Lecture 16

Artificial Neural Networks Discussion (4 of 4): Modularity in Neural Learning Systems

Monday, February 28, 2000

William H. Hsu
Department of Computing and Information Sciences, KSU
http://www.cis.ksu.edu/~bhsu

Readings:
"Modular and Hierarchical Learning Systems", M. I. Jordan and R. Jacobs
(Reference) Section 7,5, Mitchell
(Reference) Lectures 21-22, CIS 798 (Fall, 1999)

Lecture Outline

- 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*

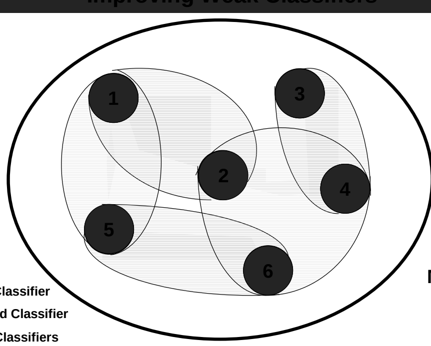
Combining Classifiers

- 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 *s*(*D*)
    - *s*: *X* vector $\rightarrow$ *X'* vector (sampling, transformation, partitioning, etc.)
    - Can think of hypotheses as definitions of prediction algorithms ("classifiers")
  - Return: new prediction algorithm (*not* necessarily $\in$ *H*) for $x \in X$ 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

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 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 (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

Principle: Improving Weak Classifiers



First Classifier
Second Classifier
Both Classifiers

Mixture Model

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 $\Rightarrow$ 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*

Weighted Majority: Idea

- Weight-Based Combiner
  - Weighted votes: each prediction algorithm (classifier) h_i maps from $x \in X$ to $h_i(x)$
  - Resulting prediction 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

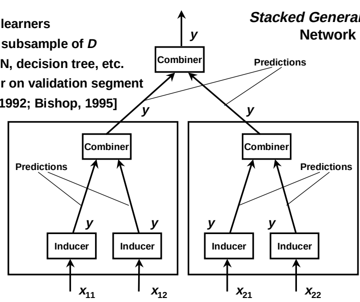
Bagging: Idea

- Bootstrap Aggregating *aka* Bagging
  - Application of bootstrap sampling
    - Given: set *D* containing *m* training examples
    - Create *S*[*i*] by drawing *m* examples at random *with replacement* from *D*
    - *S*[*i*] of size *m*: expected to leave out 0.37 of examples from *D*
  - Bagging
    - Create *k* bootstrap samples *S*[1], *S*[2], …, *S*[*k*]
    - Train distinct inducer on each *S*[*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* to *D*

Stacked Generalization: Idea

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

Stacked Generalization **Network**



Other Combiners

- So Far: Single-Pass Combiners
  - First, train each inducer
  - Then, train combiner on their output and evaluate based on criterion
    - Weighted majority: training set accuracy
    - Bagging: training set accuracy
    - Stacking: validation set accuracy
  - Finally, apply combiner function to get new prediction algorithm (classfier)
    - Weighted majority: weight coefficients (penalized based on mistakes)
    - Bagging: voting committee of classifiers
    - Stacking: validated hierarchy of classifiers with trained combiner inducer
- Next: Multi-Pass Combiners
  - 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 (WM)
  - 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 $\leftarrow$ weighted combination of classifiers
- Different from Ensemble Averaging
  - WM: all inducers trained on same *D*
  - Bagging, stacking: training/validation partitions, i.i.d. *subsamples* *S*[*i*] of *D*
  - Boosting: data sampled according to *different distributions*
- Problem Definition
  - Given: collection of multiple inducers, large data set or example stream
  - Return: combined predictor (trained committee machine)
- 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 $x \in D$, but measure error over weighted *x*

