Module 6: The Perceptron Algorithm

Machine Learning Course

Contents

1 Introduction to the Perceptron

1.1 Historical Context

The Perceptron algorithm is one of the earliest and most influential algorithms in machine learning. It was developed in the late 1950s by Frank Rosenblatt, inspired by the functioning of a biological neuron. Despite its simplicity, the Perceptron laid the groundwork for many modern machine learning techniques, including neural networks and deep learning.

1.2 From Linear Boundaries to Learning Algorithms

In previous discussions, we explored linear boundaries for binary classification:

- Data points $\mathbf{x} \in \mathbb{R}^d$ with binary labels $y \in \{-1, +1\}$
- Linear decision boundaries defined by $\mathbf{w} \cdot \mathbf{x} + b = 0$
- Prediction rule: $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$

The Perceptron algorithm provides a systematic way to learn the parameters $\mathbf{w}$ and b from training data.

2 Linear Classification Revisited

2.1 Binary Classification Framework

- Input: Feature vectors $\mathbf{x} \in \mathbb{R}^d$
- Output: Binary labels $y \in \{-1, +1\}$
- Goal: Learn a linear decision boundary that separates positive from negative examples

2.2 Linear Classifier Parameters

A linear classifier is defined by:

- Weight vector $\mathbf{w} \in \mathbb{R}^d$
- Bias term $b \in \mathbb{R}$

2.3 Prediction and Correctness

For a data point $\mathbf{x}$ with true label y:

- Prediction: $\hat{y} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$
- Classifier is correct if $y = \hat{y}$
- Equivalently, classifier is correct if $y(\mathbf{w} \cdot \mathbf{x} + b) > 0$

This last formulation is particularly useful for the Perceptron algorithm.

3 Loss Function for Classification

3.1 Defining the Loss

To learn a linear classifier, we need to define a loss function that quantifies how "wrong" our model is on a given example.

For a point $(\mathbf{x}, y)$, a natural loss function is:

- If $y(\mathbf{w} \cdot \mathbf{x} + b) > 0$ (correct prediction): Loss $= 0$
- If $y(\mathbf{w} \cdot \mathbf{x} + b) \leq 0$ (incorrect prediction): Loss $= -y(\mathbf{w} \cdot \mathbf{x} + b)$

This loss function has several desirable properties:

- Zero loss for correct predictions
- Positive loss for incorrect predictions
- Higher loss for predictions that are "more wrong"

3.2 Example of Loss Calculation

- Suppose $y = -1$ (true label) and $\mathbf{w} \cdot \mathbf{x} + b = 0.1$
- We predict $\hat{y} = +1$ (incorrect)
- Loss $= -(-1)(0.1) = 0.1$ (small loss)
- Suppose $y = -1$ (true label) and $\mathbf{w} \cdot \mathbf{x} + b = 10$
- We predict $\hat{y} = +1$ (incorrect)
- Loss $= -(-1)(10) = 10$ (large loss)

4 Stochastic Gradient Descent

4.1 Optimization Approach

To find the optimal parameters $\mathbf{w}$ and b, we use stochastic gradient descent (SGD):

- Process one training example at a time
- Update parameters in the direction that reduces the loss
- Repeat until convergence

4.2 Gradient of the Loss

For the loss function defined above:

- If $y(\mathbf{w} \cdot \mathbf{x} + b) > 0$ (correct): Gradient $= 0$ (no update)
- If $y(\mathbf{w} \cdot \mathbf{x} + b) \leq 0$ (incorrect):
  - Gradient with respect to $\mathbf{w}$ is $-y\mathbf{x}$
  - Gradient with respect to b is $-y$

4.3 Update Rules

The SGD update rules are:

- $\mathbf{w} = \mathbf{w} + \alpha y \mathbf{x}$
- $b = b + \alpha y$

where α is the learning rate (step size).

For simplicity, the Perceptron algorithm uses $\alpha = 1$.

5 The Perceptron Algorithm

5.1 Algorithm Description

The Perceptron Algorithm

1. Initialize $\mathbf{w} = \mathbf{0}$ and $b = 0$
2. Repeat until convergence:
   (a) For each training example $(\mathbf{x}, y)$:
      i. If $y(\mathbf{w} \cdot \mathbf{x} + b) \leq 0$ (misclassified):
         A. $\mathbf{w} = \mathbf{w} + y\mathbf{x}$
         B. $b = b + y$

5.2 Intuitive Explanation

The Perceptron algorithm has an intuitive interpretation:

- If a positive example ($y = +1$) is misclassified, add its feature vector to $\mathbf{w}$ and increase b
- If a negative example ($y = -1$) is misclassified, subtract its feature vector from $\mathbf{w}$ and decrease b

This gradually adjusts the decision boundary to correctly classify more training examples.

