

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

**Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών
“Πληροφορικής”**



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Οικονομικό Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Οι εικόνες προέρχονται



Σκοποί ενότητας

Εισαγωγή και εξοικείωση με τις μεθόδους k-nn, regression, logistic regression, decision trees.

Περιεχόμενα ενότητας

- Introduction to supervised learning
- Regression
- Naïve Bayes
- K-nn
- Decision Trees
- Regression re-visited

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Introduction to supervised learning

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

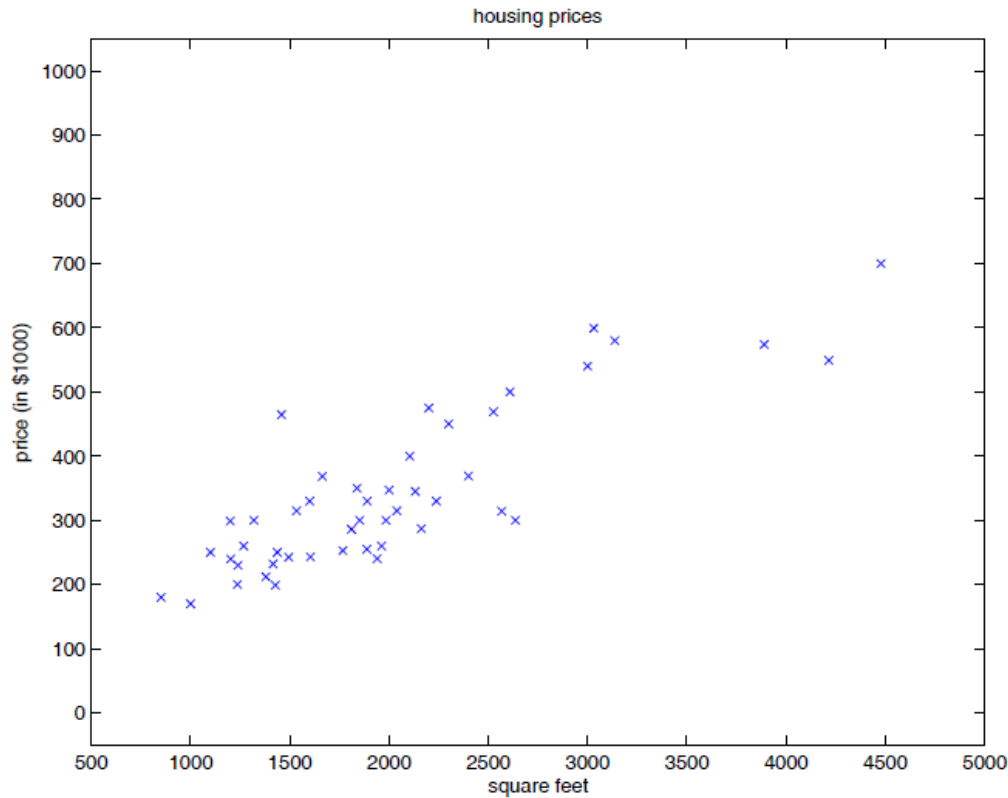
Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

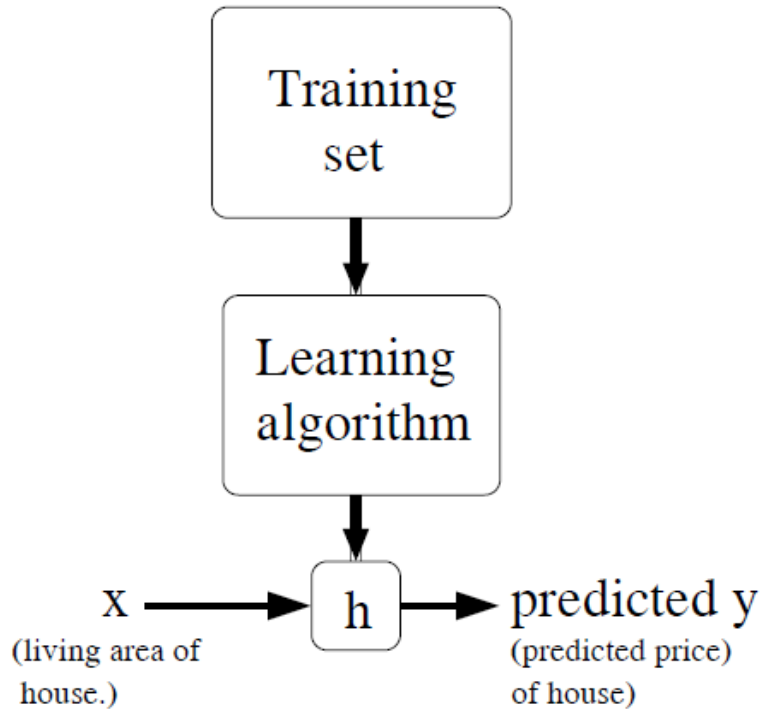
Prediction..

$\{(x_1, y_1), \dots\}$



- Can we predict the price of a house based on its size (surface in m^2) ?

Prediction..



- y continuous value: ***prediction***
- y discrete value: ***classification***

Prediction..

- $x(i)$: “input” variables (input features)
 - $y(i)$: “output” or target variable that we are trying to predict
 - A pair $(x(i), y(i))$ is called a training example,
 - The *training set* - a list of m training examples $\{(x(i), y(i)); i = 1, \dots, m\}$ —is called a training set.
 - X : space of input values, Y : output values.
 - The supervised learning problem:
 - given a training set,
 - learn a function $\mathbf{h} : \mathbf{X} \rightarrow \mathbf{Y}$ so that $\mathbf{h}(\mathbf{x})$ is a “good” predictor for the corresponding value of \mathbf{y} .
- For historical reasons, this function \mathbf{h} is called a hypothesis.

Classes of classifiers

- Class-conditional/probabilistic, based on $p(\underline{x} | c_k)$,
 - Naïve Bayes (simple, but often effective in high dimensions)
 - Parametric generative models, e.g., Gaussian (can be effective in low-dimensional problems: leads to quadratic boundaries in general)
- Regression-based, $p(c_k | \underline{x})$ directly
 - Logistic regression: simple, linear in “odds” space
 - Neural network: non-linear extension of logistic, can be difficult to work with
- Discriminative models, focus on locating optimal decision boundaries
 - Linear discriminants, perceptrons: simple, sometimes effective
 - Support vector machines: generalization of linear discriminants, can be quite effective, computational complexity is an issue
 - Nearest neighbor: simple, can scale poorly in high dimensions
 - Decision trees: “swiss army knife”, often effective in high dimensions

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Regression

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

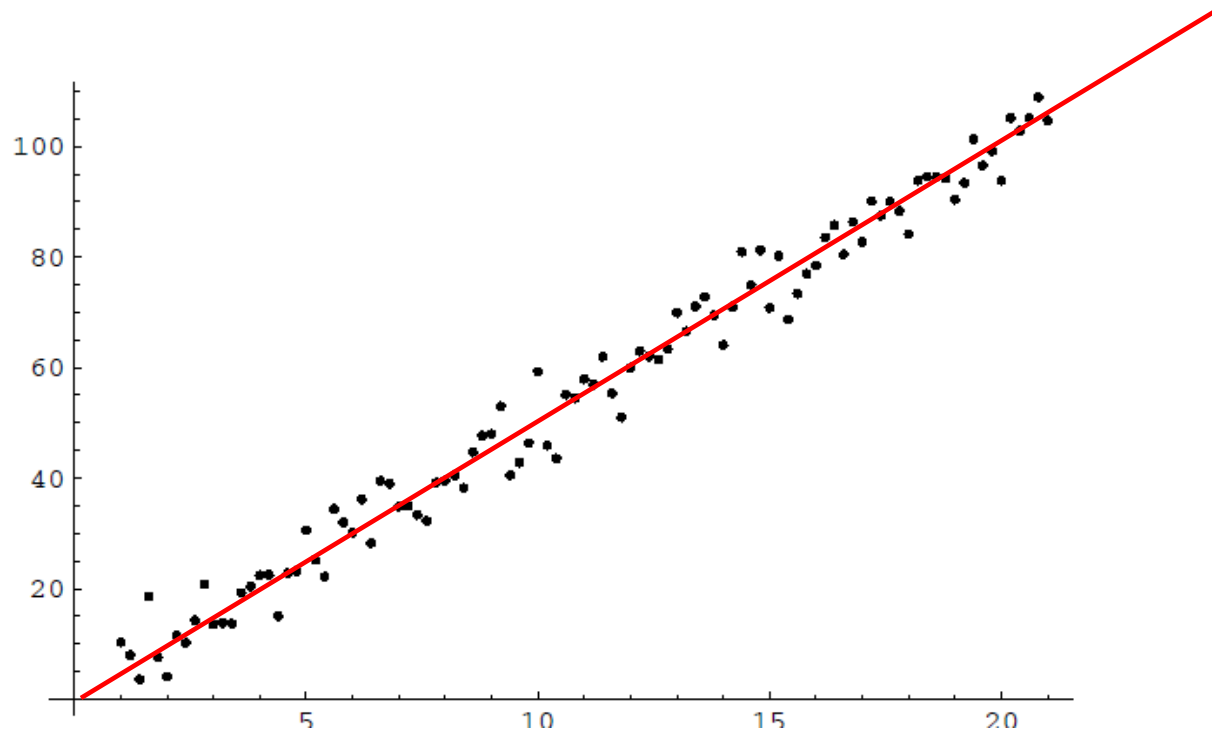
Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

Regression

- Aims at fitting a line to a set of observations $\{(x_1, y_1), \dots, (x_N, y_N)\}$, there is a straight line $y = ax + b$.



Regression

- the individual point error is: $y - (ax + b)$
- thus the error set is: $\{y_1 - (ax_1 + b), \dots, y_N - (ax_N + b)\}$.
- the total error is:
$$E(a, b) = \sum_{n=1}^N (y_n - (ax_n + b))^2 .$$

Least Squares method

- The objective is to minimize $E(a, b) = \sum_{n=1}^N (y_n - (ax_n + b))^2$.
- Thus to find values α, b such that: $\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0$.

