The Chinese University of Hong Kong, Shenzhen
School of Data Science

DDA5001 · Homework 3
Due: (23:59), November 9

Instructions:

- Homework problems must be carefully and clearly answered to receive full credit. Complete sentences that establish a clear logical progression are highly recommended.
- You must submit your assignment in Blackboard. Please combine the PDF and code files into a single .zip file for submission. The file name should be in the format **last name-first name-hw3**.
- The homework must be written in English.
- Late submission will not be graded.
- Each student **must not copy** homework solutions from another student or from any other source.

Problem 1 (40pts). Overfitting, validation, and regularization — A simple example
Suppose that we have the underlying model

$$y = x^2 + \varepsilon. \tag{1}$$

We collect $n = 10$ data points $\{(x_i, y_i)\}_{i=1}^n$; see the visualization in Figure 1. You can download the data from blackboard.

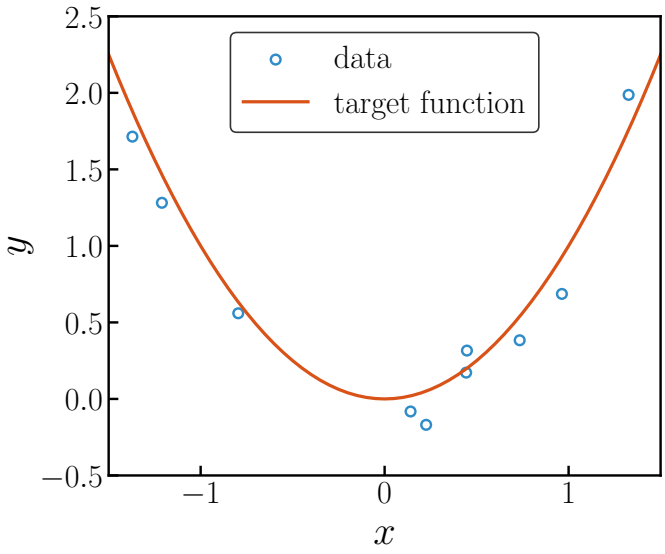

Figure 1: visualization of data

Now, suppose we are going to fit all the data using 8-th order polynomials:

$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots + \theta_8 x^8. \tag{2}$$

(a1) [5 points] Denote the $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_8) \in \mathbb{R}^9$ as the parameter. We have the following linear model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}.$$

Specify $\boldsymbol{y}$ and $\boldsymbol{X}$ using the training data $\{(x_i, y_i)\}_{i=1}^n$.

(a2) [6 points] Furthermore, we can formulate the following least squares

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^9}{\operatorname{argmin}} \ \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2. \tag{3}$$

Calculate $\widehat{\boldsymbol{\theta}}$ defined in (3) and plot the fitted curve in Figure 1 (limit x-axis from -1.5 to 1.5 and limit y-axis from -0.5 to 2.5). You can download the code used to generate Figure 1 from blackboard.

(a3) [3 points] Using the test data set, calculate the test error $\|\boldsymbol{X}_{\text{test}}\widehat{\boldsymbol{\theta}} - \boldsymbol{y}_{\text{test}}\|_2$ of $\widehat{\boldsymbol{\theta}}$ defined in (3).

(b1) [14 points] Since we know that the underlying model in (1) is quadratic, while the fitting model in (2) is polynomial of order 8, we must have overfitting, which you can see from question (a2) and (a3). One way to prevent overfitting is regularization. Instead of using (3), we formulate the following ℓ_2-regularized least squares

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathbb{R}^9}{\operatorname{argmin}} \ \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2. \tag{4}$$

However, one difficulty to implementing (4) is determining the regularization parameter λ. A too large λ leads to underfitting, while a too small λ (e.g. $\lambda = 0$) results in overfitting. Suppose we set the set of candidates of λ as

$$[10^{-5}\ 10^{-4}\ 10^{-3}\ 10^{-2}\ 10^{-1}\ 0.3\ 0.5\ 0.8\ 1\ 2\ 5\ 10\ 15\ 20\ 50\ 100]$$

using **5-fold cross-validation** to select the regularization parameter λ, and plot the validation error versus the value of λ, where error is the y-axis and λ is the x-axis (set the x-axis to log-scale).

(b2) [6 points] Based on the result in (b1), set $\lambda = 0.01$, 0.1, 0.8, and 5 in (4) and solve for the corresponding $\widehat{\boldsymbol{\theta}}$, respectively. Plot the fitted curve using the former four choices of λ in Figure 1, you have to draw four figures separately (limit x-axis from -1.5 to 1.5 and limit y-axis from -0.5 to 2.5).

(b3) [6 points] Using the test data set, calculate the test error $\|\boldsymbol{X}_{\text{test}}\widehat{\boldsymbol{\theta}} - \boldsymbol{y}_{\text{test}}\|_2$ of each of $\widehat{\boldsymbol{\theta}}$ obtained in (b2).

