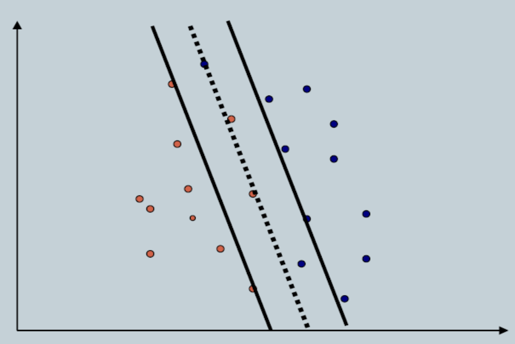
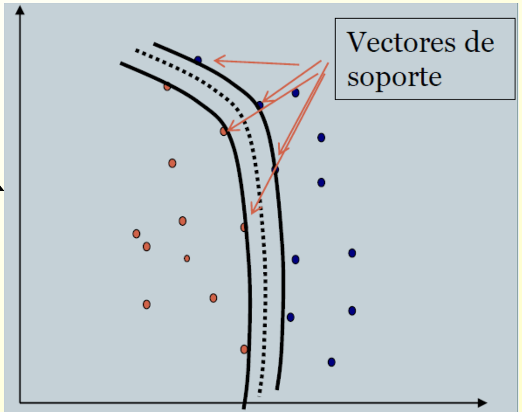
Support Vector Machines

SVMs: Support Vector Machines

Support Vector Machines (SVMs)

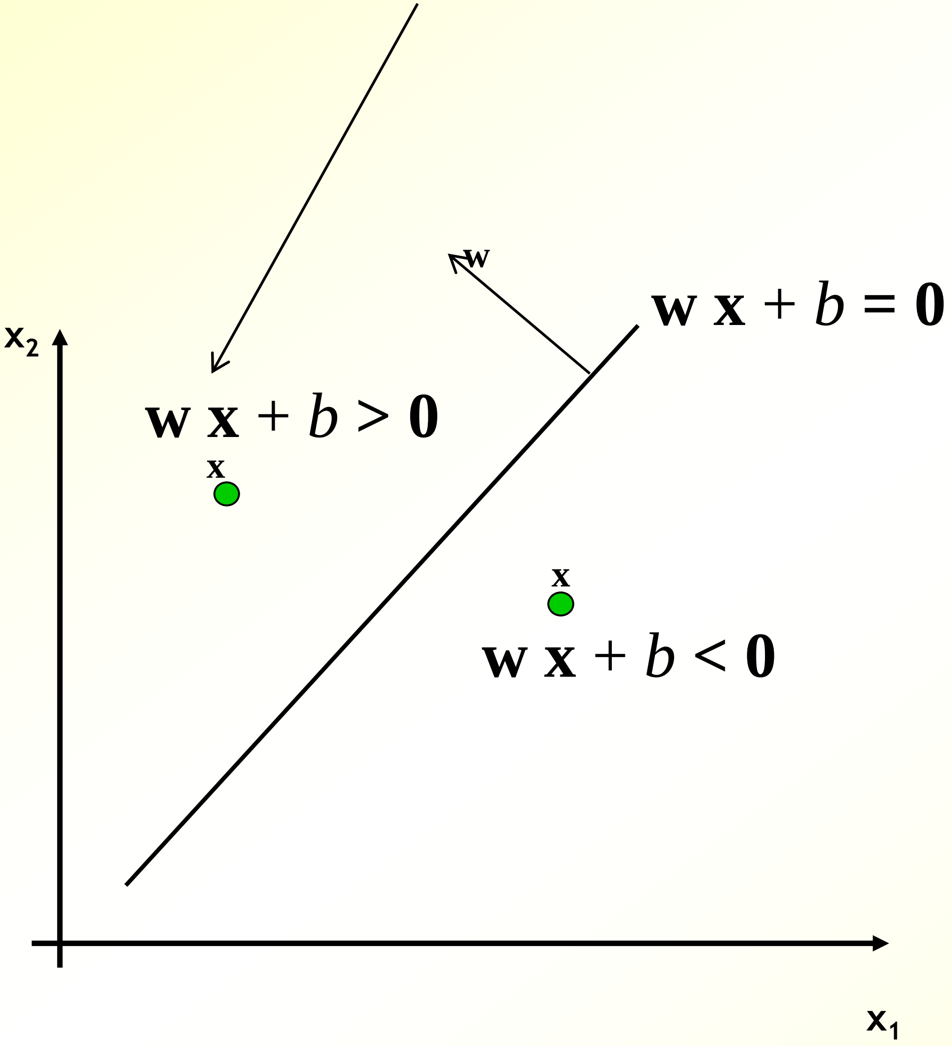
- SVMs are mathematical functions
- SVMs define boundaries in feature space (for classification). They can also be used for regression.
- Cases:
  - Linear:
  - Non-linear (kernels)





Linear models

The larger this value, the farther away from the boundary.


w
$\mathbf{w}\,\mathbf{x} + b = 0$
x_2
$\mathbf{w}\,\mathbf{x} + b > 0$
x
x
$\mathbf{w}\,\mathbf{x} + b < 0$
x_1

x is any point (instance) on the plane, to be classified by the linear model

2-D:

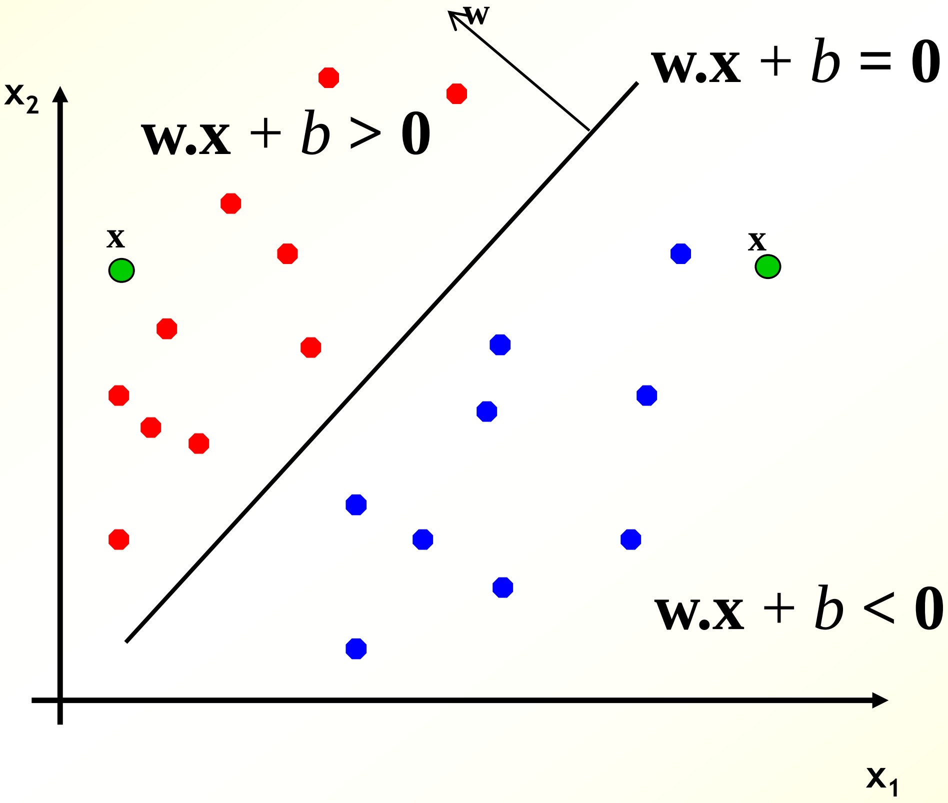
- $y = a * x + b$
- $x_2 = a * x_1 + b$
- $0 = a*x_1 - x_2 + b$
- $w_1*x_1 + w_2*x_2 + b = 0$
- $(w_1, w_2) . (x_1, x_2) + b = 0$
- $\mathbf{w} . \mathbf{x} + b = 0$

