6CS012 – Artificial Intelligence and Machine Learning.
Lecture – 02
Understanding the Components of Learning.
A Classification Perspective.

Siman Giri {Module Leader – 6CS012}

Learning Outcomes!!

- To **revise** the various **components** of (Machine) **Learning** that we discuss **@5CS037**.
- To **review and re-familiarize** above mentioned **components** with the context of
  - **Classification task** → "**Logistic Regression**".
    - By the end of the week Understand the limitation of Logistic Regression and Challenges of Machine Learning and
      - Justify the need for deep learning.

1. Understanding a Machine Learning Problem.

{Components of a Machine Learning System.}

1.1 What is Machine Learning?

- Machine/Deep learning is a sub-domain of artificial intelligence (AI) that utilizes **Statistics, Pattern recognition, knowledge discovery and data mining** to **automatically learn and improve with experiences** without **being explicitly programmed**.
- **Disclaimer!!!** "In Machine/Deep Learning we do not write a program to solve a specific problem or task instead we write a code/program to facilitate machine to learn from the data."
- A machine learning algorithm learns from the training data:
  - Input: Training Data (e.g., emails x and their labels y)
  - Output: A prediction function that produces output y given input x
- Almost any application that involves **understanding data** that come from the real world can be best **addressed using machine learning**.
  - Great examples are image classification , object detection and many kinds of language-processing tasks.

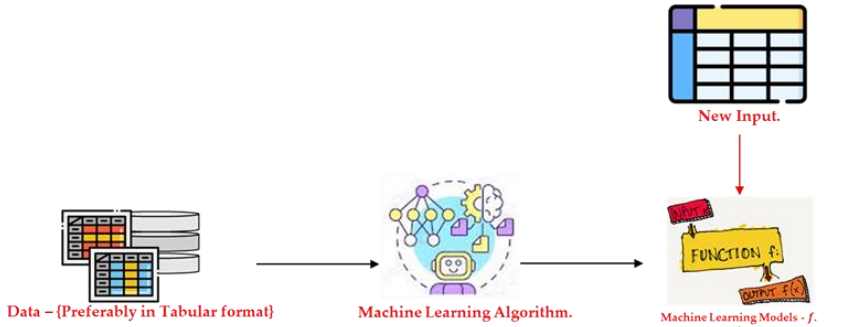


Fig: A Machine Learning System for Prediction.

1.1.1 What isn't Machine Learning?

- **What isn't it?**
  - It is not artificial intelligence:
    - At least, not exactly. Though they are in **a relationship**. ("**It's complicated**")
    - Provocatively: it's the **bit of AI** that works
- It's not on the verge of ending civilization in a **robot singularity**.
- More prosaically: it isn't **always the right tool** for the job
  - Excels for **well-defined questions** with **densely sampled data**
  - Not good at **abstract reasoning** — mostly does not even **operate in that space**
  - Avoid the human trap of imagining an **ML model understands anything**
- Machine learning is a very general and useful framework, but it is not **"magic"** and may not always work.
  - In order to better understand when it will and when it will not work, it is useful to **formalize** the **learning problem** more.

1.2 Components of a Learning System.

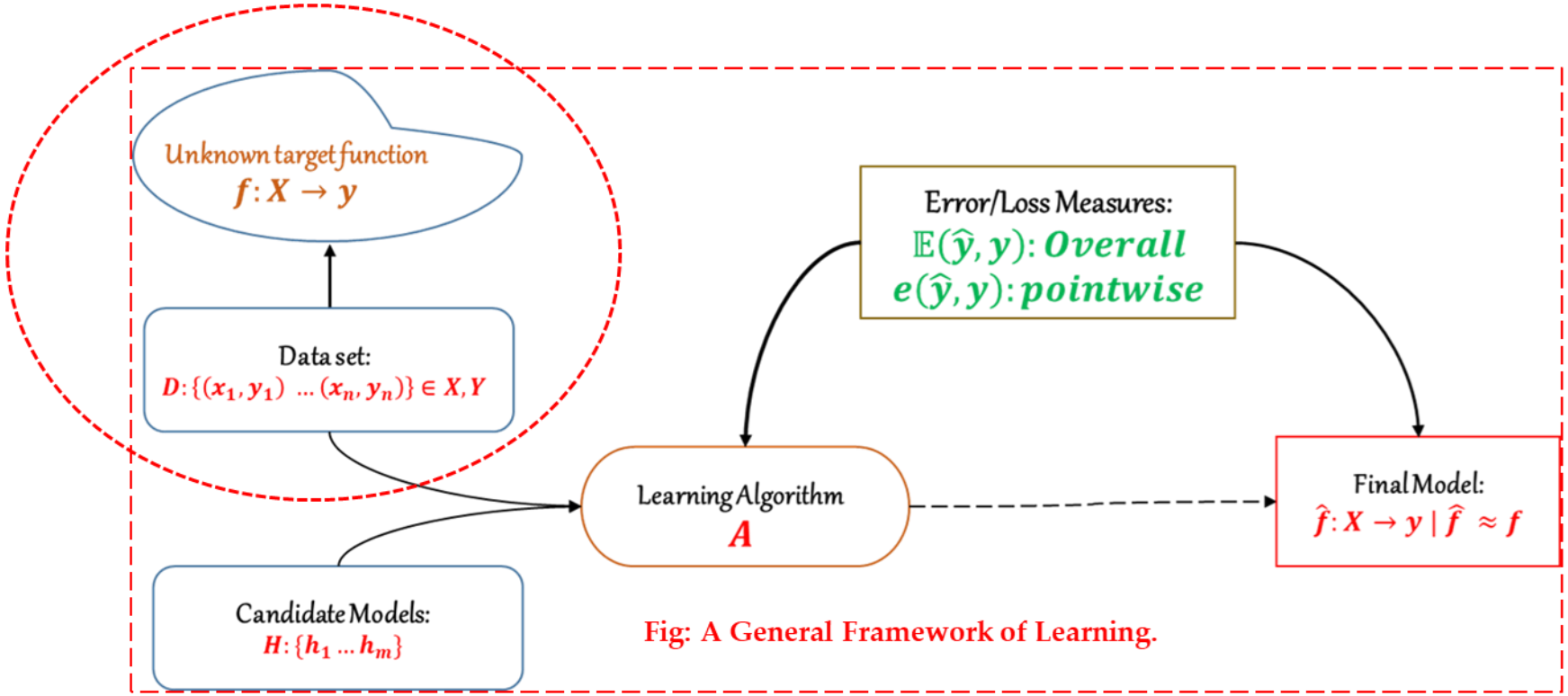


Fig: A General Framework of Learning.

Let's Dissect and understand each Components!!!

1.3.1 Data and Problem Class.

- There are many different problem class in a machine learning which vary according to what kind of data is provided and what kind of conclusions are to be drawn from it.
- The problem class could be broadly classified as:
  - **Supervised Learning:**
    - The idea of supervised learning is that the learning system is given inputs and told which specific outputs should be associated with them.
    - We expect machine to learn the function which maps input to its associated label.
  - **Unsupervised Learning:**
    - Unsupervised learning doesn't involve learning a function from inputs to outputs based on a set of **input-output pairs**.
    - Instead, one is given a **data set** and generally expected to find **some patterns or structure** inherent in it.
  - **Reinforcement Learning:**
    - In reinforcement learning, the goal is to learn a mapping from input values (typically assumed to be states of an agent or system; for now, think e.g. the velocity of a moving car) to output values (typically we want control actions; for now, think e.g. if to accelerate or hit the brake).

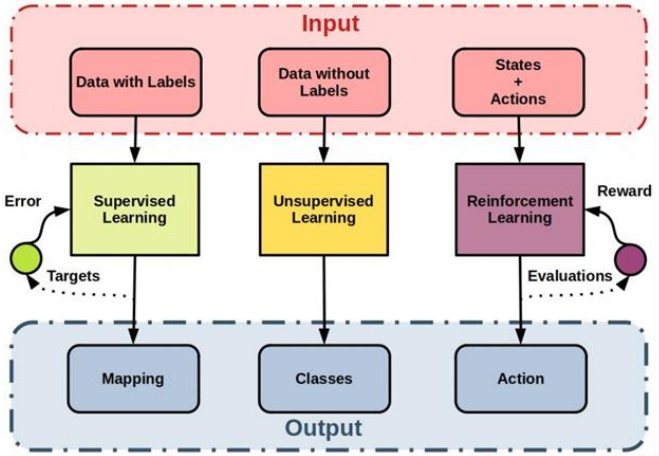
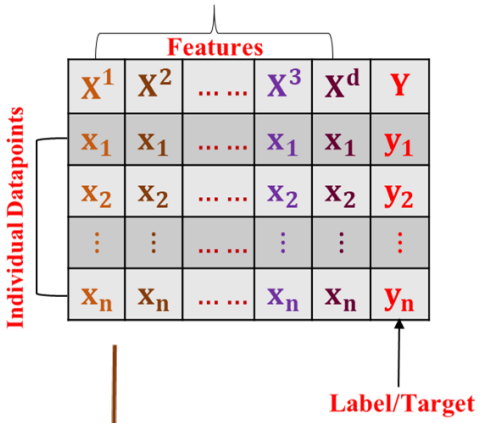


Image from the Internet: Subject to copyright.

Fig: Different Problem Class in Machine Learning.

1.3.2 Data and Data formats.

- Some terminology associated with dataset in practice:
- **Variables:**
  - Target or output variables also referred as dependent variables.
    - In General, we denoted as $\mathbf{X_{n\times d}}$ → also called Feature Matrix.
  - Predictor, Feature or input variables also referred as independent variables.
    - In General, we denoted as:
      - For actual target variable or target variable from Dataset:
        - $\mathbf{Y_n}$: $[\mathbf{y_i} \dots \mathbf{y_n}]$ → label vector.
      - For predicted target variable i.e. label your model assign to new input data:
        - $\hat{Y}_n$: $[\hat{y}_i \dots, \hat{y}_n]$ → predicted label vector
- In General Data is denoted as {caution row of x must be same as row of y i.e. n == n}:
  - $\mathfrak{D}$: $\{\mathbf{X_{n\times d}}, \mathbf{Y_n} : (\mathbf{x_1}, \mathbf{y_1}), (\dots), (\mathbf{x_n}, \mathbf{y_n})\}$



$$\text{Feature Matrix } [\mathbf{X_{n\times d}}] \rightarrow \begin{bmatrix} x_1^1 & \cdots & x_1^d \\ \vdots & \ddots & \vdots \\ x_n^1 & \cdots & x_n^d \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \leftarrow [\mathbf{Y_n}] \text{ label/target vector.}$$

Fig: Matrix Representation of our input dataset $\mathfrak{D}$

1.3.3 Data, Problem Class: Supervised Learning.

- **Data in Supervised Learning:**
  - For Supervised Learning Setup, **training data** comes in pairs of inputs **(x, y):** where $\mathbf{X} \in \mathbf{R^d}$ is the input instance and **Y** its label, which can be written as:
    - $\mathfrak{D} = \{(\mathbf{x_1}, \mathbf{y_1}) \dots (\mathbf{x_n}, \mathbf{y_n})\} \subseteq \mathbf{R^d} * \mathbf{C}$
    - Where:
      - $\mathbf{R^d}$: **d-dimensional feature space.**
      - $\mathbf{x_i}$: **input vector of the i^{th} sample.**
      - $\mathbf{y_i}$: **label of the i^{th} sample.**
      - **C: label space.**

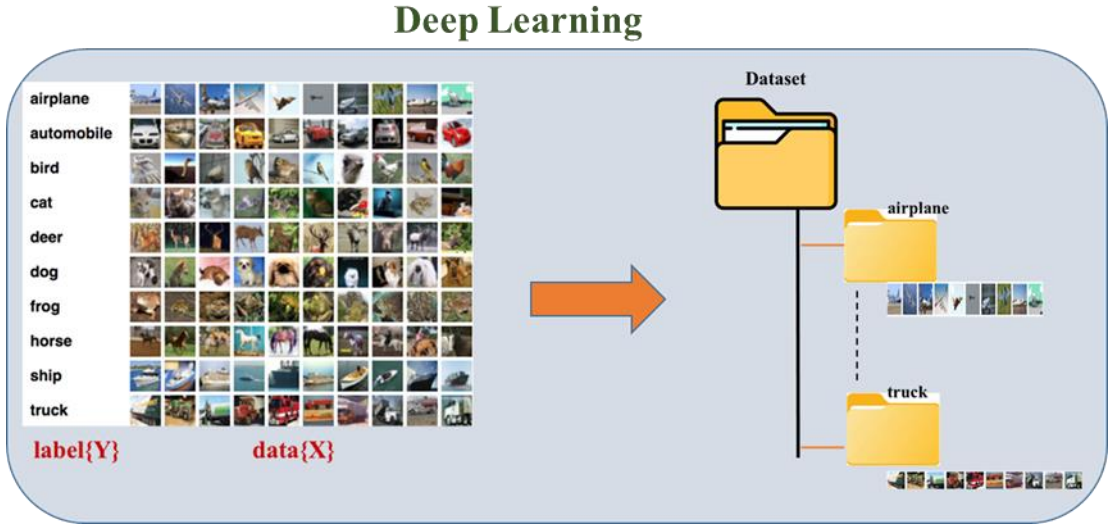
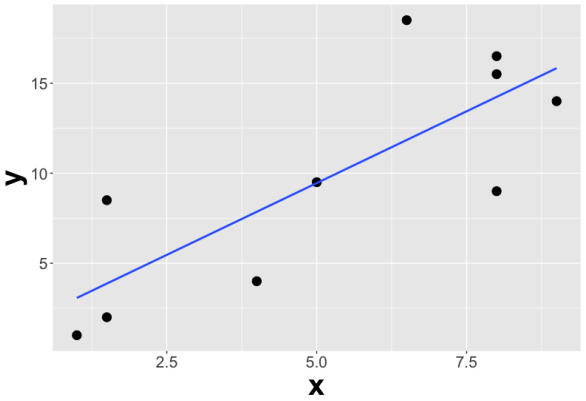


Fig: Example Dataset for Image Classification.

