Combining Models

Steven J Zeil
Old Dominion Univ.
Fall 2010

Combining Models

- **TANSTAAFL**: There is no algorithm that is always the most accurate
- Generate a group of base-learners which combine to give higher accuracy
- Different learners use different
  - Algorithms
  - Hyperparameters
  - Representations / Modalities / Views
  - Training sets
  - Subproblems
- Diversity trumps accuracy

Approaches

- **Multi-expert** combinations: base learners work in parallel
  - global: all learners generate an output and all are used
  - local: input is examined and used to select learners to be used
- **Multi-stage** combinations: base learners applied in order of increasing complexity
  - later ones trained/tested only on instances where earlier ones were inaccurate

Voting

- Learners emit posterior probs (or other similarly normalized outputs)
- Linear combination
  $$y = \sum_{j=1}^{L} w_j d_j, \ w_j \geq 0 \land \sum_{j=1}^{L} w_j = 1$$
  Weights can be based on relative accuracy.
- Other combination rules: median, minimum, maximum, product
Error-Correcting Output Codes

- K classes, L learners
- Code a matrix $W$ in terms of which classes are discriminated by which learners

$$W = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

- Columns are the discrimination implemented by a learner
- Rows denote condition for identifying a class

Seeking Robustness

- K classes, L learners
- Suppose $K = L$

$$W = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \end{bmatrix}$$

- A mistake by any one learner can lead to misclassification
- Solution: Let $L > K$ and increase the Hamming distance between rows
- Then vote based on $W$:

$$y_i = \sum_{j=1}^{L} w_{ij} d_j$$

and choose class with highest $y_i$

Bagging

- Generate L training sets (sample with replacement) and train one base-learner with each
- Use voting (average or median with regression)
- Can improve results from unstable algorithms

Boosting

- Train next learner on the mistakes of the previous ones
- 3 weak learners.
  - Train $L_1$ on 1/3 of the training set
  - Train $L_2$ on inputs from the second 1/3 of the training set that are misclassified by $L_1$
  - Train $L_3$ on inputs from the final 1/3 of the training set that are misclassified by $L_1$ and $L_2$
- During operation, present inputs to $L_1$ and $L_2$. If they agree, accepts. If they disagree, use output from $L_3$
**AdaBoost**

- Adaptive boosting - works on a smaller training set
- **Training:**
  - Associate a prob 1/N with each training input
  - Draw a sample of the training set according to those probabilities
  - Train a learner and test on entire training set
    - Decrease the probabilities of any correctly classified inputs
  - Repeat until total error is acceptable
- **Operation:** voting with weights inversely proportional to error rate during testing

**Mixture of Experts**

Voting with weights a function of the input

**Stacking**

$f()$ is another learner
- Must be trained on a separate set than that used for the base learners

**Cascading**

Use $d_i$ only if preceding ones are not confident