
Overfitting, Validation

Overview of Statistical Machine Learning

Simon Guenter

The overfitting and validation examples are taken from:

Andrew W. Moore

<http://www.cs.cmu.edu/~awm/tutorials>

Overview

What we had so far:

- Given data

Regression (linear,quadratic,exponential...):

- Decide for a type of function parametrized with weights
- Optimize weights to best fit to data

Classification (k-NN,Centroid,regression...):

- Decide for a classifier and select its parameters
- Train classifier

Today:

- Selection of optimal function / classifier

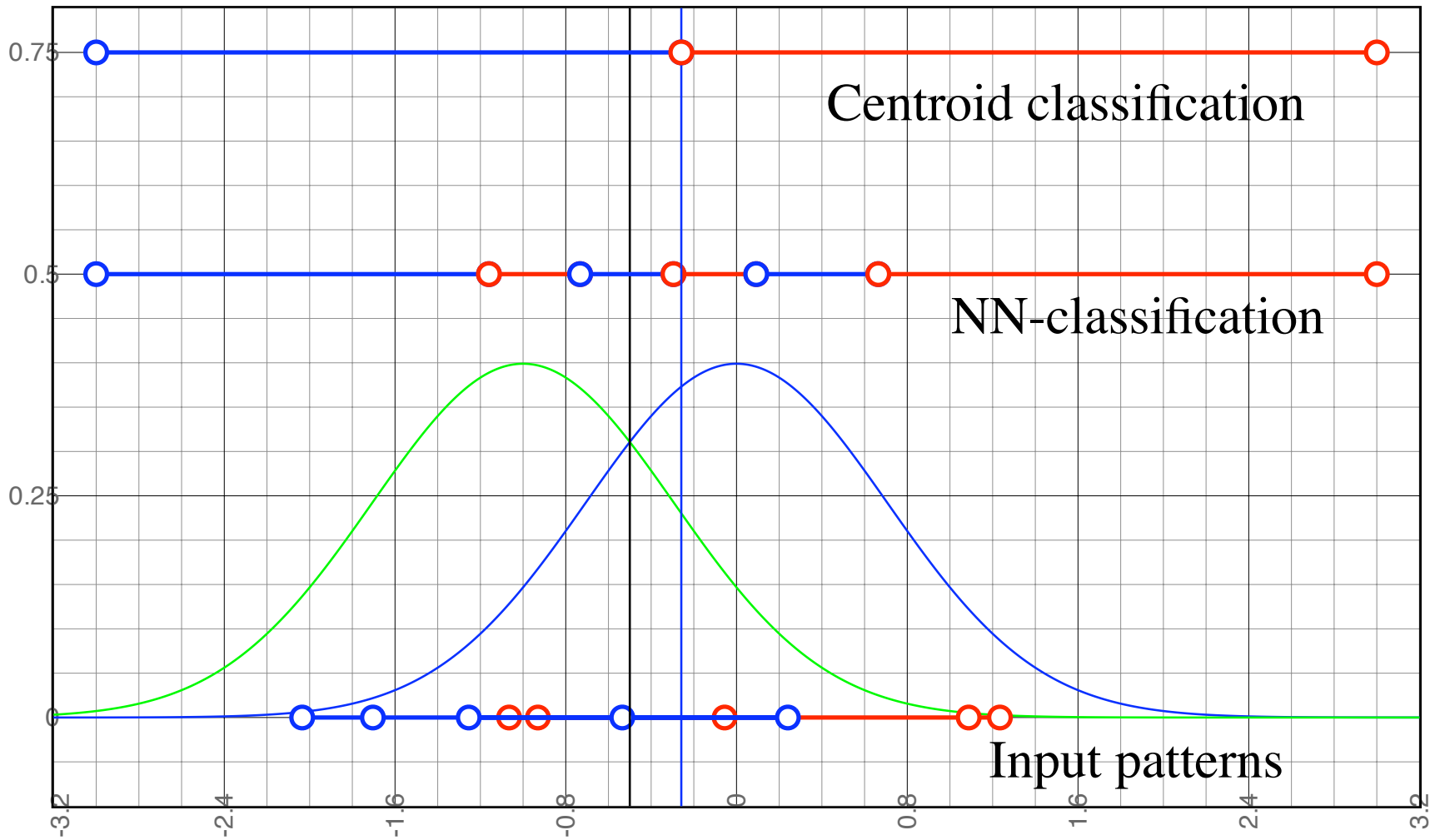
Selection of function/classifier

- ❑ Select “best” function /classifier, yet how measure quality ?
- ❑ We are interested in a model which is good for **new** data
(its purpose is to predict new data)
- ❑ Average error on new data is called generalization error, test error or true risk
- ❑ Error rate on data set is a bad measure for test error
(1-NN has always 0 error on data set!)
- ❑ Overfitting: The error on data set is much lower than test error, i.e. the model is too strongly fit to data

Overfitting example

- ❑ Two Gaussian classes $N(-1,1)$, $N(0,1)$
- ❑ 5 samples per class
- ❑ Classification by 1-NN and centroid classifier
- ❑ Error for 1-NN on data set is 0
- ❑ Error for centroid classifier is 0.3
- ❑ 1-NN is not really better -> see next slide
- ❑ Error rate of centroid classifier is just closer to test error

Overfitting example



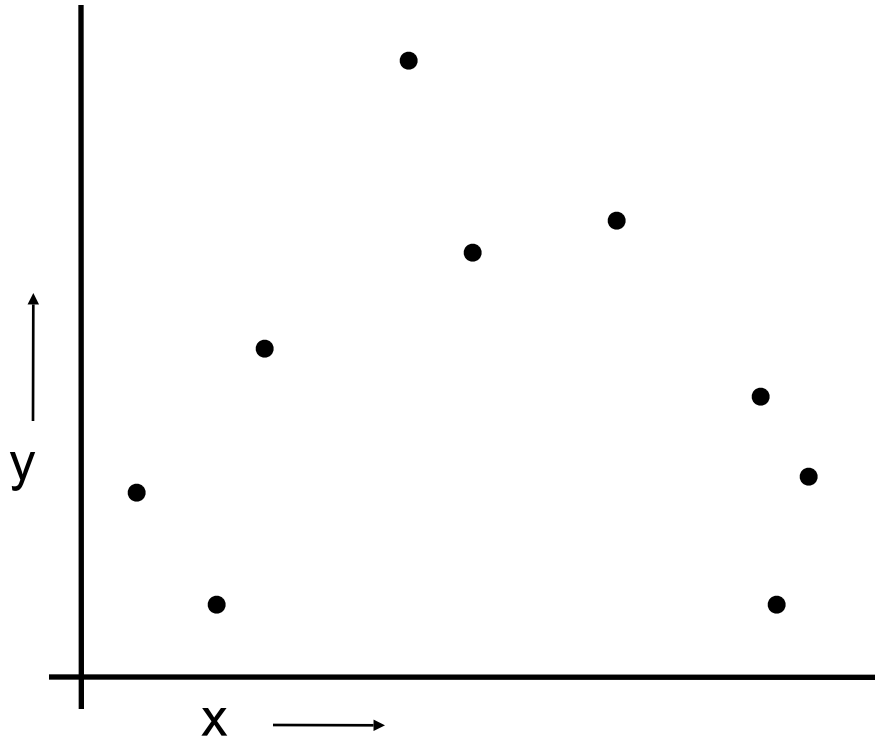
Today's Topics

- ❑ How to solve overfitting problem, i.e. measure test error accurately
- ❑ Problem solved by validation procedure

Validation Procedure:

- ❑ Cross-Validation
- ❑ Analytic validation techniques
- ❑ Demonstrated on an example where we fit three different functions to data (classification is analogous)

Example: A Regression Problem



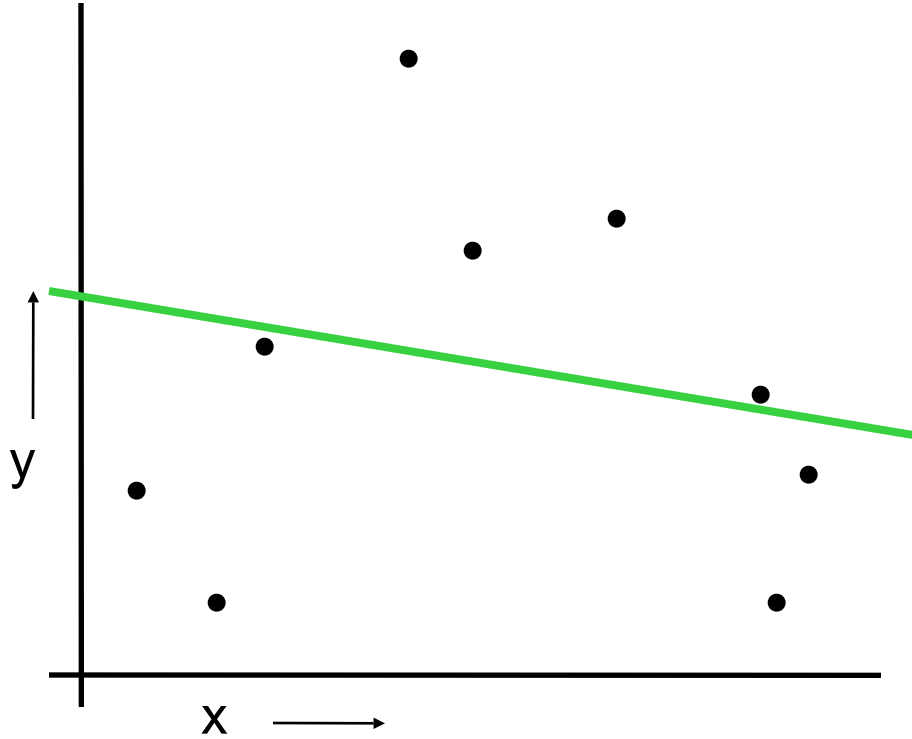
$$y = f(x) + \text{noise}$$

Can we learn f from this data?