N-D:

- $\mathbf{w} . \mathbf{x} + b = 0$
- $(w_1, w_2, \ldots, w_n) . (x_1, x_2, \ldots, x_n) + b = 0$
- $w_1 * x_1 + w_2*x_2 + \ldots + w_n * x_n + b = 0$

Two-class linear classification

$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w.x} + b)$$

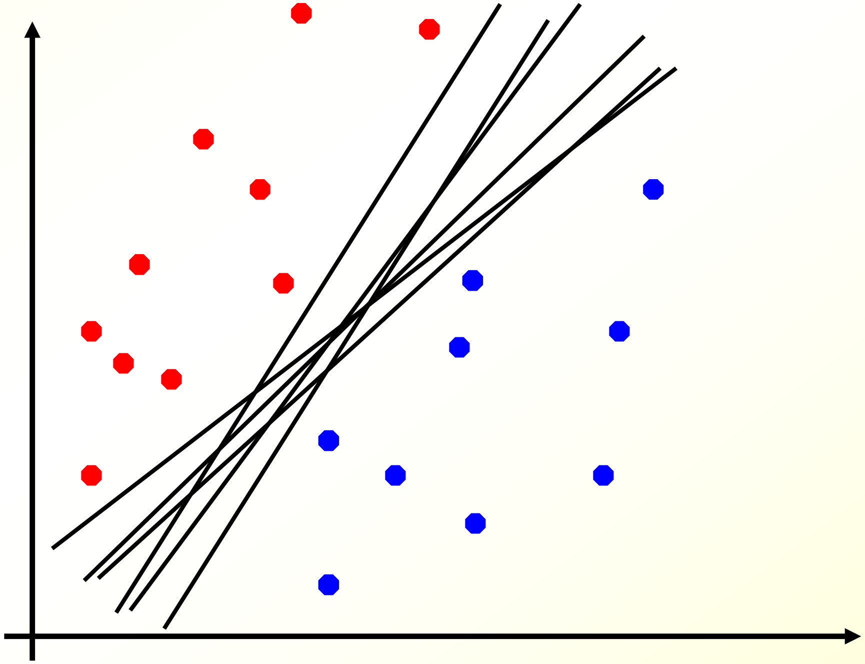


$\mathbf{w.x} + b = \mathbf{0}$

$\mathbf{w.x} + b > \mathbf{0}$

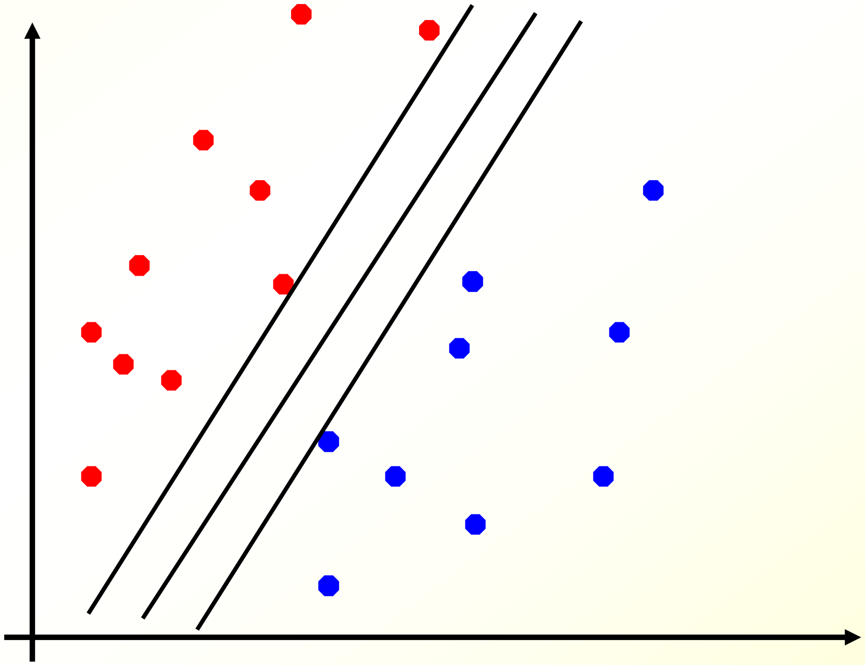
$\mathbf{w.x} + b < \mathbf{0}$

Which hyper-plane is best?



