# CS 440
# Introduction to Artificial Intelligence

Lecture 22:

Cross-Validation – Linear Regression (continued)

May 9, 2020

- **What is Learning?**
  - Use previous experience to solve problem
  - Example solutions to problems
    - Learning by demonstration
  - Examine trends in data to predict solution
    - Data mining
  - Train agent to perform task
    - Reinforcement learning

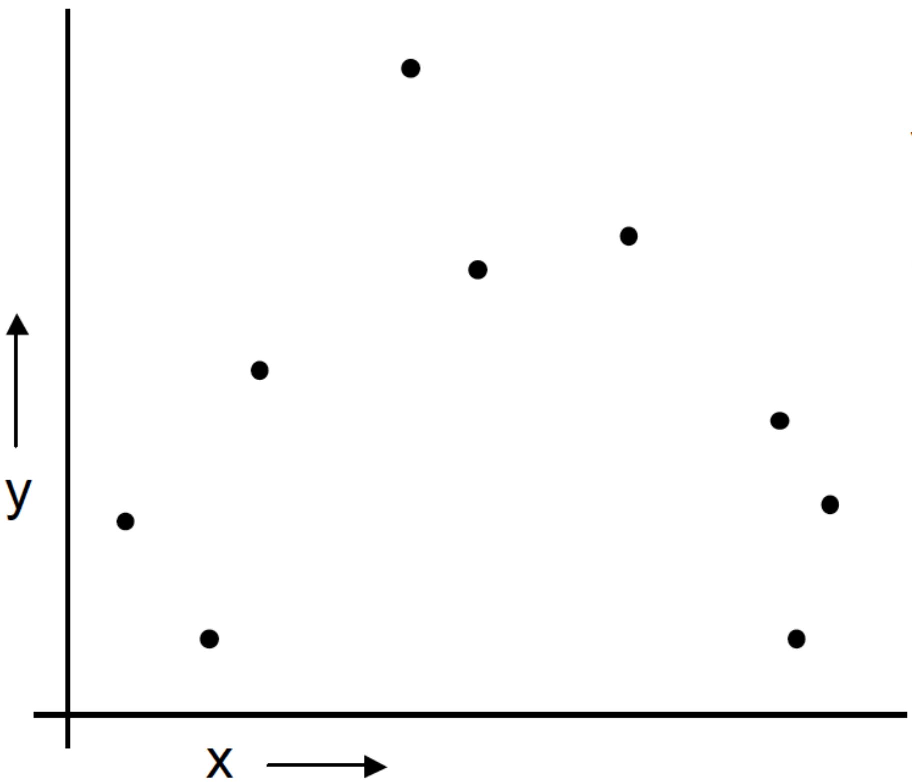
# Model

- Model
  - Representation of environment
  - Queried to find solutions to problems
- Models used in first two thirds of class
  - Examples:
    - State/Action/Transition/Reward models
    - Bayesian networks
    - Markov models
    - Logical statements
  - We defined these models
  - Agent used models we built
- What if we allowed the agent to build the models?

# Learned Model

- Allow agent to build or modify model
  - Example: Robot map its environment
    - Robot must generate some representation of its environment
    - Able to query this representation
- Models may not be intuitive to programmer
  - Mapping of inputs to solutions

# A Regression Problem

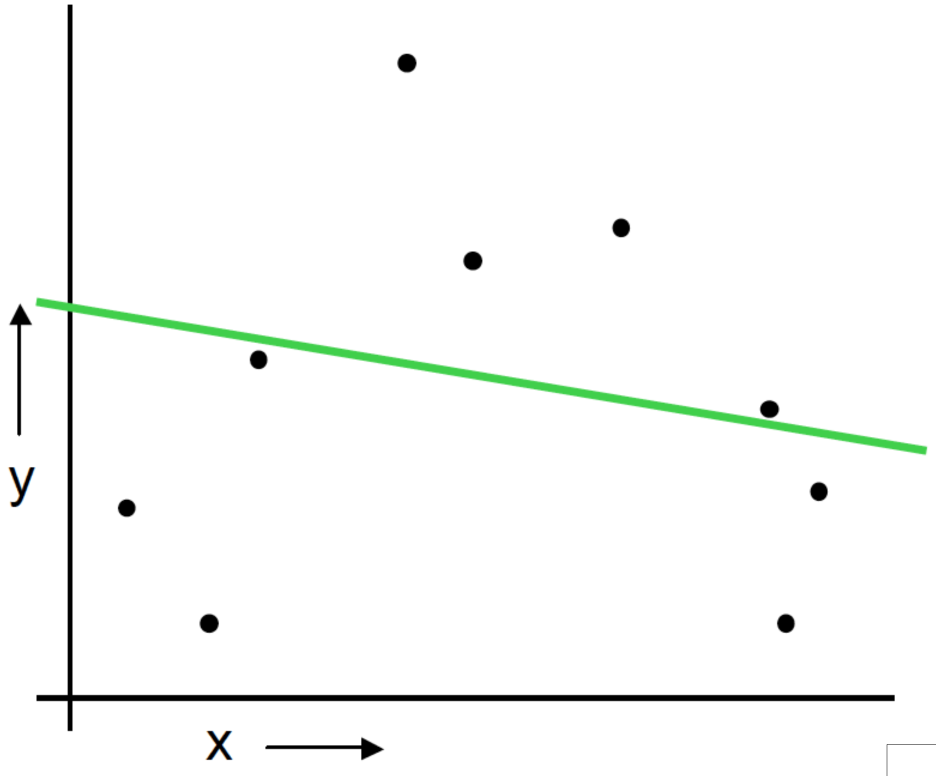



y = f(x) + noise

Can we learn f from this data?

