

Validation and cross validation

- Validation addresses the problem of **over-fitting**.
- Internal Validation: Validate your model on your current data set (cross-validation)
- External Validation: Validate your model on a completely new dataset

Holdout validation

- One way to validate your model is to fit your model on half your dataset (your “training set”) and test it on the remaining half of your dataset (your “test set”).
- If over-fitting is present, the model will perform well in your training dataset but poorly in your test dataset.
- Of course, you “waste” half your data this way, and often you don’t have enough data to spare...

When is cross-validation used?

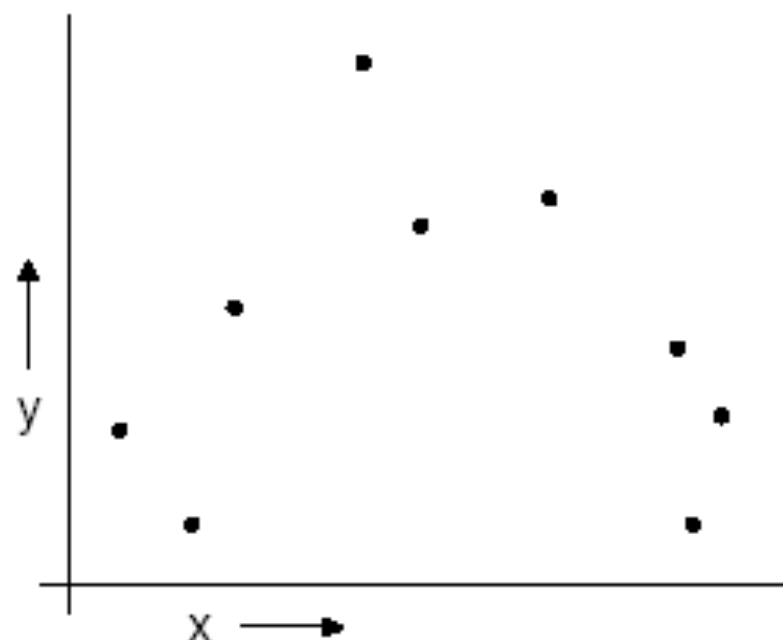
- Anytime you want to prove that your model is not over-fitting, that it will have good prediction in new datasets.

10-fold cross-validation (one example of K-fold cross-validation)

1. Randomly divide your data into 10 pieces, 1 through k.
2. Treat the 1st tenth of the data as the test dataset. Fit the model to the other nine-tenths of the data (which are now the training data).
3. Apply the model to the test data and calculate error etc.
4. Repeat this procedure for all 10 tenths of the data.
5. Calculate statistics of model accuracy and fit from the test data only.

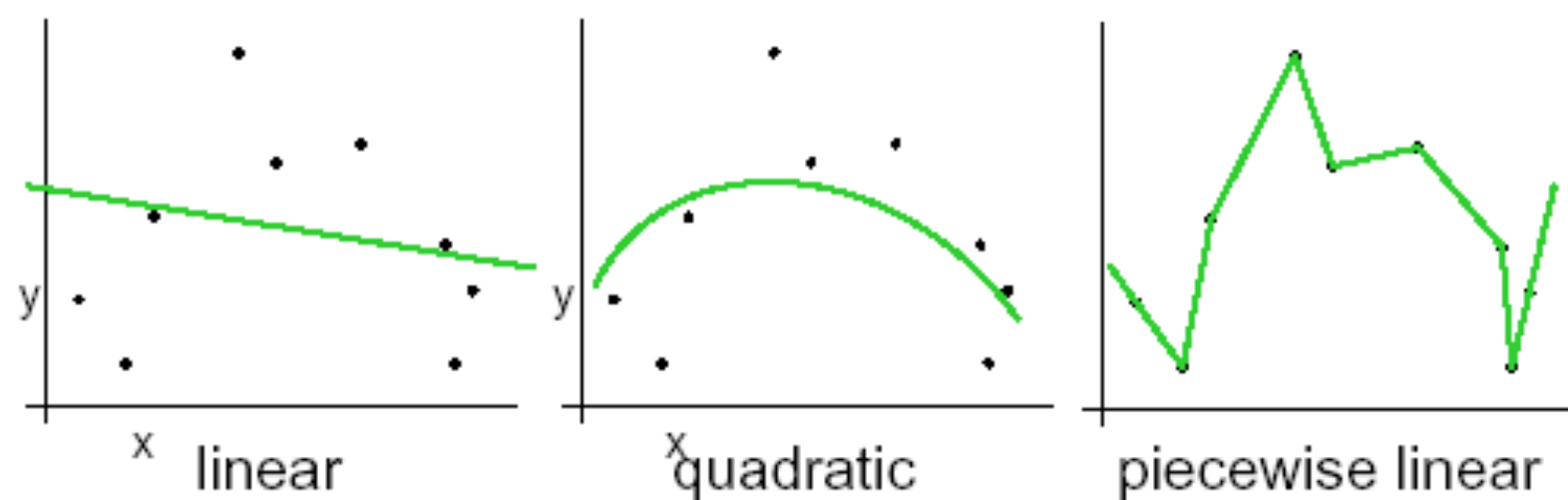
Want to learn model from data

Toy example



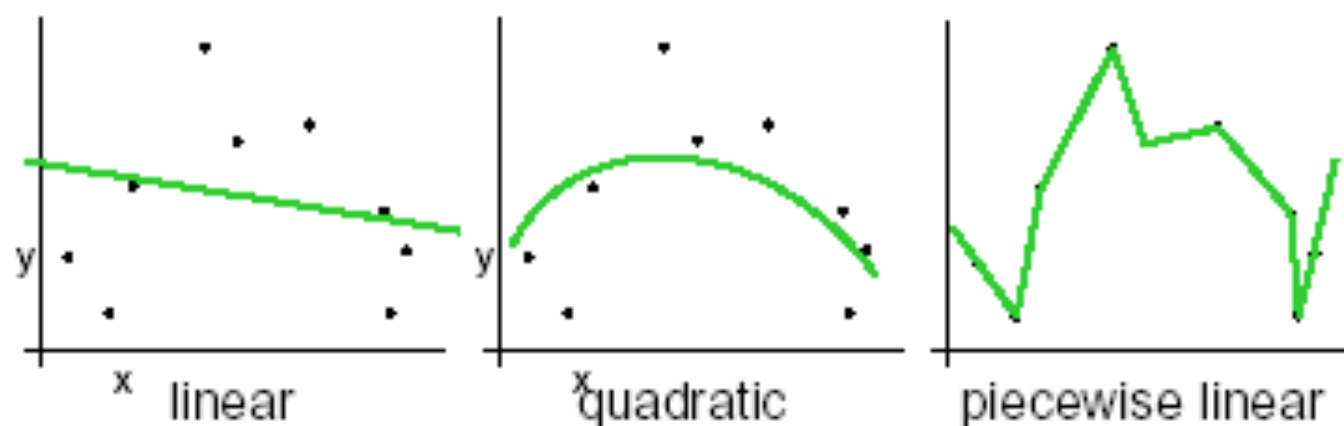
Can we learn $f(x)$ that fits our data, $y=f(x)+\text{noise}$?

Which is best?



