UVA CS 4774: Machine Learning

Lecture 6: Model Selection

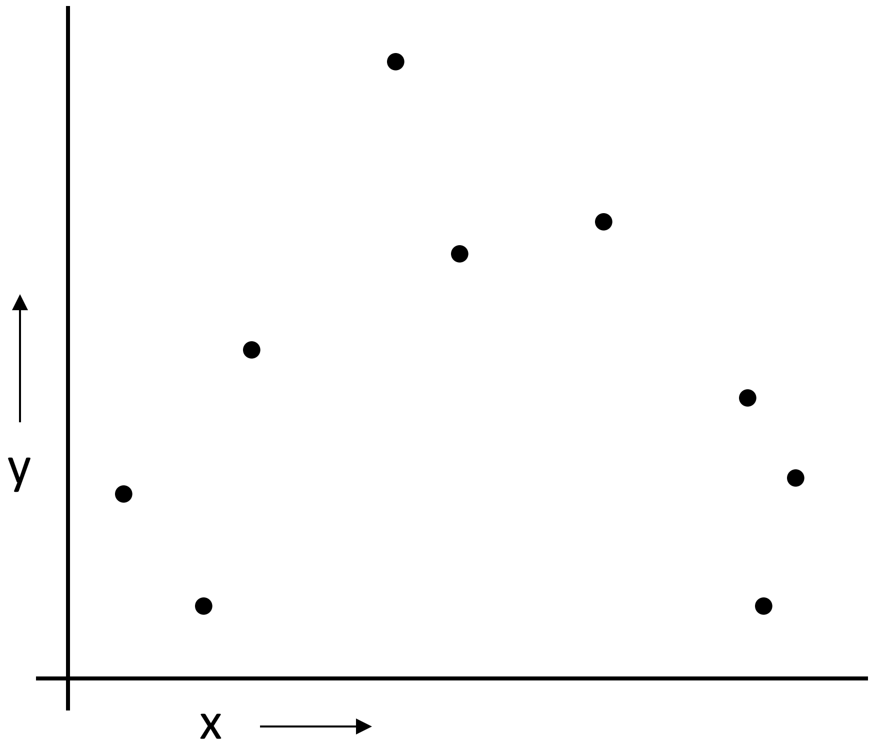
Dr. Yanjun Qi

University of Virginia
Department of Computer Science

Main issues: Model Selection

- How to select the right model type? How to select hyperparameter for a model type?
  - E.g. what polynomial degree d for polynomial regression
  - E.g., where to put the centers for the RBF kernels? How wide?
  - E.g. which basis type? Polynomial or RBF?

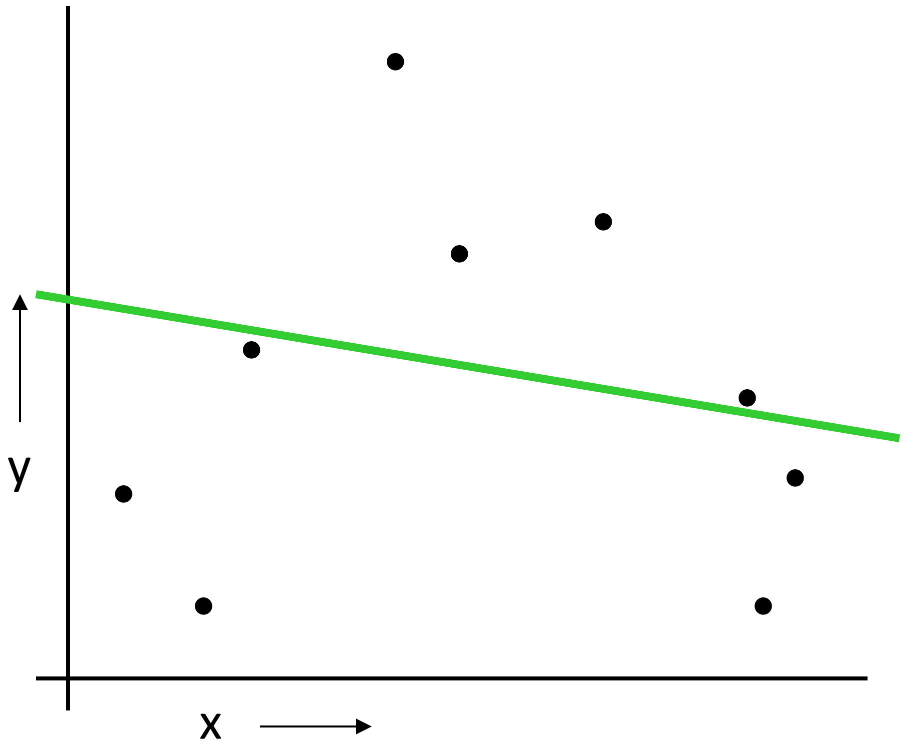
To Avoid: Overfitting or Underfitting



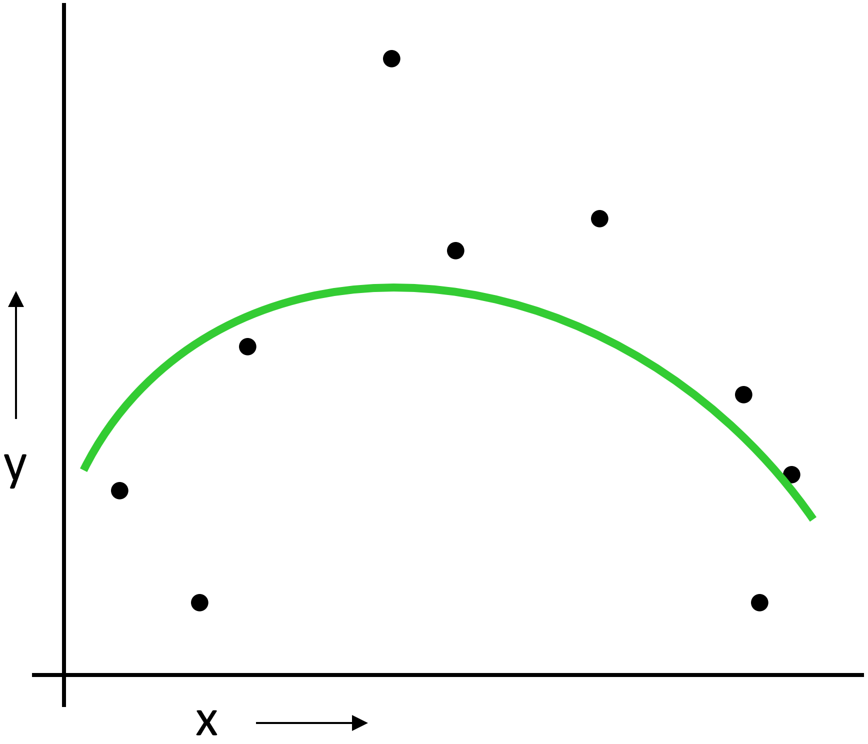
Can we learn a regression f from the data?

Let's consider three methods…

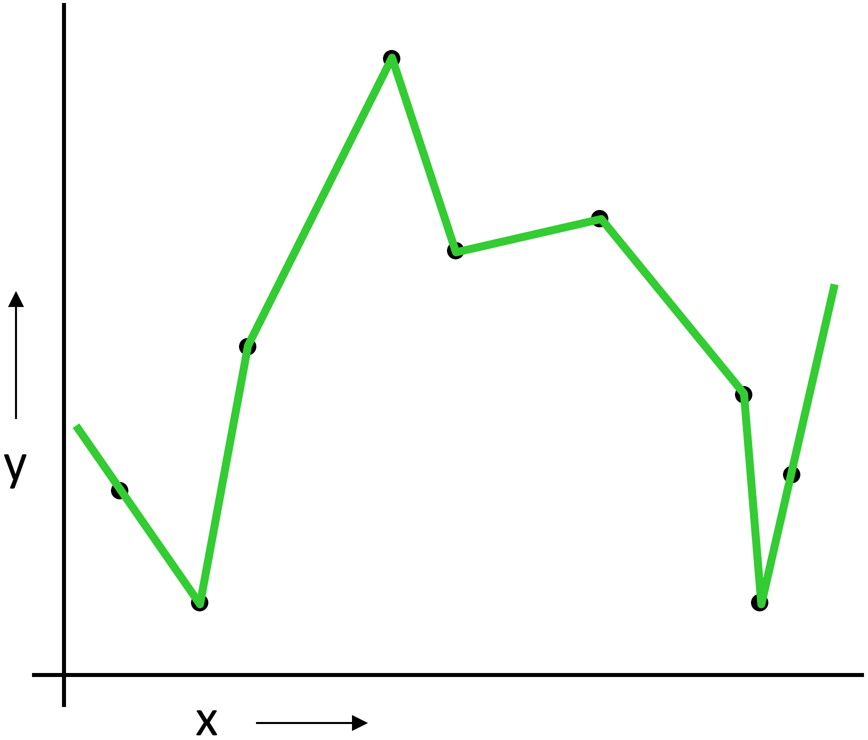
Linear Regression



Quadratic Regression

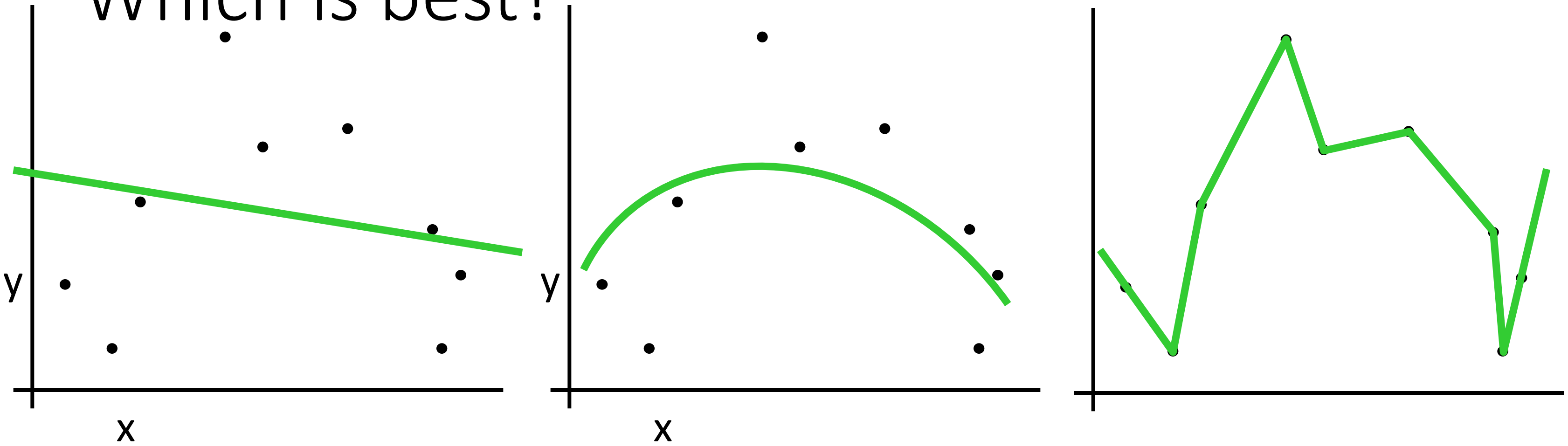


Join-the-dots



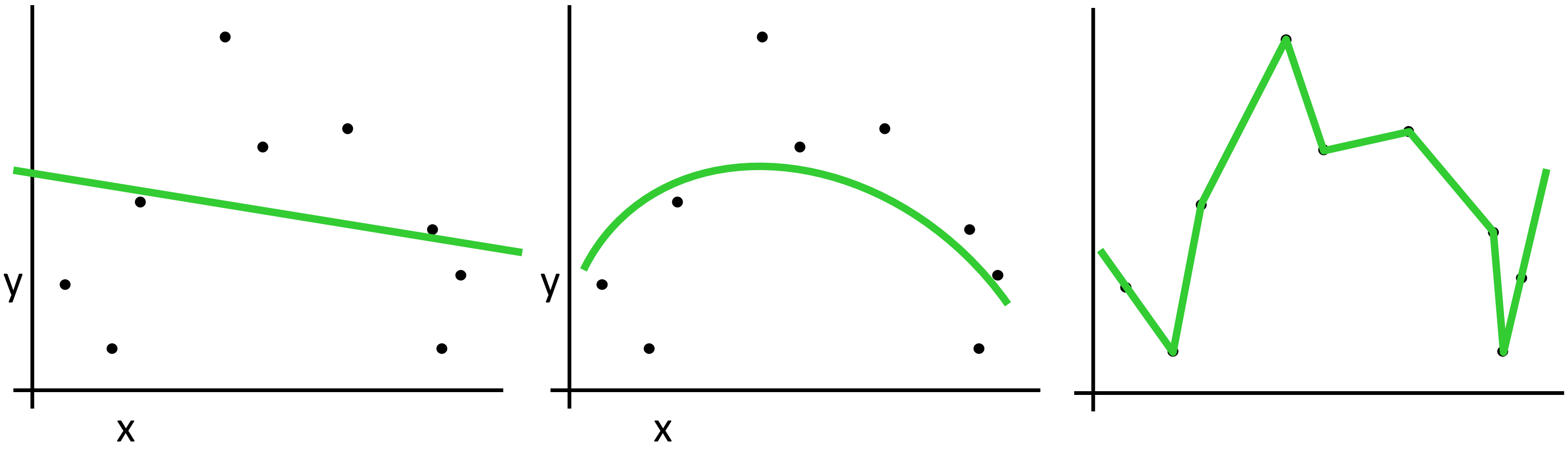
Also known as piecewise linear nonparametric regression if that makes you feel better

Which is best?



