Classification

Michael I. Jordan
University of California, Berkeley

Classification

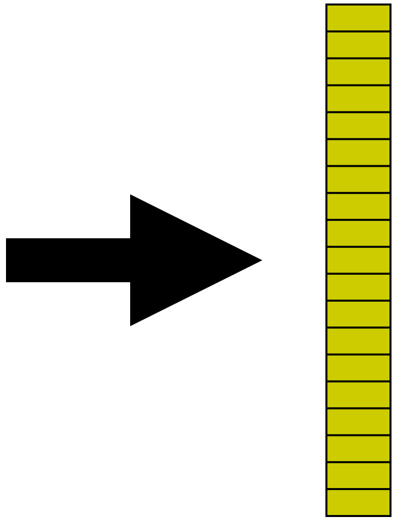
- In classification problems, each entity in some domain can be placed in one of a discrete set of categories: yes/no, friend/foe, good/bad/indifferent, blue/red/green, etc.
- Given a training set of labeled entities, develop a rule for assigning labels to entities in a test set
- Many variations on this theme:
  - binary classification
  - multi-category classification
  - non-exclusive categories
  - ranking
- Many criteria to assess rules and their predictions
  - overall errors
  - costs associated with different kinds of errors
  - operating points

Representation of Objects

- Each object to be classified is represented as a pair (x, y):
  - where x is a description of the object (see examples of data types in the following slides)
  - where y is a label (assumed binary for now)
- Success or failure of a machine learning classifier often depends on choosing good descriptions of objects
  - the choice of description can also be viewed as a learning problem, and indeed we'll discuss automated procedures for choosing descriptions in a later lecture
  - but good human intuitions are often needed here

Data Types

- Vectorial data:
  - physical attributes
  - behavioral attributes
  - context
  - history
  - etc



- We'll assume for now that such vectors are explicitly represented in a table, but later (cf. kernel methods) we'll relax that asumption

Data Types

- text and hypertext

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
		<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
		<title>Welcome to FairmontNET</title>
</head>
<STYLE type="text/css">
.stdtext {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 11px; color: #1F3D4E;}
.stdtext_wh {font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 11px; color: WHITE;}
</STYLE>

<body leftmargin="0" topmargin="0" marginwidth="0" marginheight="0" bgcolor="BLACK">
<TABLE cellpadding="0" cellspacing="0" width="100%" border="0">
  <TR>
    <TD width=50% background="/TFN/en/CDA/Images/common/labels/decorative_2px_blk.gif"> </TD>
    <TD><img src="/TFN/en/CDA/Images/common/labels/decorative.gif"></td>
    <TD width=50% background="/TFN/en/CDA/Images/common/labels/decorative_2px_blk.gif"> </TD>
  </TR>
</TABLE>
<tr>
<td align="right" valign="middle"><IMG src="/TFN/en/CDA/Images/common/labels/centrino_logo_blk.gif"></td>
</tr>
</body>
</html>
```

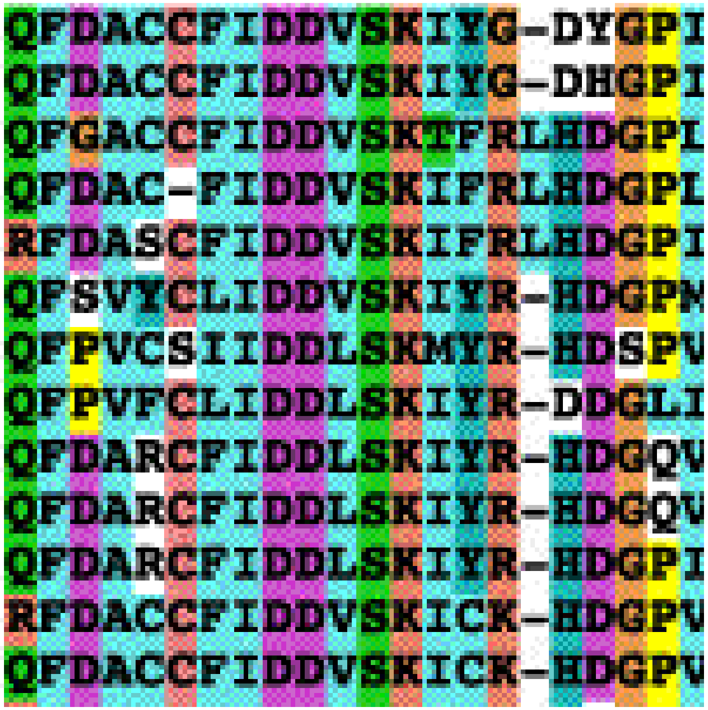
Data Types

- email

**Return-path <bmiller@eecs.berkeley.edu>Received from relay2.EECS.Berkeley.EDU
(relay2.EECS.Berkeley.EDU [169.229.60.28]) by imap4.CS.Berkeley.EDU (iPlanet Messaging Server
5.2 HotFix 1.16 (built May 14 2003)) with ESMTP id <0HZ000F506JV5S@imap4.CS.Berkeley.EDU>;
Tue, 08 Jun 2004 11:40:43 -0700 (PDT)Received from relay3.EECS.Berkeley.EDU (localhost
[127.0.0.1]) by relay2.EECS.Berkeley.EDU (8.12.10/8.9.3) with ESMTP id i58Ieg3N000927; Tue, 08
Jun 2004 11:40:43 -0700 (PDT)Received from redbirds (dhcp-168-35.EECS.Berkeley.EDU
[128.32.168.35]) by relay3.EECS.Berkeley.EDU (8.12.10/8.9.3) with ESMTP id i58IegFp007613;
Tue, 08 Jun 2004 11:40:42 -0700 (PDT)Date Tue, 08 Jun 2004 11:40:42 -0700From Robert Miller
<bmiller@eecs.berkeley.edu>Subject RE: SLT headcount = 25In-reply-
to <6.1.1.1.0.20040607101523.02623298@imap.eecs.Berkeley.edu>To 'Randy Katz'
<randy@eecs.berkeley.edu>Cc "'Glenda J. Smith'" <glendajs@eecs.berkeley.edu>, 'Gert Lanckriet'
<gert@eecs.berkeley.edu>Message-
id <200406081840.i58IegFp007613@relay3.EECS.Berkeley.EDU>MIME-version 1.0X-
MIMEOLE Produced By Microsoft MimeOLE V6.00.2800.1409X-Mailer Microsoft Office Outlook, Build
11.0.5510Content-type multipart/alternative; boundary="----
=_NextPart_000_0033_01C44D4D.6DD93AF0"Thread-
index AcRMtQRp+R26lVFaRiuz4BfImikTRAA0wf3Qthe headcount is now 32. --------------------------
------------- Robert Miller, Administrative Specialist University of California, Berkeley Electronics
Research Lab 634 Soda Hall #1776 Berkeley, CA 94720-1776 Phone: 510-642-6037 fax: 510-
643-1289**

Data Types

- protein sequences



Data Types

- sequences of Unix system calls

Process Management

pid = fork()	Create a child process
s=waitpid(pid, &status, opts)	Wait for a child to terminate
s=execve(name, argv, envp)	Replace a process' core image
exit(status)	Terminate execution
s=sigaction(sig, &act, &oact)	Specify action to take for a signal
s=kill(pid, sig)	Send a signal to process
residual=alarm(seconds)	Schedule a SIGALRM signal later
pause()	Suspend the caller until next signal

Memory Management

size=brk(addr)	Set the size of data segment

Input/Output Management

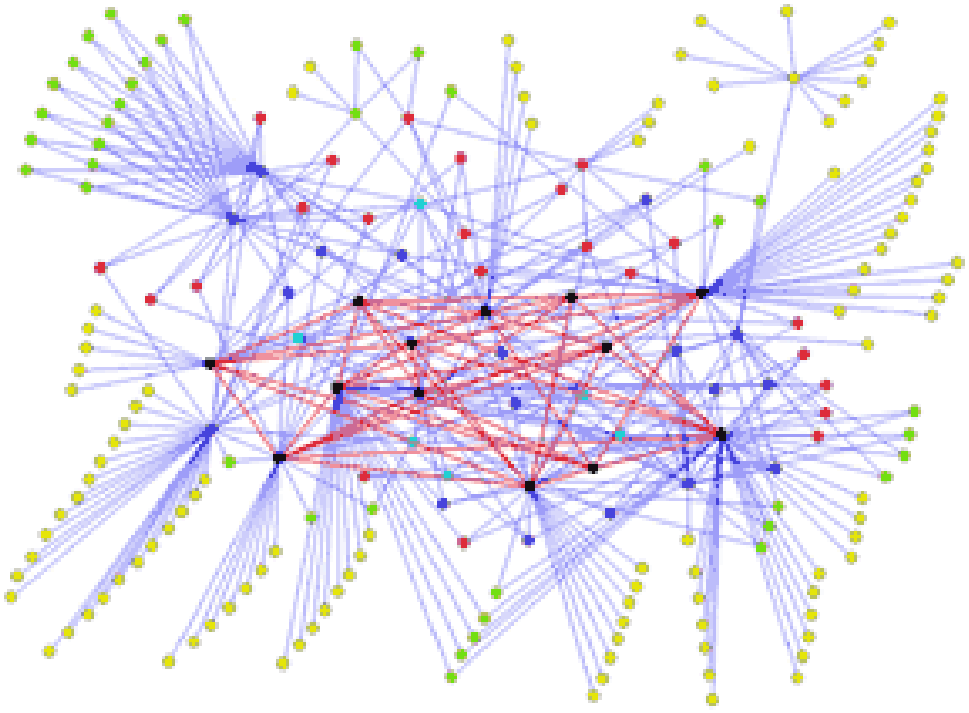
s=cfsetospeed(&termios, speed)	Set the output speed
s=cfsetispeed(&termios, speed)	Set the input speed
s=cfgetospeed(&termios, speed)	Get the output speed
s=cfgetispeed(&termios, speed)	Get the input speed
s=tcsetattr(fd, opt, &termios)	Set terminal attributes
s=tcgetattr(fd, &termios)	Get terminal attributes

Files and Directories Management

fd=create(name, mode)	Create a new file
fd=open(name, how)	Open a file for reading or writing
s=close(fd)	Close an open file
n=read(fd, buffer, nbytes)	Read data from file into a buffer
n=write(fd, buffer, nbytes)	Write data from buffer to file
pos=lseek(fd, offset, whence)	Move the file pointer somewhere
s=start(name, &buf)	Read and return info. about file
s=mkdir(name, mode)	Create a new directory
s=rmdir(name)	Delete an empty directory
s=link(name1, name2)	Create a new directory entry for an old file
s=unlink(name)	Remove a directory entry
s=chdir(dirname)	Change the working directory
s=chmod(name, mode)	Change a file's protection bits

Data Types

- network layout: graph



Data Types

- images

Example: Spam Filter

- Input: email
- Output: spam/ham
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “ham”
  - Note: someone has to hand label all this data
  - Want to learn to predict labels of new, future emails
- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

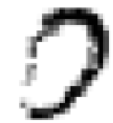
> Dear Sir.
>
> First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

> TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.
>
> 99 MILLION EMAIL ADDRESSES
> FOR ONLY $99

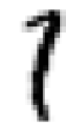
> Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

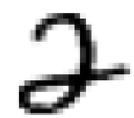
- Input: images / pixel grids
- Output: a digit 0-9
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data
  - Want to learn to predict labels of new, future digit images
- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …
- Current state-of-the-art: Human-level performance



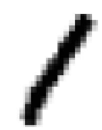
0



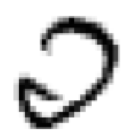
1



2



1



??