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

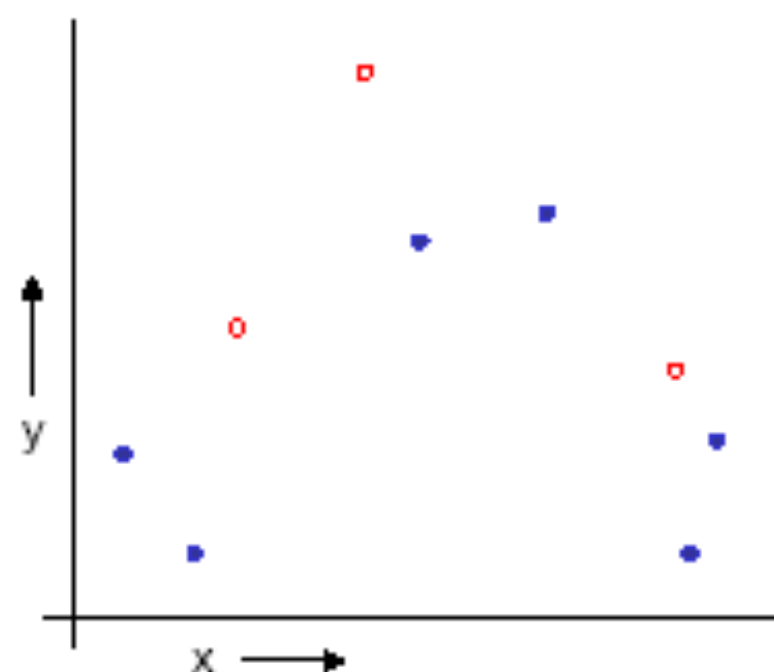
What do we really want?



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

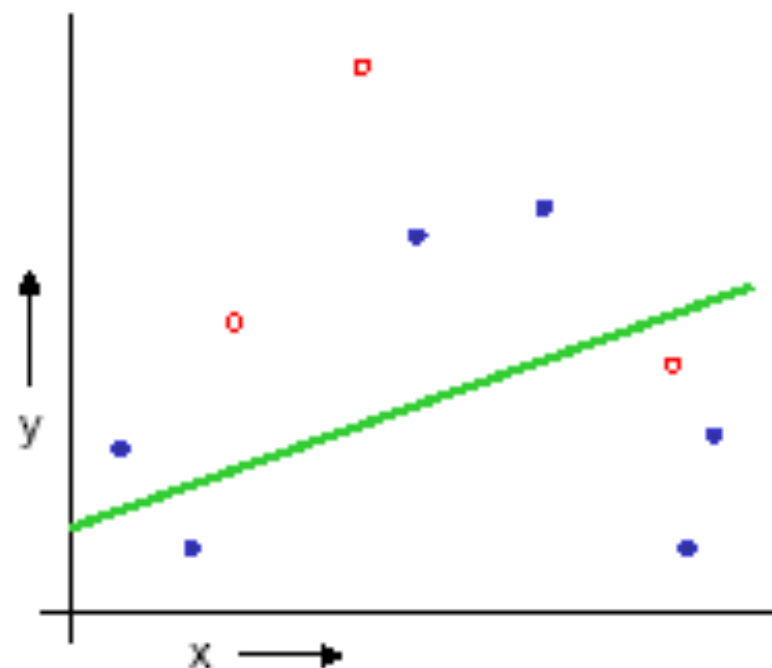
"How well are you going to predict future data drawn from the same distribution?"

The test set method



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

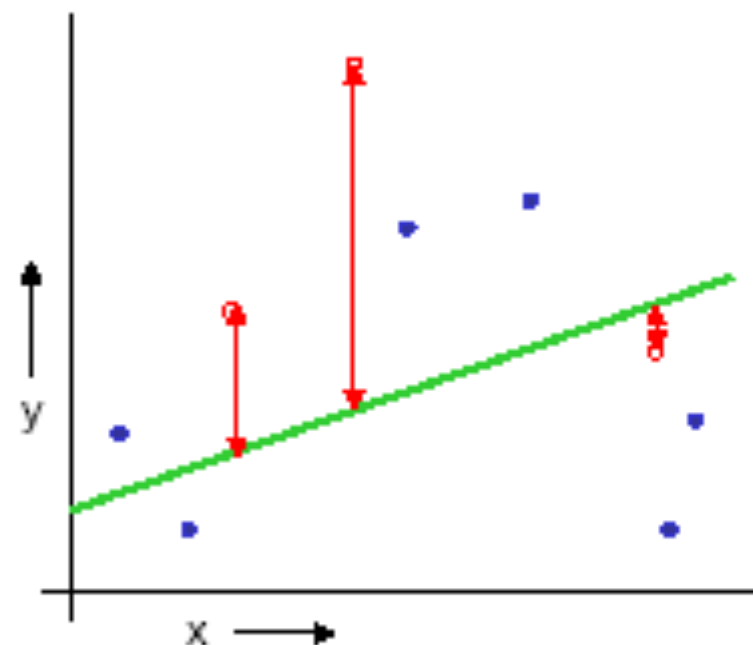
The test set method



(Linear function example)

1. Randomly choose 30% of the data to be in a **test set**
2. The remainder is a **training set**
3. Learn model from the training set

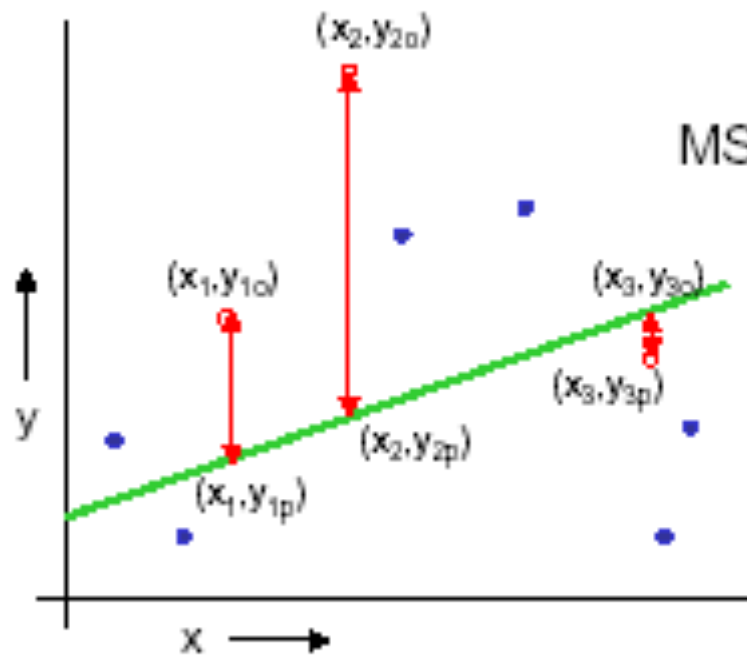
The test set method



1. Randomly choose 30% of the data to be in a **test set**
2. The remainder is a **training set**
3. Learn the model from the training set
4. Estimate your future performance with the test set

How to tell how good is the prediction?

