MASTER OF SCIENCE IN ENGINEERING

Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Machine Learning

T-MachLe

7. Support Vector Machines (SVM)

Christophe Gisler
Jean Hennebert
Andres Perez Uribe

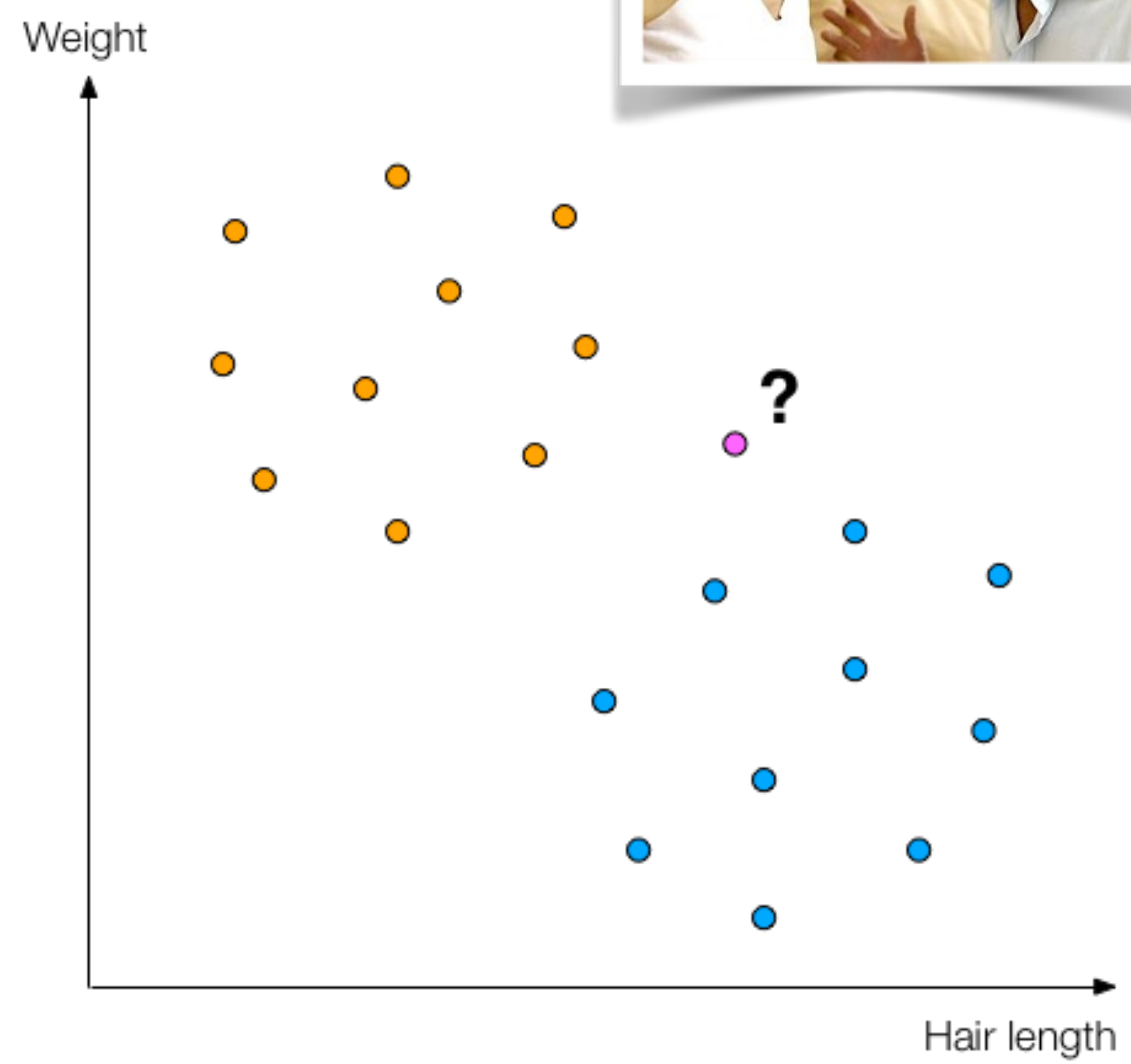
Plan - Classification task with Support Vector Machines (SVM)

7.1 Recaps On classification tasks and logistic regression

By Balu Ertl https://upload.wikimedia.org/wikipedia/commons/5/54/Euclidean_Voronoi_diagram.svg

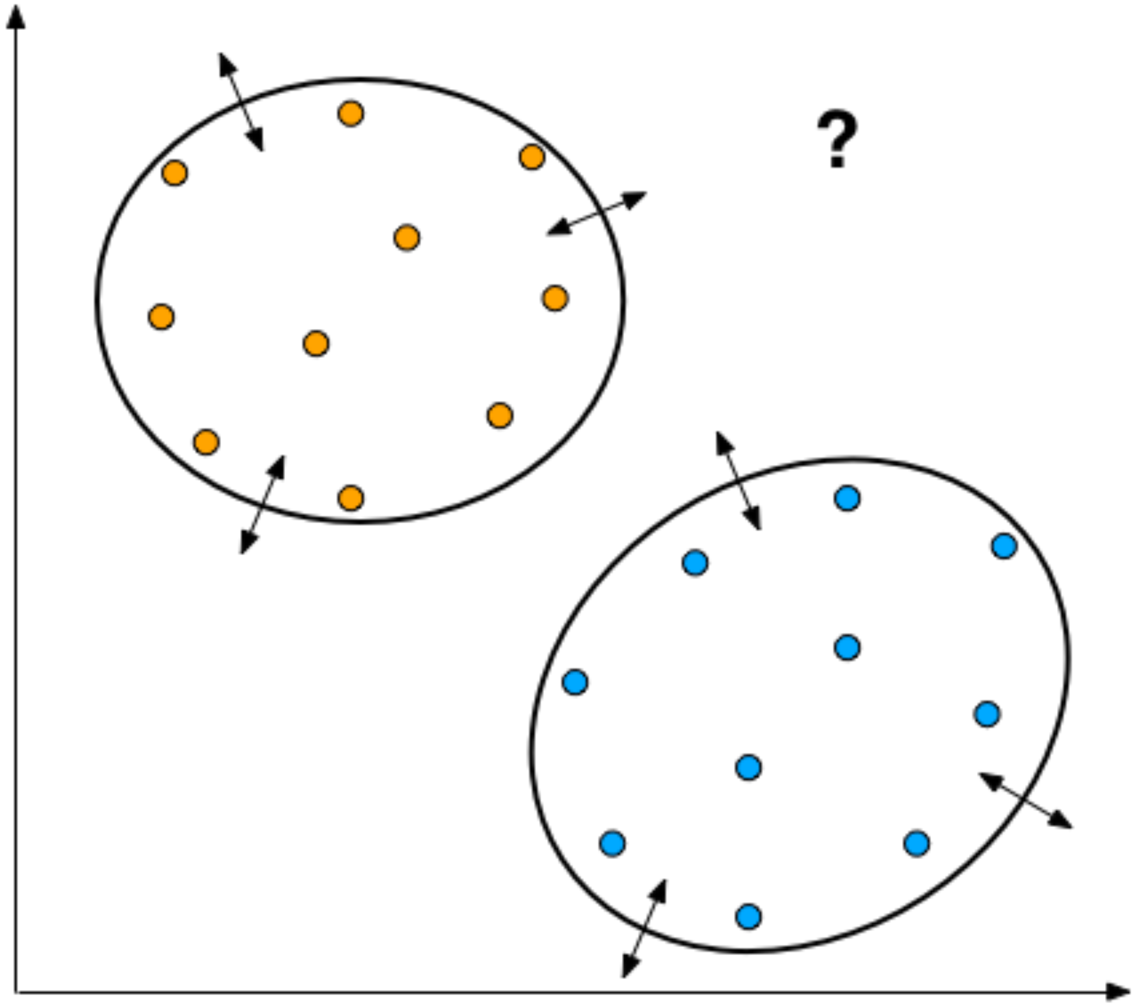
Classification task

- Example:
  - **2 classes**:
    Women (•)
    Men (•)
  - **2 features**:
    Hair length (axe X)
    Weight (axe Y)
  - **What about the new sample (•)?**
    Is it a woman or a man?



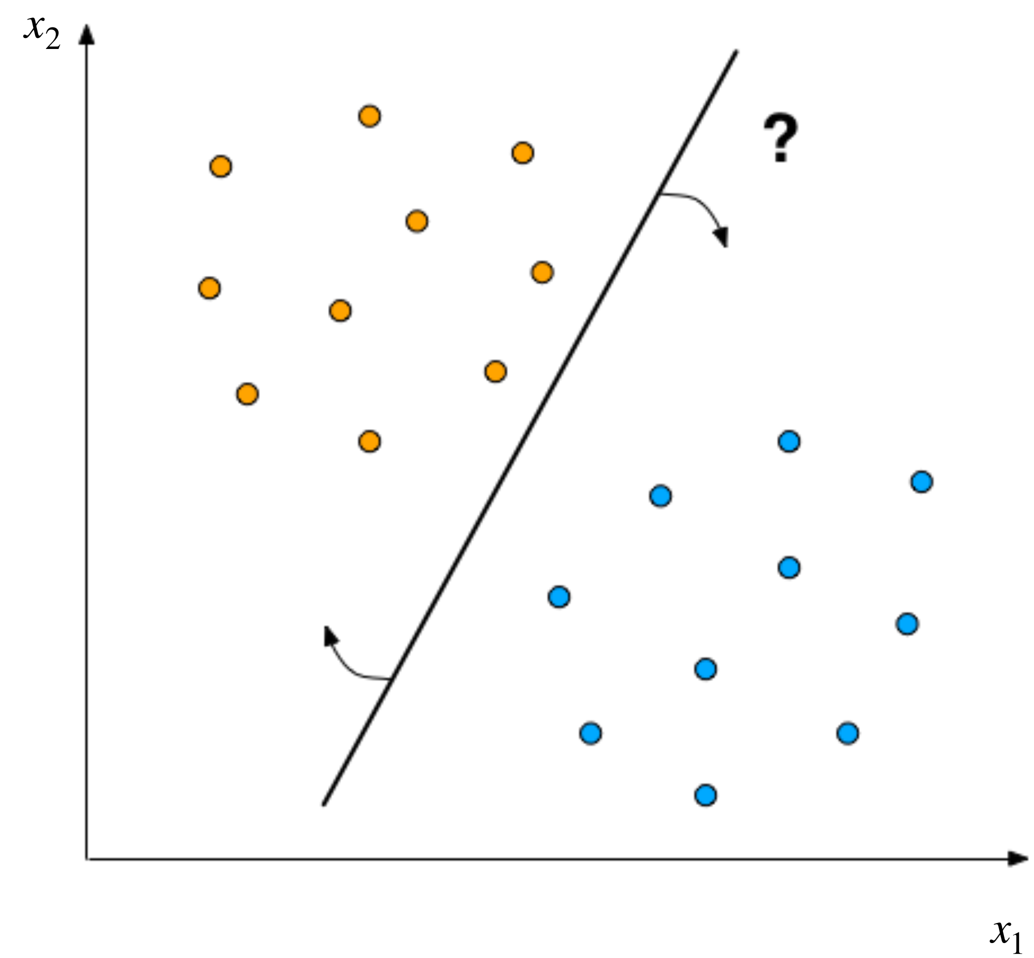
Classification task - generative models

- Some algorithms try to build a **model per class**, i.e. generative model explaining the distribution within the class
  - E.g. Likelihood estimation with Gaussian Mixture Models (**GMM**)

