Table of Contents

Support Vector Machines

We have a space with negative and positive examples. The problem is to separate the samples using a straight line. And then line is drawn with the view of putting the widest street that separates the positive from the negative examples. (The margin that I drawn is the smallest distance to the closest point)

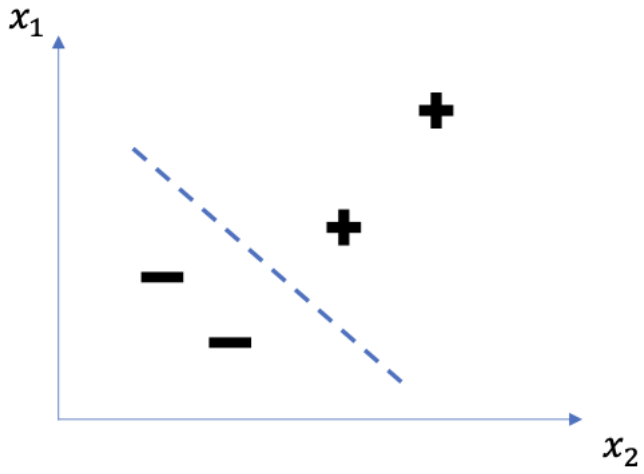


Figure 1: image

We want to create a decision rules that use that decision boundary

Imagine a vector $\vec{w}$ constraint to be perpendicular to the medium line of the street. And the we have an unkown vector $\vec{u}$ and we are interested in understanding if that vector is on the right side or on the left side of the steet. We can project $\vec{u}$ down to the one that is perpendicular to the street because then, we will have the distance in the $\vec{w}$ direction and the further out we go the closer we'll get to being on the right side of the street. So we can say

$$\vec{w} \cdot \vec{u} \geq c$$

The dot product take the projection onto w and the bigger that projection is, the further out along this projection will lie and eventually it will be so big that the projection crosses the median line of the street, and we'll say it must be a positive sample. Without loss of generality we can say that

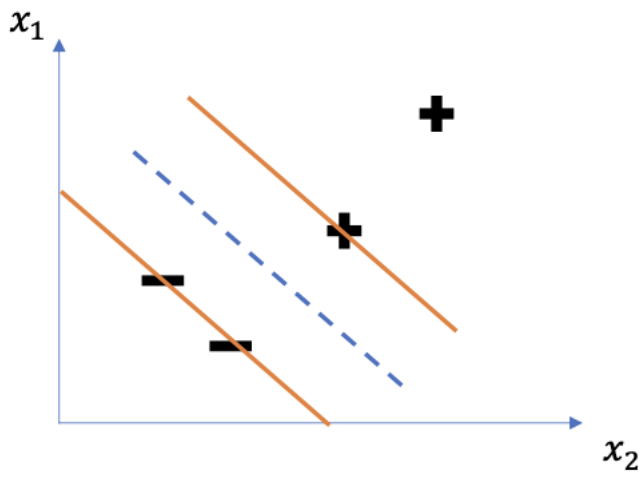


Figure 2: image

$$\vec{w} \cdot \vec{u} + b \geq c \text{THEN} +$$

$$c = -b$$

That's our decision rule.

So we define that for a positive sample this equation is true

$$\vec{w} \cdot \vec{x_+} + b \geq 1$$

Likewise for a negative sample

$$\vec{w} \cdot \vec{x_-} + b \leq -1$$

Let's introduce another variable for mathematical convenience

y_i such that $y_i = 1$ for positive samples and $y_i = -1$ for negative samples.

So:

$$y_i(\vec{w} \cdot x_i + b) \geq 1 y_i(\vec{w} \cdot x_i + b) - 1 \geq 0$$

And $y_i(\vec{w} \cdot x_i + b) - 1 = 0$ for x_i in the GUTTER

The goal is to arrange for the line to be such at the street separating the pluses from the minuses as wide as possible. So we need to express the distance between the two gutters.

We can consider the difference between the two vectors $\vec{x_+} - \vec{x_-}$ and project the difference to a unit vector that is perpendicular to the straight line.

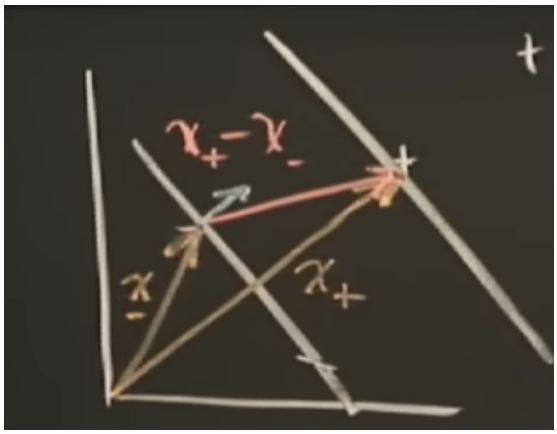


Figure 3: image

$$\text{WIDTH} = (\vec{x_+} - \vec{x_-}) \cdot \frac{\vec{w}}{||w||})$$

$$\frac{(1-b)+(1+b)}{||w||} = \frac{2}{||w||}$$

And this represent the width of the street and we want to maximize that quantity $\text{MAX}\frac{2}{||w||}$ equals to $\text{MIN}||w||$ equals to $\text{MIN}\frac{1}{2}||w||^2$. We have also a contraint that is:

$$y_i(\vec{w} \cdot x_i + b) \geq 1$$

We've now transformed the problem into a form that can be efficiently solved. The above is an optimization problem with a convex quadratic objective and only linear constraints. Its solution gives us the optimal margin classifier.

Lagrange duality

Consider a problem of the following form:

$$min_w f(w) \text{s.t} h_i(w) = 0, i = 1, ..., l$$

This problem can be solved with a method called the Lagrange multipliers. The Langrangian is defined to be:

$$L(w, \beta) = f(w) + \sum_{i}^{l} \beta_i h_i(w)$$

Here, the β_i's are called the **Lagrange multipliers**. We would then find and set L's partial derivatives to zero

$$\frac{\partial L}{\partial w_i} = 0; \frac{\partial L}{\partial \beta_i} = 0$$

and solve for w and β. ### Primal optimization problem

In this section, we will generalize this to constrained optimization problems in which we may have inequality as well as equality contraints. Due to In order to maximize the size of the street we're going to maximize the following quantity (lagrange multipliers)

$$min_w f(w) \text{s.t} g_i(w) \leq 0, i = 1, ..., k h_i(w) = 0, i = 1, ..., l.$$

To solve it, we start by defining the **generalized Lagrangian**

$$L(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} (\beta_i h_i(w))$$

Here, the α_i's and β_i's are the Lagrange multipliers. The generalized Lagrangian correspond to the dual problem

$$\text{maximize} L(w, \alpha, \beta) \text{s.t} \alpha \geq 0$$

In our case we have that:

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1]$$

So we're going to compute the partial derivative

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum \alpha_i y_i x_i = 0 \Rightarrow \vec{w} = \sum_{i} \alpha_i y_i \vec{x_i}$$

This tells us that the vector w is a linear sum of the samples. **BUT NOT ALL** the samples, because some alphas could be zero. We can differentiate also for b

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

If we compute the differentation also for alpha

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \alpha_i [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] \frac{\partial L}{\partial \alpha} = [y_i(\vec{w} \cdot \vec{x_i} + b) - 1] = 0$$

Now that we got a value for w we can substitute the formula in L

$$L = \frac{1}{2}\left(\sum \alpha_i y_i \vec{x_i}\right)\left(\sum \alpha_i y_j \vec{x_j}\right) - \sum \alpha_i y_i x_i \left(\sum \alpha_j y_j x_j\right) - \sum \alpha_i y_i b + \sum \alpha_i$$

And $\sum \alpha_i y_i b = 0$ because of the partial derivative with the respect of b

The Lagrangian can be rewrited as

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x_i} \cdot \vec{x_j}$$

We're trying to find a maximum of L and what I discovered that the optimization depends only on the dot product of pairs of samples $x_i \cdot x_j$ So now, my decision rule with this expression for w is going to be

$$\sum \alpha_i y_i \vec{x_i} \cdot \vec{u} + b \geq 0 \rightarrow +$$

So we discovered that the decison rules depends only on the dot product of those sample vectors and the unknown.

Non Linear separable data

The derivation of the SVM as presented so far assumed that the data is linearly separable. When data are non-linearly separable, we may get a separation between classes with a hyperplane only allowing that, after having defined the separating hyperplane, some pattern of the training set with positive label are classified as negative and viceversa. We must accept that some constraints are **violated**

We introduce a **slack** variable ξ_i for each constraint, in order to allow an error tolerance:

$$y^{(i)}(w^T \cdot x^i + b) \geq 1 - \xi^i$$

An additional term C is introduced in the cost function to penalize misclassification errors.

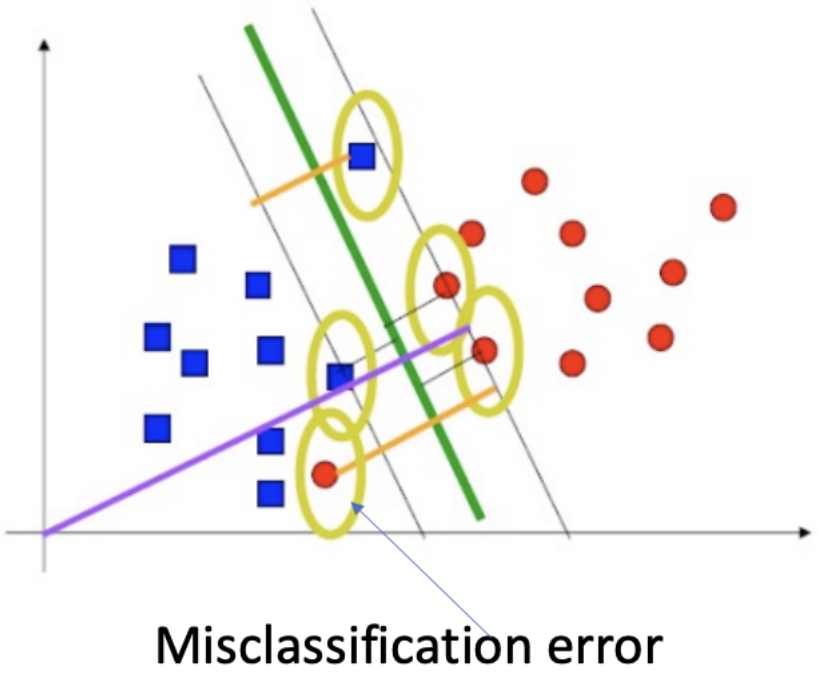


Figure 4: image

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)}$$

The ξ_i are cost variables proportional to how far the misclassified pattern is from the hyperplane. $\xi_i > 1$ indicates a misclassification error.

C (**regularization** parameter) lets to control the trade-off between hypothesis space complexity and the admissible number of errors. A big value for C gives a stronger penalization to errors. The optimization problem to solve becomes:

Minimize:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi^{(i)}\text{s.t}y^i(w^T \cdot x^i + b) \geq 1 - \xi^{(i)}\xi^{(i)} \geq 0$$

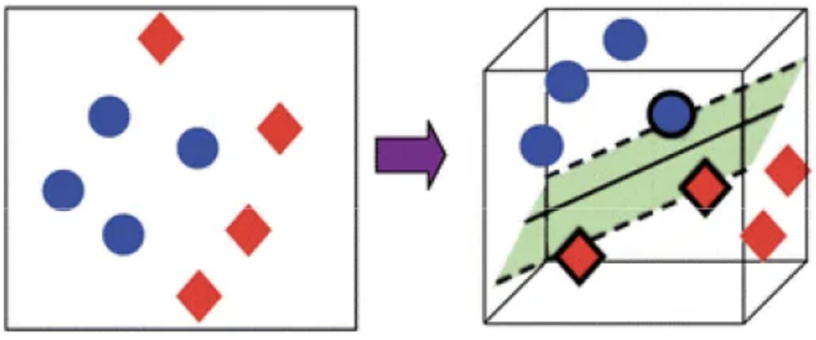
The dual problem now becomes:

$$max_\alpha \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum \alpha_i\alpha_j y_i y_j x_i^T x_j \text{s.t. } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m}\alpha_i y_i = 0$$

The dual variables are now bounded with C. The proposed solution could not be enough. It does not guarantee good performances since a hyperplane can only represent a dichotomy in the space of instances/patterns.

Cover's theorem

Cover's theorem state that, a complex pattern classification problem, cast in a high dimensional space nonlinearly, is more likely to be linearly separable than in a low dimensional space, provided that the space is not densely populated. Or in simple terms, given a set of training data that is not linearly separable, one can transform it into a training set that is linearly seperable by mapping it into a possibly higher dimensional space via some non linear transformation.



1. Patterns (input space) are mapped into a space with (much) higher dimension (feature space) through kernel functions;
2. The optimal hyperplane is defined within this feature space

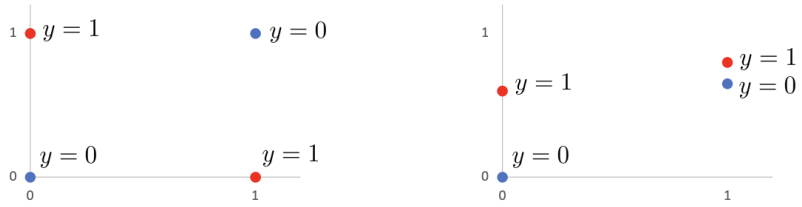


Figure 5: image

In this case the samples are not linearly separable and we are not able to find a solution for our problem, but we can move to another space that is more convenient for our purposes:

$$\vec{x} \Rightarrow \phi(\vec{x})$$

For our optimization problem now all we need is a function K that corresponds to the dot product between the samples in the new space.

This function K is called **kernel function** and we have no need to compute the transformation function $\phi(\vec{x_i})$

$$K(x_i, x_j) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

Kernel trick

To solve the optimization problem, the product $\phi(\vec{x_i}) \cdot \phi(\vec{x_j})$ does not have to be explicitly computed in the feature space once we find a kernel function (specifically a positive definite kernel).

$$K(x_i, x_j) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$$

The kernel is a function that returns the (scalar) product of projections: it avoids you to explicitly compute the projection and make the product between the projected vectors. The explicit form of the transformation function may be ignored.

If a kernel function is defined, that is a function such that:

$$K(x_i, x_j) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j}) = \sum_{k=1}^{m} \phi(\vec{x_i}) \cdot (\phi(\vec{x_j}))$$

The optimization problem becomes:

$$max_\alpha \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

Such that:

$$0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{m} \alpha_i y_i = 0$$

The projection $\phi(\vec{x_i})$ into the feature space must not be explicitly computed. I should compute the scalar product $\phi(\vec{x_i}) \cdot (\phi(\vec{x_j}))$, but I don't have to, since I can indirectly obtain it with the kernel function.

Kernel examples

Linear kernel

$$(\vec{u} \cdot \vec{v})$$

Polinomial kernel

$$(\vec{u} \cdot \vec{v} + 1)^d$$

Multi-Layer Perceptron tanh

$$tanh(b(\vec{u} \cdot \vec{v} - c))$$

Radial basis function (RBF) kernel

$$e^{-\frac{||x_i - x_j||}{\sigma}}$$

Gaussian Radial basis function kernel

$$e^{-\frac{(x_i - x_j)^2}{2\sigma^2}}$$

Parameters tuning

To use Support Vector Machines you have to define - the kernel function; - potential parameter of the kernel function; - The value for the regularization parameter C.

General rules for the set up do not exist, but you should make your choice on a validation set, usually through cross validation.

Advantages of SVMs

- There are no local minima (the optimization problem is quadratic -> $\exists$! optimal solution)
- The optimal solution can be found in polynomial time.
- There are few parameters to set up (C, type of kernel and specific kernel parameters)
- Solution is stable (ex. there is no problem of randomly initializing of weights just as in Neural Networks)
- Solution is sparse: it just involves support vectors

SVM Online resources

- https://www.youtube.com/watch?v=_PwhiWxHK8o
- https://www.youtube.com/watch?v=xpHQ6UhMlx4

Generalized Linear Models

Basic functions

It is usual that our data cannot be approximated to a linear function. What we need is to find a way to model non linear relations without increasing too much the complexity of the algorithm.

The main limiting characteristics of linear regression is the linearity of parameters

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot x_i = \theta^T x$$

h(x) keeps a linear relation w.r.t the features space X. This linearity represents a limitation of the expressiveness of the model because the hypothesis is only able to approximate linear functions of the input. GLMs represent an extension to linear models that allow non linear transformations of the input

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot x_i = \theta^T x$$

h(x) keeps a linear relation w.r.t the features space X. This linearity represents a limitation of the expressiveness of the model because the hyporthesis is only able to approximate linear functions of the input. GLMs represent an extension to linear models that allow nonlinear transformations of the input

$$h(x) = \sum_{i=0}^{n} \theta_i \cdot \psi(x_i) = \theta^T \psi(x)$$

Phi are called basic functions Example $x_i, x_2, x_3 \rightarrow x_1, x_2, x_3, x_1x_2, x_1x_3, x_1^2, x_2^2, x_3^2$

SVM notes from Waterloo lecture

Support vector machines will find a linear separator such that we have the same distance for point on one side and point of the other side and line is gonna be right in the middle. The point that are the closest are called support vector. And support vector tells us where to put the line such as it has the same distance and distance is the maximum we can obtain.

Unsupervised Learning