Lecture 5 pre-video

OLS is OLS, plus some model selection

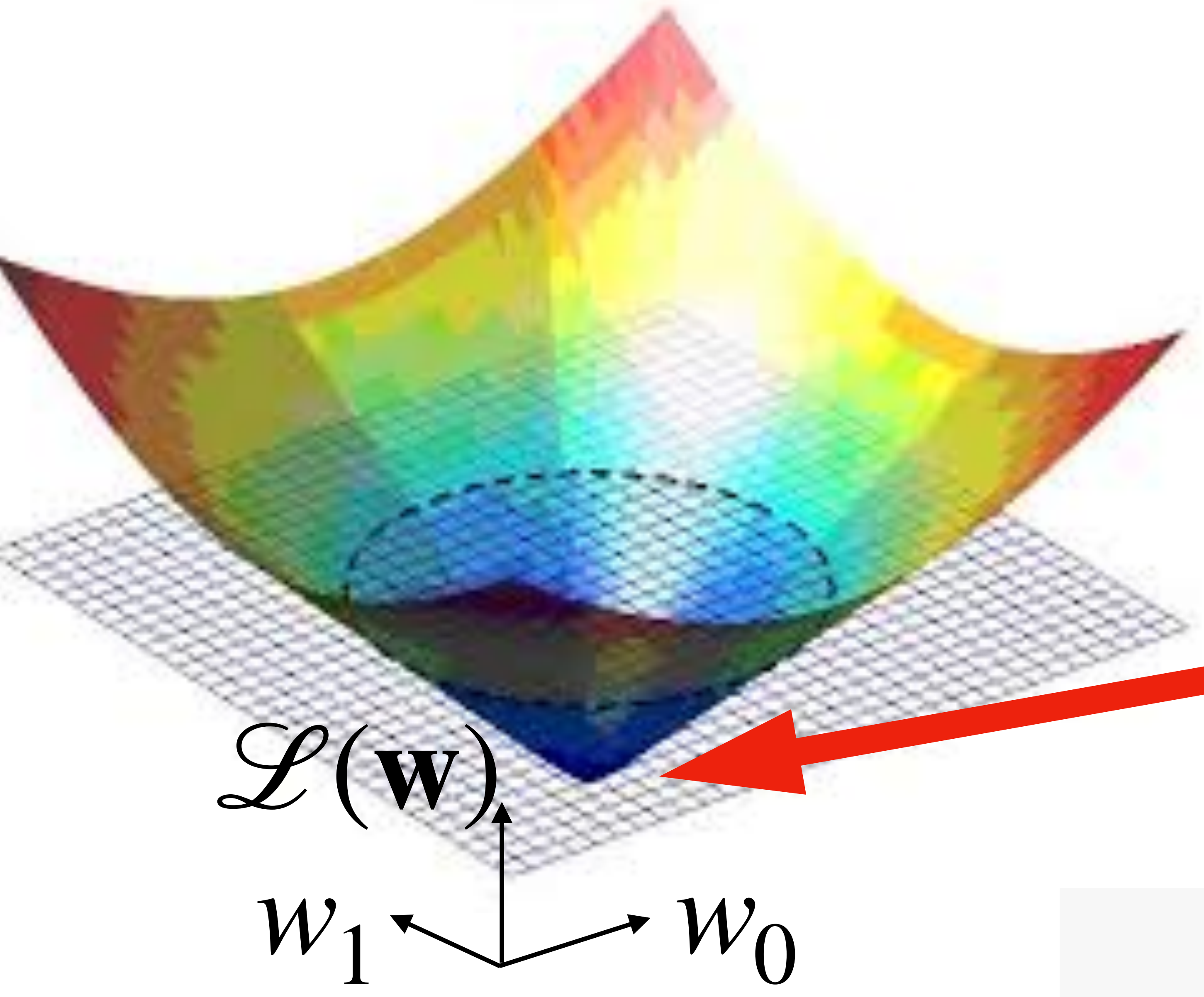
Ordinary least squares regression

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$
$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$
$$\mathscr{L}(\mathbf{w}) = \mathbf{e}^{\mathbf{T}}\mathbf{e}$$

$$\mathbf{X} = \begin{pmatrix} x_{0,0} & x_{0,1} & \dots & x_{0,d} \\ x_{1,0} & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ x_{n,0} & x_{n,1} & \dots & x_{n,d} \end{pmatrix} = \begin{pmatrix} 1 & x_{0,1} & \dots & x_{0,d} \\ 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{pmatrix}$$

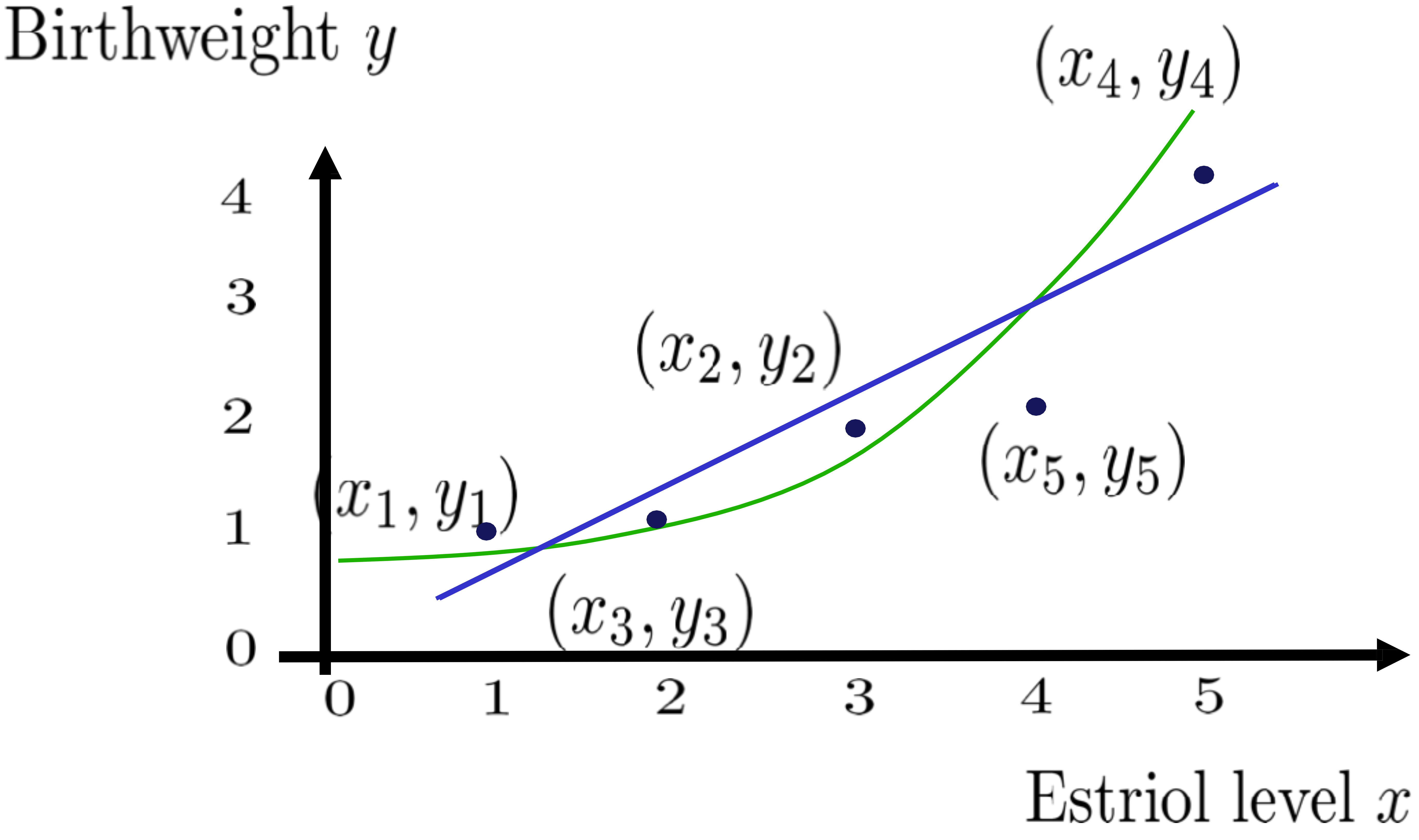
Convex because loss is quadratic in $\mathbf{w}$

$$\arg\min_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\mathbf{w})$$
$$\arg\min_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \mathbf{w}^* \text{ such that } \nabla \mathscr{L}(\mathbf{w}^*) = 0$$
$$\nabla \mathscr{L}(\mathbf{w}) = -2\mathbf{X}^{\mathbf{T}}\mathbf{y} + 2\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w}$$
$$\therefore \mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$



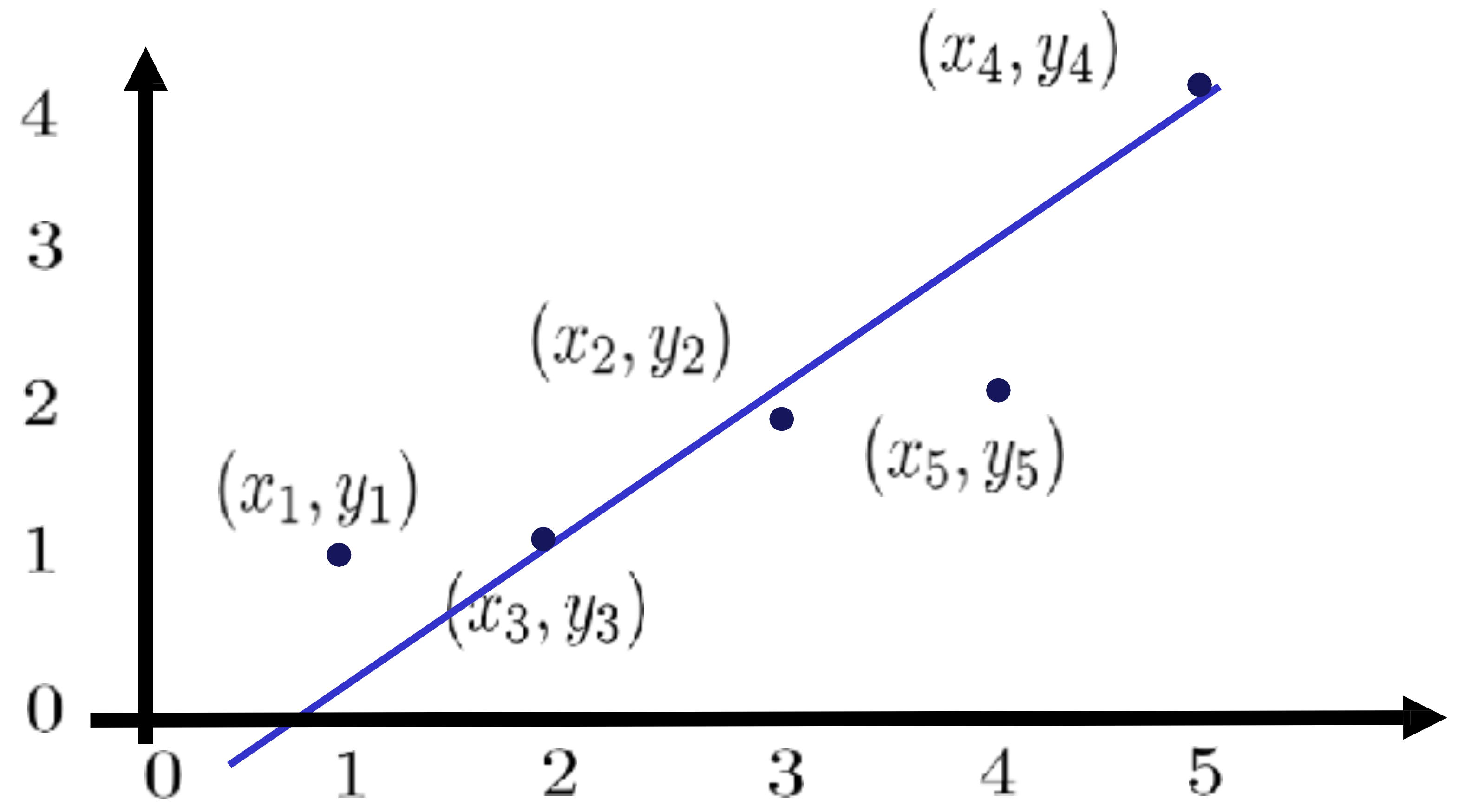
$$\mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix}$$

```
import numpy as np
from numpy.linalg import inv

psuedoinv = inv(X_train.T @ X_train ) @ X_train.T
w_star =  psuedoinv @ y_train
```

