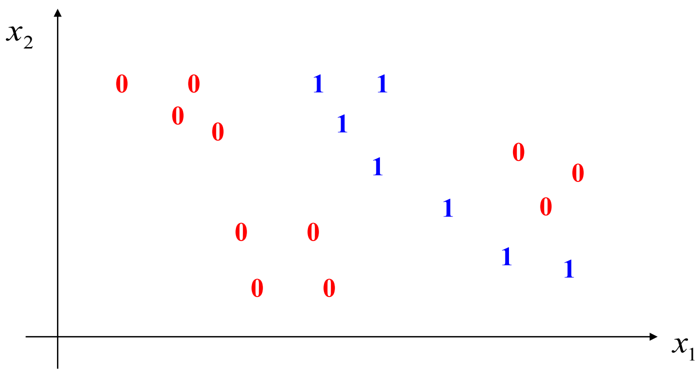
CS 1675 Introduction to Machine Learning
Lecture 14

Decision trees

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

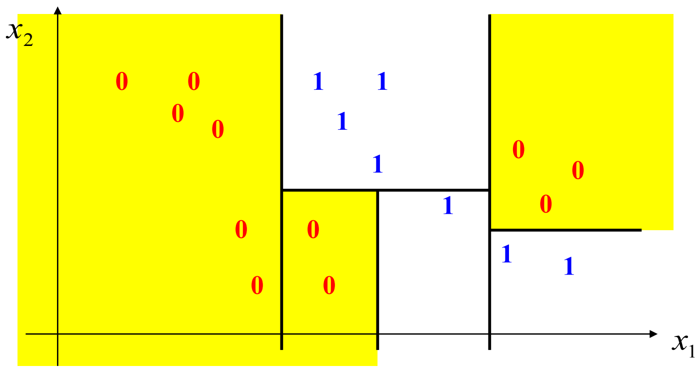
Decision tree classification

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



Decision tree classification

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



Decision tree classification

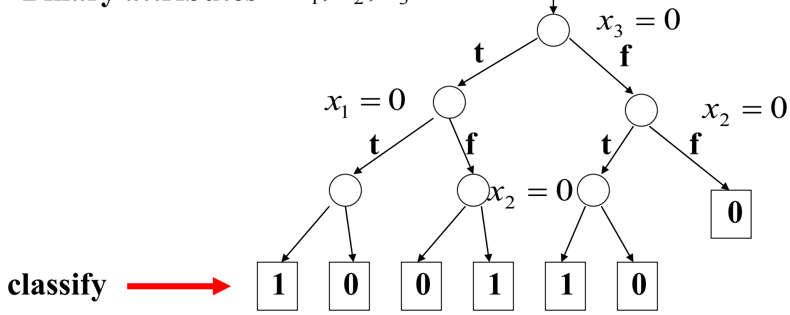
Decision tree model:

- Split recursively the input space **x** using simple conditions on x_i
- Classify at the bottom of the tree

Example:

Binary classification $\{0,1\}$

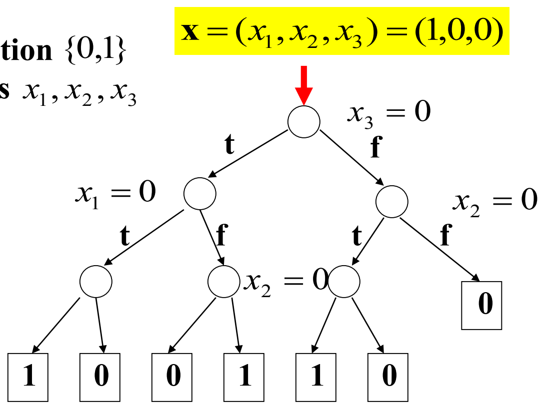
Binary attributes x_1, x_2, x_3



Decision trees

Decision tree model:
- Split recursively the input space **x** using simple conditions on x_i
- Classify at the bottom of the tree

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

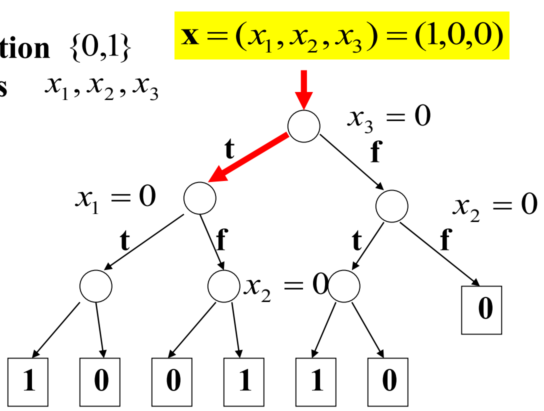


classify

Decision trees

Decision tree model:
- Split recursively the input space **x** using simple conditions on x_i
- Classify at the bottom of the tree

