Combining Classifiers

- Problem Definition
  - Given
    - Training data set \( D \) for supervised learning
    - \( D \) drawn from common instance space \( X \)
    - Collection of inductive learning algorithms, hypothesis languages (inducers)
  - Hypotheses produced by applying inducers to \( a(D) \)
  - \( a: X \to X \) vector (sampling, transformation, partitioning, etc.)
  - Can think of hypotheses as definitions of prediction algorithms ("classifiers")
  - Return: new prediction algorithm (not necessarily \( a \)) for \( x: X \) that combines outputs from collection of prediction algorithms
- Desired Properties
  - Guarantees of performance of combined prediction
  - e.g., mistake bounds: ability to improve weak classifiers
- Two Solution Approaches
  - Train and apply each inducer; learn combiner function(s) from result
  - Train inducers and combiner function(s) concurrently

Framework: Data Fusion and Mixtures of Experts

- What Is A Weak Classifier?
  - One not guaranteed to do better than random guessing (1/number of classes)
  - Goal: combine multiple weak classifiers, get one at least as accurate as strongest
- Data Fusion
  - Intuitive idea
    - Multiple sources of data (sensors, domain experts, etc.)
    - Need to combine systematically, plausibly
    - Solution approaches
      - Control of intelligent agents: Kalman filtering
      - General: mixture estimation (sources of data = predictions to be combined)
  - Mixtures of Experts
    - Intuitive idea: "experts" express hypotheses (drawn from a hypothesis space)
    - Solution approach (next time)
      - Mixture model: estimate mixing coefficients
      - Hierarchical mixture models: divide-and-conquer estimation method

Combining Classifiers: Ensemble Averaging

- Intuitive idea
  - Combine experts (aka prediction algorithms, classifiers) using combiner function
    - Combiner may be weight vector (WM), vote (bagging), trained inducer (stacking)
- Weighted Majority (WM)
  - Weights each hypothesis in proportion to its training set accuracy
  - Use this weight in performance element (and on test set predictions)
  - Mistake bound for WM
- Bootstrap Aggregating (Bagging)
  - Voting system for collection of algorithms
    - Training set for each member: sampled with replacement
    - Works for unstable inducers (search for \( h \) sensitive to perturbation in \( D \))
- Stacked Generalization (aka Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
    - Training sets for "leaves": sampled with replacement; combiner: validation set
  - Single-Pass: Train Classification and Combiner Inducers Serially
  - Static Structures: Ignore Input Signal

Problem: Improving Weak Classifiers

- First Classifier
- Second Classifier
- Both Classifiers

Mixture Model

Principle:
Improving Weak Classifiers

Outside Reading
- Section 7.5, Mitchell
- Section 5, MLC++ manual, Kohavi and Sommerfield
- Lectures 21-22, CIS 798 (Fall, 1999)
- This Week’s Paper Review: “Bagging, Boosting, and C4.5”, J. R. Quinlan
- Combining Classifiers
  - Problem definition and motivation: improving accuracy in concept learning
  - General framework: collection of weak classifiers to be improved
- Examples of Combiners (Committee Machines)
  - Weighted Majority (WM), Bootstrap Aggregating (Bagging), Stacked Generalization (Stacking), Boosting the Margin
  - Mixtures of experts, Hierarchical Mixtures of Experts (HME)
- Committee Machines
  - Static structures: Ignore input signal
  - Dynamic structures (multi-pass): use input signal to improve classifiers

Combining Classifiers

- Intuitive idea
  - Combine experts (aka prediction algorithms, classifiers) using combiner function
    - Combiner may be weight vector (WM), vote (bagging), trained inducer (stacking)
- Weighted Majority (WM)
  - Weights each hypothesis in proportion to its training set accuracy
  - Use this weight in performance element (and on test set predictions)
  - Mistake bound for WM
- Bootstrap Aggregating (Bagging)
  - Voting system for collection of algorithms
    - Training set for each member: sampled with replacement
    - Works for unstable inducers (search for \( h \) sensitive to perturbation in \( D \))
- Stacked Generalization (aka Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
    - Training sets for "leaves": sampled with replacement; combiner: validation set
  - Single-Pass: Train Classification and Combiner Inducers Serially
  - Static Structures: Ignore Input Signal

