Hewlett Packard Enterprise
Data Science Institute
UNIVERSITY OF HOUSTON

311 Introduction to Machine Learning

Summer 2024

Instructor: Ioannis Konstantinidis

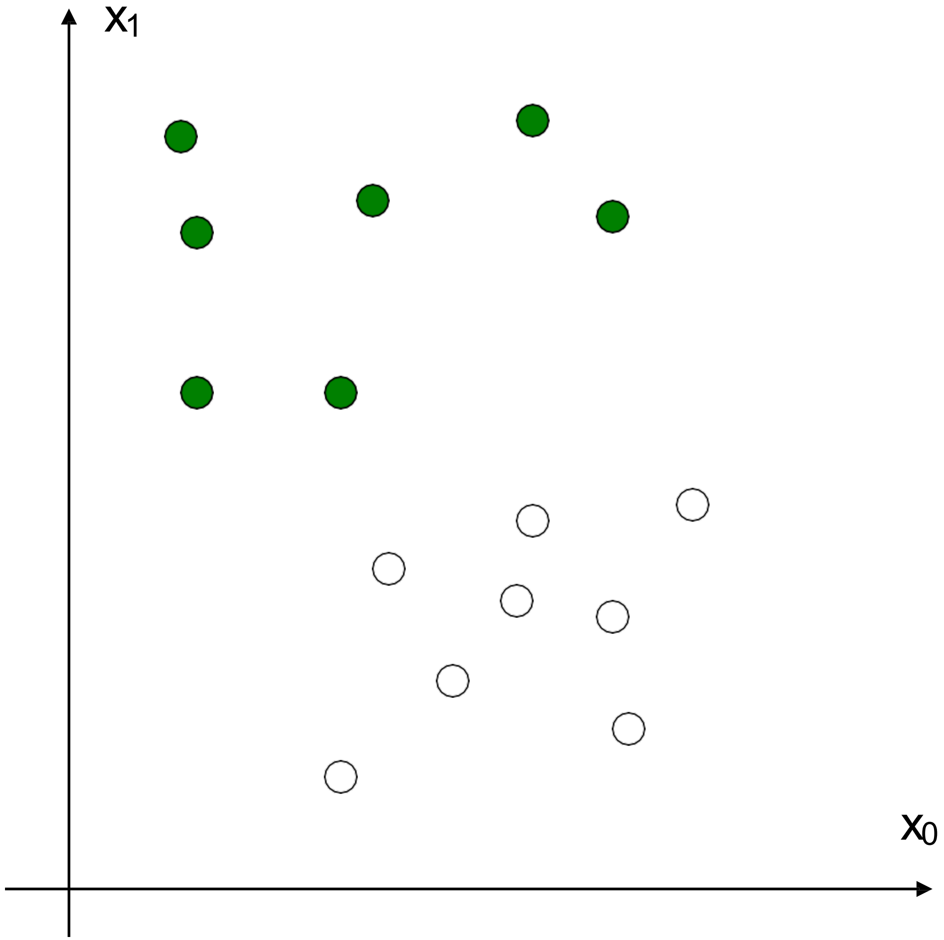
Overview

Support Vector Machines

Generalizations

- Kernels
- Regularization

How would you classify these points to minimize error?



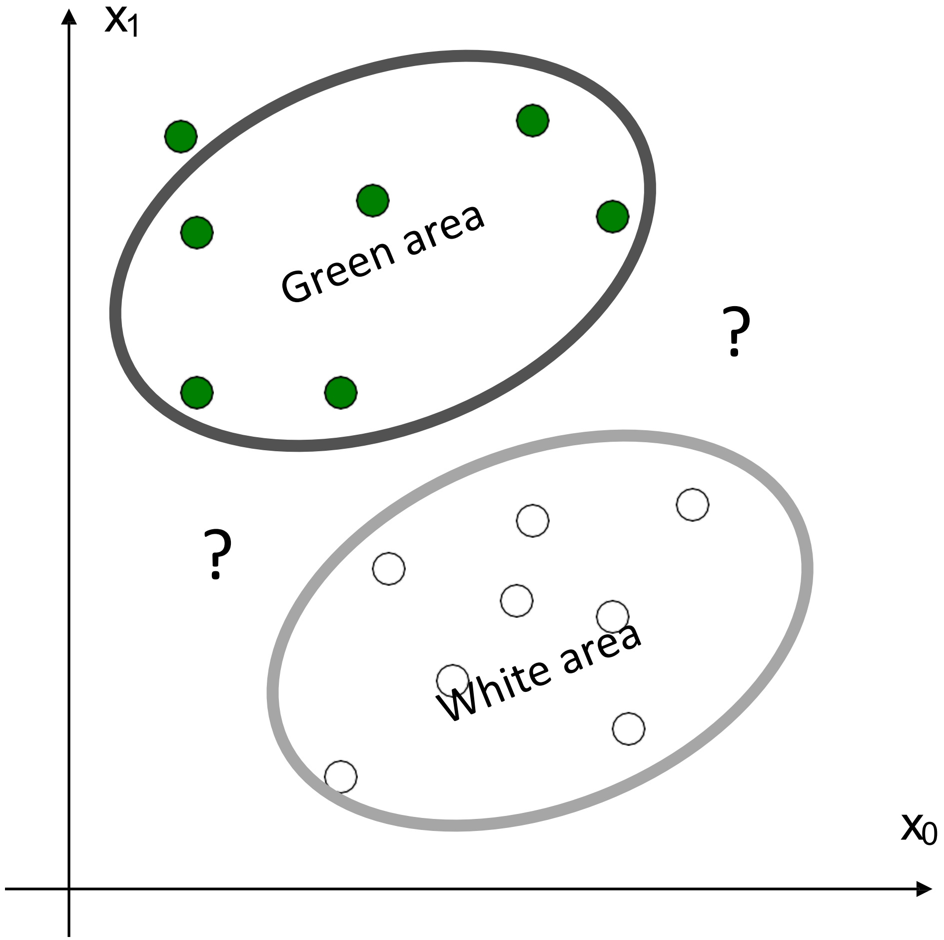
k-NN

How would you classify these points to minimize error?

- Match to the color of the nearest neighbors

$$g(\mathbf{x}) = \sum_{i \in \mathrm{kNN}(\mathbf{x})} weight(\mathbf{x}_i, \mathbf{x})\, y_i$$

- Computation based on only k training points, but they differ based on where the test point is located
- Distant points (outliers?) don't affect decision

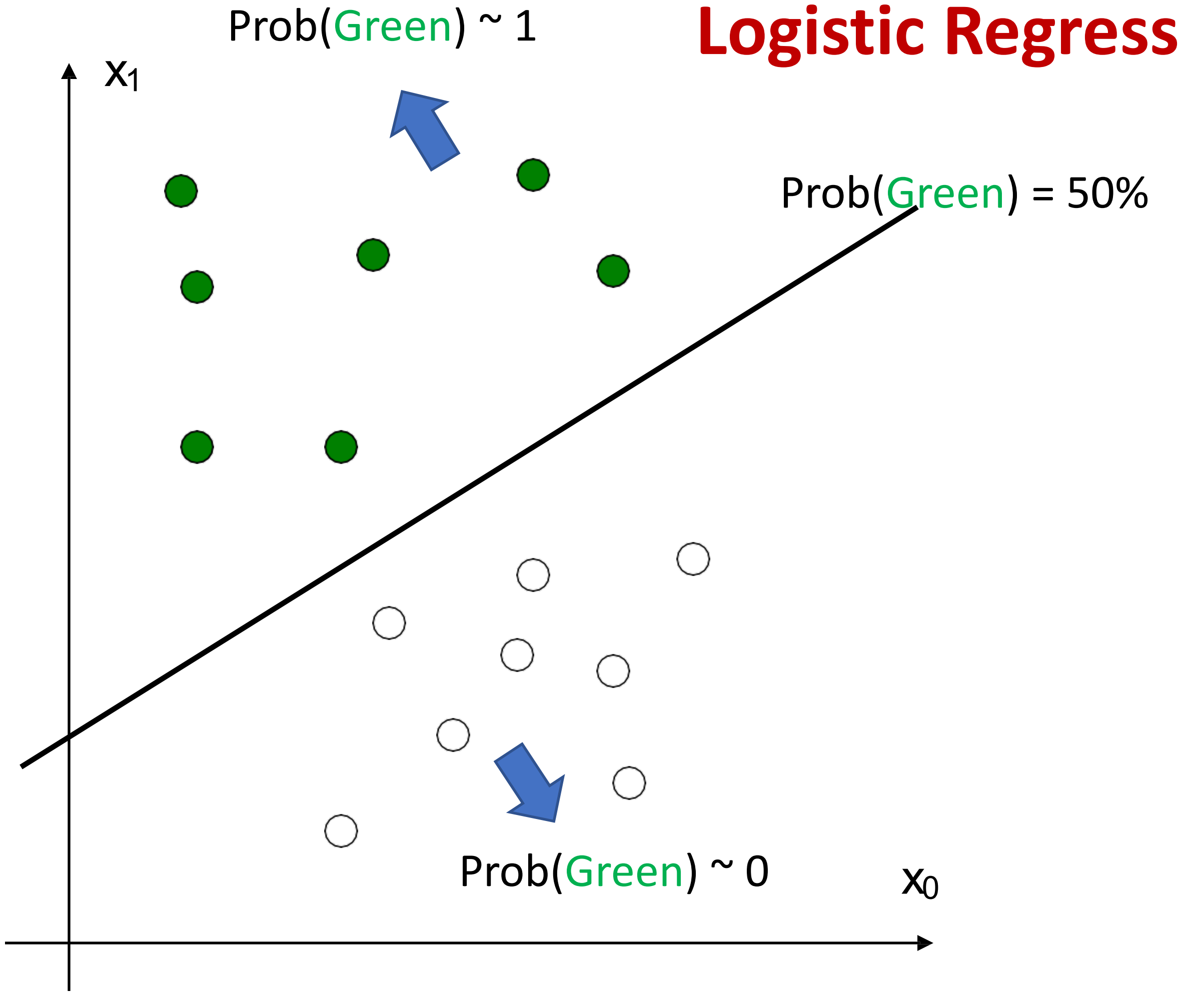


Logistic Regression

How would you classify these points to minimize error?