MSE (Mean Squared Error)



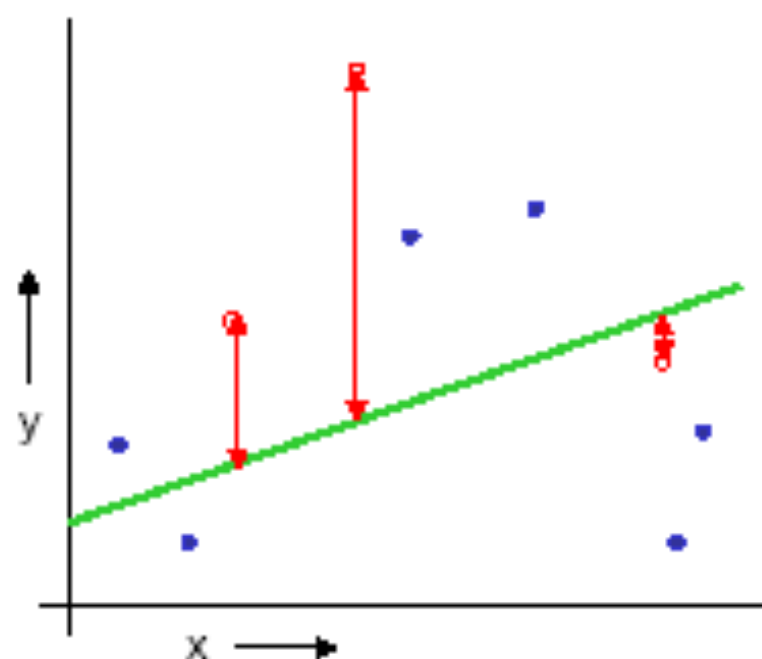
$$\text{MSE} = \frac{(y_{1p} - y_{1o})^2 + (y_{2p} - y_{2o})^2 + (y_{3p} - y_{3o})^2}{3}$$

In general,

$$\text{MSE} = \frac{\sum_{i=1}^n (\text{prediction}_i - \text{observation}_i)^2}{n}$$

(Linear function example)

The test set method

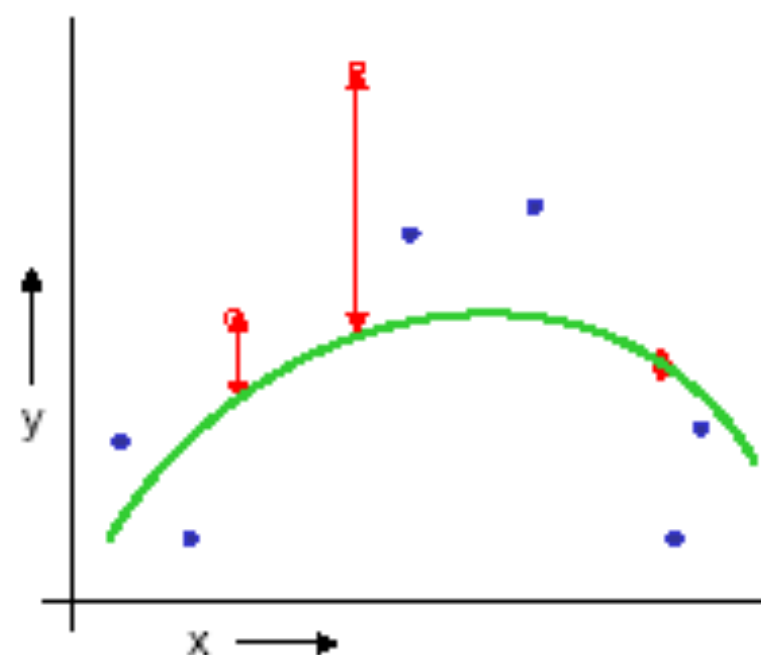


(Linear function example)

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. Learn the model from the training set
4. Estimate your future performance with the test set

The test set method

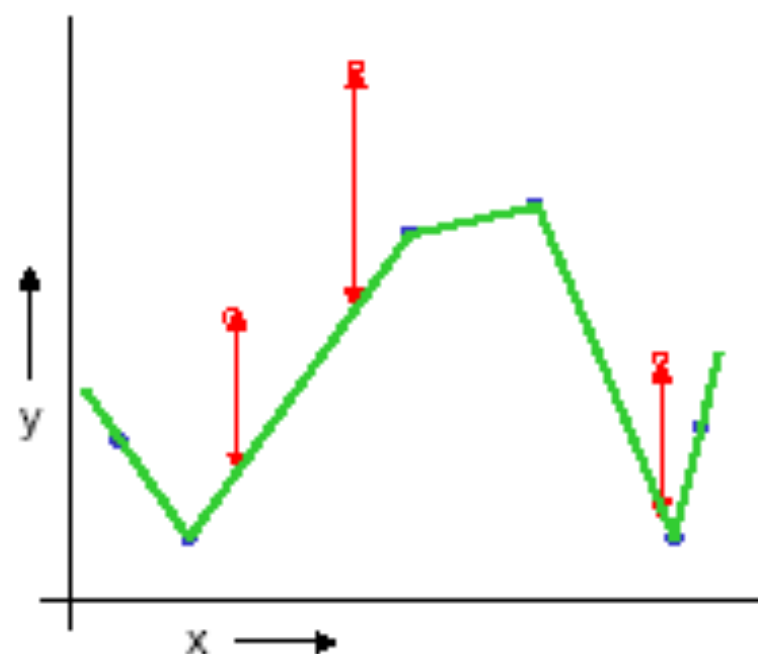


(Quadratic function)

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 model fitting on the training set
4. **Estimate your future performance with the test set**

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 model fitting on the training set
4. **Estimate your future performance with the test set**

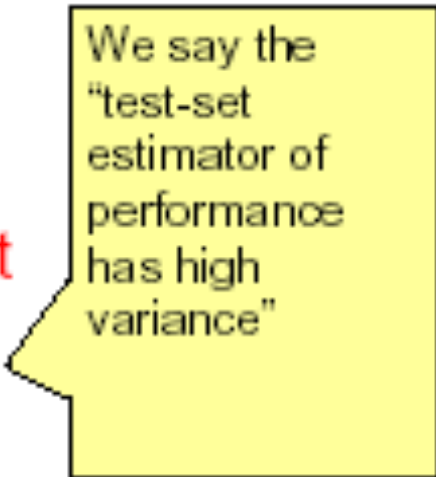
The test set method

Good news:

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

Bad news:

- Wastes data: best model is estimated using 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"

The holdout method

- The **holdout method** is the simplest kind of cross validation.
- The data set is separated into two sets, called the training set and the testing set.
- The function approximator fits a function using the training set only.
- Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before).

The holdout method

- The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model.
- The advantage of this method is that it is usually preferable to the residual method and takes no longer to compute.
- However, its evaluation can have a high variance.
- The evaluation may depend heavily on which data points end up in the training set and which end up in the test set, the evaluation may be significantly different depending on how the division is made.

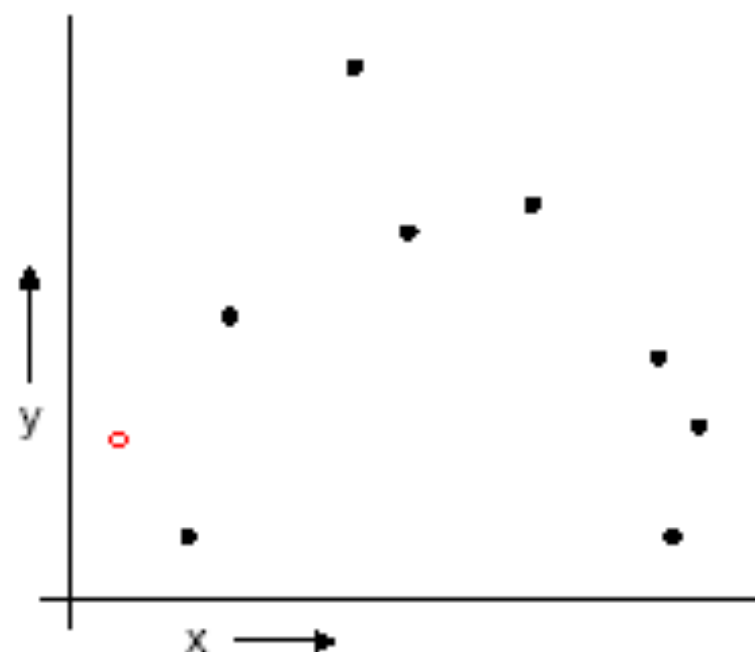
Leave-one-out cross validation

- **Leave-one-out cross validation (LOOCV)** is **K-fold cross validation** taken to its logical extreme, with K equal to N , the number of data points in the set.
- That means that N separate times, **the function approximator is trained on all the data except for one point** and **a prediction is made for that point**.
- As before the average error is computed and used to evaluate the model.

