Week 4

Deep Learning I

Multilayer Perceptron

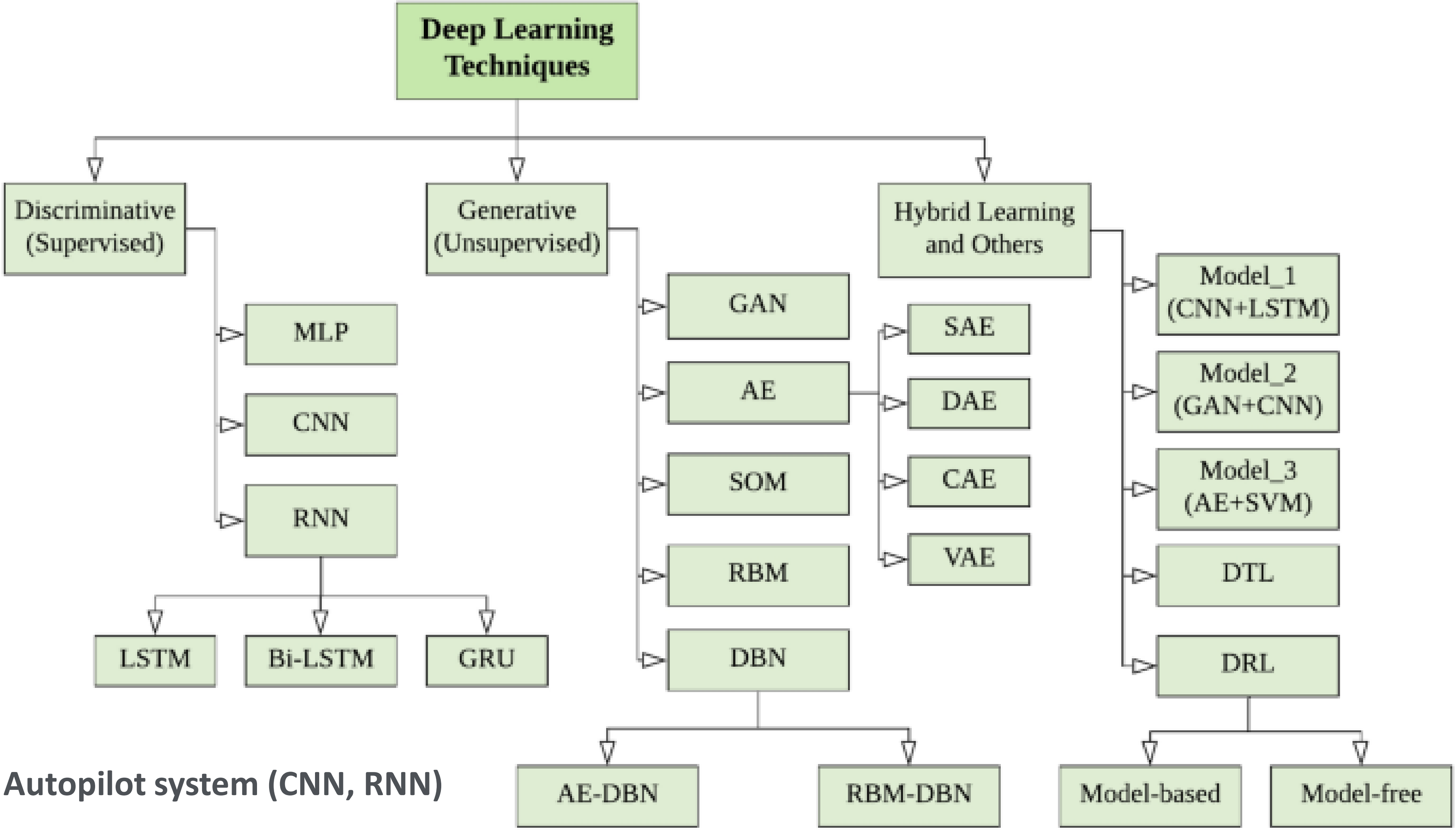
Dr Anagi Gamachchi

Discipline of Information Systems and Business Analytics,
Deakin Business School

What is the AI technique/algorithm behind ChatGPT?

What is the AI technique/algorithm behind Tesla's Autopilot system?

Name any deep machine learning technique that you know!



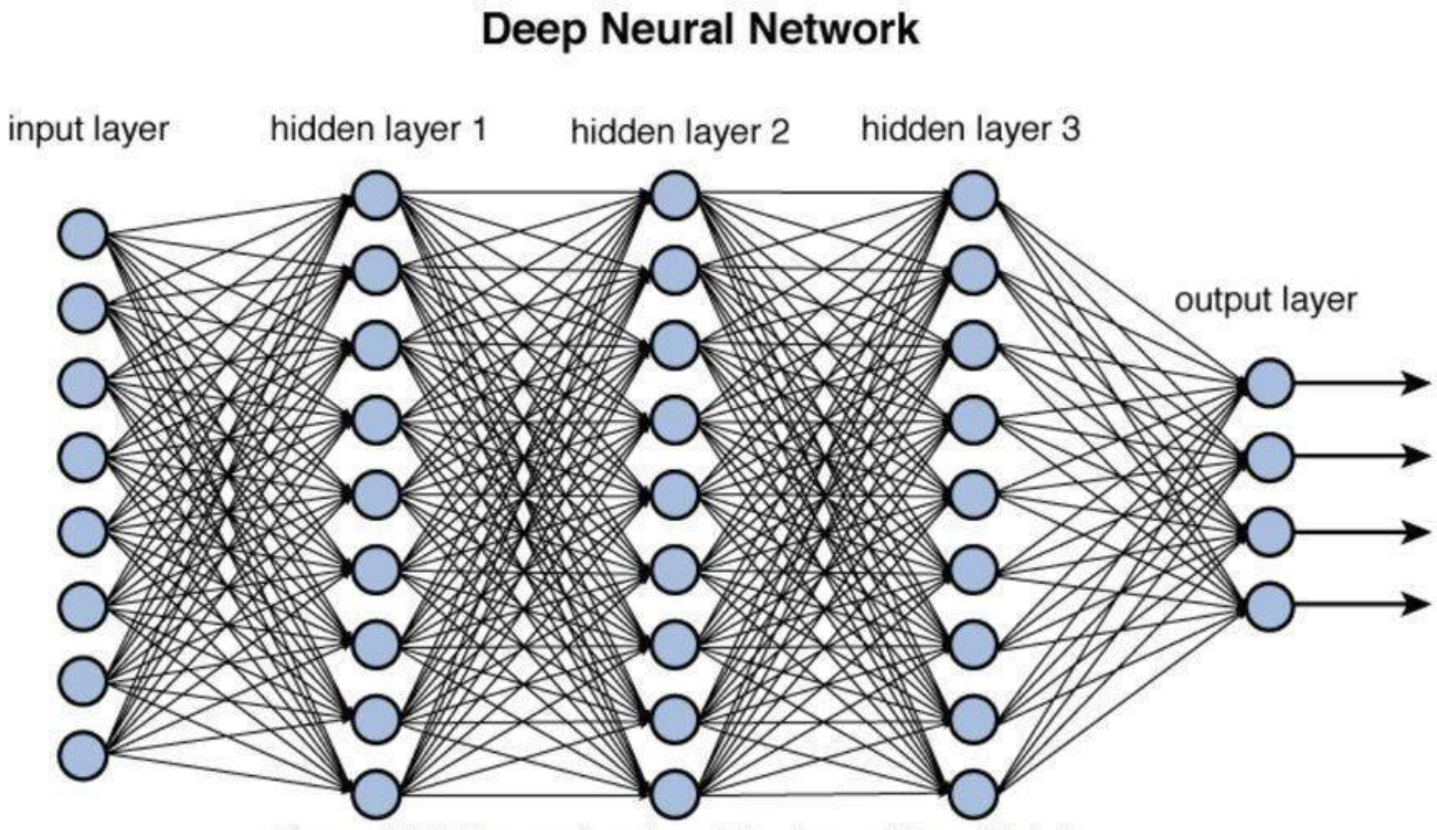
Tesla's Autopilot system (CNN, RNN)

