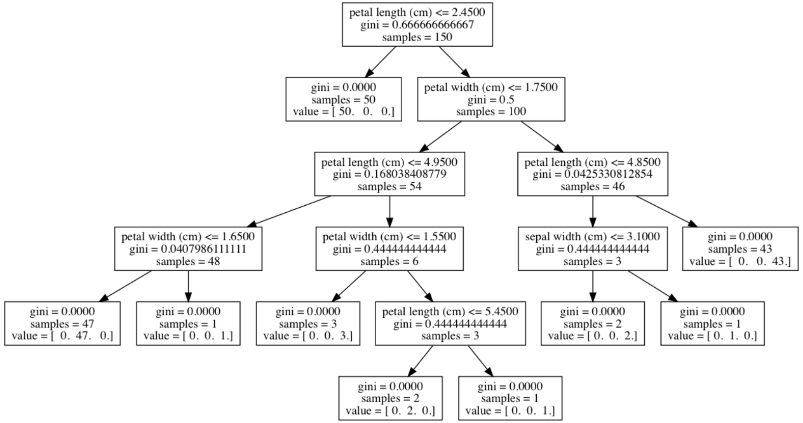
GENERAL ASSEMBLY

ENSEMBLE TECHNIQUES

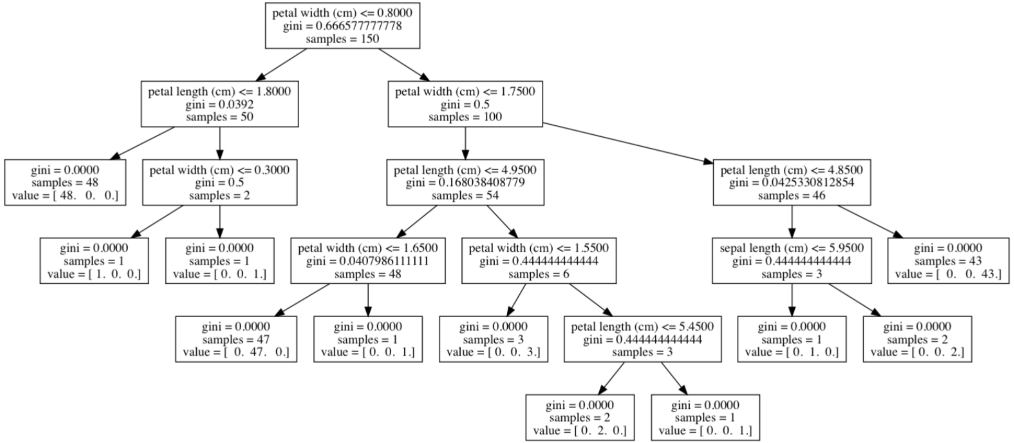
Brian Chung

- Hunt’s algorithm—This is the formal name for the heuristic we went over with the dot example.
- How are decision boundaries chosen? Along means of sample point features.
- How about pruning trees? SKlearn does not do post-pruning, only pre-pruning (i.e. how large to grow the tree). Find out best pre-pruning amount through cross-validation. However, rarely does one only use a single tree. More on that today.
- Noisiness in tree models. Let’s see with a tree.
- Bias vs. Variance in tree depth. Let’s see with a tree and a plot.

INSTABILITY IN TREE CREATION

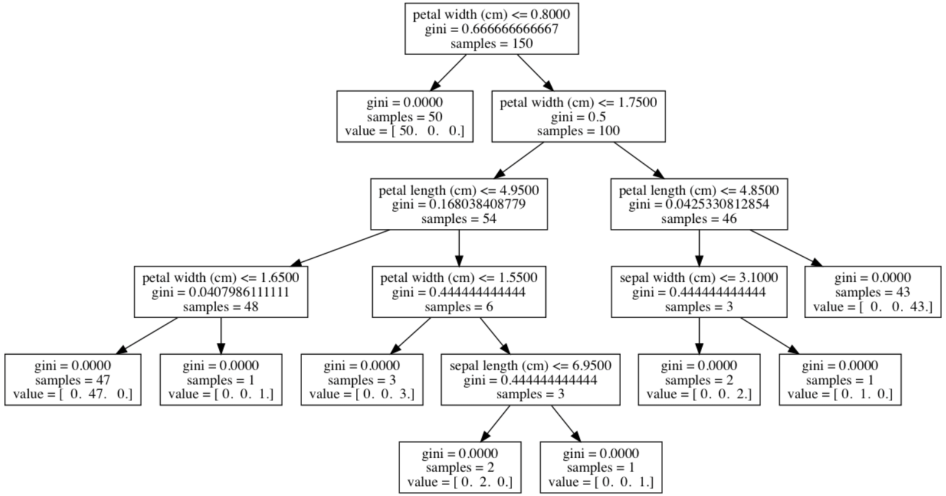


No change to data



Change one value of y

PREDICTIONS OVER LIMITING DEPTH



petal width (cm) <= 0.8000
gini = 0.666666666667
samples = 150

gini = 0.0000
samples = 50
value = [50. 0. 0.]

