Welcome to the CoGrammar Neural Networks II

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: [Feedback on Lectures]()

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

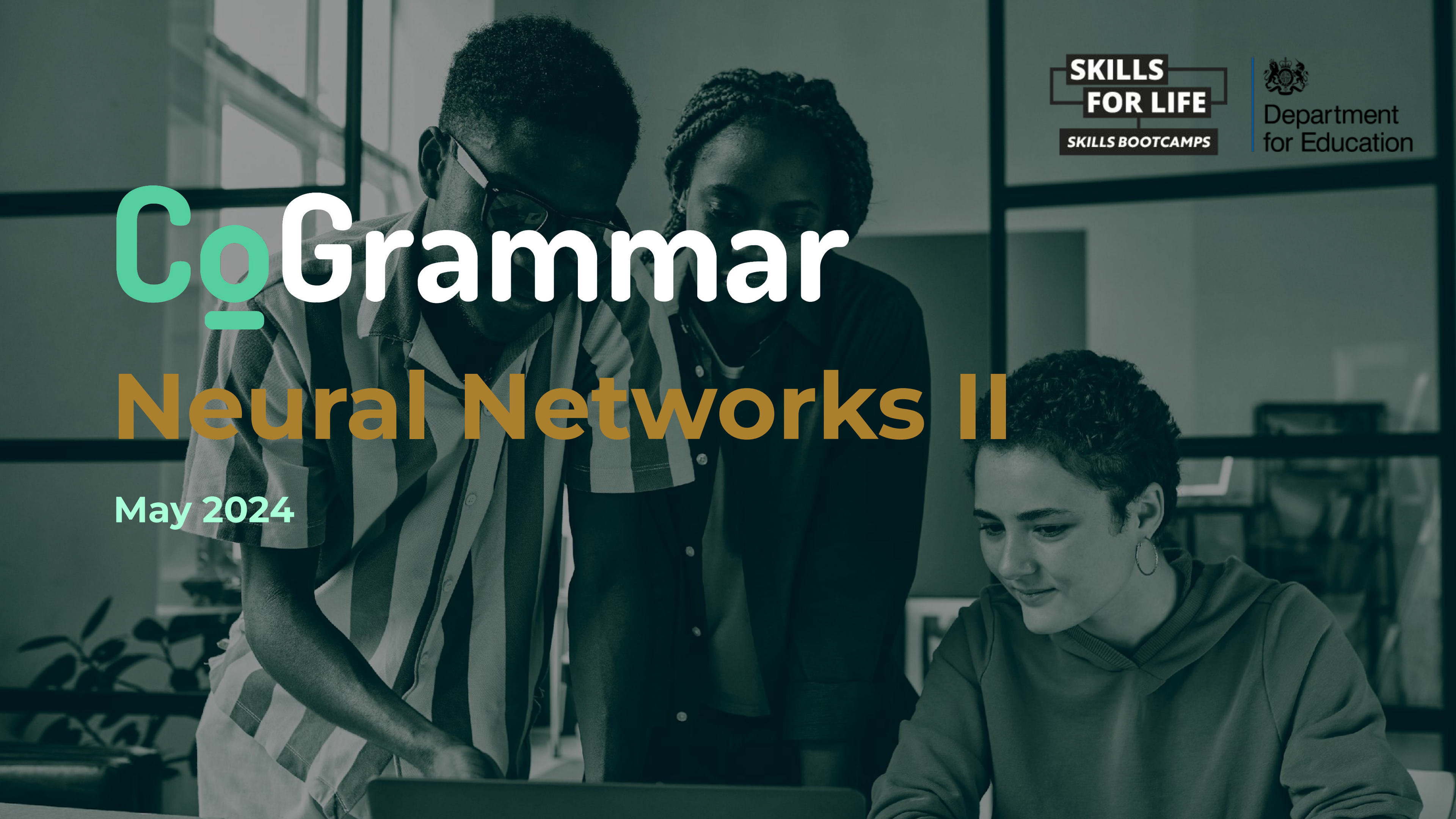
✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar



SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar

Neural Networks II

May 2024

Learning Objectives

- ❖ Understand **backpropagation** in neural networks.
- ❖ Understand the **gradient descent** estimation method.
- ❖ Describe the different **loss functions.**
- ❖ Understand **regularisation** techniques.
- ❖ Implement **Keras** and **TensorFlow** for neural networks.
- ❖ Explore the **limitations** of neural networks.

