5CS037 - Concepts and Technologies of Artificial Intelligence
Herald College
University of Wolverhampton

Tutorial - 05
Understanding Linear Regression from Machine Learning Perspective.

Siman Giri {Module Leader - 5CS037}

December - 10 - 2025

Instructions:

- Complete all the problems.

Linear Regression with Gradient Descent.

Contents

1 Supervised Learning Problems.

- **Given:** We have a sample data:

$$\mathcal{X} = \{(x_i, y_i)\}_{i=1}^{n}$$

usually iid drawn from an unknown distribution $\mathbb{P}_{x \times y}$

- **Objective:** We want to learn a useful approximation to the underlying function that generated the data.
- **Action:** Action $\mathcal{A}$ is determined by type of learning problem, for supervised learning setup and this course we define following two actions i.e.
  1. **Task of Regression:** If the label, {or target or dependent variable} y is real-valued number. i.e.

$$\mathbf{y} \in \mathbb{R}$$

Then we predict the value of y

  2. **Task of Classification:** If the label, {or target or dependent variable} y is a discrete class. i.e.

$$\mathbf{y} \in \mathbb{C} : \mathbb{C} = \{0, 1, 2, \ldots, k\}$$

$$\text{If } k \leq 2 \rightarrow \text{Binary Classification}$$

$$\text{If } k > 2 \rightarrow \text{Multi - class Classification}$$

Then we assign a class label to input feature.

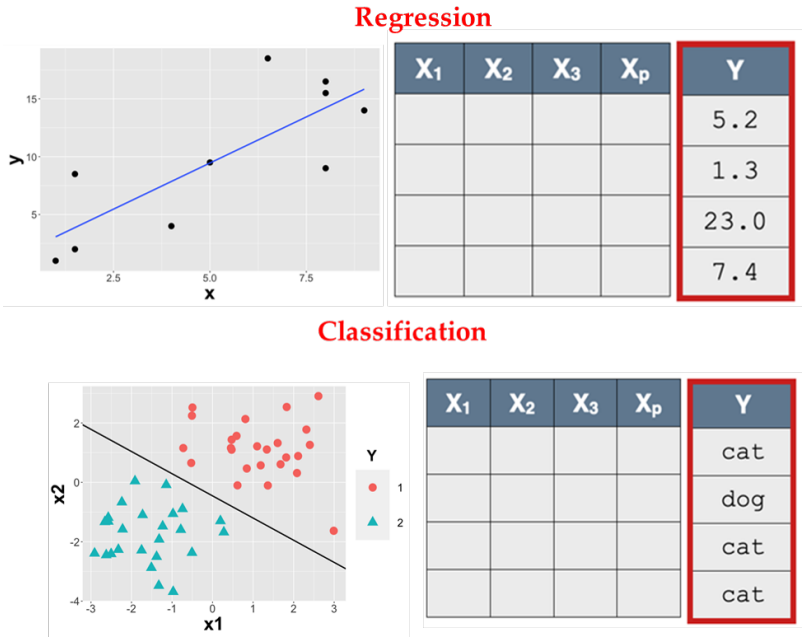

Figure 1: Task of Regression and Classification.

1.1 Setting Up Supervised Learning Problems:

1.1.1 Model Selection Problem:

Machine learning problems (classification, regression and others) are typically ill-posed i.e. the observed data is finite and does not uniquely determine the classification or regression function. In order to find a unique solution, and learn something useful, we must make assumptions { also called inductive bias of the learning algorithm}. How to choose the right model, in particular the right hypothesis class $\mathcal{H}$? `This is the model selection problem.`

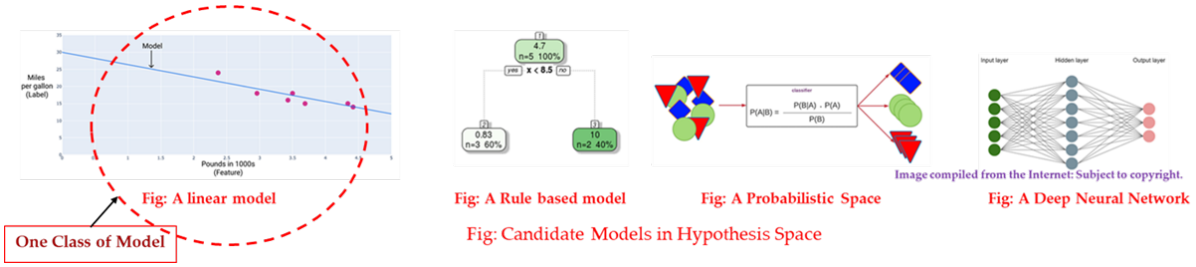

Figure 2: Problem of Model Selection - Pick One! {Slide - 8}.

Model Selection Problem:

We must choose a model or a function $\mathcal{H} \in \mathcal{M}$:

$$\mathcal{H} = \{f_\theta : f_\theta \quad \texttt{belongs to certain functional family parameterized by} \quad \theta\}$$

from a class of model $\mathcal{M}$.
Example: Linear Models, Decision Tree or Neural Network.

1. Model and Parameters:

Every function or model we pick are described by some set of parameters, $\theta \in \Theta$. Parameter θ can be one scalar value or an array of value, Thus, parameters are defined within parametric space $\Theta \in \mathbb{R}^d$. For example a linear predictor - linear regression can have two parameters slope {called weights in Machine learning} and intercept {called bias in Machine Learning}.

2. How do you select a Model from $\mathcal{H} \in \mathcal{M}$?

- In some cases, the ML practitioner will have a good idea of what an appropriate model class is, and will specify it directly.
- In other cases, we may consider several model classes and choose the best based on some objective function.

1.1.2 Model Fitting Problem:

Objective: We want model f described by set of parameters $\theta \in \Theta$ which `best fits` our sample data. The description of `best fits` is abstract and in general define with the help of Empirical Risk Minimization Framework i.e.

1. How do we fit or train the model so we get best set of parameters?
   - Using evaluation measure also called as loss function, which in general is the difference between true label and predicted label.
   - caution!!: Difference means measure of closeness and is determined by task for example regression or classification.
2. How do we determined which set of parameters best fits our function or model?
   - Which ever set of parameters describes the function and produce the minimum risk(estimated), which is an average loss on all our sampled data points.

For every model we discuss in this Module we will use this framework to estimate the best parameter for our selected model.This framework converts our estimation problem to optimization problem, Thus machine learning could be defined as an optimization problem.