Let's consider three methods…

# Linear Regression




$y = w_0 + w_1 \cdot x$

Objective: Minimize the

*Sum of Squared Errors*

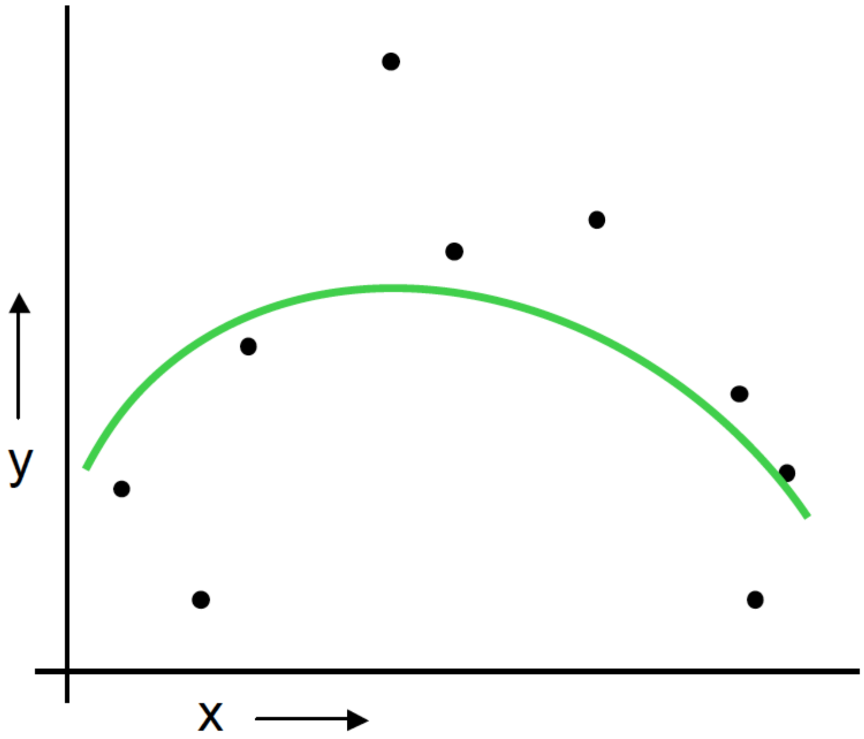
i.e. sum of squared differences

between y values

and the green line

- ŷi is the prediction of the linear model
- yi the actual value for input xi
- Then minimize: Q = Σi (ŷi – yi)2

