Support Vector Machines

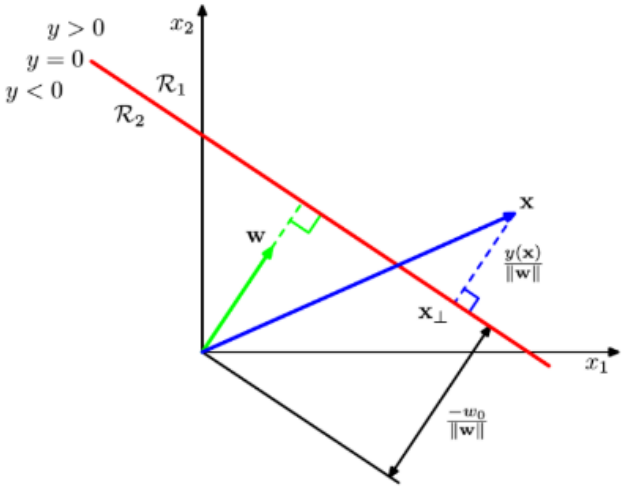
Tries to apply similar principles behind kernel methods but don't requires to compute the matrix k completely. So these techniques are called sparce kernel methods.

Separable Problems

Start from the idea of solving a binary determination techniques. The model we are working is the linear separation model seen in perceptron without the explicit mapping.

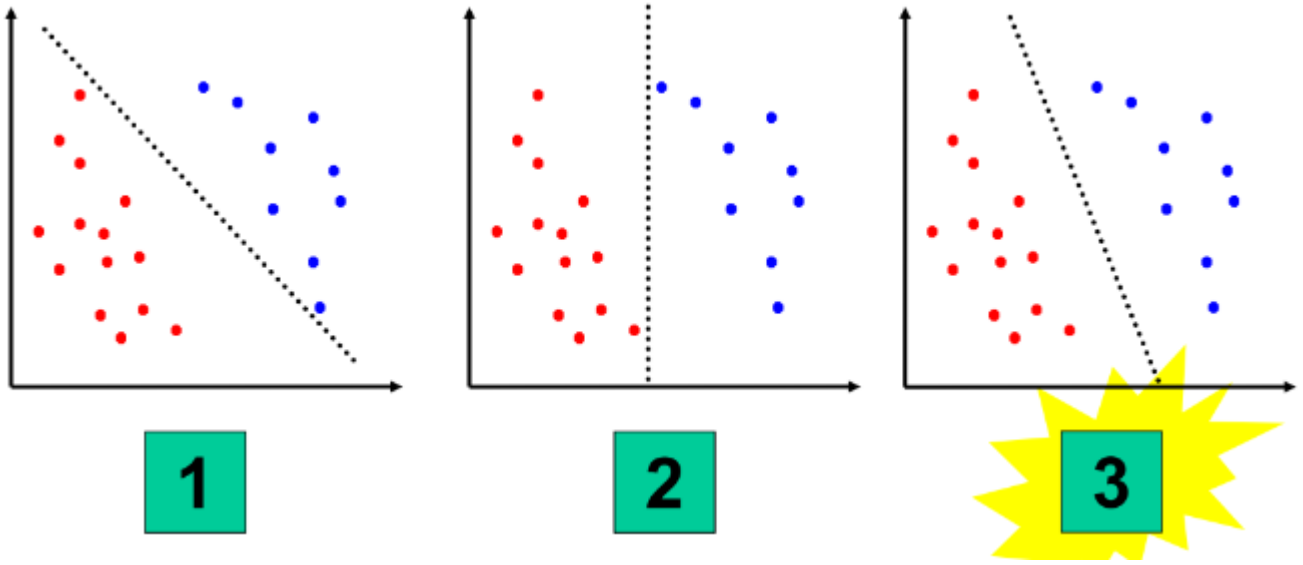
$$f(\mathbf{x}, \mathbf{w}) = \begin{cases} +1, & \text{if } \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

- Properties
  - DS is $y(\cdot) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}) + b = 0$
  - DS is orthogonal to $\mathbf{w}$
  - distance of DS from origin is $-\frac{w_0}{\|\mathbf{w}\|_2}$
  - distance of $\mathbf{x}$ from DS is $\frac{y(\mathbf{x})}{\|\mathbf{w}\|_2}$



$y(.) = \mathbf{w}^T\phi(\mathbf{x}) + b$

To get the real distance we can take the distance and multiply it by t



Different linear separation are really equals or there is one that is better than the others(Usually the best is the one that has the maximum distance by the closest point in the two separated spaces).

