Homework 3: Logistic Regression, Model Selection and Clustering

EE 459/559 2020 Spring

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

Due: 11:59pm, April 12th 2020

Late submissions will not accepted

1 Model selection

	training loss	test loss
k-SVM σ =0.2		
k-SVM σ =0.5		
k-SVM σ =1		
k-SVM σ =3		
k-SVM σ =4		
k-SVM σ =5		
k-SVM σ =10		
Perceptrion		
SVM		
3-nearest neighbor		
5-nearest neighbor		

1. (20 points) **Tuning hyperparameter (k-SVM).**

   In Homework 2, we implemented kernel SVM using Gaussian kernel with $\sigma = 1$. Complete the following tasks to understand model selection. Use the first 2000 samples in the training dataset to train your classifiers.

(a) Run kernel SVM using Gaussian kernel for different values of $\sigma = 0.2, 0.5, 1, 3, 4, 5, 10$. Evaluate the performance of kernel SVM on both the training dataset and the test dataset using 0-1 loss. Summarize the training and test losses for different values of σ using the above table.

(b) Discuss your understanding of choosing the hyperparameter σ using the test dataset. Why shall we tune the hyperparameter, and how shall we do it?

(c) Run your Perceptron algorithm, k-nearest neighbor, SVM algorithms in Homework 1 using the first 2000 samples in the training dataset. Fill out the table.

(d) Discuss your understanding of how to choose among different algorithms, i.e., how to perform model selection?

2 Logistic regression

1. (30 points) **Logistic regression.**

   Include your codes with your submission.

   Use the training dataset from homework 2 (the first 2000 training samples), and implement the logistic regression algorithm.

   Introduce dummy feature "1" for each training and test sample.

   Change the labels from $\{+1, -1\}$ ***to*** $\{0, +1\}$***.***

   Initialization: $\theta = [0, \cdots, 0]^\top$: all entries of θ are initialized as zero. The dimension of θ shall be 784+1 (since we have one dummy feature "1"), and the last entry of θ is the bias term b. Step size $\alpha = 0.5$.

   (a) (10 points) Implement batch gradient descent.
   Convergence criterion: stop the loop when the gradient $\nabla \ell(\theta)$ is less than 10^{-2}.
   Note: when the gradient is small enough, the solution is usually very close to the local optimum.
   Evaluate the loss on training and test datasets using 0-1 loss (choose threshold $\gamma = 0.5$). Compare results to those in problem 1 model selection.

   Hint: follow Algorithm 1.

```
Result: $\theta$
initialization;
while $\|\nabla\ell(\theta)\|_2 > 10^{-2}$ do
    $\nabla\ell(\theta) \leftarrow \sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$;    %compute the gradient.
    $\theta \leftarrow \theta + \alpha\nabla\ell(\theta)$;    % update $\theta$.
end
```

Algorithm 1: Batch Gradient Descent for Logistic Regression.

(b) (10 points) Implement stochastic gradient descent. You can use the function shuffle to generate a randomly shuffled sequence.

For every 200 steps (10% of the training data), compute $\nabla\ell(\theta)$ and check whether it is less than 10^{-2}. If $\nabla\ell(\theta) < 10^{-2}$, then stop the while loop.

Evaluate the loss on training and test datasets using 0-1 loss(choose threshold $\gamma = 0.5$). Compare results to those in problem 1 model selection.

Hint: follow Algorithm 2.

```
Result: $\theta$
initialization;
flag=0;
Randomly shuffle the training dataset;
while flag=0 do
    for i=1: # of training samples do
        $\theta \leftarrow \theta + \alpha(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$;    % Here, $(x^{(i)}, y^{(i)})$ is the i-th sample in the shuffled
         training dataset.
        if every 200 steps then
            $\nabla\ell(\theta) \leftarrow \sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x^{(i)}$;    % compute the full gradient, and then check
             the stopping criterion;
            if $\|\nabla\ell(\theta)\|_2 < 10^{-2}$ then
                flag=1;
                break while;
            end
        end
    end
end
```

Algorithm 2: Stochastic Gradient Descent for Logistic Regression.

