Artificial Intelligence and Machine Learning

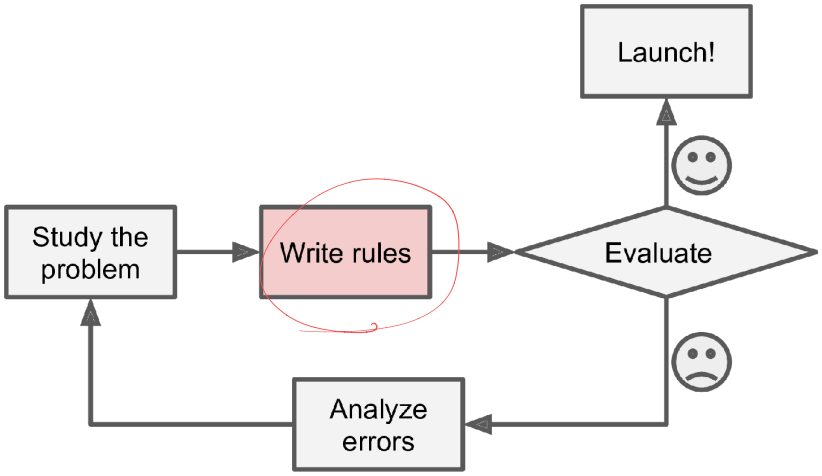
Linear Regression

Lecture 1: Outline

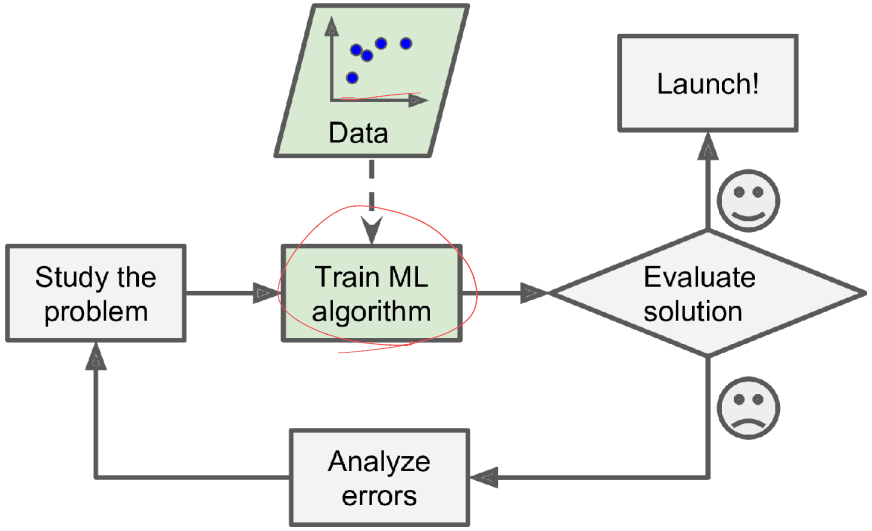
- Introduction to ML
- Linear Regression
- Optimization
- Linear Regression: Probabilistic Interpretation
- Bias-Variance Tradeoff
- Regularization

Introduction to ML

- **Machine Learning** is the science (and art) of programming computers so they can learn from data.

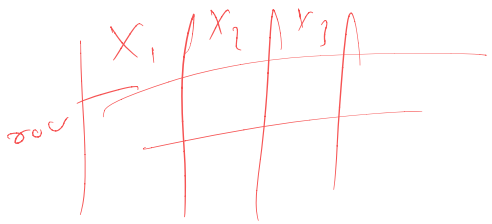


The traditional approach



The Machine Learning approach

Data Types

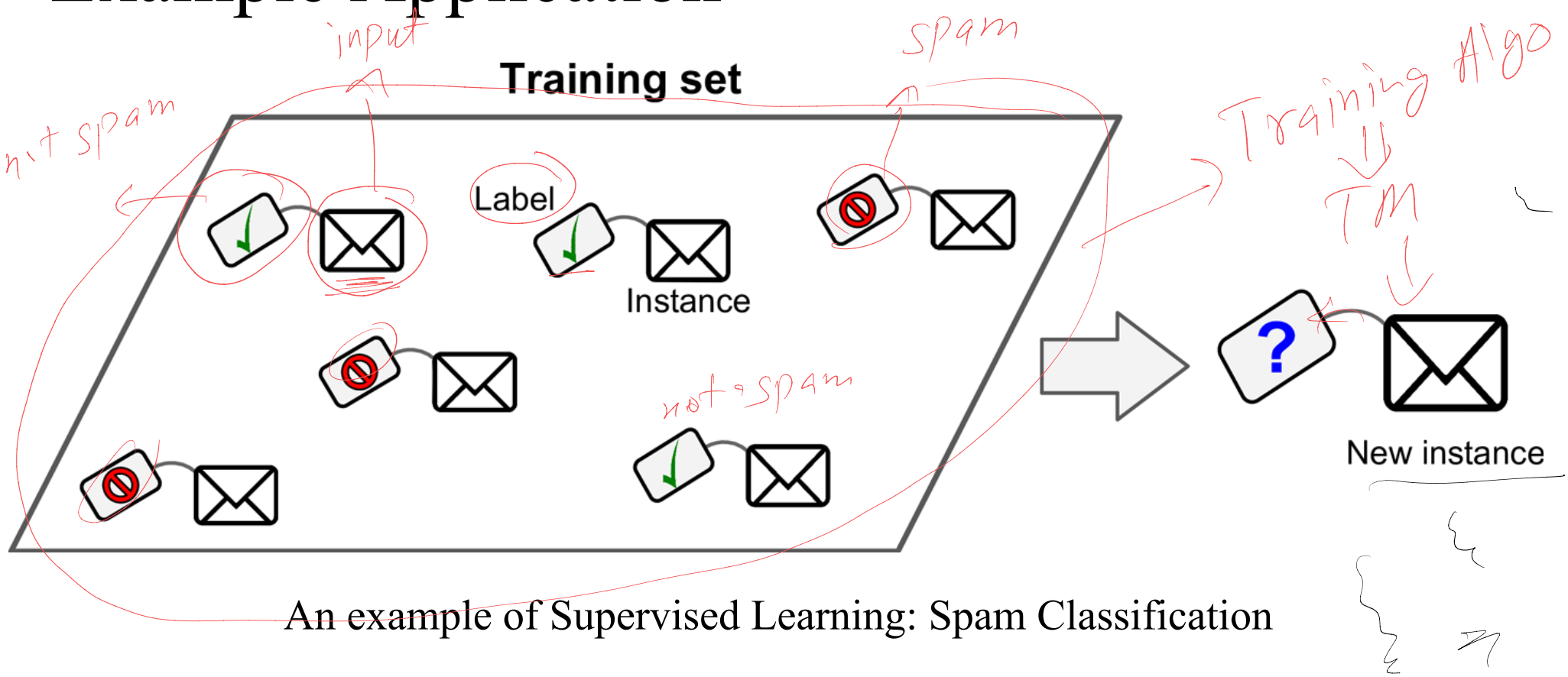


- **Tabular Data** (e.g., spreadsheets, databases)
  - Note: Columns are called **Features**. Rows are called **Samples.**
- **Time-Series Data** (e.g., stock prices, weather forecasts, IoT sensor data)
- **Text Data** (Natural Language Processing, e.g., emails, social media posts, documents)
- **Images and Videos** (Computer Vision, e.g., medical imaging, surveillance, facial recognition)
- **Audio Data** (Speech Recognition, Music Processing, e.g., voice commands, podcasts, sound classification)

ML Algorithms Types

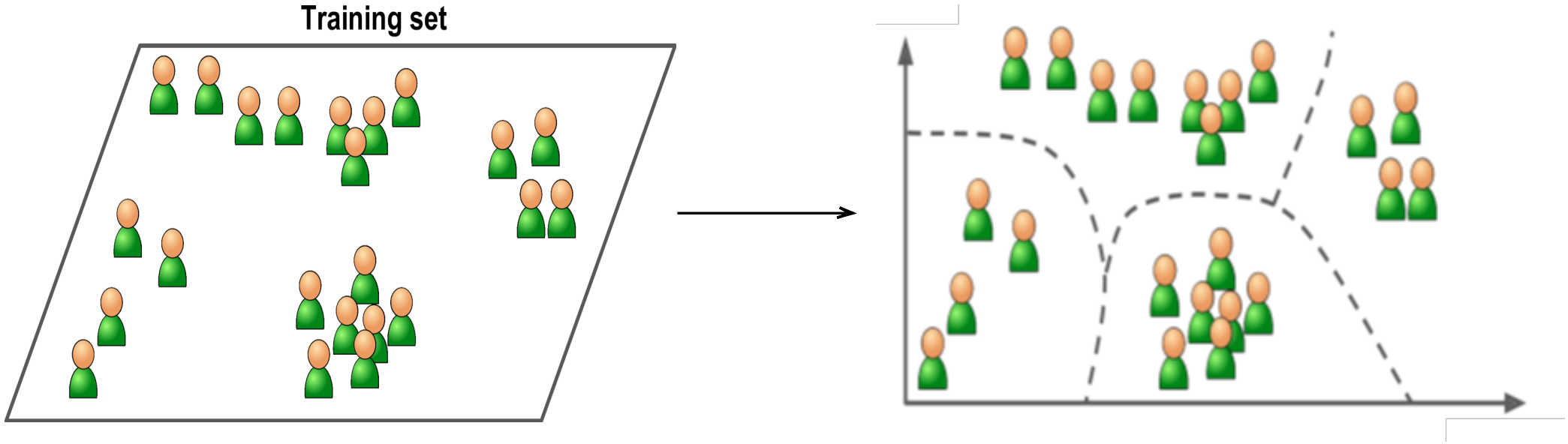
- **Supervised**: There is a target we want to predict.
  - **Regression**: Predict continuous value (e.g. house prices).
  - **Classification**: Predict discrete value (e.g. spam/not-spam).
- **Unsupervised**: There is no target. We are interested in things like:
  - **Clustering**: Grouping
  - **Dimensionality Reduction**: Reducing the Dimensions
  - **Anomaly Detection**: Detecting outliers

Example Application



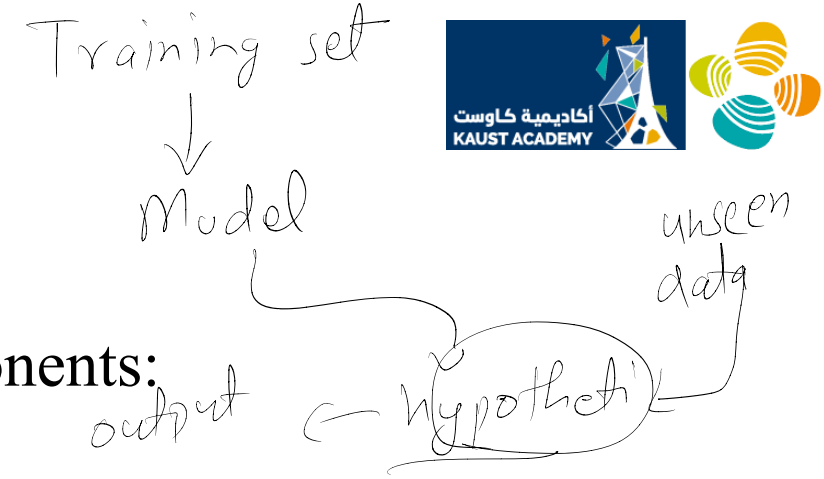
An example of Supervised Learning: Spam Classification

Example Application



An example of Unsupervised Learning: Clustering

How Does ML Work?



