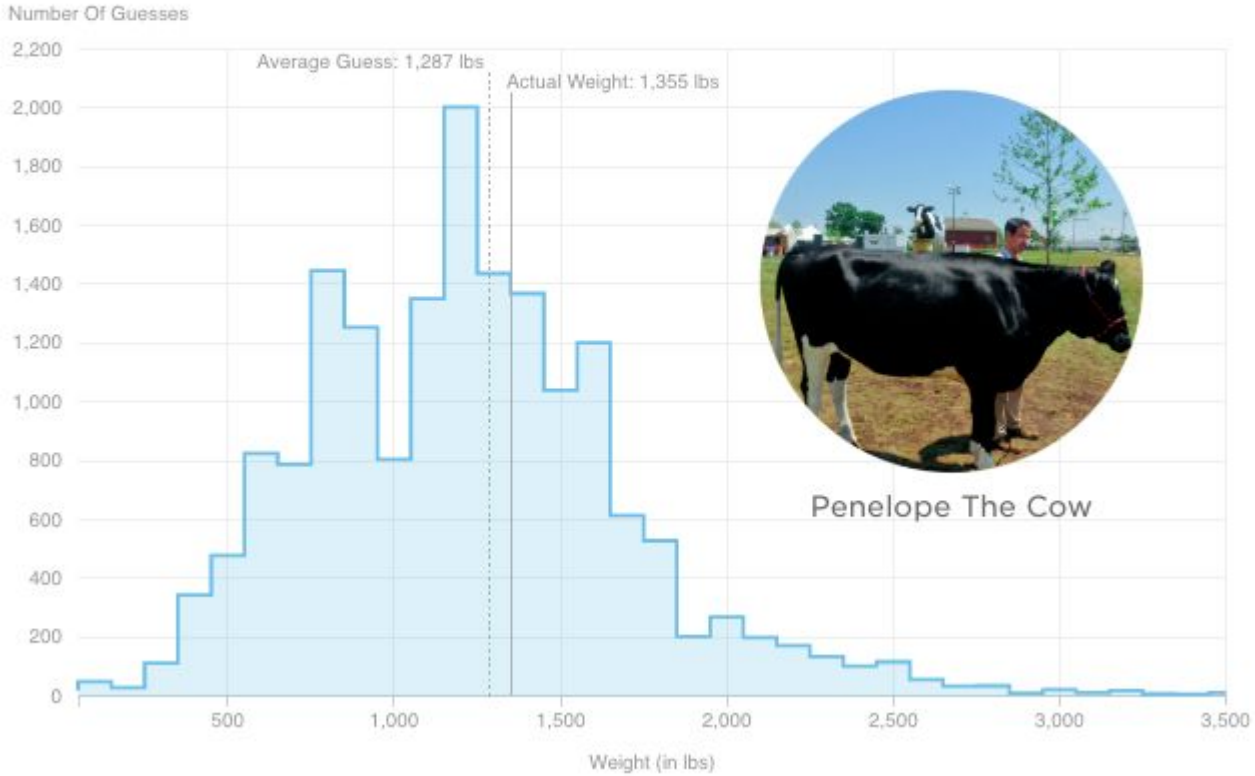
Python Machine Learning In Biology:

Ensemble Methods

Nichole Bennett

How Much Does This Cow Weigh?

(All People)



The Netflix Prize

https://www.wired.com/2009/09/how-the-netflix-prize-was-won/

What is an ensemble?

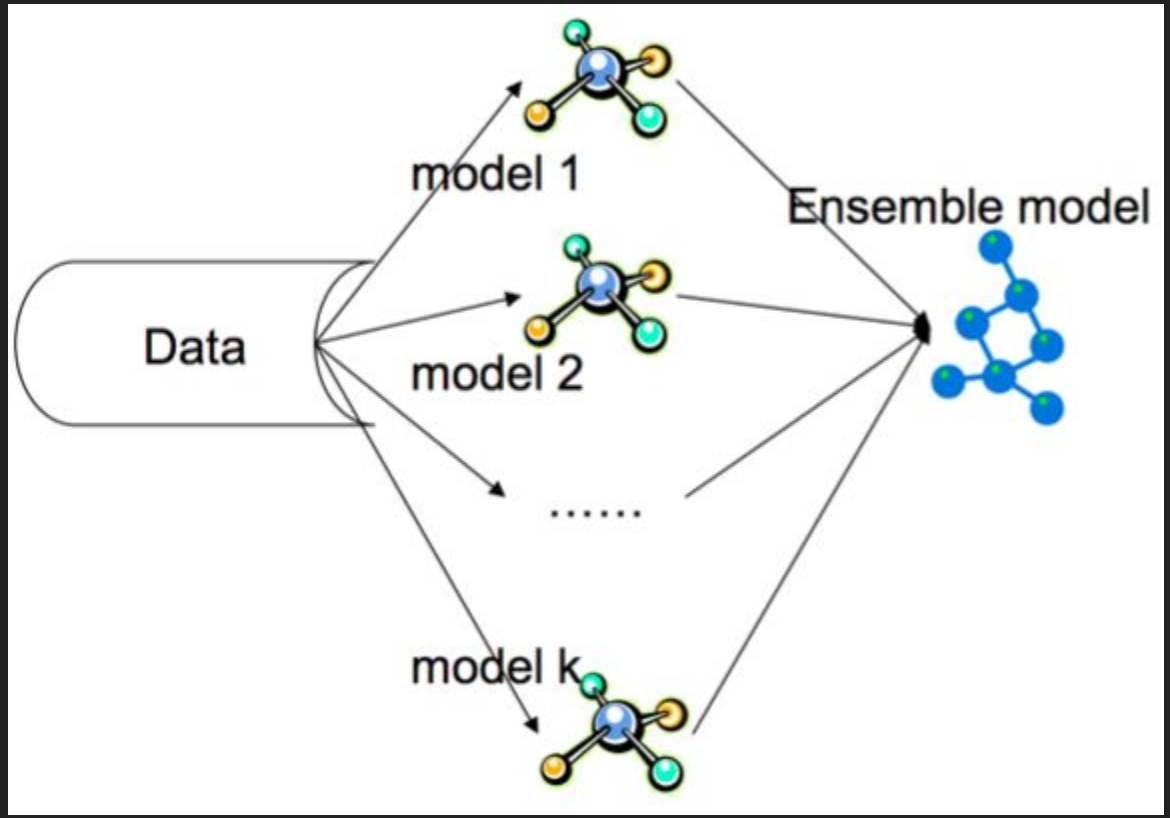
What is an ensemble?

