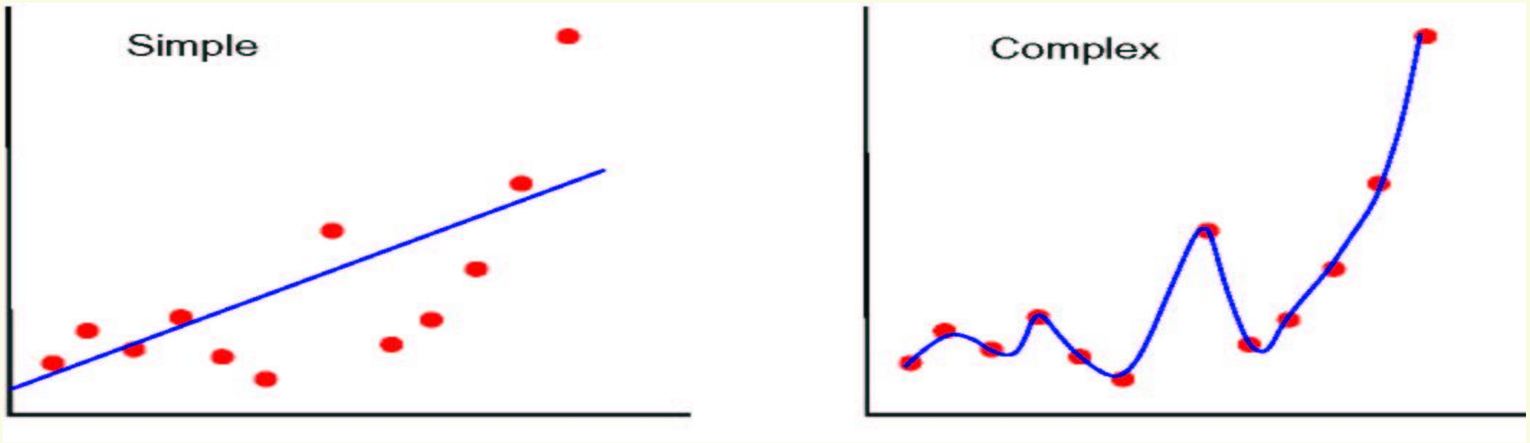
Bias-Variance Decomposition
Error Estimators
Cross-Validation

Bias-Variance tradeoff – Intuition

- Model too "simple"→does not fit the data well – a biased solution.



- Model too complex →small changes to the data, solution changes a lot – high-variance solution.

Expected Loss

- Let $h(x) = E[t|x]$ be the optimal predictor and $y(x)$ our actual predictor, which will incur the following expected loss

$$E(y(x)-t)^2 = \int\int \left(y(x)-h(x)+h(x)-t\right)^2 p(x,t)dxdt$$

$$= E\left(\left(y(x)-h(x)\right)^2 + 2\left(y(x)-h(x)\right)\left(h(x)-t\right)\right) + \left(h(x)-t\right)^2\right)$$

$$= \int \left(y(x)-h(x)\right)^2 p(x)dx + \int\int \left(h(x)-t\right)^2 p(x)dxdt$$

$$\text{since} \int \left(E[t|x]\text{-}t\right) p(t \mid x)dt = 0$$

$$E(y(x)-t)^2 = \int \left(y(x)-h(x)\right)^2 p(x)dx + \int\int \left(h(x)-t\right)^2 p(x)dt$$

Noise term

Source of error 1

Sources of error 1 –noise

- What if we have perfect learner, infinite data?
  - Our learning solution y(x) satisfies y(x)=h(x)
  - Still have remaining, *unavoidable error* of σ^2 due to noise ε

$$\int_x \int_t \left(y(x) - t\right)^2 p(f(x) = t \mid x)\, p(x)\, dt\, dx$$

h(x) $\quad h(x) + \varepsilon$

$$\varepsilon^2 \quad , \ \varepsilon \sim N(0, \sigma^2)$$

$$= \mathrm{E}[\varepsilon^2] = \sigma^2$$

Bias-variance decomposition

- Focus on $\int \left(y(x) - h(x)\right)^2 p(x)dx$
- Let us first examine expected loss averaging over data sets.
- For one data set D and one test point x:

$$\left(y(x,D) - h(x)\right)^2 = \left(y(x,D) - E_D[y(x,D)] + E_D[y(x,D)] - h(x)\right)^2$$

$$= \left(y(x,D) - E_D[y(x,D)]\right)^2 + \left(E_D[y(x,D)] - h(x)\right)^2$$

$$+ 2\left(y(x,D) - E_D[y(x,D)]\right)\left(E_D[y(x,D)] - h(x)\right)$$

→ E_D of this cancels to zero

- Hence

$$E_D\left(y(x,D) - h(x)\right)^2$$

$$= \left(E_D[y(x,D)] - h(x)\right)^2 + E_D\left(y(x,D) - E_D[y(x,D)]\right)^2$$

$$= \text{bias}^2 + \text{variance}$$

Bias

- Suppose you are given a dataset *D* with *m* samples from some distribution.
- You learn function $y(x)$ from data *D*
- If you sample a different datasets, you will learn different $y(x)$
- **Expected hypothesis**: $E_D[y(x,D)]$
- **Bias:** difference between what you expect to learn and the truth.
  - Measures how well you expect to represent true solution
  - Decreases with more complex model

$$\text{bias}^2 = \int_x \left(E_D[y(x,D)] - h(x)\right)^2 p(x)dx$$

Expected to learn

True model

Variance

- **Variance:** difference between what you expect to learn and what you learn from a particular dataset
  - Measures how sensitive learner is to a specific dataset
  - Decreases with simpler model

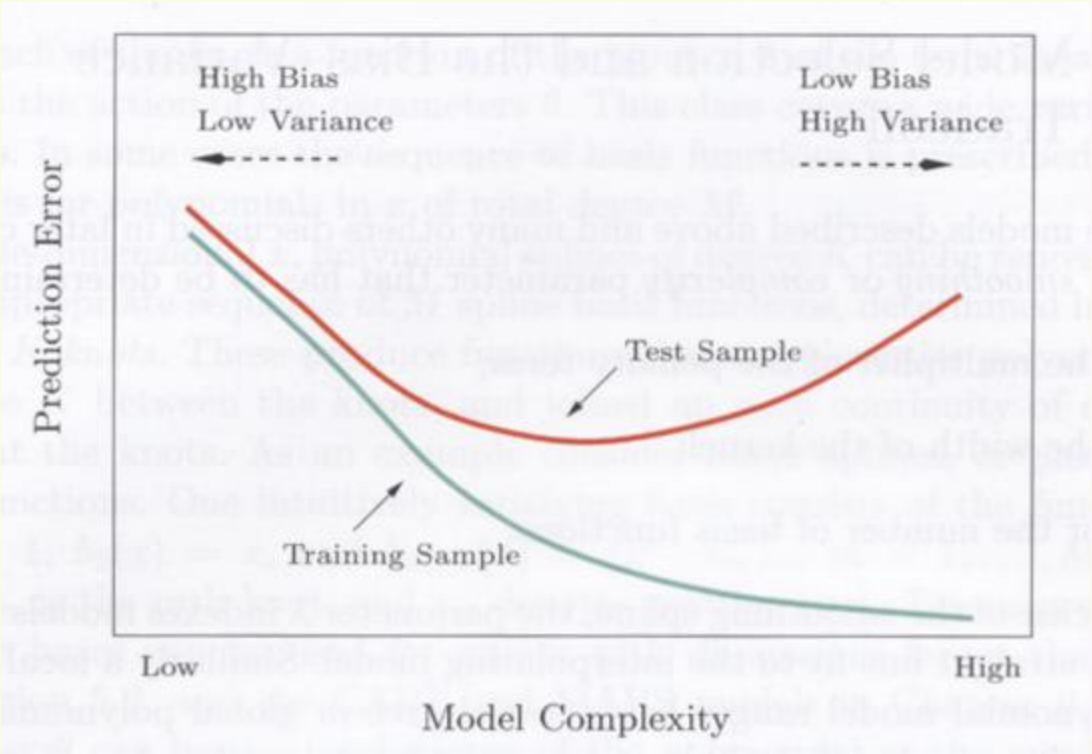
$$\text{variance} = \int_x E_D\left(\left(y(x,D) - E_D[y(x,D)]\right)^2\right)p(x)dx$$

Learned for D

Expected to learn, averaged over datasets

Bias-Variance Tradeoff

- Choice of hypothesis class introduces learning bias
- More complex class →less bias
- More complex class →more variance



The Expected Prediction Error

- The expected prediction squared error over fixed size training sets *D* drawn from P(X,T) can be expressed as sum of three components:

$$\text{unavoidable error} + \text{bias}^2 + \text{variance}$$

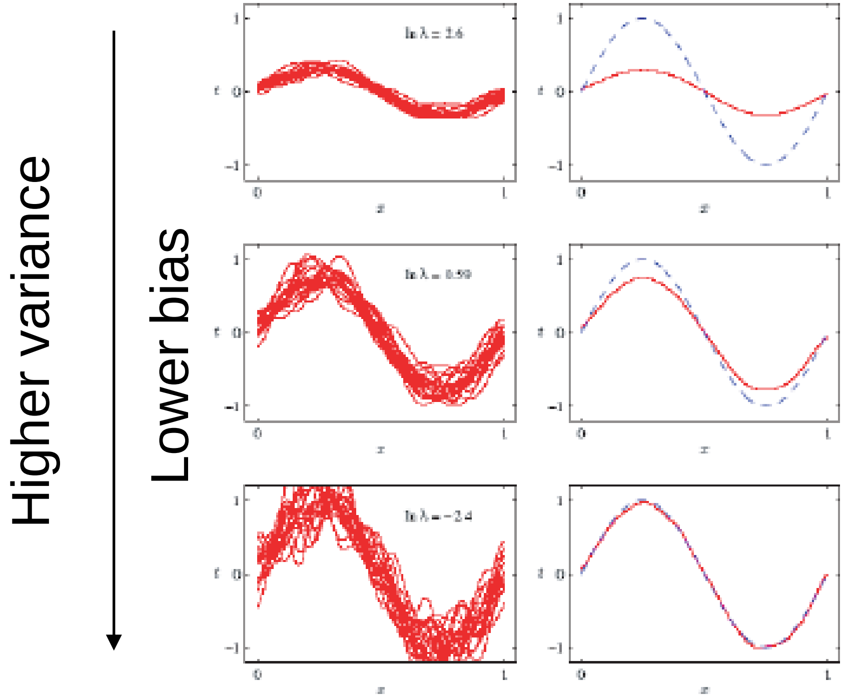
where

$$\text{unavoidable error} = \sigma^2$$

$$\text{bias}^2 = \int_x \left(E_D[y(x)] - h(x)\right)^2 p(x)dx$$

$$\text{variance} = \int E_D\left[\left(y(x) - E_D[y(x)]\right)^2\right] p(x)dx$$