1.3.3.1 Supervised Learning: Task.

Regression

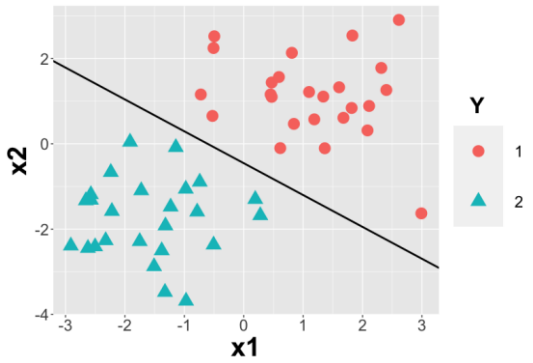
- Target is numerical, i.e
  - $\mathbf{Y \in C = \mathbb{R}}$
- e.g., predict days a patient has to stay in hospital



X_1	X_2	X_3	X_p	Y
				5.2
				1.3
				23.0
				7.4

Classification

- Target is categorical, i.e. $\mathbf{Y \in C = \{1, 2, \ldots K\}}$
  - If
    - **K = 2 i. e. two class 0 or 1 → Binary Classification.**
  - else:
    - **K ≥ 2 upto K → Multiclass Classification.**
- e.g., predict one of two risk categories for a life insurance customer .

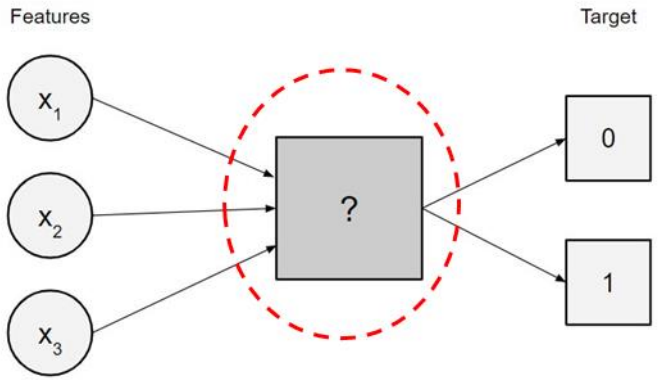


X_1	X_2	X_3	X_p	Y
				cat
				dog
				cat
				cat

1.3.4 To Summarize: Data in Supervised Learning.

- In formal notation datasets are given and are in the following form:
  - $\mathfrak{D} = \left((\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})\right) \in (\mathbf{X} \times \mathbf{Y})_\mathbf{n}$
- We call:
- $\mathbf{X_n}$: the input space define by the dimension
  - $\mathbf{d} = \mathbf{dim(X)}$,
  - Thus, the feature matrix $\mathbf{X}$ is $\mathbf{X_{n \times d}}$ and
    - $\mathbf{x_j} = \left(\mathbf{x_j^1}, \ldots, \mathbf{x_j^d}\right)$ is the $\mathbf{j^{th}}$ feature vector.
- $\mathbf{Y_n}$: the target or label vector.
  - $\mathbf{Y_n} \in \mathbf{C}$ & **if** $\mathbf{C} = \mathbb{R} \rightarrow$ Regression Task.
  - $\mathbf{Y_n} \in \mathbf{C}$ & **if** $\mathbf{C} = \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots \mathbf{K}\} \rightarrow$ Classification Task.

- In Supervised Learning:



- We assume some **kind of relationship** between the **features and the target**,
  - in a sense that the value of the target variable can **be explained** by a combination of the features.

1.2 Components of a Learning System.

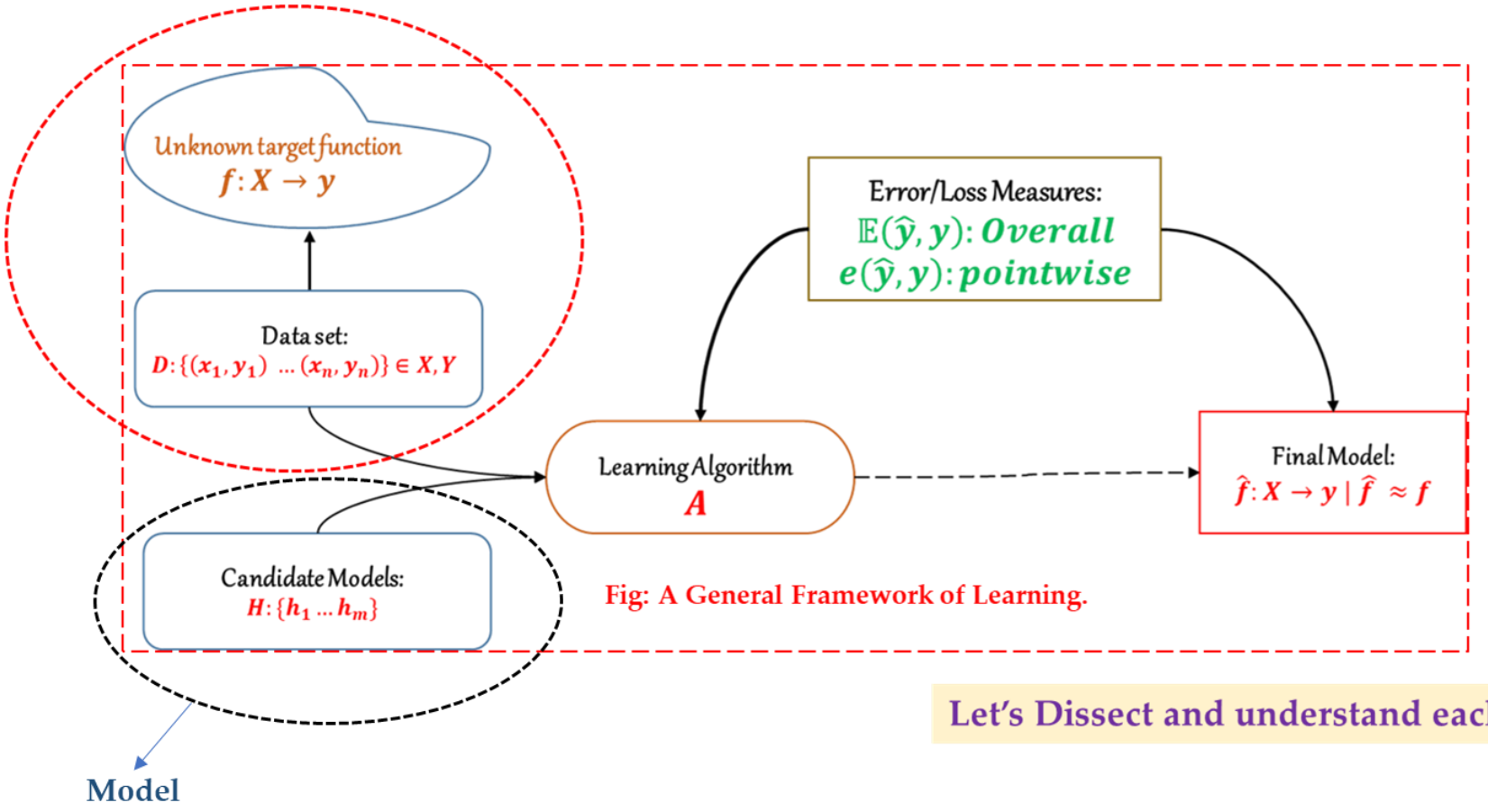


Fig: A General Framework of Learning.

Let's Dissect and understand each Components!!!

1.4 Machine Learning Model.

- Given some potentially multi– valued input $\mathbf{x_j} = [\mathbf{x_j^1}, \dots, \mathbf{x_j^d}]$ is the $\mathbf{j^{th}}$ feature vector.
- **Predict** a potentially multi – valued output $\mathbf{y} = [\mathbf{y_1}, \dots, \mathbf{y_m}]$
  - {Collectively, **x and y** are the **model variables}.**
- Formally for Supervised Learning Setup we can define model as:

A **model (or hypothesis)** is a **function** that **maps** feature vectors to **predicted** target values:
f: $\mathcal{X} \rightarrow \hat{\mathcal{y}}$

x_1	x_2
Large	300.50
Medium	2300

→ **Model: f** →

$\hat{y}$
2040
4567

f : is meant to capture **intrinsic patterns of the data**, the **underlying assumptions** being that these hold true for all data drawn from $\mathbb{P}_{\mathcal{X},\mathcal{y}}$.

Q. How do we select the Model?

1.4.1 Model Selection i.e. $\mathcal{M} \in \mathcal{H}$.

- How do we select a model from model class: $\mathcal{H} \in \mathcal{M}$?
  - 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.
- This restricted set of functions defining a specific model class is called a **hypothesis space**:
  - $\mathcal{H} = \{\mathbf{f}: \mathbf{f}$ **belongs to a certain functional family**$\}$

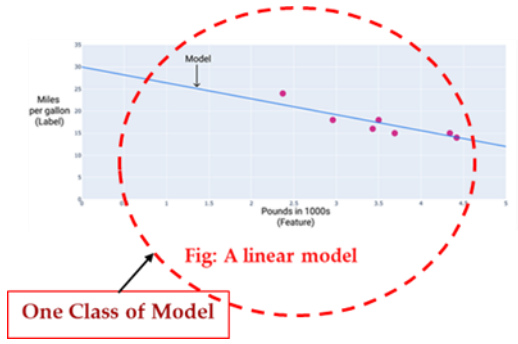


Fig: A linear model

Fig: A Rule based model

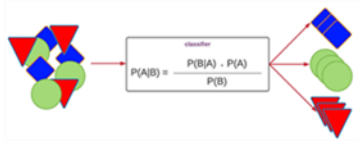


Fig: A Probabilistic Space

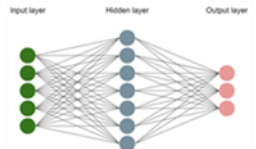


Image compiled from the Internet: Subject to copyright.

