SVM

Support Vector Machines

CS 6316 – Machine Learning
Fall 2017

OUTLINE

- Classification: Discriminative: e.g. SVM
- History of SVM
- SVM & Related Elements
- Examples
- Software (some resources)

Major Sections of Classification

- Discriminative
  - Directly estimate a decision rule/boundary
  - E.g. support vector machine, Decision tree
- Generative
  - Build a generative statistical model
  - E.g. Bayesian networks
- Instance based
  - Use observation directly (no models)
  - E.g. k-nearest neighbors

History of SVM

- SVM is inspired from statistical learning theory (Vapnik)
- SVM was first introduced in 1992
- Became popular because of its success in handwritten digit recognition (1994)
  - Test error rate for SVM
  - Similar error rates achieved by carefully constructed neural network, LeNet 4
- Was one of the hottest areas in ML ~20 years ago
- SVMs are still used and very popular today

Motivation: Philosophical

- **Classical view**: good model
  - explains the data + low complexity

  → Occam's razor (complexity ~ # parameters)
- **VC theory**: good model
  - explains the data + low VC-dimension

  → VC-falsifiability (small VC-dim ~ large falsifiability), i.e. the goal is to find a model that:

  can explain training data / cannot explain other data
- The idea: falsifiability ~ *empirical loss function*
- Large degree of falsifiability is achieved by
  - Large margin (classification)
  - Small epsilon (regression)

SVM Model Complexity

- Two ways to control model complexity

  -via model parameterization $f(\mathbf{x},\omega)$

  use fixed loss function: $L(y, f(\mathbf{x},\omega))$

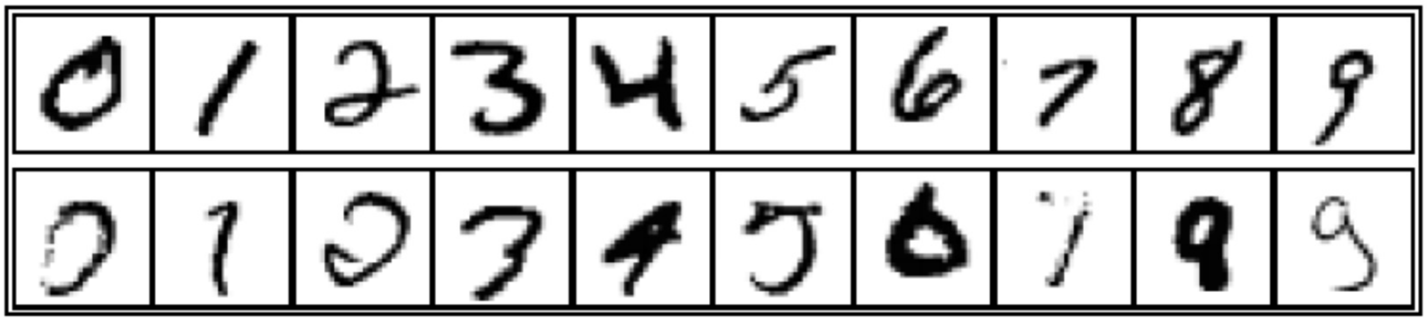
  -via adaptive loss function: $L_\Delta(y, f(\mathbf{x},\omega))$

  use fixed (linear) parameterization $f(\mathbf{x},\omega) = (\mathbf{w}\cdot\mathbf{x}) + b$

- ~ Two types of SRM structures
- Margin-based loss can be motivated by Popper's falsifiability

Brief Application Example

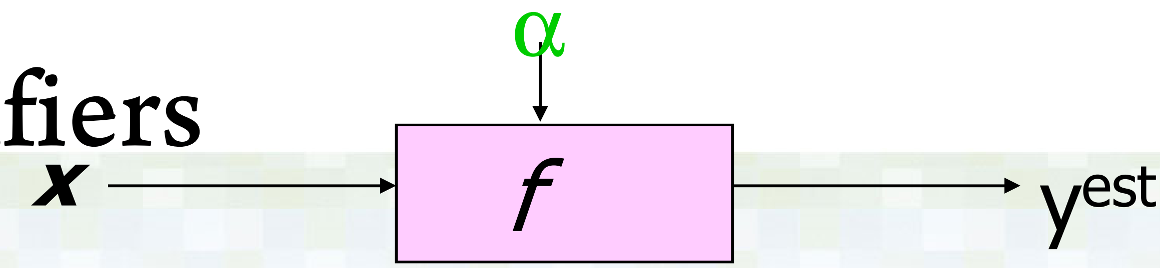
- MNIST handwritten digit database



3-nearest-neighbor = 2.4% error
400–300–10 unit MLP = 1.6% error
LeNet: 768–192–30–10 unit MLP = 0.9% error

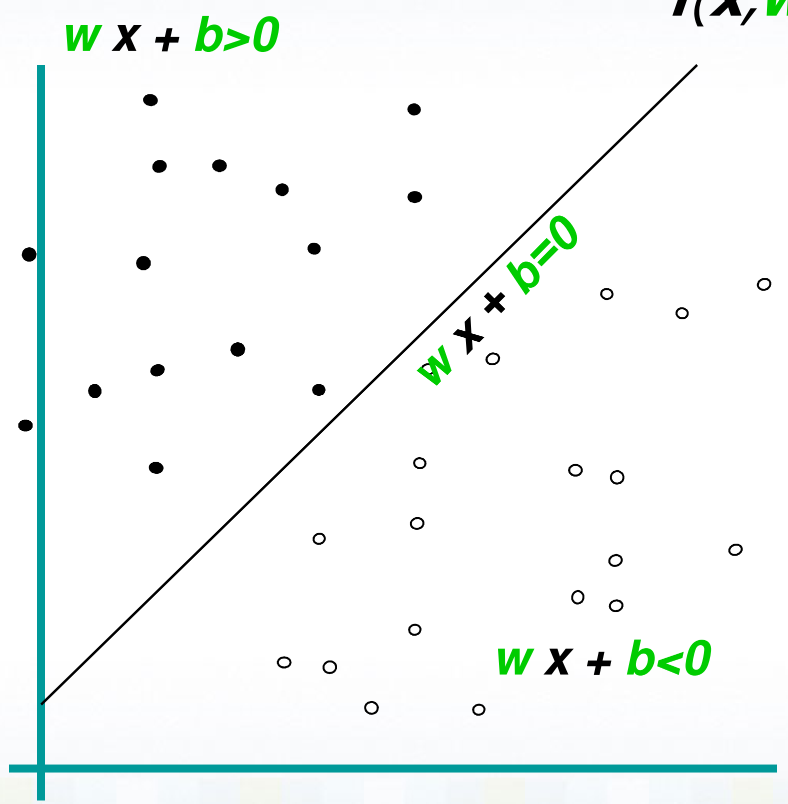
- In 90s, **SVM** achieves the best ~ 0.6% error

Linear Classifiers



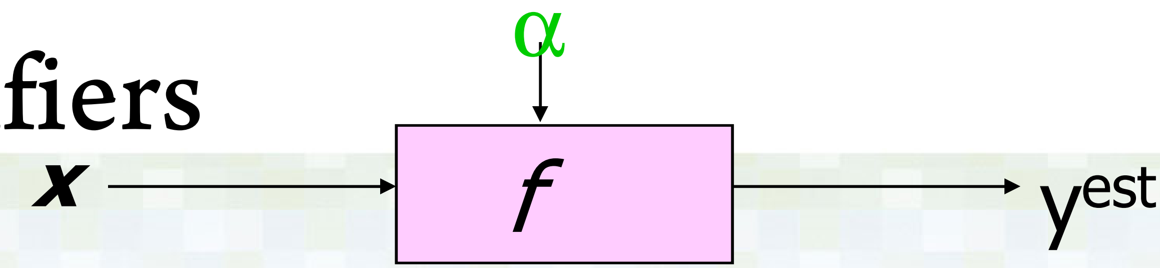
$f(x,w,b) = sign(w\ x + b)$

- • denotes +1
- ◦ denotes -1

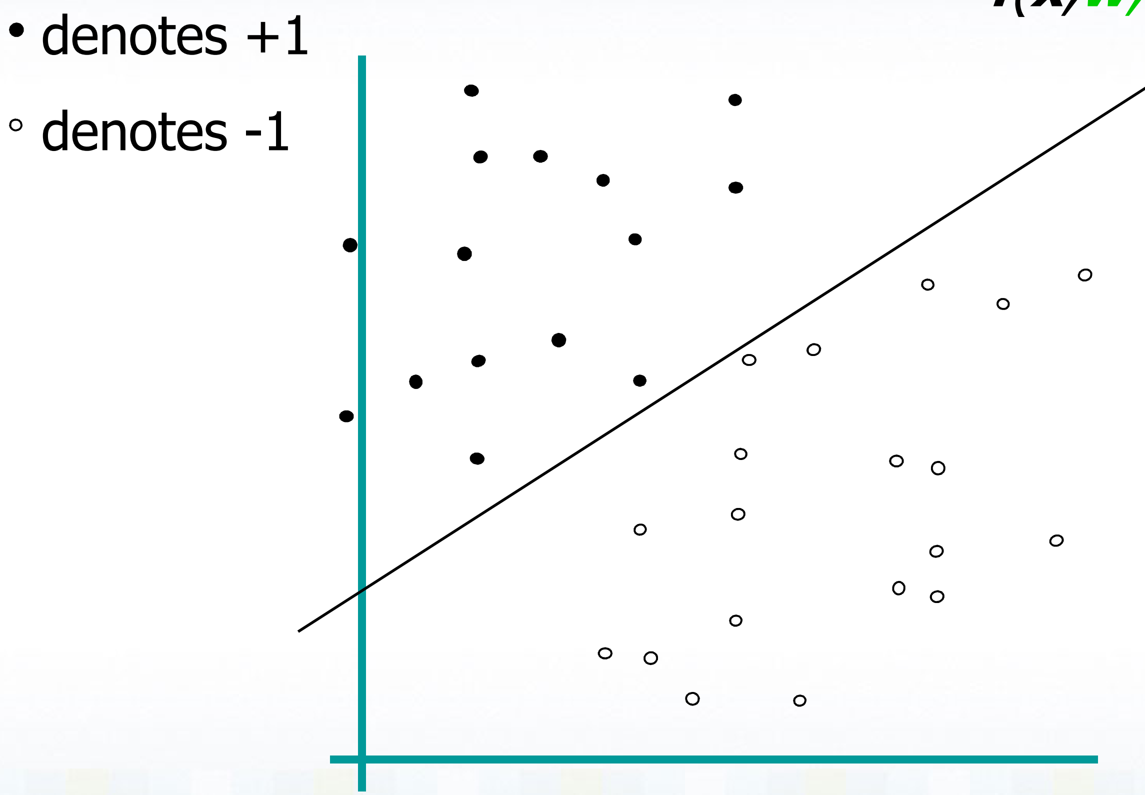


How would you classify this data?

Linear Classifiers

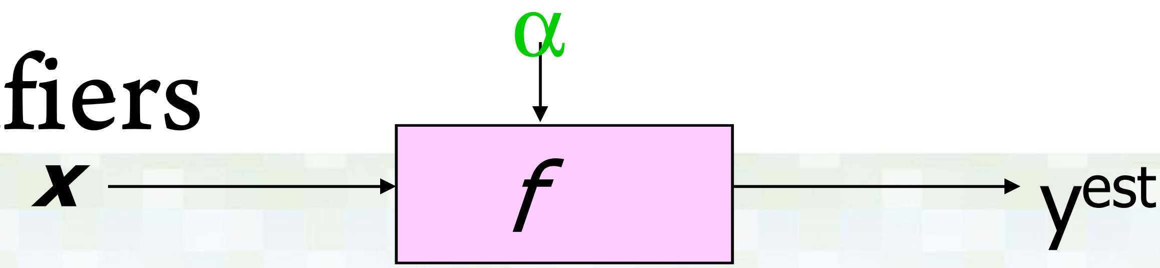


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



How would you classify this data?