Average over training sets



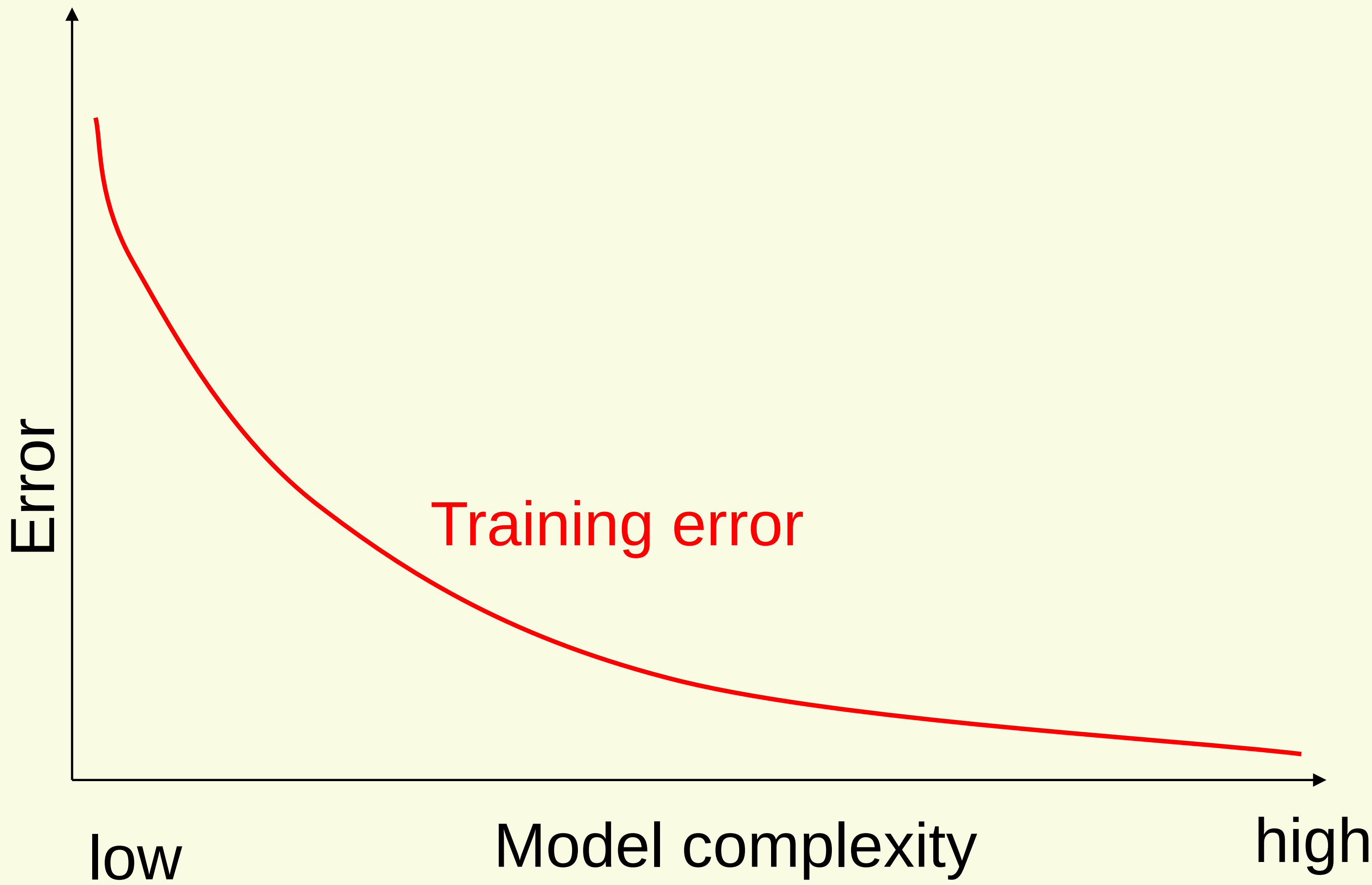
Highly regularized have low variance but high bias.

Training Error

- Given a training data, choose a loss function. (e.g., squared error (L2) for regression)
- **Training set error:** For a particular set of parameters, loss function on training data:

$$Err_{train}(w) = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left(t(x_i) - \sum_{j=1}^{M} w_i y(x_j) \right)^2$$

Training Error vs. Model Complexity



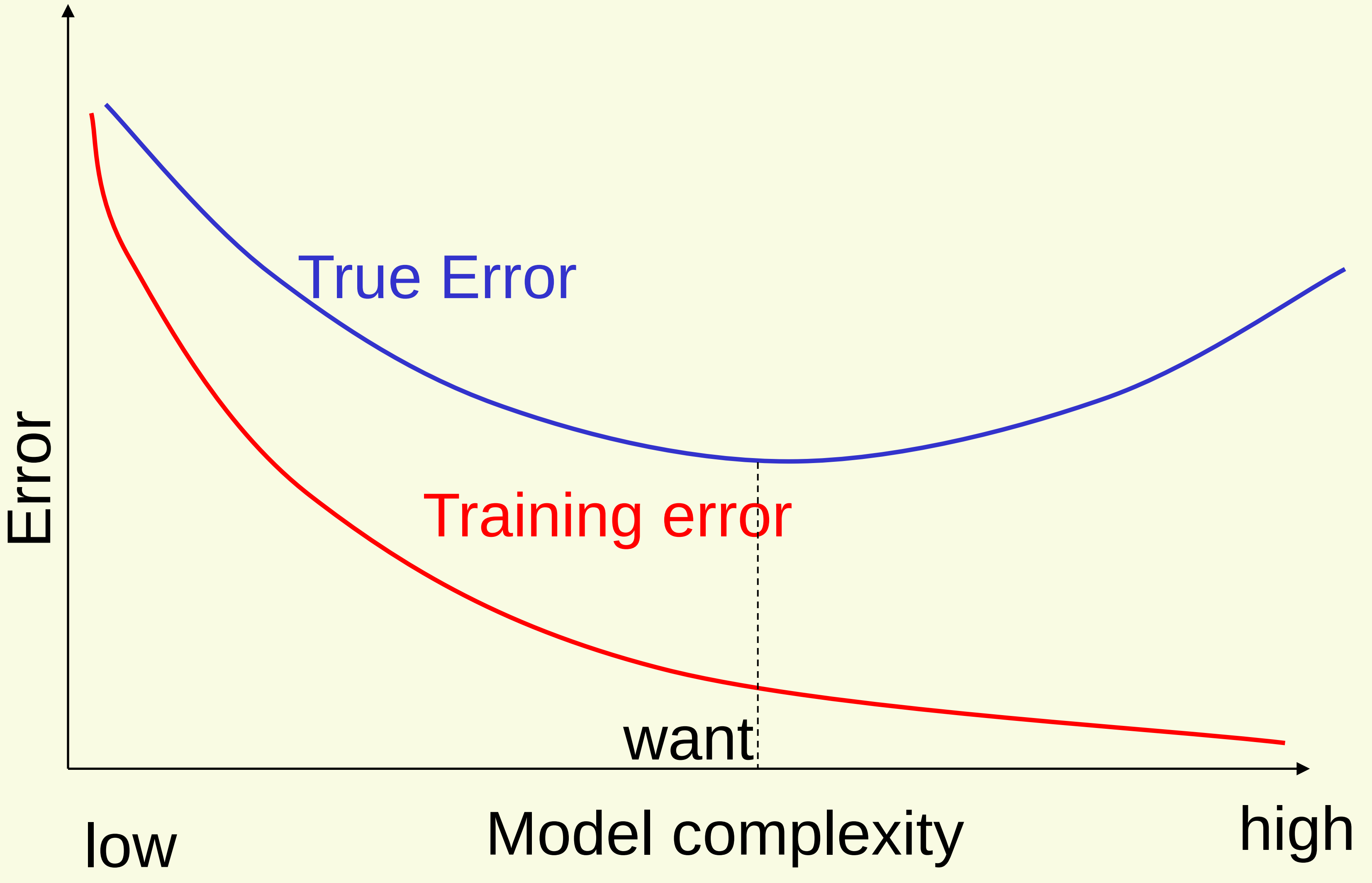
Prediction (Generalization) Error

- Training set error can be poor measure of "quality" of solution
- **Prediction error:** We really care about error over all possible input points, not just training data:

$$Err_{true}(w) = E_{\mathrm{X}}\left[\left(t(x) - \sum_{j=1}^{M} w_i y(x_j)\right)^2\right]$$

$$= \int_{\mathrm{X}} \left(t - \sum_{j=1}^{M} w_i y(x_j)\right)^2 dx$$

Prediction Error vs. Model Complexity



Why training set error doesn't approximate prediction error?

- Sampling approximation of prediction error:

$$Err_{true}(w) \approx \frac{1}{p}\sum_{i=1}^{p}\left(t - \sum_{j=1}^{M} w_i y(x_j)\right)^2$$

- Training error

$$Err_{train}(w) = \frac{1}{N_{train}}\sum_{i=1}^{N_{train}}\left(t(x_i) - \sum_{j=1}^{M} w_i y(x_j)\right)^2$$

- Very similar equations!!!

  Why is training set a bad measure of prediction error???

Why is training set a bad measure of prediction error???

- Training error is a good estimate for a single **w,**
- But we optimized **w** with respect to the training error, and found **w** that is good for this set of samples.

- **Training error is a (optimistically) biased estimate of prediction error**

Test Error

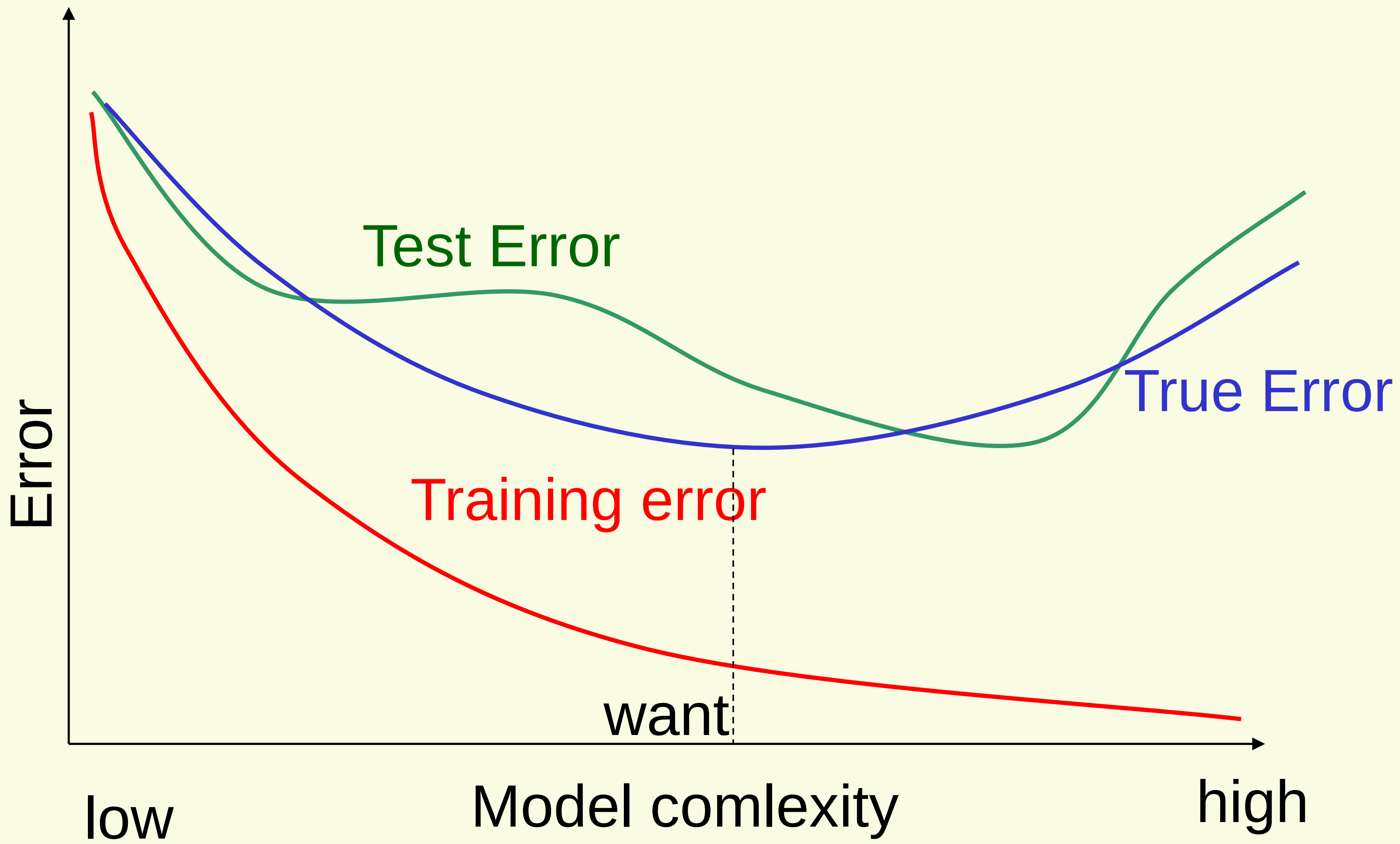
- Given a dataset, **randomly** split it into two parts:
  - Training data –{$\mathbf{x}_1, \ldots, \mathbf{x}_{Ntrain}$}
  - Test data –{$\mathbf{x}_1, \ldots, \mathbf{x}_{Ntest}$}
- Use training data to optimize parameters **w**
- **Test set error:** For the ***final solution*** **w***, evaluate the error using:

$$Err_{test}(w^*) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \left(t(x_i) - \sum_{j=1}^{M} w^*{}_i \, y(x_j) \right)^2$$