- Compute probability of match
- Computation based on ALL training points

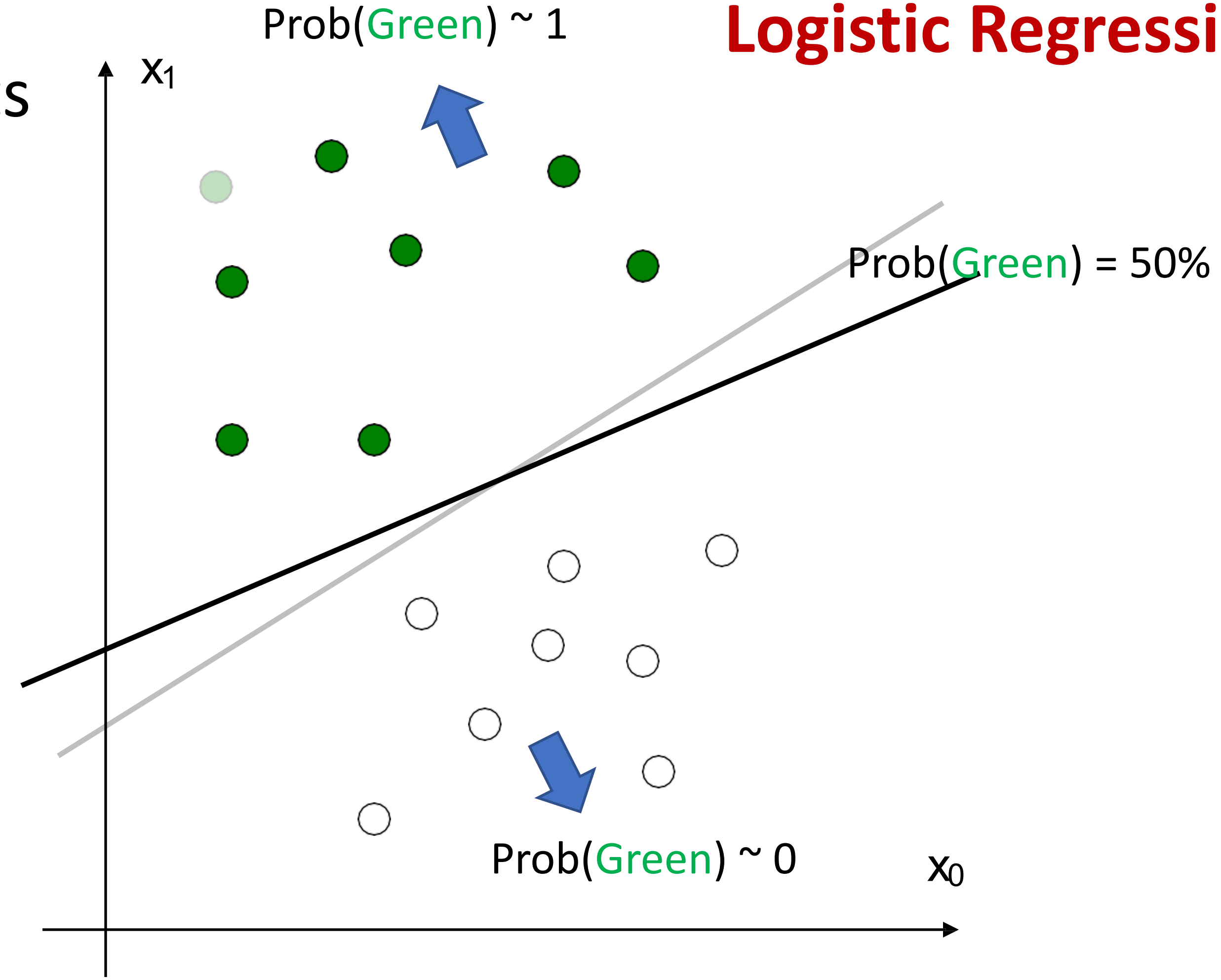
$$Prob(Green) \sim w_0 x_0 + w_1 x_1 + b = \mathbf{w}^T \mathbf{x} + b$$



Logistic Regression

How would you classify these points to minimize error?

- Compute probability of match
- Computation based on ALL training points
- Distant points (outliers?) affect probability

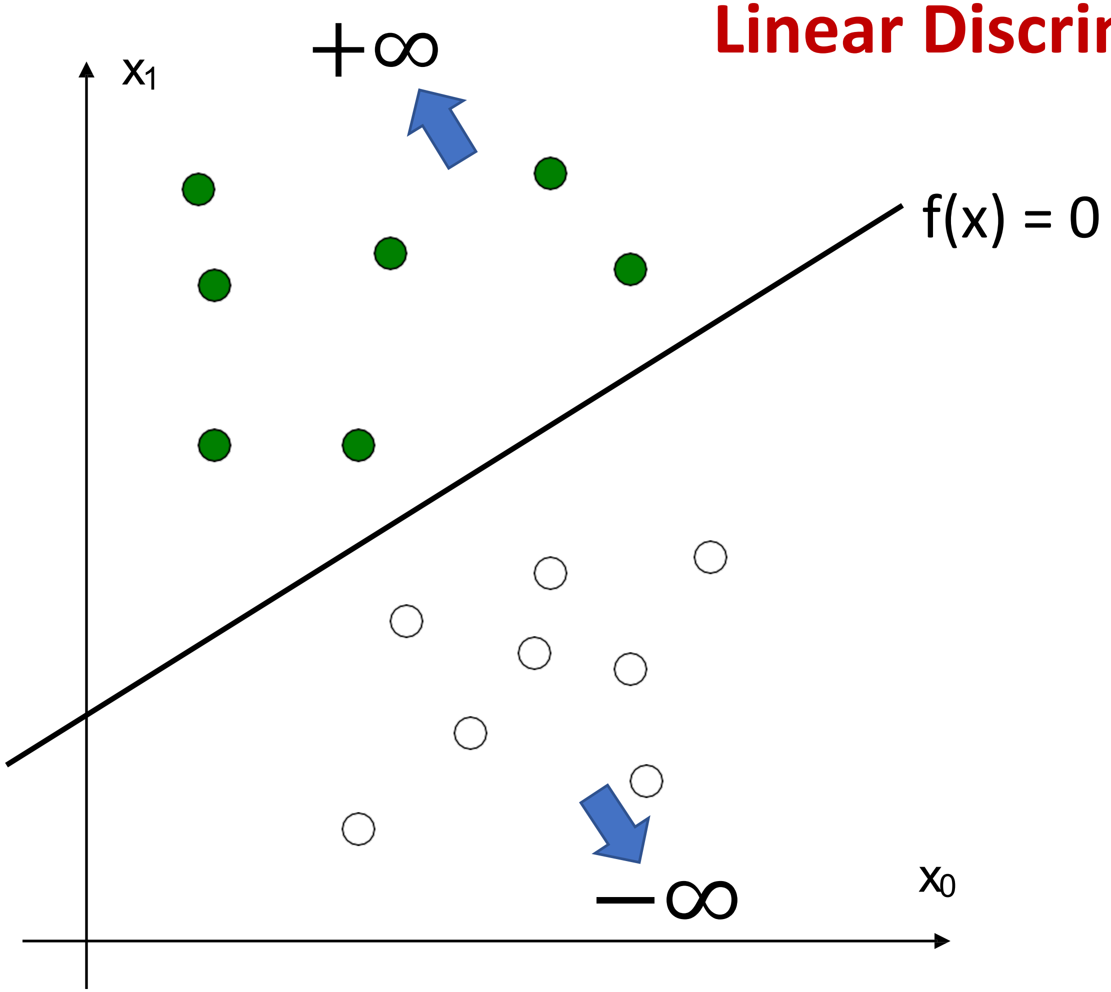


Hewlett Packard Enterprise
Data Science Institute
UNIVERSITY OF HOUSTON

New method: Support Vector Machines

Linear Discriminant

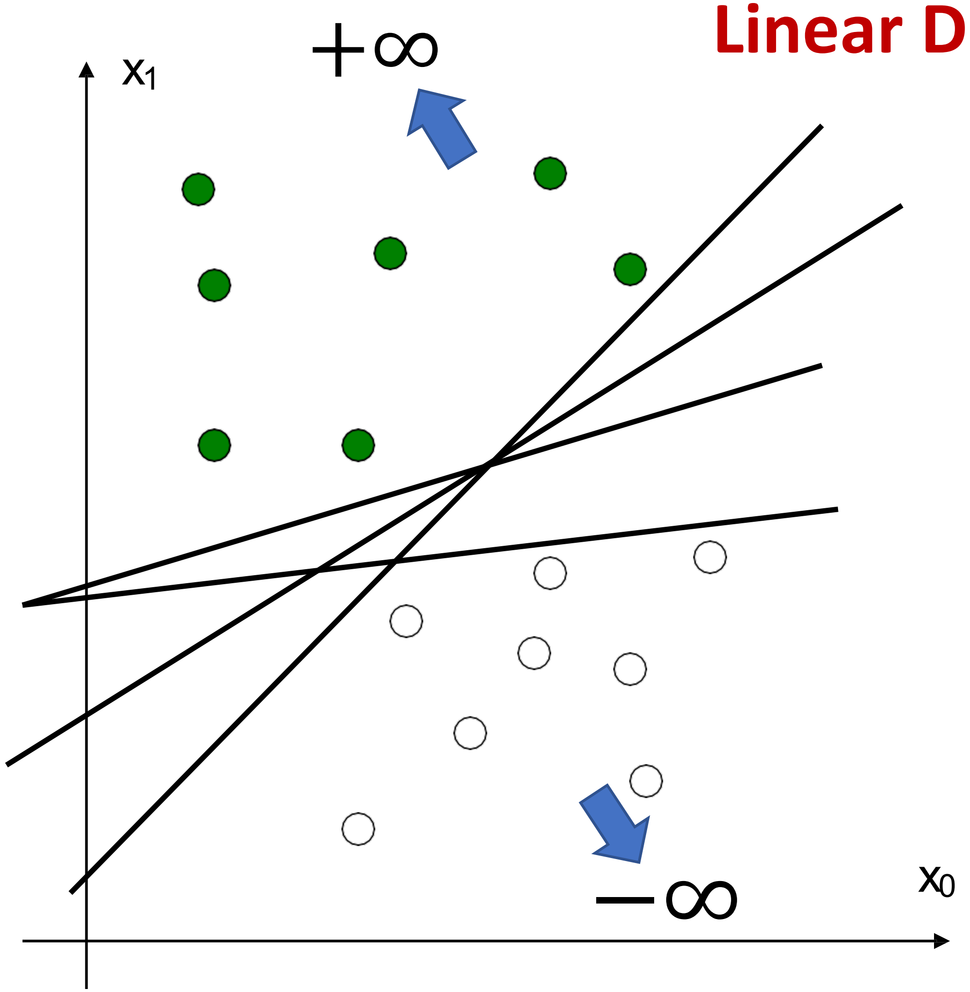
New method:



- Use a linear function f(x)= $\mathbf{w}^T\mathbf{x} + b$ as boundary (signed distance)
- Binary cut-off, not a probability

Linear Discriminants

New method:

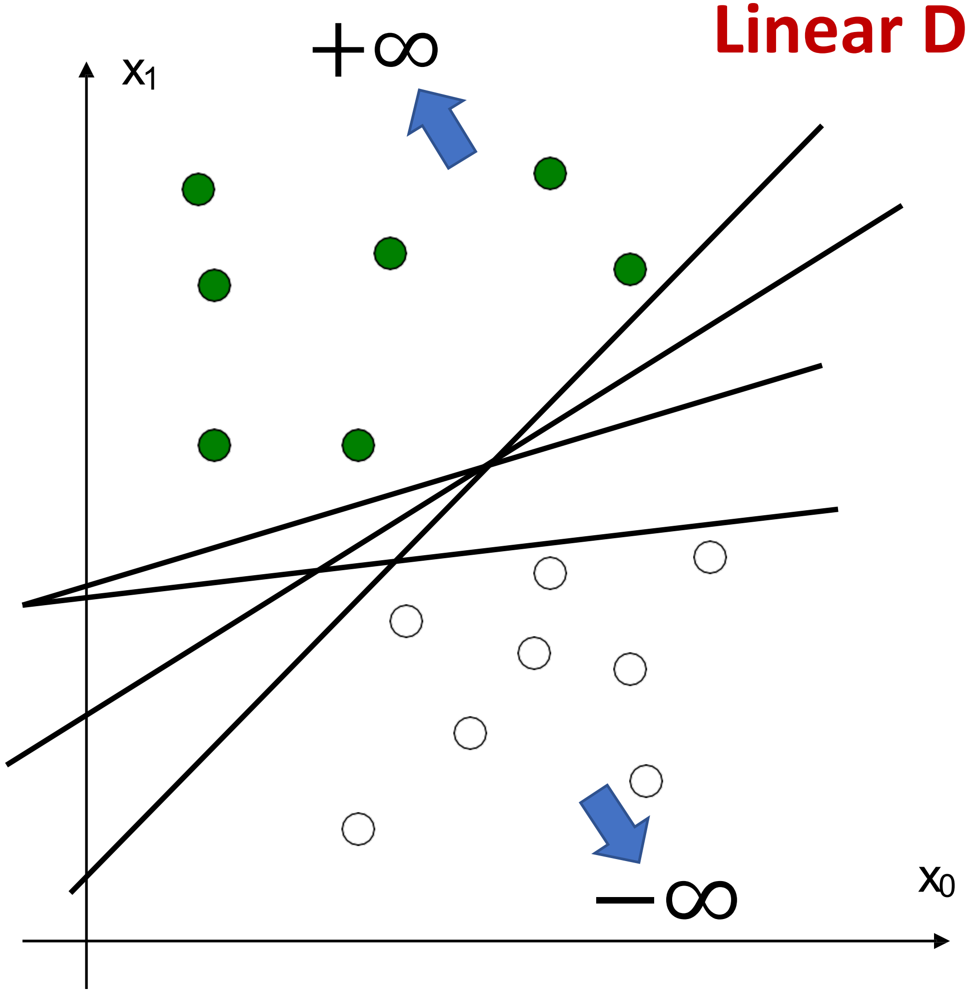


- Use a linear function f(x)= $\mathbf{w}^T \mathbf{x} + b$ as boundary (signed distance)