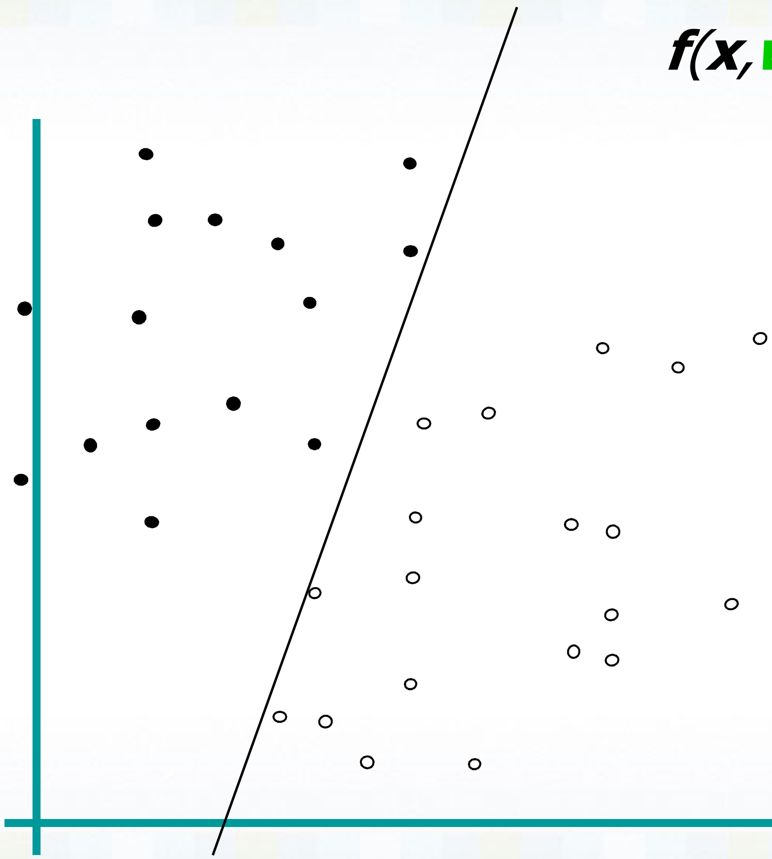
Linear Classifiers



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

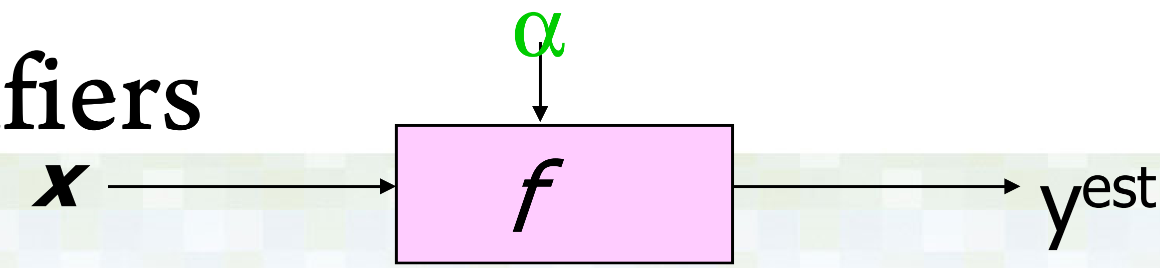
• denotes +1

◦ denotes -1



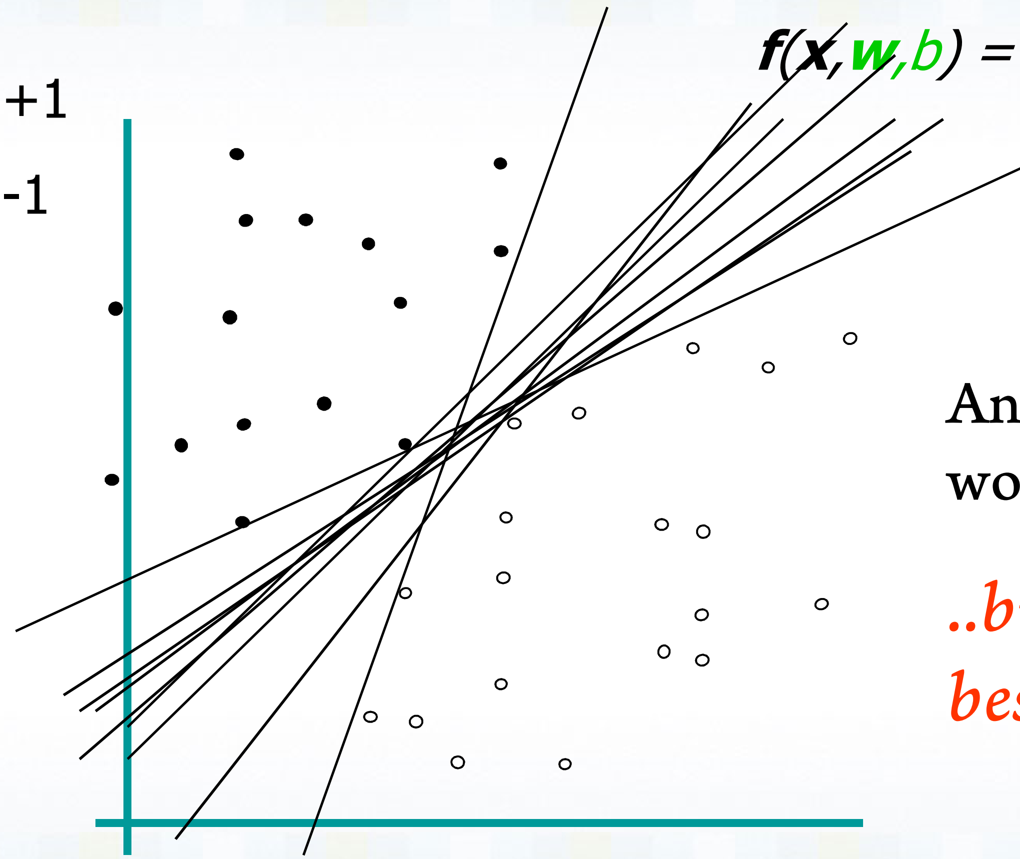
How would you classify this data?

Linear Classifiers



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

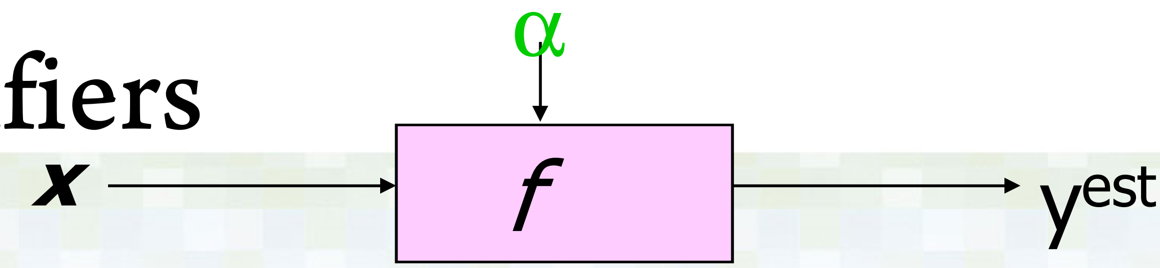
- • denotes +1
- ◦ denotes -1



Any of these would be fine..

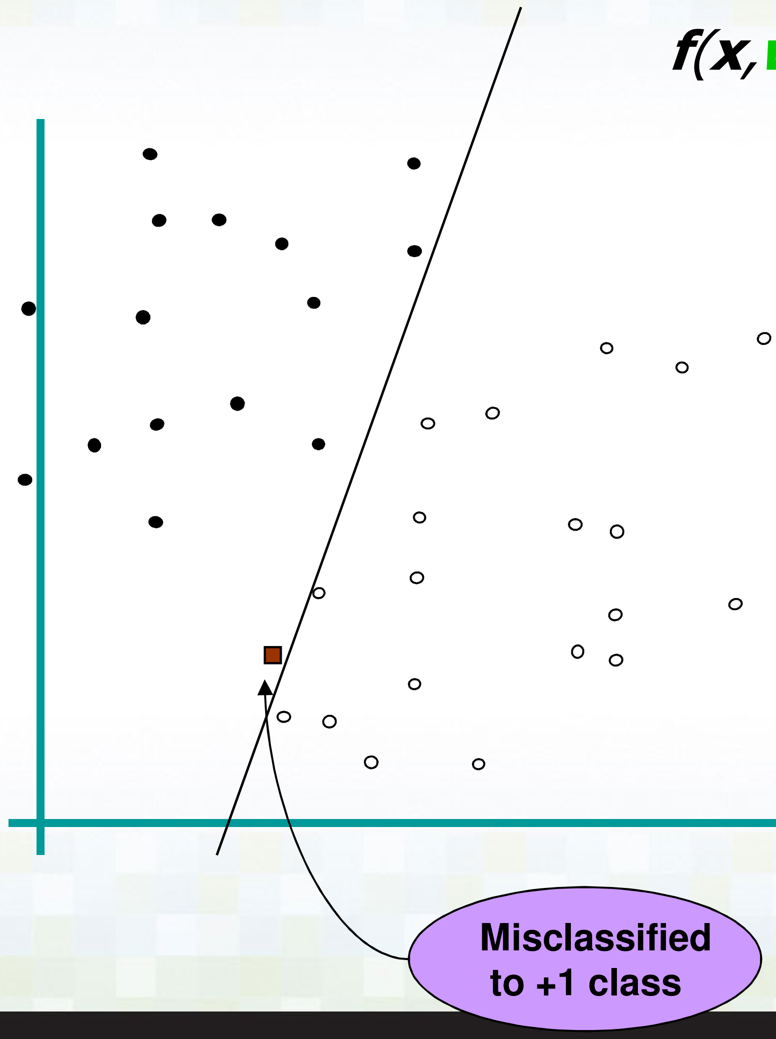
..but which is best?

Linear Classifiers



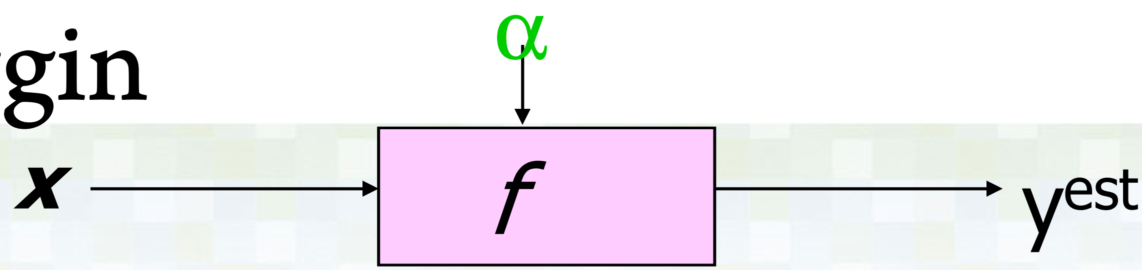
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

- • denotes +1
- ◦ denotes -1

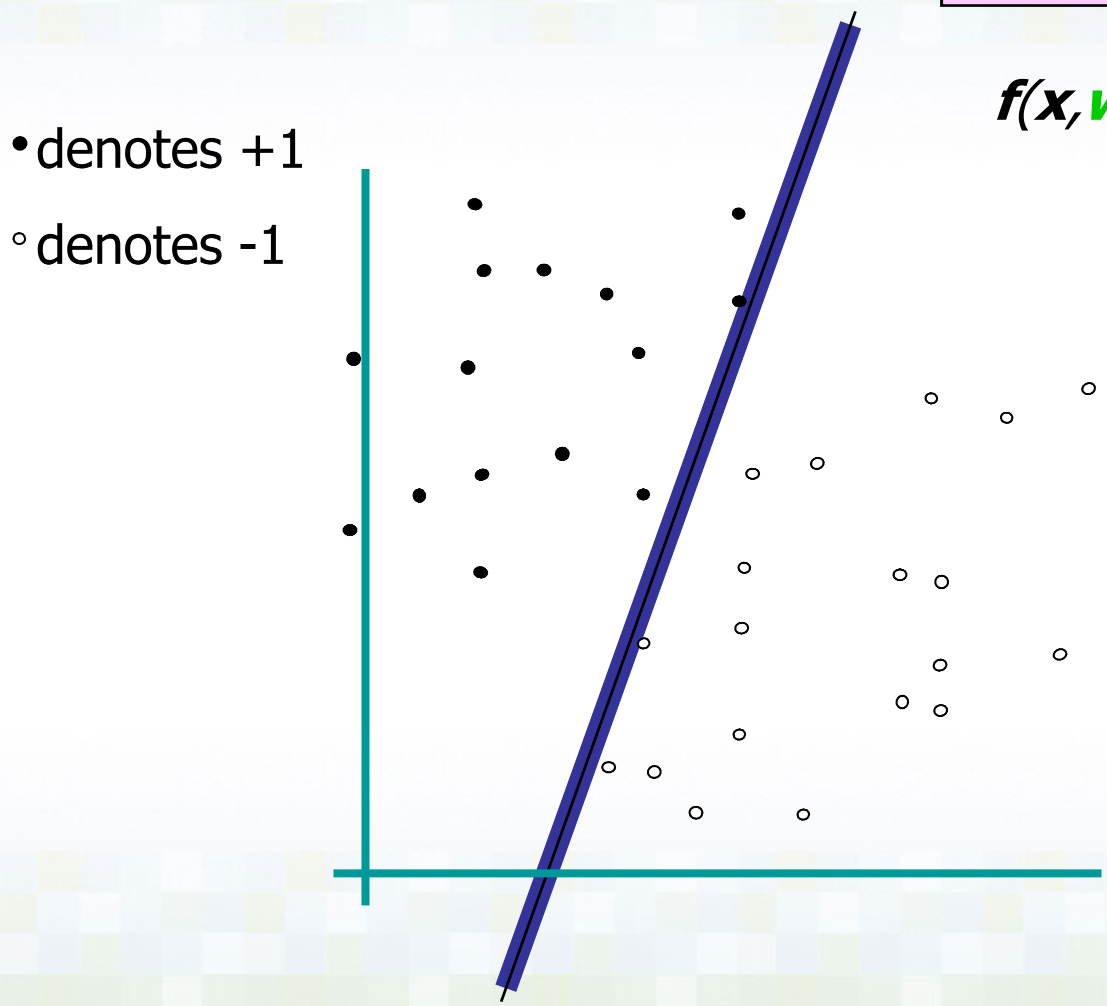


How would you classify this data?

