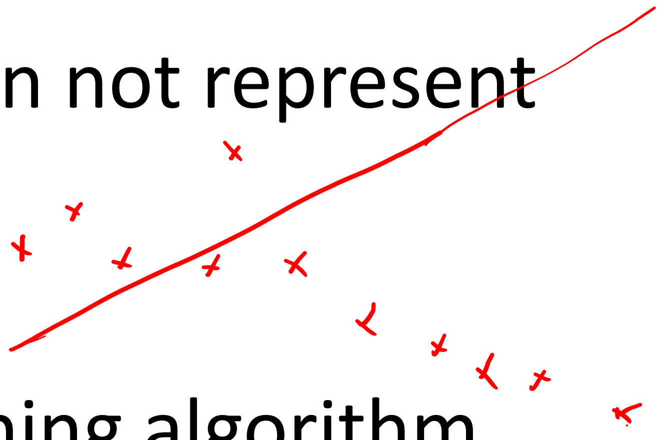
Model Evaluations

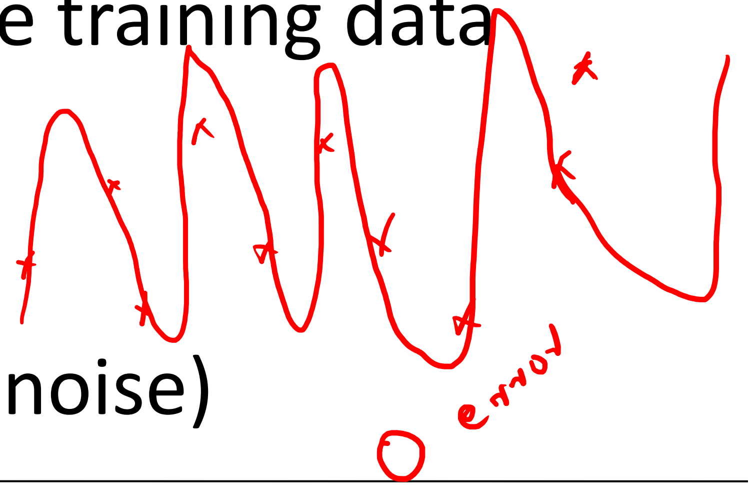
Dr. Rahul Kottath

Bias and Variance

- Bias – error caused because the model can not represent the concept

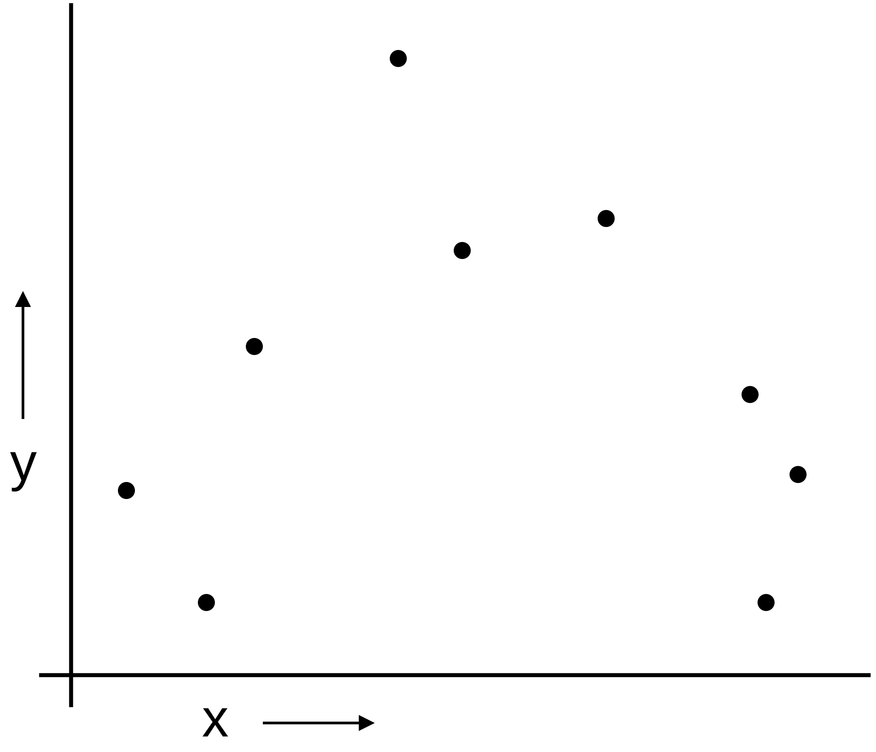


- Variance – error caused because the learning algorithm overreacts to small changes (noise) in the training data



TotalLoss = Bias + Variance (+ noise)

A Regression Problem



$y = f(x)$

y = f(x) + noise

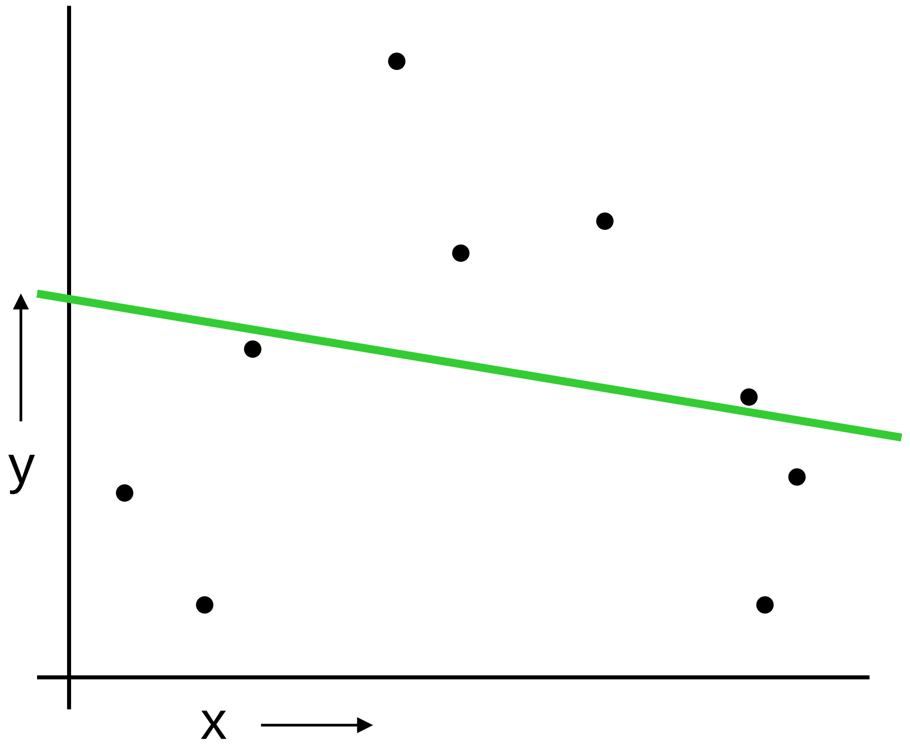
Can we learn f from this data?

