

Cross-validation for detecting and preventing overfitting

Andrew W. Moore

Professor

School of Computer Science

Carnegie Mellon University

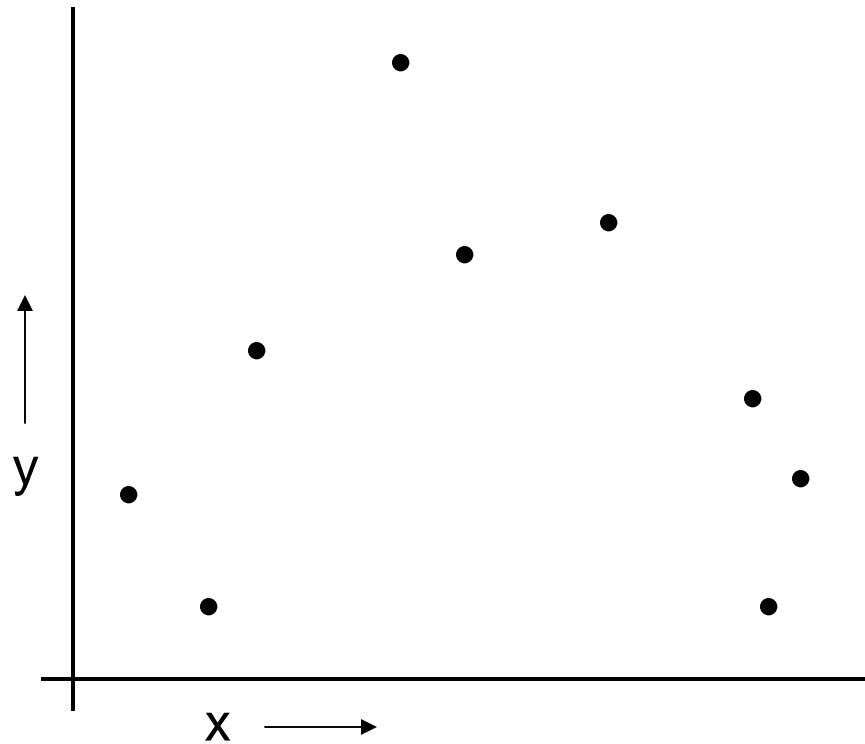
www.cs.cmu.edu/~awm

awm@cs.cmu.edu

412-268-7599

Note to other teachers and users of these slides. Andrew would be delighted if you found this source material useful in giving your own lectures. Feel free to use these slides verbatim, or to modify them to fit your own needs. PowerPoint originals are available. If you make use of a significant portion of these slides in your own lecture, please include this message, or the following link to the source repository of Andrew's tutorials: <http://www.cs.cmu.edu/~awm/tutorials> . Comments and corrections gratefully received.

A Regression Problem

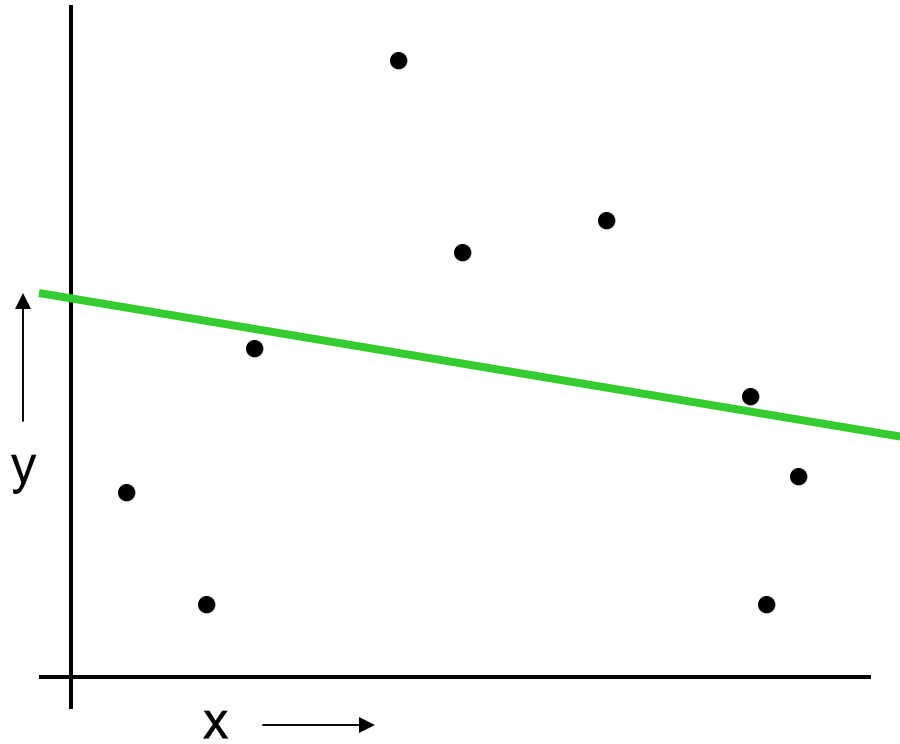


$$y = f(x) + \text{noise}$$

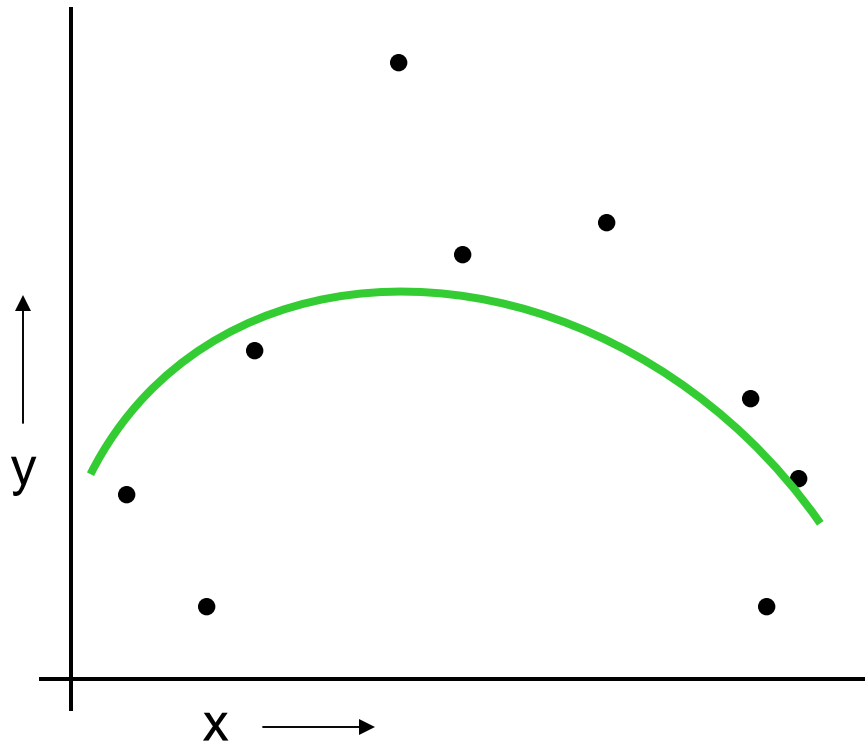
Can we learn f from this data?

Let's consider three methods...

