UVA CS 4774: Machine Learning

Lecture 6: Model Selection

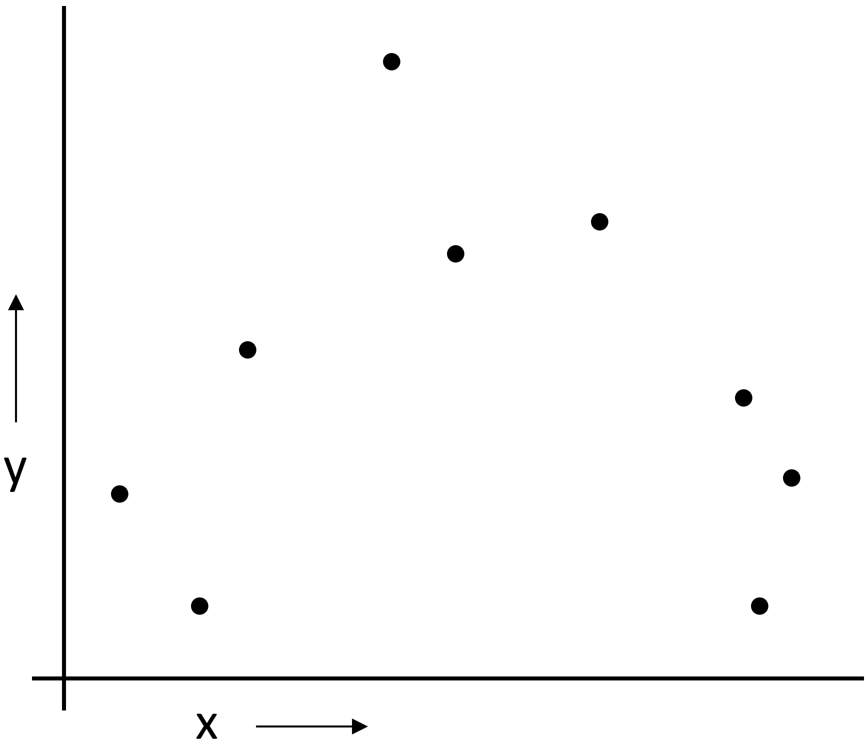
Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Main issues: Model Selection

- How to select the right model?
  - E.g. what polynomial degree d for polynomial regression
  - E.g., where to put the centers for the RBF kernels? How wide?
  - E.g. which basis type? Polynomial or RBF?

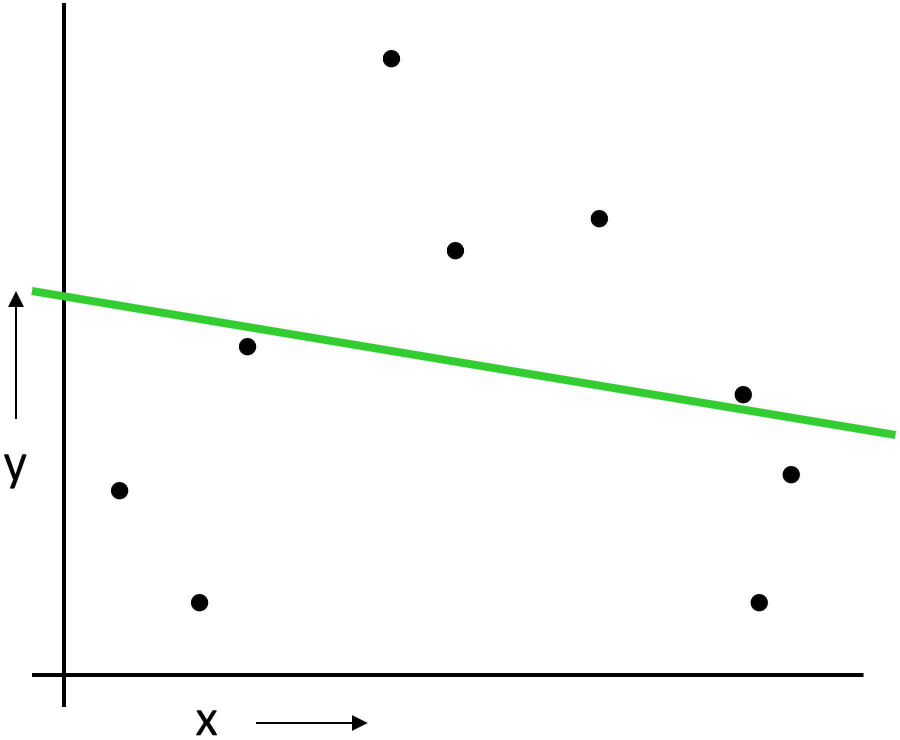
To Avoid: Overfitting or Underfitting



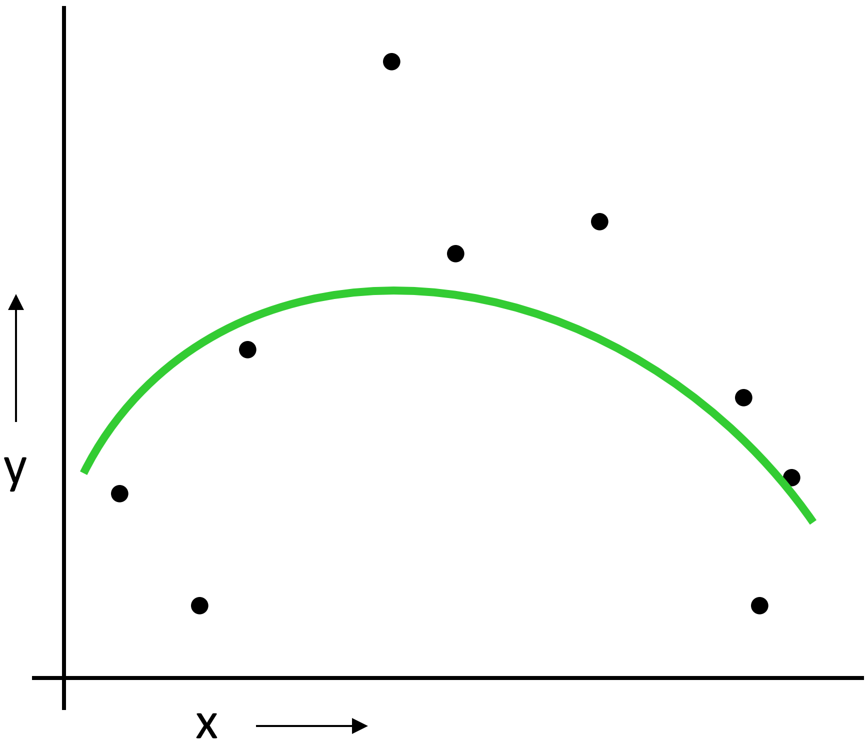
Can we learn a regression f from the data?

