Parallel Ensembles

Framework of Ensemble

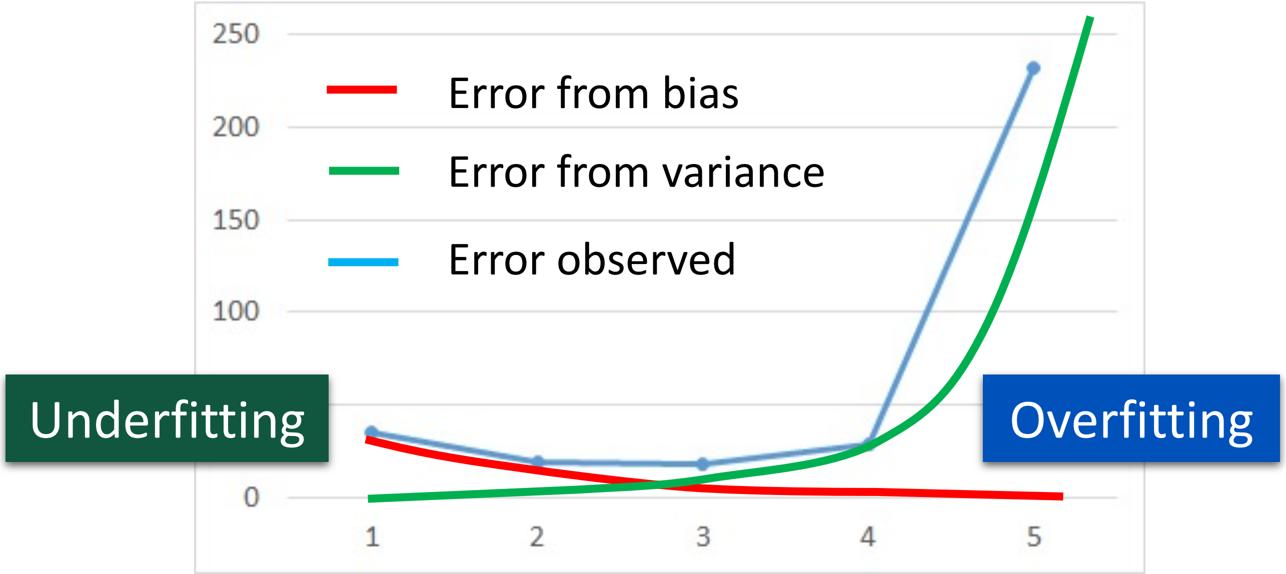
- Get a set of classifiers
  - $f_1(x), f_2(x), f_3(x), \ldots\ldots$

    They should be diverse.

- Aggregate the classifiers (*properly*)

