

CSE 445

Lecture 9

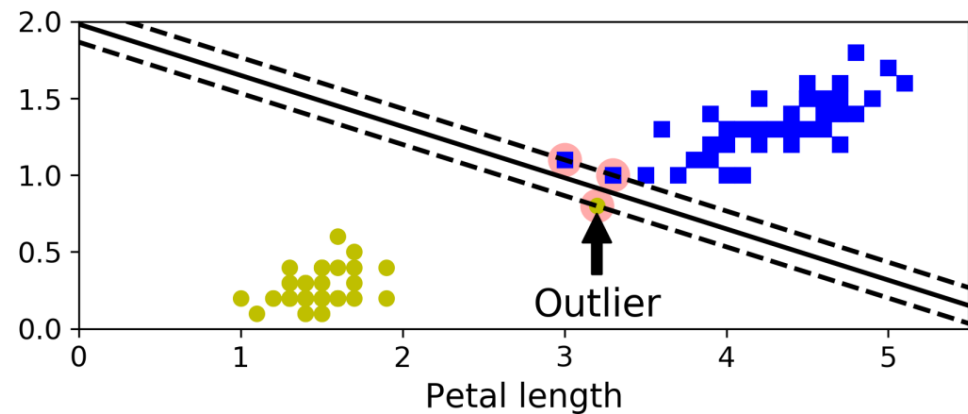
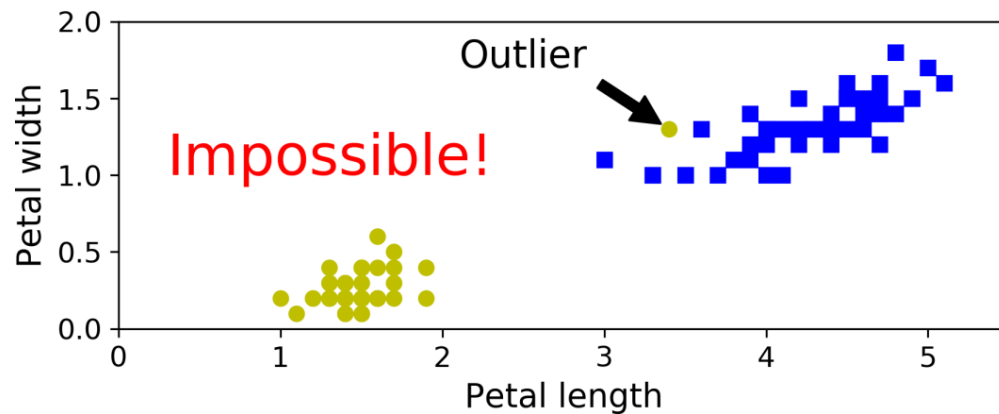
Support Vector Machine

Support Vector Machine

- A Support Vector Machine (SVM) is a very powerful and versatile Statistical Pattern Recognition model
- It can perform linear or nonlinear classification, regression, and even outlier detection

Maximum Margin Classifier

- If we strictly impose that all instances be off the street and on the right side, this is called *hard margin classification*
- There are two main issues with hard margin classification
 - It only works if the data is linearly separable
 - It is quite sensitive to outliers



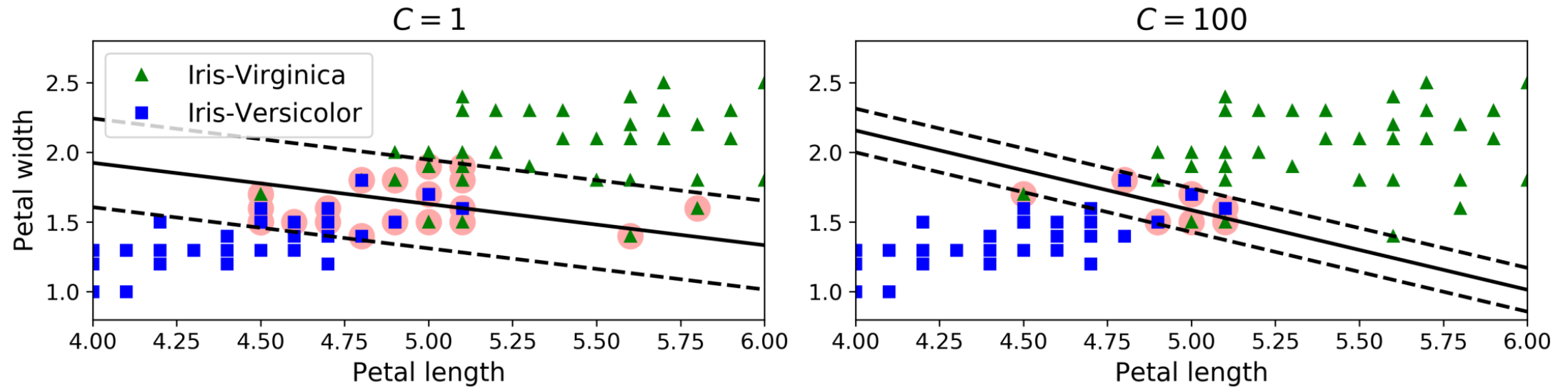
Support Vectors

- The samples on the edge of the boundary lines , are known as '**Support Vectors**'
- Support Vectors are the samples that are most difficult to classify
- They directly affect the process to find the optimum location of the decision boundaries
- Only a very small subset of training samples (Support vectors) can fully specify the decision function
- **If the Support Vectors are removed from the data set, it will potentially change the position of the dividing line**

Maximum Margin Classifier

- In Scikit-Learn's SVM classes, we can control this balance using the C hyperparameter: a smaller C value leads to a wider street but more margin violations
- Next figure shows the decision boundaries and margins of two soft margin SVM classifiers on a nonlinearly separable dataset
- On the left, using a low C value the margin is quite large, but many instances end up on the street
- On the right, using a high C value the classifier makes fewer margin violations but ends up with a smaller margin
- However, it seems likely that the first classifier will generalize better: in fact even on this training set it makes fewer prediction errors, since most of the margin violations are actually on the correct side of the decision boundary

Soft margin



Code for the petal length classification

```
import numpy as np
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
iris = datasets.load_iris()
X = iris["data"][:, (2, 3)] # petal length, petal width
y = (iris["target"] == 2).astype(np.float64) # Iris-Virginica

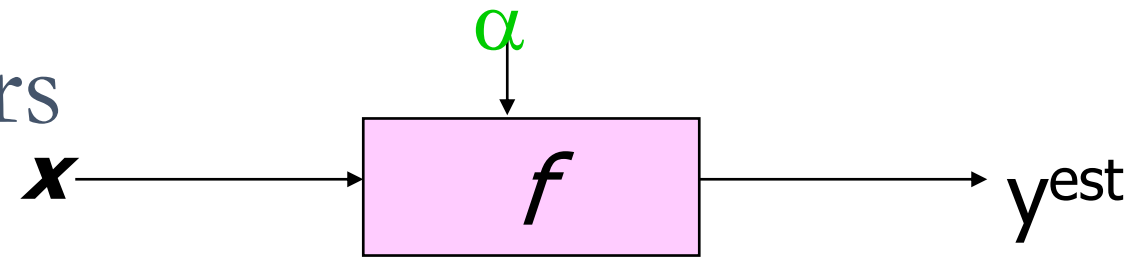
svm_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C=1, loss="hinge")),])

svm_clf.fit(X, y)
```

