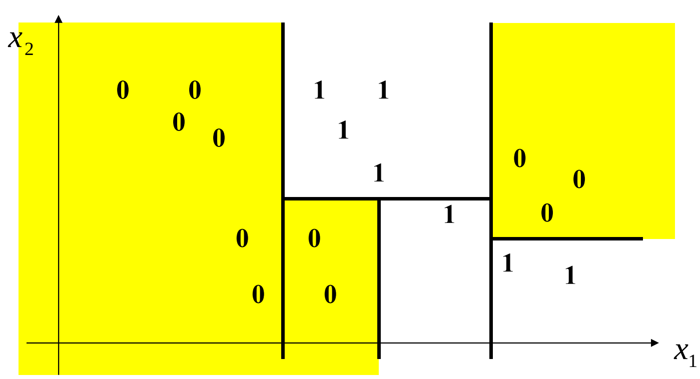
CS 2750 Machine Learning
Lecture 20

Ensemble methods: Mixtures of experts

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

Reviewing Decision trees

- An approach to classification that:
  - **Partitions the input space to regions**
  - **Classifies independently in every region**



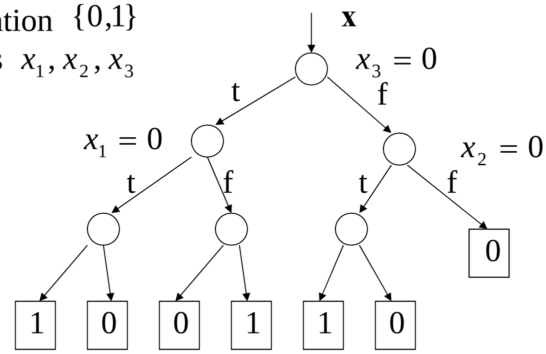
Decision trees

- The partitioning idea is used in the **decision tree model**:
  - Split the space recursively according to inputs in **x**
  - Classify (assign class label) at the bottom of the tree

Example:

Binary classification $\{0,1\}$

Binary attributes x_1, x_2, x_3



Decision tree learning

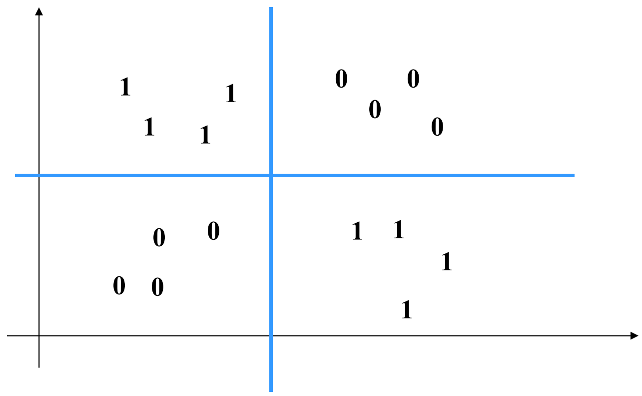
- **Greedy learning algorithm:**

  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)

Limitations of Decision trees

- **Greedy learning methods: a** combination of two or more attributes improves the impurity
- **Rectangular regions**

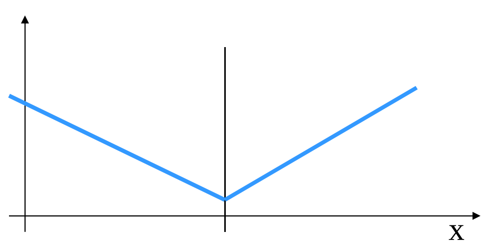


Mixture of experts model

- **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**

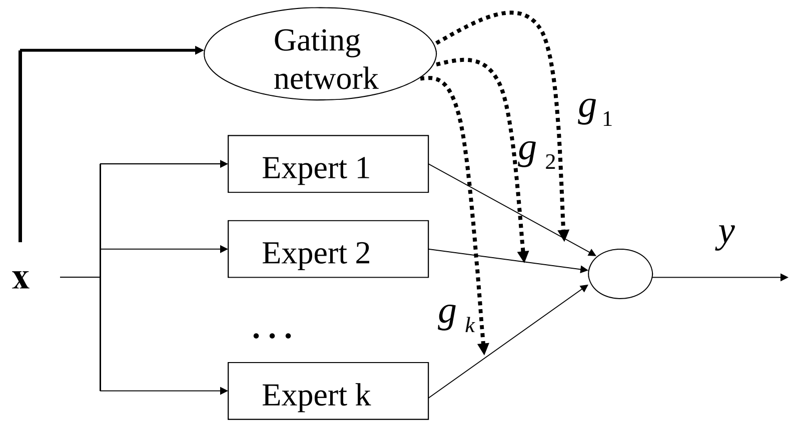
  Expert = learner



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**

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

- 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)

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

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

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

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

$g_i(\mathbf{x})$ - a prior $\qquad$ 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

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

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

Learning mixture of experts

EM algorithm for learning the mixture components

Algorithm:

Initialize parameters Θ

Repeat

Set $\Theta' = \Theta$

1. **Expectation step**

$$Q(\Theta \mid \Theta') = E_{H|\mathbf{X},\mathbf{Y},\Theta'} \log P(\mathbf{H}, \mathbf{Y} \mid \mathbf{X}, \Theta, \xi)$$

2. **Maximization step**

$$\Theta = \arg\max_{\Theta} Q(\Theta \mid \Theta')$$

until no or small improvement in $Q(\Theta \mid \Theta')$

– **Hidden variables are identities of expert networks responsible for (x,y) data points**

EM for Learning mixture of experts

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

$$\mu_i = \boldsymbol{\theta}_i^T \mathbf{x}$$

- Assume a **softmax gating network**

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}, \boldsymbol{\eta})$$

- **Q function to optimize**

$$Q(\Theta \mid \Theta') = E_{H \mid \mathbf{X},\mathbf{Y},\Theta'} \log P(\mathbf{H}, \mathbf{Y} \mid \mathbf{X}, \Theta, \xi)$$

- **Assume:**
  - l indexes different data points
  - δ_i^l an indicator variable for the data point l to be covered by an expert i

$$Q(\Theta \mid \Theta') = \sum_l \sum_i E(\delta_i^l \mid \mathbf{x}^l, y^l, \Theta', \boldsymbol{\eta}') \log(P(y^l, \omega_i \mid \mathbf{x}^l, \Theta, \boldsymbol{\eta}))$$

Learning mixture of experts with EM

- **Assume:**
  - l indexes different data points
  - δ_i^l an indicator variable for data point l and expert i

$$Q(\Theta \mid \Theta') = \sum_l \sum_i E(\delta_i^l \mid \mathbf{x}^l, y^l, \Theta', \boldsymbol{\eta}') \log(P(y^l, \omega_i \mid \mathbf{x}^l, \Theta, \boldsymbol{\eta}))$$

Responsibility of the expert *i* for *(x,y)*

$$E(\delta_i^l \mid \mathbf{x}^l, y^l, \Theta', \boldsymbol{\eta}') = h_i^l(\mathbf{x}^l, y^l) = \frac{g_i(\mathbf{x}^l)\, p(y \mid \mathbf{x}^l, \omega_i, \boldsymbol{\theta}')}{\sum_{u=1}^{k} g_u(\mathbf{x}^l)\, p(y^l \mid \mathbf{x}^l, \omega_u, \boldsymbol{\theta}')}$$

$$Q(\Theta \mid \Theta') = \sum_l \sum_i h_i^l(\mathbf{x}^l, y^l) \log(P(y^l, \omega_i \mid \mathbf{x}^l, \Theta, \boldsymbol{\eta}))$$

EM for learning the mixture of experts

- The maximization step boils down to the problem that is equivalent to the problem of finding the ML estimates of the parameters of the expert and gating networks

$$Q(\Theta \mid \Theta') = \sum_{l} \sum_{i} h_i^l(\mathbf{x}^l, y^l) \log(P(y^l, \omega_i \mid \mathbf{x}^l, \Theta, \boldsymbol{\eta}))$$

$$\log(P(y^l, \omega_i \mid \mathbf{x}^l, \Theta, \boldsymbol{\eta})) = \log P(y^l \mid \omega_i, \mathbf{x}^l, \Theta) + \log P(\omega_i \mid \mathbf{x}^l, \boldsymbol{\eta})$$

Expert network i
(Linear regression)

Gating network
(Softmax)

- Note that any optimization technique can be applied in this step

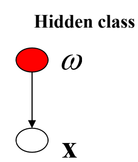
Learning mixture of experts

- **Note that we can use different expert and gating models**
- **For example:**
  – Experts: logistic regression models

$$y_i = 1/(1 + \exp(-\boldsymbol{\theta}_i^T \mathbf{x}))$$

  – Gating network: a **generative latent variable model**

$$g_i(\mathbf{x}) = P(\omega_i \mid \mathbf{x}, \boldsymbol{\eta})$$

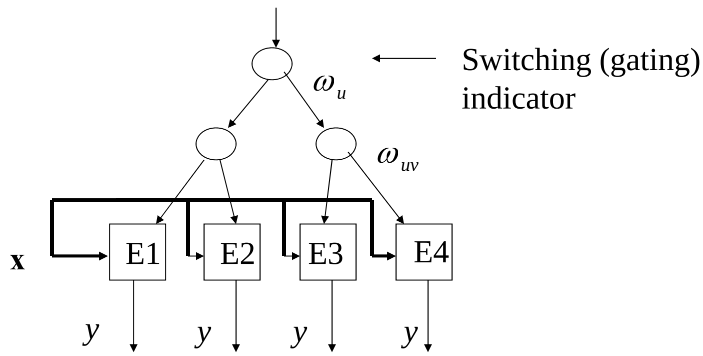


- Likelihood of y:

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

Hierarchical mixture of experts

- **Mixture of experts**: define a probabilistic split
- The idea can be extended to a **hierarchy of experts** (a kind of a probabilistic decision tree)



Hierarchical mixture model

An output is conditioned (gated) on multiple mixture levels

$$P(y \mid \mathbf{x}, \Theta) = \sum_u P(\omega_u \mid \mathbf{x}, \eta) \sum_v p(\omega_{uv} \mid \mathbf{x}, \omega_u, \xi_u) .. \sum_s P(\omega_{uv..s} \mid \mathbf{x}, \omega_u, \omega_{uv}, \ldots) \underline{P(y \mid \mathbf{x}, \omega_u, \omega_{uv}, .., \theta_{uv..s})}$$

Individual experts

- **Define** $\Omega_{uv..s} = \{\omega_u, \omega_{uv}, \ldots \omega_{uv..s}\}$

$$P(\Omega_{uv..s} \mid \mathbf{x}, \Theta) = P(\omega_u \mid \mathbf{x}) P(\omega_{uv} \mid \mathbf{x}, \omega_u) .. P(\omega_{uv..s} \mid \mathbf{x}, \omega_u, \omega_{uv}, \ldots)$$

- **Then**

$$P(y \mid \mathbf{x}, \Theta) = \sum_u \sum_v \ldots \sum_s P(\Omega_{uv..s} \mid \mathbf{x}, \Theta) P(y \mid \mathbf{x}, \Omega_{uv..s}, \Theta)$$

- Mixture model is a kind of soft decision tree model

- with a fixed tree structure !!

Hierarchical mixture of experts

- Multiple levels of probabilistic gating functions

$$g_u(\mathbf{x}) = P(\omega_u \mid \mathbf{x}, \Theta) \qquad g_{v|u}(\mathbf{x}) = P(\omega_{uv} \mid \mathbf{x}, \omega_u \Theta)$$

- Multiple levels of responsibilities

$$h_u(\mathbf{x}, y) = P(\omega_u \mid \mathbf{x}, y, \Theta) \qquad h_{v|u}(\mathbf{x}, y) = P(\omega_{uv} \mid \mathbf{x}, y, \omega_u, \Theta)$$

- How they are related?

responsibility

$$P(\omega_{uv} \mid \mathbf{x}, y, \omega_u, \Theta) = \frac{P(y \mid \mathbf{x}, \omega_u, \omega_{uv}, \Theta)\overbrace{P(\omega_{uv} \mid \mathbf{x}, \omega_u, \Theta)}^{g_{v|u}(\mathbf{x})}}{\underbrace{\sum_v P(y \mid \mathbf{x}, \omega_u, \omega_{uv}, \Theta) P(\omega_{uv} \mid \mathbf{x}, \omega_u, \Theta)}}$$

$$= \sum_v P(y, \omega_{uv} \mid \mathbf{x}, \omega_u, \Theta) = P(y \mid \mathbf{x}, \omega_u, \Theta)$$

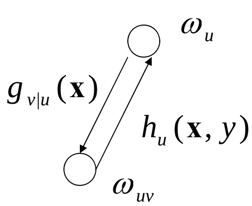
CS 2750 Machine Learning

Hierarchical mixture of experts

- **Responsibility for the top layer**

$$h_u(\mathbf{x}, y) = P(\omega_u \mid \mathbf{x}, y, \Theta) = \frac{P(y \mid \mathbf{x}, \omega_u, \Theta) P(\omega_u \mid \mathbf{x}, \Theta)}{\sum_u P(y \mid \mathbf{x}, \omega_u, \Theta) P(\omega_u \mid \mathbf{x}, \Theta)}$$

- But $P(y \mid \mathbf{x}, \omega_u \Theta)$ is computed while computing $h_{v|u}(\mathbf{x}, y) = P(\omega_{uv} \mid \mathbf{x}, y, \omega_u, \Theta)$
- **General algorithm:**
  - **Downward sweep; calculate** $g_{v|u}(x) = P(\omega_{uv} \mid \mathbf{x}, \omega_u, \Theta)$
  - **Upward sweep; calculate** $h_u(\mathbf{x}, y) = P(\omega_u \mid \mathbf{x}, y, \Theta)$



CS 2750 Machine Learning

On-line learning

- Assume linear experts $\mu_{uv} = \boldsymbol{\theta}_{uv}^{T}\mathbf{x}$
- **Gradients (vector form):**

$$\frac{\partial l}{\partial \boldsymbol{\theta}_{uv}} = h_u h_{v|u}(y - \mu_{uv})\mathbf{x}$$

$$\frac{\partial l}{\partial \boldsymbol{\eta}} = (h_u - g_u)\mathbf{x}$$ Top level (root) node

$$\frac{\partial l}{\partial \boldsymbol{\xi}} = h_u(h_{v|u} - g_{v|u})\mathbf{x}$$ Second level node

- Again: can it can be extended to different expert networks