CS 2750 Machine Learning
Lecture 12

Multi-class classification
Decision trees

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

Midterm exam

Midterm Monday, March 2, 2015

- **In-class (75 minutes)**
- **closed book**
- **material covered by February 26, 2015**

Project proposals

Due: **Monday, March 16, 2014**

- **1 page long**

Proposal

- **Written proposal:**
  1. Outline of a learning problem, type of data you have available. Why is the problem important?
  2. Learning methods you plan to try and implement for the problem. References to previous work.
  3. How do you plan to test, compare learning approaches
  4. Schedule of work (approximate timeline of work)

Project proposals

Where to find the data:

- From your research (computer, web page logs, processor workloads, network traffic, biomedical data)
- Collect them on your own (link analysis, web data)
- UC Irvine data repository
- Various other sources:
  - text document repositories
  - bioinformatics data can be found on the NIH or various university web sites (e.g. microarray data, proteomic data)
- Synthetic data that are generated to demonstrate your algorithm works

Project proposals

Problems to address:

- Important problem in your area of research, where ML or adaptation to the environment is promising and expected to help
- Get the ideas for the project by browsing the web
  - Various competitions and data used by ML and KDD communities

Problem complexity:

- It is tempting to go with one UCI data set and basic classification algorithms. Not sufficient. Try :
  - Your own data collection
  - Multiple methods, not just one method
  - More advanced methods, e.g. ensemble methods

Project proposals

Interesting problems to consider:

- Multi-class learning problems
- Multi-task learning: predict multiple labels
- Methods for learning the parameters and structure of Bayesian Belief networks
- Dimensionality reduction/feature selection
- Low-dimensional representation of data
- Learning how to act – Reinforcement learning
- Anomaly detection – how to identify outliers in data

Multiclass classification

- **Binary classification** $Y = \{0,1\}$
  - Learn: $f : X \rightarrow \{0,1\}$
- **Multiclass classification**
  - **K classes** $Y = \{0,1,\ldots,K-1\}$
  - **Goal:** learn to classify correctly K classes
  - Or **learn**

$$f : X \rightarrow \{0,1,\ldots,K-1\}$$

Multiclass classification

Approaches:

- **Generative model approach**
  - Generative model of the distribution p(**x**,y)
  - Learns the parameters of the model through density estimation techniques
  - Discriminant functions are based on the model
    - "Indirect" learning of a classifier
- **Discriminative approach**
  - Parametric discriminant functions
  - Learns discriminant functions **directly**
    - A logistic regression model.

Generative model approach

Indirect:

1. **Represent and learn the distribution** $p(\mathbf{x}, y)$
2. **Define and use probabilistic discriminant functions**

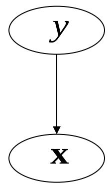
$$g_i(\mathbf{x}) = \log p(y = i \mid \mathbf{x})$$

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

- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  k class-conditional distributions

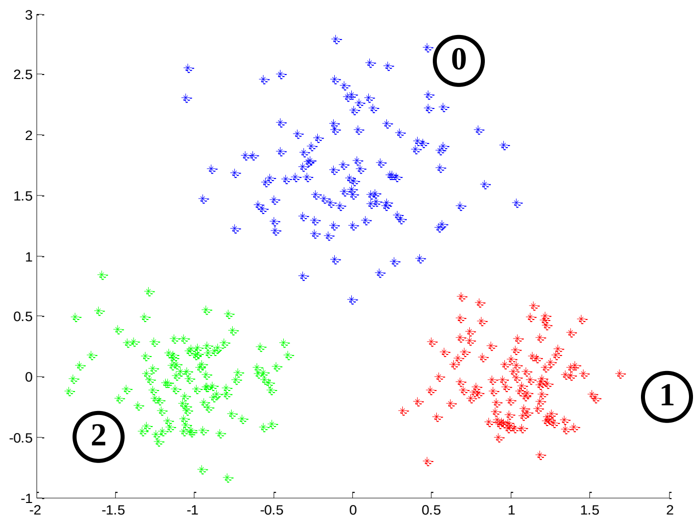
$$p(\mathbf{x} \mid y = i) \qquad \forall i \quad 0 \le i \le K-1$$