petal width (cm) <= 1.7500
gini = 0.5
samples = 100

petal length (cm) <= 4.9500
gini = 0.168038408779
samples = 54

petal length (cm) <= 4.8500
gini = 0.0425330812854
samples = 46

petal width (cm) <= 1.6500
gini = 0.0407986111111
samples = 48

petal width (cm) <= 1.5500
gini = 0.444444444444
samples = 6

sepal width (cm) <= 3.1000
gini = 0.444444444444
samples = 3

gini = 0.0000
samples = 43
value = [0. 0. 43.]

gini = 0.0000
samples = 47
value = [0. 47. 0.]

gini = 0.0000
samples = 1
value = [0. 0. 1.]

gini = 0.0000
samples = 3
value = [0. 0. 3.]

sepal length (cm) <= 6.9500
gini = 0.444444444444
samples = 3

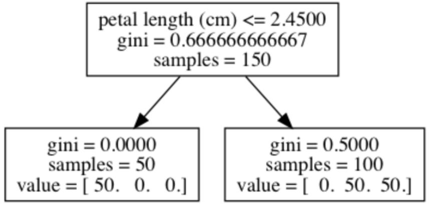
gini = 0.0000
samples = 2
value = [0. 0. 2.]

gini = 0.0000
samples = 1
value = [0. 1. 0.]

gini = 0.0000
samples = 2
value = [0. 2. 0.]

gini = 0.0000
samples = 1
value = [0. 0. 1.]

Unlimited Depth



petal length (cm) <= 2.4500
gini = 0.666666666667
samples = 150

gini = 0.0000
samples = 50
value = [50. 0. 0.]

gini = 0.5000
samples = 100
value = [0. 50. 50.]

Max_depth=1

I. ENSEMBLE TECHNIQUES

II. PROBLEMS IN CLASSIFICATION

III. BAGGING

IV. RANDOM FORESTS

V. BOOSTING

EXERCISE:

VI. ADABOOST

I. ENSEMBLE TECHNIQUES

https://youtu.be/iOucwX7Z1HU?t=1m1s

Q: What are ensemble techniques?

Q: What are ensemble techniques?

A: Methods of improving classification accuracy by aggregating predictions over several **base classifiers***.*

Q: What are ensemble techniques?

A: Methods of improving classification accuracy by aggregating predictions over several **base classifiers***.*

Ensembles are often much more accurate than the base classifiers that compose them.

Q: What are ensemble techniques?

A: Methods of improving classification accuracy by aggregating predictions over several **base classifiers***.*

Ensembles are often much more accurate than the base classifiers that compose them.

NOTE

Base classifiers and ensemble classifiers are sometimes called *weak learners* and *strong learners*.

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

1) *the bc's must be* **accurate***: they must outperform random guessing*

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

1) *the bc's must be* **accurate***: they must outperform random guessing*

2) *the bc's must be* **diverse***: their misclassifications must occur on different training examples*

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

1) *the bc's must be* **accurate***:* *low bias*

2) *the bc's must be* **diverse***:* *uncorrelated*

In order for an ensemble classifier to outperform a single base classifier, the following conditions must be met:

1) *the bc's must be* **accurate**: *low bias*

2) *the bc's must be* **diverse**: *uncorrelated*

NOTE

Ideally, we would also like the base classifiers to be *unstable* to variations in the training set.

In other words, *high variance*.

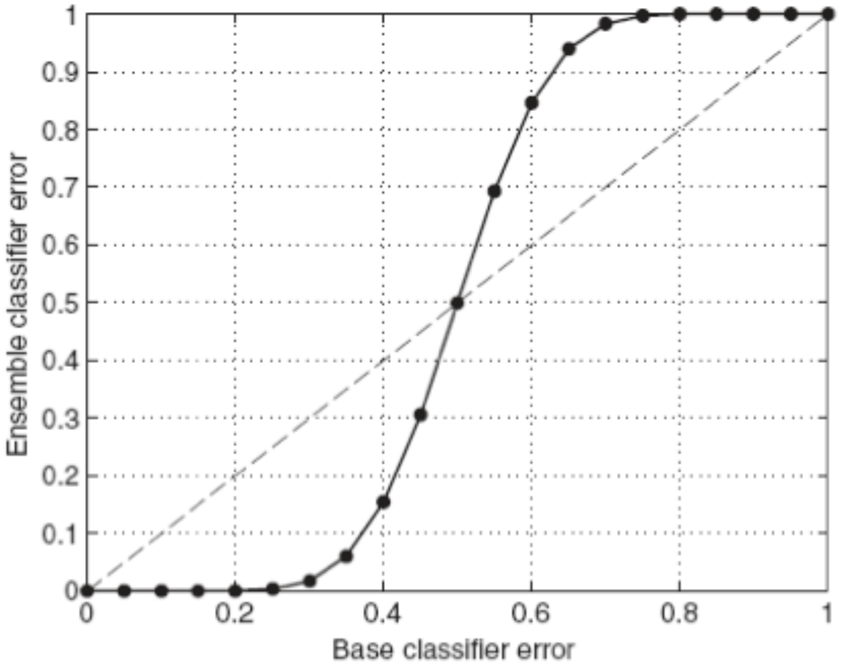


Figure 5.30. Comparison between errors of base classifiers and errors of the ensemble classifier.