# Quadratic Regression




Objective: Minimize the

*Sum of Squared Errors*

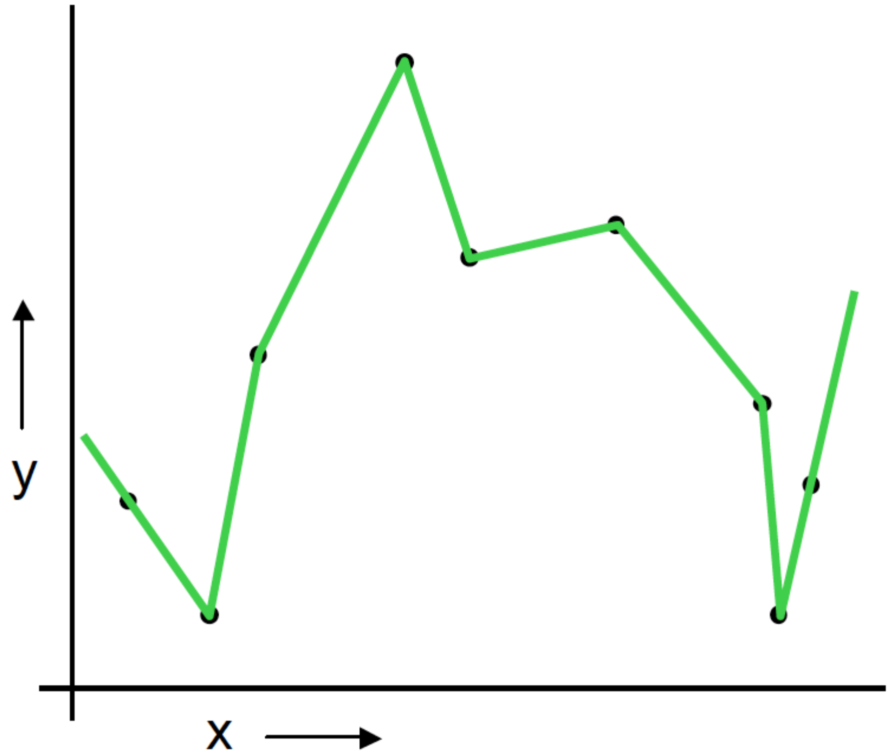
i.e. sum of squared differences

between y values

and the green curve

$$y = w_0 + w_1 \cdot x + w_2 \cdot x^2$$

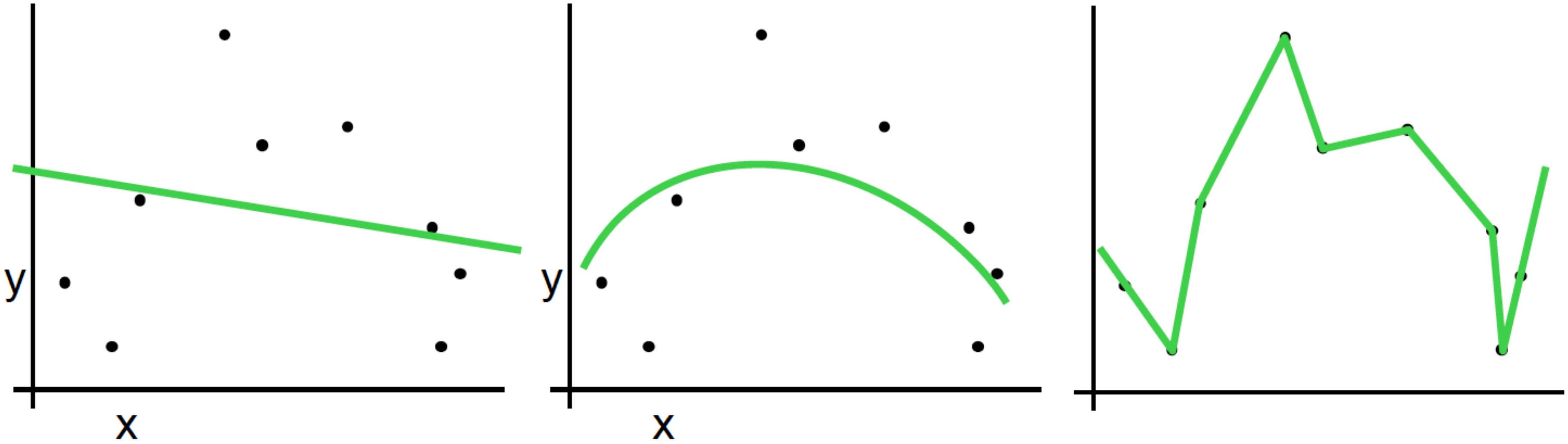
# Join-the-dots




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

y = complicated

# Which one is the best?



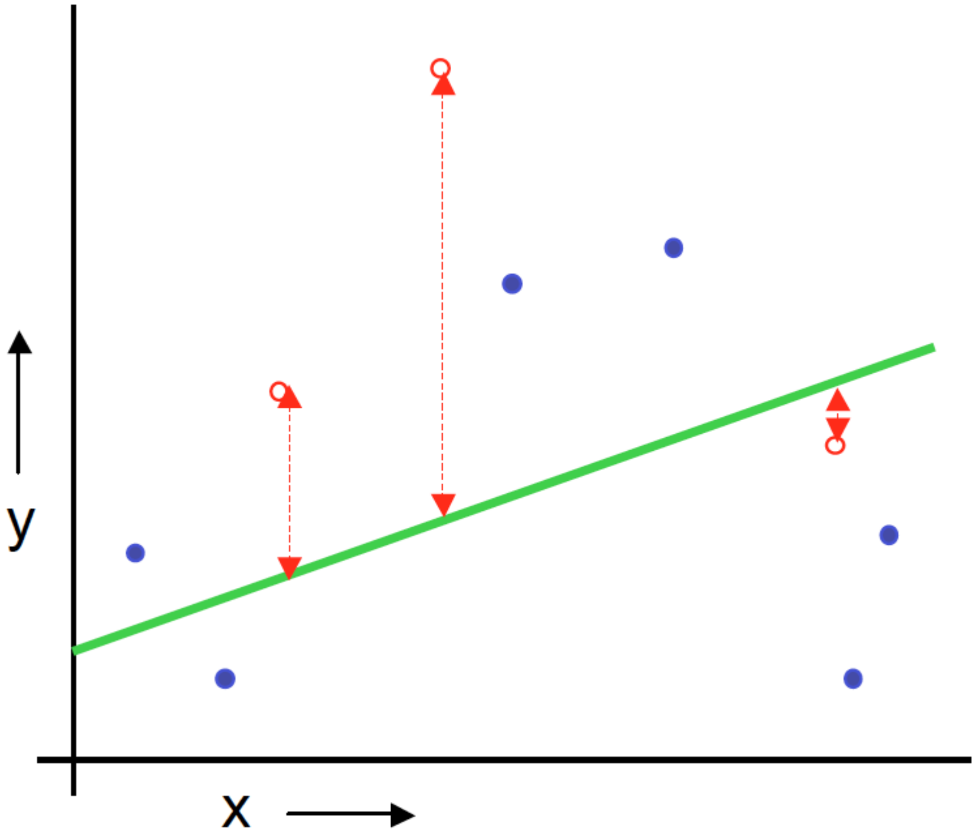

Why not choose the method with the best fit to the data?

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

# Evaluation

- How can we evaluate the fidelity of a model?
  - Minimize error function
    - Lead to over-fitting
    - More complicated model
      - More expensive to train and query
    - Performance levels off and in some cases declines

# Test Set

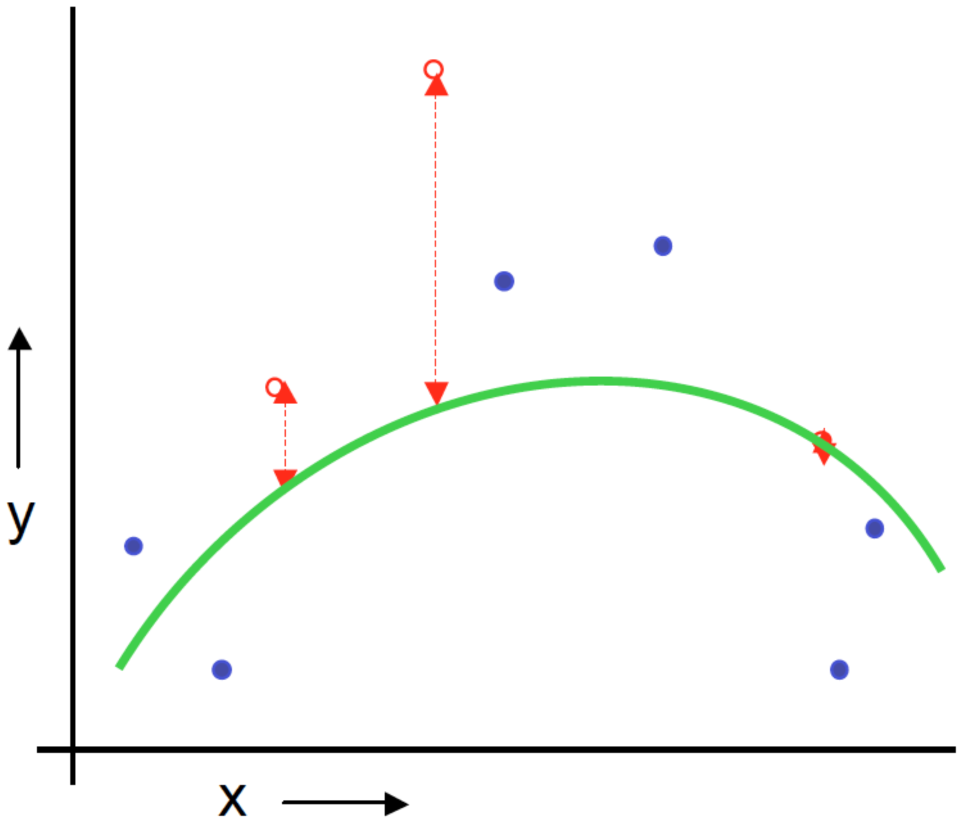



(Linear regression example)

