Introduction to ML

ACM India Summer School on
Responsible & Safe AI

3rd June, 2024

There is a function f in nature and we want to model / approximate it:

$$f : D \to R$$

We only have input - output samples available

Example:

- D: Number Tuples
- R: Real numbers

[(1,2), 3], [(2,3), 5], [(3,4), 7], [(4,5), 9]

Game: Guess the function f

Parameters and architecture

To make an approximation, we *parameterize* the function. We call this function f_θ

$$f_\theta : D \to R$$

Continuing with our previous example, let us define f_θ as:

$$f_\theta(x) = \theta_2 x_1 + \theta_1 x_0 + \theta_0$$

We have effectively defined an *architecture* for our function with parameters. Now, our problem becomes finding the right values for $\theta_0, \theta_1, \theta_2$ that "fits" the data.

<u>There are non – parametric Machine Learning Algorithms as well! E.g : knn, Decision Trees.</u>

ML Pipeline

Data

Representation

Algorithms

Evaluation

Data Preprocessing – Textual data

- Removing punctuations like . , ! $( ) * % @
- Removing URLs
- Removing Stop words – the, it, a, was
- Lower casing
- Tokenization – break a sentence into small words
- Stemming vs Lemmatization

Data Preprocessing – Textual data

Stemming	Lemmatization
Stemming is a process that stems or removes last few characters from a word, often leading to incorrect meanings and spelling.	**Lemmatization** considers the context and converts the word to its meaningful base form, which is called Lemma.
For instance, stemming the word '**Caring**' would return '**Car**'.	For instance, lemmatizing the word '**Caring**' would return '**Care**'.
Stemming is used in case of large dataset where performance is an issue.	Lemmatization is computationally expensive since it involves look-up tables and what not.

Data Representation

Bag of Words Example

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List
for
is
of
the
to

TF-IDF Example

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Data Representation

Bag of Words Example

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

Now is the time for all good men to come to the aid of their party.

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

Stopword List
for
is
of
the
to

Can be extended to n-grams.

Instead of counting words – counts grouping of words

Unigram - "The", "Car", "Truck"....
Bigram - "<s> The, "The Car", "Car Truck"....
Trigram - "<s> The Car" , "The Car Truck" ...

TF-IDF Example

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Representing words by their context

- Distributional semantics: **A word's meaning is given by the words that frequently appear close-by**
  - *"You shall know a word by the company it keeps"* (J. R. Firth 1957: 11)
  - One of the most successful ideas of modern statistical NLP!
- When a word *w* appears in a text, its **context** is the set of words that appear nearby (within a fixed-size window).
- We use the many contexts of *w* to build up a representation of *w*

...government debt problems turning into	***banking***	*crises as happened in 2009...*
...saying that Europe needs unified	***banking***	*regulation to replace the hodgepodge...*
...India has just given its	***banking***	*system a shot in the arm...*

These context words will represent ***banking*** Reference – CS224N, Stanford University

Word vectors

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts, measuring similarity as the vector dot (scalar) product

$$banking = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} \qquad monetary = \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix}$$

Note: word vectors are also called (word) embeddings or (neural) word representations
They are a distributed representation

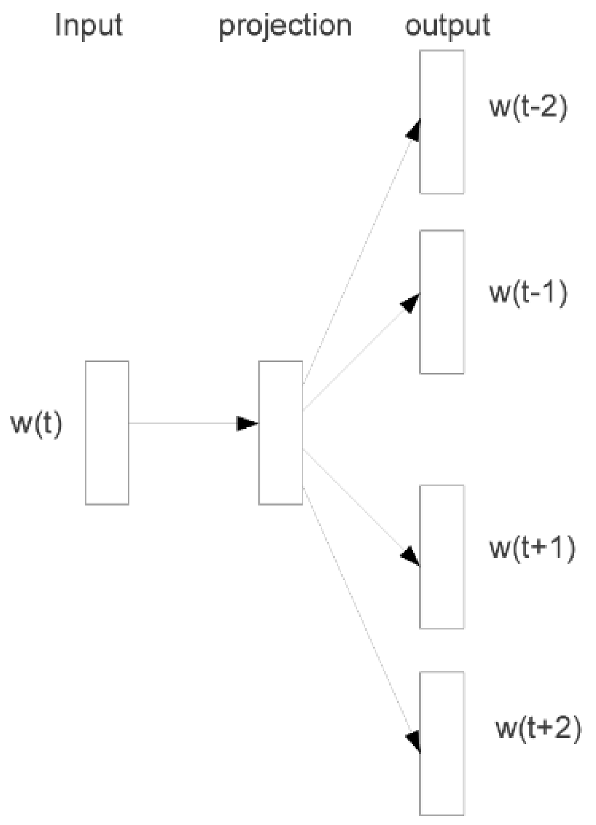
Reference – CS224N, Stanford University

Word2vec: Overview

Word2vec is a framework for learning word vectors (Mikolov et al. 2013)

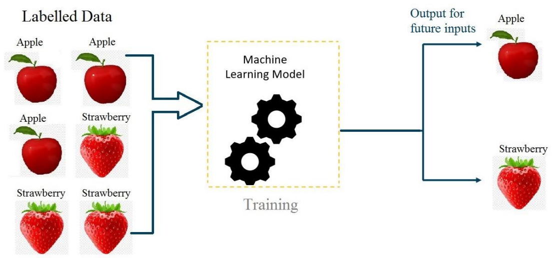
Idea:

- We have a large corpus ("body") of text: a long list of words
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context ("outside") words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

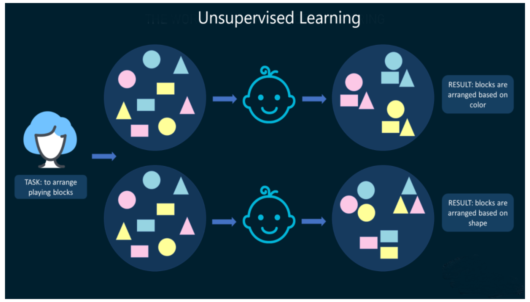


Skip-gram model
(Mikolov et al. 2013)

