510 DATA SCIENCE

Lecture 07

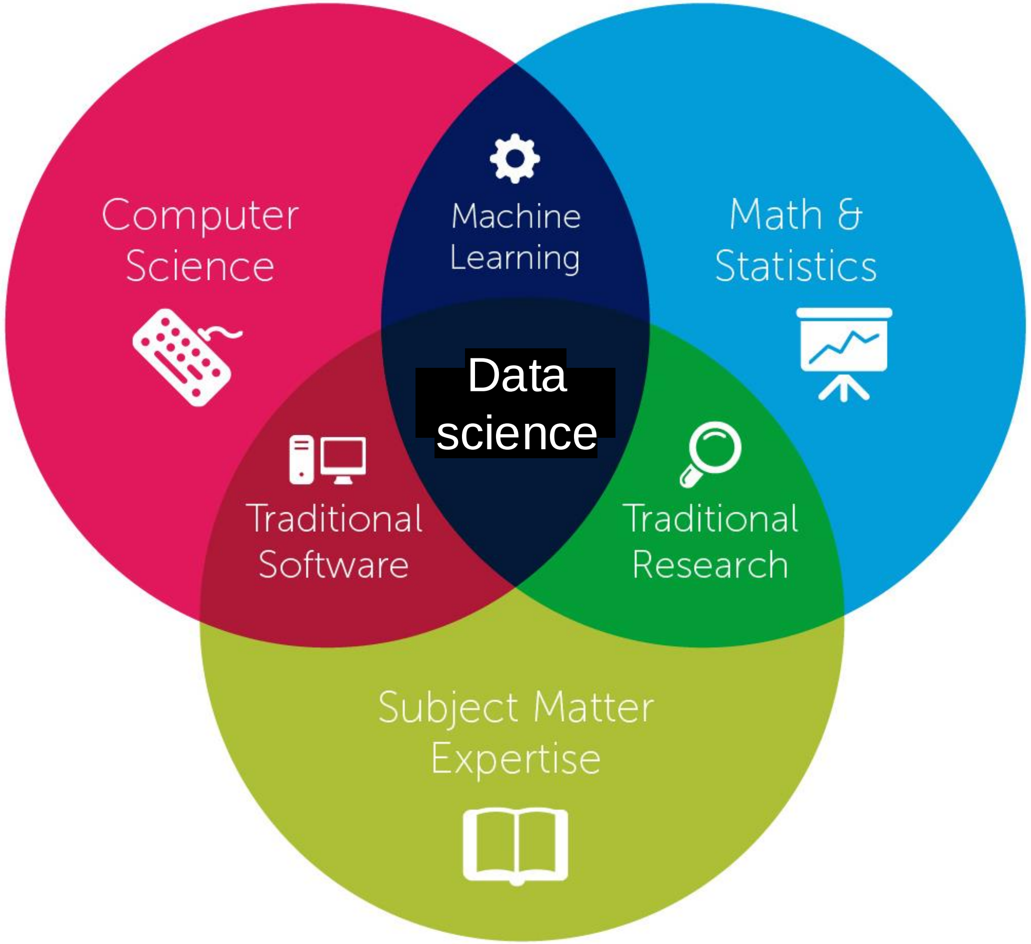
Fall 2024

Instructor: Assoc. Prof. Şener Özönder

Email: sener.ozonder@bogazici.edu.tr

Institute for Data Science & Artificial Intelligence

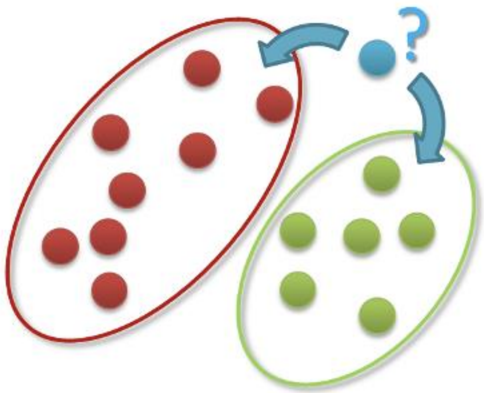
Boğaziçi University



Modeling

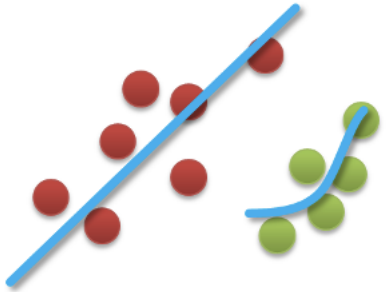
- The first thing in any project is to determine which approach is appropriate for the data and problem at hand.

Classification
(Supervised=with labels)



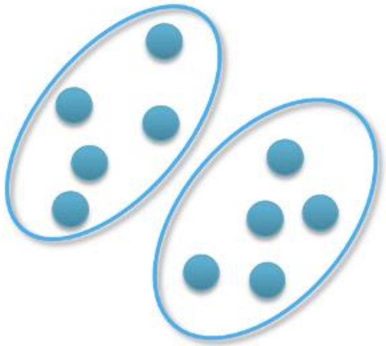
- Email is spam or not
- Image is cat or dog or horse
- Patient is cancer or not

Regression
(Supervised=with labels)



- House price based on rooms and neigborhood
- Remaining life based on tumor size

Clustering
(Unupervised=without labels)



- Clustering human groups like rich+old+average IQ people vs middle income+young+high IQ people

Modeling

- Always ask the "why question". Why do we even model data?

Inference/Prediction/Forecasting: On the unseen data

Understanding Relationships: What affects the house price the most? Neighborhood? Area?

Data Reduction: Ames housing dataset has 80 columns and not all of them affect SalePrice much.

Validation of Hypotheses: *"IQ and success in life are correlated."*

Generalization: Predicting election outcomes from surveys with limited people (a few thousand).

Automating Tasks: Such as computer vision in self driving cars.

Discovering Patterns: Which variables are correlated with each other?

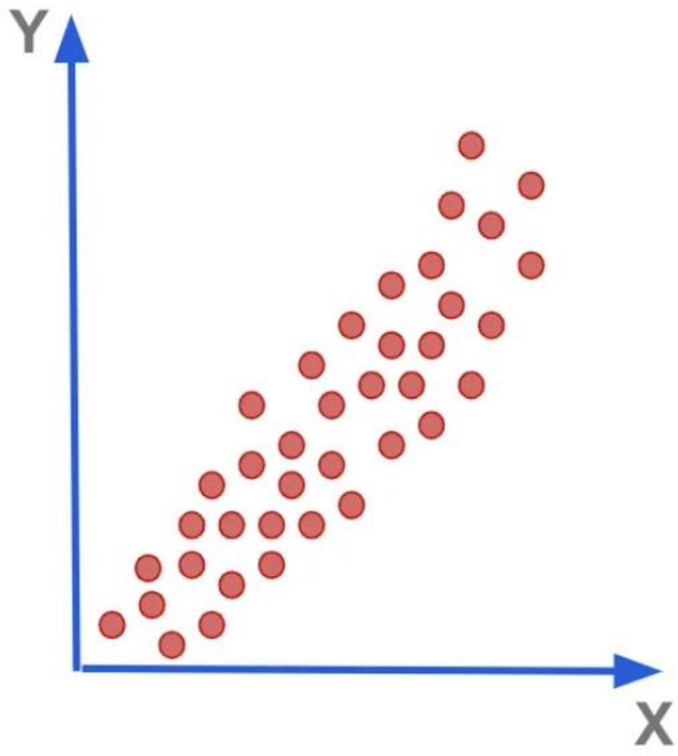
Optimization: Using models for maximum profit for the company.

Causal Inference: Which variables are the cause and which ones are the result?

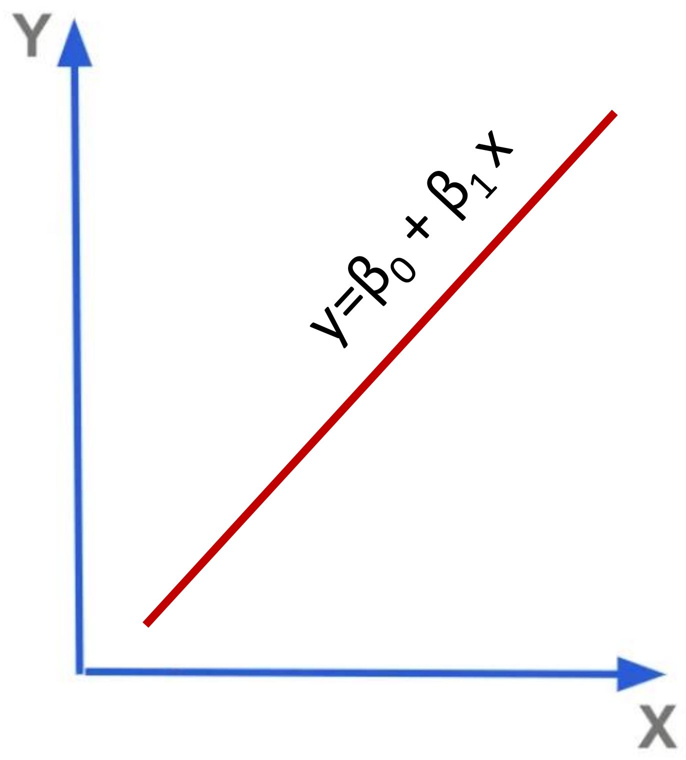
Anomaly and Fraud Detection: To trigger an alarm if the last purchase on the credit card is unusual.

Modeling

- Modeling also is lossy compression.



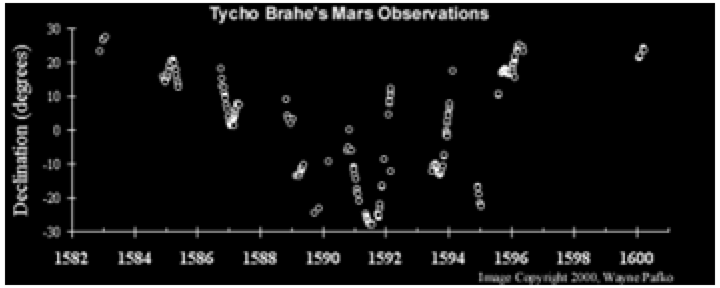
38 data points in the form (x,y)
→ 76 numbers.