Unbiased but has variance

Test set only unbiased if you never do any learning on the test data
For example, if you use the test set to select the degree of the polynomial…no longer unbiased!!!

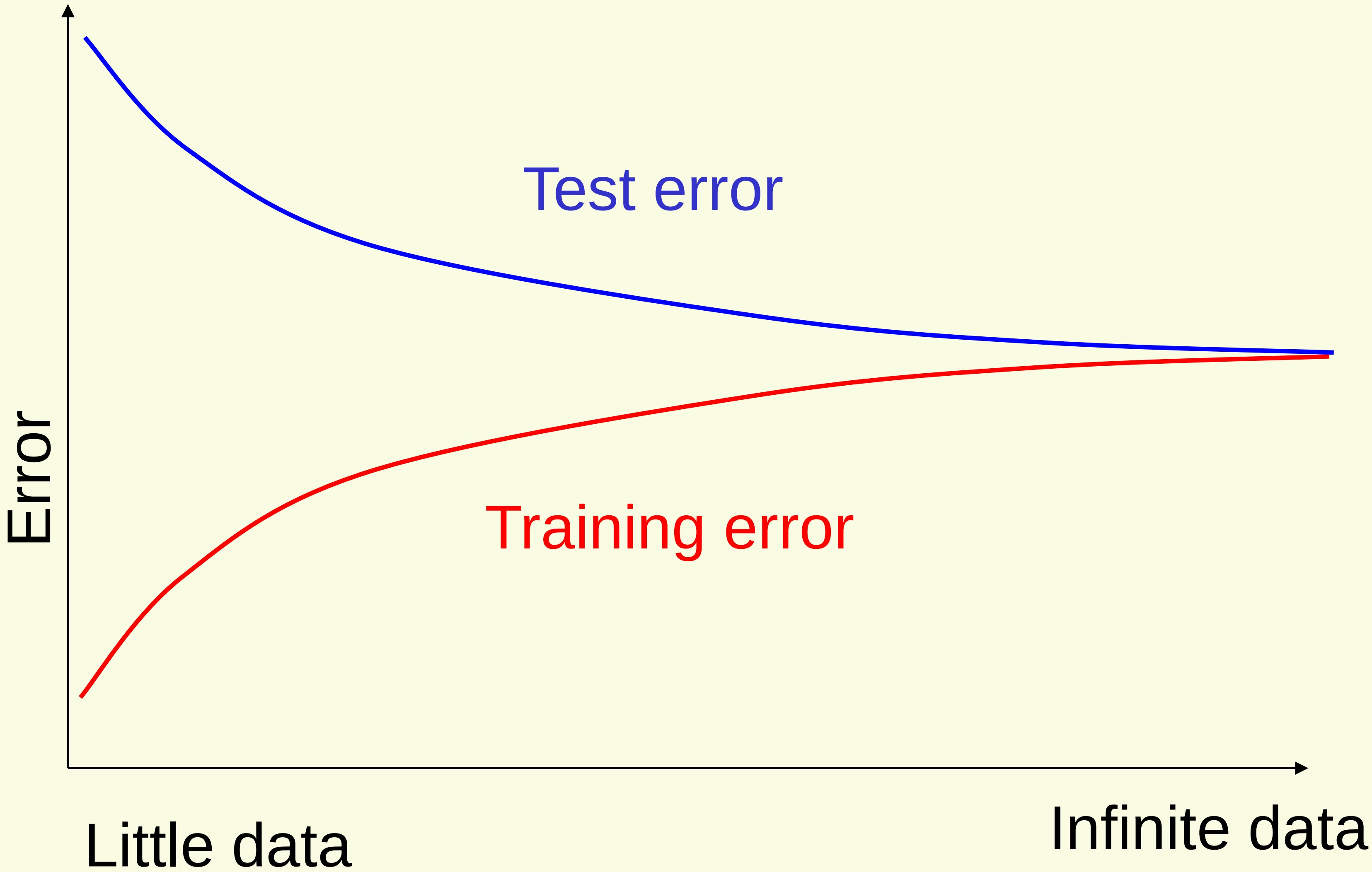
Test Set Error vs. Model Complexity



How many points to use for training/testing?

- Very hard question to answer!
  - Too few training points, learned **w** is bad
  - Too few test points, you never know if you reached a good solution
- Theory proposes error bounds (advanced course)
- Typically:
  - if you have a reasonable amount of data, pick test set “large enough” for a “reasonable” estimate of error, and use the rest for learning
  - if you have little data, then you need to use some special techniques e.g., bootstrapping

Error as a function of number of training examples for a fixed model complexity



Overfitting

- With too few training examples our model may achieve zero training error but never the less has a large generalization error

- When the training error no longer bears any relation to the generalization error the model overfits the data

Cross-validation for detecting and preventing overfitting

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.

Andrew W. Moore

Professor

School of Computer Science

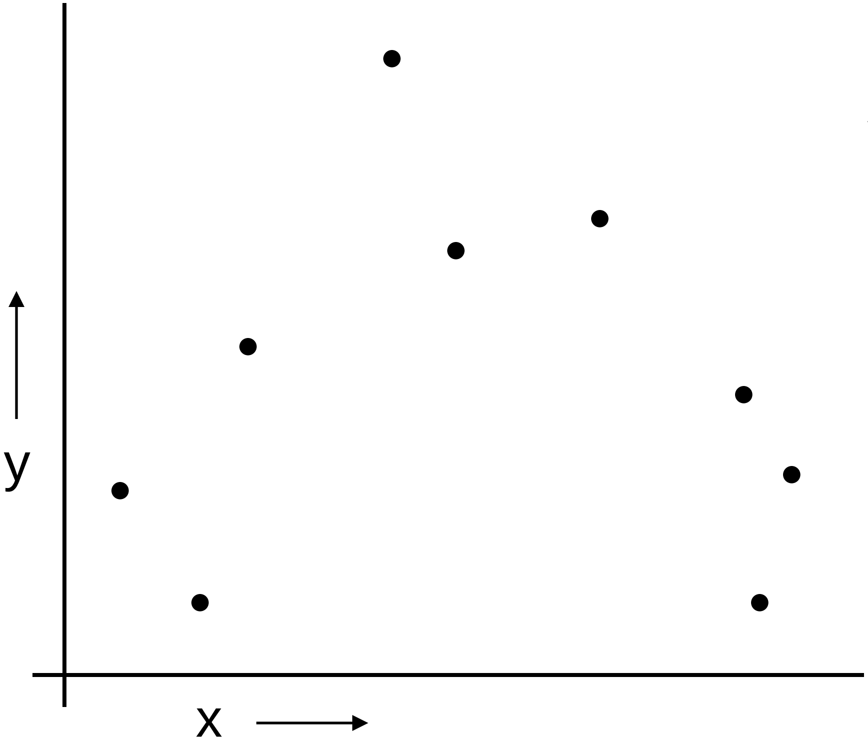
Carnegie Mellon University

www.cs.cmu.edu/~awm

awm@cs.cmu.edu

412-268-7599

A Regression Problem

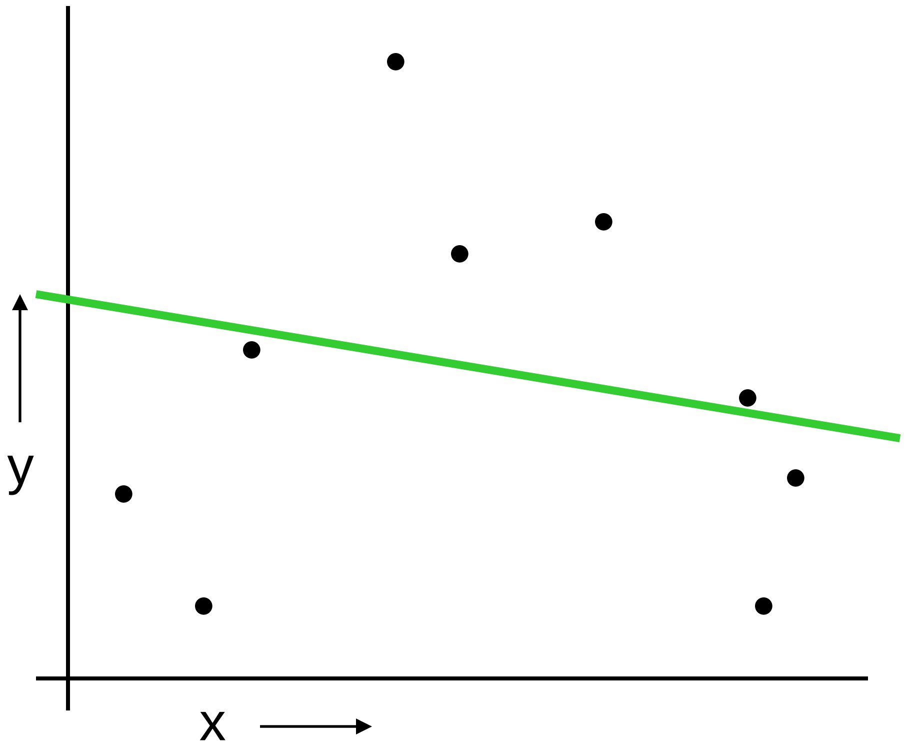


y = f(x) + noise

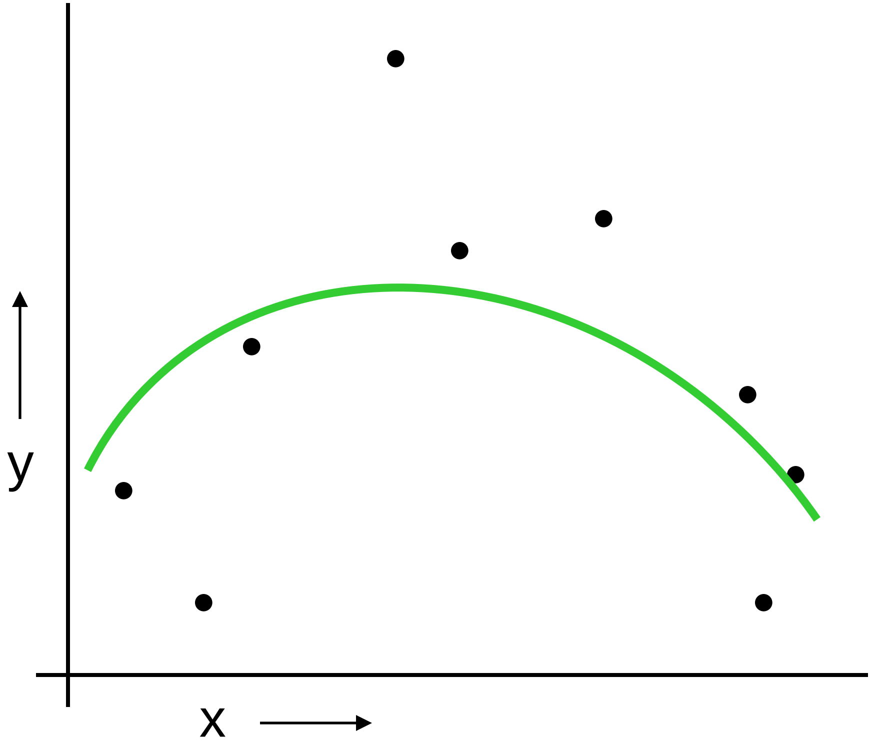
Can we learn f from this data?