CoGrammar

Recap of Neural Networks

CoGrammar

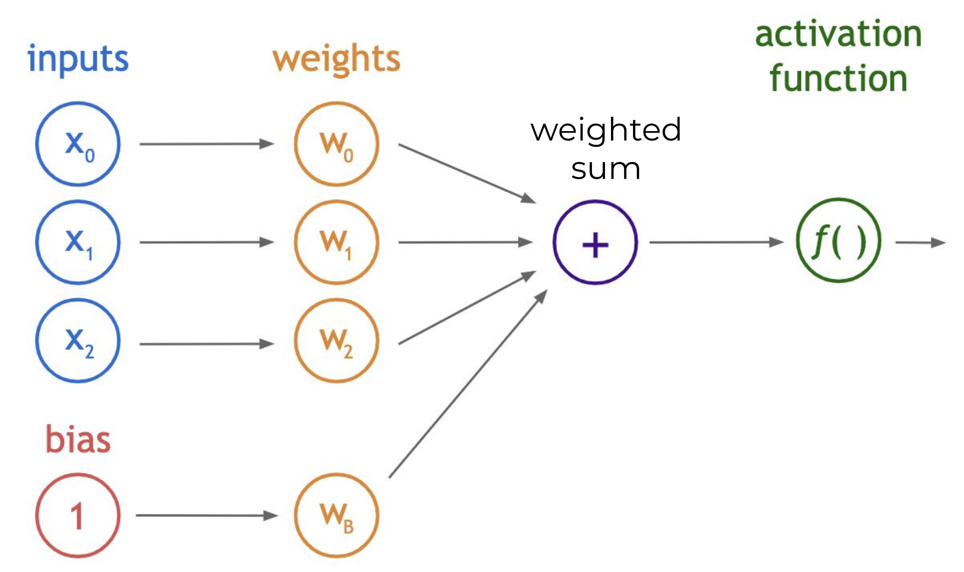
Neural Networks

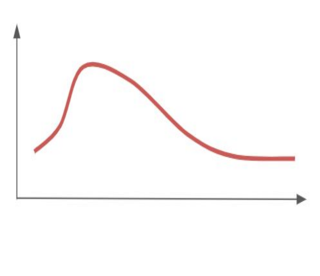
Inputs: set of features that are fed into the model for learning process.

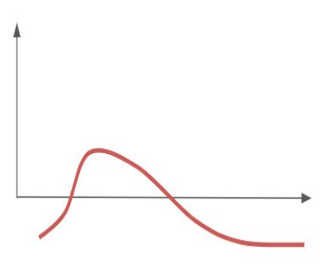
Weights: give importance to those features that contribute more towards learning

Activation function: introduces non-linearity

Bias: shift the value produced by the activation function





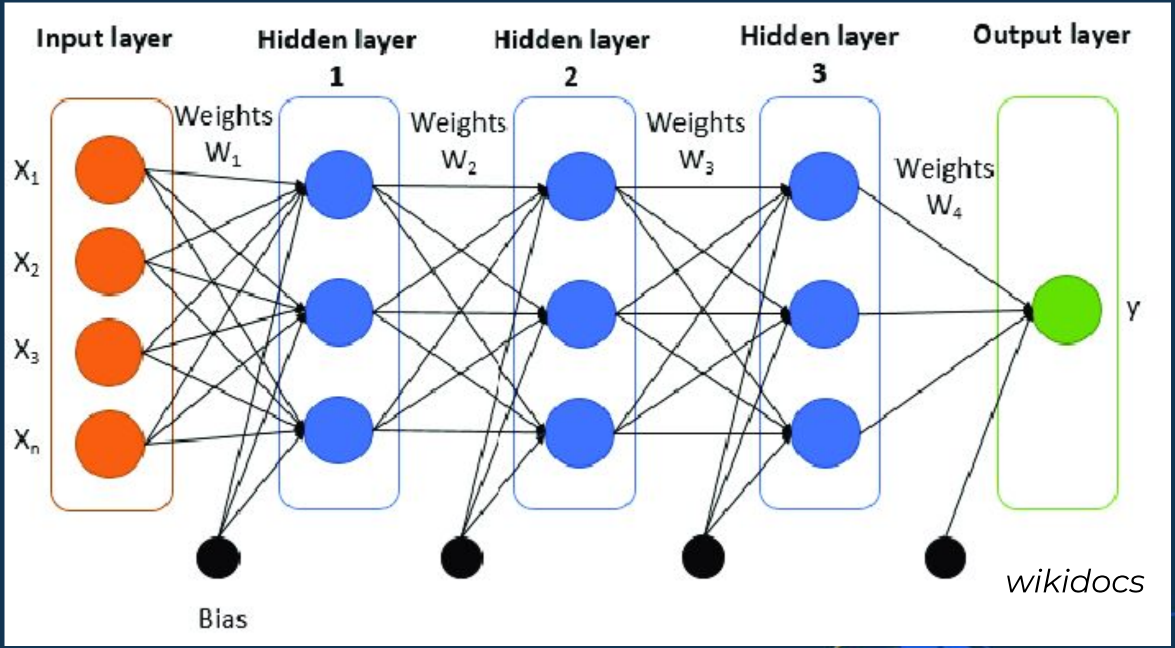


CoGrammar

Neural Networks

Input layer: sends the data to subsequent layers, no. of nodes depend on features

Hidden layer/s: weighted inputs fed into these intermediate layers for computations; extracts features from the data.



Output layer: takes input from preceding hidden layers and comes to a final prediction based on the model's learnings

