



2D1431 Machine Learning

BAYESIAN LEARNING, CONT:

Danica Kragic



November 15, 2006

Simple Bayesian Reasoning

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

$$P(D) = \sum_{i=1}^n P(D|h_i)P(h_i)$$

- For each hypothesis h_i we need to know the probability of generating any combination of data D
- The number of all possible data sets is exponential in the number of basic data attributes
- Requires huge amount of data usually not available (sparse data problem)!



Most Probable Classification of New Instances

- So far we've sought the most probable *hypothesis* given the data D (i.e., h_{MAP})
- Assumption: training set consists of instances described as **conjunctions of attribute values**, target classification based on finite set of classes V
- Task: predict the correct class for a new instance $\langle a_1, a_2 \dots a_n \rangle$

Consider:

- Three possible hypotheses:

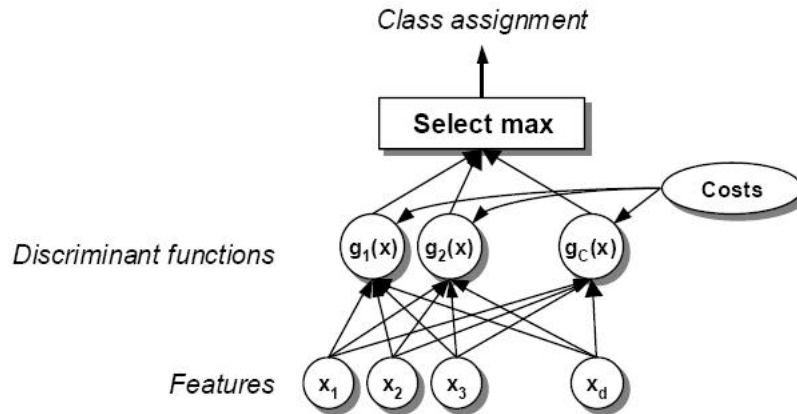
$$P(h_1|D) = .4, P(h_2|D) = .3, P(h_3|D) = .3$$

- Given new instance x ,

$$h_1(x) = +, h_2(x) = -, h_3(x) = -$$

- What's most probable classification of x ?

A classifier





Bayesian Classification

- Instead of asking “What is the most probable hypothesis given training data?”, ask
- “What is the most probable classification of the new instance given training data”
- Instead of learning “hypothesis function”, h , the Bayes optimal classifier assigns the most probable class to the input data

Types:

- Bayes Optimal Classifier
- Gibbs Classifier
- Naive Bayes Classifier
- Bayesian Belief Network (Bayes Net)

Bayes Optimal Classifier

- We want to determine the most probable classification based on the combined prediction of all hypotheses, weighted by their posterior probabilities
- Let V be a set of all possible classifications

$$P(v_j|D) = \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- Bayes Optimal Classification

$$v = \arg \max_{v_j \in V} P(v_j|D) = \arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j|h_i)P(h_i|D)$$

- also known as *model averaging* - averaging predictions of lots of models based on the posterior probability of the model's parameters



Bayes Optimal Classifier

Bayes optimal classification:

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

Example:

$$\begin{aligned} P(h_1 | D) &= .4, & P(-|h_1) &= 0, & P(+|h_1) &= 1 \\ P(h_2 | D) &= .3, & P(-|h_2) &= 1, & P(+|h_2) &= 0 \\ P(h_3 | D) &= .3, & P(-|h_3) &= 1, & P(+|h_3) &= 0 \end{aligned}$$

therefore

$$\sum_{h_i \in H} P(+|h_i) P(h_i | D) = 0.4, \quad \sum_{h_i \in H} P(-|h_i) P(h_i | D) = 0.6$$

and

$$\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D) = -$$



Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:

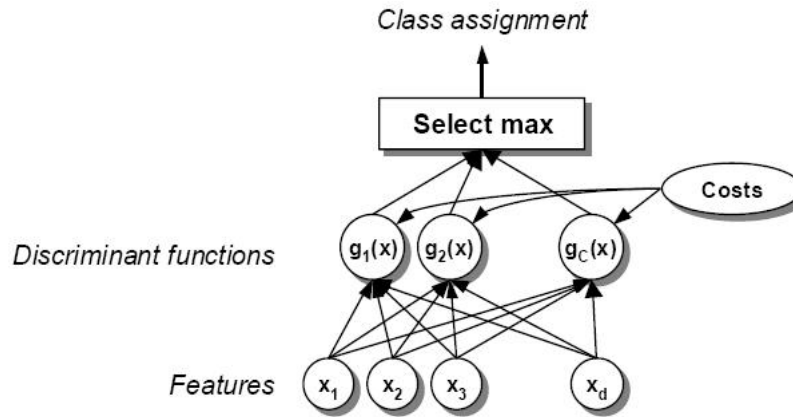
1. Choose one hypothesis at random, according to $P(h|D)$
2. Use this to classify new instance

Surprising fact: Assume target concepts are drawn at random from H according to priors on H . Then:

$$E[\text{error}_{\text{Gibbs}}] \leq 2E[\text{error}_{\text{BayesOptimal}}]$$

- Its expected error no worse than twice Bayes optimal

A classifier



- with $V = \{v_1, \dots, v_n\}$ being categories/labels/classes
- Bayes - $g_j(X) = \sum_{h_i \in H} P(v_j(X) | h_i) P(h_i | X)$
- MAP - $g_j(X) = P(h_{MAP_j} | X)$
- ML - $g_j(X) = P(X | h_{ML_j})$



Naive Bayes Classifier

Along with decision trees, neural networks, nearest nbr, one of the most practical and common learning methods.

When to use

- Moderate or large training set available
- Assumption: attributes that describe instances are conditionally independent given classification (even if violated works well)

Successful applications:

- Diagnosis
- Classifying text documents

Naive Bayes Classifier

Assume discrete target function $f(x) : X \rightarrow V$, where each instance x described by attributes $\langle a_1, a_2 \dots a_n \rangle$ and $f(x)$ takes any value from finite set V .

