UVA CS 6316: Machine Learning

Lecture 5: Non-Linear Regression Models and Model Selection

Dr. Yanjun Qi

University of Virginia

Department of Computer Science

Course Content Plan ➔ Six major sections of this course

- ❑ Regression (supervised) — Y is a continuous
- ❑ Classification (supervised) — Y is a discrete
- ❑ Unsupervised models — NO Y
- ❑ Learning theory — About f()
- ❑ Graphical models — About interactions among X1,… Xp
- ❑ Reinforcement Learning — Learn program to Interact with its environment

Last: Multivariate Linear Regression in a Nutshell

Task: y

Representation: x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters :

Regression: y continuous

Y = Weighted linear sum of X's

Sum of Squared Error (Least Squared)

Normal Equation / GD / SGD

Regression coefficients

$$\hat{y} = f(x) = \theta^T x$$

Today: Multivariate Linear Regression with basis Expansion

Task: y

Representation: x, f()

Score Function: L()

Search/Optimization : argmin()

Models, Parameters :

Regression: y continuous

Y = Weighted linear sum of (X basis expansion)

Sum of Squared Error (Least Squared)

Normal Equation / GD / SGD

Regression coefficients

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)^T \theta$$

LR with non-linear basis functions

- LR does not mean we can only deal with linear relationships

$$\hat{y} = \theta^T \mathbf{x} \quad \Rightarrow \quad \hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(\mathbf{x}) = \theta^T \varphi(\mathbf{x})$$

$$\vec{\theta} = [\theta_0, \theta_1, \cdots, \theta_m]^T$$

$$\vec{\phi} = [1, \phi_1(x), \cdots, \phi_m(x)]^T$$

LR with non-linear basis functions

- We are free to design basis functions (e.g., non-linear features:

Here $\varphi_j(x)$ are predefined basis functions (also $\varphi_0(x)=1$)

- E.g.: polynomial regression with degree up-to two (d=2) :

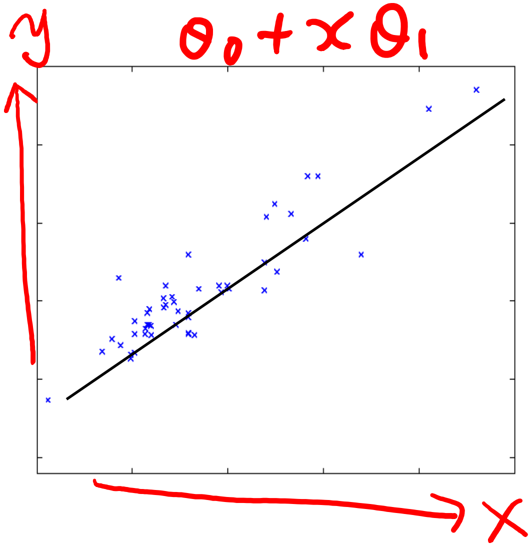
$$\varphi(x) := \left[1, x, x^2\right]^T$$

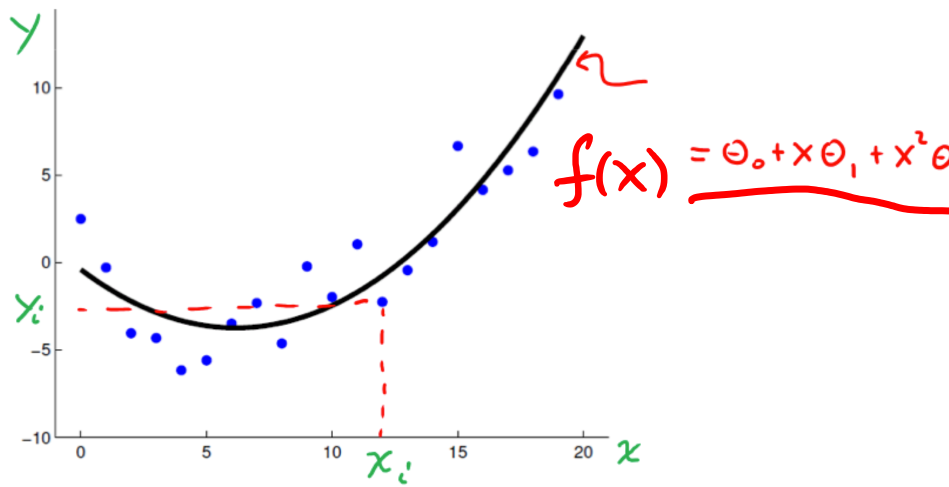
linear $\vec{x} : [1, x]^T$

e.g. (1) polynomial regression

$$\hat{y} = \theta^T \mathbf{x}$$

$$\hat{y} = \theta^T \varphi(\mathbf{x})$$





$$\theta^* = \left(X^T X\right)^{-1} X^T \vec{y}$$

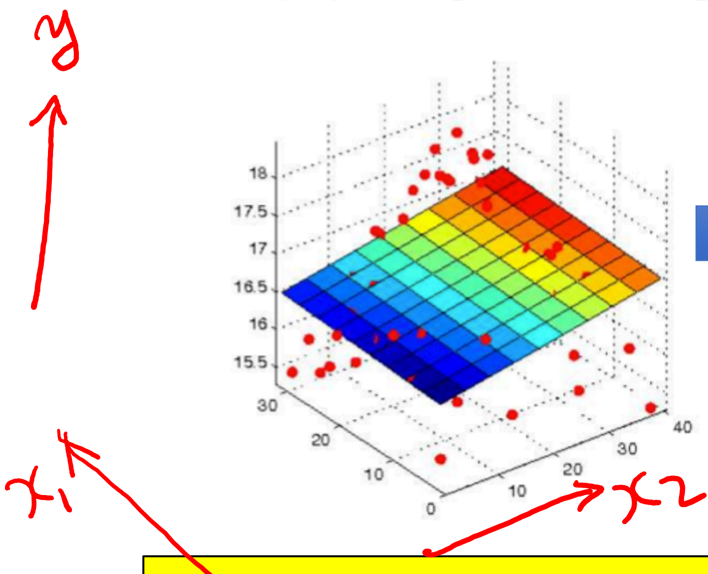
$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

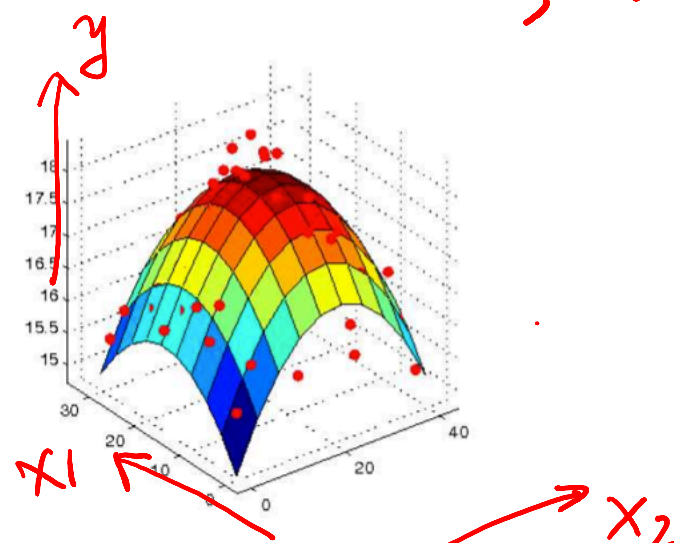
$$\varphi(x) := \left[1, x, x^2\right]^T$$

e.g. (1) polynomial regression

$$\hat{y} = \theta^T \mathbf{x} \quad \Rightarrow \quad \hat{y} = \theta^T \varphi(\mathbf{x})$$

$(\mathbf{x}) = [1, x_1, x_2]$ $\quad\quad$ $\phi(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_2^2]$ $, x_1 x_2$

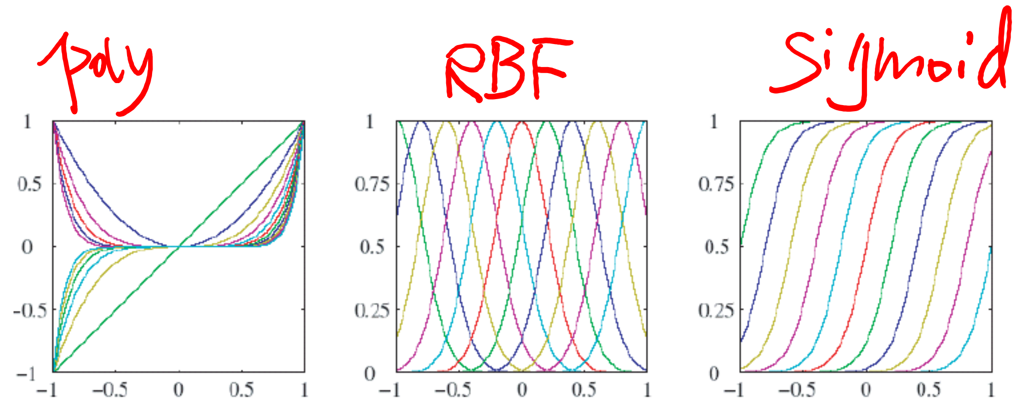




KEY: when the bases are given, the problem of learning the parameters from data is still linear.

Many Possible Basis functions

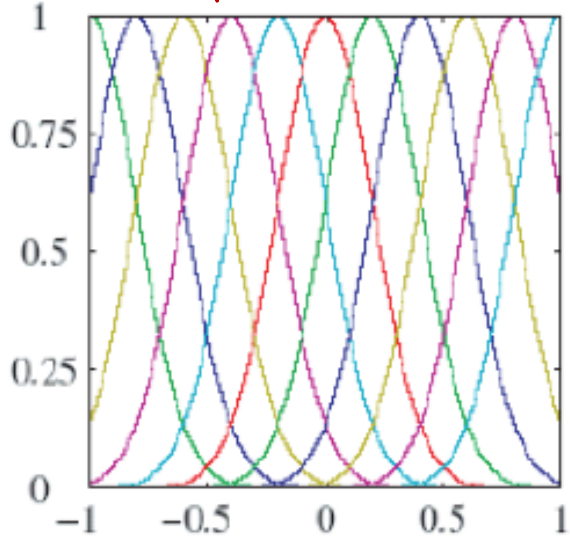
- There are many basis functions, e.g.:
  - Polynomial $$\varphi_j(x) = x^{j-1}$$ $$\left[1, x, x^2, x^3, \ldots, x^d\right]$$
  - Radial basis functions $$\varphi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\lambda^2}\right)$$
  - Sigmoidal $$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$
  - Splines,
  - Fourier,
  - Wavelets, etc



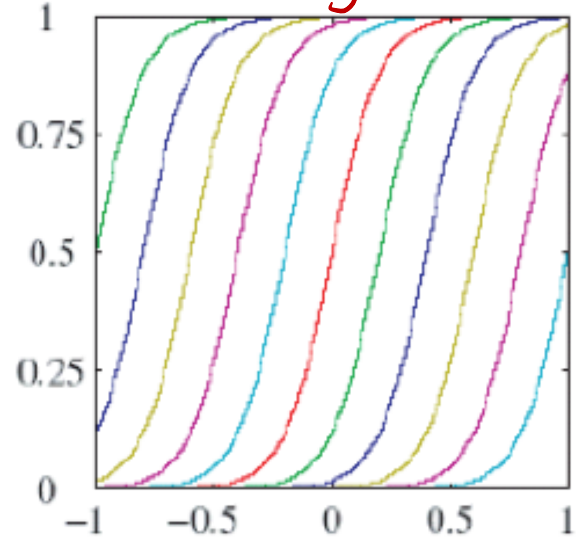
$$\varphi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{2\lambda^2}\right)$$

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right)$$

RBF

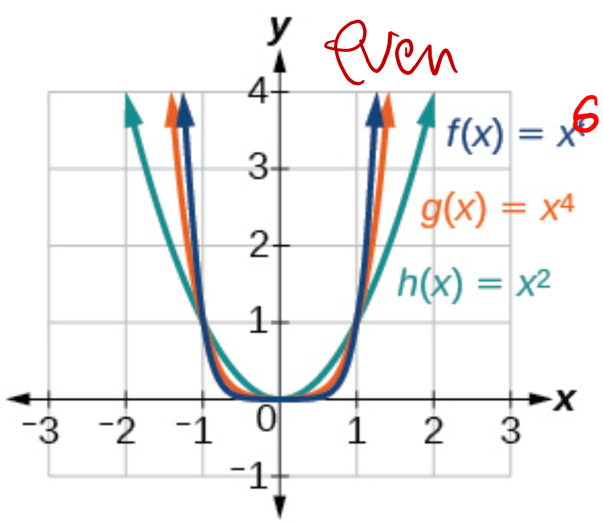


Sigmoid

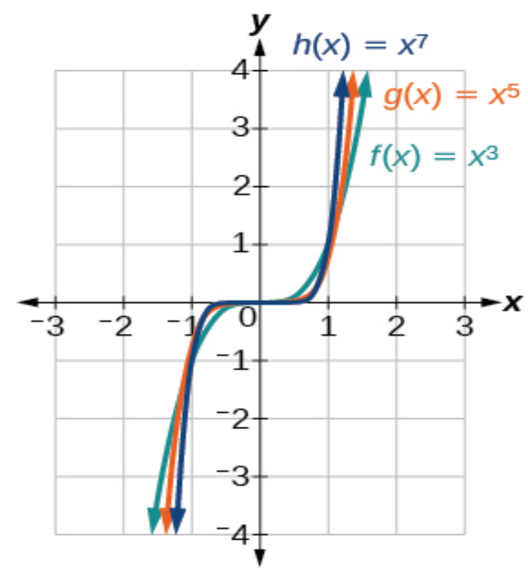


even

$$\varphi_j(x) = x^{j-1}$$



odd



e.g. (2) LR with radial-basis functions

- E.g.: LR with RBF regression:

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)^T \theta$$

$$\varphi_j(x) := K_{\lambda_j}(x, r_j) = \exp\left(-\frac{(x-\mu_j)^2}{2\lambda_j^{\,2}}\right)$$

E.g. with four predefined RBF kernels

$$\varphi(x): = \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3), K_{\lambda_4}(x, r_4)\right]^T$$

e.g. (2) LR with radial-basis functions

- E.g.: LR with RBF regression:

$$\hat{y} = \theta_0 + \sum_{j=1}^{m} \theta_j \varphi_j(x) = \varphi(x)^T \theta$$

$$\varphi(x) := \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3), K_{\lambda 4}(x, r_4) \right]^T$$

$$\vec{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4]^T$$

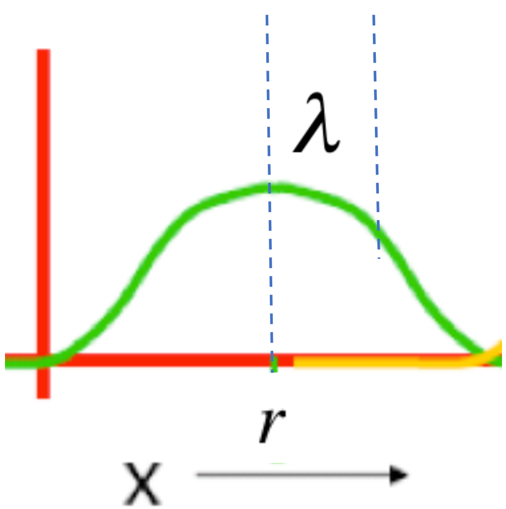
$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

RBF = radial-basis function: a function which depends only on the radial distance from a centre point

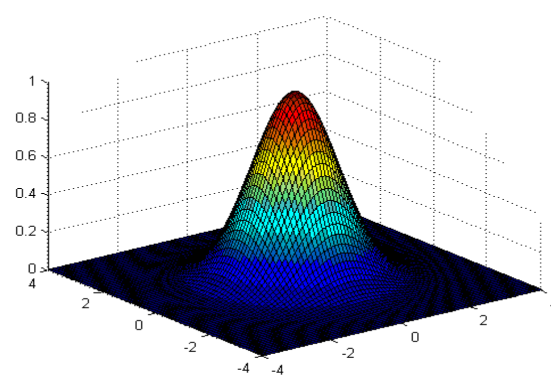
Gaussian RBF →

$$K_{\lambda}(x,r)=\exp\left(-\frac{(x-r)^2}{2\lambda^2}\right)$$

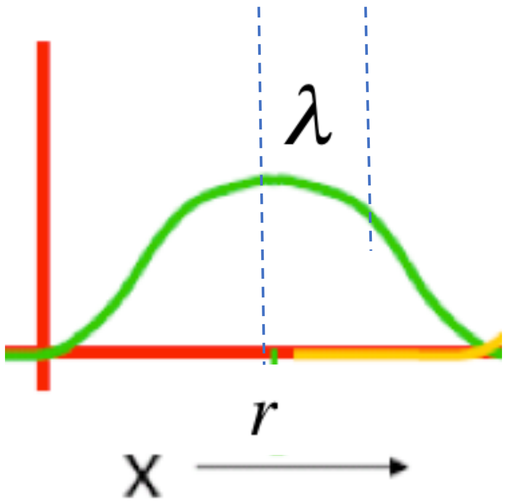
as distance from the center r increases, the output of the RBF decreases



1D case



2D case



$$K_\lambda(x,r) = \exp\left(-\frac{(x-r)^2}{2\lambda^2}\right)$$

X =	$K_\lambda(x,r) =$
r	1
$r+\lambda$	0.6065307
$r+2\lambda$	0.1353353
$r+3\lambda$	0.0001234098

e.g. another regression with 3 1D RBF basis functions (given 3 predefined centres and width)

$$\varphi(x) := \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3)\right]^T$$

$(\lambda \quad , \quad c_1) \quad (\lambda \quad , \quad c_2) \quad (\lambda \quad , \quad c_3)$

$f(x) = \theta^0 + \theta_1 K_{\lambda_1}(x, r_1) + \ldots + \theta_3 K_{\lambda_3}(x, r_3)$

$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

Given Training Data's scatter plot:

$(x_i, y_i) \quad i=1,\ldots,n$

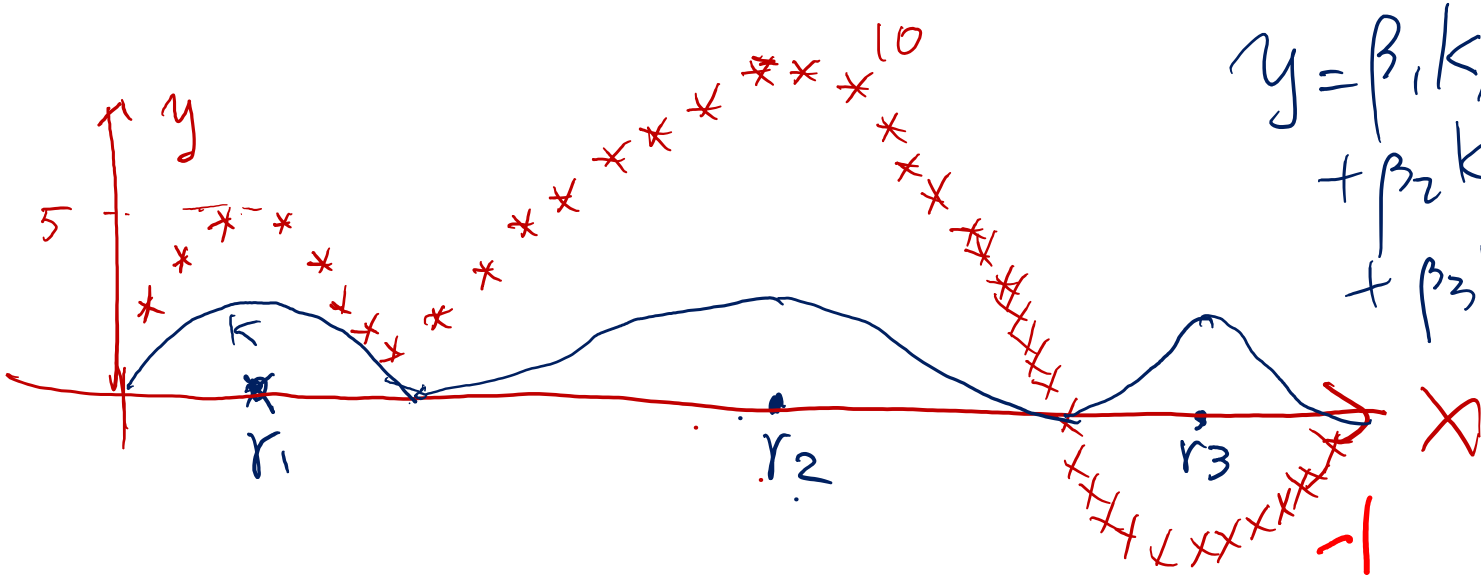
After fit:

$$f(x) = -0.5\,K_{\lambda_1}(x, r_1) + 5\,K_{\lambda_2}(x, r_2) + 2\,K_{\lambda_3}(x, r_3)$$

e.g. another regression with 3 1D RBF basis functions (assuming 3 predefined centres and width)

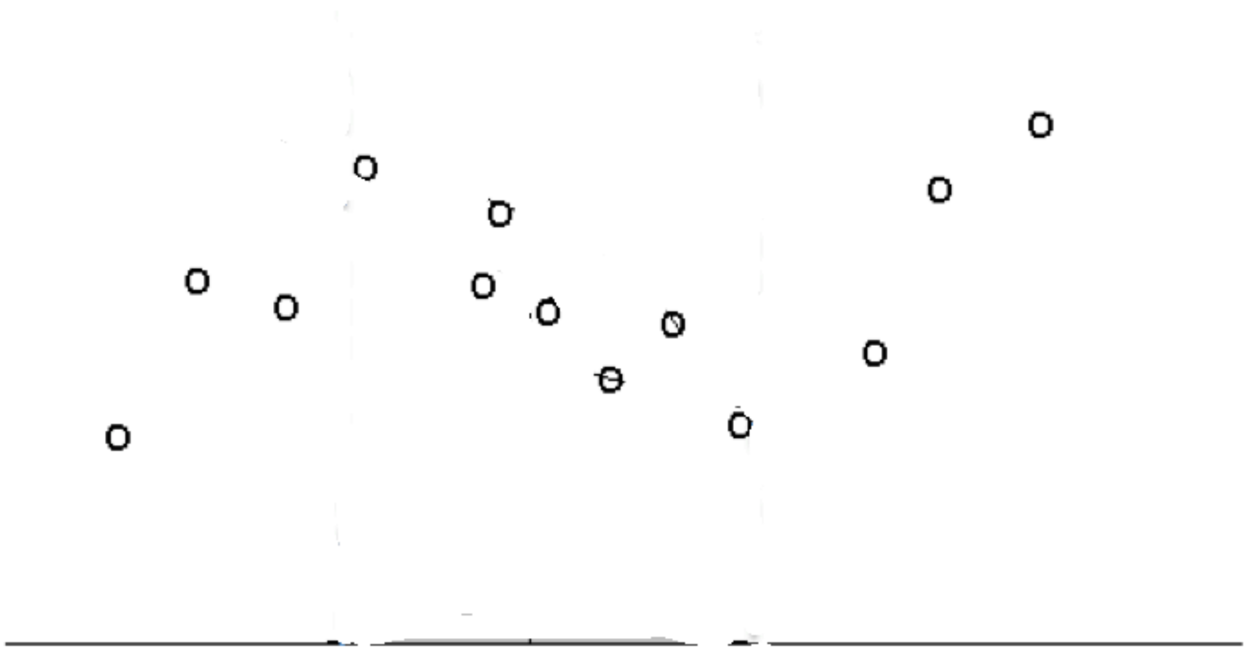
$$\varphi(x) := \left[1, K_{\lambda_1}(x, r_1), K_{\lambda 2}(x, r_2), K_{\lambda_3}(x, r_3)\right]^T$$

$$\theta^* = \left(\varphi^T \varphi\right)^{-1} \varphi^T \vec{y}$$

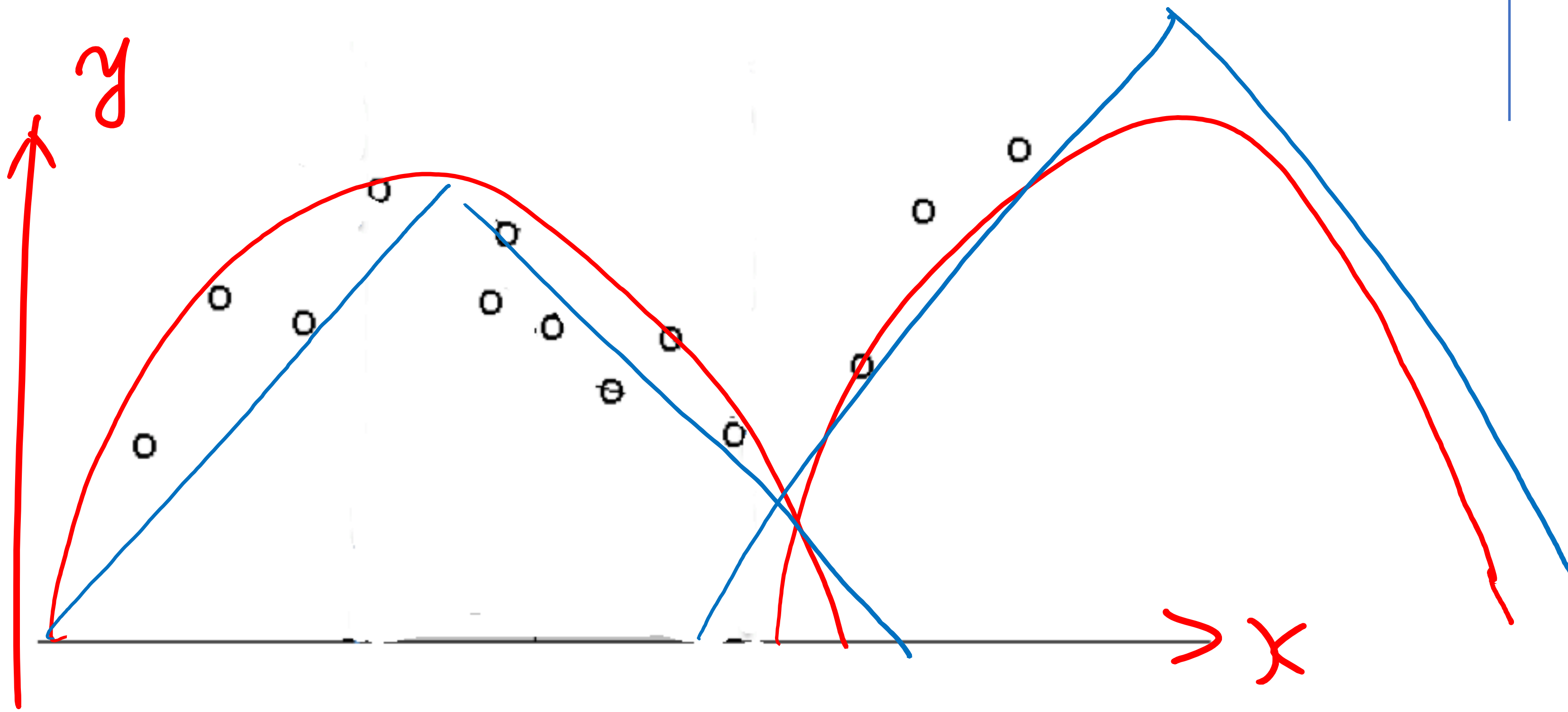


$$y = \beta_1 K_{\lambda_1}(x, r_1) + \beta_2 K_{\lambda_2}(x, r_2) + \beta_3 K_{\lambda_3}(x, r_3)$$

e.g. Another dataset: even more possible Basis Function: RBF, or Piecewise Linear based?



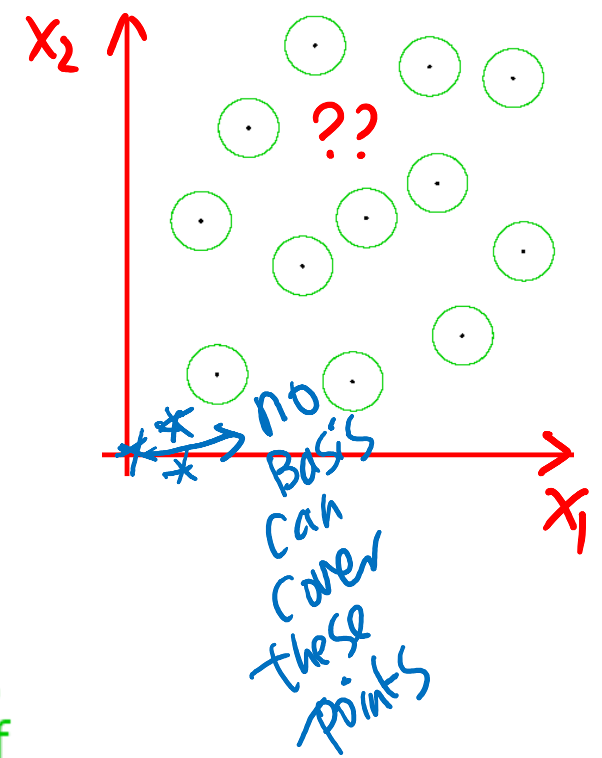
e.g. Even more possible Basis Func?



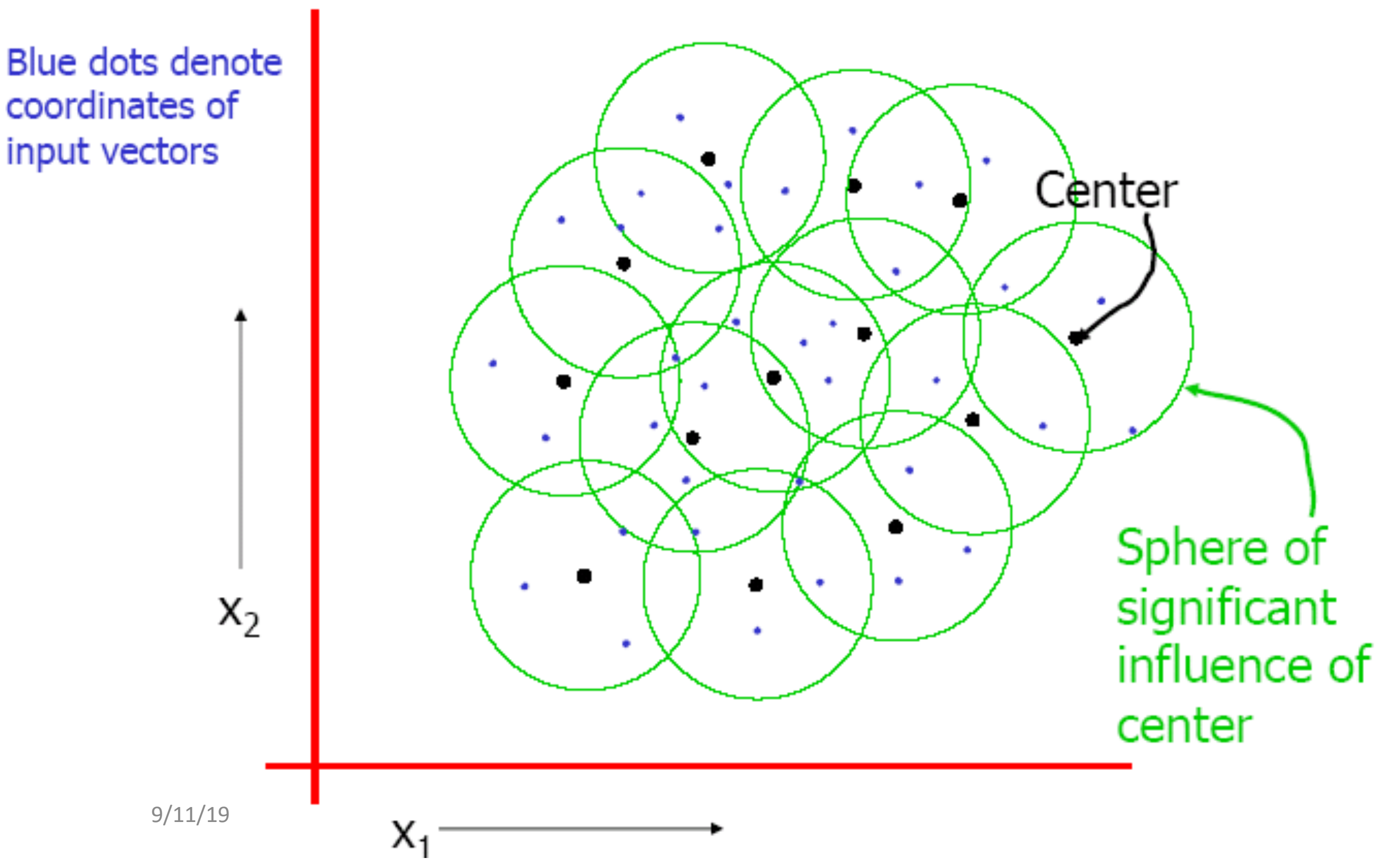
e.g. 2D Good and Bad RBF Basis

Contour view

- A Bad set of predefined 2D RBFs



- A good set of 2D predefined RBF basis



Today: Multivariate Linear Regression with basis Expansion

Task: y

↓

Representation: x, f()

↓

Score Function: L()

↓

Search/Optimization : argmin()

↓

Models, Parameters :

Regression: y continuous

↓

Y = Weighted linear sum of (X basis expansion)

↓

Sum of Squared Error (Least Squared)

↓

Normal Equation / GD / SGD

↓

Regression coefficients

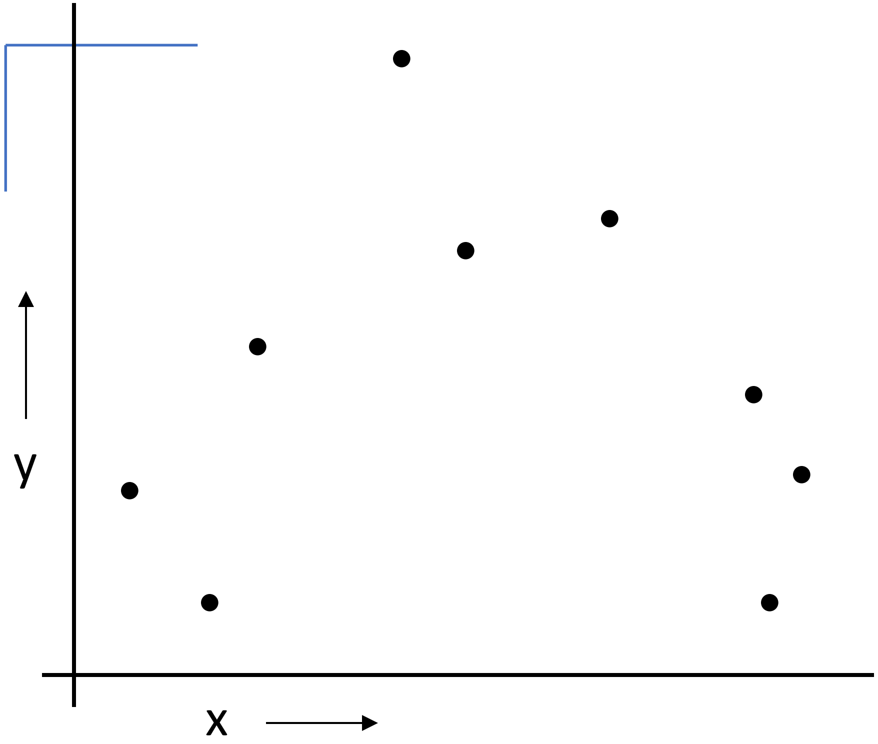
$$\hat{y} = \varphi(x)^T \theta$$

$\varphi(x)$: Which and what type?

Main issues: Model Selection

- How to select the right model?
  - 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?

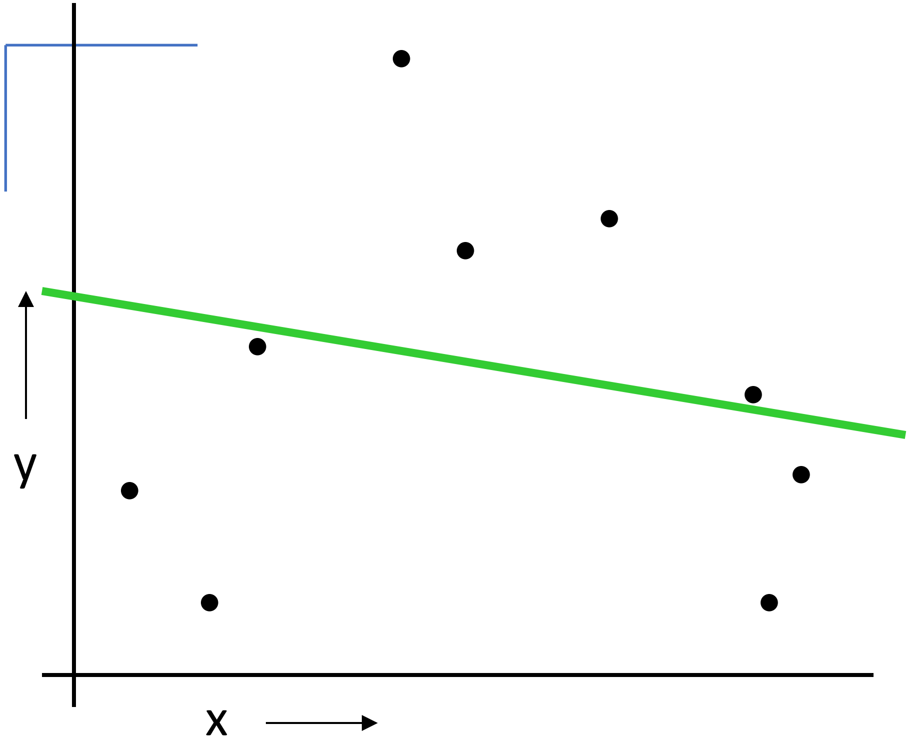
To Avoid: Overfitting or Underfitting



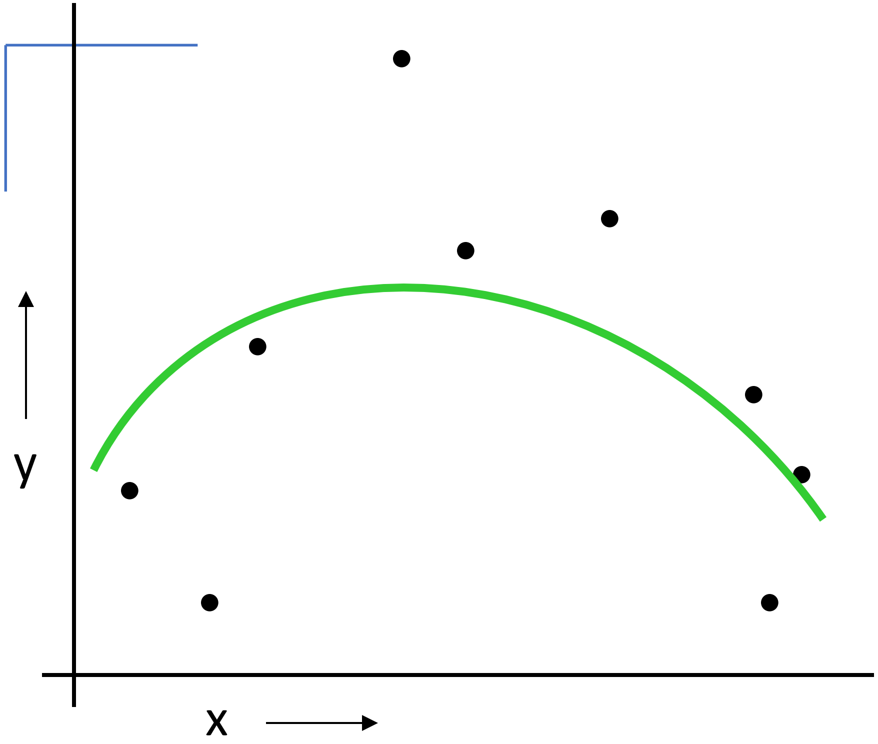
Can we learn a regression f from the data?

Let’s consider three methods…

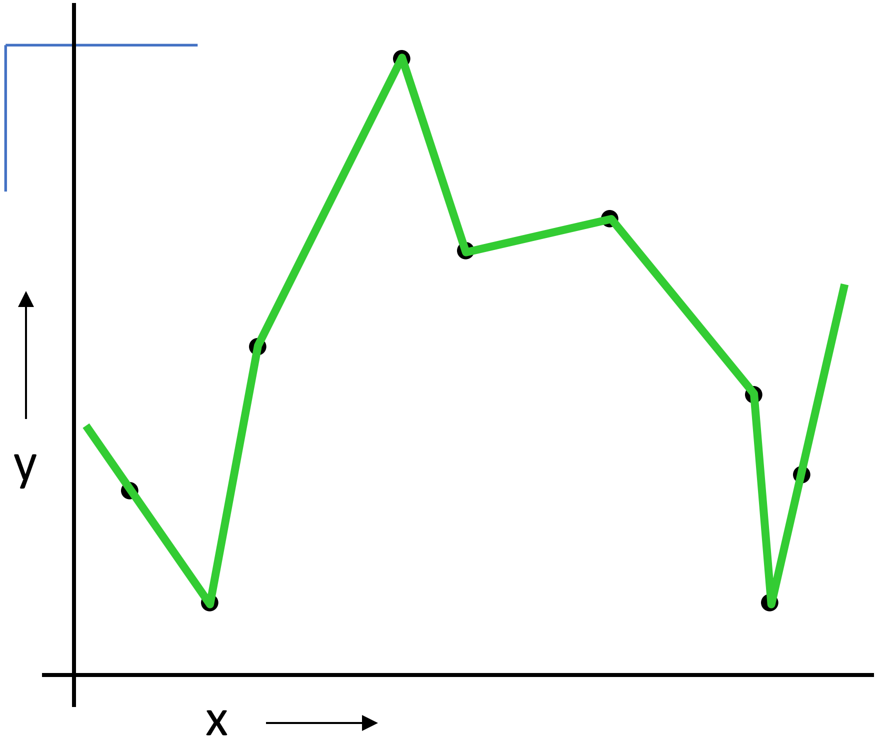
Linear Regression



Quadratic Regression

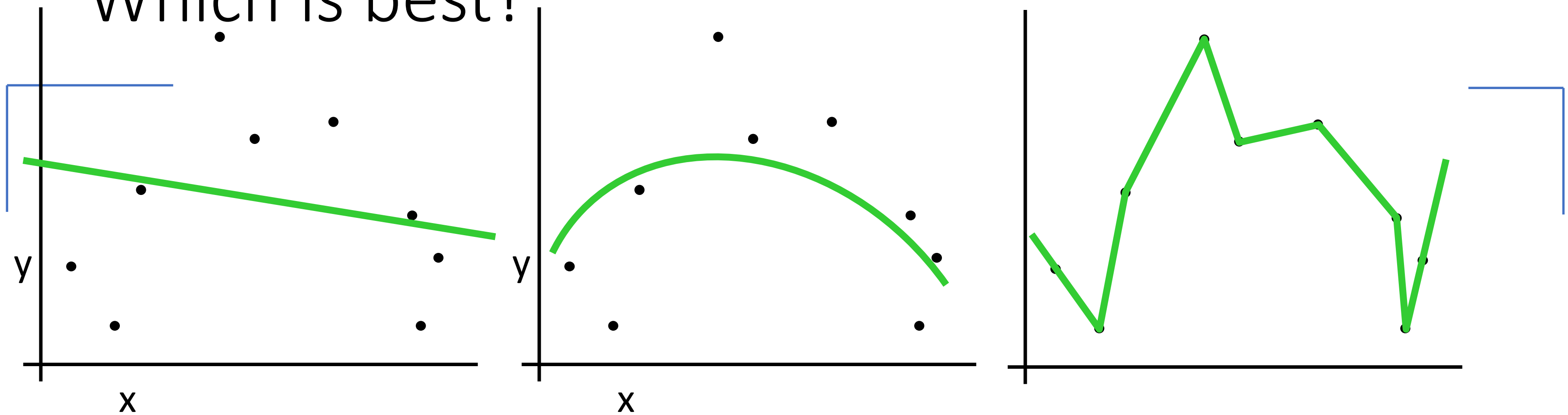


Join-the-dots



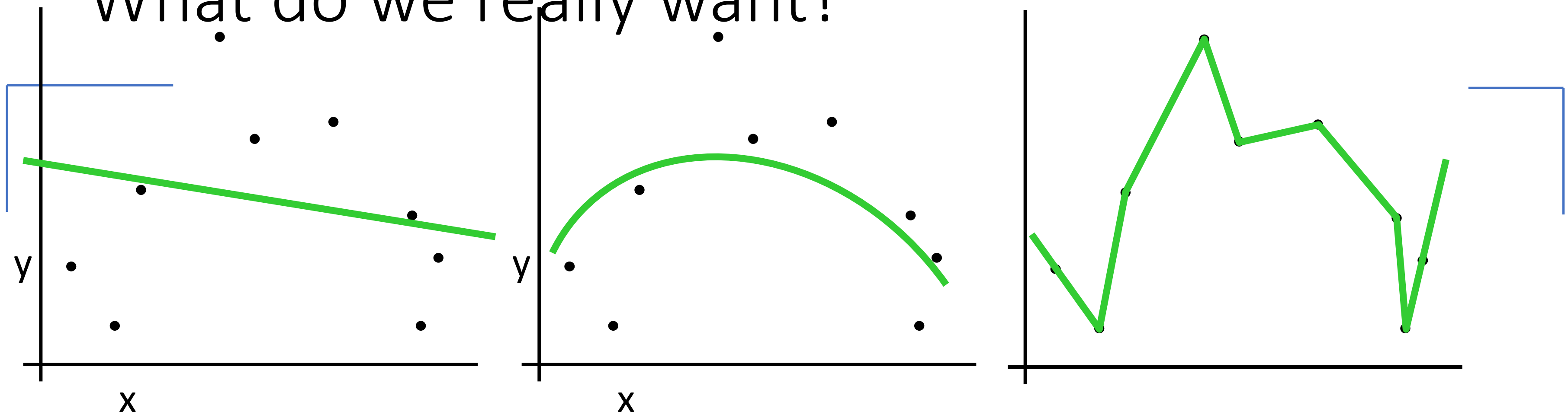
Also known as piecewise linear nonparametric regression if that makes you feel better

Which is best?



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

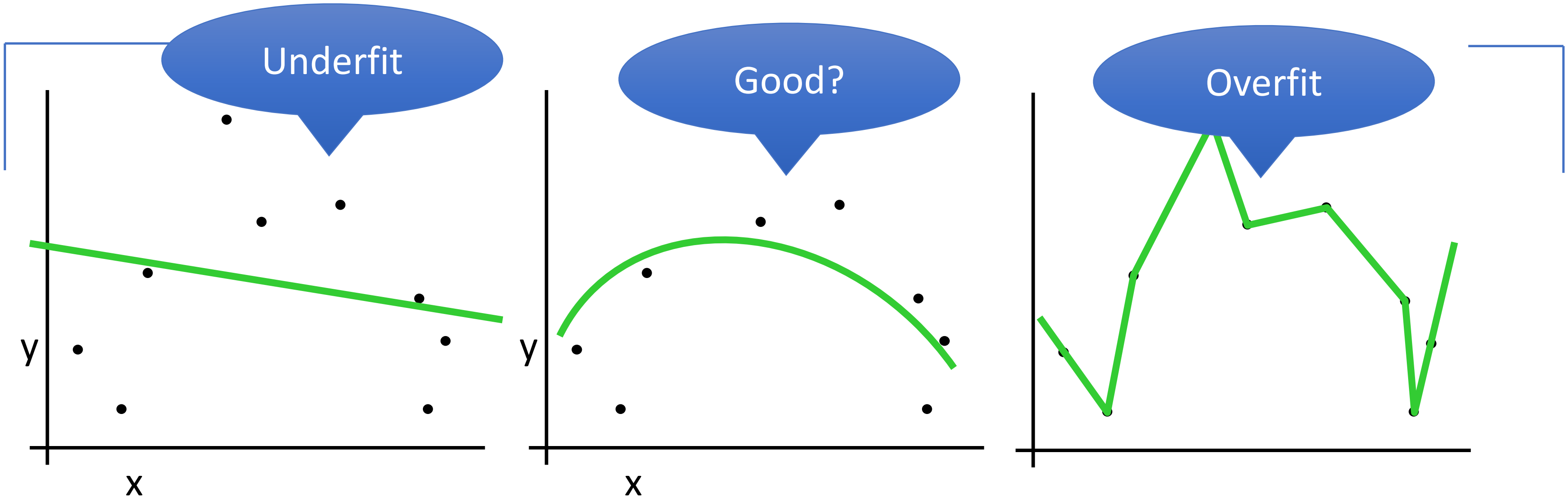
What do we really want?



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 to Select?

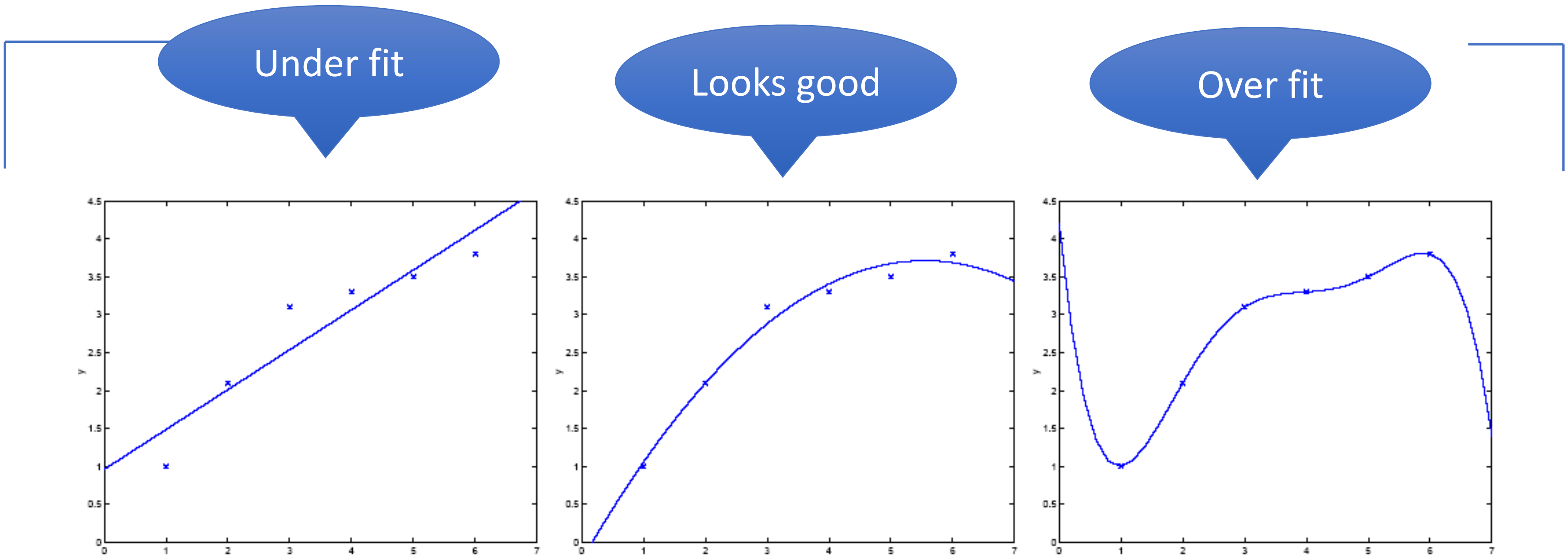


Why not choose the method with the best fit to the data?

K-fold Cross Validation / Train-Test /

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

What Model Order to Select?



$y = \theta_0 + \theta_1 x$

$y = \theta_0 + \theta_1 x + \theta_2 x^2$

$y = \sum_{j=0}^{5} \theta_j x^j$

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

K-fold Cross Validation / Train-Test /

Choice-I: Train-Test (Leave m out)

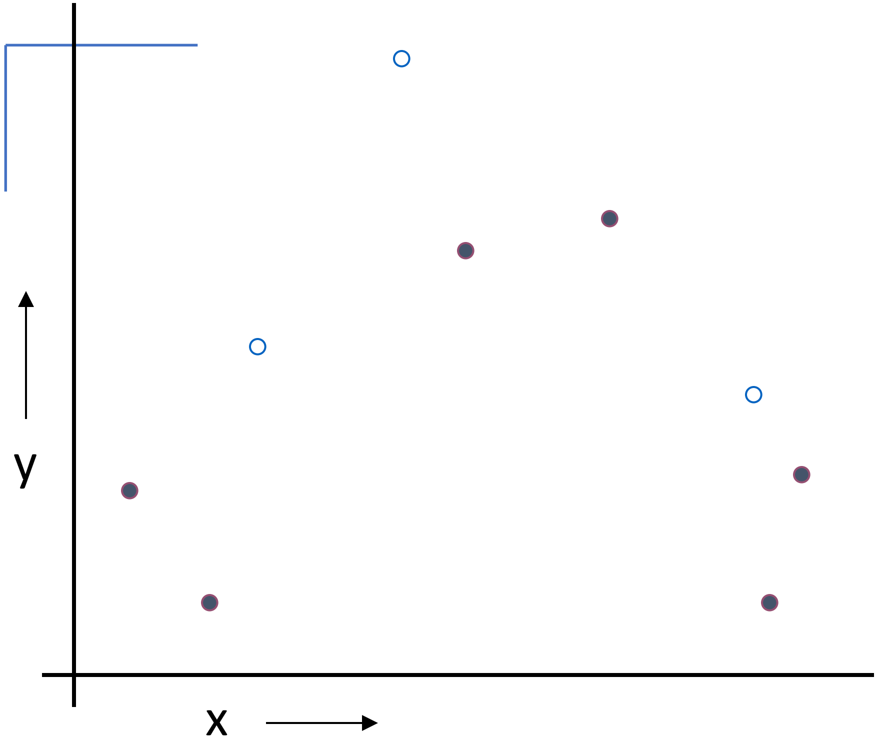
training dataset

$$\mathbf{X}_{train} = \begin{bmatrix} -- & \mathbf{x}_1^T & -- \\ -- & \mathbf{x}_2^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_n^T & -- \end{bmatrix} \qquad \vec{y}_{train} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

test dataset

$$\mathbf{X}_{test} = \begin{bmatrix} -- & \mathbf{x}_{n+1}^T & -- \\ -- & \mathbf{x}_{n+2}^T & -- \\ \vdots & \vdots & \vdots \\ -- & \mathbf{x}_{n+m}^T & -- \end{bmatrix} \qquad \vec{y}_{test} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ \vdots \\ y_{n+m} \end{bmatrix}$$

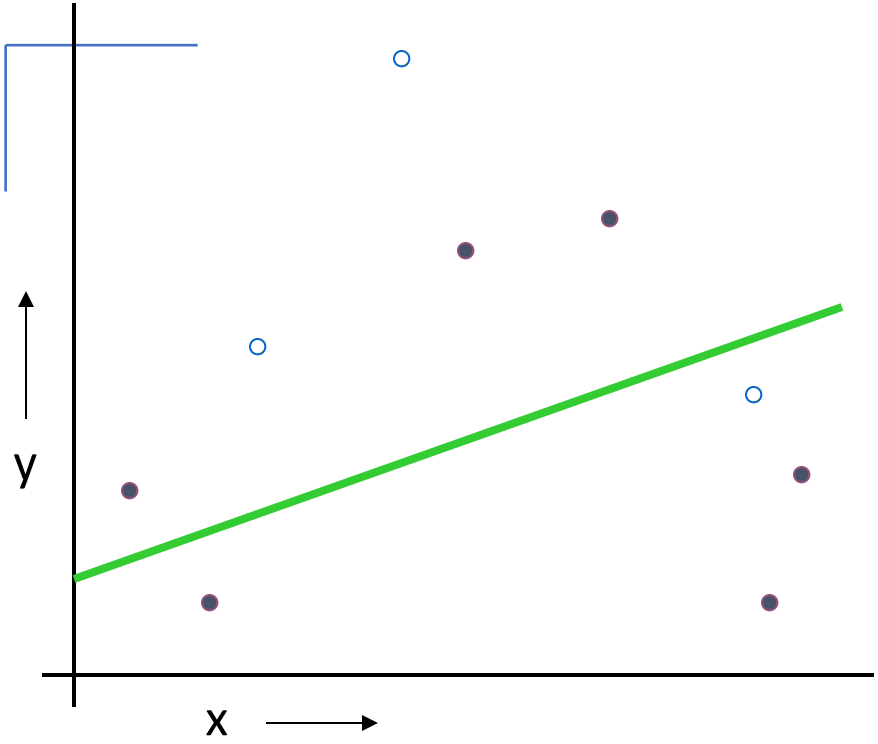
The test set method



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

Credit: Prof. Andrew Moore

The test set method

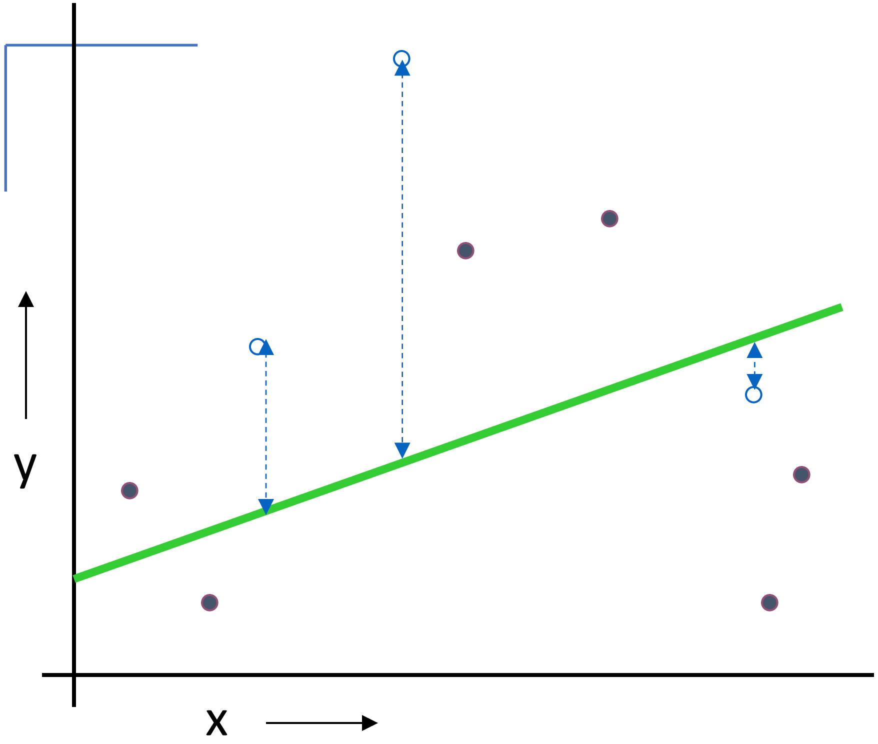


1. Randomly choose some percentage like 30% of the labeled data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set

(Linear regression example)

Credit: Prof. Andrew Moore

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

(Linear regression example)
Mean Squared Error = 2.4

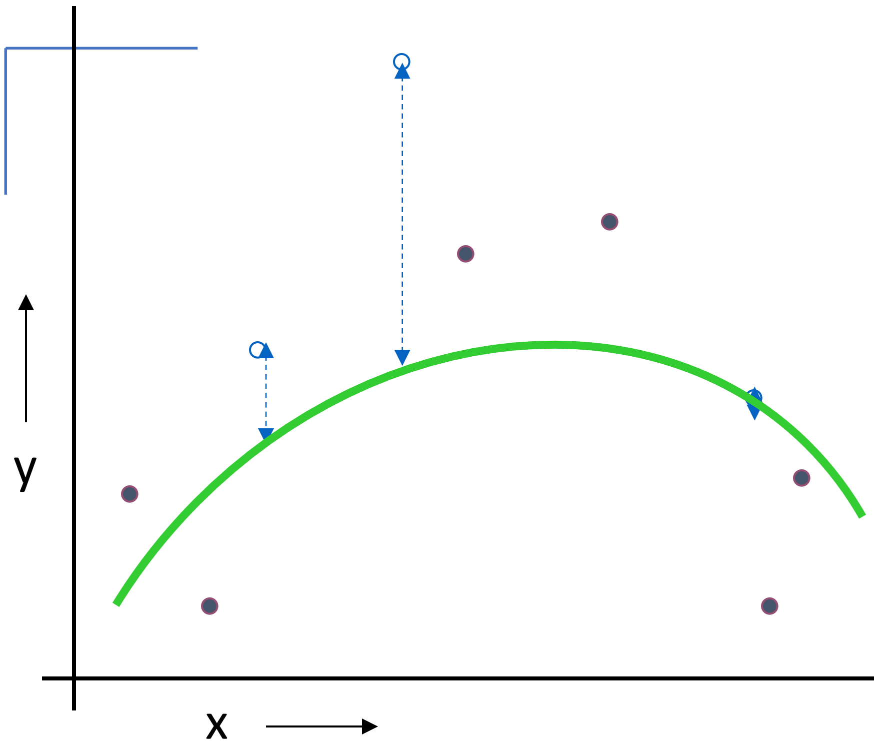
Credit: Prof. Andrew Moore

e.g. for Regression Models

- Testing 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 test set method

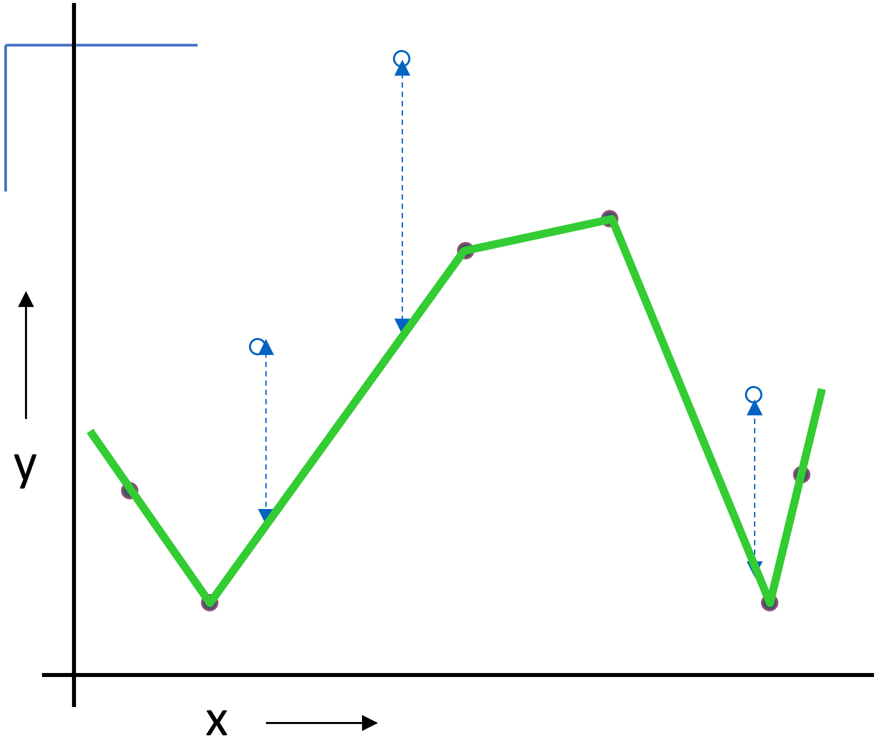


1. Randomly choose 30% of the data to be in a test set
2. The remainder is a training set
3. Perform your regression on the training set
4. Estimate your future performance with the test set

(Quadratic regression example)
Mean Squared Error = 0.9

Credit: Prof. Andrew Moore

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

Credit: Prof. Andrew Moore

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"

Credit: Prof. Andrew Moore

Choice-II: k-Fold Cross Validation

- Problem of train-test: in many cases we don’t have enough data to set aside a test set

- Solution: Each data point is used both as train and test

- Common types:
  - K-fold cross-validation (e.g. K=5, K=10)
  - Leave-one-out cross-validation (LOOCV, i.e., k=n)

e.g. By k=10 fold Cross Validation

- Divide data into 10 equal pieces
- 9 pieces as training set, the rest 1 as test set
- Collect the scores from each test
- 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

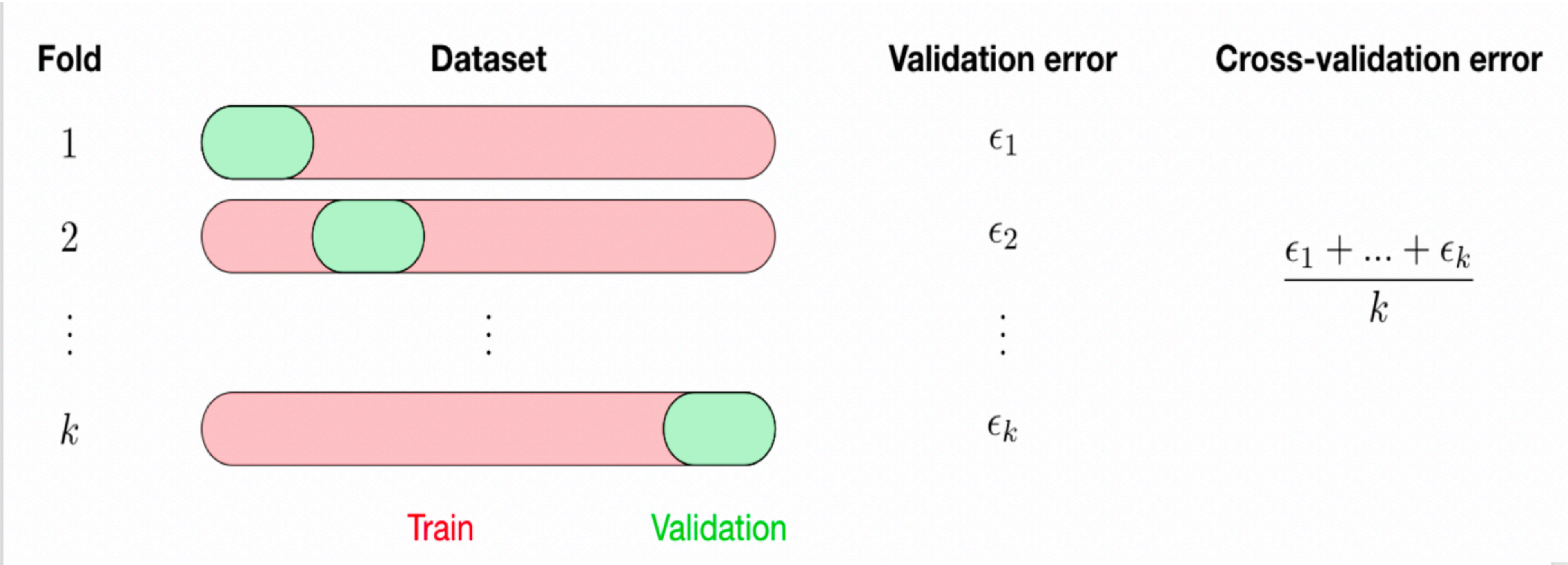
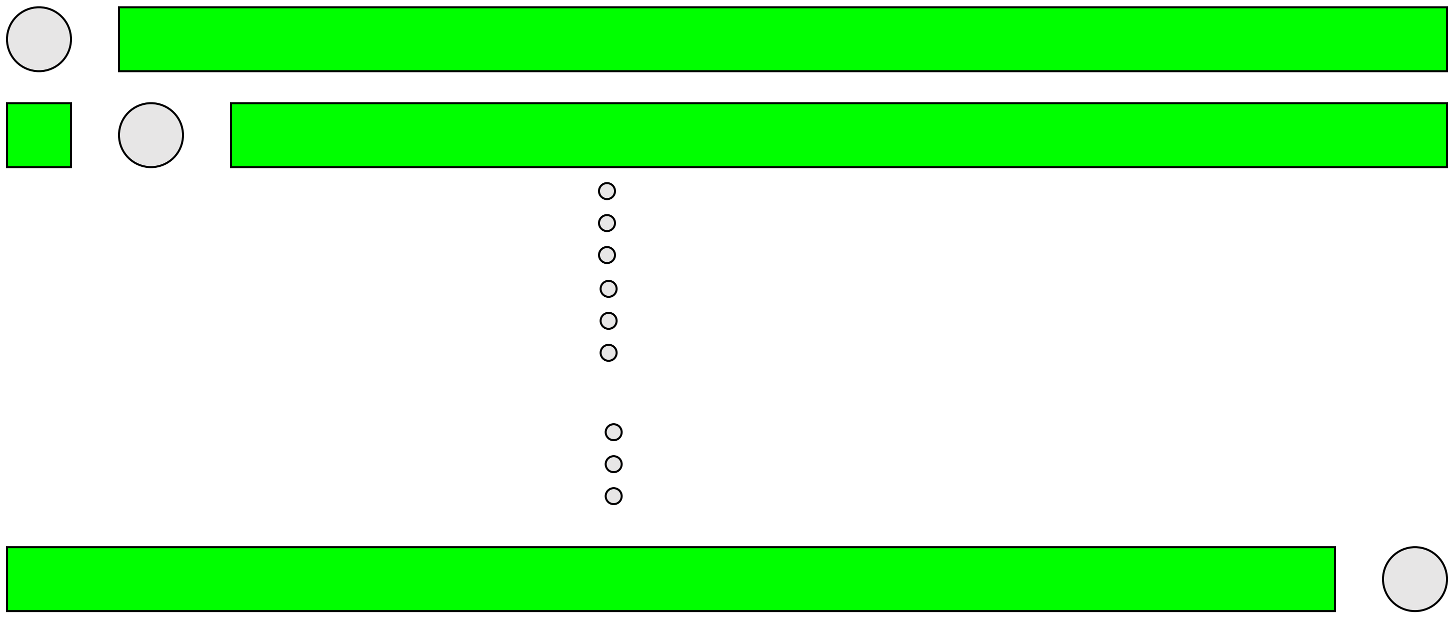


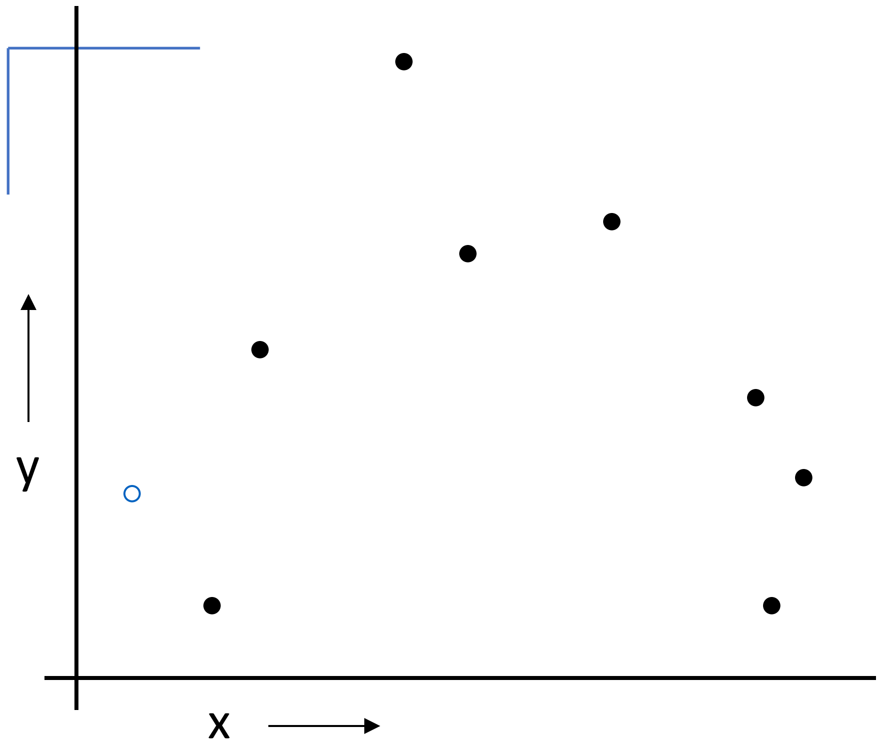
Image Credit: Stanford Machine Learning course

e.g. Leave-one-out / LOOCV (n-fold cross validation)

n is num. of data samples



LOOCV (Leave-one-out Cross Validation)

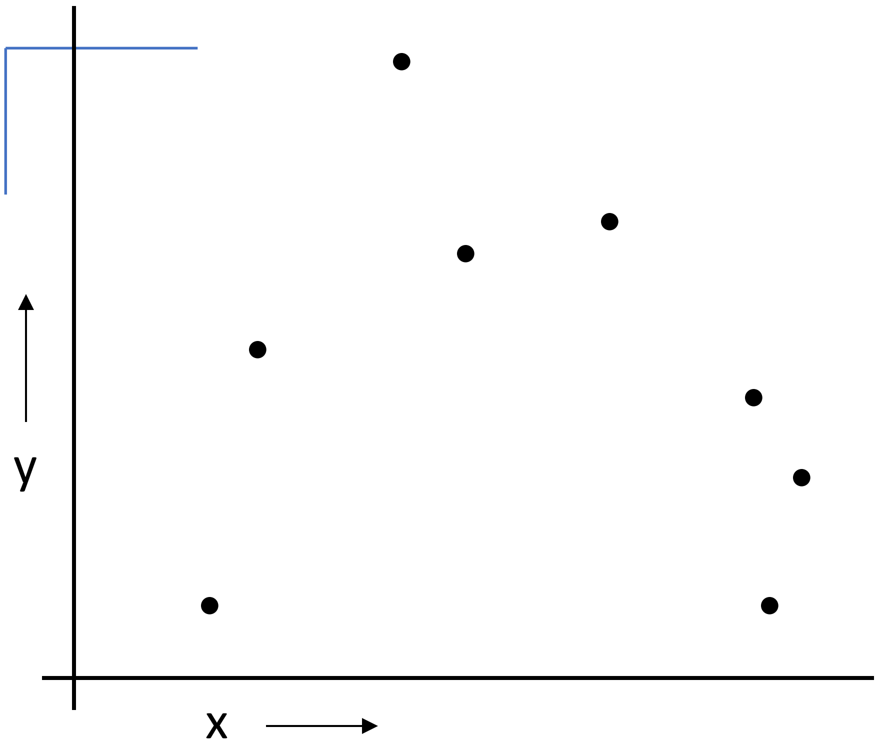


For k=1 to n

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

Credit: Prof. Andrew Moore

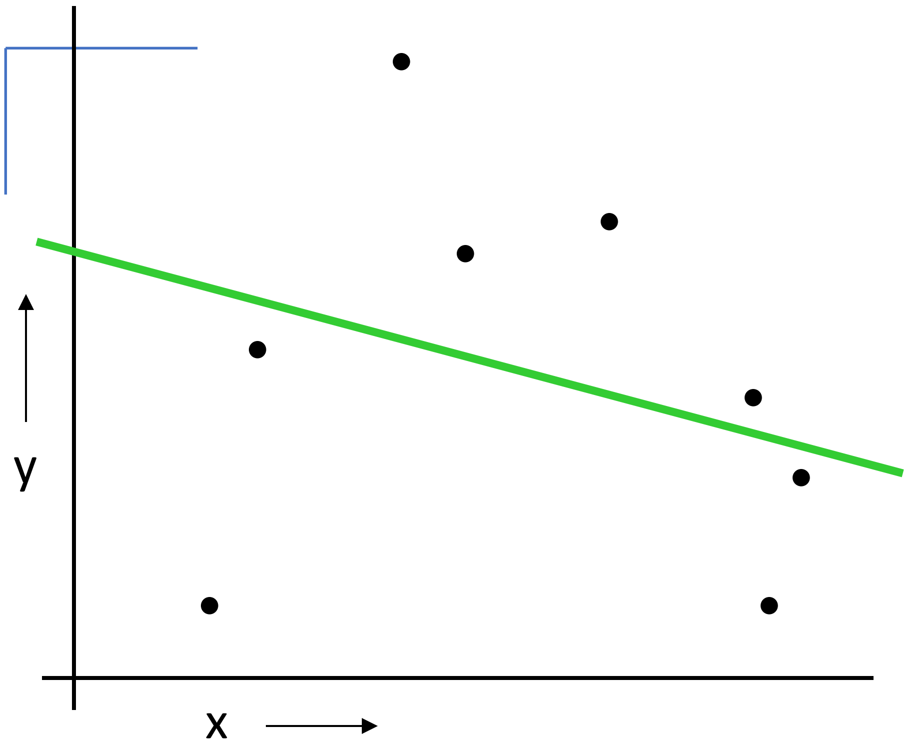
LOOCV (Leave-one-out Cross Validation)



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)

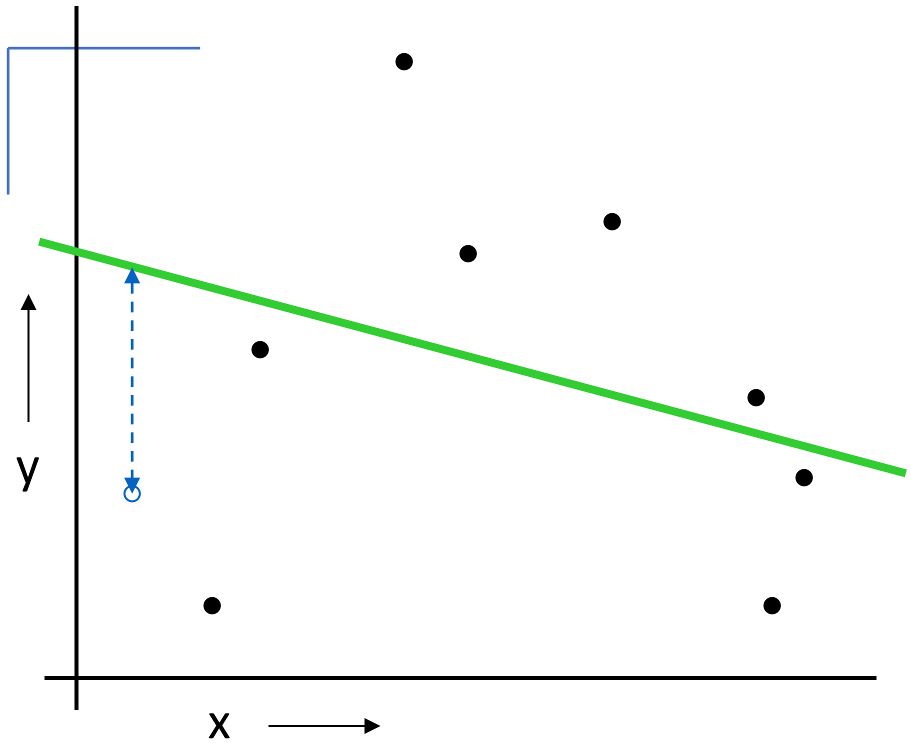


For k=1 to n

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

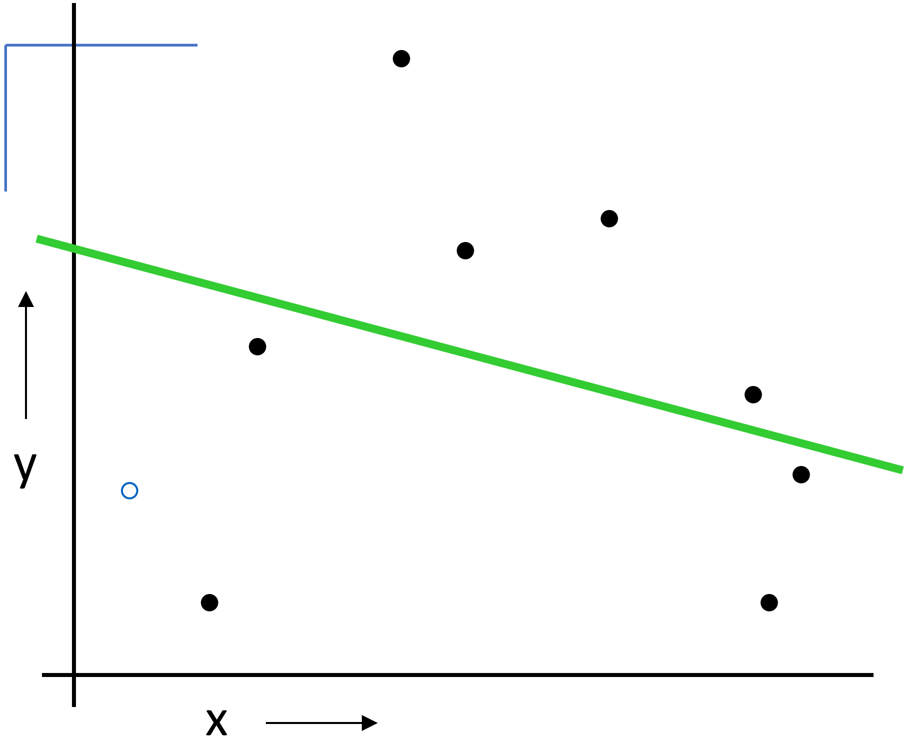
LOOCV (Leave-one-out Cross Validation)



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)

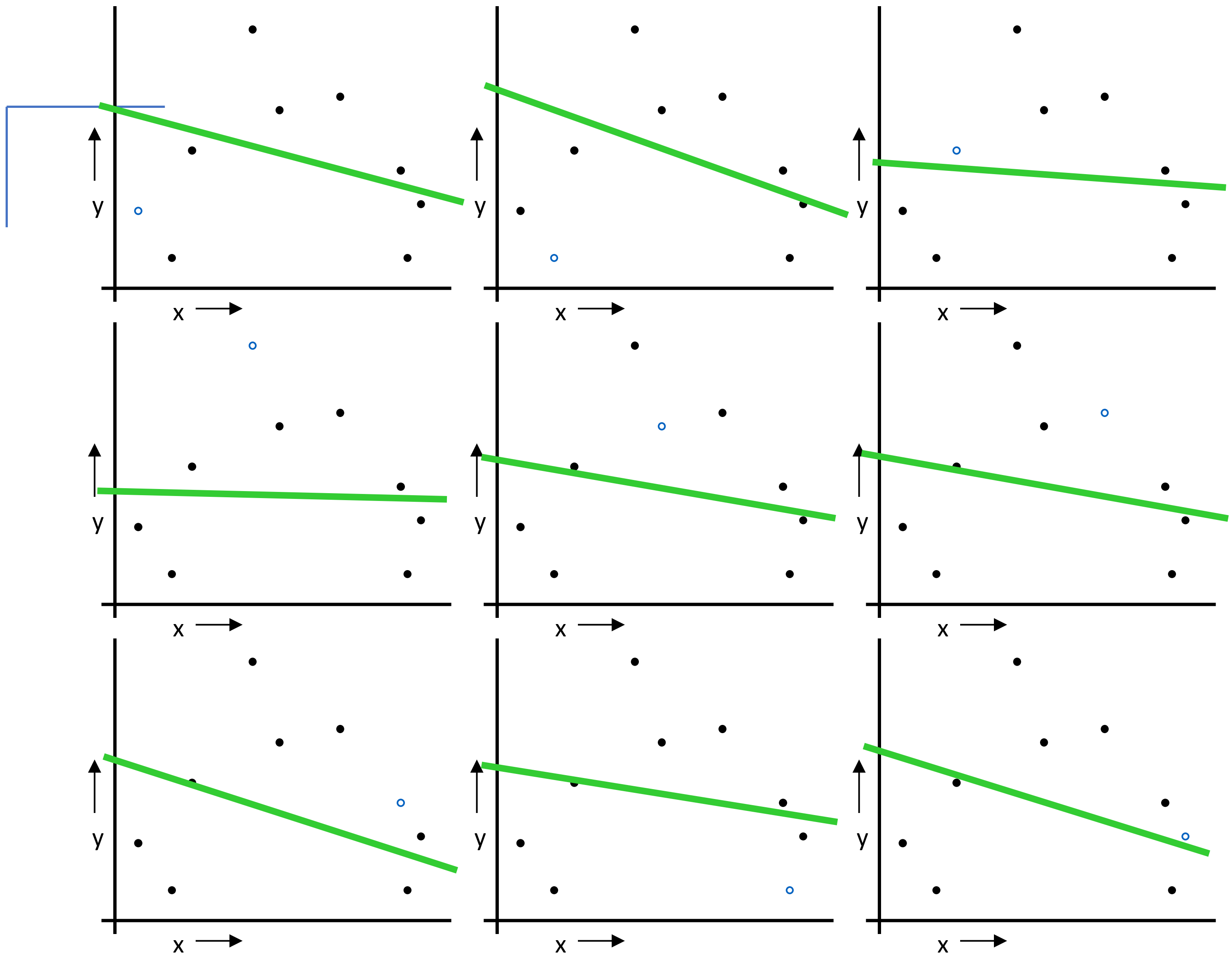


For k=1 to R

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

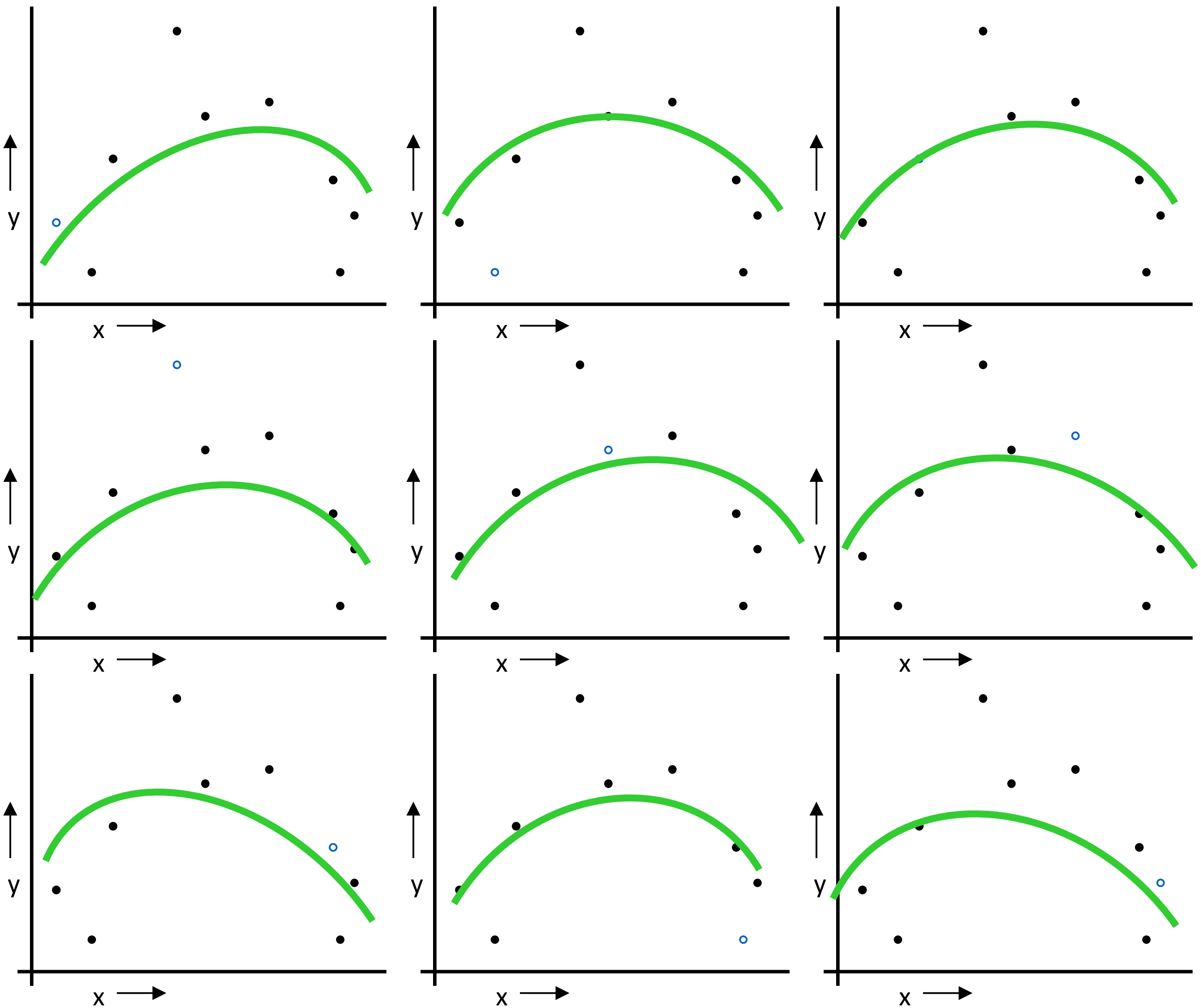
1. Let (x_k,y_k) be the k^{th} record
2. Temporarily remove (x_k,y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k,y_k)

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

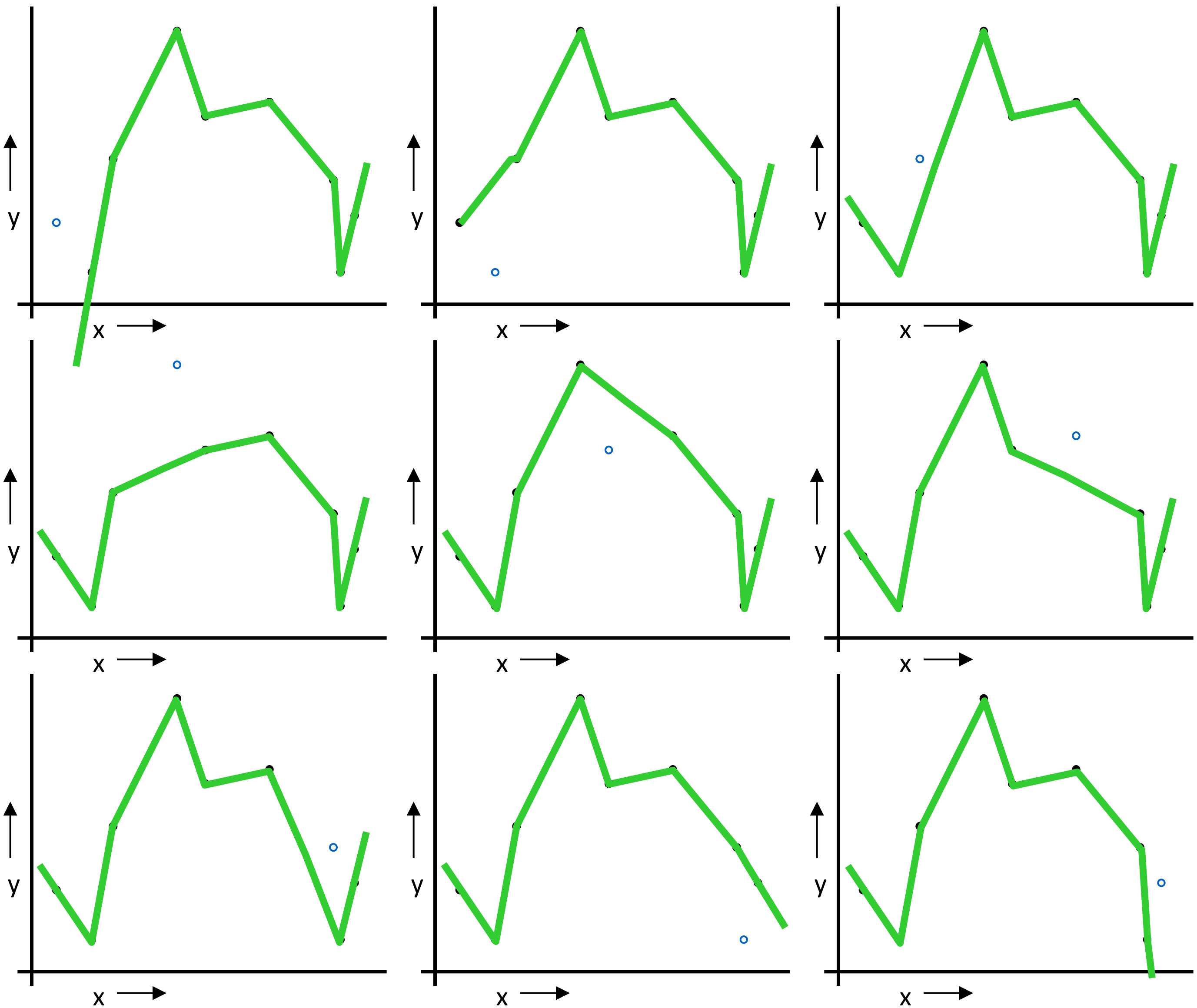
1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining n-1 datapoints
4. Note your error (x_k, y_k)

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

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

$MSE_{LOOCV}=3.33$

Credit: Prof. Andrew Moore

Which kind of Cross Validation?

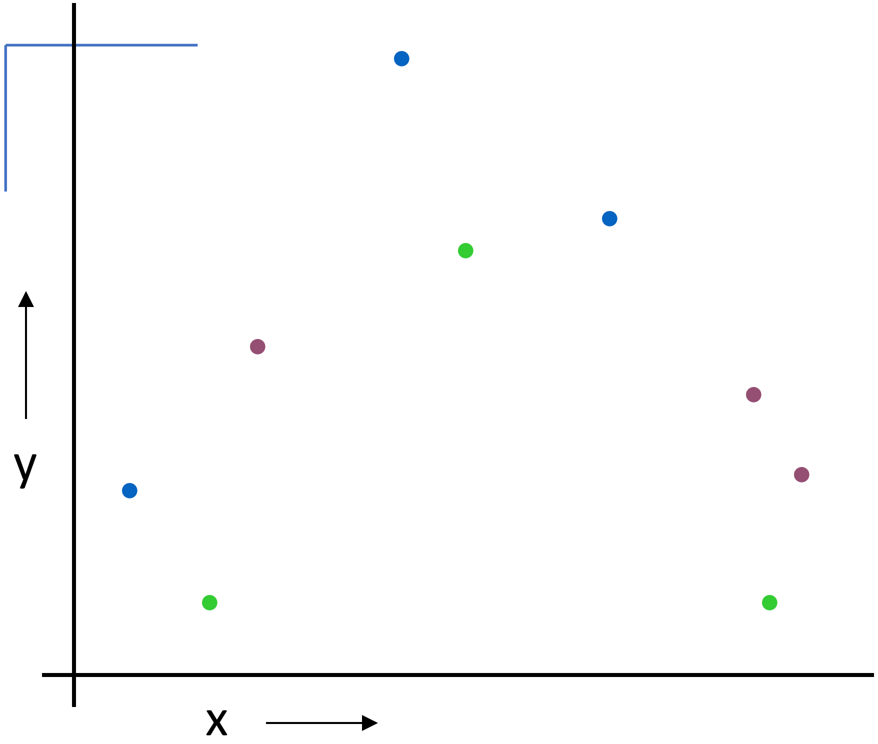
	Downside	Upside
Test-set	Variance: unreliable estimate of future performance	Cheap
Leave-one-out	Expensive. Has some weird behavior	Doesn't waste data

..can we get the best of both worlds?

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)

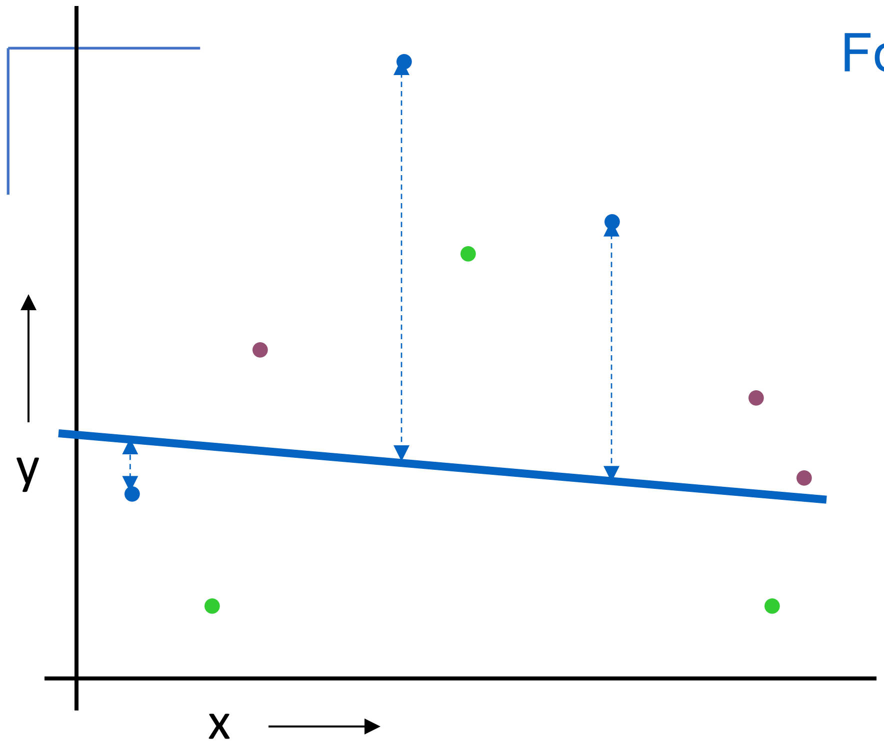


Credit: Prof. Andrew Moore

k-fold Cross Validation

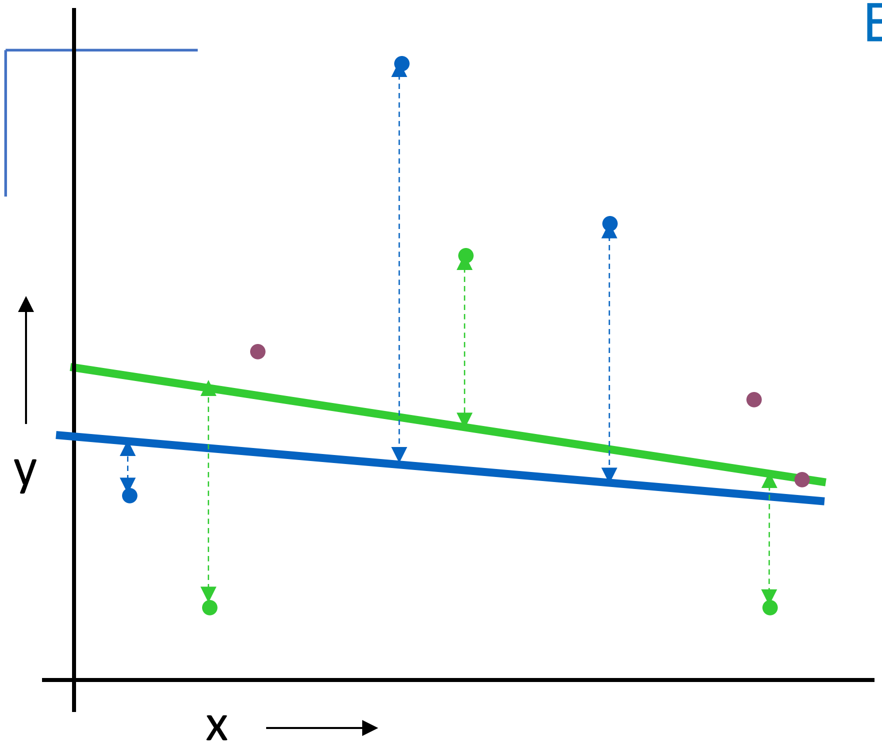
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

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



Credit: Prof. Andrew Moore

k-fold Cross Validation



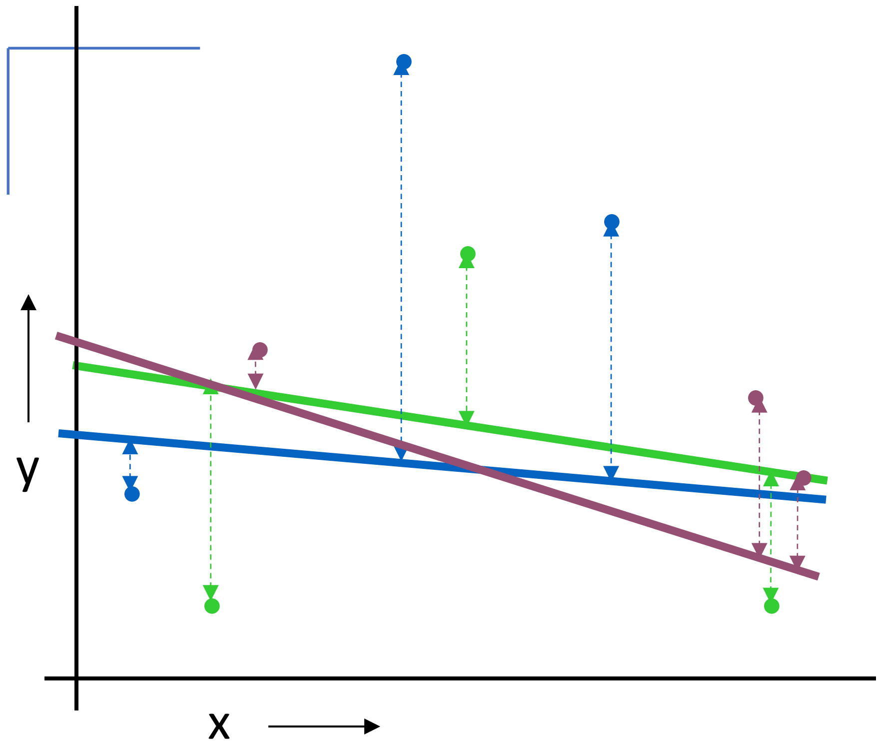
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

For the blue partition: Train on all the points not in the red partition. Find the 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.

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 test-set sum of errors on the red points.

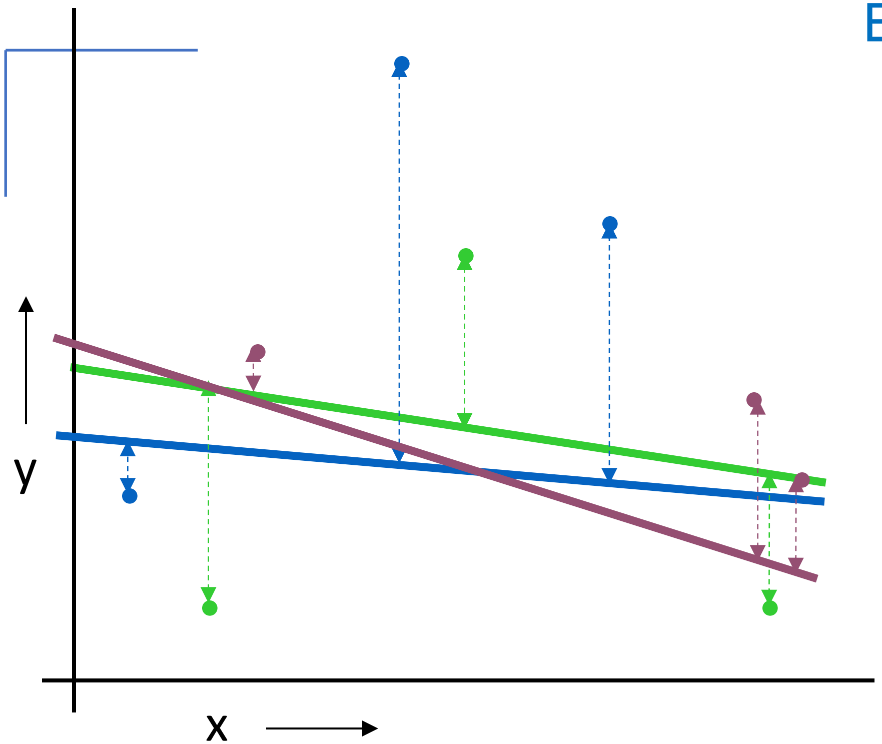
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

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

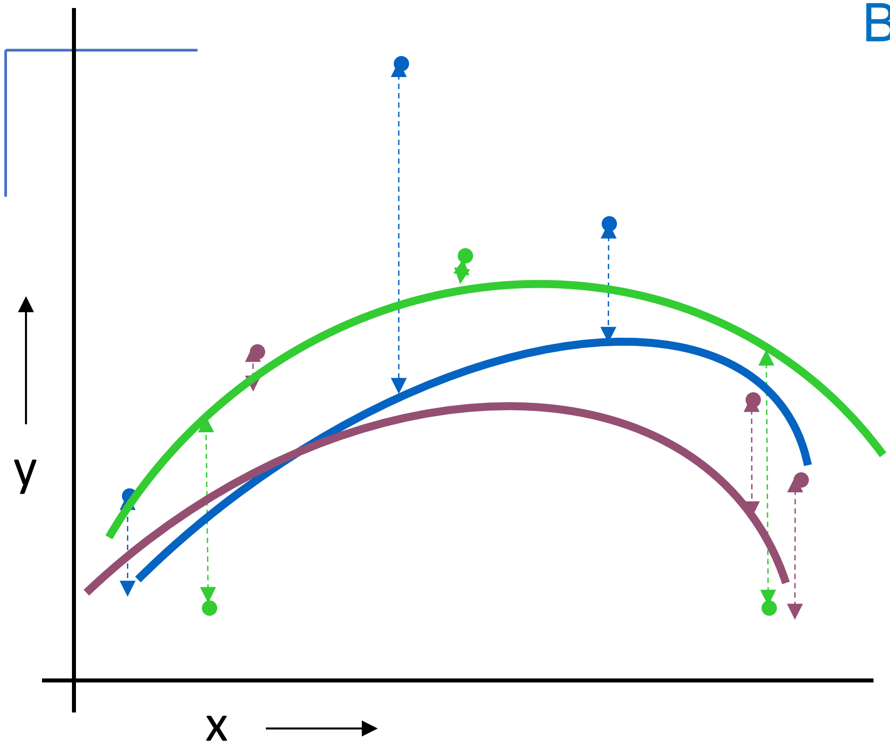
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

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

Then report the mean error

Credit: Prof. Andrew Moore

k-fold Cross Validation



Quadratic Regression $MSE_{3FOLD}=1.11$

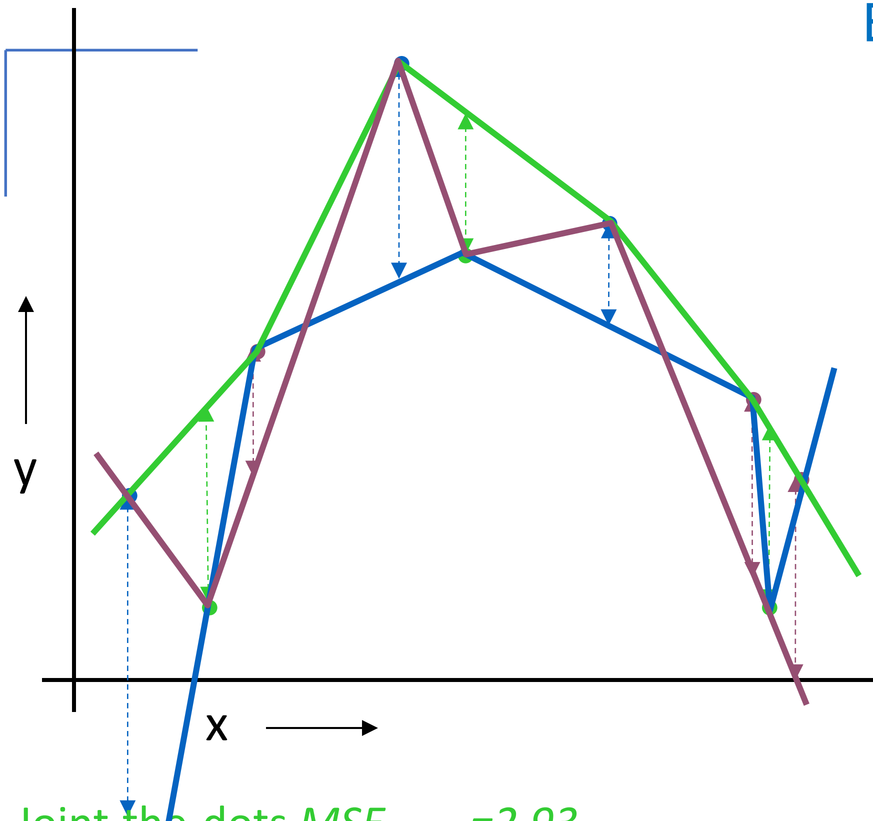
Randomly break the dataset into k partitions (in our example we'll have k=3 partitions colored Purple Green and Blue)

- 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 purple partition: Train on all the points not in the purple partition. Find the test-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 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

Credit: Prof. Andrew Moore

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 n times.
3-fold	Wastier than 10-fold. More Expensive than test set style	better than test-set
n-fold	Identical to Leave-one-out	

CV-based Model Selection

- We’re trying to decide which algorithm/model 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			⍰
4	f_4			
5	f_5			
6	f_6			

Credit: Prof. Andrew Moore

Next: More Regression (supervised)

- ❑ Four ways to train / perform optimization for linear regression models
  - ❑ ~~Normal Equation~~
  - ❑ ~~Gradient Descent (GD)~~
  - ❑ ~~Stochastic GD~~
  - ❑ ~~Newton's method~~

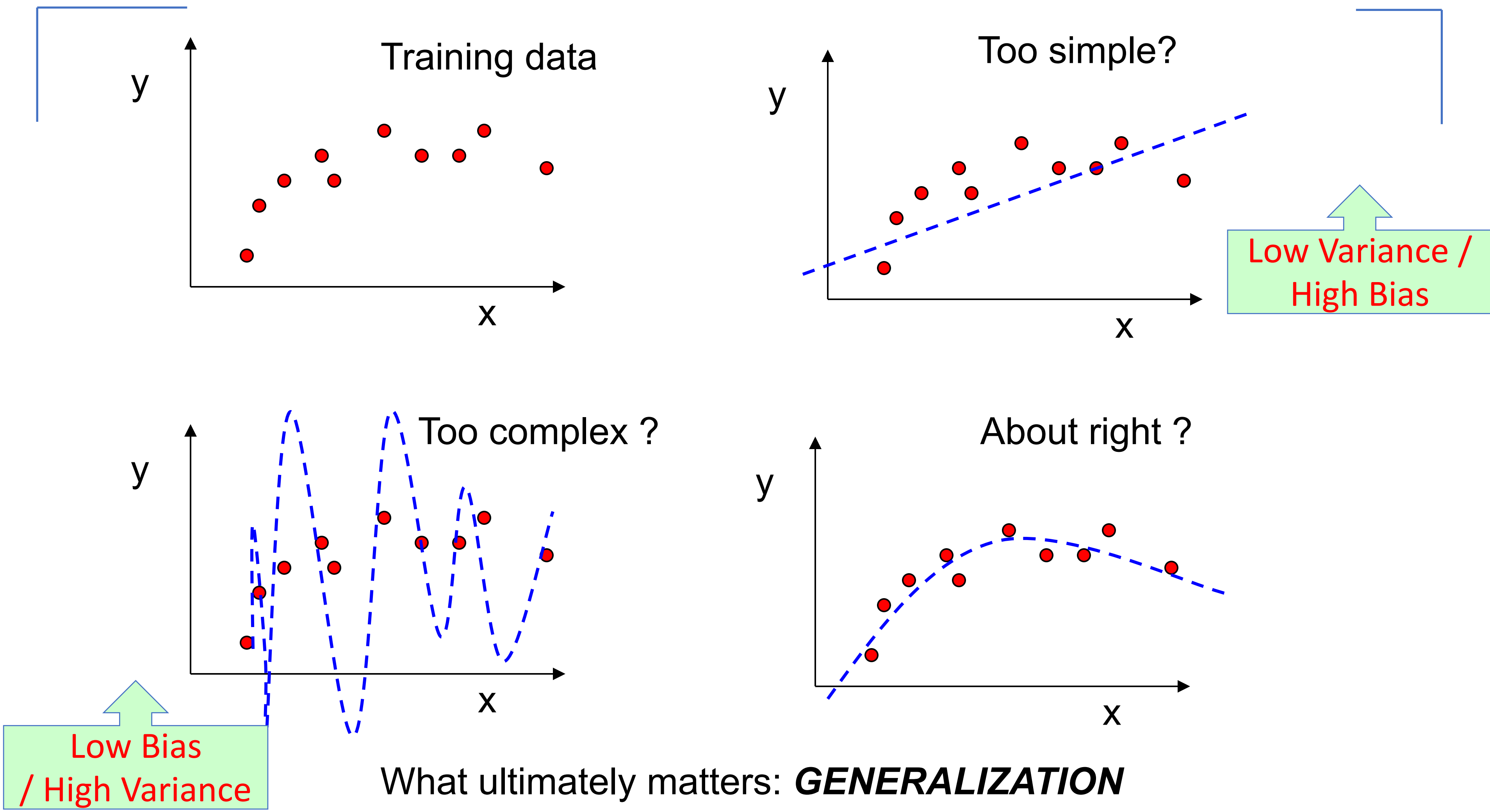
} variations of $\underset{\theta}{\operatorname{argmin}} L(\theta)$

- ❑ Supervised regression models
  - ❑ ~~Linear regression (LR)~~
  - ❑ ~~LR with non-linear basis functions~~
  - ❑ ~~Locally weighted LR~~
  - ❑ LR with Regularizations

} variations of $f(x)$

→ variations of $L(\theta)$

Later: Complexity versus Goodness of Fit



What ultimately matters: ***GENERALIZATION***

Lecture 5 Extra: K-Nearest Neighbor

Task	Regression/ classification
Representation	Local Smoothness
Score Function	NA
Search/Optimization	NA
Models, Parameters	Training Samples

Lecture 5 Extra: Nonparametric Regression Models

- K-Nearest Neighbor (KNN) and Locally weighted linear regression are **non-parametric** algorithms.

- The (unweighted) linear regression algorithm that we saw earlier is known as a **parametric** learning algorithm
  - because it has a fixed, finite number of parameters which are fit to the data;
  - Once we've fit the *theta* and stored them away, we no longer need to keep the training data around to make future predictions.
  - In contrast, to make predictions using KNN or locally weighted linear regression, we need to keep the entire training set around.

- The term "non-parametric" (roughly) refers to the fact that the amount of knowledge we need to keep, in order to represent the hypothesis grows with linearly the size of the training set.

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