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

DDA5001 · Homework 1

Due: 23:59, Sep 28.

Instructions:

- Homework problems must be clearly answered to receive full credit. Complete sentences that establish a clear logical progress are highly recommended.
- Submit your answer paper in Blackboard. Please upload a file or a zip file. The file name should be in the format **last name-first name-hw1**.
- 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. You are encouraged to discuss with others. However, you must write down the answer using your understanding (after discussion) and words.

Problem 1 (12 points). Concepts and Fundamental Knowledge

(a) (4 points) Concisely state the difference between supervised learning and unsupervised learning.

(b) (4 points) Which one of the following statements is true?

1) Regression is used to fit categorical labels.
2) Least squares must have infinitely many solutions if the number of data points is smaller than feature dimension.
3) The perceptron always converges to a unique linear classifier for a given training dataset.
4) Least squares is a maximum likelihood estimator.

(c) (4 points) Let the data matrix $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ be a full column rank matrix. Explain why $\boldsymbol{X}^\top \boldsymbol{X}$ is positive definite (PD). (Hint: Use the definition of linear independence)

Since a PD matrix must be invertible as well, this clarifies why we can take matrix inverse to obtain the unique solution of least squares in case I (where $\boldsymbol{X}$ has full column rank). From an optimization perspective, this PD property also reveals that the least squares in case I is strongly convex.

Solution.
(a) The supervised learning require the label of each data point to learn, while the unsupervised learning does not.

(b) 2). For 1), regression is for continuous labels. For 3), perceptron may have infinitely many solutions, any line that can classify the two classes. For 4), It is true under a Gaussian noise assumption.

(c) For any vector $\boldsymbol{v} \in \mathbb{R}^d \setminus \{0\}$, we have

$$\boldsymbol{v}^\top(\boldsymbol{X}^\top\boldsymbol{X})\boldsymbol{v} = (\boldsymbol{X}\boldsymbol{v})^\top(\boldsymbol{X}\boldsymbol{v}) = \|\boldsymbol{X}\boldsymbol{v}\|^2 \geq 0.$$

$\|\boldsymbol{X}\boldsymbol{v}\|^2 = 0$ if and only if $\boldsymbol{X}\boldsymbol{v} = 0$ (recall $\boldsymbol{v} \neq 0$). However, this is not possible since $\boldsymbol{X}$ has linear independent columns due to the fact that is has full column rank. Thus, $\boldsymbol{v}^\top(\boldsymbol{X}^\top\boldsymbol{X})\boldsymbol{v} > 0$ for any vector $\boldsymbol{v} \in \mathbb{R}^d \setminus \{0\}$, showing that $\boldsymbol{X}^\top\boldsymbol{X}$ is PD.

■

Problem 2 (17 points). Least Squares Without Full Column Rank

Consider the least squares problem

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}\|_2^2,$$

where $\boldsymbol{X}\in\mathbb{R}^{n\times d}$, $\boldsymbol{\theta}\in\mathbb{R}^d$, $\boldsymbol{y}\in\mathbb{R}^n$.

(a) (8 points) When $n<d$, the problem yields infinite many solutions. Assume that $\operatorname{rank}(\boldsymbol{X})=n$, derive the expression of the infinitely many solutions based on singular value decomposition (SVD). In addition, what is the optimal function value?

Note that the SVD is a notion from linear algebra, and the SVD of the matrix $\boldsymbol{X}$ can be written as

$$\boldsymbol{X}=\underbrace{\boldsymbol{V}}_{\in\mathbb{R}^{n\times n}}\underbrace{\begin{bmatrix}\boldsymbol{\Sigma}_1 & \mathbf{0}\end{bmatrix}}_{\in\mathbb{R}^{n\times d}}\underbrace{\begin{bmatrix}\boldsymbol{U}_1^\top\\ \boldsymbol{U}_2^\top\end{bmatrix}}_{\in\mathbb{R}^{d\times d}}=\boldsymbol{V}\boldsymbol{\Sigma}_1\boldsymbol{U}_1^\top.$$

where $\boldsymbol{V}\in\mathbb{R}^{n\times n}$ is an orthogonal matrix such that $\boldsymbol{V}^\top\boldsymbol{V}=\boldsymbol{V}\boldsymbol{V}^\top=\mathbf{I}$, $\boldsymbol{\Sigma}_1\in\mathbb{R}^{n\times n}$ is a diagonal matrix taking the form

$$\boldsymbol{\Sigma}_1(i,j)=\begin{cases}\sigma_i, & i=j\\ 0, & i\neq j\end{cases}\quad\text{with}\quad \sigma_1\geq\sigma_2\geq\cdots\geq\sigma_n>0,$$

