CMSC 5724 Project #2: Margin Perceptron

1155215526 Ding Luxiao *1155215527 Chen Chen*
1155215555 Chen Yubin *1155215807 Zhang Jiarui*

Abstract

In this report, we have developed a margin perceptron algorithm designed to accurately classify multidimensional data labeled as 1 and -1.

Dataset

We will use the dataset given from the project page of the course. Includes:2dr16n10000, 4dr24n10000 and 8dr12n10000. The first line of each dataset contains three numbers *n, d,* and *r,* where *n* is the number of points, *d* is the dimensionality of the instance space, and *r* is the radius. The *i-th* line (where *i* goes from *2* to *n + 1*) gives the *(i-1)th* point in the dataset as: *x1, x2, ..., xd, label*, where the first *d* values are the coordinates of the point, and *label = 1 or -1*.

Algorithm

Using the algorithm in Lecture 5, pseudocode is show below:

```
1.  R = radius
2.  gamma_guess = R
3.  wight = [0...]
4.  WHILE True:
5.      MAX_ITERS = 12 * (R^2) / (gamma_guess^2)
6.      FOR epoch in MAX_ITERS:
7.          # Find violation point
8.          IF distance < gamma_guess/2 OR predict_label != real_label:
9.              UPDATE weight
10.         ELSE:
11.             BREAK
12.     gamma_guess /= 2
13.     IF gamma_guess too small OR no violation point found:
14.         BREAK
```

Experimental Process

Data preprocessing

DataDealer.py

The readData function reads a file, extracts input vectors and their corresponding labels, and determines the dimensionality of the vectors and the radius of the data points. It returns these processed data components as a tuple.

Margin Perceptron Development

MarginPerceptron.py

This class implements the margin perceptron algorithm for binary classification tasks. Here's a brief overview of its functionality:

Attributes:

dimension: The number of features in each data point.

radius: The radius used to calculate the number of training epochs.

input: The dataset containing the input vectors.

label: The labels corresponding to each input vector.

w: The weight vector that is initialized to zeros.

gamma_guess: The initial guess for the margin.

epochs: The maximum number of training epochs.

Methods:

__init__: Initializes a new instance of MarginPerceptron with the given parameters.

get_weights: Returns the current weight vector.

max_iteration: Calculates the maximum number of epochs based on the radius and the initial margin guess.

dot_product: Computes the dot product of two vectors.

norm: Computes the Euclidean norm (L2 norm) of a vector.

iterate: Iterates over the dataset to find the first violation point, which is a point that is either

misclassified or does not satisfy the margin condition.

update_weights: Updates the weight vector based on the violation point found.

train: Trains the perceptron using the margin perceptron algorithm. It iterates until no more violation points are found or until a forced termination condition is met (when gamma_guess becomes too small).

calculate_margin: Calculates the minimum margin over all data points for the current weight vector.

The MarginPerceptron object is initialized with dimension, radius, input, and label. The train() method is then called on the MarginPerceptron instance to start training.

After iterating through all datasets, the code prints a summary of the results for each dataset, allowing a clear comparison of the gamma guess, weight, and margin achieved on each dataset.

The key methods in the class are shown below:

```
def iterate(self):
    for i, point in enumerate(self.input):
        point_label = self.label[i]
        dot_product = self.dot_product(self.w, point)
        # Determine the predicted label
        predict_label = 1 if dot_product >= 0 else -1
        # Calculate distance to margin
        norm_w = self.norm(self.w)
        distance = abs(dot_product) / norm_w if norm_w != 0 else 0
        # Check for violation: margin or misclassification
        if (distance < (self.gamma_guess / 2.0)) or (predict_label *
point_label < 0):
            return i  # Violation point index
        return -1  # No violation point found

def update_weights(self, index: int):
    for j in range(self.dimension):
        self.w[j] += self.label[index] * float(self.input[index][j])

def train(self):
    while True:
        for _ in range(self.epochs):
            violation_point_index = self.iterate()
```

```
            # Self-termi: No viola point found, training complete
            if violation_point_index == -1:
                print("Training completed with self-termination.")
                return False
            # Update weights based on the violation point
            print(f"Violation point index: {violation_point_index}")
            print(f"Current weights: {self.w}")
            self.update_weights(violation_point_index)
            print(f"Updated weights: {self.w}")
            print("-----------------------------------")

        # Forced-termi: Reduce gamma_guess and recompute epochs
        self.gamma_guess /= 2
        if self.gamma_guess <= 1e-8:
            print("Gamma guess is too small to continue training.")
            return False
            print(f"Forced termination: Reducing gamma_guess to {self.gamma_guess} and restarting training.")
            self.epochs = self.max_iteration(self.radius, self.gamma_guess)

def calculate_margin(self):
    margins = []
    norm_w = self.norm(self.w)
    if norm_w == 0:
        return 0.0  # Avoid division by zero
    for i, point in enumerate(self.input):
        margin = (self.dot_product(self.w, point) * self.label[i]) / norm_w
        margins.append(margin)
    return min(margins) if margins else 0.0
```

Experimental Result

We trained the model in all three datasets using MarginPerceptron.py.