Mixture Models: Idea

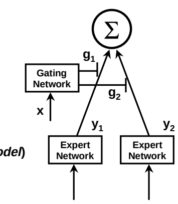
- Intuitive Idea
  - Integrate knowledge from multiple experts (or data from multiple sensors)
    - Collection of inducers organized into committee machine (e.g., modular ANN)
    - Dynamic structure: *take input signal into account*
  - References
    - [Bishop, 1995] (Sections 2.7, 9.7)
    - [Haykin, 1999] (Section 7.6)
- Problem Definition
  - Given: collection of inducers (“experts”) *L*, data set *D*
  - 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)

Mixture Models: Procedure

- Algorithm *Combiner-Mixture-Model* (*D*, *L*, *Activation*, *k*)
  - $m \leftarrow$ *D.size*
  - FOR $j \leftarrow$ 1 TO *k* DO // initialization
    - *w*[*j*] $\leftarrow$ 1
  - UNTIL the termination condition is met, DO
    - FOR $j \leftarrow$ 1 TO *k* DO
      - *P*[*j*] $\leftarrow$ *L*[*j*].*Update-Inducer* (*D*) // single training step for *L*[*j*]
    - FOR $i \leftarrow$ 1 TO *m* DO
      - *Sum*[*i*] $\leftarrow$ 0
      - FOR $j \leftarrow$ 1 TO *k* DO *Sum*[*i*] += P[*j*](*D*[*i*])
      - *Net*[*i*] $\leftarrow$ *Compute-Activation* (*Sum*[*i*]) // compute $g_j \equiv$ *Net*[*i*][*j*]
      - FOR $j \leftarrow$ 1 TO *k* DO *w*[*j*] $\leftarrow$ *Update-Weights* (*w*[*j*], *Net*[*i*], *D*[*i*])
  - RETURN (*Make-Predictor* (*P*, *w*))
- *Update-Weights*: Single Training Step for Mixing Coefficients

Mixture Models: Properties

- Unspecified Functions
  - *Update-Inducer*
    - Single training step for each expert module
    - e.g., ANN: one backprop cycle, *aka* epoch
  - *Compute-Activation*
    - Depends on ME architecture
    - Idea: smoothing of “winner-take-all” (“hard” max)
    - Softmax activation function (*Gaussian mixture model*)

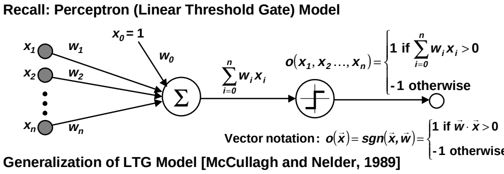


$$g_i = \frac{e^{\vec{w}_i \cdot \vec{x}}}{\sum_{j=1}^{k} e^{\vec{w}_j \cdot \vec{x}}}$$

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

Generalized Linear Models (GLIMs)

- Recall: Perceptron (Linear Threshold Gate) Model

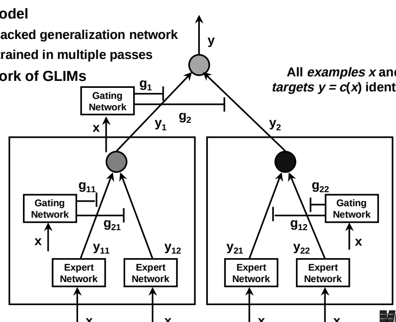


- Generalization of LTG Model [McCullagh and Nelder, 1989]
  - Model parameters: connection weights as for LTG
  - Representational power: depends on transfer (activation) function
- Activation Function
  - Type of mixture model depends (in part) on this definition
  - e.g., *o*(*x*) could be *softmax* (*x* · *w*) [Bridle, 1990]
    - *NB*: *softmax* is computed across *j* = 1, 2, …, *k* (cf. “hard” max)
    - Defines (*multinomial*) pdf over experts [Jordan and Jacobs, 1995]

Hierarchical Mixture of Experts (HME): Idea

- Hierarchical Model
  - Compare: stacked generalization network
  - Difference: trained in multiple passes
- Dynamic Network of GLIMs

