LMP 1210 - Basic principles of machine learning in biomedical research

Rahul G. Krishnan

Canada CIFAR AI Chair

Tier II Canada Research Chair
in Computational Medicine

Announcements

- Taking over (mostly) from Sana
- Assignment 2 out – Due next week [Feb 15]
- Final project proposal due Feb 22
  - Please look at updated handout!
  - Please have teams formed by then (form them in the break today!)

Recap: What we have learned so far?

KNN

Decision Trees

Linear Models

MLP

What you can do if you are a doctor now

Evaluation and recommendation → Treatment

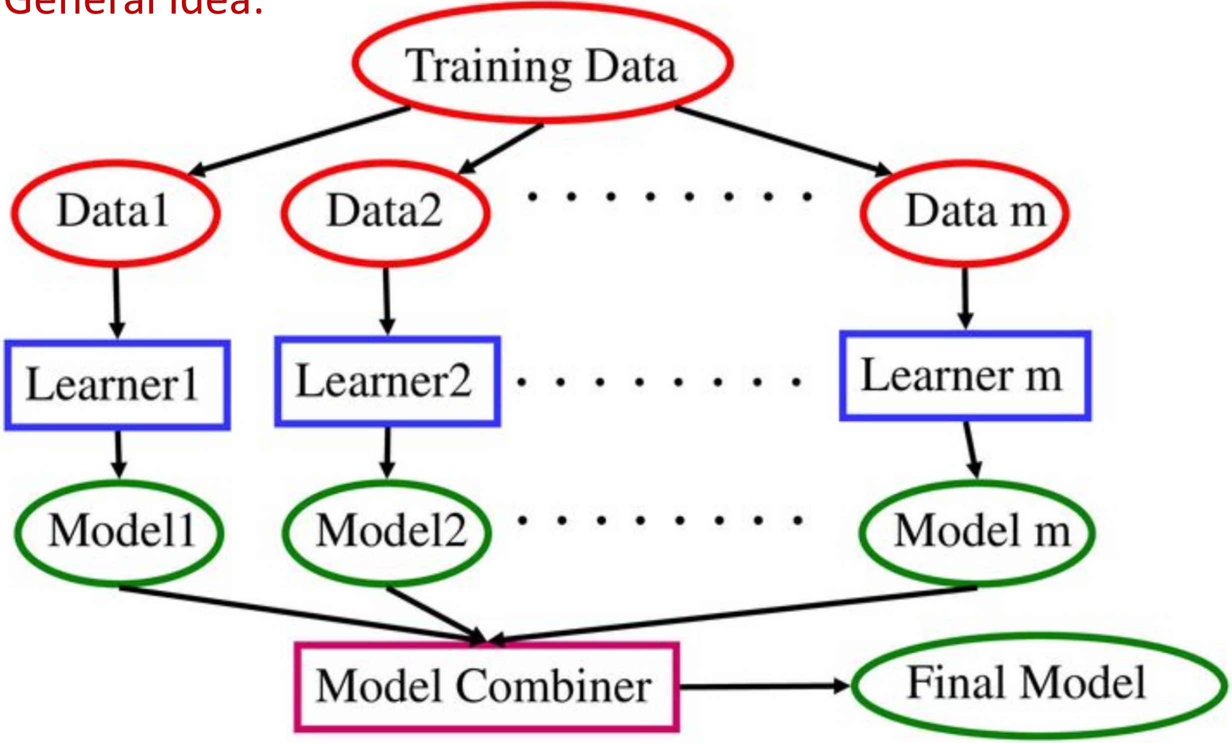
KNN,
Decision Trees,
Linear Models,
MLP

What you can do after today:
Ensemble Methods

Evaluation and recommendation
Evaluation and recommendation
Treatment
Evaluation and recommendation
Evaluation and recommendation

What are Ensemble Methods?

General Idea:



What are Ensemble Methods?

- An ensemble of predictors is a set of predictors whose individual decisions are combined in some way to predict new examples, for example by (weighted) majority vote.
- For the result to be nontrivial, the learned hypotheses must differ somehow, for example because of
  - Different algorithms
  - Different choices of hyperparameters
  - Trained on different data sets
  - Trained with different weighting of the training examples
- Ensembles are usually easy to implement. The hard part is deciding what kind of ensemble you want, based on your goals.
- Two major types of ensembles methods:
  - Bagging
  - Boosting

Why Ensemble Methods

- Based on one of two basic observations:
  1. Variance reduction: if the training sets are completely independent, it will always help to average an ensemble because this will reduce variance without affecting bias (e.g., bagging)
     - reduce sensitivity to individual data points
  2. Bias reduction: for simple models, average of models has much greater capacity than single model (e.g., hyperplane classifiers, Gaussian densities).
     - Averaging models can reduce bias substantially by increasing capacity, and control variance by fitting one component at a time (e.g., boosting)

Why Ensemble Methods

- Ensemble methods more accurate than any individual members if:
  - Accurate (better than guessing)
  - Diverse (different errors on new examples)
- Why?
- Independent errors: prob k of N classifiers (independent error rate ϵ) wrong:

$$P(\text{num errors} = k) = \binom{N}{k} \epsilon^k (1 - \epsilon)^{N-k}$$

- Probability that majority vote wrong: error under distribution where more than $N/2$ wrong