Problem 2 (35pts). Face Reconstruction by PCA

The principal component analysis (PCA) projects high dimensional data into lower dimension by optimizing the following problem

$$\min_{\boldsymbol{A}^\top \boldsymbol{A}=\mathbf{I},\boldsymbol{\Theta}} \|\boldsymbol{X}-\boldsymbol{A}\boldsymbol{\Theta}\|_F^2. \tag{5}$$

Here, $\boldsymbol{X}=[\boldsymbol{x}_1,\cdots,\boldsymbol{x}_n]\in\mathbb{R}^{d\times n}$ is the data matrix, $\boldsymbol{A}=[\boldsymbol{a}_1,\cdots,\boldsymbol{a}_k]\in\mathbb{R}^{d\times k}$ is the basis matrix, and $\boldsymbol{\Theta}=[\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_n]\in\mathbb{R}^{k\times n}$ is the principal components, i.e., the data matrix after dimension reduction. Consider the ORL dataset in our lecture notes, which consists of 400 images (40 people, 10 images for each person) of size 92×112. Hence, $d=10304$, $n=400$. In this case, the columns $\boldsymbol{a}_1,\cdots,\boldsymbol{a}_k$ can be interpreted as the prototype of faces, namely, the eigenfaces. The i-th reconstructed image $\boldsymbol{A}\boldsymbol{\theta}_i$ is the linear combination of these eigenfaces: $\sum_{j=1}^k \boldsymbol{a}_j\boldsymbol{\theta}_i[j]$.

Figure 2: Samples of the ORL dataset.

(1) [10 points] Show the top 40 eigenfaces (i.e. the eigenfaces with the top 40 largest eigenvalues).

(2) [15 points] The choice of k determines the reconstructed image's quality. Pick 5 images and plot a 5×6 figure, where the first column is the original image, and the last 5 columns are corresponding reconstructed images with $k\in\{20,40,100,200,300\}$, respectively. Briefly discuss what you discovered, e.g., the quality change of increasing k and why this happens.

(3) [10 points] Plot the signal-to-noise ratio (SNR) under different choices of k. The SNR is calculated by

$$\text{SNR}=10*\log_{10}\left(\frac{\|\boldsymbol{X}_{\text{recon}}\|_F^2}{\|\boldsymbol{X}_{\text{recon}}-\boldsymbol{X}\|_F^2}\right), \tag{6}$$

where $\boldsymbol{X}_{\text{recon}}\in\mathbb{R}^{d\times n}$ is the reconstructed image matrix.

Problem 3 (25pts). Support Vector Machine and Kernel Methods

Scikit-learn Package. We will use a famous machine learning toolbox for this problem called "scikit-learn", rather than implementing everything from scratch. The official document can be founded at `https://scikit-learn.org/stable/`. For the installation, please check `https://scikit-learn.org/stable/install.html`.

Data. The dataset we use is MNIST. To download it, run the following command

```
from sklearn.datasets import fetch_openml
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
```

Here, `X` is of dimension (num_samples, 784), where each sample is represented by a 784 dimension vector, corresponding to a 28 image. `y` is of dimension (num_samples,), representing the label. To plot j_{th} image, we can use the following code

```
import matplotlib.pyplot as plt
plt.title('The jth image is a {label}'.format(label=int(y[j])))
plt.imshow(X[j].reshape((28,28)), cmap='gray')
plt.show()
```

There are approximatly 7000 samples for each class in total. You should split the dataset, with 4000 samples in training set and remainer for test set.

Task. Our task is to use SVM to do the multi-class classification. We will use scikit-learn to do this. You may refer `https://scikit-learn.org/stable/modules/svm.html#svm-mathematical-formulation` for the formulation of scikit-learn's implementation. In their formulation, they have a parameter C, which performs exactly the similar role to our λ. You can simply regard C as $\frac{1}{\lambda}$. You should use *cross-validation* to select C. Specifically, you should split the training set into 3000 for training and 1000 for validation. You may use the following code for generating candidate C

```
C_grid = np.logspace(-3, 3, 10)
```

which returns 10 numbers spaced evenly with respect to interval $[-3, 3]$ on a log-scale.
An example code for training linear SVM with $C = 1$ is

```
from sklearn import svm
clf = svm.SVC(C=1.0, kernel='linear')
clf.fit(X_train, y_train)
```

The error for the resulting classifier can be calculated as

```
Pe = 1 - clf.score(X_test, y_test)
```

Note. To accelerate the experiment, you may use 10% of the data, i.e. for each class, select 300 for training and 100 for validation.

(a) [12 points] Train an SVM using the kernels $k(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u}^\top \boldsymbol{v} + 1)^p, p = 1, 2$, that is, the inhomogeneous linear and quadratic kernel. To do this, you can set the parameters `kernel='poly'`

and `degree=1` or `degree=2` in `svm.SVC` function, where `svm.SVC` is an embedded package in scikit-learn.

For each kernel, report the best value of C (draw a figure where y-axis is the validation error, while x-axis is the value of C), the test error corresponding to this best C (do not forget to retrain on the whole training dataset after selecting the best value of C). Hand in your code.

(b) [13 points] Repeat the above experiment using the radial basis function (RBF) kernel $k(\boldsymbol{u}, \boldsymbol{v}) = \exp(\|\boldsymbol{u} - \boldsymbol{v}\|_2^2)$ (by setting the parameters *kernel='rbf', gamma=gamma* in `svm.SVC` function. You will now need to determine the best value for both C and γ. Report the best value of C and γ (draw a figure where y-axis is the validation error, while x-axis is the value of C), the test error correspond to the best value of C and γ (do not forget to retrain on the whole training dataset after selecting the best value of C and γ). Hand in your code.