CS 1675 Introduction to Machine Learning
Lecture 20

Multi-class classification

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

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**

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

Discriminant functions

- A common way to represent a **classifier is by using**
  - **Discriminant functions**
- Works for both **the binary and multi-way classification**
- **Idea:**
  - For every class $i = 0,1, \ldots k$ 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 $g_i(\mathbf{x})$

$$y^* = \arg\max_i \; g_i(\mathbf{x})$$

Multiclass classification

Approaches to classification:

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

Question: How to learn models for more than 2 classes?

Generative model approach

Indirect:

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

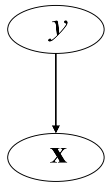
$$g_i(\mathbf{x}) = \log p(y = i \mid \mathbf{x})$$

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

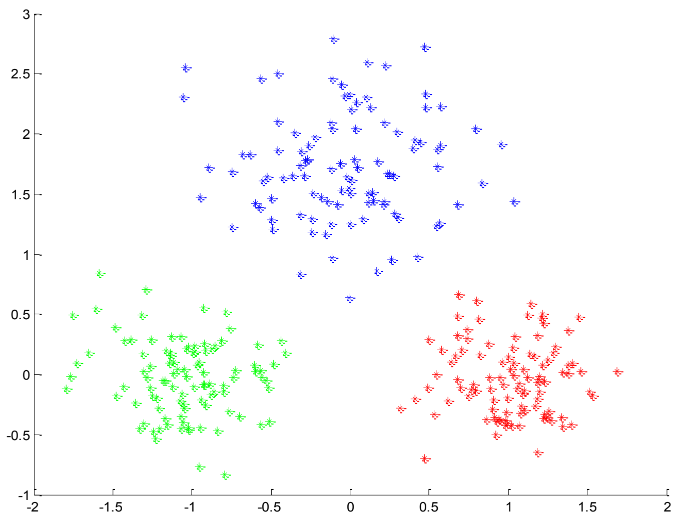
$$p(\mathbf{x} \mid y = i) \qquad \forall i \quad 0 \le i \le K-1$$



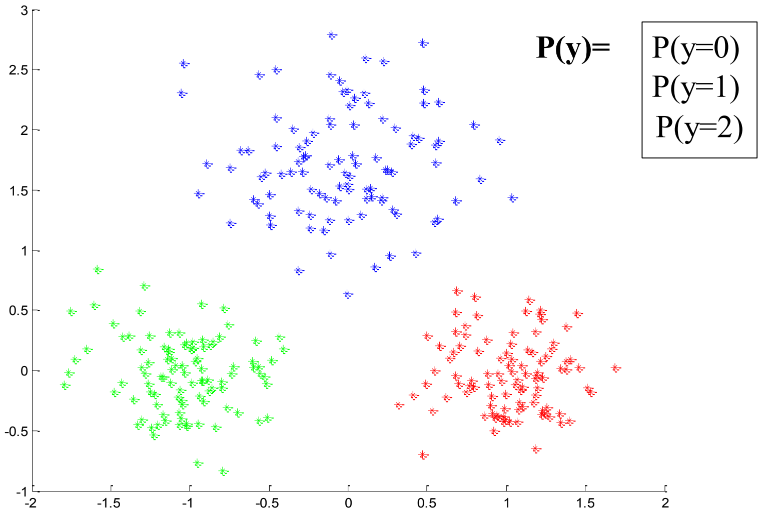
- $p(y)$ = **Priors on classes**
- **-** probability of class y

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

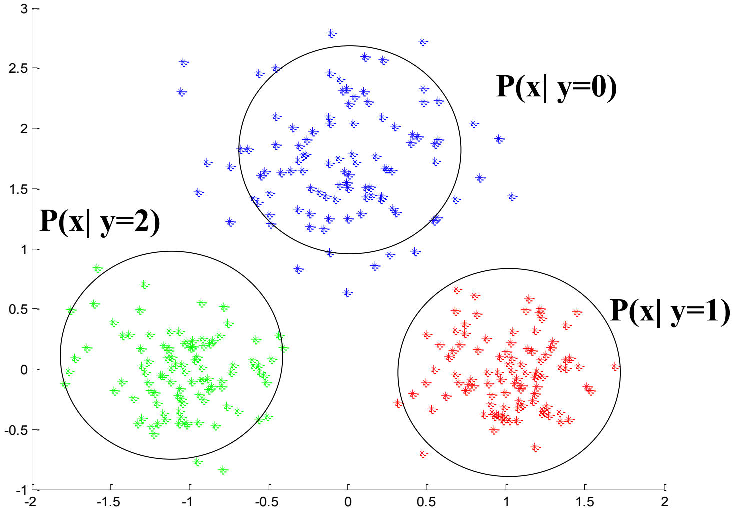
Multi-way classification:



Multi-way classification:



Multi-way classification:



Making class decision

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

Class choice $i = \arg\max_{i=0,\ldots k-1} g_i(\mathbf{x})$

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

- choose the class with higher posterior probability

Discriminative approach

- **Parametric models** 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:

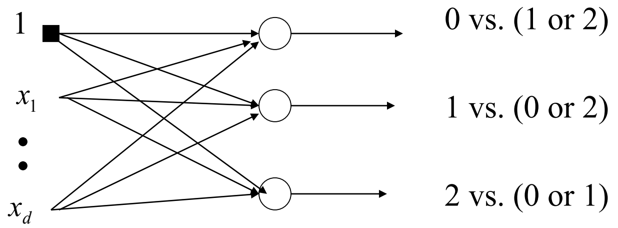
- Can we use binary classifiers to build the multi-class models?

One versus the rest (OvR)

Methods based on binary classification methods

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

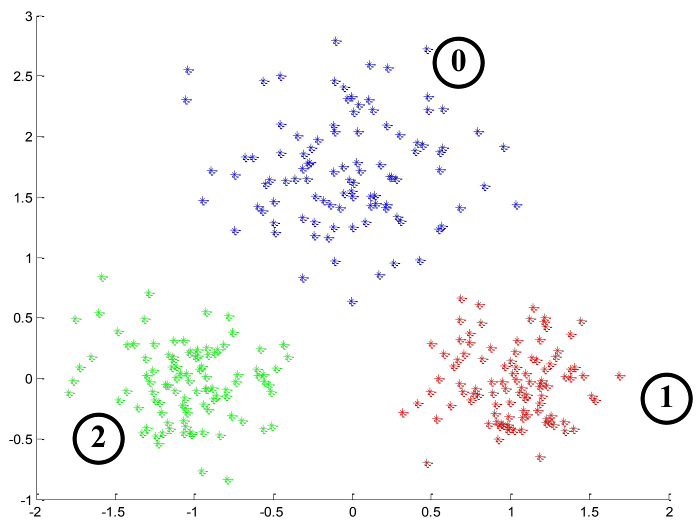
  A binary logistic regression on every class versus the rest (OvR)



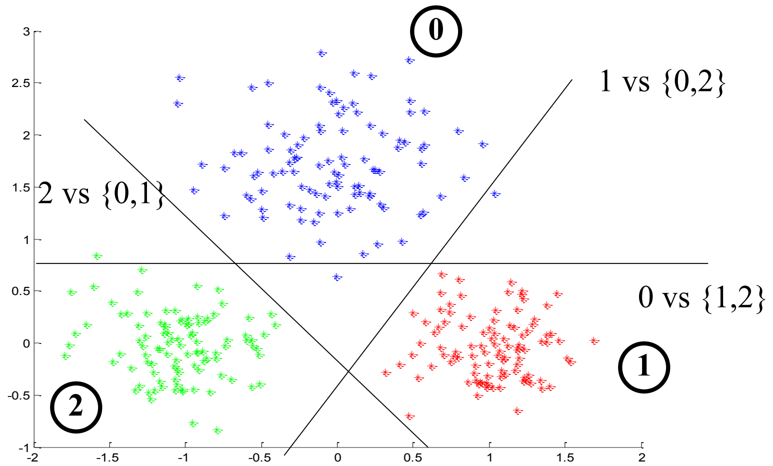
Class decision: class label for a ‘singleton’ class

– Does not work all the time

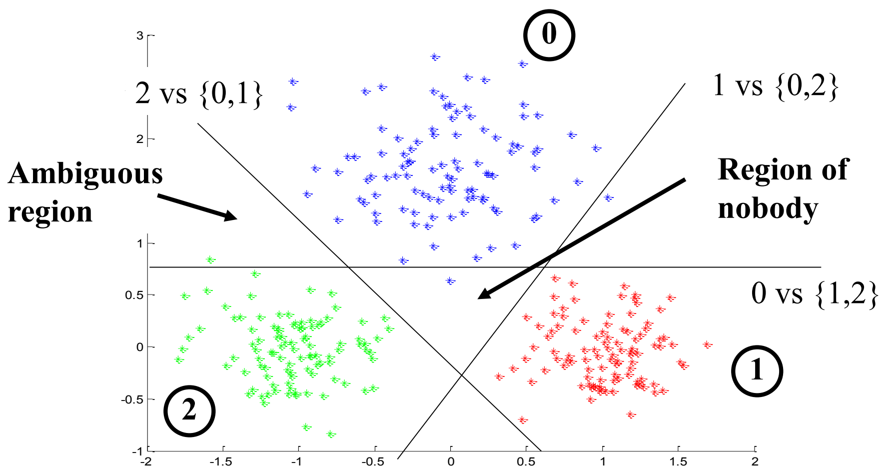
Multiclass classification. Example



Multiclass classification. Approach 1.



