AWS

MACHINE LEARNING

CERTIFICATION

DOMAIN #3: MODELING (36% EXAM MARK)

AWS ML CERTIFICATION EXAM DOMAINS

Domain	% of Examination
Domain 1: Data Engineering	20%
Domain 2: Exploratory Data Analysis	24%
Domain 3: Modeling	36%
Domain 4: Machine Learning Implementation and Operations	20%
TOTAL	100%

Source: https://d1.awsstatic.com/training-and-certification/docs-ml/AWS%20Certified%20Machine%20Learning%20-%20Specialty_Exam%20Guide%20(1).pdf

DOMAIN #3 OVERVIEW:

SECTION #8: MACHINE AND DEEP LEARNING BASICS – PART #1

- Artificial Neural Networks Basics: Single Neuron Model
- Activation Functions
- Multi-Layer Perceptron Model
- How do Artificial Neural Networks Train?
- ANN Parameters Tuning – Learning rate and batch size
- Tensorflow playground
- Gradient Descent and Backpropagation
- Overfitting and Under fitting
- How to overcome overfitting?
- Bias Variance Trade-off
- L1 Regularization
- L2 Regularization

SECTION #9: MACHINE AND DEEP LEARNING BASICS – PART #2

- Artificial Neural Networks Architectures
- Convolutional Neural Networks
- Recurrent Neural Networks
- Vanishing Gradient Problem
- LSTM Networks
- Model Performance Assessment – Confusion Matrix
- Model Performance Assessment – Precision, recall, F1-score
- Model Performance Assessment – ROC, AUC, Heatmap, and RMSE
- K-Fold Cross validation
- Transfer Learning
- Ensemble Learning – Bagging and Boosting

DOMAIN #3 OVERVIEW:

SECTION #10: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #1

- AWS SageMaker
- Deep Learning on AWS
- SageMaker Built-in algorithms
- Object Detection
- Image Classification
- Semantic Segmentation
- SageMaker Linear Learner
- Factorization Machines
- XG-Boost
- SageMaker Seq2Seq
- SageMaker DeepAR
- SageMaker Blazing Text

SECTION #11: MACHINE AND DEEP LEARNING IN AWS – BUILT-IN ALGORITHMS PART #2

- Object2Vec
- Random Cut Forest
- Neural Topic Model
- LDA
- K-Nearest Neighbours (KNN)
- K Means
- Principal Component Analysis (PCA)
- IP insights
- Reinforcement Learning
- Automatic Model Tuning
- SageMaker and Spark

DOMAIN #3 OVERVIEW:

SECTION #12: MACHINE AND DEEP LEARNING IN AWS – HIGH LEVEL AI/ML PART #3

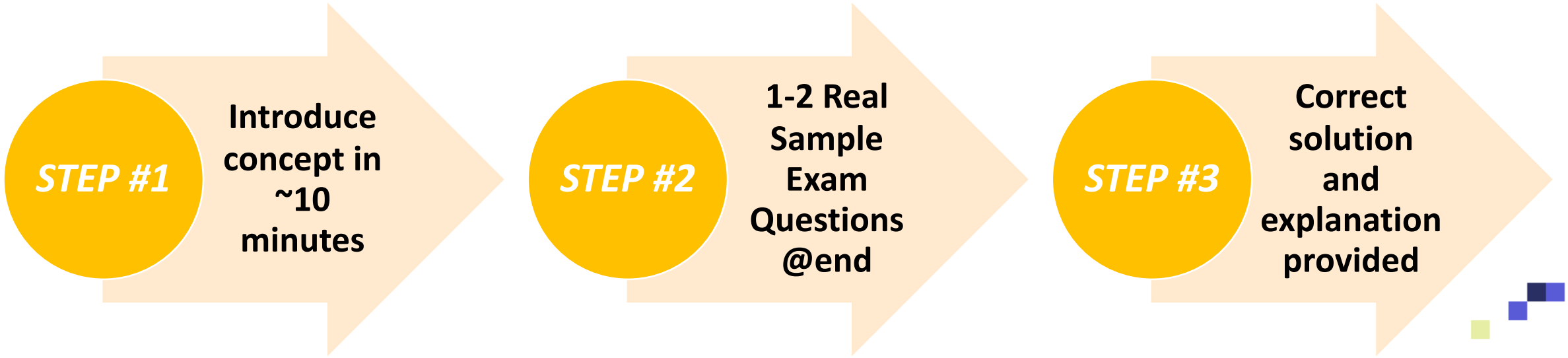
- ReKognition
- Amazon Comprehend and Comprehend Medical
- Translate
- Transcribe
- Polly
- Forecast
- Lex
- Personalize
- Textract
- AWS DeepLens
- AWS DeepRacer

LECTURE DESIGN

- We know how hard it is to study for an exam especially if you have a busy schedule.
- This course is designed to be extremely on point and optimized to pass the exam.

No boring content. Zero unnecessary information.

- Here's the lecture structure that we will follow:



VALUABLE PRIZE!

- For those of you who will successfully complete the entire Module and watch the videos till the end, they will receive a valuable prize!

10 NEW SAMPLE EXAM QUESTIONS + COMPLETE ANSWER KEY

https://pixabay.com/illustrations/trophy-win-prize-transparent-1414791/

GAME AND MINI CHALLENGES!

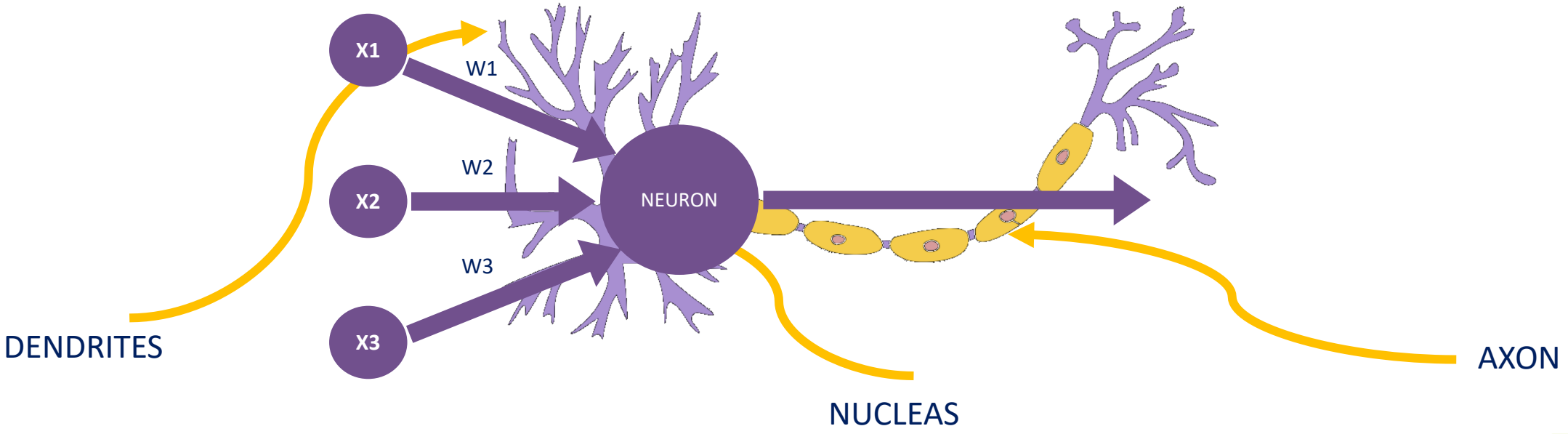
- Unfortunately, you can’t skip the videos.
- You have to collect a code throughout the lectures to unlock the exam.
- Special characters will appear at random moments throughout the video.
- You will need to collect the code and enter it to a website to access the material.
- That’s what the final code might look like!

ARTIFICIAL NEURAL NETWORKS BASICS

SINGLE NEURON MODEL

NEURON MATHEMATICAL MODEL

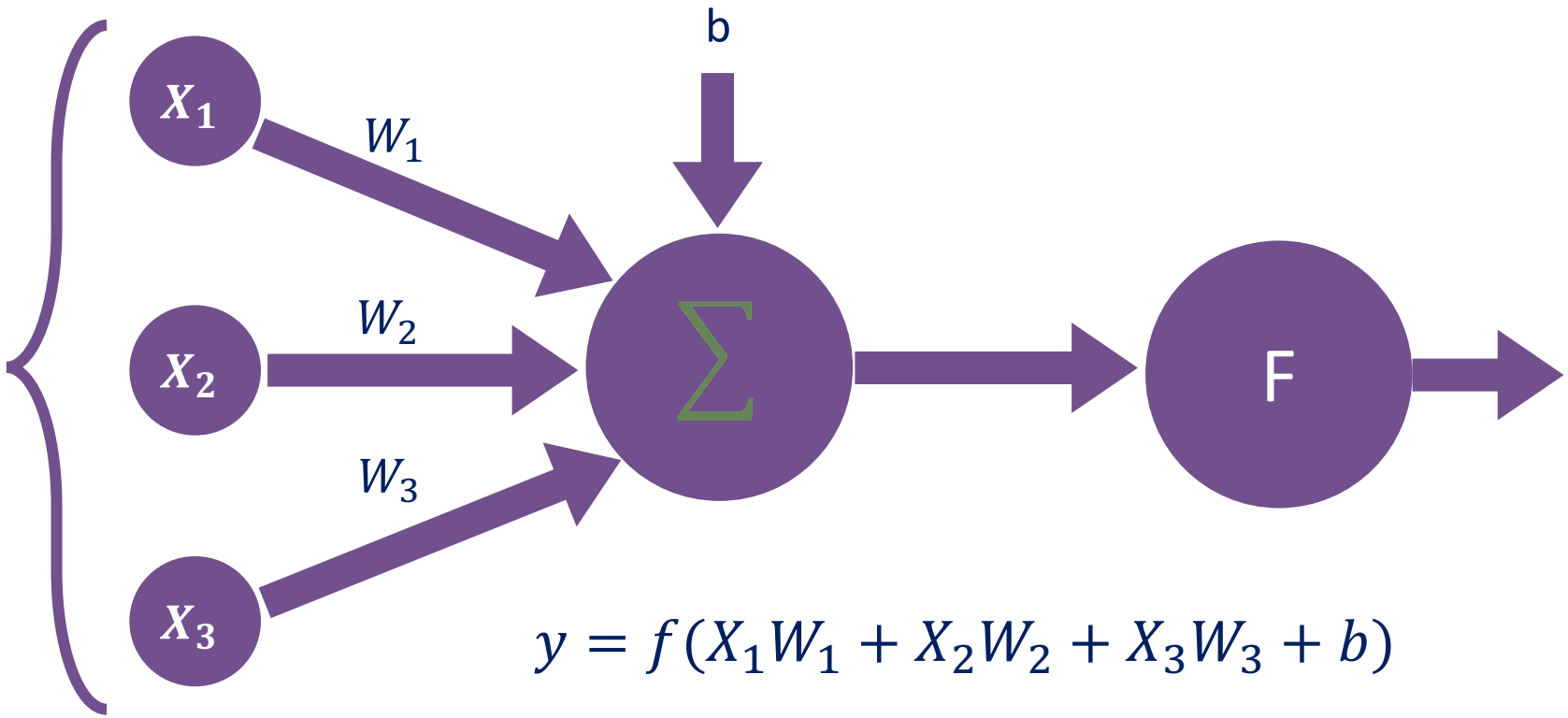
- **The neuron collects signals from input channels named dendrites, processes information in its nucleus, and then generates an output in a long thin branch called axon.**



NEURON MATHEMATICAL MODEL

- **Bias allows to shift the activation function curve up or down.**
- **Number of adjustable parameters = 4 (3 weights and 1 bias).**
- **Activation function “F”.**

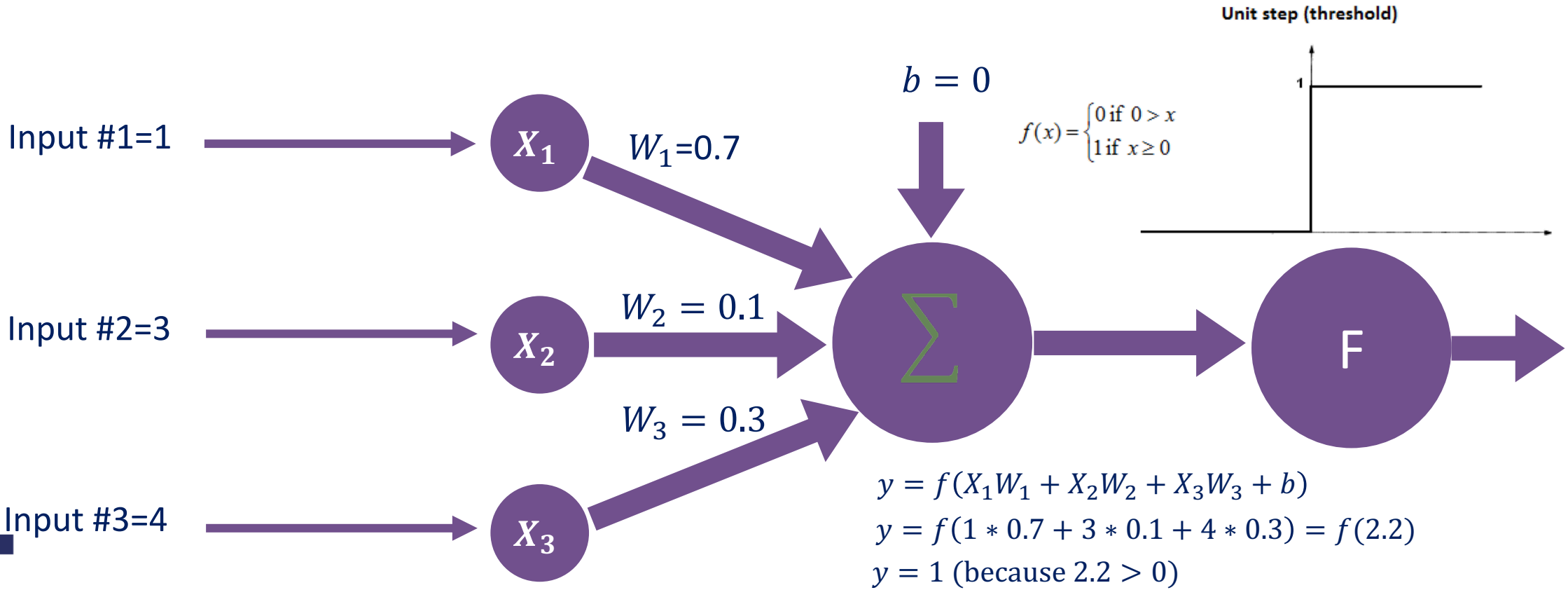
INPUTS/
INDEPENDENT
VARIABLES



$$y = f(X_1W_1 + X_2W_2 + X_3W_3 + b)$$

NEURON MODEL IN ACTION!

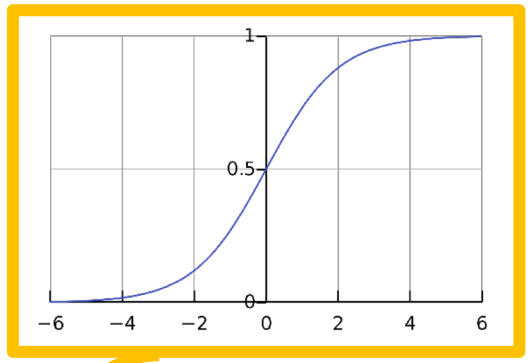
- Let's assume an activation function of Unit Step.
- The activation functions is used to map the input between (0, 1).

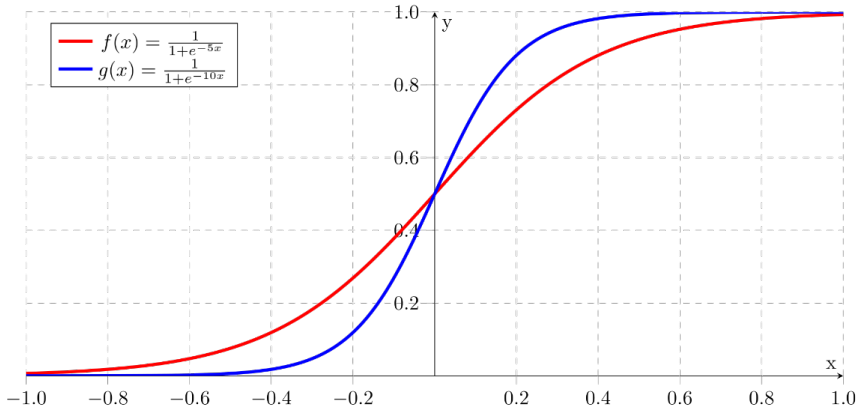


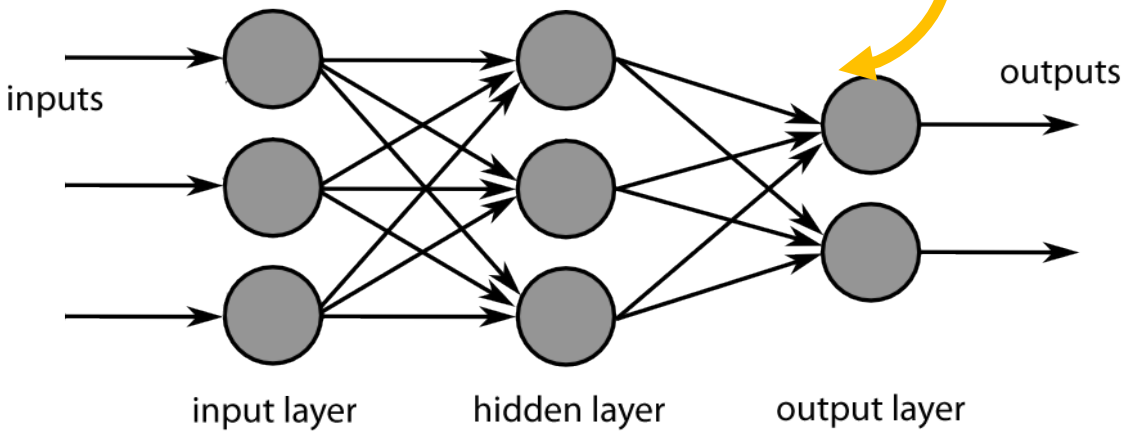
ACTIVATION FUNCTIONS

ACTIVATION FUNCTIONS

- **SIGMOID:**
  - **Takes a number and sets it between 0 and 1**
  - **Converts large negative numbers to 0 and large positive numbers to 1.**
  - **Generally used in output layer.**