- Differentiation leads to: $\frac{\partial E}{\partial a} = \sum_{n=1}^N 2(y_n - (ax_n + b)) \cdot (-x_n)$

(proof?)

$$\frac{\partial E}{\partial b} = \sum_{n=1}^N 2(y_n - (ax_n + b)) \cdot 1.$$

Least Squares method

- Thus setting $\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0.$
- Leads to:
$$\sum_{n=1}^N (y_n - (ax_n + b)) \cdot x_n = 0$$
$$\sum_{n=1}^N (y_n - (ax_n + b)) = 0.$$

- Or equivalently:

$$\left(\sum_{n=1}^N x_n^2 \right) a + \left(\sum_{n=1}^N x_n \right) b = \sum_{n=1}^N x_n y_n$$
$$\left(\sum_{n=1}^N x_n \right) a + \left(\sum_{n=1}^N 1 \right) b = \sum_{n=1}^N y_n.$$

Least Squares method

- Or equivalently:

$$\begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N x_n y_n \\ \sum_{n=1}^N y_n \end{pmatrix}$$

- Implying:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^N x_n y_n \\ \sum_{n=1}^N y_n \end{pmatrix}$$

Least Squares method

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^N x_n y_n \\ \sum_{n=1}^N y_n \end{pmatrix}$$

Solution exists only if $\begin{pmatrix} \sum_{n=1}^N x_n^2 & \sum_{n=1}^N x_n \\ \sum_{n=1}^N x_n & \sum_{n=1}^N 1 \end{pmatrix}$ is invertible

- i.e. if its determinant is **not** 0 – prove it!

Least Squares method

The method is generalized in a straight forward way:
Assume $y = af(x) + bg(x)$ then the respective result is:

$$\begin{pmatrix} \sum_{n=1}^N f(x_n)^2 & \sum_{n=1}^N f(x_n)g(x_n) \\ \sum_{n=1}^N f(x_n)g(x_n) & \sum_{n=1}^N g(x_n)^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^N f(x_n)y_n \\ \sum_{n=1}^N g(x_n)y_n \end{pmatrix}$$

Exercise: Under what conditions is the matrix invertible?

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Naïve Bayes

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

Bayesian Classification: Why?

- Probabilistic learning: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- Probabilistic prediction: Predict multiple hypotheses, weighted by their probabilities
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian classification

- The classification problem may be formalized using **a-posteriori probabilities**:
- $P(C|X)$ = prob. that the sample tuple $X = \langle x_1, \dots, x_k \rangle$ is of class C .
- E.g. $P(\text{class}=\text{N} \mid \text{outlook}=\text{sunny}, \text{windy}=\text{true}, \dots)$
- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative freq of class C samples
- C such that $P(C|X)$ is maximum =
C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!

Naïve Bayesian Classification

- Naïve assumption: **attribute independence**

$$P(x_1, \dots, x_k | C) = P(x_1 | C) \cdot \dots \cdot P(x_k | C)$$

- If i-th attribute is **categorical**:
 $P(x_i | C)$ is estimated as the relative freq of samples having value x_i as i-th attribute in class C
- If i-th attribute is **continuous**:
 - Real-valued variables discretized to create nominal versions
 - $P(x_i | C)$ is estimated thru a Gaussian density function
- Computationally feasible in both cases
- Generative probabilistic model with conditional independence assumption on $p(\underline{x} | c_k)$, i.e.

$$p(\underline{x} | c_k) = \prod p(x_j | c_k)$$

- Typically used with nominal variables
 - (alternative is to model each $p(x_j | c_k)$ with a parametric model – less widely used)

Naïve Bayes Classifiers

- Comments:
 - Simple to train (just estimate conditional probabilities for each feature-class pair)
 - Often works surprisingly well in practice
 - e.g., state of the art for text-classification, basis of many widely used spam filters
 - Feature selection can be helpful, e.g., information gain
 - Note that even if CI assumptions are not met, it may still be able to approximate the optimal decision boundaries (seems to happen in practice)
 - However... on most problems can usually be beaten with a more complex model (plus more work)

Play-tennis example: estimating $P(x_i | C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

$P(p) = 9/14$
$P(n) = 5/14$

outlook	
$P(\text{sunny} p) = 2/9$	$P(\text{sunny} n) = 3/5$
$P(\text{overcast} p) = 4/9$	$P(\text{overcast} n) = 0$
$P(\text{rain} p) = 3/9$	$P(\text{rain} n) = 2/5$
temperature	
$P(\text{hot} p) = 2/9$	$P(\text{hot} n) = 2/5$
$P(\text{mild} p) = 4/9$	$P(\text{mild} n) = 2/5$
$P(\text{cool} p) = 3/9$	$P(\text{cool} n) = 1/5$
humidity	
$P(\text{high} p) = 3/9$	$P(\text{high} n) = 4/5$
$P(\text{normal} p) = 6/9$	$P(\text{normal} n) = 2/5$
windy	
$P(\text{true} p) = 3/9$	$P(\text{true} n) = 3/5$
$P(\text{false} p) = 6/9$	$P(\text{false} n) = 2/5$

Play-tennis example: classifying X

- An unseen sample $X = \langle \text{rain, hot, high, false} \rangle$
- $P(X | p) \cdot P(p) =$
 $P(\text{rain} | p) \cdot P(\text{hot} | p) \cdot P(\text{high} | p) \cdot P(\text{false} | p) \cdot P(p) =$
 $3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$
- $P(X | n) \cdot P(n) =$
 $P(\text{rain} | n) \cdot P(\text{hot} | n) \cdot P(\text{high} | n) \cdot P(\text{false} | n) \cdot P(n) =$
 $2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = 0.018286$
- Sample X is classified in class n (don't play)

The independence hypothesis...

- ... makes computation possible
- ... yields optimal classifiers when satisfied
- ... but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
 - **Bayesian networks**, that combine Bayesian reasoning with causal relationships between attributes
 - **Decision trees**, that reason on one attribute at the time, considering most important attributes first

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

K-nn

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

Linear Discriminant Classifiers

- Linear Discriminant Analysis (LDA)
 - Earliest known classifier (1936, R.A. Fisher)
 - See section 10.4 for math details
 - Find a projection onto a vector such that means for each class (2 classes) are separated as much as possible (with variances taken into account appropriately)
 - Reduces to a special case of parametric Gaussian classifier in certain situations
 - Many subsequent variations on this basic theme (e.g., regularized LDA)
- Other linear discriminants
 - Decision boundary = $(p-1)$ dimensional hyperplane in p dimensions
 - Perceptron learning algorithms (pre-dated neural networks)
 - Simple “error correction” based learning algorithms
 - SVMs: use a sophisticated “margin” idea for selecting the hyperplane

Nearest Neighbor Classifiers

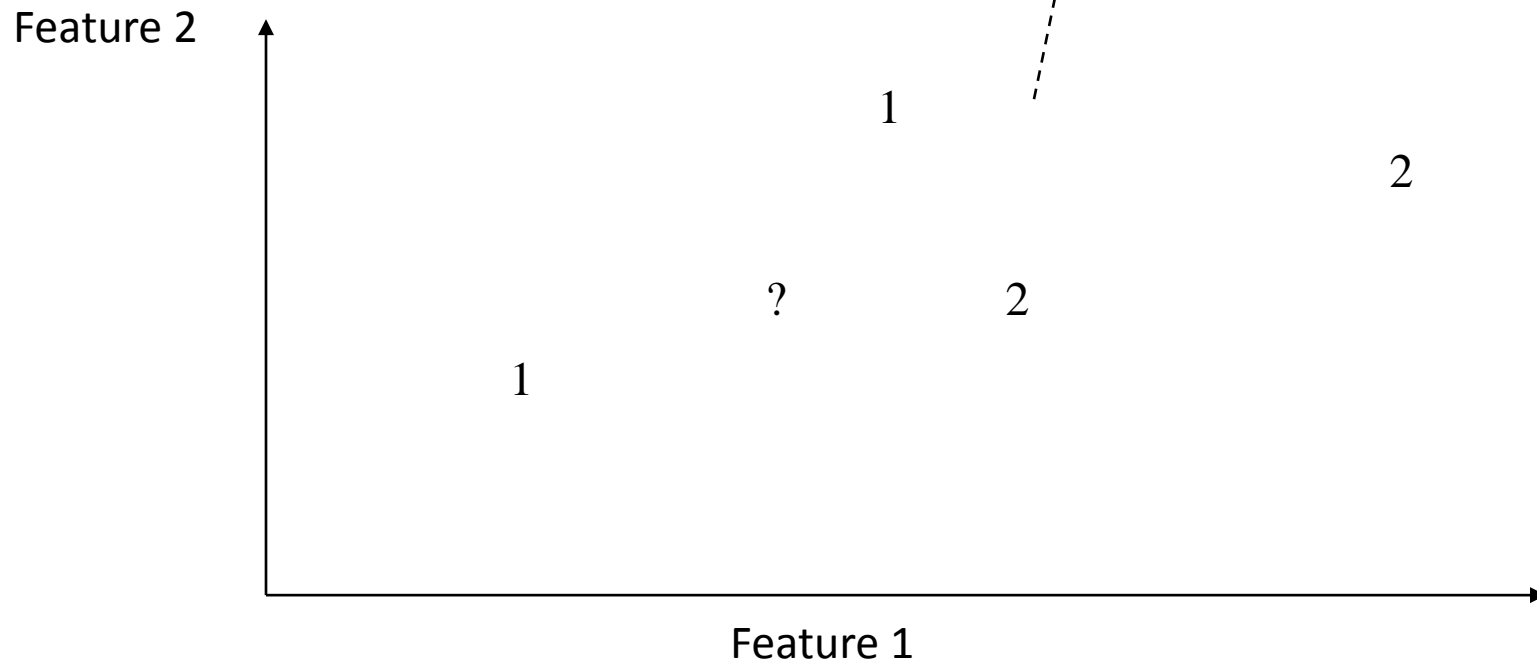
- kNN: select the k nearest neighbors to x from the training data and select the majority class from these neighbors
- k is a parameter:
 - Small k: “noisier” estimates, Large k: “smoother” estimates
 - Best value of k often chosen by cross-validation
- Comments
 - Virtually assumption free
 - Interesting theoretical properties:
Bayes error < error(kNN) < 2 x Bayes error (asymptotically)
- Disadvantages
 - Can scale poorly with dimensionality: sensitive to distance metric
 - Requires fast lookup at run-time to do classification with large n
 - Does not provide any interpretable “model”

