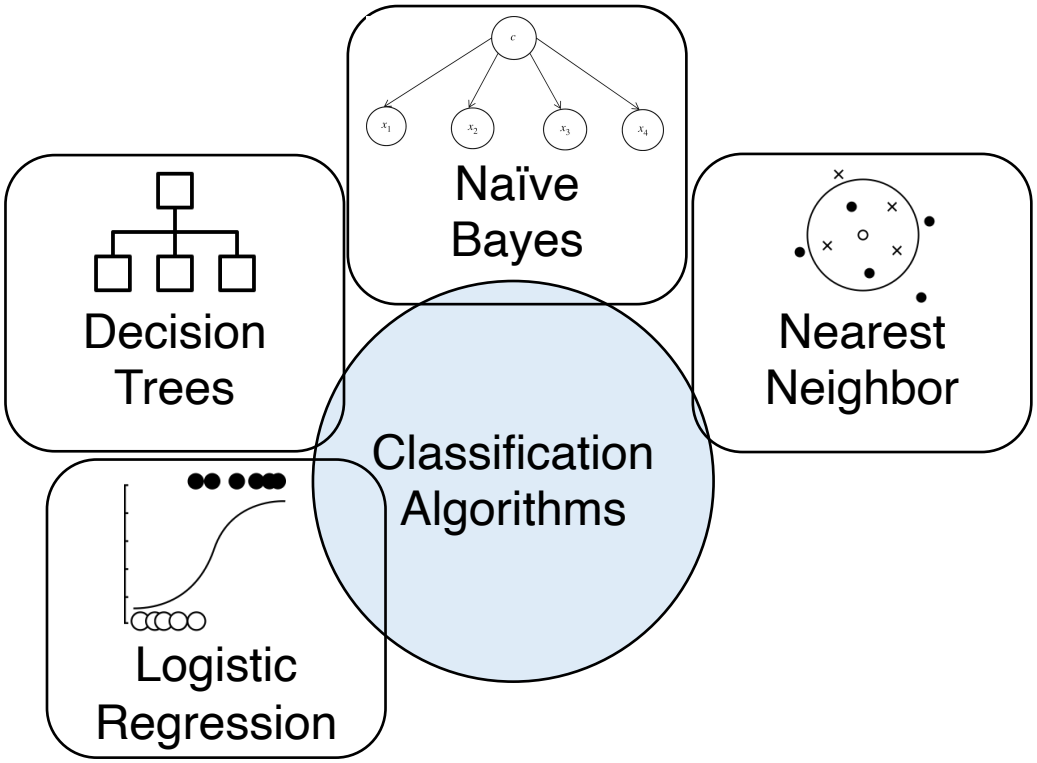
教材《机器学习与大语言模型》配套电子教案

机器学习

第七讲: 支持向量机

顾小东
上海交通大学计算机学院

分类器家族

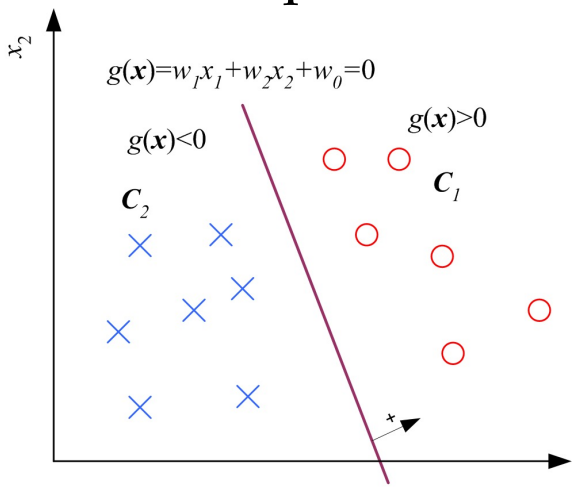

Naïve
Bayes
Decision
Trees
Nearest
Neighbor
Classification
Algorithms
Logistic
Regression

Recall: Linear Classifiers

- **Given**: a training set $\{(x^{(1)},y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$ of N samples

  $x^{(\ell)} \in \mathbf{R}^d$: input, $y^{(\ell)} \in \{-1, 1\}$:target (label)

- Suppose the data is linearly separable: there exists a linear surface to separate the two classes



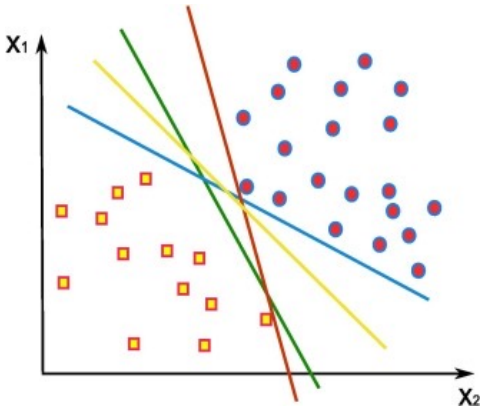
2-D: line;
3-D: plane;
…;
d-D: hyperplane ($\boldsymbol{wx}+w_0=0$)

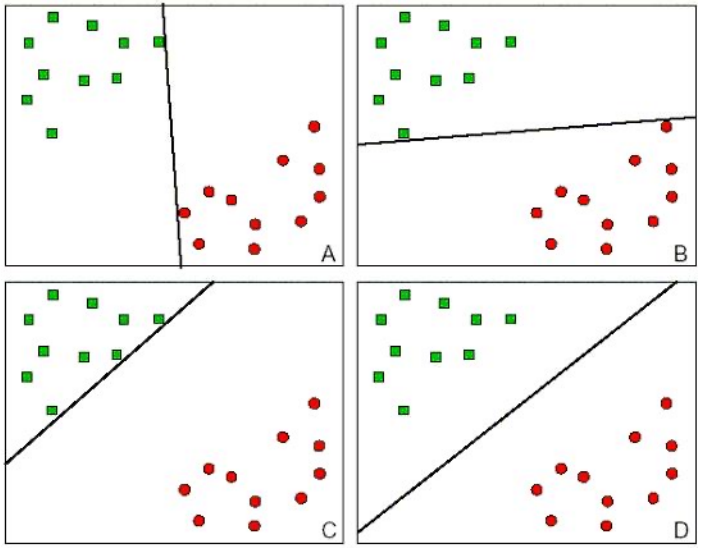
Goal:

Find $\boldsymbol{wx}+w_0=0$ that perfectly separates the two classes

The Problem

- There can be multiple hyperplanes to separate the data.

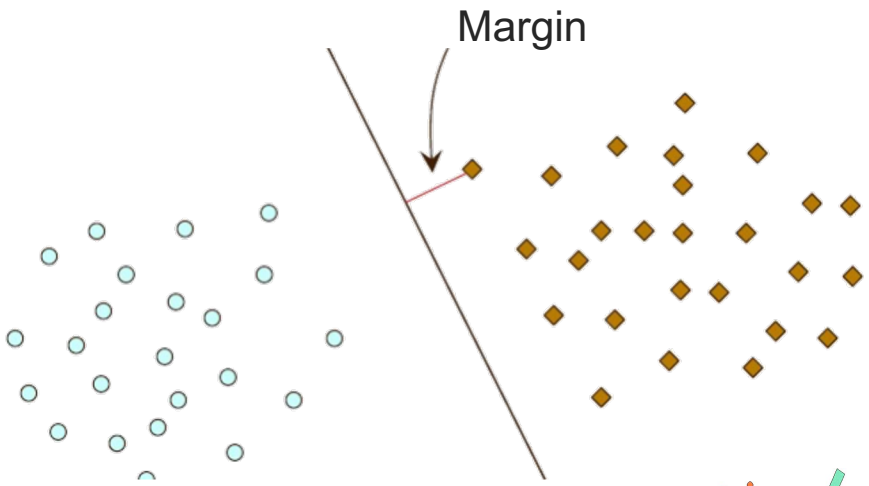




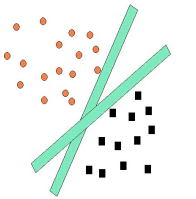
Which is the best?

The Idea:

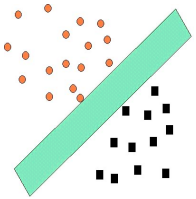
- Find a decision boundary that **maximizes the margin** between two classes.



Margin and generalization:
Statistical learning theories have shown that the boundary with the largest margin generalizes best (i.e., has the smallest generalization error).



skinny margin is more flexible, thus more complex

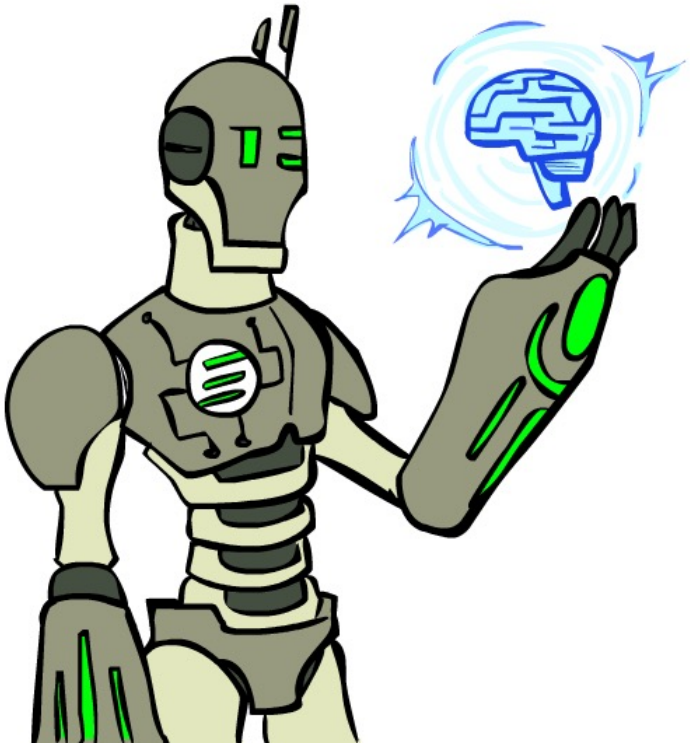


fat margin is less complex

Today

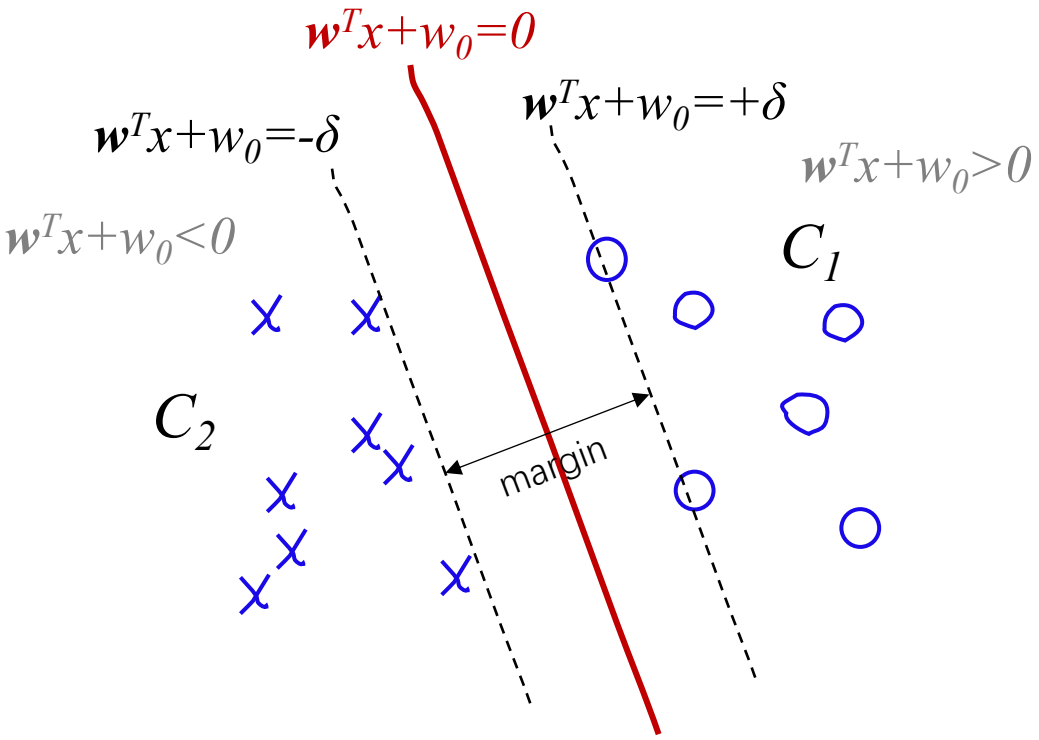
Support Vector Machine

- Problem Formulation
- Loss and Optimization



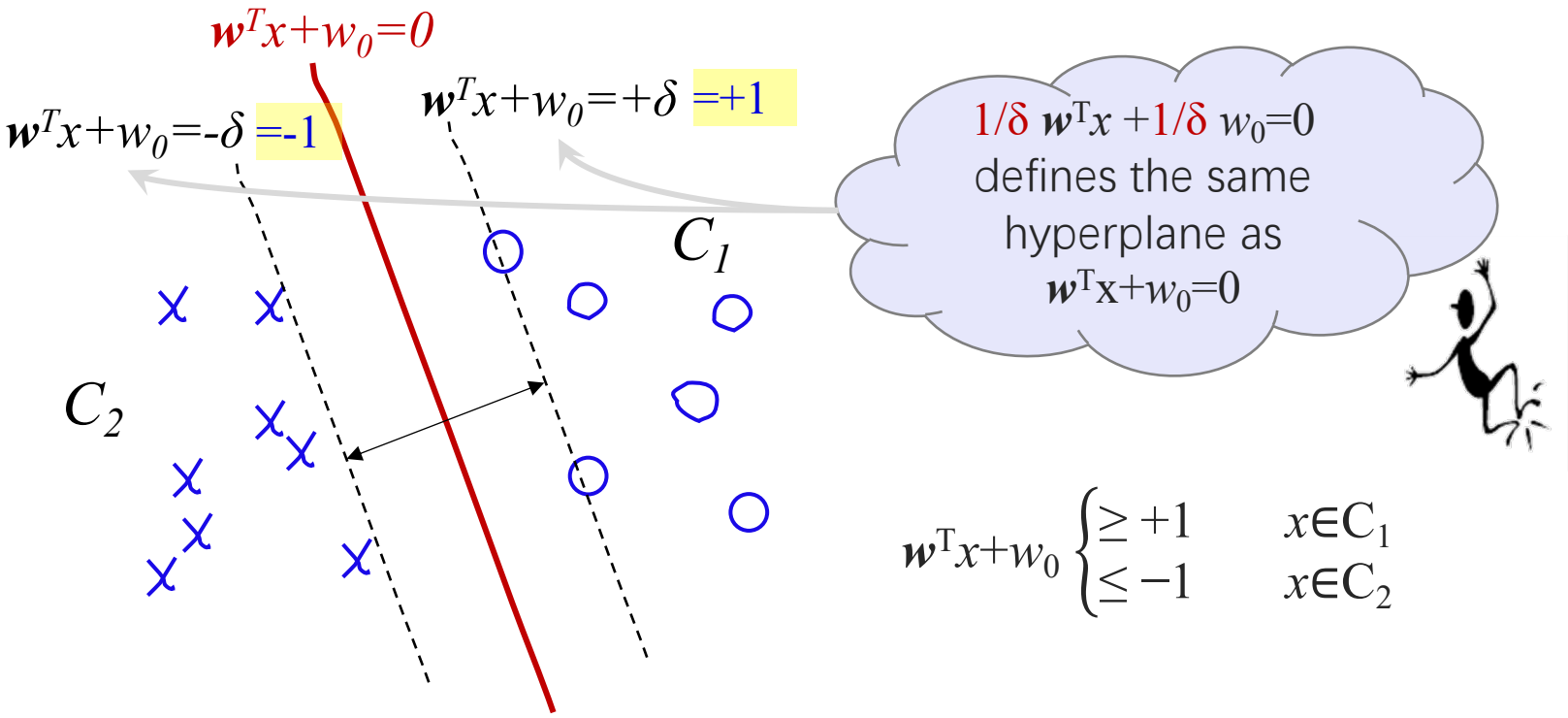
Problem Formulation

- A linear discriminant function models a **linear decision boundary** of two classes.



Problem Formulation

- A linear discriminant function models a **linear decision boundary** of two classes.



Maximizing the Margin

$$\boldsymbol{w}^{\mathrm{T}}x+w_0 = \begin{cases} 1 & \text{for the closest points on one side} \\ -1 & \text{for the closest points on the other} \end{cases}$$

- Let $x^{(1)}$ and $x^{(2)}$ be two closest points on each side of the hyperplane.
- Note that

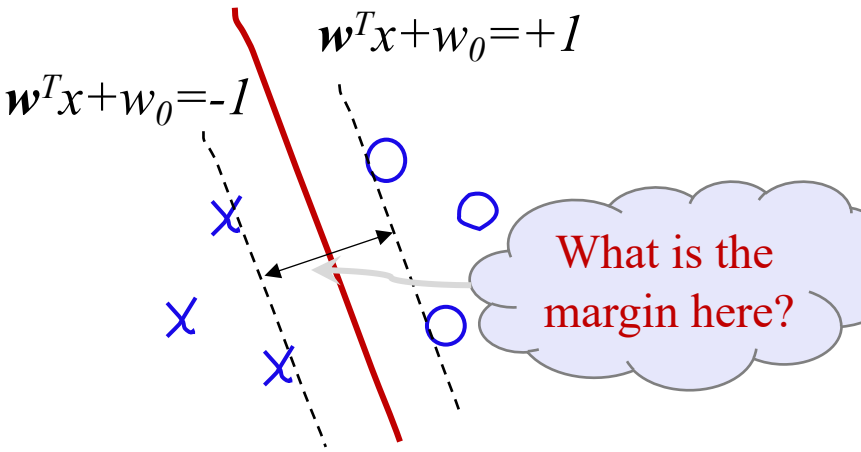
$$\boldsymbol{w}^{\mathrm{T}}\, x^{(1)} + w_0 = +1$$
$$\boldsymbol{w}^{\mathrm{T}}\, x^{(2)} + w_0 = -1$$

which imply

$$\boldsymbol{w}^{\mathrm{T}}(x^{(1)} - x^{(2)}) = 2.$$

Hence, the margin can be given by

$$\text{margin} = \frac{\boldsymbol{w}^{\mathrm{T}}(x^{(1)} - x^{(2)})}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|}$$

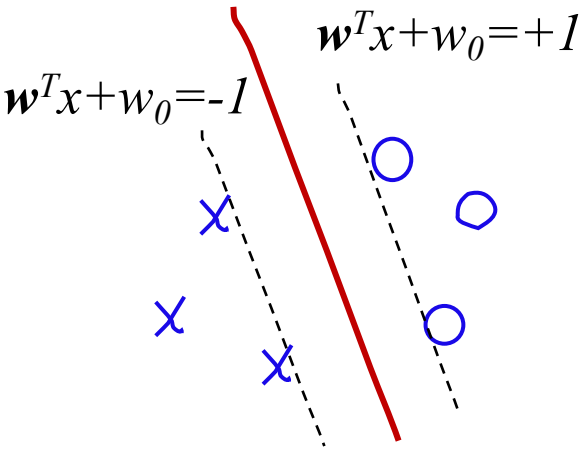


Maximizing the margin is equivalent to minimizing $\frac{1}{2}\|\boldsymbol{w}\|$

Inequality Constraints

- Given: $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$, we want $\boldsymbol{w}$ and w_0 to satisfy

$$\boldsymbol{w}^{\mathrm{T}} x^{(\ell)} + w_0 \begin{cases} \geq +1 & \text{if } y^{(\ell)} = +1 \\ \leq -1 & \text{if } y^{(\ell)} = -1 \end{cases}$$



- Or, equivalently,

$$y^{(\ell)} (\boldsymbol{w}^{\mathrm{T}} x^{(\ell)} + w_0) \geq 1$$

SVM = Solving an Optimization Problem

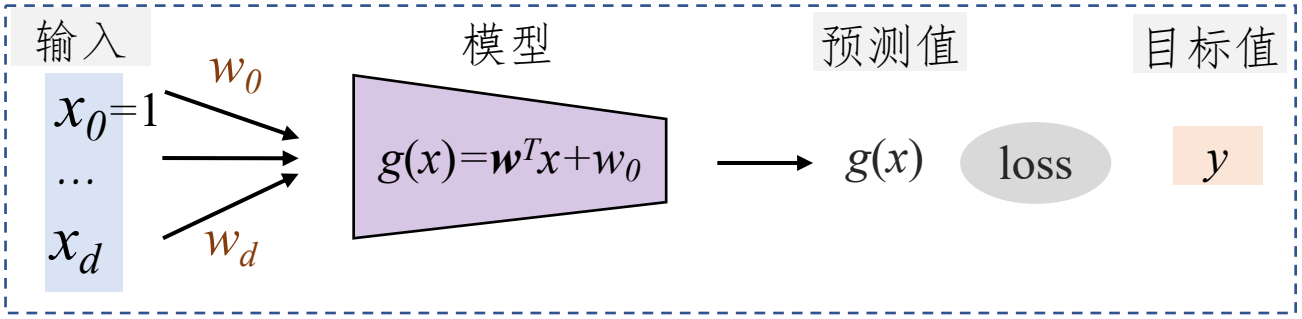
In summary, SVM aims to solve a constrained optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{subject to} && y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0)\geq 1,\ \ell=1,\ldots,\mathrm{N} \end{aligned}$$

- This is a quadratic programming (QP) problem, which is one type of convex optimization problem.
- The complexity depends on the dimensionality d of inputs

Model Architecture

- The same as Perceptron.



- Train:
  - optimize the parameters $\boldsymbol{w}$ and w_0 using data
- Test:
  - for a new input x, calculate $g(x) = \mathbf{w}^T x+w_0$ and choose C_1 if $g(x)$ > 1 otherwise C_2.

Loss Function

$$\min \ \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$\text{s.t.} \ \ y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0)\geq 1,\ \ell=1,\ldots,N$$

- For a given input x, the model outputs a score $g(x) = \boldsymbol{w}^{\mathrm{T}}x+w_0$. Let $y \in \{-1, +1\}$ be the label of the real class ($y = +1$: $x \in C_1$, $y = -1$: $x \in C_2$):
  - if $y(\boldsymbol{w}^{\mathrm{T}}x+w_0) < 1$: we aim to maximize $y(\boldsymbol{w}^{\mathrm{T}}x+w_0)$ until reaching 1, cost is $1-y(\boldsymbol{w}^{\mathrm{T}}x+w_0)$
  - if $y(\boldsymbol{w}^{\mathrm{T}}x+w_0) > 1$: outlier points, no need to optimize, cost is **0**
- Can writhe this succinctly as a **Hinge** loss:

$$\ell\,(\boldsymbol{w}, w_0 \,|\, x, y) = \max(0,\ 1-y(\boldsymbol{w}^{\mathrm{T}}x+w_0))$$

- Given: $D = \{(x^{(1)}, r^{(1)}), \ldots, (x^{(N)}, r^{(N)})\}$, the loss over the dataset is defined as:

$$L\,(\boldsymbol{w}, w_0 \,|\, D) = \frac{1}{N}\sum_{\ell=1}^{N} \max(0,\ 1-y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0)) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

Optimization – Gradient Descend

$$\boldsymbol{w}_{\text{new}} = \boldsymbol{w}_{\text{old}} - \eta \frac{\partial L}{\partial w}$$

What is $\frac{\partial L}{\partial w}$?

$$L(\boldsymbol{w}, w_0 \mid D) = \begin{cases} \frac{\lambda}{2}\|\boldsymbol{w}\|^2 & \text{if } y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0) \geq 1 \\ \frac{1}{N}\sum_{\ell=1}^{N} 1 - y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2 & \text{otherwise} \end{cases}$$

For each $\boldsymbol{w}_j$ (j =0,…,d):

$$\frac{\partial L}{\partial w_j} = \begin{cases} \lambda w_j & \text{if } y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0) \geq 1 \\ \lambda w_j - y^{(\ell)}x_j^{(\ell)} & \text{o.w.} \end{cases}$$

$$\frac{\partial L}{\partial w_0} = \begin{cases} 0 & \text{if } y^{(\ell)}(\boldsymbol{w}^{\mathrm{T}}x^{(\ell)}+w_0) \geq 1 \\ -y^{(\ell)} & \text{o.w.} \end{cases}$$

The Algorithm

Gradient Descend for SVM

Input: $D = \{(x^{(\ell)}, y^{(\ell)})\}$ (ℓ=1:N)
for $j = 0,\ldots,d$
  $\boldsymbol{w}_j \leftarrow rand\ (-0.01, 0.01)$
repeat
  for $\ell = 1,\ldots,N$
    $a \leftarrow 0$
    for $j = 0, \ldots, d$
      $a \leftarrow a + \boldsymbol{w}_j x_j^{(\ell)}$
    for $j = 0, \ldots, d$
      $\Delta \boldsymbol{w}_j \leftarrow \lambda \boldsymbol{w}_j$
    $\Delta \boldsymbol{w}_0 \leftarrow 0$
    if $y^{(\ell)} a < 1$:
      for $j = 0, \ldots, d$
        $\Delta \boldsymbol{w}_j \leftarrow \Delta \boldsymbol{w}_j - y^{(\ell)} x_j^{(\ell)}$
      $\Delta \boldsymbol{w}_0 \leftarrow \Delta \boldsymbol{w}_0 - y^{(\ell)}$
  for $j = 0,\ldots,d$
    $w_j \leftarrow w_j - \eta \Delta w_j$
until convergence

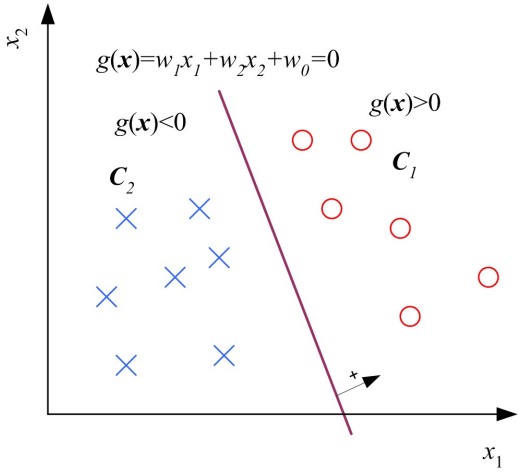
Programming Time

Tutorial: SVM from scratch with explanation (Python)

https://towardsdatascience.com/implementing-svm-from-scratch-784e4ad0bc6a

https://www.kaggle.com/code/misbahbilgili/svm-from-scratch-with-explanation/notebook

What's Next

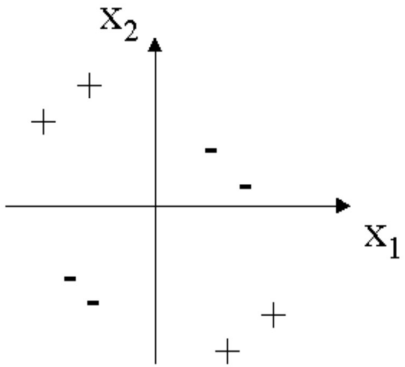


Perceprton

Logistic Regression

SVM

The curse of nonlinearity



I can approximate any non-linear functions !

Neural Networks