Music Machine Learning

III – Support Vector Machines

Master ATIAM - Informatique

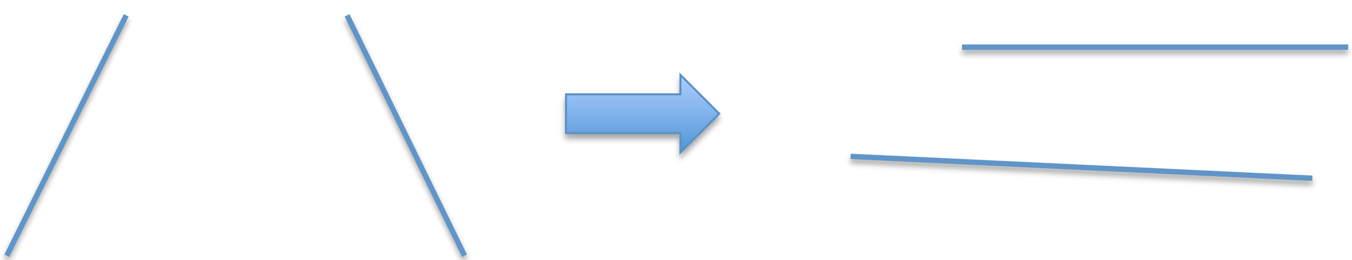
Philippe Esling (esling@ircam.fr)

Maître de conférences – UPMC

Equipe représentations musicales (IRCAM, Paris)

