Ensamble methods. Mixtures of experts

Mixture of experts model

• **Ensamble methods:**
  – Use a combination of simpler learners to improve predictions
• **Mixture of expert model:**
  – Covers different input regions with different learners
  – A “soft” switching between learners

• **Mixture of experts**
  Expert = learner

Milos Hauskrecht
milos@cs.pitt.edu
5329 Sennott Square
Mixture of experts model

- **Gating network**: decides what expert to use
  - $g_1, g_2, ..., g_k$ - gating functions

```
Gating
network
```

Learning mixture of experts

- **Learning consists of two tasks**:  
  - Learn the parameters of individual expert networks  
  - Learn the parameters of the gating network  
    - Decides where to make a split
- **Assume**: gating functions give probabilities  
  - $0 \leq g_1(x), g_2(x), ..., g_k(x) \leq 1$  
  - $\sum_{u=1}^{k} g_u(x) = 1$

- Based on the probability we partition the space  
  - partitions belongs to different experts
- How to model the gating network?  
  - **A multiway classifier model**:  
    - softmax model  
    - a generative classifier model
Learning mixture of experts

- Assume we have a set of linear experts
  \[ \mu_i = \theta_i^T x \] (Note: bias terms are hidden in x)
- Assume a softmax gating network
  \[ g_i(x) = \frac{\exp(\eta_i^T x)}{\sum_{u=1}^{k} \exp(\eta_u^T x)} \approx p(\omega_i \mid x, \eta) \]

• Likelihood of \( y \) (assumed that errors for different experts are normally distributed with the same variance)
  \[
P(y \mid x, \Theta, \eta) = \sum_{i=1}^{k} P(\omega_i \mid x, \eta) p(y \mid x, \omega_i, \Theta)
  = \sum_{i=1}^{k} \left[ \frac{\exp(\eta_i^T x)}{\sum_{j=1}^{k} \exp(\eta_j^T x)} \right] \left[ \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{\|y - \mu_i\|^2}{2\sigma^2} \right) \right]
\]

Gradient learning.

**On-line update rule for parameters \( \Theta_i \) of expert \( i \)**
- If we know the expert that is responsible for \( x \)
  \[ \theta_{y} \leftarrow \theta_{y} + \alpha_{y} (y - \mu_i) x_j \]
- If we do not know the expert
  \[ \theta_{y} \leftarrow \theta_{y} + \alpha_{y} h_{i}(y - \mu_{i}) x_j \]

\( h_{i} \) - responsibility of the \( i \)th expert = a kind of posterior

\[
h_{i}(x, y) = \frac{g_{i}(x) p(y \mid x, \omega_{i}, \theta)}{\sum_{u=1}^{k} g_{u}(x) p(y \mid x, \omega_{u}, \theta)} = \frac{g_{i}(x) \exp \left( -1/2\|y - \mu_{i}\|^2 \right)}{\sum_{u=1}^{k} g_{u}(x) \exp \left( -1/2\|y - \mu_{u}\|^2 \right)}
\]

\( g_{i}(x) \) - a prior \( \exp(...) \) - a likelihood


Learning mixtures of experts

**Gradient methods**

- **On-line learning of gating network parameters** $\eta_i$

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

- The learning with conditioned 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** offers an alternative way to learn the mixture

**Algorithm:**

Initialize parameters $\Theta$

Repeat

Set $\Theta' = \Theta$

1. **Expectation step**

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

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
Learning mixture of experts with EM

• Assume we have a set of linear experts
  \[ \mu_i = \theta_i^T x \]
• Assume a softmax gating network
  \[ g_i(x) = P(\omega_i | x, \eta) \]
• Q function to optimize
  \[ Q(\Theta | \Theta') = E_{H,X,Y,\Theta'} \log P(H,Y | X, \Theta, \xi) \]
• Assume:
  – \( l \) indexes different data points
  – \( \delta_i^l \) an indicator variable for the data point \( l \) to be covered by an expert \( i \)
  \[ Q(\Theta | \Theta') = \sum_l \sum_i E(\delta_i^l | x^l, y^l, \Theta', \eta') \log(P(y^l, \omega_i | x^l, \Theta, \eta)) \]
Learning mixture of experts with EM