NOTE

dashed line = perfectly correlated bc's (no improvement using ensemble)

solid line = perfectly uncorrelated bc's (some improvement for unbiased bc's)

II. PROBLEMS IN CLASSIFICATION

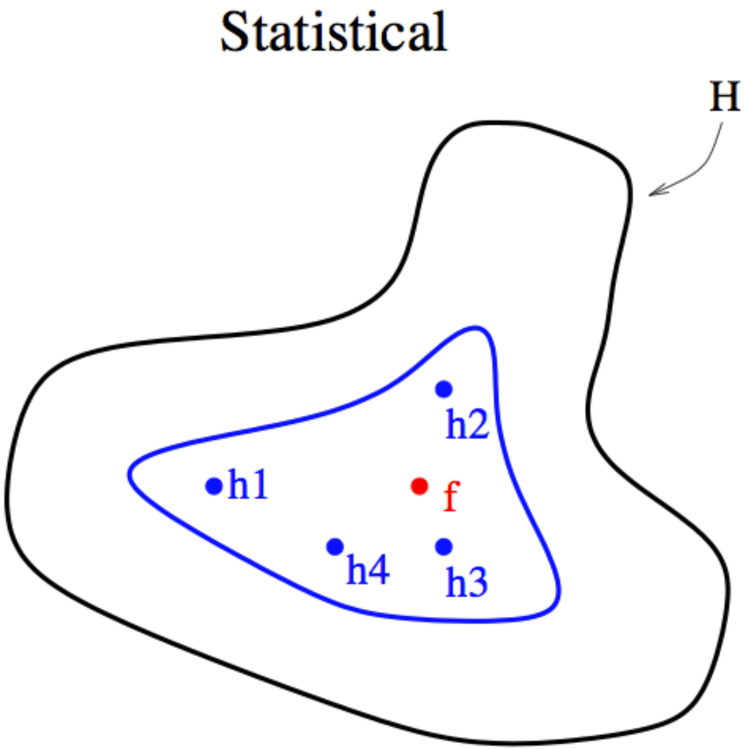
In any supervised learning task, our goal is to make predictions of the true classification function f *by learning the classifier* h*.*

In any supervised learning task, our goal is to make predictions of the true classification function f by learning the classifier h.

There are three main problems that can prevent this:

- *statistical problem*
- *computational problem*
- *representational problem*

If the amount of training data available is small, the base classifier will have difficulty converging to h.



NOTE

The true function *f* is best approximated as an average of the base classifiers.

source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

If the amount of training data available is small, the base classifier will have difficulty converging to h.

An ensemble classifier can mitigate this problem by "averaging out" base classifier predictions to improve convergence.

Even with sufficient training data, it may still be computationally difficult to find the best classifier h.

For example, if our base classifier is a decision tree, an exhaustive search of the hypothesis space of all possible classifiers is extremely complex (NP-complete).

Even with sufficient training data, it may still be computationally difficult to find the best classifier h.

For example, if our base classifier is a decision tree, an exhaustive search of the hypothesis space of all possible classifiers is extremely complex (NP-complete).

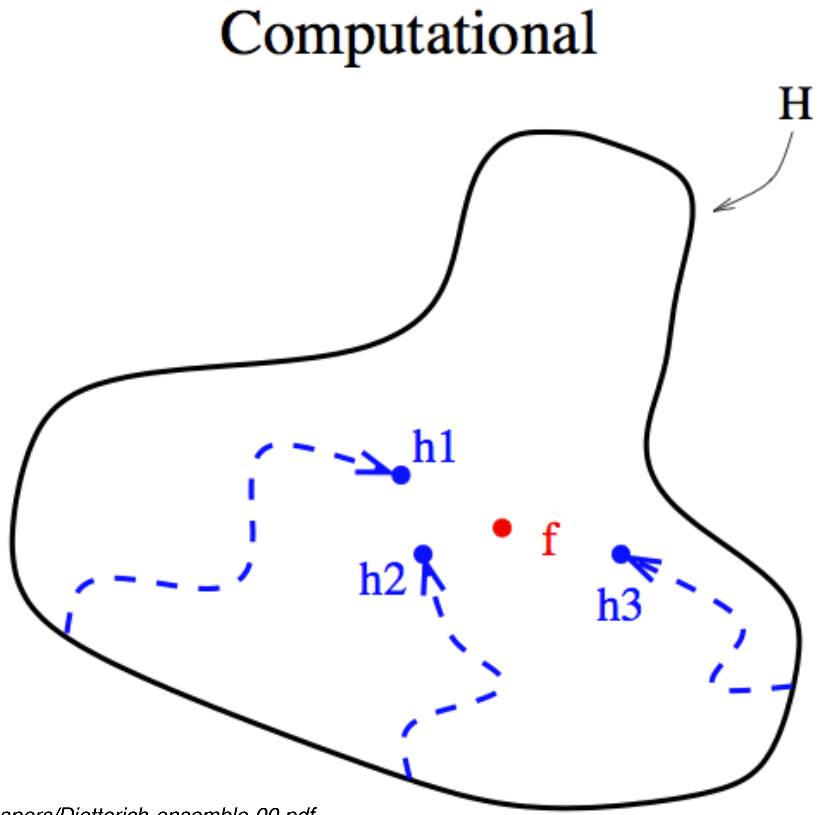
NOTE

Recall that this is why we used a *heuristic algorithm* (greedy search).