$\boldsymbol{U}_1^\top\in\mathbb{R}^{n\times d}$ is an semi-orthogonal matrix satisfying $\boldsymbol{U}_1^\top\boldsymbol{U}_1=\mathbf{I}$ (but $\boldsymbol{U}_1\boldsymbol{U}_1^\top\neq\mathbf{I}$), and $\boldsymbol{U}_2^\top\in\mathbb{R}^{(d-n)\times d}$ is an semi-orthogonal matrix satisfying $\boldsymbol{U}_2^\top\boldsymbol{U}_2=\mathbf{I}$ (but $\boldsymbol{U}_2\boldsymbol{U}_2^\top\neq\mathbf{I}$).

(Hint: Let $\boldsymbol{A}:=\boldsymbol{V}\boldsymbol{\Sigma}_1$ which is square matrix of full rank, $\boldsymbol{z}=\boldsymbol{U}_1^\top\boldsymbol{\theta}$, solve $\min_{\boldsymbol{z}}\|\boldsymbol{A}\boldsymbol{z}-\boldsymbol{y}\|_2^2$ first, then solve $\boldsymbol{U}_1^\top\boldsymbol{\theta}=\boldsymbol{z}$ by considering that $\boldsymbol{U}_1^\top\boldsymbol{U}_2=\mathbf{0}$.)

(b) (5 points) Later in the overfitting section, we will study that we can also add a regularizer to solve the issue of infinitely many solutions, resulting in the following problem:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d}\|\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{\theta}\|_2^2,$$

where λ is a positive constant. Derive the unique solution of this problem.

Solution.

(a) By the close form solution of full column rank least square, we have

$$\begin{aligned}\boldsymbol{z}&=\left((\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{y}\right)\\ &=\left(\boldsymbol{\Sigma}_1^\top\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}_1\right)^{-1}\boldsymbol{\Sigma}_1^\top\boldsymbol{V}^\top\boldsymbol{y}\\ &=\boldsymbol{\Sigma}_1^{-1}\boldsymbol{V}^\top\boldsymbol{y}.\end{aligned}$$

Hence, the optimality condition of $\boldsymbol{\theta}$ is given by

$$\begin{aligned}
&\boldsymbol{U}_1^\top \boldsymbol{\theta} = \boldsymbol{\Sigma}_1^{-1} \boldsymbol{V}^\top \boldsymbol{y} \\
\iff\ &\boldsymbol{\theta} = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{V}^\top \boldsymbol{y} + \boldsymbol{U}_2 \boldsymbol{s},\ \forall \boldsymbol{s} \\
\iff\ &\boldsymbol{\theta} = \boldsymbol{U}_1 \boldsymbol{\Sigma}_1^{-1} \boldsymbol{V}^\top \boldsymbol{y} + \boldsymbol{h},\ \forall \boldsymbol{h} \in \mathrm{null}(\boldsymbol{U}_1).
\end{aligned}$$

The optimal function value is 0 by plugging $\boldsymbol{\theta}$ back to the objective function.

(b) The gradient of the objective is given by

$$2\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}) + 2\lambda \mathbf{I}\boldsymbol{\theta}.$$

Setting the gradient to $\mathbf{0}$ gives

$$(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I})\boldsymbol{\theta} = \boldsymbol{X}^\top \boldsymbol{y}.$$

Since for any vector $\boldsymbol{z} \in \mathbb{R}^d \setminus \mathbf{0}$, we have $\boldsymbol{z}^\top \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}\right) \boldsymbol{z} > 0$, the matrix $\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}$ is positive definite and thereby invertible, we have

$$\boldsymbol{\theta} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \mathbf{I}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}.$$

$\blacksquare$

Problem 3 (21 points). Robust Linear Regression

Suppose we have the generative linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}^\star + \boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon}$ is the error term and $\boldsymbol{\epsilon} \sim_{i.i.d.} \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ and $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ has full column rank. Under this Gaussian noise setup, the maximum likelihood estimator for $\boldsymbol{\theta}^\star$ is given by the least squares:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{LS} &= \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}} \ \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 \\ &= (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}.\end{aligned}$$

(a) (6 points) Suppose the error term, $\boldsymbol{\epsilon} = [\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n]$ follows the Laplace distribution, i.e., $\varepsilon_i \overset{i.i.d}{\sim} L(0, b), i = 1, 2, \cdots, n$ and the probability density function is $p(\varepsilon_i) = \frac{1}{2b}e^{-\frac{|\varepsilon_i - 0|}{b}}$ for some $b > 0$. Under the maximum likelihood estimation principle, derive the machine learning problem formulation for estimating $\boldsymbol{\theta}^\star$.

