Imperial College London

# Asking Questions in Deep Learning

## Mathematical Approaches to Neural Networks

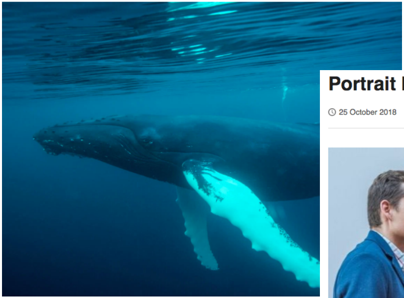
Tyler Farghly

# Introduction

**Google AI listens to 15 years of sea-bottom recordings for hidden whale songs**

Devin Coldewey @techcrunch / 3 weeks ago

Comment



**Artificial Intelligence 'outsmarts cancer'**

By James Gallagher
Health editor, BBC News website

8 June 2016

Share

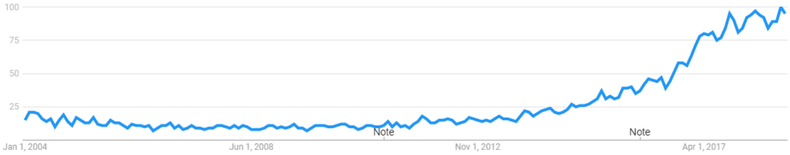

**Portrait by AI program sells for $432,000**

25 October 2018

Share

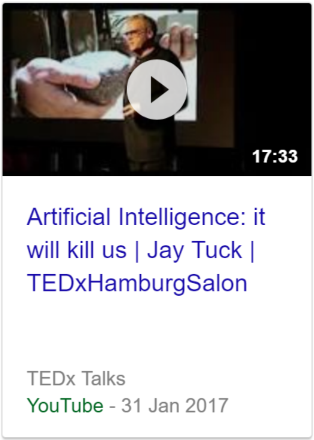

Source: bbc.co.uk, techcrunch.com

# Introduction



# Introduction

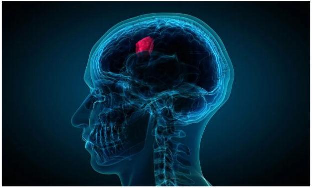
# Introduction




# Introduction

**'It's going to create a revolution': how AI is transforming the NHS**

Technology is making impressive inroads into cancer treatment, saving lives and money



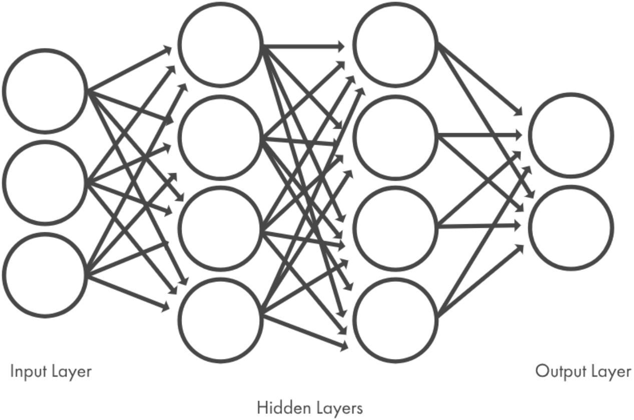
▲ Time-consuming tasks such as delineating tumours can be done in minutes with AI Photograph: Firstsignal/Getty Images/iStockphoto

# How can we fix this?

How can we fix this?
**Maths!**

# What is a Neural Network

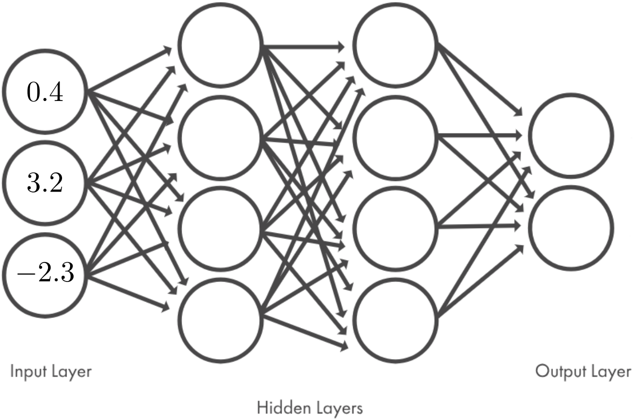
*The Deep Neural Network*



# What is a Neural Network

*The Deep Neural Network*

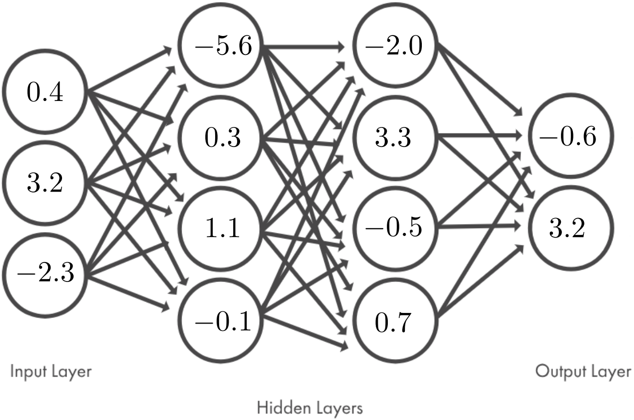
$$\begin{pmatrix} 0.4 \\ 3.2 \\ -2.3 \end{pmatrix} \rightarrow$$



