Announcements:
- Take home: due now.
- In-class exam: grading still in process.
- Project: data & instructions to be posted this weekend.

Outline for Today: Model Combination

Different goals/motivations
1) Two (or more) heads are better than one.
2) Dealing with high variance models.
3) Dealing with difficult cases.
General idea: (for regression or classification)

Most approaches work for most models

\$f_i \neq f_j \neq f_k\$ must be different. Ways to achieve this

- different feature sets
- different classifiers
- different samplings of the training data (bagging)
- different weightings of the training data (boosting)

$g_j$: most anything could be used
popular options: averaging, log average
Generic Model Combination:

A) When you don't have a lot of training data, use a model that involves no learning:

1. Train \( f_b \) on \( b = 1, \ldots, B \)
2. \( g = \frac{1}{B} \sum_{b=1}^{B} f_b(x) \)
   or \( g = \text{majority}(f_b(x)) \)

B) When you have enough training data to have a held out set:

\( D = \{ D_{tr}, D_{ho} \} \)

1. Train \( f_b \) on \( D_{tr} \) for \( b = 1, \ldots, B \)
2. Apply \( f_b \) to \( D_{ho} \) \( \Rightarrow (y_i^{\text{ho}}, x_i, y_i) \)
3. Use \((y_i^{\text{ho}}, y_i)\) to find \( g \)

   a) find \( w_b = P(B_{corr}) = \frac{n_{\text{ho}}}{n} \)

   b) find prediction model parameters (linear, SVM, NN)
In order for this to work you need the classifiers to have roughly the same performance but make different errors.

Cost: higher training & classification (or prediction) costs (computation & storage)

An example where $g$ uses $x$: Mixture of Experts

Learn with EM; expert = hidden variable
Taking a different approach:
Sampling the training set
(Features & model are same \( \& b \))
\[ \Rightarrow \text{Bagging} \]

---

To get some intuition:

\[ \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \quad E(\bar{x}) = E(x_i) \]

\[ \text{Var}(\bar{x}) = \frac{\text{Var}(x_i)}{n} \quad \text{if } x_i \text{ are i.i.d.} \]

For these to be completely independent, need to divide data
On the smaller data sets, variance is higher

Need a more clever way of getting multiple samples
\[ \Rightarrow \text{Bootstrapping} \]

Random sampling with replacement
Bagging

Given \( \mathcal{X} = \{x_1, x_2, \ldots, x_n\} \)

For \( b = 1, \ldots, B \)

Find \( x'_b \) with \( n' < n \) samples
by Sampling \( \mathcal{X} \) with replacement.

Train model on \( x'_b \)

\[
g(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x)
\]

Simple but powerful
particularly useful for decision trees (unstable)

Need \( n' < n \) can't be too big or you don't get variation
\( n' \) can't be too small or the classifiers are too weak
A few comments:

Bias is unchanged a variance is reduced for squared error loss ⇒ overall win

For 0/1 loss, no guarantee a bad classifier can become worse with bagging (see HTK)

In practice, works pretty well

Another plus: or asymmetric loss with skewed classes, you can skew sampling (related to boosting)
Data Weighting (Boosting)

Basic idea: iteratively design models based on reweighted data, with data where you make errors given more weight.

\[ G(x) = \text{Sign} \left[ \sum_{m=1}^{M} \alpha_m G_m(x) \right] \]
\[ \forall \epsilon \in \{-1, 1\} \]

AdaBoost
AdaBoost Algorithm

1. Initialize \( w_i = \frac{1}{N} \)

2. For \( m = 1, \ldots, M \)
   
   Train \( G_m(x) \)
   
   Compute \( E_m = \sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i)) \)
   
   \( \sum_{i} w_i \)

   Compute \( \alpha_m = \frac{1}{2} \log \left( \frac{1 - E_m}{E_m} \right) \)

   \( \forall i \), Update \( w_i \leftarrow w_i e^{-\alpha_m y_i G_m(x_i)} \)

3. Output \( G(x) = \text{sign} \left( \sum_{m} \alpha_m G_m(x) \right) \)
How do weights affect learning?

Different options

1) Use for sampling
2) Use in the objective

Simple learners are good for boosting!

(linear classifiers, decision “Stumps” (2 terminal tree No))

HTF Fig 10.2 & 10.9
Other comments:

- Component classifiers must perform better than chance.
- Extensions exist for regression.
- Package for text problems (categorical variables): Boostexter.