Let's consider three methods…

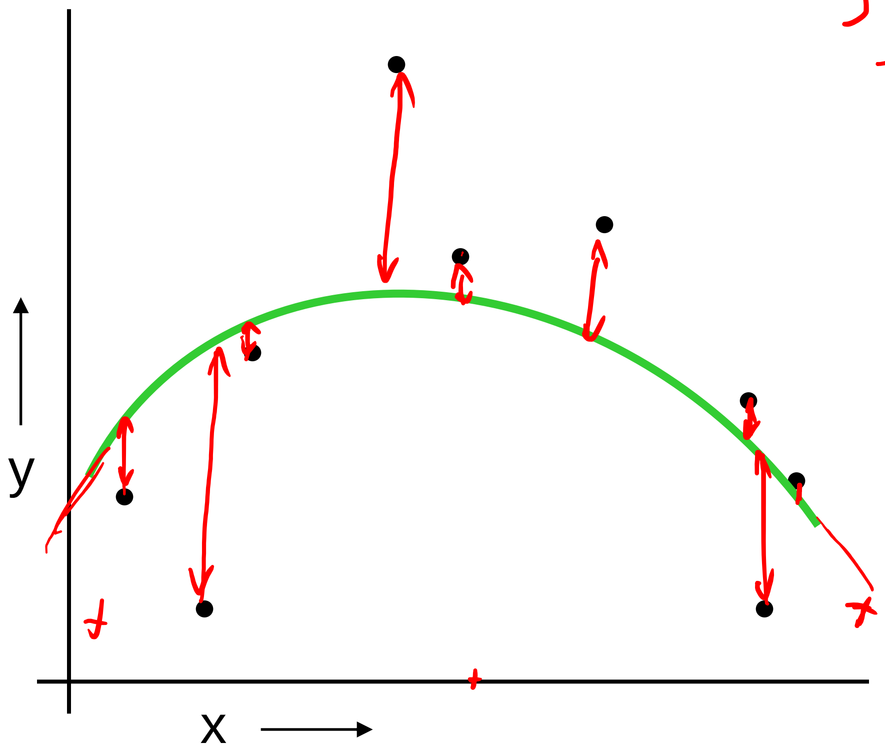
Linear Regression

$y = mx + c$

x^2



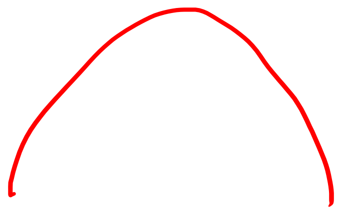
Quadratic Regression



$y = ax^2 + bx + c$

Σe = error

error > 0

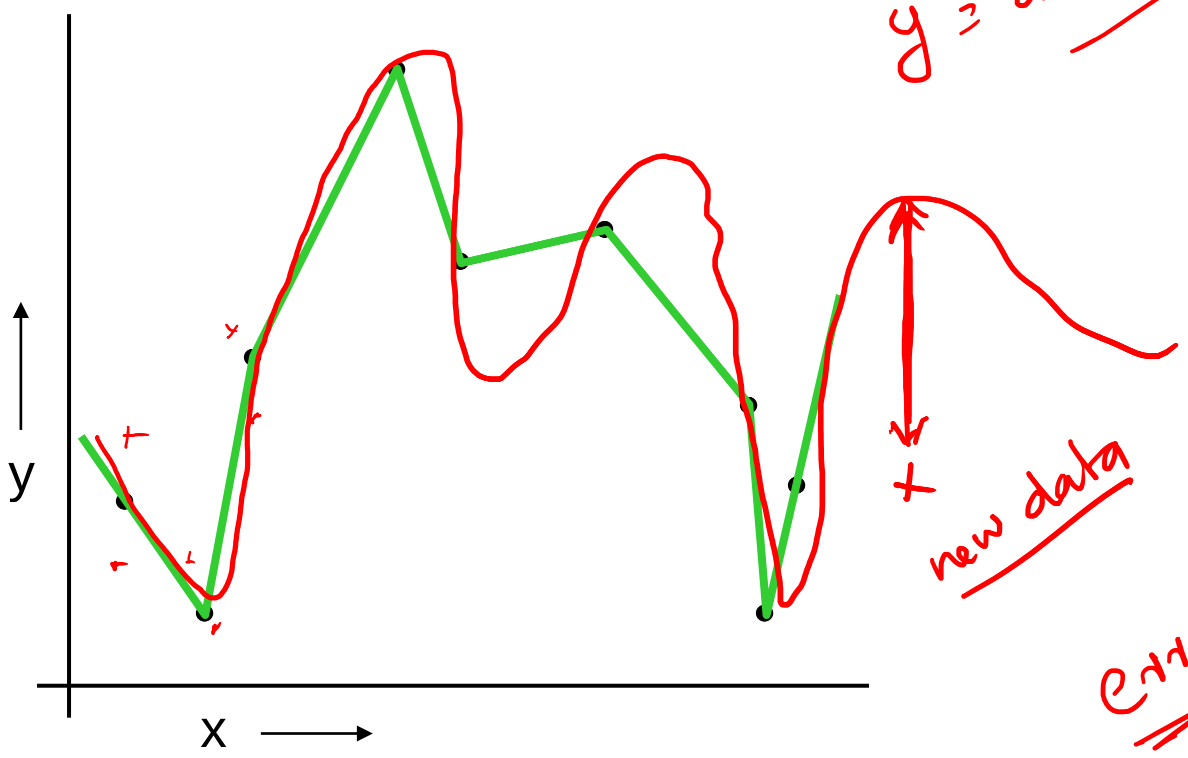


Join-the-dots

10 degree polynomial

$y = ax^{10} + bx^{9} + \cdots + k$

$y = f(x) + \text{noise}$