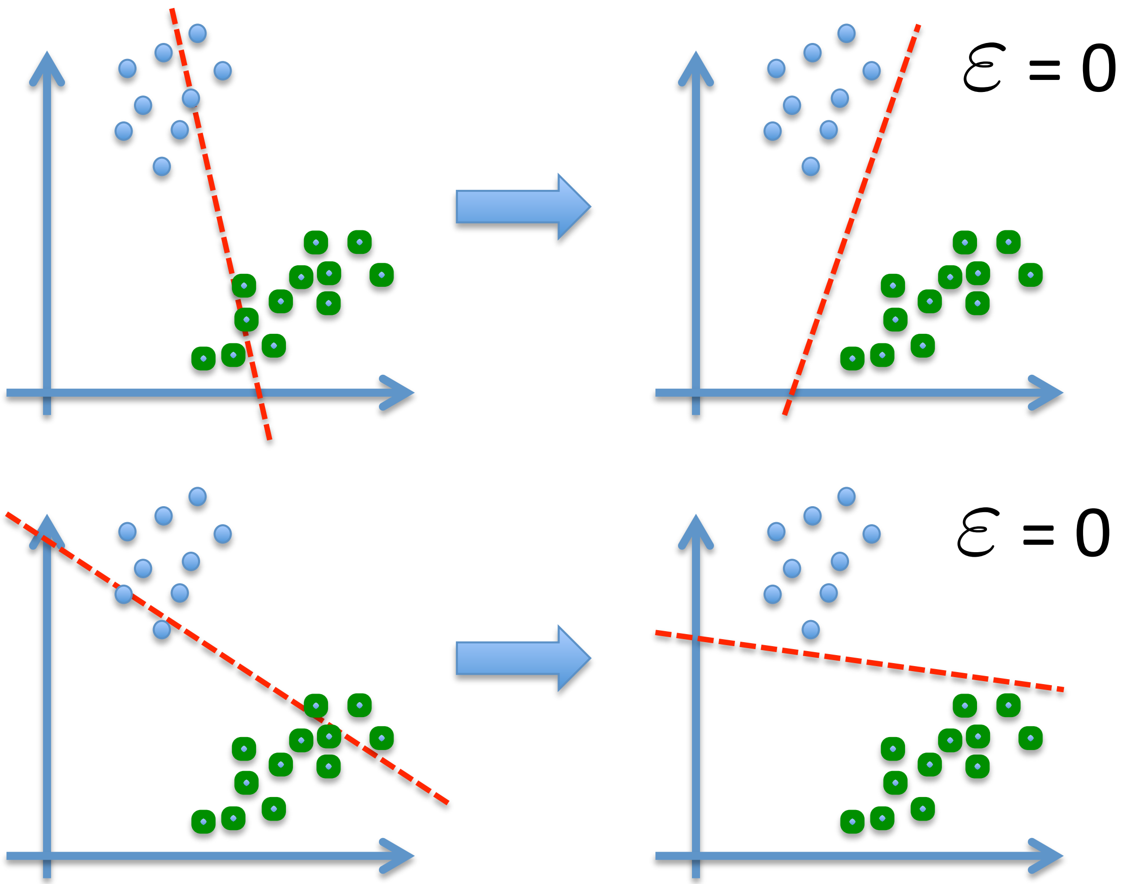
Support Vector Machines

- So all methods sort of divide the space
- But the essential question is **how** to divide the space
    1. Centroïd-based division
    2. Decision tree
- But is this space needs to be fixed?
- How things look from a different angle?
- We can obtain this if we know the geometry of the world



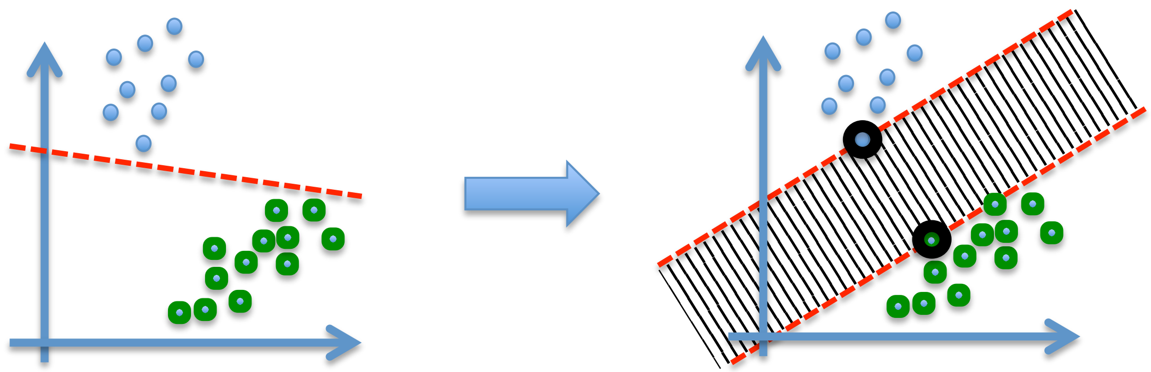
- Rich and powerful through geometrical models
- Neural networks provided a whole **set** of solutions

Support Vector Machines



Support Vector Machines

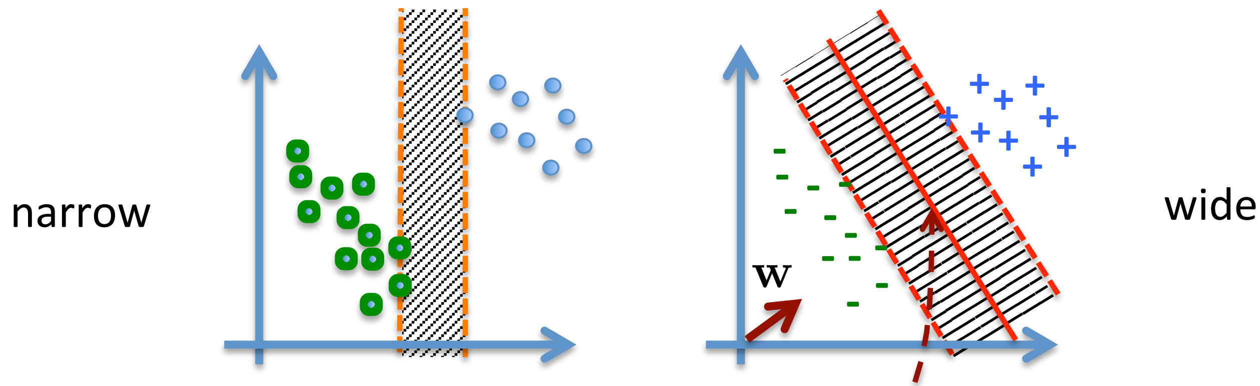
- Neural networks provided a whole set of solutions
- Each one defines a « street » between two populations
- Can we find the **widest** street (margin) between two populations
- Based on the hypothesis that the widest is the best separation



