CS725 : Ensemble Models : Boosting and Bagging

Ensemble model or learner : One which combines models for improved predictions compared to each model in the ensemble

Boosting is a very popular ensembling technique

weak learner is slightly better than chance

Main idea behind boosting : Can we combine weak learners to improve performance and come up with a strong learner

E.g. Decision stump, Perceptron learner, etc

Many forms of boosting ⇒ Adaboost, gradient boosting, XGboost, etc

Template of a boosting algorithm:

0. Initialize equal weights (that sum to 1) for all the N training points
1. Train a weak learner on the weighted training data
2. Compute a weighted error based on the weak learner from 1 on the training set
3. Compute an importance estimate corresponding to the weak learner
4. Recompute weights for all the training examples using the importance estimate in 3 so as to add more wt to misclassified examples
5. Repeat steps 2 to 4 until stopping criterion is met

AdaBoost (Freund & Schapire, 1995)

$D_t(i)$ is normalized to be a prob distribution

Given $\{x_i, y_i\}_{i=1}^n$, $y_i \in \{-1, 1\}$, initial weights $D_1(i) = \frac{1}{n} \ \forall i$

for round $t = 1, \dots, T$ → hyperparameter

Step 1: Train a weak learner on the trainset weighted by $D_t(i)$

Call it $h_t(x_i) = \{-1, 1\}$

Step 2: Compute the weighted error with using $h_t(x_i)$

$$\epsilon_t = \sum_{i=1}^{n} D_t(i) \, \mathbb{1}\left[h_t(x_i) \neq y_i\right]$$

→ INDICATOR FUNCTION

Step 3: Compute an importance estimate α_t associated with the weak learner $h_t(x)$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

for rounds $t = 1 \ldots T$

Step 4 ; Recompute the weights of the training examples

$$D_{t+1}(x_i) \propto \begin{cases} D_t(x_i)\exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ D_t(x_i)\exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$$

downweighting examples that $h_t(x)$ gets correct

upweighting misclassified examples

$$D_{t+1}(x_i) = \frac{D_t(x_i)\exp(-\alpha_t y_i h_t(x_i))}{\sum_{j=1}^{n} D_t(x_j)\exp(-\alpha_t y_j h_t(x_j))}$$

Final classifier after boosting for a test instance x: $H(x) = \underbrace{\text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)}_{\text{Ensemble classifier weighted by importance estimates } \alpha_t}$

Adaboost minimizes the following exponential loss: $L_{boost} = \frac{1}{n} \sum_i \exp(-y_i H(x_i))$

If I want to add a new weak predictor h to the final classifier i.e., $H \leftarrow H + \alpha h$ the optimal α which minimizes L_{boost} is $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

Bagging (Bootstrap Aggregating)

Consider $D_1, D_2, \ldots, D_m$ independent datasets sampled i.i.d. from an underlying distribution P.

Say we train a predictor to classify an x based on $D_i \Rightarrow h(x; D_i)$

An average predictor would just be $\frac{1}{m}\sum_{i=1}^{m} h(x; D_i)$

What can we say about the following for the average predictor?

1. BIAS ?

Bias : $\mathbb{E}_{D_1,\dots,D_m}\left[\frac{1}{m}\sum_i h(x;D_i)\right]$

$$= \frac{1}{m}\sum_i \mathbb{E}_{D_i}\left[h(x;D_i)\right]$$

$$= \mathbb{E}_D\left[h(x;D)\right]$$

UNCHANGED

② VARIANCE?

$$\operatorname{Var}\left[\frac{1}{m}\sum_{i=1}^{m} h(x;D_i)\right]$$

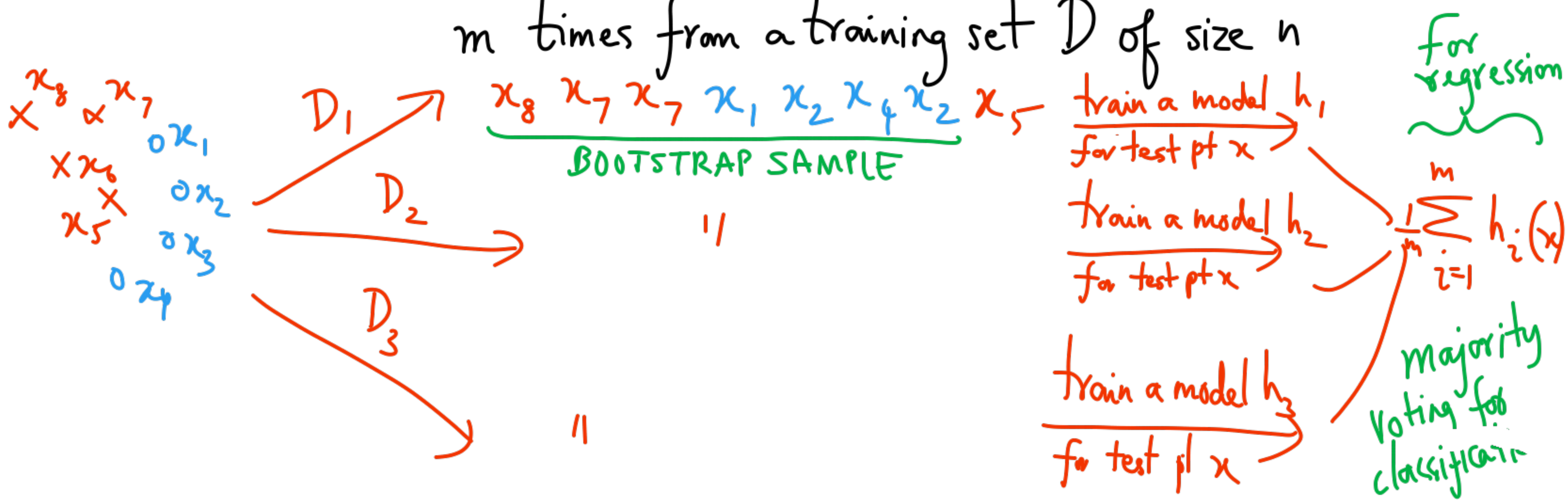
$$= \frac{1}{m^2}\sum_{i=1}^{m} \operatorname{Var}\left[h(x;D_i)\right]$$

$$= \frac{1}{m}\operatorname{Var}\left[h(x;D)\right]$$

REDUCES by a factor of m!

Catch is that we typically do not have access to independent datasets $D_1, \dots, D_m$. Bootstrap Aggregating $\Rightarrow$ <u>BAGGING</u>

<u>Solution</u>: Bootstrap! Sample at random with replacement m times from a training set D of size n



The bootstrap samples are not independent of each other. But the overall variance of the bagged classifier reduces.

Key point: In an ensemble, we want decorrelated predictions for the ensemble to generalize well

Very popular bagging technique is the RANDOM FOREST classifier

Random forests $\Rightarrow$ Decision trees + Bagging + an additional tweak

① Bootstrap from the original dataset D, $|D| = n$, and train DTs one for each bootstrap sample

② Tweak is each DT only uses a subset of features at each split [of size $k \approx \sqrt{d}$]
where d is the size of the original feature space

Consider a bootstrap sample. Any training instance that is not part of the bootstrap sample is called an OOB (out-of-bag) sample.

OOB samples can be used as a bagged estimate of a test error without using an explicit test set. <u>OOB error estimate</u>

How?

1. For every i^{th} training instance, predict labels from every bootstrap model for which this instance was OOB
2. Compute the mean (or maj vote) across these predictions & compute its error
3. Average errors across all the training instances