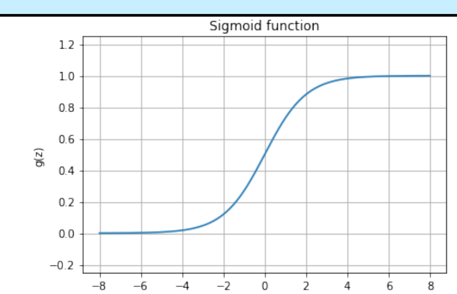


Classification task - discriminant models

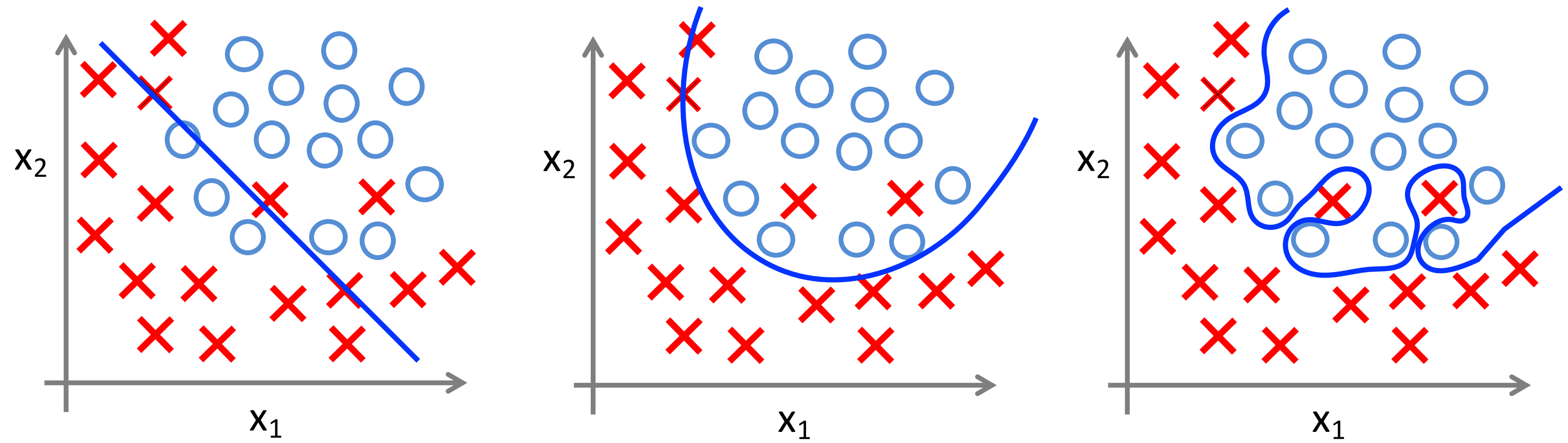
- Some algorithms try to **discriminate classes**, i.e. to build a border defining the classes
- Last week we have seen **Logistic Regression**
  - Discover the equation of the border that maximise the correct classifications
  - The equation of the border is feeding a sigmoid so that on one side of the border the probability value is larger than 0.5 (class 1) or smaller (class 0)
  - Logistic regression may find many possibilities of hyperplanes
- Which is the **best separating hyperplane**?



$$\hat{y} = P(C_1 | x) = 1 - P(C_2 | x)$$
$$\hat{y} = h_\theta(\mathbf{x}) = sigmoid(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$



Logistic regression with non-linear decision boundaries



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2$
$+\theta_3 x_1^2 + \theta_4 x_2^2$
$+\theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2$
$+\theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2$
$+\theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

- To move to non-linear decision boundaries, just add features to the x array
  - e.g. representing dependencies to the squared

$$h_\theta(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \ldots)$$

$$h_\theta(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots)$$

with $x_3 = x_1^2$

x1 = surf	x2 = rooms	x3 = surf^2	y = rented
26	1	26*26 =	0
37	1.5	1369	1
57	2	3249	0
48	2	2304	1
…	…	…	…

Training discriminant models

- A common strategy to find the best parameters through training is the **gradient descent**.

Gradient descent involves the computation of two mathematical terms:

- A **loss function** $J(\theta)$
  - It expresses how bad we are with the current values of parameters θ on the train set
  - We want to minimise this function through the training procedure
- The **gradient of the loss** with respect to the parameters $\frac{\partial J}{\partial \theta}$

- The negative of the gradient will tell us in which direction to move the parameters to minimise the loss

Loss function and gradient for logistic regression

- We want to maximise the number of correct classifications, i.e. the product over the training set of the a posteriori probabilities.
  - As the derivation of the product is not easy, we took the log of the product which transformed as a sum of the logs.

$$J(\theta) = -\frac{1}{N}\sum_{n=1}^{N} J(\theta, x_n) = -\frac{1}{N}\sum_{n=1}^{N} y_n \log h_\theta(\mathbf{x_n}) + (1-y_n)\log(1-h_\theta(\mathbf{x_n}))$$

Note : in the last class, the negative term was not there and we had to maximise the function J

When $y_n = 1 \quad J(\theta, x_n) = -\log(h_\theta(\mathbf{x_n})) = -log(\hat{y}_n)$

When $y_n = 0 \quad J(\theta, x_n) = -\log(1-h_\theta(\mathbf{x_n})) = -\log(1-\hat{y}_n)$

- After some mathematical developments, the gradient of the loss w.r.t. the parameter is

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{1}{N}\sum_{n=1}^{N}(y_n - h_\theta(\mathbf{x}_n))x_{n,i}$$

Target value

Gotten output

Training with a gradient descent

- Update any parameters of your model in the opposite direction of the gradient of the loss w.r.t. weights

$$\text{param} \leftarrow \text{param} - \alpha \frac{\partial J}{\partial \text{param}}$$

In practice we have stopping criteria, e.g. epoch < max_epochs or loss_gain < ε

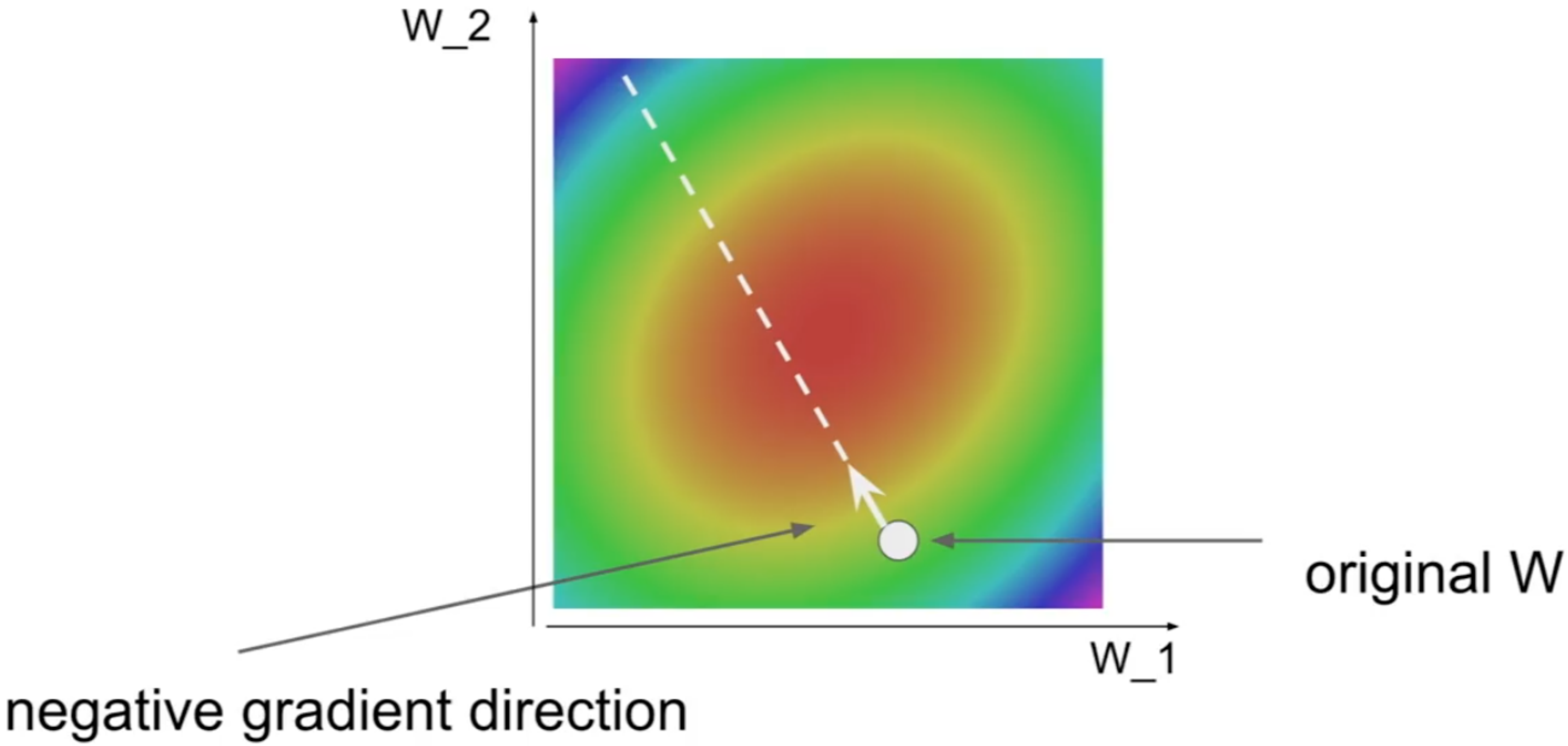
1 sample = stochastic gradient descent
B samples = mini-batch gradient descent
N samples = (full) batch gradient descent