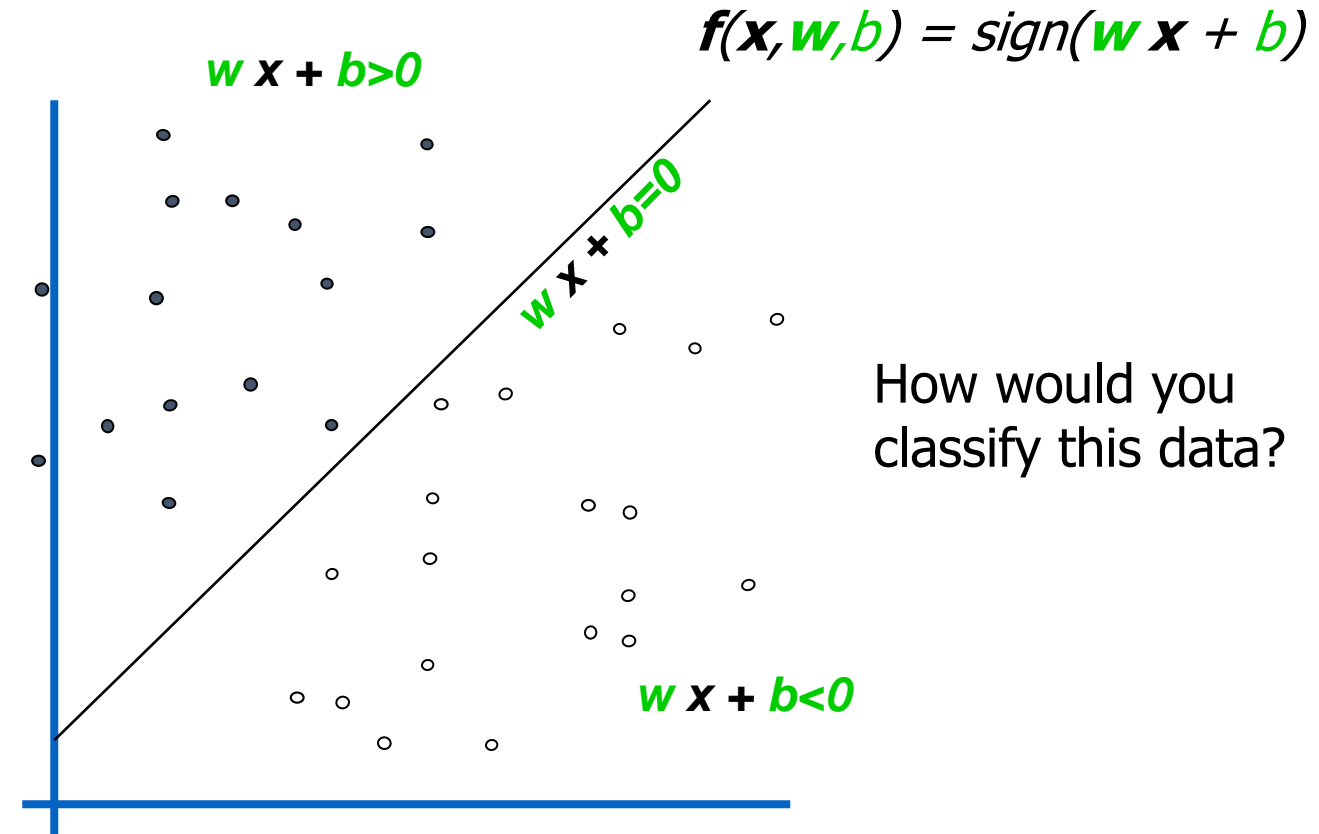
Implementation notes

- The LinearSVC class regularizes the bias term
 - So, data need to be centered around the mean
 - Here StandardScaler does it for us automatically
- We could use the SVC class, using `SVC(kernel="linear", C=1)`,
 - Just using different implementation
 - but it is much slower, especially with large training sets, so it is not recommended
- Another option is to use the SGDClassifier class, with `SGDClassifier(loss="hinge", alpha=1/(m*C))`
 - This applies regular Stochastic Gradient Descent to train a linear SVM classifier
 - It does not converge as fast as the LinearSVC class, but it can be useful to handle huge datasets that do not fit in memory (out-of-core training), or to handle online classification tasks
 -

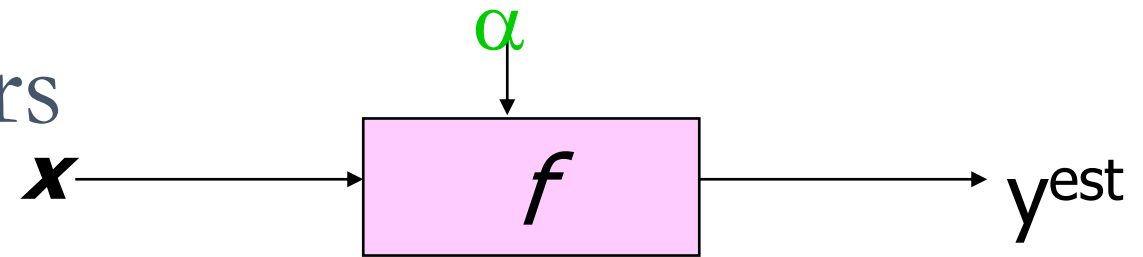
Linear Classifiers



- denotes +1
- denotes -1

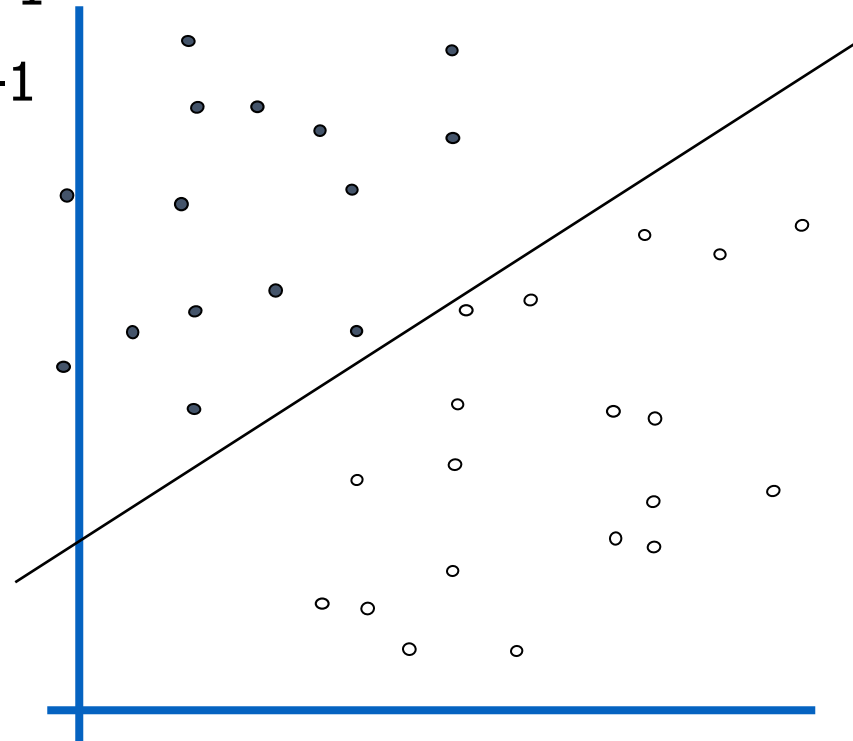


Linear Classifiers



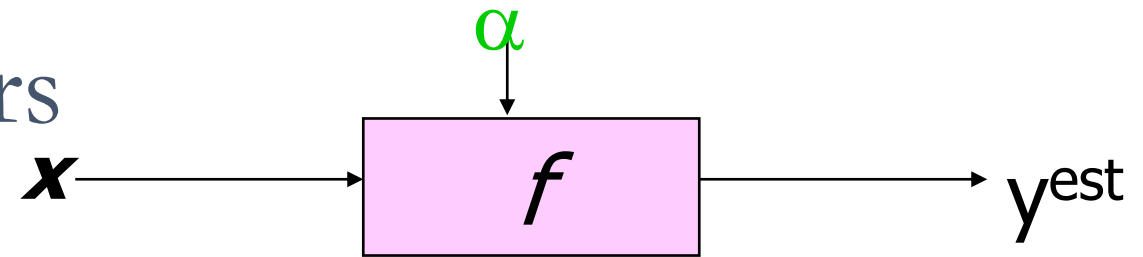
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

- denotes +1
- denotes -1



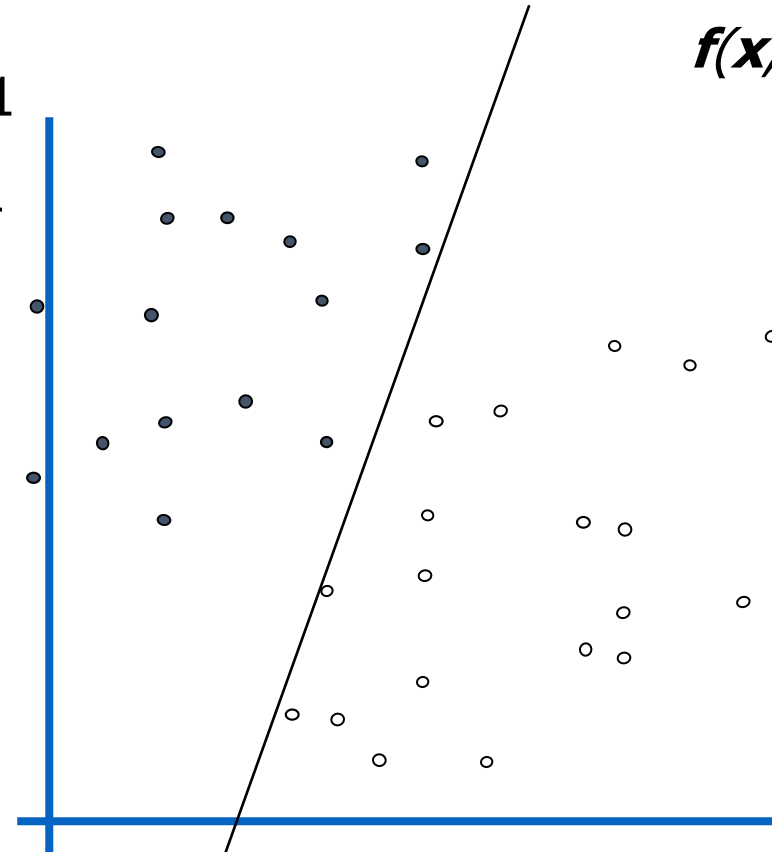
How would you classify this data?

