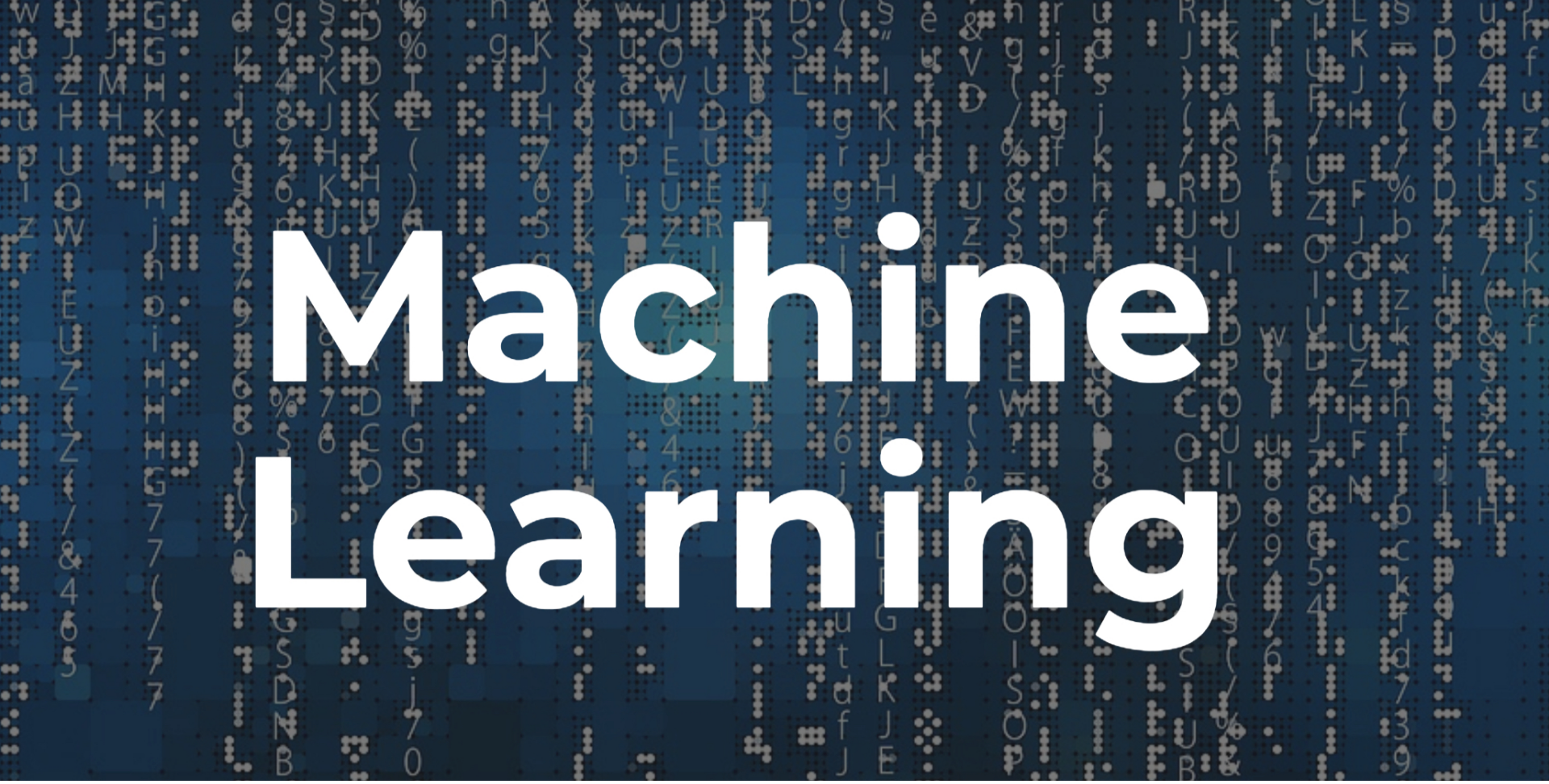


CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Decision trees (continued)
- Overfitting vs. generalization
  - bias-variance tradeoff
- Model evaluation/selection
  - Validation, Cross-Validation
- [If time] Intro to ensemble methods
  - Random forests

With credit to S. Dasgupta, C. Tan, T. Joachims

Decision tree issues

Very expressive family of classifiers:

Any type of data can be accommodated:
real, Boolean, categorical, …

Can perfectly fit any self-consistent training set
- no example pairs (x, y), (x, y')

But this also means that there is serious danger of overfitting.

Greedy decision tree building

Start with all points in a single node

Repeat

    Pick the split with greatest benefit

Until ???

When to stop?

(i) When each leaf is pure?

(ii) When the tree is already pretty big?

(iii) When each leaf has uncertainty < some threshold?

Common strategy: keep going until leaves are pure (recall: this didn't work too well for us earlier…)

Then, shorten the tree by pruning, to correct the overfitting problem.

What is overfitting?

Data comes from some true underlying distribution D on X x Y.

[X = input space, Y = label space]

All we ever see are samples from D: training set, test set, etc.

When we choose a classifier h: X → Y, we can talk about its error on the training set (x_1, y_1), …, (x_m, y_m):

$$\hat{\epsilon}(h) = \frac{1}{m} \sum_{i=1}^{m} \mathbf{1}(h(x_i) \neq y_i)$$

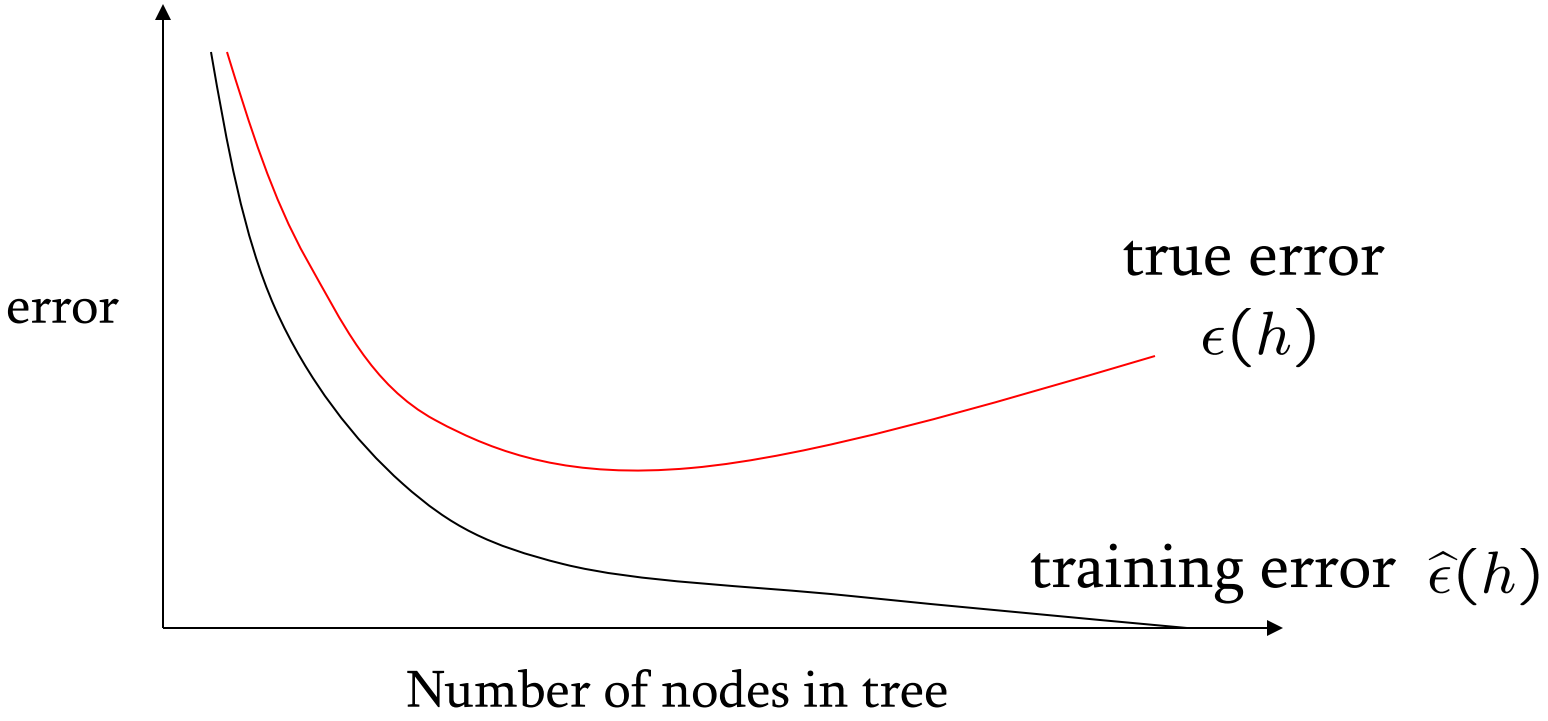
But we can also talk about its true error:

$$\epsilon(h) = \mathbf{P}_{(x,y)\sim D}(h(x) \neq y)$$

How are these two quantities related?

Overfitting: picture

E.g., building a decision tree



As we make our tree more and more complex:

- The training error keeps going down
- but the true error stops improving and may even get worse!

Overfitting: one perspective

1. The true underlying distribution D is the one whose structure we would like to capture
2. The training data reflects the structure of D, so it helps us.
3. But it also has **chance** structure of its own – we must try to avoid modeling this.

For instance: D = uniform distribution over {1,2,3,…,100}

Pick three training points: eg. 6, 12, 98.

They all happen to be even: but this is just chance structure.

It would be bad to build this into a classifier.

Overfitting: another perspective

"Fit a line to a point": absurd

"Fit a plane to two points": likewise

Moral: It is not good to use a model which is so complex that there isn't enough data to reliably estimate its parameters.

Decision tree pruning

[1] Split the training data S_{full} into two parts:

- A smaller training set S
- A validation set V (a model of reality, a surrogate test set)

[2] Build a full decision tree T using S

[3] Then prune T using V:

Repeat

if there a node u in T such that removing the subtree rooted at u decreases the error on V:

T = T – {subtree rooted at u}

Of course, V has chance structure too, but its chance structure is unlikely to coincide with that of S.