Why Ensemble Methods

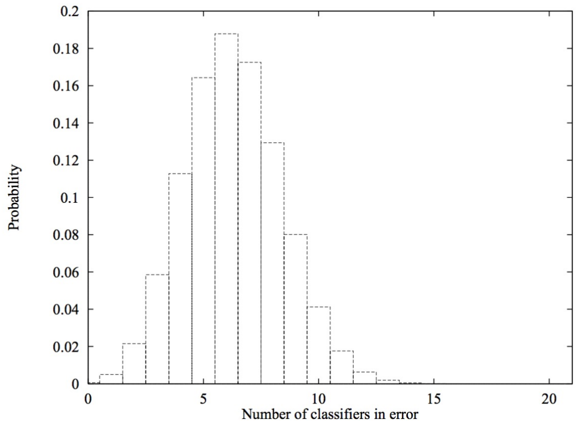


Figure : Example: The probability that exactly K (out of 21) classifiers will make an error assuming each classifier has an error rate of $\epsilon = 0.3$ and makes its errors independently of the other classifier. The area under the curve for 11 or more classifiers being simultaneously wrong is 0.026 (much less than ϵ).

[Credit: T. G Dietterich, Ensemble Methods in Machine Learning]

Why Ensemble Methods

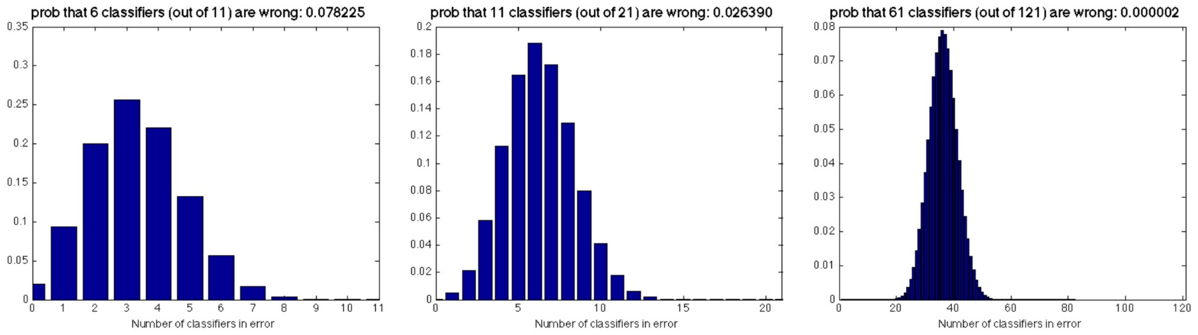


Figure : $\epsilon = 0.3$: **(left)** $N = 11$ classifiers, **(middle)** $N = 21$, **(right)** $N = 121$.

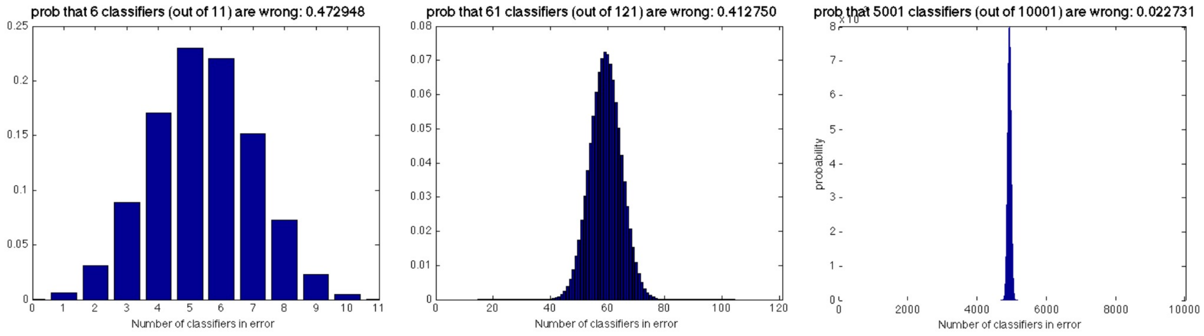
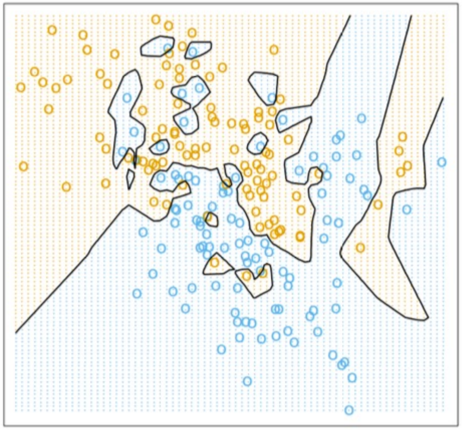


Figure : $\epsilon = 0.49$: **(left)** $N = 11$, **(middle)** $N = 121$, **(right)** $N = 10001$.