Let’s consider three methods…

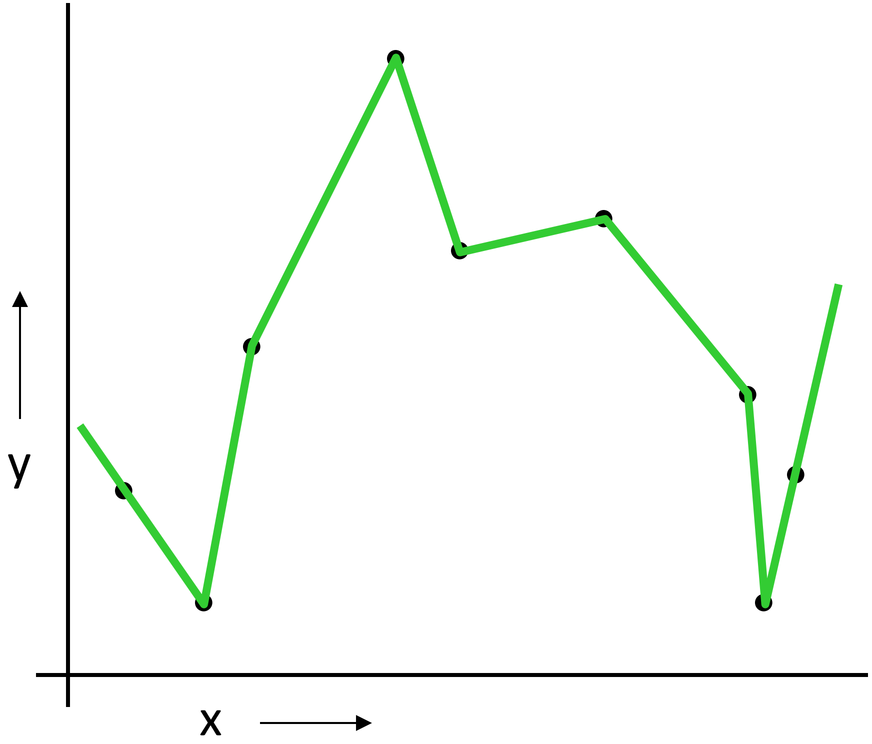
Linear Regression



Quadratic Regression

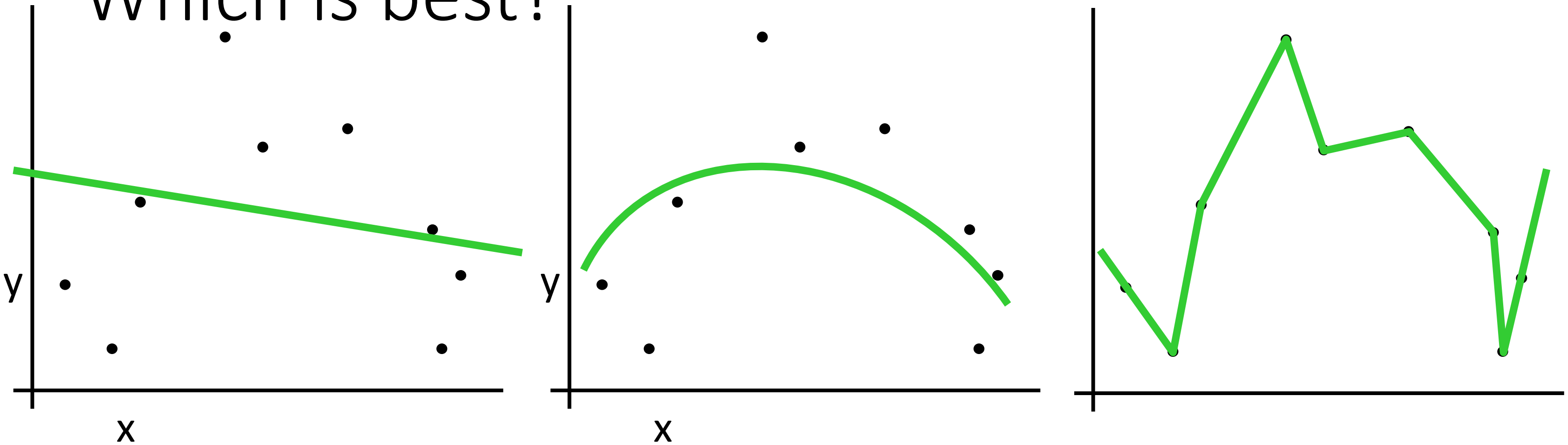


Join-the-dots



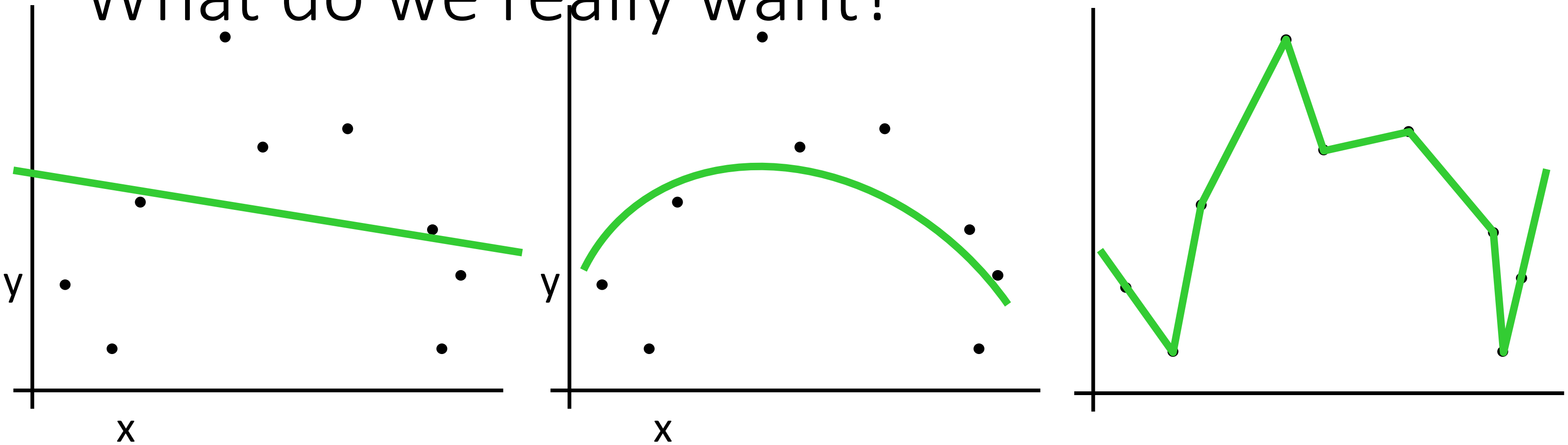
Also known as piecewise linear nonparametric regression if that makes you feel better

Which is best?



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

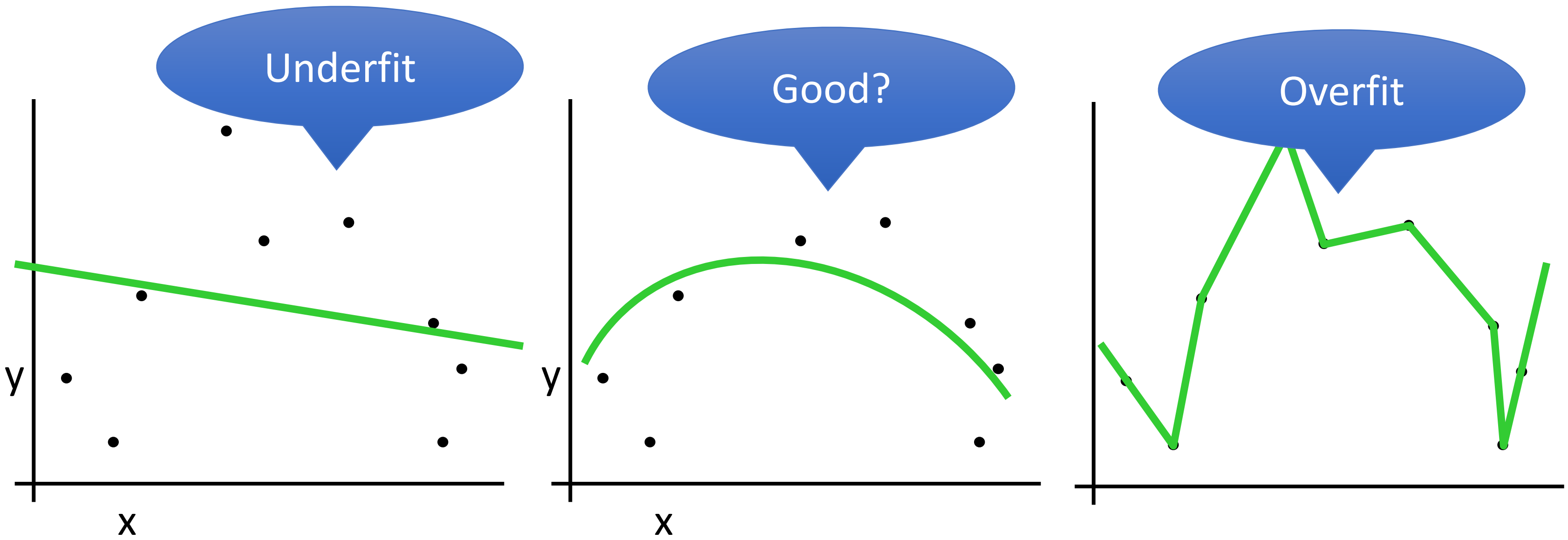
What do we really want?



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?"

What Model Type to Select?

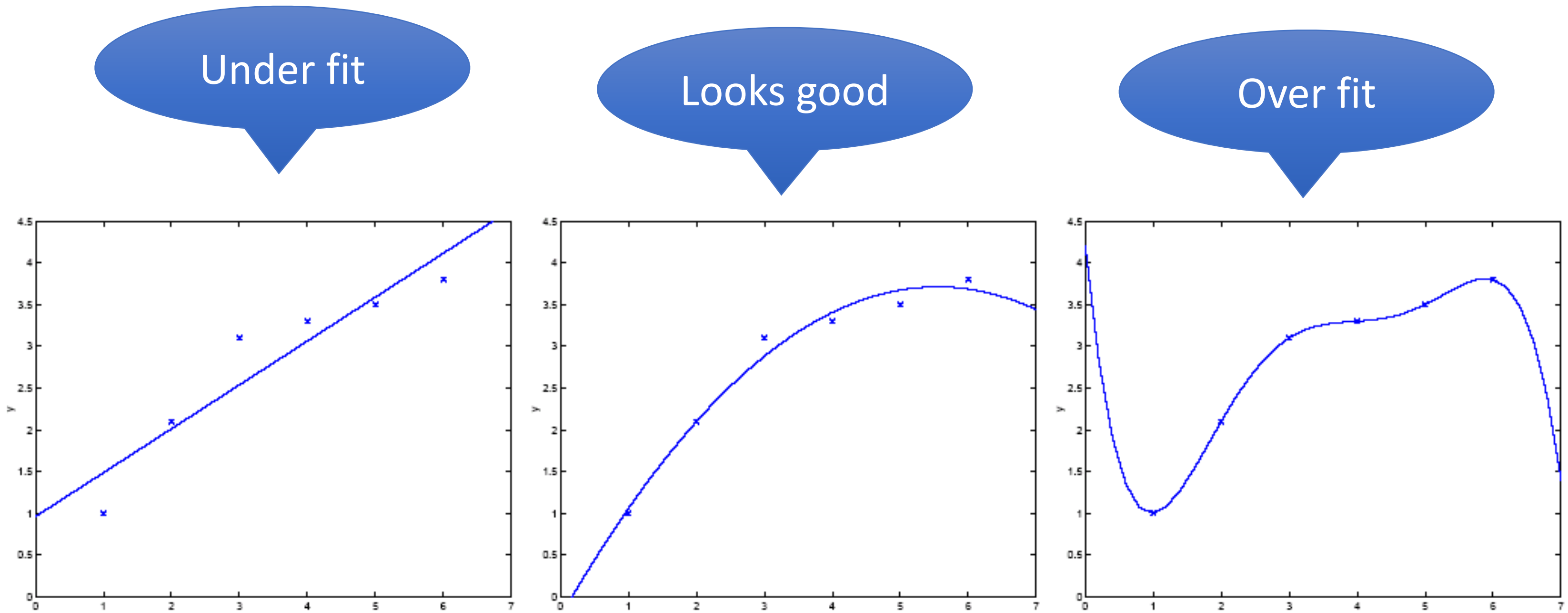


Why not choose the method with the best fit to the data?