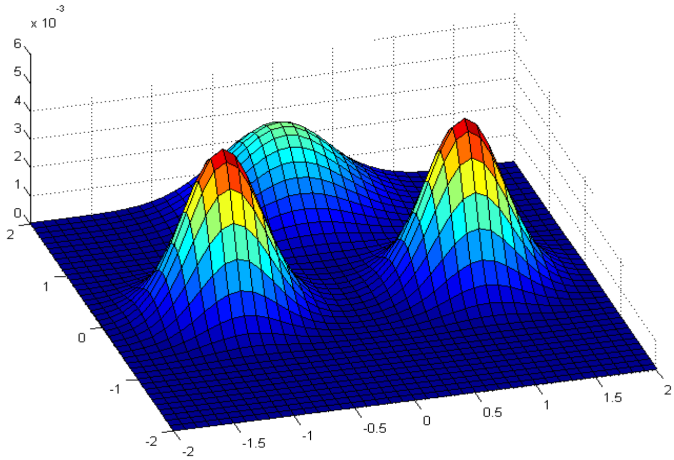
- $p(y)$ = **Priors on classes**
- **-** probability of class y

$$\sum_{i=1}^{K-1} p(y = i) = 1$$

Multi-way classification. Example



Multiclass classification



Making class decision

Discriminant functions can be based on:

- **Posterior of a class** – choose the class with higher posterior probability

  Choice: $i = \arg\max_{i=0,\ldots k-1} p(y = i \mid \mathbf{x}, \boldsymbol{\theta}_i)$

  $$p(y = i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \Theta_i) p(y = i)}{\sum_{j=0}^{k-1} p(\mathbf{x} \mid \Theta_j) p(y = j)}$$

- **Likelihood of data** – choose the class (Gaussian) that explains the input data (**x**) better (likelihood of the data)

  Choice: $i = \arg\max_{i=0,\ldots k-1} p(\mathbf{x} \mid \boldsymbol{\theta}_i)$

  $p(\mathbf{x} \mid \boldsymbol{\theta}_i) \approx p(\mathbf{x} \mid \mu_i, \boldsymbol{\Sigma}_i)$ For Gaussians

Discriminative approach

- **Parametric model** of discriminant functions:
  - $g_0(x), g_1(x), .. g_{K-1}(x)$
- Learn the discriminant functions directly

Key issues:

- How to design the discriminant functions?
- How to train them?

Another question:

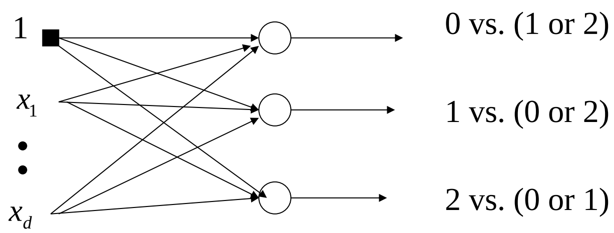
- Can we use binary classifiers to build the multi-class models?

One versus the rest (OvR)

Methods based on binary classification methods

- **Assume:** we have 3 classes labeled 0,1,2
- **Approach 1:**

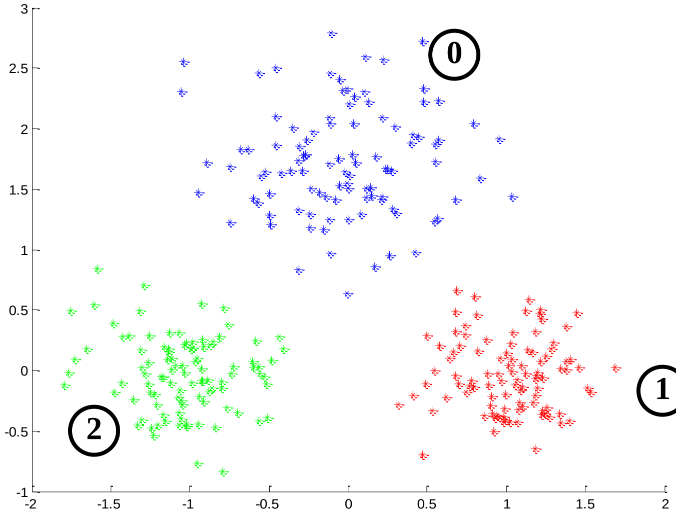
  A binary logistic regression on every class versus the rest (OvR)



