CS6510
Applied Machine Learning

Classifiers

12 Aug 2017

Vineeth N Balasubramanian

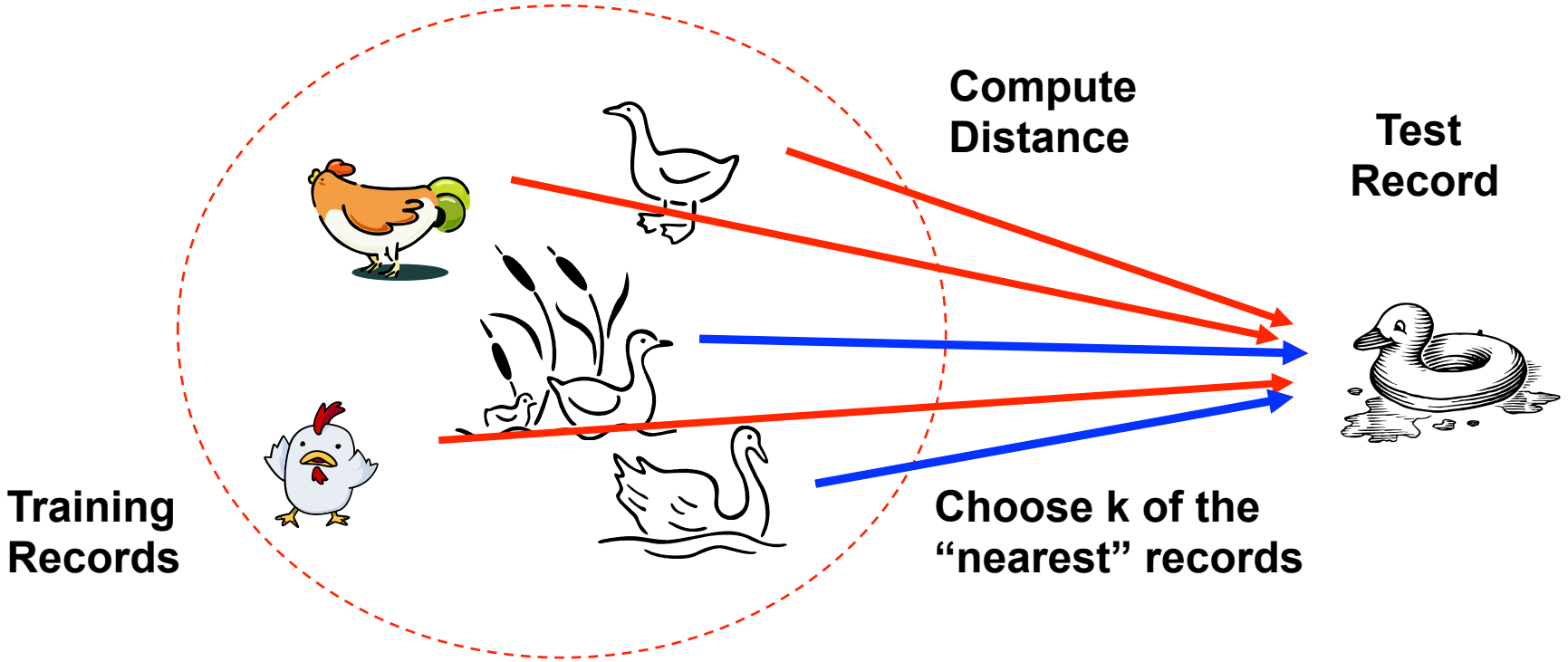
आई आई टी हैदराबाद
IIT Hyderabad

Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

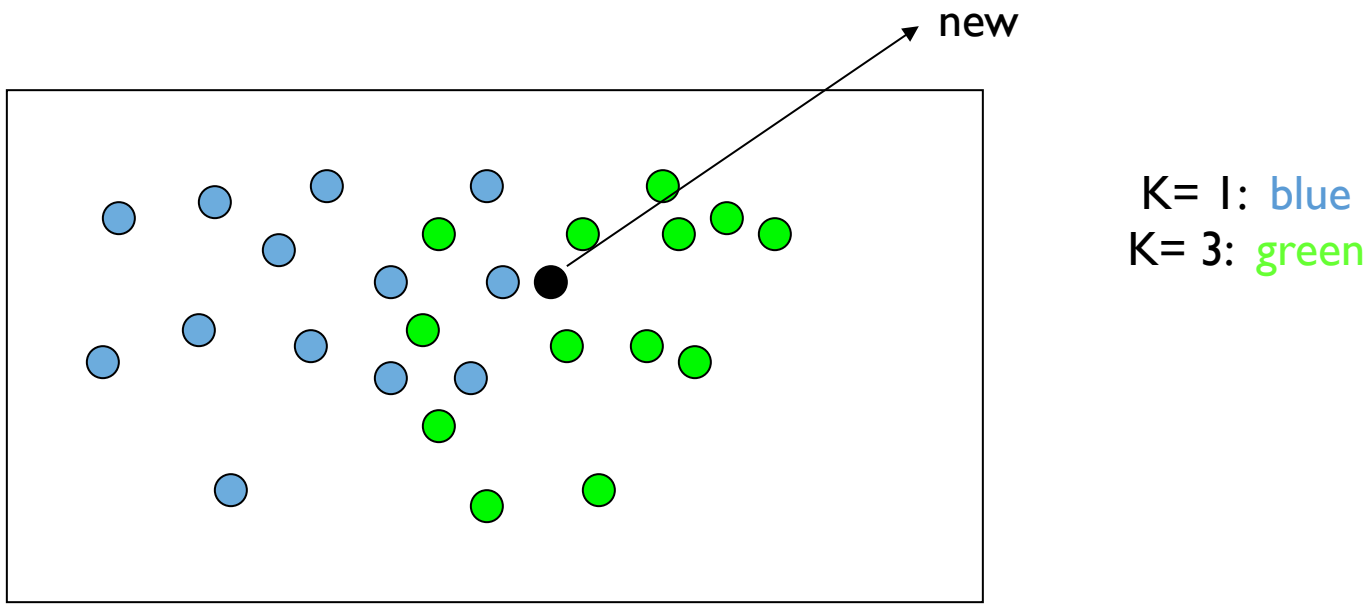
k-Nearest Neighbors

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



k-Nearest Neighbors

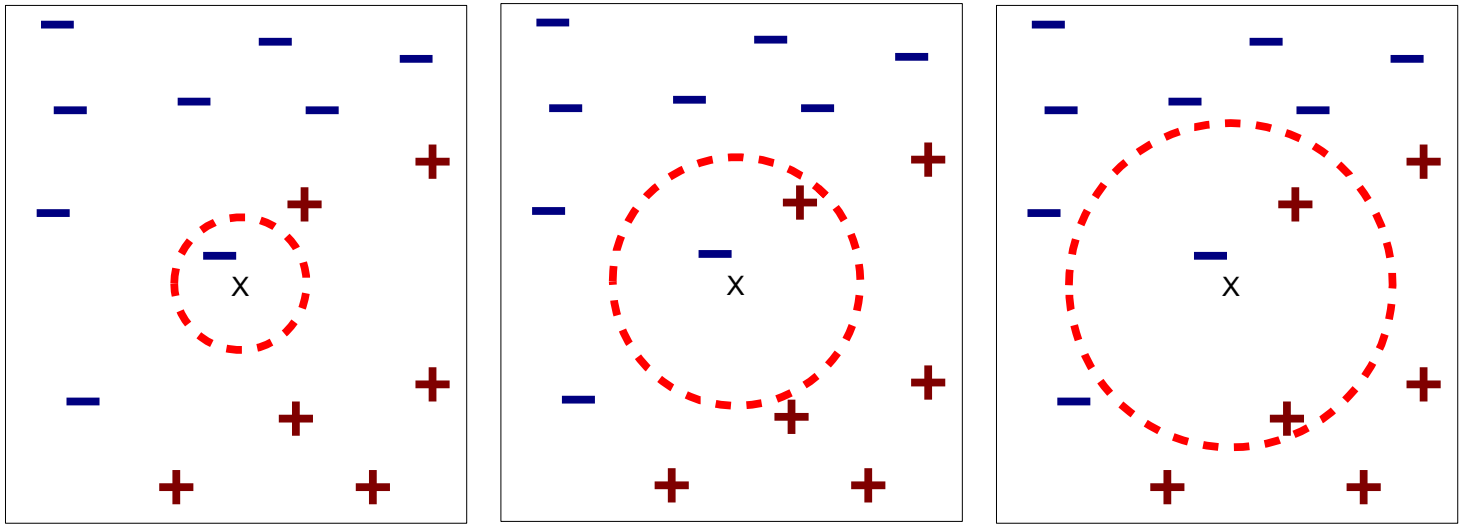
- Majority vote within the k nearest neighbors



K= 1: blue
K= 3: green

k-Nearest Neighbors

- Choosing k is important
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

k-Nearest Neighbors

- An arbitrary instance is represented by $(a_1(x), a_2(x), a_3(x),.., a_n(x))$
  - $a_i(x)$ denotes features
- Euclidean distance between two instances
  - $d(x_i, x_j)=\text{sqrt (sum for r=1 to n } (a_r(x_i) - a_r(x_j))^2)$
- In case of continuous-valued target function
  - Mean value of k nearest training examples

How to determine k

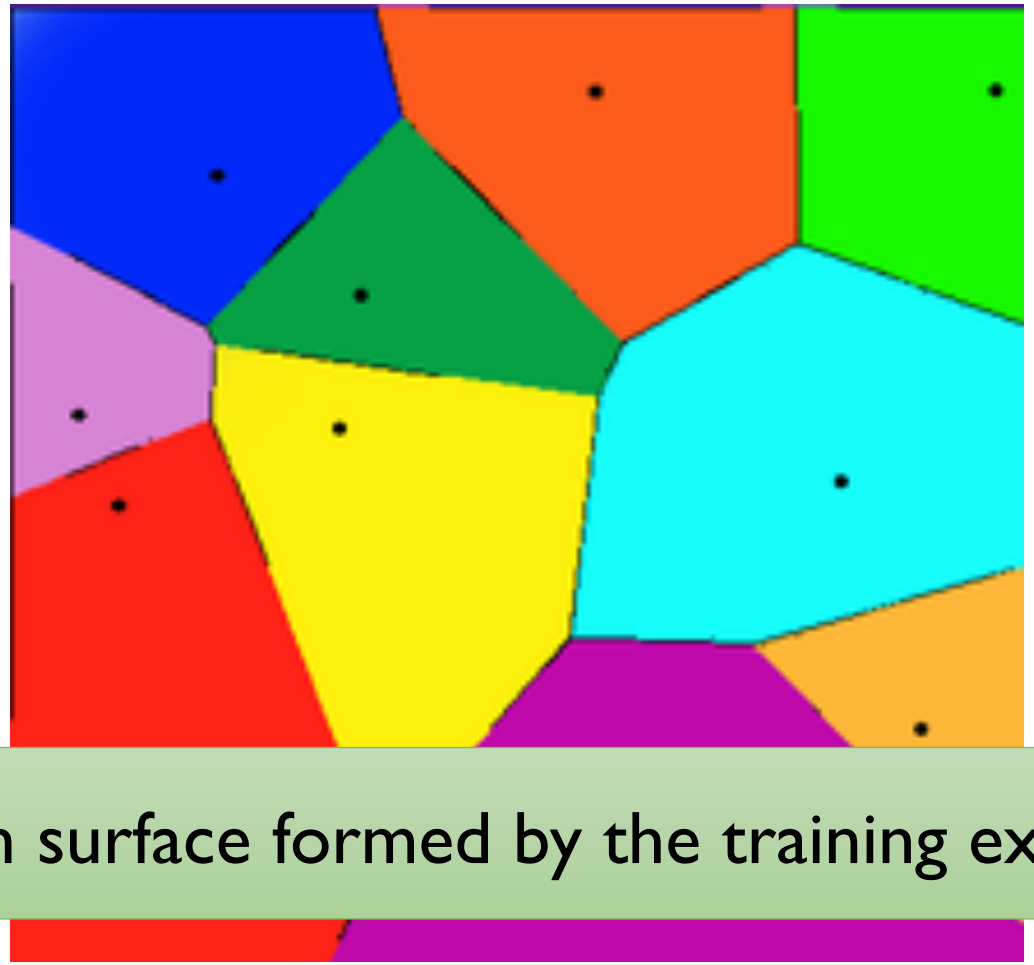
- Determined experimentally
  - Start with k=1 and use a test set to validate the error rate of the classifier
  - Repeat with k=k+2
  - Choose the value of k for which the error rate is minimum
  - Note: k typically an odd number to avoid ties in binary classification

k-Nearest Neighbors

- Eager Learning (Induction)
  - Explicit description of target function on the whole training set
- Instance-based Learning (Transduction)
  - Learning=storing all training instances
  - Classification=assigning target function to a new instance
  - Referred to as "Lazy" learning

Similar Keywords: K-Nearest Neighbors, Memory-Based Reasoning, Example-Based Reasoning, Instance-Based Learning, Case-Based Reasoning, Lazy Learning

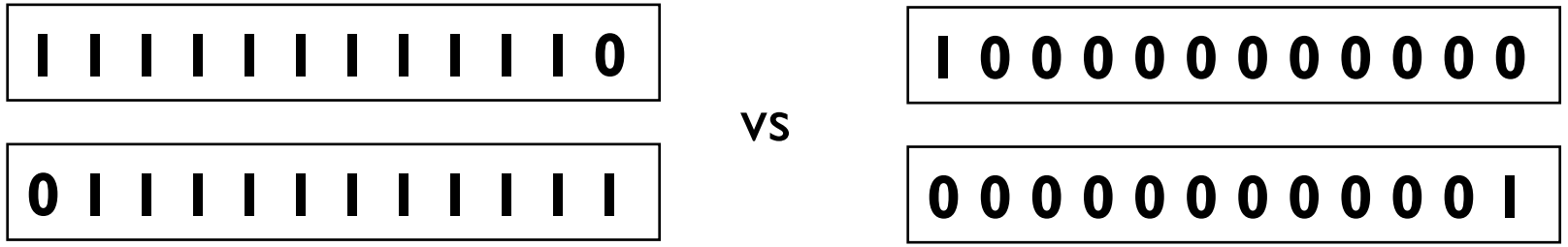
Voronoi Diagram



Decision surface formed by the training examples!

Improvements

- Distance-Weighted Nearest Neighbors
  - Assign weights to the neighbors based on their 'distance' from the query point (E.g., weight 'may' be inverse square of the distances)
- Scaling attributes for fair computation of distances



Solution: Normalize the vectors to unit length

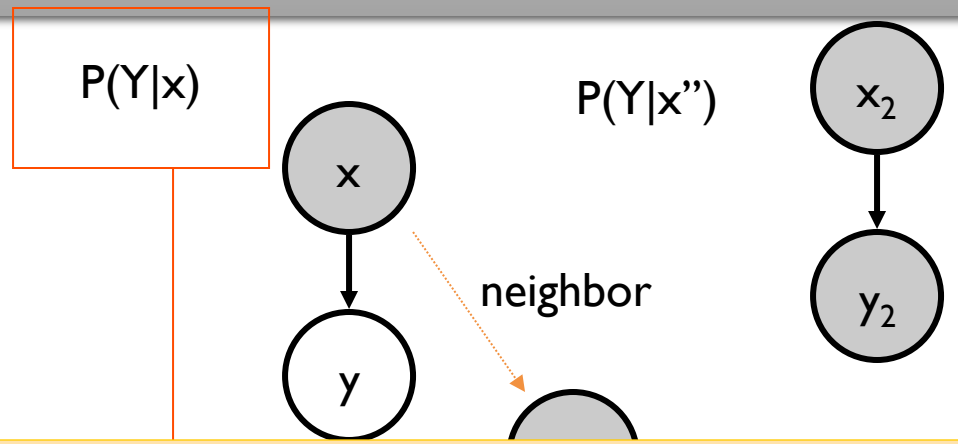
- Measure "closeness" differently
- Finding "close" examples in a large training set quickly
  - E.g. Efficient memory indexing using kd-trees

k-NN: Summary

- Pros
  - Highly effective inductive inference method for noisy training data and complex target functions
  - Target function for a whole space may be described as a combination of less complex local approximations
  - Trains very fast ("Lazy" learner)
- Cons
  - Curse of dimensionality
    - In higher dimensions, all the data points lie on the surface of the unit hypersphere (the inside is empty!) Check: http://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/chap1-high-dim-space.pdf
  - Storage: all training examples are saved in memory
    - A decision tree or linear classifier is much smaller
  - Slow at query time
    - Can be overcome and presorting and indexing training samples

Convergence of 1-NN

$$P(\text{knnError})$$
$$= 1 - \Pr(y = y_1)$$
$$= 1 - \sum \Pr(Y = y' | x)^2$$



Possible to show that: as the size of training data set approaches infinity, the one nearest neighbor classifier guarantees an error rate of no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). We will see this later.

k-Nearest Neighbors

- **Parametric** (single global model), **semiparametric** (small number of local models)
- **Nonparametric**: Similar inputs have similar outputs
- Functions (pdf, discriminant, regression) change smoothly
- Keep the training data; "let the data speak for itself"
- Given x, find a small number of closest training instances and interpolate from these