CoGrammar

Backpropagation in Neural Networks

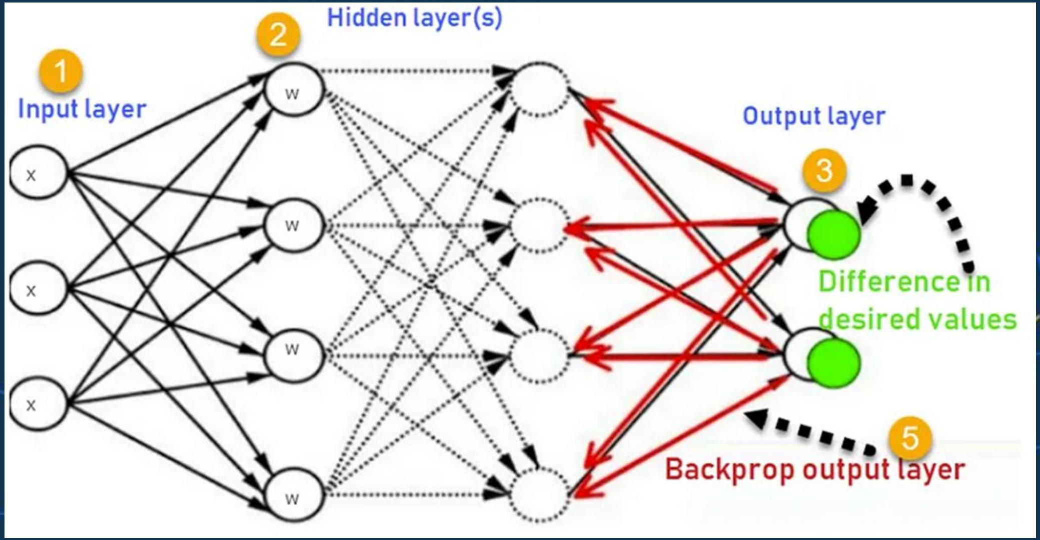
CoGrammar

Backpropagation

- Backpropagation algorithm widely used to train feedforward NNs.
- Computes the **gradient** of the **loss (cost) function** with respect to the network weights layer by layer, and iterating backward from the last layer.
- Efficiently compute the gradient concerning each weight, to train multi-layer networks and **update weights** to **minimise loss.**
- **Gradient descent** or **stochastic gradient descent** estimation method used by the optimisation algorithm to compute the network parameter updates and train neural network models.

Backpropagation

- Backpropagation **minimises the cost function** by adjusting network's **weights** and **biases**.
- The level of adjustment is determined by the **gradients of the cost function** with respect to those parameters.
- Some **calculus** is needed for this! (We will stick with basic notations only)

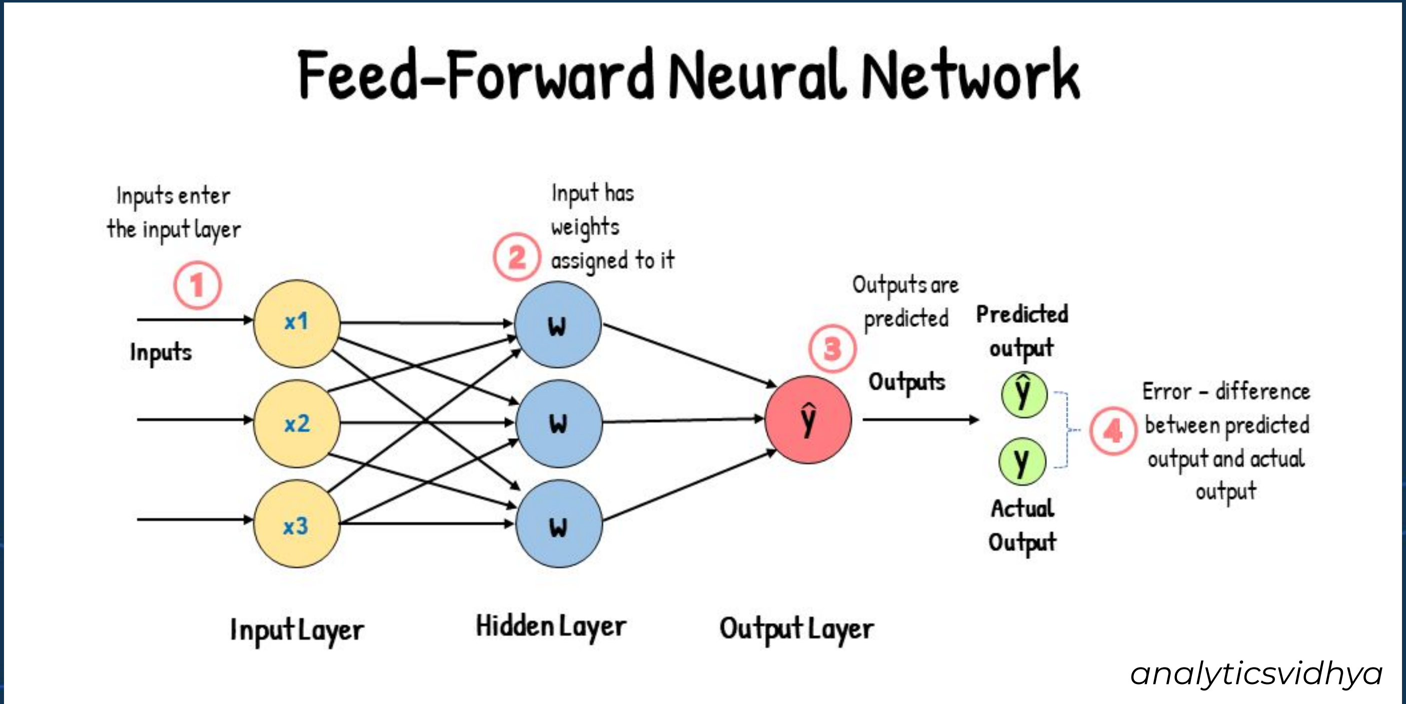


CoGrammar

blog.stackademic

Backpropagation steps

1. Traverse through the network from the input to the output by computing the hidden layers' output and the output layer. [The Feedforward Step]