Why Ensemble Methods

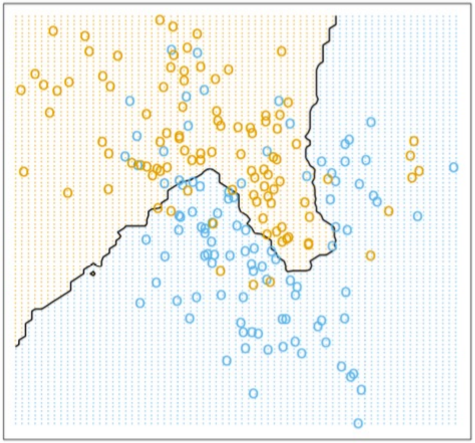
- Minimize two sets of errors:
    1. Variance: error from sensitivity to small fluctuations in the training set
    2. Bias: erroneous assumptions in the model
- Variance-bias decomposition is a way of analyzing the generalization error as a sum of 3 terms: variance, bias and irreducible error (resulting from the problem itself)

Variance-Bias Trade-off



K = 1

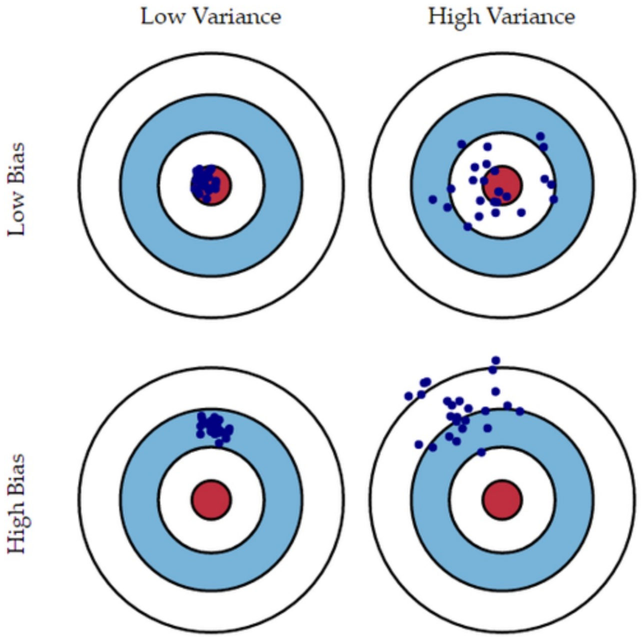
High Variance
Low Bias



K = 15

Low Variance
High Bias

Variance-Bias Trade-off



Why Ensemble Methods

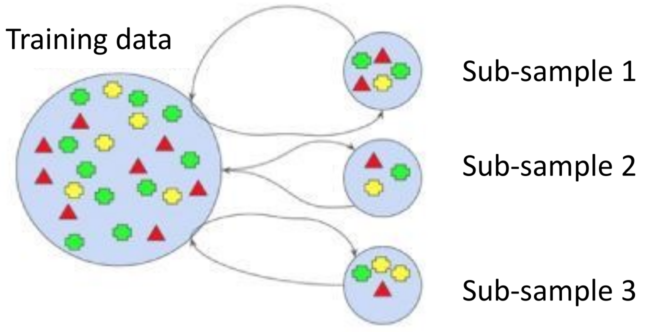
- Clear demonstration of the power of ensemble methods
- Original progress prize winner (BellKor) was ensemble of 107 models!
  - *"Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a simple technique."*
  - *"We strongly believe that the success of an ensemble approach depends on the ability of its various predictors to expose different complementing aspects of the data. Experience shows that this is very different than optimizing the accuracy of each individual predictor."*

Ensemble Methods

- Differ in training strategy, and combination method
  - Parallel training with different training sets
    1. Bagging (bootstrap aggregation) – train separate models on overlapping training sets, average their predictions
  - Sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples: boosting

Bootstrap Estimation

- Repeatedly draw n samples from D
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance
- Bagging: bootstrap aggregation (Breiman 1994)



The 0.632 bootstrap

- A particular training data has a probability of 1-1/*n* of *not* being picked
- Thus its probability of ending up in the test data (not selected) is:

$$\left(1-\frac{1}{n}\right)^{n} \approx e^{-1} = 0.368$$

- This means the training data will contain approximately 63.2% of the instances

Bagging

- Simple idea: generate M bootstrap samples from your original training set. Train on each one to get y_m, and average them

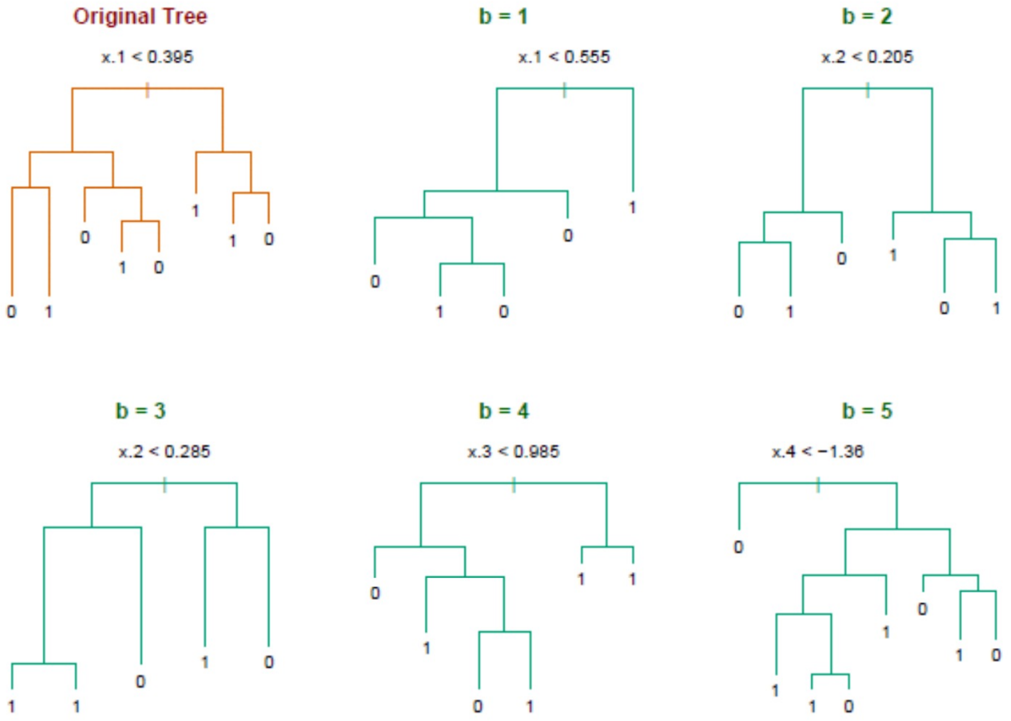
$$y_{bag}^{M}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x})$$