Example: SPAM data set

4601 points, each corresponding to an email message

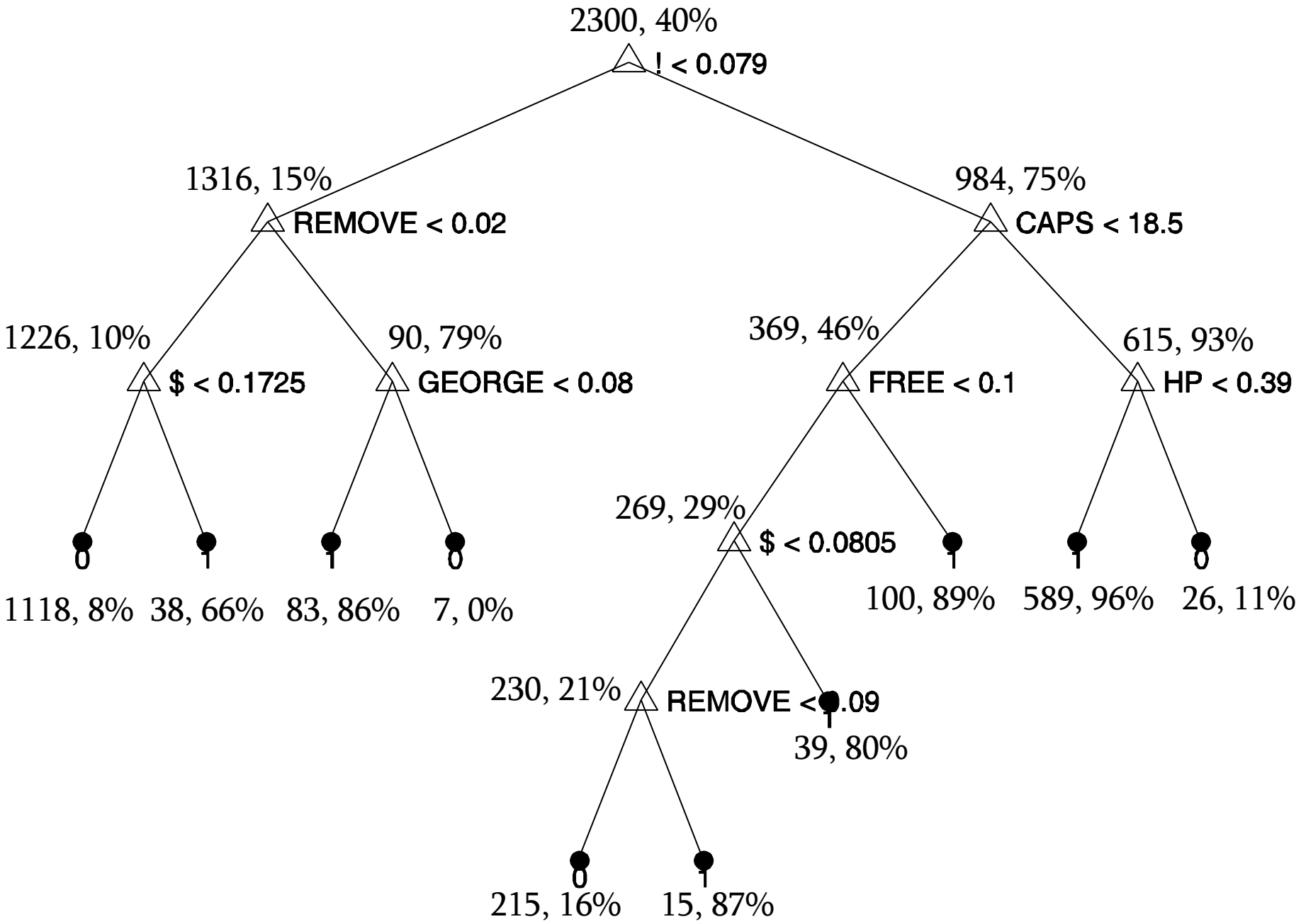
39.4% are SPAM

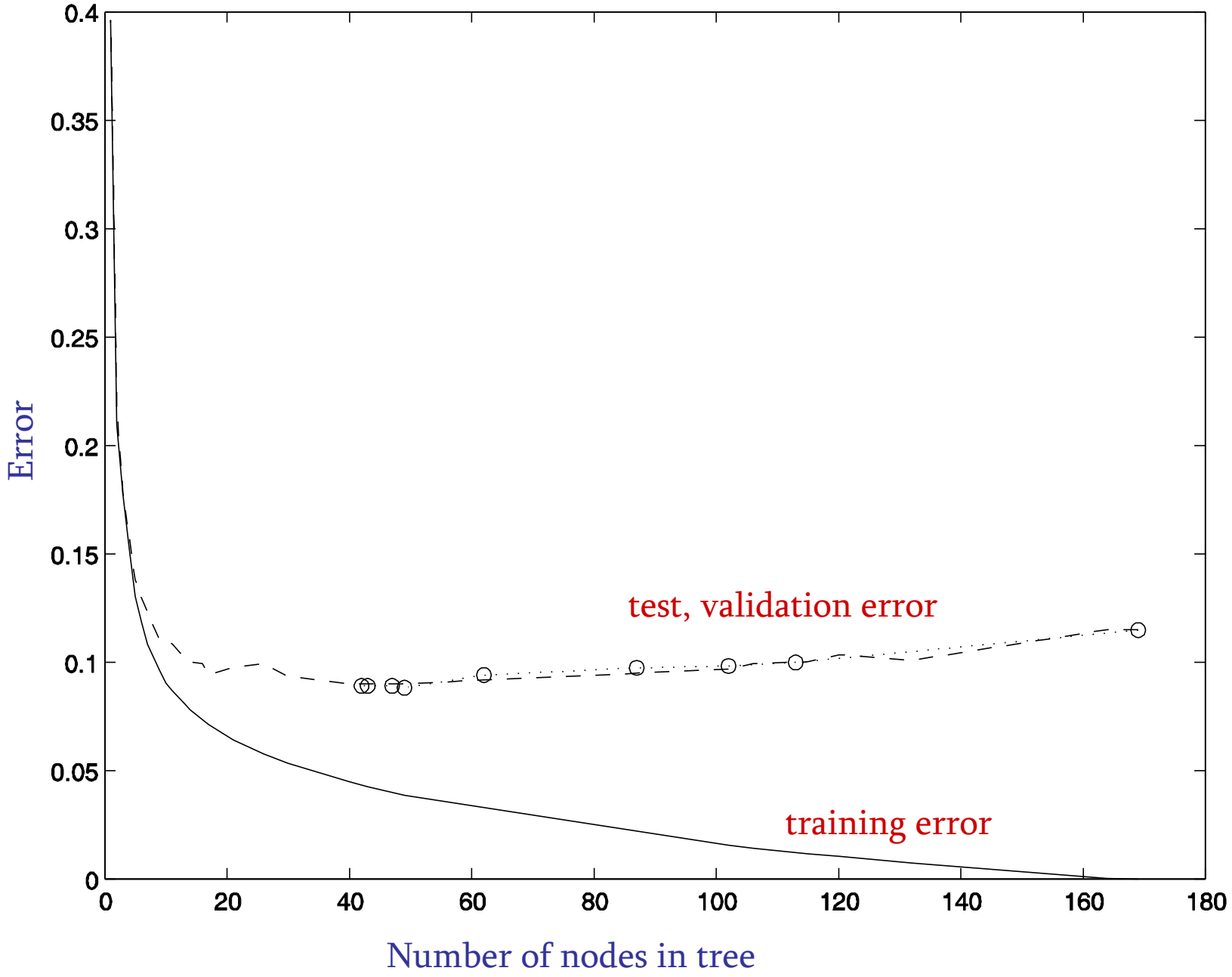
Each point has 57 features:

- 48 check for specific words, eg. FREE
- 6 check for specific characters, eg. !
- 3 others, eg. longest run of capitals

Randomly divide into three parts:

50%	training data
25%	validation
25%	testing


2300, 40%
! < 0.079
1316, 15%
REMOVE < 0.02
984, 75%
CAPS < 18.5
1226, 10%
$ < 0.1725
90, 79%
GEORGE < 0.08
369, 46%
FREE < 0.1
615, 93%
HP < 0.39
269, 29%
$ < 0.0805
230, 21%
REMOVE < 0.09
0
1
1
0
1
1
1
0
1118, 8%
38, 66%
83, 86%
7, 0%
100, 89%
589, 96%
26, 11%
39, 80%
0
1
215, 16%
15, 87%


0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
Error
0
20
40
60
80
100
120
140
160
180
Number of nodes in tree
test, validation error
training error

How accurate is the validation set?

How accurate are error estimates based on the validation set?

For any classifier h and underlying distribution D on X x Y:

"true error" $\text{err}(h) = \mathbf{P}_{(x,y)\sim D}(h(x) \neq y)$

"error on set S" $\text{err}(h, S) = \frac{1}{|S|} \sum_{(x,y)\in S} \mathbf{1}(h(x) \neq y)$

Suppose S is chosen i.i.d. (independent, identically distributed) from D. Then (over the random choices of S),

$$\mathbf{E}[\text{err}(h, S)] = \text{err}(h)$$

And the standard deviation of err(h,S) is about $1/\sqrt{|S|}$.

How accurate is the validation set?

Fix any h. Suppose S is chosen i.i.d. (independent, identically distributed) from D. Then (over the random choices of S),

$$\mathbf{E}[\text{err}(h, S)] = \text{err}(h)$$

And the standard deviation of err(h,S) is about $1/\sqrt{|S|}$

(i) In this scenario, S is used to assess the accuracy of a single, pre-specified classifier h.

Can the same S be used to check many classifiers simultaneously?