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- Given the training set X drawn i.i.d from p(x)
- Divide data into bins of size h
- Histogram:

$$\hat{p}(x) = \frac{\#\left\{x^t \text{ in the same bin as } x\right\}}{Nh}$$

- Naive estimator:

$$\hat{p}(x) = \frac{\#\left\{x - h < x^t \le x + h\right\}}{2Nh}$$

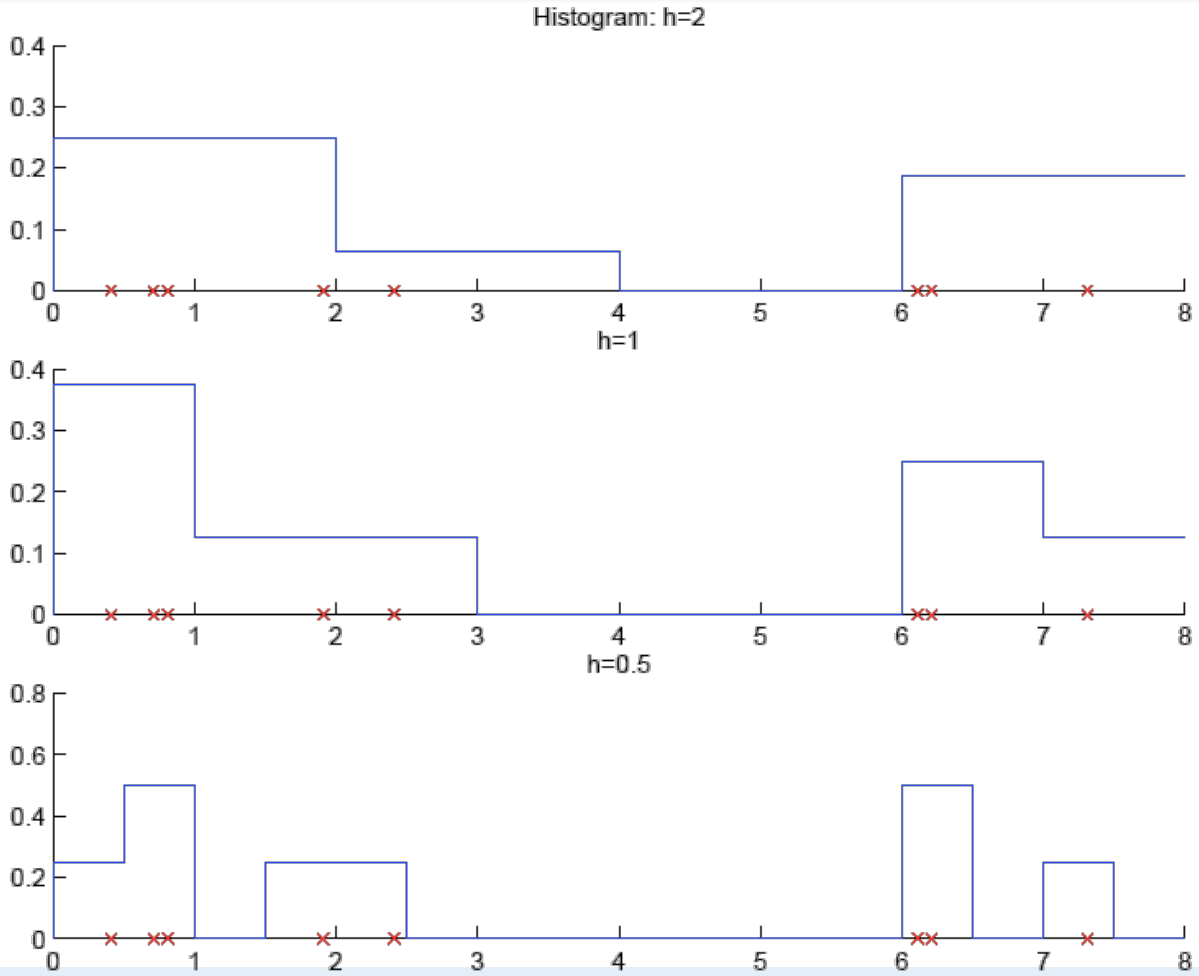
or

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} w\left(\frac{x - x^t}{h}\right) \quad w(u) = \begin{cases} 1 & \text{if } |u| < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

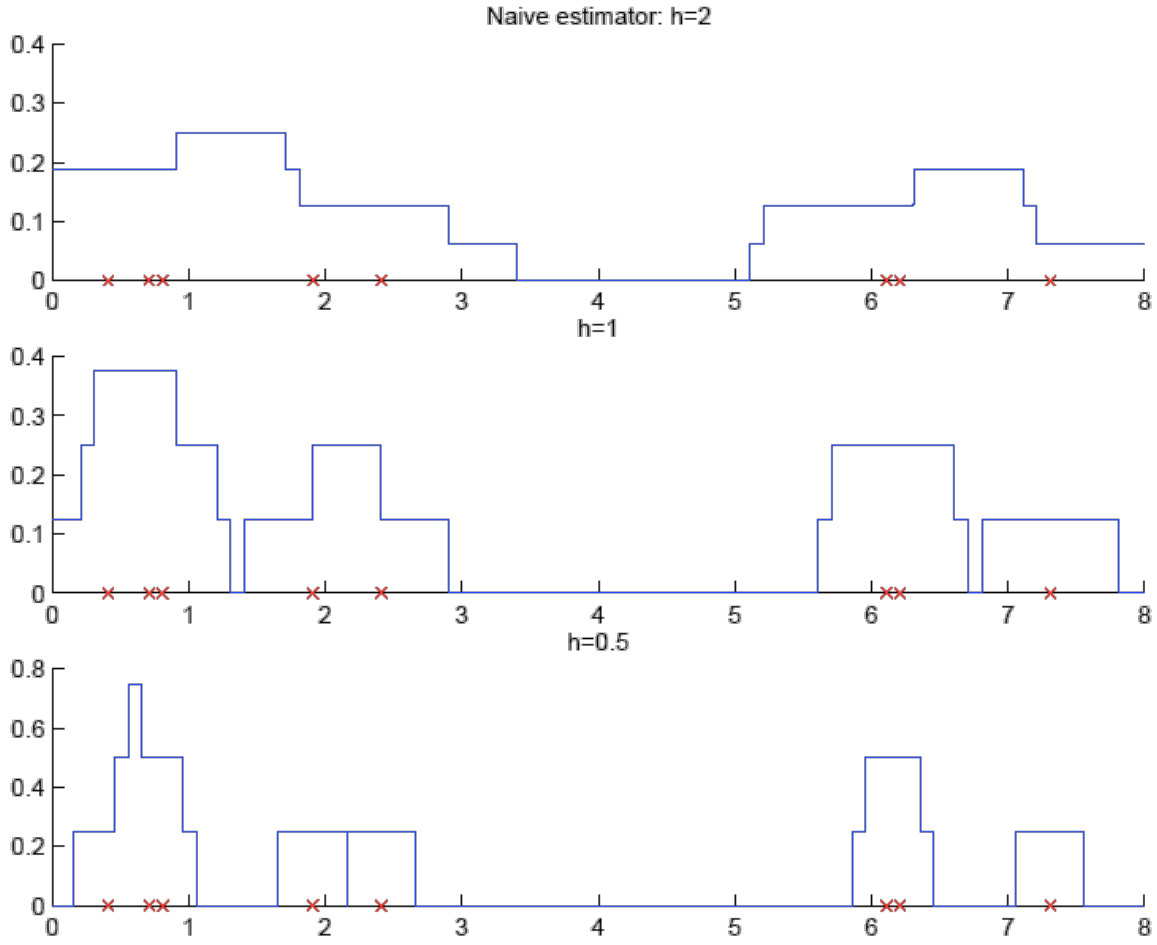
- Histogram Estimation



Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- Naïve Estimation



Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- Kernel function, e.g., Gaussian kernel:

$$K(u)=\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{u^2}{2}\right]$$

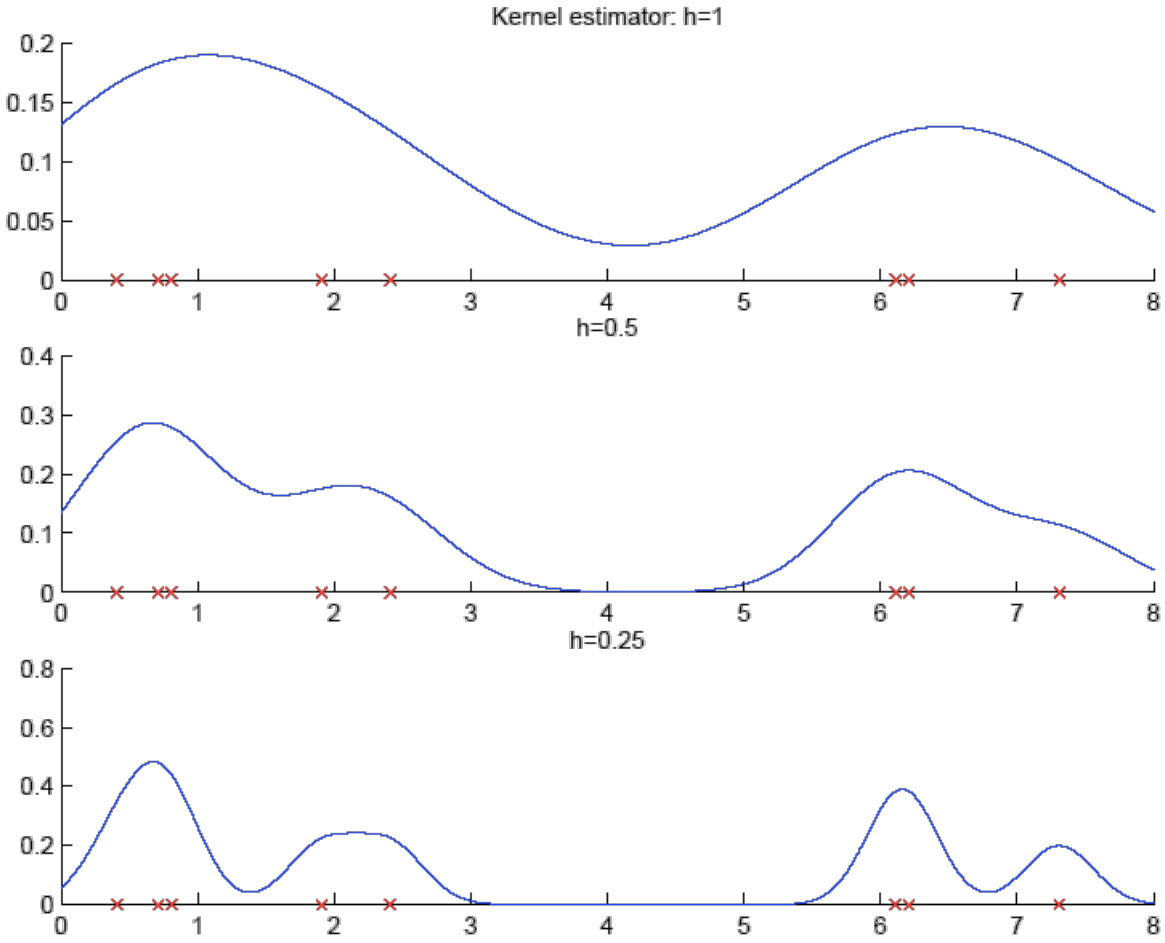
- Kernel estimator (**Parzen windows**)

$$\hat{p}(x)=\frac{1}{Nh}\sum_{t=1}^{N}K\left(\frac{x-x^t}{h}\right)$$

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- Kernel Estimation



Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- **K-Nearest Neighbor estimator**
- Instead of fixing bin width h and counting the number of instances, fix the instances (neighbors) k and check bin width

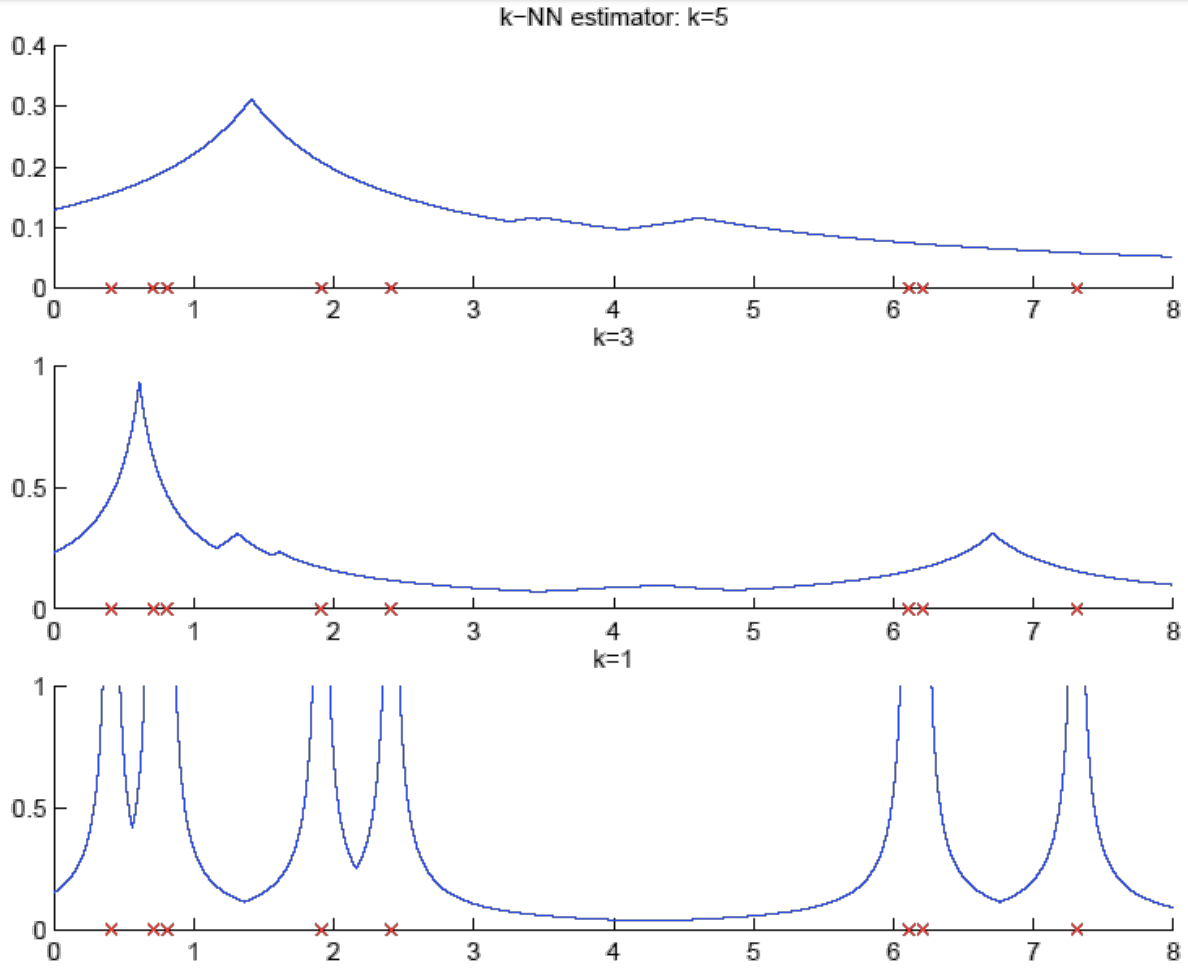
$$\hat{p}(x) = \frac{k}{2Nd_k(x)}$$

$d_k(x)$, distance to kth closest instance to x

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Non-parametric Density Estimation

- k-NN Estimation



Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Classification Methods

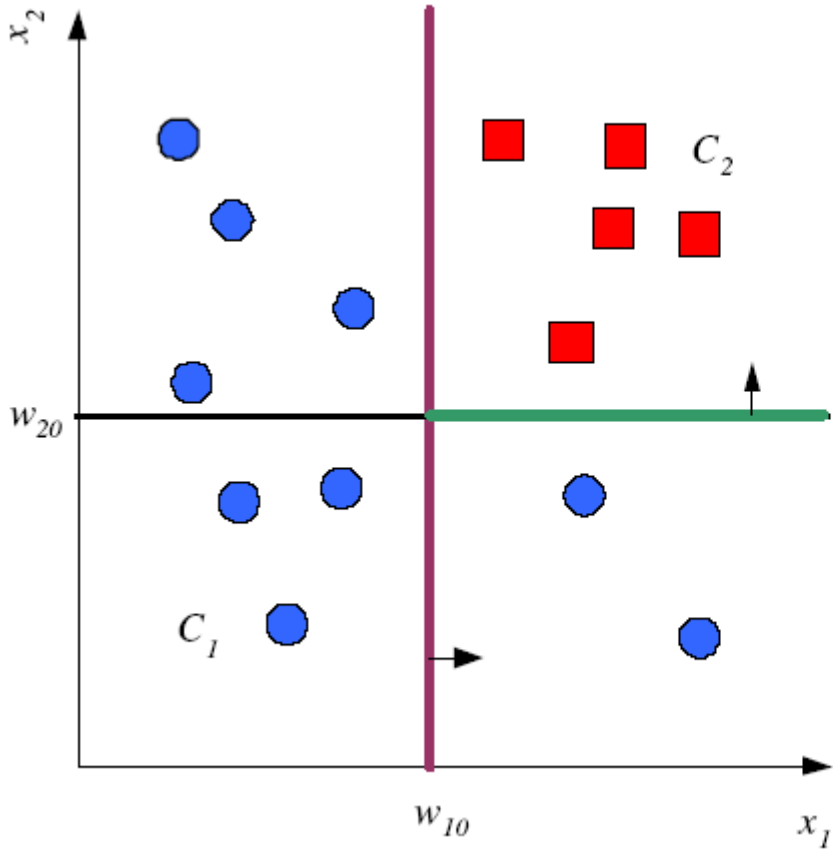
- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

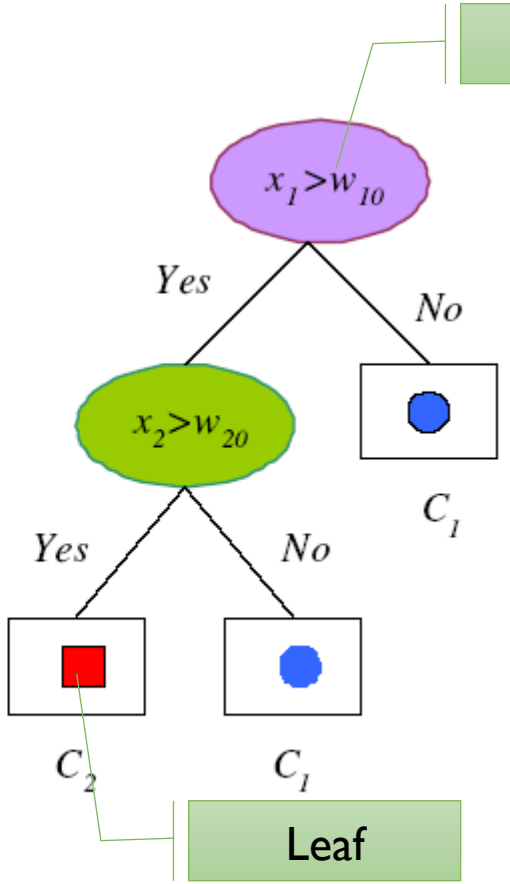
Example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Trees