Most probable target value of h is:

$$\begin{aligned} v_{MAP} &= \operatorname{argmax}_{c_j \in C} P(c_j | a_1, a_2 \dots a_n) \\ &= \operatorname{argmax}_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \\ &= \operatorname{argmax}_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \end{aligned}$$

Naive Bayes assumption:

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

which gives

$$\text{Naive Bayes classifier: } v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

Example

outlook	temp.	humidity	windy	play	outlook	temp.	humidity	windy	play
sunny	hot	high	false	no	sunny	mild	high	false	no
sunny	hot	high	true	no	sunny	cool	normal	false	yes
overcast	hot	high	false	yes	rainy	mild	normal	false	yes
rainy	mild	high	false	yes	sunny	mild	normal	true	yes
rainy	cool	normal	false	yes	overcast	mild	high	true	yes
rainy	cool	normal	true	no	overcast	hot	normal	false	yes
overcast	cool	normal	true	yes	rainy	mild	high	true	no

	outlook		temperature				humidity		windy		play		
	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
	yes	no	yes	no	yes	no	yes	no	yes	no	yes	no	
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								



Naive Bayes: Example

Consider *PlayTennis* again, and new instance

$\langle \text{Outlk} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong} \rangle$

Want to compute:

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(c_j) \prod_i P(a_i | c_j)$$

- playtennis (9+,5-): $P(\text{yes}) = 9/14$, $P(\text{no}) = 5/14$
- wind = strong (3+, 3-): $P(\text{strong} | \text{yes}) = 3/9$, $P(\text{strong} | \text{no}) = 3/5$
- \vdots

$$P(y)P(\text{sun}|y)P(\text{cool}|y)P(\text{high}|y)P(\text{strong}|y) = .005$$

$$P(n)P(\text{sun}|n)P(\text{cool}|n)P(\text{high}|n)P(\text{strong}|n) = .021$$

$$\rightarrow v_{NB} = \text{no}$$

Naive Bayes: Independence Violation

- Conditional independence assumption is often violated

$$P(a_1, a_2 \dots a_n | v_j) = \prod_i P(a_i | v_j)$$

- ...but it works surprisingly well anyway.
- Note! Don't need estimated posteriors $\hat{P}(v_j | x)$ to be correct; need only that

$$\operatorname{argmax}_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(a_i | v_j) = \operatorname{argmax}_{v_j \in V} P(v_j) P(a_1, \dots, a_n | v_j)$$

- Naive Bayes posteriors often unrealistically close to 1 or 0



Naive Bayes: Estimating Probabilities

What if none of the training instances with target value v_j have attribute value a_i ? Then

$$\hat{P}(a_i|v_j) = 0 \quad \text{and} \quad \hat{P}(v_j) \prod_i \hat{P}(a_i|v_j) = 0$$

Typical solution is Bayesian estimate for $\hat{P}(a_i|v_j)$: **m -estimate of probability**

$$\hat{P}(a_i|v_j) \leftarrow \frac{n_v + mp}{n + m}$$

where

- n is the total number of training examples for which $v = v_j$,
- n_v number of examples for which $v = v_j$ and $a = a_i$
- p is prior estimate for $\hat{P}(a_i|v_j)$
- m is weight given to prior - how heavily to weight p relative to the observed data

p - if no additional knowledge, set $p = 1/k$ where k is the number of values an attribute can have



Learning to Classify Email

Target concept *SPAM*? : $\{+, -\}$

1. Represent each email by vector of words
 - One attribute per word position in document
2. Learning: Use training examples to estimate
 - $P(+)$
 - $P(-)$

Naive Bayes conditional independence assumption

$$P(\text{email}|v_j) = \prod_{i=1}^{\text{length}(\text{email})} P(a_i = w_k|v_j)$$

where $P(a_i = w_k|v_j)$ is probability that word in position i is w_k , given v_j

one more assumption: $P(a_i = w_k|v_j) = P(a_m = w_k|v_j), \forall i, m$



Learns: 1) probability terms $P(w_k|v_j)$ - a randomly drawn word from an email of class v_j is w_k , and 2) priors $p(v_j)$.

Learn_naive_Bayes_text(Examples, V)

1. *collect all words and other tokens that occur in Examples*

- *Vocabulary* \leftarrow all distinct words and other tokens in *Examples*

2. *calculate the required $P(v_j)$ and $P(w_k|v_j)$ probability terms*

- For each target value v_j in V do

- *emails_j* \leftarrow subset of *Examples* for which the target value is v_j

- $P(v_j) \leftarrow \frac{|emails_j|}{|Examples|}$

- *Text_j* \leftarrow a single document created by concatenating all members of *emails_j*

- $n \leftarrow$ total number of word positions in all training examples whose target value is v_j

- for each word w_k in *Vocabulary*

- * $n_k \leftarrow$ number of times word w_k occurs in *Text_j*

- * $P(w_k|v_j) \leftarrow \frac{n_k+1}{n+|Vocabulary|}$



Return the estimated target value for the email $Email$. a_i denotes the word found in the i th position within $Email$.

Classify_naive_Bayes_text($Email$)

- $positions \leftarrow$ all word positions in $Email$ that contain tokens found in $Vocabulary$
- Return v_{NB} , where

$$v_{NB} = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_{i \in positions} P(a_i | v_j)$$