Let's consider three methods:

- Linear regression
- Quadratic regression
- Join-the-dots

Linear Regression



Linear Regression

with a constant term:

X	Y
3	7
1	3
⋮	⋮

$$\mathbf{X} = \begin{bmatrix} 3 \\ 1 \\ \vdots \end{bmatrix}$$

$$\mathbf{x}_1 = (3) \dots$$

$$\mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

$$y_1 = 7 \dots$$

$$\mathbf{Z} = \begin{bmatrix} 1 & 3 \\ 1 & 1 \\ \vdots & \vdots \end{bmatrix}$$

$$\mathbf{z}_1 = (1, 3) \dots$$

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

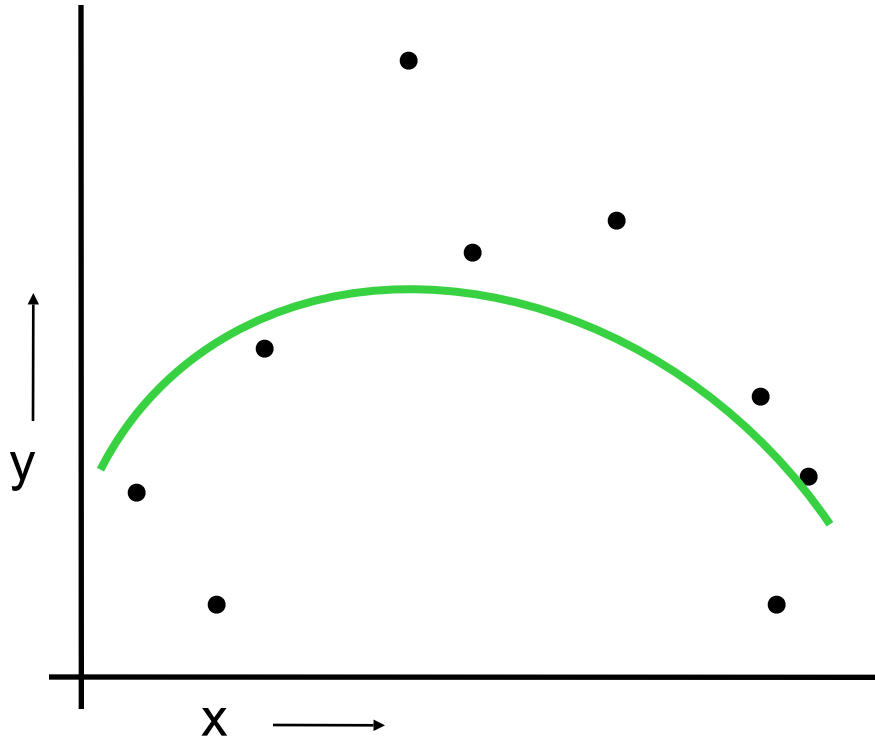
$$\mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

$$y_1 = 7 \dots$$

$$\beta = (\mathbf{Z}^T \mathbf{Z})^{-1} (\mathbf{Z}^T \mathbf{y})$$

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

Quadratic Regression



Quadratic Regression

X	Y
3	7
1	3
⋮	⋮

$$\mathbf{X} = \begin{bmatrix} 3 \\ 1 \\ \vdots \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

$y_1 = 7..$

$$\mathbf{Z} = \begin{bmatrix} 1 & 3 & 9 \\ 1 & 1 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

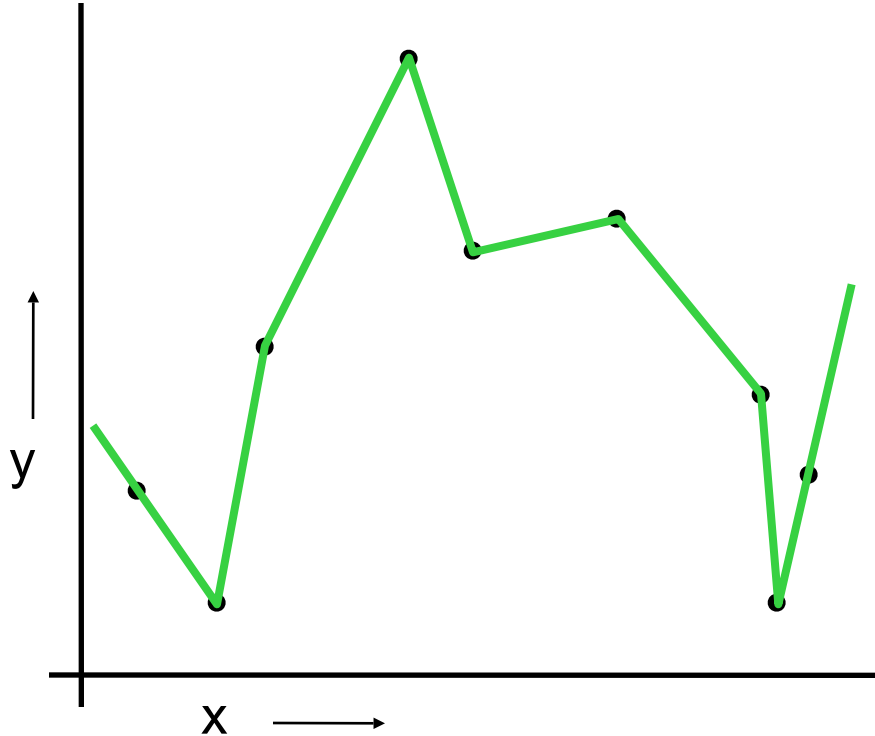
$$\mathbf{z} = (1, x, x^2)$$

$$\mathbf{y} = \begin{bmatrix} 7 \\ 3 \\ \vdots \end{bmatrix}$$

$$\beta = (\mathbf{Z}^T \mathbf{Z})^{-1} (\mathbf{Z}^T \mathbf{y})$$

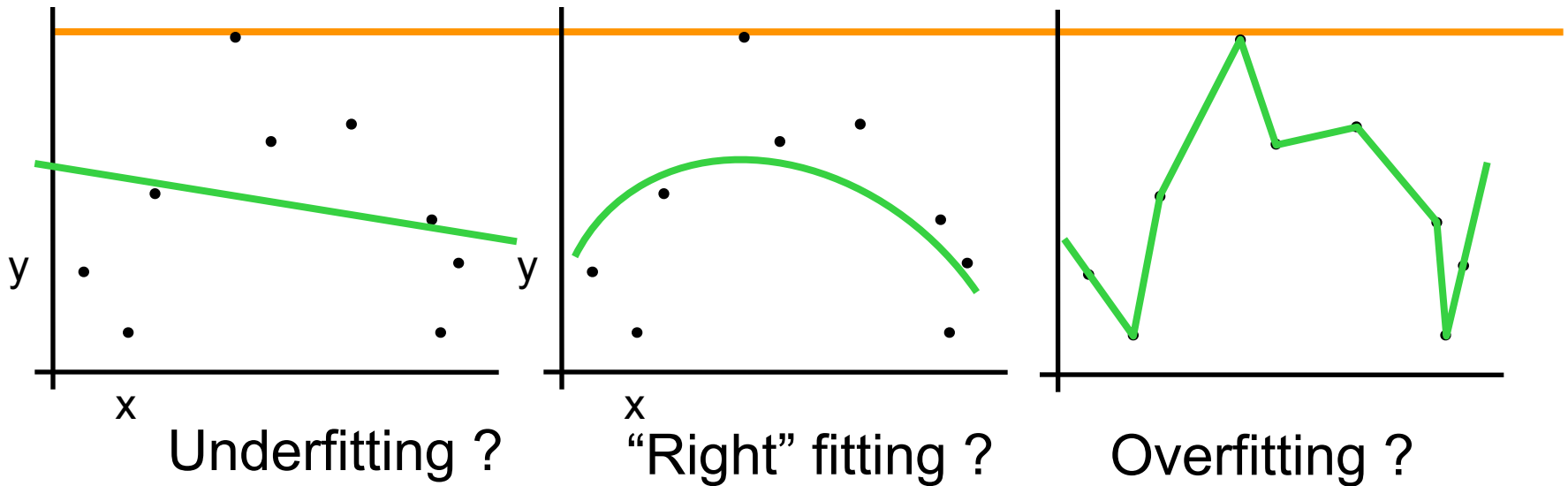
$$y^{est} = \beta_0 + \beta_1 x + \beta_2 x^2$$

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?

→ overfitting

The key question is: How well are you going to predict future data drawn from the same distribution?

Empirical Risk vs. True Risk

Risk = expected value of the loss L

Empirical risk = average value of L for given set of data.
Minimizing L over training data = empirical risk minimization (ERM)

True risk = expected value of L over true density of data:

$$R(w) = \int L(y, f(\mathbf{x}, w)) p(\mathbf{x}, y) d\mathbf{x} dy$$

The problem here: We do not know the probability density function $p(\mathbf{x}, y)$ that created our data. So we cannot compute the true risk. → Estimate the true risk

Maximum likelihood leads to specific loss functions for density estimation, regression, and classification.

Estimation of the True Risk

In **Validation**, the true risk is estimated.