LOOCV (Leave-one-out Cross Validation)

For $k=1$ to R , $R = \#$ of records

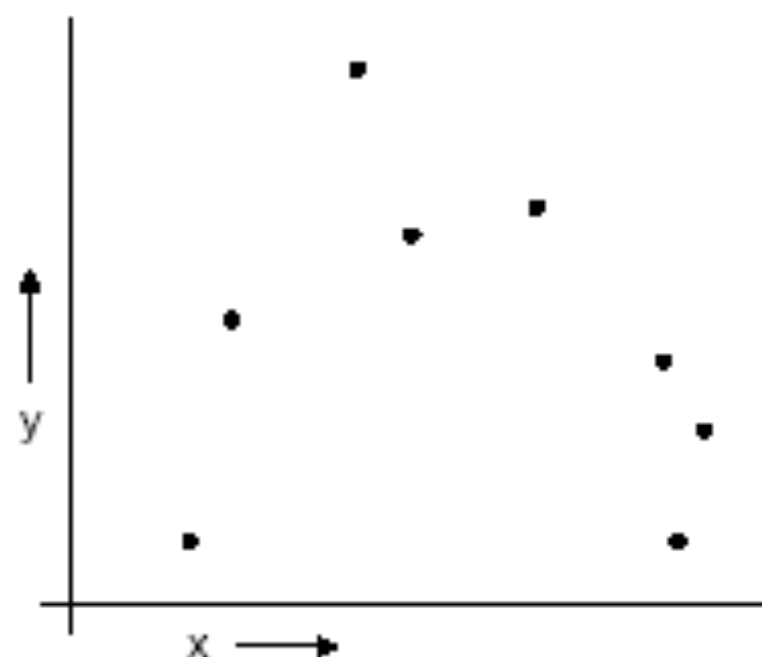
1. Let (x_k, y_k) be the k^{th} record



LOOCV (Leave-one-out Cross Validation)

For $k=1$ to R

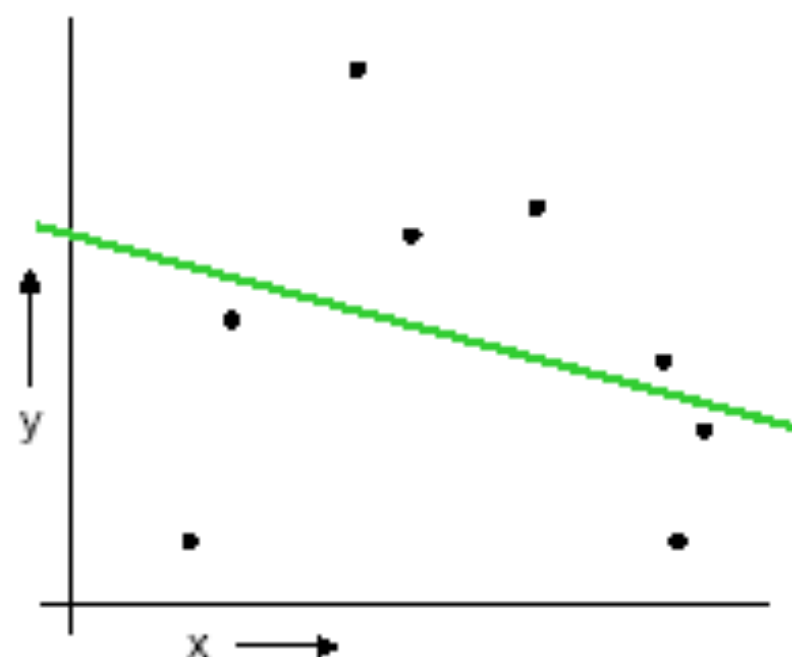
1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset



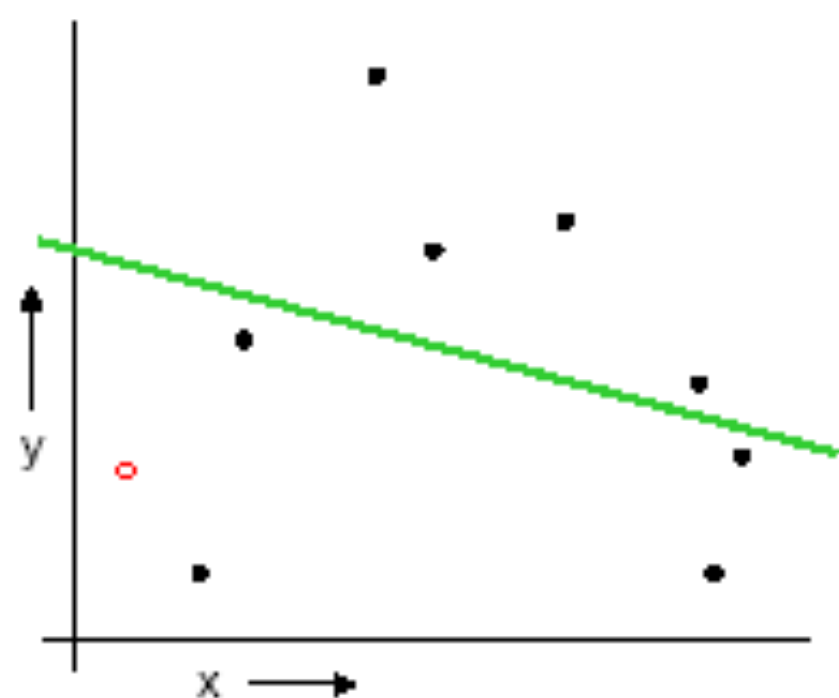
LOOCV (Leave-one-out Cross Validation)

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)

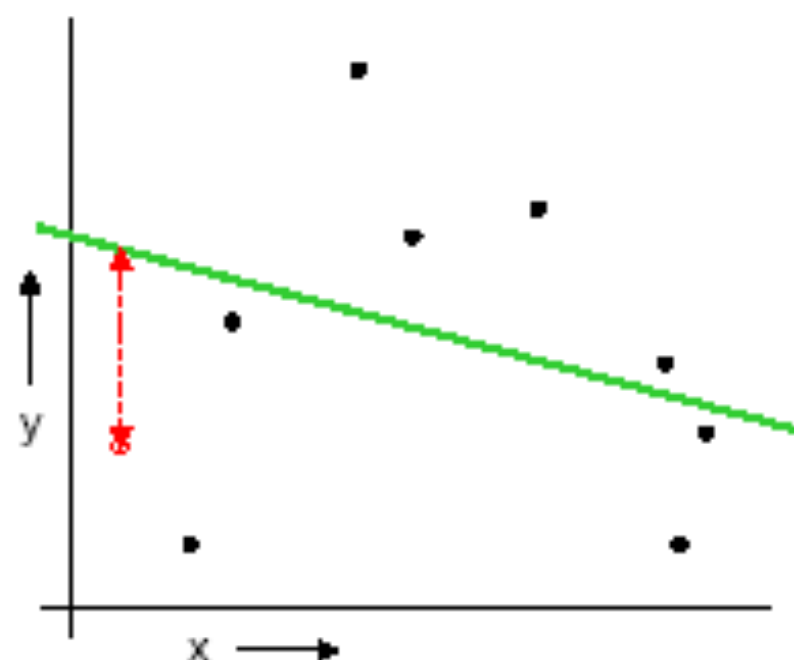


For $k=1$ to R

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)

