HPE DSI 311
Introduction to Machine Learning

Summer 2021

Instructor: Ioannis Konstantinidis

UNIVERSITY of HOUSTON
DIVISION OF RESEARCH
HEWLETT PACKARD ENTERPRISE DATA SCIENCE INSTITUTE

Overview

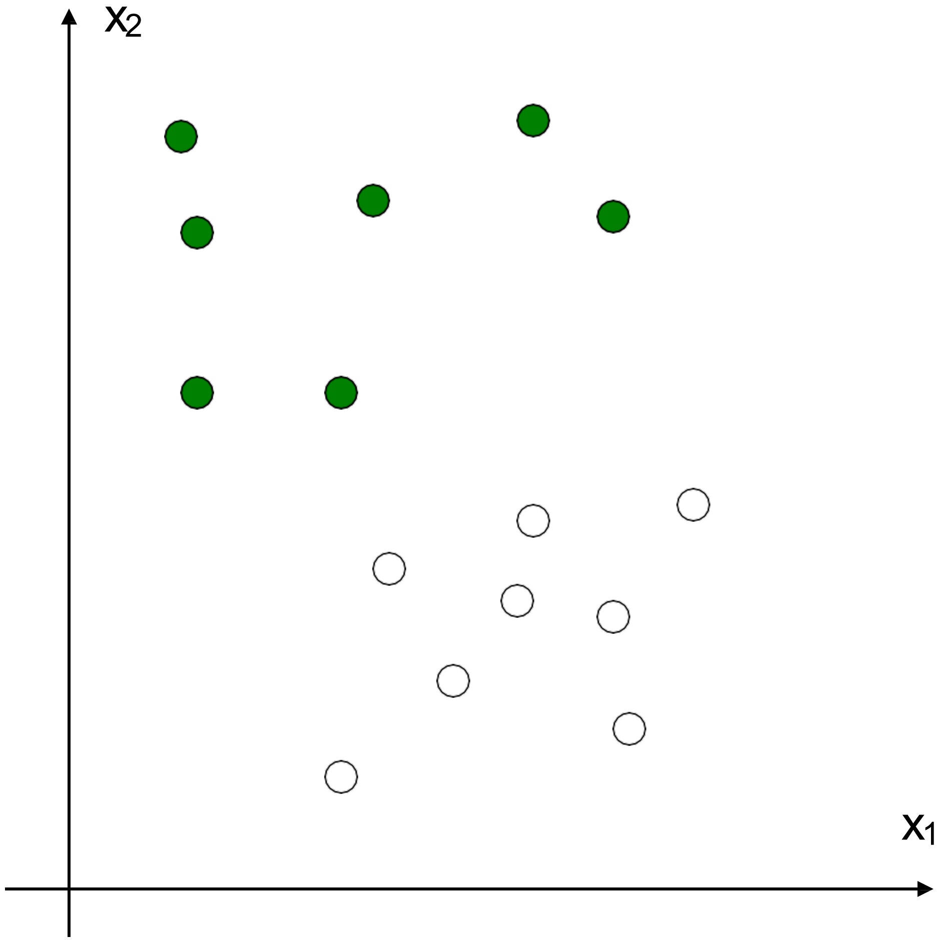
Support Vector Machines

Generalizations

- Kernels
- Regularization

New method:
Support Vector Machines

How would you classify these points to minimize error?

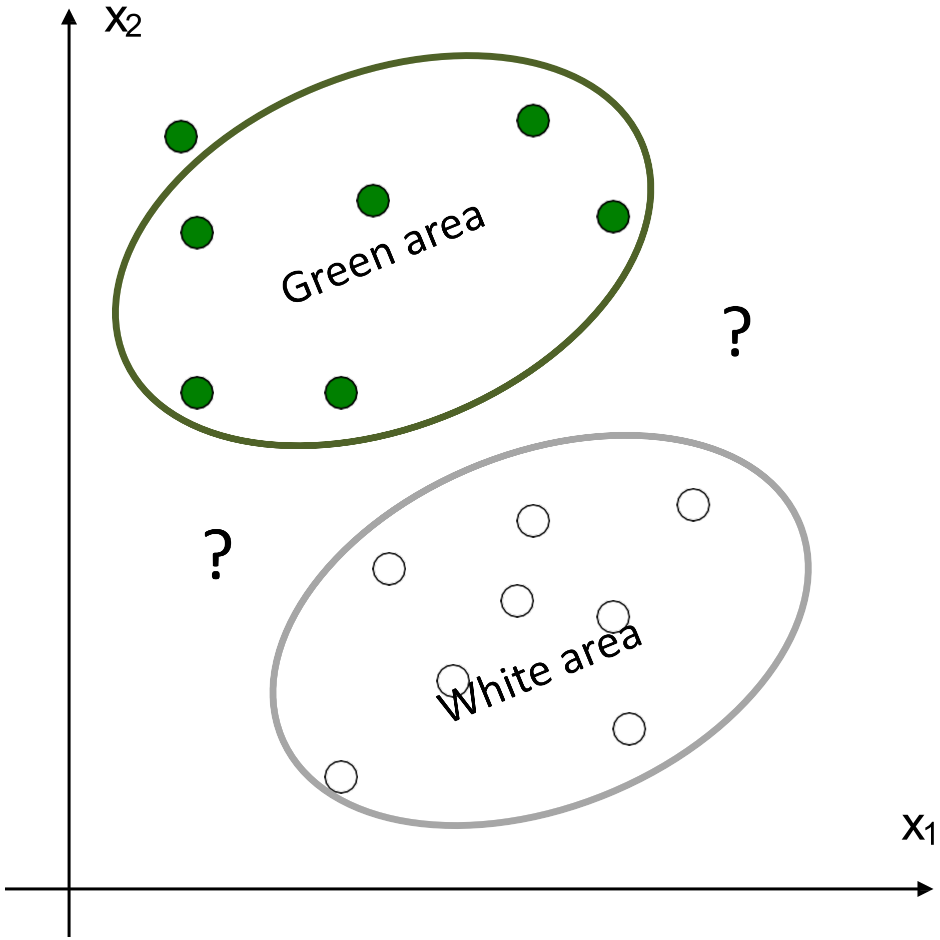


K-NN

How would you classify these points to minimize error?

- Match to the color of the nearest neighbors
- Computation based on only k training points, but they differ based on where the test point is located

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

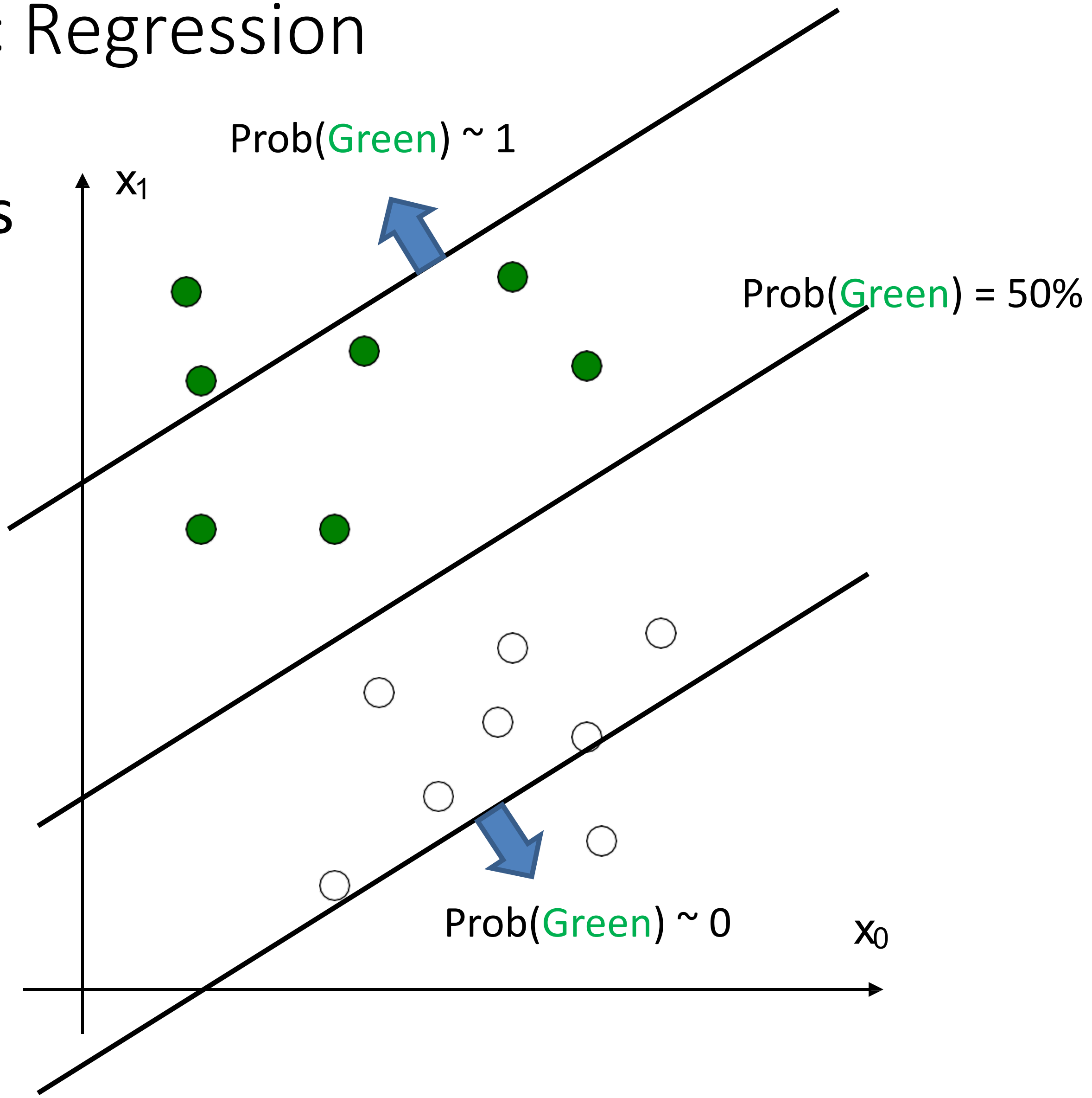


Logistic Regression

How would you classify these points to minimize error?

