Kernels in machine learning

Optimization perspective and applications in autonomous driving

Dariusz Lasecki

Separating hyperplane

A hyperplane is given by the equation

$$\mathbf{w}^T\mathbf{x} + b = 0. \tag{1}$$

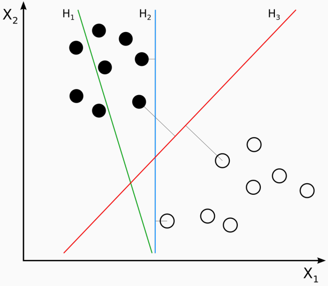


Figure 1: Separating and non-separating hyperplanes.

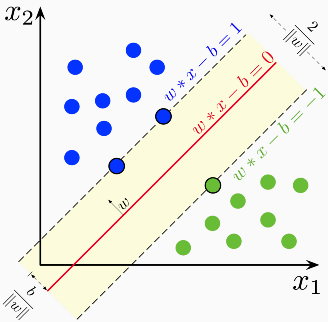
SVM as an optimization problem

SVM is a quadratic constrained optimization problem.

Given a set of points $\{\mathbf{x}_i, y_i\}$, where $y_i \in \{-1, 1\}$, the optimal separating hyperplane $(\mathbf{w}^*, b^*)$ equals the optimal solution of the following minimization problem

$$\min_{\mathbf{w},b} ||\mathbf{w}||^2$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n.$$



Linearly non-separable data

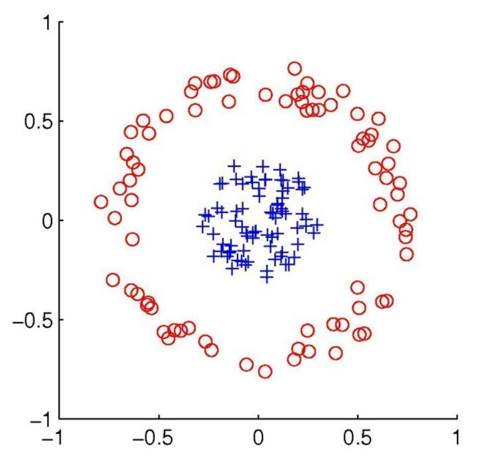


Figure 3: Linearly non-separable data.

Linearly non-separable data - the way out

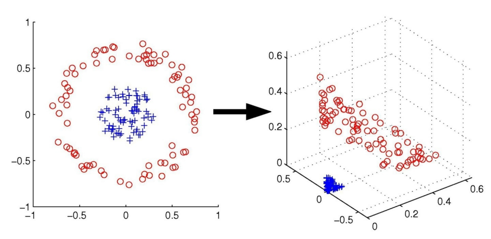


Figure 4: Linearly non-separable data - the way out.

$$\Phi\left((x_1, x_2)\right) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

SVM as an optimization problem with Φ

$$\min_{\mathbf{w},b} ||\mathbf{w}||^2$$

$$y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, \dots, n.$$

To derive the dual problem, we use the method of Lagrange multipliers

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\mathbf{w}||^2 - \sum_i \alpha_i \left[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \right].$$

The related optimization problem is then:

$$\min_{\mathbf{w},b} \max_{\boldsymbol{\alpha} \geq 0} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}).$$

Dual SVM

The dual form of SVM quadratic optimization is

$$\max_{\boldsymbol{\alpha}} \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

$$\sum_i \alpha_i y_i = 0, \quad i = 1, \ldots, n$$

The dual program does not depend on $\mathbf{w}$! It only depends on input data.

Dual SVM with Φ

$$\max_{\boldsymbol{\alpha}} \sum_{i=1} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j \Phi(\mathrm{x}_i)^T \Phi(\mathrm{x}_j)$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, n$$

$$\sum_{i} \alpha_i y_i = 0, \quad i = 1, \ldots, n$$

For $\Phi\left((x_1, x_2)\right) = (x_1^2, x_2^2, \sqrt{2}x_1 x_2)$, we have

$$\Phi(\mathrm{x}_i)^T \Phi(\mathrm{x}_j) = x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}.$$

The computational cost is the cost of mapping and the cost of an inner product. But notice that

$$(\mathrm{x}_i^T \mathrm{x}_j)^2 = x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2}.$$

The computational cost is only the cost of an inner product!

Kernels

$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^2$ is an example of a **kernel**, more specifically a polynomial kernel.

We might stop thinking at the level of Φ, we might just define kernels as atomic objects $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$. This is is called the **kernel trick**.

$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ is a valid kernel iff $\left[K_{ij}\right]$ is psd (Mercer's condition).

Common kernels include:

- polynomial kernel

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + c)^d, \tag{2}$$

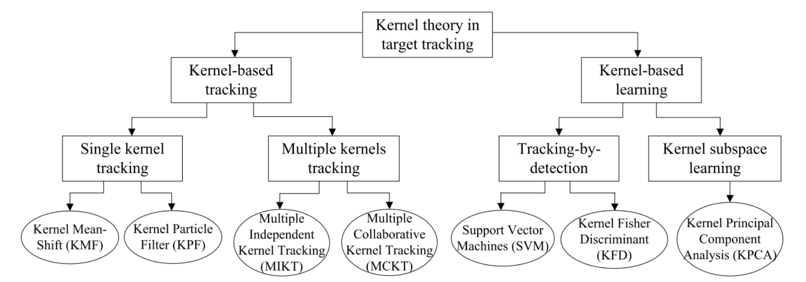
- Gaussian kernel

$$\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right). \tag{3}$$

Kernels - conclusions

- Kernels are an implicit method of representing our data in a higher dimensional space (the kernel trick).
- Computational cost of calculating a kernel does not depend on the dimension of that space (it can also be infinite).
- Kernels allow us to use **linear** ML methods for solving **non-linear** boundary problems.
- Other kernelizable techniques include: Gaussian processes, principal components analysis, ridge regression, linear adaptive filters and many others...
- It is often not clear upfront which kernel to use. Kernels are often chosen experimentally.

Applications in AD



- Kernel-based online subspace learning for target tracking (good trade-off between the stability and real-time processing). [1]

Questions?

References i

Y. W. J. Y. Y. D. Z. Hu.
Review on kernel based target tracking for autonomous driving.
Journal of Information Processing, 24, 2016.