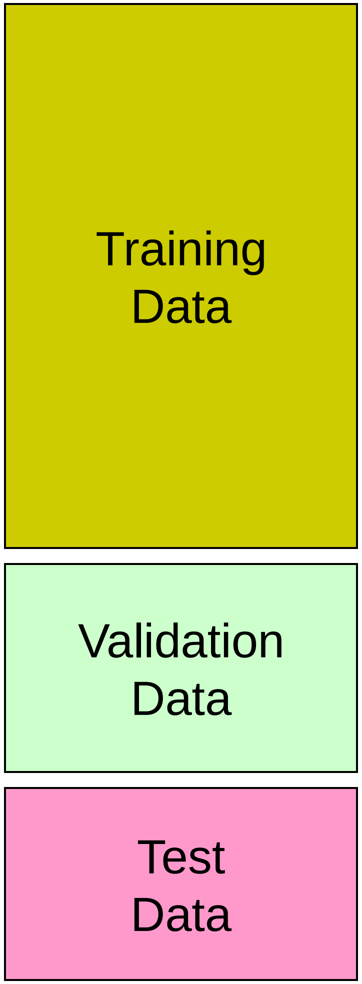
Other Examples of Real-World Classification Tasks

- Fraud detection (input: account activity, classes: fraud / no fraud)
- Web page spam detection (input: HTML/rendered page, classes: spam / ham)
- Speech recognition and speaker recognition (input: waveform, classes: phonemes or words)
- Medical diagnosis (input: symptoms, classes: diseases)
- Automatic essay grader (input: document, classes: grades)
- Customer service email routing and foldering
- Link prediction in social networks
- Catalytic activity in drug design
- … many many more

- Classification is an important commercial technology

Training and Validation

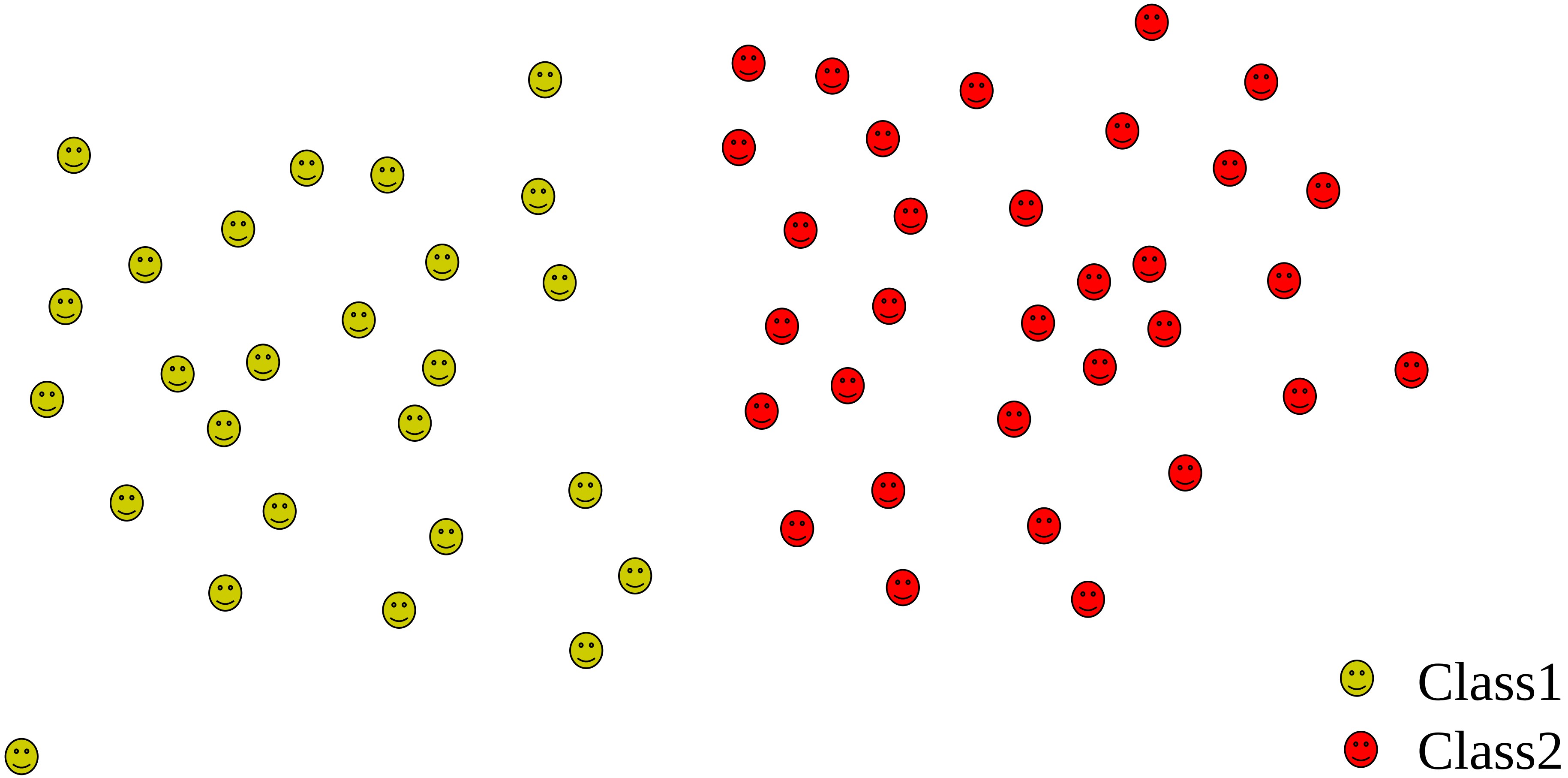
- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Validation set
  - Test set
- Training
  - Estimate parameters on training set
  - Tune hyperparameters on validation set
  - Report results on test set
  - Anything short of this yields over-optimistic claims
- Evaluation
  - Many different metrics
  - Ideally, the criteria used to train the classifier should be closely related to those used to evaluate the classifier
- Statistical issues
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - Error bars: want realistic (conservative) estimates of accuracy



Some State-of-the-art Classifiers

- Support vector machine
- Random forests
- Kernelized logistic regression
- Kernelized discriminant analysis
- Kernelized perceptron
- Bayesian classifiers
- Boosting and other ensemble methods
- (Nearest neighbor)

Intuitive Picture of the Problem



Some Issues

- There may be a simple separator (e.g., a straight line in 2D or a hyperplane in general) or there may not
- There may be “noise” of various kinds
- There may be “overlap”
- One should not be deceived by one’s low-dimensional geometrical intuition
- Some classifiers explicitly represent separators (e.g., straight lines), while for other classifiers the separation is done implicitly
- Some classifiers just make a decision as to which class an object is in; others estimate class probabilities

Methods

I) Instance-based methods:
   1) Nearest neighbor
II) Probabilistic models:
   1) Naïve Bayes
   2) Logistic Regression
III) Linear Models:
   1) Perceptron
   2) Support Vector Machine
IV) Decision Models:
   1) Decision Trees
   2) Boosted Decision Trees
   3) Random Forest

Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

IV) Decision Models:

1) Decision Trees

2) Boosted Decision Trees

3) Random Forest

$$= \frac{P(\vec{w} \mid c)\, P(c)}{\sum_{c'} P(\vec{w} \mid c')\, P(c')}$$

• note: $\theta_{w|c} = 0$ if word w was not observed in training $\Rightarrow$ overfitting $\Rightarrow$ need smoothing!

Naive Bayes – Summary

pros:
- very simple!
- can scale easily to millions of training examples; just need counts!
- do surprisingly well in practice [in text application for example]

cons:
- bogus independence assumption
- can't include complex features
- not best prediction accuracy

Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

IV) Decision Models:

1) Decision Trees

2) Boosted Decision Trees

3) Random Forest

2) Logistic Regression

* assume now $\vec{x}$ is continuous : $\vec{x} = (x_1, \dots, x_d)$

a generative model using NB assumption could be :

let $N(x \mid \mu, \sigma) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\}$

assume variance doesn't depend on C

$$p_\theta(\vec{x} \mid c) = \prod_{i=1}^{d} N(x_i \mid \mu_{i|c}, \sigma_i) \quad [NB]$$

$$p_\theta(c) = \pi_c \quad [\text{e.g. Bernoulli if } c \in \{0,1\}]$$

and parameter $\theta = (\underbrace{\mu_{i|c}}_{\text{mean}}, \underbrace{\sigma_{i|c}}_{\text{variance}}, \pi_c)$

for $i = 1, \dots, d$

$c = 0$ to # classes -1

consider decision boundary: [in binary case]

$$\frac{P(c=1 \mid \vec{x})}{P(c=0 \mid \vec{x})} = \frac{P(c=1, \vec{x}) \,/\, \cancel{p(\vec{x})}}{P(c=0, \vec{x}) \,/\, \cancel{p(\vec{x})}}$$

$$= \frac{\pi_1 \prod_{i=1}^{d} \cancel{\frac{1}{\sqrt{2\pi\sigma_i^2}}} \exp\left(-\frac{(x_i-\mu_{i,1})^2}{2\sigma_i^2}\right)}{\pi_0 \prod_{i=1}^{d} \cancel{\frac{1}{\sqrt{2\pi\sigma_i^2}}} \exp\left(-\frac{(x_i-\mu_{i,0})^2}{2\sigma_i^2}\right)}$$

square terms cancel!
[because equal variance]

$$= \frac{\pi_1}{\pi_0} \exp\left(\sum_{i=1}^{d} \left(\frac{\mu_{i,1}-\mu_{i,0}}{\sigma_i^2}\right) x_i - \frac{(\mu_{i,0}^2 + \mu_{i,1}^2)}{2\sigma_i^2} \right)$$

$$\Rightarrow \ln \frac{P(c=1 \mid \vec{x})}{P(c=0 \mid \vec{x})} = \sum_{i=1}^{d} w_i' \cdot x_i + w_0'$$

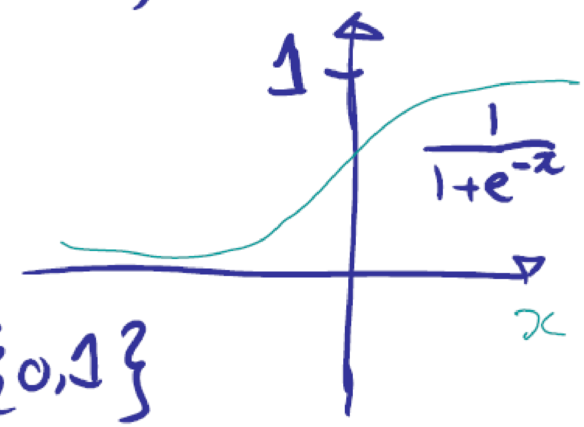
parameters

linear decision boundary!

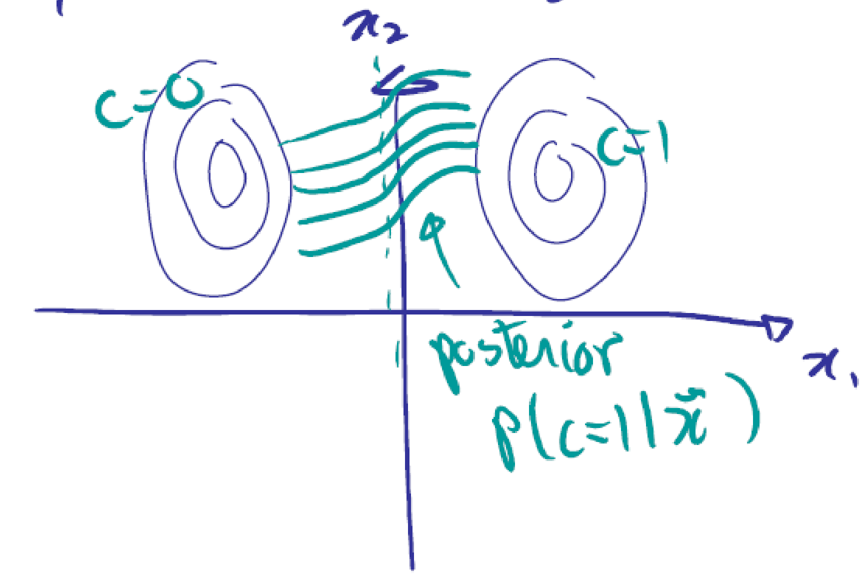
can similarly write:

$$P(c=1 \mid \vec{x}) = \frac{1}{1+\exp(\vec{w}\cdot\vec{x}+w_0)}$$

logistic function



posterior for Gaussian with equal covariance:



$y_i \in \{0,1\}$

$$h(x) = \begin{cases} 1 & \text{if } \vec{w}\cdot\vec{x}+w_0 \geq 0 \\ 0 & \text{o.w.} \end{cases}$$

learning parameters? given $(\vec{x}_i, y_i)$

- could learn $\mu_{i|y}, \sigma_i, \pi_y$ by ML: $\max_\theta P_\theta(\vec{x}_i, y_i)$
- why not learn $\vec{w}$ directly?

$\Rightarrow$ maximum conditional likelihood

$$\max_w \prod_{i=1}^{n} P_w(y_i \mid \vec{x}_i)$$

- advantage: more robust to model assumptions

[e.g. think of exponential family]

Maximum Conditional likelihood

- unfortunately, no closed formula for $\vec{w}^*$

→ use gradient ascent or your favorite optimization alg. on conditional likelihood

- log cond. likelihood

$$l(w) = \ln\left(\prod_{i=1}^{n} p_w(y_i \mid \vec{x})\right)$$

$$= \sum_{i=1}^{n} y_i\,(w_0 + \vec{w}\cdot\vec{x}) - \sum_{i=1}^{n} \ln\left(1+\exp(w_0+\vec{w}\cdot\vec{x})\right)$$

$$\Rightarrow \nabla l = \sum_{i=1}^{n} \left(y_i - p_w\right.$$

Stochastic gradient ascent:

$$\vec{w}^{(t+1)} \leftarrow \vec{w}^{(t)} + \underbrace{\eta}_{\text{step size}} \left(y_i - P_{\vec{w}^t}(\right.$$

iterate over data until convergence criterion

Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

IV) Decision Models:

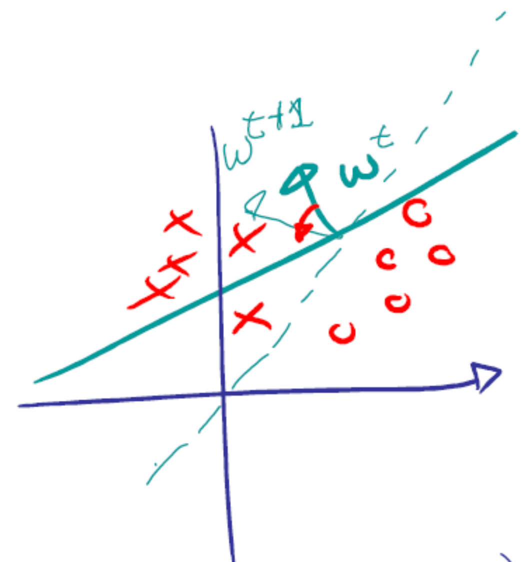
1) Decision Trees

2) Boosted Decision Trees

3) Random Forest

III) Linear Models:

1) Perceptron



$$h(x) = \begin{cases} 1 & \text{if } \vec{w}\cdot\vec{x} + b \geq 0 \\ 0 & \text{o.w.} \end{cases}$$

(compare with log. regression)

mistake driven: $\vec{w}^{t+1} = \vec{w}^{t} + \eta \big(\underbrace{y_i - h(x_i)}_{\neq 0 \text{ only if mistake}} \big) \vec{x}_i$

2) Average perceptron:

$$\vec{w}_{final} = \frac{1}{T} \sum_{t=1}^{T} \vec{w}^{(t)}$$

$$\vec{w}_{final} \cdot \vec{x} = \frac{1}{T} \sum_{t=1}^{T} \underbrace{\vec{w}^{(t)} \cdot \vec{x}}_{\text{voting scheme?}}$$

averaged perceptron: practical issues

- work quite well in practice
- but result sensitive to order of data points

  $\Rightarrow$ randomize order

- choose stopping criterion by validation set ...

Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

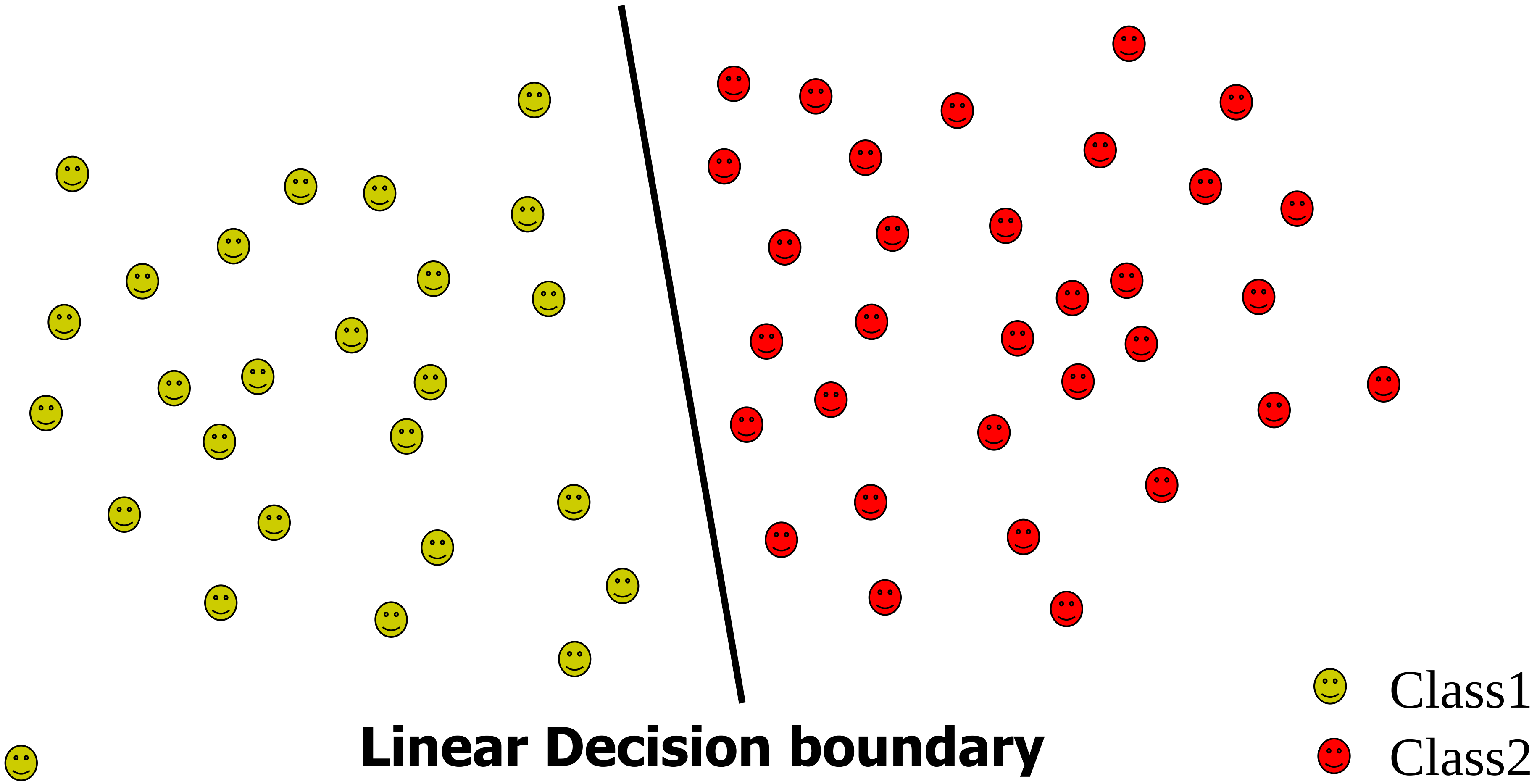
IV) Decision Models:

1) Decision Trees

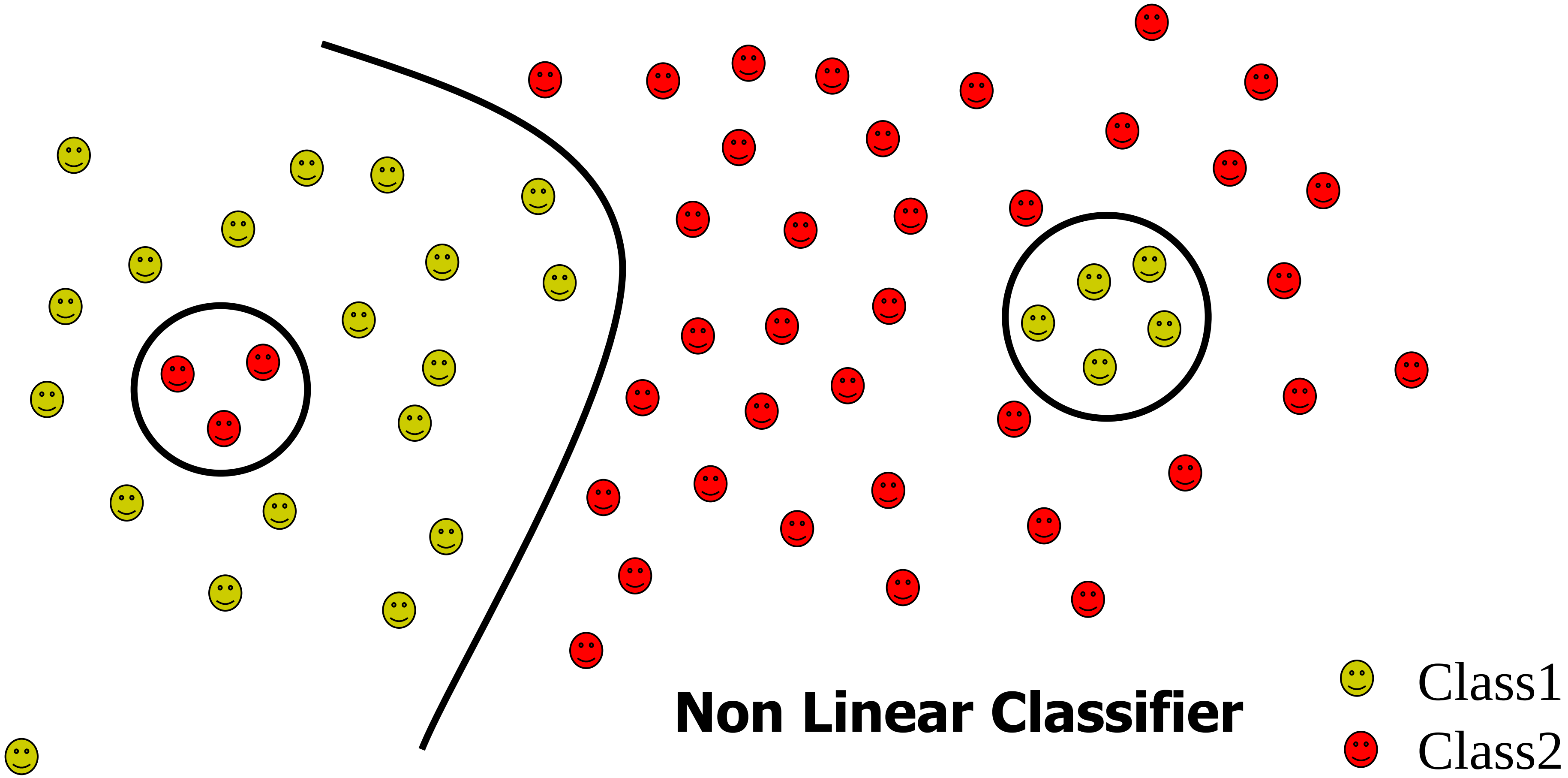
2) Boosted Decision Trees

3) Random Forest

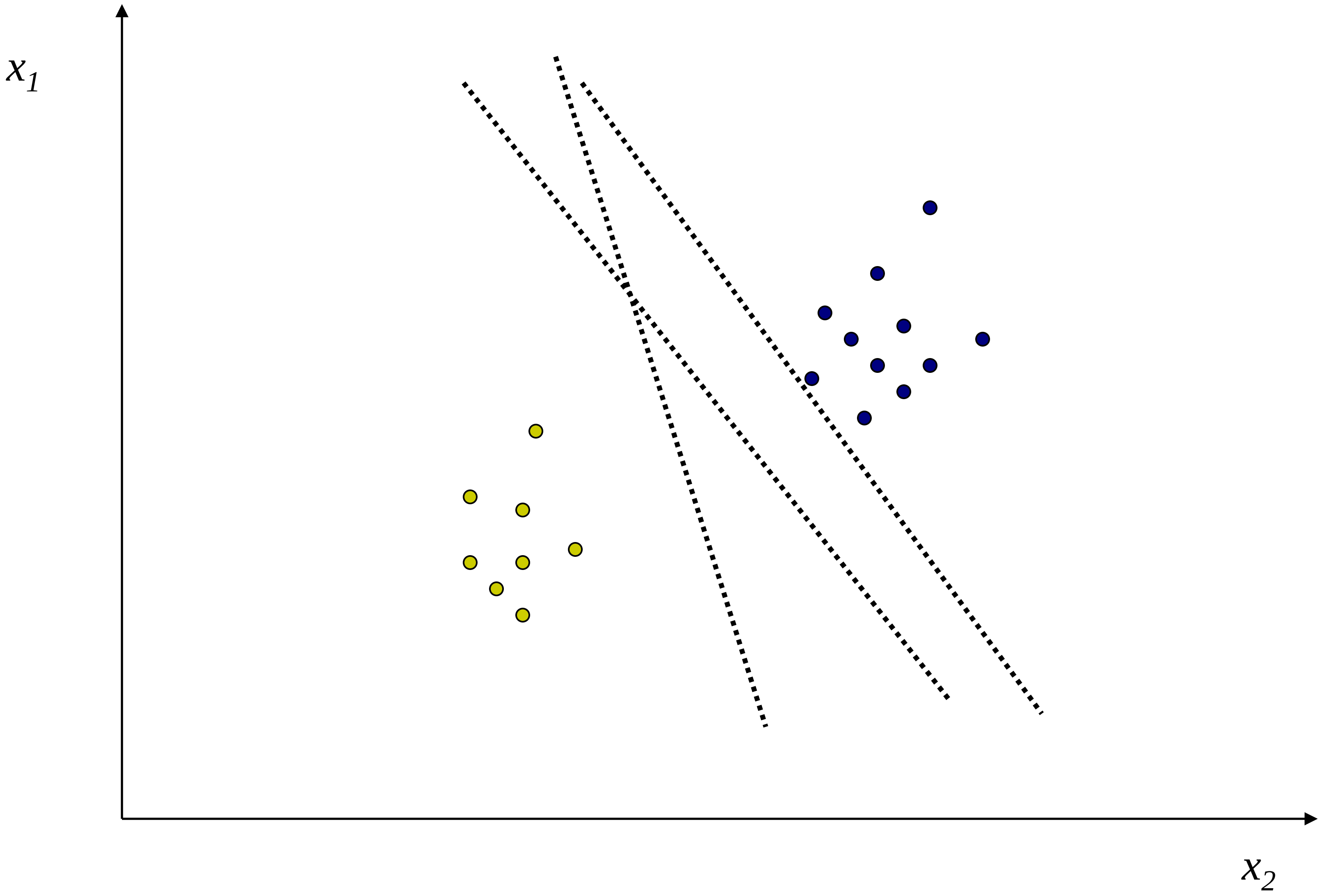
Linearly Separable Data



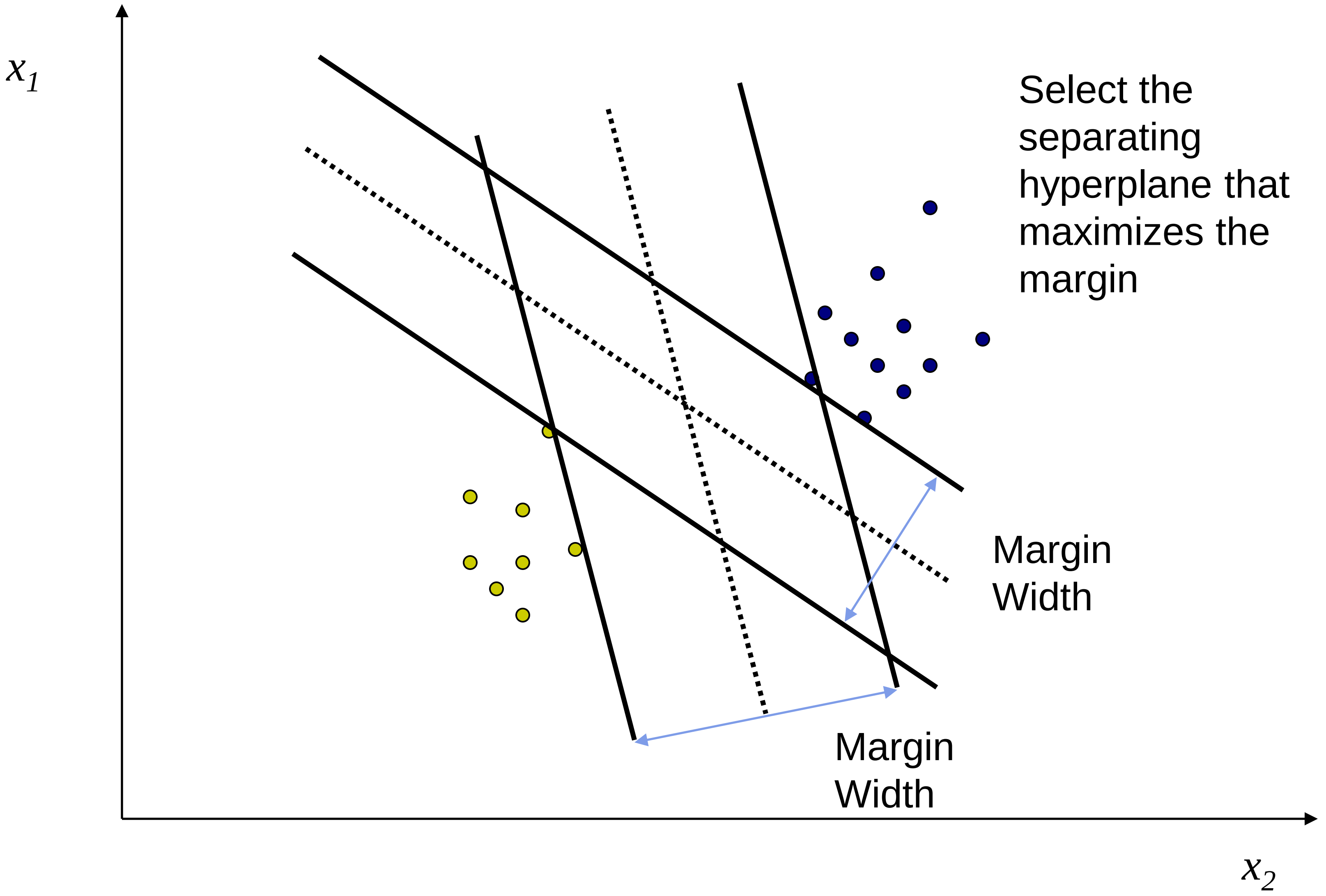
Nonlinearly Separable Data



Which Separating Hyperplane to Use?

