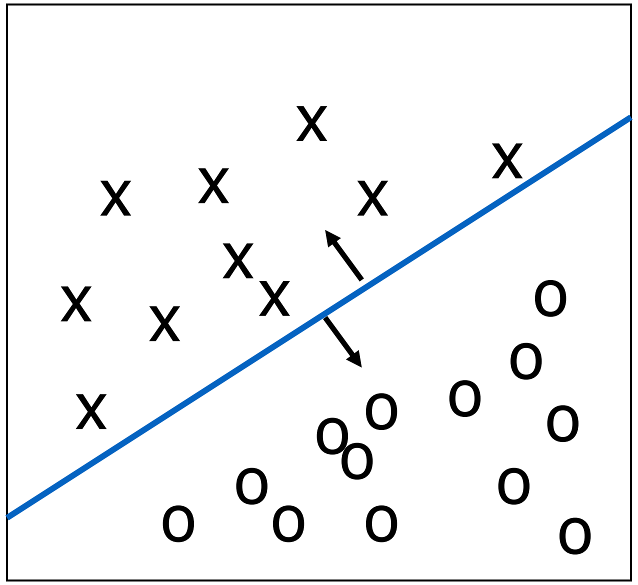
SVM, Bayesian Networks, and Ensemble Methods

COSC 757 Spring 2016

Classification: A Mathematical Mapping

- Classification: predicts categorical class labels
  - E.g., Personal homepage classification
    - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
    - x_1 : # of word "homepage"
    - x_2 : # of word "welcome"
- Mathematically, $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$,
  - We want to derive a function $f: X \rightarrow Y$
- Linear Classification
  - Binary Classification problem
  - Data above the blue line belongs to class 'x'
  - Data below blue line belongs to class 'o'
  - Examples: SVM, Perceptron, Probabilistic Classifiers



SVM: Support Vector Machines

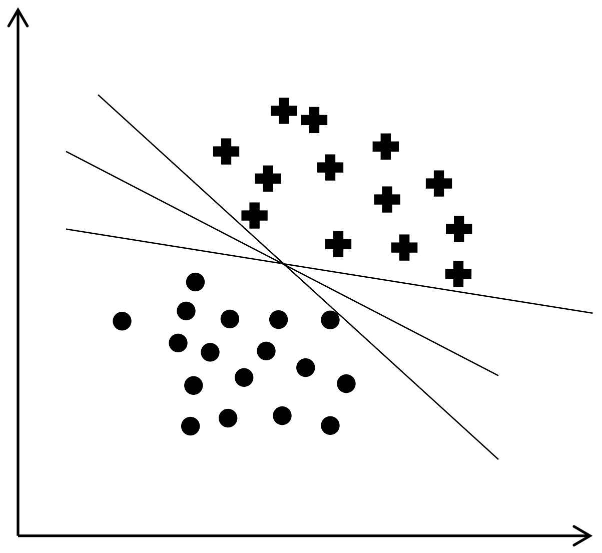
- Uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., “decision boundary”)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors (“essential” training tuples) and margins (defined by the support vectors)

SVM History and Applications

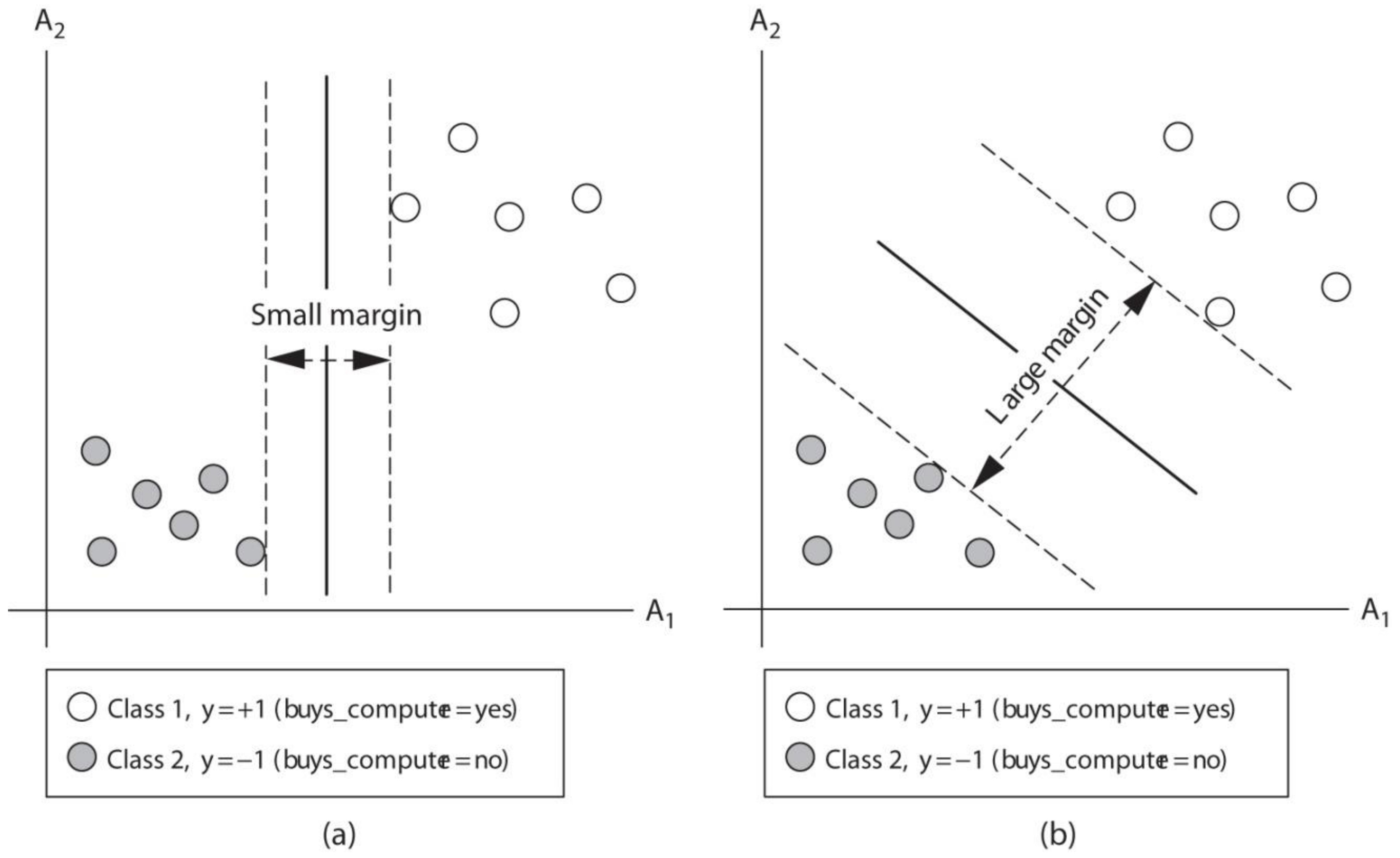
- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis’ statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used for: classification and numeric prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

Which Hyperplane?