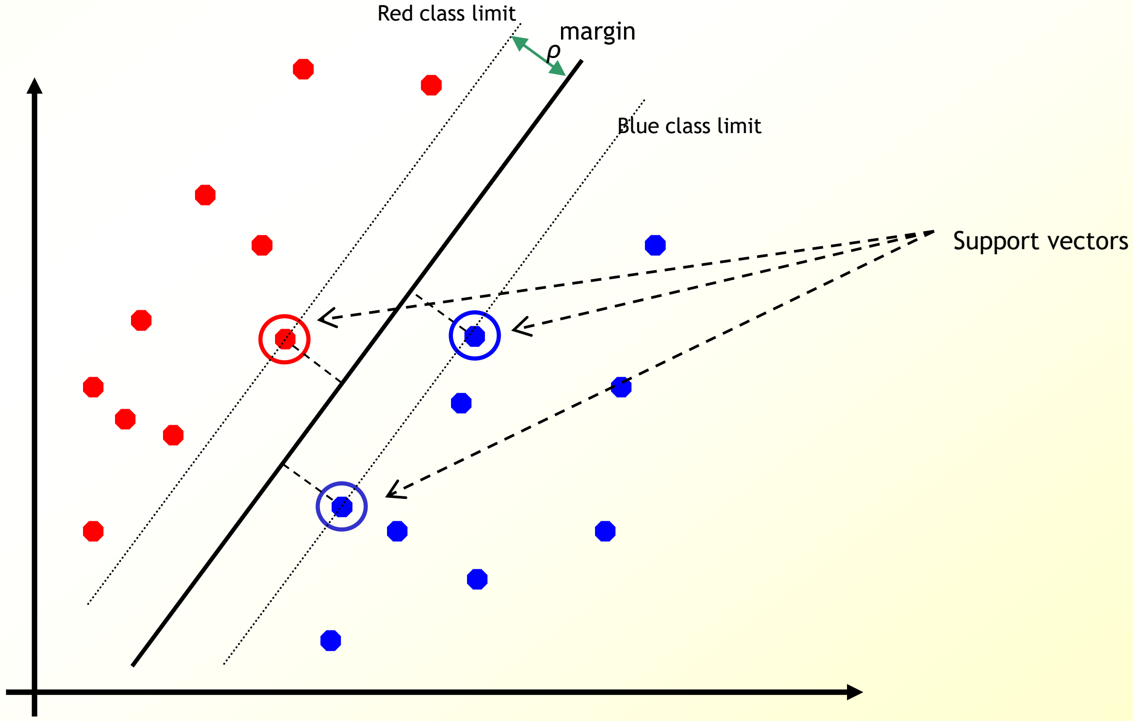
Which hyper-plane is best?

Which hyperplane generalizes better on testing data (new instances, different to training data)



Classification margin

- SVMs generate maximum margin hyper-planes, which results in the best generalization capability.
- The $\mathbf{x}_i$ at the class limits (margins) are the **support vectors**.
- Margin ρ is the perpendicular distance from the hyper-plane to the closest class limit.
- The boundary is actually defined by the support vectors



SVM optimization problem

$$\boxed{w\ x+b = w_1{}^*x_1+w_2{}^*x_2+b = 0}$$

- Find w and b that:
  - Maximize the margin (actually, it minimizes **1/margin**^{2})
  - Separate the instances belonging to the two classes.
  - The optimization process finds the support vectors.
- This optimization problem can be efficiently solved by quadratic optimization.
- If instances are linearly separable, the solution is unique.
- If not, there is no solution.

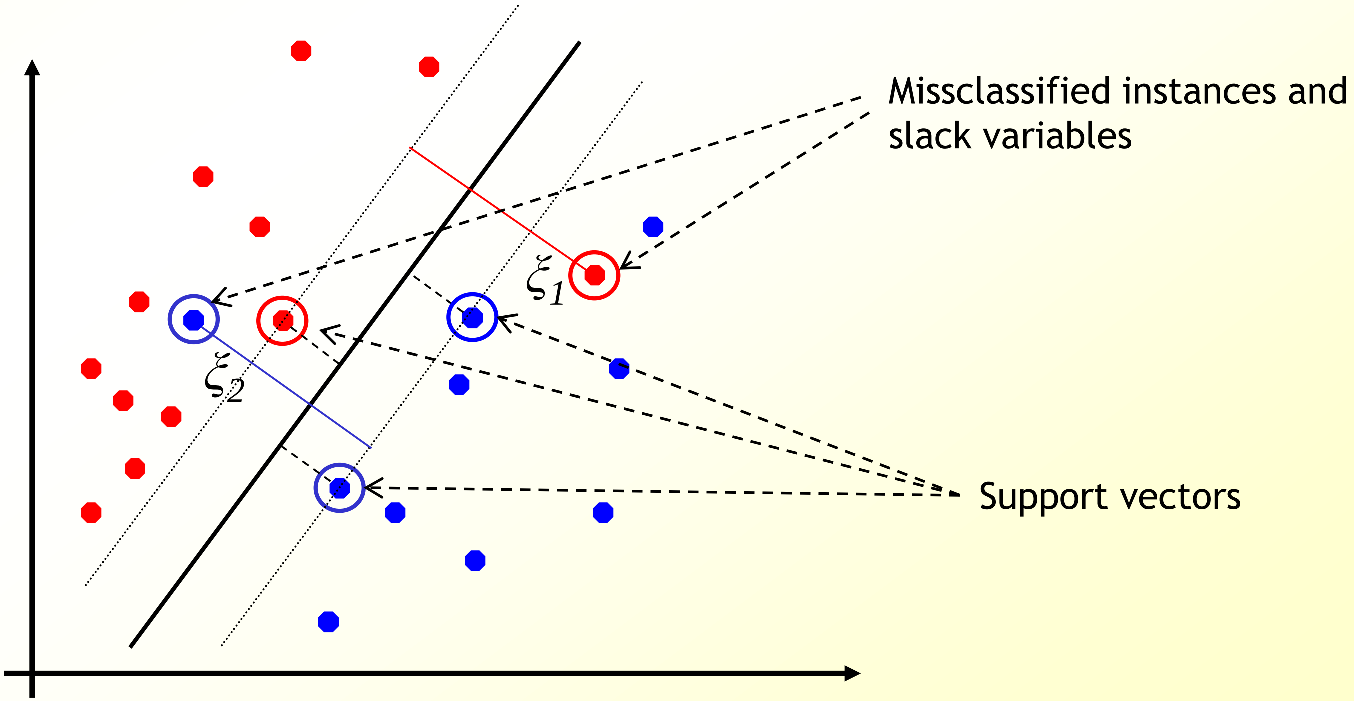
Support Vector Machines

- Cases:
  - Linear model:
    - Instances linearly separable (hard margin)
    - **Instances non-linearly separable (soft margin)**
  - Non-linear model (kernels)

Soft Margin, slack variables

- If classes overlap, linear classification is impossible.
- Solution: allow some points to remain wrongly classified.
- Slack variables ξ_i are the distances from the class limit to the wrongly classified instance.

Minimize: $(1/\mathit{margin}^2) + C*\Sigma\xi_i$



Missclassified instances and slack variables

Support vectors

*SVM optimization problem with **slack** variables (soft SVMs)*

$$\boxed{\mathbf{w}\ \mathbf{x}+b = w_1{*}x_1+w_2{*}x_2+b = 0}$$

- Find **w** and b that:
  - Maximize the margin and minimizes slack variables.
    - minimize $(1/\mathit{margin}^2) + C{*}\Sigma\xi_i$

- This optimization problema can also be solved by means of quadratic optimization. A solution always exists and it is unique.