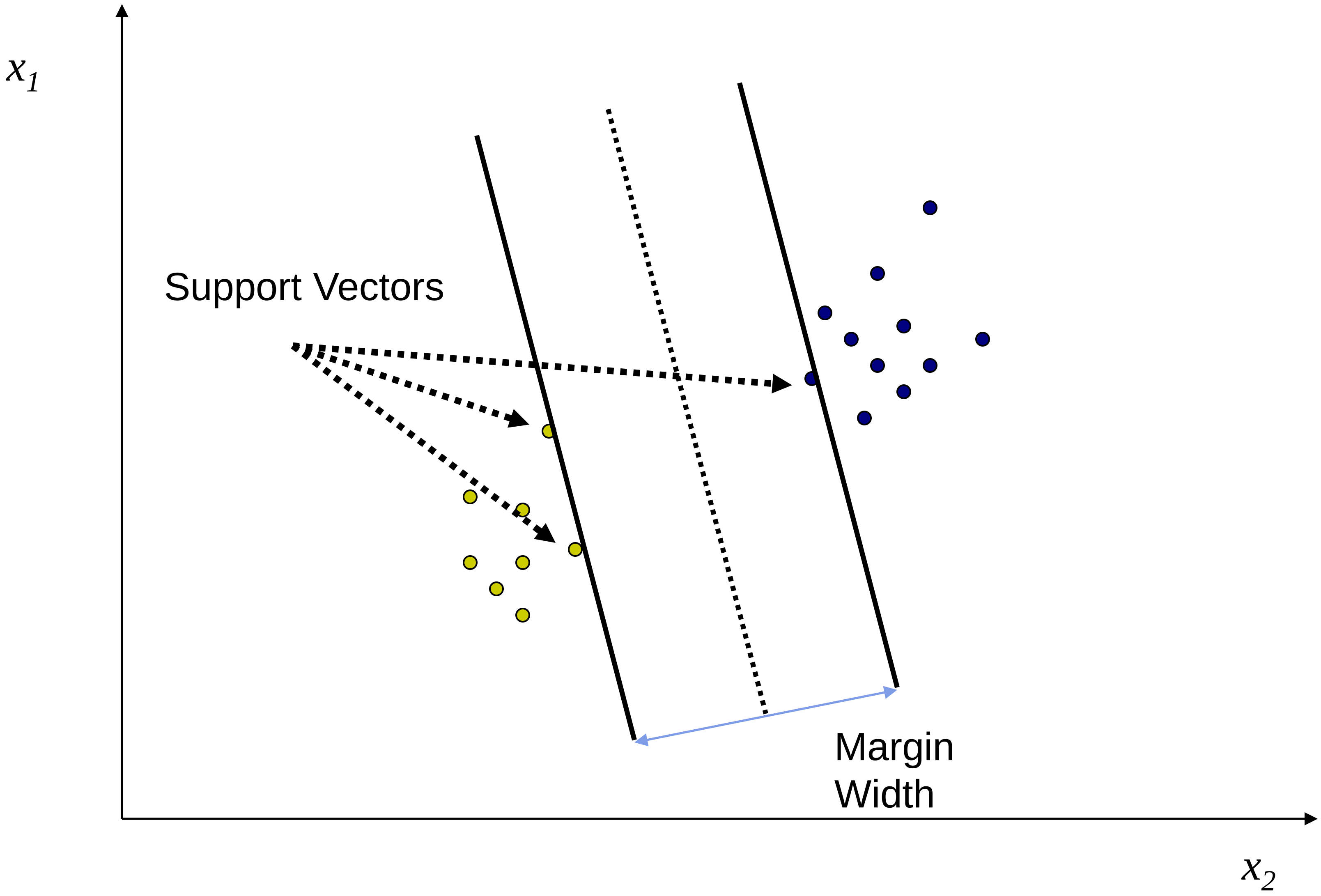


Maximizing the Margin

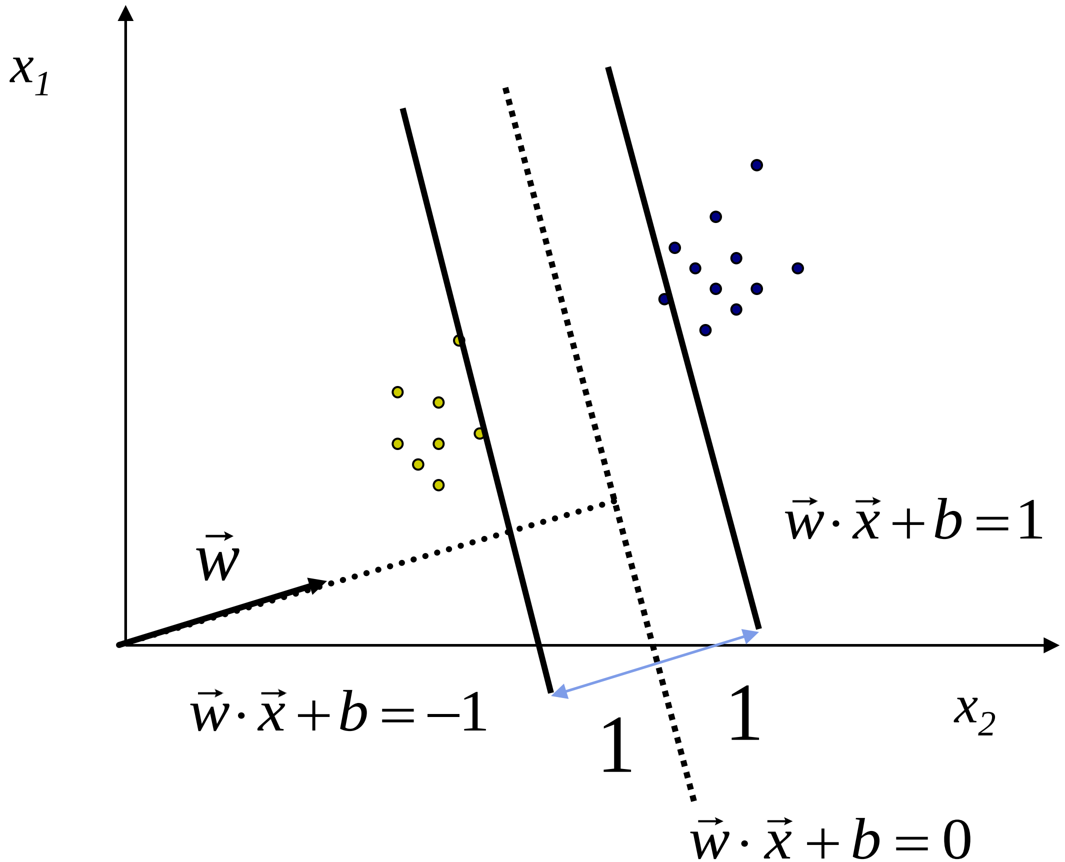


Select the separating hyperplane that maximizes the margin

Support Vectors



Setting Up the Optimization Problem



The maximum margin can be characterized as a solution to an optimization problem:

$$\max \frac{2}{\|w\|}$$

$$s.t.\ (w\cdot x+b)\geq 1,\ \forall x \text{ of class } 1$$

$$(w\cdot x+b)\leq -1,\ \forall x \text{ of class } 2$$

Setting Up the Optimization Problem

- If class 1 corresponds to 1 and class 2 corresponds to -1, we can rewrite

$$(w \cdot x_i + b) \geq 1,\ \forall x_i \text{ with } y_i = 1$$
$$(w \cdot x_i + b) \leq -1,\ \forall x_i \text{ with } y_i = -1$$

- as

$$y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

- So the problem becomes:

$$\max \frac{2}{\|w\|}$$
$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

or

$$\min \frac{1}{2}\|w\|^2$$
$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

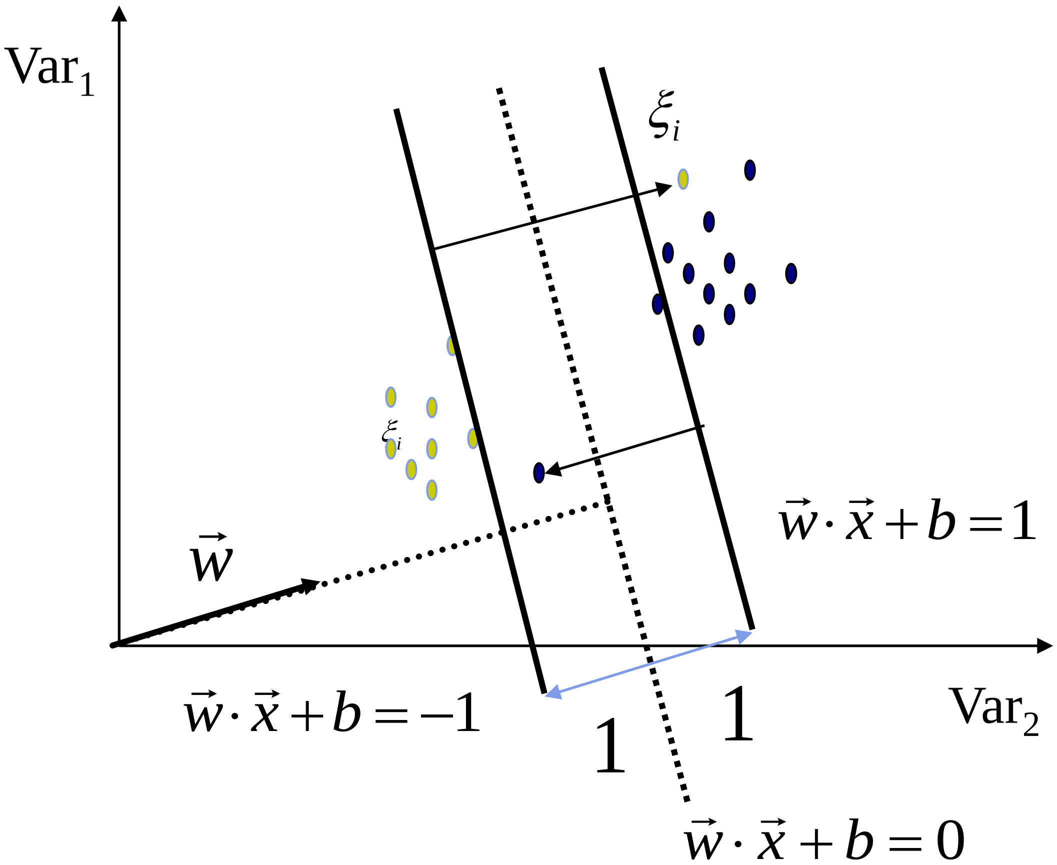
Linear, Hard-Margin SVM Formulation

- Find *w,b* that solves

$$\min \frac{1}{2}\|w\|^2$$
$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ \forall x_i$$

- Problem is convex so, there is a unique global minimum value (when feasible)
- There is also a unique minimizer, i.e. weight and *b* value that provides the minimum
- Quadratic Programming
  - very efficient computationally with procedures that take advantage of the special structure

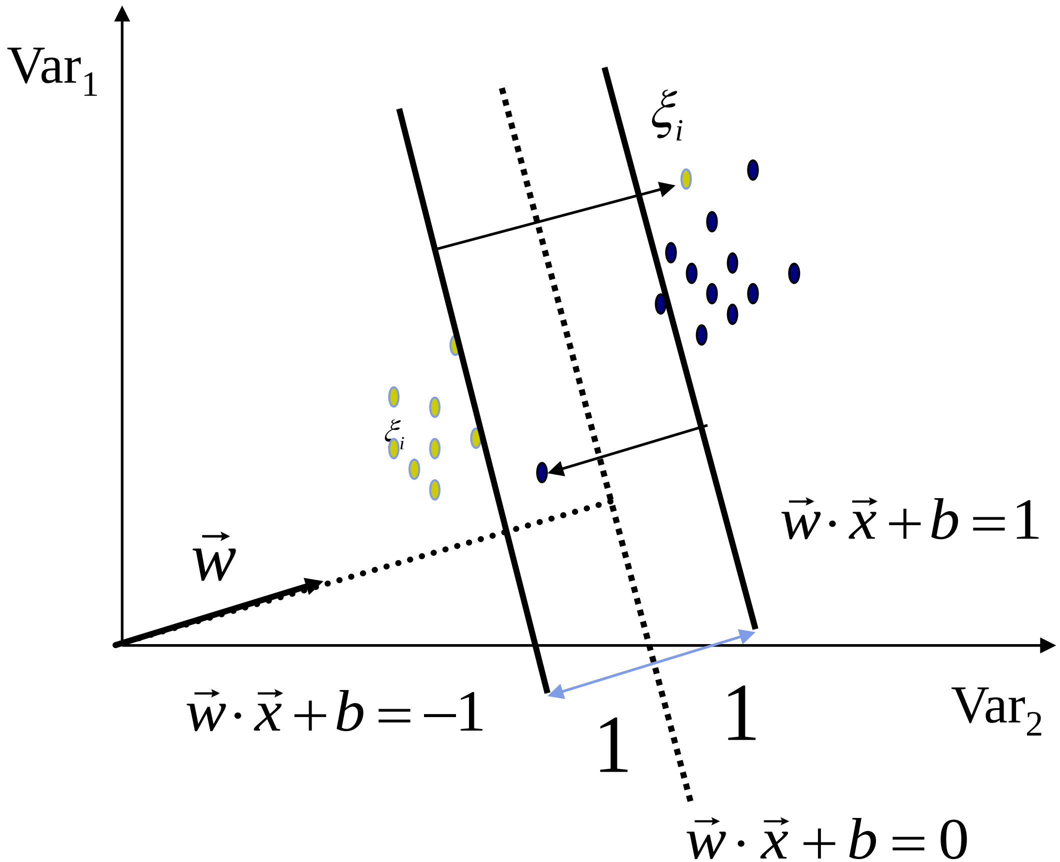
Nonlinearly Separable Data



Introduce slack variables ξ_i

Allow some instances to fall within the margin, but penalize them

Formulating the Optimization Problem



Constraints becomes :

$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$
$$\xi_i\geq 0$$

Objective function penalizes for misclassified instances and those within the margin

$$\min\frac{1}{2}\|w\|^2+C\sum_i\xi_i$$

C trades-off margin width and misclassifications

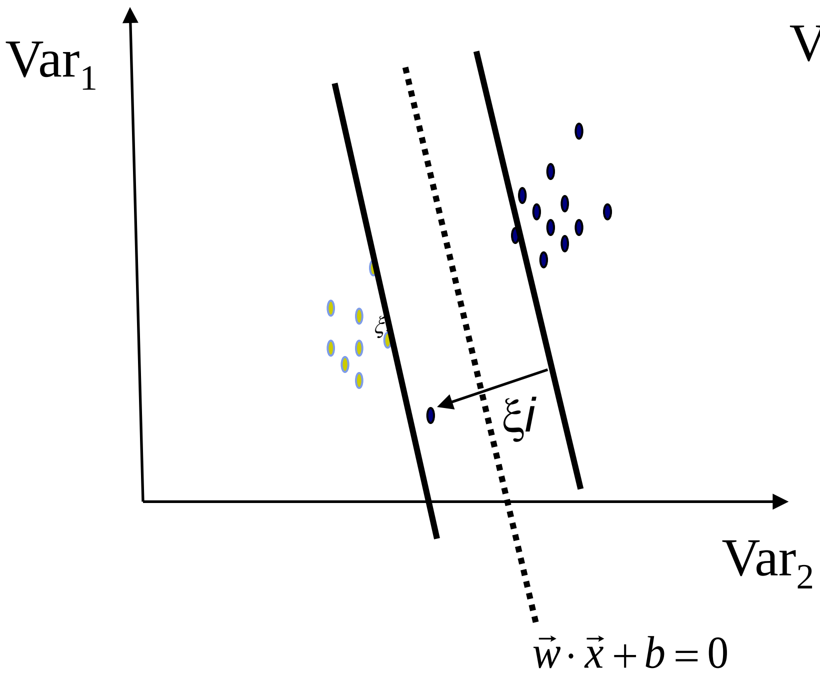
Linear, Soft-Margin SVMs

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

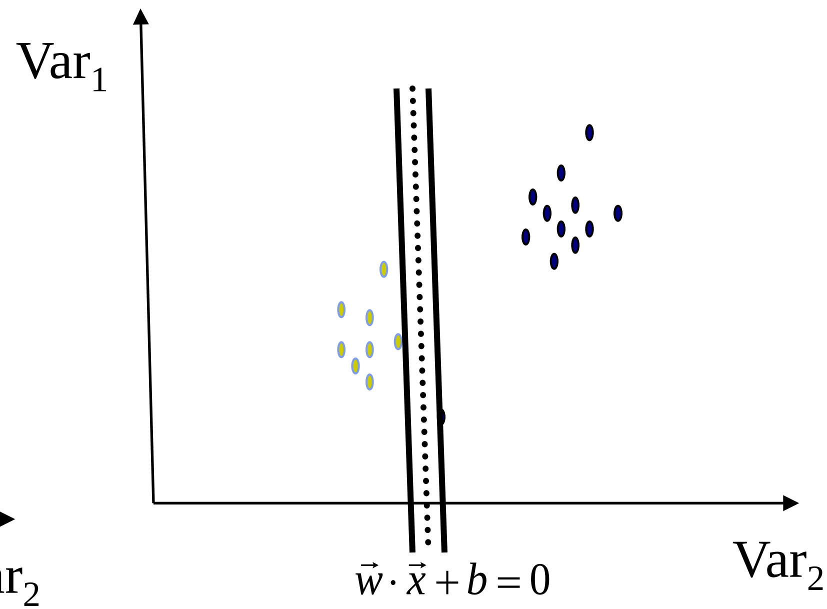
$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \;\; \forall x_i$$
$$\xi_i \geq 0$$

- Algorithm tries to maintain ξ_i to zero while maximizing margin
- Notice: algorithm does not minimize the *number* of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes
- Other formulations use ξ_i^2 instead
- As $C \to 0$, we get the hard-margin solution

Robustness of Soft vs Hard Margin SVMs

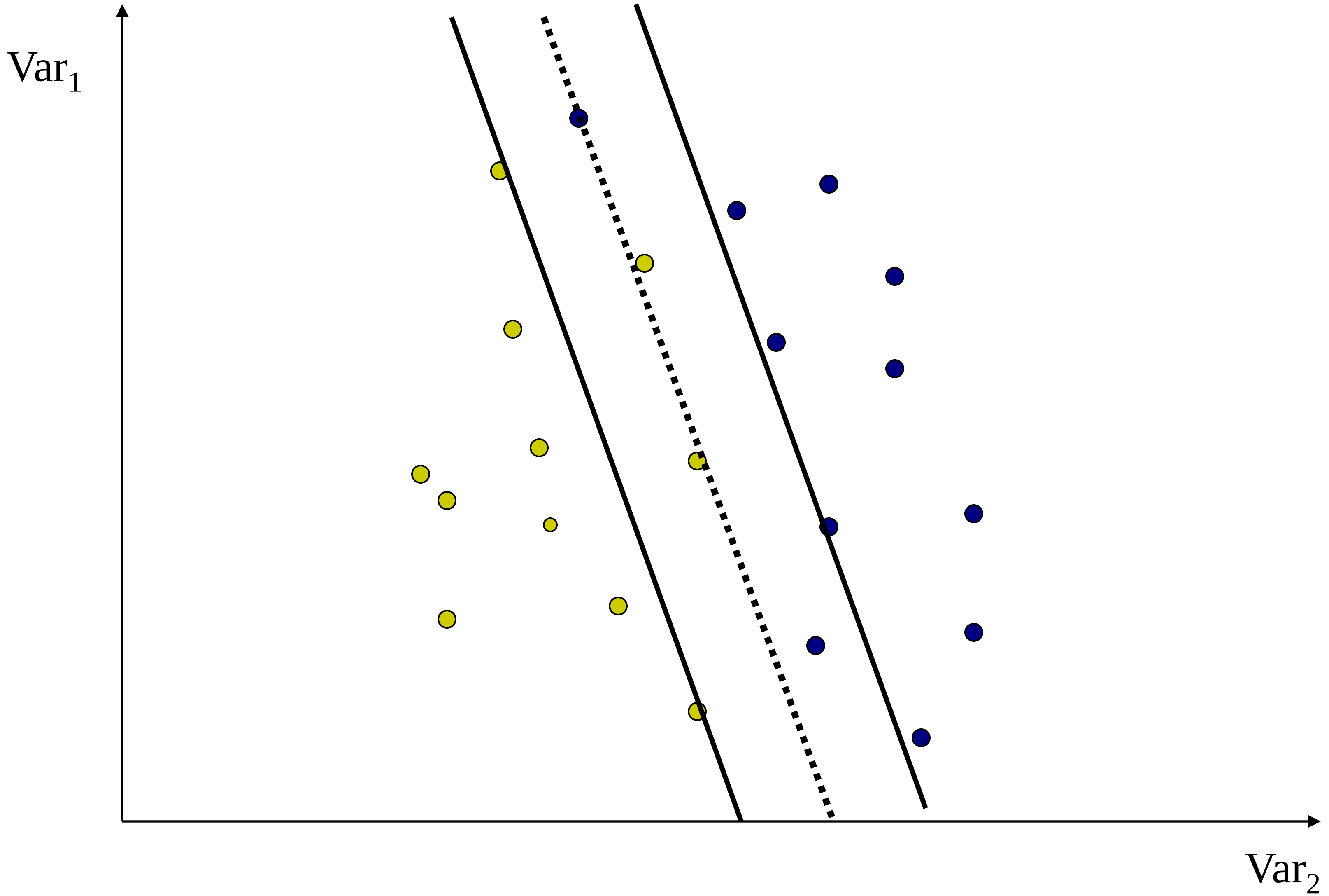


Soft Margin SVN

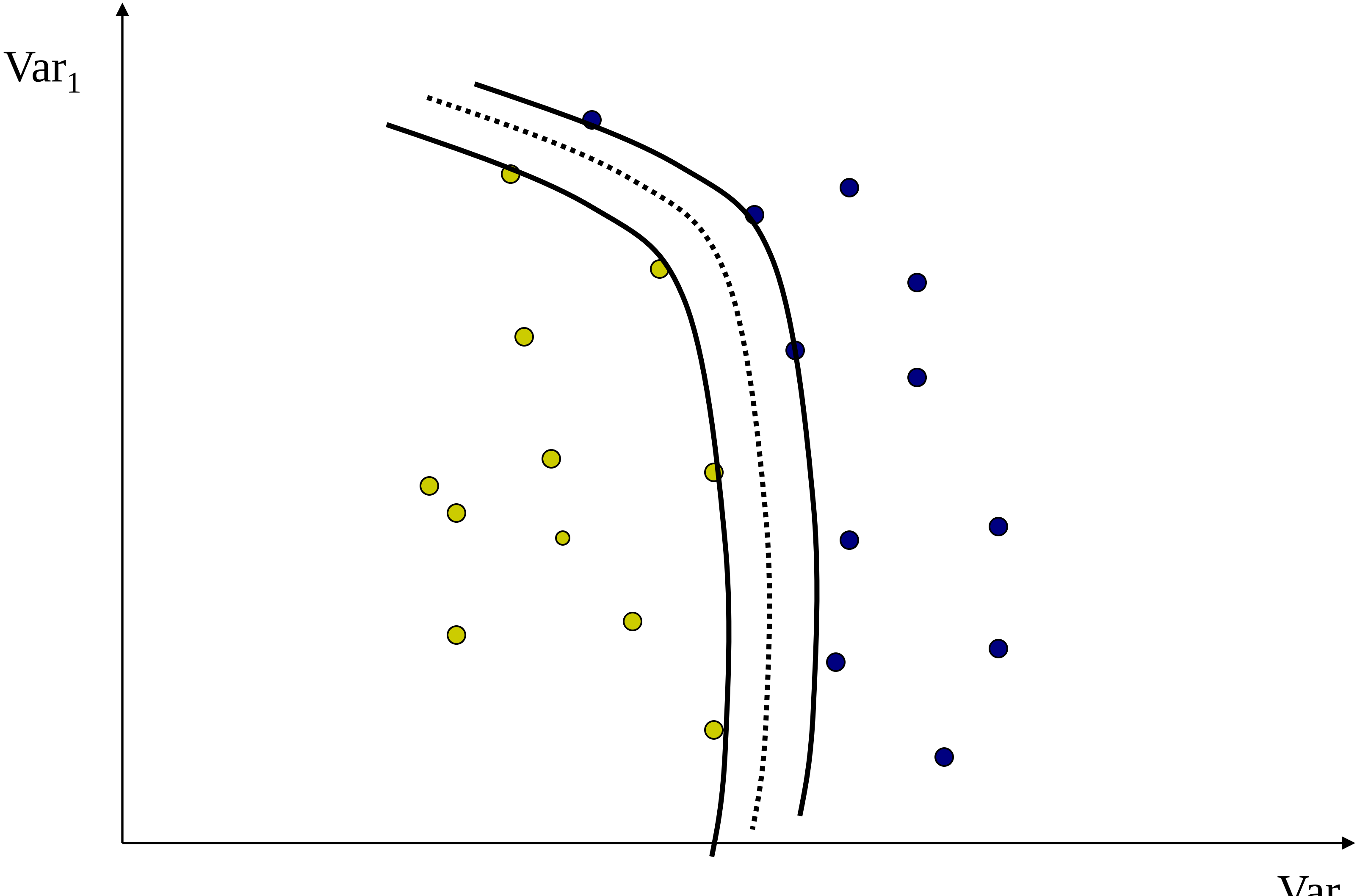


Hard Margin SVN

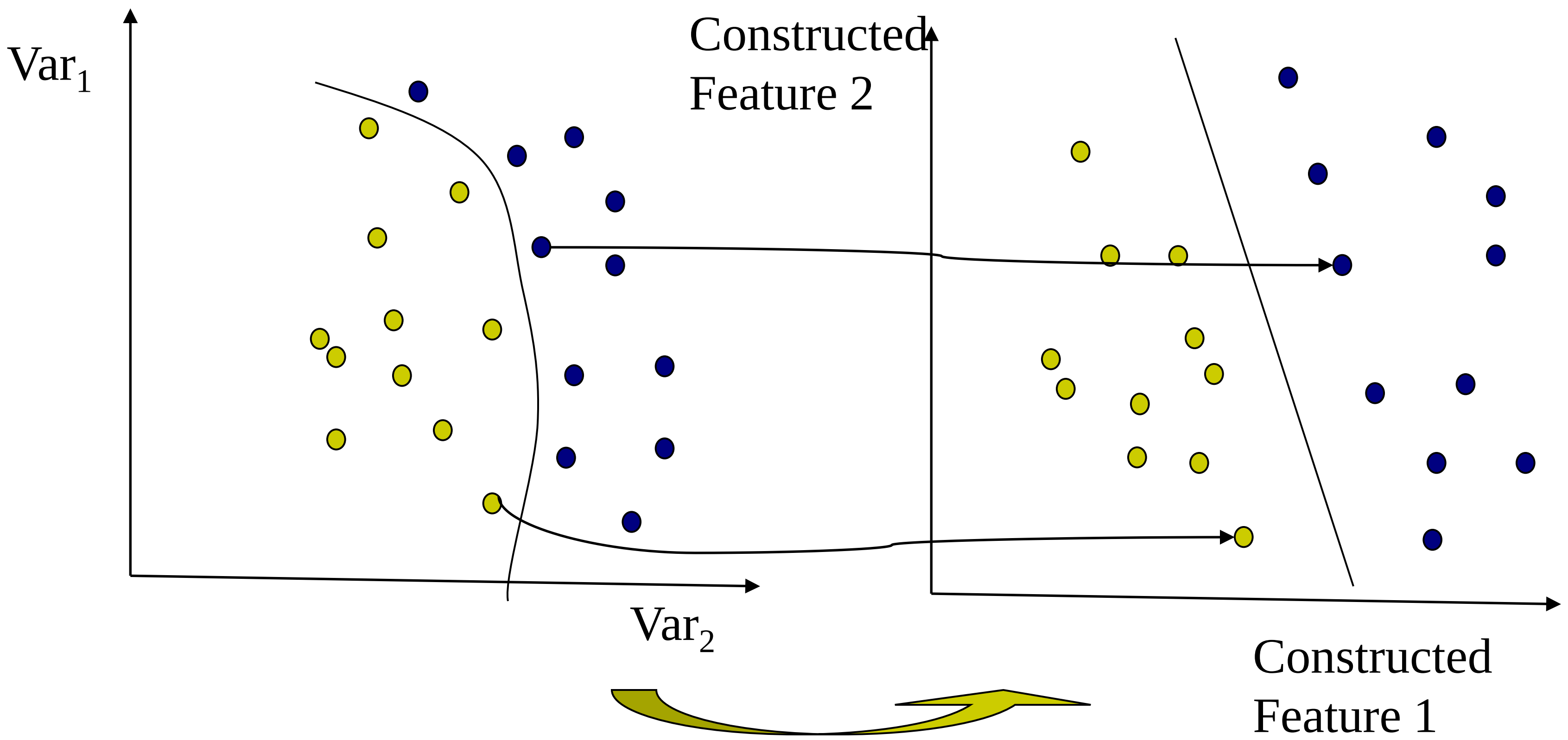
Disadvantages of Linear Decision Surfaces



Advantages of Nonlinear Surfaces



Linear Classifiers in High-Dimensional Spaces



Find function $\Phi(x)$ to map to a different space

Mapping Data to a High-Dimensional Space

- Find function Φ(x) to map to a different space, then SVM formulation becomes:

$$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \qquad s.t.\ \ y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$
$$\xi_i \geq 0$$

- Data appear as Φ(x), weights *w* are now weights in the new space
- Explicit mapping expensive if Φ(x) is very high dimensional
- Solving the problem without explicitly mapping the data is desirable

The Dual of the SVM Formulation

- Original SVM formulation
  - *n* inequality constraints
  - *n positivity constraints*
  - *n* number of ξ variables

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$$

$$s.t. \;\; y_i(w \cdot \Phi(x) + b) \geq 1 - \xi_i, \forall x_i$$

$$\xi_i \geq 0$$

- The (Wolfe) dual of this problem
  - one equality constraint
  - *n* positivity constraints
  - *n* number of α variables (Lagrange multipliers)
  - Objective function more complicated

$$\min_{w,b} \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) - \sum_i \alpha_i$$

$$s.t. \;\; \mathrm{C} \geq \alpha_{\mathrm{i}} \geq 0, \forall x_i$$

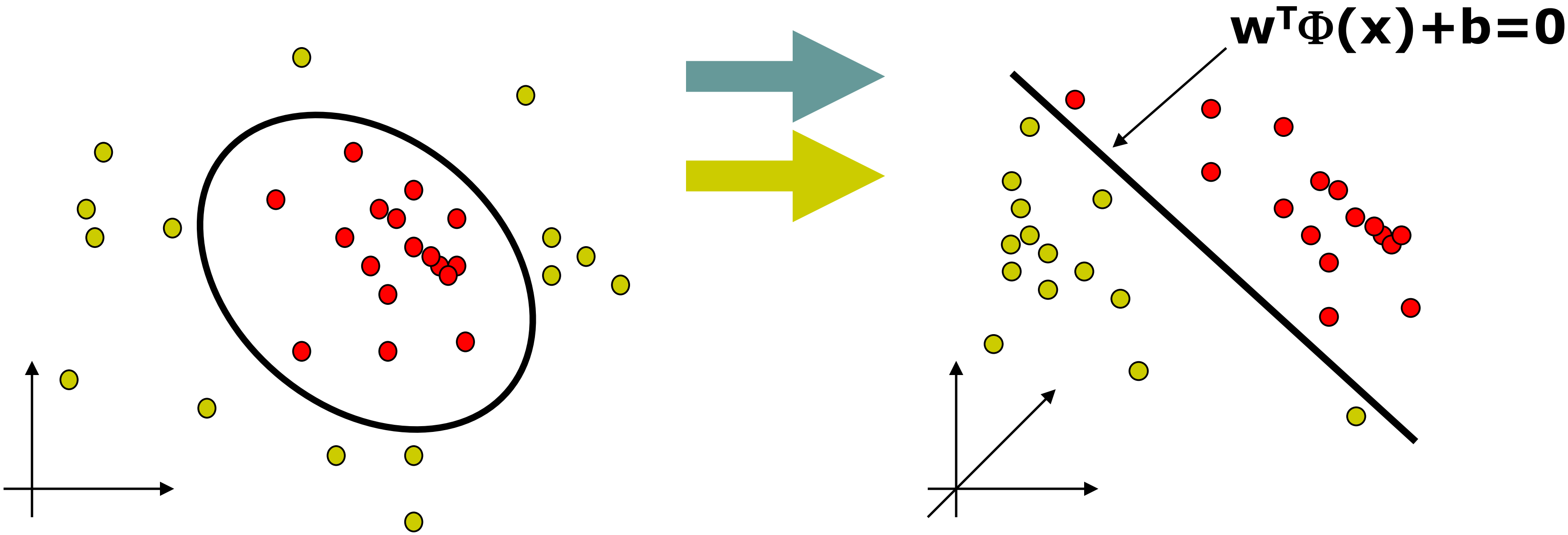
$$\sum_i \alpha_i y_i = 0$$

- NOTE: Data only appear as $\Phi(x_i) \cdot \Phi(x_j)$

The Kernel Trick

- $\Phi(x_i) \cdot \Phi(x_j)$: means, map data into new space, then take the inner product of the new vectors
- We can find a function such that: $K(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$, i.e., the image of the inner product of the data is the inner product of the images of the data
- Then, we do not need to explicitly map the data into the high-dimensional space to solve the optimization problem

Example



$X=[x\ z]$ $\qquad$ $\Phi(X)=[x^2\ z^2\ xz]$

$f(x) = \text{sign}(w_1x^2+w_2z^2+w_3xz + b)$

Example

$$X_1=[x_1\ z_1] \rightarrow \Phi(X_1)=[x_1^2\ z_1^2\ 2^{1/2}x_1 z_1]$$

$$X_2=[x_2\ z_2] \rightarrow \Phi(X_2)=[x_2^2\ z_2^2\ 2^{1/2}x_2 z_2]$$

$$\Phi(X_1)^T\Phi(X_2)= [x_1^2\ z_1^2\ 2^{1/2}x_1 z_1]\ [x_2^2\ z_2^2\ 2^{1/2}x_2 z_2]^T$$

Expensive!

$$= x_1^2 z_1^2 + x_2^2 z_2^2 + 2\ x_1\ z_1\ x_2\ z_2$$

O(d²)

$$= (x_1 z_1 + x_2 z_2)^2$$

Efficient!

$$= (X_1^T\ X_2)^2$$

O(d)

Kernel Trick

- **Kernel function**: a symmetric function

  k : R^d x R^d -> R

- **Inner product kernels**: additionally,

  k(x,z) = Φ(x)^T Φ(z)

- Example:

O(d²) **O(d)**

$$\boxed{\Phi(x)^T \Phi(z) = \sum_{i,j=(1,1)}^{d,d} (x_i x_j)(z_i z_j)} = \left(\sum_{i=1}^{d} x_i z_i \right)^2 = \boxed{(x^T z)^2 = K(x,z)}$$

Kernel Trick

- Implement an infinite-dimensional mapping implicitly
- Only inner products explicitly needed for training and evaluation
- Inner products computed efficiently, in finite dimensions
- The underlying mathematical theory is that of reproducing kernel Hilbert space from functional analysis

Kernel Methods

- If a linear algorithm can be expressed only in terms of inner products
  - it can be “kernelized”
  - find linear pattern in high-dimensional space
  - nonlinear relation in original space
- Specific kernel function determines nonlinearity

Kernels

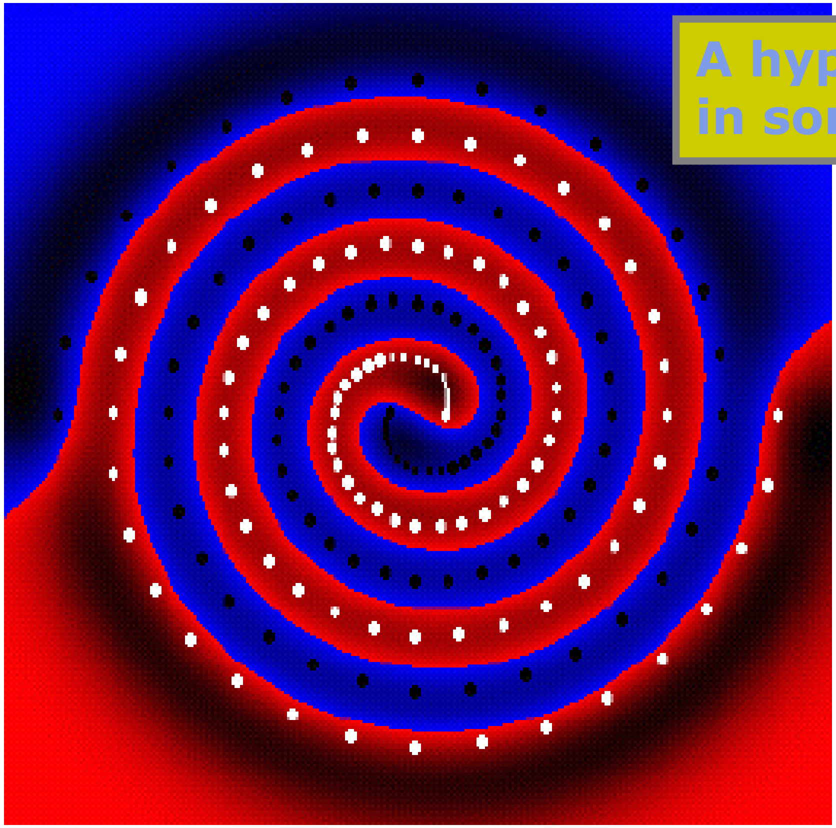
- Some simple kernels
  - Linear kernel: **$k(x,z) = x^T z$**

    → equivalent to linear algorithm
  - Polynomial kernel: **$k(x,z) = (1+x^T z)^d$**

    → polynomial decision rules
  - RBF kernel: **$k(x,z) = \exp(-||x-z||^2/2\sigma)$**

    → highly nonlinear decisions

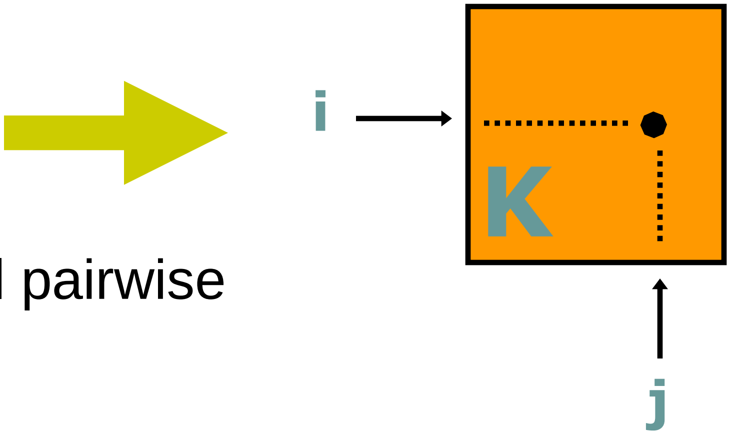
Gaussian Kernel: Example



A hyperplane in some space

Kernel Matrix

$k(x,y)$

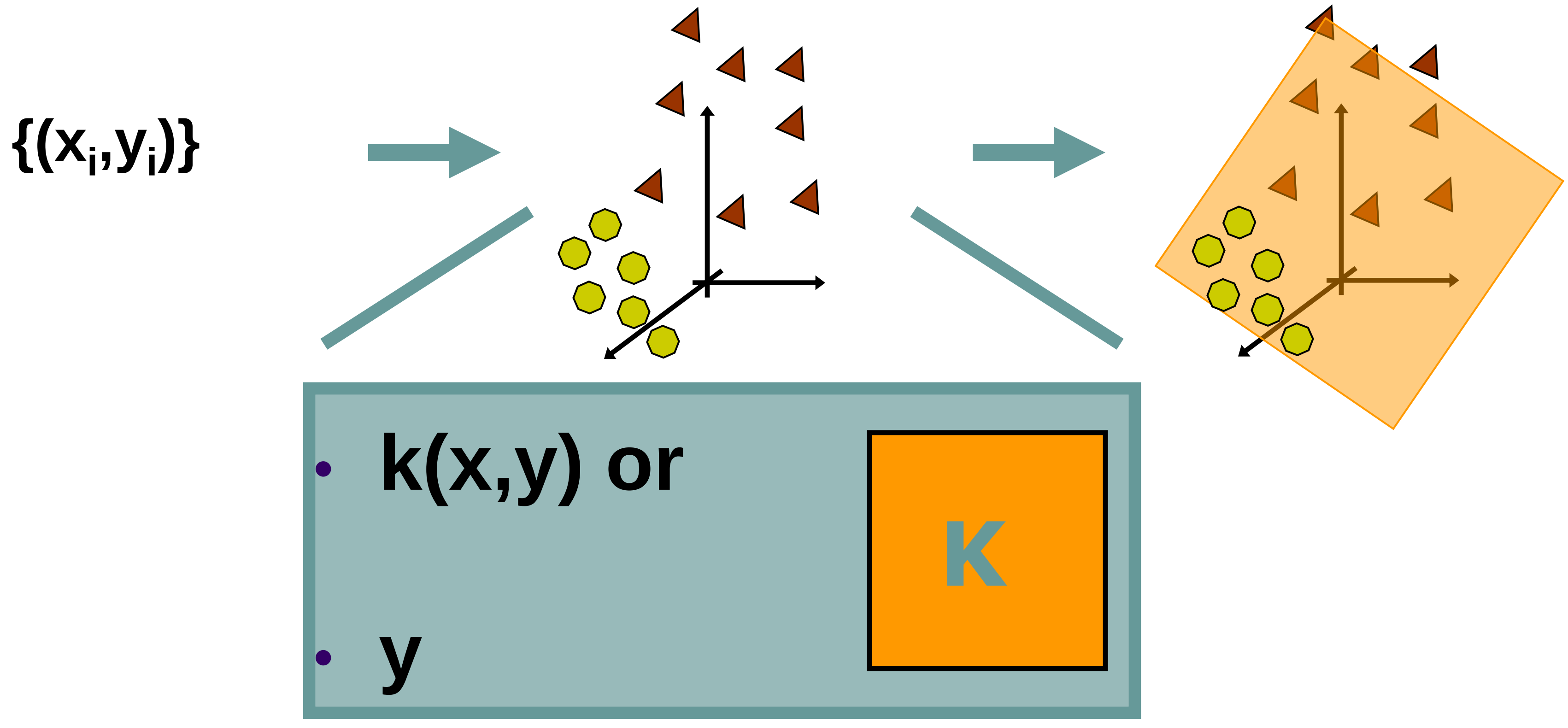


- Kernel matrix **K** defines all pairwise inner products
- Mercer theorem: **K** positive semidefinite
- Any symmetric positive semidefinite matrix can be regarded as an inner product matrix in some space

$$K_{ij}=k(x_i,x_j)$$

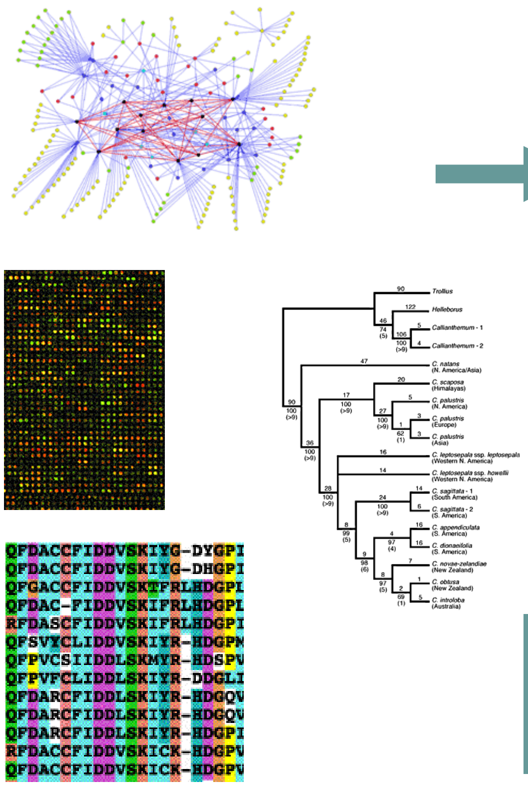
Kernel-Based Learning

Data **Embedding** **Linear algorithm**

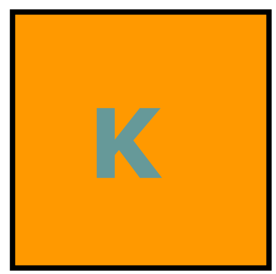


Kernel-Based Learning

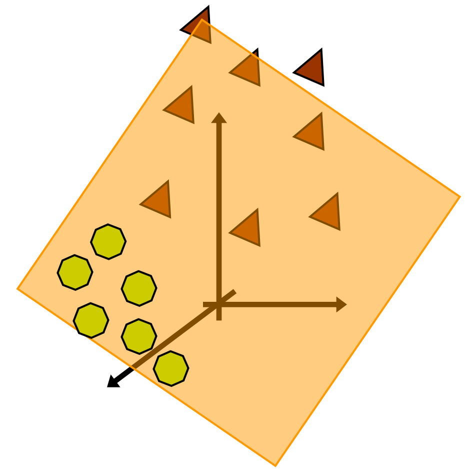
Data



Embedding



Linear algorithm



Kernel design

Kernel algorithm

Kernel Design

- Simple kernels on vector data
- More advanced
  - string kernel
  - diffusion kernel
  - kernels over general structures (sets, trees, graphs...)
  - kernels derived from graphical models
  - empirical kernel map

Methods

I) Instance-based methods:
  1) Nearest neighbor
II) Probabilistic models:
  1) Naïve Bayes
  2) Logistic Regression
III) Linear Models:
  1) Perceptron
  2) Support Vector Machine
IV) Decision Models:
  1) Decision Trees
  2) Boosted Decision Trees
  3) Random Forest

Top-Down Induction of Decision Trees [ID3, C4.5, …]

node = Root

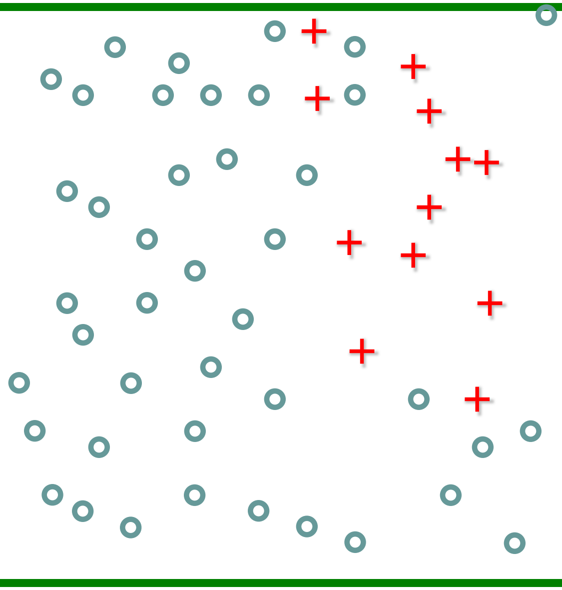
Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

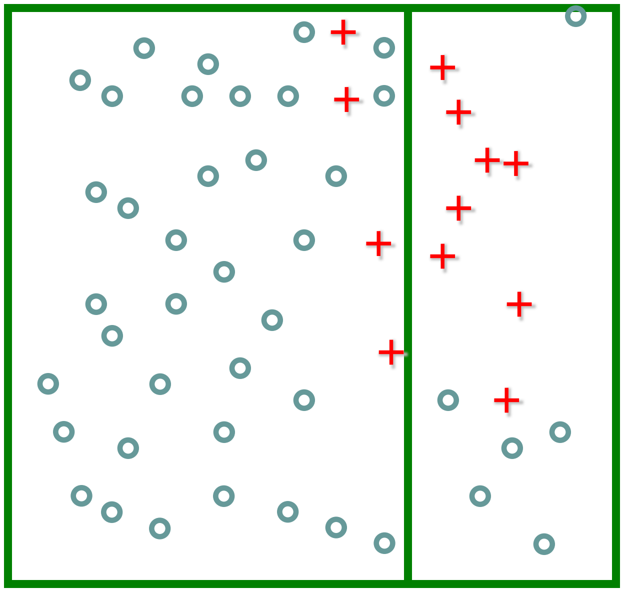
Which attribute is best?

[From Tom Mitchell's slides]

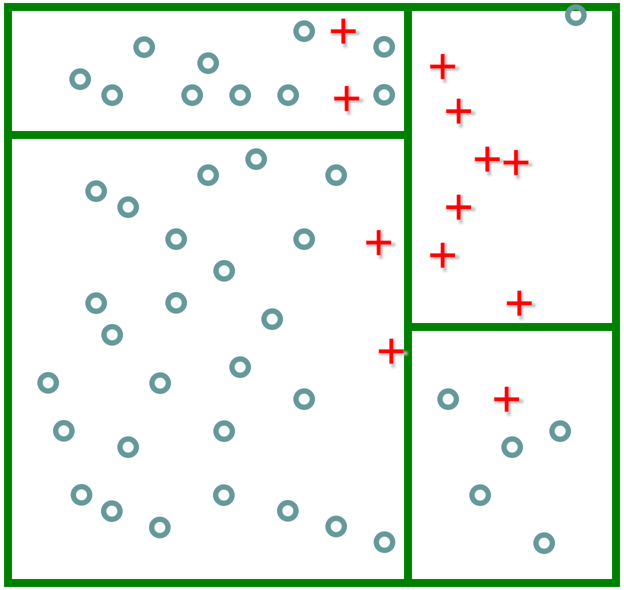
Spatial example: recursive binary splits



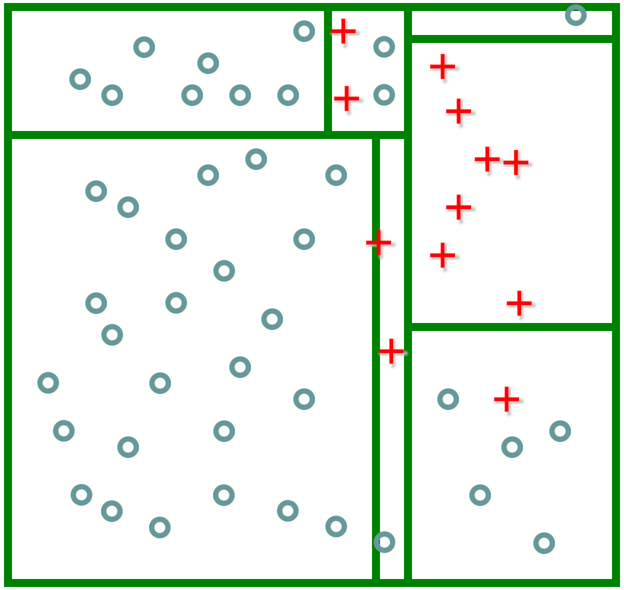
Spatial example: recursive binary splits



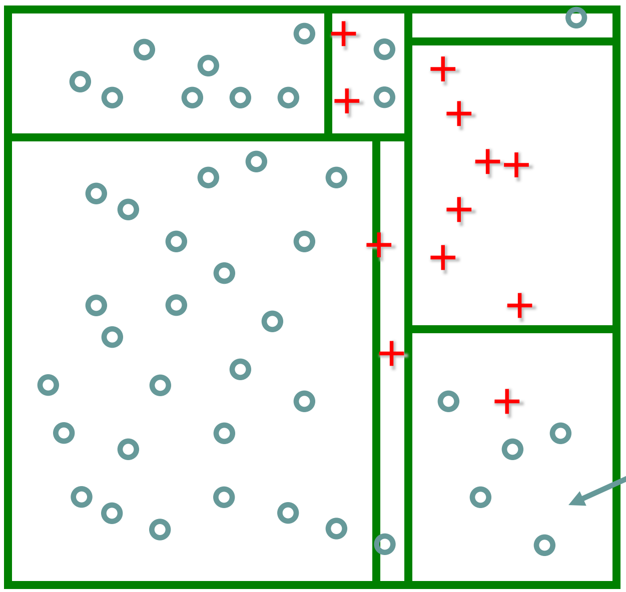
Spatial example: recursive binary splits



Spatial example: recursive binary splits



Spatial example: recursive binary splits



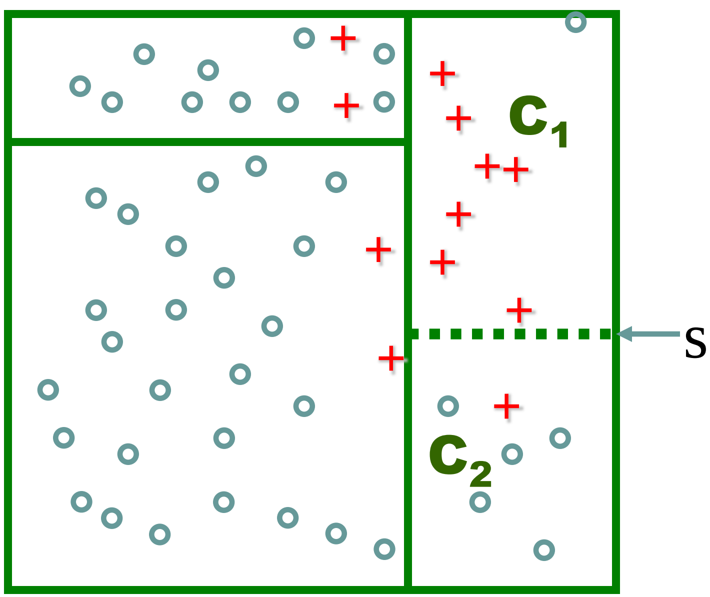
Once regions are chosen class probabilities are easy to calculate

p_m=5/6

How to choose a split

$N_1=9$

$p_1=8/9$



$p_2=5/6$

$N_2=6$

Impurity measures: L(p)

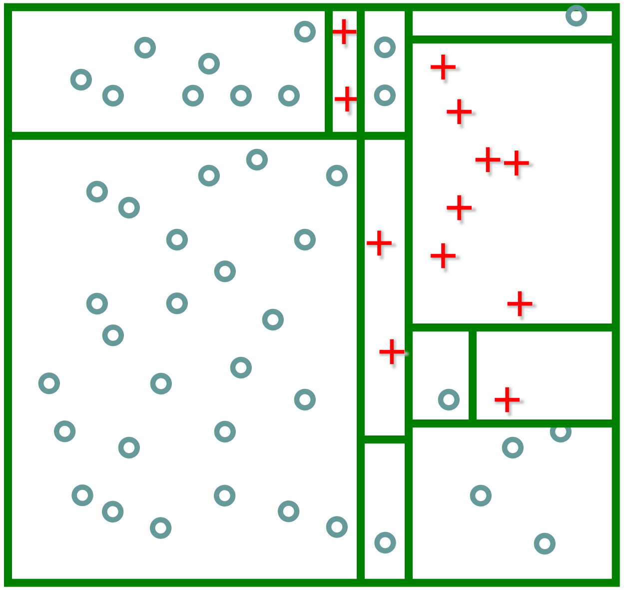
- Information gain (entropy):

  $- p \log p - (1-p) \log(1-p)$
- Gini index: $2\, p\, (1-p)$
- (0-1 error: $1-\max(p,1-p)$)

$$\boxed{\min_s N_1\, L(p_1)+N_2\, L(p_2)}$$

Then choose the region that has the best split

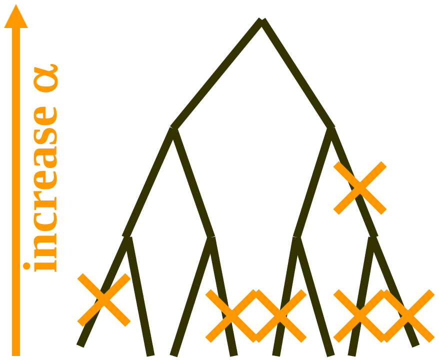
Overfitting and pruning



L: 0-1 loss

$\min_T \sum_i L(x_i) + \alpha |T|$

then choose α with CV



Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

IV) Decision Models:

1) Decision Trees

2) Boosted Decision Trees

3) Random Forest

Methods

I) Instance-based methods:

1) Nearest neighbor

II) Probabilistic models:

1) Naïve Bayes

2) Logistic Regression

III) Linear Models:

1) Perceptron

2) Support Vector Machine

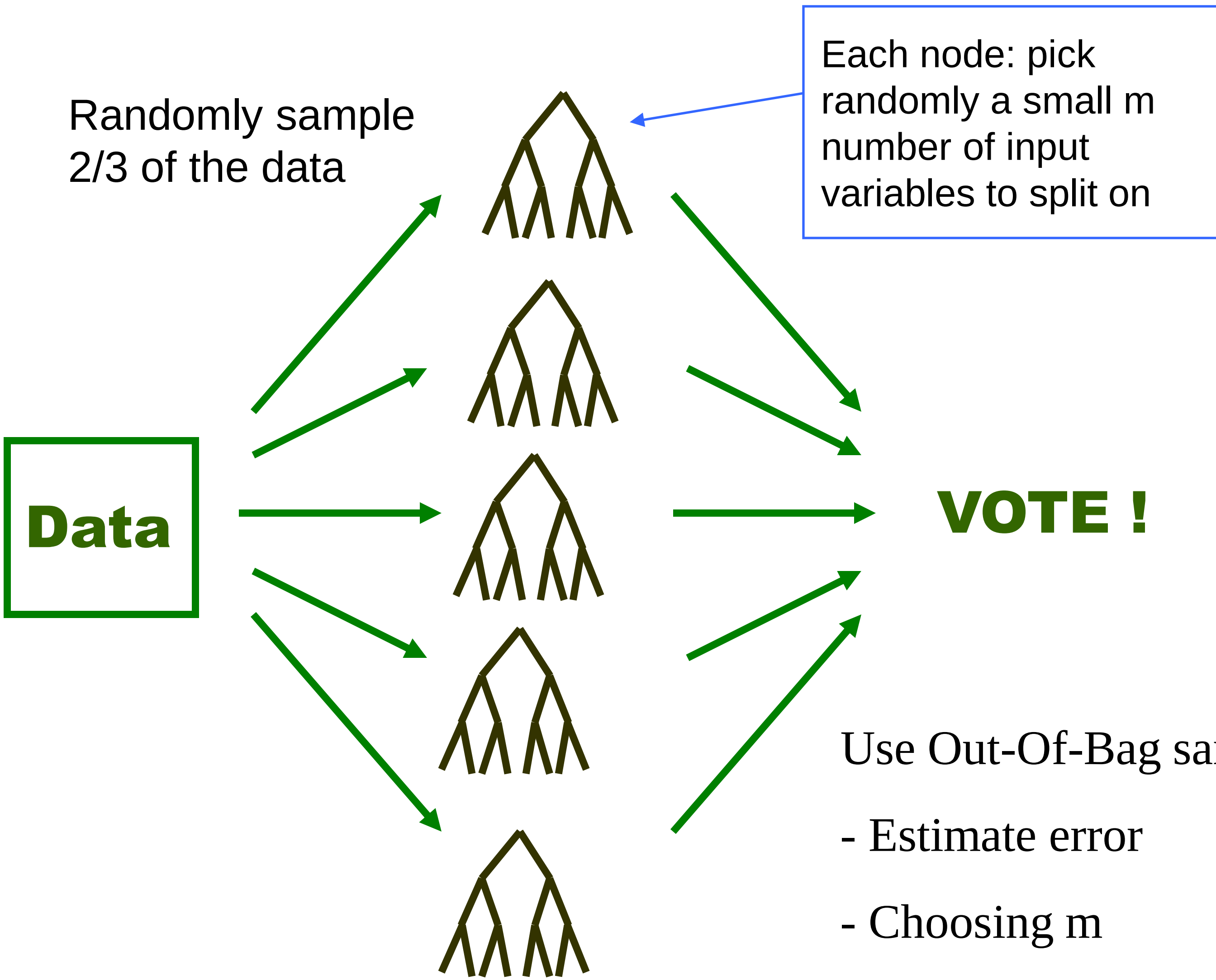
IV) Decision Models:

1) Decision Trees

2) Boosted Decision Trees

3) Random Forest

Random Forest



Use Out-Of-Bag samples to:

- Estimate error

- Choosing m

- Estimate variable importance

Summary of algorithms

$\square$ = state of the art

1) KNN: good first try if have "meaningful" $d(x, x')$

2) NB:

3) avg. perceptron / logistic regression / $\boxed{\text{SVM}}$
(in increasing order of complexity, memory requirement and prediction accuracy)
- they can all be kernelized $\Rightarrow$ can handle any data for which you can define meaningful kernel "similarity measure"
- are all vector space methods
$\Rightarrow$ deal with nominal data in unnatural way

- log. reg. output prob. [can be useful]
  - → can get prob. for SVM using Platt's trick

4) trees methods [decision tree / boosted DT / random forest]

- handle mixed attributes naturally (e.g. nominal + numeric)
- interpretable
- more scalable than SVM

TABLE 10.1. *Some characteristics of different learning methods. Key:* 🟢 *= good,* 🟡 *=fair, and* 🔴 *=poor.*

Characteristic	Neural nets	SVM	Trees	MARS	k-NN, kernels
Natural handling of data of "mixed" type	🔴	🔴	🟢	🟢	🔴
Handling of missing values	🔴	🔴	🟢	🟢	🟢
Robustness to outliers in input space	🔴	🔴	🟢	🔴	🟢
Insensitive to monotone transformations of inputs	🔴	🔴	🟢	🔴	🔴
Computational scalability (large N)	🔴	🔴	🟢	🟢	🔴
Ability to deal with irrelevant inputs	🔴	🔴	🟢	🟢	🔴
Ability to extract linear combinations of features	🟢	🟢	🔴	🔴	🟡
Interpretability	🔴	🔴	🟡	🟢	🔴
Predictive power	🟢	🟢	🔴	🟡	🟢

Reading

- All of the methods that we have discussed are presented in the following book:

  Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edition), NY: Springer.

- We haven't discussed theory, but if you're interested in the theory of (binary) classification, here's a pointer to get started:

  Bartlett, P., Jordan, M. I., & McAuliffe, J. D. (2006). Convexity, classification and risk bounds. *Journal of the American Statistical Association, 101*, 138-156.