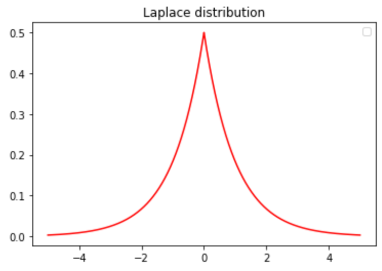


Figure 1: PDF of Laplace distribution

(b) (5 points) **Huber-smoothing.** ℓ_1-norm minimization

$$\hat{\boldsymbol{\theta}}_{L1} = \underset{\boldsymbol{\theta}\in\mathbb{R}^d}{\operatorname{argmin}} \|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|_1$$

is a popular approach for robust linear regression. However, it is nondifferentiable. We utilize smoothing technique for approximately solving the ℓ_1-norm minimization problem. Huber function is often the choice for smoothing ℓ_1 function. The definition and sketch map are shown as below. For some $\mu > 0$,

$$h_\mu(z) = \begin{cases} |z|, & |z| \geq \mu \\ \frac{z^2}{2\mu} + \frac{\mu}{2}, & |z| \leq \mu \end{cases}$$

Then,

$$H_\mu(\boldsymbol{z}) = \sum_{j=1}^{n} h_\mu(z_j).$$

By using Huber smoothing, the approximation of the above ℓ_1-norm-based robust linear regression problem can be smoothed as

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \ H_\mu(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}).$$

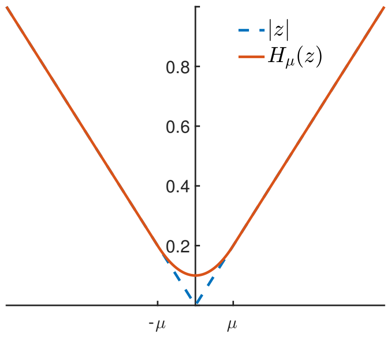


Figure 2: Huber smoothing

Define

$$\mathcal{L}(\boldsymbol{\theta}) = H_\mu(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}),$$

find the gradient $\nabla \mathcal{L}(\boldsymbol{\theta})$.

(c) (10 points) **Gradient descent for minimizing $\mathcal{L}(\boldsymbol{\theta})$.** The procedure of the gradient descent method is shown in the following algorithm pseudocode. (We will study the details of the gradient-based optimization algorithms later)

```
1. Input: Training data $\boldsymbol{X}$,$\boldsymbol{y}$ and initialization $\boldsymbol{\theta}_0$
   Huber smoothing parameter $\mu$,
   total iteration number $T$,
   learning rate $\alpha$.
2. for $k = 0, 1, \cdots, T$,do
3. $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}_k)$
4. end for
5. return $\boldsymbol{\theta}_T$
```

The data set is generated by the linear model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta}^\star + \boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_2,$$

where each entry of $\boldsymbol{\epsilon}_1 \in \mathbb{R}^n$ follows i.i.d. Gaussian distribution, while $\boldsymbol{\epsilon}_2 \in \mathbb{R}^n$ is a sparse vector that contains *outliers*. Given the observed traiing data $\boldsymbol{X}$ and $\boldsymbol{y}$,

(1) calculate the estimation $\hat{\boldsymbol{\theta}}_{LS}$ returned by least squares and compute $\left\|\hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}^\star\right\|_2$.

(2) suppose $n = 1000, d = 50$, use Python to implement the gradient descent method to minimize $\mathcal{L}(\boldsymbol{\theta})$ in part (b), the parameters are set as $\mu = 10^{-5}, \alpha = 0.001, T = 1000$, plot the error $\|\boldsymbol{\theta}_k - \boldsymbol{\theta}^\star\|_2$ as a function of iteration number. You can download the data $\{\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\theta}^\star\}$ on Blackboard.