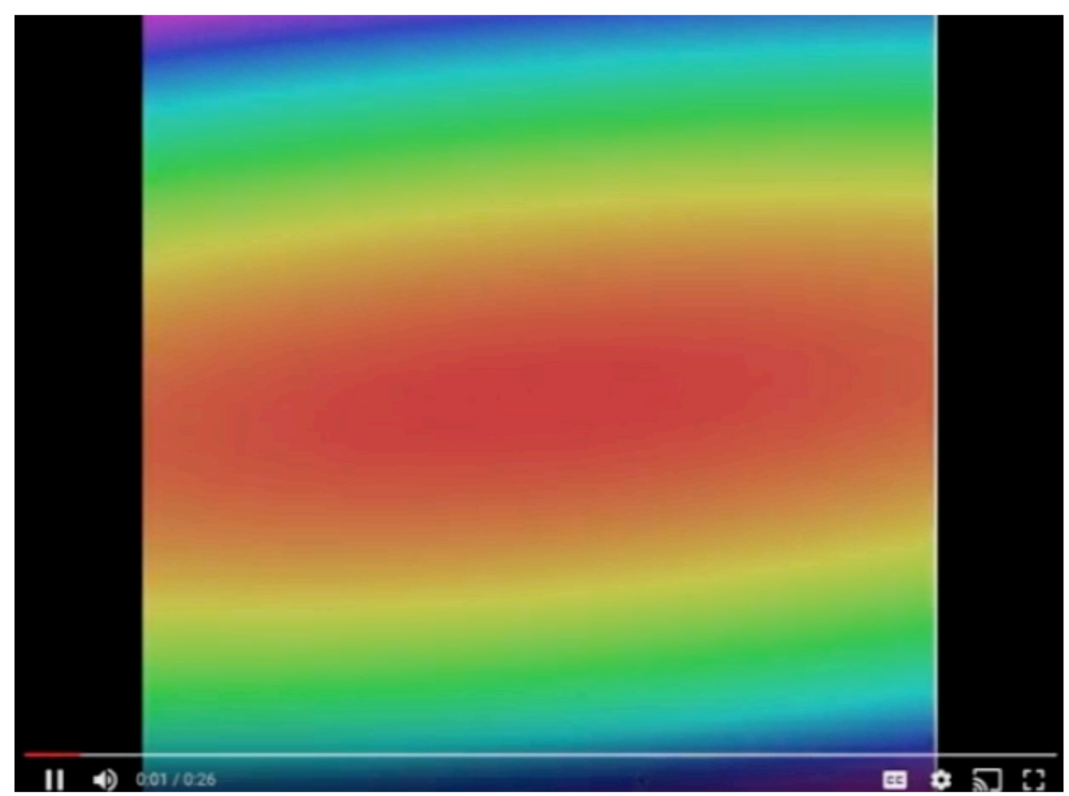
Current weighs

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

The loss function



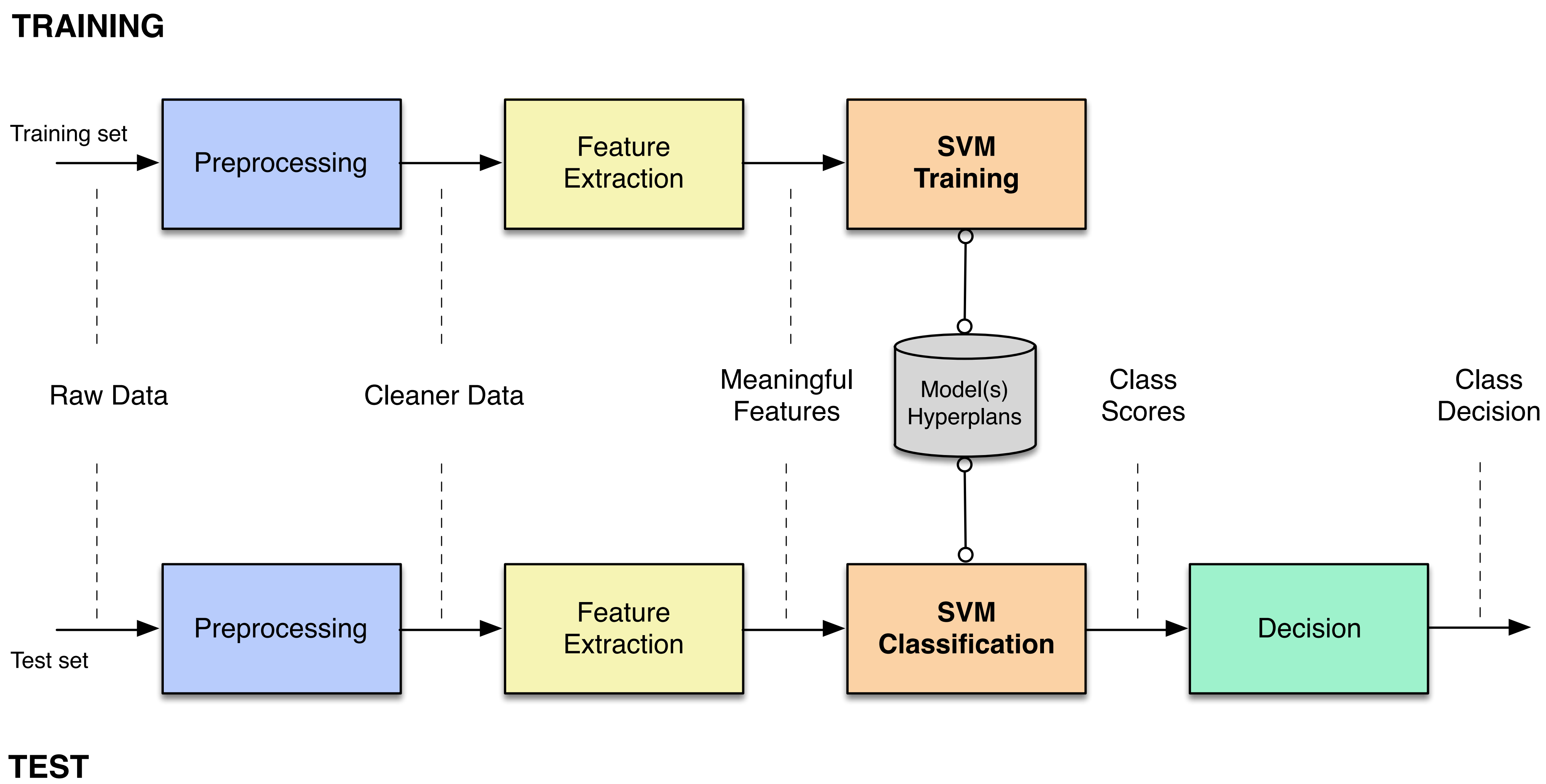


From Fei-Fei Li & Justin Johnson & Serena Yeung, April 2018: CS231n Stanford

7.2 Linear SVM

By Balu Ertl https://upload.wikimedia.org/wikipedia/commons/5/54/Euclidean_Voronoi_diagram.svg

Classification task

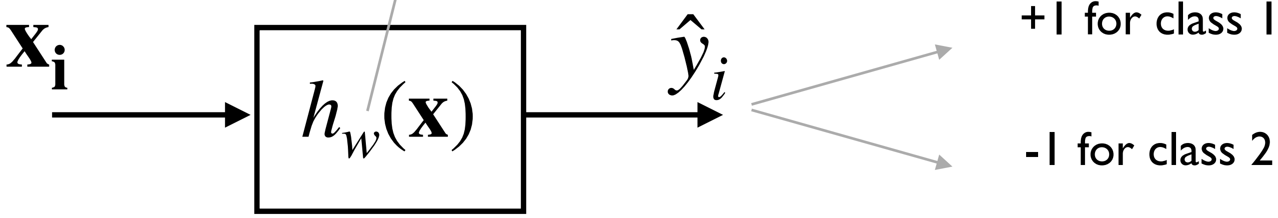


Linear SVM

- Given:
  - Training samples $\mathbf{x_i} \in \mathbb{R}^D$
  - A two class problems with targets $y_i \in \{-1, +1\}$ associated to classes C_1 , C_0
  - Number of samples : N with i = 1,…,N
  - The hypothesis function:

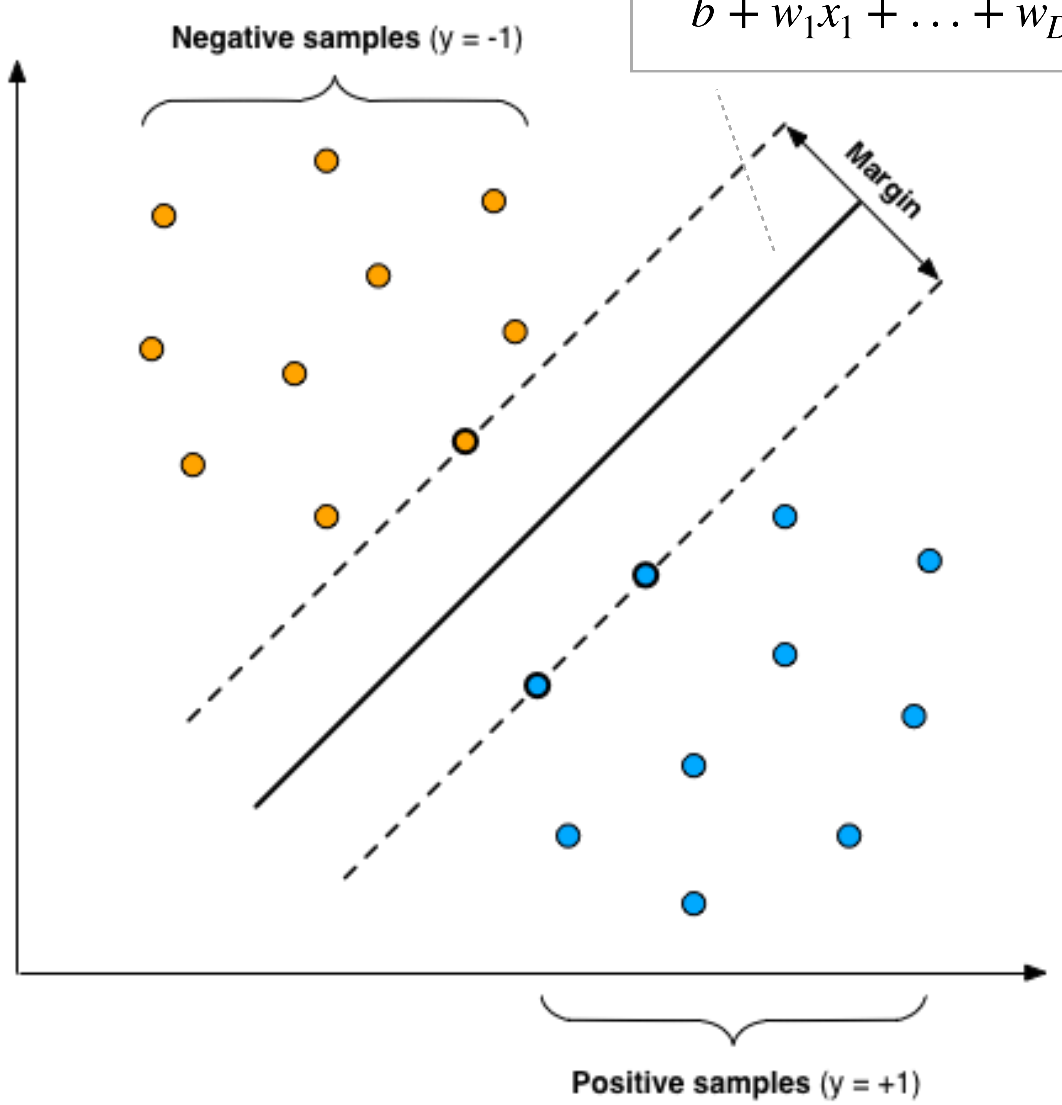
$$h_w(\mathbf{x}) = sign(b + w_1x_1 + \ldots + w_Dx_D)$$
$$h_w(\mathbf{x}) = sign(b + \mathbf{wx})$$