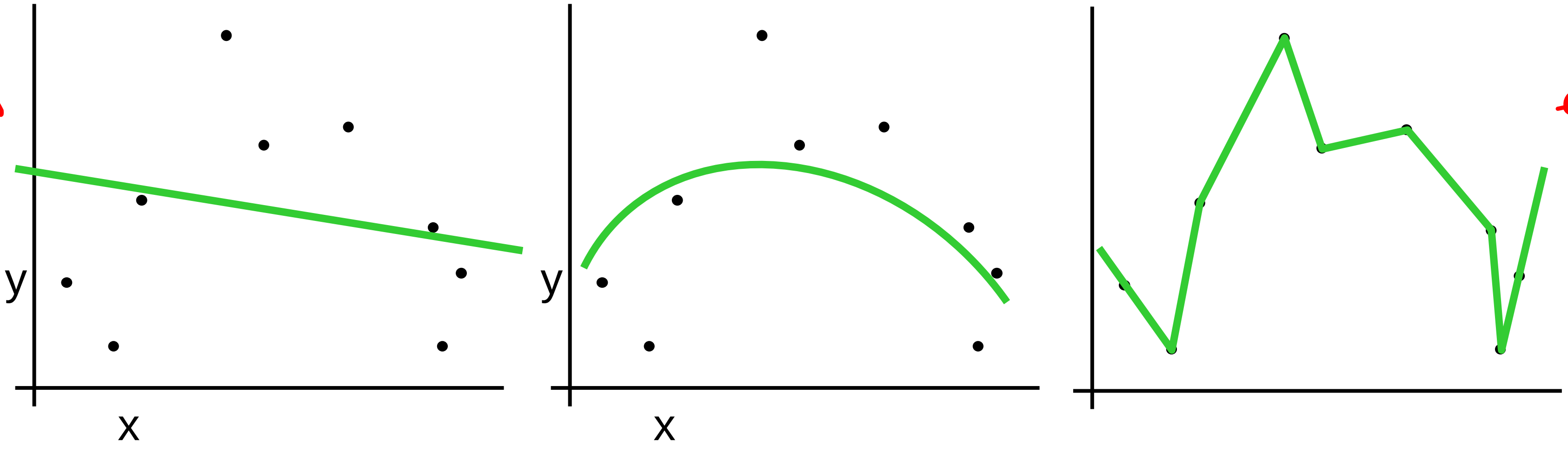
Also known as piecewise linear nonparametric regression if that makes you feel better

error ≈ 0

Which is best?

moderate training error +
moderate test error

high training error +
high test error



error ≈ 0

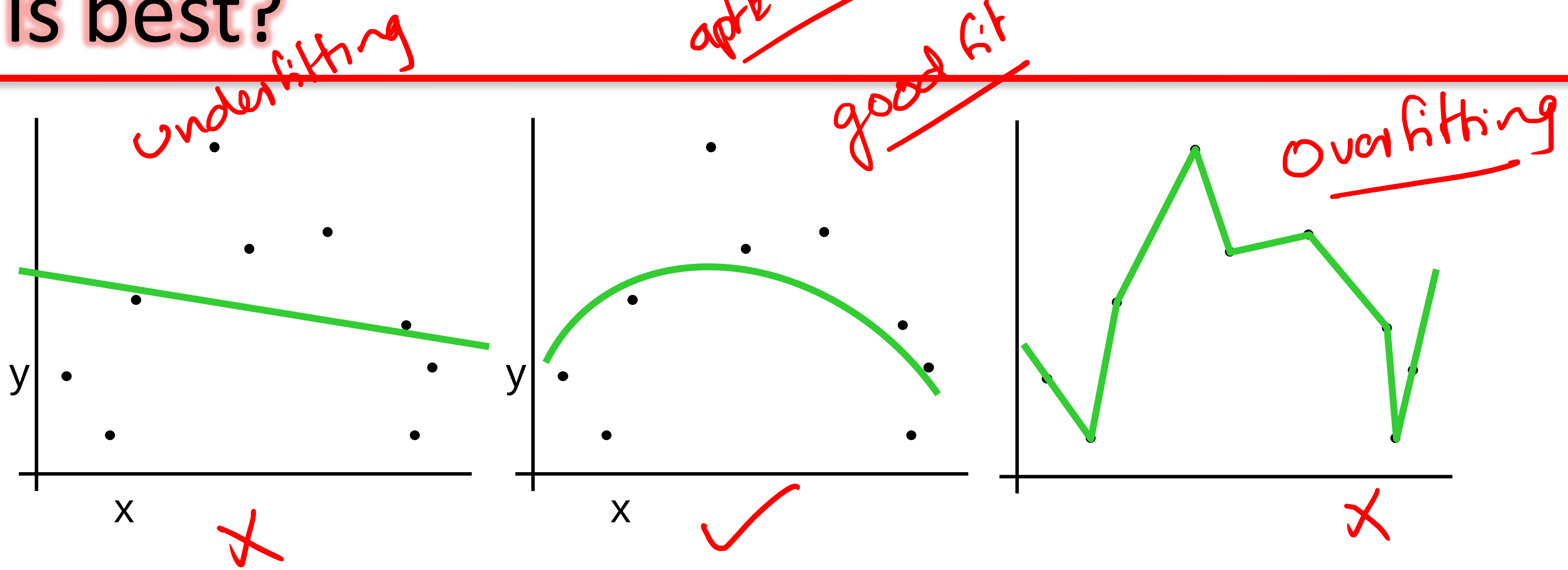
≈0 training error
high test error

Why not choose the method with the best fit to the data?