K-fold Cross Validation / Train-Test /

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

What Model Order to Select?



$$y = \theta_0 + \theta_1 x \qquad y = \theta_0 + \theta_1 x + \theta_2 x^2 \qquad y = \sum_{j=0}^{5} \theta_j x^j$$

Generalisation: learn function / hypothesis from past data in order to "explain", "predict", "model" or "control" new data examples

K-fold Cross Validation / Train-Test /

Choice-I: Train-Test (Leave m out)

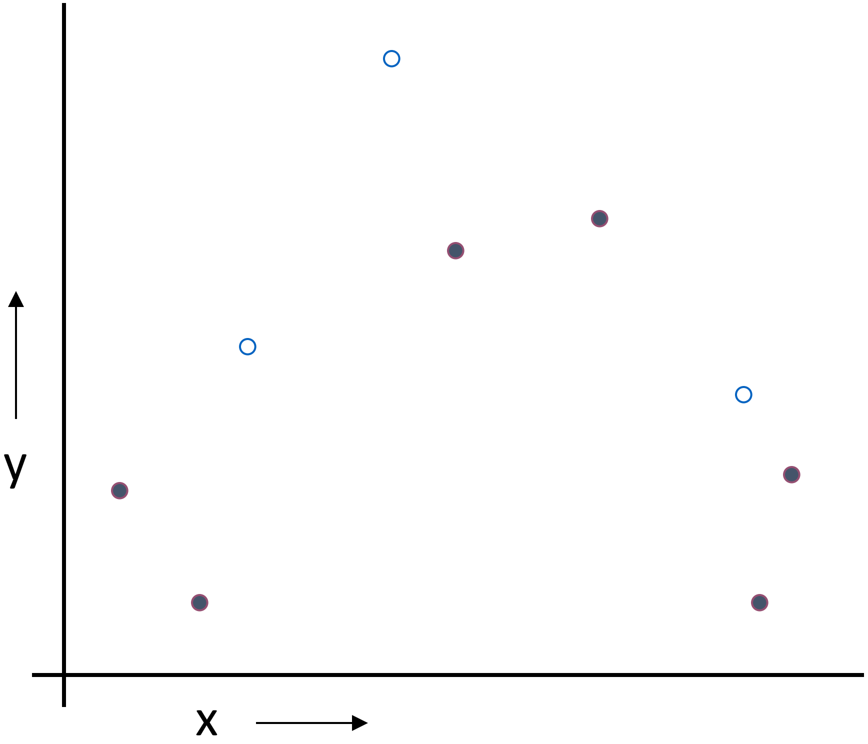
training dataset

$$\mathbf{X}_{train} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}_{train} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

test dataset

$$\mathbf{X}_{test} = \begin{bmatrix} -- & \mathbf{x}_{n+1}^T & -- \\ -- & \mathbf{x}_{n+2}^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_{n+m}^T & -- \end{bmatrix} \qquad \vec{y}_{test} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}$$

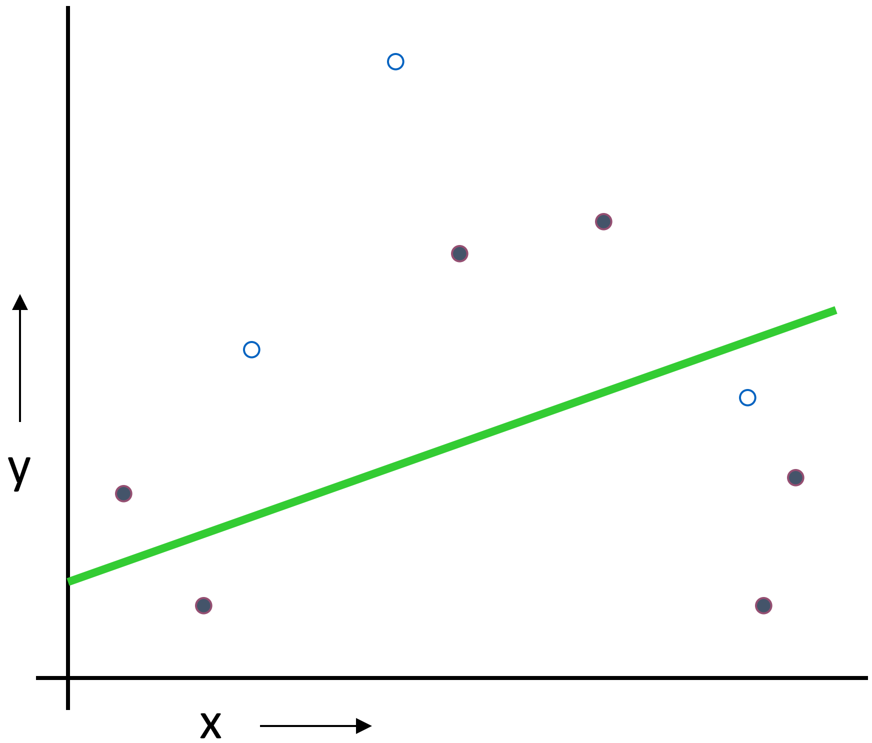
The test set method



1. Randomly choose some percentage like 30% of the labeled data to be in a test set
2. The remainder is a training set

Credit: Prof. Andrew Moore

The test set method

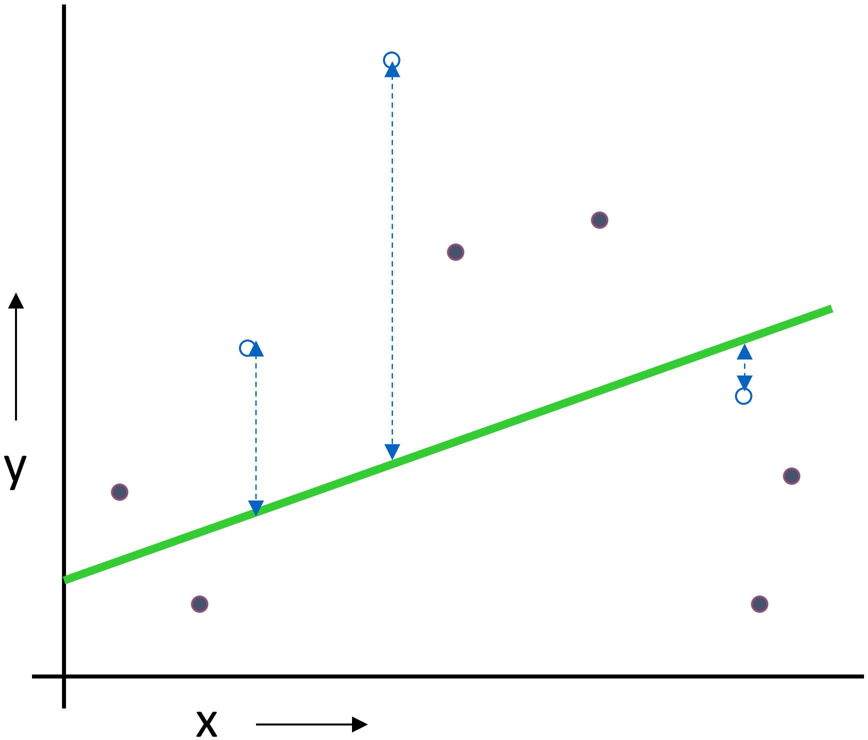


1. Randomly choose some percentage like 30% of the labeled data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set

(Linear regression example)

Credit: Prof. Andrew Moore

The test set method



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

(Linear regression example)
Mean Squared Error = 2.4

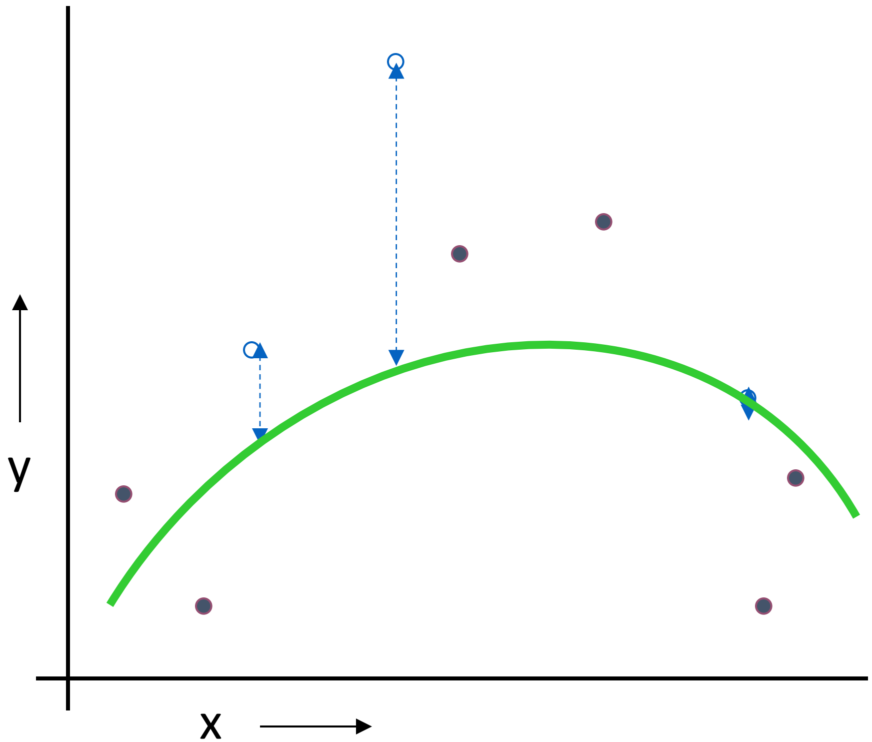
Credit: Prof. Andrew Moore

e.g. for Regression Models

- Testing Mean Squared Error - MSE to report:

$$J_{test} = \frac{1}{m}\sum_{i=n+1}^{n+m}(\mathbf{x}_i^T\theta^* - y_i)^2 = \frac{1}{m}\sum_{i=n+1}^{n+m}\varepsilon_i^2$$

The test set method

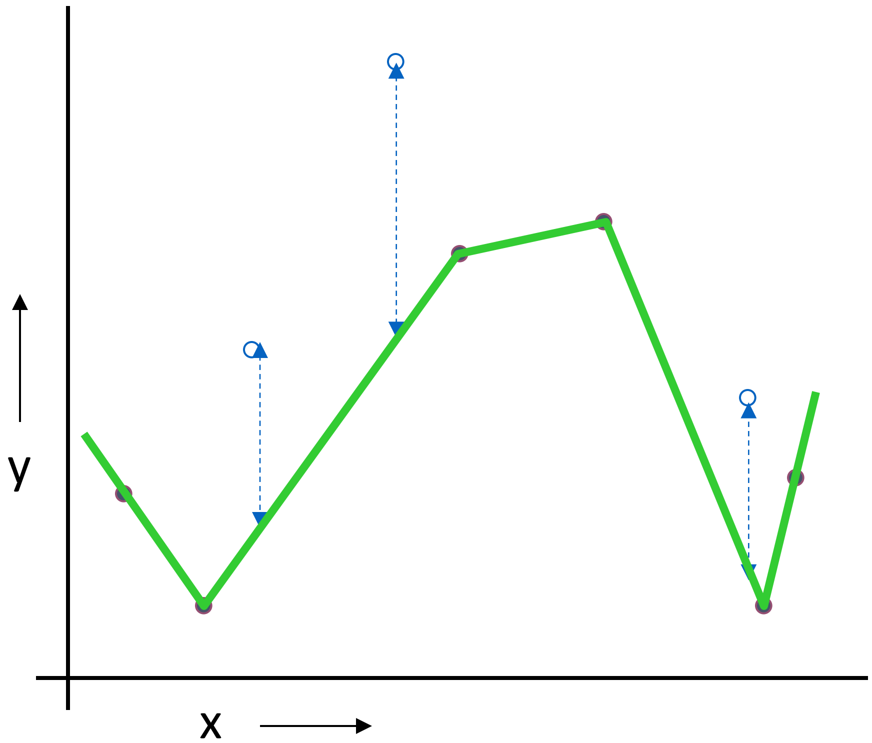


1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

(Quadratic regression example)
Mean Squared Error = 0.9

Credit: Prof. Andrew Moore

The test set method



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

Credit: Prof. Andrew Moore

The test set method

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
- If we don’t have much data, our test-set might just be lucky or unlucky