Multiclass classification. Approach 1.



One versus the rest (OvR)

Unclear how to decide on class in some regions

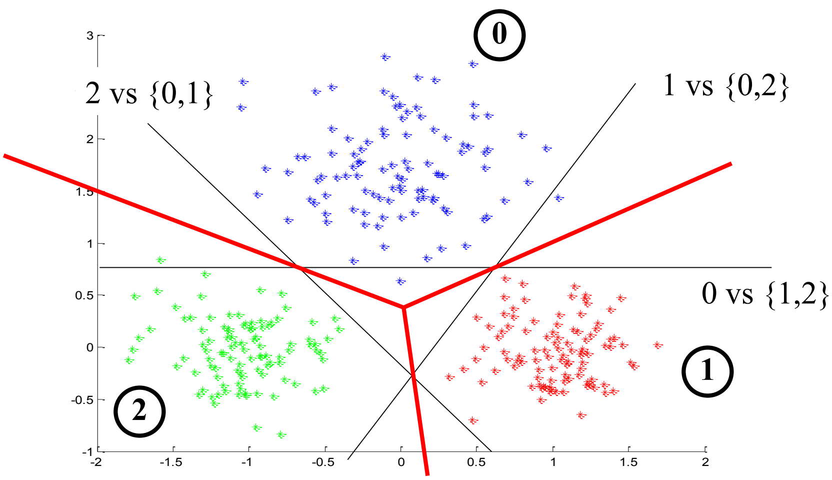
- **Ambigous region:**
  - 0 vs. (1 or 2) classifier says 0
  - 1 vs. (0 or 2) classifier says 1
- **Region of nobody:**
  - 0 vs. (1 or 2) classifier says (1 or 2)
  - 1 vs. (0 or 2) classifier says (0 or 2)
  - 2 vs (1 or 2) classifier says (1 or 2)

- **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

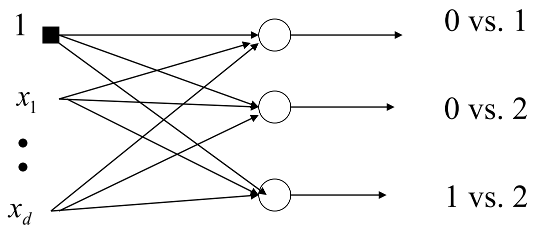
Multiclass classification. Approach 1.



Discriminative approach

Methods based on binary classification methods

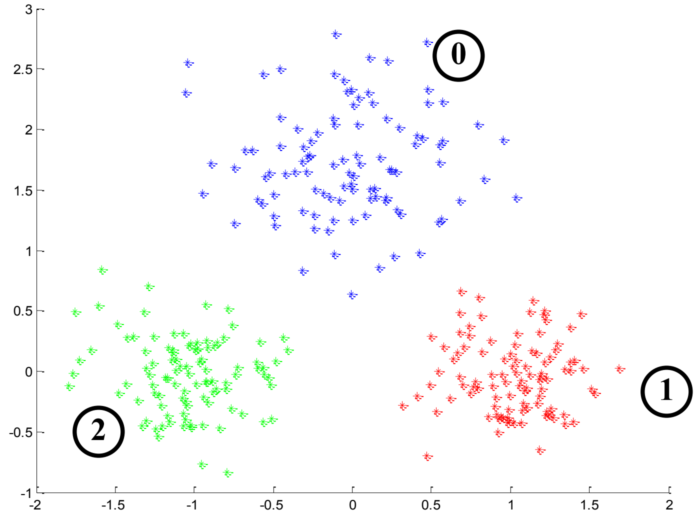
- **Assume:** we have 3 classes labeled 0,1,2
- **Approach 2:**
  - A binary logistic regression on all pairs