dataset
training   test
model

Which is best?

apt learning

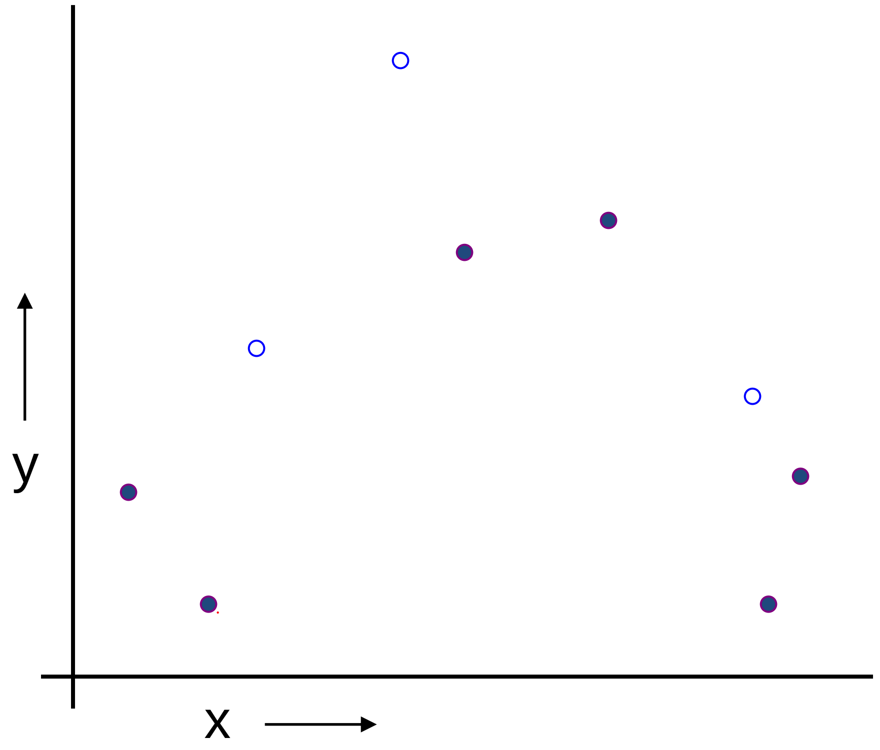


Why not choose the method with the best fit to the data?

"How well are you going to predict future data drawn from the same distribution?"

The test set method

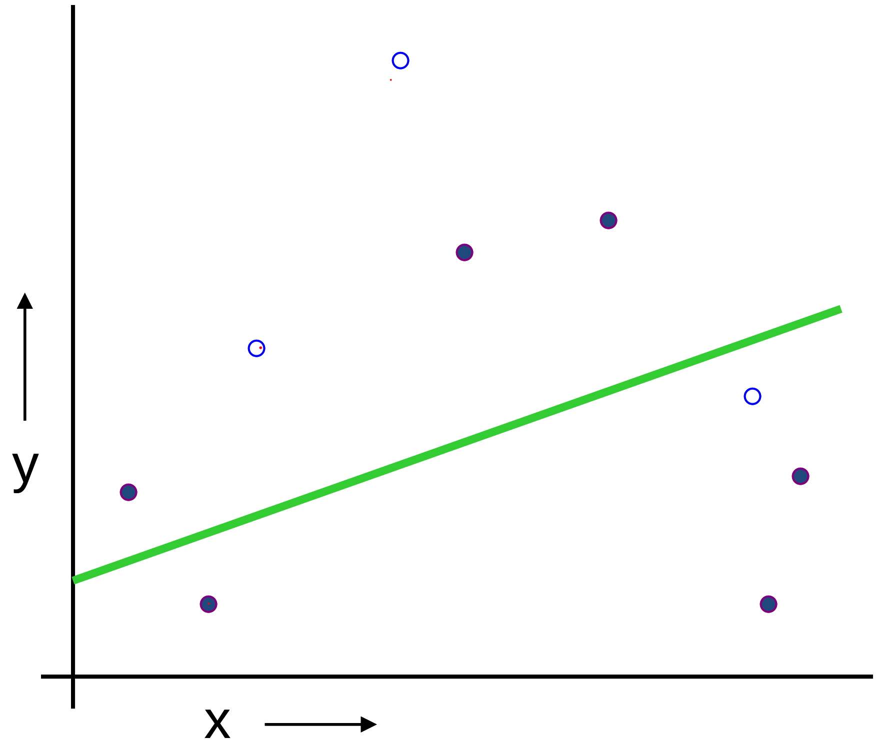
dataset → 70% training
→ 30% testing



1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

The test set method



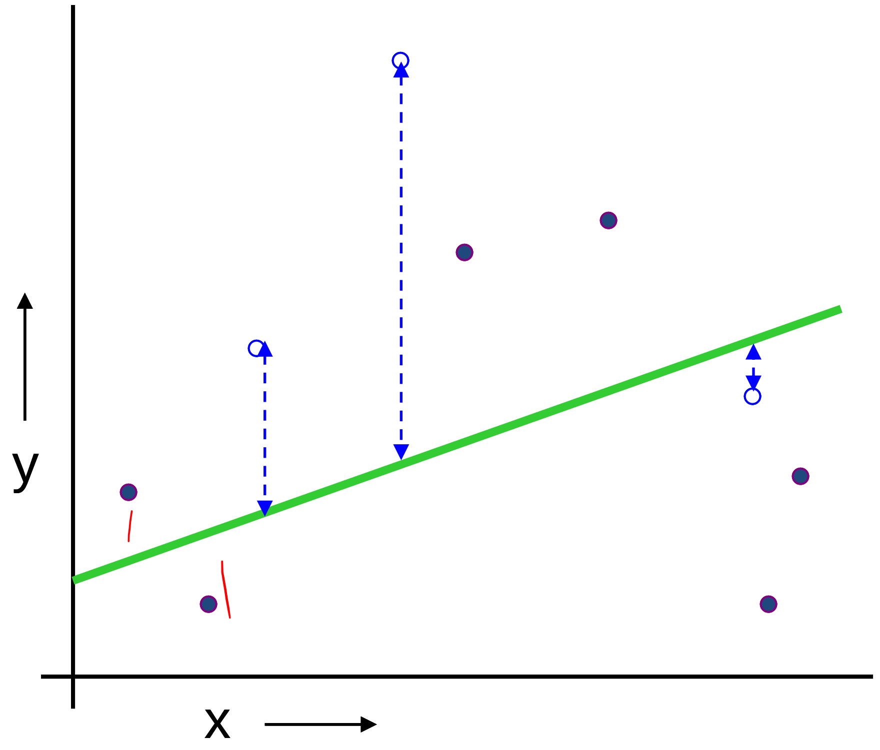
(Linear regression example)

