Linear Regression and Logistic Regression

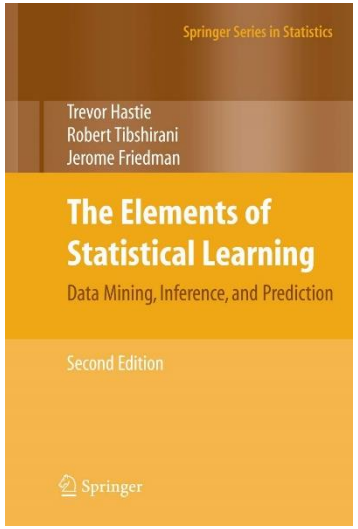
Assoc. Prof. Karl Ezra Pilario, Ph.D.
Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

Outline

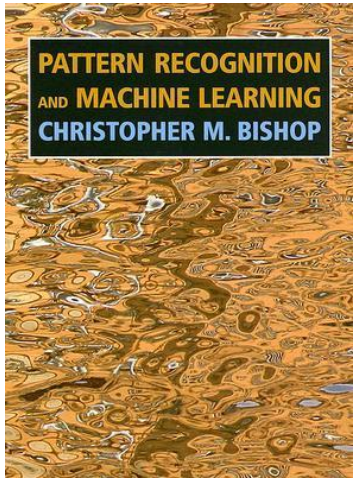
- Linear Regression
  - Solution to Linear Least Squares
  - Linear Basis Function Model
  - Performance Metrics
  - Ridge Regularization
  - Locally Weighted Linear Regression

- Logistic Regression
  - Binary Classification Problem
  - Log-loss Cost Function
  - Performance Metrics

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



Bishop (2006)
Pattern Recognition and Machine Learning. Springer.



Preliminaries

Notation:

$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}, \ldots$	Matrices
$\boldsymbol{A} \in \Re^{m \times n}$	Matrix of size m by n
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}, \ldots$	Column vectors
$\boldsymbol{a} \in \Re^{n}$	Column vector with n elements
$\boldsymbol{a}^T, \boldsymbol{b}^T, \boldsymbol{c}^T, \boldsymbol{d}^T, \ldots$	Row vectors
$\boldsymbol{a}^T \in \Re^{n}$	Row vector with n elements
$a, b, c, d, \ldots$	Scalars (1-by-1 matrices)
$\alpha, \beta, \gamma, \delta, \ldots$	Scalars (1-by-1 matrices)

Example:
House Price Data Set

House No.	Floor Area	Rooms	Age	Price
1	280	3	7	PhP 8.9M
2	140	2	5	PhP 2.3M
3	225	2	6	PhP 7.0M
4	300	4	4	PhP 9.2M
5	180	2	3	PhP 3.4M
6	195	2	5	PhP 5.6M
7	305	4	2	PhP 9.4M
8	150	3	8	PhP 3.8M

This is an example of a matrix:

$$\boldsymbol{X} = \begin{bmatrix} 280 & 3 & 7 \\ 140 & 2 & 5 \\ 225 & 2 & 6 \\ 300 & 4 & 4 \\ 180 & 2 & 3 \\ 195 & 2 & 5 \\ 305 & 4 & 2 \\ 150 & 3 & 8 \end{bmatrix} \in \Re^{8 \times 3}$$

This is an example of a column vector:

$$\boldsymbol{y} = \begin{bmatrix} 8.9 \\ 2.3 \\ 7.0 \\ 9.2 \\ 3.4 \\ 5.6 \\ 9.4 \\ 3.8 \end{bmatrix} \in \Re^{8}$$

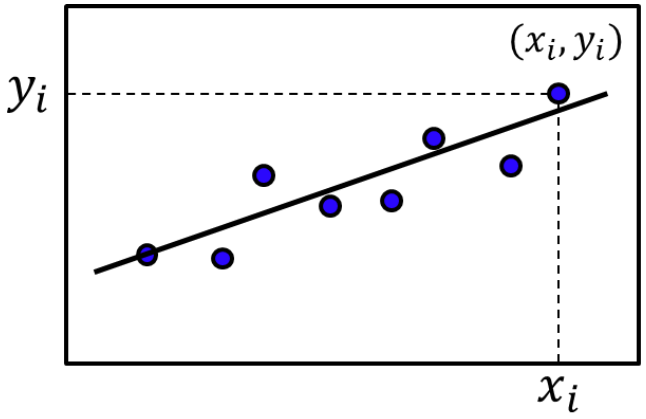
This is an example of a row vector:

$$\boldsymbol{y}^T = \begin{bmatrix} 8.9 & 2.3 & 7.0 & 9.2 & 3.4 & 5.6 & 9.4 & 3.8 \end{bmatrix} \in \Re^{1 \times 8}$$

Linear Least-Squares Regression

LINEAR REGRESSION

The simplest machine learning model is a line.



$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$N \times 1$ $\quad$ $N \times 2$ $\quad$ 2×1

Given: N data points, $(x_i, y_i) \quad i = 1, 2, \ldots, N$

Find: w_0 and w_1 in the equation of the *best-fit* line: $y = w_0 + w_1 x$

Cost Function:
Need to find $w_{0,1}$ such that

$$\min_{w} \quad f(w_0, w_1) = \sum_i [y_i - (w_0 + w_1 x_i)]^2$$

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

Partial derivative of f w.r.t. $\boldsymbol{w}$:

$$\frac{\partial f}{\partial \boldsymbol{w}} = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = 0$$

Recall: The optimum of $f(x)$ is at $\hat{x}$ where $f'(\hat{x}) = 0$.

Rearranging the equation:

$$\boldsymbol{X}^T\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = 0$$

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$$

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}}$$

Linear Least-Squares Regression

Example 1

Given: House Price Data Set

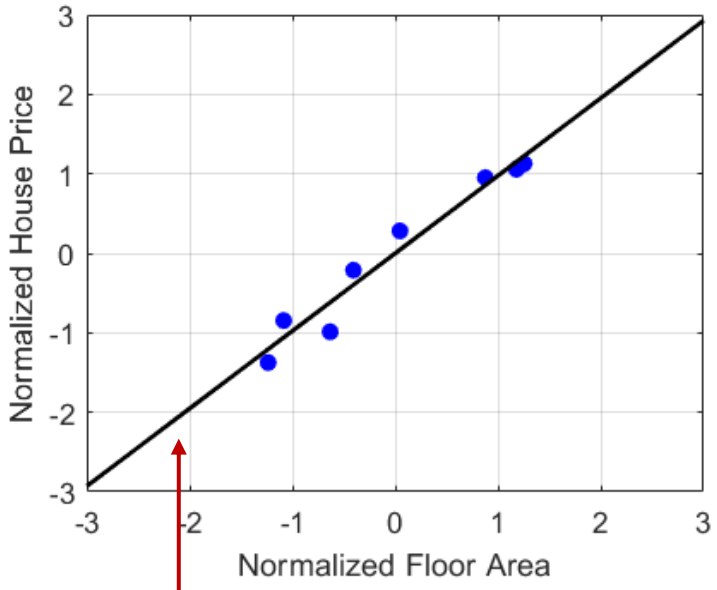
Find: w_0 and w_1 in the equation of the *best-fit* line:

$$y = w_0 + w_1 x$$

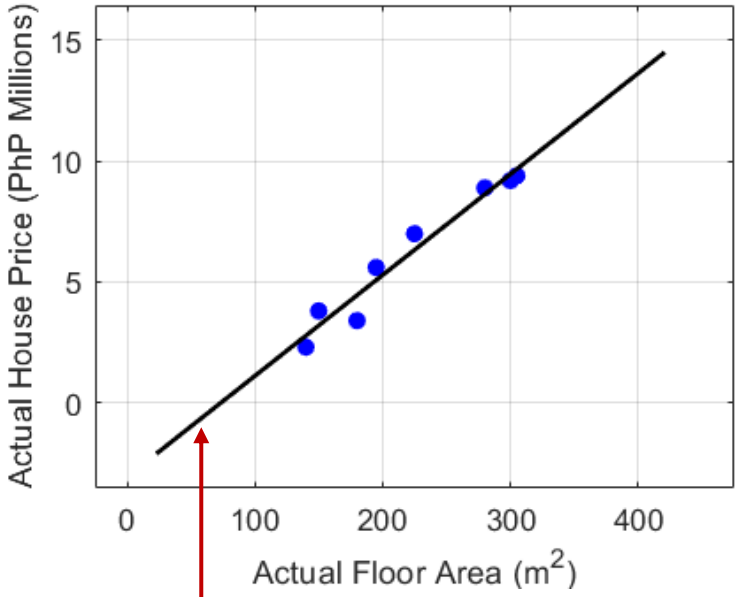
where x is the Floor Area and y is the house Price.

House No.	Floor Area	Rooms	Age	Price
1	280	3	7	PhP 8.9M
2	140	2	5	PhP 2.3M
3	225	2	6	PhP 7.0M
4	300	4	4	PhP 9.2M
5	180	2	3	PhP 3.4M
6	195	2	5	PhP 5.6M
7	305	4	2	PhP 9.4M
8	150	3	8	PhP 3.8M

Result in the **Normalized** Data space



Result in the **Original** Data space



Prediction in the Normalized space:

$$y' = 2 \times 10^{-16} + 0.9763x'$$

From the normalization earlier:

$$y' = \frac{y - 6.2}{2.84} \qquad x' = \frac{x - 221.9}{66.28}$$

$$y = -3.298 + 0.042x$$

where x is the Floor Area (m^2) and y is the house Price (millions PhP).

Linear Least-Squares Regression

Example 2

Given: House Price Data Set

Find: $\boldsymbol{w} = [w_0 \quad w_1 \quad w_2 \quad w_3]^T$ in the equation of the *best-fit* hyper-plane:

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

where x_1 is the Floor Area,
x_2 is the No. of Rooms,
x_3 is the house Age and
y is the house Price.

House No.	Floor Area	Rooms	Age	Price
1	280	3	7	PhP 8.9M
2	140	2	5	PhP 2.3M
3	225	2	6	PhP 7.0M
4	300	4	4	PhP 9.2M
5	180	2	3	PhP 3.4M
6	195	2	5	PhP 5.6M
7	305	4	2	PhP 9.4M
8	150	3	8	PhP 3.8M

Prediction in the **Normalized** space:

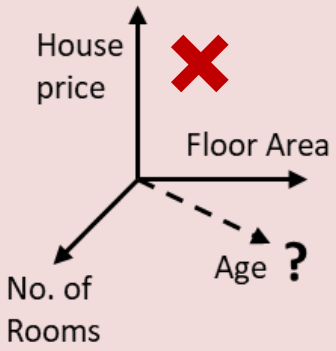
$$y' = 2 \times 10^{-16} + \underset{w_0}{} \quad \underset{w_1}{1.0734\, x_1'} \quad \underset{w_2}{-0.0404\, x_2'} \quad \underset{w_3}{+0.1821\, x_3'}$$

Since w_1 is the weight with the *largest absolute value*, the feature associated with it (x_1, Floor Area) is the *most important* predictor of house price among all x.