- Any ML system consists of three main components:

Hypothesis (Model): The function that approximates the target.

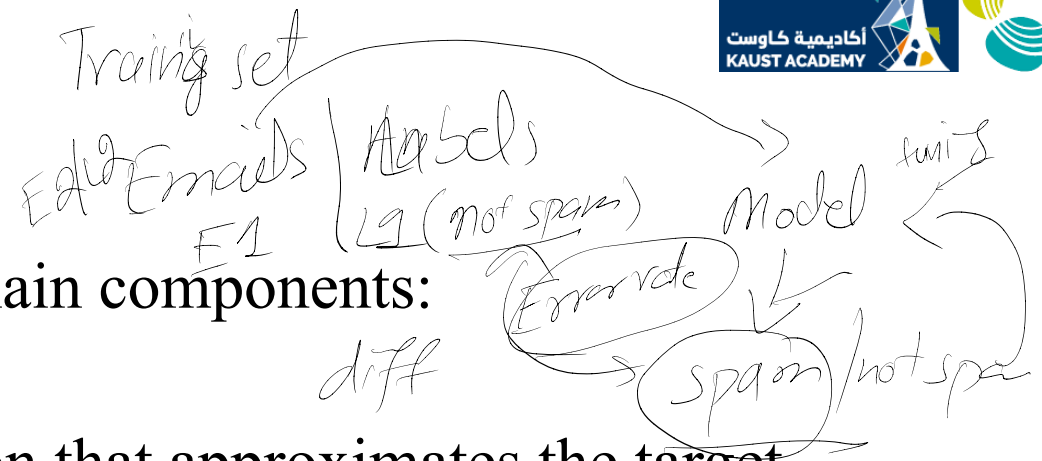
- E.g. Linear Regression, Logistic Regression, SVM, Decision Trees, NN,…

Optimizer: The mechanism for improving predictions of our model.

Loss Function: The measure of how wrong the predictions are.

objective
functio

How Does ML Work?



- Any ML system consists of three main components:

Hypothesis (Model): The function that approximates the target.

- E.g. Linear Regression, Logistic Regression, SVM, Decision Trees, NN,…

Optimizer: The mechanism for improving predictions of our model.

Loss Function: The measure of how wrong the predictions are.

How Does ML Work?

- We firstly define our task (classification/regression) then choose an appropriate **model.**
- We will use an **optimization method** to minimize the **loss function**.
- Reached a minima?

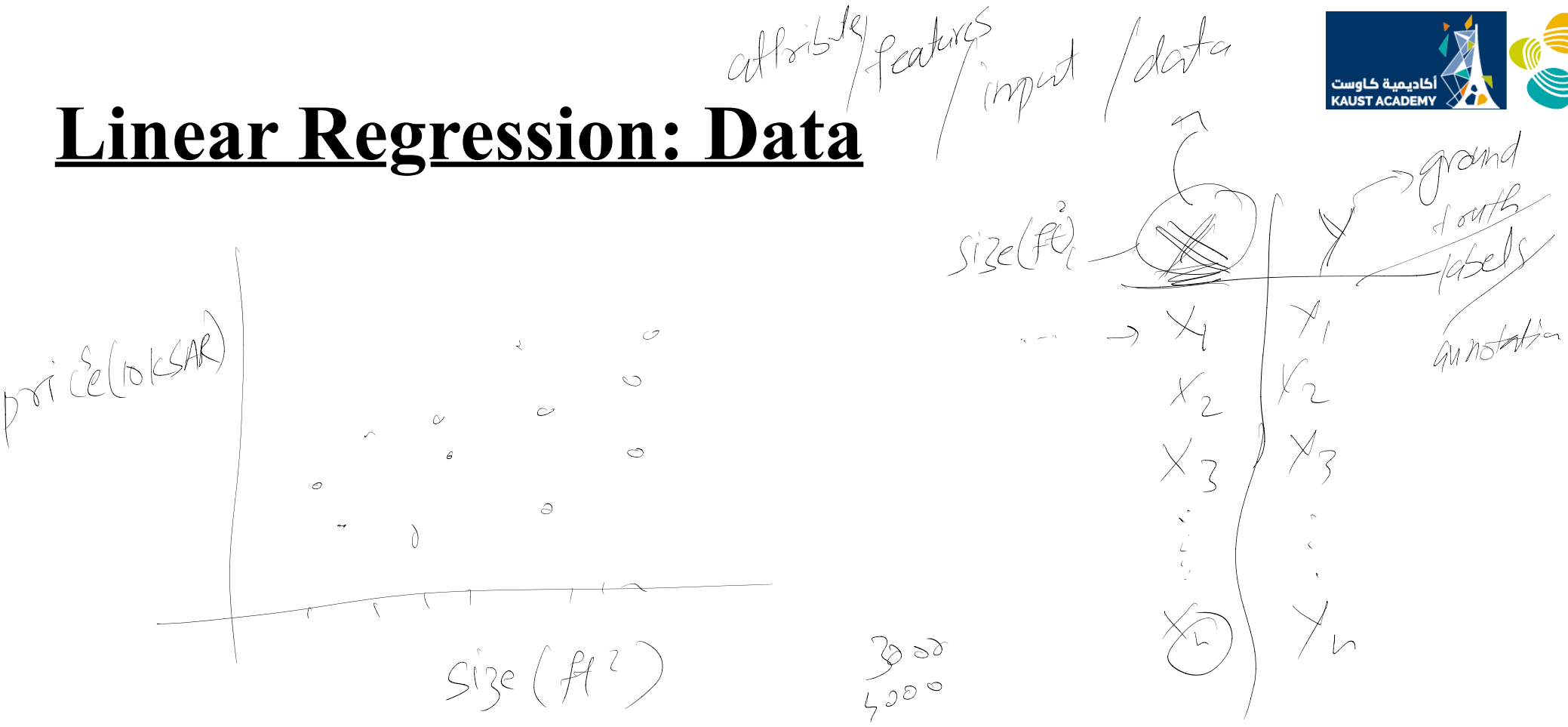
  = Model is making the least possible number of mistakes.

  = **Model trained** 🎉.

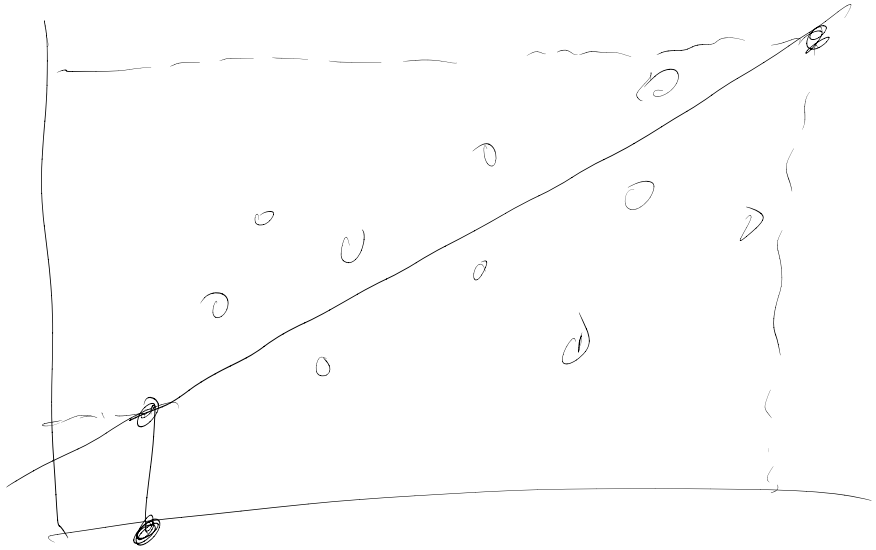
Linear Regression: Motivation

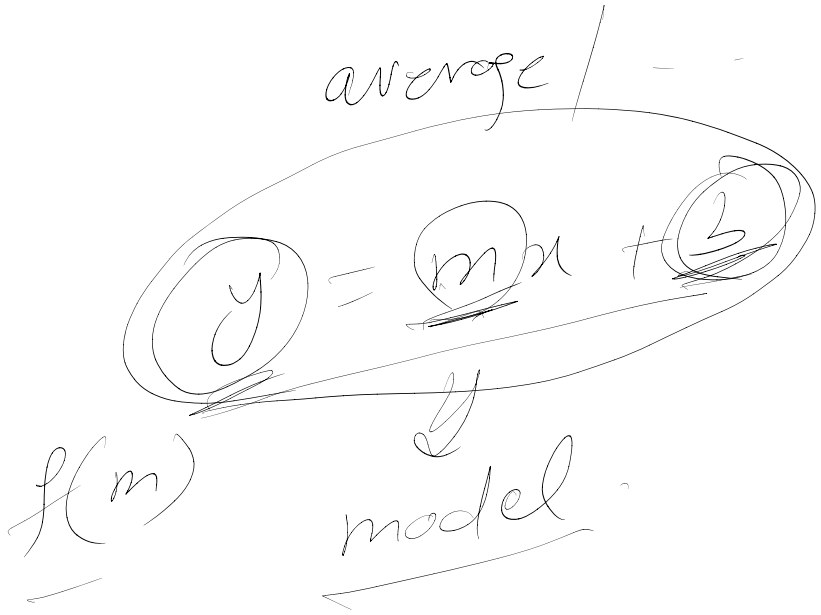
- Linear Regression is “still” one of the more widely used ML/DL Algorithms
- Easy to understand and implement
- Efficient to Solve
- We will use Linear Regression to Understand the concepts of:
  - Data
  - Models
  - Loss
  - Optimization