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

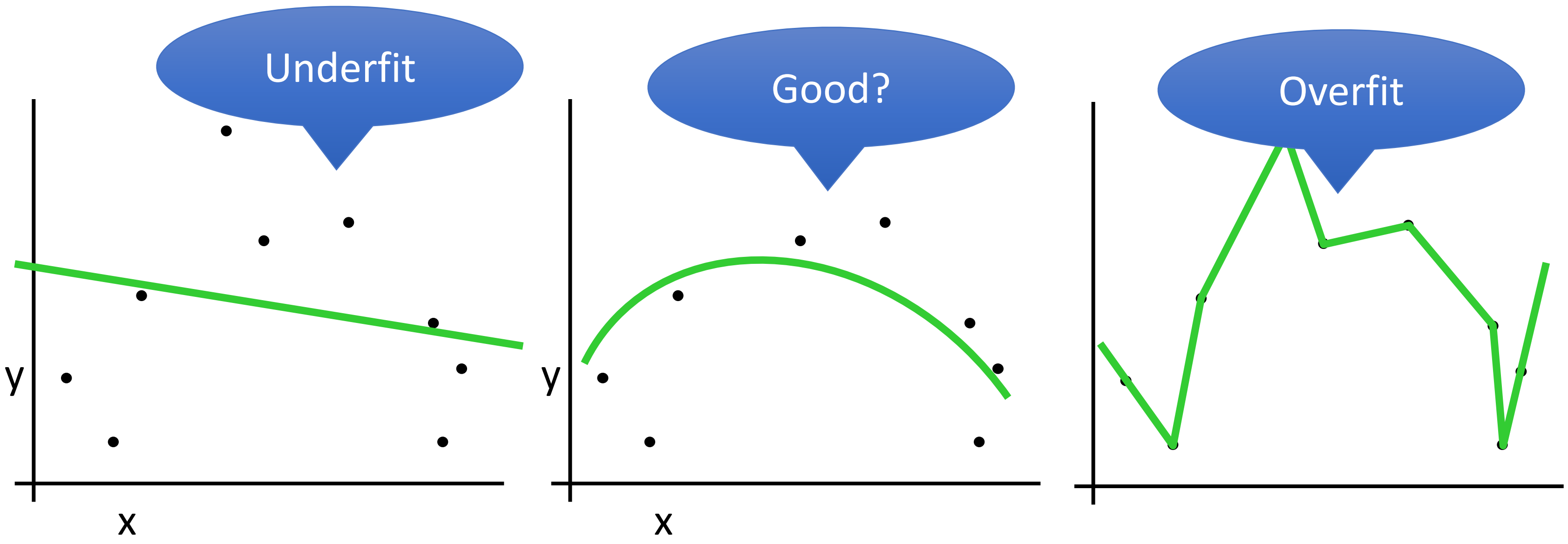
What do we really want?



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?"

What Model Type / Model Order to Select?



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

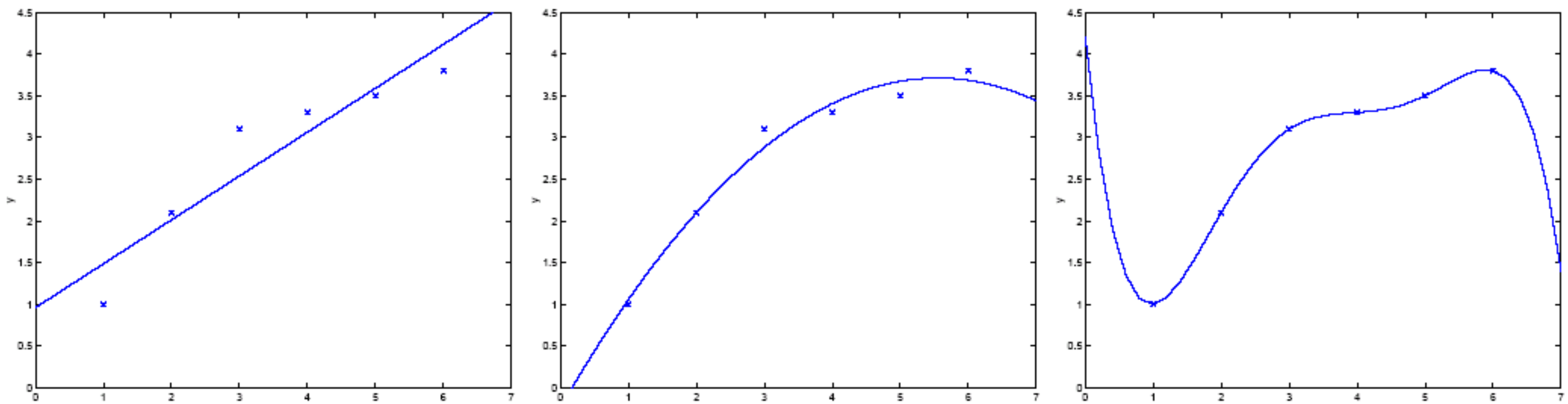
Generalisation: learn function / hypothesis from past data in order to “explain”, “predict”, “model” or “control” new data examples

What Model Order to Select?

Under fit

Looks good

Over fit



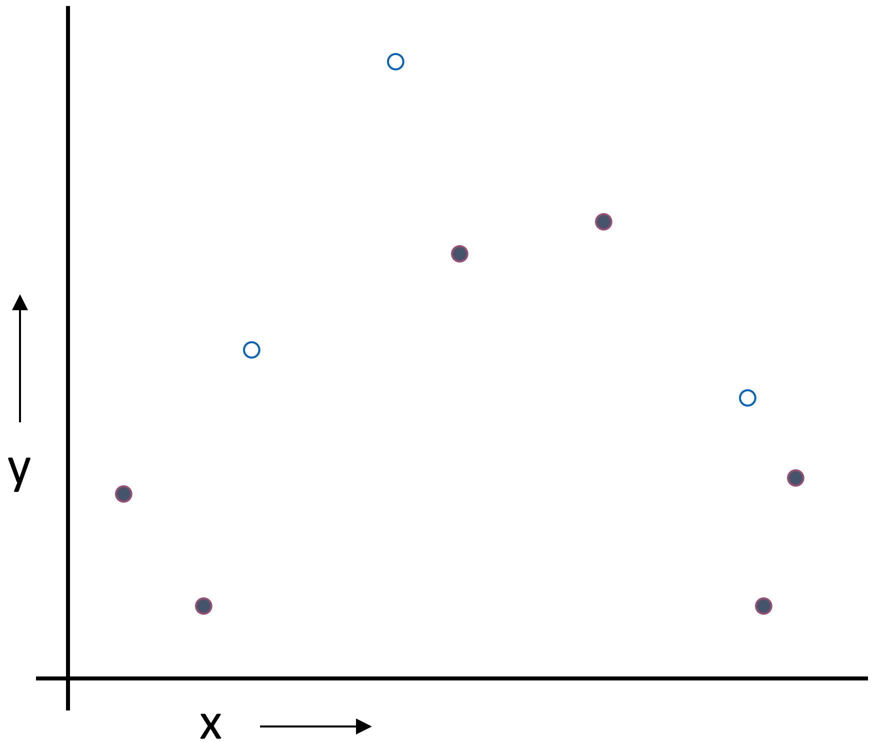
$$y = \theta_0 + \theta_1 x \qquad y = \theta_0 + \theta_1 x + \theta_2 x^2 \qquad y = \sum_{j=0}^{5} \theta_j x^j$$

Generalisation: learn function / hypothesis from past data in order to “explain”, “predict”, “model” or “control” new data examples

(a) Train-validation / (b) K-fold Cross Validation /

Choice-I: Train-Validation (Hold m out)

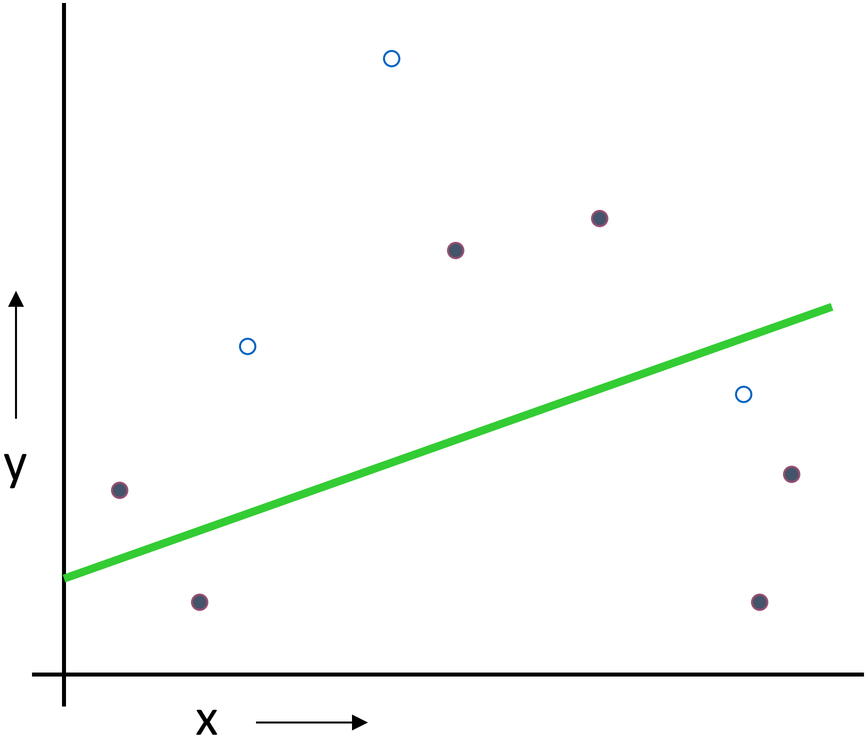
The validation set method



1. Randomly choose some percentage like 30% of the labeled data to be in a validation set
2. The remainder is a training set