1. Machine Learning as an Optimization Problem:

Machine Learning as an Optimization problem could be defined as:

Machine Learning as an Optimization Problem:

For a selected loss function $l(f(x_i, y_i))$ we want to find a set of parameters $\theta^* \in \Theta$ that produces a minimum loss value which can be written as:

$$\theta^*_{\in\Theta} = argmin\mathcal{L}(x, y, \theta, f)$$

Following are some option to solve the optimization problem:

1. Find an analytic solution by solving for root(s) of derivative
2. Try values at random
3. Perform an exhaustive search of available values
4. Perform a systematic non-exhaustive search
5. Find a numeric solution by following the (sub)gradient

1.2 Machine Learning and Generalizations:

The goal of ML is not to replicate the training data, but to predict unseen data well, i.e., to generalize well. For best generalization, we should match the complexity of the hypothesis class H with the complexity of the function underlying the data.But we do not know the data generating function hence the assumption sample data are `iid`.Thus in practice we split our data between Train set and Test set. We assume test set to act as an unseen data and loss or error in test set is an representation of generalization error. In practice we often split available dataset into three disjoint parts:

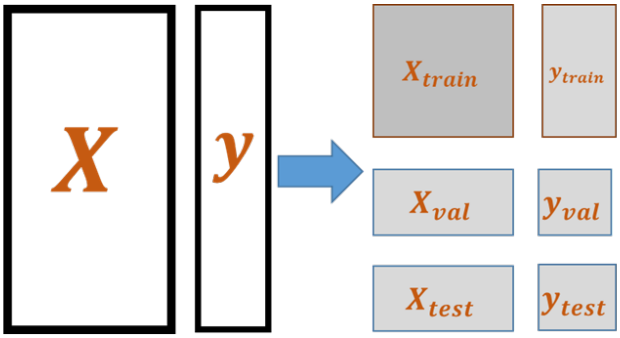

Fig: Three Way Holdout Methods.

$$\mathcal{D}_{in} \rightarrow \begin{cases} D_{train} \rightarrow size(n-(b+a)) \\ D_{val} \rightarrow size\,(b) \\ D_{test} \rightarrow size\,(a) \end{cases}$$

Figure 3: Split of Sample Data.

1. `Training Set:`
   - Used to train, i.e. to fit a function or hypothesis $f \in \mathcal{H}$.
   - Optimize parameters of f_θ, usually done with an optimization algorithm also known as learning algorithm.
2. `Validation Set:`
   - Used to minimize generalization error i.e. to observe how the loss function is behaving during training.
   - Usually done with Grid Search or Random Search.
3. `Test Set:`
   - Used to report the generalization error.
   - We optimize nothing on it, we just evaluate the final model on it.

1.2.1 Understanding Overfitting and Underfitting:

In the context of train and test error:

- Overfitting can be thought as:
  - We have low error in train set (Low training error) but high error in test set (High Test error).
  - High Variance. Characteristics of High Variance model can be:
    * Noise in the dataset
    * Complex models for simpler representation.
    * Trying to fit all the data points including irrelevant information or noise.

        * Problem of Model Fitting.
- Underfitting can be thought as:
    - High error in Train set (High training error),
    - High Bias. Characteristics of High Bias model:
        * Does not capture the true trend in dataset.
        * Problem of model selection.

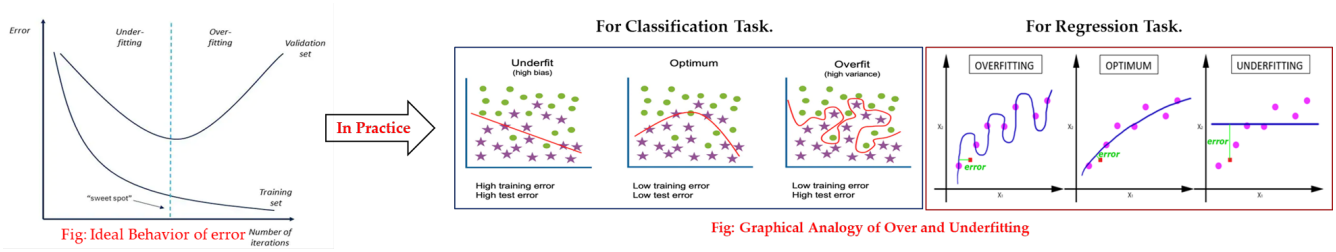

Figure 4: Understanding Over and Underfitting.

1. Some Techniques to avoid Overfitting:

- Early stopping
- Train with more data
- Feature Selection or Data augmentation
- Cross-Validation
- Ensemble Methods
- Regularization

1.3 Summarizing - Your job as a ML Engineer:

1. Choose one each from following:

We must choose:

1. Model Hypothesis:
   - A model $h(x;\Theta)$ (hypothesis class) with parameters Θ.
   - A particular value of Θ determines a particular hypothesis in the class.
   - **Example:** For linear models, $\Theta = \{w_1, w_0\}$, where w_1 is the slope and w_0 is the intercept.
2. Loss Function:
   - A loss function $L(\cdot,\cdot)$ to compute the difference between the desired output (label) y_n and our prediction $h(x_n;\Theta)$.
   - Approximation error (loss):
   $$E(\Theta;X) = \sum_{n=1}^{N} L(y_n, h(x_n;\Theta))$$
   where $E(\Theta;X)$ is the sum of errors over instances.
   - **Examples:**
     - $0/1$ loss for classification.
     - Squared error for regression.
3. Optimization Procedure:
   - An optimization procedure (learning algorithm) is used to find parameters Θ^* that minimize the error:
   $$\Theta^* = \arg\min_{\Theta} E(\Theta;X)$$
   - Different machine learning algorithms differ in any of these choices:
     - Model hypothesis class $h(x;\Theta)$.
     - Loss function $L(\cdot,\cdot)$.
     - Optimization procedure.
4. Make Sure Your Model Generalizes well: Your Model provides accepted result in both train and test set i.e. Model is neither overfitted nor underfitted.