- For regression: average predictions
- For classification: average class probabilities (or take the majority vote if only hard outputs available)
- Bagging approximates the Bayesian posterior mean. The more bootstraps the better, so use as many as you have time for
- Each bootstrap sample is drawn with replacement, so each one contains some duplicates of certain training points and leaves out other training points completely

Bagging

- Reduces overfitting (variance)
- Normally uses one type of classifier
- Decision trees are popular
- Easy to parallelize

Bagging Decision Trees



Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction", Springer (2009)

Bagging Decision Trees: Pitfalls

Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors.

Then all bagged trees will select the strong predictor at the top of the tree and therefore all trees will look similar.

In other words: Correlated Trees!

How do we avoid this?

Bagging Decision Trees: Pitfalls

Remember we want i.i.d such as the bias to be the same and variance to be less?

Other ideas?

What if we consider only a subset of the predictors at each split?

We will still get correlated trees unless ....

we **randomly** select the subset !

Random Forests

Random Forests

```
For b = 1 to B:
    (a) Draw a bootstrap sample Z∗ of size $N$ from the training data.
    (b) Grow a random-forest tree to the bootstrapped data, by
recursively repeating the following steps for each terminal node of the
tree, until the minimum node size $n_{min}$ is reached.
            i. Select $m$ variables at random from the $p$ variables.
            ii. Pick the best variable/split-point among the $m$.
            iii. Split the node into two daughter nodes.
Output: the ensemble of trees.
```

To make a prediction at a new point *x* we do:

For regression: average the results

For classification: majority vote

Tuning Random Forests

The inventors make the following recommendations:

- For classification, the default value for *m* is $\sqrt{p}$ and the minimum node size is one.
- For regression, the default value for m is *p/3* and the minimum node size is five.

In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Random Forests : Summary

Advantages

- Random forest algorithm is unbiased as there are multiple trees and each tree is trained on a subset of data.
- Random Forest algorithm is very stable. Introducing a new data in the dataset does not affect much as the new data impacts one tree and is pretty hard to impact all the trees.
- The random forest algorithm works well when you have both categorical and numerical features.
- With missing values in the dataset, the random forest algorithm performs very well.

Disadvantages

- A major disadvantage of random forests lies in their complexity. More computational resources are required and also results in the large number of decision trees joined together.
- Due to their complexity, training time is more compared to other algorithms.

Random Forests: Pitfalls

When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when *m* is small

Why?

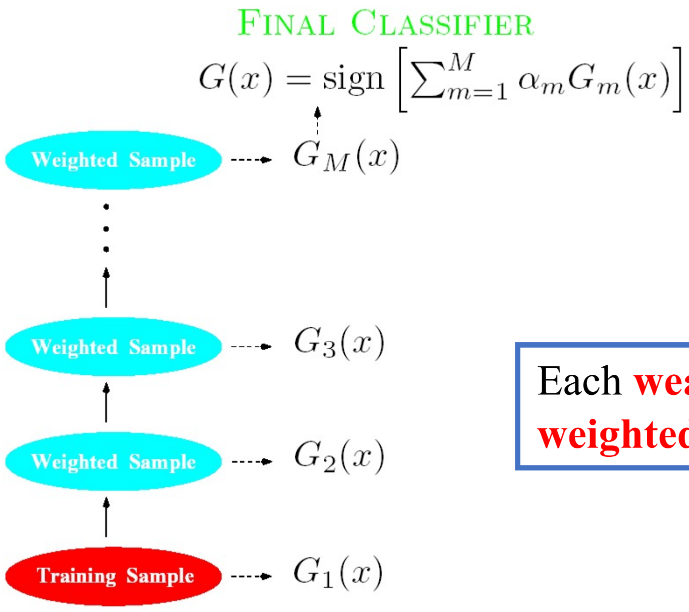
Because:

At each split the chance can be small that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Another Way of Ensemble: Boosting

Sequential



Each **weak classifier** is trained from a **weighted sample** of the training data

Boosting: Weighted Sample

- The misclassification rate $\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ weights each training example equally.
- Key idea: We can learn a classifier using different costs (aka weights) for examples.
  - Classifier "tries harder" on examples with higher cost
- Change cost function:

$$\sum_{n=1}^{N}\frac{1}{N}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \sum_{n=1}^{N} w^{(n)}\mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