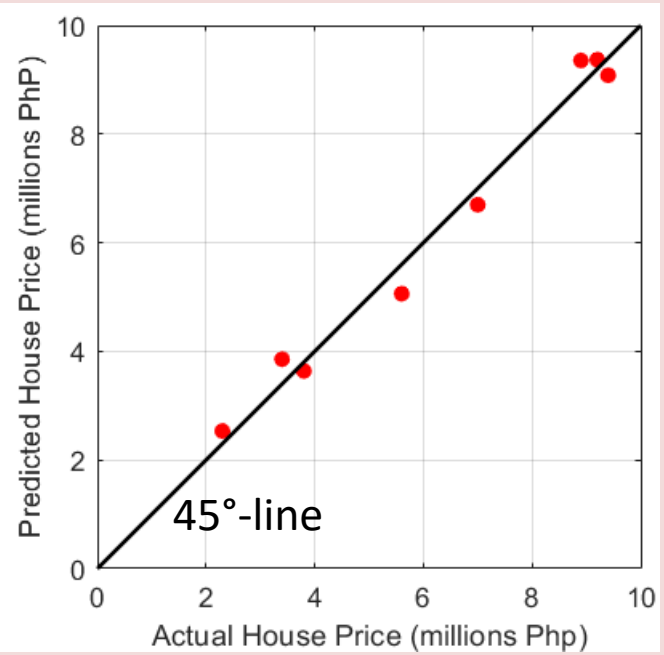
Prediction in the **Original** space:

$$y = -4.9311 + 0.0459\, x_1 - 0.1294\, x_2 + 0.2584\, x_3$$

We cannot plot the best-fit hyper-plane anymore because the visualization requires more than 3 dimensions.



But we can use a **scatter plot** of **Predicted** vs. **Actual** house price to visualize the model's performance.



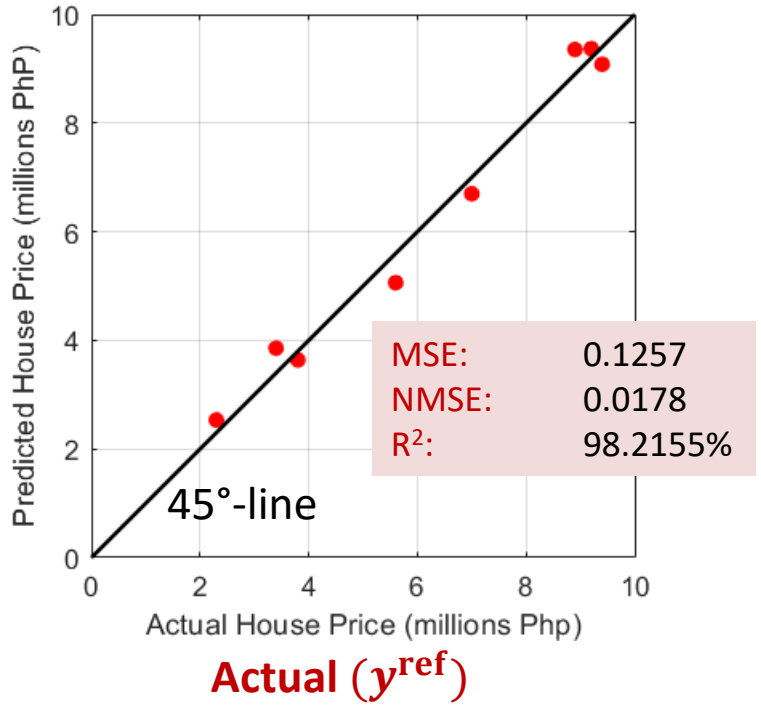
Linear Least-Squares Regression

Example 2

$$y = -4.9311 + 0.0459\,x_1 - 0.1294\,x_2 + 0.2584\,x_3$$

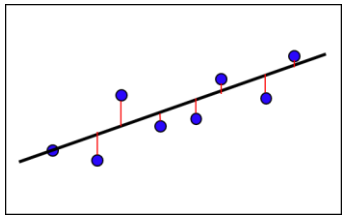
where x_1 is the Floor Area (m^2),
x_2 is the No. of Rooms,
x_3 is the house Age (years) and
y is the house Price (millions PhP).

Predicted (y^{pred})



Actual (y^{ref})

How to evaluate the goodness-of-fit?



1. Mean Squared Error (**MSE**)

$$MSE = \frac{1}{N}\sum_i \left(y_i^{\text{ref}} - y_i^{\text{pred}}\right)^2$$

MSE = 0 **(perfect fit)**
MSE = Inf **(worst fit)**

2. Normalized Mean Squared Error (**NMSE**)

$$NMSE = \frac{\sum_i \left(y_i^{\text{ref}} - y_i^{\text{pred}}\right)^2}{\sum_i \left(y_i^{\text{ref}} - \text{mean}(\boldsymbol{y}^{\text{ref}})\right)^2}$$

NMSE = 0 **(perfect fit)**
NMSE = Inf **(worst fit)**

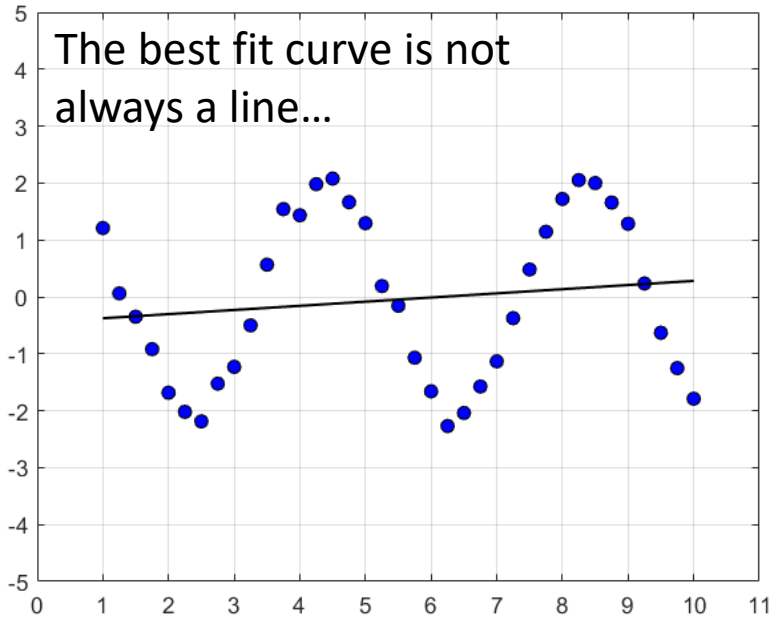
3. Coefficient of Determination ($\boldsymbol{R^2}$)

$$R^2 = 100\% \times (1 - NMSE)$$

R^2 = 100% **(perfect fit)**
R^2 = -Inf **(worst fit)**

Note: If $R^2 = 0$, then our model is no better than always reporting the *mean* of y^{ref} as the prediction y^{pred}.

Linear Least-Squares Regression



Linear Model: $y = w_0 + w_1 x$

$\phi_i(x)$ are called *basis functions*, which are user-defined.

Linear Basis Function Model: $y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$

$$y = \boldsymbol{w}^T\boldsymbol{\phi}(x)$$

where, $\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \in \Re^{m+1}$, $\boldsymbol{\phi}(x) = \begin{bmatrix} 1 \\ \phi_1(x) \\ \phi_2(x) \\ \vdots \\ \phi_m(x) \end{bmatrix} \in \Re^{m+1}$

Original Data Set → Transformed Data Set

x	y
⋮	⋮

$y = w_0 + w_1 x$

$\sin(x)$	$\cos(x)$	…	y
⋮	⋮		⋮

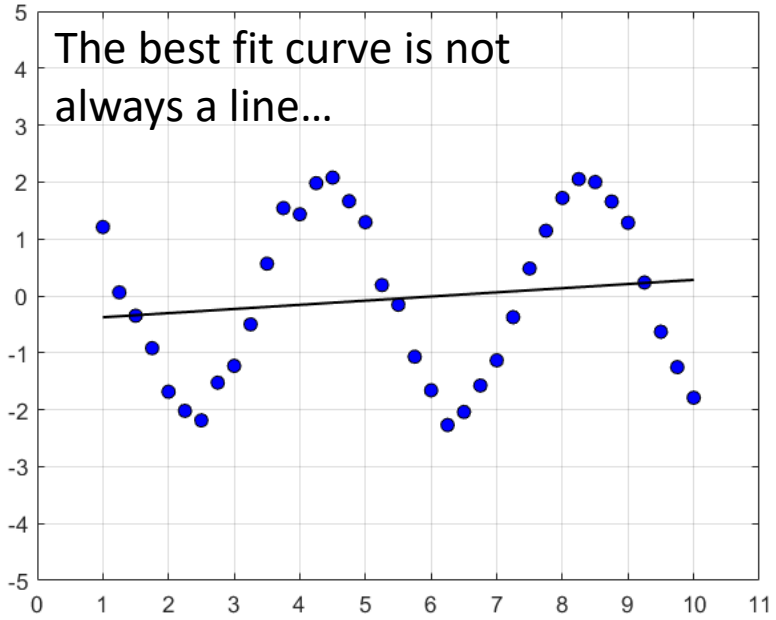
$y = w_0 + w_1 \sin(x) + w_2 \cos(x)$

Some examples:

For polynomial regression, $y = \overset{1}{w_0} + \overset{\phi_1(x)}{w_1 x} + \overset{\phi_2(x)}{w_2 x^2} + \cdots + \overset{\phi_m(x)}{w_m x^m}$

For any custom set of regressors, $y = \overset{1}{w_0} + \overset{\phi_1(x)}{w_1 \exp(x)} + \overset{\phi_2(x)}{w_2 \sin(x)} + \overset{\phi_m(x)}{w_3 \ln(x)} \ldots$

Linear Least-Squares Regression



Linear Basis Function Model:

$$y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$$

$$\boxed{y = \boldsymbol{w}^T\boldsymbol{\phi}(x)}$$

Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{w} \quad f(w_0, w_1, \dots) = \sum_i [y_i - (w_0 + w_1\phi(x_i) + \cdots)]^2$$