Combining Classifiers

- Intuitive idea
  - Combine experts (aka prediction algorithms, classifiers) using combiner function
    - Combiner may be weight vector (WM), vote (bagging), trained inducer (stacking)
- Weighted Majority (WM)
  - Weights each hypothesis in proportion to its training set accuracy
  - Use this weight in performance element (and on test set predictions)
  - Mistake bound for WM
- Bootstrap Aggregating (Bagging)
  - Voting system for collection of algorithms
    - Training set for each member: sampled with replacement
    - Works for unstable inducers (search for \( h \) sensitive to perturbation in \( D \))
- Stacked Generalization (aka Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
    - Training sets for "leaves": sampled with replacement; combiner: validation set
  - Single-Pass: Train Classification and Combiner Inducers Serially
  - Static Structures: Ignore Input Signal
Weighted Majority:

- **Idea**
  - **Weight-Based Combiner**
    - Weighted votes: each prediction algorithm (classifier) $h_i$ maps from $x \to h_i(x)$
    - Resulting in set of legal class labels
    - NB: as for Bayes Optimal Classifier, resulting predictor not necessarily in $H$
  
- **Intuitive Idea**
  - Collect votes from pool of prediction algorithms for each training example
  - Decrease weight associated with each algorithm that guessed wrong (by a multiplicative factor)
  - Combiner predicts weighted majority label

- **Performance Goals**
  - Improving training set accuracy
  - Want to combine weak classifiers
  - Want to bound number of mistakes in terms of minimum made by any one algorithm
  - Hope that this results in good generalization quality

---

Bagging:

- **Idea**
  - **Bootstrap Aggregating aka Bagging**
    - Application of bootstrap sampling
      - Given: set $D$ containing $m$ training examples
      - Create $\mathcal{S}(i)$ by drawing $m$ examples at random with replacement from $D$
      - $\mathcal{S}(i)$ of size $m$: expected to leave out $0.37/m$ of examples from $D$
    - **Bagging**
      - Create a bootstrap samples $\mathcal{S}(1), \mathcal{S}(2), \ldots, \mathcal{S}(k)$
      - Train distinct inducers $h_i$ to produce $k$ classifiers
      - Classify new instance by classifier vote (equal weights)
  
- **Intuitive Idea**
  - “Two heads are better than one”
  - Produce multiple classifiers from one data set
  - NB: same inducer (multiple instantiations) or different inducers may be used
  - Differences in samples will “smooth out” sensitivity of $L$, $H$, $O$

---

Stacked Generalization:

- **Idea**
  - **Stacked Generalization aka Stacking**
    - Train multiple learners
      - Each uses subsample of $D$
      - May be ANN, decision tree, etc.
    - Train combiner on validation segment
    - See [Wolpert, 1992; Bishop, 1995]

- **Intuitive Idea**
  - Train inducers and combiner function(s) concurrently
  - Learn how to divide and balance learning problem across multiple inducers
  - Framework: mixture estimation

---

Single Pass Combiners

- **Combining Classifiers**
  - Problem definition and motivation: improving accuracy in concept learning
  - General framework: collection of weak classifiers to be improved (data fusion)
- **Weighted Majority**
  - Weighting system for collection of algorithms
  - Weights each algorithm in proportion to its training set accuracy
  - Use this weight in performance element (and on test set predictions)
  - Mistake bound for WM
- **Bootstrap Aggregating** (Bagging)
  - Voting system for collection of algorithms
  - Training set for each member: sampled with replacement
  - Works for unstable inducers
- **Stacked Generalization** (aka Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
  - Training sets for “leaves”: sampled with replacement; combiner: validation set
- **Next:** Boosting the Margin, Hierarchical Mixtures of Experts

---

Boosting:

- **Idea**
  - **Intuitive Idea**
    - Another type of static committee machine can be used to improve any inducer
    - Learn set of classifiers from D, but reweight examples to emphasize misclassified
  - **Final classifier:** weighted combination of classifiers
  - Different from Ensemble Averaging
    - WM: all inducers trained on same $D$
    - Bagging, stacking: training/validation partitions, i.e., subsamples $\mathcal{S}(i)$ of $D$
  - **Boosting:** data sampled according to different distributions
  - **Problem Definition**
    - Given: collection of multiple inducers, large data set or example stream
  - **Solution Approaches**
    - Filtering: use weak inducers in cascade to filter examples for downstream ones
      - Resampling: reuse data from $D$ by subsampling (don’t need huge or “infinite” $D$
      - Reweighting: reuse $D$, but measure error over weighted $x$
Mixture Models: Procedure