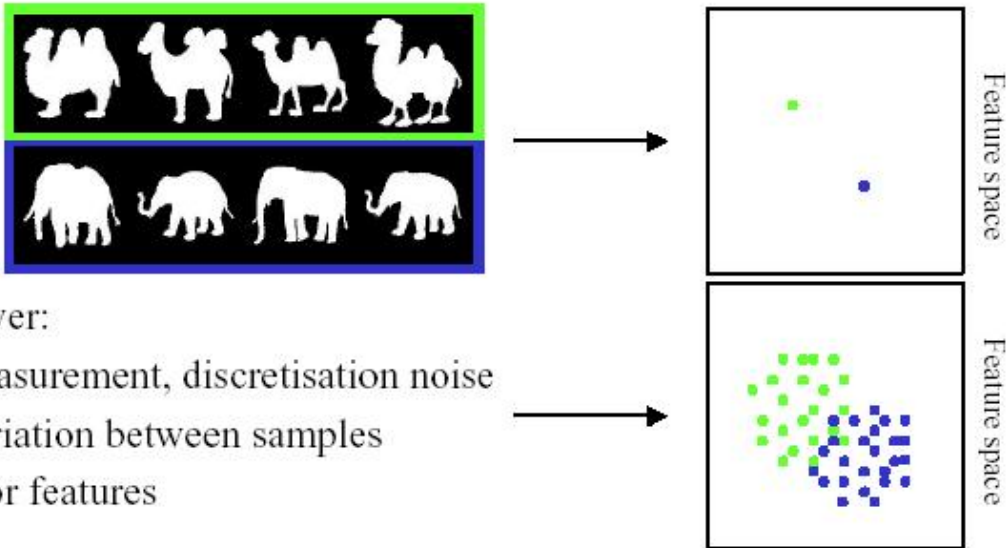


COMBINING CLASSIFIERS:

BAGGING and BOOSTING

Computer Vision

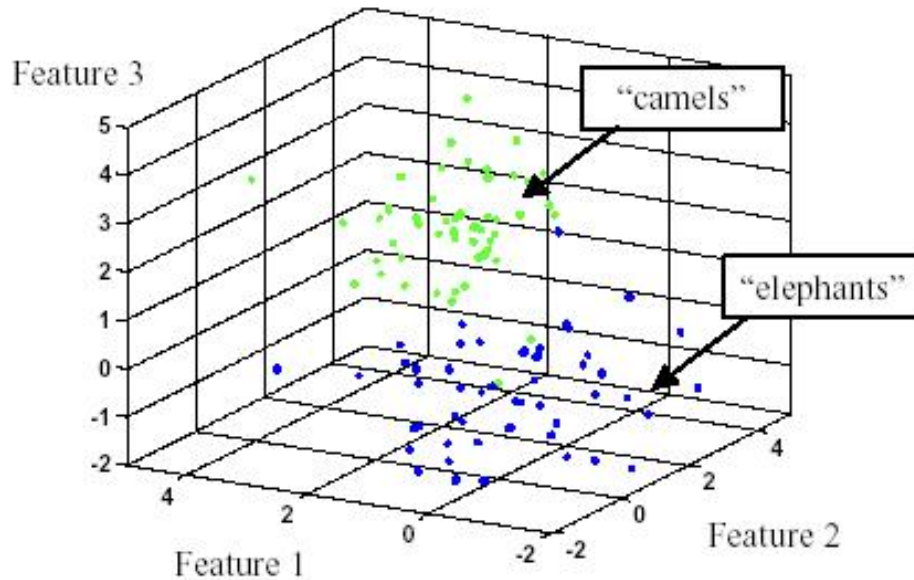
- Sensors give *measurements*, which should be converted to *features* (can be pure measurements, e.g. pixels!)
- Ideally, a feature value is identical for all *samples* in one *class*



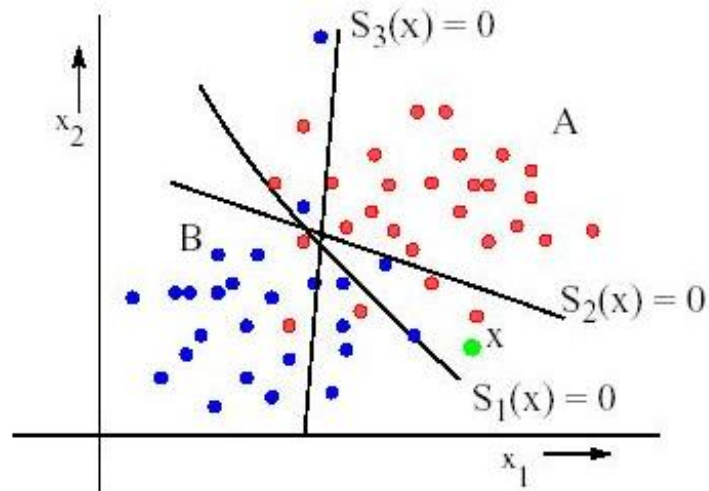
- However:
 - Measurement, discretisation noise
 - Variation between samples
 - Poor features

Coping with “bad features”

- End result: a k -dimensional space,
 - in which each dimension is a **feature**
 - containing N (labelled) **samples** (objects)

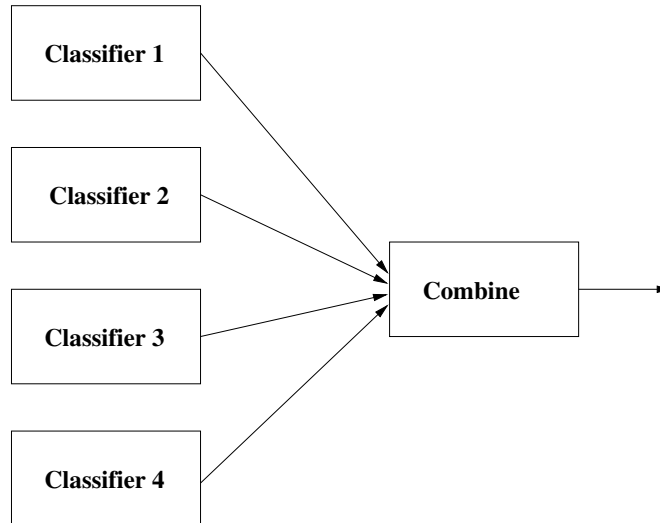


Coping with “bad features”



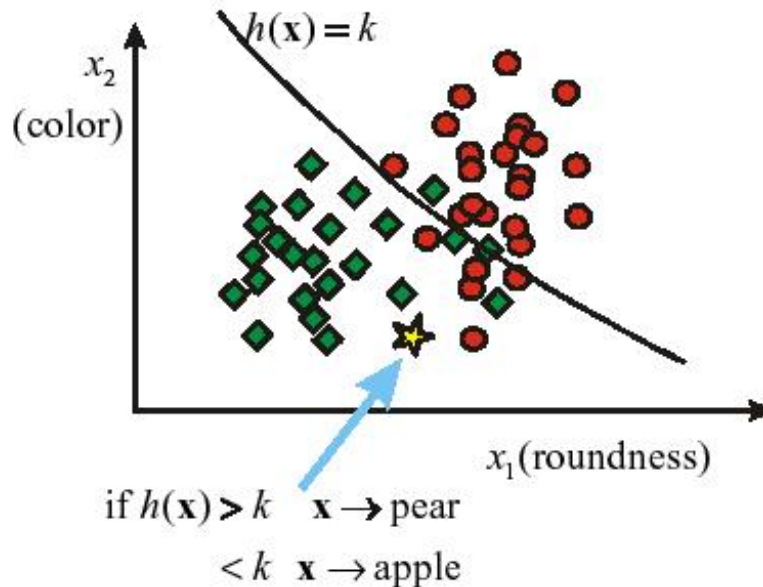
Combine Classifiers

Choice: different training sets, different feature sets,



Discriminant functions

1. Choose class of decision functions in feature space.
2. Estimate the function parameters from the training set.
3. Classify a new pattern on the basis of this decision rule.



Linear Discriminant Functions

$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

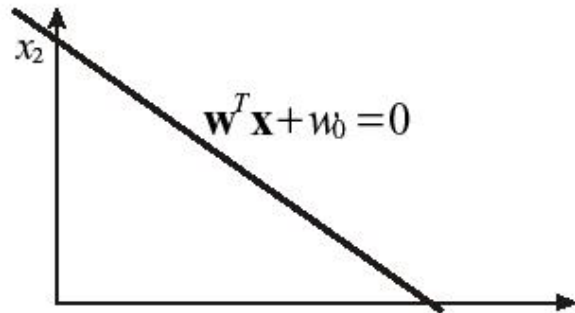
is a linear function of the features \mathbf{x} ,
where \mathbf{w} determines the slope of the
plane and w_0 determines the offset.

If we define

$$\mathbf{z} = (1, x_1, \dots, x_p)^T, \quad \mathbf{v} = (w_0, w_1, \dots, w_p)^T$$

then the function can be written as

$$h(\mathbf{x}) = \mathbf{v}^T \mathbf{z}.$$



Linear Discriminant Functions

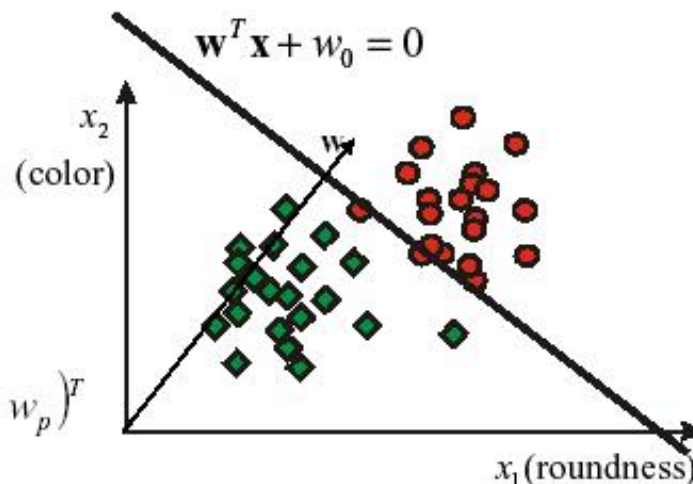
$$h(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \begin{cases} > 0 \\ < 0 \end{cases} \Rightarrow \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

or, shorter,

$$\mathbf{v}^T \mathbf{z} \begin{cases} > 0 \\ < 0 \end{cases} \Rightarrow \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases}$$

with

$$\mathbf{z} = (1, x_1, \dots, x_p)^T, \quad \mathbf{v} = (w_0, w_1, \dots, w_p)^T$$



Webb writes this as follows. Suppose

$\mathbf{y}_i = \mathbf{z}_i$ for $\mathbf{x}_i \in \omega_1$ and $\mathbf{y}_i = -\mathbf{z}_i$ for $\mathbf{x}_i \in \omega_2$

then we seek for a value of \mathbf{v} for which

$\mathbf{v}^T \mathbf{y}_i > 0$ for all \mathbf{y}_i corresponding to the \mathbf{x}_i in the train set.

Classification problem

- Assume a set S of N instances $x_i \in X$ each belonging to one of M classes $\{c^1, \dots, c^M\}$
- The training set consists of pairs $\langle x_i, c_i \rangle$
- A classifier C assigns a classification $C(x) \in \{c^1, \dots, c^M\}$ to an instance x
- The classifier learned in trial t is denoted C^t while C^* is the composite bagged or boosted classifier

Bagging and Boosting

- Bagging and Boosting aggregate multiple hypotheses generated by the same learning algorithm invoked over different distributions of training data [Breiman 1996, Freund and Schapire 1996]
- Bagging and Boosting generate a classifier with a smaller error on the training data as it combines multiple hypotheses which individually have a large error

Bagging and Boosting

- Bagging replicates training sets by sampling with replacement from the training instances.
- Boosting uses all instances but weights them and therefore produces different classifiers.
- Classifiers are then combined by voting to create a composite classifier.
- Bagging: classifiers have same votes;
- Boosting: vote dependant on the classifiers' accuracy

Bagging

- For each trial, generate *new* training set of size N with replacements (multiple occurrence of instances)
- A classifier C^t is generated for each training set
- Final classifier C^* is formed by aggregating the T classifiers
- An instance x is classified by counting votes for which