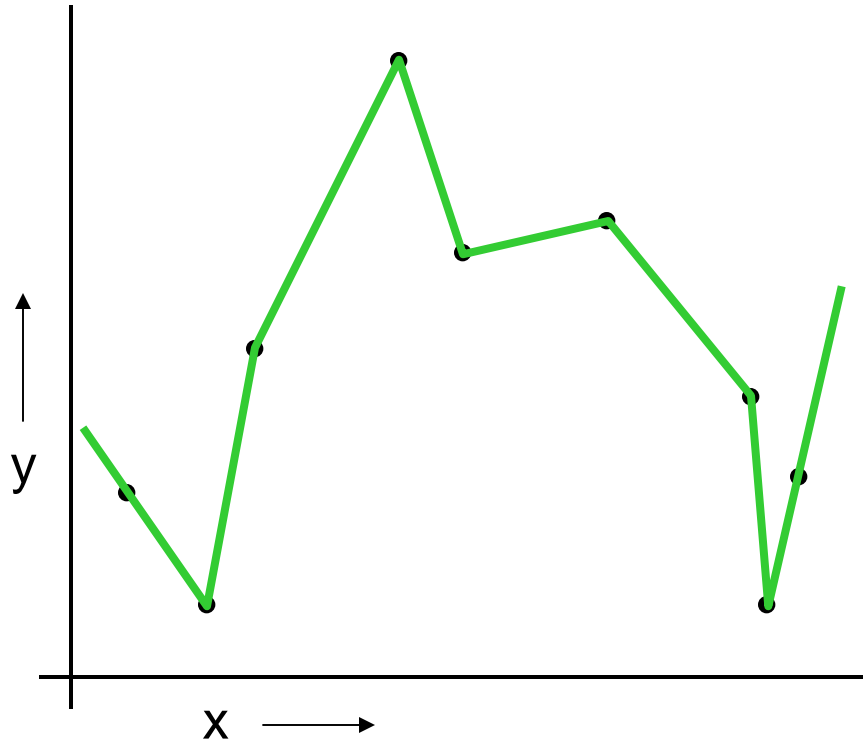
Linear Regression



Quadratic Regression

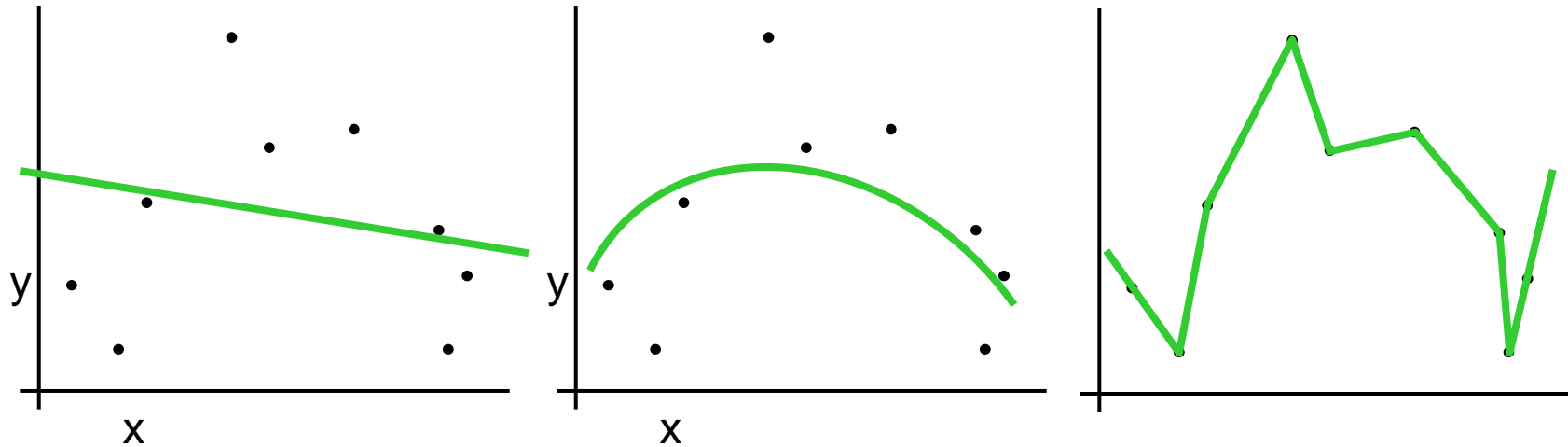


Join-the-dots



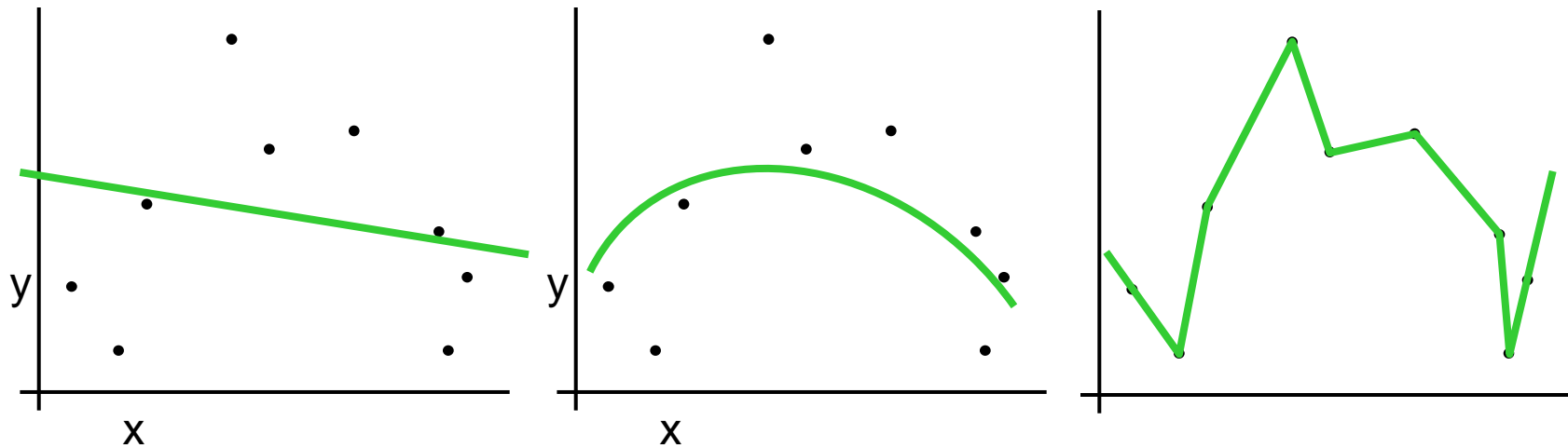
Also known as **piecewise linear nonparametric regression** if that makes you feel better

Which is best?



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

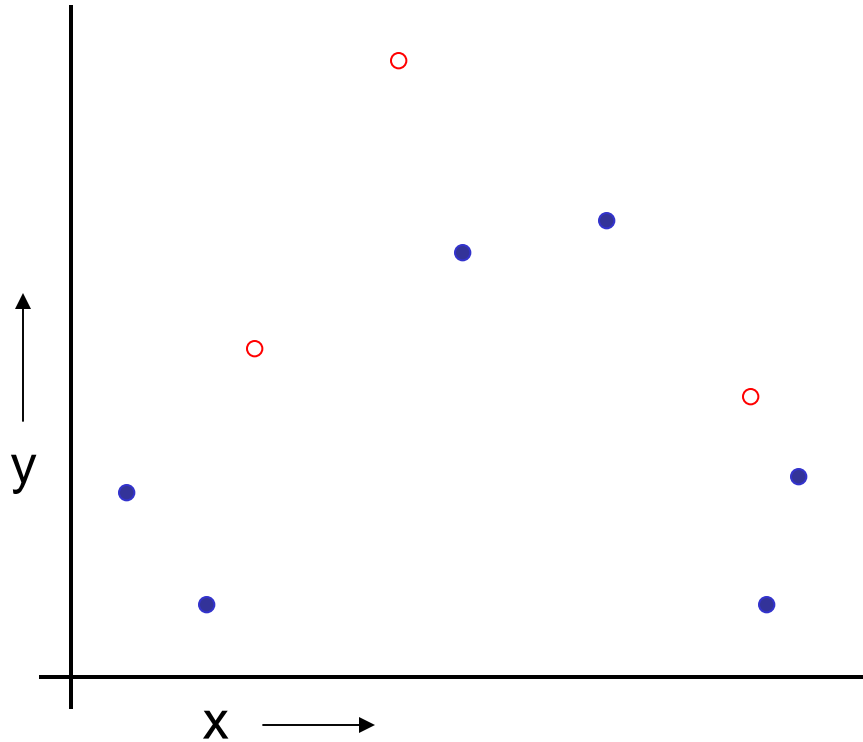
What do we really want?