Credit: Prof. Andrew Moore

The validation set method

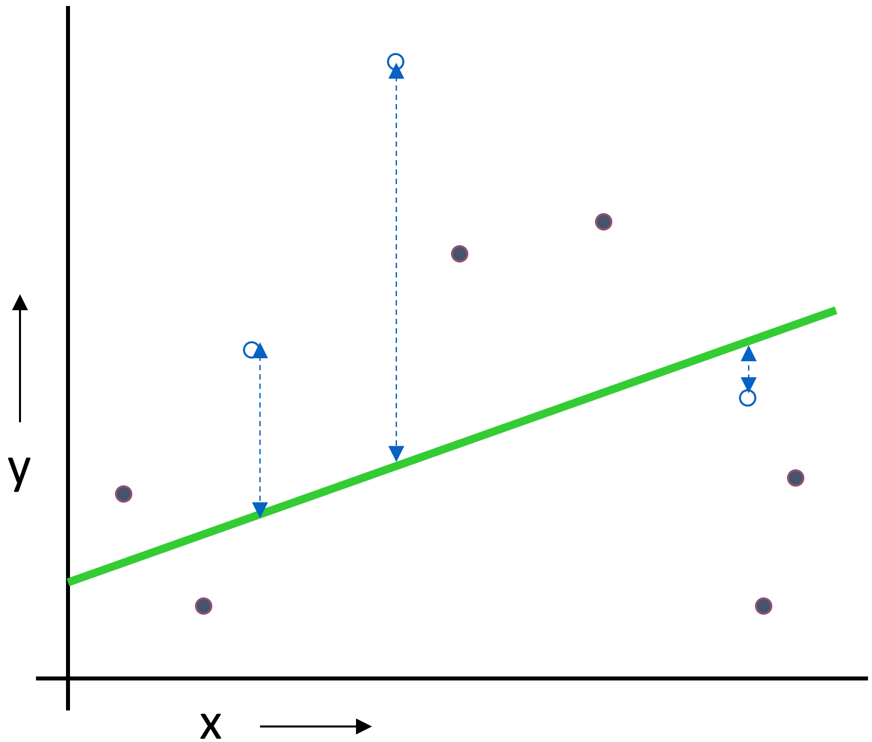


1. Randomly choose some percentage like 30% of the labeled data to be in a validation set
2. The remainder is a training set
3. Perform your regression on the training set

(Linear regression example)

Credit: Prof. Andrew Moore

The validation set method



1. Randomly choose 30% of the data to be in a validation set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the validation set

(Linear regression example)
Mean Squared Error = 2.4

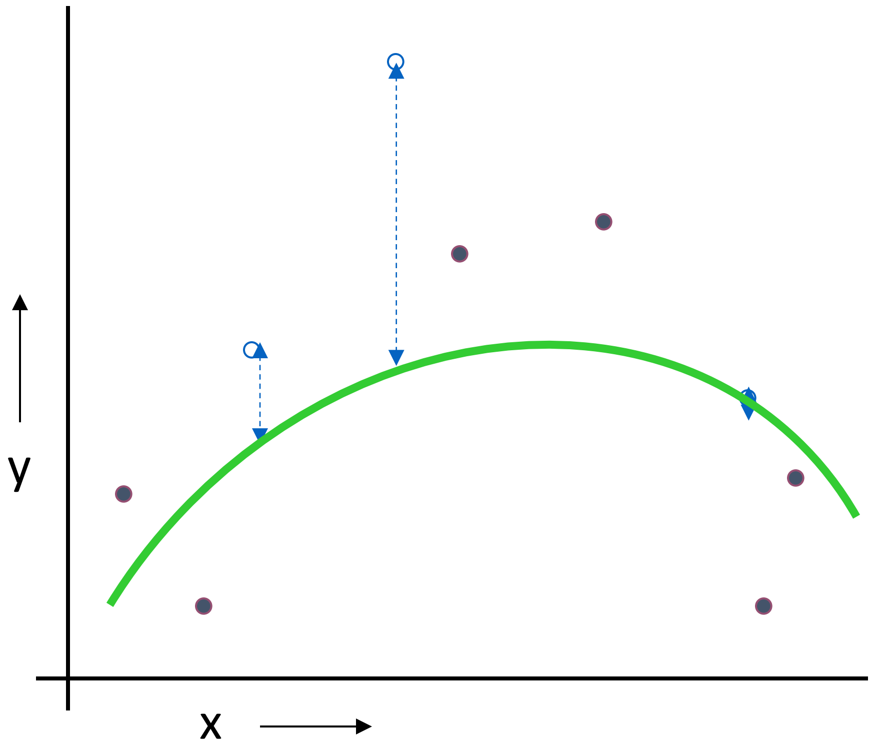
Credit: Prof. Andrew Moore

e.g. for Regression Models

- Mean Squared Error - MSE to report:

$$J_{test} = \frac{1}{m}\sum_{i=n+1}^{n+m}(\mathbf{x}_i^T\theta^* - y_i)^2 = \frac{1}{m}\sum_{i=n+1}^{n+m}\varepsilon_i^2$$

The validation set method

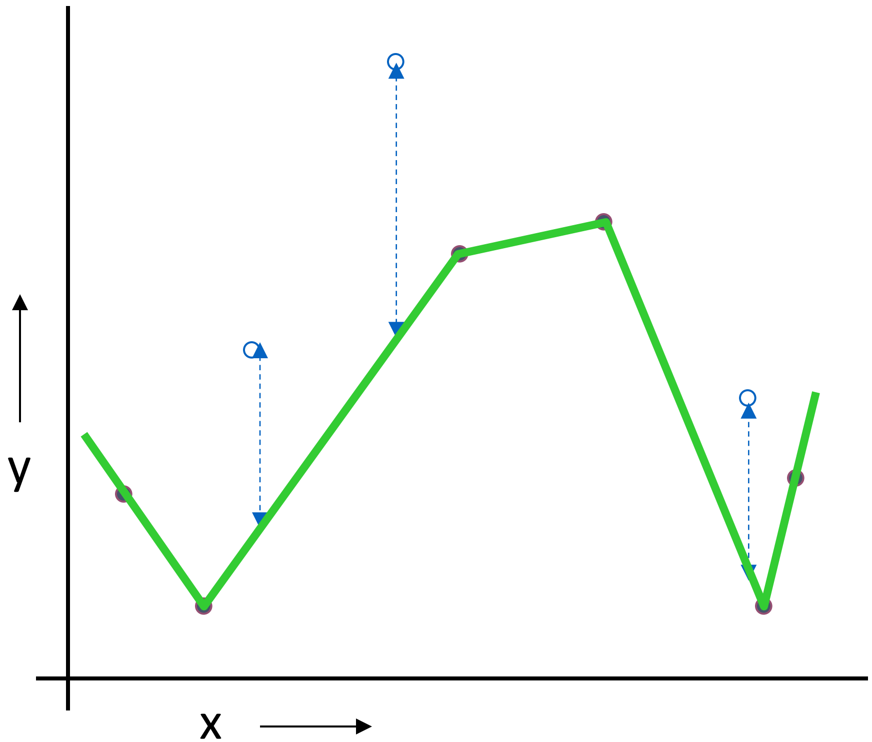


1. Randomly choose 30% of the data to be in a validation set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the validation set

(Quadratic regression example)
Mean Squared Error = 0.9

Credit: Prof. Andrew Moore

The validation set method



1. Randomly choose 30% of the data to be in a validation set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the validation set

(Join the dots example)
Mean Squared Error = 2.2

Credit: Prof. Andrew Moore

The validation set 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-set might just be lucky or unlucky

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

Credit: Prof. Andrew Moore

Choice-II: k-Fold Cross Validation

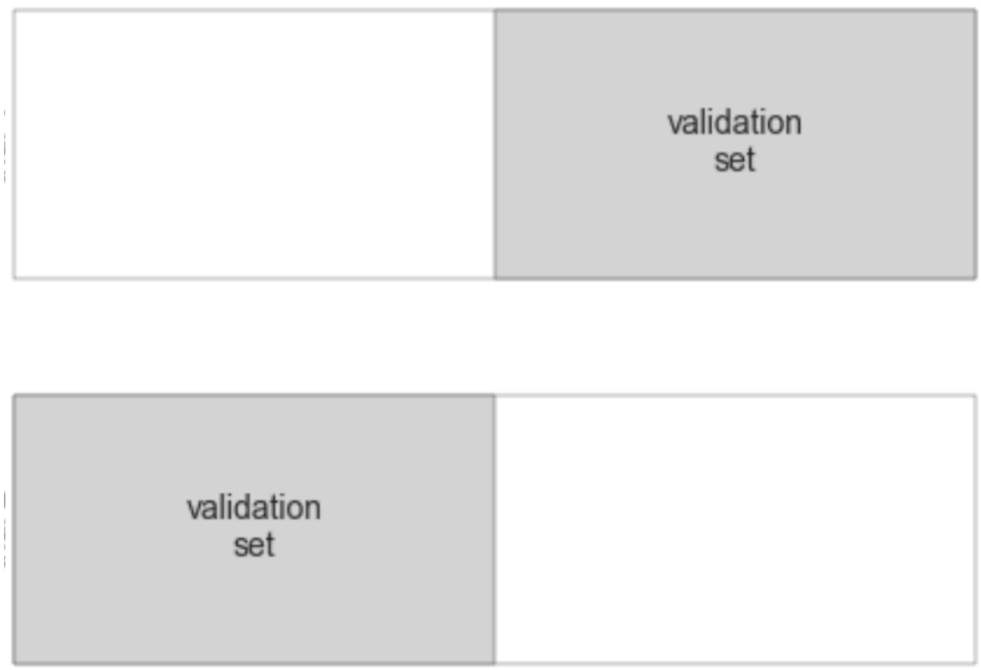
- Problem of train-validation: in many cases we don't have enough data to set aside a validation set
- Solution: Each data point is used both as train and validation
- Common types:
  - K-fold cross-validation (e.g. K=5, K=10)
  - Leave-one-out cross-validation (LOOCV, i.e., k=n)

e.g. k=10 folds Cross Validation

- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as validation set
- Collect the scores from each validation
- We normally use the mean of the scores

model	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
1	train	train	train	train	train	train	train	train	train	test
2	train	train	train	train	train	train	train	train	test	train
3	train	train	train	train	train	train	train	test	train	train
4	train	train	train	train	train	train	test	train	train	train
5	train	train	train	train	train	test	train	train	train	train
6	train	train	train	train	test	train	train	train	train	train
7	train	train	train	test	train	train	train	train	train	train
8	train	train	test	train	train	train	train	train	train	train
9	train	test	train	train	train	train	train	train	train	train
10	test	train	train	train	train	train	train	train	train	train

