

Announcements

Quiz today!

On the student feedback

On datasets and models

General questions

Attendance

INM431: Machine Learning

Random forests, bagging, boosting and error decomposition

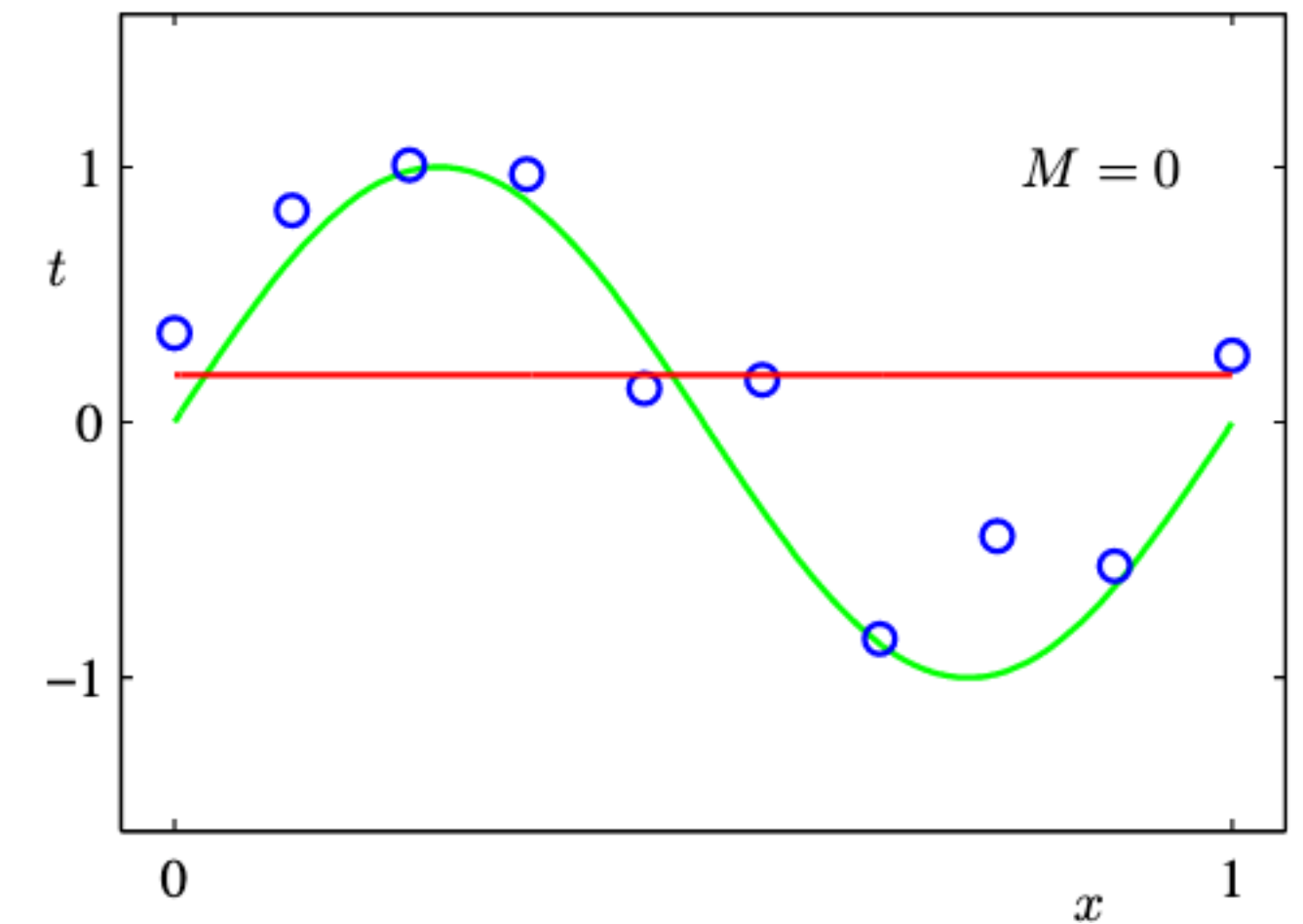
Pranava Madhyastha (pranava.madhyastha@city.ac.uk)

Conceptual understanding of generalisation

Bias - variance tradeoff

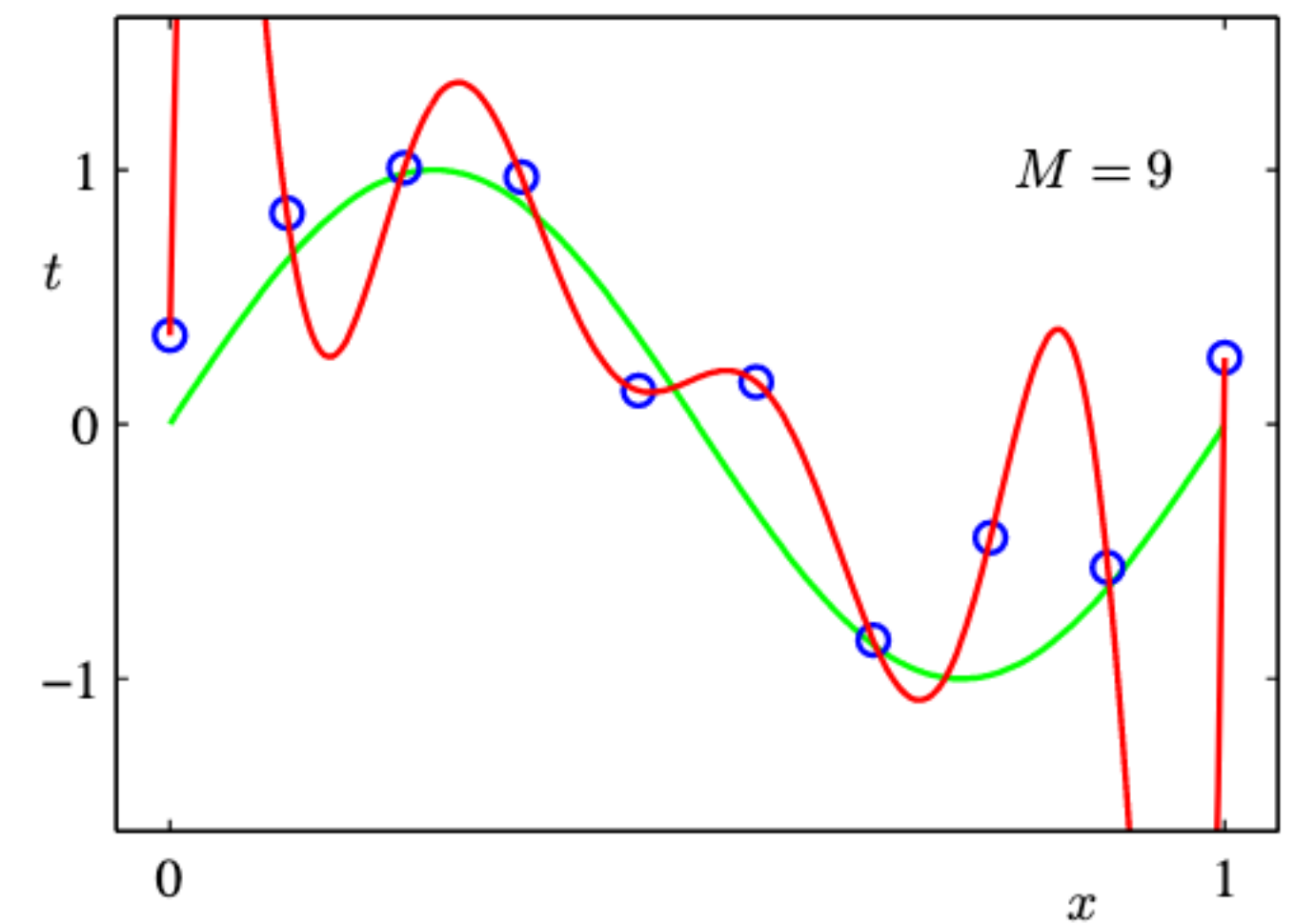
A model too simple: does not fit the data well!

