Homework 3 Written Part Report
FAI 2024

B11705048, 林杰

Problem 1 (10 points)

Consider a binary classification data set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ with $\mathbf{x}_n \in \mathbb{R}^d$ and $y_n \in \{-1, +1\}$. For any weight vector $\mathbf{w}$ within a linear model, define an error function

$$\text{err}(\mathbf{w}^T\mathbf{x}, y) = \big(\max(1 - y\mathbf{w}^T\mathbf{x}, 0)\big)^2.$$

That is,

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\big(\max(1 - y_n\mathbf{w}^T\mathbf{x}_n, 0)\big)^2$$

Running gradient descent to optimize $E_{\text{in}}(\mathbf{w})$ requires calculating its gradient direction $\nabla E_{\text{in}}(\mathbf{w})$ (and then move opposite to that direction). What is $\nabla E_{\text{in}}(\mathbf{w})$?

The error function err *is called the squared hinge error and is a core component in the so-called l_2-loss SVM.*

$$\text{err}(\vec{w}^T\vec{x}, y) = (\max(1 - y\vec{w}^T\vec{x}, 0))^2$$

$$E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N}(\max(1 - y_n\vec{w}^T\vec{x}_n, 0))^2$$

$$= \frac{1}{N}\sum_{n=1}^{N}\begin{cases}(1 - y_n\vec{w}^T\vec{x}_n)^2, & \text{if } y_n\vec{w}^T\vec{x}_n < 1\\ 0, & \text{if } y_n\vec{w}^T\vec{x}_n \geq 1\end{cases}$$

$$\nabla E_{in}(\vec{w}) = \frac{1}{N}\sum_{n=1}^{N}\begin{cases}2\cdot(1 - y_n\vec{w}^T\vec{x}_n)\cdot -(y_n\vec{x}_n), & \text{if } y_n\vec{w}^T\vec{x}_n < 1\\ 0, & \text{if } y_n\vec{w}^T\vec{x}_n \geq 1\end{cases}$$

Problem 2 (15 points)

Consider a process that generates d-dimensional vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ independently from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{u}, \mathrm{I})$, where $\mathbf{u} \in \mathbb{R}^d$ is an unknown parameter vector and $\mathrm{I} \in \mathbb{R}^{d \times d}$ is an identity matrix. The maximum likelihood estimate of $\mathbf{u}$ is

$$\mathbf{u}^* = \arg\max_{\mathbf{u} \in \mathbb{R}^d} \prod_{n=1}^{N} p_{\mathbf{u}}(\mathbf{x}_n),$$

where $p_{\mathbf{u}}$ is the probability density function of $\mathcal{N}(\mathbf{u}, \mathrm{I})$. Prove that

$$\mathbf{u}^* = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n.$$

The same derivation can be used to obtain the cross-entropy error function of logistic regression.

$N(\vec{x}; \vec{u}, I) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{u})^T(\vec{x}-\vec{u})\right)$ (since I is the identity matrix and $|I| = 1$)

Take log for a single $\vec{x}$:

$\rightarrow \log N(\vec{x}; \vec{u}, I) = -\frac{d}{2}\log 2\pi - \frac{1}{2}(\vec{x}-\vec{u})^T(\vec{x}-\vec{u})$

Terms independent from the parameters can be ignored:

$\rightarrow \log N(\vec{x}; \vec{u}, I) = -\frac{1}{2}(\vec{x}-\vec{u})^T(\vec{x}-\vec{u}) + C$, C: constant

Use X to denote the set of i.i.d. vectors from $\vec{x}_1$ to $\vec{x}_n$, we have:

$\rightarrow \log N(X; \vec{u}, I) = -\frac{1}{2}\sum_{i=1}^{n}(\vec{x}_i-\vec{u})^T(\vec{x}_i-\vec{u}) + C$, C: constant

To find the maximum likelihood estimate, we set the derivative w.r.t. $\vec{u}$ to zero:

$\rightarrow \frac{\partial}{\partial \vec{u}} \log N(X; \vec{u}, I) \overset{\text{"chain rule"}}{=} \sum_{i=1}^{n}(\vec{x}_i - \vec{u}) = 0$

We can derive that $\vec{u}^* = \frac{\sum_{i=1}^{n}\vec{x}_i}{n}$ #

Reference: https://home.ttic.edu/~shubhendu/Slides/Estimation.pdf , P56 ~ 60

Problem 3 (15 points)

A classic binary classification data set that cannot be separated by any line is called the XOR data set, with