ML Algorithms

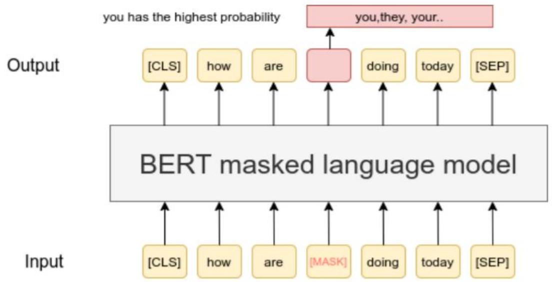


Supervised



Un-Supervised

Reinforcement



Self-Supervised

Supervised Learning

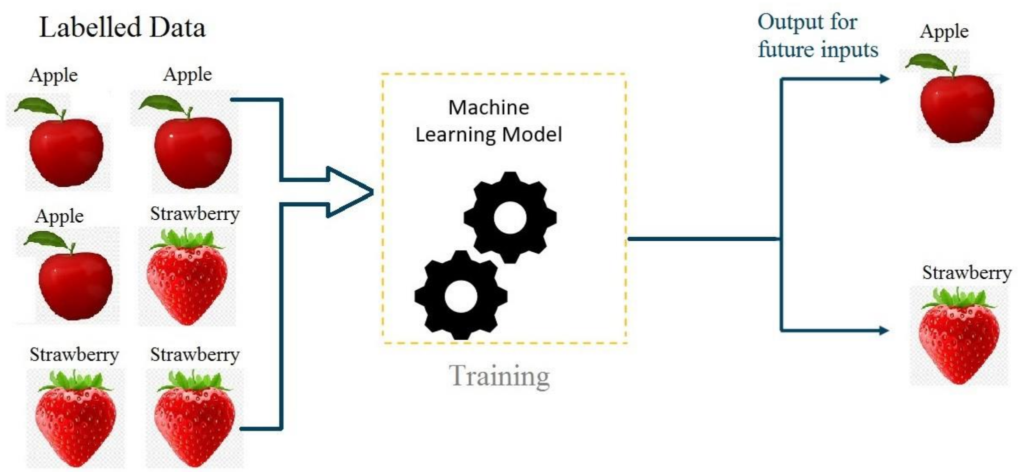
- Training on labeled dataset
- Learn mapping from inputs to outputs in training
- Predict output for new, unseen data

Given a dataset $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where x_i are the input features and y_i are the corresponding labels, the model learns a function f such that $f(x_i) \approx y_i$.

For regression: $y = f(x) + \epsilon$, where ϵ is the error term.

For classification: $P(y \mid x) = \frac{e^{f(x)}}{1+e^{f(x)}}$ (logistic regression for binary classification).

Supervised Learning - Example



Classification: Determining if an email is spam or not spam

Regression: Predicting house prices based on features like size, location, and age

Supervised Learning

Common Algorithms

Linear Regression: $y = w^T x + b$
Logistic Regression:

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w^T x + b)}}$$

Decision Trees: Recursive partitioning of data space.
Support Vector Machines (SVM): $y = \text{sign}(w^T x + b)$
Neural Networks: $y = \sigma(W^T x + b)$ (where σ is an activation function)

Un-Supervised Learning

- Train on data that does not have labeled responses
- The goal is to find hidden patterns in the input data

Given a dataset $\{x_1, x_2, \ldots, x_n\}$, the objective is to find patterns or groupings in the data without explicit labels.

For clustering: Minimize the sum of squared distances between data points and their assigned cluster centroids,

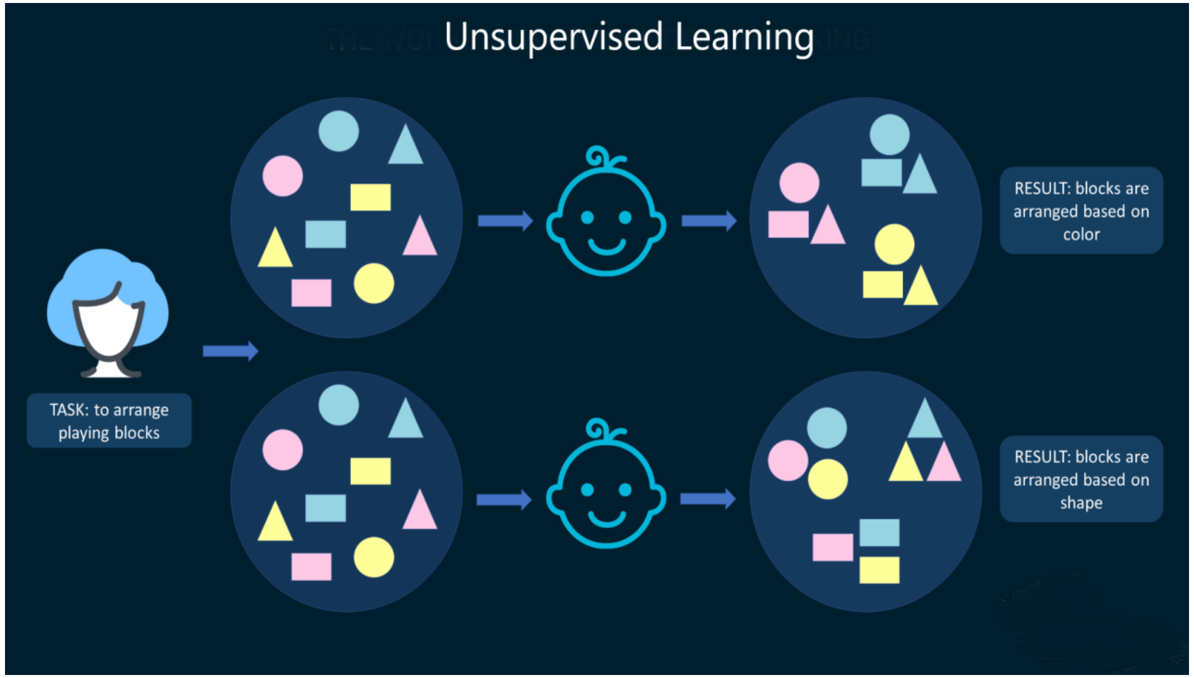
$$\sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where μ_i is the centroid of cluster C_i.