Two different approaches for validation:

- Resampling (today):

 - uses given data

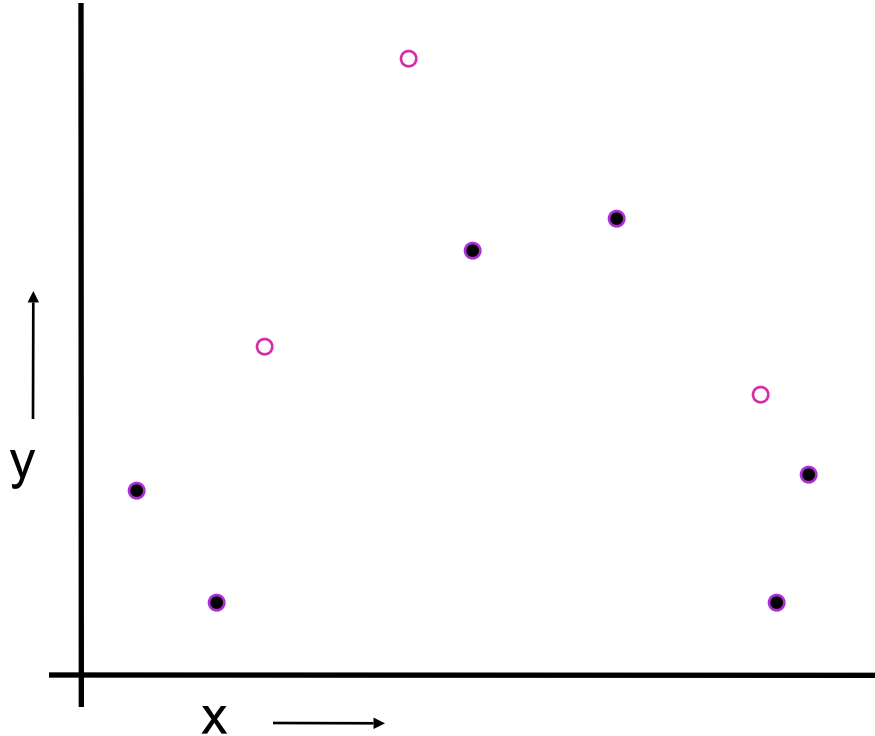
 - e.g., Test set method, Cross-Validation

- Analytic (not covered in course):

 - estimates analytically the true risk (realm of statisticians), e.g., AIC, BIC

 - depends on model complexity (VC-dimension)

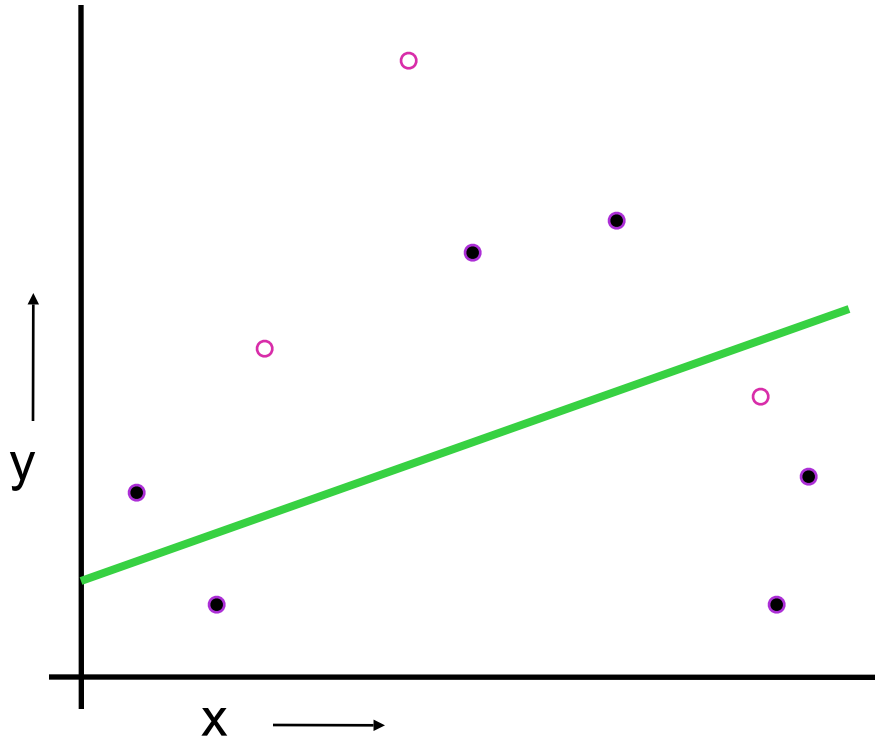
The test set method



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

2. The remainder is a **training set**

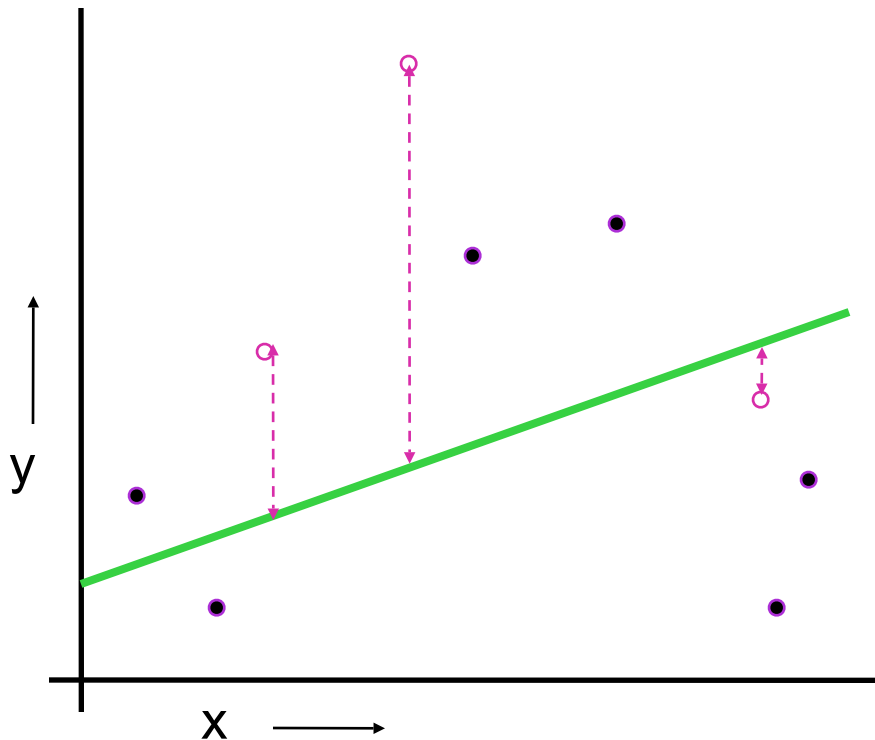
The test set method



(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

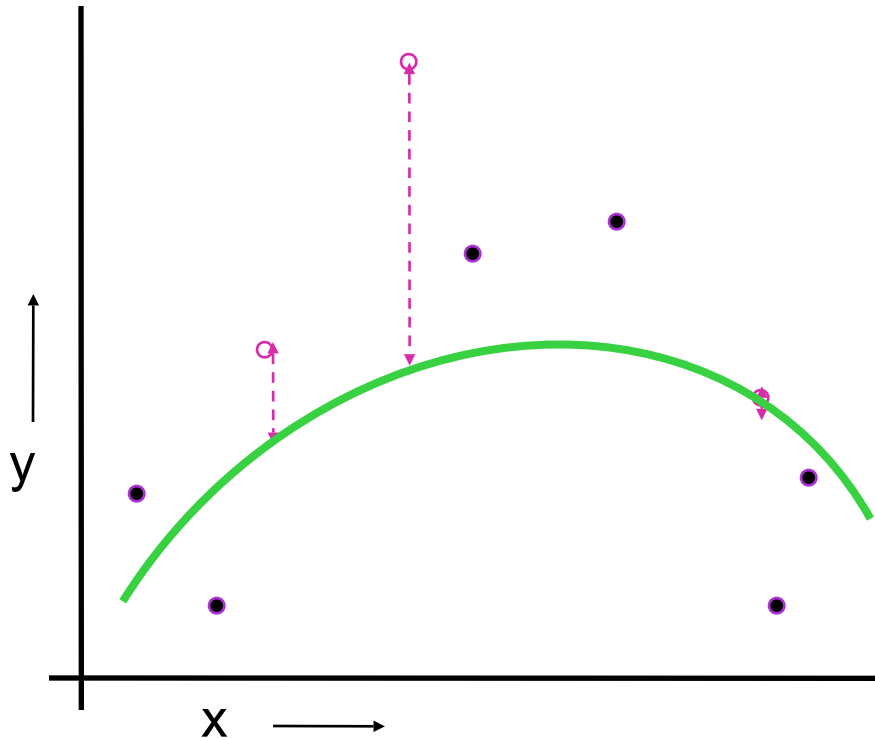


(Linear regression example)

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

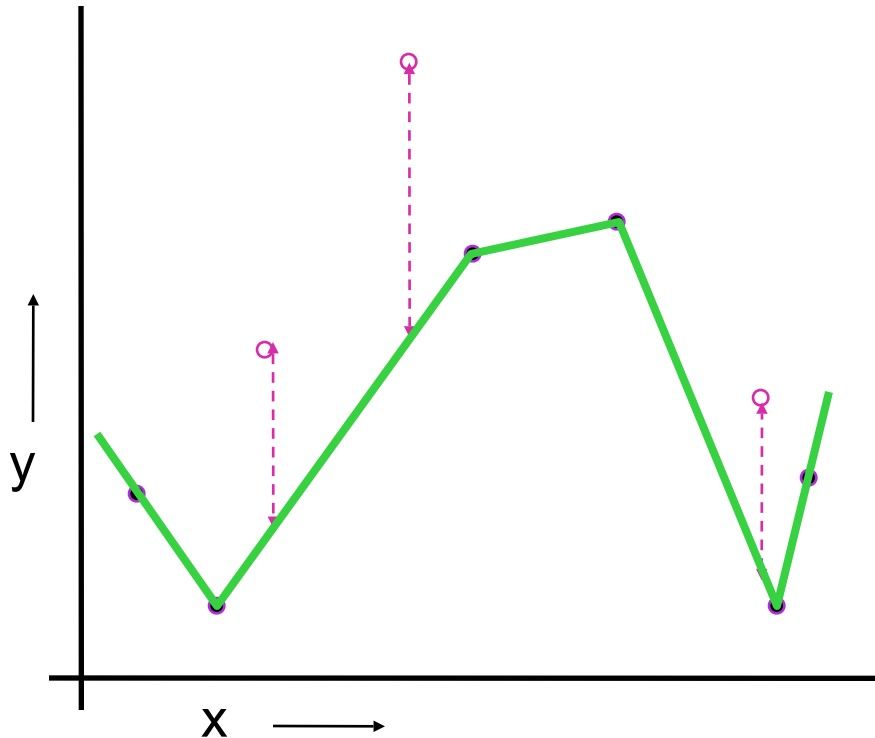


(Quadratic regression example)