Combine several base classifier models

Often better performance and generalizability than the base models that make them up

Disadvantage is that we lose transparency

When will this be useful?



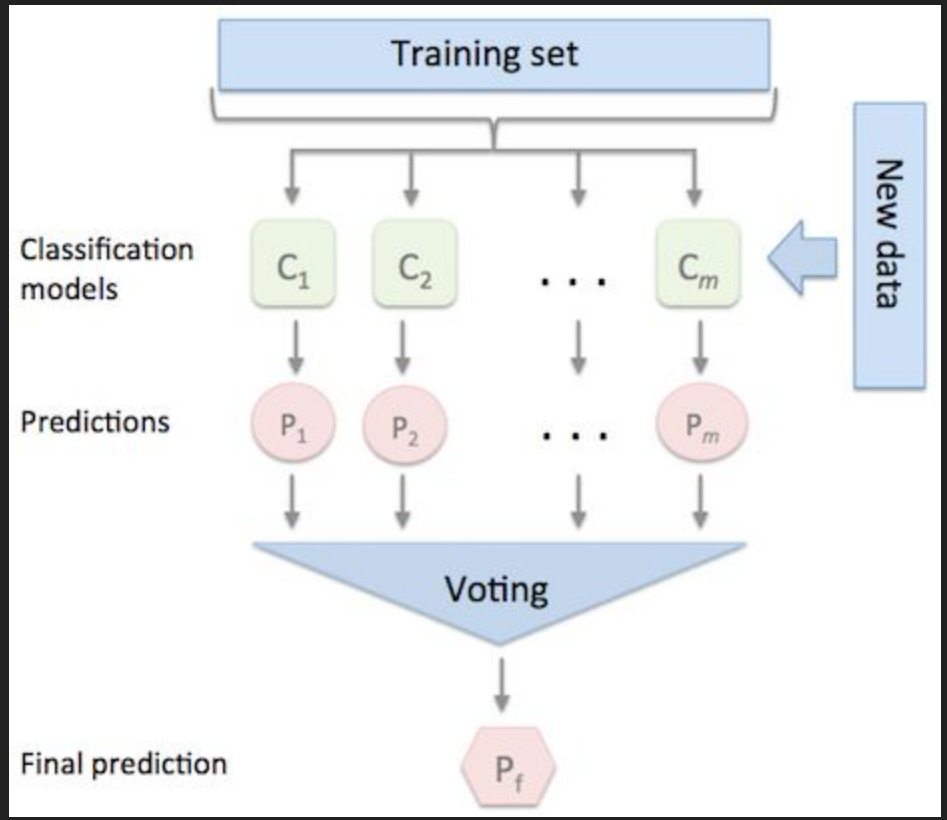
Averaging Methods

Build several models and then average their predictors

Reduces variance

- Bagging
- Random Forests

Majority voting methods