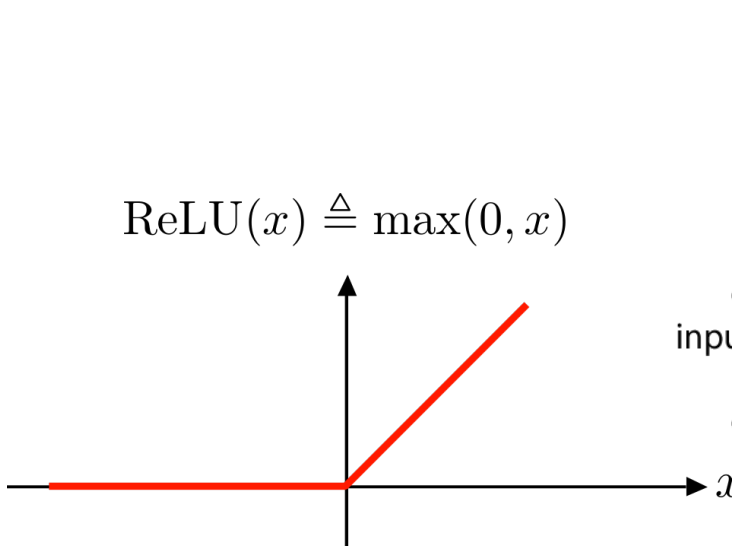


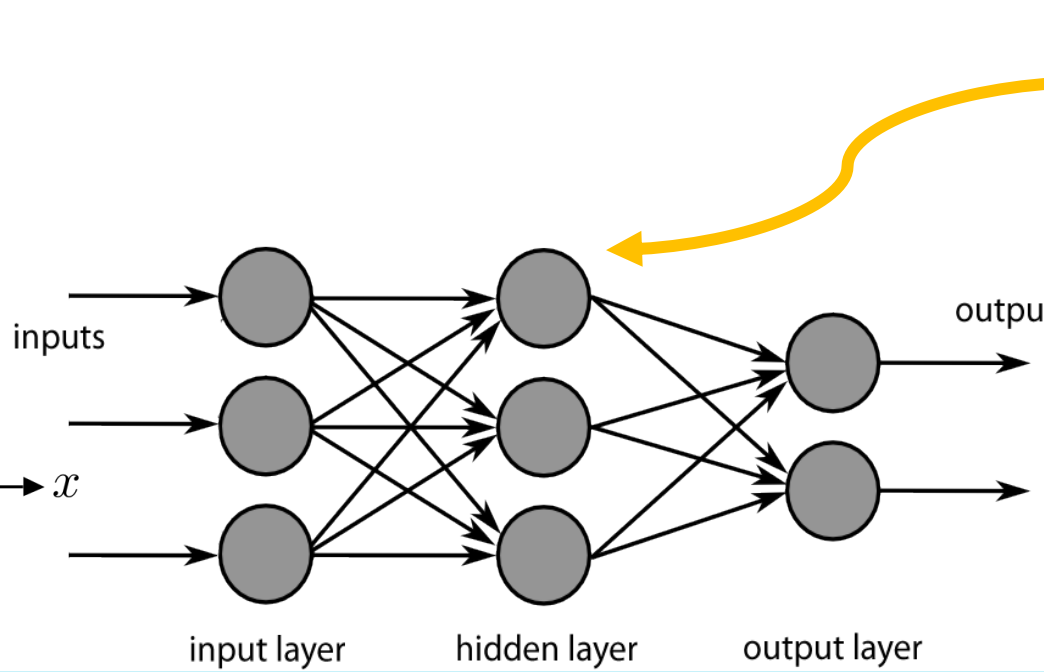


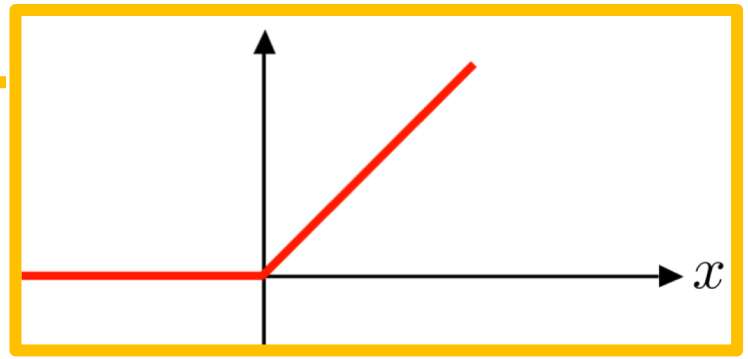
- **Photo credit:** https://commons.wikimedia.org/wiki/File:Sigmoid-function.svg
- **Photo Credit:** https://fr.m.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png
- **Photo Credit:** https://commons.wikimedia.org/wiki/File:Logistic-curve.svg

ACTIVATION FUNCTIONS

- **RELU (RECTIFIED LINEAR UNITS):**
  - **if input x < 0, output is 0 and if x > 0 the output is x.**
  - **RELU does not saturate so it avoids vanishing gradient problem.**
  - **It uses simple thresholding so it is computationally efficient.**
  - **Generally used in hidden layers.**



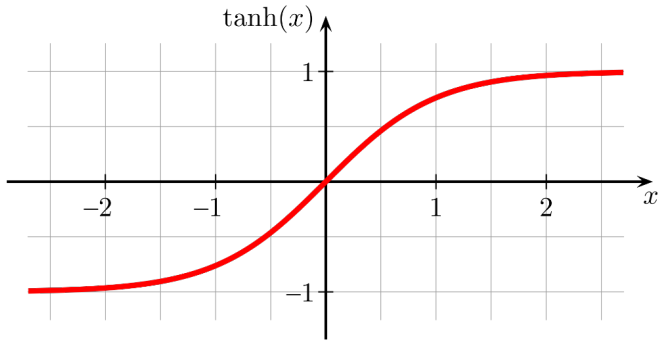


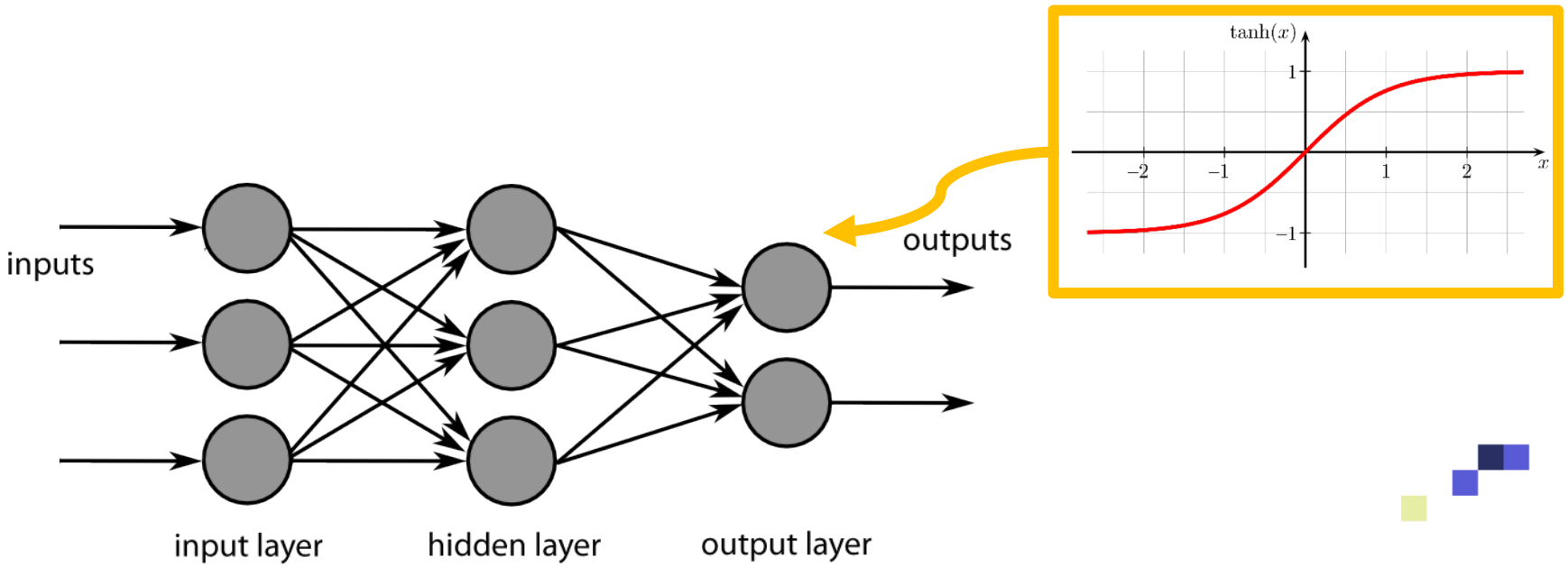


- **Photo credit:** https://commons.wikimedia.org/wiki/File:ReLU_and_Nonnegative_Soft_Thresholding_Functions.svg
- **Photo Credit:** https://fr.m.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png

ACTIVATION FUNCTIONS

- **HYPERBOLIC TANGENT ACTIVATION FUNCTION:**
  - **“Tanh” is similar to sigmoid, converts number between -1 and 1.**
  - **Unlike sigmoid, tanh outputs are zero-centered (range: -1 and 1).**
  - **Tanh suffers from vanishing gradient problem so it kills gradients when saturated.**
  - **In practice, tanh is preferable over sigmoid.**





- **Photo credit:** https://commons.wikimedia.org/wiki/File:Hyperbolic_Tangent.svg
- **Photo Credit:** https://fr.m.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png

MULTI-NEURON MODEL (MULTI-LAYER PERCEPTRON MODEL)

MULTI-LAYER PERCEPTRON NETWORK

- The network is represented by a matrix of weights, inputs and outputs.
- Total Number of adjustable parameters = 8:
- Weights = 6
- Biases = 2

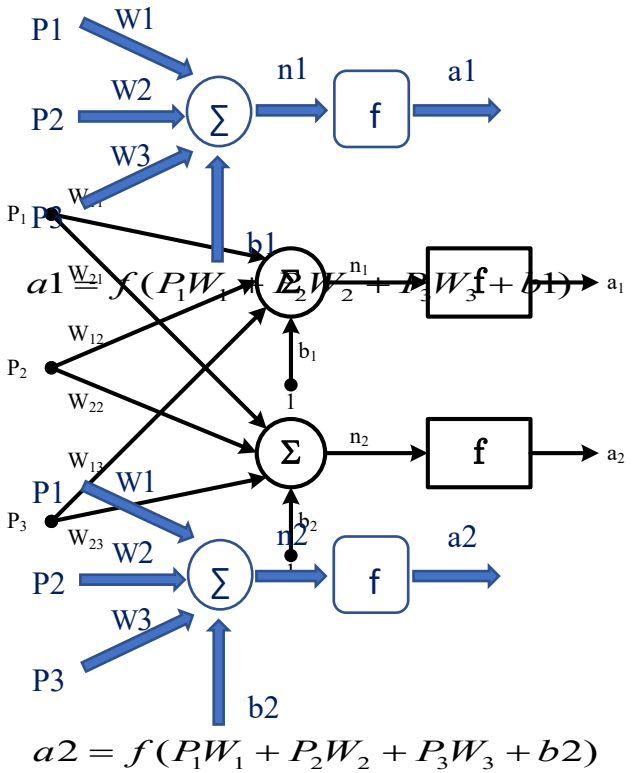
Matrix Representation

$$P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

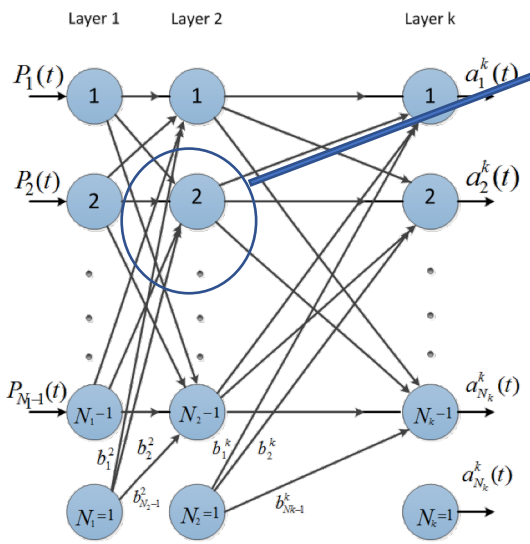
$$a = f(W \times P + b)$$

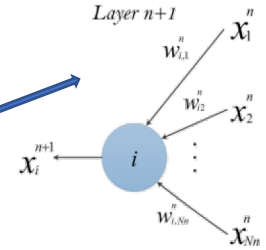


MULTI-LAYER PERCEPTRON NETWORK

- Let's connect multiple of these neurons in a multi-layer fashion.
- The more hidden layers, the more "deep" the network will get.

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{N_1} \end{bmatrix}$$





$$x_i^{n+1}(t) = \varphi\left(\sum_{j=1}^{N_n} w_{i,j}^n \, x_j^n(t)\right)$$

Node *(n+1, i)* representation

$$\begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1,N_1} \\ W_{21} & W_{22} & & W_{2,N_1} \\ \vdots & & \ddots & \vdots \\ W_{m-1,1} & W_{m-1,2} & \cdots & W_{m-1,N_1} \\ W_{m,1} & W_{m,2} & & W_{m,N_1} \end{bmatrix}$$

Non-Linear Sigmoid Activation function

$$\varphi(w) = \frac{1}{1 + e^{-w}}$$

m: number of neurons in the hidden layer

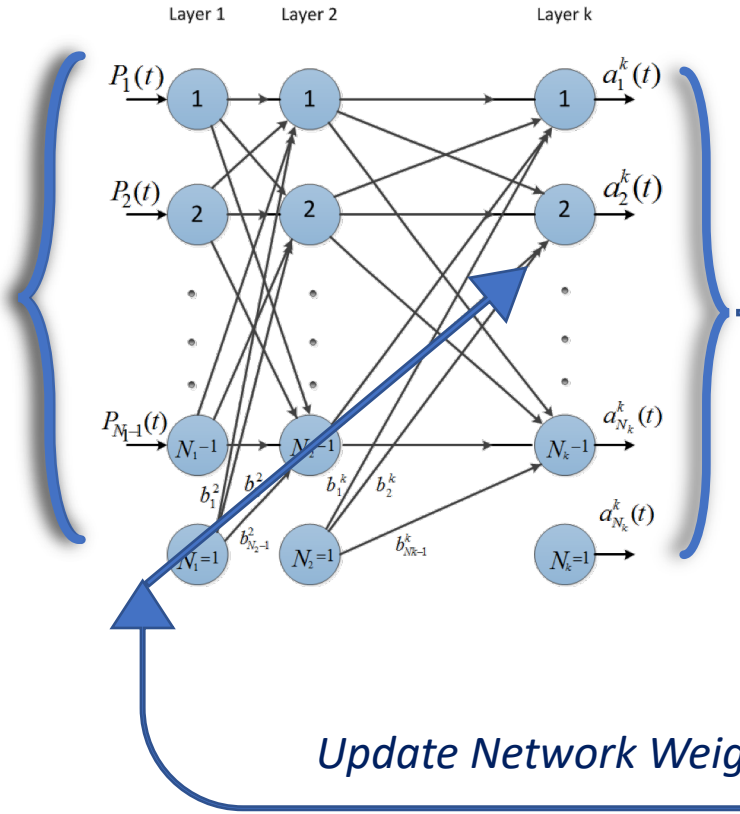
N_1: number of inputs

HOW DO ANNS TRAIN?

SUPERVISED ANN TRAINING

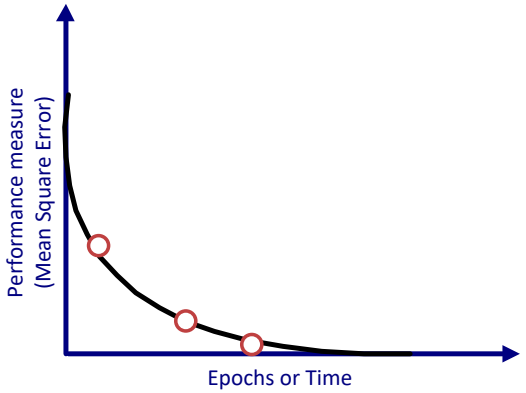
Testing inputs X

Training inputs X



Predicted Output $\hat{Y}$

Desired (True) Output Y



Error = $\hat{Y} - Y$

$$C = \tfrac{1}{2}\left(\hat{Y} - Y\right)^2$$

Update Network Weights

ANN TRAINING STRATEGIES

Supervised learning

- Used if there is large set of test data with known labels (outputs).
- The learning algorithm evaluates output (i.e.: makes predictions), compares output against the label, and adjust network and repeat.

Unsupervised learning

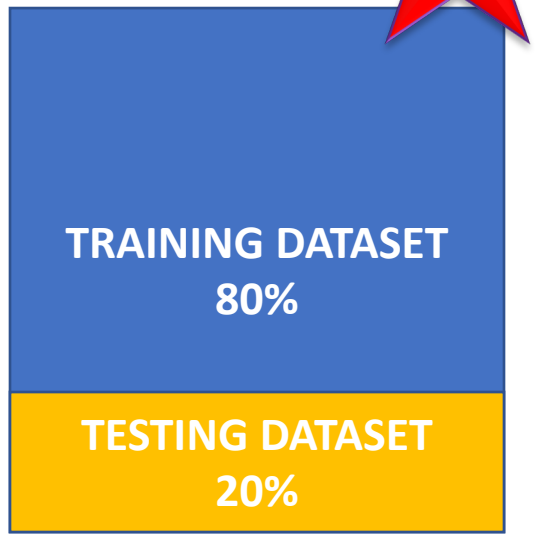
- Used with "unlabeled" data (not categorized) (Ex: k-means clustering).
- Since learning algorithm works with unlabeled data, there is no way to assess the accuracy of the structure suggested by the algorithm

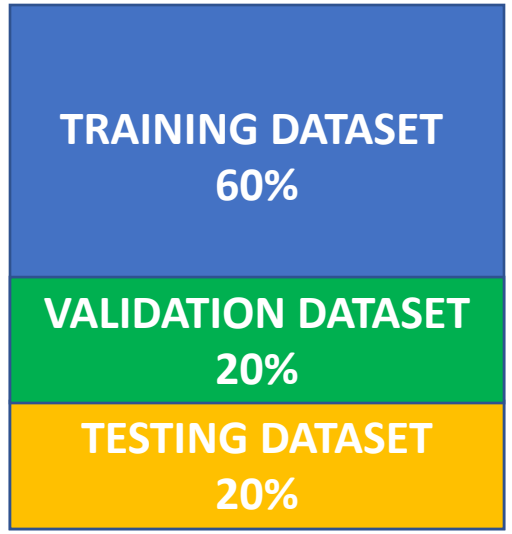
Reinforced learning

- Learning algorithm takes actions that maximizes some notion of cumulative reward.
- Over time, the network learns to prefer the right kind of action and to avoid the wrong one.