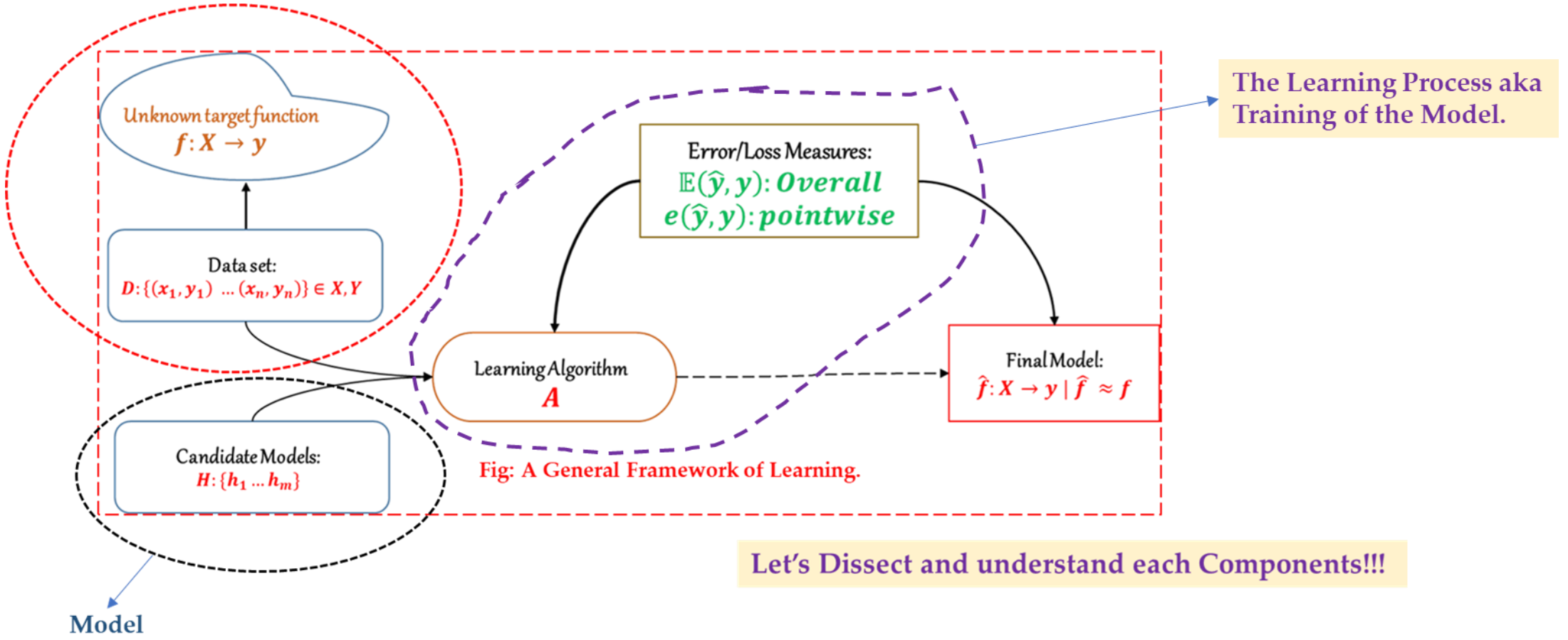
Fig: A Deep Neural Network

Fig: Candidate Models in Hypothesis Space

1.4.2 Models and Parameters.

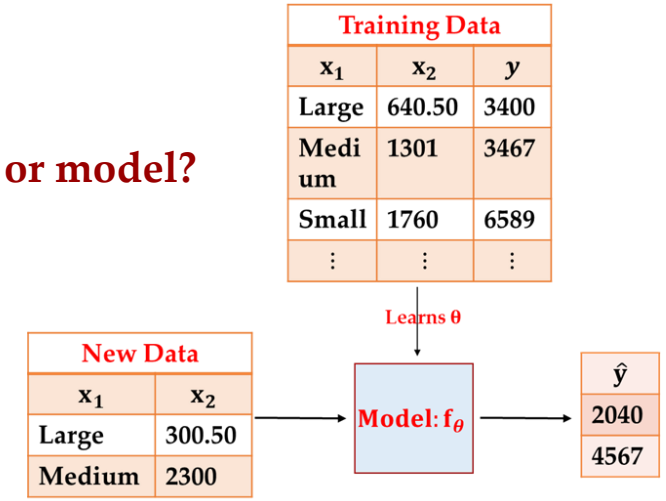
- **Models and Parameters:**
  - All models **within one hypothesis space** share a common functional structure i.e.
    - they are fully defined by some properties let's call it parameter and
      - let's denote with $\boldsymbol{\theta \in \Theta}$: **{Parameter Space}**
        - {in Machine Learning we normally use **w** also called weights}.
  - Functions are fully determined by parameters. Thus, we can re-write
    - $\boldsymbol{\mathcal{H} = \{f_\theta : f_\theta}$ **belongs to certain functional family parameterized by** $\boldsymbol{\theta\}}$
      - E.g., in the case of linear functions, $\boldsymbol{y = \theta_0 + \theta_1 x}$,
        - the parameters $\boldsymbol{\theta_0}$ **(intercept)** and $\boldsymbol{\theta_1}$ **(slope)** determine the relationship between **y and x**
        - We collect all parameters in a parameter vector
          - $\boldsymbol{\theta = [\theta_1, \theta_2, \dots, \theta_n]}$ from **parameter space** $\boldsymbol{\Theta}$
  - Finding the optimal model means **finding the optimal or best set of parameters** for chosen function from the model class $\mathcal{M}$.
  - **In supervised learning;**
    - finding the best set of parameters usually means fitting or training a model on labeled training dataset
      - such that best set of parameters could be determined.

1.2 Components of a Learning System.



1.5 The Training of Machine Learning Model.

- Learning Process aka parameter fitting or most commonly called Model training defines the task of:
  - Finding the optimal model means **finding the optimal or best set of parameters** for chosen function from the model class $\mathcal{M}$.
- In **supervised learning**;
  - finding the best set of parameters usually means fitting or training a model on labeled training dataset
    - such that best set of parameters could be determined.
- The question to be asked here is:
  - **How do we fit or train the model so we get best set of parameters?**
  - **How do we determined which set of parameters best fits our function or model?**



1.6 Elements of a Learning Process.

- Governed under the **framework** of **Empirical Risk Minimization (ERM) {learning theory}**, the following are the key elements of a supervised machine learning process:
  - **Decision Process/Function (Representation/Model):**
    - Machine learning models are used to **infer or estimate** outputs based on input data.
    - Input data can be **labeled** (e.g., supervised learning) or **unlabeled** (e.g., unsupervised learning).
  - **Error Function (Evaluation):**
    - A performance metric evaluates how well the model's predictions align with the actual outcomes.
    - The choice of metric depends on:
      - **Learning type: Supervised or unsupervised.**
      - **Task type: Classification (e.g., accuracy) or regression (e.g., mean squared error).**
  - **Model Optimization Process:**
    - An iterative algorithm updates the **model's parameters** to **minimize** the **error function**.
    - Optimization continues until a specified threshold or an acceptable evaluation metric is achieved.
    - Common methods include **gradient descent** and its **variants**.

1.6.1 Model as a Decision Function.

- A decision function (aka prediction function) gets **input x** $\in \mathcal{X}$ and produces **an action a** $\in \mathcal{A}$: i.e.
  - **f**: $\mathcal{X} \rightarrow \mathcal{A}$ where:
    - $\mathbf{x} \longmapsto \mathbf{f}(\mathbf{x})$
- In the context of Supervised Machine Learning:
  - Action **a** $\in \mathcal{A}$ depends on label $y \in \mathcal{Y}$ **i.e. if**
    - $\mathcal{Y} \in \mathbb{R}$ action → predict the value of label → Regression Task.
    - $\mathcal{Y} \in \mathcal{C}$: $\mathcal{C} = \{0, 1, \dots, K\}$ action → assign a class to a label → Classification Task.

A **model {or hypothesis)** is a **function** that **maps** feature vectors to **predicted** target values:
f: $\mathcal{X} \rightarrow \hat{\mathcal{Y}}$

x_1	x_2
Large	300.50
Medium	2300

→ **Model: f** →

$\hat{y}$
2040
4567

f : is meant to capture **intrinsic patterns of the data**, the **underlying assumptions** being that these hold true for all data drawn from $\mathbb{P}_{x,y}$.

1.6.2 Learn the Parameters: Error Metrics.

- **Goal:** Find the best parameters of a model that best fits training data.
- **Intuition:** If we can evaluate how good a prediction function is we can turn this into an optimization problem.
- The quality of predictions from a learned model is often expressed in terms of a loss function.
- We specify evaluation criteria at two levels:
  - **how an individual prediction is scored → pointwise loss, and**
  - **how the overall behavior of the prediction or estimation system is scored → average loss or Risk.**
    - "Cautions!! Risk is a Theoretical concepts backed by **Statistical Learning Theory**, Let's formally define Risk first!!!!!"

1.6.3 Statistical Learning Theory.

- Define a **space** where a decision or prediction function is applicable:
  - Assume there is a data generating distribution $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$.
  - All input/output pairs (x, y) are generated i.i.d from $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$.
  - One common expectation is to have prediction function $\mathbf{f(x)}$ "that does well on average" i.e. :
    - $\boldsymbol{\ell(\mathbf{f(x)}, \mathbf{y})}$ is usually small.

Risk – Definition:
The **risk** of a prediction function $\mathbf{f}: \mathcal{X} \rightarrow \mathcal{A}$ is:

$$\mathbf{R(f)} = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathbb{P}(\mathcal{X}\times\mathcal{Y})}[\boldsymbol{\ell}(\mathbf{f(x)}, \mathbf{y})]$$

In words, it's the expected loss of **f over** $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$.

- We can not compute the risk function, because we do not know the theoretical distribution or **true** $\mathbb{P}_{\mathcal{X}\times\mathcal{Y}}$ hence expectation could not be computed.
  - But in **Machine Learning**/statistics or data science we can **estimate** it based on our empirical observation or sampled data.

1.6.3.1 Empirical Risk.

- Empirical risk is an estimated risk of theoretical risk, and computed on provided set of sample data.
- Thus, for any sample data:
  - Let $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be drawn **i. i. d** from $\mathbb{P}_{x \times y}$.
- Empirical Risk Could be defined as:

Empirical Risk – Definition:
The **empirical risk** of a prediction function $f: \mathcal{X} \to \mathcal{A}$ with respect to $\mathcal{D}_n$ is:

$$\hat{R}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Backed by strong law of large numbers, which gives:

$$\lim_{n\to\infty} \hat{R}_n(f) = R(f)$$

- Thus, the empirical risk is average loss on our sampled data, we expect it to be as minimum as possible,

1.6.4 Empirical Risk.

- Thus, we want function **f** described by set of parameters with $\theta^* \in \Theta$ which produces the **minimum loss** on **average**.
- So, Remember the question:
  - 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.
  - 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 the minimum risk(estimated), which is an average loss on all our sampled data points.
- **The answer could be collectively described by → Empirical Risk Minimization.**

1.6.5 Empirical Risk Minimization.

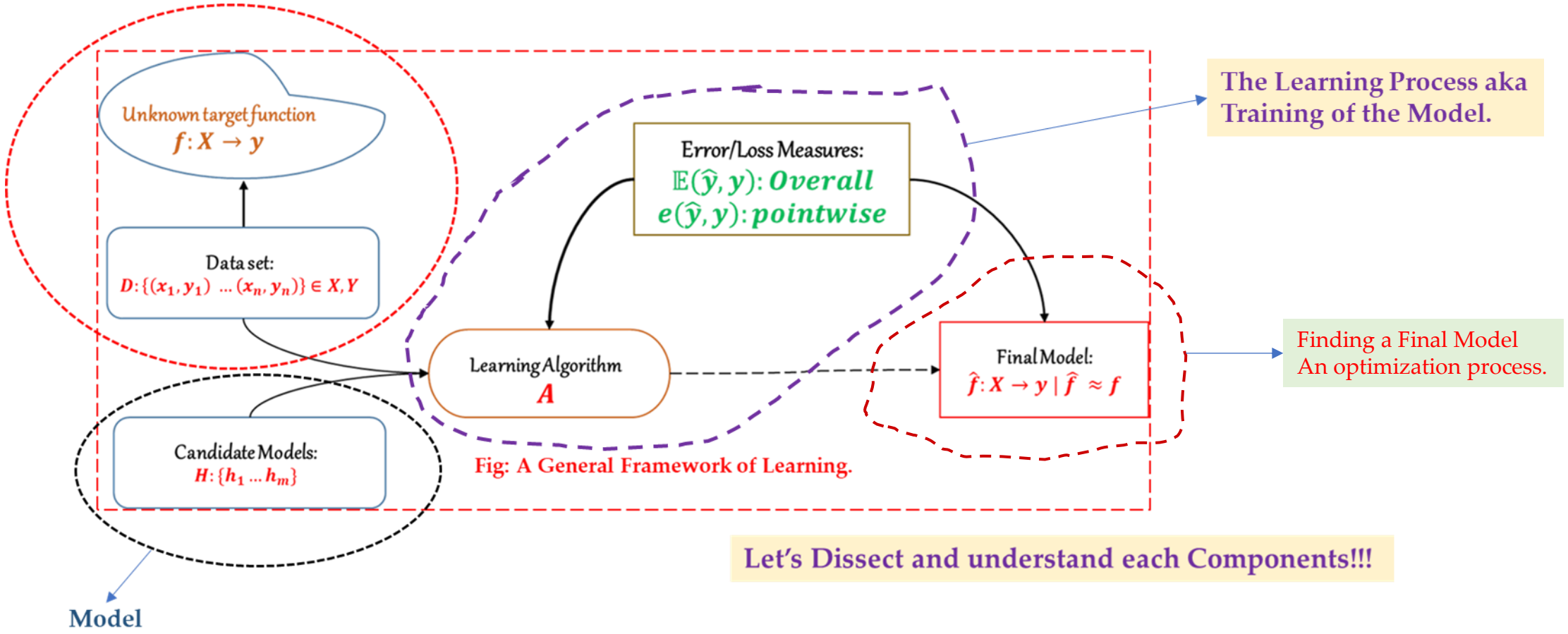
- Among a collection of candidate function $\mathbf{f}$
  - described by set of parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$
  - we desired to select a function $\mathbf{f}$ learned from sample data $\mathcal{D}_n\{(\mathbf{x_i}, \mathbf{y_i})\}$
  - which is described by parameter $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}$ and
  - produces minimum average loss – Risk $\widehat{\mathbf{R}}(\mathbf{f})$.
- Formally:

Empirical Risk Minimizer – Definition:
A function $\hat{\mathbf{f}}$ is an empirical risk minimizer if:

$$\hat{\mathbf{f}} \in \mathbf{argmin}_{\mathbf{f}}\, \widehat{\mathbf{R}}_{\mathbf{n}}(\mathbf{f}),$$

where the minimum is taken over all functions $\mathbf{f}: \mathcal{X} \rightarrow \mathcal{A}$.