6 The Perceptron in Action

6.1 Example: Linearly Separable Data

The lecture demonstrated the Perceptron algorithm on a dataset of 85 linearly separable points:

- The algorithm was run multiple times with different random orderings of the data
- Each run converged to a different linear boundary that perfectly separated the classes
- The number of iterations required for convergence varied (e.g., 9, 15, 8, 14 iterations)

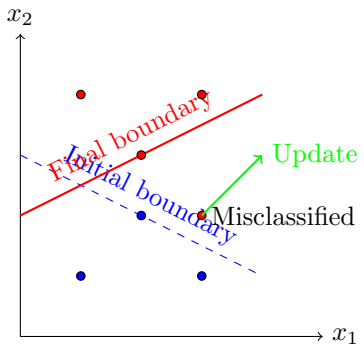

Figure 1: Illustration of a Perceptron update. The misclassified positive point causes the decision boundary to shift.

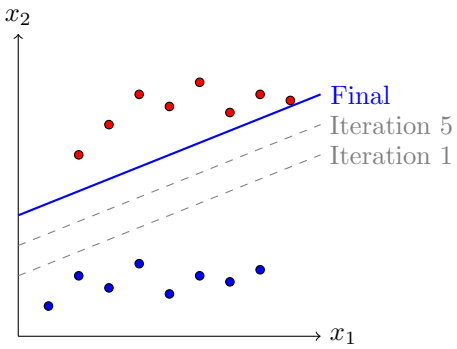

Figure 2: Illustration of Perceptron convergence on linearly separable data. The decision boundary evolves over iterations until it perfectly separates the classes.

6.2 Visualization of Convergence

7 Convergence Properties

7.1 Convergence Theorem

A fundamental result for the Perceptron algorithm:

Perceptron Convergence Theorem
If the training data is linearly separable, then:

- The Perceptron algorithm will find a linear classifier with zero training error
- It will converge within a finite number of steps

7.2 Margin and Convergence Rate

The number of iterations required for convergence depends on the margin:

- Margin: A measure of the separation between the two classes
- Larger margin → Faster convergence
- The upper bound on the number of iterations is proportional to $1/\gamma^2$, where γ is the margin

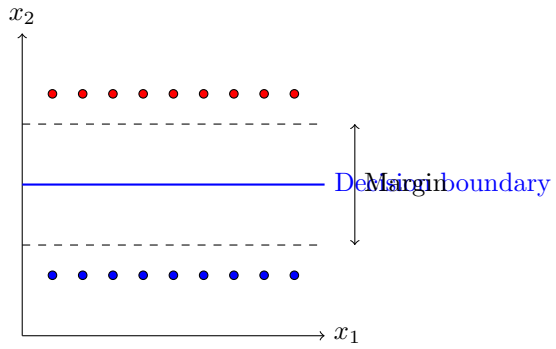

Figure 3: Illustration of margin in a linearly separable dataset. The margin is the distance between the decision boundary and the closest data points.

8 Limitations and Extensions

8.1 Non-linearly Separable Data

The Perceptron algorithm has a significant limitation:

- It is only guaranteed to converge if the data is linearly separable
- For non-linearly separable data, it may oscillate indefinitely

8.2 Pocket Algorithm

A simple extension to handle non-linearly separable data:

- Run the standard Perceptron algorithm
- Keep track of the best weights found so far (those with the lowest training error)
- Return these "pocket" weights after a fixed number of iterations

8.3 Kernel Perceptron

The Perceptron can be extended to handle non-linear decision boundaries using the kernel trick:

- Implicitly map the data to a higher-dimensional space where it becomes linearly separable
- Apply the Perceptron algorithm in this higher-dimensional space
- Common kernels: polynomial, radial basis function (RBF)

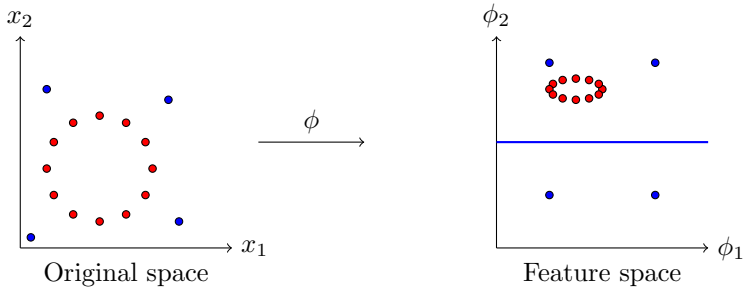

Figure 4: Kernel trick: Mapping non-linearly separable data to a space where it becomes linearly separable

9 Relationship to Neural Networks

9.1 The Perceptron as a Neuron

The Perceptron can be viewed as a single artificial neuron:

- Inputs: Feature vector $\mathbf{x}$
- Weights: Vector $\mathbf{w}$ and bias b
- Activation function: Sign function
- Output: Binary prediction $\hat{y}$

9.2 Multi-layer Perceptrons

Modern neural networks extend the Perceptron concept:

- Multiple layers of neurons
- Differentiable activation functions (e.g., sigmoid, ReLU)
- Trained using backpropagation and gradient descent

