MLISP: Machine Learning in Signal Processing

Problem set 2

Prof. V. I. Morgenshtern

Problem 1: Probabilistic interpretation of least-squares criterion

The goal of this exercise is to see that the least-squares criterion is equivalent to the maximum likelihood estimation in the model with Gaussian noise.

Suppose that the input data points $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$ are fixed. Assume that the target variables and the inputs are connected via the equation:

$$\mathrm{y}^{(i)} = \boldsymbol{\theta}^\mathsf{T} \mathbf{x}^{(i)} + \mathrm{n}^{(i)}, \quad i = 1, \dots, n, \tag{1}$$

where the noise variables $\mathrm{n}^{(i)}$ are independent over i and have the Gaussian PDF:

$$p_{\mathrm{n}}(n) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{n^2}{2\sigma^2}\right). \tag{2}$$

Our goal is to estimate $\boldsymbol{\theta}$.

1. Calculate the density function of each data point: $p_{\mathrm{y}^{(i)};\boldsymbol{\theta}}(y^{(i)})$.

2. Let $\mathbf{y} = [y^{(i)}, \dots, y^{(n)}]^\mathsf{T}$ be the vector of data points. The likelihood of the parameters $\boldsymbol{\theta}$ given the data is simply the density of the data, parameterized by $\boldsymbol{\theta}$

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{y}) = p_{\mathbf{y};\boldsymbol{\theta}}(\mathbf{y}), \tag{3}$$

   considered as a function of parameters, $\boldsymbol{\theta}$. Use the result from the previous item and the assumption that the noise variables $\mathrm{n}^{(i)}$ are independent over i to calculate this function.

3. The principle of maximum likelihood suggest to select the parameters $\boldsymbol{\theta}$ to maximize the likelihood, i.e.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathbf{y}). \tag{4}$$

   Maximum likelihood estimation is intuitively reasonable and has many precise statistical optimality properties that are beyond the scope of this class. Note that maximizing the likelihood is equivalent to maximizing the log-likelihood $\log \mathcal{L}(\boldsymbol{\theta})$, because the log function is monotonically increasing. Show that maximizing the log-likelihood is equivalent to the least-squares criterion:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} (\boldsymbol{\theta}^\mathsf{T} \mathbf{x}^{(i)} - y^{(i)})^2. \tag{5}$$

Problem 2: Nearest neighbors in high dimensions

1. Generate 100 points uniformly at random at the interval $[-0.5, 0.5]$. This is your dataset D_1. What is the distance between 0 and its nearest point in D_1? This quantity is denoted as $\rho(0, D_1)$.

2. Repeat the experiment above 50 times and record 50 different values for $\rho(0, D_1)$. Calculate the mean and the standard deviation of $\rho(0, D_1)$. These quantities are denoted mean$[\rho(0, D_1)]$ and std$[d(0, D_1)]$, respectively.

3. Repeat steps 1 and 2 starting with generating the 100 random points uniformly in the interval $[-0.5, 0.5]^p$ for the dimension $p = 2, 3, \ldots, 10$. For each p, you get mean$[\rho(0, D_p)]$ and std$[\rho(0, D_p)]$.

4. Plot mean$[\rho(0, D_p)]$ and std$[\rho(0, D_p)]$ as a function of p. What do you observe?

5. Repeat all previous steps for a ball of radius 0.5 in p dimensions.

Problem 3: Nearest neighbors vs number of DoF

The goal of this exercise is to use the binary classification problem with ORANGE and BLUE dots presented in class to better understand the key concepts of machine learning.

1. First generate an synthetic training dataset. Generate 5 points independently from the bivariate normal distribution $\mathcal{N}([1, 0]^\mathsf{T}, \mathbf{I})$. These are the centers of BLUE class, denote this set of 5 points as $\mathcal{B}$. Generate 5 points independently from the bivariate normal distribution $\mathcal{N}([0, 1]^\mathsf{T}, \mathbf{I})$. These are the centers of ORANGE class, denote this set of 5 points as $\mathcal{O}$. For the BLUE class, generate a 100 points as follows. Each point $k = 1, \ldots, 100$ selected the center $\mathbf{b}_k$ from $\mathcal{B}$ uniformly at random with probability 1/5; draw the point from the distribution $\mathcal{N}(\mathbf{b}_k, \mathbf{I}/7)$. Repeat this process for the ORANGE class, drawing the centers from the set $\mathcal{O}$. Display the BLUE and the ORANGE points using the scatter plot.

