Slide07

Haykin Chapter 7: Committee Machines

CPSC 636-600

Instructor: Yoonsuck Choe

Spring 2008

Introduction

- Divide and conquer
- Distributing the learning task among a number of experts
- Combination of experts: committee machine
- Fuses knowledge attained by individual experts to come up with an overall decision that is superior to that by any individual

Categories of committee machines

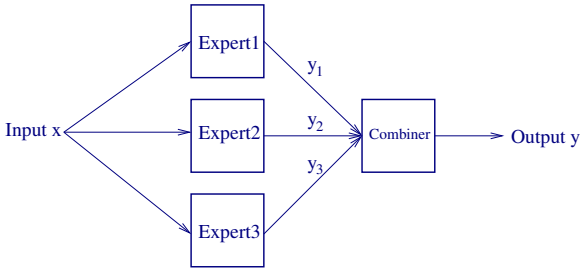
- Static structures: combine expert's response without reference to the input
  - Ensemble averaging: Linear combination of expert outputs
  - Boosting: Use weak algorithm to achieve arbitrarily high accuracy
- Dynamic structures: Input is directly involved in actuating the integration mechanism
  - Mixture of experts: Nonlinear combination of expert outputs by means of a single gating network
  - Hierarchical mixture of experts: Same as above, but with a hierarchically arranged gating networks

Modular Networks

- Mixture of experts and hierarchical mixture of experts are examples of *modular* networks (Osherson et al., 1990).

  > A neural network is said to be modular if the computation performed by the network can be decomposed into two or more modules (subsystems) that operate on distinct inputs without communicating with each other. The outputs of the modules are mediated by an integrating unit that is not permitted to feed information back to the modules. In particular, the integrating unit both (1) decides how the outputs of the modules should be combined to form the final output of the system, and (2) decides which modules should learn which training patterns.

Ensemble Averaging



- Outputs of a number of differently trained experts (given common input) are combined.
- Motivation for using ensemble averaging:
  - The whole, as a network, may contain too many tunable parameters, resulting in very long training time.
  - Risk of overfitting increases with increase in the number of parameters.

Bias vs. Variance Revisited

- $f(x)$: true function to learn; $F(x)$: nnet approximation; $\mathcal{D}$: space of all training sets and all initial conditions.
- We know that the mean error over the space $\mathcal{D}$ can be decomposed into bias and variance:

$$E_{\mathcal{D}}\left[(F(\mathbf{x}) - E[D|\mathbf{X}=\mathbf{x}])^2\right] = B_{\mathcal{D}}(F(\mathbf{x})) + V_{\mathcal{D}}(F(\mathbf{x}))$$

$$B_{\mathcal{D}}(F(\mathbf{x})) = (E_{\mathcal{D}}[F(\mathbf{x})] - E[D|\mathbf{X}=\mathbf{x}])^2,$$

$$V_{\mathcal{D}}(F(\mathbf{x})) = E_{\mathcal{D}}\left[(F(\mathbf{x}) - E_{\mathcal{D}}[F(\mathbf{x})])^2\right]$$

Note: $f(\mathbf{x}) = E[D|\mathbf{X}=\mathbf{x}]$.

Bias/Variance in Ensemble Averages

- Train ensemble average $F_I(x)$ using a set of initial conditions I. (Denote the space of all initial conditions as $\mathcal{I}$.)
- The expected error over this initial condition space $\mathcal{I}$ can also be decomposed into bias/variance:

$$E_{\mathcal{I}}\left[(F_I(\mathbf{x}) - E[D|\mathbf{X}=\mathbf{x}])^2\right] = B_{\mathcal{I}}(F(\mathbf{x})) + V_{\mathcal{I}}(F(\mathbf{x})),$$

$$B_{\mathcal{I}}(F(\mathbf{x})) = (E_{\mathcal{I}}[F_I(\mathbf{x})] - E[D|\mathbf{X}=\mathbf{x}])^2$$

$$V_{\mathcal{I}}(F(\mathbf{x})) = E_{\mathcal{I}}\left[(F_I(\mathbf{x}) - E_{\mathcal{I}}[F(\mathbf{x})])^2\right]$$