DIVIDE DATA INTO TRAINING AND TESTING

- Data set is generally divided into 80% for training and 20% for testing.
- Sometimes, we might include cross validation dataset as well and then we divide it into 60%, 20%, 20% segments for training, validation, and testing, respectively (numbers may vary).
  1. Training set: used for gradient calculation and weight update.
  2. Validation set:
     - used for cross-validation which is performed to assess training quality as training proceeds.
     - Cross-validation is implemented to overcome over-fitting (over-training). Over-fitting occurs when algorithm focuses on training set details at cost of losing generalization ability.
     - Trained network MSE might be small during training but during testing, the network may exhibit poor generalization performance.
  3. Testing set: used for testing trained network.





YPERPARAMETERS (BATCH SIZE/LEARNING RATE)

ANN HYPERPARAMETERS TUNING

HYPERPARAMETER

- **Hyper parameter:** values set prior to the training process such as number of neurons, layers, learning rate..etc
- Selecting the best hyperparameter values (via optimization) is an extremely challenging task.

PARAMETER

- **Parameter:** values that are obtained by the training process such as network weights and biases.

LEARNING RATE

- ANNs are trained using gradient descent algorithm.
- An important parameter used during training is known as the “learning rate”.
- Learning rate is a hyperparameter that represents the size of the steps taken which indicates how aggressive you’d like to update the parameters.
- **If learning rate increases, the area covered in the search space will increase so we might reach global minimum faster.**
- **However, we can overshoot the target.**
- **For small learning rates, training will take much longer to reach optimized weight values**

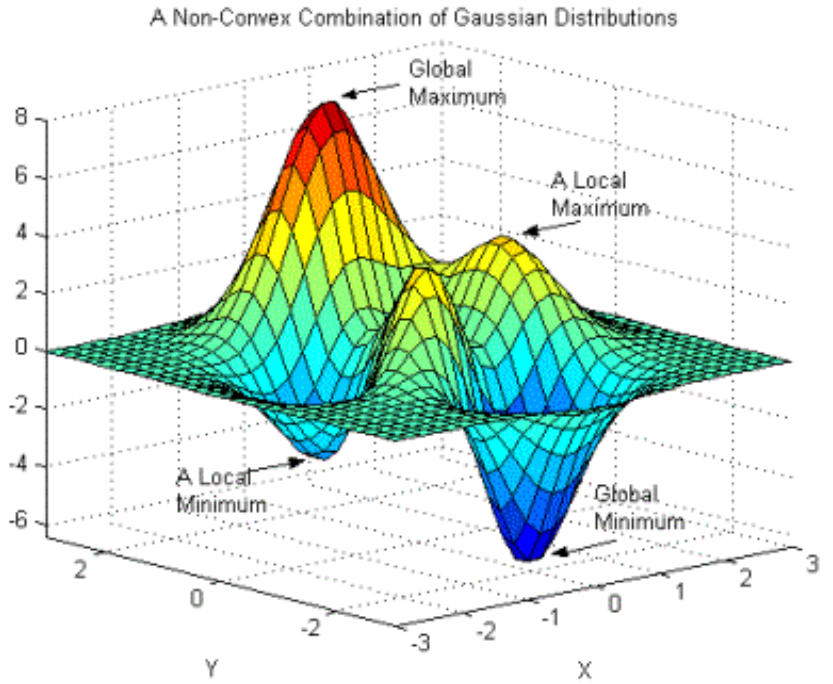
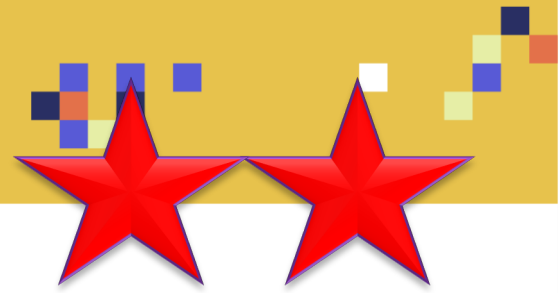


Photo Credit: https://commons.wikimedia.org/wiki/File:Non-Convex_Objective_Function.gif

BATCH SIZE



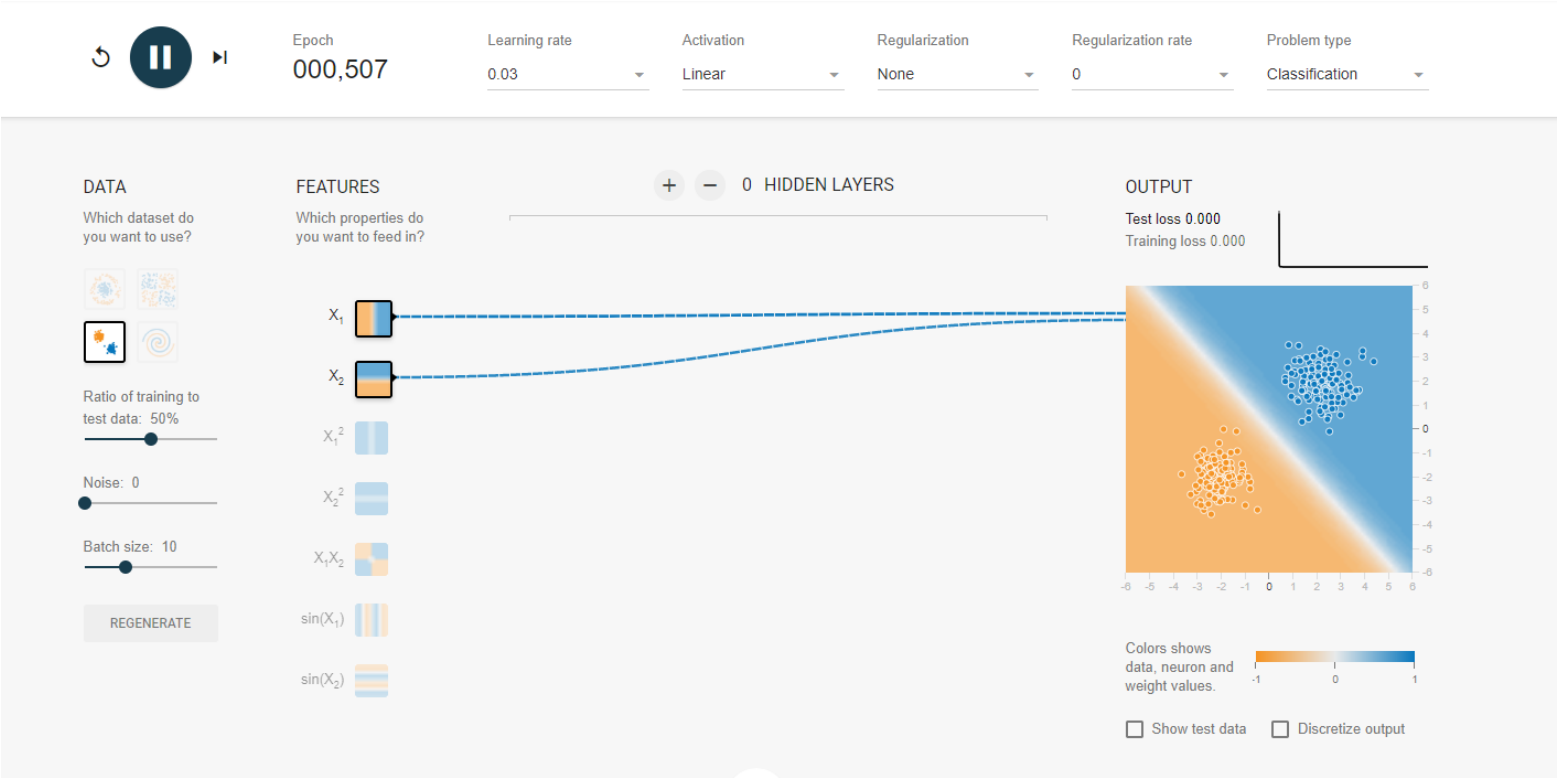
- Batch size indicates the number of samples that will propagate through the network.
- Example:
  - Let's assume that we have 1000 images for training.
  - For batch size = 50, the first 50 images (from index 1 to index 50) will be propagated to network and used for training.
  - Then the next 50 images are propagated (index 51 to index 100).
  - Procedure is repeated until we use all the training data.

- You can use large or small batch size, in the previous example, you can use batch size = 50 or 1000.
- Note that we can shuffle the training dataset between samples and this will lead to very different results every time we train the network.
- **If the batch size is small, the ANN can easily escape local minimum areas!**
- **If the batch size is large, the ANN can get stuck in a local minimum.**

TENSORFLOW PLAYGROUND

DEEP DIVE INTO TF PLAYGROUND

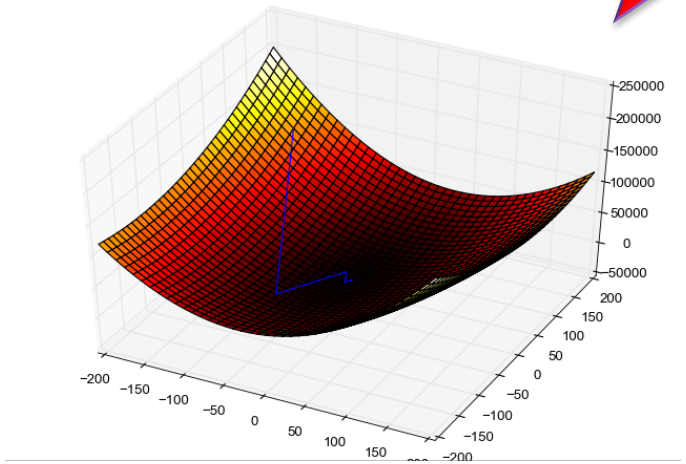
- Check this out: https://playground.tensorflow.org



GRADIENT DESCENT AND BACK PROPAGATION

GRADIENT DESCENT

- Gradient descent is an optimization algorithm used to obtain the optimized network weight and bias values
- It works by iteratively trying to minimize the cost function
- It works by calculating the gradient of the cost function and moving in the negative direction until the local/global minimum is achieved
- If the positive of the gradient is taken, local/global maximum is achieved



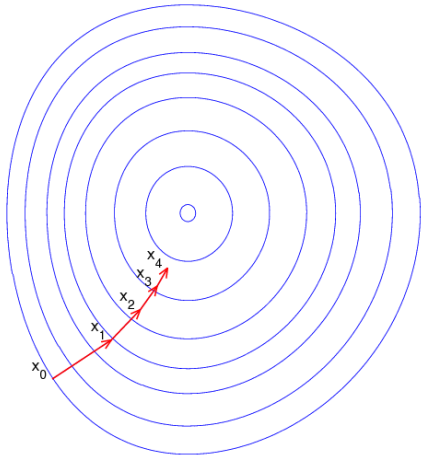


Photo Credit: https://commons.wikimedia.org/wiki/File:Gradient_descent_method.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Gradient_descent.png

GRADIENT DESCENT

Gradient descent works as follows:

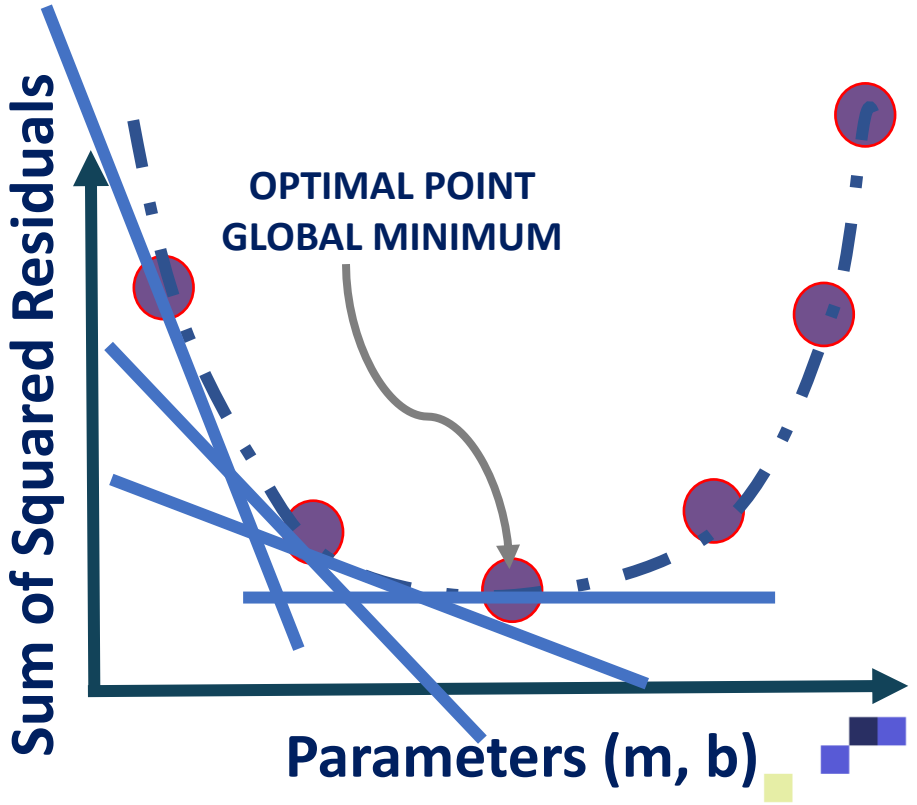
1. Calculate the derivative (gradient) of the Loss function
2. Pick random values for parameters m, b and substitute
3. Calculate the step size (how much are we going to update the parameters?)

$$Step\ size\ =\ Slope\ *\ learning\ rate$$

4. Update the parameters and repeat

$$y = b + m * x$$

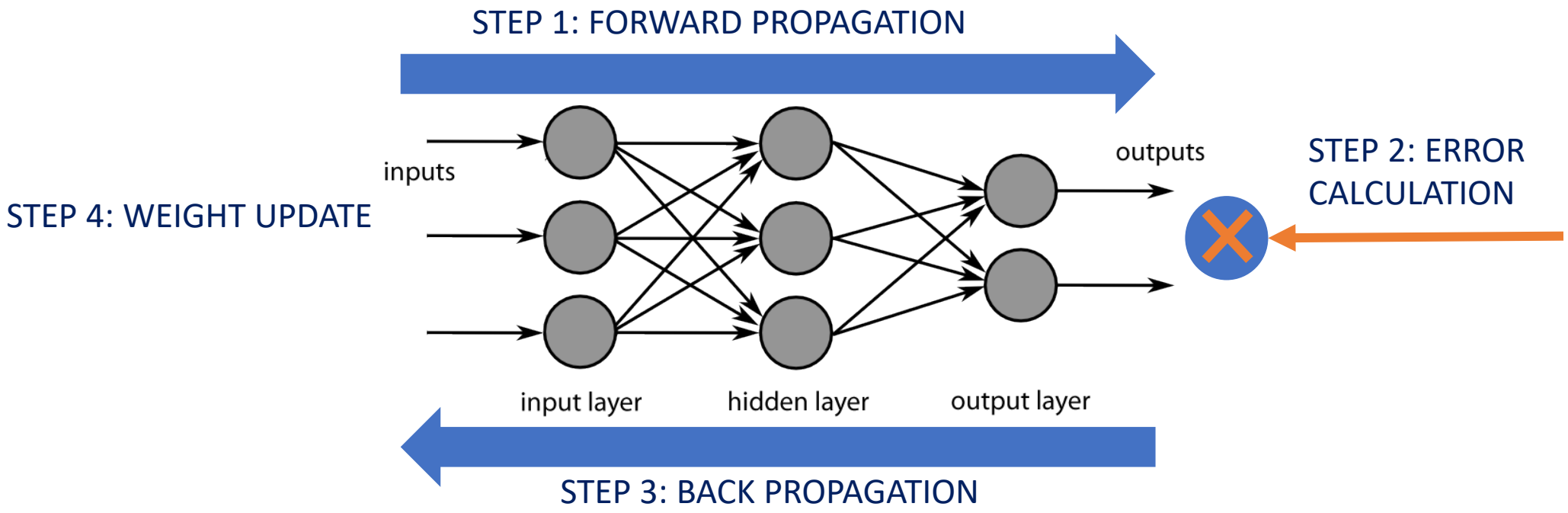
GOAL IS TO FIND
BEST PARAMETERS



**Note: in reality, this graph is 3D and has three axes, one for m, b and sum of squared residuals*

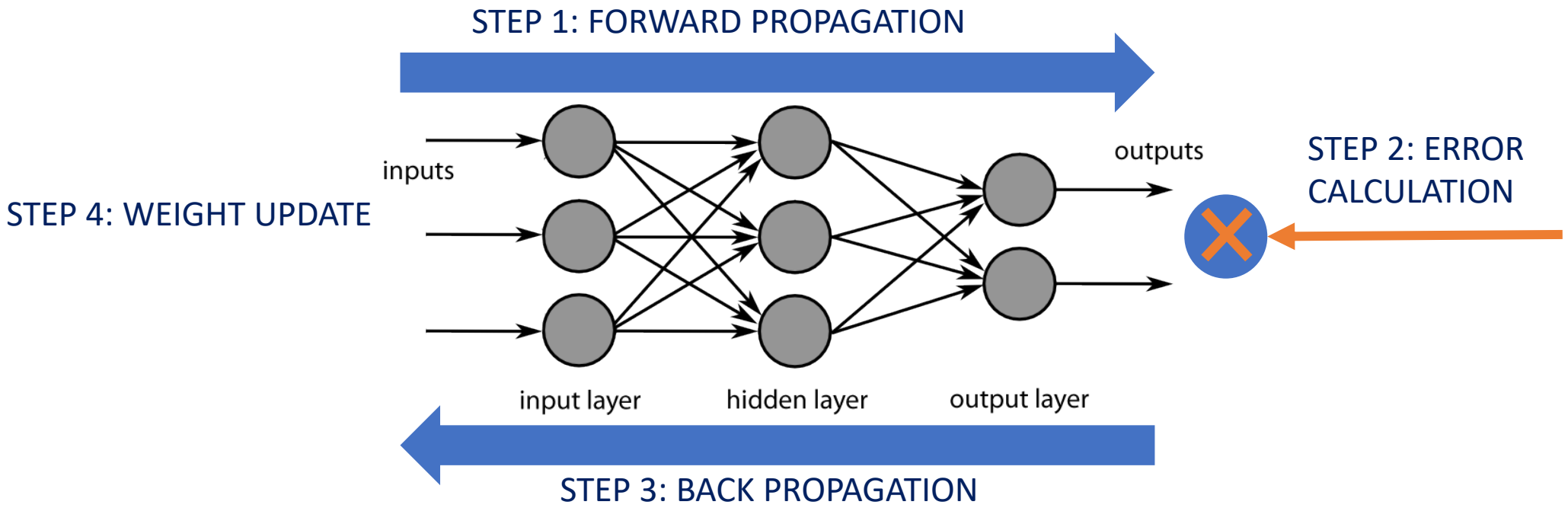
BACK PROPAGATION

- **Backpropagation is a method used to train ANNs by calculating gradient needed to update network weights.**
- **It is commonly used by the gradient descent optimization algorithm to adjust the weight of neurons by calculating the gradient of the loss function.**