1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

The test set method



(Linear regression example)

Mean Squared Error = 2.4

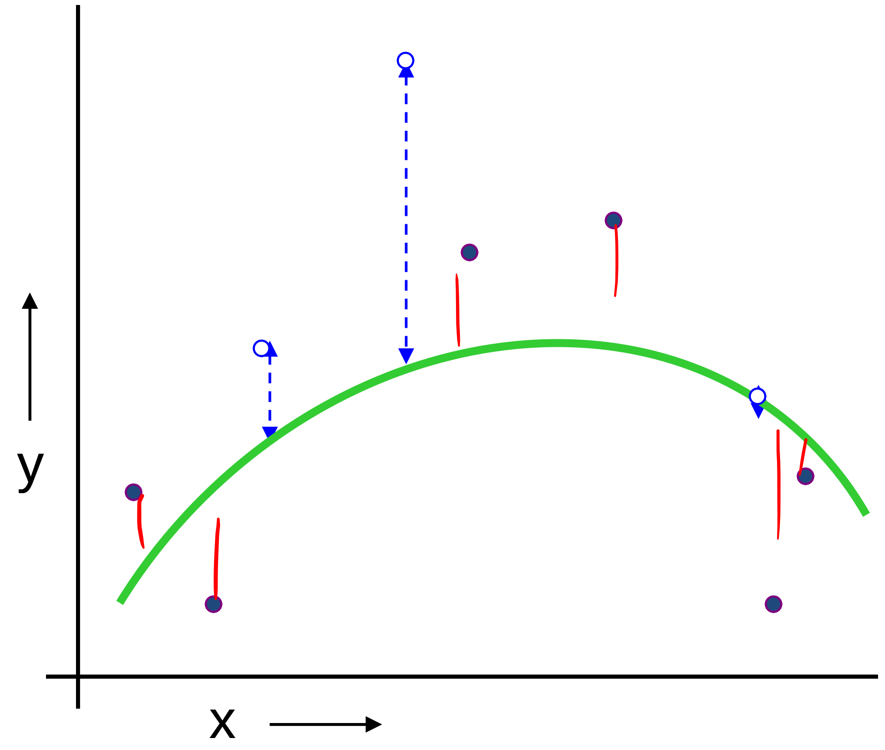
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

The test set method



(Quadratic regression example)

Mean Squared Error = 0.9

1. Randomly choose 30% of the data to be in a test set

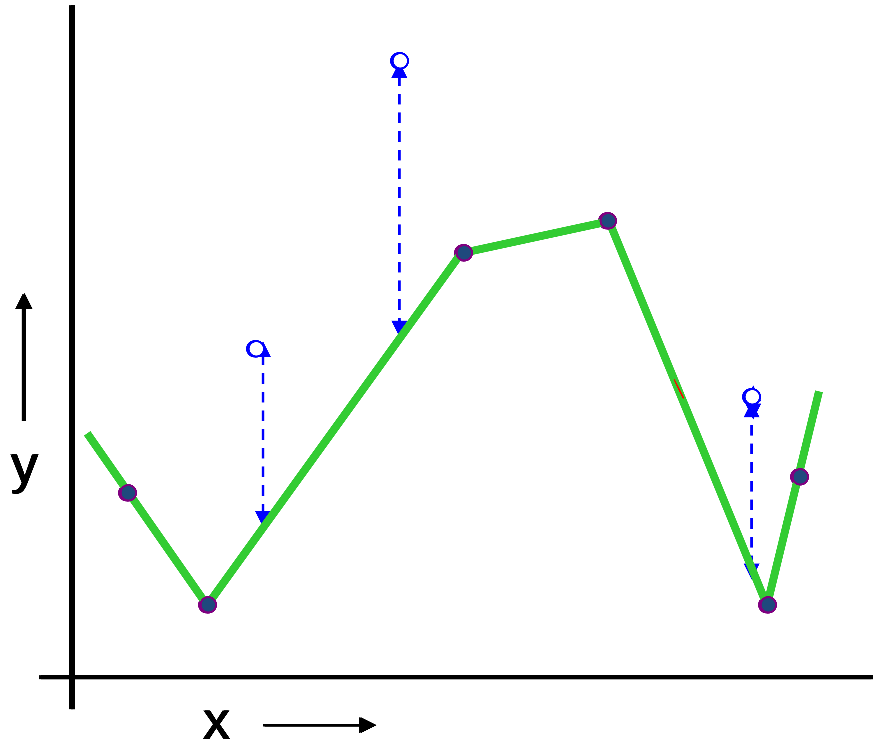
2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