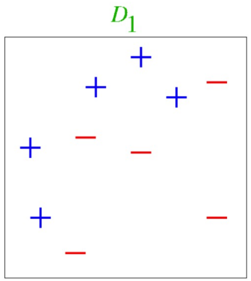
- Usually require each $w^{(n)} > 0$ and $\sum_{n=1}^{N} w^{(n)} = 1$

Boosting: Weak Learners

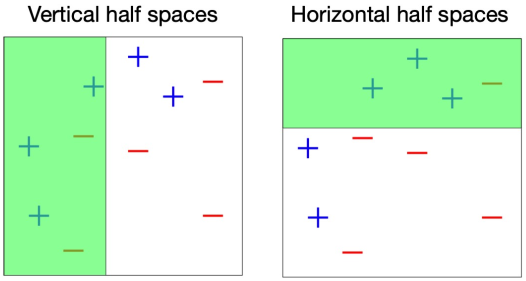
- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (i.e., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.51 in binary label case.
  - It gets slightly less than 0.5 error rate (the worst case is 0.5)
- We are interested in weak learners that are *computationally* efficient.
  - Decision trees
  - Even simpler: Decision Stump: A decision tree with a single split

[Formal definition of weak learnability has quantifiers such as “for any distribution over data” and the requirement that its guarantee holds only probabilistically.]

Boosting: Weak Learners



These weak classifiers, which are decision stumps, consist of the set of horizontal and vertical half spaces.



AdaBoost: Adaptive Boosting

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- Key steps of AdaBoost:
  1. At each iteration we re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
  2. We train a new weak classifier based on the re-weighted samples.
  3. We add this weak classifier to the ensemble of weak classifiers. This ensemble is our new classifier.
  4. We repeat the process many times.
- The weak learner needs to minimize weighted error.
- AdaBoost reduces bias by making each classifier focus on previous mistakes.

AdaBoost: Adaptive Boosting

- Input: Data $\mathcal{D}_N$, weak classifier WeakLearn (a classification procedure that returns a classifier h, e.g., best decision stump, from a set of classifiers $\mathcal{H}$, e.g., all possible decision stumps), number of iterations T
- Output: Classifier $H(x)$
- Initialize sample weights: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \text{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

$$h_t \leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{n=1}^{N} w^{(n)} \mathbb{I}\{h(\mathbf{x}^{(n)}) \neq t^{(n)}\}$$

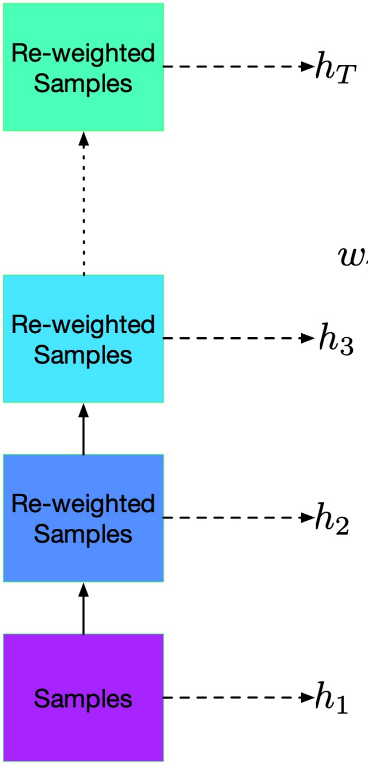
  - Compute weighted error $\text{err}_t = \frac{\sum_{n=1}^{N} w^{(n)} \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^{N} w^{(n)}}$
  - Compute classifier coefficient $\alpha_t = \frac{1}{2} \log \frac{1 - \text{err}_t}{\text{err}_t}$ $\quad (\in (0, \infty))$
  - Update data weights

$$w^{(n)} \leftarrow w^{(n)} \exp\left(-\alpha_t t^{(n)} h_t(\mathbf{x}^{(n)})\right) \left[\equiv w^{(n)} \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}\right)\right]$$

- Return $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

$\boxed{\text{Don't Panic! See next!}}$

AdaBoost: Adaptive Boosting



$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Still Panic?
See next!

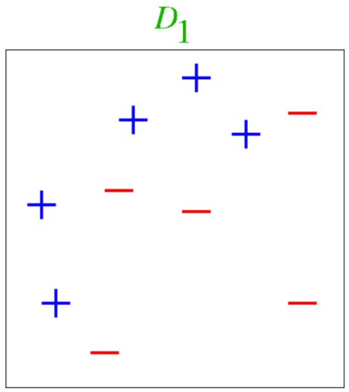
$$w_i \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x^{(i)}}) \neq \mathbf{t^{(i)}}\}\right)$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\mathrm{err}_t}{\mathrm{err}_t}\right)$$

$$\mathrm{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x^{(i)}} \neq \mathbf{t^{(i)}}\}}{\sum_{i=1}^{N} w_i}$$