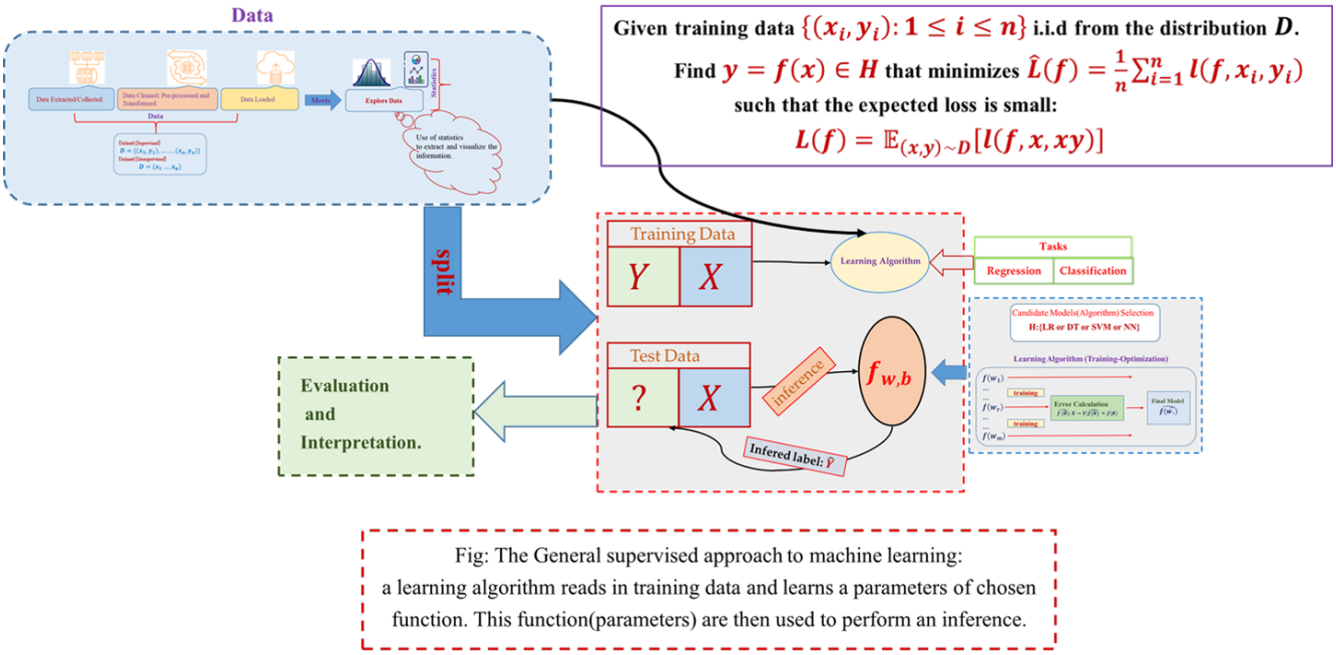

Fig: The General supervised approach to machine learning:
a learning algorithm reads in training data and learns a parameters of chosen function. This function(parameters) are then used to perform an inference.

Figure 5: Supervised Learning in a Nutshell.

2 Introduction to Regression.

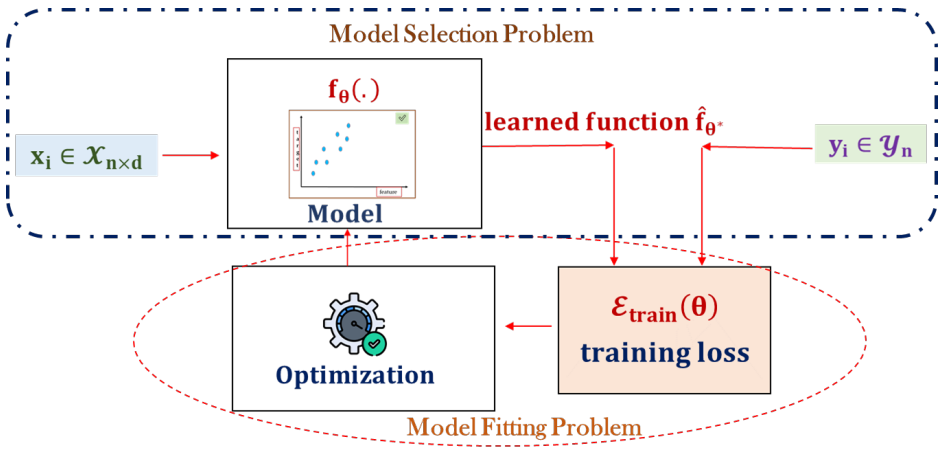

Figure 6: Task of Regression.

Regression is a process for finding the relationship between the inputs and the outputs.

Let the input be $\{x_1, x_2, \ldots, x_N\}$, e.g., circumference of neonatal head, and the output be $\{y_1, y_2, \ldots, y_N\}$, e.g., weight of the neonatal brain.

The training set D is defined as:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

The true relationship between an input x_n and the output y_n is unknown, but it is assumed to be represented by some function f, i.e.,

$$y_n = f(x_n)$$

However, we do not know $f(\cdot)$, and finding it from the training set $\mathcal{D}_(i = 1)^n$ is infeasible. There are infinitely many ways we could define $y_n = f_(\theta)(x_n)$.

To make the problem solvable, we introduce structure by approximating the unknown function $f(\cdot)$ with a proxy model $\hat{f}_\theta(\cdot)$. This proxy $\hat{f}_\theta(\cdot)$ is an approximation of the true relationship between inputs x_n and outputs y_n. For instance, if we assume that the relationship between x_n and y_n is linear, we can build a linear model to approximate the function. This assumption allows us to formulate a specific mathematical model for the task at hand.

Mathematically, this would mean defining the proxy function as:

$$\hat{f}_\theta(x_n) = \theta_1 x_n + \theta_0$$

where θ_1 and θ_0 are the parameters that we aim to estimate from the training data.

This approach provides a structured way to approximate the true, unknown function, making the problem tractable and solvable.

Thus, the goal of regression is to find the optimal parameters θ such that $g_\theta(x_n)$ is a good approximation of the true function $f(x_n)$, minimizing the error between the predicted outputs $g_\theta(x_n)$ and the true outputs y_n for all training examples.

Regression Problem Formulation:

Regression is a supervised learning problem, in which we are given a training dataset of the form:

$$\mathcal{D} = \{(\mathcal{X}_i, \mathcal{Y}_i) \mid \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^n$$

where:

- $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$: is the feature vector for the i^{th} sample.
- y_i: is the target output for the i^{th} sample.
- d: is the number of features in each input vector.

Because $\mathcal{Y}$ values are real-valued, our hypothesis will take the form:

$$f : \mathbb{R}^d \to \mathbb{R},\ f \in \mathcal{M}\ (\text{Model Class})$$

We pick linear regression as the function or hypothesis from the available class of models. Thus, the hypothesis class of linear regression predictors is defined as the set of linear functions:

$$\mathcal{H}_{\text{reg}} = \text{linear}_d = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b \mid \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

where:

- $\mathbf{w} \in \mathbb{R}^d$: is the coefficient vector of the linear function (commonly referred to as the slope), also known as weights in machine learning. There are as many weights as there are features.
- $b \in \mathbb{R}$: is the intercept of the linear function, also known as bias in machine learning. There is only one bias per model.

Regression as an Optimization Problem:

Given the data points $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, regression is the process of finding the parameter $\theta \in \{\Theta = \mathbb{R}^d\}$ of a function $f_\theta(.)$ such that the training loss is minimized:

$$\hat{\theta} = argmin_{\theta \in \mathbb{R}^d} \sum_{i=1}^{n} \mathcal{L}(y_i, f_\theta(x_i) \tag{01}$$

Here $\mathcal{L}(\cdot)$ is the loss between a pair of true observation y_i and the predicted $\hat{y}_i$.

2.1 Linear Models for Regression:

1. Linear Models Definition:

Linear models are a class of models in machine learning and statistics that assume a linear relationship between the input features ($\mathcal{X}$) and the output ($\mathcal{Y}$) variable(s). These models predict the output as a weighted sum of the input features, along with a bias term.

Linear Relationship:
The relationship between the inputs and the output is expressed as:

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

or more compactly using vector notation:

$$y = \mathbf{w}^\top \mathbf{x} + b$$

Where:

- $y \in \mathbb{R}$: is the output (dependent variable).
- $\mathbf{x} = [x_1, x_2, \ldots, x_d]^\top \in \mathbb{R}^d$: is the input vector of features (independent or explanatory variables).
- $\mathbf{w} = [w_1, w_2, \ldots, w_d]^\top \in \mathbb{R}^d$: are the weights (parameters of the model) associated with the features.
- $b \in \mathbb{R}$: is the bias (intercept), which allows the model to fit data that does not pass through the origin.