Linear Classifiers



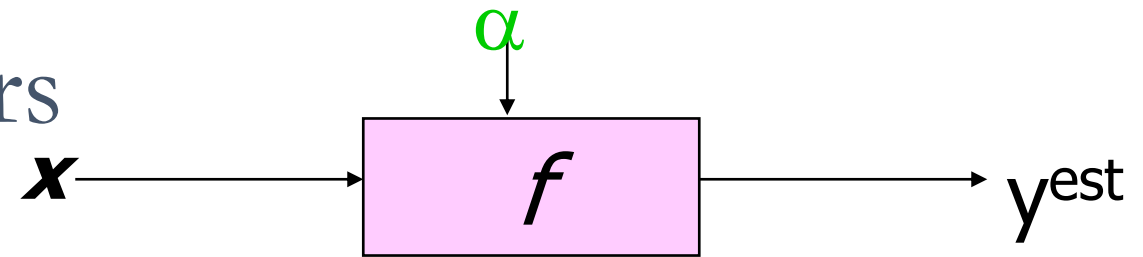
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

- denotes +1
- denotes -1

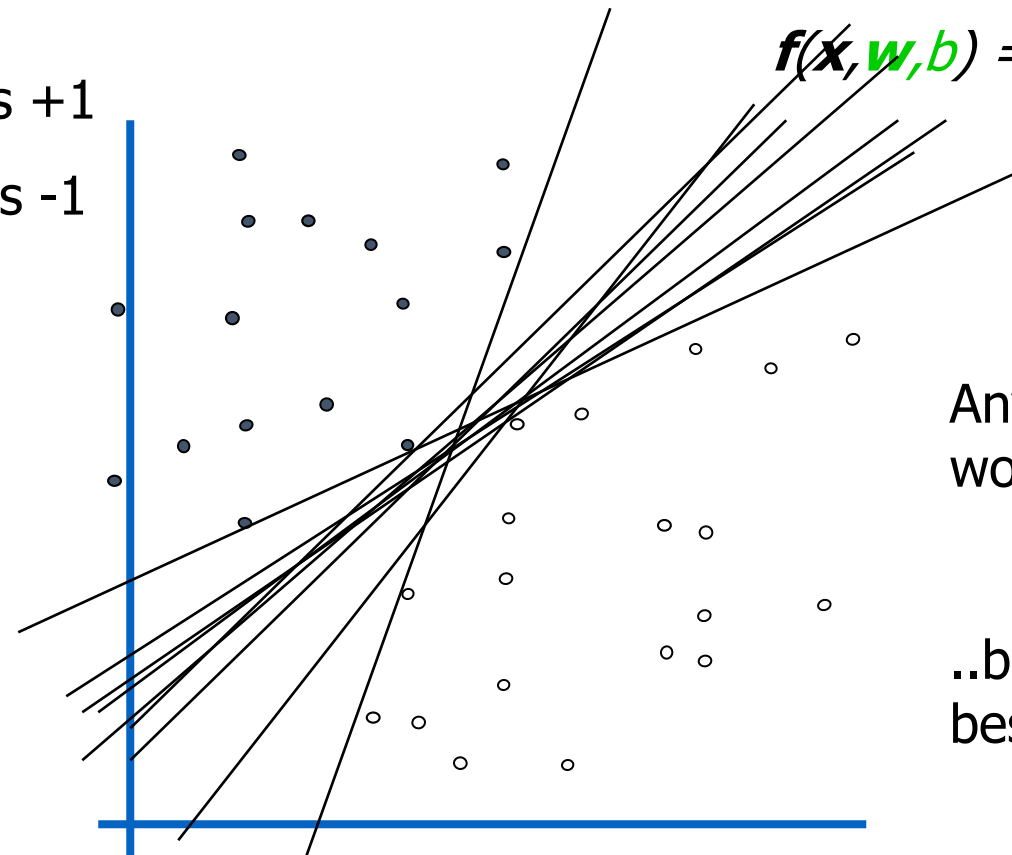


How would you classify this data?

Linear Classifiers



- denotes +1
- denotes -1

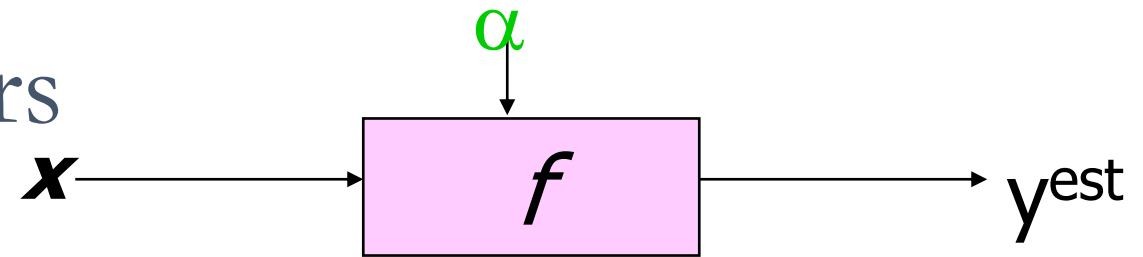


$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

Any of these
would be fine..

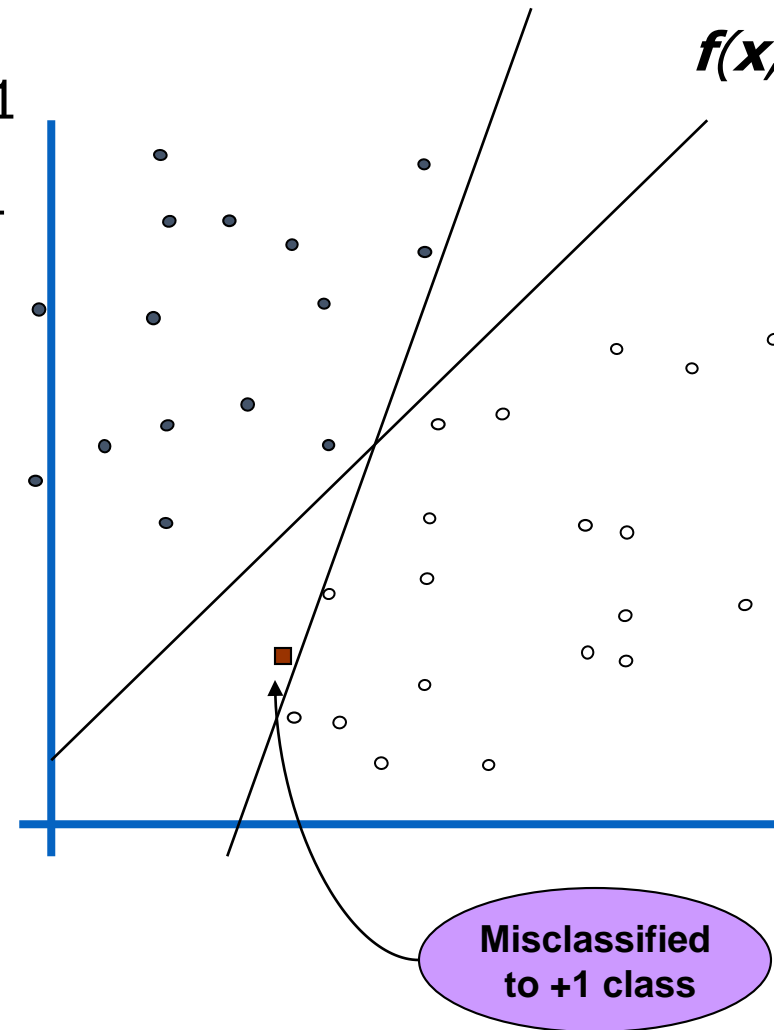
..but which is
best?