Let's consider three methods…

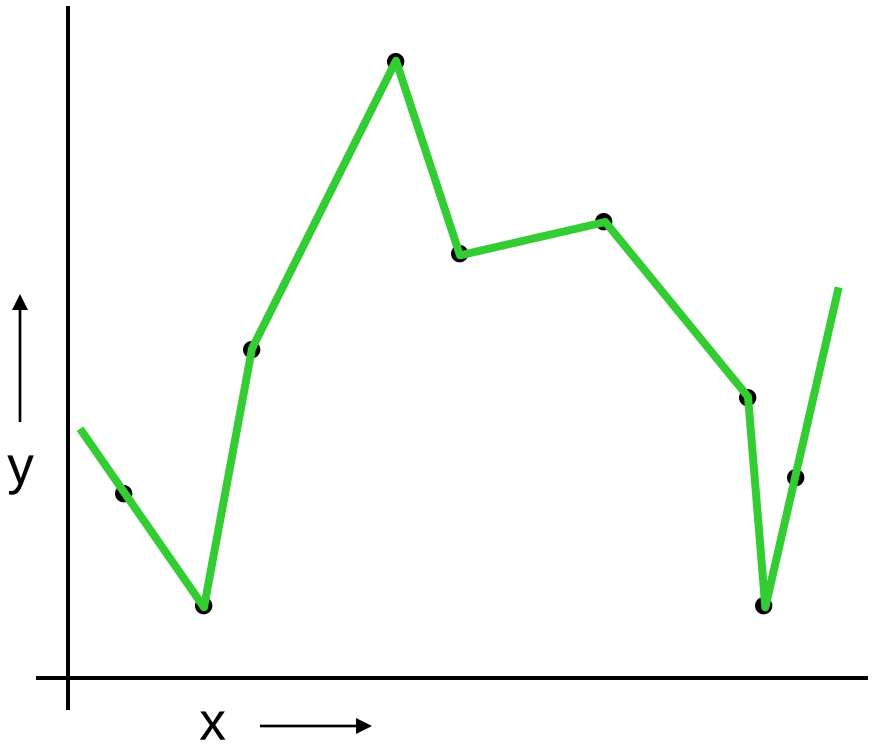
Linear Regression



Quadratic Regression

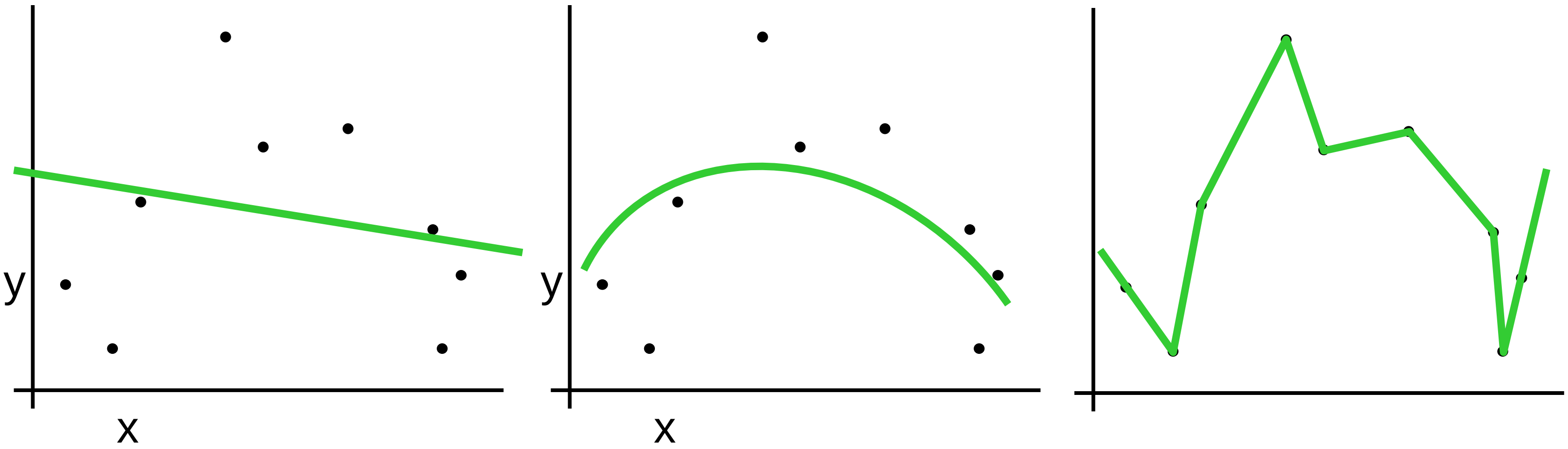


Join-the-dots



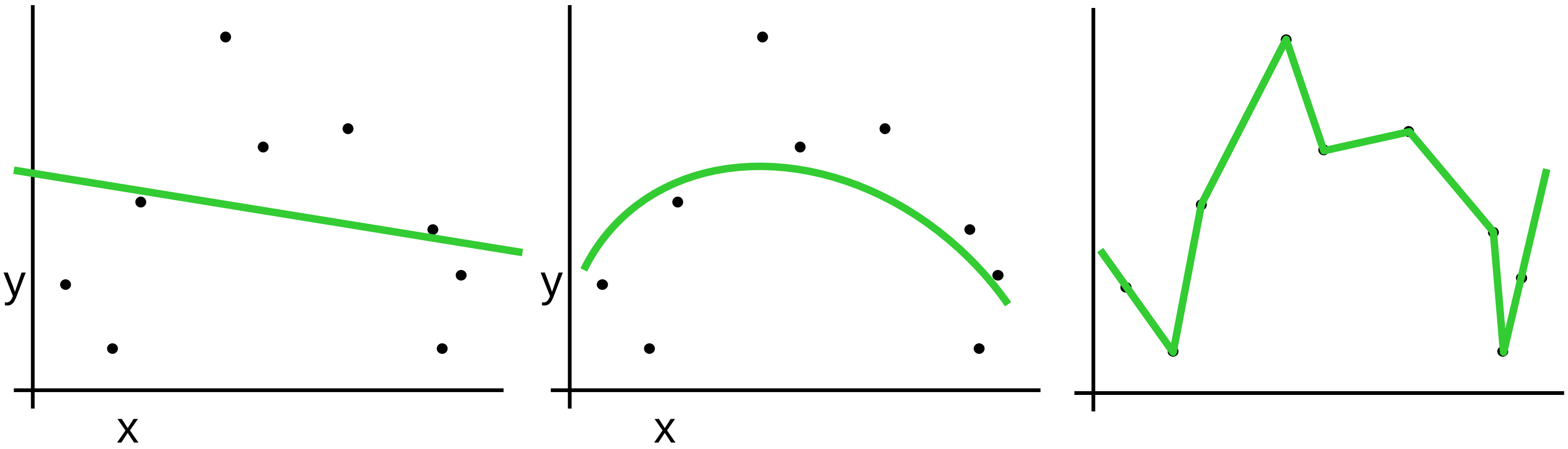
Also known as piecewise linear nonparametric regression if that makes you feel better

Which is best?



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

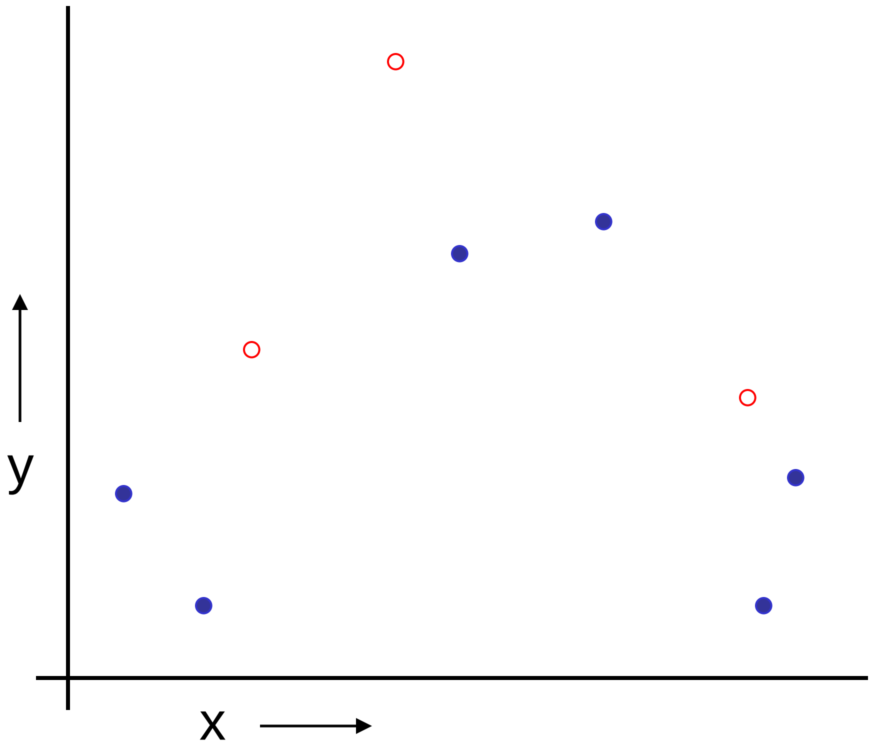
What do we really want?



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?"