- MANY choices! (infinitely many)

Linear Discriminants

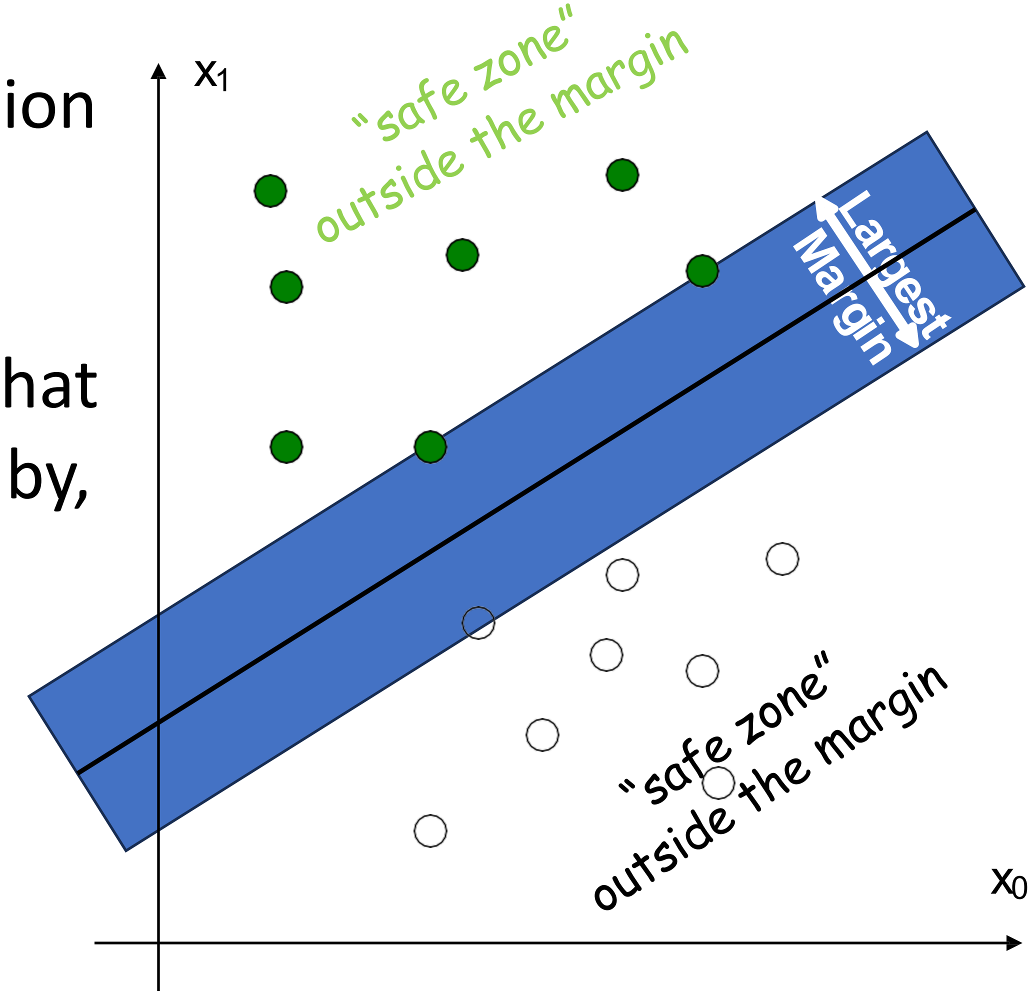
New method:



- Use a linear function f(x)= $\mathbf{w}^T\mathbf{x} + b$ as boundary (signed distance)
- MANY choices! (infinitely many)
- How to pick one?

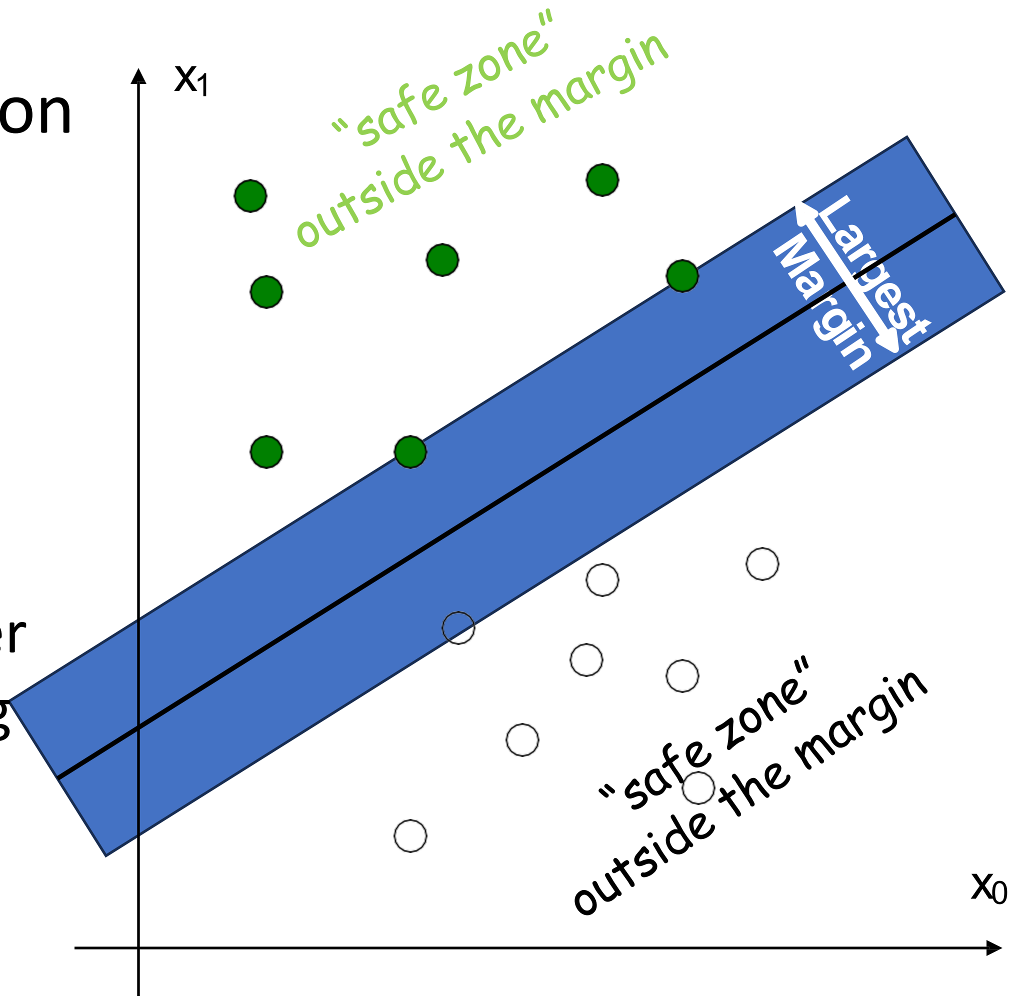
Pick the **linear discriminant** function with the **largest margin**

- Margin is defined as the width that the boundary could be increased by, before hitting a data point

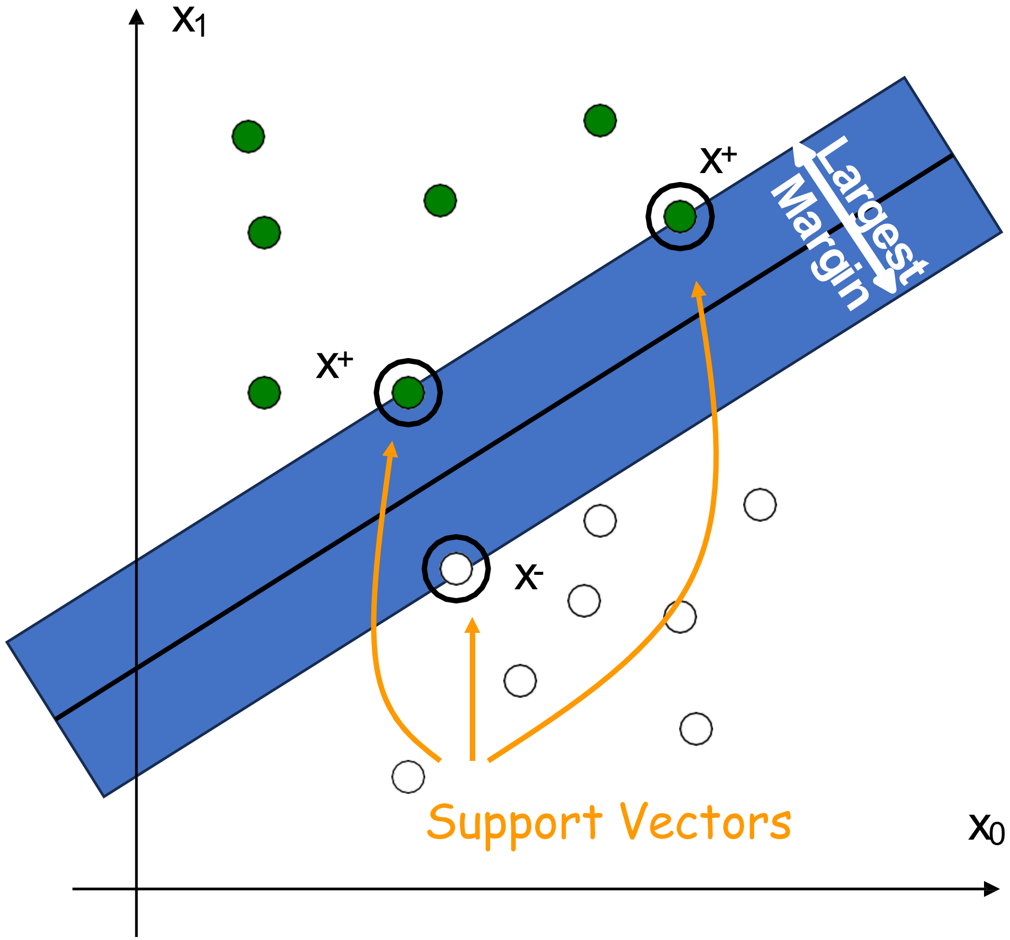


Pick the **linear discriminant** function with the **largest margin**

- Computation based only on a few “difficult” points that are near the boundary
- Robust to outliners (moving any other point does not change the separating line) and thus strong generalization ability