$\mathbf{x} = [x_1, x_2]$	y
$\mathbf{x}_1 = [+1, +1]$	$y_1 = -1$
$\mathbf{x}_2 = [-1, +1]$	$y_2 = +1$
$\mathbf{x}_3 = [-1, -1]$	$y_3 = -1$
$\mathbf{x}_4 = [+1, -1]$	$y_4 = +1$

You can see why it is called XOR by interpreting $+1$ as a boolean value of TRUE and -1 as FALSE. Consider a second-order feature transform $\Phi_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$ that converts the data set to

$\mathbf{z} = \Phi_2(\mathbf{x})$	y
$\mathbf{z}_1 = \Phi_2(\mathbf{x}_1)$	$y_1 = -1$
$\mathbf{z}_2 = \Phi_2(\mathbf{x}_2)$	$y_2 = +1$
$\mathbf{z}_3 = \Phi_2(\mathbf{x}_3)$	$y_3 = -1$
$\mathbf{z}_4 = \Phi_2(\mathbf{x}_4)$	$y_4 = +1$

Show a perceptron $\tilde{\mathbf{w}}$ in the $\mathcal{Z}$-space that separates the data. That is,

$$y_n = \text{sign}(\tilde{\mathbf{w}}^T \mathbf{z}_n) \text{ for } n = 1, 2, 3, 4.$$

Then, plot the classification boundary

$$\tilde{\mathbf{w}}^T \Phi_2(\mathbf{x}) = 0$$

in the $\mathcal{X}$-space. Your boundary should look like a quadratic curve that classifies $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ perfectly.

$\vec{z}_1 = [1, 1, 1, 1, 1, 1]$

$\vec{z}_2 = [1, -1, 1, 1, -1, 1]$

$\vec{z}_3 = [1, -1, -1, 1, 1, 1]$

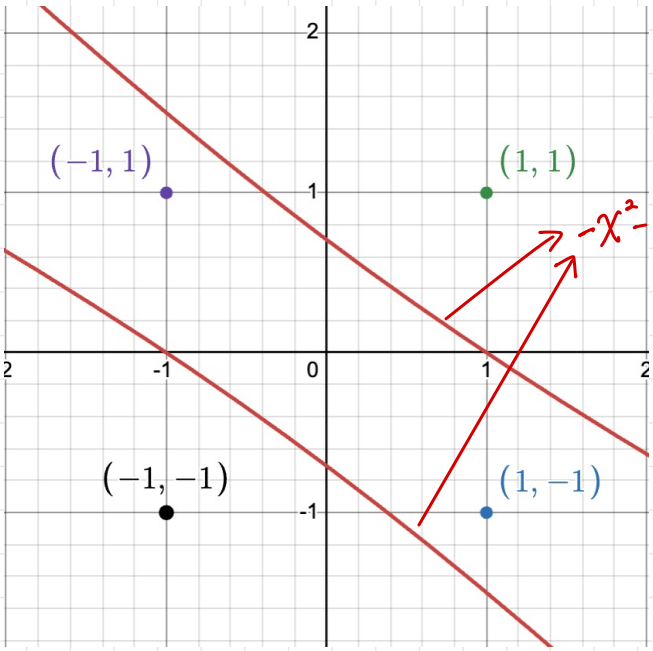
$\vec{z}_4 = [1, 1, -1, 1, -1, 1]$

Take the perceptron $\vec{w}$ as $[1, 0, 0, -1, -3, -2]$

$\vec{w}^T\vec{z}_1 = -5$ $\vec{w}^T\vec{z}_2 = 1$ $\vec{w}^T\vec{z}_3 = -5$ $\vec{w}^T\vec{z}_4 = 1$

$\text{sign}(\vec{w}^T\vec{z}_1) = -1$, $\text{sign}(\vec{w}^T\vec{z}_2) = +1$, $\text{sign}(\vec{w}^T\vec{z}_3) = -1$, $\text{sign}(\vec{w}^T\vec{z}_4) = +1$

Plot $\vec{w}^T\Phi_2(\vec{x}) = 0$:



$-x^2 - 2y^2 - 3xy + 1 = 0$

Problem 4 (15 points)

Consider building binary classification with AdaBoost algorithm, a weak classifier $g_t(\mathbf{x})$ is trained on a data set $\{\mathbf{x_n}, y_n\}_{n=1}^N$ with weights $\{w_n^t\}_{n=1}^N$ at the time step t. The error rate is defined as

$$\epsilon_t = \frac{\sum_{n=1}^N w_n^t \cdot \delta(g_t(\mathbf{x_n}), y_n)}{\sum_{n=1}^N w_n^t},$$