ChatGPT's Large Language Model

Source: https://link.springer.com/article/10.1007/s42979-021-00815-1

Deep Learning

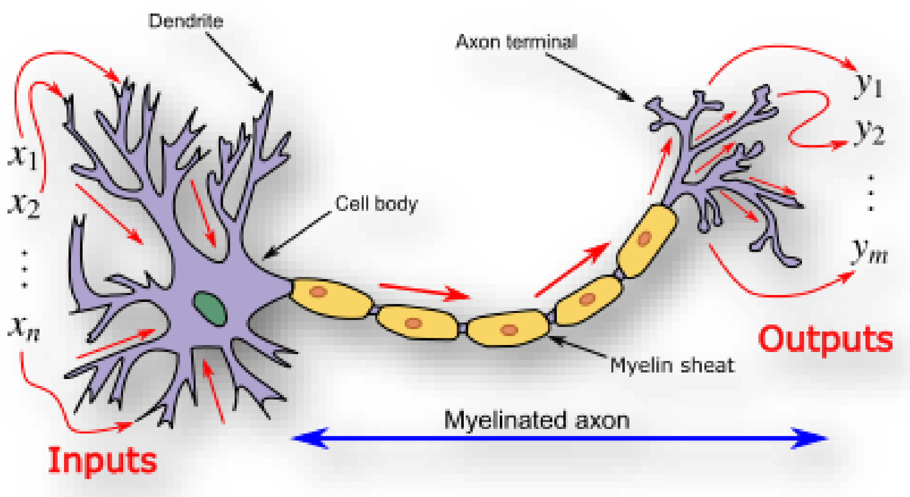
- ❑ ***Deep learning*** is a class of machine learning methods which aim at creating successive layers of increasingly more meaningful representations of data, which lead to better predictions.
- ❑ ***Artificial Neural networks*** are the most common Deep Learning technique.



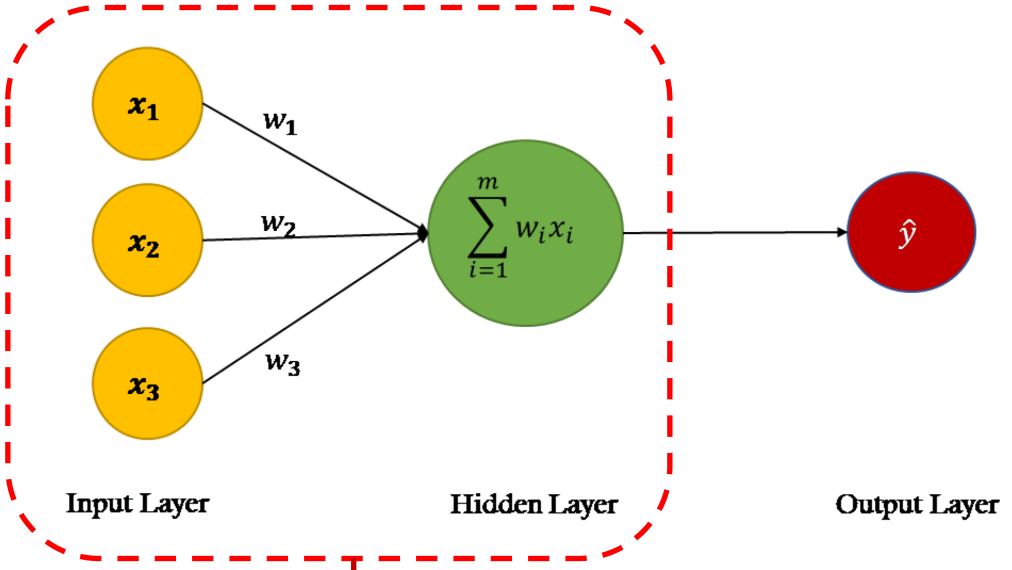
Mathematical Foundation of Artificial Neural Networks (ANN)

The architecture of ANN is inspired by biological neuron (e.g., human brain)

A biological neuron



ANN with a single neuron

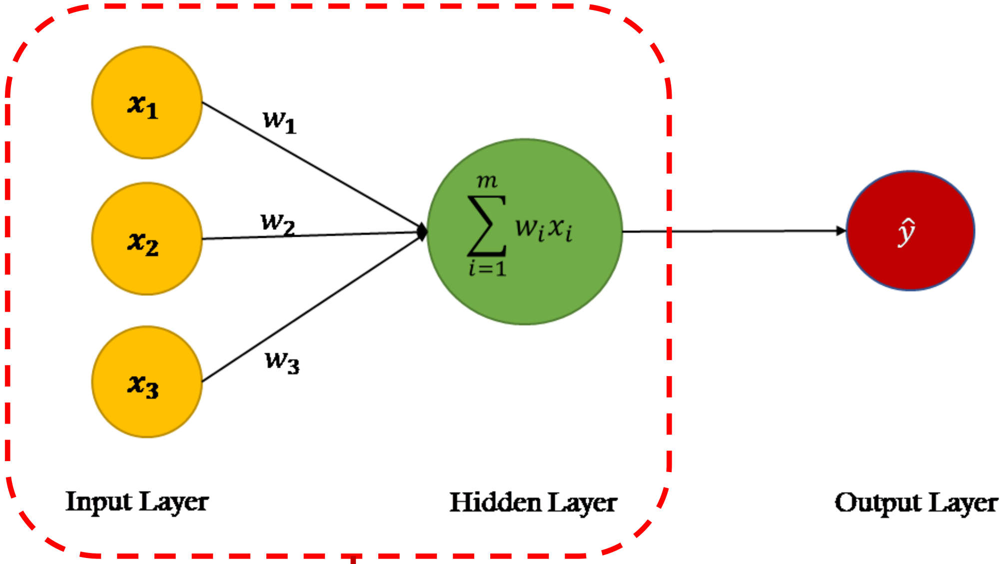


$$w_0 + w_1 \times x_1 + w_2 \times x_2 + \ldots + w_m \times x_m$$

The above formular is similar to the formular of which machine learning technique?

Mathematical Foundation of Artificial Neural Networks (ANN)

ANN with a single neuron



$$W_0 + W_1 \times X_1 + W_2 \times X_2 + \ldots + W_m \times X_m$$

Linear Regression Model

Example: Predict Height of Children, based on age, weight, …, daily intake

Multiple Linear Regression

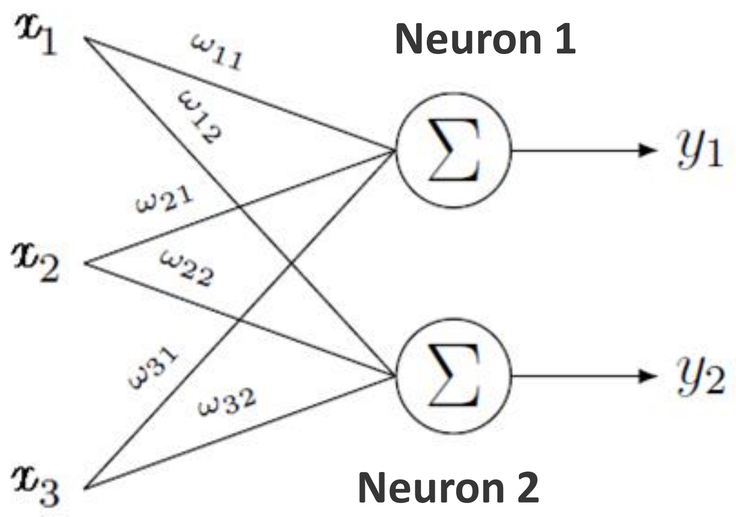
height, age, weight, daily intake

$$\boldsymbol{y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}$$

w_0=intercept/bias
w_i=slope/weight for x_i
x_i=independent variable/predictor i
y=dependent variable/label