Linear Classifiers



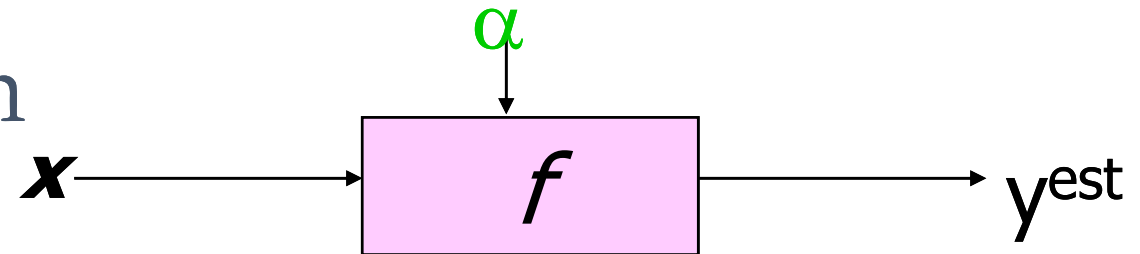
- denotes +1
- denotes -1

$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$



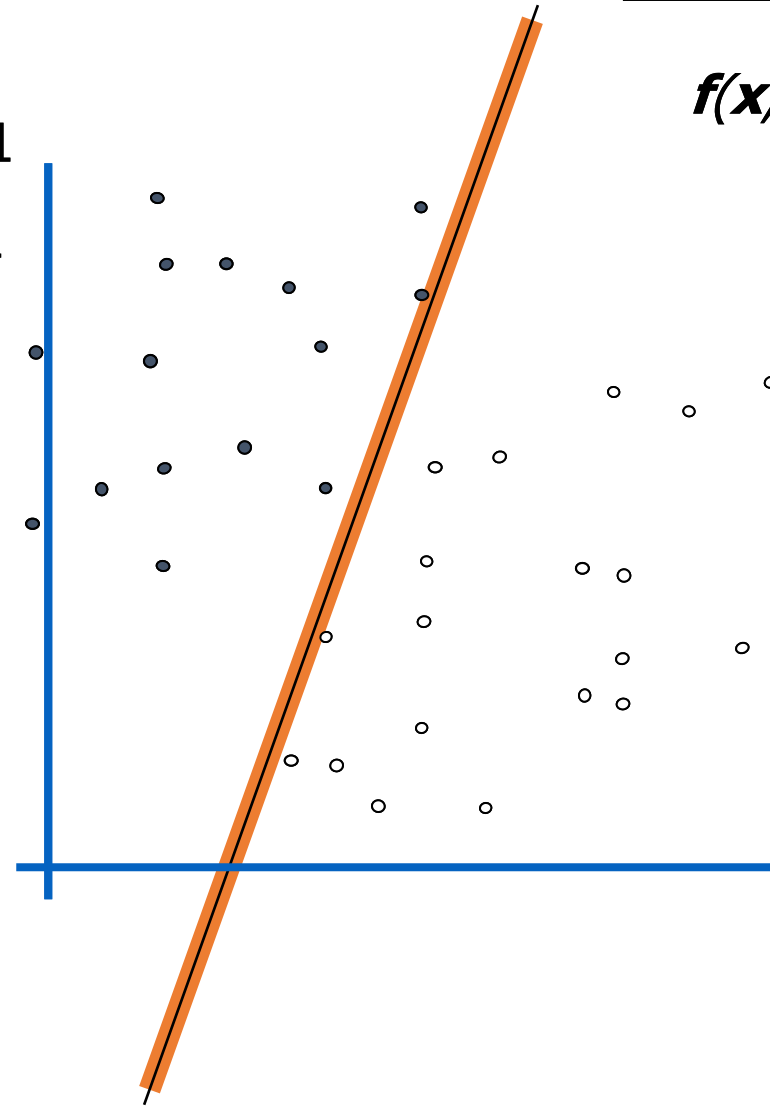
How would you classify this data?

Classifier Margin



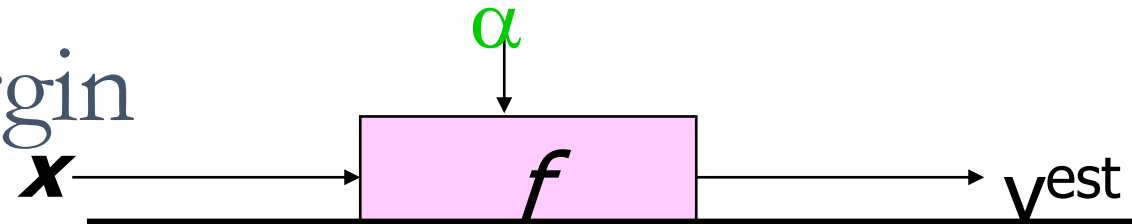
$$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w} \mathbf{x} + b)$$

- denotes +1
- denotes -1



Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

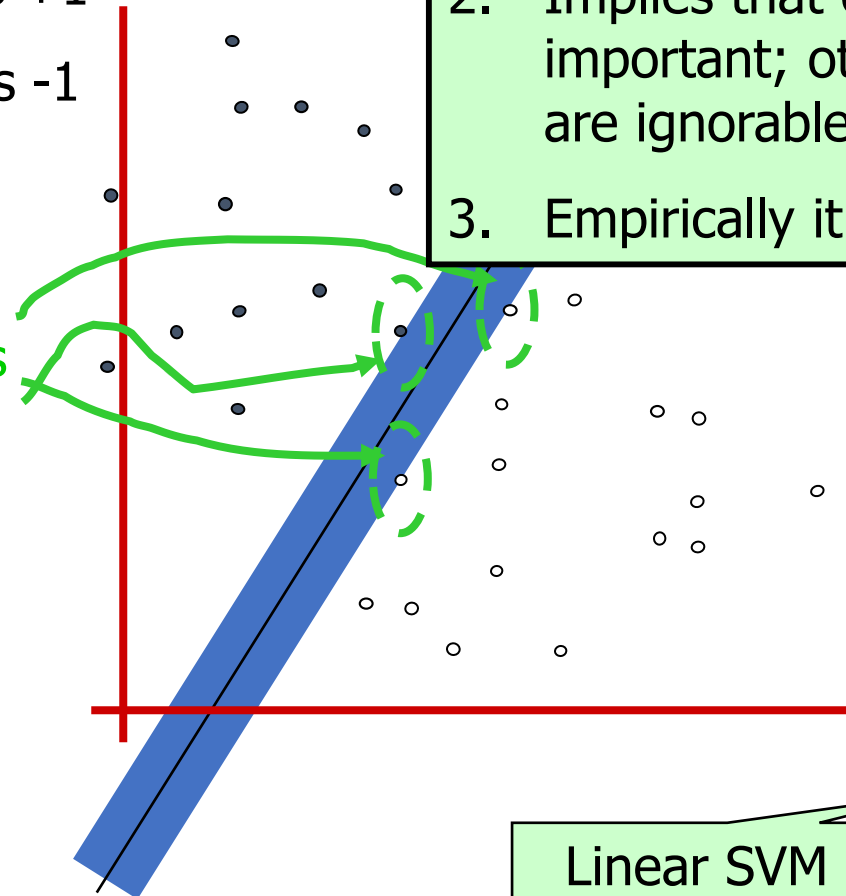
Maximum Margin



- denotes +1
- denotes -1

1. Maximizing the margin is good according to intuition
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Support Vectors
are those datapoints that the margin pushes up against

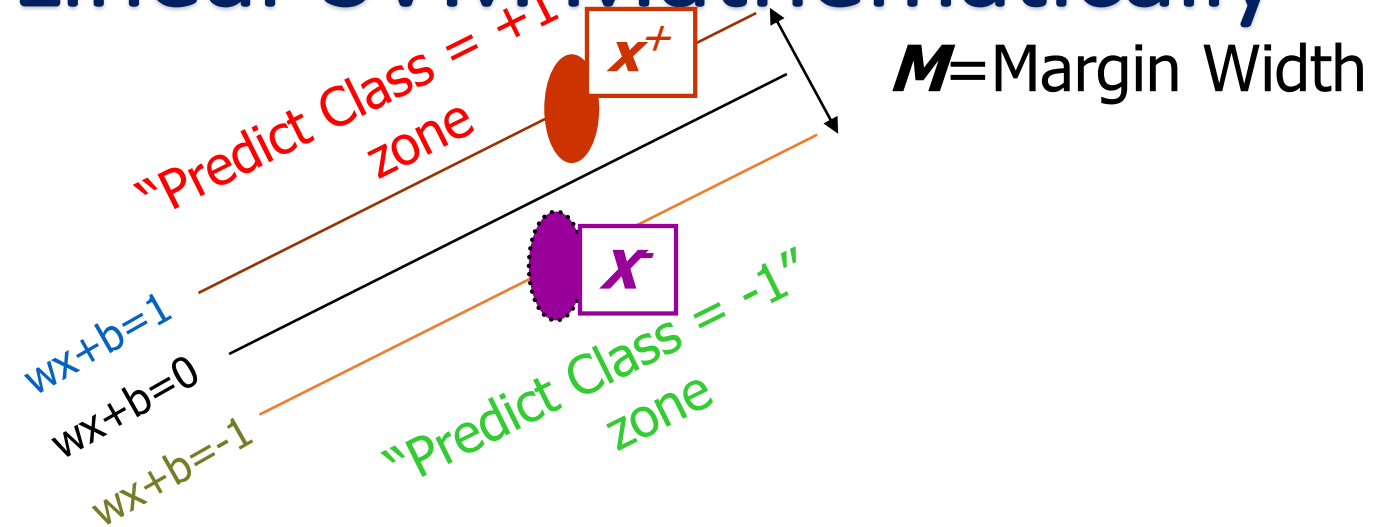


linear classifier
with the maximum
margin.

This is the
simplest kind of
SVM (Called an
LSVM)

Linear SVM

Linear SVM Mathematically



What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^- + b = -1$
- $w \cdot (x^+ - x^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

- Goal: 1) **Correctly classify all training data**

$$\left. \begin{array}{ll} wx_i + b \geq 1 & \text{if } y_i = +1 \\ wx_i + b \leq 1 & \text{if } y_i = -1 \end{array} \right\} \begin{array}{c} \text{ } \\ \text{ } \end{array}$$

$$y_i(wx_i + b) \geq 1 \quad \text{for all } i$$

- 2) **Maximize the Margin**

same as minimize

$$M = \frac{2}{|w|}$$

$$\frac{1}{2} w^t w$$

- We can formulate a Quadratic Optimization Problem and solve for w and b

- Minimize $\Phi(w) = \frac{1}{2} w^t w$
- subject to $y_i(wx_i + b) \geq 1 \quad \forall i$

Solving the Optimization Problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- **Need to optimize a *quadratic* function subject to *linear* constraints.**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Find $\alpha_1 \dots \alpha_N$ such that

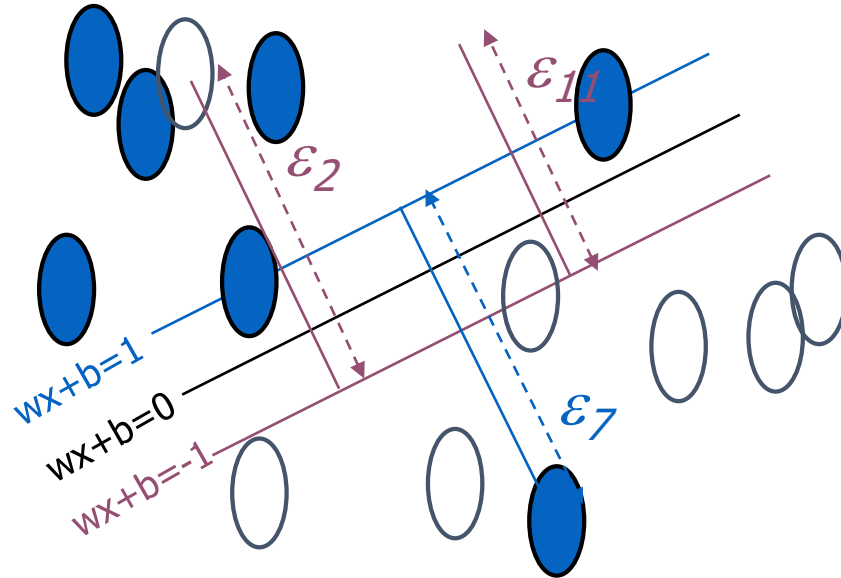
$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

(1) $\sum \alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.