- SVMs maximize the margin around the separating hyperplane
- The decision function is fully specified by a subset of training samples (support vectors)
- Quadratic programming problem



SVM: General Philosophy


A₂
A₁
Small margin
Large margin
○ Class 1, y=+1 (buys_computer=yes)
● Class 2, y=−1 (buys_computer=no)
(a)
(b)

Maximum Marginal Hyperplane

SVM: When Data is Linearly Separable

- Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i
- There are infinite lines (<u>hyperplanes</u>) separating the two classes but we want to <u>find the best one</u> (the one that minimizes classification error on unseen data)
- *SVM searches for the hyperplane with the largest margin*, i.e., **maximum marginal hyperplane** (MMH)

SVM: Linearly Separable

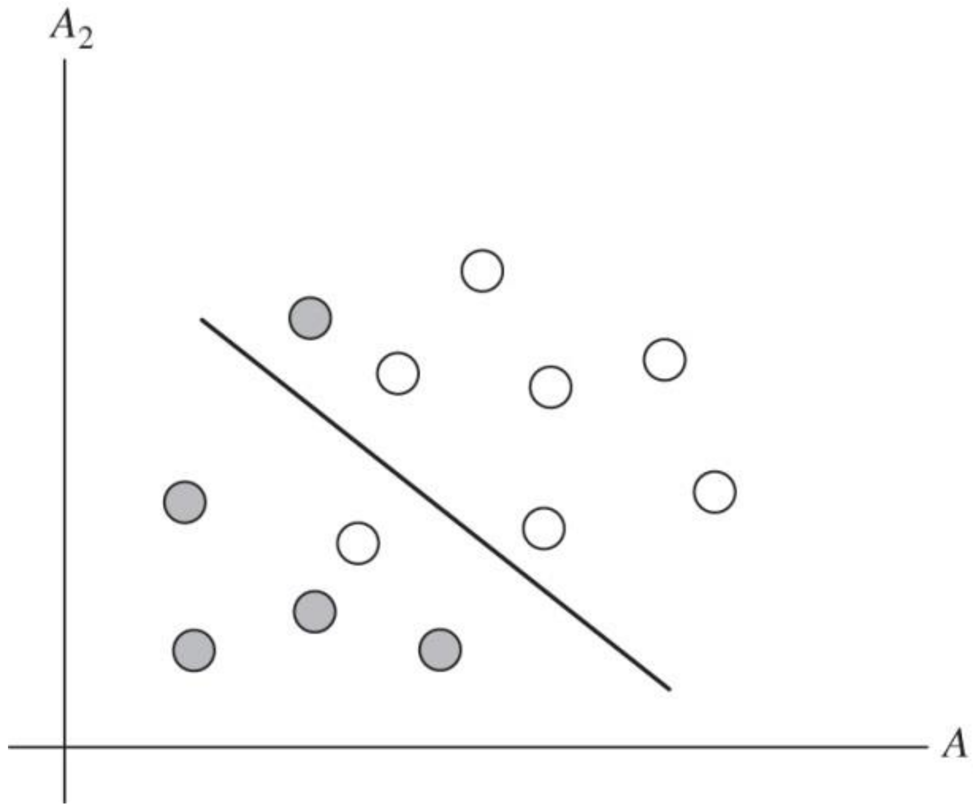
- A separating hyperplane can be written as
  - W• X + b = 0
- where W={w1, w2, …, wn} is a weight vector and b a scalar (bias)
- For 2-D it can be written as
  - w0 + w1 x1 + w2 x2 = 0
- where w0 is the bias
- Thus any point lying above the separating hyperplane satisfies:
  - w0 + w1 x1 + w2 x2 > 0
- And any point lying below the separating hyperplane satisfies:
  - w0 + w1 x1 + w2 x2 < 0
- Thus the hyperplane defining the sides of the margin can be written as:
  - H1: w0 + w1 x1 + w2 x2 ≥ 1   for yi = +1, and
  - H2: w0 + w1 x1 + w2 x2 ≤ – 1 for yi = –1
- Any training tuples that fall on hyperplanes H1 or H2 (i.e., the sides defining the margin) are support vectors
- This becomes a **constrained (convex) quadratic optimization problem**: Quadratic objective function and linear constraints; Quadratic Programming (QP); Lagrangian multipliers

Why is SVM Effective on High Dimensional Data?