We say the “test-set estimator of performance has high variance”

Credit: Prof. Andrew Moore

Choice-II: k-Fold Cross Validation

- Problem of train-test: in many cases we don't have enough data to set aside a test set
- Solution: Each data point is used both as train and test
- Common types:
  - K-fold cross-validation (e.g. K=5, K=10)
  - Leave-one-out cross-validation (LOOCV, i.e., k=n)

e.g. By k=10 fold Cross Validation

- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from each test
- We normally use the mean of the scores

model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	train	train	train	train	train	train	train	train	train	test
2	train	train	train	train	train	train	train	train	test	train
3	train	train	train	train	train	train	train	test	train	train
4	train	train	train	train	train	train	test	train	train	train
5	train	train	train	train	train	test	train	train	train	train
6	train	train	train	train	test	train	train	train	train	train
7	train	train	train	test	train	train	train	train	train	train
8	train	train	test	train	train	train	train	train	train	train
9	train	test	train	train	train	train	train	train	train	train
10	test	train	train	train	train	train	train	train	train	train

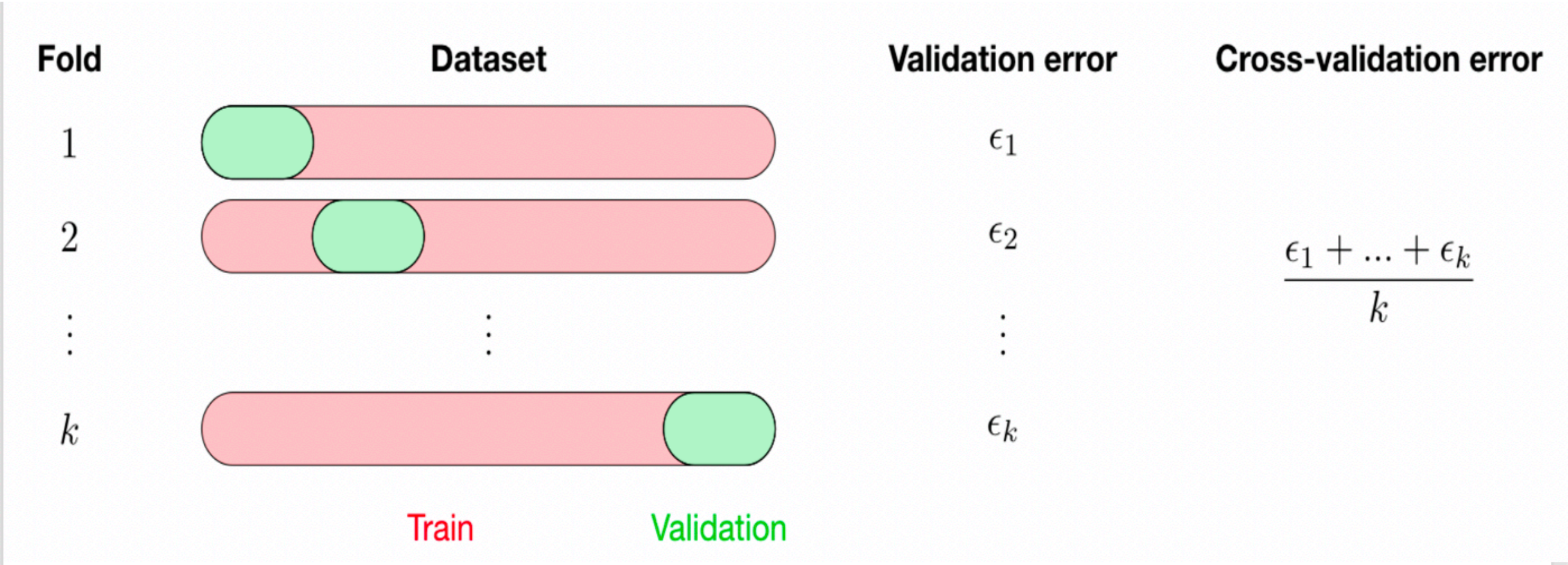
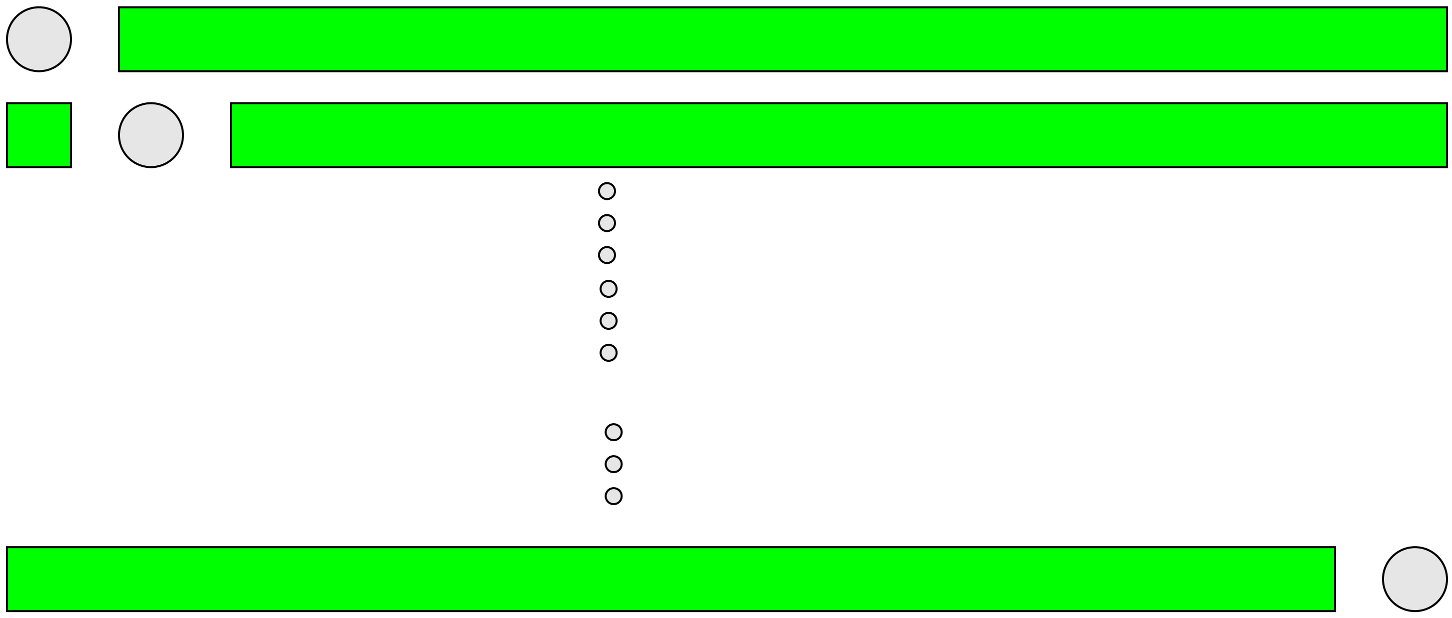


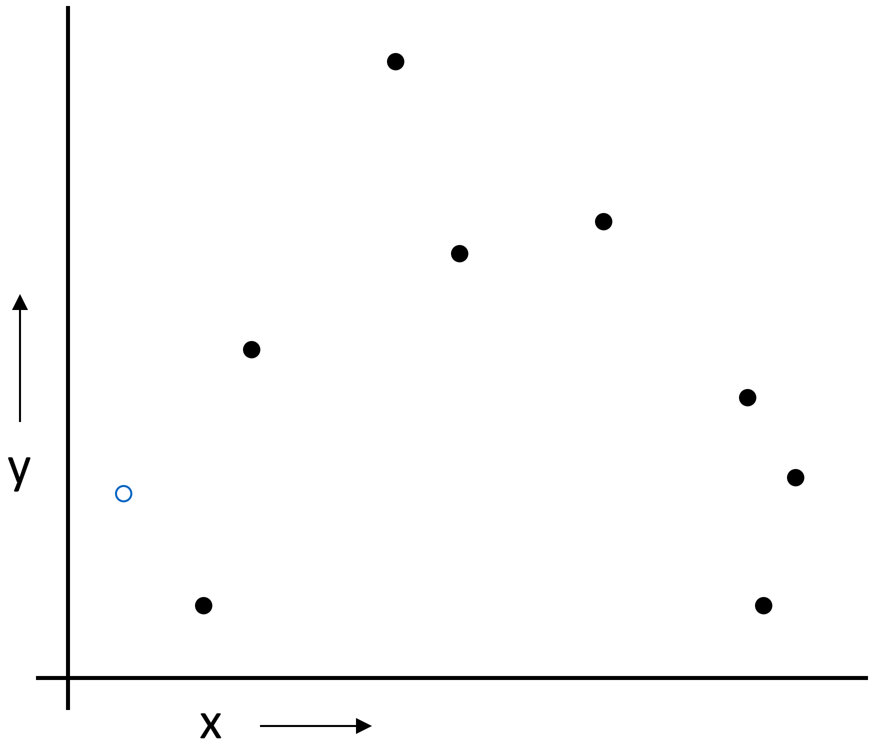
Image Credit: Stanford Machine Learning course

e.g. Leave-one-out / LOOCV (n-fold cross validation)

n is num. of data samples



LOOCV (Leave-one-out Cross Validation)

