Homework 2: SVM and Linear Regression

EE 459/559 2020 Spring

Discussion is encouraged, but homework and codes shall be written on your own.

Due: 11:59pm, Mar 18th 2020

Late submissions will not accepted

1 Lineare Regression

1. (5 points) Consider two training samples:

$$(x^{(1)}, y^{(1)}) = (1, 1); \tag{1}$$

$$(x^{(2)}, y^{(2)}) = (2, 3). \tag{2}$$

Introduce a dummy feature "1" for each training sample. Then the two training samples are

$$x^{(1)} = [1, 1]^\top, y^{(1)} = 1; \tag{3}$$

$$x^{(2)} = [2, 1]^\top, y^{(2)} = 3. \tag{4}$$

Write out the first 5 iterations of linear regression using ℓ_2 loss. (see Lecture 7 for a similar example.)

Initialization: $\theta = [0, 0]^\top$, where the second entry of θ is the bias b. Step size $\alpha = 0.1$.

2. (5 points) *Include your codes with your submission.*

Implement (a) in Python. Hint: you can use the function polyfit in Python. Read the help document for the function polyfit before you start.

Use the original feature without the dummy feature "1".

3. (10 points) *Include your codes with your submission.*

Use the code in Figure 1 to generate the data. The data are generated according to the following function:

$$y = x^3 - x^2 + 1 + N, \tag{5}$$

```
[30] import numpy as np
     noise= np.random.normal(0,1,10)
     x=(np.arange(10)-5)/2
     y=np.power(x,3)-np.power(x,2)+1+noise

     plt.plot(x,y)

     [<matplotlib.lines.Line2D at 0x7fe1af6227b8>]
```

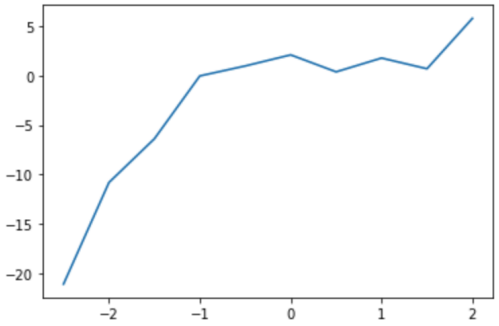

Figure 1: Code to generate training data.

where x is 1-dimensional feature, and N is a zero mean unit variance Gaussian random noise, i.e., $\mathcal{N}(0, 1)$.

For the following tasks, you can use the function polyfit in Python.

Fit the data using polynomial function with degree 1: $y = \theta_1 x + b$. This is equivalent to linear regression using the original feature in 1-dimensional space.

Fit the data using polynomial function with degree 2: $y = \theta_2 x^2 + \theta_1 x + b$. This is equivalent to linear regression using the feature mapping $\phi(x) = [x^2, x]^\top$ in 2-dimensional feature space.

Fit the data using polynomial function with degree 3: $y = \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + b$. This is equivalent to linear regression using the feature mapping $\phi(x) = [x^3, x^2, x]^\top$ in 3-dimensional feature space.

Fit the data using polynomial function with degree 4: $y = \theta_4 x^4 + \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + b$. This is equivalent to linear regression using the feature mapping $\phi(x) = [x^4, x^3, x^2, x]^\top$ in 4-dimensional feature space.

Fit the data using polynomial function with degree 5: $y = \theta_5 x^5 + \theta_4 x^4 + \theta_3 x^3 + \theta_2 x^2 + \theta_1 x + b$. This is equivalent to linear regression using the feature mapping $\phi(x) = [x^5, x^4, x^3, x^2, x]^\top$ in 5-dimensional feature space.

For the above five cases, plot y as a function of x, i.e., plot $y = \phi(x)^\top \theta$. Plot the underlying truth: $y = x^3 - x^2 + 1$ also in the same figure.

Discuss your understanding of overfitting and underfitting. Which one is underfitting/overfitting?

In practice, how will you use the training error and the test error to choose the degree in order for a good performance on unseen data?

2 SVM

Consider the following two training points:

$$x^{(1)} = 0, y^{(1)} = -1$$
$$x^{(2)} = \sqrt{2}, y^{(2)} = 1.$$

Consider a feature representation $\phi(x) = [1, \sqrt{2}x, x^2]^\top$ (which is equivalent to using a second order polynomial kernel). Recall the optimal margin classifier in the new 3d feature space:

$$\begin{aligned} &\min \|w\|_2^2 \\ &\text{s.t. } y^{(i)}(w^\top \phi(x^{(i)}) + b) \geq 1, i = 1, 2 \end{aligned} \tag{6}$$

Let w^* and b^* denote the solution of the above optimization problem.

1. (5 points)Write down a vector $\hat{w}$ that is parallel to the optimal vector w^*. Hint: for a decision boundary specified by w and b, w is perpendicular to the decision boundary.

2. (5 points)What is the value of the geometric margin that is achieved by this w^*? Hint 1: Recall the definition of geometric margin Hint 2: Think about the geometry of the points in feature space, and the vector between them.

3. (5 points)Solve for w^*, using the fact that the geometric margin is equal to $1/\|w^*\|_2$.

4. (5 points)Solve for b^* using your value for w^* and equation(6). Hint: the inequalities in (6) will be tight (i.e., becomes equalities since the solution to (6) will be on the boundary of the constraint set).

