EI320A(3) 深度學習使用 Python

Instructors

Tipajin Thaipisutikul (t.greentip@gmail.com)

Prof. Huang-Chia Shih (hcshih@Saturn.yzu.edu.tw)

Course Syllabus

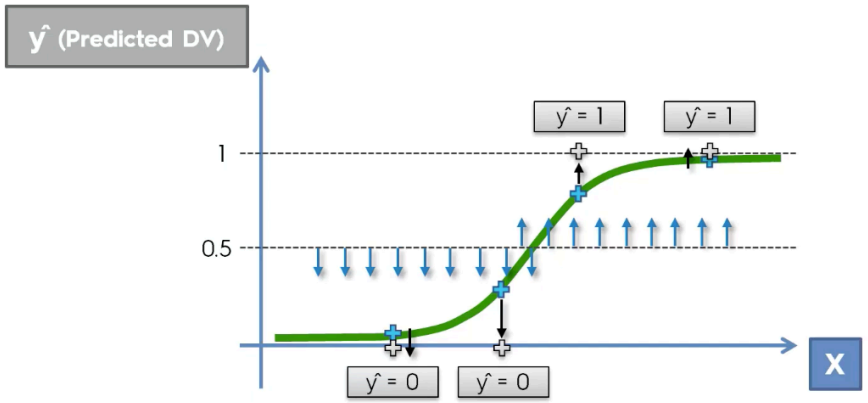
Evaluation Criteria

Tasks	Percentage
In Class Hands-on	60
Project Proposal Presentation	10
Project Final Presentation	30
Bonus (In Class Participation)	5
Total	110/100

Week	Date	Content	Note	Total
1	2/26	Welcome to the course	Download & Install Anaconda Homework (1)	1
2	3/5	Crash Course of Python, Numpy, Pandas, and Matplotlib	In class hands-on (4)	5
3	3/12	Get to know about Data ML:Classification Models	In class hands-on (5)	10
4	3/19	ML:Regression Models	In class hands-on (5)	15
5	3/26	ML:Clustering /Apriori Models	In class hands-on (5)	20
6	4/2	Holiday		
7	4/9	Introduction to Deep Learning (ANN)	In class hands-on (5)	25
8	4/16	Convolutional Neural Network (CNN)	In class hands-on (5)	30
9	4/23	Convolutional Neural Network (CNN)	In class hands-on (5)	35
10	4/30	Recurrent Neural Network (RNN)	In class hands-on (5)	40
11	5/7	Recurrent Neural Network (RNN)	In class hands-on (5)	45
12	5/14	Project Proposal Presentation	Proposal Presentation (10)	55
13	5/21	Time series with DNN, CNN, RNN	In class hands-on (5)	60
14	5/28	Attention Neural Network	In class hands-on (5)	65
15	6/4	Generative Adversarial Network (GAN)	In class hands-on (5)	70
16	6/11	Reinforcement Learning (RL)	In class hands-on (5)	75
17	6/18	Final Project Presentation	Final Presentation (30)	105

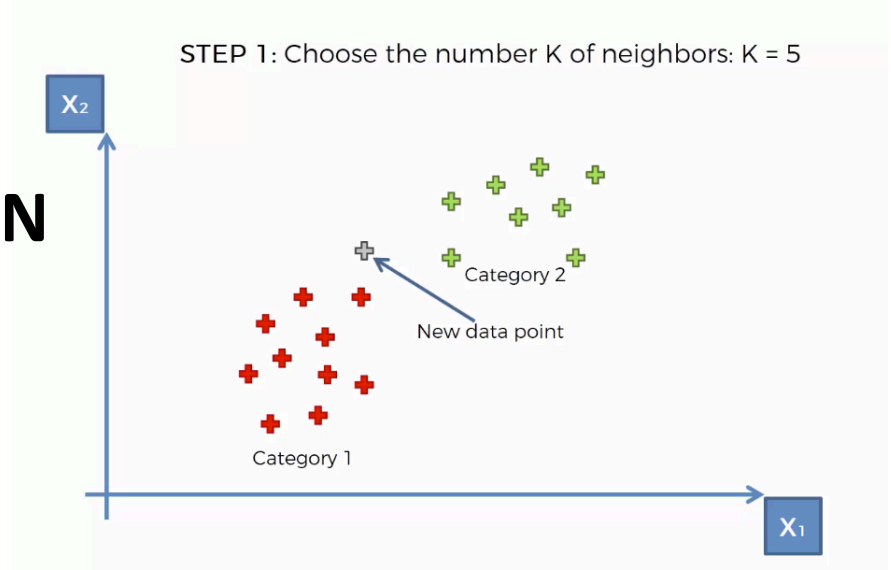
Bonus: 5 For class participation.

ML (Classification)

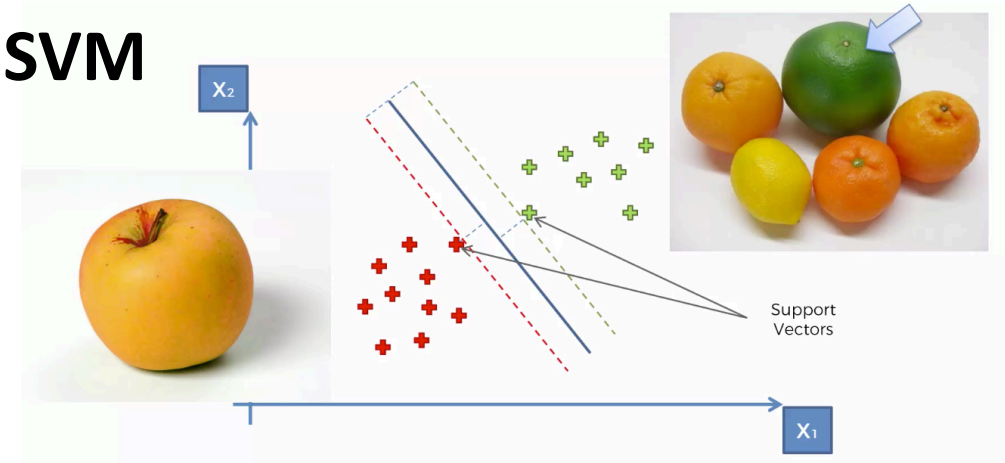


1. Logistic Regression Classifier

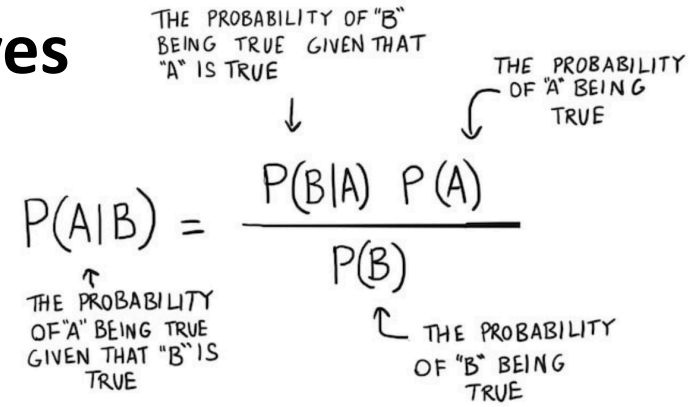
2. KNN



3. SVM

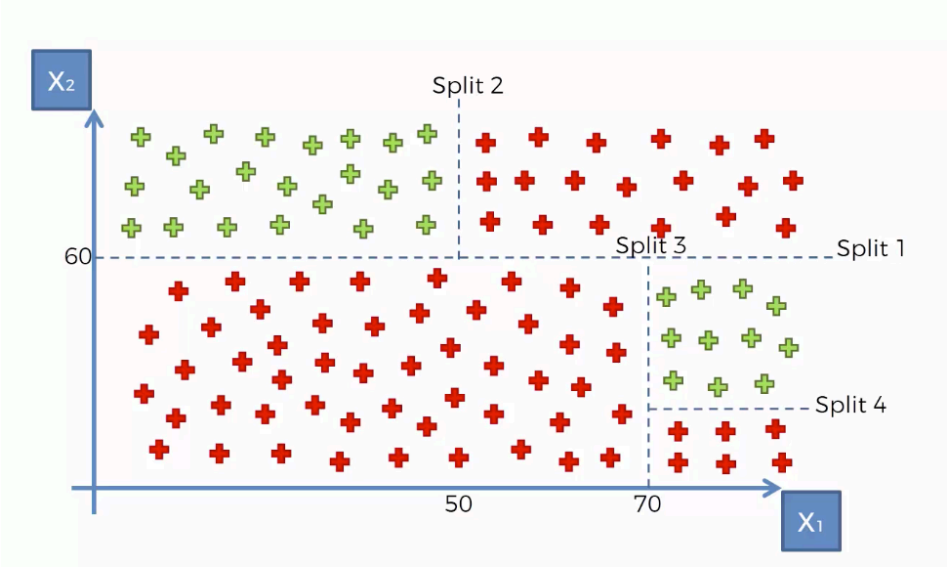


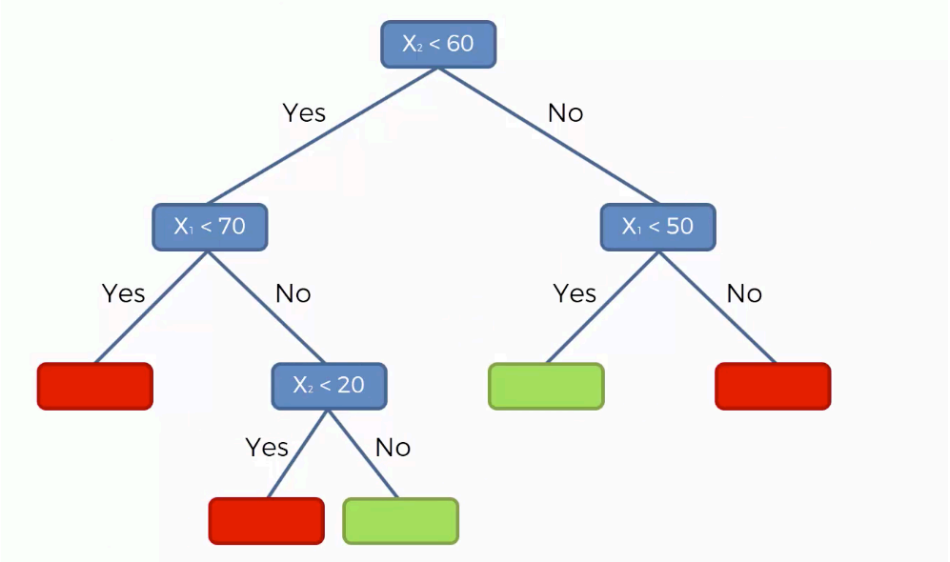
4. Naïve Bayes



ML (Classification)

5. DT





6. Naïve Bayes

STEP 1: Pick at random K data points from the Training set.

STEP 2: Build the Decision Tree associated to these K data points.

STEP 3: Choose the number Ntree of trees you want to build and repeat STEPS 1 & 2

STEP 4: For a new data point, make each one of your Ntree trees predict the category to which the data points belongs, and assign the new data point to the category that wins the majority vote.

ML (Regression)

Linear Regression:

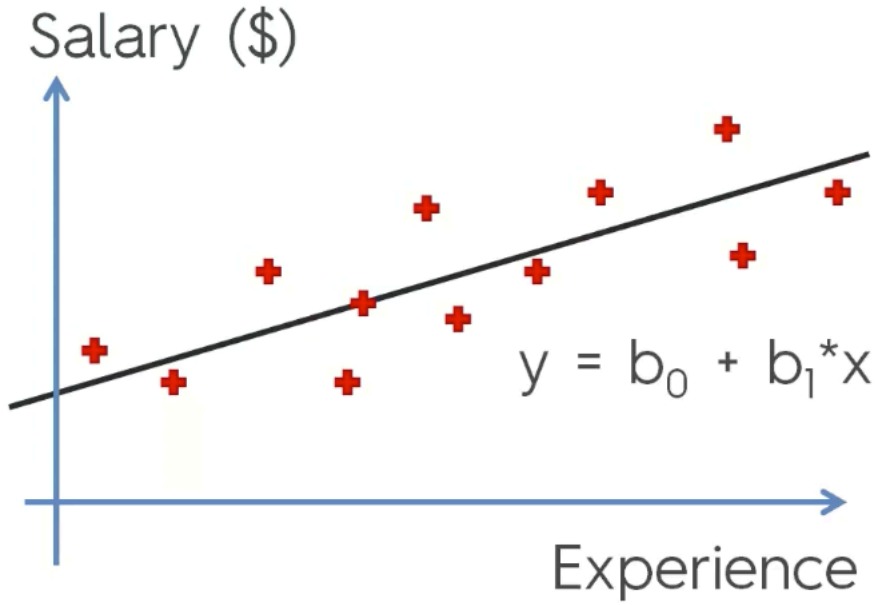
- **Simple:**

$$y = b_0 + b_1 * x$$

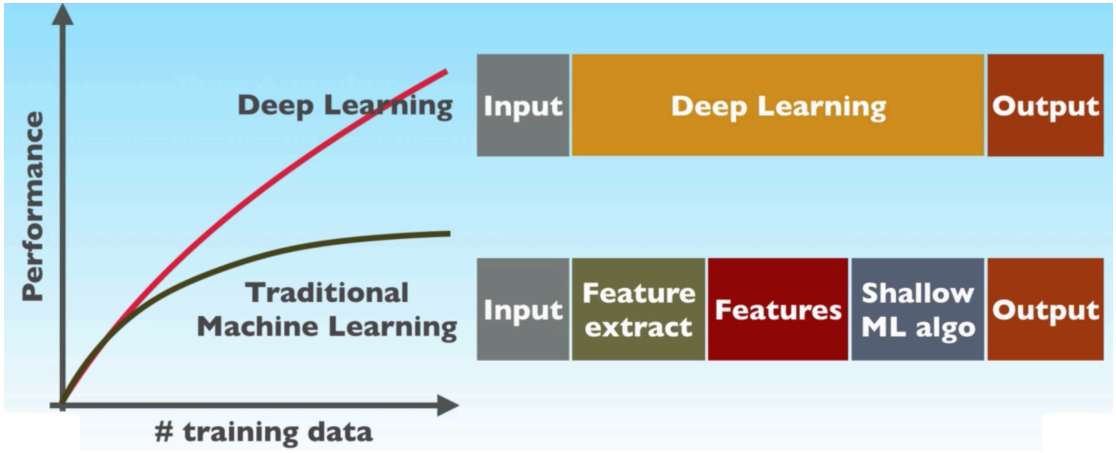
- **Multiple:**

$$y = b_0 + b_1 * x_1 + \ldots + b_n * x_n$$

We know this:

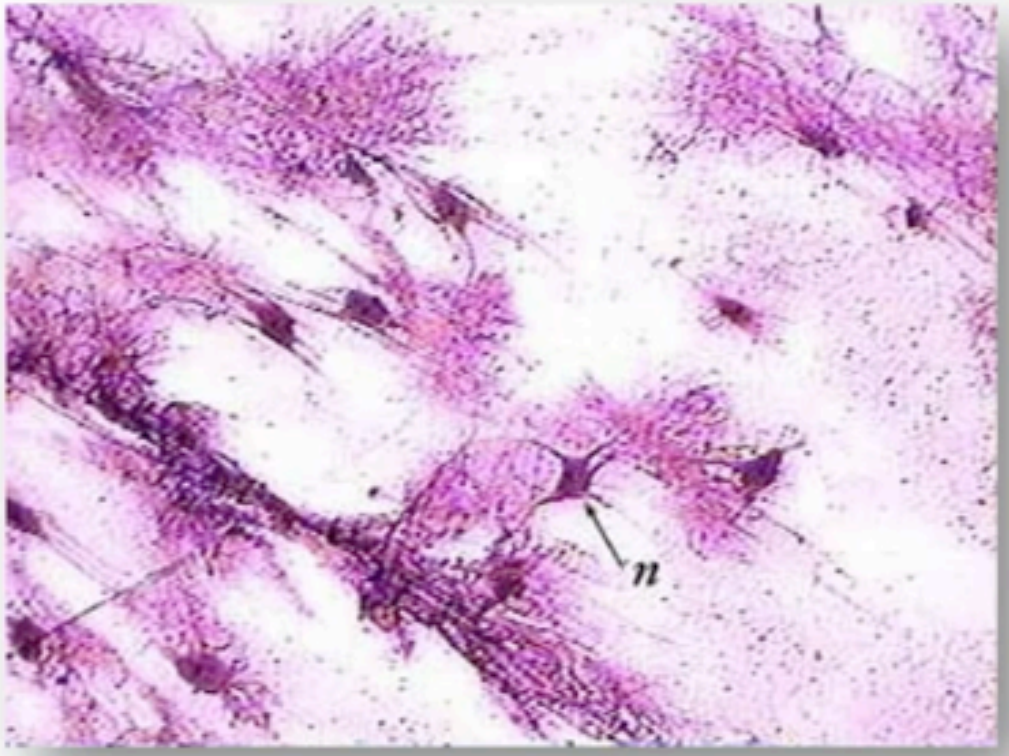


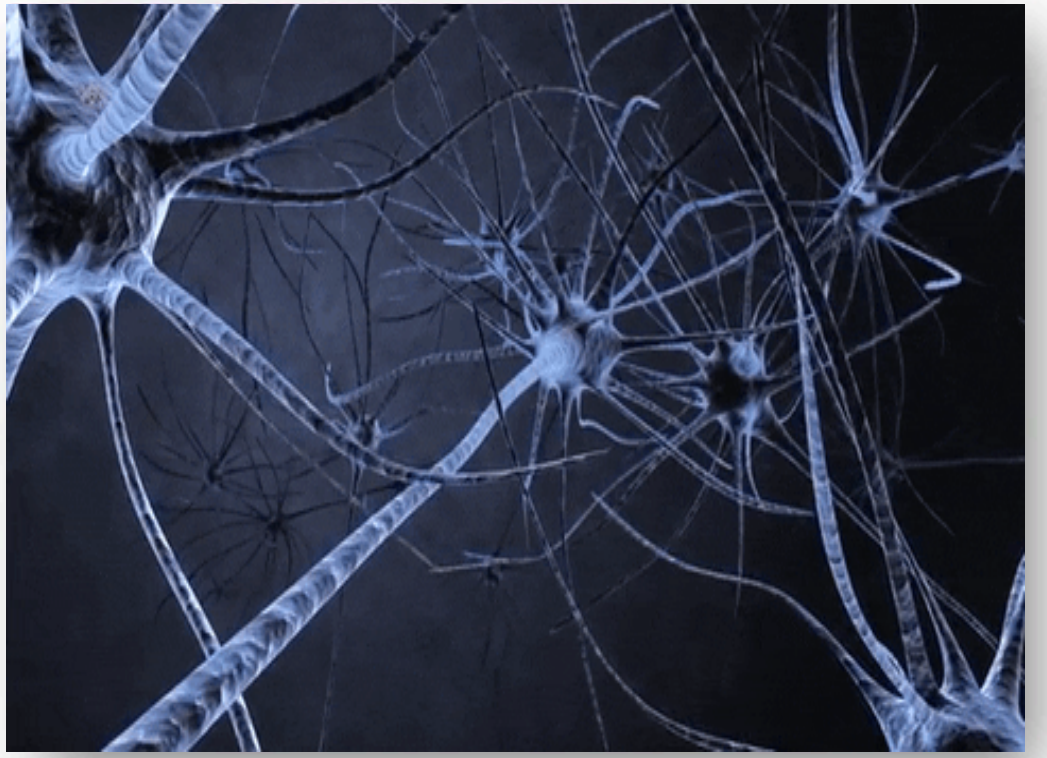
ANN Outline



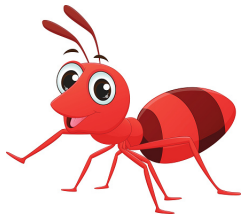
- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