Soft Margin, slack variables

- Now, soft margins are allowed, where some instances are allowed to be wrongly classified. Minimize the two aggregated goals:

$$(1/\mathit{margin}^2) + C^*\Sigma\xi_i$$

- *The C* hyper-parameter controls the weight given to each goal, which indirectly controls overfitting:
  - If *C* has a large value …
  - If *C* has a small value …

Soft Margin, slack variables

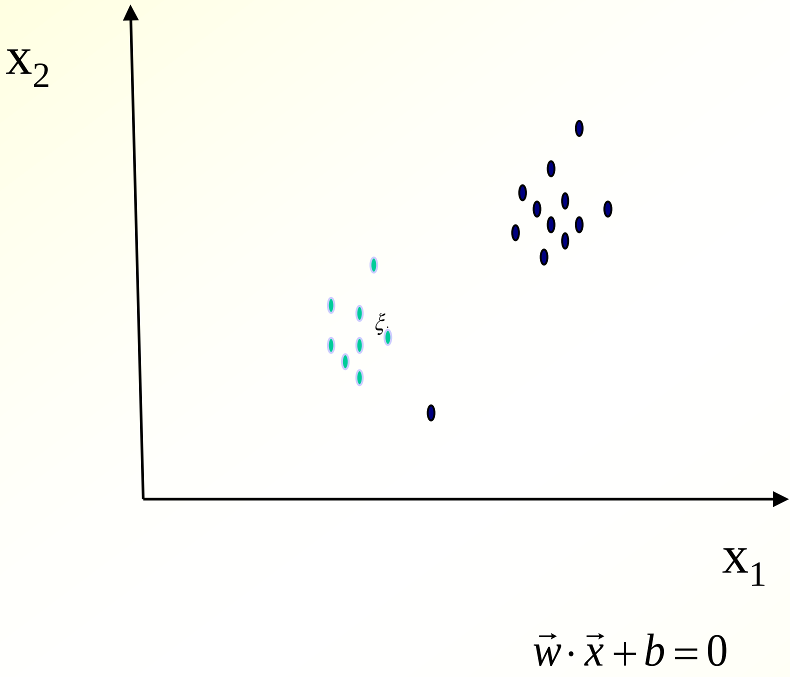
- The C hyper-parameter controls the weight given to each goal, which indirectly controls overfitting:
  - If *C* is large, it is very important for the optimizer to classify correctly all training instances (second goal). Equivalently, that slack variables are close to zero. If there is noise, there might be overfitting (memorization of noisy instances).
  - If *C* is small, the opposite happens: two many instances wrongly classified (many slack variables have large values)

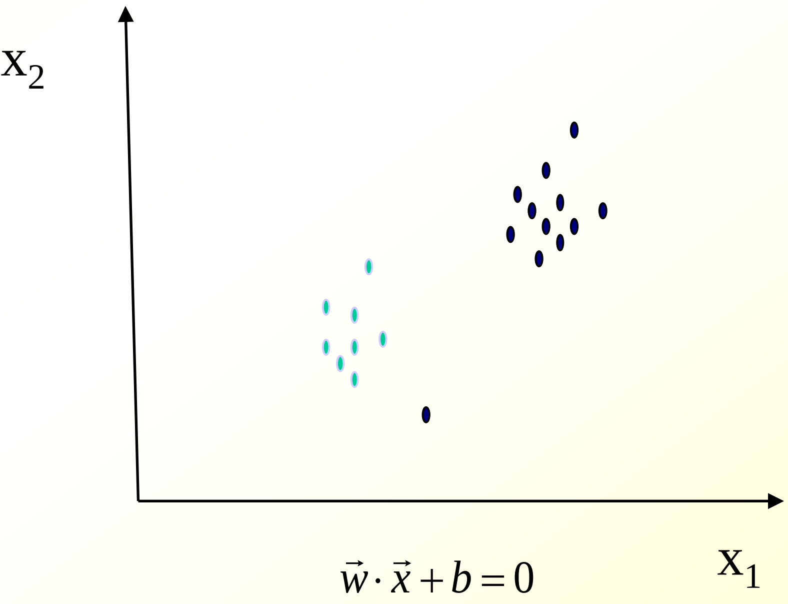
$$(1/margin^2) + C^*\Sigma\xi_i$$

$(1/margin^2) + C^*\Sigma\xi_i$

Soft Margin vs. Hard Margin

Same data, with slack variables (left) and no slack variables (right)





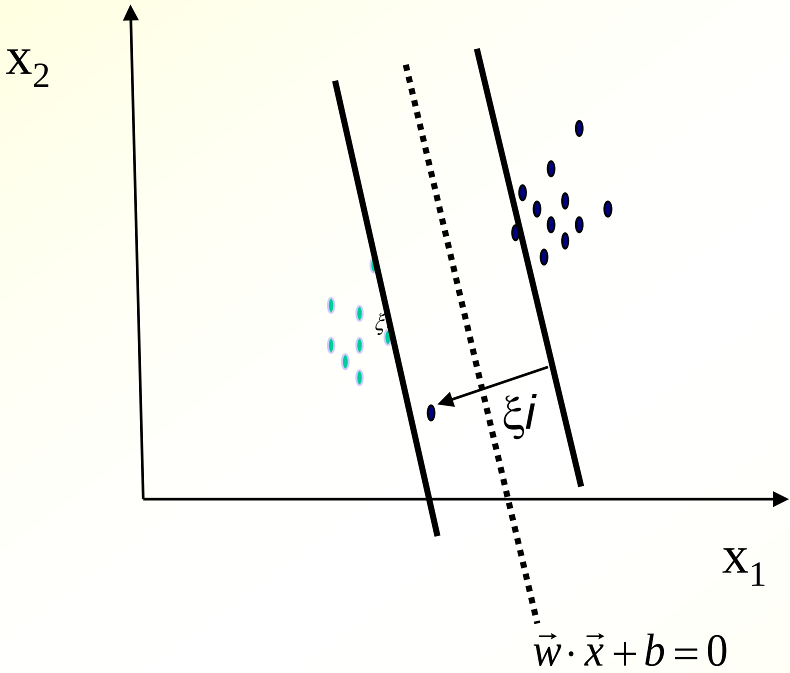
Soft Margin SVM (for an appropriate *C* value)