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?”

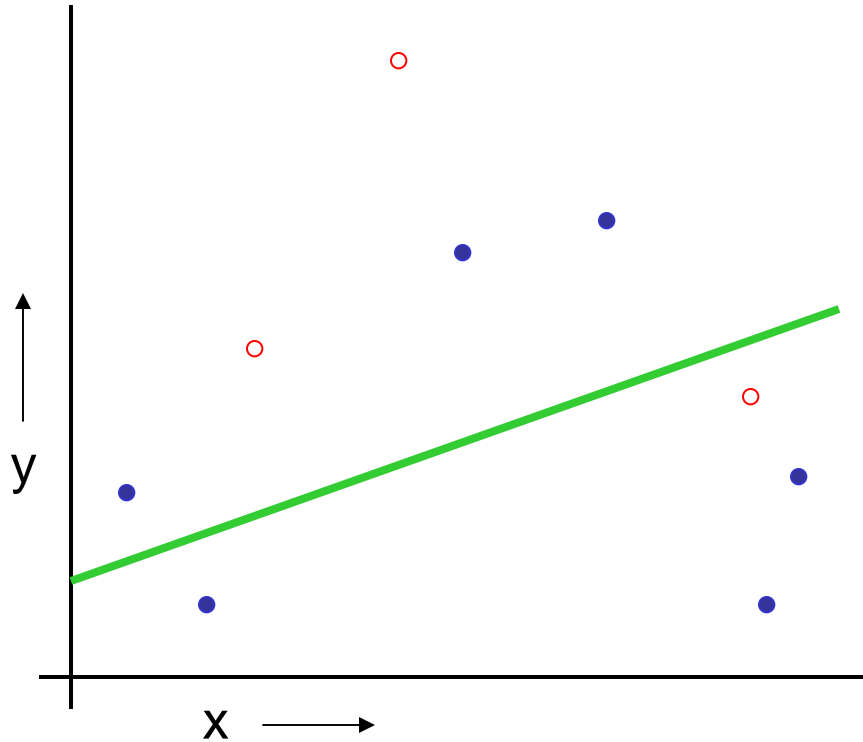
The test set method



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

2. The remainder is a
training set

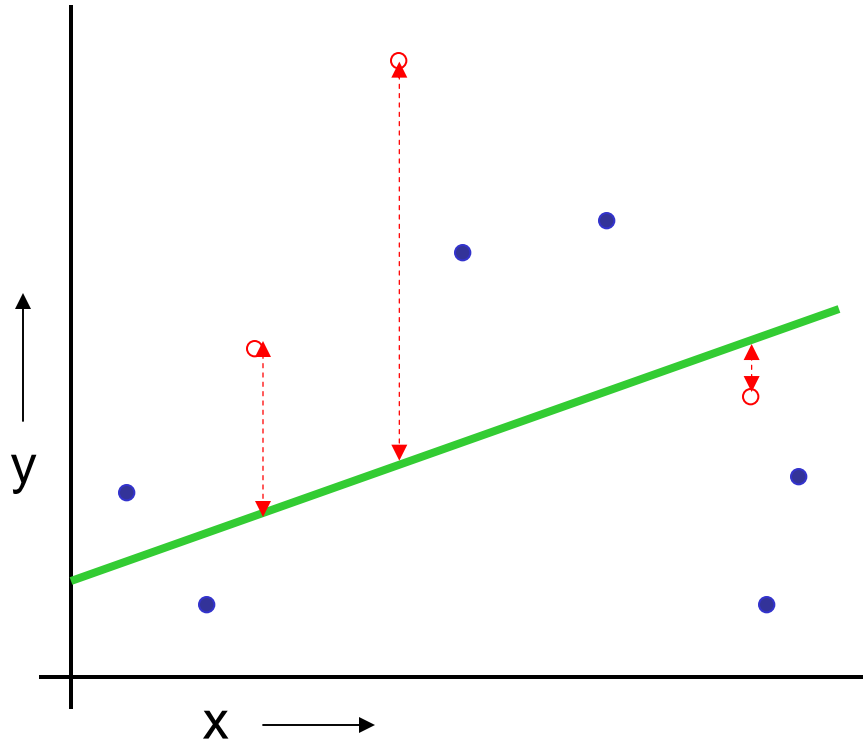
The test set method



(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

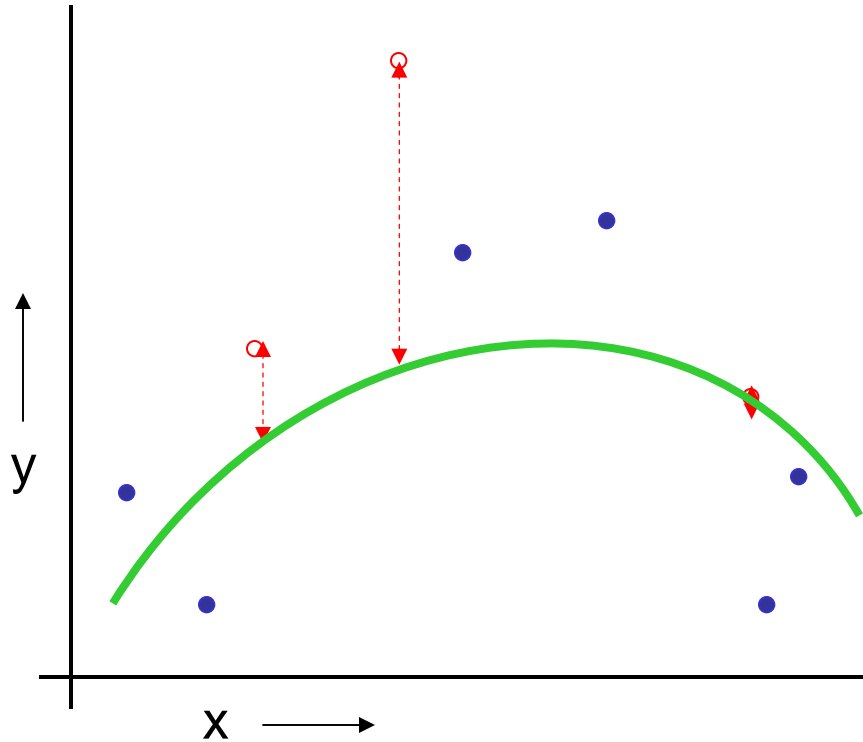
1. Randomly choose
30% of the data to be in a
test set

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

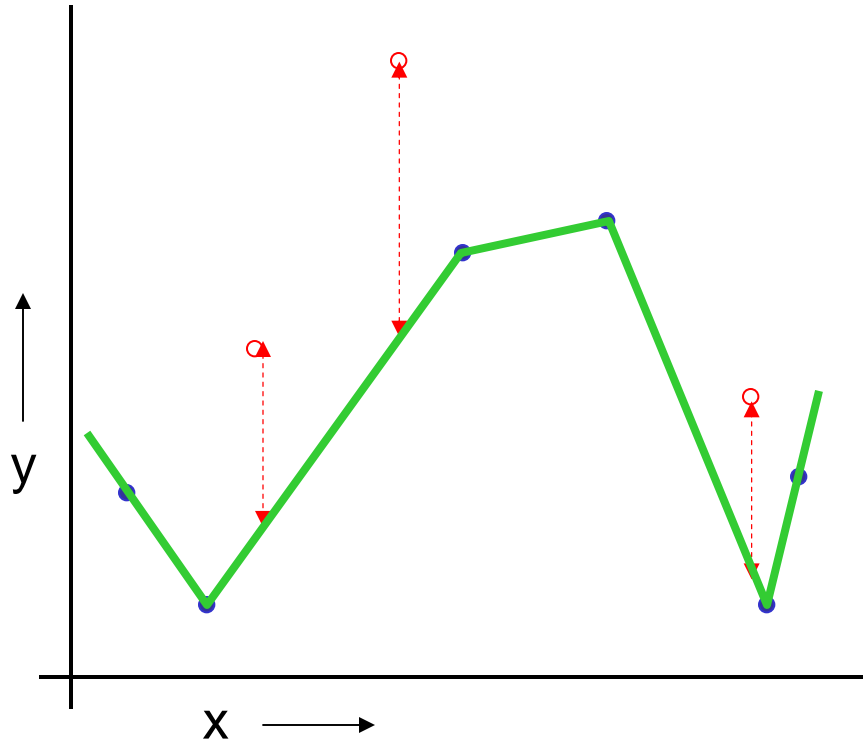
1. Randomly choose
30% of the data to be in a
test set

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



(Join the dots example)

Mean Squared Error = 2.2

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

Good news:

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

Bad news:

- What's the downside?