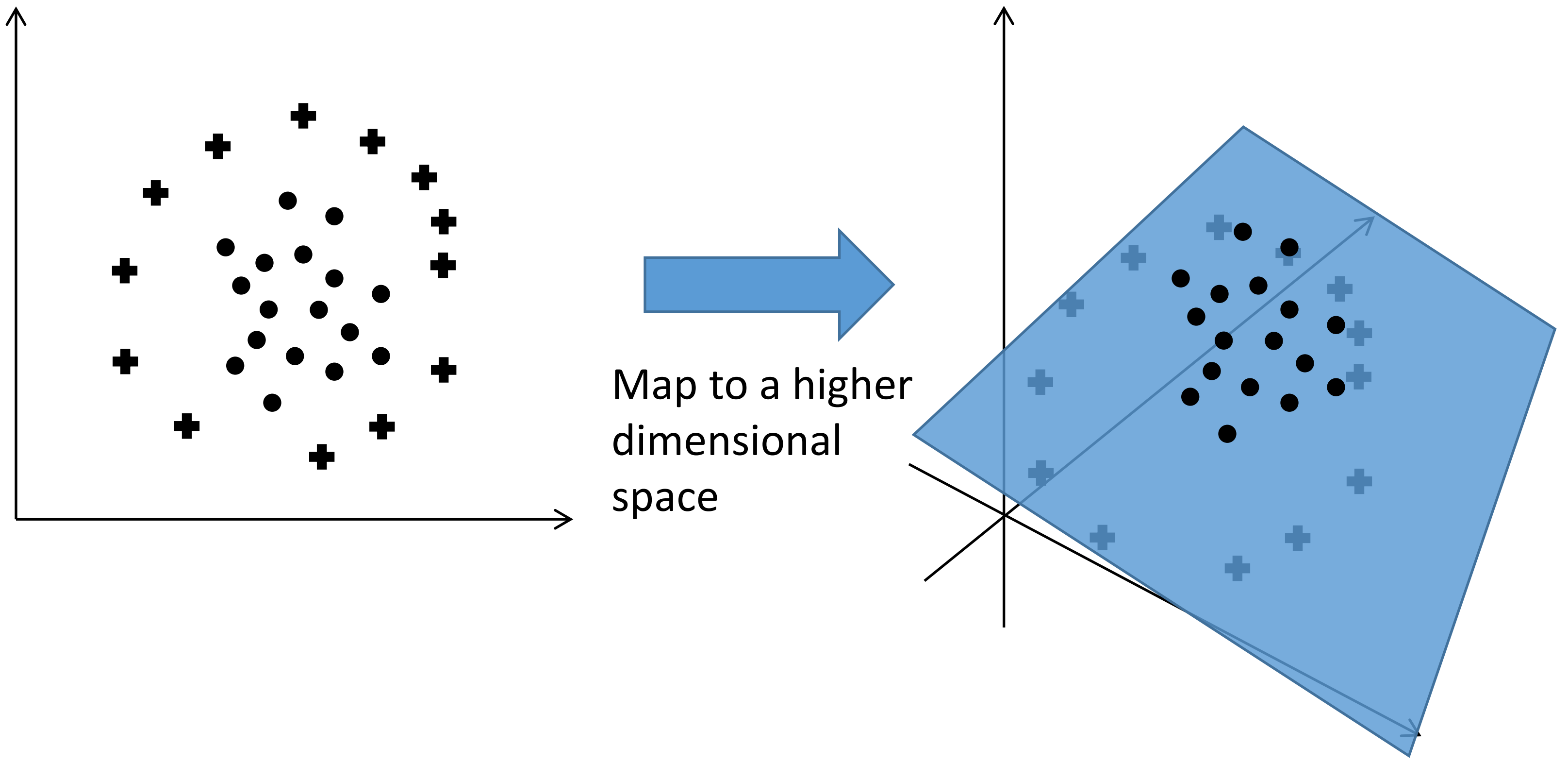
- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

SVM: Linearly Inseparable

- Extend linear SVMs
- Find non-linear decision boundaries (non-linear hypersurfaces).
- Step 1: transform the original input data into high dimensional space
- Step 2: Search for linear separating hyperplane in the new space
- The linear hyperplane in high dimensional space is non-linear in the original space



SVM Linearly Inseparable



SVM: Linearly Inseparable

- Requires a mapping to higher dimensional space
  - e.g. $\phi([x_i,x_j]) = [x_i, x_j, x_i^2+x_j^2]$
  - or $\phi(X_i,X_j) = \phi(X_i)\bullet\phi(X_j)$
- Two problems:
  - How to choose a mapping?
  - Higher number of dimensions increases computational cost
    - Computing the dot product for all of the support vectors
      - One multiplication and one addition for each dimension
- Math trick:
  - Apply a kernel function to the original input data
  - Wherever a dot product is used, replace it with a kernel function

SVM: Kernel Function

- Equivalent to the dot product
  - $K(X_i, X_j) = \phi(X_i) \bullet \phi(X_j)$
- We can solve $K(X_i, X_j)$ without having to perform any of the mappings $\phi(X_i)$
- Provide a way to manipulate data as though it were projected into a higher dimensional space by operating on it in its original space

SVM: Kernel Function

- Instead of computing the dot product on the transformed data, it is mathematically equivalent to apply a kernel function K(Xi, Xj) to the original data, i.e., K(Xi, Xj) = *Φ*(Xi) *Φ*(Xj)
- Typical Kernel Functions

Polynomial kernel of degree h : $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = (\boldsymbol{X_i} \cdot \boldsymbol{X_j} + 1)^h$

Gaussian radial basis function kernel : $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = e^{-\|\boldsymbol{X_i} - \boldsymbol{X_j}\|^2 / 2\sigma^2}$