- So how to find this « optimal » separation?
- What is my decision function? (typical optimization question)

Support Vector Machines

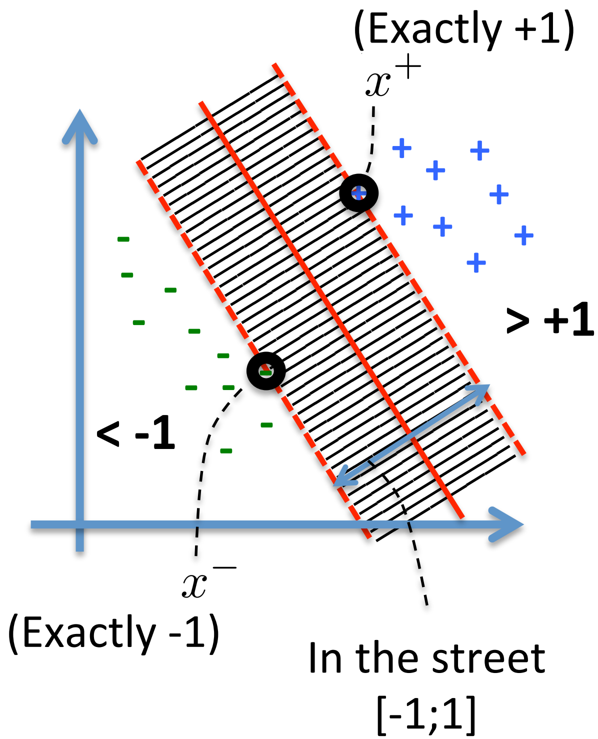
So what decision function could separate these two situations ?



1. Take the **median** of the street
2. Consider a vector $\mathbf{w}$ from origin perpendicular to it
3. Compute for any unknown spot $\mathbf{w} \cdot \mathbf{x} + \mathbf{b} \geq 0$
4. This allows to perform separation
5. So our decision function might look like

$$\mathbf{w} \cdot \mathbf{u} + \mathbf{b} \geq 0$$

Support Vector Machines



Based on our potential decision function

$$\underbrace{\mathbf{w}}\cdot\mathbf{u}+\mathbf{b}\geq 0$$

So far we only know that this is orthogonal to median

- How to ensure that we find **the widest** ?
- We can add two additional constraints

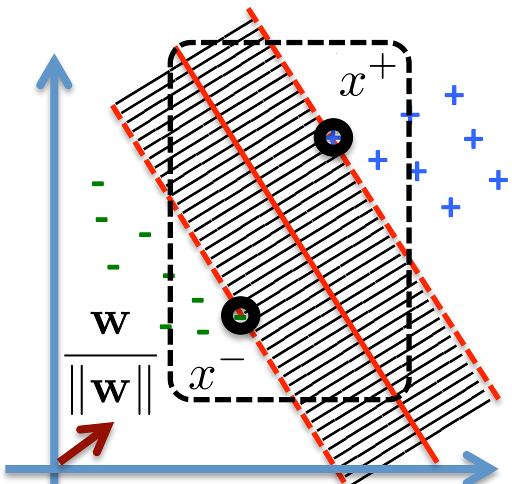
$$\mathbf{w}\cdot\mathbf{u}^{+}+\mathbf{b}\geq \boxed{1}$$
$$\mathbf{w}\cdot\mathbf{u}^{-}+\mathbf{b}\leq \boxed{-1}$$
$$y_i = +/-1$$