Boosting: A simple example

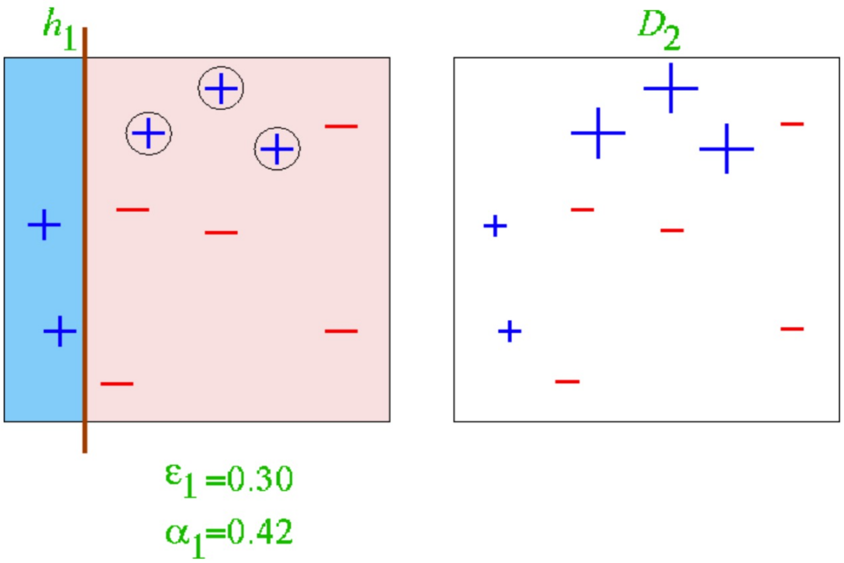
- Training data



[Slide credit: Verma & Thrun]

Boosting: A simple example

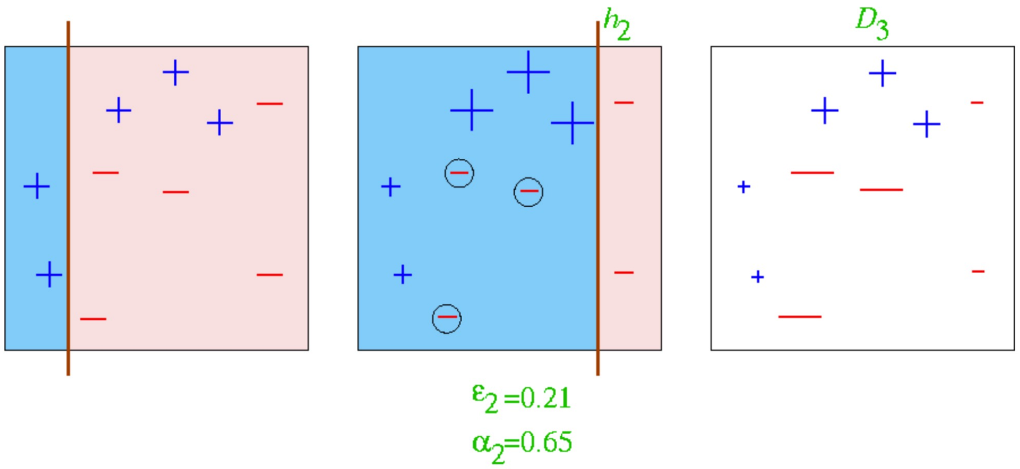
- Round 1



[Slide credit: Verma & Thrun]

Boosting: A simple example

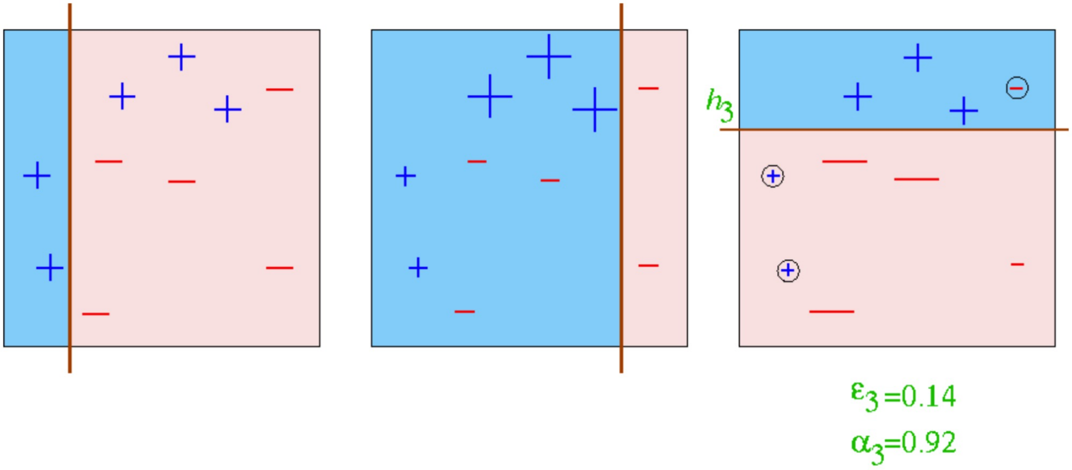
- Round 2



[Slide credit: Verma & Thrun]

