Lecture 21

Combining Classifiers: Weighted Majority, Bagging, and Stacking

Tuesday, November 9, 1999

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

Readings:
Section 7.5, Mitchell
"Bagging, Boosting, and *C4.5*", Quinlan
Section 5, "*MLC++* Utilities 2.0", Kohavi and Sommerfield

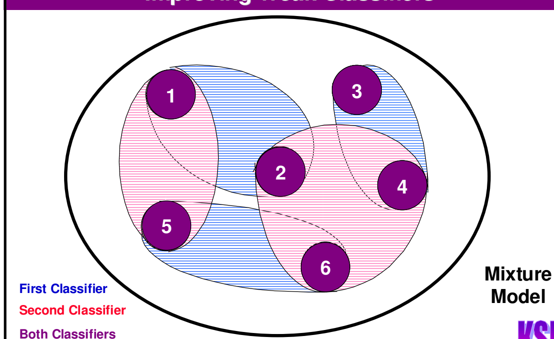
Lecture Outline

- **Readings**
  - Section 7.5, Mitchell
  - Section 5, *MLC++* manual, Kohavi and Sommerfield
- **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
- **Weighted Majority (WM)**
  - Weighting system for collection of algorithms
  - "Trusting" each algorithm *in proportion to its training set accuracy*
  - Mistake bound for WM
- **Bootstrap Aggregating (Bagging)**
  - Voting system for collection of algorithms (trained on subsamples)
  - When to expect bagging to work (unstable learners)
- **Next Lecture: Boosting the Margin, Hierarchical Mixtures of Experts**

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

Principle: Improving Weak Classifiers



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

Weighted Majority: Procedure

- Algorithm *Combiner-Weighted-Majority* (D, L)
  - $n \leftarrow L.size$ // number of inducers in pool
  - $m \leftarrow D.size$ // number of examples $<x \equiv D[j], c(x)>$
  - FOR $i \leftarrow 1$ TO n DO
    - $P[i] \leftarrow L[i].Train\text{-}Inducer\ (D)$ // $P[i]$: ith prediction algorithm
    - $w_i \leftarrow 1$ // initial weight
  - FOR $j \leftarrow 1$ TO m DO // compute WM label
    - $q_0 \leftarrow 0$, $q_1 \leftarrow 0$
    - FOR $i \leftarrow 1$ TO n DO

      IF $P[i](D[j]) = 0$ THEN $q_0 \leftarrow q_0 + w_i$ // vote for 0 (-)

      IF $P[i](D[j]) = 1$ THEN $q_1 \leftarrow q_1 + w_i$ // else vote for 1 (+)

      $Prediction[i][j] \leftarrow (q_0 > q_1)$? 0 : (($q_0 = q_1$) ? *Random* (0, 1): 1)

      IF $Prediction[i][j] \neq D[j].target$ THEN // $c(x) \equiv D[j].target$

      $w_i \leftarrow \beta w_i$ // $\beta < 1$ (i.e., penalize)
  - RETURN *Make-Predictor* (w, P)

Weighted Majority: Properties

- Advantages of WM Algorithm
  - Can be adjusted incrementally (without retraining)
  - Mistake bound for WM
    - Let D be any sequence of training examples, L any set of inducers
    - Let k be the *minimum number of mistakes made* on D by any $L[i]$, $1 \leq i \leq n$
    - Property: number of mistakes made on D by *Combiner-Weighted-Majority* is at most $2.4\ (k + \lg n)$
- Applying *Combiner-Weighted-Majority* to Produce Test Set Predictor
  - *Make-Predictor*: applies abstraction; returns funarg that takes input $x \in D_{test}$
  - Can use this for incremental learning (if $c(x)$ is available for new x)
- Generalizing *Combiner-Weighted-Majority*
  - Different input to inducers
    - Can add an argument s to sample, transform, or partition D
    - Replace $P[i] \leftarrow L[i].Train\text{-}Inducer\ (D)$ with $P[i] \leftarrow L[i].Train\text{-}Inducer\ (s(i, D))$
    - Still compute weights based on performance on D
  - Can have q_c ranging over more than 2 class labels

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

Bagging: Procedure

- Algorithm *Combiner-Bootstrap-Aggregation* (D, L, k)
  - FOR $i \leftarrow 1$ TO k DO
    - $S[i] \leftarrow Sample\text{-}With\text{-}Replacement\ (D, m)$
    - $Train\text{-}Set[i] \leftarrow S[i]$
    - $P[i] \leftarrow L[i].Train\text{-}Inducer\ (Train\text{-}Set[i])$
  - RETURN (*Make-Predictor* (P, k))
- Function *Make-Predictor* (P, k)
  - RETURN (fn $x \Rightarrow$ *Predict* (P, k, x))
- Function *Predict* (P, k, x)
  - FOR $i \leftarrow 1$ TO k DO