- a biased solution



A model too complex: small changes to the data,
The model changes by a large margin!

- a model with high variance

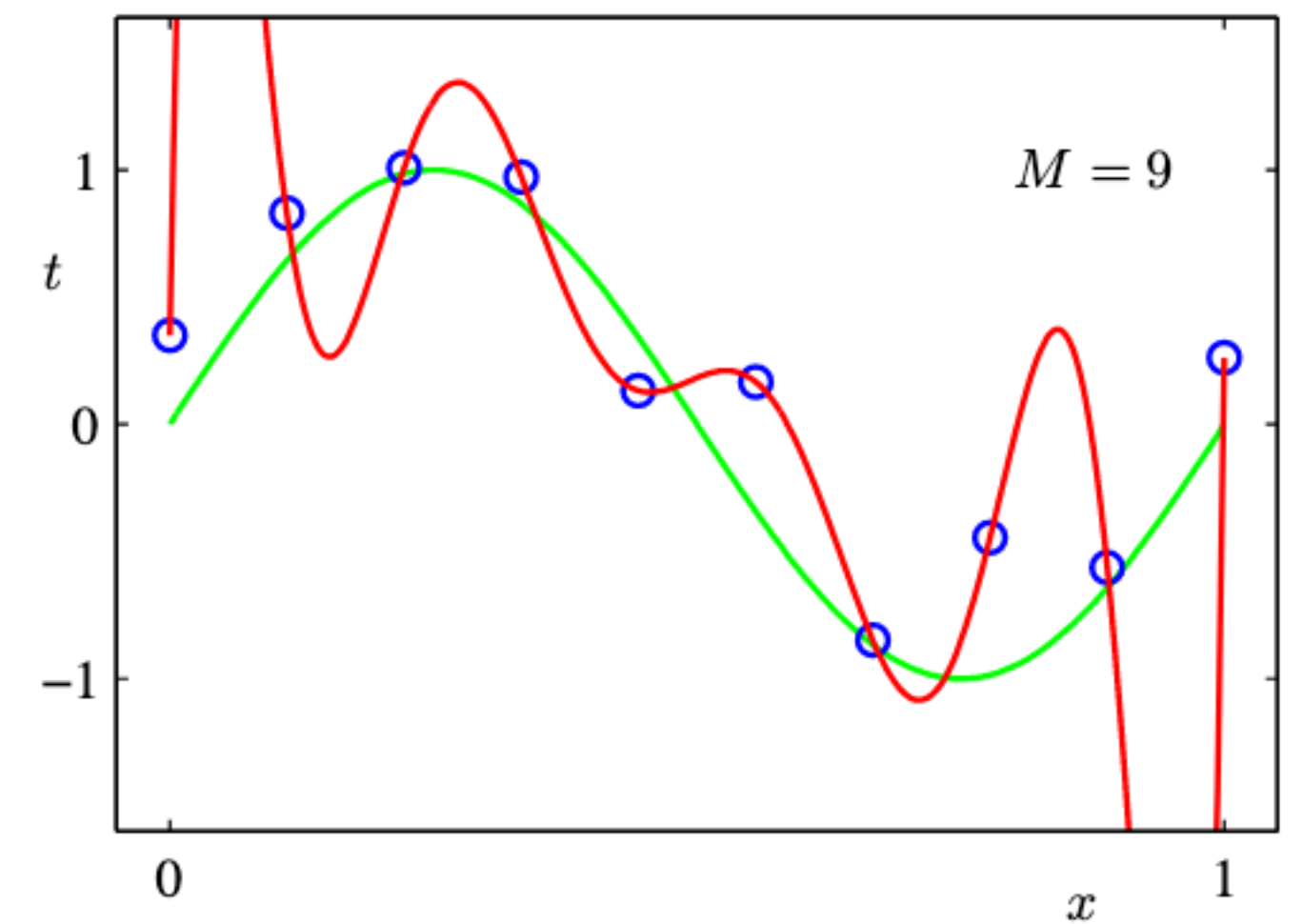
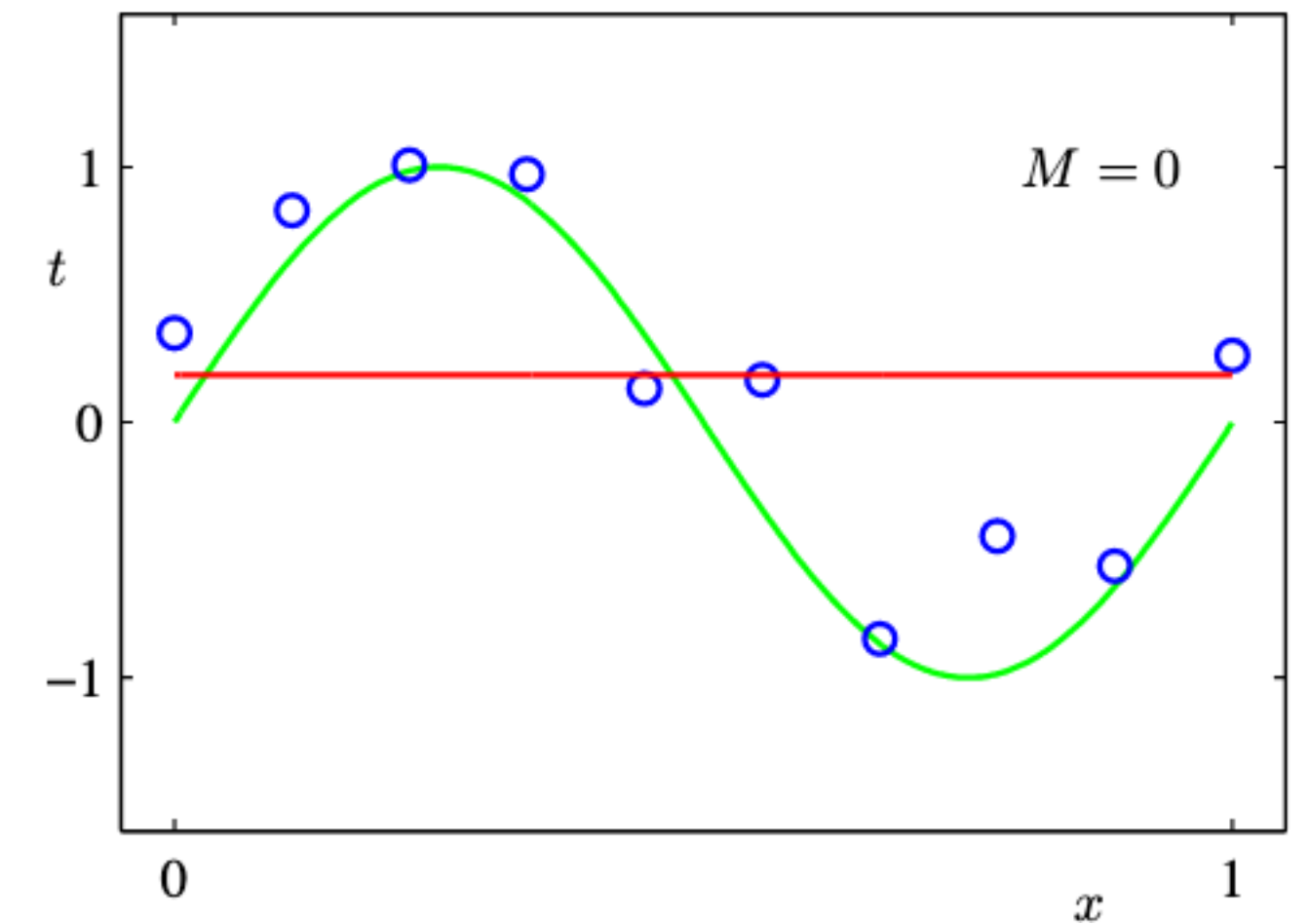