Linear Regression: Data



Linear Regression: Model





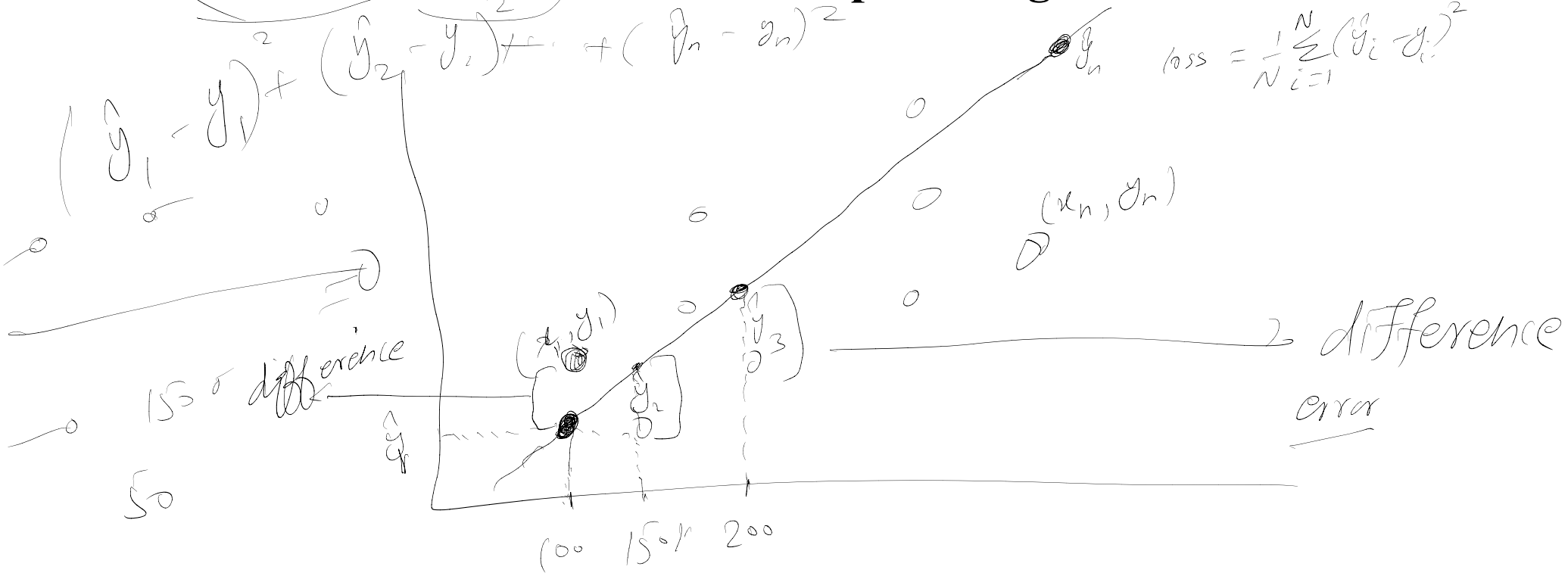
Linear Regression: Model

- **The linear regression model is function of … ?** → m & b
- **What are constant & variables**

Linear Regression: Loss

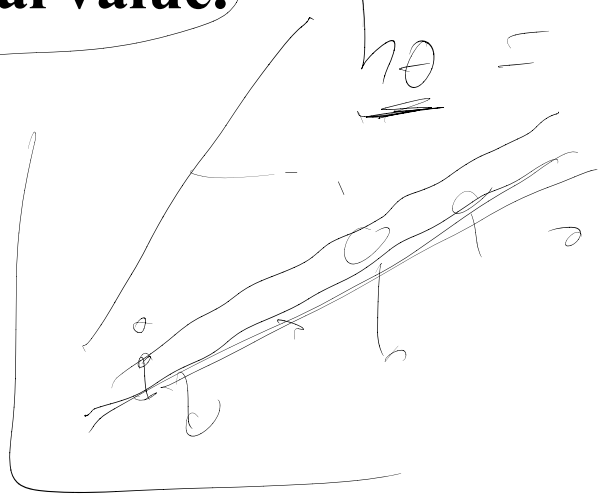
Objective

- The difference/distance between output and ground truth values.



Linear Regression: Optimization

- **Choose parameter θ such that the model prediction is equal/near to the actual value.**



$y = mx + b$

$0.3x + 0.5$

$0.2x + 0.4$

Simple Linear Regression

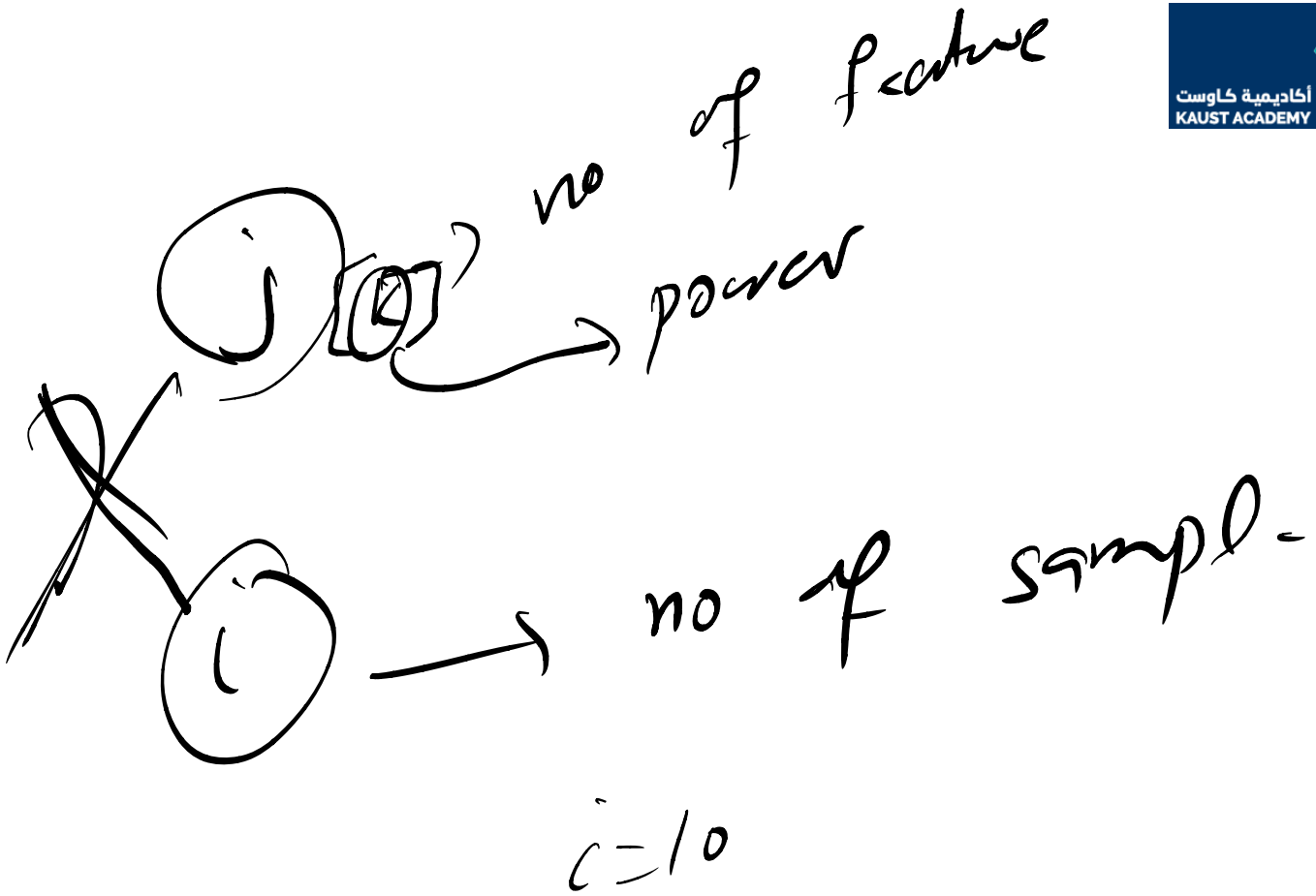
- **Hypothesis**:

$$\hat{y}_i = mx_i + b$$

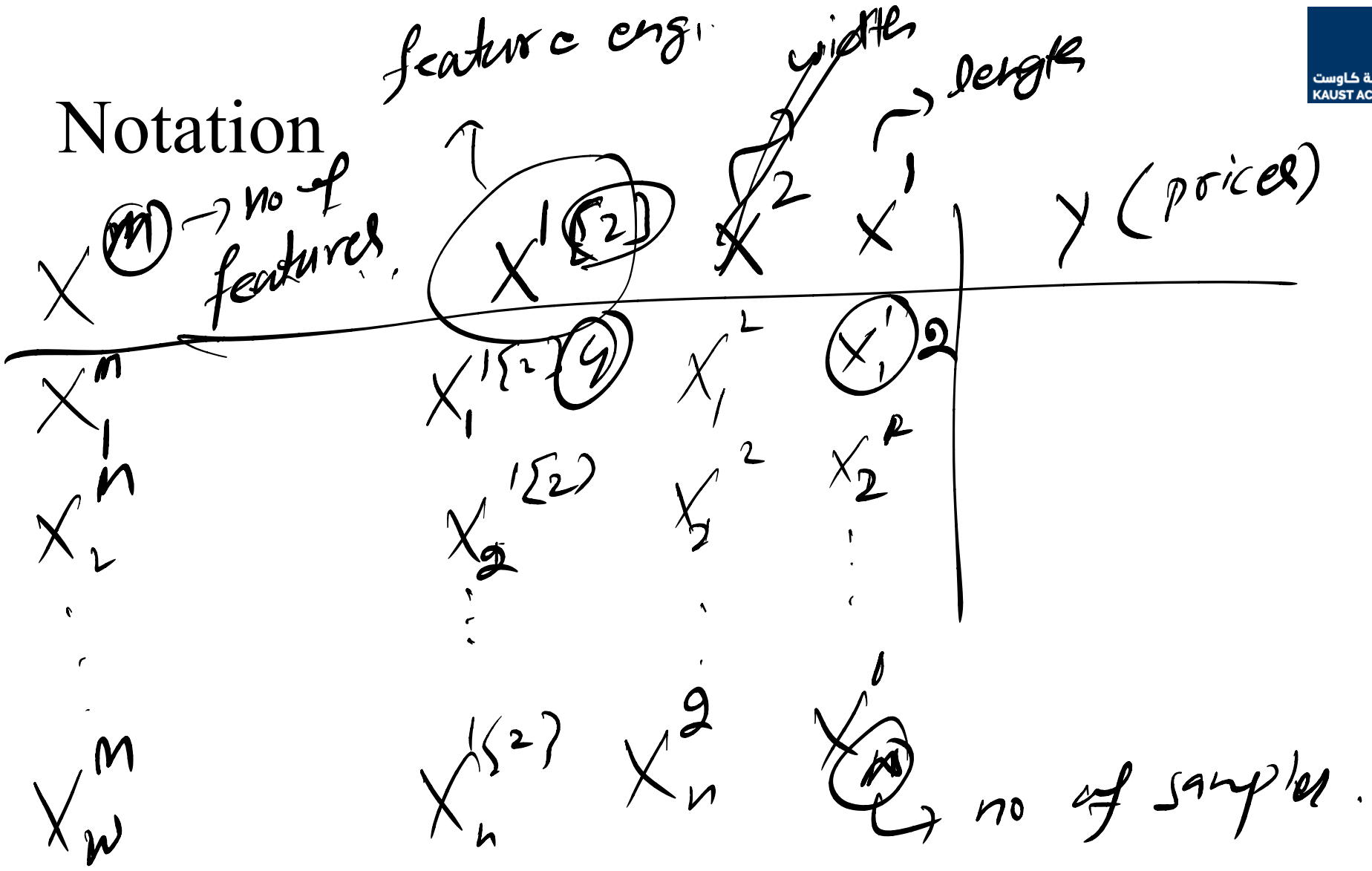
- **Input**: data (x_i, y_i), $i \in \{1,2, \dots, N\}$
  - (e.g., house size x and price y)