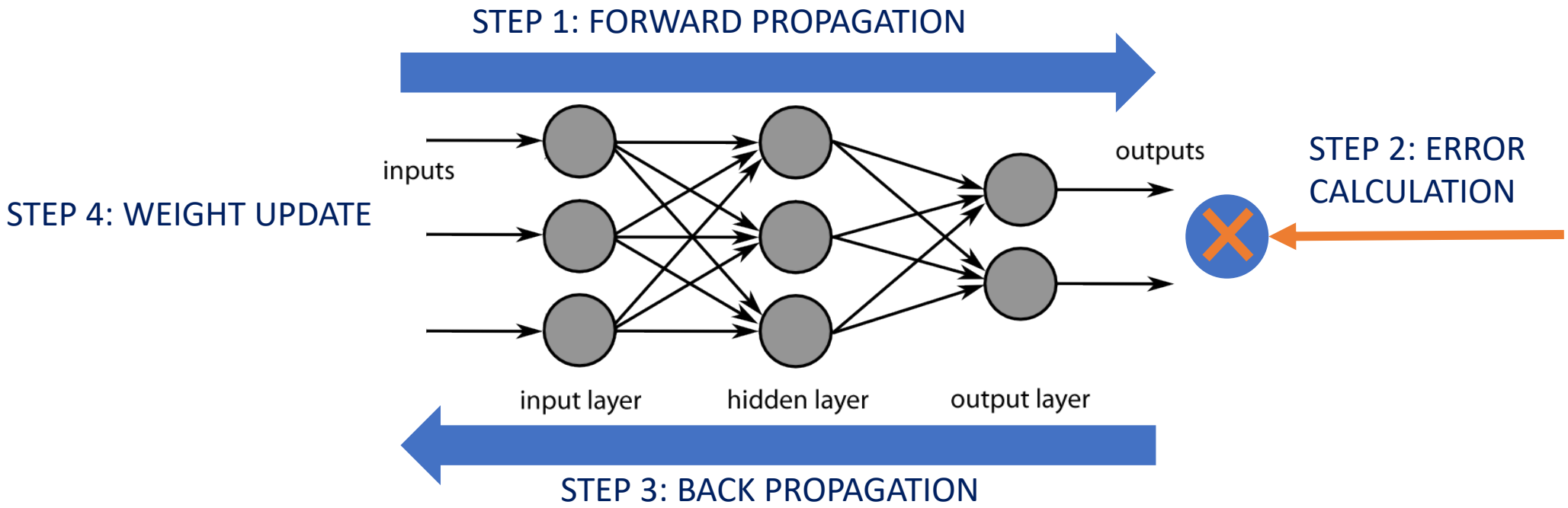
BACK PROPAGATION

- **Backpropagation Phase 1: propagation**
  - **Propagation forward through the network to generate the output value(s)**
  - **Calculation of the cost (error term)**
  - **Propagation of output activations back through network using training pattern target in order to generate the deltas (difference between targeted and actual output values)**



BACK PROPAGATION

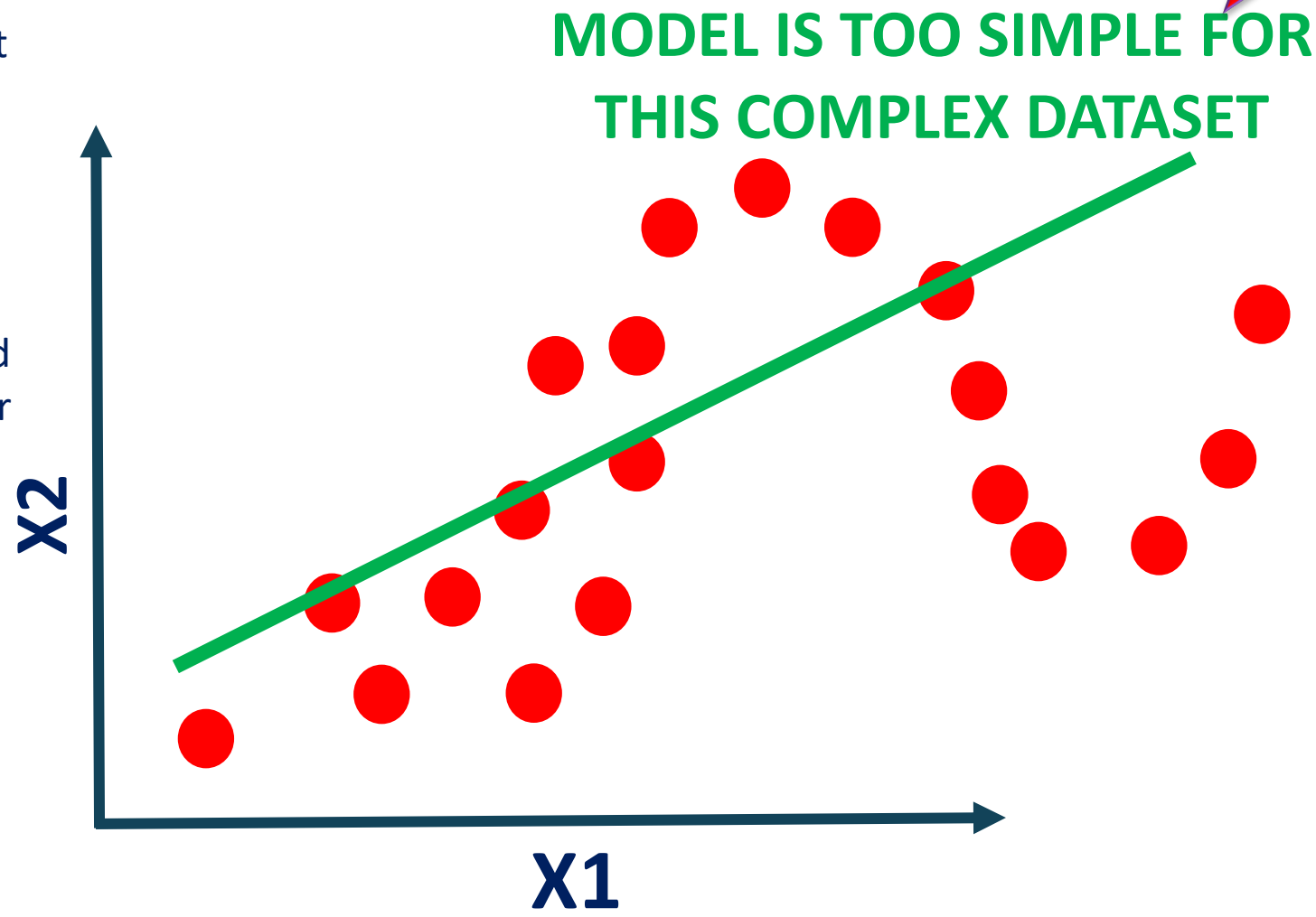
- **Phase 2: weight update**
  - **Calculate weight gradient.**
  - **A ratio (percentage) of the weight's gradient is subtracted from the weight.**
  - **This ratio influences the speed and quality of learning and called learning rate. The greater the ratio, the faster neuron train, but lower ratio, more accurate the training is.**



OVERFITTING Vs. UNDERFITTING MODELS

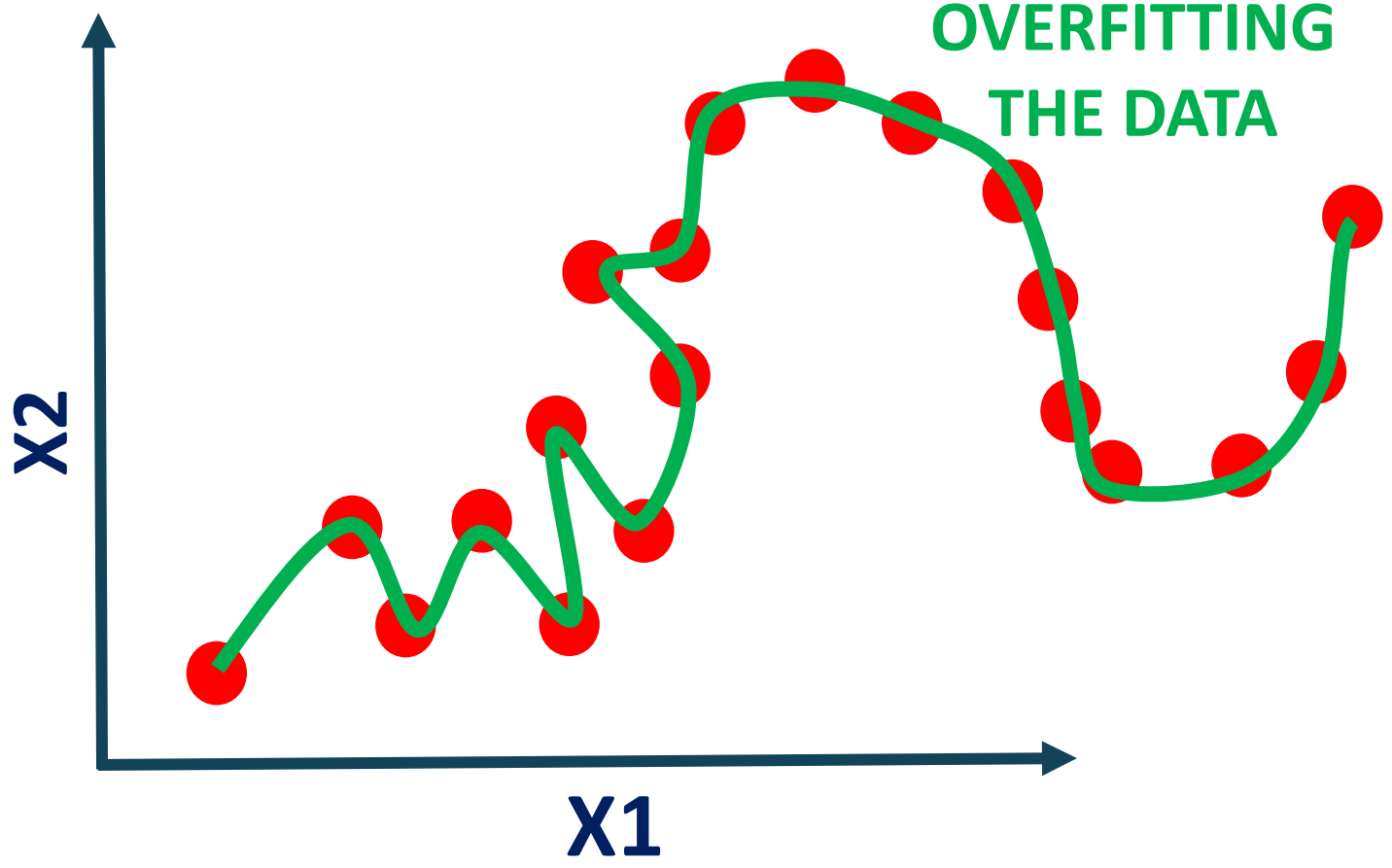
UNDERFITTING MODEL

- Model is under fitting if it's too simple that it cannot reflect the complexity of the training dataset.
- We can overcome under fitting by:
  - increasing the complexity of the model.
  - Training the model for a longer period of time (more epochs) to reduce error



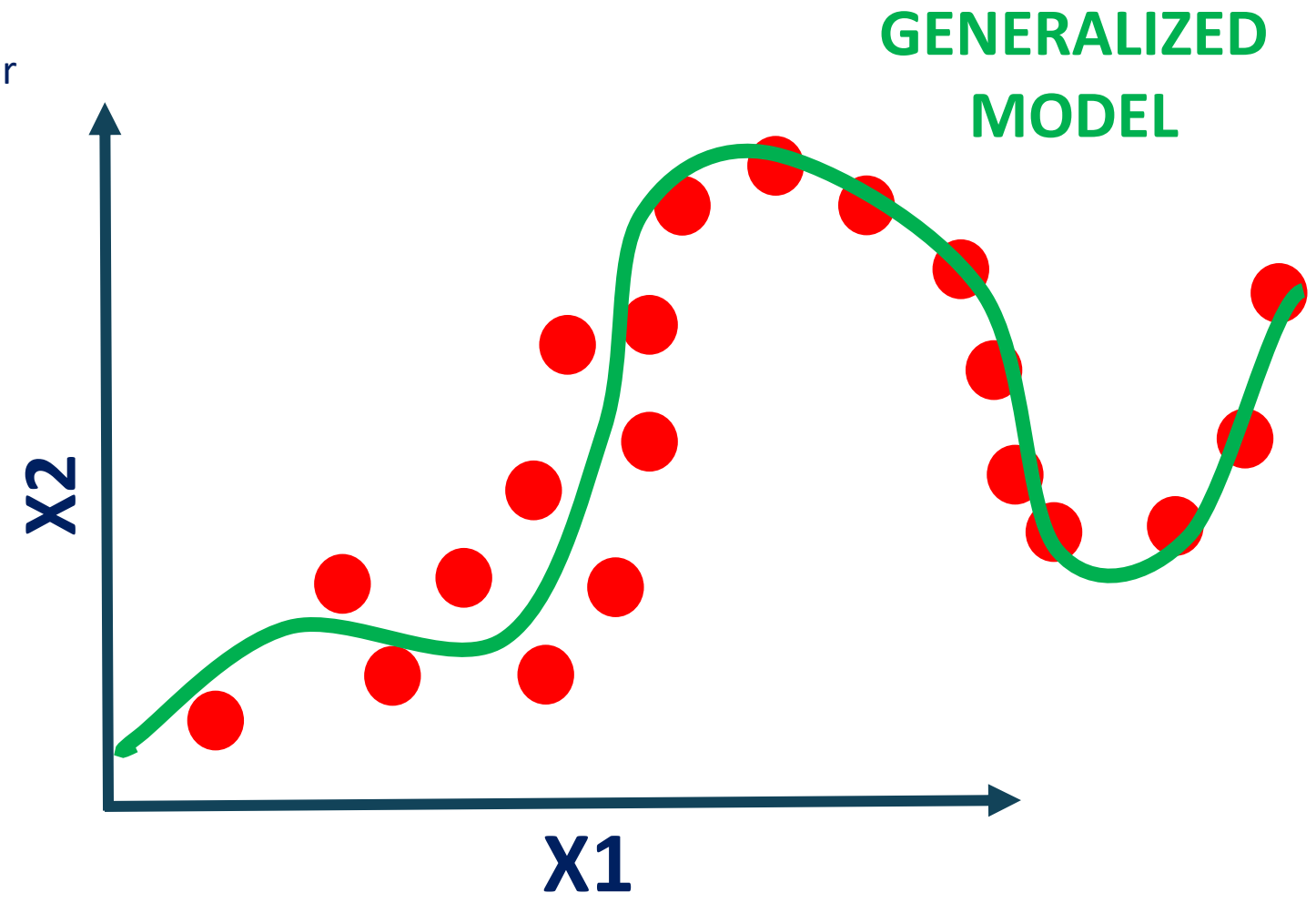
OVERFITTING MODEL

- Model is overfitting data when it memorizes all the specific details of the training data and fails to generalize.
- Overfitting models tend to perform very well on the training dataset but poorly on any new dataset (testing dataset)
- Machine learning is the art of creating models that are able to generalize and avoid memorization.



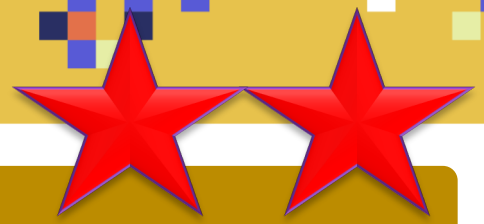
BEST MODEL (GENERALIZED)

- Model that performs well during training and testing (on new dataset that has never seen before) is considered the best model (goal).



HOW TO OVERCOME OVERFITTING?

HOW TO OVERCOME OVERFITTING?



PERFORM EARLY STOPPING

- Stop training when you notice that the validation loss increases while training loss decreases.

DO REGULARIZATION

- Regularization improves the model generalization capability.

ADD MORE DATASET

- By increasing the size of the dataset, the model might generalize more.

PERFORM FEATURE SELECTION

- Dropping useless features could improve the model generalization capability.

USE BOOSTING AND BAGGING (ENSEMBLE LEARNING)

- By combining voting from many different models via bagging and boosting, this will improve model generalization

ADD MORE NOISE

- Adding noise might enable model to become more general

USE DROPOUT TECHNIQUE

- In Artificial Neural Network training, dropping some neurons using Dropout technique improves networks generalization ability.

HOW TO AVOID OVERFITTING?

1. Early Stopping:

- In order to avoid overfitting during ANN training, early stopping technique could be used.
- This is used to when the accuracy over the training dataset is increased but the accuracy over the validation dataset starts to decrease.
- Early stopping endures that the network is able to generalize. Meaning it will perform great over training and testing dataset.

2. Dropout Regularization

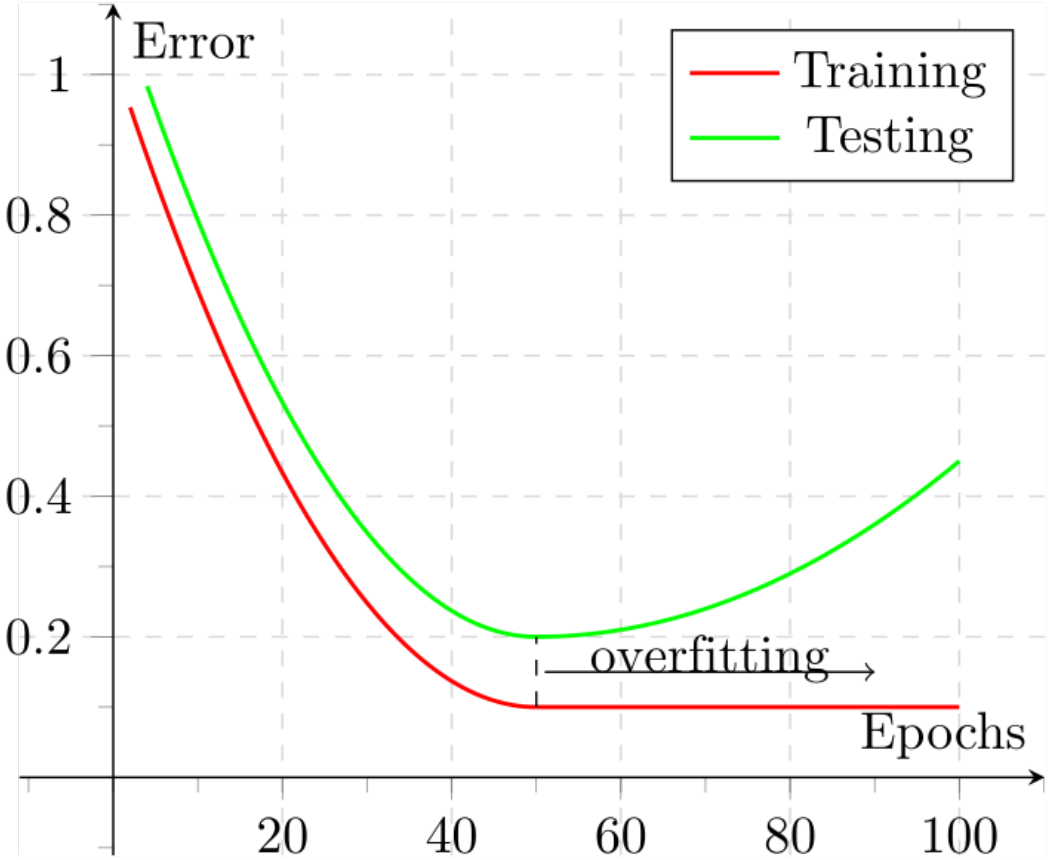
- Another technique that could be used to avoid overfitting is known as dropout regularization. Dropout regularization forces the learning to be spread out amongst the artificial neurons, further preventing overfitting.

