Random Forests → toughest lecture of the class...

* Decision trees are unstable & prone to overfitting.
* How to Reduce overfitting? Ask Leo Breiman...

- Small perturbations in the training set or in construction may lead to large changes in the constructed predictor

> Unstable methods can have their accuracy improved by perturbing and combining, that is, generate multiple versions of the predictor by perturbing the training set or construction method, then combine these multiple versions into a single predictor. [redacted]

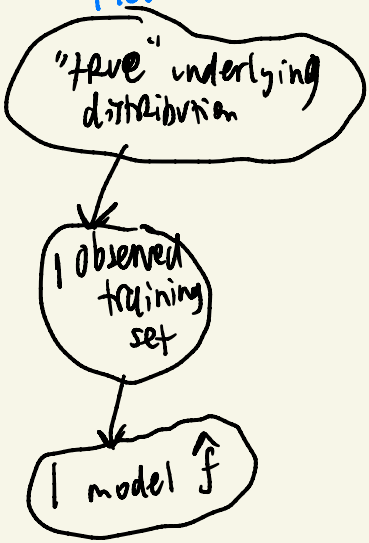
Bagging training set T consists of N instances $n=1,\dots,N$
put equal probabilities $p(n) = \frac{1}{N}$ on each instance
Sample With Replacement (Bootstrap) N times from the training set T, forming resampled training set $T^{(B)}$
Fit decision tree on $T^{(B)}$, yielding $\hat{f}^{(B)}$.
Repeat the procedure and combine,

$$\hat{f} = \frac{1}{B}\sum_{B=1}^{B} \hat{f}^{(B)}.$$

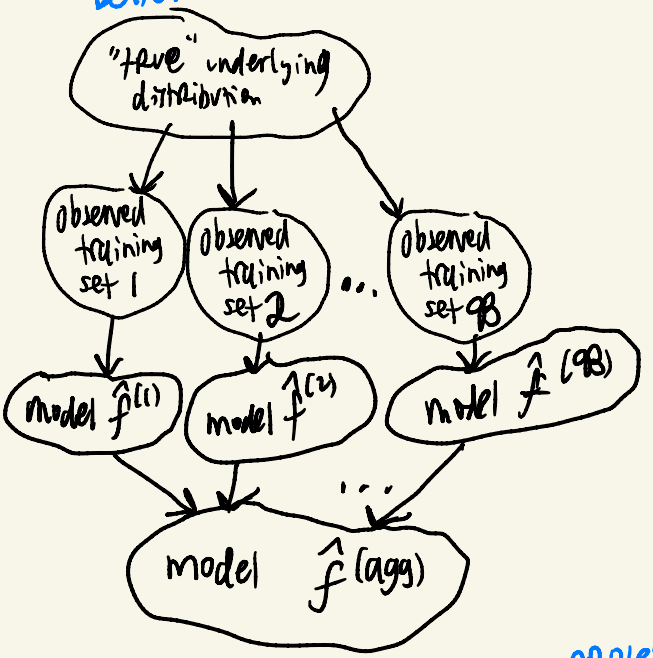
Bagging = Bootstrap - Aggregating.

Why Bootstrap?