1.2 Components of a Learning System.



1.7 An Optimization Process.

- How do you find such a function:
  - $\hat{\mathbf{f}} \rightarrow \mathbf{argmin}\ (\mathbf{f})$ is an Optimization problem
  - i.e. we can find an empirical risk minimizer $\hat{\mathbf{f}}$ by minimizing a **loss function** ℓ also called objective function.
- Machine Learning as an Optimization problem could be defined as:
  - For a given **loss function** $\ell(\boldsymbol{f}(\boldsymbol{x_i}, \boldsymbol{y_i}))$, we want to find **set of parameters** $\boldsymbol{\theta}^* \in \boldsymbol{\Theta}$ that produce a minimum loss value which can be written as:
    - $\boldsymbol{\theta}^*_{\in \boldsymbol{\Theta}} = \mathbf{argmin}\boldsymbol{\mathcal{L}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}, \mathbf{f})$
- How do we find such a parameter?

1.7.1 An Iterative Approach.

- In a nutshell, Numerical Solution are iterative systematic search:
  - If the loss is (more or less) differentiable, we can find the local gradient,
    - Unless we're at a minimum, destination must be further down*
      - So, keep taking small steps down the gradient until we get there
        - "**Gradient Descent Algorithm**"
- Iterative or numerical methods (e.g., gradient descent) systematically refine the solution by leveraging gradient information or other feedback, improving precision and convergence to an optimal solution.
- Iterative methods focus computations on regions of interest and adaptively refine the search, reducing unnecessary evaluations and overcoming inefficiency.

1.8 So, How Good is Your Model?

- If you find a function $\mathbf{f_\theta^*}$ with low loss on your data $\mathcal{D}_\mathbf{n}(\mathbf{x_i}, \mathbf{y_i})$,
  - **how can we tell how good is the learned model?**
- In practice:
  - The performance metrics are evaluation measure also called **measure of error** are used to tell how well you learned the model parameter or how best it fitted your training data.
    - **Various** kind of performance metrics are available based on **various** task Regression or Classification.
    - We can also use loss function as measure of evaluation for some task.
  - We will discuss more on this when we discuss various model in upcoming weeks, but the question we are exploring today is:
- For any evaluation metric used,
  - How could you say model learned from some sampled training data will do well on real world data which are not in sampled data?
    - "**how do you** know it will **make a correct prediction** on **new data** which are not in **training data** $\mathcal{D}_\mathbf{n}(\mathbf{x_i}, \mathbf{y_i})$?"
    - How well your model generalize?

1.9 ML in Practice: Applied ML.

- A machine learning problem could be defined with:
  - Observed input $\mathbf{x} \in \mathcal{X} \rightarrow$ **Input Space**.
  - Take action $\mathbf{a} \in \mathcal{A} \rightarrow$ **Action Space{Regression or Classification}**
  - Observe Outcome $\mathbf{y} \in \mathcal{Y} \rightarrow$ **Outcome Space**.
- Evaluate action in relation to the Outcome using ERM framework i.e.
  - Given a loss function
    - $\ell: \mathcal{A} \times \mathcal{Y} \rightarrow \mathbb{R}$,
  - Choose a hypothesis space $\mathcal{F}$ from $\mathcal{H} = \{\mathcal{F}_{\text{(model)}} \in \mathcal{M}_{\text{class of Models}}\}$
  - Use an optimization method to find an empirical risk minimizer $\hat{f}_n \in \mathcal{F}$:
    - $\hat{f}_n = \boldsymbol{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$
    - (Or find a $\tilde{f}$ that comes close to $\hat{f}$)
- Your job as an ML practitioner:
  - Choose an appropriate Model $\mathcal{F}$ from class of model $\mathcal{M}_{class\ of\ Models}$ and fit model directly and hope model will perform appropriately – ***model fitting problem***;
  - Consider several model from model class and choose the best based on some performance metric – ***model selection problem***.

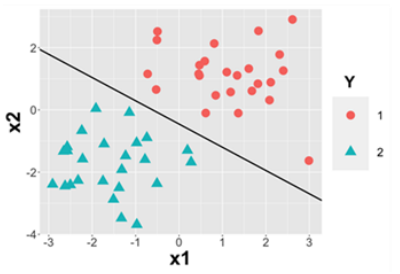
2. Understanding Classification Task.

{Building a Logistic Regression for Classification Task.}

2.1 Definition: Task of Classification.

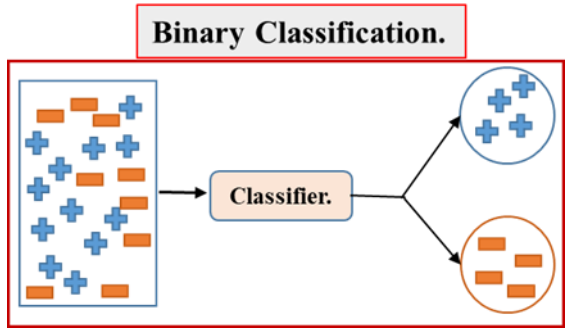
- While for **regression** the model simply maps a independent variable $\boldsymbol{X_{n\times d}}$ to dependent variable $\boldsymbol{Y_n}$ i.e.
  - $\mathbf{f}: \boldsymbol{X}_{\in \mathbb{R}^d} \rightarrow \boldsymbol{Y}_{\in \mathbb{R}}$
- For classification it is slightly more complicated as in classification our dependent variable happens to be **discrete output** called class. i.e.
  - $\mathbf{y} \in \boldsymbol{Y} = \{\mathbf{C_1}, \dots, \mathbf{C_k}\}$
  - Here:
    - $\mathbf{C_1}, \dots, \mathbf{C_k} \Rightarrow$ Discrete Classes **and** $\mathbf{2 \leq k < \infty}$ for any given data $\boldsymbol{\mathcal{D}} = \{(\mathbf{x_i}, \mathbf{y_i}), \dots, (\mathbf{x_n}, \mathbf{y_n})\} \in [\boldsymbol{X_{n\times d}} \times \boldsymbol{Y_n}]$.
- Based on target or dependent variable i.e. $\mathbf{Y \in C = \{1, 2, \dots K\}}$ **; Classification Task can be of two type:**
  - If
    - $\mathbf{K = 2}$ **i. e. two class 0 or 1** $\rightarrow$ **Binary Classification.**
  - else:
    - $\mathbf{K \geq 2}$ upto $\mathbf{K}$ $\rightarrow$ **Multiclass Classification.**

Task of Classification.

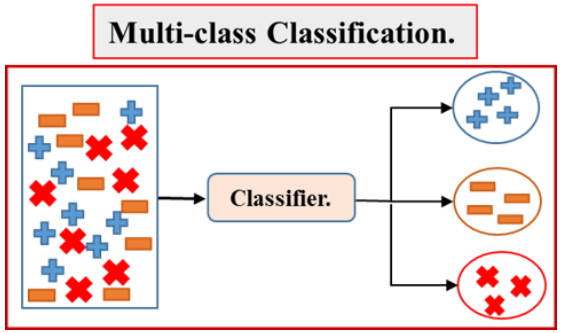


X_1	X_2	X_3	X_p	Y
				cat
				dog
				cat
				cat

2.1.1 Binary Vs. Multi Class Classification.



- For convenience, we often encode these classes differently.
- For **Binary Classification**:
  - $\mathbf{k = 2}$: **Usually use** $\boldsymbol{y = \{0, 1\}}$.



- For convenience, we often encode these classes differently.
  - For **Multiclass Classification**:
    - $\mathbf{k \geq 3}$:
    - Could use $\boldsymbol{y = \{0, 1, \ldots, k\}}$
    - but mostly preferred one hot encoding

i.e. **k-length vector** representation for each class:

- $\boldsymbol{o_i = \mathbb{I}(y = i) \in \{0, 1\}}$.
  - For example:

ID	Features	Species
1	...	Setosa
2	...	Setosa
3	...	Versicolor
4	...	Virginica
5	...	Setosa

One Hot Encoding.

Setosa	Versicolor	Viriginica
1	0	0
1	0	0
0	1	0
0	0	1
1	0	0

2.2 What are Linear Models?

- 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).**
  - **Linear Relationship**: A linear model predicts the output as a weighted sum of the input features:
    - $\mathbf{y = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b}$
    - Where:
      - $\mathbf{y \in \mathbb{R}}$: **is the output(dependent varaible)**,
      - $\mathcal{X} \in \mathbb{R}^{\mathbf{d}}$: **d is the dimension**. {For the purpose of training we are given **n observations.**}
        - $\mathbf{x_1, x_2, \ldots, x_n}$ **are the input features (independnet or explainatory variables)**
      - **b: is the bias(intercept)**
- Linear Models are only useful if there exist a linear relationship i.e. in the context of Classification Problem Linear Models are only useful for linearly separable data.
- Linearly separable data are those data, where there exist a line or hyperplane that can linearly separate the data.

2.2.1 Linear Models for Classification.

- **Can we use linear models for classification?**
  - **Yes!** If there exist a function that can:
    - Takes in the raw output of $\mathbf{w_0} + \langle \mathbf{w^T x} \rangle$ and produces a value strictly **between 0 and 1**.
    - Also **preserves the ordering** of the input values i.e. larger $\mathbf{w_0} + \langle \mathbf{w^T x} \rangle$ implies a higher probability.
- Does there exist such function?
  - Yes, and such function is called a **sigmoid function**.

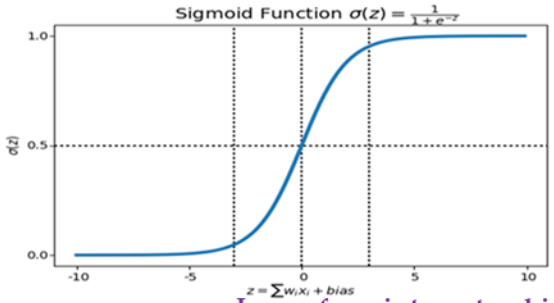


Image from internet: subject to copyright.

Fig: Sigmoid Function (σ).

2.2.2 General Introduction: Sigmoid Function.

- **Logistic/Sigmoid function:**
  - The logistic function $\boldsymbol{\sigma}$ is a function from the real line to the **unit interval (0,1)**
    - $\boldsymbol{\sigma(t) = \frac{1}{1+e^{-t}} = \frac{e^t}{1+e^t} \quad -\infty < t < \infty}$
  - The function maps any real value into another value between 0 and 1.
  - In machine learning, we use sigmoid to map **predictions to probabilities**.
  - Properties:
    - Range: $\boldsymbol{0 < \sigma(t) < 1}$.
    - **Inverse:** $\boldsymbol{t = \sigma^{-1}(p) = \ln\left(\frac{p}{1-p}\right)}$**: logit function.**
    - Derivative: $\boldsymbol{\frac{d}{dt}\sigma(t) = \sigma(t)\big(1 - \sigma(t)\big) = \sigma(t)\sigma(-t)}$

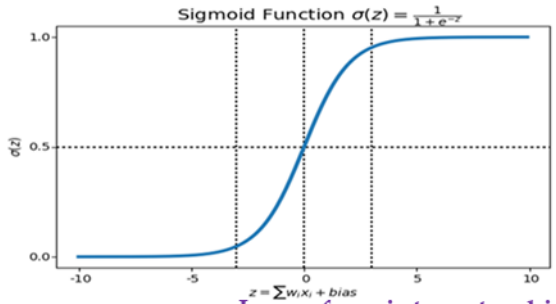


Image from internet: subject to copyright.

