Lecture 18
Ensemble

Mahdi Roozbahani
Georgia Tech

Adapted from Polo.

Which Classifier/Model to Choose?

- Possible strategies:
- Go from simplest model to more complex model until you obtain desired accuracy
- Discover a new model if the existing ones do not work for you
- Combine all (simple) models

Common Strategy: Bagging
(Bootstrap Aggregating)

Originally designed for combining multiple models,
to improve classification “stability” [Leo Breiman, 94]

Uses random training datasets
(sampled from one dataset)

http://statistics.about.com/od/Applications/a/What-Is-Bootstrapping.htm

Common Strategy: Bagging
(Bootstrap Aggregating)

Consider the data set $S = \{(X_i, Y_i)\}_{i=1,\ldots,n}$

- Pick a sample S^* with **replacement** of size *n*
  (S called a "bootstrap sample")*

$$S \rightarrow \; x = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 9 & 10 & 11 & 12 \\ 20 & 21 & 22 & 23 \\ 5 & 6 & 7 & 8 \end{bmatrix} \; y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$S^* \rightarrow \; x^* = \begin{bmatrix} 9 & 10 & 11 & 12 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 4 \end{bmatrix} \; y^* = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Common Strategy: Bagging

(Bootstrap Aggregating)

Consider the data set $S = \{(X_i, Y_i)\}_{i=1,\ldots,n}$

- Pick a sample S^* with **replacement** of size n
  (S called a "bootstrap sample")*
- Train on S^* to get a classifier f^*
- Repeat above steps B times to get $f_1, f_2, \ldots, f_B$
- Final classifier $f(x) = \text{majority}\{f_b(x)\}_{j=1,\ldots,B}$

Common Strategy: Bagging

Why would bagging work?

- Combining multiple classifiers reduces the variance of the final classifier

When would this be useful?

- We have a classifier with high variance

Bagging decision trees

Consider the data set S

- Pick a sample S^* with replacement of size n
- Grow a decision tree T_b
- Repeat B times to get $T_1, \ldots, T_B$
- The final classifier will be

$$f(x) = \text{majority}\{f_{T_b}(x)\}_{b=1,\ldots,B}$$

Random Forests

Almost identical to bagging decision trees, except we introduce some randomness:

- Randomly pick *m* of the *d* available attributes, at every split when growing the tree
  (i.e., d - m attributes ignored)

Bagged **random** decision trees =
Random forests

What are our Hyper-Parameters in Random Forest

$m = Number\ of\ randomly\ chosen\ attributes$

Usual values for $m = \sqrt{d}, 1, 10$

d is number of dimensions or features or attributes

How to optimize m? Cross-Validation

$B = Number\ of\ models\ or\ decision\ trees\ in\ Random\ Forest$

Keep adding trees until training error stabilizes (reaches to a plateau)