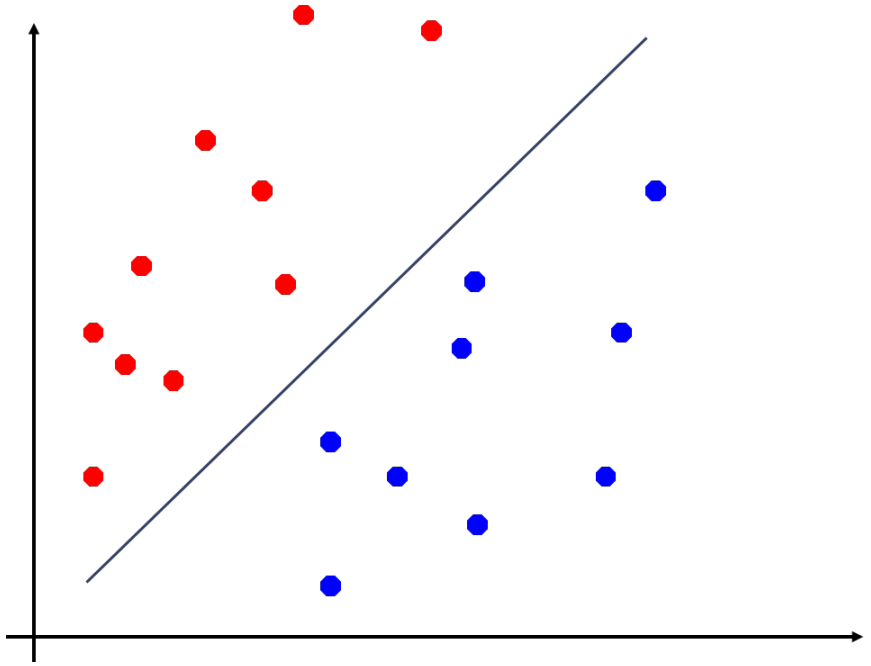
Beginner-friendly ML school:

Support Vector Machines and Hyperparameter fitting

Viviana Acquaviva
(CUNY)
vacquaviva@citytech.cuny.edu

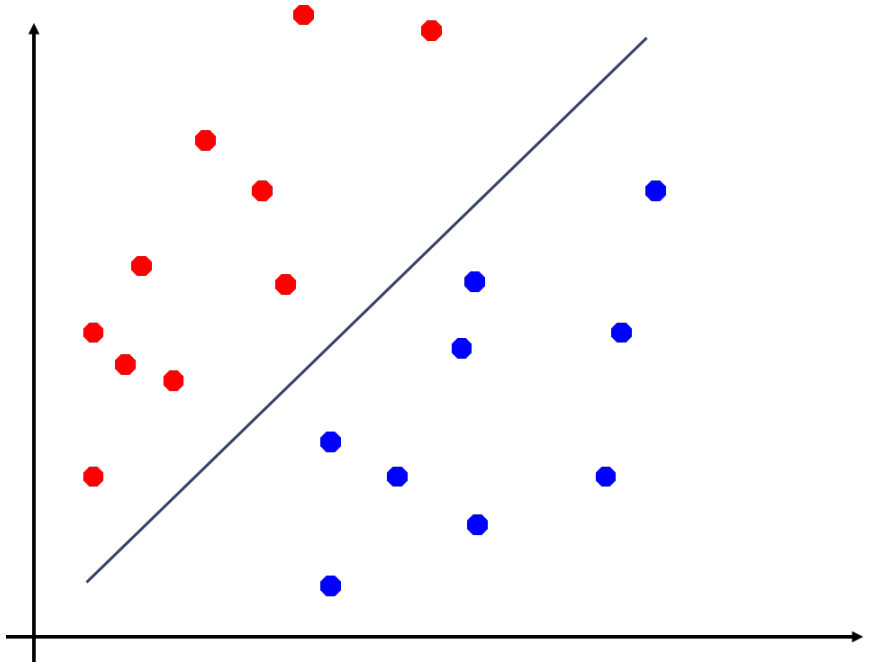
Support Vector Machines (SVM)



(was) one of the most popular classification algorithms.