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

# Test Set




(Quadratic regression example)

Mean Squared Error = 0.9

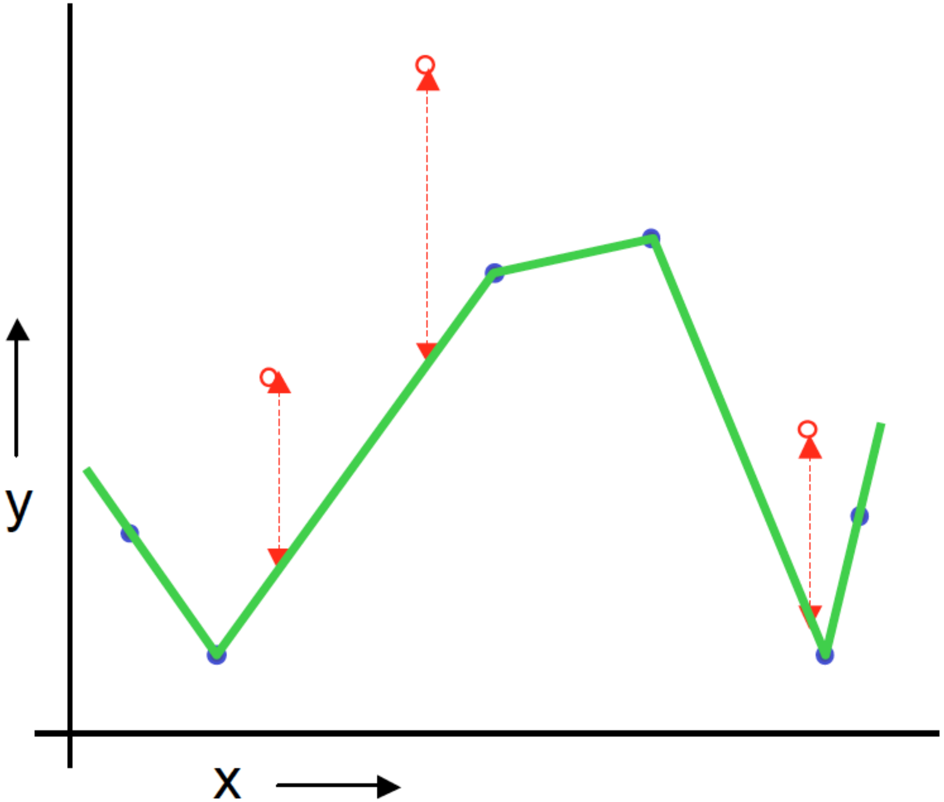
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

# Test Set




(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

# Test Set

Good news:

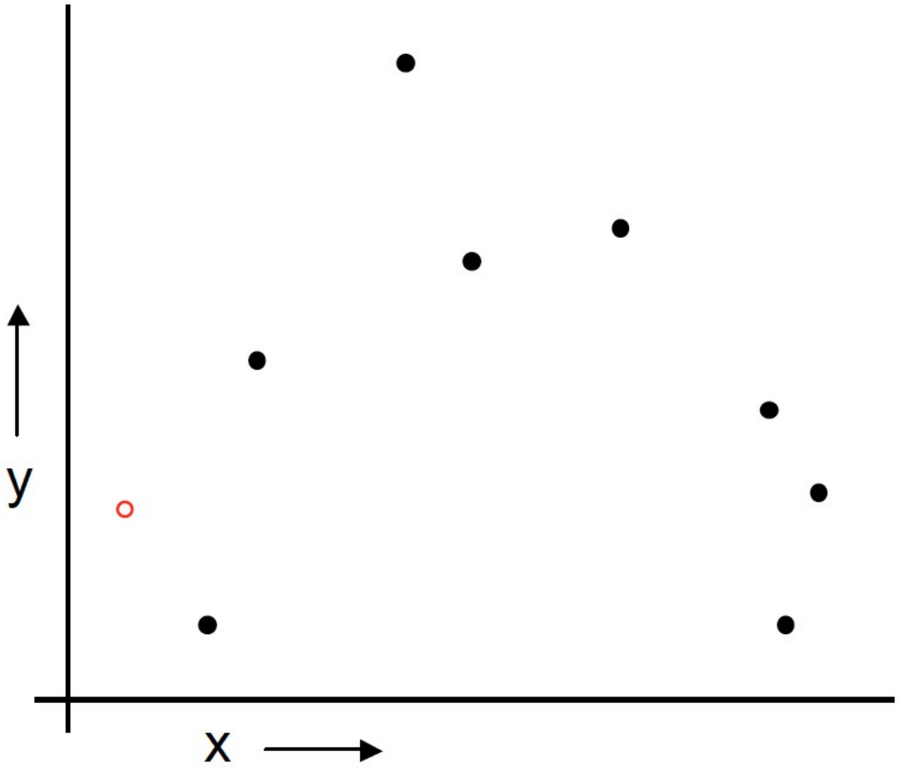
- Very very simple
- Can then simply choose the method with the best test-set score

Bad news:

- Wastes data: we get an estimate of the best method to apply to 30% less data
- 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"