- An efficient nonparametric method
- A hierarchical model
- Divide-and–conquer strategy

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Divide and Conquer

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i :
      - Binary split : $x_i > w_m$
    - Discrete x_i :
      - n-way split for n possible values
  - Multivariate: Uses more than one attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is greedy; find the best split recursively

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Classification Trees (C4.5, J48)

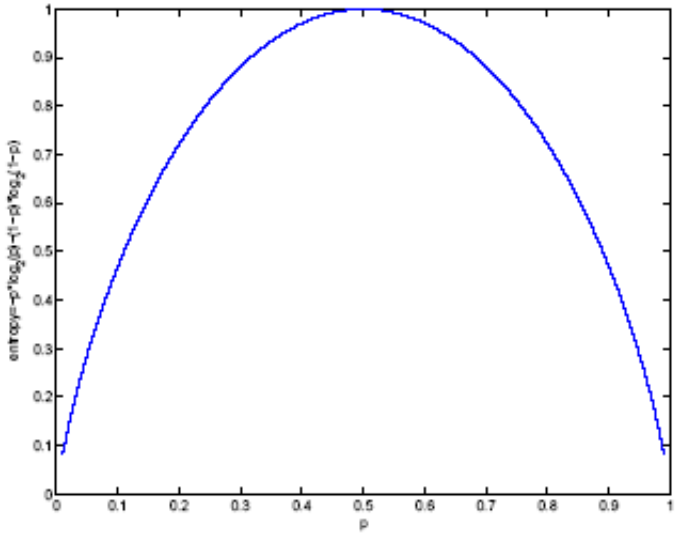
- For node m, N_m instances reach m, N^i_m belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

- Node m is **pure** if p^i_m is 0 or 1
- Measure of **impurity** is **entropy**

$$I_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$

2-class problem



Entropy in information theory specifies the average (expected) amount of information derived from observing an event

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Classification Trees

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- **Impurity after split:** N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}}$$

$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- **Information Gain:** Expected reduction in impurity measure after split
- Choose the *best* attribute(s) (**with maximum information gain**) to split the remaining instances and make that attribute a decision node
  - You can use same logic to find best splitting value too

Source: Ethem Alpaydin, Introduction to Machine Learning, 3rd Edition (Slides)

Other Measures of Impurity

- The properties of functions measuring the **impurity** of a split:
  - $\phi(1/2,1/2) \geq \phi(p,1-p),$ for any $p \in [0,1]$
  - $\phi(0,1) = \phi(1,0) = 0$
  - $\phi(p,1-p)$ is increasing in p on $[0,\frac{1}{2}]$

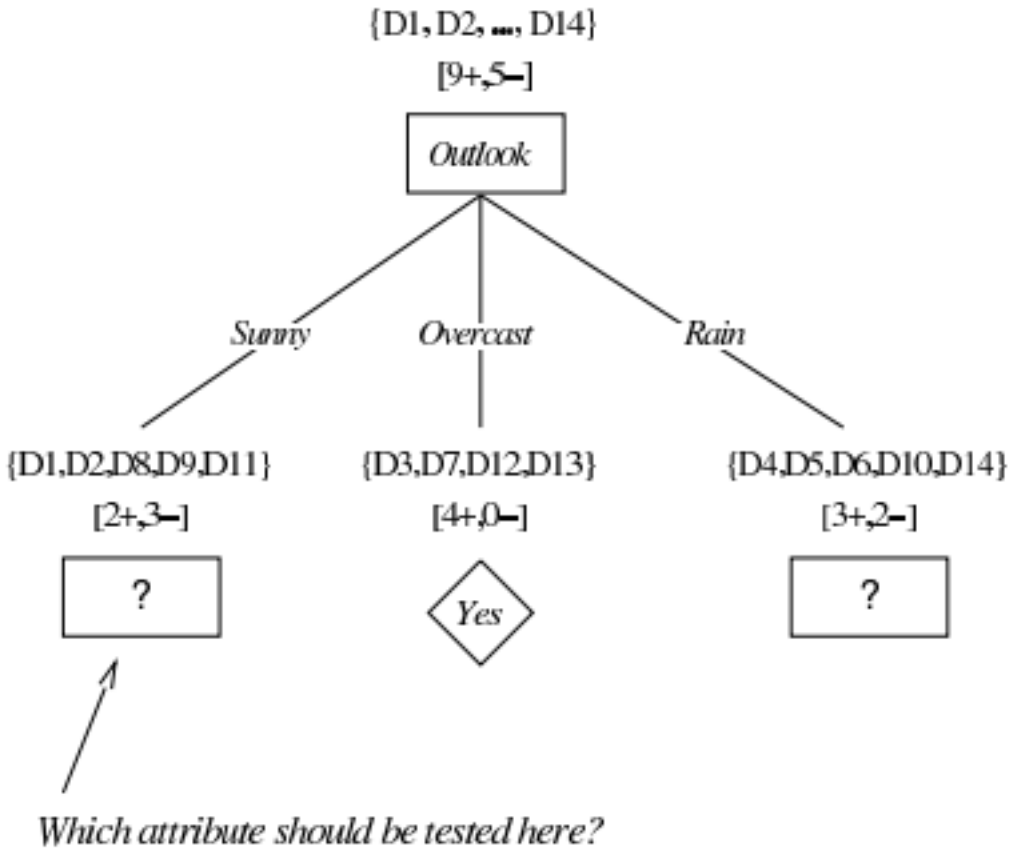
    and decreasing in p on $[\frac{1}{2},1]$
- Examples (other than entropy)
  - **Gini impurity/inde** $1 - \sum_{j=1}^{c} p_j^2$

Decision Trees: Example

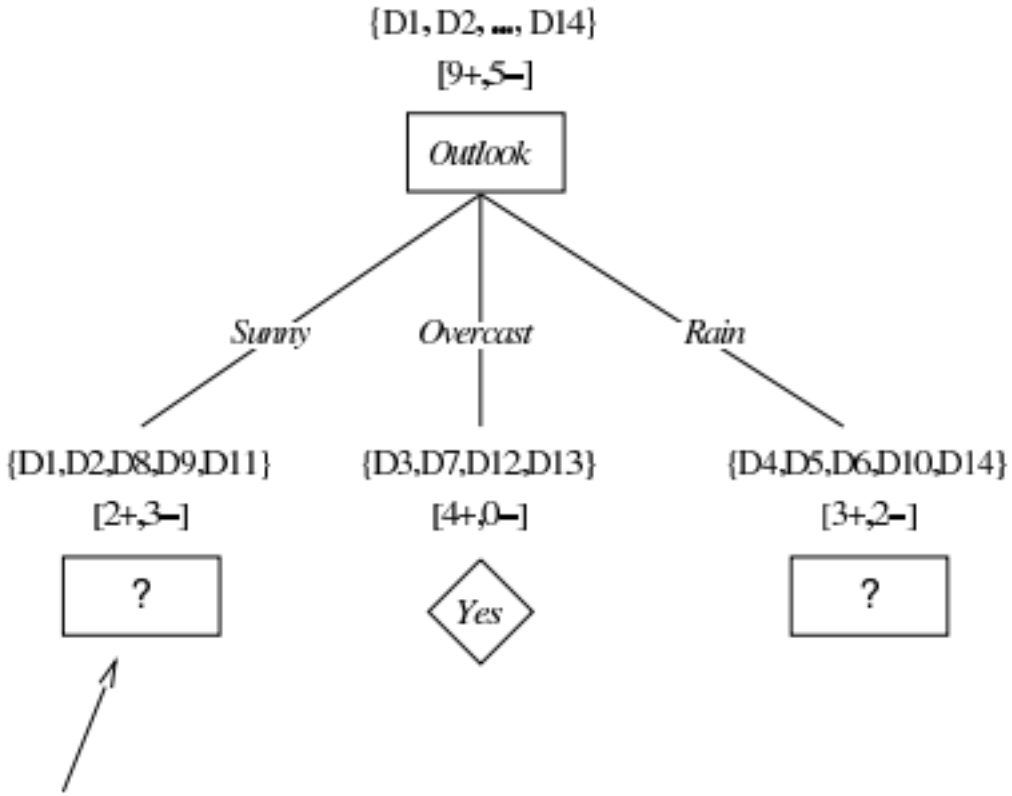
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Trees: Example



Decision Trees: Example



Which attribute should be tested here?

$S_{sunny} = \{D1,D2,D8,D9,D11\}$

$Gain(S_{sunny}, Humidity) = .970 - (3/5)\,0.0 - (2/5)\,0.0 = .970$

$Gain(S_{sunny}, Temperature) = .970 - (2/5)\,0.0 - (2/5)\,1.0 - (1/5)\,0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\,1.0 - (3/5)\,.918 = .019$

Overfitting and Generalization

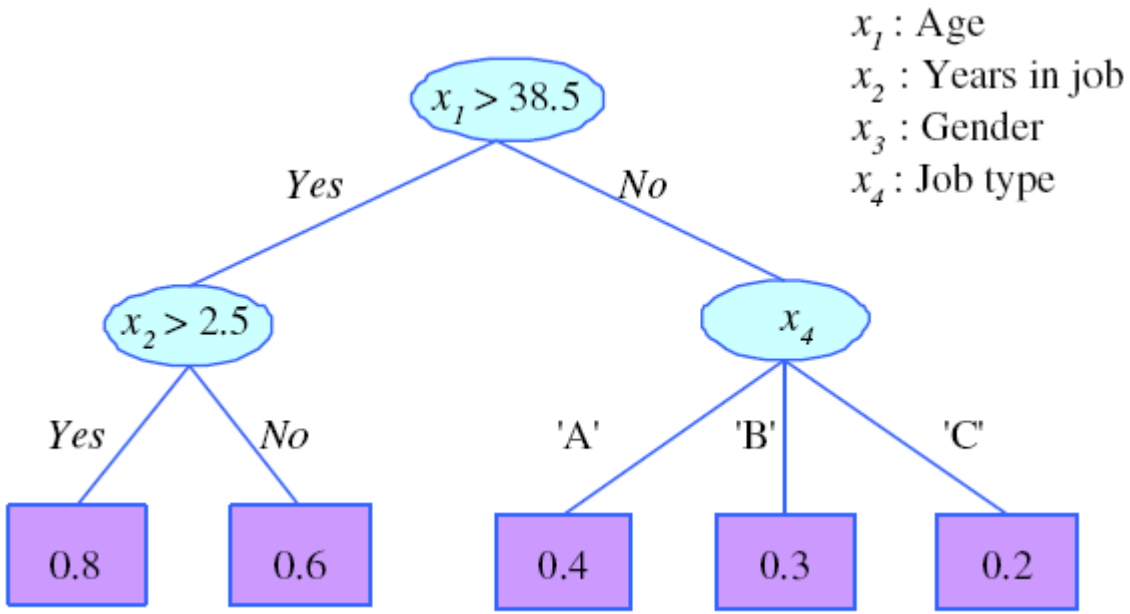
- Overfitting can occur with noisy training examples, also when small numbers of examples are associated with leaf nodes. How to handle?
- **Pruning:** Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees which overfit on the pruning set
  - Prepruning is faster, postpruning is more accurate

Overfitting and Generalization

- **Occam's Razor principle:** when multiple hypotheses can solve a problem, choose the simplest one
  - a short hypothesis that fits data unlikely to be coincidence
  - a long hypothesis that fits data might be coincidence

- How to select "best" tree:
  - Measure performance over training data
  - Measure performance over separate validation data set
  - **Minimum Description Length:** Minimize *size(tree) + size(misclassifications(tree))*

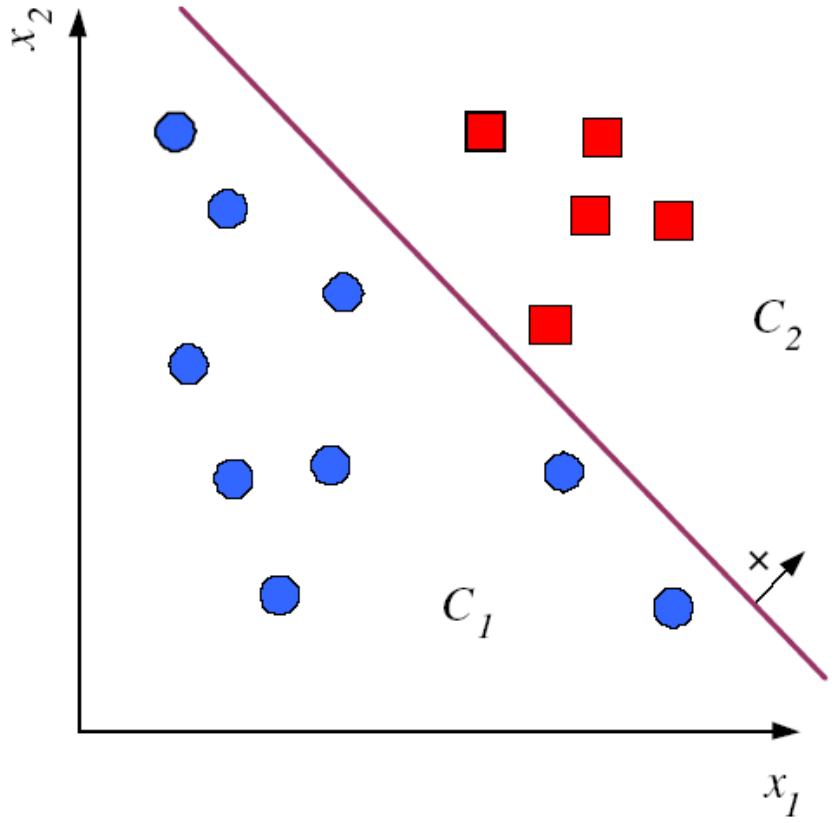
Rule Extraction from Trees

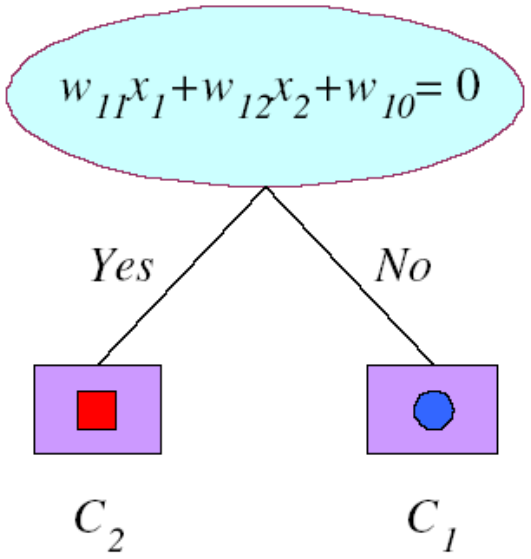


R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

- Convert tree to equivalent set of rules
- Prune each rule independently of others, by removing any preconditions that result in improving its estimated accuracy
- Sort final rules into desired sequence for use

Multivariate Trees





Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

Probability: Review

Random variable	• Result of tossing a coin is from {Heads, Tails} • Random var X from {1,0} • Bernoulli: $P\{X=1\} = p_o^X (1 - p_o)^{(1-X)}$
Joint and conditional probability	$P(A\|B) = P(A, B)/P(B)$
Bayes Theorem	$P(A\|B) = P(B\|A)\ P(A)/P(B)$

Illustration

A	0	0	1	1	1	0
B	0	1	1	0	1	1

- P(A=1) = 3/6 = 1/2, P(A=0) = 3/6 = 1/2.
- P(B=1) = 4/6 = 2/3, P(B=0) = 2/6 = 1/3.
- P(A=1, B = 1) = 2/6 = 1/3.
- P(A=1 | B = 1) = P(A=1, B = 1) / P(B=1) = 1/2.
- P(B=1 | A = 1) = P(B=1, A = 1) / P(A=1) = 2/3.
- P(A=1 | B = 1) P(B=1)/P(A=1) = 2/3 = P(B=1 | A =1).
  - Bayes' Theorem

Naïve Bayes Classifier

- Goal: Learning function f: x -> y
  - *Y*: One of *k* classes (e.g. spam/ham, digit 0-9)
  - $X=X_1,\ldots,X_n$: Values of attributes (numeric or categorical)
- Probabilistic classification
  - Most probable class given observation: $\hat{y} = \arg\max_y P(y|x)$
- Bayesian probability of a class

$$P(y|x) = \frac{\overbrace{P(x|y)}^{\text{class model}}\overbrace{P(y)}^{\text{prior}}}{\underbrace{\sum_{y'} P(x|y')P(y')}_{\text{normalizer } P(x)}}$$

Bayes Theorem

Bayes Theorem: Example

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20

- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

What is Naïve about it?

- Consider each attribute and class label as random variables
- Given a record with attributes $(A_1, A_2, \ldots, A_n)$
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes $P(C| A_1, A_2, \ldots, A_n)$
- Can we estimate $P(C| A_1, A_2, \ldots, A_n)$ directly from data?

What is Naïve about it?

- Approach:
  - compute the posterior probability $P(C \mid A_1, A_2, \ldots, A_n)$ for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    $P(C \mid A_1, A_2, \ldots, A_n)$
  - Equivalent to choosing value of C that maximizes
    $P(A_1, A_2, \ldots, A_n \mid C)\ P(C)$

Maximum A Posteriori (MAP) Rule

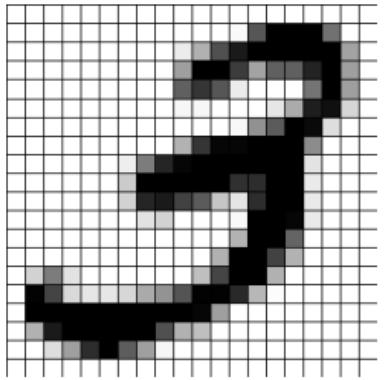
- How to estimate $P(A_1, A_2, \ldots, A_n \mid C)$?

What is Naïve about it?

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

How to compute if x is made of multiple attributes?