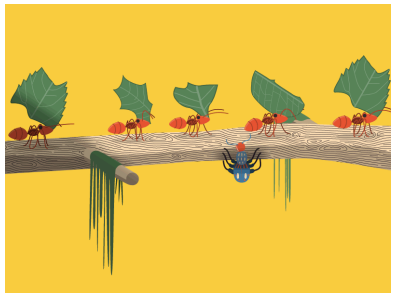
The neuron or perceptron

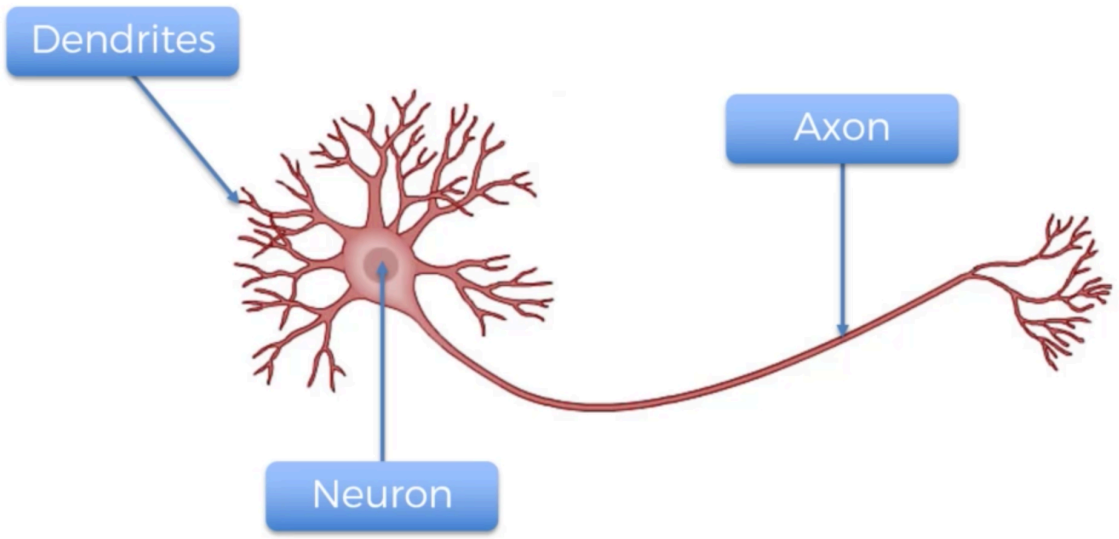


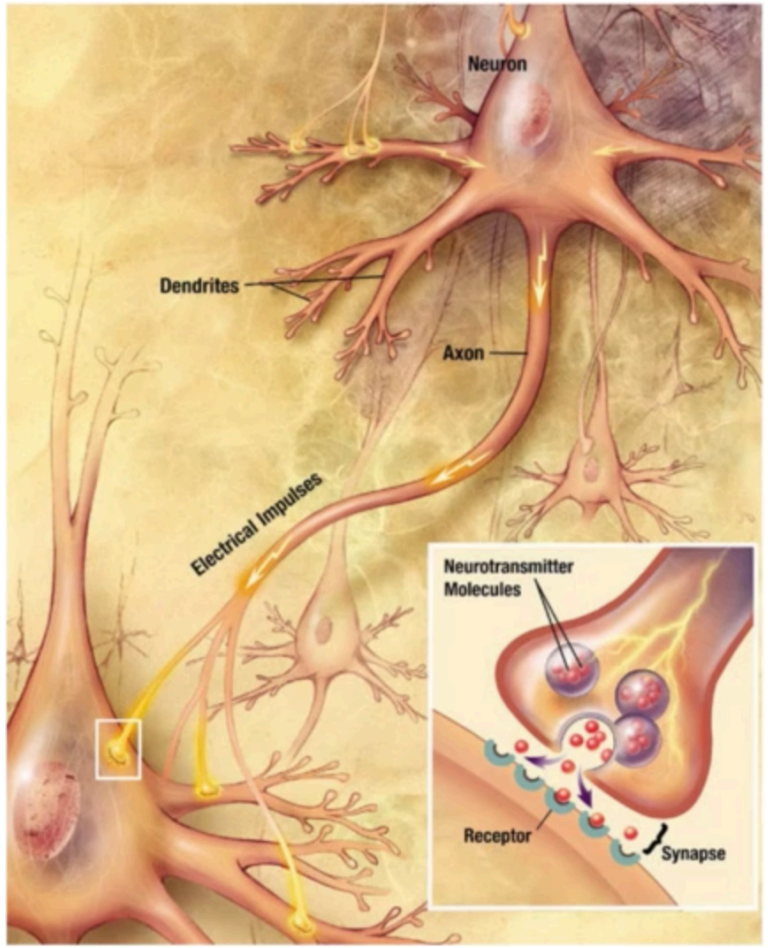


The neuron

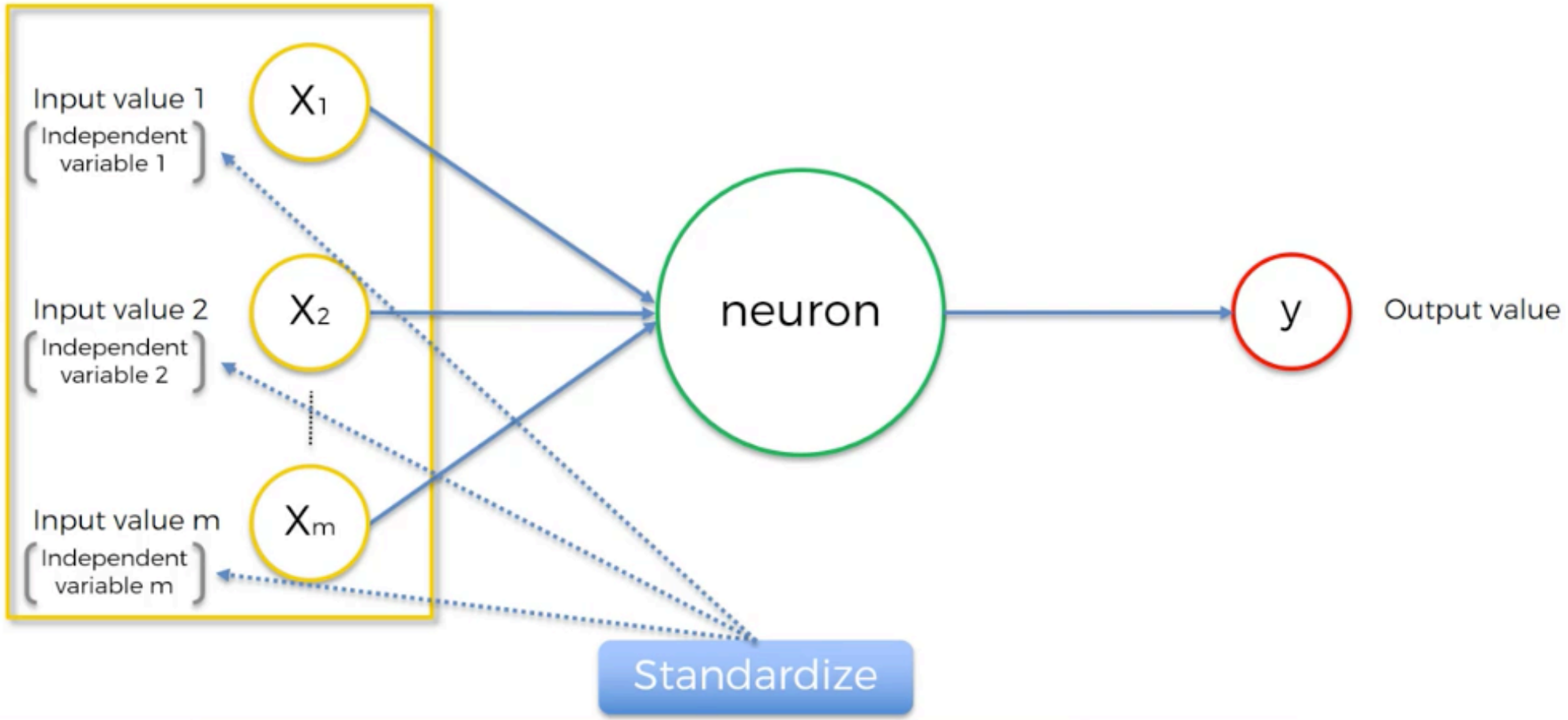




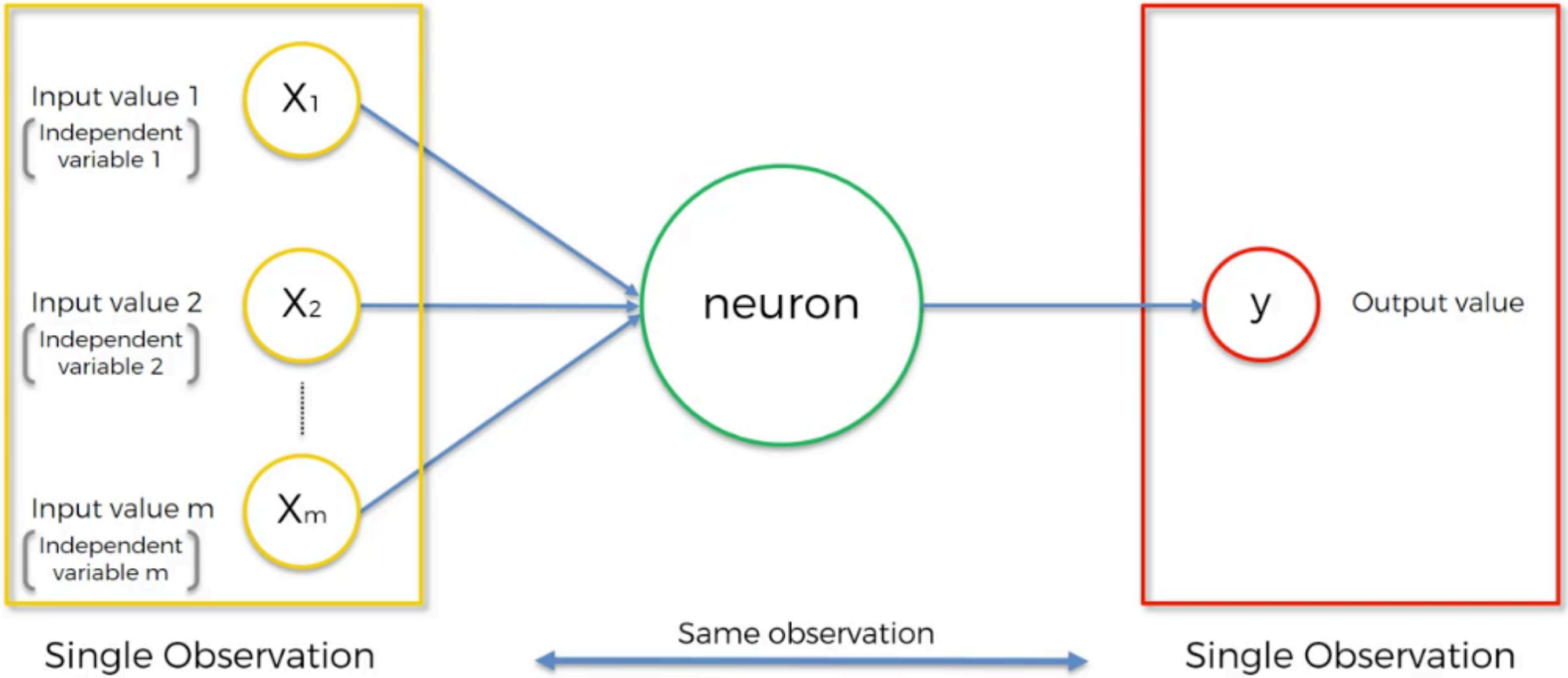

Dendrites
Axon
Neuron


Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

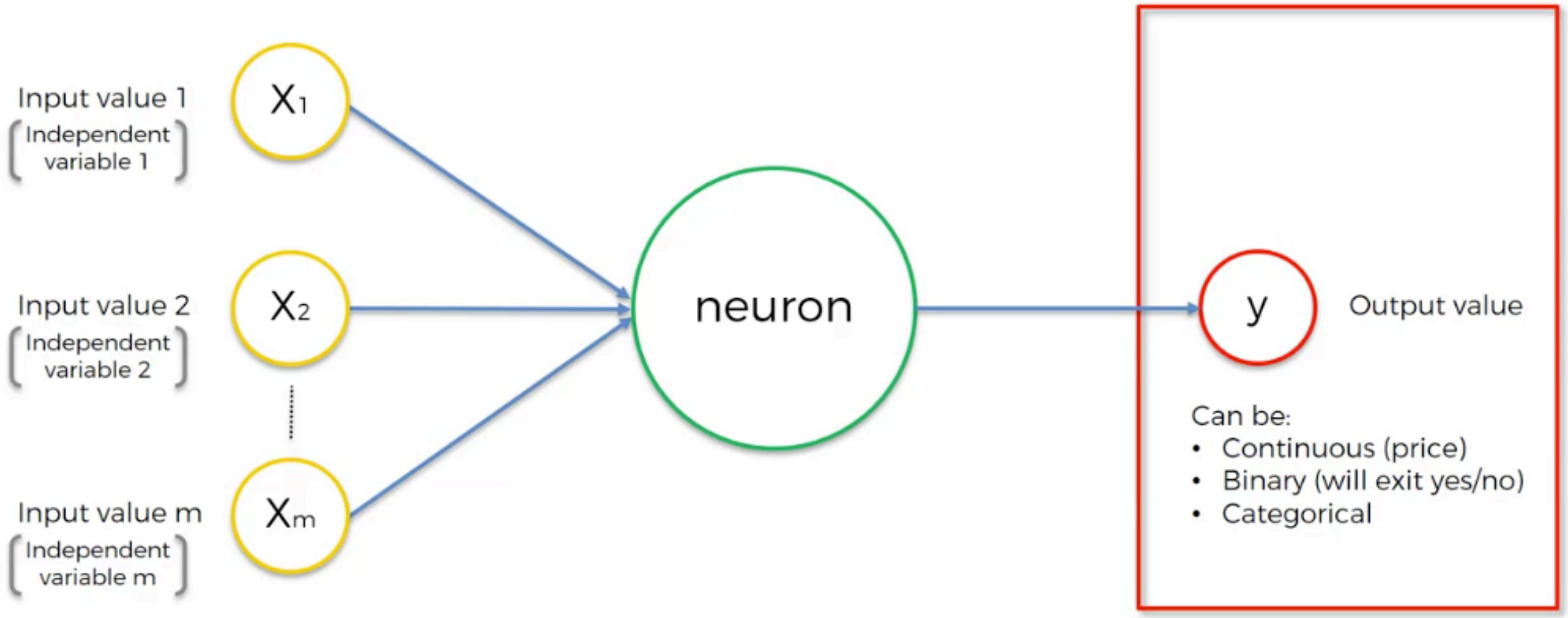
The neuron


Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Standardize

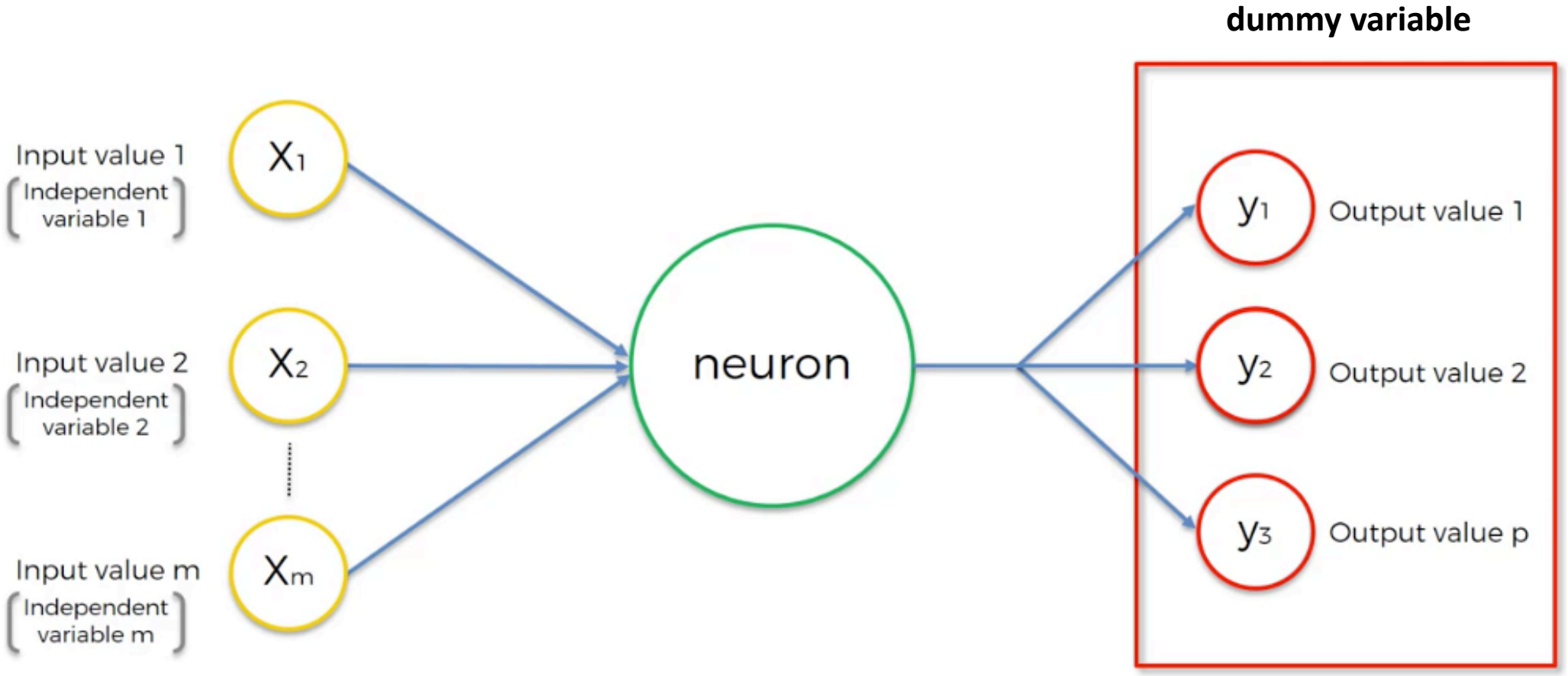
The neuron


Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Single Observation
Same observation
Single Observation