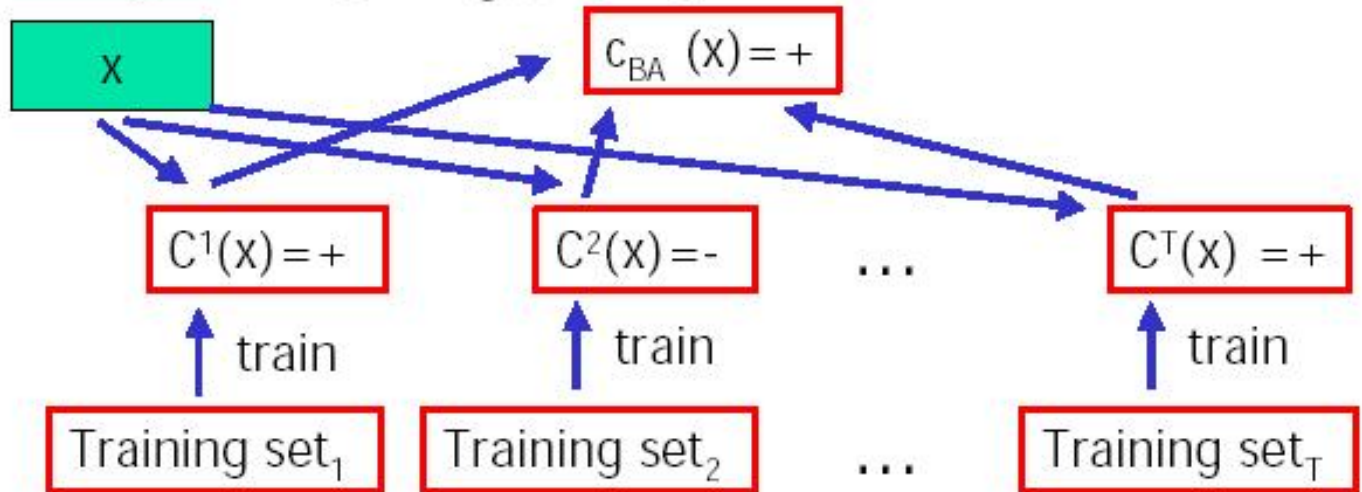
$$C^t(x) = k$$

and $C^*(x)$ represents the class with most votes

Bagging

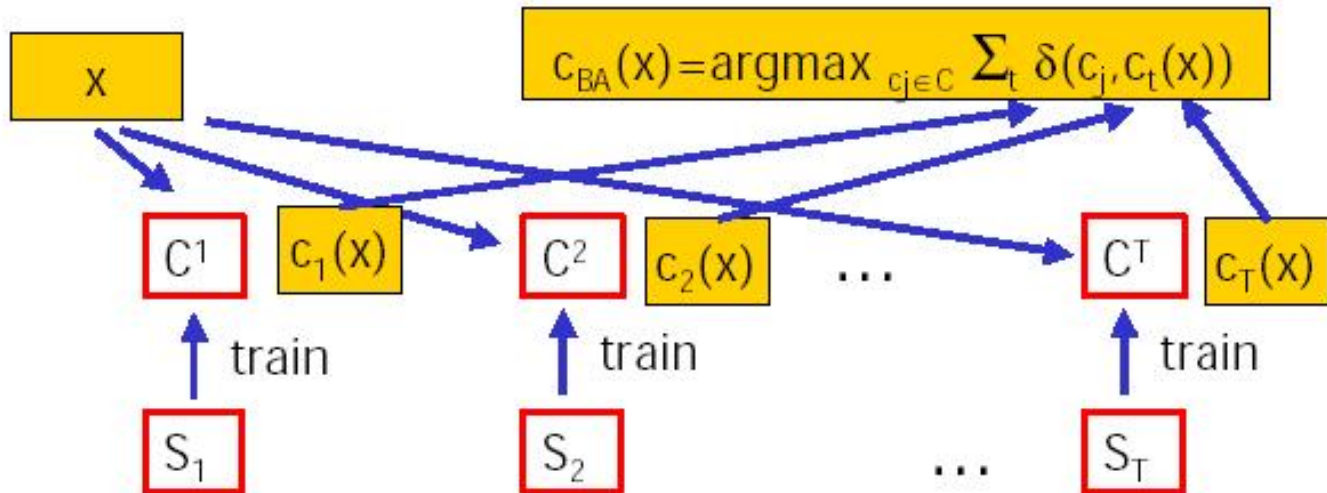
- For each trial $t=1,2,\dots,T$ create a bootstrap sample S^t .
- Obtain an hypothesis C^t on the bootstrap sample S^t .
- The final classification is the majority class

$$c_{BA}(x) = \operatorname{argmax}_{c_j \in C} \sum_t \delta(c_j, C^t(x))$$



Bagging

- From the overall training S set randomly sample (with replacement) T different training sets S_1, \dots, S_T of size N .
- For each sample set S_t obtain a hypothesis C^t .
- To an unseen instance x assign the majority classification $c_{BA}(x)$ among the hypotheses C^t classifications $c_t(x)$.

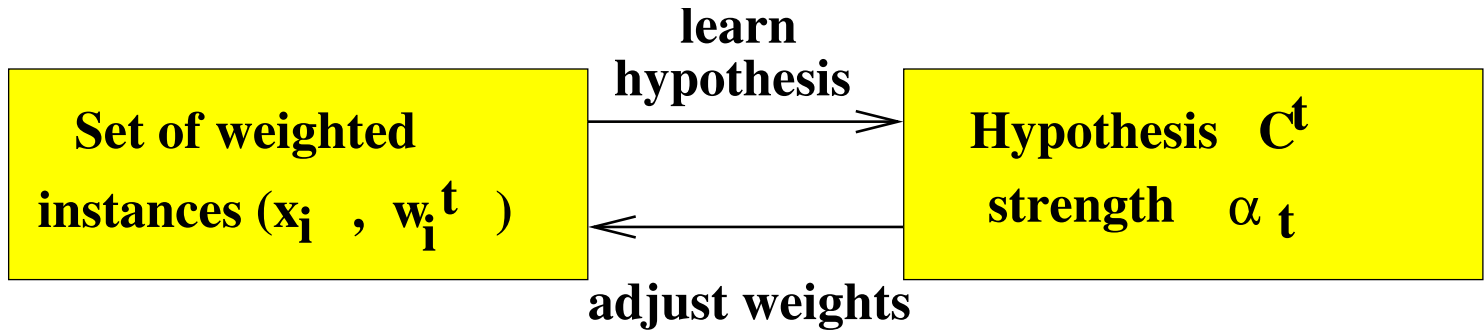


Boosting

- Boosting maintains a weight w_i for each instance $\langle x_i, \dots, c_i \rangle$ in the training set
- The higher the weight w_i , the more the instance x_i influences the next hypotheses learned
- At each trial, the weights are adjusted to reflect the performance of the previously learned hypothesis, with the result that the weight of correctly classified instances is decreased and the weight of incorrectly classified instances is increased

Boosting

- Construct a hypothesis C^t from the current distribution of instances described by w^t
- Adjust the weights according to the classification error ε^t of classifier C^t
- The strength α_t of a hypothesis depends on its training error $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$



Boosting

- The final hypothesis c_{BO} aggregates the individual hypotheses C^t by weighted voting

$$c_{BO}(x) = \operatorname{argmax}_{c_j \in \mathcal{C}} \sum_{t=1}^T \alpha_t \delta(c_j, c_t(x))$$

- Each hypothesis vote is a function of its accuracy

Boosting

- Let w_i^t denote the weight of an instance x_i at trial t , for every x_i , $w_i^1 = 1/N$. The weight w_i^t reflects the importance (e.g. probability of occurrence) of the instance x_i in the sample set S^t
- At each trial $t = 1, \dots, T$, an hypothesis C^t is constructed from the given instances under the distribution w^t . This requires that the learning algorithm can deal with fractional examples.

Boosting

- The error of the hypothesis C^t is measured with respect to the weights

$$\varepsilon_t = \sum_{\forall i \text{ such that } C^t(x_i) \neq c_i} w_i^t / \sum_i w_i^t$$

$$\alpha_t = \frac{1}{2} \ln((1 - \varepsilon_t) / \varepsilon_t)$$

- Update the weights w_i^t of *correctly* and *incorrectly* classified instances by

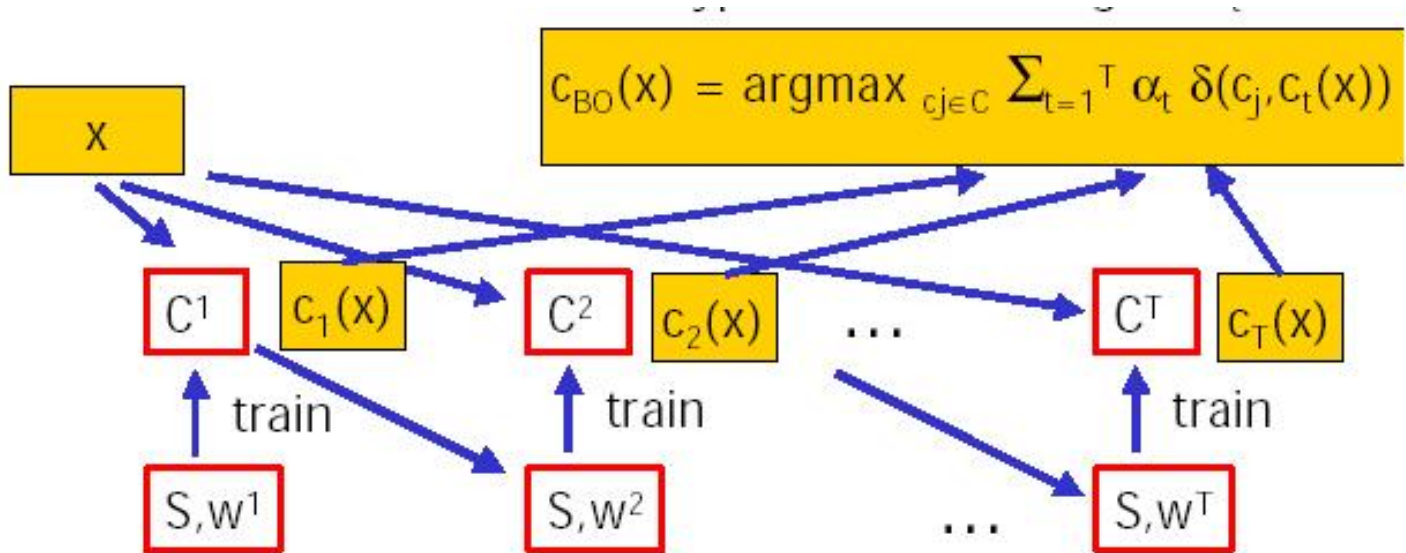
$$w_i^{t+1} = w_i^t e^{-\alpha_t} \text{ if } C^t(x_i) = c_i$$

$$w_i^{t+1} = w_i^t e^{\alpha_t} \text{ if } C^t(x_i) \neq c_i$$

- Afterwards normalize the w_i^{t+1} such that they form a proper distribution $\sum_i w_i^{t+1} = 1$

Boosting

- The classification $c_{BO}(x)$ of the boosted hypothesis is obtained by summing the votes of the hypotheses C^1, C^2, \dots, C^T where the vote of each hypothesis C^t is weight α_t



Boosting

Given $\{\langle x_1, c_1 \rangle, \dots, \langle x_m, c_m \rangle\}$, initialize $w_i^1 = 1/m$

For $t = 1, \dots, T$

train weak learner using distribution w_i^t

get weak hypothesis $C^t : X \rightarrow C$ with error

$$\varepsilon_t = \sum_{\forall i \text{ such that } C^t(x_i) \neq c_i} w_i^t / \sum_i w_i^t$$

choose $\alpha_t = \frac{1}{2} \ln((1 - \varepsilon_t)/\varepsilon_t)$

update

$$w_i^{t+1} = w_i^t e^{-\alpha_t} \text{ if } C^t(x_i) = c_i$$

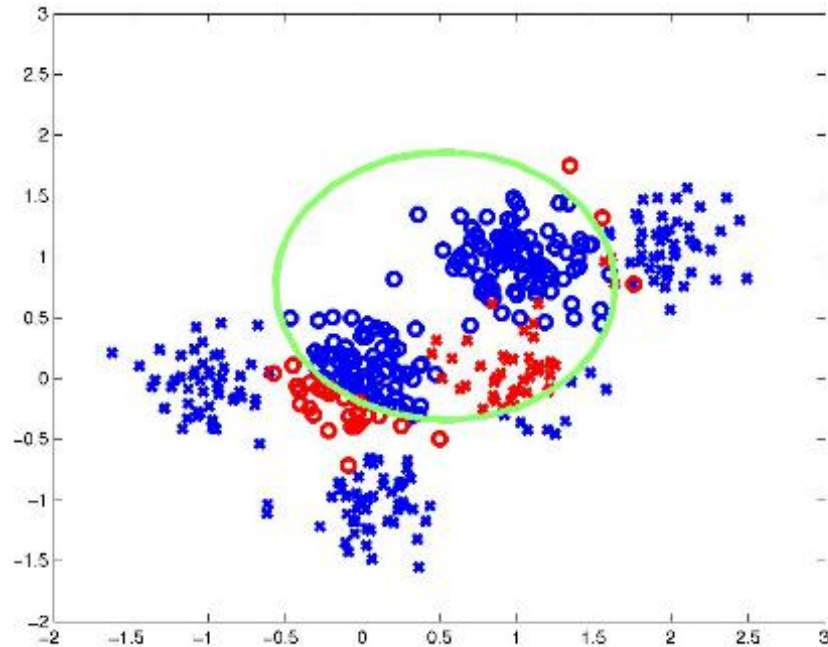
$$w_i^{t+1} = w_i^t e^{\alpha_t} \text{ if } C^t(x_i) \neq c_i$$

Output the final hypothesis

$$c_{BO}(x) = \operatorname{argmax}_{c_j \in C} \sum_{t=1}^T \alpha_t \delta(c_j, c_t(x))$$

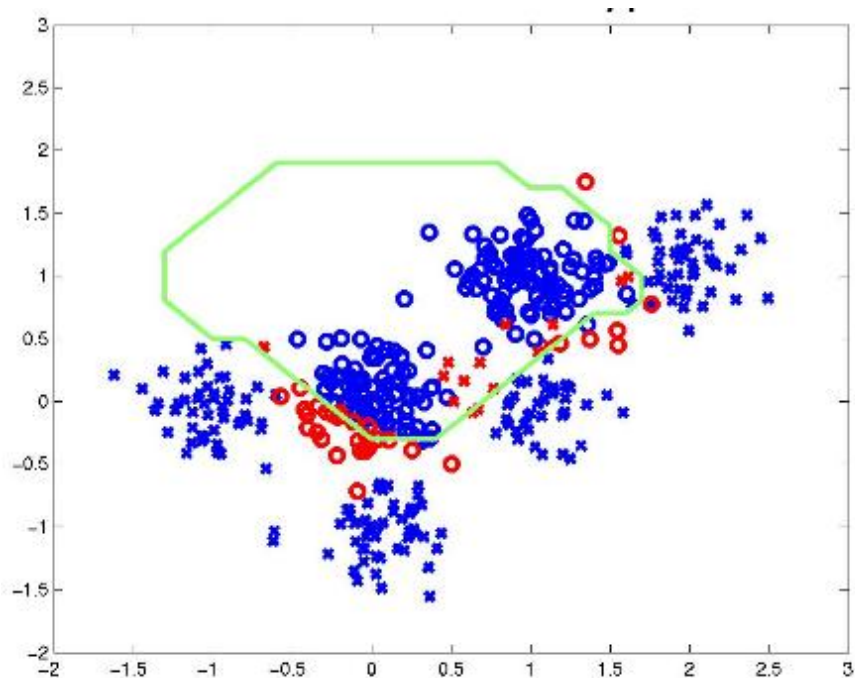
Bayes MAP Hypothesis

- Bayes MAP hypothesis for two classes x and o
- red: incorrect classified instances

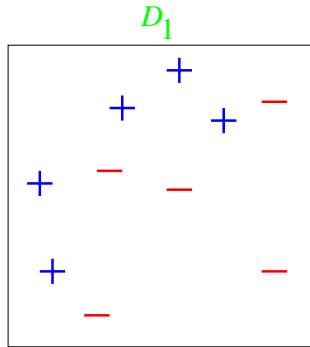


Boosted Bayes MAP Hypothesis

- Boosted Bayes MAP hypothesis has more complex decision surface than individual hypothesis alone

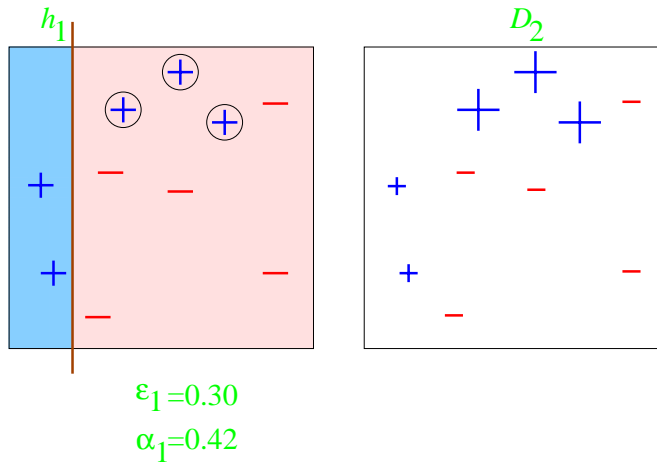


Toy Example

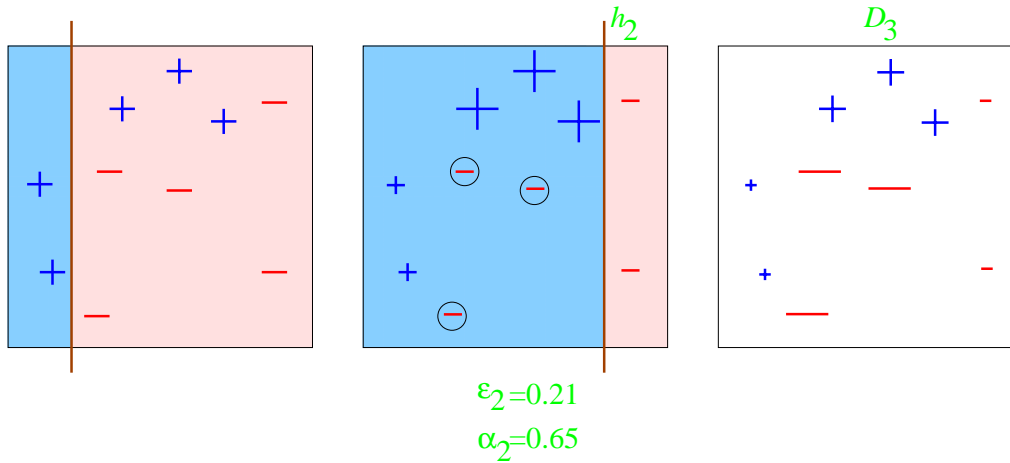


weak classifiers = vertical or horizontal half-planes

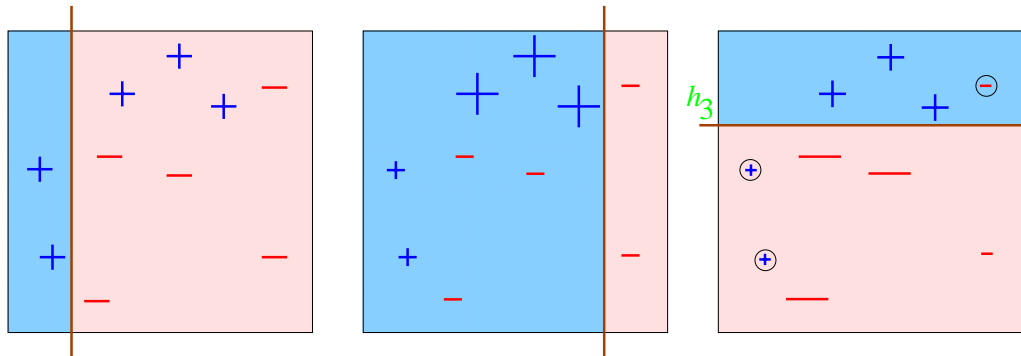
Round 1



Round 2



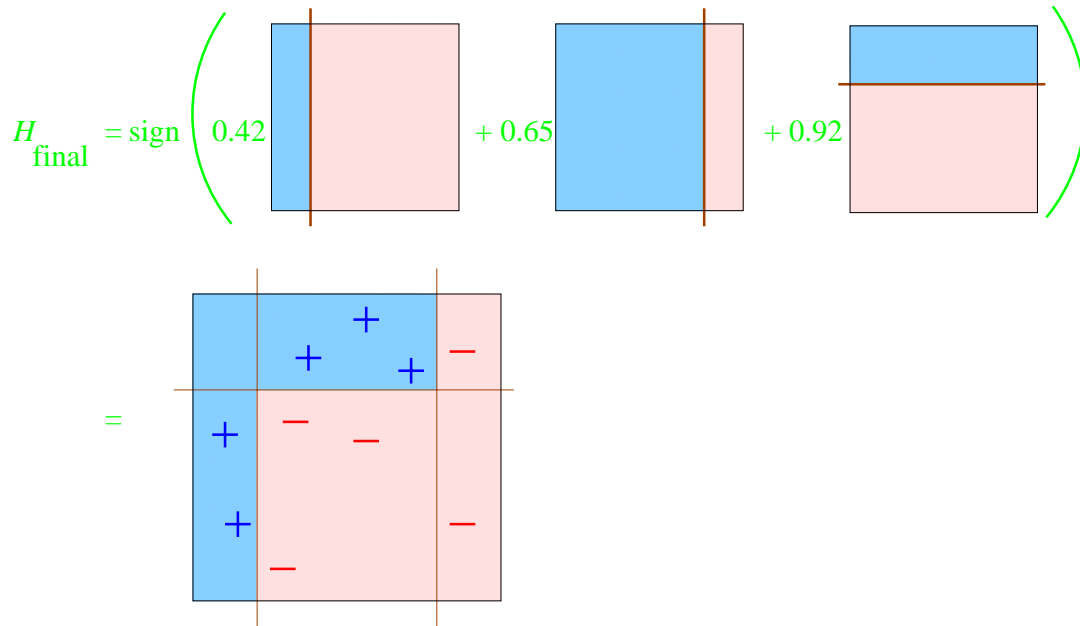
Round 3



$$\epsilon_3 = 0.14$$

$$\alpha_3 = 0.92$$

Final Classifier





Solution to the Prince and King problem

Under assumption that prince chooses door A initially and that king opens door B

The a priori probability that the princess is behind any door X, $P(X) = 1/3$

The probability that King opens door B if the princess was behind A, $P(\text{King opens B} \mid A) = 1/2$

The probability that King opens door B if the princess was behind B, $P(\text{King opens B} \mid B) = 0$

The probability that King opens door B if the princess was behind C, $P(\text{King opens B} \mid C) = 1$

The probability that King opens door B is then

$$p(\text{King opens B}) = p(A) \cdot p(\text{K.o. B} \mid A) + p(B) \cdot p(\text{K.o. B} \mid B) + p(C) \cdot p(\text{K.o. B} \mid C) = 1/6 + 0 + 1/3 = 1/2$$

- Then, by Bayes' Theorem,

$$P(A \mid \text{K.o. B}) = p(A) \cdot p(\text{K.o. B} \mid A) / p(\text{K.o. B}) = (1/6) / (1/2) = 1/3$$

$$P(C \mid \text{K.o. B}) = p(C) \cdot p(\text{K.o. B} \mid C) / p(\text{K.o. B}) = (1/3) / (1/2) = 2/3$$

In other words, the probability that the princess is behind door C is higher when King opens door B, and prince SHOULD switch!