For dimensionality reduction (e.g., PCA): Maximize the variance retained in the projected data,

$$\max \|XW\|^2 \quad \text{subject to} \quad W^T W = I.$$

Unsupervised Learning - Example



Clustering: Grouping customers based on purchasing behavior.

Dimensionality Reduction: Reducing the number of features in a dataset while retaining its essential information (e.g., using PCA).

Metrics for Evaluation

Metrics for Evaluation

Ground Truth

Predictions

	A	B
A	98	0
B	2	0

Confusion Matrix

		gold standard labels	
		gold positive	gold negative
system output labels	system positive	**true positive**	**false positive**
	system negative	**false negative**	**true negative**

$$\textbf{precision} = \frac{\text{tp}}{\text{tp+fp}}$$

$$\textbf{recall} = \frac{\text{tp}}{\text{tp+fn}}$$

$$\textbf{accuracy} = \frac{\text{tp+tn}}{\text{tp+fp+tn+fn}}$$

Confusion Matrix

		gold standard labels	
		gold positive	gold negative
system output labels	system positive	**true positive**	**false positive**
	system negative	**false negative**	**true negative**

Confusion Matrix

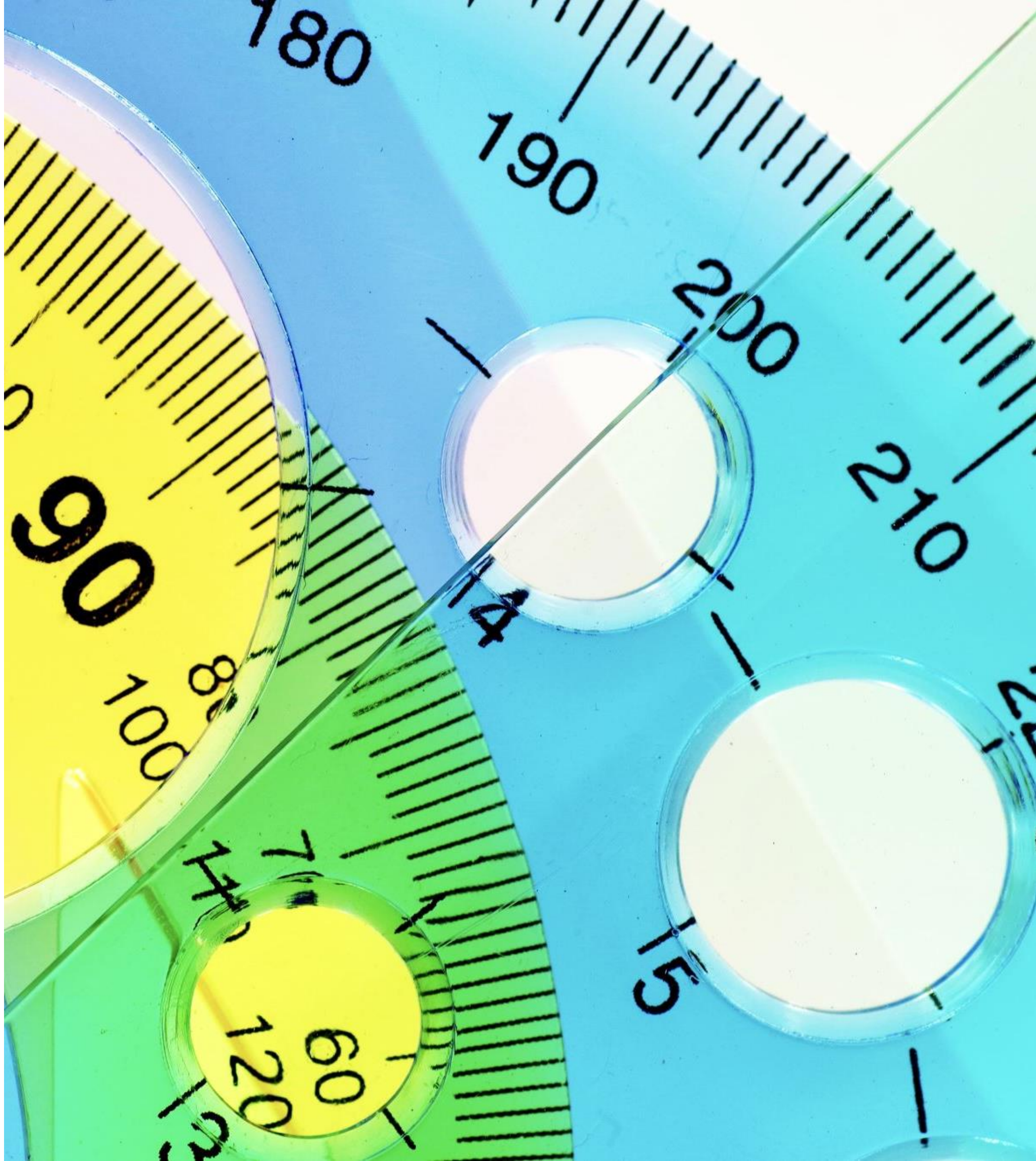
		gold labels			
		urgent	normal	spam	
system output	urgent	8	10	1	**precision**$_u = \frac{8}{8+10+1}$
	normal	5	60	50	**precision**$_n = \frac{60}{5+60+50}$
	spam	3	30	200	**precision**$_s = \frac{200}{3+30+200}$
		recall$_u = \frac{8}{8+5+3}$	**recall**$_n = \frac{60}{10+60+30}$	**recall**$_s = \frac{200}{1+50+200}$	

Let's calculate other metrics

- Precision = 2/(2+1) = 67%
- Recall = 2/(2+3) = 40%
- Accuracy = (94+2)/100 = 96%

- Which measure should we account?