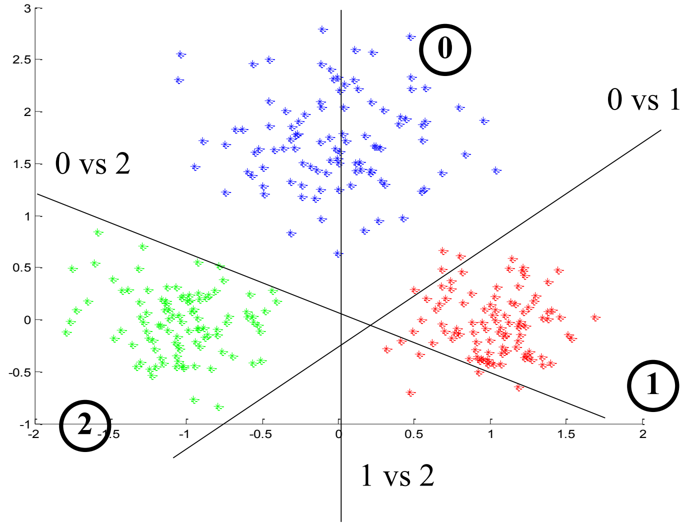
Class decision: class label based on who gets the majority

- Does not work all the time

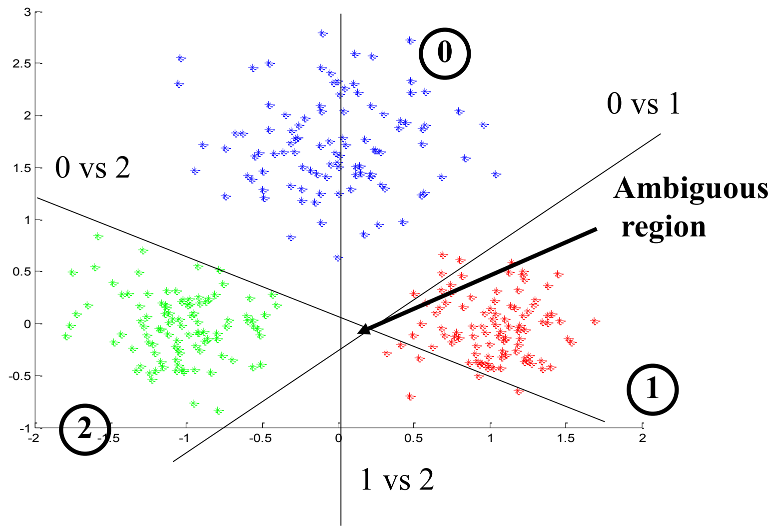
Multiclass classification. Example



Multiclass classification. Approach 2



Multiclass classification. Approach 2



One vs one model

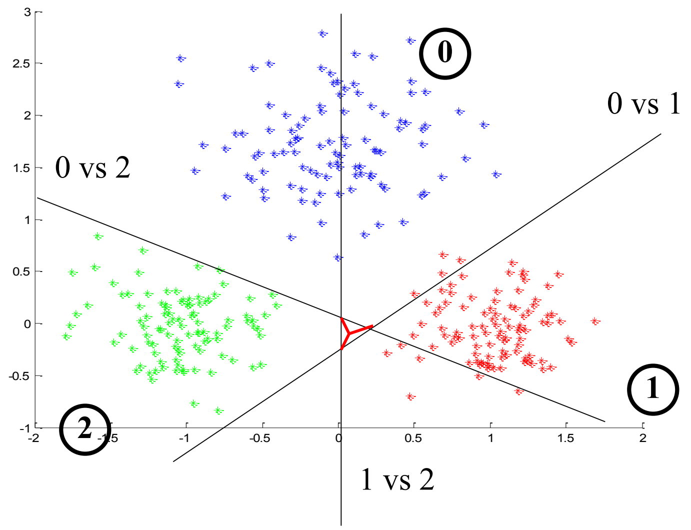
Unclear how to decide on class in some regions

- **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

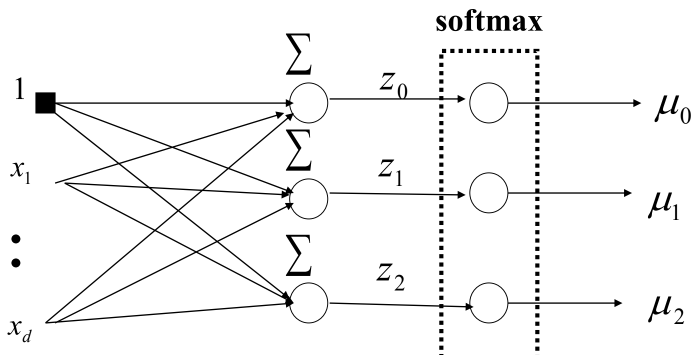
$$g_i(\mathbf{x}) = \sum_{j \neq i} g_{i\ vs\ j}(\mathbf{x})$$