Answer: VC theory

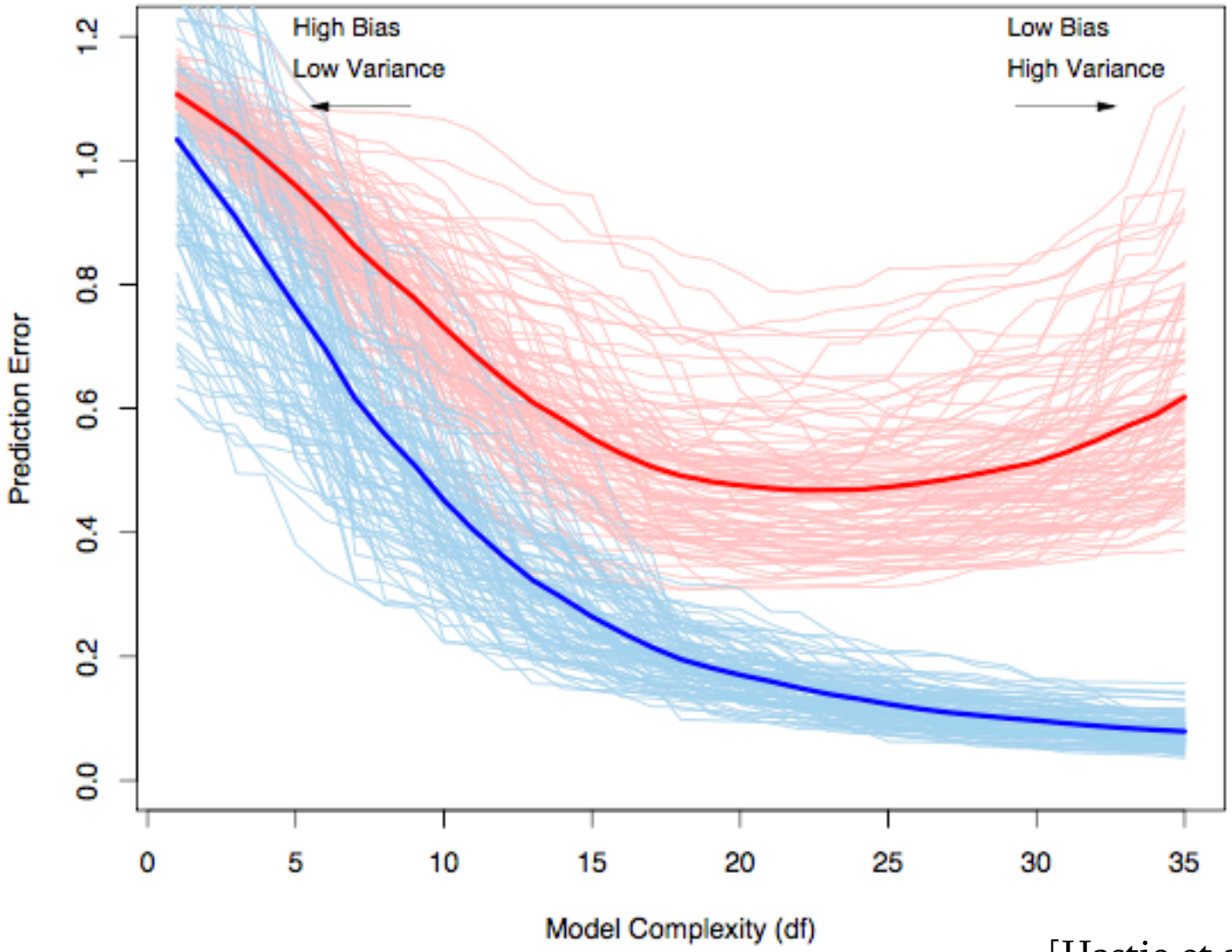
(ii) Note however, if h was created using S as a training set, then the above scenario does not apply. In such situations, err(h,S) might be a very poor estimate of err(h).

Example

Predict flu trends using search data

- X: search data, Y: fraction of population with flu
- S_{train} = all data before 2012
- S_{test} = all data in 2012

Example



Test error

Training error

[Hastie et al., 2001. Fig. 7.1]

Bias-Variance Tradeoff

Allowing the model (hypothesis class) to be more and more complex, the resulting classifier, *h*, will have a better and better fit to the training data, thus the bias of *h* reduces.

But as complexity increases, *h* will increasingly overfit to the chance structure of the particular training data set. So there will be higher variance of the true error of classifiers, *h'*, learned over different training sets.

Bias-Variance Error Decomposition

If we assume all data points (x,y) obey: $y = f(x) + \epsilon$

where: $\mathrm{E}[\epsilon] = 0$

$\mathrm{Var}(\epsilon) = \sigma^2$

Then, $\mathrm{Err_h}(x) = \mathrm{E}[(y - h(x))^2 | X = x]$

$= \sigma^2 + \mathrm{Var}[h] + (\mathrm{Bias}[h])^2$

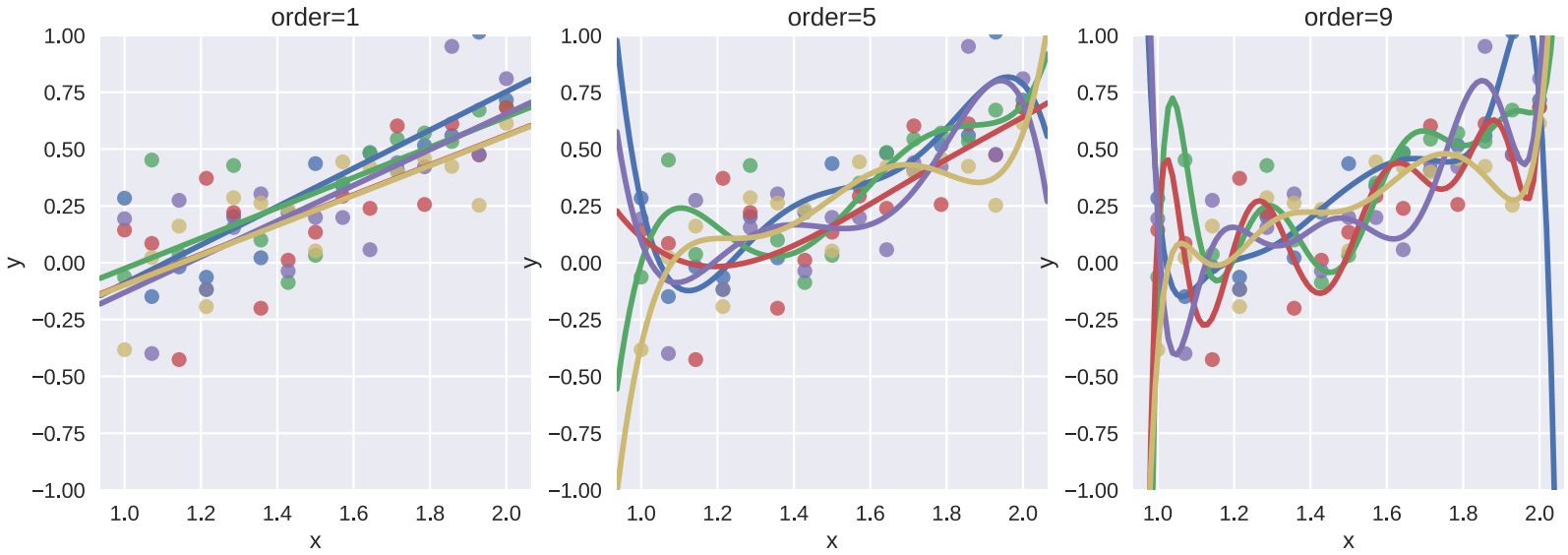
Irreducible error (can't be changed by choice of h)

where: $\mathrm{Var}[h] = \mathrm{E}[h(x)^2] - \mathrm{E}[h(x)]^2$

$\mathrm{Bias}[h] = (f(x) - \mathrm{E}[h(x)])$

NOTE: All expectations are w.r.t. the random training set *h* is learned from, NOT *x*.

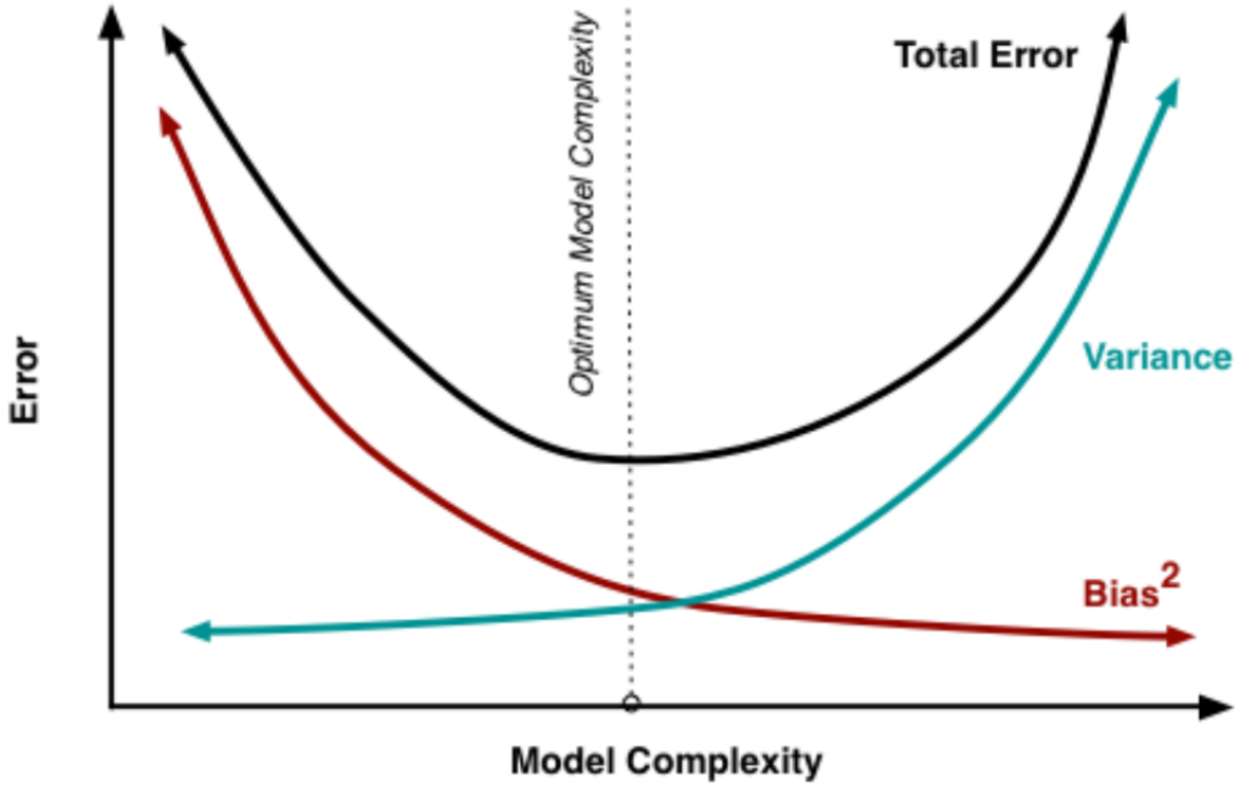
Example



Increasing the order of the polynomial model used to fit the data, increases model complexity eventually leading to overfitting.

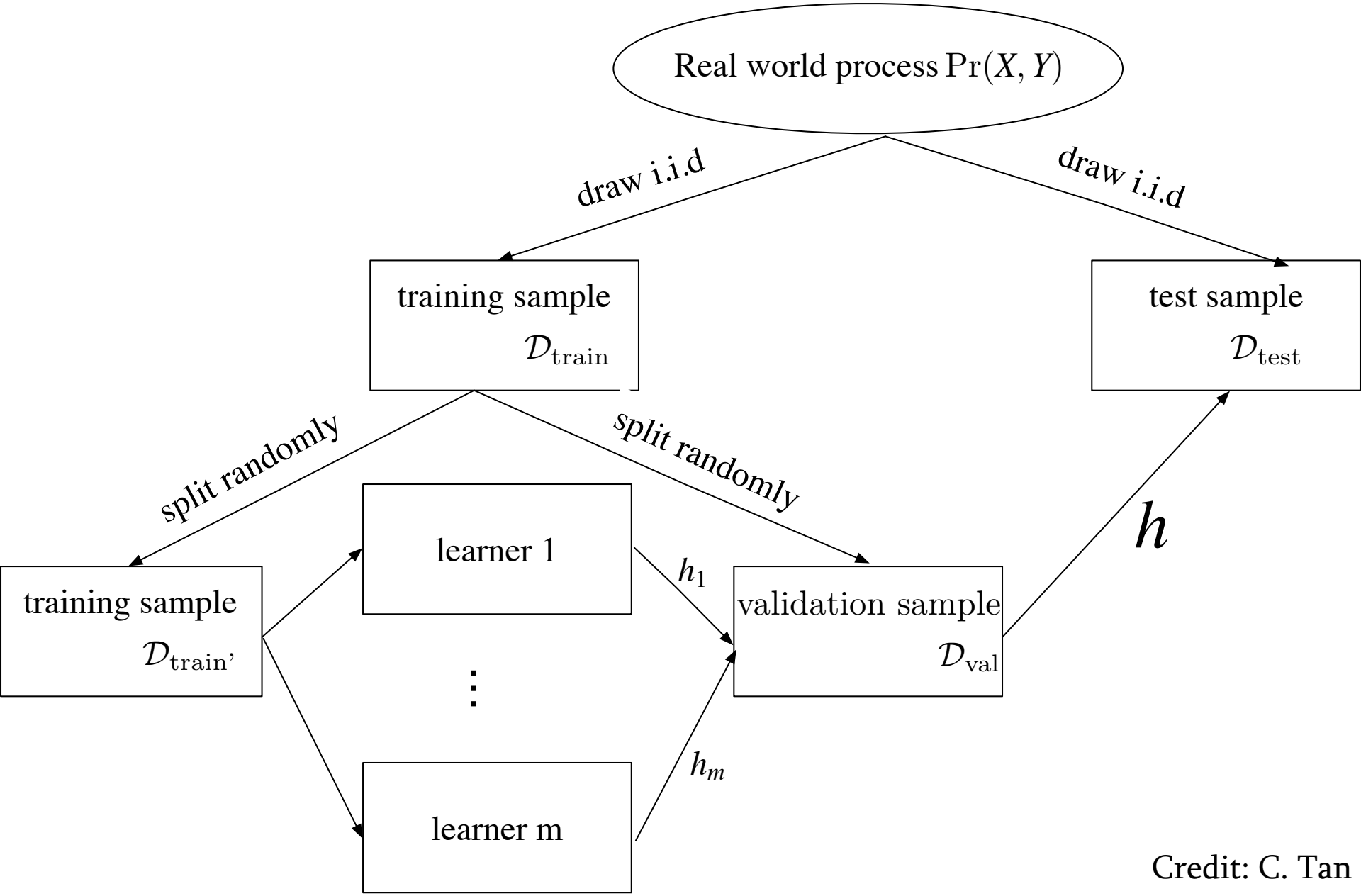
Credit: C. Tan

Bias-Variance Tradeoff



Credit:
http://scott.fortmann-roe.com/docs/BiasVariance.html

Model Selection



Credit: C. Tan

Managing data sets: Train v. Test

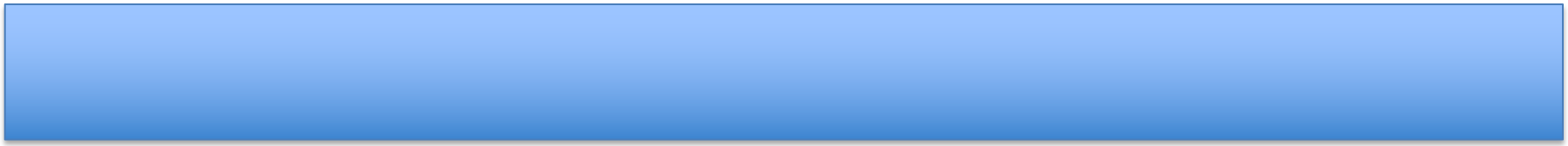
- In general, you should have a separate training set and test set.
- Ideally they will be drawn from the same distribution.
- E.g. randomly permute your entire input data set, then fix T, and use all (x_t,y_t) for $t < T$ for Train, and the remaining data for Test.