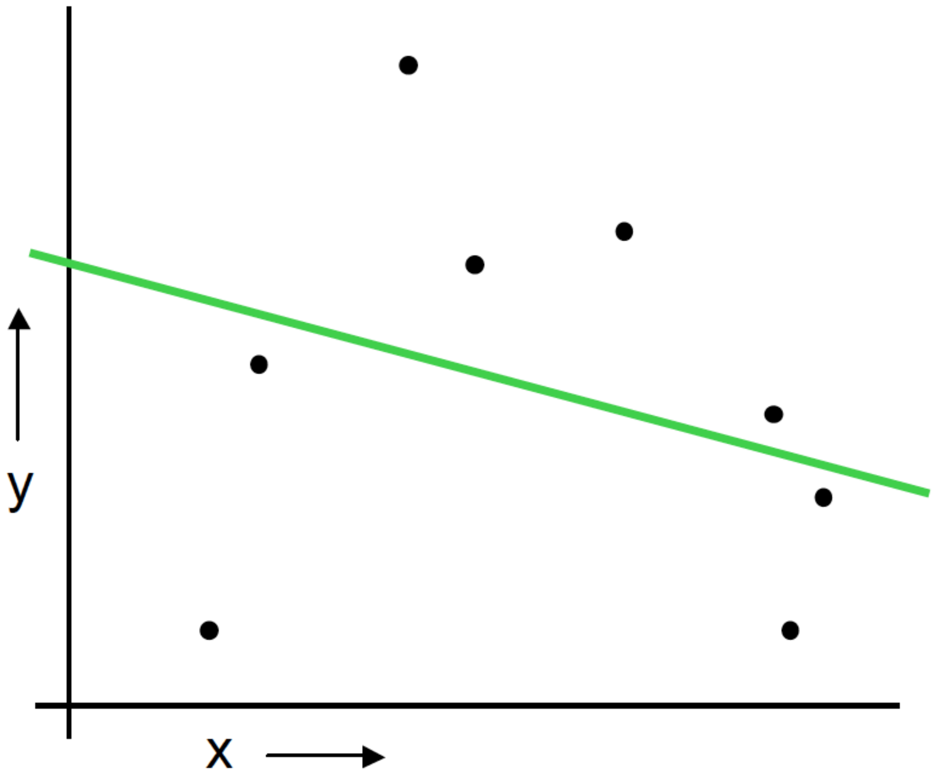
# Leave One Out Cross Validation




For k=1 to R

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

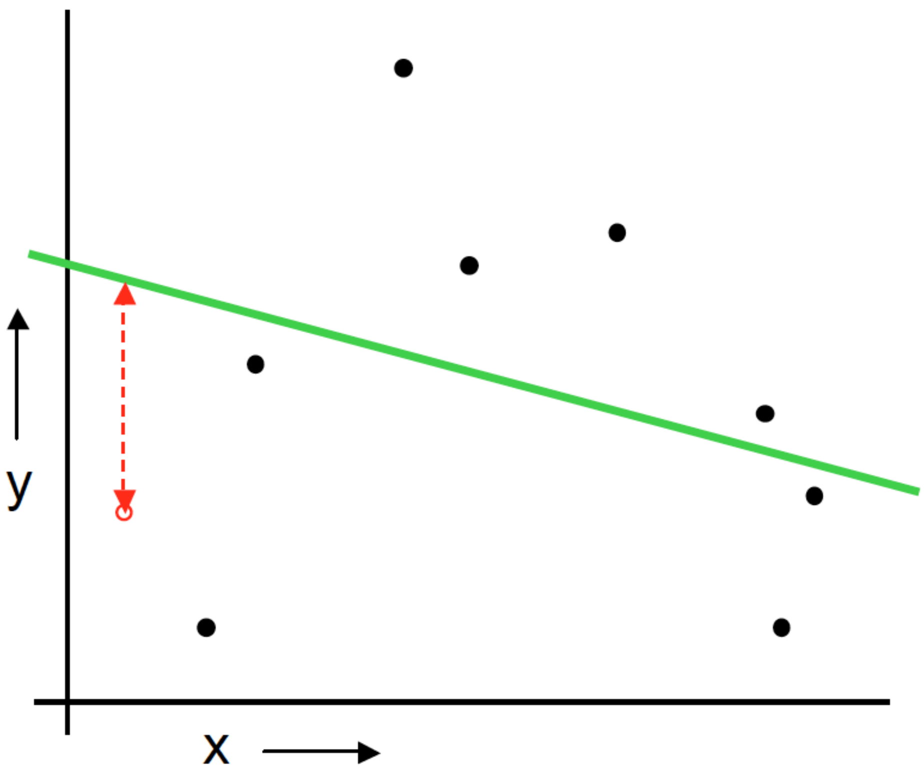
# Leave One Out Cross Validation




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