Basics

$D = \left\{ (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, \bar{y}_n) \right\}$ drawn from
some distribution $P(X, Y)$

Assume that we are using a regression model
- it generalises for classification problems too

Assumption: any given input x there might not
exist a unique label y



Basics

$D = \left\{ (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, \bar{y}_n) \right\}$ drawn from some distribution $\mathcal{D}(X, Y)$

Assume that we are using a regression model
- it generalises for classification problems too

Assumption: any given input x there might not exist a unique label y

Say, we have to predict the value of stock for company O and P, they have similar features, but different price!

Expected value ($\forall x \in \mathbb{R}^d$)

$$\bar{y}(\mathbf{x}) = E_{y|\mathbf{x}}[Y] = \int_y y P(y | \mathbf{x}) d y$$

The expected value (or the expected score/label) is the label you would expect to obtain for a feature \mathbf{x} .

ML Algorithm

Say, we have infinite training points and we sample training set D with n samples from distribution \mathcal{D}

We use ML algorithm \mathcal{A} and fit it on the data and learn our model

$$h_D = \mathcal{A}(D)$$

where, h_D is a particular model that is learned on particular dataset D sampled from \mathcal{D}

Generalisation error with squared loss

The generalisation error over the test set is given as:

$$E_{(x,y) \sim \mathcal{D}} \left[\left(h_D(\mathbf{x}) - y \right)^2 \right] = \int_{xy} \left(h_D(\mathbf{x}) - y \right)^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

- D itself is drawn from \mathcal{D} and hence is a random variable
- h_D is also an instance of possible models, hence it is also a random variable

Expected classifier (for a given ML algorithm)

$$\bar{h} = E_{D \sim \mathcal{D}} [h_D] = \int_D h_D P(D) dD$$

here, $P(D)$ is the probability of drawing D from \mathcal{D} ,
 \bar{h} is average over all the possible h_D

(again) Generalisation error

$$E_{D \sim \mathcal{D}} \left[\left(h_D(\mathbf{x}) - y \right)^2 \right] = \int_D \int_{\mathbf{x}} \int_y \left(h_D(\mathbf{x}) - y \right)^2 P(\mathbf{x}, y) P(D) d\mathbf{x} dy dD$$

- the holy grail!

This is an important expression - it helps us understand the quality of the ML algorithm with respect to real data distribution.

Decomposition

$$E_{\mathbf{x},y,D} \left[\left[h_D(\mathbf{x}) - y \right]^2 \right] = E_{\mathbf{x},y,D} \left[\left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right) + \left(\bar{h}(\mathbf{x}) - y \right) \right]^2 \right]$$

adding and subtracting $\bar{h}(x)$

$$= E_{\mathbf{x},D} \left[\underbrace{\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2}_{a^2} \right] + \underbrace{2E_{\mathbf{x},y,D} \left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right) \left(\bar{h}(\mathbf{x}) - y \right) \right]}_{2ab} + \underbrace{E_{\mathbf{x},y} \left[\left(\bar{h}(\mathbf{x}) - y \right)^2 \right]}_{b^2}$$

expanding the expression

Decomposition

$$E_{\mathbf{x},D} \left[\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2 \right] + \underbrace{2E_{\mathbf{x},y,D} \left[\left(h_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right) (\bar{h}(\mathbf{x}) - y) \right]}_{\downarrow} + E_{\mathbf{x},y} \left[(\bar{h}(\mathbf{x}) - y)^2 \right]$$

$$\begin{aligned} & E_{\mathbf{x},y} \left[E_D \left[h_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right] (\bar{h}(\mathbf{x}) - y) \right] \\ &= E_{\mathbf{x},y} \left[\left(E_D \left[h_D(\mathbf{x}) \right] - \bar{h}(\mathbf{x}) \right) (\bar{h}(\mathbf{x}) - y) \right] \\ &= E_{\mathbf{x},y} \left[\underbrace{(\bar{h}(\mathbf{x}) - \bar{h}(\mathbf{x}))}_{0} (\bar{h}(\mathbf{x}) - y) \right] \end{aligned}$$

Decomposition

$$E_{\mathbf{x},y,D} \left[\left[h_D(\mathbf{x}) - y \right]^2 \right] = E_{\mathbf{x},D} \left[\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2 \right] + \underbrace{E_{\mathbf{x},y} \left[\left(\bar{h}(\mathbf{x}) - y \right)^2 \right]}$$

adding and subtracting $\bar{y}(\mathbf{x})$

$$E_{\mathbf{x},y} \left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}) \right) + \left(\bar{y}(\mathbf{x}) - y \right)^2 \right]$$

expanding

$$= E_{\mathbf{x},y} \left[\left(\bar{y}(\mathbf{x}) - y \right)^2 \right] + E_{\mathbf{x}} \left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}) \right)^2 \right] + \underbrace{2E_{\mathbf{x},y} \left[\left(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}) \right) \left(\bar{y}(\mathbf{x}) - y \right) \right]}_0$$

very similar to the earlier step

Decomposition of the error

$$\underbrace{E_{\mathbf{x},y,D} \left[\left[h_D(\mathbf{x}) - y \right]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} \left[(\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$

$$\underbrace{E_{\mathbf{x},y,D} \left[[h_D(\mathbf{x}) - y]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} [(\bar{y}(\mathbf{x}) - y)^2]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} [(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2]}_{\text{Bias}^2}$$

Variance: the difference between what you expect to learn and what you learn from a particular dataset

- Measures how sensitive learner is to specific dataset
- Decreases with simpler model

$$\underbrace{E_{\mathbf{x},y,D} \left[[h_D(\mathbf{x}) - y]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} [(\bar{y}(\mathbf{x}) - y)^2]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} [(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2]}_{\text{Bias}^2}$$

Bias: difference between expected prediction and truth

- Measures how well you expect to represent true solution
- Decreases with more complex model
- Model is biased towards a particular solution

$$\underbrace{E_{\mathbf{x},y,D} \left[[h_D(\mathbf{x}) - y]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} [(\bar{y}(\mathbf{x}) - y)^2]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} [(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2]}_{\text{Bias}^2}$$

Noise: The data-specific noise

- problems due to data distribution/measurement/feature representation
- this is intrinsic to the data
- can't be evaded

More discussion in PRML Chapter 3 (Bishop's book), <https://scott.fortmann-roe.com/docs/BiasVariance.html>, Elements of statistical learning - Chapter 7