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

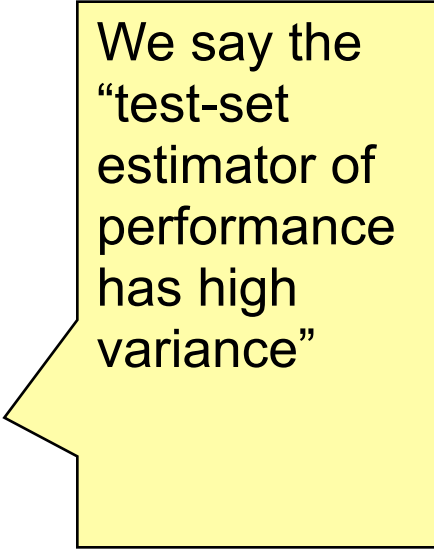
The test set method

Good news:

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

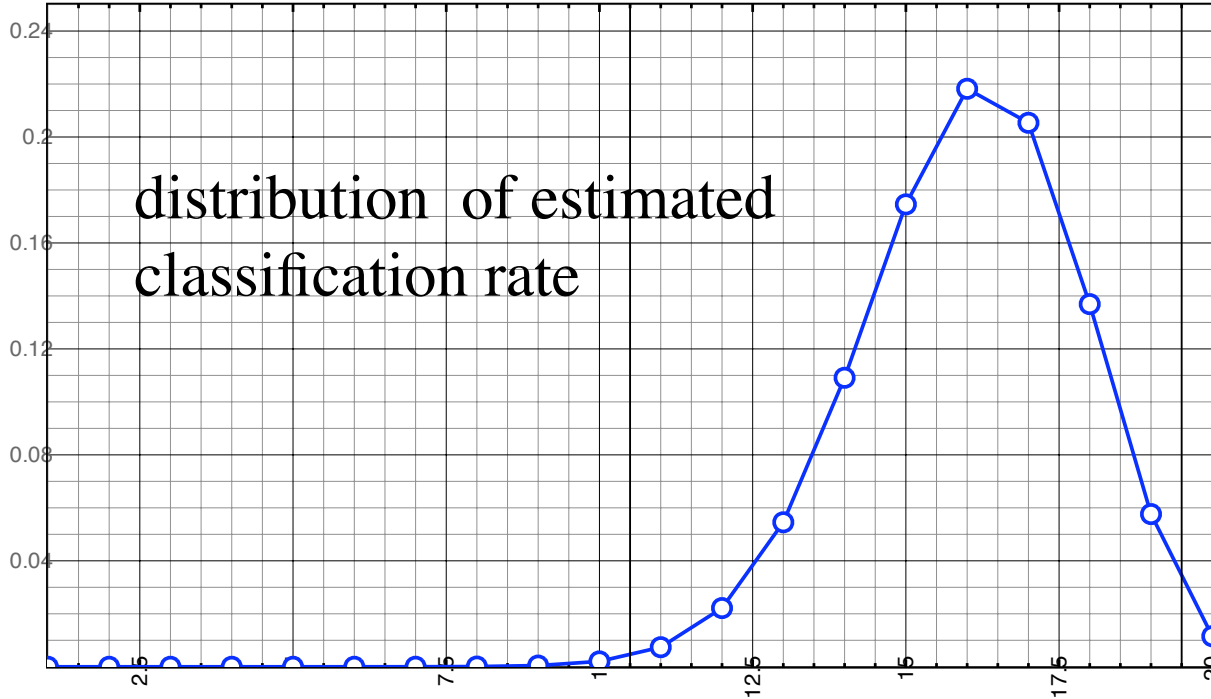
Bad news:

- 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”

High variance on test set



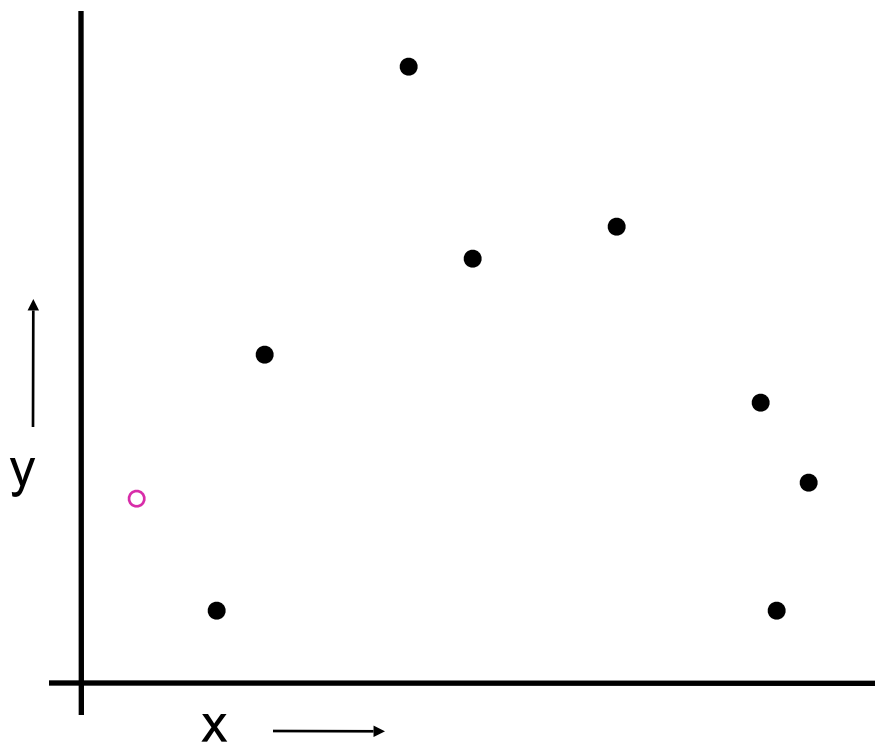
20 test patterns

true risk 0.2

Only error rate 0 (20/20) and rates higher and equal to 0.5 (10/20) are statistically significant* different to error rate 0.2 (4/20)

**significance level 5%*

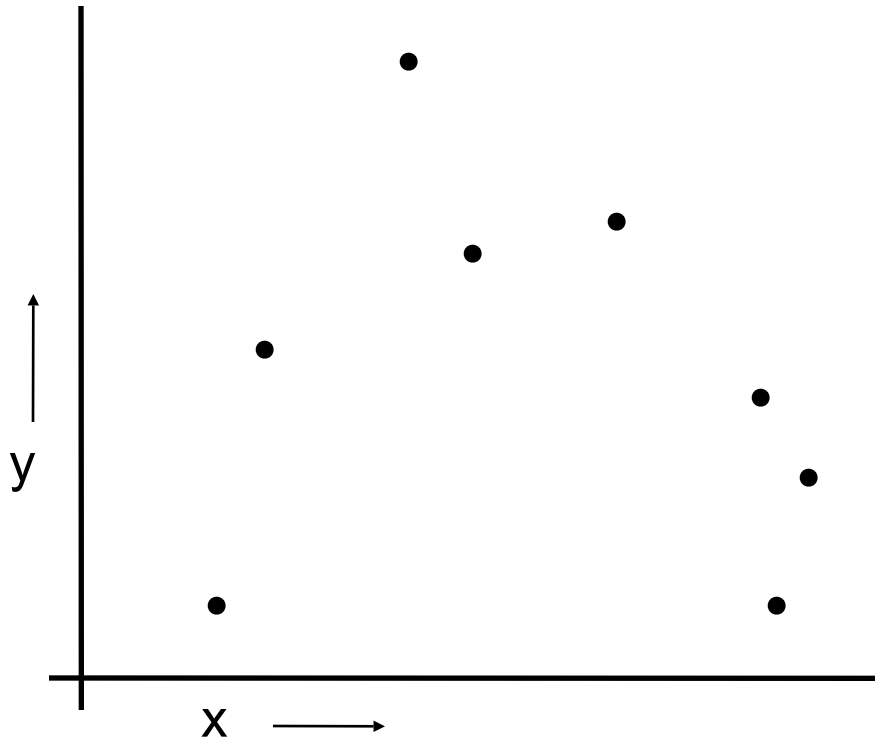
LOOCV (Leave-one-out Cross Validation)



For $k=1$ to R

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

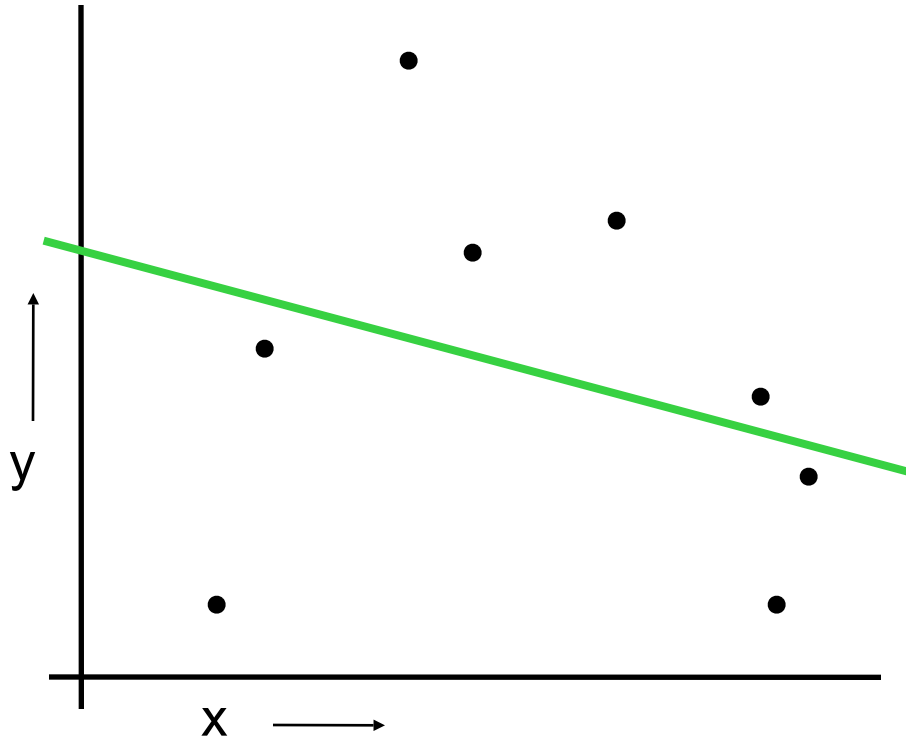
LOOCV (Leave-one-out Cross Validation)



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

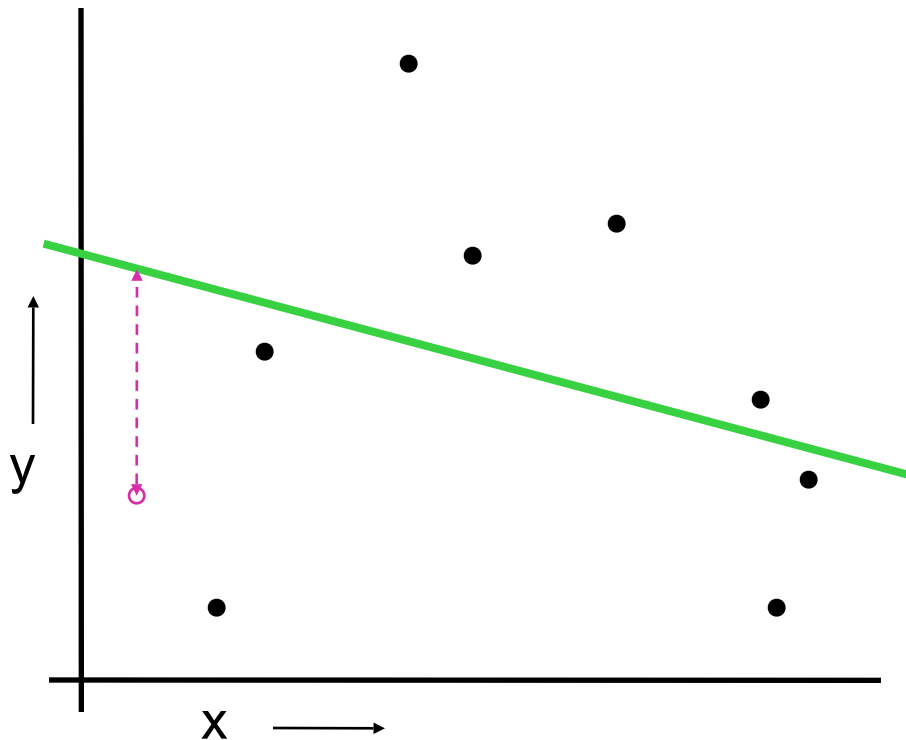
LOOCV (Leave-one-out Cross Validation)



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

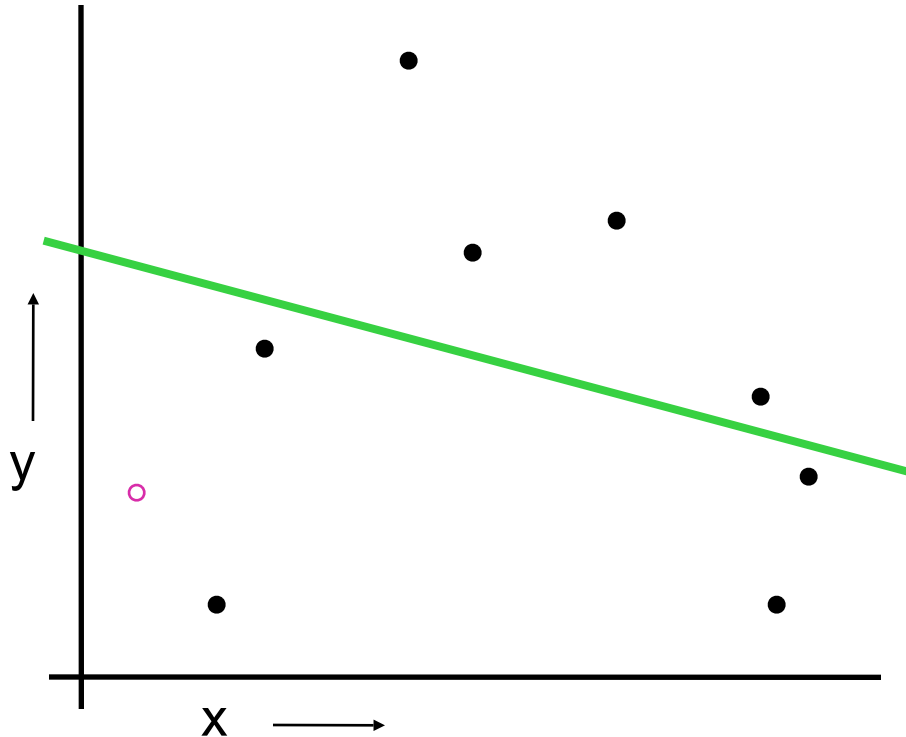
LOOCV (Leave-one-out Cross Validation)



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)

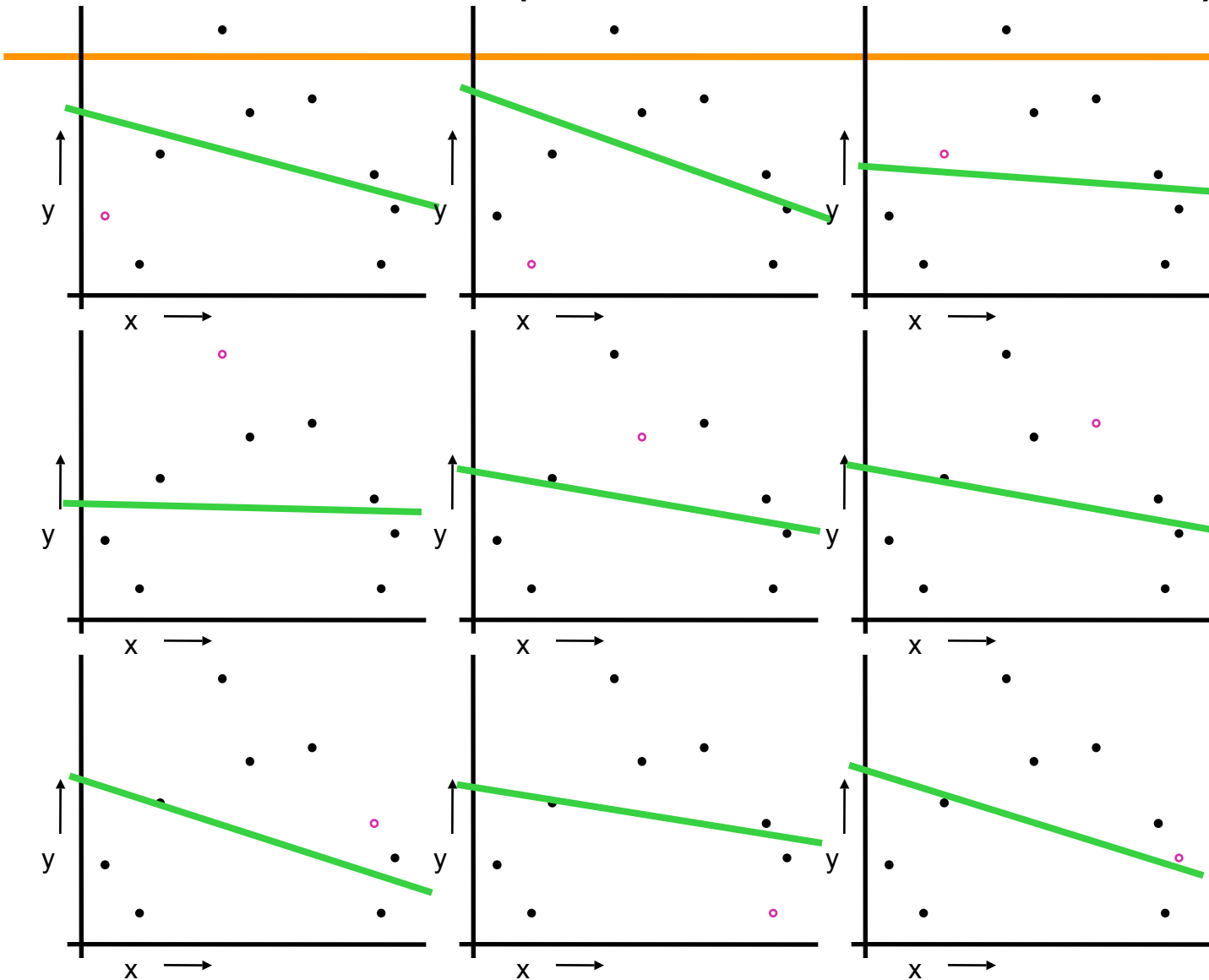


For $k=1$ to R

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)



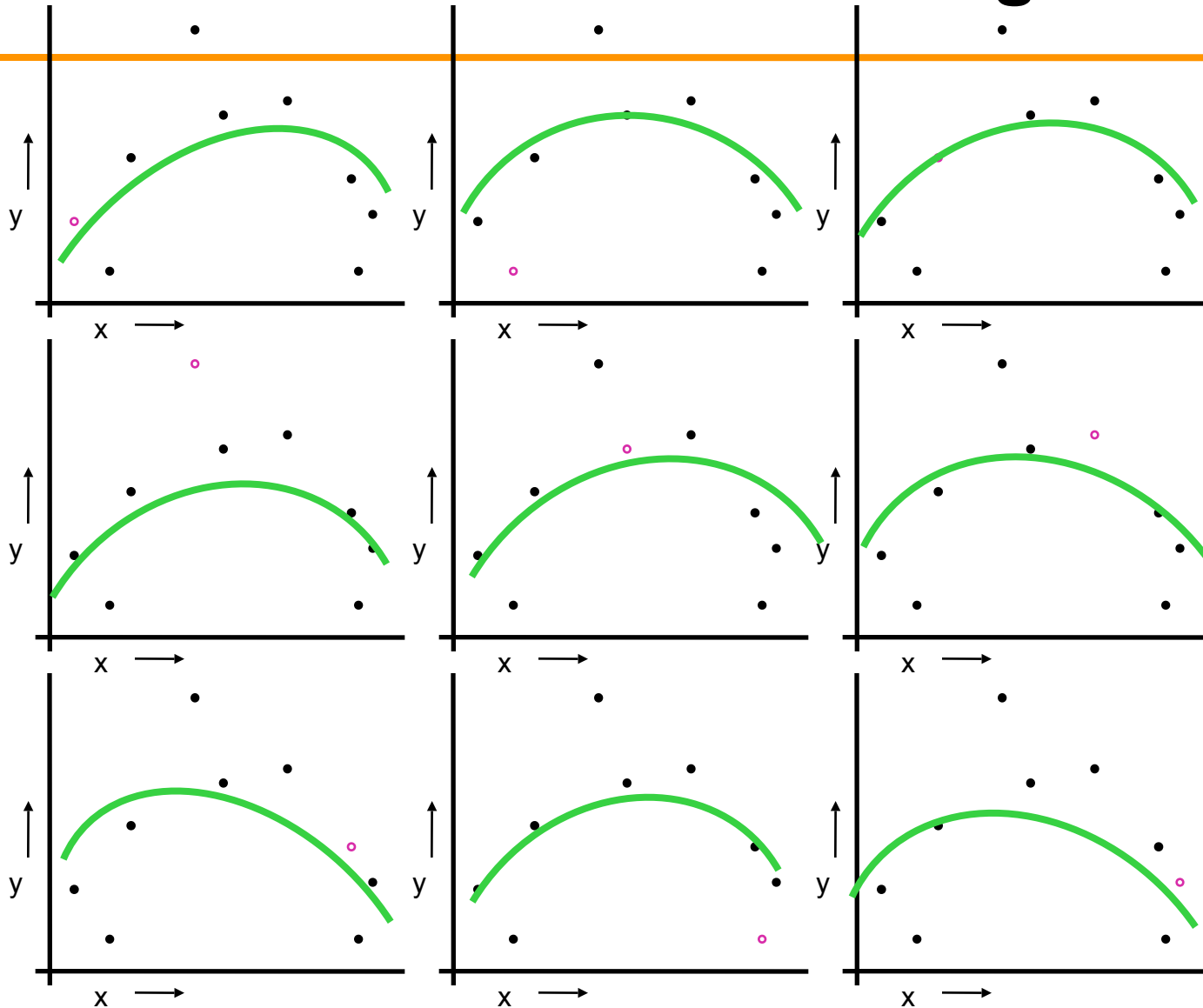
For $k=1$ to R

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



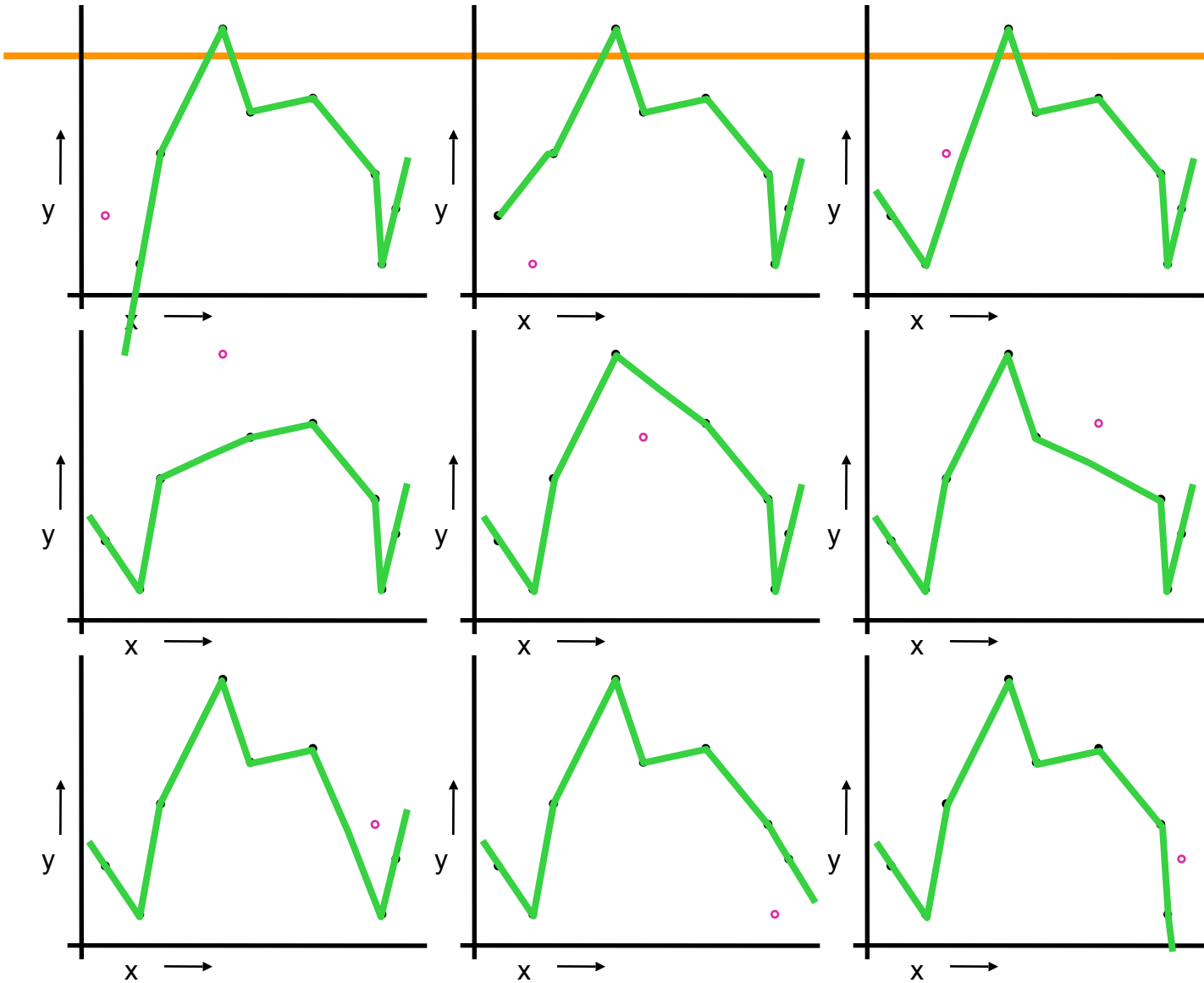
For $k=1$ to R

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.

$$MSE_{LOOCV} = 3.33$$

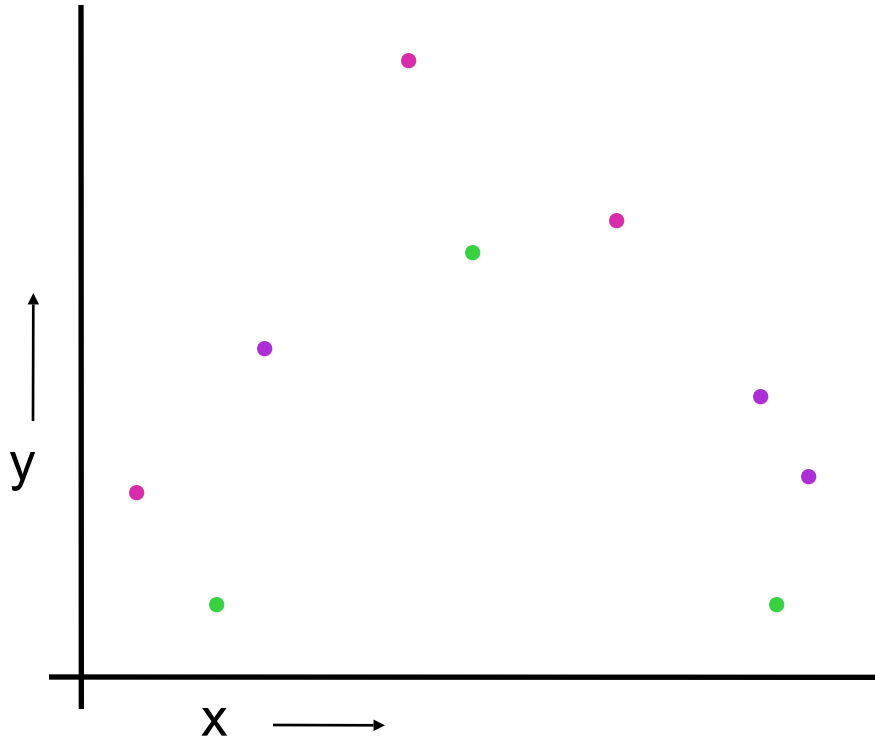
Which kind of Cross Validation?

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

..can we get the best of both worlds?