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



classify

Decision trees

Decision tree model:

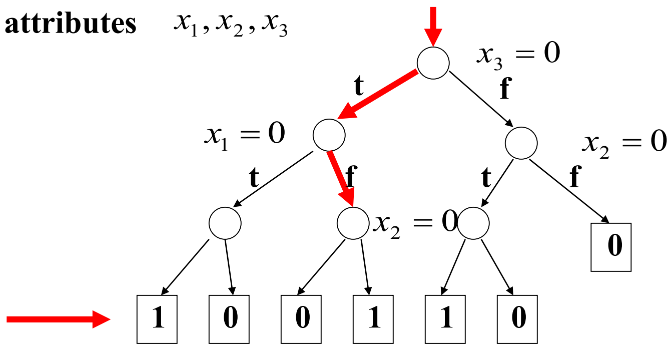
- Split recursively the input space **x** using simple conditions on x_i
- Classify at the bottom of the tree

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:

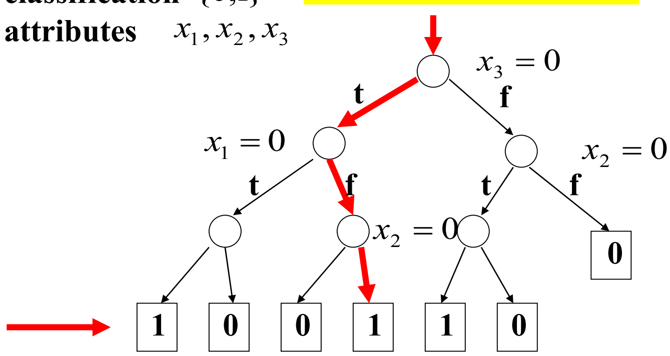
- Split recursively the input space **x** using simple conditions on x_i
- Classify at the bottom of the tree

Example:

Binary classification $\{0,1\}$

Binary attributes x_1, x_2, x_3

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

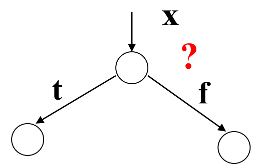


classify

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*(D)**:
  - measures the degree of mixing of the two classes in the subset of the training data D
  - Worst (maximum impurity) when # of 0s and 1s is the same
- Splits: **finite or continuous value attributes**

Continuous value attributes conditions: $x_3 \leq 0.5$

Impurity measure

Let $|D|$ **- Total number of data instances in D**

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

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

Impurity measure *I*(D)

- Measures the degree of mixing of the two classes in D
- The impurity measure should satisfy:
  - Largest when data are split evenly for attribute values

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

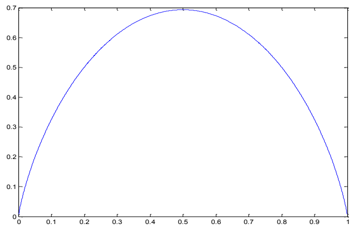
  - Should be 0 when all data belong to the same class

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)**

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

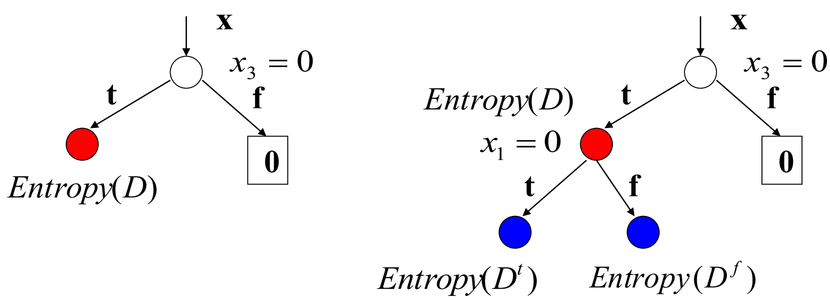
Impurity measures

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

Split condition

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

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



Decision tree learning

- **Greedy learning algorithm:**
  - Builds the tree in the top-down fashion
  - Gradually expands the leaves of the partially built tree