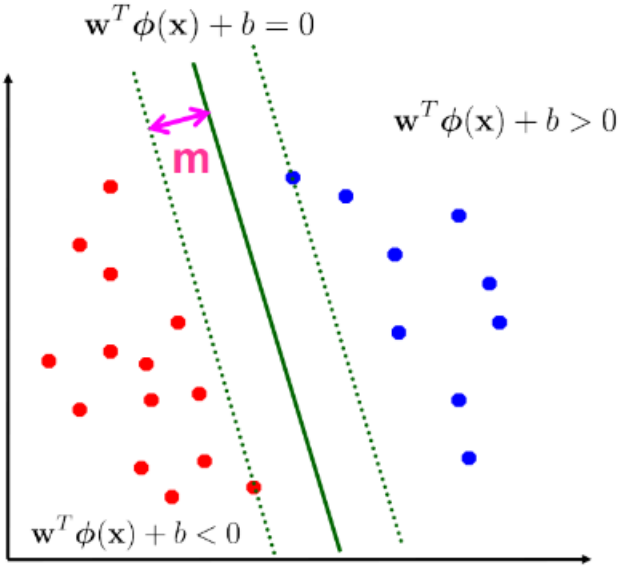
- The margin will be:

$$margin = \min_n \frac{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)}{||\mathbf{w}||}$$

- Thus, the optimal **hyperplane** will be:

$$\arg\max_{\mathbf{w},b} \left\{ \min_n \left[\frac{t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b)}{||\mathbf{w}||} \right] \right\}$$

- Unfortunately, solving this optimization problem would be very complex...



I want to find the one that is maximising the margin. In reality the margin needs to be symmetric in respect to the positive and negative points. There is a bit of ambiguity in how the position vector and the weight b are defining the solution.
The first part of the solution is to introduce Canonical Hyperplanes(Linear separator):

- **Canonical** hyperplane
  - There are infinite equivalent solutions:

$$\kappa\mathbf{w}^T\phi(\mathbf{x}) + \kappa b \quad \forall\kappa > 0$$

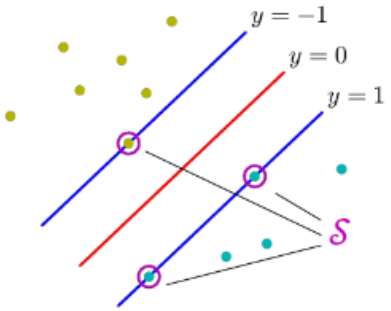
  - We will consider only solutions such that:

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) = 1 \quad \forall\mathbf{x}_n \in \mathcal{S}$$

- Equivalent **quadratic programming** problem

$$\textbf{Minimize} \quad \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$\textbf{Subject to} \quad t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1, \text{ for all } n$$



We want to be sure that the numerator of the distant equation computed in the two closest points is equal to 1/-1. Requiring this I am forcing the problem to consider only one solution. So the only thing affecting the margin the only thing I can focus on is the norm of the weight vector needing to minimise it(constrain optimisation problem).
We want to solve these type of problem to maintain the kernel trick is relying to the technique called Lagrange Multiplier:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(t_i \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b\right) - 1\right)$$

We need to maximize $\mathcal{L}$ with respect to α and minimize it with respect to $\mathbf{w}$ and b
We can compute the gradient w.r.t. $\mathbf{w}$ and b and derive dual representation

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i t_i \boldsymbol{\phi}(\mathbf{x}_i)$$

$$\frac{\partial}{\partial b}\mathcal{L} = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

The problem is translated in a new one where the variable α is non negative and we want to maximise the new function in respect to α and we want to mimize in respect to the original variable w.

We can now rewrite the optimization problem as:

$$\textbf{Maximize} \quad \tilde{\mathcal{L}}(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\textbf{Subject to} \quad \alpha_n \geq 0 \text{ and } \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \text{for } n = 1, \ldots, N$$

- where the explicit feature mapping does not appear explicitly anymore

If the points aren't sit on the margins the parenthesis part is positive and so L contains something positive and something negative and then α needs to be 0.