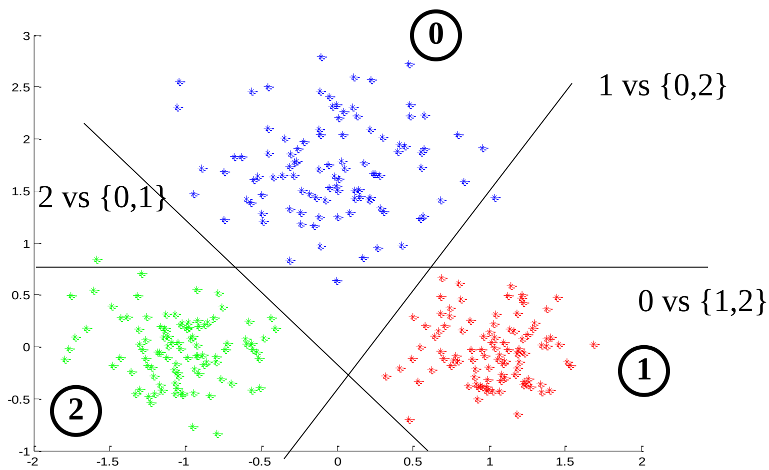
Class decision: class label for a 'singleton' class

– Does not work all the time

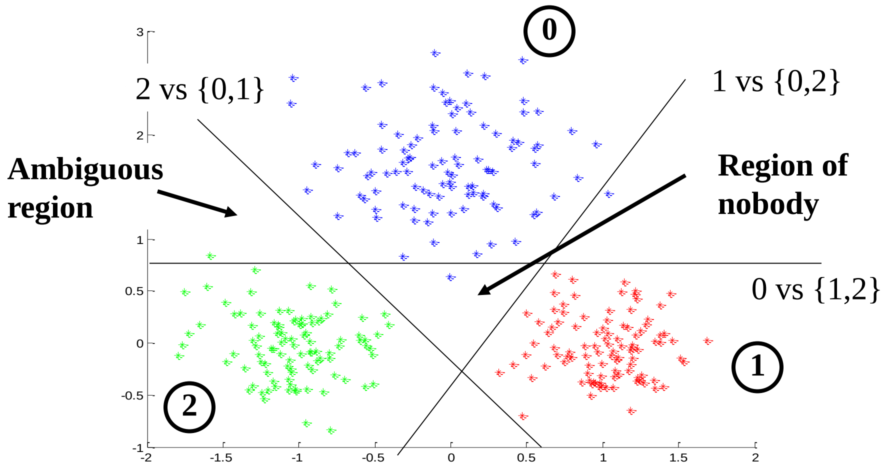
Multiclass classification. Example



Multiclass classification. Approach 1.



Multiclass classification. Approach 1.



One versus the rest (OvR)

Unclear how to decide on class in some regions

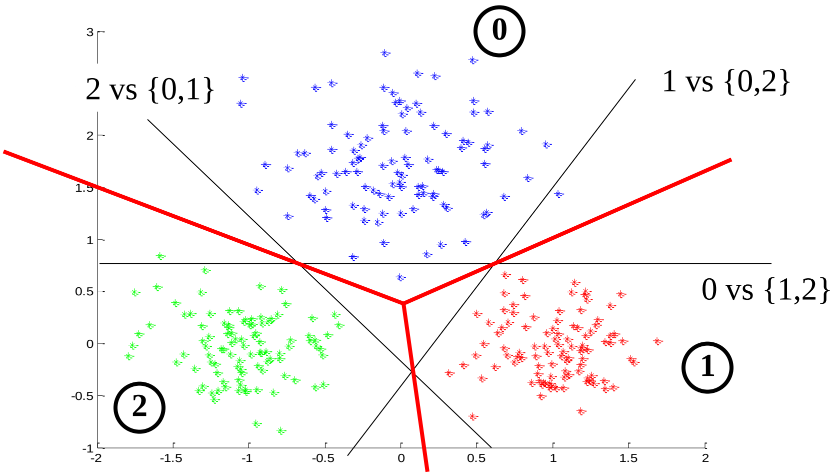
- **Ambigous region:**
  - 0 vs. (1 or 2) classifier says 0
  - 1 vs. (0 or 2) classifier says 1
- **Region of nobody:**
  - 0 vs. (1 or 2) classifier says (1 or 2)
  - 1 vs. (0 or 2) classifier says (0 or 2)
  - 2 vs (1 or 2) classifier says (1 or 2)

- **One solution:** compare discriminant functions defined on binary classifiers for single option:

  $g_i(\mathbf{x}) = g_{i \text{ vs rest}}(\mathbf{w}^T\mathbf{x})$

  – discriminant function for i trained on i vs. rest

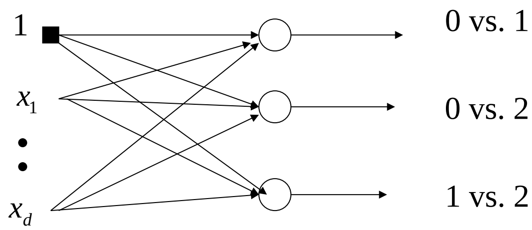
Multiclass classification. Approach 1.



Discriminative approach

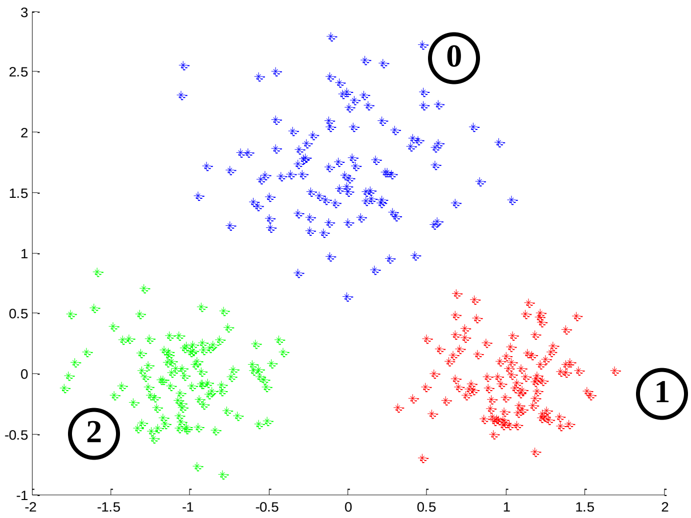
Methods based on binary classification methods

- **Assume:** we have 3 classes labeled 0,1,2
- **Approach 2:**
  - A binary logistic regression on all pairs

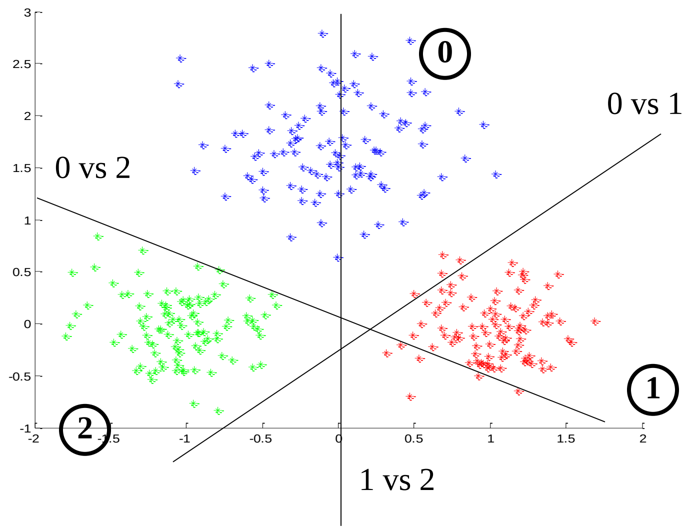


Class decision: class label based on who gets the majority
- Does not work all the time