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

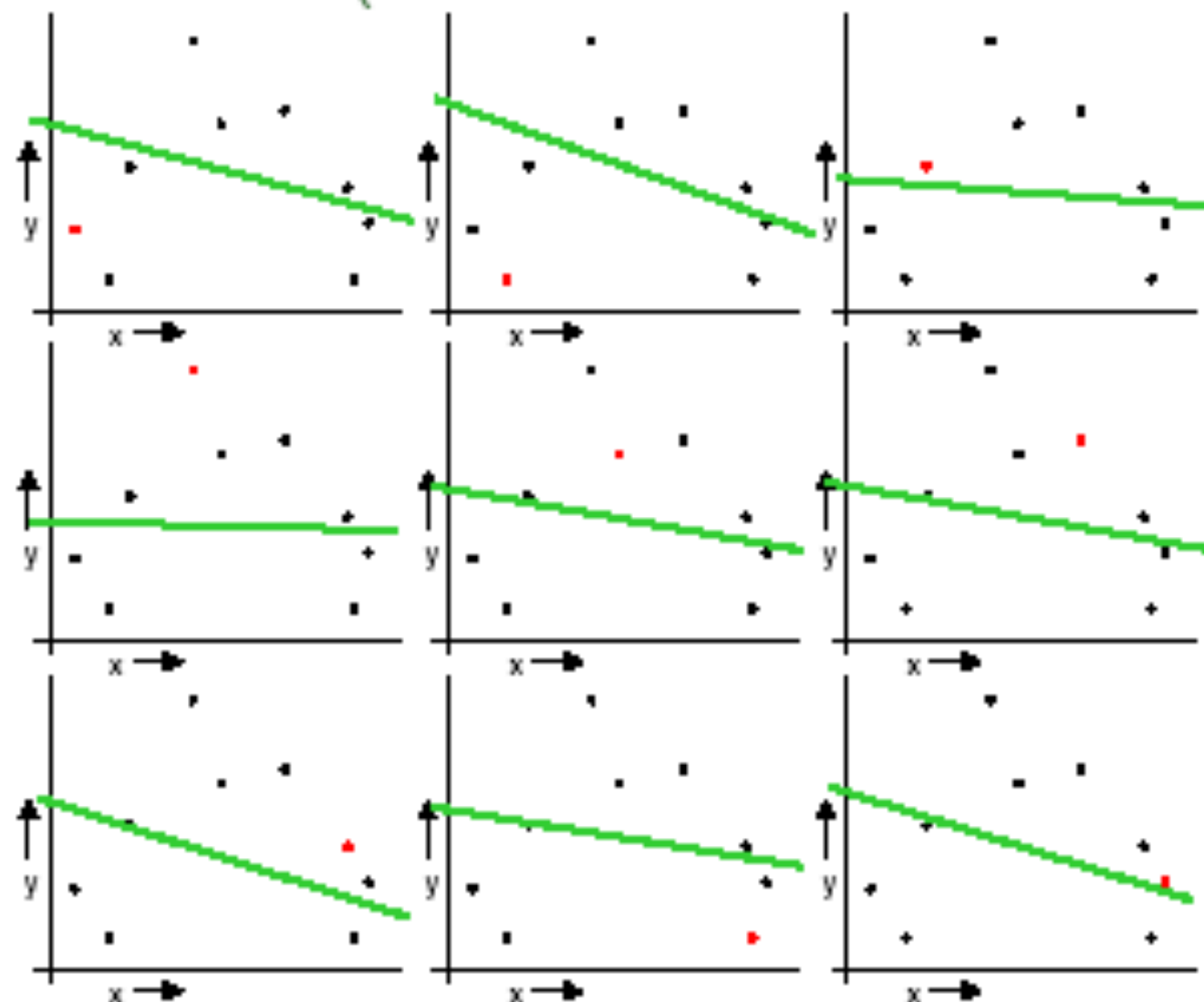
LOOCV (Leave-one-out Cross Validation)



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)

LOOCV (Leave-one-out Cross Validation)