- RECALL

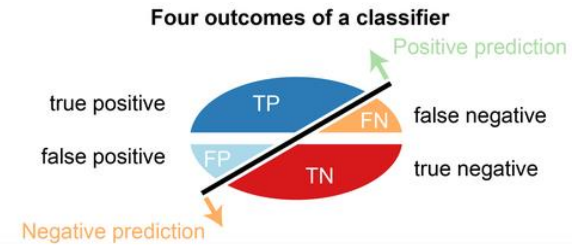


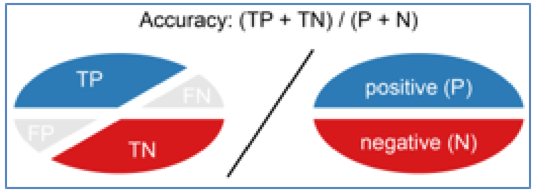
Another example

- A system which needs to launch a missile at a terrorist hideout located in a dense urban area.
- Which metric should we consider?
- Precision not 100% - civilian causalities

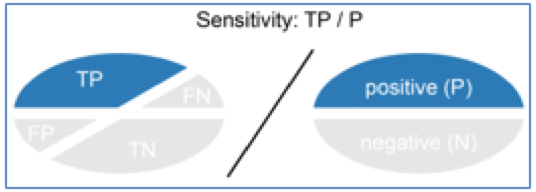
Accuracy vs Precision vs Recall

- Monitor Precision if a false positive carries higher cost.
- Monitor Recall if a false negative carries higher cost.

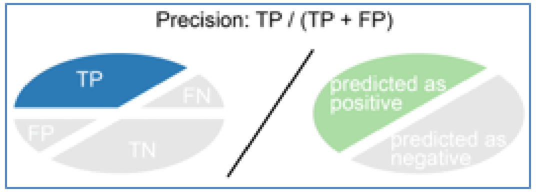




% of correct predictions



% of + class correctly predicted
[aka Recall / TPR]



correct prediction of + class
[aka Precision]

F1- Score

- What to do when one classifier has better precision but worse Recall, while other classifier behaves exactly opposite?

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- F1 measure punishes extreme values more !

- Definition of Recall and Precision have same numerator, different denominators. A sensible way to combine them is harmonic mean.

Evaluation metrics

- For Regression algorithms, we can't use accuracy as metric to judge models' performance

- We use Root Mean Square Error | Mean Absolute Error, among others

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

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

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

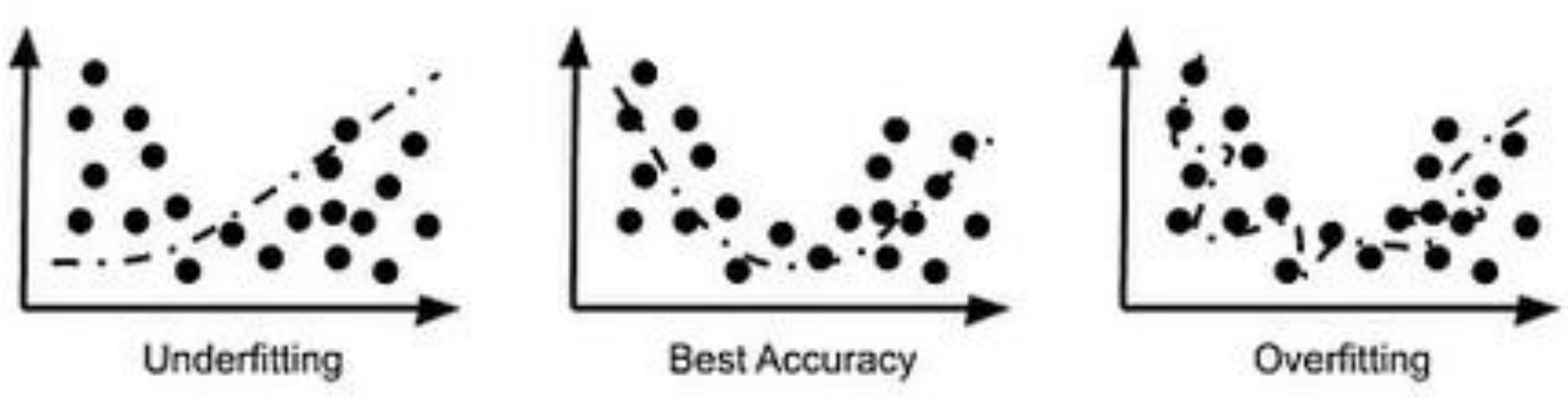
Where,

$\hat{y}$ $-$ *predicted value of* y

$\bar{y}$ $-$ *mean value of* y

Underfitting - Overfitting

- Underfitting: Happens when model is too simple to capture the underlying structure of the data
- Overfitting: This happens when a model is too complex and captures noise or random fluctuations in the training data

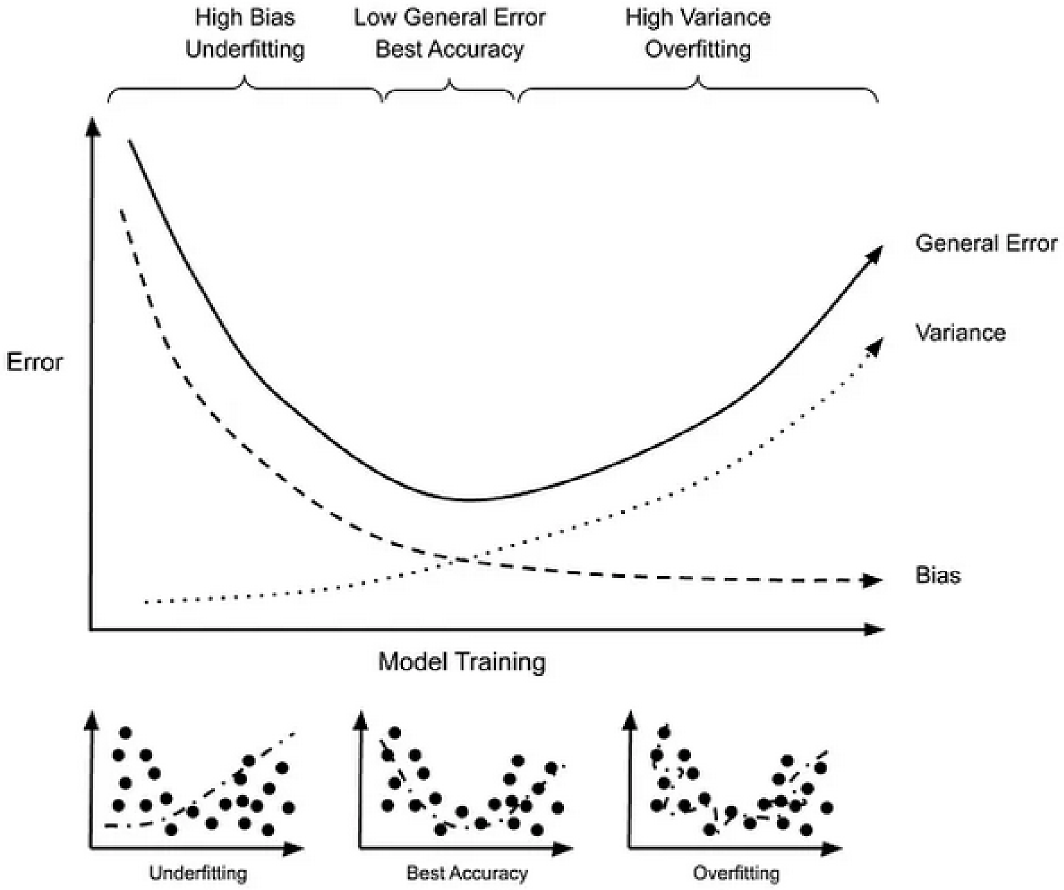


Bias- Variance

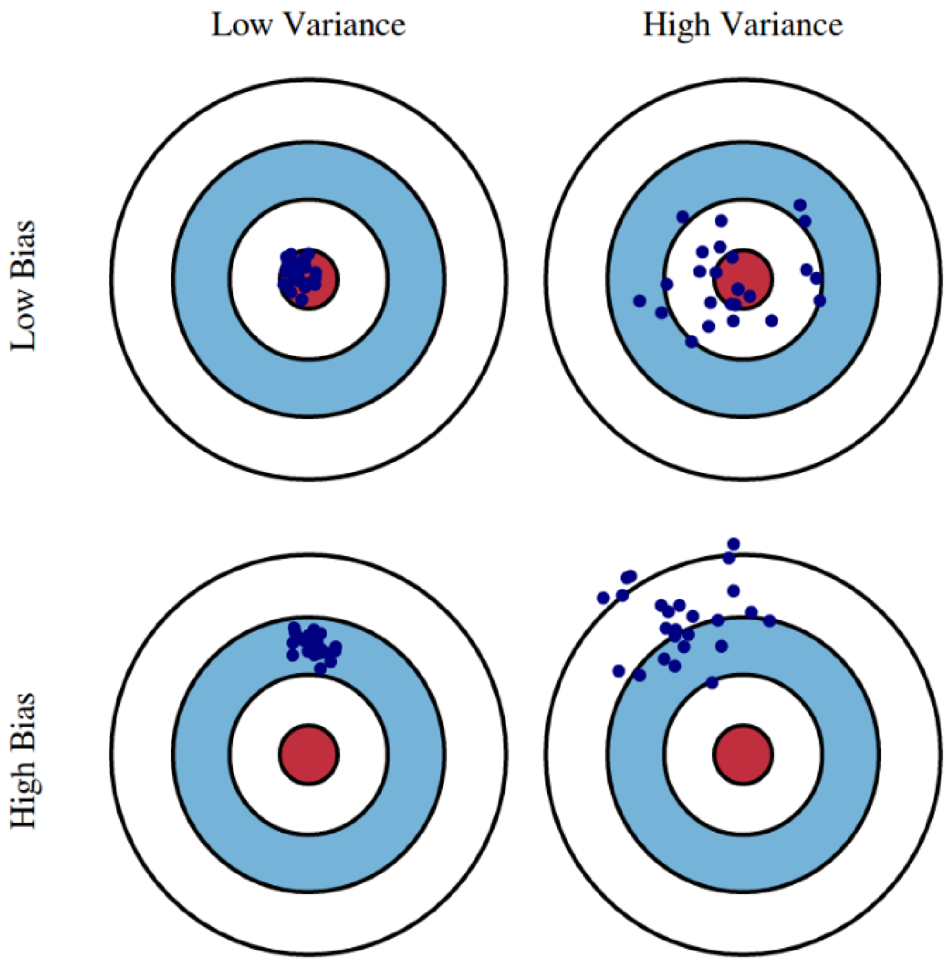
- **Bias (Underfitting)**
  - Poor predictive performance as the model cannot capture the complexity of the problem.
  - High bias leads to underfitting, where the model oversimplifies underlying patterns - Error due to overly simplistic assumptions in the learning algorithm.
  - Example: Linear regression model assuming a strictly linear relationship for predicting housing prices.