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w})$$

Original Data Set → Transformed Data Set

x	y
⋮	⋮

$y = w_0 + w_1 x$

$\sin(x)$	$\cos(x)$	…	y
⋮	⋮		⋮

$y = w_0 + w_1 \sin(x) + w_2 \cos(x)$

Notation:

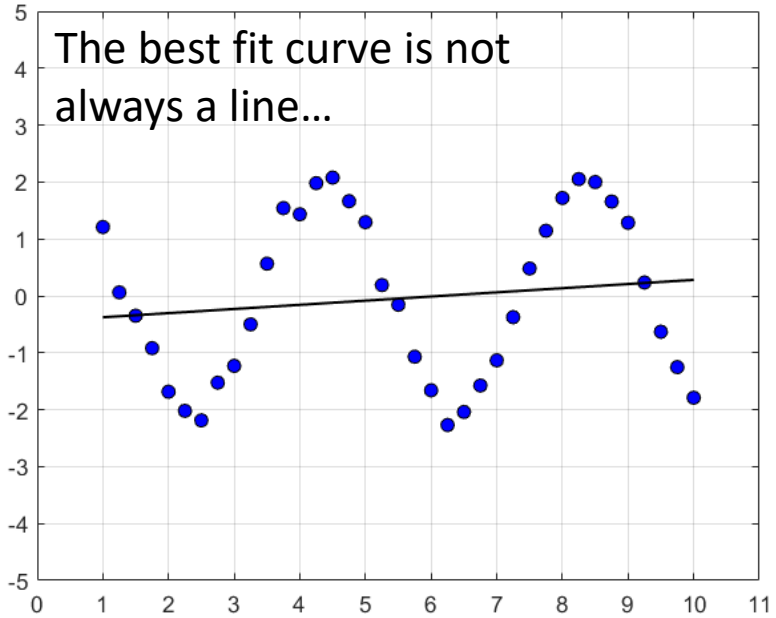
$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_N \end{bmatrix} \in \Re^N$$

Design matrix

$$\boldsymbol{\Phi}(\mathbf{x}) = \begin{bmatrix} \boldsymbol{\phi}(x_1)^T \\ \boldsymbol{\phi}(x_2)^T \\ \boldsymbol{\phi}(x_3)^T \\ \vdots \\ \boldsymbol{\phi}(x_N)^T \end{bmatrix} = \begin{bmatrix} 1 & \phi_1(x_1) & \dots & \phi_m(x_1) \\ 1 & \phi_1(x_2) & \dots & \phi_m(x_2) \\ 1 & \phi_1(x_3) & \dots & \phi_m(x_3) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \phi_1(x_N) & \dots & \phi_m(x_N) \end{bmatrix} \in \Re^{N\times(m+1)}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \in \Re^{m+1}$$

Linear Least-Squares Regression



Linear Basis Function Model: $y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$

$$\boxed{y = \boldsymbol{w}^T\boldsymbol{\phi}(x)}$$

Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w})$$

Partial derivative of f w.r.t. $\boldsymbol{w}$:

$$\frac{\partial f}{\partial \boldsymbol{w}} = -2\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi w}) = 0$$

$$\boldsymbol{\Phi}^T\boldsymbol{y} - \boldsymbol{\Phi}^T\boldsymbol{\Phi w} = 0$$

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi w} = \boldsymbol{\Phi}^T\boldsymbol{y}$$

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}}$$

Moore-Penrose pseudo-inverse of $\boldsymbol{\Phi}$

Original Data Set → Transformed Data Set

x	y
⋮	⋮

$y = w_0 + w_1x$

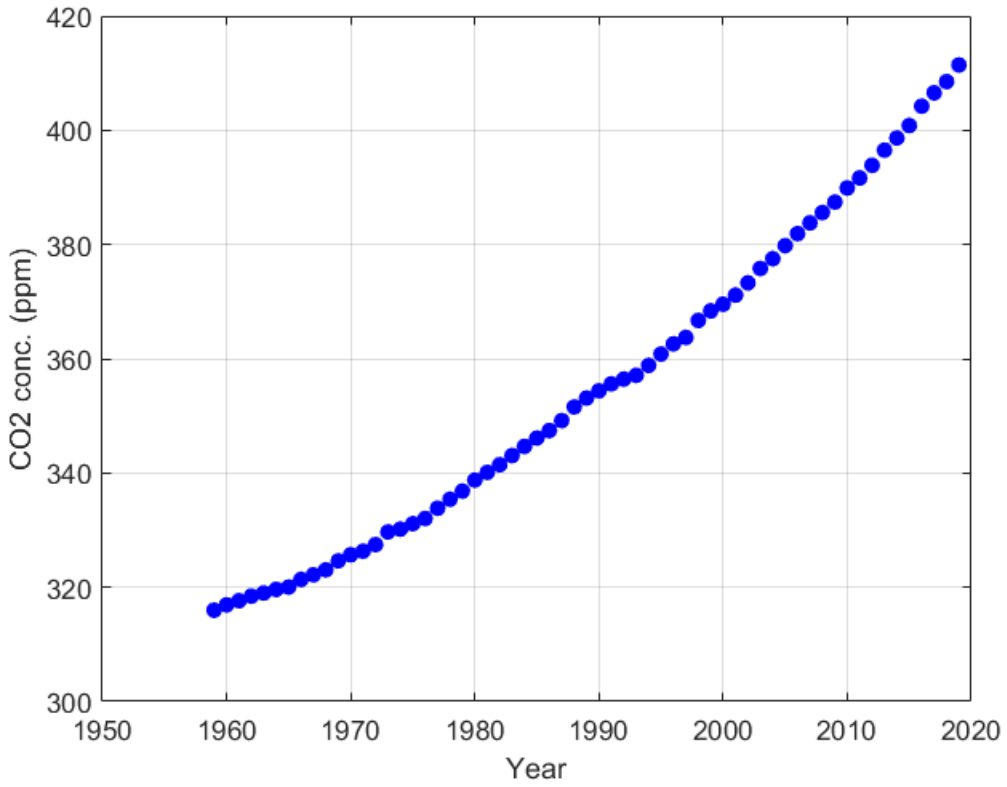
$\sin(x)$	$\cos(x)$	…	y
⋮	⋮		⋮

$y = w_0 + w_1\sin(x) + w_2\cos(x)$

Linear Least-Squares Regression

Example 3: Atmospheric CO_2 data set

Source: https://www.esrl.noaa.gov/gmd/ccgg/trends/data.html



The annual mean *atmospheric CO_2 concentration* data set (in ppm) from *1959 to 2019* measured in *Mauna Loa, Hawaii* is given in the following table.

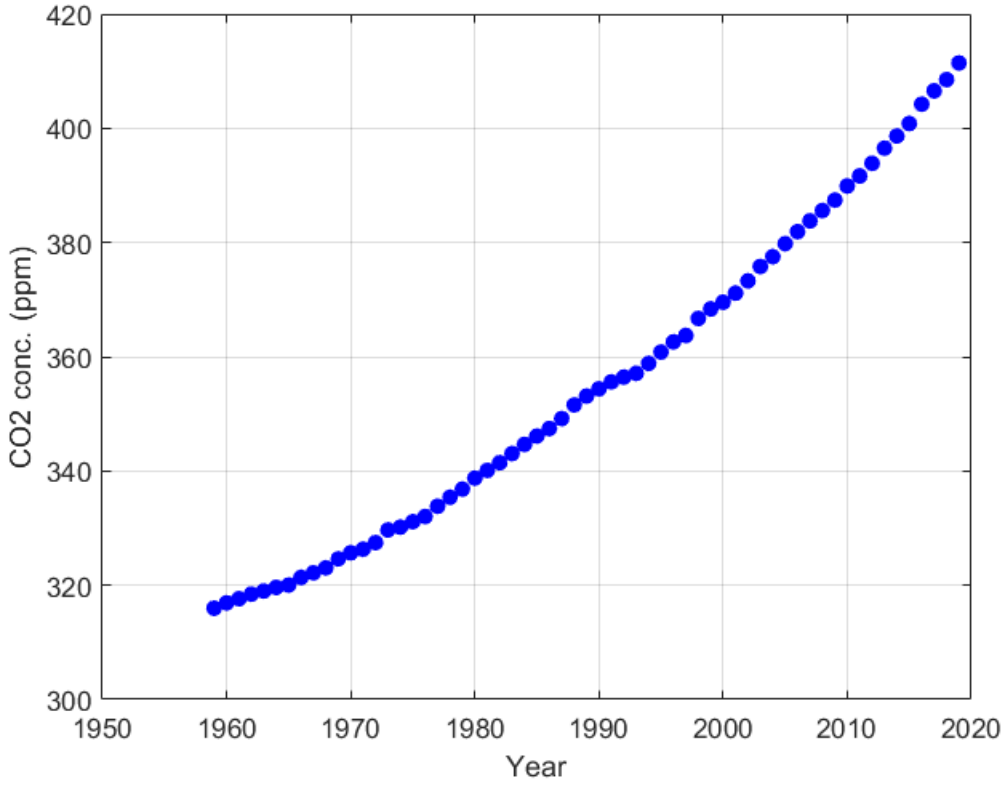
Predict the CO_2 conc. in 2030 using a cubic polynomial model.