Conceptualisation

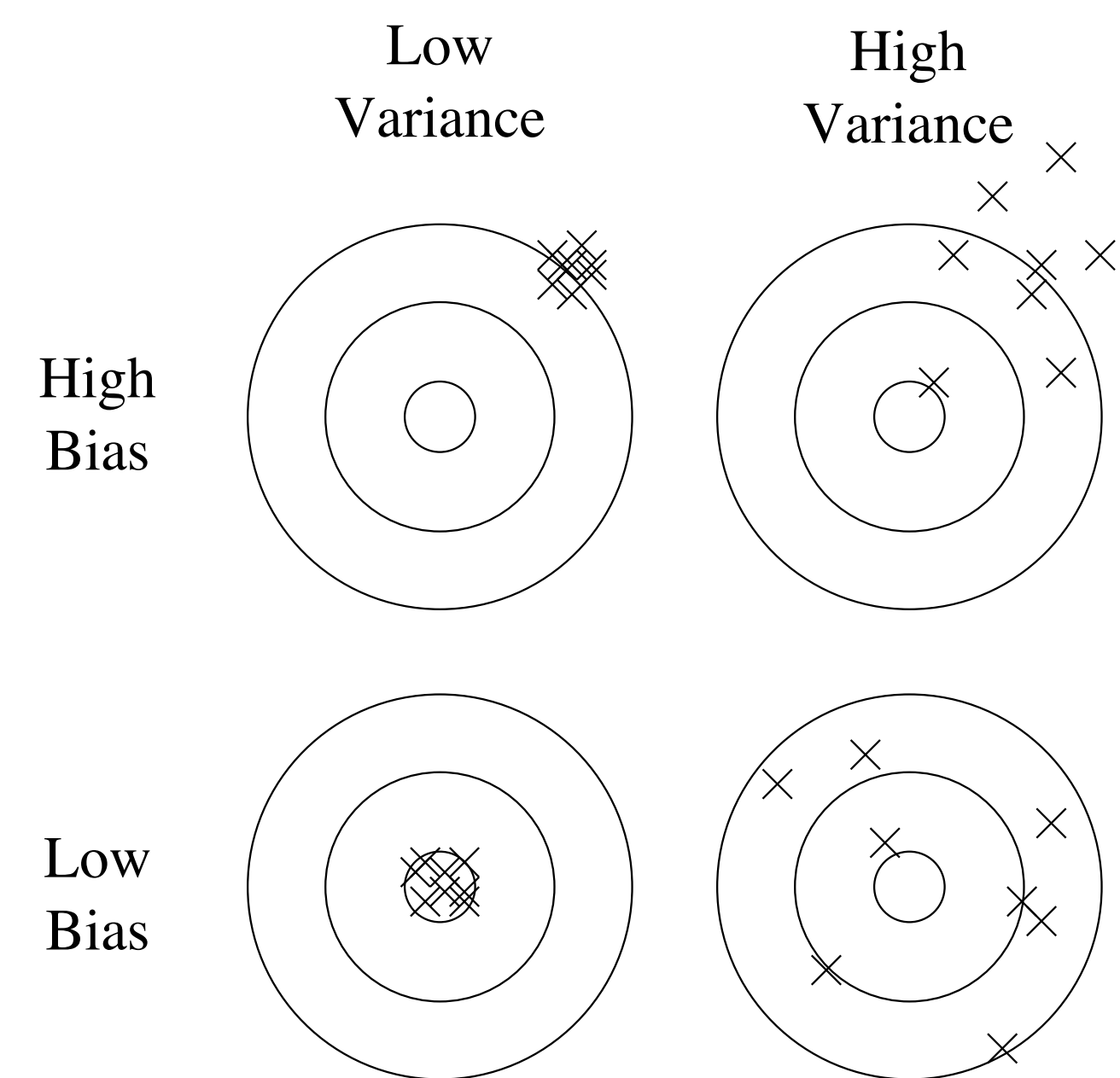


Figure 1: Bias and variance in dart-throwing.

Decision trees (recall)

Decision tree learning performs hill-climbing search (with information gain as a heuristic) over the space of all decision trees, with no backtracking

Best binary partitioning

Bias: shorter trees are preferred; trees that place attributes with higher information gain closest to the root are preferred

Decision trees (recall)

We have seen two measures for deciding when to split!

- Entropy: $-\sum_k p_k \log(p_k)$

- Gini index: $\sum_k p_k(1 - p_k)$

where, $p_k = \frac{|S_k|}{|S|}$

Decision trees (recall)

We saw information gain as a measure:

$$IG(X) = \mathcal{H}(Y) - \mathcal{H}(Y|X)$$

which is the reduction in uncertainty by knowing Y

Information gain: (information before split) - (information after split)

X_1	X_2	Y	<i>Counts</i>
+	+	T	2
+	-	T	2
-	+	F	5
-	-	T	1

Decision trees (recall)

$$IG(X_1, Y) = H(Y) - H(Y|X_1)$$

$$H(Y) = -\frac{5}{10} \log\left(\frac{5}{10}\right) - \frac{5}{10} \log\left(\frac{5}{10}\right) = 1$$

$$H(Y|X_1) = P(X_1 = +)H(Y|X_1 = +) + P(X_1 = -)H(Y|X_1 = -)$$

$$\begin{aligned} &= \frac{4}{10}(1 \log 1 + 0 \log 0) + \frac{6}{10}\left(\frac{5}{6} \log \frac{5}{6} + \frac{1}{6} \log \frac{1}{6}\right) \\ &= 0.39 \end{aligned}$$

$$IG(X_1, Y) = 1 - 0.39 = 0.61$$

X_1	X_2	Y	<i>Counts</i>
+	+	T	2
+	-	T	2
-	+	F	5
-	-	T	1

Decision trees (recall)