    $Vote[i] \leftarrow P[i](x)$
  - RETURN (*argmax* (Vote[i]))
- Function *Sample-With-Replacement* (D, m)
  - RETURN (m data points sampled i.i.d. uniformly from D)

Bagging: Properties

- Experiments
  - [Breiman, 1996]: Given sample S of labeled data, do 100 times and report average
    - 1. Divide S randomly into test set D_{test} (10%) and training set D_{train} (90%)
    - 2. Learn decision tree from D_{train}

      $e_S \leftarrow$ error of tree on T
    - 3. Do 50 times: create bootstrap $S[i]$, learn decision tree, prune using D

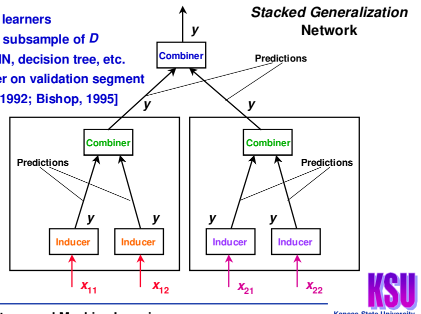
      $e_B \leftarrow$ error of majority vote using trees to classify T
  - [Quinlan, 1996]: Results using UCI Machine Learning Database Repository
- When Should This Help?
  - When learner is unstable
    - Small change to training set causes large change in output hypothesis
    - True for decision trees, neural networks; not true for k-nearest neighbor
  - Experimentally, bagging can help substantially for unstable learners, can somewhat degrade results for stable learners

Bagging: Continuous-Valued Data

- Voting System: Discrete-Valued *Target Function* Assumed
  - Assumption used for WM (version described here) as well
    - Weighted vote
    - Discrete choices
  - Stacking: generalizes to continuous-valued targets *iff* combiner inducer does
- Generalizing Bagging to Continuous-Valued Target Functions
  - Use mean, not mode (*aka* argmax, majority vote), to combine classifier outputs
  - Mean = expected value
    - $\phi_A(x) = E_D[\phi(x, D)]$
    - $\phi(x, D)$ is base classifier
    - $\phi_A(x)$ is aggregated classifier
  - $(E_D[y - \phi(x, D)])^2 = y^2 - 2y \cdot E_D[\phi(x, D)] + E_D[\phi^2(x, D)]$
    - Now using $E_D[\phi(x, D)] = \phi_A(x)$ and $EZ^2 \geq (EZ)^2$, $(E_D[y - \phi(x, D)])^2 \geq (y - \phi_A(x))^2$
    - Therefore, we expect lower error for the bagged predictor ϕ_A

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: Procedure

- Algorithm *Combiner-Stacked-Gen* (*D, L, k, n, m', Levels*)
  - Divide *D* into *k* segments, *S*[1], *S*[2], …, *S*[*k*] // Assert *D.size* = *m*
  - FOR *i* ← 1 TO *k* DO
    - *Validation-Set* ← *S*[*i*] // *m*/*k* examples
    - FOR *j* ← 1 TO *n* DO
      Train-Set[*j*] ← *Sample-With-Replacement* (*D* ~ *S*[*i*], *m'*) // *m* - *m*/*k* examples
      IF *Levels* > 1 THEN
      P[*j*] ← *Combiner-Stacked-Gen* (*Train-Set*[*j*], *L*, *k*, *n*, *m'*, *Levels* - 1)
      ELSE // Base case: 1 level
      P[*j*] ← *L*[*j*].*Train-Inducer* (*Train-Set*[*j*])
    - *Combiner* ← *L*[0].*Train-Inducer* (*Validation-Set.targets*, *Apply-Each* (*P*, *Validation-Set.inputs*))
  - *Predictor* ← *Make-Predictor* (*Combiner*, *P*)
  - RETURN *Predictor*
- Function *Sample-With-Replacement*: Same as for Bagging

Stacked Generalization: Properties

- Similar to Cross-Validation
  - *k*-fold: rotate validation set
  - Combiner mechanism based on validation set as well as training set
    - Compare: committee-based combiners [Perrone and Cooper, 1993; Bishop, 1995] *aka* consensus under uncertainty / fuzziness, consensus models
    - Common application with cross-validation: treat as overfitting control method
  - *Usually improves generalization performance*
- Can Apply Recursively (Hierarchical Combiner)
  - Adapt to inducers on different subsets of input
    - Can apply *s*(*Train-Set*[*j*]) to transform each input data set
    - e.g., attribute partitioning [Hsu, 1998; Hsu, Ray, and Wilkins, 2000]
  - Compare: Hierarchical Mixtures of Experts (HME) [Jordan *et al*, 1991]
    - Many differences (validation-based vs. mixture estimation; online vs. offline)
    - Some similarities (hierarchical combiner)

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

Terminology

- 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* ~ *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

Summary Points

- 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 Lecture: Boosting the Margin, Hierarchical Mixtures of Experts