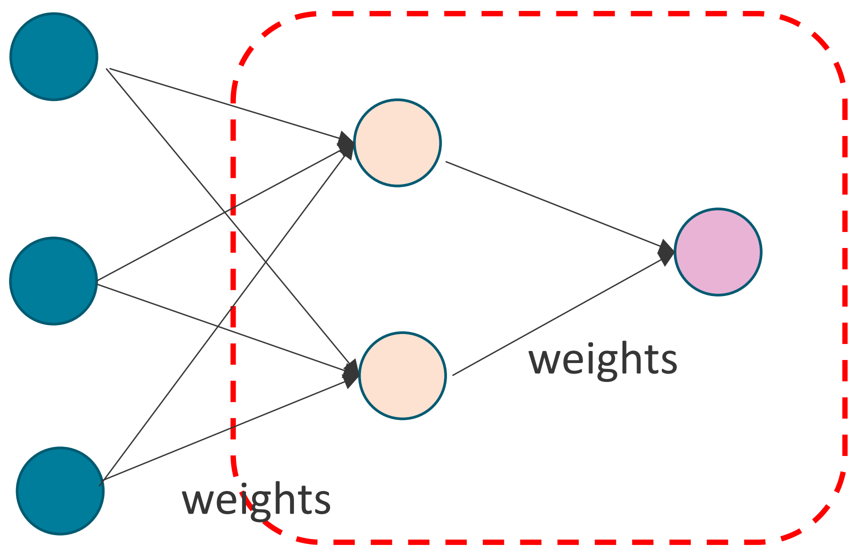
Model parameters

ANN with Multiple Neurons



Not a complete ANN architecture yet!

Number of nodes are varied depending on applications!

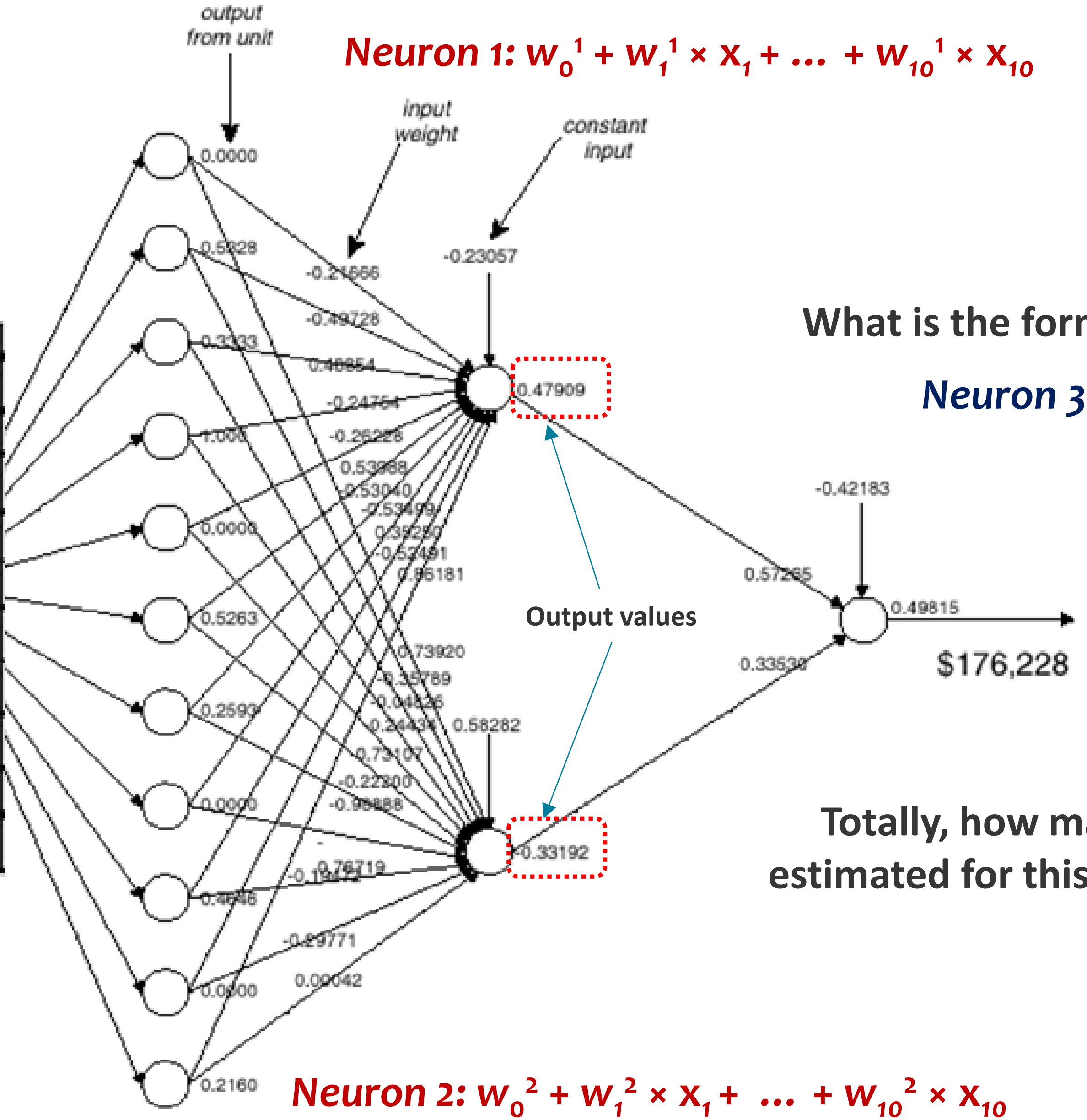


Combine outputs of all neurons for final output layer

A complete ANN for house price prediction

$\textit{Neuron 1: } w_0^1 + w_1^1 \times x_1 + \dots + w_{10}^1 \times x_{10}$

Num_Apartments	1	0.0000
Year_Built	1923	0.5328
Plumbing_Fixtures	9	0.3333
Heating_Type	B	1.0000
Basement_Garage	0	0.0000
Attached_Garage	120	0.5263
Living_Area	1614	0.2593
Deck_Area	0	0.0000
Porch_Area	210	0.4646
Recroom_Area	0	0.0000
Basement_Area	175	0.2160



What is the formular for the output layer?

$\textit{Neuron 3: } w_0^3 + w_1^3 \times n_1 + w_2^3 \times n_2$

Totally, how many parameters to be estimated for this neural network model?