- **Goal**: learn values of variable (m, b)

Notation

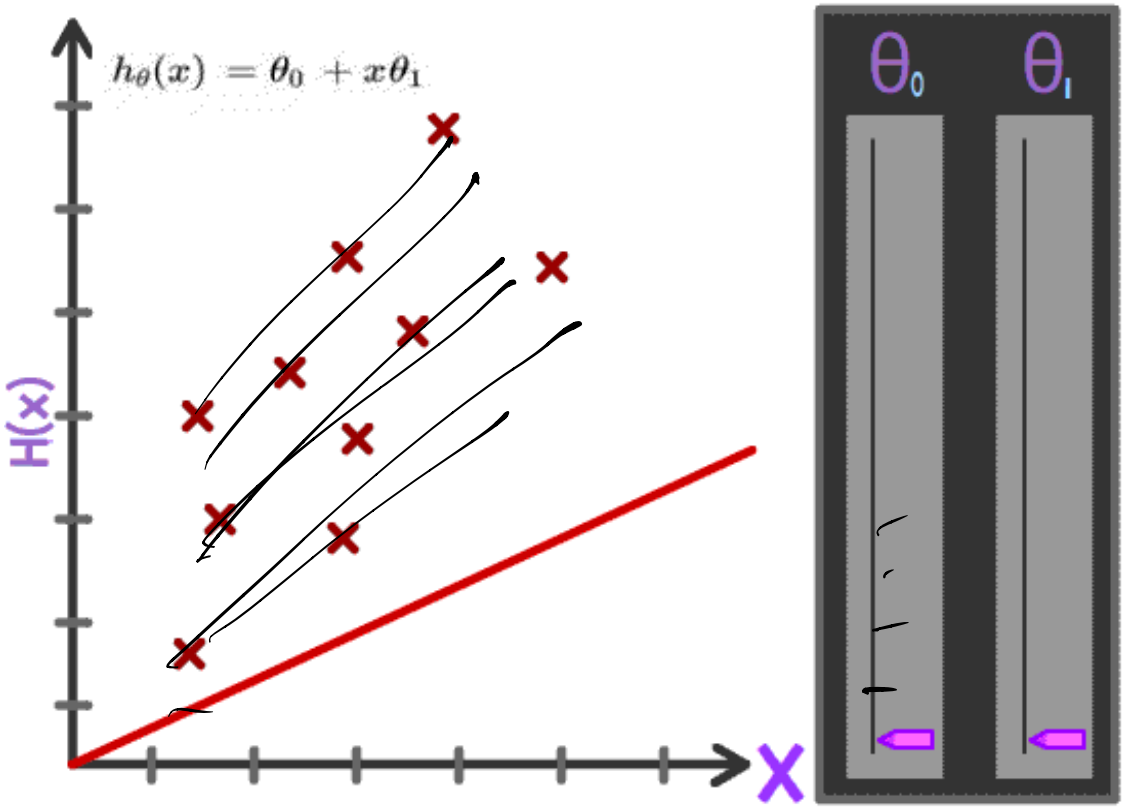


Notation


feature eng.
width
height
$x^{(m)}$ → no of features.
x^{1} x^{2} x^{1} x^{1}
y (price)
x_1^m $x_1^{1(2)}$ x_1^2 x_1^2
x_2^m $x_2^{1(2)}$ x_2^2 x_2^2
x_n^m $x_n^{1(2)}$ x_n^2 x_n^1
→ no of samples.

Solution Strategy for Solving the Problem

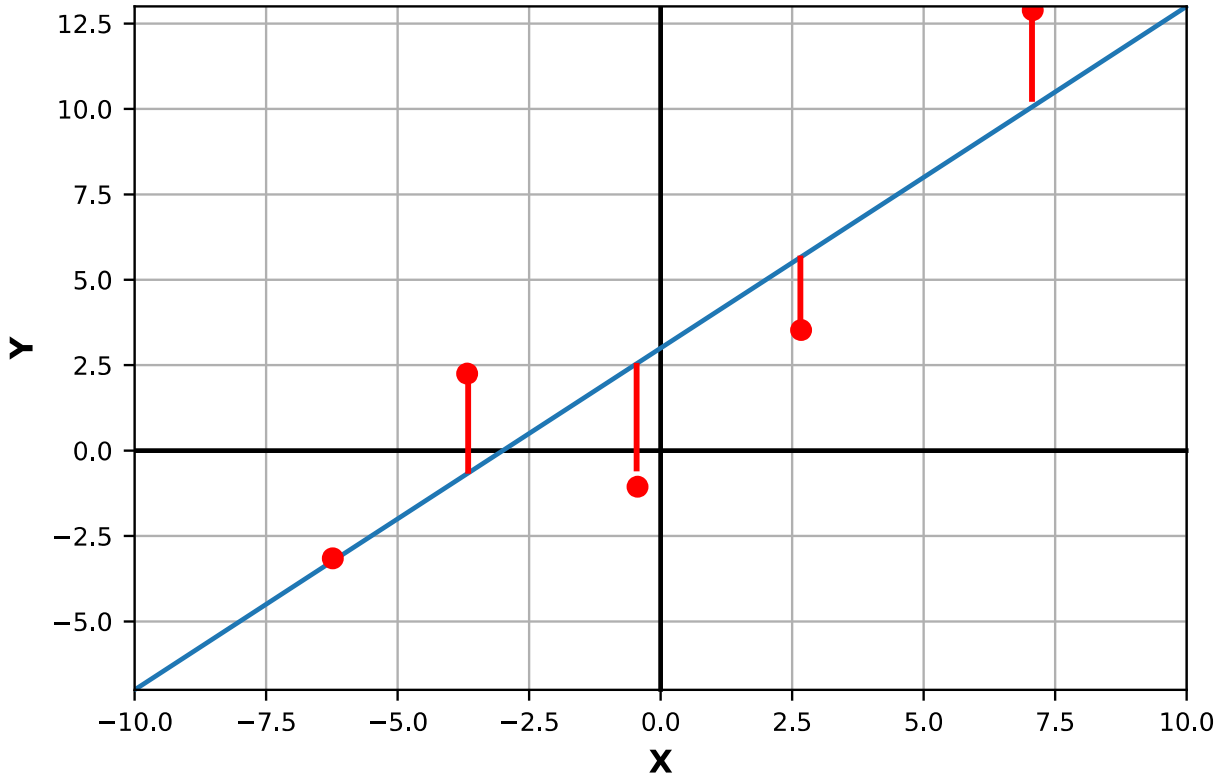
- There are countless possible lines.
- We want a line which is in some sense the "average line" that represents the data.
- Any ideas as to how we can do it?



https://medium.com/analytics-vidhya/machine-learning-intuition-b4b49a671f65

Optimization

- To find the "best line," we should minimize the **distances** between our line's predictions and all the data points.
- How to define that mathematically?



Loss Function

- For one sample, this can be represented mathematically by:

$$(y - \hat{y})$$ (Error)

- But this could result in negative value if $\hat{y} > y$. Let's square it to remove the negative sign:

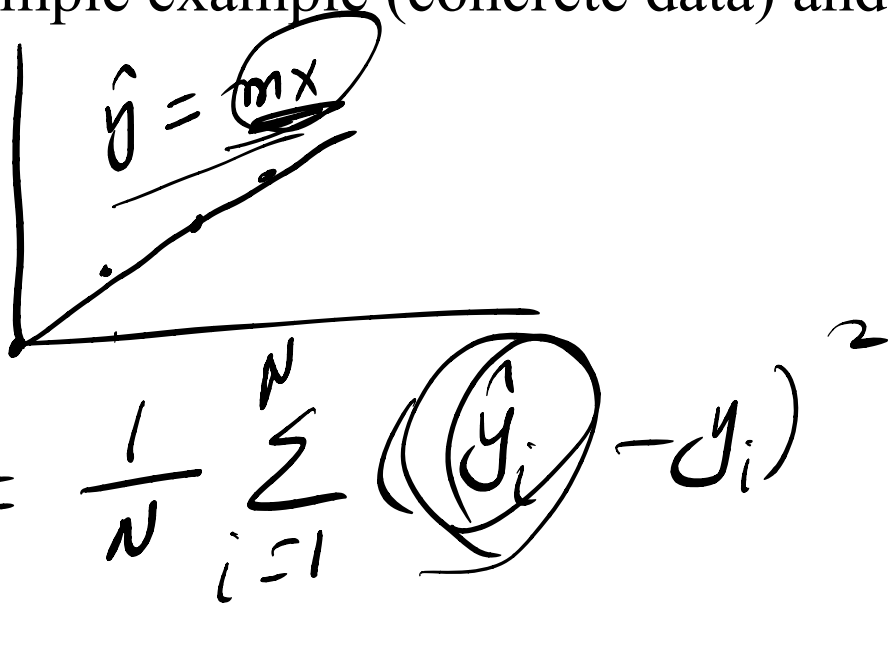
$$(y - \hat{y})^2$$ (Squared Error)

- But we have N samples, not only one. So, let's sum the errors and take the average:

$$Loss\ (MSE) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$ (Mean Squared Error)

Example Data and its Solution

- Let's take a simple example (concrete data) and solve it for *m*.



X	Y
0	0
1	1
2	2

Example Data and its Solution

- Let's take a simple example (concrete data) and solve it for *m*.

$$J(m) = \frac{1}{N}\sum_{i=1}^{N}(mx_i - y)^2$$

$$= \frac{1}{3}\left[(m\cdot 0 - 0)^2 + (m\cdot 1 - 1)^2 + (m\cdot 2 - 2)^2\right]$$

$$= \frac{1}{3}\left[(m-1)^2 + (2m-2)^2\right].$$

Example Data and its Solution

- Let's take a simple example (concrete data) and solve it for *m*.

$m = 0$

$$J(0) = \frac{1}{3}\left[(0-1)^2 + (2(0)-2)^2\right]$$

$$(y - \hat{y})^2$$

$$J(0) = \frac{1}{3}\left[1 + 4\right]$$

$$J(0) = \frac{5}{3}$$

Example Data and its Solution

- Let's take a simple example (concrete data) and solve it for *m*.

$m = 1$

$J(1) = \frac{1}{3}[0 + 0]$

$J(1) = 0$

$m_j := m_j - \alpha \frac{\partial}{\partial m}$

$\max J$

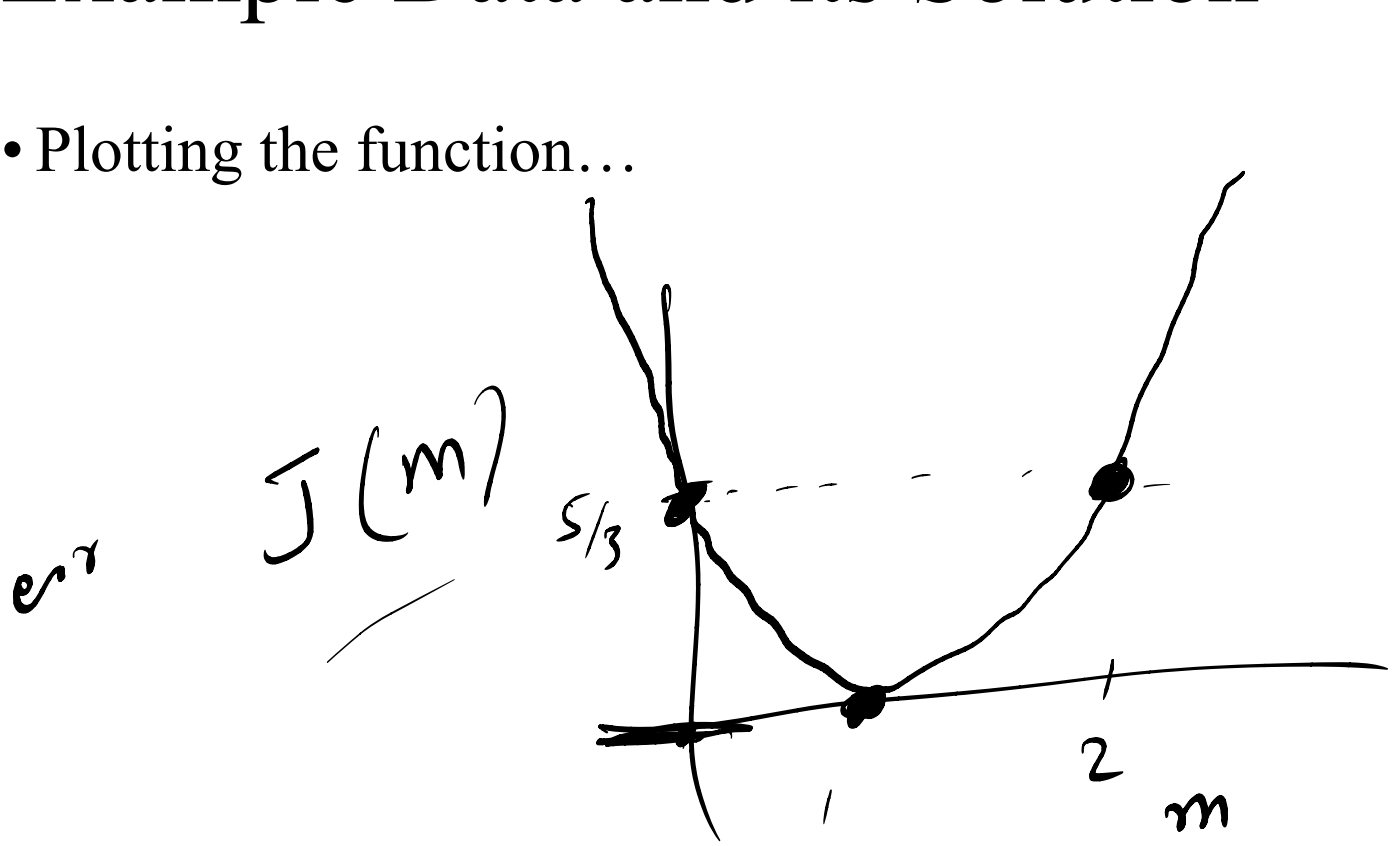
$m = 2$

$J(2) = \frac{1}{3}[(+4]$

$J(2) = -\frac{5}{3}$

Example Data and its Solution

- Plotting the function…



Example Data and its Solution

- Plotting the function…

- Convex function i.e., it has only one critical point, which is the global minimal.
- Set derivative equal to zero and the critical point acquired is the minima

Example Data and its Solution

- Now solve it for the optimal value.

$$\frac{dJ}{dm} = \frac{d}{dm}\left(\frac{1}{3}\left((m-1)^2 + (2m-2)^2\right)\right)$$

$$= \frac{1}{3}\left[\frac{d}{dm}(m-1)^2 + \frac{d}{dm}(2m-2)^2\right]$$

$$= \frac{1}{3}\left[2(m-1) + 2(2m-2)\cdot 2\right]$$

$$= \frac{1}{3}\left[2m - 2 + 8m - 8\right]$$

$$=$$

Example Data and its Solution

- Now solve it for the optimal value.

$$\frac{dJ}{dm} = \frac{1}{3}\left\{ 10m - 10 \right\}$$

$$\frac{dJ}{dm} = 0$$

$$\frac{10}{3}m - \frac{10}{3} = 0$$

$$m = 1$$

Example Data and its Solution

- Now solve it for the optimal value.

① select model based on data.

↳ find the loss => suitable one

↳ for the optimization.

① derivative = 0 → closs

How to find minima of a function (Review):

- There are two approaches to find the minima:

  - **Exact (Closed-form)**: Directly calculates the solution mathematically by solving for $f'(x) = 0$. useful in Lr, useless in the rest

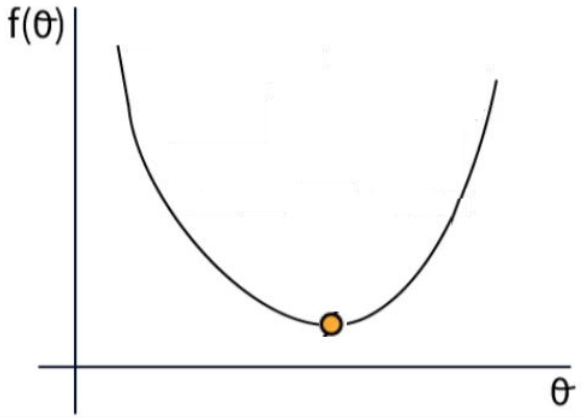
    Important Note: can be used only with a very limited number of algorithms.

  - **Approximation (Iterative approach):** Gradually improves the solution step by step.

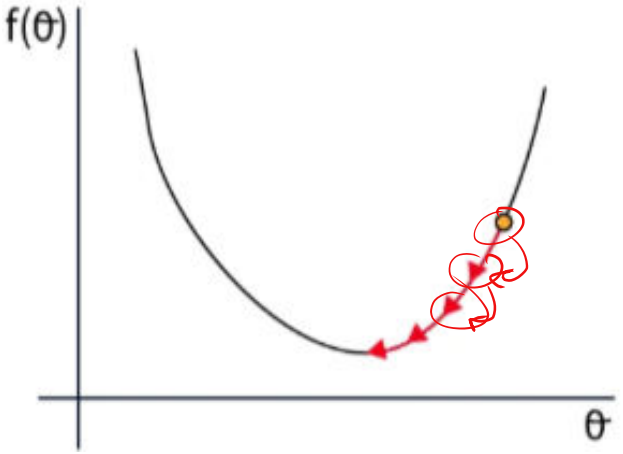
    Done by optimizers (e.g. Gradient Descent, ADAM,…etc).

How to find minima of a function (Review):

Closed-form:



Iterative:

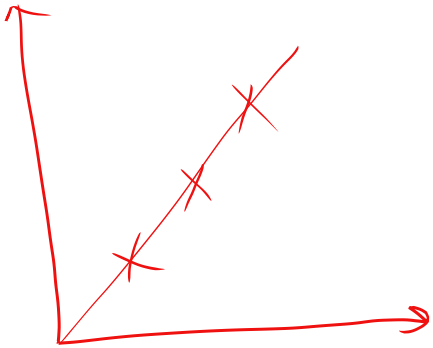


- Example: $y = x^2$ (Solution: $x = 0$)
  - Closed-form Final Result: $x = 0$
  - Iterative Final Result: $x = 0.00001$ (close enough)

How to find minima of a function (Review):

- Let's try to solve this using the closed-form here: assume $\hat{y} = mx$

$$J(m) = \sum_{i=1}^{3} (y_i - mx_i)^2 \qquad J(m) = \sum_{i=1}^{3} (i - mi)^2$$



$$\frac{dJ(m)}{dm} = \frac{d}{dm} \sum_{i=1}^{3} (i - mi)^2 \qquad \frac{dJ(m)}{dm} = \sum_{i=1}^{3} \frac{d}{dm} (i - mi)^2$$

$$\frac{dJ(m)}{dm} = \sum_{i=1}^{3} -2i(i - mi) \qquad -2 \sum_{i=1}^{3} i^2 + 2m \sum_{i=1}^{3} i^2 = 0 \qquad m = 1$$

Hypothesis Function with 2 Variables

- Let's setup regression for linear fur
- The hypothesis function is:

$$\hat{y}_i = mx_i + b$$

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$\hat{y}_i = mx_i + b$$

Gradient of the loss function

- We get the following expressions for the gradient of the cost function

$$\frac{\partial J}{\partial m} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)x_i$$

$$\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N} -2(y_i - \hat{y}_i)$$

Gradient of the loss function

- Simplifying the above expressions, we get:

$$\frac{\partial J}{\partial m} = \frac{-2}{N}\sum_{i=1}^{N} y_i x_i + \frac{2m}{N}\sum_{i=1}^{N} x_i^2 + \frac{2b}{N}\sum_{i=1}^{N} x_i$$

$$\frac{\partial J}{\partial b} = \frac{-2}{N}\sum_{i=1}^{N} y_i + \frac{2m}{N}\sum_{i=1}^{N} x_i + \frac{2b}{N}\sum_{i=1}^{N} 1$$