Pros: Powerful, accurate Cons: Slow

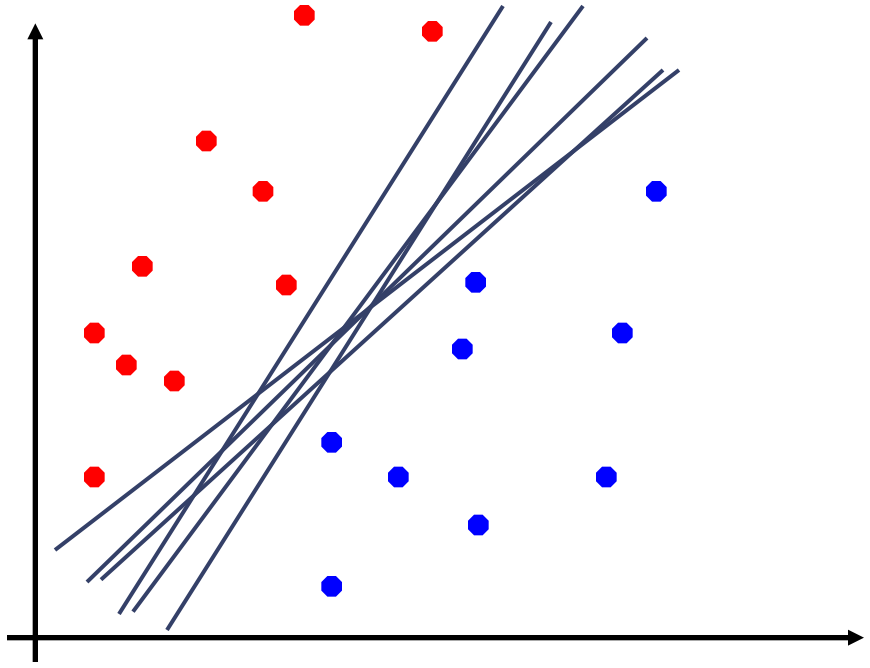
Support Vector Machines (SVM)



How can we split the two classes?

Linear SVMs

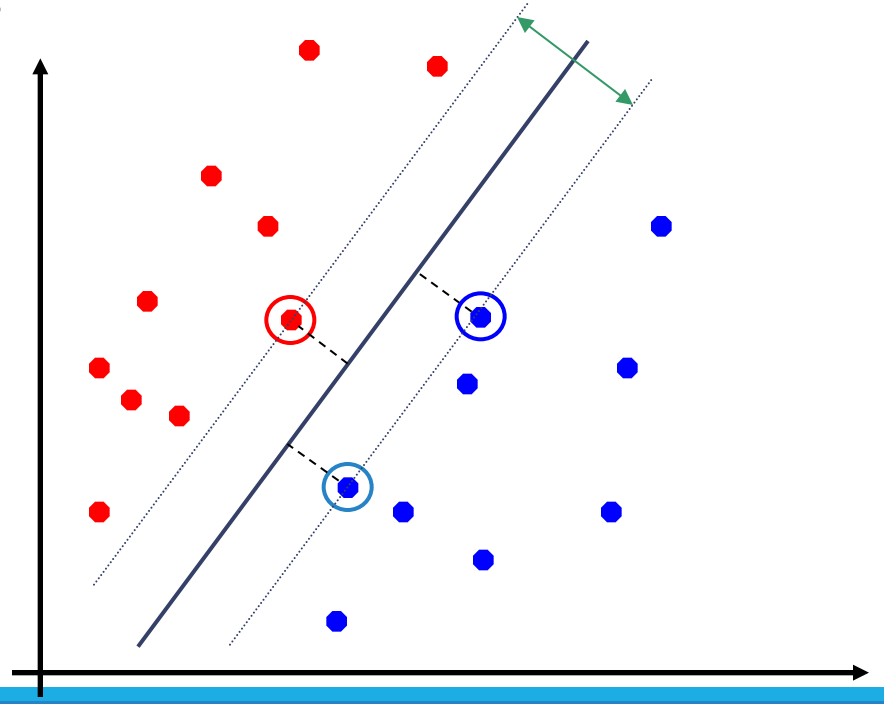
Which of the linear separators is optimal?