5. (5 points)Write down the form of the discriminant function $f(x) = {w^*}^\top \phi(x) + b^*$. Plot the 2 points $(x^{(1)}, y^{(1)})$ and $(x^{(2)}, y^{(2)})$ in the dataset, along with $f(x)$ in a 2d plot. You may generate this plot by hand, or using a computational tool like Python.

3 Classification on MNIST

In this task, we will perform a digit recognition task, where we are given an image of a handwritten digit, and the goal is to predict which number it represents. We consider a simplified version

```
[33] import tensorflow as tf
     import numpy as np
     import matplotlib.pyplot as plt

[34] #load data and preprocess data
     mnist = tf.keras.datasets.mnist
     #choose pictures of number 3 and number 5
     (train_images, train_labels),(test_images,test_labels)=mnist.load_data()#include all numbers from 0 to 9
     index_train=np.where((train_labels ==3) | (train_labels ==5))#index of numbers 3 and 5 in training data
     index_test=np.where((test_labels ==3) | (test_labels ==5))#index of numbers 3 and 5 in test data
     train_images_35=train_images[index_train]
     train_images_35=train_images_35.reshape((len(train_images_35), train_images_35[1].size))
     #label of number 3: -1; label of number 5: +1
     train_labels_35=train_labels[index_train].astype('int')
     test_images_35=test_images[index_test]
     test_images_35=test_images_35.reshape((len(test_images_35), train_images_35[1].size))
     test_labels_35=test_labels[index_test].astype('int')
     #change labels from '3' and '5' to '-1' and '+1'
     train_labels_35[np.where(train_labels_35==3)]= -1
     train_labels_35[np.where(train_labels_35==5)]=1
     test_labels_35[np.where(test_labels_35==3)]= -1
     test_labels_35[np.where(test_labels_35==5)]=1

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
     11493376/11490434 [==============================] - 0s 0us/step

[35] train_images_35=train_images_35/255 #normalizing feature vector
     test_images_35=test_images_35/255 #normalizing feature vector

     train_images_35_hw2=train_images_35[range(2000)] #choose a subset of the entire training dataset
     train_labels_35_hw2=train_labels_35[range(2000)] #choose a subset of the entire training dataset
```

Figure 2: Use a subset of the trainingdata.

of this problem, i.e., a binary classification problem between two numbers 3 and 5. This programming assignment shall be written using Python. For the programming questions, a brief explanation of your results together with your codes shall be submitted. You may use basic packages from Python like Numpy or CVXPY (refer to https://www.cvxpy.org/ for how to use this package for optimization problems), but you shall not use the packages which directly solve the problems, like sklearn.svm.

For the first four tasks below, we only use a subset of the training dataset in Homework 1. We take the first 2000 samples as our new training dataset. See code in Fig 2. To implement SVM, we do not introudce a dummy feature for each feature vector.

For the fifth task, we use the entire training dataset, i.e., use train_images_35 and train_labels_35 in the fifth task below.

1. (10 points) Recall the optimal margin classifier:

$$\min_{w,b} \|w\|_2^2$$
$$\text{s.t. } y^{(i)}(w^\top x^{(i)} + b) \geq 1, i = 1, 2, \ldots, m$$

Implement the optimal margin classifier in Python. Get the optimal w^* and b^*. Compute the training error on the training dataset using 0-1 loss. Evaluate the performance on the test dataset testx and testy using 0-1 loss.

2. (10 points) The dual form of the optimal margin classifier is

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$$
$$\text{s.t. } \alpha_i \geq 0, i = 1, 2, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \tag{7}$$

Implement the dual form of the optimal margin classifier. Get the optimal w^* and b^*. Compute the training error on the training dataset using 0-1 loss. Compare with the solution to the primal form of the optimal margin classifier.

Hint: Denote the solution to the above dual form as α^*. Then w^* and b^* can be computed as follows:

$$w^* = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)},$$
$$b^* = -\frac{\max_{i:y^{(i)}=-1} w^{*\top} x^{(i)} + \min_{i:y^{(i)}=1} w^{*\top} x^{(i)}}{2}.$$

3. (10 points) The dual form of optimal margin classifier is

$$\min_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y^{(i)} y^{(j)} \alpha_i \alpha_j < x^{(i)}, x^{(j)} >$$
$$\text{s. t. } 0 \leq \alpha_i, \, i = 1, \ldots, m$$
$$\sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \tag{8}$$

The kernel trick is to replace the inner product $< x^{(i)}, x^{(j)} >$ by $k(x^{(i)}, x^{(j)})$. Implement kernel SVM using Gaussian kernel, where

$$k(x^{(i)}, x^{(j)}) = \exp(-\|x^{(i)} - x^{(j)}\|_2^2/(2\sigma^2)).$$

In this task, set $\sigma = 1$. (Follows steps in page 18 and page 19 in the lecture notes SVM and kernel trick to compute the output of your classifier. Do not use the equations in problem 2 to compute w^* and b^*! That is only for linear classification in the original feature space, not for kernel SVM.) Compute the training error on the training dataset using 0-1 loss. Evaluate the performance of the kernel SVM on the test dataset testx and testy using 0-1 loss.

4. (5 points) Compare the performance of the optimal margin classifier, its dual form, kernel SVM, k-nearest neighbor classifier (k=3, 5) on the test dataset using 0-1 loss. Summarize your training errror and test error in a table format.

5. (20 points) Repeat tasks 1, 2, 3 and 4 using the **entire** dataset (i.e., the dataset as in homework 1) for **SVM with soft constraints**. Choose $C = 1, 3, 5$ respectively in your experiments.