CS 541 Artificial Intelligence: Homework 3

Instructor: Jie Shen

Due: 10/28/2018, 11:00 pm EST

Instructions: This homework is designed to test your understanding of the course. Discussion is encouraged but you are expected to think about a problem for at least 24 hours before discussing with others. Please acknowledge those who helped you when you turn in your homework.

Submission: You need to typeset the report with either MS Word or Latex. For programming problems, make sure that

- do *not* upload the data in your experiments;
- keep your python code well-documented.

Notation: The solution $\boldsymbol{w} \in \mathbb{R}^d$ is always a column vector, and the feature vector $\boldsymbol{x} \in \mathbb{R}^d$ is always a row vector.

1 Stochastic Gradient Descent (30 pts)

The gradient descent (GD) algorithm enjoys optimal iteration complexity but suffers a high computational cost per iteration. Therefore, stochastic gradient descent (SGD) was proposed as a popular alternative for large-scale optimization problems. Consider the following general convex program:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w}). \tag{1}$$

If there is no structure in the objective function $F(\boldsymbol{w})$, we have to apply GD to solve it. However, in a broad range of machine learning problems it holds that

$$F(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{w}). \tag{2}$$

With the assumption on hand, SGD (or stochastic subgradient descent) starts with an arbitrary point $\boldsymbol{w}^0$, say $\boldsymbol{w}^0 = \mathbf{0}$, and in each iteration t, it picks an index $i_t \in \{1, 2, \dots, n\}$ uniformly random, and updates the solution as follows:

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \eta_t \nabla f_{i_t}(\boldsymbol{w}^{t-1}) \tag{3}$$

1. Show that in general a necessary condition for the convergence of SGD is

$$\lim_{t \to +\infty} \eta_t = 0.$$

2(a). Consider the linear regression problem where we are given $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$, and we are trying to find a parameter $\boldsymbol{w}$ that best fits the training data by solving (1) with

$$F(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i \cdot \boldsymbol{w})^2. \tag{4}$$

Derive the expression of $f_i(\boldsymbol{w})$ in this case.

2(b). In practice, we need to consider the noisy data and the closely related issue of over-fitting. Thus, in place of minimizing the least-squares loss (4), it is often useful to regularize it with 2-norm, known as *ridge regression*:

$$F(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i \cdot \boldsymbol{w})^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2, \ \lambda \geq 0. \tag{5}$$

Derive the expression of $f_i(\boldsymbol{w})$ for the above.

3. Now consider the classification problem where each $y_i \in \{+1, -1\}$. One may turn to the support vector machine (SVM) to obtain an accurate classifier. Consider the SVM with hinge loss formulation:

$$F(\boldsymbol{w}) = \frac{1}{n} \sum_{i=1}^{n} \max\{1 - y_i \boldsymbol{x}_i \cdot \boldsymbol{w}, 0\} + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2. \tag{6}$$

- Show that the objective function is λ-strongly convex, and thus the solution is unique.
- Derive the expression of $f_i(\boldsymbol{w})$.
- Write down the detailed update rule (3) for SVM. You can use the subgradient of hinge loss and the theoretical value of learning rate η_t without proof.

2 Support Vector Machine

MNIST[1] is a benchmark data set for classification. The full MNIST consists of 70,000 gray-scale images with size 28-by-28, each of which contains a digit from 0 to 9. In the experiments, we will use part of it for binary classification. We will also use the raw pixel as the feature vector $\boldsymbol{x}_i$. Therefore, the feature dimension $d = 28 \times 28 = 784$.

Download the "`mnist_01.mat`" in Canvas. This file is in Matlab format, and you will need to figure out a proper python API to load it. It stores four variables:

- $\boldsymbol{X}_{\text{train}}$: a $10,000 \times 784$ matrix, with each row being the raw pixel values (from 0 to 255) of an image, totally 10,000 training samples;
- $\textbf{label}_{\text{train}}$: a $10,000$-dimensional vector, with each row being either 0 or 1, indicating the labels for the images in $\boldsymbol{X}_{\text{train}}$;
- $\boldsymbol{X}_{\text{test}}$: a 1000×784 matrix, used for testing;
- $\textbf{label}_{\text{test}}$: a 1000-dimensional vector, containing the labels (0 or 1) for $\boldsymbol{X}_{\text{test}}$.

[1] `http://yann.lecun.com/exdb/mnist/`

2.1 Data Preparation (10 pts)

The first step in scientific discovery is to visualize the data set, to check if the data is correct, and to perform proper pre-processing.

- Randomly pick 10 images from the training set, and another 10 images from the testing set. Use python to display them. Note that the images are stored as a 784-dimensional row vector. You need to reshape it into 28×28.
- Print the corresponding image labels in $\textbf{label}_{\text{train}}$ and $\textbf{label}_{\text{test}}$ to see if the data has correct annotation.
- Normalize all the feature vectors (i.e. the rows of $\boldsymbol{X}_{\text{train}}$ and $\boldsymbol{X}_{\text{test}}$) such that they have unit ℓ_2-norm.
- The labels $\{y_1, \dots, y_n\}$ fed to SVM are required to be either $+1$ or -1. Therefore, you need to prepare the appropriate labels $\boldsymbol{y}_{\text{train}}$ and $\boldsymbol{y}_{\text{train}}$. For example, $\boldsymbol{y}_{\text{train}}$ could be a copy of $\textbf{label}_{\text{train}}$ but with 0 replaced by -1.

2.2 Evaluation Metric (5 pts)

For classification tasks the performance is usually evaluated by the 0/1 accuracy. That is, given a solution $\boldsymbol{w}$ and a data set $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, the accuracy is defined as:

$$\text{accuracy}(\boldsymbol{w}, \boldsymbol{X}, \boldsymbol{y}) = \frac{|\{i : \ y_i = \text{sign}\,(\boldsymbol{x}_i \cdot \boldsymbol{w})\}|}{n}$$

where $|\cdot|$ in the numerator denotes cardinality of a finite set, $\boldsymbol{X}$ is the matrix with each row being $\boldsymbol{x}_i$ and $\boldsymbol{y} = (y_1, \dots, y_n)$. Implement the above function. Note that you can make use of the matrix-vector multiplication for computational efficiency.

2.3 Convergence of SGD (15 pts)

Recall the SGD update rule you derived in Problem 1. Suppose that the hyper-parameter $\lambda = 1$.

- Use $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ to train a binary SVM classifier. You need to run SGD for 20,000 iterations, and plot the curves of "$F(\boldsymbol{w}^t)$ v.s. t" and "$F(\boldsymbol{w}^t)$ v.s. $1/t$". State when the curve gets stationary and what is the implication.
- For comparison, run SGD with constant step size, say $\eta_t = 0.01$ for all iterations. Plot the curve of "$F(\boldsymbol{w}^t)$ v.s. t" and summarize your observation.
- What happens if we choose $\eta_t = 1/t^2$?

2.4 Hyper-Parameter (40 pts)

From now on, let us stick with the theoretical value of η_t for SGD. Let us further fix the total iteration number as 10,000.

- The hyper-parameter λ in (6) plays an important role in controlling over-fitting. Run SGD with the following different choices of λ:

$$\lambda \in \{10^{-6},\ 10^{-3},\ 0.1,\ 0.5,\ 1,\ 2,\ 5,\ 10,\ 20,\ 50,\ 100,\ 500,\ 1000,\ 10000\}$$

  For each λ, we will have an iterate $\boldsymbol{w}^{10000}$ produced by SGD. Evaluate the accuracy of the obtained solution on both training and testing sets. Plot the curve of "training accuracy v.s. λ" and "testing accuracy v.s. λ" in a single figure. Also record the quantity $\left\|\boldsymbol{w}^{10000}\right\|_2 / d$ for each λ in a table. Summarize your findings.

Denote the λ and the solution $\boldsymbol{w}^{10000}$ that gives the best performance on the testing set by λ^* and $\boldsymbol{w}^*$.

- Due to the simplicity of SVM, it is possible to interpret the model $\boldsymbol{w}^*$ we learned. Visualize the model and summarize your observation. You may refer to Figure 7 (on page 24) of `http://jmlr.org/papers/volume18/16-299/16-299.pdf` for a way of visualization.
- In practice, the ℓ_2-regularizer usually encourages a dense solution. You may have observed it in the previous problem. As a remedy, one may apply hard thresholding for a sparse model that better explains the underlying data. Consider the sparsity

$$s \in \{10, 20, 50, 100, 200, 400\}.$$

  For each sparsity parameter s, denote

$$\boldsymbol{w}_s^* = \mathcal{H}_s(\boldsymbol{w}^*)$$

  where $\mathcal{H}_s(\boldsymbol{w}^*)$ is the hard thresholding operator that preserves the s largest elements (in magnitude) of $\boldsymbol{w}^*$ and sets the remaining to 0.

  Visualize the model $\boldsymbol{w}_s^*$ and compute its accuracy on the testing set. Summarize your observation.
- Explain why for a proper sparsity s, $\boldsymbol{w}_s^*$ performs as well as $\boldsymbol{w}^*$.

2.5 Noisy Labels (20 pts)

Noisy labels are ubiquitous in real-world applications, due to possible human mistakes and system errors. Thus during the design of an algorithm robustness has to be taken into account. It turns out that SVM is robust to noisy data with a proper choice of λ, and you need verify it in this section.

Let the noise level

$$\rho \in \{0, 0.01,\ 0.1,\ 0.2,\ 0.3,\ 0.5,\ 0.7\}.$$

- For each value of ρ, randomly select ρ fraction of the labels in $\boldsymbol{y}_{\text{train}}$ and flip them. In this way you will have a noisy data set. Train the SVM classifier with the noisy data by running SGD for 5000 iterations, with different λ:

$$\lambda \in \{10^{-6},\ 10^{-3},\ 0.1,\ 0.5,\ 1,\ 5,\ 10,\ 20,\ 50\}.$$

  Denote the λ value and the corresponding SGD solution $\boldsymbol{w}^{5000}$ that gives best testing accuracy by λ_ρ^* and $\boldsymbol{w}_\rho^*$ respectively. Fill in the following table and summarize your findings.

ρ	λ_ρ^*	Training Acc.	Testing Acc.	$\left\|\boldsymbol{w}_\rho^*\right\|_2/d$
0	-	-	-	-
0.01	-	-	-	-
0.1	-	-	-	-
0.2	-	-	-	-
0.3	-	-	-	-
0.5	-	-	-	-
0.7	-	-	-	-