Algorithm sketch:

Repeat until no or small improvement in the impurity

- Find the attribute with the highest gain
- Add the attribute to the tree and split the set accordingly

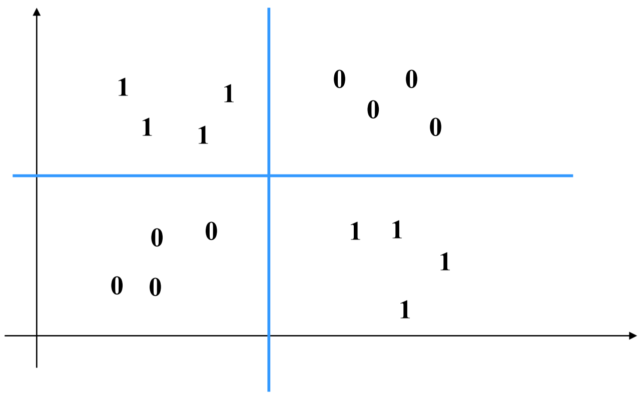
The method is greedy:

- 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 only 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

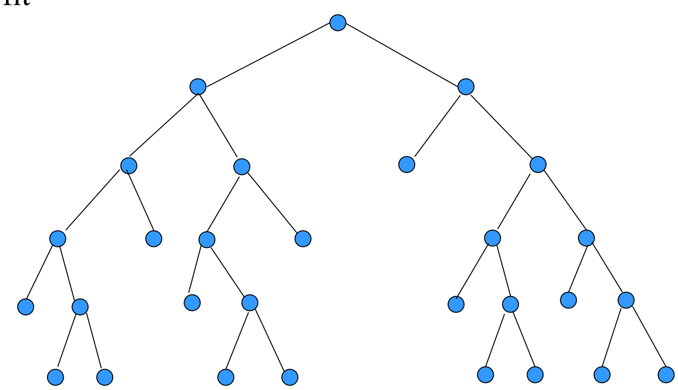
- We may split and classify very well the training set, but we may do worse in terms of the generalization error

Solutions to the overfitting problem:

- **Solution 1. Build the tree then prune the branches**
  - Build the tree, then eliminate leaves that overfit
  - Use **validation set** to test for the overfit
- **Solution 2. Prune while building the tree**
  - Test for the overfit in the tree building phase
  - Stop building the tree when performance on the **validation set** deteriorates

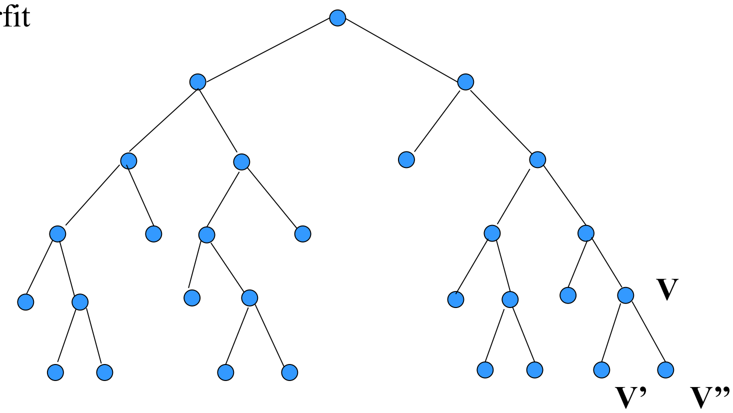
Decision tree learning

Backpruning: Prune branches of the tree built in the first phase in the botton-up fashion by using the validation set to test for the overfit



Decision tree learning

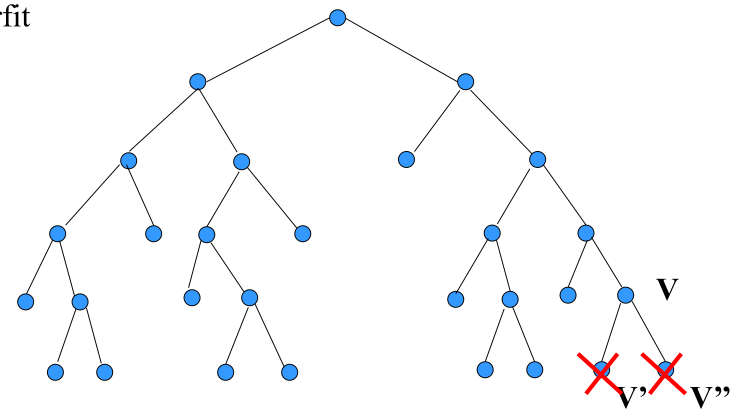
Backpruning: Prune branches of the tree built in the first phase in the botton-up fashion by using the validation set to test for the overfit