- Need a single function to optimize, introduce y_i

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

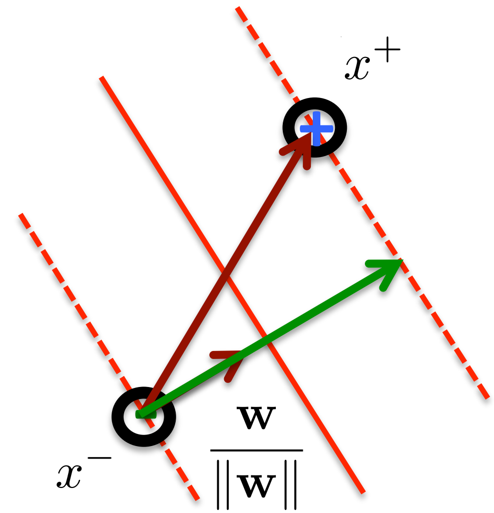
1. Some elements « on the margin » have to be exactly $+/-1$ (labeled x^+ and x^-)
2. In the street the values goes from [-1;1], everything else is < -1 or > 1
3. We can take the x^+ and x^- vectors (*not* perpendicular to the median)

Support Vector Machines



$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (x^+ - x^-) = ?$$

$$= Width\,!$$



- Dot product projects one on another and gives us the **width** of the street
- If we dot $\mathbf{w}$ with x^+ and with x^-, $\mathbf{b}$ drops if we substract

$$\left.\begin{aligned} \mathbf{w} \cdot \mathbf{x}^+ + \mathbf{b} &= 1 \\ \mathbf{w} \cdot \mathbf{x}^- + \mathbf{b} &= -1 \end{aligned}\right\} \quad \mathbf{w} \cdot (\mathbf{x}^+ - \mathbf{x}^-) = 2$$

$$\frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot (x^+ - x^-) = \underbrace{\frac{2}{\|\mathbf{w}\|}}_{\text{minimize}} = \underbrace{Width}_{\text{maximize}}$$

Support Vector Machines

- So the goal is to solve $min\ \|\mathbf{w}\|$ under $+/-1$ constraints
- Looks like a form of Lagrange multipliers
- However Lagrange use equalities (instead of inequalities)
- Hence we need a search that can be written as the **primal formulation**

$$min\left(P\left(\mathbf{w},b\right)\right)=\frac{\|\mathbf{w}\|}{2}+t.\sum_i\varepsilon_i$$

Tradeoff error / margin minimization

Minimize error

- If we go through the Lagrange formulation, we find (*cf. 2nd part of this course*)

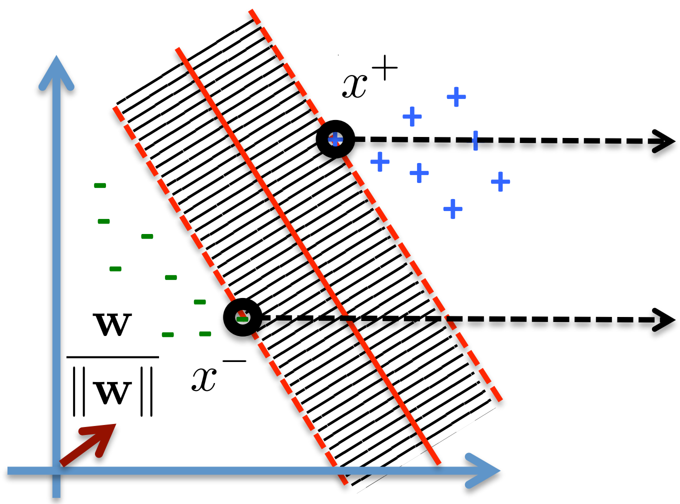
$$\mathbf{w}=\sum_i c_i\cdot x_i$$

Dual formulation (cf. end slides)

- So we can answer our problem, and all we need is
  - To optimize the c_i
  - Only requires the dot product

Support Vector Machines

- Based on our new formulation, a marvelous property appears $\mathbf{w} = \sum c_i \cdot x_i$
- Compared to neural nets, the space we are spanning is gloriously **convex**
- There is only **a global maximum**, no local ones
- You also discover that most c_i = 0 (so most points are useless)
- Only points on the fronteers *dictates* the value of $\mathbf{w}$



These points « support » the separation
=> Called **support vectors**

Support Vector Machines

- In order to obtain the complete problem to optimize
- We need to transform the problem to its **dual formulation**

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

- Under the constraints

$\sum_i \alpha_i y_i = 0$ and $\alpha_i \geq 0$

- So we have a final equation in the form of

$$\mathbf{w} = \sum_{i=0}^{n} \alpha_i y_i x_i$$

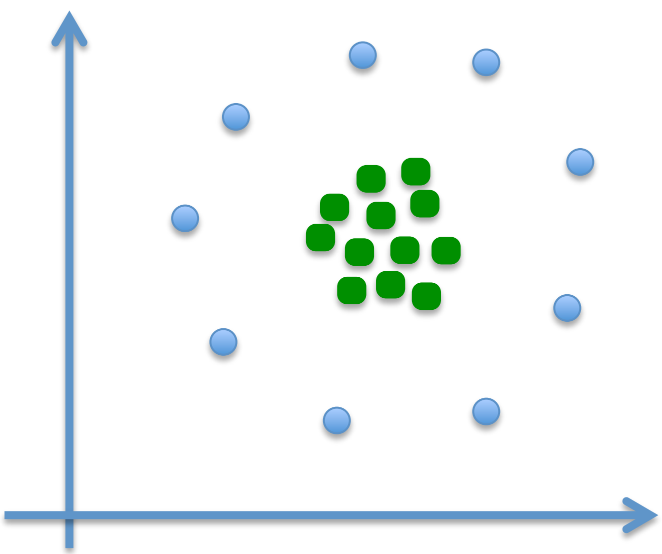
We need to optimize this

Coordinates in the space

Value of decision

Support Vector Machines

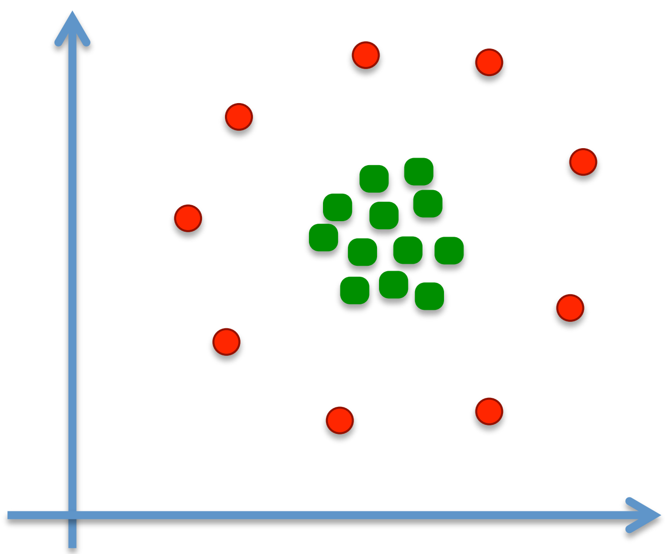
- Support Vector Machines (SVM) can find the **best straight line**
- And that is absolutely all that the SVM can do !...
- Unfortunately our world can rarely be separated by a straight line
- So **what to do when the data is non-linearly separable ?...**
- I am **stating** that the following groups can be separated by a single straight line

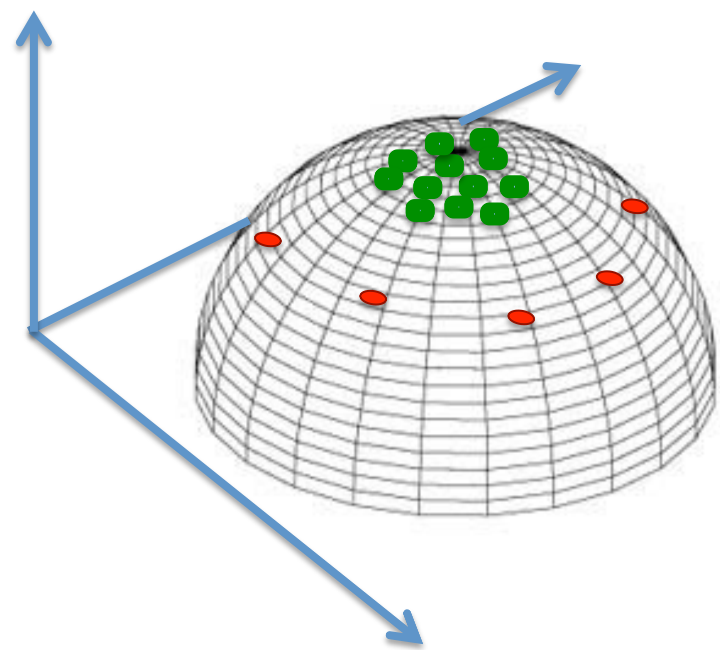


HOW ?

