Support Vector Machines

Maria Kesa

14.05.2015

Table of contents

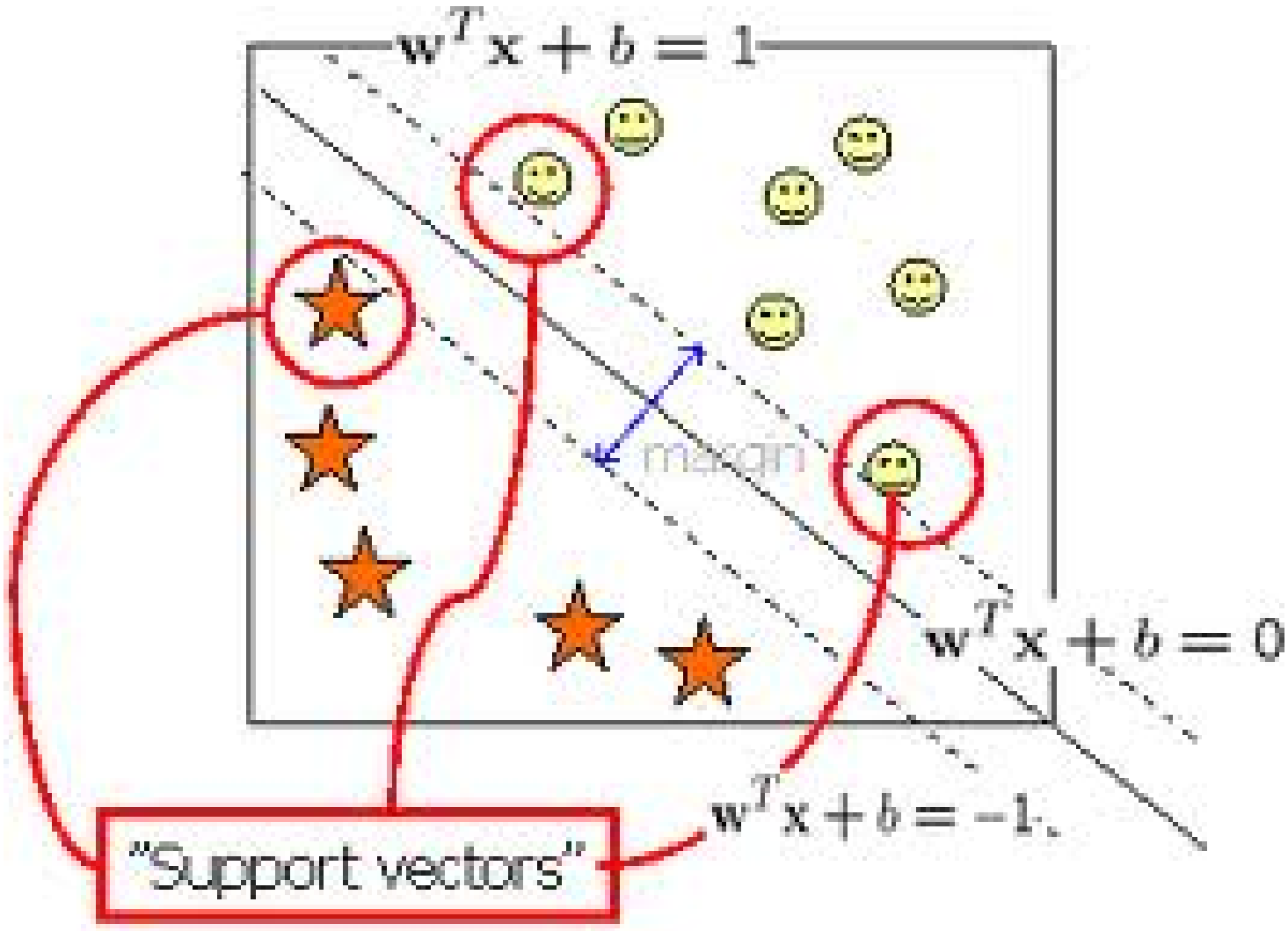
- Problem Statement
- Illustration
- How to define an optimal decision boundary?
- Equation for the decision boundary
- Finding the optimal decision boundary
- Optimizing the Cost Function
- Lagrangian for Support Vector Machines
- Alternative optimization problem (Dual)
- Sparsity in computation
- Non-linear classification
- Kernels
- Examples from our work

Problem statement

Support Vector Machines is an algorithm for classification.

It finds an optimal decision boundary to separate points in high-dimensional spaces.