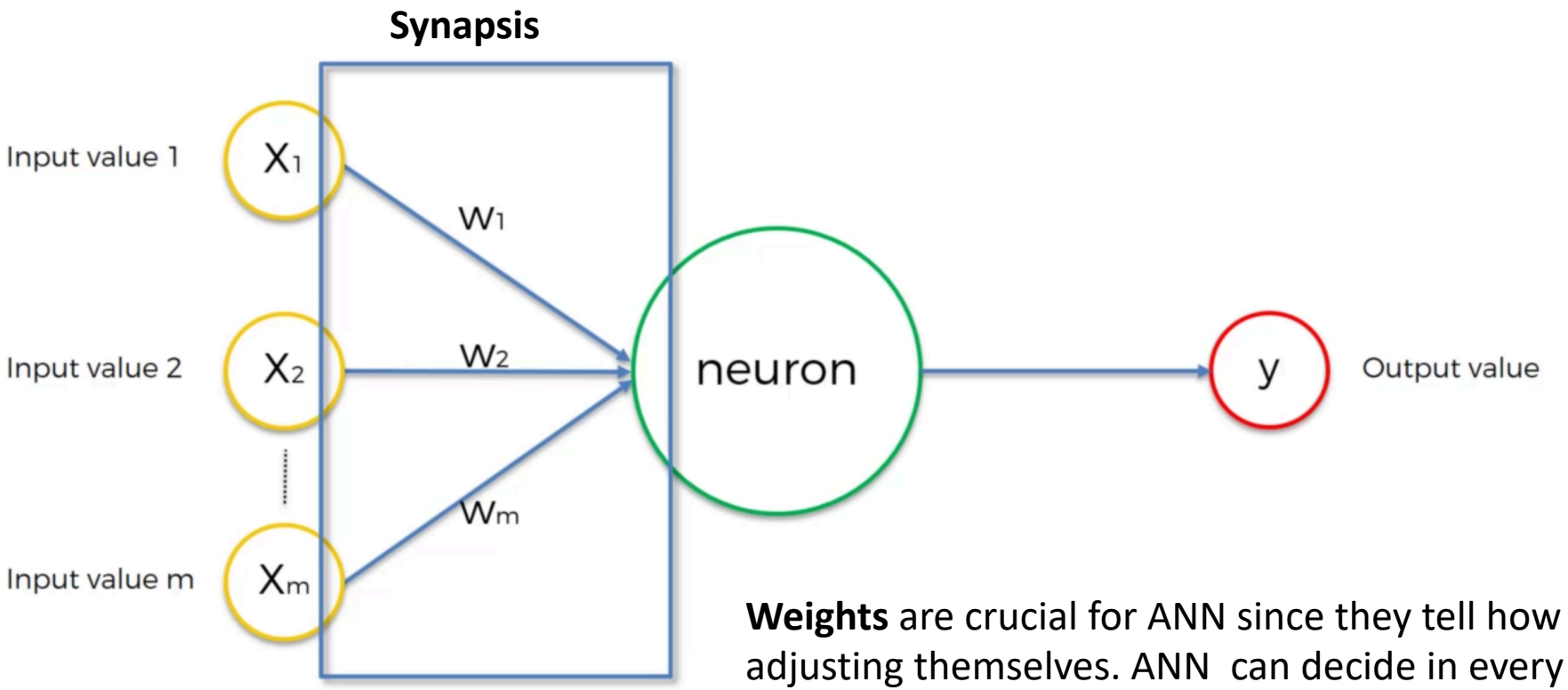
The neuron


Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y
Output value
Can be:
• Continuous (price)
• Binary (will exit yes/no)
• Categorical

The neuron


dummy variable
Input value 1
Independent variable 1
X1
Input value 2
Independent variable 2
X2
Input value m
Independent variable m
Xm
neuron
y1
Output value 1
y2
Output value 2
y3
Output value p

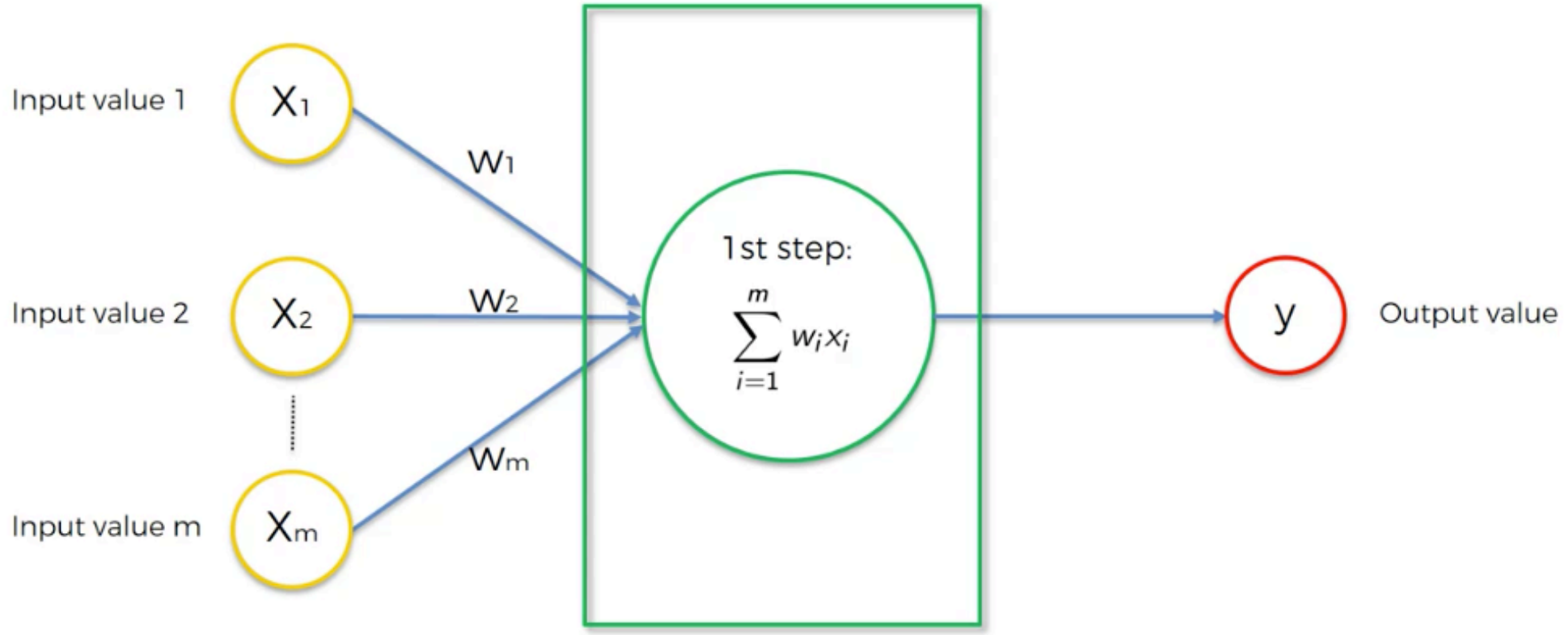
The neuron



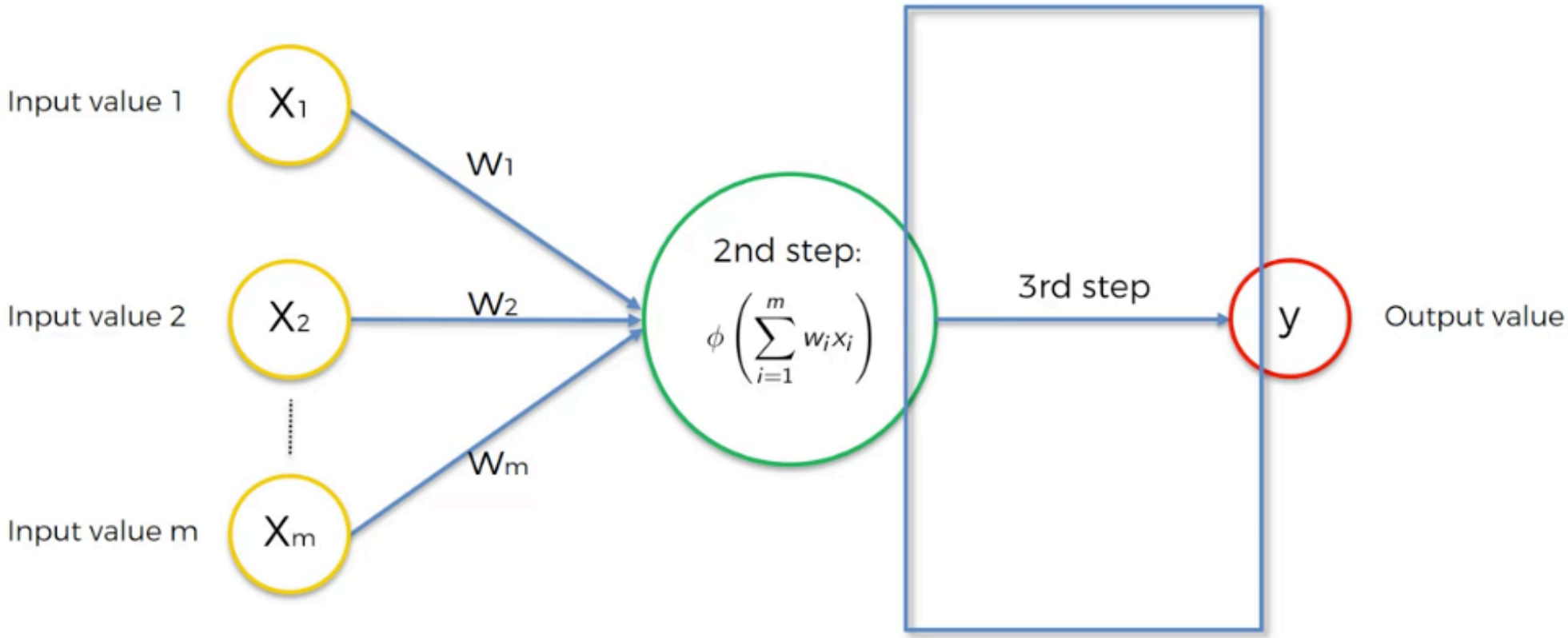
Weights are crucial for ANN since they tell how NN learn by adjusting themselves. ANN can decide in every single case what signal is poor/strength to pass along. [Training = adjusting the weights]

The neuron

(+ Bias for each neuron)



The neuron


Input value 1
X1
W1
Input value 2
X2
W2
Input value m
Xm
Wm
2nd step:
$\phi\left(\sum_{i=1}^{m} w_i x_i\right)$
3rd step
y
Output value

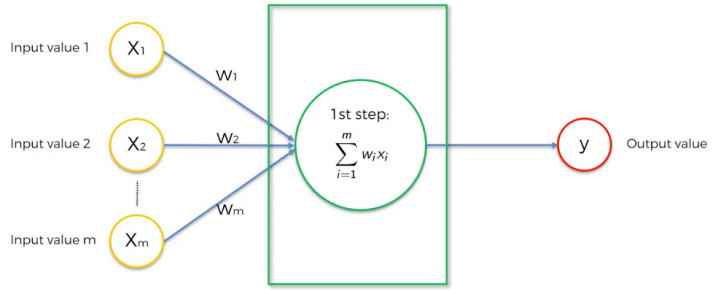
ANNs Plan of Attack

- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

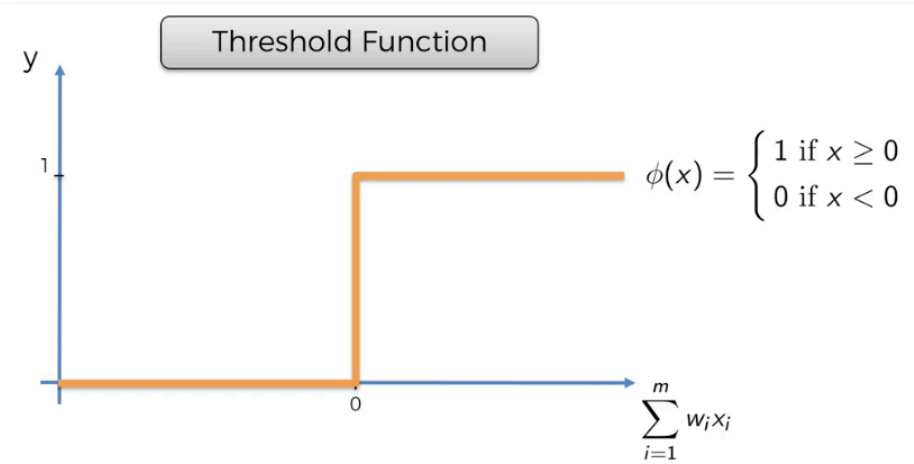
Key Takeaways:

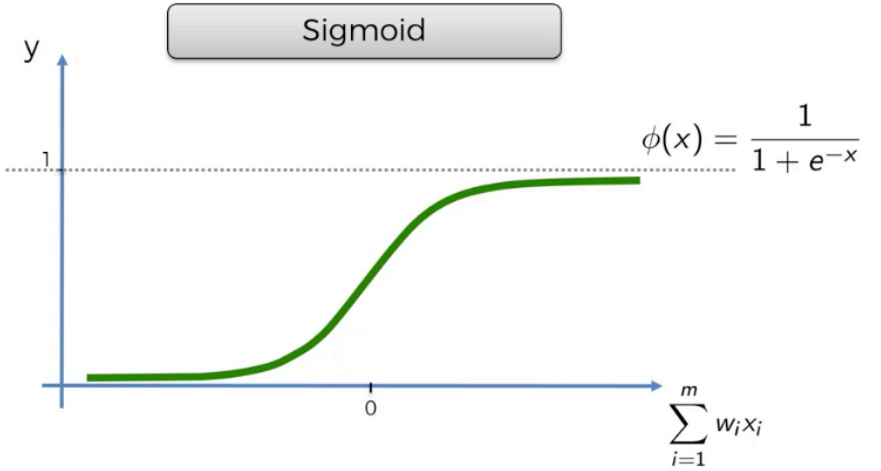
1. Activation Fn. Makes NNs non-linear
2. Non-linearities are the secret of NNs

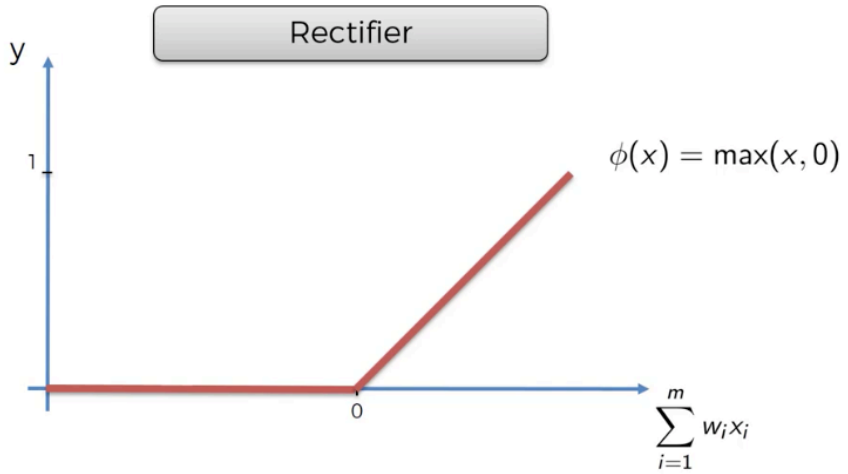
Binary Output: Threshold, Sigmoid

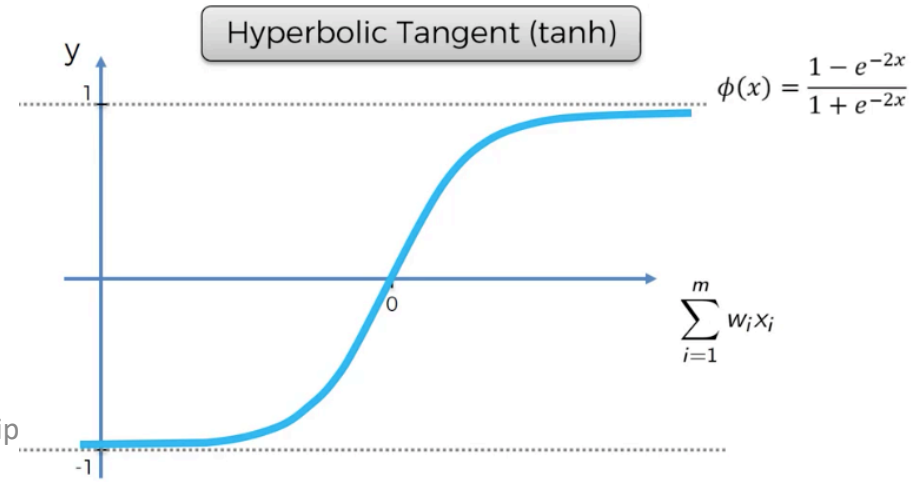


The Activation Function

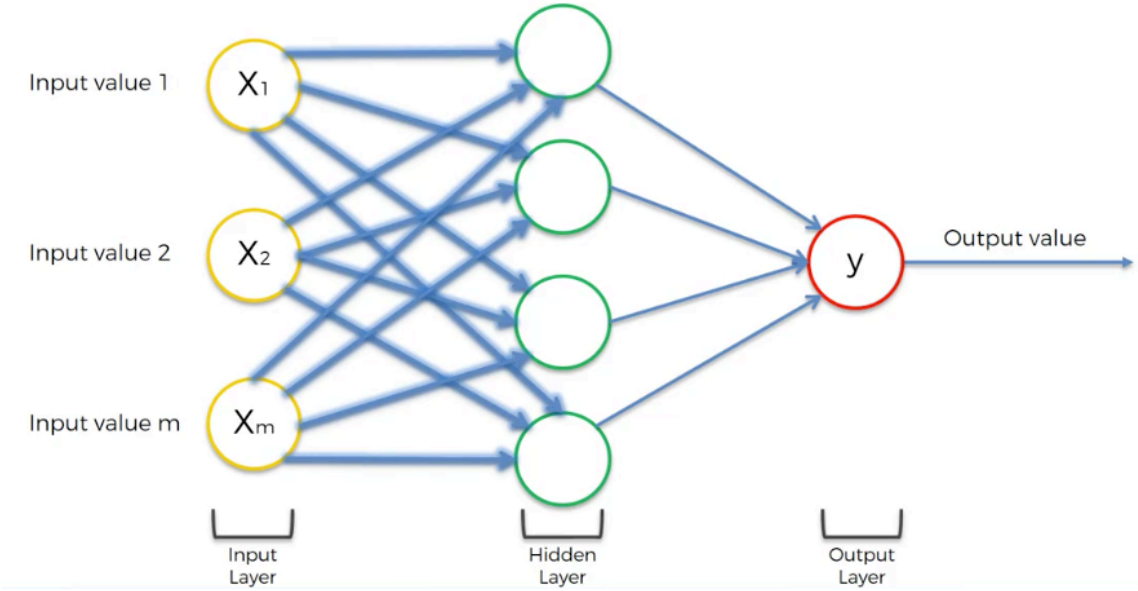


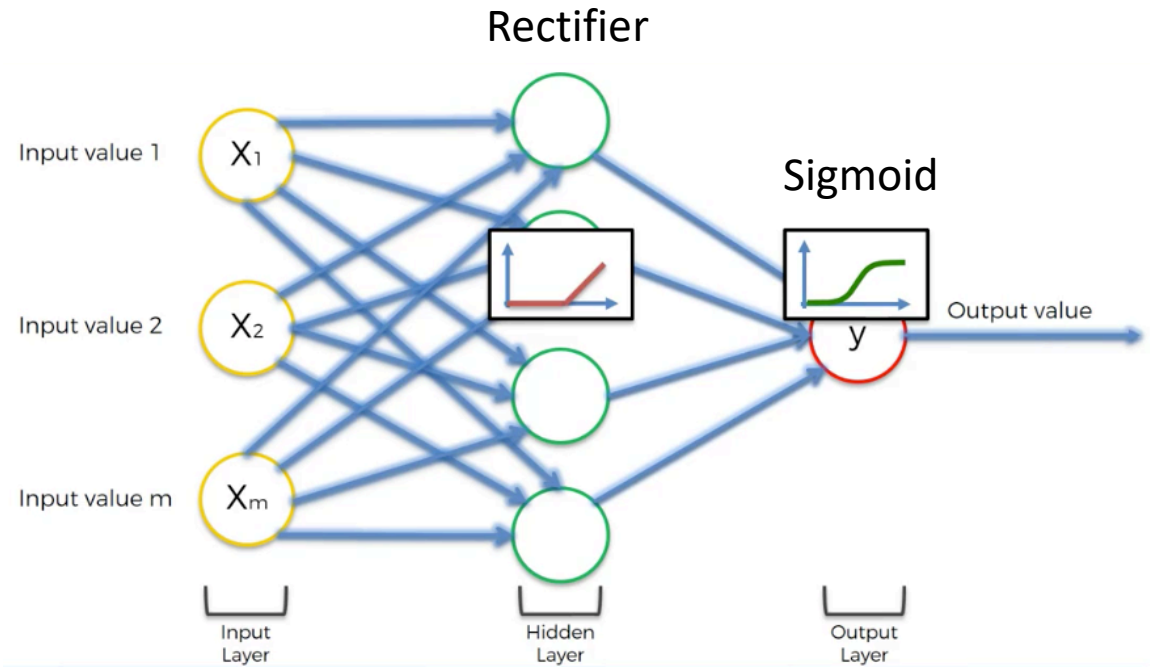






The Activation Function



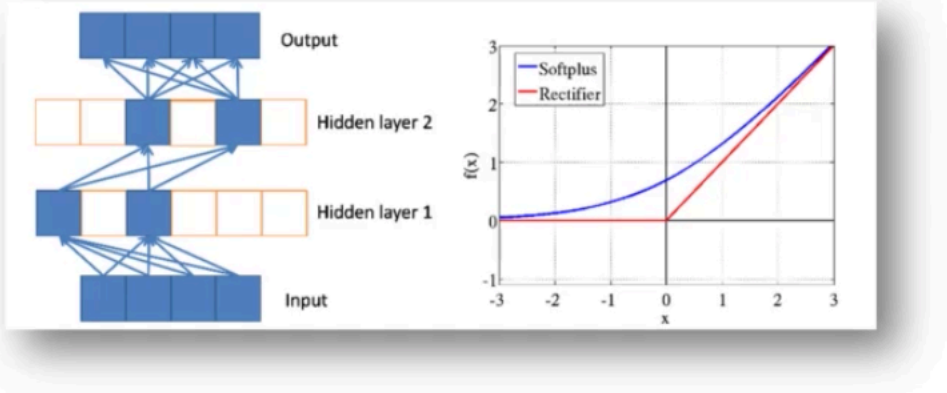


The Activation Function

Additional Reading:

Deep sparse rectifier neural networks

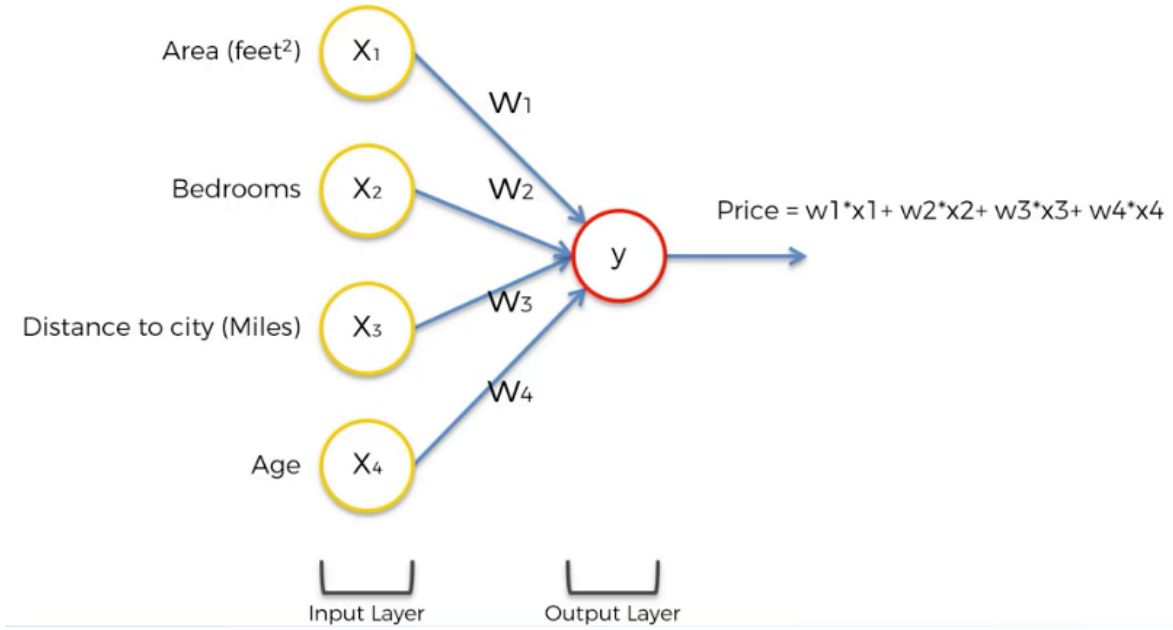
By Xavier Glorot et al. (2011)



ANNs Plan of Attack

- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

How do Neural Networks work?

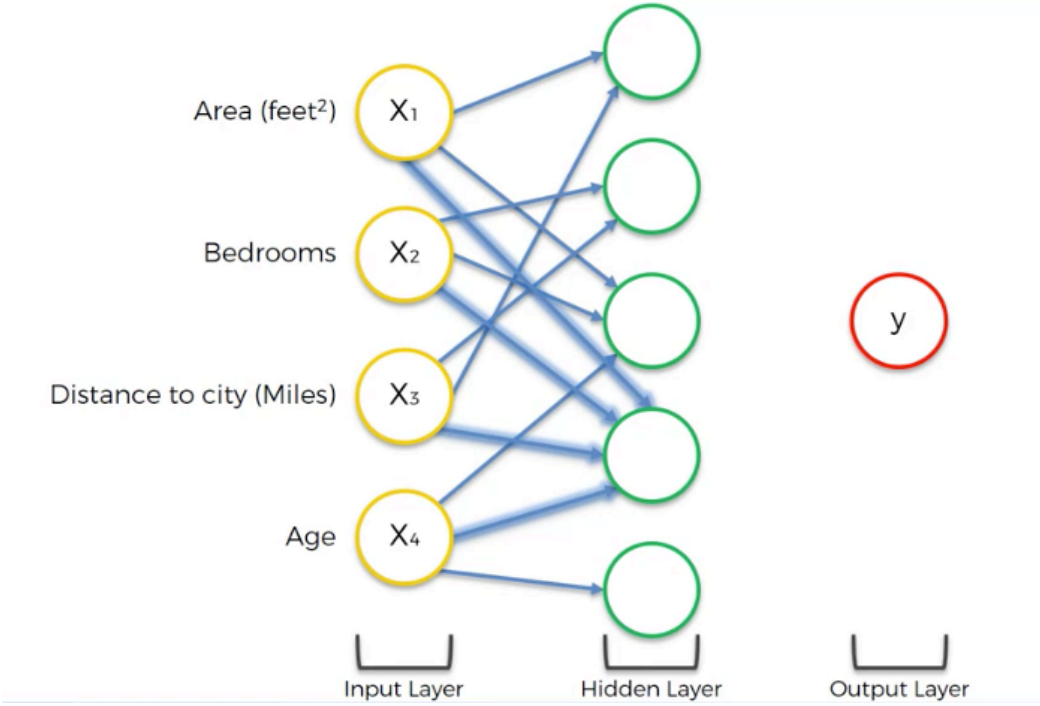


How do Neural Networks work?

How do Neural Networks work?

How do Neural Networks work?

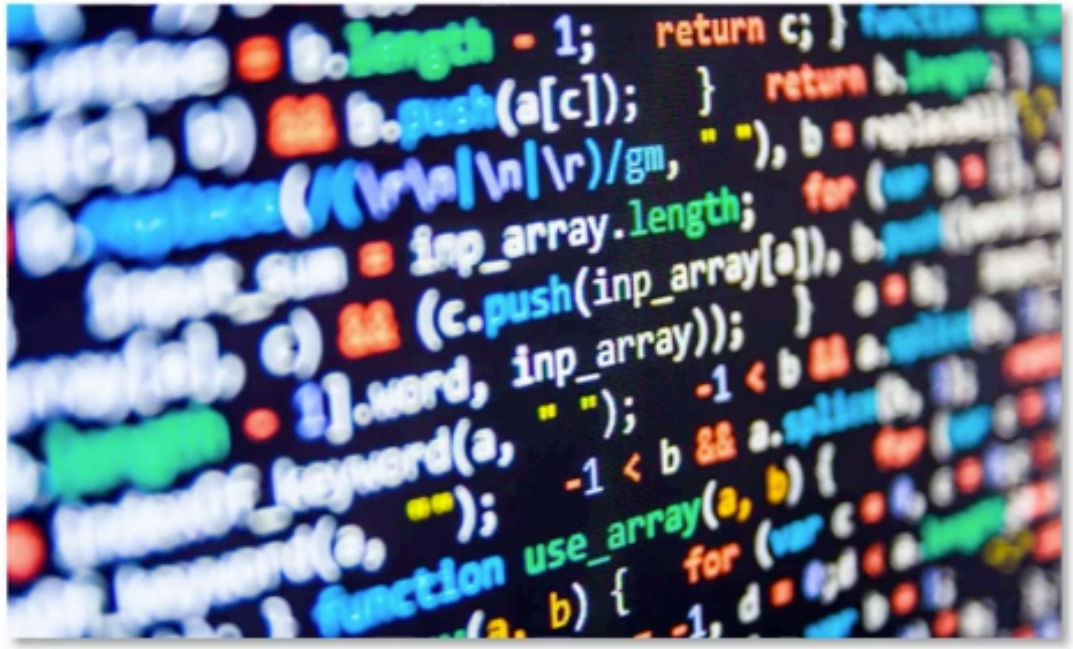
How do Neural Networks work?



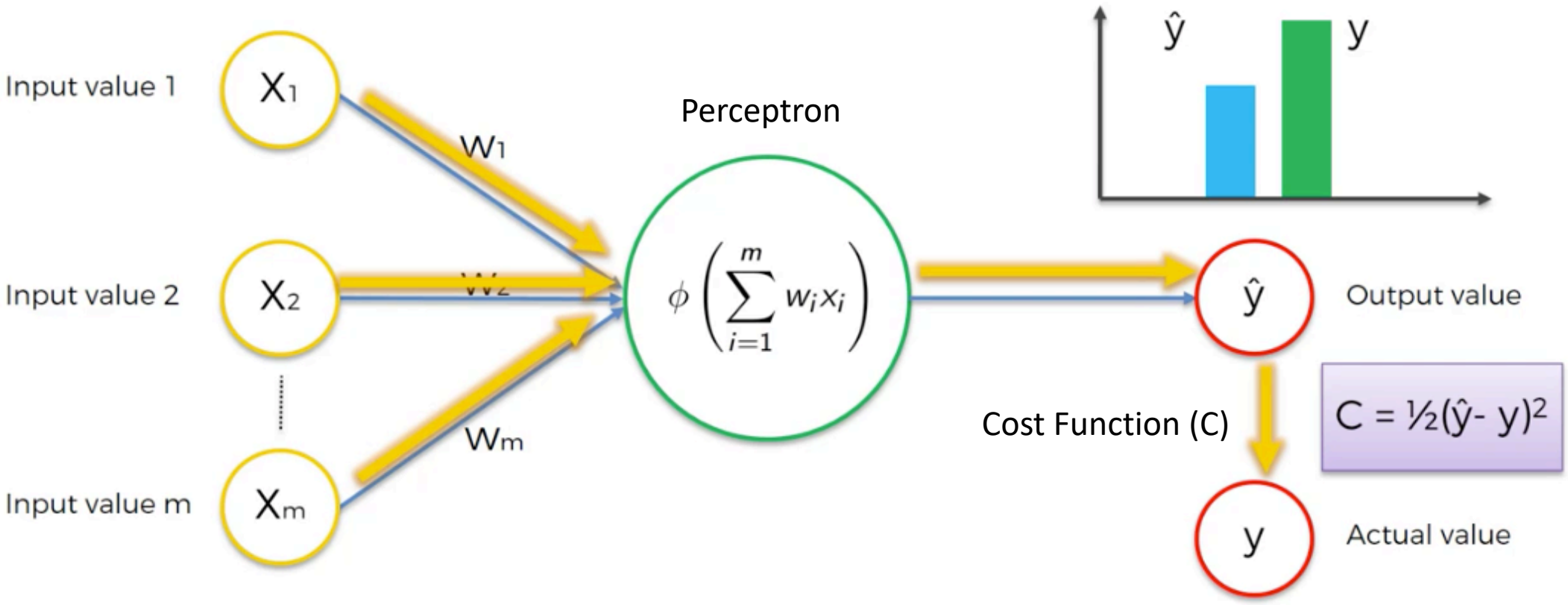
ANNs Plan of Attack

- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

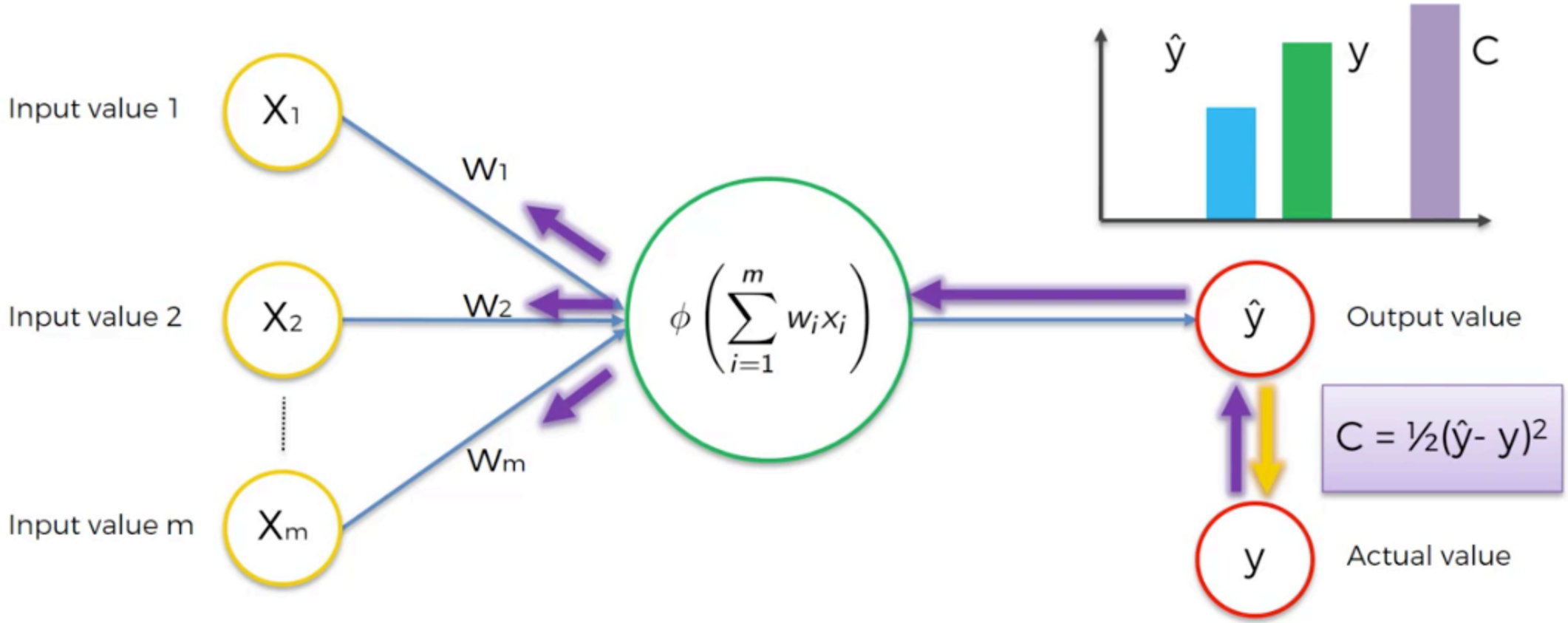
How do Neural Networks learn?



How do Neural Networks learn?



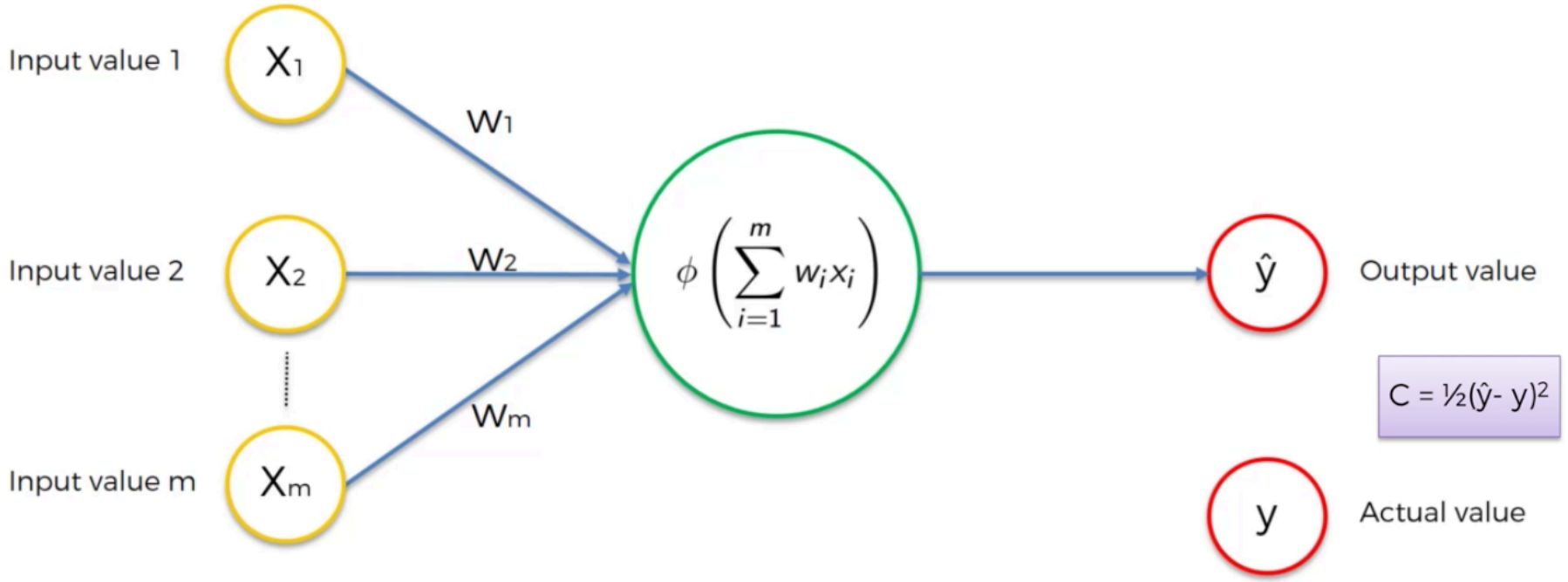
How do Neural Networks learn?



Ex: Dealing with only 1 row into our NN

How do Neural Networks learn?