Boosting Methods

Building base models sequentially to reduce bias

- AdaBoost
- Gradient Tree Boosting

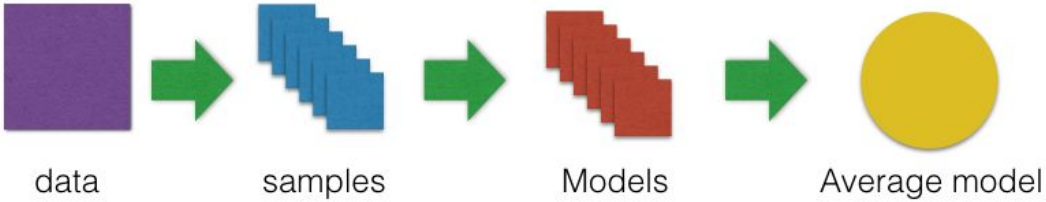
We'll come back to this in a bit

Bagging and Boosting

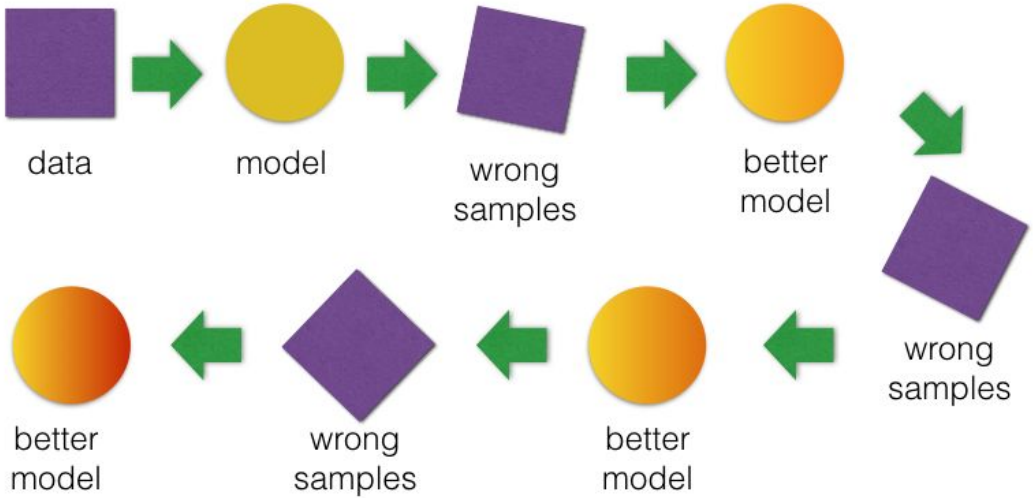
Bagging aims to reduce variance

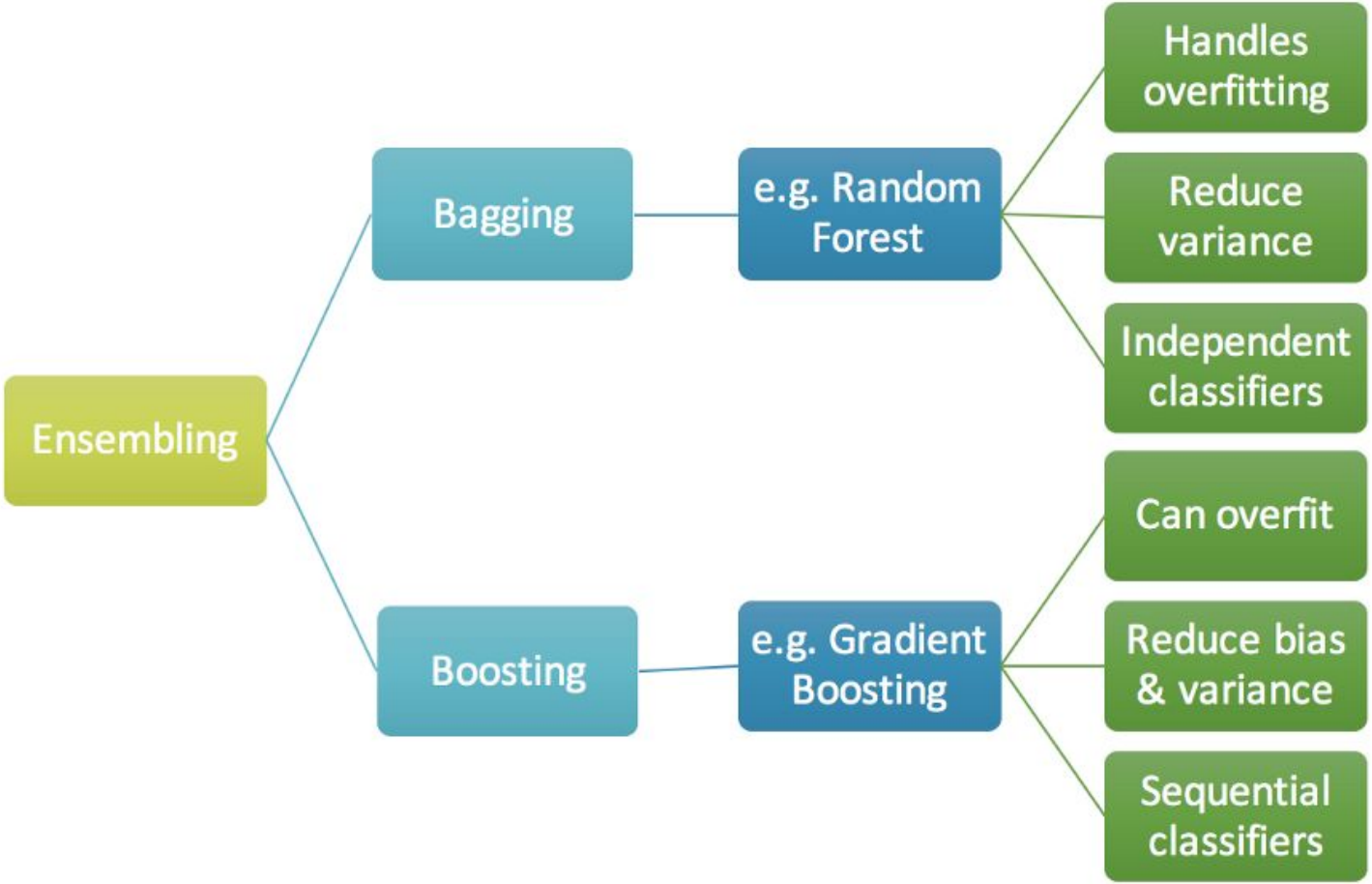
Boosting aims to reduce bias (and some variance)

