T5: Linear methods and Perceptron

Fundamentos del Aprendizaje Automático

Curso 2025/2026

Structure

Problem statement

- So far:

Problem statement

- So far:

  T2: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$

Problem statement

- So far:

  T2: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$

  T4: Nonparametric density estimator for approximating *likelihood*

Problem statement

- So far:
  - **T2**: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$
  - **T4**: Nonparametric density estimator for approximating *likelihood*
  - **T4**: Distance-based classification for *posterior* $\hat{P}(\omega|\mathbf{x})$

Problem statement

- So far:

  T2: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$
  T4: Nonparametric density estimator for approximating *likelihood*
  T4: Distance-based classification for *posterior* $\hat{P}(\omega|\mathbf{x})$

- **T5**: Parametric model for discriminant functions $\Rightarrow g(\mathbf{x}; \boldsymbol{\theta})$

Problem statement

- So far:
  T2: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$
  T4: Nonparametric density estimator for approximating *likelihood*
  T4: Distance-based classification for *posterior* $\hat{P}(\omega|\mathbf{x})$

- **T5**: Parametric model for discriminant functions $\Rightarrow g(\mathbf{x}; \boldsymbol{\theta})$
  $\rightarrow$ $\boldsymbol{\theta} \Rightarrow$ weights of the model

Problem statement

- So far:

  T2: Parametric models for modeling *likelihood*: $P(\mathbf{x}|\omega) \approx \hat{P}(\mathbf{x}|\omega, \boldsymbol{\theta})$
  T4: Nonparametric density estimator for approximating *likelihood*
  T4: Distance-based classification for *posterior* $\hat{P}(\omega|\mathbf{x})$

- **T5**: Parametric model for discriminant functions $\Rightarrow g(\mathbf{x}; \boldsymbol{\theta})$
  $\rightarrow$ $\boldsymbol{\theta} \Rightarrow$ weights of the model
  $\rightarrow$ $J(\boldsymbol{\theta}) \Rightarrow$ criterion function to estimate $\boldsymbol{\theta}$

Outline

Linear discriminant function

Linear discriminant function

- Simplest type of ML model

Linear discriminant function

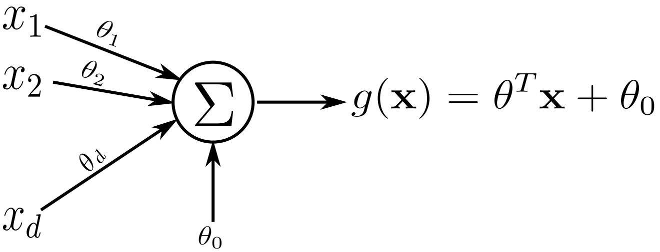
- Simplest type of ML model
- **Assumption**: output is a linear combination of the input features

Linear discriminant function

- Simplest type of ML model
- **Assumption**: output is a linear combination of the input features
- Two-category case ($\mathcal{W} = \{\omega_1, \omega_2\}$):

$$g(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x} + \theta_0$$

$\mathbf{x} \in \mathbb{R}^d$ feature vector
$\boldsymbol{\theta} \in \mathbb{R}^d$ weight vector
$\theta_0 \in \mathbb{R}$ weight threshold



Decision surface

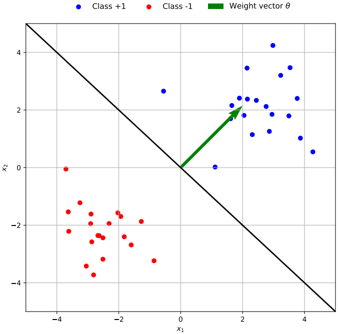
- Decision surface ($\mathcal{H}$): Given by equation $g(\mathbf{x}) = 0$

Decision surface

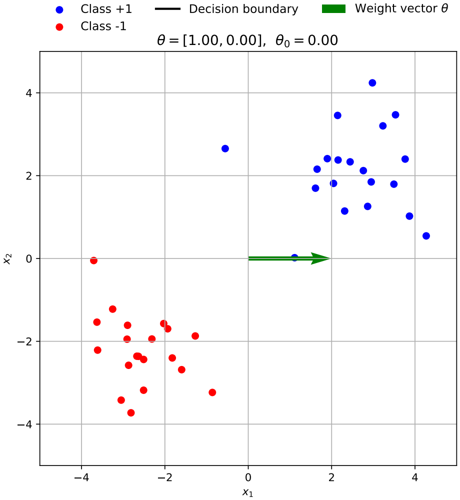
- Decision surface ($\mathcal{H}$): Given by equation $g(\mathbf{x}) = 0$
  - → If $g(\mathbf{x}) > 0$: Region $\mathcal{R}_1 \Rightarrow \hat{\omega} = \omega_1$
  - → If $g(\mathbf{x}) < 0$: Region $\mathcal{R}_2 \Rightarrow \hat{\omega} = \omega_2$

