Model Ensembles

**Bagging**

- Generate “bootstrap” replicates of training set by sampling with replacement
- Learn one model on each replicate
- Combine by uniform voting

**Boosting**

- Maintain vector of weights for examples
- Initialize with uniform weights
- Loop:
  - Apply learner to weighted examples (or sample)
  - Increase weights of misclassified examples
- Combine models by weighted voting

---

**Model Ensembles**

- **Basic idea:**
  Instead of learning one model, learn several and combine them
- **Typically improves accuracy, often by a lot**
- **Many methods:**
  - Bagging
  - Boosting
  - ECOC (error-correcting output coding)
  - Stacking
  - Etc.

---

**ADABOOST** $(S, \text{Learn}, k)$

- $S$: Training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, $y_i \in Y$
- $\text{Learn}$: Learn$(S, \text{weights})$
- $k$: # Rounds
- For all $i$ in $S$: $w_1(i) = 1/m$
- For $r = 1$ to $k$ do
  - For all $i$: $p_r(i) = w_r(i) / \sum w_r(i)$
  - $h_r = \text{Learn}(S, p_r)$
  - $\epsilon_r = \sum p_r(i) \[ h_r(i) \neq y_i \]$
  - If $\epsilon_r > 1/2$ then
    - $k = r - 1$
    - Exit
  - $\beta_r = \epsilon_r / (1 - \epsilon_r)$
  - For all $i$: $w_{r+1}(i) = w_r(i) \beta_r^{-1} \cdot [ h_r(x_i) \neq y_i ]$
- Output: $h(x) = \arg \max_y \sum_r \left( \log \frac{1}{\beta_r} \right) 1[h_r(x) = y]$
**Error-Correcting Output Coding**

- **Motivation:**
  Applying binary classifiers to multiclass problems
- **Train:** Repeat $L$ times:
  - Form a binary problem by randomly assigning classes to “superclasses” 0 and 1
    - E.g.: A, B, D $\rightarrow$ 0; C, E $\rightarrow$ 1
  - Apply binary learner to binary problem
- **Test:**
  - Apply each classifier to test example, forming vector of predictions $P$
  - Predict class whose vector is closest to $P$ (Hamming)

---

**Model Ensembles: Summary**

- Learn several models and combine them
- Bagging: Random resamples
- Boosting: Weighted resamples
- ECOC: Recode outputs
- Stacking: Multiple learners

---

**Stacking**

- Apply multiple base learners (e.g.: decision trees, naive Bayes, neural nets)
- Meta-learner: Inputs = Base learner predictions
- Training by leave-one-out cross-validation:
  Meta-L. inputs = Predictions on left-out examples