Idea 1. If the data are separable, find the decision boundary that maximizes separation between the two classes

Examples closest to the hyperplane are ***support vectors***.

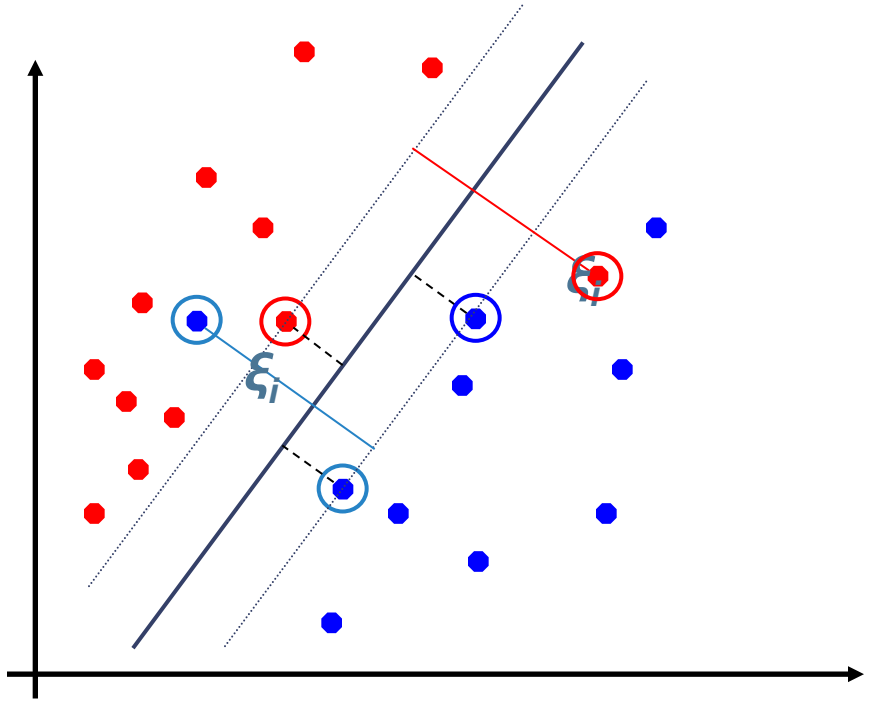
Margin of the separator is the distance between support vectors of opposite classes.



Idea 2. Add slack variables that allow for misclassifications

What if the training set is not linearly separable?

Slack variables can be added to allow misclassification of difficult or noisy examples close to the boundary, while keeping separation large **(resulting in a "soft" margin).**

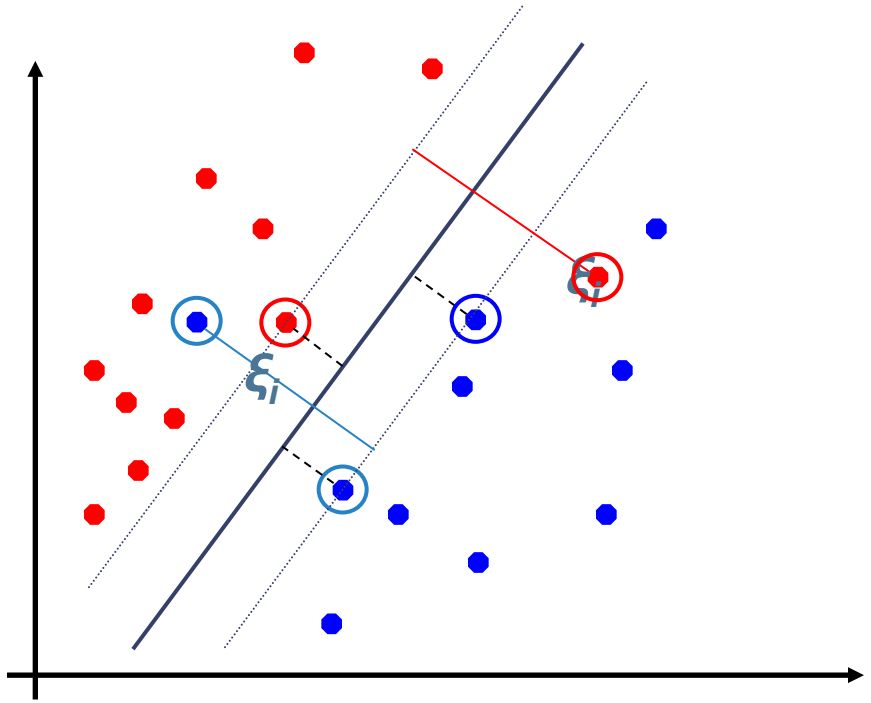


Idea 2. Add slack variables ξ_i that allow for misclassifications

What if the training set is not linearly separable?