Decision surface

- Decision surface ($\mathcal{H}$): Given by equation $g(\mathbf{x}) = 0$
  - $\rightarrow$ If $g(\mathbf{x}) > 0$: Region $\mathcal{R}_1 \Rightarrow \hat{\omega} = \omega_1$
  - $\rightarrow$ If $g(\mathbf{x}) < 0$: Region $\mathcal{R}_2 \Rightarrow \hat{\omega} = \omega_2$
- Geometric interpretation:
  - $\rightarrow$ Vector $\boldsymbol{\theta}$: normal to surface $\mathcal{H}$ and defines its *orientation*
  - $\rightarrow$ Bias θ_0: Shifts surface $\mathcal{H}$ along $\boldsymbol{\theta}$



Decision surface



Formalization

- Multiclass: Generalizes to one linear function per label:

$$g_k(\mathbf{x}) = \boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}, \quad k = 1, 2, ..., |\mathcal{W}|$$

Formalization

- Multiclass: Generalizes to one linear function per label:

$$g_k(\mathbf{x}) = \boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}, \quad k = 1, 2, ..., |\mathcal{W}|$$

- Decision rule:

$$\hat{\omega} = \underset{k=1,...,|\mathcal{W}|}{\arg\max}\ g_k(\mathbf{x}) = \underset{k=1,...,|\mathcal{W}|}{\arg\max} \left(\boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}\right)$$

Formalization

- Multiclass: Generalizes to one linear function per label:

$$g_k(\mathbf{x}) = \boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}, \quad k = 1, 2, ..., |\mathcal{W}|$$

- Decision rule:

$$\hat{\omega} = \underset{k=1,...,|\mathcal{W}|}{\arg\max}\ g_k(\mathbf{x}) = \underset{k=1,...,|\mathcal{W}|}{\arg\max} \left(\boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}\right)$$

- Decision surface(s):
  - $\rightarrow$ Each duple $(\boldsymbol{\theta}_k^T, \theta_{0k})$ defines a hyperplane

Formalization

- Multiclass: Generalizes to one linear function per label:

$$g_k(\mathbf{x}) = \boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}, \quad k = 1, 2, ..., |\mathcal{W}|$$

- Decision rule:

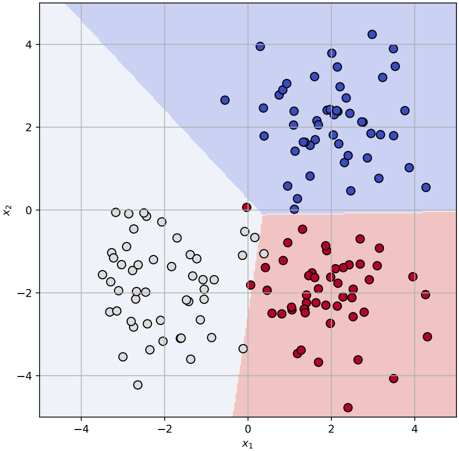
$$\hat{\omega} = \underset{k=1,...,|\mathcal{W}|}{\arg\max}\; g_k(\mathbf{x}) = \underset{k=1,...,|\mathcal{W}|}{\arg\max} \left(\boldsymbol{\theta}_k^T \mathbf{x} + \theta_{0k}\right)$$

- Decision surface(s):
  - $\rightarrow$ Each duple $(\boldsymbol{\theta}_k^T, \theta_{0k})$ defines a hyperplane

- Decision boundaries: Pairs $< i, j > \in \{1, \ldots, |\mathcal{W}|\}$ have equal scores:

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \Rightarrow (\boldsymbol{\theta}_i - \boldsymbol{\theta}_j)^T \mathbf{x} + (\theta_{0i} - \theta_{0j}) = 0$$

Decision frontiers



Introduction

- Consider a reference set of samples $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$

Introduction

- Consider a reference set of samples $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$
- **Goal**: Estimate weights $\boldsymbol{\theta}$ in discriminant function $g(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{x}$

Introduction

- Consider a reference set of samples $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$
- **Goal**: Estimate weights $\boldsymbol{\theta}$ in discriminant function $g(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{x}$
  $\rightarrow$ Minimize discrepancy between predictions and ground truth (ω) on $\mathcal{D}$:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}\left(g(\mathbf{x}_i; \boldsymbol{\theta}), \omega_i\right)$$