In previous classes, we used symbol θ instead of **w**



A linear SVM tries to find the **hyperplane** that **separates** the 2 classes and that **maximizes** the **margin** between the 2 classes

Hyperplane equation:
$b + w_1x_1 + \ldots + w_Dx_D = 0$



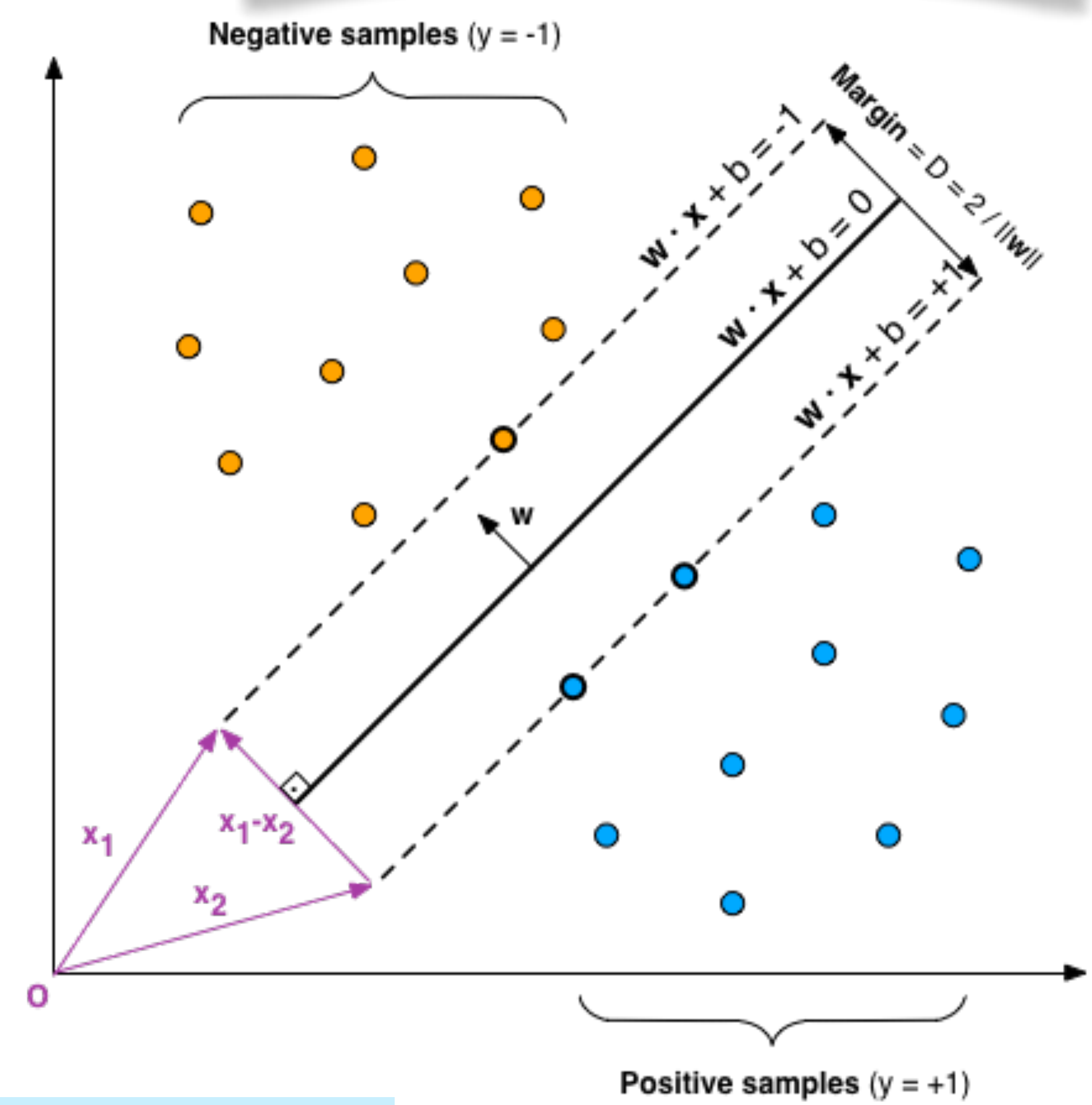
Linear SVM

> By geometric definition **w** is the normal vector of the hyperplane, i.e. perpendicular to the hyperplane

> ||**x**|| is the Euclidian norm of vector **x**
>
> $$\|\boldsymbol{x}\|_2 := \sqrt{x_1^2 + \cdots + x_n^2}.$$

What does it mean to maximise the margin?

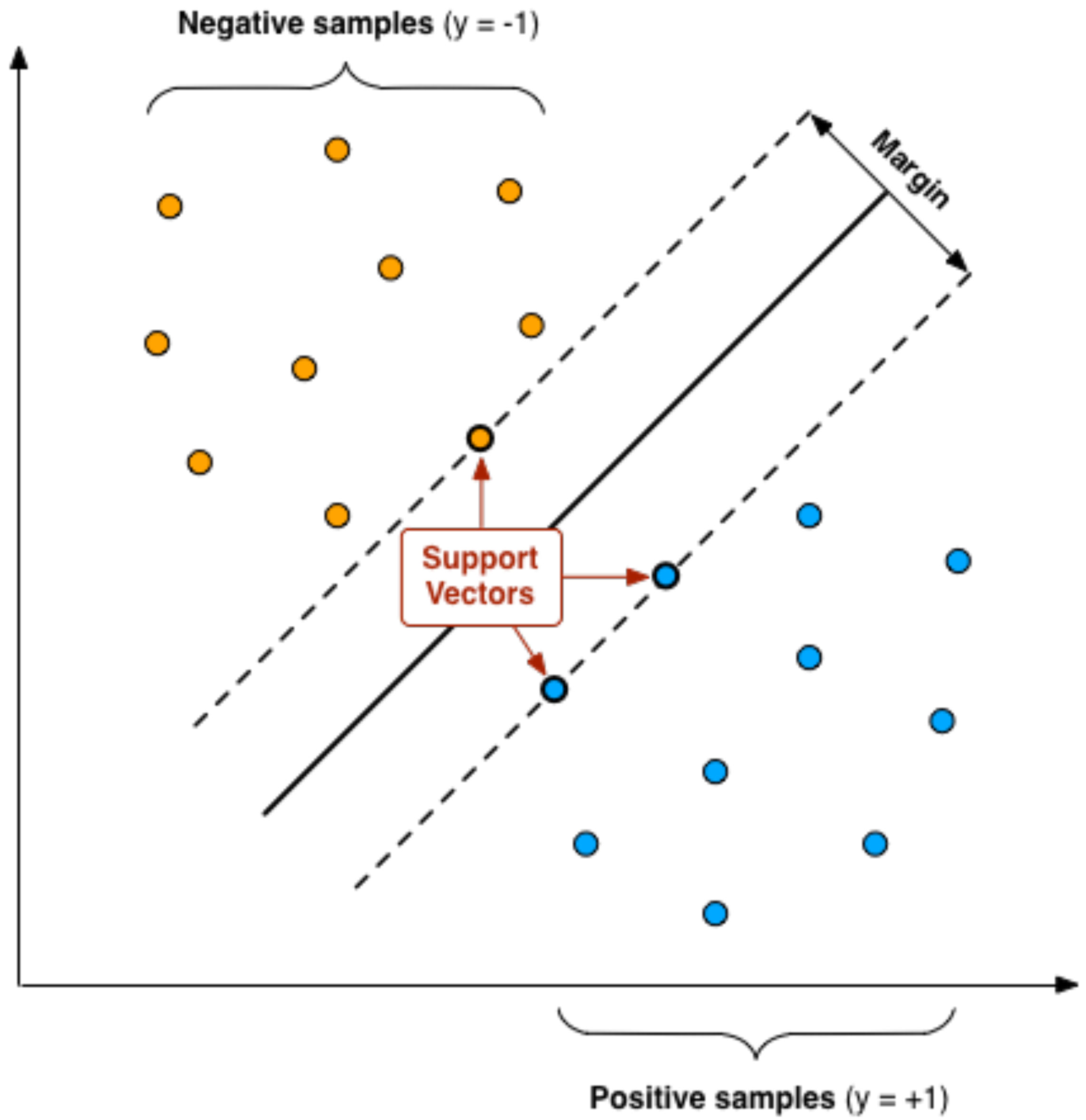
- Margin = distance M between the 2 parallel hyperplanes (on boundaries):
  $\mathbf{w} \cdot \mathbf{x} + b = -1$ (a is a constant >0)
  $\mathbf{w} \cdot \mathbf{x} + b = +1$
- Let's define $\mathbf{x_1}$, $\mathbf{x_2}$ as points on these 2 hyperplanes such that: $\mathbf{x_1} - \mathbf{x_2} = t \cdot \mathbf{w}$ (t is a scalar)
- So we have: $\mathbf{x_2} = \mathbf{x_1} + t \cdot \mathbf{w}$
  $M = \|\mathbf{x_1} - \mathbf{x_2}\| = \|t \cdot \mathbf{w}\| = t \cdot \|\mathbf{w}\|$ (1)
  $\mathbf{w} \cdot \mathbf{x_1} + b = -1$ (2)
  $\mathbf{w} \cdot \mathbf{x_2} + b = +1$ (3)
- (3) - (2) gives: $\mathbf{w} \cdot (\mathbf{x_2} - \mathbf{x_1}) = 2$
  $\Longleftrightarrow$ $\mathbf{w} \cdot (\mathbf{x_1} + t \cdot \mathbf{w} - \mathbf{x_1}) = 2$
  $\Longleftrightarrow$ $\mathbf{w} \cdot t \cdot \mathbf{w} = 2$ $\Longleftrightarrow$ $t\,(\mathbf{w} \cdot \mathbf{w}) = 2$
  $\Longleftrightarrow$ $t\,\|\mathbf{w}\|^2 = 2$ $\Longleftrightarrow$ $t = 2 / \|\mathbf{w}\|^2$ (4)
- (4) in (1) gives: $M = 2 / \|\mathbf{w}\|$
- Hence to maximize M, we have to minimize ||**w**|| while respecting the correct classification constraints (see next slide)