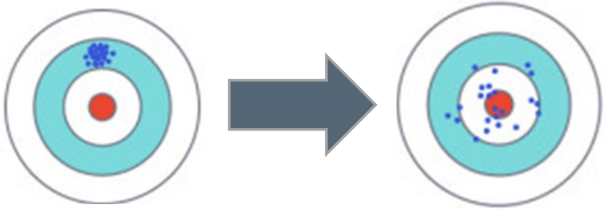
Ensemble: Bagging

Review: Bias Versus Variance



Large Bias

Small Variance

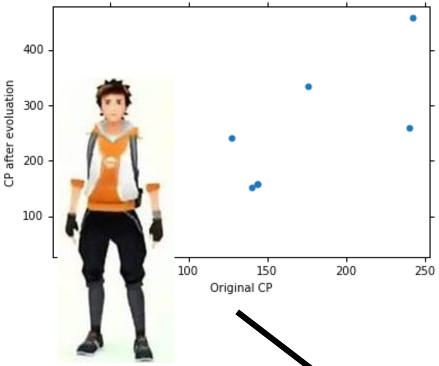


Small Bias

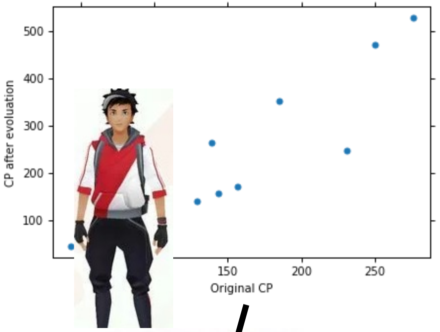
Large Variance

Review: Bias Versus Variance, Cont'd

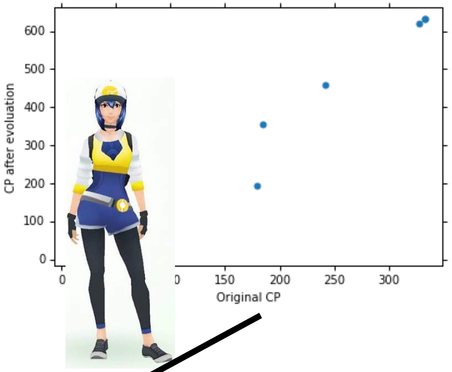
Universe 1

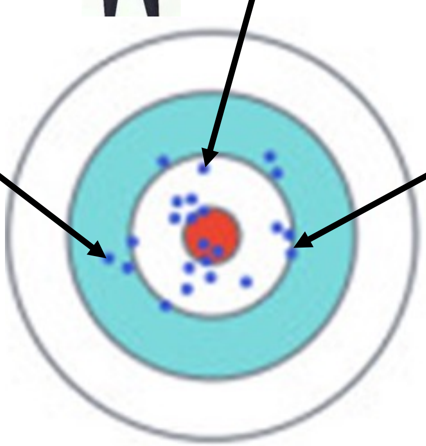


Universe 2



Universe 3





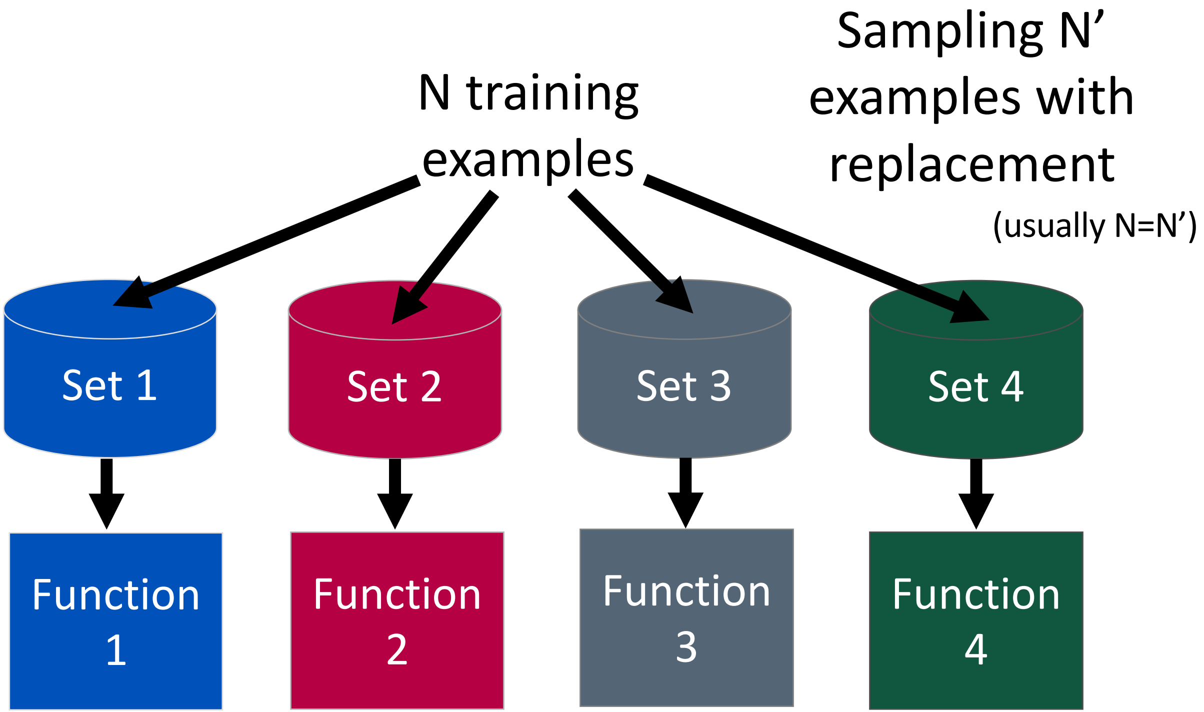
A complex model will have large variance.

