UCLA

CS M146: Introduction to Machine Learning
Perceptrons

Aditya Grover

UCLA

The instructor gratefully acknowledges Sriram Sankararaman (UCLA), Piyush Rai (IIT) and Alan Fern (OSU) for some of the materials and organization used in these slides, and many others who made their course materials freely available online.

https://aditya-grover.github.io/ @adityagrover_

Classification

Given

A training dataset consisting of n labelled training examples

- Input **features** $\mathbf{X} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(n)}\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$
- Corresponding **labels** $\boldsymbol{y} = \{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)}$ is **discrete** with some domain $\mathcal{Y}$.
  - E.g., Binary classification: $\mathcal{Y} = \{-1, 1\}$

Output

Hypothesis function $h: \mathbb{R}^d \to \mathcal{Y}$ such that $h(\boldsymbol{x}) \approx y$

Hyperplanes

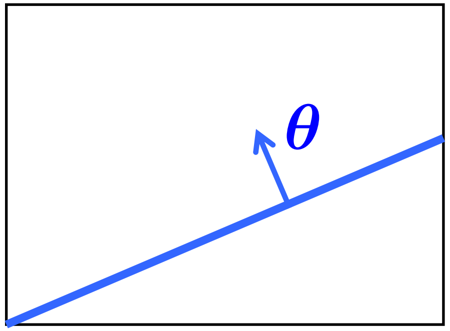
- Linear models represent hypothesis as **hyperplanes**

$$\theta_1 x_1 + \theta_2 x_2 + \ \dots + \theta_d x_d + b = 0$$

- Hyperplane partitions $\mathbb{R}^d$ into two half-spaces

  Half-space 1: $\theta_1 x_1 + \theta_2 x_2 + \ \dots + \theta_d x_d + b > 0$

  Half-space 2: $\theta_1 x_1 + \theta_2 x_2 + \ \dots + \theta_d x_d + b > 0$

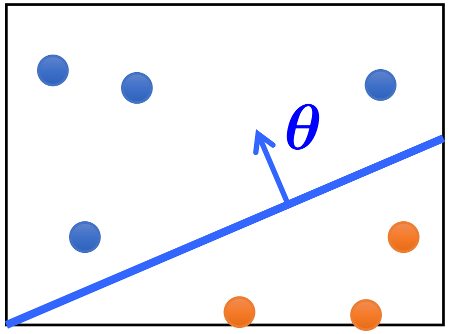


- Defined by the normal vector $\boldsymbol{\theta} \in \mathbb{R}^d$
  - $\boldsymbol{\theta}$ is orthogonal to any vector lying on the hyperplane
- Note: If we include bias as $b = \theta_0$ and add an extra dimension $x_0 = 1$, then we can represent $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$ and hyperplane passes through the origin

Perceptron

- Consider classification with +1, -1 labels
- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$



- For example: **Perceptrons**

$$h_\theta(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{\theta}^T \boldsymbol{x}) \text{ where } \mathrm{sign}(\boldsymbol{z}) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \end{cases}$$

(undefined if $z = 0$)

Learning Perceptrons

$$h_\theta(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^T\boldsymbol{x}) \text{ where } \text{sign}(\boldsymbol{z}) = \begin{cases} +1 & \text{if } z > 0 \\ -1 & \text{if } z < 0 \end{cases}$$

- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

- If the prediction matches the label, make no change
- Otherwise, adjust $\boldsymbol{\theta}$