Solution.

(a) Denote $D = (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)$, since $\boldsymbol{\epsilon} = \boldsymbol{y} - \boldsymbol{X\theta}$, the likelihood function is

$$\Pr(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{1}{2b} e^{\frac{-|\varepsilon_i|}{b}},$$

we have

$$\begin{aligned}
\log \Pr(D|\boldsymbol{\theta}) &= \sum_{i=1}^{n} (\log \frac{1}{2b} + \log e^{\frac{-|\varepsilon_i|}{b}}) \\
&= \sum_{i=1}^{n} \log \frac{1}{2b} - \frac{1}{b} \sum_{i=1}^{n} |\varepsilon_i| \\
&= \sum_{i=1}^{n} \log \frac{1}{2b} - \frac{1}{b} \sum_{i=1}^{n} |y_i - x_i^\top \boldsymbol{\theta}| \propto -\frac{1}{b} \sum_{i=1}^{n} |\boldsymbol{y} - \boldsymbol{X\theta}|_1
\end{aligned}$$

Hence, the learning problem is formulated as

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \|\boldsymbol{y} - \boldsymbol{X\theta}\|_1 .$$

(b) $\nabla \mathcal{L}(\boldsymbol{\theta}) = \boldsymbol{X}^\top \nabla H_\mu(\boldsymbol{X\theta} - \boldsymbol{y})$, $\nabla H_\mu(\boldsymbol{z}) = \sum \nabla h_\mu(z_i)$, where

$$\nabla h_\mu(z) \begin{cases} 1, & z \geq \mu \\ \frac{z}{\mu}, & |z| \leq \mu \\ -1, & z \leq \mu \end{cases}$$

(c) (1) $\hat{\boldsymbol{\theta}}_{LS} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$, $\left\| \hat{\boldsymbol{\theta}}_{LS} - \boldsymbol{\theta}^\star \right\|_2 = 59.5$; (2) See the file: “p3.py”.

■

Problem 4 (24 points). Convergence of The Perceptron for Linearly Separable Data

In lecture, we studied that one iteration of perceptron algorithm will move forward to the currently wrongly classified data. Though such an observation does provide some intuition that perceptron will eventually converge to a linear classifier that correctly separate all the (linearly separable) training data, it is not rigorously justified.

Let us first review the algorithm. The perceptron applies to the linear model

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}.$$

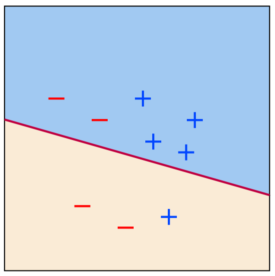
At k-the iteration, the algorithm has parameter $\boldsymbol{\theta}_{k-1}$. Based on $\boldsymbol{\theta}_{k-1}$, the algorithm first pick a wrongly classified data point $(\boldsymbol{x}_{k-1}, y_{k-1})$ from all the data points $((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y_i), \ldots, (\boldsymbol{x}_n, y_n))$, i.e.,

$$\operatorname{sign}\left(\boldsymbol{\theta}_{k-1}^\top \boldsymbol{x}_{k-1}\right) \neq y_{k-1}.$$

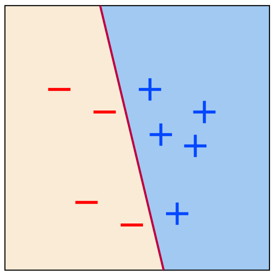
Then, perceptron updates as

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + y_{k-1}\boldsymbol{x}_{k-1} \quad \forall k = 1, 2, \ldots \tag{1}$$

In this question, you will be guided to show that perceptron will eventually stop, i.e., there exists a $\bar{k}$ such that no wrongly classified data exists after at most $\bar{k}$ number of iterations, and then the perceptron learning algorithm will automatically terminate; see Fig. 3 for an illustration.



(a) Perceptron at the k-th iteration, not all data are correctly classified



(b) Learned perceptron model after at most $\bar{k}$ iterations, all data are correctly classified

Figure 3: Illustration of the convergence progress of the perceptron learning algorithm.

Let $\boldsymbol{\theta}^*$ be corresponding to a classifier that correctly separates all the data points like the one in Fig. 3b. For simplicity, we assume that the initial point of the perceptron algorithm is $\boldsymbol{\theta}_0 = \mathbf{0}$.