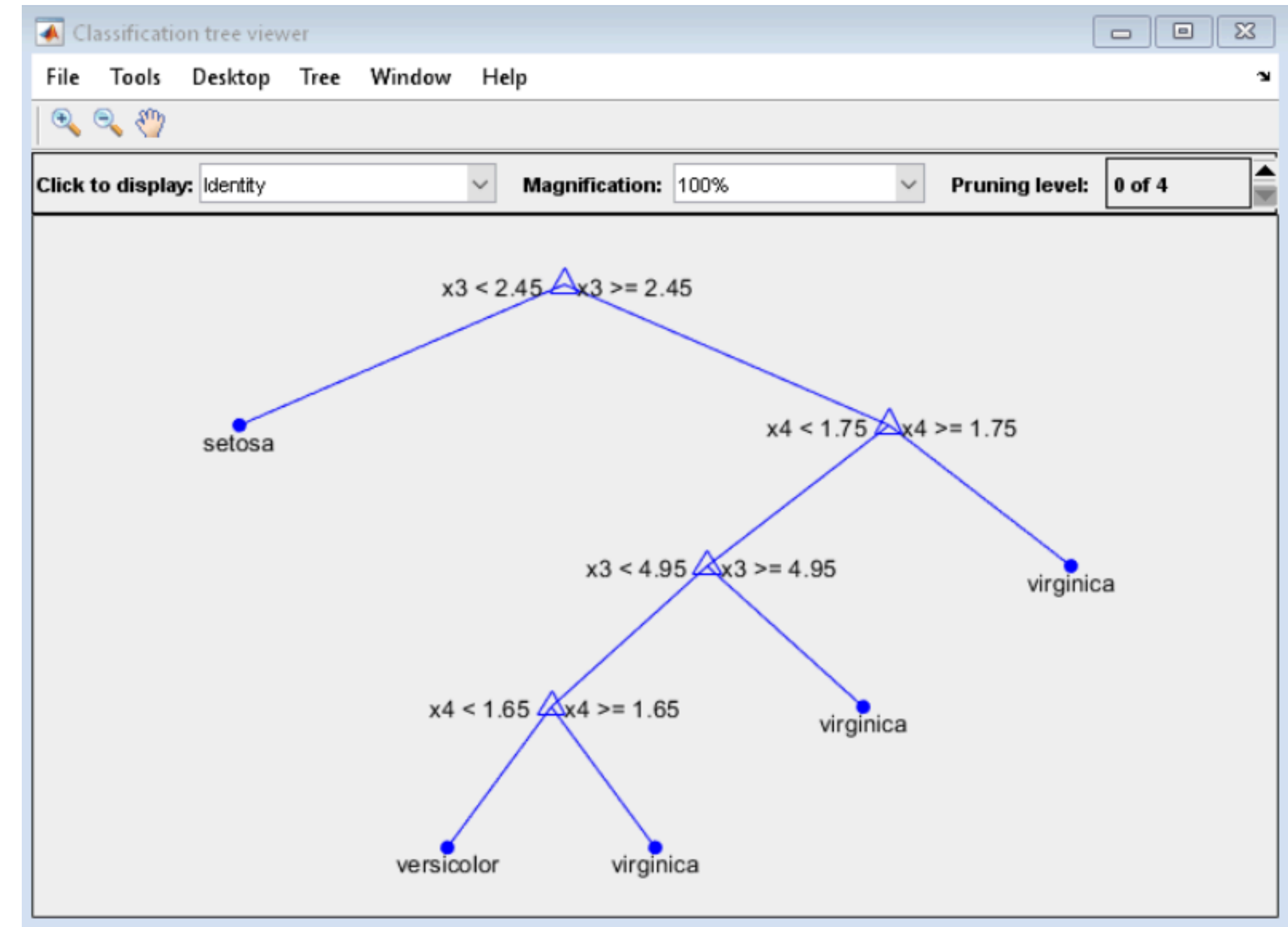
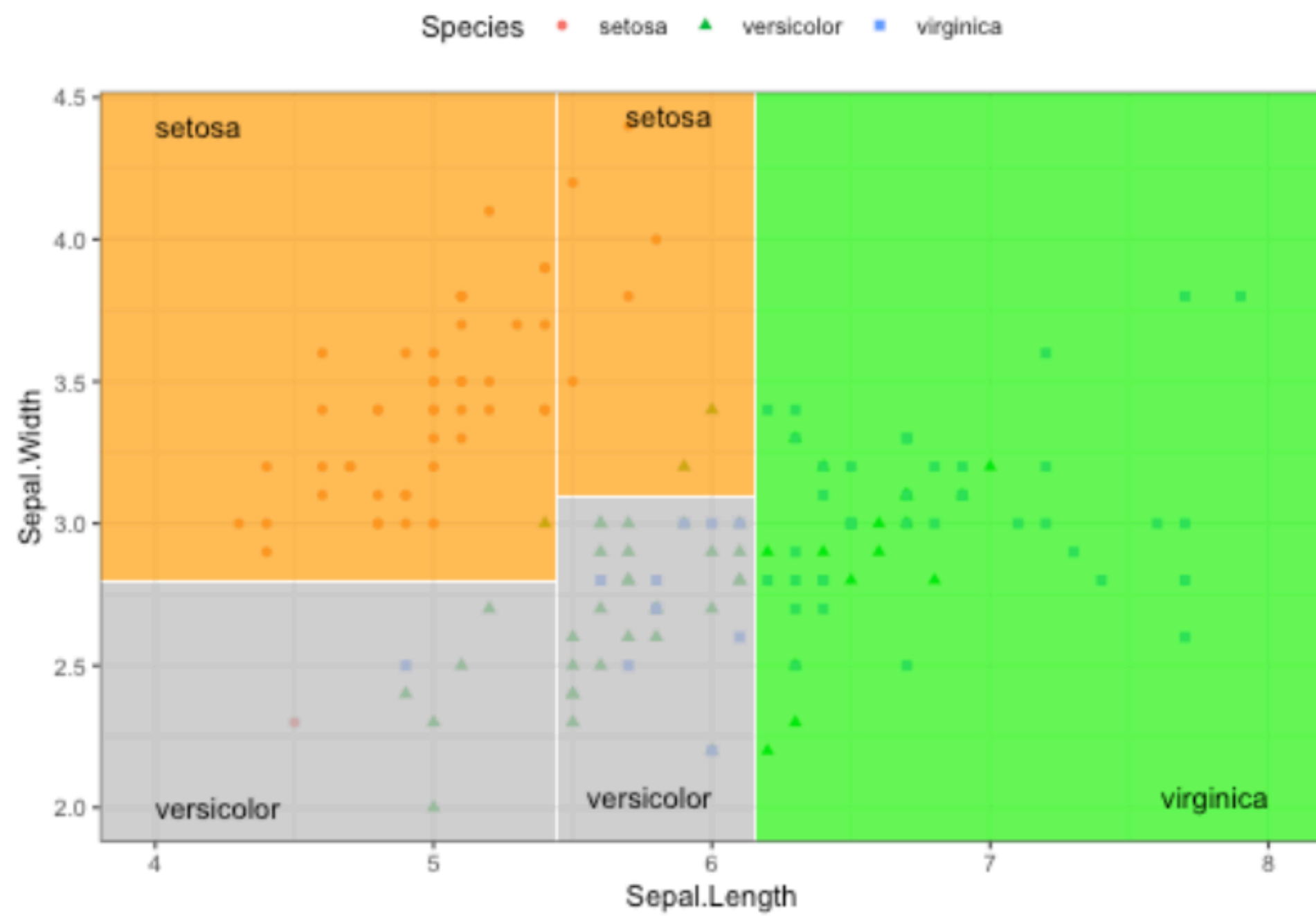
Similar operations:

$$IG(X_2, Y) = 0.12$$

Pick attribute with most $IG \rightarrow X_1$

X_1	X_2	Y	<i>Counts</i>
+	+	T	2
+	-	T	2
-	+	F	5
-	-	T	1

Decision trees (recall)