Local Decision Boundaries

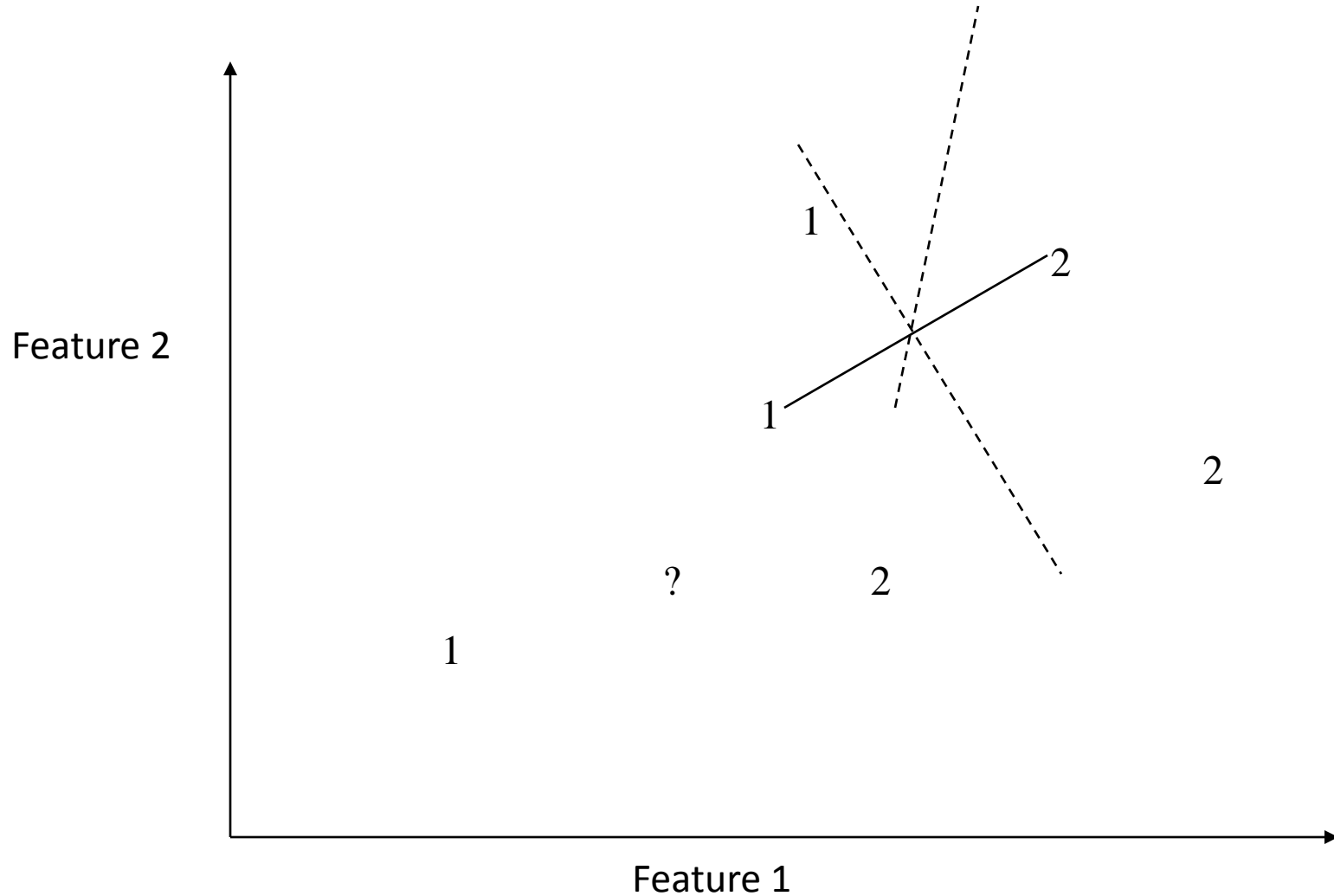
Boundary? Points that are equidistant between points of class 1 and 2

Note: locally the boundary is

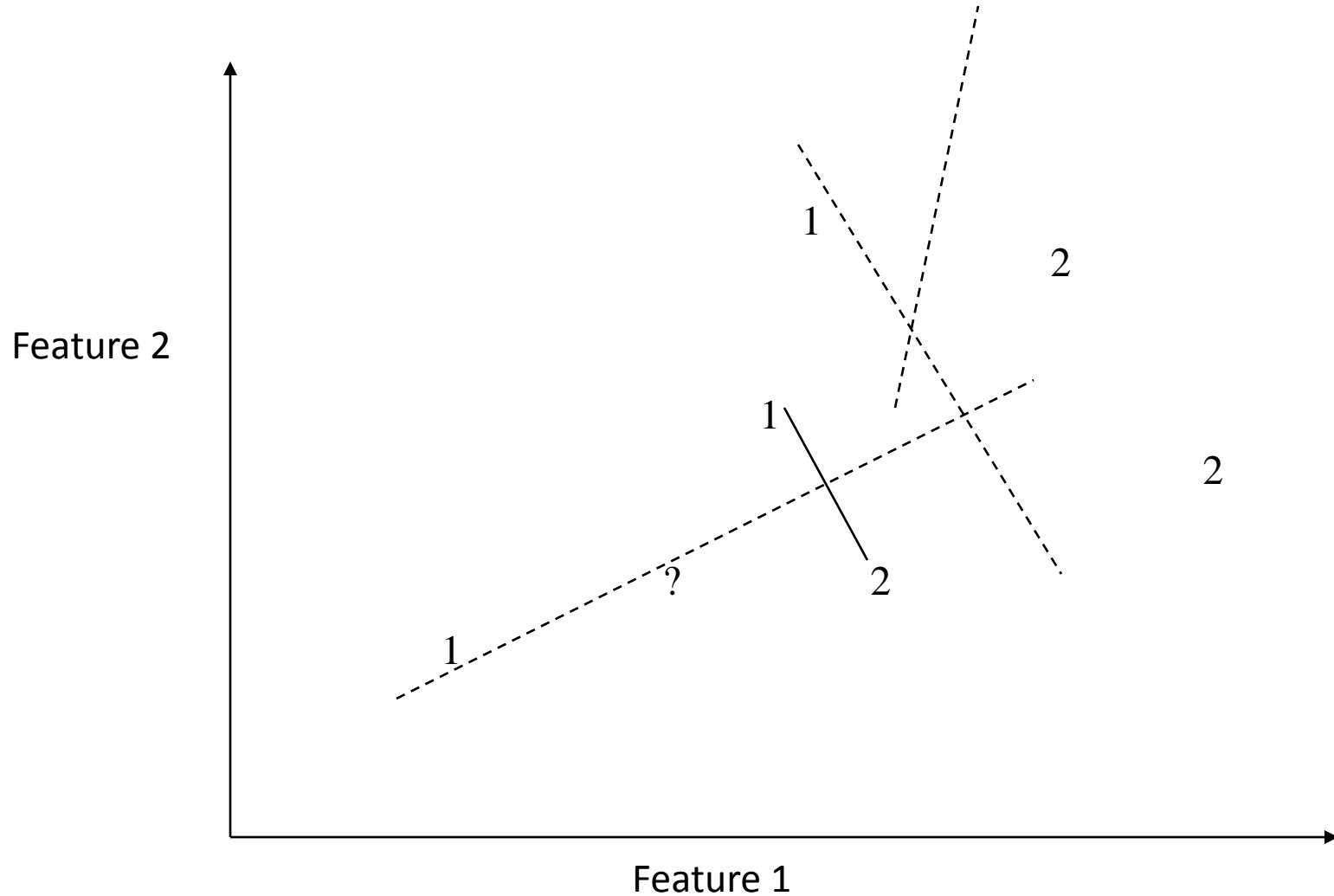
- (1) linear (because of Euclidean distance)
- (2) halfway between the 2 class points
- (3) at right angles to connector