Introduction

- Consider a reference set of samples $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$
- **Goal**: Estimate weights $\boldsymbol{\theta}$ in discriminant function $g(\mathbf{x}; \boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{x}$
  $\rightarrow$ Minimize discrepancy between predictions and ground truth (ω) on $\mathcal{D}$:

$$\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathcal{L}\left(g(\mathbf{x}_i; \boldsymbol{\theta}), \omega_i\right)$$

- This minimization process may be solved analytically in some cases
  $\rightarrow$ Practical scenarios: iterative optimization process

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = \eta(0)/k$

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\boldsymbol{\nabla} J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - → Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined
  2. Vector $\boldsymbol{\theta}_0$ is arbitrarily chosen

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - → Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined
  2. Vector $\boldsymbol{\theta}_0$ is arbitrarily chosen
  3. Gradient vector $\nabla J\left(\boldsymbol{\theta}^{(0)}\right)$ is computed

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined
  2. Vector $\boldsymbol{\theta}_0$ is arbitrarily chosen
  3. Gradient vector $\nabla J\left(\boldsymbol{\theta}^{(0)}\right)$ is computed
  4. Value $\boldsymbol{\theta}^{(1)}$: moving $\propto \eta(k)$ from $\boldsymbol{\theta}^{(0)}$ in the steepest descent

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\nabla J\left(\boldsymbol{\theta}^{(k)}\right)$$

- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = \eta(0)/k$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined
  2. Vector $\boldsymbol{\theta}_0$ is arbitrarily chosen
  3. Gradient vector $\nabla J\left(\boldsymbol{\theta}^{(0)}\right)$ is computed
  4. Value $\boldsymbol{\theta}^{(1)}$: moving $\propto \eta(k)$ from $\boldsymbol{\theta}^{(0)}$ in the steepest descent
  5. Repeat from (3) until convergence criterion is achieved

Gradient descent

Gradient descent algorithm

$$\boldsymbol{\theta}_0 = \text{arbitrary}$$

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k)\boldsymbol{\nabla} J\left(\boldsymbol{\theta}^{(k)}\right)$$

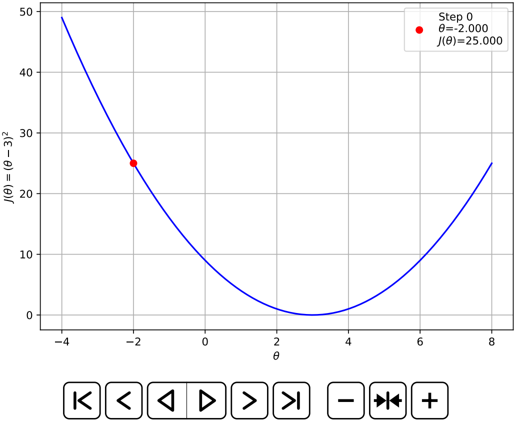
- Learning rate $\eta(k)$: controls the amount of change at each iteration
  - $\rightarrow$ Typical values: constant, $\eta(k) = {}^{\eta(0)}/_{k}$

- Process description:
  1. Criterion function $J(\boldsymbol{\theta})$ is defined
  2. Vector $\boldsymbol{\theta}_0$ is arbitrarily chosen
  3. Gradient vector $\boldsymbol{\nabla} J\left(\boldsymbol{\theta}^{(0)}\right)$ is computed
  4. Value $\boldsymbol{\theta}^{(1)}$: moving $\propto \eta(k)$ from $\boldsymbol{\theta}^{(0)}$ in the steepest descent
  5. Repeat from (3) until convergence criterion is achieved
     - $\rightarrow$ Example: $\left|\eta(k)\boldsymbol{\nabla} J\left(\boldsymbol{\theta}^{(k)}\right)\right| < \delta$

Gradient descent

Example:

- Function $J(\theta) = (\theta - 3)^2$
- Initial value $\theta_0 = -2$
- Learning rate $\eta(k) = 0.1$



Outline

Description

- Introduced by Rosenblatt in 1958

Description

- Introduced by Rosenblatt in 1958
- Practical linear binary classifier
  - $\rightarrow$ Linear model is wrapped in a step function $\Rightarrow$ activation

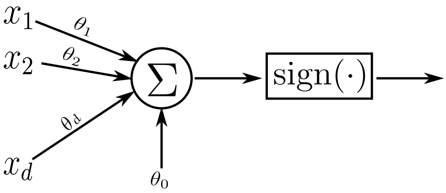
Description

- Introduced by Rosenblatt in 1958
- Practical linear binary classifier
  - $\rightarrow$ Linear model is wrapped in a step function $\Rightarrow$ activation
- Represents the foundation of neural models

Description

- Introduced by Rosenblatt in 1958
- Practical linear binary classifier
  - $\rightarrow$ Linear model is wrapped in a step function $\Rightarrow$ activation
- Represents the foundation of neural models

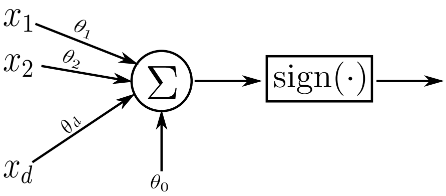
$$\hat{\omega} = \text{sign}(\boldsymbol{\theta}^T \mathbf{x} + \theta_0)$$



Description

- Introduced by Rosenblatt in 1958
- Practical linear binary classifier
  - $\rightarrow$ Linear model is wrapped in a step function $\Rightarrow$ activation
- Represents the foundation of neural models

$$\hat{\omega} = \text{sign}(\boldsymbol{\theta}^T \mathbf{x} + \theta_0) \qquad \text{sign}(\cdot) = \begin{cases} +1 \text{ if } \boldsymbol{\theta}^T \mathbf{x} + \theta_0 \geq 0 \\ -1 \text{ if } \boldsymbol{\theta}^T \mathbf{x} + \theta_0 < 0 \end{cases}$$



Context

- Assume binary scenario $\mathcal{W} = \{\omega_1, \omega_2\}$

Context

- Assume binary scenario $\mathcal{W} = \{\omega_1, \omega_2\}$
- Dataset $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\omega_i \in \mathcal{W}$ for $1 \leq i \leq |\mathcal{D}|$

Context

- Assume binary scenario $\mathcal{W} = \{\omega_1, \omega_2\}$
- Dataset $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\omega_i \in \mathcal{W}$ for $1 \leq i \leq |\mathcal{D}|$
- Vector of parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]$

Context

- Assume binary scenario $\mathcal{W} = \{\omega_1, \omega_2\}$
- Dataset $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\omega_i \in \mathcal{W}$ for $1 \leq i \leq |\mathcal{D}|$
- Vector of parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]$
- We may define vector $\mathbf{z} = [z_1, \ldots, z_{|\mathcal{D}|}]$ such that:

$$z_i = \begin{cases} +1 \text{ if } \omega_i = \omega_1 \\ -1 \text{ if } \omega_i = \omega_2 \end{cases}$$

Context

- Assume binary scenario $\mathcal{W} = \{\omega_1, \omega_2\}$
- Dataset $\mathcal{D} = \{(\mathbf{x}_i, \omega_i)\}_{i=1}^{|\mathcal{D}|}$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $\omega_i \in \mathcal{W}$ for $1 \leq i \leq |\mathcal{D}|$
- Vector of parameters $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]$
- We may define vector $\mathbf{z} = [z_1, \ldots, z_{|\mathcal{D}|}]$ such that:

$$z_i = \begin{cases} +1 \text{ if } \omega_i = \omega_1 \\ -1 \text{ if } \omega_i = \omega_2 \end{cases}$$

- This way, all samples in $\mathcal{D}$ will be correctly classified if and only if:

$$z_i \boldsymbol{\theta}^T \mathbf{x}_i > 0 \; \forall i$$

The Perceptron Criterion

How can we estimate $\boldsymbol{\theta}$?

The Perceptron Criterion

How can we estimate $\boldsymbol{\theta}$? Define $J(\boldsymbol{\theta})$

The Perceptron Criterion

How can we estimate $\boldsymbol{\theta}$? Define $J(\boldsymbol{\theta})$

- ✗ Number of misclassified samples $\Rightarrow$ unsuitable for gradient search

The Perceptron Criterion

How can we estimate $\boldsymbol{\theta}$? Define $J(\boldsymbol{\theta})$

- ✗ Number of misclassified samples $\Rightarrow$ unsuitable for gradient search
- ✓ Sum of the distances to surface $\mathcal{H}$ for misclassified samples:

$$J_p(\boldsymbol{\theta}) = \sum_{i:z_i\boldsymbol{\theta}^T\mathbf{x}_i \leq 0} (-z_i\boldsymbol{\theta}^T\mathbf{x}_i)$$

The Perceptron Criterion

How can we estimate $\boldsymbol{\theta}$? Define $J(\boldsymbol{\theta})$

- ✗ Number of misclassified samples $\Rightarrow$ unsuitable for gradient search
- ✓ Sum of the distances to surface $\mathcal{H}$ for misclassified samples:

$$J_p(\boldsymbol{\theta}) = \sum_{i:z_i\boldsymbol{\theta}^T\mathbf{x}_i\leq 0} (-z_i\boldsymbol{\theta}^T\mathbf{x}_i)$$

$\rightarrow$ The corresponding gradient is:

$$\nabla J_p(\boldsymbol{\theta}) = \sum_{i:z_i\boldsymbol{\theta}^T\mathbf{x}_i\leq 0} (-z_i\mathbf{x}_i)$$

The Perceptron Criterion

Batch Perceptron algorithm

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \cdot \sum_{i:z_i\boldsymbol{\theta}^T\mathbf{x}_i \leq 0} (-z_i\mathbf{x}_i)$$

The Perceptron Criterion

Batch Perceptron algorithm

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \cdot \sum_{i:z_i \boldsymbol{\theta}^T \mathbf{x}_i \leq 0} (-z_i \mathbf{x}_i)$$

- Instead of batches, correction may be done on a singe-sample policy:

The Perceptron Criterion

Batch Perceptron algorithm

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \cdot \sum_{i: z_i \boldsymbol{\theta}^T \mathbf{x}_i \leq 0} (-z_i \mathbf{x}_i)$$

- Instead of batches, correction may be done on a singe-sample policy:

$\rightarrow$ $$J_p(\boldsymbol{\theta}, \mathbf{x}_i) = \begin{cases} -z_i \boldsymbol{\theta}^T \mathbf{x}_i & \text{if } z_i \boldsymbol{\theta}^T \mathbf{x}_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

The Perceptron Criterion

Batch Perceptron algorithm

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta(k) \cdot \sum_{i:z_i\boldsymbol{\theta}^T\mathbf{x}_i \leq 0} (-z_i\mathbf{x}_i)$$

- Instead of batches, correction may be done on a singe-sample policy:

$\rightarrow$ $$J_p(\boldsymbol{\theta}, \mathbf{x}_i) = \begin{cases} -z_i\boldsymbol{\theta}^T\mathbf{x}_i & \text{if } z_i\boldsymbol{\theta}^T\mathbf{x}_i \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

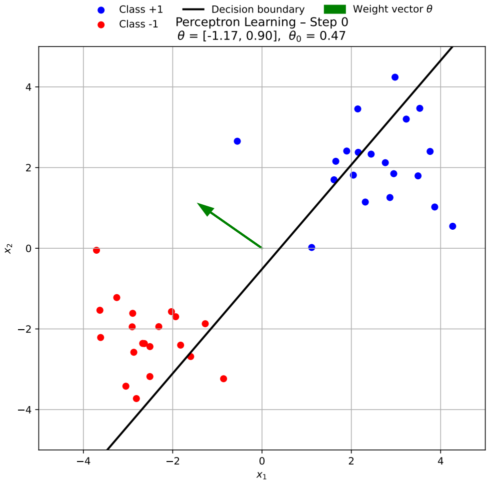
Single-sample Perceptron algorithm

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + \eta(k) z_i\mathbf{x}_i$$

The Perceptron Criterion

Approach	Update frequency	Pros	Cons
Single-sample	Every sample	Fast reaction to new data stochasticity	Noisy updates
Batch	Accumulation	Smoother Stable convergence	Slower

The Perceptron Criterion

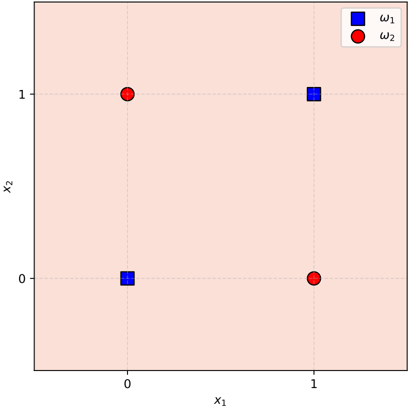


The XOR problem

The *Exclusive OR* (XOR) operation $\Rightarrow$ binary classification task:

- Feature space: $\mathbf{x} \in \{0,1\}^2$
- Label space: $\mathcal{W} = \{0,1\}$

- Function $f_{xor}(\mathbf{x}) = \begin{cases} \omega_1 \text{ if } x_1 = x_2 \\ \omega_2 \text{ if } x_1 \neq x_2 \end{cases}$



The XOR problem

- Showcases the main limitation of the percepton mechanism

- Unable to address non-linearly separable labels
  - → Need for non-linearities in the scheme

- Two related mechanisms:
  - → Stacking perceptrons into layers
  - → Using non-linear activation functions