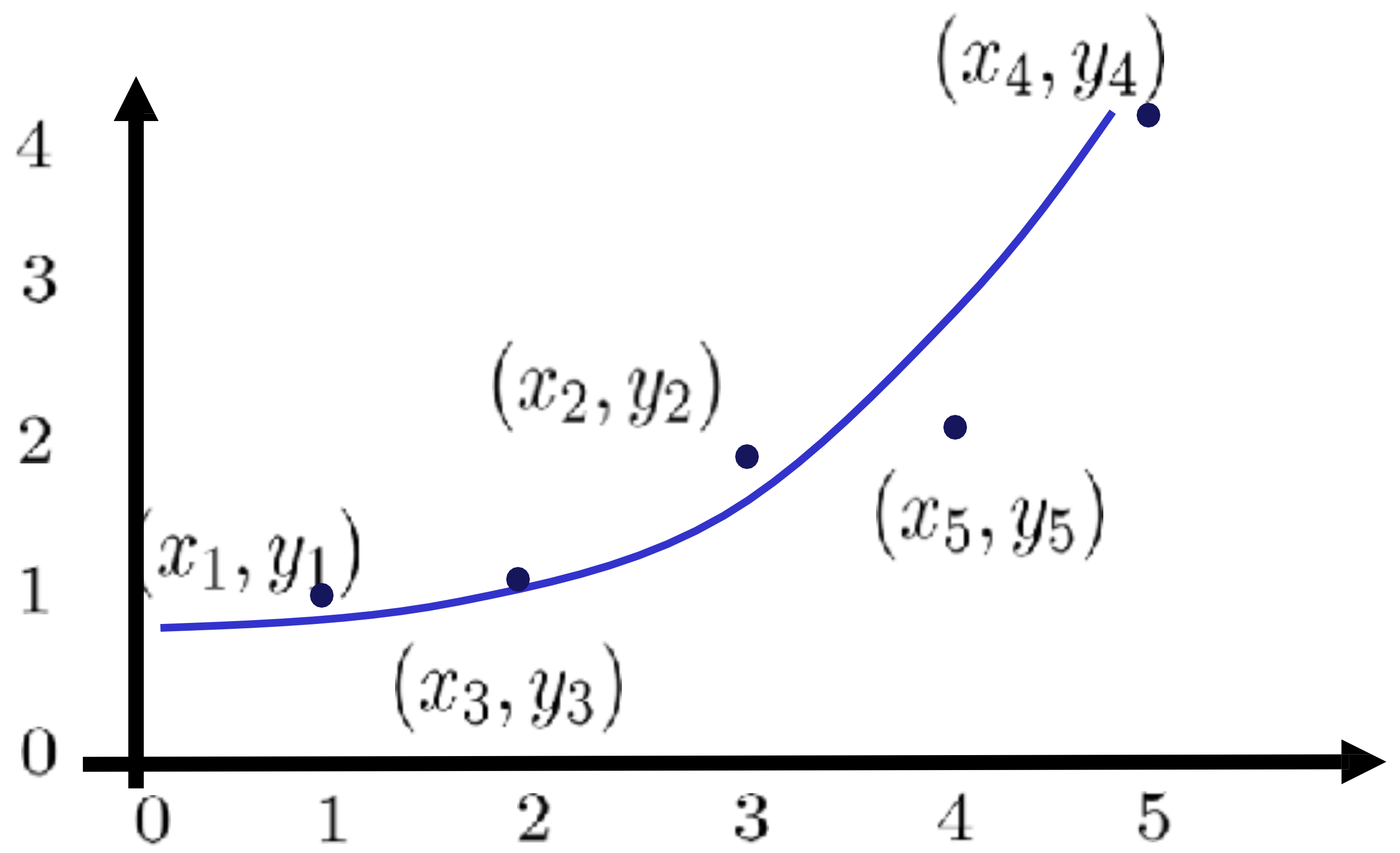


One obvious metric to compare models:

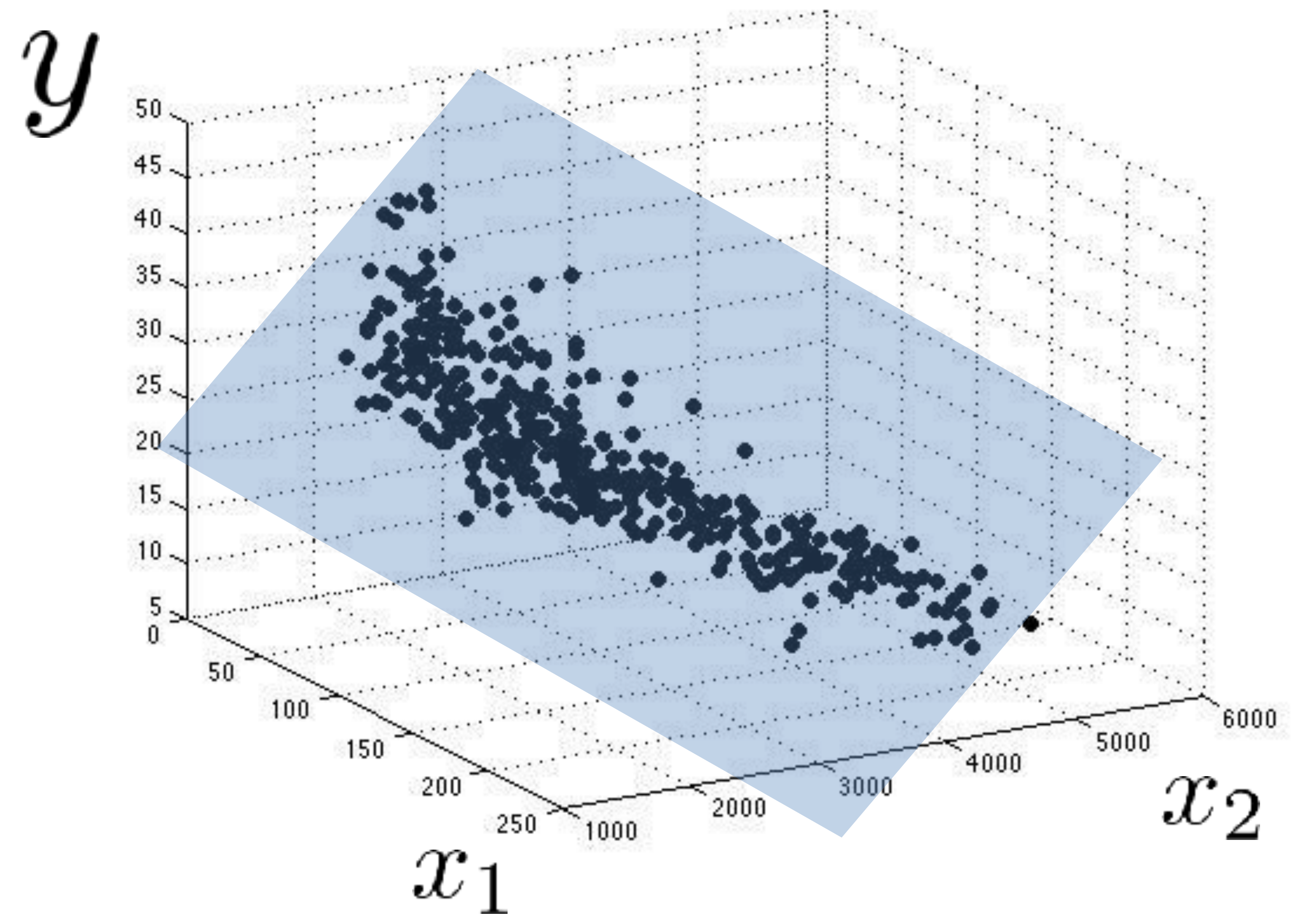
RSS itself, $\mathscr{L}(\mathbf{w})$



Univariate



Polynomial

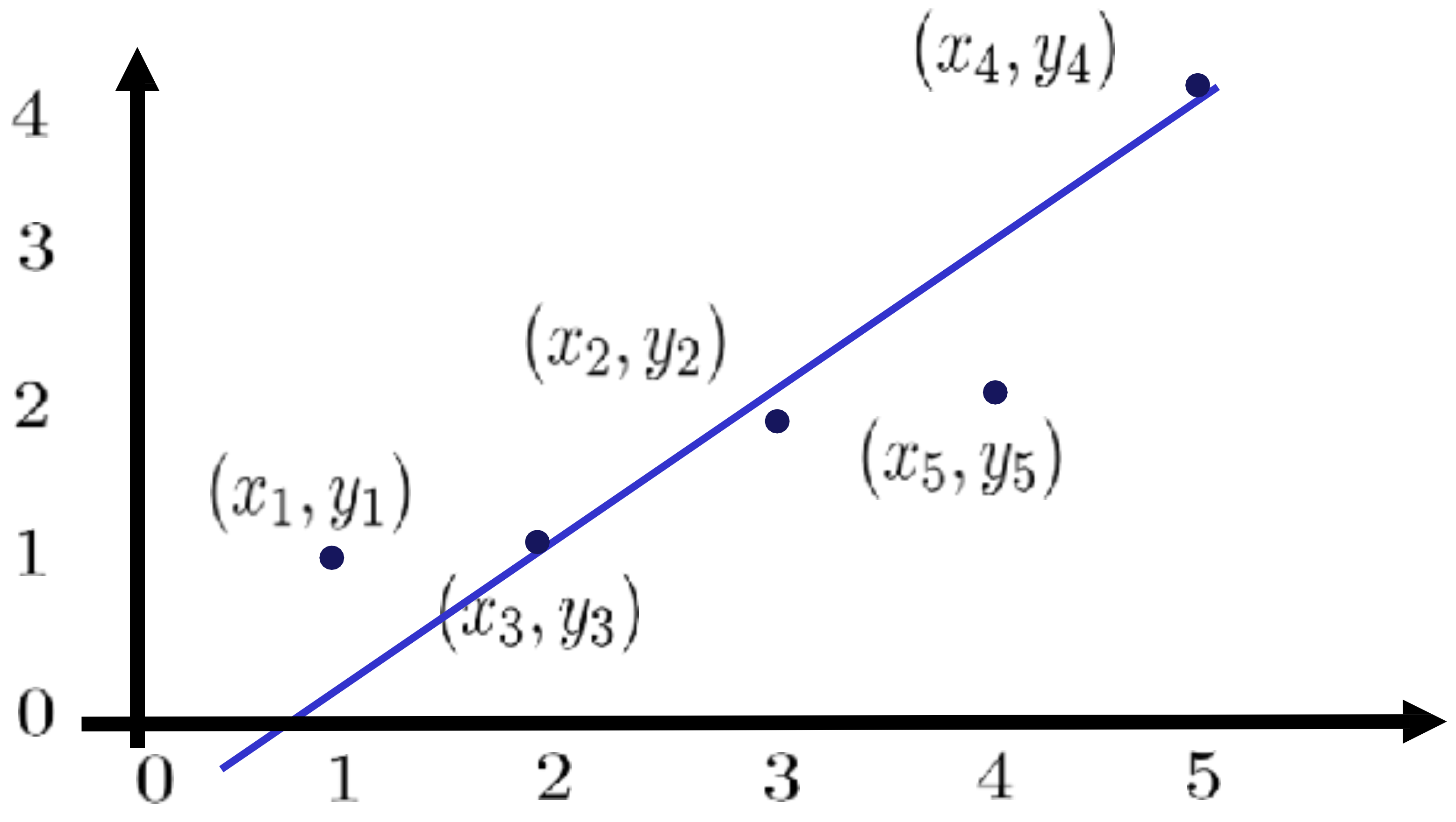


Multivariate

$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} \\ \mathbf{e} &= \mathbf{y} - \hat{\mathbf{y}} \\ \mathscr{L}(\mathbf{w}) &= \mathbf{e}^{\mathbf{T}}\mathbf{e} \end{aligned}$$