We can average complex models to reduce variance.

If we average all the f^*, is it close to $\hat{f}$

$$E[f^*] = \hat{f}$$

Bagging: Part I



The Bootstrap Sample (Sampling With Replacement)

Original Data (Z)

Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Z^{*1}

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Z^{*2}

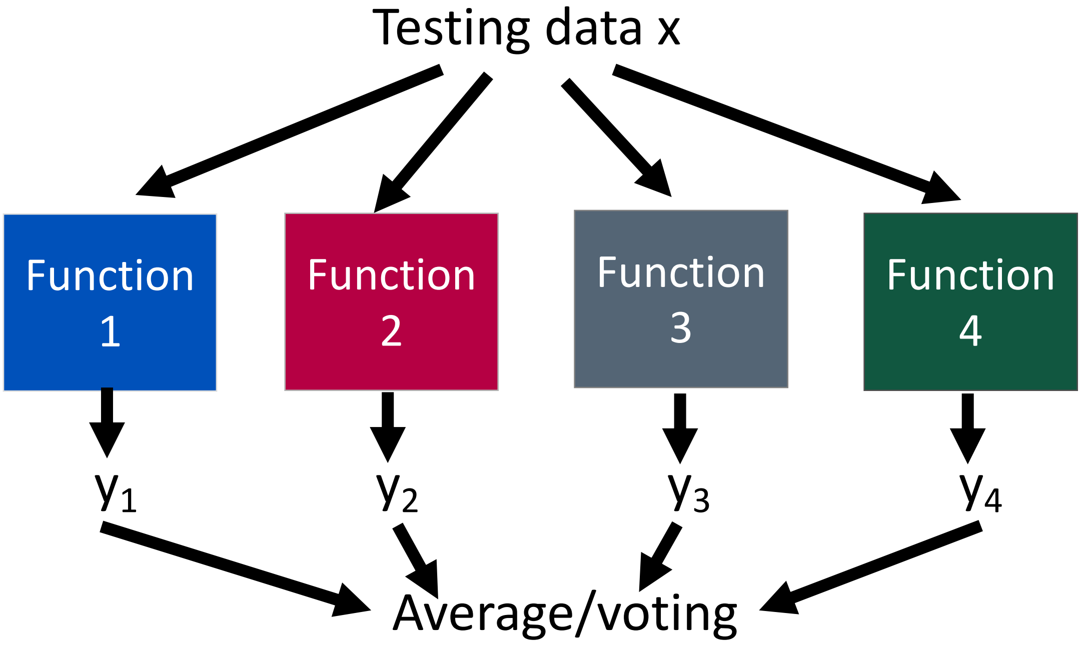
Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Z^{*B}

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

Bagging: Part II

This approach would be helpful when your model is complex, easy to overfit. For example, the following decision tree.