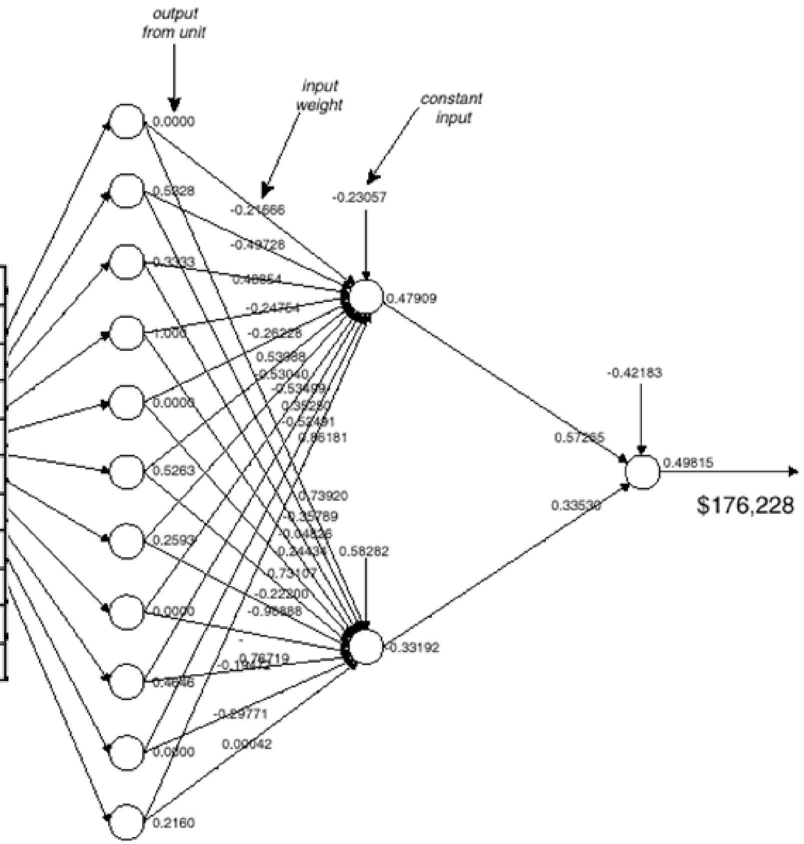
$\textit{Neuron 2: } w_0^2 + w_1^2 \times x_1 + \dots + w_{10}^2 \times x_{10}$

Discussion Question

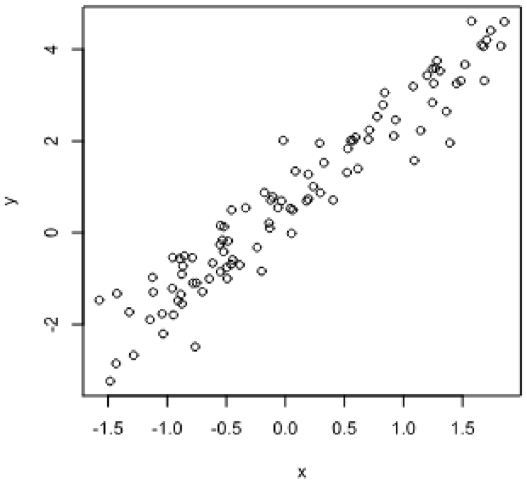
Given the same data set (e.g. house price), which model would have better performance?

Linear Regression vs. Artificial Neural Networks

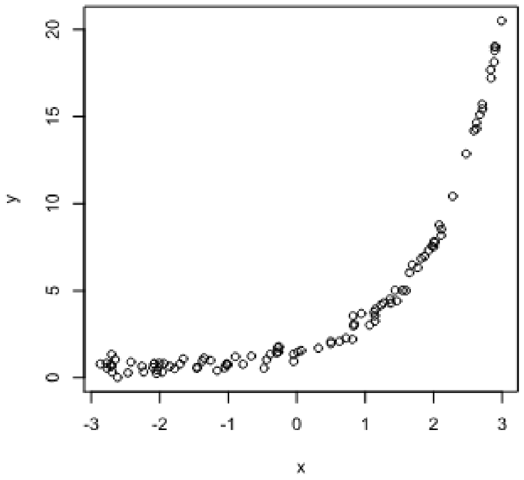
Num_Apartments	1	0.0000
Year_Built	1923	0.5328
Plumbing_Fixtures	9	0.3333
Heating_Type	B	1.0000
Basement_Garage	0	0.0000
Attached_Garage	120	0.5263
Living_Area	1614	0.2593
Deck_Area	0	0.0000
Porch_Area	210	0.4646
Recroom_Area	0	0.0000
Basement_Area	175	0.2160



Linear Data



Non-Linear Data

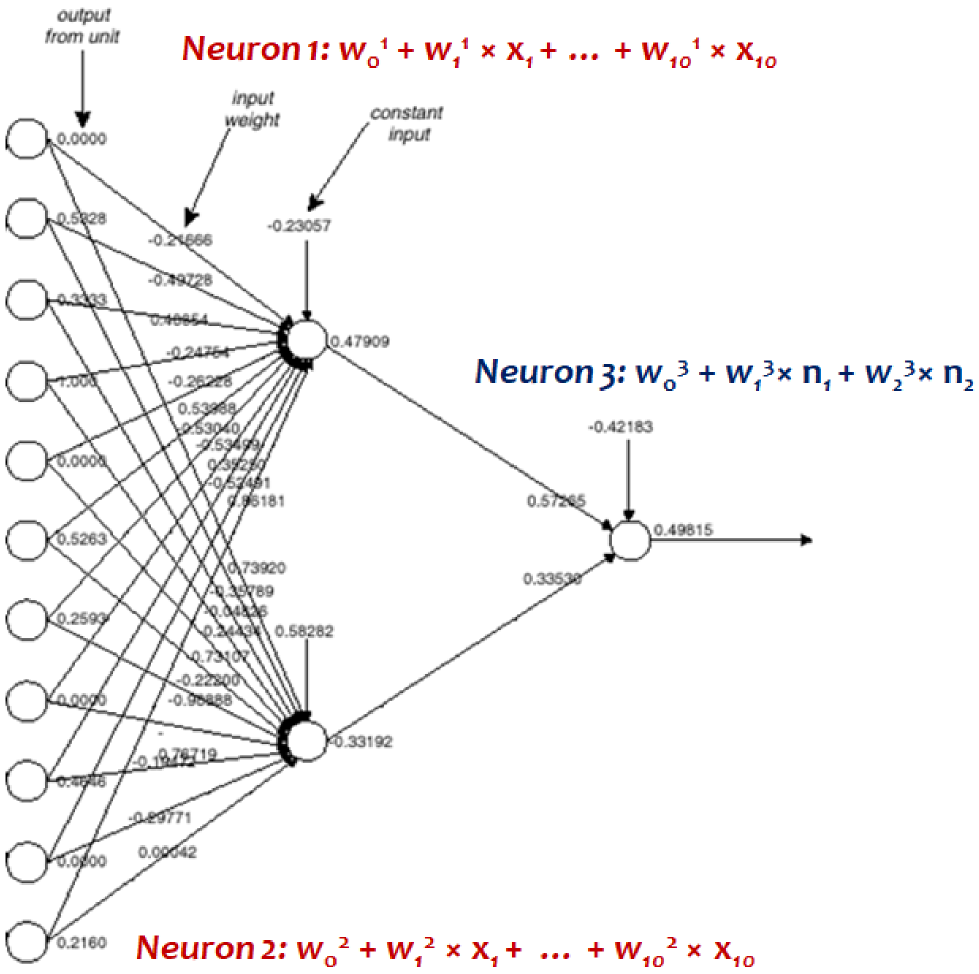


Model Training

- **At learning time**, an ANN…
  1. Initializes network weights
  2. Takes the inputs and the desired outputs
  3. Predicts the output
  4. Calculating error (difference between desired and predicted values)
  5. Updates its internal state (i.e., weights) to minimize error rate in the outcome (final) node/s

  The learning cycle can be repeated in several **epochs**

  Learning stops when…
  - The *error rate does not change significantly*, or
  - A set *number of epochs* are completed
- **At testing time**, an ANN…
  1. Takes the unseen inputs
  2. Generates/predicts the output using the internal state based on its "training experience"



Hyper-Parameters

- **Learning algorithm** is based on gradient descent optimization to find the optimum model.
- At each layer, the **weights** are adjusted in the direction of the greatest gradient descent. This results in iterative error reduction.