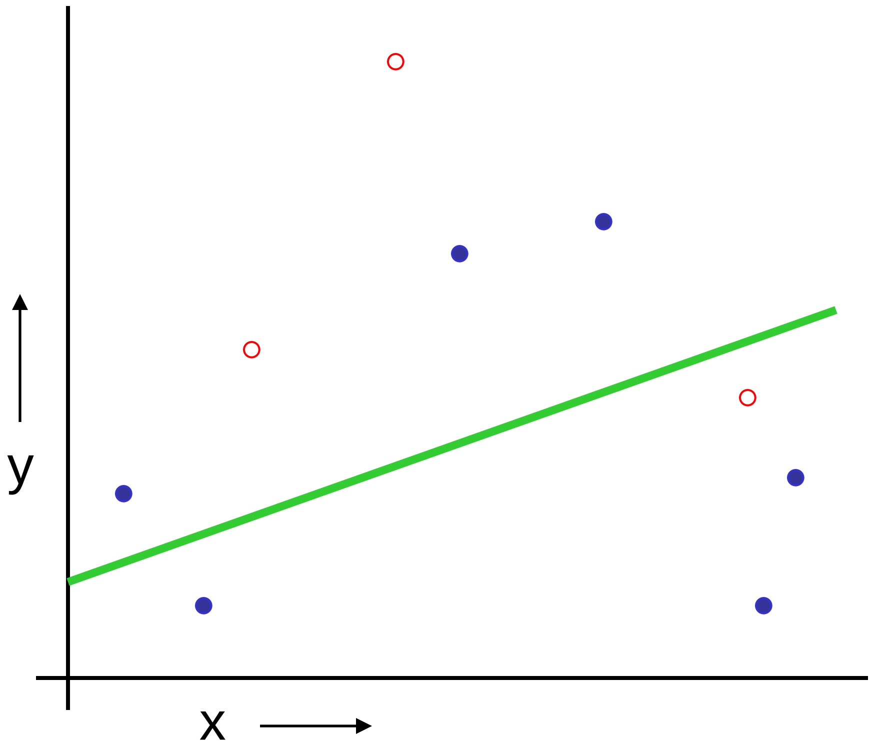
The test set method



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

2. The remainder is a training set

The test set method



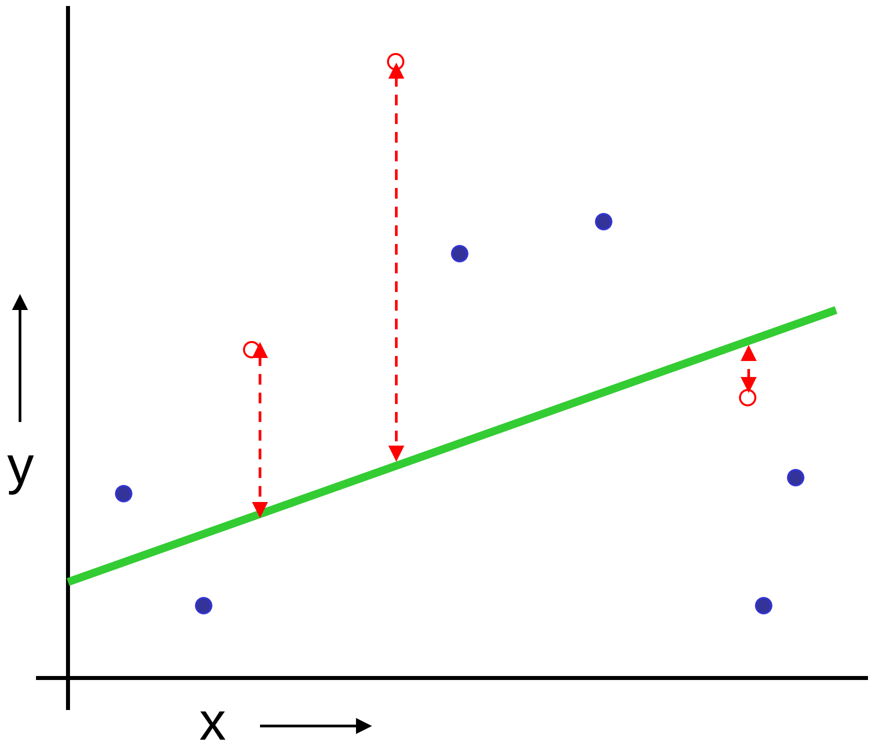
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

(Linear regression example)

The test set method



(Linear regression example)

Mean Squared Error = 2.4

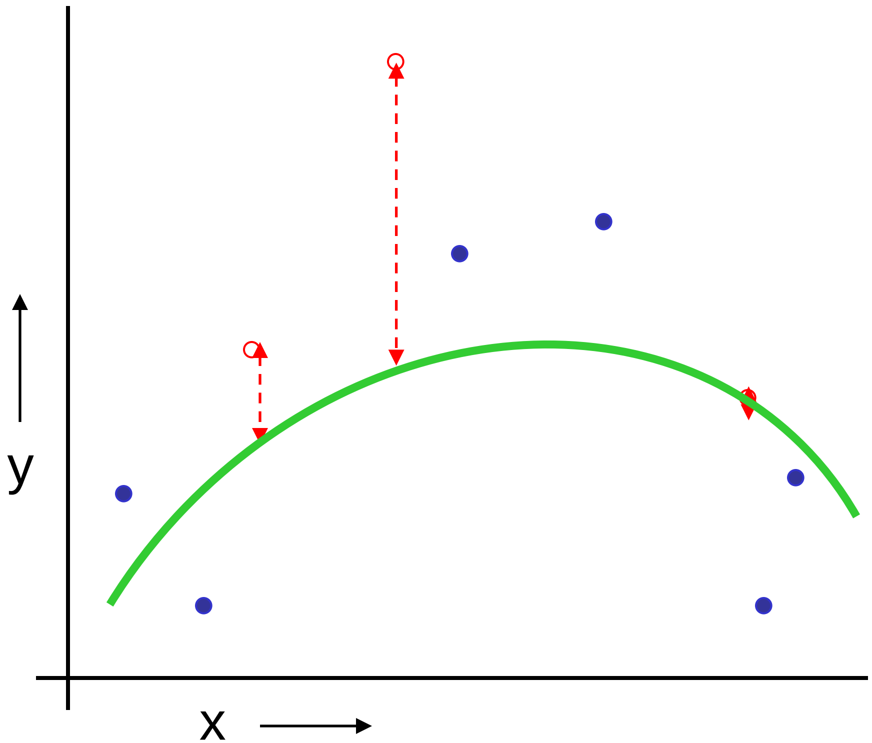
1. Randomly choose 30% of the data to be in a test set

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



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

2. The remainder is a training set

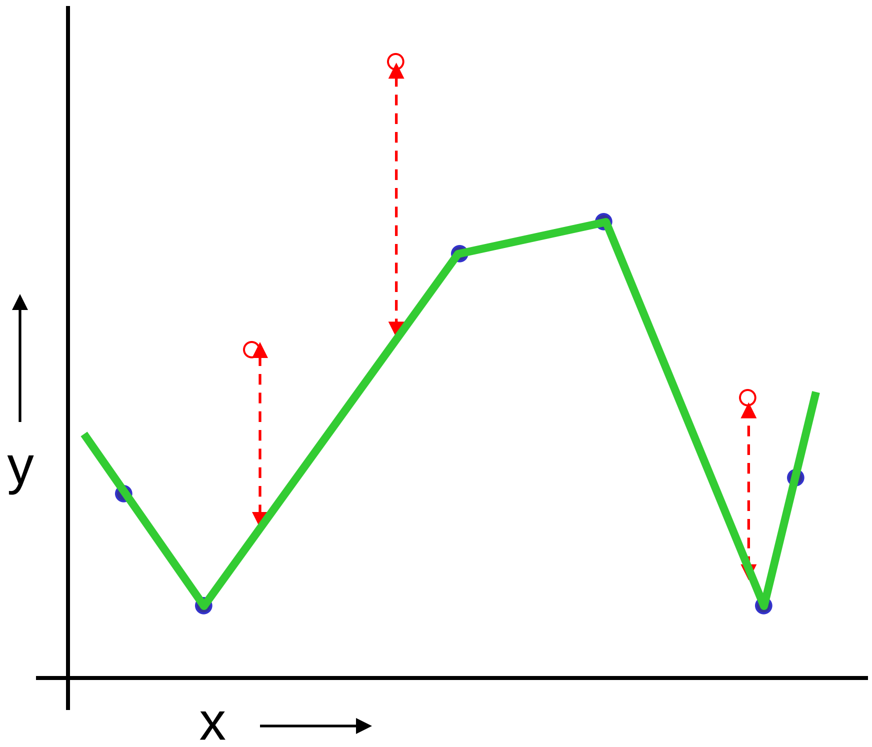
3. Perform your regression on the training set

4. Estimate your future performance with the test set

(Quadratic regression example)

Mean Squared Error = 0.9

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

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

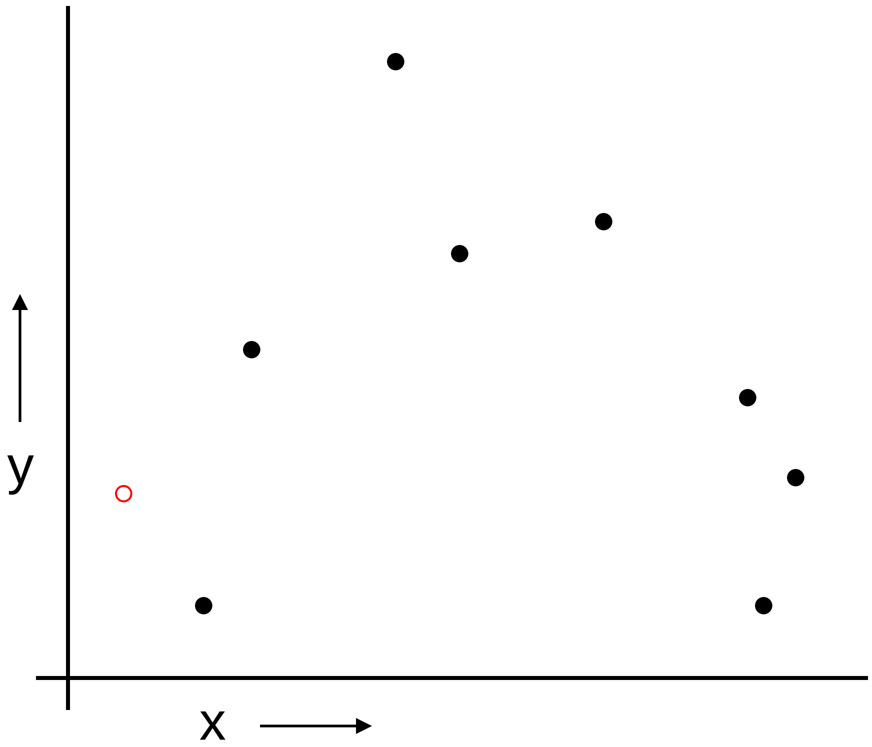
Bad news:

•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"

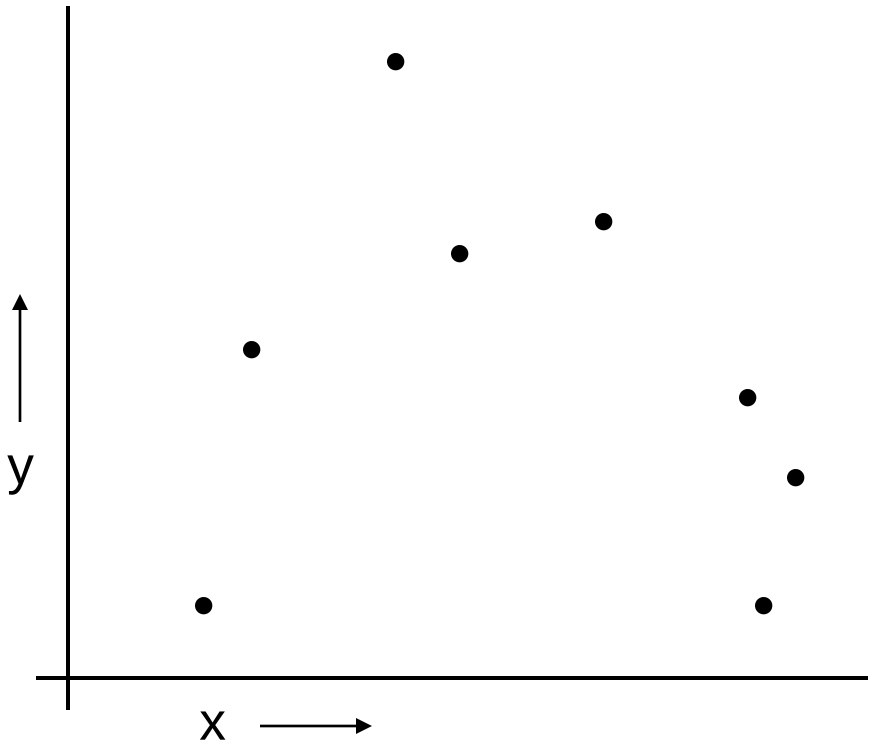
LOOCV (Leave-one-out Cross Validation)