The full proof procedure is divided into 5 steps.

(a) (3 points) Let $\rho = \min_{1\le i\le n} y_i(\boldsymbol{\theta}^{*\top}\boldsymbol{x}_i)$. Show that $\rho > 0$.

Solution: This is trivially true since $\boldsymbol{\theta}^*$ correctly classifies all the data points, and thus $\boldsymbol{\theta}^{*\top}\boldsymbol{x}_i$ has the same sign as y_i.

(b) (6 points) Show that $\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^* \geq \boldsymbol{\theta}_{k-1}^\top \boldsymbol{\theta}^* + \rho$, and hence conclude that $\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^* \geq k\rho$.

(Hint: Use the update (1).)

Solution: From the update (1), we have $\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^* = (\boldsymbol{\theta}_{k-1} + y_{k-1}\boldsymbol{x}_{k-1})^\top \boldsymbol{\theta}^* = \boldsymbol{\theta}_{k-1}^\top \boldsymbol{\theta}^* + y_{k-1}\boldsymbol{x}_{k-1}^\top \boldsymbol{\theta}^* \geq \boldsymbol{\theta}_{k-1}^\top \boldsymbol{\theta}^* + \rho$, where the last inequality is from the definition of ρ. Now, we just unroll this inequality from k-th iteration to 0-th iteration and use $\boldsymbol{\theta}_0 = \mathbf{0}$, we have $\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^* \geq k\rho$.

(c) (6 points) Show that $\|\boldsymbol{\theta}_k\|^2 \leq \|\boldsymbol{\theta}_{k-1}\|^2 + \|\boldsymbol{x}_{k-1}\|^2$.

(Hint: Use the fact that $\boldsymbol{x}_{k-1}$ is misclassified by $\boldsymbol{\theta}_{k-1}$.)

Solution: From the update (1), we have $\|\boldsymbol{\theta}_k\|^2 = \|\boldsymbol{\theta}_{k-1} + y_{k-1}\boldsymbol{x}_{k-1}\|^2 = \|\boldsymbol{\theta}_{k-1}\|^2 + 2y_{k-1}\boldsymbol{\theta}_{k-1}^\top \boldsymbol{x}_{k-1} + \|\boldsymbol{x}_{k-1}\|^2$, where we have used $y_{k-1} \in \{+1, -1\}$. Since $\boldsymbol{x}_{k-1}$ is misclassified by $\boldsymbol{\theta}_{k-1}$, we must have $y_{k-1}\boldsymbol{\theta}_{k-1}^\top \boldsymbol{x}_{k-1} < 0$, and then the desired result follows.

(d) (3 points) Show that $\|\boldsymbol{\theta}_k\|^2 \leq kR^2$, where $R = \max_{1 \leq i \leq n} \|\boldsymbol{x}_i\|$.

Solution: By unrolling the inequality in the last step from k-th iteration to 0-th iteration and $\boldsymbol{\theta}_0 = \mathbf{0}$, we obtain the desired result.

(e) (6 points) Using steps 2 and 4, show that

$$\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|} \geq \sqrt{k}\frac{\rho}{R}$$

and hence show that the perceptron learning algorithm must stop within at most

$$\bar{k} \leq \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\rho^2}$$

number of iterations.

(Hint: Use the fact $\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|\|\boldsymbol{\theta}^*\|} \leq 1$. Why?)

In practice, perceptron will converge often faster than the above bound. However, since we do not know ρ and $\boldsymbol{\theta}^*$ in advance, we actually cannot determine the number of iterations to convergence, which does pose a problem if we have non-linearly separable data.

Solution: To lower bound $\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|}$, we can lower bound $\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*$ using step 2 and upper bound $\|\boldsymbol{\theta}_k\|$ using step 4, then we get $\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|} \geq \sqrt{k}\frac{\rho}{R}$.

Rearranging terms gives

$$k \leq \frac{(\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*)^2 R^2}{\|\boldsymbol{\theta}_k\|^2 \rho^2} = \frac{(\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*)^2 R^2 \|\boldsymbol{\theta}^*\|^2}{\|\boldsymbol{\theta}_k\|^2 \|\boldsymbol{\theta}^*\|^2 \rho^2} \leq \frac{R^2\|\boldsymbol{\theta}^*\|^2}{\rho^2}.$$

Here, we have used $\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|\|\boldsymbol{\theta}^*\|} \leq 1$ in the last inequality, which is due to the fact that the quantity $\frac{\boldsymbol{\theta}_k^\top \boldsymbol{\theta}^*}{\|\boldsymbol{\theta}_k\|\|\boldsymbol{\theta}^*\|}$ equals to the cosine of the angle between $\boldsymbol{\theta}_k$ and $\boldsymbol{\theta}^*$, which is always less than or equal to 1.