All *examples x* and *targets y* = *c*(*x*) identical



Hierarchical Mixture of Experts (HME): Procedure

- Algorithm *Combiner-HME* (*D*, *L*, *Activation*, *Level*, *k*, *Classes*)
  - $m \leftarrow$ *D.size*
  - FOR $j \leftarrow$ 1 TO *k* DO *w*[*j*] $\leftarrow$ 1 // initialization
  - UNTIL the termination condition is met DO
    - IF *Level* > 1 THEN
      - FOR $j \leftarrow$ 1 TO *k* DO
        - *P*[*j*] $\leftarrow$ *Combiner-HME* (*D*, *L*[*j*], *Activation*, *Level* - 1, *k*, *Classes*)
    - ELSE
      - FOR $j \leftarrow$ 1 TO *k* DO *P*[*j*] $\leftarrow$ *L*[*j*].*Update-Inducer* (*D*)
    - FOR $i \leftarrow$ 1 TO *m* DO
      - *Sum*[*i*] $\leftarrow$ 0
      - FOR $j \leftarrow$ 1 TO *k* DO
        - *Sum*[*i*] += P[*j*](*D*[*i*])
      - *Net*[*i*] $\leftarrow$ *Compute-Activation* (*Sum*[*i*])
      - FOR $l \leftarrow$ 1 TO *Classes* DO *w*[*l*] $\leftarrow$ *Update-Weights* (*w*[*l*], *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
    - Duality: data fusion ⇔ problem redistribution
    - Recursive decomposition: until good fit found to “local” structure of *D*
  - Implements soft 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
  - Maximize log-likelihood function $L(\Theta) = \lg P(D \mid \Theta)$
  - Compute

$$\frac{\partial L}{\partial w_{ij}}, \frac{\partial L}{\partial a_j}, \frac{\partial L}{\partial a_{ij}}$$

  - Finds MAP values
    - Expert network (leaf) weights w_{ij}
    - Gating network (interior node) weights at lower level (a_{ij}), upper level (a_j)
- Expectation-Maximization (EM) Algorithm
  - Recall definition
    - Goal: maximize incomplete-data log-likelihood function $L(\Theta) = \lg P(D \mid \Theta)$
    - Estimation step: calculate E[*unobserved variables* $\mid \Theta$], assuming current Θ
    - Maximization step: update Θ to maximize $E[\lg P(D \mid \Theta)]$, $D \equiv$ *all variables*
  - Using EM: estimate with gating networks, then adjust $\Theta \equiv \{w_{ij}, a_{ij}, a_j\}$

Methods for Combining Classifiers: Committee Machines

- Framework
  - Think of collection of trained inducers as *committee of experts*
  - Each produces predictions given input ($s(D_{test})$, i.e., new x)
  - Objective: combine predictions by vote (subsampled D_{train}), learned weighting function, or more complex combiner inducer (trained using D_{train} or $D_{validation}$)
- Types of Committee Machines
  - Static structures: based only on *y* 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 (mixture of experts *aka* ME): one combiner (gating) level
    - Hierarchical Mixtures of Experts (HME): multiple combiner (gating) levels
    - Specialist-Moderator (SM) networks: partitions of *x* given to combiners

Terminology [1]: Single-Pass Combiners

- Combining Classifiers
  - Weak classifiers: not guaranteed to do better than random guessing
  - Combiners: functions *f*: *prediction vector* × *instance* → *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 ($D \sim \mathcal{D}$)
    - 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
- Static Structures
  - 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 inducers (weak classifiers), pass on to next classifier *iff* conflict encountered (consensus model)
    - Resampling: *aka* subsampling ($S[i]$ 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 Model, *aka* Mixture of Experts (ME)
  - Expert (classification), gating (combiner) inducers (modules, “networks”)
  - Hierarchical Mixtures of Experts (HME): multiple combiner (gating) levels

Summary Points

- Committee Machines *aka* Combiners
- Static Structures (Single-Pass)
  - Ensemble averaging
    - For improving weak (especially 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), resampling (*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)