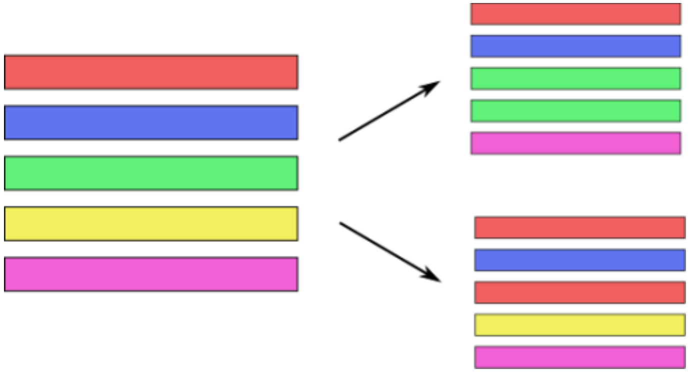


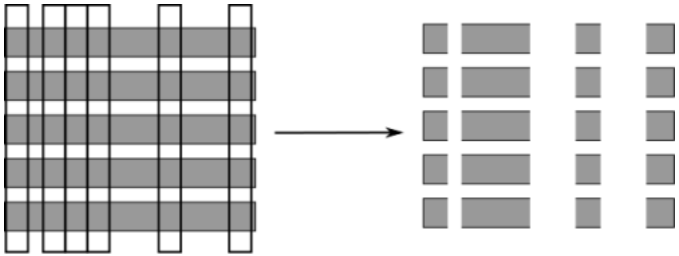
Random Forest: Part III

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ......
- Random forest: Bagging of decision tree
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split

Randomize in two Ways

- For each tree:
  - Pick bootstrap sample of data
- For each split:
  - Pick random sample of features
- More trees are always better





Random Forest: Part IV

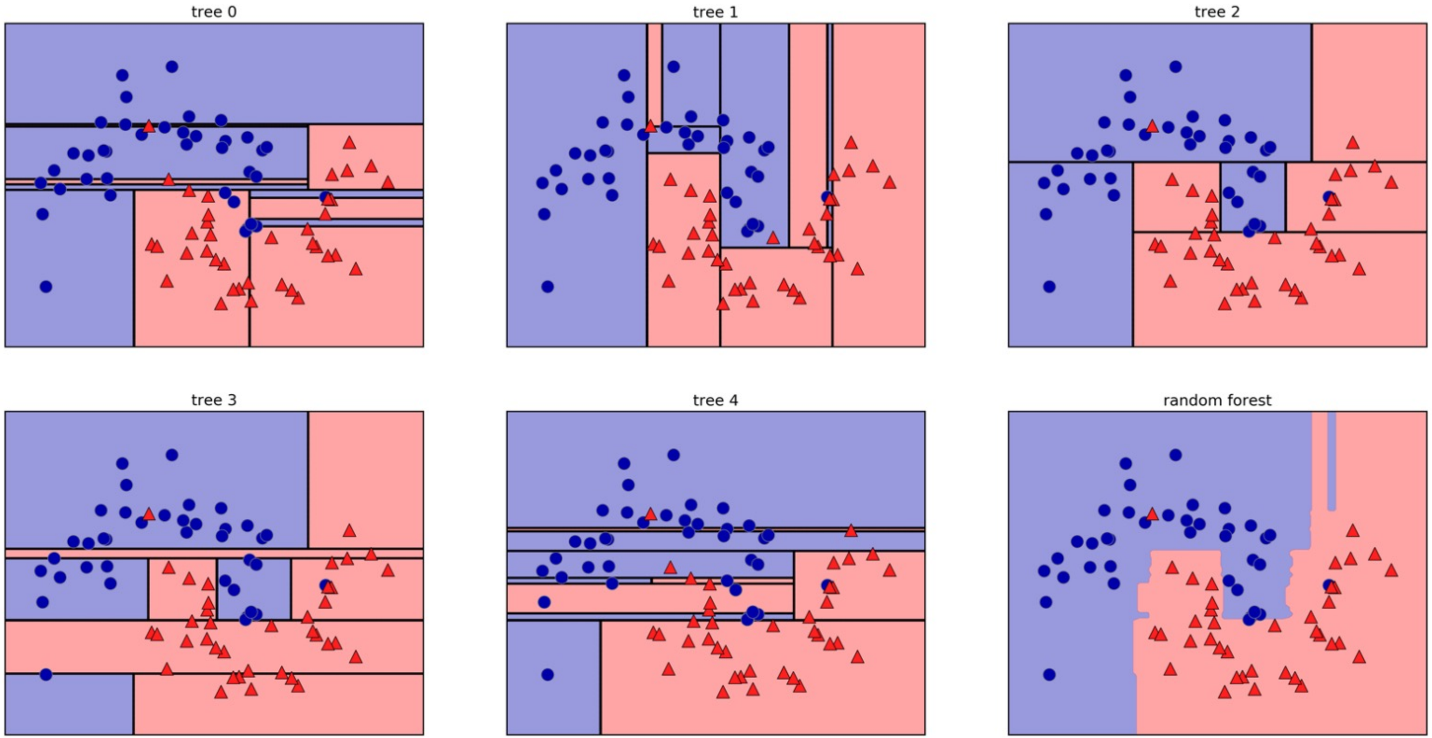
train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2
  - Using RF = f_1+f_4 to test x^3
  - Using RF = f_1+f_3 to test x^4