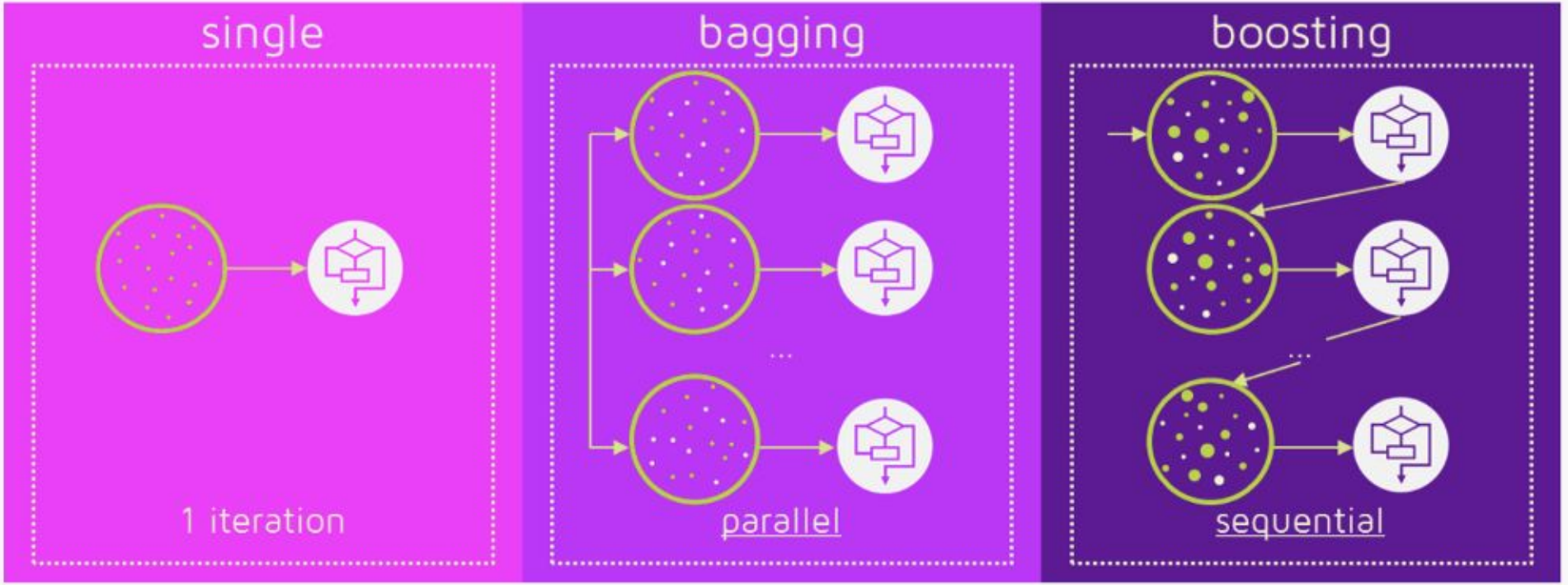
Bagging



Boosting



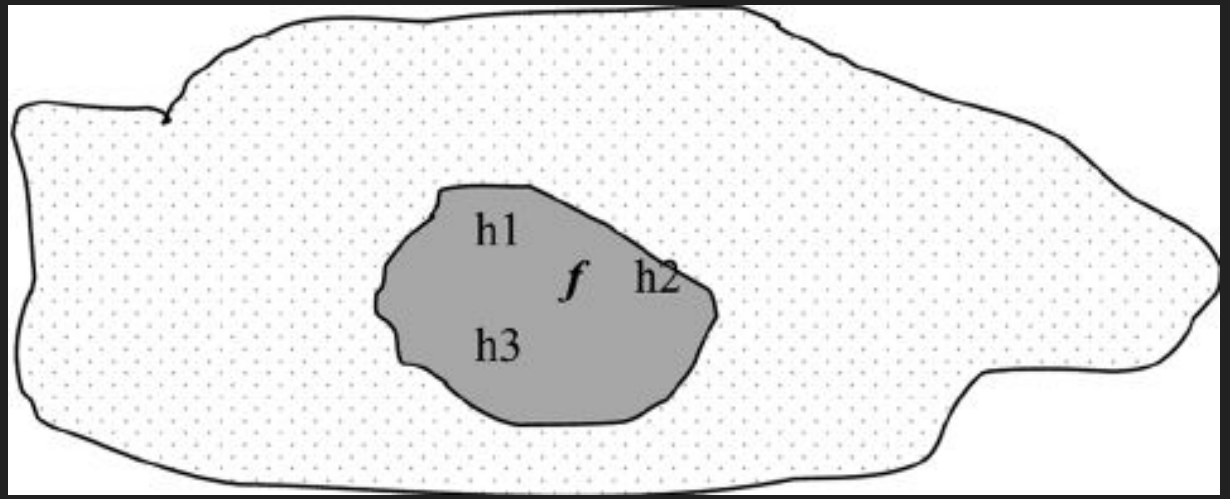

Ensembling
Bagging
Boosting
e.g. Random Forest
e.g. Gradient Boosting
Handles overfitting
Reduce variance
Independent classifiers
Can overfit
Reduce bias & variance
Sequential classifiers


single
1 iteration
bagging
parallel
boosting
sequential

Statistical Problem

If we have a limited amount of training data, classifier will have difficulty converging within our entire hypothesis space.

The true function f is found by “averaging out” the base classifiers.

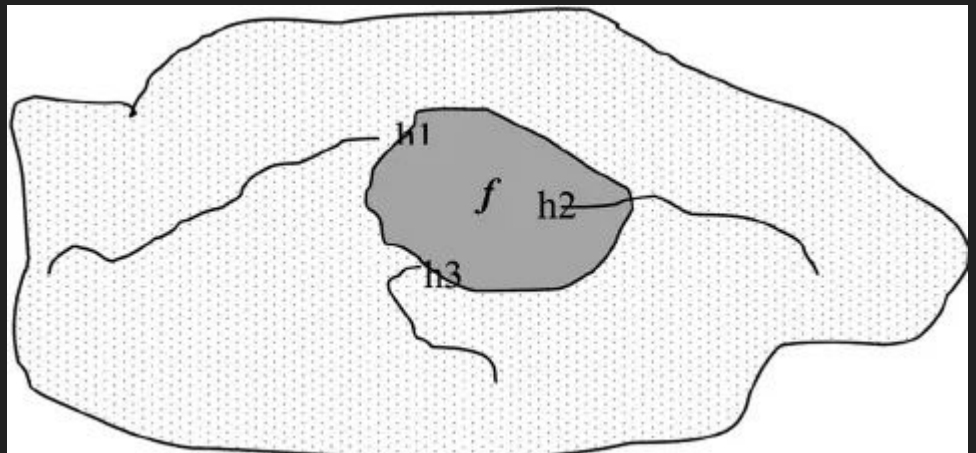


Computational Problem

Even if we have enough training data, it still may be computationally difficult to find the best classifier.

Exhaustive search of all of hypothesis space is complex.

Base classifiers with different starting points give us a better approximation to f



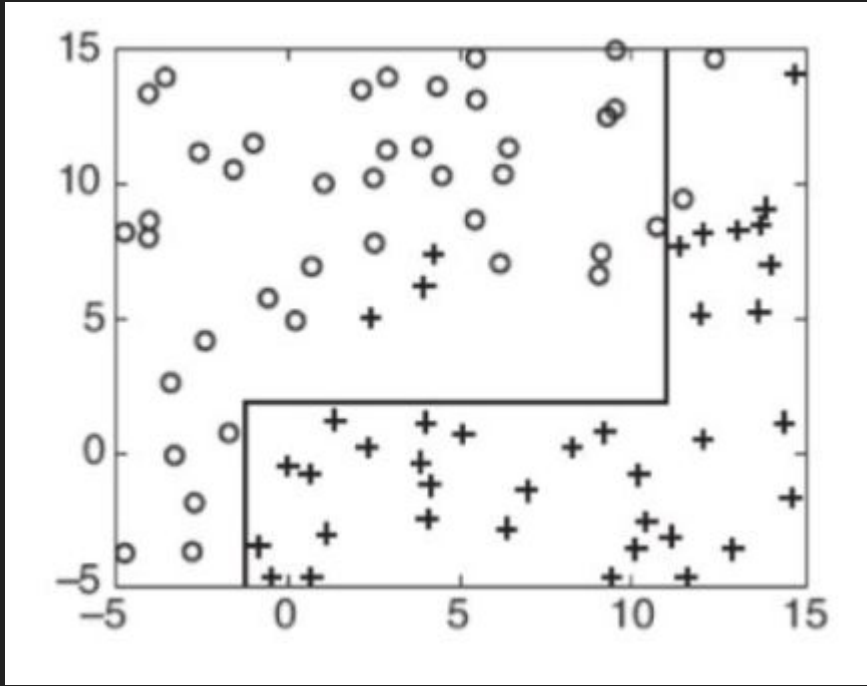
Representational Problem

Sometimes f can’t be explained in terms of our hypothesis space at all.

If we have a decision tree as our base classifier, it will form a rectinlear partition around our feature space.

But what if f is a diagonal line? We won’t be able to get there with finitely many rectangles.

Ensemble methods will help us better approximate a line.



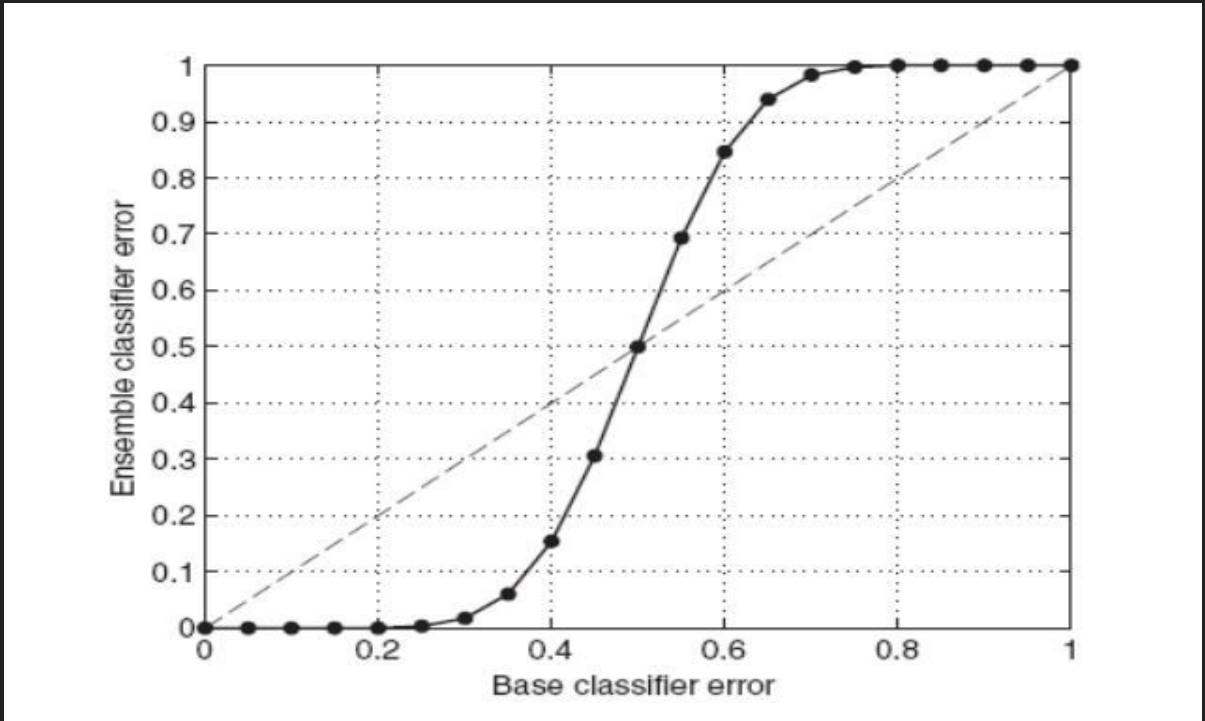
When to use an ensemble model?

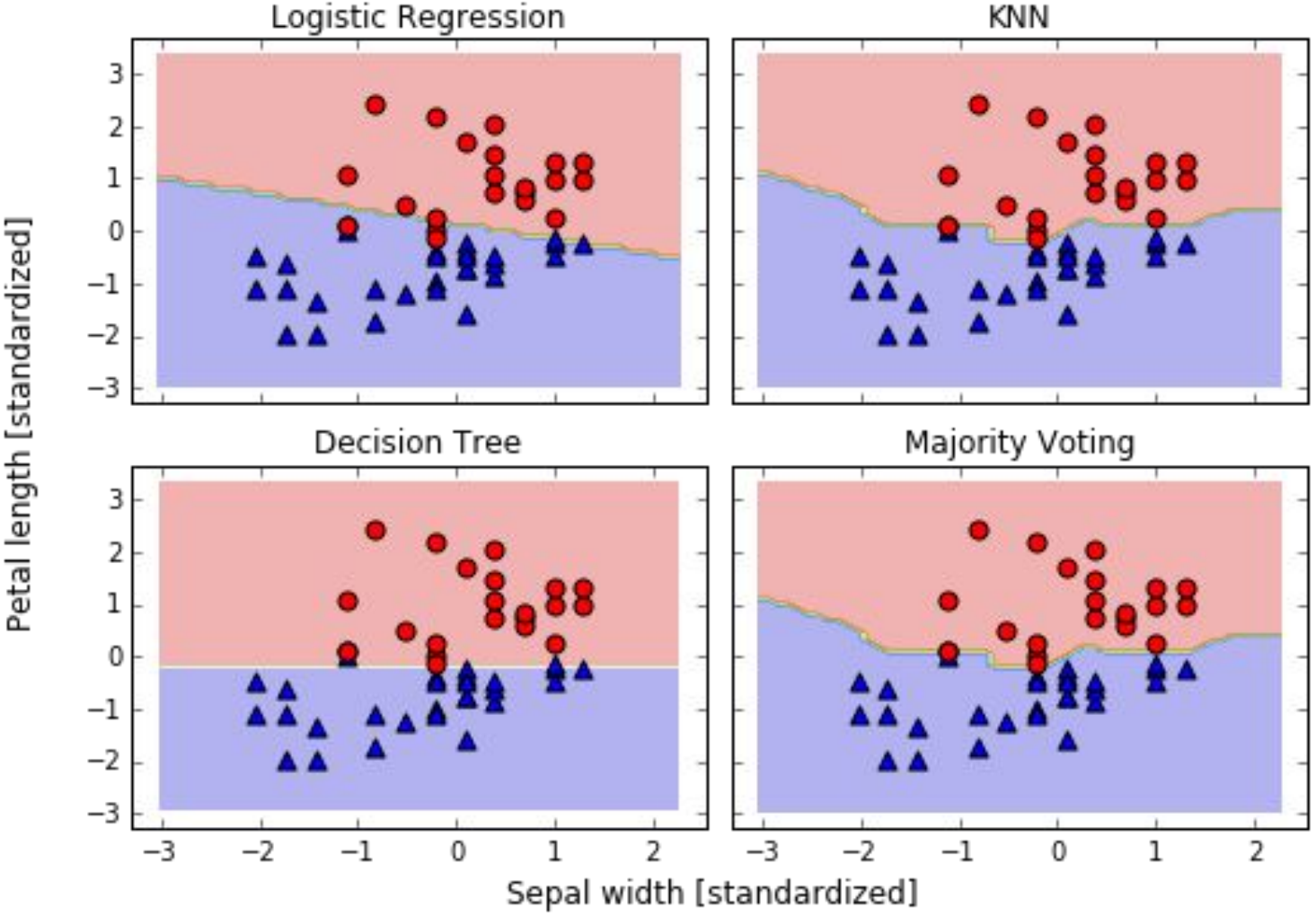
For an ensemble model to outperform a single baseline classifier,

Accuracy: baseline classifier must be better than random guessing

Diversity: misclassification must occur on different training sets (baseline models must be wrong in different ways)

Ensemble vs. base classifier error




Logistic Regression
KNN
Decision Tree
Majority Voting
Petal length [standardized]
Sepal width [standardized]
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2

Bagging (Bootstrap Aggregating)

Ensemble method where we resample the training set (randomly drawn subsets)

Creates samples with uniform weights

Then train these samples.

Random Forests are a special case of bagging which also use random feature subsets to fit the individual decision trees)

Original Data								
	A	B	C	D	E	F	G	H

Training Sets								
Training set 1	B	G	H	B	A	C	D	A
Training set 2	F	F	D	E	A	B	G	A
Training set 3	C	A	B	G	H	E	B	B
Training set 4	D	E	A	E	C	4	A	B

Bagging

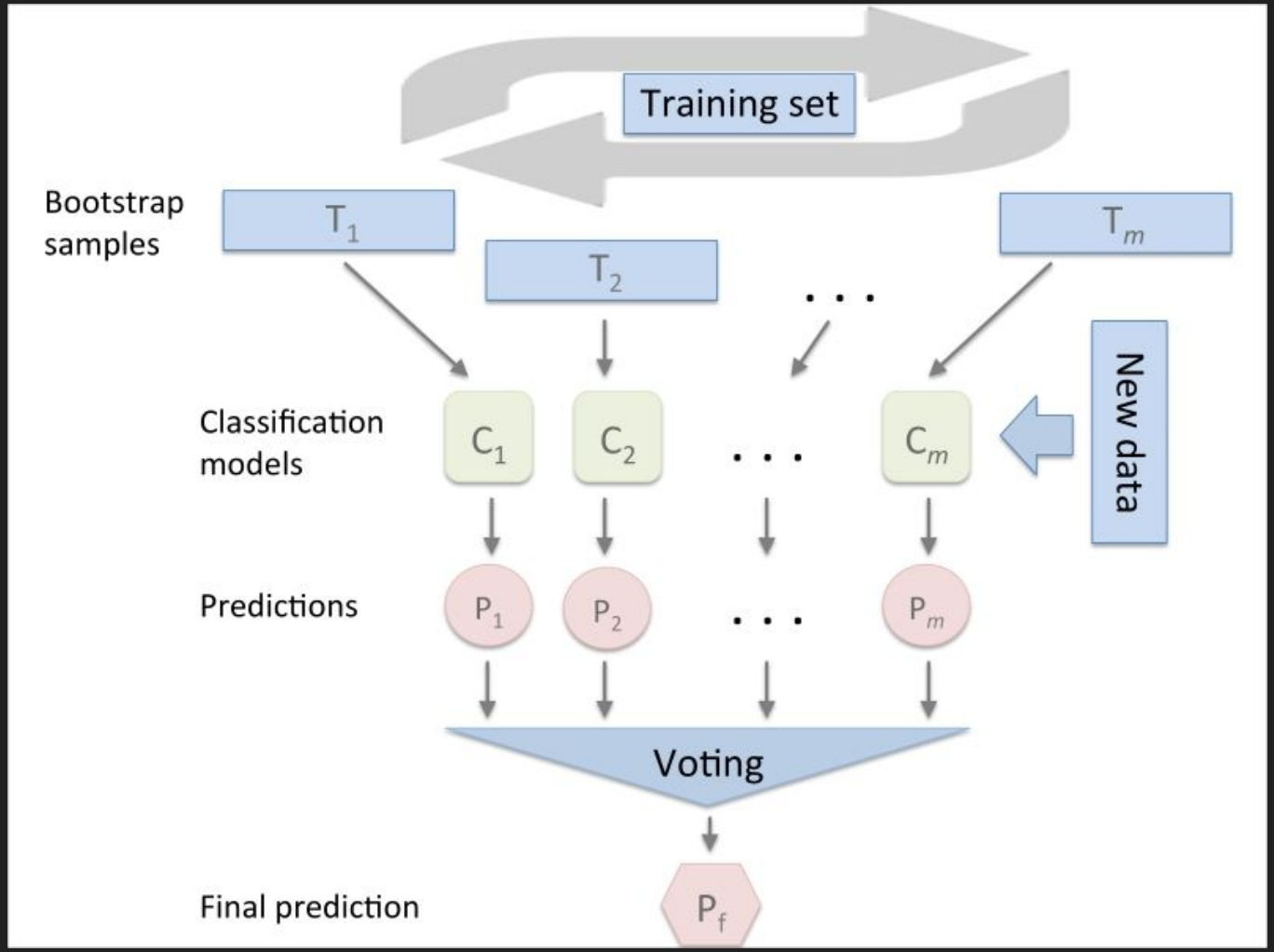
Reduces variance

Less susceptible to overfitting with noisy data

Most of the error is from bias

Best for strong and complex models (well-developed decision trees)

For weak models (shallow decision trees), boosting is a better method


Training set
Bootstrap samples
T_1
T_2
T_m
Classification models
C_1
C_2
C_m
New data
Predictions
P_1
P_2
P_m
Voting
Final prediction
P_f

Random Forests

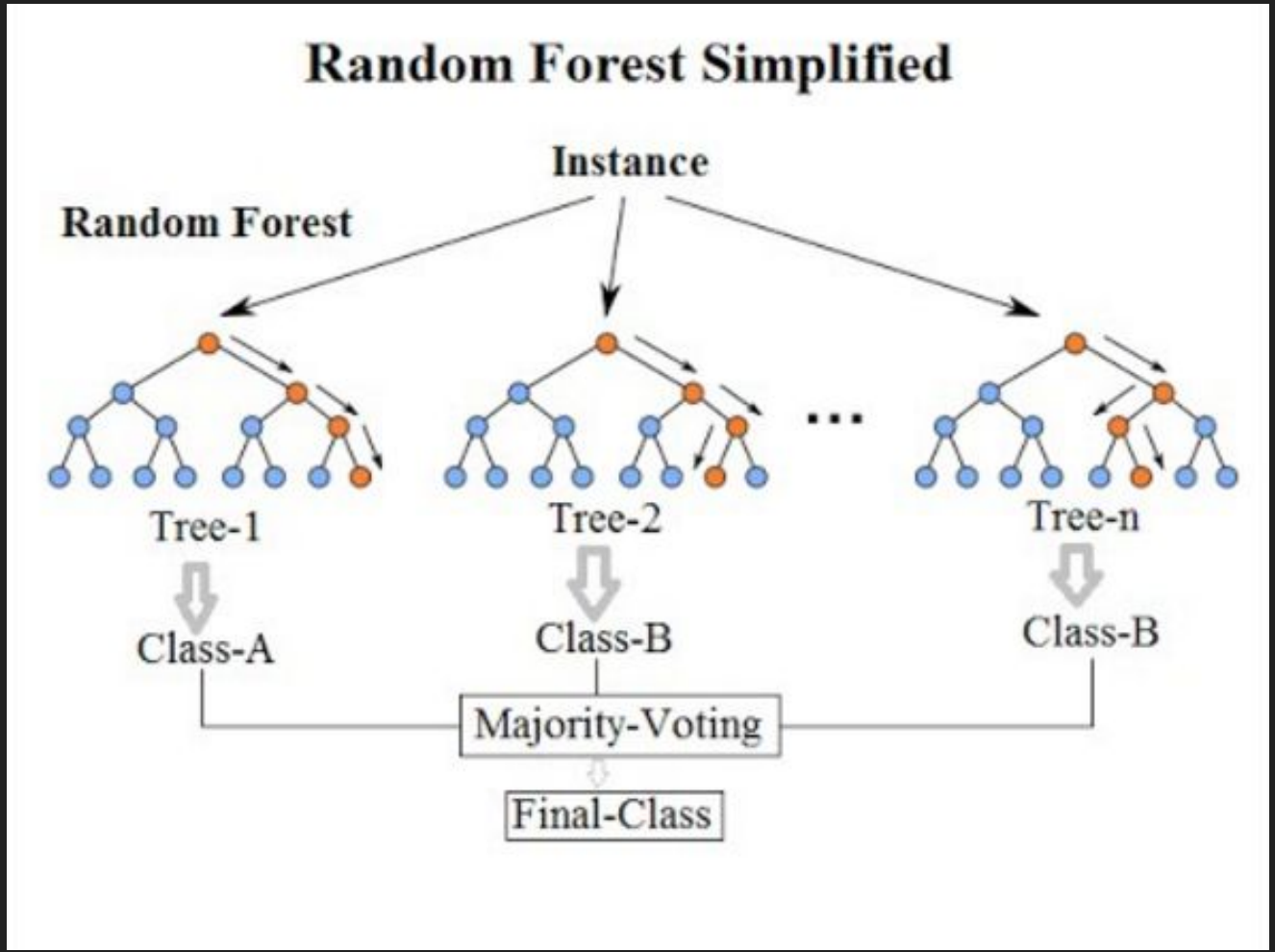
What was the main disadvantage of Decision Trees?

Random forests average multiple deep decision trees

Each trained on a different part of the same training set (bagging)

Reduces variance

Comes at the cost of increasing bias and losing interpretability


Random Forest Simplified
Instance
Random Forest
Tree-1
Tree-2
Tree-n
Class-A
Class-B
Class-B
Majority-Voting
Final-Class

Random Forests vs. Bagging Decision Trees

Feature Bagging: At each candidate split in the learning process selects a random subset of the features.

Ordinary bootstrap samples can become correlated. If one or a few of the features are strong predictors, they'll be selected in many of the base trees. You'll end up with a lot of the same base predictors.

For a problem with p features, it is typical to use:

p^(1/2) (rounded down) features in each split for a classification problem.

Pros and Cons of Random Forests

Pros

- Widely used
- Excellent prediction performance
- Doesn’t need careful normalization of features or lots of hyperparameter tuning
- Handles a mixture of feature types (like decision trees)
- Easily parallelized

Cons

- Models are difficult for humans to interpret
- May not be a good choice for high-dimensional tasks (e.g. fast linear methods may be better for something like text classification)

Boosting

With bagging and random forests we train models on separate subsets and then combine their prediction--parallelizing the training and then combining.

Boosting is a different ensemble technique that is sequential.

Uses weak learners as base models

Each weak learner has low variance and high bias (only slightly better than random guessing--e.g. decision tree stump)

Focuses on the samples that are hard to classify.

Boosting

1. Draw a random subset of training samples (d_1) without replacement from the training set (D) to train a weak learner (C_1)
2. Draw a second random training subset (d_2) without replacement from the training set and add 50% of the samples that were previously misclassified to train a weak learner (C_2)
3. Find the training samples (d_3) in the training set (D) on which C_1 and C_2 disagree to train a weak learner C_3.
4. Combine the weak learners (C_1, C_2, and C_3) with majority voting.

Boosting

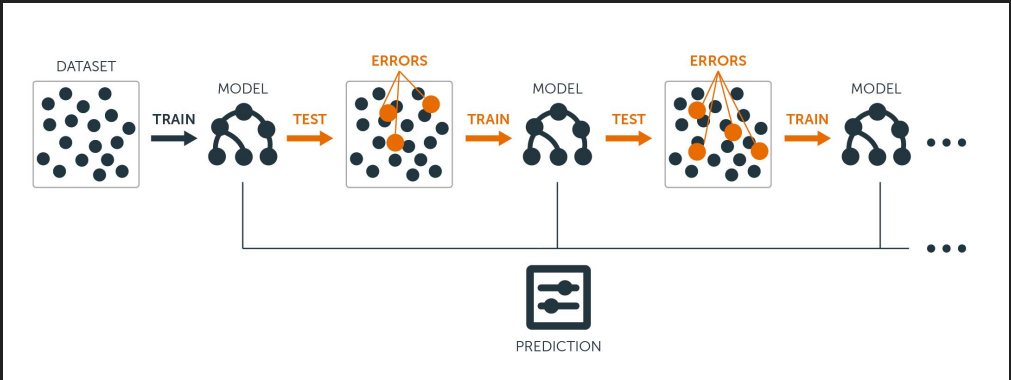
Since the base classifier's focus more and more closely on records that are difficult to classify as the sequence of iterations progresses, they are faced with progressively more difficult learning problems.

Boosting takes a base weak learner and tries to make it a strong learner by re-training it on the misclassified samples.

There are several algorithms for boosting. `AdaBoost` and `GradientBoostingClassifier` are in scikit learn.

Gradient Boosted Trees

- Build a series of small decision trees
- Each tree tries to correct the errors from previous stage

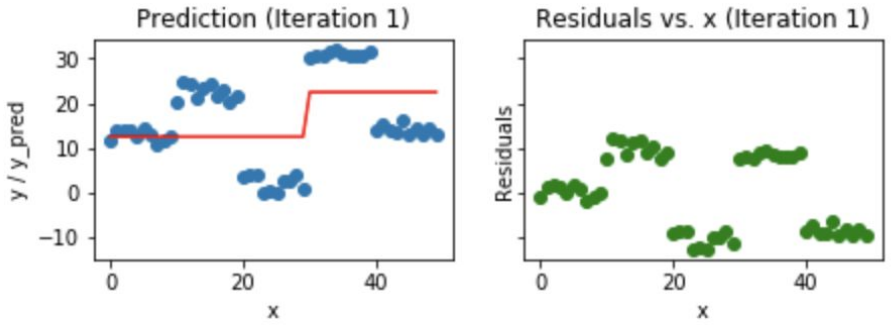


- Learning rate controls how hard each tree tries to correct remaining mistakes from previous round
  - High learning rate: more complex tree
  - Low learning rate: simpler trees

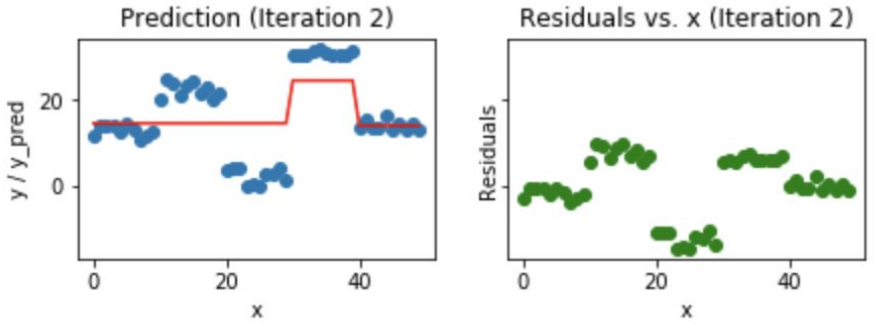
Gradient Boosting

1. Fit a model F to the data.
2. Look at the difference between our observed y and our model F. (residuals)
3. Fit a second model F_2 to (roughly) the residuals.
4. Aggregate your model F and F_2.

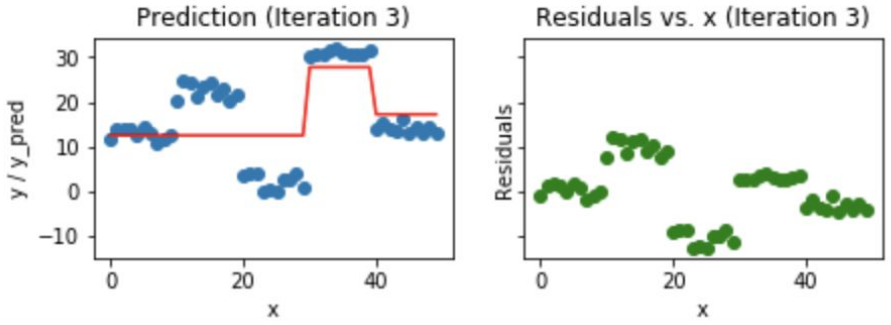
We can interpret residuals as negative gradients. By doing this, we can apply our gradient descent algorithm to optimize our loss and generalize this to many loss functions.


Prediction (Iteration 1)
y / y_pred
x

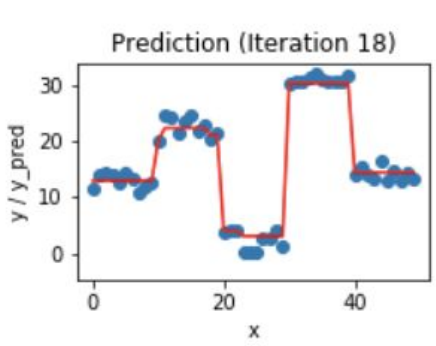
Residuals vs. x (Iteration 1)
Residuals
x

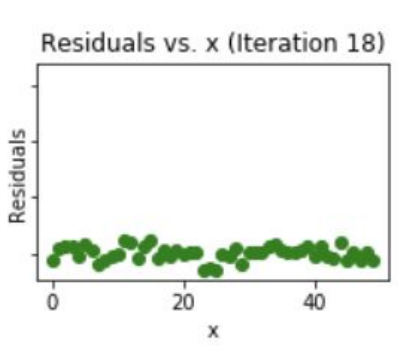

Prediction (Iteration 2)
y / y_pred
x

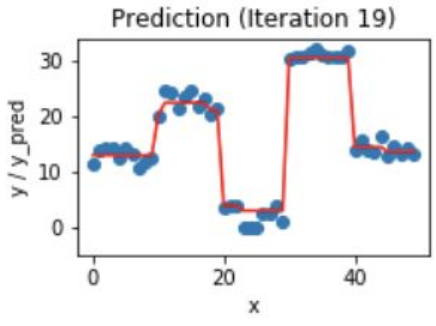
Residuals vs. x (Iteration 2)
Residuals
x

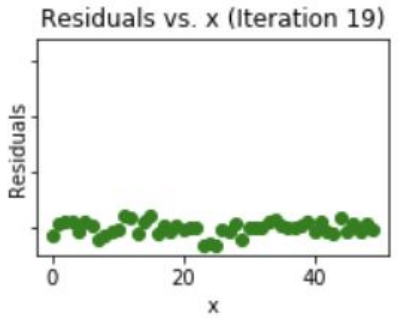

Prediction (Iteration 3)
y / y_pred
x

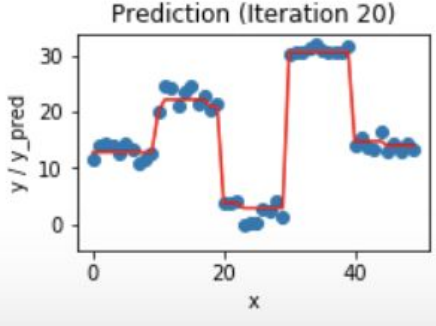
Residuals vs. x (Iteration 3)
Residuals
x

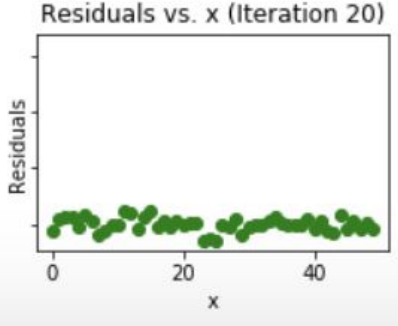

Prediction (Iteration 18)
y / y_pred
x


Residuals vs. x (Iteration 18)
Residuals
x


Prediction (Iteration 19)
y / y_pred
x


Residuals vs. x (Iteration 19)
Residuals
x


Prediction (Iteration 20)
y / y_pred
x


Residuals vs. x (Iteration 20)
Residuals
x

Pros and Cons of Gradient Boosting

Pros

Achieves higher performance than bagging when hyper-parameters tuned properly.

Can be used for classification and regression equally well.

Easily handles mixed data types.

Doesn’t require normalization.

Can use "robust" loss functions that make the model resistant to outliers.

Cons

Difficult and time consuming to properly tune hyper-parameters.

Cannot be parallelized like bagging (bad scalability when huge amounts of data).

More risk of overfitting compared to bagging.

Hard to interpret.

Not recommended for high-dimensional data.