The test set method

Good news:

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

Bad news:

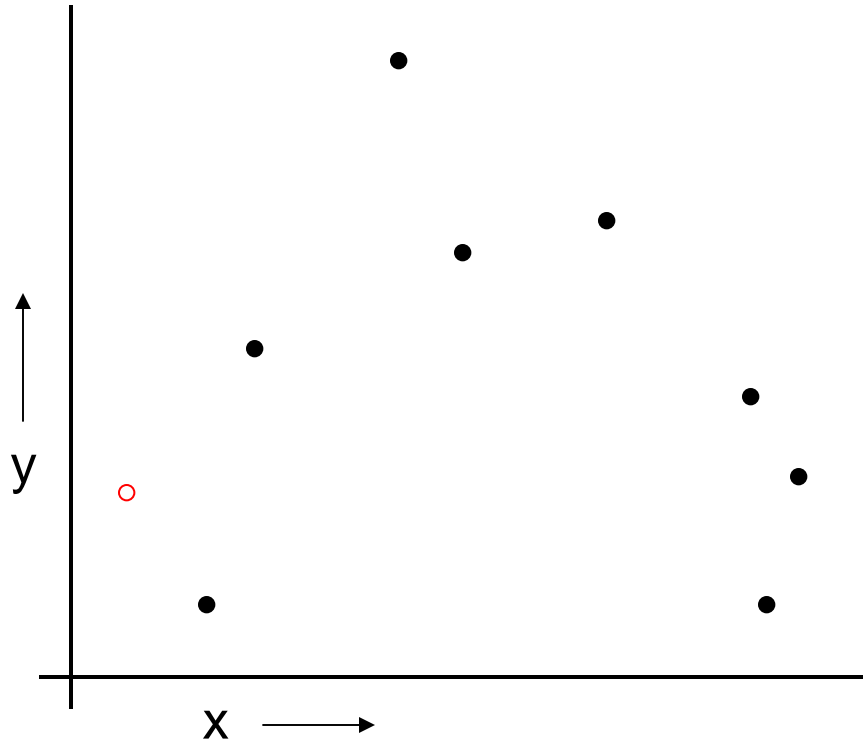
- Wastes data: we get an estimate of the best method to apply to 30% less data
- If we don't have much data, our test-set might just be lucky or unlucky

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

LOOCV (Leave-one-out Cross Validation)

For $k=1$ to R

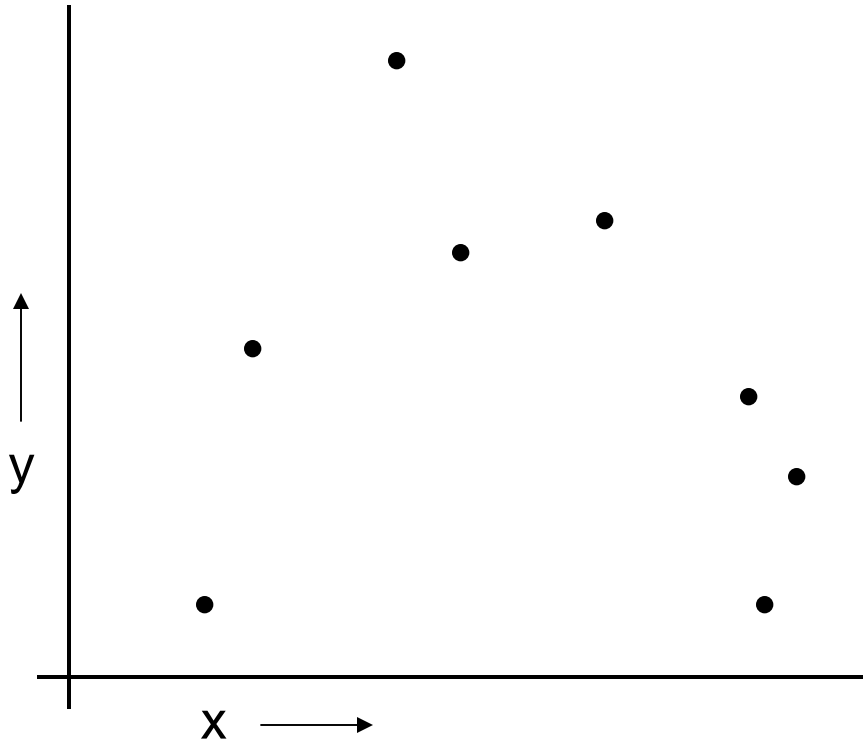
1. Let (x_k, y_k) be the k^{th} record



LOOCV (Leave-one-out Cross Validation)

For $k=1$ to R

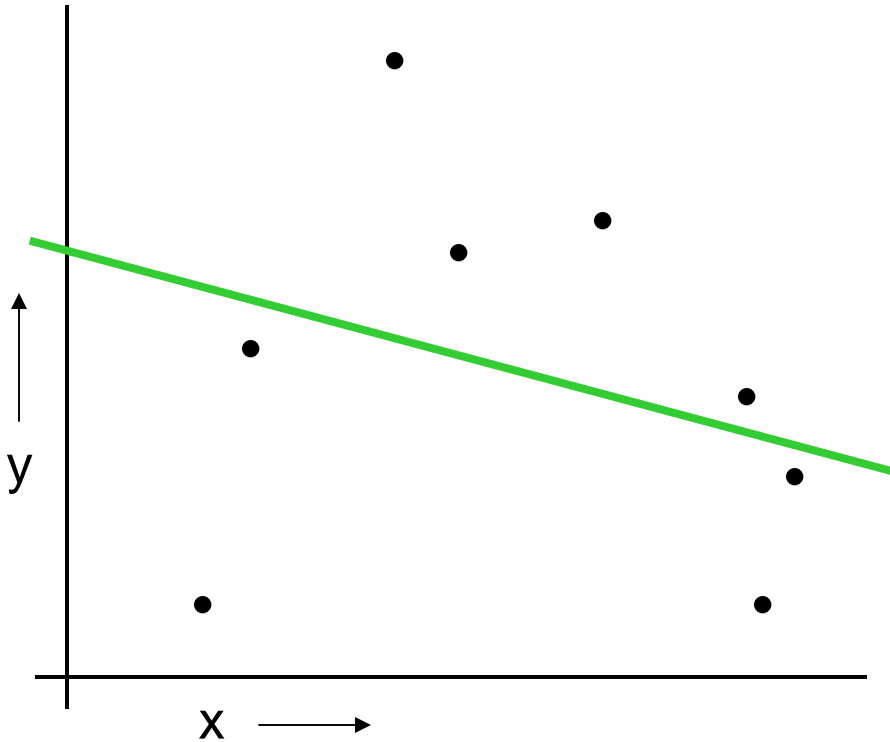
1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset



LOOCV (Leave-one-out Cross Validation)