Support Vector Machines

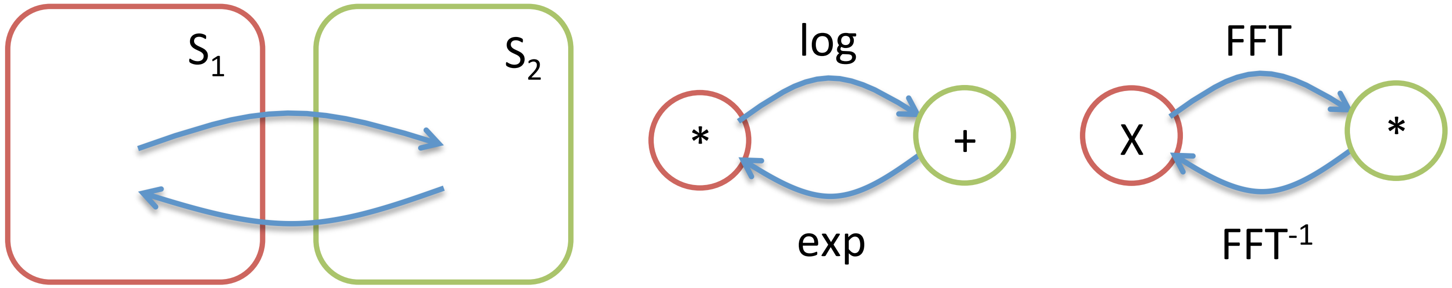
- Support Vector Machines (SVM) are find the **best straight line**
- And that is absolutely all that the SVM can do !...
- Unfortunately our world can rarely be separated by a straight line
- So **what to do when the data is non-linearly separable ?...**
- I am **stating** that the following groups can be separated by a single straight line