Gradient of the loss function

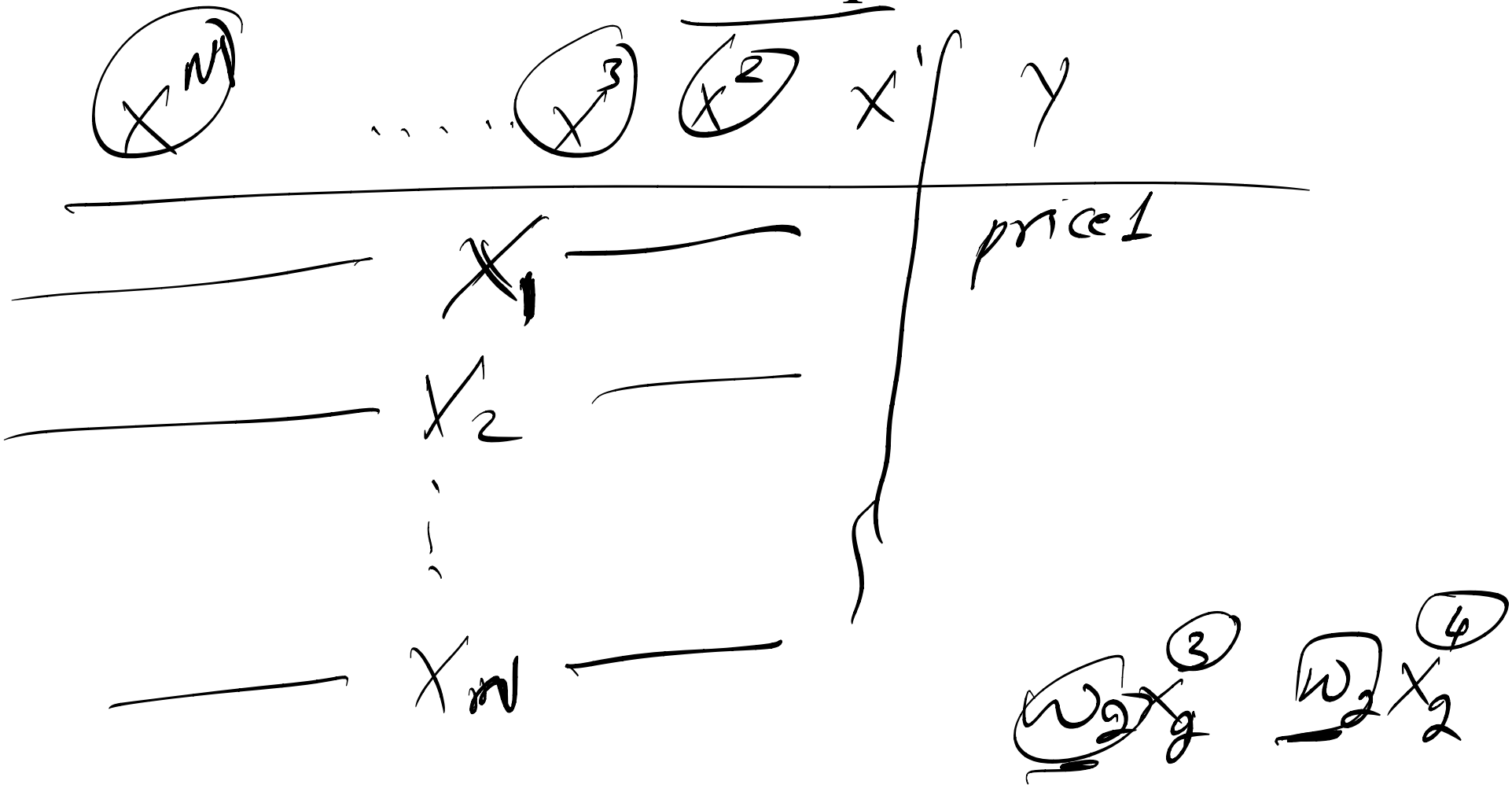
- Setting the Gradient equal to 0, and solving for m and b, we get

$$\begin{bmatrix} \frac{\sum_i x_i^2}{N} & \frac{\sum_i x_i}{N} \\ \frac{\sum_i x_i}{N} & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} \frac{\sum_i x_i y_i}{N} \\ \frac{\sum_i y_i}{N} \end{bmatrix}$$

$$\theta = A^{-1} B$$

$$\theta = A^{-1} B$$

Solution for Data with Multiple Features



Solution for Data with Multiple Features

$$\hat{y}_1 = 1 \cdot w_0 + w_1 x_1^1 + w_2 x_1^2 + w_3 x_1^3 + \cdots + w_m x_1^m$$

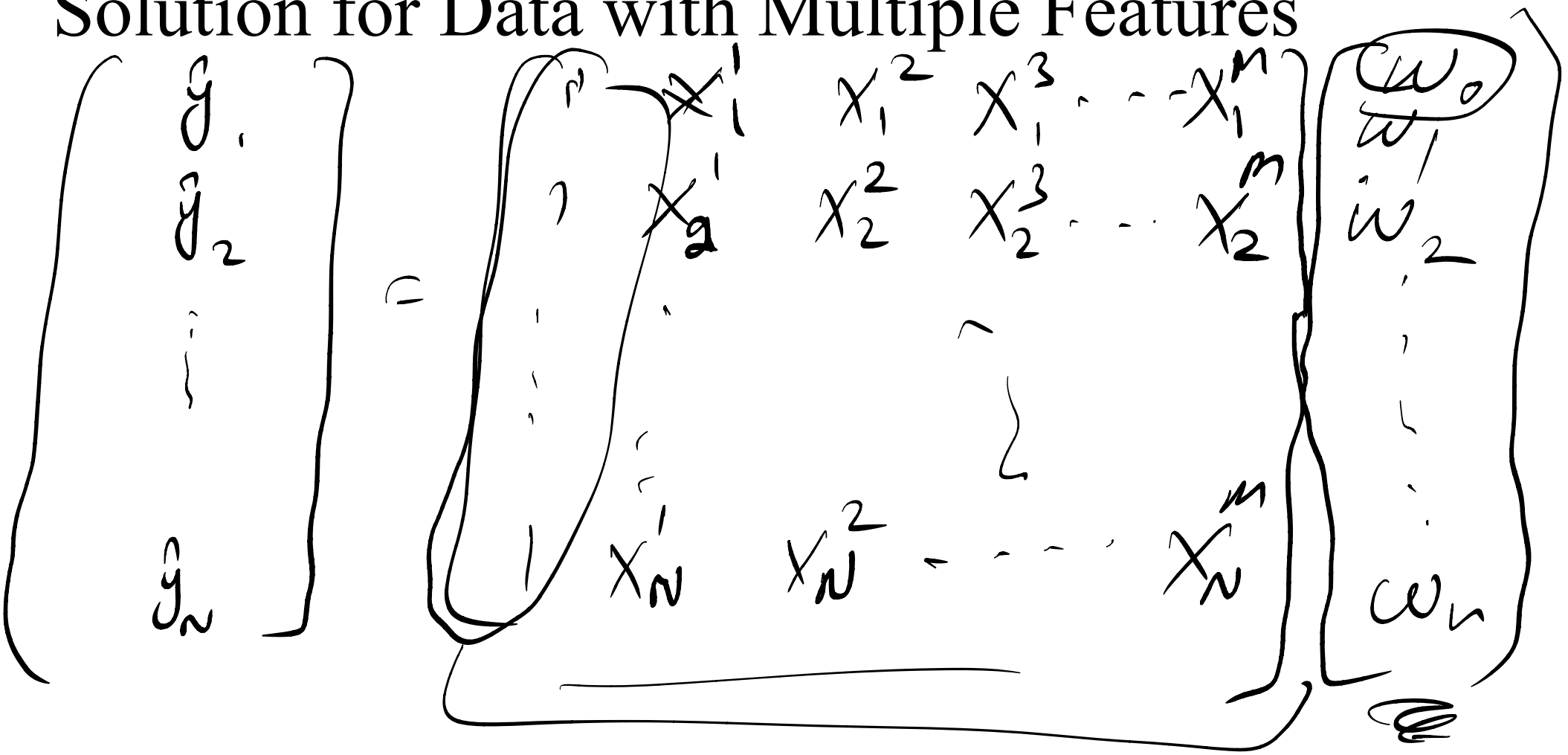
$$\hat{y}_2 = 1 \cdot w_0 + w_1 x_2^1 + w_2 x_2^2 + w_3 x_2^3 + \cdots + w_m x_2^m$$

$$\hat{y}_3 =$$

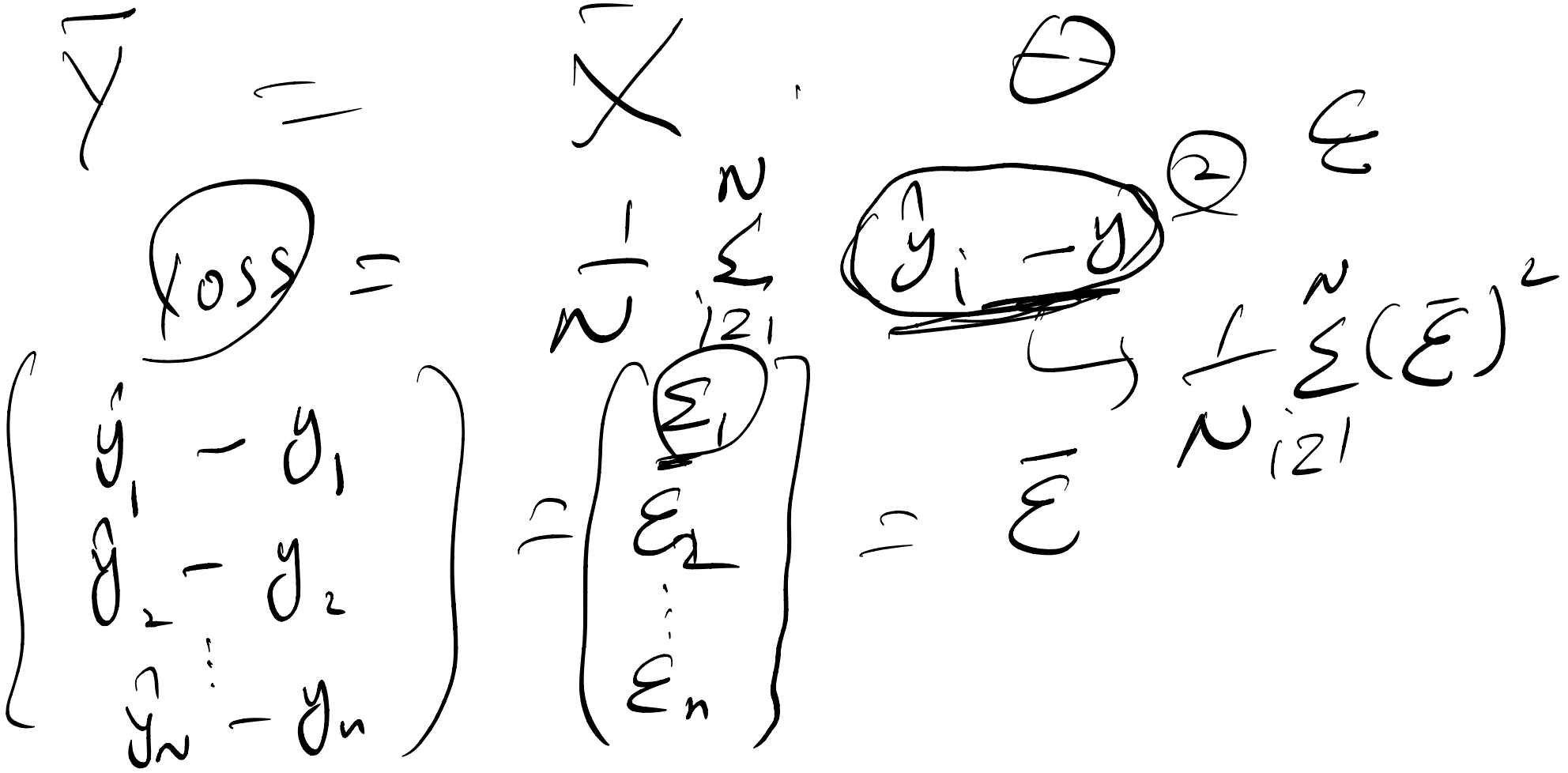
$$\vdots$$

$$\hat{y}_{(n)} = 1 \cdot w_0 + w_1 x_N^1 + w_2 x_N^2 + w_3 x_N^3 + \cdots + w_m x_N^m$$