Error-Driven Learning

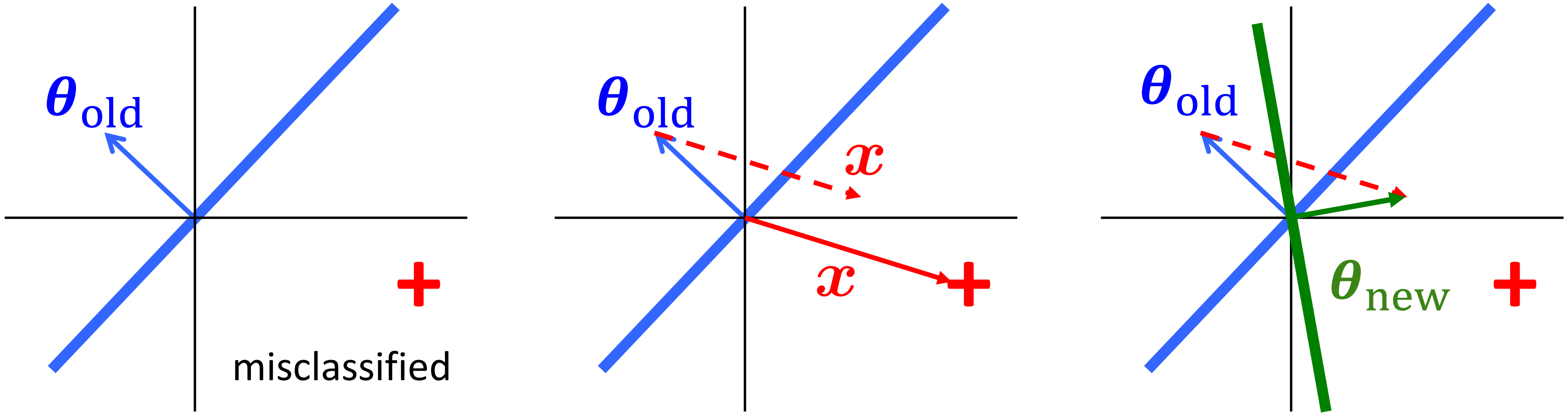
- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{\alpha}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)} \right)}_{\text{either 2 or -2}} x_j^{(i)}$$

- Rewrite as $\theta_j \leftarrow \theta_j + \alpha y^{(i)} x_j^{(i)}$ if $\boldsymbol{x}^{(i)}$ is misclassified
- For simplicity, we will fix $\alpha = 1$ henceforth

Perceptron Rule: If $\boldsymbol{x}^{(i)}$ is misclassified, do $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + y^{(i)} \boldsymbol{x}^{(i)}$

Why the Perceptron Update Works



Why the Perceptron Update Works

- Consider the misclassified example, say with label $y = +1$
- Perceptron wrongly thinks that $\boldsymbol{\theta}_{\text{old}}^T \boldsymbol{x} < 0$
- Update:

$$\boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} + y\boldsymbol{x} \quad = \boldsymbol{\theta}_{\text{old}} + \boldsymbol{x} \qquad (\text{since } y = +1)$$

- Note:

$$\begin{aligned} \boldsymbol{\theta}_{\text{new}}^{\mathsf{T}} \boldsymbol{x} &= (\boldsymbol{\theta}_{\text{old}} + \boldsymbol{x})^{\mathsf{T}} \boldsymbol{x} \\ &= \boldsymbol{\theta}_{\text{old}}^{\mathsf{T}} \boldsymbol{x} + \boldsymbol{x}^{\mathsf{T}} \boldsymbol{x} \end{aligned}$$

$\|\boldsymbol{x}\|_2^2 > 0$

- Therefore, $\boldsymbol{\theta}_{\text{new}}^T \boldsymbol{x}$ is less negative than $\boldsymbol{\theta}_{\text{old}}^T \boldsymbol{x}$
  - So, we are making ourselves more correct on this example!