3. Remove ANNs Layers:

- Another technique is to reduce the complexity of the ANN by removing layers, rather than adding them. This might also help prevent an overly complex model from being created.

1. EARLY STOPPING

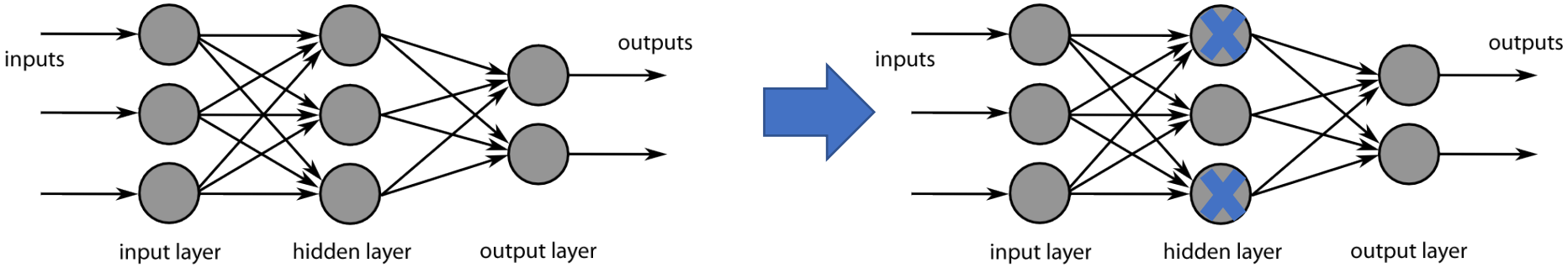
- **Early stopping can be used when the accuracy over the training dataset is increased but the accuracy over the validation (evaluation) dataset starts to decrease.**



- **Photo Credit:** https://commons.wikimedia.org/wiki/File:2d-epochs-overfitting.svg

2. DROPOUT

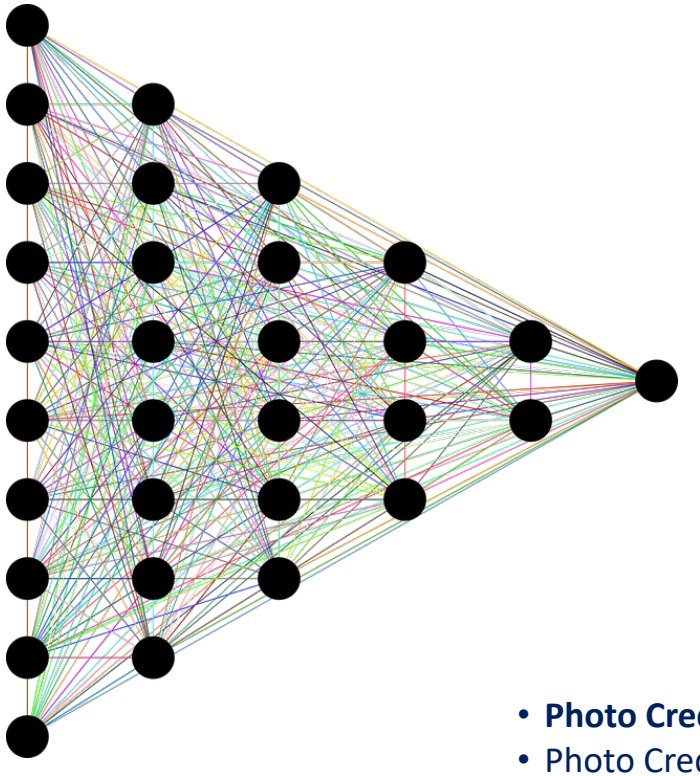
- Improve accuracy by adding a dropout.
- Dropout refers to dropping out units in a neural network.
- Neurons develop co-dependency amongst each other during training
- Dropout is a regularization technique for reducing overfitting in neural networks.
- It enables training to occur on several architectures of the neural network



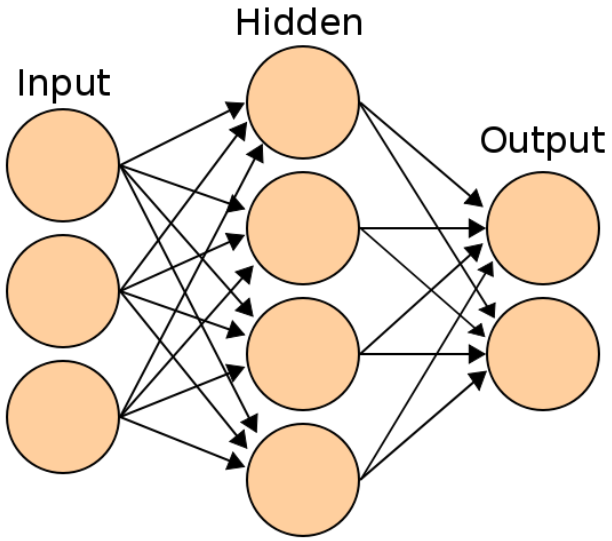
- **Photo Credit:** https://fr.m.wikipedia.org/wiki/Fichier:MultiLayerNeuralNetworkBigger_english.png

3. REMOVE ANNS LAYERS OR NEURONS (REDUCE MODEL COMPLEXITY

COMPLEX ANN WITH MANY LAYERS AND NEURONS



SIMPLE ANN WITH FEW LAYERS AND NEURONS



- **Photo Credit:** https://pixabay.com/vectors/neural-network-thought-mind-mental-3816319/
- Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

BIAS VARIANCE TRADE-OFF

BIAS VARIANCE INTUITION

- Let's assume that we want to get the relationship between the Temperature and Bike rental usage.
- As temperature experience increase, the bike rental usage tend to increase as well.
- As temperature goes beyond a certain limit, usage tend to plateau and it does not increase anymore.

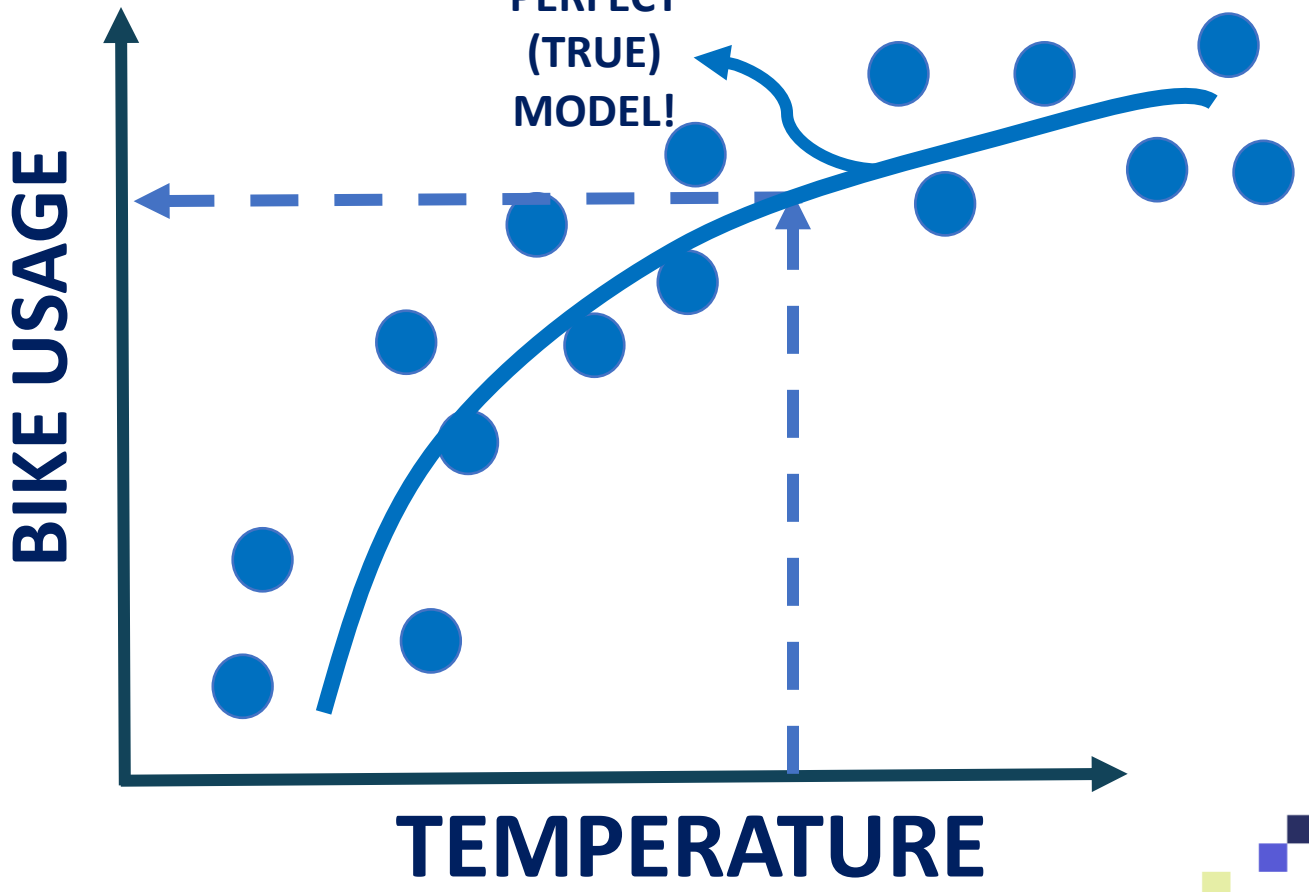
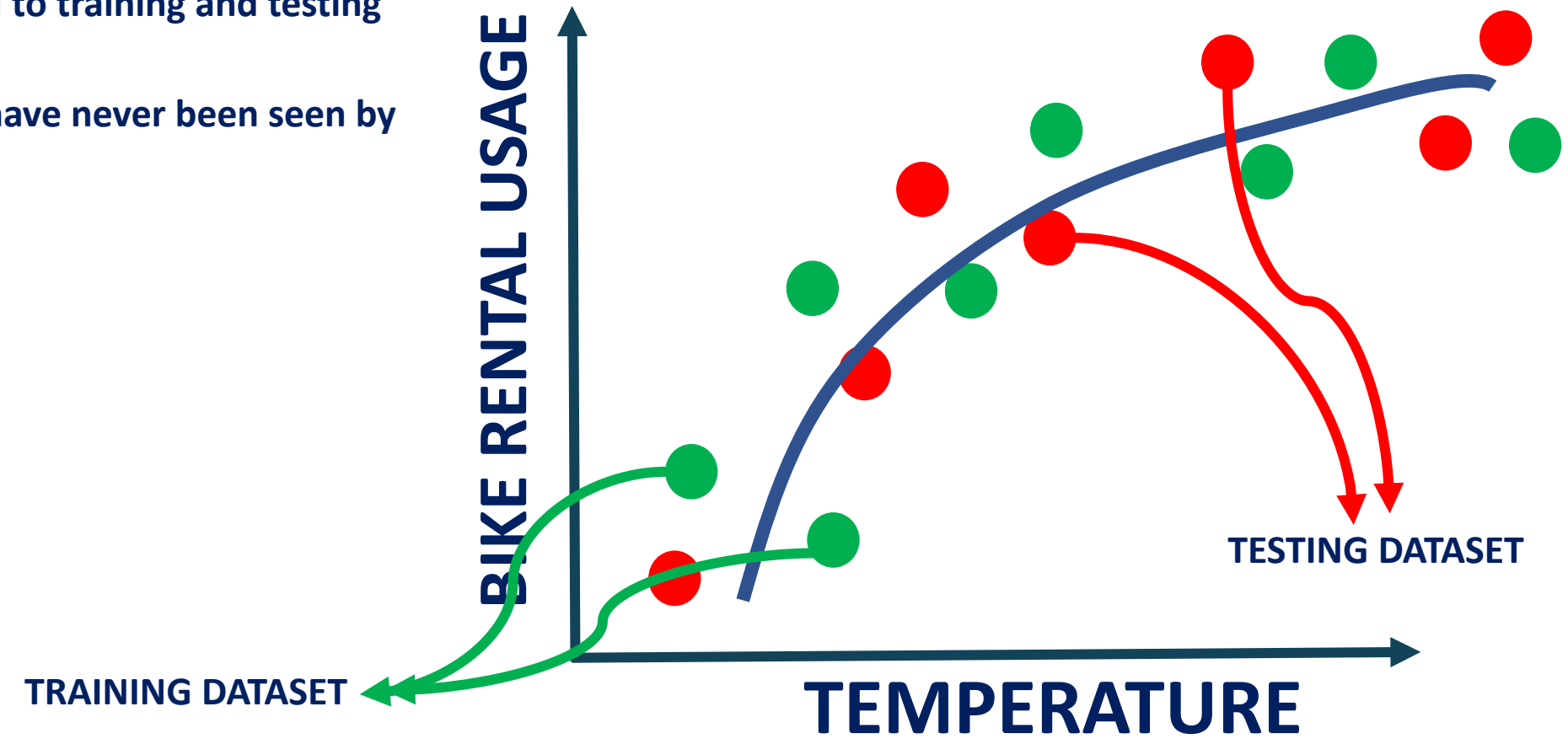


Image Source: https://pixabay.com/photos/bike-rental-bikes-rent-pay-2284380/

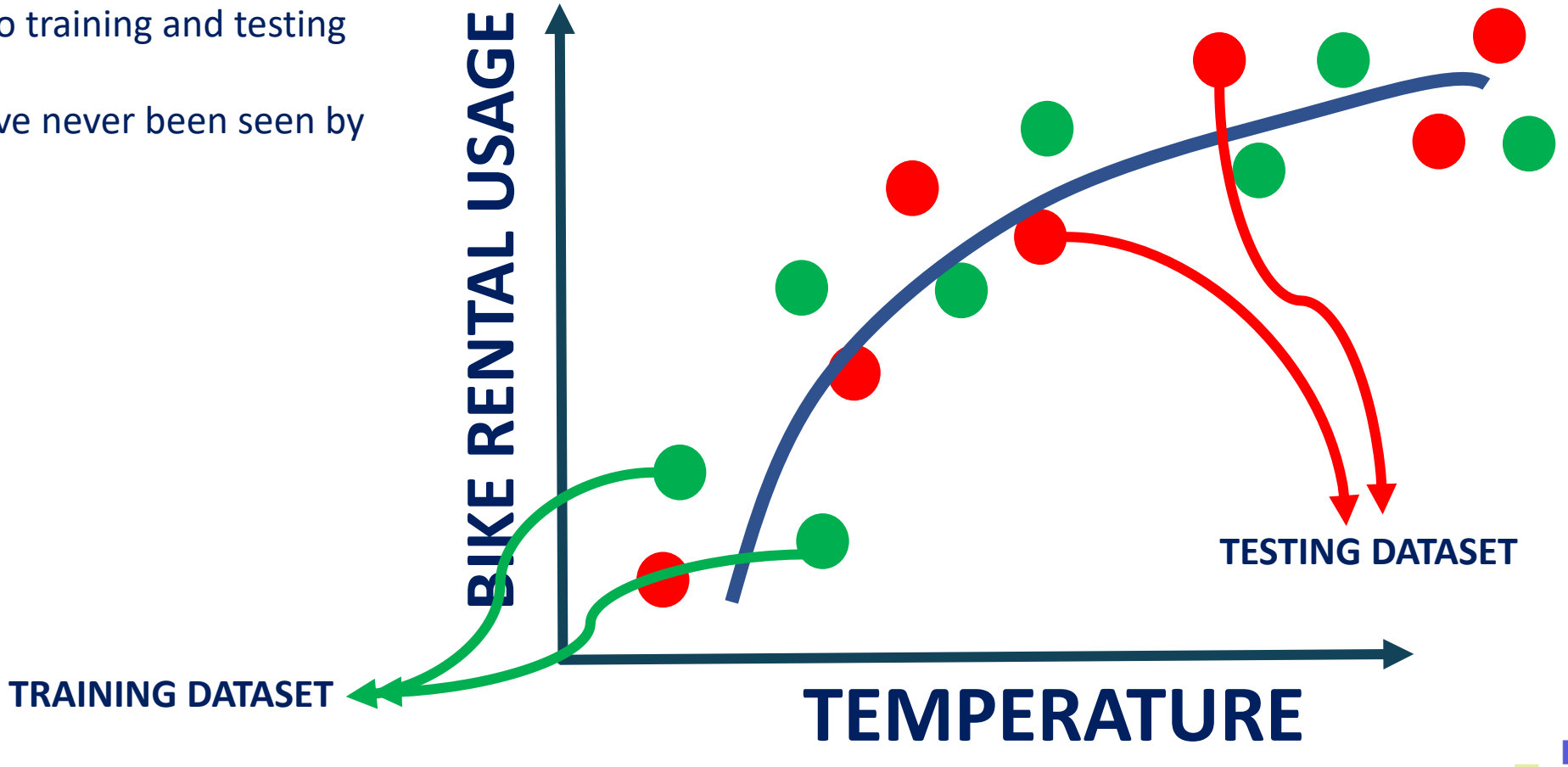
BIAS VARIANCE TRAINING VS. TESTING DATASETS

- Dataset is divided to training and testing datasets
- Testing datasets have never been seen by the model before



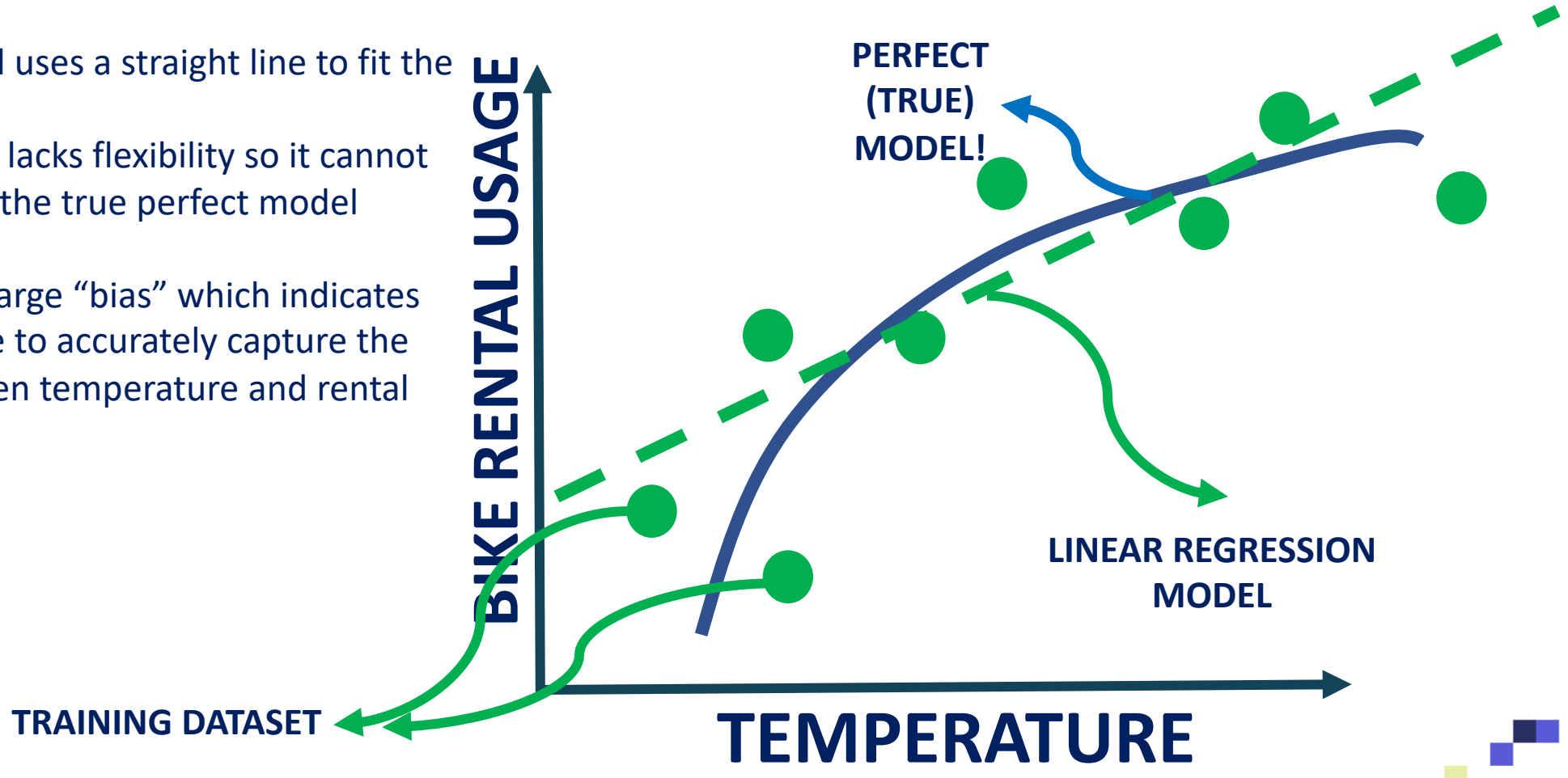
BIAS AND VARIANCE: TRAINING VS. TESTING DATASETS

- Dataset is divided to training and testing datasets
- Testing datasets have never been seen by the model before



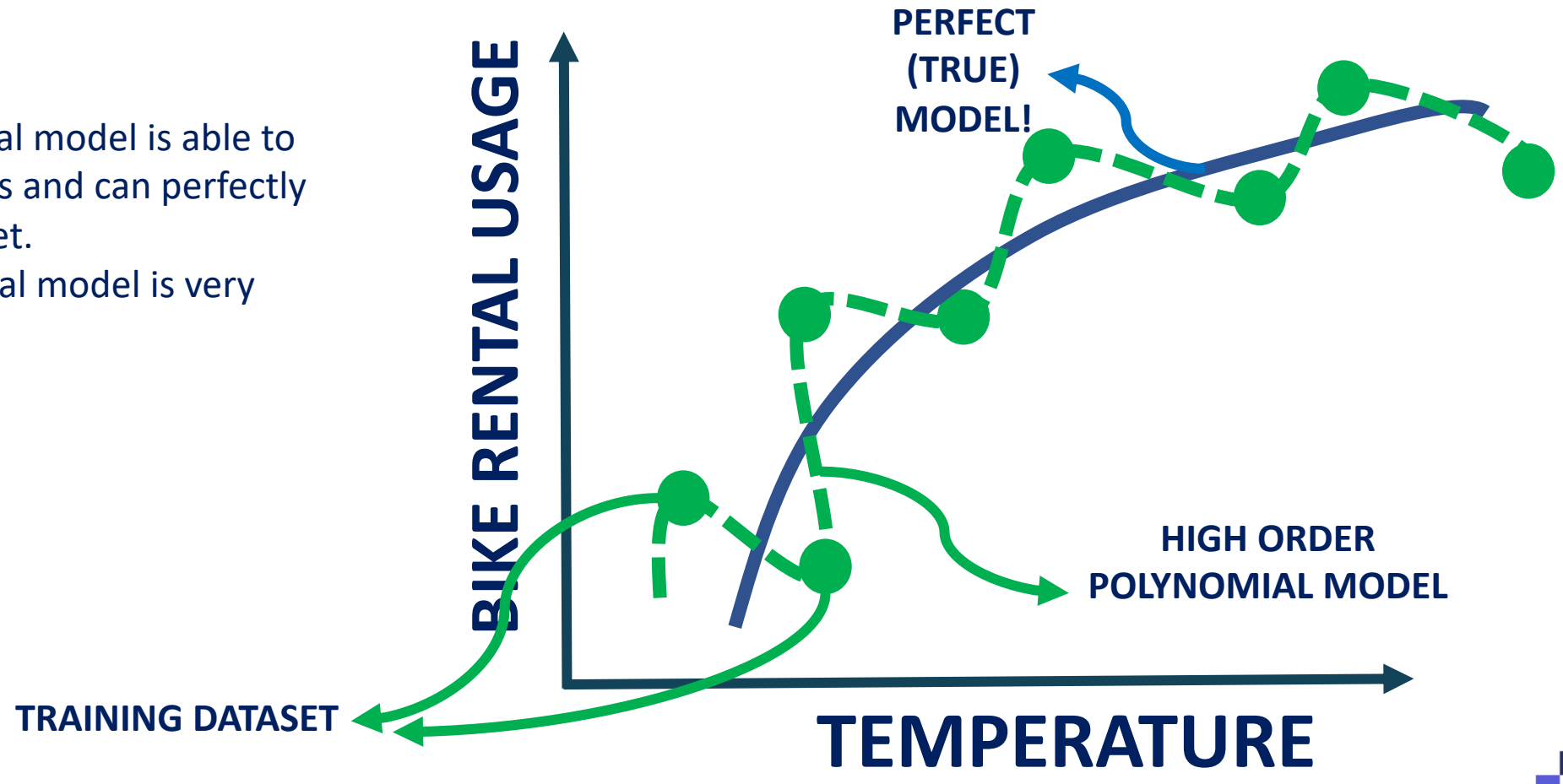
BIAS AND VARIANCE: MODEL #1– LINEAR REGRESSION (SIMPLE)

- Linear Regression model uses a straight line to fit the training dataset
- Linear regression model lacks flexibility so it cannot properly fit the data (as the true perfect model does!)
- The linear model has a large “bias” which indicates that the model is unable to accurately capture the true relationship between temperature and rental usage.

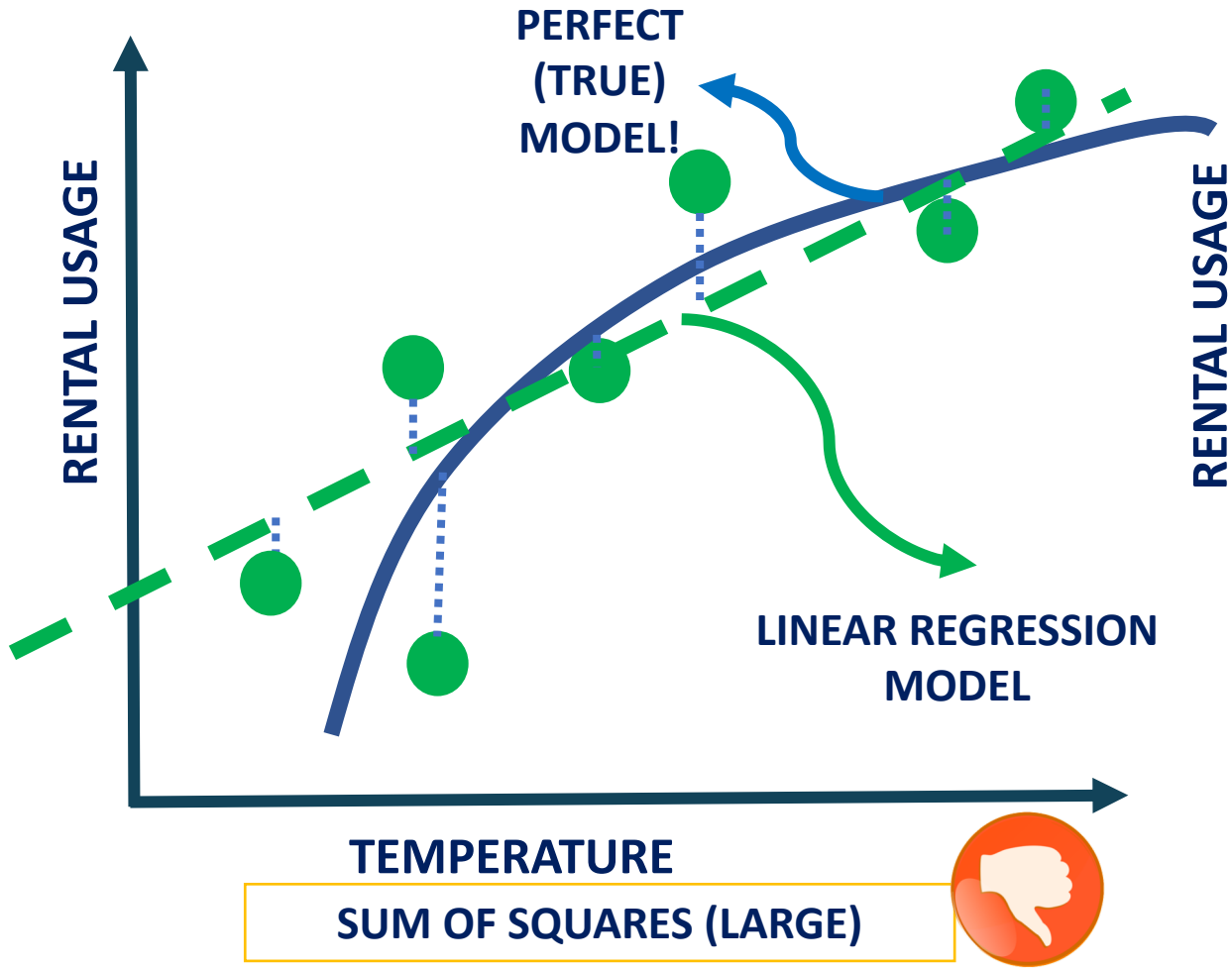


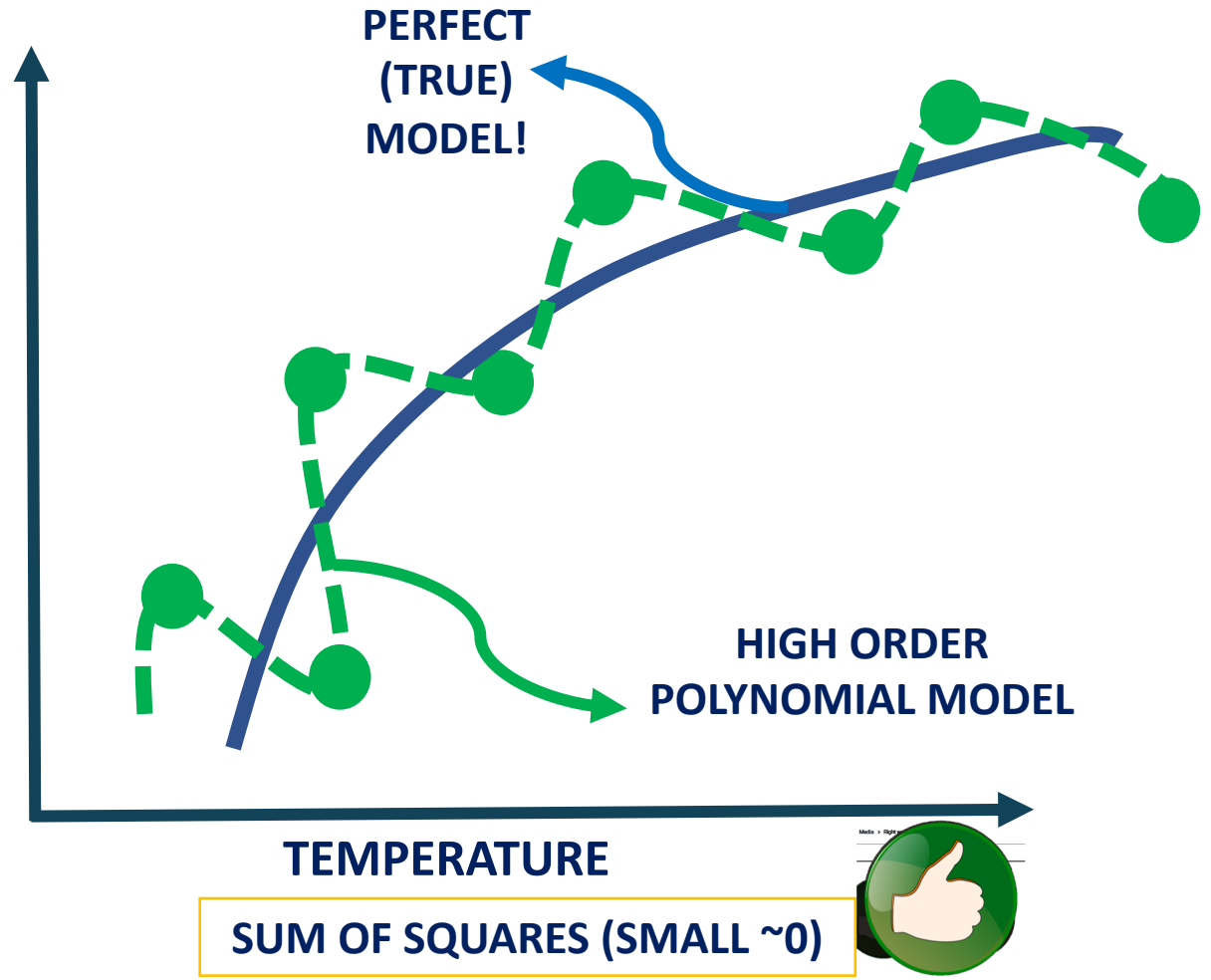
BIAS AND VARIANCE: MODEL #2 – HIGH ORDER POLYNOMIAL REGRESSION (COMPLEX)

- High order polynomial model is able to have a very small bias and can perfectly fit the training dataset.
- High-order polynomial model is very flexible



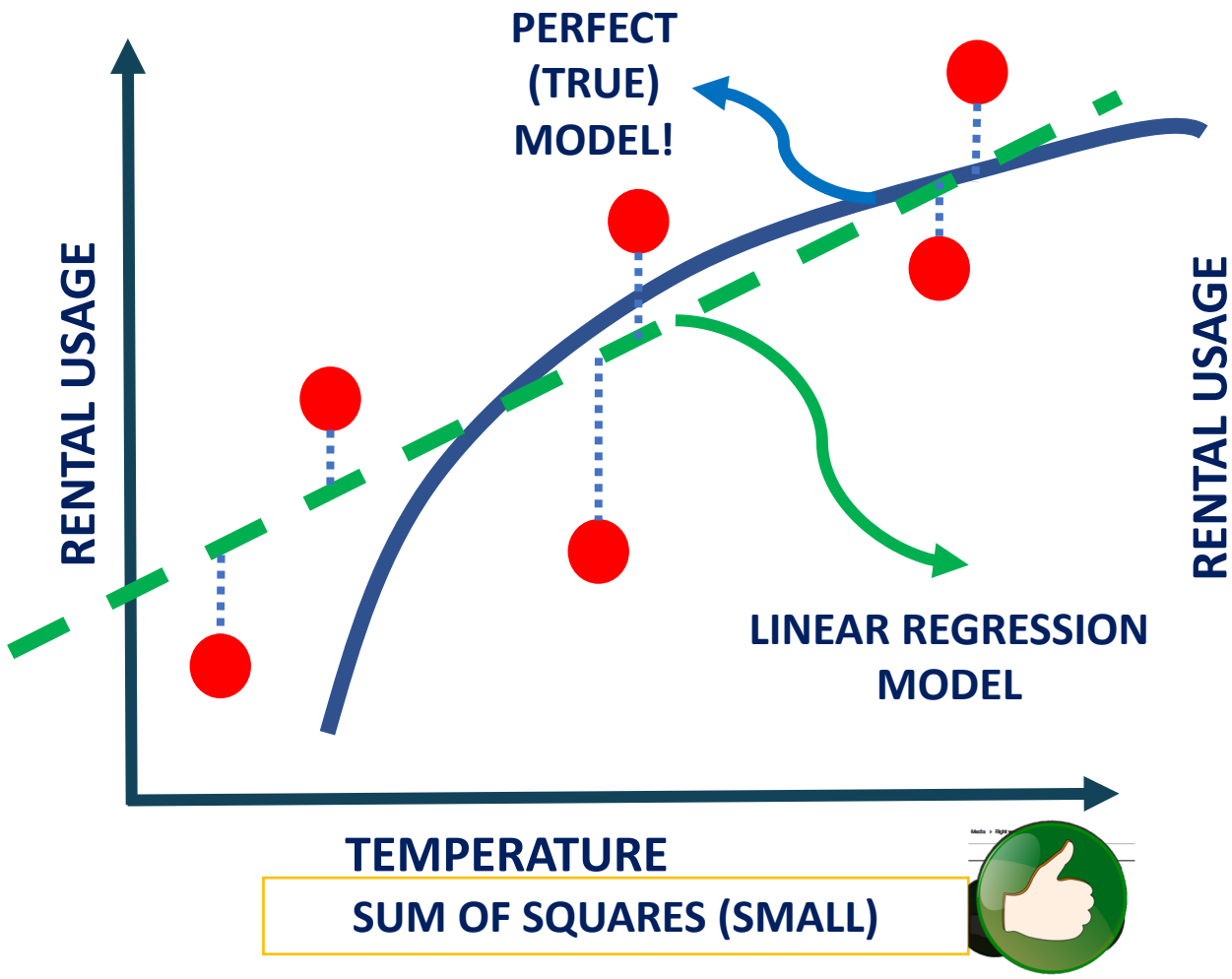
BIAS AND VARIANCE: MODEL #1 Vs. MODEL #2 DURING TRAINING

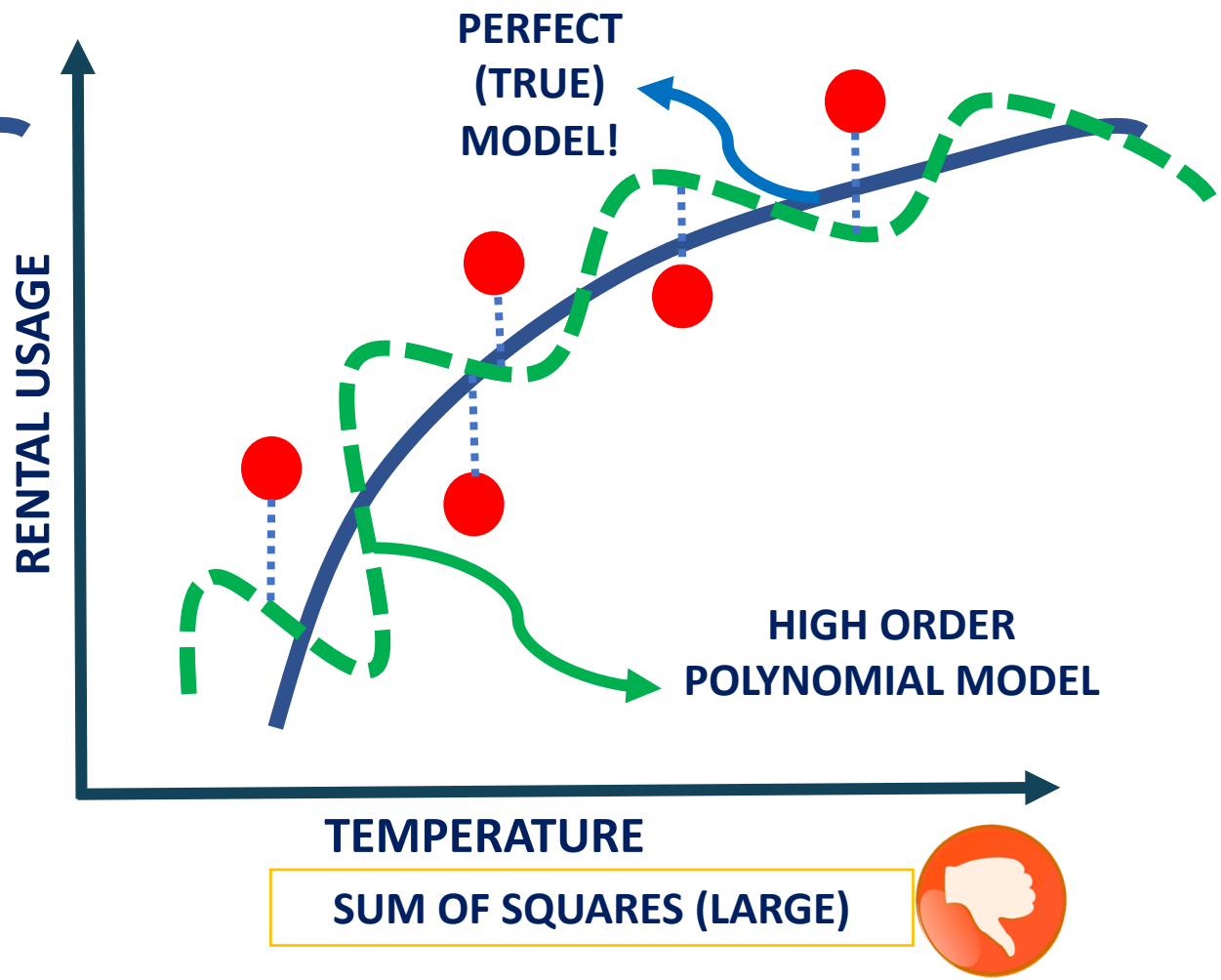




THIS IS NOT THE WHOLE STORY!!

BIAS AND VARIANCE: MODEL #1 Vs. MODEL #2 DURING TESTING

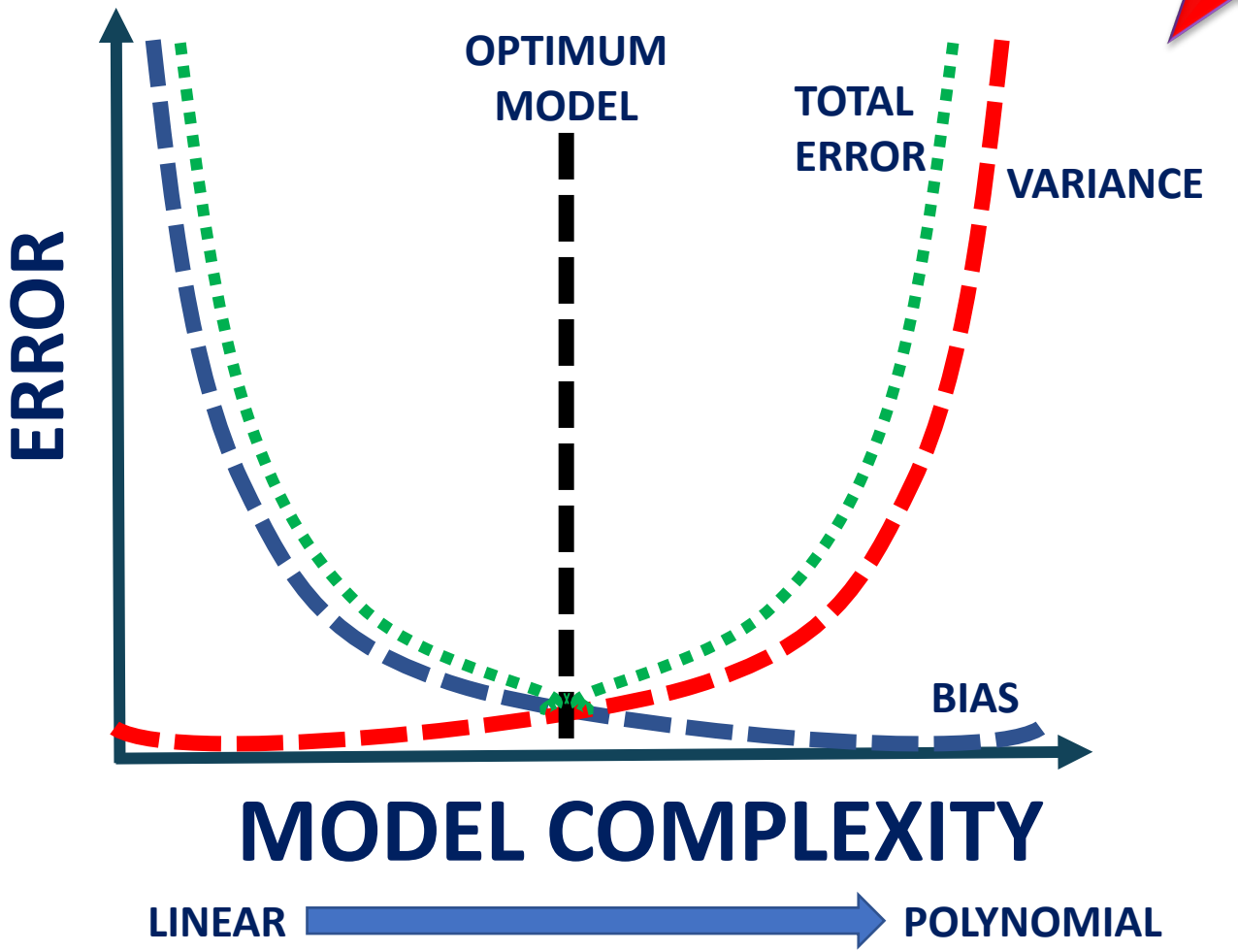




The polynomial model performs poorly on the testing dataset and therefore it has large variance

MODEL COMPLEXITY VS. ERROR

- Regularization works by reducing the variance at the cost of adding some bias to the model.
- A trade-off between variance and bias occurs



MODEL COMPLEXITY VS. ERROR

MODEL #1 (LINEAR REGRESSION) (SIMPLE)	MODEL #2 (HIGH ORDER POLYNOMIAL) (COMPLEX)
Model has **High bias** because it is very rigid (not flexible) and cannot fit the training dataset well	Model has **small bias** because it is flexible and can fit the training dataset very well.
Has **small variance (variability)** because it can fit the training data and the testing data with similar level (the model is able to generalize better) and avoids overfitting	Has **large variance (variability)** because the model over fitted the training dataset and it performs poorly on the testing dataset
Performance is consistent between the training dataset and the testing dataset	Performance varies greatly between the training dataset and the testing dataset (high variability)
Good generalization	Over fitted

- *Variance measures the difference in fits between the training dataset and the testing dataset*
- *If the model generalizes better, the model has small variance which means the model performance is consistent among the training and testing datasets*
- *If the model over fits the training dataset, the model has large variance*

PERFECT REGRESSION MODEL SHALL HAVE SMALL BIAS AND SMALL VARIABILITY!
A TRADEOFF BETWEEN THE BIAS AND VARIANCE SHALL BE PERFORMED FOR ULTIMATE RESULTS

L2 REGULARIZATION (RIDGE REGRESSION)

REGULARIZATION: INTUITION

- Regularization techniques are used to avoid networks overfitting
- ANN overfitting occurs when the network provide great results on the training data but performs poorly on testing dataset.
- Overfitting occurs when the ANN learns all the patterns of the training dataset but fails to generalize.
- Overfitted ANNs generally provide high accuracy on training dataset but low accuracy on testing and validation (evaluation) datasets

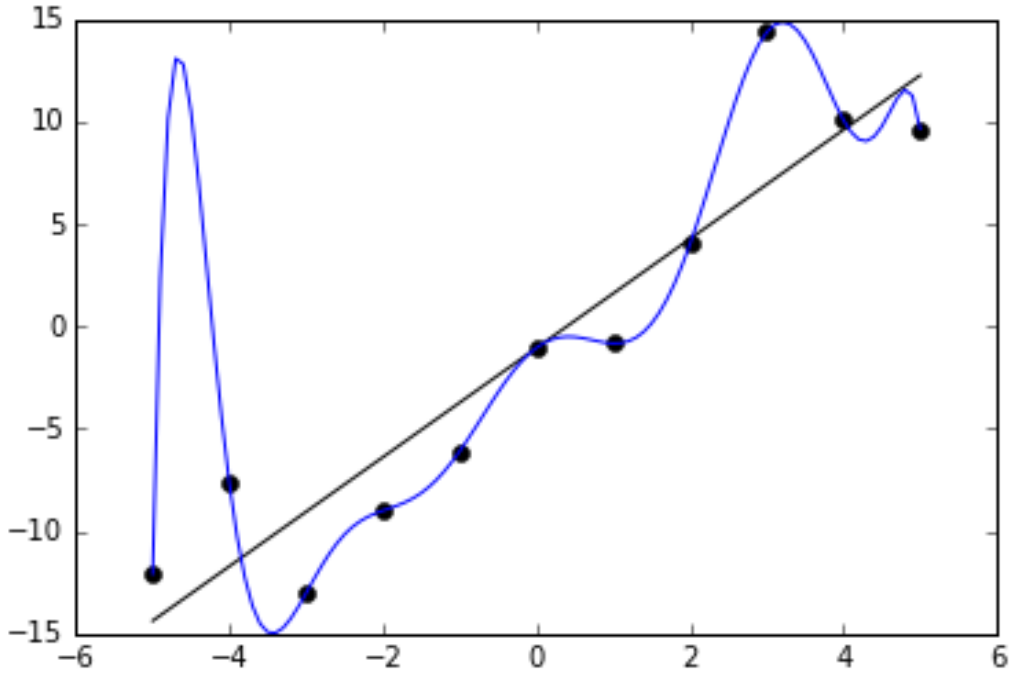
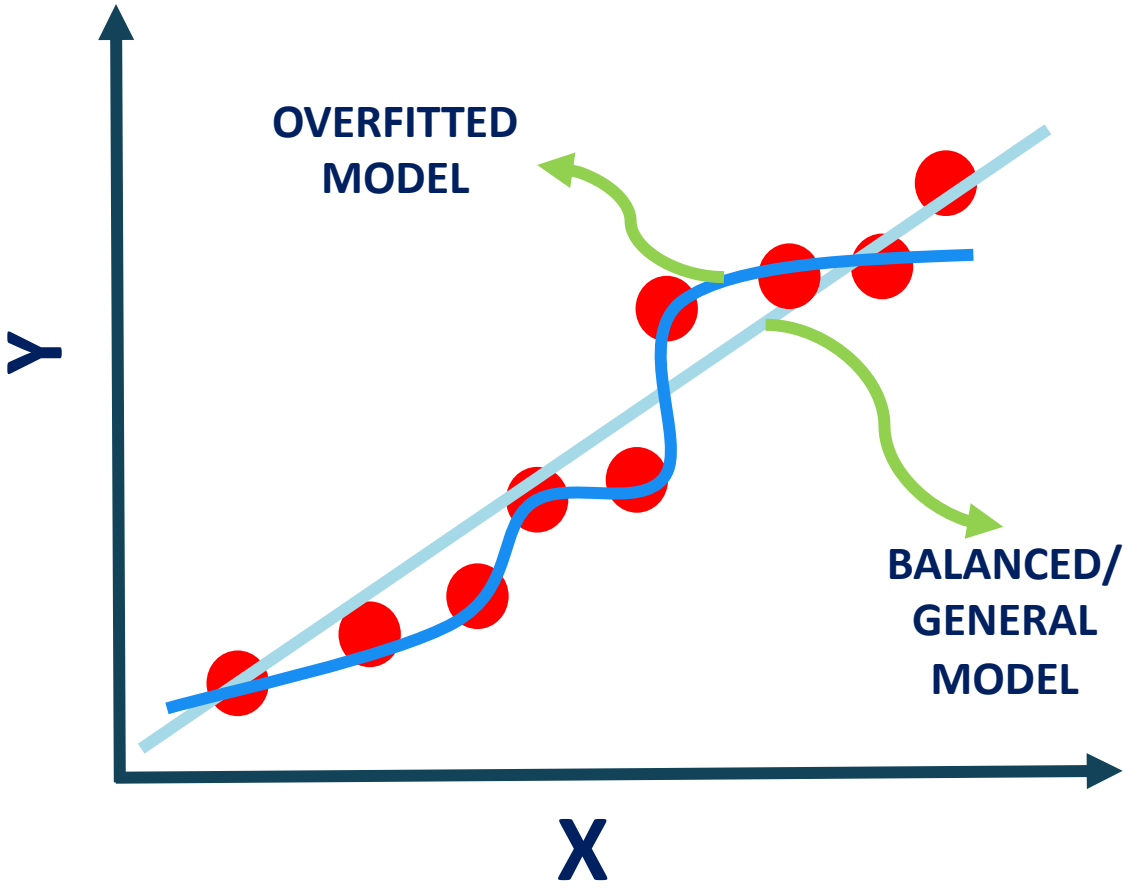


Photo Credit: https://commons.wikimedia.org/wiki/File:Overfitted_Data.png

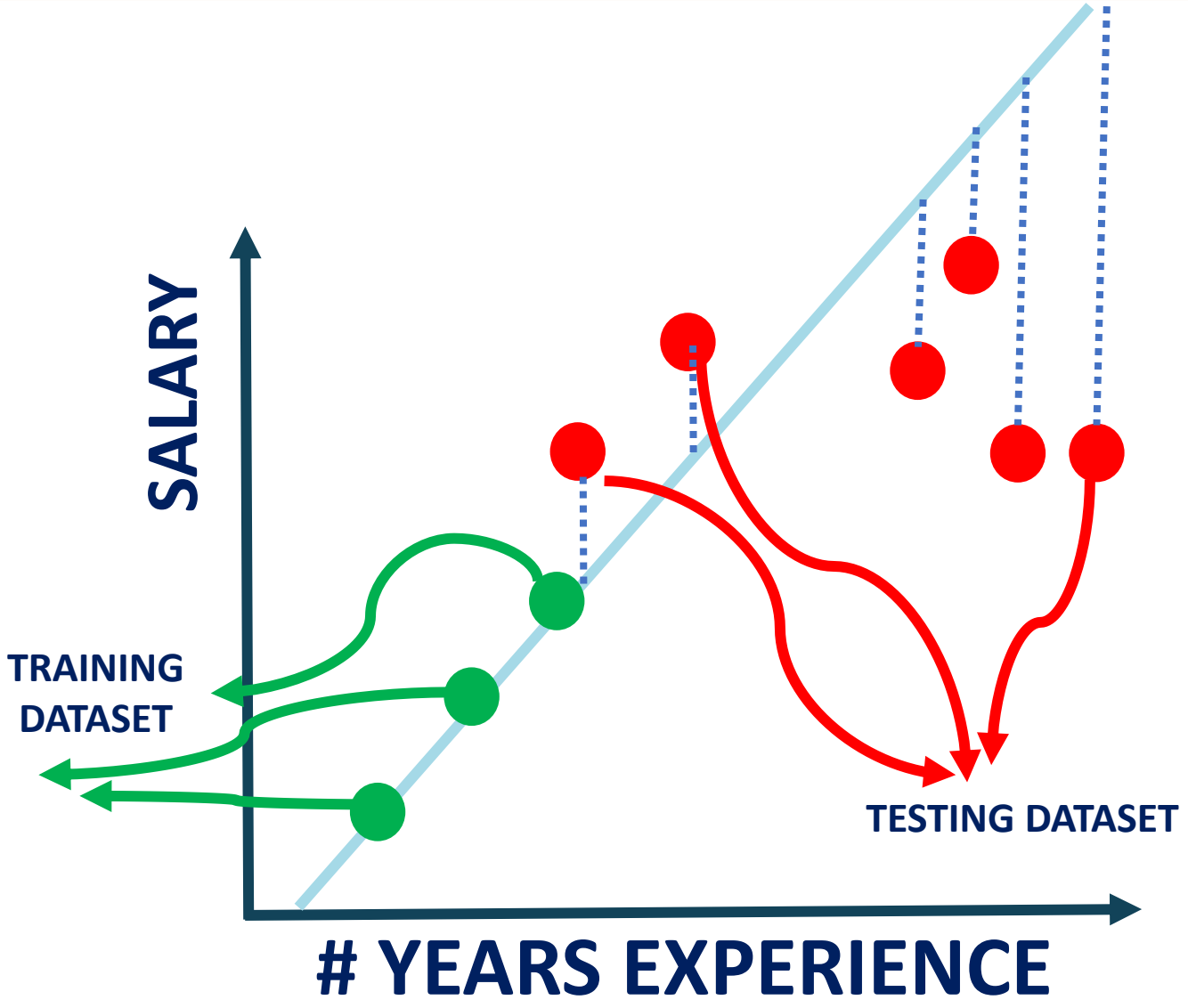
RIDGE REGRESSION (L2 REGULARIZATION): INTUITION

- Ridge regression advantage is to avoid overfitting.
- Our ultimate model is the one that could generalize patterns; i.e.: works best on the training and testing dataset
- Overfitting occurs when the trained model performs well on the training data and performs poorly on the testing datasets
- Ridge regression works by applying a penalizing term (reducing the weights and biases) to overcome overfitting.