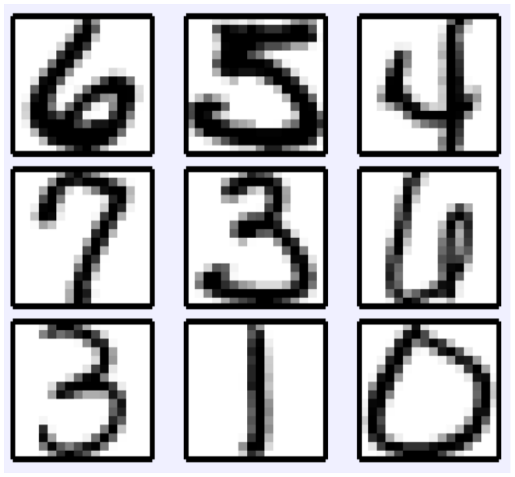
Problem 5 (26 points). Pocket Algorithm for Non-Seperable data

In Problem 4, we have shown that the perceptron algorithm will converge for linearly seperable data. However, for non-linearly separable data, the algorithm will never terminate. In fact, the behavior becomes quite unstable and can jump from a good perceptron to a bad one with only one update (as we will see in this problem). Hence, the quality of $\mathrm{Er}_{\mathrm{in}}$ cannot be guaranteed.

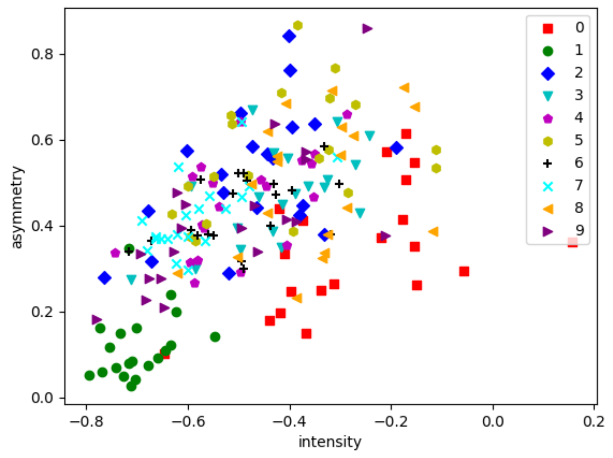
One simple approach to deal with the unstability is to use the *pocket algorithm*. The algorithm preserves the parameter $\boldsymbol{\theta}$ with the best in-sample error during the update, which ensures monotonically decreasing in-sample error. The algorithm is summarized below.

```
    Pocket Algorithm
1.  set the pocket weight parameter $\hat{\boldsymbol{\theta}}$ to be $\boldsymbol{\theta}_0$
2.  for k = $0, \cdots, T$,  do
3.    pick a wrongly classified sample from dataset as $(\boldsymbol{x}_k, y_k)$
4.    update as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + y_k \boldsymbol{x}_k$
5.    if  $\mathrm{Er}_{\mathrm{in}}(\boldsymbol{\theta}_{k+1}) < \mathrm{Er}_{\mathrm{in}}(\hat{\boldsymbol{\theta}})$:
6.      $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{k+1}$
7.  end for
8.  return  $\hat{\boldsymbol{\theta}}$
```

In this problem, we will implement the performance of perceptron and pocket algorithms based on the US Post Office Zip Code Data, which is a dataset of 16 $\times$ 16 grayscale images for digits (0-9). The training dataset has 7291 samples and the test dataset has 2007 samples.



(a) Sample digit images.



(b) Feature plot of different classes.

Figure 4: Illustration of sample digits and corresponding feature distribution.

We may extract heuristic features to help us do the classification. Let us focus on the binary classification between digits ”1” and ”5”. In this case, the *intensity* may be a good feature: the digit ”5” usually occupies larger area than digit ”1”. The *asymmetry* is another useful feature, as

"1" tends to be more symmetric than "5". We can define the asymmetry feature as the average absolute difference between an image and its flipped version.