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

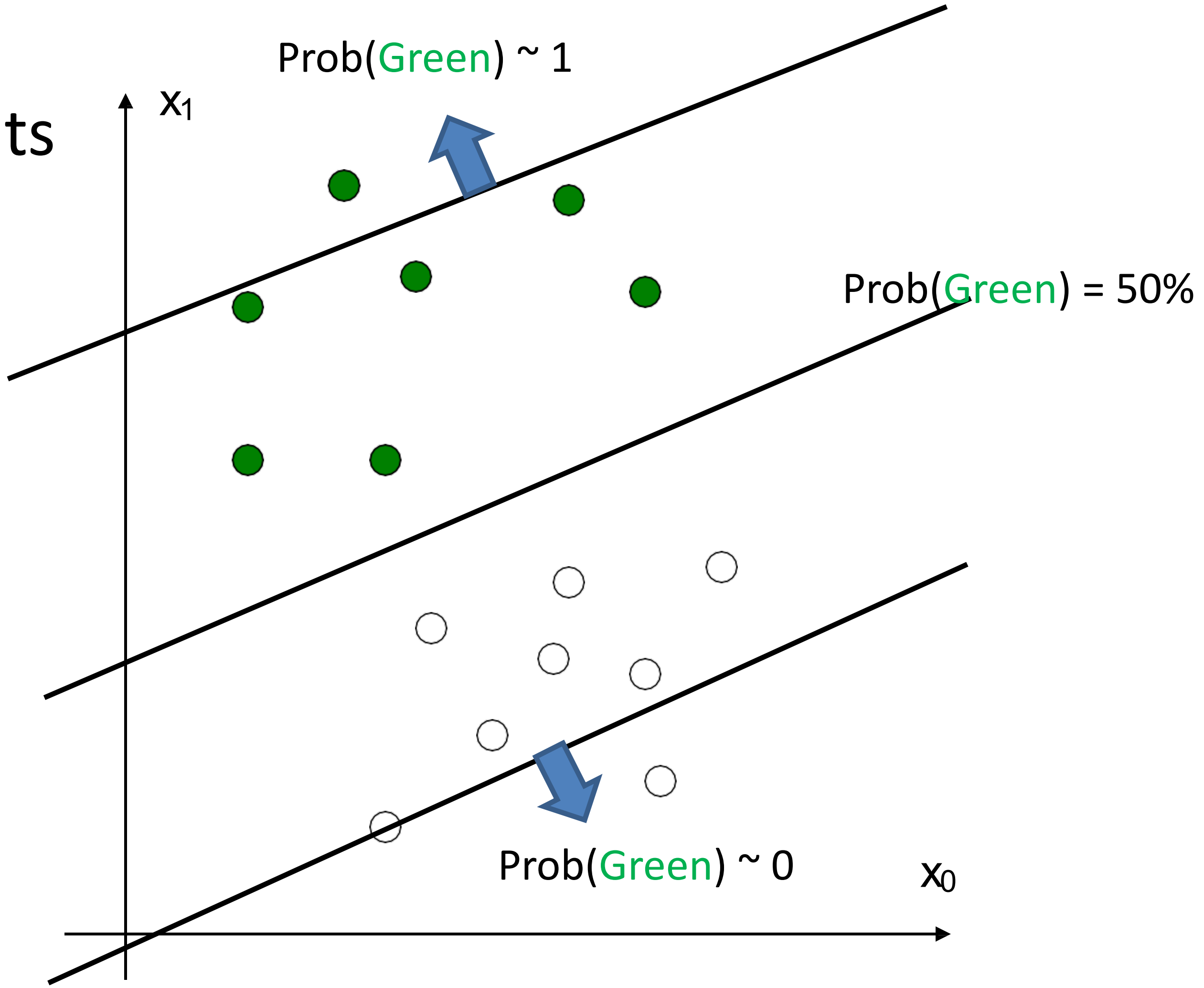
$$Prob(Green) \sim w_0 x_0 + w_1 x_1 + b = \mathbf{w}^T \mathbf{x} + b$$



Logistic Regression

How would you classify these points to minimize error?

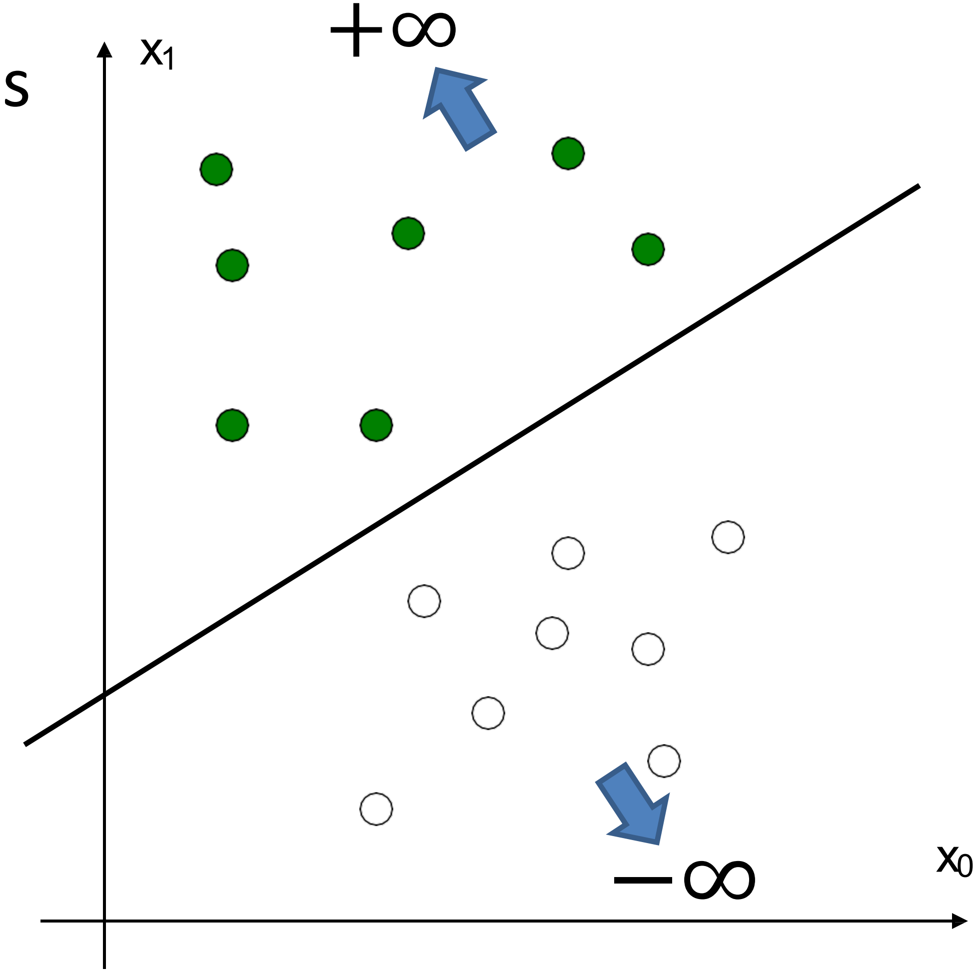
- Compute probability of match
- Computation based on ALL training points
- Distant points (outliers?) affect decision boundary



Linear Classifiers

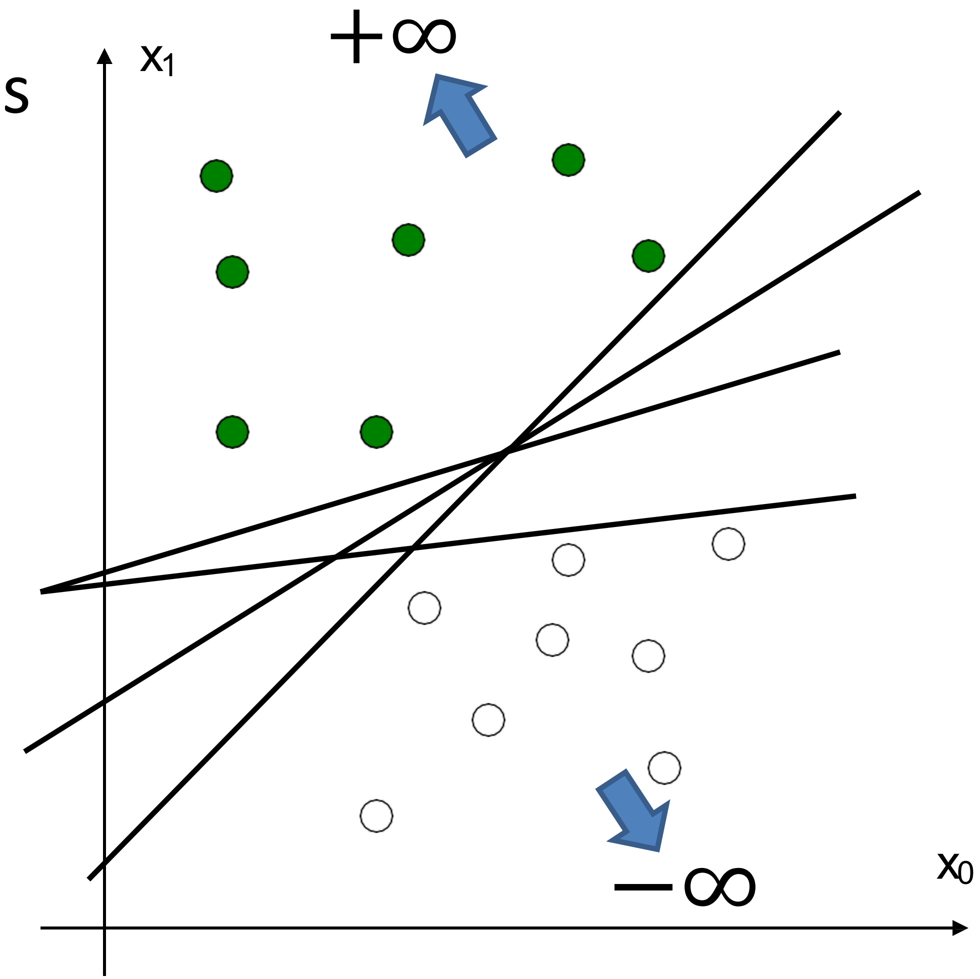
How would you classify these points to minimize error?