- By partitioning $\mathcal{D}$ into $\mathcal{I}$ and the remnant $\mathcal{D}'$, we can also write:

$$E_{\mathcal{D}'}\left[(F_I(\mathbf{x}) - E[D|\mathbf{X}=\mathbf{x}])^2\right] = B_{\mathcal{D}'}(F_I(\mathbf{x})) + V_{\mathcal{D}'}(F_I(\mathbf{x}))$$

Bias/Variance in Ensemble Avg. (cont'd)

- From $E_{\mathcal{D}'}\left[(F_I(\mathbf{x}) - E[D|\mathbf{X}=\mathbf{x}])^2\right] = B_{\mathcal{D}'}(F_I(\mathbf{x})) + V_{\mathcal{D}'}(F_I(\mathbf{x}))$ we know that

$$B_{\mathcal{D}'}(F_I(\mathbf{x})) = (E_{\mathcal{D}'}[F_I(\mathbf{x})] - E[D|\mathbf{X}=\mathbf{x}])^2$$

$$V_{\mathcal{D}'}(F_I(\mathbf{x})) = E_{\mathcal{D}'}\left[(F_I(\mathbf{x}) - E_{\mathcal{D}'}[F_I(\mathbf{x})])^2\right]$$

- Since we also know that

$$E_{\mathcal{D}'}[F_I(\mathbf{x})] = E_{\mathcal{D}}[F(\mathbf{x})],$$

$$B_{\mathcal{D}'}(F_I(\mathbf{x})) = (E_{\mathcal{D}}[F(\mathbf{x})] - E[D|\mathbf{X}=\mathbf{x}])^2 = B_{\mathcal{D}}[F(\mathbf{x})]$$

- From the above and $E[(X - E[X])^2] = E[X^2] - E[X]^2$, we can also deduce that

$$\begin{aligned} V_{\mathcal{D}'}(F_I(\mathbf{x})) &= E_{\mathcal{D}'}[(F_I(\mathbf{x}))^2] - (E_{\mathcal{D}'}[F_I(\mathbf{x})])^2 \\ &= E_{\mathcal{D}'}[(F_I(\mathbf{x}))^2] - (E_{\mathcal{D}}[F(\mathbf{x})])^2 \end{aligned}$$

Bias/Variance in Ensemble Avg. (cont'd)

- From the following

$$B_{\mathcal{D}'}(F_I(\mathbf{x})) = B_{\mathcal{D}}[F(\mathbf{x})]$$

$$V_{\mathcal{D}'}(F_I(\mathbf{x})) = E_{\mathcal{D}'}[(F_I(\mathbf{x}))^2] - (E_{\mathcal{D}}[F_I(\mathbf{x})])^2$$

and the observation that

$$V_{\mathcal{D}}(F_I(\mathbf{x})) = E_{\mathcal{D}}[(F_I(\mathbf{x}))^2] - (E_{\mathcal{D}}[F(\mathbf{x})])^2$$

$$E_{\mathcal{D}}[F(\mathbf{x})^2] \geq E_{\mathcal{D}'}[(F_I(\mathbf{x})^2],$$

we can conclude that

$$V_{\mathcal{D}'}(F_I(\mathbf{x})) \leq V_{\mathcal{D}}(F(\mathbf{x}))$$

- *In sum, the bias of ensemble averaged $F_I(\mathbf{x})$ is the same as that of $F(\mathbf{x})$, and the variance is less.*

Bias/Variance in Ensemble Averaging

Main result:

- Bias in ensemble-averaged $F_{\mathrm{I}}(\mathbf{x})$ is the same as that of the constituent experts.
- Variance of the ensemble-averaged $F_{\mathrm{I}}(\mathbf{x})$ is less than that of the constituent experts.