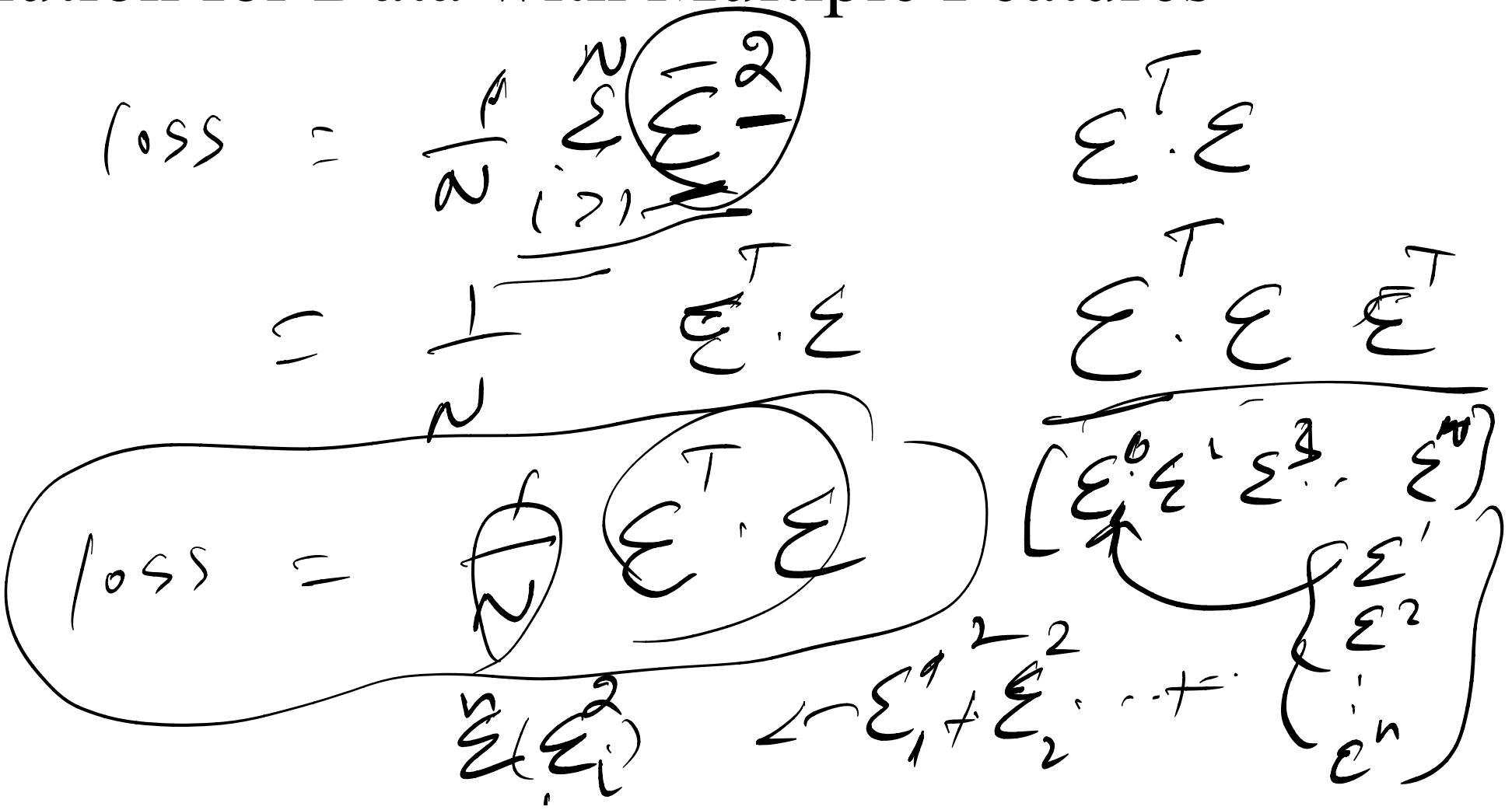
Solution for Data with Multiple Features



Solution for Data with Multiple Features



Solution for Data with Multiple Features



Solution for Data with Multiple Features

$$\text{loss} = \underline{\mathcal{E}}^T \cdot \underline{\mathcal{E}}$$

$$= (\underline{\hat{y}} - y)^T \cdot (\underline{\hat{y}} - y)$$

$$\text{loss} = (mx - y)^T (mx - y) = 0$$

$$\frac{\partial}{\partial m} (mx - y)^T (mx - y) = 0$$

Solution for Data with Multiple Features

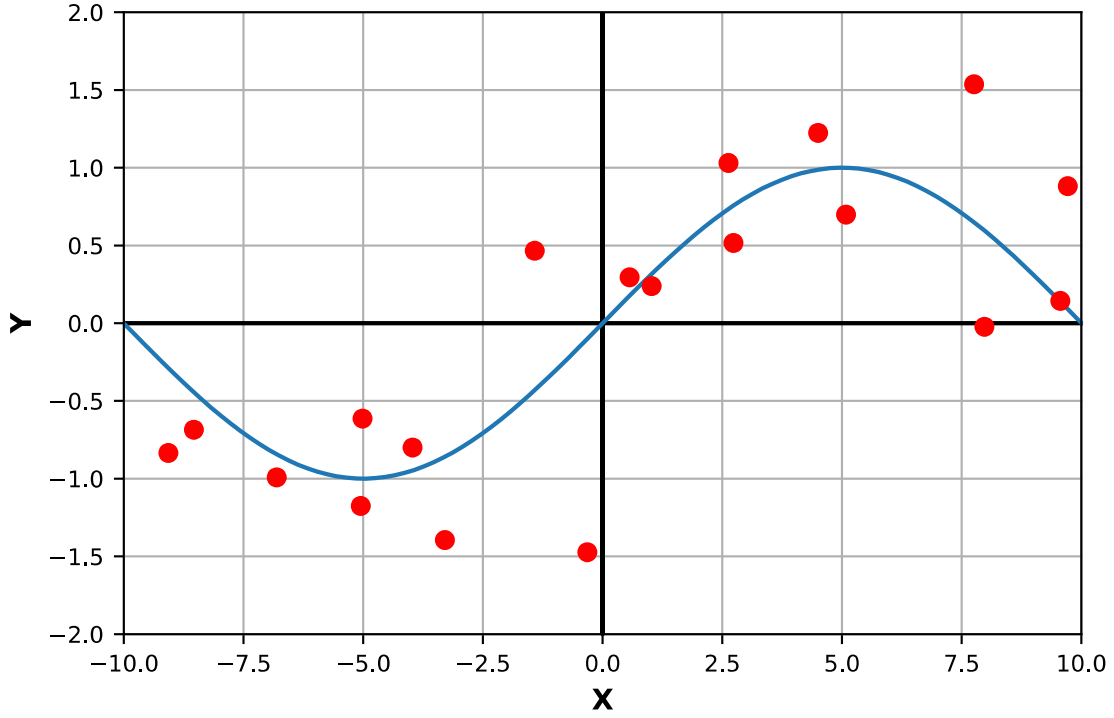
$$\theta = (X^T \cdot X)^{-1} \cdot (X^T \cdot y)$$

$$\theta = A^{-1} \cdot B$$

Normal equation

Fitting Non-linear Data

- What if y is a non-linear function of x, will this approach still work?



Transforming the Feature Space (Feature Engineering)

- We can transform features *xi*

$$x_i = (x_i^1, x_i^2, x_i^3, ..., x_i^m)$$

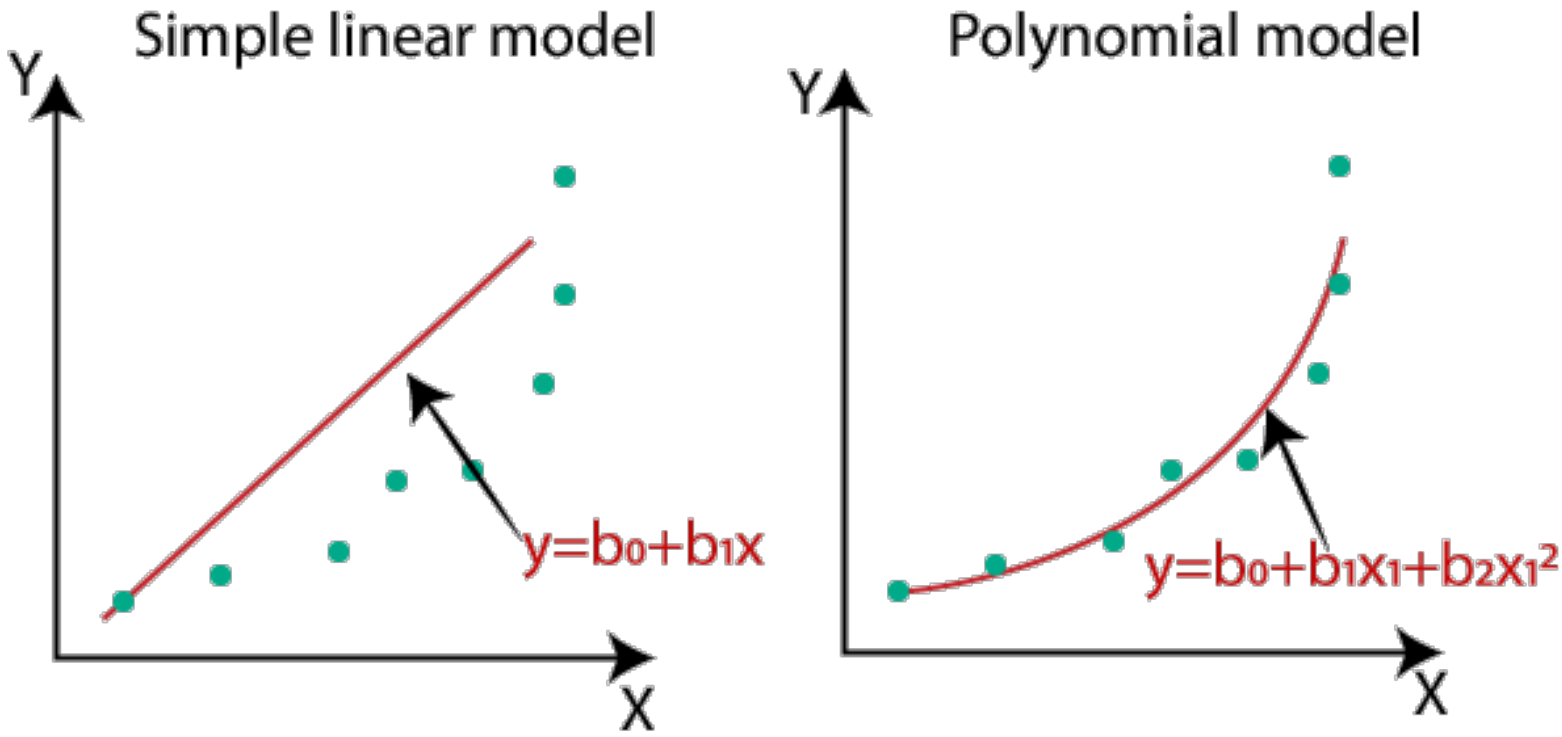
- We will apply some non-linear transformation ϕ:

$$\phi : \mathbb{R}^m \rightarrow \mathbb{R}^M$$

- For example, Polynomial transformation:

$$\phi(x_i) = \{1, x_i^1, x_i^{1,[2]}, ..., x_i^{1,[k]}, x_i^2, x_i^{2,[2]}, ..., x_i^{2,[k]}, ..., x_i^m, x_i^{m,[2]}, ..., x_i^{m,[k]}\}$$