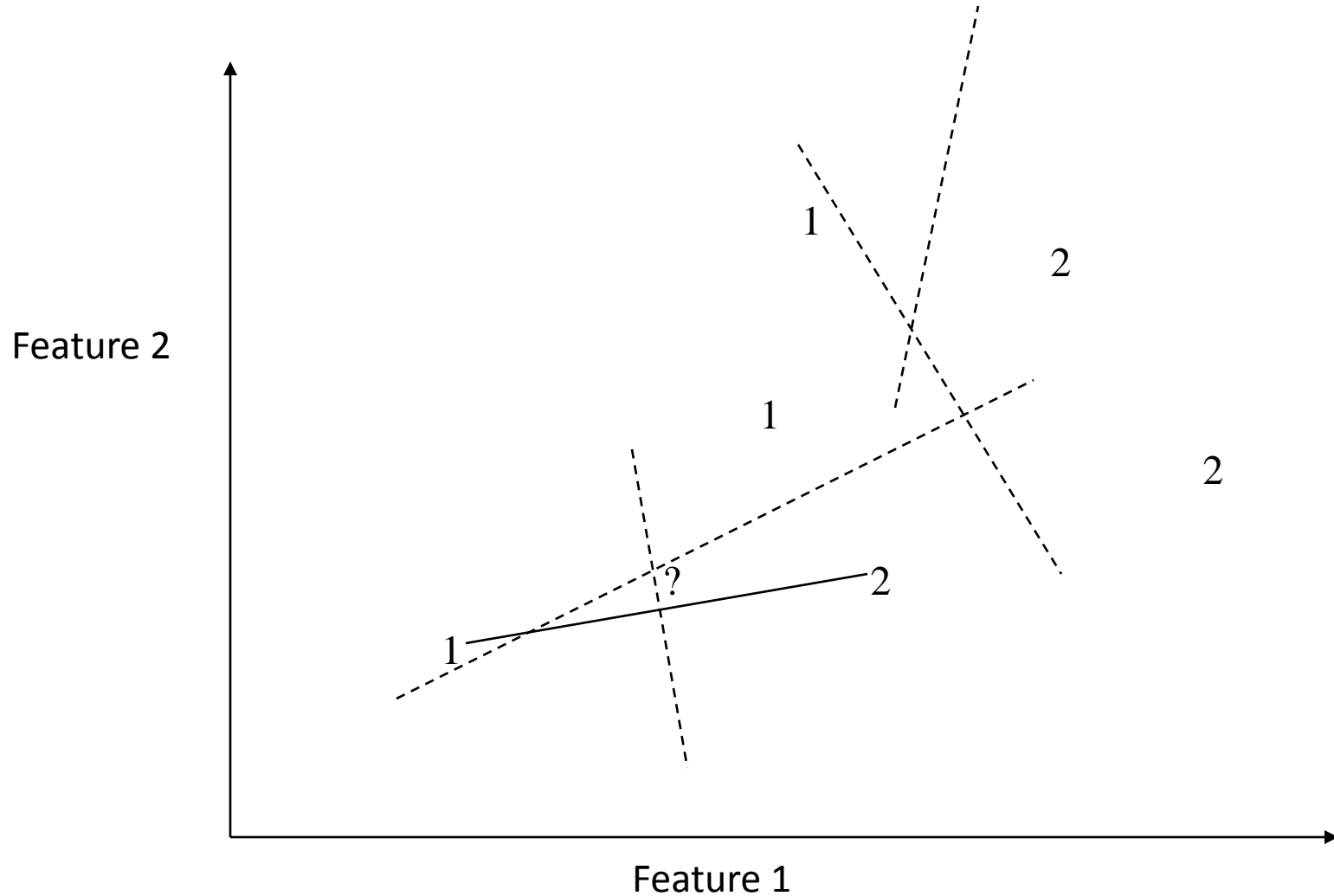
Finding the Decision Boundaries



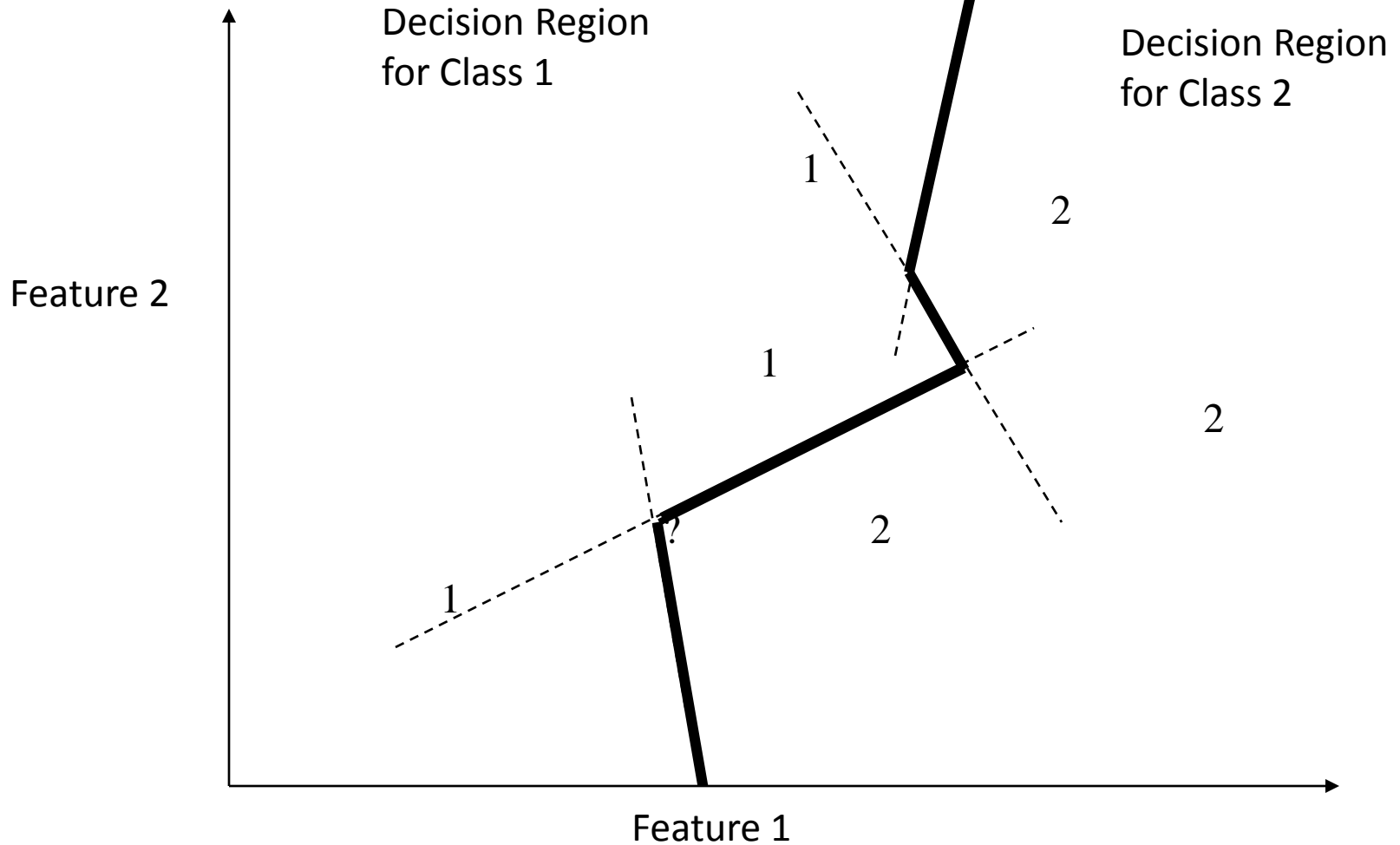
Finding the Decision Boundaries



Finding the Decision Boundaries



Overall Boundary = Piecewise Linear



**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Decision Trees

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

Decision Tree Classifiers

- Widely used in practice
 - Can handle both real-valued and nominal inputs (unusual)
 - Good with high-dimensional data
- similar algorithms as used in constructing regression trees
- historically, developed both in statistics and computer science
 - Statistics:
 - Breiman, Friedman, Olshen and Stone, CART, 1984
 - Computer science:
 - Quinlan, ID3, C4.5 (1980' s-1990' s)

Entropy & Information gain

- **Entropy**: measures the randomness of a statistical variable (or otherwise the information the variable carries)

$$H(x) = \sum_{i=1}^n p(i) \log_2 \left(\frac{1}{p(i)} \right) = - \sum_{i=1}^n p(i) \log_2 p(i).$$

- **Information gain** is the change in entropy from a prior state to a state that takes some information as given:

$$IG(Ex, a) = H(Ex) - H(Ex \mid a)$$

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, P and N
 - Let the set of examples S contain p elements of class P and n elements of class N
 - The amount of information, needed to decide if an arbitrary example in S belongs to P or N is defined as

$$I(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that using attribute A a set S will be partitioned into sets $\{S_1, S_2, \dots, S_v\}$
 - If S_i contains p_i examples of P and n_i examples of N , the **entropy**, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on A

$$Gain(A) = I(p, n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”
- $I(p, n) = I(9, 5) = 0.940$
- Compute the entropy for *age*:

$$E(\text{age}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

$$\text{Gain}(\text{age}) = I(p, n) - E(\text{age})$$

Similarly

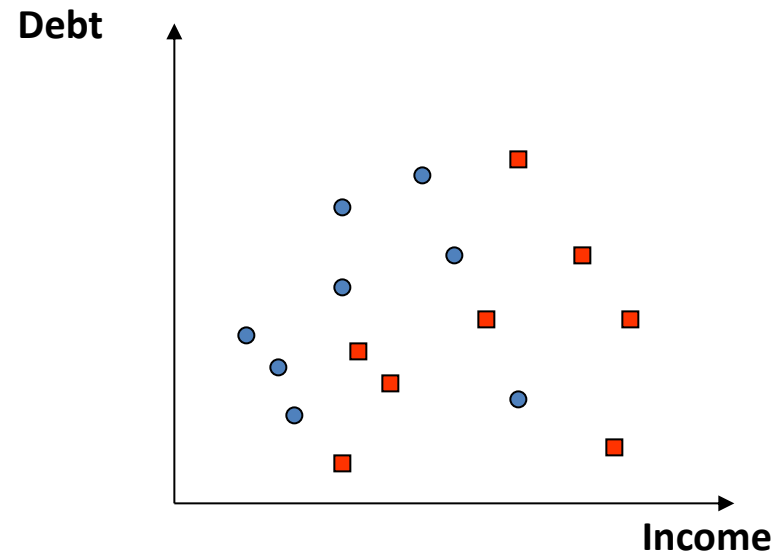
$$\text{Gain}(\text{income}) = 0.029$$

$$\text{Gain}(\text{student}) = 0.151$$

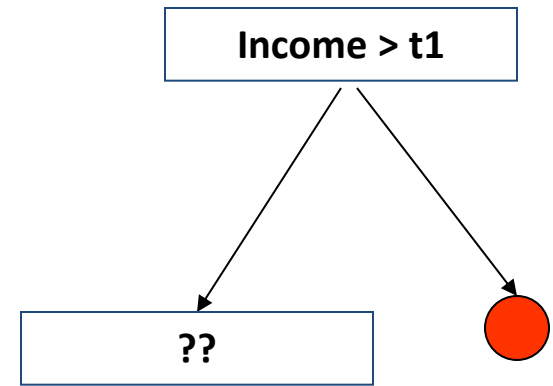
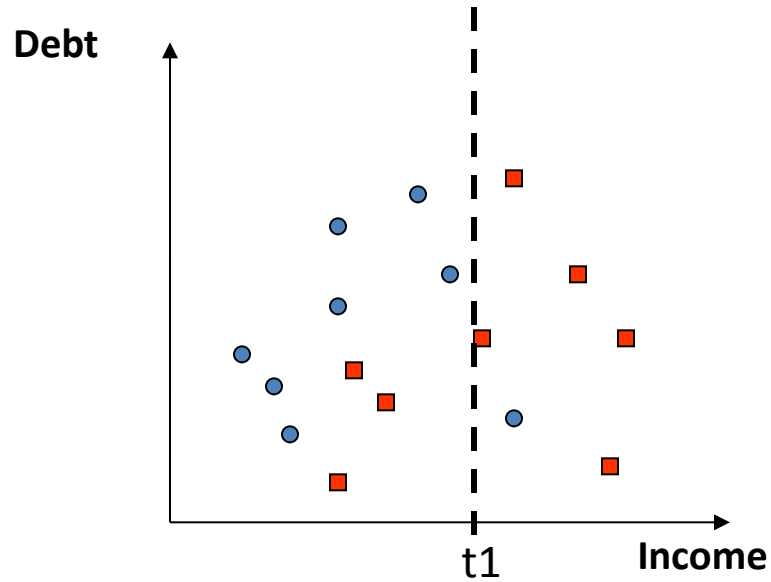
$$\text{Gain}(\text{credit_rating}) = 0.048$$