e.g. k=2 folds Cross Validation



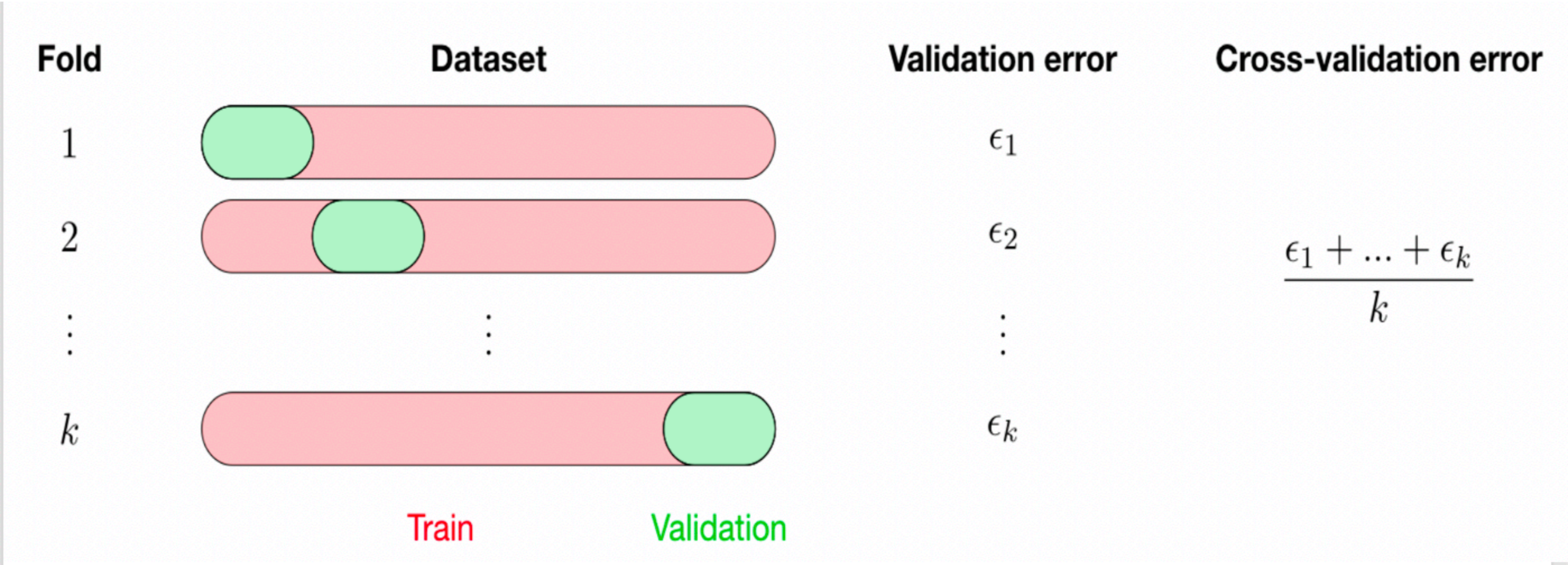


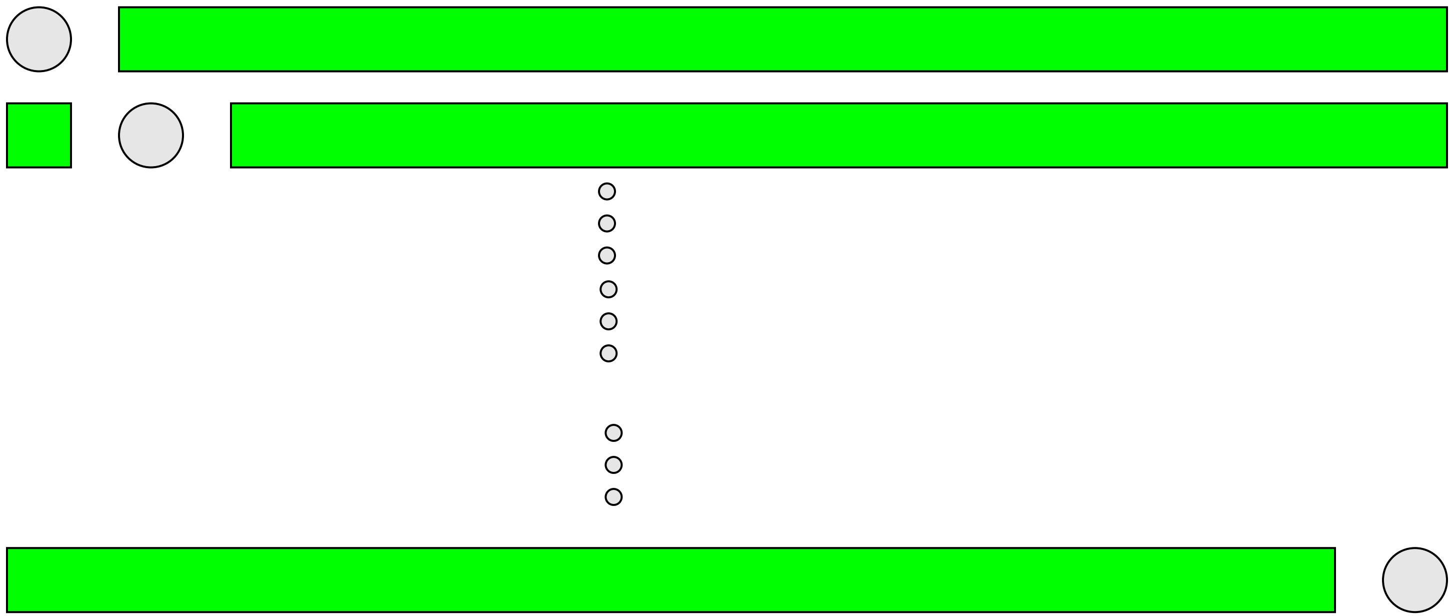
Image Credit: Stanford Machine Learning course

Which K?

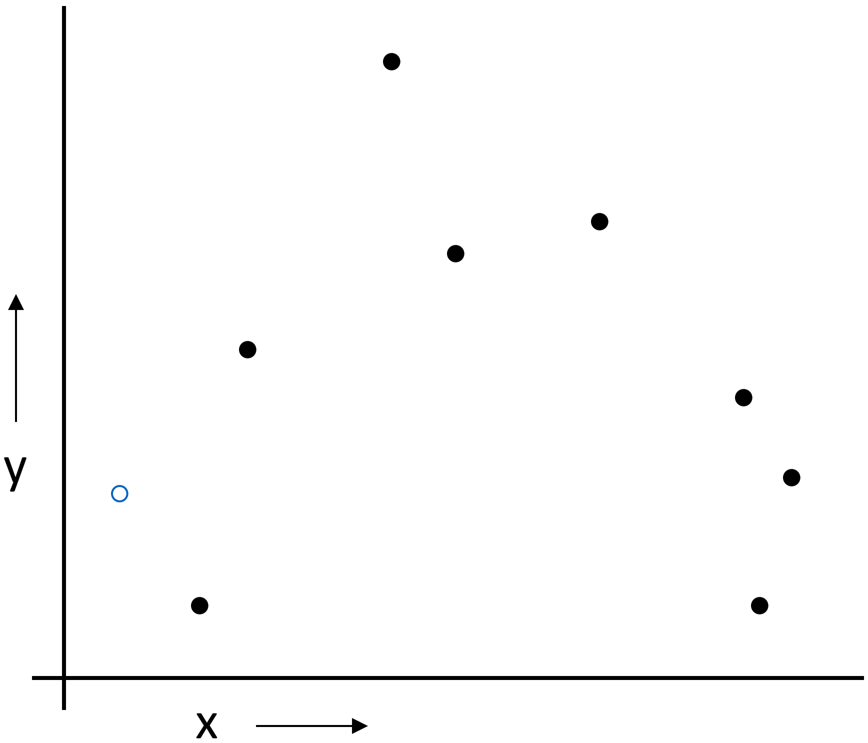
Leave-one-out / LOOCV:

(k=n-fold cross validation)

n is num. of data samples



LOOCV (Leave-one-out Cross Validation)

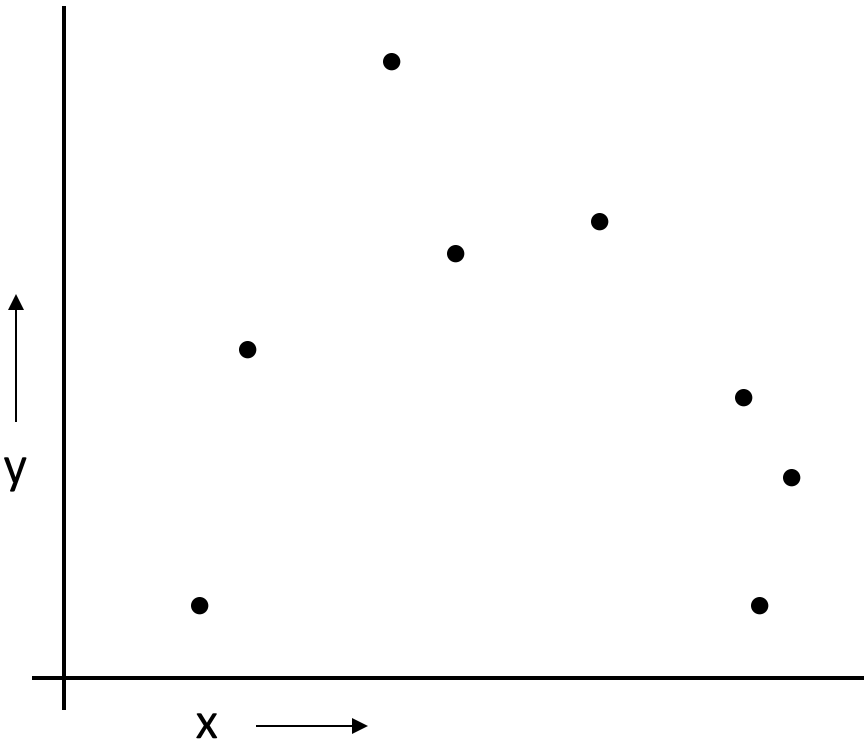


For k=1 to n

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

Credit: Prof. Andrew Moore

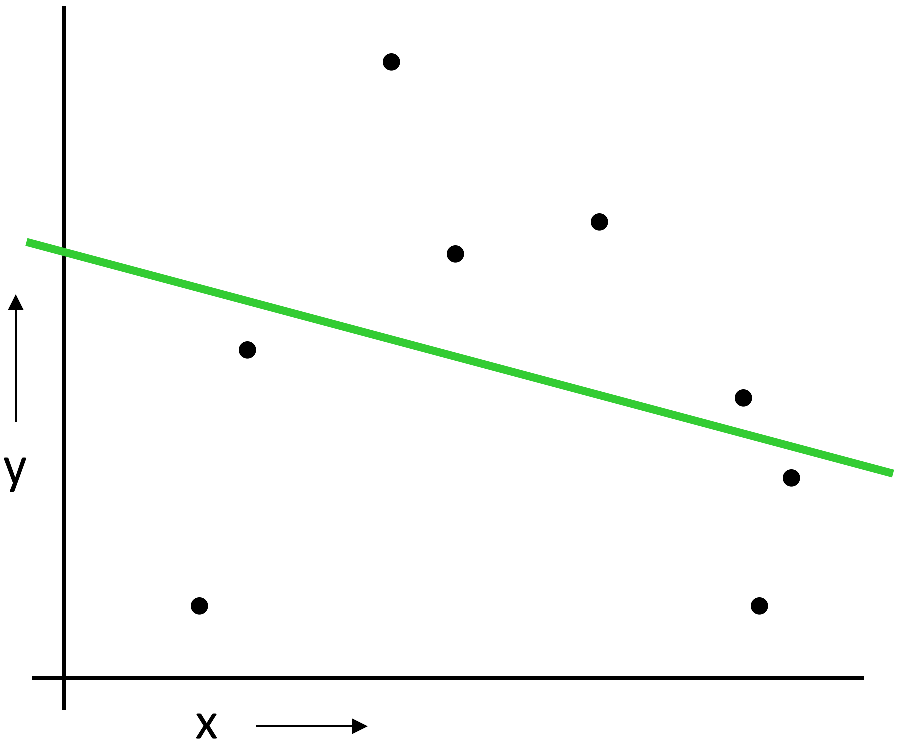
LOOCV (Leave-one-out Cross Validation)



For k=1 to n

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)