2. Generate the test set. To do so, keep the centers $\mathcal{B}$ and $\mathcal{O}$ as above: do not modify/regenerate them. Generate a fresh set of 100 points for the BLUE class and 100 points for the ORANGE class. Display the BLUE and the ORANGE points from the test set using the scatter plot. Compare to the training set above.

3. Fit the linear regression with thresholding to the training set as explained in class. Display the decision boundary on the scatter plot.

4. Calculate the fraction of points of the training set that are misclassified: devide the total number of misclassified points (BLUE or ORANGE) by the total number of points, 200. Do the same for the test set. Which fraction is larger?

5. Fit k-nearest neighbors classifier as explained in class for $k = 1, \ldots, 20$.

6. Calculate the fraction of points of the training set that are misclassified, plot this as a function of the number of degrees of freedom, $200/k$. Calculate the fraction of points of the test set that are misclassified, display on the same plot. Display on the same plot the misclassification fraction for the linear boundary on the training set and on the test set.

7. For each point $\mathbf{x} = [x_1, x_2]^\mathsf{T}$ on the plane, calculate the most likely class at that point

$$h(\mathbf{x}) = \underset{c \in \{\text{BLUE,ORANGE}\}}{\arg\max} P(\mathtt{y} = c | \mathtt{x} = \mathbf{x}) \tag{6}$$

and plot the optimal Bayesian decision boundary on the scatter plot from above.

Hint: Calculate $P(\mathtt{y} = c|\mathtt{x} = \mathbf{x})$ on a fine grid for $\mathtt{x}$. Use Bayes rule to express $P(\mathtt{y} = c|\mathtt{x} = \mathbf{x})$ via $P(\mathtt{x} = \mathbf{x}|\mathtt{y} = c)$.

Problem 4: Logistic regression [Ref: Stanford CS229 class]

The file `exams.txt` contains the historical data on student admissions into a university. For each training example, you have the applicant's scores on two exams and the admissions decision. Your task is to build a classification model that estimates an applicant's probability of admission based the scores from those two exams.

1. Split your dataset into two random parts: 2/3 of the rows in the training set and 1/3 of the rows in the data in the test set. Let n be the number of rows in the training set.
2. Use scatter plot to visualize the data. Observe that the data can possibly be separated by a linear decision boundary.
3. Implement the stochastic gradient descent algorithm for logistic regression as explained in Lecture 7. Use the training data to fit the model. *[Hint: Use batched gradeant descent for fine tuning the alpha values to get more consistent results. To calculate the convergence rate divide the new cost by the old cost after each iteration. The faster this rate converges to one, the better are the alpha values.]*
4. Calculate the accuracy of classification on the training and on the test set.

Problem 5: Logistic regression with features and regularization [Ref: Stanford CS229 class]

The file `chips.txt` contains test results for some microchips on a microchipe factory on two different tests. The first column contains the results of the first test, the second column contains the results of the second test, the third column contains '1' if the chip should be accepted and '0' if the chip should be rejected.

1. Split your dataset into two random parts: 2/3 of the rows in the training set and 1/3 of the rows in the data in the test set. Let n be the number of rows in the training set.
2. Use scatter plot to visualize the data. Observe that the data cannot be separated by a linear decision boundary. One way to separate the data might be to introduce new features. Let x_1 and x_2 represent the variables in column one and two in the original dataset. The new feature vector will be $\mathbf{f} = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, \ldots x_1x_2^5, x_2^6]^\mathsf{T}$, i.e. all polynomial terms in two variables up to sixth power. The original vector with two features has been transformed into a $p = 28$ dimensional vector. Extra new features make our model prone to overfitting.