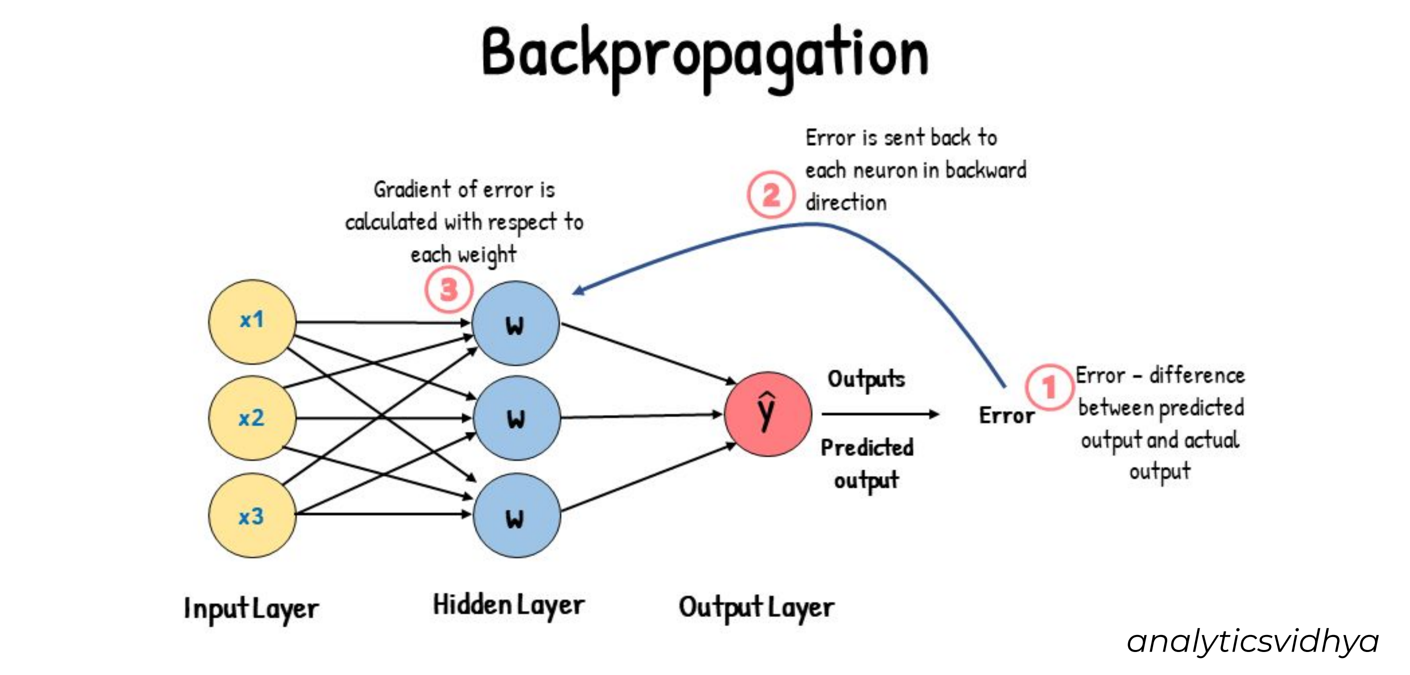


CoGrammar

Backpropagation steps

2. In the output layer, calculate the derivative of the cost function with respect to the input and the hidden layers.

3. Repeatedly update the weights until they converge or the model has undergone enough iterations.



CoGrammar

Gradient Descent Method

CoGrammar

What are gradients?

- Let us take a function **f(x) = y**, which depends on **x** only.
- **Gradient or first derivative of function** **f'(x) = dy / dx** = change in y / change in x i.e. how much the output of a function changes if you change the inputs a little bit.
- In neural networks, let us consider the **cost/loss function y** which depends on the **weights** w_1 **,** w_2 **,** w_3 **, …,** w_n
- **Gradient** / **first derivative or slope of a function** in mathematical terms, measures the change in all weights about the change in error - it is a vector of the partial derivatives of the error with respect to each w_i

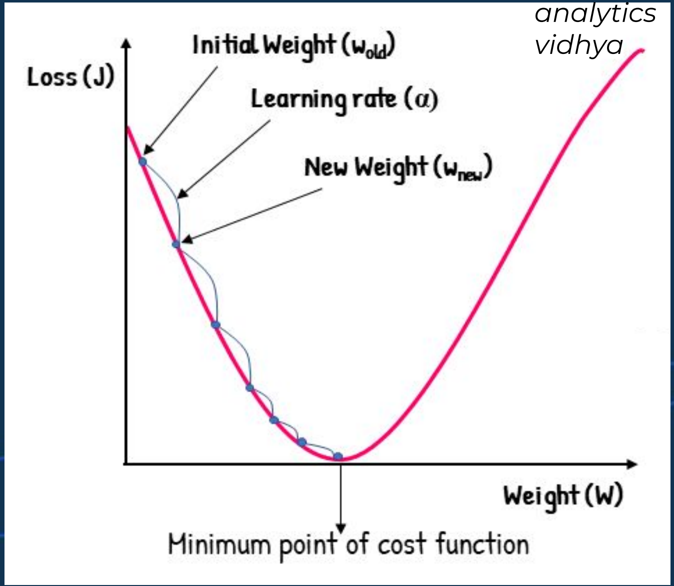
Gradient Descent

Updating the weights

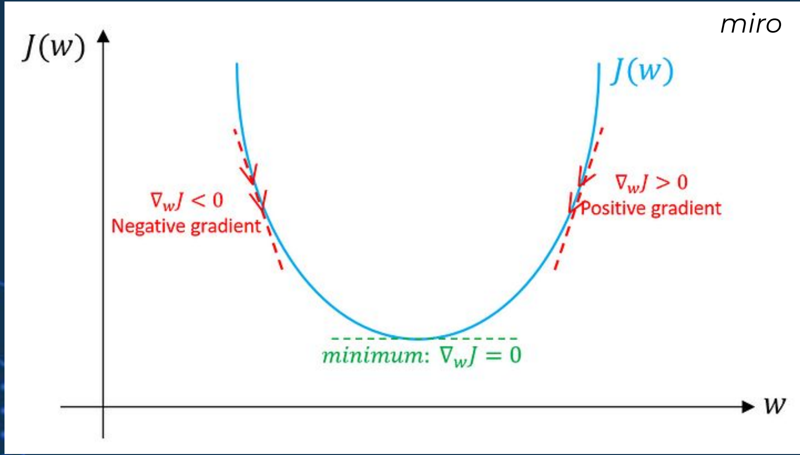
$$w_{new} = w_{old} - \alpha \frac{\delta J}{\delta w}$$

α = **Learning rate**

δJ/δw = **partial derivative of the loss function for each weight w** (rate of change of the loss function to the change in weight.)



Gradient of J = **∇ J** has all the partial derivatives (**∇ = Del / nabla operator)**



CoGrammar

Minimising Loss Functions

- Aim of **gradient descent: minimize the cost function**, or the error between predicted and actual y.

- Requires a **learning rate** and a **cost function** to gradually arrive (in some iterations) at the local or global minimum (i.e. point of convergence).

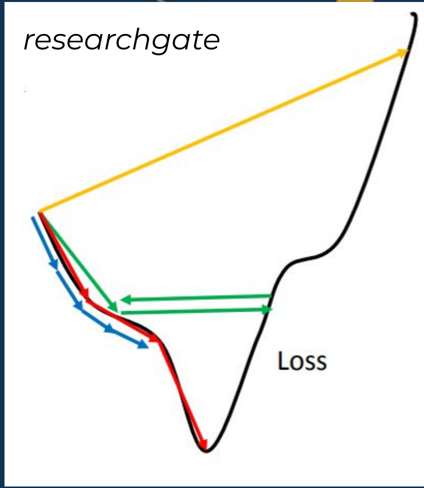
- **Learning rate:** size of steps taken to reach the minimum, usually starts with a small value and updated based on the cost function.

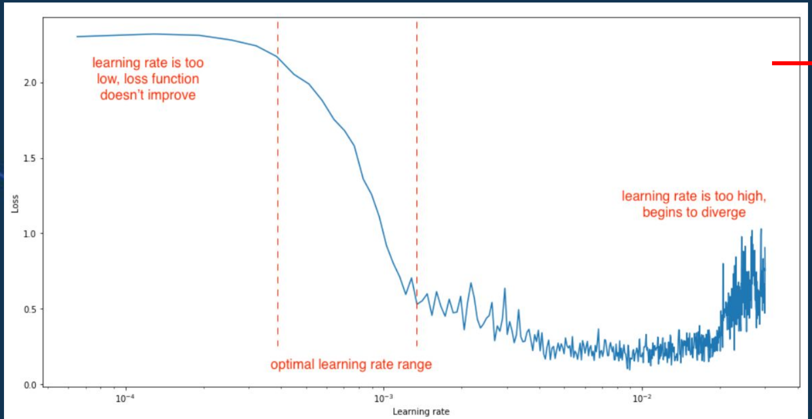
CoGrammar

Learning Rates

High learning rates: result in larger steps but risks overshooting the minimum. (yellow and green lines)

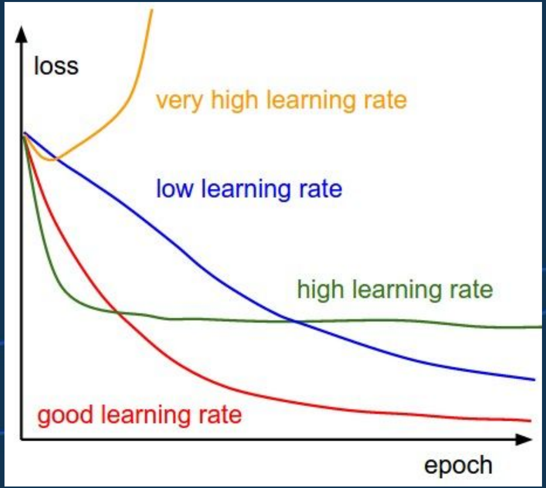
Low learning rate: small step sizes, more precision, but compromises overall efficiency, more iterations takes more time and computations to reach the minimum (blue lines)





jeremyjordan.me

Set learning rate bounds to observe all three phases, making the optimal range trivial to identify.



CoGrammar

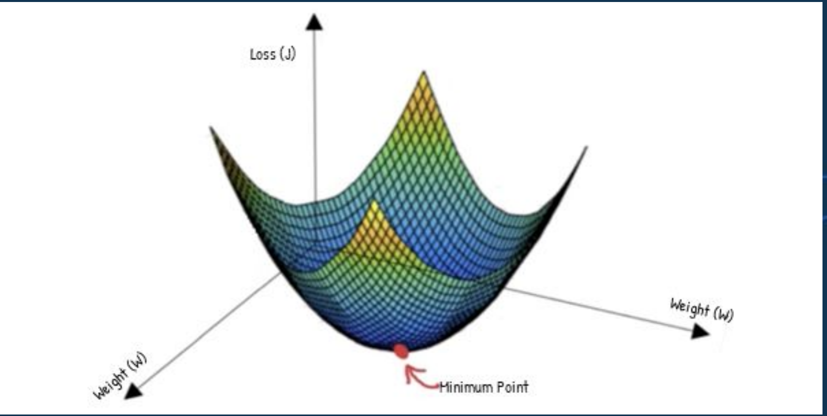
Minimising Loss Functions

- **Cost (or loss) function** measures the difference, or error, between actual y and predicted y at its current position.

- Improves machine learning model's efficacy by providing feedback to the model so that it can adjust the parameters to **minimise the error** and **find the local or global minimum**.

- Continuously iterates, moving along direction of steepest descent (or the negative gradient) until the cost function is close to or at zero.

- At this point, the model will stop learning.