- 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

\[ Q(\Theta | \Theta') = \sum_i \sum_l h_i^l(x_i, y_i) \log(P(y_i, \omega_i | x_i, \Theta, \eta)) \]

\[ \log(P(y_i, \omega_i | x_i, \Theta, \eta)) = \log P(y_i | \omega_i, x_i, \Theta) + \log P(\omega_i | x_i, \eta) \]

- 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
    \[ y_i = 1/(1 + \exp(-\theta_i^T x)) \]
  - Gating network: a generative latent variable model

  \[ g_i(x) = P(\omega_i | x, \eta) \]

- Likelihood of \( y \):
  \[ P(y | x, \Theta, \eta) = \sum_{u=1}^{k} P(\omega_u | x, \eta) p(y | x, \omega_u, \Theta) \]
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)

\[ \omega_u \]
\[ \omega_{uv} \]

Switching (gating) indicator

\[ x \]
\[ E1 \]
\[ E2 \]
\[ E3 \]
\[ E4 \]
\[ y \]

Hierarchical mixture model

An output is conditioned (gated) on multiple mixture levels

\[ P(y | x, \Theta) = \sum_u P(\omega_u | x, \eta) \sum_v P(\omega_{uv} | x, \omega_u, \xi_v) \sum_s P(\omega_{uv..s} | x, \omega_u, \omega_{uv}, \ldots) P(y | x, \omega_u, \omega_{uv}, \omega_{uv..s}, \Theta) \]

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

\[ P(\Omega_{uv..s} | x, \Theta) = P(\omega_u | x) P(\omega_{uv} | x, \omega_u) \ldots P(\omega_{uv..s} | x, \omega_u, \omega_{uv}, \ldots) \]

- **Then**

\[ P(y | x, \Theta) = \sum_u \sum_v \ldots \sum_s P(\Omega_{uv..s} | x, \Theta) P(y | x, \Omega_{uv..s}, \Theta) \]

- 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
  \[ g_u(x) = P(\omega_u | x, \Theta) \quad g_{vju}(x) = P(\omega_{uv} | x, \omega_u, \Theta) \]

• Multiple levels of responsibilities
  \[ h_u(x, y) = P(\omega_u | x, y, \Theta) \quad h_{vju}(x, y) = P(\omega_{uv} | x, y, \omega_u, \Theta) \]

• How they are related?

\[
P(\omega_{uv} | x, y, \omega_u, \Theta) = \frac{P(y | x, \omega_u, \omega_{uv}, \Theta) P(\omega_{uv} | x, \omega_u, \Theta)}{\sum_u P(y | x, \omega_u, \omega_{uv}, \Theta) P(\omega_{uv} | x, \omega_u, \Theta)} = \sum_y P(y, \omega_{uv} | x, \omega_u, \Theta) = P(y | x, \omega_u, \Theta)
\]

Hierarchical mixture of experts

• Responsibility for the top layer
  \[ h_u(x, y) = P(\omega_u | x, y, \Theta) = \frac{P(y | x, \omega_u, \Theta) P(\omega_u | x, \Theta)}{\sum_u P(y | x, \omega_u, \Theta) P(\omega_u | x, \Theta)} \]

• But \( P(y | x, \omega_u, \Theta) \) is computed while computing
  \[ h_{vju}(x, y) = P(\omega_{uv} | x, y, \omega_u, \Theta) \]

• General algorithm:
  – Downward sweep; calculate
    \[ g_{vju}(x) = P(\omega_{uv} | x, \omega_u, \Theta) \]
  – Upward sweep; calculate
    \[ h_u(x, y) = P(\omega_u | x, y, \Theta) \]
On-line learning

- Assume linear experts \( \mu_{uv} = \theta_{uv}^T x \)
- Gradients (vector form):
  \[
  \frac{\partial l}{\partial \theta_{uv}} = h_u h_{v|u} (y - \mu_{uv}) x
  \]
  \[
  \frac{\partial l}{\partial \eta} = (h_u - g_u) x \quad \text{Top level (root) node}
  \]
  \[
  \frac{\partial l}{\partial \xi} = h_u (h_{v|u} - g_{v|u}) x \quad \text{Second level node}
  \]
- Again: can it can be extended to different expert networks