For Training:
Given n observations in a training set, the model seeks to learn the parameters $\mathbf{w}$ and b that best fit the observed data.

2. Cautions in Fitting Linear Models:

Before fitting a linear model to observed data, it is essential to assess whether a meaningful relationship exists between the explanatory (independent) variables and the dependent variable. This can be evaluated through exploratory data analysis, such as examining scatter plots. If no discernible association (e.g., increasing or decreasing trends) is evident, applying a linear regression model may be inappropriate and is unlikely to yield a reliable or useful representation of the data.

Furthermore, attempting to fit a linear model in the absence of an underlying linear relationship can lead to misleading results and poor predictive performance. Therefore, it is critical to validate the assumptions of linear regression before proceeding with model fitting.

2.2 Fitting Linear Regression to ERM Framework:

1. Loss Function for Linear Regression:

The loss function quantifies the error between the predicted values y_i and the actual values $\hat{\mathrm{y}}_i$.For linear regression, the goal is to minimize this error to find the optimal parameters θ. Among the various functions used to measure how close $\hat{y}_i$ and y_i are, the most common option is to use the squared distance between these values:

$$d^2(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2 = (\hat{y}_i - y_i)^2 \implies \mathrm{err}_i \quad \text{(called "Residuals")}.$$

This represents the point-wise measure of loss or error. However, we are interested in an overall measure, i.e., the empirical risk, which is an average of the point-wise losses.

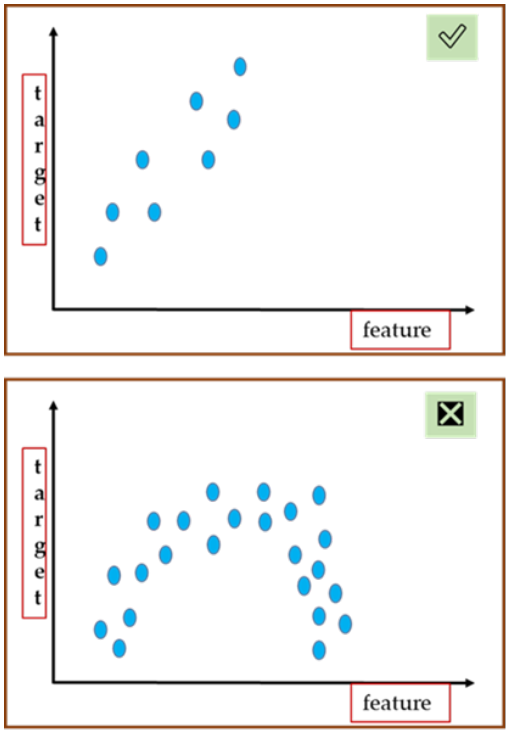

Figure 7: When Linear Models are good fit?

The Sum of Squared Errors (SSE):

$$SSE = \sum_{i=1}^{n} \text{err}_i$$

The Mean Squared Error (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^{n} \text{err}_i$$

2. Why Squared errors?

- Penalizes Larger Errors: Squaring ensures all errors are positive and gives higher weight to larger errors, making the model sensitive to outliers.
- Smooth Differentiability: The squared error function is smooth and differentiable, making it easy to compute gradients during optimization.

3. Linear Regression in ERM Framework:

Why?

By putting Linear regression into ERM framework allowed us to define learning Linear regression as Optimization

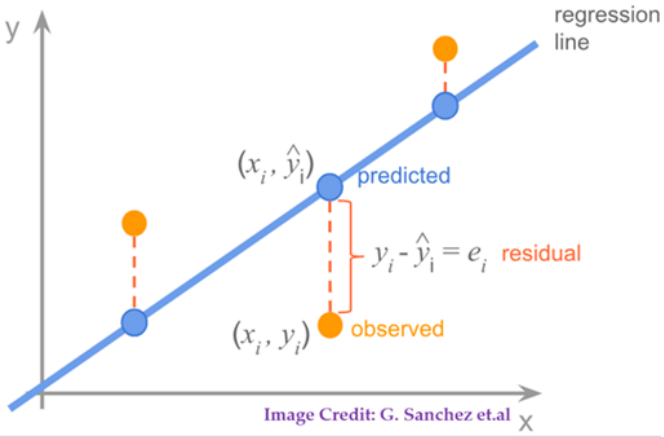

Fig: Residuals aka error aka squared error.

Figure 8: Squared Error - Intuition {slide - 31}

problem.

Linear Regression as a Learning Problem:

Given: A dataset

$$\mathcal{D}_n = \{(\mathbf{x}_i, y_i) \mid \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1, 2, \ldots, n\},$$

A set of candidate linear functions:

$$\mathcal{H}_{\text{reg}} = \text{linear}_d = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle + b \mid \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

parameterized by

$$\theta = [w^0, w^1, w^2, \ldots, w^d] \in \mathcal{W} \to \mathbb{R}^{d+1} \subset \Theta.$$

A loss function

$$\ell(f_\theta(\mathbf{x}_i), y_i) \to (f_\theta(\mathbf{x}_i) - y_i)^2 \quad \text{(squared error)}.$$

The goal is to find θ^* that minimizes the empirical risk given by:

$$\hat{R}(f_\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathbf{x}, y, \theta, f) = \frac{1}{n} \sum_{i=1}^{n} (f_\theta(\mathbf{x}_i) - y_i)^2.$$

Therefore, the learned function $\hat{f}$ can be written as:

$$\hat{f} = \underset{\theta^* \to \{\mathbf{w}, b\}}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} (f_{\theta^*}(\mathbf{x}_i) - y_i)^2.$$

Linear Regression as Optimization Problem:

For linear regression, the optimization problem can be written as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \quad \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b.$$

Breaking it into components:

- Objective: Minimize the squared error between the predicted value $\hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b$ and the true value y_i.
- Solution: Estimate $\boldsymbol{\theta}^* = [\mathbf{w}, b]$ that best fit the data.

Reminder: There exists an analytical or closed-form solution for the problem known as the least square method, given by the following Normal equation:

$$\boldsymbol{\theta}^* = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

However, we are interested in learning the parameter $\boldsymbol{\theta}^* = \{[w^0, w^1, w^2, \ldots, w^d] \in \mathbf{W} \rightarrow \mathbb{R}^{d+1}\}$.

2.3 Gradient Descent Overview:

2.3.1 Gradient Descent in General:

Gradient Descent is a first-order iterative optimization algorithm used to minimize a cost (or loss) function. The core idea of gradient descent is to move towards the minimum of the cost function by following the direction of the steepest descent, which is the negative of the gradient of the function.