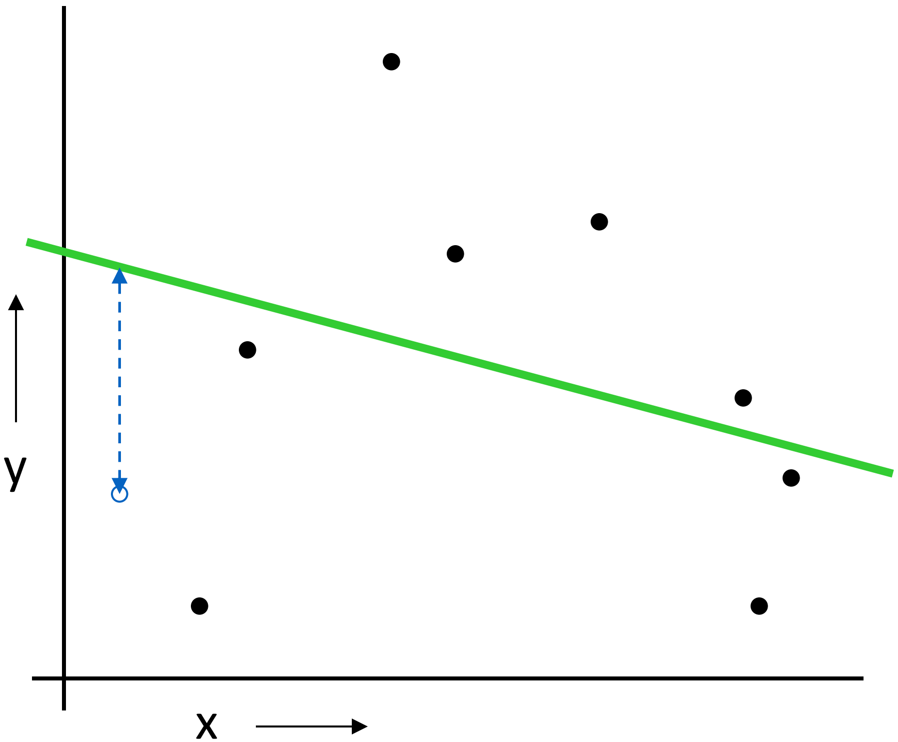


For k=1 to n

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 n-1 datapoints

Credit: Prof. Andrew Moore

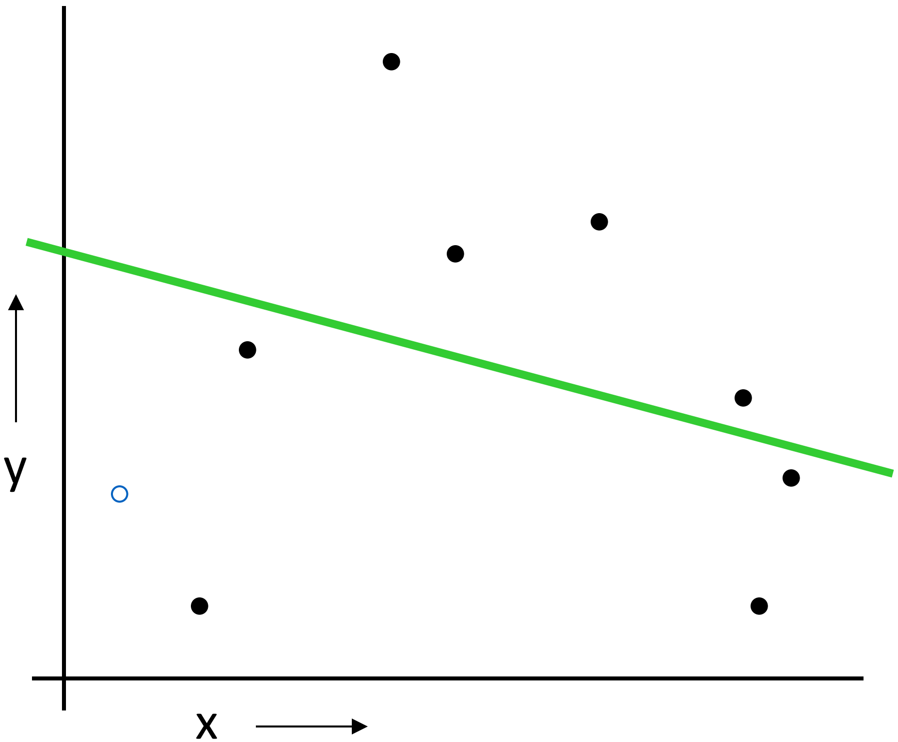
LOOCV (Leave-one-out Cross Validation)



For k=1 to n

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)

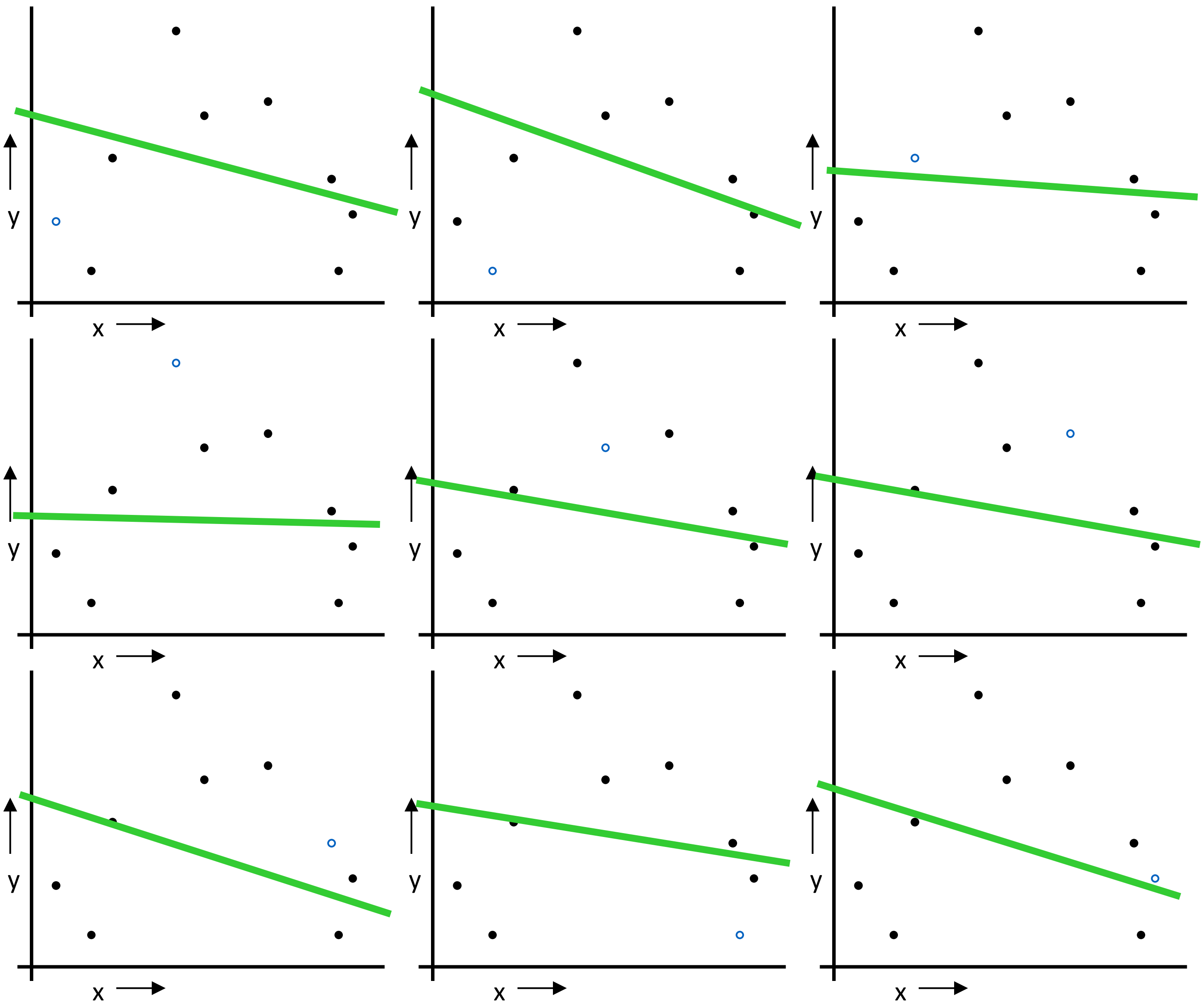


For k=1 to R

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 for Linear Regression



For k=1 to n

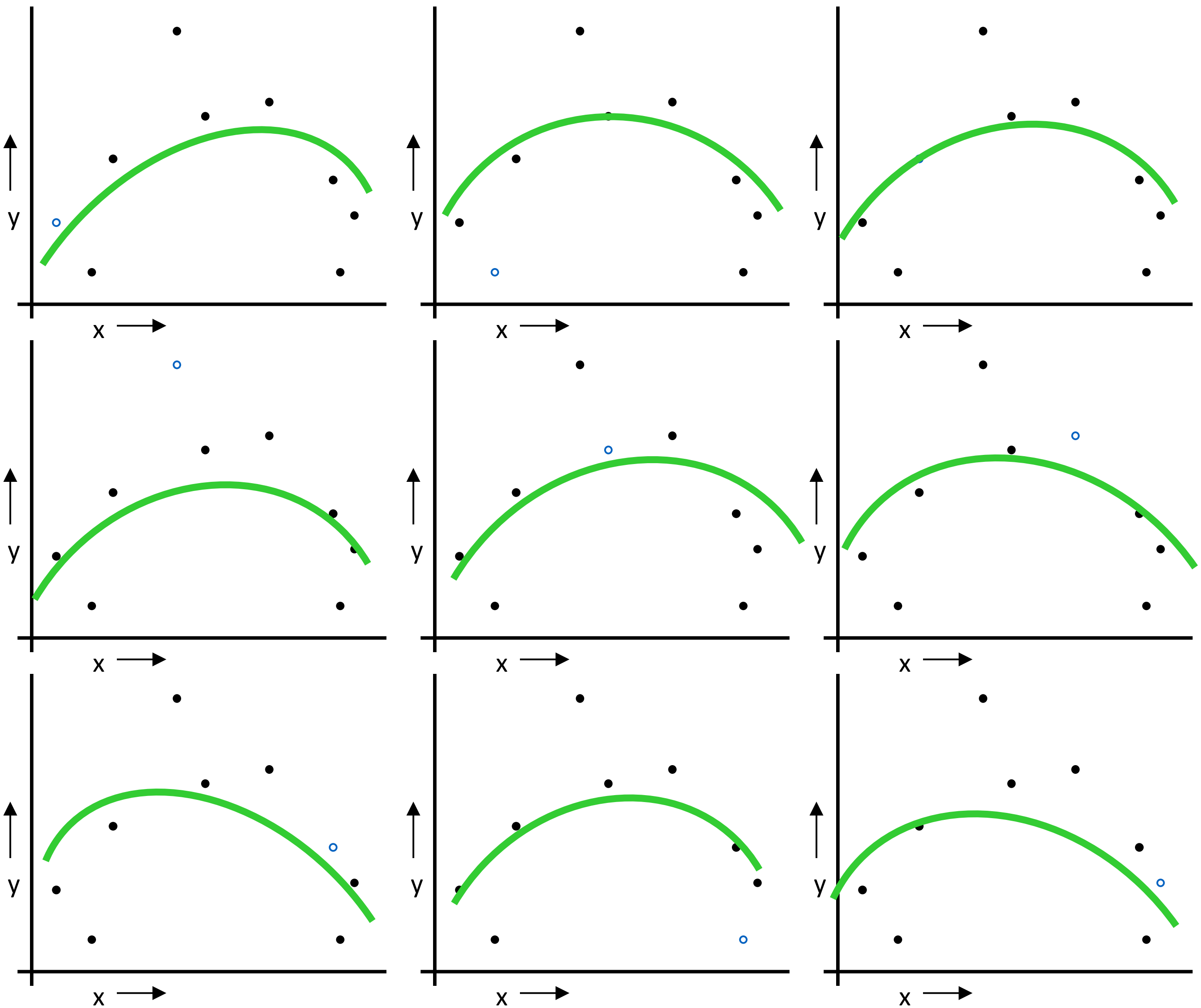
1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k,y_k)