Year	CO2	Year	CO2	Year	CO2	Year	CO2
1959	315.98	**1974**	330.18	**1989**	353.12	**2004**	377.52
1960	316.91	**1975**	331.12	**1990**	354.39	**2005**	379.80
1961	317.64	**1976**	332.04	**1991**	355.61	**2006**	381.90
1962	318.45	**1977**	333.83	**1992**	356.45	**2007**	383.79
1963	318.99	**1978**	335.40	**1993**	357.10	**2008**	385.59
1964	319.62	**1979**	336.84	**1994**	358.83	**2009**	387.43
1965	320.04	**1980**	338.75	**1995**	360.82	**2010**	389.90
1966	321.37	**1981**	340.11	**1996**	362.61	**2011**	391.65
1967	322.18	**1982**	341.45	**1997**	363.73	**2012**	393.86
1968	323.05	**1983**	343.05	**1998**	366.70	**2013**	396.52
1969	324.62	**1984**	344.66	**1999**	368.38	**2014**	398.64
1970	325.68	**1985**	346.12	**2000**	369.55	**2015**	400.83
1971	326.32	**1986**	347.43	**2001**	371.14	**2016**	404.22
1972	327.46	**1987**	349.18	**2002**	373.28	**2017**	406.55
1973	329.68	**1988**	351.57	**2003**	375.80	**2018**	408.52
						2019	411.43

Linear Least-Squares Regression

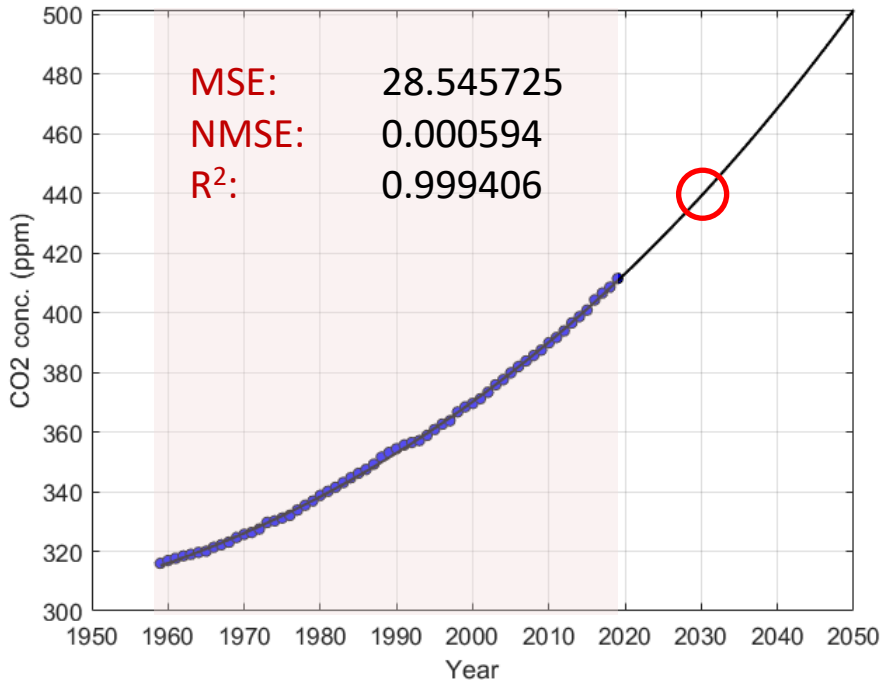
Example 3: Atmospheric CO_2 data set

Source: https://www.esrl.noaa.gov/gmd/ccgg/trends/data.html



Answer: Using a cubic polynomial, the CO_2 conc. in 2030 is projected to be 440 ppm.

$$y = -203{,}688.6 + 330.03x - 0.178\,x^2 + 3.2 \times 10^{-5}\,x^3$$



Ridge Regularization

Linear Basis Function Model:

$$y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$$

$$\boxed{y = \boldsymbol{w}^T\boldsymbol{\phi}(x)}$$

- λ is called the regularization parameter.
- It *penalizes* weights when their magnitudes become too large.
- The higher the λ, the bigger the penalty.
- "Ridge" means that the penalty is an **L²-norm**, or $\boldsymbol{w}^T\boldsymbol{w}$, or $\|\boldsymbol{w}\|^2$.
- Other regularizers include: LASSO, elastic net, etc.

Cost function *without* Ridge Regularization

Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w})$$

Partial derivative of f w.r.t. $\boldsymbol{w}$:

$$\frac{\partial f}{\partial \boldsymbol{w}} = -2\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi w}) = 0$$

$$\boldsymbol{\Phi}^T\boldsymbol{y} - \boldsymbol{\Phi}^T\boldsymbol{\Phi w} = 0$$

$$\boldsymbol{\Phi}^T\boldsymbol{\Phi w} = \boldsymbol{\Phi}^T\boldsymbol{y}$$

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}}$$

Cost function *with* Ridge Regularization

Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w}) + \lambda\|\boldsymbol{w}\|^2$$

Partial derivative of f w.r.t. $\boldsymbol{w}$:

$$\frac{\partial f}{\partial \boldsymbol{w}} = -2\boldsymbol{\Phi}^T(\boldsymbol{y} - \boldsymbol{\Phi w}) + 2\lambda\boldsymbol{I}\boldsymbol{w} = 0$$

$$\boldsymbol{\Phi}^T\boldsymbol{y} - \boldsymbol{\Phi}^T\boldsymbol{\Phi w} + \lambda\boldsymbol{I}\boldsymbol{w} = 0$$

$$(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})\boldsymbol{w} = \boldsymbol{\Phi}^T\boldsymbol{y}$$

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}}$$

Ridge Regularization

Linear Basis Function Model:

$$y = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \cdots + w_m\phi_m(x)$$

$$\boxed{y = \boldsymbol{w}^T\boldsymbol{\phi}(x)}$$

Cost function *with* Ridge Regularization

Cost function:

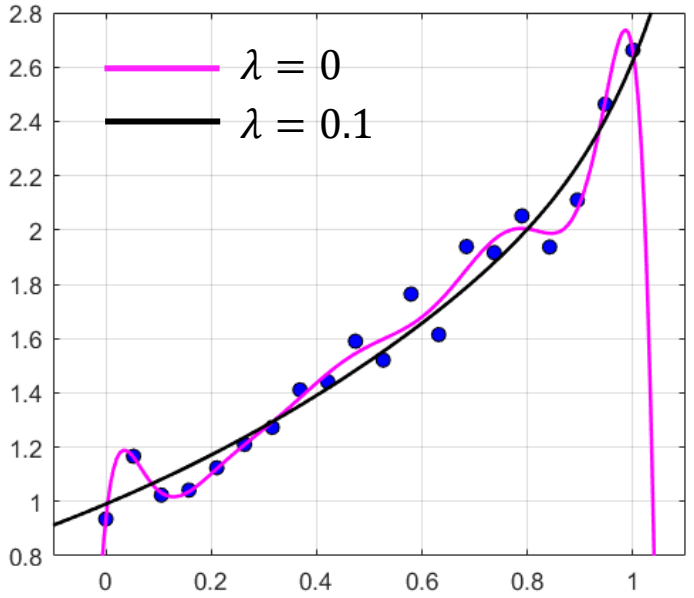
Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}}$$

Example 4: Growth Data

Find a *10-degree polynomial* that best fits the following growth data (●):



- Without regularization ($\lambda = 0$), the best-fit curve is incorrect because it does not show growth.
- However, the curve with $\lambda = 0$ will show a "more accurate fit" based on its R^2!

w/o regularization

```
y = 0.936
  + 18.359 x^1
  + -456.485 x^2
  + 4821.277 x^3
  + -27667.619 x^4
  + 96311.191 x^5
  + -211987.750 x^6
  + 296084.370 x^7
  + -253904.301 x^8
  + 121723.073 x^9
  + -24940.390 x^10
R^2: 0.98830
```

with regularization

```
y = 0.991
  + 0.819 x^1
  + 0.399 x^2
  + 0.137 x^3
  + 0.013 x^4
  + -0.026 x^5
  + -0.019 x^6
  + 0.012 x^7
  + 0.054 x^8
  + 0.099 x^9
  + 0.145 x^10
R^2: 0.96571
```

Overfitting vs. Underfitting

a.k.a. the Bias-Variance Trade-off

The **expected error** (or mean squared error, *MSE*) of any ML prediction can be decomposed as:

$$MSE = \text{Bias}^2 + \text{Variance} + \sigma^2$$

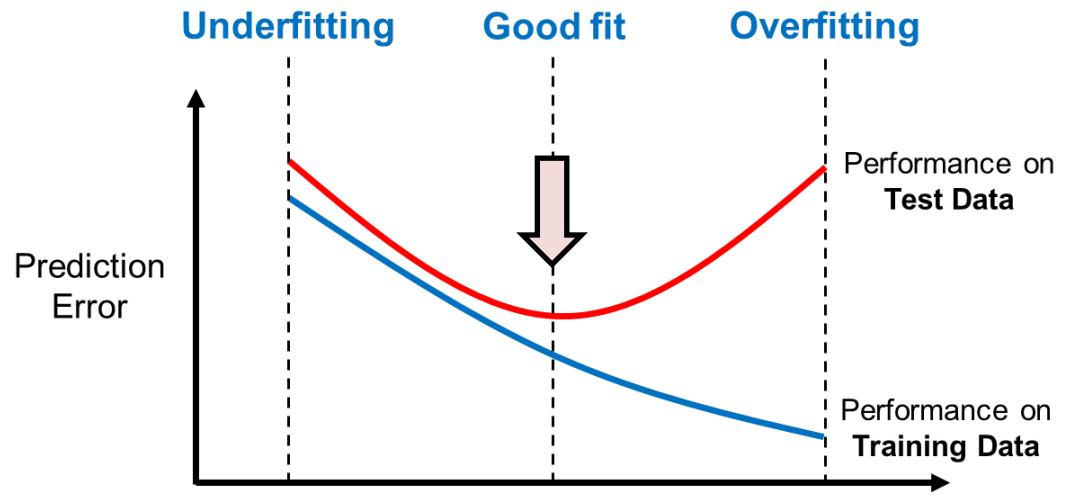
Bias = error from having wrong / too simple assumptions in the learning algorithm.

Variance = error resulting from sensitivity to the noise / fluctuations in the training data.

σ^2 (Irreducible error) = error resulting from noise in the problem itself.

In linear regression, the model becomes *more complex* when:

- More basis functions are added.
- Regularization λ decreases to 0.