Boosting: A simple example

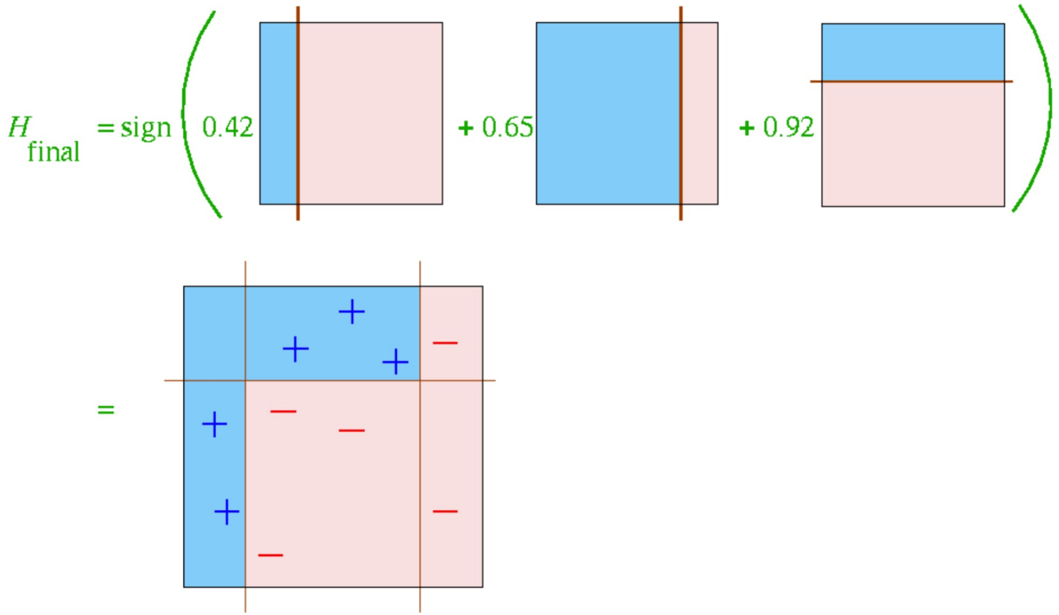
- Round 3



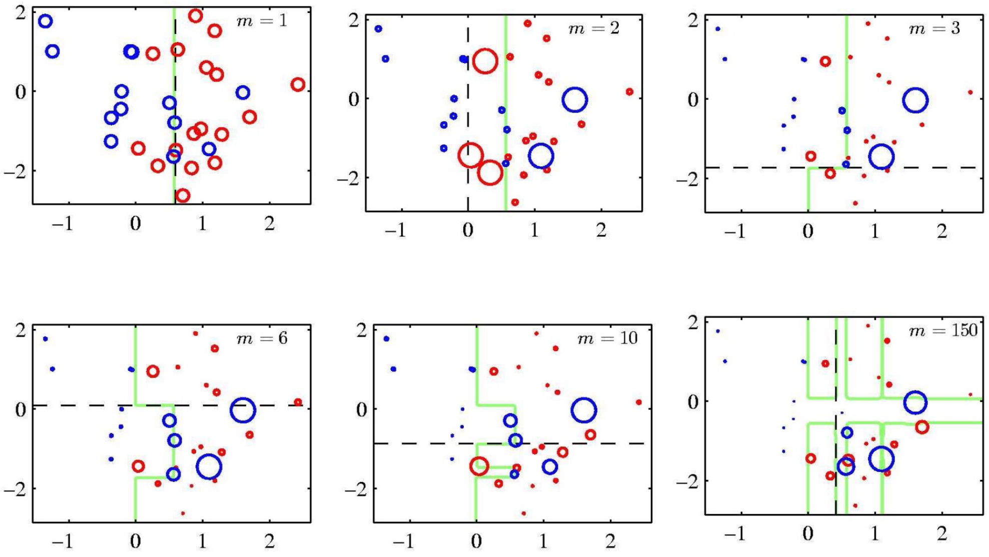
[Slide credit: Verma & Thrun]

Boosting: A simple example

- Final classifier


$H_{\text{final}} = \text{sign}\big(0.42$ + 0.65 + 0.92 $\big)$
=

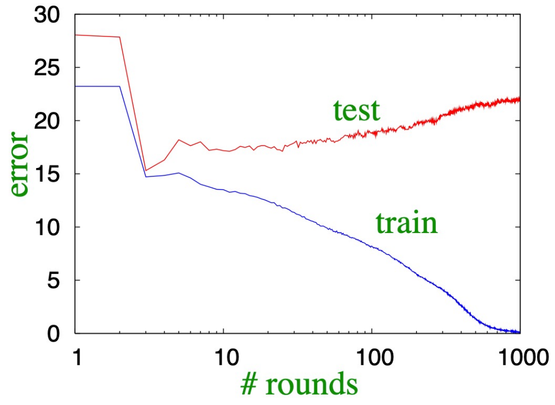
Boosting: A simple example



- Each figure shows the number *m* of base learners trained so far, the decision of the most recent learner (dashed black), and the boundary of the ensemble (green)

AdaBoost: Generalization

- AdaBoost's training error (loss) converges to zero. What about the test error of H?
- As we add more weak classifiers, the overall classifier H becomes more "complex".
- We expect more complex classifiers overfit.
- If one runs AdaBoost long enough, it can in fact overfit.



Other Boosting:
Gradient Boosting, **XGBoost**

- Gradient Boosting uses regression trees as weak learners.
- Gradient Boosting uses additive losses and gradient descent to solve the optimization problem.
- XGBoost is a regularized version of gradient boosting with more efficient implementation.
- XGBoost, by far, is the most widely used boosting algorithm.