CoGrammar

Gradient Descent Types

- **Batch gradient descent (BGD):** involves calculations over the full training set at each step as a result of which it is very slow on very large training data. Thus, it becomes **very computationally expensive** to do Batch GD.

- **Stochastic gradient descent (SGD):** computes the gradient using only a single training example or a small subset of examples in each iteration. (easier to allocate in desired memory, **robust**, more efficient for large datasets)

- **Mini-Batch gradient descent:** split the training dataset into small batches that are used to calculate model error and updated model coefficients. (the most common implementation, easier to fit in allocated memory, computationally efficient, produces stable gradient descent convergence.)

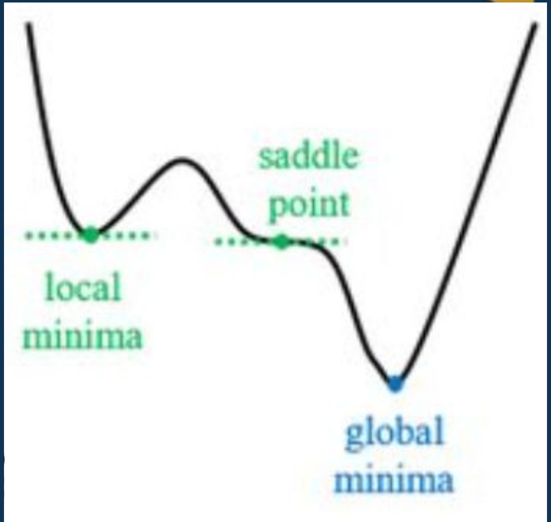
Gradient Descent Issues

- Gradient descent can converge to **local minima** or a **saddle point (plateau)** instead of the global minima, especially if the cost function has multiple peaks and valleys.

- For **high learning rate**, algorithm may overshoot the minimum; if it is **too low,** it may take too long to converge.

- Can **overfit** the training data if the model is too complex or the learning rate is too high (use regularisation to prevent)

- **Slow convergence** for larger datasets.



Overcome by adaptive learning rate (Adam), or momentum based methods, or second-order methods. Choose right regularisation method, and hyperparameters.

CoGrammar

Different Loss Functions

CoGrammar

Loss Function

- A **loss function or cost function** measures how good a neural network model is in performing a regression or classification task.

- Minimise the loss function value during the backpropagation step to make the neural network perform better.

- **Loss function:** Used when we refer to the error for a single training example.

- **Cost function:** Used to refer to an average of the loss functions over an entire training data.

CoGrammar

Cross-Entropy Loss Function

- **Entropy** = degree of randomness or disorder within a system, measures the uncertainty of an event.
- **Cross-Entropy Loss** is also called **logarithmic loss** or **log-loss,** used in **classification** tasks to predict probabilities.
- Measures the **error** between a **predicted probability** and the label which represents the **actual class.**
- **Binary cross-entropy** is used when performing binary classification, and **categorical cross-entropy** is used for multi-class classification.

Cross-Entropy Loss

$$\mathcal{L}(\theta) = -\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i)$$

θ = weight vector
y = true labels
$\hat{y}$ = predictions

Mean Square Error (MSE) Loss Function

- **MSE loss function** used for **regression** tasks, when we want the network to predict continuous numbers.

- MSE measures the **average squared difference** between the **predicted** and **actual** values.

$$\text{MSE} = \frac{1}{N}\sum_{i=0}^{N}(y_i - \hat{y}_i)^2$$

y = true values

$\hat{y}$ = predictions

CoGrammar

Mean Absolute Percentage Error (MAPE) Loss Function

- **MAPE loss function** used during **demand forecasting** to check network performance during training time, **regression** tasks.

- **Demand forecasting: predictive analytics** dedicated to predicting the expected demand for an item or service in the near future.

- E.g: travel agents looking at optimal prices for hotels, flights, which destinations should be spotlighted or what types of packages should be advertised.

$$MAPE = \frac{100\%}{N} \sum_{i=0}^{N} \frac{|y_i - \hat{y}_i|}{\hat{y}_i}$$

y = true values

$\hat{y}$ = predictions

CoGrammar

MSE vs MAPE

Let us look at two cases:

- **Case 1:** true= 1000, predicted= 1010
- **Case 2:** true= 5, predicted= 15

Case 1: MSE = $(1000 - 1010)^2$ = **100**

MAPE = 100 * | (1000 - 1010) | / 1010 ~ **1 %**

Case 2: MSE = $(5 - 15)^2$ = **100**

MAPE = 100 * | (5 - 15) | / 15 ~ **67 %**

Same **MSE** indicates that both model performances are same.

However, **MAPE** shows that the **performance of these two models is very different**

MAPE loss function judges the two model performances better than MSE for demand forecasting.

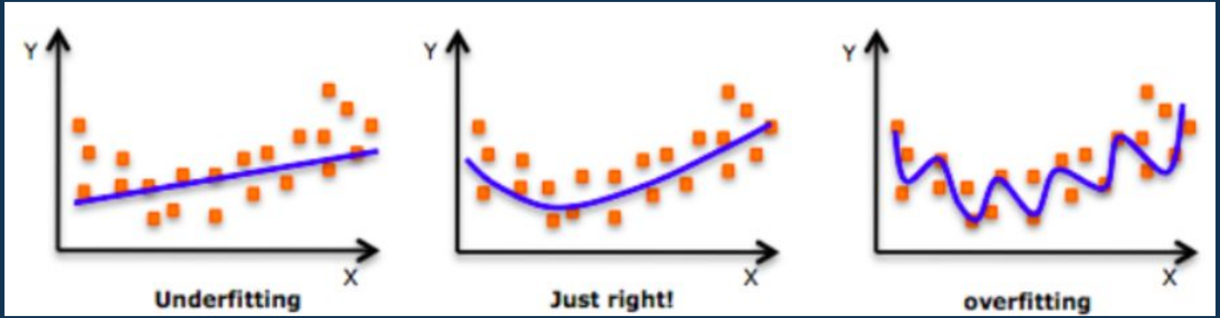
CoGrammar

Regularisation Techniques

CoGrammar

Regularisation Techniques

Regularisation techniques improve neural network's generalisation ability by reducing overfitting.