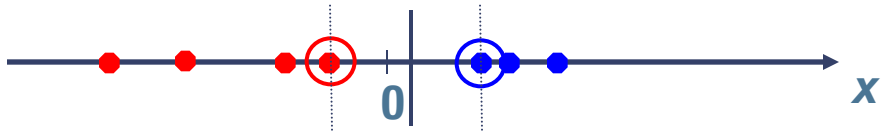
The problem becomes a minimization of

1/(width of margin) + C $\sum_i \xi_i$



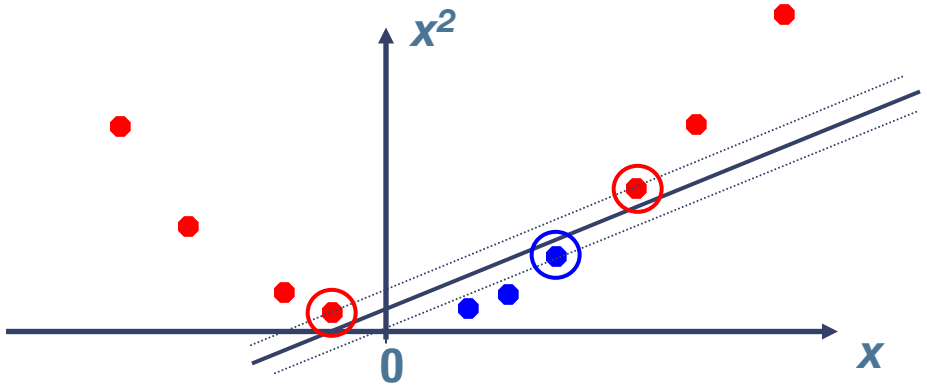
Non-linear SVMs

Datasets that are linearly separable, even with some noise, work out great:



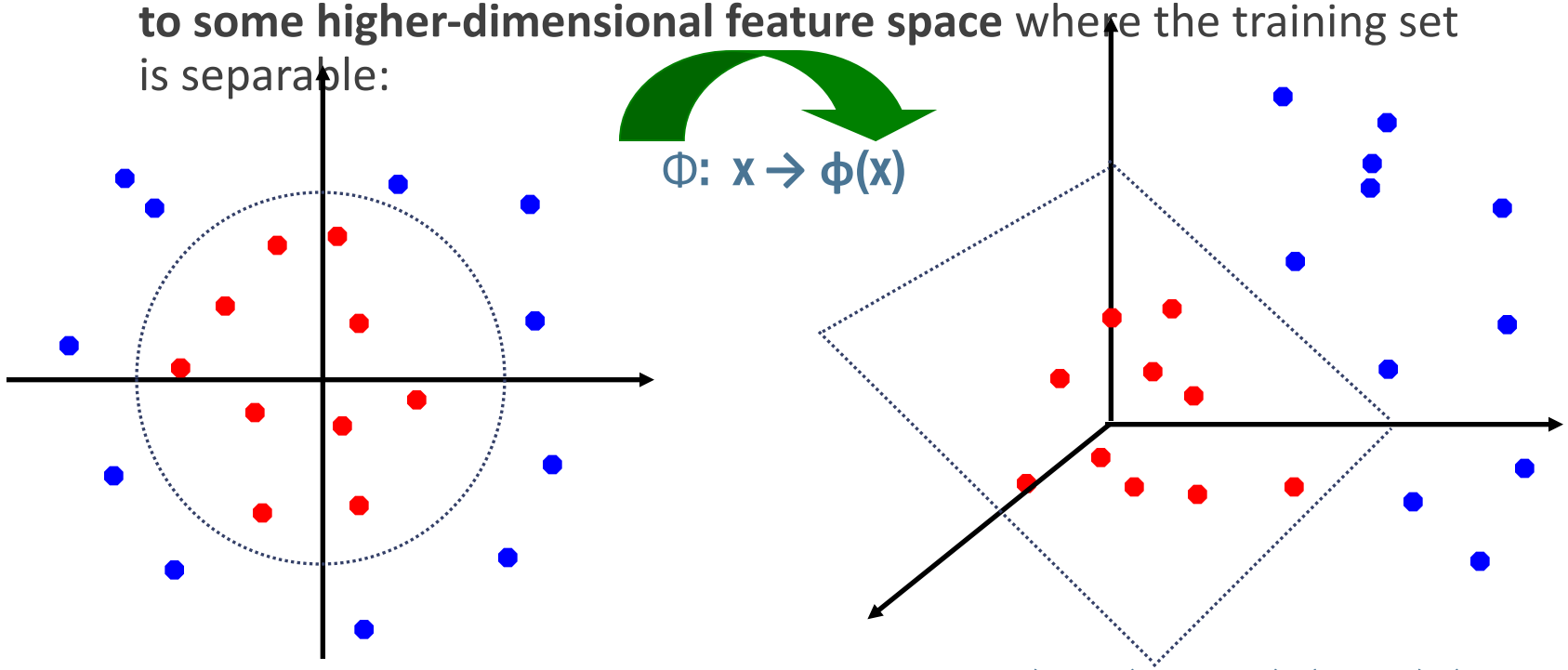
But what are we going to do if the dataset cannot be separated with a line?

How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces (kernel trick)

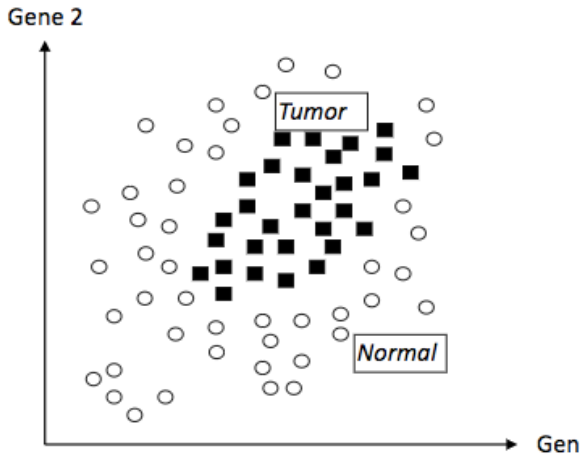
General idea: the original feature space **can always be mapped to some higher-dimensional feature space** where the training set is separable:



The mapping function defines a kernel $K(x, z) = \phi(x)^T \phi(z)$.

Often linear, polynomial, or Gaussian.

Example: Gaussian (RBF) Kernel



Data is not linearly separable in the input space

Data is linearly separable in the feature space obtained by a kernel

Magic: You don't need to specify or calculate the mapping function, only the kernel, because it is possible to re-write the decision function as a function of $K(x, x_j)$ and show that this depends on the inner product $x \cdot x_j$, where x_j are the support vectors. Additionally, that's O(n) to boot

In fact: the math of SVM is beautiful
(this is a tiny preview of how I understand it)

Given boundary equation $w x + b = 0$:

- rescale w and b so that the edges of margin have equations $w x + b = -1$; $w x + b = 1$

- for all correctly classified examples, $y \cdot (wx + b) > 0$ and the decision function is $g(w x + b) > 1$ or $g(w x + b) < -1$, g is step function

- maximize $1/||\ w\ ||_2$ with constraint $y \cdot (wx + b) > 0$ an all training data (Quadratic Programming)

- To solve efficiently: turn this constrained maximization problem into a Lagrange multiplier formulation (primal/dual), allows one to write decision function as a function of inner product of example x with training examples xj

- Non separable case: add slack variables (for which $g(w x + b) < -1$ does not hold) and maximize $1/||\ w\ ||_2 + C \sum i\ \xi i$

- To solve: find new mapping $x \rightarrow \Phi(x)$ to higher dimensional space where instances are more separated

