Ensemble methods:
• Mixtures of experts
• Bagging

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 to improve predictions

- **Mixture of expert model:**
  - Different input regions covered with different learners
  - A “soft” switching between learners

- **Mixture of experts**
  Expert = learner

\[ X \]
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 network
    • Decides where to make a split
• **Assume:** gating functions give probabilities
  \[ 0 \leq g_1(x), g_2(x), \ldots, g_k(x) \leq 1 \]
  \[ \sum_{u=1}^{k} g_{u}(x) = 1 \]
• Based on the probability we partition the space
  – partitions belongs to different experts
• How to model the gating network?
  – **A multi-way 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 \( y \) (linear regression – assume errors for different experts are normally distributed with the same variance)

  \[
P(y \mid x, \Theta, \eta) = \sum_{i=1}^{k} P(\omega_i \mid x, \eta) p(y \mid x, \omega_i, \Theta) = \sum_{i=1}^{k} \left[ \frac{\exp(\eta_i^T x)}{\sum_{j=1}^{k} \exp(\eta_j^T x)} \right] \left[ \frac{1}{\sqrt{2\pi}\sigma} \exp\left( -\frac{\|y - \mu_i\|^2}{2\sigma^2} \right) \right]
  \]
Learning mixture of experts

Gradient learning.

**On-line update rule for parameters** \( \theta_i \) **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** \( i \text{th} \) **expert** = a kind of posterior

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

\( g_i(x) \) - a prior \quad \exp(...) - a likelihood
Learning mixtures of experts

Gradient methods

• On-line learning of gating network parameters $\eta_i$
  \[
  \eta_{ij} \leftarrow \eta_{ij} + \beta_{ij} (h_i(x, y) - g_i(x))x_j
  \]

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

2. **Maximization step**

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

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

   - 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
  \[ g_i(x) = P(\omega_i \mid x, \eta) \]
- Q function to optimize
  \[ Q(\Theta \mid \Theta') = E_{H \mid X, Y, \Theta} \log P(H, Y \mid X, \Theta, \xi) \]
- Assume:
  - \( l \) indexes different data points
  - \( \delta_i^l \) an indicator variable for the data point \( l \) to be covered by an expert \( i \)

\[ Q(\Theta \mid \Theta') = \sum_l \sum_i E(\delta_i^l \mid x^l, y^l, \Theta', \eta') \log(P(y^l, \omega_i \mid x^l, \Theta, \eta)) \]
Learning mixture of experts with EM

• **Assume:**
  
  – \( l \) indexes different data points
  – \( \delta_{li} \) an indicator variable for data point \( l \) and expert \( i \)

\[
Q(\Theta | \Theta') = \sum_l \sum_i E(\delta_{li} | x^l, y^l, \Theta', \eta') \log(P(y^l, \omega_i | x^l, \Theta, \eta))
\]

\[
E(\delta_{li} | x^l, y^l, \Theta', \eta') = h_i(x^l, y^l) = \frac{g_i(x^l) p(y | x^l, \omega_i, \Theta')}{\sum_{u=1}^{k} g_u(x^l) p(y^l | x^l, \omega_u, \Theta')}
\]

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

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

$$Q(\Theta | \Theta') = \sum_i \sum_l h_i^l(x^l, y^l) \log(P(y^l, \omega_i | x^l, \Theta, \eta))$$

$$\log(P(y^l, \omega_i | x^l, \Theta, \eta)) = \log P(y^l | \omega_i, x^l, \Theta) + \log P(\omega_i | x^l, \eta)$$

- 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)
Hierarchical mixture model

An output is conditioned (gated) on multiple mixture levels

\[ P(y \mid x, \Theta) = \sum_u P(\omega_u \mid x, \eta) \sum_v p(\omega_{uv} \mid x, \omega_u, \xi) \sum_s P(\omega_{uv..s} \mid x, \omega_u, \omega_{uv}, \ldots)P(y \mid x, \omega_u, \omega_{uv}, \ldots, \theta_{uv..s}) \]

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

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

- Then

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

- 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
  \[ g_u(x) = P(\omega_u \mid x, \Theta) \]  \[ g_{v \mid u}(x) = P(\omega_{uv} \mid x, \omega_u, \Theta) \]

- Multiple levels of responsibilities
  \[ h_u(x, y) = P(\omega_u \mid x, y, \Theta) \]  \[ h_{v \mid u}(x, y) = P(\omega_{uv} \mid x, y, \omega_u, \Theta) \]