Thoughts:

- The experts may be identical, with the only difference being initial condition.
- Each expert is overtrained (reduce bias, while increased variance).
- Variance is subsequently reduced through ensemble averaging.

Boosting

- Experts are trained on data sets with entirely different distributions.
- This is a general method that can improve the performance of *any* learning algorithm.

Three Approaches to Boosting

1. Boosting by filtering: Filter training examples with different versions of a weak learning algoritjm. Assumes a large (in theory, infinite) source of examples, where examples are kept or discarded during training. Small memory requirement.
2. Boosting by subsampling: Training sample of fixed size. The examples are "resampled" according to a given probability distribution during training. Error calculated with a fixed training sample.
3. Boosting by reweighting: Training sample of fixed size. Weak learning algorithms can receive "weighted" examples. Error calculated with respect to the weighted samples.

Strong vs. Weak Learning Model

- PAC learning: strong learning model
  - Less than ϵ error, with the probability of $(1 - \delta)$
- Weak learning model:
  - Drastically relaxed requirement
  - Hypothesis needs to have error rate slightly less than 1/2.
  - Note, for binary concepts, a totally random guessing algorithm will make 1/2 error.
- Hypothesis boosting problem: *Are the notions of strong and weak learning equivalent?* (Kearns and Valiant 1989)
- Answer: Yes! Concept classes that are weakly learnable are also strongly learnable. (Shapire 1990)

Boosting by Filtering

1. First expert trained with N_1 examples.
2. Expert 1 used to *filter* another set of examples:
   - Flip a fair coin to generate random guess.
   - If *head*, pass new input pattern through expert 1 and discard if correctly classified, until a pattern is misclassified. Add the misclassified pattern to the training set for expert 2.
   - If *tail*, do the opposite.
   - Repeat until N_1 samples have been filtered.
   - Expert 2 trained on the filtered samples (Expert 1 makes exactly 1/2 mistakes on this set).
3. Expert 3 trained:
   - Pass a new input pattern through expert 1 and expert 2. If the two agrees, discard the pattern. Otherwise, add to expert 3 training set.
   - Continue until N_1 examples are generated.
   - Train expert 3 with those samples.

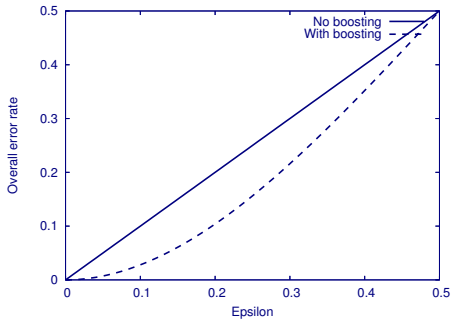
Boosting by Filtering: Computational Considerations

- Data needed:
  - N_1 for expert 1.
  - N_2 for generating N_1 inputs for expert 2.
  - N_3 for generating N_1 inputs for expert 3.
  - Total: $N_4 = N_1 + N_2 + N_3$.
- Computation:
  - Total: $3N_1$ inputs used for the training of three experts.
- Main idea: Resulting distribution focus on "hard-to-learn" part of instance space.

Boosting by Filtering: Classifying New Instances

- Original (Shapire 1990):
  - If expert 1 and 2 agree, use that decision.
  - Otherwise, use expert 3.
- Modified (Drucker et al. 1993, 1994):
  - Addition of respective outputs of the three experts.

Boosting by Filtering: Error Bound



- Schapire (1990) showed that the overall error of a committee machine with each experts committing $\epsilon < 1/2$ is bounded by:

$$g(\epsilon) = 3\epsilon^2 - 2\epsilon^3.$$

Boosting by Filtering: Discussion

- Weak learning model can be converted to strong learning model.
- Needs a lot of input for training.
- What to do when the input is limited?: Use AdaBoost

Boosting by Resampling: AdaBoost

- Freund and Schapire (1996a, 1996b)
- Overcomes excessive input requirement in boosting by filtering.
- Permits the reuse of the training set when resampling.