Managing data sets: Train v. Test

- First create the holdout data set, the Test set, for computing the final test error. For best performance it will be a (labeled) dataset from the same distribution as the Training set.
  - E.g. technique on previous slide.

- With the remaining data set, you can either divide it into two parts for Train and Validation, as specified on the previous slide, or you can run Cross-Validation on it.
  - This is for model selection / parameter tuning

- Whole data set:



- Separate the test set, and hold it out:

- From the remaining data, use part for Validation:

- Or run cross validation to train and fit parameters:

Cross validation: motivation

- To get a more robust estimate of the error rate, we should run more experiments, and average the error rates over the experiments.

- If we have a huge amount of labeled data, we can run several disjoint experiments (e.g. further split Train and Test into more data sets) and average the test error over the results. This is usually not the case.
  - Instead we do Cross-Validation.

N-fold Cross-validation

- Cross validation data set:

- Fold 1:

- Fold 2:

- …
- Fold n

N-fold cross-validation

- Fix N, for example 5 or 10.
- Run N experiments. In each experiment, 1/N fraction of the data is used for Test and the remaining data is used for Train. The Test sets over the N experiments are disjoint, and the Train sets are overlapping.
- Average the Test error over the N experiments.
- Note: different classifiers will likely be learned in each experiment, as the Train sets differ slightly.

Leave-one-out cross-validation

- N is set to the size of the data set. For each experiment, Test consists of the i-th point, and Train consists of all points but the i-th point.
- Average the test error over the N experiments.

Model selection / Parameter tuning

- For each meta-parameter, e.g. *k* in *k*-NN, explore a range of parameter values. Log-search can be performed by doubling, or halving the value of the parameter.

- On the tuning data set (a.k.a. Validation set, or CV on the training data), try to find the "knee" in the curve: the parameter setting such that the tuning test error is minimized, before increasing again.

Ensemble methods

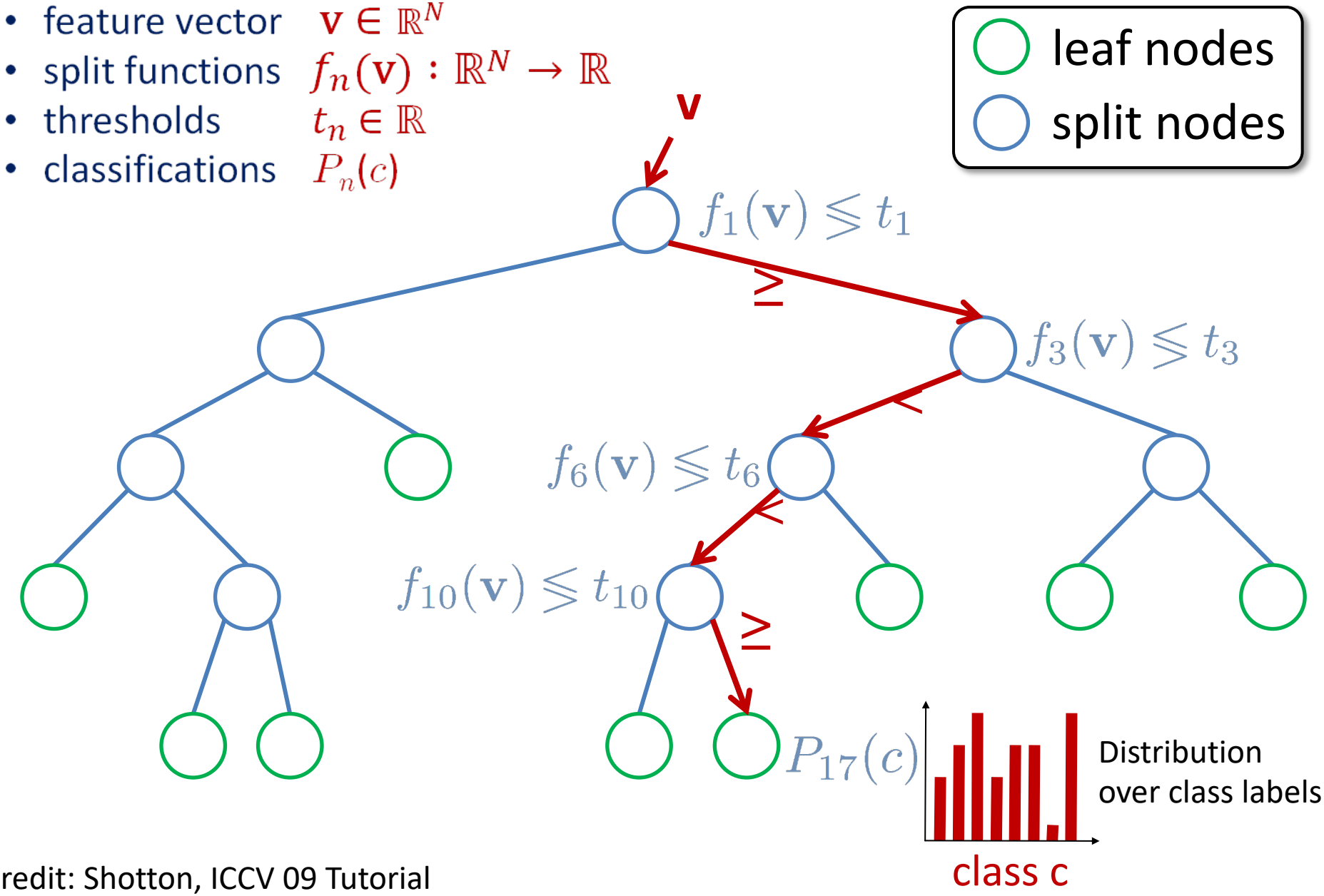
An ensemble classifier combines a set of weak "base" classifiers into a "strong" ensemble classifier.

- "boosted" performance
- more robust against overfitting

- Decision Forests, Random Forests [Breiman '01], Bagging
- Voted-Perceptron
- Boosting
- Learning with expert advice
- ….