Training Parameters:

Training cycles: how many times the training cycle is repeated.

Learning rate: the rate of weight change. Larger learning rates converge quicker, but lead to inaccuracies. Smaller learning rates are slower but more accurate.

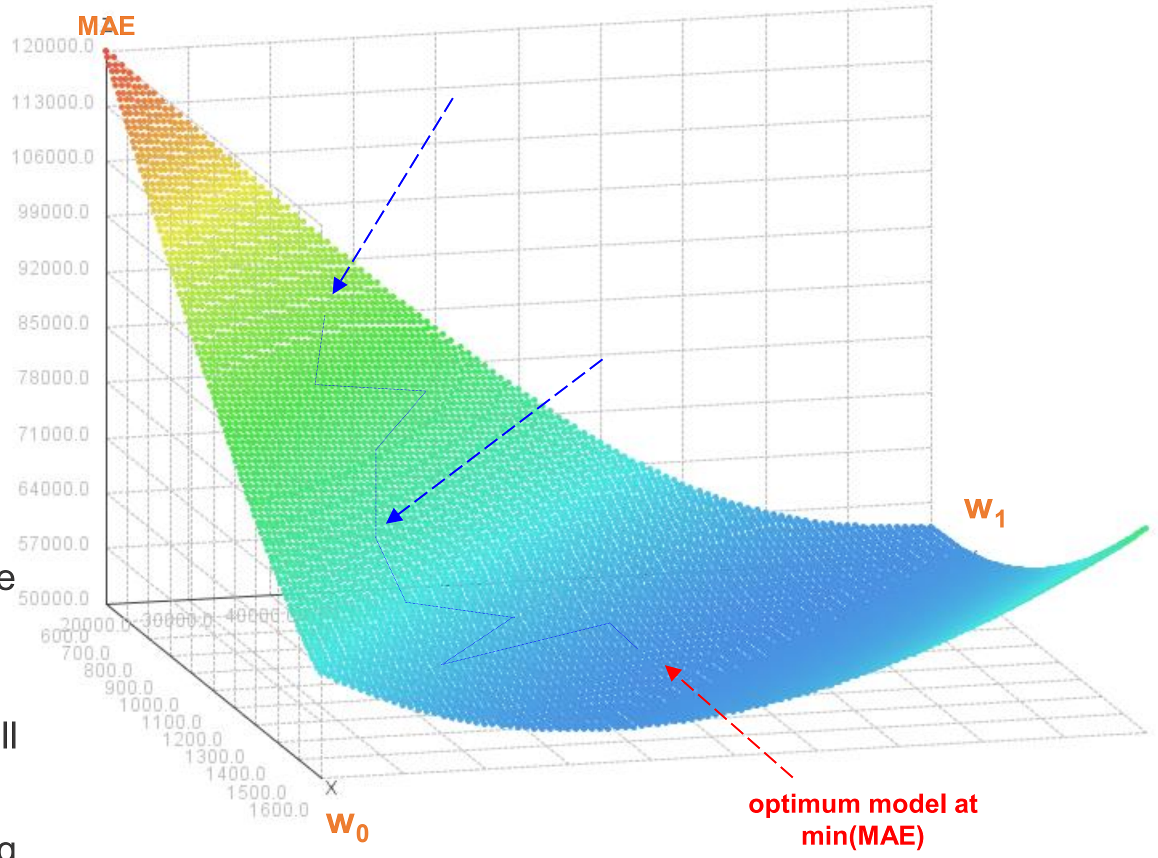
Momentum: represents a fraction of the previous weight to be added to the new weight, it prevents learning getting stuck in local minima.

Decay: The learning rate reduces down to zero over the training cycles so that the learning could be more precise near the minimum.

Epsilon: Is the error threshold at which, the error is considered small enough for learning to stop.

Shuffling: At each cycle the training sample is randomised to avoid sequence learning.

Simple Demonstration of Gradient Descent



Assume that the model is simple with only 2 weights to be estimated

$$w_0 + w_1 \times x_1$$

Deep Learning Software

Software	Creator	First Out	Open Source	Platform	Interface	OpenCL Support	CUDA Support	Petrained Models	RNets	CNN	RBM/ DBNs	Actively Developed
Caffe	Berkeley Vision and Learning Center	2013	Yes	Linux, macOS, Windows	Python, MATLAB, C++	TBD	Yes	Yes	Yes	Yes	No	No
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	2014	Yes	Linux, macOS, Windows, Android (Cross-platform)	Java, Scala, Clojure, Python (Keras), Kotlin	No	Yes	Yes	Yes	Yes	Yes	
Keras	François Chollet	2015	Yes	Linux, macOS, Windows	Python, R	Via a backend	Yes	Yes	Yes	Yes	No	Yes
MATLAB+ Deep Learning Toolbox	MathWorks		No	Linux, macOS, Windows	MATLAB	No	Indirect	Yes	Yes	Yes	Yes	Yes
Microsoft Cognitive Toolkit (CNTK)	Microsoft Research	2016	Yes	Windows, Linux (macOS via Docker)	Python (Keras), C++, Command line	No	Yes	Yes	Yes	Yes	No	No (Azure)
Apache MXNet	Apache Software Foundation	2015	Yes	Linux, macOS, Windows, AWS, Android, iOS, JavaScript	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl, Clojure	TBD	Yes	Yes	Yes	Yes	Yes	Yes
PyTorch	Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan (Facebook)	2016	Yes	Linux, macOS, Windows	Python, C++, Julia	Via separate package	Yes	Yes	Yes	Yes		Yes
TensorFlow (with Keras)	Google Brain	2015	Yes	Linux, macOS, Windows, Android	Python (Keras), C/C++, Java, Go, JavaScript, R, Julia, Swift	Partial	Yes	Yes	Yes	Yes	Yes	Yes
Theano	Université de Montréal	2007	Yes	Cross-platform	Python (Keras)	TBD	Yes	Via model zoo	Yes	Yes	Yes	No

(Wikipedia)

Available in Google Colab

Deep Learning for Tabular Data

❑ Your task is to:

Estimate the house price in Ames (USA) (Regression Problem)

❑ Tool:

Python + Tensorflow (with Keras)

❑ Method:

Multi-Layer Perceptron (MLP)

Ames Real Estate

ExampleSet (2930 examples, 2 special attributes, 79 regular attributes)