AdaBoost: General

- Weak learning model available; Goal is to learn an approximation with low error rate relative to a given distribution $\mathcal{D}$ over the labeled training examples.
- Different from boosting by filtering:
  - Adjusts *adaptively* to the errors of the weak method.
  - Bound on performance depends only on the performance of the weak learning model on those input distributions that are actually generated during the learning process.

AdaBoost Algorithm

- On iteration n, the boosting algorithm provides the weak learning model with a distribution $\mathcal{D}_n$ over the training sample $\mathcal{T}$.
- In response, the weak learning model computes $F_n : X \to Y$ that correctly classifies a fraction of the training samples. The error is measured with respect to $\mathcal{D}_n$.
- The process continues for T iterations, then all $F_1, F_2, ..., F_T$ are combined into F_{fin}.

AdaBoost: Sketch

- Updating $\mathcal{D}_n$:
  - Start with uniform distibution $\mathcal{D}_1(i) = 1/n$ for all i.
  - Learn F_n given $\mathcal{D}_n$.
  - Change distribution: multiply by $\beta_n = err/(1 - err)$ if $F_n(x_i) = d_i$ (reduce weight) and leave alone if not. Normalize with sum of $\mathcal{D}_n$.
- Combining :
  - Take weighted vote of $F_1, F_2, ..., F_T$.
  - Given input x, F_{fin} outputs the label d that maximizes the sum of weights of the F_i predicting that label.
  - Weight is $\log(1/\beta_n)$, which is larger for smaller error.

AdaBoost: Algorithm

Input: Training sample $\{\langle x_i, d_i \rangle\}_{i=1}^N$;
Distribution $\mathcal{D}$ over N labeled examples;
Weak learning model; Number of iterations T

Init: Set $\mathcal{D}_1(i) = 1/N$ for all i.

Computation: Do the following for $n = 1, 2, ..., T$.

1. Call weak learning model, with distribution $\mathcal{D}_n$.
2. Get back $F_n : X \to Y$.
3. Calculate error of F_n:

$$\epsilon_n = \sum_{i:F_n(x_i) \neq d_i} \mathcal{D}_n(i)$$

AdaBoost: Algorithm (Cont'd)

4. Set $\beta_n = \frac{\epsilon_n}{1-\epsilon_n}$ (note: $\beta_n \in [0, 1)$)
5. Update distribution $\mathcal{D}_n$:

$$\mathcal{D}_{n+1}(i) = \frac{D_n(i)}{Z_n} \times \begin{cases} \beta_n & \text{if } F_n(x_i) = d_i \\ 1 & \text{otherwise} \end{cases}$$

where Z_n is a normalization constant.

6. Output: The final approximation is

$$F_n(x) = \arg\max_{d \in \mathcal{D}} \sum_{n:F_n(x)=d} \log \frac{1}{\beta_n}$$

AdaBoost: Theoretical Importance

Freund and Schapire (1996a):

- Suppose the weak learning model, when called by AdaBoost, generates $F_i(\mathbf{x})$ with errors $\epsilon_1, \epsilon_2, ..., \epsilon_T$, where

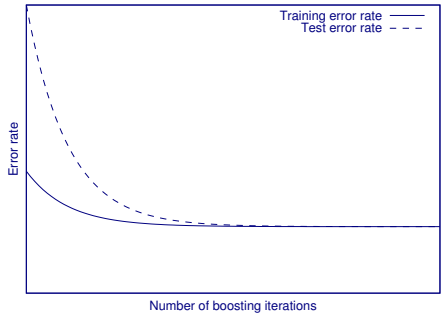
$$\epsilon_n = \sum_{i:F_n(x_i)\neq d_i} \mathcal{D}_n(i).$$