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

$MSE_{LOOCV} = 2.12$

Credit: Prof. Andrew Moore

LOOCV for Quadratic Regression



For k=1 to n

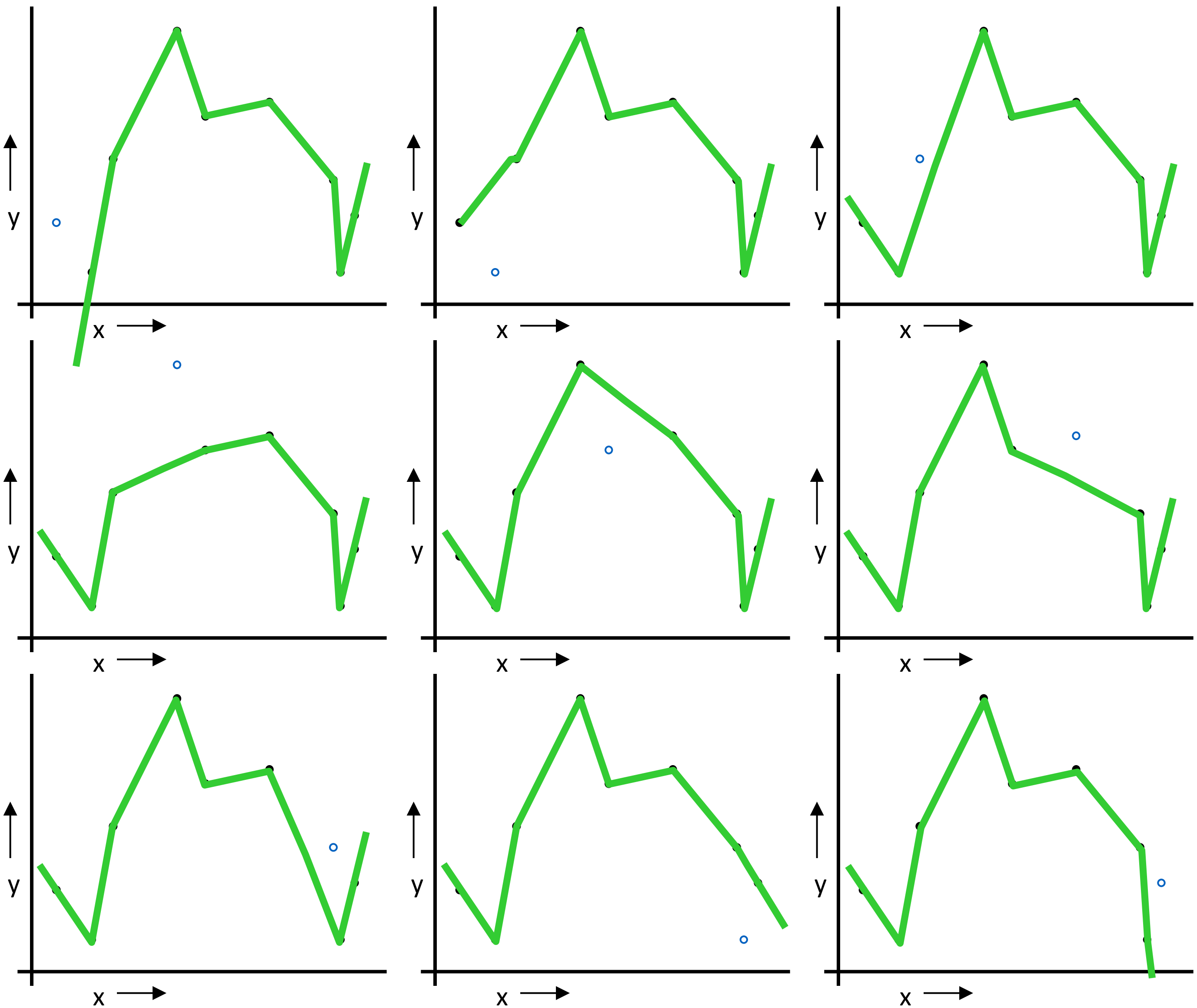
1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k,y_k)

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

$MSE_{LOOCV}=0.962$

Credit: Prof. Andrew Moore

LOOCV for Join The Dots



For k=1 to n

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 n-1 datapoints
4. Note your error (x_k,y_k)

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

$MSE_{LOOCV}=3.33$

Credit: Prof. Andrew Moore

Which kind of Cross Validation?

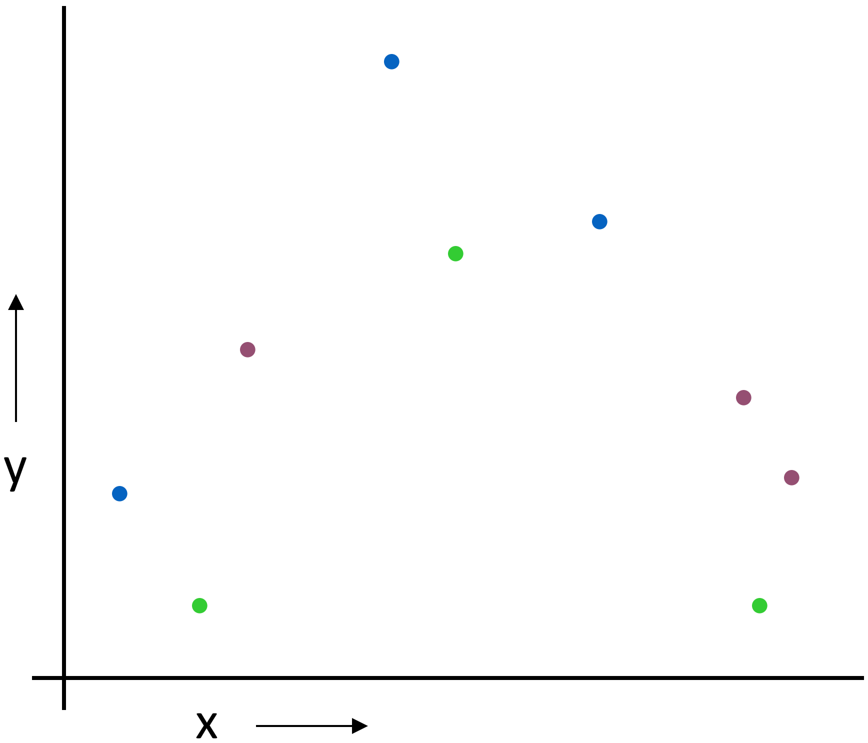
	Downside	Upside
validation-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data

..can we get the best of both worlds?

Credit: Prof. Andrew Moore

k-fold Cross Validation

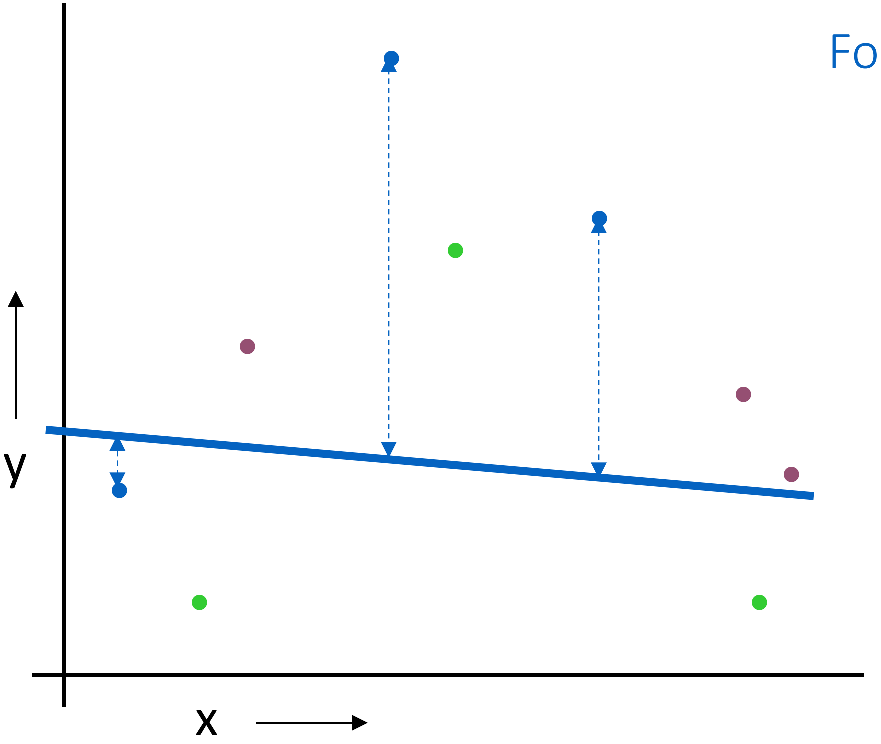
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)



Credit: Prof. Andrew Moore

k-fold Cross Validation

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

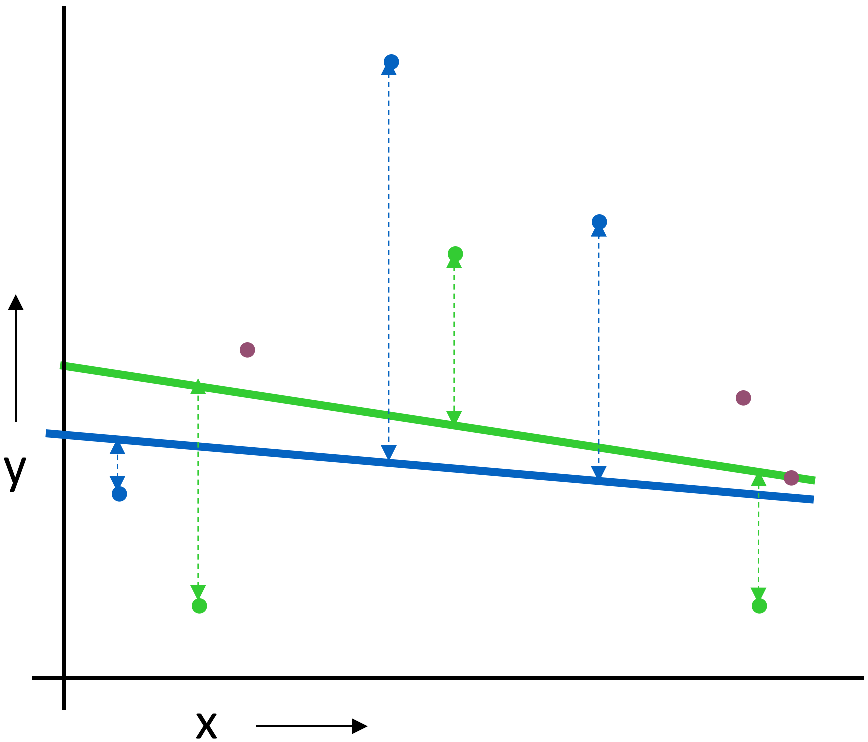


For the blue partition: Train on all the points not in the blue partition. Find the validation-set sum of errors on the blue points.

Credit: Prof. Andrew Moore

k-fold Cross Validation

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



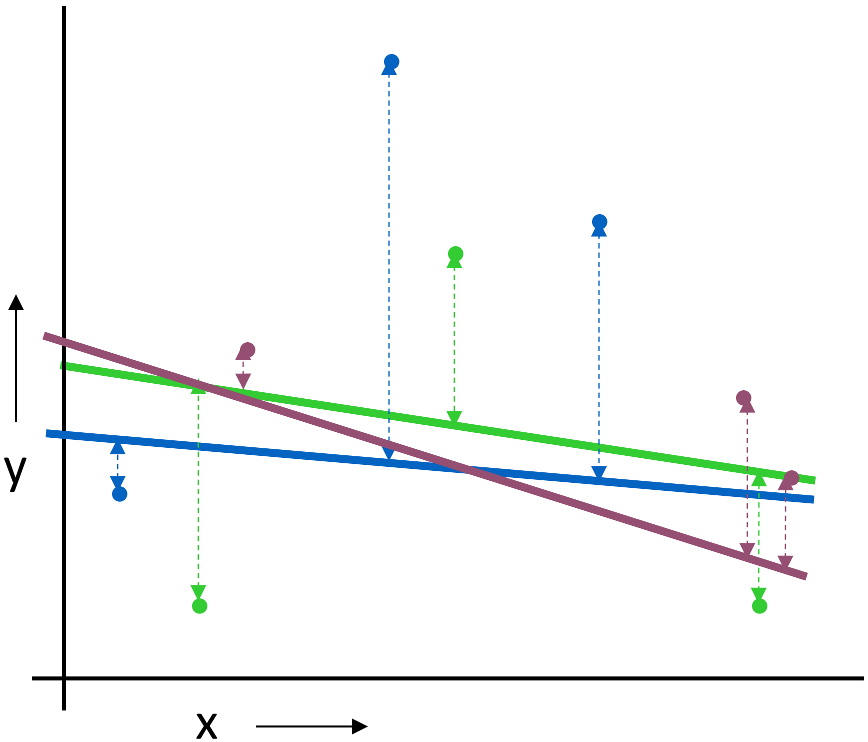
For the blue partition: Train on all the points not in the red partition. Find the validation-set sum of errors on the red points.

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.

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple 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.