Soft Margin with very large *C*

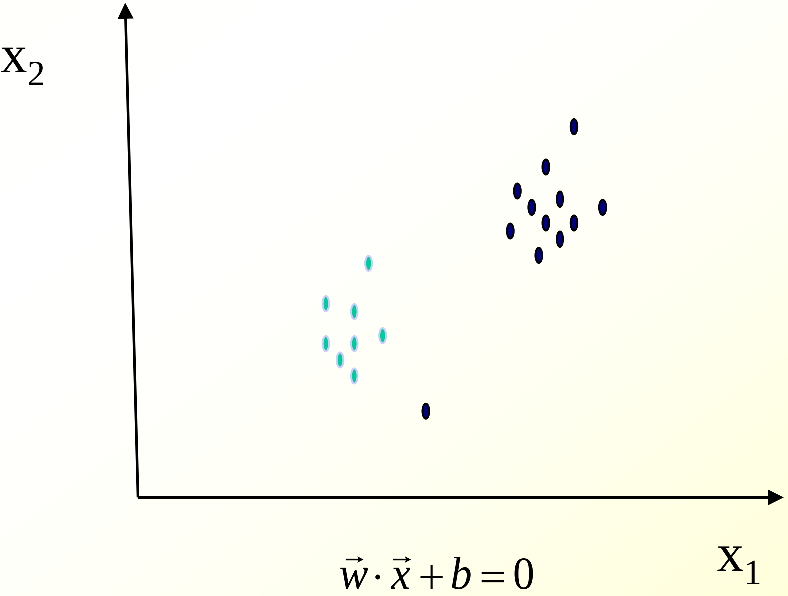
$$(1/\mathit{margin}^2) + C^{*}\Sigma\xi_i$$

Soft Margin vs. Hard Margin

Same data, with slack variables (left) and no slack variables (right)



Soft Margin SVM (for an appropriate *C* value)

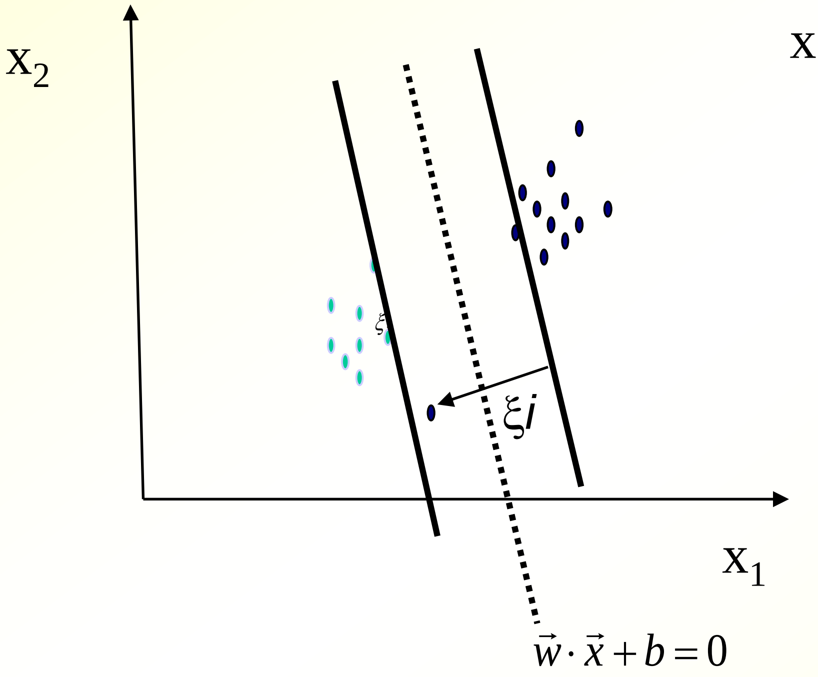


Soft Margin with very large *C*

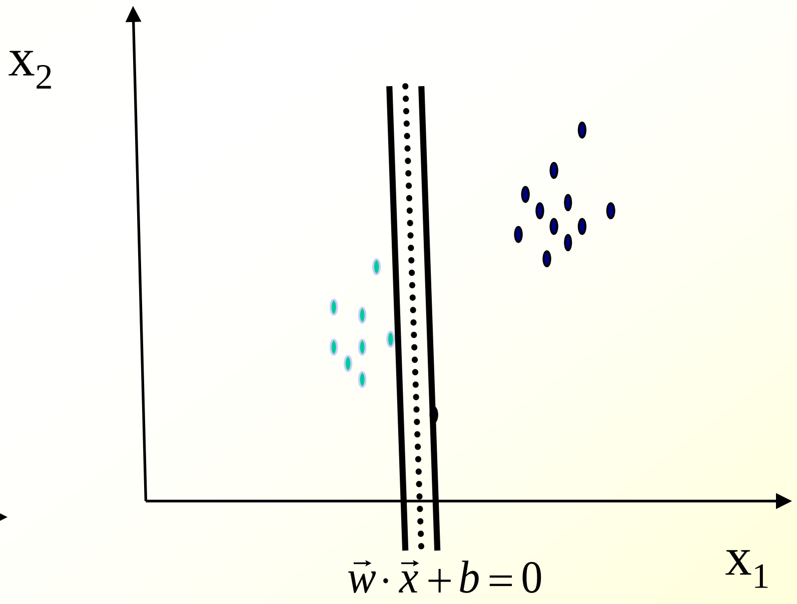
$$(1/\mathit{margin}^2) + C^*\Sigma\xi_i$$

Soft Margin vs. Hard Margin

Same data, with slack variables (left) and no slack variables (right)



Soft Margin SVM (for an appropriate *C* value)

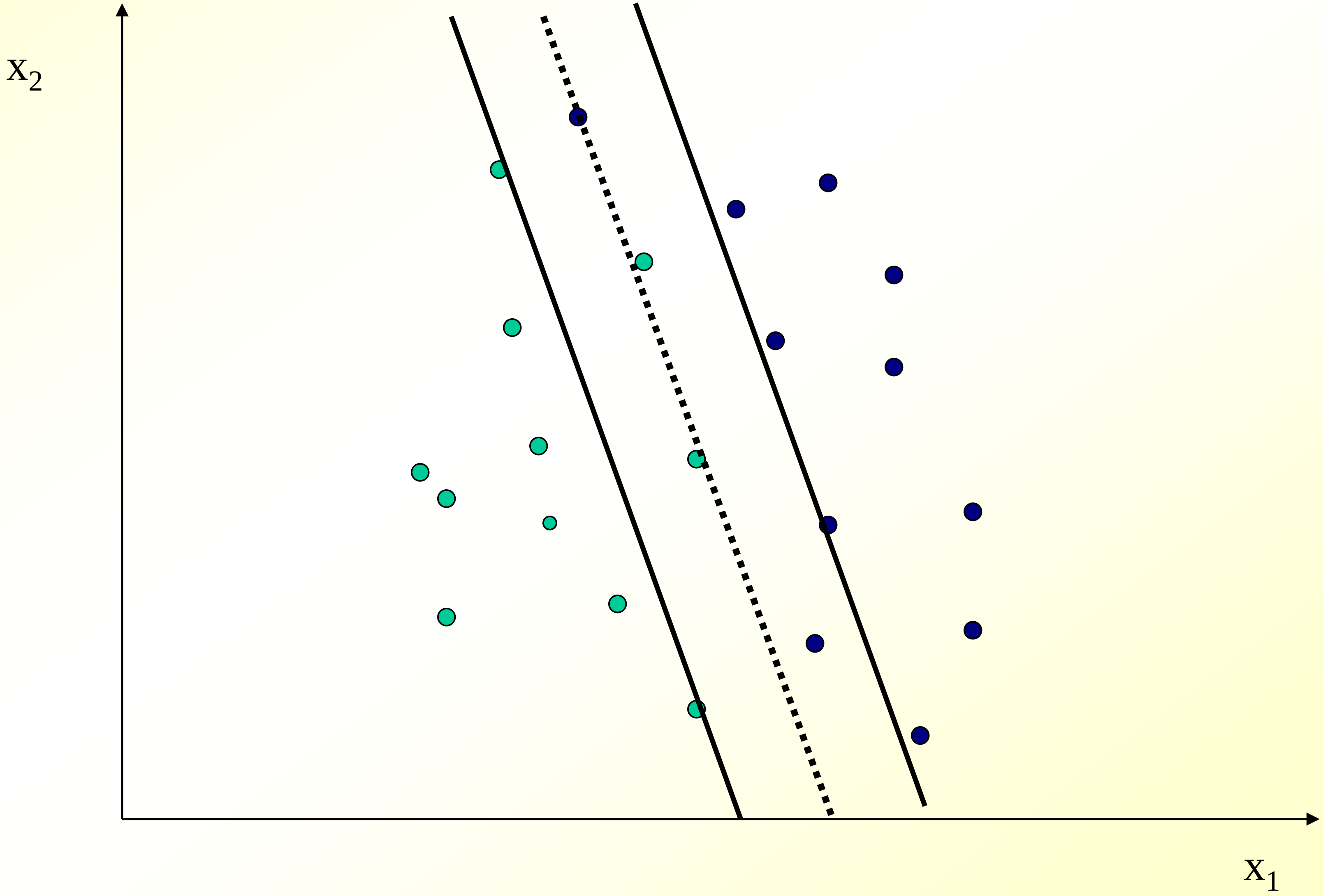


Soft Margin with very large *C*

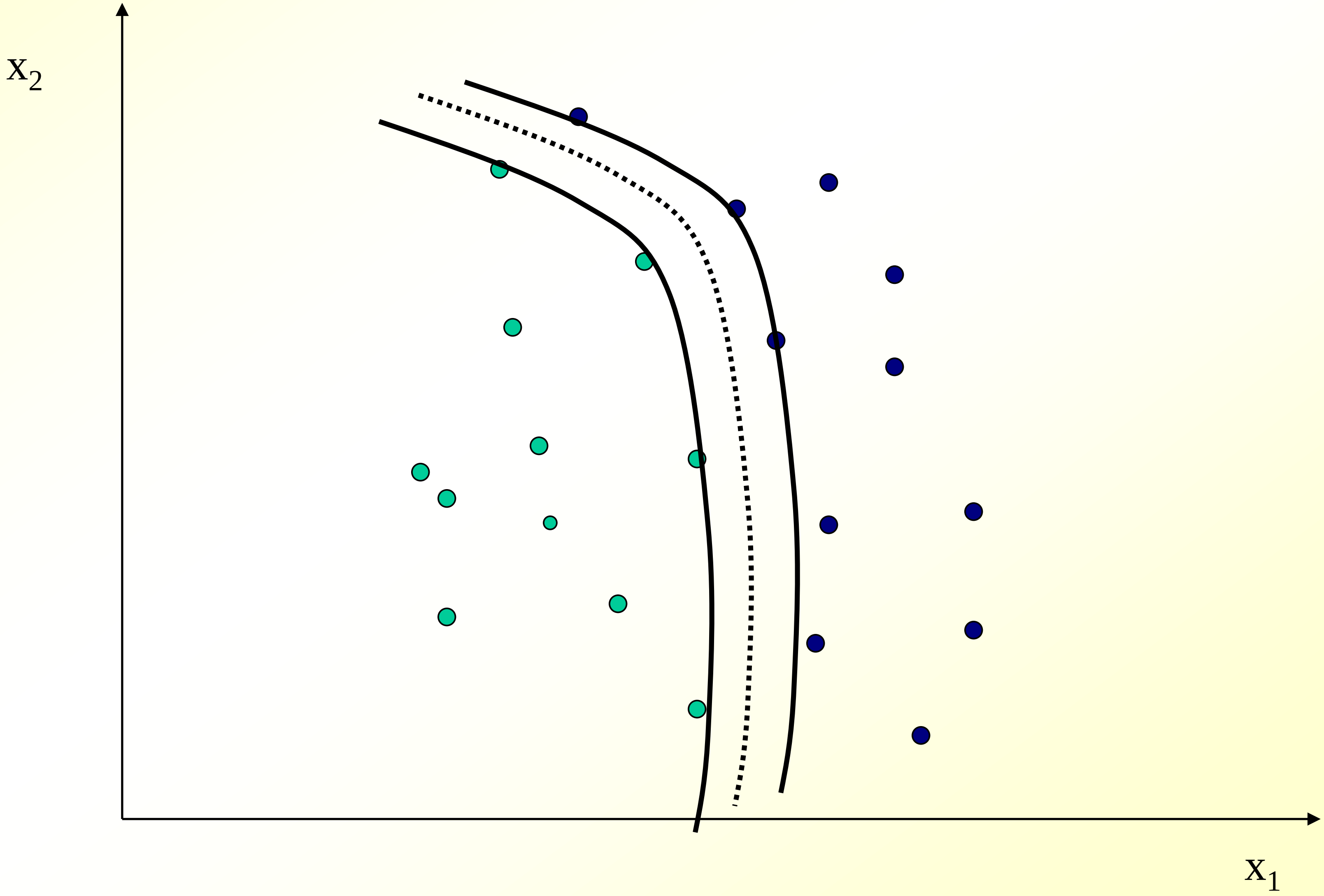
Support Vector Machines

- Cases:
  - Linear model:
    - Instances linearly separable (hard margin)
    - Instances non-linearly separable (soft margin)
  - **Non-linear model (kernels)**