- Assume that $\epsilon_n \leq 1/2$, and let $\gamma_n = 1/2 - \epsilon_n$. Then the following upper bound holds on the error of the final approximation:

$$\frac{1}{N}\left|\{i : F_{\text{fin}}(x_i) \neq d_i\}\right| \leq \prod_{n=1}^{T} \sqrt{1-4\gamma_n^2} \leq \exp\left(-2\sum_{n=1}^{T}\gamma_n^2\right).$$

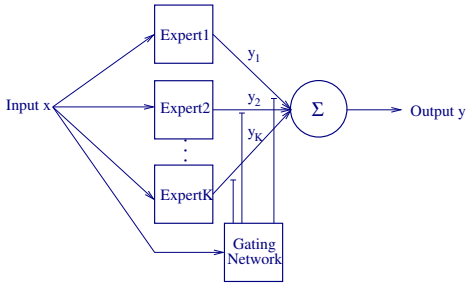
- In other words, if weak algorithm does slightly better than 1/2, training error of F_{fin} drops to zero exponentially fast.

Training and Generalization in AdaBoost



- Theoretical bound on training error is often weak.
- Generalization error tends to be much better than what the theory would suggest.
- Very often, test error continues to decrease even after training error reaches 0. (No over-fitting!)

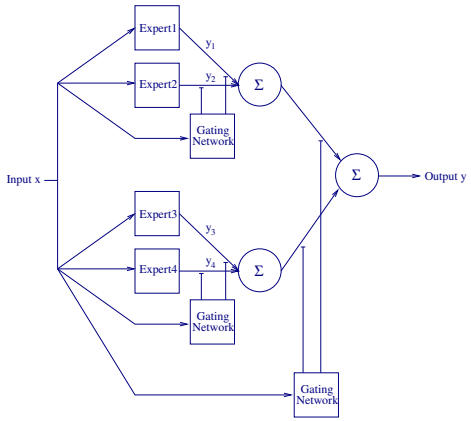
Mixture of Experts



Dynamic (input influences the committee decision)

- Experts: $y_k = \mathbf{w}_k^T \mathbf{x}$
- Gating: $g_k = \frac{\exp(u_k)}{\sum_{j=1}^{K}\exp(u_j)}, u_k = \mathbf{a}_k^T \mathbf{x}$
- Final output: $y = \sum_{k=1}^{K} g_k y_k$.

Hierarchical Mixture of Experts (HME)



- Dynamic (input influences the committee decision)
- Multiple levels of gating decisions.

Hierarchical Mixture of Experts (cont'd)

- HME is based on a *divide and conquer* strategy.
- HME is a *soft-decision tree*: it is a probabilistic generalization of the standard decision tree (hard
- HME may perform better than hard decision trees:
  - Hard decisions result in loss of information.
  - Hard decisions are irrevertible, and thus suffer problems inherent in greedy methods.

Learning in HME

- **Stochastic gradient approach**:
  - Conduct gradient descent on $\mathbf{w}_{jk}$ of each experts.
  - Conduct gradient descent on $\mathbf{a}_k$ of the gating network (top level)
  - Conduct gradient descent on $\mathbf{a}_{jk}$ of the gating network (intermediate levels)
- **Expectation-maximixation approach** (EM: Dempster 1970)
  - Expectation step: using observable data and current estimate of the parameters, construct the unobserved (missing) data.
  - Maxmization step: given the complete data (observable data + current estimate of the missing data), tune the parameters.
  - *Indicator variables* are introduced as "dummy" missing data, to facilitate the use of EM in HME learning.

Summary

- Static: Ensemble averaging and boosting
- Dynamic: Mixture of experts, Hierarchical mixture of experts
- Balances between
  - Simple learning model's understandability
  - Complex learning model's performance