IEOR 290 002 - Spring 2017
Homework 3
Due 02/27/2017 at 11:59 pm

Instruction

- We suggest you type your answer using LaTeX or Microsoft Word.
- This homework contains both theoretical and coding parts. For questions with `[coding]` mark, you need to write a small program to answer the question. We recommend you use Python, R or MATLAB to do the problem. Append your code after the answer.
- Submit a single PDF of your write-up on bCourses. Make sure your answer is legible.

1. **(MLE of multivariate normal distribution)**

   Let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^d$ be n independent samples drawn from the following multivariate normal distributions:

   (a) $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and diagonal covariance matrix $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}_{d\times d}$

   (b) $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and diagonal covariance matrix

   $$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_d^2 \end{bmatrix}$$

   Compute the maximum likelihood estimator for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and compute the Fisher information matrix for part (a).

2. **(Weighted least squares)**

   Consider a dataset $(\boldsymbol{x}_i, t_i)_{i=1,\ldots,n}$ which each data is associated with a weighting factor β_i, so the sum-of-squares error function is

   $$f(\boldsymbol{w}) = \sum_{i=1}^{n} \beta_i (t_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2$$

   Write $f(\boldsymbol{w})$ in matrix form using

   $\boldsymbol{X}$: $n \times d$ design matrix
   $\boldsymbol{t}$: $n \times 1$ target vector
   $\boldsymbol{B}$: $n \times n$ diagonal matrix with β_i on the diagonal

   and find an expression for the solution $\boldsymbol{w}^*$.

3. **(Linear regression on MNIST dataset)**

`[Coding]` In this problem we will apply the least squares linear regression model on the hand-written digit recognition MNIST dataset.

The dataset can be downloaded from bCourses. It contains four files: 60000 training images, 60000 training labels, 10000 testing images and 10000 testing labels. Images are originally 28 pixel by 28 pixel grey scale image of a hand-written digit from 0 to 9. The actual data $\boldsymbol{x}_i$ is flattened into a $28 \times 28 = 784$ dimensional vector. Label is a single number shows the true digit of the corresponding image represents.

To train a least squares linear regression model, we solve

$$\boldsymbol{W}^* \quad = \quad \arg\min_{\boldsymbol{W}} \sum_{i=1}^{n} \|\boldsymbol{W}^\top \boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 \tag{1}$$

where

n:	Number of samples ($= 60000$)
k:	Number of classes ($= 10$)
$\boldsymbol{x}_i$:	784×1 vector denotes a single data point
$\boldsymbol{y}_i$:	10×1 vector denotes the one-hot encoding of the label, i.e. map $j = \{0, 1, \ldots, 9\}$ to the standard basis vector $\boldsymbol{e}_j$.
$\boldsymbol{W}$:	784×10 matrix denotes the regression coefficients

By convention the above optimization problem usually is written as

$$\boldsymbol{W}^* \quad = \quad \arg\min_{\boldsymbol{W}} \|\boldsymbol{X}\boldsymbol{W} - \boldsymbol{Y}\|_F^2 \tag{2}$$

where

$\boldsymbol{X}$:	60000×784 matrix where each row represents a single data $\boldsymbol{x}_i^\top$
$\boldsymbol{Y}$:	60000×10 matrix where each row represents a single encoded label $\boldsymbol{y}_i^\top$
$\|\cdot\|_F$:	denote the Frobenius norm *[wiki]*

Usually machine learning package takes $\boldsymbol{X}$ and $\boldsymbol{Y}$ as input.

In this problem, you need to use machine learning package to perform the following tasks:

- Read the data
- One-hot encode the labels to $\boldsymbol{y}_i$
- Train the model using least squares regression
- Make predictions on both training and testing data
- Decode the predicted $\boldsymbol{y}_i$ to labels

- Report the training and testing accuracy
- Find the first mis-classified data in the testing set and visualize the data as an image. What is the predicted digit using linear regression? What is your own prediction before seeing the true label? Are you doing better than the linear regression model?

Skeleton code using Python with `scikit-learn` package is provided on the next page.

```python
from mnist import MNIST
import numpy as np
import matplotlib.pyplot as plt
import sklearn.metrics as metrics
from sklearn import linear_model
from sklearn.preprocessing import OneHotEncoder


def load_dataset():
    mndata = MNIST('./data/')
    X_train, labels_train = map(np.array, mndata.load_training())
    X_test, labels_test = map(np.array, mndata.load_testing())
    X_train = X_train/255.0
    X_test = X_test/255.0

    X_train = np.asarray(X_train).reshape(60000, 784)
    labels_train = np.asarray(labels_train).reshape(60000, 1)
    X_test = np.asarray(X_test).reshape(10000, 784)
    labels_test = np.asarray(labels_test).reshape(10000, 1)

    return X_train, labels_train, X_test, labels_test

def plot(x):
    image = x.reshape(28, 28)
    plt.imshow(image, cmap='grey')
    plt.axis('off')

    plt.show()

if __name__ == "__main__":

    #==== Read data ====
    X_train, labels_train, X_test, labels_test = load_dataset()

    #==== One hot encode ====
    # (Use sklearn.preprocessing.OneHotEncoder)

    #==== Train ====
    # (Use sklearn.linear_model.LinearRegression)

    #==== Predict ====
    # (Use sklearn.linear_model.LinearRegression)

    #==== Decode ====
    # (Use numpy.argmax)

    #==== Print accuracy ====
    print("Train accuracy: {0}".format(metrics.accuracy_score(labels_train, pred_labels_train)))
    print("Test accuracy: {0}".format(metrics.accuracy_score(labels_test, pred_labels_test)))

    #==== Plot first mis-classified data ====
    # (Use the provided plot(x) function)
```

code/IEOR290_HW3_P3_skeleton.py