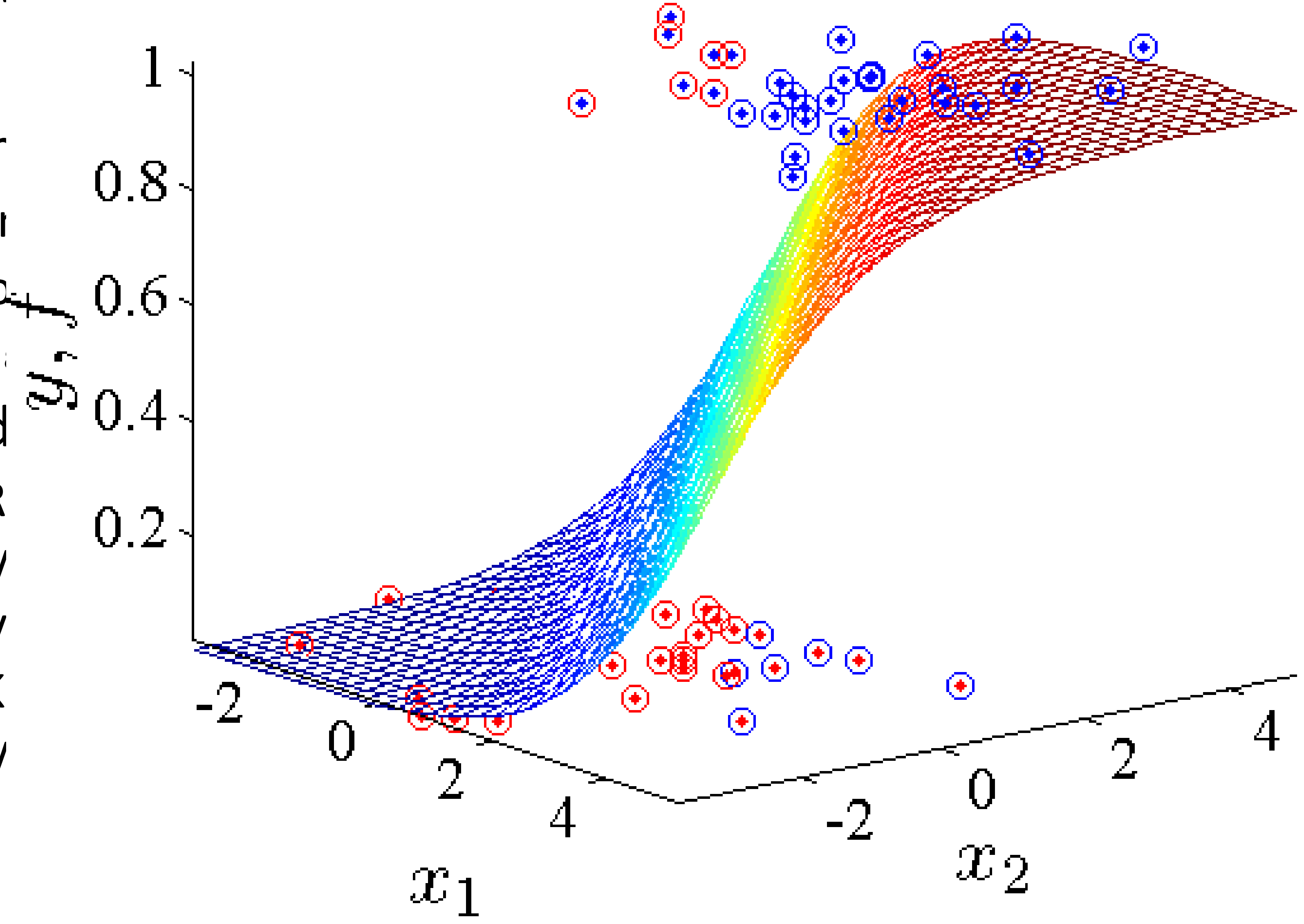
Sigmoid kernel : $K(\boldsymbol{X_i}, \boldsymbol{X_j}) = \tanh(\kappa \boldsymbol{X_i} \cdot \boldsymbol{X_j} - \delta)$

- After applying the kernel function, finding maximal margin hyperplane is similar to linear SVM with an upper bound on the Lagrange multipliers
- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional parameters)

https://www.youtube.com/watch?v=3liCbRZPrZA

SVI

- Esser
  - Bi
  - Bo
    La
- Key d
  - LR
  - SV
    by
  - “K
  - SV

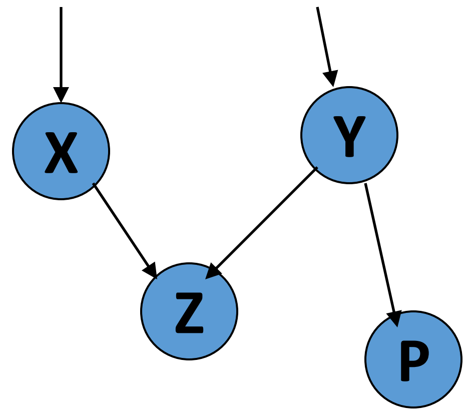


it and

defined

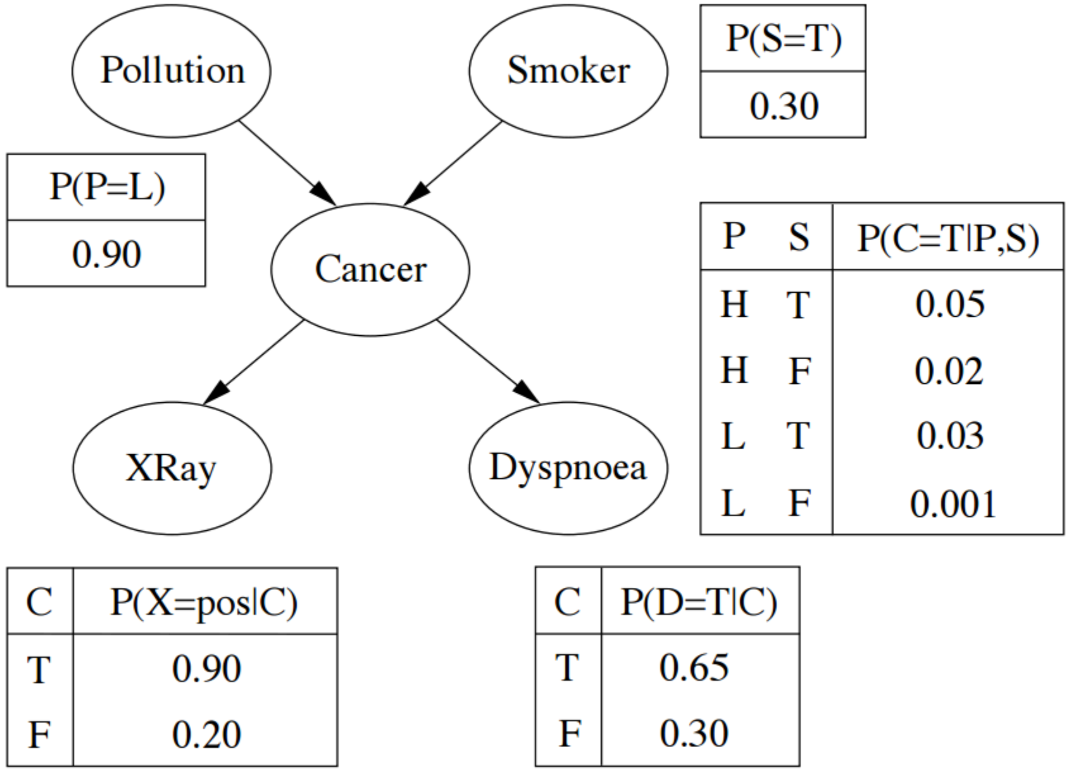
Bayesian Belief Networks

- **Bayesian belief networks** (also known as **Bayesian networks**, **probabilistic networks**): allow *class conditional independencies* between *subsets* of variables
- A (*directed acyclic*) graphical model of causal relationships
  - Represents <u>dependency</u> among the variables
  - Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops/cycles