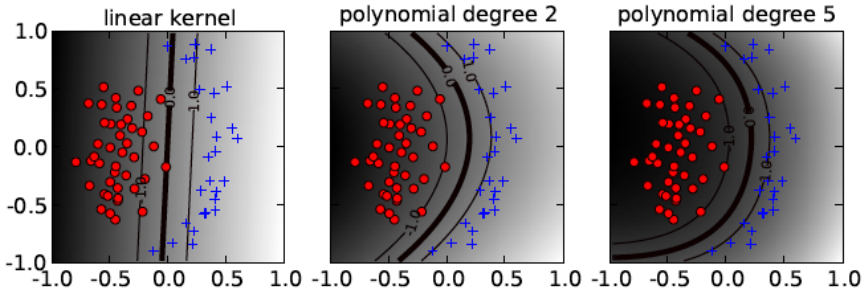
- Kernel trick (Mercer theorem): it can be shown that if K is semipositive definite, inner product of $\Phi(xi)\ \Phi\ (xj)$ is prop to $K(xi, xj)$ and its complexity is O(n). Gives decision function in mapping space without explicit mapping.

But if you really want to see it, see Andrew Ng's notes

http://cs229.stanford.edu/notes/cs229-notes3.pdf

Hyperparameters of SVMs

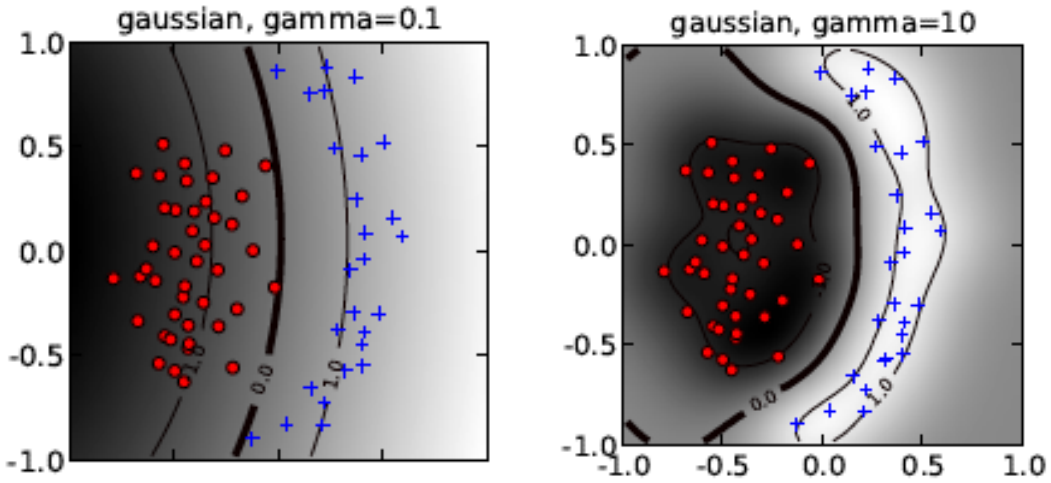
(hyperparameters = parameters of the model)