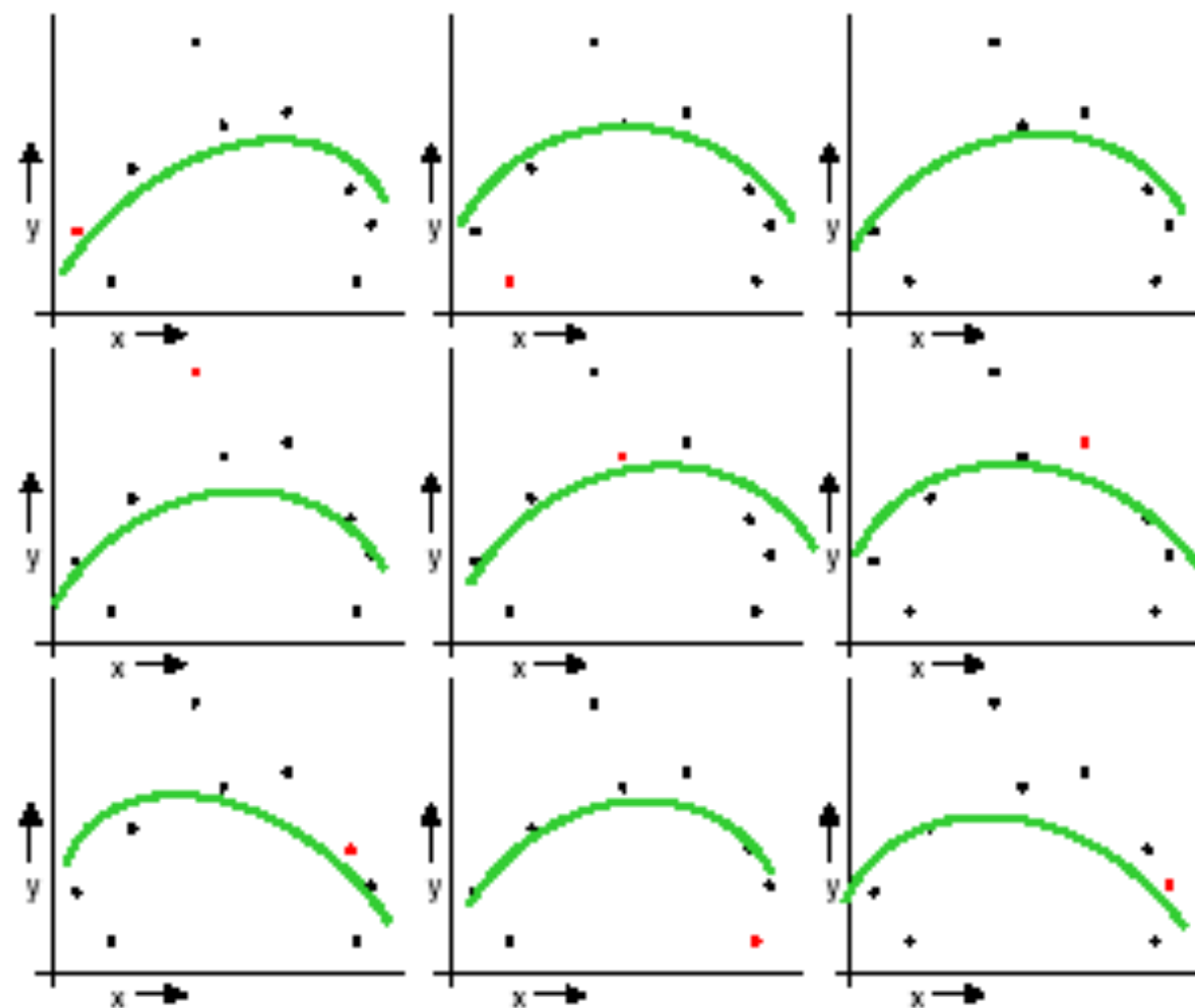
For $k=1$ to R

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 Function



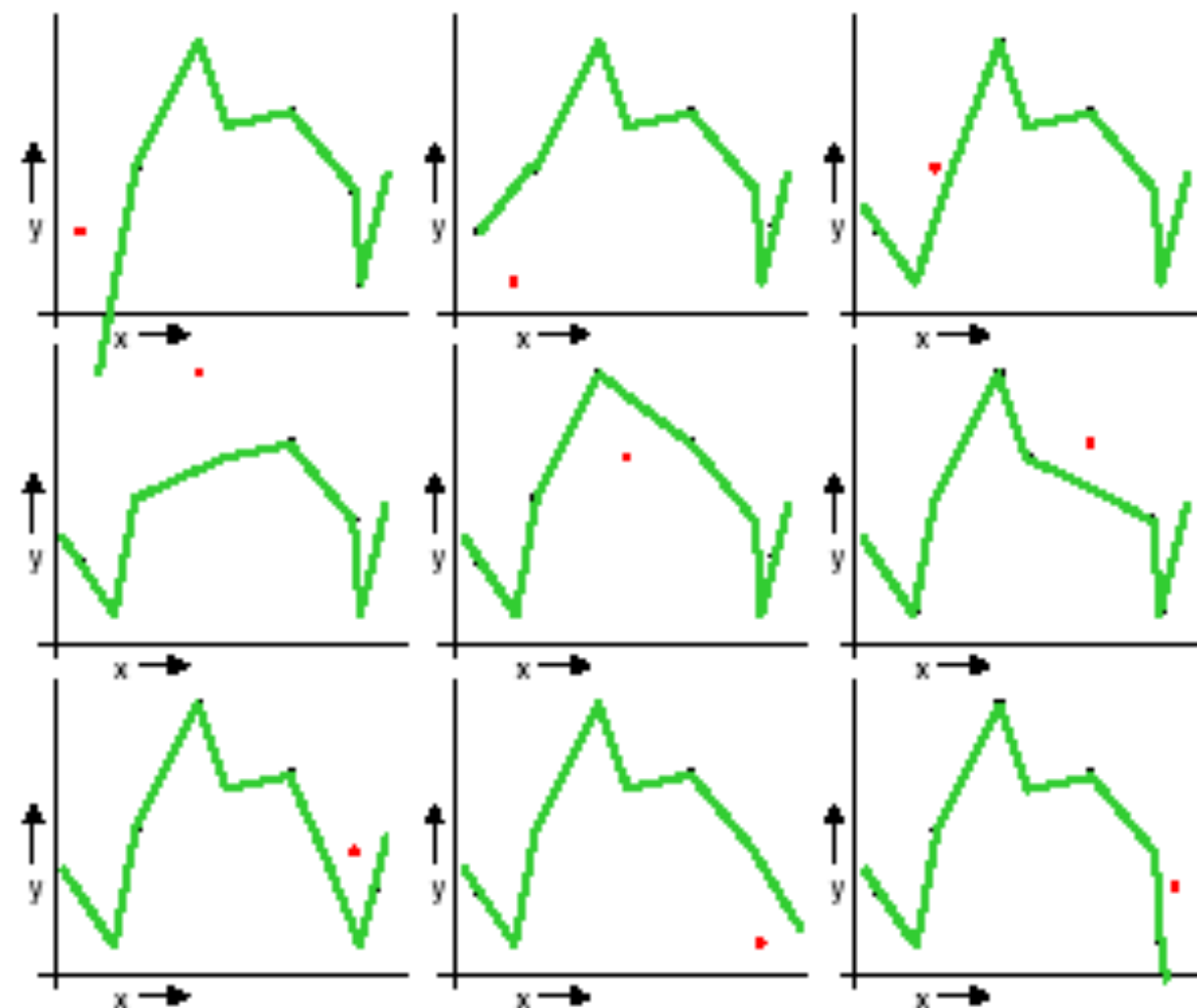
For $k=1$ to R

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.

$$MSE_{LOOCV} = 3.33$$

Leave-one-out cross validation

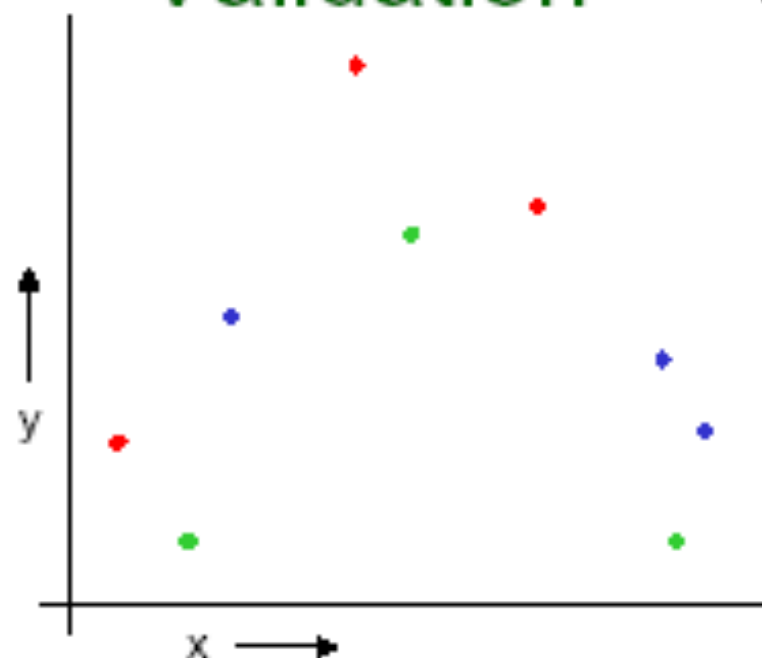
- The evaluation given by *leave-one-out cross validation error (LOO-CVE)* is good, but at first pass it seems *very expensive to compute*.
- But for some data (e.g. time series on a number of subjects), this is the best
 - Leaving out 1 point but keeping the other 99 from that subject is no good
 - So, actually do *leave-one-subject-out* cross validation

K-fold cross validation

- **K-fold cross validation** is one way to improve over the holdout method.
- The data set is divided into k subsets, and the holdout method is repeated k times.
- Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set.
- Then the average error across all k trials is computed.
- The advantage of this method is that it matters less how the data gets divided.
- Every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times.

k-fold Cross Validation

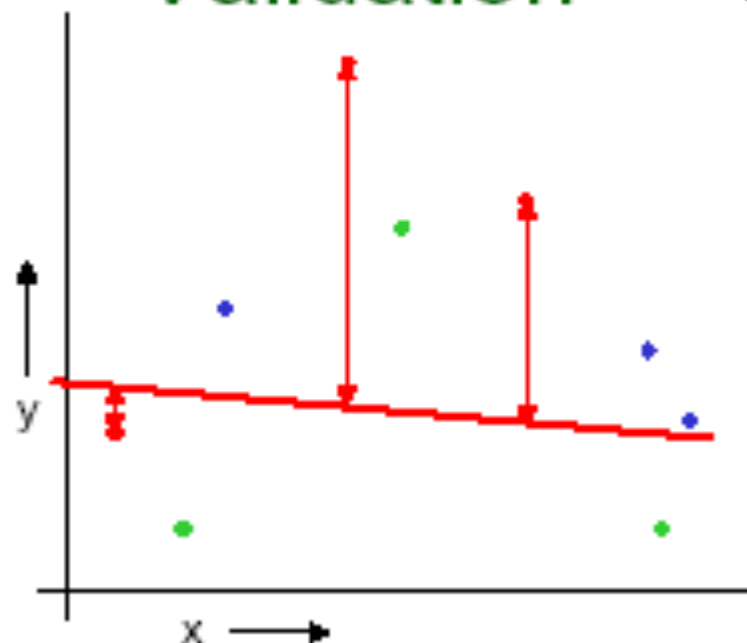
Randomly break the dataset into k partitions (in our example we'll have $k=3$ partitions colored Red Green and Blue)



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.