In other words, a term on the loss function will include ||**w**||

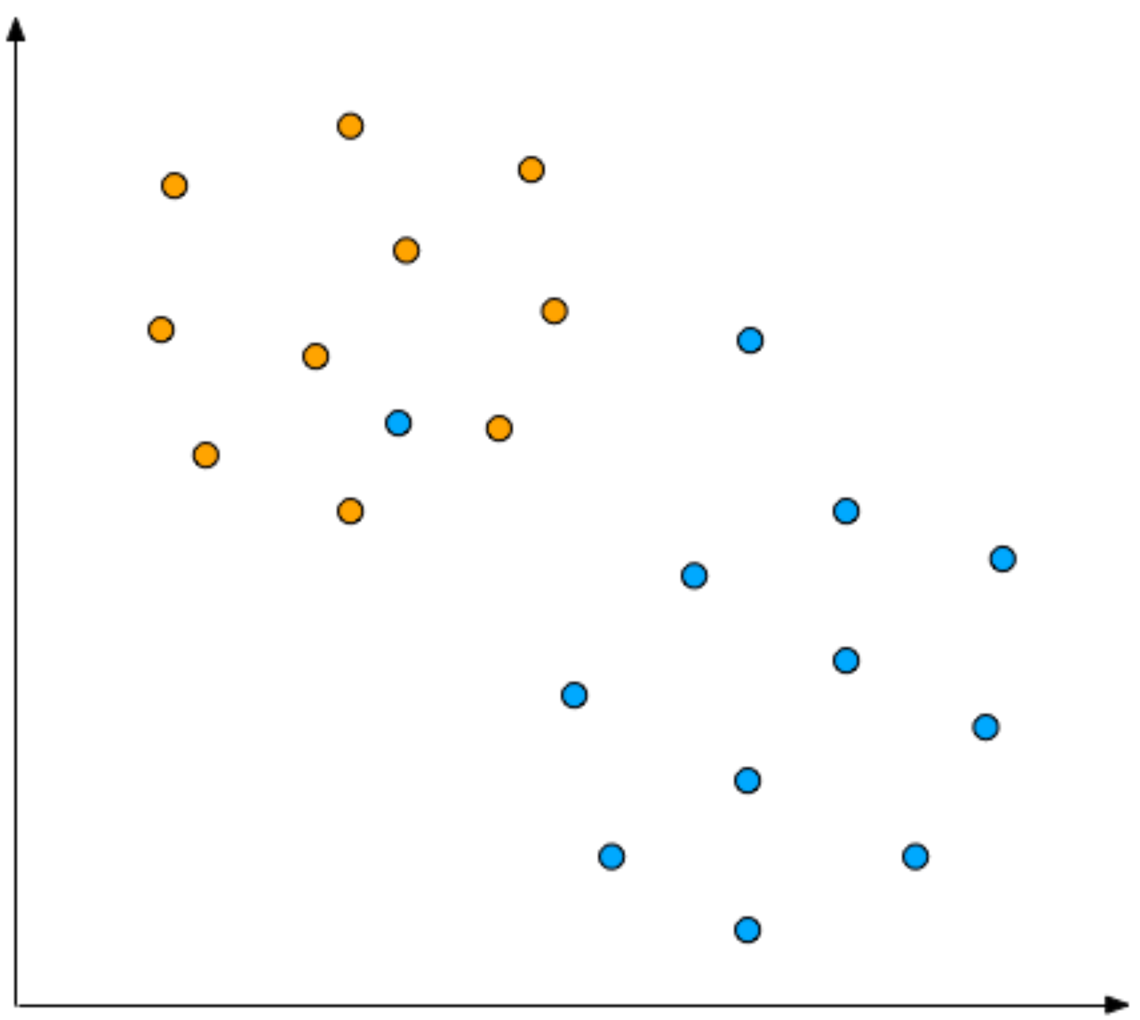
Linear SVM

- Training samples on the margin boundaries are called the **support vectors**



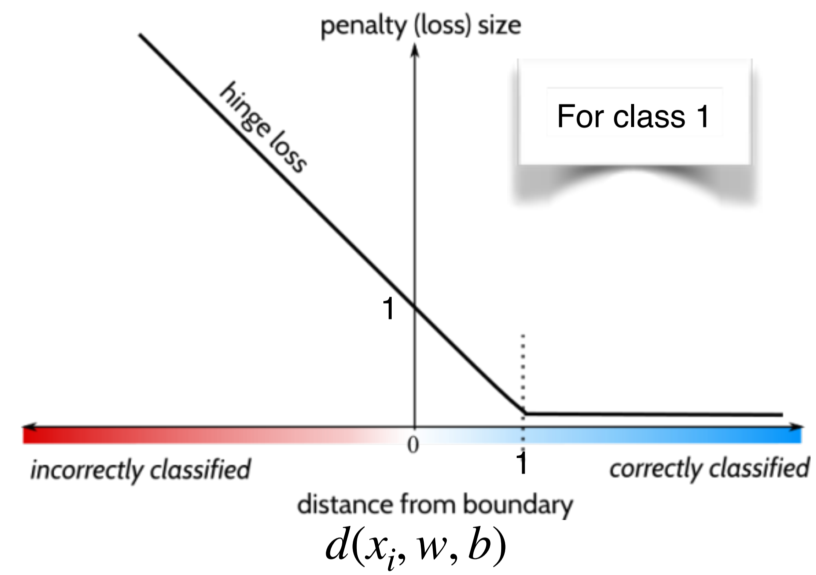
Linear SVM for not linearly separable data

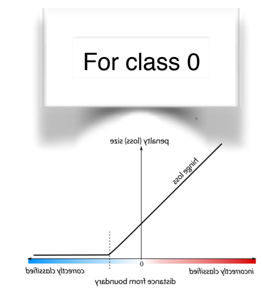
- **Problem:** how to linearly separate these 2 classes?
- We need to inject another term in the cost function that will minimise the number of incorrectly classified samples.
- For SVM, the “Hinge” loss is used
  - It takes into account the notion of margin

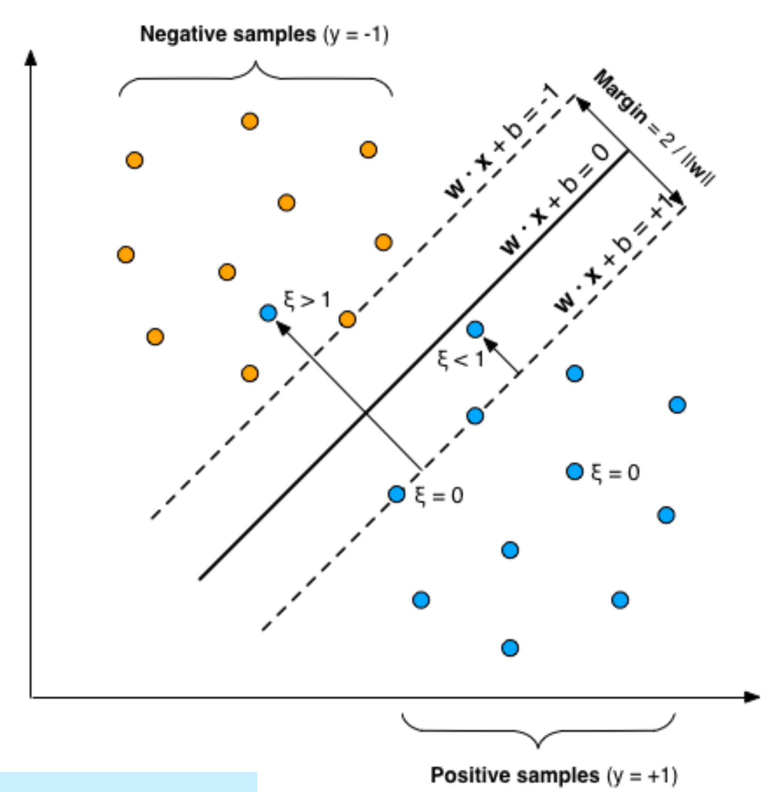


Linear SVM for not linearly separable data

- For each sample $\mathbf{x}_i$, compute the Hinge loss ξ_i
  - = 0 if the point falls above the margin
  - = 1 if the point falls on the hyperplane
  - > 1 if the point is on the wrong side of the hyperplane
- The $\xi \geq 0$ measure the degree of misclassification of samples in terms of distance to the plane
- Overall loss is quantified by: $\sum \xi_i$ for i=1,…,N
- In the SVM terminology, the ξ_i are also called the **slack** variables.



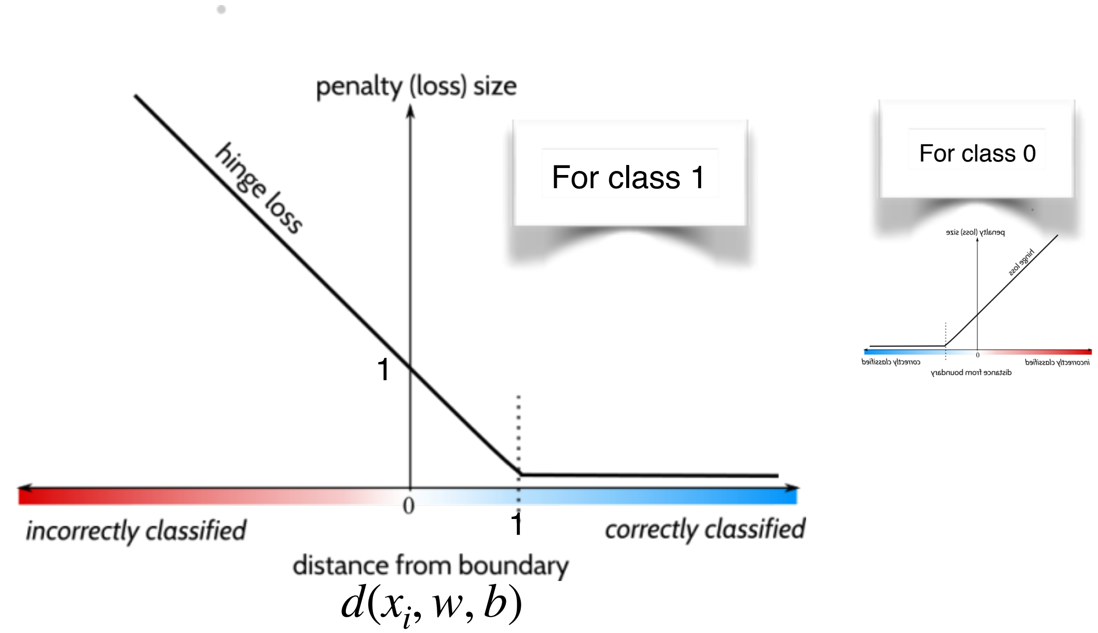




In other words, a term on the loss function will include $\sum \xi_i$

SVM loss function

- We can then define the SVM loss function taking into account of two terms to minimise:
  - One based on the Hinge loss for both classes C_1 and C_0
  - One based on ||**w**||
    - For mathematical reasons, we instead minimise $||\mathbf{w}||^2/2$



$$J(\theta) = C\left[\frac{1}{N}\sum_{i=1}^{N} y_i \text{ hinge}_{C1}(d(x_i, w, b)) + (1 - y_i) \text{ hinge}_{C0}(d(x_i, w, b))\right] + \left[\frac{1}{2}||w||^2\right]$$

