, 1e-5, 1e-3, 1e-2 , 0.1, 1, 10, 100, 1e3, 1e4, 1e5, 1e6] run the Ridge regression and Lasso on training data to obtain a model $\mathbf{w}$ and then compute the root mean square error (RMSE[5]) on both the training and the testing data of the obtained model. (ii) Plot the error curves for root mean square error on both the training data and the testing data vs different values of λ. You need to show the curves, and discuss your observations of the error curves, and report the best value of λ and the corresponding testing error. (iii) Plot the curve of number of nonzero elements in the solution $\mathbf{w}$ vs different values of λ. Discuss your observations. (iv) Plot the curve of $\|\mathbf{w}\|_2^2$ vs different values of λ. Discuss your observations.

(3) Cross-validation: Use the given training data and follow the 5-fold cross-validation procedure to select the best value of λ for both Ridge regression and Lasso. Then train the model on the whole training data using the selected λ and compute the root mean square error on the testing data. Report the best λ and the testing error for both Ridge regression and Lasso.

[1] Please refer to the code base "Ridge_Lasso_Regression.py"

[2] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html`

[3] `https://scikit-learn.org/stable/datasets/loading_other_datasets.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html`

[5] For a set of examples $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, the root mean square error of a prediction function $f(\cdot)$ is computed by RMSE $= \sqrt{\sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2/n}$.

Problem 4: SGD for Logistic Regression(15 points)

In this problem you are asked to run SGD for optimizing the objective of logistic regression. You need to run SGD with different learning rate schedulers and compare their convergence performance. The objective of logistic regression on a set of training examples $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, where $y_i \in \{1, -1\}, \mathbf{x}_i \in \mathbb{R}^d$:

$$\min_{\mathbf{w}} L(\mathbf{w}) := \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-\mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i))$$

We will use breast-cancer dataset and try 3 schedulers (step decay, cosine decay, and polynomial decay). The SGD update is given by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \frac{1}{B} \sum_{i=1}^{B} \nabla_{\mathbf{w}} \log(1 + \exp(-\mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i)), t = 1, \ldots, T$$

where $\nabla_{\mathbf{w}} \log(1 + \exp(-\mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i)) = -\frac{\exp(-y_i \mathbf{w}^\top \mathbf{x}_i)}{1+\exp(-\mathbf{y}_i \mathbf{w}^\top \mathbf{x}_i)} y_i \mathbf{x}_i$ is the gradient of the logistic loss.

You need to implement SGD with different learning rate schedulers and compare their convergence curves. The convergence curve is the curve of the training objective vs the number of iterations. This can be plotted by recording the objective value after each epoch (an epoch means a full pass of the dataset). Different learning rate schedules are described below.

- Denote η_0 as the initial learning rate.
- Step decay: Decay the learning rate only at several specific iterations and keep unchanged at other iterations. (Got the name from its curve looking like steps)

  $$\eta_t = \eta_0 \gamma^{\lfloor \frac{t}{t_0} \rfloor},$$

  where $\gamma < 1$ is the decay rate and t_0 is the number of iterations after which the learning rate is decreased by a factor of γ.
- Cosine decay: Decay the learning rate like a cosine curve.

  $$\eta_t = \eta_{min} + \frac{1}{2}(\eta_0 - \eta_{min})(1 + \cos \frac{t\pi}{T})$$

  where η_{min} is the minimum learning rate.
- Polynomial decay: Decay the learning rate like a polynomial curve.

  $$\eta_t = \frac{\eta_0}{t^\alpha}$$

Hyper parameters:

- initial learning rate η_0: Tune them from $[0.1, 0.3, 1, 3, 10]$.
- minimum learning rate η_{min}: 0.001

- total epoch: 30 (Note that here the total number of iterations is $T = 30 * \lceil \frac{\text{Trainset Size}}{\text{batch size}} \rceil$)
- t_0: $15 * \lceil \frac{\text{Trainset Size}}{\text{batch size}} \rceil$ This means that learning rate decays every 15 epochs.
- γ: 10
- α: 0.5
- batch size: 16

Requirements:

- For each learning rate schedule, you need to plot the convergence curves for different initial learning rates.
- Compare the best convergence of each learning rate schedule and plot them in a single figure.