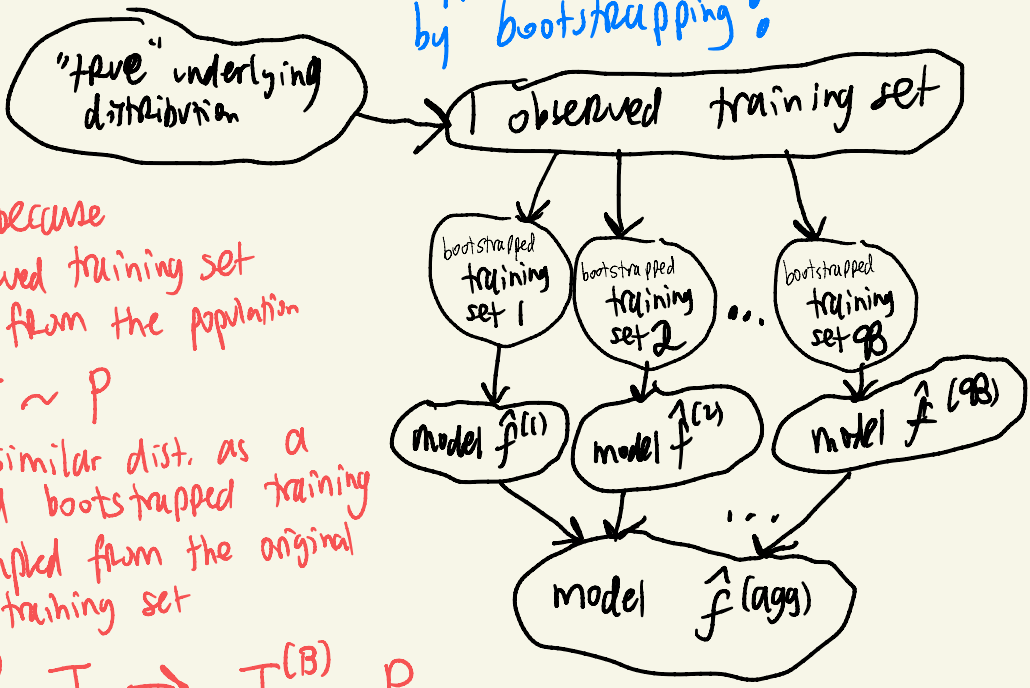
Have:



Something better:



Approximate this process by bootstrapping:

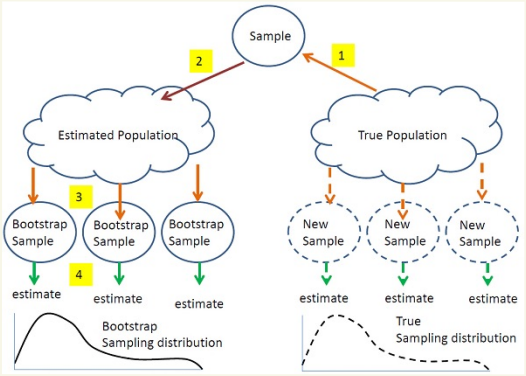


Works because the observed training set sampled from the population

$$T \sim P$$

has a similar dist. as a re-sampled bootstrapped training set sampled from the original observed training set

$$T^{(B)} \sim T \Rightarrow T^{(B)} \approx P$$



ARcing (i.e. Boosting; Adaboost)

- start with $p(n) = \frac{1}{N}$ and re-sample from T to form the first training set $T^{(1)}$, fit decision tree $\hat{f}^{(1)}$ on $T^{(1)}$
- as the sequence of training sets and classifiers is being built, increase $p(n)$ for those cases that have been most frequently misclassified
- at termination, combine the classifier by weighted or simple voting $\hat{f} = \frac{1}{B} \sum_{B=1}^{B} \hat{f}^{(B)}$.

ARcing = adaptive resampling and combining.

To better understand stability and instability and what bagging and arcing do, we define the concepts of bias and variance for classifiers in Section 2. The difference between the test set misclassification error for the classifier and the minimum error achievable is the sum of the bias and variance. Unstable classifiers such as trees characteristically have high variance and low bias.

* The main effect of Bagging and ARcing is to Reduce Variance (a la the Bias-Variance tradeoff).

The Bias and Variance of a Classifier

Setup

output variable $y \in \{1, \ldots, J\}$ in classification (e.g. win/loss)

training set $T = \{(x_n, y_n)\}_{n=1}^{N}$

given T, some method (e.g. decision tree) is used to construct a classifier $C(x, T)$ for predicting future y values.
(a classifier guesses one class in $\{1, \ldots, J\}$)

Assume training data consists of iid observations from the distribution of (X, y).

def Misclassification Error

$$PE[C(\cdot, T)] = P_{X,Y}[C(X, T) \neq Y]$$

$$PE(C) = E_T[PE[C(\cdot, T)]]$$

* Misclassification error is the probability, over the "true" dist. of all X and y (including unseen), that C classifies incorrectly.
The 2nd one: averaged over the randomness of generating a training dataset T

def $P(j|x) = P(Y=j|X=x)$
$P(dx) = P(X \in dx)$
} "true" dists of X and y

def Bayes optimal classifier $C^*(x) = \underset{j}{\text{argmax}}\ P(j|x)$

def avg. class probability estimates implied by classifier C is

$$Q(j|x) = P_T(C(x,T) = j)$$

Note: $Q(j|x) \approx \frac{1}{M}\sum_{m=1}^{M} \mathbb{1}\{C(x,T^{(m)}) = j\}$

if $T^{(1)}, \ldots, T^{(M)} \overset{iid}{\sim} P$ "true" dist of (X,y)

* Avg. Implied class prob. estimate Q is the prob. that, averaged over the randomness of generating a training dataset T, that our classifier C classifies x as j

def aggregated classifier

$$C_A(x) = \underset{j}{\text{argmax}}\ Q(j|x) = \underset{j}{\text{argmax}}\ P_T(C(x,T) = j)$$

Def $C(x,\bullet)$ is unbiased at x, denoted $x \in U$, if $C^*(x) = C_A(x)$

Def Biased set $B = U^c$

Def $\text{Bias}(C) = P_{X,Y}[C^*(X)=Y, X\in B] - E_T P_{X,Y}[C(X,T)=Y, X\in B]$

Def $\text{var}(C) = P_{X,Y}[C^*(X)=Y, X\in U] - E_T P_{X,Y}[C(X,T)=Y, X\in U]$

Def $PE(C^*) = P_{X,Y}[C^*(X)\neq Y]$

Thm Bias Variance Decomposition $PE(C) = PE(C^*) + \text{Bias}(C) + \text{Var}(C)$

* $C(x,\bullet)$ is unbiased at x $(x\in U)$ if over the Replications of T as $T_1,\dots,T_n$, $C(x,\bullet)$ picks $C^*(x)$ more than it picks any other class

* If $x\in B$, $C_A(x) \neq C^*(x)$, so over the replications of T as $T_1,\dots,T_n$, $C(x,\bullet) \neq C^*(x)$ more often than not, so the bias measures by how much these differ on average (averaged over the randomness of the training set)

* If $x\in U$, $C_A(x) = C^*(x)$, so over the Replications of T as $T_1,\dots,T_n$, $C(x,\bullet)$ picks $C^*(x)$ more often than it picks any other class, but Noise in the training set could make the probability of correct classification less than the optimal value, so the variance measures by how much these differ on average

* $PE(C^*)$ = fundamental classification error, even for optimal classifier

* Bias of classifier C:

Probability that X is in the biased set and optimal classifier classifies correctly minus expected probability (over training sets) that X is biased and our classifier C classifies correctly

* variance of classifier C:

probability that X is in the unbiased set $[C_A(x) = C^*(x)]$ and optimal classifier classifies correctly minus expected probability (over training sets) that X is unbiased and our classifier C classifies correctly

$$VAR(C_A) = 0$$

Facts

1. Bias and variance are always ≥ 0
2. $VAR(C_A) = 0$
3. If C is deterministic (doesn't depend on T), then its variance is 0
4. Bias of C^* is 0

Pf 2,3 for $x \in U$, $C_A(x) = C^*(x)$

Pf 4 Trivial

Pf 1 HW (or see paper) → use:

Prop Bayes optimal classifier $C^*(x)$ has minimum misclassification Rate

$$PE(C^*) = 1 - \int \left[\max_j P(j|x)\right] P(dx)$$

Pf $PE(C^*) = E_{P_{X,Y}}\left[C^*(X,Y) \neq Y\right] = E\left[1 - P_{X,Y}\left[Y = C^*(x)\right]\right]$

$$= 1 - \int_X P_Y(Y = C^*(x)) \cdot P(dx)$$

$$= 1 - \int_x \left(\max_j P(j|x)\right) \cdot P(dx)$$

∎

* Bagging (bootstrap aggregating) Reduces variance!

* Bagging: obtain $\{T^{(B)}\}$ each by sampling from T with replacement,

Bootstrap dist. $P^{(B)} \approx$ true dist P on training samples,

and so $\{C_A(x, T^{(B)})\} = C_A(x, P^{(B)}) \approx C_A(x, P)$

Bootstrap aggregated classifier

"true" aggregated classifier

$C(x, \bullet)$ is unbiased at x, denoted $x \in U$, if $C^*(x) = C_A(x)$

$$\text{var}(C) = P_{X,Y}\left[C^*(X) = Y, X \in U\right] - E_T P_{X,Y}\left[C(X,T) = Y, X \in U\right]$$

* Bagging lowers variance because,
instead of just one decision tree $C(x,T)$ from our 1 observed training set T,
we have an aggregation of $\mathcal{B}$ trees,

$$\tilde{C}_A(x,T) = \frac{1}{\mathcal{B}} \sum_{B=1}^{\mathcal{B}} C(x, T^{(B)})$$

where $T^{(B)} \overset{\text{boot}}{\sim} T$.