Discriminant Function and Support Vectors

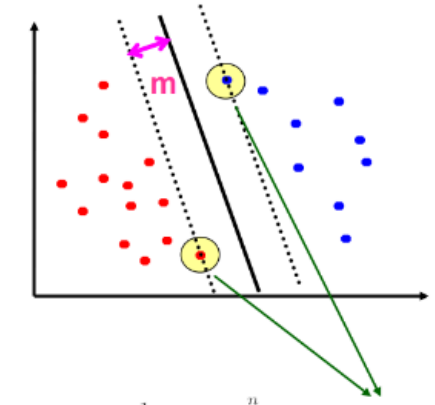
- The resulting discriminant function is:

$$y(\mathbf{x}) = \sum_{n=1}^{N} \alpha_n t_n k(\mathbf{x}, \mathbf{x}_n) + b$$

  - Only samples on the margin does contribute (i.e., $\alpha_i > 0$),
  - These samples are the Support Vectors
  - The bias is computed as:

$$b = \frac{1}{|\mathcal{S}|} \sum_{\mathbf{x}_n \in \mathcal{S}} \left(t_n - \sum_{\mathbf{x}_m \in \mathcal{S}} \alpha_m t_m k(\mathbf{x}_n, \mathbf{x}_m) \right)$$

Support Vectors ($\mathcal{S}$)



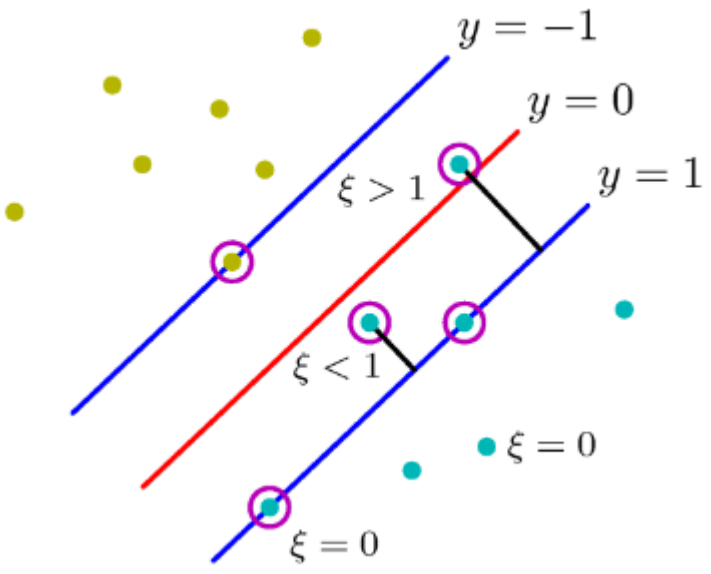
$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left(t_i \left(\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i) + b \right) - 1 \right)$$

0

This because our problem is still expensive and it depends also on the dimension of the space. The benefit is that I have a free choice on the kernel function permitting to make complex problem linearly separable.

Non-separable Problems

In some case it might not be convenient to find a linear separator as the model becomes complex in regards to the bias-variance separators. The non separable problems are problems resolved by a simpler model regarding bias- covarinace. In these problems i introduce some variables called Slack variables representing the error I am committing. This variable will be zero if they are on/outside the margins but it will positive if the point is on the positive side and it is inside the margin.



So the problem settings become:

$$\textbf{Minimize} \quad \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{n=1}^{N}\xi_n$$

$$\textbf{Subject to} \quad t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b) \geq 1 - \xi_n, \qquad \text{for all } n$$

$$\xi_n \geq 0, \qquad \text{for all } n$$

ξ_n are called **slack variables** and represents **penalties** to **margin violation**
C is a tradeoff parameter between error and margin

- it allows to adjust the **bias-variance tradeoff**
- **tuning** is required to find optimal value for C

If we increase C the model becomes more complex as every mistake it is expensive.

Dual Representation (Box Constraints Problem)

$$\textbf{Maximize} \quad \tilde{\mathcal{L}}(\boldsymbol{\alpha}) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m)$$

$$\textbf{Subject to} \quad 0 \leq \alpha_n \leq C,$$

$$\sum_{n=1}^{N}\alpha_n t_n = 0, \qquad \text{for } n = 1, \ldots, N$$

- As usual, **support vectors** are the samples for which $\alpha_n > 0$
  - If $\alpha_n < C \Longrightarrow \xi_n = 0$, i.e., the sample is on the margin
  - If $\alpha_n = C$ the sample can be inside the margin and be either correctly classified ($\xi_n \leq 1$) or misclassified ($\xi_n > 1$)

