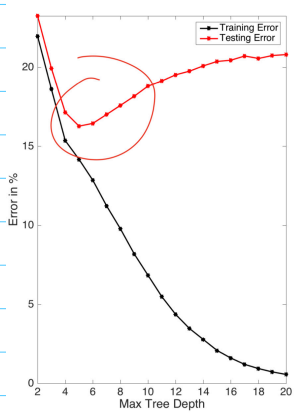
Bagging: Bootstrap Aggregating

1. As long as each sample in the training set has unique features, Decision tree can always obtain 0 training loss
2. If depth of Decision tree is shallow it underfits
3. If we go all out on depth it overfits badly to training data
4. Depth is not a fine grained knob for good Bias-Variance Tradeoff



1. Ensemble methods: combine multiple models to improve performance
2. Boosting: Works to reduce bias (improve underfitting)
3. Bagging: works to reduce variance (improve overfitting)

Eg: Decision trees to max depth will highly overfit

Recall Bias Variance Decomposition:

$$E[(h_D(x)-y)^2] = \underbrace{E_{(x,y)\sim D}[(h_D(x)-\bar{h}(x))^2]}_{\text{variance}} + \underbrace{E[(\bar{h}(x)-\bar{y}(x))^2]}_{\text{Bias}} + \underbrace{E[(\bar{y}(x)-y)^2]}_{\text{inherent noise}}$$

Our Goal is to reduce the variance.

Say we could draw m datasets of size n $D_1, D_2, \ldots, D_m$ drawn iid from P
Run algorithm m times to get $h_{D1}, h_{D2}, \ldots, h_{Dm}$ set

$$\hat{h} = \frac{1}{m}\sum_{i=1}^{m} h_{Di}$$

What is the variance of h ?

Quiz: $X_1, \ldots, X_m \overset{iid}{\sim} P$ s.t. $\text{var}(X_i) = \sigma^2$ Say $E[X] = 0$

what is the variance of $\hat{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$?

what if $X_1, \ldots, X_m$ are correlated ?

Weak law of large numbers $\hat{h} \to \bar{h}$

Why cant we use the above m sample method to reduce variance?

Ans:

Bootstrapping: Simulate m samples $D_1, D_2, \ldots, D_m$ as follows:

1. Take original training sample D
2. Create size n data set D_i by drawing repeatedly with replacement elements from sample D uniformly at random
3. Train $h_{D_1}, \ldots, h_{D_m}$ on each of these m datasets
4. Return the average hypothesis:

$$\hat{h}_{Bag} = \frac{1}{m} \sum_{i=1}^{m} h_{D_i}$$

Illustration:

Why does Bagging work?

Say Q was the distribution over samples given by the following procedure:

1. Draw D of size n iid from P
2. Draw samples from D uniformly at random with replacement
3. While draw of samples from Q are not iid, we can show that marginal distribution of samples matches P

Claim: Say $\Omega = \{x_1, \ldots, x_N\}$ (for simplicity assume finite space)

$\forall x_i \in \Omega \quad Q(x = x_i) = P(x = x_i)$

Proof:

$$Q(X = x_i) = \underbrace{\sum_{k=1}^{n} \binom{n}{k} p_i^k (1-p_i)^{n-k}}_{\text{Probability that are k copies of } x_i \text{ in D}} \underbrace{\frac{k}{n}}_{\text{Probability pick one of these copies}}$$

$$= \frac{1}{n} \underbrace{\sum_{k=1}^{n} \binom{n}{k} p_i^k (1-p_i)^{n-k} k}_{\substack{\text{Expected value of Binomial Distribution with parameter } p_i \\ \mathbb{E}[\mathbb{B}(p_i,n)]=np_i}}$$

$$= \frac{1}{n} n p_i$$

$$= p_i \leftarrow \underline{TATAAA}!! \text{ Each data set } D_l' \text{ is drawn from P, but not independently}$$

$P(X = x_i) = p_i$

Advantages of Bagging

- Easy to implement
- Reduces variance, so has a strong beneficial effect on high variance classifiers.
- As the prediction is an average of many classifiers, you obtain a mean score *and variance*. Latter can be interpreted as the uncertainty of the prediction. Especially in regression tasks, such uncertainties are otherwise hard to obtain. For example, imagine the prediction of a house price is $300,000. If a buyer wants to decide how much to offer, it would be very valuable to know if this prediction has standard deviation +-$10,000 or +-$50,000.

- Bagging provides an unbiased estimate of the test error, which we refer to as the *out-of-bag error*. The idea is that each training point was not picked and all the data sets D_k. If we average the classifiers h_k of all such data sets, we obtain a classifier (with a slightly smaller m) that was not trained on $(\mathbf{x}_i, y_i)$ ever and it is therefore equivalent to a test sample. If we compute the error of all these classifiers, we obtain an estimate of the true test error. The beauty is that we can do this without reducing the training set. We just run bagging as it is intended and obtain this so called out-of-bag error for free.

  More formally, for each training point $(\mathbf{x}_i, y_i) \in D$ let $S_i = \{k | (\mathbf{x}_i, y_i) \notin D_k\}$ - in other words S_i is a set of all the training sets D_k, which do not contain $(\mathbf{x}_k, y_k)$. Let the averaged classifier over all these data sets be

$$\tilde{h}_i(\mathbf{x}) = \frac{1}{|S_i|} \sum_{k \in S_i} h_k(\mathbf{x}).$$

  The-of-bag error becomes simply the average error/loss that all these classifiers yield

$$\epsilon_{\text{OOB}} = \frac{1}{n} \sum_{(\mathbf{x}_i, y_i) \in D} l(\tilde{h}_i(\mathbf{x_i}), y_i).$$

This is an estimate of the test error, because for each training point we used the subset of classifiers that never saw that training point during training. if m is sufficiently large, the fact that we take out some classifiers has no significant effect and the estimate is pretty reliable.

Random Forest

One of the most famous and useful bagged algorithms is the **Random Forest**! A Random Forest is essentially nothing else but bagged decision trees, with a slightly modified splitting criteria.

The algorithm works as follows:

1. Sample m data sets $D_1, \ldots, D_m$ from D with replacement.
2. For each D_j train a full decision tree $h_j()$ (max-depth=∞) with one small modification: before each split randomly subsample $k \leq d$ features (without replacement) and only consider these for your split. (This further increases the variance of the trees.)
3. The final classifier is $h(\mathbf{x}) = \frac{1}{m} \sum_{j=1}^{m} h_j(\mathbf{x})$.

The Random Forest is one of the best, most popular and easiest to use out-of-the-box classifier. There are two reasons for this:

- The RF only has two hyper-parameters, m and k. It is extremely *insensitive* to both of these. A good choice for k is $k = \sqrt{d}$ (where d denotes the number of features). You can set m as large as you can afford.
- Decision trees do not require a lot of preprocessing. For example, the features can be of different scale, magnitude, or slope. This can be highly advantageous in scenarios with heterogeneous data, for example the medical settings where features could be things like *blood pressure*, *age*, *gender*, ..., each of which is recorded in completely different units.