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%

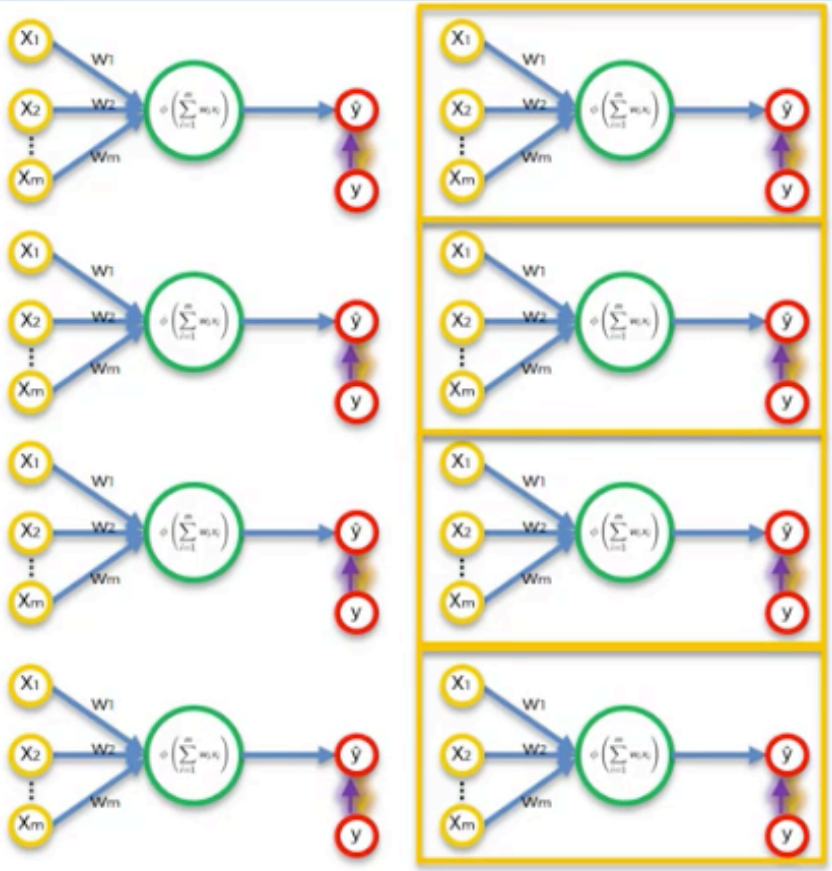


Notes:

- Neural networks require the input to be scaled in a consistent way.
- You can rescale it to the range between 0 and 1 called normalization.
- Another popular technique is to standardize it so that the distribution of each column has the mean of zero and the standard deviation of 1.

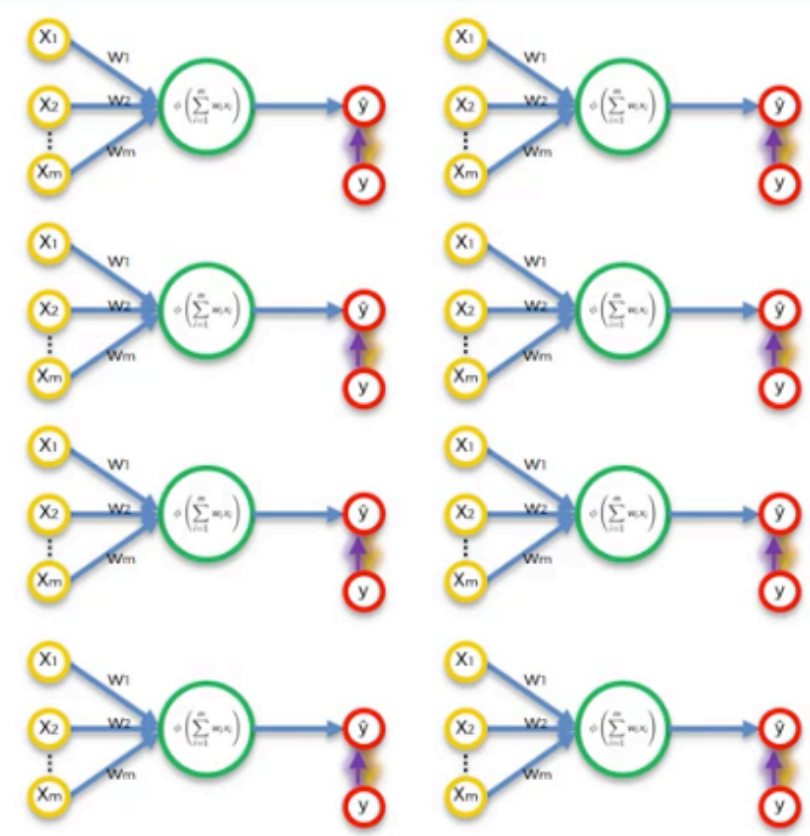
1 Epoch = We go through the whole dataset

How do Neural Networks learn?



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

How do Neural Networks learn?



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

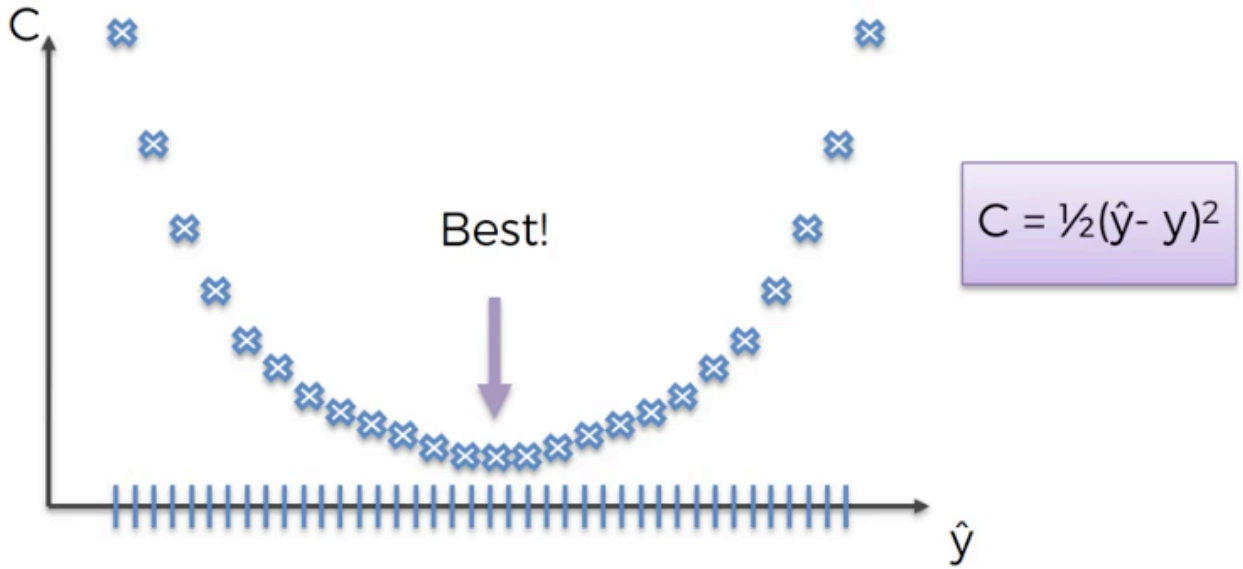
Goal: Minimize the cost function

ANNs Plan of Attack

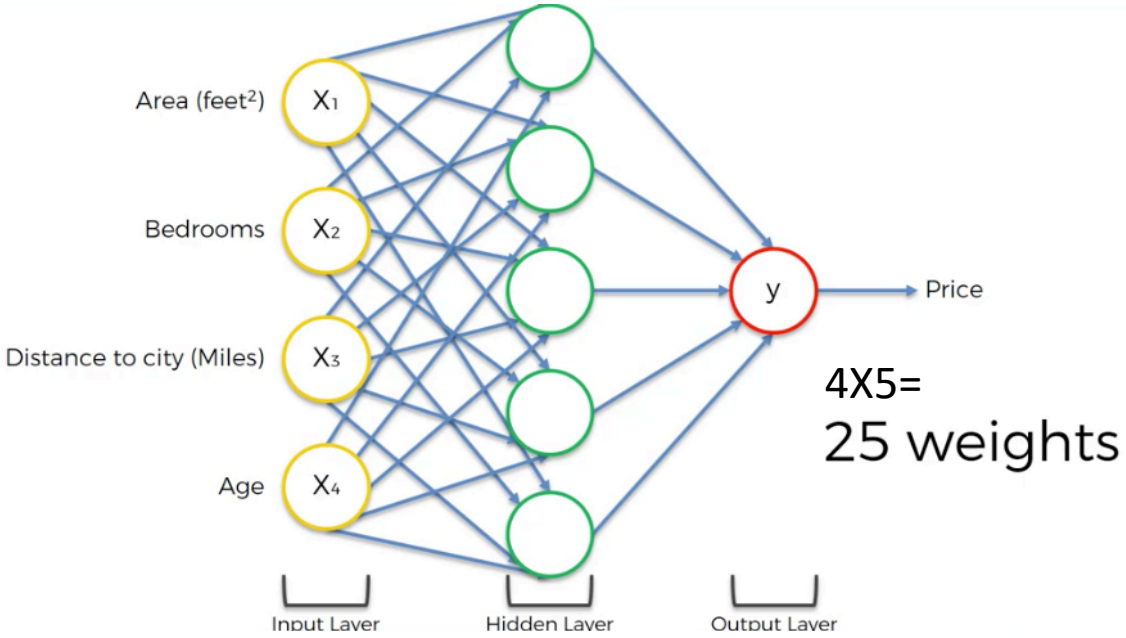
- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

Gradient Descent

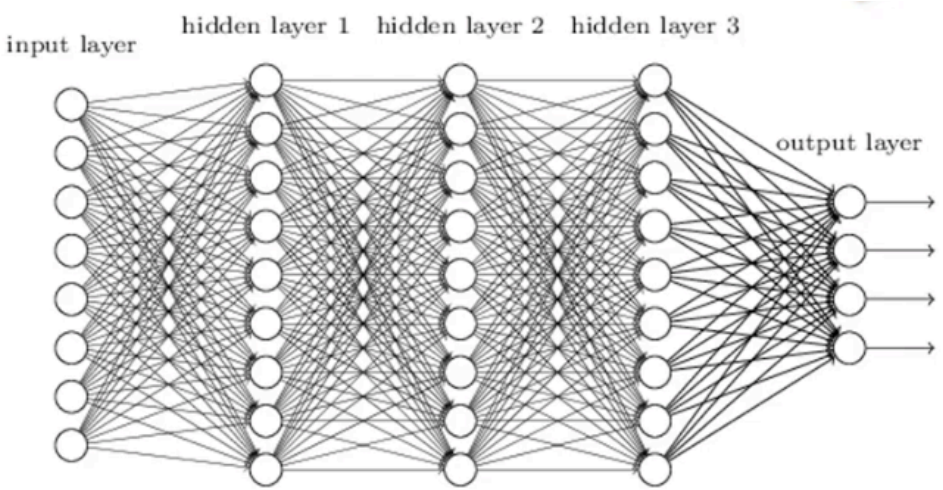
Brute Force
Problem: Curse of Dimensionality (Many weights)