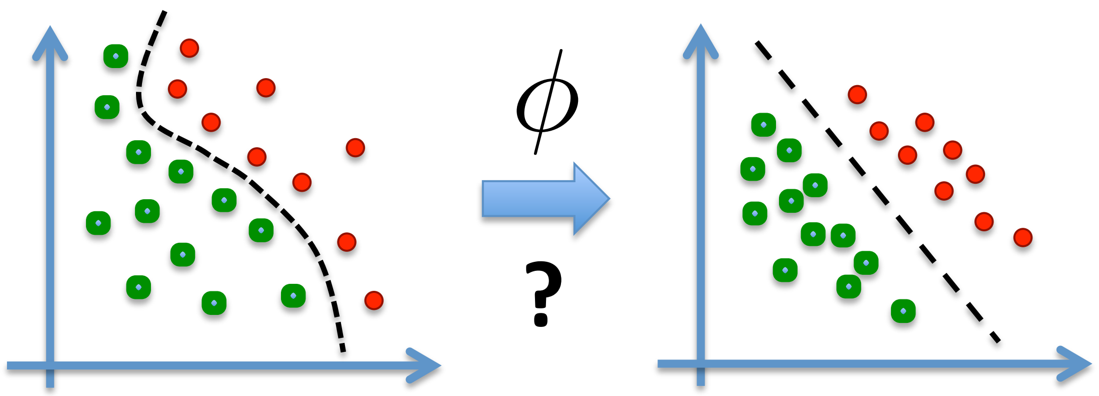


Support Vector Machines

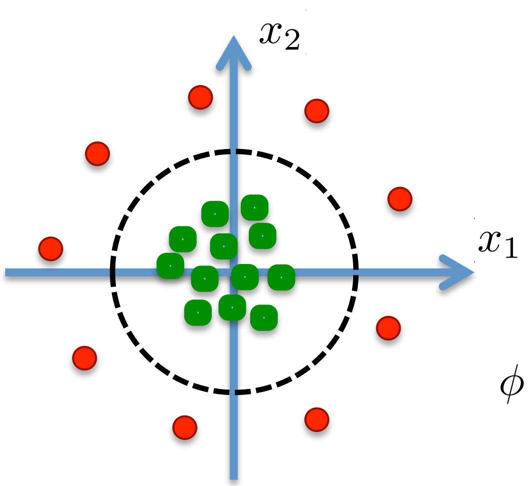
- If a problem is hard in a given space S_1
- It is usual to go to a second space S_2 where it is easier to solve
- And then apply the inverse transformation (going back from S_2 to S_1)



- Same here, if we don't have a straight line in S_1 maybe we have one in S_2

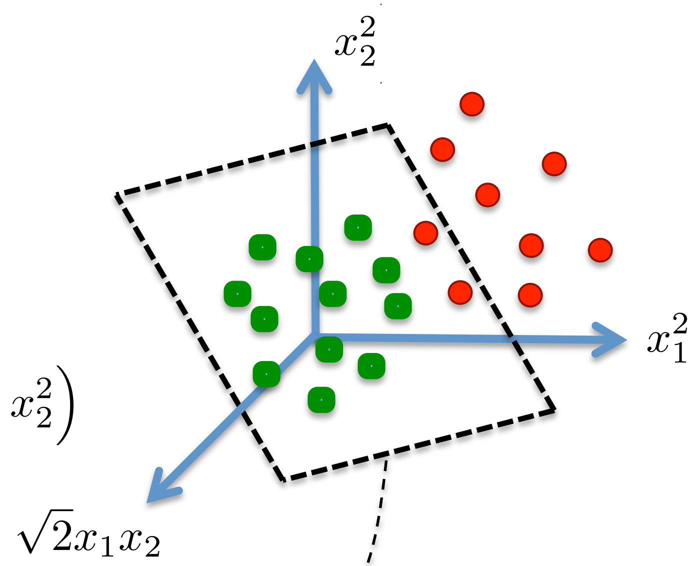


Support Vector Machines



$$\phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$\phi : (x_1, x_2) \to \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



Hyperplane defined by $\langle \mathbf{w} \cdot \mathbf{z} \rangle = 0$

$$w_1x_1^2 + w_2\sqrt{2}x_1x_2 + w_3x_2^2 = 0$$

Problem: The dimensionality of ϕ can be very large

- Vector $\mathbf{w}$ is hard to keep in memory with larger dimensions
- Quadratic programming is also hard to solve
- Instead of optimizing $\mathbf{w}$ directly, could we optimize only the α_i ?

Support Vector Machines

Reminder: We have been using the *primal formulation*

$$min\left(\frac{1}{2}\|\mathbf{w}\|^2\right) \quad \text{with constraint} \quad \forall i \;\; d_i\left(w\cdot x_i + w_0\right) \geq 1$$

- Both objectives are strictly convex
- Hence, we can express this as a Lagrangian

$$L\left(w, w_0, \alpha\right) = \frac{\|\mathbf{w}\|^2}{2} - \sum_i \alpha_i\left(d_i\left(w\cdot x_i + w_0\right) - 1\right)$$

- Remember that we want to maximize the margin, which means

$$\frac{\delta L}{\delta w_0} = 0 \Leftrightarrow \sum_i \alpha_i d_i \;= 0$$

$$\frac{\delta L}{\delta w} = 0 \Leftrightarrow w - \sum_i \alpha_i d_i x_i = 0 \Leftrightarrow w = \sum_i \alpha_i d_i x_i$$

- If we plug back this $\mathbf{w}$ into the primal formulation (as a Lagrangian)

$$L\left(w, w_0, \alpha\right) = \frac{1}{2}\underbrace{\left\langle \sum_i \alpha_i d_i x_i, \sum_j \alpha_j d_j x_j \right\rangle}_{} - \underbrace{\sum_i \alpha_i d_i\left(\sum_j \alpha_j d_j x_j\right) x_i}_{} - w_0 \underbrace{\sum_i \alpha_i d_i}_{=0} + \sum_i \alpha_i$$