After training all, we find the final result as is shown in Table 1, saved in res.txt.

Table 1: Results of Margin Perceptron Training

result\dataset	2d-r16-n10000	4d-r24-n10000	8d-r12-n10000
gamma	4.0	12.0	6.0
weight	[33.088, -272.951]	[32.930, -77.161, -113.534, 98.710]	[-19.555, -28.034, -56.472, -13.556, 70.532, 49.546, 61.369, -10.071]
margin	2.630	6.007	3.031

Based on the results, here is an analysis of the Margin Perceptron algorithm's performance across different datasets:

1. Relationship Between Margin and Gamma Guess

2d-r16-n10000 Dataset:

The initial gamma guess is 4.0, and the final margin achieved is 2.63.

The weight vector is [33.088,−272.95]

In this dataset, the relatively small 2D instance space with a radius of 16 limits the achievable margin, but the algorithm still finds a reasonable decision boundary.

4d-r24-n10000 Dataset:

The initial gamma guess is 12.0, with a margin of 6.01, significantly higher than in the 2D case.

The weight vector is [32.930,−77.161,−113.534,98.710] with notable variation across dimensions, indicating that each dimension contributes differently to classification.

The higher margin and gamma in this dataset suggest that, as dimensionality increases, the Margin Perceptron can better adapt to complex feature spaces and achieve a wider decision boundary.

8d-r12-n10000 Dataset:

The gamma guess is 6.0, resulting in a margin of 3.03.

The weight vector is [−19.555, −28.034, −56.472, −13.556, 70.532, 49.546, 61.369, −10.071], showing dispersed weights across dimensions.

In this higher-dimensional dataset with a relatively small radius (r=12), the weight distribution is more scattered, and the margin is lower, possibly due to the challenge of finding a broad margin in a densely packed feature space.

2. Effect of Dimensionality on Margin

As the data dimensionality increases from 2D to 4D, the margin significantly increases, suggesting that the Margin Perceptron algorithm is better able to establish a larger classification boundary in higher-dimensional spaces.

However, with further dimensionality increase to 8D, the margin slightly decreases, potentially due to data sparsity in high-dimensional spaces, making it harder for the classifier to maintain a large margin.

3. Effect of Radius on Margin

The radius of the dataset also significantly impacts the final margin. Datasets with a larger radius (e.g., 4d-r24-n10000) show higher margins, suggesting that in a larger instance space, the Margin Perceptron is more capable of finding a stable decision boundary.

In datasets with smaller radius values (e.g., 8d-r12-n10000), the margin is relatively low, likely due to the more compact instance space and denser data distribution.

4. Summary

The Margin Perceptron algorithm displays varied margin sizes across datasets with different dimensionalities and radii. Generally, with an increase in both dimensions and radius, the achievable margin grows, though there is a tendency for margin reduction in high-dimensional, low-radius conditions.

The differences in the weight vectors and margin values indicate how dataset characteristics influence classification results.

Appendix:

1. GitHub repo: https://github.com/Chenchen2001/5724Proj2

2. Part of Output in res.txt

```
========== TRAINING ON 2d-r16-n10000 ==========
Violation point index: 0
Current weights: [0.0, 0.0]
Updated weights: [8.459748692855033, -5.621530829461882]
-----------------------------------
Violation point index: 1
Current weights: [8.459748692855033, -5.621530829461882]
Updated weights: [12.696957513609405, -12.08432381879284]
-----------------------------------
……
-----------------------------------
Violation point index: 15
Current weights: [26.339398459843217, -269.8048498249309]
Updated weights: [33.088781575317, -272.9515380243361]
-----------------------------------
Forced termination: Reducing gamma_guess to 4.0 and restarting training.
Training completed with self-termination.
========== RESULT ==========
After training, gamma_guess is    4.0
After training, weight is [33.088781575317, -272.9515380243361]
Final margin after training is: 2.630504567982789
====================================
……
========== FINAL RESULT STATISTIC ==========
---------- 2d-r16-n10000 ----------
The gamma_guess is: 4.0
The weight is: [33.088781575317, -272.9515380243361]
The margin is: 2.630504567982789
---------- 4d-r24-n10000 ----------
The gamma_guess is: 12.0
The weight is: [32.93034675105003, …, 98.71015647510539]
The margin is: 6.007701533717606
---------- 8d-r12-n10000 ----------
The gamma_guess is: 6.0
The weight is: [-19.55598600673023, …, -10.071357363006381]
The margin is: 3.031740039940767
====================================
```