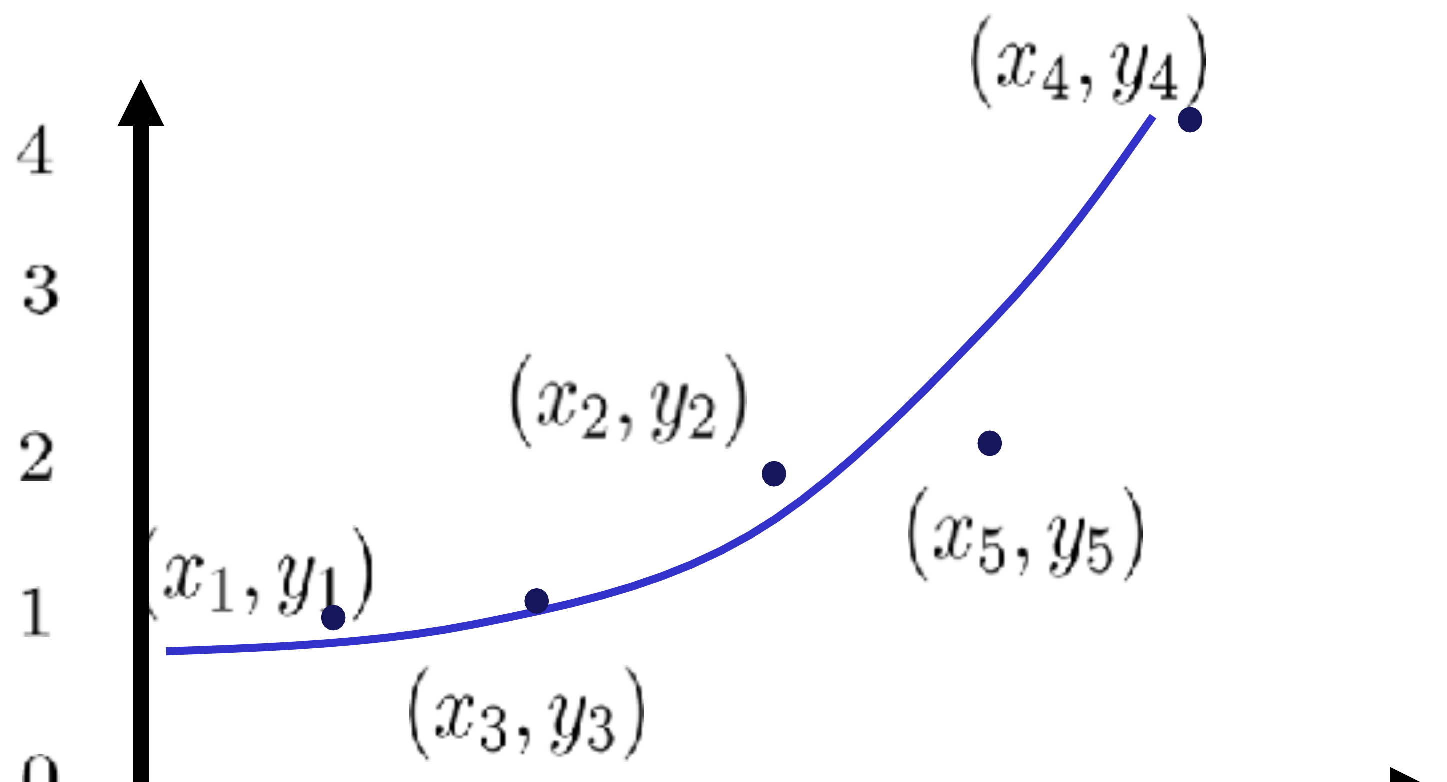
$$\mathbf{X} = \begin{pmatrix} 1 & x_{0,1} & \dots & x_{0,d} \\ 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{pmatrix}$$

$$\mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$



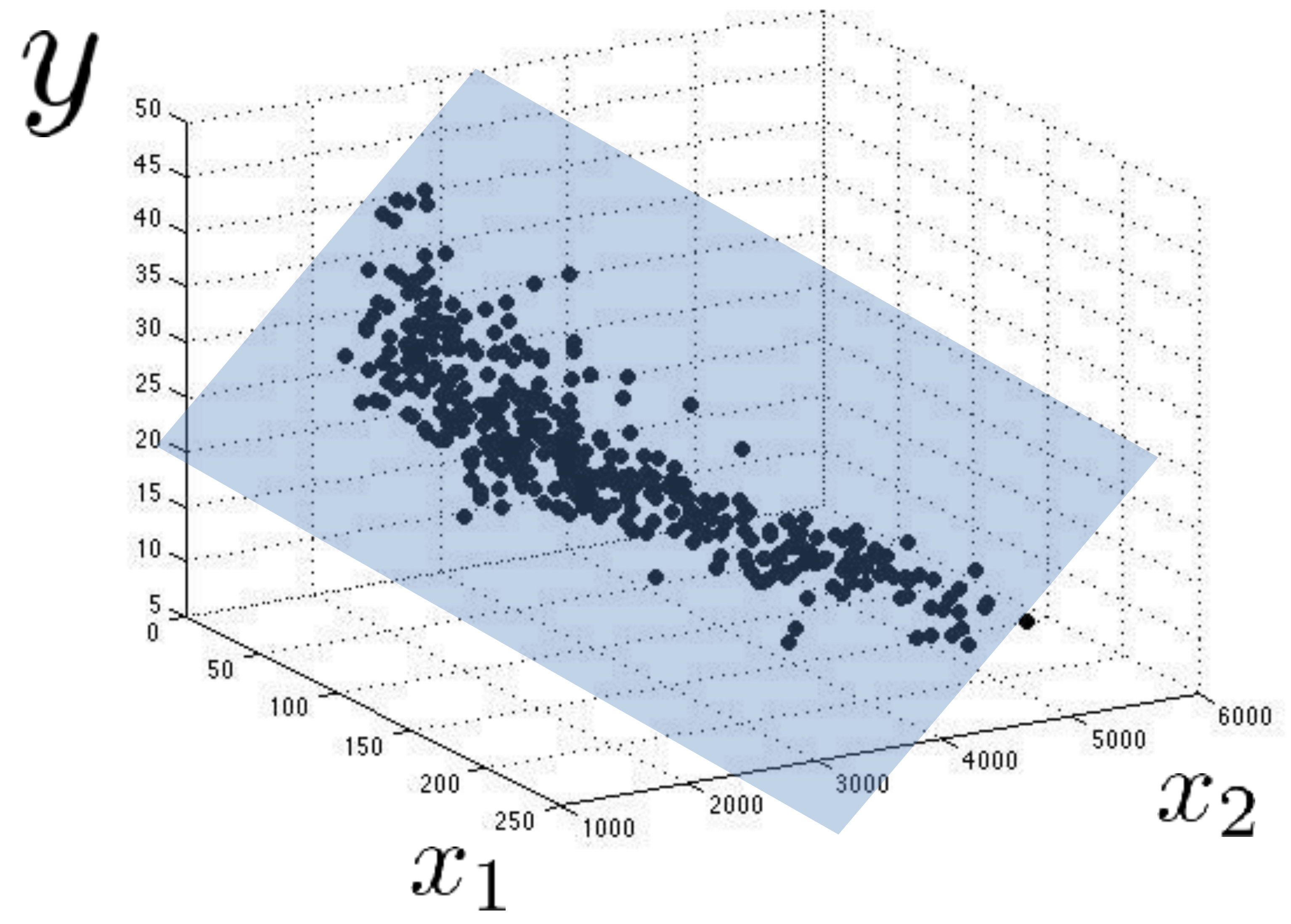
Univariate

$$\mathscr{L}(\mathbf{w}) = 0.21$$



Polynomial

$$\mathscr{L}(\mathbf{w}) = 0.063$$



Multivariate

$$\mathscr{L}(\mathbf{w}) = 0.052$$

Which dataset should we compare on to figure out the best model?

Training? Validation? Testing?

OLS II - The Wrath of Khan*

* old man movie reference

Jason G. Fleischer, Ph.D.
Asst. Teaching Professor
Department of Cognitive Science, UC San Diego

jfleischer@ucsd.edu

@jasongfleischer

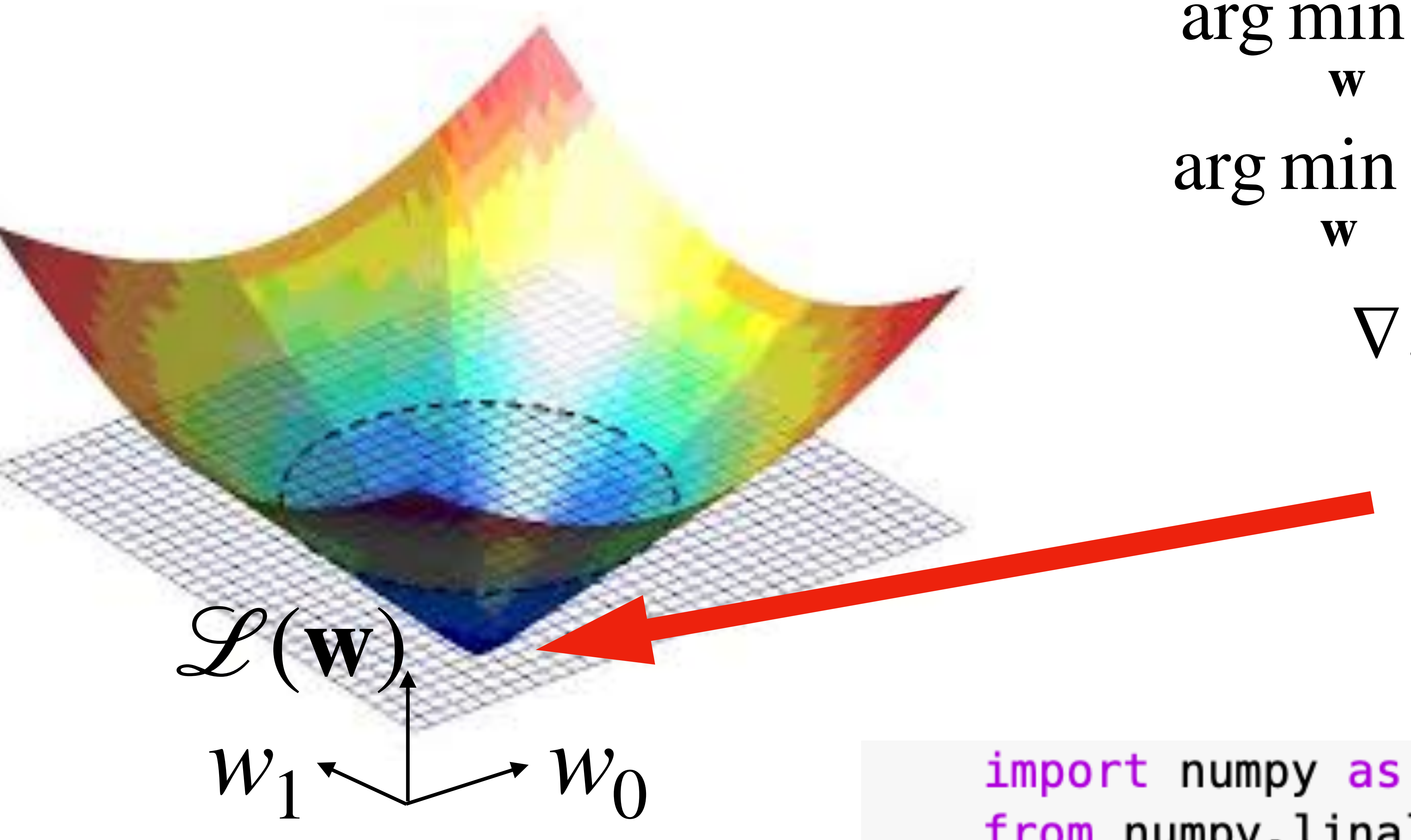
https://jgfleischer.com

Ordinary least squares regression

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$
$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$$
$$\mathscr{L}(\mathbf{w}) = \mathbf{e}^{\mathbf{T}}\mathbf{e}$$

$$\mathbf{X} = \begin{pmatrix} x_{0,0} & x_{0,1} & \dots & x_{0,d} \\ x_{1,0} & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ x_{n,0} & x_{n,1} & \dots & x_{n,d} \end{pmatrix} = \begin{pmatrix} 1 & x_{0,1} & \dots & x_{0,d} \\ 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{pmatrix}$$

Convex because loss is quadratic in $\mathbf{w}$



$$\arg\min_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\mathbf{w})$$
$$\arg\min_{\mathbf{w}} \mathscr{L}(\mathbf{w}) = \mathbf{w}^* \text{ such that } \nabla\mathscr{L}(\mathbf{w}^*) = 0$$

$$\nabla\mathscr{L}(\mathbf{w}) = -2\mathbf{X}^{\mathbf{T}}\mathbf{y} + 2\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{w}$$

$$\therefore \mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

$$\mathbf{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{pmatrix}$$

```
import numpy as np
from numpy.linalg import inv

psuedoinv = inv(X_train.T @ X_train ) @ X_train.T
w_star =  psuedoinv @ y_train
```

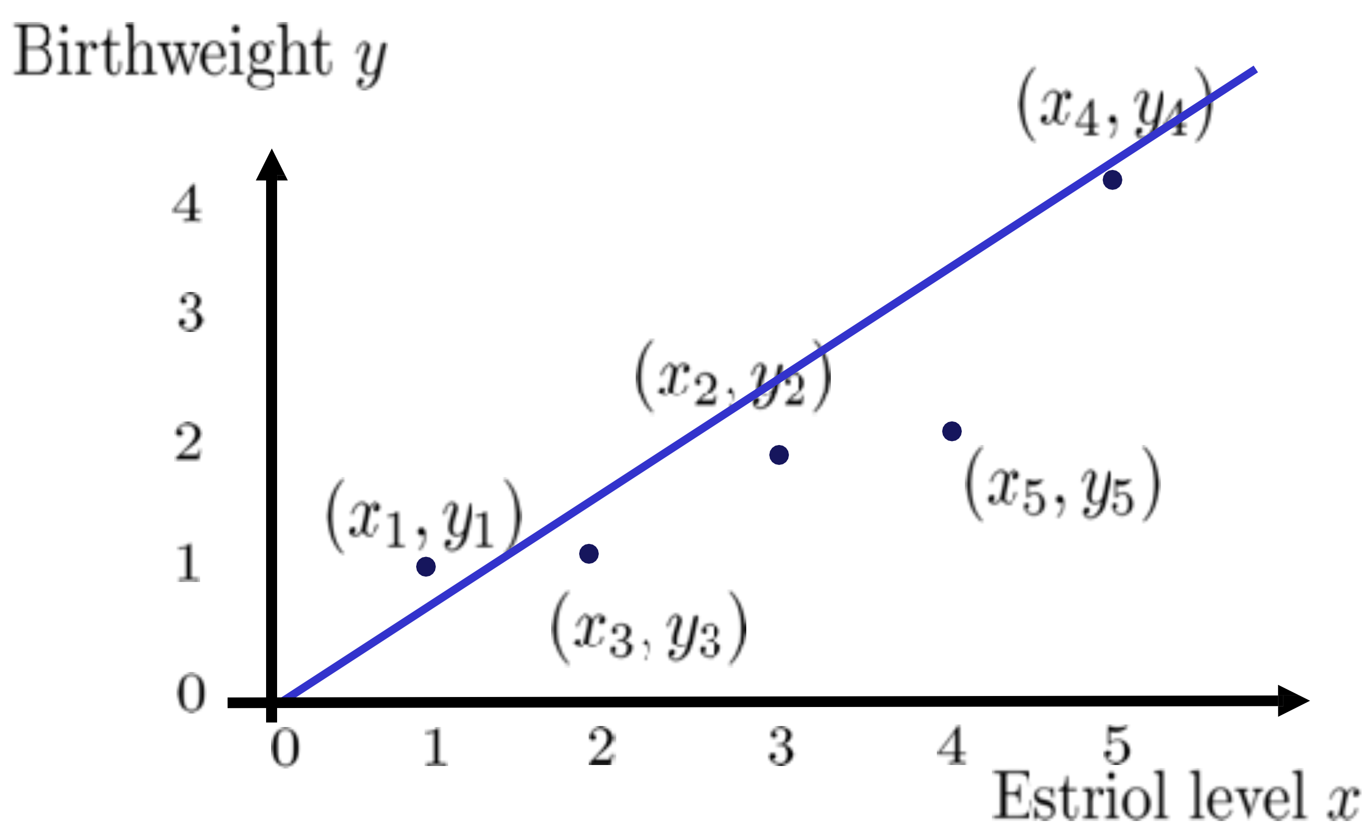
Matrix form for univariate OLS

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1),(3,1.9),(2,1.05),(5,4.1),(4,2.1)\}$$

Basic equation

$$y = w_0 + w_1 x$$

$$\begin{aligned} 1 &= w_0 + w_1 \times 1 \\ 1.9 &= w_0 + w_1 \times 3 \\ 1.05 &= w_0 + w_1 \times 2 \\ 4.1 &= w_0 + w_1 \times 5 \\ 2.1 &= w_0 + w_1 \times 4 \end{aligned}$$



Matrix form

$$\hat{\mathbf{y}} = \mathbf{X} \mathbf{w}$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1, 1 \\ 1, 3 \\ 1, 2 \\ 1, 5 \\ 1, 4 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \end{pmatrix}$$

$$\mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

```
import numpy as np
from numpy.linalg import inv

psuedoinv = inv(X_train.T @ X_train ) @ X_train.T
w_star =  psuedoinv @ y_train
```

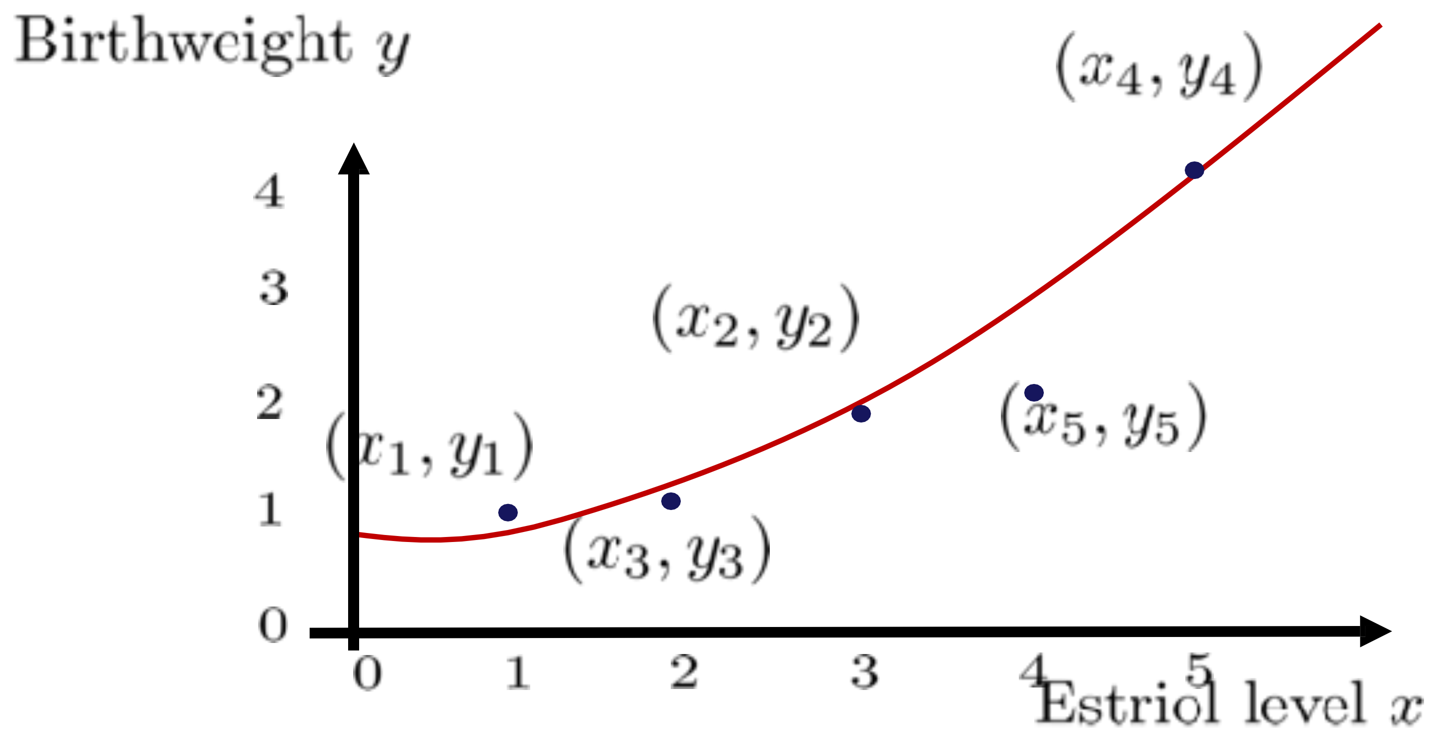
Matrix form for polynomial OLS

$$S_{training} = \{(x_i, y_i), i = 1..n\} = \{(1,1), (3,1.9), (2,1.05), (5,4.1), (4,2.1)\}$$

Basic equation

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

$$1 = w_0 + w_1 \times 1 + w_2 \times 1$$
$$1.9 = w_0 + w_1 \times 3 + w_2 \times 9$$
$$1.05 = w_0 + w_1 \times 2 + w_2 \times 4$$
$$4.1 = w_0 + w_1 \times 5 + w_2 \times 25$$
$$2.1 = w_0 + w_1 \times 4 + +w_2 \times 16$$



Matrix form

$$\hat{\mathbf{y}} = \mathbf{X} \quad \mathbf{w}$$

$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1,1,1 \\ 1,3,9 \\ 1,2,4 \\ 1,5,25 \\ 1,4,16 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$\mathbf{w}* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$

```
import numpy as np
from numpy.linalg import inv

psuedoinv = inv(X_train.T @ X_train ) @ X_train.T
w_star =  psuedoinv @ y_train
```

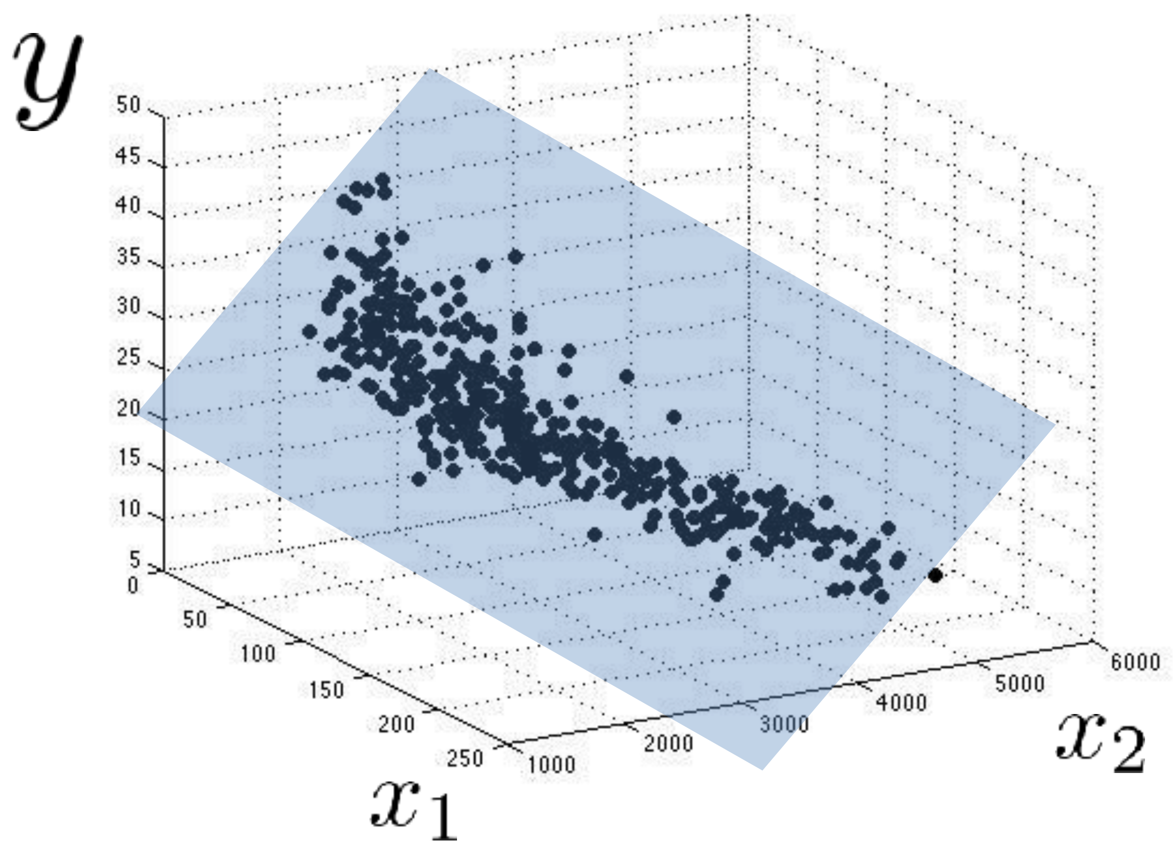
Matrix form for multivariate OLS

$$S_{training} = \{((x_{i1}, x_{i2}), y_i), i = 1..n\} = \{((1, 0.5), 1), ((3, 0.9), 1.9), ((2, 1.0), 1.05), ((5, 6.7), 4.1), ((4, 2.5), 2.1)\}$$

Basic equation

$$y = w_0 + w_1 x_1 + w_2 x_2$$

$$\begin{aligned} 1 &= w_0 + w_1 \times 1 + w_2 \times 0.5 \\ 1.9 &= w_0 + w_1 \times 3 + w_2 \times 0.9 \\ 1.05 &= w_0 + w_1 \times 2 + w_2 \times 1.0 \\ 4.1 &= w_0 + w_1 \times 5 + w_2 \times 6.7 \\ 2.1 &= w_0 + w_1 \times 4 + w_2 \times 2.5 \end{aligned}$$



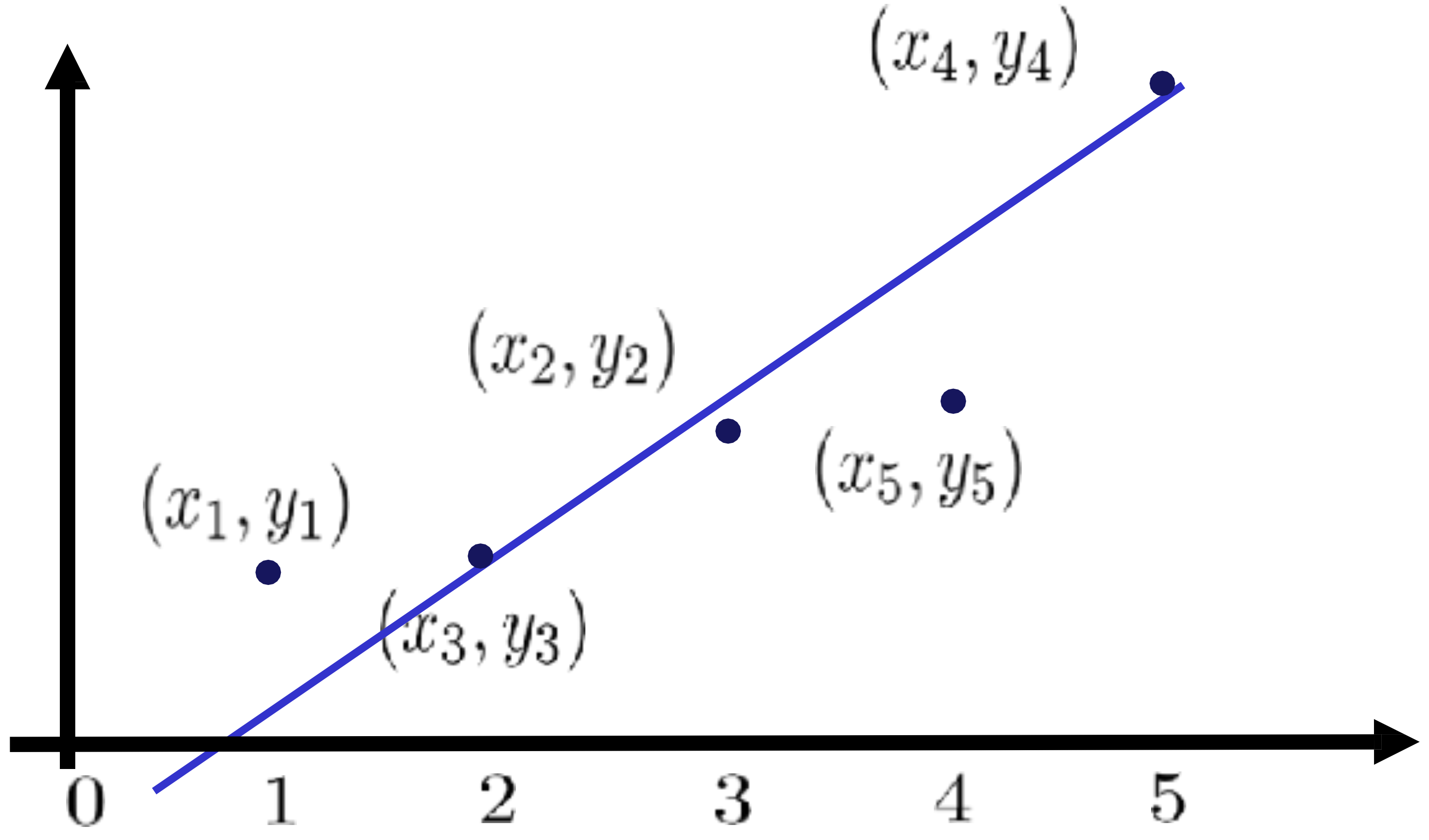
Matrix form

$$\hat{\mathbf{y}} = \mathbf{X} \quad \mathbf{w}$$

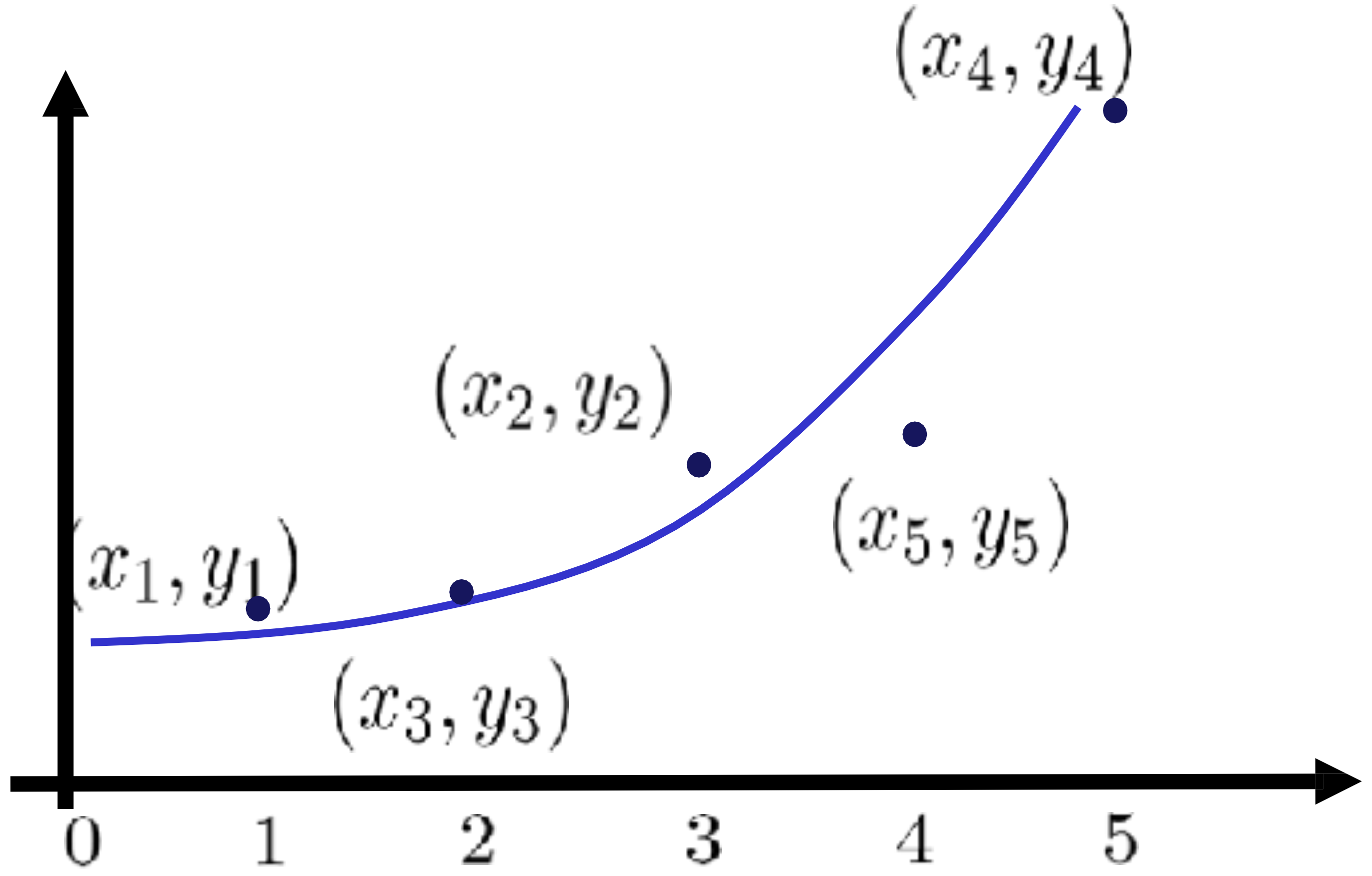
$$\begin{pmatrix} 1 \\ 1.9 \\ 1.05 \\ 4.1 \\ 2.1 \end{pmatrix} = \begin{pmatrix} 1, 1, 0.5 \\ 1, 3, 0.9 \\ 1, 2, 1.0 \\ 1, 5, 6.7 \\ 1, 4, 2.5 \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix}$$

$$\mathbf{w}* = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

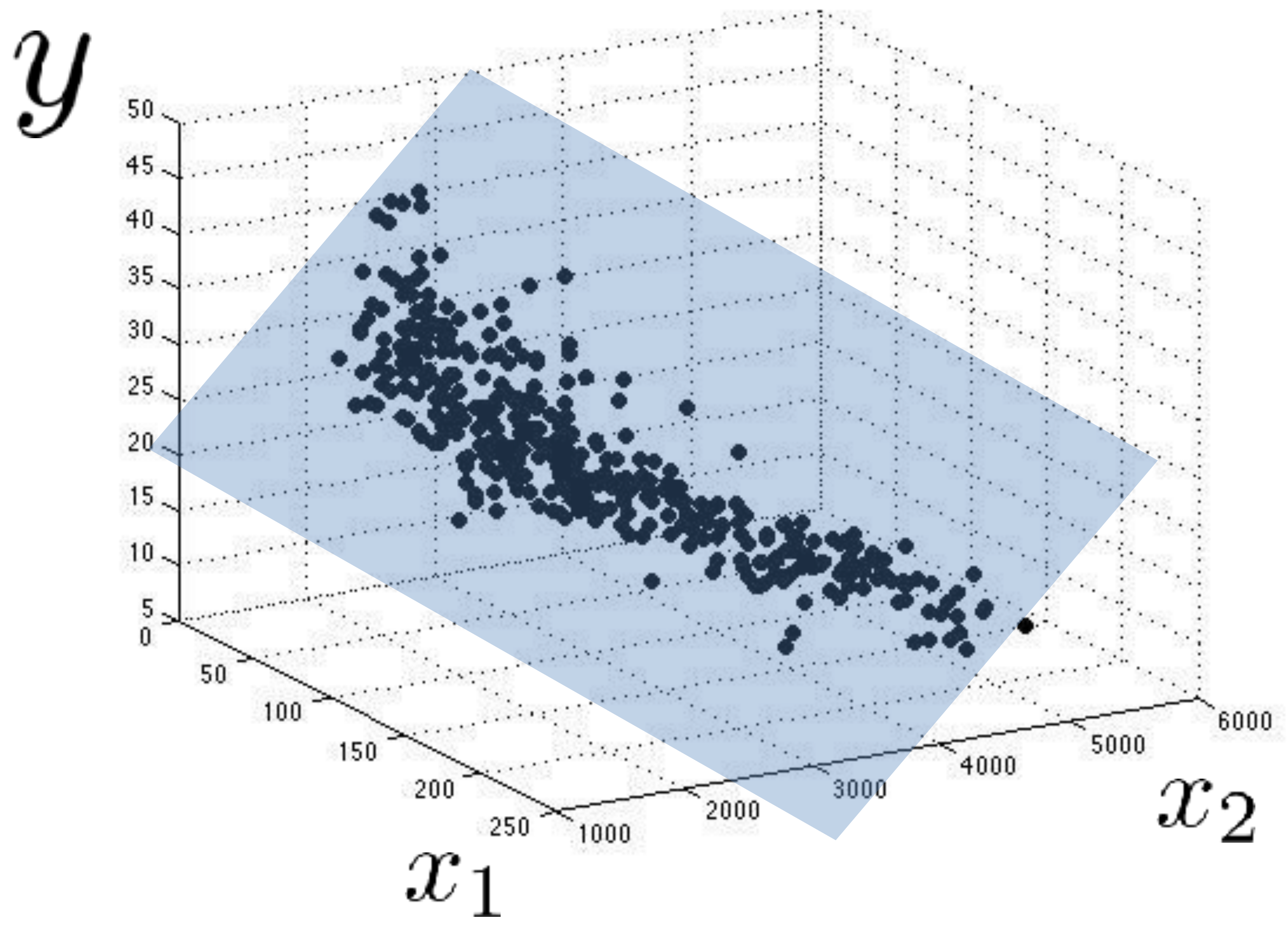
```
import numpy as np
from numpy.linalg import inv

psuedoinv = inv(X_train.T @ X_train ) @ X_train.T
w_star =  psuedoinv @ y_train
```



Univariate



Polynomial

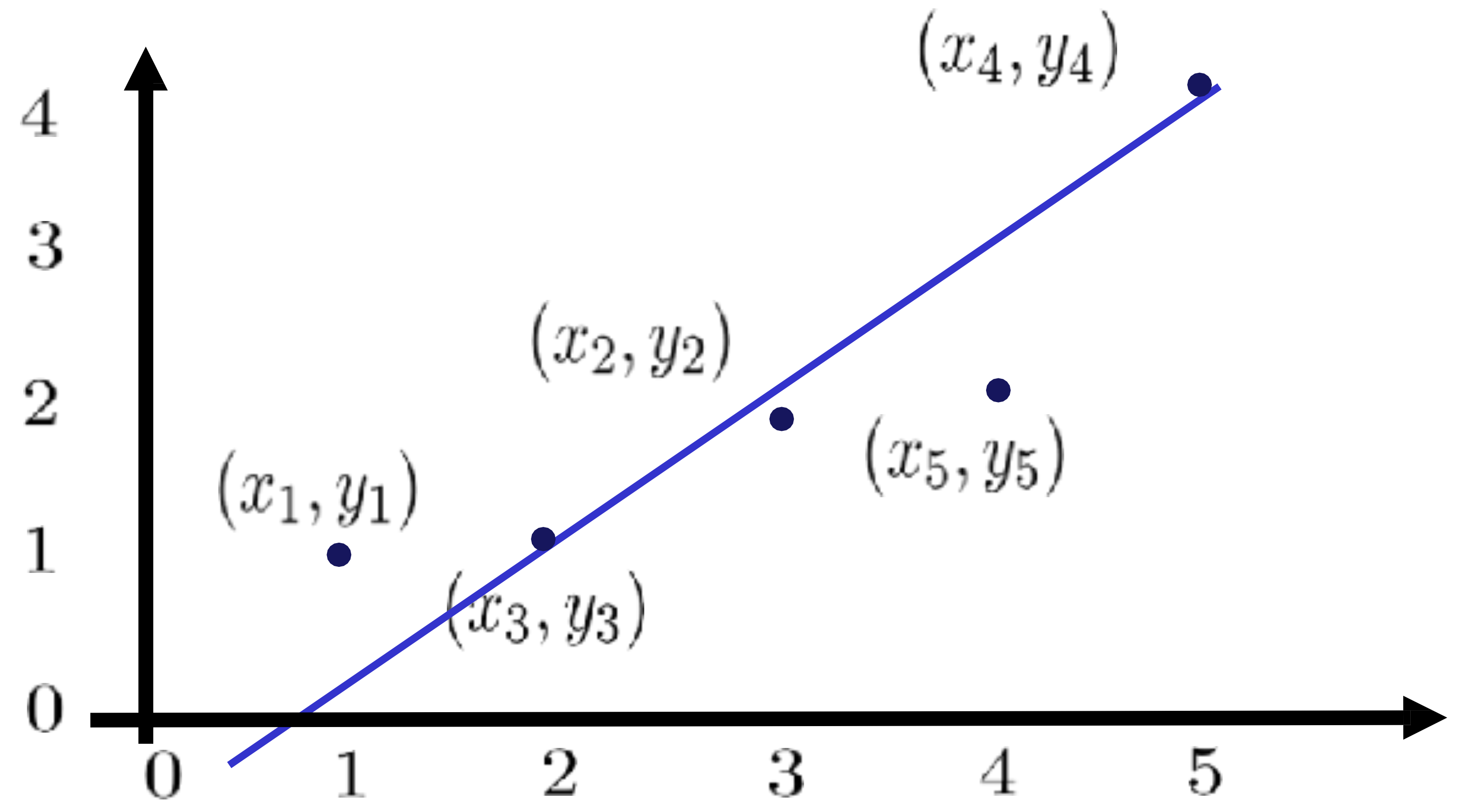


Multivariate

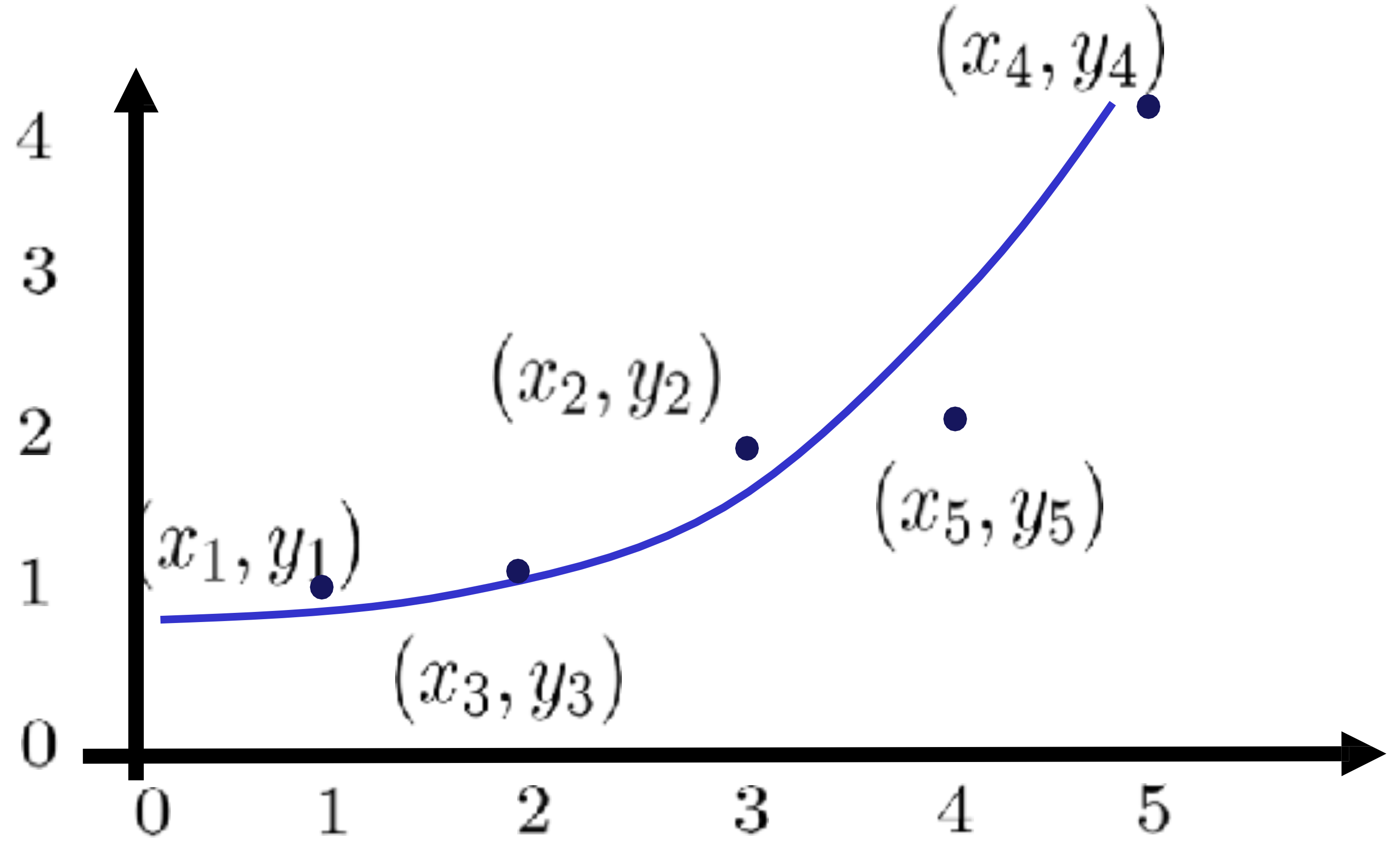
$$\begin{aligned} \hat{\mathbf{y}} &= \mathbf{X}\mathbf{w} \\ \mathbf{e} &= \mathbf{y} - \hat{\mathbf{y}} \\ \mathscr{L}(\mathbf{w}) &= \mathbf{e}^{\mathbf{T}}\mathbf{e} \end{aligned}$$

$$\mathbf{X} = \begin{pmatrix} 1 & x_{0,1} & \dots & x_{0,d} \\ 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \dots & x_{n,d} \end{pmatrix}$$

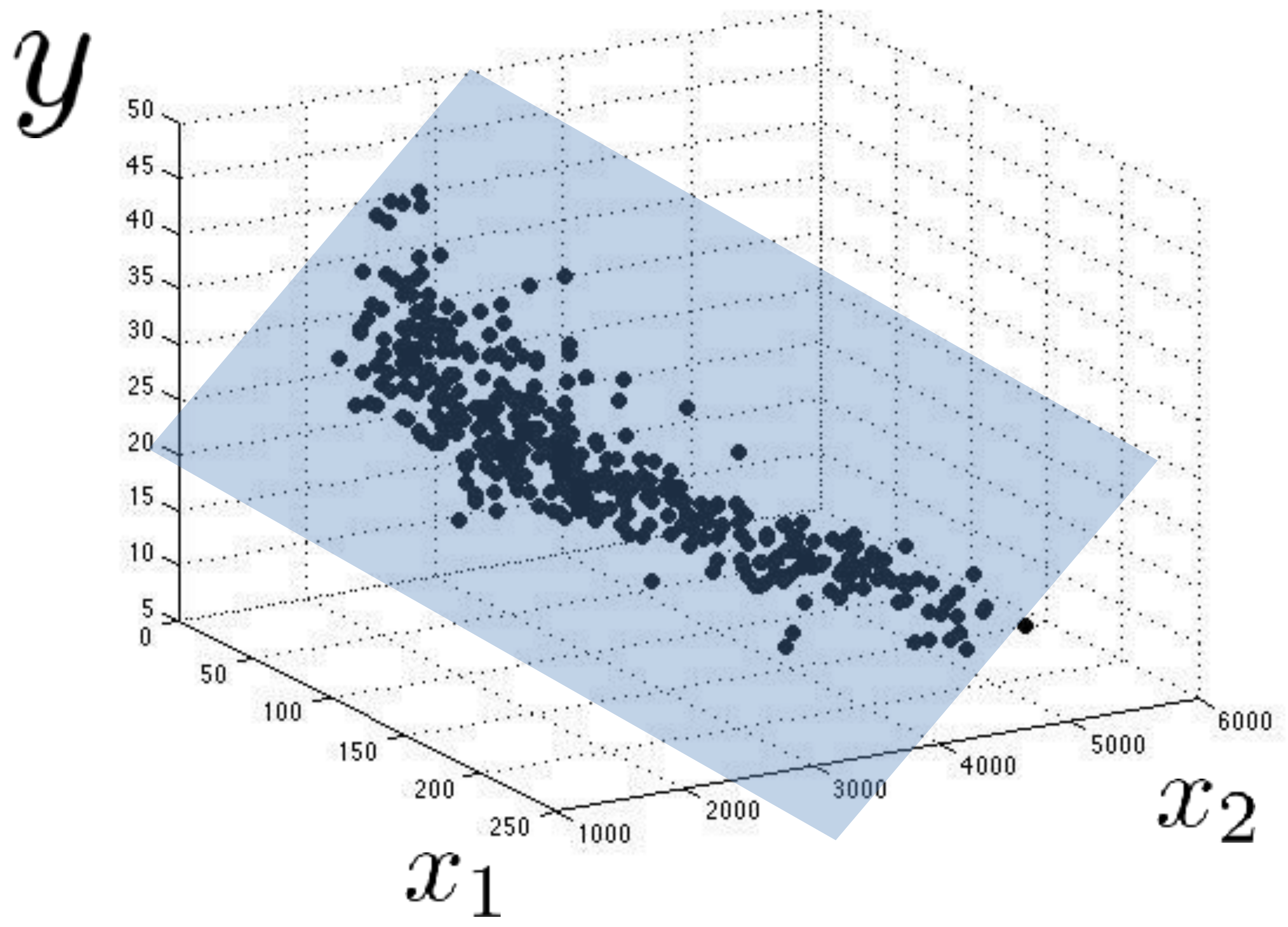
$$\mathbf{w}^* = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}$$



Univariate



Polynomial



Multivariate

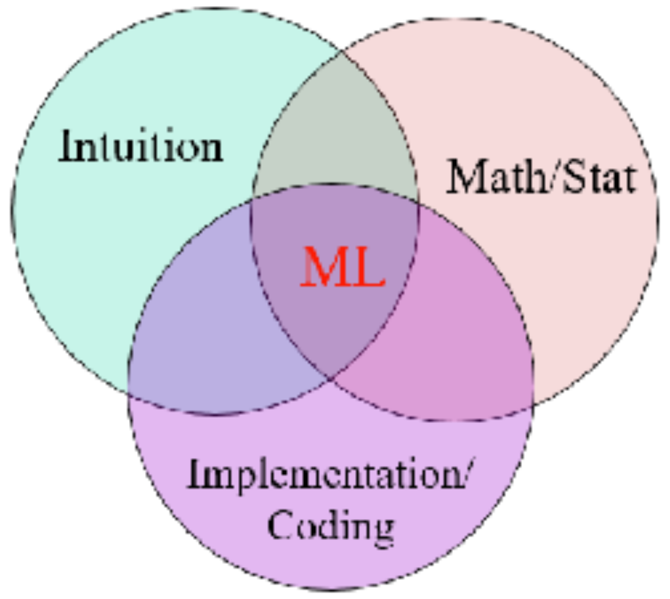
$\mathscr{L}_{\text{train}}(\mathbf{w}) = 0.21$

$\mathscr{L}_{\text{train}}(\mathbf{w}) = 0.063$

$\mathscr{L}_{\text{train}}(\mathbf{w}) = 0.052$

Which is the best model to choose?

Would your answer change if this was $\mathscr{L}_{\text{validate}}$?



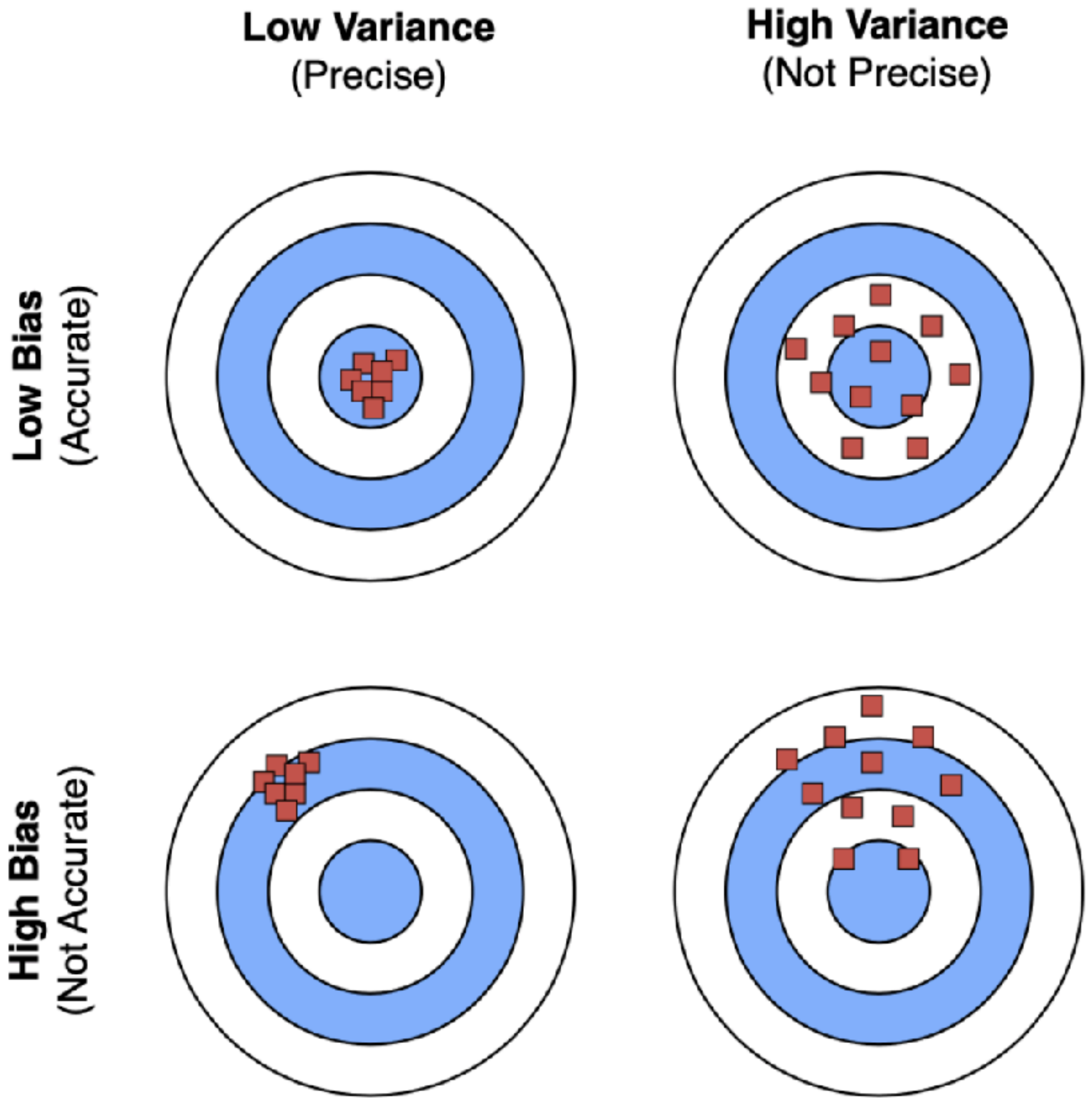
Implementation

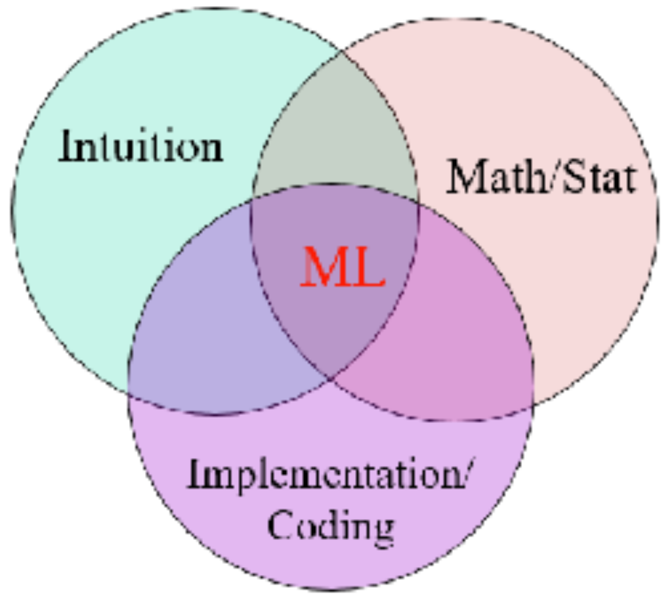
Linear Regression using Ordinary Least Squares

https://colab.research.google.com/github/COGS118A/demo_notebooks/blob/main/lecture_04_linear_regression.ipynb

https://github.com/COGS118A/demo_notebooks.git

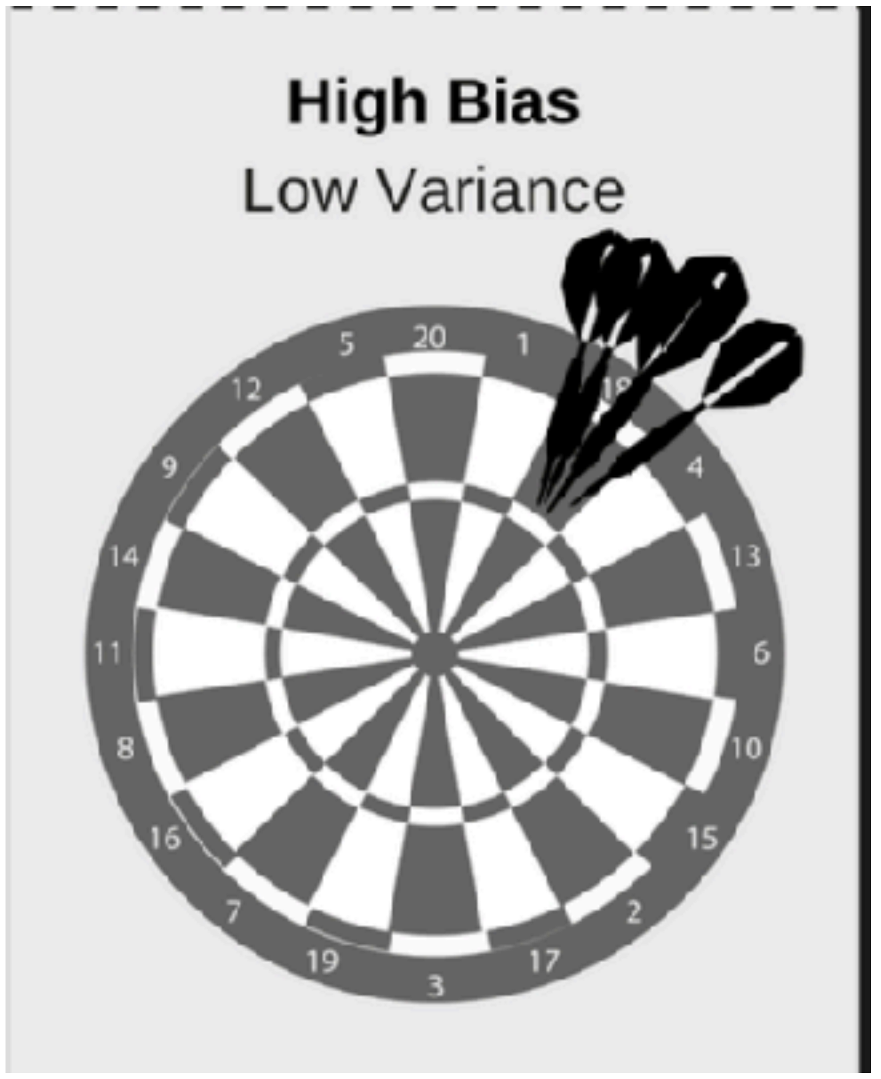
Bias Variance Tradeoff

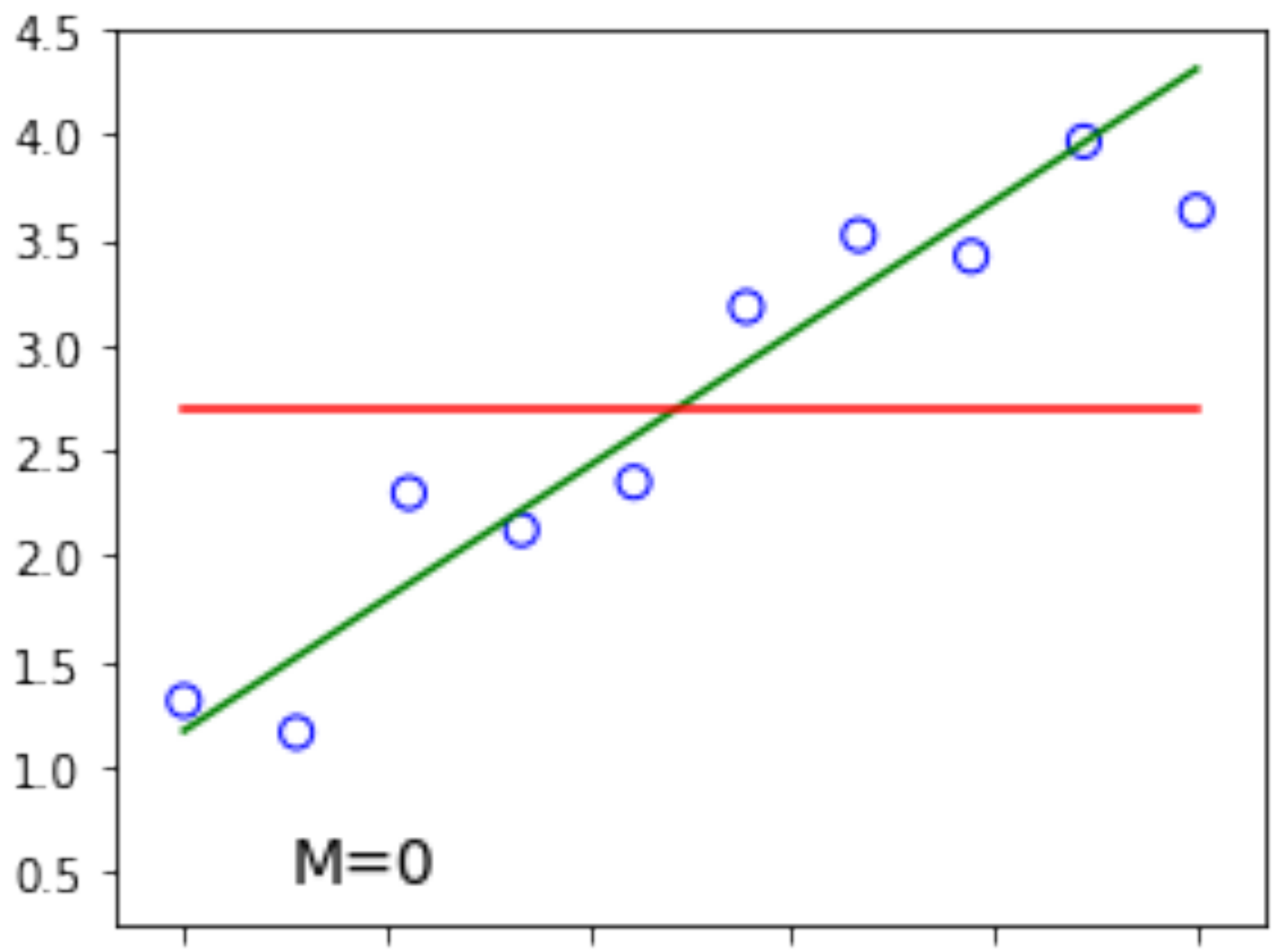




Intuition

Bias-variance tradeoff in model complexity

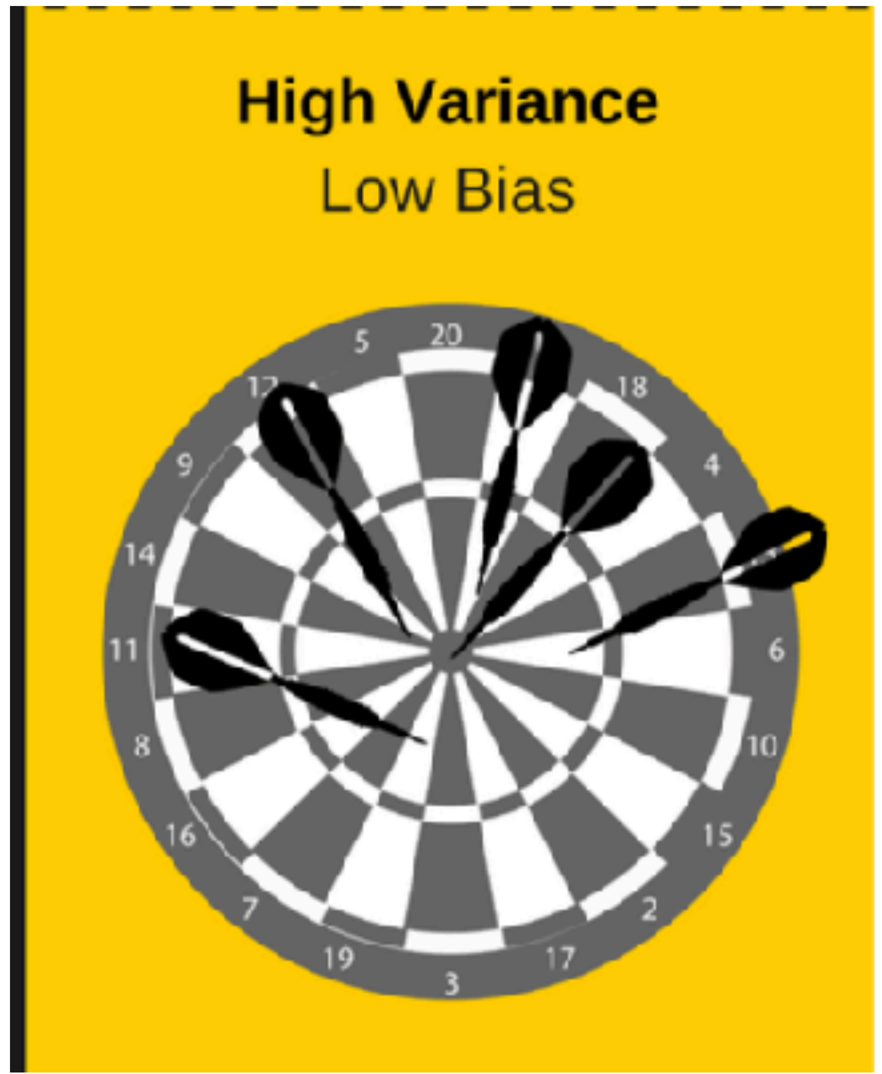


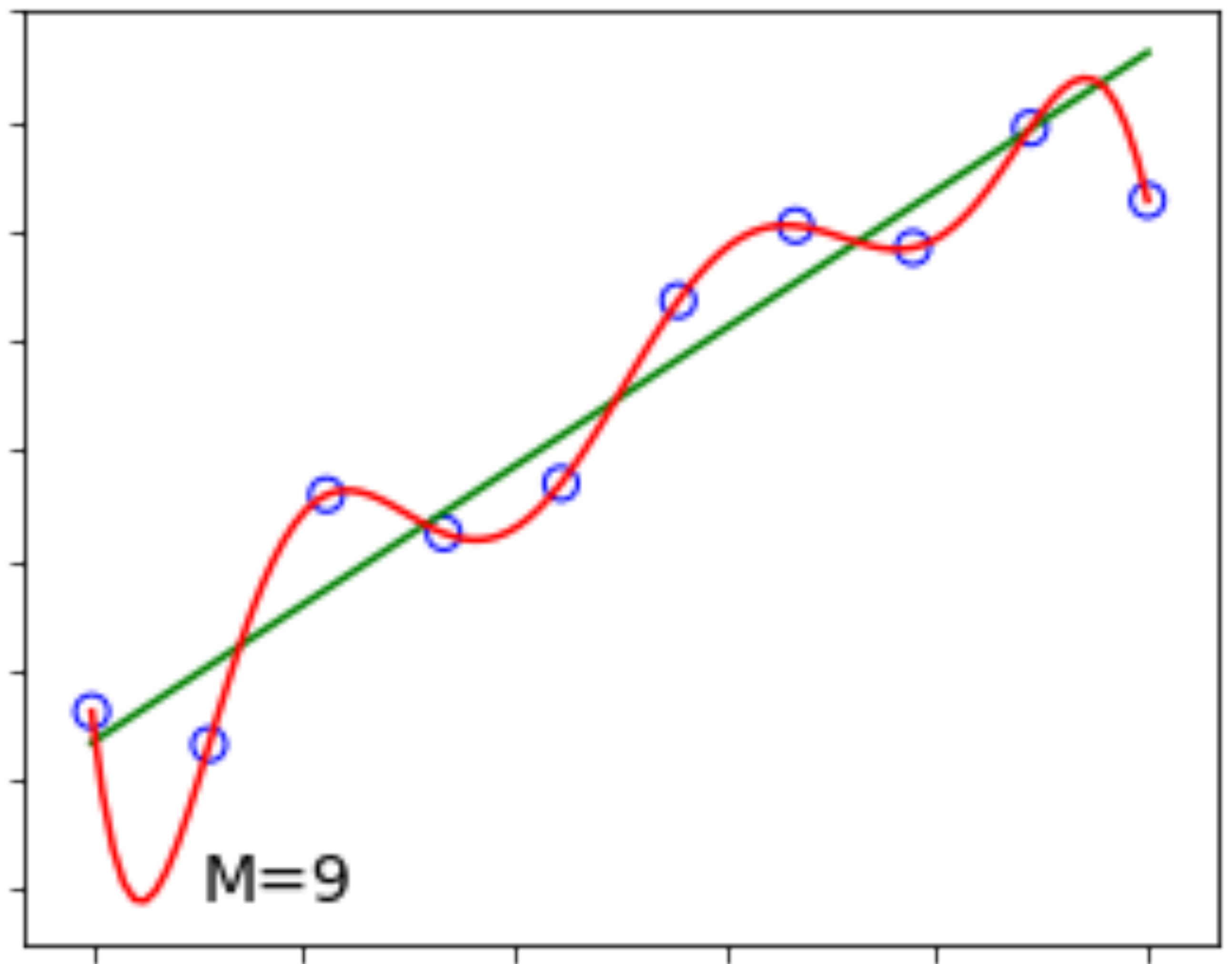


Underfitting

(Model is too simple!)

Biased to same answer no matter the random noise of sampling





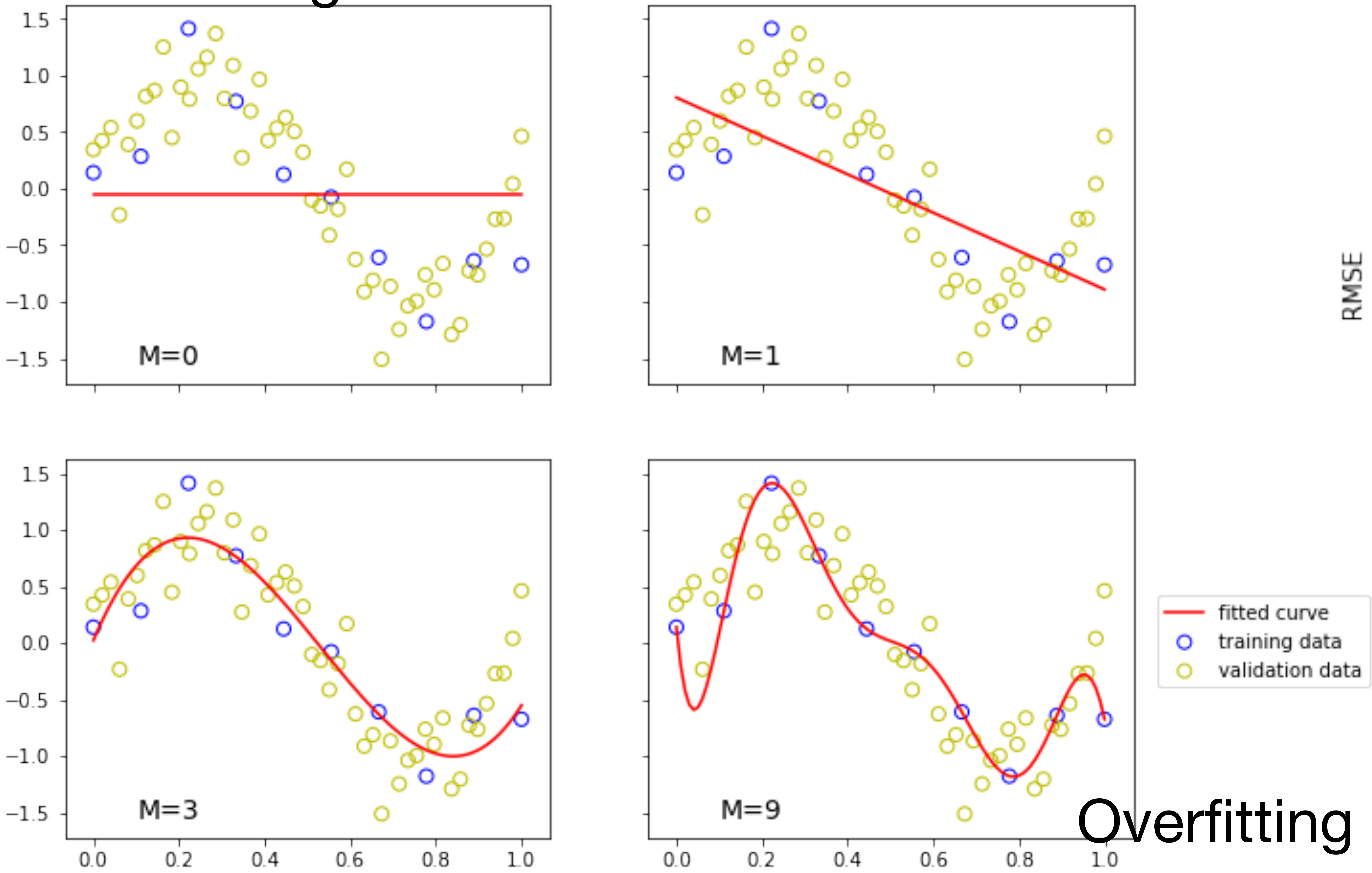
Overfitting

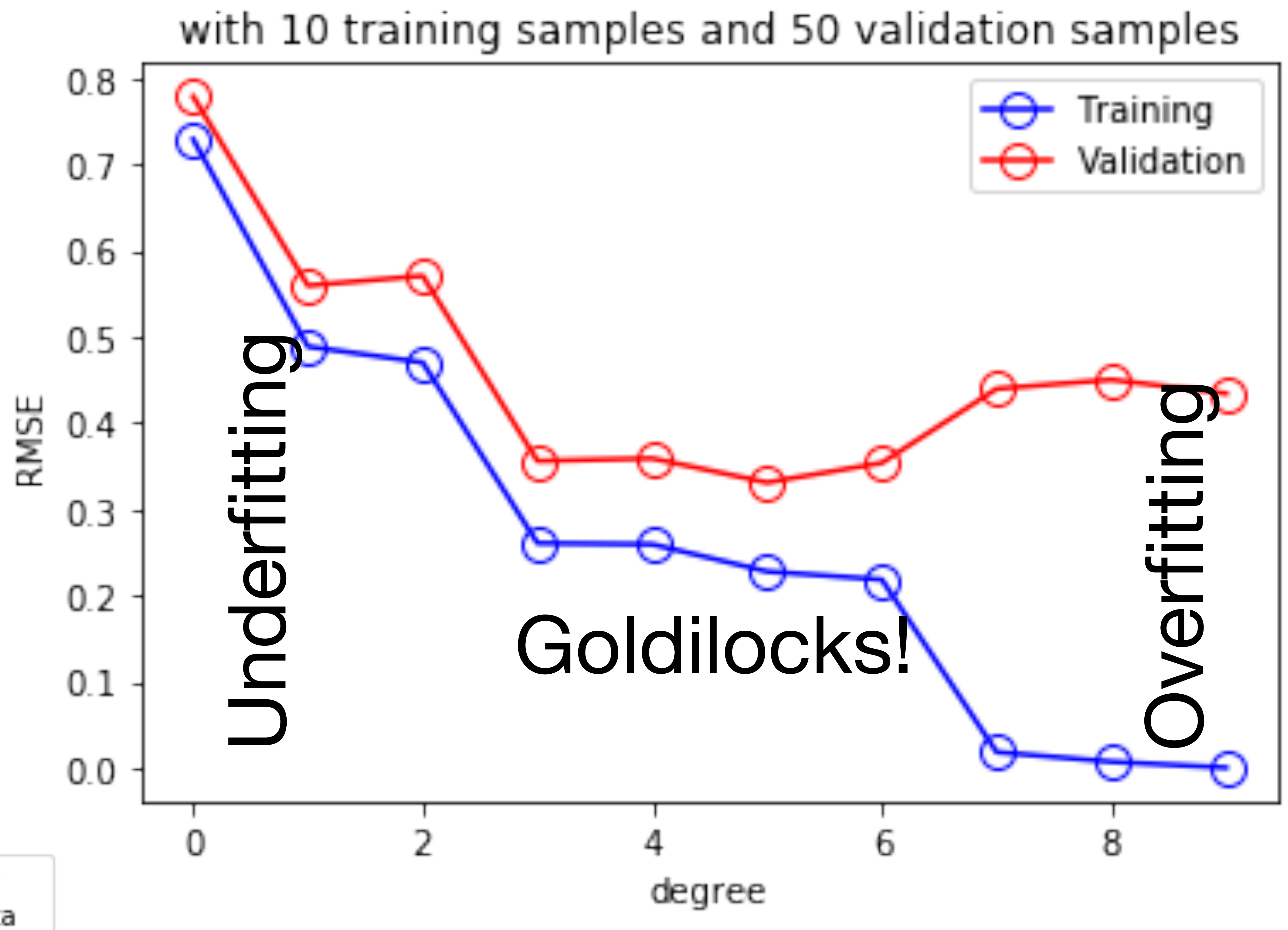
(Model is too complex!)

Highly variable answer depending on the random noise of sampling

Training set -> set the parameters
Validation set -> try different models, select best
Test set -> how good is your chosen model

Underfitting





A “validation curve”

https://scikit-learn.org/stable/auto_examples/model_selection/plot_validation_curve.html#plotting-validation-curves