The idea is that the points outside the margins are going to be considered as zero not counting in the determination. The points on the margins/inside the margins will be counted in the solution and will divide in on margin and inside the margin.
One of the problem is that the optimal choice of C is very critical. For this reason there is an effort for searching an alternative settings where we don't search C:

Alternative formulation: ν-SVM

$$\begin{aligned}
\textbf{Maximize} \quad & \tilde{\mathcal{L}}(\boldsymbol{\alpha}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \\
\textbf{Subject to} \quad & 0 \leq \alpha_n \leq 1/N, \\
& \sum_{n=1}^{N}\alpha_n t_n = 0, \\
& \sum_{n=1}^{N}\alpha_n \geq \nu \qquad\qquad \text{for } n = 1, \ldots, N
\end{aligned}$$

- Where $0 \leq \upsilon < 1$ is a user parameter that allow to control both the margin errors and the number of support vectors:

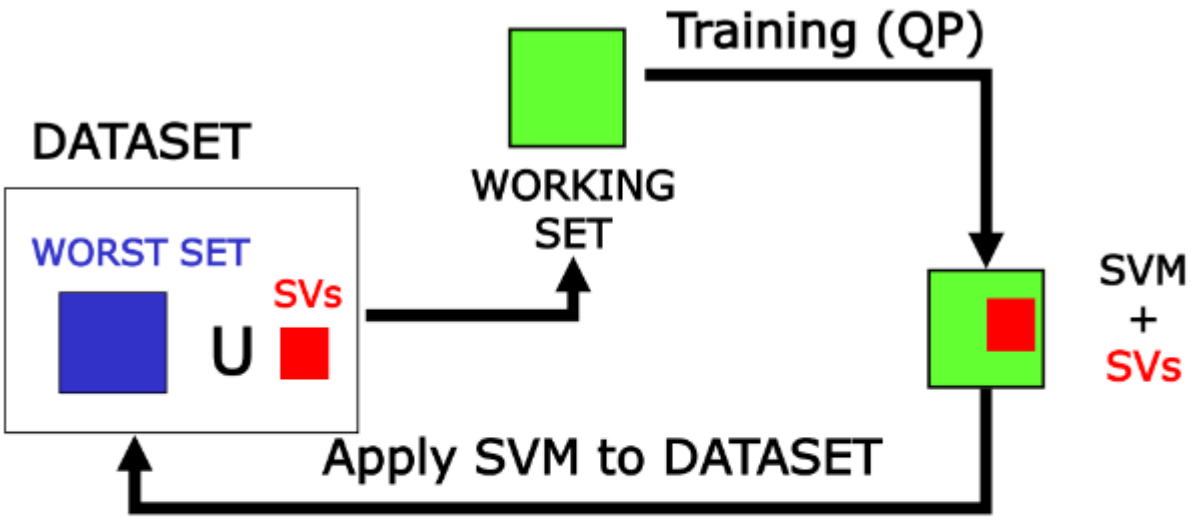
fraction of Margin Errors ≤ ν ≤ fraction of SVs

The cost of the solution cannot go below v(If I have a lot of Margin Error I also have a lot of Support Vectors).

Training SVM in Practice

How can I train in practice? The sparcity of the machine is sparcity of the solution. The sparcity property is exploited after we found the solution and not resolve the computational cost of the computation.

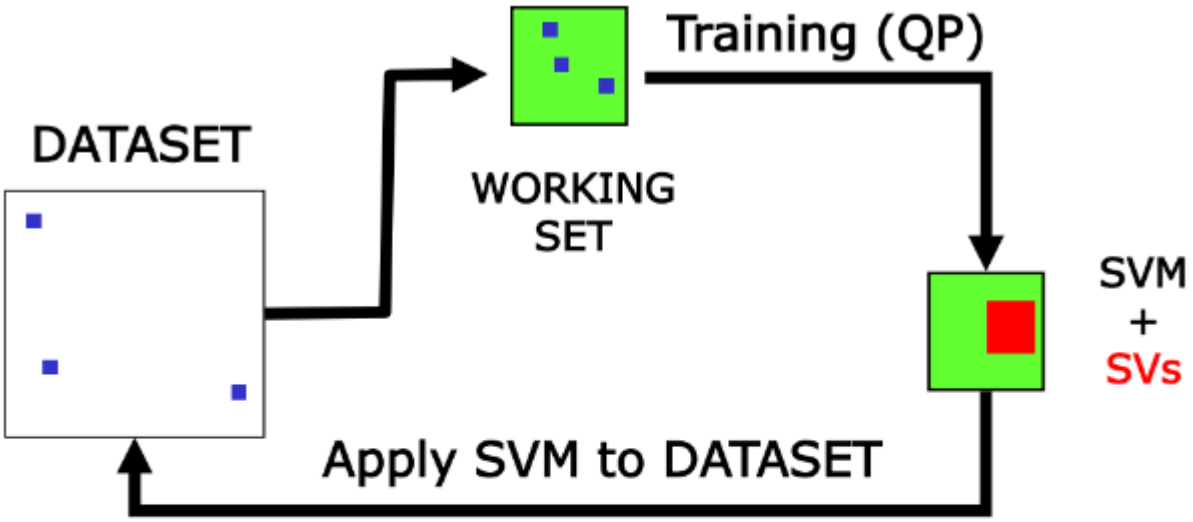
Chunking



- Solves iteratively a sub-problem (working set)
- Build the working set with current SVs and the M samples with the bigger error (worst set)
- Size of the working set may increase!
- Converges to optimal solution!

Instead of solving a large optimization problem I resolve iteratively small optimization problem. I start with solving my working set and use training to find some support vector. Then I apply the model learned on the entire dataset finding the Worst set(worst point in the original data, larger slack variable, its size is up to equally the working set). I then use this set and the previous Support vector to find the new dataset. One problem is that the dimension of the vector is not kept in check and it can increase indefinitely per each iteration.

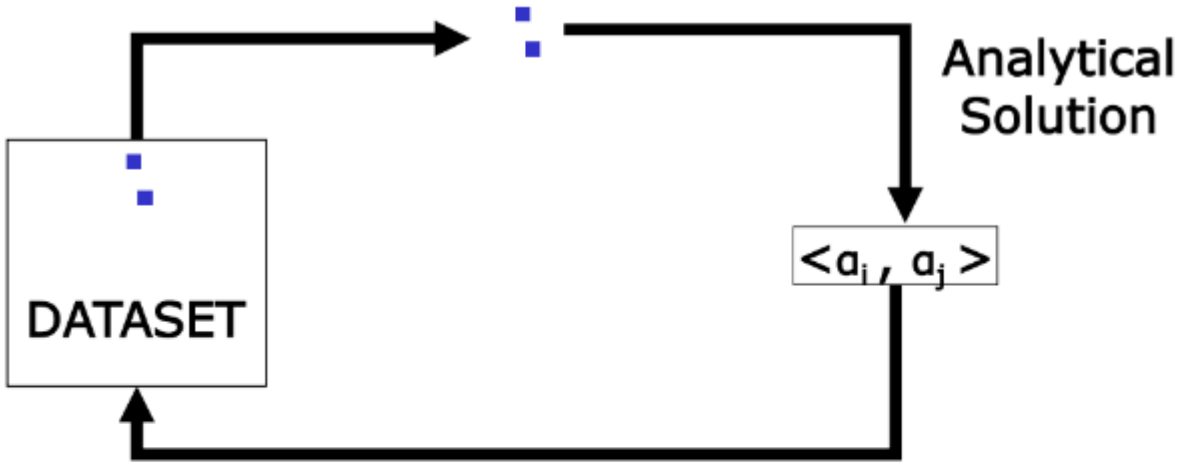
Osuna's Method



- Solves iteratively a sub-problem (working set)
- Replace some samples in the working set with missclassified samples in data set
- Size of the working set is fixed!
- Converges to optimal solution!

Similar put solve the problem of the size of the working set. It don't identify the Worst set but identifies only a small part of the dataset and substitute them in the working set.

Sequential Minimal Optimization



- Works iteratively only on two samples
- Size of the working set is minimal and multipliers are found analytically
- Converges to optimal solution!

The idea is to sequentially solve simple problem(I consider two sample at once). I can compute analytically the solution for these two points and use the Lagrange computation to update the

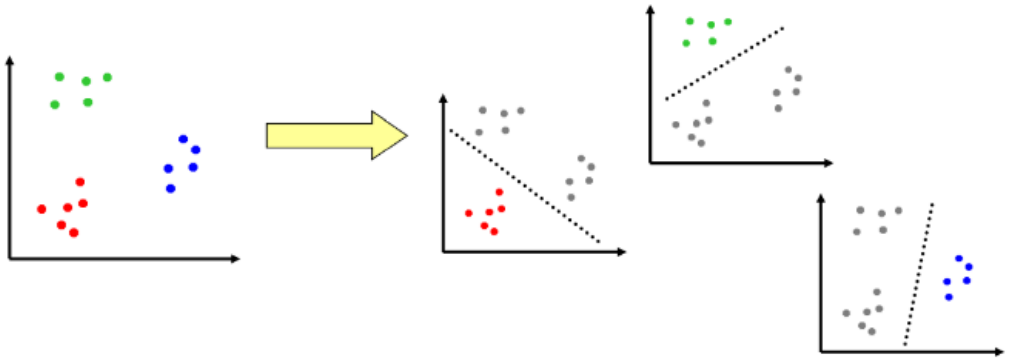
Lagrange multiplier for the points in the dataset.

Multi-class SVM

Difficult to transfer the bidimensional SVM to multiclass problem so we use the previous shceme:

One-against-all

- A k-class problem is decomposed in k binary (2-class) problems

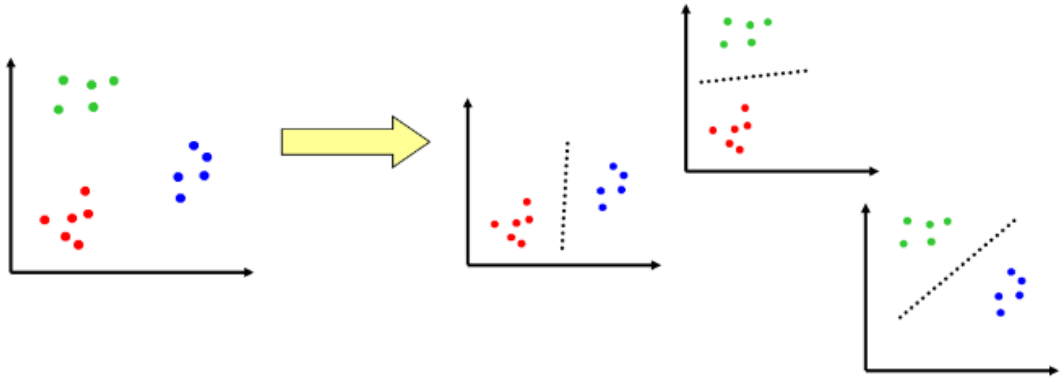


- Training is performed on the entire dataset and involves k SVM classifiers
- Test is performed choosing the class selected with the highest margin among the k SVM classifiers

Solve a multiclass problem training several binary classification problems

One-against-one

- A k-class problem is decomposed in k(k-1)/2 binary (2-class) problems

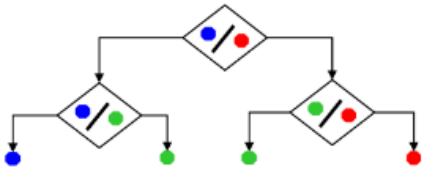


- The k(k-1)/2 SVM classifiers are trained on subsets of the dataset
- Test is performed by applying all the k(k-1)/2 classifiers to the new sample and the most voted label is chosen

You need way more model so if we have 3 is 3 but with 5 we need 10 model but now I train the model to discriminate between the classes so I train only on a part of the data.
In general I have some tradeoff as it requires more models but these models are less expensive. The test could be expensive as I am testing on a large number of models.