To avoid overfitting, use regularized logistic regression to fit the data. Recall that the cost function of regularized logistic regression is given by

$$J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left[-y^{(i)}\log(h_{\boldsymbol{\theta}}(\mathbf{f}^{(i)})) - (1-y^{(i)})\log(1-h_{\boldsymbol{\theta}}(\mathbf{f}^{(i)}))\right] + \frac{\lambda}{2n}\sum_{j=1}^{p}\theta_j^2 \tag{7}$$

where

$$h_{\boldsymbol{\theta}}(\mathbf{f}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^{\top}\mathbf{f})}.$$

3. Calculate the gradient of $J(\boldsymbol{\theta})$. Hint: note that θ_0 is special, you do not need to add this term to regularization and also the gradient component corresponding to θ_0 is special.

4. Set $\lambda = 1$. Use gradient descent to fit the model by minimizing the cost. Report the cost on the training set and the cost on the test set.

5. Organize a search over the values of λ to optimize the performance on the test set. Report misclassification rate as a function of λ.

6. For the optimal choice of λ compute the classifiers predictions on a dense evenly spaced grid of points in $\mathbb{R}^2$ and plot the decision boundary on top of you initial scatter plot in item one.

Problem 6: Multi-class classification for MNIST

In this exercise you will create a simple MNIST digit classifier using logistic regression.

1. Download a subset of MNIST dataset using the following commands in Python:

```
from sklearn.datasets import load_digits
digits = load_digits()
```

Now that you have the dataset loaded you can use the commands below to see that there are 1797 images and 1797 labels in the dataset:

```
print('Image Data Shape' , digits.data.shape)
print('Label Data Shape', digits.target.shape)
```

Use the following code to visualize the data and understand its structure:

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(20,4))
for i, data in enumerate(zip(digits.data[0:5], digits.target[0:5])):
    image = data[0]
    label = data[1]
    plt.subplot(1, 5, i + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('Training: %i\n' % label, fontsize = 20)
```

2. Split the dataset into the training and the test set. Put 80 percent of the data into the training set.

3. This is a multi-class problem, because the labels are not binary $\{0, 1\}$, but take values in the set $\{0, 1, 2, \ldots, 9\}$ instead. Use the all-vs-one approach to multiclass classification and train 10 logistic regression classifiers for each digit separately. For example, to train the classifier for digit '3', relabel all digits in the training set that are not '3' into a single new 'not 3' label. Now you are back to familiar binary classification setup where you are learning a $\{3, \text{not } 3\}$ classifier. Do this for all digits, creating 10 different classifiers.

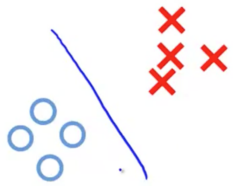
4. Use the 10 classifiers to make predictions for the multi-class problem. For each example, run all 10 classifiers. This gives 10 probabilities of the form

$$P(\mathbb{y}^{(i)} = 0 \mid \mathbb{x}^{(i)} = \mathbf{x}^{(i)})$$
$$P(\mathbb{y}^{(i)} = 1 \mid \mathbb{x}^{(i)} = \mathbf{x}^{(i)})$$
$$P(\mathbb{y}^{(i)} = 2 \mid \mathbb{x}^{(i)} = \mathbf{x}^{(i)})$$
$$\ldots$$

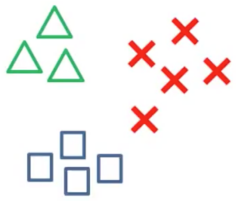
   Now simply predict the digit with the highest conditional probabilities.

5. Use the test set to create a confusion 10 by 10 matrix. The element M_{ij} of the matrix contains the percentage of examples with the correct label i that were classified as digit j by the algorithm. If your classifier works well, you should see large numbers on the diagonal of the matrix and small numbers everywhere else.

Problem 7: Exam practice (Multi-class classification) Logistic regression is an inherently binary linear classifier. It can be used to separate datasets like this:



Now suppose you need to solve a classification problem with three classes:



Explain in details how to use one-vs-all technique to adapt the logistic regression for the three class problem (at train time and at test time).