1. (10 points) Consider the binary classification problem for digits "1" and "5". Implement the perceptron algorithm and pocket algorithm. The suggested hyperparameters are: threshold = 1e-4, total iteration = 2000, initial parameter $\boldsymbol{\theta}_0 = \mathbf{0}$. (You may implement the algorithms based on the provided source code, where you should finish the part marked as `#TODO`)

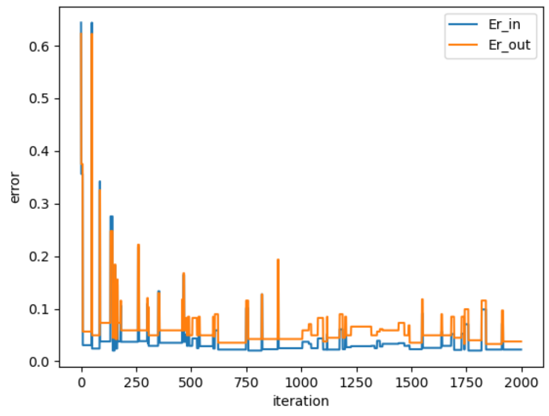
2. (8 points) Plot the in-sample error and the out-of-sample error of these two algorithms versus the number of iterations (for this problem, we consider the training error as the in-sample error, and the test error as out-of-sample error). Note that for the pocket algorithm, the errors are calculated with respect to $\hat{\boldsymbol{\theta}}$ (instead of $\boldsymbol{\theta}_k$). Briefly discuss the results.

3. (8 points) Plot 500 data points from class "1" and "5", respectively (similar to the Fig. 4b but with only 2 classes). On this figure, plot the final classification boundary of the two algorithms.

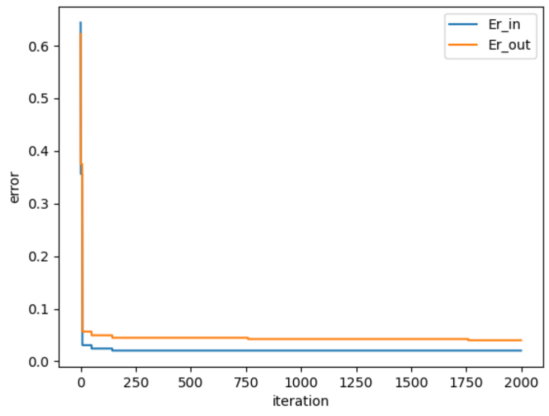
Solutions.

1. See attachment.

2. The errors are summarized below. We can see that the perceptron's training curve is unstable compared with the pocket algorithm.

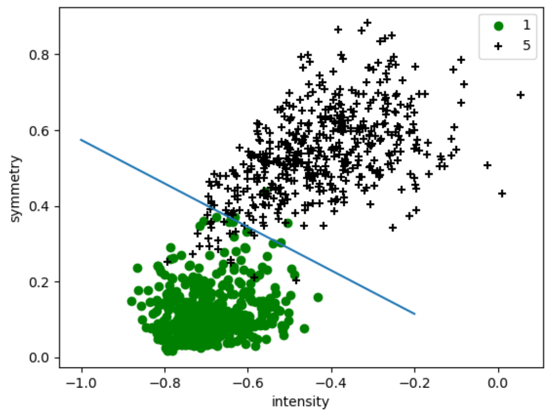


(a) Perceptron error.

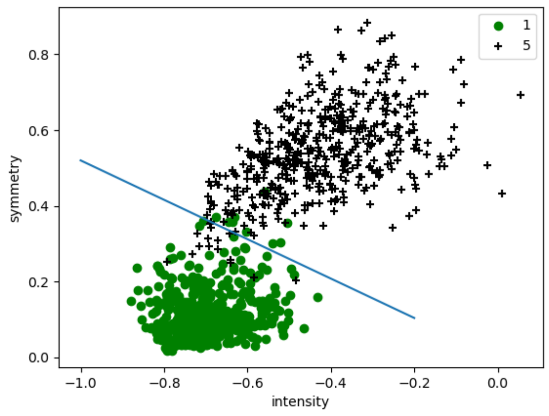


(b) Pocket error.

3. The decision boundary for two algorithms are shown below. We can see that the pocket algorithm enjoys relatively better classification result on boundary cases.



(a) Perceptron decision boundary.



(b) Pocket decision boundary.