1. Steps in Gradient Descent:

1. `Initialize Parameters`: Choose initial values for the parameters (e.g., weights and bias in linear regression). These are often initialized randomly or with zeros.
2. `Compute the Gradients`: Calculate the gradient of the cost function with respect to each parameter. The gradient is a vector of partial derivatives, pointing in the direction of the steepest increase of the cost function.
3. `Update the Gradients`: Update the parameters by moving in the direction of the negative gradient, with a step size proportional to the gradient (controlled by the learning rate).
4. `Repeat`: Repeat steps 2 and 3 until convergence, i.e., the change in the cost function becomes very small, or a maximum number of iterations is reached.

Gradient Descent Update Rule:

At each iteration, the parameters are updated based on the gradient of the cost function $J(\theta)$ with respect to each parameter θ_i as:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

2. Convergence and Learning Rate:

The choice of the learning rate α is crucial for the convergence of gradient descent:

- If α is too small, the convergence will be slow.
- If α is too large, the algorithm may overshoot the minimum and fail to converge.

$$w_1 = w_2 - \alpha \frac{d\,E(w_1)}{dw_1}$$
$$= \alpha: Too\ small\ number.$$

$$w_1 = w_1 - \alpha \frac{d\,E(\theta_1)}{d\,w_j}$$
$$= \alpha: Too\ Large\ number.$$

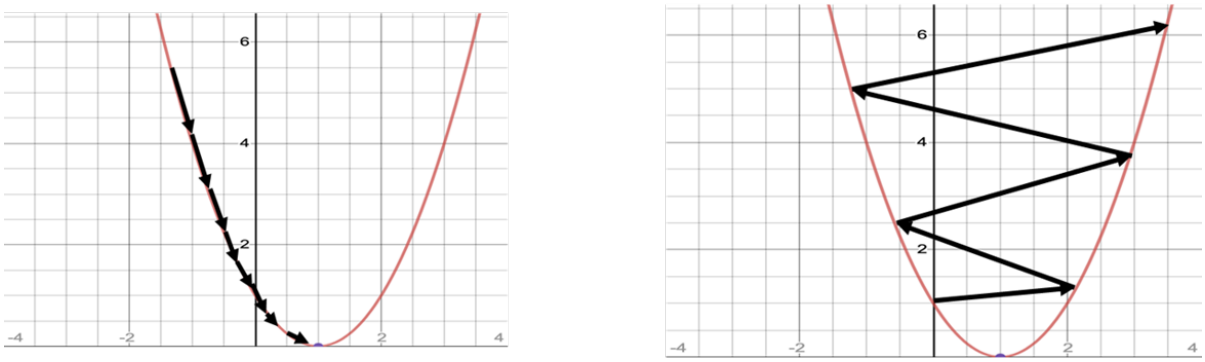

Fig: Understanding the Impact of Learning Rate.

Figure 9: Effect of α on Convergence.

3. Convergence Criteria:

The algorithm stops when:

- The cost function $J(\theta)$ does not change significantly (below a threshold) between iterations.
- A predefined number of iterations is reached.

2.4 Example Exercise on Gradient Descent:

Problem - 1:

Consider the quadratic function:

$$f(x) = (x-3)^2 + 4$$

We aim to find the value of $x)$ that minimizes $f(x)$ using gradient descent.

Sample Solution:

Let's calculate the first few iterations of gradient descent by hand for the given function:

Problem Setup:

- The function:
$$f(x) = (x-3)^2 + 4$$
- The derivative:
$$f'(x) = 2(x-3)$$
- The gradient descent update rule:
$$x_{\text{new}} = x_{\text{old}} - \alpha \cdot f'(x_{\text{old}})$$
- Parameters:
  - **Initial guess:** $x_0 = 0$
  - **Learning rate:** $\alpha = 0.1$

Iterations:

1. **Iteration - 1:**

$$\begin{aligned} \texttt{Start with } x_0 &= 0. \\ \texttt{Compute the gradient: } f'(x_0) &= 2(0-3) = -6. \\ \texttt{Update } x: \quad x_1 &= x_0 - \alpha \cdot f'(x_0) \\ &= 0 - 0.1 \cdot (-6) = 0.6. \end{aligned}$$

2. **Iteration - 2:**

$$\begin{aligned} \texttt{Start with } x_1 &= 0.6. \\ \texttt{Compute the gradient: } f'(x_1) &= 2(0.6-3) = 2 \cdot (-2.4) = -4.8. \\ \texttt{Update } x: \quad x_2 &= x_1 - \alpha \cdot f'(x_1) \\ &= 0.6 - 0.1 \cdot (-4.8) = 0.6 + 0.48 = 1.08. \end{aligned}$$

3. **Iteration - 3:**

$$\begin{aligned} \text{Start with } x_2 &= 1.08. \\ \text{Compute the gradient: } f'(x_2) &= 2(1.08-3) = 2 \cdot (-1.92) = -3.84. \\ \text{Update } x: \quad x_3 &= x_2 - \alpha \cdot f'(x_2) \\ &= 1.08 - 0.1 \cdot (-3.84) = 1.08 + 0.384 = 1.464. \end{aligned}$$

4. **Iteration - 4:**

$$\begin{aligned} \text{Start with } x_3 &= 1.464. \\ \text{Compute the gradient: } f'(x_3) &= 2(1.464-3) = 2 \cdot (-1.536) = -3.072. \\ \text{Update } x: \quad x_4 &= x_3 - \alpha \cdot f'(x_3) \\ &= 1.464 - 0.1 \cdot (-3.072) = 1.464 + 0.3072 = 1.7712. \end{aligned}$$

Summarizing the Result:

Iteration	x_{old}	$f'(x)$	x_{new}
1	0.0000	−6.0000	0.6000
2	0.6000	−4.8000	1.0800
3	1.0800	−3.8400	1.4640
4	1.4640	−3.0720	1.7712

Further iterations will refine this value, and the process converges as the gradient becomes smaller.

2.5 Task - to - Do:

1. For Problem -1:

1. Implement gradient descent in Python to minimize $f(x)$ and run for more iterations.
2. Plot the function $f(x)$ and visualize the steps of gradient descent.

2. Minimize the function:

Consider the cubic function:

$$f(x) = x^3 - 4x^2 + 6x$$

We aim to minimize $f(x)$ using gradient descent. Using following Parameters:

- Start with an initial guess $x_o = 0$.
- Learning rate: $\alpha = 0.1$
- Stop when $|x_{new} - x_{old}| < 0.0001$ or after 5 iterations.

Solve step - by - step with pen and on paper.

2.5.1 Gradient Descent for Linear Regression:

1. The Objective Function:

For linear regression, the cost function is the Mean Squared Error (MSE):

$$\mathcal{L}(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2,$$

where

$$\hat{y}_i = w^\top x_i + b, \quad \text{and} \quad \theta = \{w, b\}.$$

The goal is to minimize the loss function $\mathcal{L}(w, b)$.

2. Gradient Descent Algorithm:

Initialize Parameters: Start with initial values for

$$\theta = \{w, b\},$$

often set to zeros or small random values.