Classifier Margin

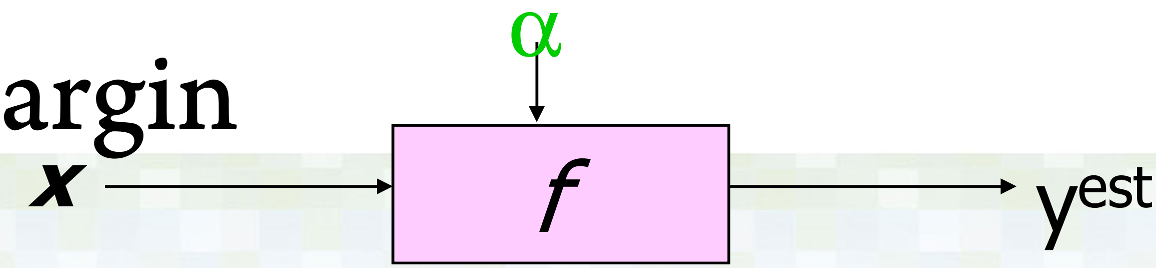


$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



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

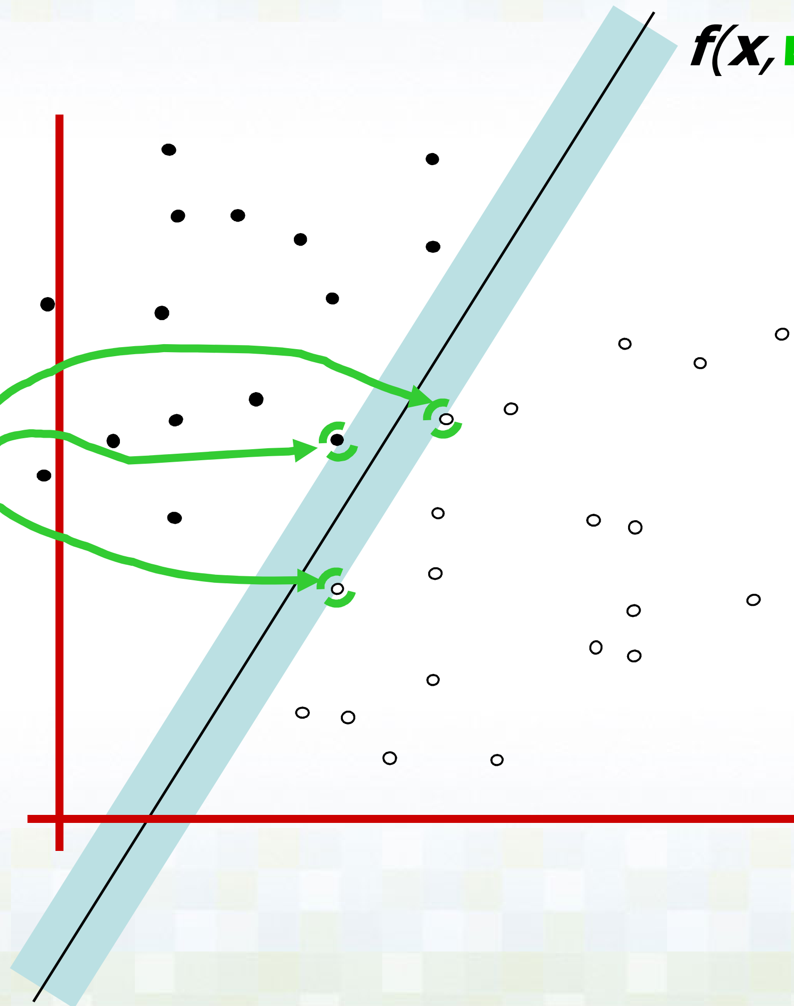
Maximum Margin



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

• denotes +1

° denotes -1



Support Vectors are those datapoints that the margin pushes up against

The maximum margin linear classifier is the linear classifier with the ... maximum margin!

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

~ *Linear SVM*)

Maximum Margin

α

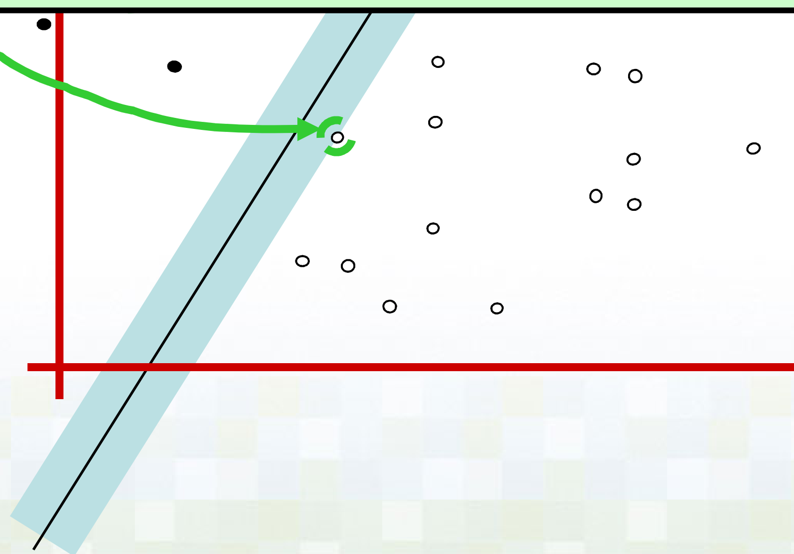
y^{est}

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

• denotes +1

° denotes -1

gn(***w*** ***x*** + *b*)



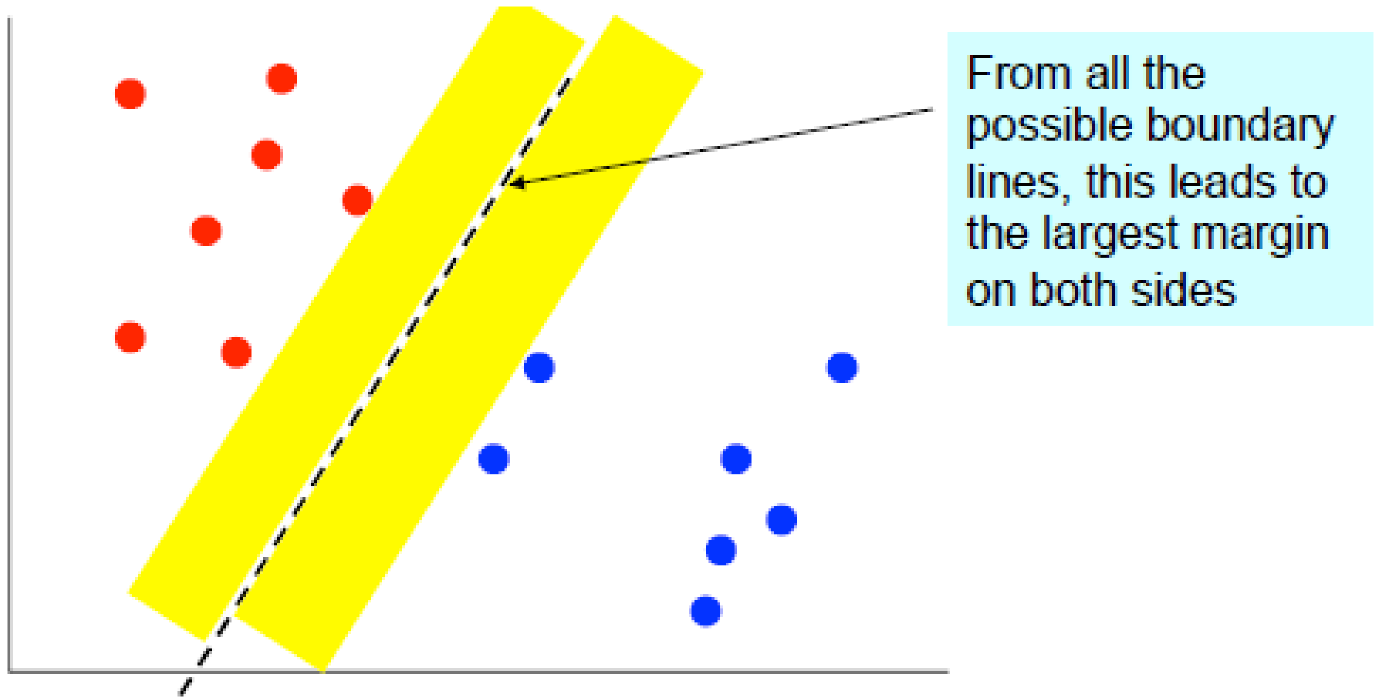
Support Vectors are those datapoints that the margin pushes up against

maximum gin linear ifier is the r classifier with the … maximum margin!

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

Maximum Margin Classifiers

- Instead of fitting ALL points, focus on boundary points
- Learn a boundary that leads to the largest margin from both sets of points

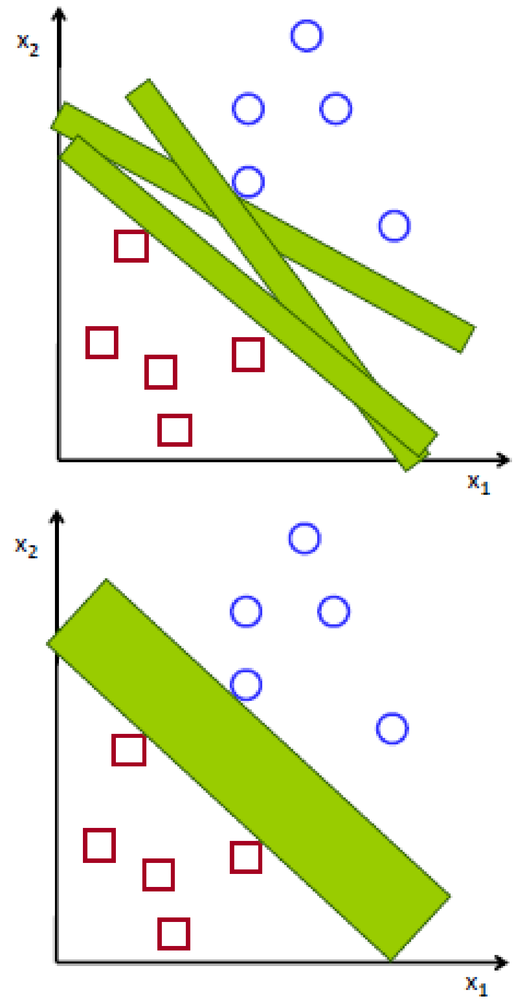


To further understand the relationship between margin and capacity, consider the two separating hyperplanes below

- A "skinny" one (small margin), which will be able to adopt many orientations
- A "fat" one (large margin), which will have limited flexibility