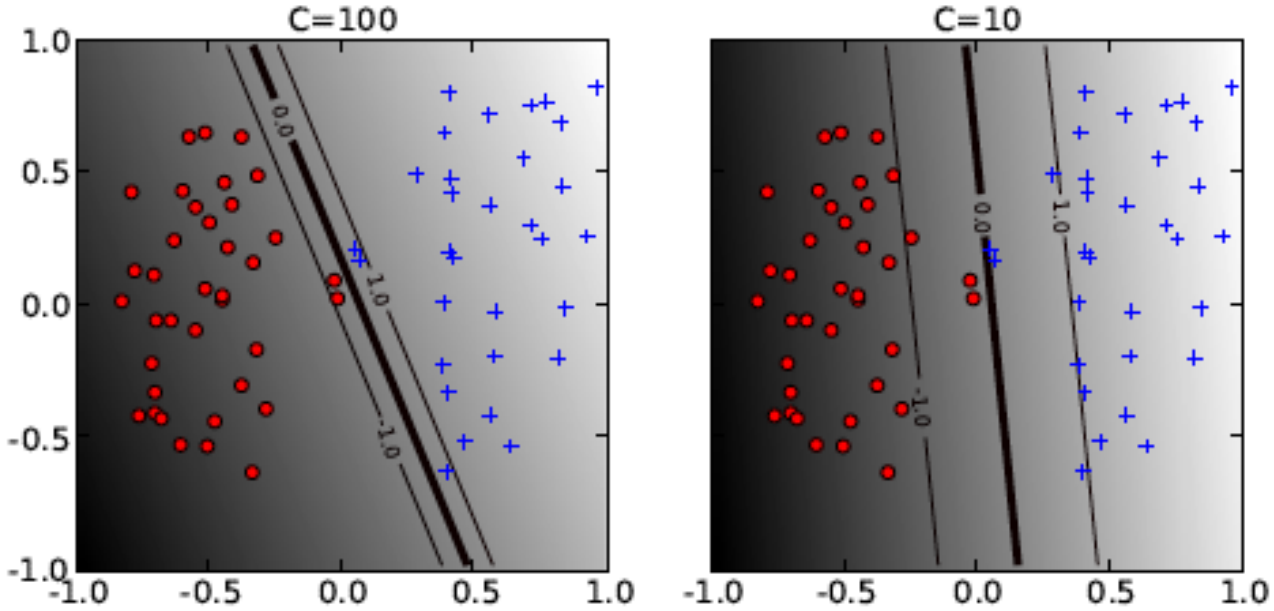
1) Type of kernel (functional form of function describing the boundary).

A linear kernel is a line or plane. A polynomial kernel corresponds to a more curvy boundary.

A common choice is a Gaussian kernel, called rbf (radial basis function).



2) Gamma (shape factor of Gaussian kernel boundary). Small gamma is more linear, high gamma is more wiggly.

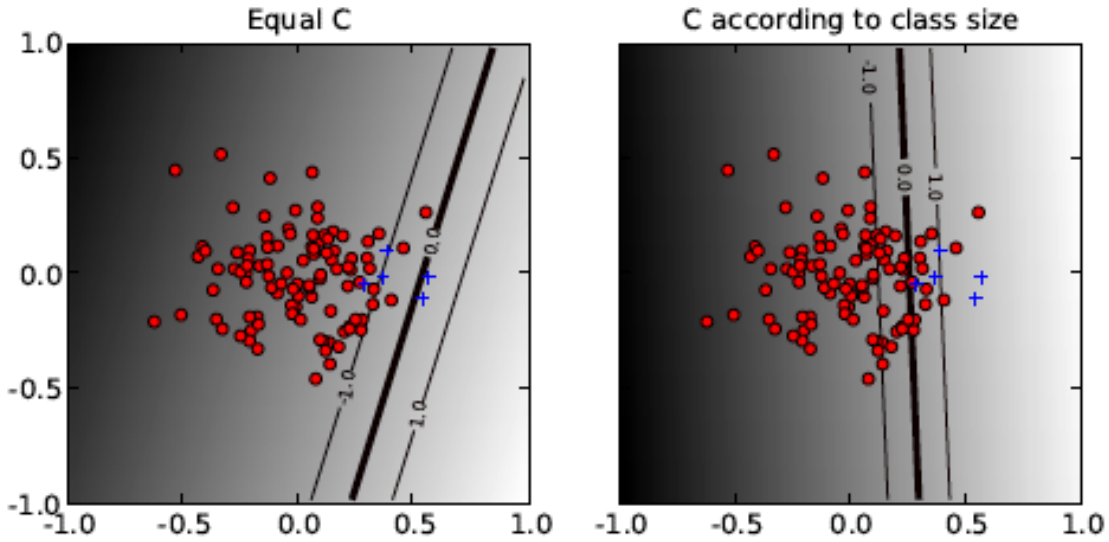


3) C = Penalty function for misclassifications in soft margin SVM.

Large C means that misclassifications near the boundary pay a large penalty, and drives the separation between classes to be small.

Small C means that misclassifications near the boundary pay a small penalty, and creates a larger separation (margin) between classes.

A cool hyperparameter: class weight



4) Class weight

It gives a different penalty for each class, useful for unbalanced data or when we care about one type of error (e.g. false positive) more than the other