Update Rule: Iteratively update θ in the direction of the negative gradient of the cost function:

$$\theta_{new} := \theta_{old} - \alpha \nabla_\theta J(\theta),$$

where:

- α is the learning rate (step size).
- $\nabla_\theta J(\theta)$ is the gradient of the cost function.

$$\texttt{also written as:} \frac{\partial \mathcal{L}(\{w, b\})}{\partial \{w, b\}}$$

3. Gradient Computations:

The gradients for the parameters $\theta = w, b$ are computed as follows:

Computing the Gradients - {Optional}:

1. Gradient w.r.t. W

Substitute $\hat{y}_i = wx_i + b$ into the cost function:

$$J(w, b) = \frac{1}{n} \sum_{i=1}^{n} (y_i - (wx_i + b))^2 .$$

Differentiate $J(w, b)$ with respect to w:

$$\frac{\partial J}{\partial w} = \frac{\partial}{\partial w} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - (wx_i + b))^2 \right].$$

Since $\frac{1}{n}$ is constant, it can be factored out:

$$\frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial w} (y_i - (wx_i + b))^2 .$$

Using the chain rule:

$$\frac{\partial}{\partial w} \left(e_i^2 \right) = 2e_i \cdot \frac{\partial e_i}{\partial w},$$

where $e_i = y_i - (wx_i + b)$.
Compute $\frac{\partial e_i}{\partial w}$:

$$e_i = y_i - (wx_i + b) \quad \Longrightarrow \quad \frac{\partial e_i}{\partial w} = -x_i.$$

Substitute back:

$$\frac{\partial}{\partial w} \left(e_i^2 \right) = 2e_i \cdot (-x_i).$$

Therefore:

$$\frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} [2 \cdot (y_i - (wx_i + b)) \cdot (-x_i)] .$$

Simplify:

$$\frac{\partial J}{\partial w} = -\frac{2}{n}\sum_{i=1}^{n} x_i \left(y_i - \hat{y}_i\right),$$

where $\hat{y}_i = wx_i + b$.

2. Gradient w.r.t. b

Differentiate $J(w, b)$ with respect to b:

$$\frac{\partial J}{\partial b} = \frac{\partial}{\partial b}\left[\frac{1}{n}\sum_{i=1}^{n}\left(y_i - (wx_i + b)\right)^2\right].$$

Factor out the constant $\frac{1}{n}$:

$$\frac{\partial J}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial b}\left(y_i - (wx_i + b)\right)^2.$$

Using the chain rule:

$$\frac{\partial}{\partial b}\left(e_i^2\right) = 2e_i \cdot \frac{\partial e_i}{\partial b}.$$

Compute $\frac{\partial e_i}{\partial b}$:

$$e_i = y_i - (wx_i + b) \quad \Longrightarrow \quad \frac{\partial e_i}{\partial b} = -1.$$

Substitute back:

$$\frac{\partial}{\partial b}\left(e_i^2\right) = 2e_i \cdot (-1).$$

Therefore:

$$\frac{\partial J}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\left[2 \cdot \left(y_i - (wx_i + b)\right) \cdot (-1)\right].$$

Simplify:

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

Final Results:

The gradients of the MSE loss are:

$$\frac{\partial \text{MSE}}{\partial w} = \frac{-2}{n}\sum_{i=1}^{n}\left(y_i - \left(w^\top x_i + b\right)\right)x_i,$$

$$\frac{\partial \text{MSE}}{\partial b} = \frac{-2}{n}\sum_{i=1}^{n}\left(y_i - \left(w^\top x_i + b\right)\right).$$

The weights w and bias b are updated as:

$$w \leftarrow w - \alpha\frac{\partial \text{MSE}}{\partial w}, \quad b \leftarrow b - \alpha\frac{\partial \text{MSE}}{\partial b}.$$

3. Intuition Behind the Gradients

Gradient w.r.t. w

The derivative incorporates the input feature x_i, weighting the contribution of each error $y_i - \hat{y}_i$. This ensures that updates to w are proportional to both the magnitude of the error and the importance of the input feature.

Gradient w.r.t. b

This derivative does not involve x_i, as b is independent of the input feature. The update to b is based solely on the average error across all data points.

2.6 Example: Gradient Descent for Minimizing Mean Squared Error (MSE):

Problem Statement

Suppose we have a dataset with $n = 3$ points:

$$(x_1, y_1) = (1, 2), \quad (x_2, y_2) = (2, 4), \quad (x_3, y_3) = (3, 6).$$

We aim to fit a line $y = wx + b$ to minimize the Mean Squared Error (MSE), defined as:

$$\text{MSE}(w, b) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - (wx_i + b)\right)^2,$$

where:

- n: Number of data points.
- y_i: Actual value for the i-th data point.
- x_i: Input feature for the i-th data point.
- w: Weight (slope of the line).
- b: Bias (intercept of the line).

We will minimize the MSE using gradient descent for the parameters w and b.

1. Define the Gradients of MSE

The gradient of the Mean Squared Error (MSE) with respect to w and b are given by:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{n} \sum_{i=1}^{n} x_i \cdot \left(y_i - (wx_i + b)\right),$$

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{n} \sum_{i=1}^{n} \left(y_i - (wx_i + b)\right).$$

2. Gradient Descent Update Rules

The parameters w and b are updated iteratively using the following rules:

$$w_{\text{new}} = w_{\text{old}} - \alpha \cdot \frac{\partial \text{MSE}}{\partial w},$$

$$b_{\text{new}} = b_{\text{old}} - \alpha \cdot \frac{\partial \text{MSE}}{\partial b},$$

where:

- α: Learning rate.
- $w_{\text{old}}, b_{\text{old}}$: Current values of the parameters.
- $w_{\text{new}}, b_{\text{new}}$: Updated values of the parameters.

3. Initial Values and Learning Rate

We use the following initial values and learning rate:

$$w_0 = 0, \quad b_0 = 0, \quad \alpha = 0.01.$$

4. Perform Calculations for 3 Iterations

For $n = 3$ data points

$$(x_1, y_1) = (1, 2), (x_2, y_2) = (2, 4), (x_3, y_3) = (3, 6)$$

calculate the updates step by step for 3 iterations.

$$\texttt{Initial Guess:} \quad w_0 = 0; \quad b_0 = 0$$

Iteration - 1:

1. Gradient of MSE with respect to w

The gradient of the Mean Squared Error (MSE) with respect to w is given by:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{3} \sum_{i=1}^{3} x_i \cdot (y_i - (w_0 x_i + b_0)) \,.$$

Substitute $w_0 = 0$ and $b_0 = 0$:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{3} \left[1 \cdot (2 - 0) + 2 \cdot (4 - 0) + 3 \cdot (6 - 0)\right].$$

Simplifying the terms:

$$\frac{\partial \text{MSE}}{\partial w} = -\frac{2}{3} \cdot (2 + 8 + 18) = -\frac{2}{3} \cdot 28 = -18.67.$$