Compare: #Errors (V) vs #Error (V') + # Errors(V'')

Decision tree learning

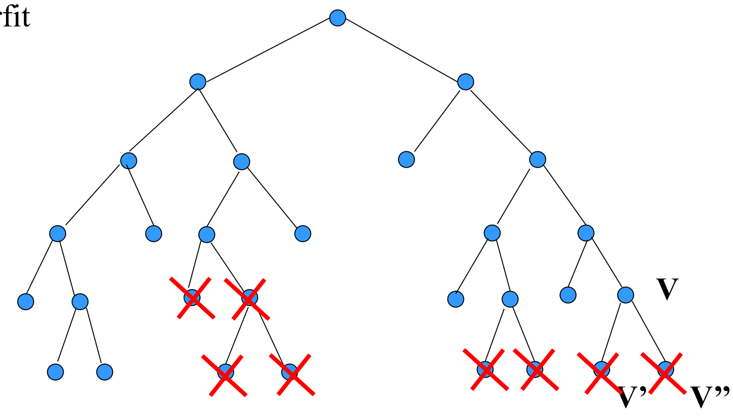
Backpruning: Prune branches of the tree built in the first phase in the botton-up fashion by using the validation set to test for the overfit



Compare: #Errors (V) < #Error (V') + # Errors(V'')

Decision tree learning

Backpruning: Prune branches of the tree built in the first phase in the botton-up fashion by using the validation set to test for the overfit



Compare: #Errors (V) < #Error (V') + # Errors(V'')

Nonparametric classification models

We have covered multiple non-parametric density estimation approaches

How can we use them in classification?

Nonparametric classification models

We have a set D of <**x**,y> pairs

We have a new data point **x** and want to assign it a class y

How ?

Algorithm 1. Generative model

Step 1: Estimate p(y=1) and p(y=0)

Step 2: Estimate p(**x** |y=1) and p(**x** |y=0) using nonparametric estimation methods and labels

Step 3: choose a class by comparing

p(**x** |y=1) p(y=1) with p(**x** |y=0) p(y=1)

Nonparametric classification models

We have a set D of <**x**,y> pairs

We have a new data point **x** and want to assign it a class y

Algorithm 2 (K nearest neighbors)

Recall:

Step 1: Find the closest K examples to x

Step 2: choose a class by considering the majority of the class labels

A special case: **the nearest neighbour algorithm**

Multiclass classification

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 K discriminant functions**

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

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

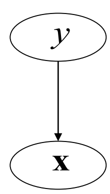
- $p(\mathbf{x} \mid y)$ = **Class-conditional distributions (densities)**

  k class-conditional distributions

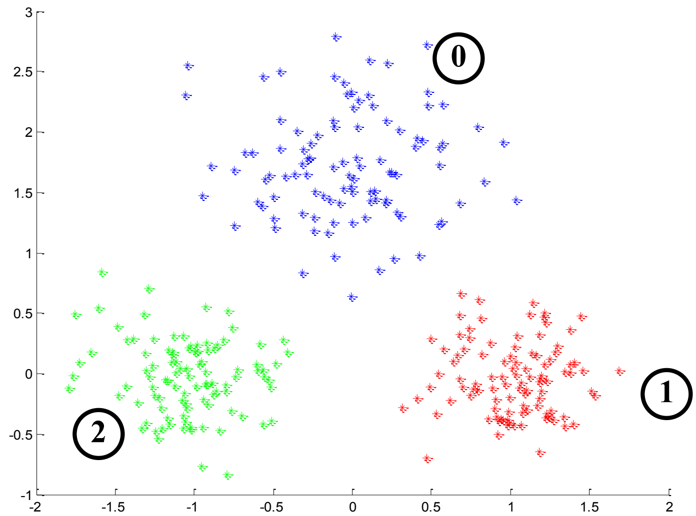
$$p(\mathbf{x} \mid y = i) \qquad \forall i \quad 0 \le i \le K - 1$$

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

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



Multi-way classification. Example



Making class decision

Discriminant functions:

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

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

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

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:

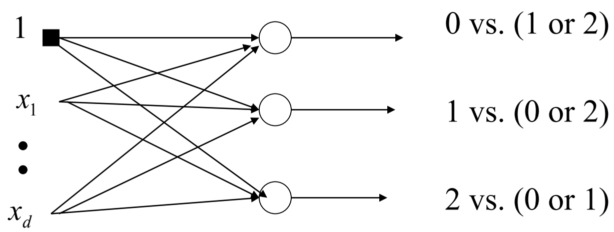
- Can we use binary classifiers to build the multi-class models?