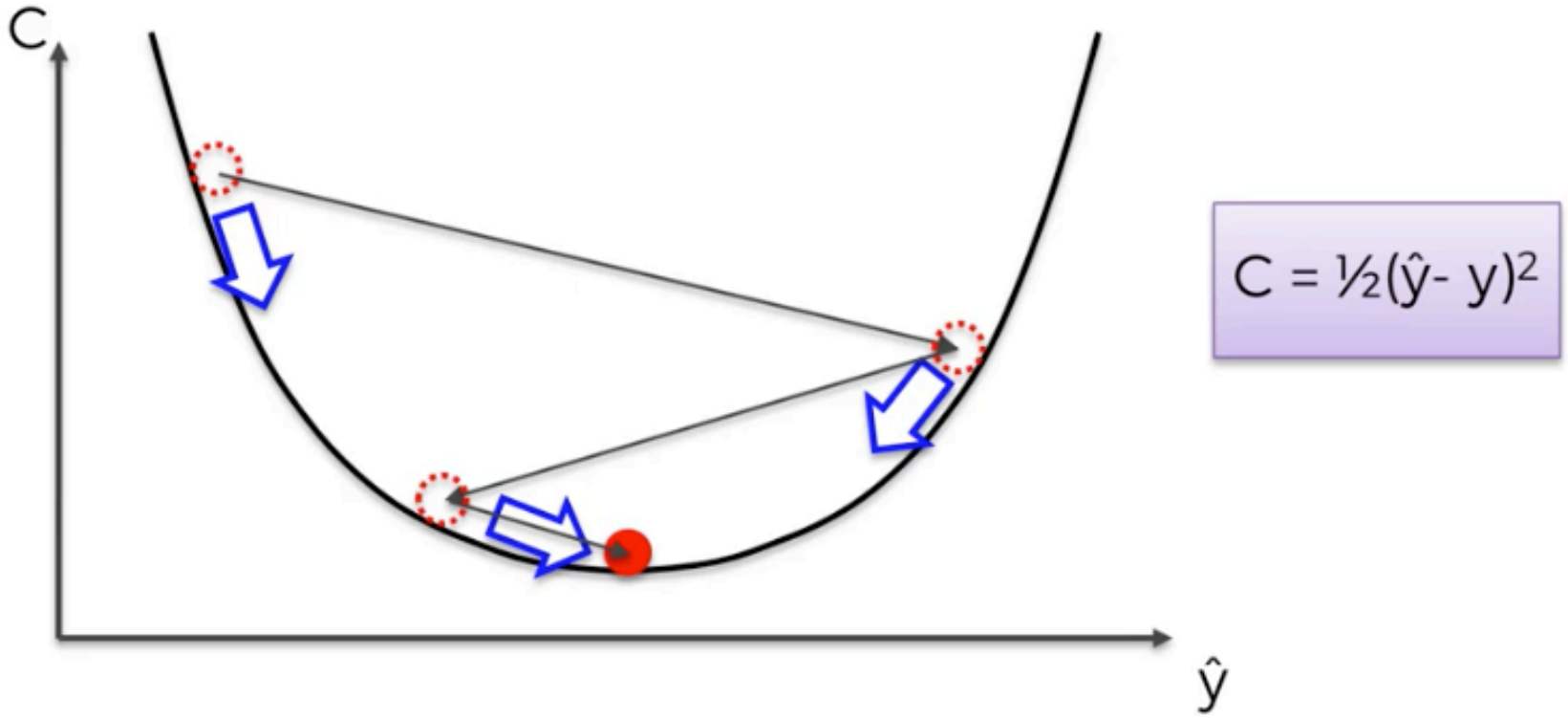
$$C = \frac{1}{2}(\hat{y} - y)^2$$



4X5=
25 weights



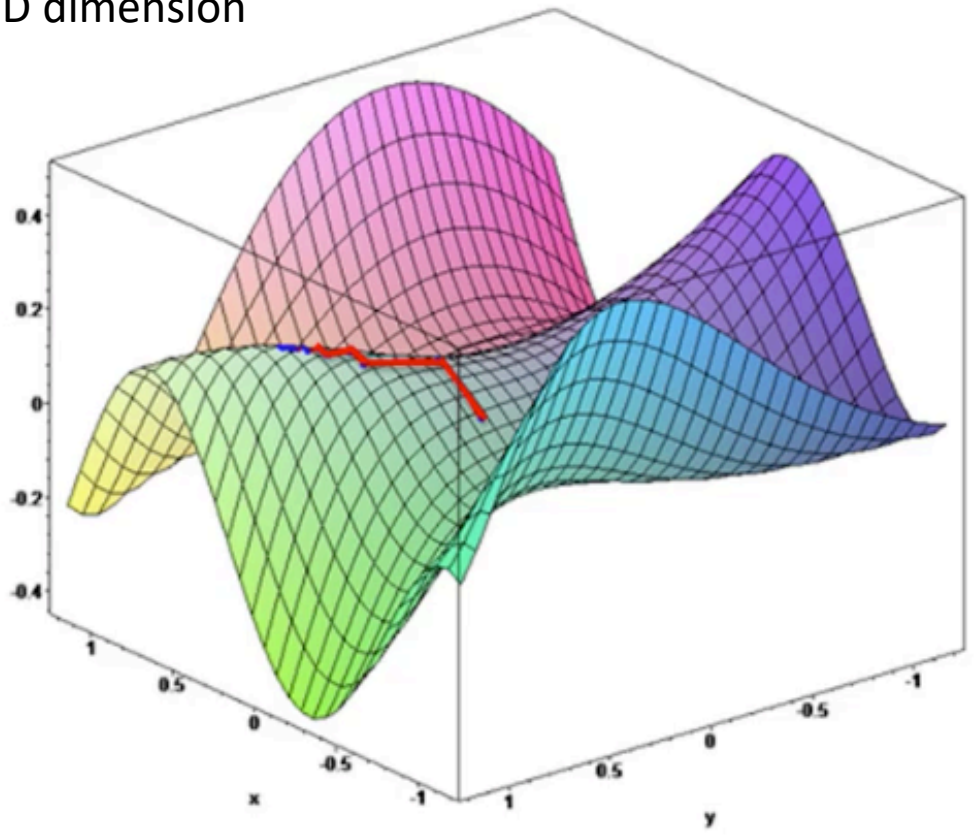
Gradient Descent

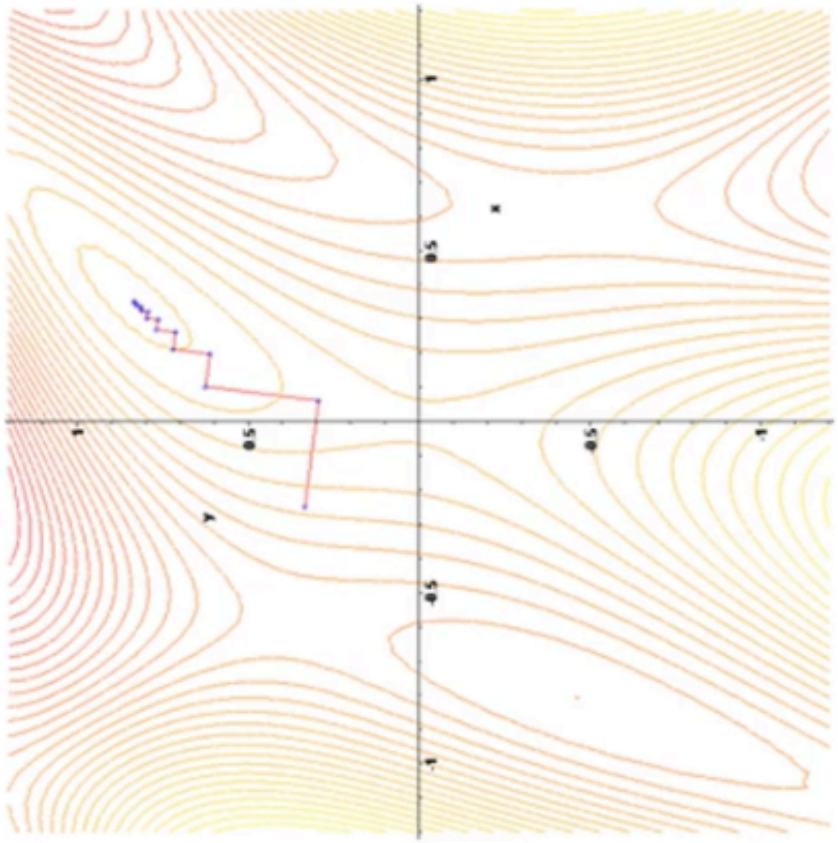


$$C = \frac{1}{2}(\hat{y} - y)^2$$

Gradient Descent

3D dimension



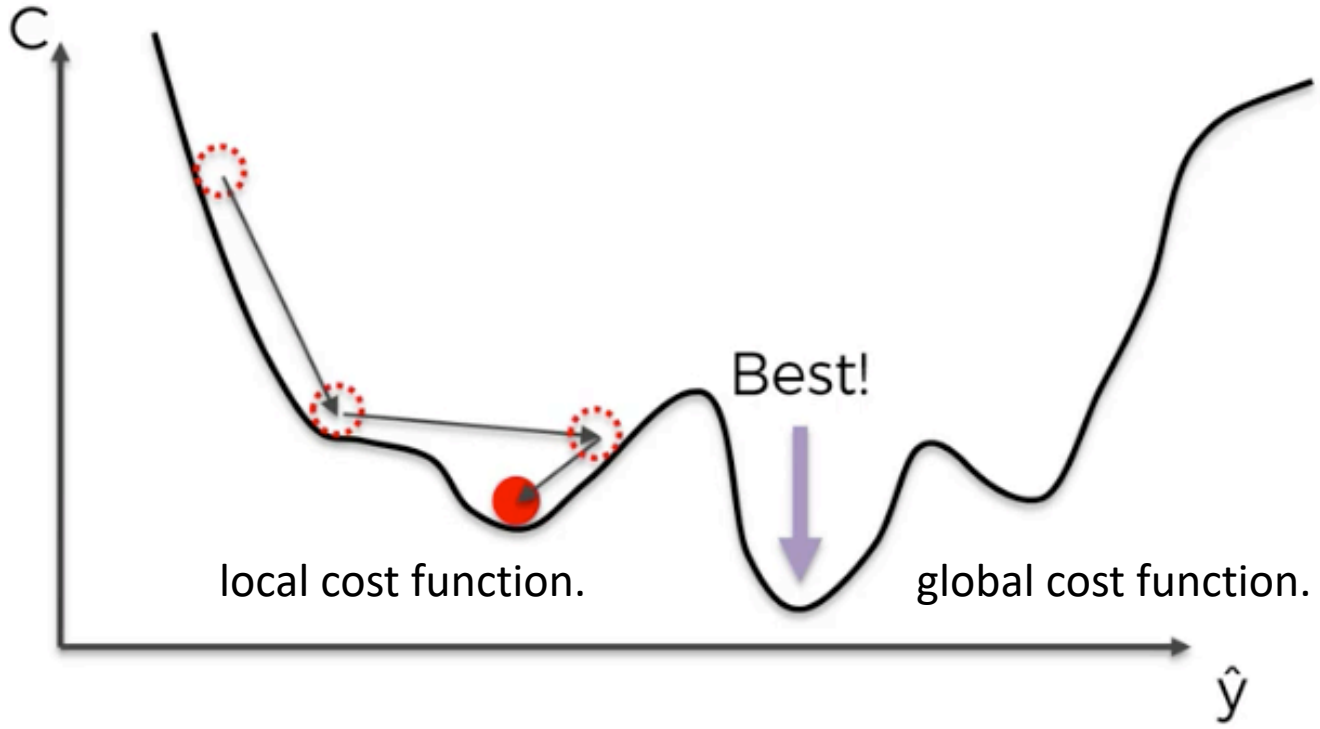


ANNs Plan of Attack

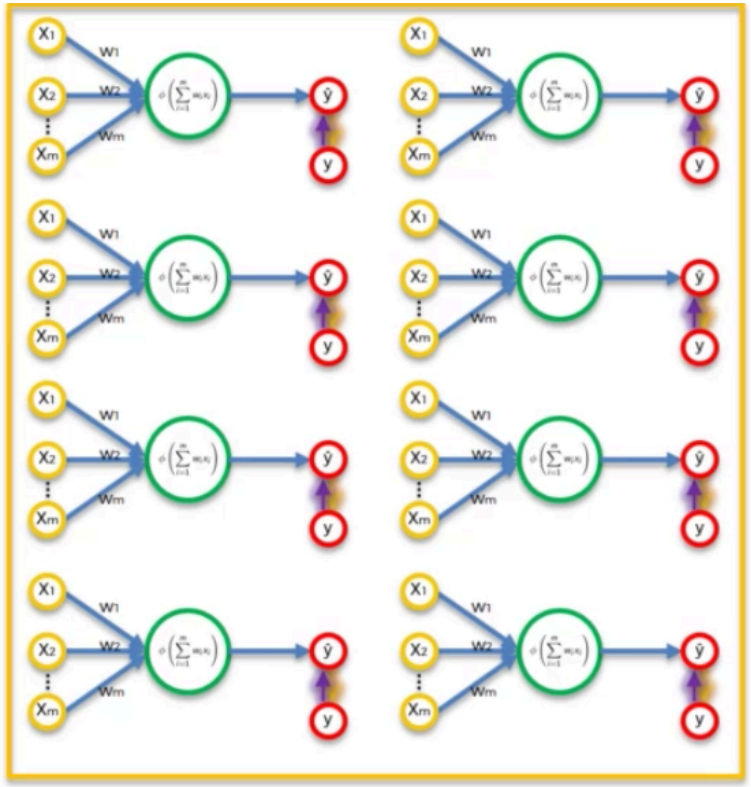
- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

Stochastic Gradient Descent

Gradient Descent require convex

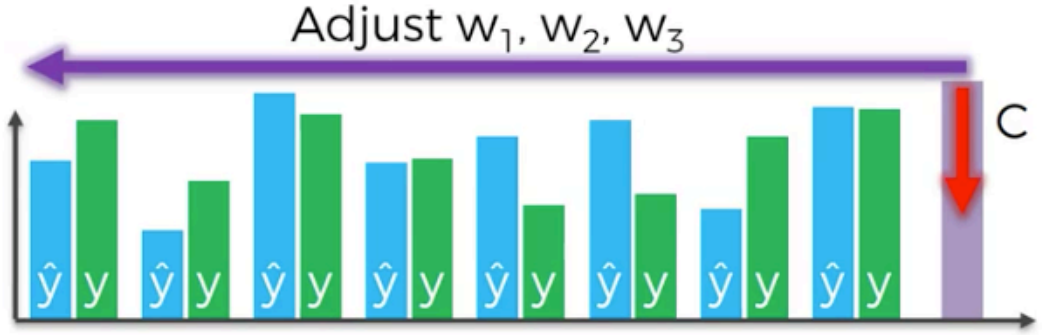


Stochastic Gradient Descent



Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$

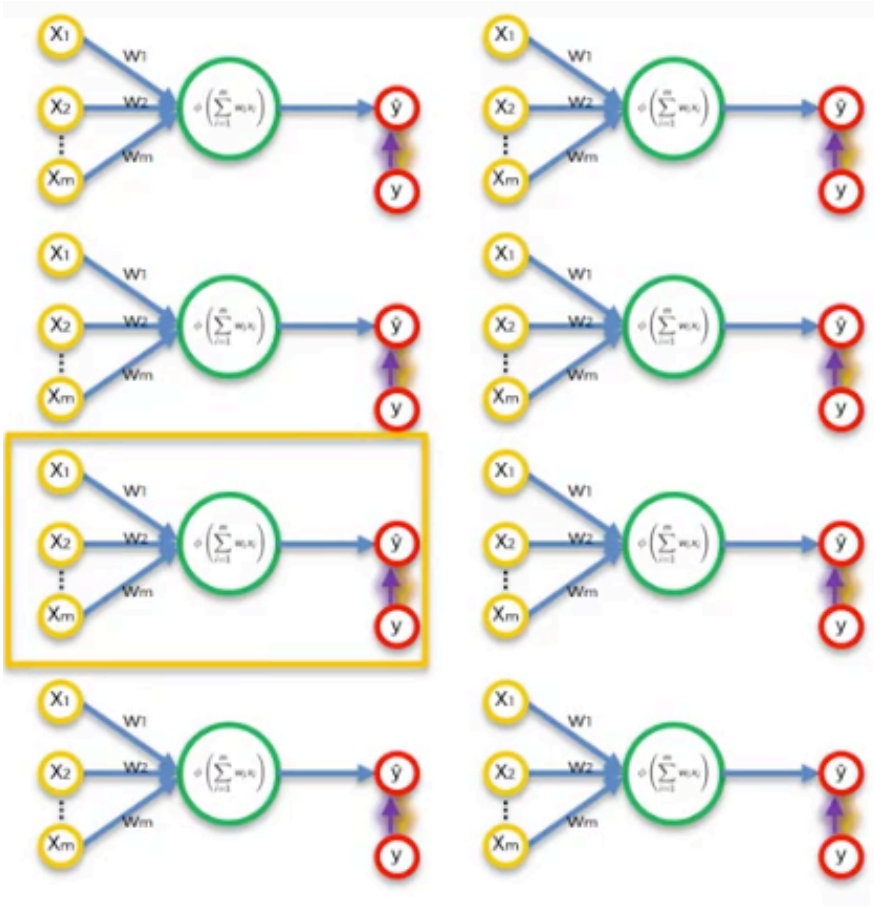


Gradient Descent or Batch gradient Descent

We take the whole batch

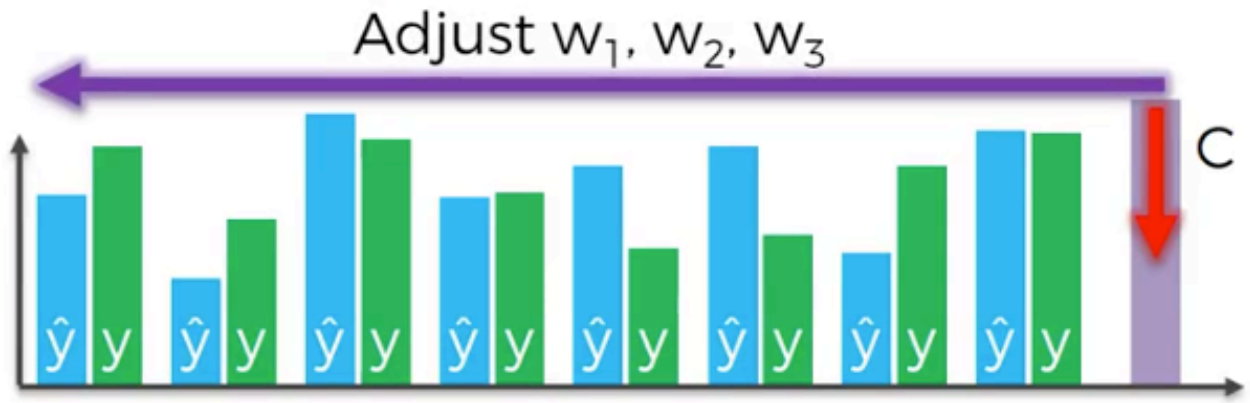
Stochastic Gradient Descent

Adjust the weight every single row.
Help you avoid local global minimum



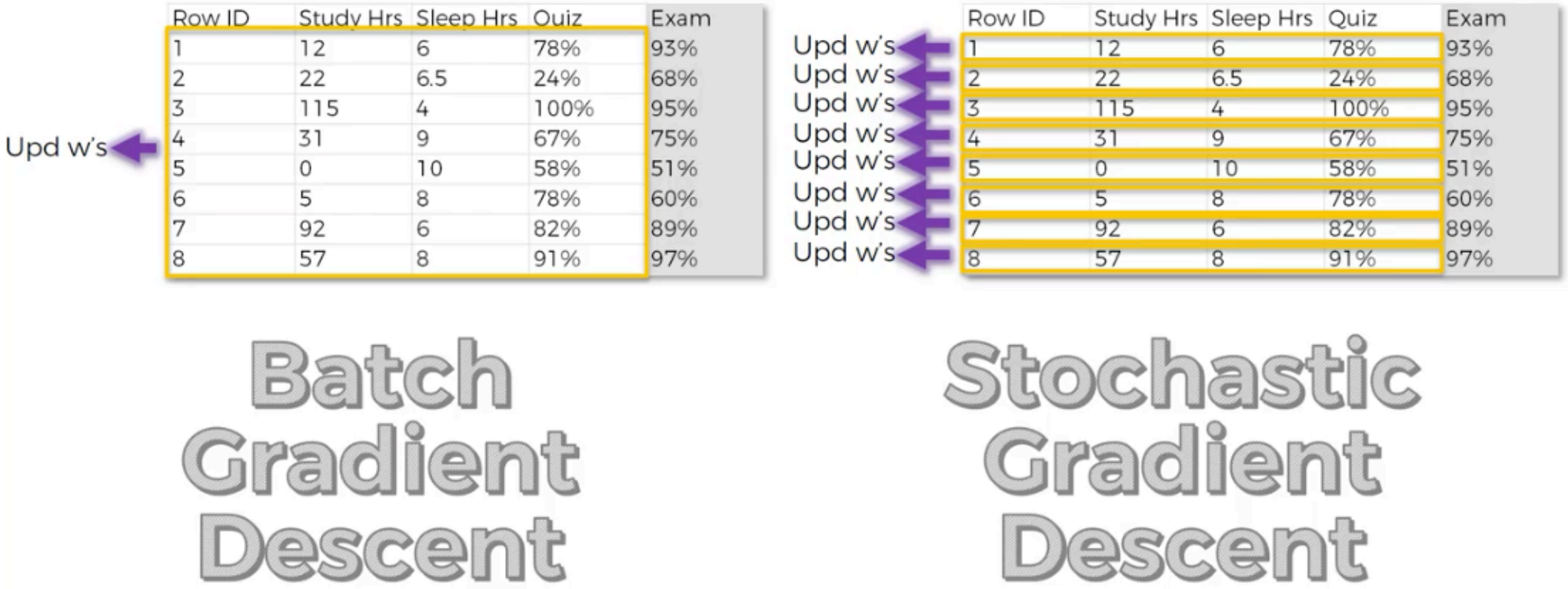
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

$$C = \sum \tfrac{1}{2}(\hat{y} - y)^2$$



Stochastic Gradient Descent

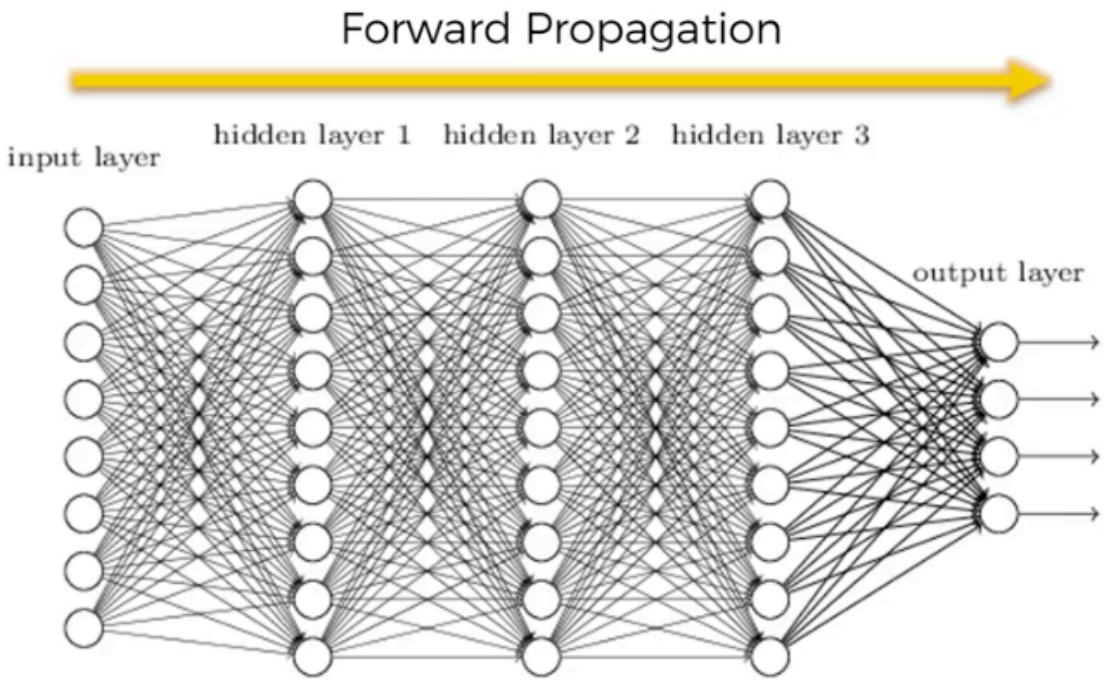
- **Batch GD** is optimistic method but may have local minimum problem.
- **SGD** = we can random pick the row -> High fluctuation -> More likely to find global minimum -> Faster cz they don't need to load the whole data into memory
- **Mini-batch GD** = combine both methods

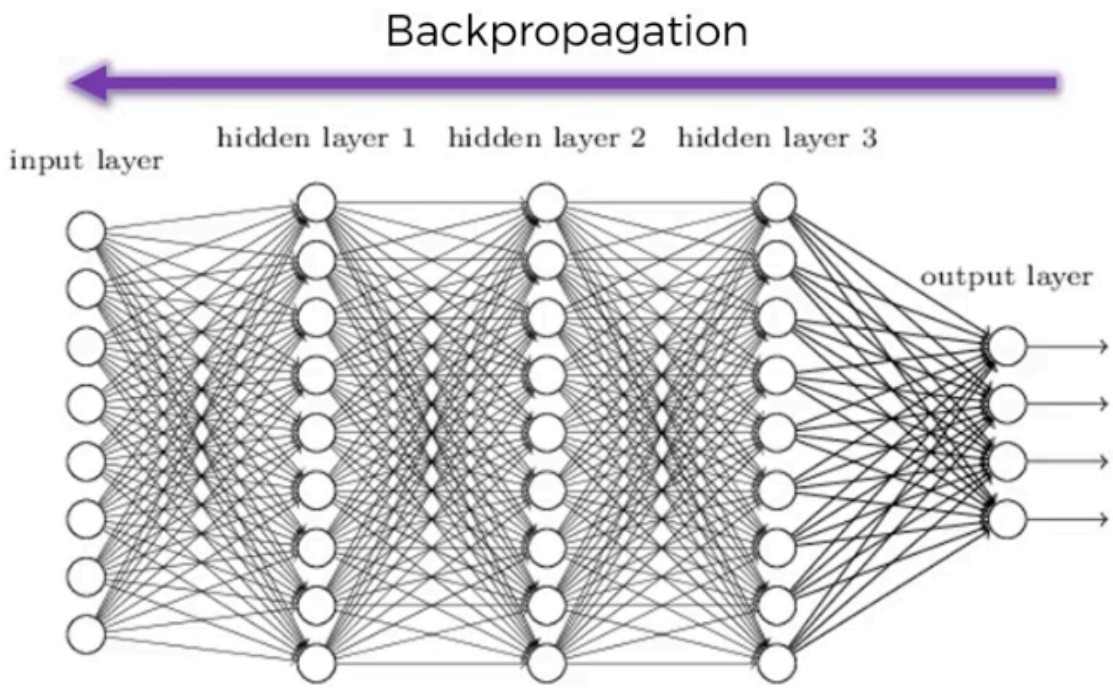


ANNs Plan of Attack

- The Neuron
- The Activation Function
- How do Neural Networks work? (example)
- How do Neural Networks learn?
- Gradient Descent
- Stochastic Gradient Descent
- Back propagation

Back propagation





Wrap-up all together!!

Training the ANN with Stochastic Gradient Descent

STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or: Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.