2. Gradient of MSE with respect to b

The gradient of the Mean Squared Error (MSE) with respect to b is given by:

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{3} \sum_{i=1}^{3} (y_i - (w_0 x_i + b_0)) \,.$$

Substitute $w_0 = 0$ and $b_0 = 0$:

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{3} \left[(2 - 0) + (4 - 0) + (6 - 0)\right].$$

Simplifying the terms:

$$\frac{\partial \text{MSE}}{\partial b} = -\frac{2}{3} \cdot 12 = -8.$$

3. Update Parameters using Gradient Descent

$$w_1 = w_0 - \alpha \cdot \frac{\partial \text{MSE}}{\partial w} = 0 - 0.01 \cdot (-18.67) = 0.1867.$$

$$b_1 = b_0 - \alpha \cdot \frac{\partial \text{MSE}}{\partial b} = 0 - 0.01 \cdot (-8) = 0.08.$$

Thus, after the first iteration:

$$w_1 = 0.1867, \quad b_1 = 0.08.$$

Iteration - 2:

1. Gradient of MSE with respect to w

`Your Turn - Compute.`

2. Gradient of MSE with respect to b

`Your Turn - Compute.`

3. Update Parameters using Gradient Descent

Now, using the updated gradients, we can update the parameters:

$$w_2 = w_1 - \alpha \cdot \frac{\partial \text{MSE}}{\partial w} = 0.1867 - 0.01 \cdot (-16.75) = 0.1867 + 0.1675 = 0.3542.$$

$$b_2 = b_1 - \alpha \cdot \frac{\partial \text{MSE}}{\partial b} = 0.08 - 0.01 \cdot (-7.50) = 0.08 + 0.075 = 0.155.$$

Thus, after the second iteration:

$$w_2 = 0.3542, \quad b_2 = 0.155.$$

Iteration - 3:

1. Gradient of MSE with respect to w

`Your Turn - Compute.`

2. Gradient of MSE with respect to b

`Your Turn - Compute.`

3. Update Parameters using Gradient Descent

`Your Turn - Compute.`

Thus, after the third iteration:

$$w_3 = 0.5016, \quad b_3 = 0.2177.$$

2.7 Exercise - Implement Linear Regression using Gradient Descent with Pen and on Paper:

Given the dataset:

x	y
1.0	2.2
2.0	2.8
3.0	4.5
4.0	3.7
5.0	5.1

The initial parameters are $w = 0$, $b = 0$, and the learning rate $\alpha = 0.01$.

Task:

1. Compute the predicted values y_{pred} for the given dataset.
2. Calculate the gradients for w and b using the Mean Squared Error (MSE) cost function.
3. Update the parameters w and b using the gradient descent update rules.
4. Compute the updated predictions and calculate the cost after one epoch.
5. Summarize the results after one epoch, including the updated parameters, predictions, and cost.

Step-by-Step Calculation for One Epoch

Step -1 - Compute Predictions:

The predicted values are given by:

$$y_{\text{pred}} = w \cdot x + b$$

Substitute $w = 0$ and $b = 0$:

$$y_{\text{pred}} = [0, 0, 0, 0, 0]$$

Step - 2 - Compute Gradients:

The gradients for w and b are:

$$\frac{\partial \text{Cost}}{\partial w} = -\frac{2}{n} \sum x \cdot (y - y_{\text{pred}})$$

$$\frac{\partial \text{Cost}}{\partial b} = -\frac{2}{n} \sum (y - y_{\text{pred}})$$

error:

$$y - y_{\text{pred}} = [2.2, 2.8, 4.5, 3.7, 5.1]$$

Gradient for w:

$$\begin{aligned} x \cdot (y - y_{\text{pred}}) &= [1 \cdot 2.2, 2 \cdot 2.8, 3 \cdot 4.5, 4 \cdot 3.7, 5 \cdot 5.1] \\ &= [2.2, 5.6, 13.5, 14.8, 25.5] \\ \sum x \cdot (y - y_{\text{pred}}) &= 2.2 + 5.6 + 13.5 + 14.8 + 25.5 = 61.6 \\ \frac{\partial \text{Cost}}{\partial w} &= -\frac{2}{5} \cdot 61.6 = -24.64 \end{aligned}$$

Gradient for b:

$$\begin{aligned} \sum (y - y_{\text{pred}}) &= 2.2 + 2.8 + 4.5 + 3.7 + 5.1 = 18.3 \\ \frac{\partial \text{Cost}}{\partial b} &= -\frac{2}{5} \cdot 18.3 = -7.32 \end{aligned}$$

Step - 3 - Update Parameters:

Using the gradient descent update rules:

$$w \leftarrow w - \alpha \cdot \frac{\partial \text{Cost}}{\partial w}, \quad b \leftarrow b - \alpha \cdot \frac{\partial \text{Cost}}{\partial b}$$

Update w:

$$\begin{aligned} w &= 0 - 0.01 \cdot (-24.64) \\ &= 0 + 0.2464 \\ w &= 0.2464 \end{aligned}$$

Update b:

$$\begin{aligned} b &= 0 - 0.01 \cdot (-7.32) \\ &= 0 + 0.0732 \\ b &= 0.0732 \end{aligned}$$

Step - 4 - Compute Updated Predictions:

The updated predictions are:

$$\begin{aligned} y_{\text{pred}} &= w \cdot x + b \\ &= [0.2464 \cdot 1 + 0.0732,\ 0.2464 \cdot 2 + 0.0732,\ 0.2464 \cdot 3 + 0.0732,\ 0.2464 \cdot 4 + 0.0732,\ 0.2464 \cdot 5 + 0.0732] \\ &= [0.3196, 0.566, 0.8124, 1.0588, 1.3052] \end{aligned}$$

Step - 5 - Compute Cost:

The cost function is:

$$\text{Cost} = \frac{1}{n}\sum(y - y_{\text{pred}})^2$$

errors:

$$\begin{aligned} y - y_{\text{pred}} &= [2.2 - 0.3196,\ 2.8 - 0.566,\ 4.5 - 0.8124,\ 3.7 - 1.0588,\ 5.1 - 1.3052] \\ &= [1.8804, 2.234, 3.6876, 2.6412, 3.7948] \end{aligned}$$

Squared errors:

$$\begin{aligned} (y - y_{\text{pred}})^2 &= [1.8804^2, 2.234^2, 3.6876^2, 2.6412^2, 3.7948^2] \\ &= [3.5379, 4.992, 13.6005, 6.976, 14.398] \end{aligned}$$

Sum of Squared errors:

$$\sum(y - y_{\text{pred}})^2 = 3.5379 + 4.992 + 13.6005 + 6.976 + 14.398 = 43.5044$$

Cost:

$$\text{Cost} = \frac{1}{5} \cdot 43.5044 = 8.7009$$

Final Summary After one Epoch:

After one epoch:

- Updated parameters: $w = 0.2464$, $b = 0.0732$
- Updated predictions: $[0.3196, 0.566, 0.8124, 1.0588, 1.3052]$
- Cost: 8.7009

Your Turn:

Repeat for second epochs.