- 20 x 20 image of digit = 2^{400} possible combinations!



Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C_j) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - New point is classified to C_j if
    $P(C_j) \prod_j P(A_i \mid C_j) = P(C_j)\, P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
    is maximal

Example

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example: Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=No
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

P(Play=*Yes*) = 9/14 *P*(Play=*No*) = 5/14

Example: Test Phase

- Given a new instance,
- $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)
- Look up tables

P(Outlook=*Sunny* | Play=*Yes*) = 2/9
P(Temperature=*Cool* | Play=*Yes*) = 3/9
P(Huminity=*High* | Play=*Yes*) = 3/9
P(Wind=*Strong* | Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=S*unny* | Play=*No*) = 3/5
P(Temperature=*Cool* | Play==*No*) = 1/5
P(Huminity=*High* | Play=*No*) = 4/5
P(Wind=*Strong* | Play=*No*) = 3/5
P(Play=*No*) = 5/14

- MAP rule

P(*Yes* | $\mathbf{x}'$): [P(*Sunny* | Yes)P(*Cool* | *Yes*)P(*High* | Yes)P(*Strong* | *Yes*)]P(Play=*Yes*) = 0.0053
P(*No* | $\mathbf{x}'$): [P(*Sunny* | No) P(*Cool* | No)P(*High* | *No*)P(*Strong* | *No*)]P(Play=*No*) = 0.0206

Given the fact P(*Yes* | $\mathbf{x}'$) < P(*No* | $\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

Example: Another

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A \mid M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A \mid N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A \mid M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A \mid N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes Classifier: Practical Issues

- Robust to isolated noise points
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold always

Naïve Bayes Classifier: Practical Issues

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: (A < v) or (A > v)
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a parametrized distribution, e.g. normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation) using Maximum Likelihood Estimation
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

Naïve Bayes Classifier: Example

- Problem Definition
  - Given a set of news articles that are of interest, we would to like to learn to classify the articles by topic.
- Naïve Bayes is among the most effective algorithms to perform this task.
- What will be attributes to represent the documents?
  - Vector of words – one attribute per word position in the document
- What is the Target concept
  - Is the document interesting?
  - Topic of the document

Example: 20 Newsgroup

- Given 1000 training documents from each group
- Learn to classify new documents according to the newsgroup it came from
- NBC – 89% accuracy

comp.graphics	misc.forsale
comp.os.ms-windows.misc	rec.autos
comp.sys.ibm.pc.hardware	rec.motorcycles
comp.sys.mac.hardware	rec.sport.baseball
comp.windows.x	rec.sport.hockey
alt.atheism	sci.space
soc.religion.christian	sci.crypt
talk.religion.misc	sci.electronics
talk.politics.mideast	sci.med
talk.politics.misc	talk.politics.guns

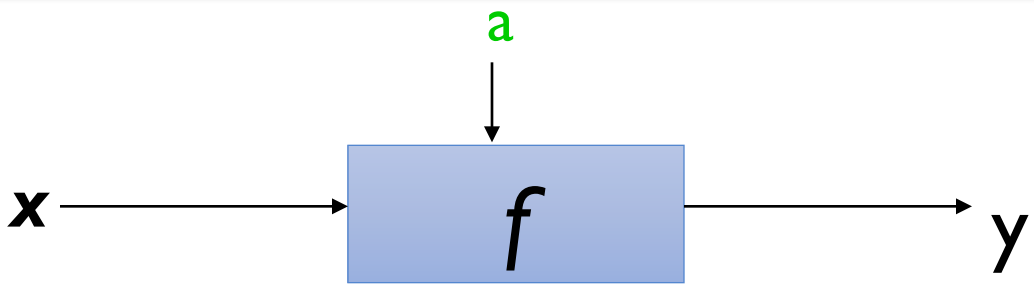
Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

SVM: History

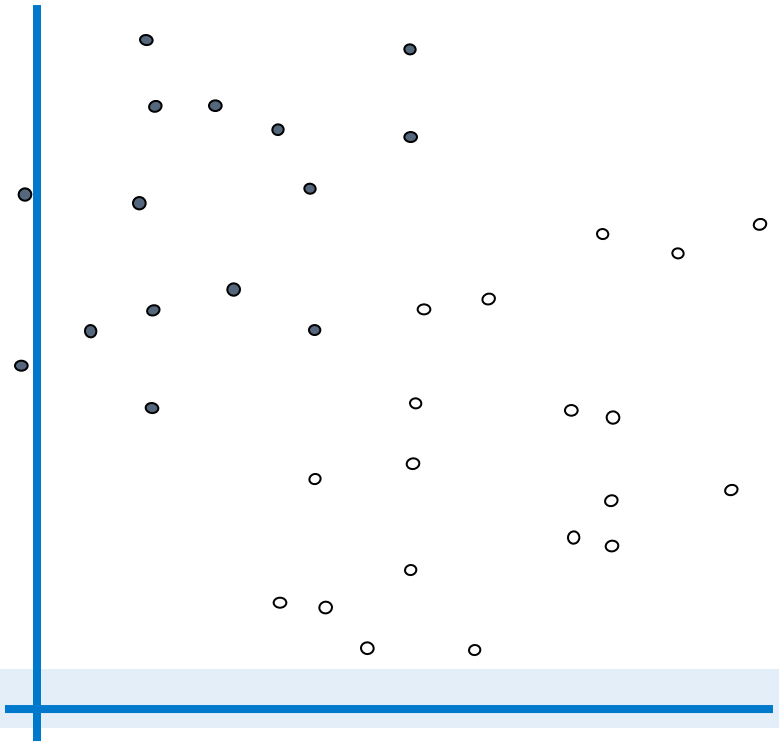
- SVM is inspired from statistical learning theory
- SVM was first introduced in 1992
- SVM becomes popular because of its success in handwritten digit recognition

Linear Classifiers



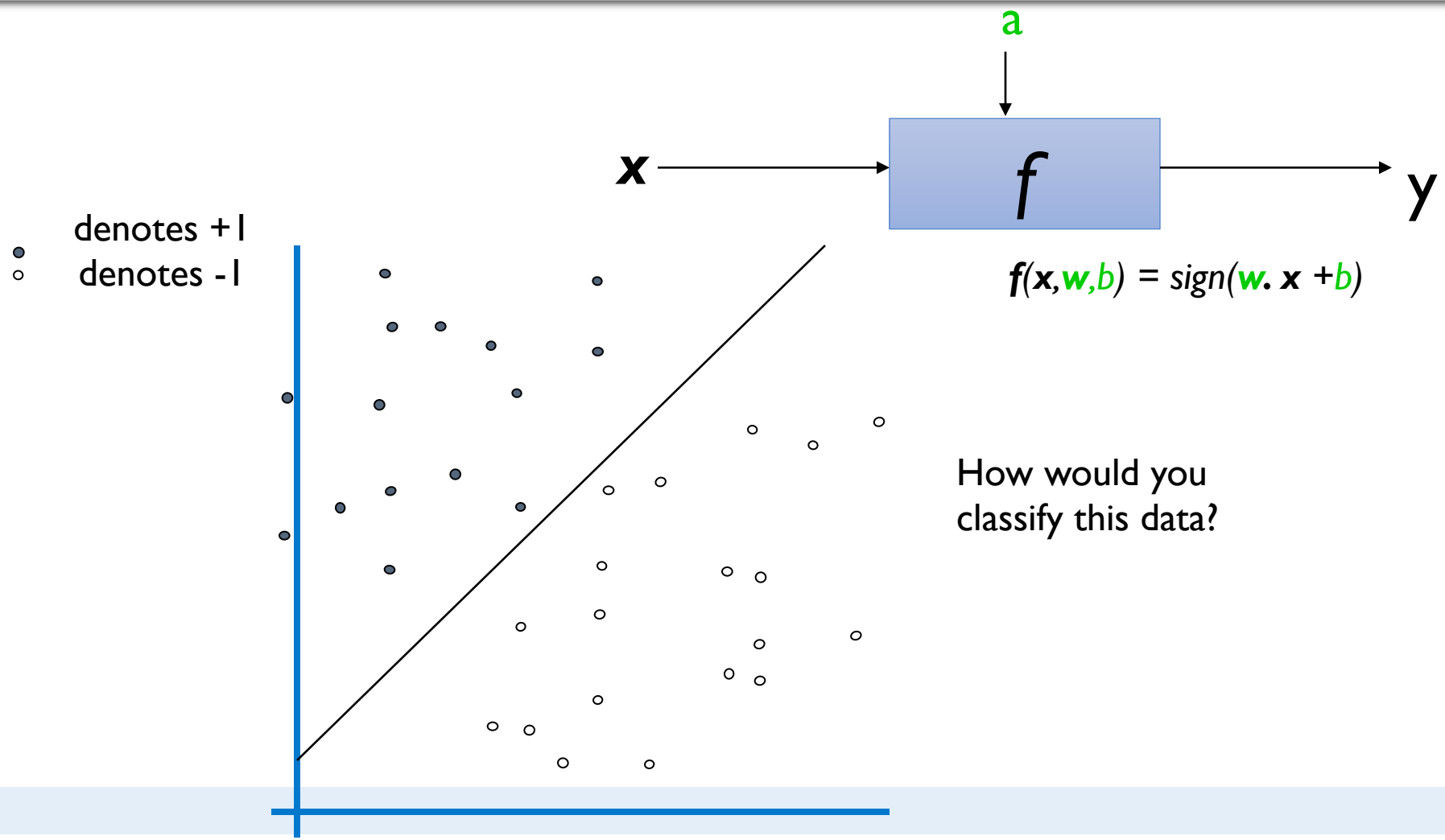
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\mathbf{x} + b)$

denotes +1
denotes -1

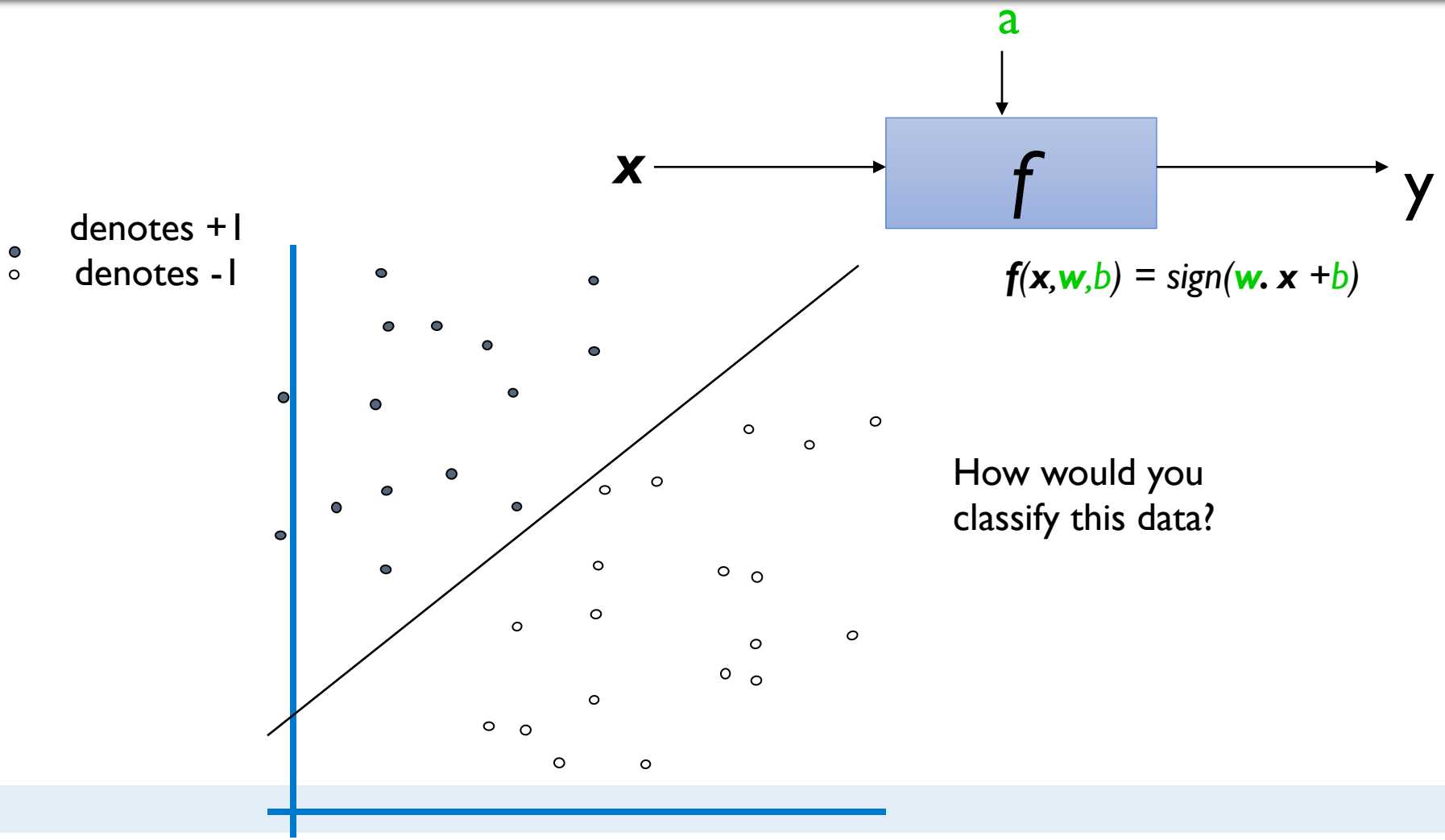


How would you classify this data?