Multiclass classification. Approach 2



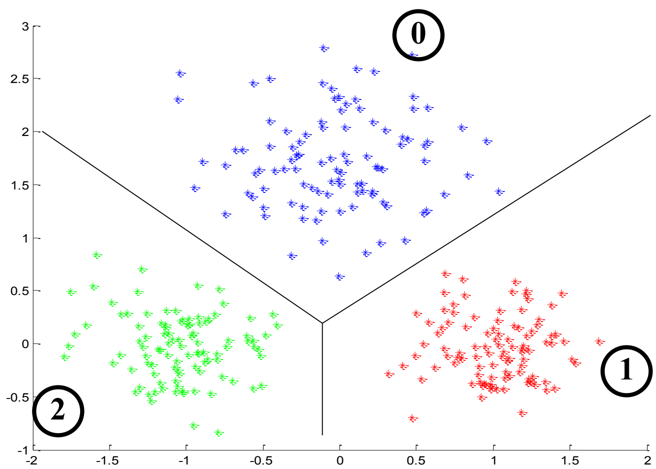
Multiclass classification with softmax

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



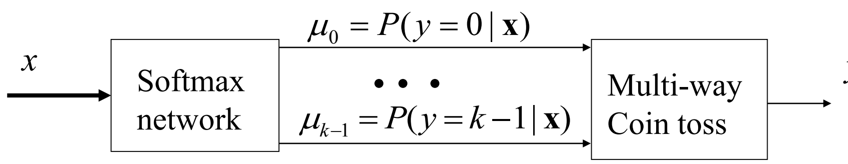
$$g_i(\mathbf{x}) = p(y = i \mid \mathbf{x}) = \mu_i = \frac{\exp(\mathbf{w}_i^T \mathbf{x})}{\sum_j \exp(\mathbf{w}_j^T \mathbf{x})} \qquad \sum_i g_i(\mathbf{x}) = 1$$

Multiclass classification with softmax



Learning of the softmax model

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



Assume outputs y are transformed as follows

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

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**

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

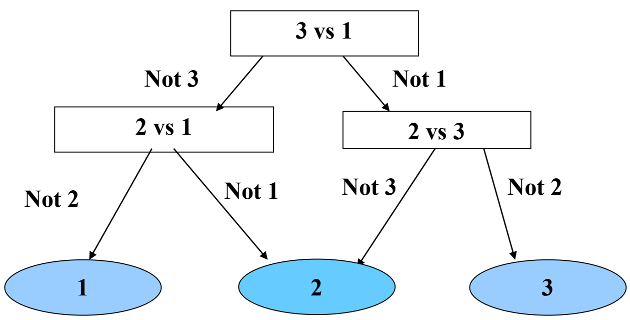
- The same very easy **gradient update** as used for the binary logistic regression

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

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

Multi-way classification

- **Yet another approach to multiway classification**

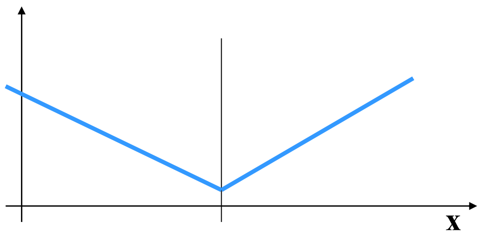


Mixture of experts model

- **Ensamble methods:**
  - Use a combination of simpler learners/model to improve their predictions
- **Mixture of expert model:**
  - Different input regions covered with different learners
  - A "soft" switching between learners

- **Mixture of experts**

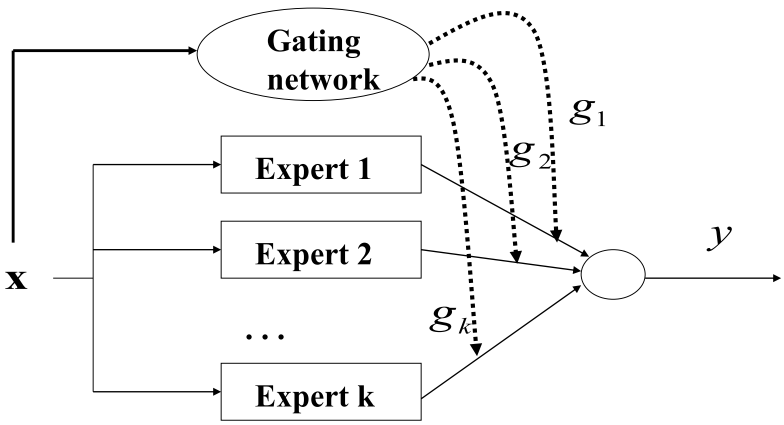
  Expert = learner



Mixture of experts model

- **Gating network** : decides what expert to use

  $g_1, g_2, \ldots g_k$ - **gating functions**



Learning mixture of experts

- **Learning consists of two tasks:**
  - Learn the parameters of individual expert networks
  - Learn the parameters of the gating (switching) network
    - Decides where to make a split
- **Assume:** gating functions give probabilities

$$0 \le g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots g_k(\mathbf{x}) \le 1 \qquad \sum_{u=1}^{k} g_u(\mathbf{x}) = 1$$

- Based on the probability we partition the space
  - partitions belongs to different experts
- How to model the gating network?
  - **A multi-way classifier model:**
    - **softmax model**

Learning mixture of experts

- Assume we have a **set of linear experts**

$$\mu_i = \boldsymbol{\theta}_i^T \mathbf{x}$$ **(Note: bias terms are hidden in x)**

- Assume a **softmax gating network**

$$g_i(\mathbf{x}) = \frac{\exp(\boldsymbol{\eta}_i^T \mathbf{x})}{\sum_{u=1}^{k} \exp(\boldsymbol{\eta}_u^T \mathbf{x})} \approx p(\omega_i \mid \mathbf{x}, \boldsymbol{\eta})$$

- Likelihood of y (linear regression – assume errors for different experts are normally distributed with the same variance)

$$P(y \mid \mathbf{x}, \Theta, \boldsymbol{\eta}) = \sum_{i=1}^{k} P(\omega_i \mid \mathbf{x}, \boldsymbol{\eta}) p(y \mid \mathbf{x}, \omega_i, \boldsymbol{\Theta})$$

$$= \sum_{i=1}^{k} \left[\frac{\exp(\boldsymbol{\eta}_i^T \mathbf{x})}{\sum_{j=1}^{k} \exp(\boldsymbol{\eta}_j^T \mathbf{x})} \right] \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\| y - \mu_i \|^2}{2\sigma^2} \right) \right]$$

Learning mixture of experts

Learning of parameters of expert models:

On-line update rule for parameters $\boldsymbol{\theta}_i$ of expert *i*

– If we know the expert that is responsible for **x**

$$\theta_{ij} \leftarrow \theta_{ij} + \alpha_{ij}(y - \mu_i)x_j$$

– If we do not know the expert

$$\theta_{ij} \leftarrow \theta_{ij} + \alpha_{ij} h_i (y - \mu_i)x_j$$

h_i **- responsibility of the *i*th expert = a kind of posterior**

$$h_i(\mathbf{x}, y) = \frac{g_i(\mathbf{x}) p(y \mid \mathbf{x}, \omega_i, \boldsymbol{\theta})}{\sum_{u=1}^{k} g_u(\mathbf{x}) p(y \mid \mathbf{x}, \omega_u, \boldsymbol{\theta})} = \frac{g_i(\mathbf{x}) \exp\left(-1/2 \|y - \mu_i\|^2\right)}{\sum_{u=1}^{k} g_u(\mathbf{x}) \exp\left(-1/2 \|y - \mu_u\|^2\right)}$$

$g_i(\mathbf{x})$ **- a prior** exp(...) **- a likelihood**

Learning mixtures of experts

Learning of parameters of the gating/switching network:

- **On-line learning of gating network parameters $\boldsymbol{\eta}_i$**

$$\eta_{ij} \leftarrow \eta_{ij} + \beta_{ij}(h_i(\mathbf{x}, y) - g_i(\mathbf{x}))x_j$$

- The learning with conditional mixtures can be extended to learning of parameters of an **arbitrary expert network**
  – e.g. logistic regression, multilayer neural network

$$\theta_{ij} \leftarrow \theta_{ij} + \beta_{ij} \frac{\partial l}{\partial \theta_{ij}}$$

$$\frac{\partial l}{\partial \theta_{ij}} = \frac{\partial l}{\partial \mu_i} \frac{\partial \mu_i}{\partial \theta_{ij}} = h_i \frac{\partial \mu_i}{\partial \theta_{ij}}$$