Filter (2,930 / 2,930 examples): all

Row No.	PID	SalePrice	MS_SubClass	MS_Zoning	Lot_Frontage	Lot_Area	Street	Alley	Lot_Shape	Land_Contour	Utilities	Lot_Config	Land_Slope	Neighborhood	Condition_1	Condition_2	Bldg_Type	House_Style
1	526301100	215000	20	RL	141	31770	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story
2	526350040	105000	20	RH	80	11622	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Feedr	Norm	1Fam	1Story
3	526351010	172000	20	RL	81	14267	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story
4	526353030	244000	20	RL	93	11160	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story
5	527105010	189900	60	RL	74	13830	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	2Story
6	527105030	195500	60	RL	78	9978	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	2Story
7	527127150	213500	120	RL	41	4920	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	StoneBr	Norm	Norm	TwnhsE	1Story
8	527145080	191500	120	RL	43	5005	Pave	NA	IR1	HLS	AllPub	Inside	Gtl	StoneBr	Norm	Norm	TwnhsE	1Story
9	527146030	236500	120	RL	39	5389	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	StoneBr	Norm	Norm	TwnhsE	1Story
10	527162130	189000	60	RL	60	7500	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	2Story
11	527163010	175900	60	RL	75	10000	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	Gilbert	Norm	Norm	1Fam	2Story
12	527165230	185000	20	RL	?	7980	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	1Story
13	527166040	180400	60	RL	63	8402	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	2Story
14	527180040	171500	20	RL	85	10176	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Gilbert	Norm	Norm	1Fam	1Story
15	527182190	212000	120	RL	?	6820	Pave	NA	IR1	Lvl	AllPub	Corner	Gtl	StoneBr	Norm	Norm	TwnhsE	1Story
16	527216070	538000	60	RL	47	53504	Pave	NA	IR2	HLS	AllPub	CulDSac	Mod	StoneBr	Norm	Norm	1Fam	2Story
17	527225035	164000	50	RL	152	12134	Pave	NA	IR1	Bnk	AllPub	Inside	Mod	Gilbert	Norm	Norm	1Fam	1.5Fin
18	527258010	394432	20	RL	88	11394	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	StoneBr	Norm	Norm	1Fam	1Story
19	527276150	141000	20	RL	140	19138	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	Gilbert	Norm	Norm	1Fam	1Story
20	527302110	210000	20	RL	85	13175	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Norm	1Fam	1Story
21	527358140	190000	20	RL	105	11751	Pave	NA	IR1	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Norm	1Fam	1Story
22	527358200	170000	85	RL	85	10625	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NWAmes	Norm	Norm	1Fam	SFoyer
23	527368020	216000	60	FV	?	7500	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	Somerst	Norm	Norm	1Fam	2Story
24	527402200	149000	20	RL	?	11241	Pave	NA	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	Norm	1Fam	1Story
25	527402250	149900	20	RL	?	12537	Pave	NA	IR1	Lvl	AllPub	CulDSac	Gtl	NAmes	Norm	Norm	1Fam	1Story
26	527403020	142000	20	RL	65	8450	Pave	NA	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	Norm	1Fam	1Story
27	527404120	126000	20	RL	70	8400	Pave	NA	Reg	Lvl	AllPub	Corner	Gtl	NAmes	Norm	Norm	1Fam	1Story

Some conditions with NNs and DLs:

- ❑
- ❑ *All data must be numeric*

categorical variables must be encoded.

- ❑ *Missing values are not permitted*.

With **79 variables** describing (almost) every aspect of residential homes in Ames, Iowa, this neural network application challenges you to predict the **sale price** of each home.

Data Pre-Processing

Data format for ANN models.

- **No Missing values** (e.g., remove attributes, fill-in missing values, imputation)
- Data needs to be scaled to **[0,1]**
- Data processing needs to be applied to both training and testing sets.

Processed Data Set

PID	Order	MS_SubClass	Lot_Frontage	Lot_Area	Overall_Qual	Overall_Cond	Year_Built
534450180	0.044042	0.000000	0.099315	0.027610	0.444444	0.75	0.623188
905101310	0.074087	0.411765	0.174658	0.044301	0.333333	0.50	0.695652
533127080	0.853875	0.235294	0.166376	0.061890	0.777778	0.75	0.876812
908276150	0.307614	0.000000	0.166376	0.035645	0.333333	0.25	0.608696
902330040	0.245476	0.294118	0.342466	0.076520	0.777778	1.00	0.072464
907135040	0.286787	0.000000	0.311644	0.044680	0.444444	0.50	0.884058
534431130	0.209286	0.000000	0.160959	0.039580	0.444444	0.50	0.601449
527455030	0.574257	0.588235	0.010274	0.004693	0.666667	0.75	0.768116
905106210	0.925913	0.000000	0.166376	0.047924	0.444444	0.50	0.695652
528176010	0.015705	0.000000	0.304795	0.060763	0.888889	0.50	0.949275

```
Training shape: (2051, 73)
Training samples:  2051
Validation samples:  879
```

Why does the range on the test (validation) set is not between 0 and 1?

```
X train min = 0.0 ; max = 1.0
X valid min = -0.1111 ; max = 1.3333
```

Multi-Layer Perceptron for Regression

Load required libraries for Deep Learning with Sequential model.

```
import tensorflow as tf
from tensorflow.keras import metrics
from tensorflow.keras import regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Nadam, RMSprop
```

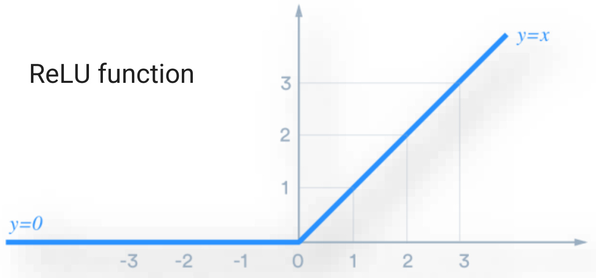
Model evaluation metrics

Improve model generality

Deep Learning Model

Network Layers

Optimizers to train the model



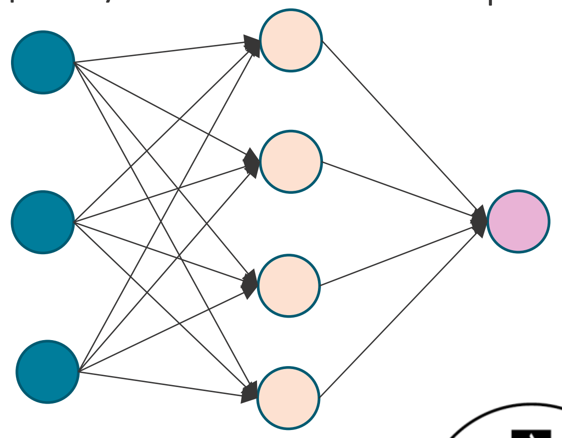

```
def basic_model_1(x_size, y_size):
    t_model = Sequential()
    t_model.add(Dense(100, activation="relu", input_shape=(x_size,)))
    t_model.add(Dense(y_size))
    t_model.compile(
        loss='mean_squared_error',
        optimizer=RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-07, weight_decay=0.0),
        metrics=[metrics.mae])
    return(t_model)
```

Hidden (Dense) layers

Out layers

Setup Model Optimizer

Conceptual MLP Architecture For Regression



EFMD EQUIS ACCREDITED AACSB ACCREDITED DEAKIN BUSINESS SCHOOL

Multi-Layer Perceptron for Regression (cont.)

The created MLP model

Initialize shape of input data and out labels

```
model = basic_model_1(arr_x_train.shape[1], arr_y_train.shape[1])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               7400

 dense_1 (Dense)             (None, 1)                 101

=================================================================
Total params: 7,501
Trainable params: 7,501
Non-trainable params: 0
_________________________________________________________________
```

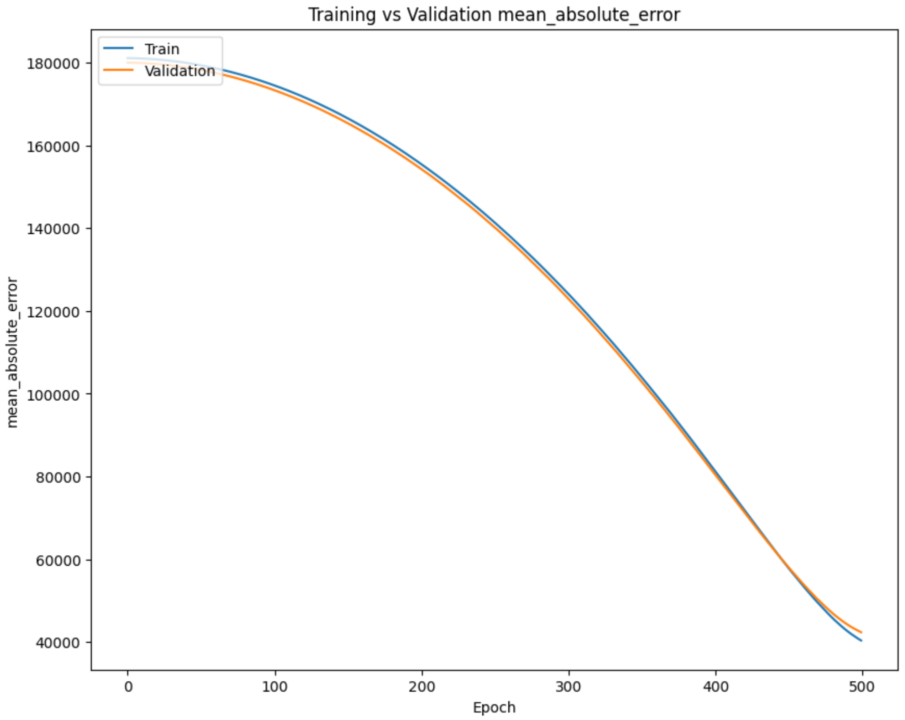
(73 input predictors + 1 bias) x 100 hidden nodes