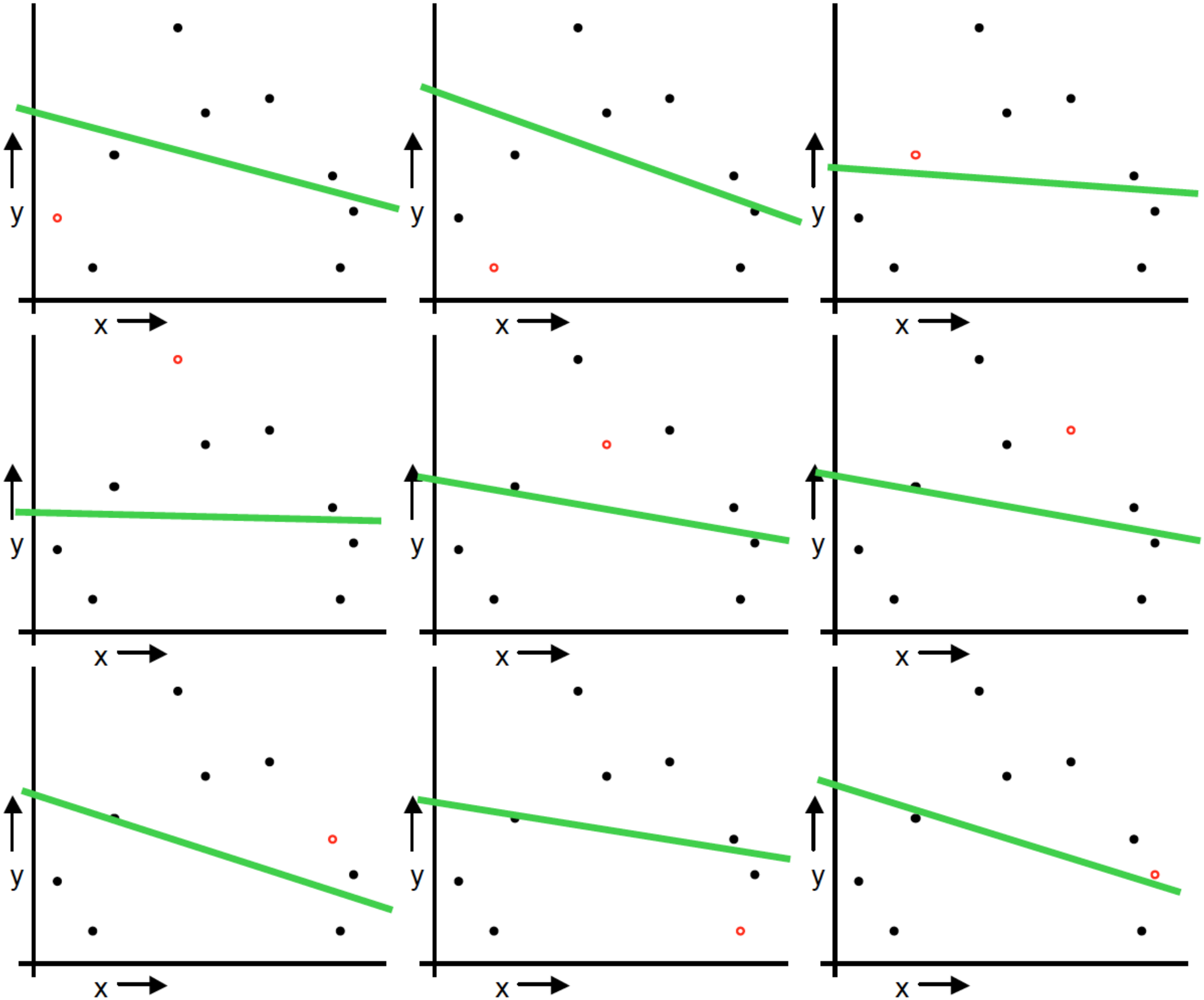
# Leave One Out Cross Validation




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 $(x_k,y_k)$

# Leave One Out Cross Validation



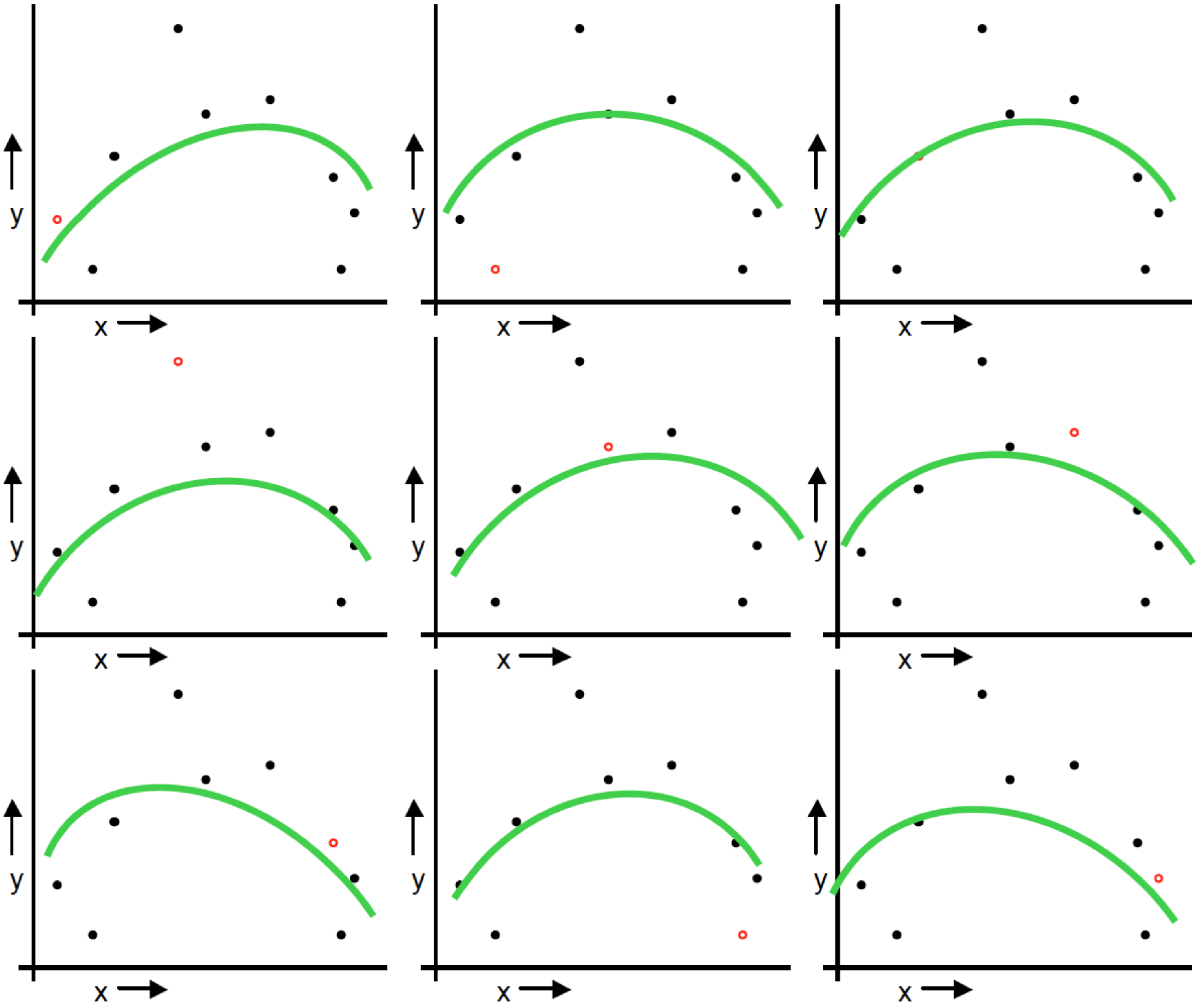

For k=1 to R

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.