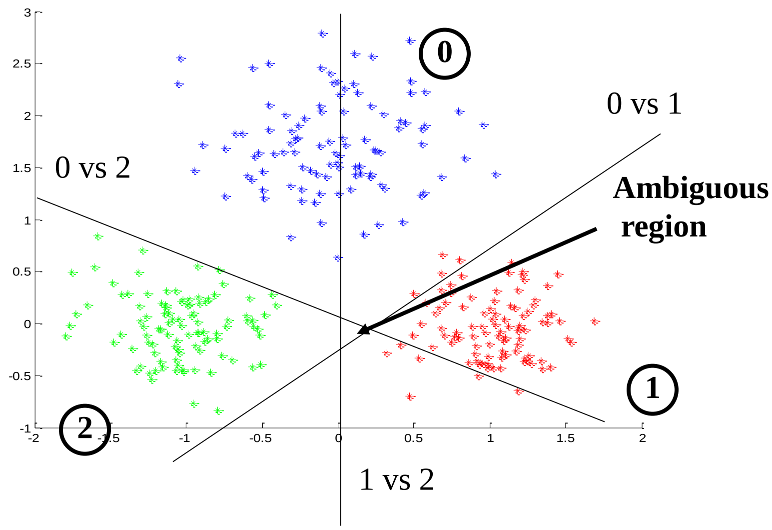
Multiclass classification. Example



Multiclass classification. Approach 2



Multiclass classification. Approach 2



One vs one model

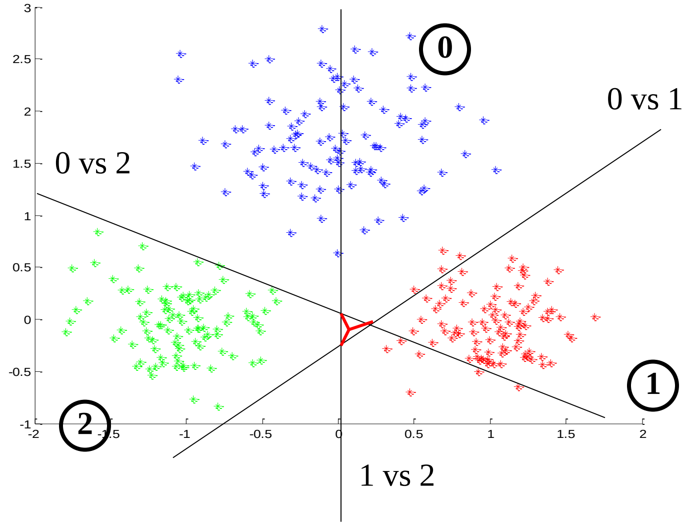
Unclear how to decide on class in some regions

- **Ambigous region:**
  - 0 vs. 1 classifier says 0
  - 1 vs. 2 classifier says 1
  - 2 vs. 0 classifier says 2

- **One solution:** define a new discriminant function by adding the corresponding discriminant functions for pairwise classifiers

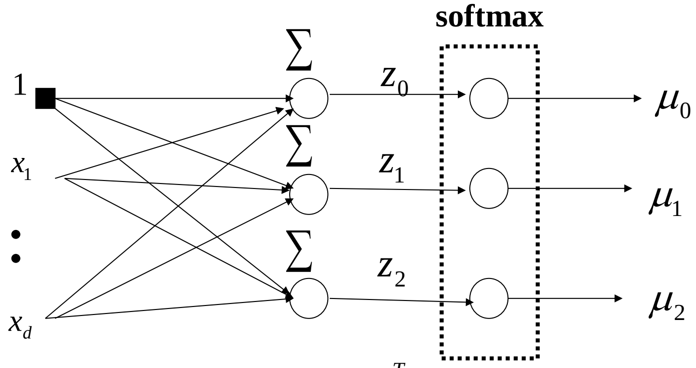
  $g_i(\mathbf{x}) = \text{sum\_j} \, (g_{i \text{ vs } j} (\mathbf{w}^T\mathbf{x}))$

Multiclass classification. Approach 2



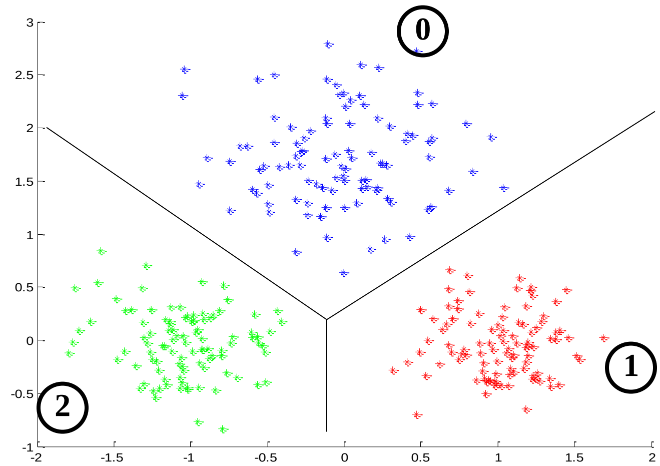
Multiclass classification with softmax

- A solution to the problem of having an ambiguous region:
  - tie discriminant functions together