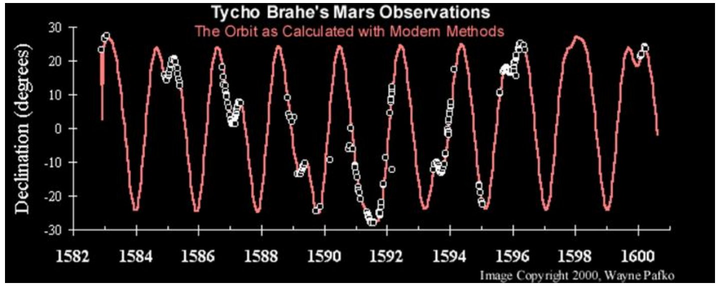


2 coefficients: β_0 and β_1.
→ just 2 numbers (for linear reg.)

Modeling

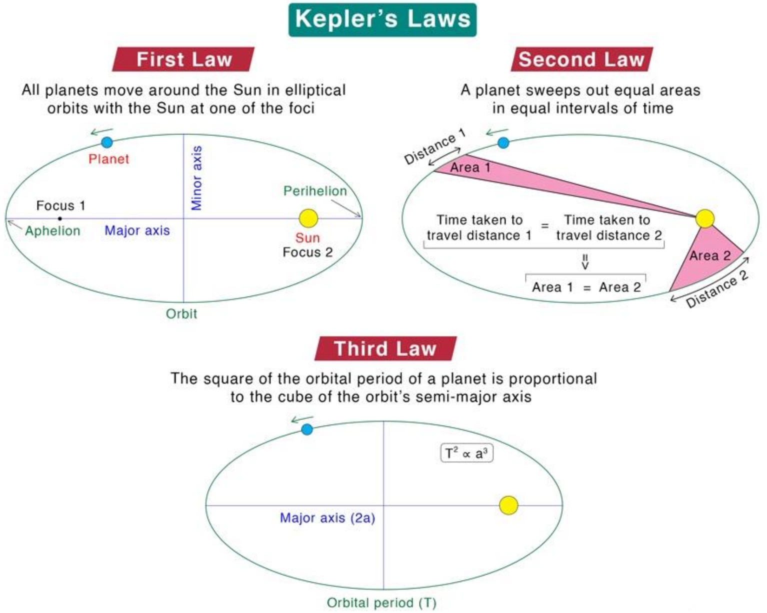
- Modeling also is a lossy compression.





Tycho Brahe
(1546-1601)

Johannes Kepler
(1571–1630)



- Kepler *compressed* the data Brahe collected on Mars's position over several years into two rules and one formula (the three Kepler's laws).

Linear Regression

For 1D linear regression
Model: *y=b + w x*
Data: *(x,y) pairs*
Unknown: *model coefficients/parameters b and w.*
For linear regression, b: intercept, w: slope.

- To find the trained model parameters, you need to train it on the data. Training is done by minimizing the loss function.
- Loss function or cost function or error measures how bad the model is. Loss function depends on both model parameters and the data!

$$L\left[\phi, \underbrace{\mathrm{f}[\mathbf{x}, \phi]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{I}}_{\text{train data}}\right]$$

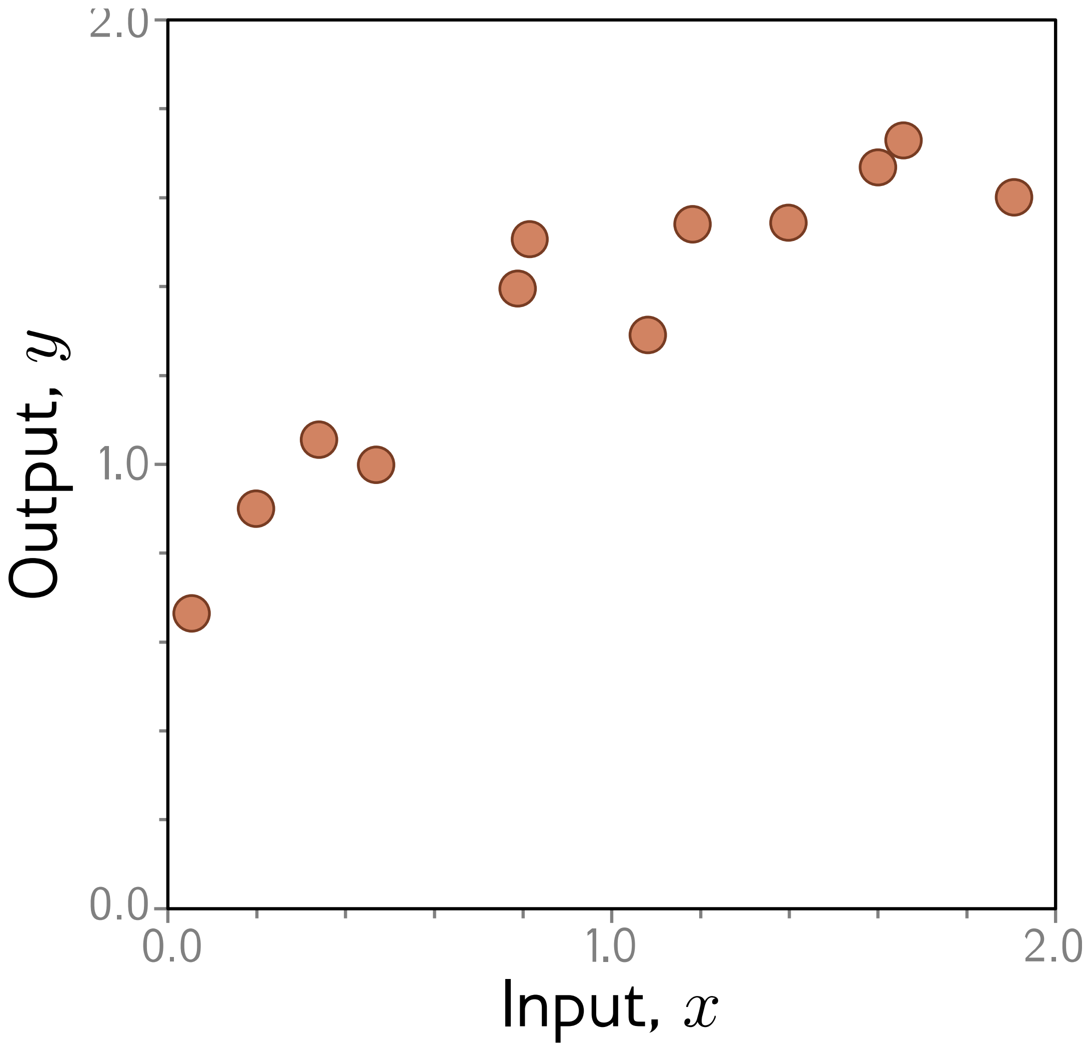
Model parameters ϕ are not known before training.

Loss L returns a scalar (number) for a specific model parameters. The smaller L is, the better the model is.

- The purpose of training is to find the set of model parameters ϕ that minimize the value of loss function.

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}}\Big[\mathrm{L}\left[\phi\right]\Big]$$

Linear Regression: 1D linear regression



"Least squares loss function"

dataset

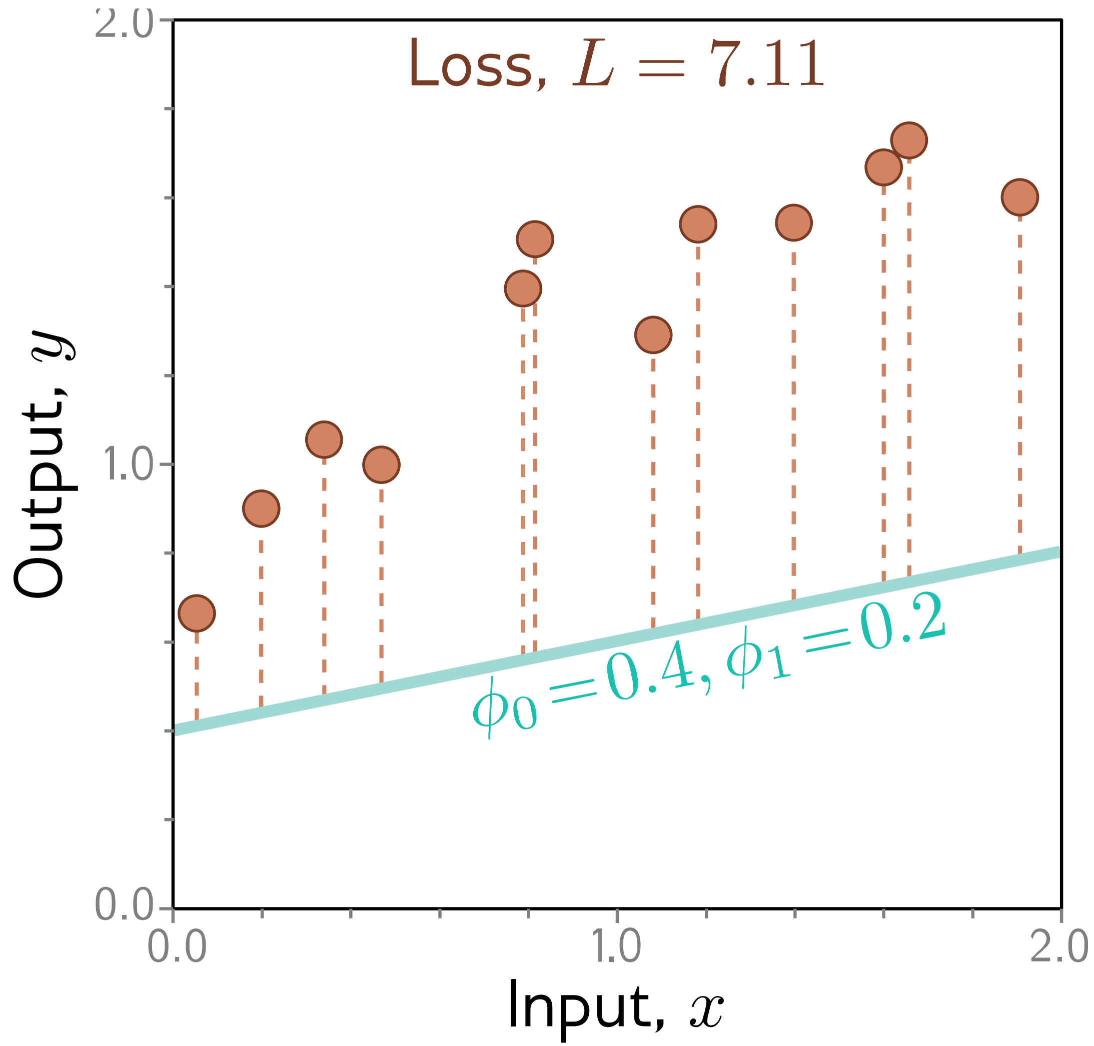
$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