Support Vector Machines

We obtain the **dual formulation** of the SVM optimization

$$L(w, w_0, \alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i d_i \alpha_j d_j \langle x_i, x_j \rangle$$

- Why should we care about the dual formulation?
- Remember that we want to perform a space transformation ϕ

Primal $$F(x) = \sum_i w_i \cdot \underbrace{\phi(x_i)}_{} + w_0$$

Need to actually do the transform ϕ

Dual $$F(x) = \sum_i \alpha_i - \frac{1}{2}\sum_i\sum_j \alpha_i d_i \alpha_j d_j \underbrace{\langle \phi(x_i), \phi(x_j) \rangle}_{}$$

There is no need to perform the transform ϕ

We just need to define the behavior of $K(x_i, x) = \phi(x_i) \cdot \phi(x_j)$

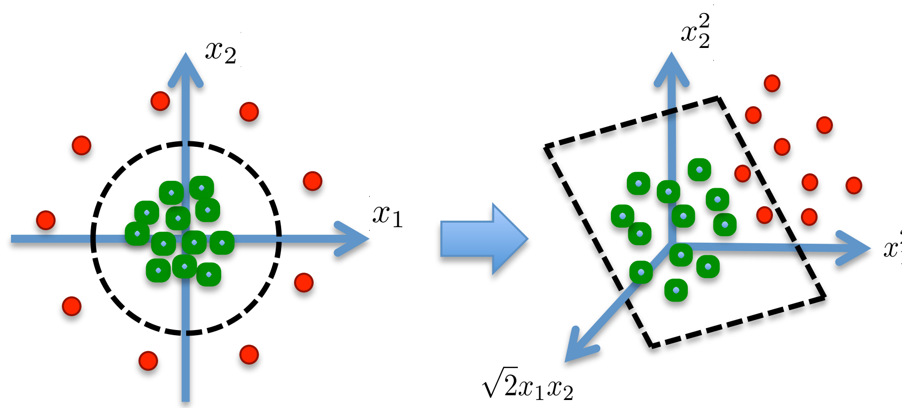
Math magic: We don't need to know what ϕ is but just what $\phi(x_i) \cdot \phi(x_j)$ does

Support Vector Machines

Remember our example transform ϕ ?

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$\phi : (x_1, x_2) \rightarrow \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right)$$



What happens if we just introduce

$$K(x, z) \quad = \quad \langle x, z \rangle^2$$

(the square of the dot product)

$$\begin{aligned}
K(x, z) &= \langle x, z \rangle^2 = (x_1 z_1 + x_2 z_2)^2 \\
&= \left(x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2\right) \\
&= \left\langle \left(x_1^2, \sqrt{2}x_1x_2, x_2^2\right) \cdot \left(z_1^2, \sqrt{2}z_1z_2, z_2^2\right) \right\rangle \\
&= \boxed{\langle \phi(x) \cdot \phi(z) \rangle}
\end{aligned}$$

Can be plugged directly into the dual formulation without even having to compute the transform ϕ

Dual $\quad F(x) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i d_i \alpha_j d_j \boxed{\langle \phi(x_i), \phi(x_j) \rangle}$

Support Vector Machines

Decision $$f\left(x\right)=\sum_{i}\alpha_i\phi\left(x_i\right)\cdot\phi\left(x_j\right)+b$$

Kernel function $$K\left(x_i,x\right)=\phi\left(x_i\right)\cdot\phi\left(x_j\right)$$

Some of the most used kernel functions are

Polynomial $$K\left(x,y\right)=\left(x^Ty+1\right)^d$$

Gaussian $$K(x,y)=exp(-\psi(x-y)^2)$$

Radial Basis $$K\left(x,y\right)=exp\left(-\|x-y\|^2/(2\sigma^2)\right)$$

Sigmoïd $$K\left(x,y\right)=tanh\left(kx^Ty+\Theta\right)$$