The test set method

1. under fit
2. Good fit
3. Over fit



1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

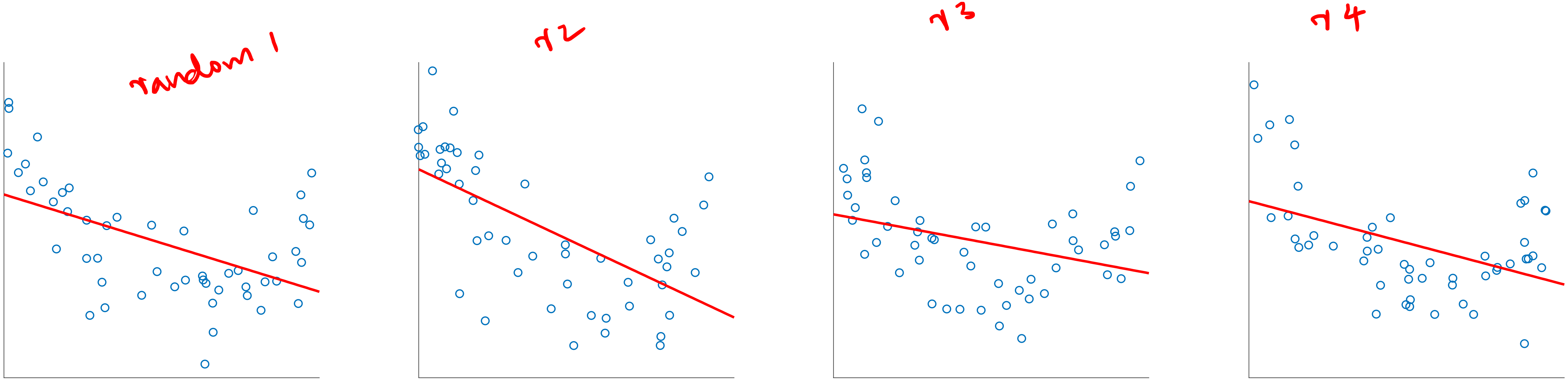
4. Estimate your future performance with the test set

(Join the dots example)

Mean Squared Error = 2.2 (test error)

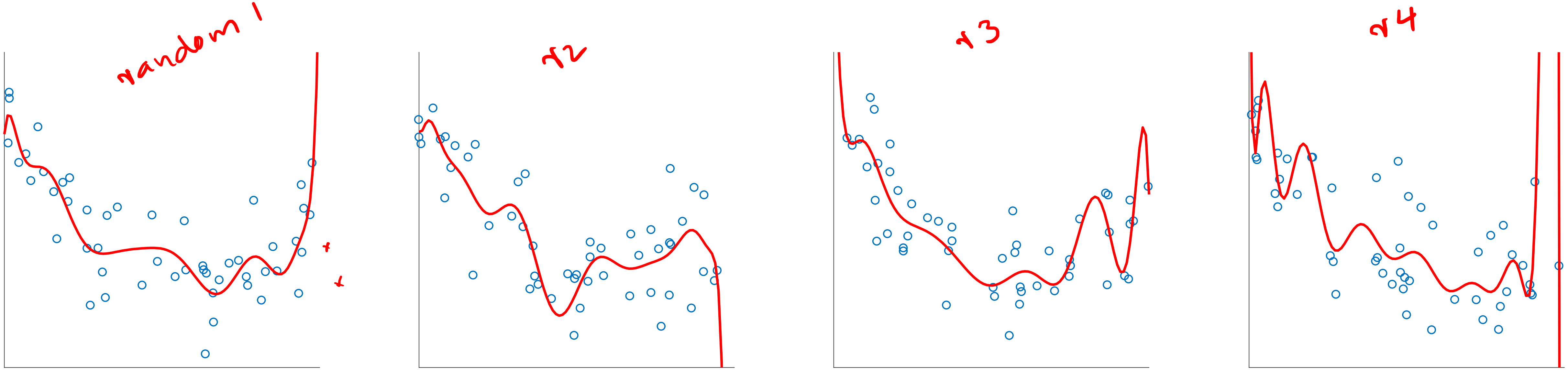
training error ≈ 0

Bias



Regardless of training sample, or size of training sample, model will produce consistent errors