Bayesian Belief Network: An Example



P(S=T)
0.30

P(P=L)
0.90

P	S	P(C=T|P,S)
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

C	P(X=pos|C)
T	0.90
F	0.20

C	P(D=T|C)
T	0.65
F	0.30

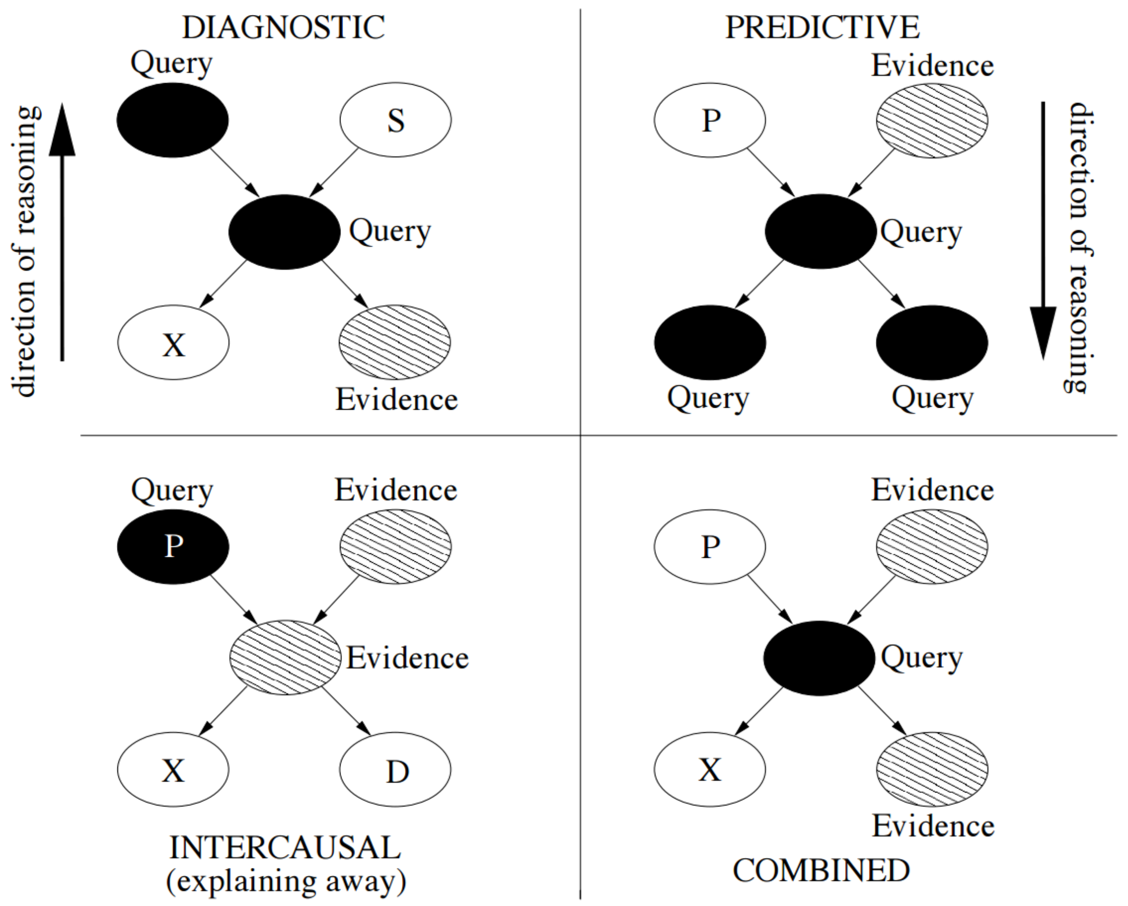
Derivation of the conditional probability of a particular combination of values of **X**, from CPT:

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i | Parents(X_i))$$

$$\begin{aligned} & P(X = pos \wedge D = T \wedge C = T \wedge P = low \wedge S = F) \\ = \; & P(X = pos | D = T, C = T, P = low, S = F) \\ & \times P(D = T | C = T, P = low, S = F) \\ & \times P(C = T | P = low, S = F) P(P = low | S = F) P(S = F) \\ = \; & P(X = pos | C = T) P(D = T | C = T) P(C = T | P = low, S = F) \\ & \times P(P = low) P(S = F) \end{aligned}$$