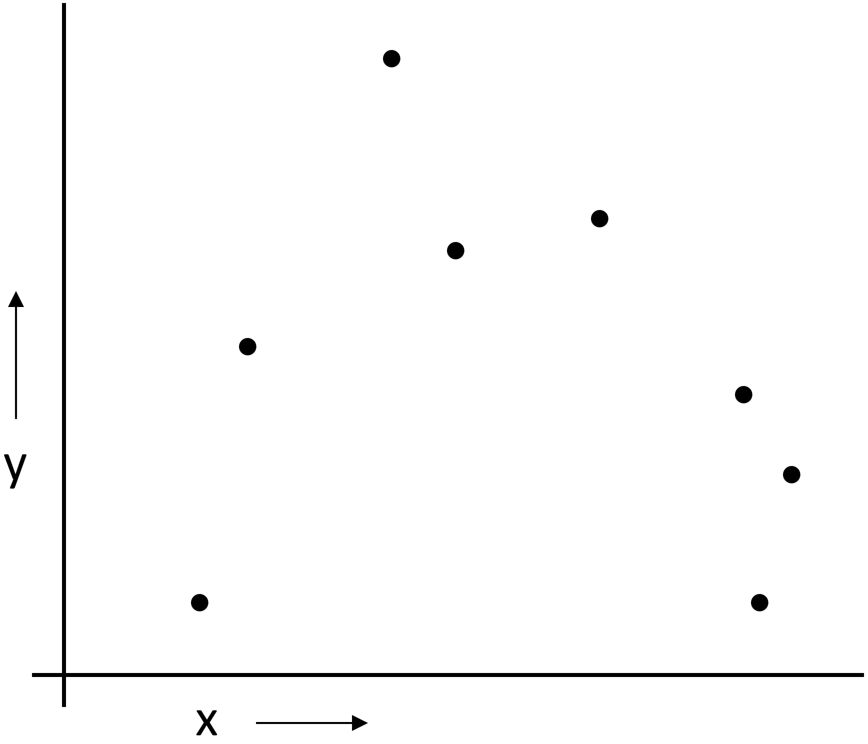


For k=1 to n

1. Let (x_k,y_k) be the k^{th} record

Credit: Prof. Andrew Moore

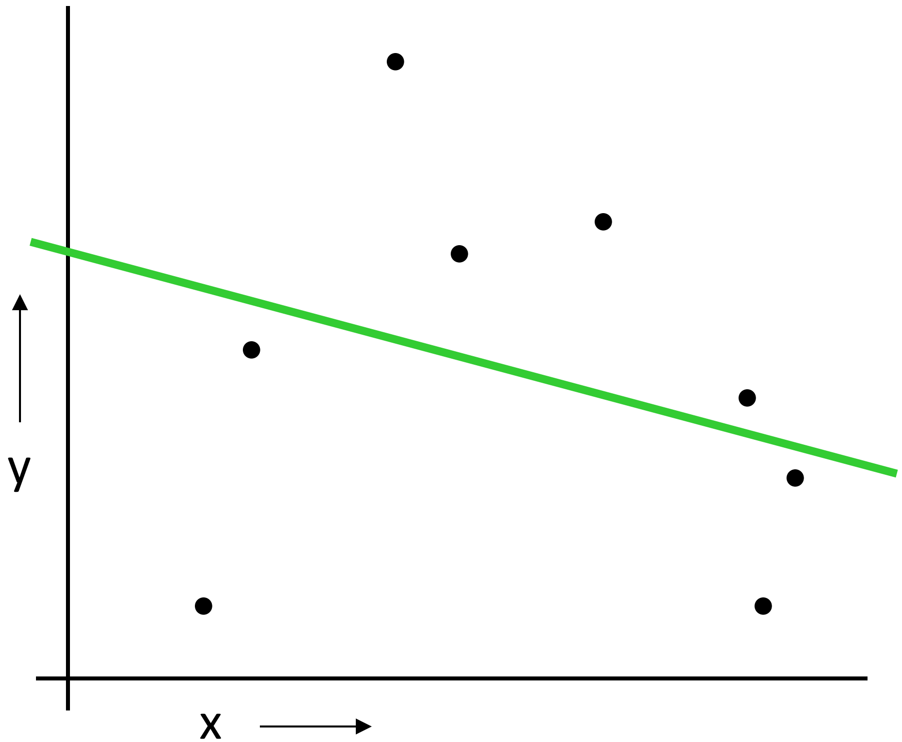
LOOCV (Leave-one-out Cross Validation)



For k=1 to n

1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset

LOOCV (Leave-one-out Cross Validation)

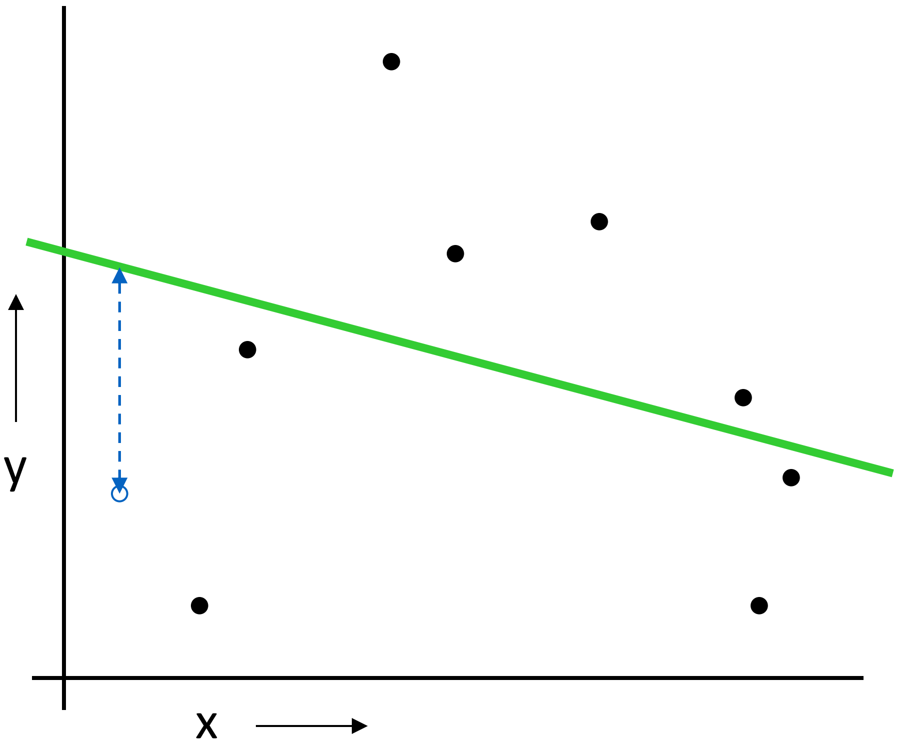


For k=1 to n

1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining n-1 datapoints

Credit: Prof. Andrew Moore

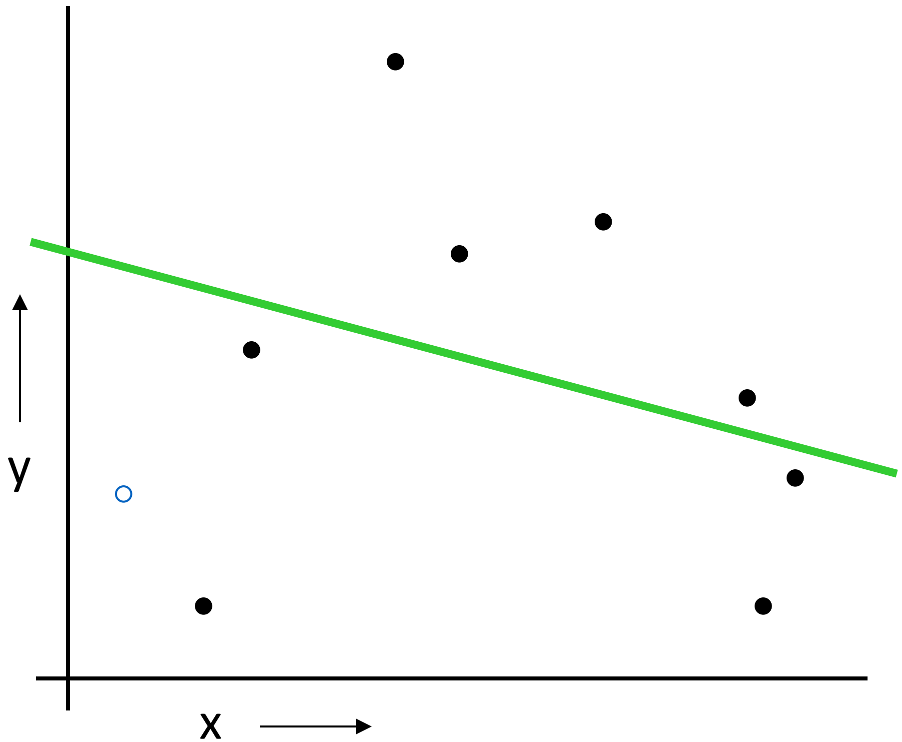
LOOCV (Leave-one-out Cross Validation)



For k=1 to n

1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k,y_k)

LOOCV (Leave-one-out Cross Validation)