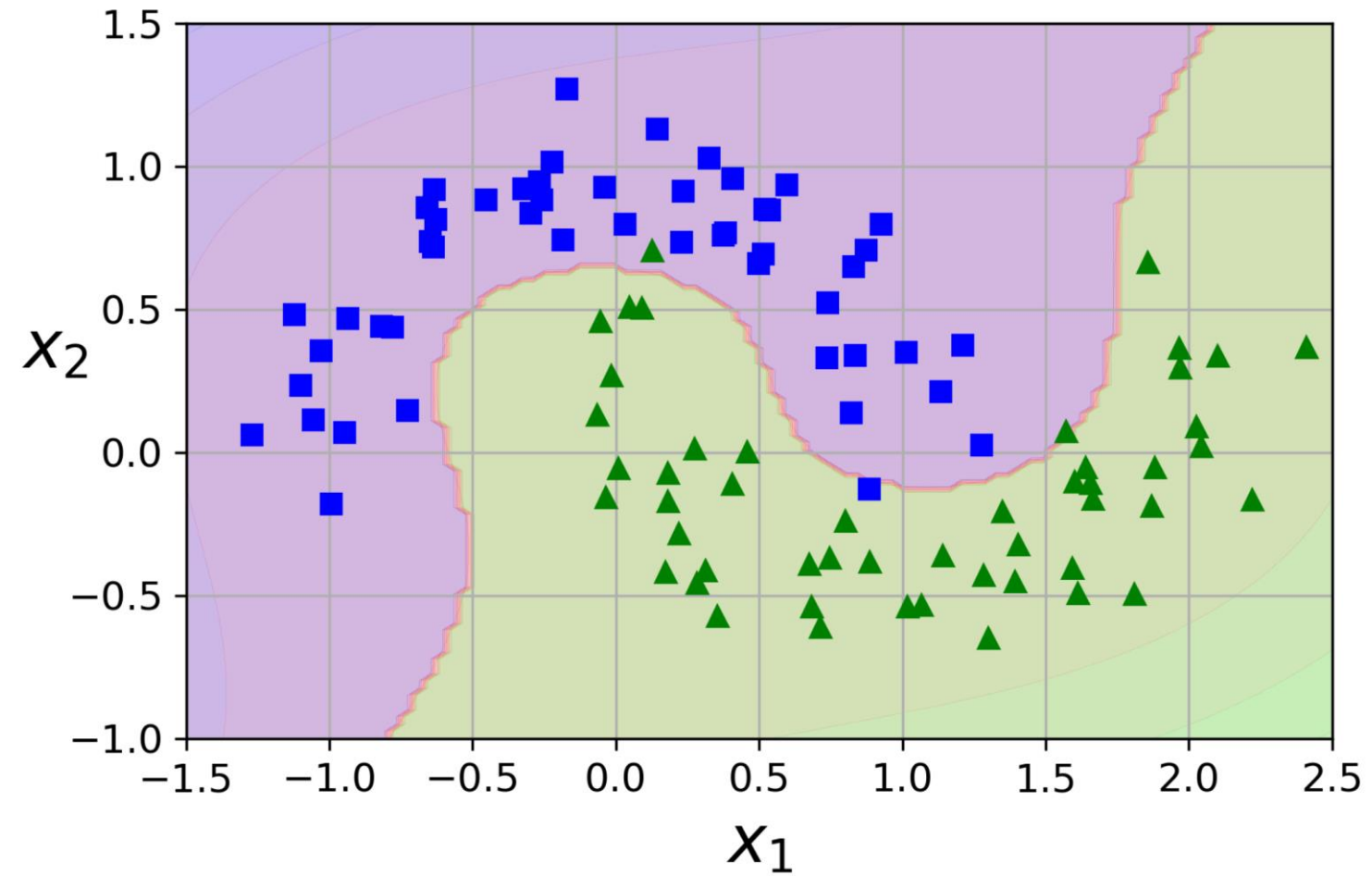


What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \xi_k$$

Non linear SVM

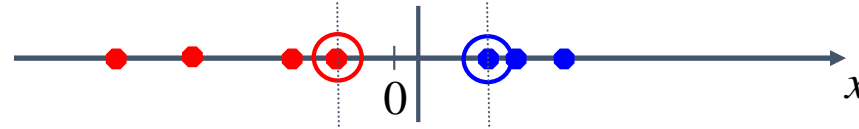


Non linear SVM

- Although linear SVM classifiers are efficient and work surprisingly well in many cases, many datasets are not even close to being linearly separable
- One approach to handling nonlinear datasets is to add more features, such as polynomial features
- In some cases this can result in a linearly separable dataset

Non-linear SVMs

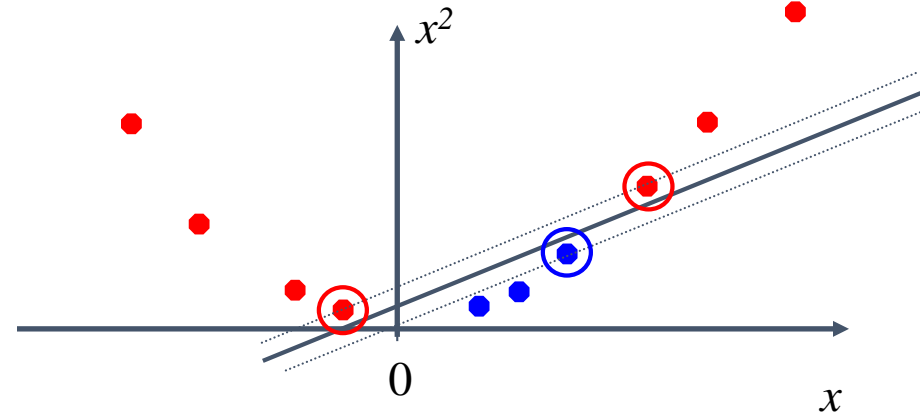
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

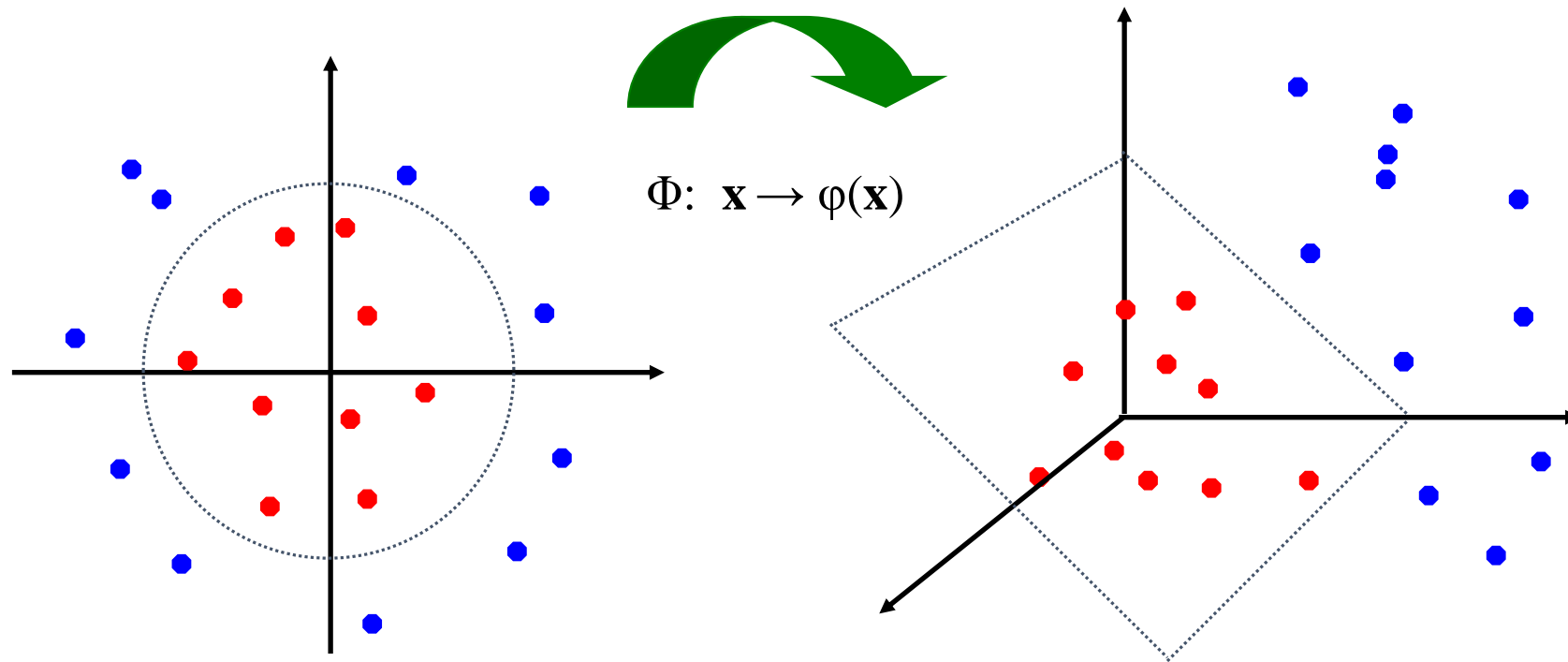


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

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



Some kernel functions

$K(x_i, x_j) = (x_i \cdot x_j + 1)^p$; polynomial kernel.