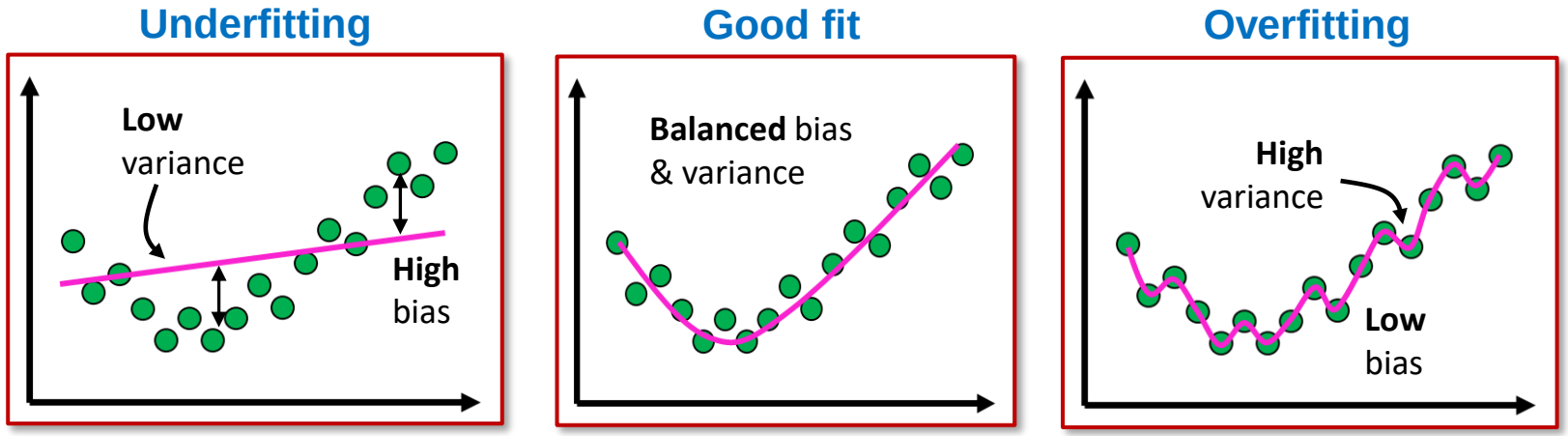


Model is too simple

High Bias, Low Variance

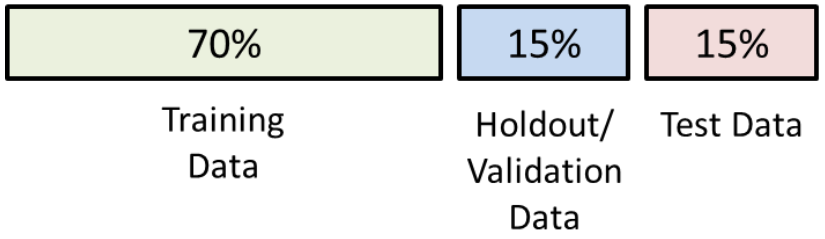
Model is too complex

Low Bias, High Variance

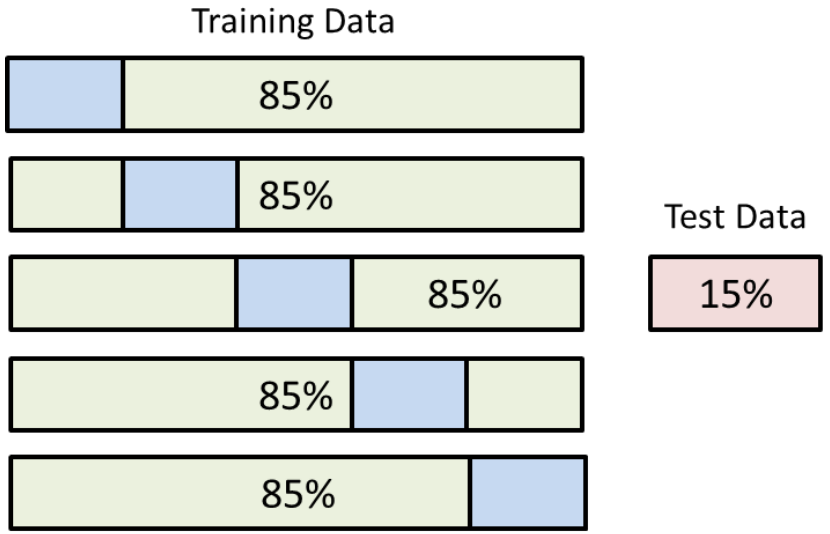


How to Avoid Over- or Underfitting?

1. Holdout Validation



2. *k*-fold Cross-Validation



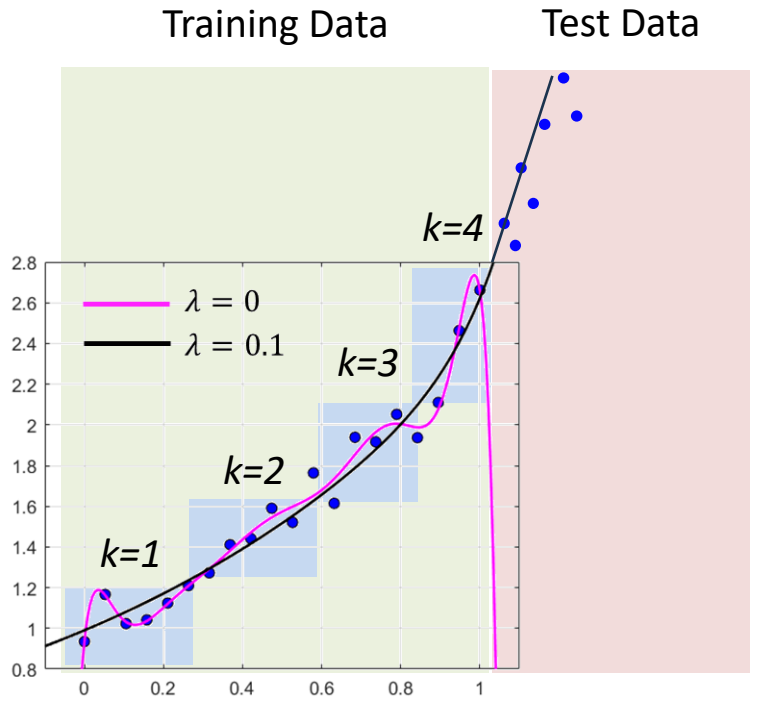
Steps:

1. Split the data into training, validation, and testing data.
2. Train the model with an initial guess on its complexity.
3. Find the model complexity that performs **best** on the validation data. Use a suitable search method.
4. Make one final performance check on the test data.

Applying 4-fold cross-validation to our example…

We use the available data set wisely by **training** a model to only 75% of the points and **validating** it on 25% of the points, then repeat it 4 times with different subsets each time.

Then, get a new test data set as a final test if the model can still be accurate for unseen data. If yes, then we say that the model can **generalize well**.



Holdout Validation

Example 5: Auto MPG Data Set

Source: https://archive.ics.uci.edu/ml/datasets/Auto+MPG

The data set consists of 392 car samples, with 4 columns of attributes:

X1 - Displacement,
X2 - Horsepower,
X3 - Weight,
X4 - Acceleration,

and 1 column of target variable:

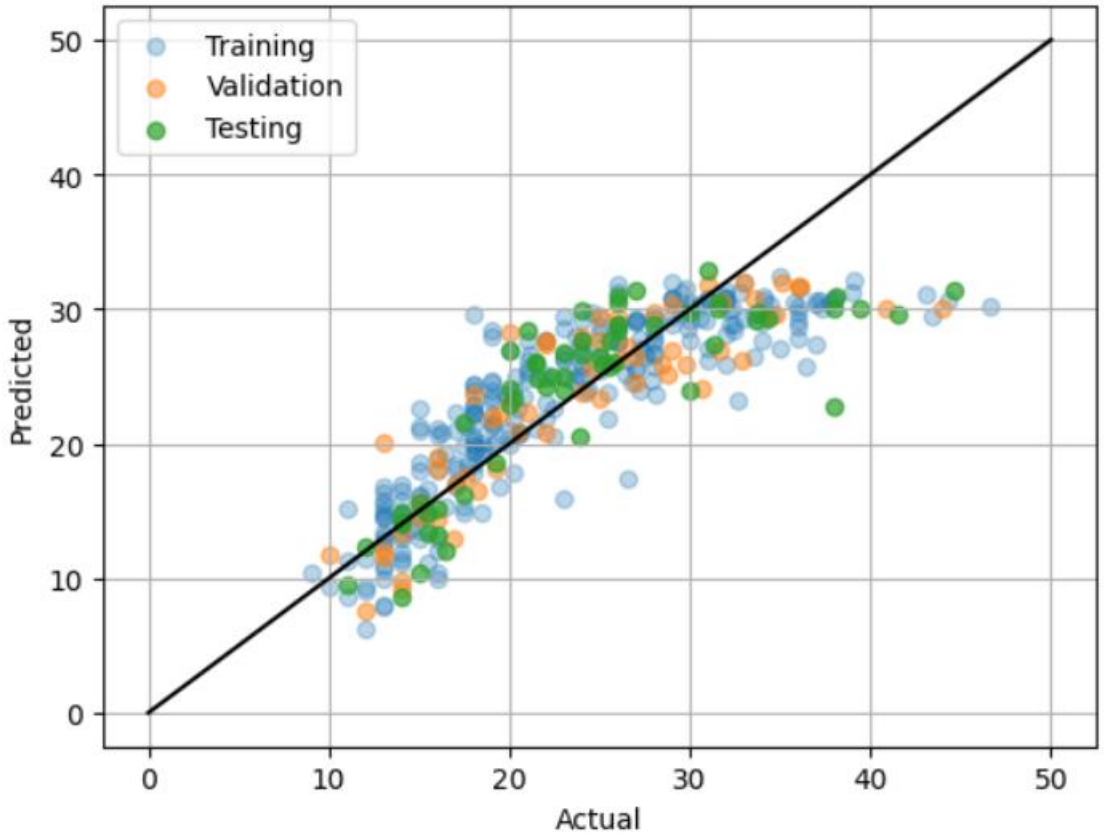
Y – MPG (miles per gallon) or mileage.

Split the data into 70% training, 15% validation, and 15% testing samples. Perform linear regression on the *training data* to predict the MPG, with these values of regularization:

$$\text{alpha} = 0, 0.1, 0.5, 1, 5, 10, 50, 100$$

Pick the alpha with the highest R^2 in the *validation data*. Then make one final evaluation of this best model on the *test data*.

	mpg	displacement	horsepower	weight	acceleration
0	18.0	307.0	130.0	3504	12.0
1	15.0	350.0	165.0	3693	11.5
2	18.0	318.0	150.0	3436	11.0
3	16.0	304.0	150.0	3433	12.0
4	17.0	302.0	140.0	3449	10.5
..	...	...	...	...	...
393	27.0	140.0	86.0	2790	15.6
394	44.0	97.0	52.0	2130	24.6
395	32.0	135.0	84.0	2295	11.6
396	28.0	120.0	79.0	2625	18.6
397	31.0	119.0	82.0	2720	19.4



Results:

Coefficients:
[-1.91969233 -1.58158997 -2.88250108 -0.08746244]

Intercept: 23.16897810218978

Training accuracy (R^2): 70.92%
Validation accuracy (R^2): 73.00%
Testing accuracy (R^2): 62.36%

Best alpha: 10

Locally Weighted Linear Regression

One way to avoid choosing features or basis functions.