Linear model vs. non-linear model



Linear model vs. non-linear model



Linear: w x+b=0

non-linear SVMs

- Mathematically, non-linear SVMs are:

$$\left(\sum_i (\alpha_i y_i) K(\vec{x}_i, \vec{x}) \right) + b = 0$$

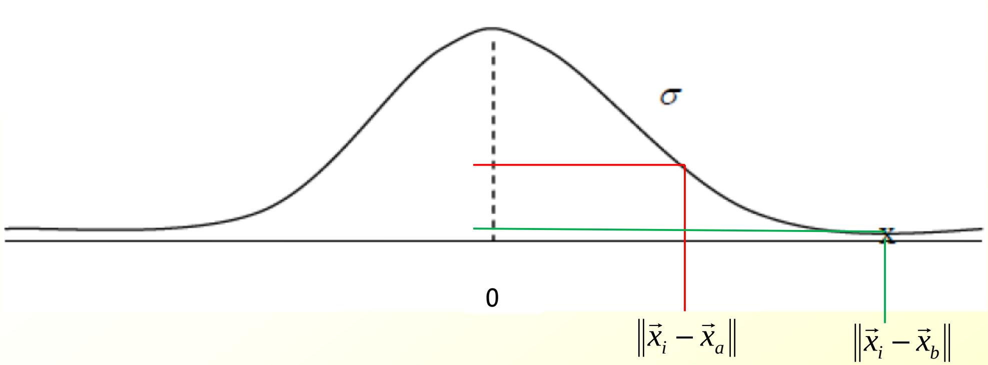
- ***x**$_i$* are the support vectors.
- *K(**x**$_i$, **x**)* is the kernel function (similarity) (it measures the similarity between support vector ***x**$_i$* and the instance to be classified ***x***)
- *y$_i$* = +1 o -1, positive or negative class of ***x**$_i$*
- *a$_i$* is a positive real number that represents the weight of the support vector in the final classification.

Kernels

- *K(**x**$_i$, **x**)* is the kernel function or “similarity”.
  - **x** instance to be classified
  - **x**$_i$ i-th support vector
- There are different kernels. They differ on how similarity is computed.
- The most useful kernel is the gaussian (radial) one. The smaller the euclidean distance |**x**$_i$ - **x** |, the larger the similarity (case **x**$_a$). But beyond some distance, similarity is close to zero (and does not change much) (case x$_b$).

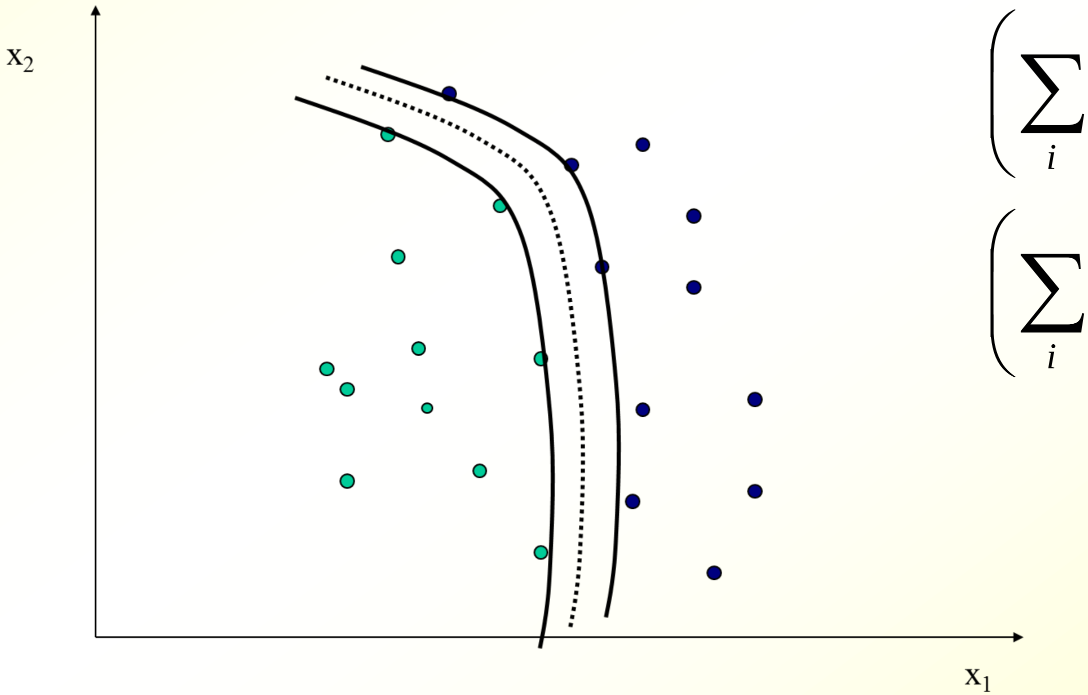
$$K(\vec{x}_i, \vec{x}) = \exp(-\|\vec{x}_i - \vec{x}\|^2 / 2\sigma^2)$$

$$K(\vec{x}_i, \vec{x}) = \exp(-\gamma\|\vec{x}_i - \vec{x}\|^2)$$



non-linear SVMs

- SVMs with radial kernel can be understood as a weighted summation of similarities to the support vectors, of the positive class (y_i = +1) , and of the negative class (y_i = -1).
- The winner class is that with a larger weighted summation.
- The result is something like the figure.



$$\left(\sum_i (\alpha_i y_i) K(\vec{x}_i, \vec{x})\right) + b = 0$$

$$\left(\sum_i (\alpha_i y_i) \exp(-\gamma \|\vec{x} - \vec{x}_i\|^2)\right) + b = 0$$

Non-linear SVMs

- Non-linear SVMs are actually linear, but in a transformed space.
- For example, radial SVMs are linear in a space of infinite dimensions. It is in that space where the máximum margin hyper-plane is constructed (and slack variables minimized).
- This linear boundary in the transformed space is non-linear in the original feature space.
- For example, a dataset for which the separation boundary is a circumference in feature space (x_1, x_2), is a linear boundary in a transformed space (v_1, v_2), for transformation $v_1=x_1^2$, $v_2=x_2^2$