This process can be repeated for additional epochs until convergence.

2.8 Gradient Descent in Vector - Matrix Notation {Optional}:

The starting point is the formula for the cost function $E(w)$:

$$E(w) = \frac{1}{n}(y - Xw)^T(y - Xw)$$

which expands to:

$$E(w) = \frac{1}{n}\left(w^T X^T X w - 2w^T X^T y + y^T y\right) \tag{1}$$

We can find a closed form for its gradient $\nabla E(w)$:

$$\nabla E(w) = \frac{1}{n}\left(2X^T X w - 2X^T y\right)$$

which simplifies to:

$$\nabla E(w) = \frac{2}{n}X^T(Xw - y) \tag{2}$$

Gradient Descent Algorithm

The gradient descent algorithm iteratively updates the weights w.

Initialization: Start with an initial vector:

$$w^{(0)} = \left[w_0^{(0)}, w_1^{(0)}, \ldots, w_p^{(0)}\right]$$

Update Rule: Repeat the following steps for $s = 0, 1, 2, \ldots$ until convergence:

$$w^{(s+1)} = w^{(s)} - \alpha \nabla E(w^{(s)})$$

Substituting the gradient $\nabla E(w^{(s)})$ from Equation (2):

$$w^{(s+1)} = w^{(s)} - \alpha \cdot \frac{2}{n}X^T(Xw^{(s)} - y) \tag{3}$$

Convergence: The algorithm converges when there is little change between successive iterations:

$$\|w^{(k+1)} - w^{(k)}\| < \epsilon$$

where k is the current iteration, and ϵ is a small threshold. At convergence, the solution is:

$$w^* = w^{(k+1)}$$

3 Model Evaluation Metrics: Accessing Goodness of Fit.

After estimating a model, it is crucial to assess how well it performs. One fundamental question is whether the model provides better predictions than simply predicting the average value for everyone. To answer this, we evaluate the model's performance using specific metrics that quantify prediction errors.

1. Sum of Squared Errors (SSE) and Mean Squared Error (MSE)

A natural measure of model performance is its loss or the squared error, which quantifies the discrepancy between observed and predicted values.

Sum of Squared Errors (SSE):

SSE grows with the number of data points, so a large SSE may indicate poor model fit or simply a large dataset.

Measured in squared units of the dependent variable (e.g., if is in dollars, SSE will be in dollars-squared), which makes it hard to interpret.

Mean Squared Error (MSE):

Averaging over the number of data points resolves the dependency of SSE on dataset size.

Still measured in squared units, which remains difficult to interpret.

2. Root Mean Square Error:

To make the metric more interpretable, we use the Root Mean Square Error (RMSE):

Key Features of RMSE:

Same Units as : RMSE is expressed in the same units as the dependent variable, making it easier to interpret. Sensitivity to Large Errors: RMSE penalizes large errors more than MSE, making it effective at highlighting poor predictions.

Challenges of RMSE:

- RMSE is difficult to compare across datasets with different units or scales.
- When variables have inherently different units (e.g., income vs. temperature), RMSE cannot determine which model performs better.

2. R-Squared Error:

To address the challenges of scale and unit dependency, we use **R-squared (R^2)**, which measures the proportion of variation in the dependent variable explained by the model.

$$R^2 = 1 - \frac{SS_{\text{Res}}}{SS_{\text{TOT}}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

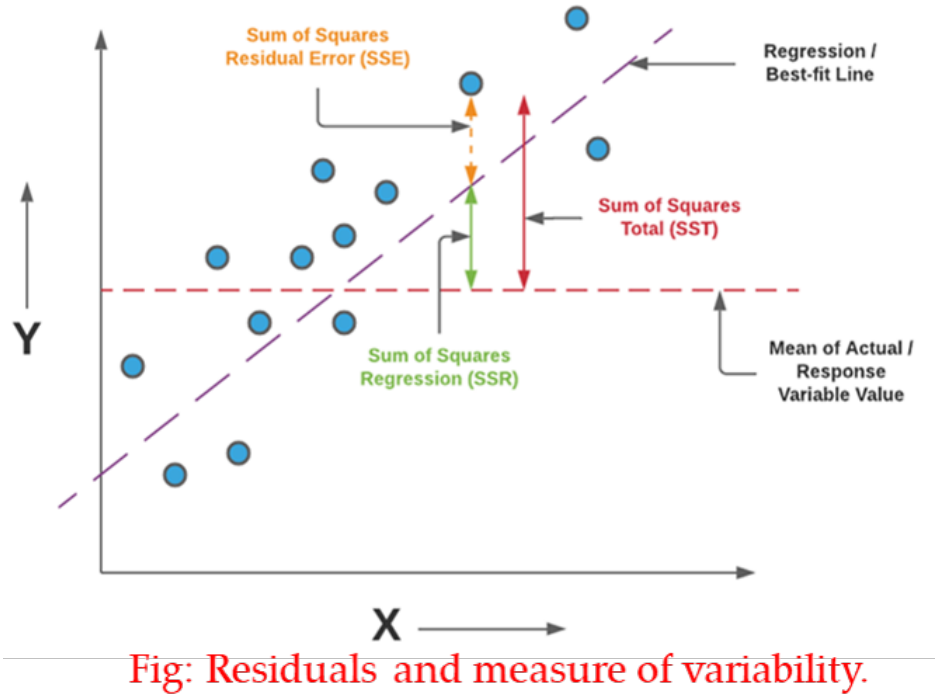

Figure 10: Understanding the Residuals.

Definitions of Components

- **Residual Sum of Squares (SS_{Res}):** The Residual Sum of Squares (SS_{Res}) represents the variation in the dependent variable that is not explained by the model. It is the sum of the squared differences between the observed values (y_i) and the predicted values ($\hat{y}_i$) for each data point. Mathematically, it is expressed as:

$$SS_{\text{Res}} = \sum_i (y_i - \hat{y}_i)^2$$

- **Total Sum of Squares (SS_{TOT}):** The Total Sum of Squares (SS_{TOT}), also known as the **Total Variation**, measures the total variability of the observed values in relation to their mean value ($\bar{y}$). It quantifies how much the observed values differ from the mean of the dependent variable. Mathematically, it is expressed as:

$$SS_{\text{TOT}} = \sum_i (y_i - \bar{y})^2$$

Interpretation of R^2

- Values range from **0 to 1**.
- A value closer to **1** indicates that the model explains most of the variation in the data.
- A value near **0** suggests that the model fails to explain the variability and performs no better than the mean.

Important Note: Despite its name, R-squared (R^2) is **not the square of the correlation coefficient (R)**.

Limitations of R^2

- R^2 does not account for overfitting; adding more variables to the model can increase R^2 even if those variables are not meaningful.
- It cannot distinguish between correlation and causation, making it less informative when key explanatory variables are omitted.

——————— Towards Handling Overfitting. ———————