對正確給 1 分，錯誤給 0 分

where $\delta(g_t(\mathbf{x_n}), y_n) = 1$ if $g_t(\mathbf{x_n}) \neq y_n$ and $\delta(g_t(\mathbf{x_n}), y_n) = 0$ otherwise. For the next time step, the data set is reweighed to emphasize on misclassified samples through the following rules

$$w_n^{t+1} = \begin{cases} w_n^t \cdot d_t & \text{if } g_t(\mathbf{x_n}) \neq y_n \\ w_n^t / d_t & \text{if } g_t(\mathbf{x_n}) = y_n \end{cases}.$$

Show that $d_t = \sqrt{(1-\epsilon_t)/\epsilon_t}$ can degrade the previous classifier $g_t(\mathbf{x})$ and make its error rate become 0.5 with new weights $\{w_n^{t+1}\}_{n=1}^N$:

$$\frac{\sum_{n=1}^N w_n^{t+1} \delta(g_t(\mathbf{x_n}), y_n)}{\sum_{n=1}^N w_n^{t+1}} = 0.5.$$

Let W_I^t denotes the summation of weights where $g_t(\vec{x}_n) \neq y_n$ (At time t): $W_I^t = \sum_{n=1}^N W_n^t \cdot \delta(g_t(\vec{x}_n), y_n)$

Let W_C^t denots the summation of weights where $g_t(\vec{x}_n) = y_n$ (At time t): $W_C^t = (\sum_{n=1}^N W_n^t) - W_I^t$

→ We can know that $\epsilon_t = \frac{W_I^t}{W_I^t + W_C^t}$, $d_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \sqrt{\frac{W_C^t}{W_I^t}}$

For the new weights after adjusting, $W_I^{t+1} = d_t \cdot W_I^t = \sqrt{\frac{W_C^t}{W_I^t}} \cdot W_I^t$

$$W_C^{t+1} = \frac{1}{d_t} \cdot W_C^t = \sqrt{\frac{W_I^t}{W_C^t}} \cdot W_C^t$$

The new error rate $\frac{\sum_{n=1}^N W_n^{t+1} \delta(g_t(\vec{x}_n), y_n)}{\sum_{n=1}^N W_n^{t+1}} = \frac{W_I^{t+1}}{W_I^{t+1} + W_C^{t+1}} = \frac{d_t W_I^t}{d_t W_I^t + \frac{1}{d_t} W_C^t}$

$$= \frac{(d_t)^2 W_I^t}{(d_t)^2 W_I^t + W_C^t} = \frac{\left(\frac{W_C^t}{W_I^t}\right) \cdot W_I^t}{\frac{W_C^t}{W_I^t} \cdot W_I^t + W_C^t} = \frac{W_C^t}{2W_C^t} = \frac{1}{2} \#$$

Homework 3 Programming Part Report

Foundations of Artificial Intelligence (CSIE3005)

Department of Information Management, NTU - B11705048 林杰

(A) Performance Comparison: Different Models

(Parameters for this subproblem: learning_rate = 0.01, iterations = 1000, max_depth = 5, n_estimators = 100.)

Classification Task:

For the classification task, the evaluation metric is the accuracy.

$$accuracy = \frac{correct\ pedictions}{total\ predictions}$$

Logistic Regression Accuracy: 66.67%

Decision Tree Classifier Accuracy: 88.89%

Random Forest Classifier Accuracy: 95.56%

Analysis:

Decision trees can model complex, non-linear relationships between the features and the target variable, while logistic regression assumes a linear relationship between the input features. Therefore, decision tree classifier can reach higher accuracy than simple logistic regression.

Random forest classifier randomly selects samples from the data and train separate decision trees based on each sample. Then, the predictions from all the individual trees are combined together using a voting mechanism for the final classification result. Therefore, the decision tree classifier can reach the highest accuracy among the three models.

Regression Task:

For the classification task, the evaluation matric is the mean square error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)$$

Linear Regression MSE: 43.41

Decision Tree Regressor MSE: 28.34

Ransom Forest Regressor MSE: 22.96

Analysis:

Similar to the previous analysis, decision trees can model more complex and

non-linear relationships between the features. Therefore, the accuracy rate is higher than the simple linear regression model.

Random forest classifier randomly selects samples from the data and train separate decision trees based on each sample. For regression problems, the average of the predictions from all trees is used. Therefore, the decision tree classifier can reach the highest accuracy among the three models.

(B) Performance Comparison: Normalization and Standardization

(Parameters for this subproblem: learning_rate = 0.01, iterations = 1000)
For this subproblem, I analyze the impact of normalization and standardization techniques on the **logistic regression model**.