1D linear reg model prediction

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 \underbrace{x_i}_{\text{data}} - \underbrace{y_i}_{\text{label}})^2$$

$$= (\phi_0 + \phi_1 0.1 - 0.65)^2$$
$$+ (\phi_0 + \phi_1 0.2 - 0.9)^2$$
$$+ (\phi_0 + \phi_1 0.35 - 1.1)^2$$
$$+ \ldots$$

Linear Regression: 1D linear regression



"Least squares loss function"

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

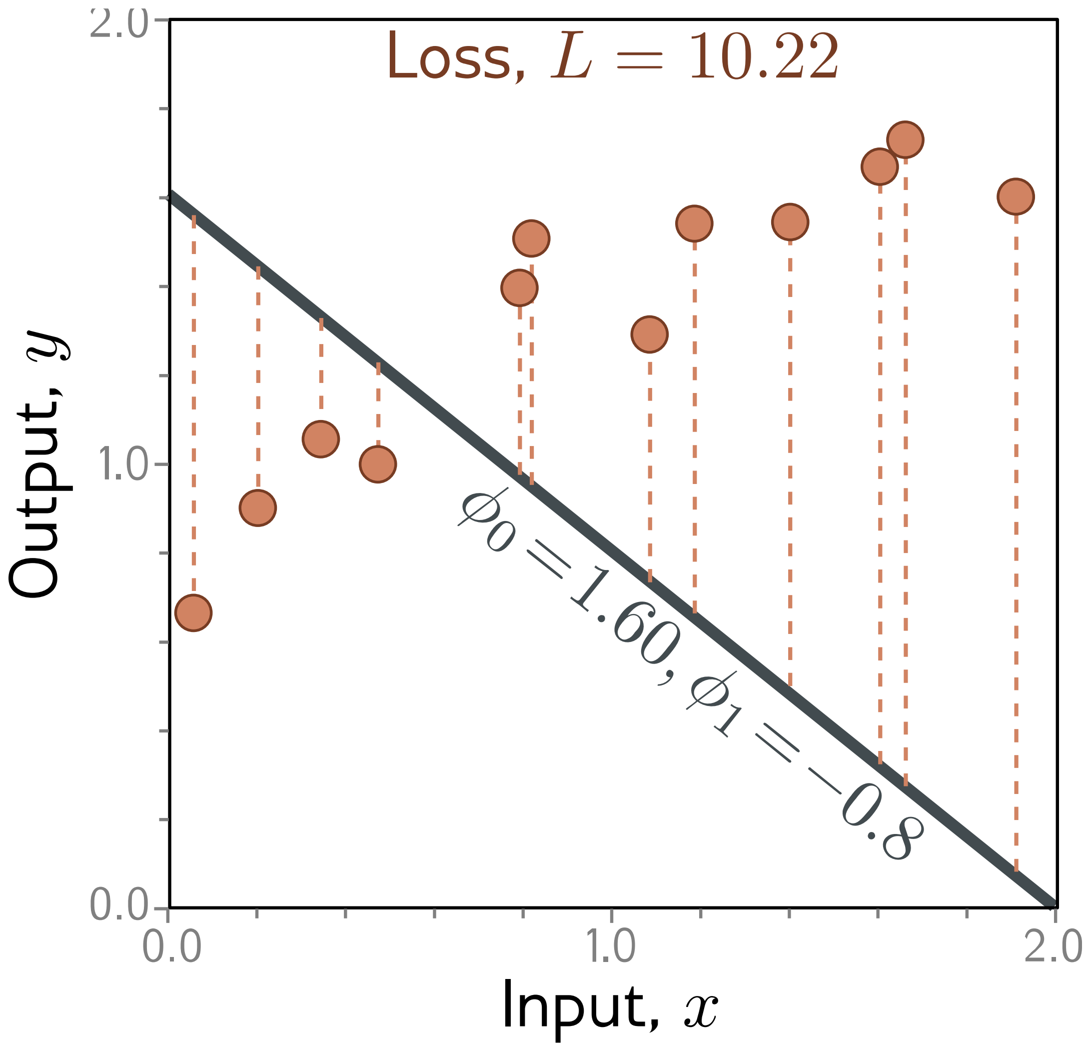
I → dataset

$$= \sum_{i=1}^{I} (\underbrace{\phi_0 + \phi_1 x_i}_{\text{1D linear reg model prediction}} - y_i)^2$$

x_i: data, y_i: label

$$= (\phi_0 + \phi_1 0.1 - 0.65)^2$$
$$+ (\phi_0 + \phi_1 0.2 - 0.9)^2$$
$$+ (\phi_0 + \phi_1 0.35 - 1.1)^2$$
$$+ \ldots$$

Linear Regression: 1D linear regression



"Least squares loss function"

dataset

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

1D linear reg model prediction

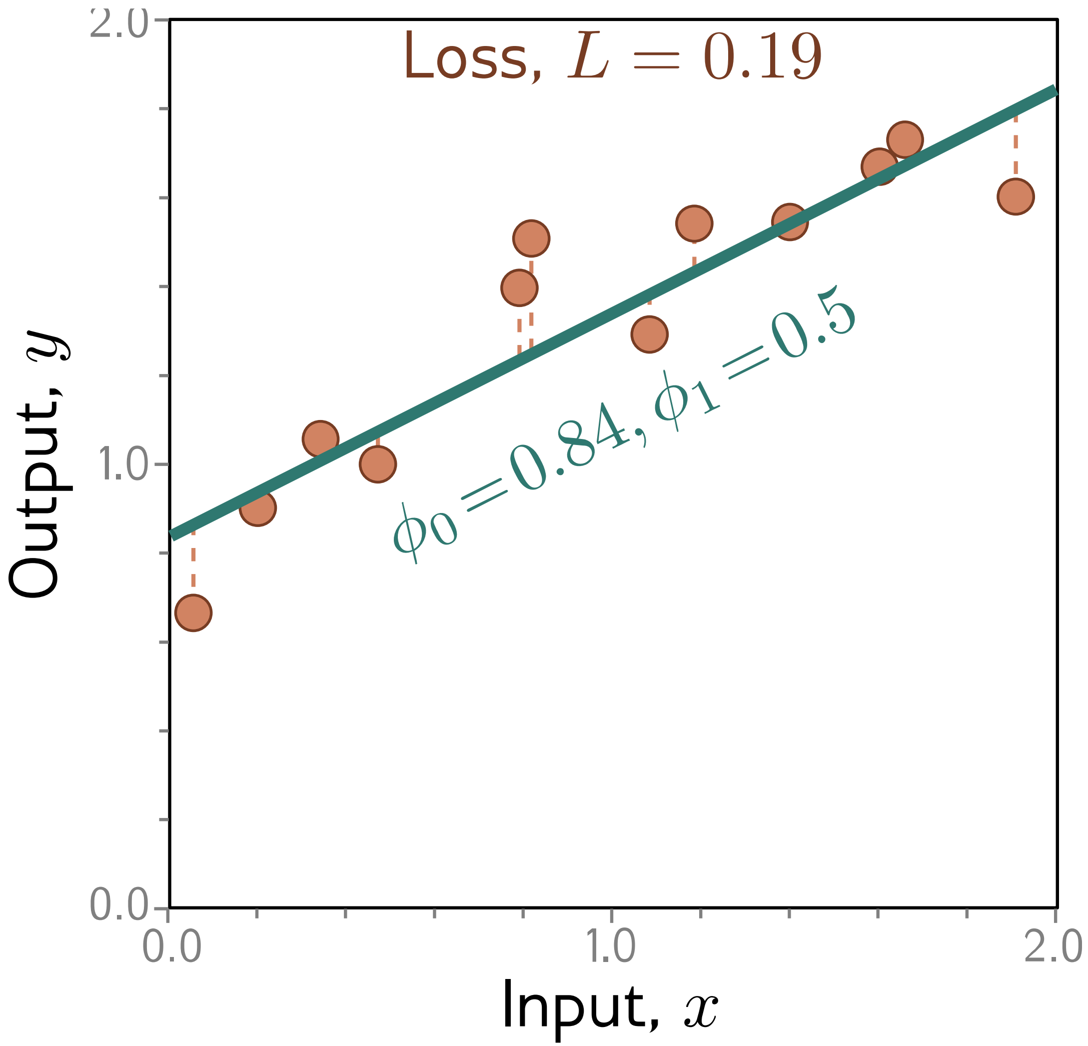
data

label

$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

$$= (\phi_0 + \phi_1 0.1 - 0.65)^2$$
$$+ (\phi_0 + \phi_1 0.2 - 0.9)^2$$
$$+ (\phi_0 + \phi_1 0.35 - 1.1)^2$$
$$+ \ldots$$

Linear Regression: 1D linear regression



"Least squares loss function"

$$L[\phi] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \phi] - y_i)^2$$

I → dataset

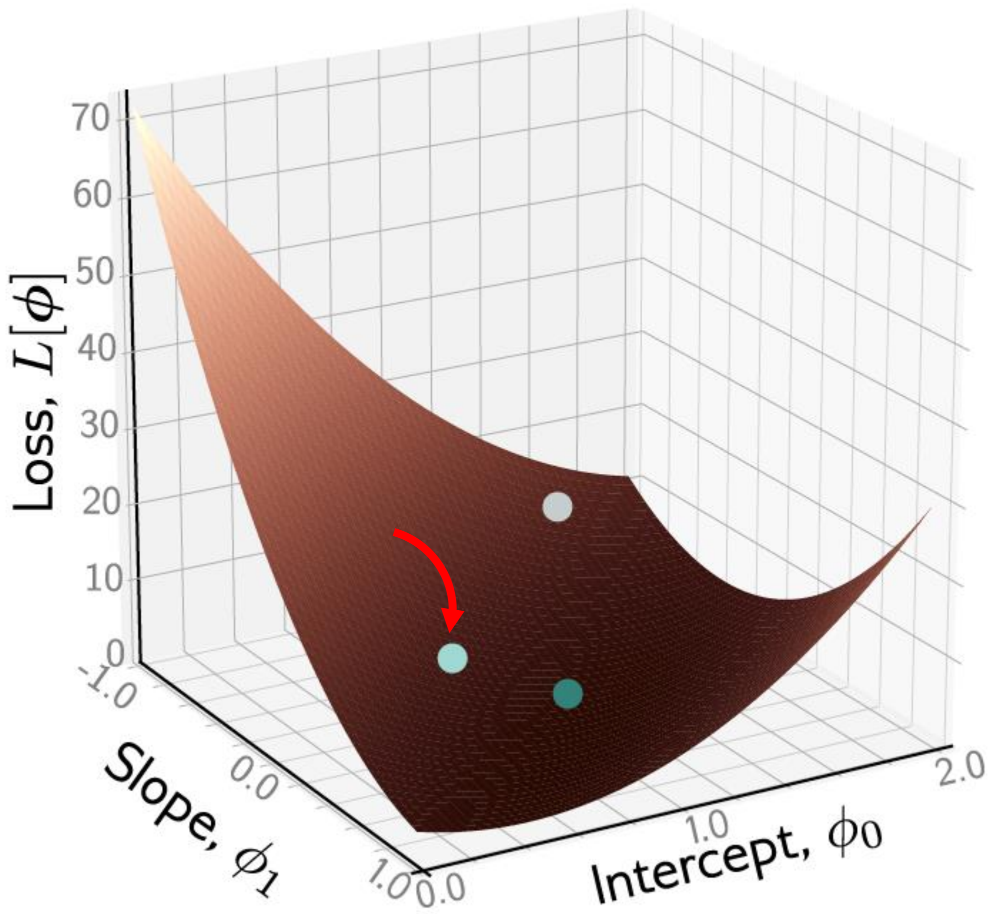
$$= \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