- Algorithm Combiner-Mixture-Model (D, L, Activation, k)
  - m = D.size
  - FOR j = 1 TO k DO // initialization
    w[j] = 1
  - UNTIL the termination condition is met, DO
    - FOR j = 1 TO k DO
      - P[j] = L[j].Update-Inducer (D) // single training step for L[j]
  - FOR i = 1 TO m DO
    Sum[i] = 0
  - FOR j = 1 TO k DO
    Sum[i] += P[j](D[i])
  - Net[i] = Compute-Activation (Sum[i]) // compute g[i] = Net[i]
  - FOR i = 1 TO m DO
    w[i] = Update-Weights (w[i], Net[i], D[i])
  - RETURN (Make-Predictor (P, w))

- Algorithm Combiner-HME (D, L, Activation, Level, k, Classes)
  - m = D.size
  - FOR j = 1 TO k DO // initialization
    w[j] = 1
  - UNTIL the termination condition is met DO
    - IF Level > 1 THEN
      - FOR j = 1 TO k DO
        - P[j] = Combiner-HME (D, L[j], Level - 1, k, Classes)
    - ELSE
      - FOR j = 1 TO k DO
        - P[j] = L[j].Update-Inducer (D)
    - FOR i = 1 TO m DO
      Sum[i] = 0
    - FOR j = 1 TO m DO
      Sum[i] += P[j](D[i])
    - Net[i] = Compute-Activation (Sum[i]) // compute g[i] = Net[i]
    - FOR i = 1 TO Classes DO
      w[i] = Update-Weights (w[i], Net[i], D[i])
    - RETURN (Make-Predictor (P, w))

Mixture Models: Idea

- Intuitive Idea
  - Integrate knowledge from multiple experts (or data from multiple sensors)
  - Collection of inducers organized into committee machine (e.g., modular ANN)
  - Dynamic structure: take input signal into account

- References
  - [Bishop, 1995] (Sections 2.7, 9.7)
  - [Haykin, 1999] (Chapter 7.6)

- Problem Definition
  - Given: collection of inducers ("experts") D, data set X
  - Perform: supervised learning using inducers and self-organization of experts
  - Return: committee machine with trained gating network (combiner inducer)

- Solution Approach
  - Let combiner inducer be generalized linear model (e.g., threshold gate)
  - Activation functions: linear combination, vote, "smoothed" vote (softmax)

- Possible Modifications
  - † Batch (as opposed to online) updates: lift Update-Weights out of outer FOR loop
  - † Classification learning (versus concept learning): multiple y values
  - † Arrange gating networks (combiner inducers) in hierarchy (HME)

- Algorithm Combiner-Mixture-Model (D, L, Activation, k)
  - m = D.size
  - FOR j = 1 TO k DO // initialization
  - w[j] = 1
  - UNTIL the termination condition is met, DO
    - FOR j = 1 TO k DO
      - P[j] = L[j].Update-Inducer (D) // single training step for L[j]
    - FOR i = 1 TO m DO
      Sum[i] = 0
    - FOR j = 1 TO k DO
      Sum[i] += P[j](D[i])
    - Net[i] = Compute-Activation (Sum[i]) // compute g[i] = Net[i]
    - FOR i = 1 TO m DO
      w[i] = Update-Weights (w[i], Net[i], D[i])
    - RETURN (Make-Predictor (P, w))
Hierarchical Mixture of Experts (HME): Properties

- **Advantages**
  - Benefits of ME: base case is single level of expert and gating networks
  - More combiner inducers ⇒ more capability to decompose complex problems

- **Views of HME**
  - Expresses divide-and-conquer strategy
  - Problem is distributed across subtrees "on the fly" by combiner inducers
  - Quality: data fusion ⇒ problem redistribution
  - Recursive decomposition: until good fit found to "local" structure of $D$
  - Implementation: a-level decision tree
  - Mixture of experts: 1-level decision tree (decision stump)
  - Information preservation compared to traditional (hard) decision tree
  - Dynamics of HME improves on greedy (high-commitment) strategy of decision tree induction

Training Methods for Hierarchical Mixture of Experts (HME)