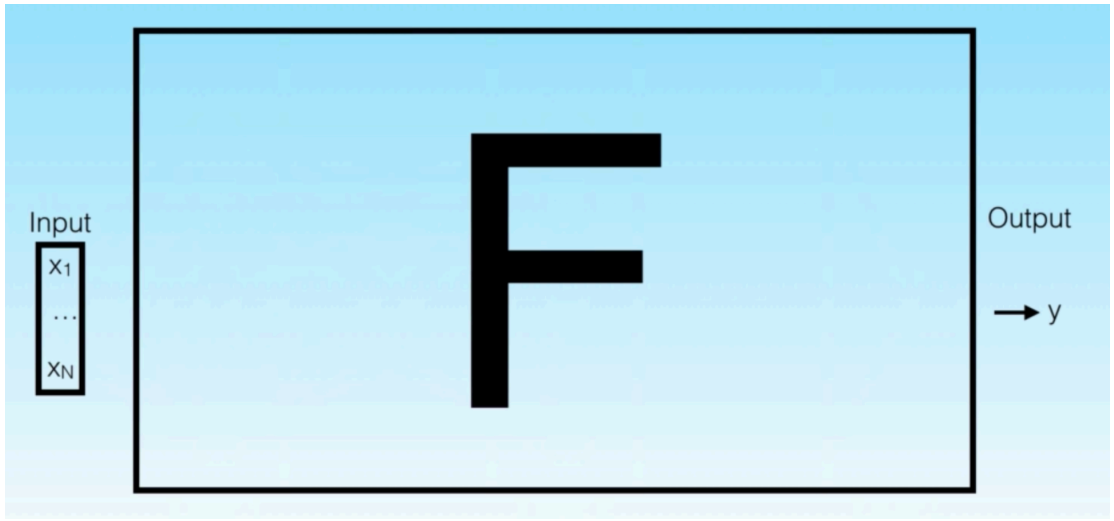
More Details…

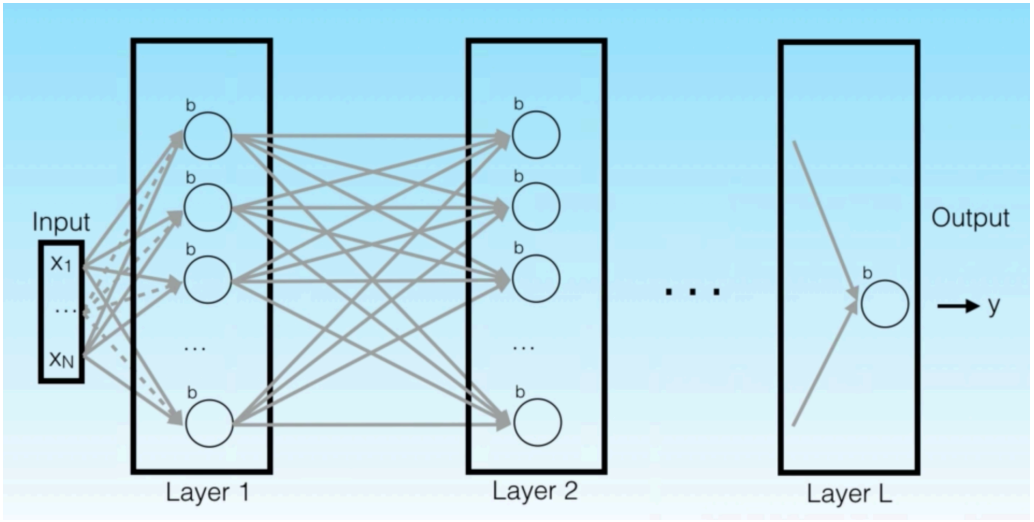
Output Layer

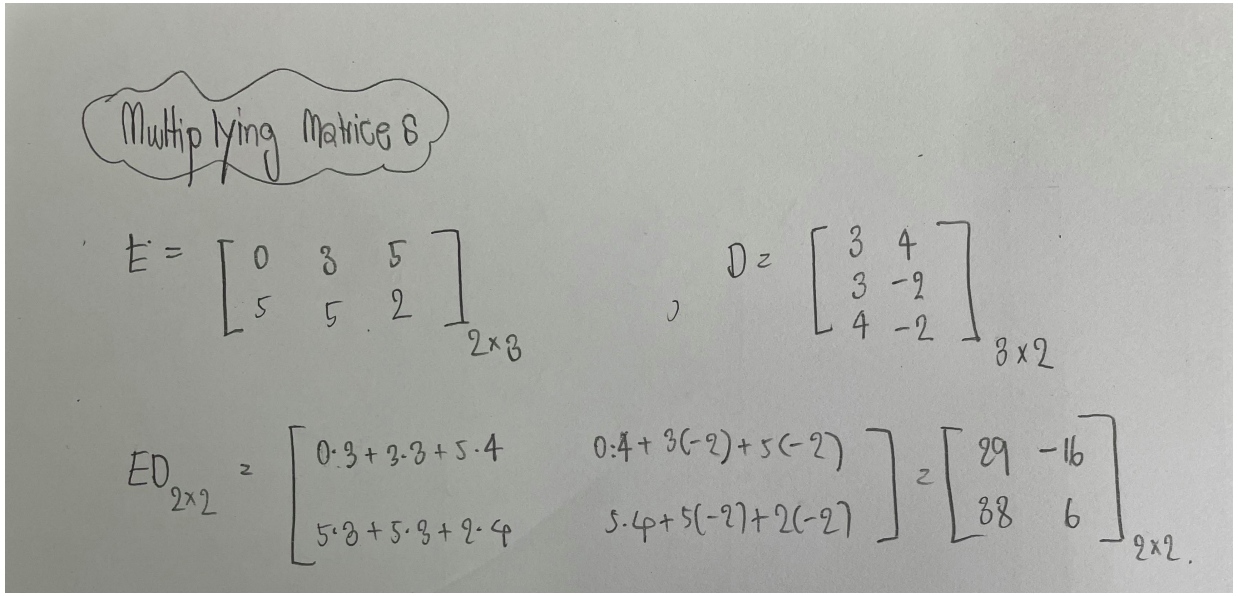
The final hidden layer is called the output layer and it is responsible for outputting a value or vector of values that correspond to the format required for the problem. The choice of activation function in the output layer is strongly constrained by the type of problem that you are modeling. For example:

- A regression problem may have a single output neuron and the neuron may have no activation function.
- A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the primary class. This can be turned into a crisp class value by using a threshold of 0.5 and snap values less than the threshold to 0 otherwise to 1.
- A multiclass classification problem may have multiple neurons in the output layer, one for each class (e.g. three neurons for the three classes in the famous iris flowers classification problem). In this case a softmax activation function may be used to output a probability of the network predicting each of the class values. Selecting the output with the highest probability can be used to produce a crisp class classification value.

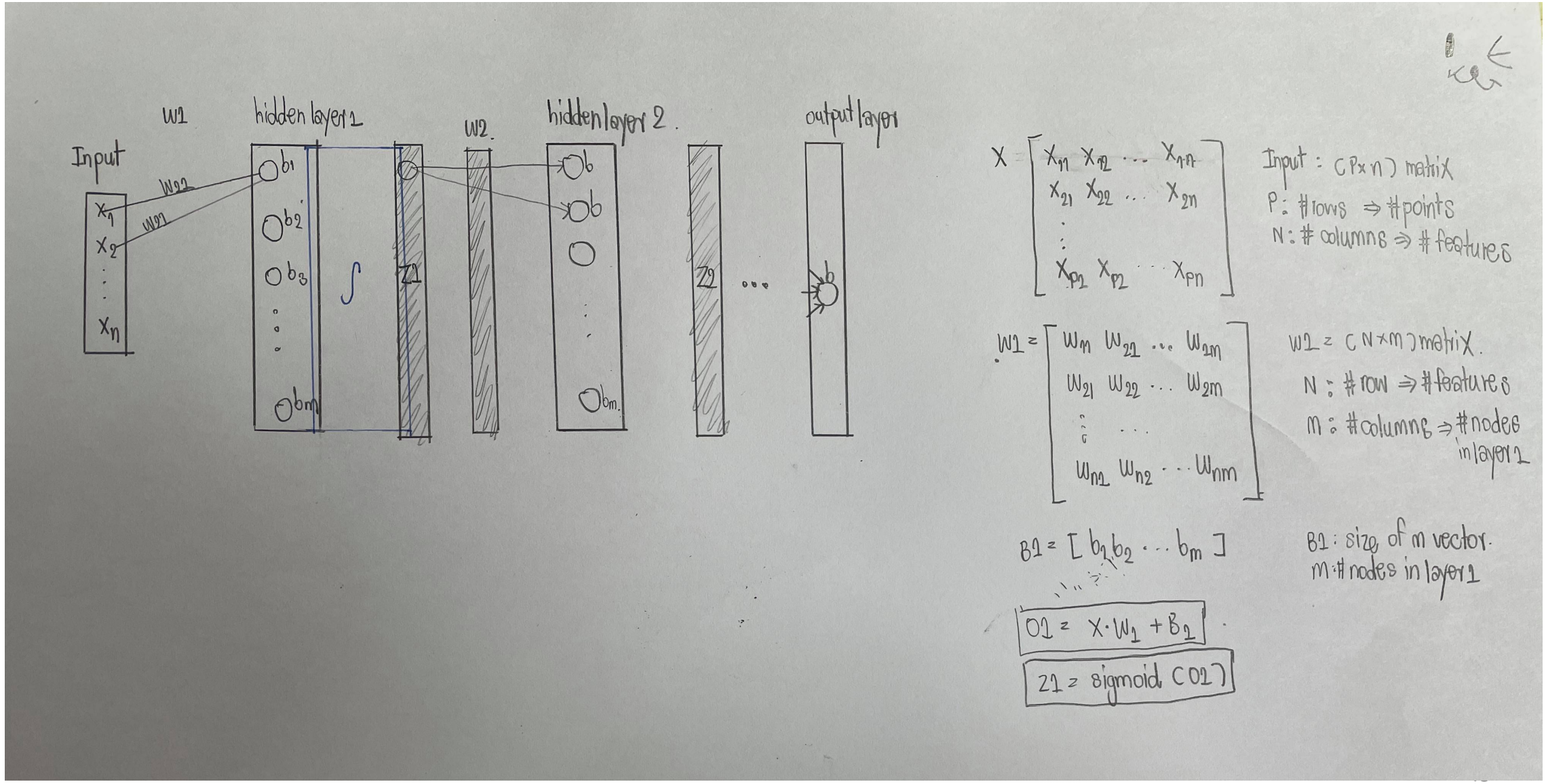
A Forward Pass







A Forward Pass



$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{bmatrix}$$

Input : (P × n) matrix
P : # rows ⇒ # points
N : # columns ⇒ # features

$$W1 = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{2m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & & \cdots & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

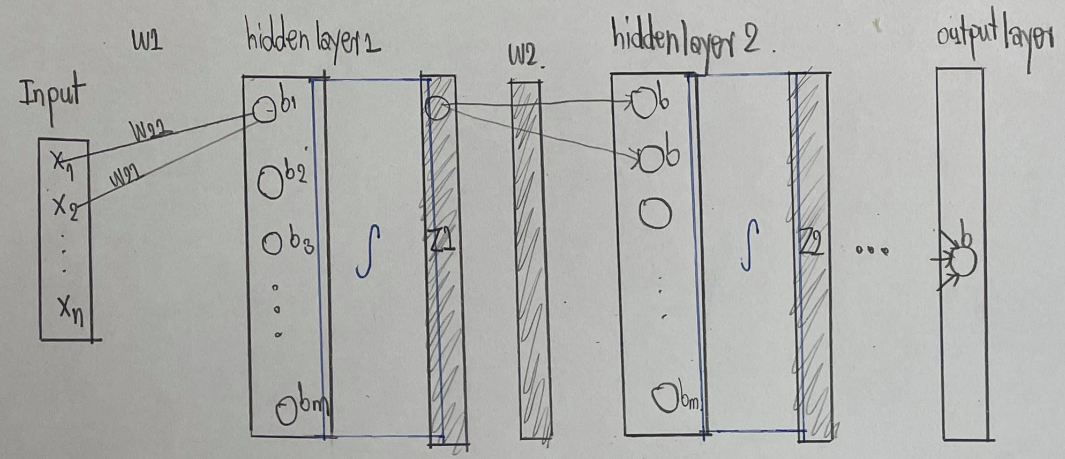
W1 = (N × m) matrix.
N : # row ⇒ # features
m : # columns ⇒ # nodes in layer 1

$$B1 = [\, b_1 \; b_2 \; \cdots \; b_m \,]$$

B1 : size of m vector.
m : # nodes in layer 1

$$O1 = X \cdot W_1 + B_1$$

$$Z1 = \text{sigmoid}(O1)$$



$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & & & \\ x_{p1} & x_{p2} & \cdots & x_{pn} \end{bmatrix}$$

Input : (P × n) matrix
P : # rows ⇒ # points
N : # columns ⇒ # features

$$W1 = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & & \cdots & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

W1 = (N × m) matrix.
N : # row ⇒ # features
m : # columns ⇒ # nodes in layer 1

$$B1 = [\, b_1\ b_2 \cdots b_m \,]$$

B1 : size of m vector.
m : # nodes in layer 1

$$O1 = X \cdot W_1 + B_1$$

$$Z1 = \text{sigmoid}(O1)$$

(cont.)

$$Z1 = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & & & \\ z_{p1} & z_{p2} & \cdots & z_{pm} \end{bmatrix}$$

Input (P × m) matrix
P : # rows ⇒ points
m : # cols ⇒ features.

$$W2 = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1K} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \vdots & & & \\ w_{m1} & w_{m2} & \cdots & w_{mk} \end{bmatrix}$$

W2 : (m × k) matrix
M : # rows ⇒ features
K : # columns ⇒ # node in layer 2.

$$B2 = [\, b_1\ b_2 \cdots b_k \,]$$

B2 : size k vector
k : # nodes in layer 2.

$$O2 = Z1 \cdot W_2 + B2 \rightarrow Z2 = \text{sigmoid}(O_2)$$

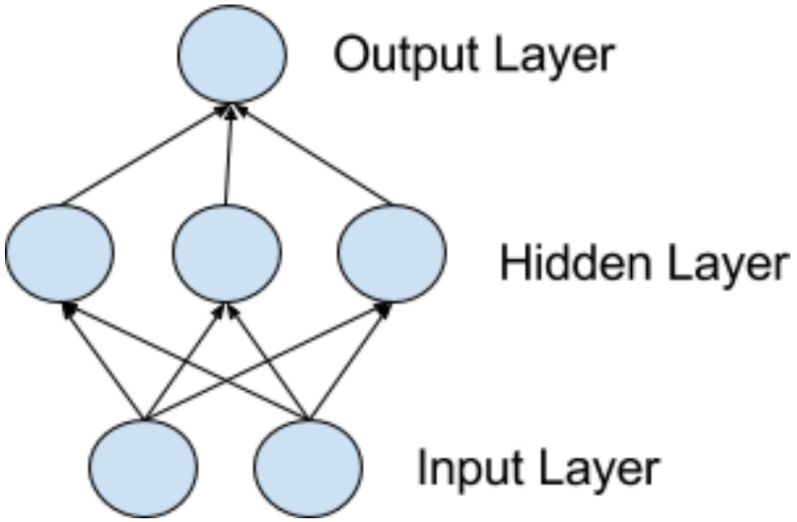
Exercise1

1. This problem should be Classification or Regression? Why?
2. Write the network structure of this keras neural network code

```
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Exercise2

1. Write Keras Code For this neural network
2. How many trainable parameters?




```
model = Sequential()
model.add(Dense(3, input_shape=(2,), activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(Adam(lr=0.1),
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 3)                 9
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 4
=================================================================
Total params: 13
Trainable params: 13
Non-trainable params: 0
_________________________________________________________________
```

Exercise3

1. Why there are 15 trainable parameters?