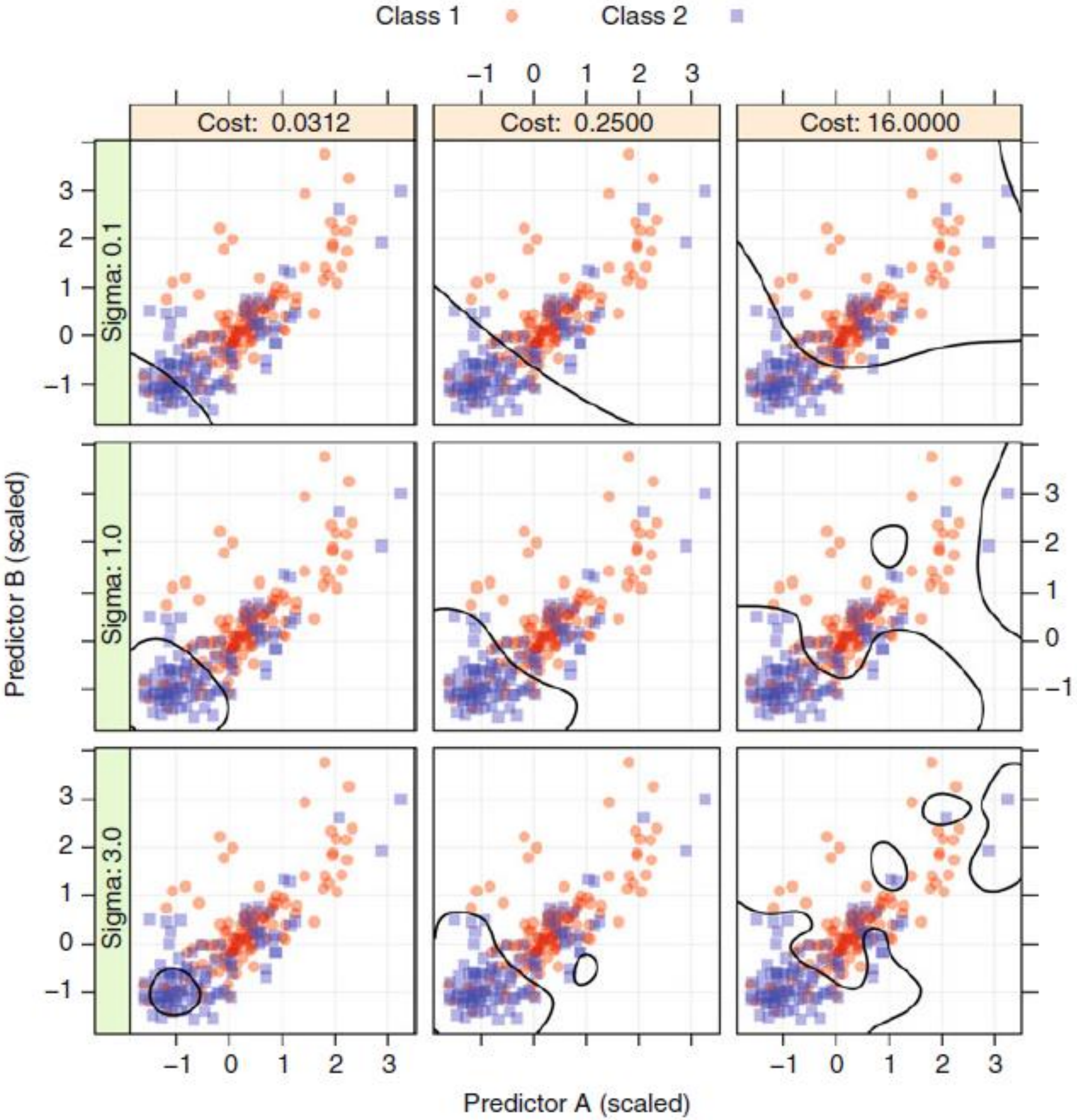
Methodology for using SVMs

1. Scale features (scaler, minmaxscaler, robustscaler, ...)
2. Hyper-parameter tuning:
   - *C* (cost): $(1/\mathit{margen}^2) + C * \Sigma \xi_i$
   - Kernel:
     - Linear (also called polynomial with degree 1): no hyper-parameters
     - Gaussian or radial (rbf): gamma.

$$\left(\sum_i (\alpha_i y_i) \exp(-\gamma \|\vec{x} - \vec{x}_i\|^2)\right) + b = 0$$

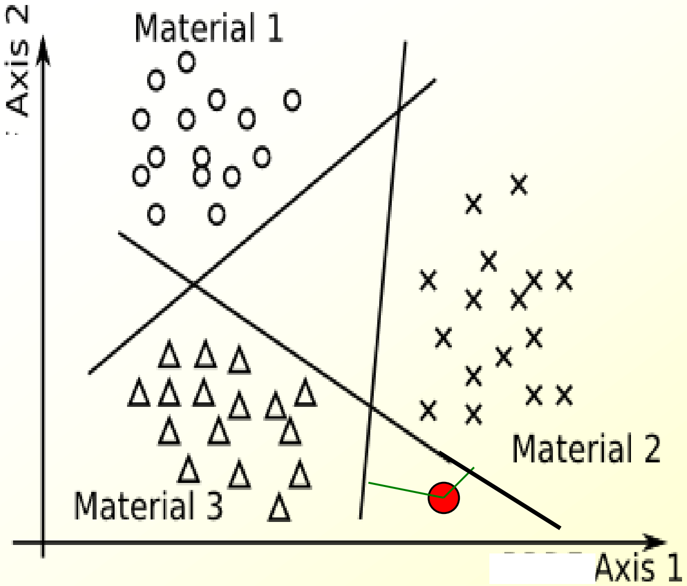
   - The values of C and gamma to be tried, should grow exponentially
     - 'C': [0.1,1, 10, 100, 1000] = [10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}]
     - 'gamma': [1,0.1,0.01,0.001, 0.0001] = [10^{0}, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]

- Note: try first with linear kernel, then with RBF kernel.


Class 1 ●
Class 2 ■
Cost: 0.0312
Cost: 0.2500
Cost: 16.0000
Sigma: 0.1
Sigma: 1.0
Sigma: 3.0
Predictor B (scaled)
Predictor A (scaled)
−1
0
1
2
3

From two-class to multi-class

- **One versus all** (or one versus rest): as many separating hyperplanes as classes are used. Each hyperplane separates one class from all the others. In the end, the hyperplane from which the instance to be classified is the farthest away, is the one that wins (the further a point is from a hyperplane, the more certainty it belongs to the class)
- **One versus one**: as many hyperplanes as pairs of classes. Each hyperplane separates one class from another. For example, if there are 4 classes, there will be 6 hyperplanes: 1 | 2, 1 | 3, 1 | 4, 2 | 3, 2 | 4, 3 | 4. In the end, the hyperplanes vote. This approach is usually too computationally expensive, if the number of classes is large.



From two-class to multi-class

- One-versus-all:
  - Eg: with 3 classes:
    - Boundary 1 vs. 2+3
    - Boundary 2 vs. 1+3
    - Boundary 3 vs. 1+2
    - When classifying new instances, choose the boundary for which the distance to the instance is larger.
- One-versus-one:
  - Eg: with 3 clases:
    - Boundary 1 vs. 2
    - Boundary 1 vs. 3
    - Boundary 2 vs. 3
    - When classifying new instances, boundaries vote.
- sklearn uses one-vs-one