Variance



Different samples of training data yield different model fits

Bias-Variance Trade Off

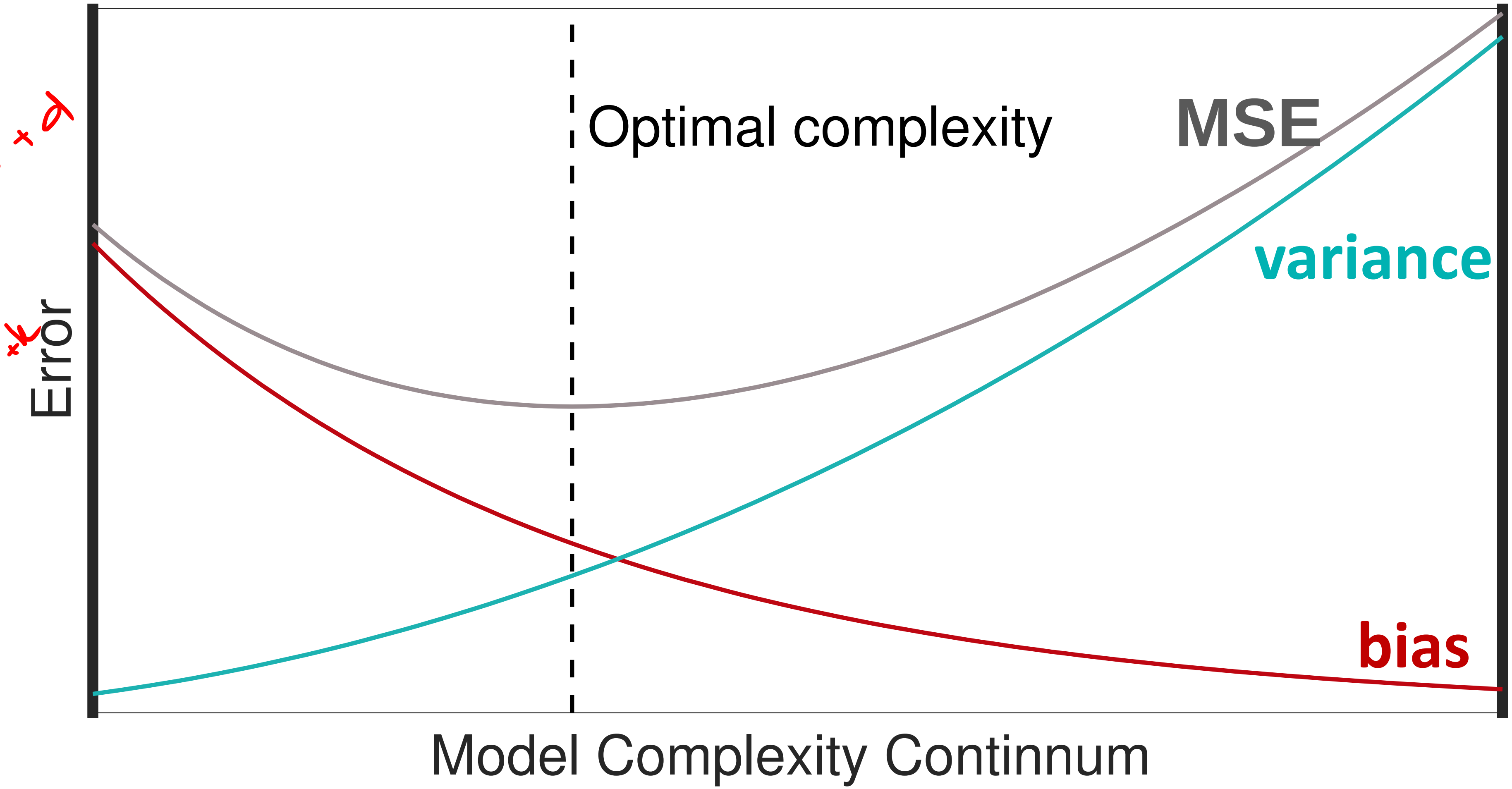
$y = ax + b$

$y = ax^2 + bx + c$

$y = ax^3 + bx^2 + cx + d$

$\vdots$

$y = ax^{10} + bx^9 + \cdots + k$



The test set method

training set → model evaluation

dataset

test set

Good news:

- Very very simple
- Can then simply choose the method with the best test-set score

Bad news:

- Wastes data: we get an estimate of the best method to apply to 30% less data (70%, 30%)
- If we don't have much data, our test-set might just be lucky or unlucky

Which kind of Cross Validation?

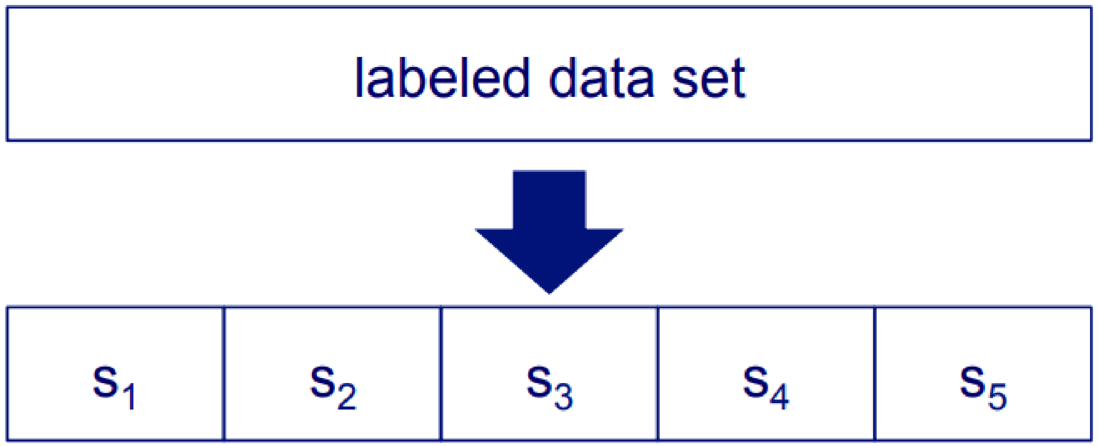
CV LOOCV 100 99 - 1

dataset set 1 set 2 set 3

	Downside	Upside
Test-set	Variance: unreliable estimate of future performance 30%	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data
10-fold	Wastes 10% of the data. 10 times more expensive than test set	Only wastes 10%. Only 10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

Cross validation

partition data
into n subsamples



dataset
100%

iteratively leave one subsample out for the test set, train on the rest

iteration	train on	test on
1	s_2 s_3 s_4 s_5	s_1
2	s_1 s_3 s_4 s_5	s_2
3	s_1 s_2 s_4 s_5	s_3
4	s_1 s_2 s_3 s_5	s_4
5	s_1 s_2 s_3 s_4	s_5

Cross validation example

20 , 20,20,20,20

Suppose we have 100 instances, and we want to estimate accuracy with cross validation

iteration	train on	test on	correct
1	s_2 s_3 s_4 s_5	s_1	11 / 20
2	s_1 s_3 s_4 s_5	s_2	17 / 20
3	s_1 s_2 s_4 s_5	s_3	16 / 20
4	s_1 s_2 s_3 s_5	s_4	13 / 20
5	s_1 s_2 s_3 s_4	s_5	16 / 20

