Department of

Electrical & Electronics Engineering

UNIVERSITI TEKNOLOGI PETRONAS

AI 101: Your First Step into Machine Learning with Tensorflow

Workshop morning session

UNIVERSITI TEKNOLOGI PETRONAS
energising futures

Volintine Ander

volintine_21001524@utp.edu.my

Table of contents

Basic ideas

Weights and biases

We can transform a number linearly[1] using the equation of a straight line $y = wx + b$

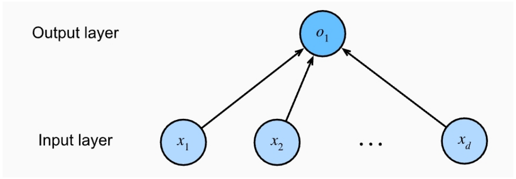
Definition

For an input layer with n nodes, the output of the layer is expressed as a weighted sum

$$y = \sum_{i}^{n} (w_i x_i) + b$$

where w_i is the weight of the ith node and b is the bias.

x_i can be thought of as our input, and y is the network's prediction given the values of x_i. A node can be thought of as the model's neuron.



[1] Some mathematicians are less than excited about people using *linear* to mean a first-degree polynomial.

Weights & biases

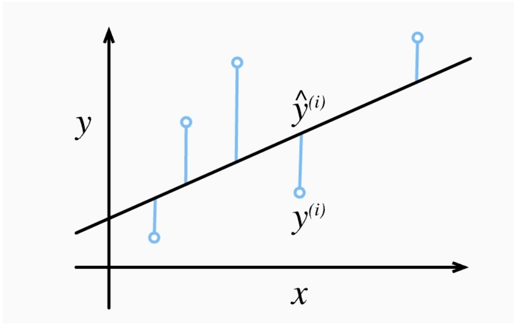
Think about this

The most basic artificial 'intelligence' is a straight line of best fit. When one performs experiments and plots the measured samples, one can try to find a straight line that minimizes the distances of each sample point from the line.

In this case, the equation of best-fitting straight line is the model.

This model can predict the values of y, given x.

This allows us to get an approximation of the variable y given x without redoing the experiment to measure y for a particular x.

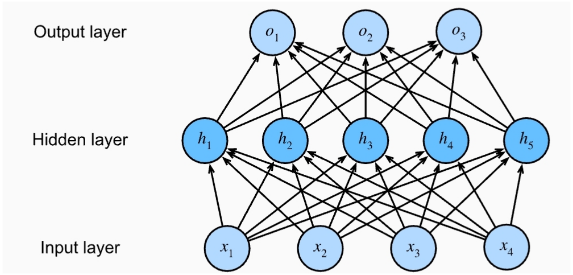


Nodes, layers, & activation functions

A node is the simplest unit in a network. Connections between nodes are called synapses.

We can take multiple nodes together to form a single layer. At the input layer, each node will take one input.

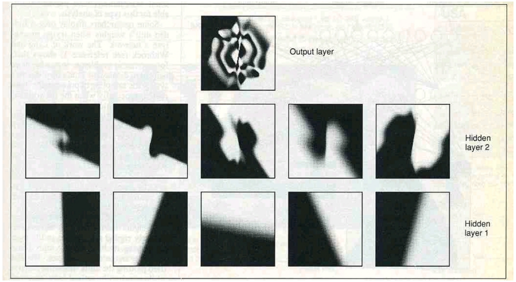
At the output layer, the number of nodes is not necessarily the same as at the input.

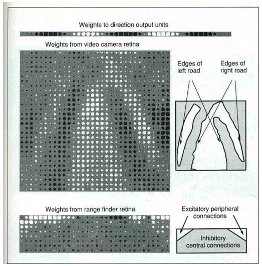


Hidden layers

In between the input and output layers are ‘hidden’ layers. Hidden layers are used to extract relevant features (e.g., specific shape and color combinations).

Each node’s output goes through an activation function. Activation functions are important because they add some nonlinearity[2] to the network. This is key to allowing a network to learn more complex patterns.




Weights to direction output units
Weights from video camera retina
Edges of left road
Edges of right road
Weights from range finder retina
Excitatory peripheral connections
Inhibitory central connections

Nonlinearity can also occur from rounding errors during training, without activation functions. Read this paper for a jocular explanation: http://tom7.org/grad/
Corresponding video by the author: https://youtu.be/Ae9EKCyI1xU

Loss function

The loss function, simply put, is a way of representing errors in the model's predictions. Like activation functions, there are many kinds of loss functions to choose from. The loss function plays an important role in backpropagation, which is a way of providing feedback to the network given its output and a predetermined expected output.

Definition (Mean square error)

MSE is a popular loss function used in machine learning.

$$\text{MSE} = \frac{1}{N} \sum_{(x,y) \in D} (y - \text{prediction}(x))^2$$

where D is a dataset of the ordered pairs (x, y) where x is a feature (input variable) used for prediction and y is the label (thing predicted). N is the number of (x, y) pairs in D.

How computers learn

Data encoding

Encoding

All information on a computer is represented in binary.

This means that all forms of information, be it text, sound, images, are all preprocessed into huge arrays of numbers before a model can be trained on the dataset.

Word embedding

1. Words can be grouped together as vectors in multidimensional space.
2. The coordinates of each word are correlated with how related they are to other words.

In a model, these coordinates are learned parameters that are tuned based on how well the model does on a specific task such as *sentiment analysis*.

Tokenization

In Natural Language Processing (NLP), text needs to be broken up so that a model can 'see' patterns that matter in human language. These little pieces of information are called tokens, which are usually words, pieces of words, punctuation, and numbers.

Problem

The issue is that it will be harder for the model to learn grammar, which often manifests in each word as inflected forms. Treating affixes as tokens by cutting them off words will make it more obvious to the model.

Example: Kadazandusun

An agglutinative language from Sabah. Consider the root word *akan* (to eat).

makan	having a meal
taakanon	food
miakan	two people eating each other
mangakan	is/are eating
minakan	was/were having a meal

How tokenization helps

Separating affixes from root words will ‘tell’ the model that all it has to do is to join these affixes to root words to create different meanings.

If done properly, tokenization allows for efficient feature extraction.

In this example, tokenization separates the root word from its affixes beforehand.

This allows the model to see affixes and root words as parts of a word instead of seeing the different inflections as different words completely.

Possible tokenization

Word	Token string
makan	m | akan
taakanon	ta | akan | on
miakan	mi | akan
mangakan	mang | akan
minakan	m | in | akan

The | symbol delineates each token.

Forward propagation, backpropagation, & gradient descent

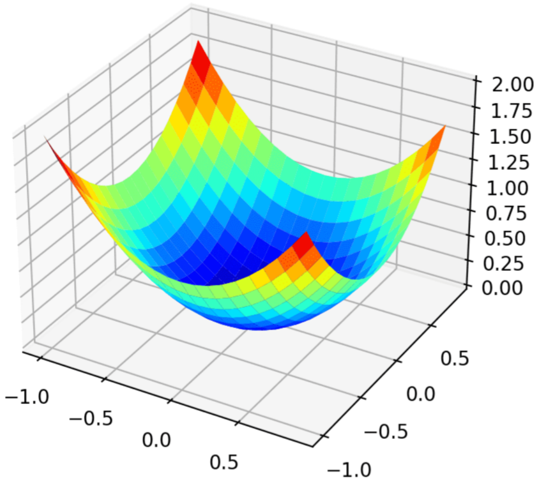
What we need

Forward pass or forward propagation is done to find the objective function.

Backpropagation is done to find the gradient of network parameters[3]. Here, we are finding the partial derivative of the objective function with respect to the weights.

The goal

The process that eventually optimizes the weights is called gradient descent.
This algorithm requires the partial derivative of the objective function with respect to the model parameters.



A parameter of a network is any quantity that is tuned by the network during training. Weights and biases are parameters. In contrast, hyperparameters are quantities that we choose.

Example

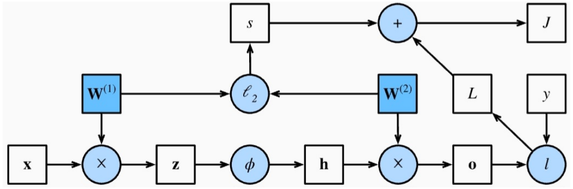


Figure: Computational graph representation of forward propagation

For a single-hidden-layer perceptron without a bias term

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \frac{\partial J}{\partial \mathbf{z}}\mathbf{x}^\top + \lambda \mathbf{W}^{(1)}$$

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \frac{\partial J}{\partial \mathbf{o}}\mathbf{h}^\top + \lambda \mathbf{W}^{(2)}$$

where J is the objective function, $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$ are weight tensors for the hidden layer and output layer respectively. $\mathbf{x}$ is the input, $\mathbf{z}$ is the hidden layer transformation on $\mathbf{x}$, $\mathbf{o}$ is the output, and $\mathbf{h} = \varphi(\mathbf{z})$, with φ representing the activation function.

$$J = \ell(\mathbf{o}, y) + s$$

y is the label and s is the regularization term that helps to prevent underfitting and overfitting.

Activity: Tokenization with Tensorflow

Tokenizers

Task 1

Open the Google Colab notebook link which will be given to you. Each participant will be given unique notebooks.

Acquire text and perform **whitespace tokenization**,
as well as **Unicode Script tokenization**.

You may use any grammatical English text that you can find or generate.

Observe the output of the whitespace tokenizer. Why were the punctuation marks not tokenized separately?
Observe the output of the Unicode Script tokenizer. Why were the punctuation marks tokenized separately?

Tokenizers

Task 2

Acquire an excerpt of Chinese text and perform **whole-word tokenization**.

You may use Google Translate or other text that you can find or generate.

Observe the output.
Why do you think this is called whole-word tokenization?

Task 3

Train a tokenizer on Kadazandusun text using the SentencePiece library.

Observe the output.
Why do you think the tokenization performed is called subword tokenization?
The dataset contains 22,485 words. How many words in a dataset do you think it will take to create a Kadazandusun ChatGPT?

Explore Tensorflow Projector & Hugging Face

Explore

Visualize word embeddings

Go to projector.tensorflow.org and explore.

Can you explain what is being shown?

Discover Hugging Face

Go to huggingface.co and sign up through your Google account if you don't have an account yet.

Explore the demos you find the most interesting.

Further reading

You don't need a lot of mathematical background to be involved in AI and machine learning, but it helps to brush up on linear algebra.

1. K. Kuttler. *A First Course in Linear Algebra*. lyryx.com/first-course-linear-algebra/
2. J. Gareth, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor. *An Introduction to Statistical Learning*. statlearning.com
3. *Dive into Deep Learning* d2l.ai