Fig: Sigmoid Function (σ).

2.3 Logistic Regression for Binary Classification.

- **Logistic Regression aka sigmoid regression** is a supervised learning problem, in which we are given a training dataset of the form:
  - Given a dataset of n samples: $\mathcal{D} = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) | \mathbf{x_i} \in \mathbb{R}^{\mathbf{d}}, \mathbf{y_i} \in \mathbb{R}\}_{\mathbf{i=1}}^{\mathbf{n}}$ where;
    - $\mathbf{x_i} = [\mathbf{x_{i1}}, \mathbf{x_{i2}}, \dots, \mathbf{x_{id}}]$ is the feature vector for $\mathbf{i^{th}}$sample.
    - $\mathbf{y_i} \in \{\mathbf{0}, \mathbf{1}\}$ is the binary target output for i–th sample
    - $\mathbf{d}$ is the number of features in each input vector,
  - The goal of logistic regression is to model the probability of $\mathbf{y_i = 1}$ given the input vector x_i:
    - $\mathbf{P(y_i = 1|x_i) = f(x_i; w, b) = \sigma(\langle w, x_i \rangle + b)}$
      - Where $\boldsymbol{\sigma(z)}$ is the sigmoid function given:
        - $\boldsymbol{\sigma(z) = \frac{1}{1+e^{-z}}}$;
  - Decision or Prediction function for logistic regression can be written as:
    - we predict the probability of $\mathbf{y = 1}$ as:
      - $\mathbf{P(y = 1|x) = \sigma(\langle w, x \rangle + b)}$
    - To make a binary decision $\mathbf{(y = 1 or\ y = 0)}$, we use the following decision rule:
      - $\hat{\mathbf{y}} = \begin{cases} \mathbf{1\ if\ P(y = 1|x) \geq 0.5} \\ \mathbf{0, if\ P(y = 1|x) < 0.5} \end{cases}$
    - The threshold of 0.5 corresponds to the decision boundary $\mathbf{w_0 + w^T x = 0.}$

Pictorial Representation of a Sigmoid Regression.

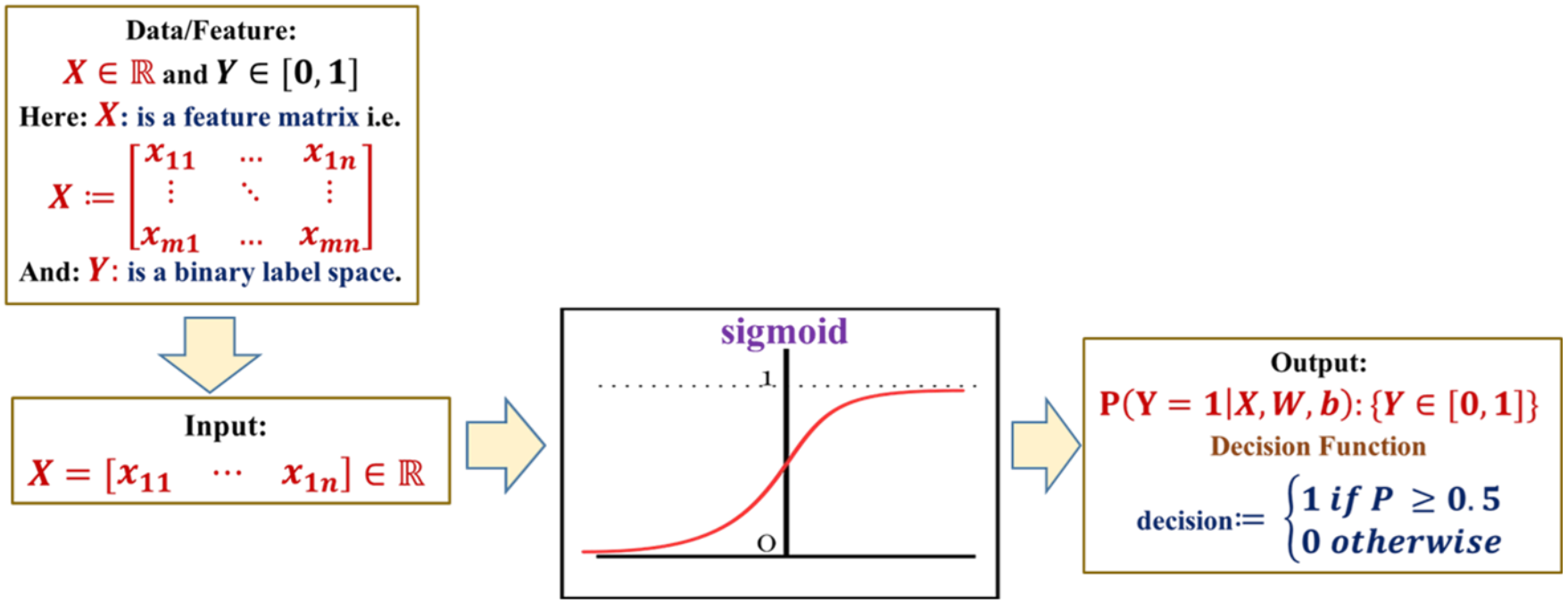


Fig: A General Workflow of Logistic Regression with Sigmoid Function.

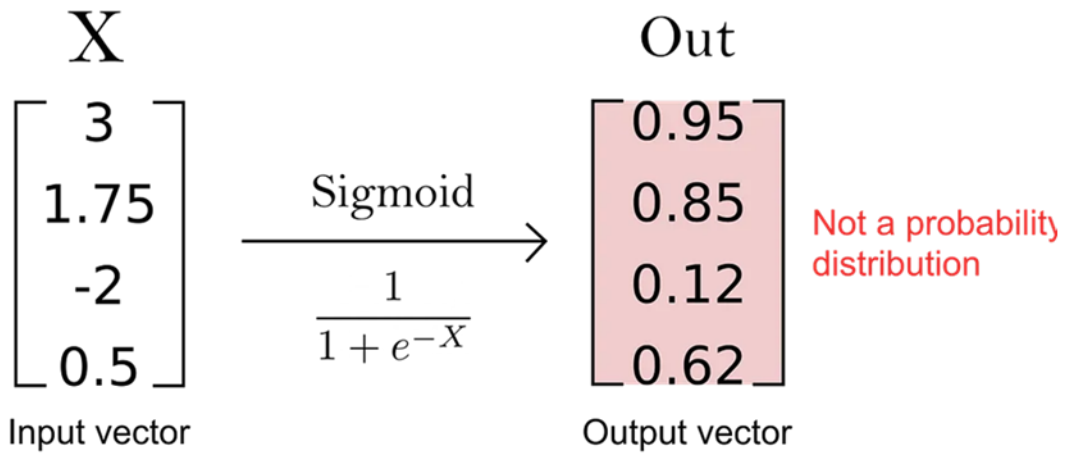
2.4 Logistic Regression for Multi Class Classification.

- **Softmax regression**, also known as **multinomial logistic regression**:
  - Here's how it works:
    - For K classes, the model predicts:
      - $\mathbf{P(y = k|x) = \frac{\exp(\langle w_k, x_i \rangle + b_k)}{\sum_{(j=1)}^{k} \exp(\langle w_j x_i \rangle + b_j)}}$
    - Here:
      - $\mathbf{w_k}$ **and** $\mathbf{b_k}$ $\rightarrow$ **are weight and bias for class k;**
      - $\mathbf{x_i}$ $\rightarrow$ **input feature vector,**
      - $\mathbf{P(y = k|x)}$ $\rightarrow$ **is the probability of class k;**
  - Softmax is a normalized form of sigmoid function:
    - This normalization step allows the model to directly output a set of probabilities that reflect the relative likelihood of each class.

2.5 Sigmoid Vs. Softmax.

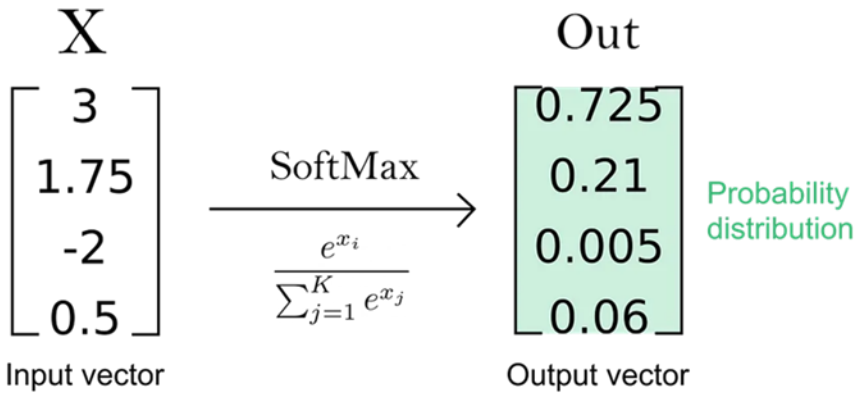
Sigmoid

$$\text{Out} \Rightarrow P(Y = \text{class } 1|X)$$



Softmax

$$\text{Out} \Rightarrow \text{armax} \begin{cases} P(Y = \text{class } 1|X) \\ P(Y = \text{class } 2|X) \\ \vdots \\ P(Y = \text{class } k|X) \end{cases}$$



We will be discussing Multi Class Classification. Thus further discussion will be based on Softmax Regression.

3. Softmax Regression in ERM Framework.

{ERM Framework – Decision Process, Loss Function, Empirical Risk, Optimization}

3.1 Softmax Regression in ERM Framework: Model Definition.

- **Model**: Multiclass logistic regression or softmax regression uses **the softmax function** to compute class probabilities:
  - $\mathbf{f_{w,b} = \hat{y}_i = softmax(z_i)}$;
  - where:
    - $\mathbf{z_i = w^T x_i + b}$,
  - and:
    - $\mathbf{softmax(z_i)_j = \frac{e^{z_{ij}}}{\sum_{k=1}^{C} e^{z_{ik}}}, \forall j \in \{1, \dots, C\}}$.
      - here, $\mathbf{C}$ is the number of classes.
- **Decision Function:**
  - The model's prediction $\mathbf{y_{pred}}$ is the class whose **probability is maximal** i.e.:
  - $\mathbf{y_{pred} = argmax_i P(Y = i | x, W, b)}$

3.2 Softmax Regression in ERM Framework: Loss Function.

- **Loss function – Categorical Cross Entropy:**
  - **Categorical Cross-Entropy Loss** measures how well the predicted probability distribution matches the true class labels in a classification task.
  - It is widely used in multiclass classification problems.
  - The goal of the loss is to maximize the probability of the correct class.
- **Formula**: For a single sample $\mathbf{x_i}$ with true label $\mathbf{y_i}$ (one hot encoded) and predicted probabilities $\hat{\mathbf{y}}_\mathbf{i}$, the loss is:
  - $\ell(\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i}) = -\sum_{\mathbf{k=1}}^{\mathbf{C}} \mathbf{y_{ik}} \mathbf{log}(\hat{\mathbf{y}}_{\mathbf{ik}})$
  - Where:
    - $\mathbf{C} \rightarrow$ **is the number of classes.**
    - $\mathbf{y_{ik}} = \mathbf{1}: \rightarrow$ if the **k-th class** is the true class for sample i, otherwise $\mathbf{y_{ik}} = \mathbf{0}$ {one hot encoding}.
    - $\hat{y}_{ik} \rightarrow$ is the predicted probability **for class k**.
  - For a dataset of n samples, the average loss (**empirical risk**) given by:
    - $\widehat{\mathbf{R}} = -\frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{i=1}}^{\mathbf{n}} \sum_{\mathbf{k=1}}^{\mathbf{C}} \mathbf{y_{ik}} \, \mathbf{log}(\hat{\mathbf{y}}_{\mathbf{ik}})$ **{also called cost function}**

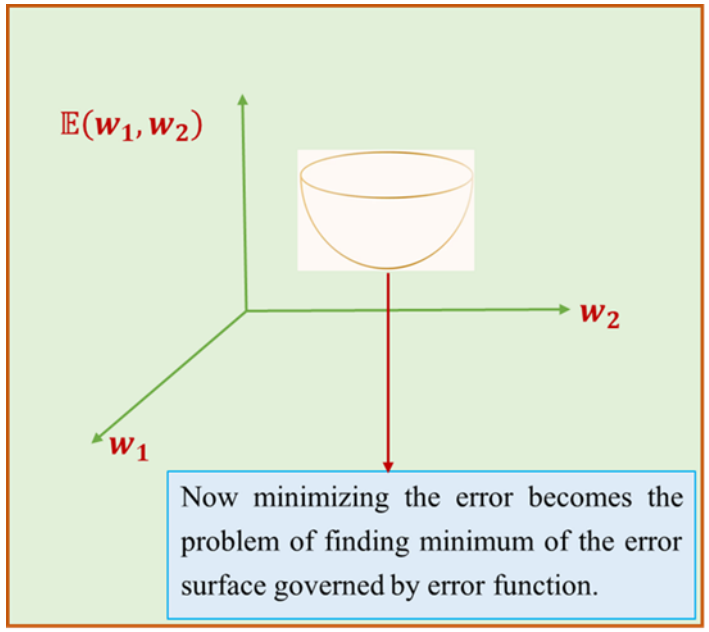
3.3 Softmax Regression in ERM Framework: Model Fitting.