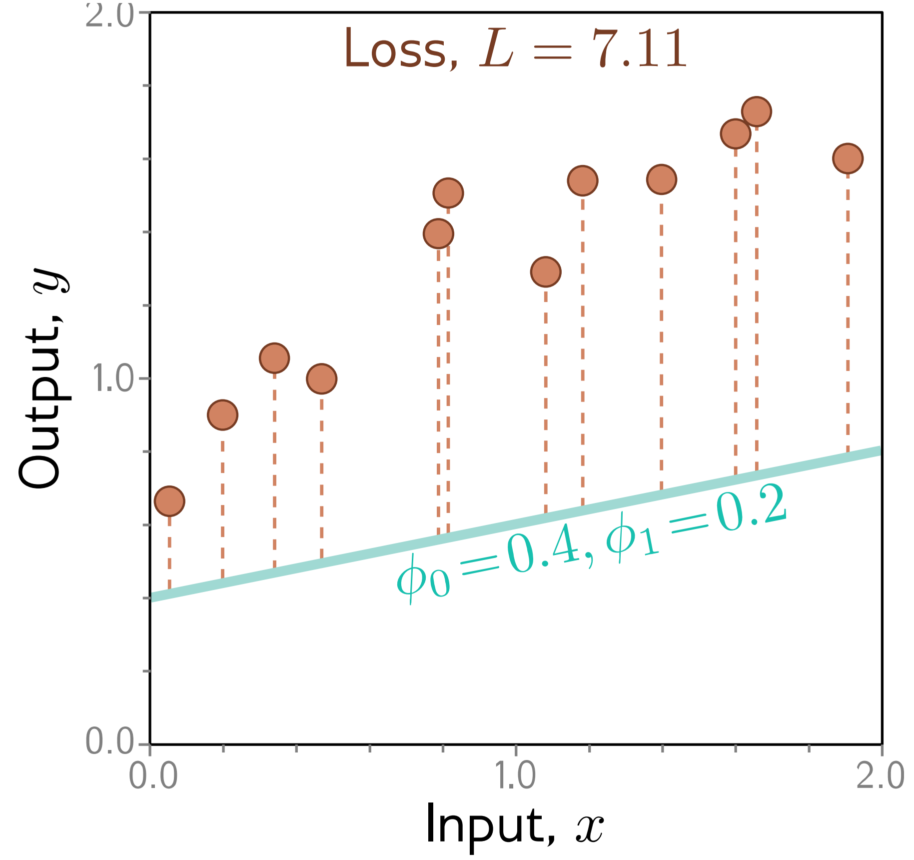
1D linear reg model prediction: $\phi_0 + \phi_1 x_i$

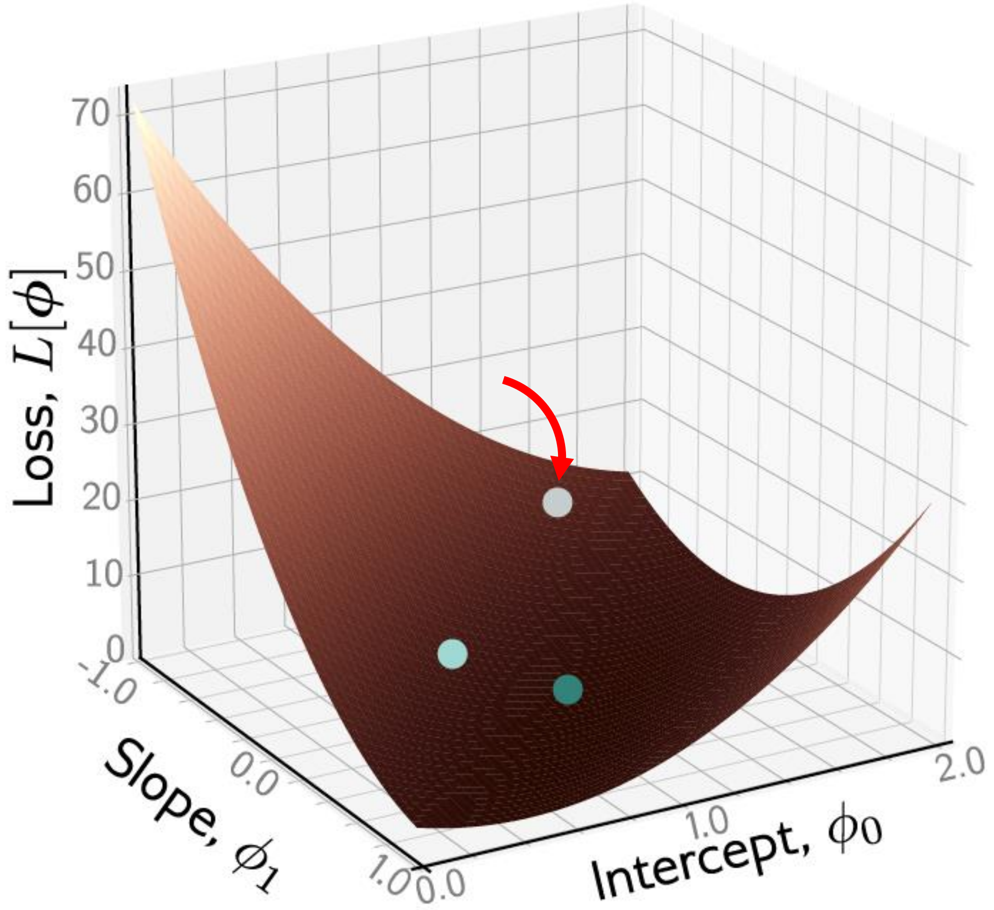
x_i: data; y_i: label

$$=(\phi_0 + \phi_1 0.1 - 0.65)^2$$
$$+ (\phi_0 + \phi_1 0.2 - 0.9)^2$$
$$+ (\phi_0 + \phi_1 0.35 - 1.1)^2$$
$$+ \ldots$$

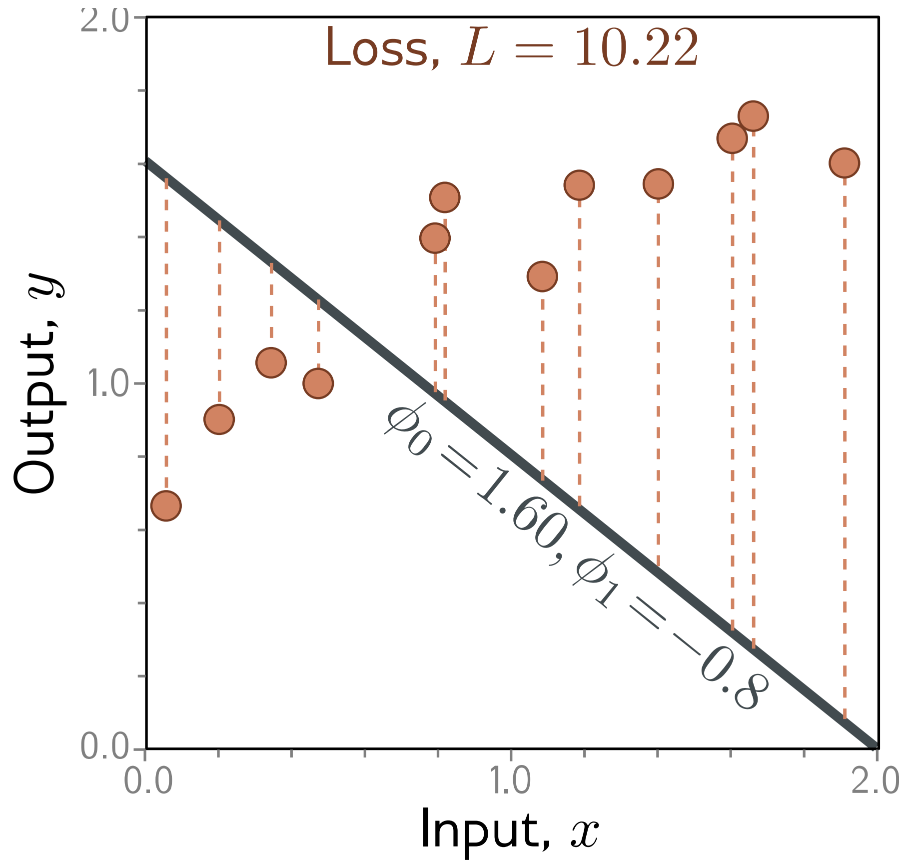
Linear Regression: 1D linear regression





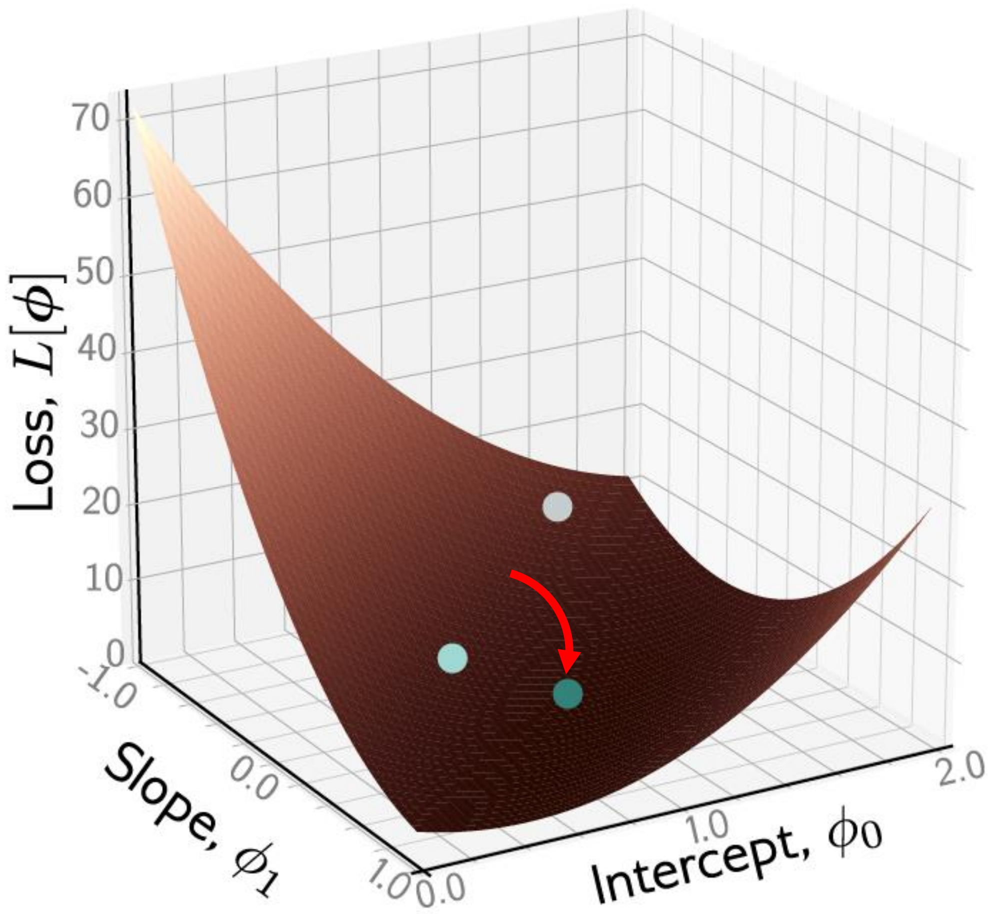


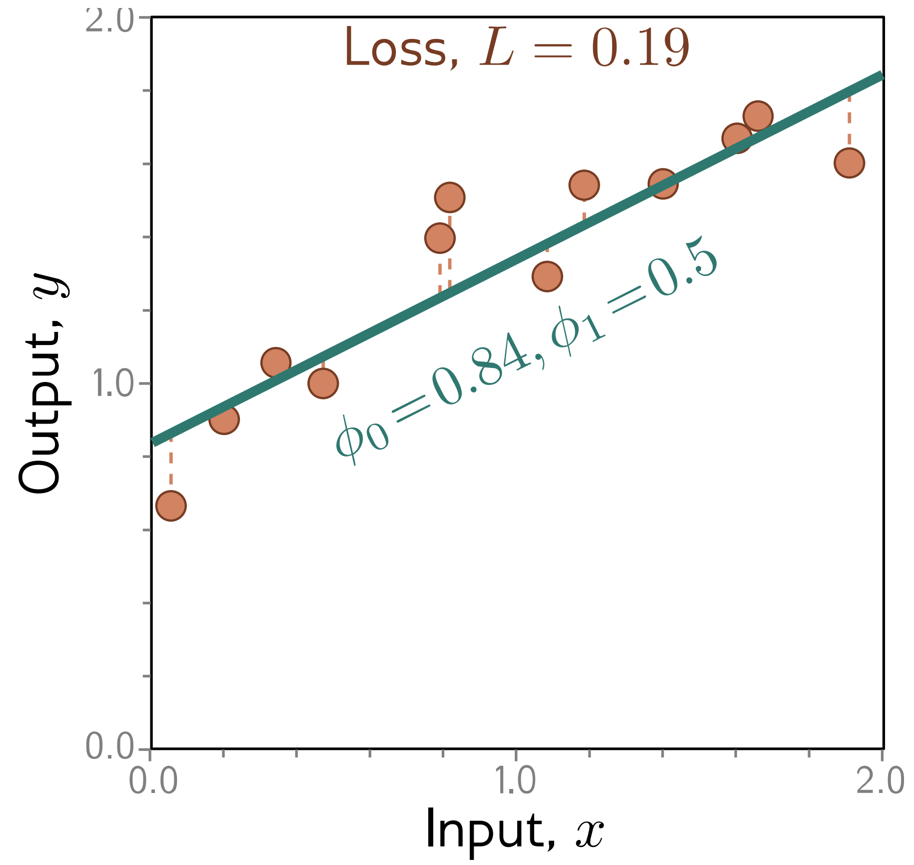
Loss, $L[\phi]$
70
60
50
40
30
20
10
0
-1.0
0.0
1.0
Slope, ϕ_1
0.0
1.0
2.0
Intercept, ϕ_0



Loss, $L = 10.22$
$\phi_0 = 1.60, \phi_1 = -0.8$
2.0
1.0
0.0
Output, y
0.0
1.0
2.0
Input, x

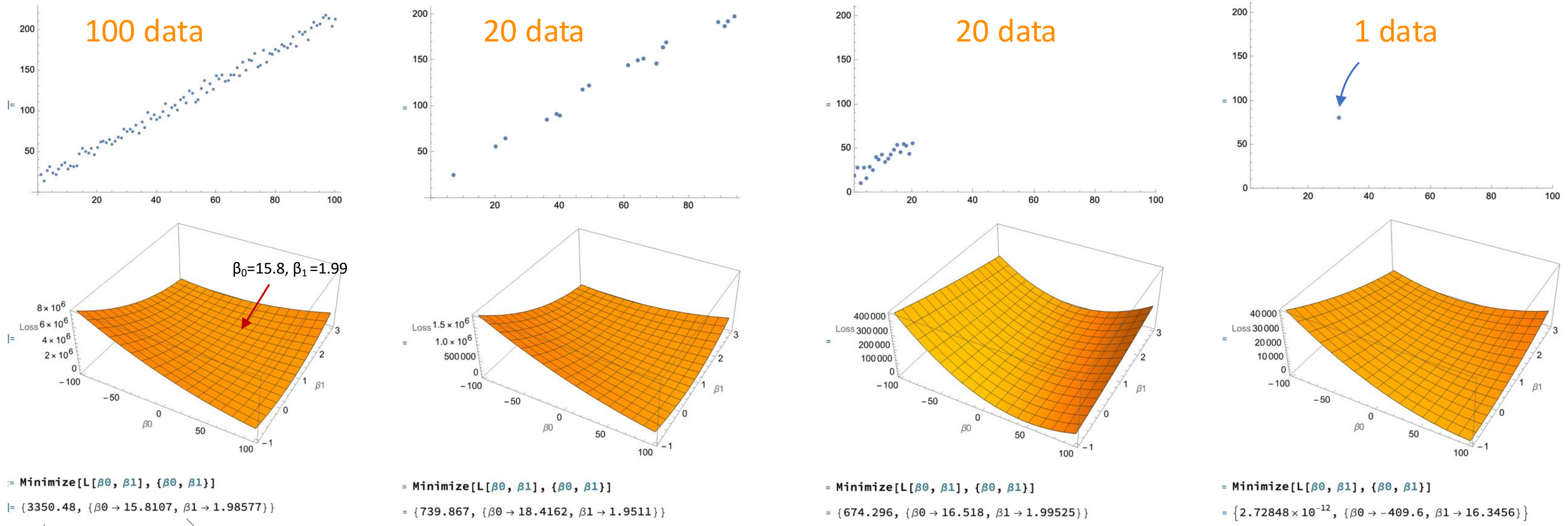
Linear Regression: 1D linear regression





Loss function depends on data and model chosen!

- For simplicity, consider simple linear regression $y=\beta_0+\beta_1 x$ and 100 data points. See below how loss fnc depends on data. (Data is sampled from $y= 15 + 2x$ + noise.)



loss at min

very close to $\beta_0=15$ and $\beta_1=2$ (coefficients that are used to generate the data)

Worst (β_0,β_1) values.

Loss function depends on data and model chosen!

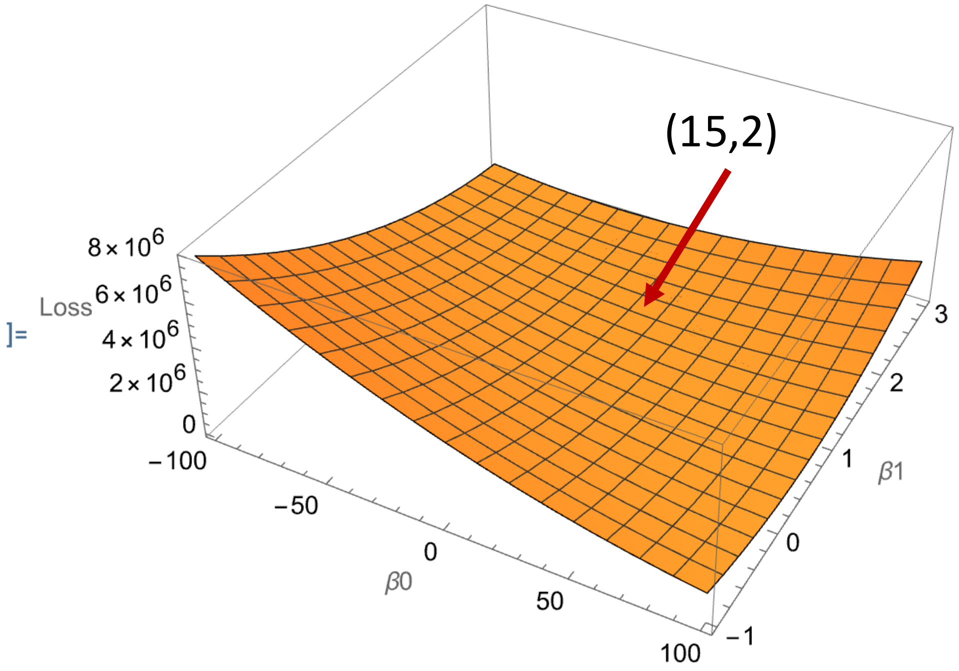
- You need to choose a good loss function. If you're doing linear regression, most use loss function is Mean Square Error (MSE).
- $(\beta_0, \beta_1)=(15,2)$ is the correct minimum for data for MSE.

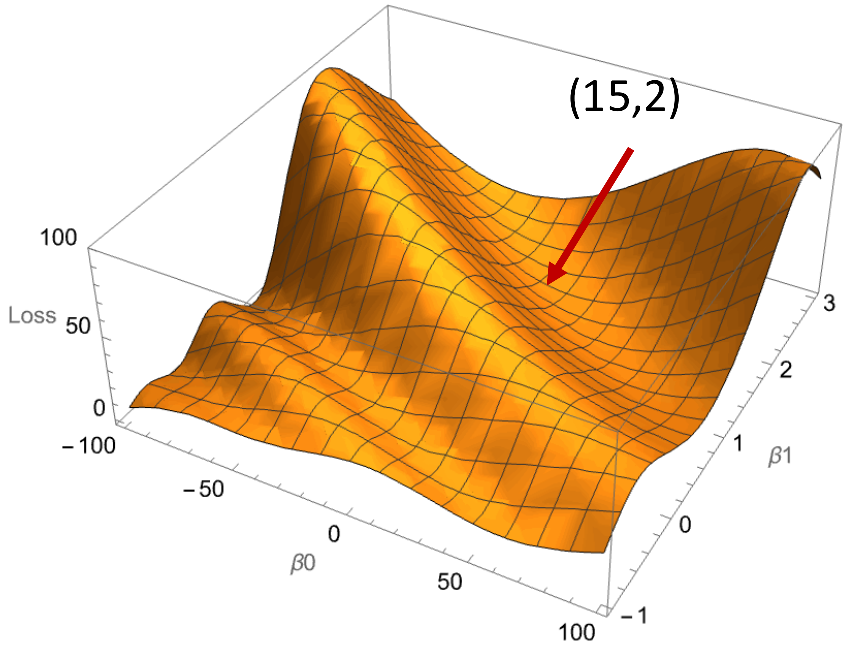
Mean squared error (MSE) loss fnc.

$$L[\phi] = \sum_{i=1}^{I} (f[x_i, \phi] - y_i)^2 = \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$$

$$L[\phi] = \sum_{i=1}^{I} \sin[\frac{1}{10000}(\phi_0 + \phi_1 x_i - y_i)^2]$$

Some unusual loss fnc I made up.

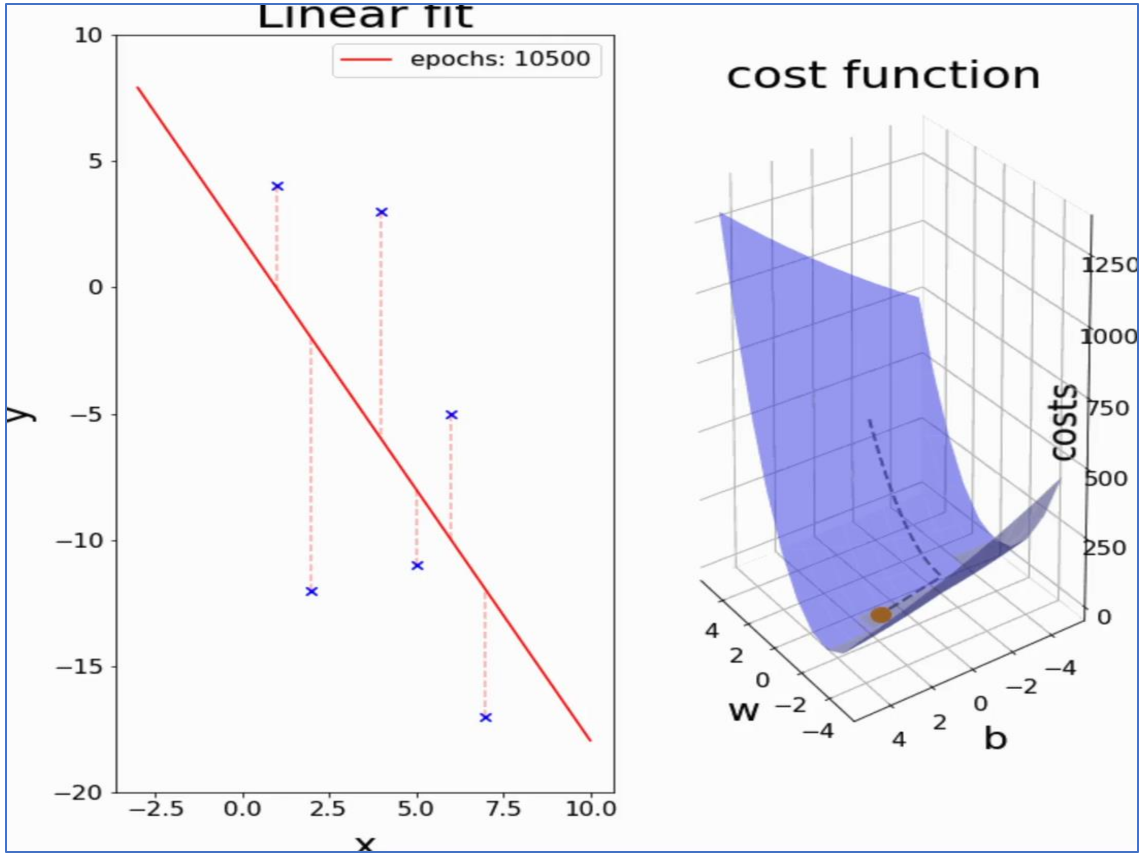




ML with gradient descend

For 1D linear regression
Model: *y=b + w x*
Data: *(x,y) pairs*
Unknown: *model coefficients/parameters b and w.*
For linear regression, b: intercept, w: slope.

- The process of using the data to get the fitted model is machine learning. There are special algorithms such as gradient descend that take data and output the fitted/trained model.
- We use simple linear regression as an example. The purpose is finding the line where the total error (average of squared vertical distances between the line and data points) is minimized.
- The "*error*" or "*loss*" or "*cost*" is a function that depends on coefficients (*b*,*w*) of the model *y=b + w x*. We're looking for the (*b*,*w*) pair that minimize *loss*[*b*,*w*]. (b: bias, w: weight)



https://towardsdatascience.com/gradient-descent-animation-1-simple-linear-regression-e49315b24672

ML with gradient descend

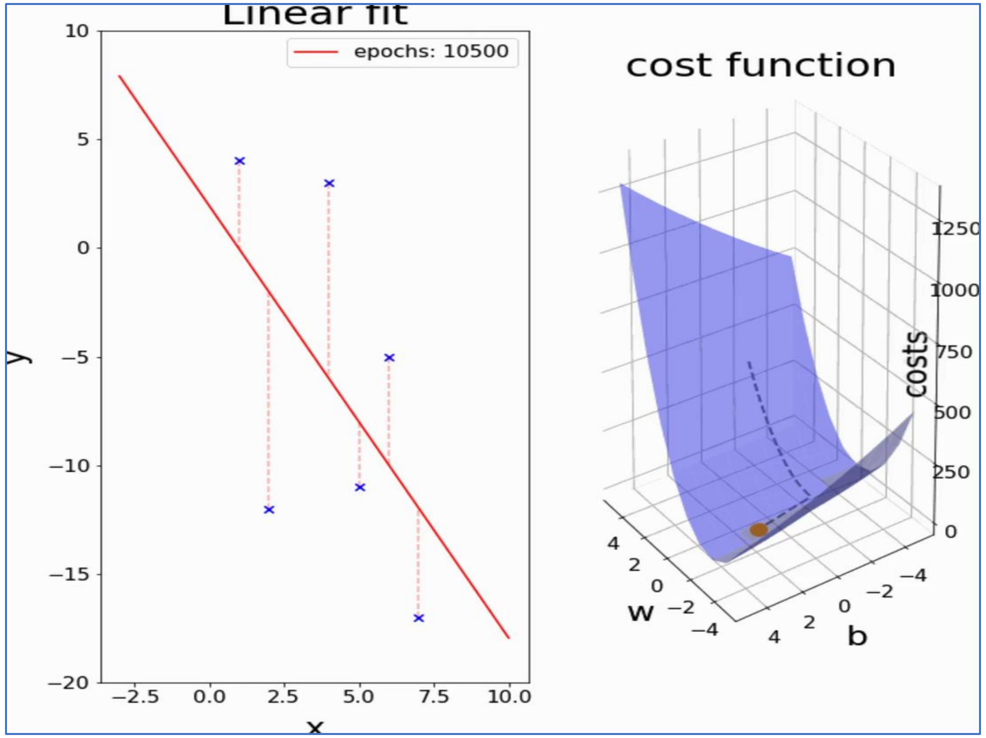
- Gradient descent algorithm (GD)

Let *L(w,b)* be the cost function "Mean Square Error" (MSE). When *L(w,b)* is minimum, we arrive at the best (*w*,*b*). Learning rate α is about 0.001. It determines step size in GD optimization algoritm.

iterate until *L*(*w*,*b*) is minimum

$$L_{new} = L_{old} - \alpha \frac{\partial L[w,b]}{\partial w}\bigg|_{w=w_{old}}$$

$$L_{new} = L_{old} - \alpha \frac{\partial L[w,b]}{\partial b}\bigg|_{b=b_{old}}$$



- 1 iteration = 1 epoch [using all data points to calculate the cost function *L*(*w*,*b*)]

Linear Regression

- **Linear regression:** "Observed data is the result of an underlying linear model and some error"
- **Simple Linear regression:** one independet variable x, hence two coefficients (β_0 ,β_1).

model

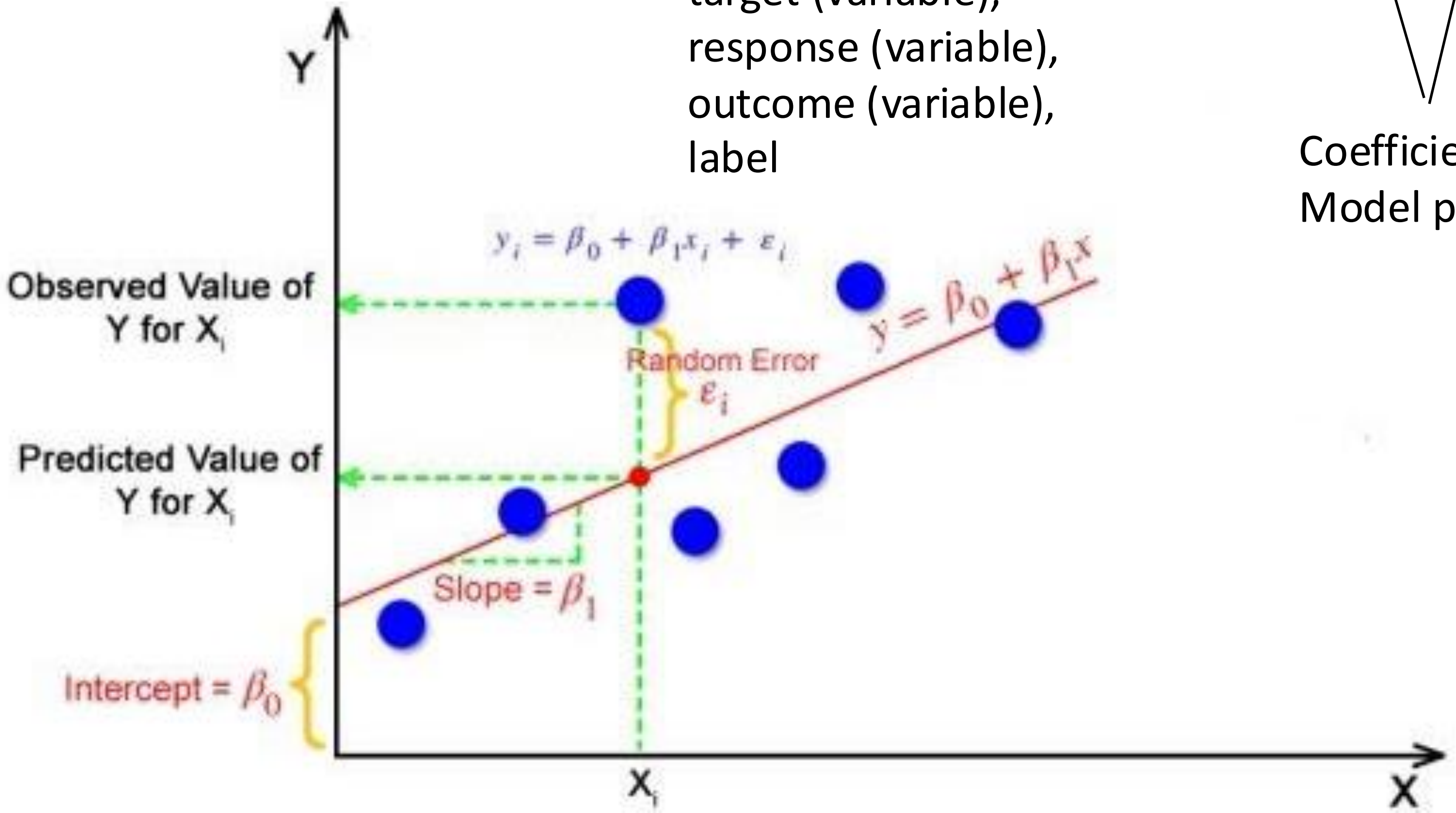
$$y_i = \boxed{\beta_0 + \beta_1 x_i} + \varepsilon_i$$

Dependent variable, predicted variable, target (variable), response (variable), outcome (variable), label

Coefficients, Model parameters

Independent variable, predictor, feature, attribute, input variable, explanotary variable, factor

Error term, residual



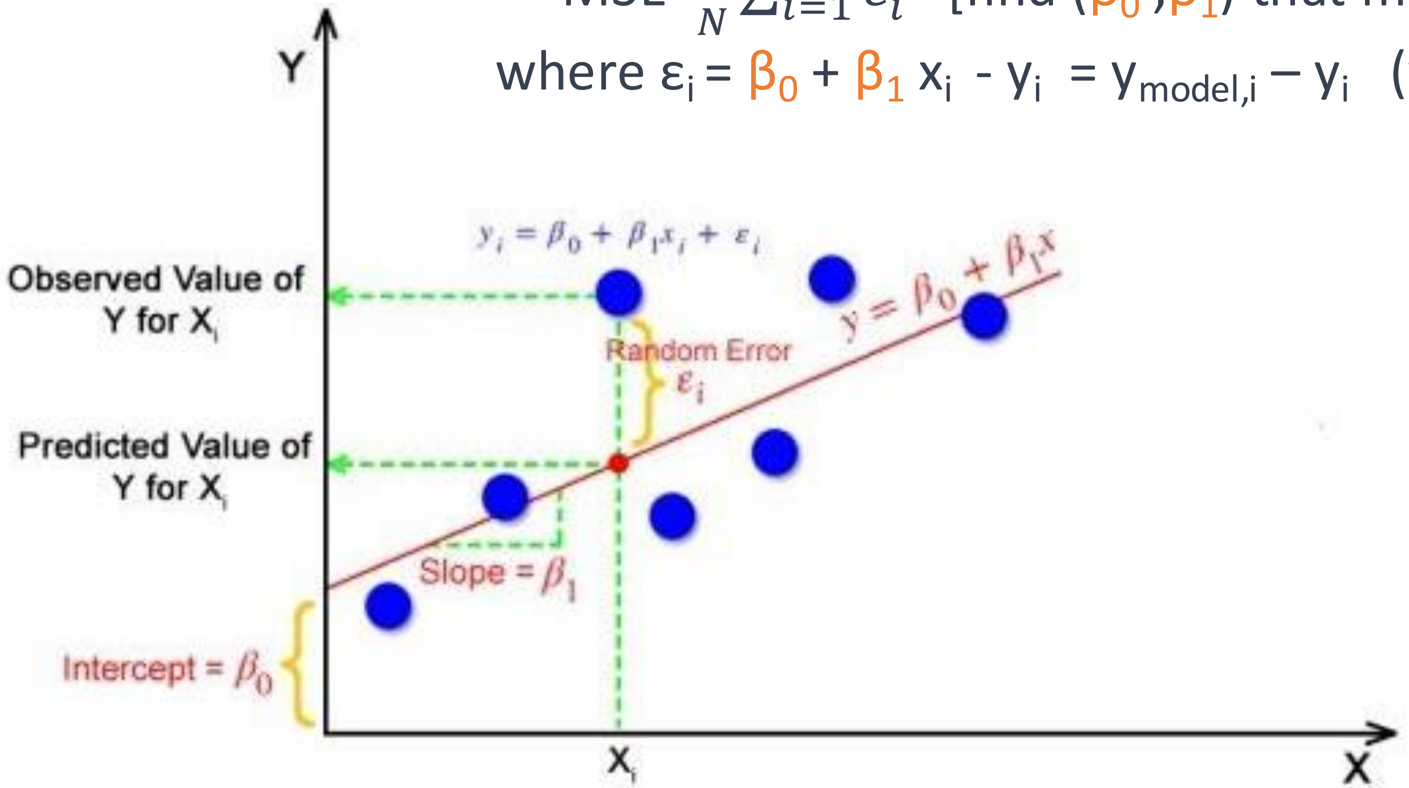
Linear Regression

Simple Linear regression: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

- The aim is to find the values of (β_0, β_1) that minimize mean of error values, which are the vertical distances between the observed value and predicted value.
- Most used cost function is mean square error.

$$\text{MSE}=\frac{1}{N}\sum_{i=1}^{N}\varepsilon_i^2 \quad \text{[find } (\beta_0, \beta_1) \text{ that minimizes MSE]}$$

where $\varepsilon_i = \beta_0 + \beta_1 x_i - y_i = y_{model,i} - y_i$ (y_i here is target).



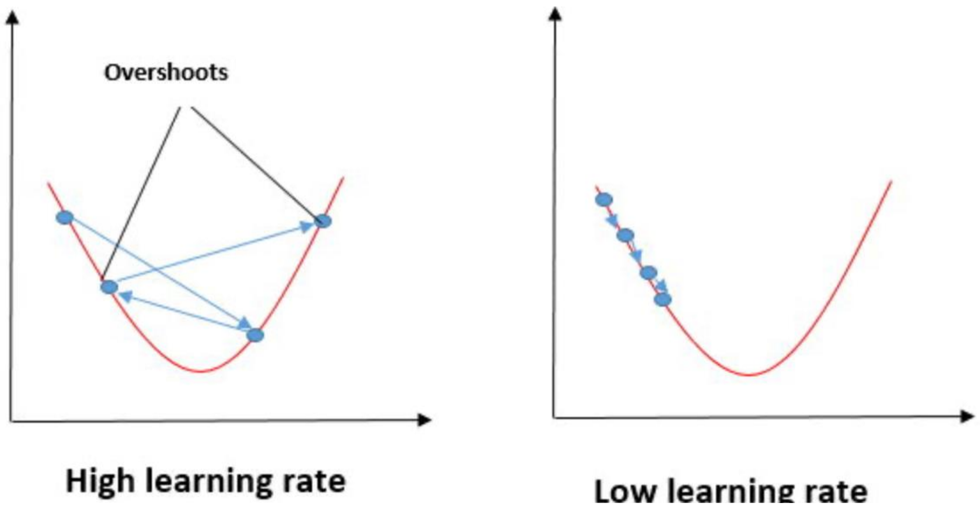
Linear Regression

Simple Linear regression: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

- But how do we minimize the cost function? There are two primary methods.

1. Gradient descent (Numerical solution): This gives approximate values of (β_0, β_1).
2. Ordinary least squares (OLS) (Analytical solution): This gives the exact values of (β_0, β_1).

1. Gradient descent: Iteratively tries to find the values (β_0, β_1) that lead to min(MSE). This optimization process (not the model itself) has one hyperparameter called learning rate α. You need to try different values to choose a good learning rate so (i) you can find the minimum of the cost function and (ii) you can find it in the least steps/time (there's a trade-off between these two).



Linear Regression

Simple Linear regression: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

2. Ordinary least squares (OLS): One can find analytically (=doing math by using pen and paper) the coeffients that minimize the summation of least squares $\sum_{i=1}^{N} \varepsilon_i^2$. "Squares" refers to the error terms (residuals) ε_i^2. "Least" refers to the minimization their summation. "Ordinary" refers to the fact that we're not adding any term to $\sum_{i=1}^{N} \varepsilon_i^2$; later we'll add other terms to this expression, and it'll not be called "ordinary" in those cases. Some math gives the analytical solution for the coefficients:

slope

$$\beta_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Intercept

$$\beta_0 = \bar{y} - \beta_1 \bar{x}$$

Example:

$\bar{x} = 3, \bar{y} = 65$

$\beta_1 = \frac{\sum(x_i - 3)(y_i - 65)}{\sum(x_i - 3)^2}$ =7

$\beta_0 = \bar{y} - \beta_1 \bar{x}$ =44

Hours of Study (x)	Test Score (y)
1	50
2	60
3	65
4	70
5	80

regression line: y= 44 + 7x

Linear Regression

Simple Linear regression: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

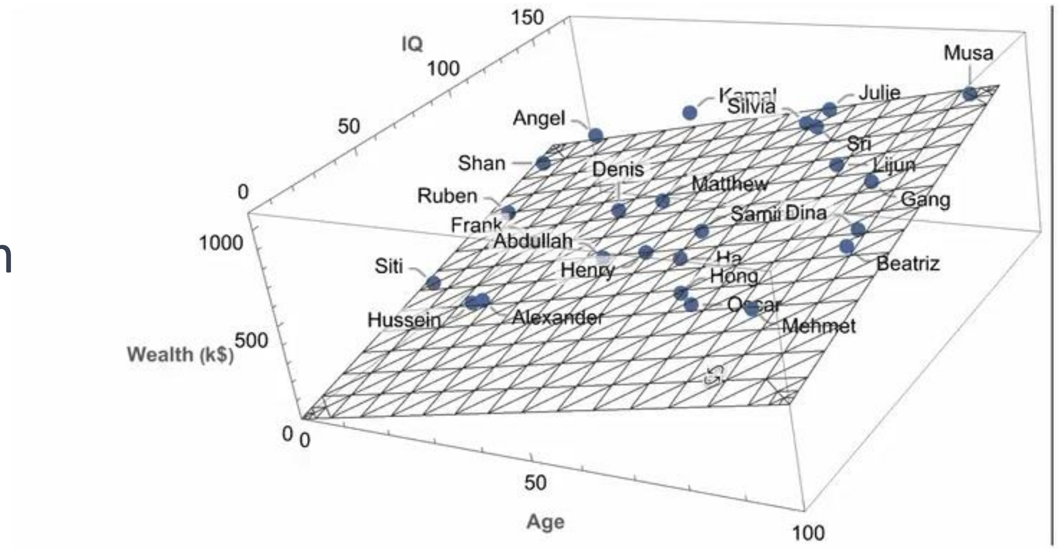
Multiple Linear regression: $y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{3,i} + \ldots + \varepsilon_i$

Regression with multiple predictors ($x_1, x_2, \ldots$). Use this when you have more than one features in the data.

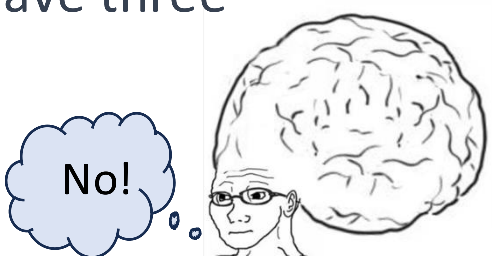
Let's consider two predictors case for simplicity:

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$.

The model in this case is a 2D plane where the sum of squared errors between the data points and the plane is minimum.



What would the model be like if we have three predictors? Can you visualize?



Linear Regression

Simple Linear regression: $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$

Multiple Linear regression: $y_i = \beta_0 + \beta_1 x_{1,i} + \beta_2 x_{2,i} + \beta_3 x_{3,i} + \ldots + \varepsilon_i$

Ok, what's the formula for the predictors in this case?

$$\beta = (X^T X)^{-1} X^T Y$$

Example:

TV ads (x1)	Radio ads (x2)	Sales (y)
1	1	6
2	1	8
3	2	11
4	2	13

Coeffients vector:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix}$$

Design matrix:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 3 & 2 \\ 1 & 4 & 2 \end{bmatrix}$$

First column is always all ones.

Target vector:

$$Y = \begin{bmatrix} 6 \\ 8 \\ 11 \\ 13 \end{bmatrix}$$

```python
import numpy as np

# Given data
X_data = np.array([
    [1, 1],
    [2, 1],
    [3, 2],
    [4, 2]
])
Y_data = np.array([
    [6],
    [8],
    [11],
    [13]
])

# Add a column of ones to X_data for the intercept
X = np.hstack([np.ones((X_data.shape[0], 1)), X_data])

# Calculate the coefficients using the formula: beta = (X^TX)^-1 X^TY
beta = np.linalg.inv(X.T @ X) @ X.T @ Y_data

print(beta)
```

```
[[3.]
 [2.]
 [1.]]
```

$\longrightarrow$ $\beta_0=3$, $\beta_1=2$, $\beta_2=1$

OLS vs Gradient Descent

$$\beta = (X^T X)^{-1} X^T Y$$

- In OLS formula, there is a matrix inversion term; if the data dimensionality (columns) and amount of records (rows) are large, you'll need to invert a large matrix. Inversion of large matrices are hard for computers. So, although OLS gives the exact model parameters **β**, it may be slow for large datasets. In that case, use the numerical method gradient descent.
- In the case online learning, data may be streaming in continuously, and you might want to update the model parameters on-the-fly. OLS is not appropriate in this scenario because after each streaming data point second you'll need to build the entire design matrix *X* and use the matrix formula repeatedly. Gradient descent is better suited here.
- Especially when you want to avoid overfitting, you add regularization terms to the cost function such as $\sum_{i=1}^{N} \varepsilon_i^2 + \lambda \sum_{i=1}^{p} \beta_i^k$ (not "ordinary" anymore) where *p* is number of predictors and *k* is some number. For these types of cost functions, the closed form matrix solution for **β** may not be known. Therefore in this case use gradient descent.

Linear Regression

Polynomial regression: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2 + \ldots + \varepsilon_i$

- Wait, how can this be linear regression and have nonlinear polynomial terms such as x_1^2?

This is a linear regression; "linear" here refers to the linearity of the coefficients **β** in the sense that we have coefficients appearing with power of 1 and not multiplied with each other.

Examples:

$y = \beta_0 + \beta_1 x^2$ (linear regression)

$y = \beta_0 + x^{\beta_1}$ (not linear regression)

these break linearity of coefficients

$y = \beta_0 + e^{\beta_1 x}$ (not linear regression because $e^{\beta_1 x} = 1 + \beta_1 x + \frac{\beta_1^2 x^2}{2!} + \frac{\beta_1^3 x^3}{2!} + \cdots$)

$y = \beta_0 + \beta_1 e^{x}$ (linear regression)

$y = \beta_0 + \beta_1 e^{\sin(x)+\log(x^2)}$ (linear regression)

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$ (linear regression)

- Let's do an example for the model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_2$ where x_1 is polynomial here.

Linear Regression

Polynomial regression: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \ldots + \varepsilon_i$

Example: Fit the data below to the model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2$

Example:

TV ads (x1)	Radio ads (x2)	Sales (y)
1	1	6
2	1,5	8
3	2	11
4	2	13

Coeffients vector:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$

Design matrix:

$$X = \begin{array}{c} \begin{matrix} & x_1 & x_2 & x_1^2 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 1.5 & 4 \\ 1 & 3 & 2 & 9 \\ 1 & 4 & 2 & 16 \end{bmatrix} \end{array}$$

First column is always all ones.

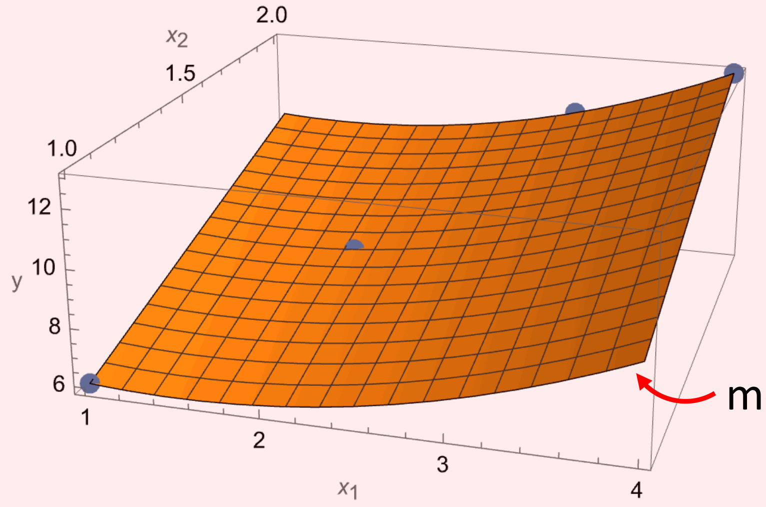
Target vector:

$$Y = \begin{bmatrix} 6 \\ 8 \\ 11 \\ 13 \end{bmatrix}$$

Again use:

$$\beta = (X^T X)^{-1} X^T Y$$

Result: **β**=(3, -1.5, 0.5, 4)

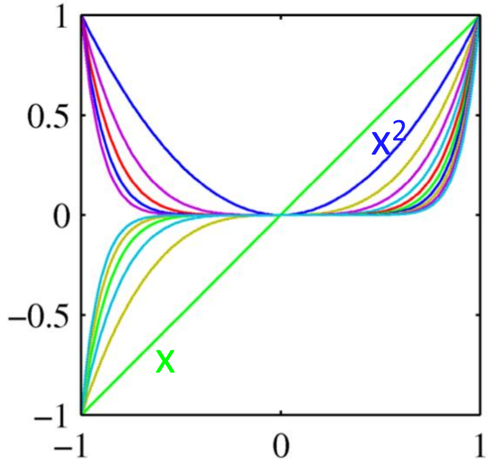


model

Basis functions

- **Polynomial basis** function are the easiest: x, x^2, x^3, ... And the coefficients β are the weights that determine how much we include in our model from each polynomial degree.
- But in linear regression, we're not restricted to use them. We can use any basis functions $\phi_j(x)$ so our model becomes $y = \beta_0 + \beta_1 \phi_1(x) + \beta_2 \phi_2(x) + \ldots$
- Below, different colors correspond to different values of the means μ_j

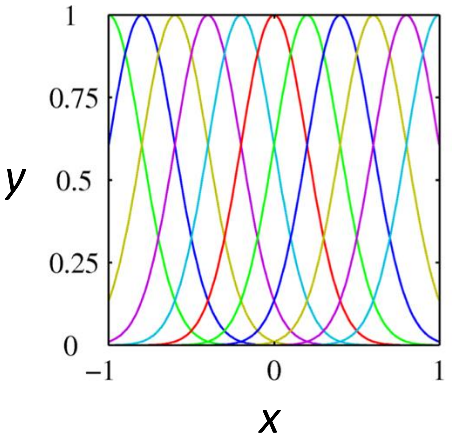
$$\phi_j(x) = x^j$$



- **Polynomial basis** are not local, meaning each term $\phi_j(x)$ contributes to y along the whole x axis. But **Gaussian** and **Sigmoid** bases are "local", meaning each term contributes to y only in the vicinity of their mean μ_j.
- How many basis can I take? What should be the separation between each μ_j? *You* should determine them; these are all depends on the data and underlying dynamics of it.

Gaussian basis functions

$$\phi_j(x) = \exp\left[-\frac{(x-\mu_j)^2}{2\sigma^2}\right]$$



Sigmoid basis functions

$$\phi_j(x) = \frac{1}{1+\exp(\frac{x-\mu_j}{\sigma})}$$