(100 hidden nodes previous layer) + 1 bias) x 1 output node

Evaluate MPL model

```
train_score = model.evaluate(arr_x_train, arr_y_train, verbose=0)
valid_score = model.evaluate(arr_x_valid, arr_y_valid, verbose=0)
```

```
Train MAE:  40228.7 , Train Loss:  4453106176.0
Val MAE:  42337.91 , Val Loss:  4594756096.0
```



Train the model

```
history = model.fit(arr_x_train, arr_y_train,
    batch_size=64,
    epochs=500,
    shuffle=True,
    verbose=2,
    validation_data=(arr_x_valid, arr_y_valid),
    callbacks=keras_callbacks)
```

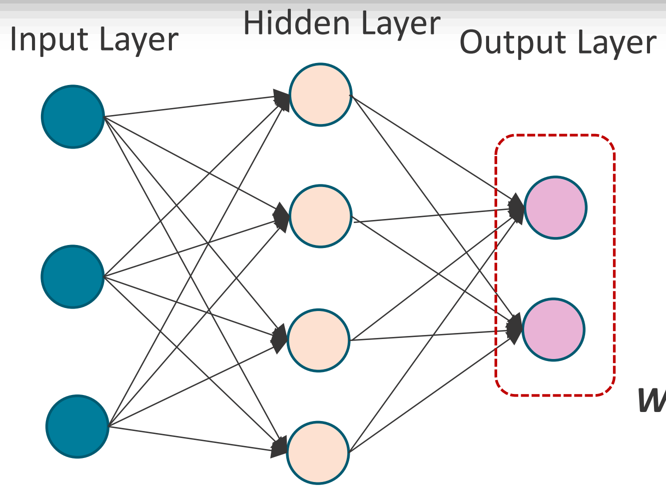
Multi-Layer Perceptron for Classification

default_of_credit_card_clients data from Kaggle

23 attributes + 1 label (**dpnm** column - 1 for default, and 0 for not-default)

ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	dpnm
1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	0	0	0	689	0	0	0	0	1
2	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3272	3455	3261	0	1000	1000	1000	0	2000	1
3	90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
4	50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
5	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	20940	19146	19131	2000	36681	10000	9000	689	679	0
6	50000	1	1	2	37	0	0	0	0	0	0	64400	57069	57608	19394	19619	20024	2500	1815	657	1000	1000	800	0
7	500000	1	1	2	29	0	0	0	0	0	0	367965	412023	445007	542653	483003	473944	55000	40000	38000	20239	13750	13770	0
8	100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380	601	221	-159	567	380	601	0	581	1687	1542	0
9	140000	2	3	1	28	0	0	2	0	0	0	11285	14096	12108	12211	11793	3719	3329	0	432	1000	1000	1000	0
10	20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0	0	0	13007	13912	0	0	0	13007	1122	0	0

Conceptual MLP Architecture For Classification




```
# convert class vectors to binary class matrices
arr_y_train = to_categorical(arr_y_train, 2)
arr_y_valid = to_categorical(arr_y_valid, 2)
```

Use One-hot Coding for labels
Positive class: [1, 0]
Negative class: [0, 1]

```
Train shape: x= (21000, 23) , y= (21000, 2)
Test shape: x= (9000, 23) , y= (9000, 2)
```

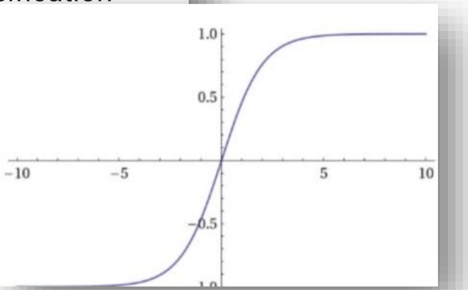
Why 2 output nodes?

Multi-Layer Perceptron for Classification (cont.)

```
def basic_model_1():
    t_model = Sequential()
    t_model.add(Dense(100, activation="relu", input_shape=(23,)))
    t_model.add(Dense(2, activation='softmax'))
    t_model.summary()
    return(t_model)
```

softmax activation function for classification

Out layers
(2 nodes for 2 classes)



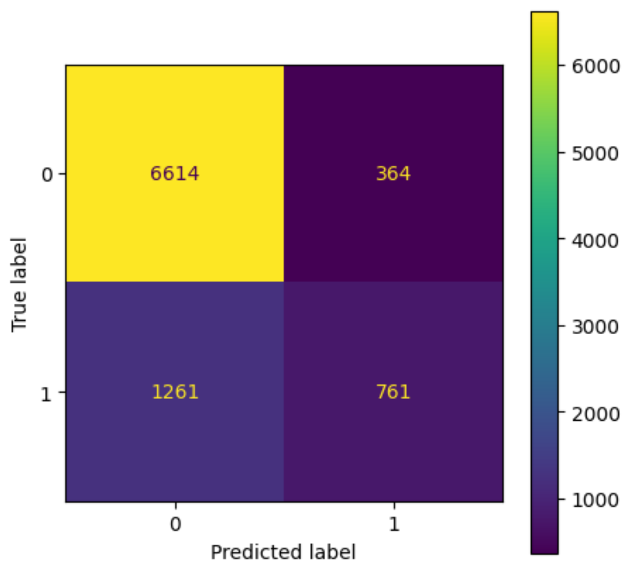

```
Train Accuracy:  0.83 , Train Loss:  0.41
Val Accuracy:  0.82 , Val Loss:  0.44
```

```
The result of Kappa is : 0.385
The result of the classification report is:
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      6978
           1       0.68      0.38      0.48      2022

    accuracy                           0.82      9000
   macro avg       0.76      0.66      0.69      9000
weighted avg       0.80      0.82      0.80      9000
```