$K(x_i, x_j) = e^{\frac{-1}{2\sigma^2} (x_i - x_j)^2}$; Gaussian kernel; Special case of Radial Basis Function.

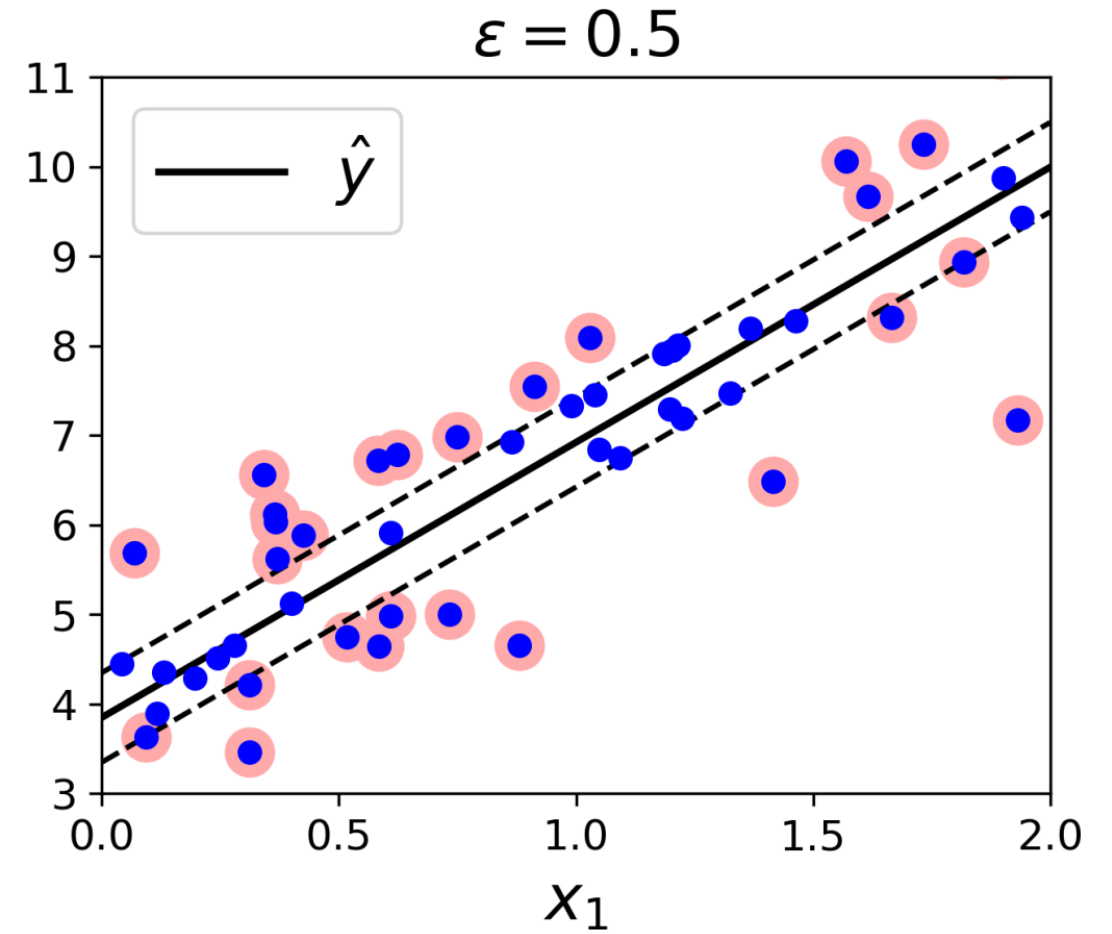
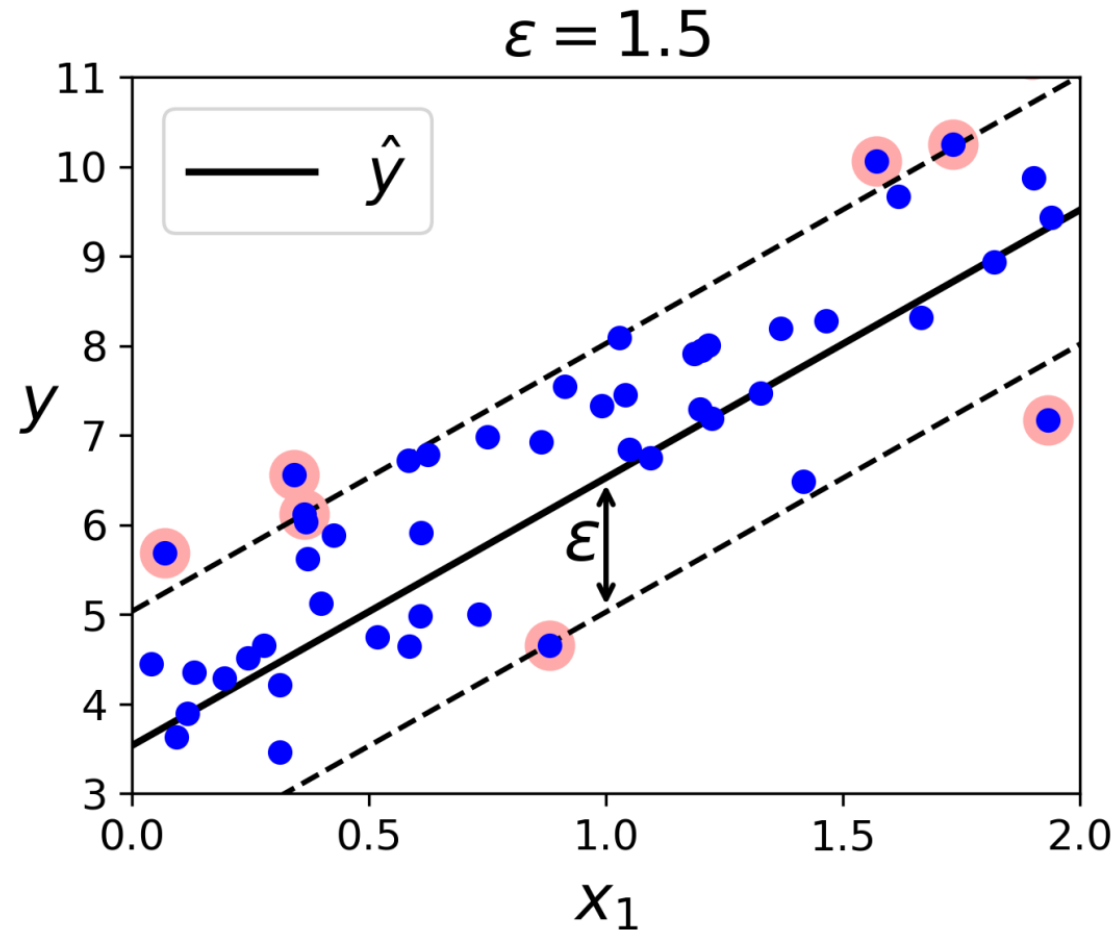
$K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$; RBF Kernel

$K(x_i, x_j) = \tanh(\eta x_i \cdot x_j + \nu)$; Sigmoid Kernel; Activation function for NN.