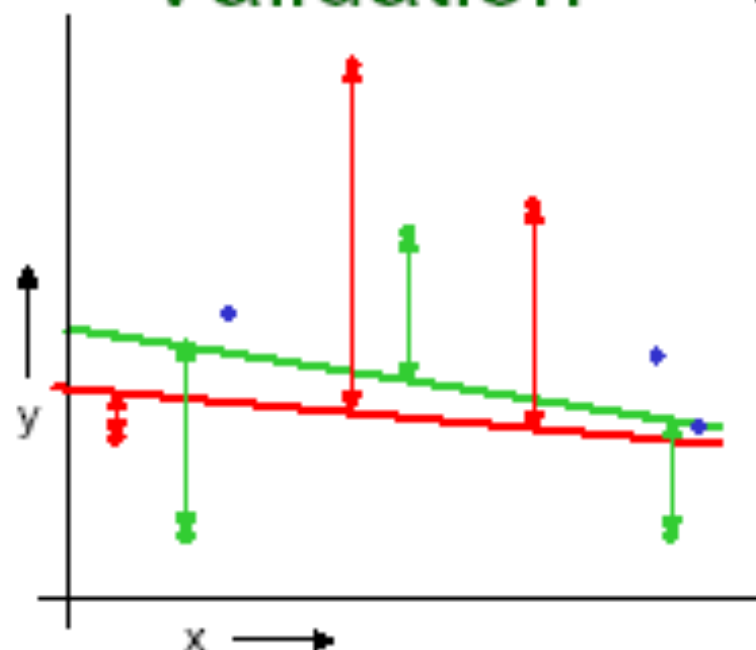


k-fold Cross Validation

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

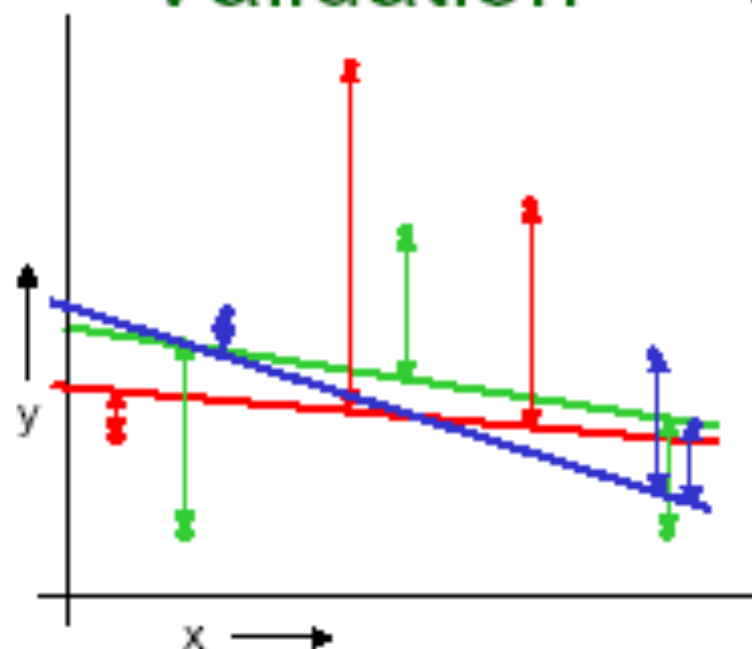
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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)



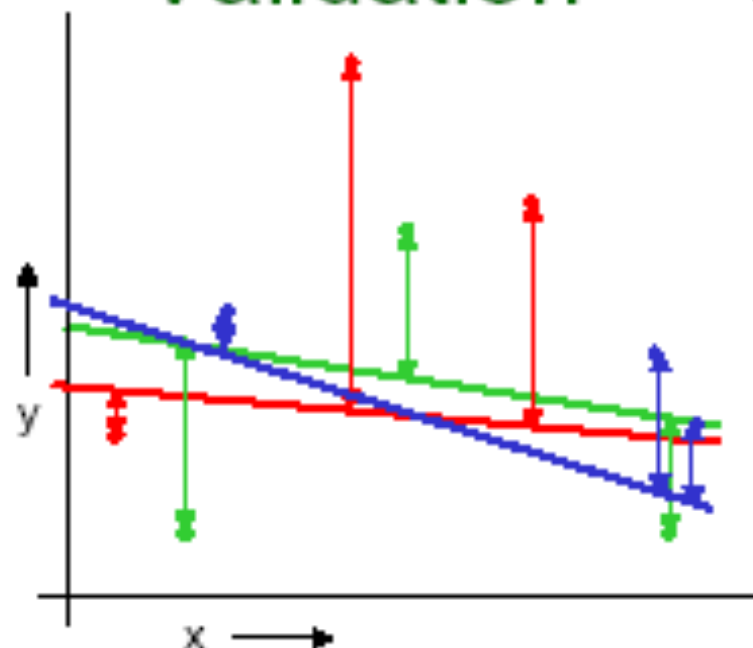
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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)



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.

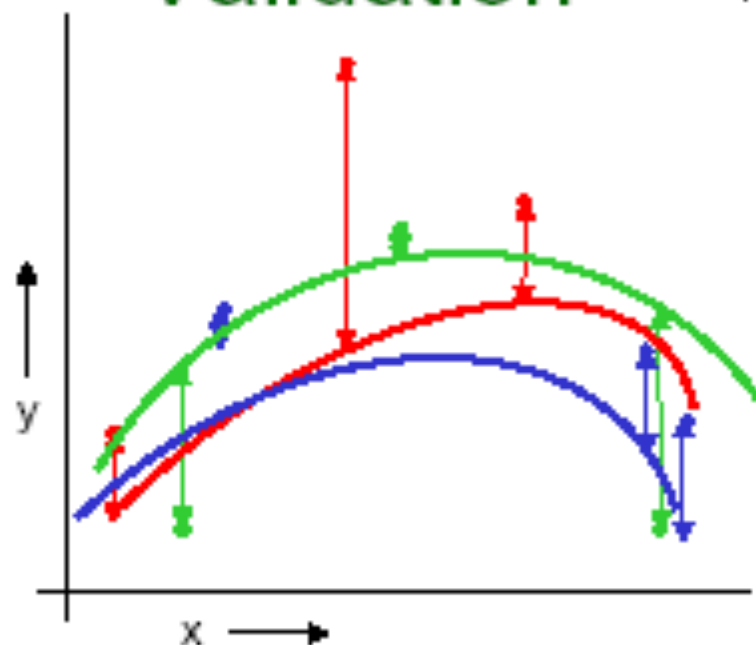
For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Linear model
 $MSE_{3FOLD} = 2.05$

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.