Normalization (Min-Max Scaling):

This method will rescale the value to the range of $[0,1]$:

$$x' = \frac{\mathrm{x} - \min(\mathrm{x})}{\max(\mathrm{x}) - \min(\mathrm{x})}$$

Min-Max scaling can be useful when we want the value to be on the same scale. This can sometimes improve performance and the convergence speed. However, this method may be skewed to the minimum and maximum value, resulting to poor scaling.

Setting / Statistics	With Normalization	Without Normalization
Accuracy	66.67%	100%
Execution Time (s)	0.0178	0.0296

From the above table, we can find out that using the normalization method cannot improve performance. However, the execution time with normalization is faster.

Standardization (Z-Score Standardization):

This method can center the feature values around the mean with a unit standard deviation:

$$x' = \frac{x - \mu}{\sigma}$$

This method can be useful when we assume the data is normally distributed and can deal with outliers properly. However, this method may not lead to good result when the data distribution is skewed.

Setting / Statistics	With Standardization	Without Standardization
Accuracy	88.88%	100%
Execution Time (s)	0.0277	0.0296

From the above table, we can find out that using the standardization method cannot improve performance based on the given data. The execution time is slighted faster.

(C) Performance Comparison: Learning Rate and # of Iterations

For this subproblem, I analyze the impact of the changes of learning rate and number of iterations on the **linear regression model**.

Configuration 1: (Learning rate = 0.005, iterations = 4000)
Linear Regression MSE: 39.7053163178137
Execution Time: 0.015612 seconds

Configuration 2: (Learning rate = 0.01, iterations = 1000)
Linear Regression MSE: 43.41392463386301
Execution Time: 0.039103 seconds

From the above statistics, we can find out that using lower learning rate and more iterations can result in lower error (assume overfitting does not happen). A lower learning rate allows the model to make more precise updates to the weights during each iteration, leading to a more stable convergence process. A higher number of iterations give the model more chances to adjust the parameters and get closer to optimal. However, more iterations require more execution time. But if overfitting is observed, maybe lower learning rate and less iterations will be preferred.

(D) Impact of Hyperparameters on the Random Forest Model

Results of using different hyperparameters on the random forest model:

Setting / Task	# of trees = 50 Max depth = 5	# of trees = 100 Max depth = 5	# of trees = 50 Max depth = 10
Classification Task (Accuracy)	93.33%	95.56%	93.33%
Regression Task (MSE)	22.6455	22.9648	22.7834

From the above table, we can see that in the given data, a random forest with more trees can result in a better prediction in terms of classification task. Deeper trees can also give a better result, assume overfitting does not happen.

How ever, there is no significance difference between the MSE of the regression task. The following are the discussion about the two hyperparameters.

- **# of trees:** Generally, increasing the number of trees can improve model performance by making the predictions more stable. A higher number of trees can improve generalization, but with higher computational complexity and longer training time.
- **Maximum Depth:** Deeper trees have the ability to capture more complex patterns but may overfit. Shallow trees may underfit. Deeper trees increase the model's complexity and computational cost.

How I select the hyperparameters: By trial and error, use a loop to find the hyperparameters that gives the best result. (Best parameters I find: learning_rate = 0.05, iterations = 4000, max_depth = 5, n_estimators = 100. This is implemented in hw3.py)

(E) Strength and Weaknesses of the Implemented Models

Logistic Regression, Linear Regression:

Best for the following cases:

- When the relationship between the input features and the target variable is nearly linear.
- When you need a model that is easy to interpret and explain.

Advantages:

- Fast and efficient to train and predict.
- Less possible to overfit.

Limitations:

- It assumes a linear relationship between the input and outcome.
- Not suitable for complex relationships (e.g. non-linear) or interactions between features.

Decision Tree:

Best for the following cases:

- When the relationship between features and the target is non-linear.
- When you need a model that can be visualized and easy to understand.

Advantages:

- Easy to interpret and visualize.
- Can deal with non-linear relationships and interactions between features.

Limitations:

- When the tree is deep, the model can overfit easily.
- Sensitive to small changes in the data

Decision Tree:

Best for the following cases:

- When dealing with complex data with non-linear relationships.
- When you want a more robust model that is less possible to overfit than a single decision tree

Advantages:

- Reduces overfitting compared to individual decision trees.
- Higher prediction accuracy.

Limitations:

- Less interpretable compared to single decision trees.
- Needs lots of computational resources.

We can select the suitable model based on the points listed above based on our problem.

Reference:

https://medium.com/aimonks/harmonizing-the-scales-the-essential-role-of-feature-scaling-in-machine-learning-0b49e1e99aac

ChatGPT