## CS 1571 Introduction to AI
## Lecture 27

# Binary classification:
- **Naïve Bayes model**
- **Decision trees**

**Milos Hauskrecht**
milos@cs.pitt.edu
5329 Sennott Square



## Administration

- **Final exam:**
  - **December 12, 2014 at 4:00-5:50pm**
  - **In SENSQ 5129**
- **Exam is:**
  - Closed-book
  - Cumulative with more weight placed on the second part of the course
  - Similar in format to the midterm exam:
    - No programming questions
  - Please bring your calculators

## Supervised learning

**Data:** $D = \{D_1, D_2, .., D_n\}$ **a set of *n* examples**

$D_i = < \mathbf{x}_i, y_i >$

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \cdots x_{i,d})$ is an input vector of size *d*

$y_i$ is the desired output (given by a teacher)

**Objective:** learn the mapping $f : X \rightarrow Y$

s.t. $y_i \approx f(\mathbf{x}_i)$ for all $i = 1,.., n$

- **Regression:** Y is **continuous**
  Example: earnings, product orders $\rightarrow$ company stock price
- **Classification:** Y is **discrete**
  Example: handwritten digit in binary form $\rightarrow$ digit label



## Discriminative classification model

- **A classification model is defined using**
  - **discriminant functions**
- **Idea:**
  - For each class *i* define a function $g_i(\mathbf{x})$ mapping $X \rightarrow \Re$
  - When the decision on input **x** should be made choose the class with the highest value of $g_i(\mathbf{x})$

$$\text{class} = \arg\max_i \; g_i(x)$$

**Two models covered:**

- **Logistic regression** $g_1(\mathbf{x}) = g(\mathbf{w}^T\mathbf{x} + w_0)$

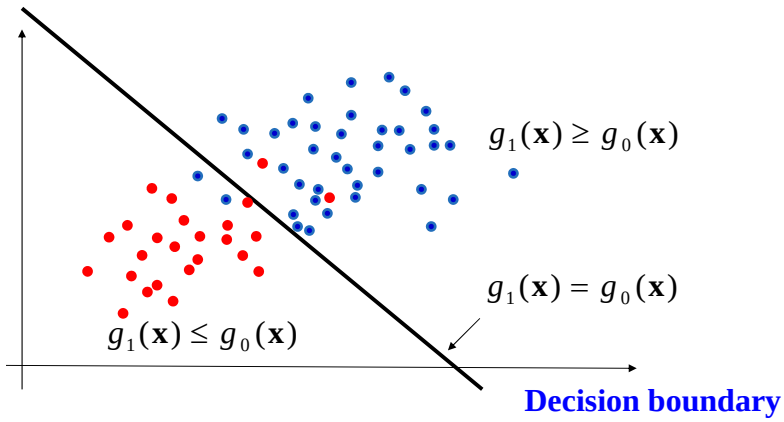
  $g_0(\mathbf{x}) = 1 - g(\mathbf{w}^T\mathbf{x} + w_0)$

  where $g(z) = 1/(1 + e^{-z})$
- **Support vector machines** $g_1(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$

  $g_0(\mathbf{x}) = -(\mathbf{w}^T\mathbf{x} + w_0)$

## Discriminant functions: review

- Assume a binary classification problem with classes 0 and 1
- Discriminant functions $g_0(\mathbf{x})$ and $g_1(\mathbf{x})$



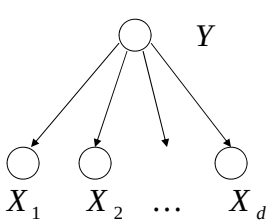



## Naïve Bayes classifier

- **A generative classification model:**
- **models and learns the joint distribution of x and** y

$$p(\mathbf{x}, y) = p(y)\,p(\mathbf{x} \mid y)$$

- **Calculates posterior probability:** $p(y \mid \mathbf{x}) = \dfrac{p(y)\,p(\mathbf{x} \mid y)}{p(\mathbf{x})}$
- **Naïve Bayes model**
  - **simplifying assumption:** All input attributes are conditionally independent of each other given the class.




$$p(\mathbf{x}, y) = p(y)\,p(\mathbf{x} \mid y)$$
$$= p(y)\prod_{i=1}^{d} p(x_i \mid y)$$

## Making class decision for the Naïve Bayes

**Discriminant functions based on posterior of a class**

$g_1(\mathbf{x}) = p(y=1 \mid \mathbf{x})$ $\qquad$ $g_0(\mathbf{x}) = p(y=0 \mid \mathbf{x}) = (1 - g_1(\mathbf{x}))$

–the model chooses the class with better posterior probability

$$g_1(x) = p(y=1 \mid \mathbf{x}) = \frac{\left(\prod_{i=1}^{d} p(x_i \mid y=1)\right) p(y=1)}{\left(\prod_{i=1}^{d} p(x_i \mid y=0)\right) p(y=0) + \left(\prod_{i=1}^{d} p(x_i \mid y=1)\right) p(y=1)}$$

**Alternative discriminant function based on likelihood of data:**

– chooses the class that explains the input data (**x**) better

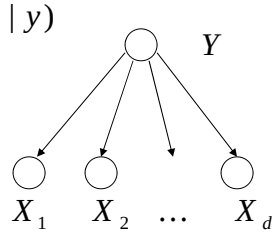
$$\underbrace{\prod_{i=1}^{d} p(x_i \mid y=1)}_{g_1(\mathbf{x})} > \underbrace{\prod_{i=1}^{d} p(x_i \mid y=0)}_{g_0(\mathbf{x})} \Rightarrow \begin{array}{l} \text{then } y=1 \\ \text{else } y=0 \end{array}$$



## Learning of the Naïve Bayes classifier

**Model:**

$$p(\mathbf{x}, y) = p(y)\, p(\mathbf{x} \mid y) = p(y) \prod_{i=1}^{d} p(x_i \mid y)$$




**Learning**

**= learning of parameters of the BBN:**

– **Class prior**

$p(y)$

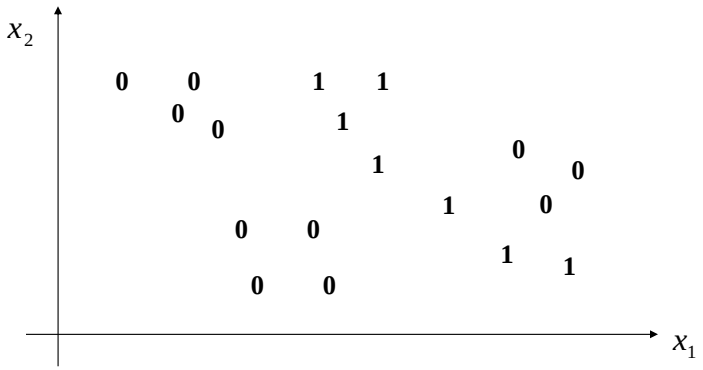
– **class conditional distributions**:

$p(x_i \mid y) \quad for\ all\ \ i = 1,..d$

- In practice class conditional distribution can have different models: e.g. one attribute can be modeled using the Bernoulli, the other as Gaussian density, or as a Poisson distribution

## Decision trees

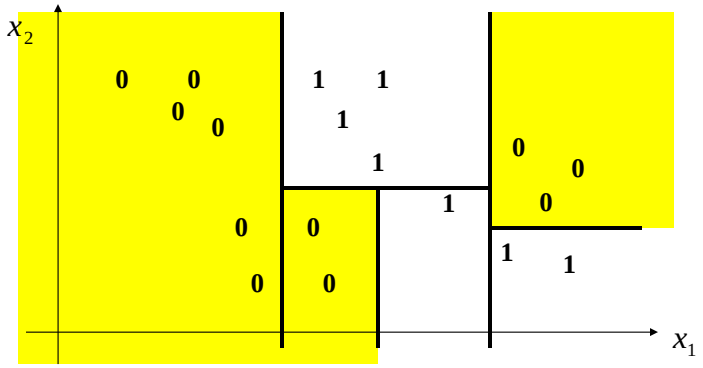
- An alternative approach to classification:
  - **Partition the input space to regions**
  - **Regress or classify independently in every region**






## Decision trees

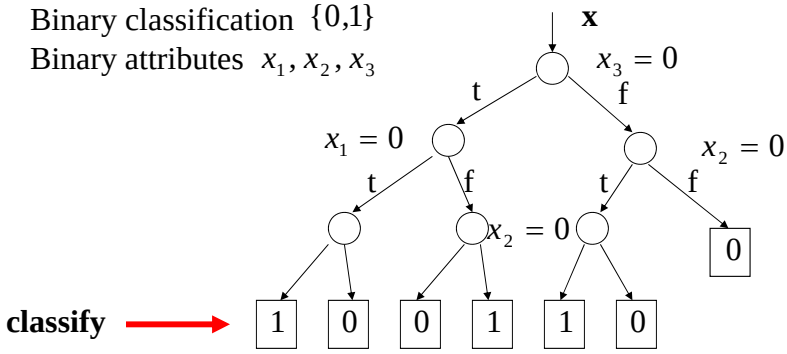
- An alternative approach to classification:
  - **Partition the input space to regions**
  - **Regress or classify independently in every region**



# Decision trees

- **Decision tree model**:
  - Splits the space recursively according to inputs in **x**
  - Classify at the bottom of the tree

**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$

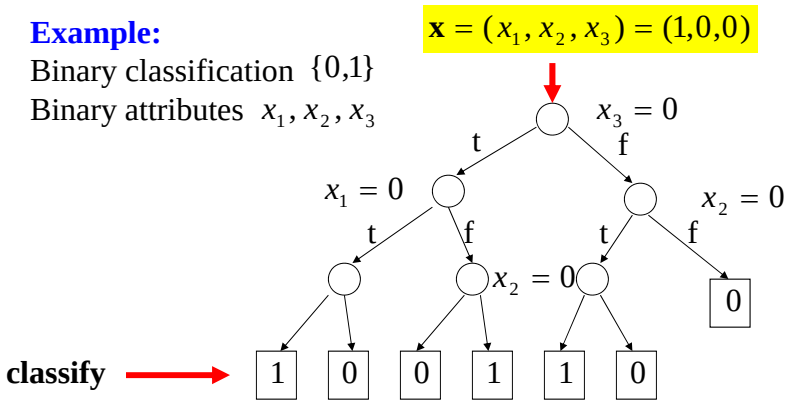





# Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in **x**
  - Classify at the bottom of the tree

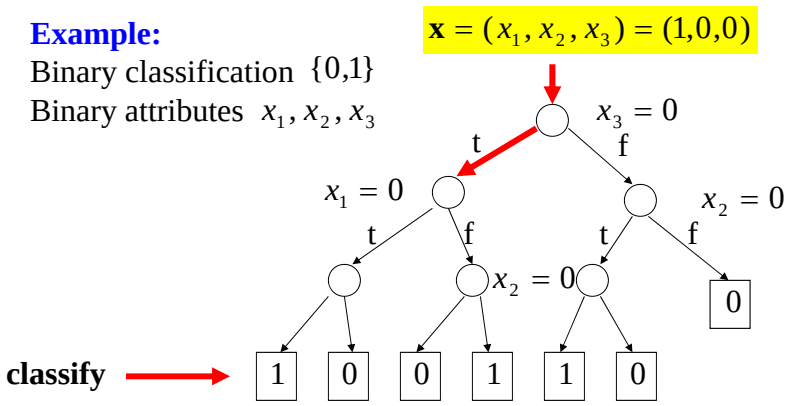
**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$



# Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in **x**
  - Classify at the bottom of the tree

**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$

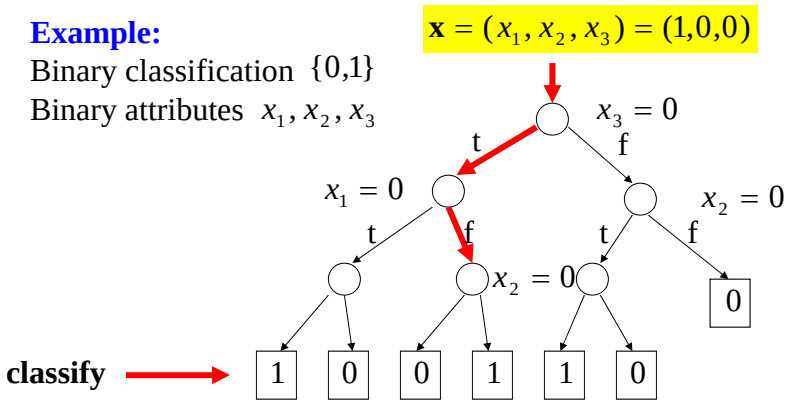





# Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in **x**
  - Classify at the bottom of the tree

**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$

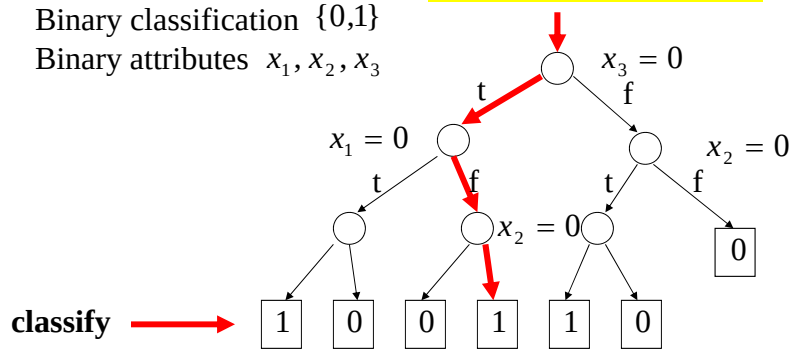


# Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in **x**
  - Classify at the bottom of the tree

**Example:**
Binary classification $\{0,1\}$
Binary attributes $x_1, x_2, x_3$

$\mathbf{x} = (x_1, x_2, x_3) = (1,0,0)$

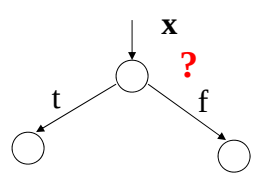





# Learning decision trees

**How to construct /learn the decision tree?**

- **Top-bottom algorithm**:
  - Find the best split condition (quantified based on the impurity measure)
  - Stops when no improvement possible




- **Impurity measure I**:
  - measures how well are the two classes in the training data D separated …. I(D)
  - Ideally we would like to separate all 0s and 1
- Splits: **finite or continuous value attributes**

Continuous value attributes conditions: $x_3 \leq 0.5$

## Impurity measure

Let $|D|$ - Total number of data entries in the training dataset

$|D_i|$ - Number of data entries classified as class $i$

$p_i = \frac{|D_i|}{|D|}$ - ratio of instances classified as class $i$

**Impurity measure** I(D)

- defines how well the classes are separated
- in general the impurity measure should satisfy:
  - Largest when data are split evenly for attribute values

$$p_i = \frac{1}{\text{number of classes}}$$

  - Should be 0 when all data belong to the same class

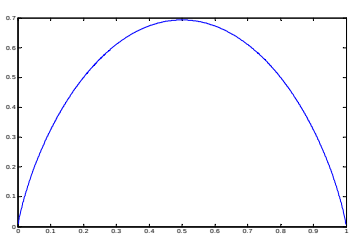


## Impurity measures

- There are various impurity measures used in the literature
  - **Entropy based measure (Quinlan, C4.5)**

$$I(D) = Entropy\ (D) = -\sum_{i=1}^{k} p_i \log\ p_i$$

Example for k=2



  - **Gini measure (Breiman, CART)**

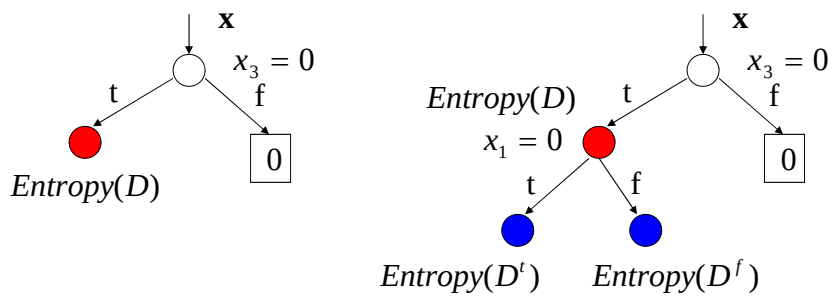
$$I(D) = Gini\ (D) = 1 - \sum_{i=1}^{k} p_i^{\ 2}$$

## Impurity measures

- **Gain due to split** – expected reduction in the impurity measure (entropy example)

$$Gain\,(D, A) = Entropy\;(D) - \sum_{v \in Values\;(A)} \frac{|D^v|}{|D|} Entropy\;(D^v)$$

$|D^v|$ - a partition of $D$ with the value of attribute A = $v$






## Decision tree learning

- **Greedy learning algorithm:**

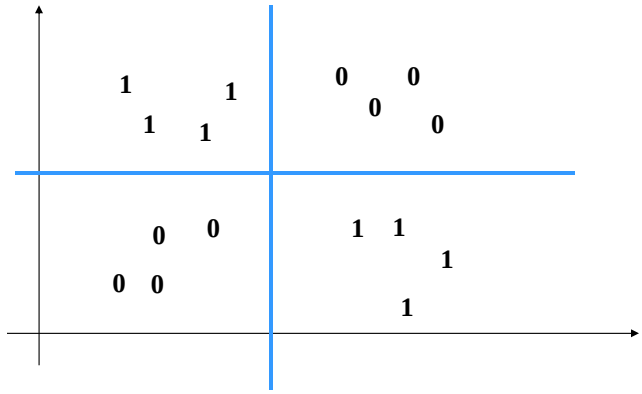
  Repeat until no or small improvement in the purity
  - Find the attribute with the highest gain
  - Add the attribute to the tree and split the set accordingly

- Builds the tree in the top-down fashion
  - Gradually expands the leaves of the partially built tree
- The method is greedy
  - It looks at a single attribute and gain in each step
  - May fail when the combination of attributes is needed to improve the purity (parity functions)

## Decision tree learning

- **Limitations of greedy methods**

  Cases in which a combination of two or more attributes improves the impurity






## Decision tree learning

By reducing the impurity measure we can grow **very large trees**

**Problem: Overfitting**

- We may split and classify very well the training set, but we may do worse in terms of the generalization error

**Solutions to the overfitting problem**:

- **Solution 1.**
  - Prune branches of the tree built in the first phase
  - Use validation set to test for the overfit
- **Solution 2**.
  - Test for the overfit in the tree building phase
  - Stop building the tree when performance on the validation set deteriorates