Decision trees for regression

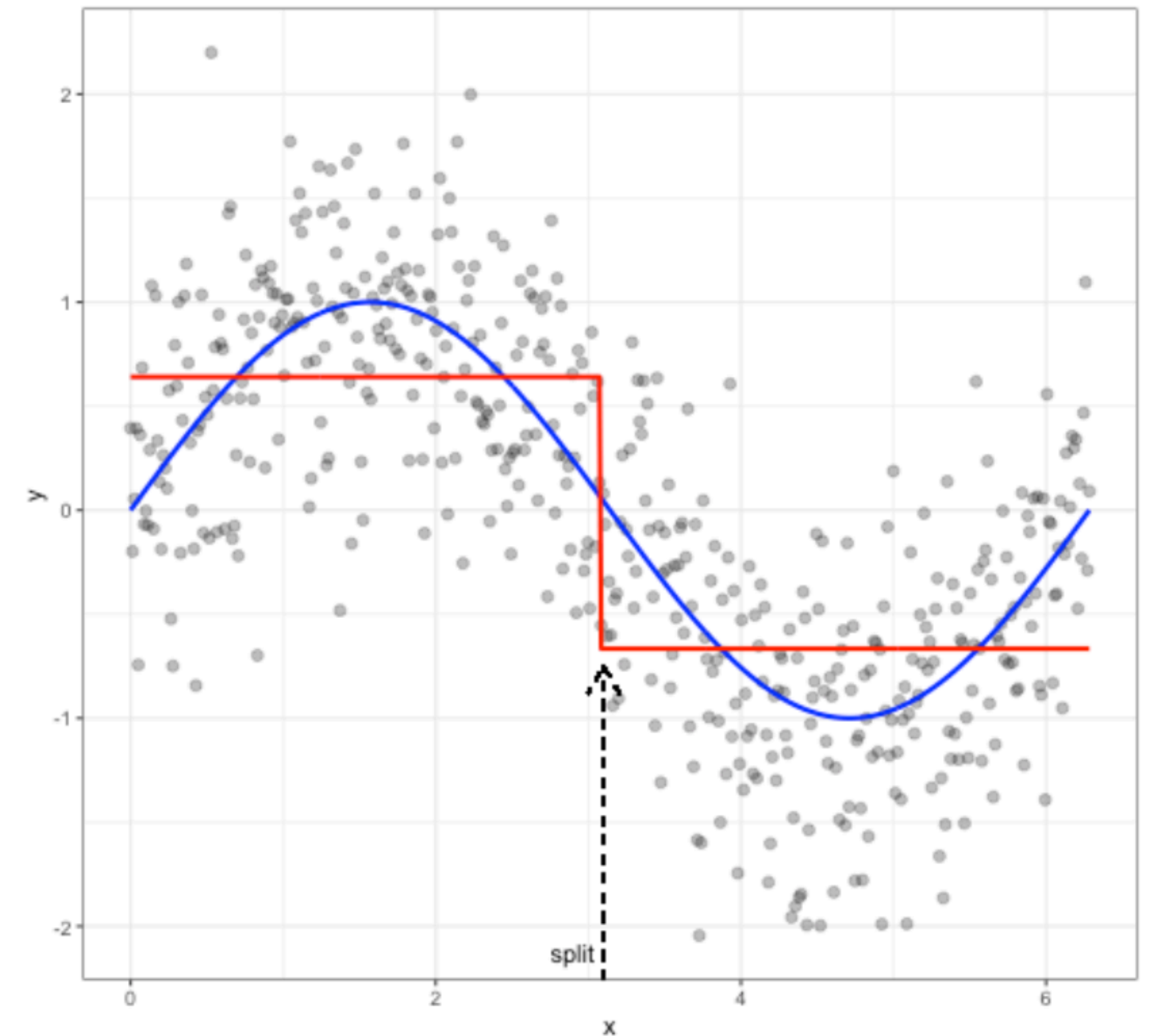
CART trees (classification and regression trees)

PRML Section 14.4

Splitting based on:

$$\mathcal{L}(s) = \frac{1}{|s|} \sum_{(x,y) \in s} (y - \mu)^2$$

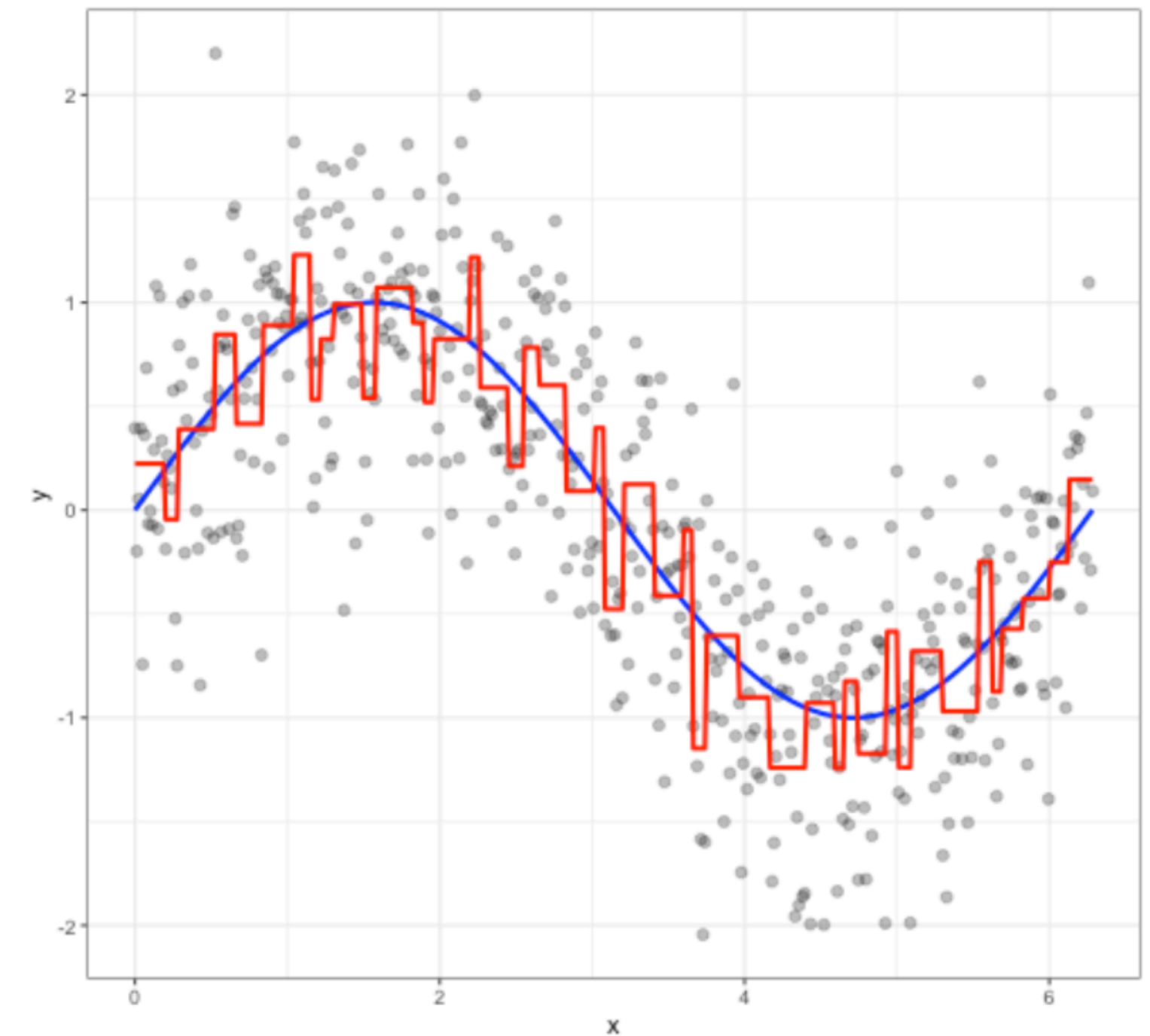
$$\text{where, } \mu = \frac{1}{n} \sum_{(x,y) \in s} y$$