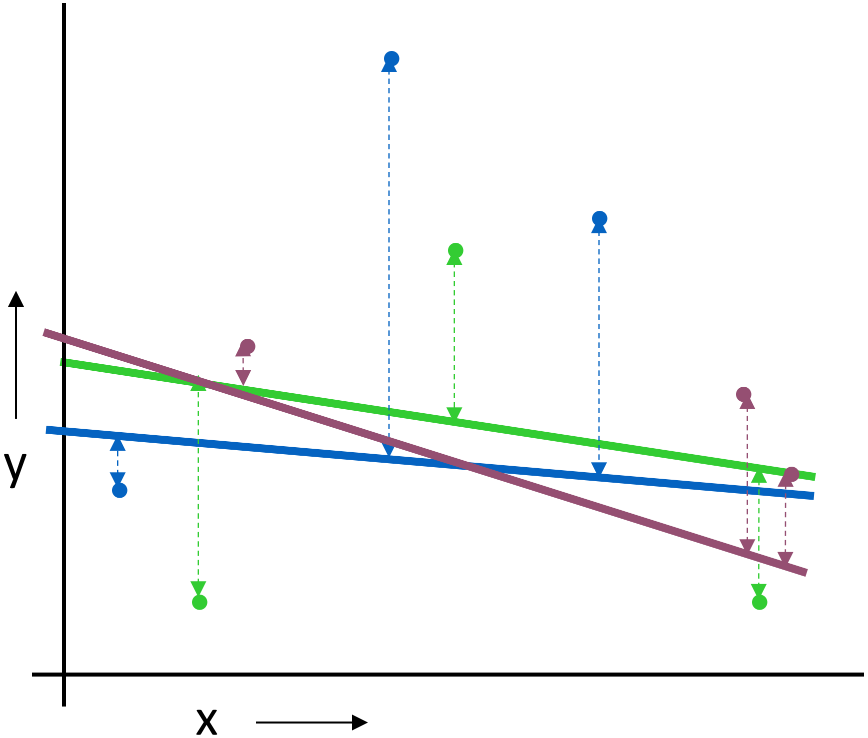
For the green partition: Train on all the points not in the green partition. Find the validation-set sum of errors on the green points.

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

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple 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 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.

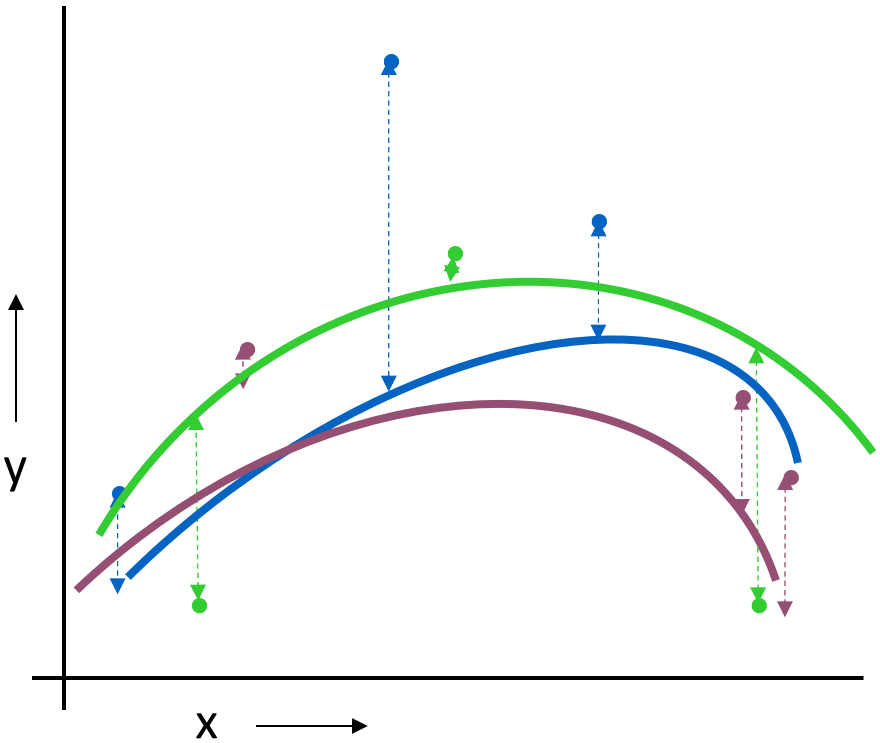
For the purple partition: Train on all the points not in the purple partition. Find the validation-set sum of errors on the purple points.

Then report the mean error

Credit: Prof. Andrew Moore

k-fold Cross Validation

Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple 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 validation-set sum of errors on the red points.

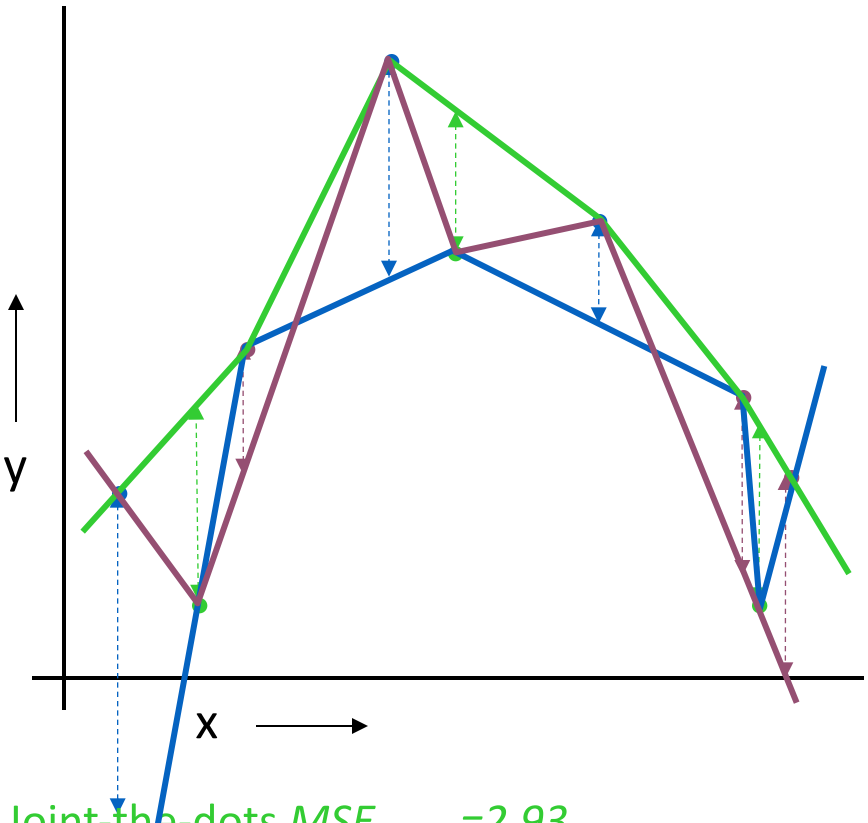
For the green partition: Train on all the points not in the green partition. Find the validation-set sum of errors on the green points.

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

Then report the mean error

Credit: Prof. Andrew Moore

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 Purple 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

Credit: Prof. Andrew Moore

Which kind of Cross Validation?

	Downside	Upside
validation-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 validation set	Only wastes 10%. Only 10 times more expensive instead of n times.
3-fold	Wastier than 10-fold. More Expensive than validation set style	better than validation-set
n-fold	Identical to Leave-one-out	

CV-based Model Selection

- We're trying to decide which algorithm/model/ hyperpara to use.
- We train/learn/fit each model and make a table…

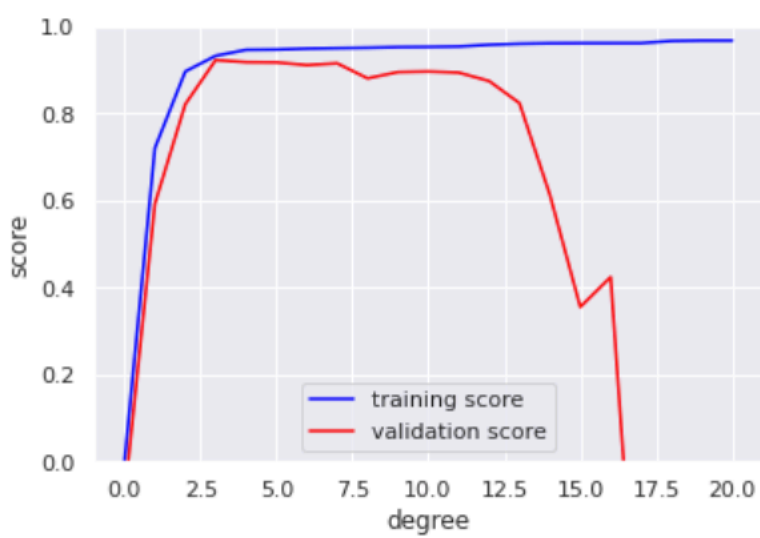
i	f_i	TRAINERR	k-FOLD-CV-ERR	Choice
1	f_1			
2	f_2			
3	f_3			YEAH!!!!
4	f_4			
5	f_5			
6	f_6			

Credit: Prof. Andrew Moore

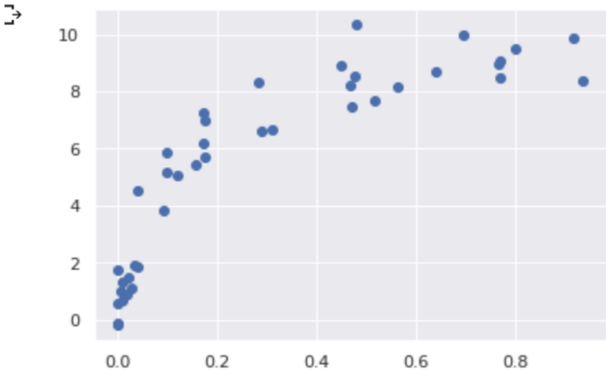
I will code-run: https://colab.research.google.com/drive/1MFy_6da9zL4yqGXTZg80My_2KACo0pY8#scrollTo=T-a0H80OQgHD

Adapted from:
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.03-Hyperparameters-and-Model-Validation.ipynb
https://scikit-learn.org/stable/modules/learning_curve.html

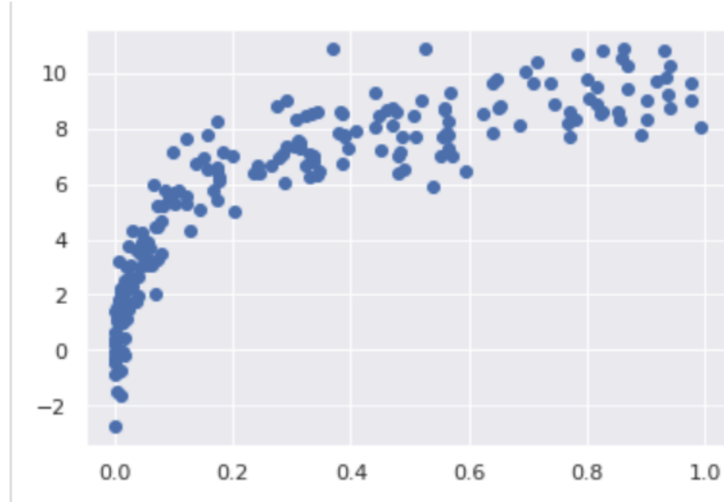
```
[21] from sklearn.model_selection import validation_curve

     degree = np.arange(0, 21)
     train_score, val_score = validation_curve(PolynomialRegression(), X, y,
                                               'polynomialfeatures__degree', degree, cv=7)

     plt.plot(degree, np.median(train_score, 1), color='blue', label='training score')
     plt.plot(degree, np.median(val_score, 1), color='red', label='validation score')
     plt.legend(loc='best')
     plt.ylim(0, 1)
     plt.xlabel('degree')
     plt.ylabel('score');
```



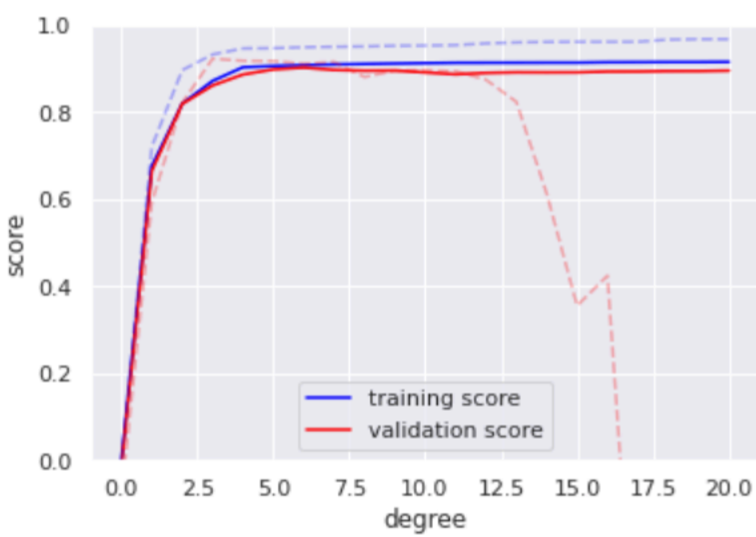
```
X, y = make_data(40)
plt.scatter(X, y);
```



```
X2, y2 = make_data(200)
plt.scatter(X2.ravel(), y2);
```