(23 input predictors + 1 bias) x 100 hidden nodes

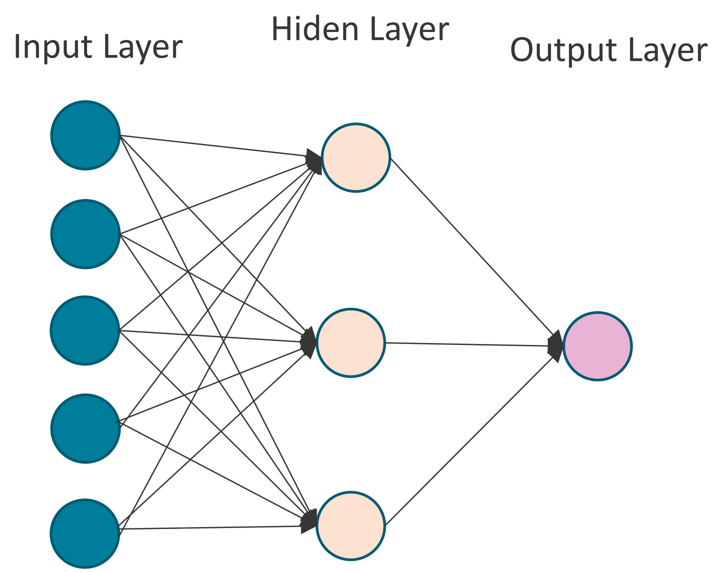
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               2400

 dense_1 (Dense)             (None, 2)                 202

=================================================================
Total params: 2,602
Trainable params: 2,602
Non-trainable params: 0
_________________________________________________________________
```

(100 hidden nodes previous layer) + 1 bias) x 2 output node



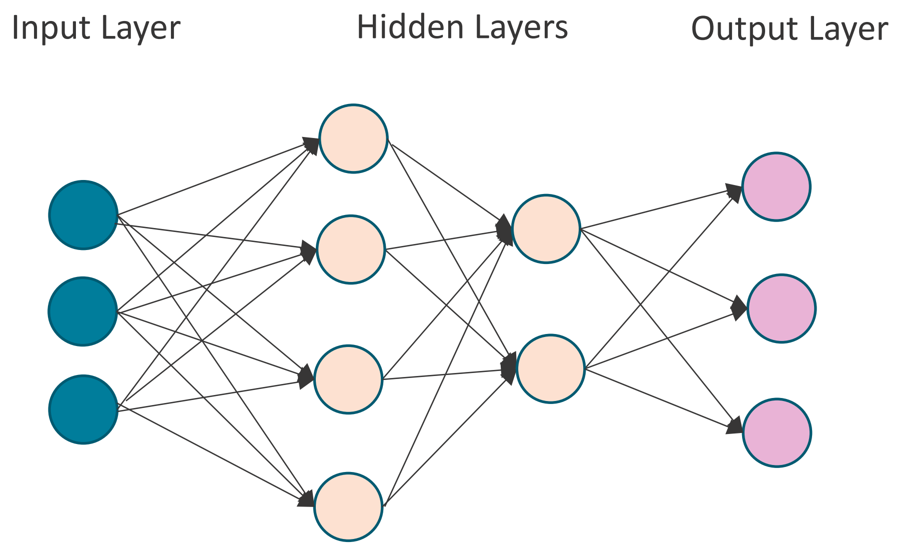
Exercises

Draw the architecture of an ANN model for predicting car price based on 5 features (e.g., number of door, number of seats, engine size). The hidden layer has 3 hidden nodes.



How many parameters (weights) to be estimated for this model?

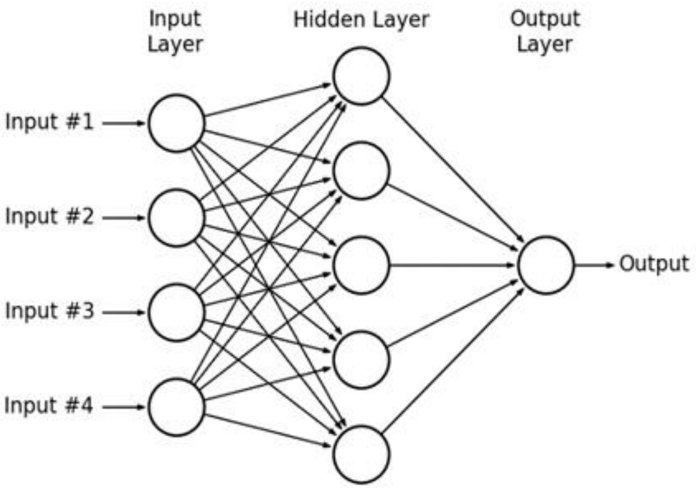
Draw the architecture of an ANN model for predicting 3 types of vehicles (e.g., car, truck, train) based on 3 features (e.g., number of wheels, number of seats, engine size). There are 2 hidden layers (4 and 2 nodes) respectively.

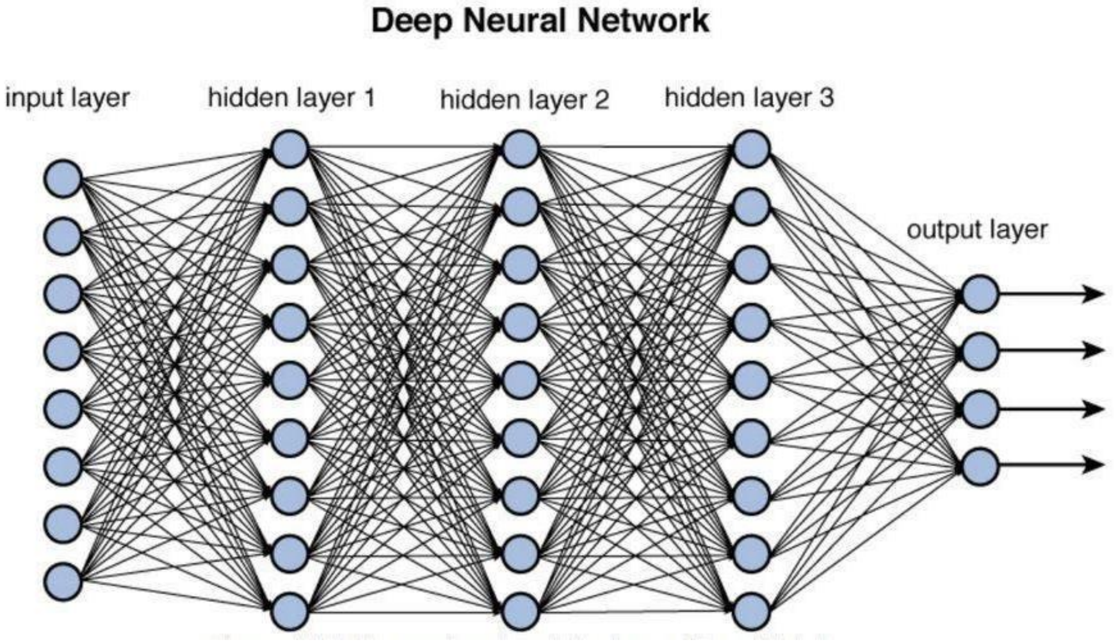


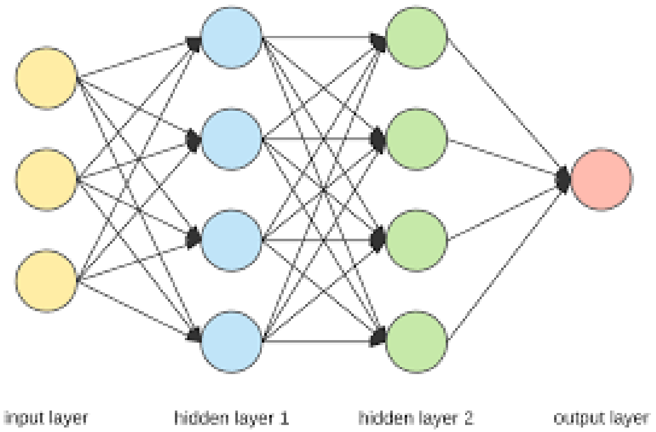
How many parameters (weights) to be estimated for this model?

Discussion Question

How many hidden layers/how many hidden nodes should be specified for ANN models?







Discussion Question

When considering an ANN…

Which statement/s is/are not correct?

- A: The only part in an ANN that does not have interactions with the outside world is the input layer.
- B: Can ANN have only one hidden layer with one hidden node?
- C: The SoftMax function is not needed if ANN is used for regression modelling.
- D: The non-linear relationship between predictors and outcome can be modelled.

In this lecture, we have covered:

- Introduction to the concepts of deep learning and neural networks.
- Applications of deep learning to tabular data for regression and classification.

Summary