SVM regression

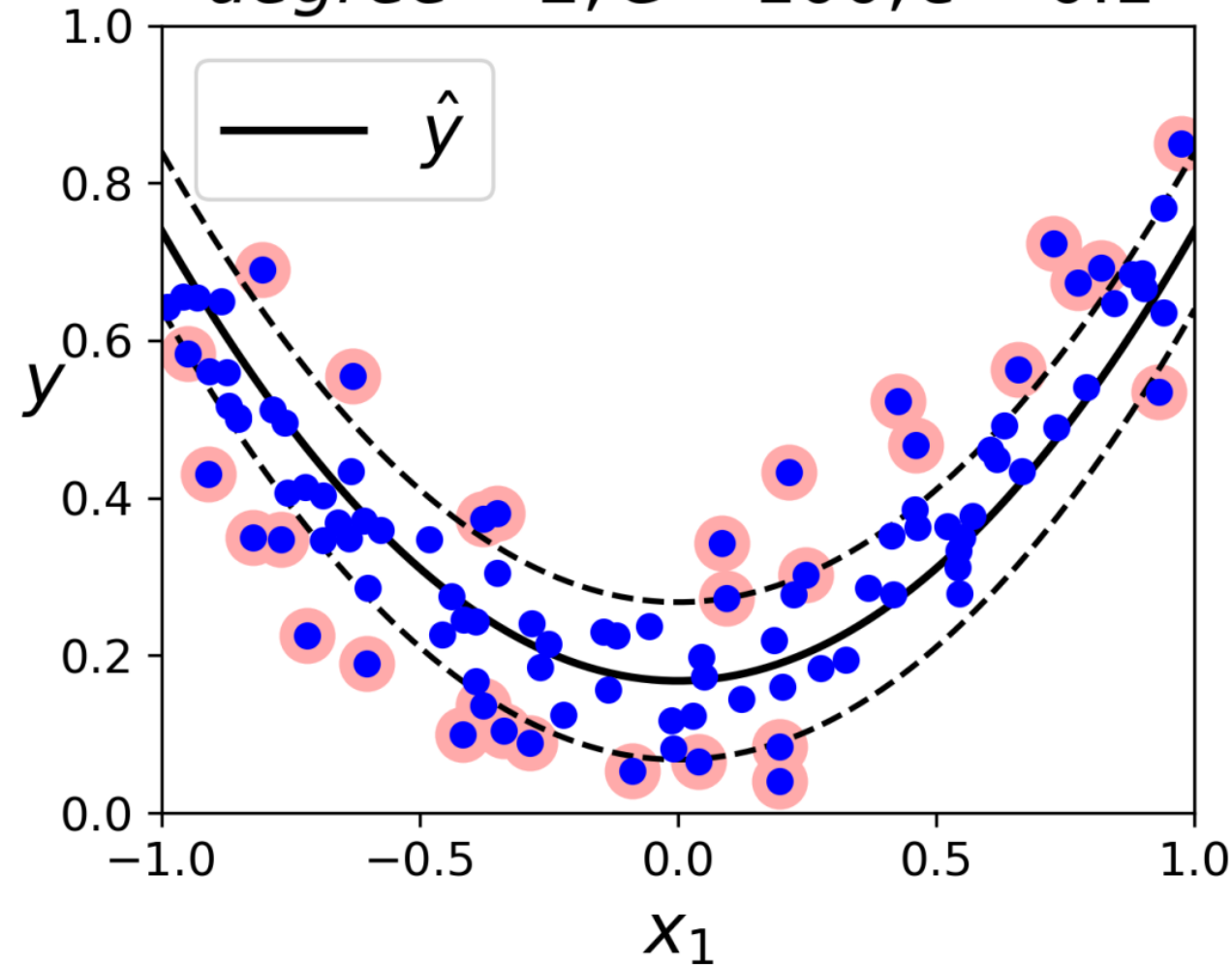
- As we mentioned earlier, the SVM algorithm is quite versatile: not only does it support linear and nonlinear classification, but it also supports linear and nonlinear regression
- The trick is to reverse the objective: instead of trying to fit the largest possible street between two classes while limiting margin violations, SVM Regression tries to fit as many instances as possible *on* the street while limiting margin violations (i.e., instances *off* the street)
- The width of the street is controlled by a hyper-parameter ϵ .

SVM regression

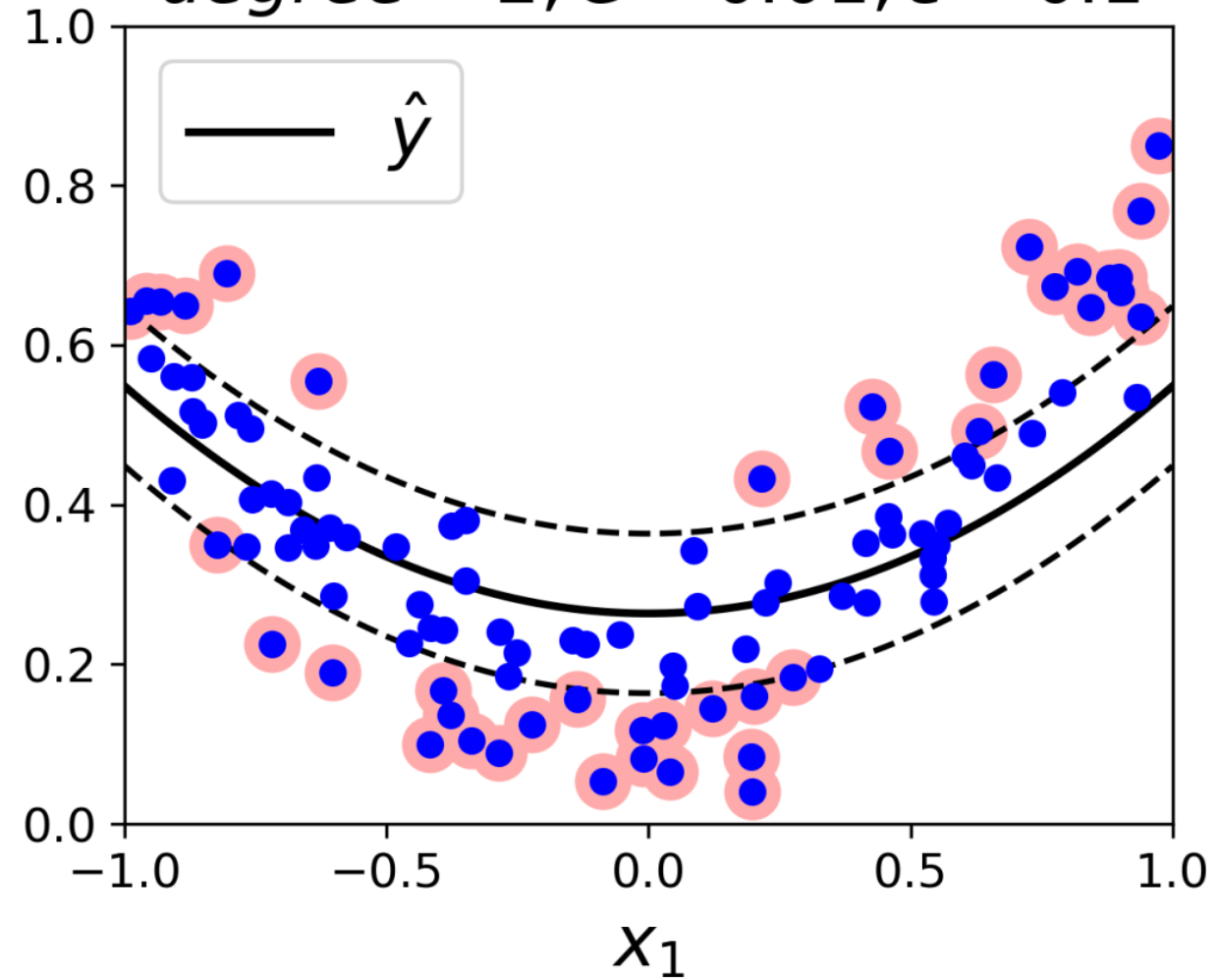


SVM regression

degree = 2, $C = 100, \varepsilon = 0.1$



degree = 2, $C = 0.01, \varepsilon = 0.1$

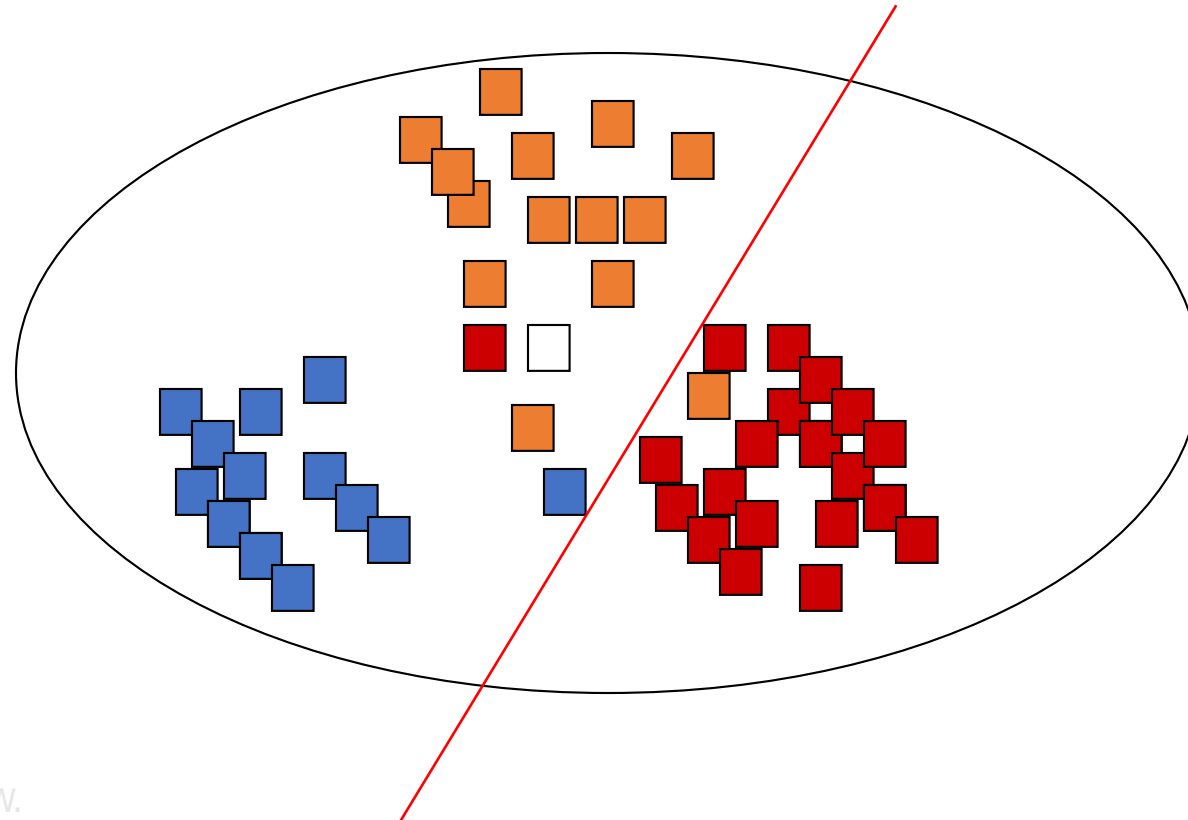


Doing multi-class classification



- SVMs can only handle two-class outputs (i.e. a categorical output variable with arity 2).
- How to handle multiple classes
 - E.g., classify documents into three categories: *sports, business, politics*
- Answer: one-vs-all, learn N SVM's
 - SVM 1 learns "Output==1" vs "Output != 1"
 - SVM 2 learns "Output==2" vs "Output != 2"
 - :
 - SVM N learns "Output==N" vs "Output != N"