$MSE_{LOOCV} = 2.12$

# Leave One Out Cross Validation



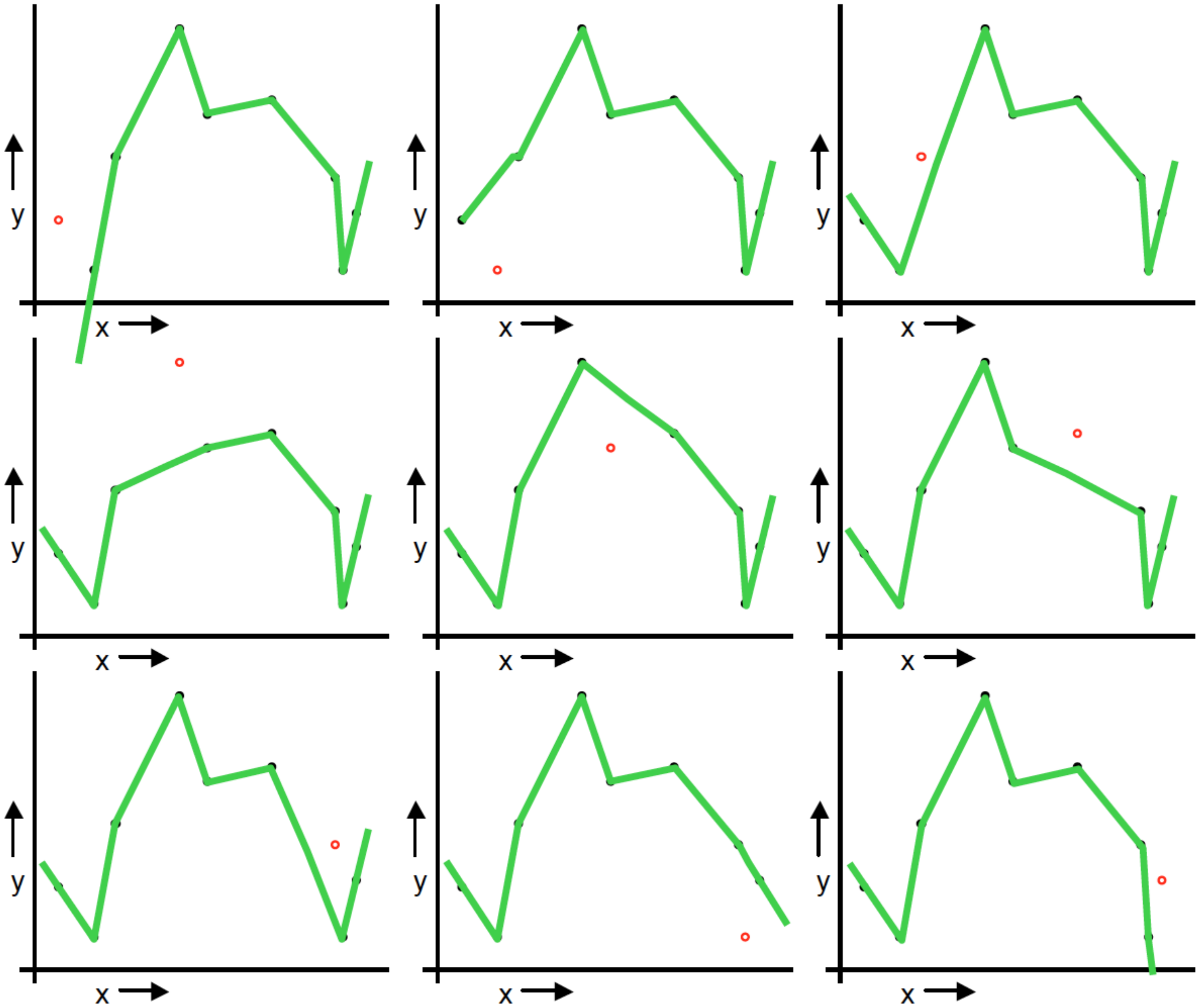

For k=1 to R

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.

$MSE_{LOOCV} = 0.962$

# 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 $(x_k, y_k)$

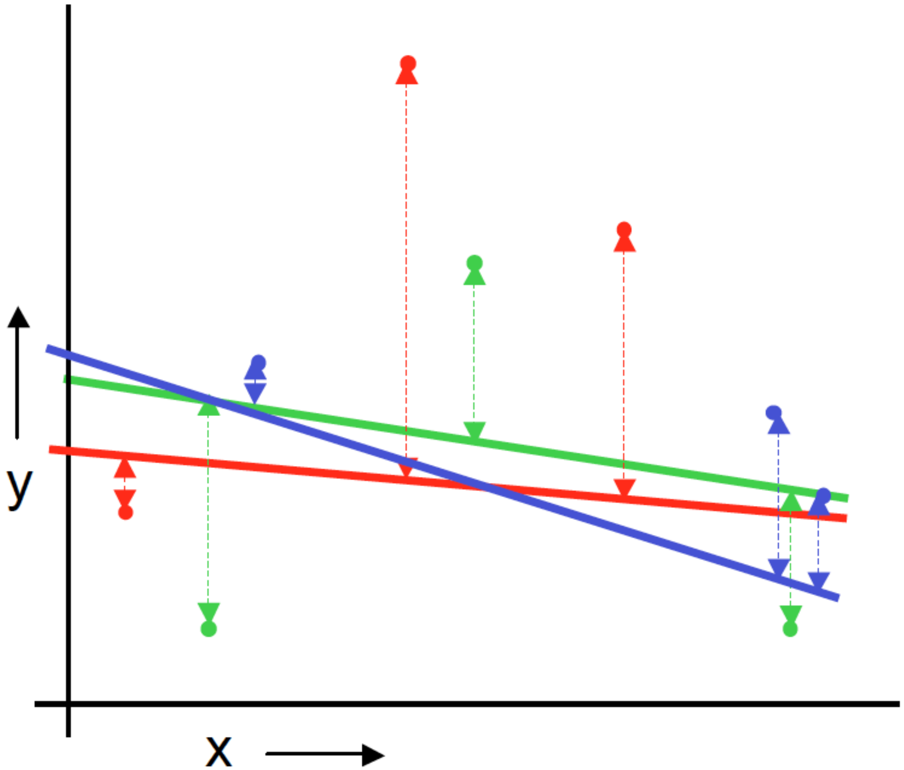
When you've done all points, report the mean error.