- How they are related?

  responsibility
  \[
  P(\omega_{uv} \mid x, y, \omega_u, \Theta) = \frac{P(y \mid x, \omega_u, \omega_{uv}, \Theta)P(\omega_{uv} \mid x, \omega_u, \Theta)}{\sum_v P(y \mid x, \omega_u, \omega_{uv}, \Theta)P(\omega_{uv} \mid x, \omega_u, \Theta)}
  \]

  \[
  = \sum_v P(y, \omega_{uv} \mid x, \omega_u, \Theta) = P(y \mid x, \omega_u, \Theta)
  \]
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 \( h_{v|u}(x, y) = P(\omega_{uv} | x, y, \omega_u, \Theta) \)

- **General algorithm:**
  - Downward sweep; calculate \( g_{v|u}(x) = P(\omega_{uv} | x, \omega_u, \Theta) \)
  - 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
Ensemble methods

• **Mixture of experts**
  – Multiple ‘base’ models (classifiers, regressors), each covers a different part (region) of the input space

• **Committee machines:**
  – Multiple ‘base’ models (classifiers, regressors), each covers the complete input space
  – Each base model is trained on a slightly different train set
  – Combine predictions of all models to produce the output
    • **Goal:** Improve the accuracy of the ‘base’ model
    • **Methods:**
      • Bagging
      • Boosting
      • Stacking (not covered)
Bagging (Bootstrap Aggregating)

• **Given:**
  – Training set of $N$ examples
  – A class of learning models (e.g. decision trees, neural networks, …)

• **Method:**
  – Train multiple (k) models on different samples (data splits) and average their predictions
  – Predict (test) by averaging the results of k models

• **Goal:**
  – Improve the accuracy of one model by using its multiple copies
  – Average of misclassification errors on different data splits gives a better estimate of the predictive ability of a learning method
Bagging algorithm

• **Training**
  – In each iteration $t$, $t=1,\ldots,T$
    • Randomly sample with replacement $N$ samples from the training set
    • Train a chosen “base model” (e.g. neural network, decision tree) on the samples

• **Test**
  – For each test example
    • Start all trained base models
    • Predict by combining results of all $T$ trained models:
      – **Regression**: averaging
      – **Classification**: a majority vote
Simple Majority Voting

Test examples

- $H_1$
- $H_2$
- $H_3$
- Final

Class “yes”

Class “no”
Analysis of Bagging

• **Expected error** = **Bias** + **Variance**
  
  – *Expected error* is the expected discrepancy between the estimated and true function
    
    \[ E\left[ (\hat{f}(X) - E[f(X)])^2 \right] \]

  – *Bias* is squared discrepancy between averaged estimated and true function
    
    \[ \left( E[\hat{f}(X)] - E[f(X)] \right)^2 \]

  – *Variance* is expected divergence of the estimated function vs. its average value
    
    \[ E\left[ (\hat{f}(X) - E[\hat{f}(X)])^2 \right] \]
When Bagging works?
Under-fitting and over-fitting

- **Under-fitting:**
  - High bias (models are not accurate)
  - Small variance (smaller influence of examples in the training set)

- **Over-fitting:**
  - Small bias (models flexible enough to fit well to training data)
  - Large variance (models depend very much on the training set)
**Averaging decreases variance**

- **Example**
  - Assume we measure a random variable $x$ with a $N(\mu, \sigma^2)$ distribution
  - If only one measurement $x_1$ is done,
    - The expected mean of the measurement is $\mu$
    - Variance is $\text{Var}(x_1) = \sigma^2$
  - If random variable $x$ is measured $K$ times $(x_1, x_2, \ldots, x_K)$ and the value is estimated as: $(x_1 + x_2 + \ldots + x_K)/K$,
    - Mean of the estimate is still $\mu$
    - But, variance is smaller:
      - $[\text{Var}(x_1) + \ldots + \text{Var}(x_K)]/K^2 = K\sigma^2 / K^2 = \sigma^2 / K$
  - Observe: **Bagging is a kind of averaging!**
When Bagging works

• **Main property of Bagging** (proof omitted)
  – Bagging *decreases variance* of the base model without changing the bias!!!
  – Why? averaging!

• **Bagging typically helps**
  – When applied with an *over-fitted base model*
    • High dependency on actual training data

• **It does not help much**
  – High bias. When the base model is robust to the changes in the training data (due to sampling)