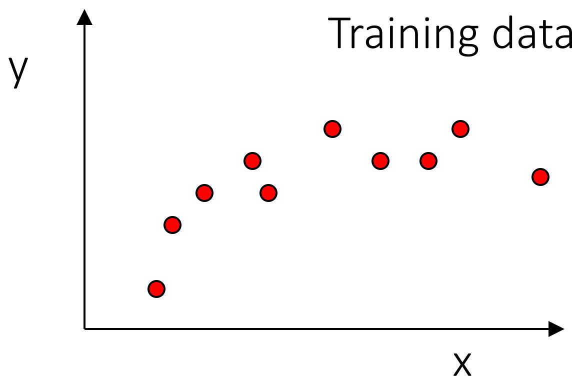
```
plt.plot(degree, np.median(train_score2, 1), color='blue'
plt.plot(degree, np.median(val_score2, 1), color='red', l
plt.plot(degree, np.median(train_score, 1), color='blue',
plt.plot(degree, np.median(val_score, 1), color='red', al
plt.legend(loc='lower center')
plt.ylim(0, 1)
plt.xlabel('degree')
plt.ylabel('score');
```

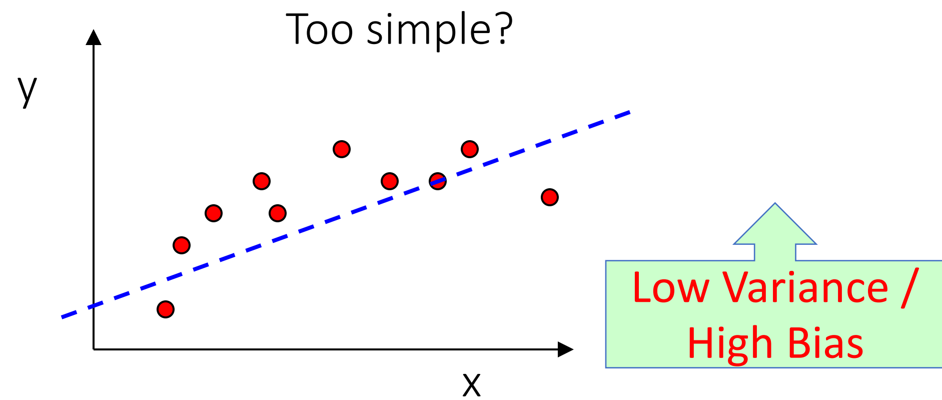


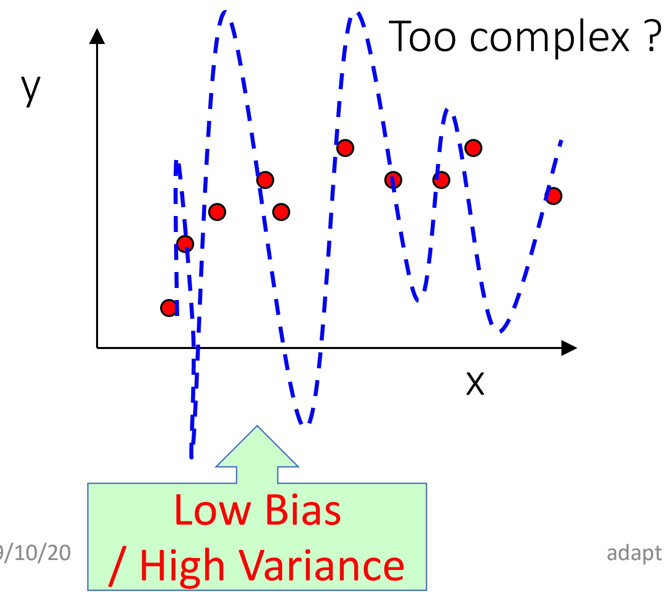
Behavior of the validation curve:

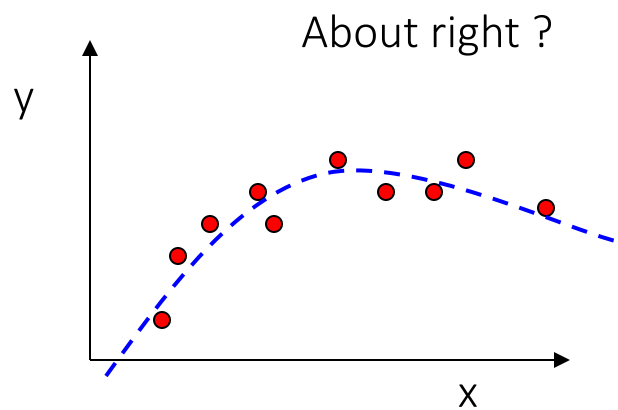
- the model complexity
- the number of training points

Later: Complexity versus Goodness of Fit



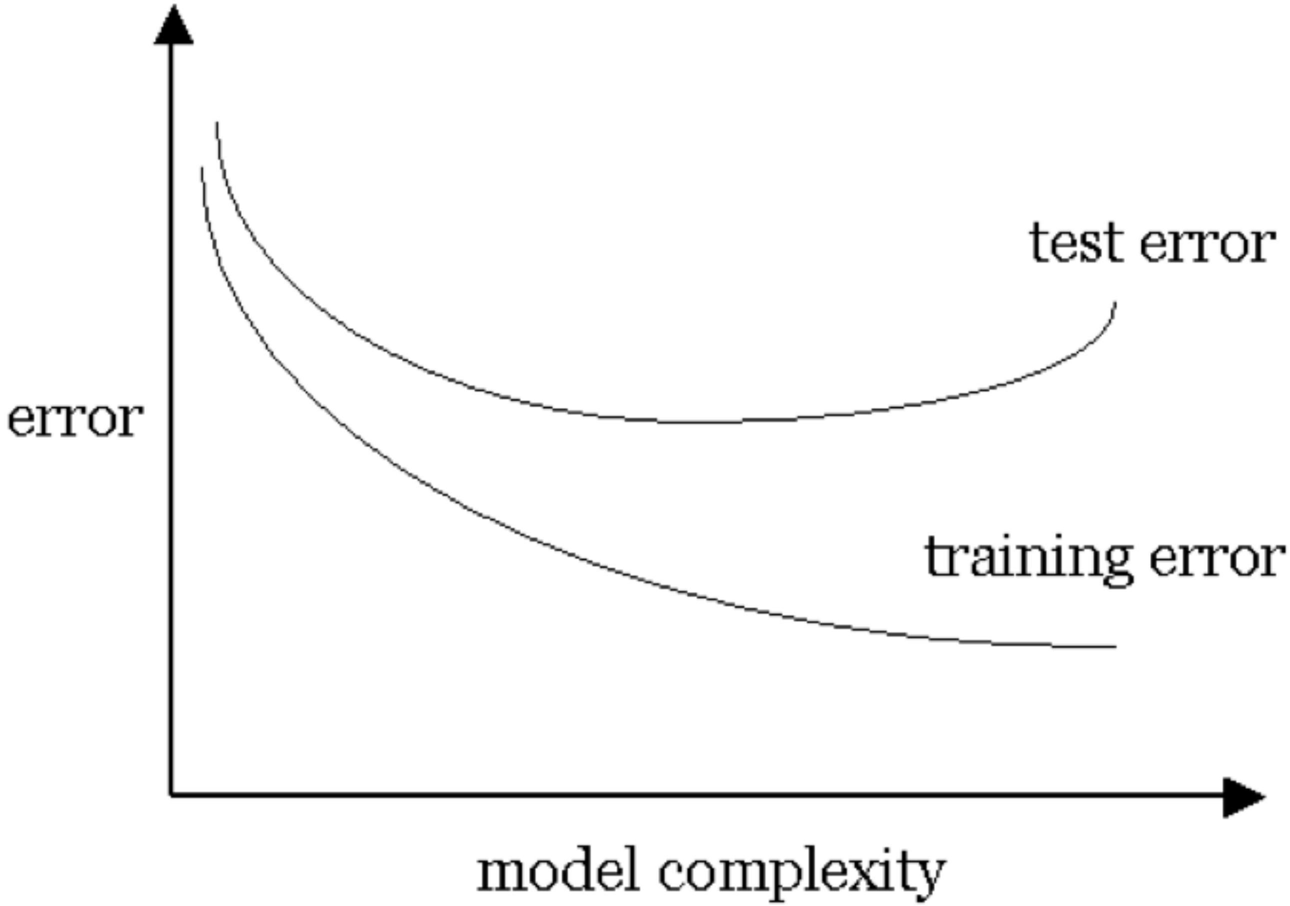






What ultimately matters: GENERALIZATION

Model Selection vs. Model Assessment



k-CV on train to choose model and hyperparameter / then a separate test set to assess future performance

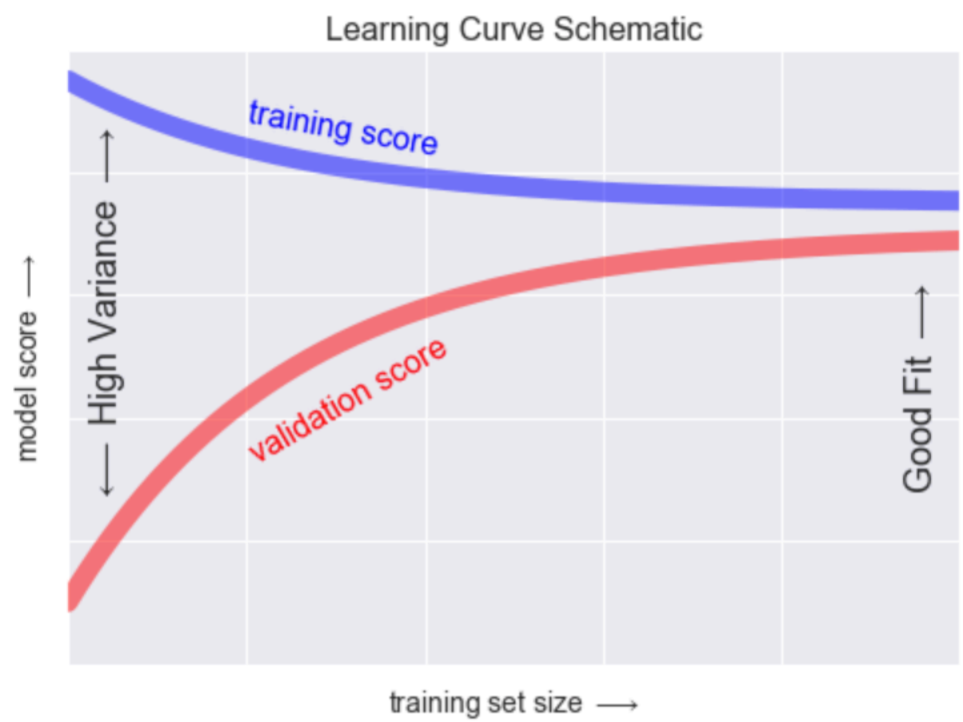
A plot of the training/validation score with respect to the size of the training set is known as a *learning curve.*

The general behavior we would expect from a learning curve is this:

•A model of a given complexity will *overfit* a small dataset: this means the training score will be relatively high, while the validation score will be relatively low.

•A model of a given complexity will *underfit* a large dataset: this means that the training score will decrease, but the validation score will increase.

•A model will never, except by chance, give a better score to the validation set than the training set: this means the curves should keep getting closer together but never cross.



https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.03-Hyperparameters-and-Model-Validation.ipynb

References

- Big thanks to Prof. Eric Xing @ CMU for allowing me to reuse some of his slides

- [ ] Prof. Nando de Freitas's tutorial slide
- [ ] Prof. Andrew Moore's slides @ CMU