CS 2750 Machine Learning
Lecture 20

Learning with multiple models. Mixture of Experts. Bagging.

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

Learning with multiple models

We know how to build different classification or regression models from data

- **Question:**
  – Is it possible to learn and combine multiple (classification/regression) models and improve their predictive performance ?
- **Answer: yes**
- There are different ways of how to do it…

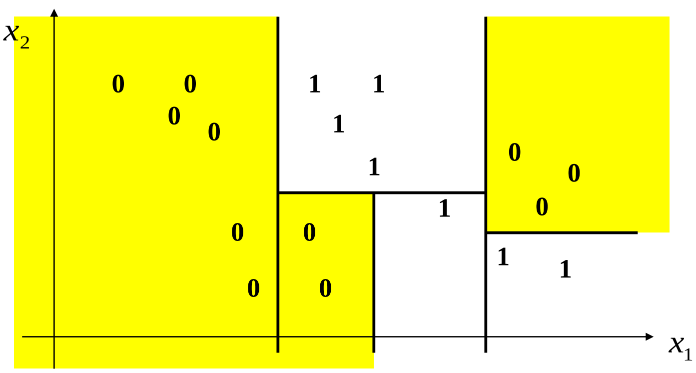
Learning with multiple models

- **Question:**
  - Is it possible to learn and combine multiple (classification/regression) models and improve their predictive performance ?
- There are different ways of how to do it…

- Assume you have models M1, M2, … Mk
- **Approach 1:** use different models (classifiers, regressors) to cover **the different parts of the input (x) space**
- **Approach 2:** use different models (classifiers, regressors) that cover **the complete input (x) space**, and combine their predictions

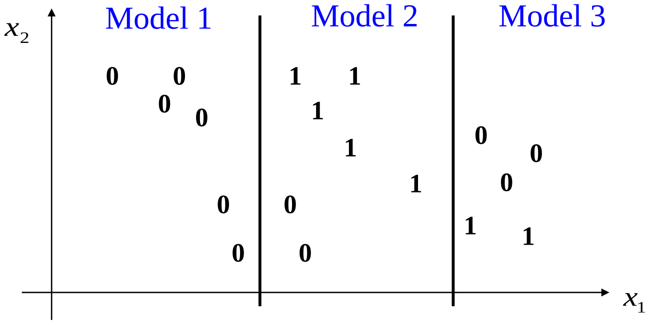
Approach 1

- Recall the decision tree:
  - **It partitions the input space to regions**
  - **It picks the class independently in every region**



Approach 1

- Recall the decision tree:
  - **It partitions the input space to regions**
  - **picks the class independently**
- **What if we define a more general partitions of the input space and learn a model specific to these partitions**

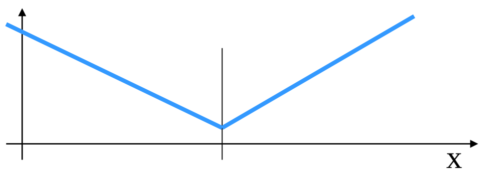


Learning with multiple models: Approach 1

Define a more general partitions of the input space and learn a model specific to these partitions

Example:

- 2 linear functions covering two regions of the input space



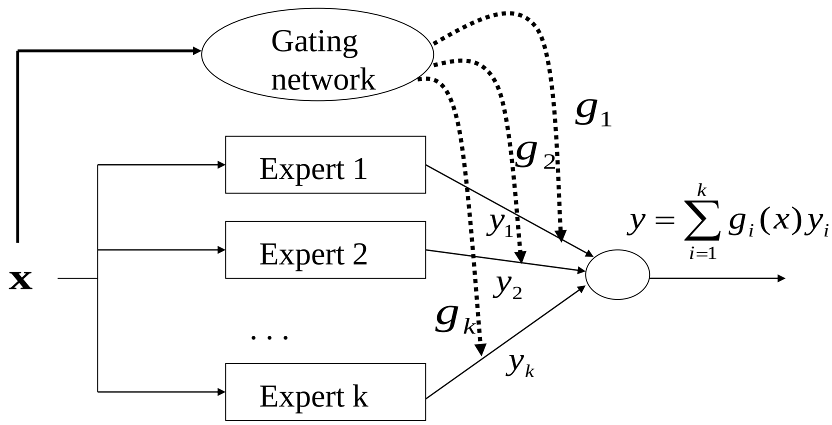
Mixture of expert model:

- Expert = learner (model)
- Different input regions covered with a different learner/model
- A "soft" switching between learners