- **Softmax Regression Model Fitting Problem:**
  - **ERM Objective {Explicitly for Softmax Regression}:**
    - The objective is to minimize the average loss (empirical risk) over the training dataset:
      - $\mathcal{L}(\mathbf{W}, \mathbf{b}) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{C} y_{ik} \log(\hat{y}_{ik})$;
    - Substituting $\hat{y}_{ik} = \frac{\exp(z_i)}{\sum_{k=1}^{C}\exp(z_{ik})}$, the loss can be written as:
      - $\mathcal{L}(\mathbf{W}, \mathbf{b}) = -\frac{1}{n}\sum_{i=1}^{n} \log\left(\frac{\exp(z_i)}{\sum_{k=1}^{C}\exp(z_{ik})}\right)$;
    - here $\mathbf{z_i} = \mathbf{W}^T\mathbf{x_i} + \mathbf{b}$; $\mathbf{W} \in \mathbb{R}^{d\times C}$ → is the weight matrix and $\mathbf{b} \in \mathbb{R}^{C}$ → is the bias vector.
  - **Formulating as an Optimization problem:**
    - For any parameter(s) → $\boldsymbol{\theta}^* = [\mathbf{w}, \mathbf{b}: \mathbf{w} \in \mathbb{R}^{d}, \mathbf{b} \in \mathbb{R}] \in \boldsymbol{\Theta}$:
      - $\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}^*} \mathcal{L}(\mathbf{w}, \mathbf{b})$
    - This means **finding** the *weight vector* $\mathbf{w} \in \mathbb{R}^{d\times C}$ and *bias term* $\mathbf{b} \in \mathbb{R}^{C}$ that *minimize the average log loss* over the *training data*.

Towards Gradient Descent!!

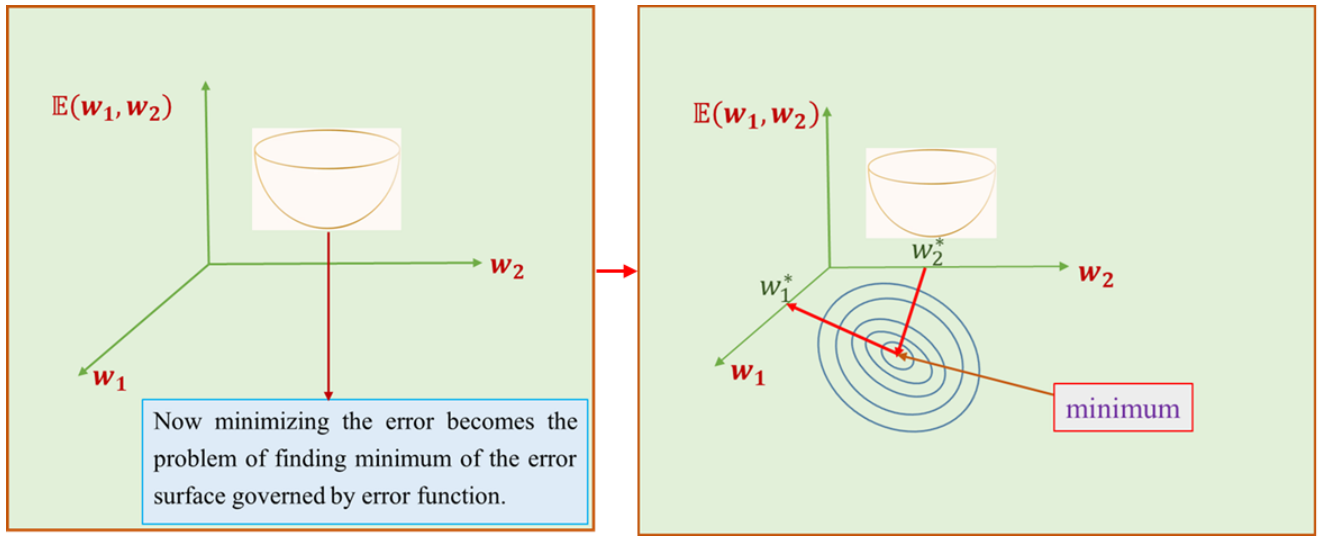
3.4 Before Gradient Descent.

- **Understanding Loss/Error Surface:**
  - It is the surface obtained by plotting average loss or error $\mathbb{E}$ against weights **W**?
  - The shape of the plot depends on the loss function we use, ideally, we want our plot to be convex as shown in picture.



Now our Optimization problem can be thought as:
Finding a point ($W \in [w_1, w_2]$) which produce the minimum error.
How?

3.4.1 Solving Analytically.



We know how to find a minimum of a function.

- Find a critical points by solving:

$$\Delta_W(\mathcal{L}) == \frac{\partial \mathcal{L}}{\partial W} = 0$$

- Evaluate a function on critical points and End points.
- The end points are $[-\infty : \infty]$
- function ranges from (–ve inf) to (+ve inf) as
  - $W \in \mathbb{R}^{d+1}$
- **Solution → Gradient Descent.**

Fig: We can not find such point analytically, thus we depend on iterative algorithm called Gradient Descent.

3.4.2 Remember Gradient.

- Gradient:
  - The gradient of a function of multiple variables is the vector of partial derivatives of the function with respect to each variable.
  - Scalar-by-vector $\{f: \mathbb{R} \to \mathbb{R}^n\}$:
    - The derivative of a scalar function y with respect to a vector $x = [x_1, x_2, \dots, x_n]^T \in \mathbb{R}^n$ is written as:
      - gradients of y: $\nabla y = \frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x_1} \quad \frac{\partial y}{\partial x_2} \quad \dots \quad \frac{\partial y}{\partial x_n}\right]^T$. → gradients.
        - **{Stack the partial derivative against all the element of vector *x*}**
  - Scalar-by-Matrix $\{f: \mathbb{R} \to \mathbb{R}^{n \times m}\}$:
    - The derivative of a scalar function y with respect to a $n \times m$ matrix X is written as:
      - gradients of y: $\nabla y = \frac{\partial y}{\partial X} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \dots & \frac{\partial y}{\partial x_{n1}} \\ \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{1m}} & \dots & \frac{\partial y}{\partial x_{nm}} \end{bmatrix}$
      - **{Stack the partial derivative against all the element of Matrix X.}**

gradient is also the direction of steepest ascent, What does that mean?

3.4.3 Gradient Descent – Idea.

- It is an iterative methods used to find the parameter where the loss function is minimum.
  - The gradient $\nabla_W L$ at any point is the direction of the steepest increase.
    - The negative gradient is the direction of steepest decrease.
- By following the **–ve gradient**, we can eventually find the lowest point.
  - This method is called **Gradient Descent**.

3.4.3 Gradient Descent Algorithm.

- **Algorithm**:
  - For some cost/loss functions: $\mathbb{E}(w_0, \dots, w_d)$.
  - Start off with some guesses for $w_0, \dots, w_d$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero.
  - Repeat until Convergence:{

$$w_{new} := w_{old} - \alpha \frac{\partial \mathbb{E}(w_0, \dots, w_d)}{\partial w}$$

Partial Derivative

Learing Rate

}

3.4.4 Gradient Descent – Impact of Partial Derivative.

- Assume it is simple linear regression and we only have one parameter w and no bias.
- Our Objective is to minimize $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{f}_{\mathbf{w},})$ for the model represented as:
- $\hat{\mathbf{y}} = \mathbf{f}_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}_1\mathbf{x}.$

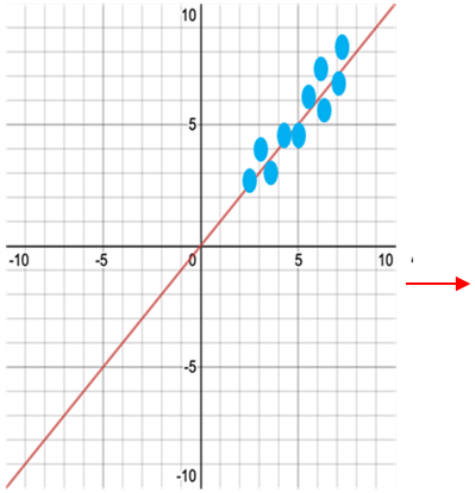


Fig: Regression Line on Data.

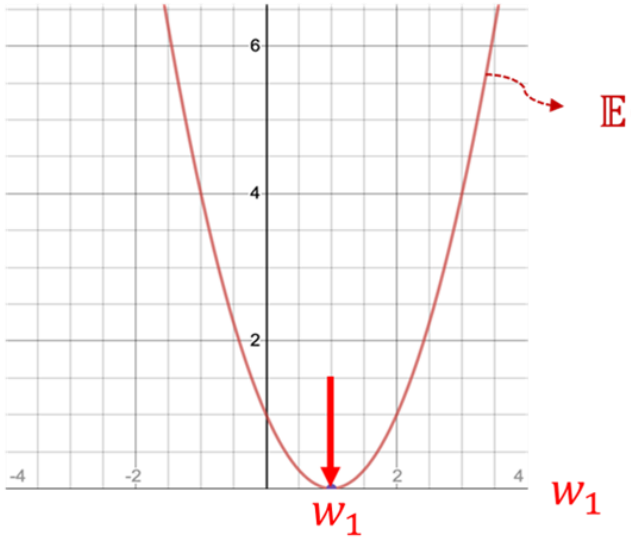


Fig: Weights in Loss Surface.

- **Objective** Understand the impact of Gradient or Partial Derivative.

3.4.5 Partial Derivative is {positive value}.

- Our Objective is to minimize $\mathbb{E}(\mathbf{w_1})$ for the model represented as:
  - $\hat{y} = f_w(x) = W_1 x$.
- Applying Gradient Descent:
  - $w_{1updated} = w_1 - \alpha \frac{d\,E(w_1)}{d\,w_{1_j}}$
  - $w_{1updated} = w_1 - \alpha$ (**Positive**)
- Decrease w_1 by a certain value

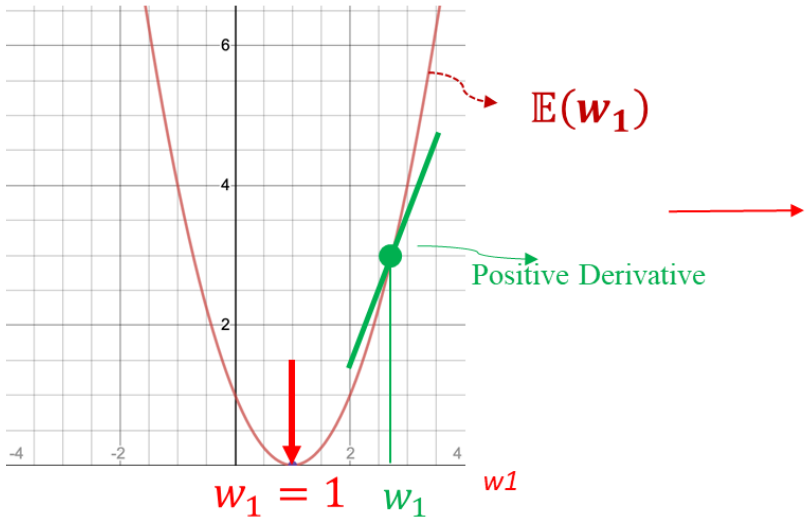


Fig: Regression Line on Data.

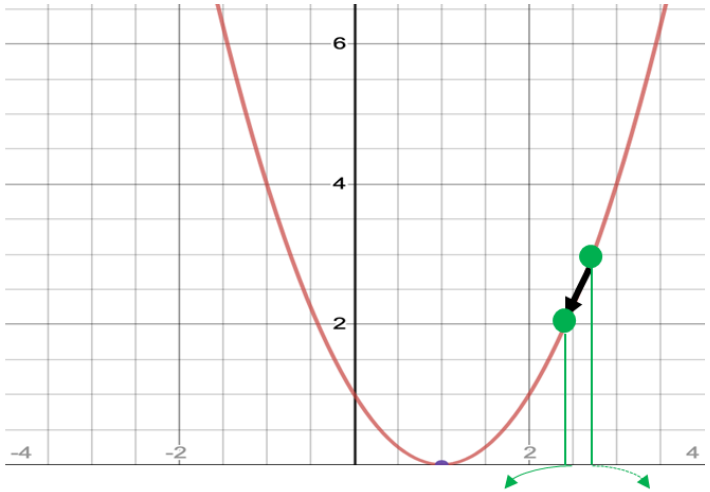


Fig: Weights in Loss Surface.

3.4.6 Partial Derivative is {negative value}.

- Our Objective is to minimize $\mathbb{E}(\mathbf{w_1})$ for the model represented as:
  - $\hat{y} = f_w(x) = W_1 x$.
- Applying Gradient Descent:
  - $w_{1updated} = w_1 - \alpha \frac{d\,E(w_1)}{d\,w_{1j}}$
  - $w_{1updated} = w_1 - \alpha$ (**Negative**)
- Increase w_1 by a certain value

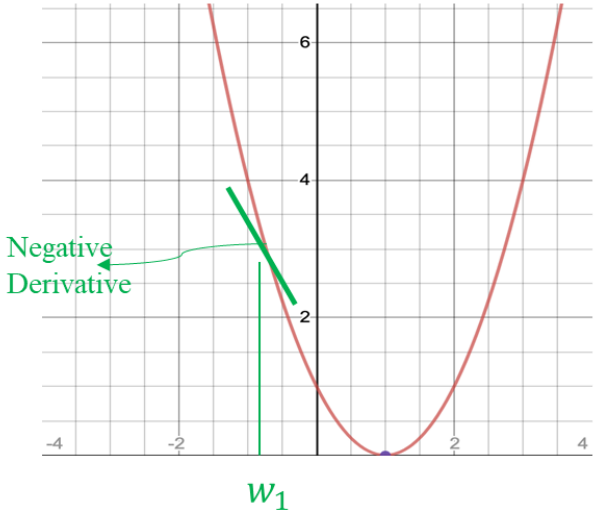


Fig: Regression Line on Data.

Fig: Weights in Loss Surface.

3.4.7 Partial Derivative is {zero}.

- Our Objective is to minimize $\mathbb{E}(\mathbf{w_1})$ for the model represented as:
  - $\hat{\mathbf{y}} = \mathbf{f_w(x)} = \mathbf{W_1 x}$.
- Applying Gradient Descent:
  - $\mathbf{w_{1updated}} = \mathbf{w_1} - \boldsymbol{\alpha} \frac{\mathbf{d\, E(w_1)}}{\mathbf{d\, w_{1_j}}}$
  - $\mathbf{w_{1updated}} = \mathbf{w_1} - \boldsymbol{\alpha}\,(\mathbf{0})$
- $\mathbf{w_1}$ remains the same

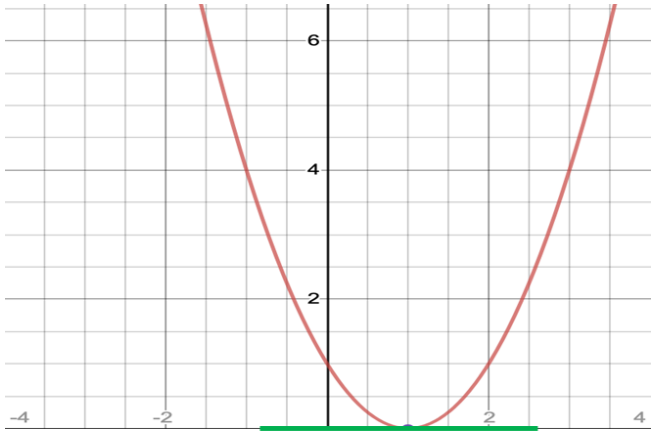


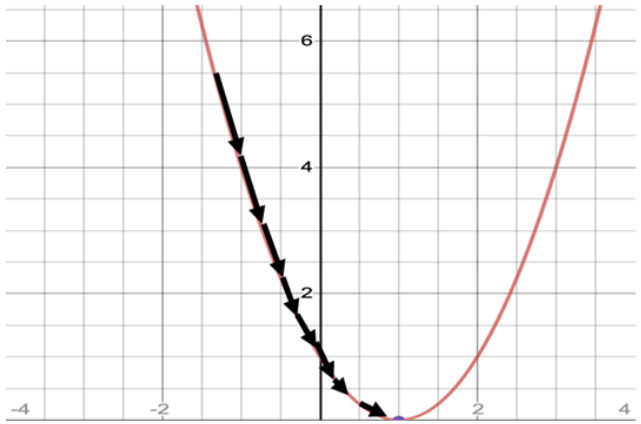
Fig: We found the optimal parameter.

3.4.7 Gradient Descent – Learning Rate.

- Our Objective is to minimize $\mathbb{E}(\mathbf{w_1})$ for the model represented as:
  - $\hat{\mathbf{y}} = \mathbf{f_w(x)} = \mathbf{W_1 x}$.
- Applying Gradient Descent:
  - $\mathbf{w_{1updated}} = \mathbf{w_1} - \boldsymbol{\alpha} \frac{\mathbf{d\,E(w_1)}}{\mathbf{d\,w_{1_j}}}$

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

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



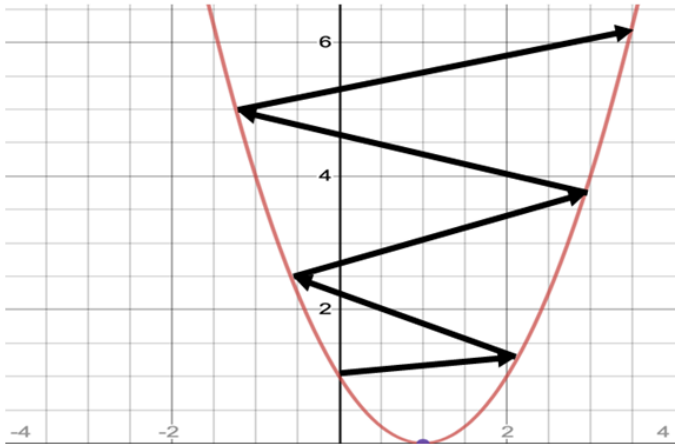


Fig: Understanding the Impact of Learning Rate.

3.5 Softmax Regression in ERM Framework: Optimization.

- **To minimize we typically use gradient descent:**
  - For gradient descent we first compute a gradient of the loss function with respect to **w and b** which is given by:

Gradient against W

The gradients of the Categorical Cross-Entropy Loss are:

$$\frac{\partial \text{Categorical-Cross-Entropy Loss}}{\partial \mathbf{W}} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right)\mathbf{x}_i^\top,$$

where $\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{z}_i)$ and $\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i + \mathbf{b}$.

Gradient against b

The gradients of the Categorical Cross-Entropy Loss are:

$$\frac{\partial \text{Categorical-Cross-Entropy Loss}}{\partial \mathbf{b}} = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right),$$

where $\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{z}_i)$ and $\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i + \mathbf{b}$.

3.5.1 Softmax Regression in ERM Framework: Gradient Descent.

Algorithm 1 Gradient Descent for Softmax Regression

1: **Input:** Dataset $\mathcal{D}_n = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, 2, \ldots, n\}$, learning rate $\alpha > 0$, number of iterations T

2: **Initialize:** Parameters $\mathbf{W} \in \mathbb{R}^{d \times C}$ and $\mathbf{b} \in \mathbb{R}^{C}$ (e.g., set to 0 or small random values)

3: **for** t = 1 to T **do**

4: Compute logits for all samples:

$$\mathbf{z}_i = \mathbf{W}^\top \mathbf{x}_i + \mathbf{b} \quad \forall i \in \{1, 2, \ldots, n\}$$

5: Compute predictions using the softmax function:

$$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{z}_i) = \frac{\exp(\mathbf{z}_i)}{\sum_{c=1}^{C} \exp(z_{ic})} \quad \forall i \in \{1, 2, \ldots, n\}$$

6: Compute gradients:

$$\frac{\partial \texttt{Categorical-Cross-Entropy Loss}}{\partial \mathbf{W}} = \frac{1}{n} \sum_{i=1}^{n} (\hat{\mathbf{y}}_i - \mathbf{y}_i) \mathbf{x}_i^\top$$

$$\frac{\partial \texttt{Categorical-Cross-Entropy Loss}}{\partial \mathbf{b}} = \frac{1}{n} \sum_{i=1}^{n} (\hat{\mathbf{y}}_i - \mathbf{y}_i)$$

7: Update parameters:

$$\mathbf{W} \leftarrow \mathbf{W} - \alpha \frac{\partial \texttt{Cross-Entropy Loss}}{\partial \mathbf{W}}$$

$$\mathbf{b} \leftarrow \mathbf{b} - \alpha \frac{\partial \texttt{Cross-Entropy Loss}}{\partial \mathbf{b}}$$

8: **end for**

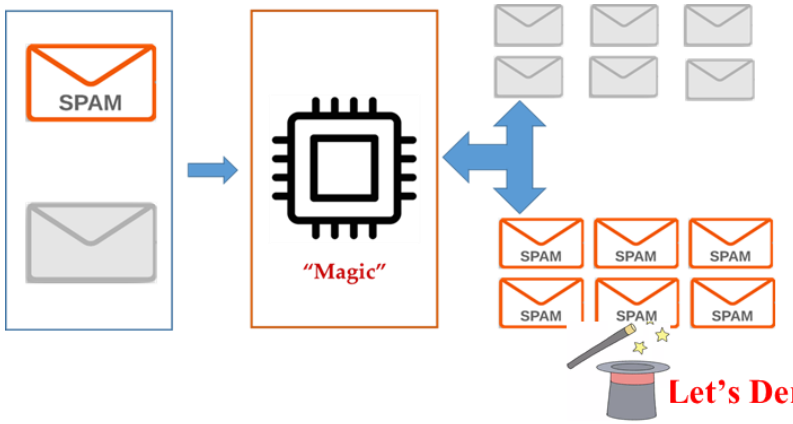
9: **Output:** Optimal parameters $\mathbf{W}^*$ and $\mathbf{b}^*$

How Good is Your Model/Classifier?

4. The Evaluation Metrics ~ For Binary Classification.

{Idea can be extended to Multi-class}

4.1 Example: Remember!!!



We replaced the magic with Logistic Regression and Got the following Result

Let's Demystify the Magic!!

	Email	Actual	Predicted
1	I need help please wire me $1000 right now	1 - Spam	1 - Spam
2	Hot new investment, don't miss this!	1 - Spam	0 - Not Spam
3	Please help me?	1 - Spam	0 - Not Spam
4	Your parcel will be delivered today	0 - Not Spam	0 - Not Spam
5	Review changes to our Terms and Conditions	0 - Not Spam	0 - Not Spam
6	Weekly sync notes	0 - Not Spam	0 - Not Spam
7	Re: Follow up from today's meeting	0 - Not Spam	0 - Not Spam
8	Pre-read for tomorrow	0 - Not Spam	0 - Not Spam
9	Brief results from new UX study	0 - Not Spam	0 - Not Spam
10	Meeting notes from today	0 - Not Spam	0 - Not Spam
11	A reminder about your next appointment	0 - Not Spam	1 - Spam
12	An invitation to a conference call tomorrow	0 - Not Spam	0 - Not Spam
13	We have some news about your application	0 - Not Spam	1 - Spam
14	Final briefing notes for your meeting	0 - Not Spam	0 - Not Spam
15	Attached are the updated guidelines for this project	0 - Not Spam	0 - Not Spam
16	New feedback on your project from last week's meeting	0 - Not Spam	0 - Not Spam
17	Thank you for participating in our survey	0 - Not Spam	0 - Not Spam
18	Your account has been upgraded to Premium status	0 - Not Spam	1 - Spam
19	Confirming the dates for the annual board meeting	0 - Not Spam	0 - Not Spam
20	Please review these updated guidelines	0 - Not Spam	0 - Not Spam

How good of a job we did?