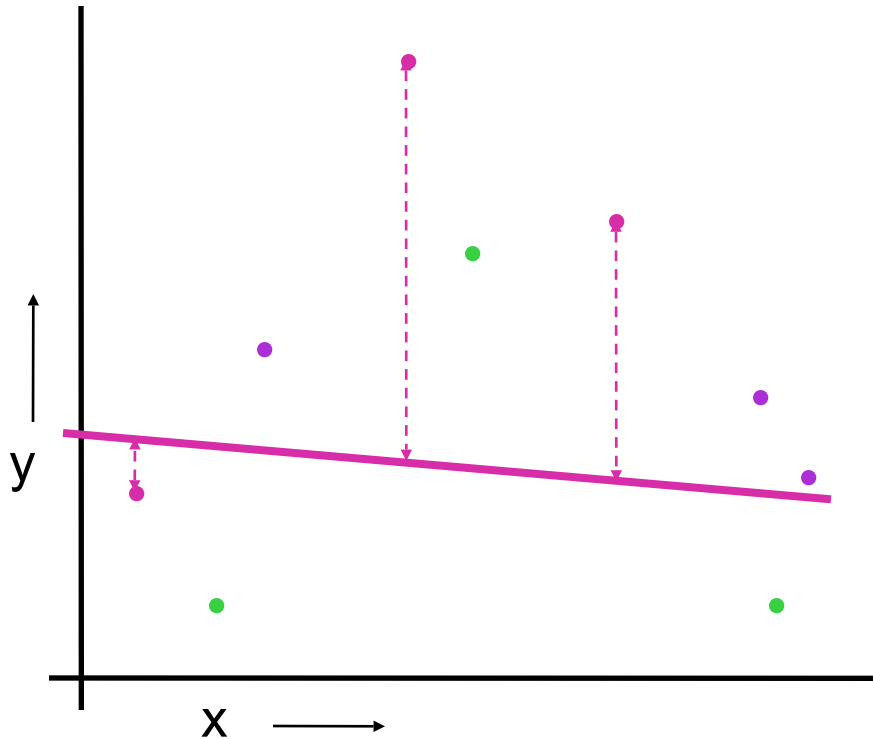
k-fold Cross Validation

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

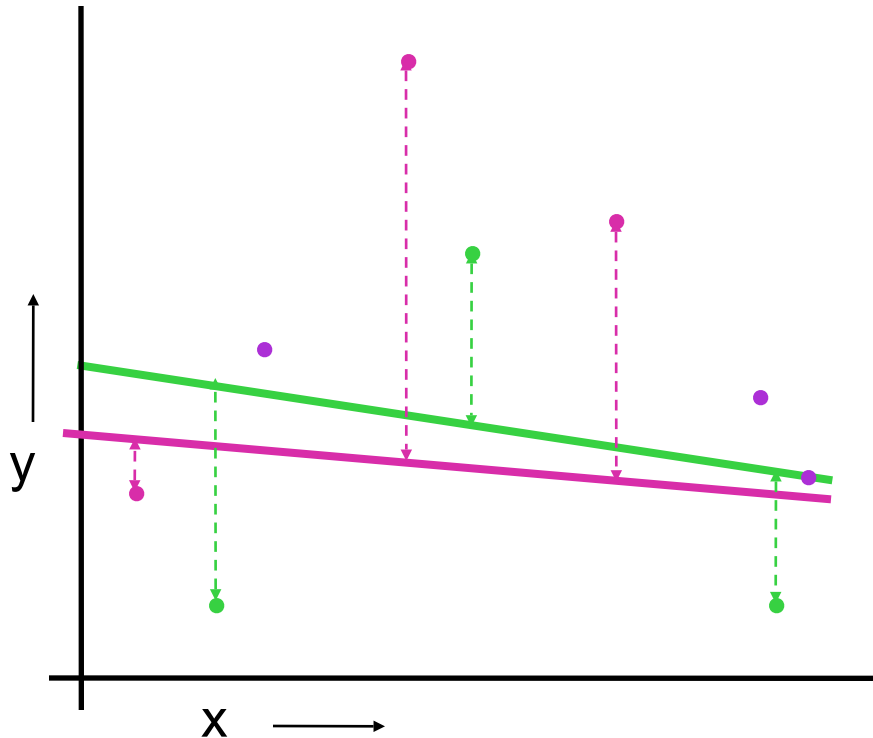
Randomly break the dataset into k partitions (in our example we'll have $k=3$ partitions colored Red Green and Blue)



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)

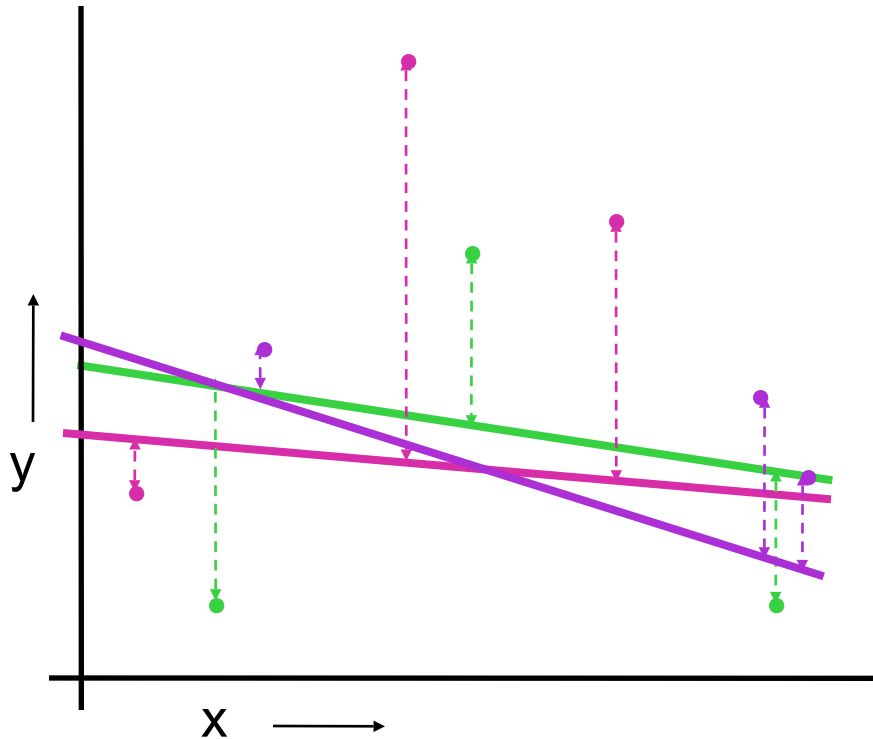


For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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)



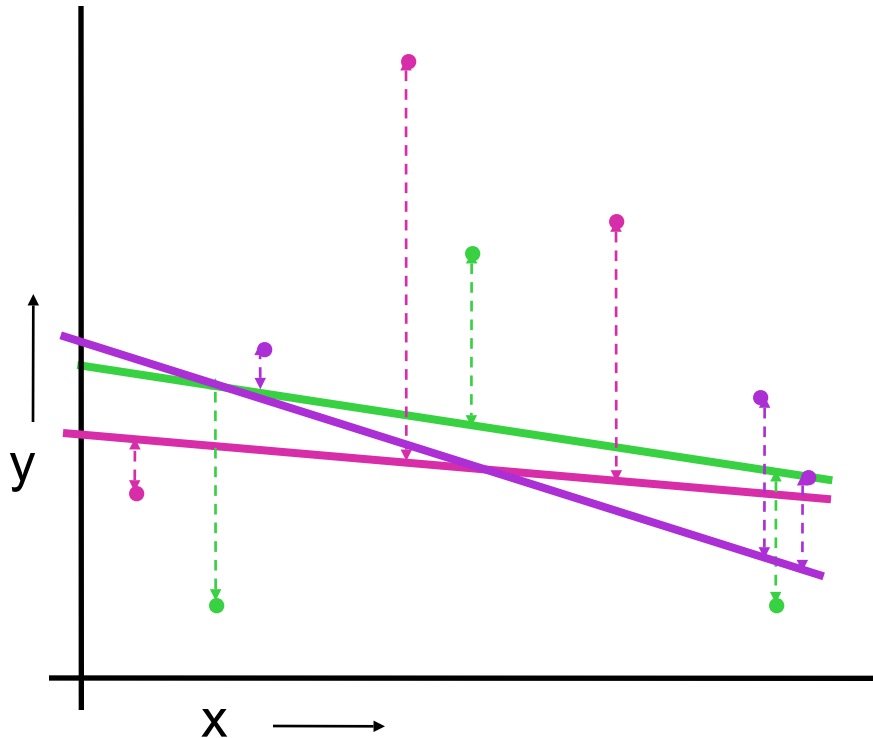
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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.

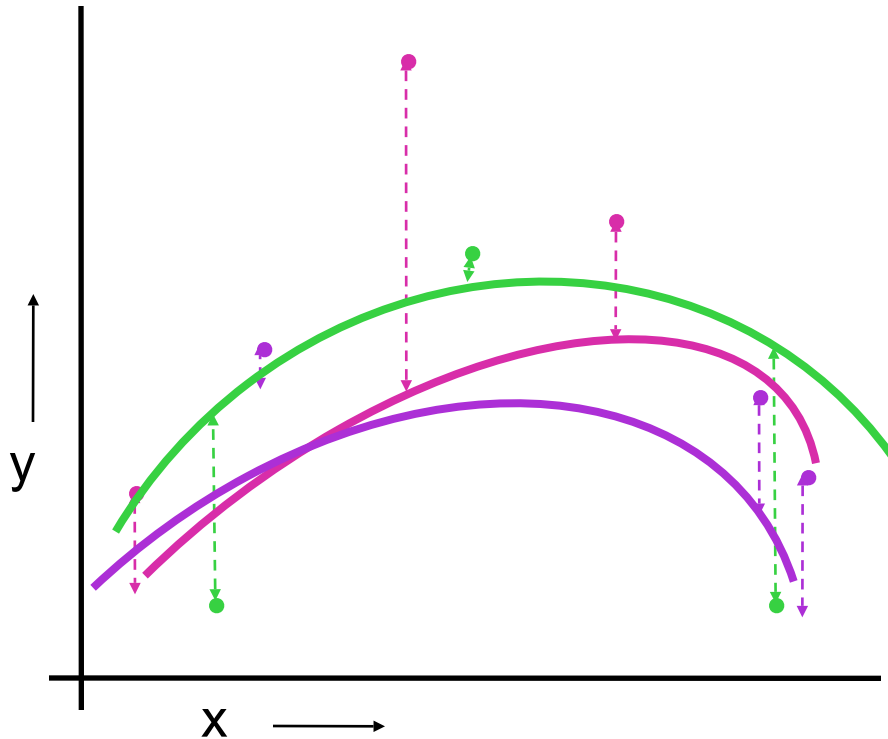
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

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.