A larger margin necessarily results in lower capacity

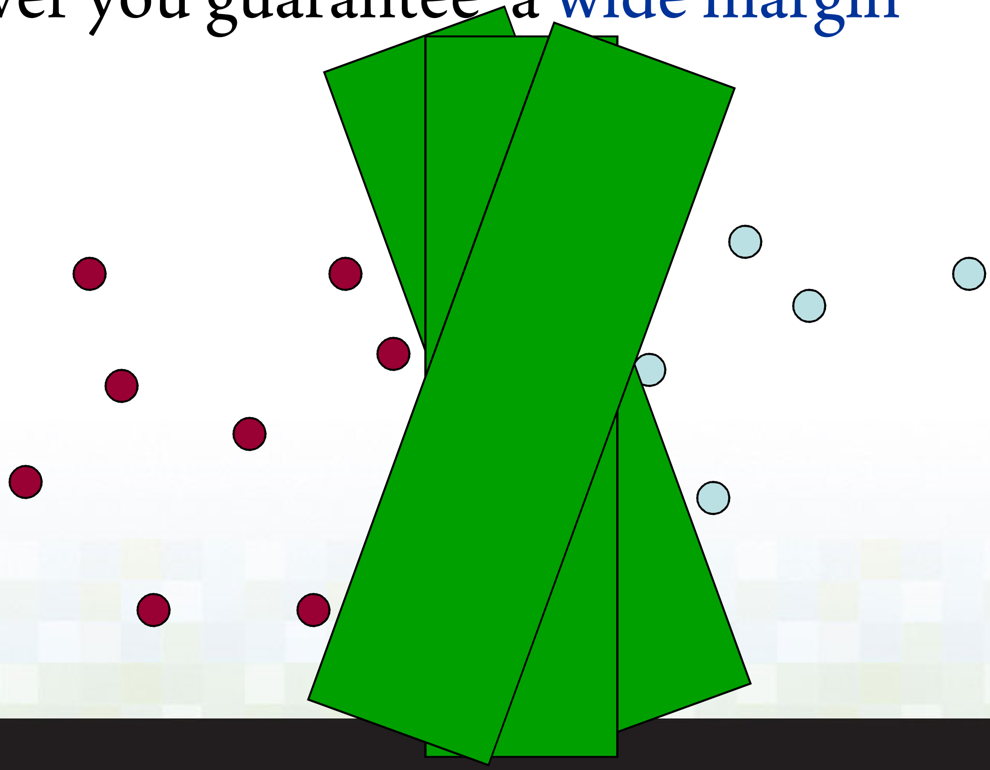
- We normally think of complexity as being a function of the number of parameters
  - Instead, SLT tells us that if the margin is sufficiently large, the complexity of the function will be low even if the dimensionality is very high!



[Bennett and Campbell, 2000]

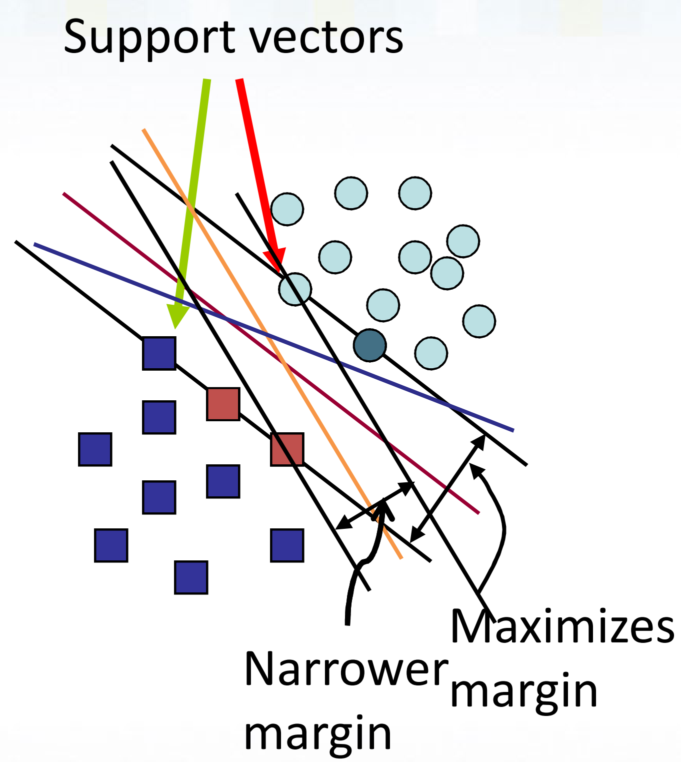
“Fat” separators

- If you have to place a ‘fat’ separator between classes you have less choices, and so the capacity of the model has been decreased
- However you guarantee a wide margin



Support Vector Machine (SVM)

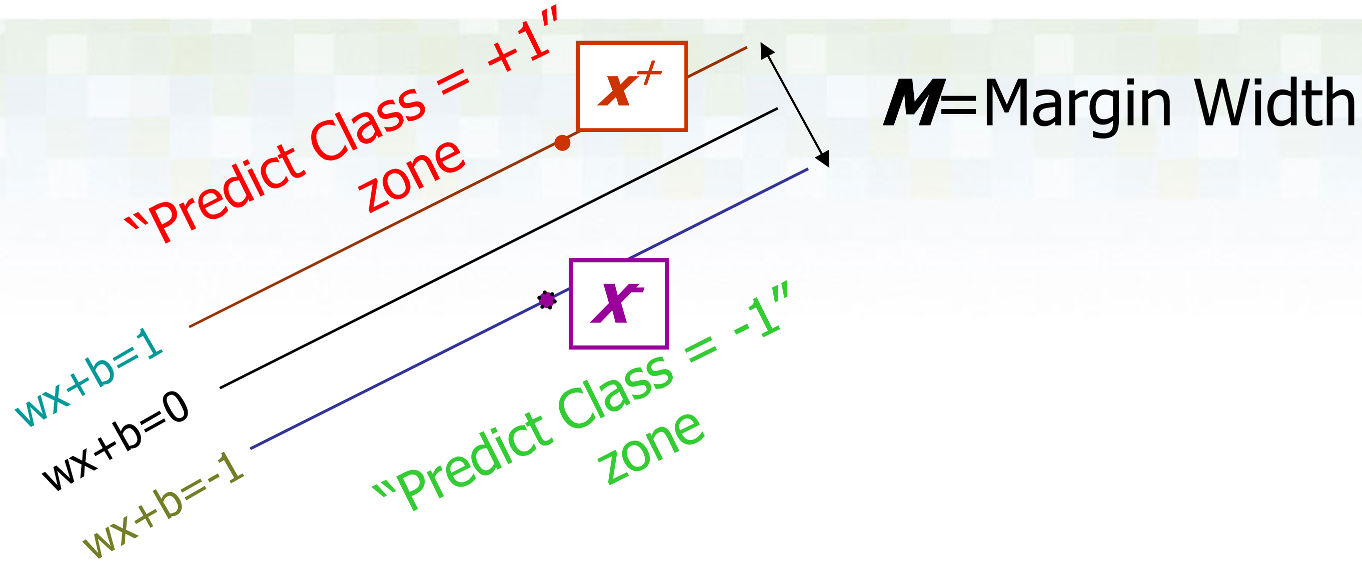
- SVMs maximize the margin around the separating hyperplane.
  - A.k.a. large margin classifiers
- The decision function is fully specified by a subset of training samples, the support vectors.
- Solving SVMs is a *quadratic programming* problem



Maximum Margin: Formalization

- $\mathbf{w}$: decision hyperplane normal vector
- $\mathbf{x}_i$: data point i
- y_i: class of data point i (+1 or -1) Note: Not 1/0
- Classifier is: $f(\mathbf{x}_i) = \text{sign}(\mathbf{w}^T\mathbf{x}_i + b)$
- Functional margin of $\mathbf{x}_i$ is: $y_i\,(\mathbf{w}^T\mathbf{x}_i + b)$
  - But note that we can increase this margin simply by scaling **w**, **b**….
- Functional margin of dataset is twice the minimum functional margin for any point
  - The factor of 2 comes from measuring the whole width of the margin

Linear SVM Mathematically



What we know:

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

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

Linear SVM Mathematically

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

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

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

2) Maximize the Margin $M = \dfrac{2}{|w|}$

same as minimize $\dfrac{1}{2} w^t w$

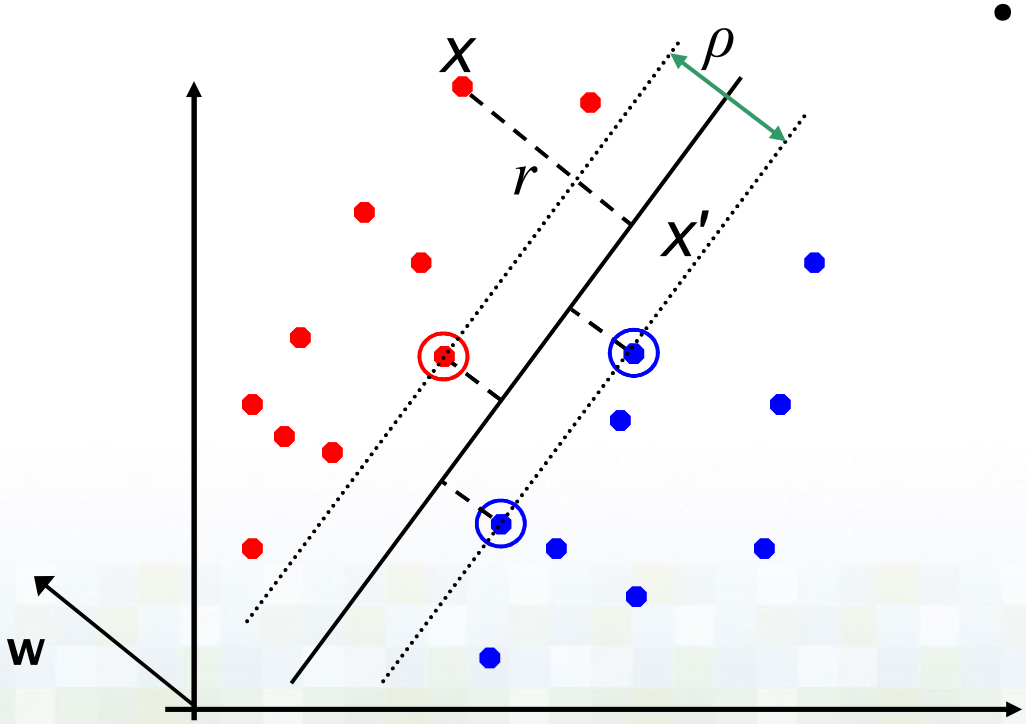
- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\Phi(w) = \dfrac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \qquad \forall i$

Linear SVM (Additional)

- Distance from example to the separator is $r = y\frac{\mathbf{w}^T\mathbf{x}+b}{\|\mathbf{w}\|}$
- Examples closest to the hyperplane are ***support vectors***



- ***Margin*** ρ of the separator is the width of separation between support vectors of classes.

$$\rho = \frac{2}{\|\mathbf{w}\|}$$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = \frac{1}{2}\, \mathbf{w}^T\mathbf{w}$ 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 $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x_i}^T\mathbf{x_j}$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

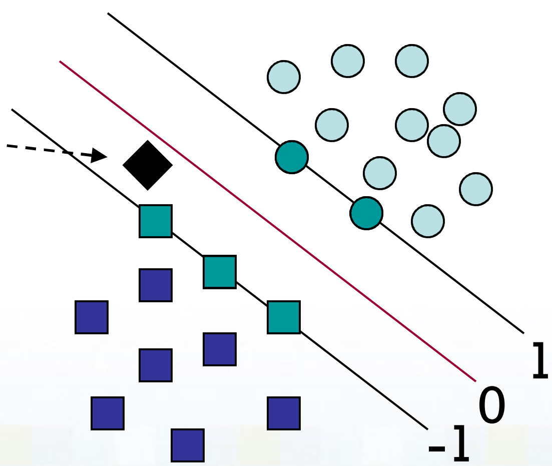
Classification with SVM

- Compute score, decide class based on < or > 0
- Can set confidence threshold ‘t’

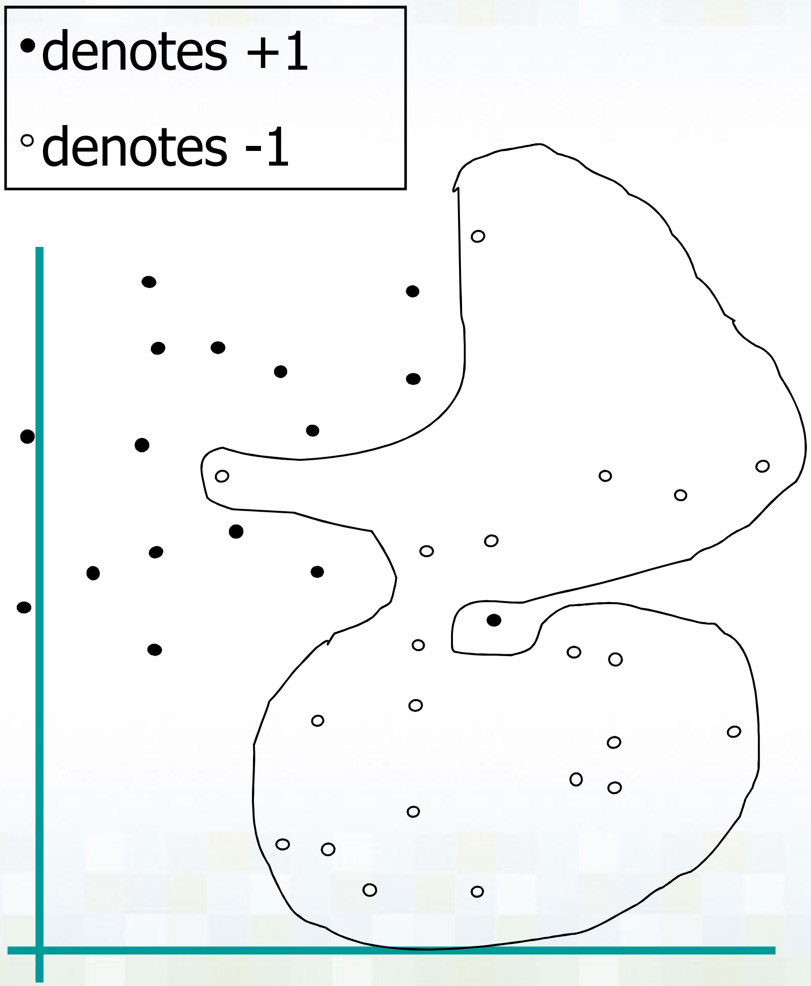
Score > t: yes

Score < t: no

Else: don’t know



Dataset with Noise

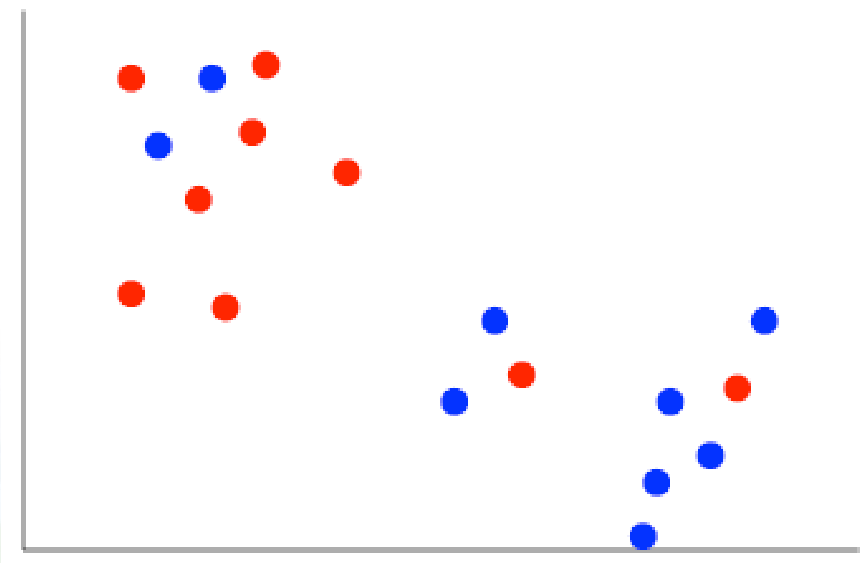


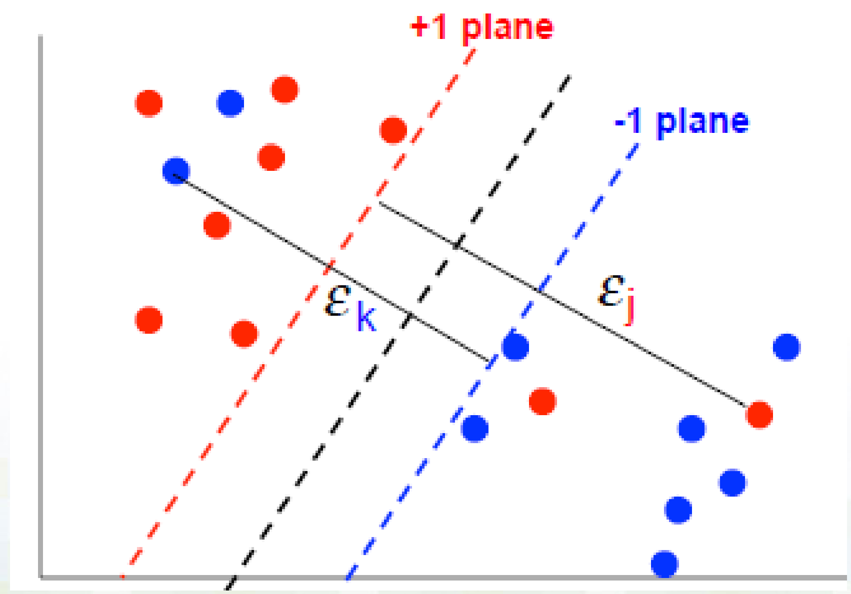
- Hard Margin: So far we require all data points be classified correctly
  - No training error
- What if the training set is noisy?
  - Solution 1: use very powerful kernels

OVERFITTING!

Linearly NON-separable cases??

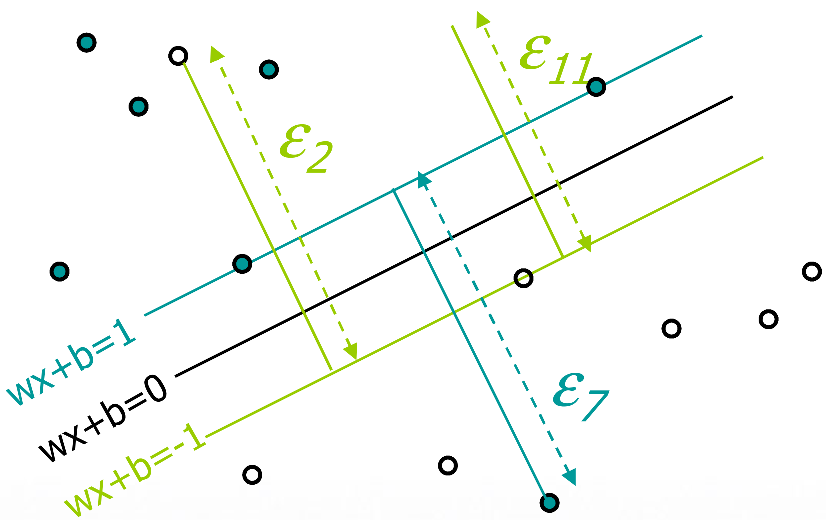
- So far we've assumed that a linear plane can perfectly separate the points
- But, this is not usually the case
  - Noise, outliers, etc…





Soft Margin Classification

Instead of minimizing the number of misclassified points we can minimize the distance between these points and their correct plane



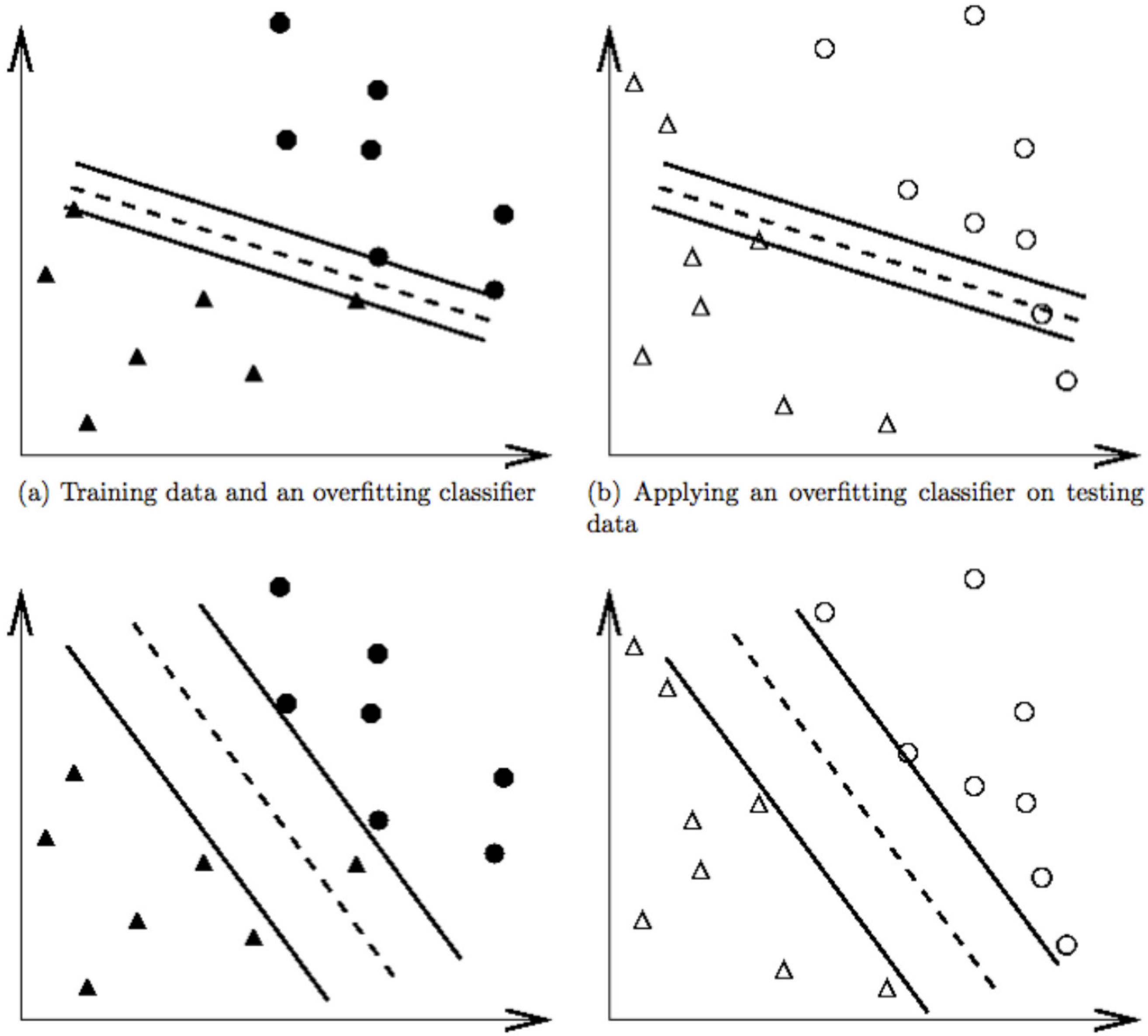
What should our quadratic optimization criterion be?

Minimize

$$\boxed{\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k}$$

- Large C:

- Small C *(better classifier)*



(a) Training data and an overfitting classifier (b) Applying an overfitting classifier on testing data

(c) Training data and a better classifier (d) Applying a better classifier on testing data

Hard Margin vs Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi}(\mathbf{w}) = ½\, \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\, (\mathbf{w^T x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

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

Linear SVMs: Overview

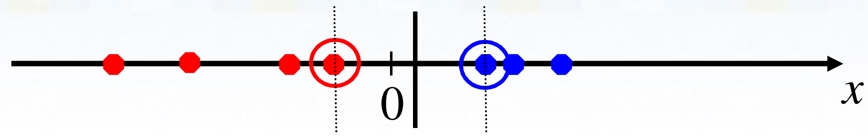
- The classifier is a *separating hyperplane*
- Most "important" training points are support vectors; they define the hyperplane
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

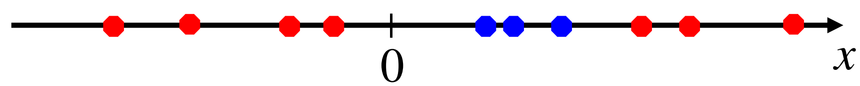
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

Non-linear SVMs

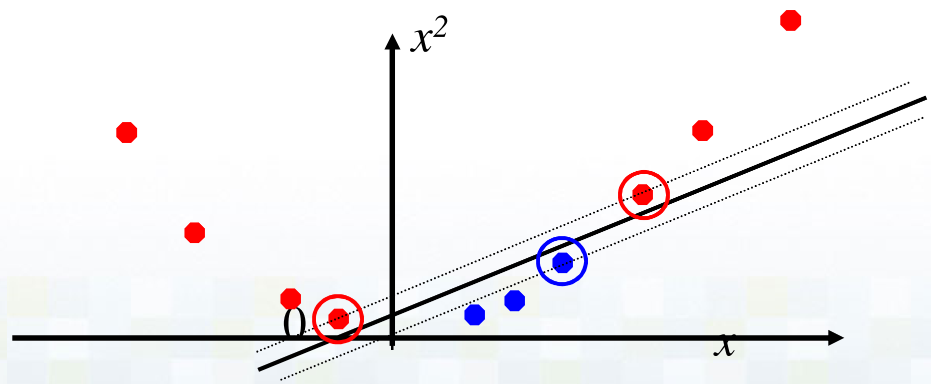
- Datasets that are linearly separable with some noise work out great:



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

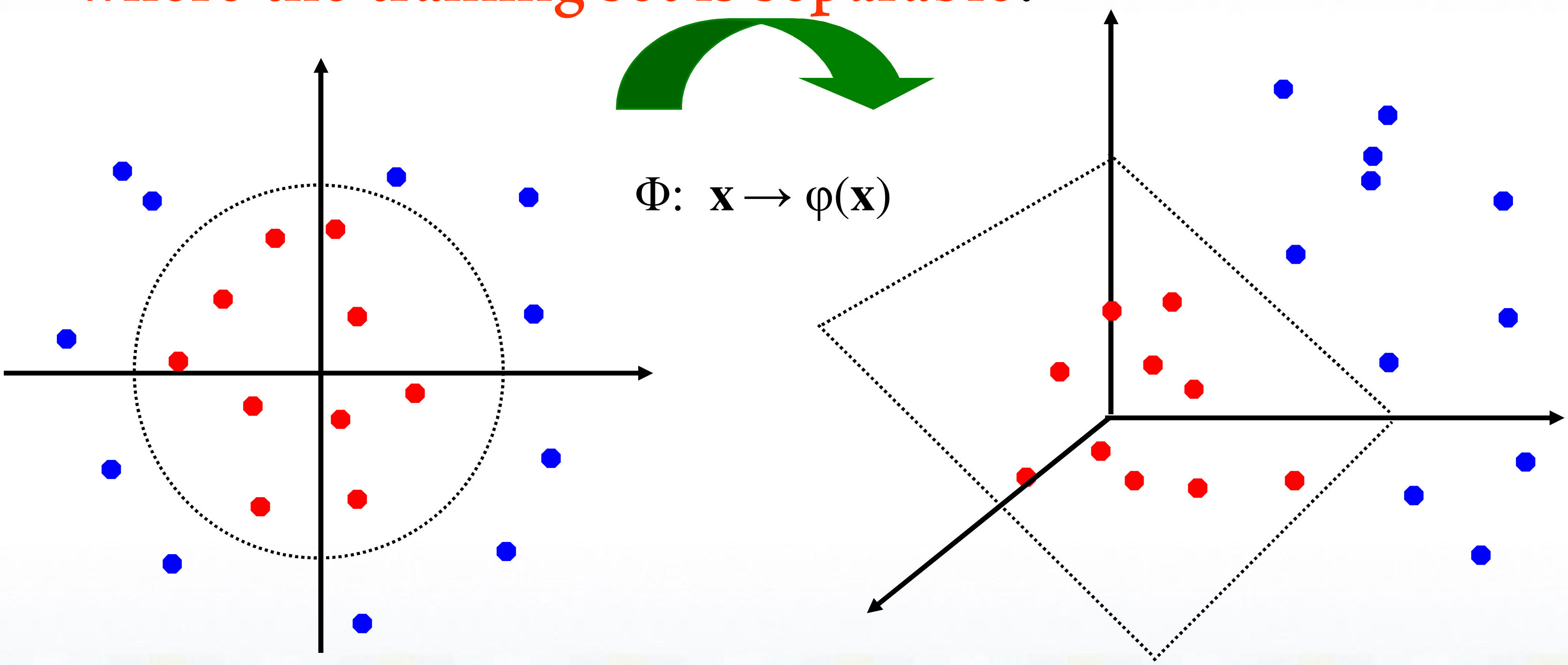


- How about… mapping data to a higher-dimensional space:



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:



The "Kernel Trick"

- The linear classifier relies on dot product between vectors $\boldsymbol{K}(\mathbf{x_i},\mathbf{x_j})=\mathbf{x_i}^{\mathbf{T}}\mathbf{x_j}$
- If every data point is mapped into high-dimensional space via some transformation Φ: $x \rightarrow \varphi(x)$, the dot product becomes:

$$\boldsymbol{K}(\mathbf{x_i},\mathbf{x_j}) = \varphi(\mathbf{x_i})^{\mathbf{T}}\varphi(\mathbf{x_j})$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$

- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j}) = (1+ \mathbf{x_i}^T\mathbf{x_j})^p$

- Gaussian (radial-basis function network):

$$K(\mathbf{x_i},\mathbf{x_j}) = \exp\left(-\frac{\left\|\mathbf{x_i}-\mathbf{x_j}\right\|^2}{2\sigma^2}\right)$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j}) = \tanh(\beta_0\mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathrm{x_i}, \mathrm{x_j})$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$\boxed{f(\mathrm{x}) = \Sigma\alpha_i y_i K(\mathrm{x_i}, \mathrm{x_j}) + b}$$

- **Optimization techniques for finding α_i's remain the same!**

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space

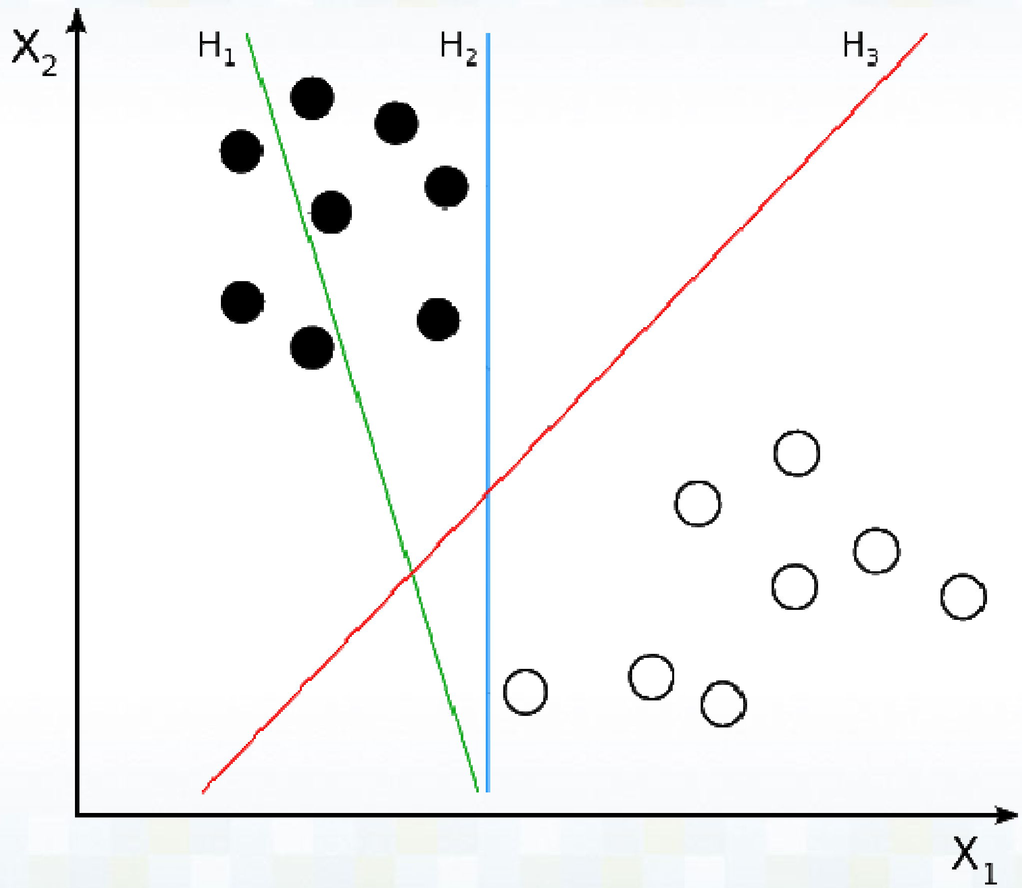
Some Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach

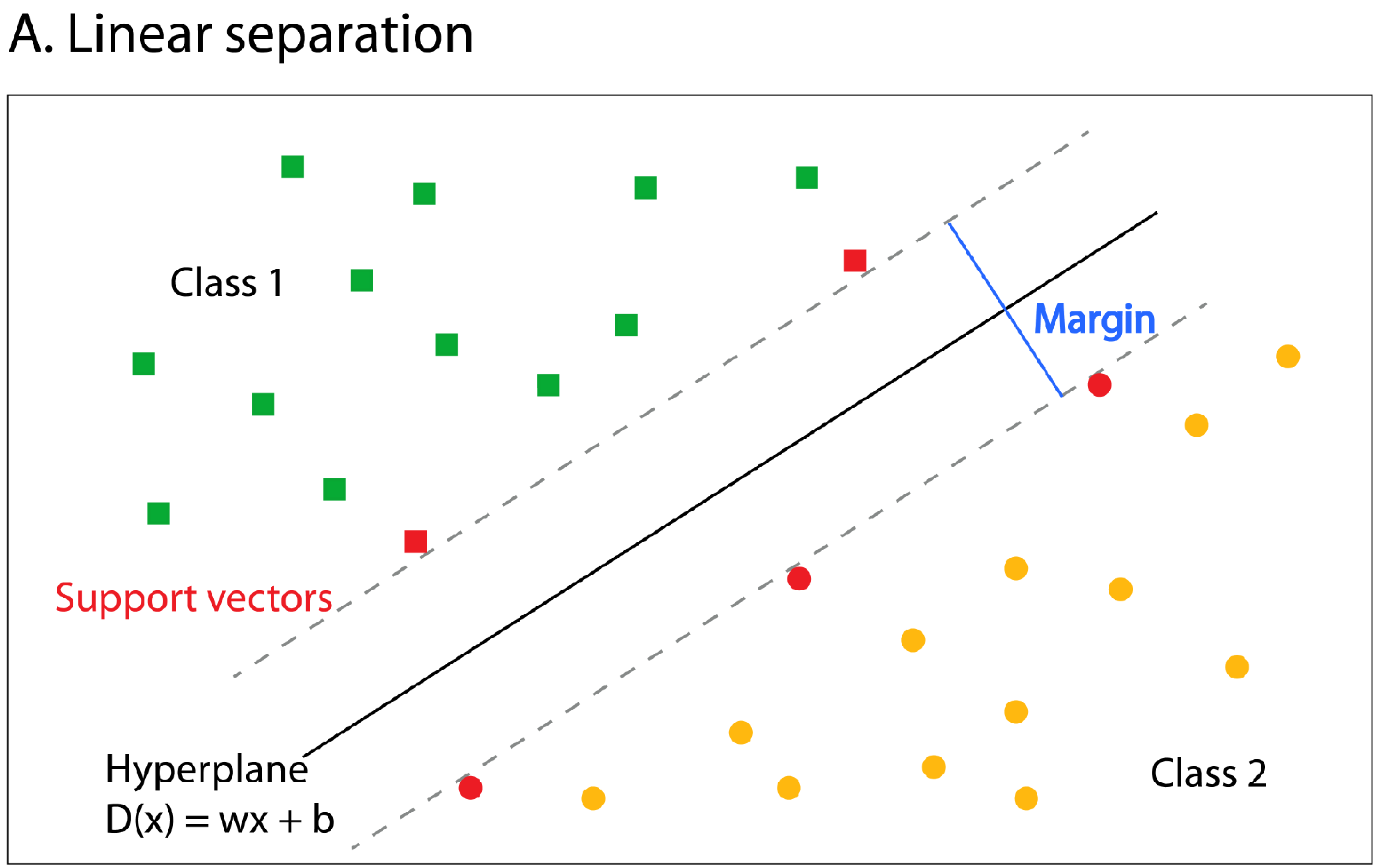
SUMMARY:

Choosing a Separating Hyperplane

Graphic showing how a support vector machine would **choose** a separating hyperplane for two classes of points in 2D. H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximum margin.