For $k=1$ to R

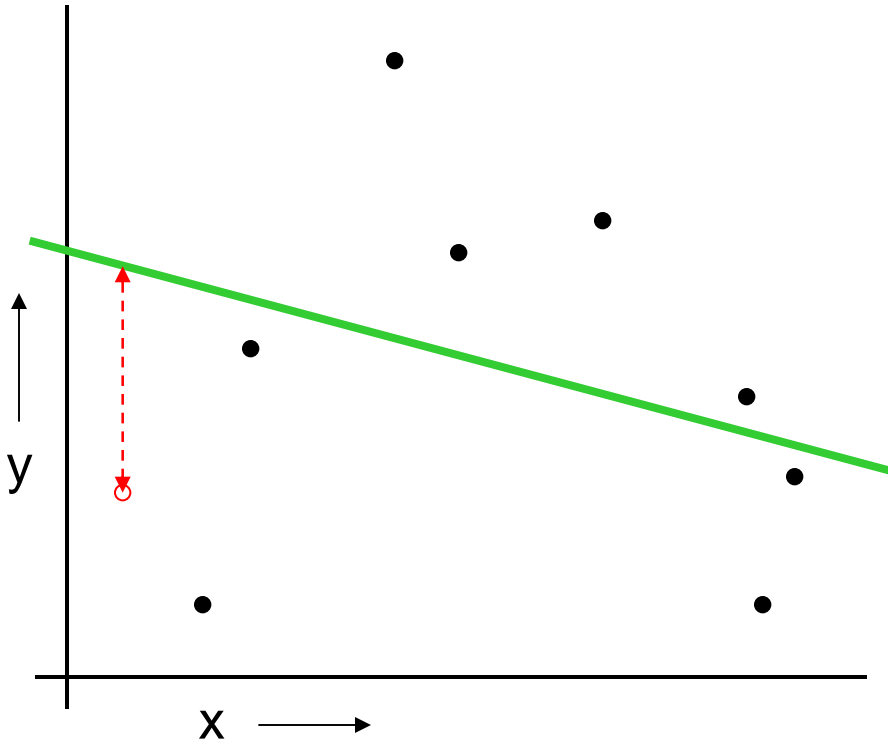
1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining $R-1$ datapoints



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)

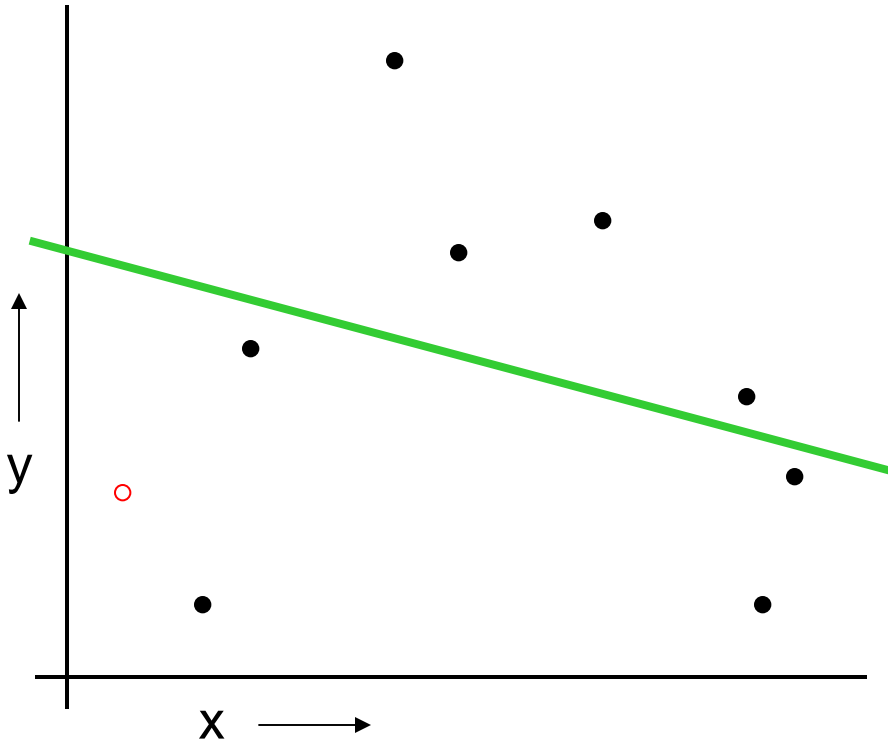


LOOCV (Leave-one-out Cross Validation)

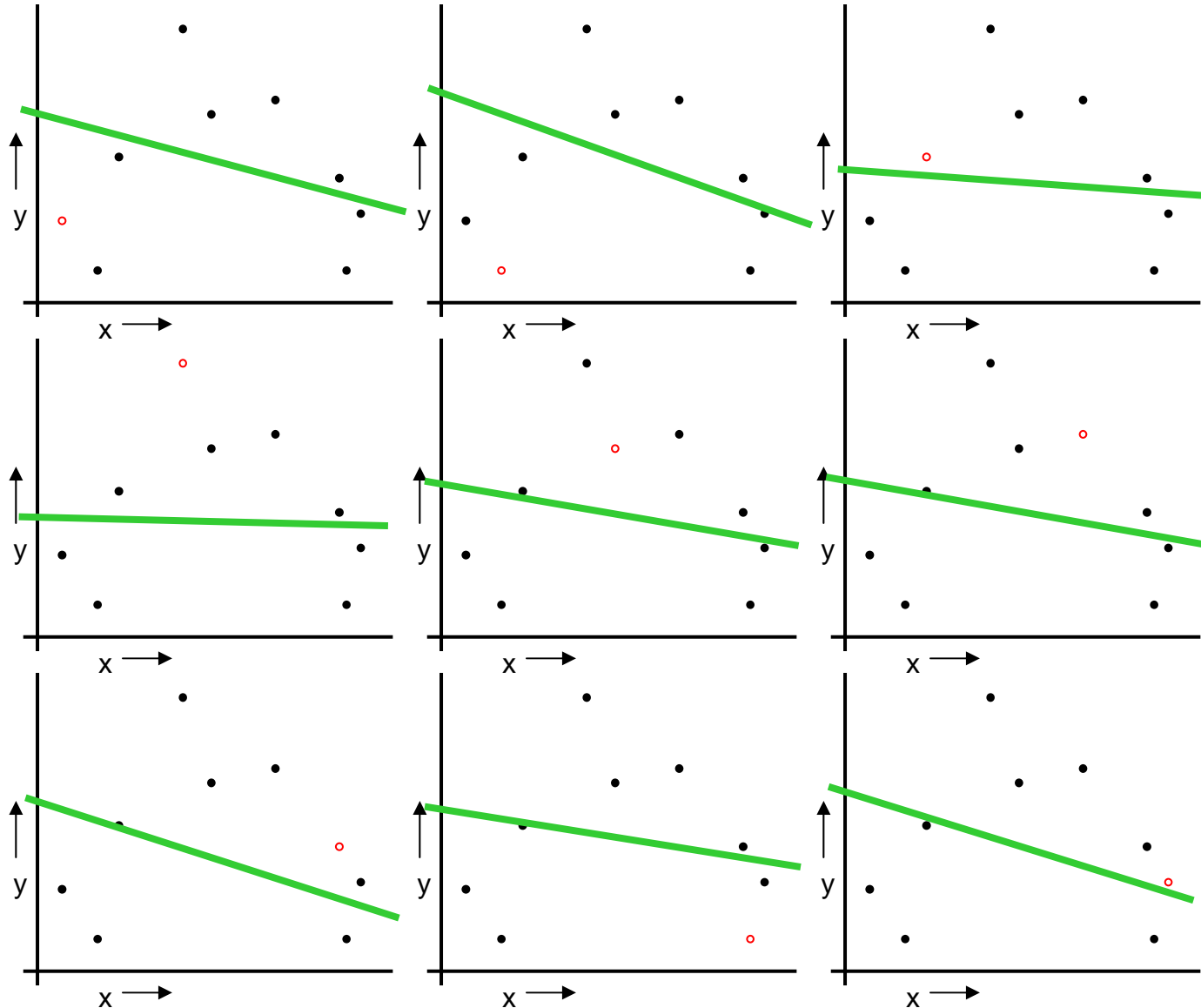
For $k=1$ to R

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 (Leave-one-out Cross Validation)



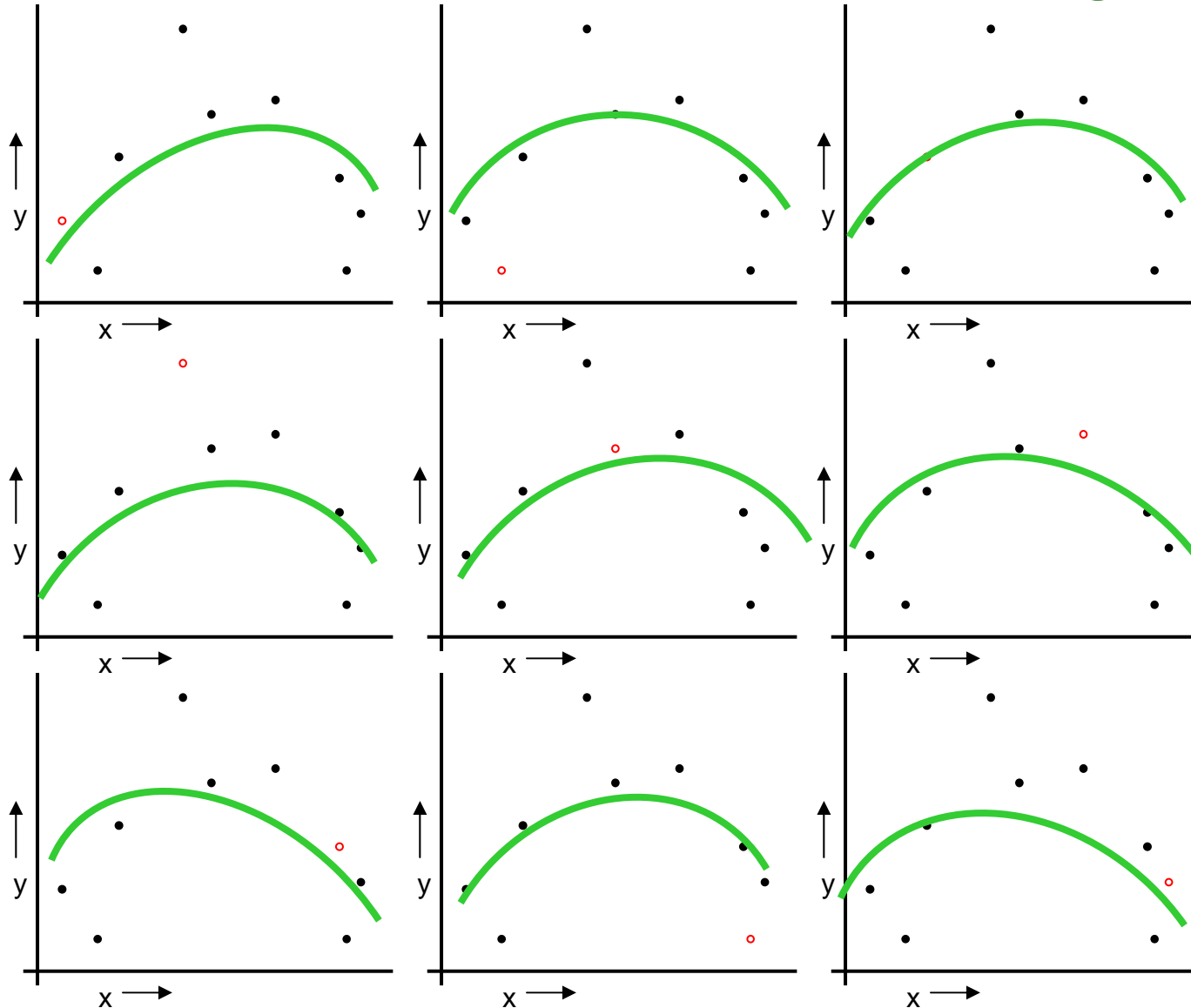
For $k=1$ to R

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.

$$MSE_{LOOCV} = 2.12$$

LOOCV for Quadratic Regression



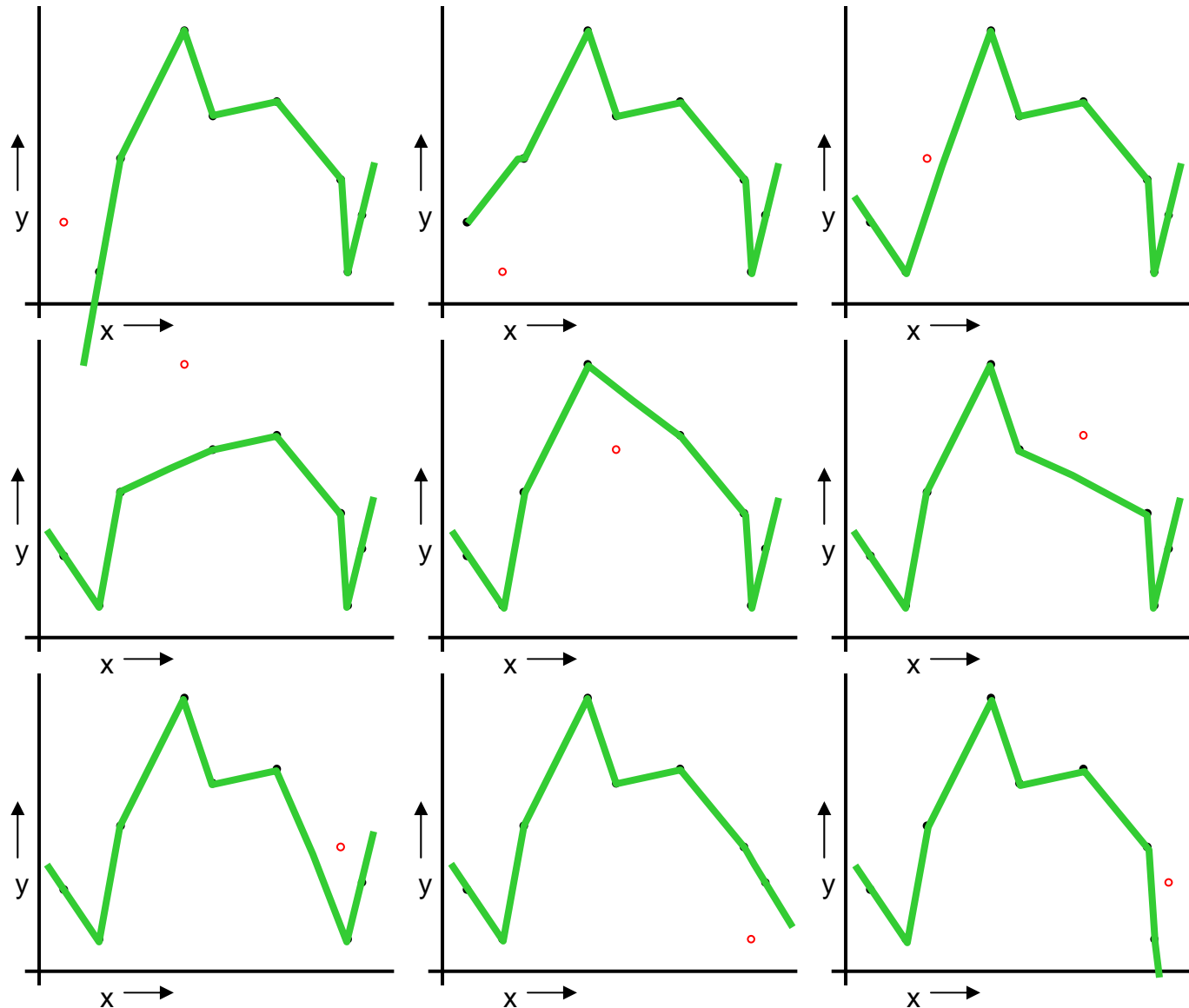
For $k=1$ to R

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.

$$MSE_{\text{LOOCV}} = 0.962$$

LOOCV for Join The Dots



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.

$$MSE_{LOOCV} = 3.33$$

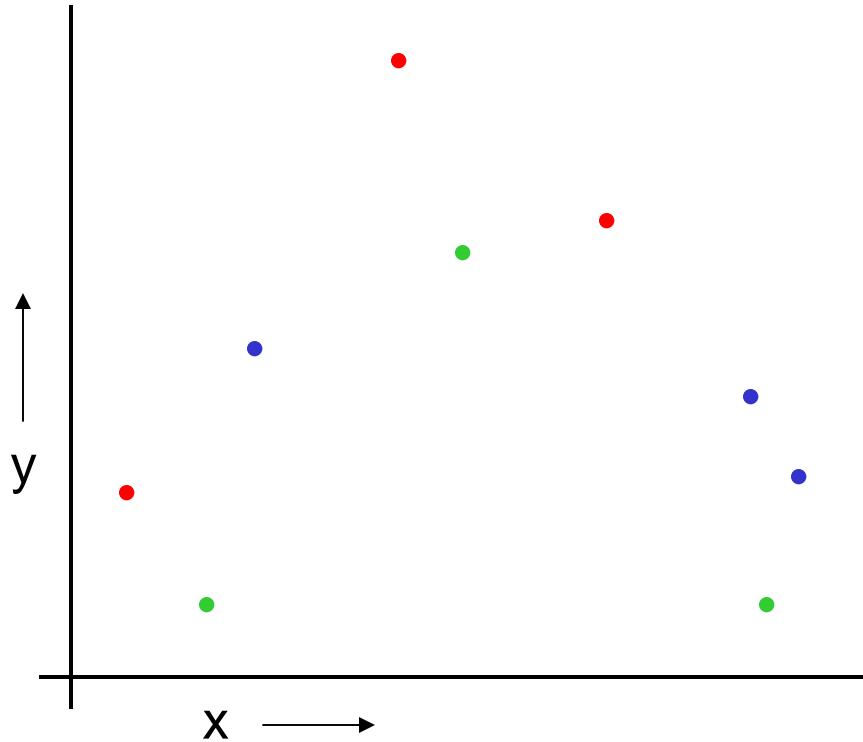
Which kind of Cross Validation?

	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?

k-fold Cross Validation

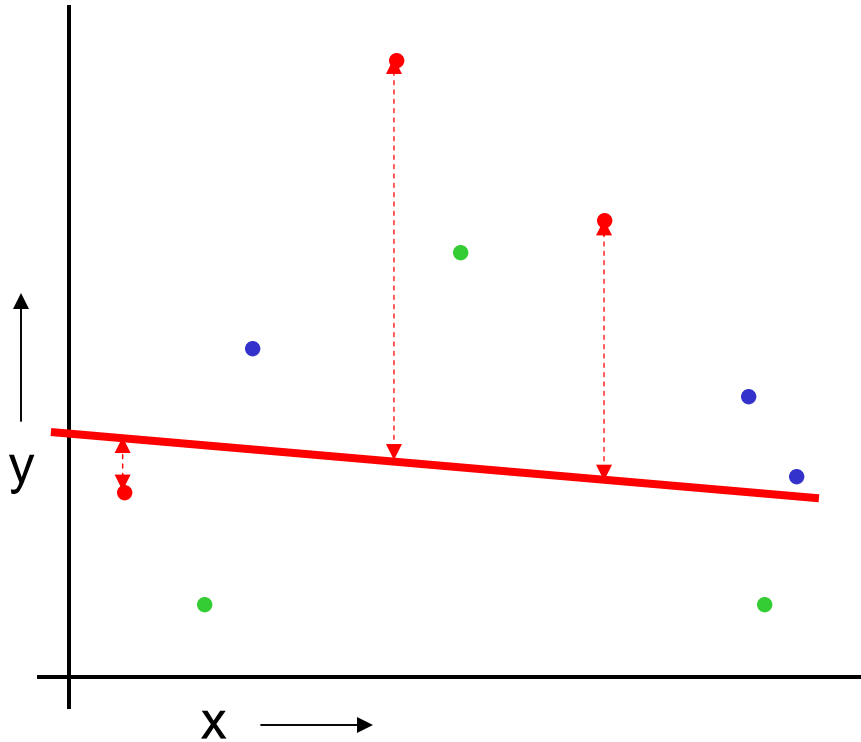
Randomly break the dataset into k partitions (in our example we'll have $k=3$ partitions colored Red Green and Blue)



k-fold Cross Validation

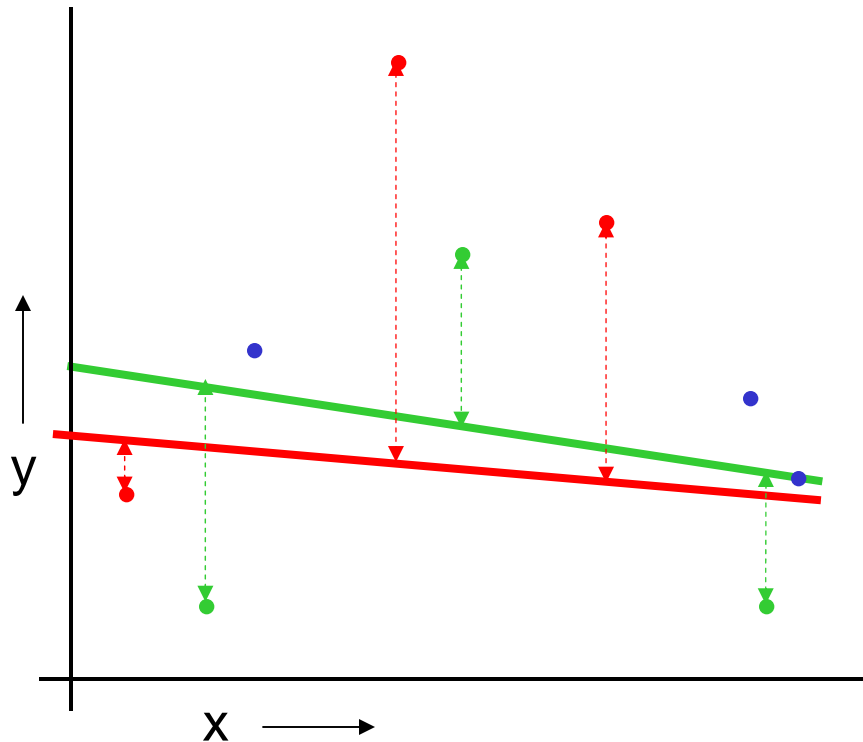
Randomly break the dataset into k partitions (in our example we'll have $k=3$ partitions colored Red Green and Blue)

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.



k-fold Cross Validation

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

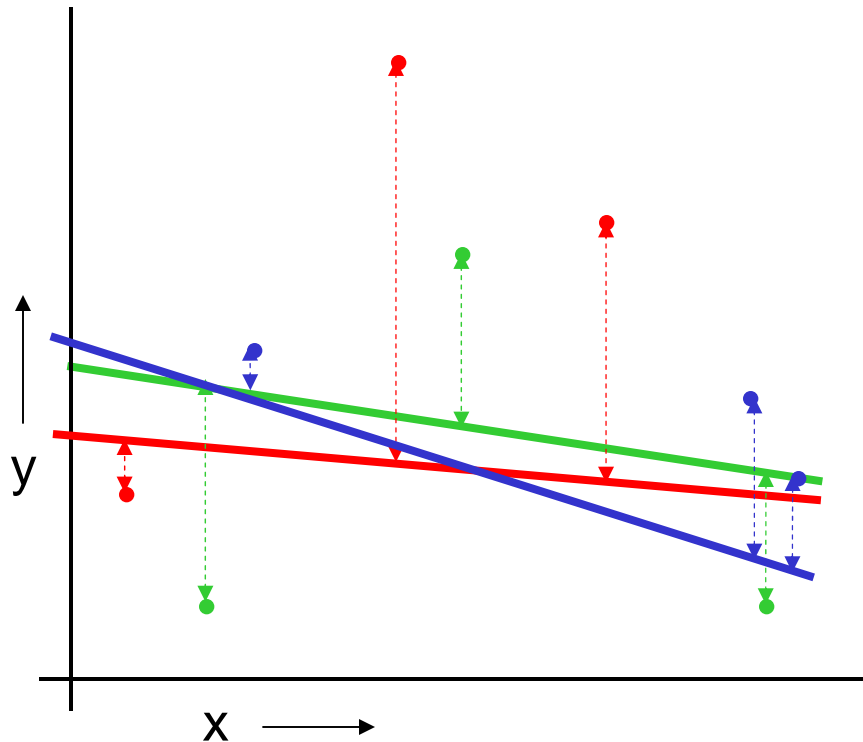


For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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.

k-fold Cross Validation

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



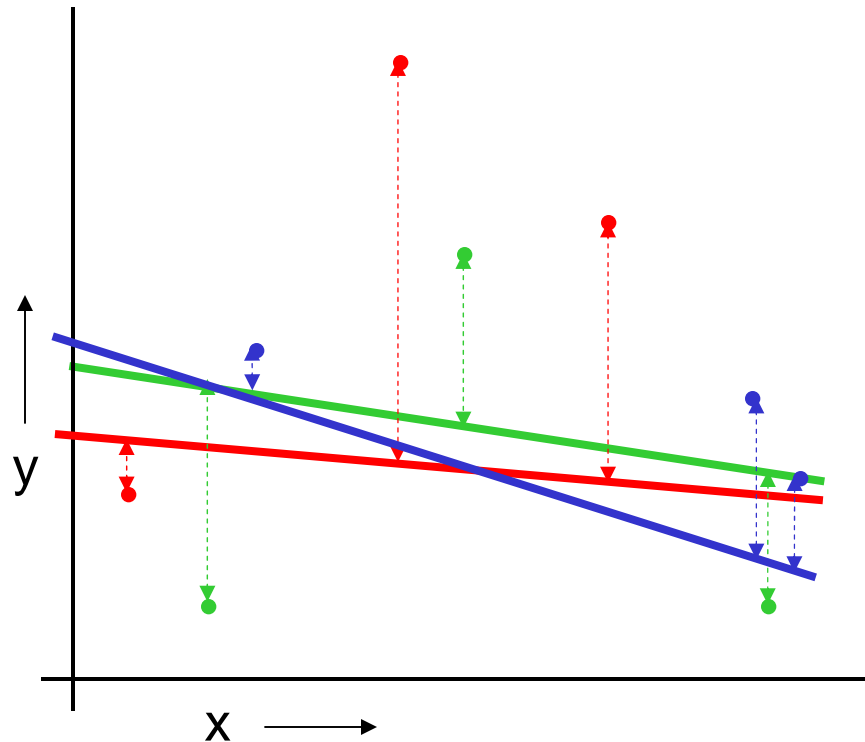
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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.

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red 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.

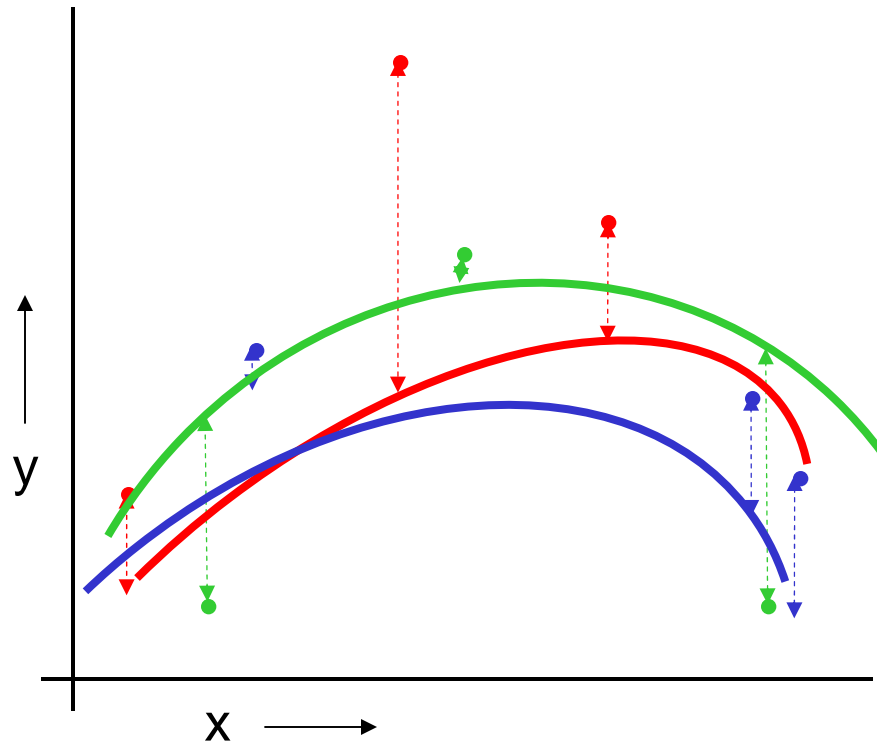
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the 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

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red 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 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

k-fold Cross Validation

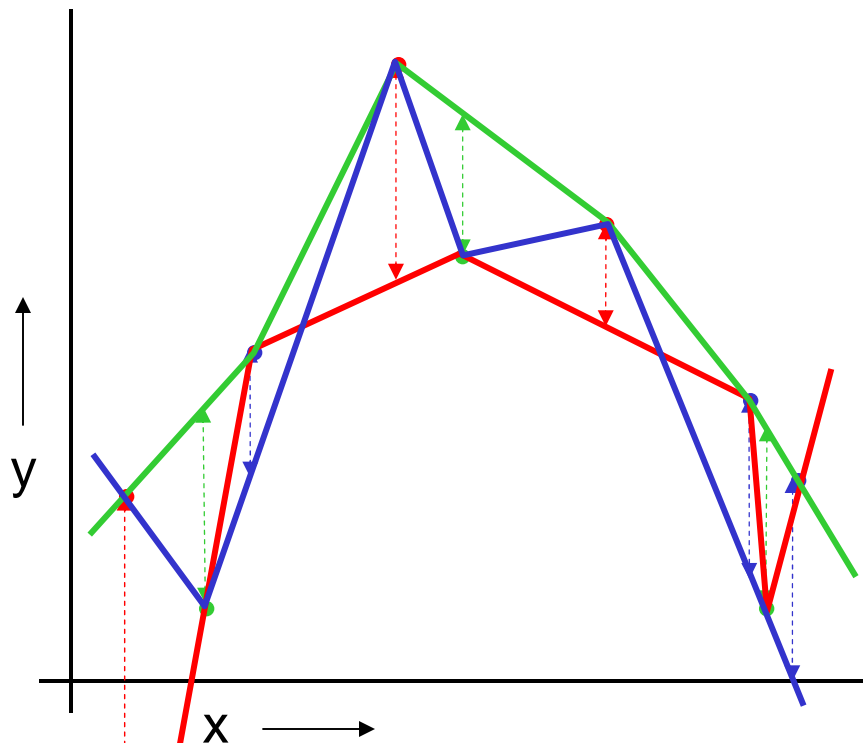
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

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



Joint-the-dots
 $MSE_{3FOLD}=2.93$

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 R times.
3-fold	Wastier than 10-fold. Expensivier than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	












Which kind of Cross Validation?

	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	
10-fold	Wastes 10% of the data 10 times more expensive than testset	ly sive
3-fold	Wastier than 10-fold. Expensivier than testset	instead of R times. Slightly better than test-set
R-fold	Identical to Leave-one-out	

But note: One of Andrew's joys in life is algorithmic tricks for making these cheap

CV-based Model Selection

- We're trying to decide which algorithm to use.
- We train each machine and make a table...

i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			
4	f_4			
5	f_5			
6	f_6	