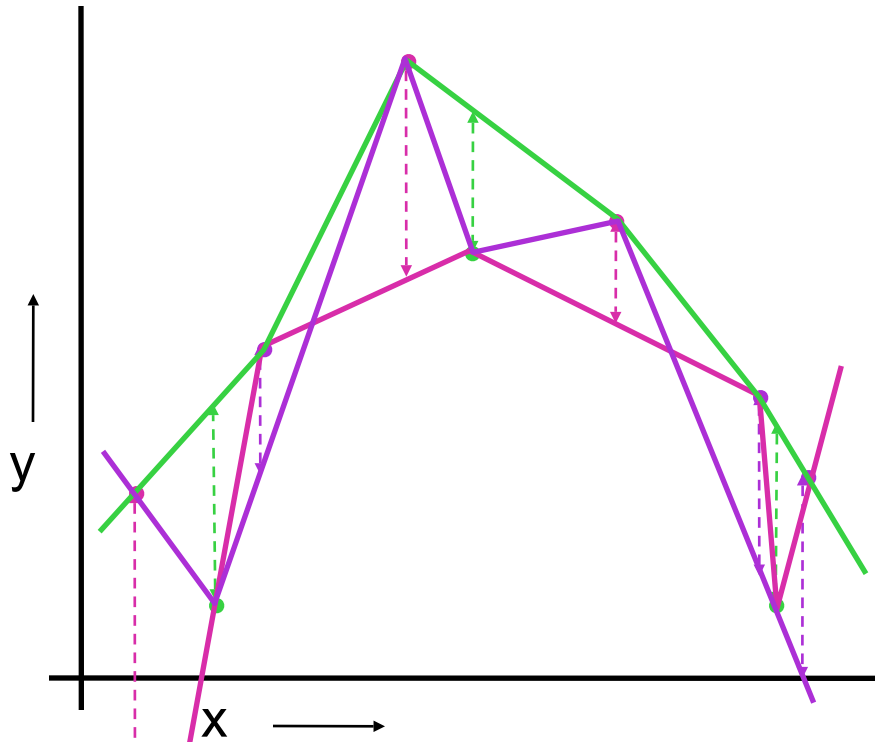
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

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)



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

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.	Wastes $1/R$ data
7-fold	Wastes $1/7$ of the data. 7 times more expensive than test set	Only wastes $1/7$. Only 7 times more expensive instead of R times.
3-fold	Wastes $1/3$ of the data.	Quite cheap (3 times more expensive instead of R times).
R-fold	Identical to Leave-one-out	

CV-based Model Selection

Decide for a CV technique, e.g., 3-fold CV:

Linear Regression $MSE_{3FOLD}=2.05$

Quadratic Regression $MSE_{3FOLD}=1.11$

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

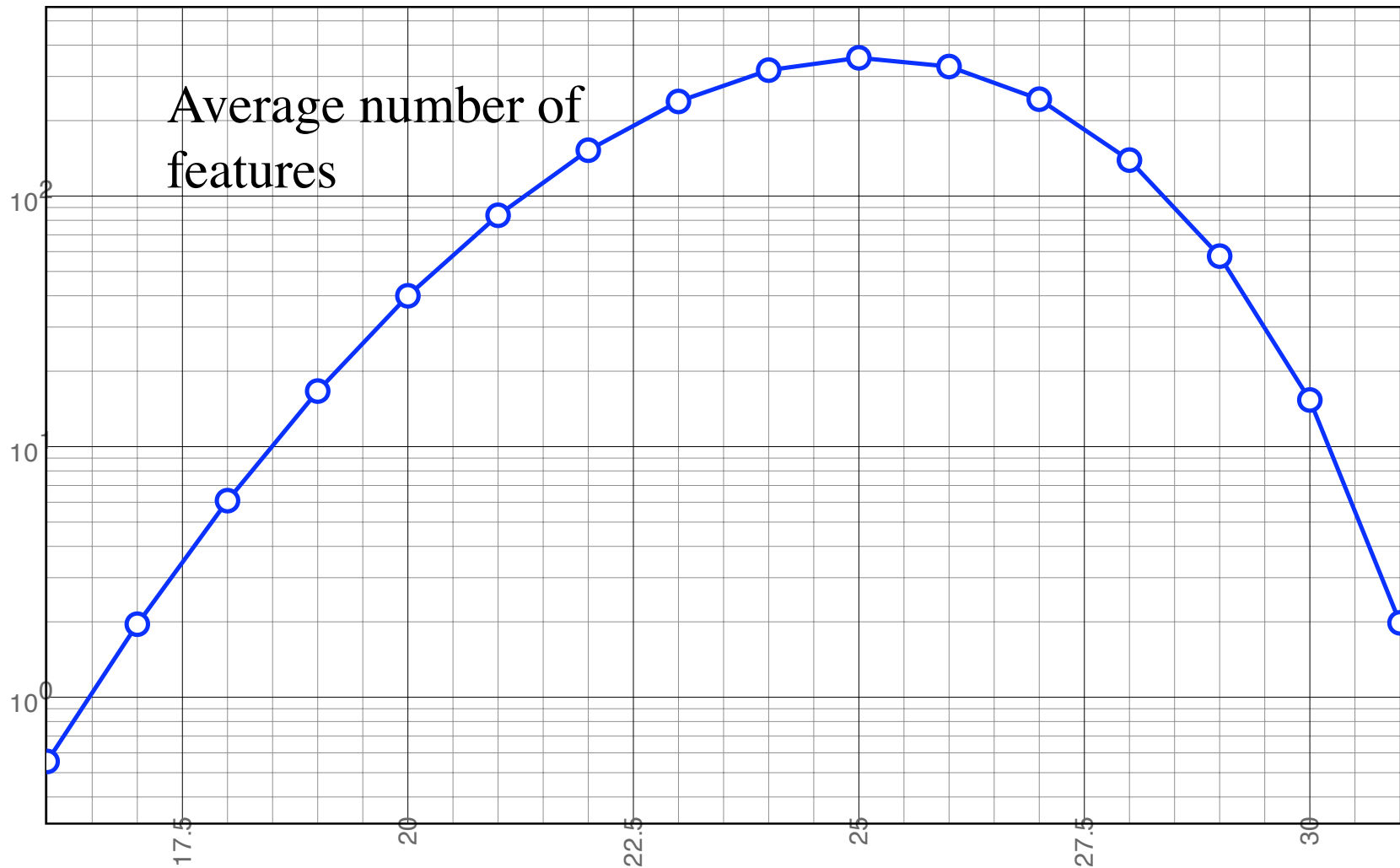
→ Quadratic Regression is the best method.

Note that for all CV methods, the quadratic regression showed the lowest MSE compared with the other two methods.

Problematic cases for Cross-validation

- ❑ CV works not very well if the data set is small and the number of parameters/ functions is large
- ❑ Example: Micro-array data of cancer cells
- ❑ 31 patterns, 2000 dimensional input space (2000 features)
- ❑ Selection of best feature with CV
- ❑ Number of tested patterns when using CV is always number of data patterns (31 in this case)
- ❑ Assumption: Each feature leads to 0.2 error rate

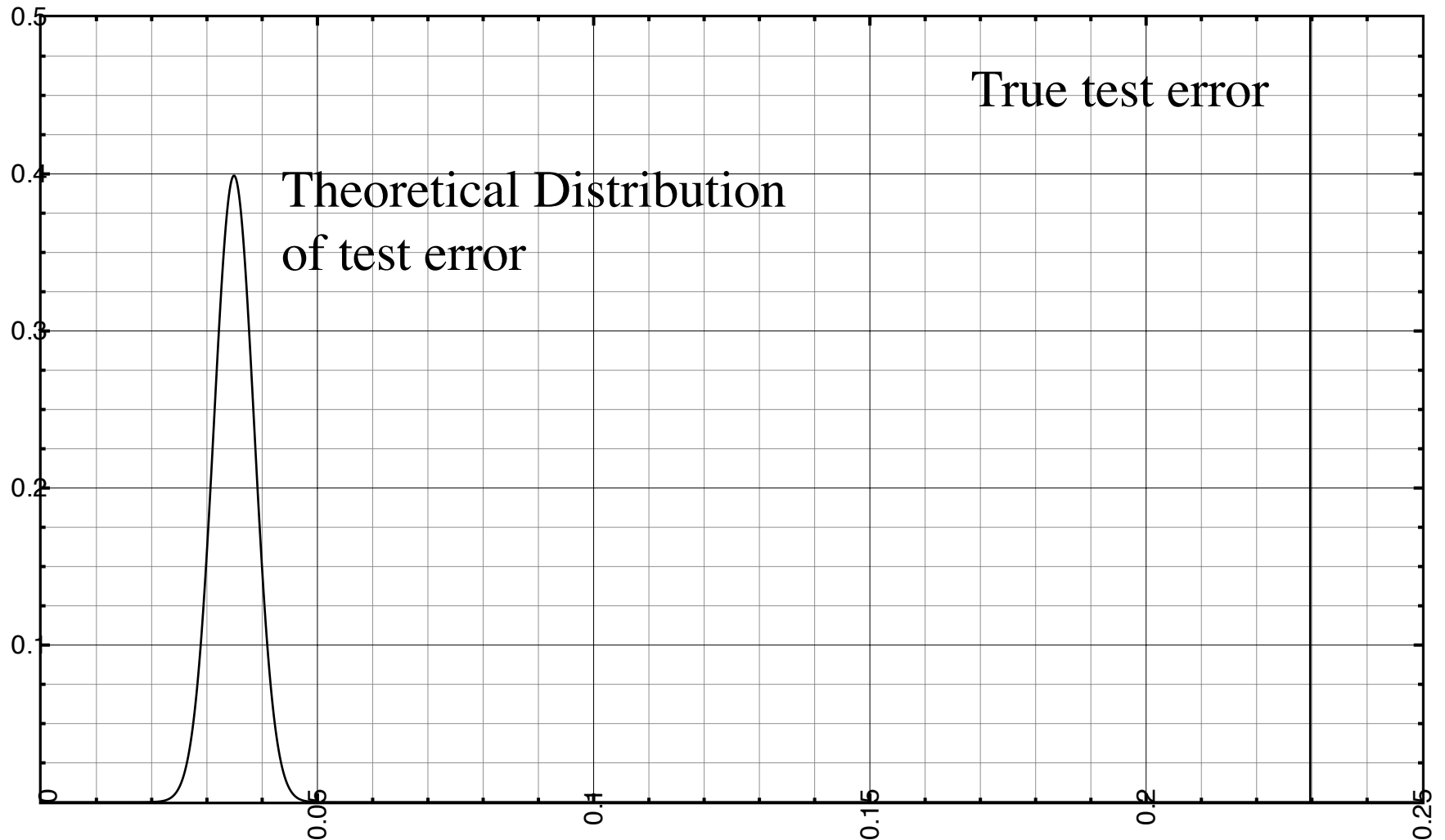
Performance of features



CV failure example

- ❑ We get in average about 2 features with perfect performance
- ❑ Yet true error is 20 % -> CV overestimates 20 %
- ❑ CV doesn't solve everything
- ❑ Other approach: 3-leave-out (all permutations of using 3 patterns as validation set)
- ❑ Number of patterns tested is now 13485 (maximum score)
- ❑ Validation error is 3.49 %
- ❑ Test error (measured on separate data set) is 22.97 %

leave-3-out failure example



leave-3-out failure example

- ❑ Test error is more than 50 standard deviation away => systematical error
- ❑ Overlap in validation sets => Independence assumption violated
- ❑ Conclusion: CV doesn't work well for applications with small data sets and high number of parameters