Quadratic model
 $MSE_{3FOLD}=1.11$

Then report the mean error

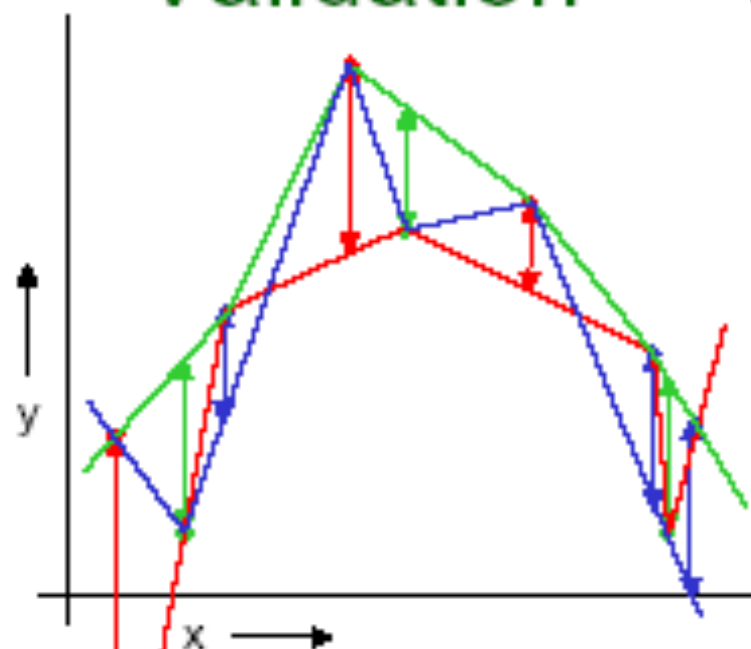
k-fold Cross Validation

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.



Piecewise linear model
 $MSE_{3FOLD} = 2.93$














Then report the mean error

Cross-Validation is useful

- Preventing overfitting
- Comparing different learning algorithms
- Choosing the number of hidden units in a neural net
- Feature selection (see later)
- Choosing a polynomial degree

CV-based Model Selection

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

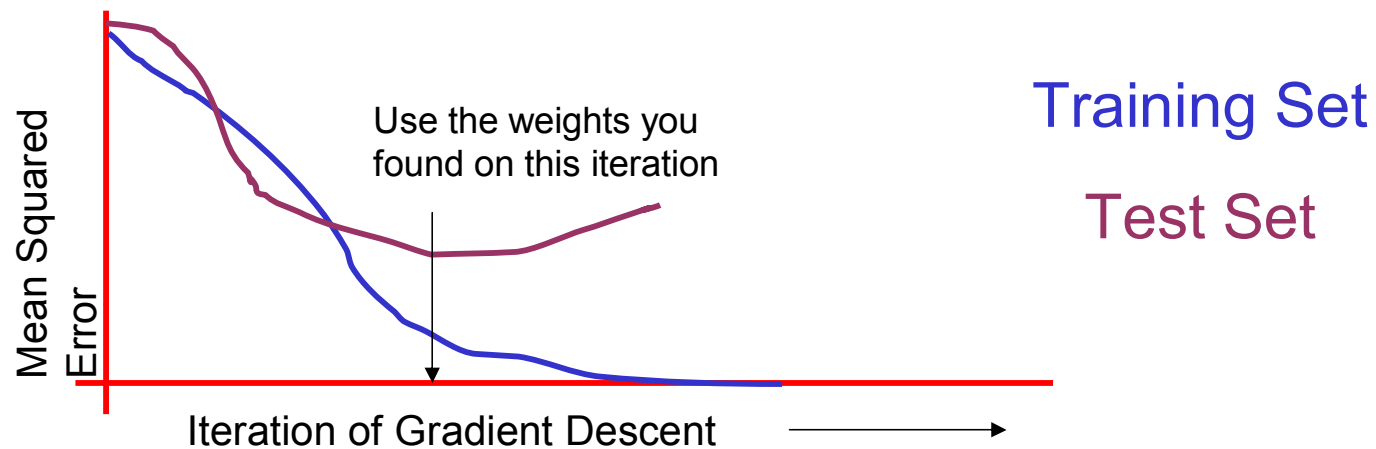
i	f_i	TRAINERR	10-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			
4	f_4			
5	f_5			
6	f_6			

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	More waste than 10-fold. " expensive than test set	Slightly better than test-set
R-fold	Identical to Leave-one-out	

Supervising Gradient Descent

- This is a common use of Test-set validation
- Suppose you have a neural net with too many hidden units. It will overfit.
- As gradient descent progresses, maintain a graph of MSE-testset-error vs. Iteration

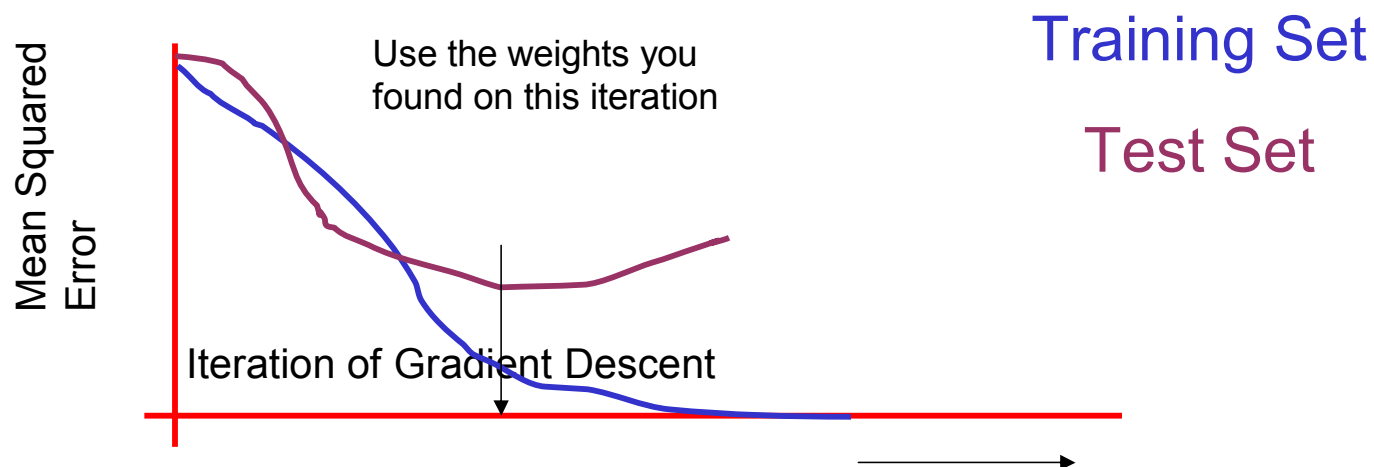


Supervising Gradient Descent

- This is a common use of Test-set validation
- Suppose you have a neural net with too many hidden units. It will overfit
- As gradient descent minimizes the error on the training set, the error on the test set will increase

Relies on an intuition that a not-fully-minimized set of weights is somewhat like having fewer parameters.

Works pretty well in practice, apparently



Model Selection vs. Validation

- Suppose we've gone through an iterative model-building process.
 - Fit several models on the **training** data
 - Tested/compared them on the **test** data
 - Selected the “best” model
- The test result of the best model might *still* be overly optimistic.
 - Why: we used the test data to select the best model.
 - Implicitly, it was used for modeling.

Validation Data

- It is therefore preferable to divide the data into three pieces:
 - **Training** Data: data used to **fit** model
 - **Test** Data: “fresh” data used to **select** model
 - **Validation** Data: data used to **evaluate** the final, selected model.
- **Train/Test** data is iteratively used for model building, model selection.
 - During this time, **Validation** data set aside and not touched.
- We can still cheat! Train 1000s of times ...
 - I only use 1 test set, prefer cross-validation!

Ten Easy Pieces

- Divide data into 10 equal pieces $P_1 \dots P_{10}$.
- Fit 10 models, each on 90% of the data.
- Each data point is treated as an out-of-sample data point by exactly one of the models.

[illegible]

Ten Easy Pieces

- Collect the scores from the red diagonal...
- ...You have an out-of-sample result based on the entire dataset.
- Even though the entire dataset was *also* used to fit the models.

[illegible]