Mixture of experts model

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

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



Mixture of experts model

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

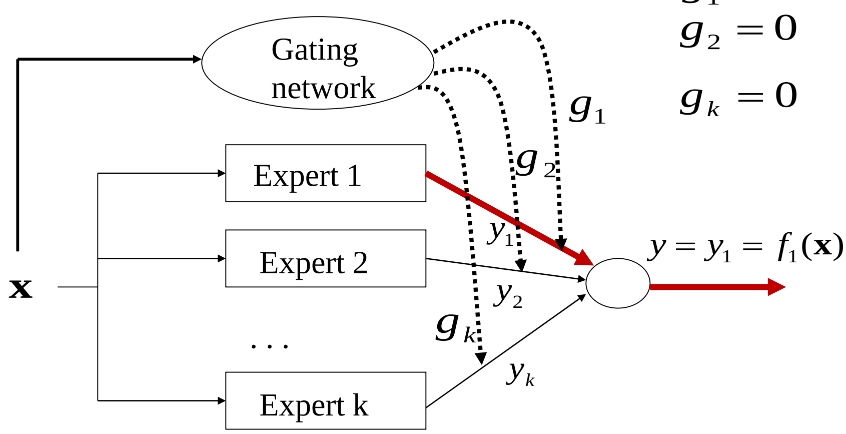
  $g_1, g_2, \ldots g_k$ - gating functions

Assume

$g_1 = 1$

$g_2 = 0$

$g_k = 0$



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

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

$$y = \sum_{u=1}^{k} g_u(\mathbf{x}) f_u(\mathbf{x})$$

- 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**

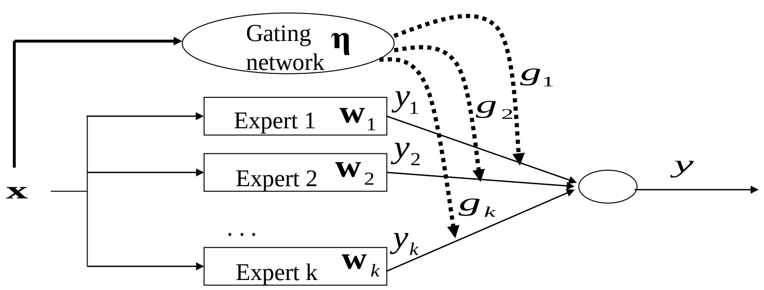
Learning mixture of experts

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

$$y_i = \mathbf{w}_i^T \mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0, \sigma)$$ (Note: bias terms are hidden in x)

- Assume a **softmax gating network**

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



Learning mixture of experts

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

$$y_i = \mathbf{w}_i^T \mathbf{x} + \varepsilon \qquad \varepsilon \sim N(0,\sigma)$$ (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)

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

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

Learning mixture of experts

Learning of parameters of expert models:

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

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

$$w_{ij} \leftarrow w_{ij} + \alpha_{ij} (y - \mathbf{w}_i^T \mathbf{x}) x_j$$

– If we do not know the expert

$$\boxed{w_{ij} \leftarrow w_{ij} + \alpha_{ij} h_i (y - \mathbf{w}_i^T \mathbf{x}) x_j}$$

h_i - responsibility of the ith expert = a kind of posterior

$$h_i(\mathbf{x}, y) = \frac{g_i(\mathbf{x})\, p(y \mid \mathbf{x}, \omega_i, \mathbf{W})}{\sum_{u=1}^{k} g_u(\mathbf{x})\, p(y \mid \mathbf{x}, \omega_u, \mathbf{W})} = \frac{g_i(\mathbf{x}) \exp\left(-1/2 \left\| y - \mathbf{w}_i^T \mathbf{x} \right\|^2 \right)}{\sum_{u=1}^{k} g_u(\mathbf{x}) \exp\left(-1/2 \left\| y - \mathbf{w}_u^T \mathbf{x} \right\|^2 \right)}$$

$g_i(\mathbf{x})$ - a prior $\qquad$ $\exp(\ldots)$ - 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 with multiple models: Approach 2

- **Approach 2:** use multiple models (classifiers, regressors) that cover the complete input (x) space and combines their outputs

- **Committee machines:**
  - Combine predictions of all models to produce the output
    - **Regression:** averaging
    - **Classification:** a majority vote
  - **Goal:** Improve the accuracy of the 'base' model

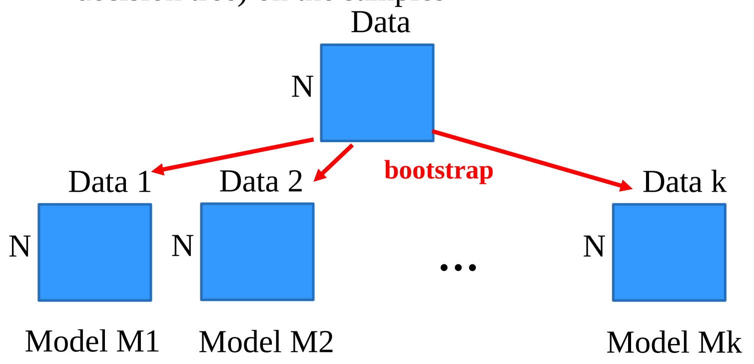
- **Methods:**
  - **Bagging (the same base models)**
  - **Boosting (the same base models)**
  - Stacking (different base model) not covered