Transforming the Feature Space (Feature Engineering)



Transforming the Feature Space (Feature Engineering)

Example: assume you have:

$$x_1 : Length$$
$$x_2 : Width$$

You can add $x_3 : Area = x_1 * x_2$ to the dataset.

Other types:

- Cosine, splines, radial basis functions, etc.
- Encoding (Label encoding, One-hot,…)
- Domain-related features (e.g. financial measures)
- Time-related features (Day, month, year,…)

Gradient of the loss function

- Let’s get back to the gradients…

Issues with the Approach

- Assume we have 100 variables instead of 2.
- Calcuting gradients like this can quickly become tedious
- **Notice**: Each term on either side of the experssion can be written a dot product of two vectors (maybe we can calculate it more efficiently)?

- Let's explore if we can do something better through **vectorization** (Writing equations as matrices).

Vectorization

- To truly appreciate the power of vectorization. Let's make the problem a little more complex. The hypothesis function is now

$$\hat{y}_i = w_0 + w_1 x_i^1 + w_2 x_i^2 + \cdots + w_M x_i^M$$

- Where w_j are the unknown weights of the data x^j features of the input
- Next, we denote the discrepency between y_i and $\hat{y}_i$ as ϵ_i

$$y_i = \hat{y}_i + \epsilon_i$$

Vectorization

- Now let's collect the above equation for all N datapoints

$$y_1 = \hat{y}_1 + \epsilon_1$$
$$y_2 = \hat{y}_2 + \epsilon_2$$
$$\vdots$$
$$y_N = \hat{y}_N + \epsilon_N$$

Vectorization

- Replacing the values of $\hat{y}$, we get:

$$y_1 = w_0 + w_1 x_1^1 + w_2 x_1^2 + ... + w_M x_1^M + \epsilon_1$$
$$y_2 = w_0 + w_1 x_2^1 + w_2 x_2^2 + ... + w_M x_2^M + \epsilon_2$$
$$\vdots$$
$$y_N = w_0 + w_1 x_N^1 + w_2 x_N^2 + ... + w_M x_N^M + \epsilon_N$$

Vectorization

- Collecting the equations in matrix form:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & x_1^2 & \dots & x_1^M \\ 1 & x_2^1 & x_2^2 & \dots & x_2^M \\ 1 & x_3^1 & x_3^2 & \dots & x_3^M \\ & . & & & \\ & . & & & \\ & . & & & \\ 1 & x_N^1 & x_N^2 & \dots & x_N^M \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

Vectroization

- Notice the rows of the matrix on the right are data samples:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} = \begin{bmatrix} \dots & \mathbf{x_1} & \dots \\ \dots & \mathbf{x_2} & \dots \\ \dots & \mathbf{x_3} & \dots \\ & . & \\ & . & \\ & . & \\ \dots & \mathbf{x_N} & \dots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

Vectorization

$$\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}_{\mathbf{i=1}}^{\mathbf{N}}$$

- Let's formalize some notaitons:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_N \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} \dots & \mathbf{x_1} & \dots \\ \dots & \mathbf{x_2} & \dots \\ \dots & \mathbf{x_3} & \dots \\ & . & \\ & . & \\ & . & \\ \dots & \mathbf{x_N} & \dots \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} w_0 \\ w_1 \\ . \\ . \\ . \\ w_M \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ . \\ . \\ . \\ \epsilon_N \end{bmatrix}$$

$$\boldsymbol{y} = \boldsymbol{X\theta} + \boldsymbol{\epsilon}$$

Cost function for the Vectorized form

- Notice that we are using the MSE cost function:

- Using the defintion of epsilon we can write the above as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2$$

- Using the definition of dot product the above can be written as:

$$J = \frac{1}{N}\sum_{i=1}^{N}(\epsilon_i)^2 = \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}$$

Optimization

- The optimization problem is now:

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon}$$

$$\min_{\boldsymbol{\theta}} \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = \min_{\boldsymbol{\theta}} (\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{\theta}))^T (\boldsymbol{y} - (\boldsymbol{X}\boldsymbol{\theta}))$$

- We will use chain rule to calculate the gradient of the cost function:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \frac{dJ}{d\boldsymbol{\epsilon}} \nabla_{\boldsymbol{\theta}} \boldsymbol{\epsilon}$$

Linear Least Squares

- We get:

$$\frac{\partial}{\partial \boldsymbol{\theta}} J = \boldsymbol{X}^T 2(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\theta})$$

- Setting it equal to zero we can solve for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

Closed-form solution for Linear Regression

$x^2 + y^2 = r$

$2x^2 + 2y^2 = 2r$

Regularization

$$w = \begin{bmatrix} 1 \\ 5 \\ 0 \\ 1 \end{bmatrix}$$

- Non-linear function with **multiple independent** variables.
- What do we do for single variable?

$x^1 = x^3$	x^2	x^1	y

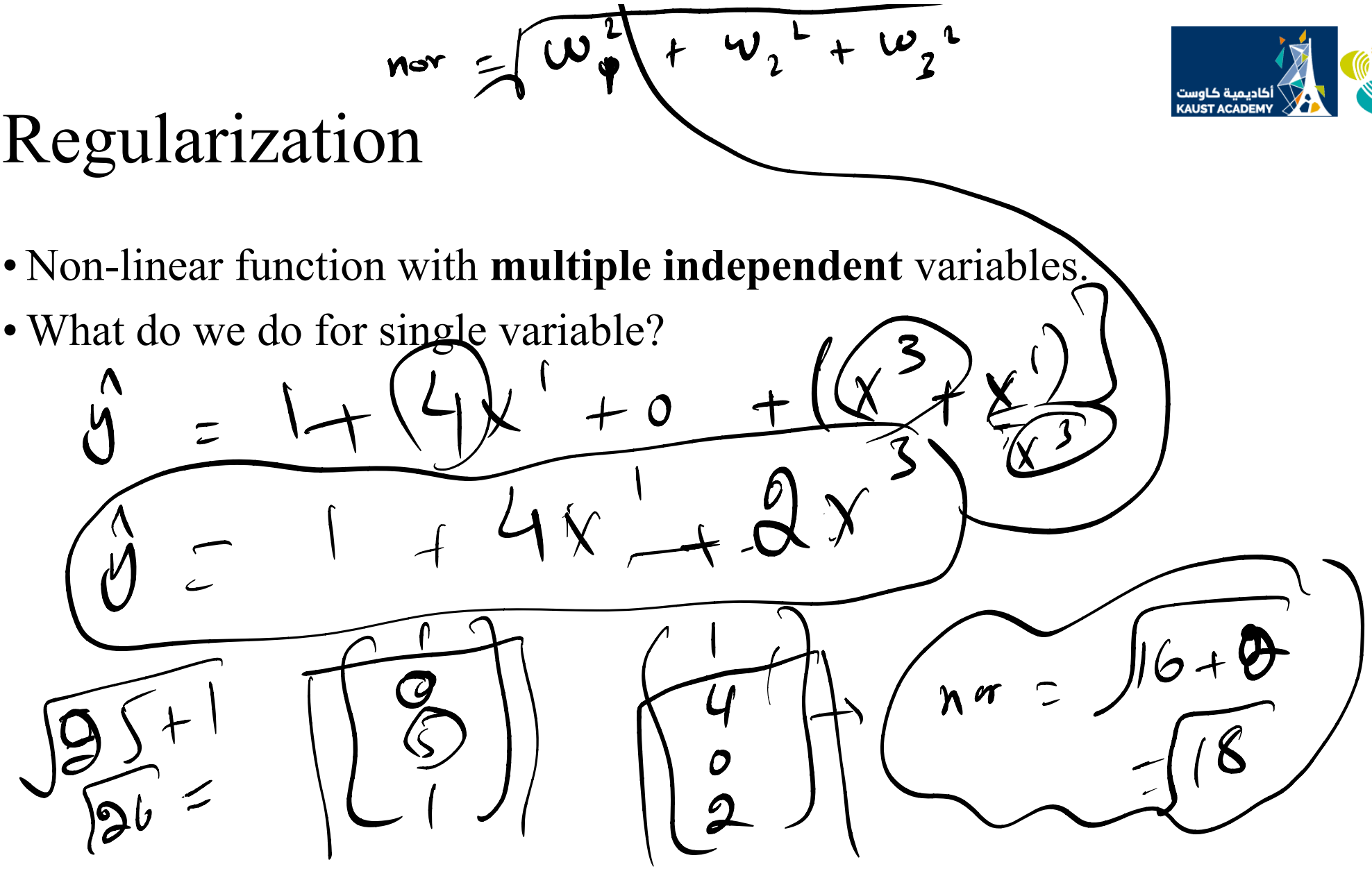
$$\hat{y} = w_0 + w_1 x^1 + w_2 x^2 + w_3 x^3$$

$$\hat{y} = 1 + 5x^1 + 0 + x^3$$

one solution

Regularization

- Non-linear function with **multiple independent** variables.
- What do we do for single variable?



Regularization

- Non-linear function with **multiple independent** variables.
- What do we do for single variable?

$$(X\theta - y)^T (X\theta - y) + \lambda \theta^T \theta$$

$$\theta = (X^T X + \lambda I)(X^T y)$$

Regularization

- Non-linear function with **multiple independent** variables.
- What do we do for single variable?

$$L1$$

$$(X\theta - y)^T (X^T\theta \cdot y) + \sum_{i=1}^{n} |\theta_j|$$