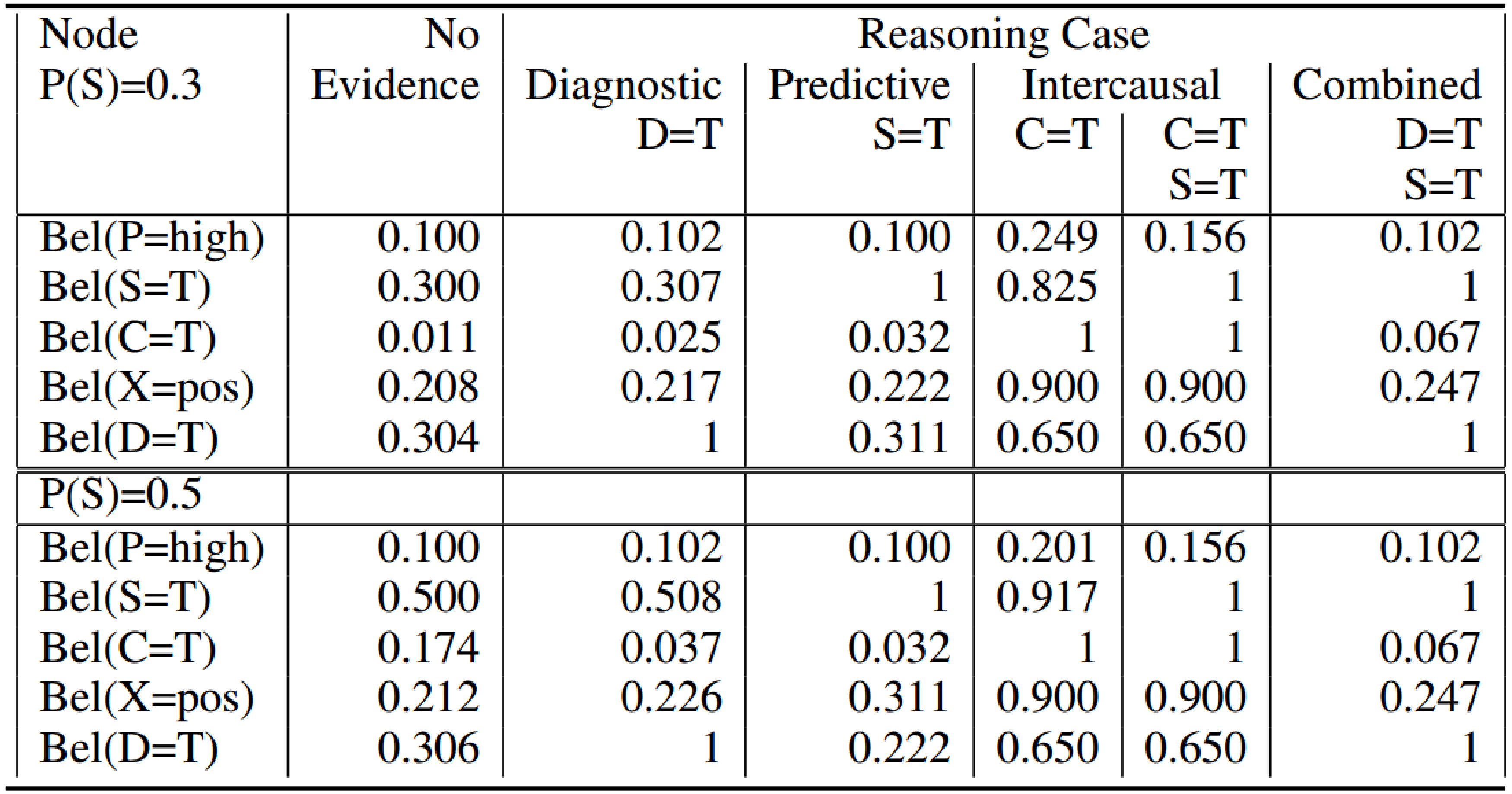
Bayesian Belief Network: Types of Reasoning

- Bayesian Networks can support any direction of reasoning
- Diagnostic reasoning
  - Reasoning from symptom to cause
- Predictive reasoning
  - Reasoning from new information about causes to new beliefs about effects
- Intercausal reasoning
  - Reasoning about the mutual causes of a common effect
- Combined reasoning
  - Combining above types of reasoning in any way



Updating Beliefs Given New Information

P=Pollution
S=Smoker
C=Cancer
X=Xray
D=Dyspnoea



Node	No	Reasoning Case				
P(S)=0.3	Evidence	Diagnostic	Predictive	Intercausal		Combined
		D=T	S=T	C=T	C=T	D=T
					S=T	S=T
Bel(P=high)	0.100	0.102	0.100	0.249	0.156	0.102
Bel(S=T)	0.300	0.307	1	0.825	1	1
Bel(C=T)	0.011	0.025	0.032	1	1	0.067
Bel(X=pos)	0.208	0.217	0.222	0.900	0.900	0.247
Bel(D=T)	0.304	1	0.311	0.650	0.650	1
P(S)=0.5						
Bel(P=high)	0.100	0.102	0.100	0.201	0.156	0.102
Bel(S=T)	0.500	0.508	1	0.917	1	1
Bel(C=T)	0.174	0.037	0.032	1	1	0.067
Bel(X=pos)	0.212	0.226	0.311	0.900	0.900	0.247
Bel(D=T)	0.306	1	0.222	0.650	0.650	1

How are Bayesian Networks Constructed?

- **Subjective construction**: Identification of (direct) causal structure
  - People are quite good at identifying direct causes from a given set of variables & whether the set contains all relevant direct causes
  - Markovian assumption: Each variable becomes independent of its non-effects once its direct causes are known
  - E.g., S ‹— F —› A ‹— T, path S—›A is blocked once we know F—›A
  - HMM (Hidden Markov Model): often used to model dynamic systems whose states are not observable, yet their outputs are
- **Synthesis from other specifications**
  - E.g., from a formal system design: block diagrams & info flow
- **Learning from data**
  - Learn parameters give its structure or learn both structure and parms
  - Maximum likelihood principle: favors Bayesian networks that maximize the probability of observing the given data set

Training Bayesian Networks

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden (missing values): ***gradient descent*** method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, without backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- More:
  - http://www.csse.monash.edu.au/bai/book/BAI_Chapter2.pdf
  - D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models,* M. Jordan, ed.. MIT Press, 1999.

Ensemble Classifiers

- Combine the results from a set of classification models in order to increase accuracy and reduce variability of the classification
- Benefits:
  - The ensemble classifier is likely to have a lower error rate (boosting)
  - The variance of the ensemble classifier will be lower than had we used certain unstable classification models (such as decision trees and neural nets) that have high variability (bagging and boosting)
- Ensemble methods consider the prediction for each classifier and the one with the most votes will be chosen as the output class
- If the individual classifiers classify the cases similarly, then the ensemble classifier will follow suit.
- If the individual classifiers are independent, the voting mechanism ensures that the classifier will only make an error when the majority of classifiers make an error

Ensemble Classifiers

Let ε represent the individual classifier error rate. The probability that *k* of the 5 individual classifiers will make the same wrong prediction if ε =0.2 is:

$$\binom{5}{k}\epsilon^k(1-\epsilon)^{5-k} = \binom{5}{k}0.2^k(1-0.2)^{5-k}$$

And the probability that all 5 of the classifiers will make an error is:

$$\binom{5}{5}0.2^5(0.8)^0 = 0.00032$$

And the error rate for the ensemble is:

$$\begin{aligned} \mathit{Error\ rate}_{\mathit{Ensemble\ classifier}} &= \sum_{i=3}^{5}\binom{5}{i}\epsilon^i(1-\epsilon)^{5-i} = \sum_{i=3}^{5}\binom{5}{i}0.2^i(0.8)^{5-i} \\ &= 0.0512 + 0.0064 + 0.0003 = 0.05792 \end{aligned}$$

Bias, Variance, and Noise

- Bias refers to the average distance between predictions
- Variance measures the variability in the predictions themselves
- Noise represents the lower bound on the prediction error that the predictor can possibly achieve
- Prediction error can be reduced by reducing the bias, variance, or noise
- Noise cannot be reduced
- Bagging reduces the variance of the classifier models
- Boosting can reduce both the bias and variance

Bagging (Bootstrap Aggregating Algorithm)

- Samples (with replacement) are repeatedly taken from the training data set, so that each record has an equal probability of being selected, and each sample is the same size as the original training data set. These are the bootstrap samples.
- A classification or estimation model is trained on each bootstrap sample drawn in Step 1, and a prediction is recorded for each sample.
- The bagging ensemble prediction is then defined to be the class with the most votes in Step 2 (for classification models) or the average of the predictions made in Step 2 (for estimation models).