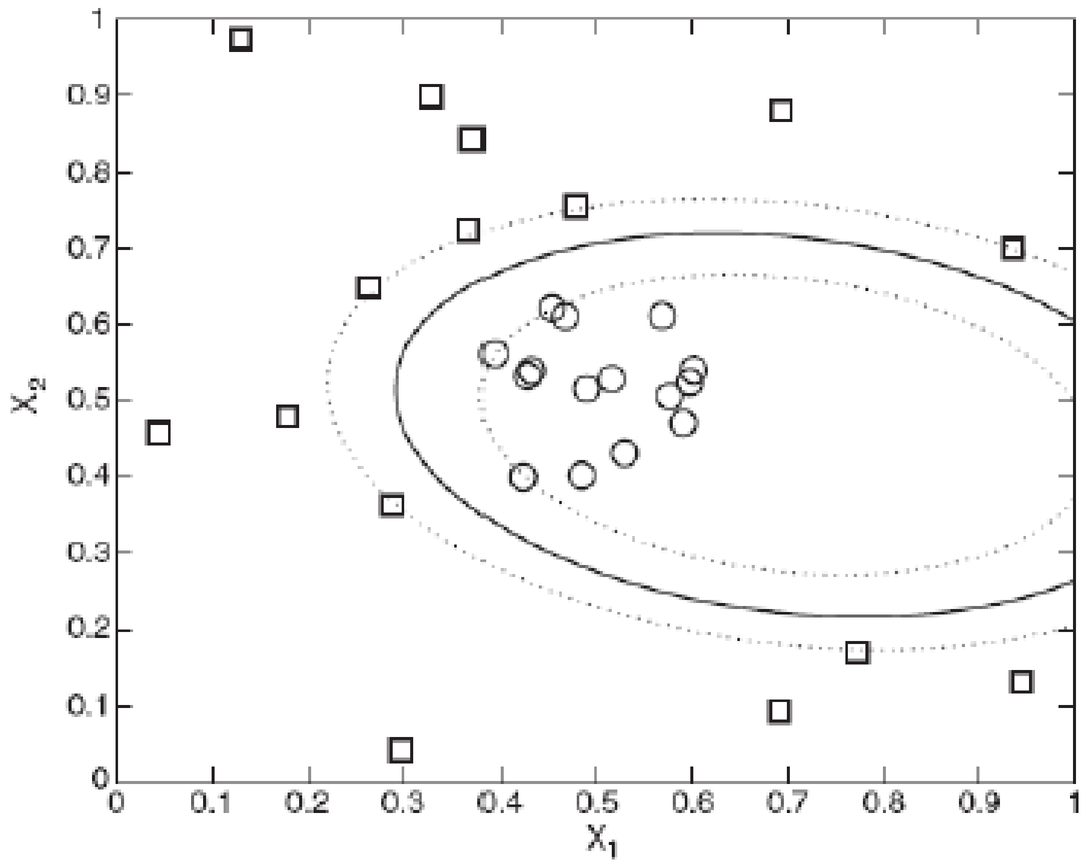


Example SVM (*linear kernel*)



Example SVM (*non-linear kernel*)

- SMV with polynomial kernel

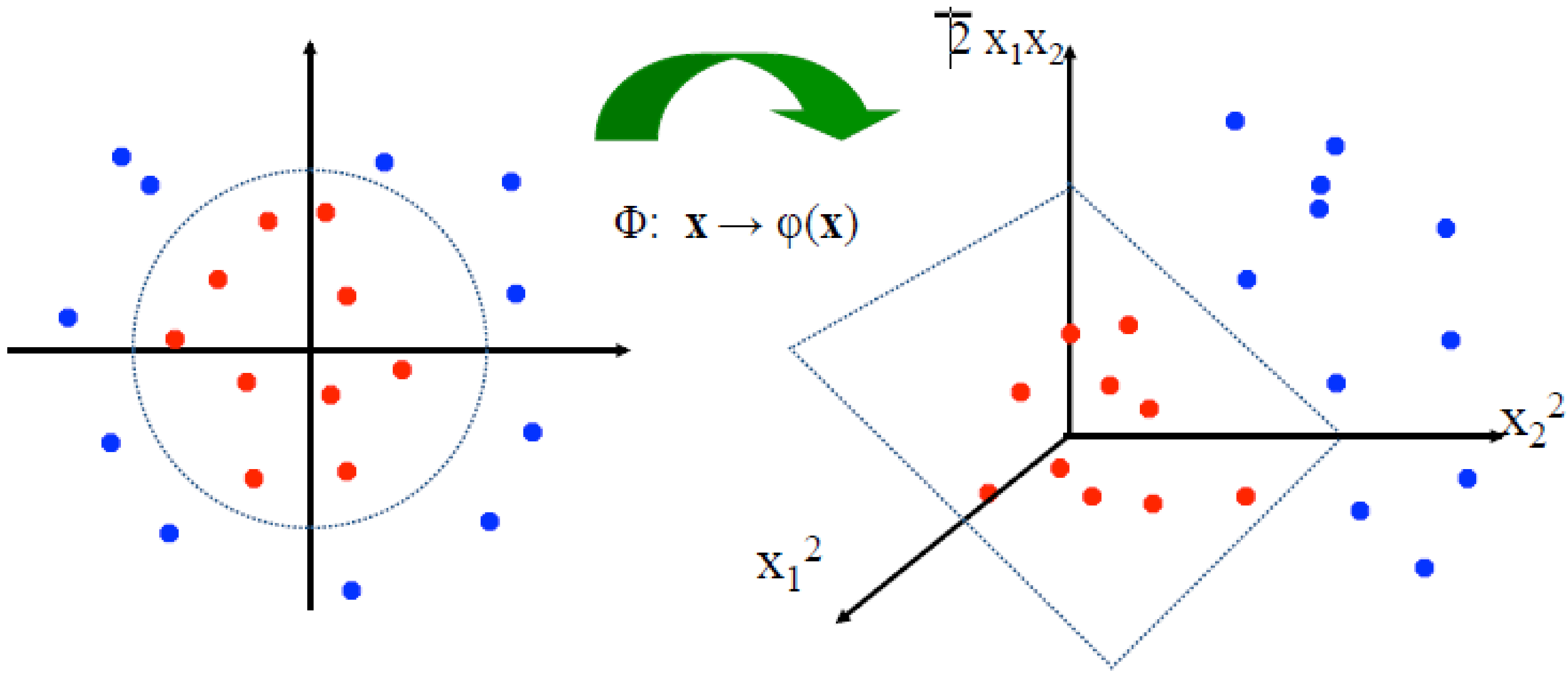


Decision boundary produced by a nonlinear SVM with polynomial kernel.

Example SVM (*non-linear*)

The original input space (x) can be mapped to some higher-dimensional feature space ($\phi(\mathbf{x})$)where the training set is separable:

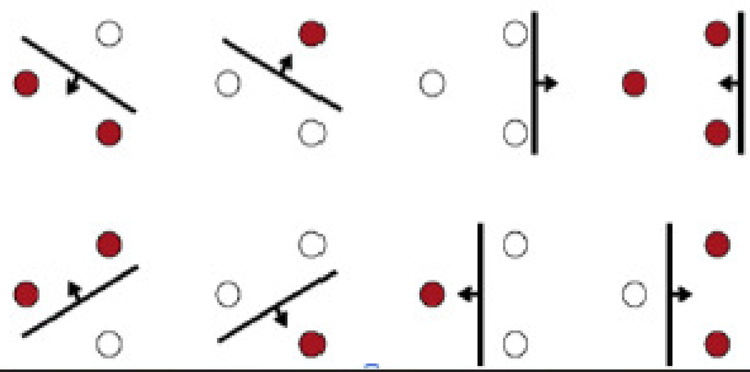
$\mathrm{x}=(x_1,x_2)$ $\varphi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}\, x_1 x_2)$



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm_tutorial.ppt***

Separability

- If data is mapped into sufficiently high dimension, then samples will in general be linearly separable
- N data points are in general separable in a space of N-1 dimensions or more!
- VC dimension of a set of oriented lines in 2-dim (R^2) is 3
  - The VC dimension of the family of oriented separating hyperplanes in R^N is at least N+1



Why do SVMs work?

If we are using large feature spaces (e.g. with kernels) how come we are not overfitting the data?

- Parameters remains the same
- While we have a lot of input values, we only care about the support vectors and these are usually a small group of samples
- The maximization of the margin acts as a sort of regularization term leading to reduced overfitting

Why do SVMs work?

- Vapnik argues that the flexibility of a classifier should not be characterized by the number of params, but by the capacity of a classifier
  - Formalized by the "VC dimension" of a classifier

Overfitting

- Occam's Razor:
  - Simpler system are better than more complex ones.
  - In SVM case: fewer support vectors mean a simpler representation of the hyperplane
  - Example: Understanding a certain cancer if it can be described by one gene
  - is easier than if we have to describe it with 5000

Weakness of SVM

- SVMs are sensitive to noise
  - A relatively small number of mislabeled examples can dramatically decrease the performance

- SVMs only considers two classes

What to do?

  - *THINK ABOUT IT!*

SVM Code Snippet (Python)

http://scikit-learn.org/stable/auto_examples/svm/plot_iris.html

```
...
from sklearn import svm, datasets
# import some data to play with
iris = datasets.load_iris()
# Take the first two features. We could avoid this by using a two-dim dataset
X = iris.data[:, :2]
y = iris.target
# we create an instance of SVM and fit out data. We do not scale our
# data since we want to plot the support vectors
C = 1.0  # SVM regularization parameter
models = (svm.SVC(kernel='linear', C=C),
        svm.LinearSVC(C=C),
        svm.SVC(kernel='rbf', gamma=0.7, C=C),
        svm.SVC(kernel='poly', degree=3, C=C))
models = (clf.fit(X, y) for clf in models)
...
```

Applications of SVMs

- Computer Vision
- Text (and hypertext) Categorization
- Ranking (e.g., Google searches)
- Handwritten Character Recognition
- Image classification
- Time series analysis
- Bioinformatics (e.g. protein classification, cancer classification, …)
- … and much more!

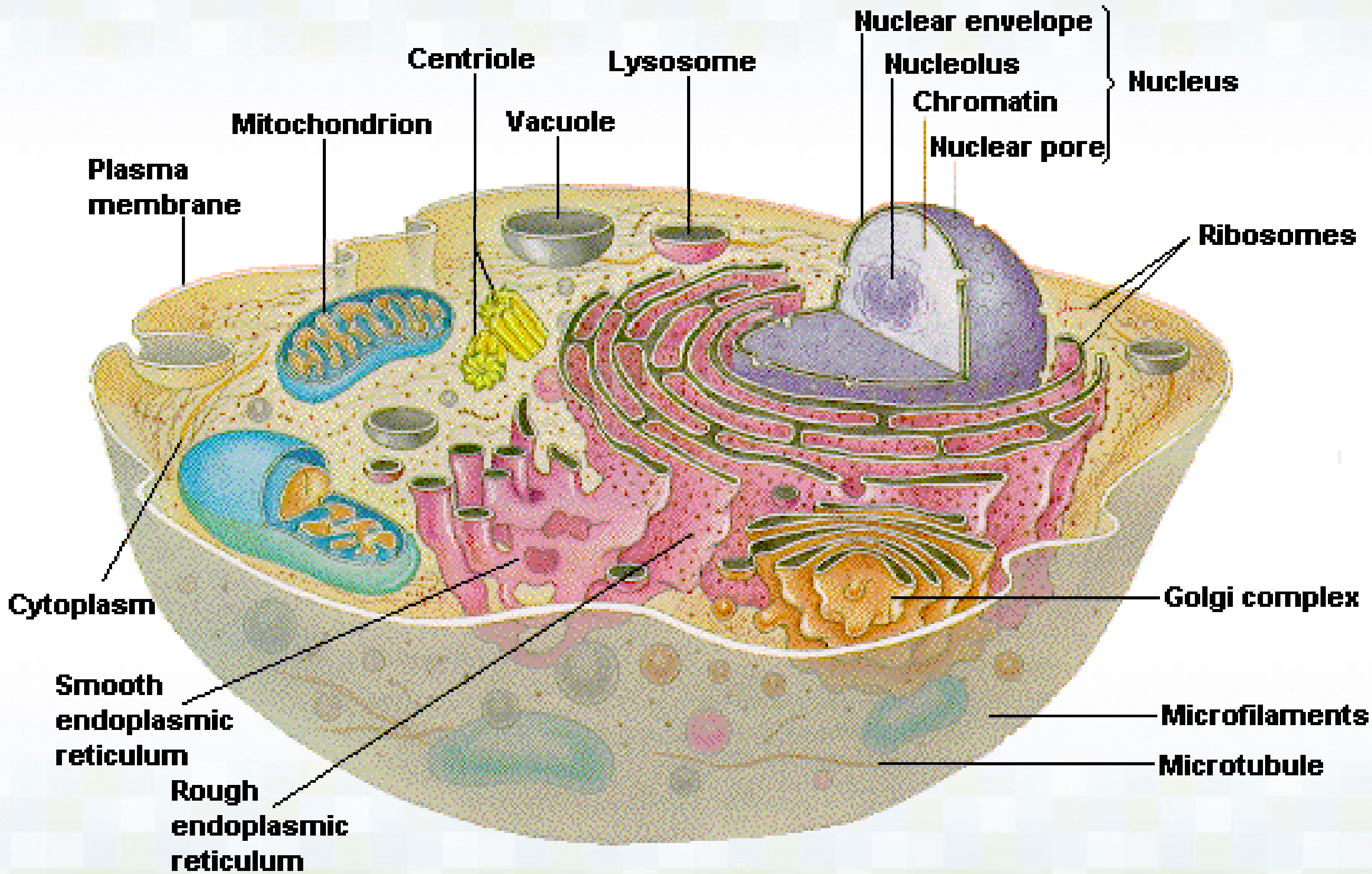
SVM Example

Protein Localization:

- Proteins are synthesized in the cytosol
- Transported into different subcellular locations where they carry out their functions

- Aim: To predict in what location a certain protein will end up!