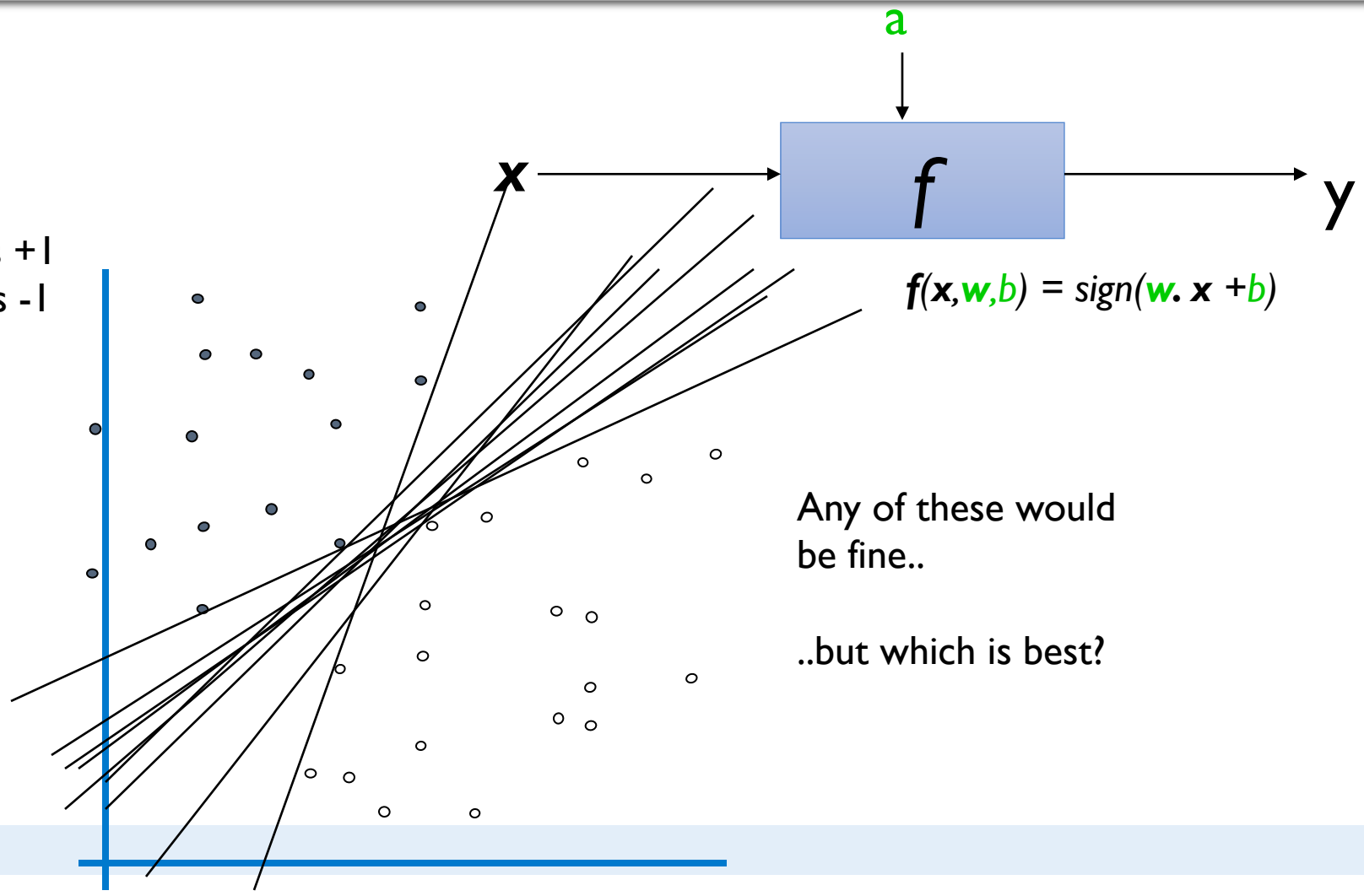
Linear Classifiers



Linear Classifiers

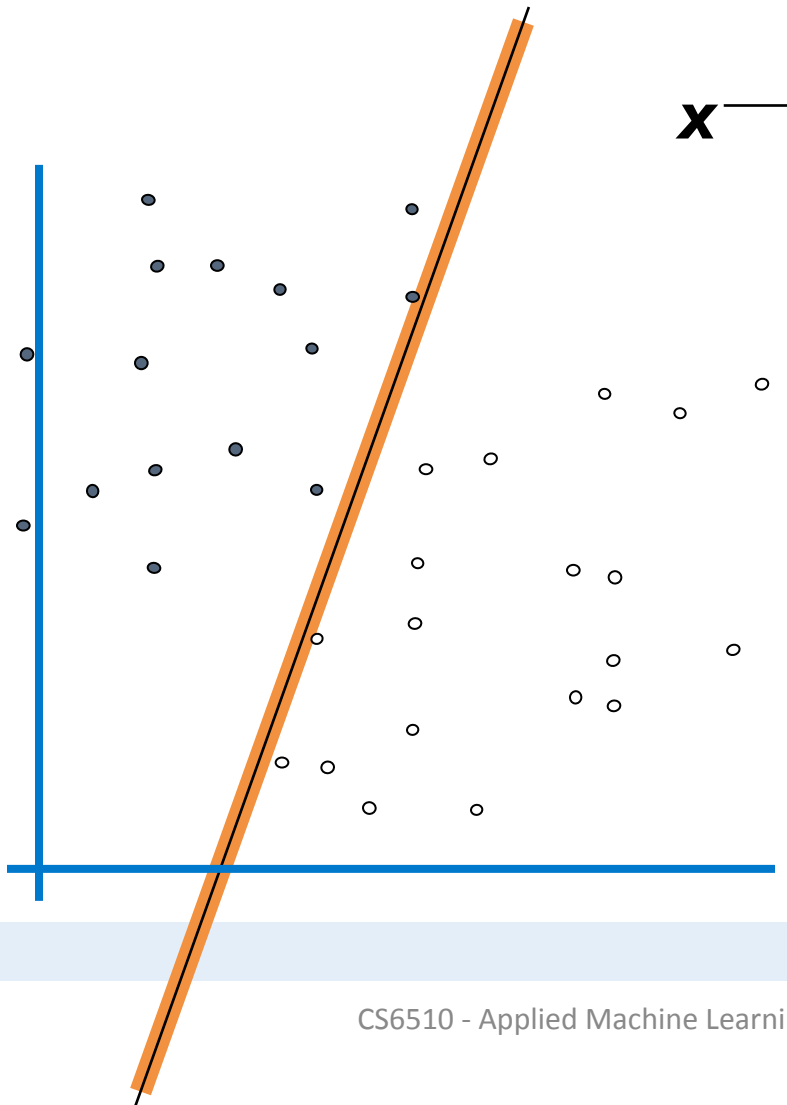


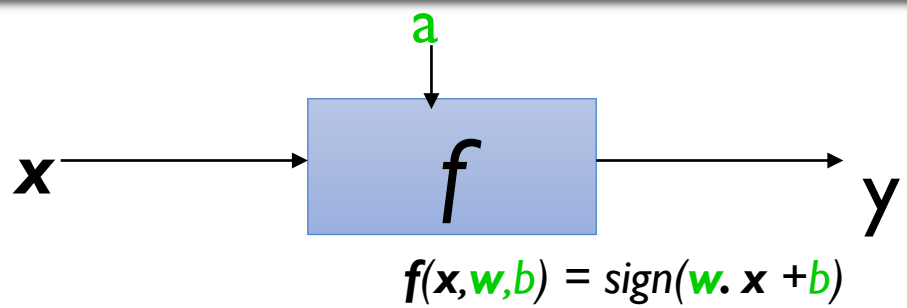
Linear Classifiers



Linear Classifiers

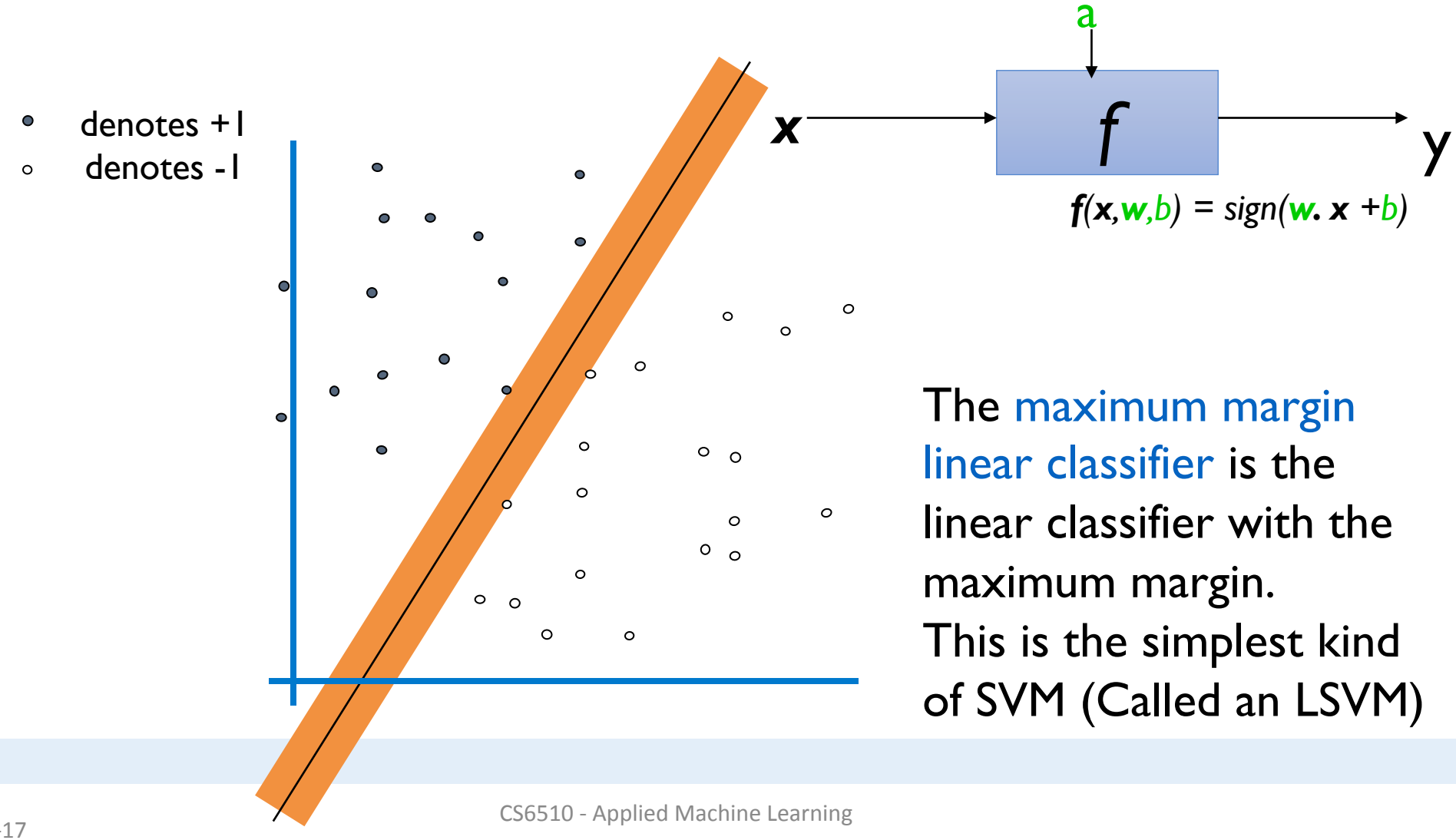
denotes +1
denotes -1





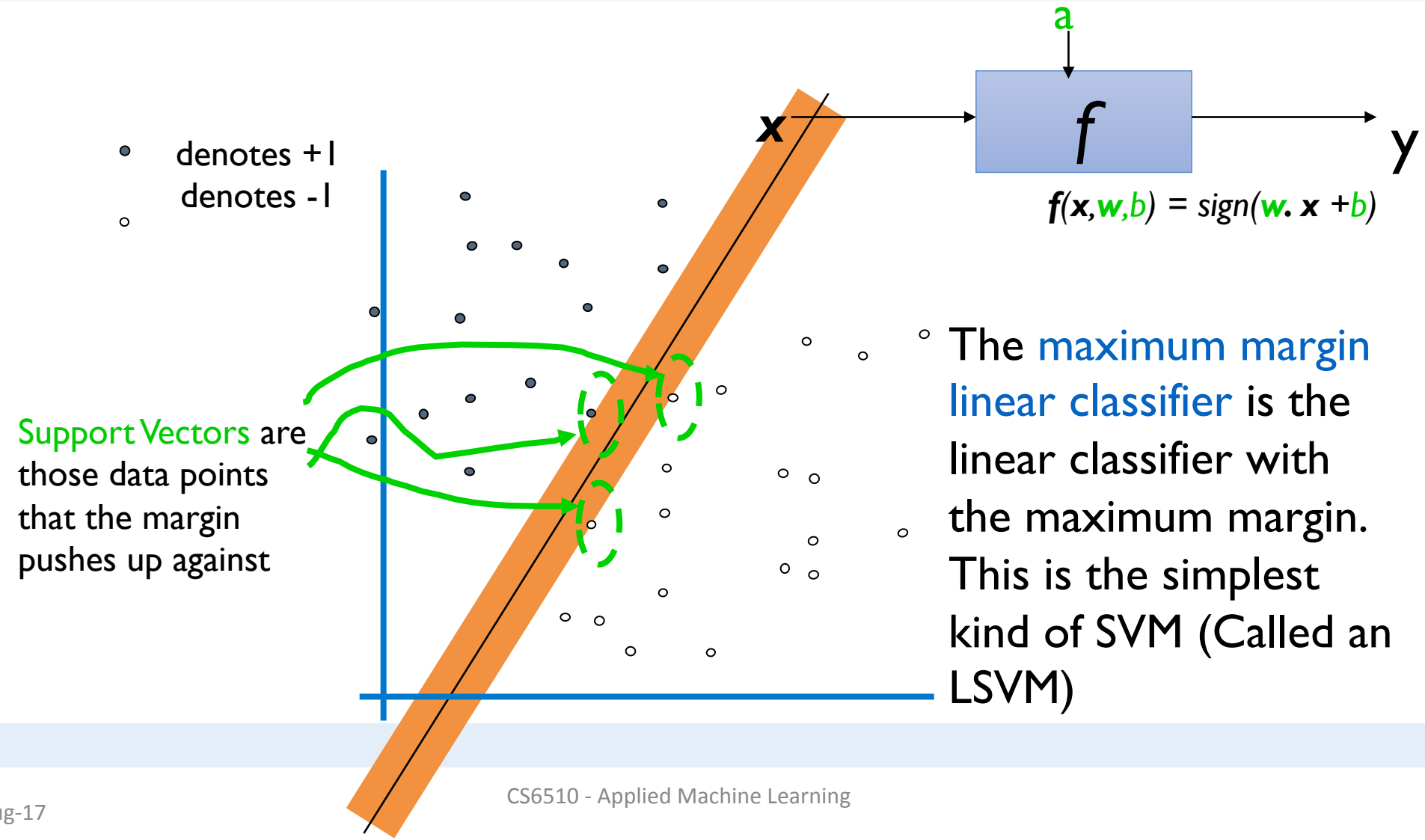
Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Linear Classifiers



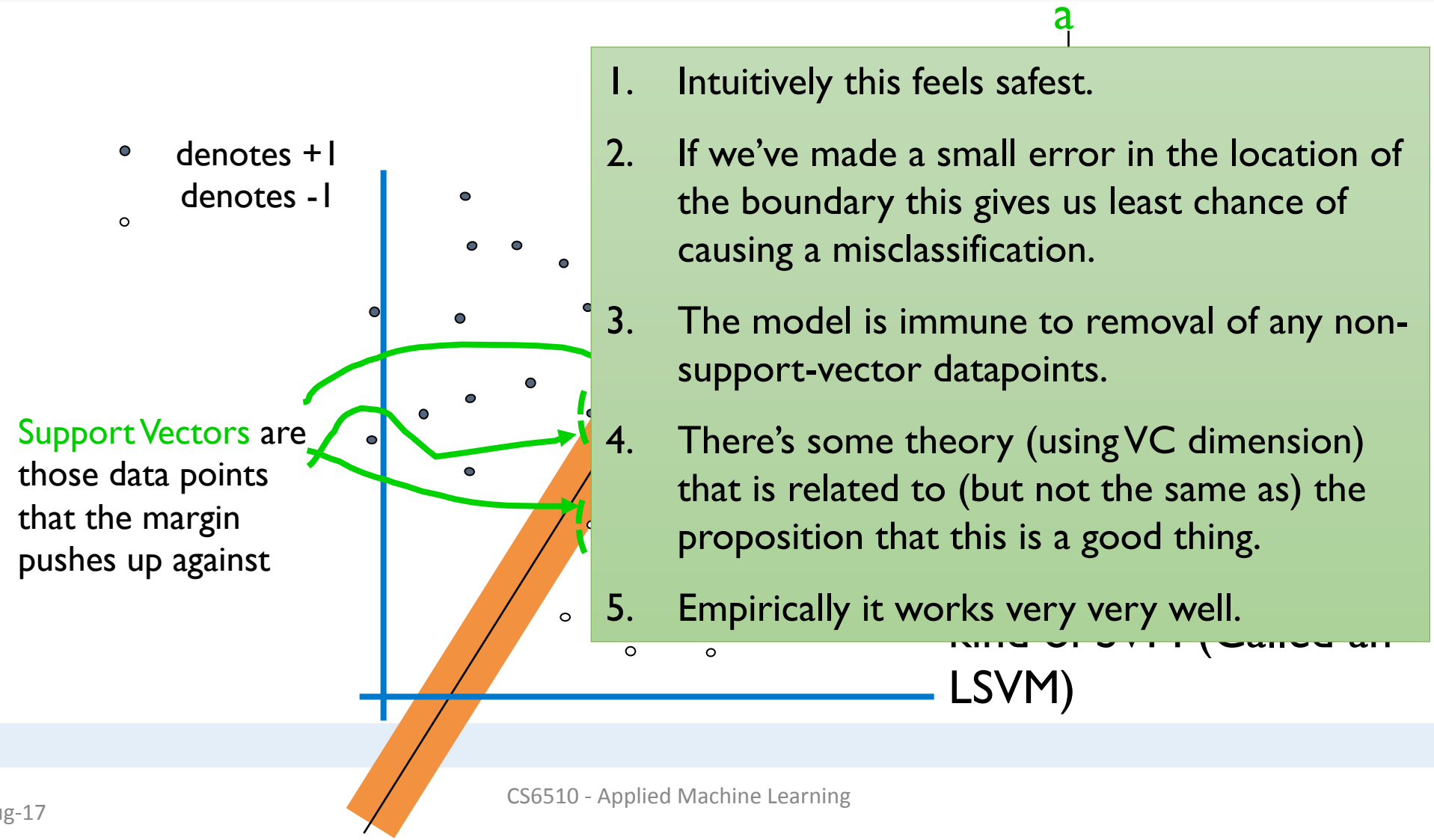
The maximum margin linear classifier is the linear classifier with the maximum margin.
This is the simplest kind of SVM (Called an LSVM)

Maximum Margin Classifier



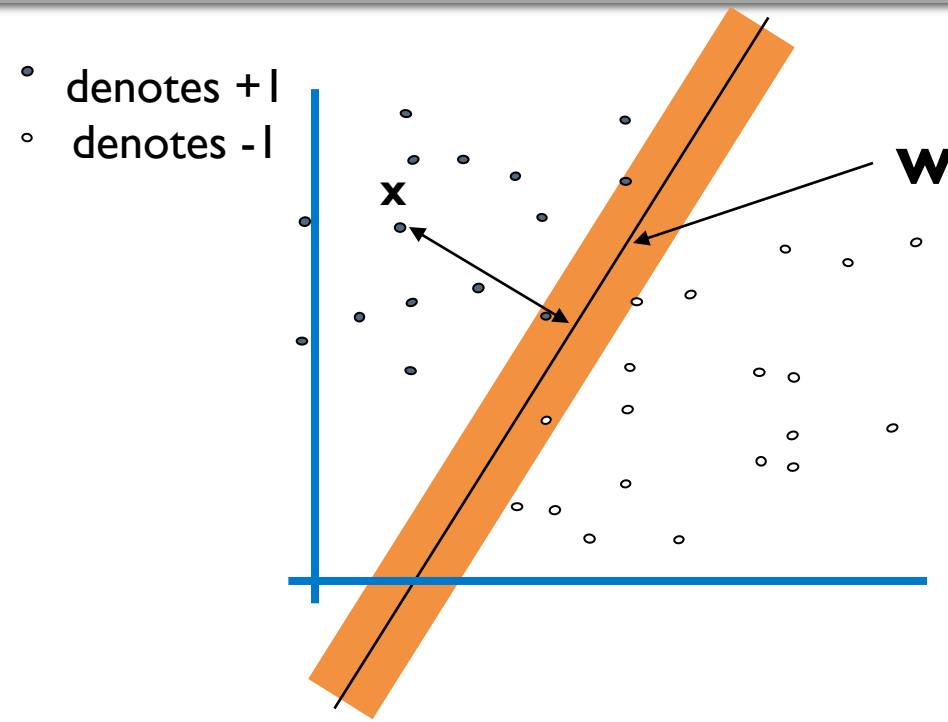
The maximum margin linear classifier is the linear classifier with the maximum margin. This is the simplest kind of SVM (Called an LSVM)

Why Maximum Margin?



1. Intuitively this feels safest.
2. If we've made a small error in the location of the boundary this gives us least chance of causing a misclassification.
3. The model is immune to removal of any non-support-vector datapoints.
4. There's some theory (using VC dimension) that is related to (but not the same as) the proposition that this is a good thing.
5. Empirically it works very very well.