One versus the rest (OVR)

Methods based on binary classification methods

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

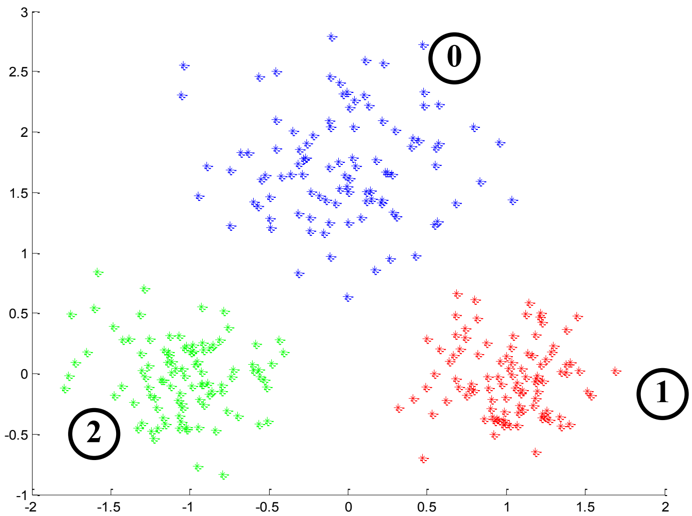
  A binary logistic regression on every class versus the rest (OvR)



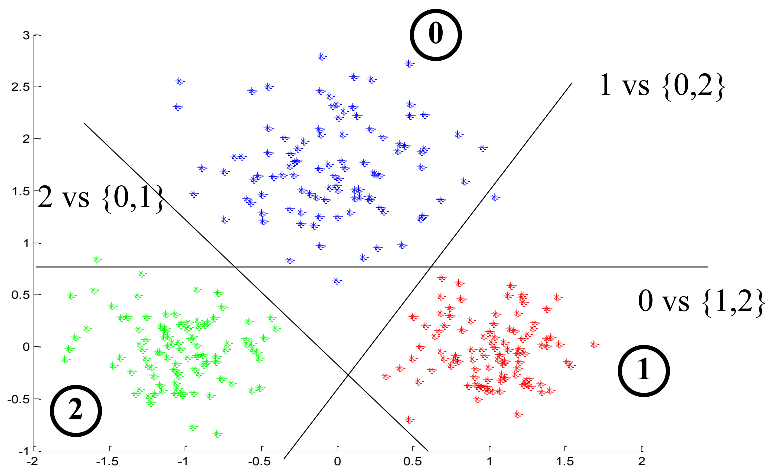
Class decision: class label for a 'singleton' class

– Does not work all the time

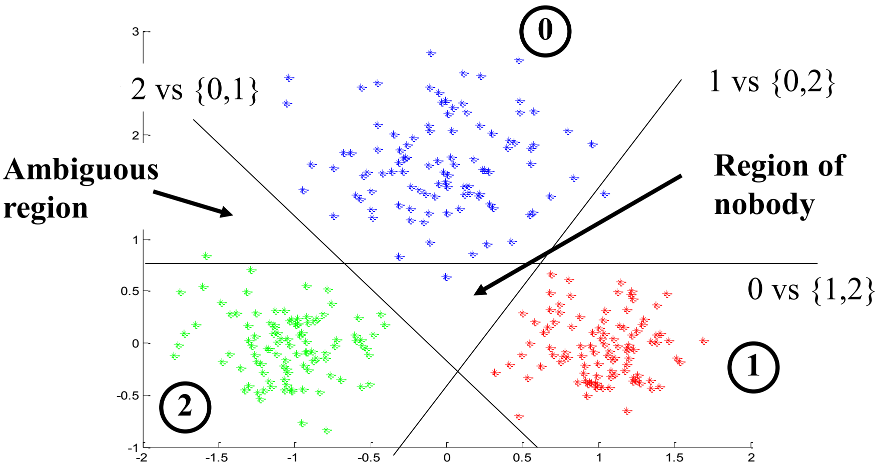
Multiclass classification. Example



Multiclass classification. Approach 1.