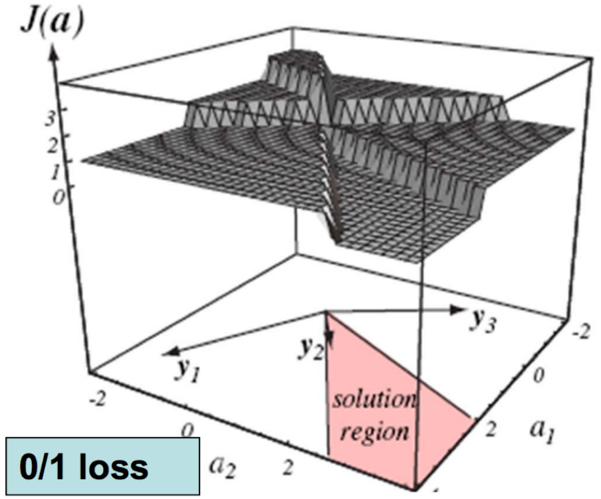
Loss Function for Classification

- Ideal per-instance loss: 0/1 loss

$$\ell_{0/1}\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right) = 0 \text{ if } h_\theta\left(\boldsymbol{x}^{(i)}\right) = y^{(i)} \text{ and 1 otherwise}$$

- Candidate loss function: average 0/1 loss

$$J_{0/1}(\boldsymbol{\theta}) = \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \ell_{0/1}\left(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}\right)$$



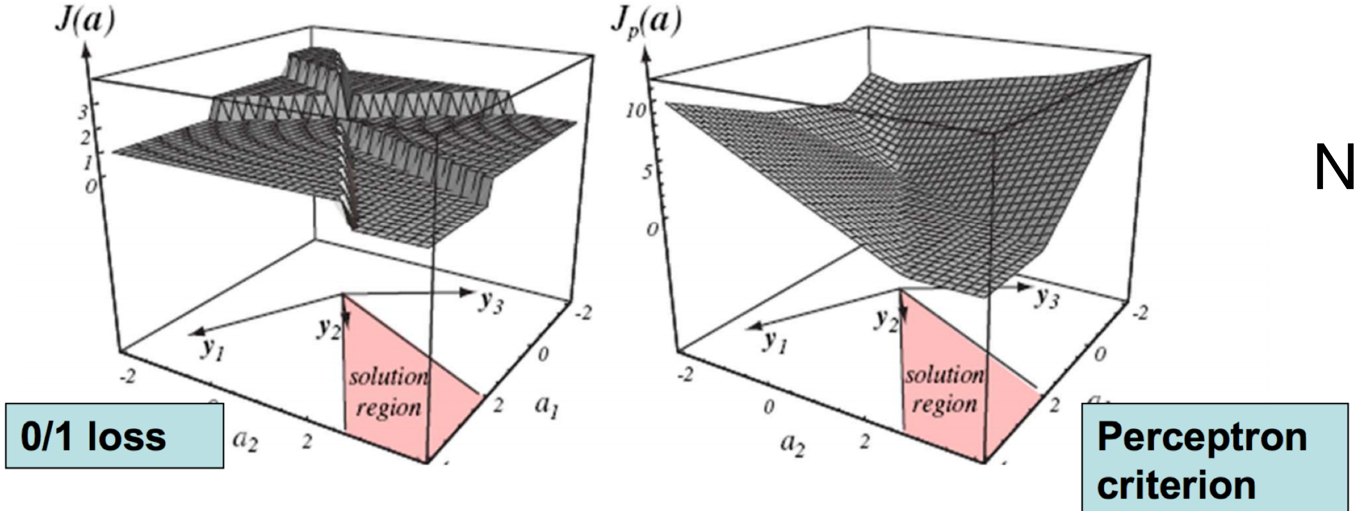
Doesn't produce a useful gradient

The Perceptron Loss Function

- The perceptron uses the following “surrogate” loss function

$$J_{\text{perceptron}}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \max(0, -y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)})$$

- Prediction is correct if $y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} > 0$
- Perceptron loss is 0 if the prediction is correct
- Otherwise it is the confidence in the misprediction