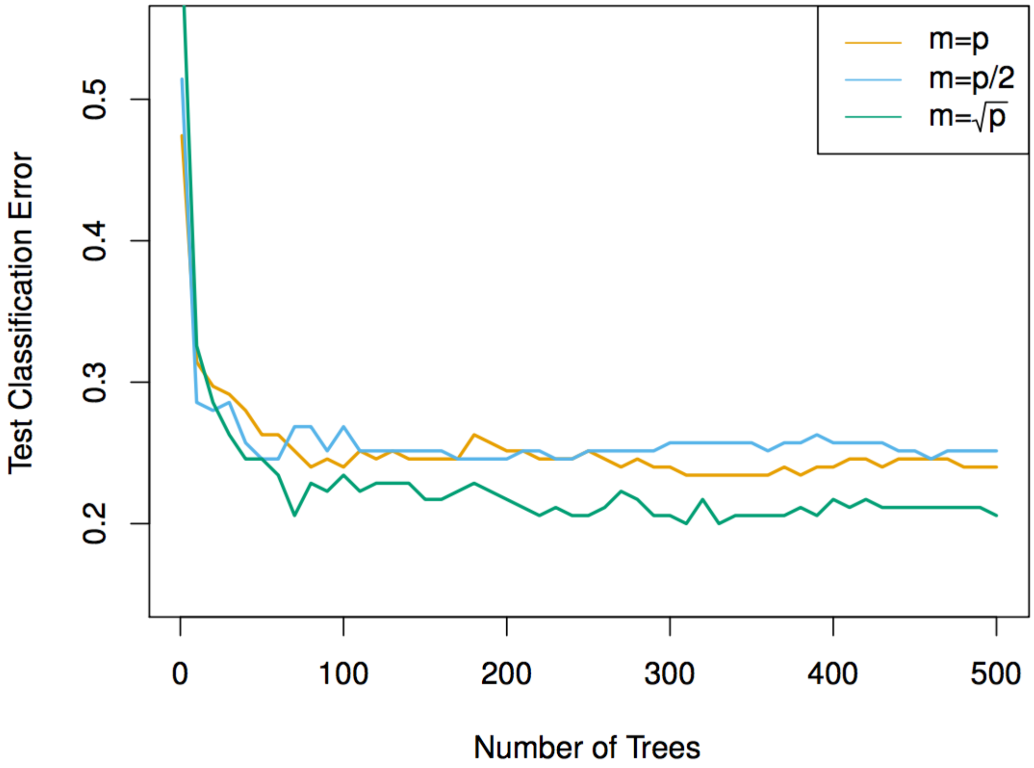
Out-of-bag (OOB) error

Good error estimation of testing set

Random Forests



Impact of max_features



Random Forest : Pros and Cons

Pros

- Widely used, excellent prediction performance on many problems.
- Doesn't require careful normalization of features or extensive parameter tuning.
- Like decision trees, handles a mixture of feature types.
- Easily parallelized across multiple CPUs.

Cons

- The resulting models are often difficult for humans to interpret.
- Like decision trees, random forests may not be a good choice for very high-dimensional tasks (e.g., text classifiers) compared to fast, accurate linear models.