Linear Regression:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

$$\boxed{y = \boldsymbol{w}^T \boldsymbol{x}}$$

Locally Weighted Linear Regression (LWR):

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_m x_m$$

$$\boxed{y = \boldsymbol{w}^T \boldsymbol{x}}$$

Cost function *for Linear Regression*

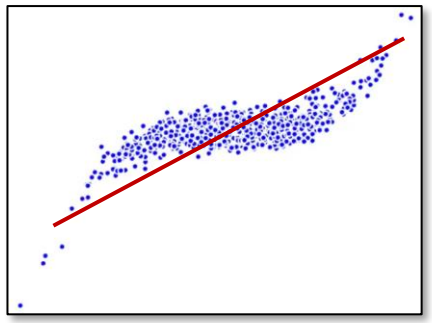
Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \sum_i (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

$$\boxed{\hat{\boldsymbol{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{y}}$$

Linear Regression might work but only with carefully chosen basis functions.



Cost function *for Locally Weighted Linear Regression*

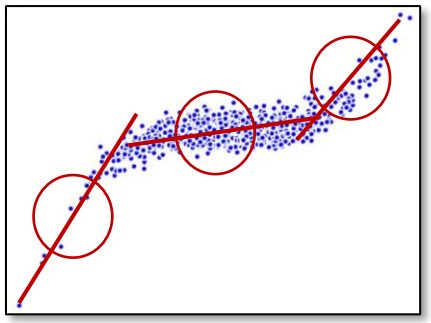
Cost function:

Need to find $\boldsymbol{w}$ such that $f(\boldsymbol{w})$ is minimum.

$$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \sum_i \omega_i(\boldsymbol{x}) \times (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

$$\boxed{\hat{\boldsymbol{w}} = (\mathbf{X}^T \boldsymbol{W} \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{W} \boldsymbol{y}}$$

No need to choose basis functions; but now there is a *weighting function*, ω_i.



$$\omega_i(\boldsymbol{x}) = \exp\left(-\frac{(\boldsymbol{x}_i - \boldsymbol{x})^2}{2\tau^2}\right)$$

where:

- ω_i is a weighting function
- $\boldsymbol{x}$ is the query input
- τ is the bandwidth

Locally Weighted Linear Regression

One way to avoid choosing features or basis functions.

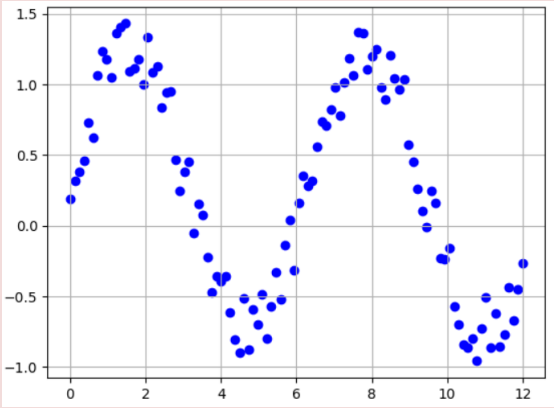
Example 6: Sine Data Set

The following data set was generated from a sine wave with random noise:

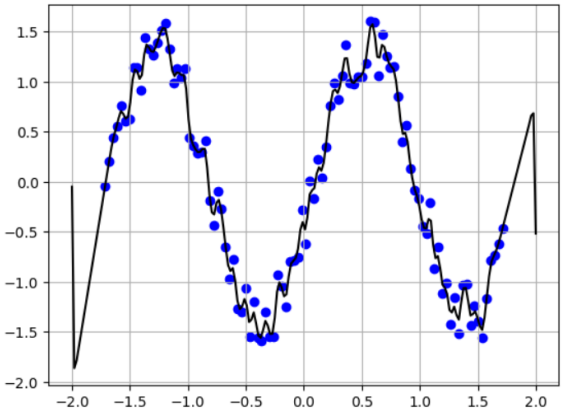
$$y = \sin x + \varepsilon$$

where $\varepsilon \sim \mathcal{N}(0, 0.5)$ is Gaussian distributed.

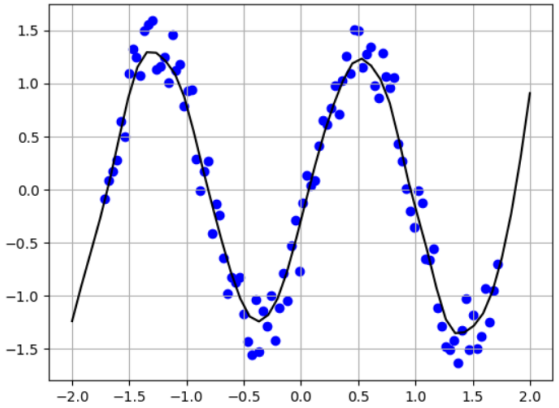
Fit a locally weighted linear regression model with $\tau = 0.1$.



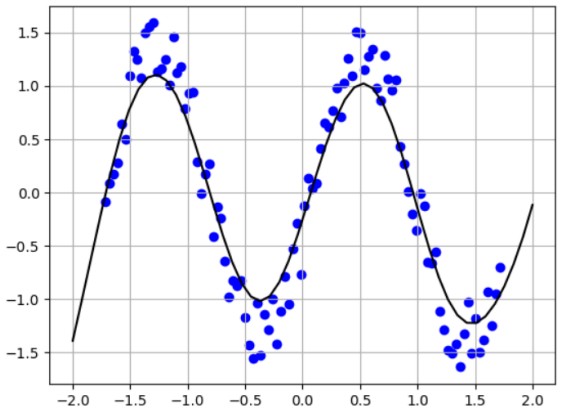
$\tau = 0.02$



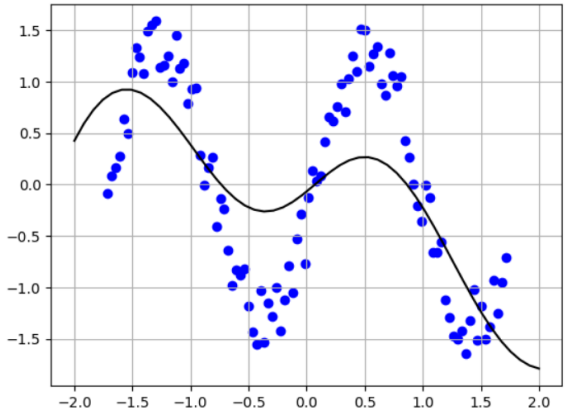
$\tau = 0.1$



$\tau = 0.2$



$\tau = 0.5$



Outline

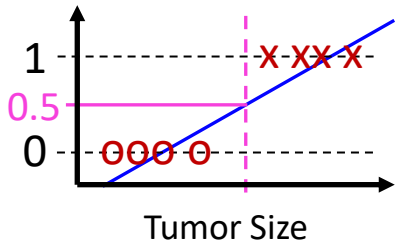
- Linear Regression
  - Solution to Linear Least Squares
  - Linear Basis Function Model
  - Performance Metrics
  - Ridge Regularization
  - Locally Weighted Linear Regression

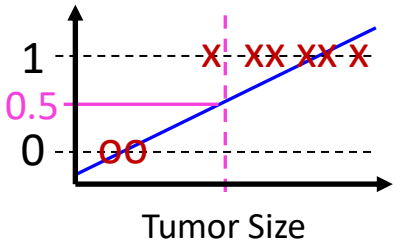
- **Logistic Regression**
  - Binary Classification Problem
  - Log-loss Cost Function
  - Performance Metrics

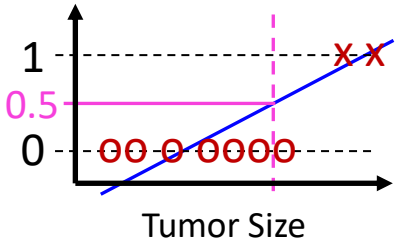
Classification Problems

- In classification, the targets (y) are discrete or categorical, whereas in regression, the targets are continuous.
- If the target only has 2 classes, then it's called *binary classification*.
- Examples of classification problems:
  - Spam vs. Non-spam email (*2 classes*)
  - Benign vs. Malignant Tumor (*2 classes*)
  - Handwritten Digits Classification (*10 classes*)
  - In Engineering:
    - Normal vs. Faulty equipment (*2 classes*)
    - Flow regime classification (*3 or more*)
    - Wastewater effluent (5 *classes*)
    - Land Use (4 *classes*)
    - Images in manufacturing (3 *or more*)
    - Images in general ('00,000s *of classes*)

Linear Regression will not work well for classification:







Best-fit line
Predicted Threshold
x Malignant samples
o Benign samples

- If we force a best-fit line on discrete-valued y data, we can predict a classifier threshold x at the point where the line crosses $y = 0.5$.
- However, the examples on the left show that this is a bad idea.
- We should fit a nonlinear function on the data set to perform classification.

*Example taken from Andrew Ng, Coursera, Stanford University

Classification Problems

Linear Regression

$$y = w_0 + w_1 x$$

$$y = \boldsymbol{w}^T \boldsymbol{\phi}(x)$$

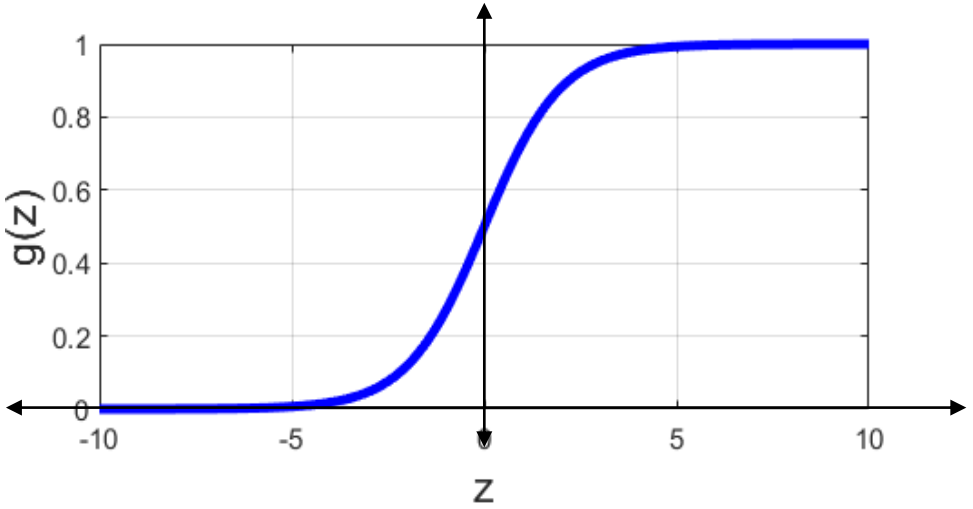
Logistic Regression

$$y = g(w_0 + w_1 x)$$

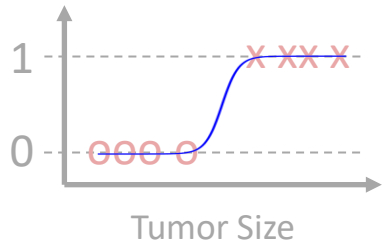
$$y = g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x)\right)$$

