1

Nature-inspired computing

- Nature has always served as a source of inspiration for engineers and scientists
- The best problem solver known in nature is:
  - the (human) brain that created "the wheel, New York, wars and so on" (after Douglas Adams' Hitch-Hikers Guide)
  - the evolution mechanism that created the human brain (after Darwin's Origin of Species)
- Answer 1 → neurocomputing
  - Today
- Answer 2 → evolutionary computing
  - Weeks 7 + 8

2 / 39

2

Outline

- Artificial Neural Networks (ANNs)
  - Neural networks
  - Logistic regression – revision
  - Different learning tasks
  - ANNs
  - Cost function
  - Gradient descent
  - Backpropagation
- Deep Learning

3

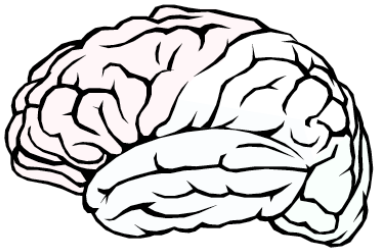
Supervised Learning using Artificial Neural Networks

4

Neural Networks

- Human brain consists of around 86 billion interconnected "neurons"
- Each neuron is connected to about 1000+ neighbouring neurons
- Each neuron responds to stimuli and passes output to other neurons

- In ML, artificial neural networks are loosely modelled on the human brain



5

Biological Networks of Neurons

- A neuron receives inputs (electrochemical signals) from its neighbours
- If enough inputs are received at the same time, it gets *activated*
- Once activated, a neuron fires, giving an output that may (in turn) activate connected neurons
- The behaviour of a biological network of neurons is dependent on connection types and activation strength (*synaptic function*)

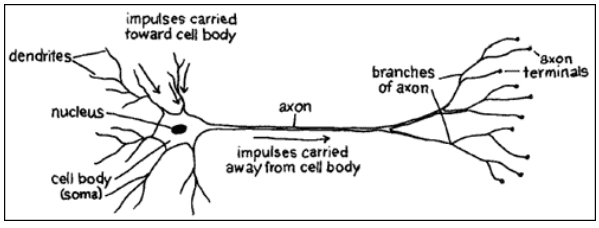
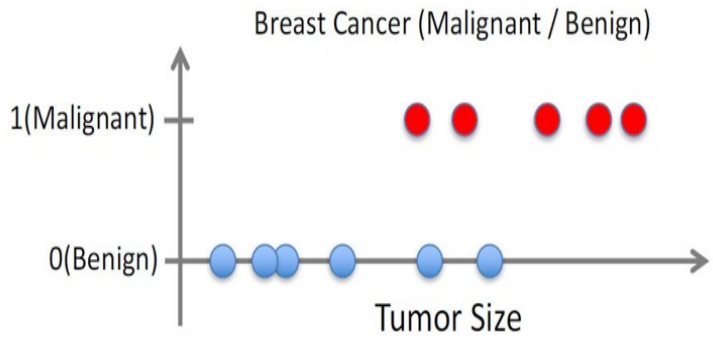


Image Source: https://science.education.nih.gov/supplements/webversions/BrainAddiction/guide/lesson2-1.html

6

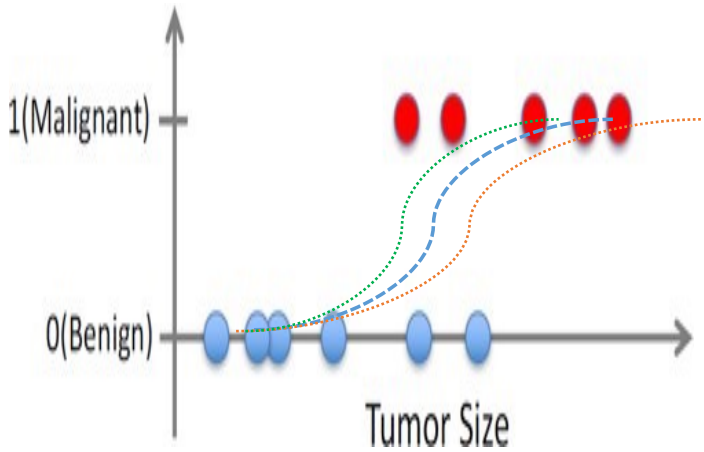
Review - Logistic regression

- It is actually a classification technique!



Source: Andrew Ng/Stanford Online

7



- Note that the blue model is not the only one. There are many. We still face the problem of which function is the "right" one

- → **minimize the cost function using Gradient Descent (GD) or Maximum Likelihood Estimation (MLE)**

8

Logistic regression - mathematically

- $\hat{y} = f_{W,b}(x) = \sigma\left(g_{W,b}(x)\right) = \frac{1}{1+e^{-g_{W,b}(x)}}$
- More precisely: $\text{Prob}(y = 1|x) = \frac{1}{1+e^{-g_{W,b}(x)}}$
- E.g., $\text{Prob}(y = \text{Malignant}|size) = \frac{1}{1+e^{-g_{w,b}(size)}}$
- where $g_{w,b}(size) = b + w * size$

9

Logistic regression - mathematically

- $\hat{y} = f_{W,b}(x) = \sigma\left(g_{W,b}(x)\right) = \frac{1}{1+e^{-g_{W,b}(x)}}$
- Of course, for multivariate problem:

$$g_{W,b}(x_1, \dots, x_n) = b + w_1 * x_1 + \cdots + w_n * x_n$$

W is a vector $W = (w_1, \dots, w_n)$

Some people write: $g_W(x_1, \dots, x_n) = w_0 + w_1 * x_1 + \cdots + w_n * x_n = Wx$

by assuming: $x_0 = 1$ and the vector $W = (w_0, w_1, \dots, w_n)$

We are supposed to write $W^T x$ but we can keep it simple

10

Logistic regression - mathematically

- $\hat{y} = f_W(x) = \sigma\big(g_W(x)\big) = \frac{1}{1+e^{-g_W(x)}}$

- The cost function:

$$J(W) = \frac{1}{m}\sum_{i=1}^{m} y^i \log \hat{y}^i + (1-y^i)\log(1-\hat{y}^i)$$

- The model f_W can then be trained by minimizing $J(W)$ using the Gradient Descent (GD) algorithm.

11

Logistic Regression for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$

(32x32x3)

$$\rightarrow \begin{pmatrix} x_0 = 1 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \rightarrow \boxed{Wx \mid \sigma} \rightarrow \hat{y} = f_{W^*}(x) = \sigma(W^*x)$$

General steps in a Gradient Descent (GD) algorithm:

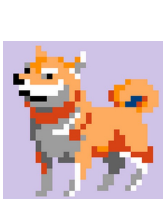
- Initialise (randomly) W
- Repeat the updates $W := W - \alpha \frac{\partial J(W)}{\partial W}$ until convergence to a minimum

Use the trained model f_{W^*} for inference: Given an input x, $\hat{y} = f_{W^*}(x)$

12

Logistic Regression for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$



(32x32x3)

$(x_0 = 1)$

x_1

x_2

$\vdots$

(artificial) neuron

$\hat{y} = f_{W^*}(x) = \sigma(Wx)$

More generally:

Neuron = Linear + Activation

(& there are many **activation functions**: Sigmoid, step, linear, tanH, Gaussian, ReLU, etc.)

General step

- Initialise (
- Repeat the updates $W := W - \alpha \frac{\partial J(W)}{\partial W}$ until convergence to a minimum

Use the trained model f_{W^*} for inference: Given an input x, $\hat{y} = f_{W^*}(x)$

13

Logistic Regression for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$

(32x32x3)

$(x_0 = 1)$

x_1

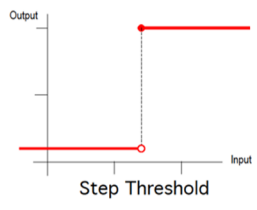
x_2

(artificial) neuron

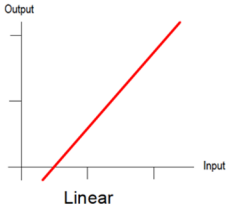
General s

- Initiali
- Find th
- Use W

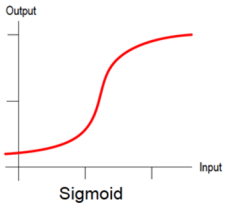
$y = \begin{cases} A_1, if\ w.x + b > 0 \\ A_0, \quad otherwise \end{cases}$



$y = \sum_{i=1}^{N} x_i w_i$



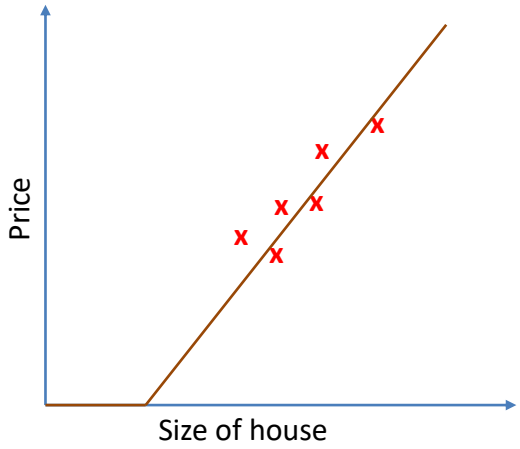
$y = \frac{1}{1+e^{-x}}$

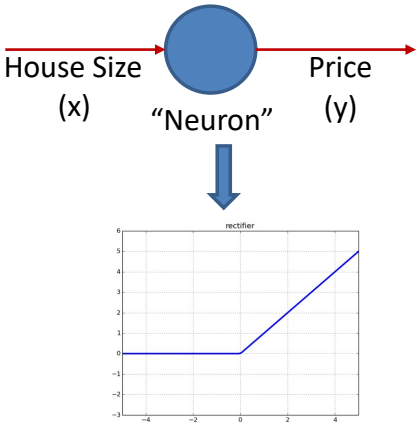


Learn more about activation functions - https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

14

Price Prediction –Single Neuron





Rectified **L**inear **U**nit

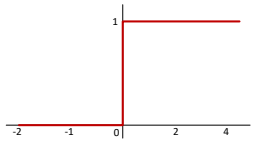
15

Perceptron

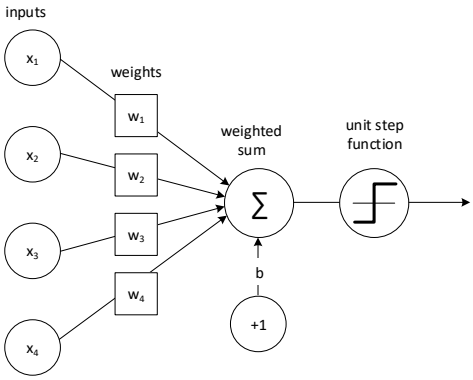
- Simplest neural network possible
- Single neuron that performs binary classification
- Uses a *step activation function*
  - Also referred to as *Heaviside Step Function*

$$f(x) = \begin{cases} 1, & if\ w.x + b > t \\ 0, & otherwise \end{cases}$$

where $w.x$ is the dot product $\sum_{i=1}^{N} x_i w_i$ and t is the threshold



Step Activation Function



16

What about Multinomial Classification?

(aka. Multiclass classification)

Task 2: Classifying images into cat/dog/bird?

(32x32x3)

Label: $\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$

(dog)

$(x_0 = 1)$

$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}$

$W^{[1]}x \mid \sigma \rightarrow \hat{y}_c = \sigma(W^{[1]}x)$

$W^{[2]}x \mid \sigma \rightarrow \hat{y}_d = \sigma(W^{[2]}x)$

$W^{[3]}x \mid \sigma \rightarrow \hat{y}_b = \sigma(W^{[3]}x)$

17

What about Multinomial Classification?

(aka. Multiclass classification)

Task 2: Classifying images into cat/dog/bird?

(32x32x3)

$(x_0 = 1)$

$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix}$

$W^{[1]}x \mid \sigma \rightarrow \hat{y}_c = \sigma(W^{[1]}x)$

$W^{[2]}x \mid \sigma \rightarrow \hat{y}_d = \sigma(W^{[2]}x)$

$W^{[3]}x \mid \sigma \rightarrow \hat{y}_b = \sigma(W^{[3]}x)$

Cost function:

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{3}\left(y^i \log\left(\sigma\left(W^{[j]}x^i\right)\right) + (1-y^i)\log\left(1-\sigma\left(W^{[j]}x^i\right)\right)\right)$$

18

What about Multinomial Classification?

(aka. Multiclass classification)

Task 2: Classifying images into cat/dog/bird?

(32x32x3)

$$(x_0 = 1),\ x_1, x_2, \vdots, x_{n-1}, x_n$$

$$W^{[1]}x \,|\, \sigma \rightarrow \hat{y}_c = \sigma(W^{[1]}x)$$

$$W^{[2]}x \,|\, \sigma \rightarrow \hat{y}_d = \sigma(W^{[2]}x)$$

$$W^{[3]}x \,|\, \sigma \rightarrow \hat{y}_b = \sigma(W^{[3]}x)$$

Label: $\begin{pmatrix}0\\1\\0\end{pmatrix}$ (dog)

And what about this example + label?

$\begin{pmatrix}1\\1\\0\end{pmatrix}$ (dog & cat)

Task 2': Is there a cat/dog/bird in an image.

19

Multinomial Classification under constraints

Task 3: Which animal (a cat, dog or bird) is in an image (and at most one of them)?

The **softmax multiclass regression**

(32x32x3)

$$(x_0 = 1),\ x_1, x_2, \vdots, x_{n-1}, x_n$$

$$Z^1 = W^{[1]}x$$

$$Z^2 = W^{[2]}x$$

$$Z^3 = W^{[3]}x$$

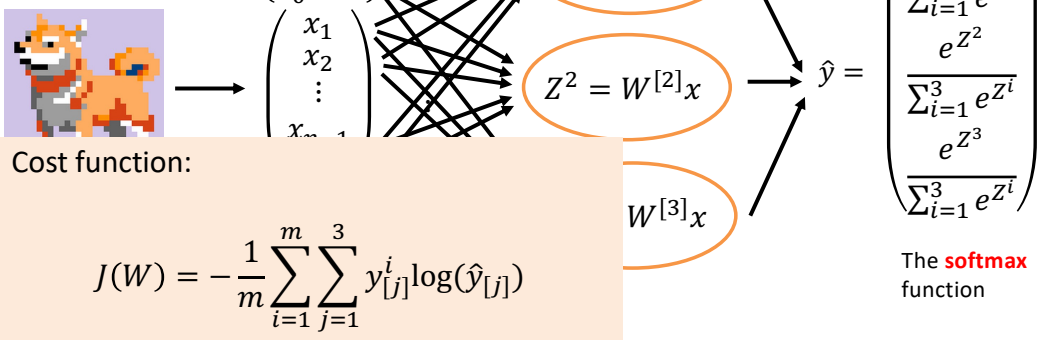
$$\hat{y} = \begin{pmatrix} \frac{e^{Z^1}}{\sum_{i=1}^{3} e^{Z^i}} \\ \frac{e^{Z^2}}{\sum_{i=1}^{3} e^{Z^i}} \\ \frac{e^{Z^3}}{\sum_{i=1}^{3} e^{Z^i}} \end{pmatrix}$$

The **softmax** function

20

Multinomial Classification under constraints

Task 3: Which animal (a cat, dog or bird) is in an image (and at most one of them)?



The **softmax** function

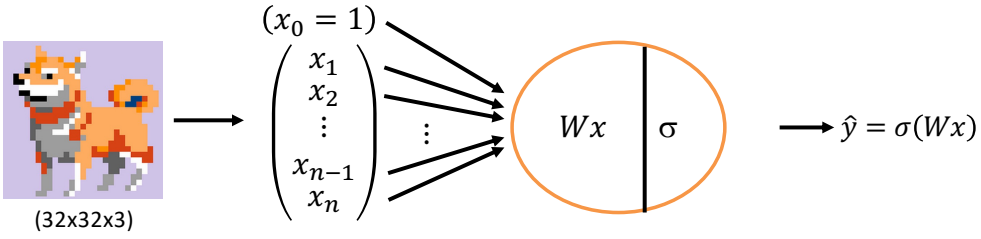
Cost function:

$$J(W) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{3} y^i_{[j]}\log(\hat{y}_{[j]})$$

21

Logistic Regression for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$



(32x32x3)

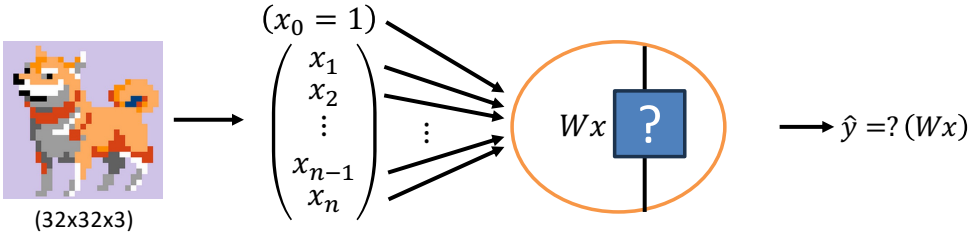
General steps in a Logistic Regression algorithm:

- Initialise (randomly) W
- Find the optimal W^* ($W = W - \alpha \frac{\partial J(W)}{\partial W}$ over a number of iterations)
- Use W^* for inference: Given an input x, $\hat{y} = \sigma(W^* x)$

22

Machine Learning for Image processing

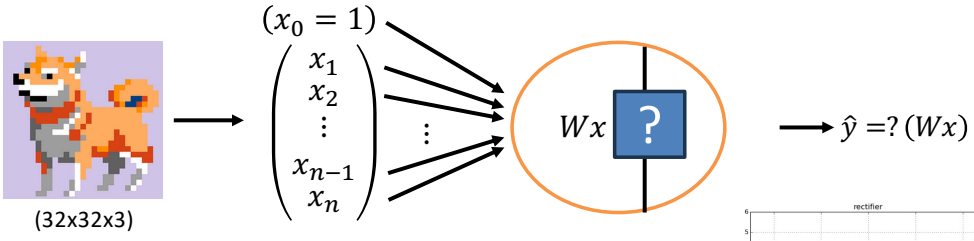
Task 1': Determine the age of the dog is in the image?



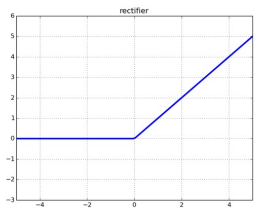
23

Machine Learning for Image processing

Task 1': Determine the age of the dog is in the image?



Remember the ReLU activation function?



Rectified Linear Unit

24

Here come the big question…

- Would it suffice to have just linear + activation (with the right activation function, of course)
  - Classification? ✓
  - Regression? ✓
- Then should we complicate our lives with neurons and neural networks?

25

Things are not always simple

- There can be hundreds/thousands/ even millions of people we need to recognise
- There are millions of species of animals

26

Things are not always simple

- There are thousands of languages used by people around the world
  - Different syntaxes and grammars
  - Different rules

27

Things are not always simple

- Even in a single language (e.g., English), voice recognition systems have to deal with many different accents

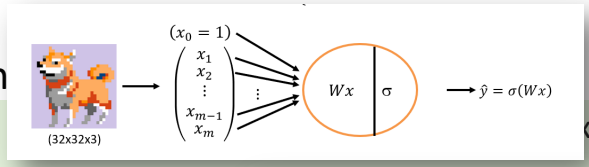
28

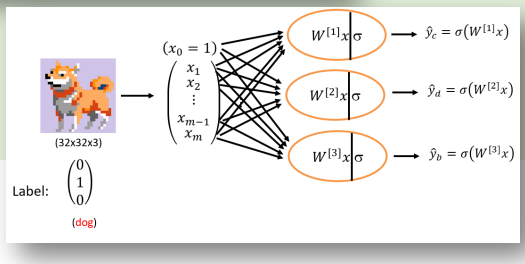
Here come the big question…

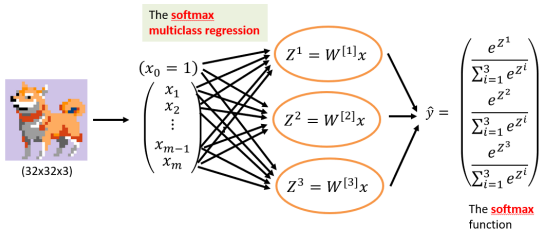
- Would it suffice to … the right activation function …

We have seen different … kity:

- Task 1 (and 1')
- Task 2 (and 2')
- Task 3







29

These can be your task 100th, 157th, …

- There can be hundreds/thousands/ even millions of people we need to recognise
- There are millions of specie…

Do we really have to design a new algorithm for each task?

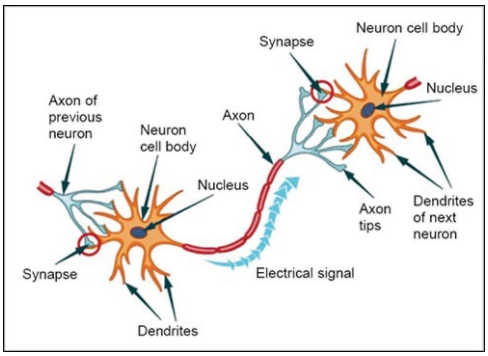
Is there a magical formula to get most of them into a single "**architecture template**" that we only need to make small adjustments and start with training our ML models?

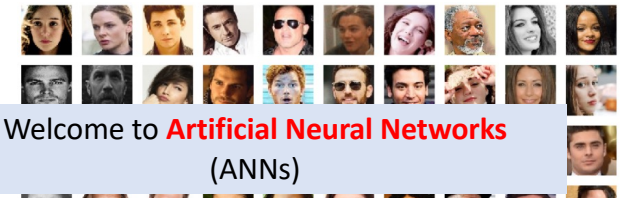
Welcome to **Artificial Neural Networks** (ANNs)

30

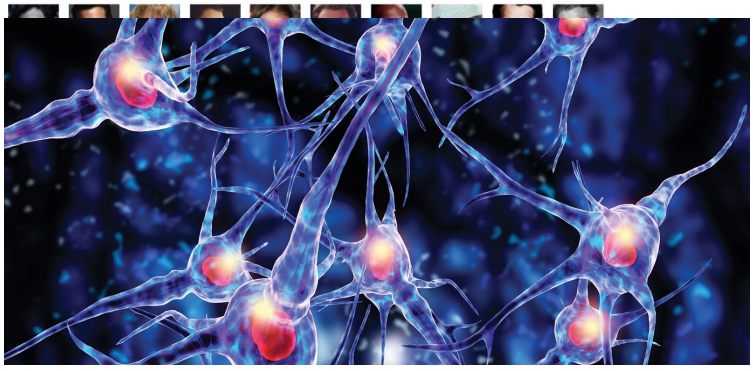
These can be your task 100th, 157th, …

- There can be hundreds/thousands/ even millions of people we need to recognise





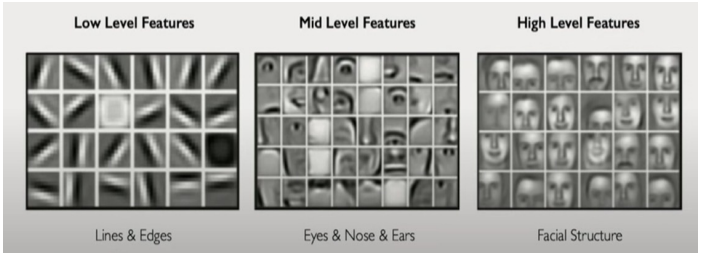
Welcome to **Artificial Neural Networks** (ANNs)

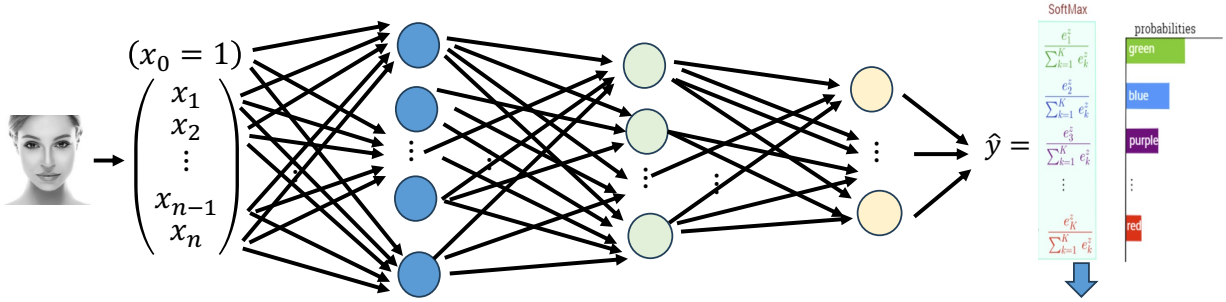


SWINBURNE UNIVERSITY OF TECHNOLOGY

31

Welcome to Artificial Neural Networks (ANNs)





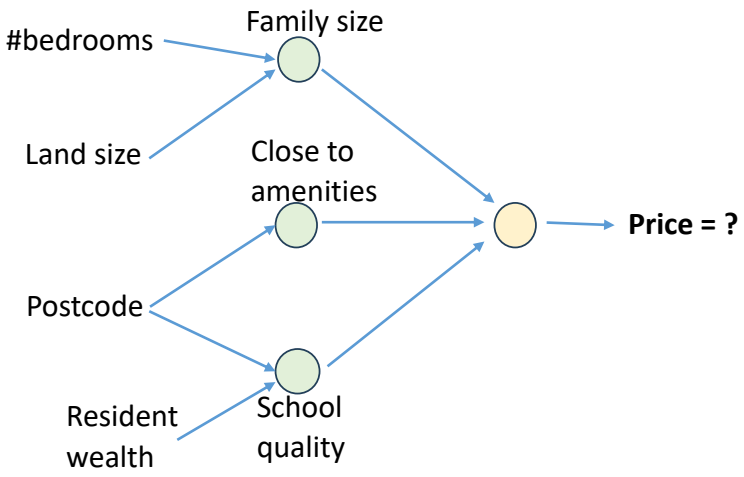
It's **Taylor**! **72%** certain!

SWINBURNE UNIVERSITY OF TECHNOLOGY

32

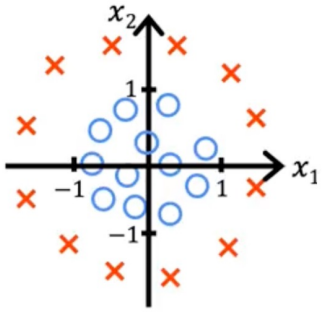
(Automatic) Feature Engineering

Task: Predict the house price.

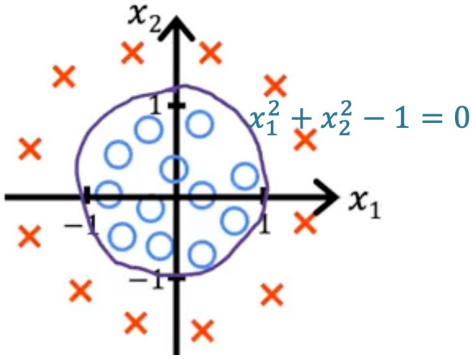


33

(Automatic) Feature Engineering



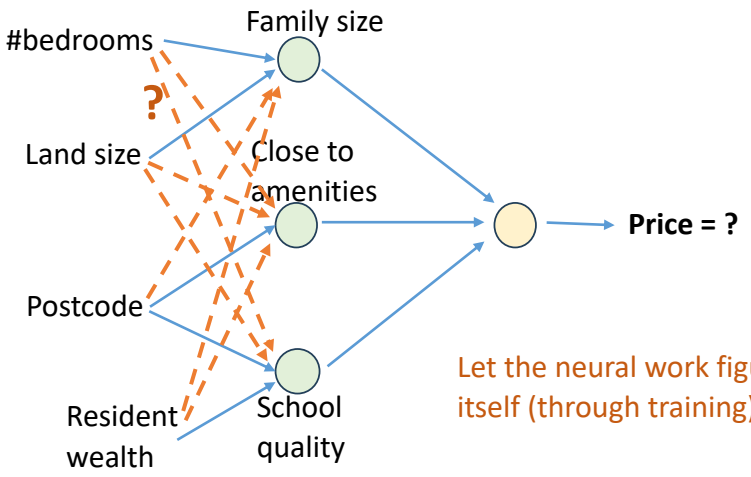
$$f_{\vec{w},b}(\vec{x}) = g(z) = g(\overbrace{w_1 x_1^2 + w_2 x_2^2 + b}^{z})$$



34

(Automatic) Feature Engineering

Task: Predict the house price.

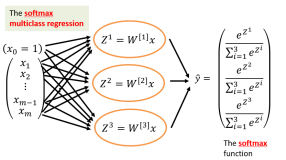


Let the neural work figure this question out by itself (through training)

35

Important observation

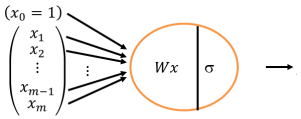
- There can be many ML problems/tasks but (most of) them can be dealt with by neural networks of different architectures (after a training process)

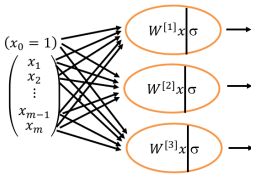


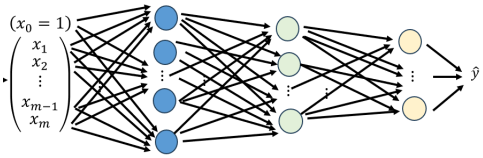
Artificial Neural Network (ANN)/model = **architecture** + **parameters**

ML practitioners

Trained with data



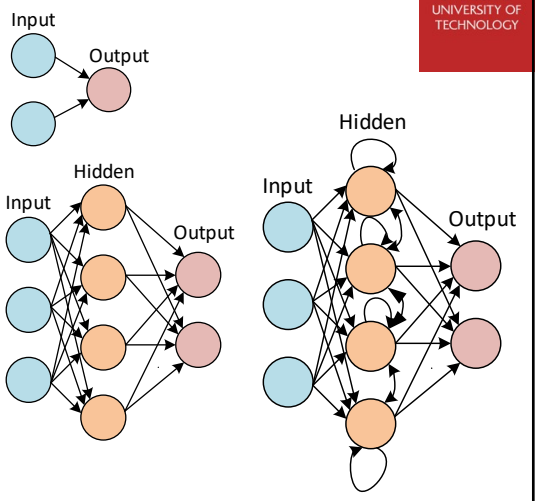




36

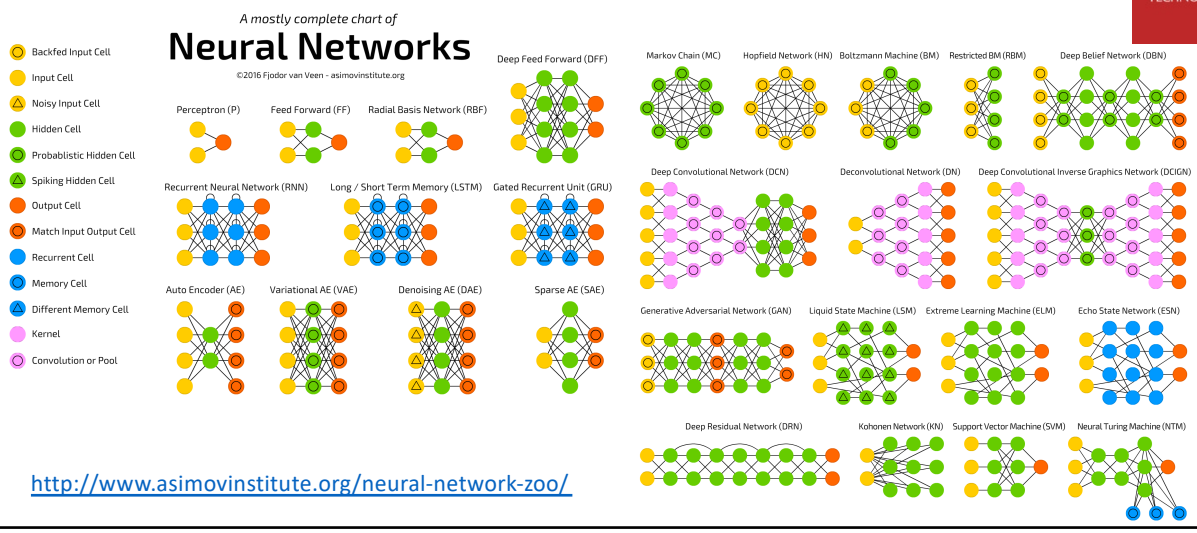
Complex Neural Network - Architectures

- There are 2 main categories of neural network structures:
  - Acyclic/Feed-Forward Networks – no loops, no internal state
    - Single-layer perceptron
    - Multi-layer perceptron
  - Cyclic/Recurrent Networks – feeds its outputs back into its own inputs, directed cycles, supports short-term memory



37

Types of Neural Networks

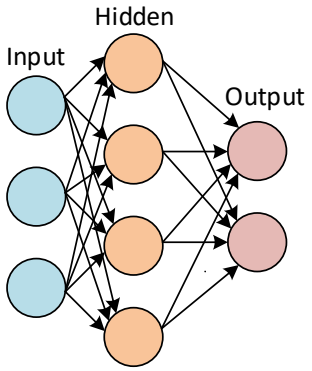


http://www.asimovinstitute.org/neural-network-zoo/

38

Multiple-Layer Neural Networks

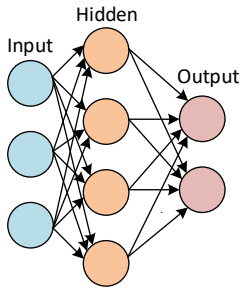
- A complex neural network involves a set of neurons (e.g. perceptrons) interconnected in layers to solve more complex problems
  - Input layer neurons connect to hidden layer neurons
  - Hidden layer neurons connect to output layer neurons
  - Output layer can have multiple outputs
- Multilayer neural networks can classify a range of functions including non-linearly separable ones



39

Learning in Multiple-Layer Neural Networks

- Multilayer NN learn the same way as perceptrons
- **Forward pass** – compute the values from input to output
- **Backward pass** – propagate errors backwards to adjust the connection weights
- Error occurs when expected output != actual output
- **Cost function** – a measure of "how good or bad" a NN did w.r.t its given training sample and the expected output



40

Gradient Descent for Artificial Neural Networks

41

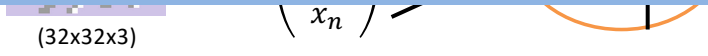
Logistic Regression for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$

It's unlikely giving us a good classifier:

The flattened single feature vector representing the *m* pixels loses the spatial relationships within the image

→ Require effective feature engineering to extract meaningful features (e.g., edges, textures, colour histograms, etc.)



General steps in a Gradient Descent (GD) algorithm:

- Initialise (randomly) W
- Repeat the updates $W := W - \alpha \frac{\partial J(W)}{\partial W}$ until convergence to a minimum

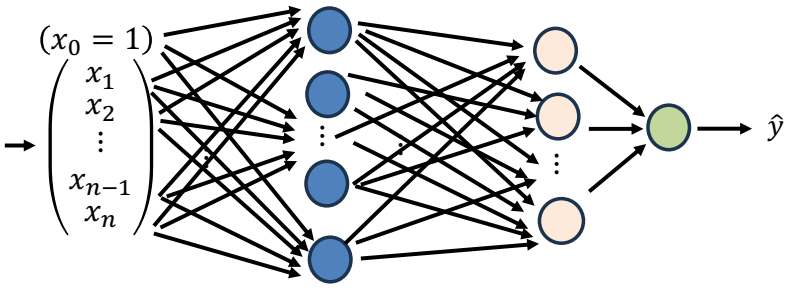
Use the trained model f_{W^*} for inference: Given an input x, $\hat{y} = f_{W^*}(x)$

42

ANN for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$

(32x32x3)

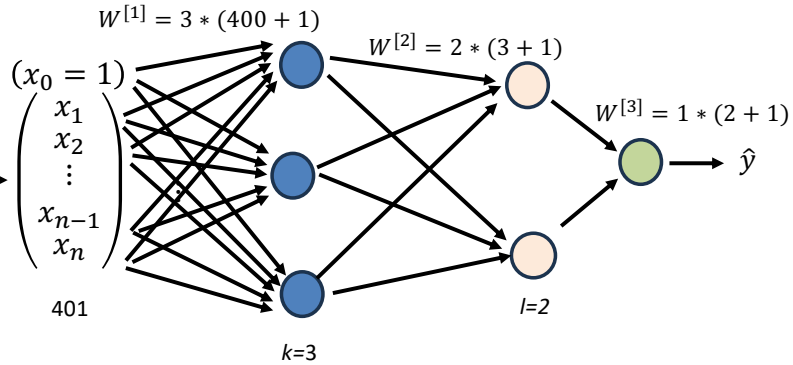


43

ANN for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$

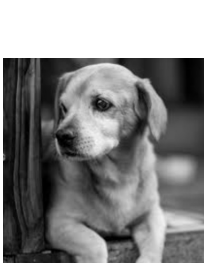
(20x20x1)
n = 400



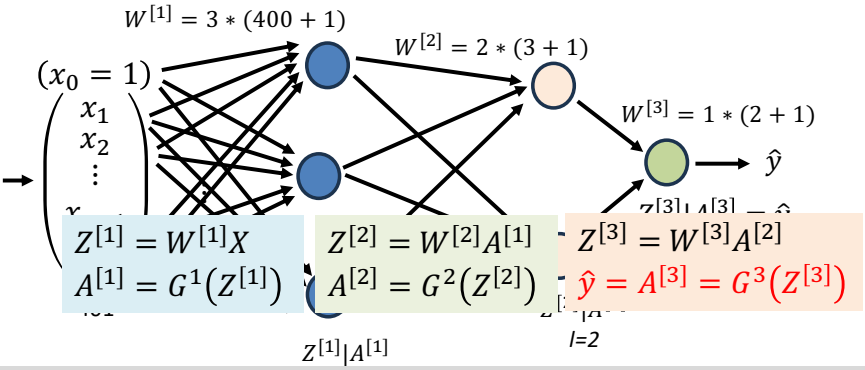
44

ANN for Image classification

Task 1: Determine if a dog is in the image: $\begin{cases} 1 & \text{if there is a dog} \\ 0 & \text{if there isn't a dog} \end{cases}$



(20x20x1)
n = 400



$Z^{[1]} = W^{[1]}X$
$A^{[1]} = G^1(Z^{[1]})$

$Z^{[2]} = W^{[2]}A^{[1]}$
$A^{[2]} = G^2(Z^{[2]})$

$Z^{[3]} = W^{[3]}A^{[2]}$
$\hat{y} = A^{[3]} = G^3(Z^{[3]})$

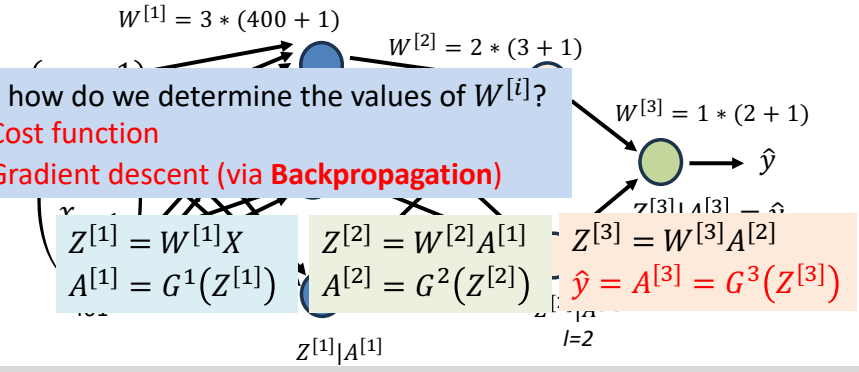
Note: G^i are the activation functions used at the neurons (and can be different at different neurons)

45

ANN for Image classification

If we have the model (i.e., have determined the values of $W^{[i]}$'s), then the steps below enable us to use the model for inference

(20x20x1)
n = 400



But, how do we determine the values of $W^{[i]}$?
- Cost function
- Gradient descent (via **Backpropagation**)

$Z^{[1]} = W^{[1]}X$
$A^{[1]} = G^1(Z^{[1]})$

$Z^{[2]} = W^{[2]}A^{[1]}$
$A^{[2]} = G^2(Z^{[2]})$

$Z^{[3]} = W^{[3]}A^{[2]}$
$\hat{y} = A^{[3]} = G^3(Z^{[3]})$

Note: G^i are the activation functions used at the neurons (and can be different at different neurons)

46

ANN – Cost function

$$J(\boldsymbol{W}) = J\left(W^{[1]}, W^{[2]}, W^{[3]}\right)$$
$$= \sum_{i=1}^{m} loss^{i}(\boldsymbol{W}) = \sum_{i=1}^{m} loss\left(\hat{y}^{(i)}, y^{(i)}; \boldsymbol{W}\right)$$

For instance, for the image classification problem:

$loss(\hat{y}^{(i)}, y^{(i)}; \boldsymbol{W}) = -y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$

That is, we want to compute

$$W^{[l]} := W^{[l]} - \alpha \frac{\partial J(\boldsymbol{W})}{\partial W^{[l]}} \text{ for every layer } l = 1,2,3$$

- Repeat the updates $\boldsymbol{W} := \boldsymbol{W} - \alpha \frac{\partial J(\boldsymbol{W})}{\partial \boldsymbol{W}}$ until convergence to a minimum

47

ANN – Backpropagation

Remember: $Z^{[3]} = W^{[3]} A^{[2]}$, $\hat{y} = A^{[3]} = G^{3}(Z^{[3]})$

$$J(\boldsymbol{W}) = \sum_{i=1}^{m} loss\left(\hat{y}^{(i)}, y^{(i)}; \boldsymbol{W}\right)$$

Chain rule:

$$\frac{\partial J(\boldsymbol{W})}{\partial W^{[3]}} = \frac{\partial J(\boldsymbol{W})}{\partial A^{[3]}} \cdot \frac{\partial A^{[3]}}{\partial Z^{[3]}} \cdot \frac{\partial Z^{[3]}}{\partial W^{[3]}}$$

Similarly: $Z^{[2]} = W^{[2]} A^{[1]}$, $A^{[2]} = G^{2}(Z^{[2]})$

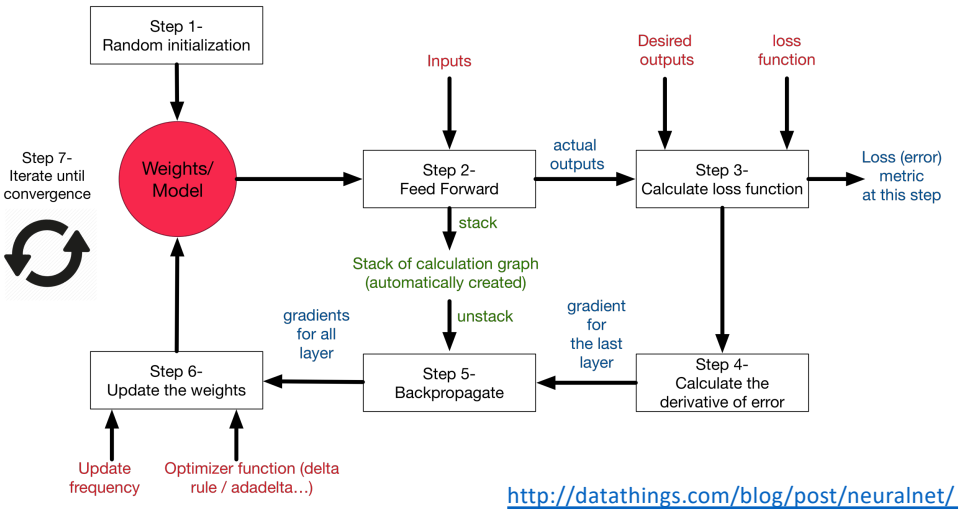
$Z^{[1]} = W^{[1]} X$, $A^{[1]} = G^{1}(Z^{[1]})$

$$\frac{\partial J(\boldsymbol{W})}{\partial W^{[2]}} = \frac{\partial J(\boldsymbol{W})}{\partial Z^{[3]}} \cdot \frac{\partial Z^{[3]}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial W^{[2]}}$$

$$\frac{\partial J(\boldsymbol{W})}{\partial W^{[1]}} = \frac{\partial J(\boldsymbol{W})}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial W^{[1]}}$$

48

Learning in Multiple-Layer Neural Networks

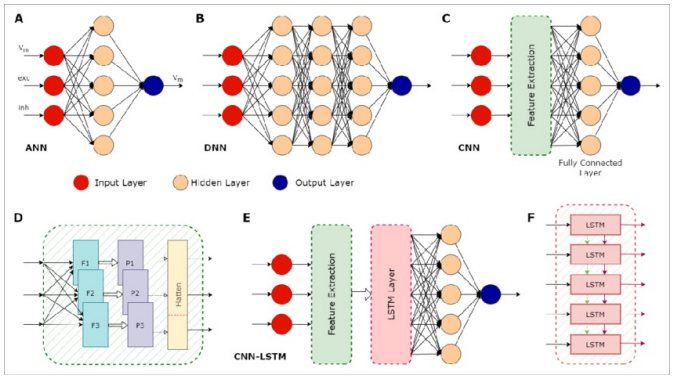


http://datathings.com/blog/post/neuralnet/

49

Why DL now?

- Better algorithms & understanding
- Computing power (GPUs, TPUs, ...)
- Data with labels
- Open-source tools and models

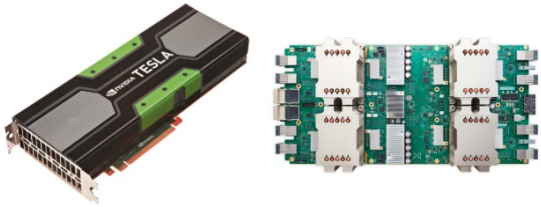


Source: **https://www.ip-paris.fr/education/masters/mention-mathematiques-appliquees-statistique/master-year-2-data-science**

50

Why DL now?

- Better algorithms & understanding
- Computing power (GPUs, TPUs, ...)
- Data with labels
- Open-source tools and models



GPU and TPU

Source: **https://www.ip-paris.fr/education/masters/mention-mathematiques-appliquees-statistique/master-year-2-data-science**

51

Why DL now?

- Better algorithms & understanding
- Computing power (GPUs, TPUs, ...)
- Big data
- Open-source tools and models

Source: **https://www.ip-paris.fr/education/masters/mention-mathematiques-appliquees-statistique/master-year-2-data-science**

52

Why DL now?

- Better algorithms & understanding
- Computing power (GPUs, TPUs, ...)
- Data with labels
- Open-source tools and models

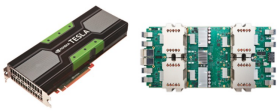
Source: **https://www.ip-paris.fr/education/masters/mention-mathematiques-appliquees-statistique/master-year-2-data-science**

53

Why GPU/TPU? Parallelization?

- Computing power (GPUs, TPUs, ...)
- Data with labels
- Open-source tools and models

(32x32x3)



GPU and TPU

$$J(W) = \sum_{i=1}^{m} loss\left(\hat{y}^{(i)}, y^{(i)}; W\right)$$

ANN – Backpropagation

Remember: $Z^{[3]} = W^{[3]}A^{[2]}$, $\hat{y} = A^{[3]} = G^3(Z^{[3]})$, $J(W) = \sum_{i=1}^{m} loss(\hat{y}^{(i)}, y^{(i)}; W)$

Chain rule: $\frac{\partial J(W)}{\partial W^{[3]}} = \frac{\partial J(W)}{\partial A^{[3]}} \cdot \frac{\partial A^{[3]}}{\partial Z^{[3]}} \cdot \frac{\partial Z^{[3]}}{\partial W^{[3]}}$

Similarly: $Z^{[2]} = W^{[2]}A^{[1]}$, $A^{[2]} = G^2(Z^{[2]})$, $Z^{[1]} = W^{[1]}X$, $A^{[1]} = G^1(Z^{[1]})$

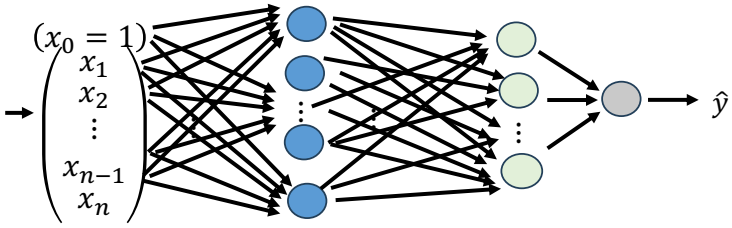
$\frac{\partial J(W)}{\partial W^{[2]}} = \frac{\partial J(W)}{\partial Z^{[3]}} \cdot \frac{\partial Z^{[3]}}{\partial A^{[2]}} \cdot \frac{\partial A^{[2]}}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial W^{[2]}}$

$\frac{\partial J(W)}{\partial W^{[1]}} = \frac{\partial J(W)}{\partial Z^{[2]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdot \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[1]}}{\partial W^{[1]}}$

54

What about batch, epoch, SGD?

(32x32x3)
=3,072



$$J(\boldsymbol{W}) = \sum_{i=1}^{m} loss\left(\hat{y}^{(i)}, y^{(i)}; \boldsymbol{W}\right)$$

m=10,000,000??

55

Wh

(32x3
=3,0

$J(\boldsymbol{W})$ =

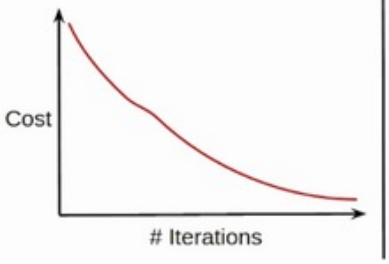
Aspect	Batch Gradient Descent	Stochastic Gradient Descent (SGD)
Data Processing	Uses the whole training dataset to compute the gradient.	Uses a single training sample to compute the gradient.
Convergence Speed	Slower, takes longer to converge.	Faster, converges quicker due to frequent updates.
Convergence Accuracy	More accurate, gives precise gradient estimates.	Less accurate due to noisy gradient estimates.
Computational and Memory Requirements	Requires significant computation and memory.	Requires less computation and memory.
Optimization of Non-Convex Functions	Can get stuck in local minima.	Can escape local minima and find the global minimum.
Suitability for Large Datasets	Not ideal for very large datasets due to slow computation.	Can handle large datasets effectively.
Nature	Deterministic: Same result for the same initial conditions.	Stochastic: Results can vary with different initial conditions.

56

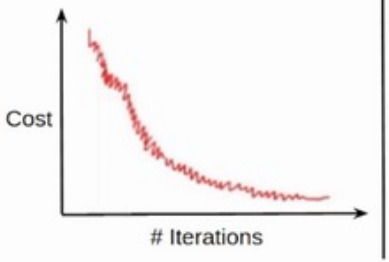
Wh

Aspect	Batch Gradient Descent	Stochastic Gradient Descent (SGD)
Data Processing	Uses the whole training dataset to compute the gradient.	Uses a single training sample to compute the gradient.
Convergence Speed	Slower, takes longer to converge.	Faster, converges quicker due to frequent updates.
Convergence Accuracy	More accurate, gives precise gradient	Less accurate due to noisy
Nature	Deterministic: Same result for the same initial conditions.	Stochastic: Results can vary with different initial conditions.

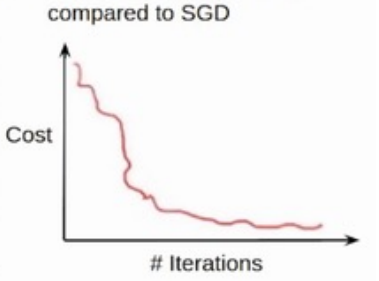
- Cost function reduces smoothly



- Lot of variations in cost function



- Smoother cost function as compared to SGD



57

Developing a Neural Network

- Needs a fair degree of mathematical competence
- Not simple (also very time consuming)
- Considerations include:
  - Identification of features in the problem domain worth mapping into inputs/outputs (Problem Representation)
  - Determining the number of hidden layers and nodes
  - Setting up initial parameters – weights, thresholds, increments
  - What learning method to use?
- No rules exist, based on experience (and solid understanding of problem domain)

58

Weakness of Neural Networks

- Neural networks are *opaque* – it is very hard to check that the weights after training are sensible
- A neural network can never explain how it arrived at a particular conclusion
- Loss of human control, lack of trust since AI cannot explain all decisions and actions, or take responsibility – **Black Box AI**
- Some work is being done on combating the opaqueness problem
- Partnership on AI (https://www.partnershiponai.org) is a step towards safe and trustworthy AI technologies – supported by tech giants including Google, IBM and Microsoft

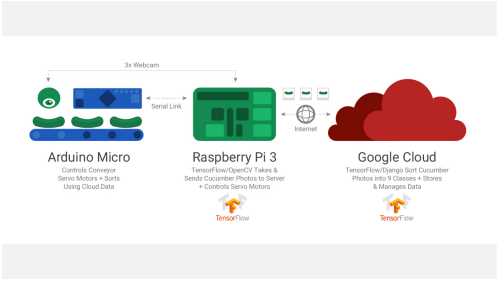
59

Machine Learning as a Service

- Machine learning is no longer for experts only
- Software developers can use cutting-edge, commercially usable machine learning frameworks & related libraries:
  - Scikit-learn
  - Google Tensorflow, Keras
  - PyTorch
  - Microsoft Cognitive Toolkit
  - Hugging Face Transformers
  - FastAI
  - NumPy, Pandas, Matplotlib, Seaborn, etc.
- Major cloud providers including Amazon, Google and Microsoft offer machine learning as a service

60

Cucumber Farming & TensorFlow



- Cucumber classification into 9 different classes using TensorFlow based on size, thickness, colour, texture, small scratches, whether they are crooked, whether they have prickles, etc.

https://cloud.google.com/blog/products/ai-machine-learning/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow

61