Multiclass classification. Approach 1.



One versus the rest (OVR)

Unclear how to decide on class in some regions

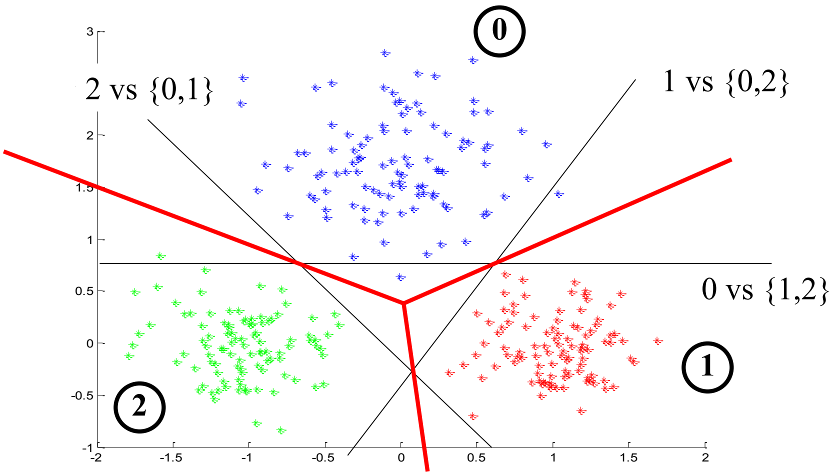
- **Ambigous region:**
  - 0 vs. (1 or 2) classifier says 0
  - 1 vs. (0 or 2) classifier says 1
- **Region of nobody:**
  - 0 vs. (1 or 2) classifier says (1 or 2)
  - 1 vs. (0 or 2) classifier says (0 or 2)
  - 2 vs (1 or 2) classifier says (1 or 2)

- **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

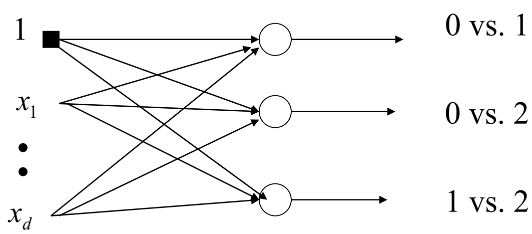
Multiclass classification. Approach 1.



One vs One (OVO)

Methods based on binary classification methods

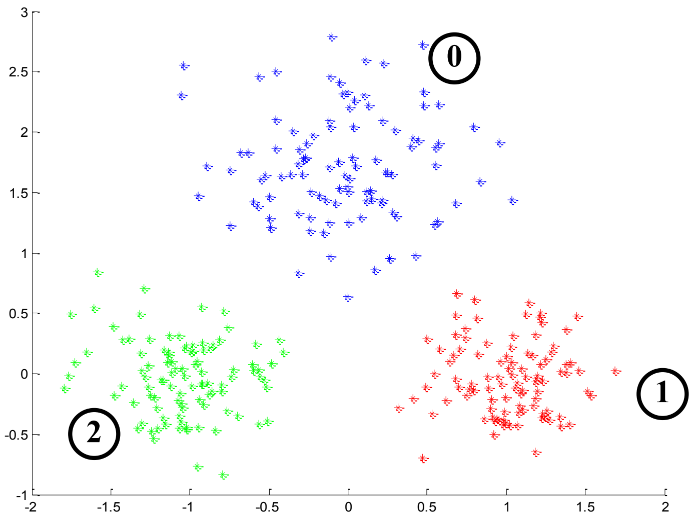
- **Assume:** we have 3 classes labeled 0,1,2
- **Approach 2:**
  - A binary logistic regression on all pairs