(c) (10 points) For the first two tasks, record how many times the whole training dataset is used before stopping.

Hint: For batch gradient descent, every time when $\nabla\ell(\theta)$ is computed, the whole dataset is used once. For stochastic gradient descent, for every For loop, the entire dataset is used once (in this task we ignore the computation used during checking the stopping criterion). Compare the times that the whole training dataset is used for two different approaches to reach the same level of accuracy (i.e., the full gradient $\nabla\ell(\theta) < 10^{-2}$), and answer the question that which one is more computationally efficient.

(d) Evaludate the two classifiers you obtained using gradient descent and stochastic gradient descent on the test dataset using different thresholds. Plot the ROC curves for the two classifiers. Dicuss which classifier is better.

(e) (optional, bonus 10 points) Repeat (a) (b) and (c) but change the accuracy from 10^{-2} to 10^{-5}. Discuss your findings.

Hint: stochastic gradient descent with constant step size will oscillate near the local optimum, and may never converge exactly. In this task, it may never converge to the one with accuracy 10^{-5}. This is usually not a problem in practice, since we may only need θ to be "close enough" to the local optimum.

3 K-fold cross validation

1. (20 points) Consider the following models. Consider the kernel SVM using Gaussian kernel with $\sigma = 0.2, 0.5, 1, 3, 4, 5, 10$. Implement the k-fold cross validation for $k = 10$ on the training dataset (the first 2000 samples). Report the estimated loss ε for each model. Choose the best σ, and check whether the best one here is the same as the best one for problem 1.

4 K-means clustering (50 points)

1. (20 points) Implement the k-means clustering algorithm by yourself. Use Euclidean (ℓ_2) distance.

   Dataset: hw3_data.mat.

   Use the following code to import data:

   from scipy.io import loadmat

   hw3data = loadmat('hw3_data.mat')

   X contains 600 samples. Each feature vector $\in \mathbb{R}^2$.

   Implement the k-means clustering algorithm. Use the function *scatter* to plot your clustering output for $k = 2, 3, 4$. Use different colors to represent different clusters. See an example picture with $k = 3$ as in Fig. 1. Try different initializations, and discuss your results.

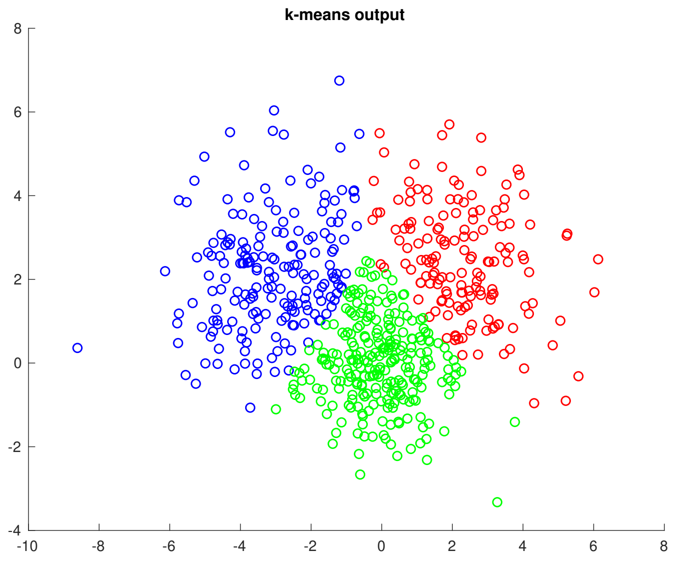

Figure 1: K-means output.

2. (10 points) Use the function *sklearn.cluster.KMeans* in python to solve the above question again. Use the function *scatter* to plot your clustering output for k=2, 3, 4. Use different colors to represent different clusters.