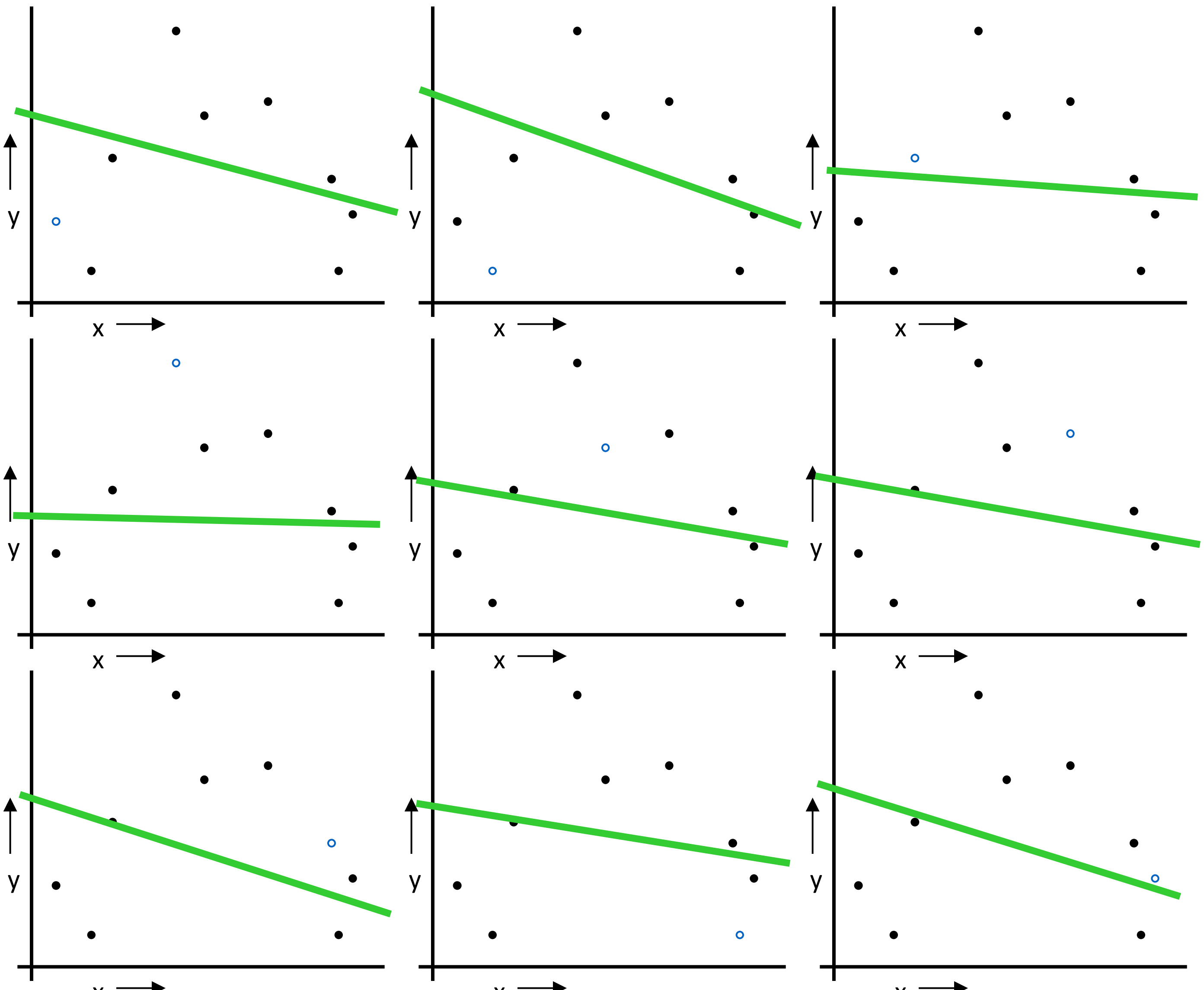


For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

LOOCV for Linear Regression



For k=1 to n

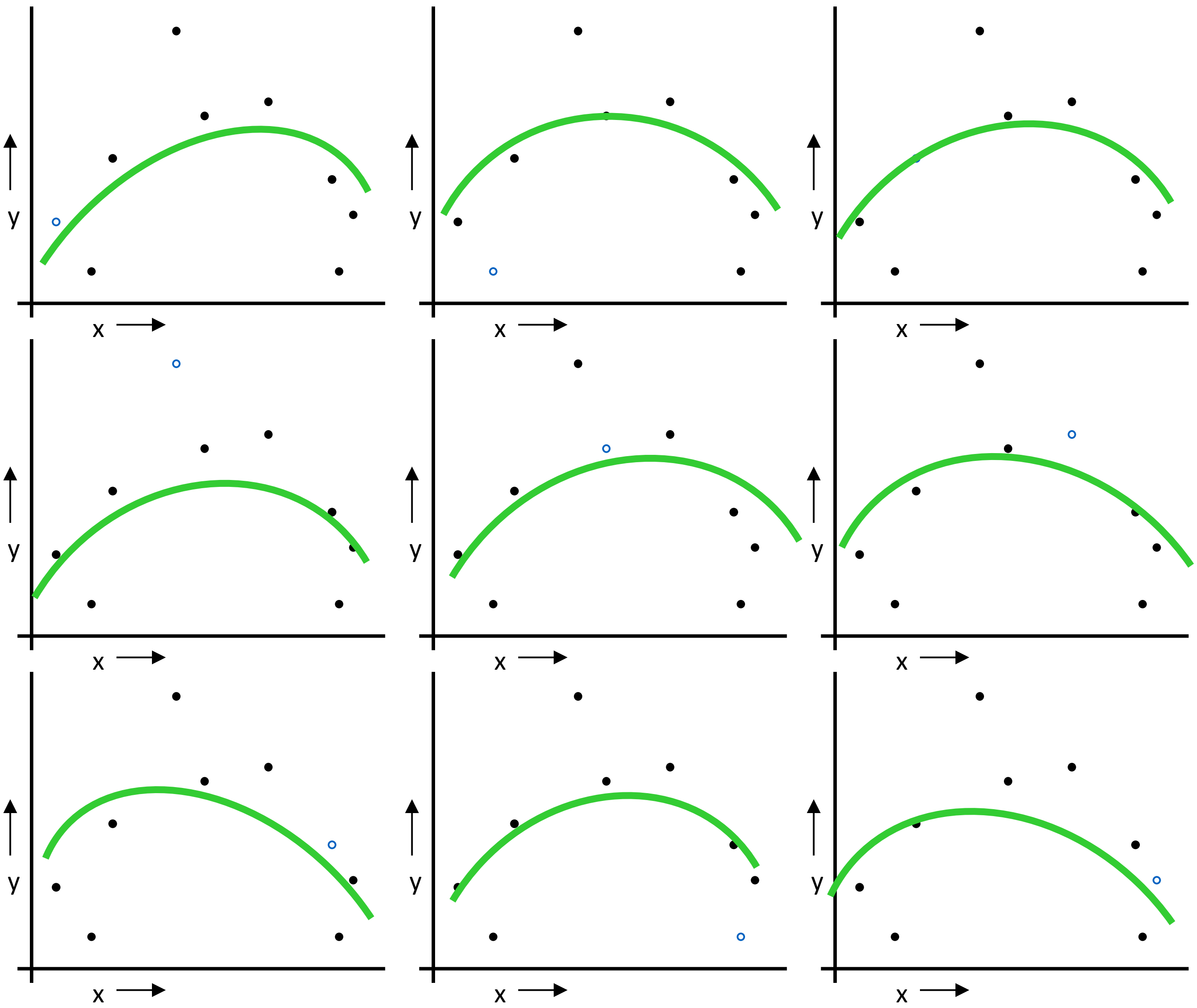
1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k,y_k)

When you've done all points, report the mean error.

$MSE_{LOOCV} = 2.12$

Credit: Prof. Andrew Moore

LOOCV for Quadratic Regression



For k=1 to n

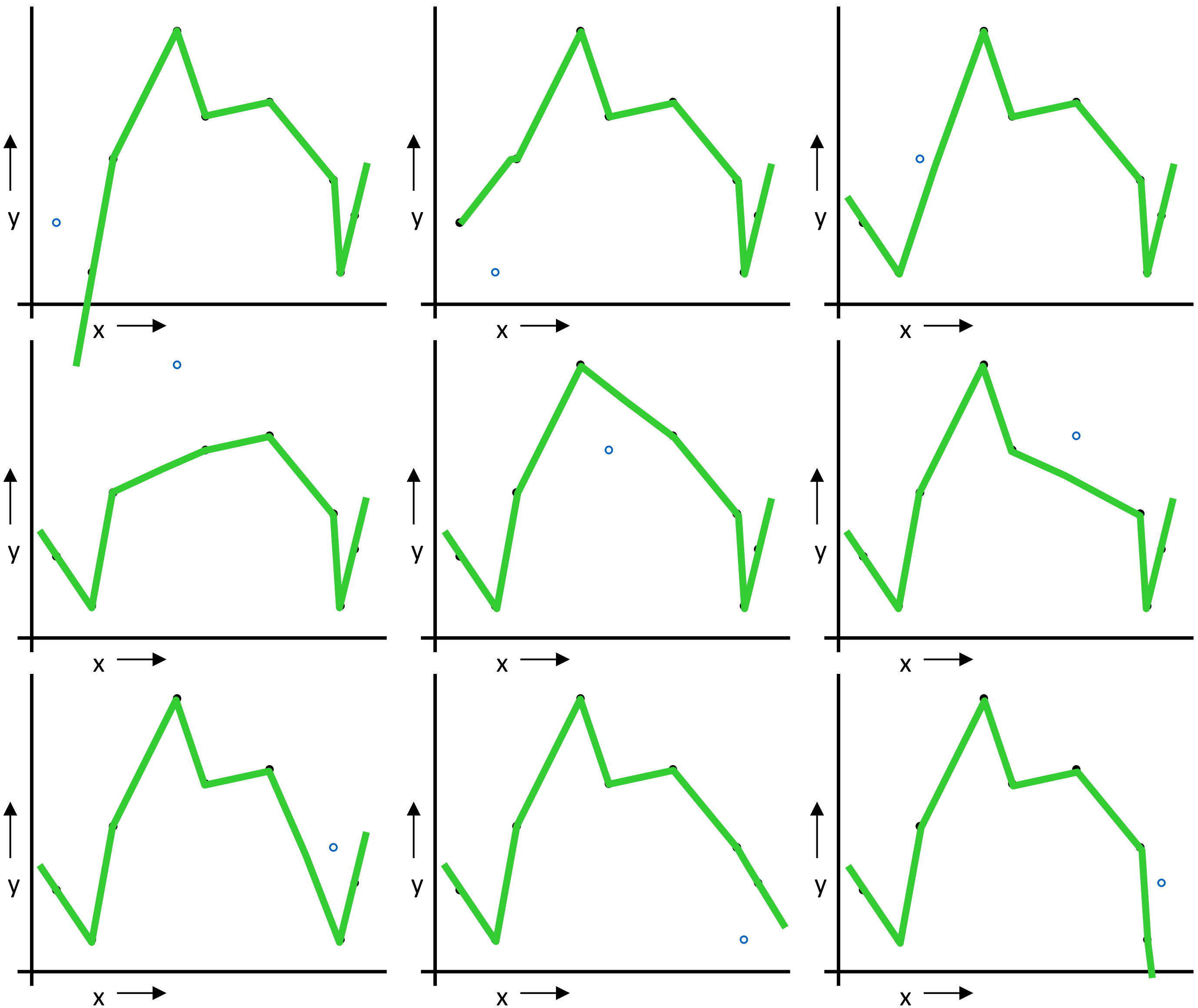
1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

MSE_{LOOCV}=0.962

Credit: Prof. Andrew Moore

LOOCV for Join The Dots



For k=1 to n

1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k,y_k)

When you've done all points, report the mean error.

$MSE_{LOOCV}=3.33$

Credit: Prof. Andrew Moore

Which kind of Cross Validation?

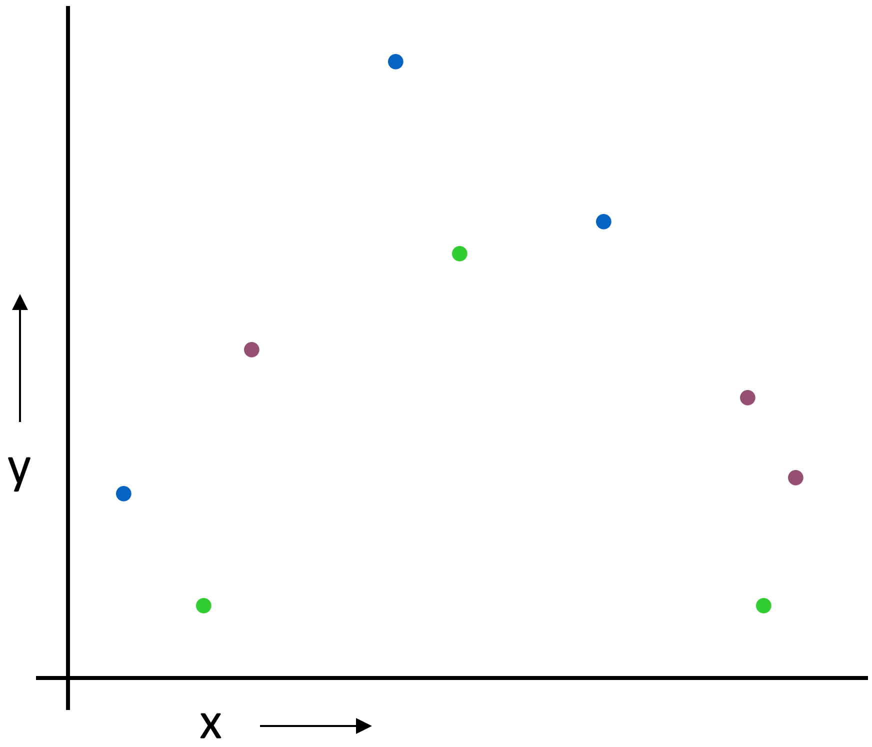
	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data

..can we get the best of both worlds?

Credit: Prof. Andrew Moore

k-fold Cross Validation

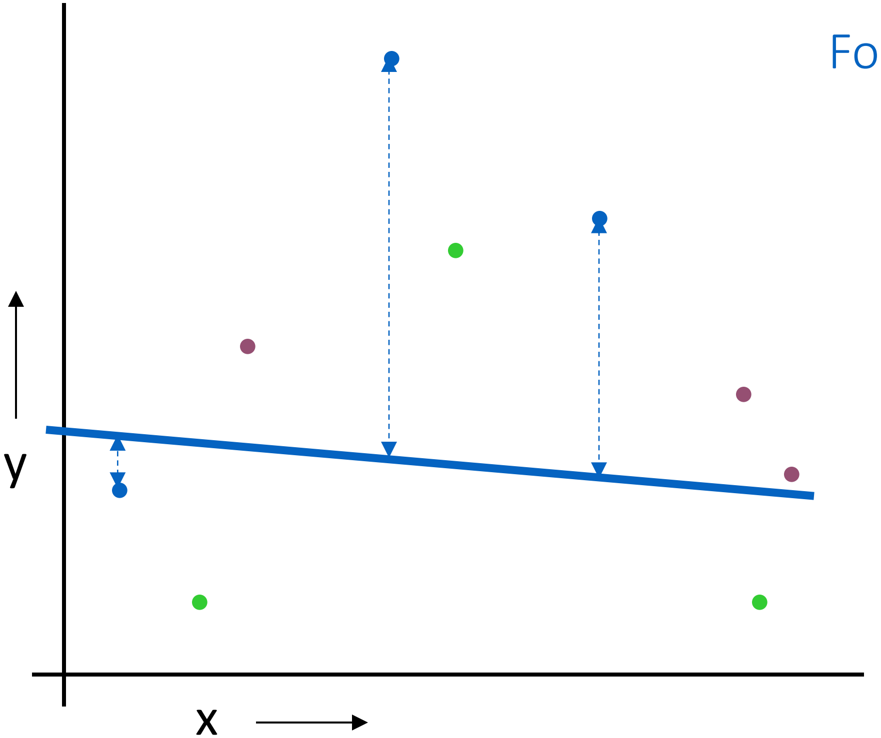
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)



Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

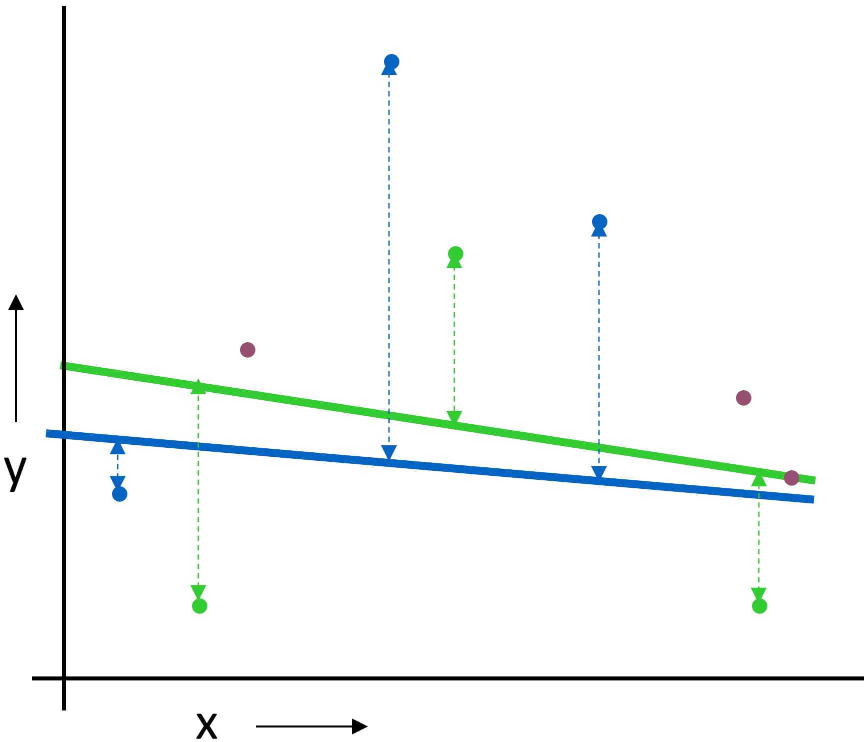


For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

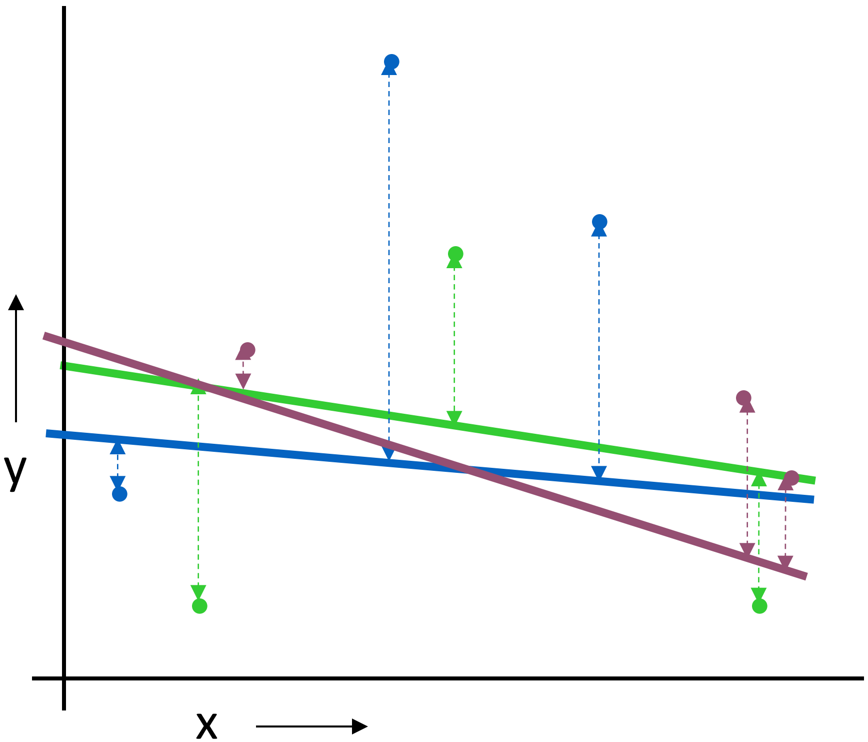


For the blue partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

Credit: Prof. Andrew Moore

k-fold Cross Validation



Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

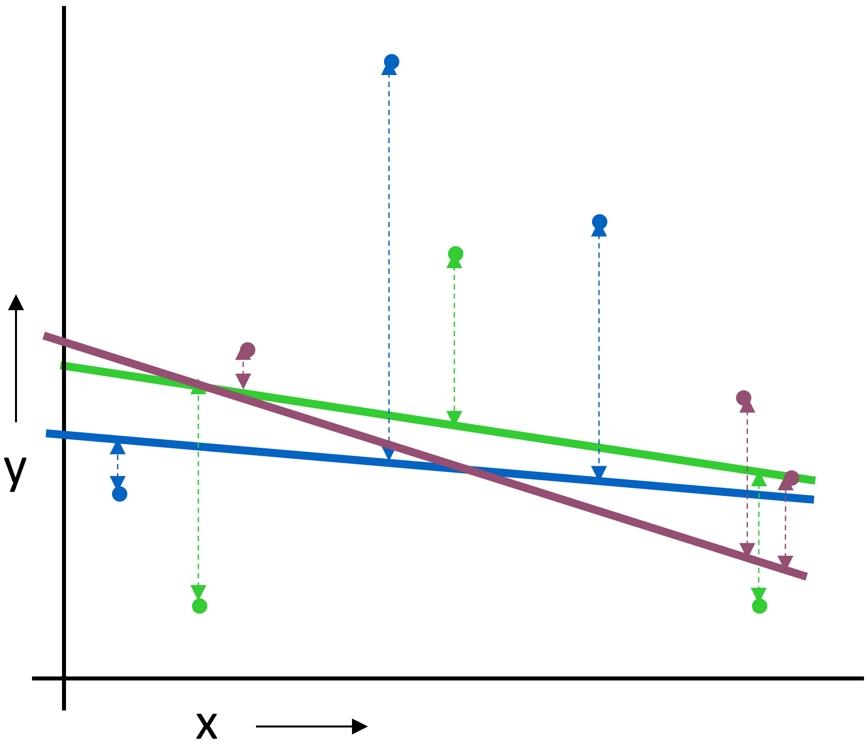
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the purple partition: Train on all the points not in the purple partition. Find the test-set sum of errors on the purple points.