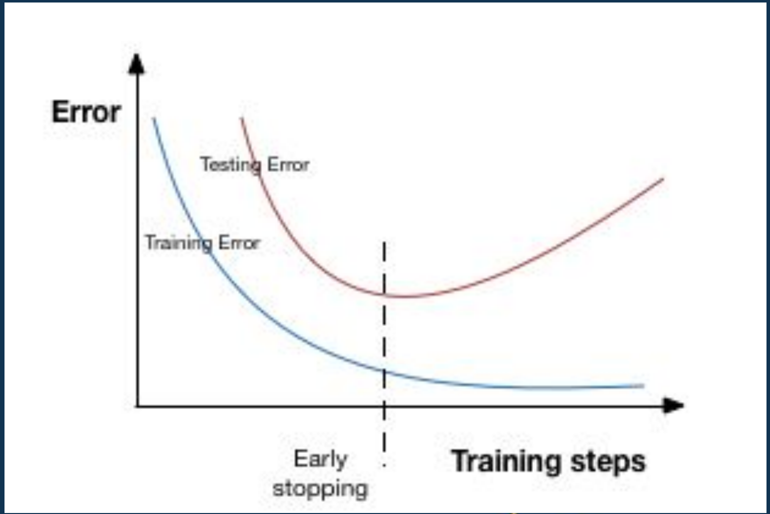


- Done by minimising complexity, exposing network to more diverse data.
- Can add a penalty term to the loss function during training which discourages the model from becoming too complex or having large parameter values.
- Models become more robust and better at making accurate predictions on unseen data.

CoGrammar

Early Stopping

- Training error becomes too low and reaches arbitrarily close to zero, then the network is sure to overfit on the training dataset.



- A high variance model performs badly on test data that it has never seen.

- **Early stopping:** Prevent training loss from becoming arbitrarily low, model is less likely to overfit on training dataset, and will generalise better.

CoGrammar

Data Augmentation

- **Increase the training data sample** or **more diverse training examples** to prevent overfitting.

- **Data augmentation:** e.g. applies transformations to **images** (rotating the image, flipping, scaling, shifting) to create a larger dataset.

- Apply any label-invariant transformation.
  - Color space transformations such as change of pixel intensities
  - Rotation and mirroring
  - Noise injection, distortion, and blurring

Ex: handwritten digits dataset

Mixup, Cutout, CutMix, and AugMix

CoGrammar

Data Augmentation

Audio Data Augmentation

- **Noise injection:** add gaussian or random noise to the audio dataset..
- **Shifting:** shift audio left (fast forward) or right with random seconds.
- **Changing the speed:** stretches times series by a fixed rate.
- **Changing the pitch:** randomly change the pitch of the audio.

Text Data Augmentation

- **Word/sentence shuffling:** randomly changing word position or sentence.
- **Word replacement:** replace words with synonyms.
- **Syntax-tree manipulation:** paraphrase the sentence using the same word.
- **Random word insertion and deletion**

CoGrammar

Advanced Techniques: Generative adversarial networks (GANs) and Neural Style Transfer

L1 and L2 regularisation

- Most common types of regularisation - update the general cost function by the regularisation term.

Cost function = Loss + L1 or L2 Regularisation term

- Values of weight matrices decrease as a neural network with smaller weight matrices leads to simpler models, reducing overfitting.

L1: Regularisation parameter λ and absolute value of weights. (robust to outliers, feature selection for highly correlated features)

$$\lambda \sum_{i=1}^{N} |\theta_i|$$

L2: sum of squares of all of the feature weights (computationally cheaper)

$$\lambda \sum_{i=1}^{N} \theta_i^2$$

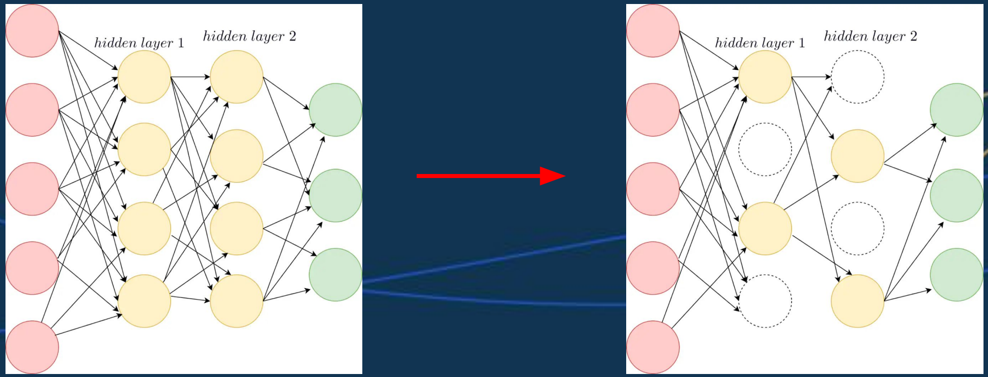
CoGrammar

Dropout

- **Dropout** involves removing random nodes at each iteration.

- Each neuron is assigned a “dropout”probability, e.g. 0.5 i.e. 50% chance of each neuron participating in training within each training batch.