Subcellular Locations



Method

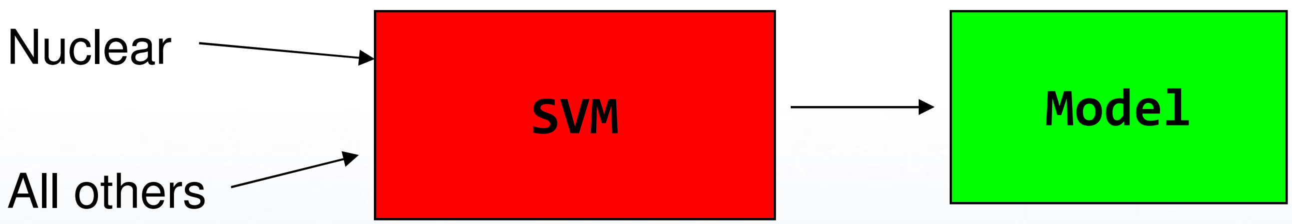
- Hypothesis: The amino acid composition of proteins from different compartments should differ
- Extract proteins with know subcellular location from SWISSPROT
- Calculate the amino acid composition of the proteins
- Try to differentiate between: cytosol, extracellular, mitochondria and nuclear by using SVM

Input Encoding

- E.g.: Prediction of **nuclear proteins**:
- Label the known nuclear proteins as +1 and all others as –1
- The input vector xi represents the amino acid composition

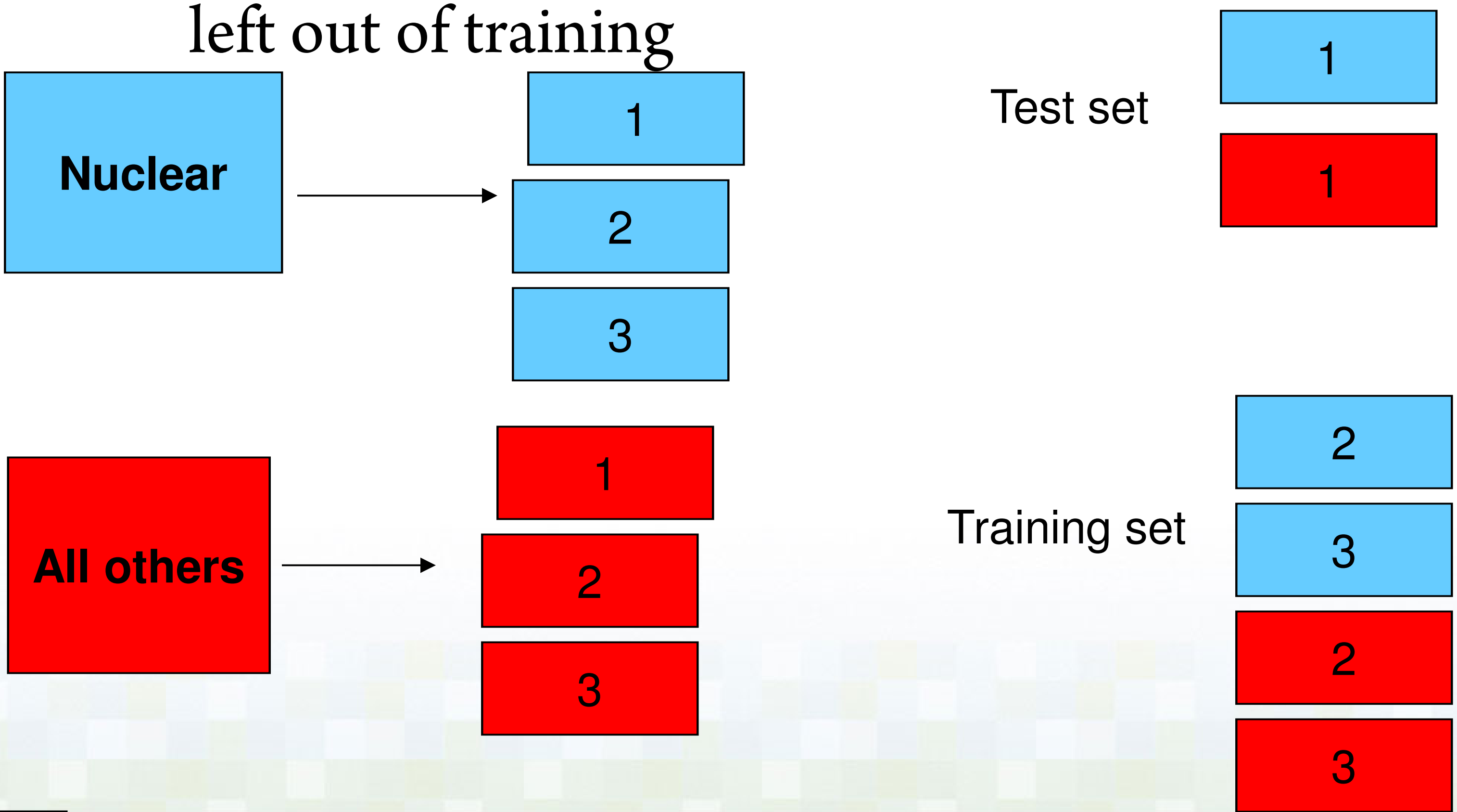
Eg xi =(4.2,6.7,12,….,0.5)

A , C , D,….., Y)

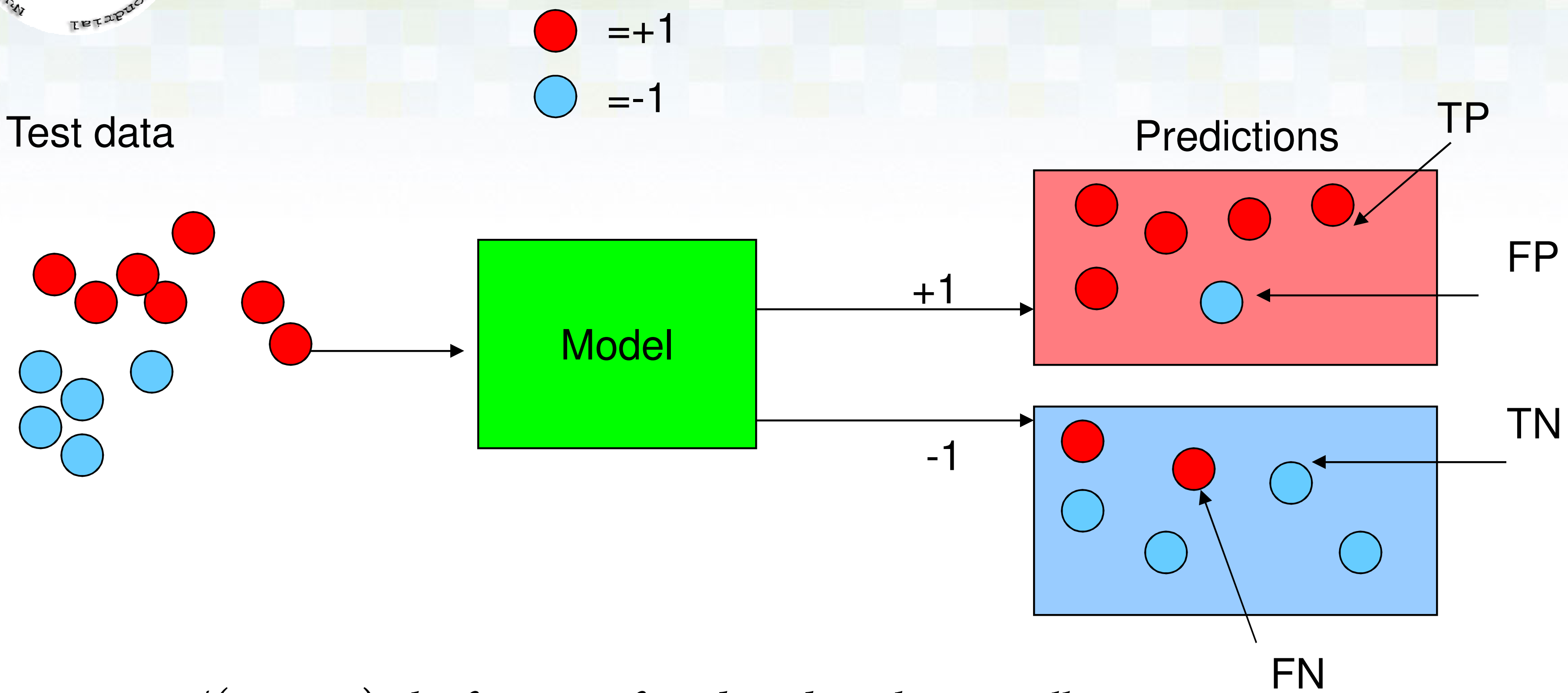


Cross-Validation

- Cross validation:
  - Split the data into n sets, train on n-1 set, test on the set left out of training



Performance Measurements



SP = TP /(TP+FP), the fraction of predicted +1 that actually are +1

SE = TP /(TP+FN), the fraction of the +1 that actually are predicted as +1

In this case: SP=5/(5+1) =0.83

SE = 5/(5+2) = 0.71

A Cautionary Example

Image classification of tanks. Autofire when an enemy tank is spotted.

Input data: Photos of own and enemy tanks

Worked really good with the training set used

IN REALITY IT FAILED COMPLETELY!

Reason: All enemy tank photos taken in the morning. All own tanks in dawn.

The classifier could recognize dusk from dawn!!!!

Resources

Some resources for SVM:

- A list of SVM implementation can be found at
  - http://www.kernel-machines.org/software.html
- Some implementation (such as LIBSVM) can handle multi-class classification
- SVMLight is among one of the earliest implementation of SVM

Resources / LIBSVM

- www.csie.ntu.edu.tw/~cjlin/libsvm/
- Developed by Chih-Jen Lin et al.
- Tools for Support Vector classification
- Also support multi-class classification
- C++/Java/Python/Matlab/Perl wrappers
- Linux/UNIX/Windows
- SMO implementation is fast

Resources / LIBSVM

- Training.dat

```
+1 1:0.708333 2:1 3:1 4:-0.320755
-1 1:0.583333 2:-1 4:-0.603774 5:1
+1 1:0.166667 2:1 3:-0.333333 4:-0.433962
-1 1:0.458333 2:1 3:1 4:-0.358491 5:0.374429
…
```

- Testing.dat

Resources / LIBSVM

- (1) Categorical feature
  - Recommend using m numbers to represent an m-category attribute
  - Only one of the m numbers is **one**, and others are zero
  - E.g. three-category attribute such as {red, green, blue} can be represented as (0,0,1), (0,1,0), and (1,0,0)
- (2) Scaling before applying SVM is important
  - Avoid some attributes dominating others
  - E.g. linear scaling, each attribute to the range [0,1] or [-1,1]
- (3) Address missing values

Overview Procedure

1. Train / Test
2. K-fold cross validation
3. K-CV on train to choose hyperparameter; then test

Material adapted (with thanks!) from Pr.Ricardo Osuna slides & Dr.Qi slides

Additional references listed below.

References:

~Support Vector Machines ~ A.W.Moore tutorials ~ cs.cmu.edu/~awm

~ A Training Algorithm for Optimal Margin Classifiers ~ B.E. Boser et al. ~ Computational Learning Theory, 1992

~ Comparison of classifier methods: a case study in handwritten digit recognition ~ L.Bottou et al. ~ IAPR International Conference on Pattern Recognition, 1994

~ The Nature of Statistical Learning Theory ~ V.Vapnik, 1999

~A Practical Guide to Support Vector Classification ~ Chih-Wei Hsu et al. ~ hkp://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

~An Introduction to Support Vector Machines ~ H.Kautz, 2005

~Support Vector Machine ~ M.Tan ~ UBC

~Statistical Learning Theory ~ V.Vapnik, 1998