Bagging Example

- Consider a small data set in which x is the variable value and y is the classification.

x	**0.2**	**0.4**	**0.6**	**0.8**	**1**
y	1	0	0	0	1

- We have a one-level decision tree classifier that chooses a value of k to minimize leaf node entropy
- If bagging is not used the best the classifier can do is a 20% error rate with a split at $x \leq 0.3$ or $x \leq 0.9$

Bagging Example

- Step 1: Bootstrap samples are taken
- Step 2: one-level decision tree classifiers (base classifiers) are trained on each sample.

Bootstrap Sample							**Base Classifier**
1	x	0.2	0.2	0.4	0.6	1	$x \leq 0.3 \Rightarrow y = 1$
	y	1	1	0	0	1	$otherwise\ y = 0$
2	x	0.2	0.4	0.4	0.6	0.8	$x \leq 0.3 \Rightarrow y = 1$
	y	1	0	0	0	0	$otherwise\ y = 0$
3	x	0.4	0.4	0.6	0.8	1	$x \leq 0.9 \Rightarrow y = 0$
	y	0	0	0	0	1	$otherwise\ y = 1$
4	x	0.2	0.6	0.8	1	1	$x \leq 0.9 \Rightarrow y = 0$
	y	1	0	0	1	1	$otherwise\ y = 1$
5	x	0.2	0.2	1	1	1	$x \leq 0.1 \Rightarrow y = 0$
	y	1	1	1	1	1	$otherwise\ y = 1$

Bagging Example

- Step 3. For each record tally votes and select the majority class. Since we have 0/1 classifier the majority is the proportion of 1's. If the proportion is less than 0.5 then the bagging prediction is 0, otherwise 1.

Bootstrap Sample	$x = 0.2$	$x = 0.4$	$x = 0.6$	$x = 0.8$	$x = 1$
1	1	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	1
4	0	0	0	0	1
5	1	1	1	1	1
Proportion	0.6	0.2	0.2	0.2	0.6
Bagging Prediction	1	0	0	0	1

When to Apply Bagging

“Some classification and regression methods are unstable in the sense that small perturbations in their training sets or in construction may result in large changes in the constructed predictor.” (Breiman 1998)

Classification Algorithm	**Stable or Unstable**
Classification and Regression Trees	Unstable
C4.5	Unstable
Neural Networks	Unstable
k-Nearest Neighbor	Stable
Discriminant Analysis	Stable
Naïve Bayes	Stable

When to Apply Bagging

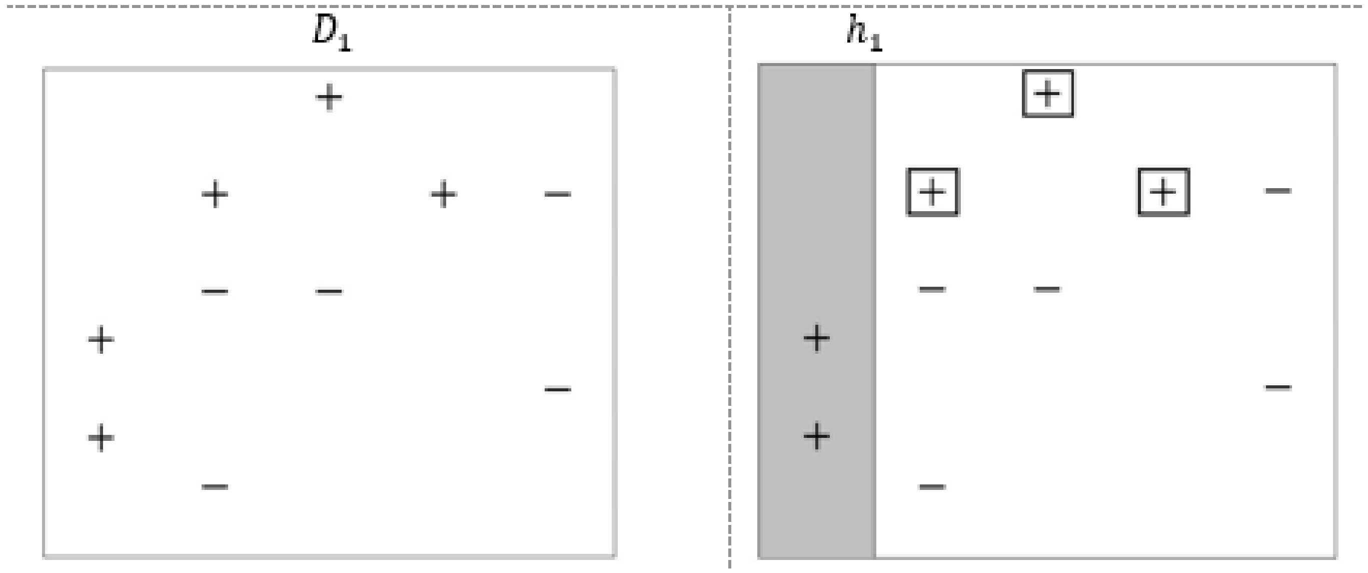
- Bagging works best with unstable models where there is room for improvement in reducing variability as it is a method for reducing variance.
- Applying bagging to stable models can degrade their performance
- Bagging works with bootstrap samples of the original data, each of which contains about 63% of the data.

Boosting

- Reduces both error due to variance and error due to bias
- All observations have equal weight in the original training data set D _1. An initial “base” classifier h_1 is determined.
- The observations that were incorrectly classified by the previous base classifier have their weights increased, while the observations that were correctly classified have their weights decreased. This gives us data distribution D_m, m=2,…,M. A new base classifier h_m, m=2, …, M is determined, based on the new weights. This step is repeated until the desired number of iterations M is achieved.
- The final boosted classifier is the weighted sum of the M base classifiers.