- **Emulate ensemble technique:** slightly different network architecture for each batch, equivalent to training different neural networks on different subsets of the training data, performs better.



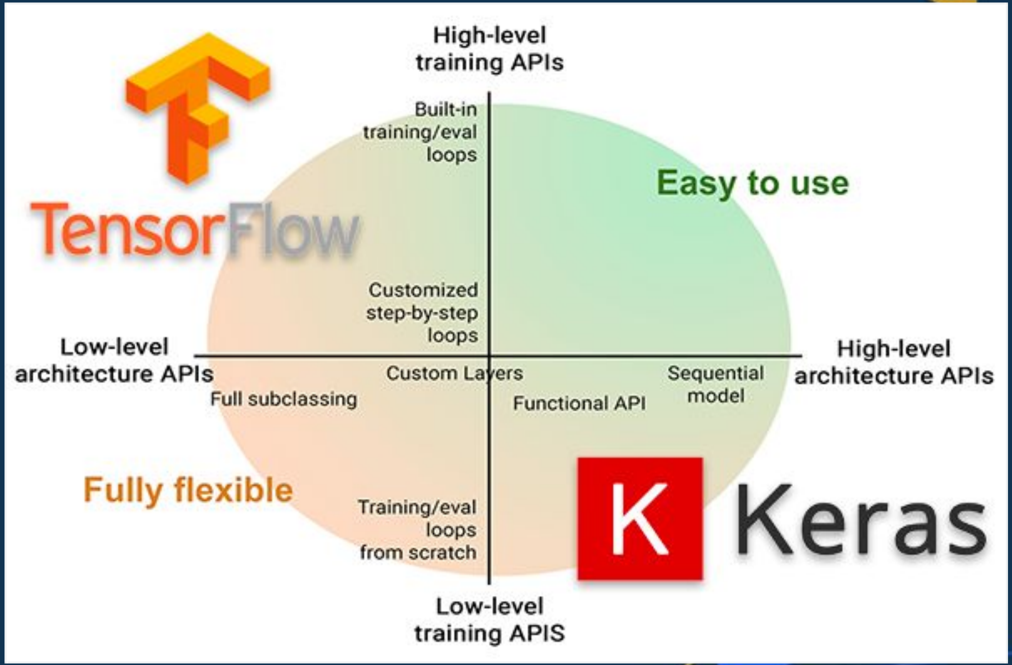
CoGrammar

Implementing Keras and TensorFlow

CoGrammar

Keras and TensorFlow

- **Keras** is a Python library including an API for working with neural networks and deep learning frameworks.
- **Tensorflow** (previously most widely used) is complex, serves as backend for Keras*.
- In Keras, creating, training and tracking a neural network is straightforward.



*Other backends: Theano or Cognitive Toolkit (CNTK)

```
pip install tensorflow
```

```
pip install keras
```

CoGrammar

Keras Architecture

1. Model

Sequential Model: a linear composition of Keras Layers, easy, minimal and has the ability to represent nearly all available neural networks.

Functional API : used to create complex models.

Importing Sequential model from Keras models
Create a sequential model

```
from keras.models import Sequential

model = Sequential()
```

CoGrammar

Keras Architecture

2. Layer

Keras provides pre-build layers (input, hidden and output) so that any complex neural network can be easily created.

Examples: Core, Convolution, Pooling, and Recurrent Layers

3. Core Modules

Activations module (softmax, relu), **Loss module** (mean_squared_error, mean_absolute_error, poisson), **Optimizer module** (adam, sgd), **Regulariser module** (L1 regularizer, L2 regularizer)

CoGrammar

Keras Architecture

Importing **Dense layer, Activation module**, and **Dropout layer** to handle overfitting

Adds a **dense layer** (Dense API) with **relu activation** (using Activation module) function.
Adds a **dropout layer** (Dropout API) to handle over-fitting.
Adds final dense layer (Dense API) with **softmax activation** (using Activation module) function.

```
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout

model = Sequential()
model.add(Dense(512, activation = 'relu', input_shape = (784,)))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation = 'softmax'))
```

CoGrammar

MLPClassifier vs TensorFlow and Keras

- In the associated Notebooks, we will demonstrate image classification using MLPClassifier from scikit-learn and also using TensorFlow and Keras.

- The Keras applications module is used to deploy deep neural network models that have already been trained. Keras models are used for feature extraction, prediction, and fine-tuning.

CoGrammar

Summary: Advantages and Limitations

CoGrammar

Advantages of Neural Networks

- **Adaptability:** useful for activities where link between inputs and outputs is complex or not well defined.

- **Pattern Recognition:** efficacious in tasks like audio and image identification, natural language processing, intricate data patterns.

- **Parallel Processing:** can process numerous jobs at once, which speeds up and improves the efficiency of computations.

- **Non-Linearity:** model and comprehend complicated relationships in data by virtue of the non-linear activation functions found in neurons.

Limitations of Neural Networks

- **Black boxes:** hard to understand how they process the input data and what features they learn and use; pose problems for applications that require transparency, accountability, and trust, such as healthcare, finance, or law.
- **Vulnerable to subtle perturbations** or modifications of the input data, which can cause them to produce incorrect or misleading outputs.
- **Tend to overfit the data**: limit ability to adapt to changing or diverse environments or scenarios.
- **Extensive data and computation requirements**: affect feasibility and scalability

References

- https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/
- https://wikidocs.net/165315
- https://cs231n.github.io/neural-networks-3/?ref=jeremyjordan.me#annealing-the-learning-rate
- https://www.analyticsvidhya.com/blog/2021/11/training-neural-network-with-keras-and-basics-of-deep-learning/

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar