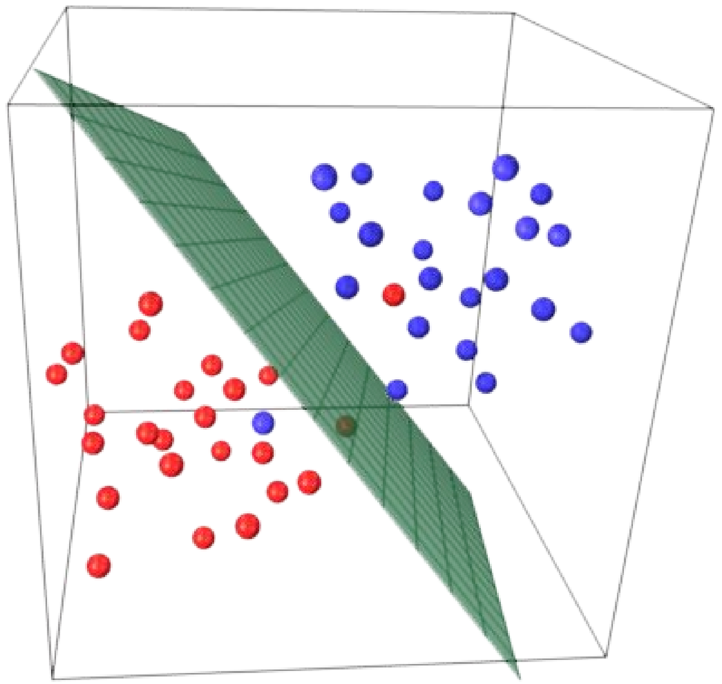
Support Vector Machines

DSI SEA, jf.omhover



Support Vector Machines

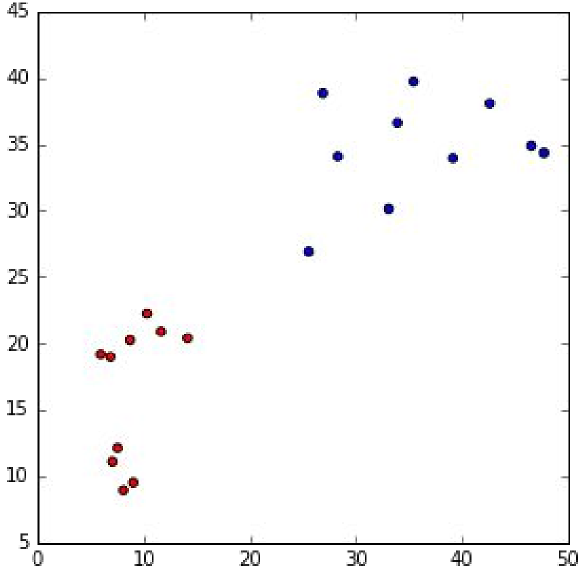
DSI SEA, jf.omhover

OBJECTIVES

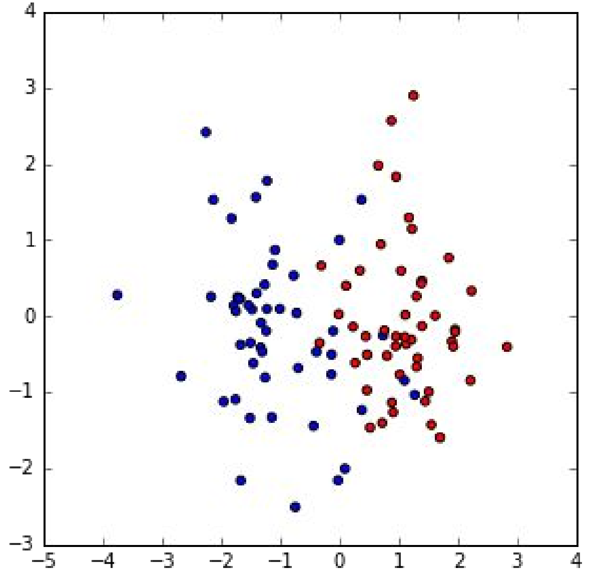
- **Understand** the notion of decision boundaries
- **Describe** the function and parameters of SVMs
- **Investigate** how SVMs perform in terms of Bias-Variance
- **Investigate** some of the maths behind SVMs
- Get your **mind blown**

Brainstorm : what's a good decision boundary ?

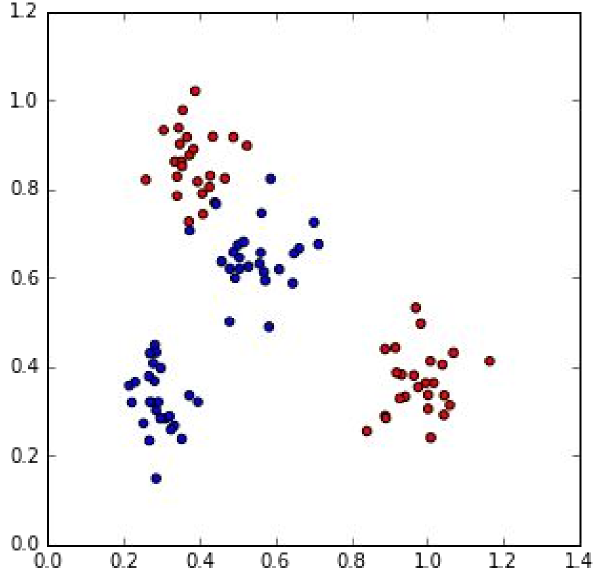
MMC ✓



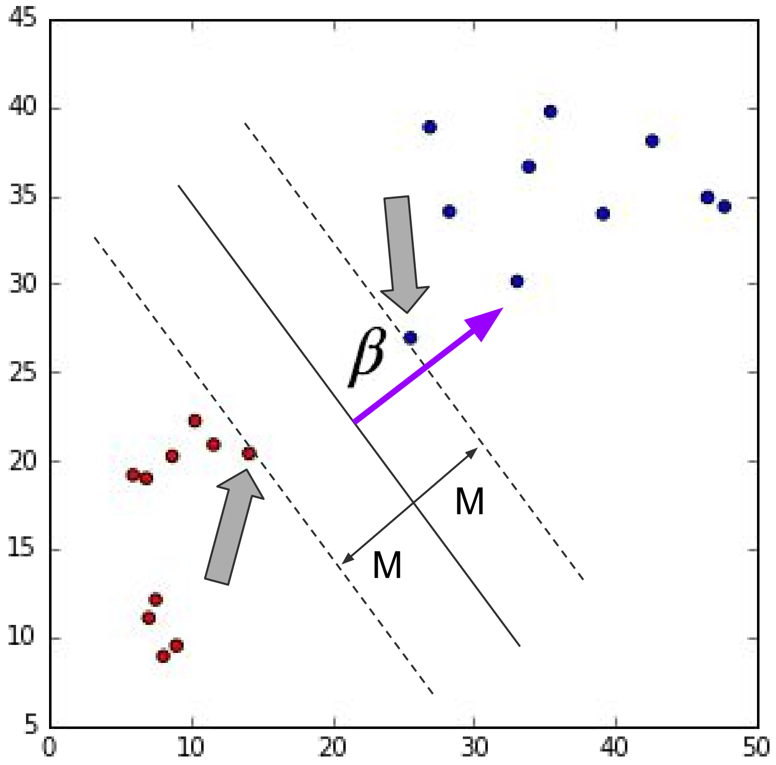
?



?



Maximum Margin Classification



$$max_{\beta_0,\ldots,\beta_p} M$$

subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i . (\beta_0 + \beta_1 . x_{i,1} + \cdots + +\beta_p . x_{i,p}) \geq M$$

$$y_i . (\beta_0 + x_i^T . \beta) \geq M$$

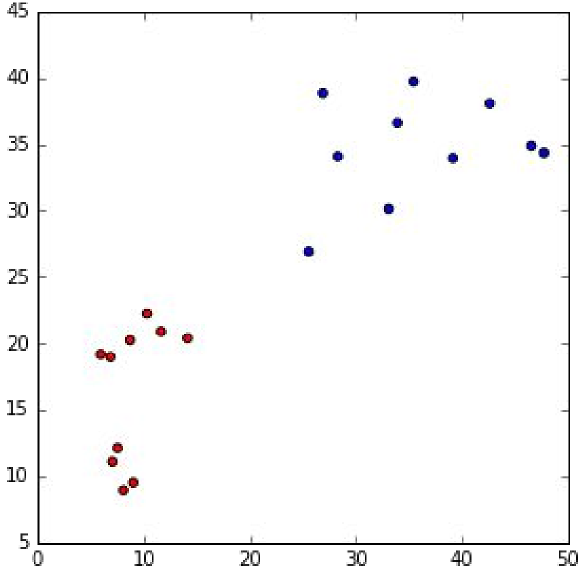
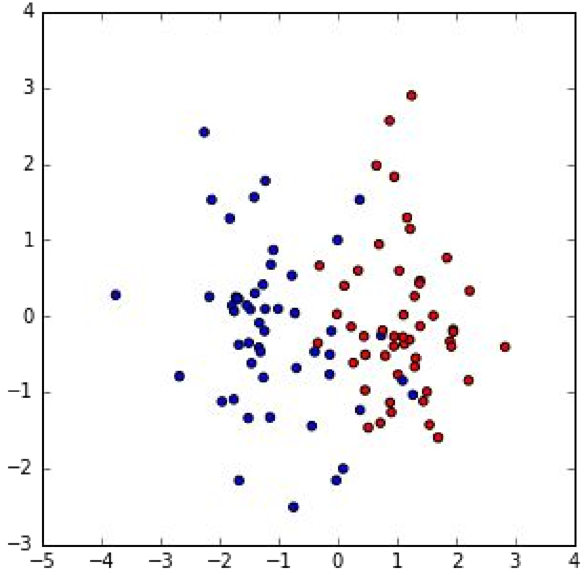
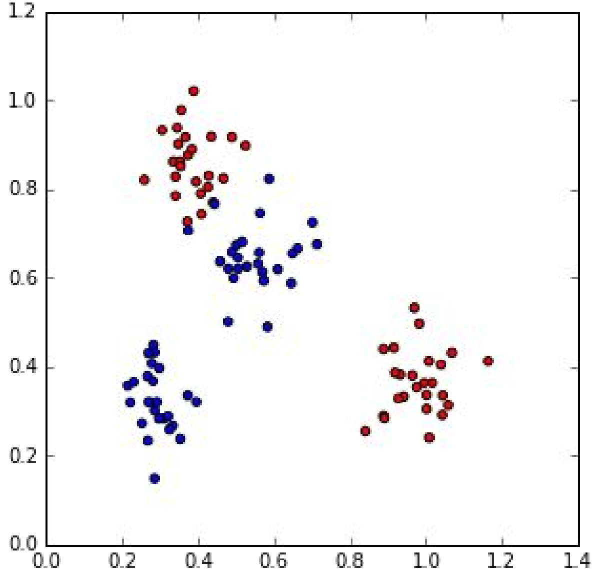
Points that condition the margin.
Points that have a direct influence on the margin.
Points that end up being the closest to the hyperplane.

Brainstorm : what’s a good decision boundary ?

MMC ✓

SVC ?

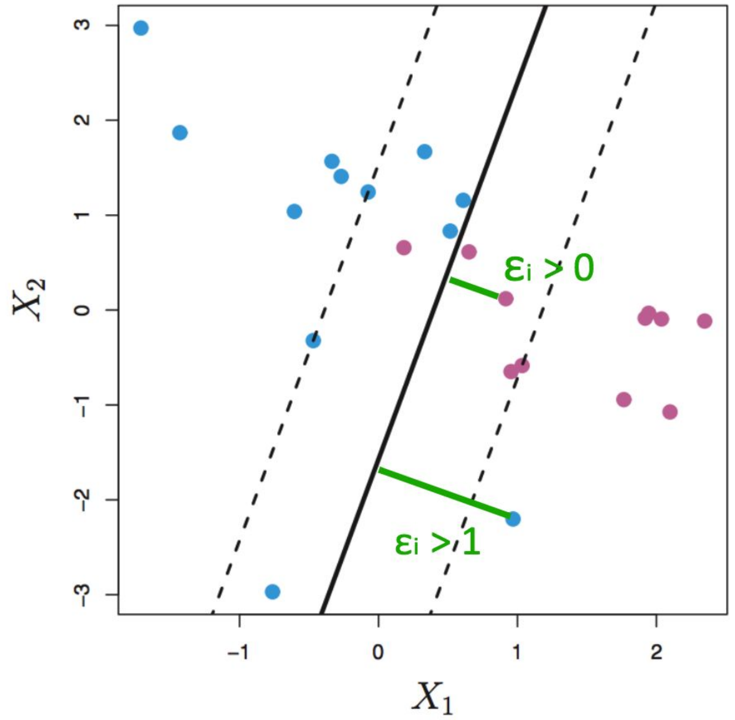
?

Support Vectors Classifier

introducing soft margins

Soft Margins / Support Vectors Classifier



Relax the constraints for a limited number of vectors

C : our “budget” to spend on relaxing this constraint

Each vector can expense ϵ_i on the margin (slack)

$$max_{\beta_0,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_p} M$$

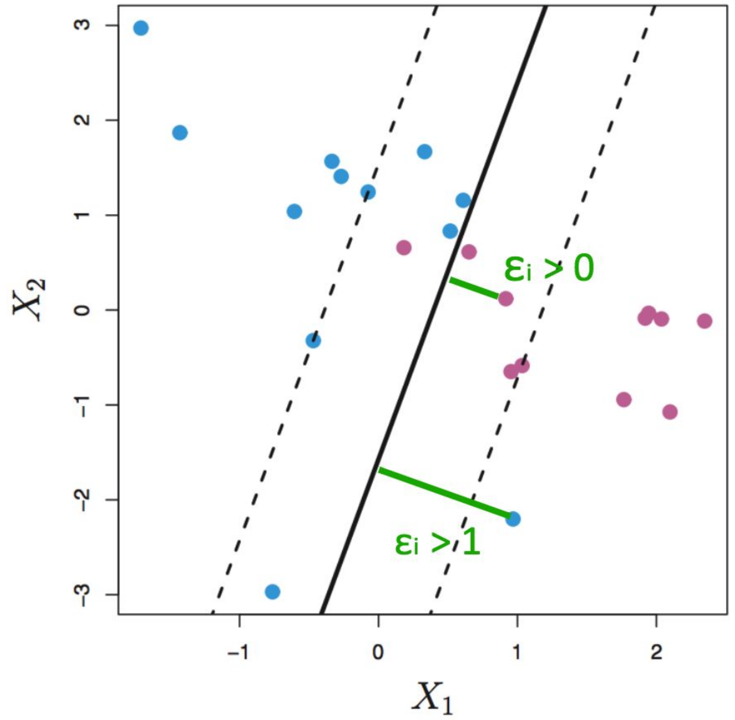
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$

$$y_i . (\beta_0 + \beta_1 . x_{i,1} + \cdots + +\beta_p . x_{i,p}) \geq M(1 - \epsilon_i)$$

$$y_i . (\beta_0 + x_i^T . \beta) \geq M(1 - \epsilon_i)$$

subject to $\forall i,\ \epsilon_i \geq 0$ and $\sum_{i=1}^{n} \epsilon_i \leq C$

How to solve that ?



Namedropping : Lagrange dual objective function

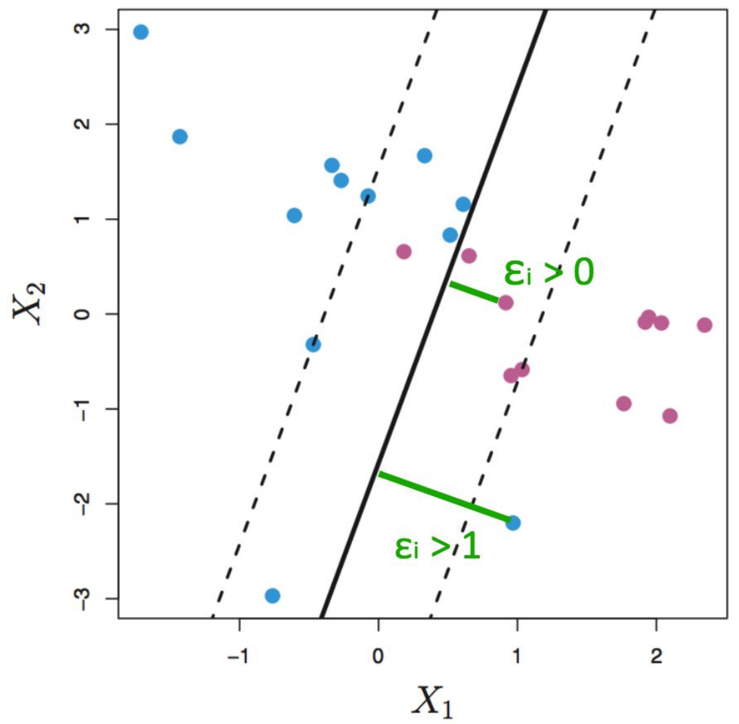
$$L_D = \sum_i^n \alpha_i - 1/2 \sum_i^n \sum_{i'}^n \alpha_i \alpha_{i'} y_i y_{i'} \; < x_i, x_i' >$$

$f_{\beta_0,\beta_1,\ldots,\beta_p}$ is a solution to that maximization problem

$$f_{\beta_0,\beta_1,\ldots,\beta_p}(x) = \beta_0 + \sum_{i=1}^p \alpha_i < x, x_i >$$

Alpha is non zero only for support vectors

Soft Margins / Support Vectors



“Support vectors” :
Only the observations that lie on the margin or violate it will affect the hyperplane

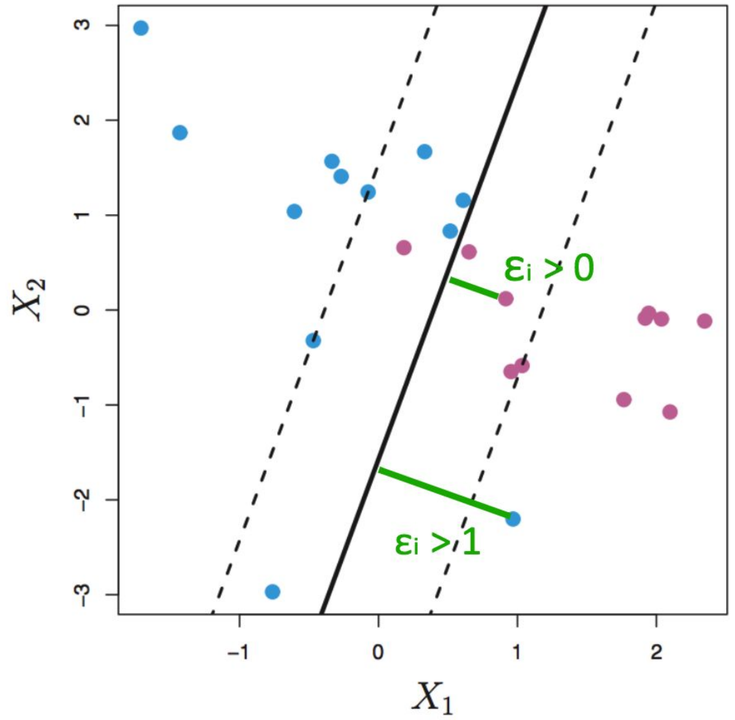
Large C : many support vectors
Small C : few support vectors

$f_{\beta_0,\beta_1,\dots,\beta_p}$ is a solution to that maximization problem

$$f_{\beta_0,\beta_1,\dots,\beta_p}(x) = \beta_0 + \sum_{i=1}^{p} \alpha_i < x, x_i >$$

Alpha is non zero only for support vectors

Soft Margins / SVC / Hyperparameter C



Relax the constraints for a limited number of vectors

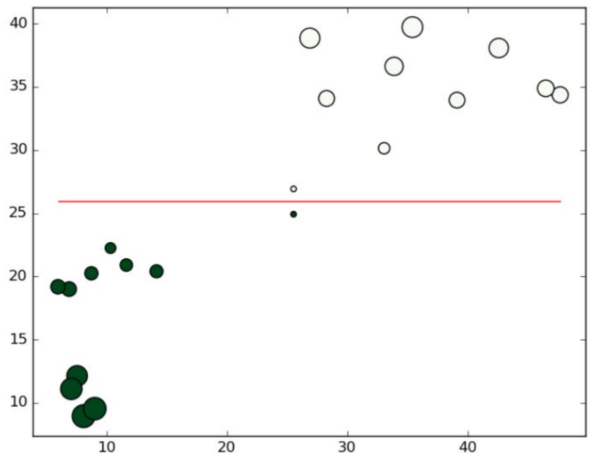
C : our “budget” to spend on relaxing this constraint

Each vector can expense ϵ_i on the margin (slack)

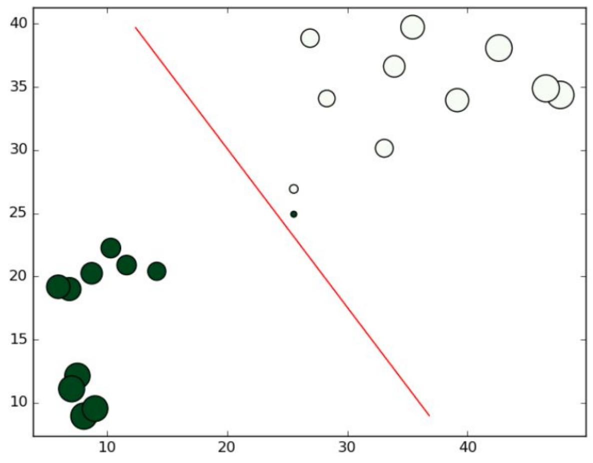
Small C : narrow margins, highly fit, low bias, high variance

Large C : wider margins, fitting less hard, high bias, low variance

Soft Margins / Hard Margins



Hard Margin

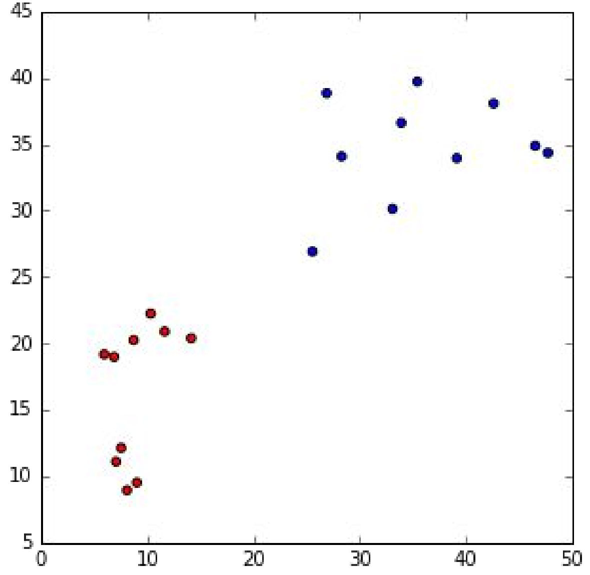


Soft Margin

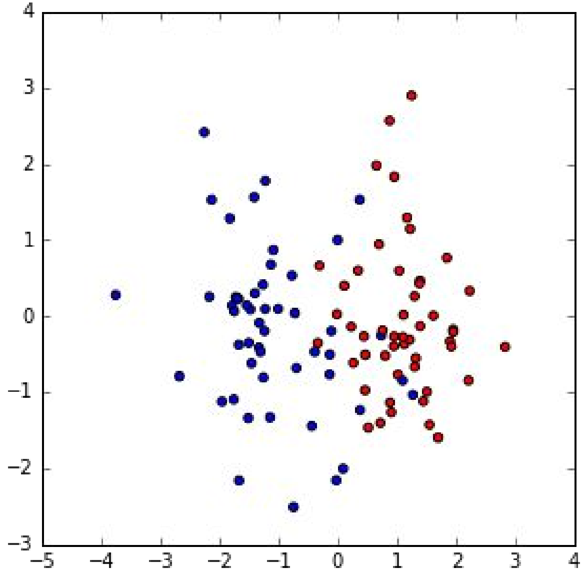
Figure from R. Henning

Brainstorm : what’s a good decision boundary ?

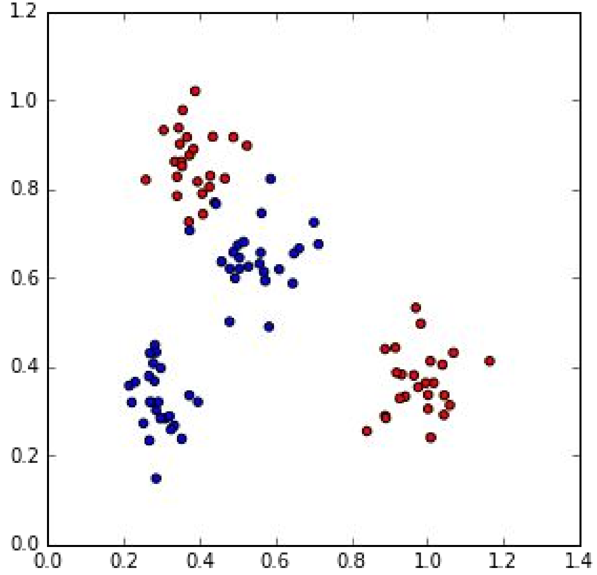
MMC ✓



SVC ✓



?

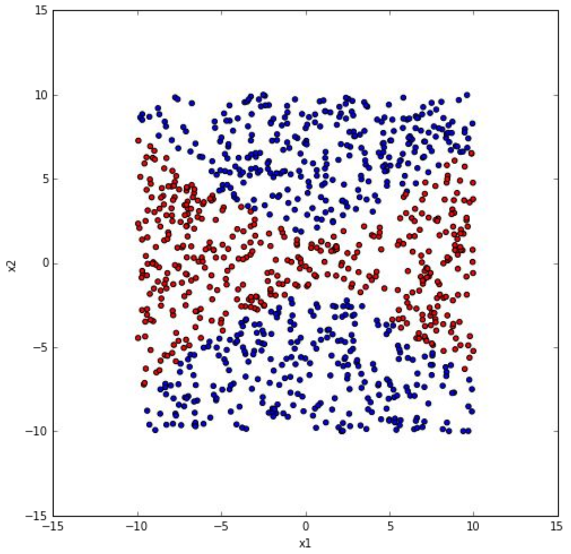


Support Vectors Machines

introducing kernels

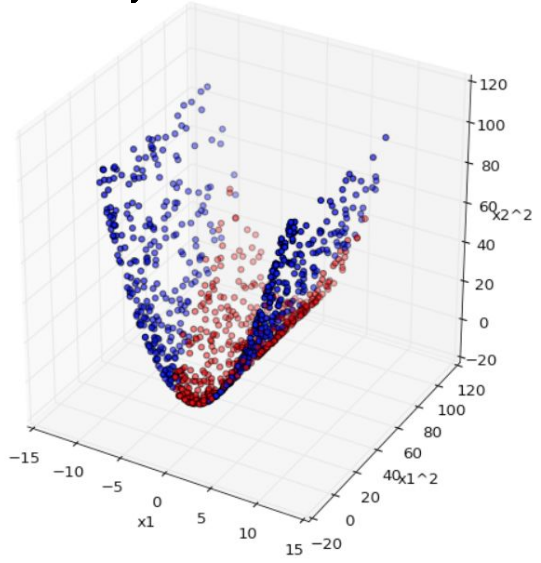
Lyrics of the Kernel Trick song

Life is not linearly separable...



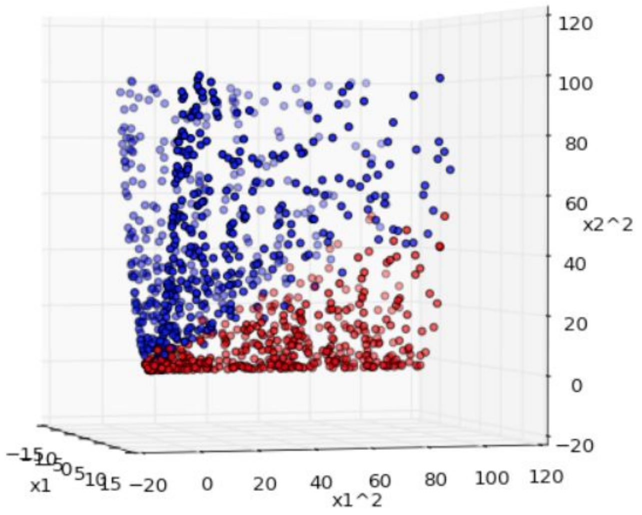
(x1,x2)

But maybe by thinking outside your dimensions...



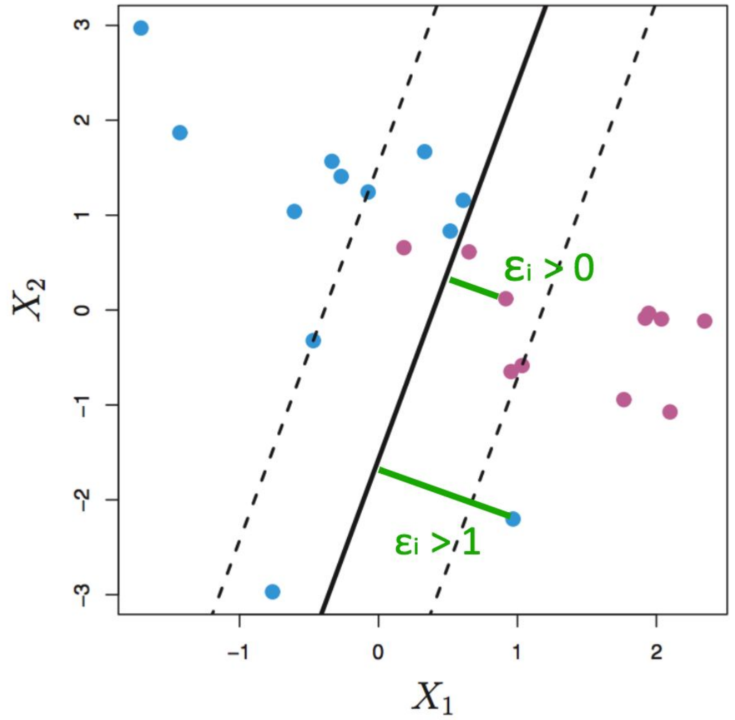
(x1,x2, x1^2, x2^2)

It is...



(x1,x2, x1^2, x2^2) > 0

Sol 1: Expending in the polynomial realm...



Adding to the input space...

$$max_{\beta_0,\cdots,\beta_p,,\beta_{2,1},\cdots,\beta_{2,p},\epsilon_1,\cdots,\epsilon_p} M$$

subject to $\sum_{j=1}^{p} \beta_j^2 + \sum_{j=1}^{p} \beta_{2,j}^2 = 1$

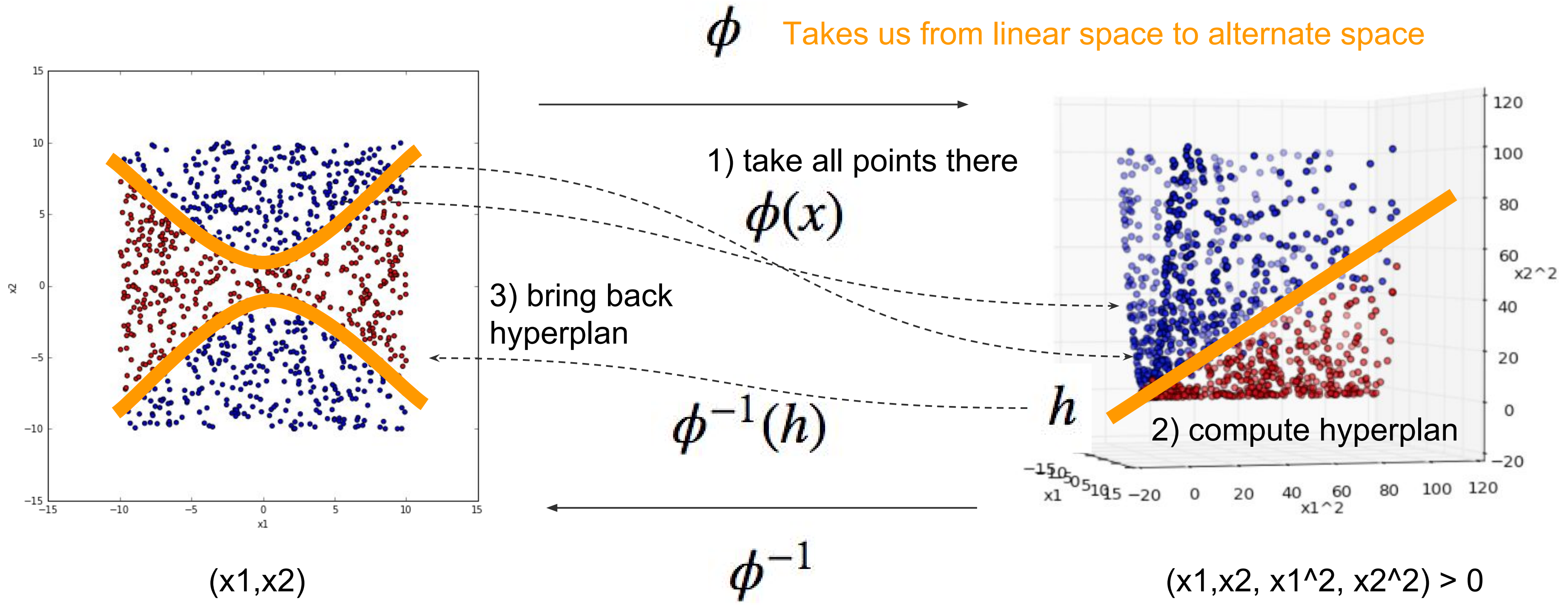
$$y_i.(\beta_0 + \sum_{j=1}^{p} \beta_j.x_{i,j} + \sum_{j=1}^{p} \beta_2,j.x_{i,j}^2) \geq M(1-\epsilon_i)$$

$$y_i.(\beta_0 + x_i^T.\beta) \geq M(1-\epsilon_i)$$

subject to $\forall i,\ \epsilon_i \geq 0$ and $\sum_{i=1}^{n} \epsilon_i \leq C$

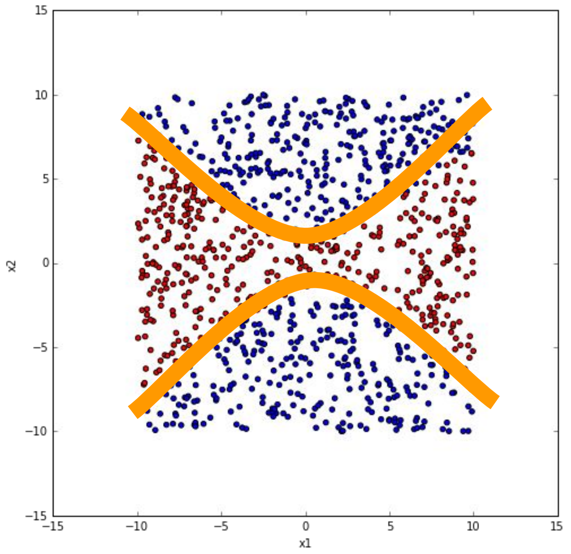
Sol 2: Let’s devise an alternate space

ϕ Takes us from linear space to alternate space



Takes us back

How to solve that ?



Namedropping : Lagrange dual objective function

$$L_D = \sum_i^n \alpha_i - 1/2 \sum_i^n \sum_{i'}^n \alpha_i \alpha_{i'} y_i y_{i'} K(x_i, x_i')$$

$$K(x, x_i) = < \phi(x), \phi(x_i) >$$

$f_{\beta_0,\beta_1,\dots,\beta_p}$ is a solution to that maximization problem

$$f_{\beta_0,\beta_1,\dots,\beta_p}(x) = \beta_0 + \sum_{i=1}^p \alpha_i < \phi(x), \phi(x_i) >$$

Alpha is non zero only for support vectors

Kernels we use...

Solution to SVC only involves inner product of observations

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle$$ ← SVC

$$f(x) = \beta_0 + \sum_{\boxed{i \in \mathcal{S}}} \alpha_i \langle x, x_i \rangle$$ ← Only requires support vectors

More generally, instead of just taking inner product, we can use *Kernels*

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$ ← SVM, since using Kernels now

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$$ Linear Kernel

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$ Polynomial Kernel

$$K(x_i, x_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2\right)$$ Radial Basis Function Kernel (“Gaussian”)

Ref Tammy Lee

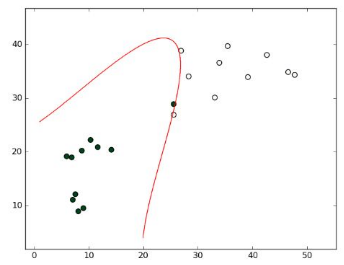
Polynomial Kernel

$$K(x^{(i)}, x^{(j)}) = (1 + x^{(i)} \cdot x^{(j)})^d$$

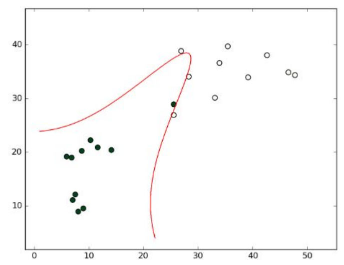
- equivalent to the dot product in the d-order ϕ space
- requires an extra hyper-parameter, d, for “degree”

Ref Ryan Henning

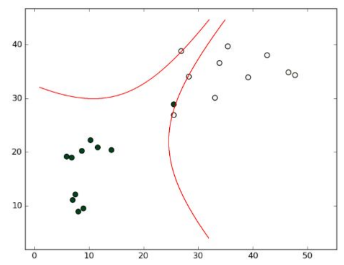
Polynomial Kernel



svc = SVC(C=10000.0, kernel='poly', degree=3)



svc = SVC(C=10000.0, kernel='poly', degree=5)



svc = SVC(C=10000.0, kernel='poly', degree=10)

Ref Ryan Henning

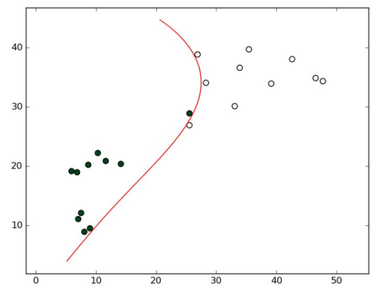
RBF Kernel (Radial Basis Function)

$$K(x^{(i)}, x^{(j)}) = \exp\left(-\gamma ||x^{(i)} - x^{(j)}||^2\right)$$

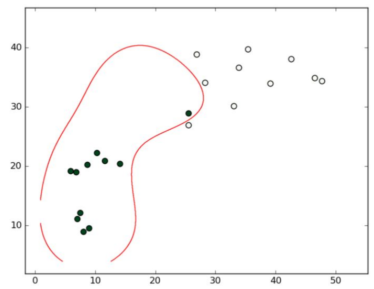
- equivalent to the dot product in the Hilbert space of infinite dimensions
- requires an extra hyper-parameter, γ, "gamma"

Ref Ryan Henning

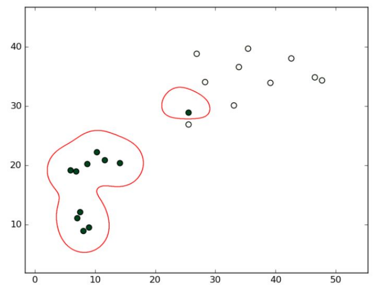
RBF Kernel (Radial Basis Function)



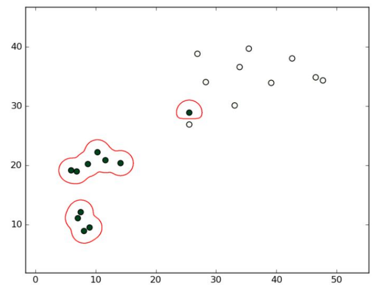
svc = SVC(C=10000.0, kernel='rbf', gamma=0.001)



svc = SVC(C=10000.0, kernel='rbf', gamma=0.01)



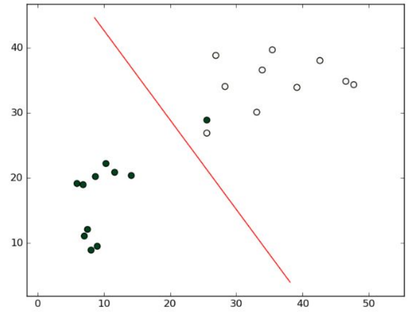
svc = SVC(C=10000.0, kernel='rbf', gamma=0.1)



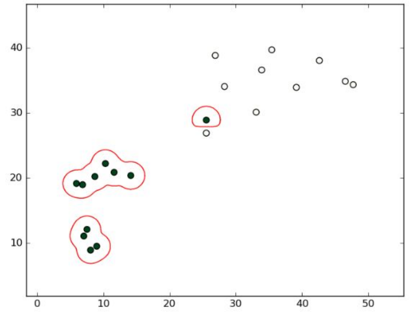
svc = SVC(C=10000.0, kernel='rbf', gamma=1.0)

Ref Ryan Henning

Best fit ?



svc = SVC(C=0.01, kernel="linear")



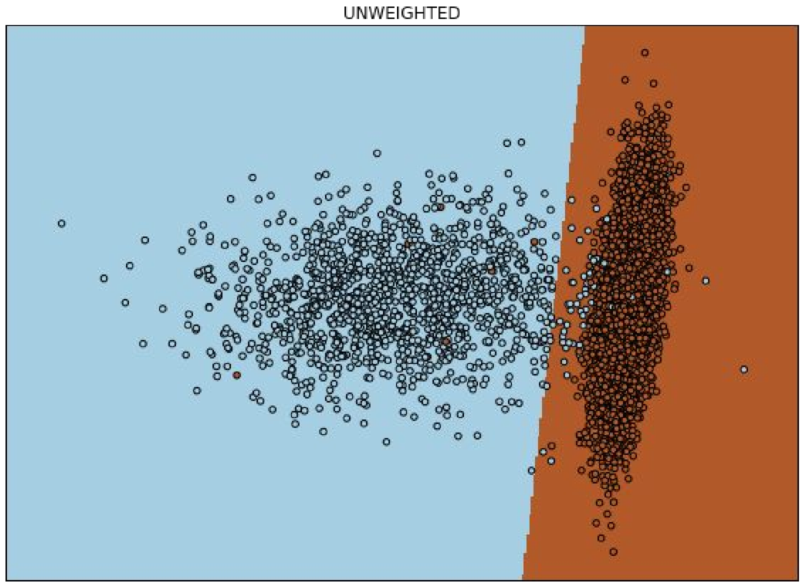
svc = SVC(C=10000.0, kernel='rbf', gamma=1.0)

Ref Ryan Henning

Adaptation of SVM to real-world problems

Unbalanced classes

Adjusting weights (beta)
so that they are inversely proportional
to class representativity



Multiple classes

One-versus-One :

Train a classifier on all pairs of classes

Use each pair for determining the class of an observation (aggregating)

One-versus-All / One-versus-Rest :

Train a classifier on each class (+1), considering all other as the rest (-1)

Use this classifier for determining if an observation fits in that class

Pair Assignment

Snippets...

```
from sklearn.svm import SVC

svm = SVC(kernel='linear').fit(X, y)
svm = SVC(kernel='linear', C=x).fit(X,y)
svm = SVC(kernel='poly', C=c, degree=5).fit(X,y)
svm = SVC(kernel='rbf', C=c, gamma=0.5).fit(X,y)
```