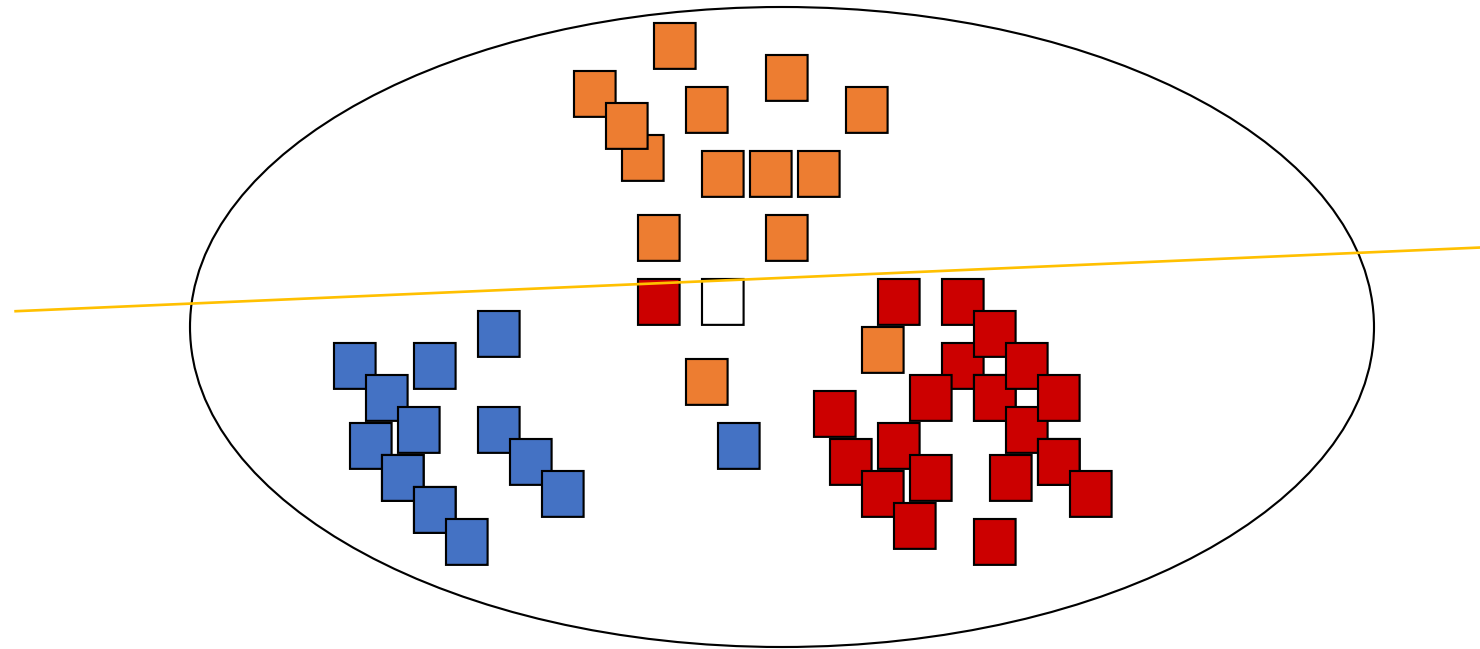
One-vs-All

- ■ vs the other classes: red(d)






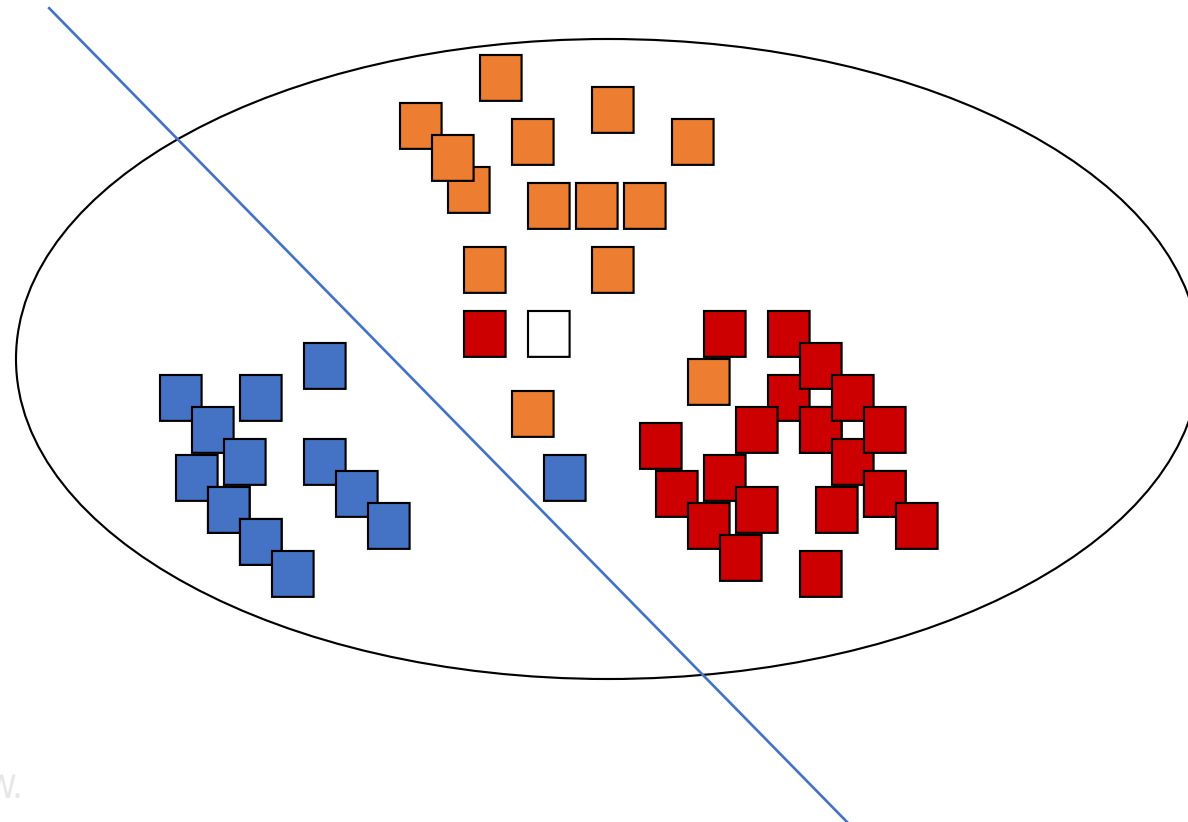
One-vs-All

-  vs the other classes: red(d)
-  vs the other classes: yellow(d)






One-vs-All




-  vs the other classes: $\text{red}(d)$
-  vs the other classes: $\text{yellow}(d)$
-  vs the other classes: $\text{cyan}(d)$



One-vs-All

-  vs the other classes: red(d)
-  vs the other classes: yellow(d)
-  vs the other classes: cyan(d)
- Given a test document d, how to decide its color ?

One-vs-All

-  vs the other classes: red(d)
-  vs the other classes: yellow(d)
-  vs the other classes: cyan(d)
- Given a test document d, how to decide its color ?
- Assign d to the color function with the largest score

Weakness of SVM

- It is sensitive to noise
 - A relatively small number of mislabeled examples (especially support vectors) can dramatically decrease the performance
- It only considers two classes
 - how to do multi-class classification with SVM?
 - Answer:
 - 1) with output arity m , learn m SVM's
 - SVM 1 learns "Output==1" vs "Output != 1"
 - SVM 2 learns "Output==2" vs "Output != 2"
 - :
 - SVM m learns "Output== m " vs "Output != m "
 - 2) To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Model selection Issues

- **Choice of kernel**
 - RBF/Gaussian or polynomial kernel is default
 - if ineffective, more elaborate kernels are needed
 - domain experts can give assistance in formulating appropriate similarity measures
- **Choice of kernel parameters**
 - e.g. σ in Gaussian kernel
 - σ is the distance between closest points with different classifications
 - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- **Optimization criterion** – Hard margin v.s. Soft margin
 - a lengthy series of experiments in which various parameters are tested