- Use a linear function as boundary (signed distance)



Linear Classifiers

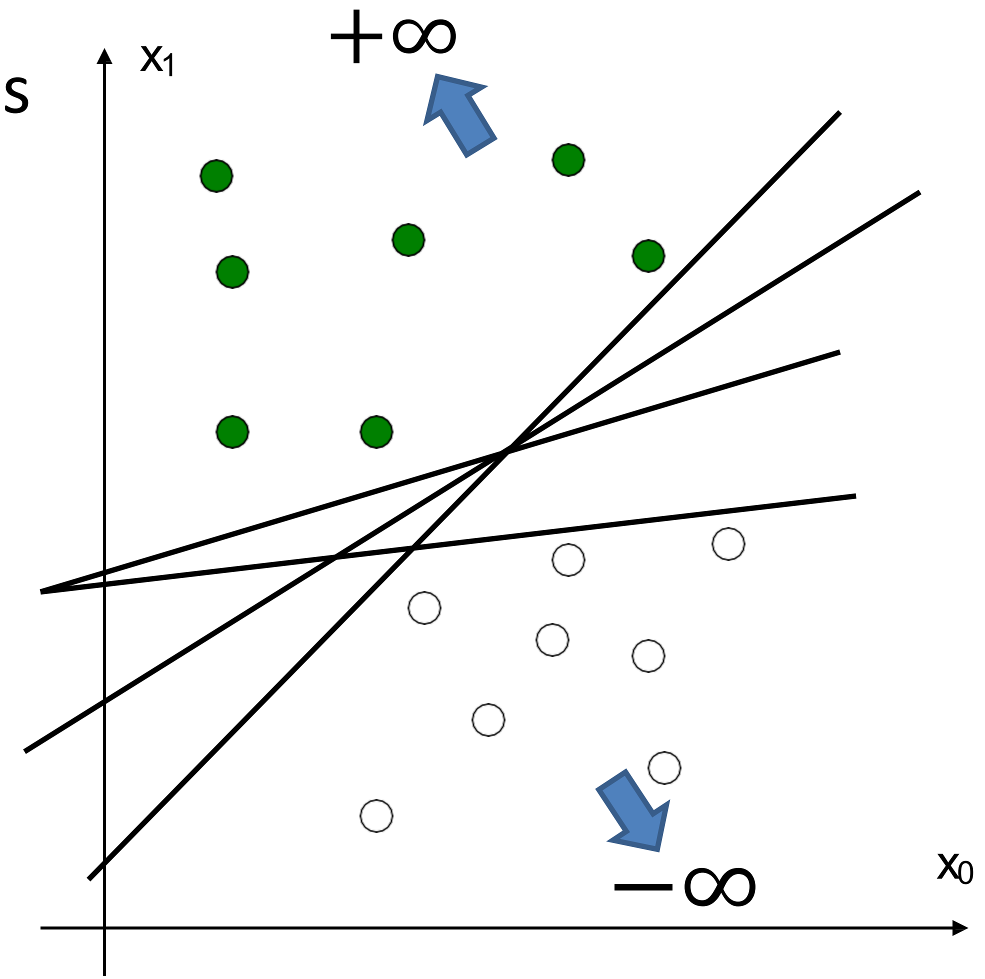
How would you classify these points to minimize error?



- Use a linear function as boundary (signed distance)
- MANY choices! (infinitely many, tbh)

Linear Classifiers

How would you classify these points to minimize error?

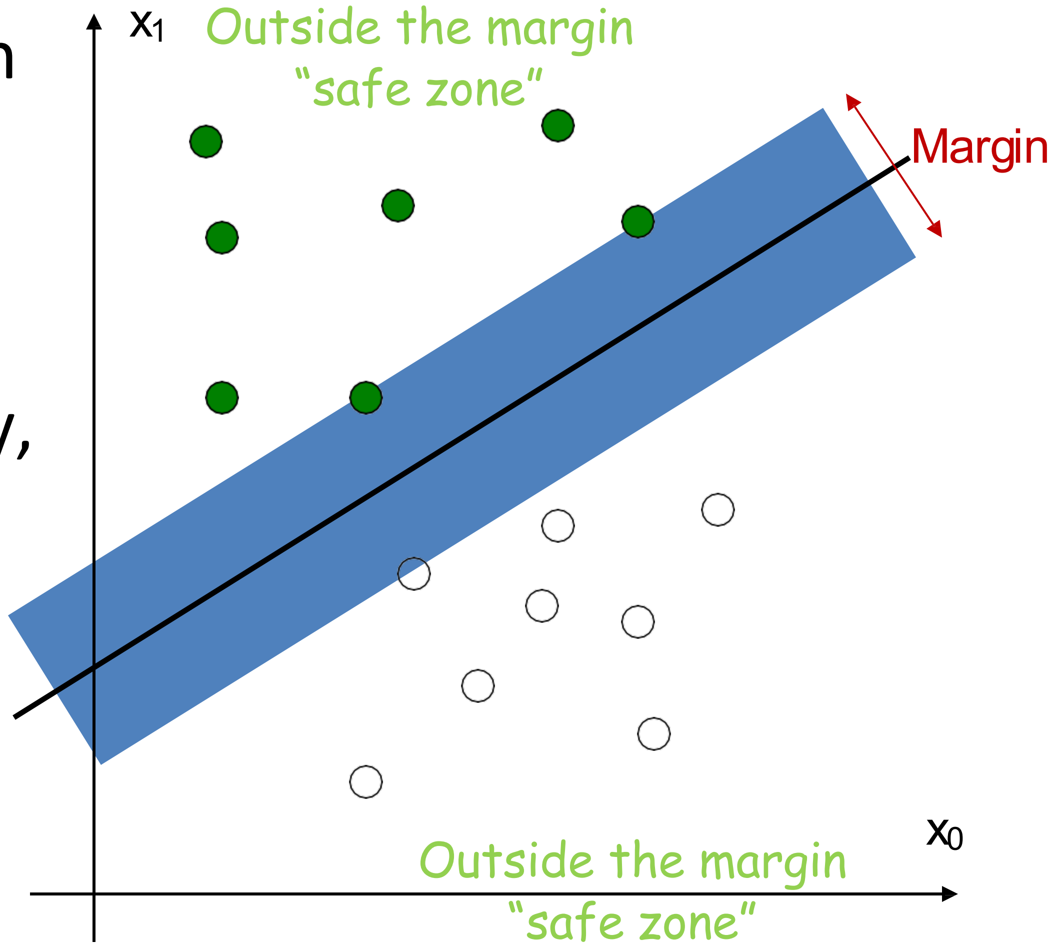


- Use a linear function as boundary (signed distance)
- MANY choices! (infinitely many, tbh)
- How to pick one?

Largest Margin Linear Classifier

Pick the linear discriminant function with the maximum margin

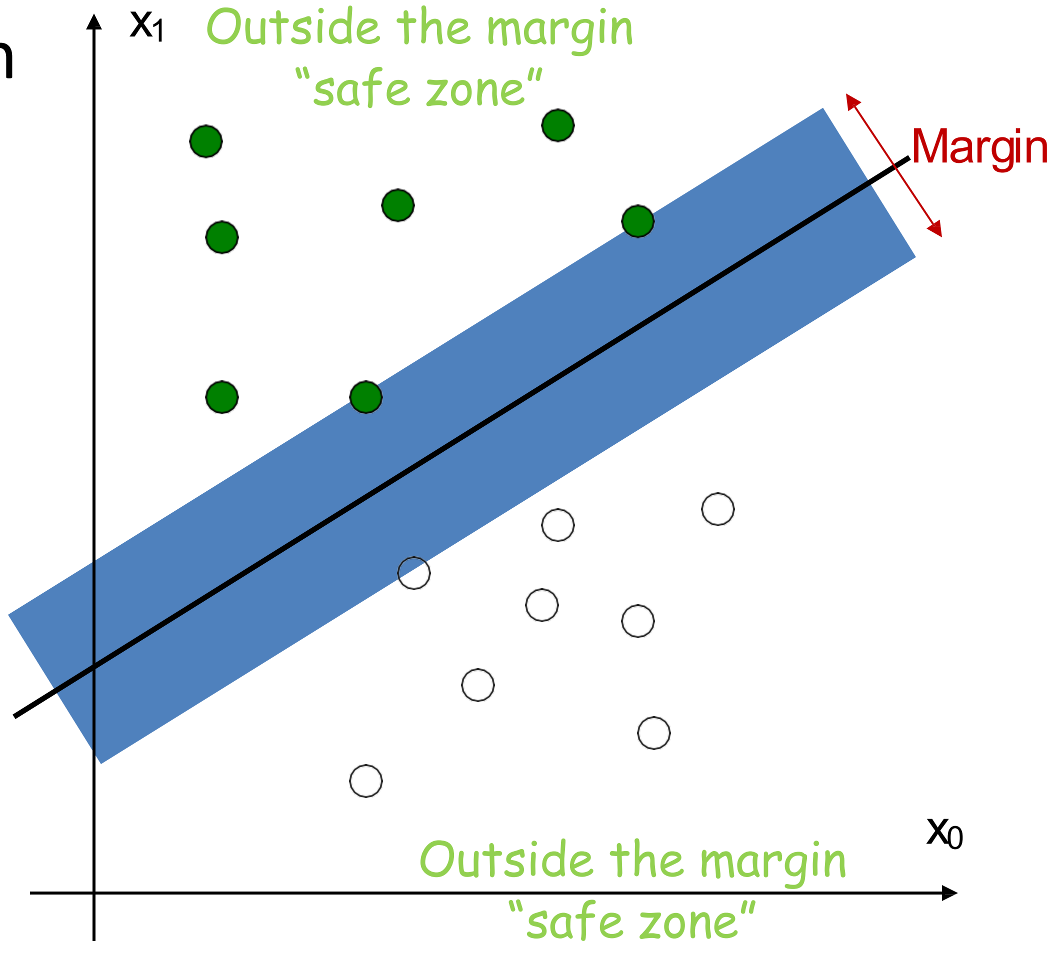
Margin is defined as the width that the boundary could be increased by, before hitting a data point



Largest Margin Linear Classifier

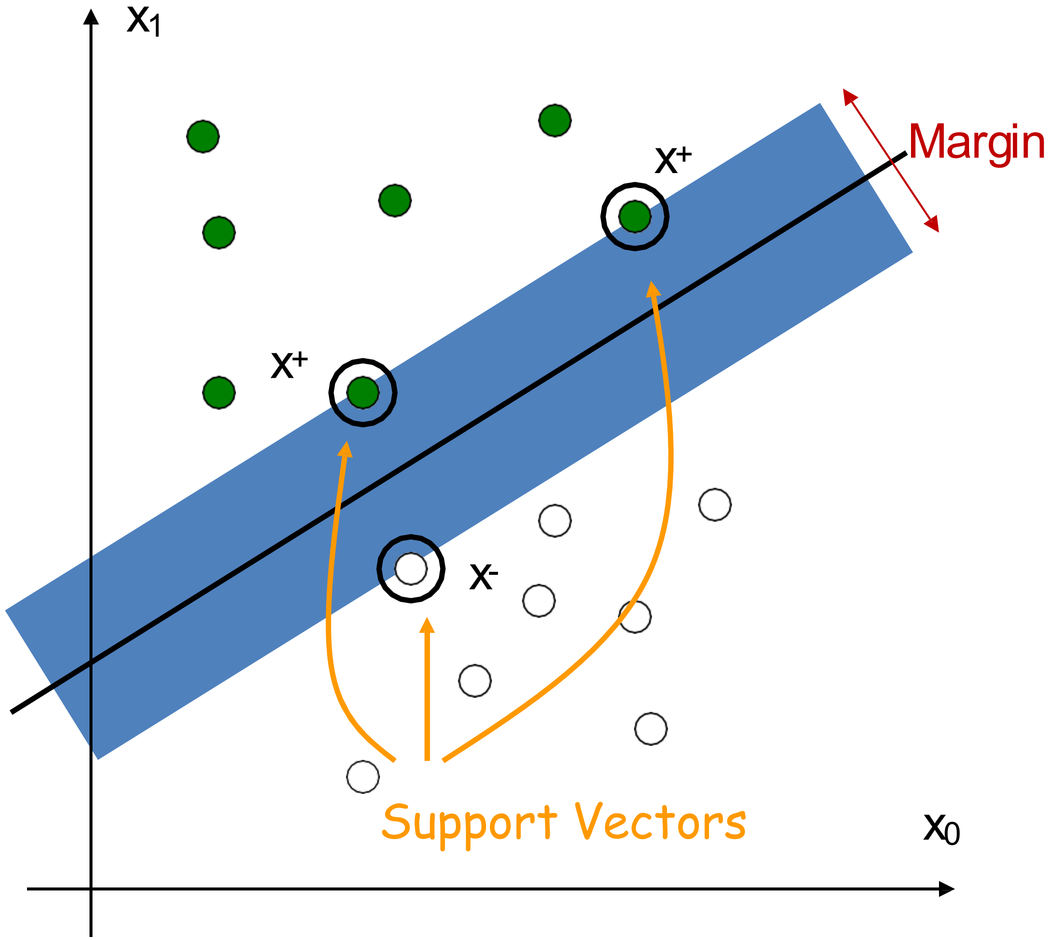
Pick the linear discriminant function with the maximum margin

- 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



Largest Margin Linear Classifier

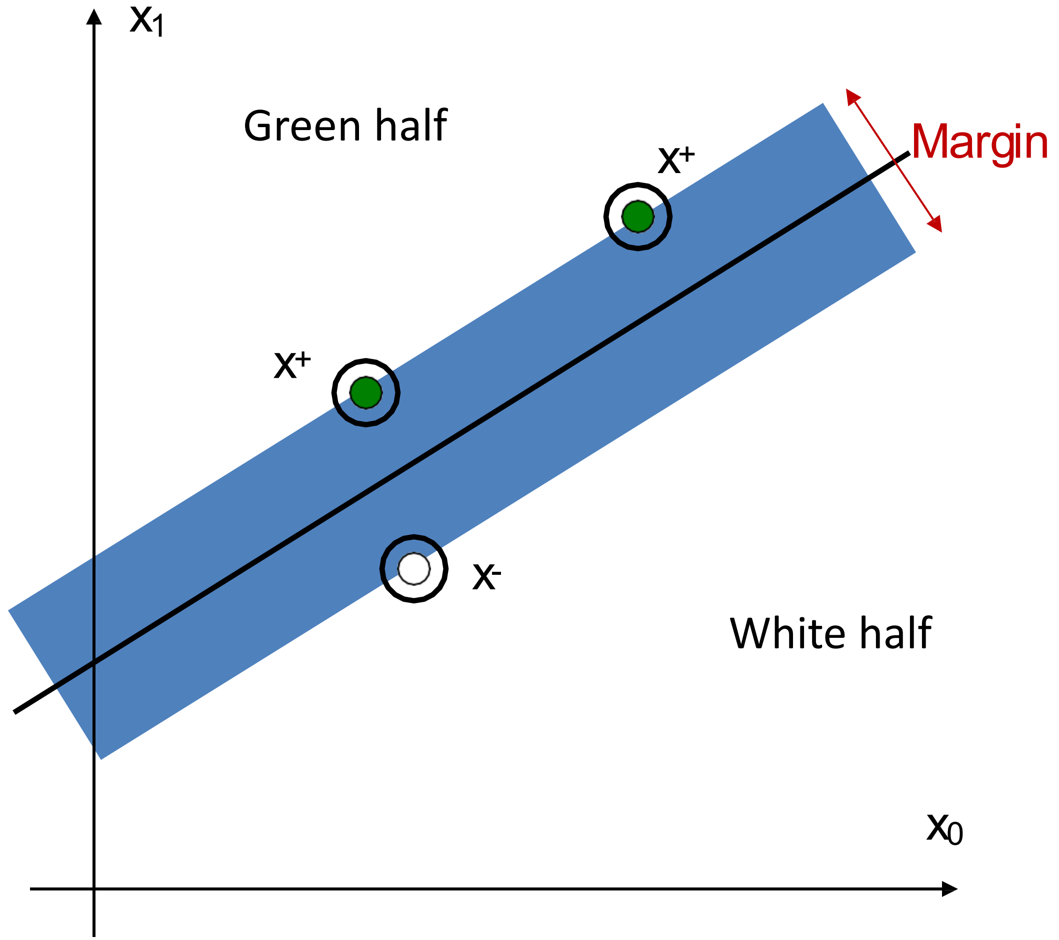
These data points that define the margin are called support vectors



Largest Margin Linear Classifier

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



Largest Margin Linear Classifier

Points on the decision boundary:

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

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

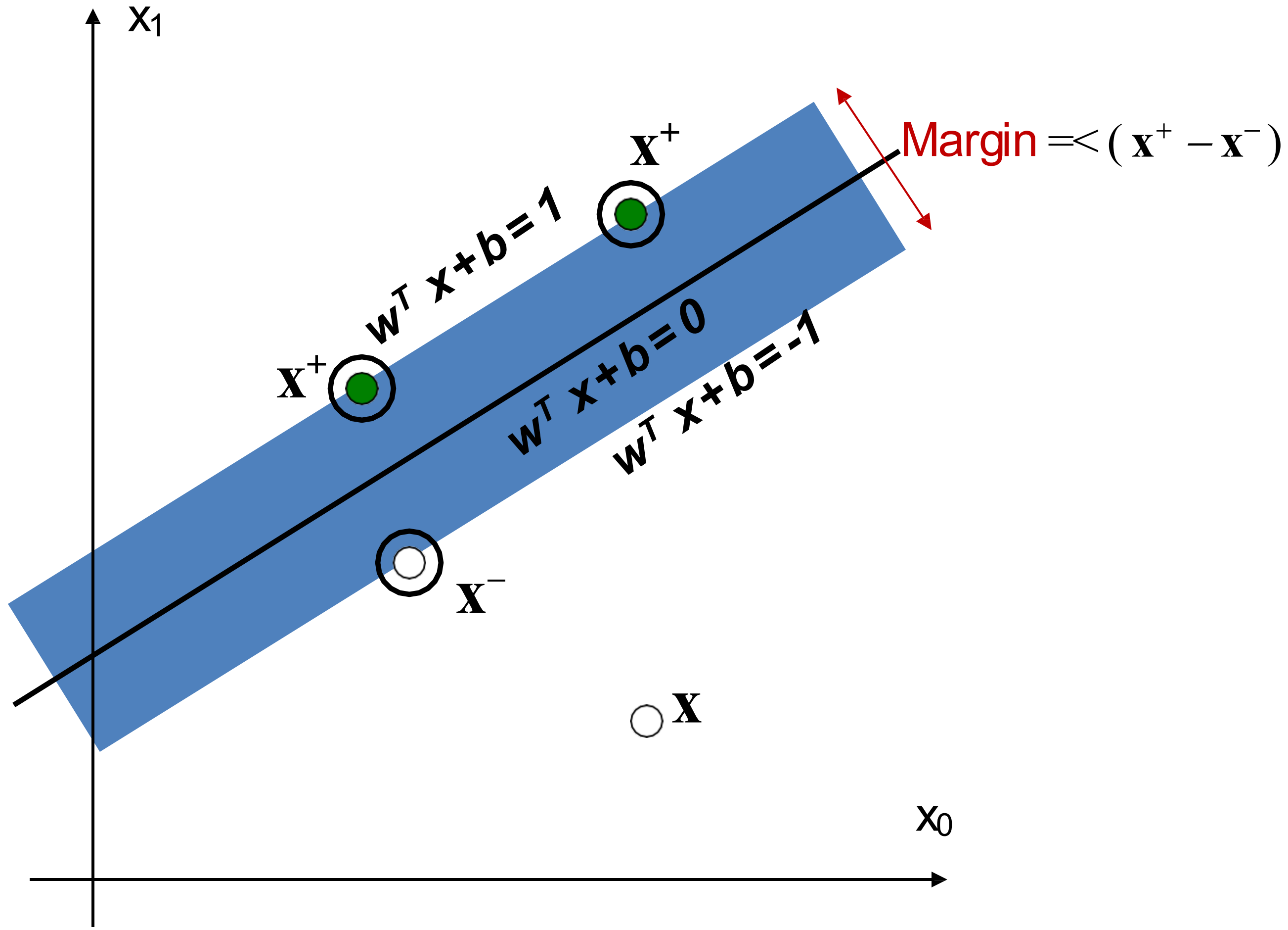
$$\mathbf{w}^T\mathbf{x}^+ + b = 1$$

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

Points elsewhere:

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

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



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:

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

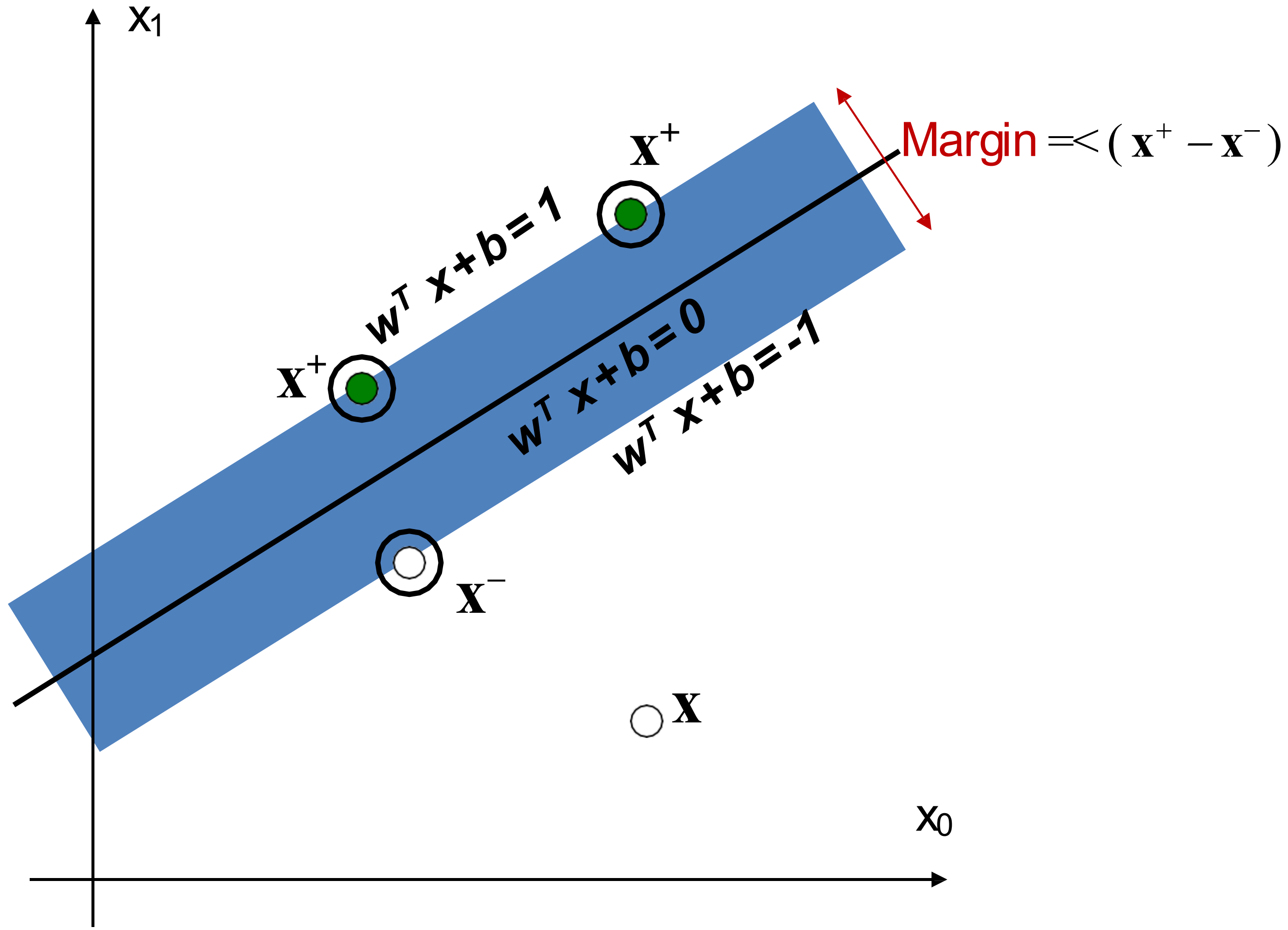
Largest Margin Linear Classifier

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

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

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

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



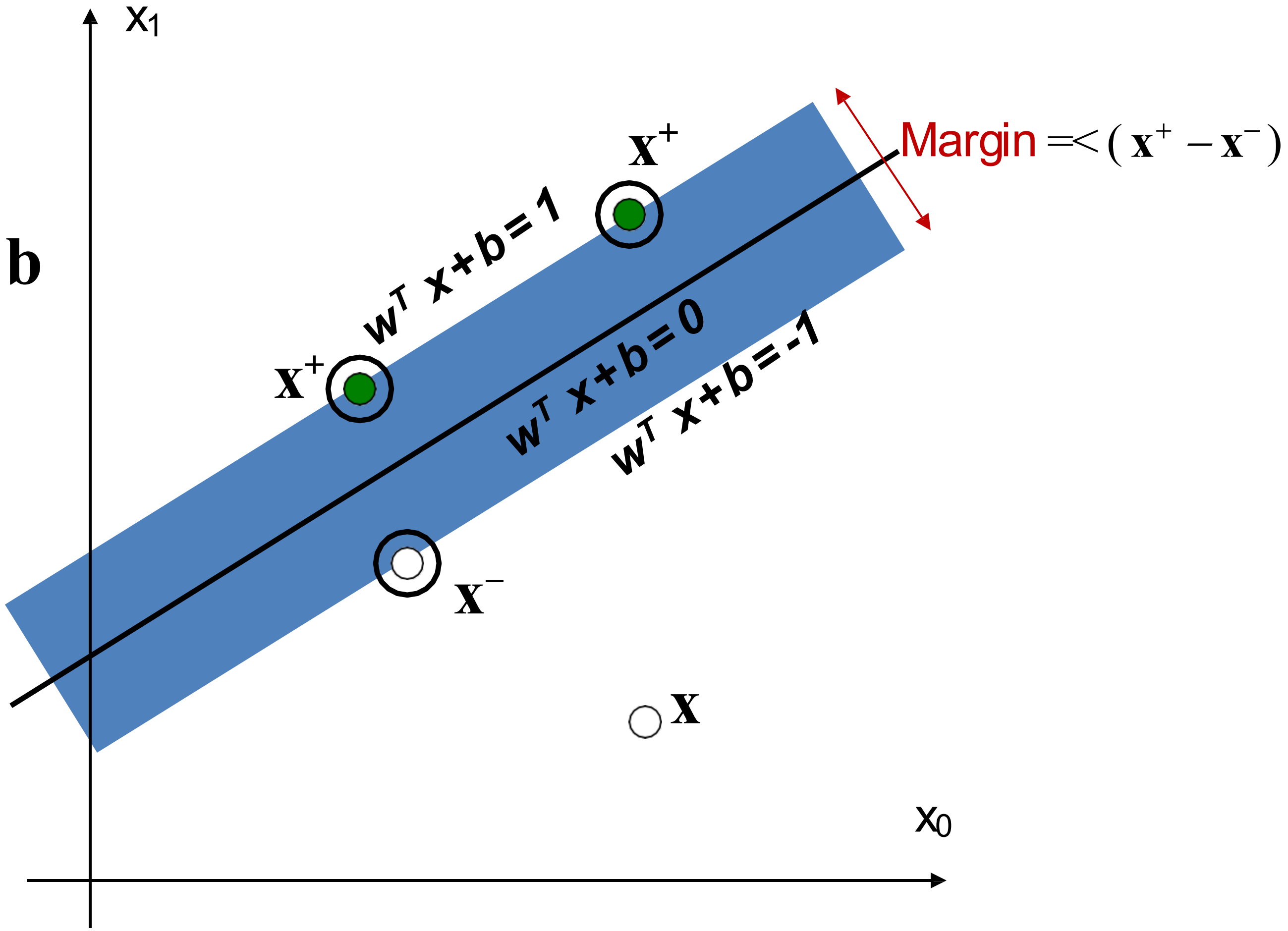
Largest Margin Linear Classifier

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 = Weighted Neighbors Nearest to Margin

<u>For training</u>

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

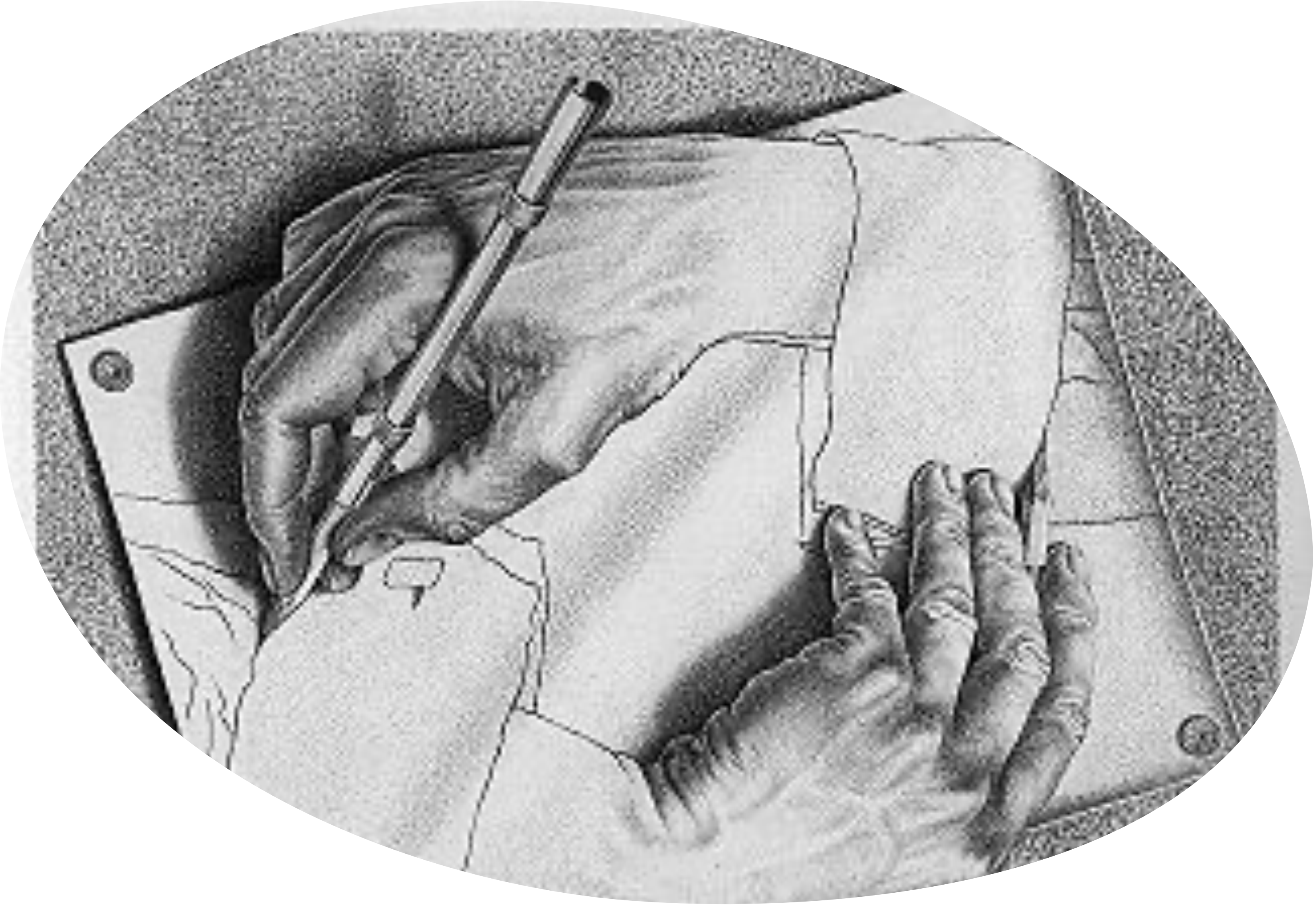
Data points on the margin boundary are important:

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

<u>For testing</u>

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

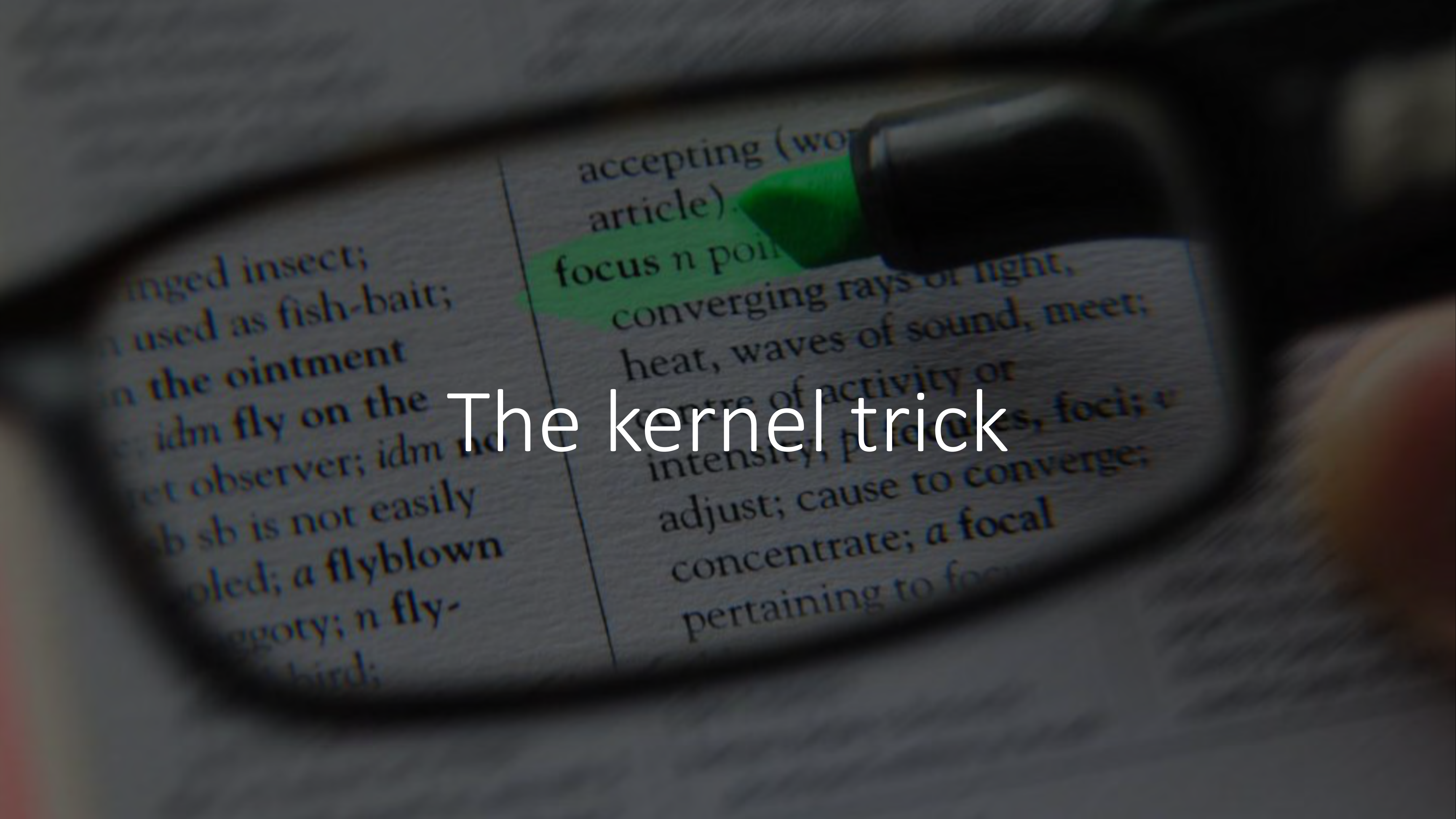


Hands-on Example:

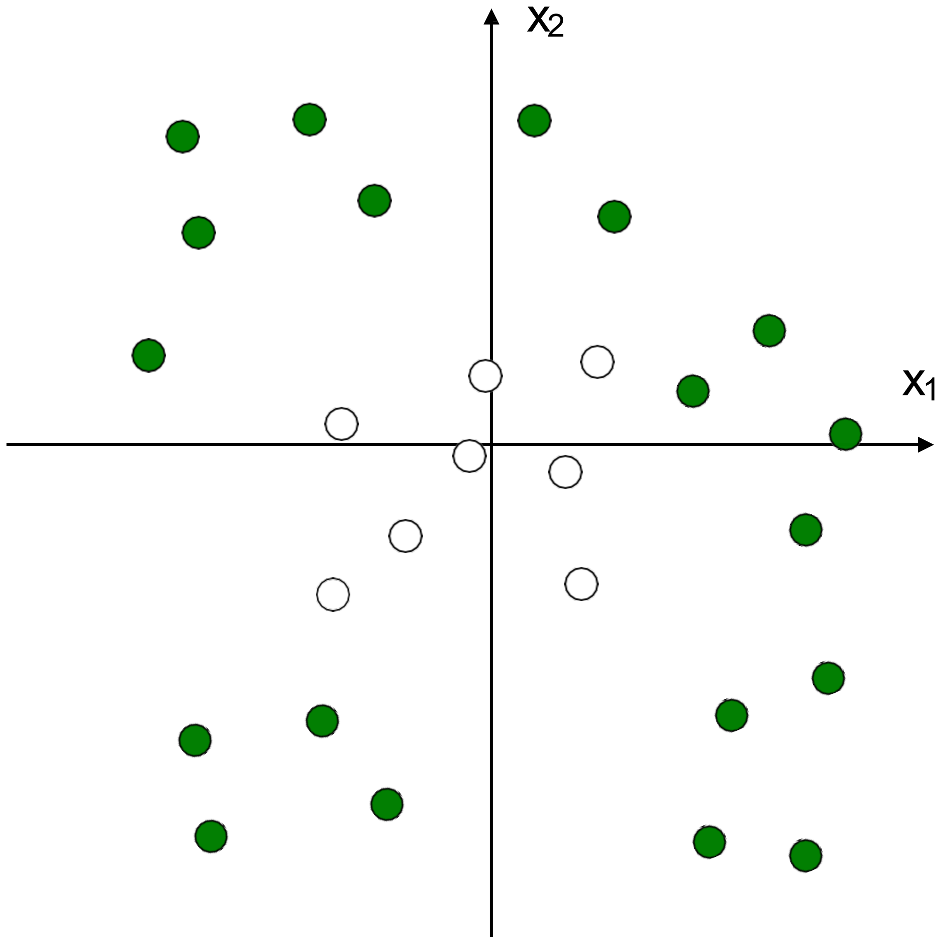
SVC

What if the points don’t line up exactly?

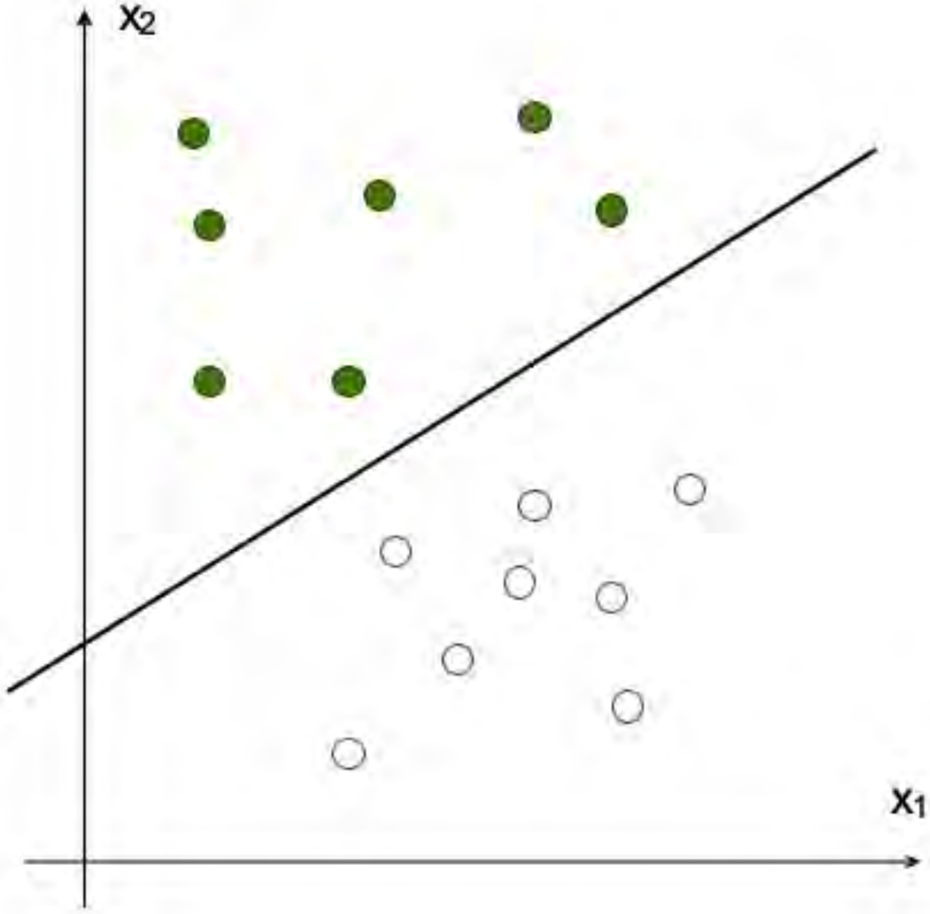
The kernel trick



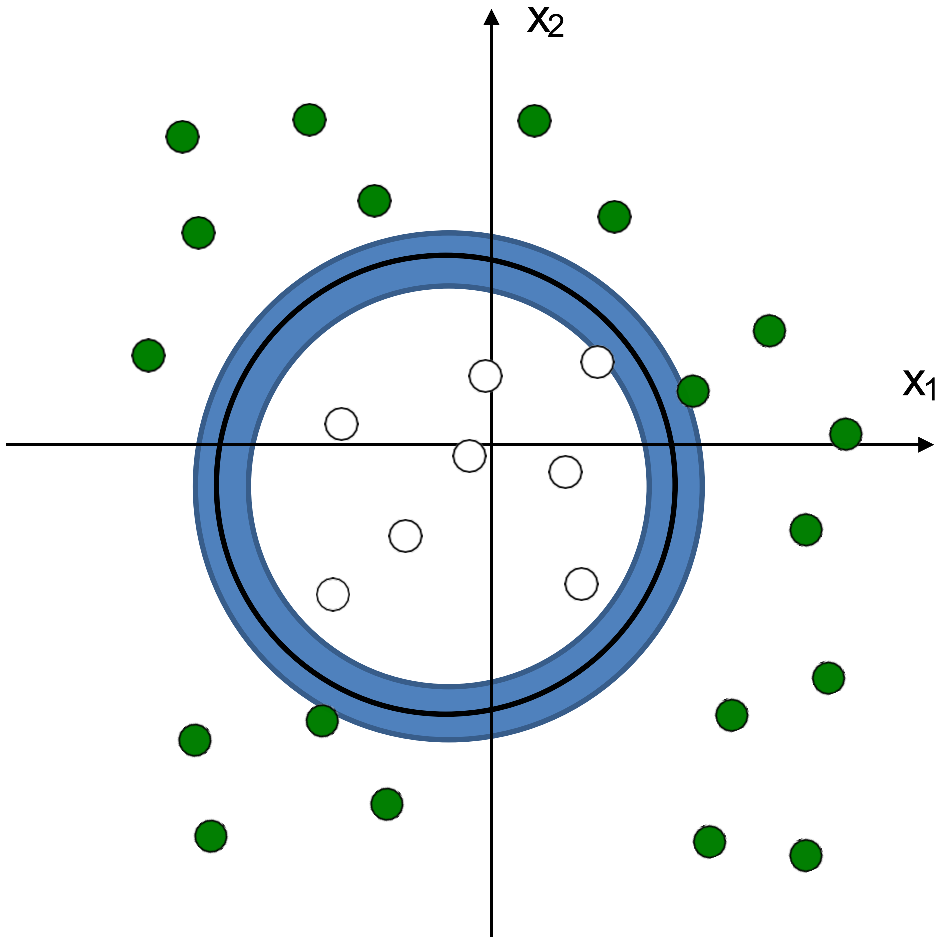
Non-linearity



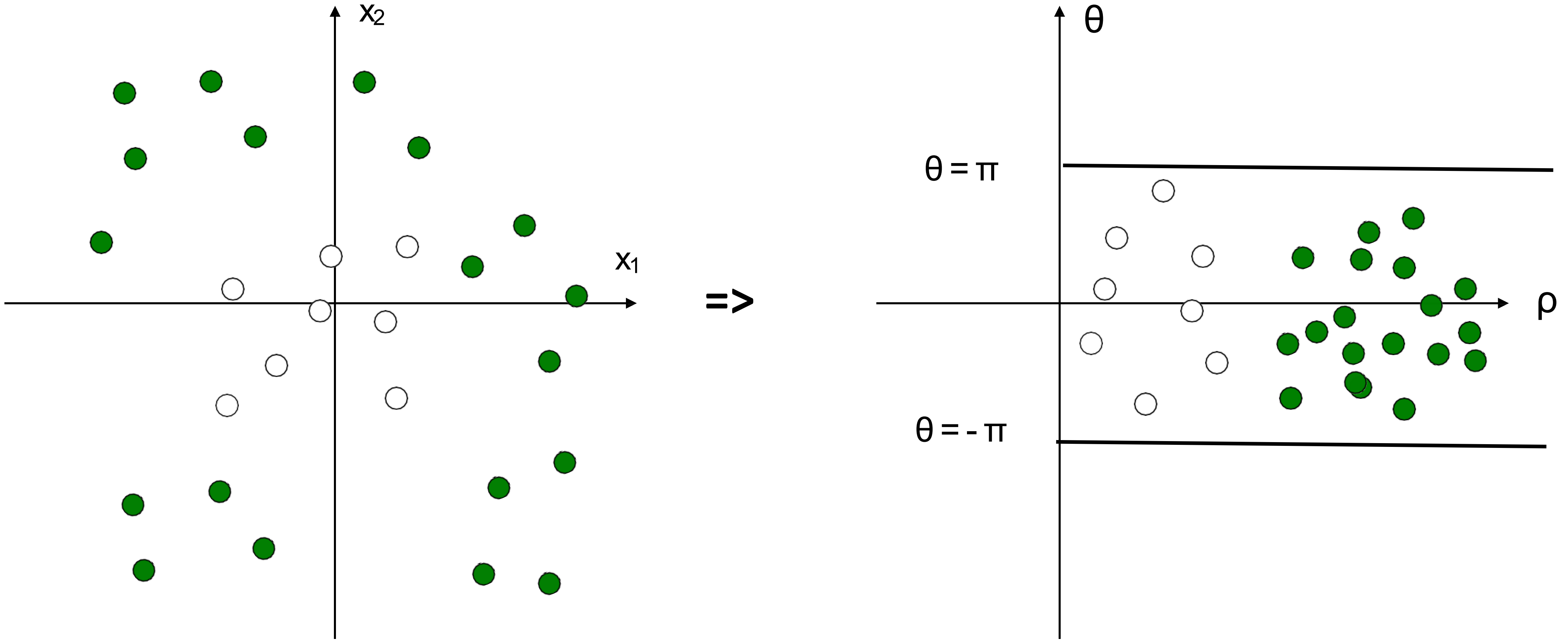
vs.



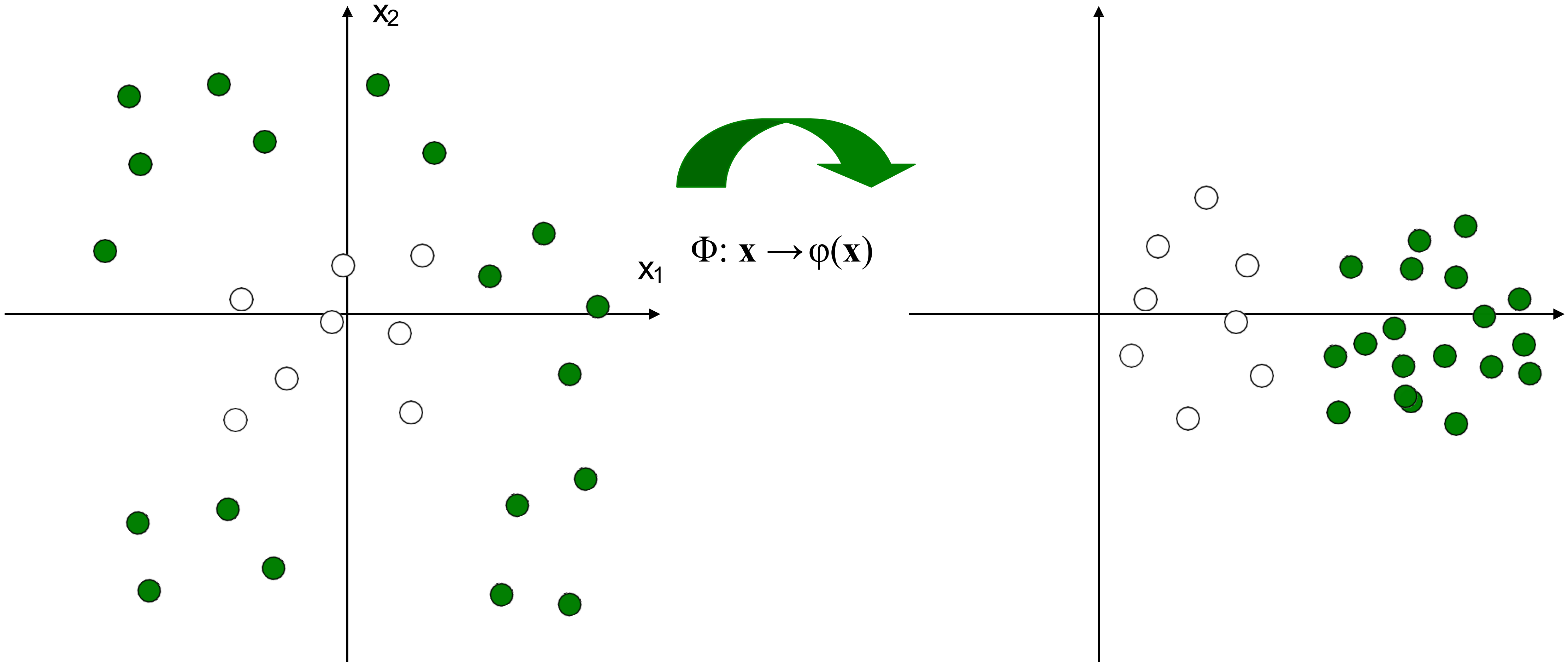
Circle Machines / Discriminant Analysis?



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

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 $\boldsymbol{\varphi(x_i)^T \varphi(x)}$

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 $\boldsymbol{\varphi}(x_i)^T \boldsymbol{\varphi}(x)$

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:

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

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) $$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2s^2}\right)$$

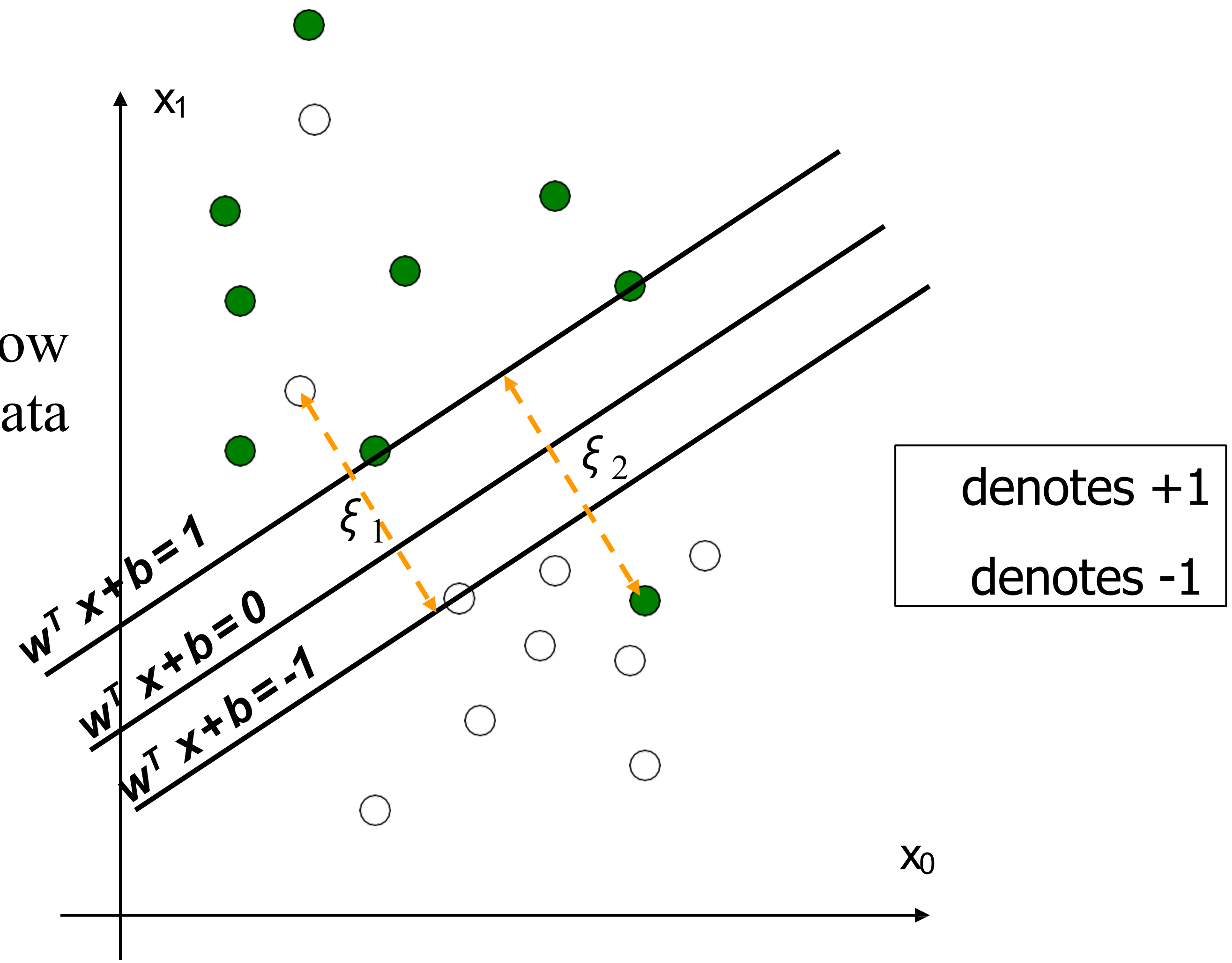
Sigmoid $$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(b_0 \mathbf{x}_i^T \mathbf{x}_j + b_1)$$

The regularization trick

Largest Margin Linear Classifier

What if data is not linear separable?
(noisy data, outliers, etc.)

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



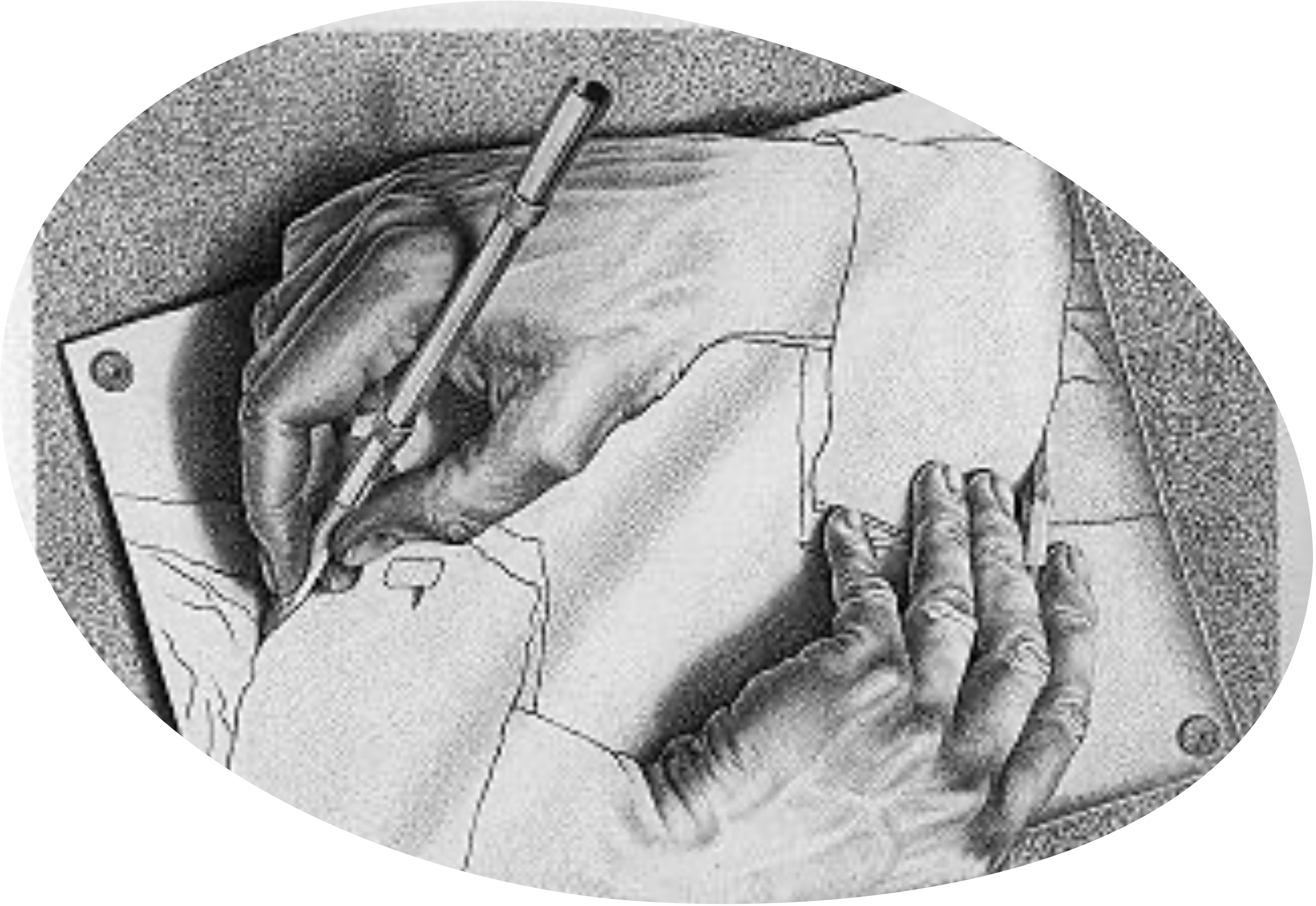
Optimization Problem: computing **w**

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.

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’.

SVC()

Probability Whether to enable probability estimates. This must be enabled prior to calling fit, will slow down that method as it internally uses 5-fold cross-validation

class_weight
max_iter
random_state

What if there are more than two classes?

The Problem with Multiple Classes

How do we use a linear discriminant when we have more than two classes?

There are several approaches:

1. Learn one discriminant function for each class (ovr)
2. Learn a discriminant function for all pairs of classes (ovo)

If c is the number of classes, in the first case we have c functions and in the second case we have c(c-1) / 2 functions.

In both cases we are left with ambiguous regions.

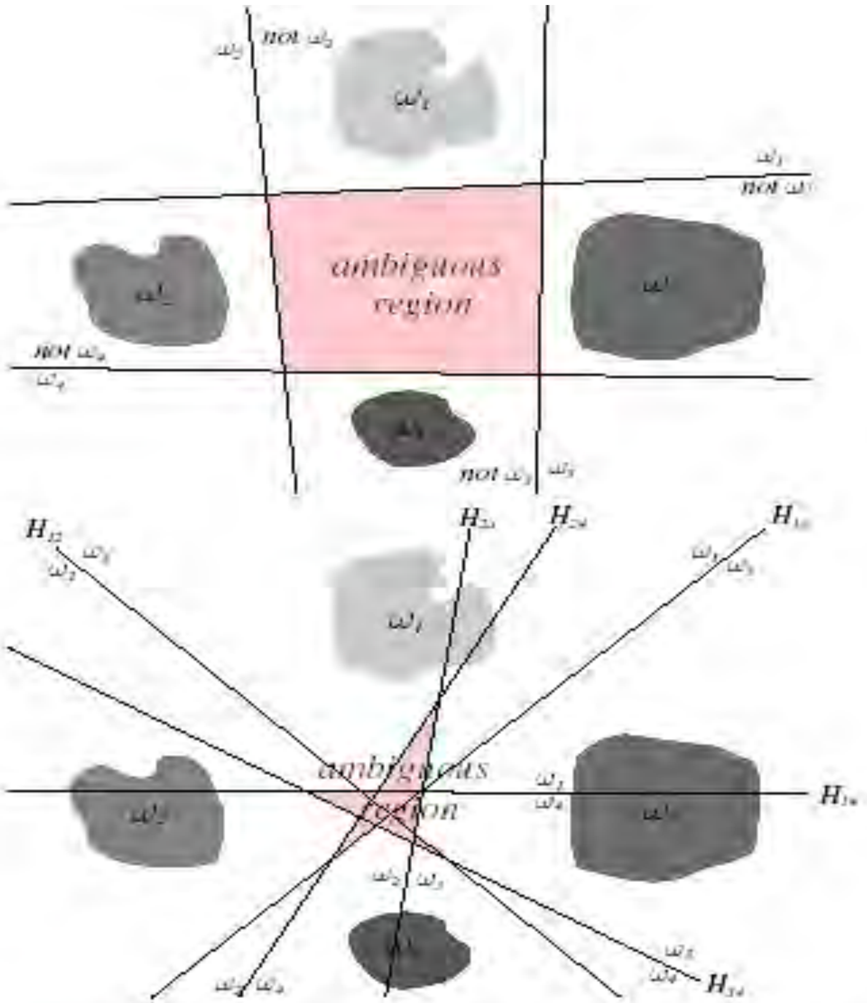


FIGURE 5.3. Linear decision boundaries for a four-class problem. The top figure shows ω_i/not ω_i dichotomies while the bottom figure shows ω_i/ω_j dichotomies and the corresponding decision boundaries H_{ij}. The pink regions have ambiguous category assignments. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Linear Machines

To avoid the problem of ambiguous regions we can use linear machines:

- We define c linear discriminant functions:

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

- For each point x we pick the largest value $g_k(x)$

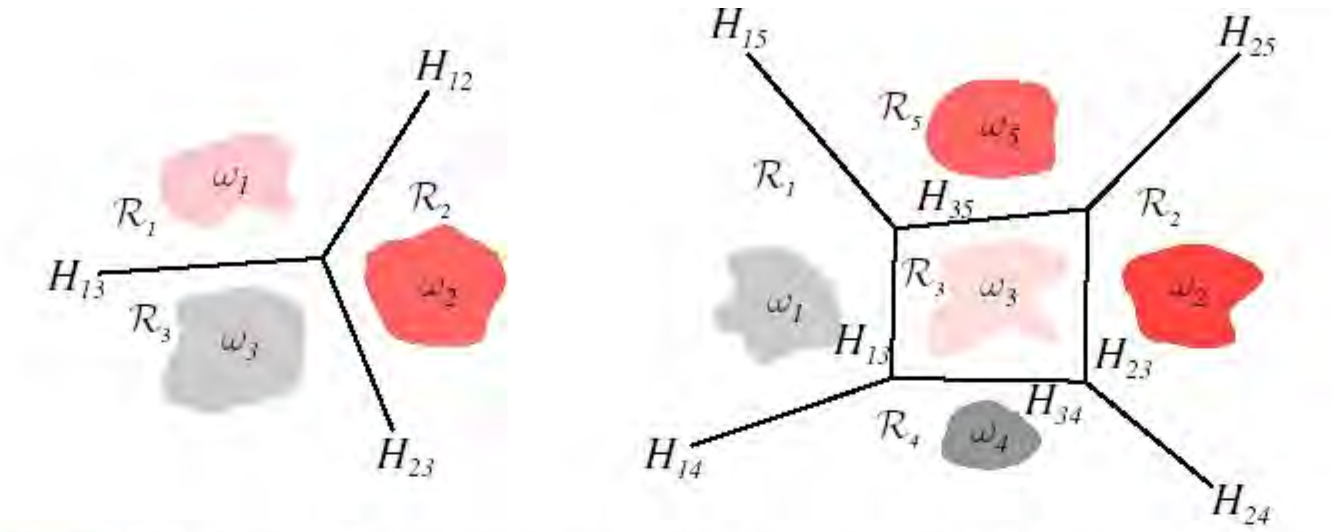


FIGURE 5.4. Decision boundaries produced by a linear machine for a three-class problem and a five-class problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.