Cross-validation for detecting and preventing overfitting

Andrew W. Moore

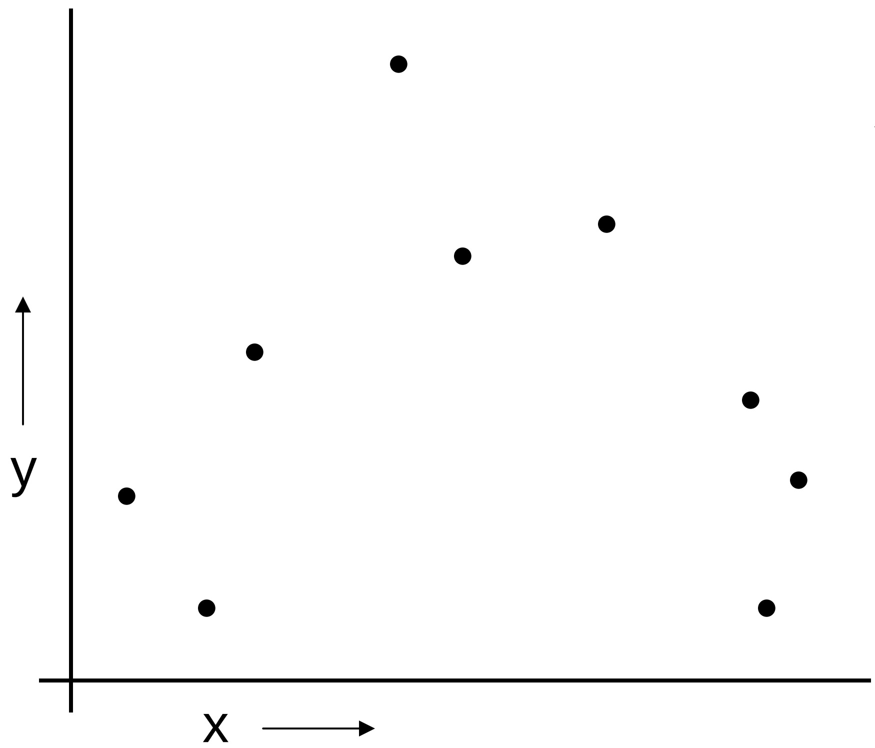
Professor

School of Computer Science

Carnegie Mellon University

Some edits by MW

A Regression Problem

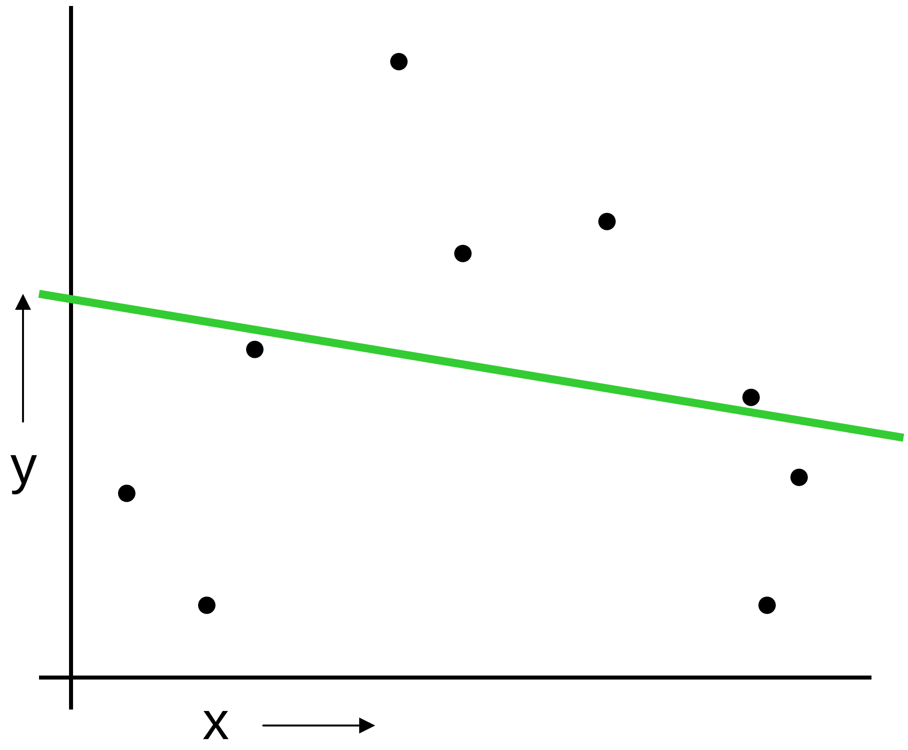


y = f(x) + noise

Can we learn f from this data?

Let's consider three methods…

Linear Regression



Linear Regression

Univariate Linear regression with a constant term:

X	Y
3	7
1	3
:	:

$$\mathbf{X}=\begin{bmatrix}3\\1\\:\end{bmatrix} \qquad \mathbf{y}=\begin{bmatrix}7\\3\\:\end{bmatrix}$$

$\mathbf{x}_1=(3)$.. $\qquad y_1=7$..

Originally discussed in the previous Andrew Lecture: “Neural Nets”

Linear Regression

Univariate Linear regression with a constant term:

X	*Y*
3	7
1	3
:	:

$\mathbf{X}=\begin{bmatrix}3\\1\\:\end{bmatrix}$ $\mathbf{y}=\begin{bmatrix}7\\3\\:\end{bmatrix}$

$y_1=7..$

$\mathbf{X}=\begin{bmatrix}1 & 3\\1 & 1\\: & \end{bmatrix}$ $\mathbf{y}=\begin{bmatrix}7\\3\\:\end{bmatrix}$

$\boldsymbol{x}_1=(1,3)'.$ $y_1=7..$

$.\boldsymbol{x}_k=(1,x_k)'$

Linear Regression

Univariate Linear regression with a constant term:

X	*Y*
3	7
1	3
:	:

X=

3
1
:

y=

7
3
:

$y_1=7..$

X'=

1	3
1	1
:	

y=

7
3
:

$\mathbf{x}_1=(1,3)..$

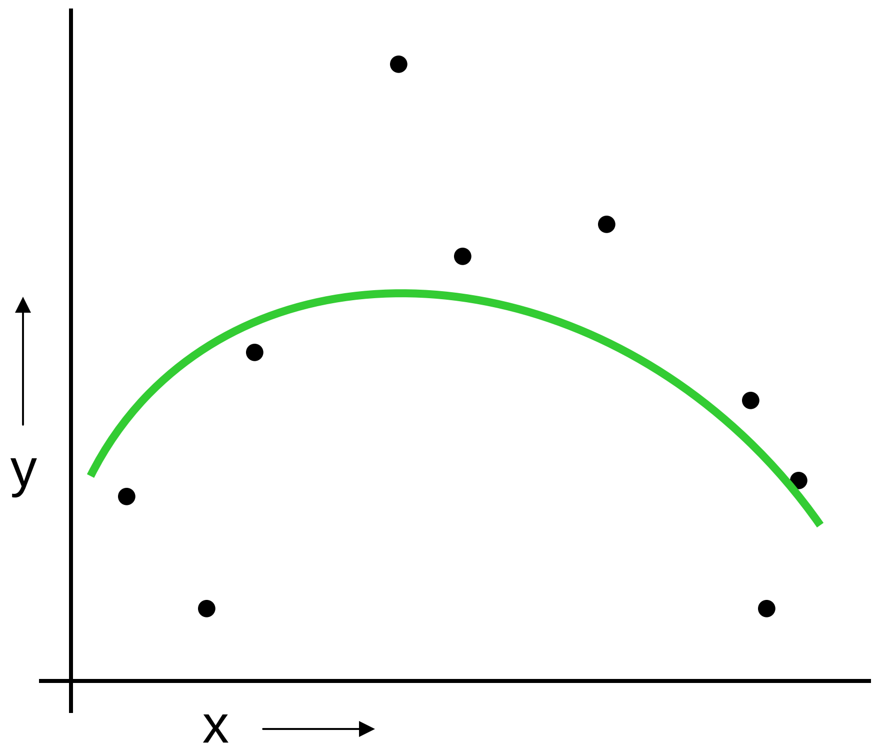
$y_1=7..$

$\mathbf{x}_k=(1,x_k)$

$$\beta=(\mathbf{XX'})^{-1}(\mathbf{Xy})$$

$$y^{est} = \beta_0+ \beta_1 x$$

Quadratic Regression



Quadratic Regression

X	Y
3	7
1	3
:	:

$$\mathbf{X} = \begin{bmatrix} 3 \\ 1 \\ : \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ : \end{bmatrix}$$

$y_1 = 7..$

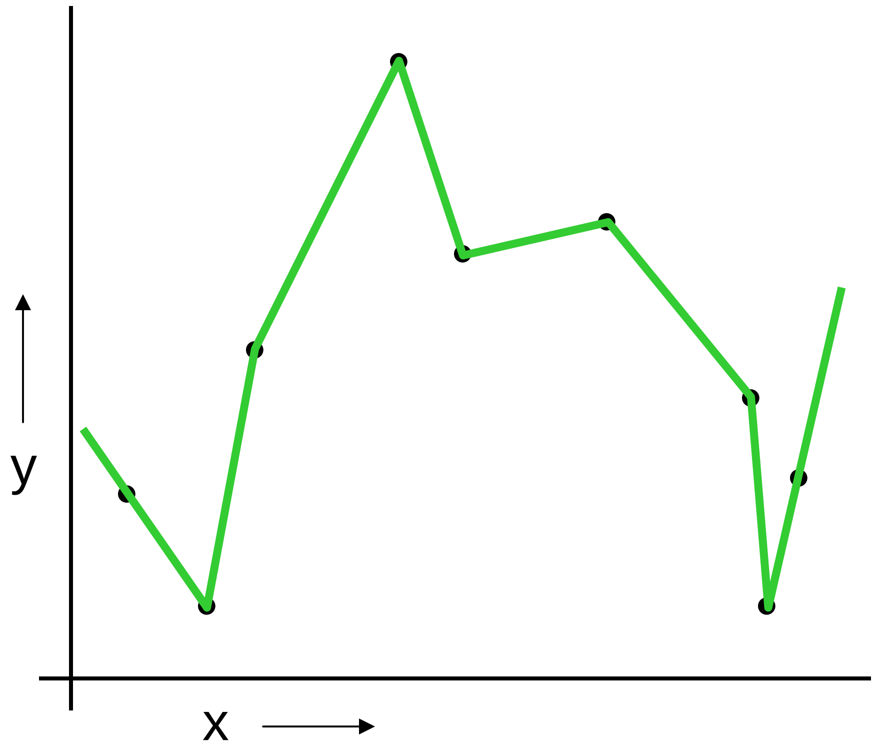
$$\mathbf{X'} = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 1 & 1 \\ : & & \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ : \end{bmatrix}$$

$\boldsymbol{x} = (1\ ,\ x, x^2)'$

$$\omega = (\mathbf{XX'})^{-1}(\mathbf{Xy})$$

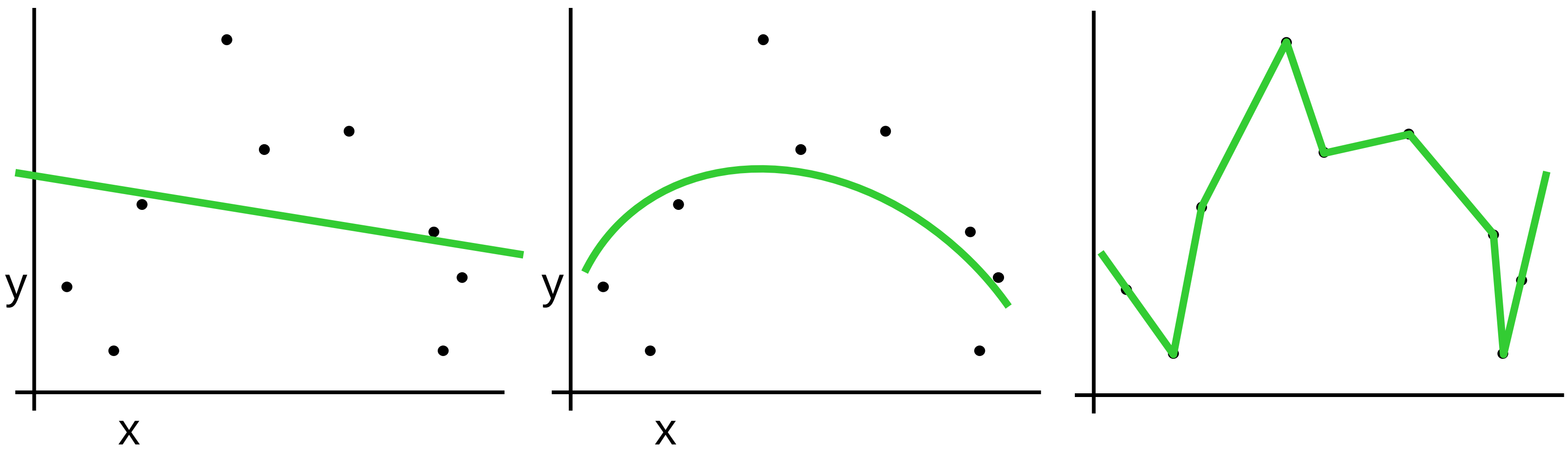
$$y^{est} = \omega_0 + \omega_1 x + \omega_2 x^2$$

Join-the-dots



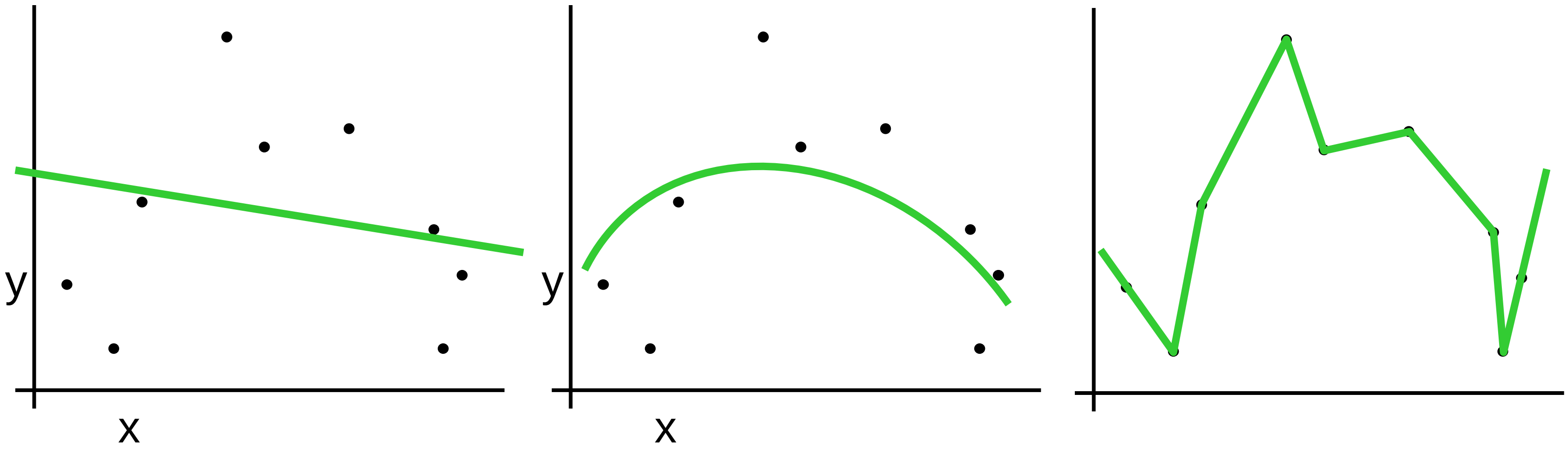
Also known as piecewise linear nonparametric regression if that makes you feel better

Which is best?



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

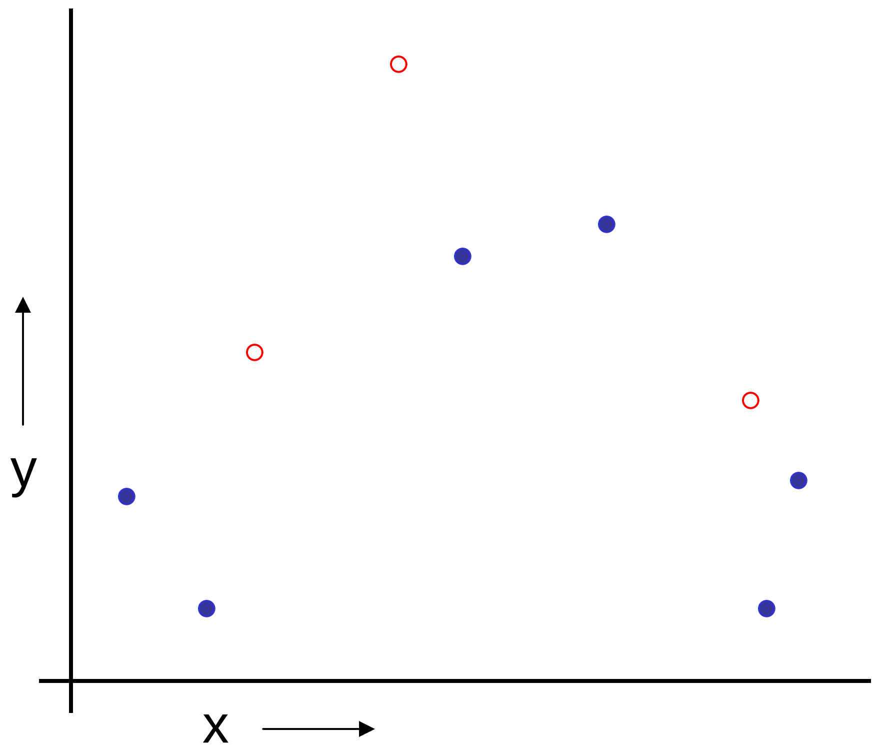
What do we really want?



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?"

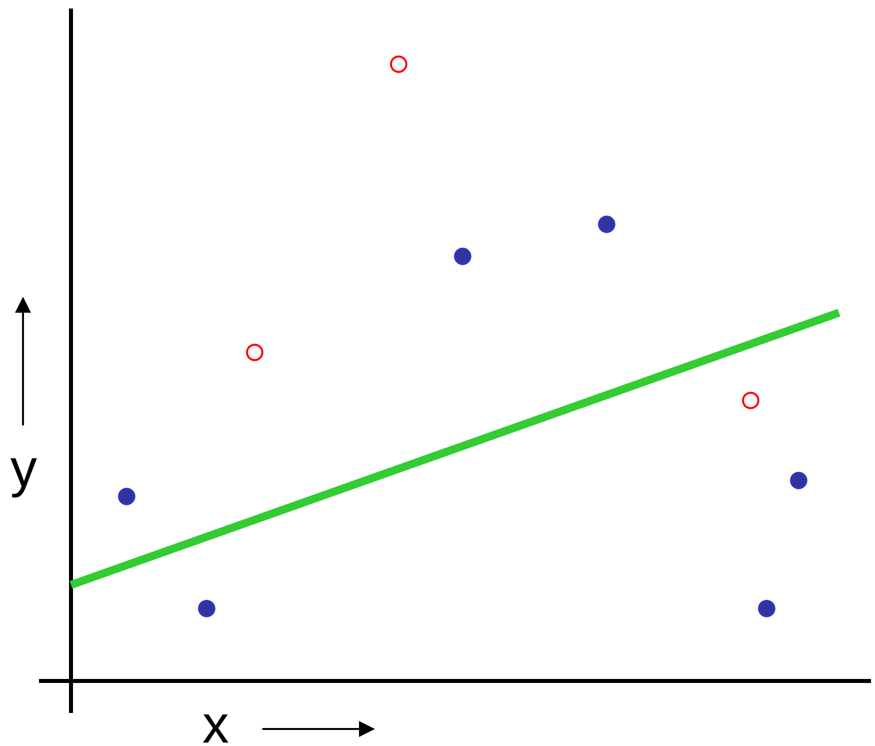
The single split method



1. Randomly choose 30% of the training set to be in a validation set

2. The remainder is a actual training set

The single split method



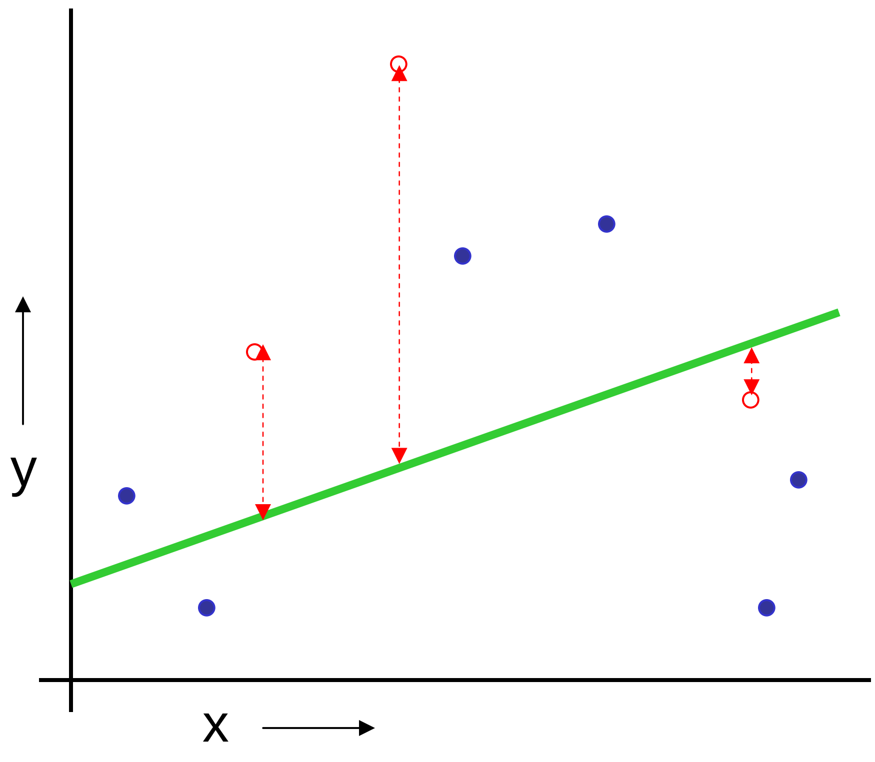
1. Randomly choose 30% of the training set to be in a validation set

2. The remainder is a actual training set

3. Perform your regression on the actual training set

(Linear regression example)

The single split method



(Linear regression example)

Mean Squared Error = 2.4

0. Split in to training and test set

1. Randomly choose 30% of the training set to be in a validation set

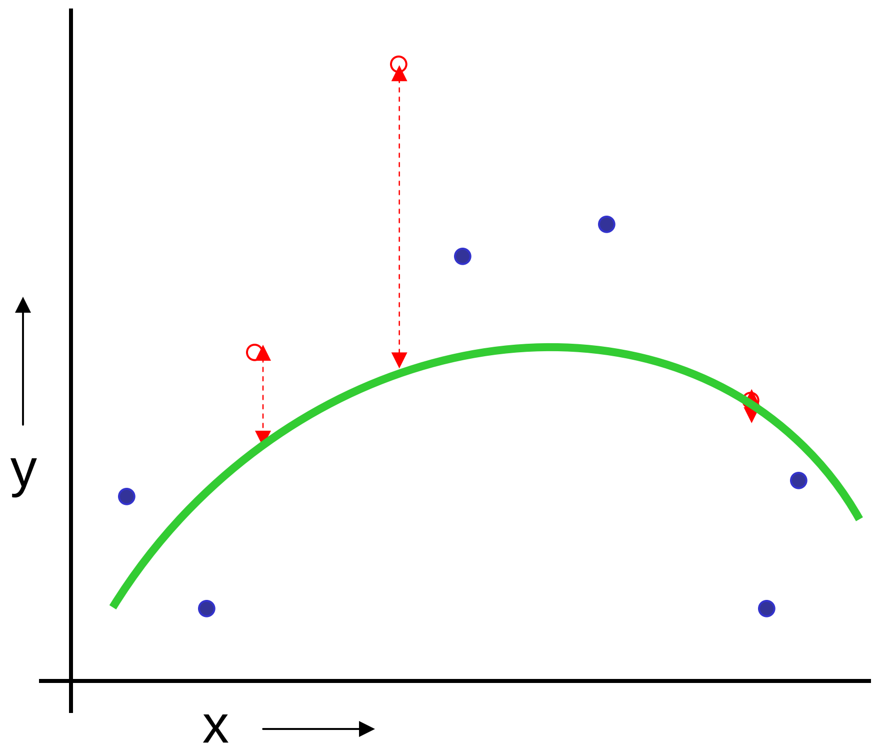
2. The remainder is a actual training set

3. Perform your regression on the actual training set

4. Choose best model based on validation set

5. Report performance on test set

The single split method



(Quadratic regression example)

Mean Squared Error = 0.9

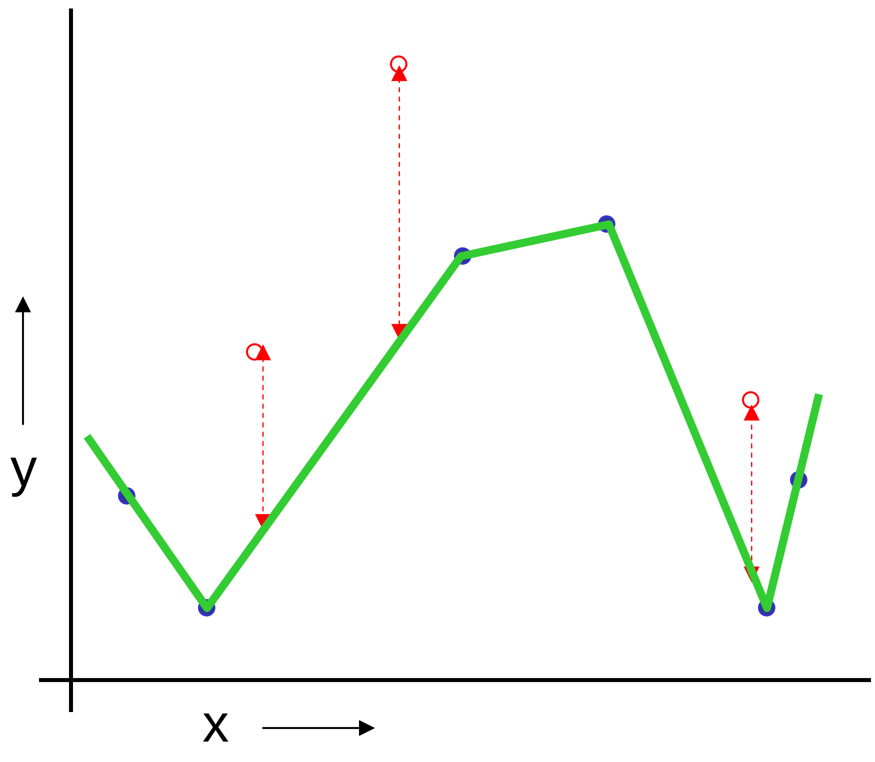
1. Randomly choose 30% of the data to be in a test set

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 single split method



(Join the dots example)

Mean Squared Error = 2.2

1. Randomly choose 30% of the training set to be in a validation set

2. The remainder is a actual training set

3. Perform your regression on the actual training set

4. Estimate your future performance with the test set

The single split method

Good news:

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

Bad news:

- What's the downside?

The single split method

Good news:

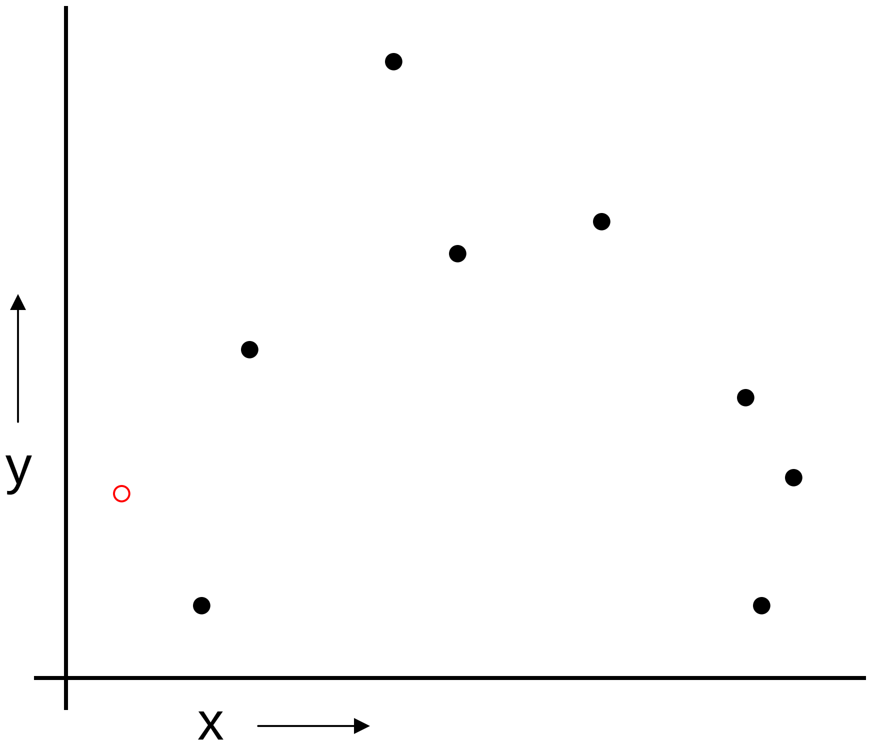
- Very very simple
- Can then simply choose the method with the best validation-set score

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 validation might just be lucky or unlucky

We say the "validation-set estimator of performance has high variance"

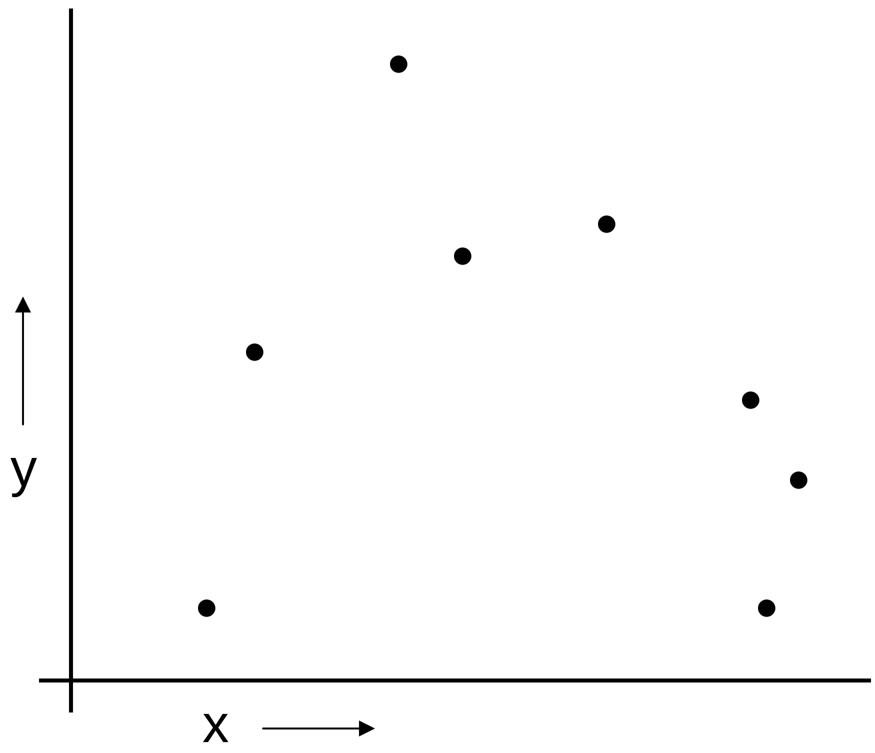
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

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

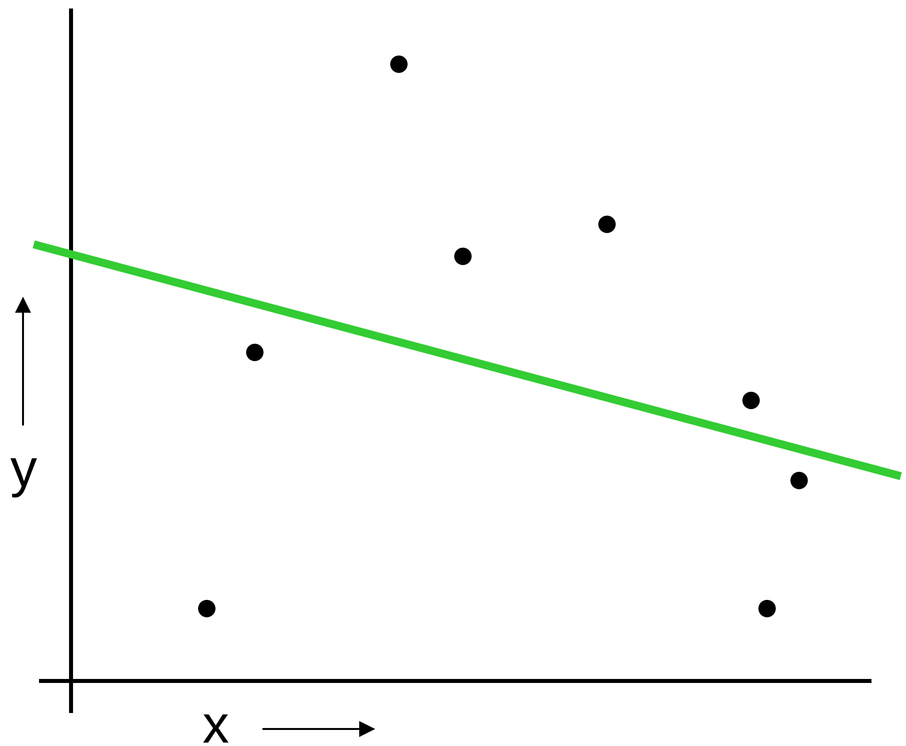
LOOCV (Leave-one-out Cross Validation)



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

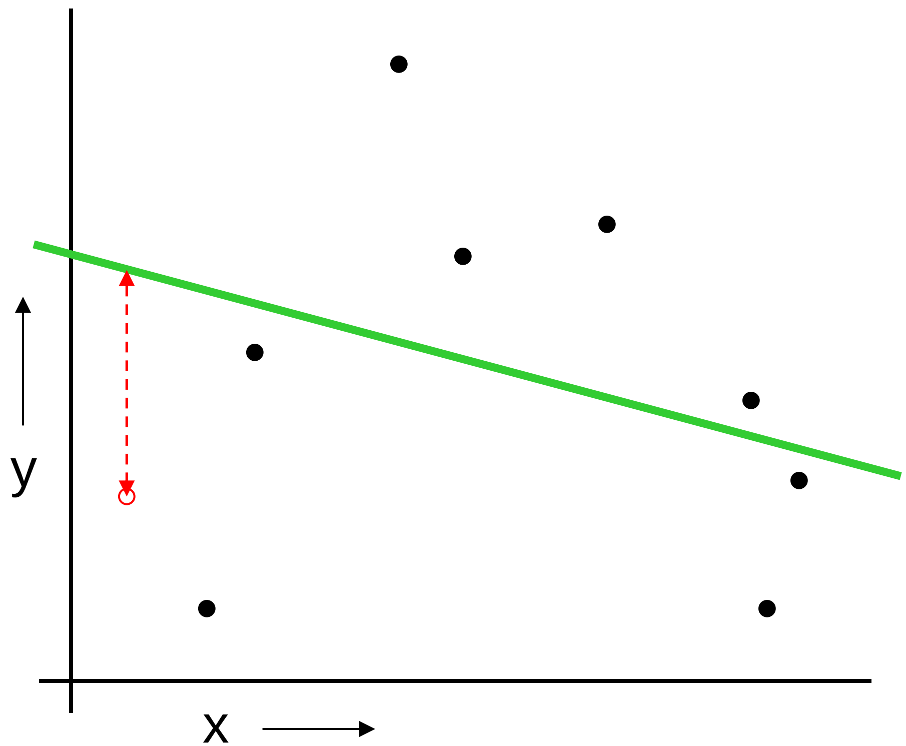
LOOCV (Leave-one-out Cross Validation)



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

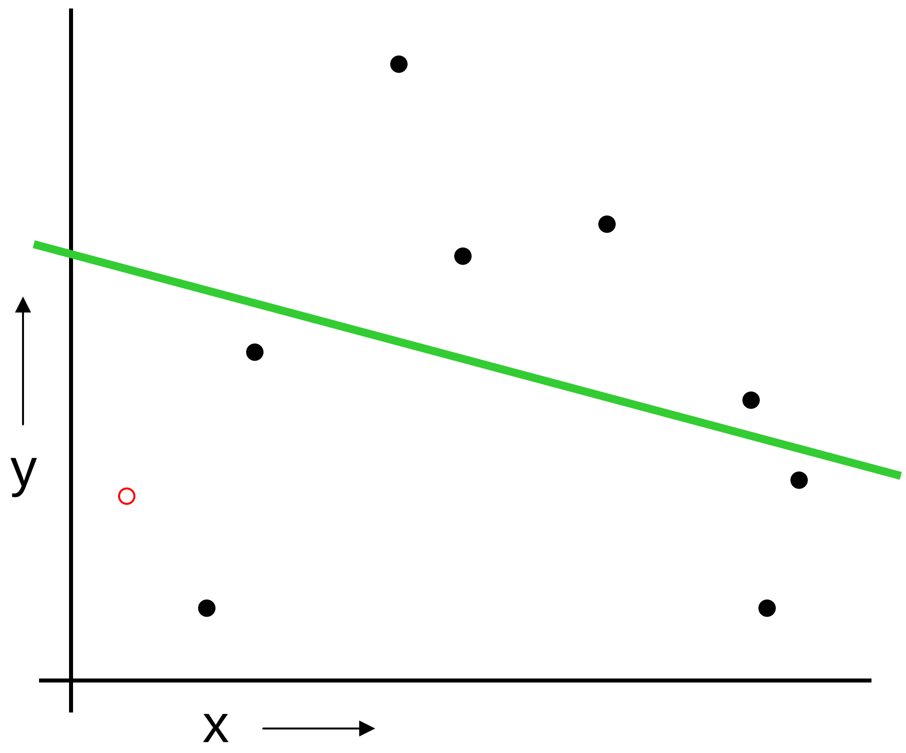
LOOCV (Leave-one-out Cross Validation)



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)

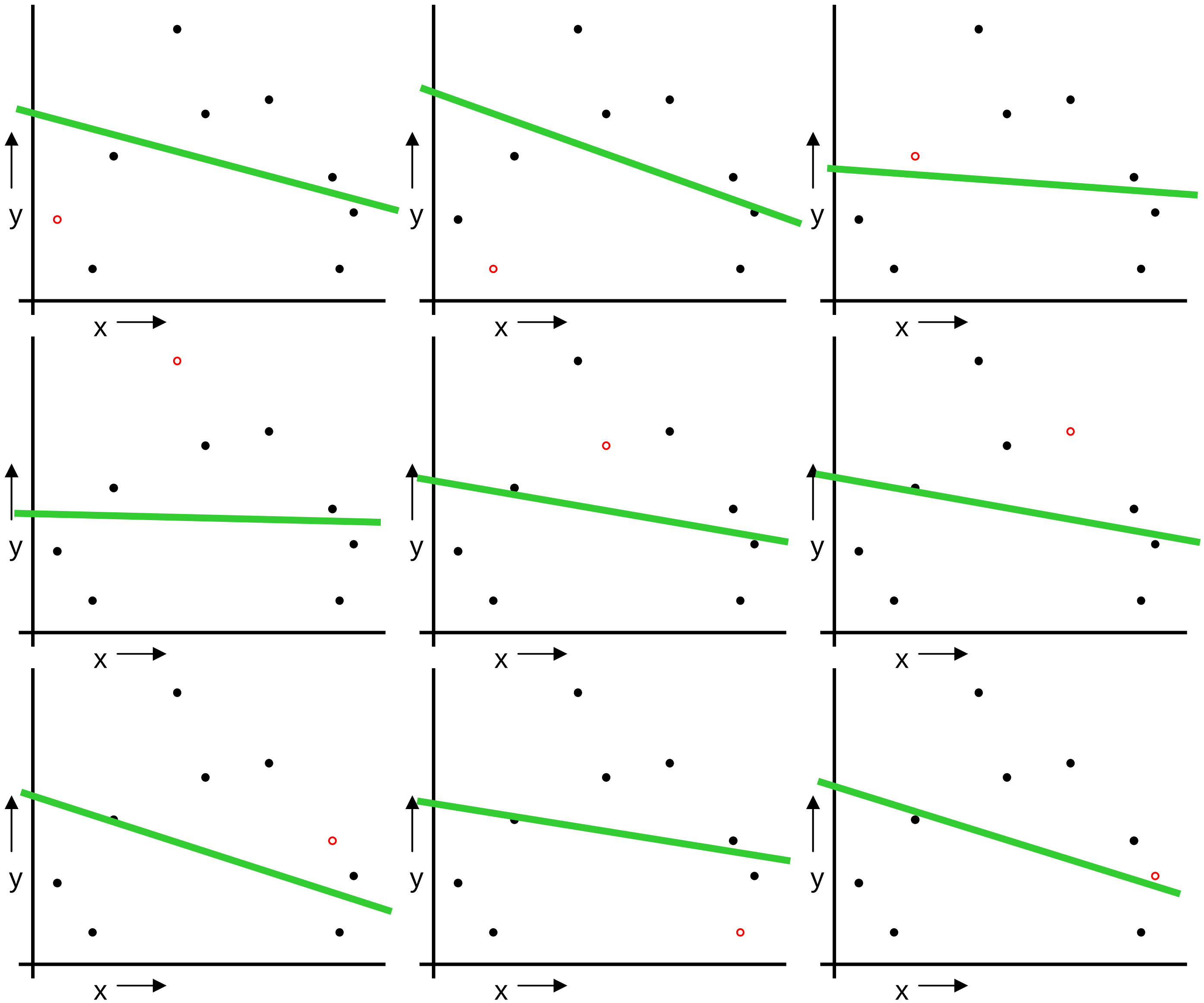


For k=1 to R

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)



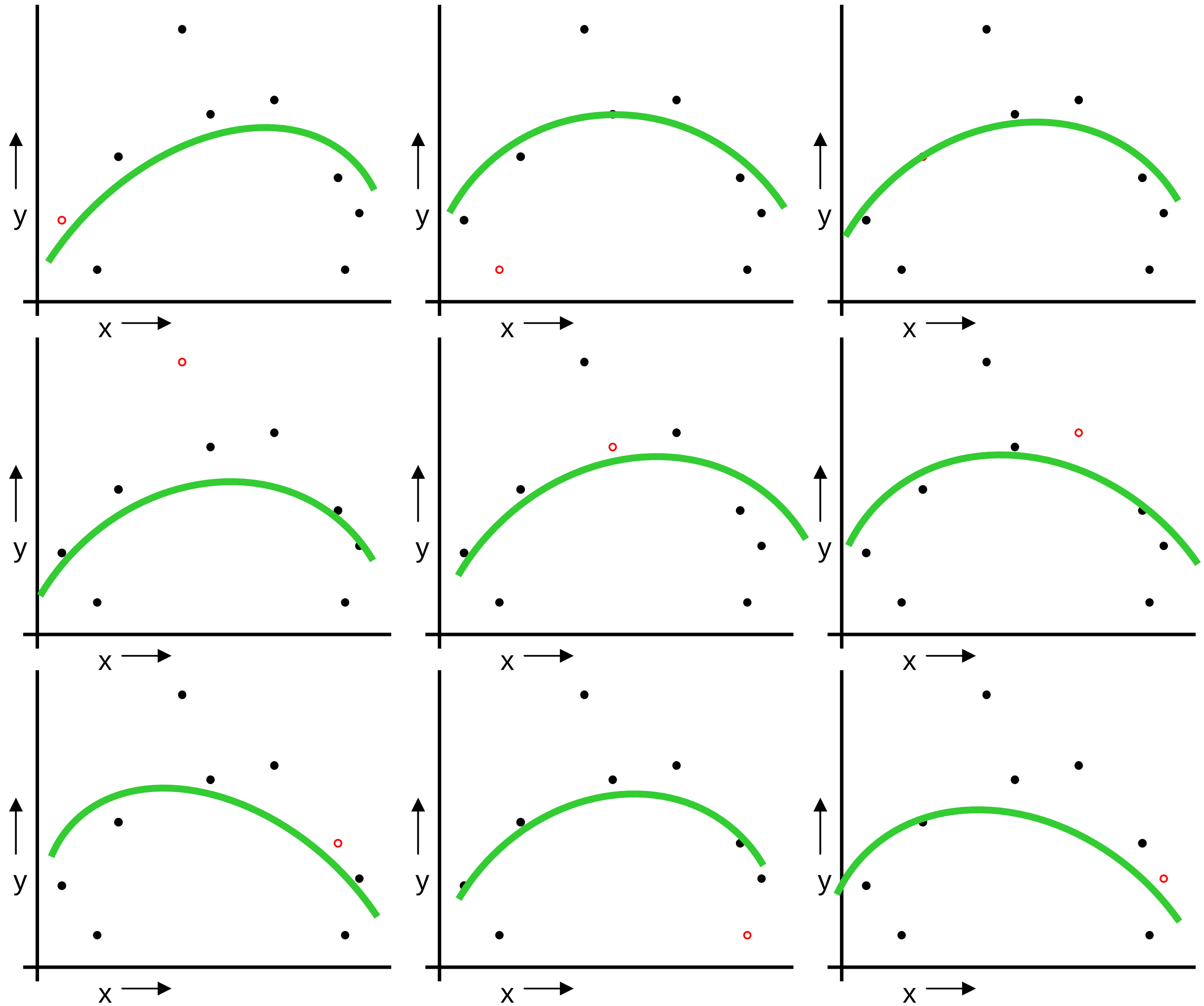
For k=1 to R

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



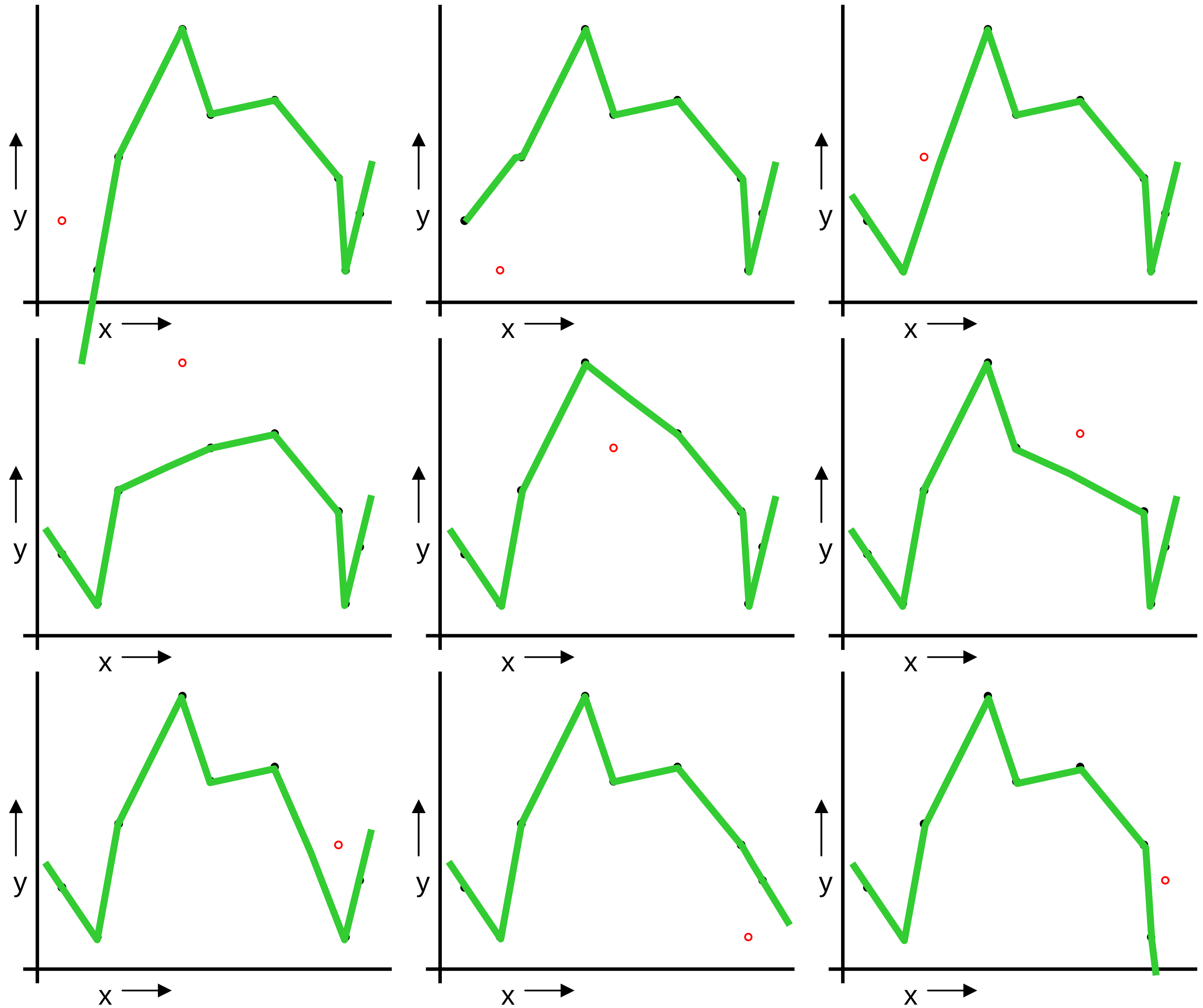
For k=1 to R

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.

$MSE_{LOOCV} = 0.962$

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

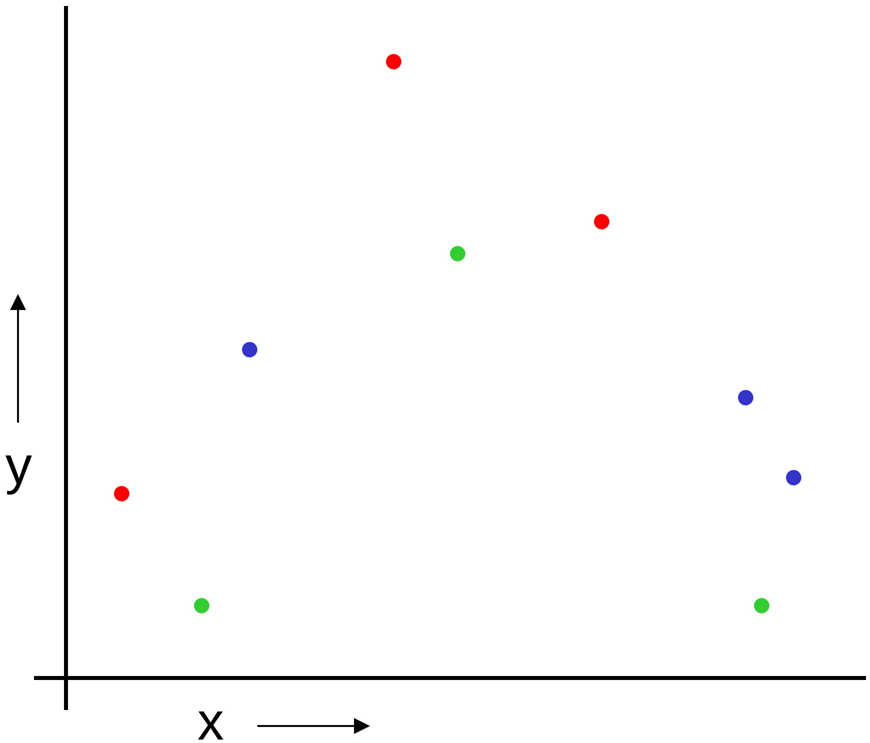
Which kind of Cross Validation?

	Downside	**Upside**
Single split	Variance: unreliable estimate of future performance-wasteful	Cheap
Leave-one-out	Expensive Has some weird behavior	Doesn't waste data

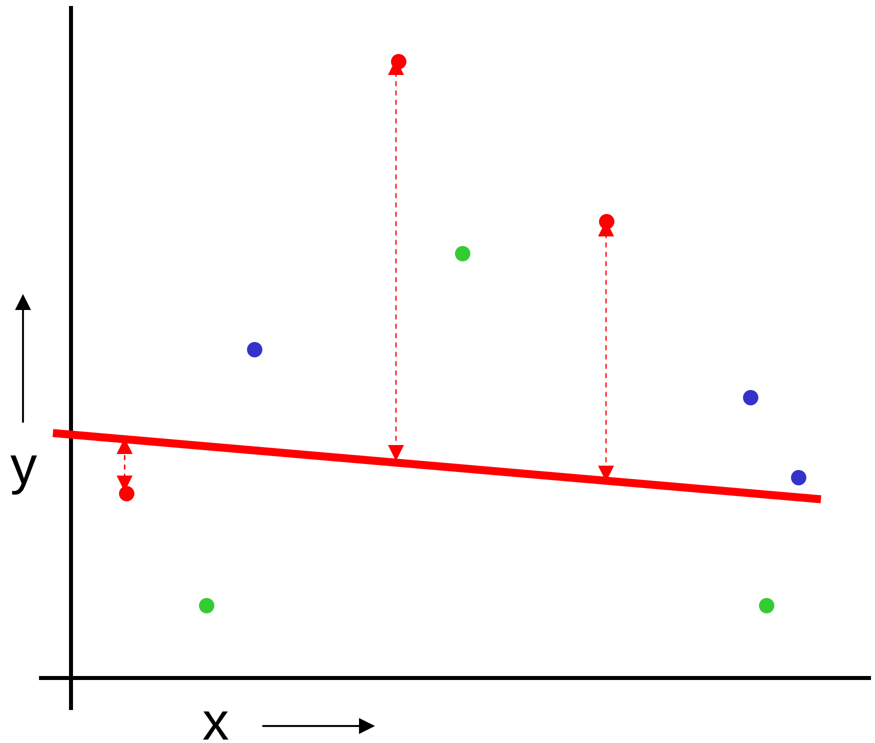
..can we get the best of both worlds?

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)



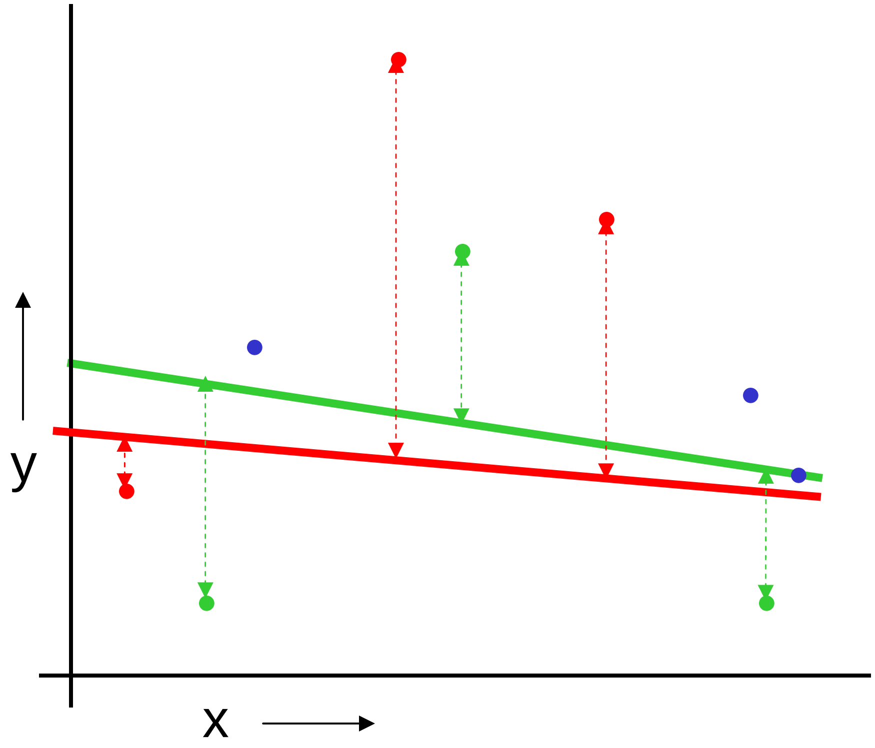
k-fold Cross Validation



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

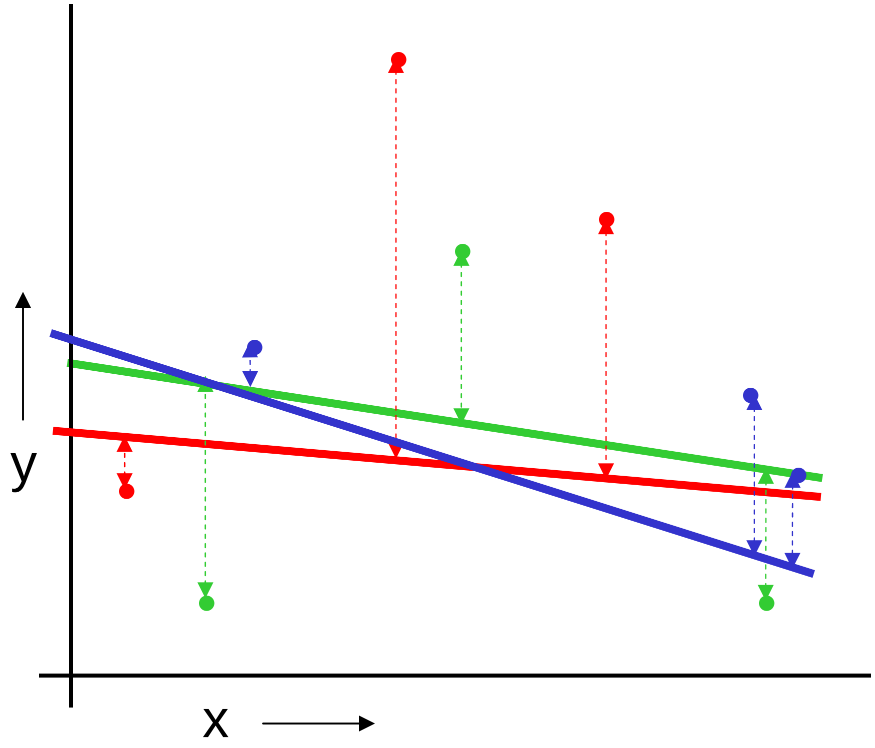


Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

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



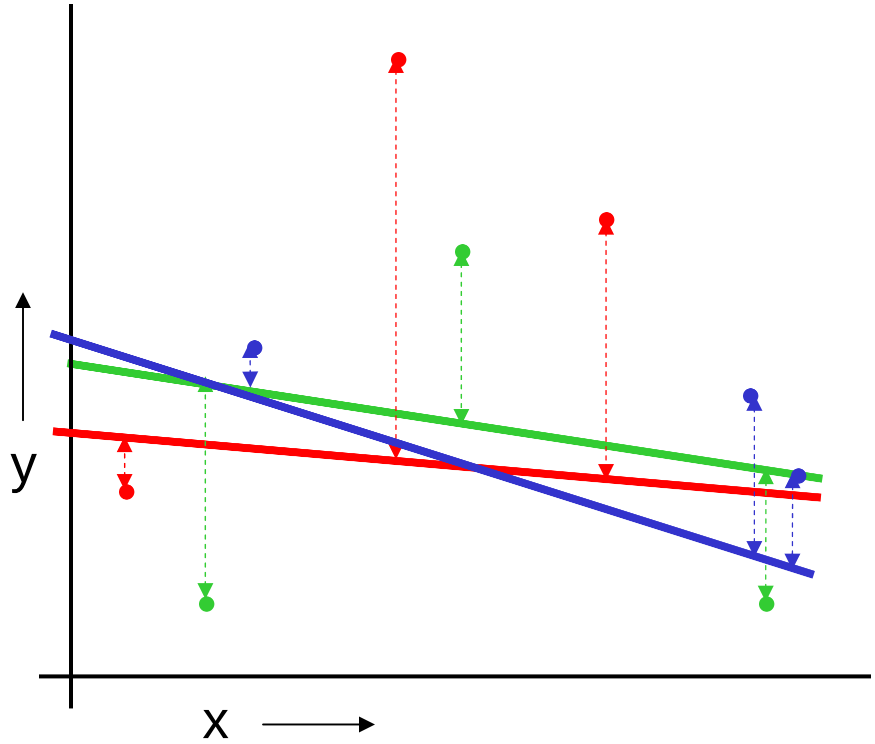
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

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



Linear Regression
MSE_{3FOLD}=2.05

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

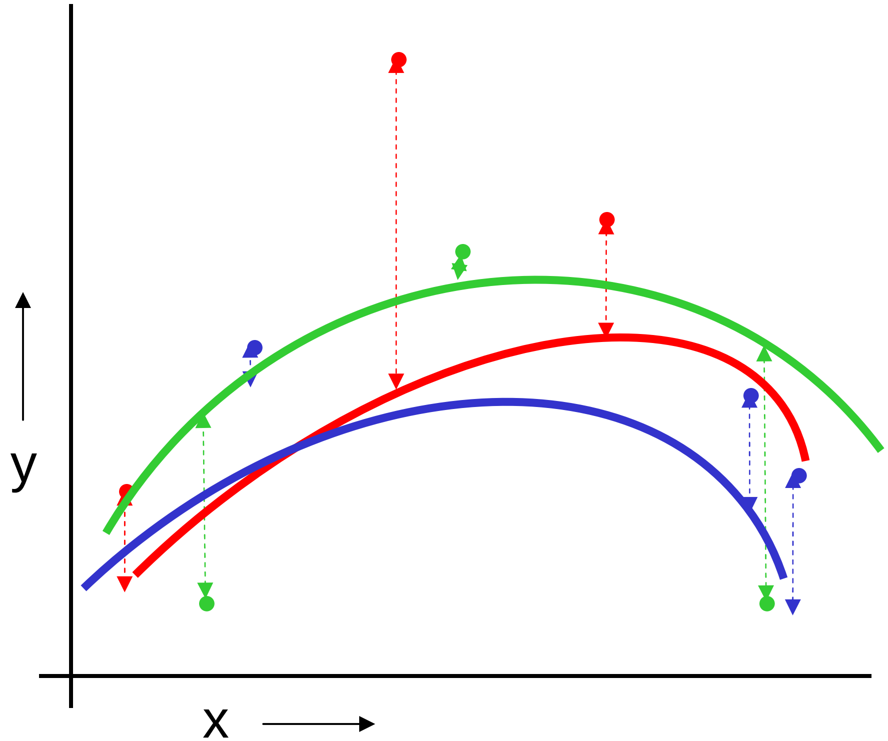
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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

k-fold Cross Validation



Quadratic Regression
$MSE_{3FOLD}=1.11$

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

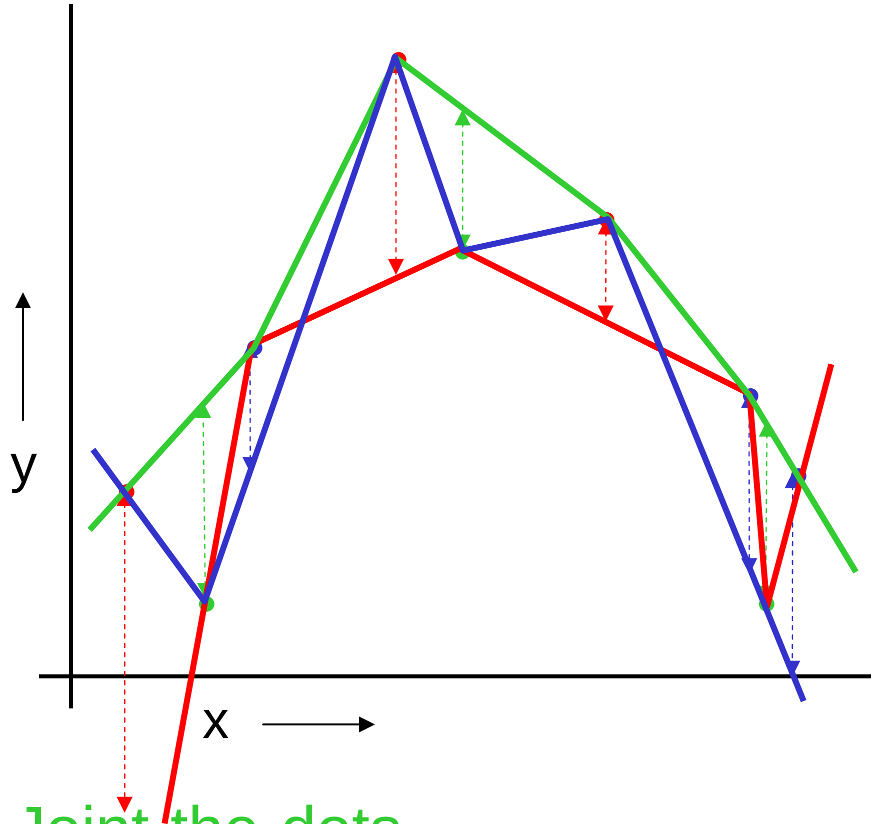
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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

k-fold Cross Validation



Joint-the-dots

$MSE_{3FOLD}=2.93$

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 validation-set sum of errors on the red points.

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

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

Then report the mean error

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	10 times more expensive instead of R times.
3-fold	Wastier than 10-fold. Expensivier than testset	Slightly better than test-set
R-fold	Identical to Leave-one-out	

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			

CV-based Model Selection

- Example: Choosing number of hidden units in a one-hidden-layer neural net.
- Step 1: Compute 10-fold CV error for six different model classes:

Algorithm	TRAINERR	10-FOLD-CV-ERR	Choice
0 hidden units			
1 hidden units			
2 hidden units			⌦
3 hidden units			
4 hidden units			
5 hidden units			

- Step 2: Whichever model class gave best CV score: train it with all the data, and that's the predictive model you'll use.

CV-based Model Selection

- Example: Choosing “k” for a k-nearest-neighbor regression.
- Step 1: Compute LOOCV error for six different model classes:

Algorithm	TRAINERR	10-fold-CV-ERR	Choice
K=1			
K=2			
K=3			
K=4			⌧
K=5			
K=6			

- Step 2: Whichever model class gave best CV score: train it with all the data, and that’s the predictive model you’ll use.

CV-based Algorithm Choice

- Example: Choosing which regression algorithm to use
- Step 1: Compute 10-fold-CV error for six different model classes:

Algorithm	TRAINERR	10-fold-CV-ERR	Choice
1-NN			
10-NN			
Linear Reg'n			
Quad reg'n			⌦
LWR, KW=0.1			
LWR, KW=0.5			

- Step 2: Whichever algorithm gave best CV score: train it with all the data, and that's the predictive model you'll use.