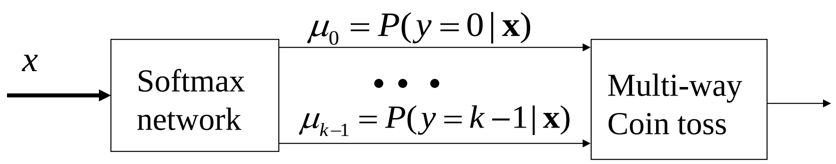
$$p(y = i \mid \mathbf{x}) = \mu_i = \frac{\exp(\mathbf{w}_i^T \mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})} \qquad \sum_i \mu_i = 1$$

Multiclass classification with softmax



Learning of the softmax model

- Learning of parameters **w**: statistical view



Assume outputs y are transformed as follows

$$y \in \{0 \quad 1 \quad .. \quad k-1\} \quad \Rightarrow \quad y \in \left\{ \begin{pmatrix} 1 \\ 0 \\ .. \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ .. \\ 0 \end{pmatrix} \; .. \; \begin{pmatrix} 0 \\ 0 \\ .. \\ 1 \end{pmatrix} \right\}$$

Learning of the softmax model

- Learning of the parameters **w**: statistical view
- **Likelihood of outputs**

$$L(D,\mathbf{w}) = p(\mathbf{Y} \mid \mathbf{X}, w) = \prod_{i=1,..n} p(y_i \mid \mathbf{x}_i, \mathbf{w})$$

- We want parameters **w** that maximize the likelihood
- **Log-likelihood trick**
  - Optimize log-likelihood of outputs instead:

$$l(D,\mathbf{w}) = \log \prod_{i=1,..n} p(y_i \mid \mathbf{x}, \mathbf{w}) = \sum_{i=1,..n} \log p(y_i \mid \mathbf{x}, \mathbf{w})$$

$$= \sum_{i=1,..n} \sum_{q=0}^{k-1} \log \mu_i^{y_{i,q}} = \sum_{i=1,..n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$$

- **Objective to optimize** $J(D_i, \mathbf{w}) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$

Learning of the softmax model

- **Error to optimize:**

$$J(D_i, \mathbf{w}) = -\sum_{i=1}^{n} \sum_{q=0}^{k-1} y_{i,q} \log \mu_{i,q}$$

- **Gradient**

$$\frac{\partial}{\partial w_{jq}} J(D_i, \mathbf{w}) = \sum_{i=1}^{n} -x_{i,j}(y_{i,q} - \mu_{i,q})$$

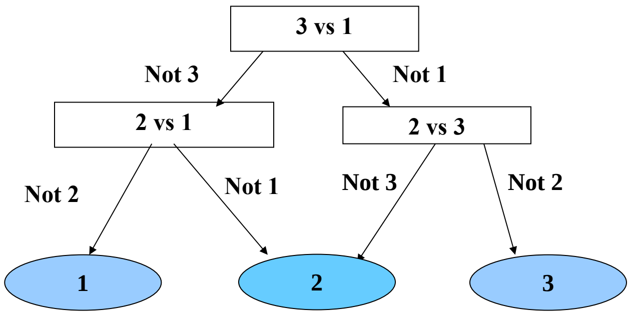
- The same very easy **gradient update** as used for the binary logistic regression

$$\mathbf{w}_q \leftarrow \mathbf{w}_q + \alpha \sum_{i=1}^{n} (y_{i,q} - \mu_{i,q}) \mathbf{x}_i$$

- But now we have to update the weights of k networks

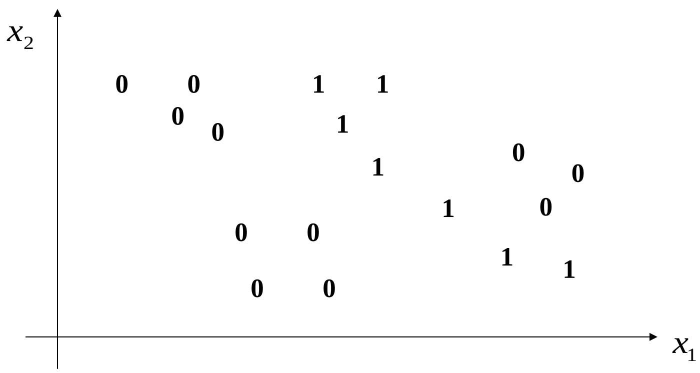
Multi-way classification

- **Yet another approach 3**



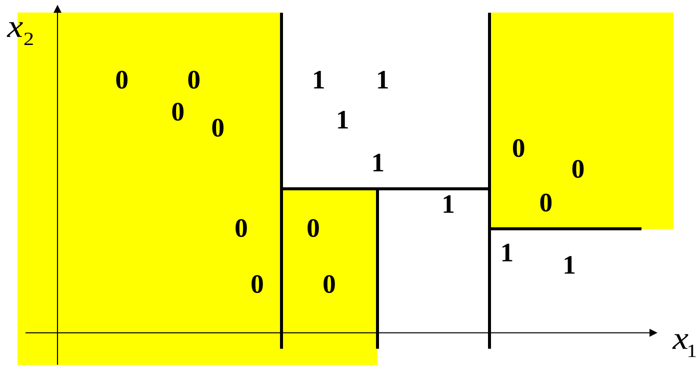
Decision trees

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



Decision trees

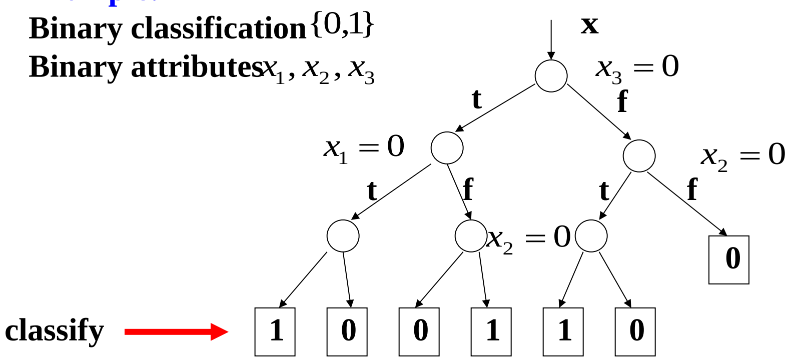
- An alternative approach to classification:
  - **Partition the input space to regions**
  - **Regress or classify independently in every region**



Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in $\mathbf{x}$
  - Classify at the bottom of the tree

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

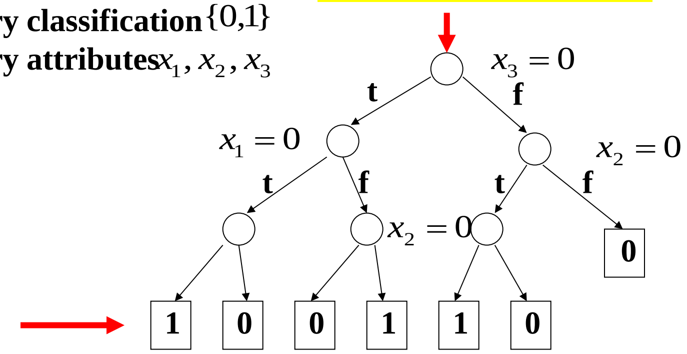


Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in $\mathbf{x}$
  - Classify at the bottom of the tree

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

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



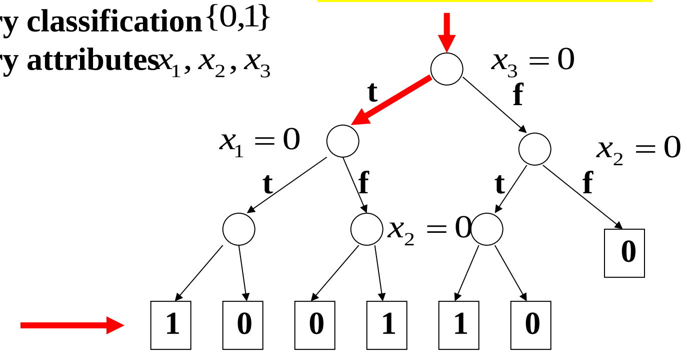
classify

Decision trees

- **Decision tree model**:
  - Split the space recursively according to inputs in $\mathbf{x}$
  - Classify at the bottom of the tree

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

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



classify

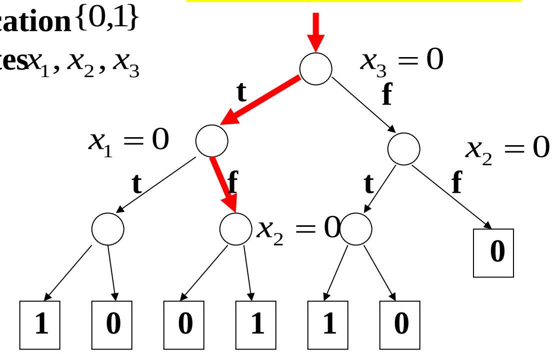
Decision trees

- **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

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

classify



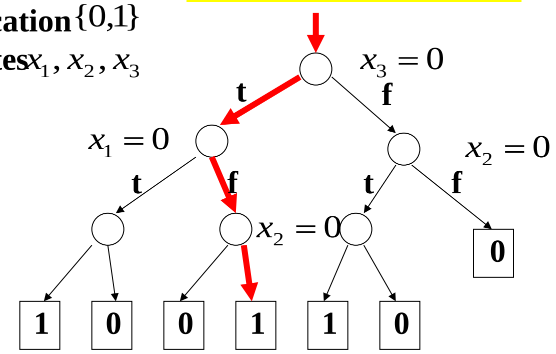
Decision trees

- **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

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

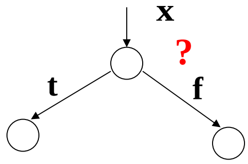
classify



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 *i***

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

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

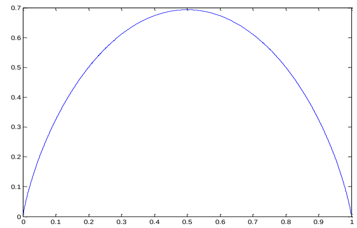
  – Should be 0 when all data belong to the same class

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)**

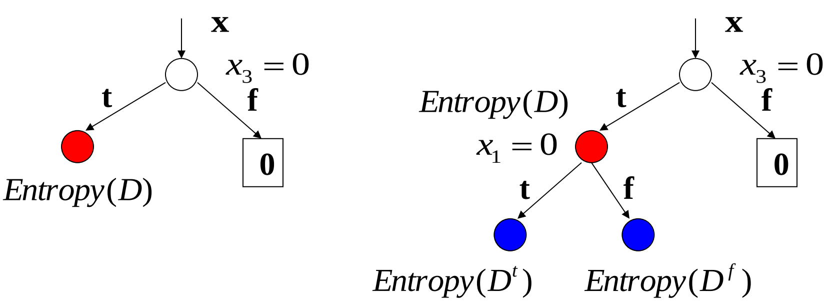
$$I(D) = Gini(D) = 1 - \sum_{i=1}^{k} p_i^{\,2}$$

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**:

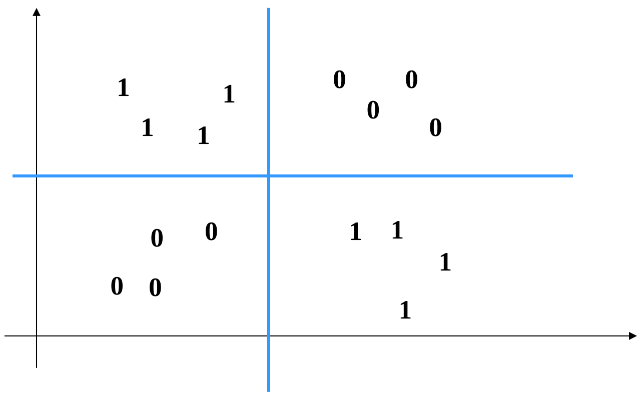
  Repeat until no or small improvement in the purity
  - Find the attribute with the highest gain
  - Add the attribute to the tree and split the set accordingly

- 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