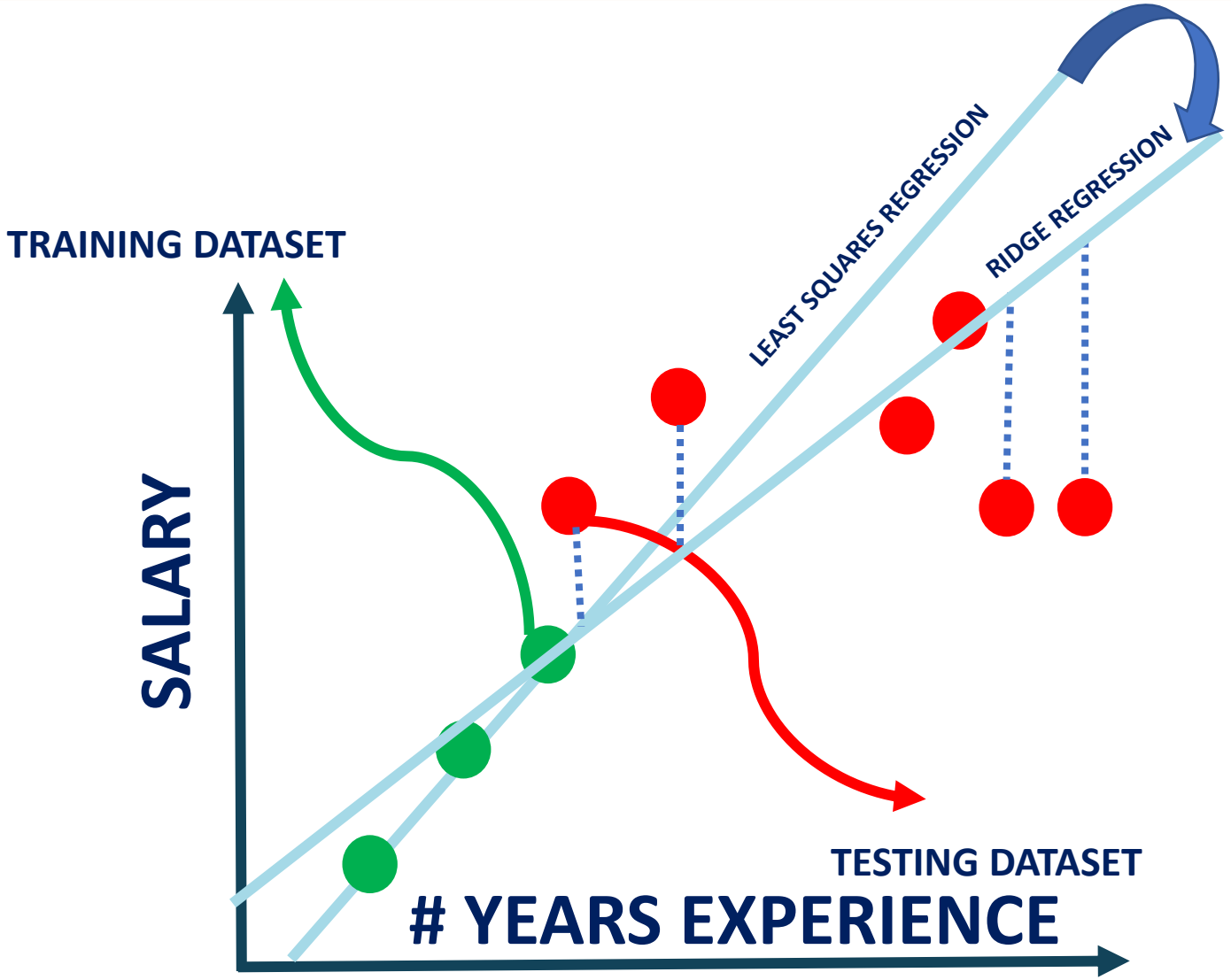
RIDGE REGRESSION (L2 REGULARIZATION): INTUITION

- Least sum of squares is applied to obtain the best fit line
- Since the line passes through the 3 training dataset points, the sum of squared residuals = 0
- However, for the testing dataset, the sum of residuals is large so the line has a high variance.
- Variance means that there is a difference in fit (or variability) between the training dataset and the testing dataset.
- This regression model is overfitting the training dataset



RIDGE REGRESSION (L2 REGULARIZATION): INTUITION

- Ridge regression works by attempting at increasing the bias to improve variance (generalization capability)
- This works by changing the slope of the line
- The model performance might be little poor on the training set but it will perform consistently well on both the training and testing datasets.



RIDGE REGRESSION (L2 REGULARIZATION): MATH

- Slope has been reduced with ridge regression penalty and therefore the model becomes less sensitive to changes in the independent variable (#Years of experience)

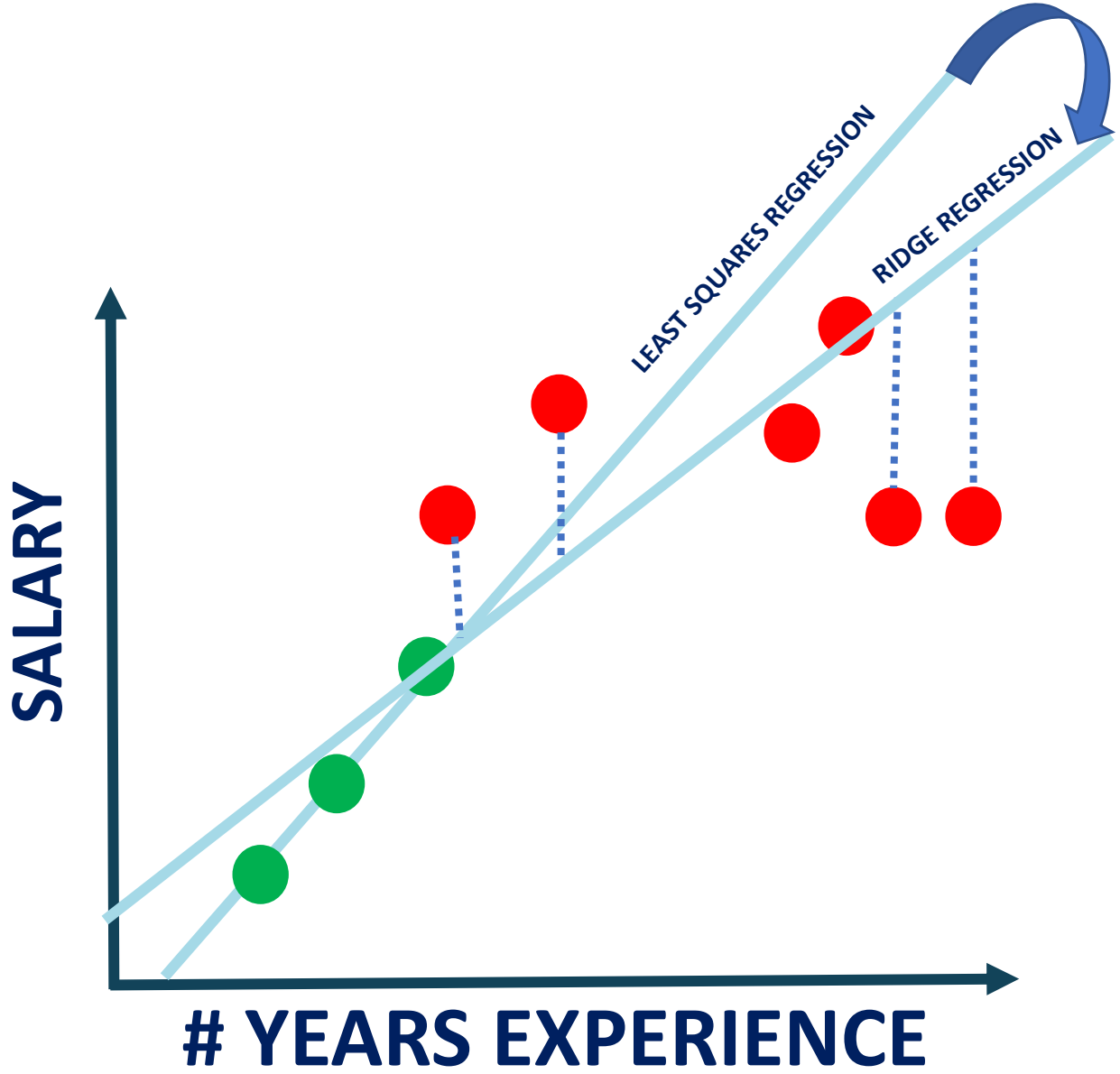
PENALTY TERM

Least Squares Regression:

$$Min(sum\ of\ the\ squared\ residuals)$$

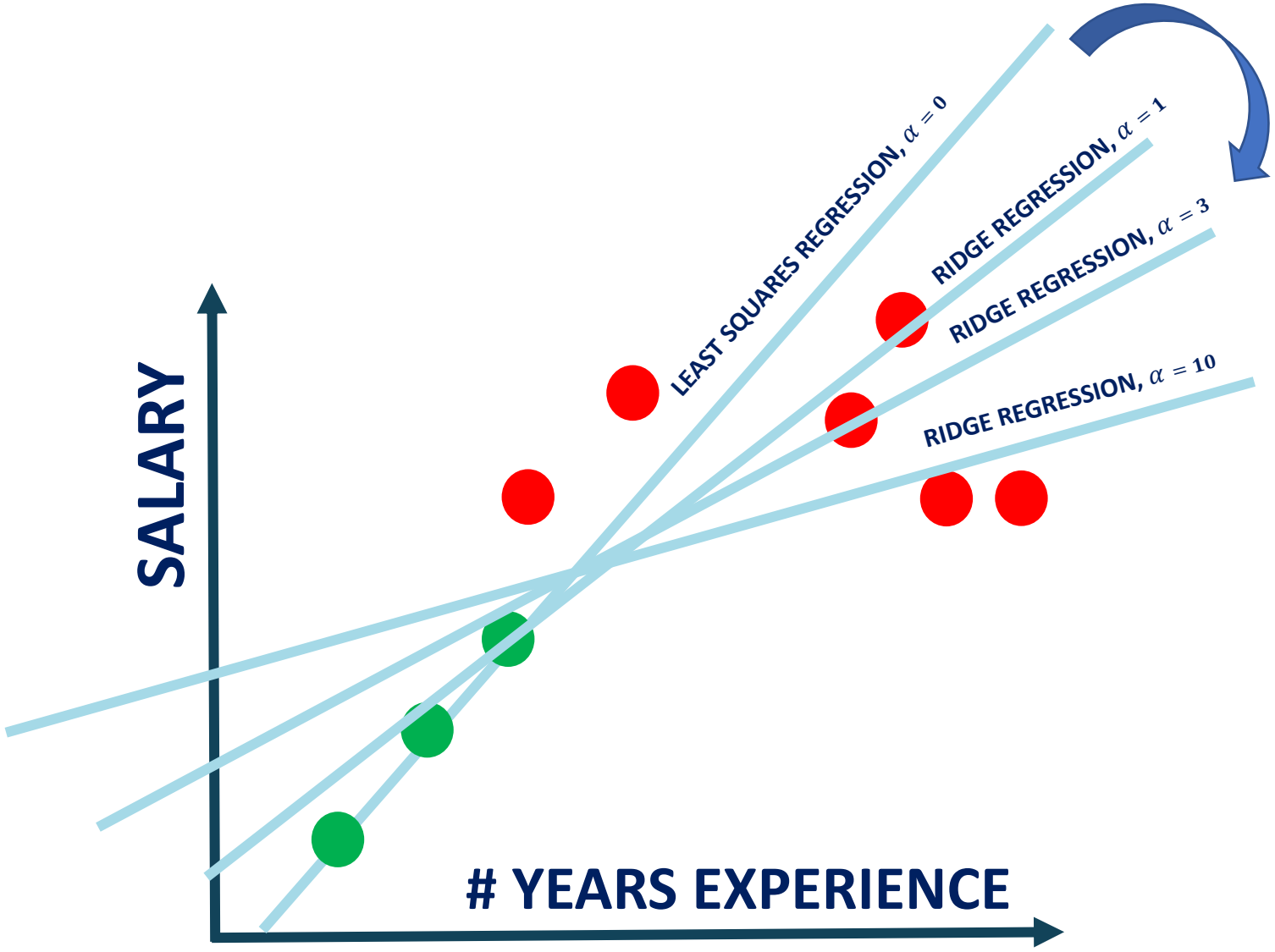
Ridge Regression:

$$Min(sum\ of\ squared\ residuals + \alpha * slope^2)$$



RIDGE REGRESSION (L2 REGULARIZATION): ALPHA EFFECT

- As Alpha increases, the slope of the regression line is reduced and becomes more horizontal.
- As Alpha increases, the model becomes less sensitive to the variations of the independent variable (# Years of experience)



L1 REGULARIZATION
(LASSO REGRESSION)

LASSO REGRESSION (L1 REGULARIZATION): MATH

- Lasso Regression is similar to Ridge regression
- It works by introducing a bias term but instead of squaring the slope, the absolute value of the slope is added as a penalty term

PENALTY TERM

Least Squares Regression:

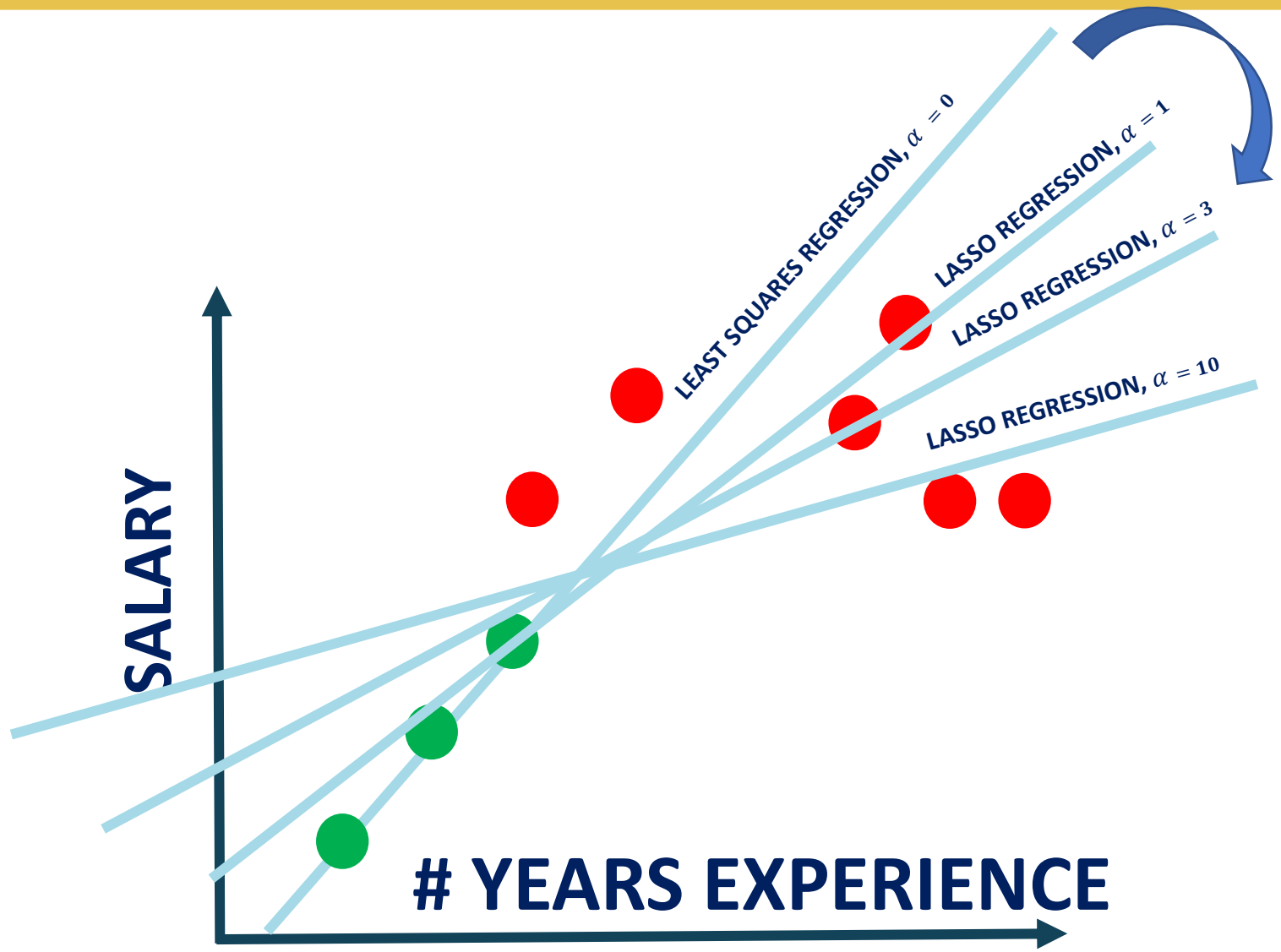
$$Min(sum\ of\ the\ squared\ residuals)$$

Lasso Regression:

$$Min(sum\ of\ squared\ residuals + \alpha * |slope|)$$

LASSO REGRESSION (L1 REGULARIZATION)

- The effect of Alpha on Lasso regression is similar to its effect on ridge regression
- As Alpha increases, the slope of the regression line is reduced and becomes more horizontal.
- As Alpha increases, the model becomes less sensitive to the variations of the independent variable (# Years of experience)



LASSO REGRESSION: MATH

- **Lasso regression (L1 regularization) helps reduce overfitting and it is particularly useful for feature selection**
- **Lasso regression (L1 regularization) can be useful if we have several independent variables that are useless**
- Ridge regression can reduce the slope close to zero (but not exactly zero) but Lasso regression can reduce the slope to be exactly equal to zero.

Least Squares Regression:

$$Min(sum\ of\ the\ squared\ residuals)$$

Ridge Regression (L2 regularization):

$$Min(sum\ of\ squared\ residuals + \alpha * slope^2)$$

Lasso Regression (L1 regularization):

$$Min(sum\ of\ squared\ residuals + \alpha * |slope|)$$

IN SUMMARY

L1 Regularization	L2 regularization
Used to perform feature selection so some features are allowed to go to zero	All features are maintained, but weighted accordingly. No features are allowed to go to zero
Computationally inefficient	Computationally efficient
Sparse output	Dense output

- ***When to choose L1?***
  - *If you believe that some features are not important and you can afford to lose them, then L1 regularization is a good choice.*
  - *The output might become sparse since some features might have been removed.*
- ***When to choose L2?***
  - *If you believe that all features are important and you'd like to keep them but weigh them accordingly.*