Support Vector Machines, Kernel Trick

Machine Learning 1 UE (INP.33761UF)

Thomas Wedenig

June 05, 2024

Institute of Theoretical Computer Science
Graz University of Technology, Austria

Table of contents

Organization

- **No in-person session next week !**
  - Appointment is cancelled in TUGonline
- Recording will be uploaded on TUbe

Support Vector Machines

- **Binary Classification:** $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with $y^{(i)} \in \{-1, 1\}$

- **Binary Classification:** $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with $y^{(i)} \in \{-1, 1\}$

Support Vector Machine (SVM)

- SVMs are **linear models**:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & \text{else} \end{cases}$$

- Usually used for **classification**
  - We will discuss binary classification
  - Multiclass classification is a **straightforward extension**
  - Can also be used for **regression** (not discussed)

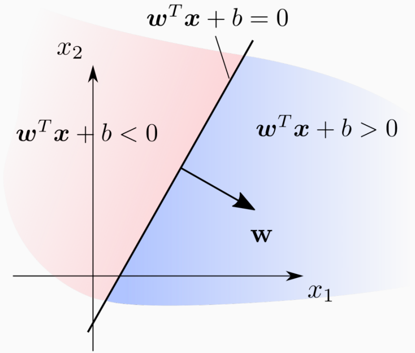
- **Binary Classification:** $\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ with $y^{(i)} \in \{-1, 1\}$

Support Vector Machine (SVM)

- SVMs are **linear models**:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} + b \geq 0 \\ -1 & \text{else} \end{cases}$$

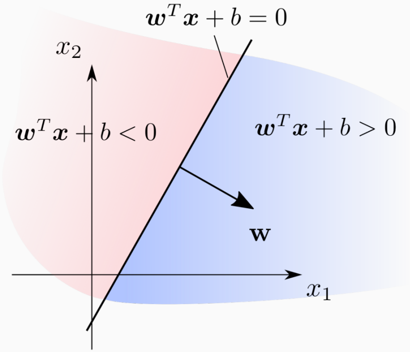
- Usually used for **classification**
  - We will discuss binary classification
  - Multiclass classification is a **straightforward extension**
  - Can also be used for **regression** (not discussed)



- Decision boundary is a **hyperplane**:

$$\mathcal{H} = \{\mathbf{x} : \mathbf{w}^T\mathbf{x} + b = 0\}$$

- Partitions space into two **halfspaces**



- We already know a **binary classifier** with **linear decision boundary** 🤔

Other Linear Classifiers

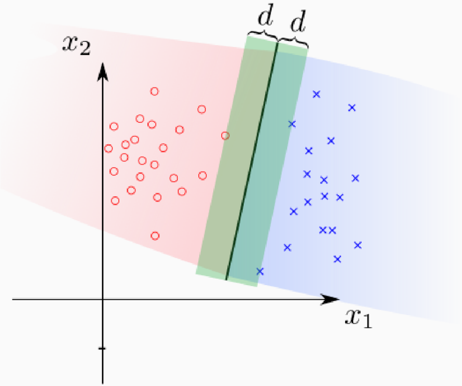
- We already know a **binary classifier** with **linear decision boundary** 🤔
- **Logistic Regression:** $p(y \mid \mathbf{x}) = \sigma(\boldsymbol{\theta}^T \mathbf{x} + \theta_0)$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } p(y \mid \mathbf{x}) \geq 0.5 \\ -1 & \text{else} \end{cases} = \begin{cases} 1 & \text{if } \boldsymbol{\theta}^T \mathbf{x} + \theta_0 \geq 0 \\ -1 & \text{else} \end{cases}$$

- **However,** SVMs are **trained differently** ❗

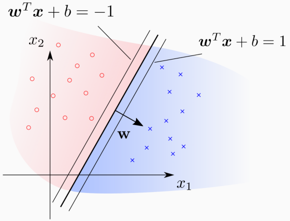
- Logistic Regression is trained with **maximum likelihood** (binary cross-entropy)
  - Will be discussed next week
- SVMs try to **maximize the "classification margin"**

- **Classification Margin:** minimal distance to any training example



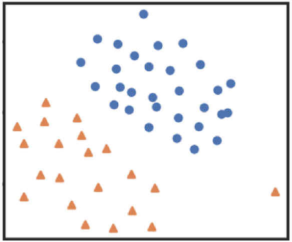
- We can achieve this by finding

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 \qquad \text{s.t.} \quad y^{(i)}\left(\mathbf{w}^T\mathbf{x}^{(i)}\right) \geq 1 \quad \text{for all } i \in \{1,\ldots,N\}$$

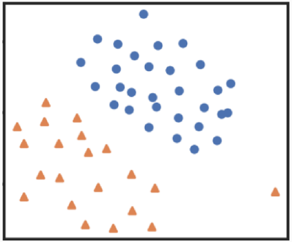


- **Note:** This assumes dataset is **linearly separable** ❗
- **Convex** optimization problem
- **Specialized, efficient optimizers** available
- Next assignment: solve a different formulation of this objective with **gradient descent**

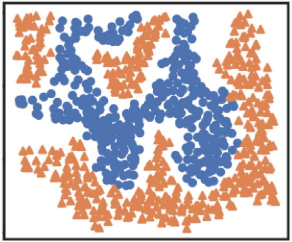
- What if the dataset is **not linearly separable** ?
- We can still fit a linear SVM using the **soft-margin extension**
  - By introducing **slack** variables



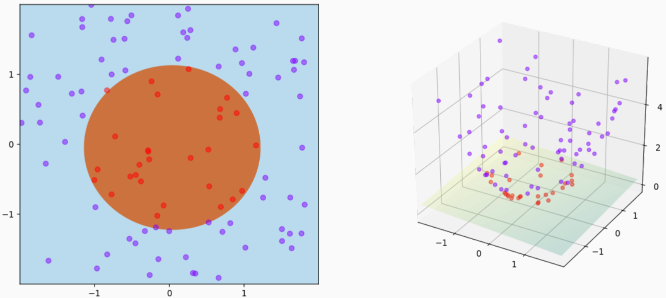
- What if the dataset is **not linearly separable** ❓
- We can still fit a linear SVM using the **soft-margin extension**
  - By introducing **slack** variables



- What if the **dataset is highly non-linear?** 🤔



- **Non-linear** feature transforms $\phi(\mathbf{x})$ to the rescue **!**
- Even if data is **not linearly separable** in the **original space**, it **is** linearly separable in a higher dimensional space with high probability



https://en.wikipedia.org/wiki/File:Kernel_trick_idea.svg

Kernel Trick

- Kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$
- k is **positive definite** iff $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$ for some ϕ
- **Implicitly computes similarities in a transformed space** !

- Kernel $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$
- k is **positive definite** iff $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T \phi(\mathbf{x}^{(j)})$ for some ϕ
- **Implicitly computes similarities in a transformed space** !
- **Polynomial Kernel** corresponds to ϕ computing all polynomials up to degree p:

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \left(1 + \mathbf{x}^{(i)^T} \mathbf{x}^{(j)}\right)^p$$

- **Gaussian Kernel** corresponds to ϕ transforming $\mathbf{x}$ to an **infinite dimensional** space:

$$k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \exp\left(-\gamma \|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2\right)$$

- The **dual problem** to the soft-margin SVM (primal problem) when transforming the input using a feature transform ϕ:

$$\max_{\boldsymbol{\lambda}} \; -\frac{1}{2}\sum_{i,j}\lambda^{(i)}\lambda^{(j)}y^{(i)}y^{(j)}\phi(\mathbf{x}^{(i)})^T\phi(\mathbf{x}^{(j)}) + \sum_i \lambda^{(i)}$$

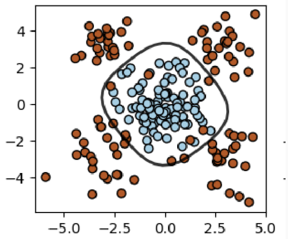
- Subject to constraints $\sum_i \lambda^{(i)}y^{(i)} = 0$ and $0 \leq \lambda^{(i)} \leq C$ for all $i \in \{1, \ldots, N\}$

- **Only depends** on pairwise "similarities" $\phi(\mathbf{x}^{(i)})^T\phi(\mathbf{x}^{(j)})$
- Replace with **kernel** $k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = \phi(\mathbf{x}^{(i)})^T\phi(\mathbf{x}^{(j)})$

- We **never need to even know ϕ explicitly** ❗
- Classification for a new **x**:

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_i \lambda^{(i)*} y^{(i)} k(\mathbf{x}^{(i)}, \mathbf{x}) - b^*\right)$$

- Now we can fit **non-linear** decision boundaries 😍
  - i.e., **non-linear in the original space**



(Kernelized) SVM – Summary

Pros 👍

- Nice thoery
- Finding the global optimum is guaranteed
- Maximizing margin can avoid overfitting
- Kernelized SVMs can solve non-linear problems

Cons 👎

- Fit time scales at least quadratically in the number of samples
- Not trained with a probabilistic objective
  - No sensible “probabilistic interpretation” of output

SVM Demo

Assignment 3 Handout