Illustration



How do we define an optimal decision boundary?

A margin is the distance from the decision boundary to the nearest data point–support vector– of any class.

We say that a hyperplane is statistically an optimal separator when it maximizes the margin.
Why does this make sense?

Equation for the decision boundary

The decision boundary is simply a line or hyperplane and is therefore described by a linear equation.
The output of the classifier
This is the equation for the line:

$$w^T \cdot x + b = 0$$

This is the equation for the line touching the support vector belonging to the positive class:

$$w^T \cdot x + b = 1$$

This is the equation for the line touching the support vector belonging to the negative class:

$$w^T \cdot x + b = -1$$

Finding the optimal decision boundary

We now define a cost function that finds the decision boundary with the maximal margin.

$$min_{w,b} w \cdot w^T + C \sum_i \xi_i$$

Subject to the constraints:

$$\xi_i \geq 1 - y_i(w^T \cdot x_i + b)$$

$$\xi_i \geq 0$$

$$\forall i = 1, ..., n$$

y_i is the class label $-1, 1$
w is the weight of the decision boundary, b is the bias
ξ_i is the slack variable that relaxes inequality constraints to accomodate data points that are not linearly separable
C is the misclassification penalty

Optimizing the cost function

We can solve the minimization problem on the previous slide by maximizing the corresponding Lagrangian.
Lagrangians attach so called "Lagrange Multipliers" to each one of the constraints. The idea is to make the problem easier to solve by not enforcing the constraints strictly, but instead imposing a "cost" on each one of the constraints. The scheme allows to find the optimal values by optimizing the cost function and the constraints jointly.
For a problem in two dimensions the corresponding equations look like this:
Minimize $F(x, y)$ subject to a constraint $G(x, y)$, with $G(x, y) = 0$
To solve the problem we introduce an auxiliary variable λ that we call the Lagrange multiplier and add the constraint to the objective in the following manner:

$$L(x, y, \lambda) = F(x, y) + \lambda \cdot G(x, y)$$

Lagrangian for Support Vector Machines

For Support Vector Machines the minimization problem becomes a maximization problem of the Lagrangian:

$$max_{w,b,\lambda,\alpha} L(w, b, \lambda, \alpha) = \frac{1}{2} w^T w + C \sum_i \xi_i + \sum_i \alpha_i (1 - y_i (w^T \cdot x_i + b) - \xi_i) +$$

We find the optimal solution by forcing the partial derivatives of the Lagrangian to vanish:

$$\frac{\partial L}{\partial w} = w - \sum_i y_i \alpha_i x_i = 0$$

$$\frac{\partial L}{\partial b} = -\sum_i y_i \alpha_i = 0$$

From the equations from partial derivatives we can see that we can calculate the weights in the following way:

$$w = \sum_i y_i \alpha_i x_i$$

We can see that the solution can be expressed as a linear combination of the training vectors

Lagrangian for Support Vector Machines

Alternatively we can use quadratic programming to maximize this equation:

$$w(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \alpha_i \alpha_j y_i y_j x_i x_j$$