age	p_i	n_i	$I(p_i, n_i)$
≤ 30	2	3	0.971
30...40	4	0	0
> 40	3	2	0.971

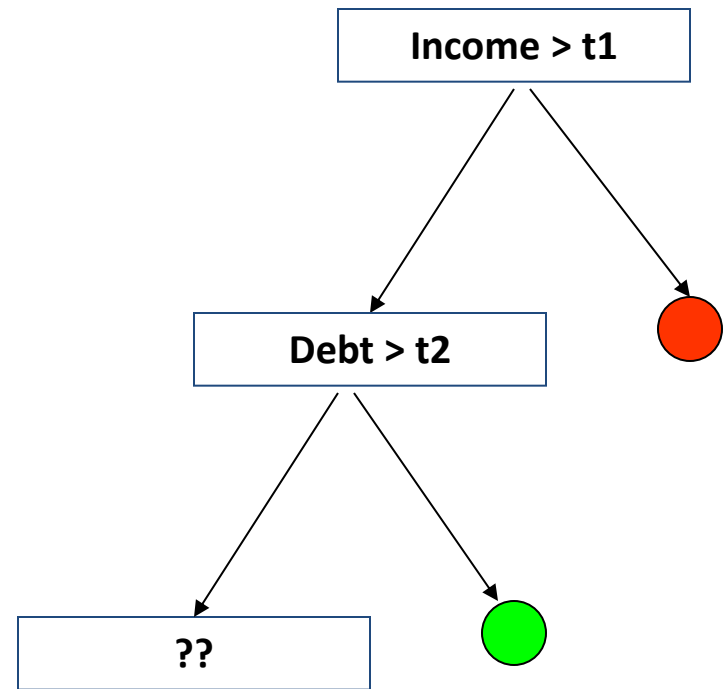
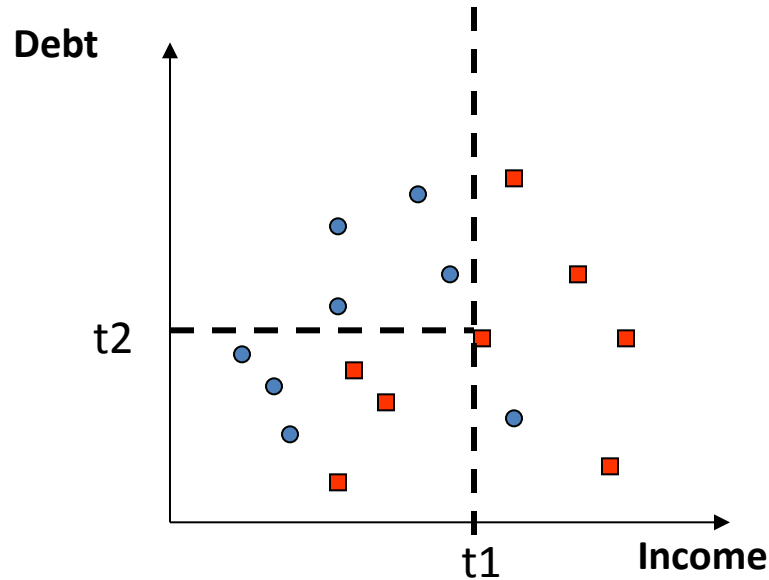
Decision Tree Example



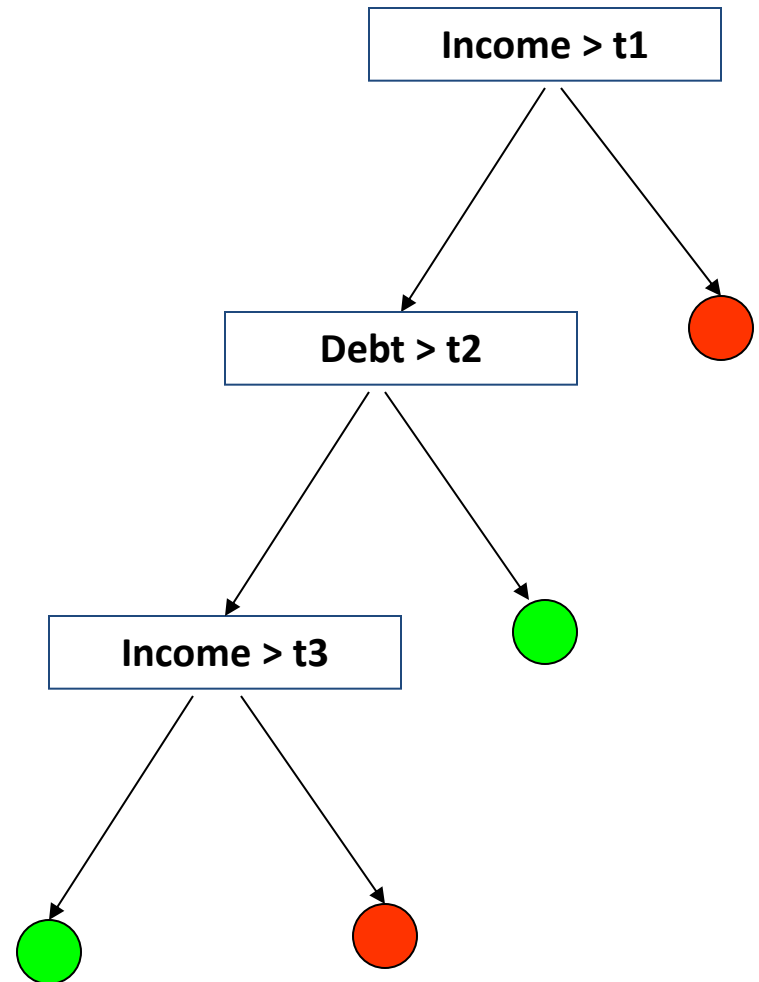
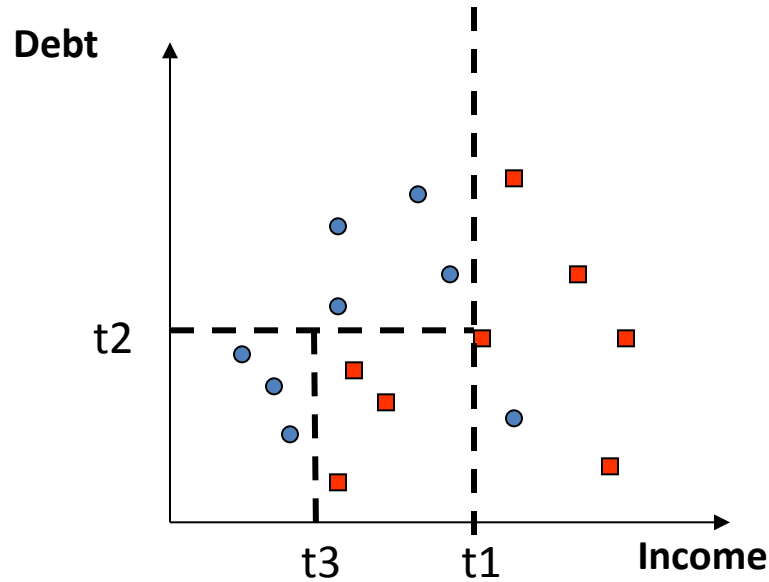
Decision Tree Example



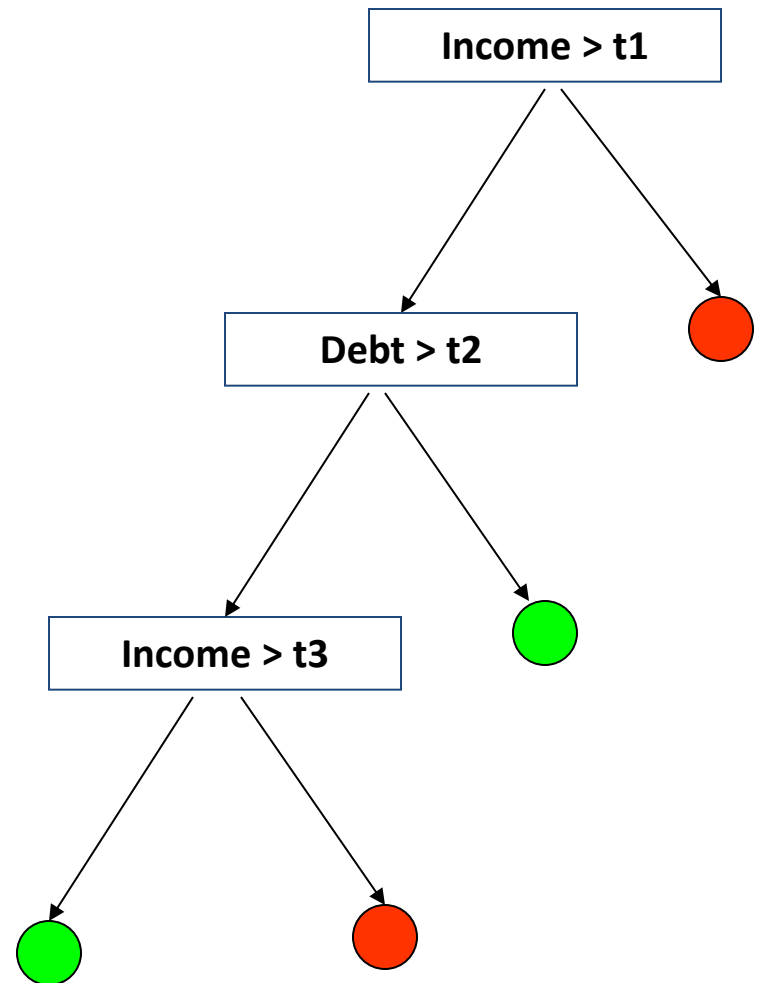
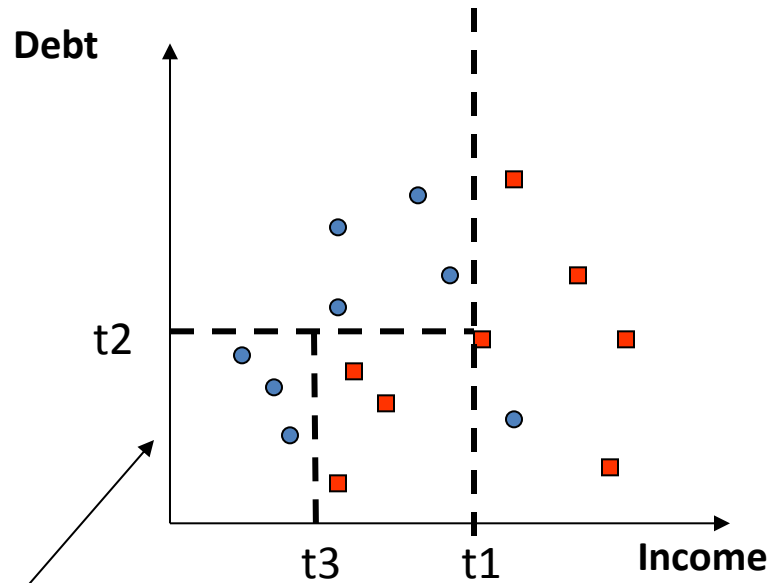
Decision Tree Example