DAGSVM

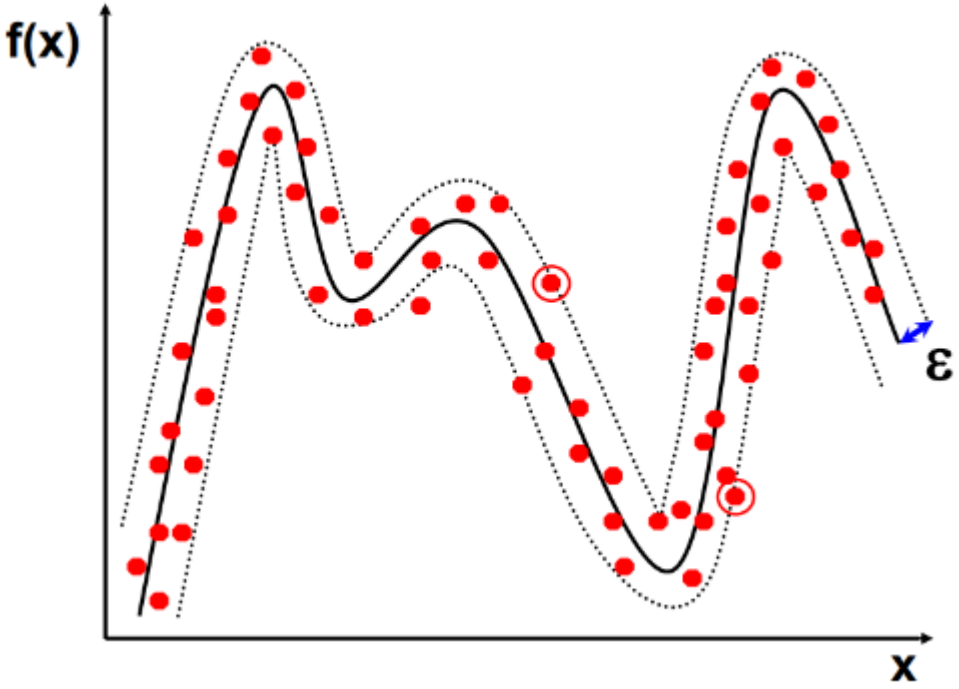
- In DAGSVM, the k-class problem is decomposed in k(k-1)/2 binary (2-class) problems as in one-against-one
- Training is performed as in one-against-one
- But test is performed using a Direct Acyclic Graph to reduce the number of SVM classifiers to apply:

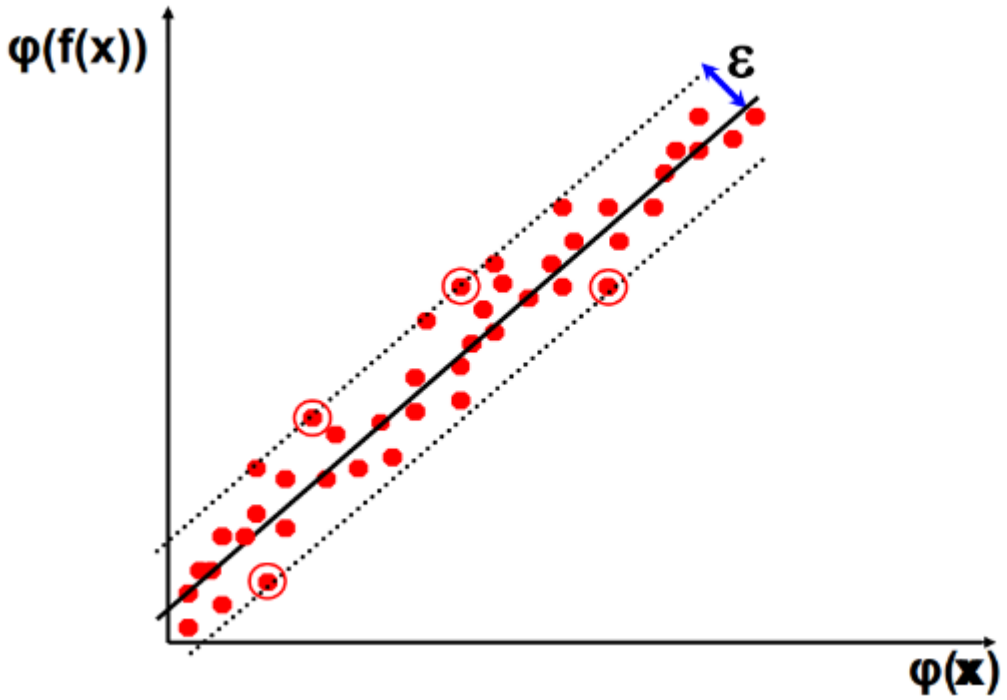


- The test process involves only k-1 binary SVM classifiers instead of k(k-1)/2 as in one-against-one approach

The measure problem is that this model have critical case concentrated on the initial part of the tree. But at the end there is no ambiguity and it retains the benefit to train small data on small portion of data but at test time you need to find the single path and the number of test is linear to the number of classes.

SVM for Regression





It not include a linear separator, can be found in ML libraries. The idea is I have my model and have a regression line and use it as my thresholds on the maximum error value. The points that are Support vectors now are the points that are on the margins. It is happening not in the input space. You are applying the kernel trick and working in the hyperspace and will have a linear model. The model won't be a parametric model but the parameters of the model depends on the support vectors found during the calculations.