For k=1 to R

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

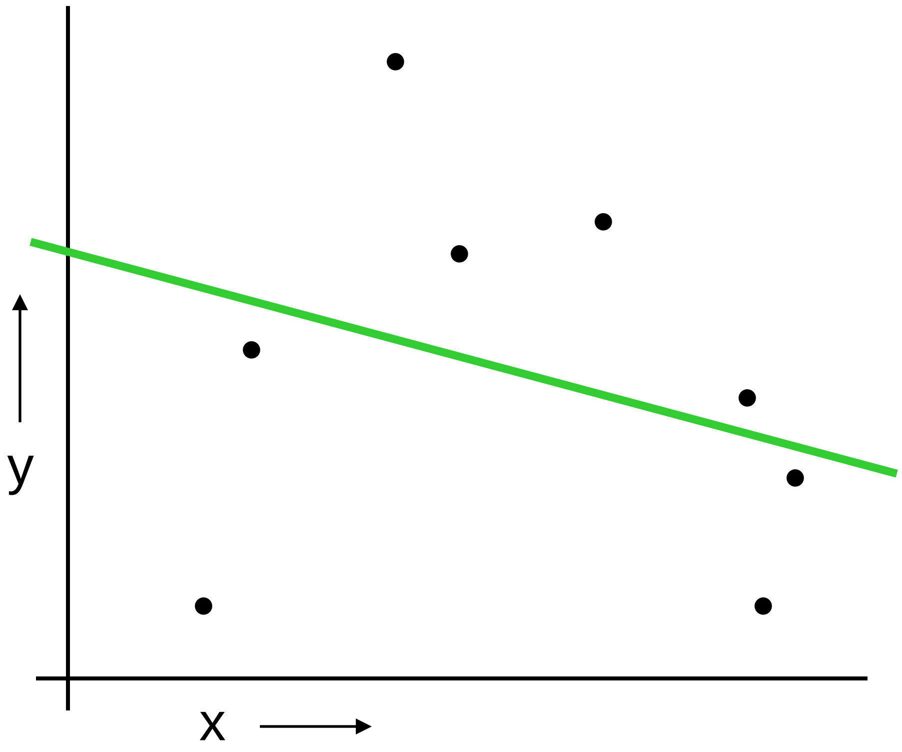
LOOCV (Leave-one-out Cross Validation)



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

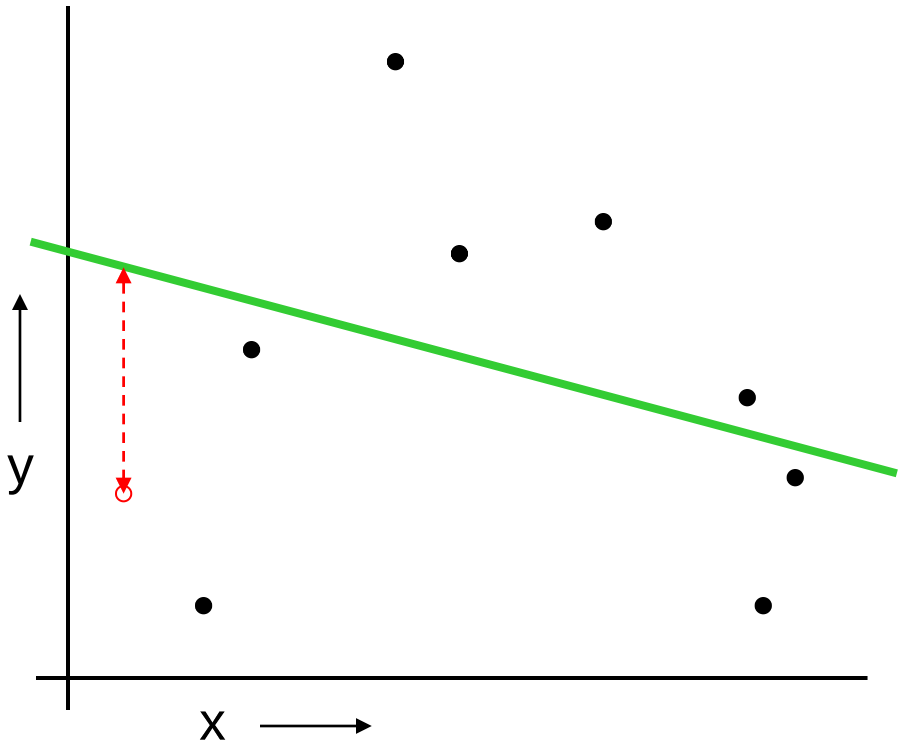
LOOCV (Leave-one-out Cross Validation)



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

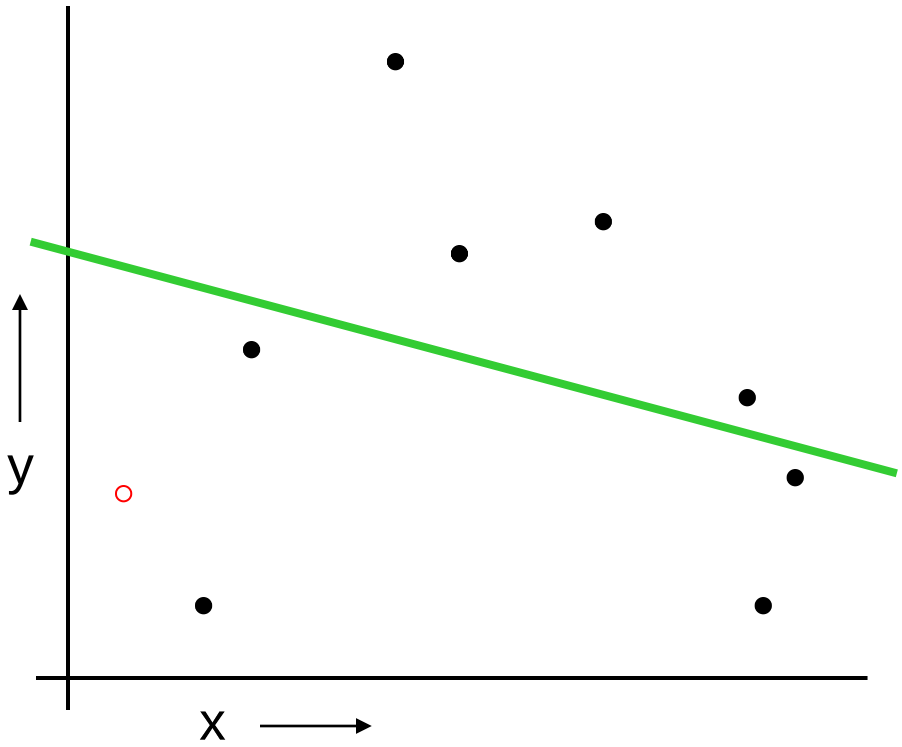
LOOCV (Leave-one-out Cross Validation)



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)

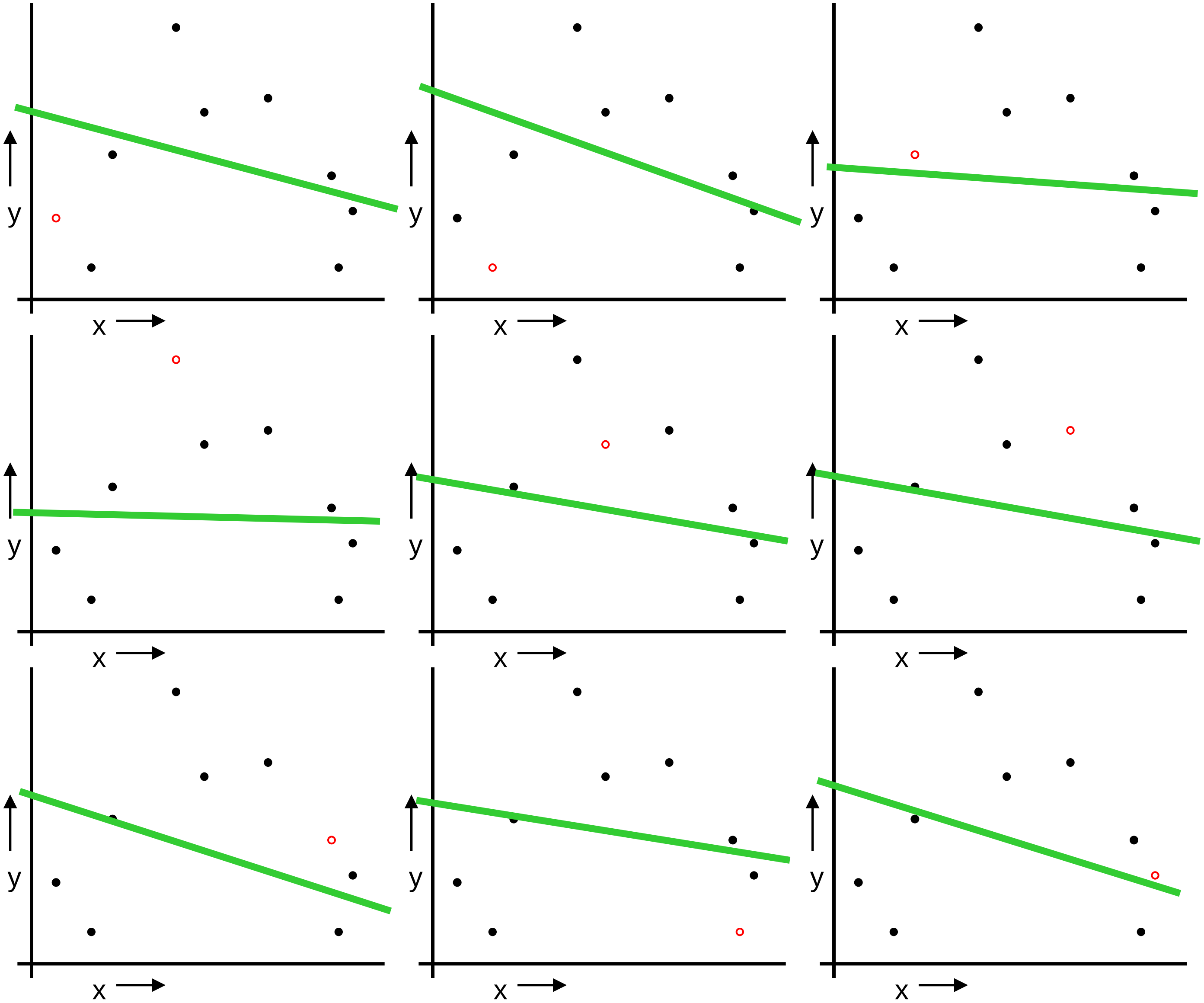


For k=1 to R

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)