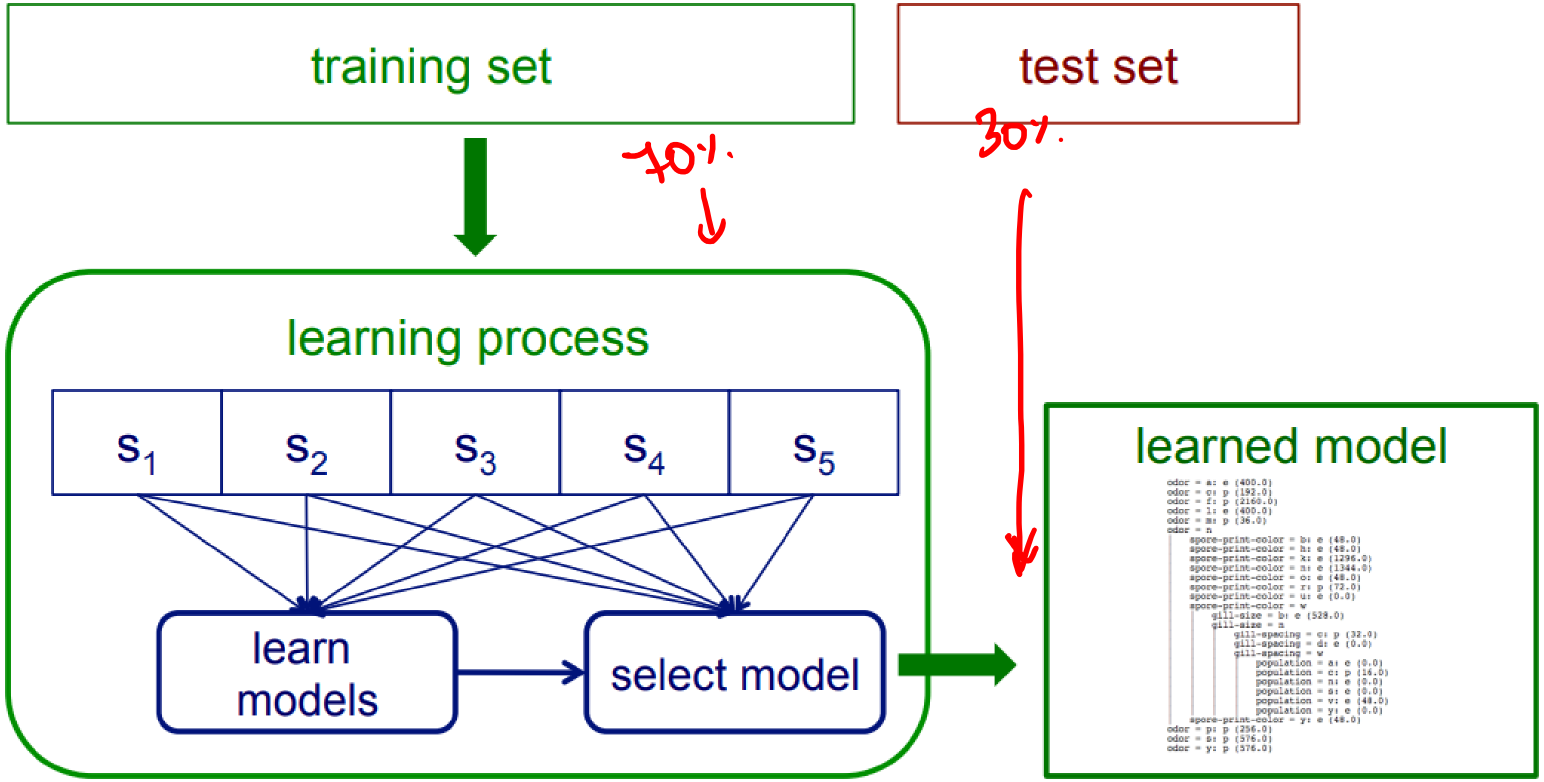
accuracy = 73/100 = 73%

Cross validation

- 10-fold cross validation is common, but smaller values of
- n are often used when learning takes a lot of time
- in leave-one-out cross validation, n = # instances
- in stratified cross validation, stratified sampling is used when partitioning the data
- CV makes efficient use of the available data for testing
- note that whenever we use multiple training sets, as in CV and random resampling, we are evaluating a learning method as opposed to an individual learned model

Internal cross validation

- Instead of a single validation set, we can use cross-validation within a training set to select a model (e.g. to choose the best level of decision-tree pruning)



Example: using internal cross validation to select k in k-NN

given a training set

1. partition training set into n folds, $s_1 \dots s_n$
2. for each value of k considered

   for i = 1 to n

   learn k-NN model using all folds but s_i

   evaluate accuracy on s_i

3. select k that resulted in best accuracy for $s_1 \dots s_n$
4. learn model using entire training set and selected k

- the steps inside the box are run independently for each training set (i.e. if we're using 10-fold CV to measure the overall accuracy of our k-NN approach, then the box would be executed 10 times)

Confusion matrices

How can we understand what types of mistakes a learned model makes?

1000

activity recognition from video

actual class \ predicted class	bend	jack	jump	pjump	run	side	skip	walk	wave1	wave2
bend	100	0	0	0	0	0	0	0	0	0
jack	0	100	0	0	0	0	0	0	0	0
jump	0	0	89	0	0	0	11	0	0	0
pjump	0	0	0	100	0	0	0	0	0	0
run	0	0	0	0	89	0	11	0	0	0
side	0	0	0	0	0	100	0	0	0	0
skip	0	0	0	0	0	0	100	0	0	0
walk	0	0	0	0	0	0	0	100	0	0
wave1	0	0	0	0	0	0	0	0	67	33
wave2	0	0	0	0	0	0	0	0	0	100

actual class

predicted class

Confusion matrix for 2-class problems

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

Is accuracy an adequate measure of predictive performance?

- accuracy may not be useful measure in cases where
  - there is a large class skew
    - Is 98% accuracy good if 97% of the instances are negative?
  - there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong
  - Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

50% + 50% −

97% + 3% −

55 − 45% + −

97 → table

3 → chair

100 → table

97%

$\frac{97}{100}$ = always

Other accuracy metrics

		actual class	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

Plot b/w TPR vs FPR

ROC

$$\text{true positive rate (recall)} = \frac{\text{TP}}{\text{actual pos}} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{false positive rate} = \frac{\text{FP}}{\text{actual neg}} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

Curse of Dimensionality

x Y 1000 1x1 = 2D

- The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience.
- The expression was coined by Richard E. Bellman when considering problems in dynamic optimization.
- There are multiple phenomena referred to by this name in domains such as numerical analysis, sampling, combinatorics, machine learning, data mining, and databases. The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse.

Curse of Dimensionality

Solution for unbalanced and high dimensionality data

Unbalanced data

Stratification

Rows Reduction

Bad / Bood group Balancing

Dimensionality Reduction

Feature Selection

Column Reduction

Python Packages needed

- pandas
  - Data Analytics
- numpy
  - Numerical Computing
- matplotlib.pyplot
  - Plotting graphs
- sklearn
  - Classification and Regression Classes

Implementation Using sklearn

Let's go to Jupyter Notebook!