- **Stochastic Gradient Ascent**
  - Maximizes log-likelihood function $L(\theta) = \log P(D | \theta)$
  - Compute
    $$\frac{\partial L}{\partial w_i} = \frac{1}{n} \sum_{j=1}^{n} \left[ y_j \cdot h(x_j; \theta) - \hat{y}_j \right]$$
  - Finds MAP values
    - Expert network (leaf) weights $w_2$
    - Gating network (interior node) weights $\alpha_j$ at lower level ($a_j$), upper level ($a_{ij}$)

- **Expectation-Maximization (EM) Algorithm**
  - Recall definition
    - Goal: maximize incomplete-data log-likelihood function $L(\theta) = \log P(D | \theta)$
  - Estimation step: calculate $E$[unobserved variables] $\hat{\theta}$, assuming current $\theta$
  - Maximization step: update $\theta$ to maximize $L(\theta | \hat{\theta})$, $D$ or all variables

Methods for Combining Classifiers: Committee Machines

- **Framework**
  - Think of collection of trained inducers as committee of experts
  - Each produces predictions given input $(x(D_{train}), x_{new}, x)$

- **Types of Committee Machines**
  - Static structures: based only on $x$ coming out of local inducers
    - Single-pass, same data or independent subsamples: WM, bagging, stacking
    - Cascade training: AdaBoost
    - Iterative reweighting: boosting by reweighting
  - Dynamic structures: take $x$ into account
    - Mixture models (mixtures of experts aka ME): one combiner (gating) level
    - Hierarchical Mixture of Experts (HME): multiple combiner (gating) levels
    - Specialist-B catalyst (SB) networks: partitions of $x$ given to combiners

Terminology [1]: Single-Pass Combiners

- **Combining Classifiers**
  - Weak classifiers: not guaranteed to do better than random guessing
    - Combining functions: $f$ prediction vector $\rightarrow$ instance $\rightarrow$ prediction

- **Single-Pass Combiners**
  - Weighted Majority (WM)
    - Weights prediction of each inducer according to its training-set accuracy
    - Mistake bound: maximum number of mistakes before converging to correct $h$
    - Incrementality: ability to update parameters without complete retraining
  - Bootstrap Aggregating (aka Bagging)
    - Takes vote among multiple inducers trained on different samples of $D$
  - Subsampling: drawing one sample from another ($D = \tilde{D}$)
  - Unstable inducer: small change to $D$ causes large change in $h$
  - Stacked Generalization (aka Stacking)
    - Hierarchical combiner can apply recursively to re-stack
  - Trains combiner inducer using validation set

Terminology [2]: Static and Dynamic Mixtures

- **Committee Machines aka Combiners**
  - Ensemble averaging
    - Single-pass, separately trained inducers, common input
    - Individual outputs combined to get scalar output (e.g., linear combination)
  - Boosting the margin: separately trained inducers, different input distributions
    - Filtering: feed examples to trained inducer (weak classifiers), pass on to next classifier if conflict encountered (ensemble model)
    - Resampling: aka subsampling (drawing of fixed size $m$) resampled from $D$
    - Reweighting: fixed size $S_i$ containing weighted examples for inducer

- **Dynamic Structures**
  - Mixture of experts: training in combiner inducer (aka gating network)
  - Hierarchical mixtures of experts: hierarchy of inducers, combiners

- **Mixture Models aka Mixture of Experts (ME)**
  - Expert (classification) gating (combiner) inducers (modules, "networks")
  - Hierarchical Mixture of Experts (HME): multiple combiner (gating) levels

Summary Points

- **Committee Machines aka Combiners**
  - Ensemble averaging
    - For improving weak (esp. unstable) classifiers
    - E.g., weighted majority, bagging, stacking
  - Boosting the margin
    - Improve performance of any inducer: weight examples to emphasize errors
    - Variants: filtering (aka consensus), reweighting (aka subsampling), reweighting

- **Dynamic Structures (Multi-Pass)**
  - Mixture of experts: training in combiner inducer (aka gating network)
  - Hierarchical mixtures of experts: hierarchy of inducers, combiners

- **Mixture Model (aka Mixture of Experts)**
  - Estimation of mixture coefficients (i.e., weights)
  - Hierarchical Mixtures of Experts (HME): multiple combiner (gating) levels

Next Topic: Reasoning under Uncertainty (Probabilistic KDD)