$C_A(x,T) = \frac{1}{M} \sum_{m=1}^{M} C(x, T^{(m)}) \approx \tilde{C}_A(x,T)$ since bootstrap works...

Recall $\text{VAR}(C_A) = 0$. So (hope) $\text{VAR}(\tilde{C}_A) \approx 0$ and, bias remains similar...

* Empirically, via simulated data, Arcing (boosting) also reduces variance (and it reduces bias), but it is less clear HOW...

Bias Variance Decomposition

$$PE(C) = PE(C^*) + Bias(C) + Var(C)$$

Pf $PE(C) = \mathbb{E}\left[PE(C(\cdot, T))\right]$

$= \mathbb{E}_T P_{X,Y}\left[C(X,T) \neq Y\right]$

$= 1 - \mathbb{E}_T P_{X,Y}\left[C(X,T) = Y\right]$

$= 1 - \mathbb{E}_T\left(P_{X,Y}\left[C(X,T) = Y, X \in U\right] + P_{X,Y}\left[C(X,T) = Y, X \in B\right]\right)$

$= Bias(C) - \left(P_{X,Y}\left[C^*(X) = Y, X \in B\right] - \mathbb{E}_T P_{X,Y}\left[C(X,T) = Y, X \in B\right]\right)$

$+ Var(C) - \left(P_{X,Y}\left[C^*(X) = Y, X \in U\right] - \mathbb{E}_T P_{X,Y}\left[C(X,T) = Y, X \in U\right]\right)$

$+ 1 - \mathbb{E}_T\left(P_{X,Y}\left[C(X,T) = Y, X \in U\right] + P_{X,Y}\left[C(X,T) = Y, X \in B\right]\right)$

$= Bias(C) + Var(C) + \left(1 - P_{X,Y}\left[C^*(X) = Y\right]\right)$

$= Bias(C) + Var(C) + PE(C^*)$ ∎

- $Bias(C)$: error from misclassification due to our classifier model C being a wrong or bad model itself
- $Var(C)$: error from misclassification by C due to training set noise
- $PE(C^*)$: fundamental misclassification, even by the optimal classifier

Random Forest

Input: training data $T = (X, y)$

Hyperparameters: n, m

for $B = 1, \ldots, \mathcal{B}$:

1. Bootstrap (Row subsampling): sample n of the N training data instances with replacement
2. Column subsampling: sample m of the M features

This yields a resampled training set $T^{(B)}$

3. Fit one decision tree $\hat{f}^{(B)}$ from $T^{(B)}$

Aggregate: $\hat{f} = \frac{1}{\mathcal{B}} \sum_{B=1}^{\mathcal{B}} \hat{f}^{(B)}$

* Bootstrapping + Feature Subsampling and Aggregating Reduces Variance and hence improves predictive performance! This stuff works...

Note the Random Forest aggregation procedure,

e.g. $\widehat{WP}(x)$ = proportion of fitted bootstrapped trees which classify game-state x as 1 (win)

isn't a-priori guaranteed to be a calibrated probability...
Why should this produce a good prob.?
See Olson Wyner 2018.

Hyperparameters to Increase the Predictive Power

n_estimators: Number of trees the algorithm builds before averaging the predictions.

max_features: Maximum number of features random forest considers splitting a node.

mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

criterion: How to split the node in each tree? (Entropy/Gini impurity/Log Loss)

max_leaf_nodes: Maximum leaf nodes in each tree

Hyperparameters to Increase the Speed

n_jobs: it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor, but if the value is -1, there is no limit.

random_state: controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and has been given the same hyperparameters and training data.

oob_score: *OOB* means out of the bag. It is a random forest cross-validation method. In this, one-third of the sample is not used to train the data; instead used to evaluate its performance. These samples are called out-of-bag samples.

* HW: fit a random forest NFL win prob. model. How much better is it than a single decision tree?

* Next time: Boosting (XGBoost)