Even with sufficient training data, it may still be computationally difficult to find the best classifier h.

For example, if our base classifier is a decision tree, an exhaustive search of the hypothesis space of all possible classifiers is extremely complex (NP-complete).

An ensemble composed of several BC's with different starting points can provide a better approximation to f than any individual BC.



NOTE

The true function *f* is often best approximated by using several starting points to explore the hypothesis space.

source: http://www.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf

Sometimes f cannot be expressed in terms of our hypothesis at all.

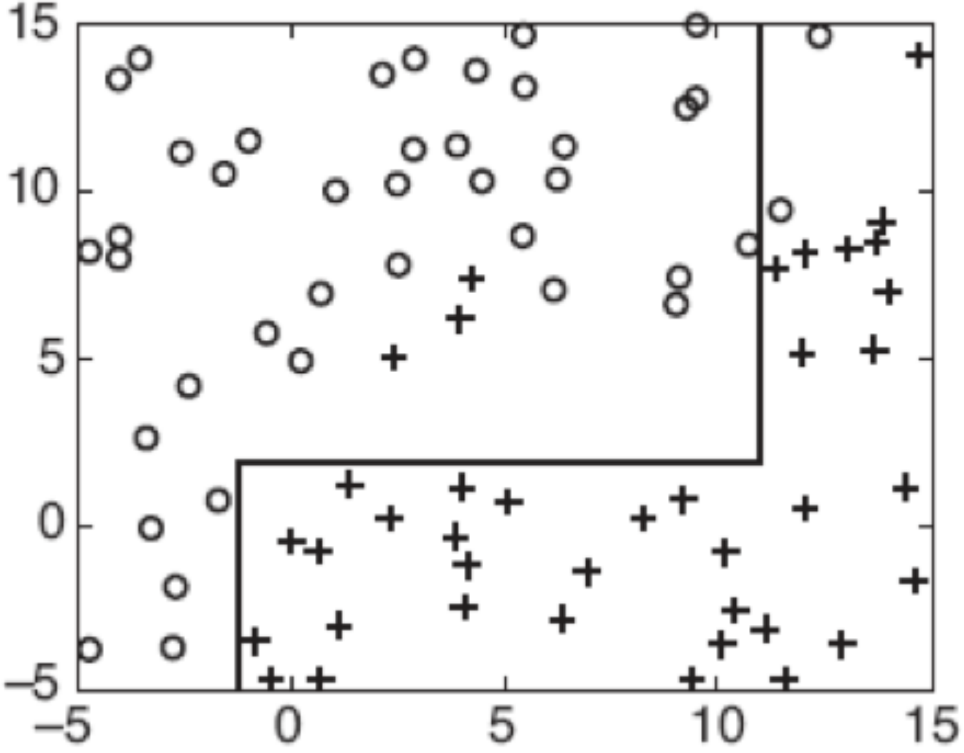
Sometimes f cannot be expressed in terms of our hypothesis at all.

To illustrate this, suppose we use a decision tree as our base classifier.

Sometimes f cannot be expressed in terms of our hypothesis at all.

To illustrate this, suppose we use a decision tree as our base classifier.

A decision tree works by forming a rectilinear partition of the feature space.



NOTE

What is a *rectilinear* decision boundary?

One whose segments are *orthogonal* to the x & y axes.

But what if f is a diagonal line?

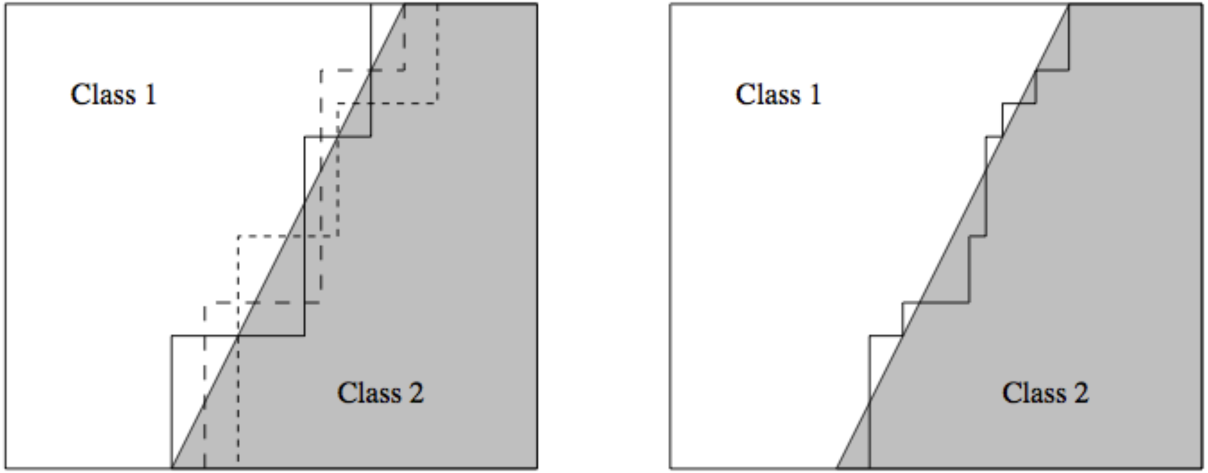
But what if f is a diagonal line?

Then it cannot be represented by finitely many rectilinear segments, and therefore the true decision boundary cannot be obtained by a decision tree classifier.

But what if f is a diagonal line?

Then it cannot be represented by finitely many rectilinear segments, and therefore the true decision boundary cannot be obtained by a decision tree classifier.

However, it may be still be possible to approximate f or even to expand the space of representable functions using ensemble methods.

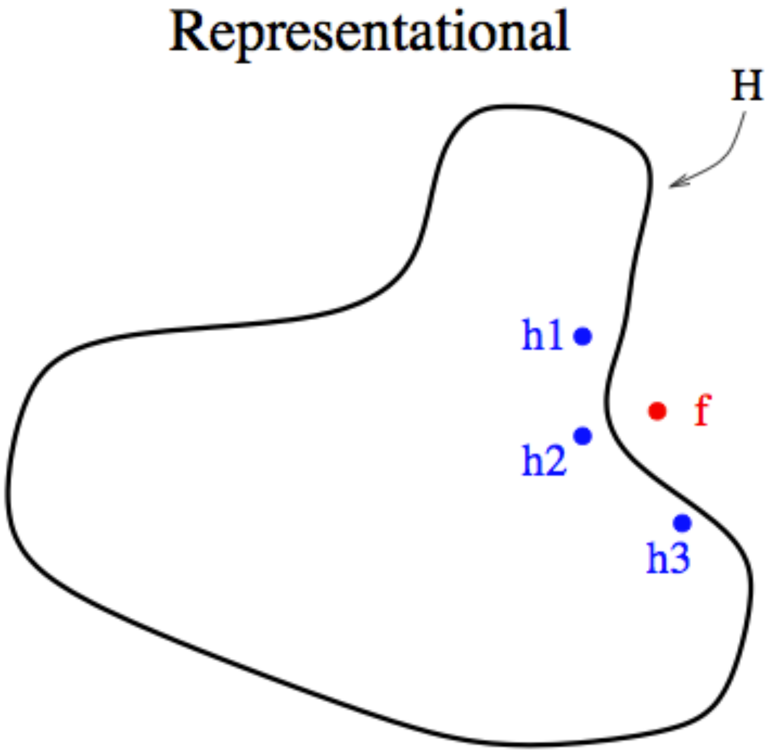


NOTE

An ensemble of decision trees can approximate a diagonal decision boundary.

Fig. 4. The left figure shows the true diagonal decision boundary and three staircase approximations to it (of the kind that are created by decision tree algorithms). The right figure shows the voted decision boundary, which is a much better approximation to the diagonal boundary.

THE REPRESENTATIONAL PROBLEM – EXPANDING THE HYPOTHESIS SPACE



NOTE

Ensemble classifiers can be effective even if the true decision boundary lies outside the hypothesis space.

Q: How do you create an ensemble classifier?

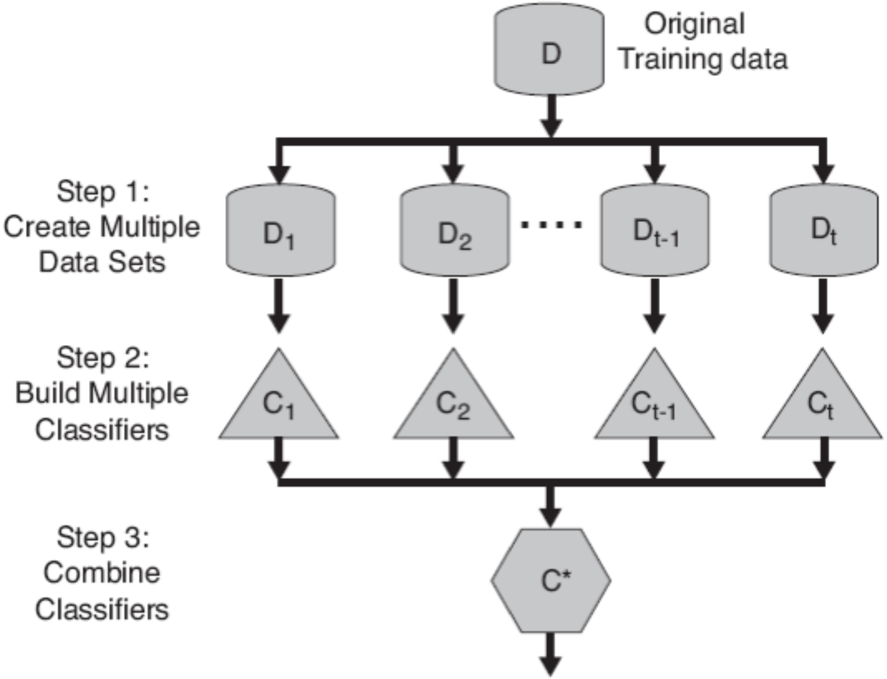


Figure 5.31. A logical view of the ensemble learning method.

Q: How do you generate several base classifiers?

Q: How do you generate several base classifiers?

A: There are several ways to do this:

- *manipulating the training set*
- *manipulating the output labels*
- *manipulating the learning algorithm itself*

We will talk about a few examples of each of these.

III. BAGGING

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling**.

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling***.*

We learn k *base classifiers on* k *different samples of training data.*

These samples are independently created by resampling the training data using uniform weights (eg, a uniform **sampling distribution***).*

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling***.*

We learn k base classifiers on k different samples of training data.

These samples are independently created by resampling th data using uniform weights (eg, a uniform **sampling distril**

NOTE

Each training sample is the same size as the original training set.

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling***.*

We learn k base classifiers on k different samples of training data.

These samples are independently created by resampling th data using uniform weights (eg, a uniform **sampling distril**

NOTE

Resampling means that some training records may appear in a sample more than once, or even not at all.

Bagging *(bootstrap aggregating) is a method that involves manipulating the training set by* **resampling***.*

We learn k base classifiers on k different samples of training data.

These samples are independently created by resampling the training data using uniform weights (eg, a uniform **sampling distribution***).*

The final prediction is made by taking a majority vote across bc's.

Bagging reduces the variance in our generalization error by aggregating multiple base classifiers together (provided they satisfy our earlier requirements).

Bagging reduces the variance in our generalization error by aggregating multiple base classifiers together (provided they satisfy our earlier requirements).

If the base classifier is stable, then the ensemble error is primarily due to bc bias, and bagging may not be effective.

Bagging reduces the variance in our generalization error by aggregating multiple base classifiers together (provided they satisfy our earlier requirements).

If the base classifier is stable, then the ensemble error is primarily due to bc bias, and bagging may not be effective.

Since each sample of training data is equally likely, bagging is not very susceptible to overfitting with noisy data.

IV. RANDOM FORESTS

A random forest is an ensemble of decision trees where each base classifier is grown using a random effect.

A random forest is an ensemble of decision trees where each base classifier is grown using a random effect.

One way to do this is to randomly choose the top amongst k random features to split each node.

A random forest is an ensemble of decision trees where each base classifier is grown using a random effect.

One way to do this is to randomly choose the top amongst k random features to split each node.

For a small number of features, we can also create linear combinations of features and select splits from the enhanced feature set (Forest-RC).

A random forest is an ensemble of decision trees where each base classifier is grown using a random effect.

One way to do this is to randomly choose the top amongst k random features to split each node.

For a small number of features, we can also create linear combinations of features and select splits from the enhanced feature set (Forest-RC).

Or, we can select splitting features completely at random (Forest-RI).

Random forests are robust to noise, and can also have better runtime than other ensemble methods (since the feature space is reduced in some cases).

Random forests are robust to noise, and can also have better runtime than other ensemble methods (since the feature space is reduced in some cases).

In addition, they are ***simple to tune*** *since there is only one hyperparameter—the number of trees to build*

V. BOOSTING

Boosting is an iterative procedure that adaptively changes the weights of training records at each iteration.

Boosting is an iterative procedure that adaptively changes the weights of training records at each iteration.

The first iteration uses uniform weights (like bagging). In subsequent iterations, the weights are adjusted to emphasize records that were misclassified in previous iterations.

Boosting is an iterative procedure that adaptively changes the weights of training records at each iteration.

The first iteration uses uniform weights (like bagging). In subsequent iterations, the weights are adjusted to emphasize records that were misclassified in previous iterations.

The final prediction is constructed by a weighted vote (where the weights for a bc depends on its training error).

Boosting is an iterative procedure that adaptively changes the weights of training records at each iteration.

The first iteration uses uniform weights (like bagging). In su
iterations, the weights are adjusted to emphasize records t
misclassified in previous iterations.

The final prediction is constructed by a weighted vote (whe
weights for a bc depends on its training error).

NOTE

The bc's focus more and more closely on records that are difficult to classify as the sequence of iterations progresses.

Thus the bc's are faced with progressively more difficult learning problems.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

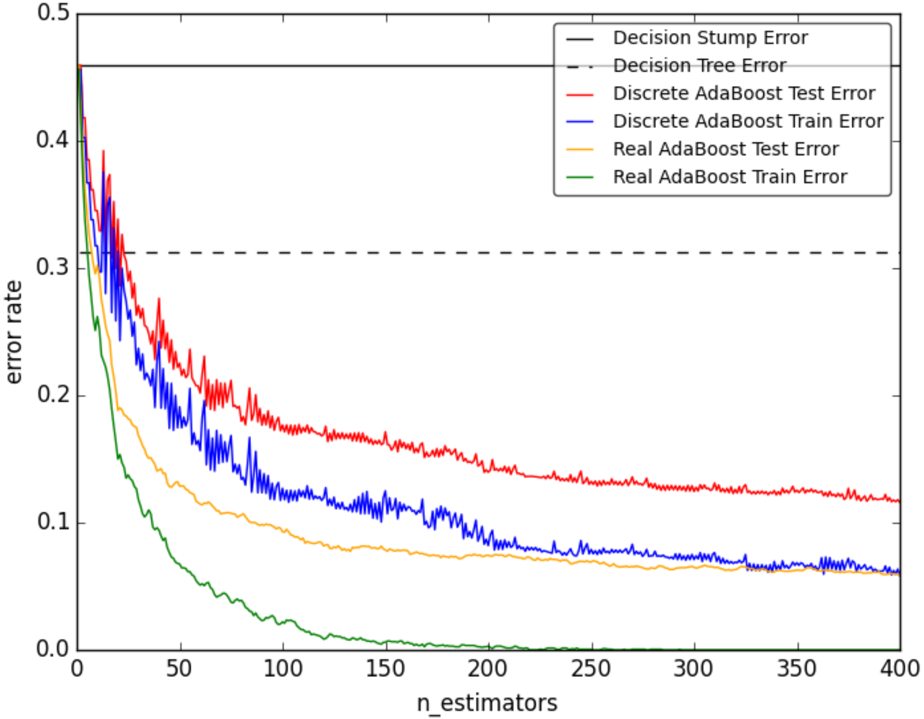
By explicitly trying to optimize the weighted ensemble vote, boosting attacks the representation problem head-on.

Updating the sampling distribution and forming an ensemble prediction leads to a nonlinear combination of the base classifiers.

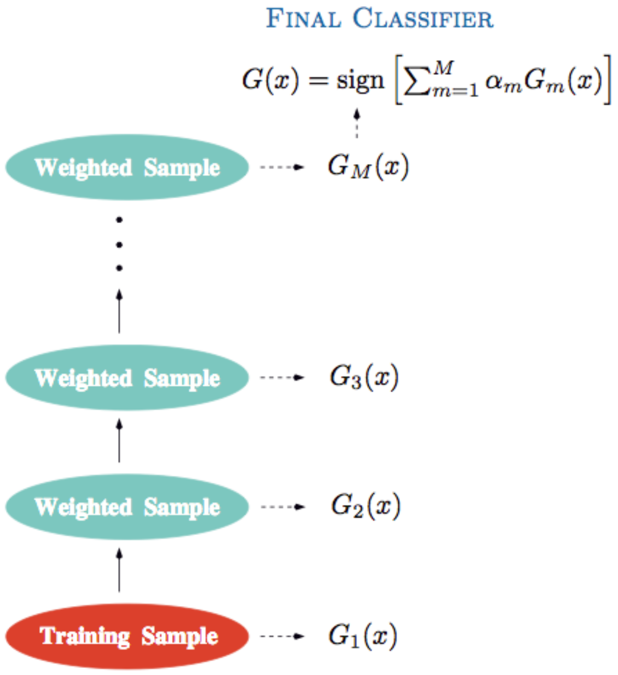
By explicitly trying to optimize the weighted ensemble vote, boosting attacks the representation problem head-on.

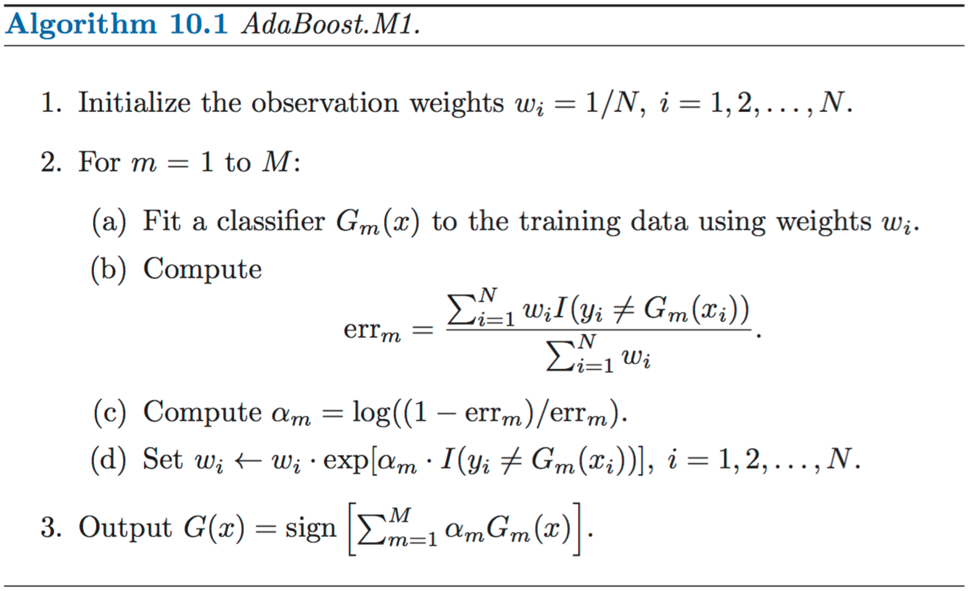
Boosting tends to work well with simpler base classifiers (i.e. high bias, lower variance)

In addition, the training error can hit 0, but the testing error will still improve in boosted models!



BOOSTING



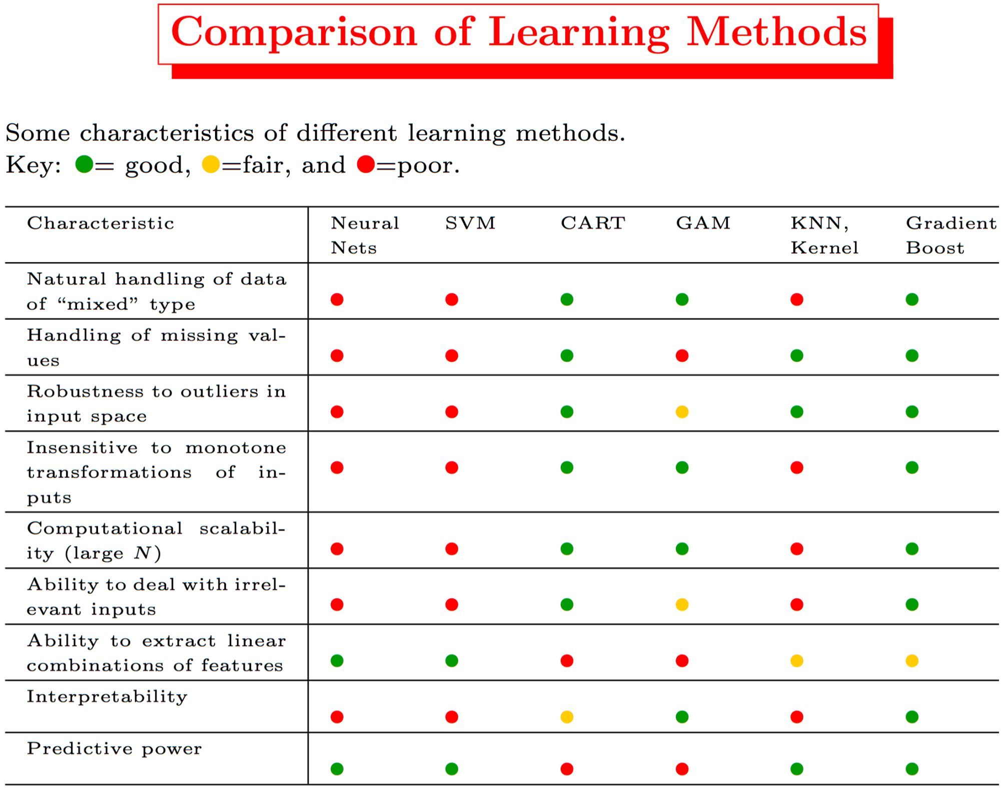


Algorithm 10.1 *AdaBoost.M1.*

```
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
    (a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
    (b) Compute
            $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
    (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
    (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.
```

From Elements of Statistical Learning

COMPARISONS



Comparison of Learning Methods

Some characteristics of different learning methods.
Key: 🟢= good, 🟡=fair, and 🔴=poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, Kernel	Gradient Boost
Natural handling of data of “mixed” type	🔴	🔴	🟢	🟢	🔴	🟢
Handling of missing values	🔴	🔴	🟢	🔴	🟢	🟢
Robustness to outliers in input space	🔴	🔴	🟢	🟡	🟢	🟢
Insensitive to monotone transformations of inputs	🔴	🔴	🟢	🟢	🔴	🟢
Computational scalability (large N)	🔴	🔴	🟢	🟢	🔴	🟢
Ability to deal with irrelevant inputs	🔴	🔴	🟢	🟡	🔴	🟢
Ability to extract linear combinations of features	🟢	🟢	🔴	🔴	🟡	🟡
Interpretability	🔴	🔴	🟡	🟢	🔴	🟢
Predictive power	🟢	🟢	🔴	🔴	🟢	🟢

http://jessica2.msri.org/attachments/10778/10778-boost.pdf

THAT'S IT!

- Exit Tickets: DAT1 - Lesson 13 - Ensemble Methods
- More Ensemble resources:
- SKLearn docs: http://scikit-learn.org/stable/modules/ensemble.html
- Blog post on RF: http://blog.yhat.com/posts/random-forests-in-python.html
- Good article on ensemble learning: http://web.cs.iastate.edu/~jtian/cs573/Papers/Dietterich-ensemble-00.pdf