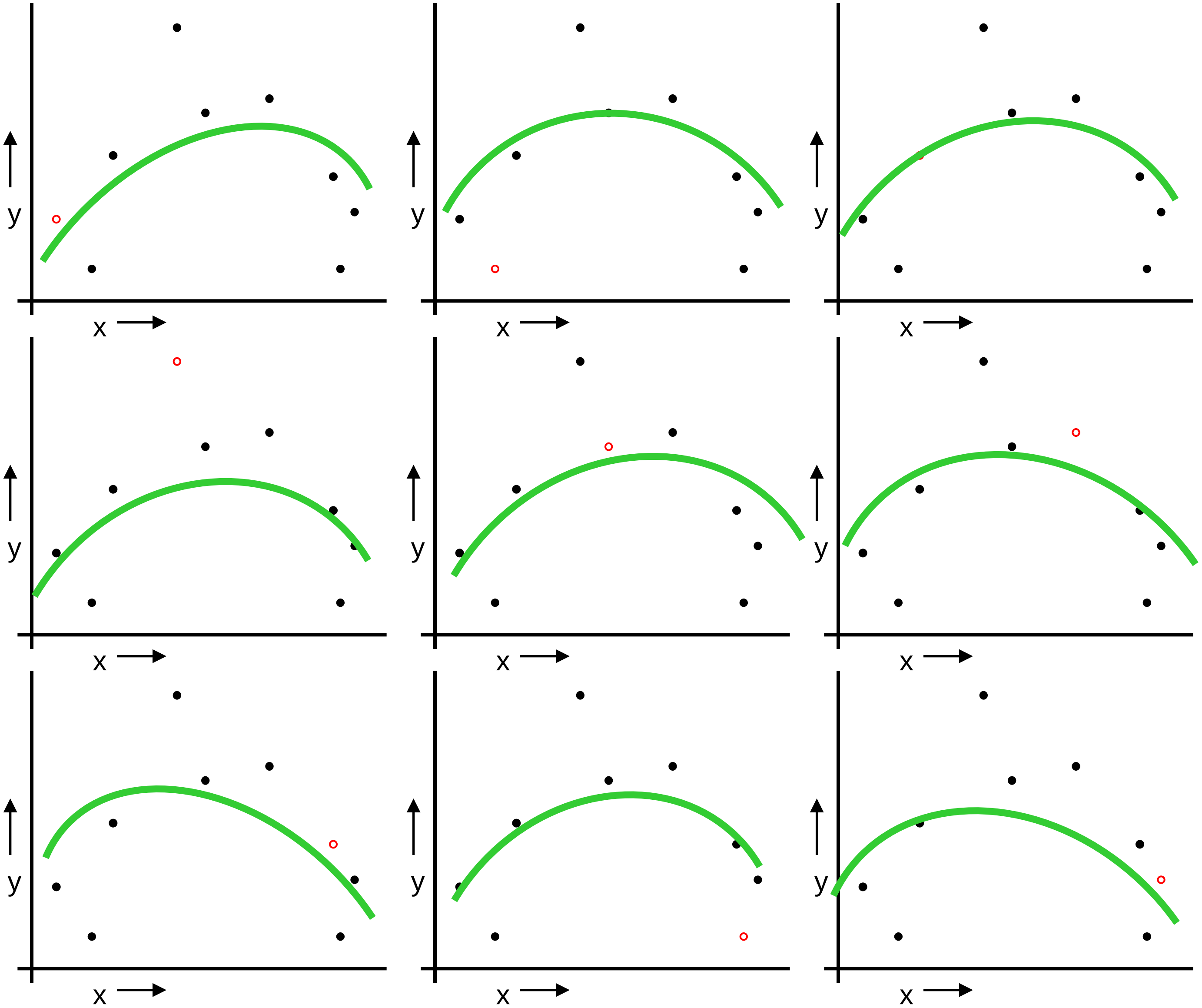
For k=1 to R

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



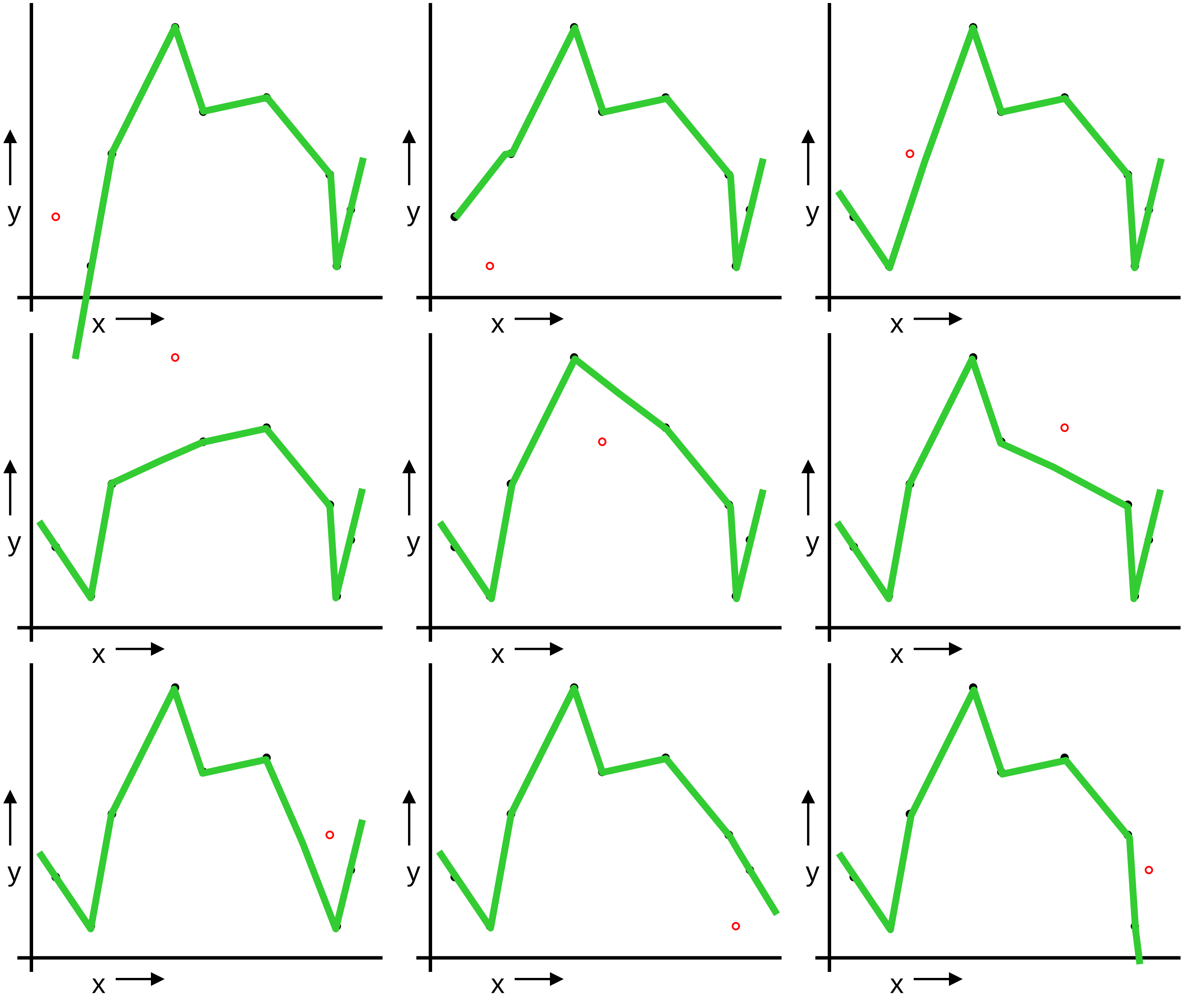
For k=1 to R

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.

$MSE_{LOOCV} = 0.962$

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.

$MSE_{LOOCV} = 3.33$

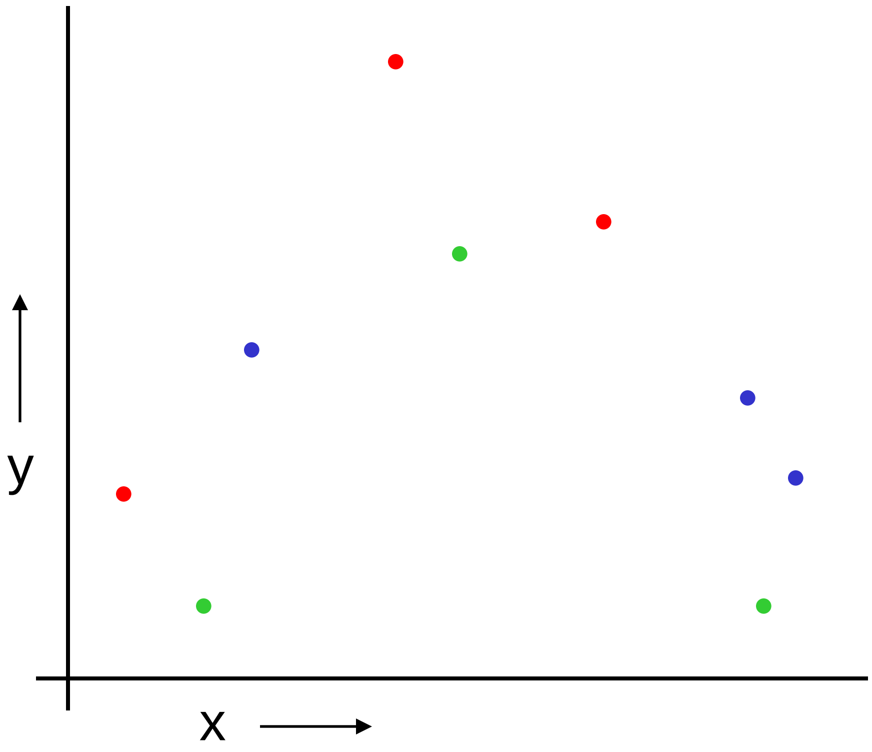
Which kind of Cross Validation?

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

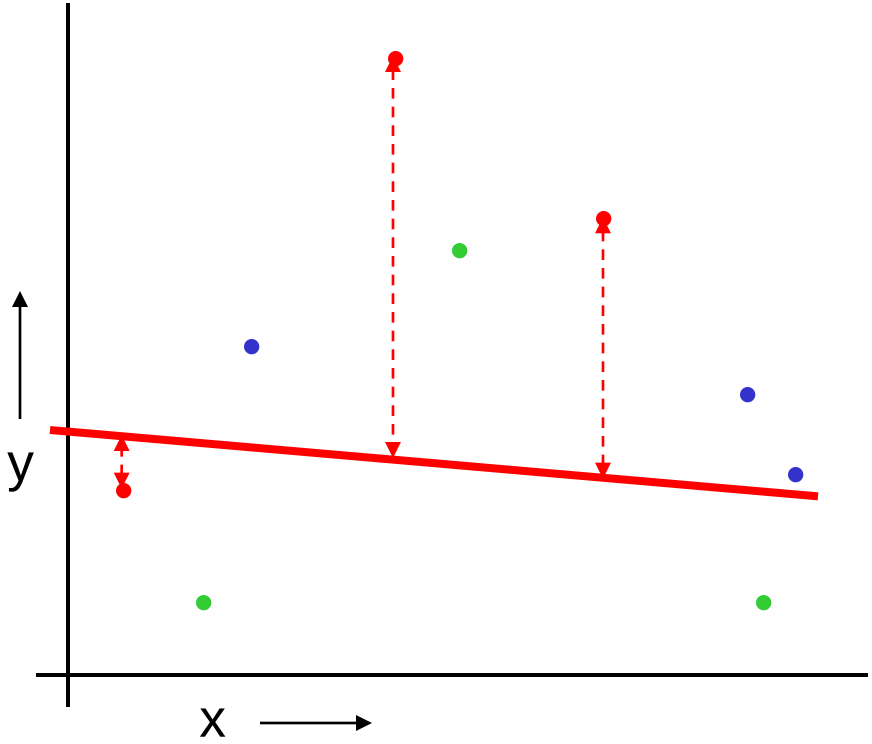
..can we get the best of both worlds?