$MSE_{LOOCV} = 3.33$

# Pros and Cons

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

# 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)

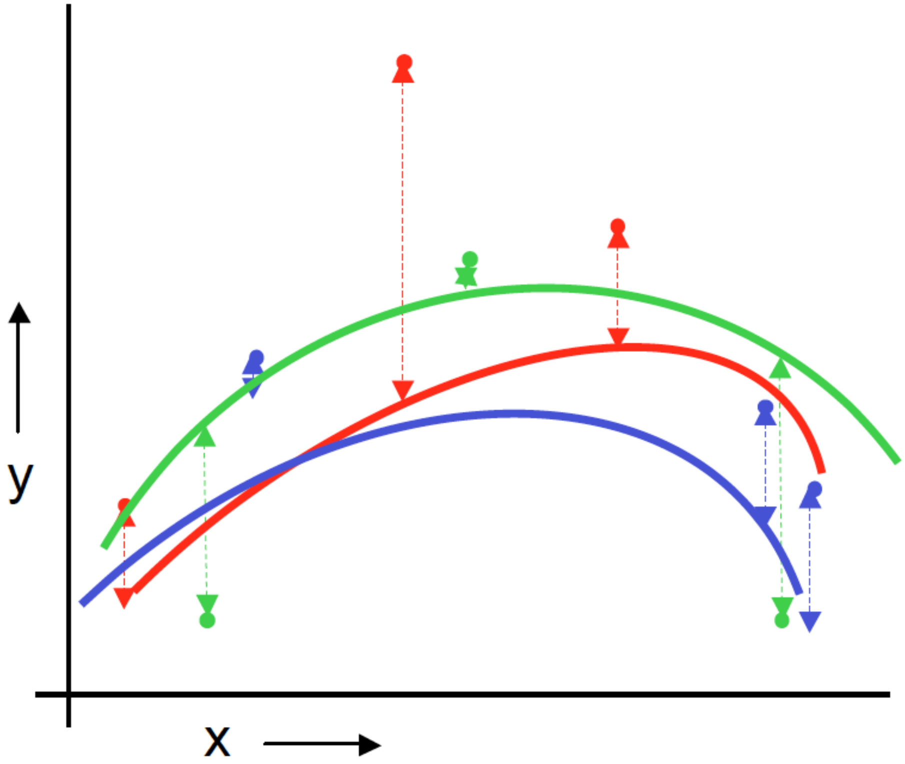
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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)

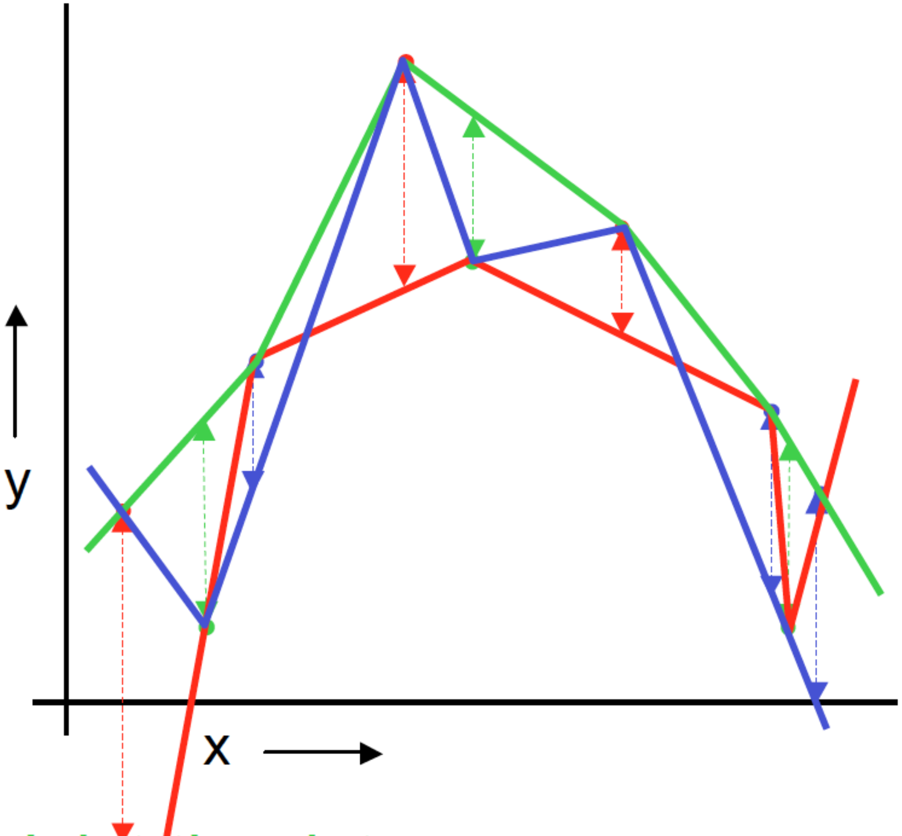
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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 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