Department of Mathematics, University of Udine

SVM and their use in IR

Lorenzo Zanolin.
lorenzo.zanolin@spes.uniud.it

May x, 2023

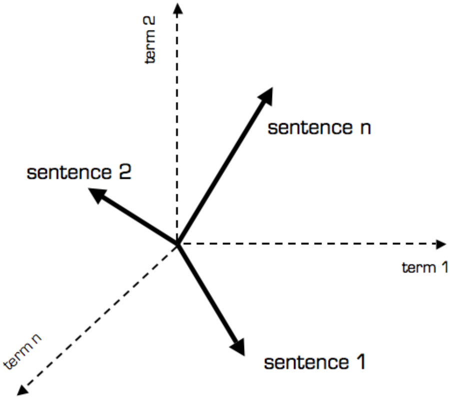
Outline

1. SVM over linearly separable data

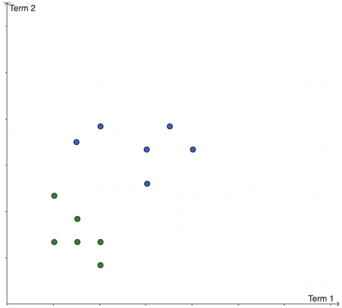
2. SVM over non linearly separable data

Suppose you have to classify whether a document is *relevant* or not. We can think to use *terms* as features to divide properly the data. We will use a *t-dimensional* vector space to represent our documents.

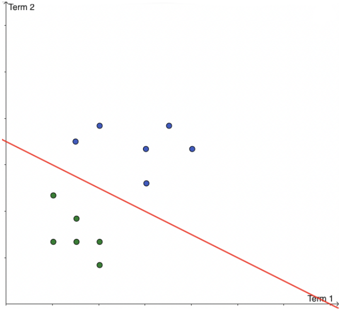
Suppose you have to classify whether a document is *relevant* or not. We can think to use *terms* as features to divide properly the data. We will use a *t-dimensional* vector space to represent our documents.



For simplicity, we can reason using a 2D Space.
Data can be separated using a *decision boundary*, which is an *hyperplane*.



For simplicity, we can reason using a 2D Space.
Data can be separated using a *decision boundary*, which is an *hyperplane*.



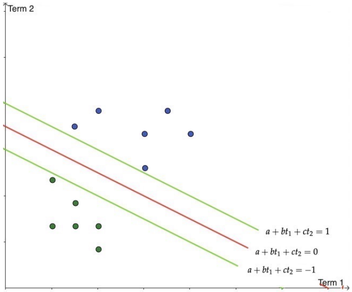
Support Vectors

In the previous example the *decision boundary* is a line, represented by the equation $a + bt_1 + ct_2 = 0$.
We can introduce two parallels hyperplanes (lines) to the decision boundary, called *support vectors* whose equations are

$$\begin{cases} a + bt_1 + ct_2 = 1 \\ a + bt_1 + ct_2 = -1 \end{cases} \tag{1}$$

In the previous example the *decision boundary* is a line, represented by the equation $a + bt_1 + ct_2 = 0$.
We can introduce two parallels hyperplanes (lines) to the decision boundary, called *support vectors* whose equations are

$$\begin{cases} a + bt_1 + ct_2 = 1 \\ a + bt_1 + ct_2 = -1 \end{cases} \tag{1}$$



Margin

For convenience, we can rewrite equations using *vectorization* notation.

$$\begin{cases} b^T t + a = 0 \\ b^T t + a = \pm 1 \end{cases} \tag{2}$$

For convenience, we can rewrite equations using *vectorization* notation.

$$\begin{cases} b^T t + a = 0 \\ b^T t + a = \pm 1 \end{cases} \tag{2}$$

Intuitively, we can define the *margin* of the two support vectors as the distance between them.

For convenience, we can rewrite equations using *vectorization* notation.

$$\begin{cases} b^T t + a = 0 \\ b^T t + a = \pm 1 \end{cases} \tag{2}$$

Intuitively, we can define the *margin* of the two support vectors as the distance between them.

We can consider two points x_1, x_2 that lie respectively on the two support vectors, their distance is $\lambda\|b\| = \frac{2}{\sqrt{b^T b}}$.

SVM are used to find the *maximum margin linear classifier*, thus we want to maximize the margin.

Remembering we want to classify documents, our goal is to find specific b s.t. given a document x belonging to class y the decision boundary behave the following:

$$\begin{cases} b^T x + a \geq 1 & \quad if\ y = 1 \\ b^T x + a \leq -1 & \quad if\ y = -1 \end{cases} \tag{3}$$

Remembering we want to classify documents, our goal is to find specific b s.t. given a document x belonging to class y the decision boundary behave the following:

$$\begin{cases} b^T x + a \geq 1 & \textit{if } y = 1 \\ b^T x + a \leq -1 & \textit{if } y = -1 \end{cases} \tag{3}$$

We can define the *cost function* as a system of equations.

$$\begin{cases} \min_{b,a} \frac{\sqrt{b^T b}}{2} \\ \textit{subject to} \quad y_i(b^T x_i + a) \geq 1 \quad \forall x_i \end{cases} \tag{4}$$

Soft-Margin

In real world problem it is not likely to get an exactly separate line dividing the data within the space. It would be better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers.

Soft-Margin

In real world problem it is not likely to get an exactly separate line dividing the data within the space. It would be better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers.

So, we will use *slack variables* to introduce a penalty for each misclassified point.

In real world problem it is not likely to get an exactly separate line dividing the data within the space. It would be better for the smooth boundary to ignore few data points than be curved or go in loops, around the outliers.