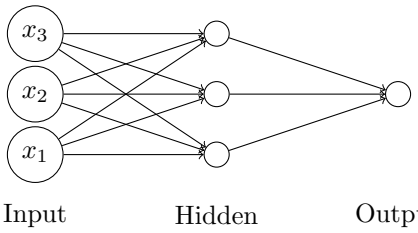

Figure 5: A multi-layer perceptron with one hidden layer

10 Worked Examples

10.1 Example 1: Perceptron Learning on a Simple Dataset

Consider the following 2D dataset:

- Positive examples: $(1, 1)$, $(2, 2)$
- Negative examples: $(1, 2)$, $(2, 1)$

Let's trace through the Perceptron algorithm:

Initialization: $\mathbf{w} = (0, 0)$, $b = 0$

Iteration 1:

- Example $(1, 1)$, $y = +1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = 1 \cdot ((0, 0) \cdot (1, 1) + 0) = 0 \leq 0 \tag{1}$$

  Misclassified, so update:

$$\mathbf{w} = (0, 0) + 1 \cdot (1, 1) = (1, 1) \tag{2}$$
$$b = 0 + 1 = 1 \tag{3}$$

- Example $(2, 2)$, $y = +1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = 1 \cdot ((1, 1) \cdot (2, 2) + 1) = 5 > 0 \tag{4}$$

  Correctly classified, no update.

- Example $(1, 2)$, $y = -1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = -1 \cdot ((1, 1) \cdot (1, 2) + 1) = -4 < 0 \tag{5}$$

  Correctly classified, no update.

- Example $(2, 1)$, $y = -1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = -1 \cdot ((1, 1) \cdot (2, 1) + 1) = -4 < 0 \tag{6}$$

  Correctly classified, no update.

After one pass through the data, all examples are correctly classified. The final decision boundary is $x_1 + x_2 + 1 = 0$ or $x_2 = -x_1 - 1$.

10.2 Example 2: Non-convergence on Non-linearly Separable Data

Consider the XOR problem:

- Positive examples: $(0, 0)$, $(1, 1)$
- Negative examples: $(0, 1)$, $(1, 0)$

This dataset is not linearly separable. Let's see what happens with the Perceptron:

Initialization: $\mathbf{w} = (0, 0)$, $b = 0$

Iteration 1:

- Example $(0, 0)$, $y = +1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = 1 \cdot ((0, 0) \cdot (0, 0) + 0) = 0 \leq 0 \tag{7}$$

  Misclassified, so update:

$$\mathbf{w} = (0, 0) + 1 \cdot (0, 0) = (0, 0) \tag{8}$$
$$b = 0 + 1 = 1 \tag{9}$$

- Example $(1, 1)$, $y = +1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = 1 \cdot ((0, 0) \cdot (1, 1) + 1) = 1 > 0 \tag{10}$$

  Correctly classified, no update.

- Example $(0, 1)$, $y = -1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = -1 \cdot ((0, 0) \cdot (0, 1) + 1) = -1 < 0 \tag{11}$$

  Correctly classified, no update.

- Example $(1, 0)$, $y = -1$:

$$y(\mathbf{w} \cdot \mathbf{x} + b) = -1 \cdot ((0, 0) \cdot (1, 0) + 1) = -1 < 0 \tag{12}$$

  Correctly classified, no update.

At this point, $\mathbf{w} = (0, 0)$ and $b = 1$, which classifies everything as positive. This is clearly not correct for the negative examples. If we continue iterating, the algorithm will keep updating without converging to a solution that correctly classifies all examples.

11 Practical Considerations

11.1 Initialization

While the standard Perceptron initializes $\mathbf{w} = \mathbf{0}$ and $b = 0$, different initializations can lead to different solutions and convergence rates.

11.2 Data Ordering

The order in which training examples are processed can significantly impact:

- The number of iterations required for convergence
- The specific decision boundary found

Randomizing the order of examples between epochs is a common practice.

11.3 Learning Rate

While the standard Perceptron uses a learning rate of 1, a smaller learning rate can sometimes lead to more stable convergence, especially in noisy settings.

11.4 Feature Scaling

As with most linear models, the Perceptron benefits from feature scaling:

- Standardization: $x' = \frac{x-\mu}{\sigma}$
- Min-max scaling: $x' = \frac{x-\min(x)}{\max(x)-\min(x)}$

12 Summary and Key Takeaways

12.1 Core Concepts

- The Perceptron is an algorithm for learning linear classifiers
- It updates the model parameters only when a misclassification occurs
- The update rule is simple: add the feature vector (scaled by the label) to the weight vector
- For linearly separable data, the Perceptron is guaranteed to converge in a finite number of steps

12.2 Strengths

- Simple and intuitive algorithm
- Guaranteed convergence for linearly separable data
- Online learning capability (can process one example at a time)
- Historically significant as a foundation for neural networks

12.3 Limitations

- Only works for linearly separable data
- No convergence guarantee for non-linearly separable data
- The solution depends on the order of training examples
- Does not provide probability estimates

12.4 Extensions and Related Concepts

- Pocket Algorithm for non-linearly separable data
- Kernel Perceptron for non-linear decision boundaries
- Multi-layer Perceptrons and neural networks
- Support Vector Machines for maximum-margin linear classification