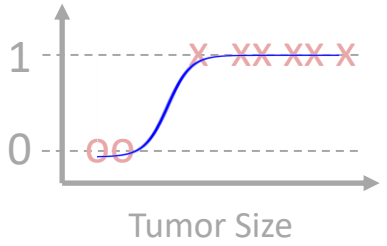
where $g(z) = \dfrac{1}{1 + e^{-z}}$

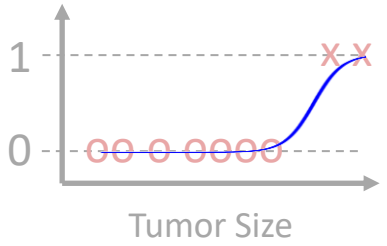
(sigmoid function or logistic function)



Fitted sigmoid functions:







Sigmoid Function

- Outputs a value from 0 to 1 only, regardless of the input value from $-\infty$ to $+\infty$. Hence, it is an example of a *squashing function*.
- $\lim_{z \to +\infty} g(z) = \lim_{z \to +\infty} \frac{1}{1+e^{-z}} = \mathbf{1}$
- $\lim_{z \to -\infty} g(z) = \lim_{z \to -\infty} \frac{1}{1+e^{-z}} = \mathbf{0}$
- $g(z)$ crosses the value of **0.5** when $z = 0$.
- The locus of points where $g(z) = 0.5$ is called the *decision boundary*.
- The output of $g(z)$ is called a *score*. It can be interpreted as the "estimated probability that $y = 1$, given an input x."

If $g(z) \geq 0.5$, predict Class 1.
If $g(z) < 0.5$, predict Class 0.

Logistic Regression

	Linear Regression	Logistic Regression
Prediction Model	$y = w_0 + w_1 x$ $y = \boldsymbol{w}^T \boldsymbol{\phi}(x)$	$y = g(w_0 + w_1 x)$ $y = g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x)\right)$
Cost Function	$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \sum_{i=1}^{N} \left(y_i - \boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)^2$ Sum-of-squares loss	$\min_{\boldsymbol{w}} \quad f(\boldsymbol{w}) = \sum_{i=1}^{N} Cost(\boldsymbol{w}, x_i, y_i)$ Cross-entropy loss or log-loss

where

$$g(z) = \frac{1}{1 + e^{-z}}$$

(sigmoid function or logistic function)

where $\quad Cost(\boldsymbol{w}, x_i, y_i) = \begin{cases} -\log\left(g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right), & \text{if } y_i = 1 \\ -\log\left(1 - g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right), & \text{if } y_i = 0 \end{cases}$

which can also be written as:

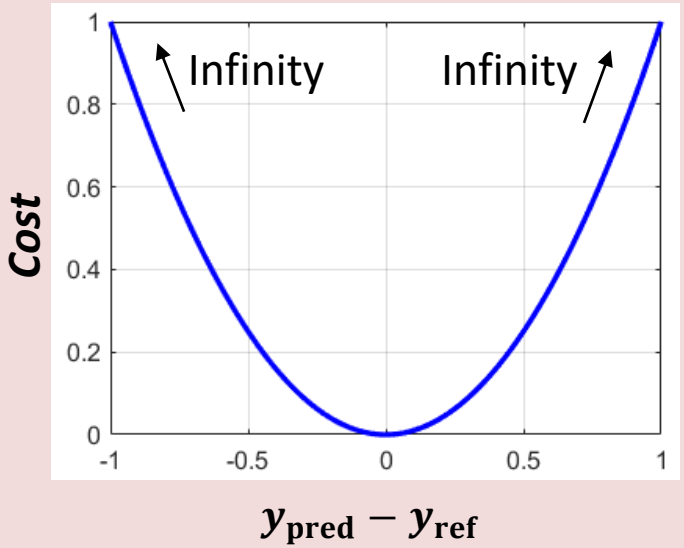
$$Cost(\boldsymbol{w}, x_i, y_i) = -y_i \log\left(g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right) - (1 - y_i) \log\left(1 - g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right)$$

Logistic Regression

What is the behavior of the cross-entropy loss?

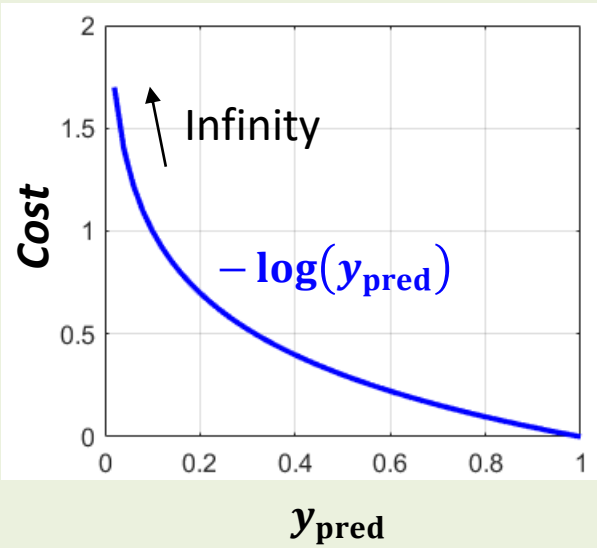
$$Cost(\boldsymbol{w}, x_i, y_i) = -y_i \log\left(g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right) - (1 - y_i) \log\left(1 - g\left(\boldsymbol{w}^T \boldsymbol{\phi}(x_i)\right)\right)$$

Behavior of Sum-of-Squares Loss

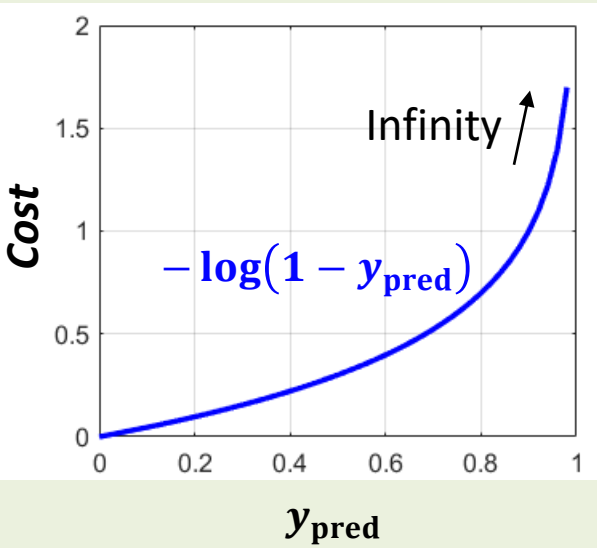


Behavior of Cross-entropy Loss

If $y_{\text{ref}} = 1$ (Positive class),



If $y_{\text{ref}} = 0$ (Negative class),



- For classification models, the cross-entropy loss makes training faster and improves generalization.

Logistic Regression

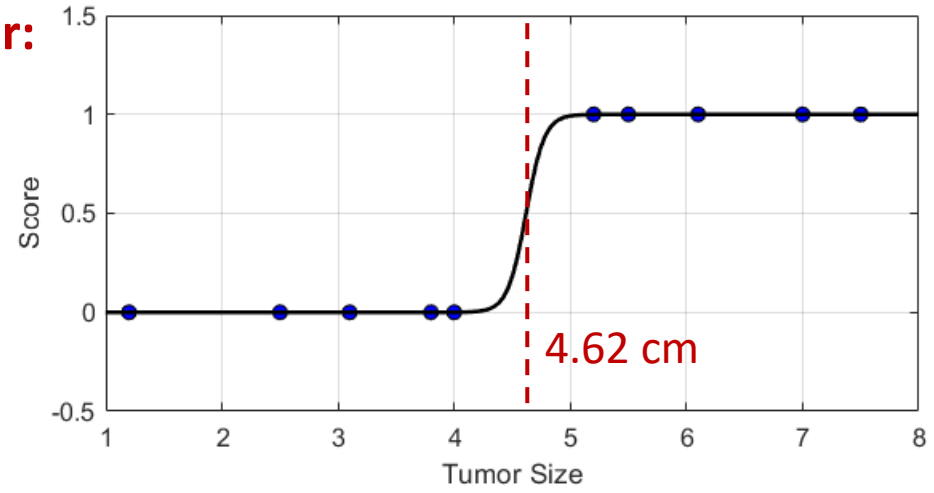
Example 7: Tumor Size Classification

Consider a hypothetical data set of tumor size (in cm) versus whether they are benign (0) or malignant (1):

Size	1.2	3.1	4.0	3.8	2.5	7.0	5.2	5.5	6.1	7.5
Class	0	0	0	0	0	1	1	1	1	1

Train a **linear classifier** by logistic regression.

Answer:

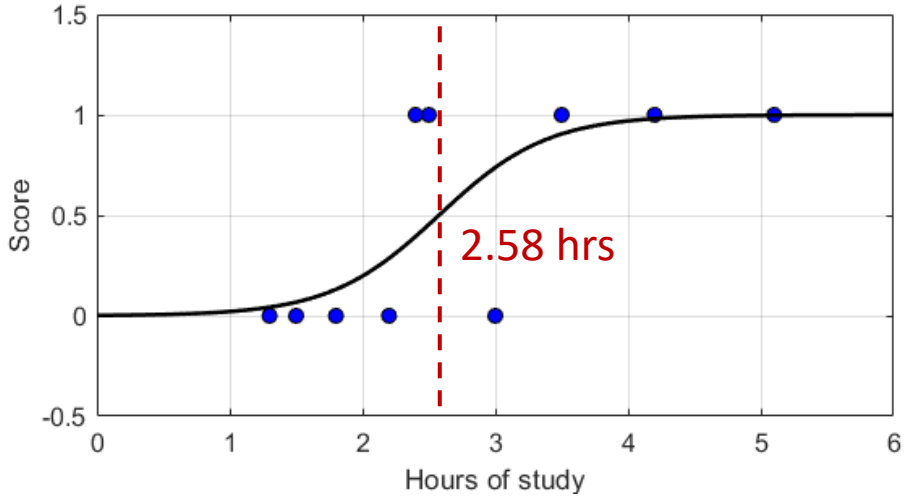


Example 8: Student Hours of Study

Based on the following student data, how many hours should we study in order to pass?

Hours of study	4.2	2.2	3.5	2.5	5.1	1.3	3.0	1.8	2.4	1.5
Pass?	1	0	1	1	1	0	0	0	1	0

Train a **linear classifier** by logistic regression.