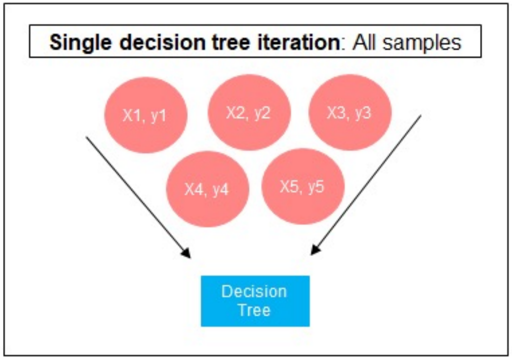
Boost: Summary

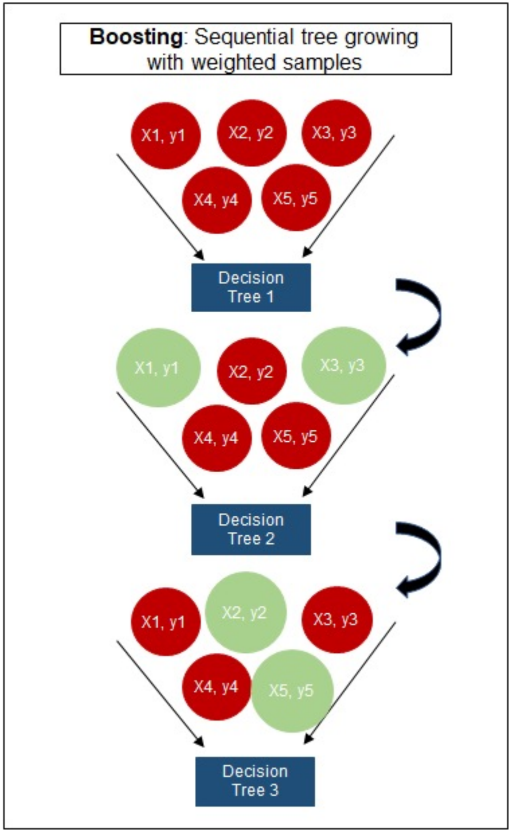
- Boosting reduces bias by generating an ensemble of weak classifiers.
- Each classifier is trained to reduce errors of previous ensemble.
- It is quite resilient to overfitting, though it can overfit.

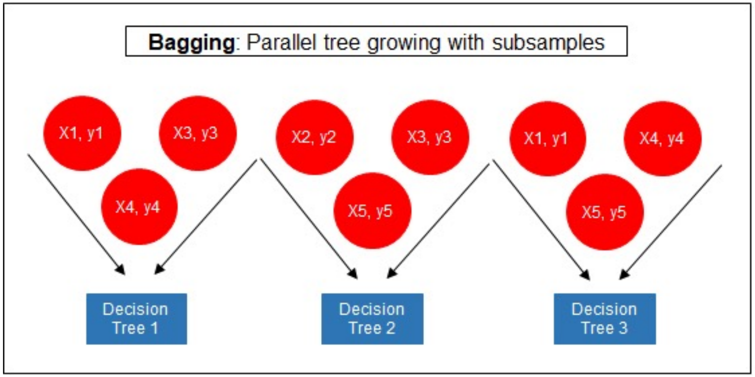
Ensemble Methods: Summary

- Ensembles combine classifiers to improve performance
- Boosting
  - Reduces bias
  - Increases variance (large ensemble can cause overfitting)
  - Sequential
  - High dependency between ensemble elements
- Bagging
  - Reduces variance (large ensemble can't cause overfitting)
  - Bias is not changed (much)
  - Parallel
  - Want to minimize correlation between ensemble elements.

Ensemble Methods: Summary

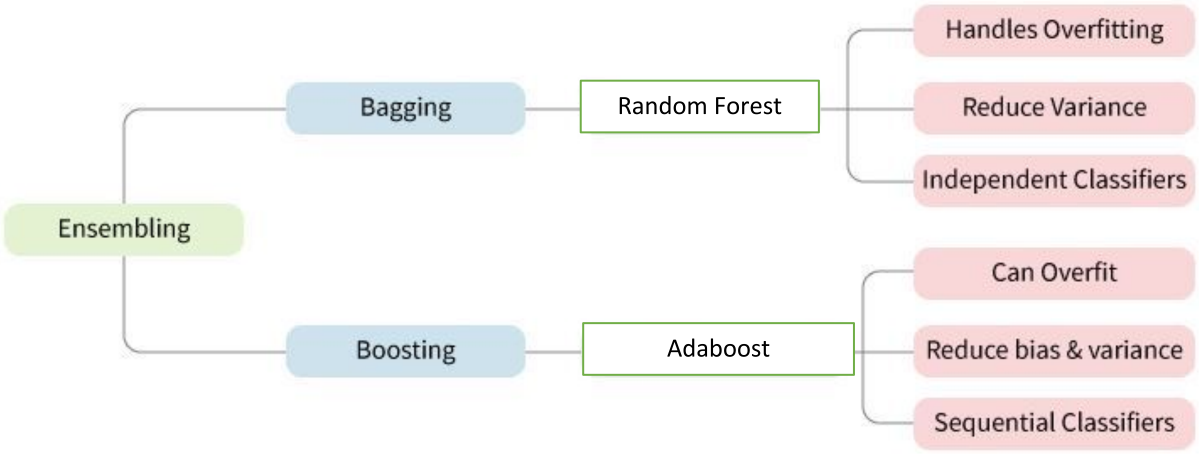






Source: https://towardsdatascience.com/the-ultimate-guide-to-adaboost-random-forests-and-xgboost-7f9327061c4f

Ensemble Methods: Summary



Which type of model would you prefer if you are a patient?