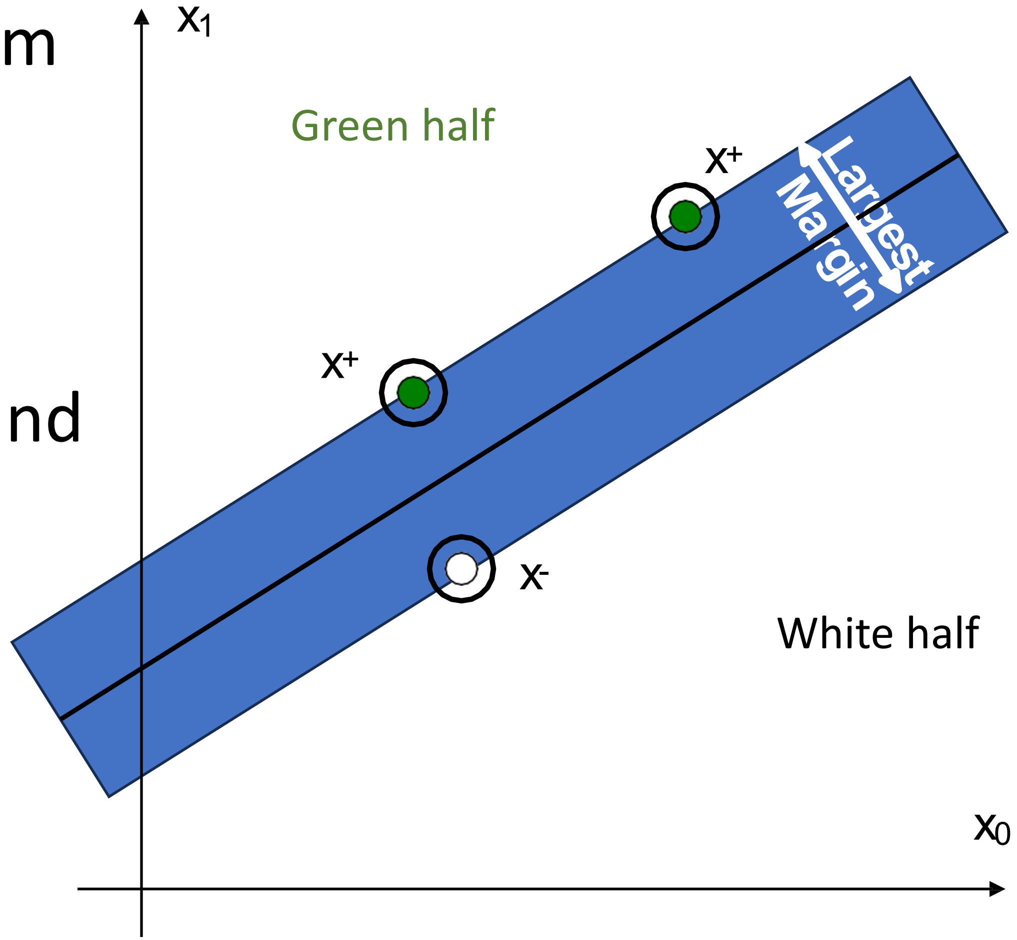


These data points that define the margin are called **support vectors**



- The data points further away from the margin do not count

- Fitting the model is about identifying the support vectors and throwing away the rest



Points on the decision boundary:

$$\mathbf{f(x)} = \mathbf{w}^T \mathbf{x}^+ + b = 0$$

Points elsewhere:

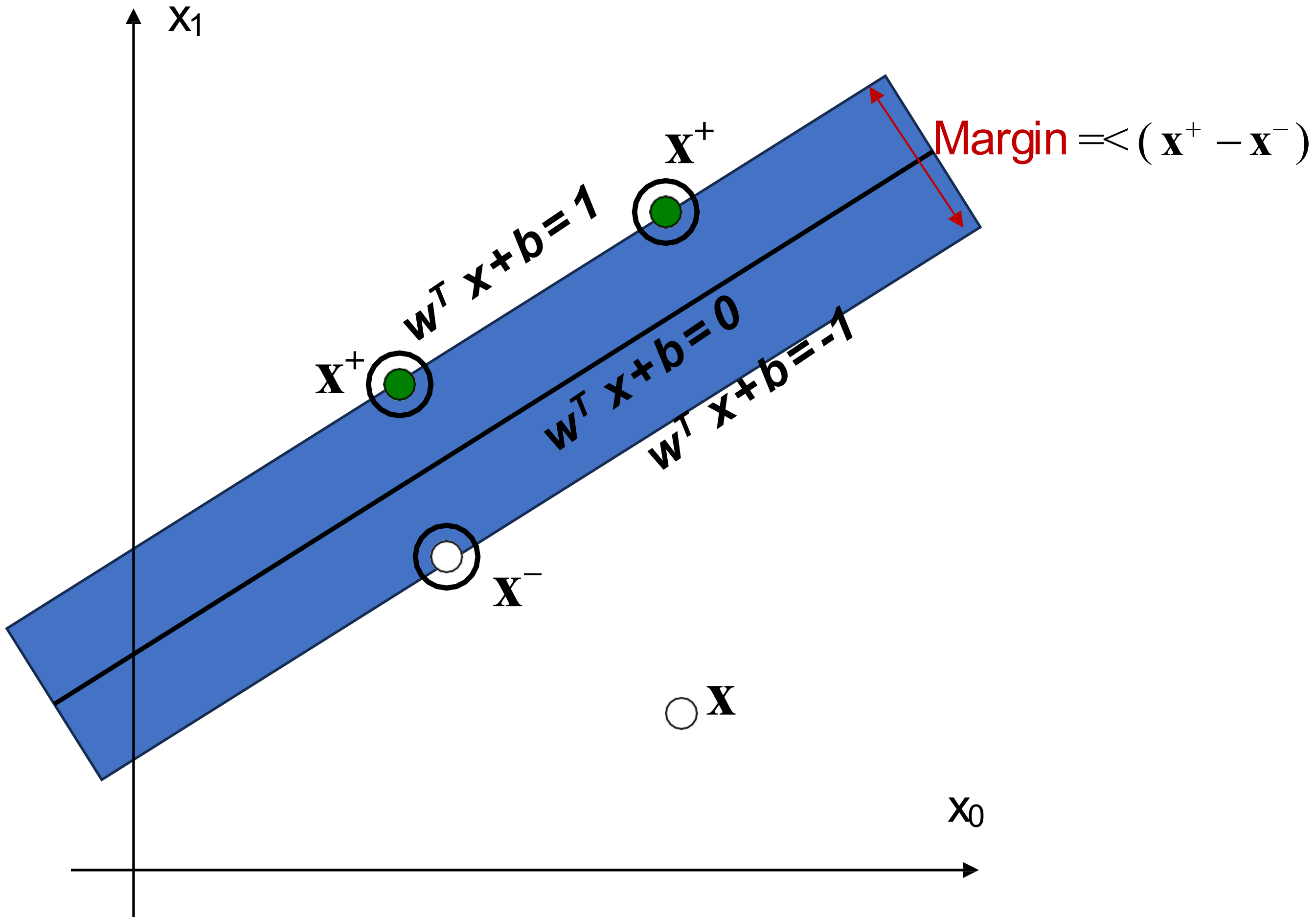
$\mathbf{w}^T \mathbf{x} + b > 0$ implies label=green

$\mathbf{w}^T \mathbf{x} + b < 0$ implies label=white

Points on the edge of the margin (support vectors):

$$\mathbf{w}^T \mathbf{x}^+ + b = 1$$

$$\mathbf{w}^T \mathbf{x}^- + b = -1$$



Developed by Ioannis Konstantinidis

Optimization Problem: computing w

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{s.t.} \quad y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$

Lagrangian Function

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \lambda_i \left(y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 \right)$$

$$\text{s.t.} \quad \lambda_i \geq 0$$

In the end:

$$\mathbf{w} = \sum_{i \in \mathrm{SV}} \lambda_i\, y_i\, \mathbf{x}_i$$

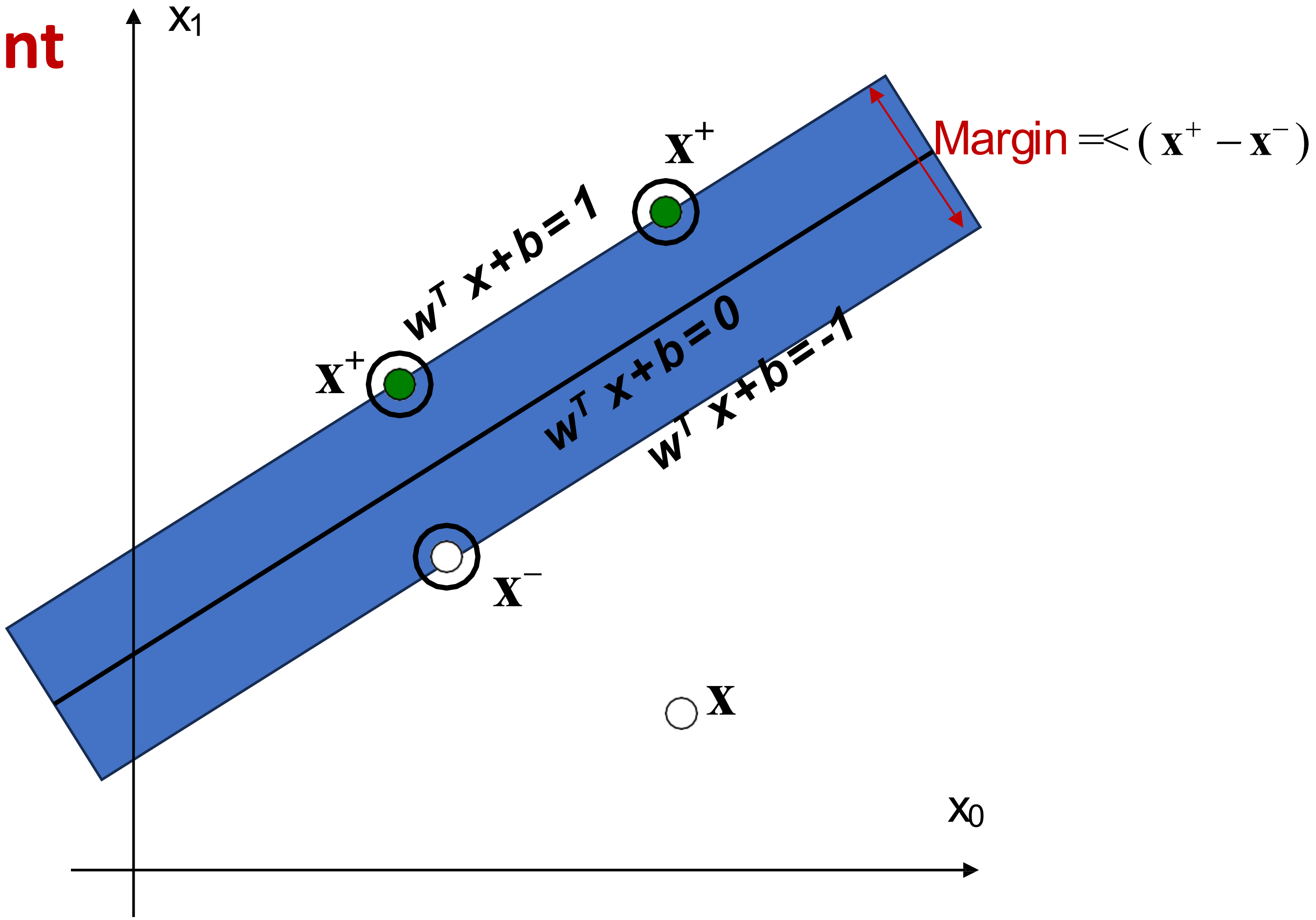
The **largest margin linear discriminant** function:

$$\mathbf{w} = \sum_{i \in \mathrm{SV}} \lambda_i \, \mathbf{x}_i y_i$$

$$\mathbf{w}^T \mathbf{x} + b = \sum_{i \in \mathrm{SV}} \lambda_i \, \mathbf{x}_i^T \mathbf{x} \, y_i + \mathbf{b}$$