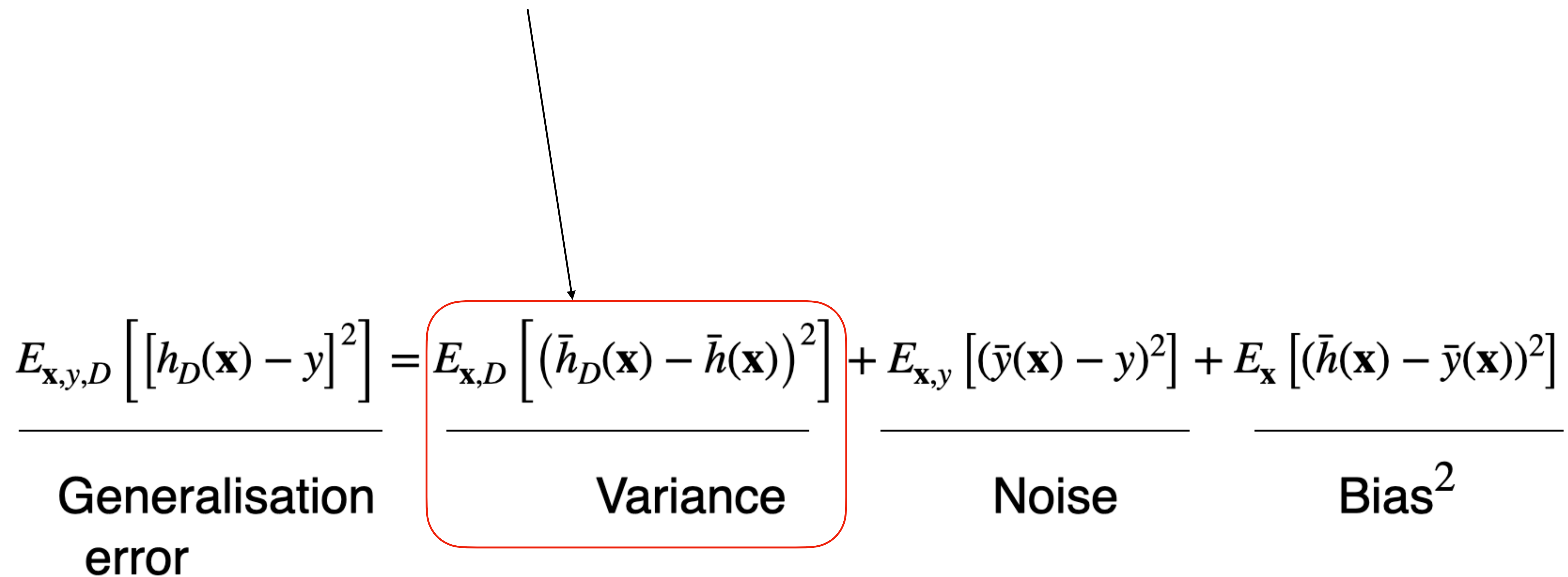
Decision trees for regression

Also known as stumps

Final CART function fitted over the curve.



Decision trees in general


$$\frac{E_{\mathbf{x},y,D} \left[[h_D(\mathbf{x}) - y]^2 \right]}{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} [(\bar{y}(\mathbf{x}) - y)^2]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} [(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2]}_{\text{Bias}^2}$$

Can we reduce the variance?

We saw this earlier:

$$\bar{h} = E_{D \sim \mathcal{D}} [h_D] = \int_D h_D P(D) dD$$

Simplified version:

$$\hat{h} = \frac{1}{m} \sum_{i=1}^m h_{D_i} \rightarrow \bar{h}$$

Ensemble of classifiers

Ensemble of classifiers: if we have a bunch of classifiers and our final result is a combination of classifiers obtained by:

- averaging the results
- based on some version of voting

Can we reduce the variance?

$$\hat{h} = \frac{1}{m} \sum_{i=1}^m h_{D_i} \rightarrow \bar{h}$$

As a consequence:

$$E_{\mathbf{x}, D} \left[\left(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}) \right)^2 \right] \rightarrow 0$$

But, how do we get the $D_1 \cdots D_n$?

Enter 'BAGGING' by bootstrapping

Sample m datasets $D_1 \cdots D_m$ from D
- **sample with replacement**

For each D_j train a classifier h_{d_j}

Final classification result: $\hat{h}(x) = \frac{1}{m} \sum_{j=1}^m h_{D_j}(x)$

Random forests

Bagged decision trees, with a modified splitting criteria

Sample m datasets $D_1 \cdots D_m$ from D

- **sample with replacement**

For each D_j train a decision tree classifier h_{d_j} (max depth is okay) with one constraint: before each split - randomly sub sample $k \leq d$ features (without replacement) and only consider these. (Theoretically $k = \sqrt{d}$)

Final classification result: $\hat{h}(x) = \frac{1}{m} \sum_{j=1}^m h_{D_j}(x)$

Random forests

Prediction is an average of many classifiers:

- We obtain a mean and variance for each decision
- The variance can be interpreted as the uncertainty of the prediction.