So, we will use *slack variables* to introduce a penalty for each misclassified point.

The new cost function will be

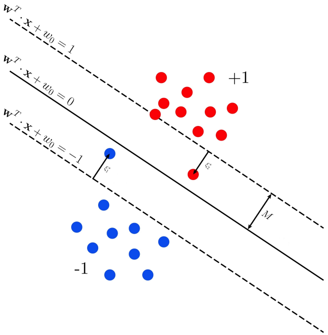
$$\begin{cases} \min_{b,a} \frac{\sqrt{b^T b}}{2} + C \sum_i \xi_i \\ \textit{subject to} \quad y_i(b^T x_i + a) \geq 1 - \xi_i \quad \textit{and } \xi_i > 0 \quad \forall x_i \end{cases} \tag{5}$$

The larger is C the stricter the classification is, since a larger C will give more evidence to slack variables.

$$\begin{cases} \min_{b,a} \frac{\sqrt{b^T b}}{2} + C \sum_i \xi_i \\ \textit{subject to} \quad y_i(b^T x_i + a) \geq 1 - \xi_i \quad \textit{and } \xi_i > 0 \quad \forall x_i \end{cases} \tag{6}$$

The larger is C the stricter the classification is, since a larger C will give more evidence to slack variables.

$$\begin{cases} \min_{b,a} \frac{\sqrt{b^T b}}{2} + C \sum_i \xi_i \\ \textit{subject to} \quad y_i(b^T x_i + a) \geq 1 - \xi_i \quad \textit{and } \xi_i > 0 \quad \forall x_i \end{cases} \tag{6}$$

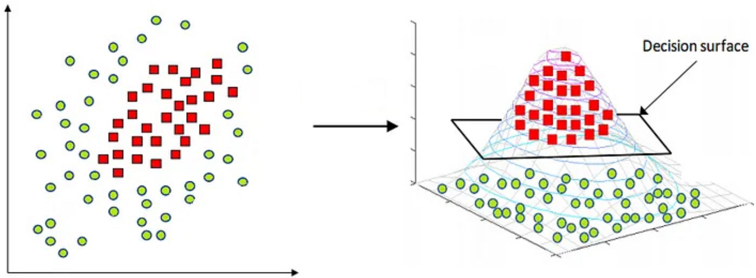


Non linearly separable data

What if our data is not linearly separable?

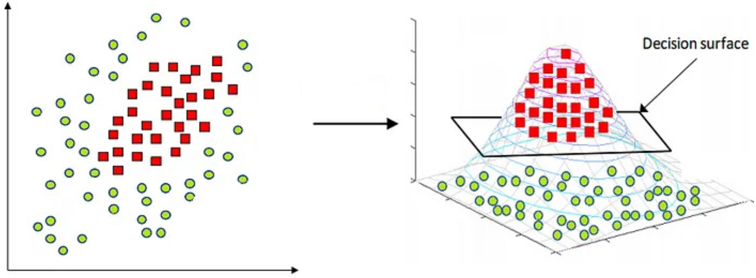
Non linearly separable data

What if our data is not linearly separable? It should be better to reason in a bigger dimensional space.



Non linearly separable data

What if our data is not linearly separable? It should be better to reason in a bigger dimensional space.



But augmenting dimensions costs a lot…

Kernel Trick

Consider the function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$ used to map points in a new vector space. Calculating the *similarity* $\phi(x_i)^T \phi(x_j)$ between each point may be intractable.

Kernel Trick

Consider the function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$ used to map points in a new vector space. Calculating the *similarity* $\phi(x_i)^T \phi(x_j)$ between each point may be intractable.

Here *Kernel Trick* comes handy.

Kernel Trick

Consider the function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$ used to map points in a new vector space. Calculating the *similarity* $\phi(x_i)^T \phi(x_j)$ between each point may be intractable.

Here *Kernel Trick* comes handy.

It consists of a simple linear algebra reformulation,

$$\phi(x_i)^T \phi(x_j) = K(x_i, x_j) = (1 + x_i{}^T x_j)^2. \tag{7}$$

Consider the function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$ used to map points in a new vector space. Calculating the *similarity* $\phi(x_i)^T \phi(x_j)$ between each point may be intractable.

Here *Kernel Trick* comes handy.

It consists of a simple linear algebra reformulation,

$$\phi(x_i)^T \phi(x_j) = K(x_i, x_j) = (1 + {x_i}^T x_j)^2. \tag{7}$$

Instead of doing the complex computations in the 10-dimensional space, we reach the same result within the 3-dimensional space by calculating the dot product.

Gaussian Kernel

There are lots of Kernel functions, an example is the *Gaussian*.

Gaussian Kernel

There are lots of Kernel functions, an example is the *Gaussian*.

The idea behind is to use all n training points as *landmarks*.
Then we can calculate the similarities of each point and all the landmarks.

There are lots of Kernel functions, an example is the *Gaussian*.

The idea behind is to use all n training points as *landmarks*.
Then we can calculate the similarities of each point and all the landmarks.

$$\forall l_i \; similarity(x, l^{(i)}) = \exp\left(-\frac{\left\|x - l^{(i)}\right\|^2}{2\sigma^2}\right) \tag{8}$$

There are lots of Kernel functions, an example is the *Gaussian*.

The idea behind is to use all n training points as *landmarks*. Then we can calculate the similarities of each point and all the landmarks.

$$\forall l_i \; similarity(x, l^{(i)}) = \exp(-\frac{\left\| x - l^{(i)} \right\|^2}{2\sigma^2}) \tag{8}$$

The result will be a mapping for each point in a n-dimensional space. Finally, we can identify an hyperplane which can divide correctly the two classes of points.