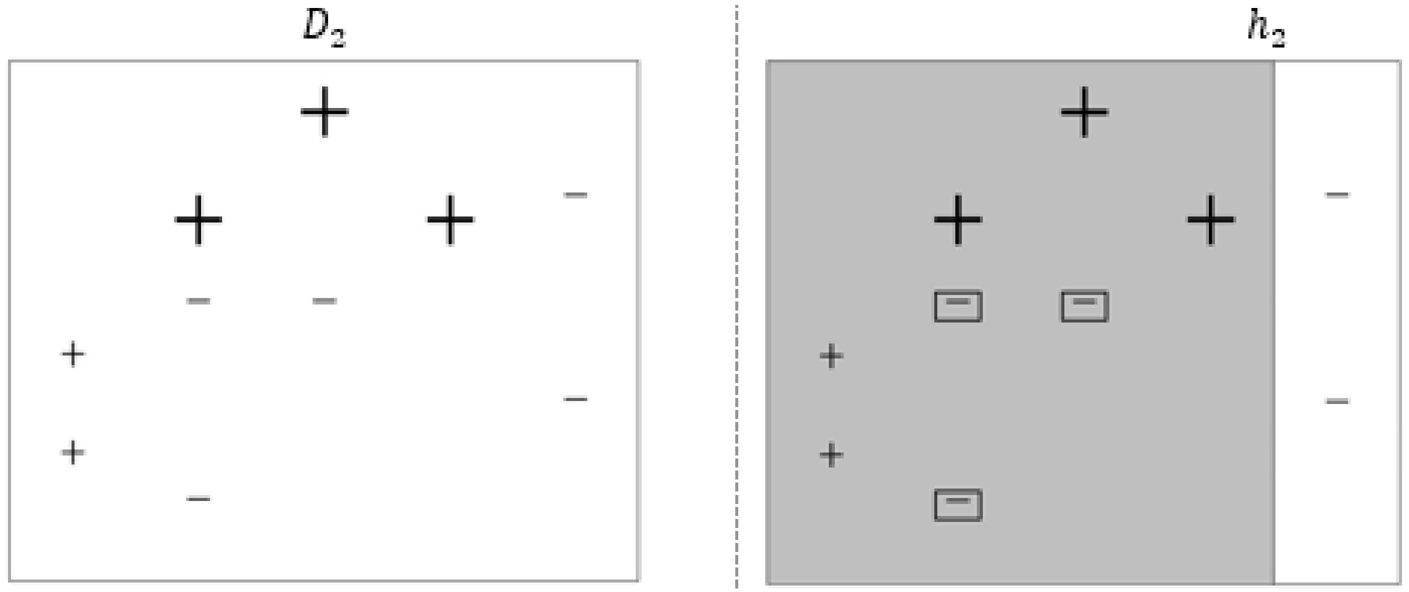
Boosting Step 1

- Training data D1 consists of 10 dichotomous values as shown below. An initial base classifier h1 is determined to separate the two leftmost values. Shaded area represents values classified as “+” boxed values are those incorrectly classified.



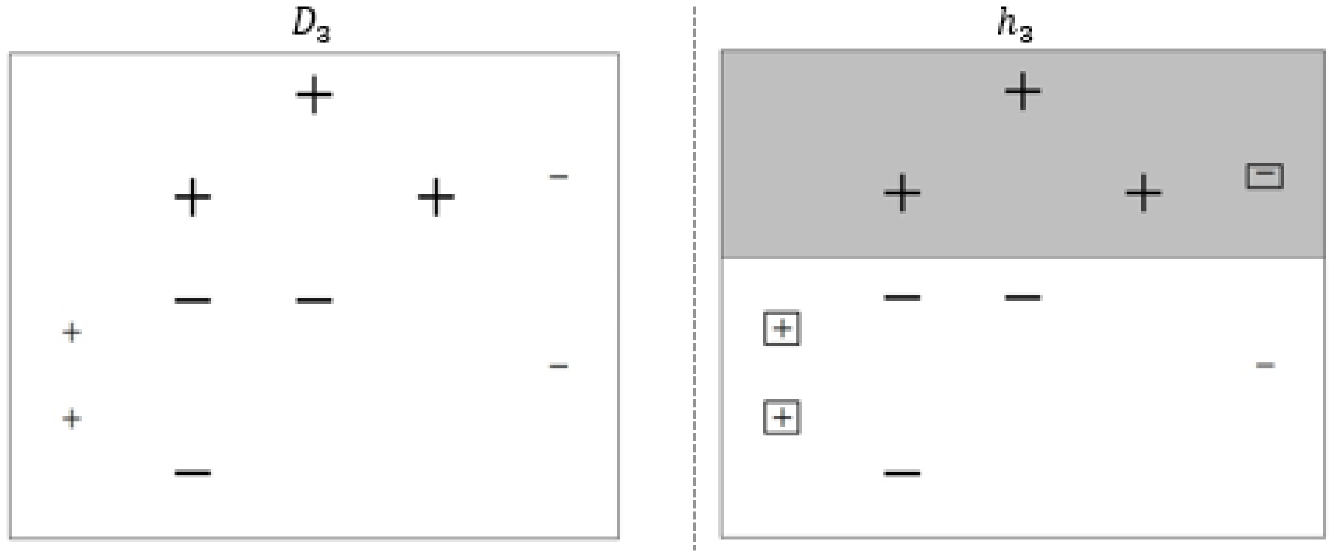
Boosting Step 2 (first pass)

- 3 values incorrectly classified by h1 have weights (represented by relative size in diagrams) increased while other 7 have weights decreased. Based on the new weights a new classifier h2 is determined



Boosting Step 2 (second pass)

- 3 values incorrectly classified by h2 have weights increased while other 7 have weights decreased. Based on the new weights a new classifier h3 is determined



Boosting Step 3

- Final boosted classifier is the weighted sum of the M = 3 base classifiers: α_1 h_1+α_2 h_2+α_3 h_3.

- Weights assigned each classifier proportional to accuracy of classfier
- Boosting performs best when base classifiers area unstable
- Boosting can increase the variance when the base classifier is stable