Review: (Binary) Decision Trees

- feature vector $\mathbf{v} \in \mathbb{R}^N$
- split functions $f_n(\mathbf{v}) : \mathbb{R}^N \rightarrow \mathbb{R}$
- thresholds $t_n \in \mathbb{R}$
- classifications $P_n(c)$



Credit: Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

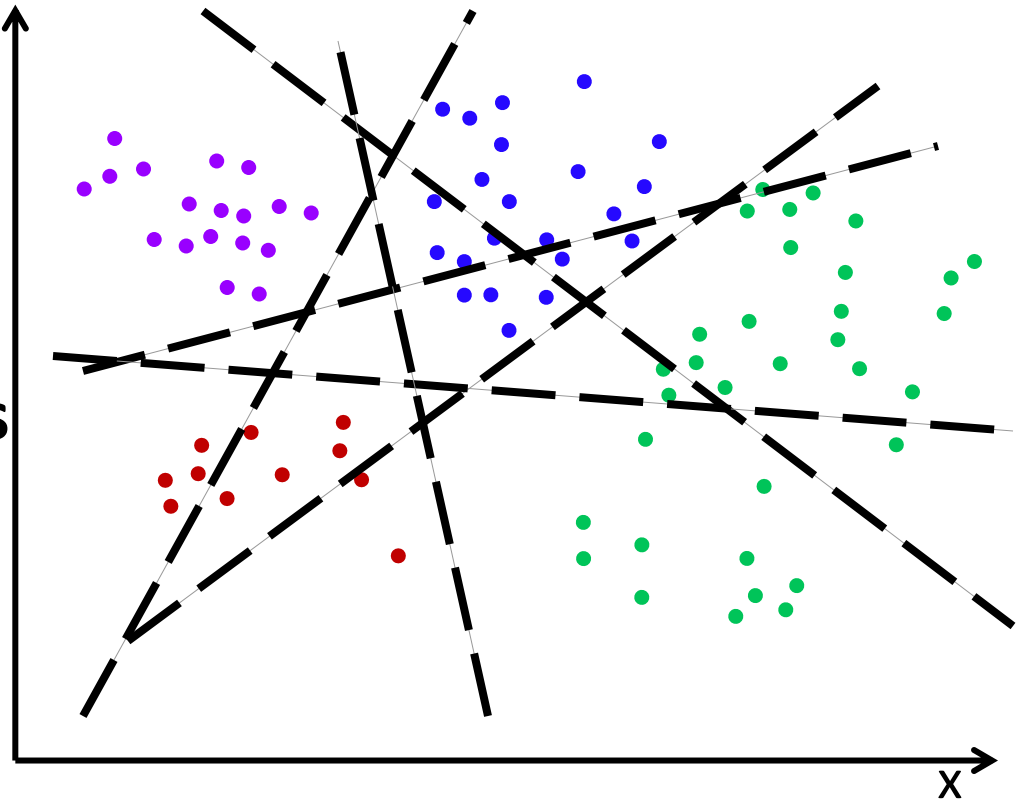
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Animation: Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

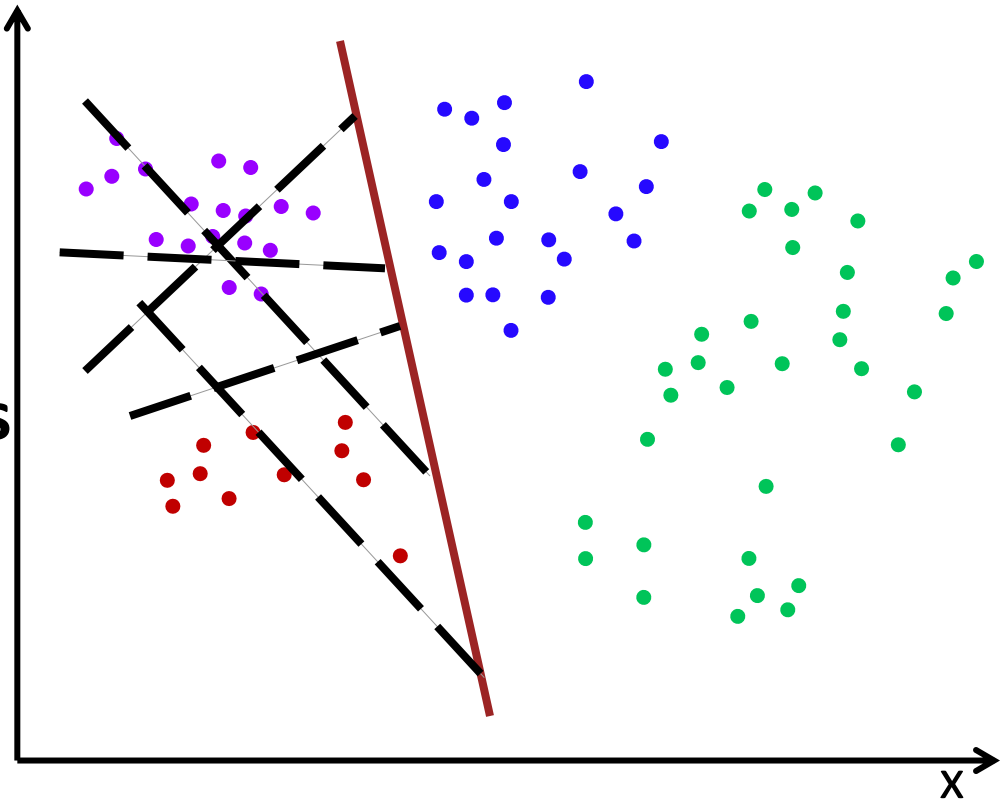
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

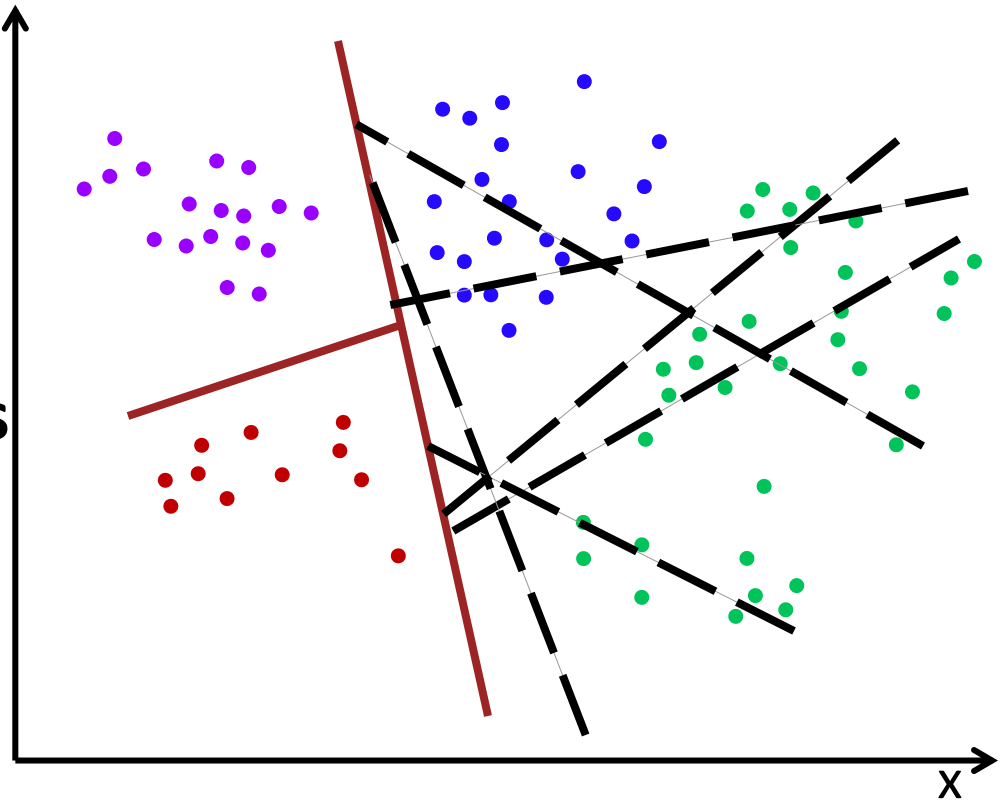
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Shotton, ICCV 09 Tutorial