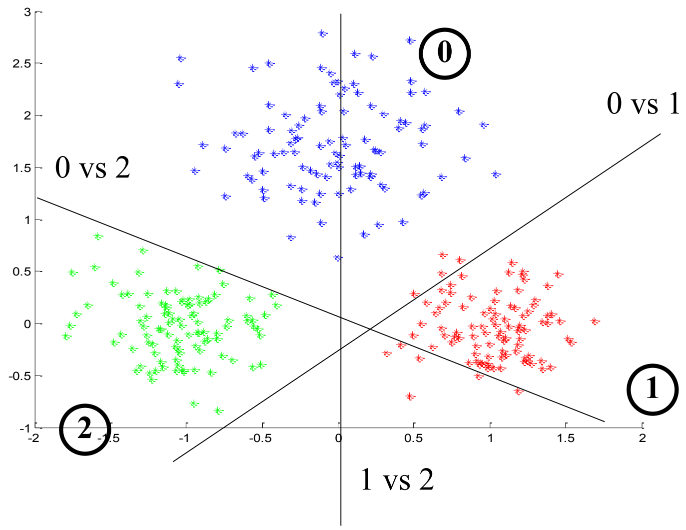
Class decision: class label based on who gets the majority

– Does not work all the time

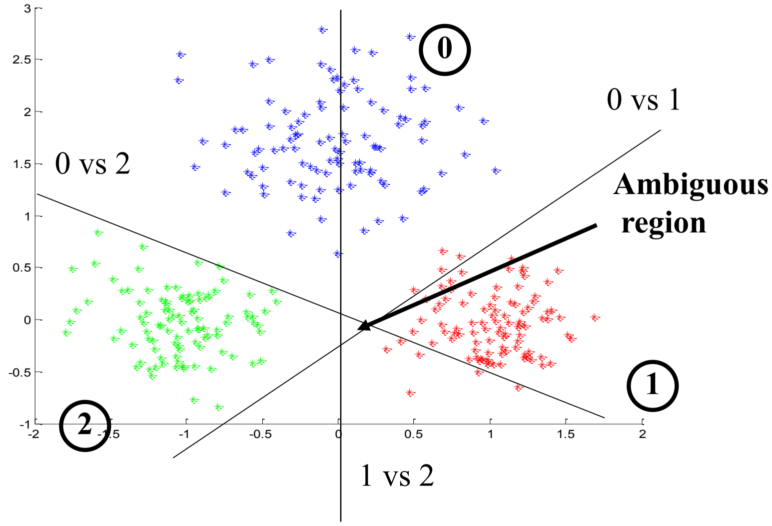
Multiclass classification. Example



Multiclass classification (OVO)



Multiclass classification OVO



One vs one (OVO) model

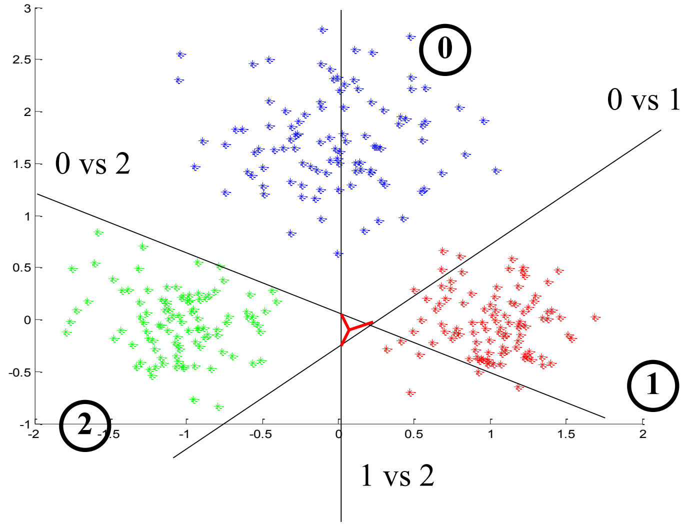
Unclear how to decide on class in some regions

- **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 discriminant functions for pairwise classifiers

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

Multiclass classification.



Multiclass classification

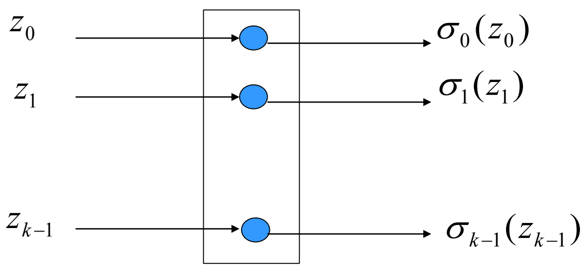
OVR and OVO:
- learn the discriminant functions for binary classification problems
- combine them to define the multiclass discriminant functions

Issues:
- calibration of the discriminant functions

Question:
- can we learn the discriminant function for the multiclass problem jointly