$\mathbf{w}^T \mathbf{x} + b > 0$ implies label=green

$\mathbf{w}^T \mathbf{x} + b < 0$ implies label=white



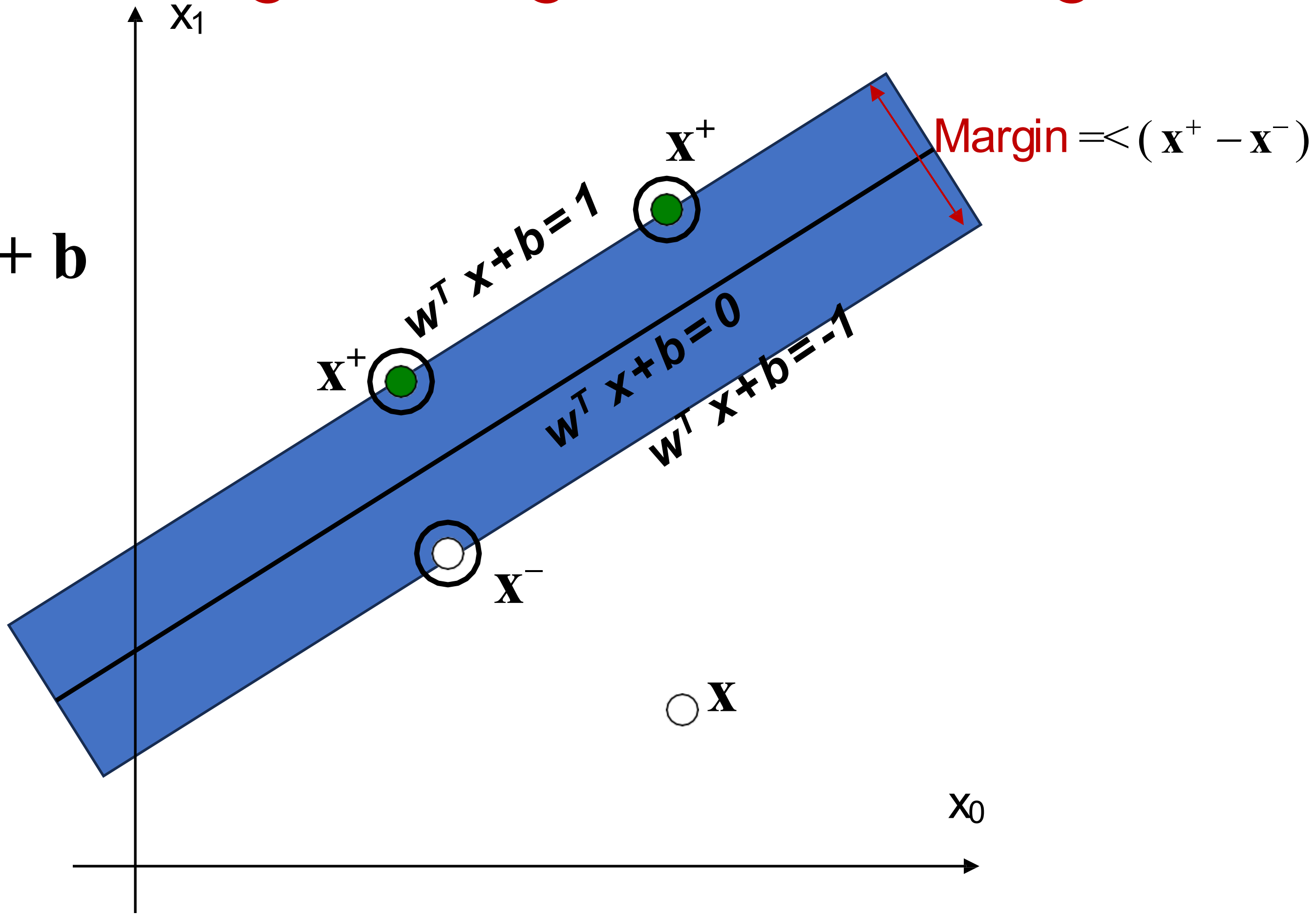
Largest Margin Linear Discriminant = Weighted Neighbors on the Margin

Similar to kNN, it is a weighted average of labels

$$\sum_{i\in \mathrm{SV}} \lambda_i \, \mathbf{x}_i^T \mathbf{x} \, y_i + \mathbf{b} = \sum_{i\in \mathrm{SV}} weight(\mathbf{x}_i,\mathbf{x}) \, y_i + \mathbf{b}$$

Similar to Logistic Regression, it is a linear classifier $\mathbf{w}^T\mathbf{x} + b$

But we only consider the training points that define the margin, not the training points close to the test point



SVM = Largest Margin Linear Discriminant = Weighted Neighbors on the Margin

For training

Data points outside the margin are redundant:

- all get a zero weight, and
- support vectors get to represent all of them in the voting process

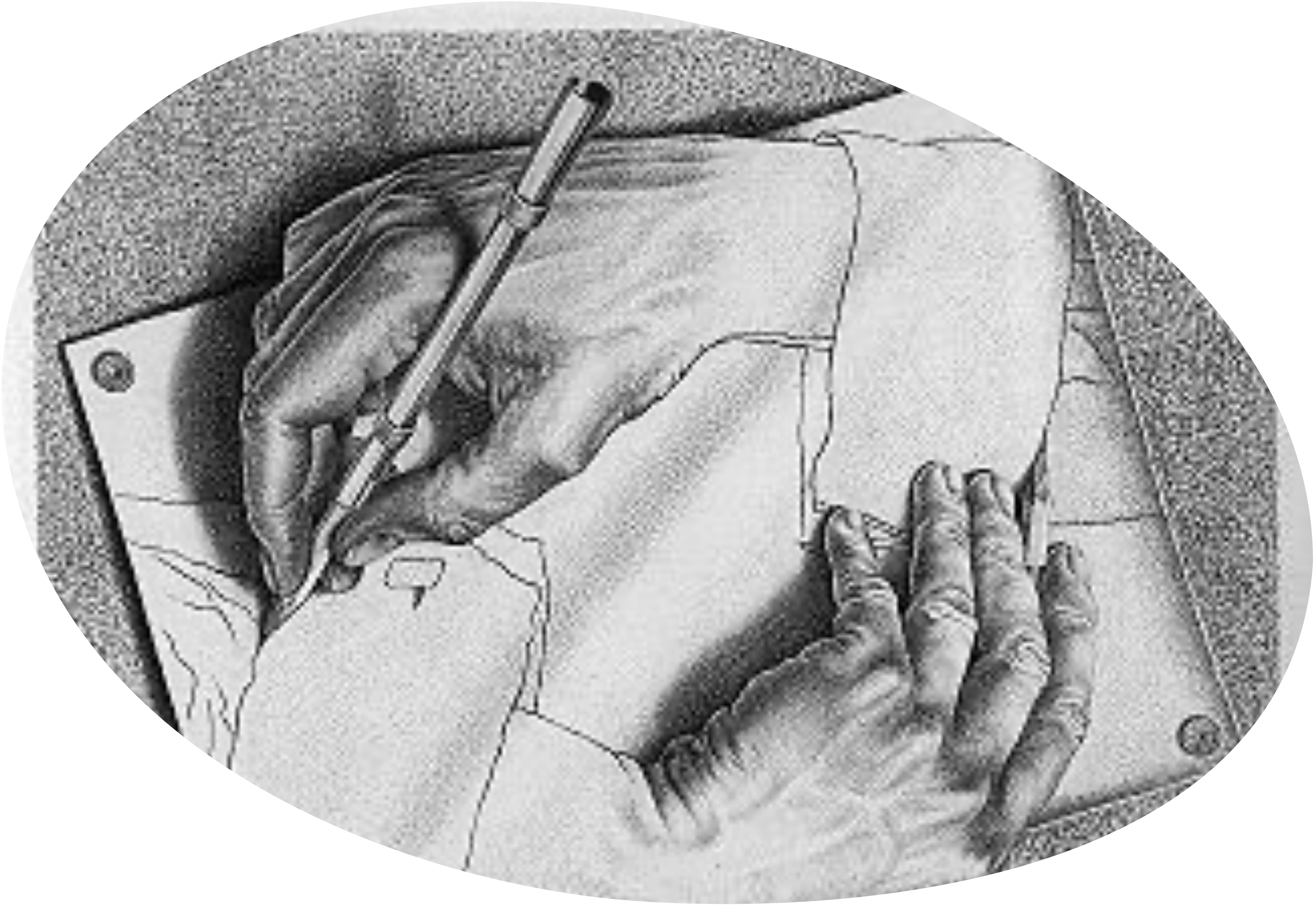
Data points on the margin boundary are important:

- they become support vectors on either side of the boundary margin

For testing

Support vectors that are similar to the test point count more

- If $\mathbf{x}_i^T\mathbf{x}$ is small, it does not contribute much to the sum

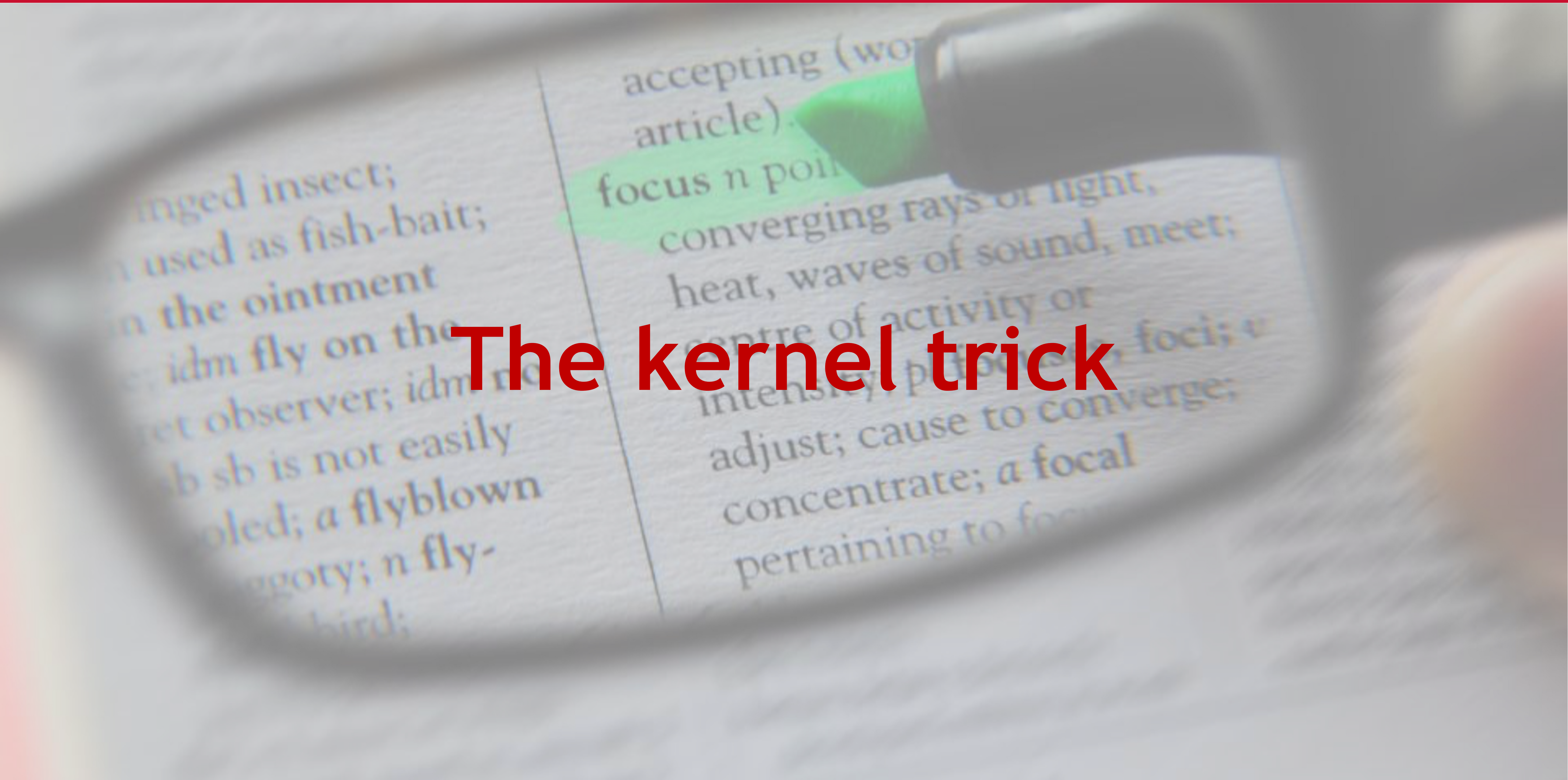


Hands-on Example:

SVC

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

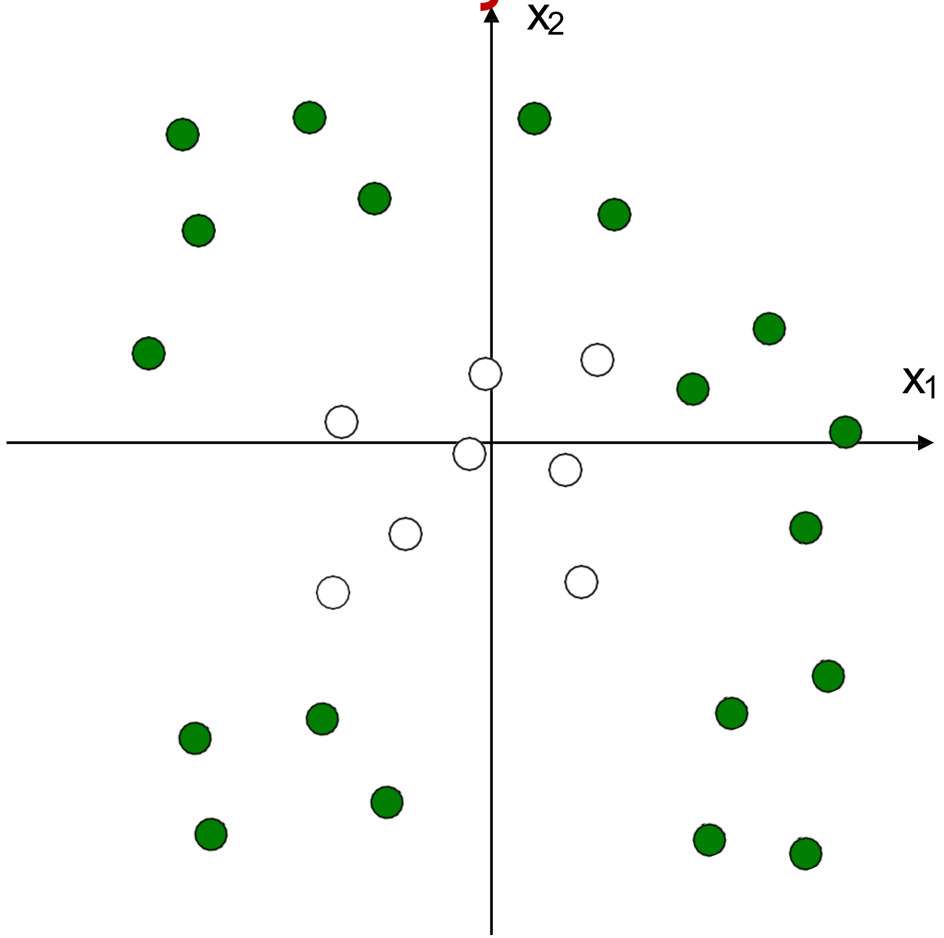
The kernel trick



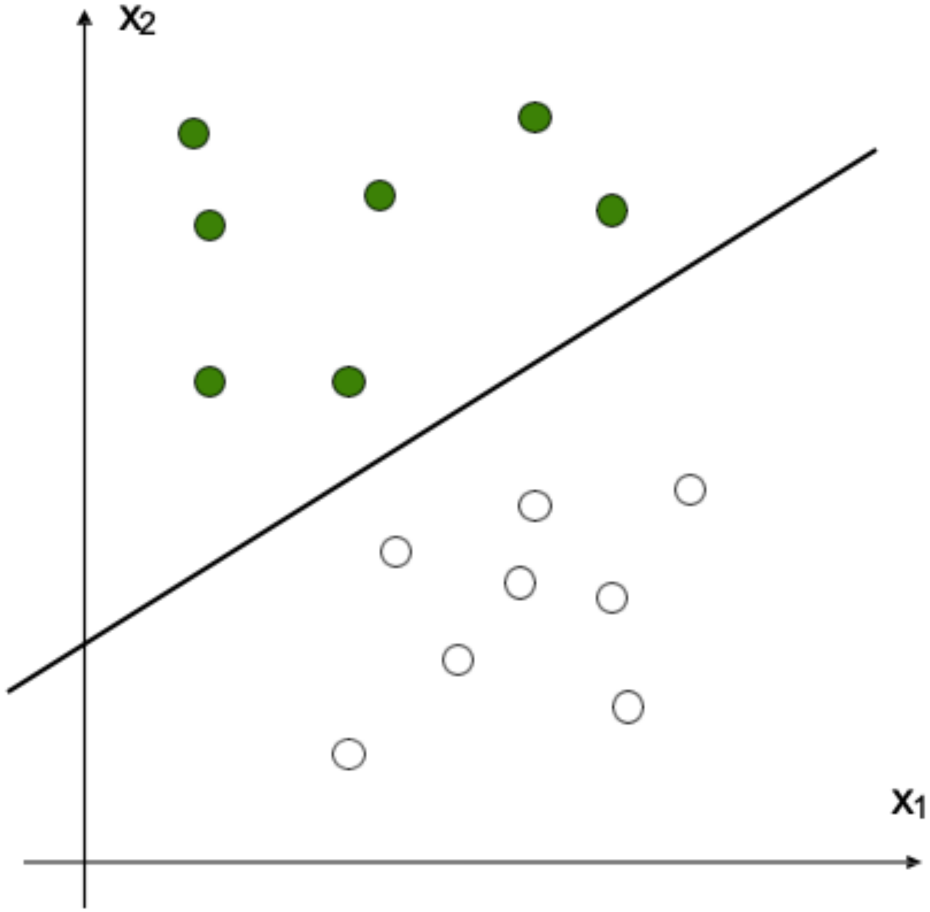
What if the points don't line up exactly?

Hewlett Packard Enterprise
Data Science Institute
UNIVERSITY OF HOUSTON

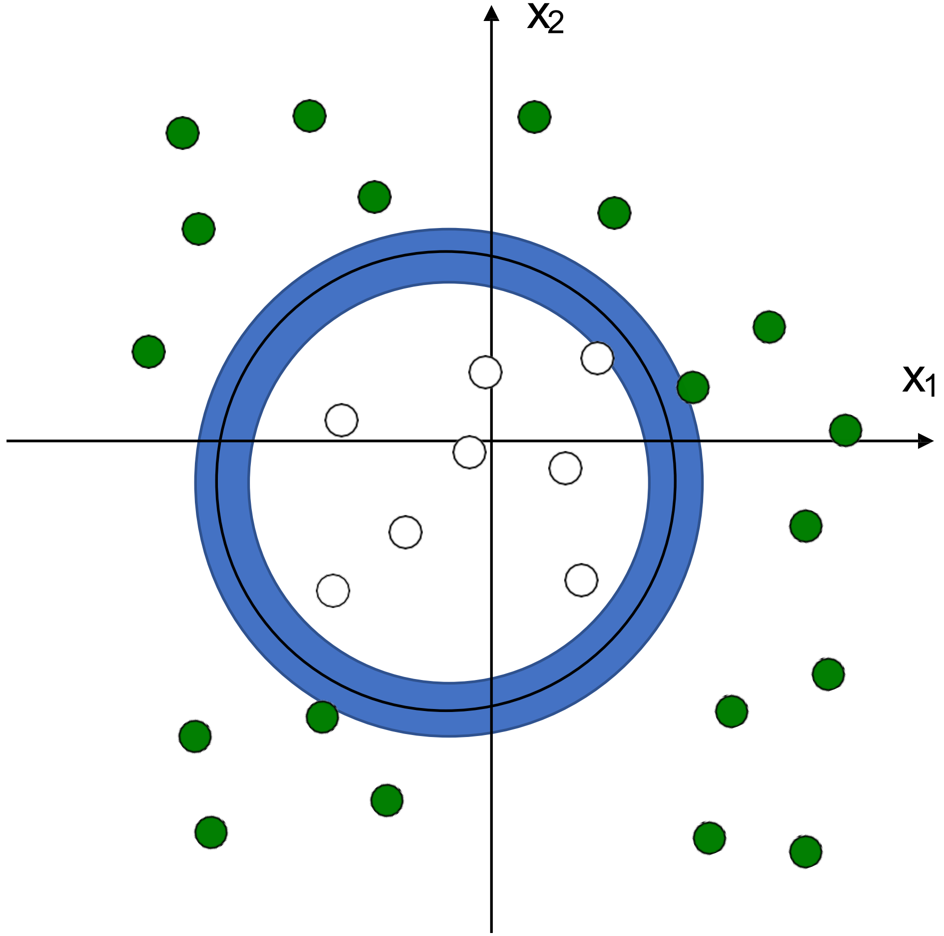
Non-linearity



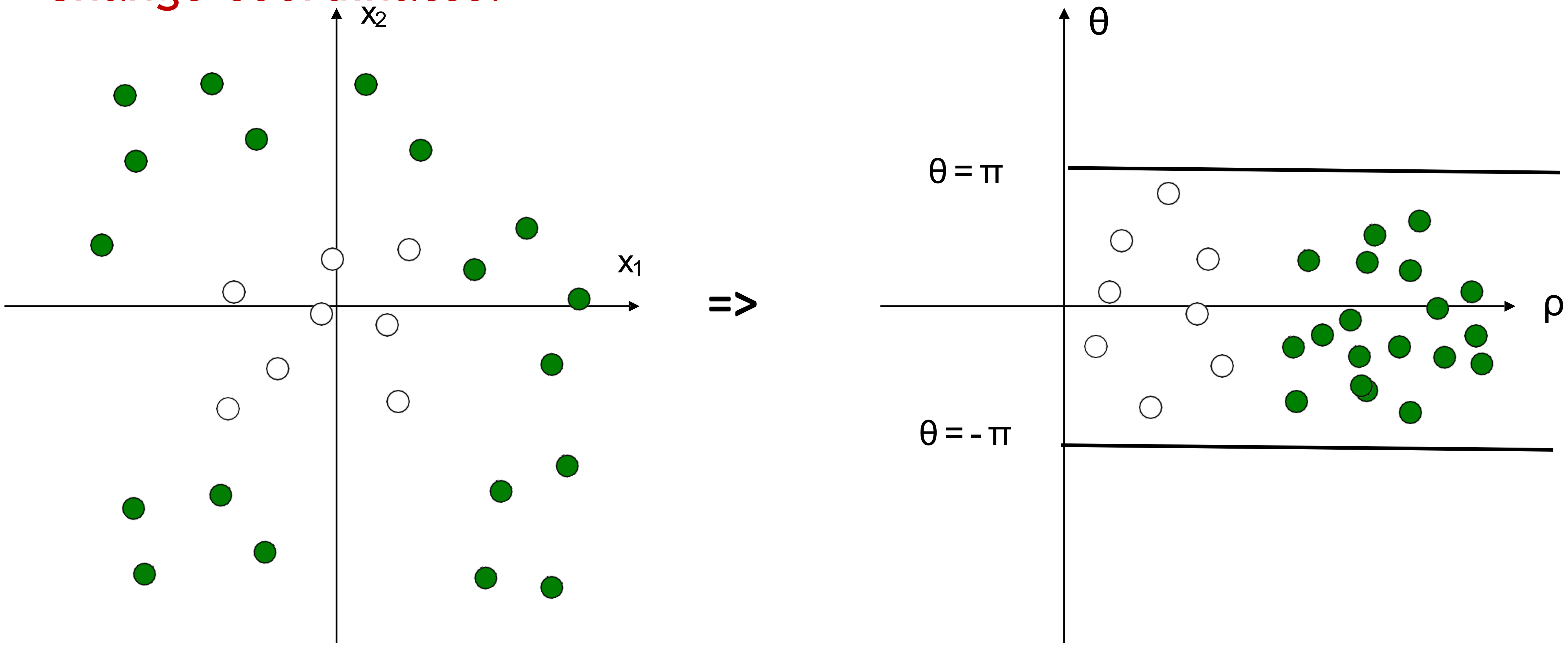
vs.