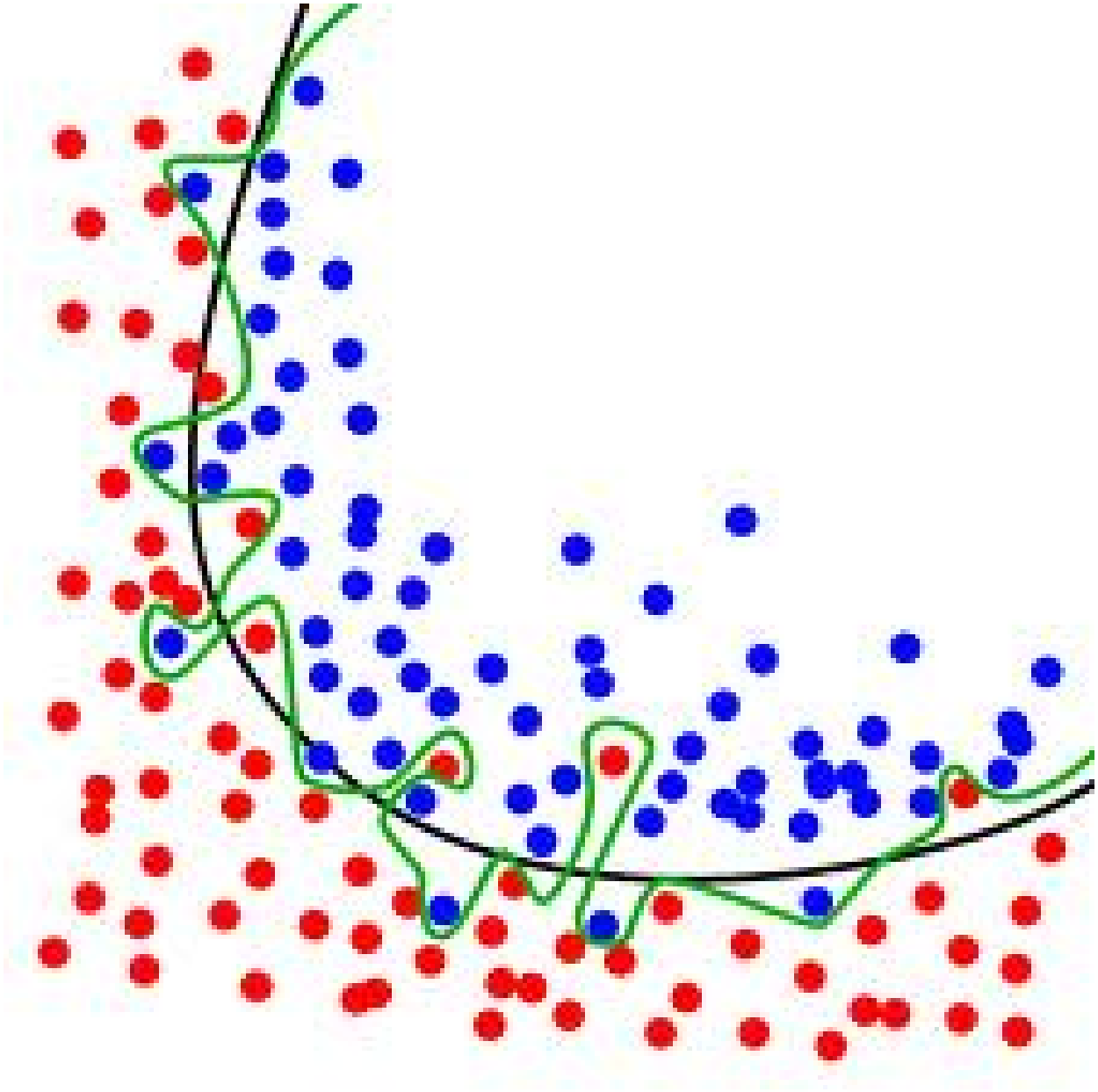
with the constraints $\alpha_i \geq 0$, $\sum_i \alpha_i y_i = 0$
Once we maximize this equation we can again recover w via the following expression:

$$w = \sum_i y_i \alpha_i x_i$$

Sparsity in computation

Most α_i are 0. It turns out that the α_i's that matter in the solution correspond to the support vectors, the data points that are closest to the margin.
This is why the algorithm is called "Support Vector Machine"– it uses only a small number of data points to find the optimal decision boundary.

Non-linear classification



Kernels

The invention of kernels allowed to apply the techniques of finding linear patterns (e.g. linear regression, finding linear hyperplanes to separate data points) to non-linear patterns in data.

The idea is to project the observed features into a space of higher dimensionality, where they start to exhibit linear structure– for example, become linearly separable (important for classification).

This mapping is done through a function $\phi(\mathbf{x})$ that projects the data vector into a higher dimensional space.

Kernel

A kernel is a function κ that for all $\mathbf{x}, \mathbf{z} \in \mathbf{X}$ satisfies

$$\kappa(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle$$

Kernels permit us to operate on the inner products of the data points, without explicitly computing on the coordinates.