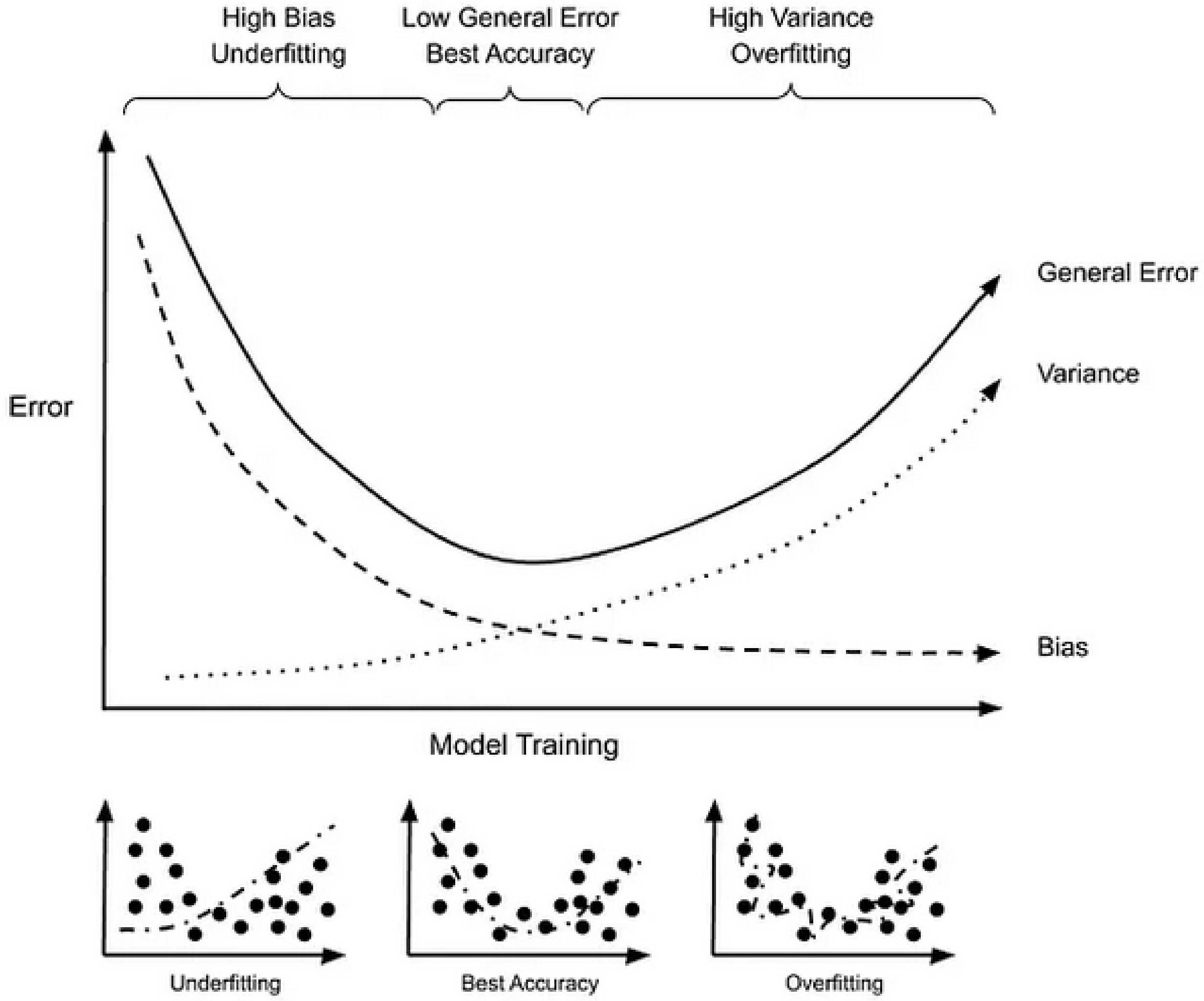
- **Variance (Overfitting)**
  - Poor generalization to unseen data due to essentially memorizing the training examples. Error due to excessive complexity in the learning algorithm.
  - High variance captures both underlying patterns and noise in the training data.
  - Example: Decision tree with a very deep structure trained on handwritten digits.

Bias- Variance Tarde off



Bias- Variance Tarde off

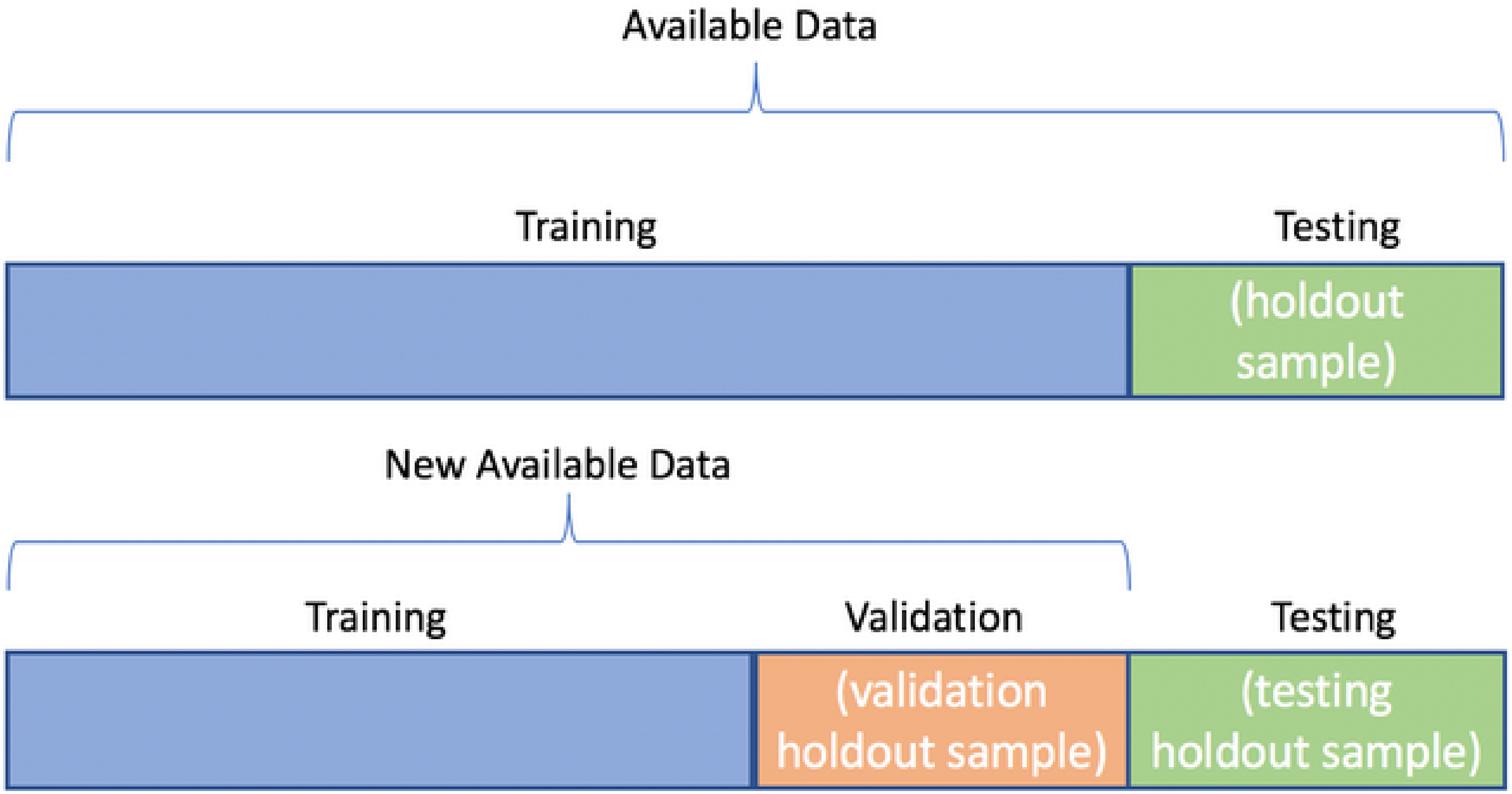




Bias- Variance Tarde off

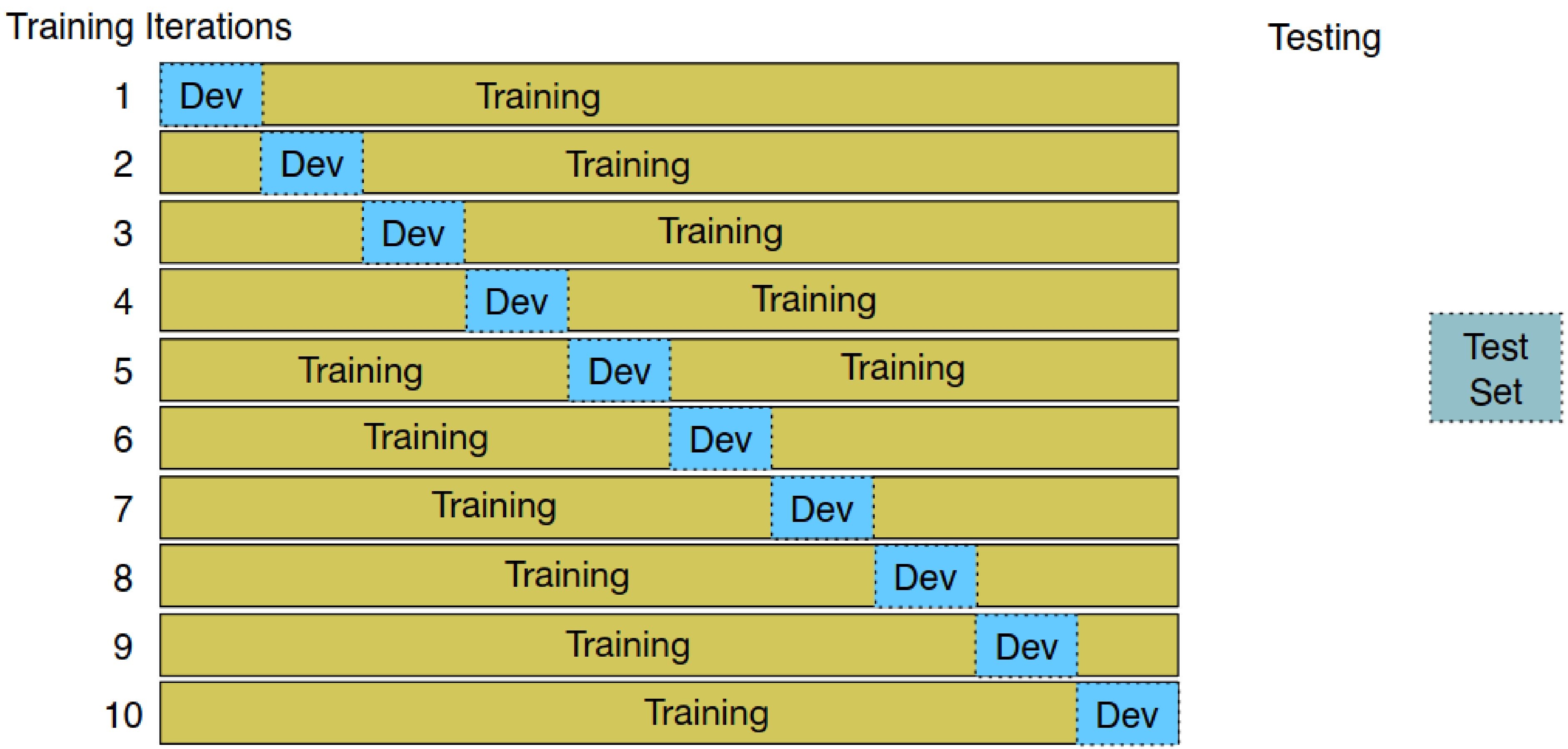
- The bias-variance tradeoff is the delicate equilibrium between underfitting and overfitting.
- The goal is to find the optimal level of complexity that allows a model to generalize effectively to unseen data.

Train-Val-Test paradigm



https://datascience.stackexchange.com/questions/61467/clarification-on-train-test-and-val-and-how-to-use-implement-it

K-fold Cross Validation



Loss Functions

Why Loss Functions

- Let's say I am on the top of mountain and need to climb down. How do I decide where to walk towards?
  - Look around to see all the possible paths
  - Reject the ones going up
  - Finally, take the path that I think has the most slope downhill

- This intuition that I just judged my decisions against? This is exactly what loss function provides

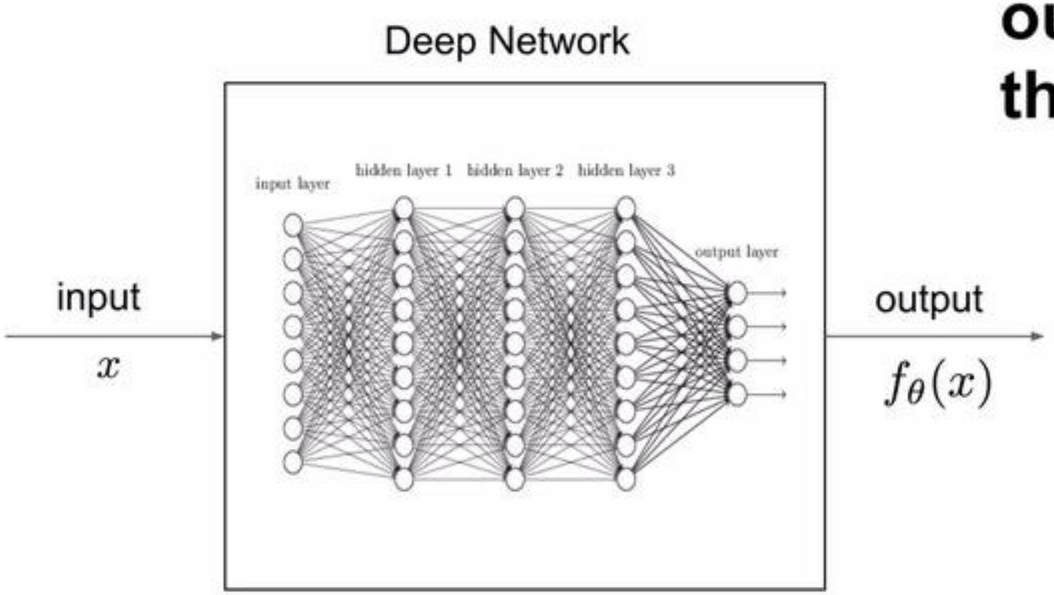
Loss Functions

- Loss functions are used to quantify how well or bad a model can reproduce the values of the training set.
- The appropriate loss function depends on the type of problems and the algorithm we use.
- Loss function = Cost function = Objective function = Error function
- Loss function does not want to measure the entire performance of the network against a validation/test dataset

Loss Functions

The loss function is used to **guide** the **training process** in order to find a set of parameters that reduce the value of the loss function.

Loss Function



How good does our network with the training data?

labels (ground truth)

input

$$\mathcal{L}(w) = distance(f_\theta(x), y)$$

error

parameters (weights, biases)

Training Process

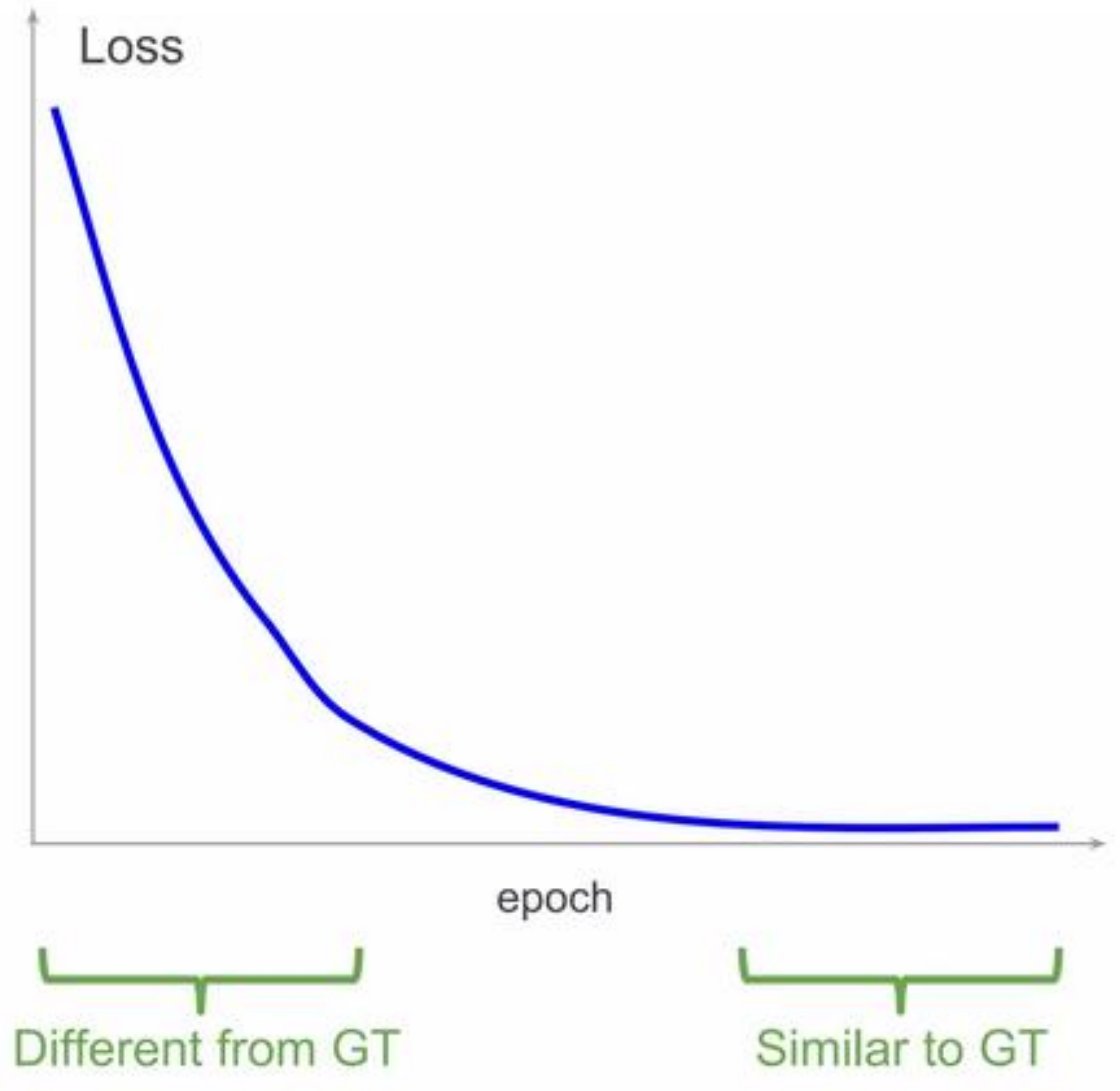
Stochastic gradient descent

- Find a set of parameters which make the loss as small as possible.
- Change parameters at a rate determined by the partial derivatives of the loss function:

$$\frac{\partial \mathcal{L}}{\partial w} \quad \frac{\partial \mathcal{L}}{\partial b}$$

Properties

- Minimum when the output of the network is equal to the ground truth data
- Increase value when the output differs from the ground truth



Introduction to SK-learn

- Code