Softmax function

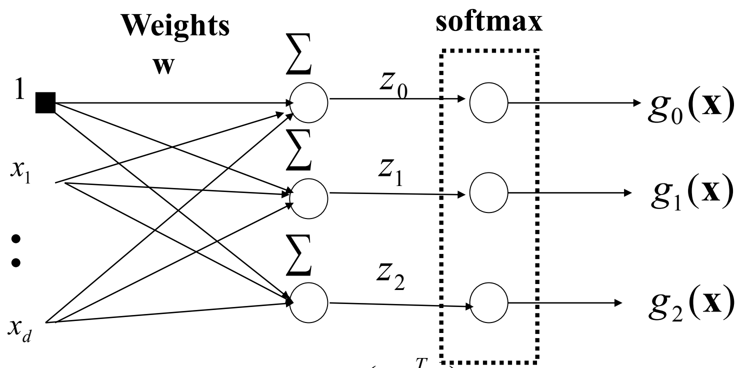
- **Multiple inputs → outputs probabilities**



$$\sigma_i(z_i) = \frac{\exp(z_i)}{\sum_{j=0}^{k-1} \exp(z_j)} \qquad \sum_{i=0}^{k-1} \sigma_i(z_i) = 1$$

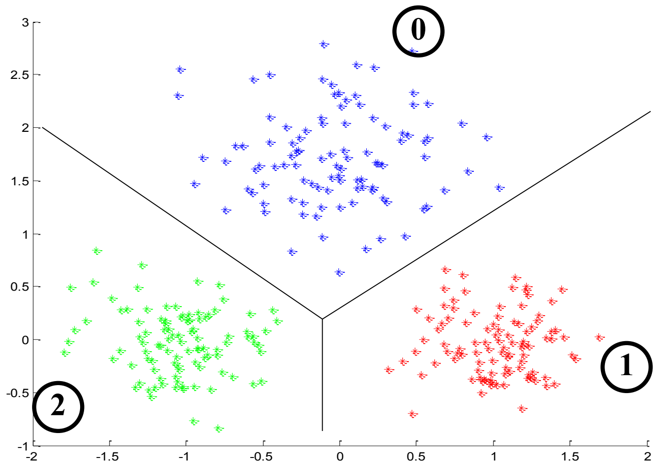
Multiclass classification with softmax

- learns the multiclass discriminant functions jointly



$$g_i(\mathbf{x}) = p(y = i \mid \mathbf{x}) = \frac{\exp(\mathbf{w}_i^T \mathbf{x})}{\sum_{j=0}^{k-1} \exp(\mathbf{w}_j^T \mathbf{x})} \qquad \sum_i g_i(\mathbf{x}) = 1$$

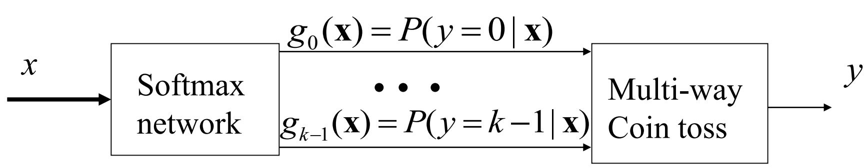
Multiclass classification with softmax



CS 2750 Machine Learning

Learning of the softmax model

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



Assume outputs y are transformed as follows

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

CS 2750 Machine Learning

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:

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

$$= \sum_{i=1,..n} \sum_{j=0}^{k-1} \log g_j(\mathbf{x}_i)^{y_{i,j}} = \sum_{i=1,..n} \sum_{j=0}^{k-1} y_{i,j} \log g_j(\mathbf{x}_i)$$

- **Objective to optimize** $J(D,\mathbf{w}) = -\sum_{i=1}^{n} \sum_{j=0}^{k-1} y_{i,j} \log g_j(\mathbf{x}_i)$

Learning of the softmax model

- **Error to optimize:**

$$J(D,\mathbf{w}) = -\sum_{i=1}^{n} \sum_{j=0}^{k-1} y_{i,j} \log g_j(\mathbf{x}_i)$$

- **Gradient**

$$\frac{\partial}{\partial w_{ju}} J(D\ ,\mathbf{w}) = \sum_{i=1}^{n} -x_{i,u}(y_{i,j} - g_j(\mathbf{x}_i))$$

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

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \alpha \sum_{i=1}^{n} (y_{i,j} - g_j(\mathbf{x}_i))\mathbf{x}_i$$

- We have to update the weights of k networks