Nice gradient

Perceptron Algorithm

Given training data $\left\{(\boldsymbol{x}^{(i)}, y^{(i)})\right\}_{i=1}^{n}$
Let $\boldsymbol{\theta} \leftarrow [0, 0, \ldots, 0]$
Repeat:
    for $i = 1 \ldots n$, do
        if $y^{(i)}\boldsymbol{\theta}^T\boldsymbol{x}^{(i)} \leq 0$ // prediction for ith instance is incorrect
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha y^{(i)}\boldsymbol{x}^{(i)}$
Return $\boldsymbol{\theta}$

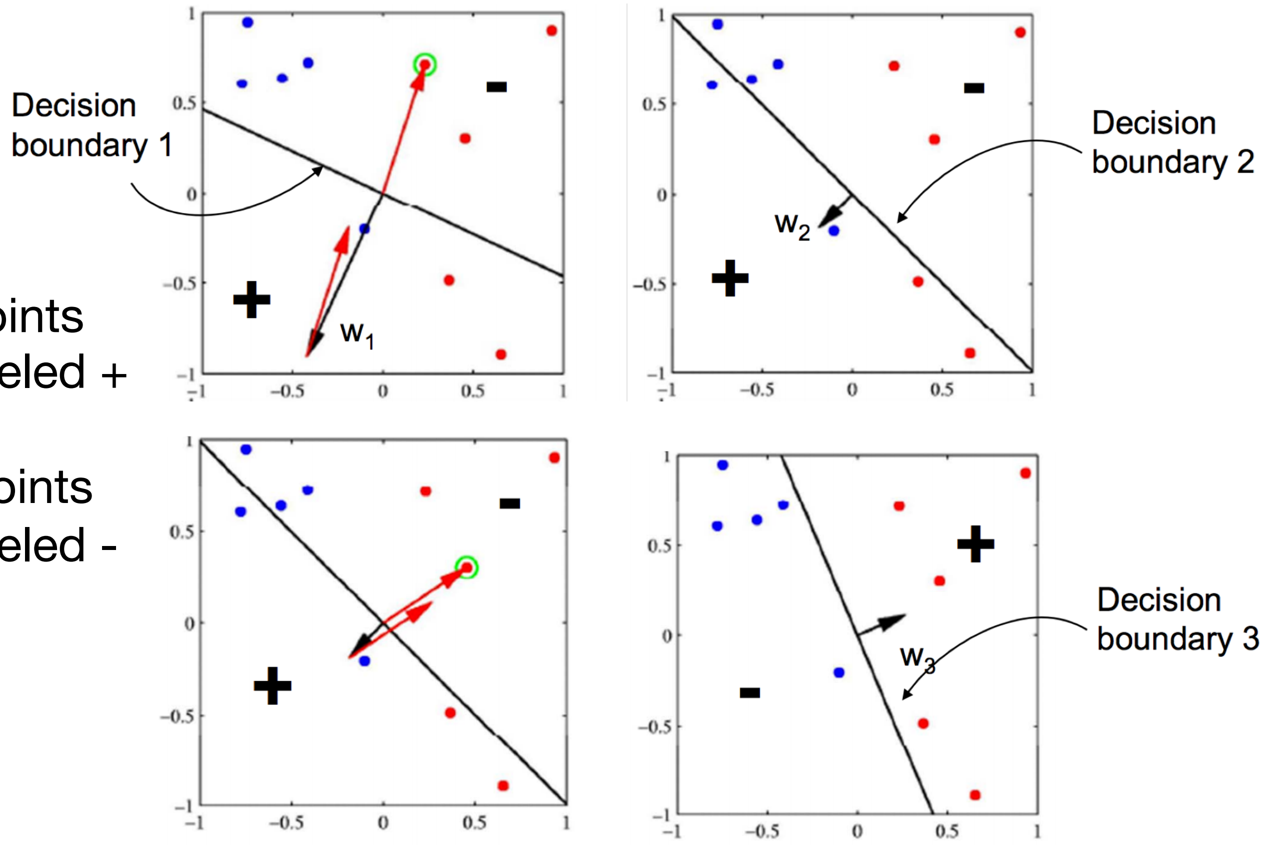
- How often to repeat? **Algorithmic Hyperparameter**
- Practical tip: Shuffling the data improves convergence speed