How good is my Model?

4.2 Accuracy!!

- The most straightforward way to measure a classifier's performance is using the Accuracy metric.
- Here, we compare the actual and predicted class of each data point, i.e. for total predictions how many were correctly predicted.
- Accuracy is given as:
  - **accuracy = $\frac{\text{Number of correct predictions}}{\text{Total number of predictions.}}$**
- For our Example:

	Email	Actual	Predicted	
1	I need help please wire me $1000 right now	1 - Spam	1 - Spam	✓
2	Hot new investment, don't miss this!	1 - Spam	0 - Not Spam	✗
3	Please help me?	1 - Spam	0 - Not Spam	✗
4	Your parcel will be delivered today	0 - Not Spam	0 - Not Spam	✓
5	Review changes to our Terms and Conditions	0 - Not Spam	0 - Not Spam	✓
6	Weekly sync notes	0 - Not Spam	0 - Not Spam	✓
7	Re: Follow up from today's meeting	0 - Not Spam	0 - Not Spam	✓
8	Pre-read for tomorrow	0 - Not Spam	0 - Not Spam	✓
9	Brief results from new UX study	0 - Not Spam	0 - Not Spam	✓
10	Meeting notes from today	0 - Not Spam	0 - Not Spam	✓
11	A reminder about your next appointment	0 - Not Spam	1 - Spam	✗
12	An invitation to a conference call tomorrow	0 - Not Spam	0 - Not Spam	✓
13	We have some news about your application	0 - Not Spam	1 - Spam	✗
14	Final briefing notes for your meeting	0 - Not Spam	0 - Not Spam	✓
15	Attached are the updated guidelines for this project	0 - Not Spam	0 - Not Spam	✓
16	New feedback on your project from last week's meeting	0 - Not Spam	0 - Not Spam	✓
17	Thank you for participating in our survey	0 - Not Spam	0 - Not Spam	✓
18	Your account has been upgraded to Premium status	0 - Not Spam	1 - Spam	✗
19	Confirming the dates for the annual board meeting	0 - Not Spam	0 - Not Spam	✓
20	Please review these updated guidelines	0 - Not Spam	0 - Not Spam	✓

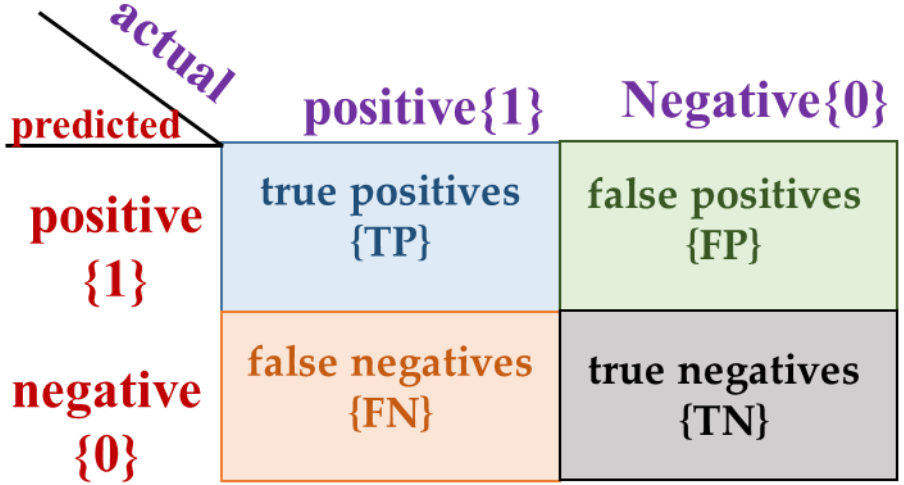
$$\text{accuracy} = \frac{15}{20} \times 100\% = 75\%$$

4.3 Is accuracy an adequate evaluation metrics?

- Accuracy is often used as the measure of classification performance because it is simple to compute and easy to interpret.
- However, it can turn out to be misleading in some cases. For instances:
  - **class imbalance**: scenario where certain classes contain way more data points than the others.
    - In our example: **17 out of 20 datapoints are of not-spam class**, and only **3 are from spam class**.
      - What if our model has predicted all datapoints to be not-spam, accuracy would have been:= $\frac{17}{20} \times 100\% = 85\%$.
  - **differential misclassification costs** – getting a positive wrong costs more than getting a negative wrong.
    - For example: we built a model to identify Tumor from the given datasets and overall accuracy more than 90% {**out of 10 times 9 times we predicted tumor correctly**}.
      - What happens for onetime we predicted person to have Tumor and the person do not have tumor?

4.4 Other Accuracy Metrics: Confusion Matrix

- A confusion matrix, is a technique for summarizing the performance of classification algorithm.
- The Confusion Matrix takes the classification results and groups them into four categories:
- For our email example we assign:
  - spam a 1 label{class of interest}
  - not-spam a 0 label

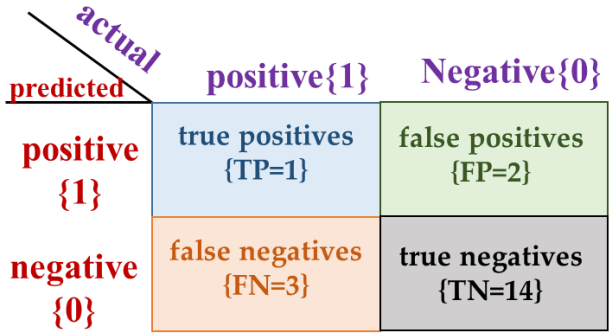


confusion matrix

4.5 Confusion Matrix: Example.

- Let's populate the confusion matrix with email classification example.
- For our email example we assign:
  - spam a **1** label{class of interest}
  - **not-spam** a **0** label
- Thus:
  - TP:→ actual **spam {1}** predicted **spam {1}** := 1
  - FP:→ actual **spam{1}** predicted **not-spam{0}**:= 2
  - TN:→ actual **not-spam{0}** predicted **not-spam{0}**:= 14
  - FN:→ actual **not-spam{0}** predicted **spam{1}**:= 3

	Email	Actual	Predicted
1	I need help please wire me $1000 right now	1 - Spam	1 - Spam
2	Hot new investment, don't miss this!	1 - Spam	0 - Not Spam
3	Please help me?	1 - Spam	0 - Not Spam
4	Your parcel will be delivered today	0 - Not Spam	0 - Not Spam
5	Review changes to our Terms and Conditions	0 - Not Spam	0 - Not Spam
6	Weekly sync notes	0 - Not Spam	0 - Not Spam
7	Re: Follow up from today's meeting	0 - Not Spam	0 - Not Spam
8	Pre-read for tomorrow	0 - Not Spam	0 - Not Spam
9	Brief results from new UX study	0 - Not Spam	0 - Not Spam
10	Meeting notes from today	0 - Not Spam	0 - Not Spam
11	A reminder about your next appointment	0 - Not Spam	1 - Spam
12	An invitation to a conference call tomorrow	0 - Not Spam	0 - Not Spam
13	We have some news about your application	0 - Not Spam	1 - Spam
14	Final briefing notes for your meeting	0 - Not Spam	0 - Not Spam
15	Attached are the updated guidelines for this project	0 - Not Spam	0 - Not Spam
16	New feedback on your project from last week's meeting	0 - Not Spam	0 - Not Spam
17	Thank you for participating in our survey	0 - Not Spam	0 - Not Spam
18	Your account has been upgraded to Premium status	0 - Not Spam	1 - Spam
19	Confirming the dates for the annual board meeting	0 - Not Spam	0 - Not Spam
20	Please review these updated guidelines	0 - Not Spam	0 - Not Spam



4.6.1 Evaluation Metrics

- Confusion metrics are then utilized to generate various other metrics including accuracy.
- **Accuracy:**
  - simply a ratio of correctly predicted observation to the total observations.
    - $\textbf{accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$
- **Precision:**
  - Only looks at positive class/label.
  - Precision is the ratio of **correctly predicted positive observations** to the **total predicted positive observations**.
    - {Out of all **predicted positive class** how many **are correct**}
    - $\textbf{precison} = \frac{TP}{TP+FP}$

4.6.2 Evaluation Metrics

- Confusion metrics are then utilized to generate various other metrics including accuracy.
- Recall {aka sensitivity aka true positive rate}:
  - Recall is the ratio of correctly predicted positive observations to the all observations in actual class – yes
  - Out of all the positive classes, how many instances were identified correctly.
  - i.e. Sensitivity describes how good a model at predicting positive classes.
  - higher the sensitivity value means your model is good in predicting positive classes
  - {Among total positive classes in dataset: How many were correctly classified}

$$\textbf{recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

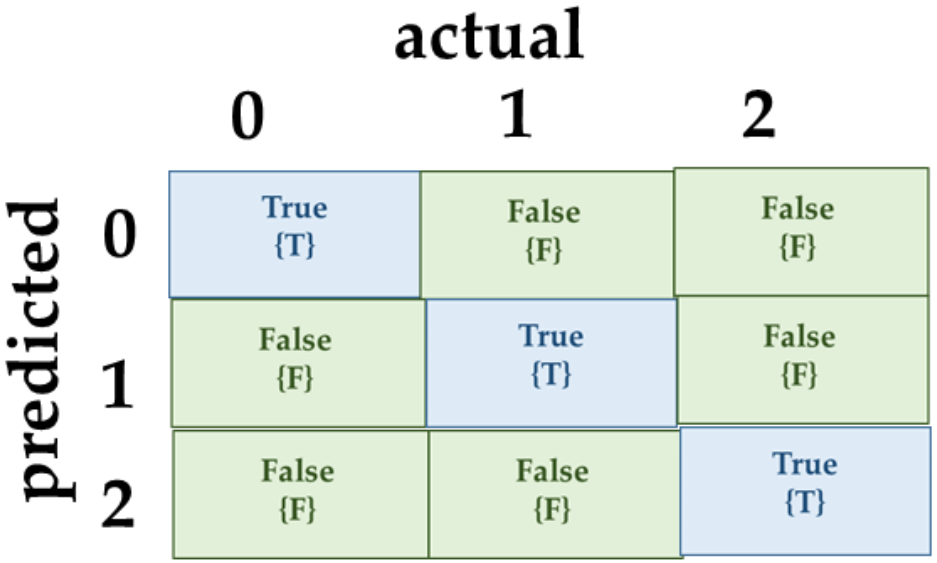
4.6.3 Evaluation Metrics

- If you are not sure about which metrics is better for you task in hand, there is always F1-Score:
- F1 score is the weighted average of Precision and Recall.

$$\mathbf{F1-Score} = \frac{\mathbf{Precision} * \mathbf{Recall}}{\mathbf{Precision} + \mathbf{Recall}}$$

4.7 Confusion Matrix : Multi-class Problem

- Error in classification problem can be broadly of two kind i.e.
  - True:
    - label {y} is 0 predicted {$\hat{y}$} is 0.
    - label {y} is 1 predicted {$\hat{y}$} is 1.
  - False:;
    - label {y} is 0 predicted {$\hat{y}$} is 1.
    - label {y} is 1 predicted {$\hat{y}$} is 0.
- We can extend this idea to build confusion Matrix for multi class problem.



Confusion matrix with 3 class

4.7.1 Extending to: Precision and Recall

- In our example of email classification, we only have two class **spam and not a spam**.
- Now let's imagine there are three different kind of email tags namely:
  - **urgent, normal and spam**
- We built a Multinomial Logistic Regression or Softmax Regression we can determine precision and recall as:

	urgent	normal	spam	
urgent	8	10	1	$\textbf{precision}_u = \frac{8}{8+10+1}$
normal	5	60	50	$\textbf{precision}_n = \frac{60}{5+60+50}$
spam	3	30	200	$\textbf{precision}_s = \frac{200}{3+30+200}$
	$\textbf{recall}_u = \frac{8}{8+5+3}$	$\textbf{recall}_n = \frac{60}{10+60+30}$	$\textbf{recall}_s = \frac{200}{1+50+200}$	

The – End.