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)



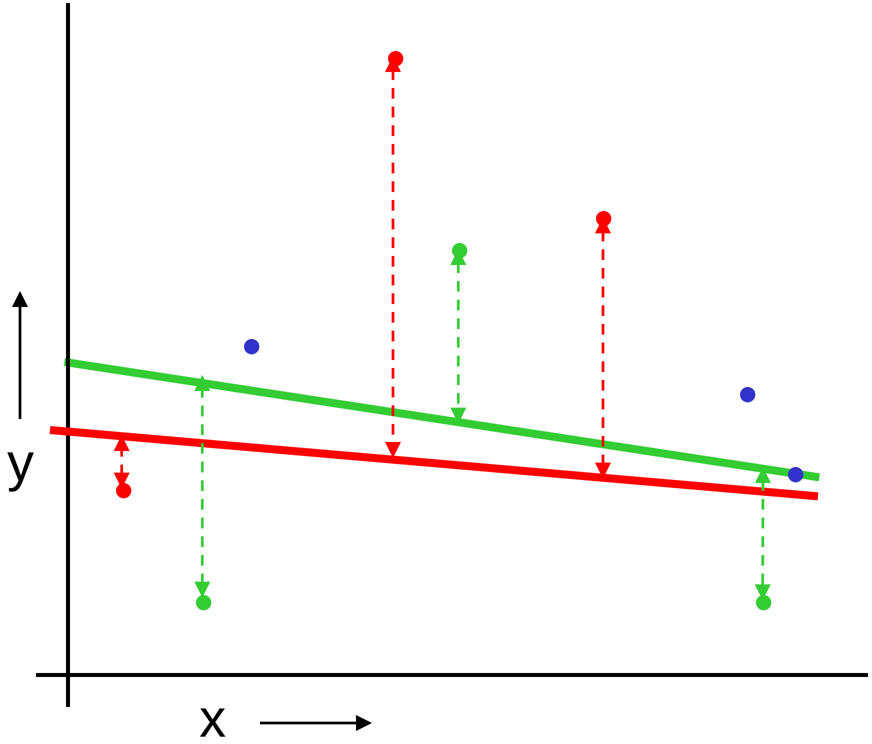
k-fold Cross Validation



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

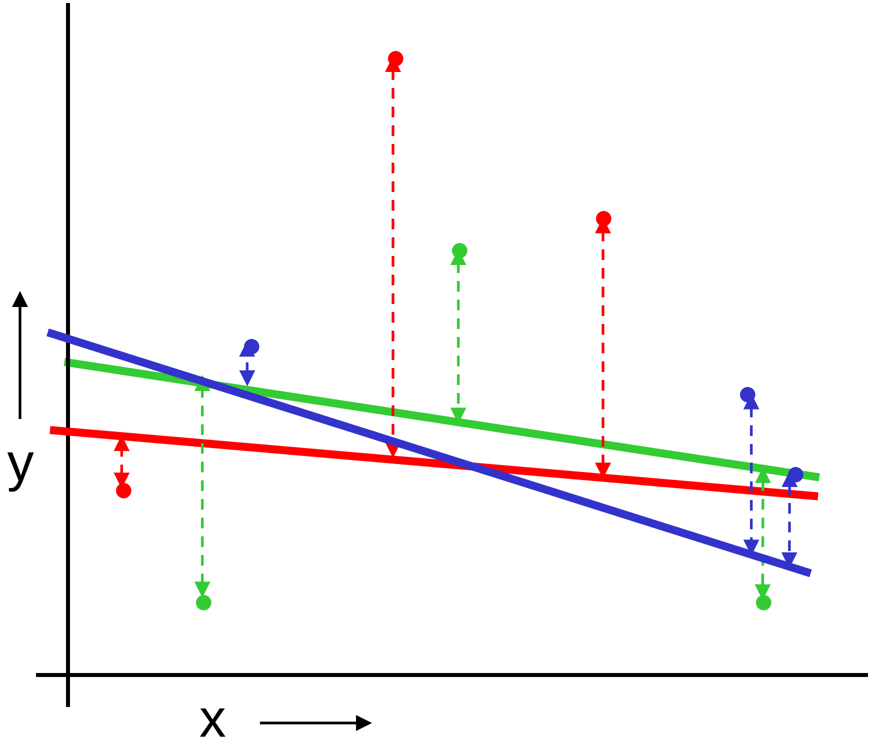


Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

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



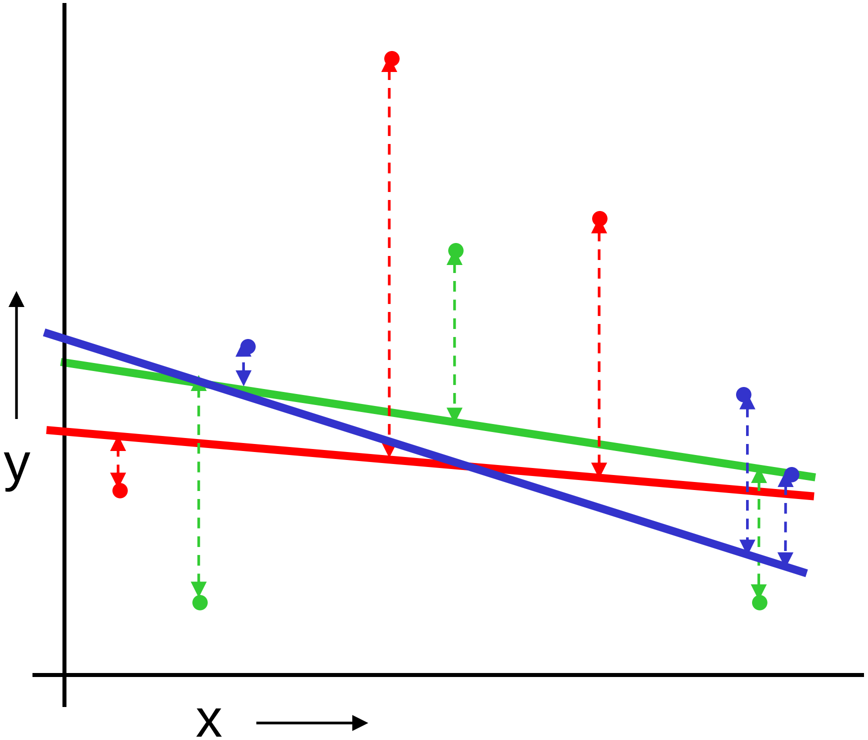
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Red Green and Blue)

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



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)

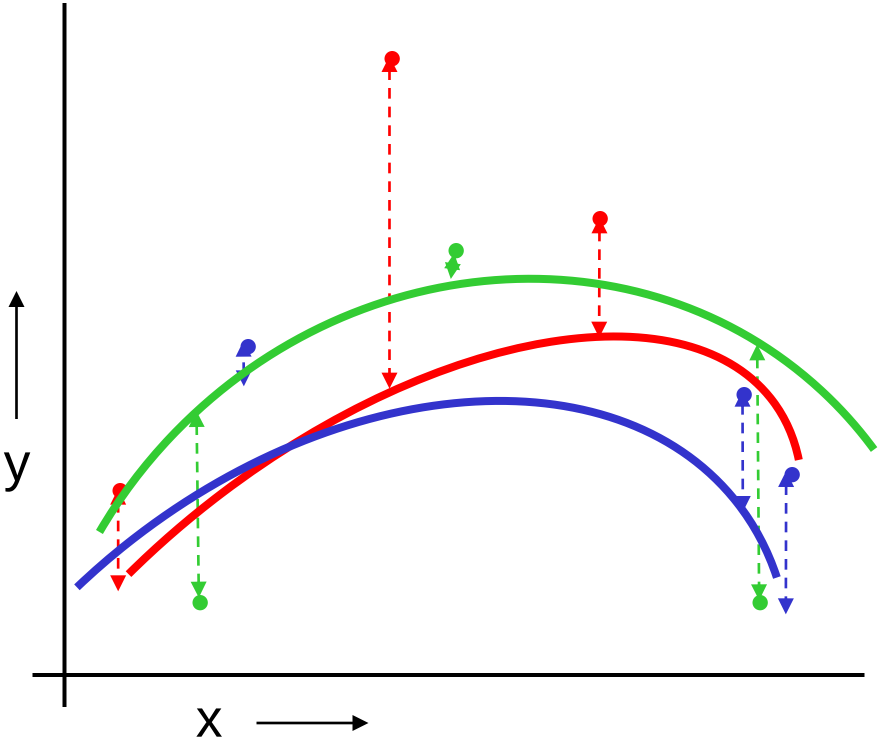
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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)

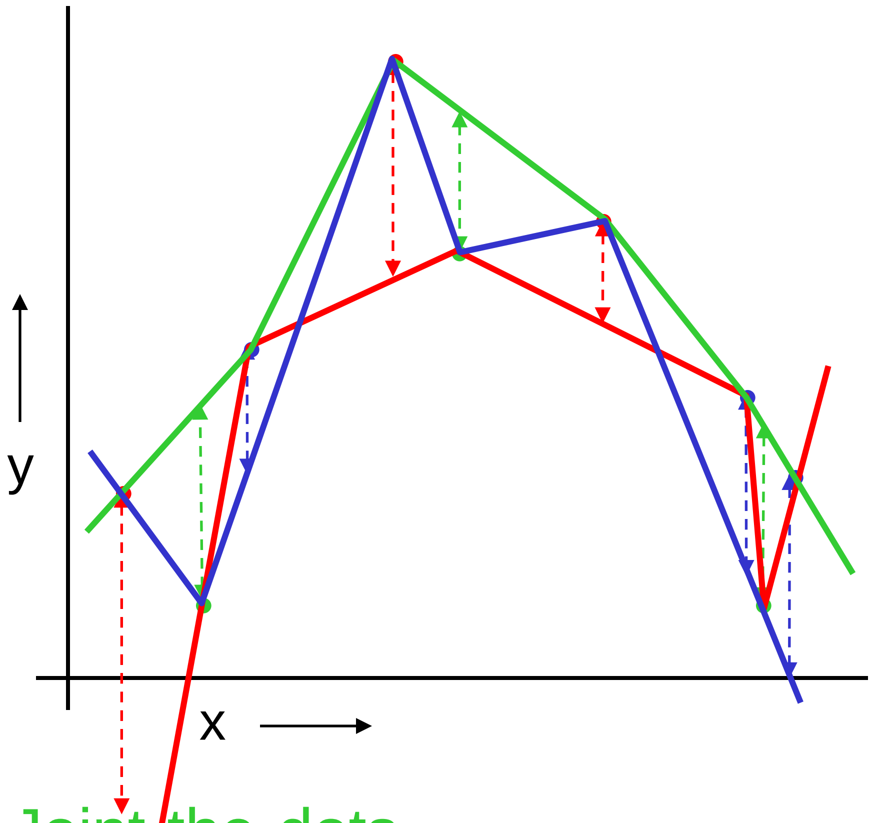
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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

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	