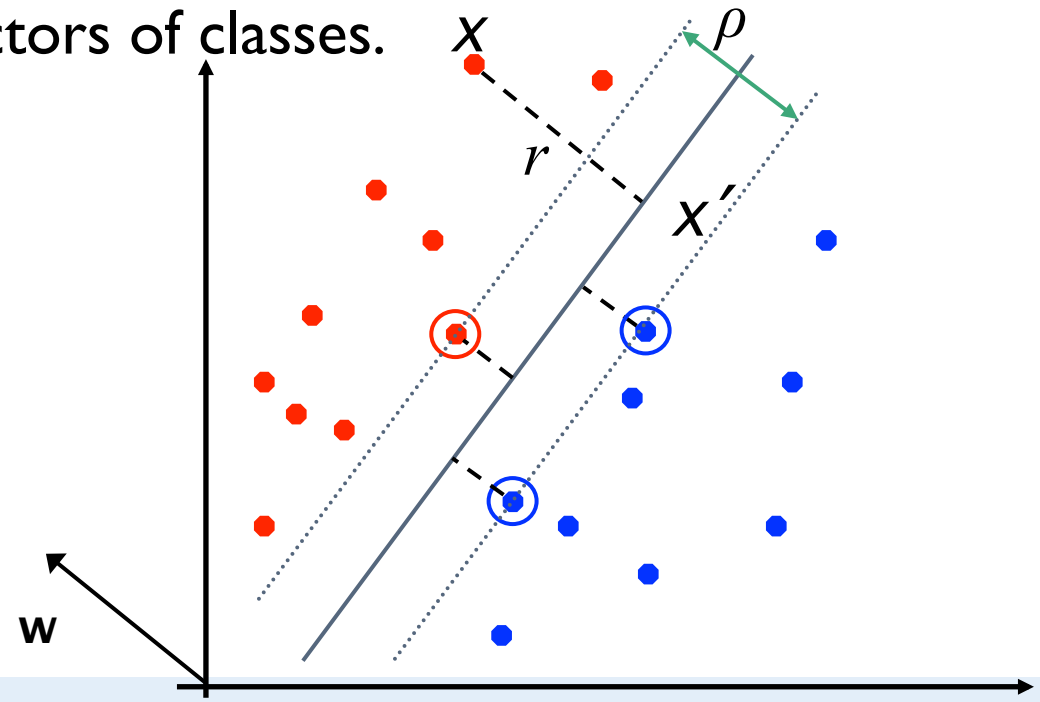
Estimating the Margin



- What is the distance expression for a point **x** to a line **wx**+b= 0?

Estimating the Margin

- 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.



Derivation of finding *r*:

Dotted line $\mathbf{x}' - \mathbf{x}$ is perpendicular to decision boundary so parallel to $\mathbf{w}$.

Unit vector is $\mathbf{w}/|\mathbf{w}|$, so line is $r\mathbf{w}/|\mathbf{w}|$.

$\mathbf{x}' = \mathbf{x} - yr\mathbf{w}/|\mathbf{w}|$.

$\mathbf{x}'$ satisfies $\mathbf{w}^T\mathbf{x}' + b = 0$.

So $\mathbf{w}^T(\mathbf{x} - yr\mathbf{w}/|\mathbf{w}|) + b = 0$

Recall that $|\mathbf{w}| = \sqrt{\mathbf{w}^T\mathbf{w}}$.

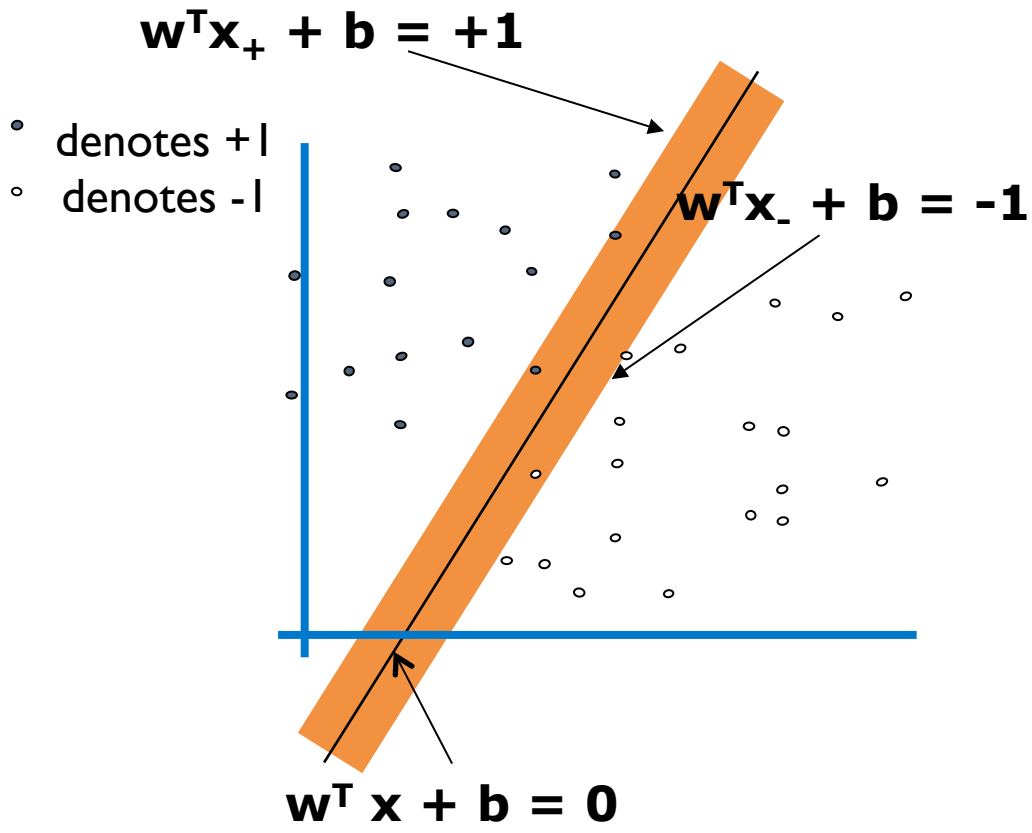
So $\mathbf{w}^T\mathbf{x} - yr|\mathbf{w}| + b = 0$

So, solving for r gives:

$r = y(\mathbf{w}^T\mathbf{x} + b)/|\mathbf{w}|$

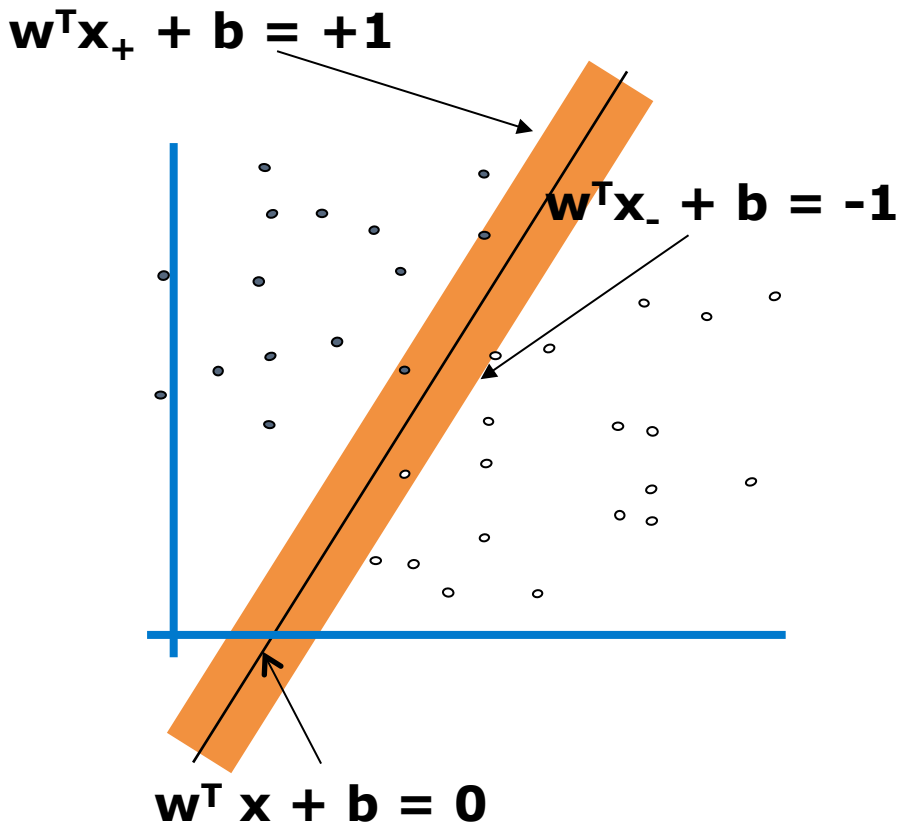
Estimating the Margin

- Since $\mathbf{w}^T\mathbf{x} + \mathbf{b} = 0$ and $c(\mathbf{w}^T\mathbf{x} + \mathbf{b}) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$ (i.e. c)
- Let us choose normalization such that $\mathbf{w}^T\mathbf{x}_+ + \mathbf{b} = +1$ and $\mathbf{w}^T\mathbf{x}_- + \mathbf{b} = -1$ for the positive and negative support vectors respectively

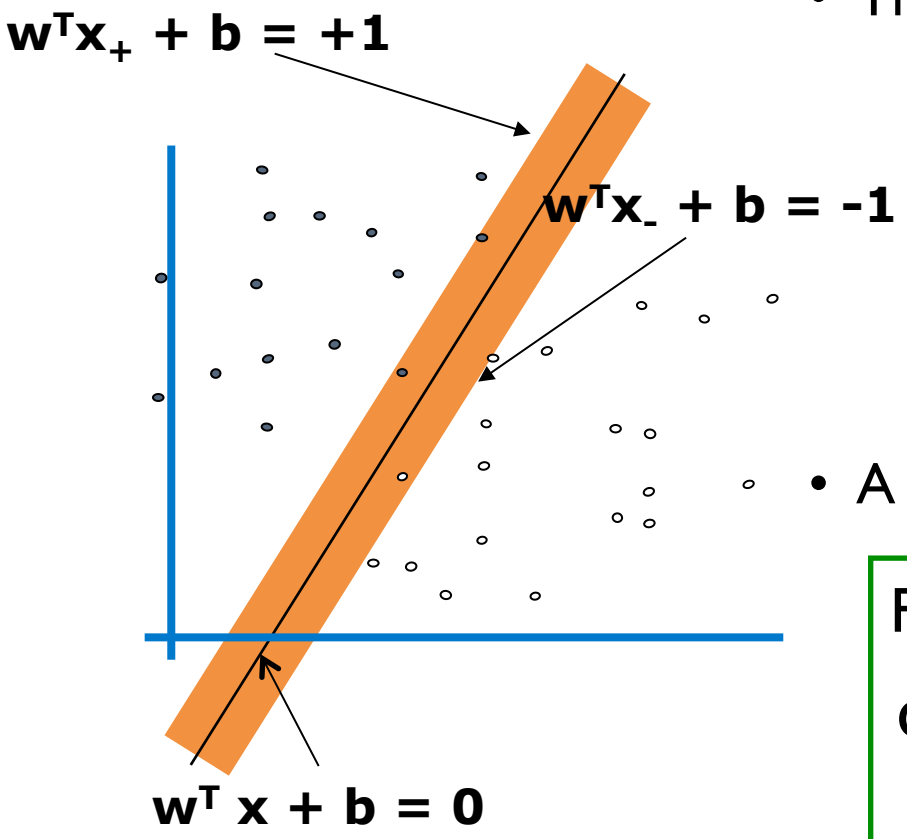


Estimating the Margin

- Since $\mathbf{w}^T\mathbf{x} + \mathbf{b} = 0$ and $c(\mathbf{w}^T\mathbf{x} + \mathbf{b}) = 0$ define the same plane, we have the freedom to choose the normalization of $\mathbf{w}$ (i.e. c)
- Let us choose normalization such that $\mathbf{w}^T\mathbf{x}_+ + \mathbf{b} = +1$ and $\mathbf{w}^T\mathbf{x}_- + \mathbf{b} = -1$ for the positive and negative support vectors respectively
- Hence, margin now is: $2 * \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$



Maximizing the Margin



- Then we can formulate the *quadratic optimization problem:*

Find **w** and *b* such that

$$\rho = \frac{2}{\|\mathbf{w}\|}$$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

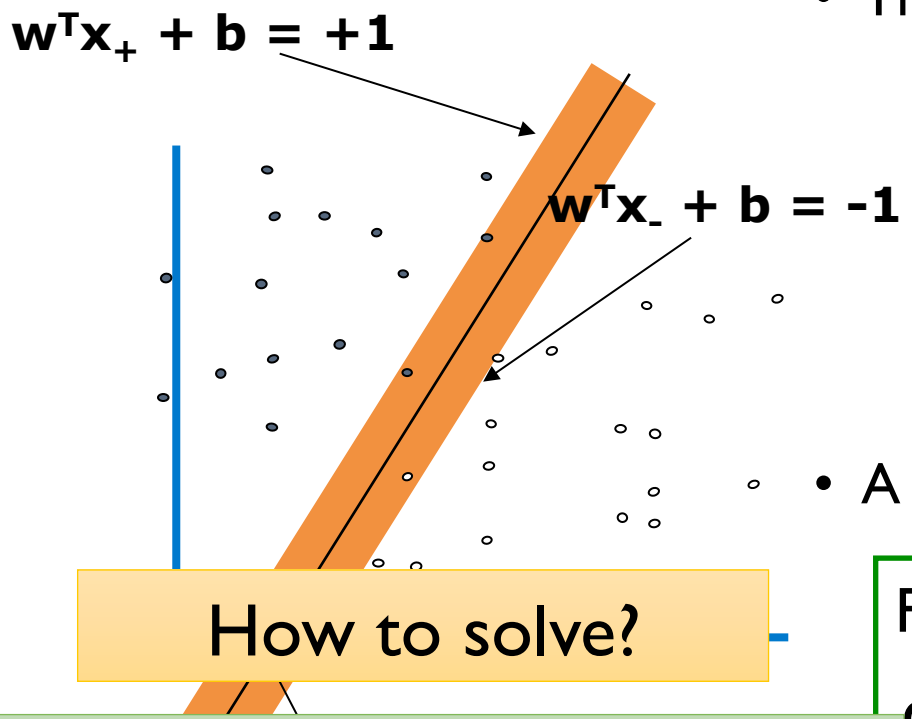
- A better formulation (min $||\mathbf{w}||$ = max $1/||\mathbf{w}||$):

Find **w** and *b* such that

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

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

Maximizing the Margin



- Then we can formulate the *quadratic optimization problem:*

Find **w** and *b* such that

$\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized; and for all $\{(\mathbf{x_i}, y_i)\}$

$\mathbf{w}^T\mathbf{x_i} + b \geq 1$ if $y_i = 1$; $\mathbf{w}^T\mathbf{x_i} + b \leq -1$ if $y_i = -1$

- A better formulation (min ||**w**|| = max 1/ ||**w**||):

Find **w** and *b* such that

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

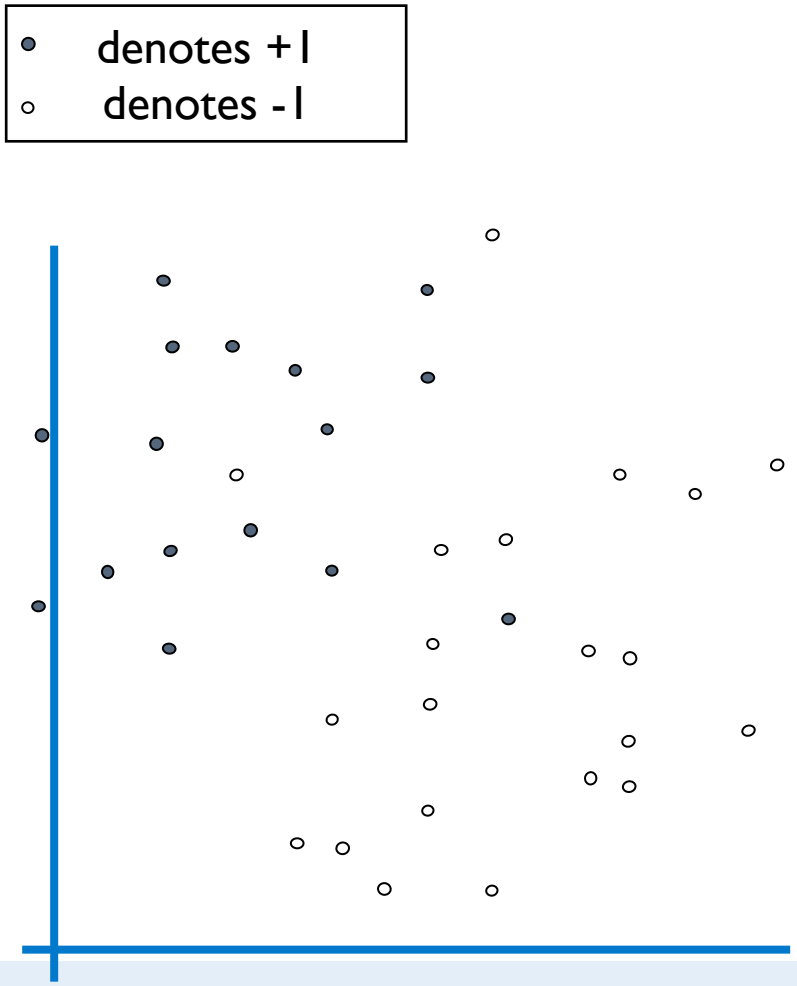
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

How to solve?

Quadratic Programming

67

Non-separable Data



This is going to be a problem!
What should we do?