```
model = Sequential()
model.add(Dense(3, input_shape=(4,), activation='softmax'))
model.compile(Adam(lr=0.1),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 3)                 15
=================================================================
Total params: 15
Trainable params: 15
Non-trainable params: 0
_________________________________________________________________
```

Build Deep Learning Models with Keras

The focus of Keras is the idea of a model. The main type of model is a sequence of layers called a `Sequential` which is a linear stack of layers. You create a `Sequential` and add layers to it in the order that you wish for the computation to be performed. Once defined, you compile the model which makes use of the underlying framework to optimize the computation to be performed by your model. In this you can specify the loss function and the optimizer to be used.

Once compiled, the model must be fit to data. This can be done one batch of data at a time or by firing off the entire model training regime. This is where all the compute happens. Once trained, you can use your model to make predictions on new data. We can summarize the construction of deep learning models in Keras as follows:

1. **Define your model**. Create a `Sequential` model and add configured layers.
2. **Compile your model**. Specify loss function and optimizers and call the `compile()` function on the model.
3. **Fit your model**. Train the model on a sample of data by calling the `fit()` function on the model.
4. **Make predictions**. Use the model to generate predictions on new data by calling functions such as `evaluate()` or `predict()` on the model.

Demo

1. Binary Classification
2. Multiclass Classification
3. Regression

1. Binary Classification: Titanic Survival Prediction

Data Descriptions: Understanding the data is must before it's manipulation and analysis.

1. PassengerID
2. Survival: 0 = No, 1 = Yes
3. pclass (Ticket class): 1 = 1st, 2 = 2nd, 3 = 3rd
4. Name
5. sex: Sex
6. Age: Age in years
7. sibsp: number of siblings/spouses aboard the Titanic
8. parch: number of parents/children aboard the Titanic
9. ticket: Ticket number
10. fare: Passenger fare
11. cabin: Cabin number
12. embarked: Port of Embarkation, C = Cherbourg, Q = Queenstown, S = Southampton

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

https://www.kaggle.com/c/titanic/overview

How to use Learning Curves to Diagnose DL Model Performance

- A learning curve is a plot of model learning performance over experience or time. It is a widely used diagnostic tool in machine learning for algorithms that learn from a training dataset incrementally.
- The model can be evaluated on the training dataset and on a hold out validation dataset after each update during training and plots of the measured performance can created to show learning curves.
- After learning this, you will know:
  - Learning curves are plots that show changes in learning performance over time in terms of experience.
  - Learning curves of model performance on the train and validation datasets can be used to diagnose an underfit, overfit, or well-fit model.
  - Learning curves of model performance can be used to diagnose whether the train or validation datasets are not relatively representative of the problem domain.

How to use Learning Curves to Diagnose DL Model Performance

We will include 3 parts:

1. Learning Curves
2. Diagnosing Model Behavior
3. Diagnosing Unrepresentative Datasets

How to use Learning Curves to Diagnose DL Model Performance

1. Learning Curves

- During the training of a machine learning model, the current state of the model at each step of the training algorithm can be evaluated.

It can be evaluated on the training dataset to give an idea of how well the model is "*learning*." It can also be evaluated on a hold-out validation dataset that is not part of the training dataset. Evaluation on the validation dataset gives an idea of how well the model is "*generalizing*."

  - **Train Learning Curve**: Learning curve calculated from the training dataset that gives an idea of how well the model is learning.
  - **Validation Learning Curve**: Learning curve calculated from a hold-out validation dataset that gives an idea of how well the model is generalizing.

- It is common to create dual learning curves for a machine learning model during training on both the training and validation datasets.

How to use Learning Curves to Diagnose DL Model Performance

1. Learning Curves

- In some cases, it is also common to create learning curves for multiple metrics, such as in the case of classification predictive modeling problems, where the model may be optimized according to cross-entropy loss and model performance is evaluated using classification accuracy.
- In this case, two plots are created, one for the learning curves of each metric, and each plot can show two learning curves, one for each of the train and validation datasets.
  - **Optimization Learning Curves**: Learning curves calculated on the metric by which the parameters of the model are being optimized, e.g. loss.
  - **Performance Learning Curves**: Learning curves calculated on the metric by which the model will be evaluated and selected, e.g. accuracy.

How to use Learning Curves to Diagnose DL Model Performance

We will include 3 parts:

1. Learning Curves
2. Diagnosing Model Behavior
3. Diagnosing Unrepresentative Datasets

How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

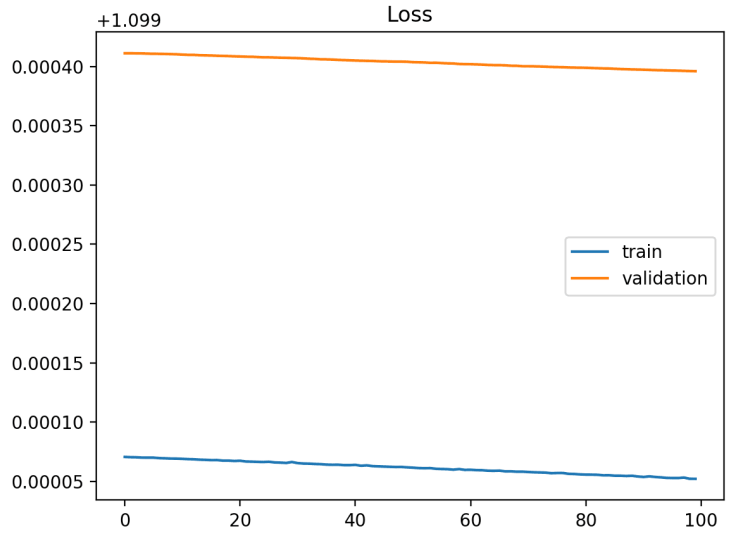
- The shape and dynamics of a learning curve can be used to diagnose the behavior of a machine learning model and in turn perhaps suggest at the type of configuration changes that may be made to improve learning and/or performance.
- There are three common dynamics that you are likely to observe in learning curves
    1. Underfit.
    2. Overfit.
    3. Good Fit.
- We will take a closer look at each with examples. The examples will assume that we are looking at a minimizing metric, meaning that smaller relative scores on the y-axis indicate more or better learning.

How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Underfit Learning Curves

- Underfitting refers to a model that cannot learn the training dataset.
  - An underfit model can be identified from the learning curve of the training loss only.
  - It may show a flat line or noisy values of relatively high loss, indicating that the model was unable to learn the training dataset at all.
  - An example of this is provided below and is common when the model does not have a suitable capacity for the complexity of the dataset.

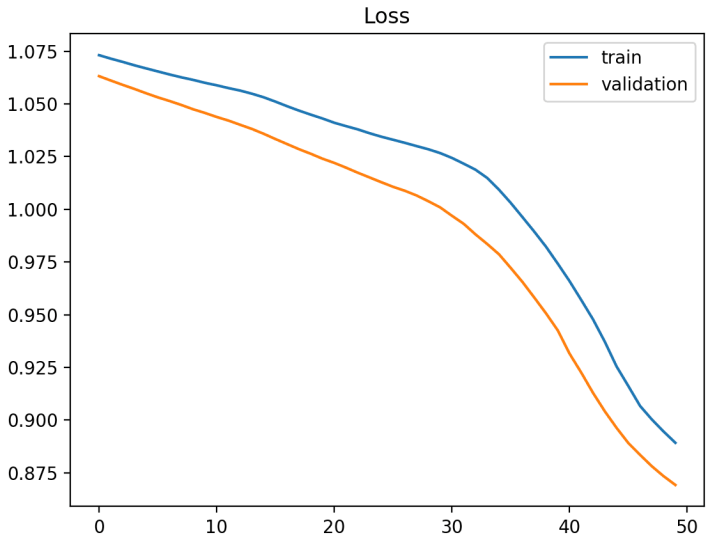


How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Underfit Learning Curves

- Underfitting refers to a model that cannot learn the training dataset.
  - An underfit model may also be identified by a training loss that is decreasing and continues to decrease at the end of the plot.
  - This indicates that the model is capable of further learning and possible further improvements and that the training process was halted prematurely.

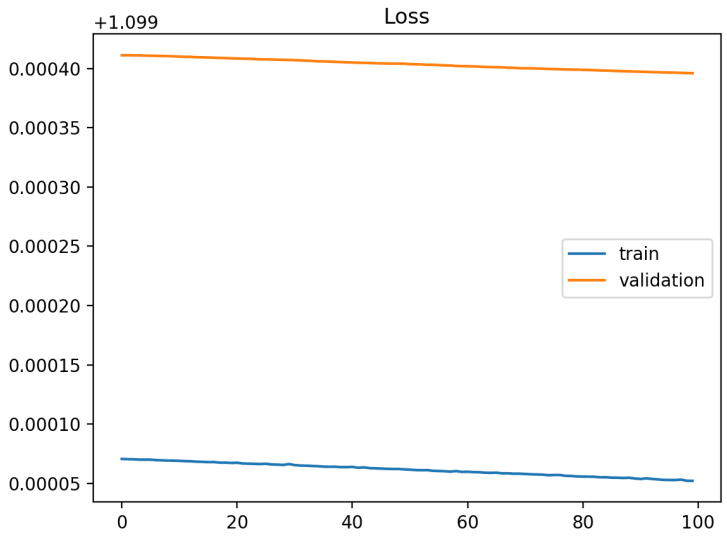


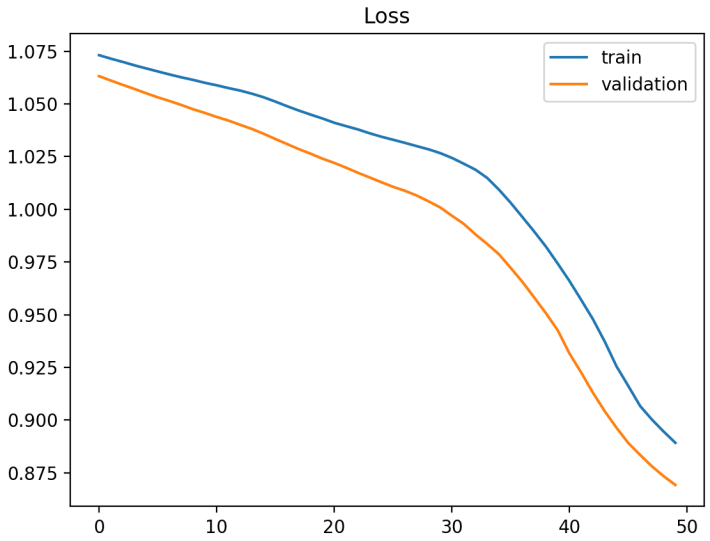
How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Underfit Learning Curves

- A plot of learning curves shows underfitting if:
  1. The training loss remains flat regardless of training.
  2. The training loss continues to decrease until the end of training.





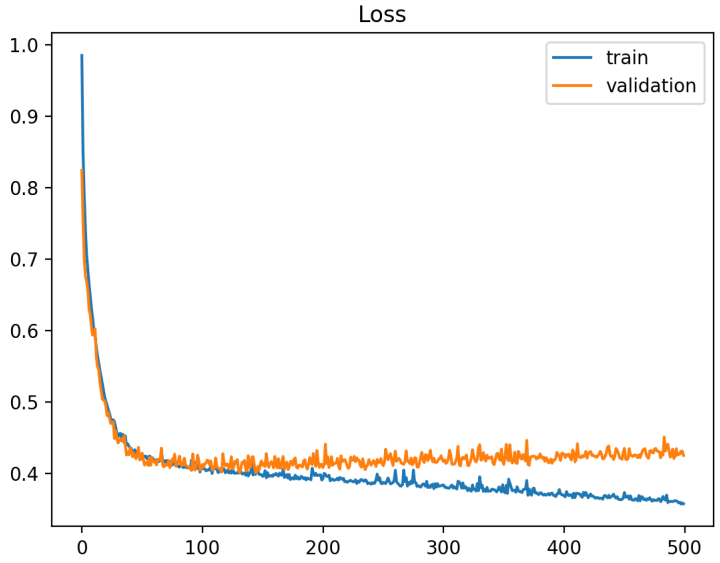
How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Overfit Learning Curves

Overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset.

This often occurs if the model has more capacity than is required for the problem, and, in turn, too much flexibility. It can also occur if the model is trained for too long.



A plot of learning curves shows overfitting if:

- The plot of training loss continues to decrease with experience.
- The plot of validation loss decreases to a point and begins increasing again.

The inflection point in validation loss may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Good Fit Learning Curves

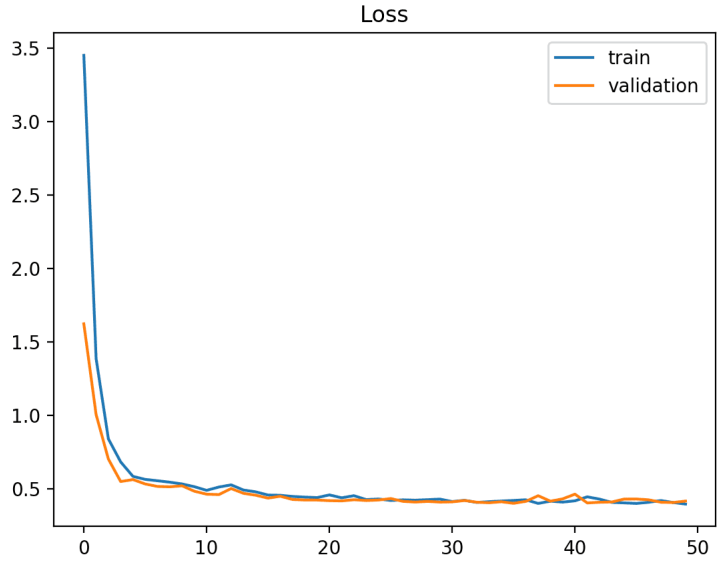
- A good fit is the goal of the learning algorithm and exists between an overfit and underfit model.
- A good fit is identified by a training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values.
- The loss of the model will almost always be lower on the training dataset than the validation dataset. This means that we should expect some gap between the train and validation loss learning curves. This gap is referred to as the "generalization gap."

How to use Learning Curves to Diagnose DL Model Performance

2. Diagnosing Model Behavior

Good Fit Learning Curves

- A plot of learning curves shows a good fit if:
  - The plot of training loss decreases to a point of stability.
  - The plot of validation loss decreases to a point of stability and has a small gap with the training loss.
- **Continued training of a good fit will likely lead to an overfit.**



How to use Learning Curves to Diagnose DL Model Performance

We will include 3 parts:

1. Learning Curves
2. Diagnosing Model Behavior
3. Diagnosing Unrepresentative Datasets

How to use Learning Curves to Diagnose DL Model Performance

3. Diagnosing Unrepresentative Datasets

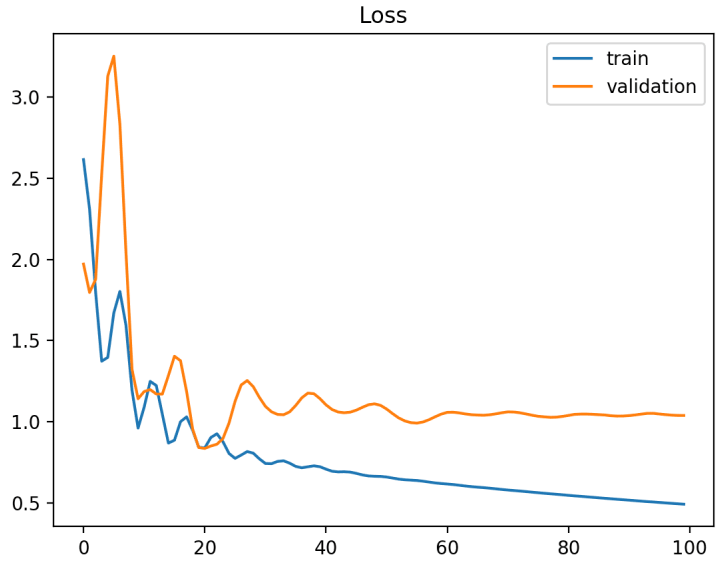
- An unrepresentative dataset means a dataset that may not capture the statistical characteristics relative to another dataset drawn from the same domain, such as between a train and a validation dataset.
- This can commonly occur if the number of samples in a dataset is too small, relative to another dataset.
- There are two common cases that could be observed:
    1. Training dataset is relatively unrepresentative.
    2. Validation dataset is relatively unrepresentative.

How to use Learning Curves to Diagnose DL Model Performance

3. Diagnosing Unrepresentative Datasets

Unrepresentative Train Dataset

- An unrepresentative training dataset means that the training dataset does not provide sufficient information to learn the problem, relative to the validation dataset used to evaluate it.
- This may occur if the training dataset has too few examples as compared to the validation dataset.
- This situation can be identified by a learning curve for training loss that shows improvement and similarly a learning curve for validation loss that shows improvement, but a large gap remains between both curves.

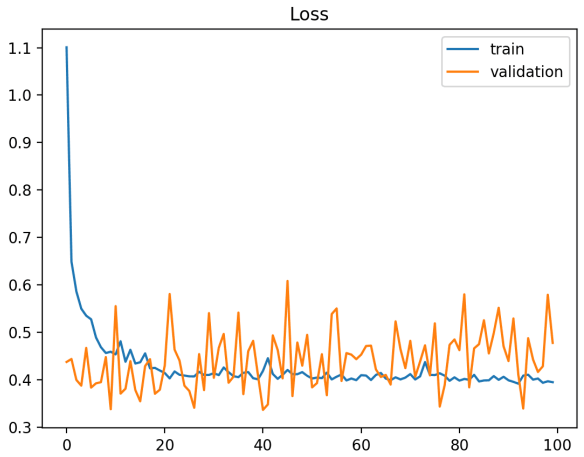


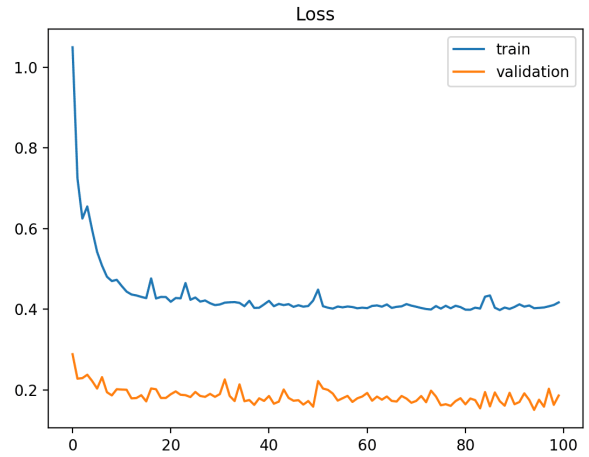
How to use Learning Curves to Diagnose DL Model Performance

3. Diagnosing Unrepresentative Datasets

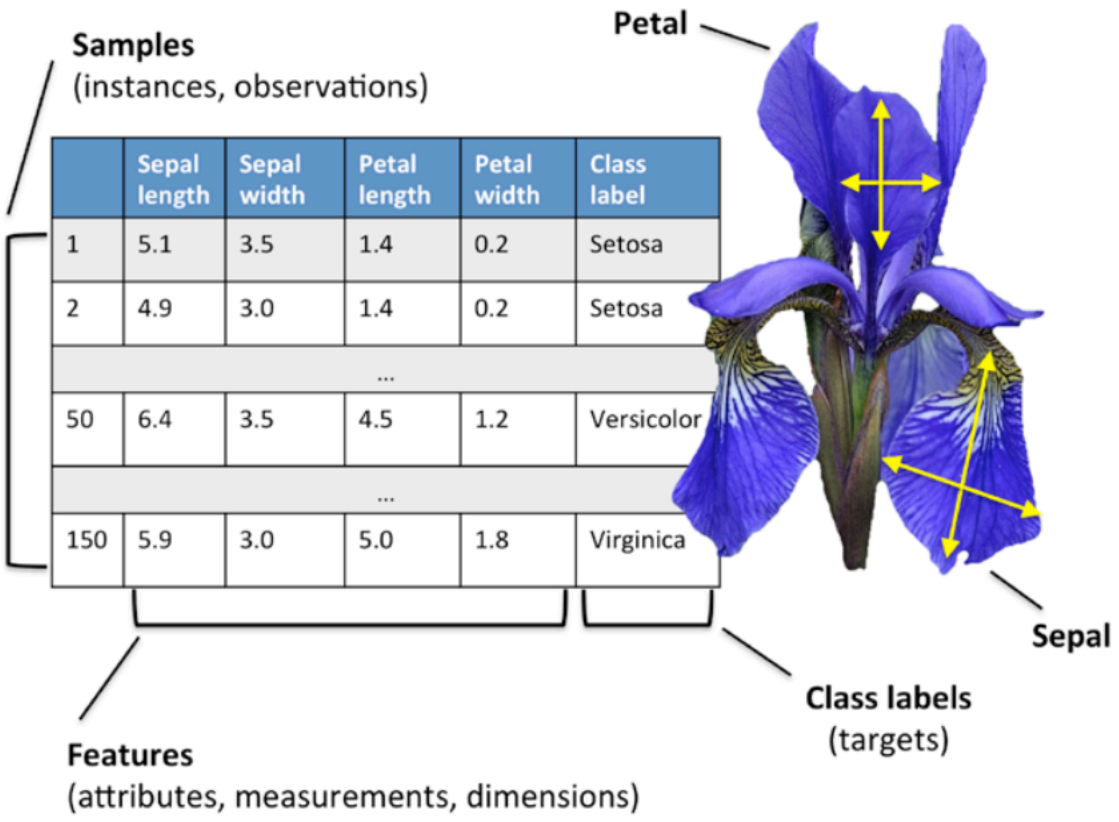
Unrepresentative Validation Dataset

- An unrepresentative validation dataset means that the validation dataset does not provide sufficient information to evaluate the ability of the model to generalize.
- This may occur if the validation dataset has too few examples as compared to the training dataset.
- This case can be identified by a learning curve for training loss that looks like a good fit (or other fits) and a learning curve for validation loss that shows noisy movements around the training loss.
- It may also be identified by a validation loss that is lower than the training loss. In this case, it indicates that the validation dataset may be easier for the model to predict than the training dataset.





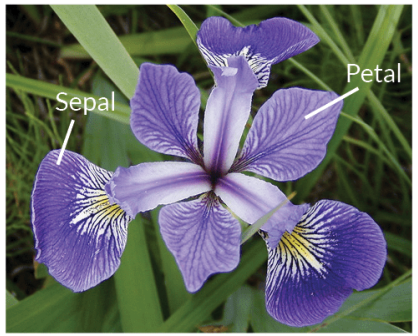
2. Multiclass Classification: Iris Species



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Row: samples, records, instances, observations
Column: attributes, features, measurements, dimensions

More than 2 output classes (3 classes) →
multi-class classification problem



Iris Versicolor

Iris Setosa

Iris Virginica

3. Regression: Predict House Price (Small Dataset)

	sqft	bdrms	age	price
0	2104	3	70	399900
1	1600	3	28	329900
2	2400	3	44	369000
3	1416	2	49	232000
4	3000	4	75	539900