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)



Linear Regression $MSE_{3FOLD}=2.05$

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

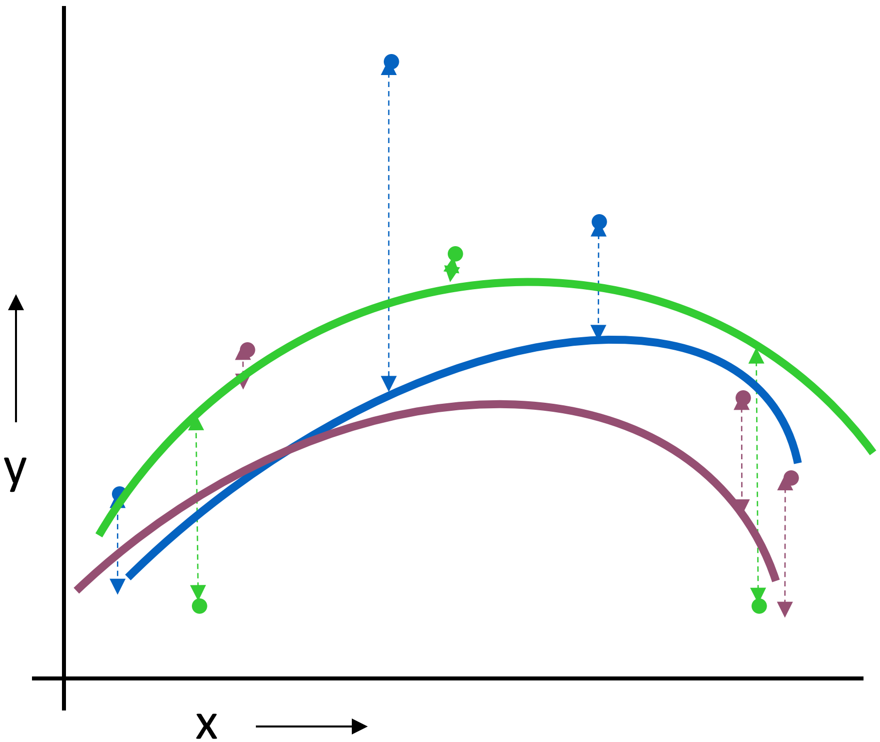
For the purple partition: Train on all the points not in the purple partition. Find the test-set sum of errors on the purple points.

Then report the mean error

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)



Quadratic Regression $MSE_{3FOLD}=1.11$

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

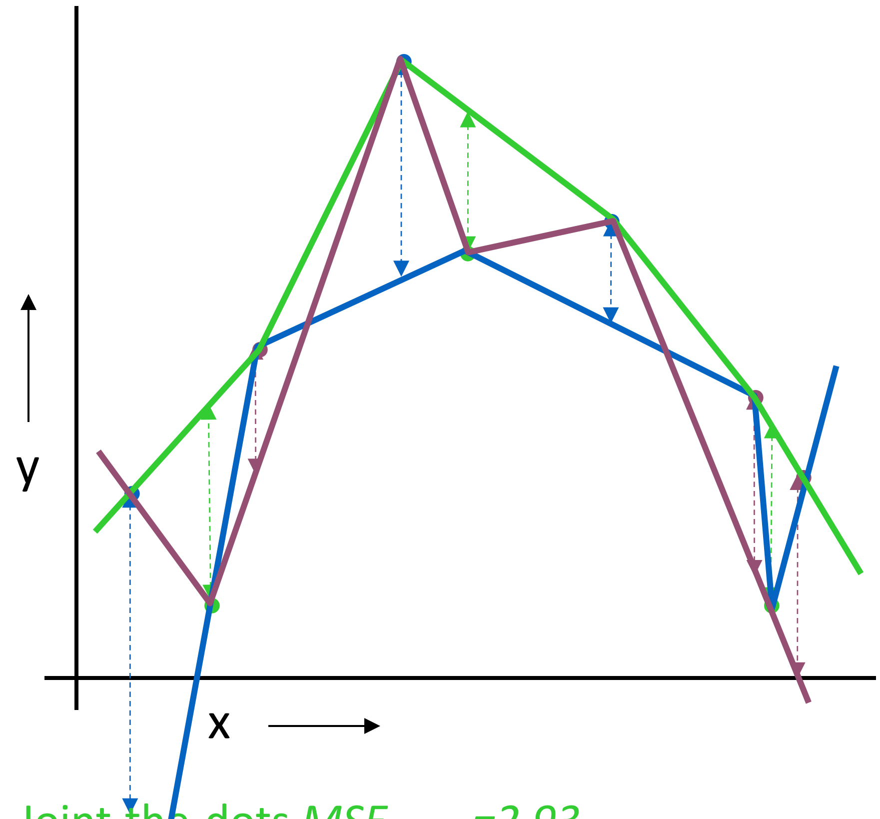
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the purple partition: Train on all the points not in the purple partition. Find the test-set sum of errors on the purple points.

Then report the mean error

Credit: Prof. Andrew Moore

k-fold Cross Validation



Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

Credit: Prof. Andrew Moore

Which kind of Cross Validation?

	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	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 n times.
3-fold	Wastier than 10-fold. More Expensive than test set style	better than test-set
n-fold	Identical to Leave-one-out	

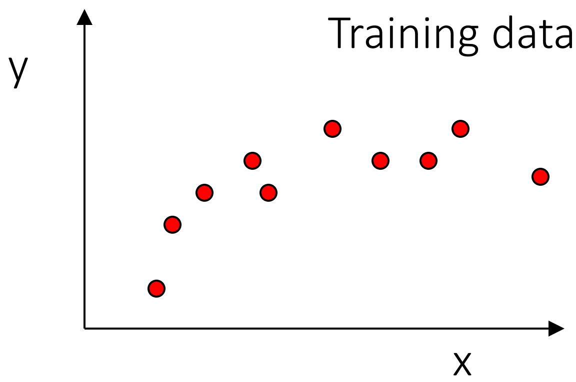
CV-based Model Selection

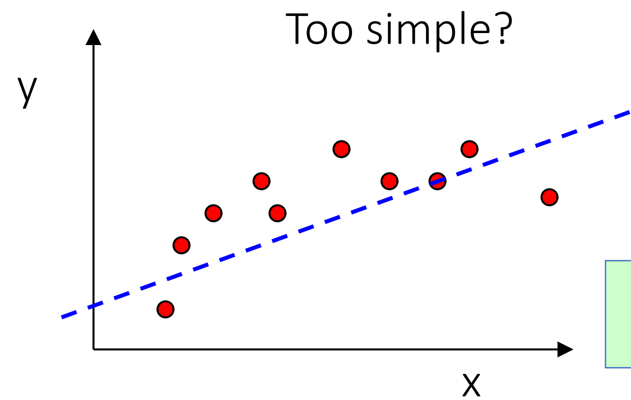
- We're trying to decide which algorithm/model to use.
- We train/learn/fit each model and make a table...

i	f_i	TRAINERR	k-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			?
4	f_4			
5	f_5			
6	f_6			

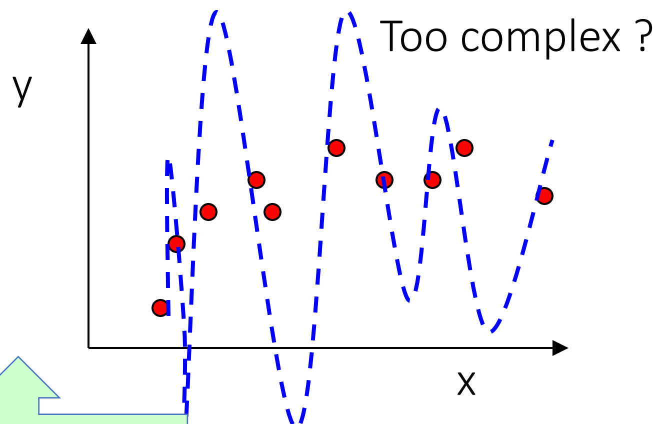
Credit: Prof. Andrew Moore

Later: Complexity versus Goodness of Fit

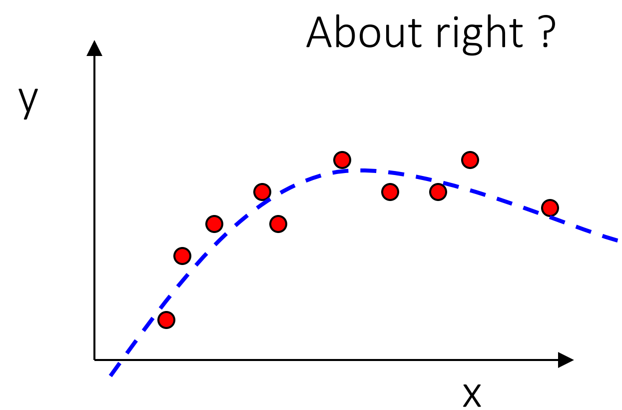




Low Variance / High Bias



Low Bias / High Variance



What ultimately matters: GENERALIZATION

References

- Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides

❑ Prof. Nando de Freitas’s tutorial slide

❑ Prof. Andrew Moore’s slides @ CMU