Optimization problem using kernels

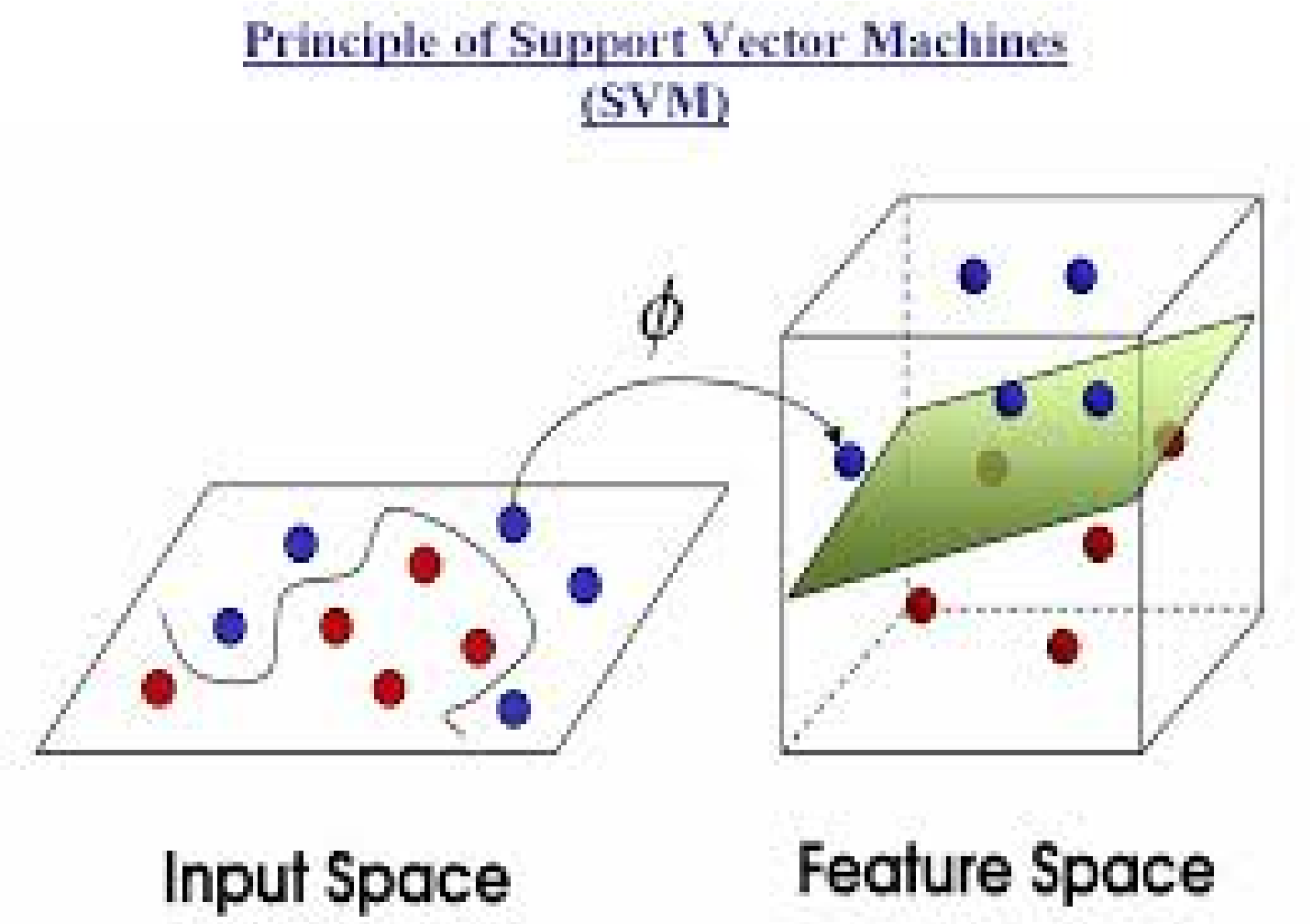
To find a non-linear decision boundary using a kernel, we compute the "Gram Matrix"– the inner product or similarity measure between all data points in the kernel space– $\phi(x)^T \cdot \phi(x)$.
The quadratic optimization problem becomes:

$$w(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$

with the constraints $\alpha_i \geq 0$, $\sum_i \alpha_i y_i = 0$
The new weights are given by:

$$w = \sum_i y_i \alpha_i \phi(x_i)$$

Schematic


Principle of Support Vector Machines
(SVM)
ϕ
Input Space
Feature Space

Constructing Kernels

Kernel functions are very flexible tools. They can be constructed for numeric data as well as data types such as strings (for text analysis), sequences (applications in bioinformatics), graphs, trees, images etc.

The real challenge in using kernels is constructing a kernel that works in a particular application. In fact, when we construct a kernel, we do so based on our prior knowledge about the data that we are analyzing and about the patterns that it contains.
Kernels are thus application and data specific.

Examples of Kernels

Linear kernel $k(x, y) = x^T y)$

Gaussian Radial Basis Function Kernel $k(x, y) = e^{\frac{-(x-y)^2}{\sigma^2}}$

Polynomial kernel $k(x, y) = (ax^T y + b)^p$

Examples of what you can do with SVM's

Medical classification– detecting disease from brain images and gene expression measurements

"Mind Reading"– Deducing which stimulus a person was presented with from fMRI timeseries'