Classification: Performance Metrics

A common way to report the performance of a *binary classifier* is to use a confusion matrix.

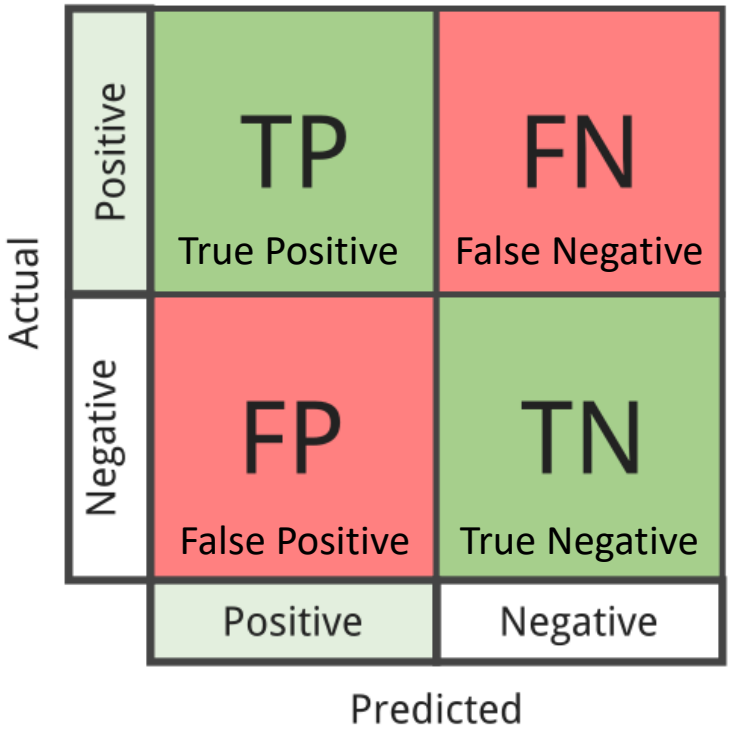
Confusion Matrix

Actual \ Predicted	Positive	Negative
Positive	TP True Positive	FN False Negative
Negative	FP False Positive	TN True Negative

Classification: Performance Metrics

A common way to report the performance of a *binary classifier* is to use a confusion matrix.

Confusion Matrix



↑ $\text{Recall} = \frac{TP}{TP + FN}$
(Sensitivity)
(True Positive Rate)

Among the *actually* malignant tumors, how many were correctly detected as malignant?

↑ $\text{Precision} = \frac{TP}{TP + FP}$

Among the tumors *predicted* to be malignant, how many were *actually* malignant?

↑ $\text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$

Harmonic mean of recall and precision, which captures both metrics.

↑ $\text{Accuracy} = \frac{TP + TN}{\text{Total no. of samples}}$

Among *all* the tumor samples, how many were correctly predicted in either class?

↓ $\text{False Alarm Rate} = \frac{FP}{FP + TN}$
(False Positive Rate)

Among the *actually* benign tumors, how many were wrongly predicted as malignant?

↓ $\text{Missed Detection Rate} = \frac{FN}{TP + FN}$

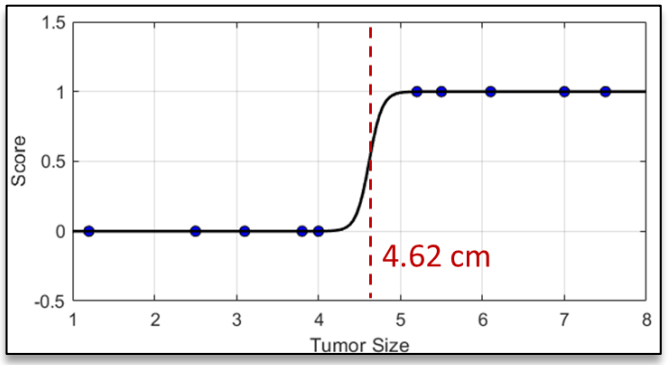
Among the *actually* malignant tumors, how many were missed to be detected?

*These are what we desire for a good classifier.

Logistic Regression

Example 7: Tumor Size Classification

Size	1.2	3.1	4.0	3.8	2.5	7.0	5.2	5.5	6.1	7.5
Actual Class	0	0	0	0	0	1	1	1	1	1
Predicted Class	0	0	0	0	0	1	1	1	1	1



Confusion Matrix

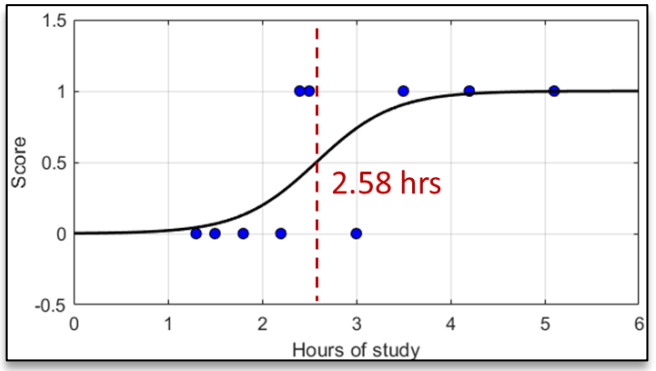
Actual \ Predicted	M	B
M	5	0
B	0	5

$$\text{Recall} = \frac{TP}{TP + FN} = \boxed{1.00} \qquad \text{Accuracy} = \frac{TP + TN}{\text{Total no. of samples}} = \boxed{1.00}$$

$$\text{Precision} = \frac{TP}{TP + FP} = \boxed{1.00} \qquad \text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} = \boxed{1.00}$$

Example 8: Student Hours of Study

Hours of study	4.2	2.2	3.5	2.5	5.1	1.3	3.0	1.8	2.4	1.5
Actual Class	1	0	1	1	1	0	0	0	1	0
Predicted Class	1	0	1	0	1	0	1	0	0	0



Confusion Matrix

Actual \ Predicted	P	F
P	3	2
F	1	4

$$\text{Recall} = \frac{TP}{TP + FN} = \boxed{0.60} \qquad \text{Accuracy} = \frac{TP + TN}{\text{Total no. of samples}} = \boxed{0.70}$$

$$\text{Precision} = \frac{TP}{TP + FP} = \boxed{0.75} \qquad \text{F1} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} = \boxed{0.67}$$

Classification: Performance Metrics

There is a whole plethora of other metrics that can be derived from the confusion matrix. Note: Different fields of study use different metrics.

Sources: [1][2][3][4][5][6][7][8][9] view · talk · edit

		Predicted condition			
	Total population $= P + N$	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

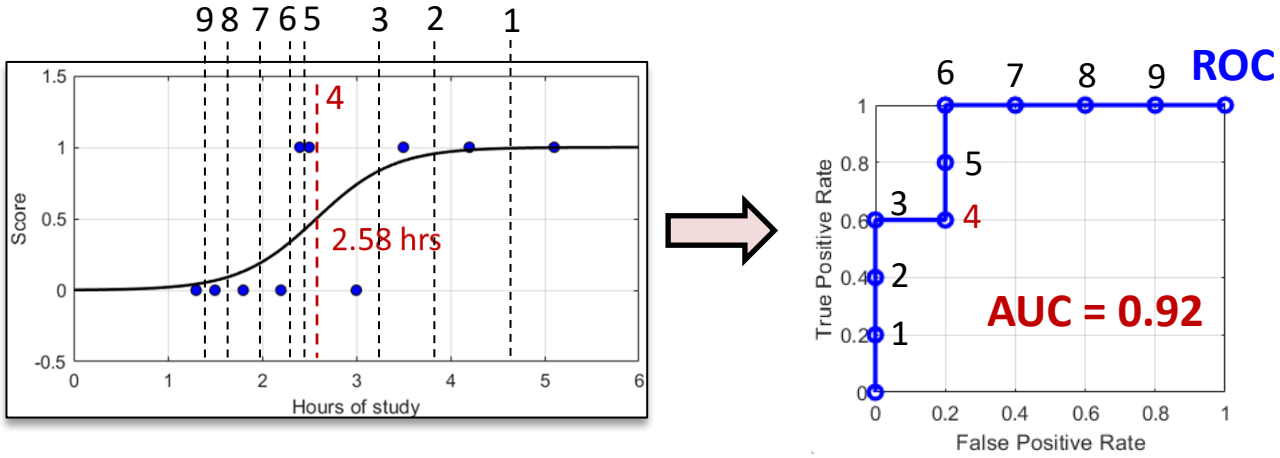
Source: Wikipedia, Retrieved from https://en.wikipedia.org/wiki/Template:Diagnostic_testing_diagram

Classification: Performance Metrics

We can compare the performance of different binary classifiers on the same task *visually* using their **ROC curves**.

ROC = Receiver Operating Characteristic

A 2-D plot of the True Positive Rate (TPR) vs. False Positive Rate (FPR) obtained while moving the decision boundary across all data points.



Steps to Create an ROC Curve:

1. Sort all the scores $g(z)$ computed by the trained binary classifier.
2. Compute the TP rate and FP rate, (TPR_i, FPR_i), by assuming different decision boundaries from $g(z) = 0$ to 1 rather than at 0.5.
3. Plot the TP rate vs. FP rate.

Perfect classifier

Notes:

- The perfect classifier will reach the point $(0, 1)$.
- A classifier that predicts either class at random will have the *diagonal line* as an ROC curve.
- We can compute the **area under the curve (AUC)** of the ROC curve. The higher the AUC, the more trustworthy the classifier.
- The Gini index (G) is related to the AUC by: $G = 2 \cdot AUC - 1$.
- To gain strength from different classifiers, you can keep only the classifiers that define the *ROC convex hull*.

Outline

- Linear Regression
  - Solution to Linear Least Squares
  - Linear Basis Function Model
  - Performance Metrics
  - Ridge Regularization
  - Locally Weighted Linear Regression

- Logistic Regression
  - Binary Classification Problem
  - Log-loss Cost Function
  - Performance Metrics

Further Reading

- Hastie et al. (2008). *The Elements of Statistical Learning*. 2nd Ed. Springer.
- Bishop (2006). *Pattern Recognition and Machine Learning*. Springer.
- Bishop (1995). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, vol. 7, no. 1. https://ieeexplore.ieee.org/abstract/document/6796505
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
- https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html
- https://realpython.com/linear-regression-in-python/
- An introduction to ROC analysis: https://people.inf.elte.hu/kiss/11dwhdm/roc.pdf
- Andrew Ng. CS 229 ML: https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU