STAT 37710: Machine Learning Autumn 2022

Homework 2

Due date: 11:59pm on Monday Oct. 24th

1. MAP (*20 points*)

Assume that for each $1 \le i \le n$, one has $y_i = w^\top x_i + \varepsilon_i$ with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.

a.(*10 points*) Assume a prior $w \sim \mathcal{N}(0, r^2 I_d)$, where I_d is the identity matrix in $d \times d$ dimension. Show that the MAP estimator in this case is equivalent to the ridge regression estimator. Please be precise about the choice of the regularization parameter λ.

b.(*10 points*) Assume a prior $p(w) = (\tau/2)^d \exp(-\tau \|w\|_1)$, where τ is a parameter for this prior. Show that the MAP estimator in this case is equivalent to the Lasso estimator. Please be precise about the choice of the regularization parameter λ.

2. Regression function and Bayes classifier (*20 points*)

Consider a binary classification problem with $\mathcal{Y} = \{0, 1\}$. Let $g^\star(x)$ be the Bayes optimal classifier, and $m^\star(x)$ be the optimal regression function. Let $\hat{m}$ be a fixed function from $\mathcal{X}$ to $\mathbb{R}$. Define the plug-in decision $\hat{g}$ by

$$\hat{g}(x) = \begin{cases} 1, & \text{if } \hat{m}(x) \ge 1/2, \\ 0, & \text{if } \hat{m}(x) < 1/2. \end{cases}$$

Prove the following statements.

$$\begin{aligned} 0 &\le \mathbb{P}(\hat{g}(X) \ne Y) - \mathbb{P}(g^\star(X) \ne Y) \\ &\le 2\mathbb{E}_X[|\hat{m}(X) - m^\star(X)|] \le 2(\mathbb{E}_X[|\hat{m}(X) - m^\star(X)|^2])^{1/2}. \end{aligned}$$

3. Multi-class logistic regression (*20 points*)

The posterior probabilities for mulitclass logistic regression can be given as a softmax transformation of hyperplanes, such that:

$$P(Y = k \mid \mathbf{X} = \mathbf{x}) = \frac{\exp(\mathbf{a}_k^\top \mathbf{x})}{\sum_j \exp(\mathbf{a}_j^\top \mathbf{x})}$$

If we consider the use of maximum likelihood to determine the parameters $\mathbf{a}_k$, we can take the negative logarithm of the likelihood function to obtain the *cross-entropy* error function for multiclass logistic regression:

$$E(\mathbf{a}_1, \ldots, \mathbf{a}_K) = -\ln \left(\prod_{n=1}^{N} \prod_{k=1}^{K} P(Y = k \mid \mathbf{X} = \mathbf{x}_n)^{t_{nk}} \right) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

where $t_{nk} = 1\{\text{labelOf}(\mathbf{x}_n) = k\}$, and $y_{nk} = P(Y = k \mid \mathbf{X} = \mathbf{x}_n)$.

a.(*10 points*) For $j \in \{1, \ldots, K\}$, prove that

$$\frac{\partial t_{nk} \ln y_{nk}}{\partial \mathbf{a}_j} = t_{nk}(1\{k = j\} - y_{nj})\mathbf{x}_n$$

b.(*10 points*) Based on the result in (a), show that the gradient of the error function can be stated as given below

$$\nabla_{\mathbf{a}_k} E(\mathbf{a}_1, \ldots, \mathbf{a}_K) = \sum_{n=1}^{N} [y_{nk} - t_{nk}]\, \mathbf{x}_n$$

4. Programming assignment: Lasso (*40 points*)

In this question, you are required to fit data with a Lasso regression. Recall that the Lasso objective is to minimize $\text{RSS}(\beta) + \lambda \sum_i \beta_i$. Following the script below here, you will be able to generate the training and testing data.

```
#python
import numpy as np
np.random.seed(0)
N_fold = 10
N_test = 500
N_train = 1000
N = N_test + N_train
# Specify feature dimensions of X and Y
X_dim = 20
Y_dim = 10
X = np.random.randn(N,X_dim)

# Only have 10 non-zero entries in beta,
nnz = 10
beta = np.zeros((X_dim * Y_dim))
nnz_idx = np.random.choice(X_dim * Y_dim, nnz, replace = False)
beta[nnz_idx] = np.random.randn(nnz) * 2

beta = beta.reshape(X_dim, Y_dim)
Y =  X @ beta + np.random.rand(N, Y_dim)

# Split training and testing set
X_test = X[:N_test]
Y_test = Y[:N_test]
X_train = X[N_test:]
Y_train = Y[N_test:]
```

a.(*20 points*) Write a function to fit the Lasso regression on the training data and calculate the MSE on the training set. Choosing λ from 0 to 0.04 (with a step of 0.001), compute the estimate $\hat{y}$ for different values λ, and plot the MSE as a function of λ.

b.(*20 points*) Implement 10-fold cross validation on the training set to select λ. Plot and compare the MSE on the hold-out set with the true MSE which is computed on the test set. And see how we get to finding the "best" λ.