Trade-off coef : giving more or less importance to perf term

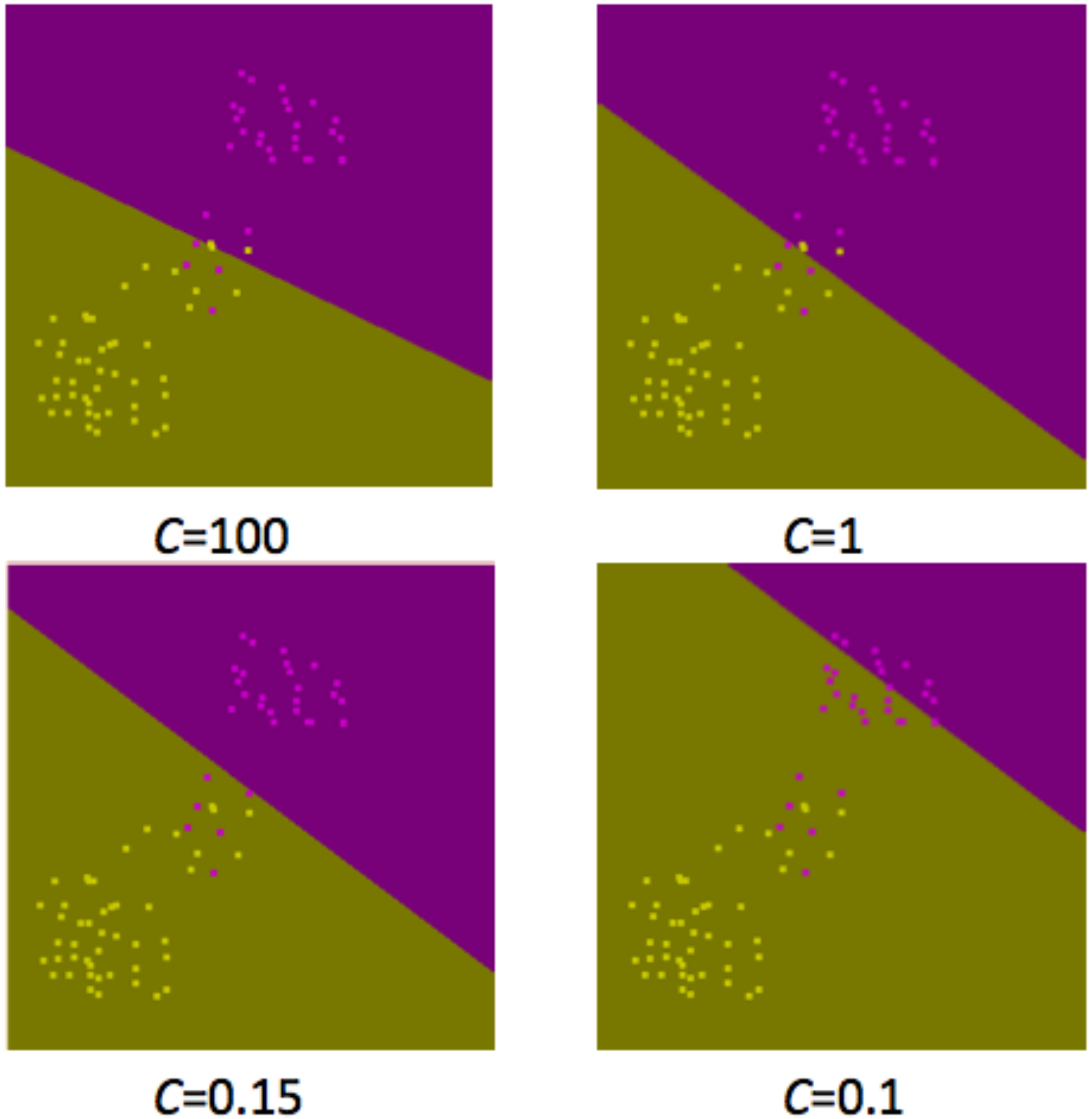
Performance term: on the data, how far are we predicting from the ground truth?

Regularisation term: impeach too large values of **w**

Impact of parameter C

- The factor C is a regularization parameter which trades off the margin size and the training error

- The smaller C, the greater the number of admitted misclassified train samples



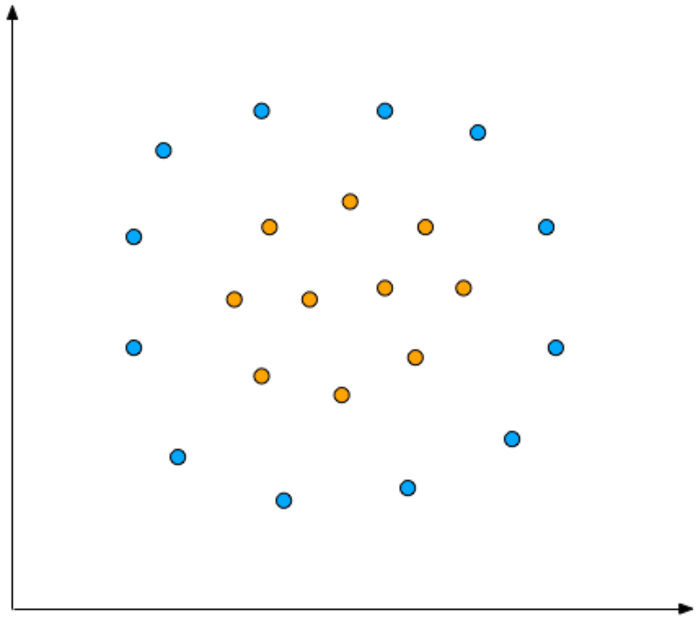
Minimizing the loss function

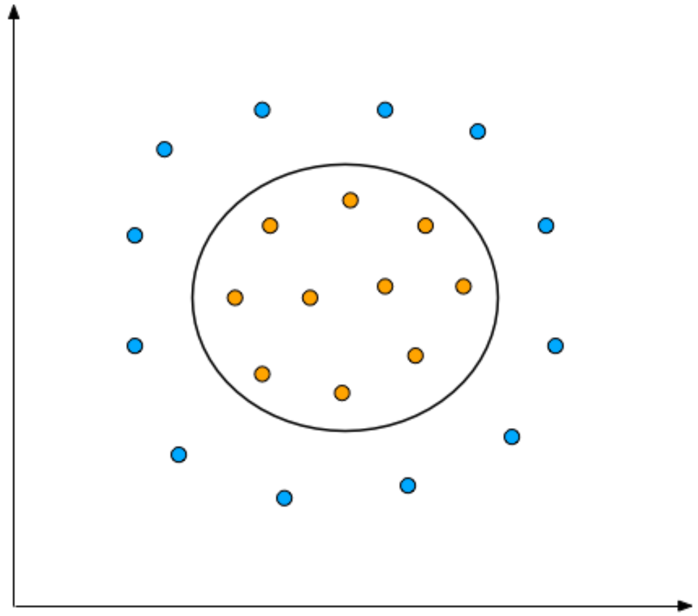
- 2 possibilities
- Use math toolboxes for minimisation problems under constraints
  - For example: SciKit Learn **`svm.SVC()`** based on the popular library libsvm
    - http://scikit-learn.org/stable/modules/svm.html
    - http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
    - https://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Practical considerations: usually less tuning to perform, libsvm is well optimised and stabilised library
- Use gradient descent approaches: compute the gradient of the loss w.r.t. the parameters **w** and apply gradient descent as usual
  - For example: SciKit Learn **`linear_model.SGDClassifier()`** with parameters: `loss='hinge'` and `penalty='l2'` . Note: instead of parameter C giving weight to the classification performance, they use parameter alpha giving weight to the regularisation term.
    - http://scikit-learn.org/stable/modules/sgd.html#sgd
    - http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
  - Practical considerations: we need to tune the learning rate, could be advantageous for very large training sets and cases where incremental learning is needed

7.3 Nonlinear problems

Nonlinear problems

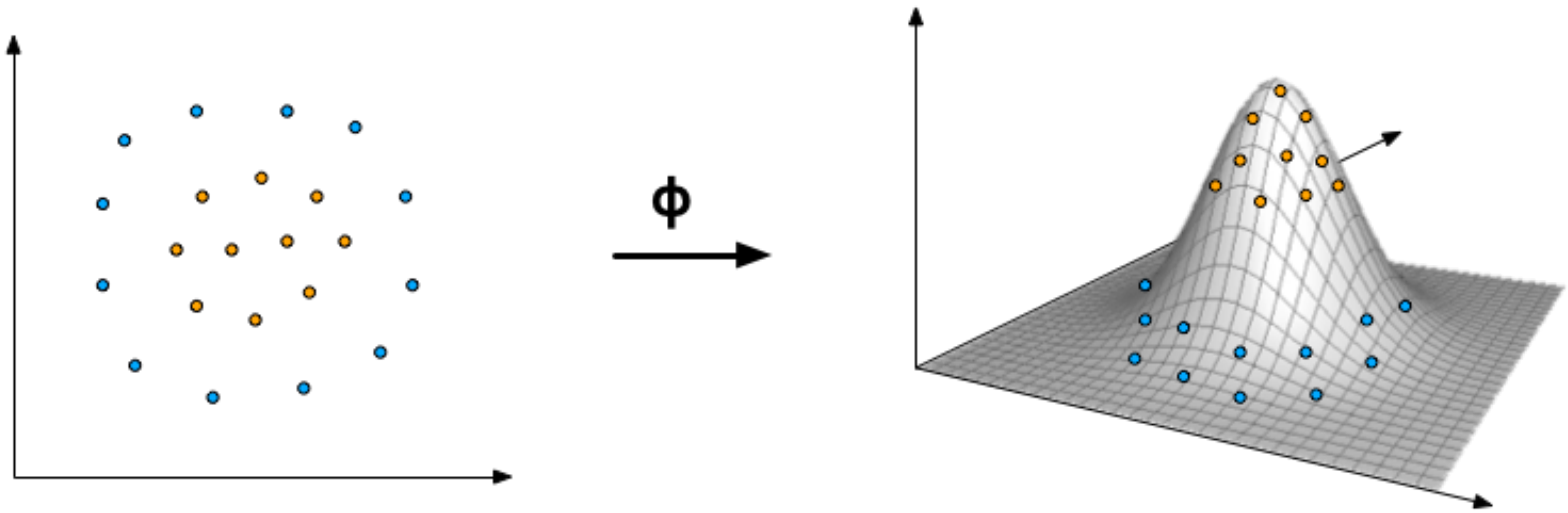
- **Problem**: how to linearly separate these 2 classes of samples?
- 2 solutions
  - move to non-linear decision boundaries by adding non-linear features to the x array and then use a linear SVM as usual - see slide 7
  - use non-linear SVMs with **kernels**
    - **see next slides**