Bagging (Bootstrap Aggregating)

- **Given:**
  - Training set of *N* examples
  - A base learning model (e.g. decision tree, neural network, …)
- **Method:**
  - Train multiple (k) base models on slightly different datasets
  - 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

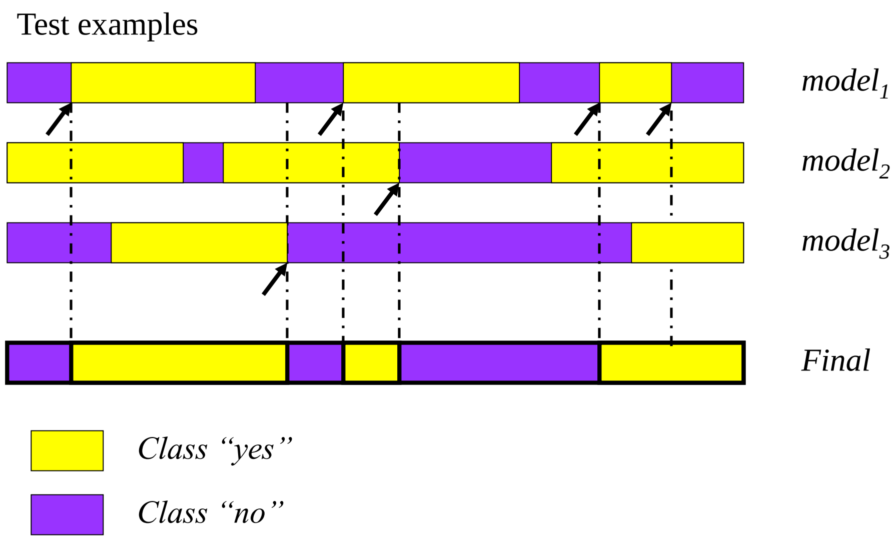
- **Training**
- For each model M1, M2, … Mk
  - Randomly sample with replacement *N* samples from the training set (bootstrap)
  - Train a chosen "base model" (e.g. neural network, decision tree) on the samples



Bagging algorithm

- **Training**
- For each model M1, M2, … Mk
    - 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
        - Run all base models M1, M2, … Mk
        - Predict by combining results of all T trained models:
            - **Regression:** averaging
            - **Classification:** a majority vote

Class decision via majority voting



Analysis of Bagging

- **Expected error= Bias+Variance**
  - *Expected error* is the expected discrepancy between the estimated and true function

$$E\left[\left(\hat{f}(X)-E[f(X)]\right)^2\right]$$

  - *Bias* is a squared discrepancy between *averaged* estimated and true function

$$\left(E\left[\hat{f}(X)\right]-E[f(X)]\right)^2$$

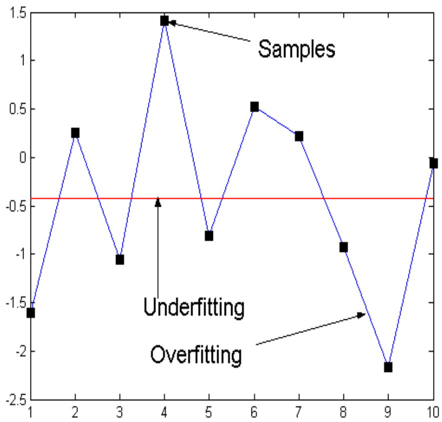
  - *Variance* is an expected divergence of the estimated function vs. its average value

$$E\left[\left(\hat{f}(X)-E\left[\hat{f}(X)\right]\right)^2\right]$$

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 a random variable x with a $N(\mu,\sigma^2)$ distribution
  - **Case 1:** we draw one example/measurement x_1 and use it to estimate the mean $\mu' = x_1$
    - The expected mean of the estimate is μ
    - The variance of the mean estimate is $Var(x_1)=\sigma^2$
  - **Case 2:** a variable x is measured K times $(x_1,x_2,\ldots x_k)$ and the mean is estimated as: $\mu'= (x_1+x_2+\ldots+x_k)/K$,
    - The expected mean of the estimate is still μ
    - But, the variance of the mean estimate is smaller:
      - $[Var(x_1)+\ldots Var(x_k)]/K^2=K\sigma^2 / K^2 = \sigma^2/K$
- Relation to bagging: **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
    - Example: fully grown decision trees
- **It does not help much**
  - High bias. When the base model is robust to the changes in the training data (due to sampling)