SVM for Noisy Data

$$\{\vec{w}^*, b^*\} = \min_{w,b} \sum_{i=1}^{d} w_i^2 + c\sum_{j=1}^{N} \varepsilon_j$$

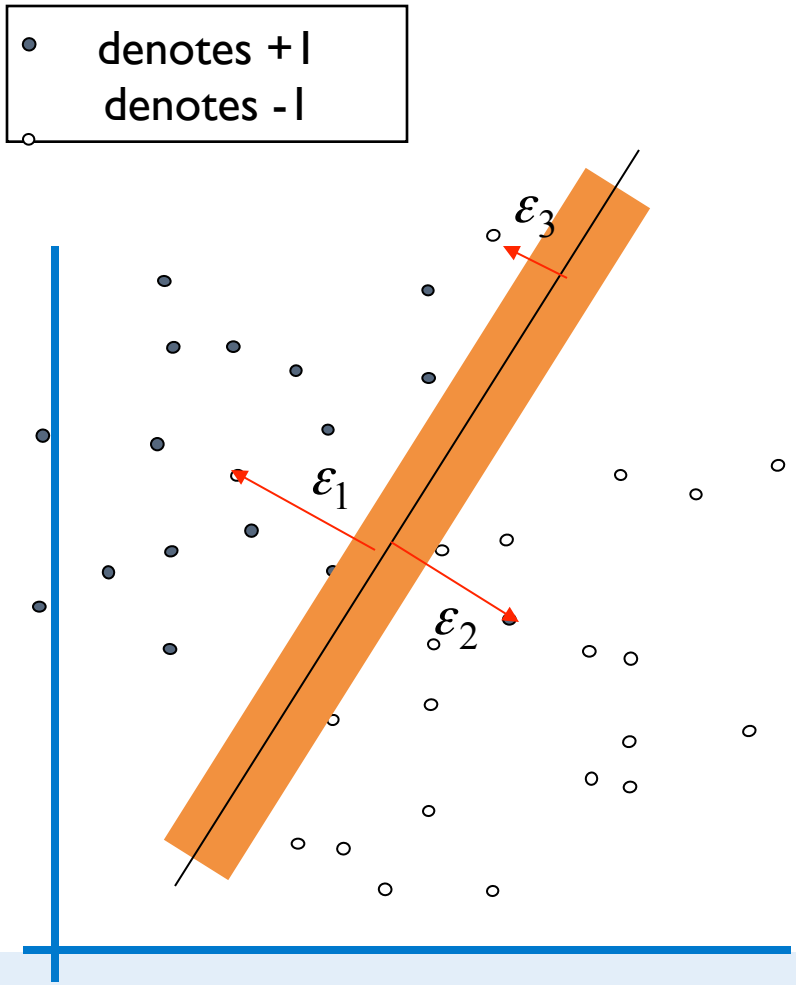
$$y_1\left(\vec{w}\cdot\vec{x}_1 + b\right) \geq 1 - \varepsilon_1, \varepsilon_1 \geq 0$$

$$y_2\left(\vec{w}\cdot\vec{x}_2 + b\right) \geq 1 - \varepsilon_2, \varepsilon_2 \geq 0$$

...

$$y_N\left(\vec{w}\cdot\vec{x}_N + b\right) \geq 1 - \varepsilon_N, \varepsilon_N \geq 0$$

- Balance the trade off between margin and classification errors

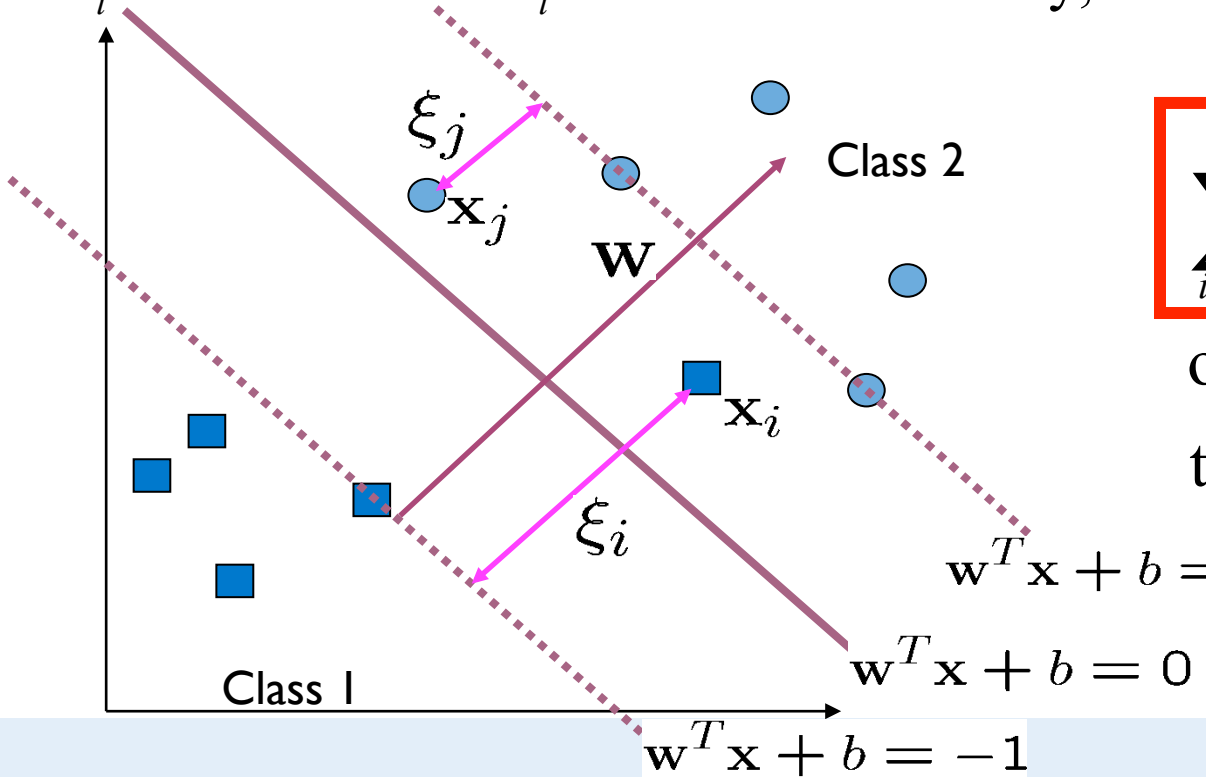


SVM for Noisy Data

$\varepsilon_i \geq 1 \quad \Leftrightarrow \quad y_i(wx_i + b) < 0,$ i.e., misclassification

$0 \prec \varepsilon_i \prec 1 \quad \Leftrightarrow \quad x_i$ is correctly classified, but lies inside the margin

$\varepsilon_i = 0 \quad \Leftrightarrow \quad x_i$ is classified correctly, and lies outside the margin



$\sum_{i=1}^{k} \varepsilon_i$ is an upper bound on the number of training errors.

SVM for Noisy Data

Use the Lagrangian formulation for the optimization problem.

Introduce a positive Lagrangian multiplier for each inequality constraint.

$$y_i(x_i \bullet w + b) - 1 + \varepsilon_i \geq 0, \text{ for all i.}$$

$$\varepsilon_i \geq 0, \text{ for all i.}$$

α_i Lagrangian multipliers

β_i

Get the following Lagrangian: $L_p = \|w\|^2 + c\sum_i \varepsilon_i - \sum_i \alpha_i \{y_i(x_i \bullet w + b) - 1 + \varepsilon_i\} - \sum_i \beta_i \varepsilon_i$

SVM for Noisy Data

$$L_p = \|w\|^2 + c\sum \varepsilon_i - \sum \alpha_i \{y_i (x_i \bullet w + b) - 1 + \varepsilon_i\} - \sum \beta_i \varepsilon_i$$

Take the derivatives of L_p with respect to w, b, and ε_i.

$$\frac{\partial L_p}{\partial w} = 2w - \sum_i \alpha_i y_i x_i = 0 \quad \Rightarrow \quad w = \frac{1}{2}\sum_i \alpha_i y_i x_i$$

$$\frac{\partial L_p}{\partial b} = -\frac{1}{2}\sum_i \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_i \alpha_i y_i = 0$$

$$\frac{\partial L_p}{\partial \varepsilon_i} = c - \beta_i - \alpha_i = 0 \quad \Rightarrow \quad c = \beta_i + \alpha_i$$

Karush-Kuhn-Tucker Conditions

$$0 \le \alpha_i \le c \quad \forall i$$

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i \bullet x_j)$$

Both ε_i and its multiplier β_i are not involved in the function.

Dual Form

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints: $0 \le \alpha_k \le c \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

$$y_i (x_i \bullet w + b) - 1 = 0$$

$$b = -y_i (y_i (x_i \bullet w) - 1)$$

Then classify with:

***f*(*x*,*w*,*b*) = *sign*(*w*. *x* + *b*)**

Dual Form

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints: $0 \le \alpha_k \le c \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

$$y_i (x_i \bullet w + b) - 1 = 0$$

$$b = -y_i (y_i (x_i \bullet w) - 1)$$

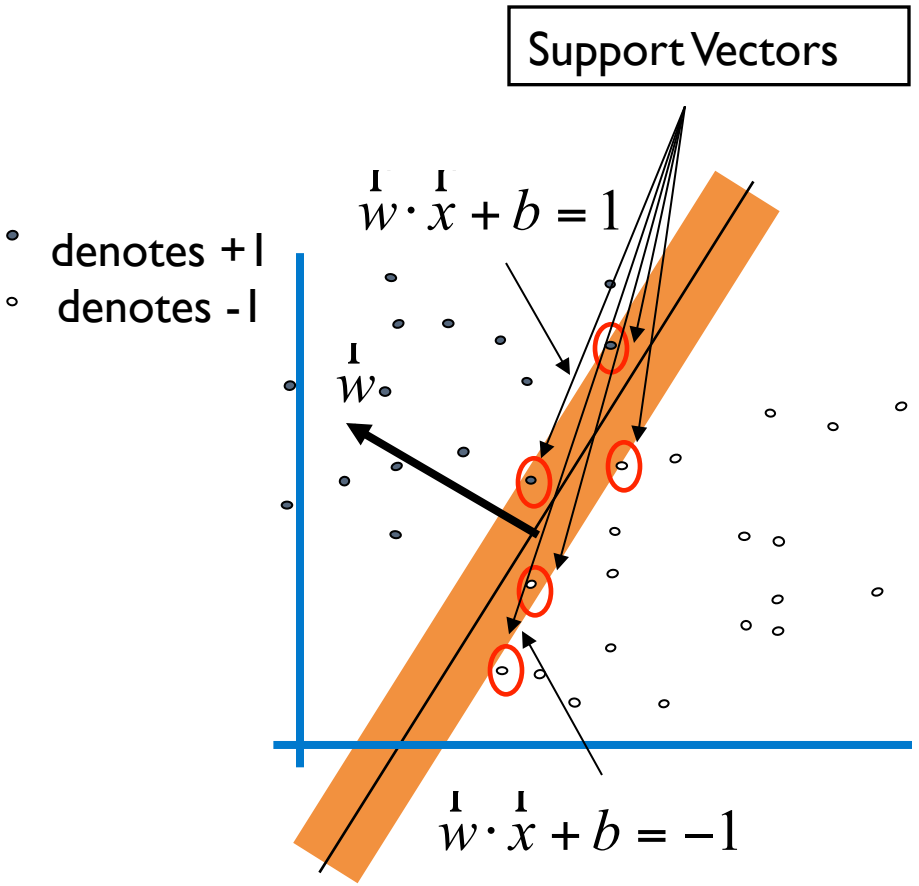
Datapoints with $\alpha_k > 0$ will be the support vectors

..so this sum only needs to be over the support vectors.

Then

f(x,w,b) = sign(w. x + b)

Support Vectors



$$\forall i : \alpha_i \left(y_i \left(\vec{w} \cdot \vec{x}_i + b \right) - \left(1 - \varepsilon_i \right) \right) = 0$$

$\alpha_i = 0$ for non-support vectors

$\alpha_i \neq 0$ for support vectors

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

Decision boundary is determined only by those support vectors !

CS6510
Applied Machine Learning

Classifiers

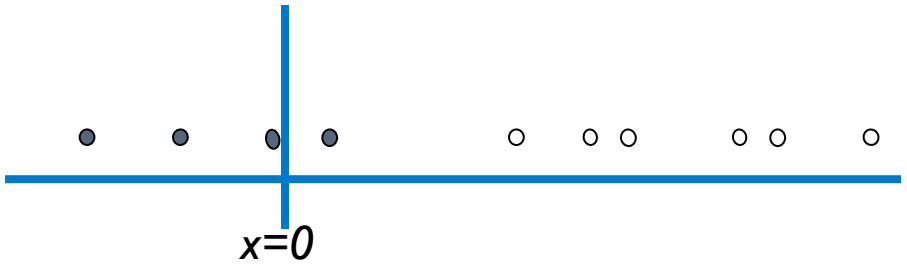
3 Sep 2016

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

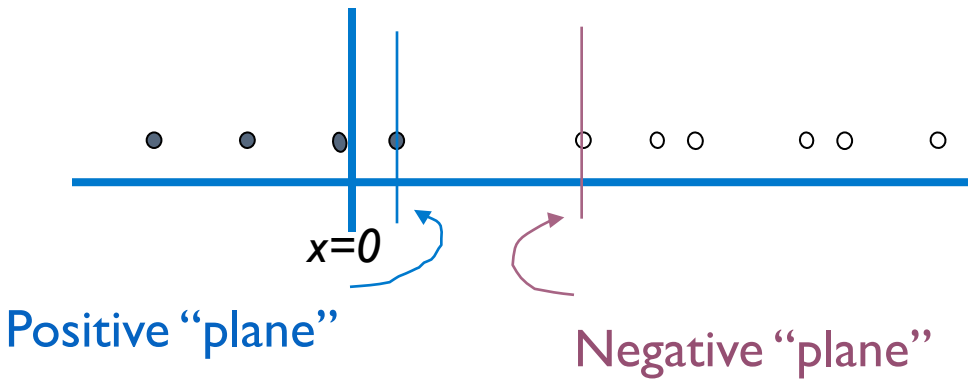
Assume we are in 1-dimension

What would SVMs do with this data?



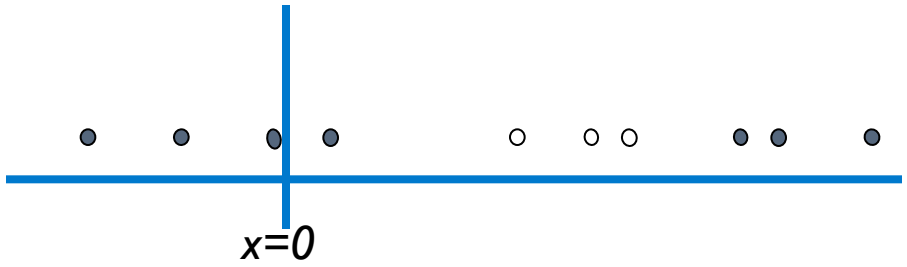
Assume we are in 1-dimension

Not a big surprise

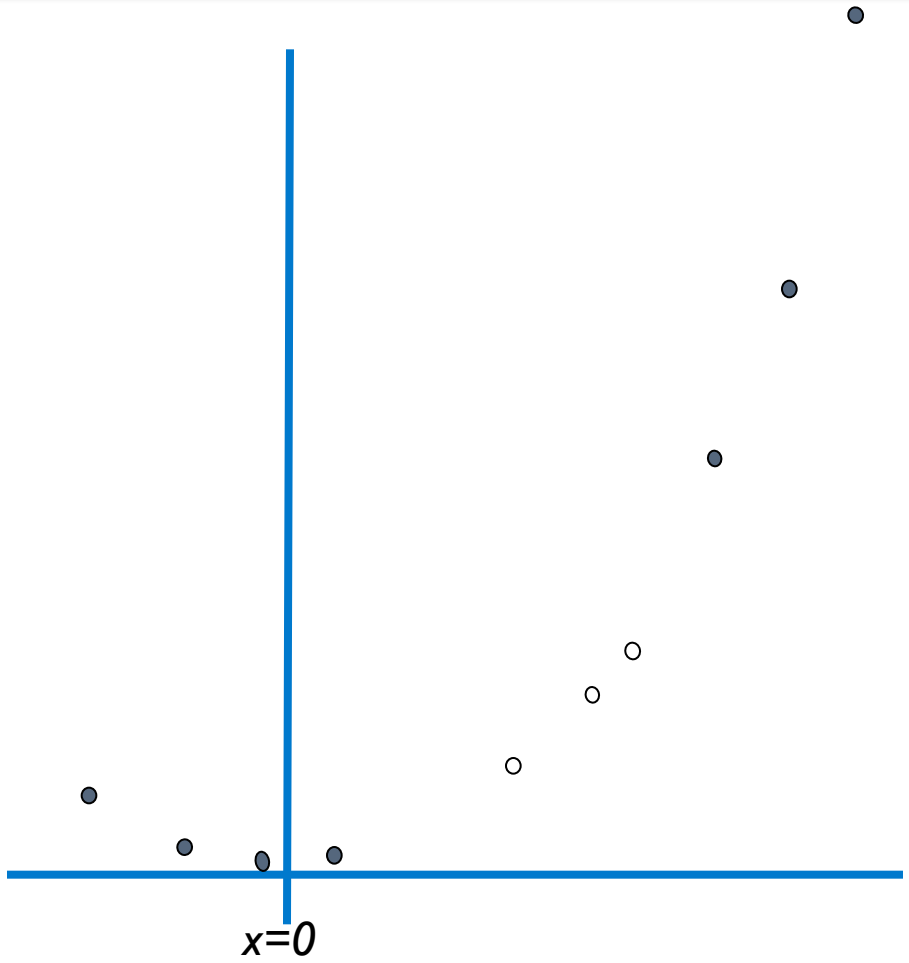


Harder 1-dimensional Dataset

What can be done about this?



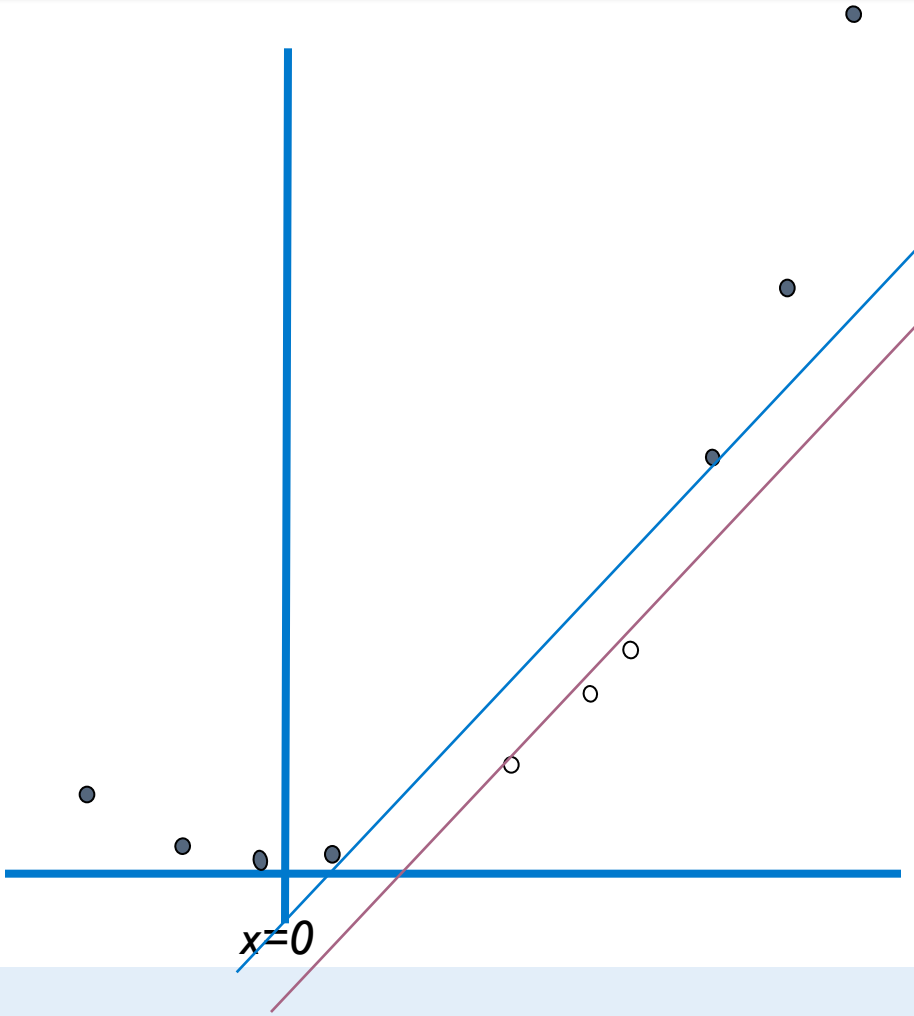
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

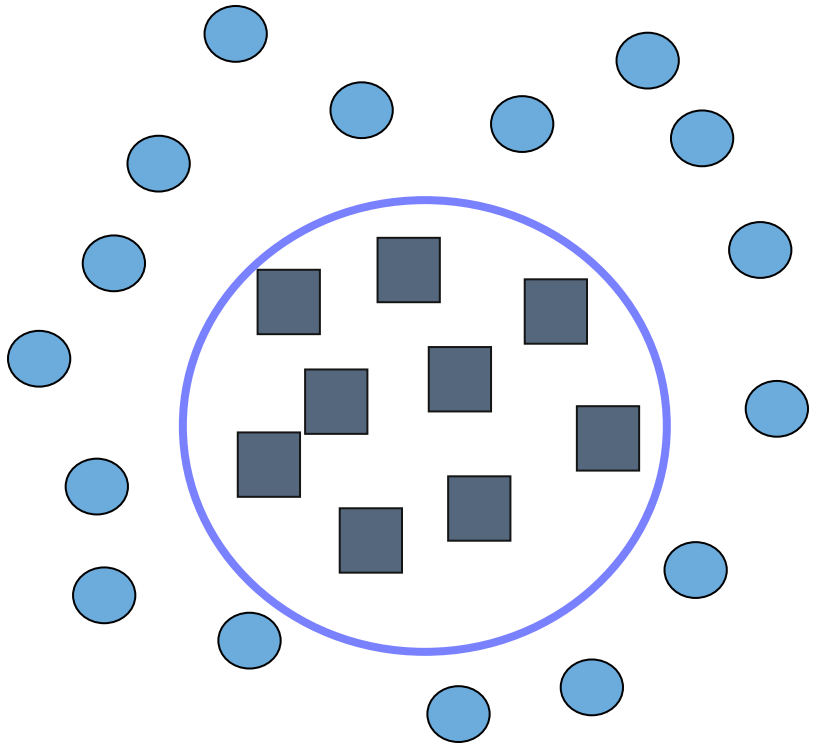
Harder 1-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, x_k^2)$$

Harder 2-dimensional Dataset



Apply the following map

$$\mathbf{z}_k = (x_k, y_k, x_k^2, y_k^2, x_k y_k)$$

Common SVM Basis Functions

$\mathbf{z}_k$ = (polynomial terms of $\mathbf{x}_k$ of degree 1 to q)

$\mathbf{z}_k$ = (radial basis functions of $\mathbf{x}_k$)

$$\mathbf{z}_k[j] = \varphi_j(\mathbf{x}_k) = \exp\left(-\frac{|\mathbf{x}_k - \mathbf{c}_j|^2}{\sigma^2}\right)$$

$\mathbf{z}_k$ = (sigmoid functions of $\mathbf{x}_k$)

Recall: Dual Form

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{x}_k \cdot \mathbf{x}_l)$

Subject to these constraints: $0 \le \alpha_k \le c \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \frac{1}{2} \sum_{k=1}^{R} \alpha_k y_k \mathbf{x}_k$$

$$y_i (x_i \bullet w + b) - 1 = 0$$

$$b = -y_i (y_i (x_i \bullet w) - 1)$$

Then classify with:

***f*(*x*,*w*,*b*) = *sign*(*w*. *x* + *b*)**

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l))$

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k = 0$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Then classify with:

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

Most important change:

$\mathbf{x} \rightarrow \mathbf{\Phi}(\mathbf{x})$

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k).\Phi(\mathbf{x}_l))$

Subject to these constraints: $0 \le \alpha_k \le C$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

We must do $R^2/2$ dot products to get this matrix ready

Assuming a quadratic polynomial kernel, each dot product requires $m^2/2$ additions and multiplications

The whole thing costs $R^2 m^2 /4$.

$x \rightarrow \Phi(x)$

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \bullet \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix} = 1 + \sum_{i=1}^{m} 2a_ib_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2a_ia_jb_ib_j$$

Quadratic Dot Products

Quadratic Dot Products

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = 1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

Quadratic Dot Products

They're the same!
And this is only O(m) to compute!

$$\Phi(\mathbf{a}) \bullet \Phi(\mathbf{b}) = 1 + 2\sum_{i=1}^{m} a_i b_i + \sum_{i=1}^{m} a_i^2 b_i^2 + \sum_{i=1}^{m} \sum_{j=i+1}^{m} 2 a_i a_j b_i b_j$$

Just out of interest, let's look at another function of ***a*** and ***b***:

$$(\mathbf{a}.\mathbf{b} + 1)^2$$

$$= (\mathbf{a}.\mathbf{b})^2 + 2\mathbf{a}.\mathbf{b} + 1$$

$$= \left(\sum_{i=1}^{m} a_i b_i \right)^2 + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

$$= \sum_{i=1}^{m} (a_i b_i)^2 + 2\sum_{i=1}^{m} \sum_{j=i+1}^{m} a_i b_i a_j b_j + 2\sum_{i=1}^{m} a_i b_i + 1$$

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l (\Phi(\mathbf{x}_k).\Phi(\mathbf{x}_l))$

Subject to these constraints: $0 \le \alpha_k \le C$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \Phi(\mathbf{x}_k)$$

We must do $R^2/2$ dot products to get this matrix ready

Now, each dot product now only requires m additions and multiplications

$f(\mathbf{x}, \mathbf{w}, b) = \text{sign}(\mathbf{w}.\Phi(\mathbf{x}) - b)$

Most important change:

$\mathbf{x} \rightarrow \Phi(\mathbf{x})$

Higher-Order Polynomials

Poly-nomial	f(x)	Cost to build Q_{kl} matrix traditionally	Cost if 100 dimensions	f(a).f(b)	Cost to build Q_{kl} matrix sneakily	Cost if 100 dimensions
Quadratic	All $m^2/2$ terms up to degree 2	$m^2 R^2 /4$	2,500 R^2	$(a.b+1)^2$	$m R^2 / 2$	50 R^2
Cubic	All $m^3/6$ terms up to degree 3	$m^3 R^2 /12$	83,000 R^2	$(a.b+1)^3$	$m R^2 / 2$	50 R^2
Quartic	All $m^4/24$ terms up to degree 4	$m^4 R^2 /48$	1,960,000 R^2	$(a.b+1)^4$	$m R^2 / 2$	50 R^2

SVM QP with Basis Functions

Maximize $\sum_{k=1}^{R} \alpha_k - \frac{1}{2} \sum_{k=1}^{R} \sum_{l=1}^{R} \alpha_k \alpha_l Q_{kl}$ where $Q_{kl} = y_k y_l K(\mathbf{x}_k, \mathbf{x}_l)$

Kernel gram matrix

Subject to these constraints: $0 \le \alpha_k \le C \quad \forall k$ $\sum_{k=1}^{R} \alpha_k y_k$

Then define:

$$\mathbf{w} = \sum_{k \text{ s.t. } \alpha_k > 0} \alpha_k y_k \mathbf{\Phi}(\mathbf{x}_k)$$

Then classify with:

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

Most important change:

$$\mathbf{\Phi}(\mathbf{x}_k).\mathbf{\Phi}(\mathbf{x}_l) \rightarrow K(\mathbf{x}_k, \mathbf{x}_l)$$

SVM Kernel Functions

- $K(\boldsymbol{a},\boldsymbol{b})=(\boldsymbol{a} \cdot \boldsymbol{b} + 1)^d$ is an example of an SVM Kernel Function
- Beyond polynomials there are other very high dimensional basis functions that can be made practical by finding the right Kernel Function
  - Radial-Basis-style Kernel Function:

$$K(\mathbf{a},\mathbf{b}) = \exp\left(-\frac{(\mathbf{a}-\mathbf{b})^2}{2\sigma^2}\right)$$

  - Sigmoidal function

Kernel Tricks

- Replacing dot product with a kernel function
- Not all functions are kernel functions
  - Need to be decomposable
    - $K(a,b) = \phi(a) \cdot \phi(b)$

- **Mercer's condition**

To expand Kernel function K(x,y) into a dot product, i.e. $K(x,y)=\Phi(x)\cdot\Phi(y)$, K(x, y) has to be positive semi-definite function, i.e., for any function f(x) whose $\int f^2(x)dx$ is finite, the following inequality holds

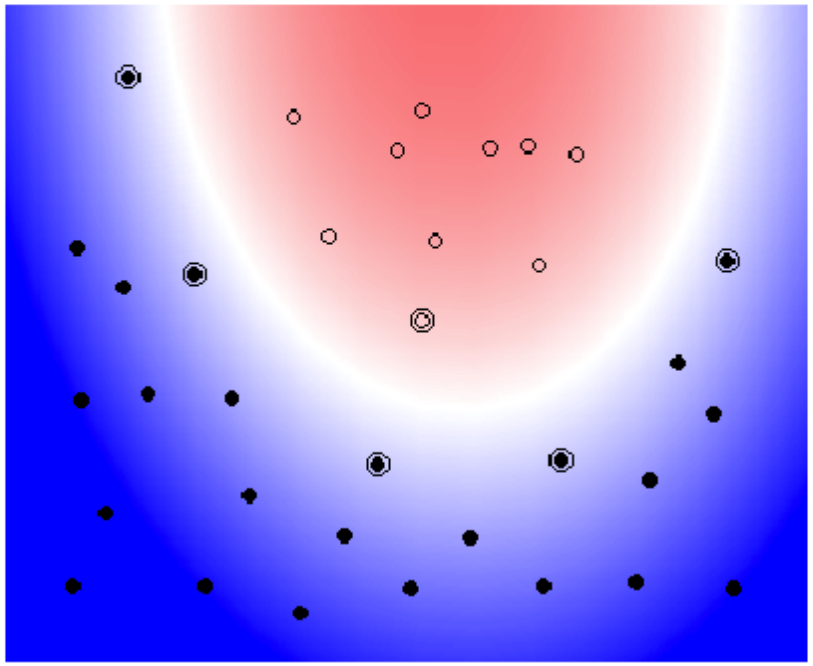
$$\int dxdyf(x)K(x,y)f(y) \geq 0$$

Non-Linear Kernel: Visualization

Example: SVM with Polynomial of Degree 2

Kernel: $K(\vec{x}_i, \vec{x}_j) = [\vec{x}_i \cdot \vec{x}_j + 1]^2$

plot by Bell SVM applet



How to choose a kernel function?

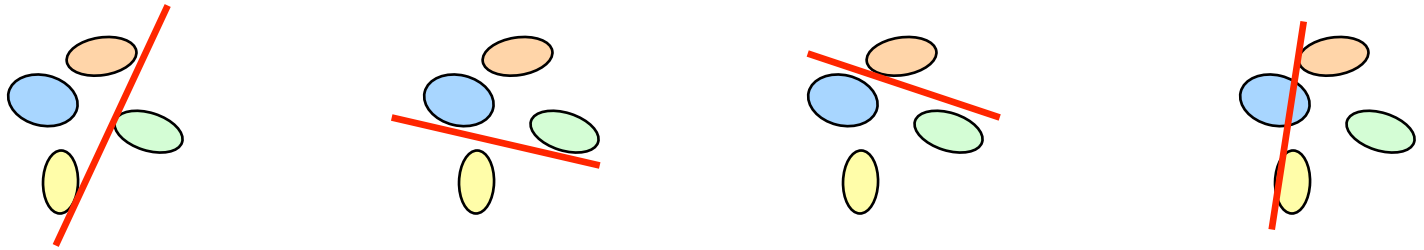
- Not easy! Remember – this depends on your data geometry
- If linear works, go with it
- RBF kernels are considered good in general, especially for images (and other smooth functions/data)
- For discrete data, chi-square kernel preferred of late (especially for histogram data)
- You can also do Multiple Kernel Learning
  - Beyond scope of current course – you can read more at: https://en.wikipedia.org/wiki/Multiple_kernel_learning
- Still not sure? Use cross-validation to select a kernel function from some basic options

From Binary Classification to Multi-class Classification

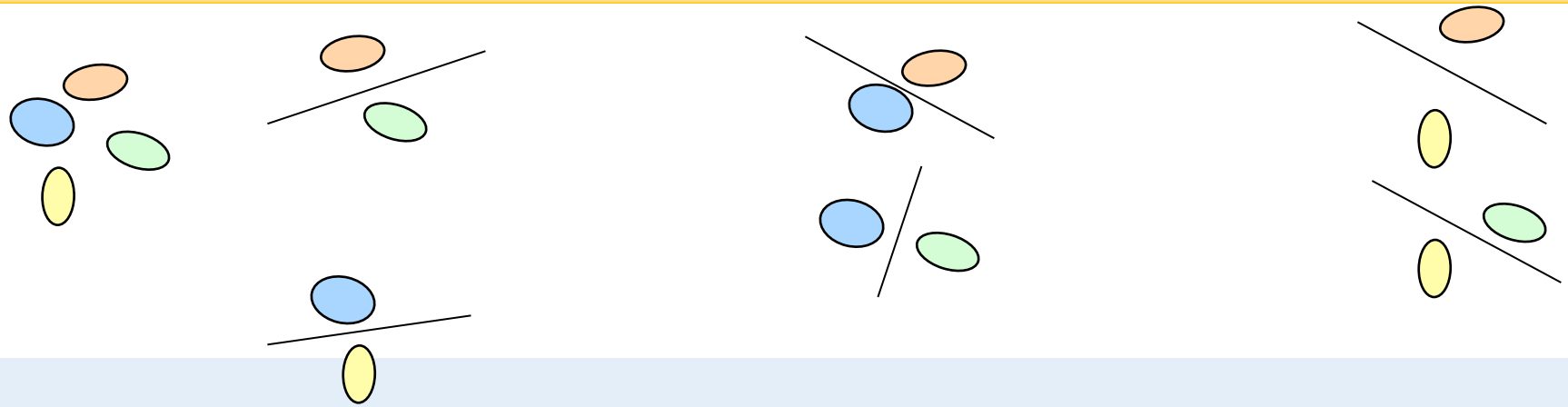
- SVMs can only handle two-class outputs.
- What can be done?
- Answer: with output arity N, learn N SVM's
  - SVM 1 learns "Output==1" vs "Output != 1"
  - SVM 2 learns "Output==2" vs "Output != 2"
  - :
  - SVM N learns "Output==N" vs "Output != N"
- Then to predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.
- Other approaches
  - Pair-wise SVM, Tree-structured SVM

Multi-class Classification using SVM

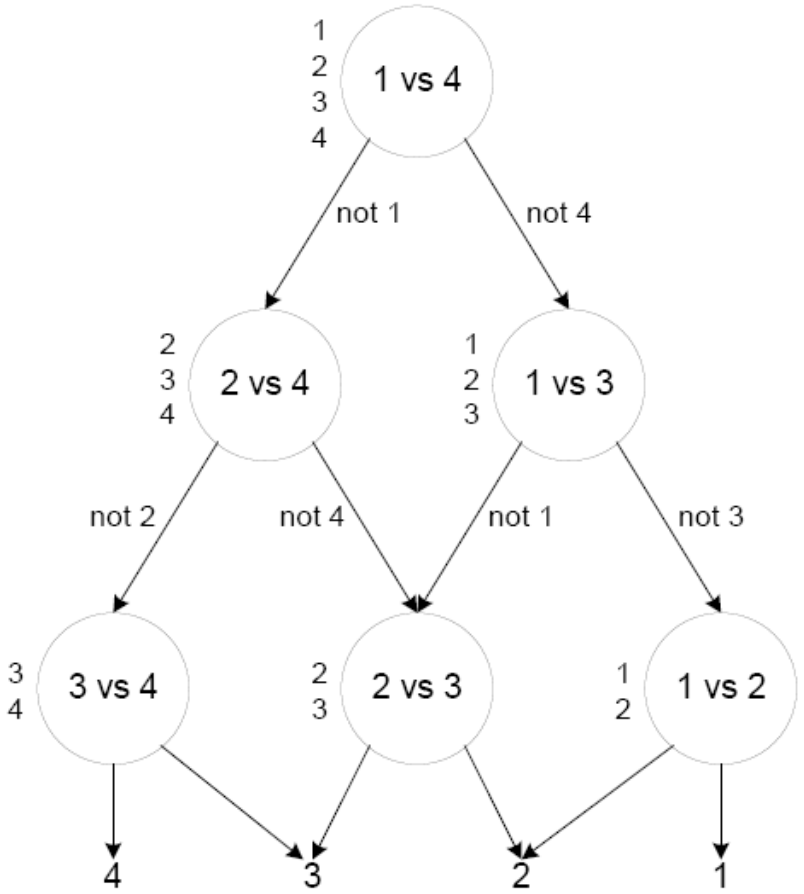
One- versus-all



One- versus-one



Tree-Structured SVM



Readings

- "Introduction to Machine Learning" by Ethem Alpaydin, Chapters 8, 9, 13, 16.1-16.3