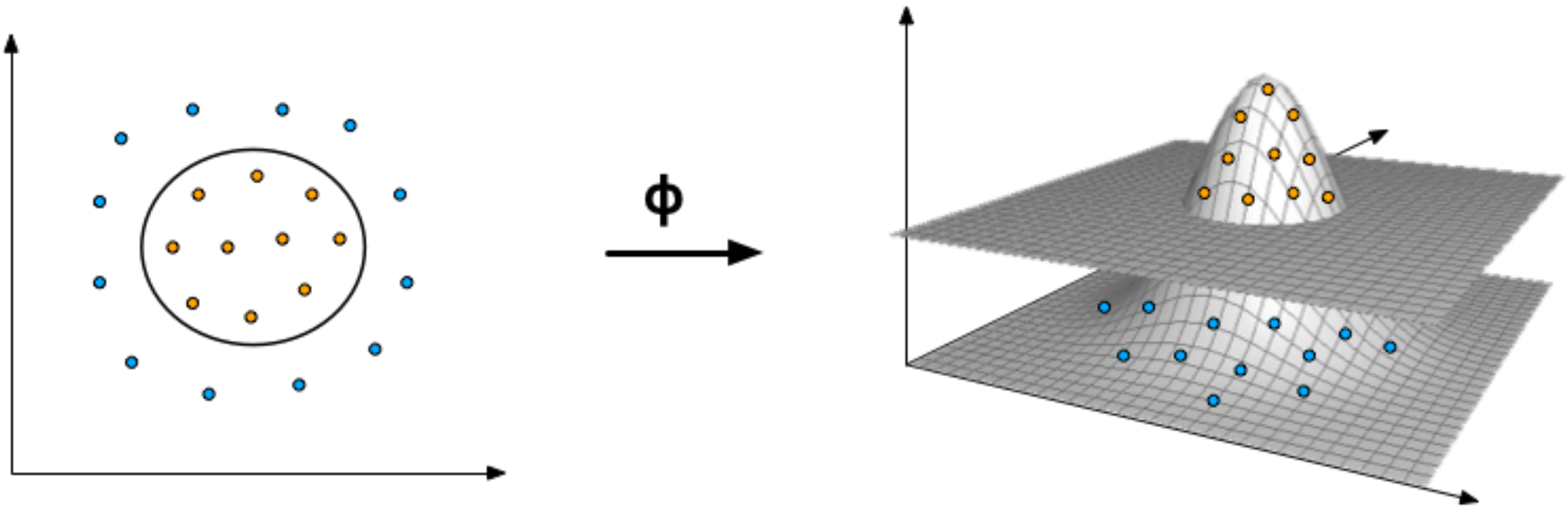
Nonlinear SVM for not linearly separable data

- **Map** the sample **input space to** a higher dimensional space (called **feature space**) with a function **ϕ**…



Nonlinear SVM for not linearly separable data

- …where samples are linearly separable by SVM !



Nonlinear SVM for not linearly separable data

- In summary, the SVM problem remains the same as before except that $\mathbf{x}$ is replaced by $\phi(\mathbf{x})$ in all equations

- The vector $\mathbf{w}$ defining the **optimal hyperplane** is found through the learning process using
$\mathbf{w} \cdot \phi(\mathbf{x}) + b = 0$

- Then a new sample $\mathbf{x_t}$ is classified by computing
$\mathrm{sign}(\mathbf{w} \cdot \phi(\mathbf{x_t}) + b)$

In red what changes from previous methods

Nonlinear SVM for not linearly separable data

- The functions ϕ are computed through *kernel functions* located at each training points $\mathbf{x_i}$
  - A new test sample $\mathbf{x_t}$ is classified by computing the sign of $\mathbf{w} \cdot \phi(\mathbf{x_t}) + b = \sum \alpha_i \, y_i \, K(\mathbf{x_i}, \mathbf{x_t}) + b$

- **Common kernels** $K(\mathbf{x_i}, \mathbf{x_j})$: (i, j = 1,…,N)
  - Linear: $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \cdot \mathbf{x_j}$
  - Polynomial: $K(\mathbf{x_i}, \mathbf{x_j}) = (\mathbf{x_i} \cdot \mathbf{x_j} + 1)^d$ (where d is the polynom degree)
  - Gaussian or radial basis function (RBF): $K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \, ||\mathbf{x_i} - \mathbf{x_j}||^2)$ (where $\gamma = 1/2\sigma^2 > 0$)
  - Hyperbolic tangent: $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(k \, \mathbf{x_i} \cdot \mathbf{x_j} + c)$ (for some $k > 0$ and $c < 0$)

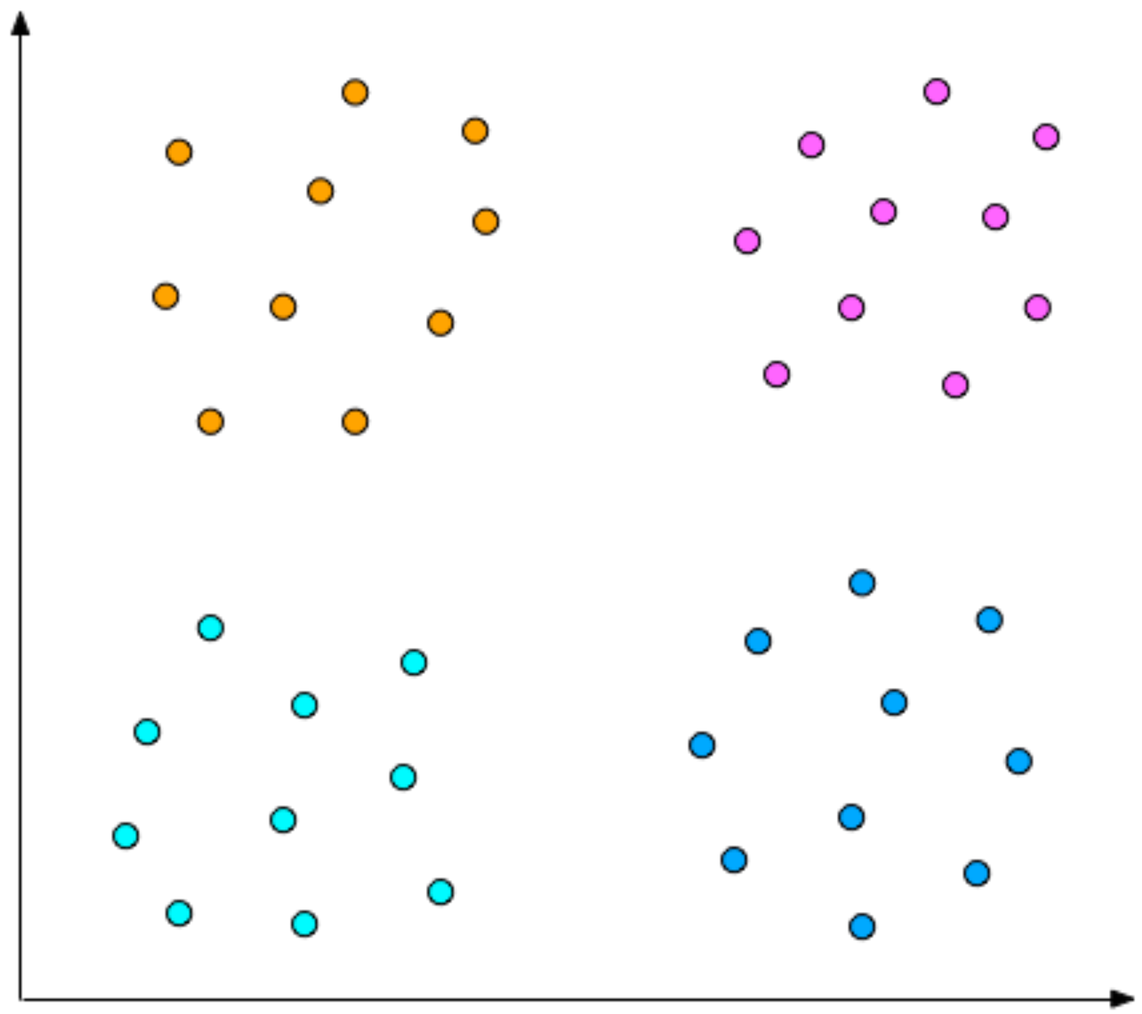
7.4 Multiclass SVM

By Balu Ertl https://upload.wikimedia.org/wikipedia/commons/5/54/Euclidean_Voronoi_diagram.svg

Multiclass SVM

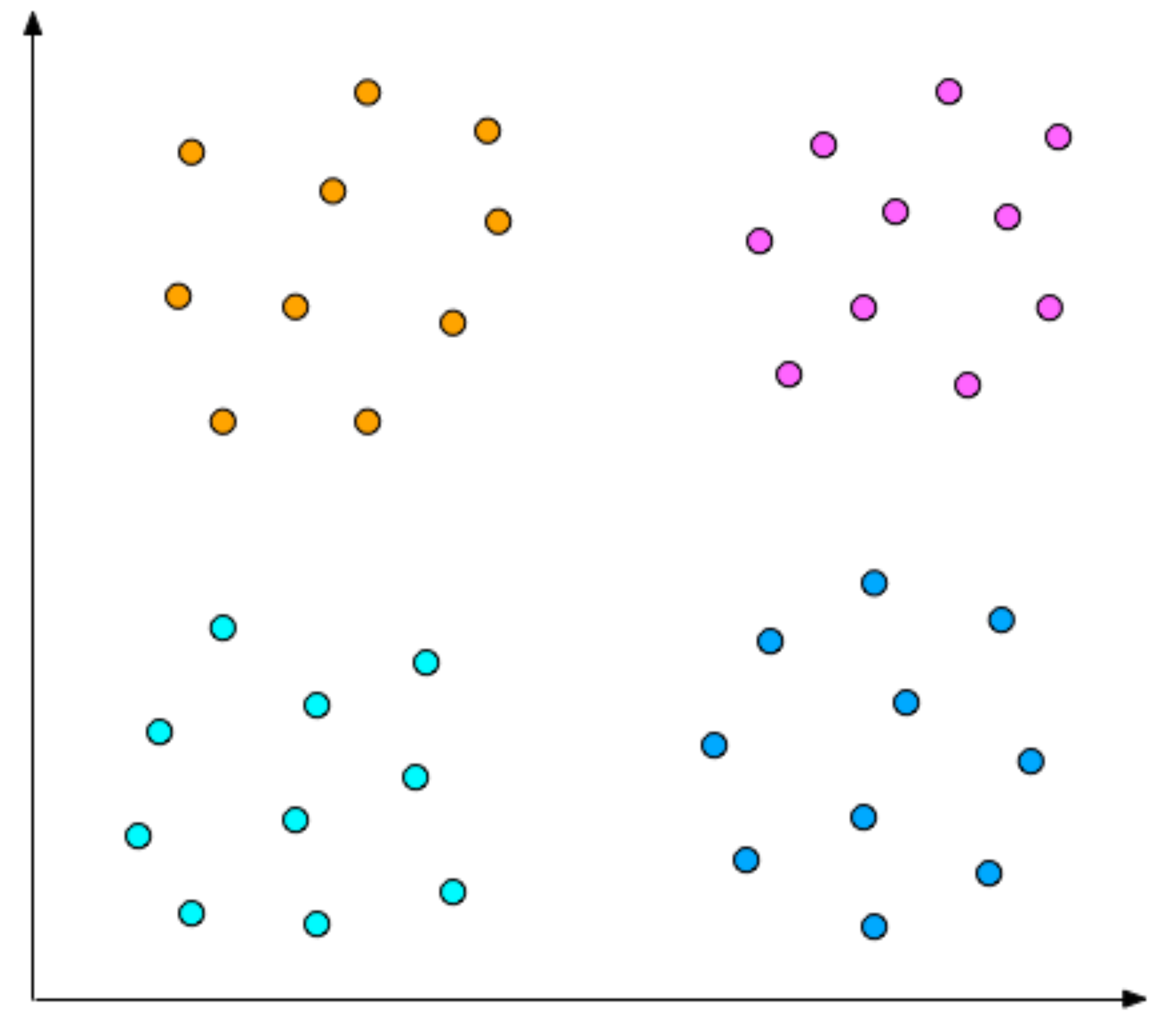
- Initially, **SVM** are a **binary classifier**, i.e. can separate 2 classes of samples

- **Problem:** how to separate multiple classes?