Speeding up Decision Tree Learning

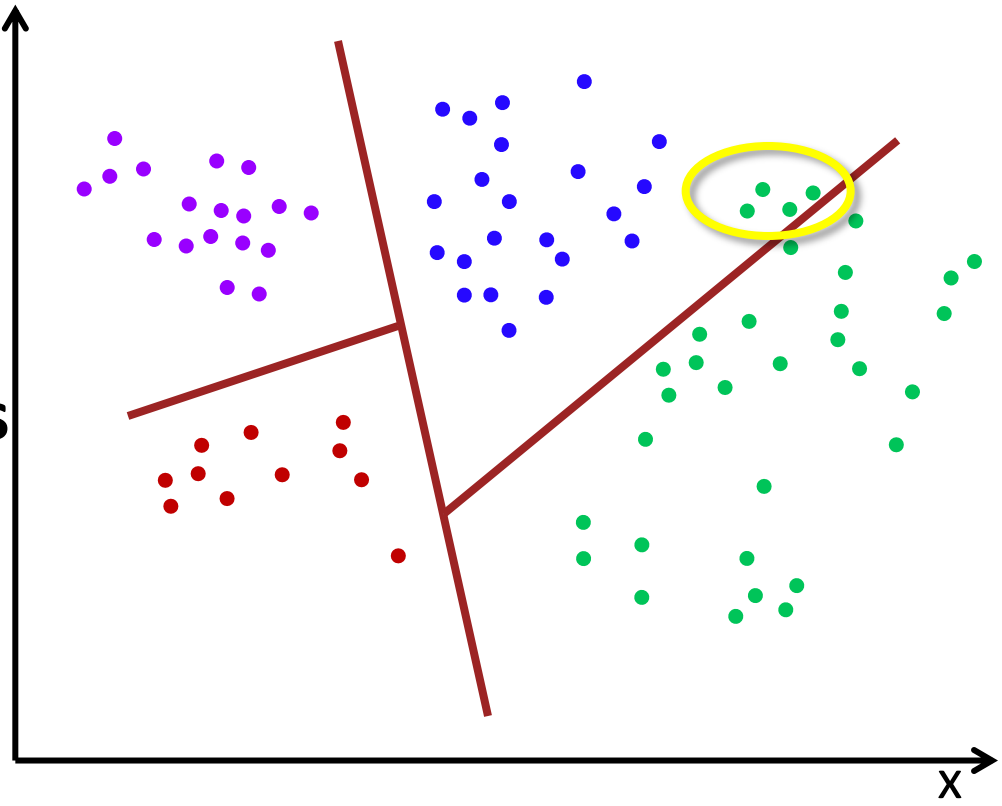
It is cumbersome to test all possible splits

Try several random splits

Keep the split that best separates data

- Reduces uncertainty

Recurse



Random Decision Tree

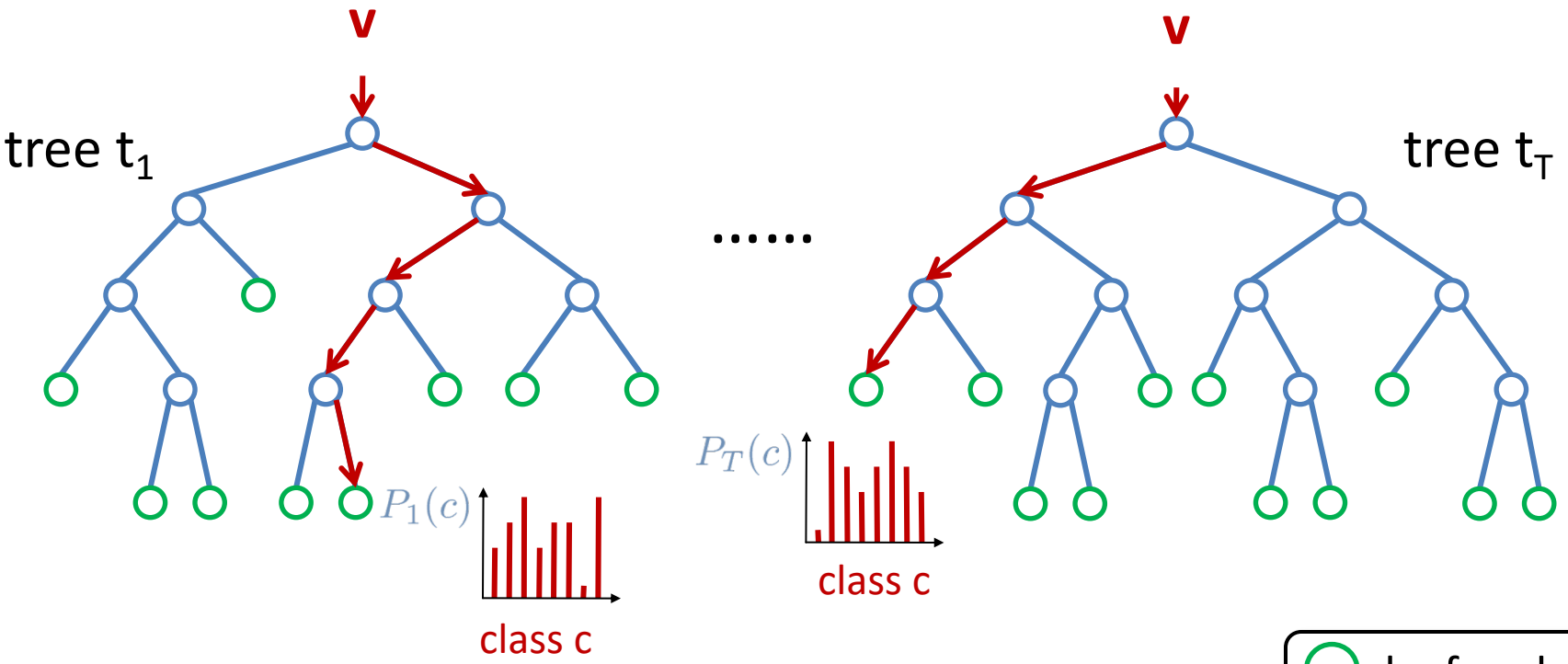
Random Decision Trees

- How many features and thresholds to try?
  - just one: "extremely randomized" [Geurts *et al.* 06]
  - few: fast training, may under-fit
  - many: slower training, may over-fit

- When to stop growing the tree?
  - maximum depth
  - minimum entropy gain
  - threshold changes in class distribution
  - pruning

Decision Forests

- A forest is an ensemble of several decision trees



Classification: $$P(c|\mathbf{v}) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|\mathbf{v})$$

Shotton, ICCV 09 Tutorial

Learning a Forest

- Divide training examples into T subsets S_t
  - improves generalization
  - reduces memory requirements & training time

- Train each decision tree, *t,* on subset S_t
  - same decision tree learning as before

- Easy to parallelize