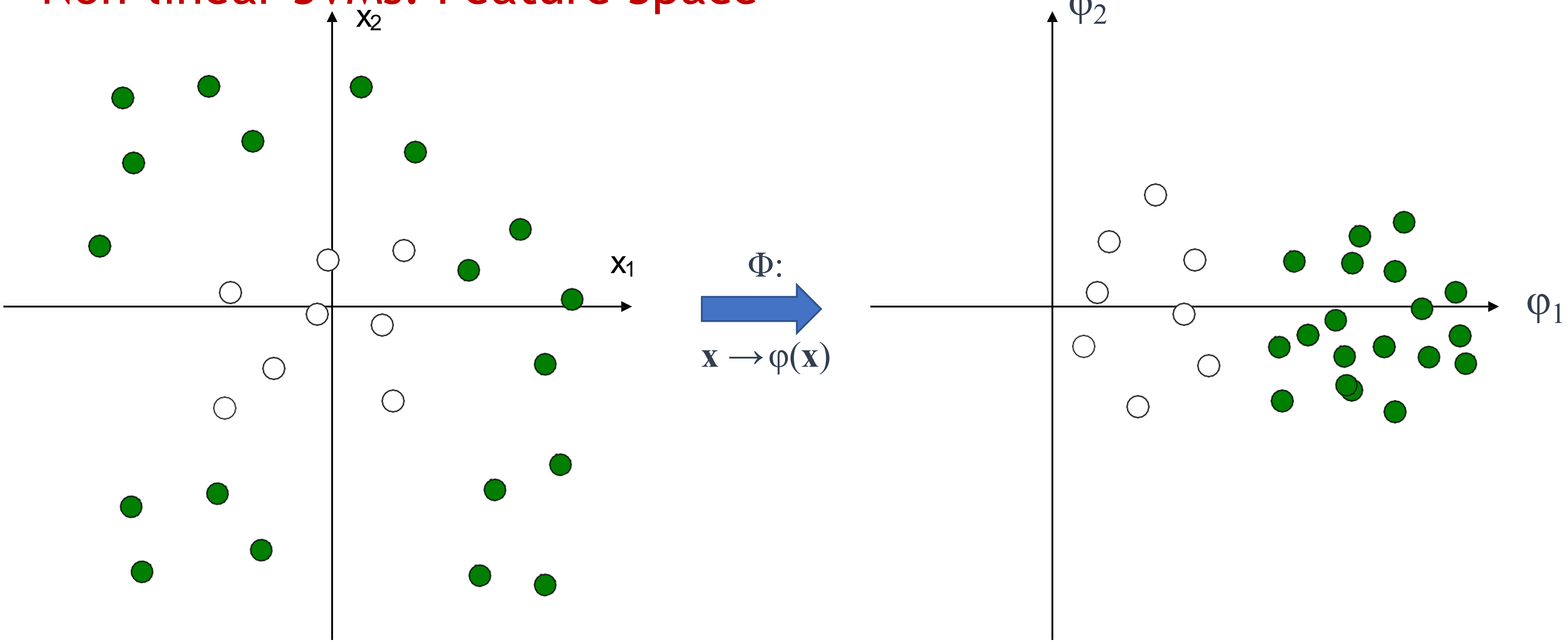
Circle Machines / Discriminant Analysis?



Change coordinates!



Non-linear SVMs: Feature Space



General idea: the original input space can be mapped to some transformed feature space where the training set is linearly separable

Solving the Optimization Problem

- The linear discriminant function is:

$$g(\mathbf{x}) = \sum \lambda_i \boxed{\varphi(\mathbf{x}_\mathbf{i})^T \varphi(\mathbf{x})} y_i + b$$

Solving the Optimization Problem

- The linear discriminant function is: $$g(\mathbf{x}) = \sum \lambda_i \boxed{\varphi(\mathbf{x_i})^T \varphi(\mathbf{x})} y_i + b$$

- This is the same as saying that to classify a new point, we look at how **similar** it is to every other point in the training data, but use the new dot product $\varphi(x_i)^T \varphi(x)$

Solving the Optimization Problem

- The linear discriminant function is: $$g(\mathbf{x}) = \sum \lambda_i \boxed{\varphi(\mathbf{x_i})^T \varphi(\mathbf{x})} y_i + b$$

- This is the same as saying that to classify a new point, we look at how **similar** it is to every other point in the training data, but use the new dot product $\boldsymbol{\varphi}\ (\boldsymbol{x_i})\ ^T\boldsymbol{\varphi}\ (\boldsymbol{x})$

- No need to know this mapping $\boldsymbol{\varphi}$ explicitly, because we only use the new dot product of feature vectors in both the training and test.

Solving the Optimization Problem

- The linear discriminant function is:

$$g(\mathbf{x}) = \sum \lambda_i \boxed{\varphi(\mathbf{x_i})^T \varphi(\mathbf{x})} y_i + b$$

- This is the same as saying that to classify a new point, we look at how **similar** it is to every other point in the training data, but use the new dot product $\boldsymbol{\varphi}\ (\boldsymbol{x_i})\ ^T\boldsymbol{\varphi}\ (\boldsymbol{x})$

- No need to know this mapping $\boldsymbol{\varphi}$ explicitly, because we only use the new dot product of feature vectors in both the training and test.

- A *kernel function* is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x_i})^T \varphi(\mathbf{x_j})$$

Nonlinear SVMs: similarity, not distance!

Example of commonly used kernel functions:

- Linear $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (Radial Basis Function, or RBF) $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2s^2}\right)$
- Sigmoid $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(b_0 \mathbf{x}_i^T \mathbf{x}_j + b_1)$

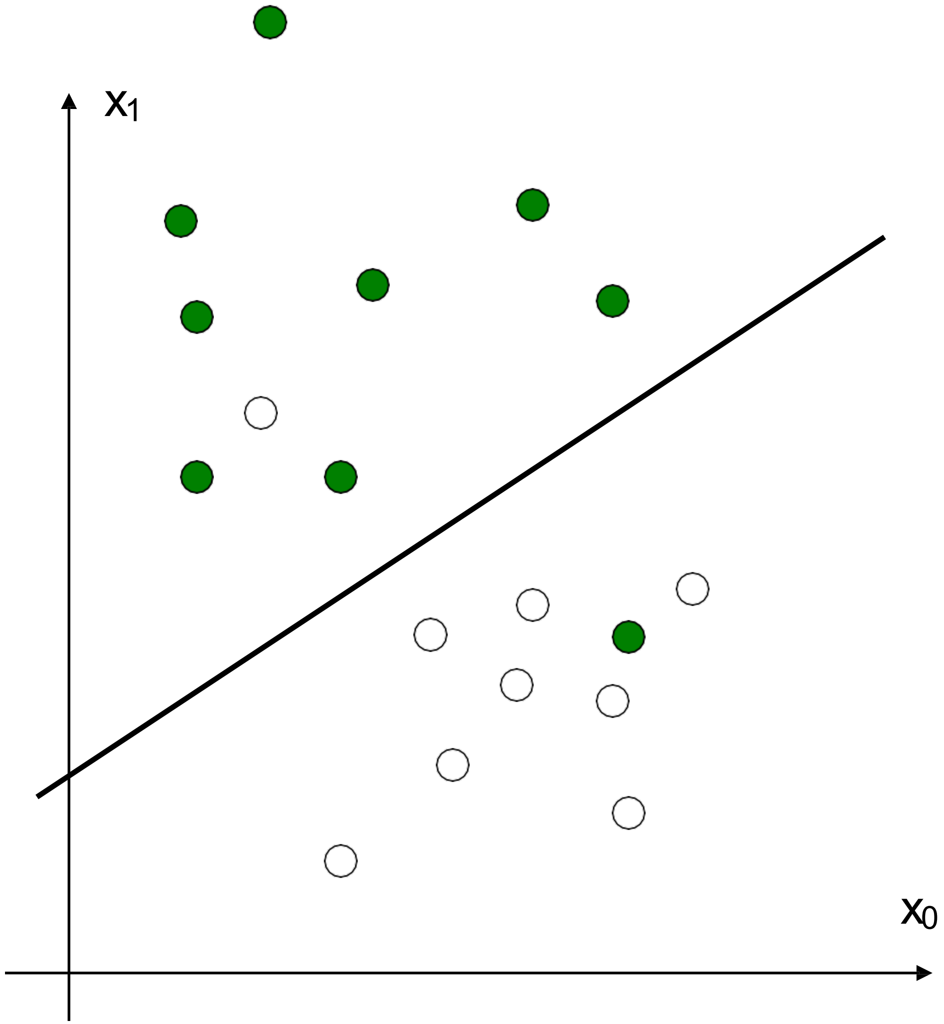
SVC()

- ***Kernel*** Specifies the kernel type to be used in the algorithm. It must be one of ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable. If none is given, ‘rbf’ will be used.
- ***Degree*** Degree of the polynomial kernel function (‘poly’). Ignored by all other kernels.
- ***Gamma*** Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’.
  - if gamma='scale' (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
  - if ‘auto’, uses 1 / n_features.
- ***Coef*** Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.

Hewlett Packard Enterprise Data Science Institute
UNIVERSITY OF HOUSTON

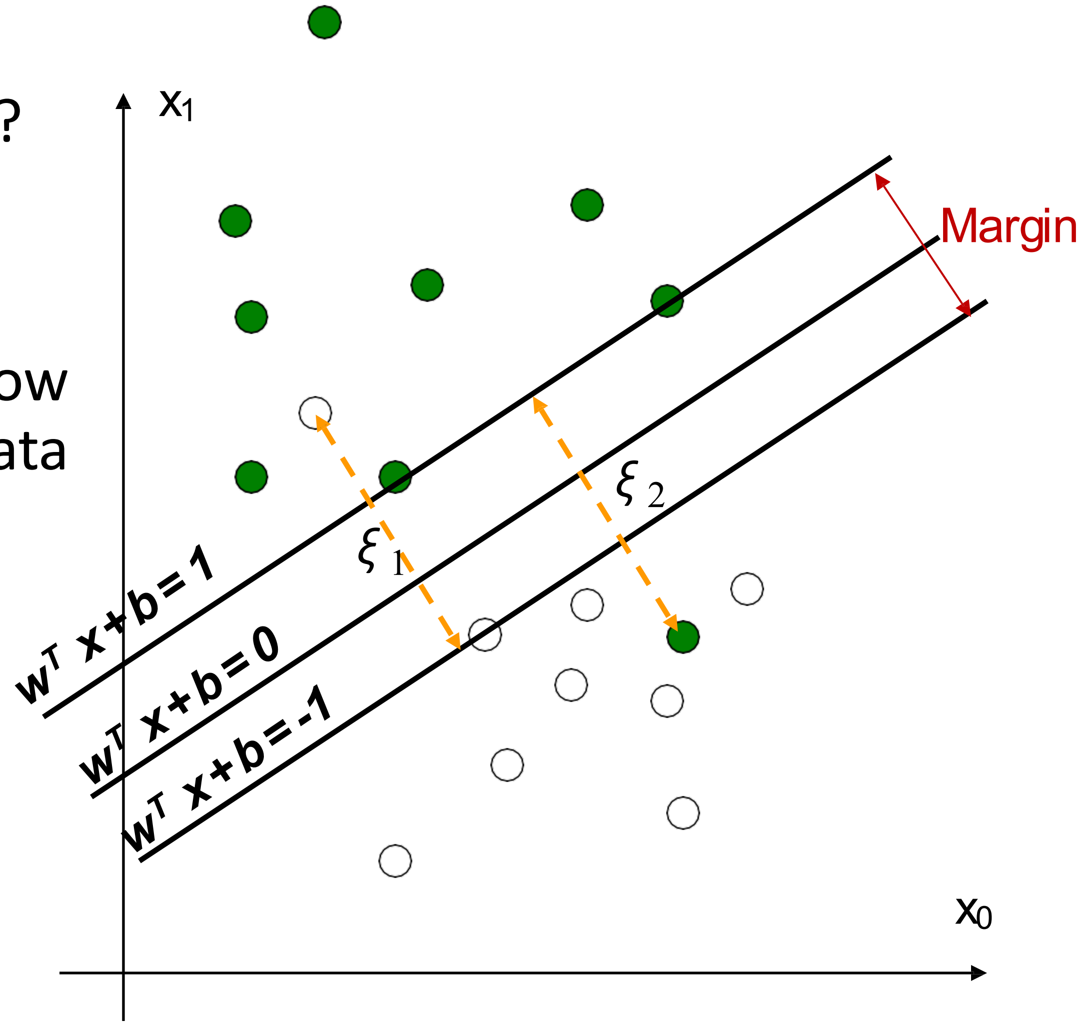
The regularization trick

- What if data is not "cleanly" separable? (noisy data, outliers, etc.)



- What if data is not “cleanly” separable? (noisy data, outliers, etc.)

- Slack variables ξ_i can be added to allow misclassification of difficult or noisy data points



Developed by Ioannis Konstantinidis

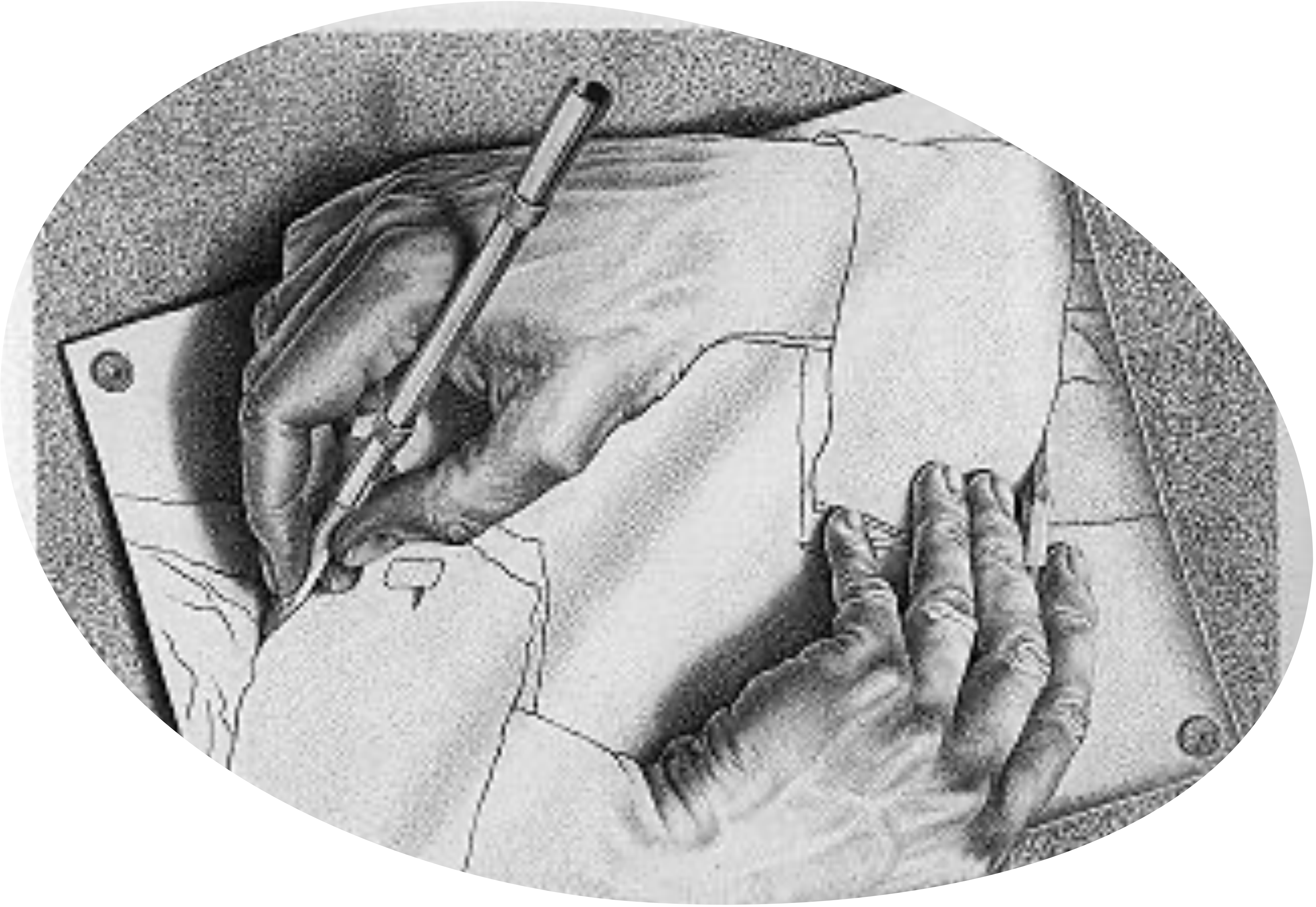
Optimization Problem: computing w

Quadratic programming with linear constraints

$$\text{minimize} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \xi_i$$

$$\text{s.t.} \quad y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \qquad \xi_i \geq 0$$

Parameter *C* can be viewed as a way to control over-fitting



Hands-on Example:

SVC

SVC()

C Regularization parameter.

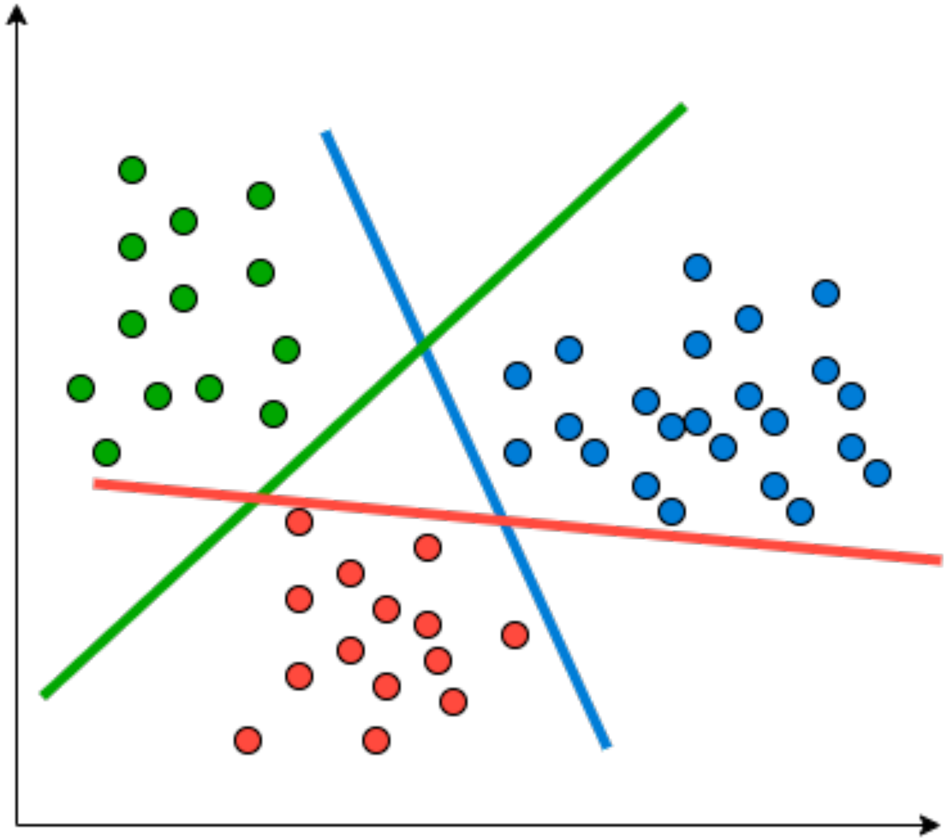
- The strength of the regularization is inversely proportional to C. Must be strictly positive.
- The penalty is a squared l2 penalty.

Developed by Ioannis Konstantinidis

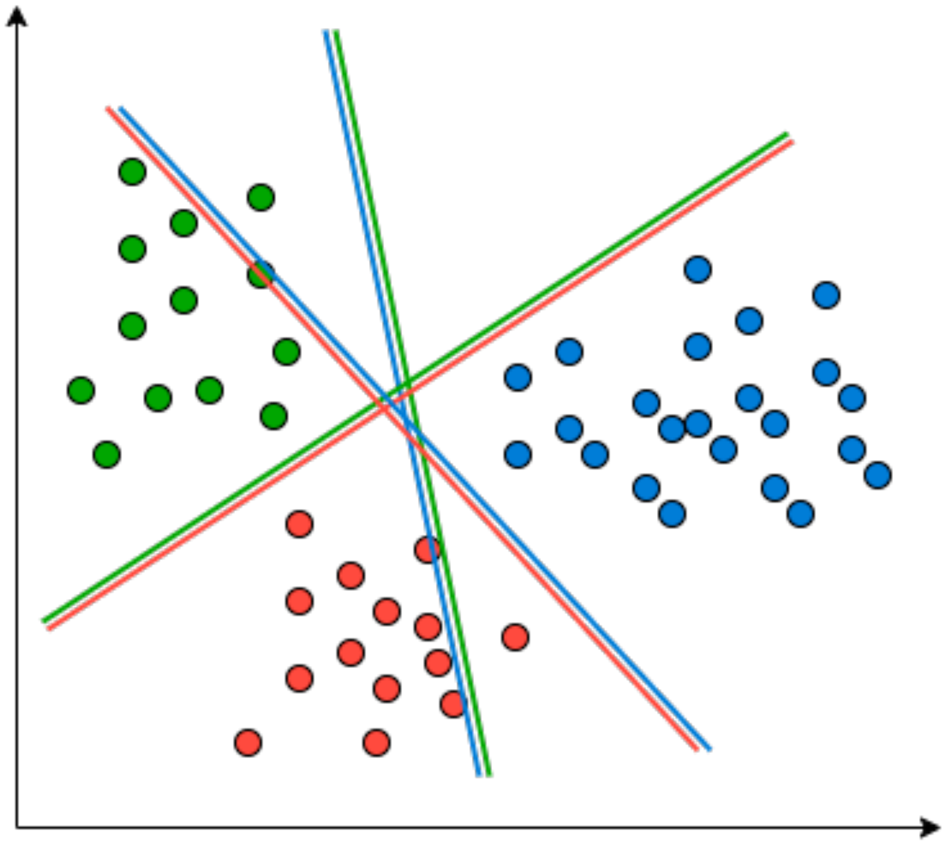
What if there are more than two classes?

Multiple Classes

Learn one discriminant function
for EVERY CLASS (one vs. rest/all, OvR/OvA)



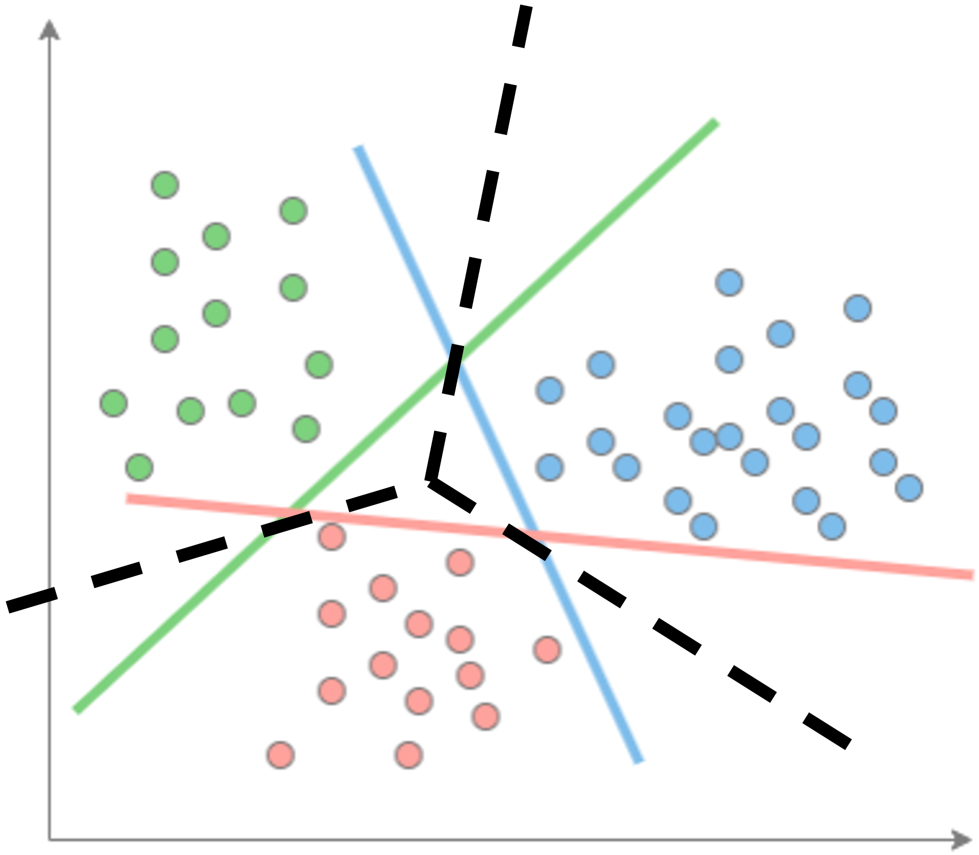
Learn a discriminant function
for EVERY PAIR of classes (one vs. one, or OvO)



Multiple Classes

linear discriminant functions:

$f_k(x) = \mathbf{w}^T_k\, x + \mathbf{w}_0 \qquad k = 1, \ldots, c$



For each point x,

pick the largest value $f_k(x)$