Multiclass SVM

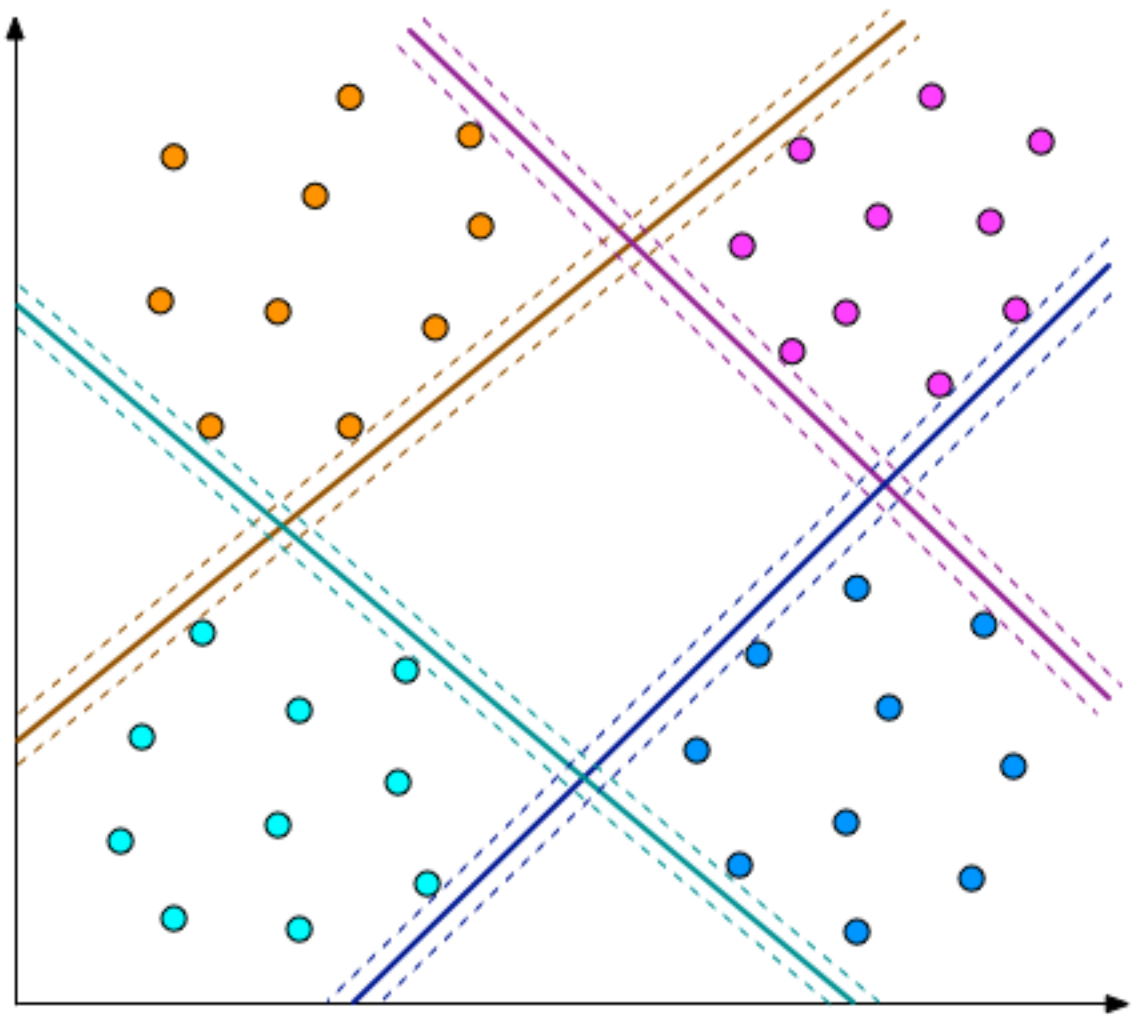
- **Solution:** reduce the single multiclass problem into **multiple binary classification problems**



Multiclass SVM

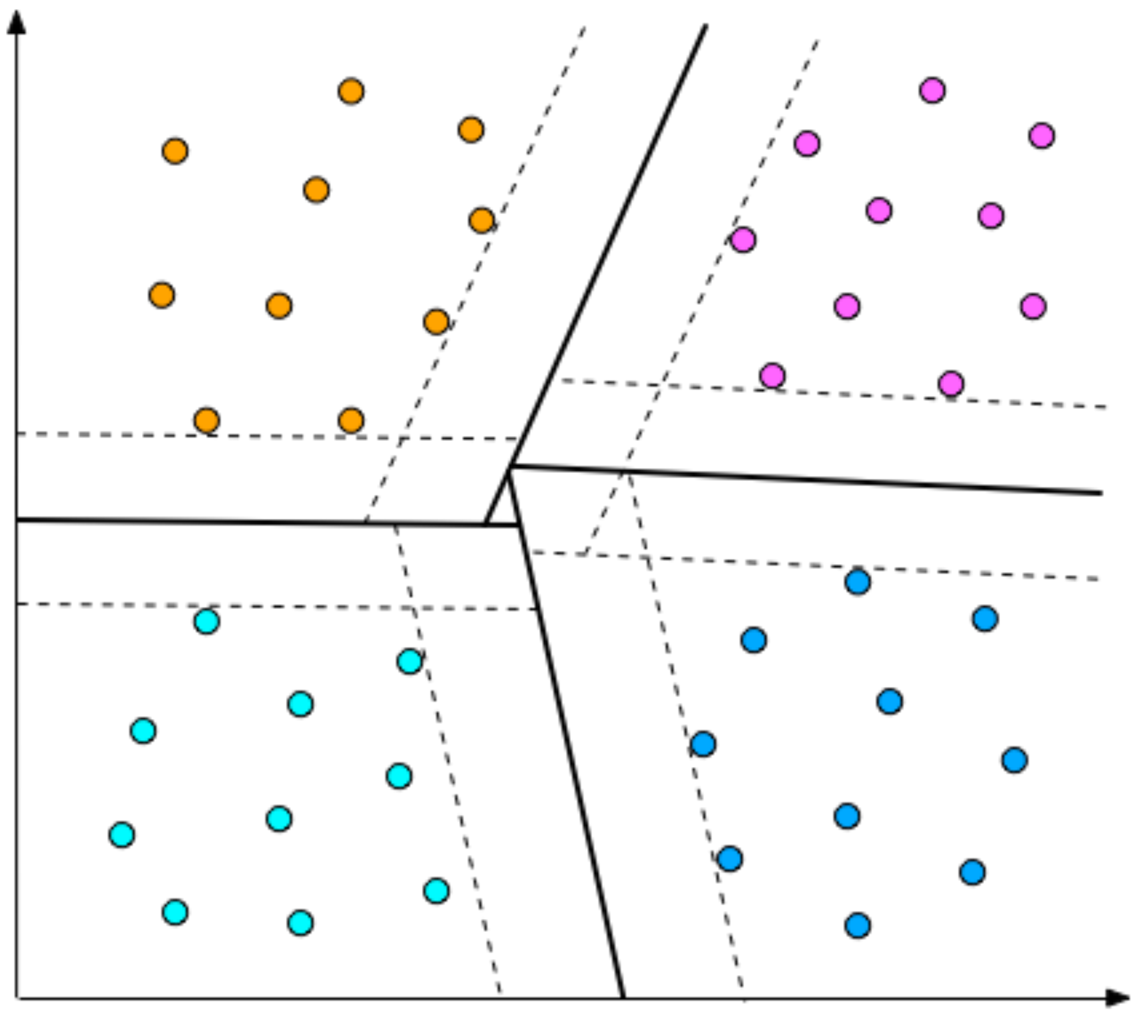
- **One-vs-all method:** the classification of new samples is done by a winner-takes-all strategy, in which **the SVM with the highest output value assigns the class to a given sample**

- For a given test sample $\mathbf{x_t}$, the output value is given by the function $\mathbf{w_c} \cdot \mathbf{x_t} + b_c$, where $c = 1,\ldots,M$ and M is the number of classes



Multiclass SVM

- **One-vs-one method:** the classification of new samples is done by a max-wins voting strategy, in which **every SVM classifier assigns a given sample to one of the two classes**, then the number of votes for the assigned class is increased by one, and finally **the class with the highest number of votes is assigned to the sample**



7.5 History & References

By Balu Ertl https://upload.wikimedia.org/wikipedia/commons/5/54/Euclidean_Voronoi_diagram.svg

History of SVM

- **1960**: Beginning of SVM development
- **1963**: Original **linear SVM** algorithm proposed by Vladimir N. Vapnik
- **1964**: **Kernel trick** first published by M. Aizerman, E. Braverman, and L. Rozonoer
- **1992**: **Nonlinear SVM** (using **kernel trick**) proposed by Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik
- **1995**: Current **standard SVM** (using **soft margin**) proposed by Vladimir N. Vapnik and Corinna Cortes
- **1996**: SVM for regression (**SVR**) proposed by Vladimir N. Vapnik
- **2008**: Vladimir N. Vapnik and Corinna Cortes received the ACM Paris Kanellakis Award for their scientifical contribution
- **Today**: SVM are widely used because of their efficiency

References

- **SVM on Wikipedia:**
  http://en.wikipedia.org/wiki/Support_vector_machine
- **SVM tutorials:**
  http://www.svms.org/tutorials
  http://www.kernel-machines.org/tutorials
- **Nice SVM presentation (with biomedical application):**
  A. Statnikov, D. Hardin, I. Guyon, C. F. Aliferis,
  A Gentle Introduction to Support Vector Machines in Biomedicine,
  http://www.med.nyu.edu/chibi/sites/default/files/chibi/Final.pdf
- **Books:**
  C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006
  R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification (2nd Ed.)*,
  Wiley-Interscience, 2001