Decision Tree Example

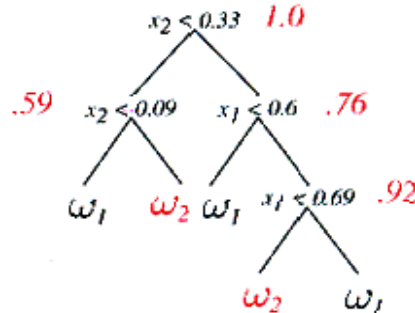
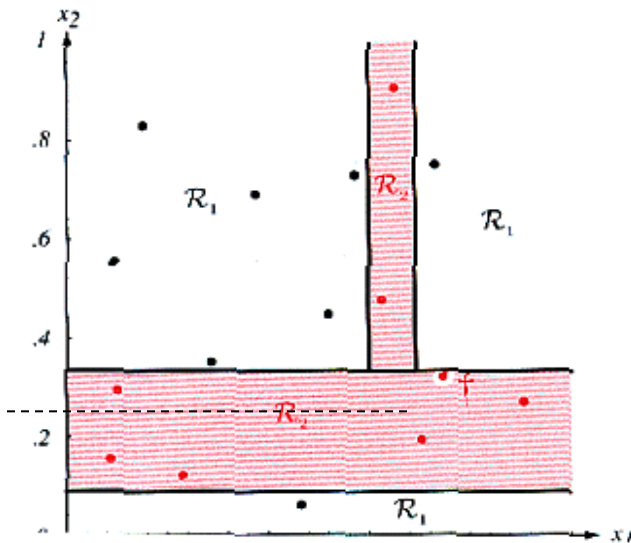
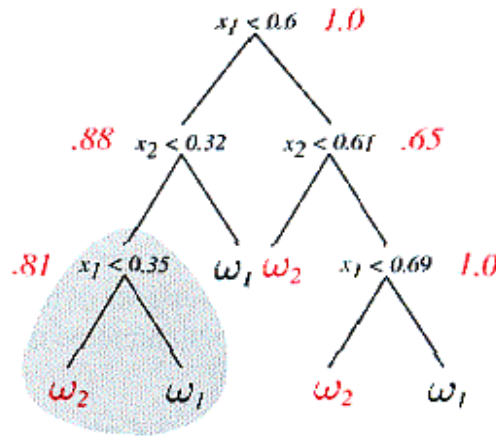
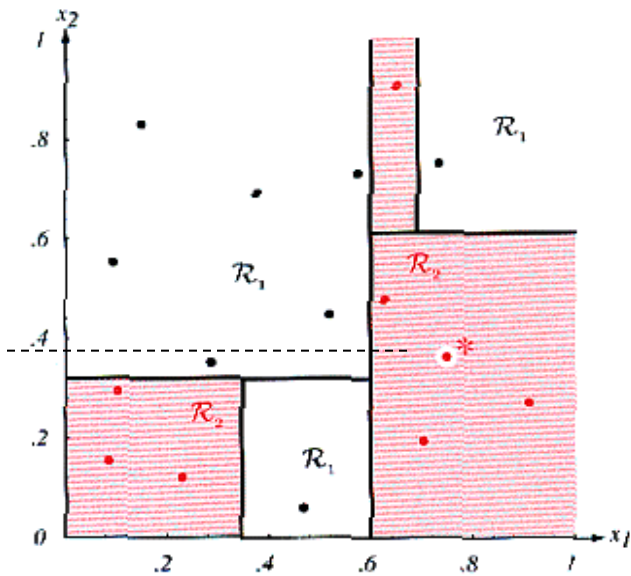


Decision Tree Example



Note: tree boundaries are piecewise linear and axis-parallel

Decision Trees are not stable



Moving just one example slightly may lead to quite different trees and space partition!

Lack of stability against small perturbation of data.

Figure from Duda, Hart & Stork, Chap. 8

Decision Tree Pseudocode

```
node = tree-design (Data = {X,C})
```

```
    For i = 1 to d
```

```
        quality_variable(i) = quality_score(Xi, C)
```

```
    end
```

```
    node = {X_split, Threshold } for max{quality_variable}
```

```
    {Data_right, Data_left} = split(Data, X_split, Threshold)
```

```
    if node == leaf?
```

```
        return(node)
```

```
    else
```

```
        node_right = tree-design(Data_right)
```

```
        node_left = tree-design(Data_left)
```

```
    end
```

```
end
```


Binary split selection criteria

- $Q(t) = N_1 Q_1(t) + N_2 Q_2(t)$, where t is the threshold

- Let p_{1k} be the proportion of class k points in region 1

- Error criterion for a branch

$$Q_1(t) = 1 - p_{1k^*}$$

- Gini index:

$$Q_1(t) = \sum_k p_{1k} (1 - p_{1k})$$

- Cross-entropy:

$$Q_1(t) = \sum_k p_{1k} \log p_{1k}$$

- Cross-entropy and Gini work better in general

- Tend to give higher rank to splits with more extreme class distributions
- Consider [(300,100) (100,300)] split versus [(400,0) (200 200)]

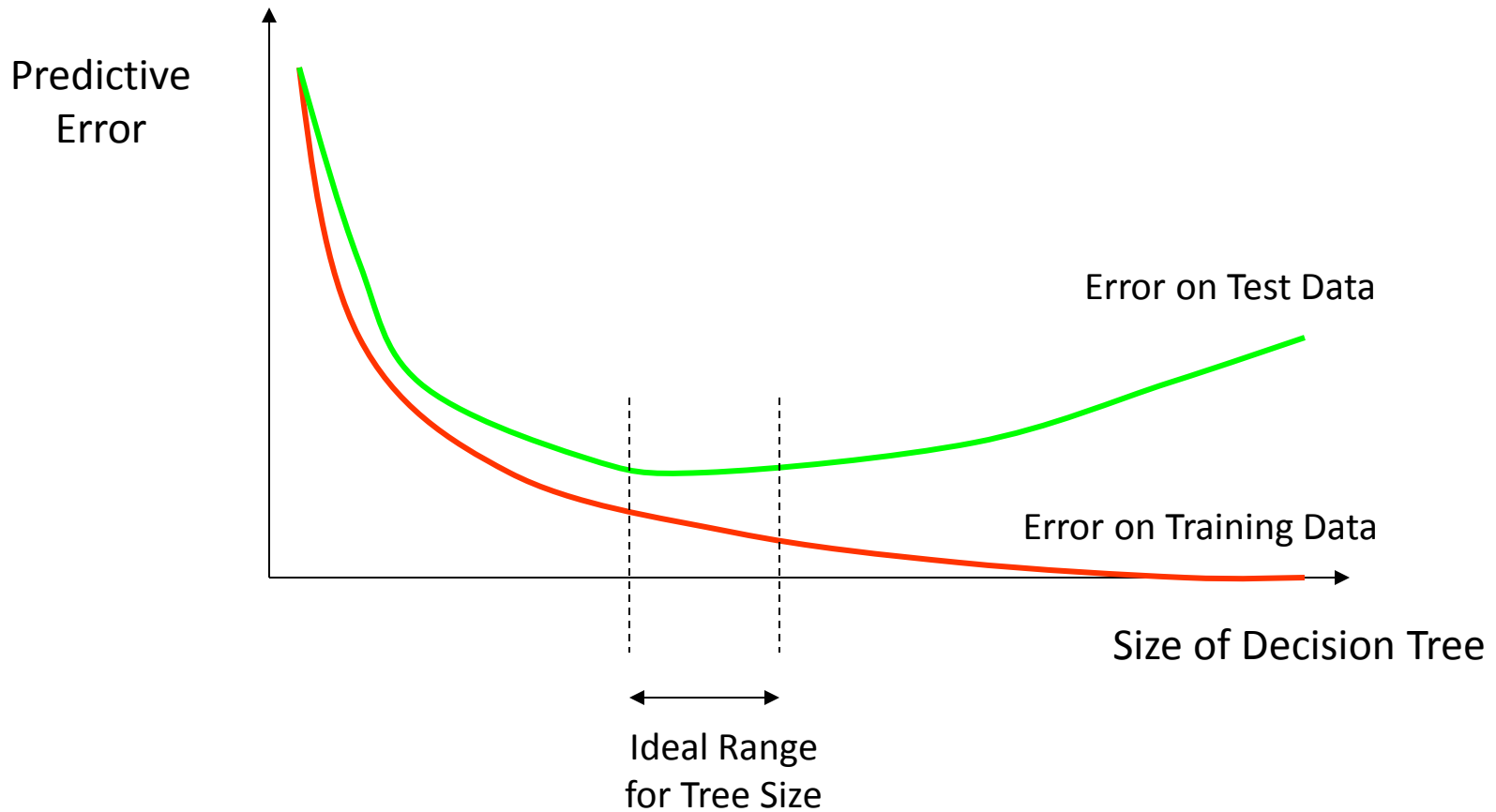
Computational Complexity for a Binary Tree

- At the root node, for each of p variables
 - Sort all values, compute quality for each split
 - $O(pN \log N)$ time for real-valued or ordinal variables
- Subsequent internal node operations each take $O(N' \log N')$
 - e.g., balanced tree of depth K requires
$$pN \log N + 2(pN/2 \log N/2) + 4(pN/4 \log N/4) + \dots + 2^K(pN/2^K \log N/2^K)$$
$$= pN(\log N + \log(N/2) + \log(N/4) + \dots + \log N/2^K)$$
- This assumes data are in main memory
 - If data are on disk then repeated access of subsets at different nodes may be very slow (impossible to pre-index)

Splitting on a nominal attribute

- Nominal attribute with m values
 - e.g., the name of a state or a city in marketing data
- 2^{m-1} possible subsets \Rightarrow exhaustive search is $O(2^{m-1})$
 - For small m , a simple approach is to branch on specific values
 - But for large m this may not work well
- Neat trick for the 2-class problem:
 - For each predictor value calculate the proportion of class 1's
 - Order the m values according to these proportions
 - Now treat as an ordinal variable and select the best split (linear in m)
 - This gives the optimal split for the Gini index, among all possible 2^{m-1} splits (Breiman et al, 1984).

How to Choose the Right-Sized Tree?



Choosing a Good Tree for Prediction

- General idea
 - grow a large tree
 - prune it back to create a family of subtrees
 - “weakest link” pruning
 - score the subtrees and pick the best one
- Massive data sizes (e.g., $n \sim 100k$ data points)
 - use training data set to fit a set of trees
 - use a validation data set to score the subtrees
- Smaller data sizes (e.g., $n \sim 1k$ or less)
 - use cross-validation
 - use explicit penalty terms (e.g., Bayesian methods)

Why Trees are widely used in Practice

- Can handle high dimensional data
 - builds a model using 1 dimension at time
- Can handle any type of input variables
 - categorical, real-valued, etc
 - most other methods require data of a single type (e.g., only real-valued)
- Trees are (somewhat) interpretable
 - domain expert can “read off” the tree’s logic
- Tree algorithms are relatively easy to code and test

Limitations of Trees

- Representational Bias
 - classification: piecewise linear boundaries, parallel to axes
 - regression: piecewise constant surfaces
- High Variance
 - trees can be “unstable” as a function of the sample
 - e.g., small change in the data -> completely different tree
 - causes two problems
 - 1. High variance contributes to prediction error
 - 2. High variance reduces interpretability
 - Trees are good candidates for model combining
 - Often used with boosting and bagging
- Trees do not scale well to massive data sets (e.g., N in millions)
 - repeated random access of subsets of the data

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Regression re-visited

*Inspired by notes of A. Ng (Stanford) on Machine Learning - CS 229

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό

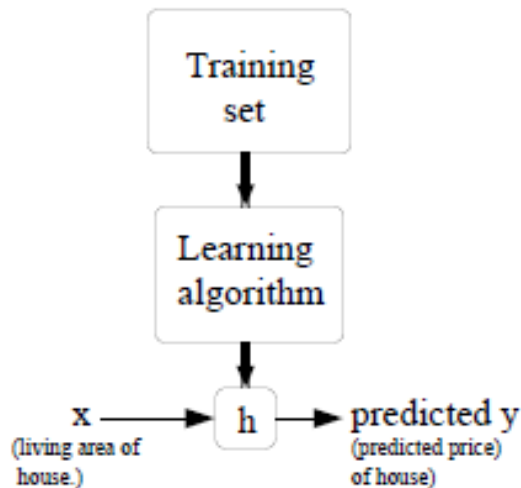
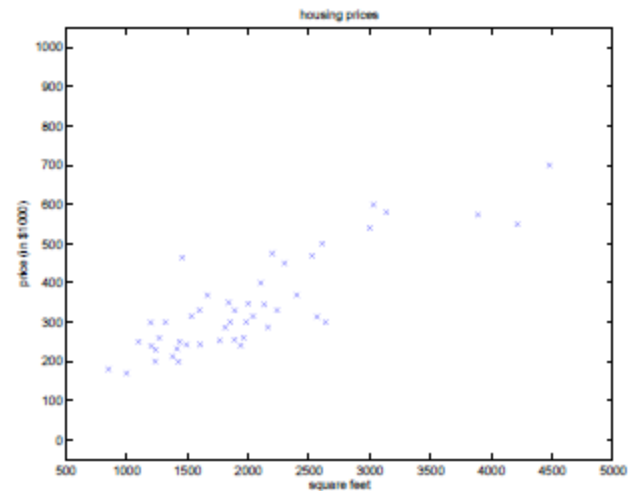
Ενότητα # 3: Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης

Τμήμα: Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”

Prediction – Classification..

Living area (feet ²)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540



Learning – linear regression

Living area (feet ²)	#bedrooms	Price (1000\$)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$H(\theta)$: hypothesis

Θ : parameters (weights)

Assume intercept $x_0=1$, hence

Define the cost function:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2.$$

$$h(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x,$$

LMS – gradient descent

- We want to choose θ to minimize $J(\theta)$.
- gradient descent algorithm, which starts with some initial θ , and repeatedly performs the update:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

- simultaneously performed for all values of $j = 0, \dots, n$.
- α : learning rate
- to implement algorithm: partial derivative term on the right hand side.

- for one training example (x, y) :
$$\frac{\partial}{\partial \theta_j} J(\theta) = (h_\theta(x) - y) x_j$$

- Thus update rule:
$$\theta_j := \theta_j + \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}.$$

- LMS update rule - also known as the *Widrow-Hoff learning rule*.

LMS rule properties

- $$\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}.$$
- gradient descent always converges (assuming the learning rate α is not too large) to the global minimum.
 - J is a convex quadratic function.
- the magnitude of the update is proportional to the error term
 - For a training example where prediction nearly matches the actual value of $y^{(i)}$, small change on the parameters;
 - if our prediction $h(x^{(i)})$ has a large error (i.e., if it is very far from $y^{(i)}$) a larger change to the parameters will be made

LMS – batch gradient descent

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^m (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)} \quad (\text{for every } j).$$

}

- * m number of training examples
- The method looks at every sample for each step
- Global optima – hence converges to local minimum
- The “ α ” value should not be too high

LMS – stochastic gradient descent

```
Loop {  
    for i=1 to m, {  
         $\theta_j := \theta_j + \alpha (y^{(i)} - h_{\theta}(x^{(i)})) x_j^{(i)}$     (for every  $j$ ).  
    }  
}
```

- * m number of training examples
- Parameters are updated for each training example
- Lower complexity
- Converges faster but may oscillate around values
- for large training sets , stochastic gradient descent is often preferred over batch gradient descent.

LMS revisited – normal equations

- *Directly* minimize J by derivation with respect to θ_j 's and setting them to 0
- Some notation:

For a function $f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$ mapping from m -by- n matrices to the real numbers, we define the derivative of f with respect to A to be:

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

Thus, the gradient $\nabla_A f(A)$ is itself an m -by- n matrix, whose (i, j) -element is $\partial f / \partial A_{ij}$.

Trace operator properties

- Trace operator: $\text{tr}A = \sum_{i=1}^n A_{ii}$

- Properties

$$\text{tr}AB = \text{tr}BA.$$

$$\text{tr}ABC = \text{tr}CAB = \text{tr}BCA,$$

$$\text{tr}ABCD = \text{tr}DABC = \text{tr}CDAB = \text{tr}BCDA.$$

- Let A, B square matrices and α a real number:

$$\text{tr}A = \text{tr}A^T$$

$$\nabla_A \text{tr}AB = B^T$$

$$\text{tr}(A + B) = \text{tr}A + \text{tr}B$$

$$\nabla_A \text{tr}f(A) = (\nabla_A f(A))^T$$

$$\text{tr}aA = a\text{tr}A$$

$$\nabla_A \text{tr}ABA^T C = CAB + C^T AB^T$$

$$\nabla_A |A| = |A|(A^{-1})^T.$$

LMS revisited..

- Aim to find the closed form of θ that minimizes $J(\theta)$
- Let X the training set of vectors and y the target values vector

$$X = \begin{bmatrix} \text{---} & (\mathbf{x}^{(1)})^T & \text{---} \\ \text{---} & (\mathbf{x}^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (\mathbf{x}^{(m)})^T & \text{---} \end{bmatrix} \cdot \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \cdot$$

LMS revisited.. – direct error minimization

- As $h_{\theta}(x^{(i)}) = (x^{(i)})^T \theta$, then $X\theta - \vec{y} = \begin{bmatrix} (x^{(1)})^T \theta \\ \vdots \\ (x^{(m)})^T \theta \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$
- Thus:

$$\frac{1}{2}(X\theta - \vec{y})^T (X\theta - \vec{y}) = \frac{1}{2} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 = J(\theta)$$

$$= \begin{bmatrix} h_{\theta}(x^{(1)}) - y^{(1)} \\ \vdots \\ h_{\theta}(x^{(m)}) - y^{(m)} \end{bmatrix}.$$

- To minimize J, derivatives with respect to θ (with 2,3):

$$\nabla_{A^T \text{tr} ABA^T C} = B^T A^T C^T + BA^T C$$

- set the derivatives to 0: $\nabla_{\theta} J(\theta) = X^T X \theta - X^T \vec{y}$
- normal equations: $X^T X \theta = X^T \vec{y}$
- Thus θ minimizing $J(\theta)$: $\theta = (X^T X)^{-1} X^T \vec{y}$.

Regression – probabilistic interpretation

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

- $\epsilon^{(i)}$ some error function iid normally distributed

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$

$$p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X; \theta).$$

$$L(\theta) = \prod_{i=1}^m p(y^{(i)} | x^{(i)}; \theta)$$

- Maximize the likelihood:
- Assuming independence:

$$= \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

Regression – probabilistic interpretation

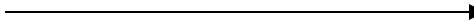
- Maximize the log likelihood

$$\ell(\theta) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2.$$

- Equivalent to minimize: $\frac{1}{2} \sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2$
- This is the $J(\theta)$.. Solved above..

Evaluating Classification Results (in general)

- Summary statistics:
 - empirical estimate of score function on test data, eg., error rate

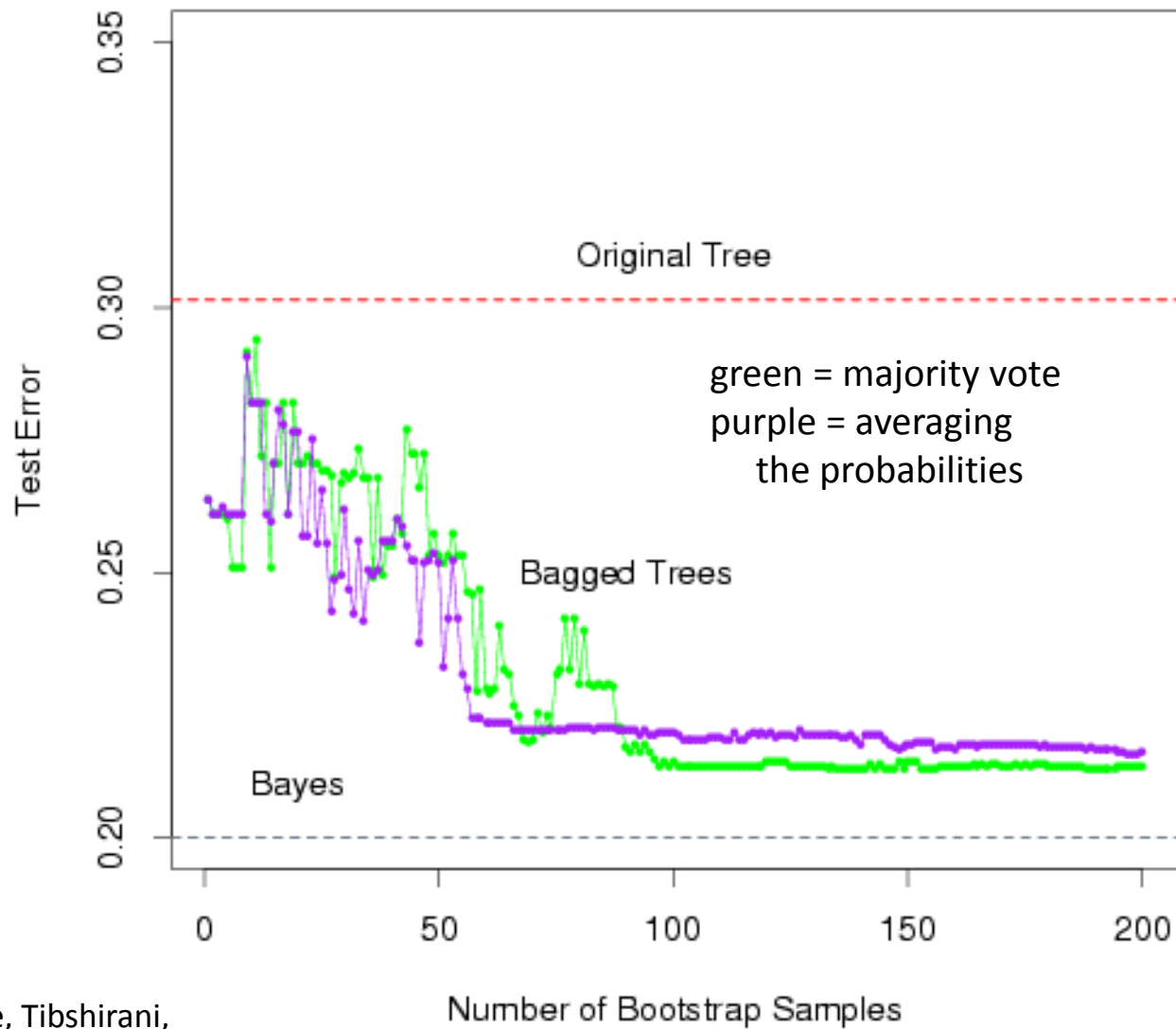
- More detailed breakdown 
 - E.g., “confusion matrices”
 - Can be quite useful in detecting systematic errors

	Predicted	
True	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%

- Detection v. false-alarm plots (2 classes)
 - Binary classifier with real-valued output for each example, where higher means more likely to be class 1
 - For each possible threshold, calculate
 - Detection rate = fraction of class 1 detected
 - False alarm rate = fraction of class 2 detected
 - Plot y (detection rate) versus x (false alarm rate)
 - Also known as ROC, precision-recall, specificity/sensitivity

Bagging for Combining Classifiers

- Training data sets of size N
- Generate B “bootstrap” sampled data sets of size N
 - Bootstrap sample = sample with replacement
 - e.g. $B = 100$
- Build B models (e.g., trees), one for each bootstrap sample
 - Intuition is that the bootstrapping “perturbs” the data enough to make the models more resistant to true variability
- For prediction, combine the predictions from the B models
 - E.g., for classification $p(c | x) =$ fraction of B models that predict c
 - Plus: generally improves accuracy on models such as trees
 - Negative: lose interpretability



From Hastie, Tibshirani,
And Friedman, 2001

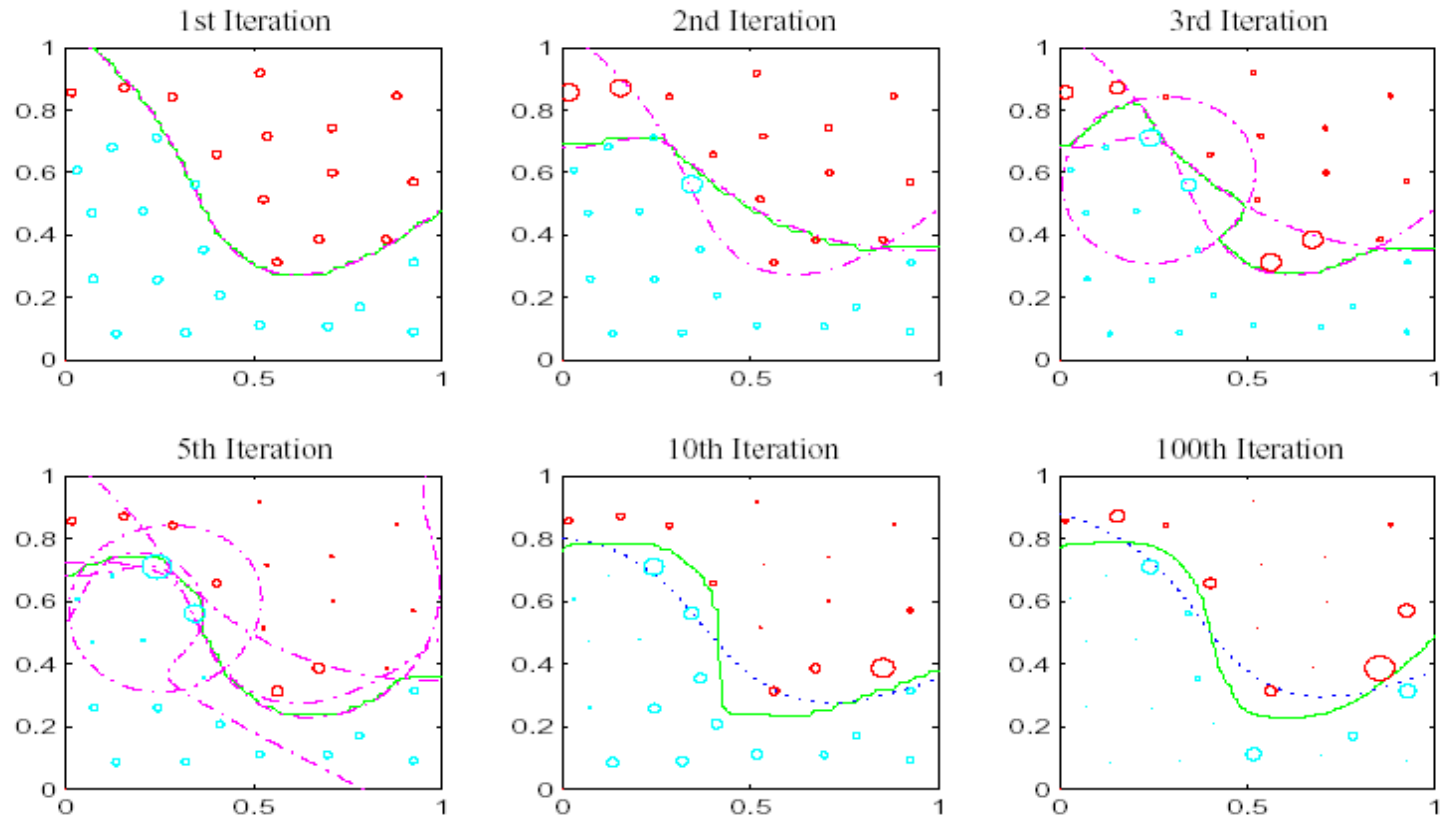


Illustration of Boosting:

Color of points = class label

Diameter of points = weight at each iteration

Dashed line: single stage classifier. Green line: combined, boosted classifier

Dotted blue in last two: bagging

(from G. Rätsch, Phd thesis, 2001)

References – Further material

- Regression & LSE

http://www.williams.edu/go/math/sjmillier/public_html/BrownClasses/54/mynotes52.htm

- Supervised learning

Andrew NG, Stanford Univ, CS229 - Machine Learning

(<http://www.stanford.edu/class/cs229/>)

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

Τέλος Ενότητας # 3

Μάθημα: Εξόρυξη γνώσης από Βάσεις Δεδομένων και τον Παγκόσμιο Ιστό, **Ενότητα # 3:** Supervised learning

Διδάσκων: Μιχάλης Βαζιργιάννης, **Τμήμα:** Προπτυχιακό Πρόγραμμα Σπουδών “Πληροφορικής”



Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο



ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