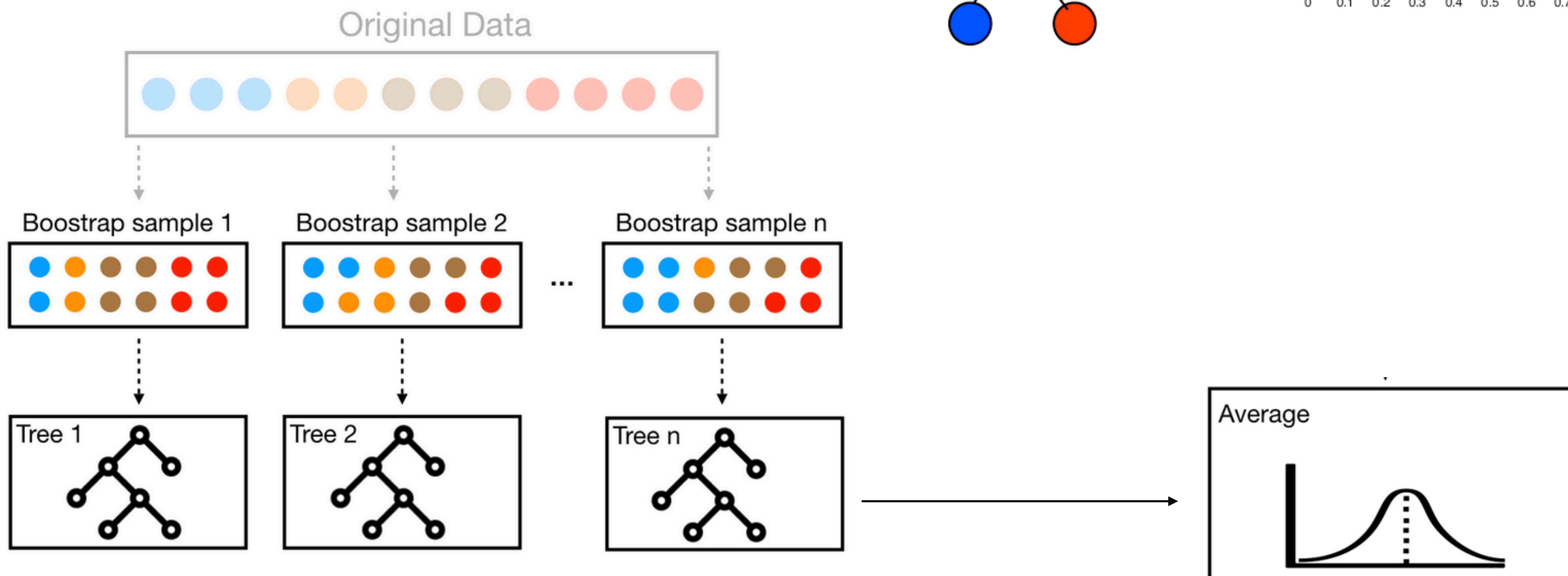
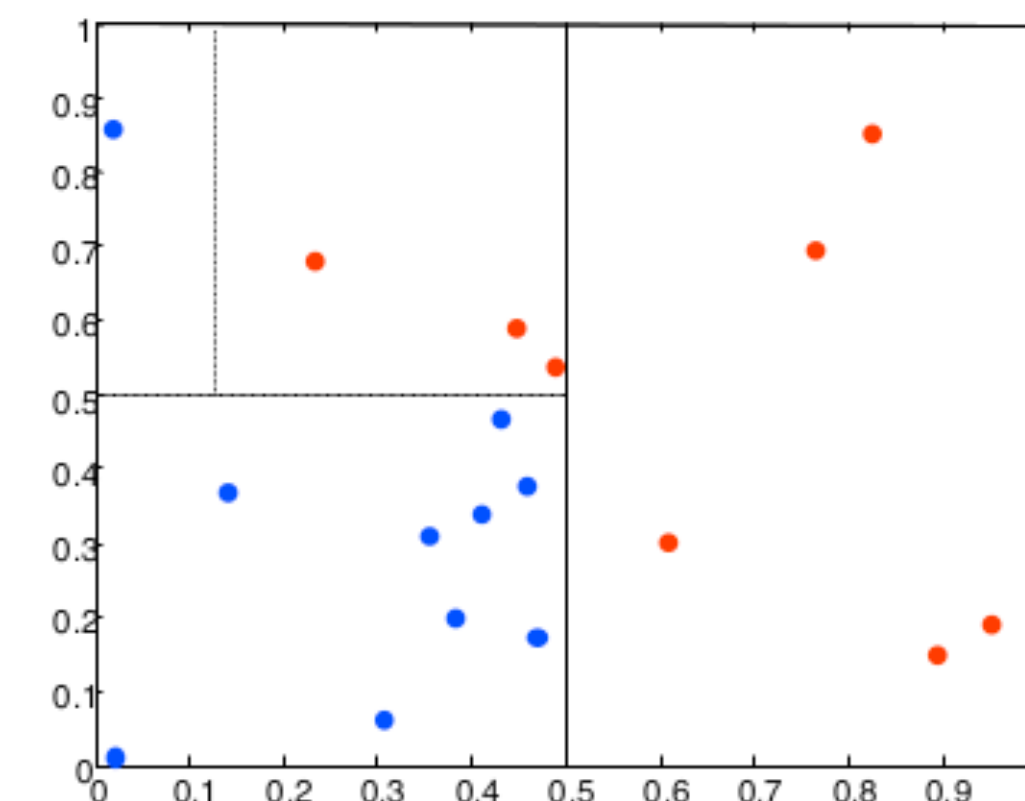
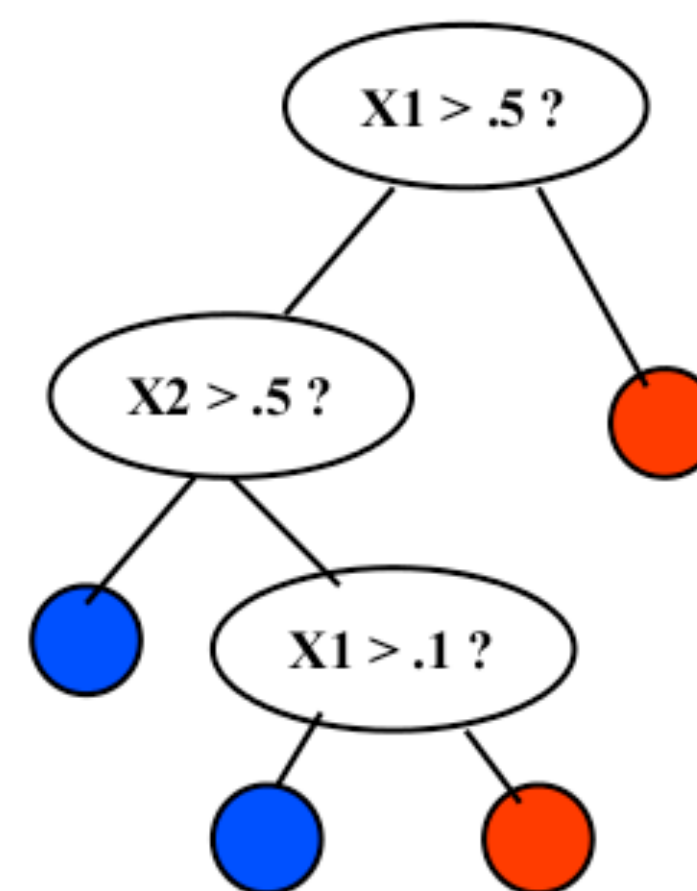
You can also use voting:

- Treat the proportions (voting proportions) as probabilities

In general it is the forests of decision trees!

- very powerful, very general, extremely useful
- can be used for getting feature importance

Random forests



Random forests

One of the most useful algorithms that makes very few assumptions

You can train over the entire training set, without having to create a validation set

Very powerful algorithm, also provide some level of uncertainty estimates

Doesn't require a lot of pre-processing, features can be of different scales, types, etc.,

Boosting

$$\underbrace{E_{\mathbf{x},y,D} \left[[h_D(\mathbf{x}) - y]^2 \right]}_{\text{Generalisation error}} = \underbrace{E_{\mathbf{x},D} \left[(\bar{h}_D(\mathbf{x}) - \bar{h}(\mathbf{x}))^2 \right]}_{\text{Variance}} + \underbrace{E_{\mathbf{x},y} \left[(\bar{y}(\mathbf{x}) - y)^2 \right]}_{\text{Noise}} + \underbrace{E_{\mathbf{x}} \left[(\bar{h}(\mathbf{x}) - \bar{y}(\mathbf{x}))^2 \right]}_{\text{Bias}^2}$$

Boosting

Classifier has a large bias and the training error is high
Weak learner

Q. Can weak learners be combined to generate a strong learner?

A. Create an ensemble of weak learners in an iterative fashion.

Boosting

$$H_T(\vec{x}) = \sum_{t=1}^T \alpha_t h_t(\vec{x})$$

Iterative addition of weak learners

At iteration t add a new weak learner $h_t(x)$ with a step-size α_t (a constant)

Test time, return the weighted sum!

Instead of learning a single classifier, learn many weak classifiers that are good at different parts of the input space

Boosting

This is similar to gradient descent!

Instead of updating the parameters or the weight vectors/matrices, we are directly adding functions.

It's gradient descent in the functional space

Boosting

The effective loss function:

$$\ell(H) = \frac{1}{n} \sum_{i=1}^n \ell \left(H(\mathbf{x}_i), y_i \right)$$

The new h_{t+1} is written as:

$$h_{t+1} = \operatorname{argmin}_{h \in \mathbb{H}} \ell \left(H_t + \alpha h_t \right)$$

Boosting (Shapire, 1989)

Given a weak learner, run it multiple times (reweighted) training data, then let learned classifiers vote

On each iteration t :

- weight each training example by "how incorrectly" it was classified
- Learn a weak classifier
- A strength for this hypothesis

Next week

A bit more on boosting

K-Means clustering and a bit of K-Nearest Neighbours

Introduction to mixture models