Perceptron Algorithm

When an error is made, moves the weight in a direction that corrects the error

Red points are labeled +

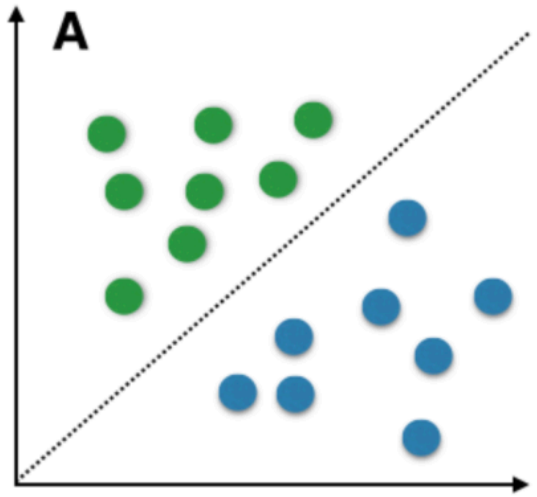
Blue points are labeled -



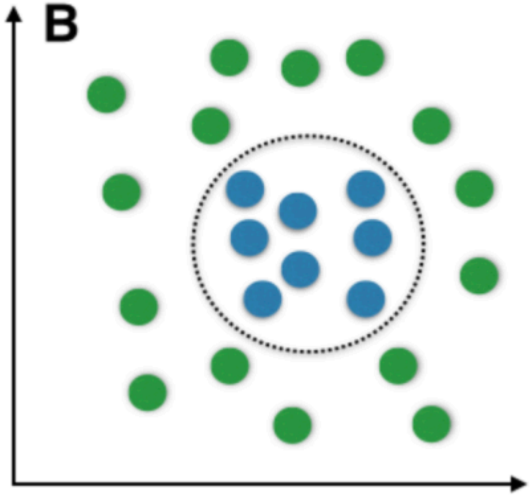
See the perceptron in action: www.youtube.com/watch?v=vGwemZhPlsA

Linear Separability

- A dataset is **linearly separable** if there exists some hyperplane that puts all the positive examples on one side and all the negative examples on the other side.



Linearly separable



Linearly inseparable

Convergence

- **Convergence theorem** (stated without proof)

  If there exist a set of perceptron parameters such that the data is linearly separable, the perceptron algorithm will converge.

- **Cycling theorem** (stated without proof)

  If the training data is not linearly separable, then the learning algorithm will eventually repeat the same set of parameters and enter an infinite loop

Improving the Perceptron

- The perceptron produces many $\boldsymbol{\theta}$ during training

- The standard perceptron simply uses the final $\boldsymbol{\theta}$ at test time
  - This may sometimes not be a good idea!
  - Some other $\boldsymbol{\theta}$ may be correct on 1,000 consecutive examples, but one mistake ruins it!

- **Idea:** Use a combination of multiple perceptrons
  - (i.e., neural networks!)

Summary

3 Steps:

- Representation

Linear functions: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{\theta}^T \boldsymbol{x})$

- Define a loss function

Perceptron loss: $J(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \max\left(0, -y^{(i)}\boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)$

- Optimize loss to find best hypothesis

Gradient descent: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\alpha}{2}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)\boldsymbol{x}^{(i)}$

or $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \mathbf{1}\left[h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) \neq y^{(i)}\right] y^{(\boldsymbol{i})}\boldsymbol{x}^{(i)}$

Summary

Perceptrons

- A linear model for classification based on error-driven learning

Learning a Perceptron

- Surrogate objective has better gradients than 0/1 loss
- Guaranteed to converge if data is linearly separable