# What is a Neural Network

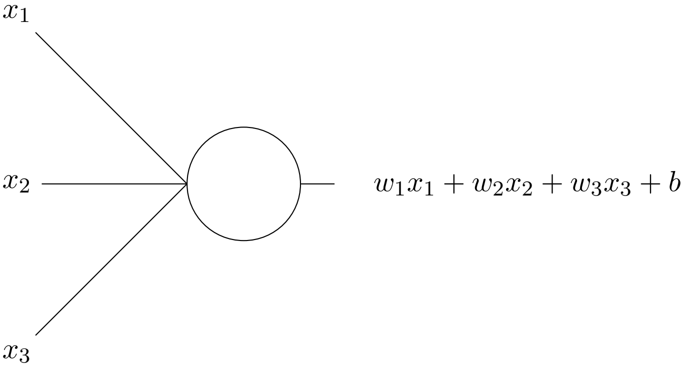
## *The Deep Neural Network*

$$\begin{pmatrix} 0.4 \\ 3.2 \\ -2.3 \end{pmatrix} \rightarrow$$



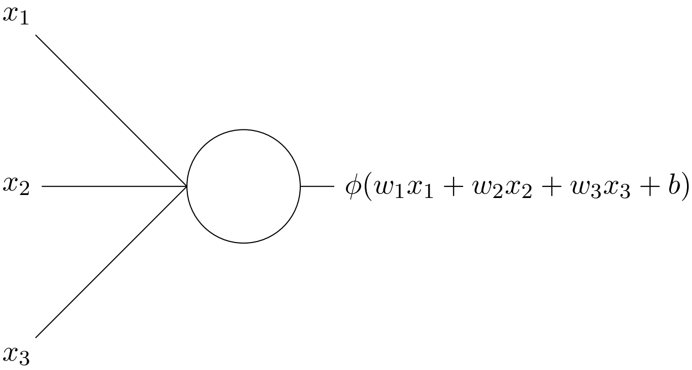
# What is a Neural Network

*The Perceptron*



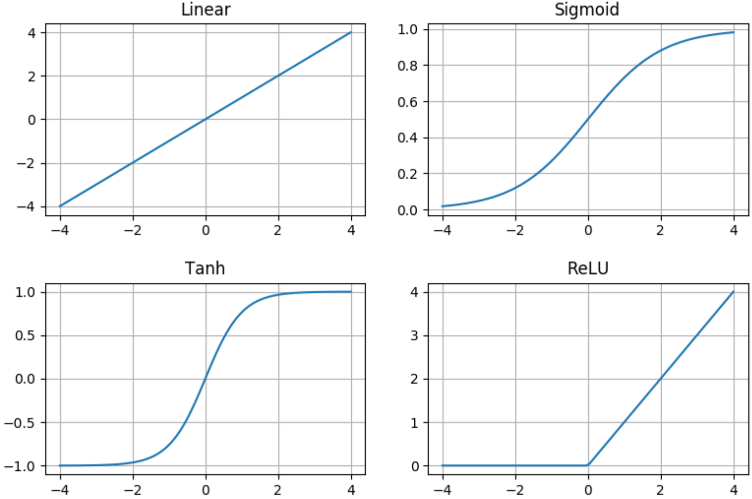
## What is a Neural Network

*The Perceptron*



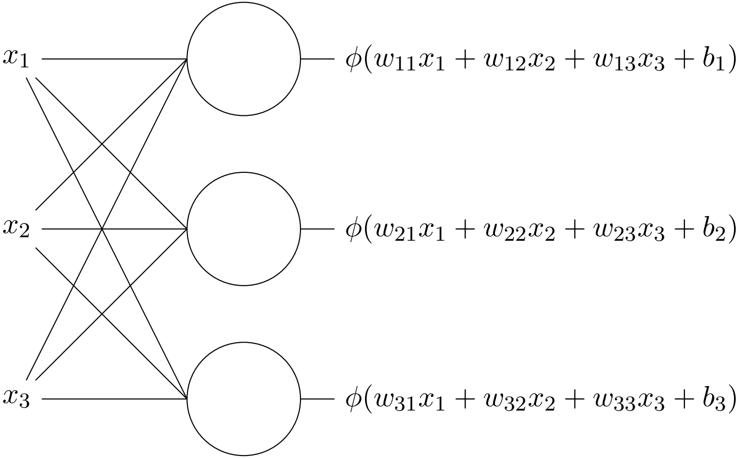
# What is a Neural Network

## *Activation Functions*



# What is a Neural Network

*The Perceptron*



# What is a Neural Network

*Multilayer Perceptron Networks*

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1 \\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2 \\ w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3 \end{pmatrix}$$

# What is a Neural Network

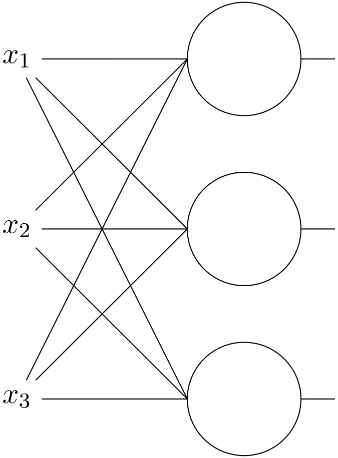
*Multilayer Perceptron Networks*

$$\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1 \\ w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + b_2 \\ w_{31}x_1 + w_{32}x_2 + w_{33}x_3 + b_3 \end{pmatrix}$$

$$\mathbf{W}\mathbf{x} + \mathbf{b}$$

# What is a Neural Network

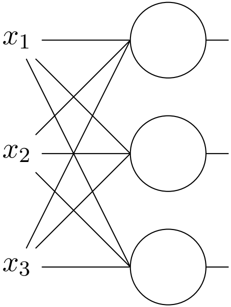
*Multilayer Perceptron Networks*




$$\phi\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

# What is a Neural Network

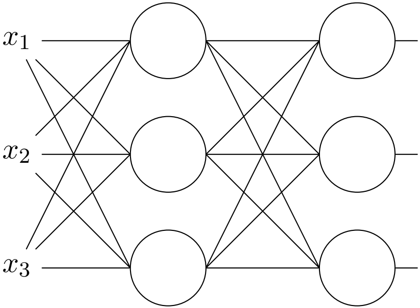
*Multilayer Perceptron Networks*




$$\phi\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right)$$

# What is a Neural Network

*Multilayer Perceptron Networks*




$$\phi\Big(\mathbf{W}_2\,\phi\left(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1\right)+\mathbf{b}_2\Big)$$

# What is a Neural Network

*Multilayer Perceptron Networks*

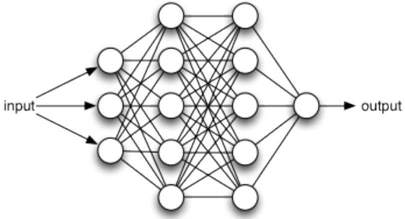
$$l_i(\mathbf{v}) = \phi\left(\mathbf{W}_i\mathbf{v} + \mathbf{b}_i\right)$$

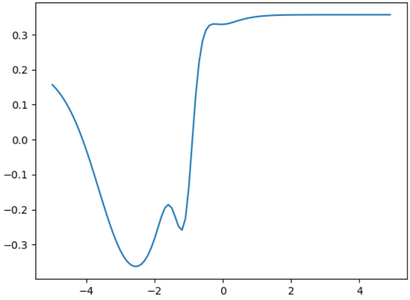
$$N(\mathbf{x}) = l_K \circ ... \circ l_1(\mathbf{x})$$

Let $\mathcal{W}$ denote the set of all weights, $\mathcal{B}$ denote the set of all biases in the NN

# What is a Neural Network

*Example*



# What is a Neural Network

## *Training Data*

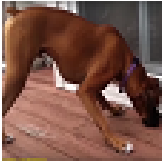
We denote training data by

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{d}$$

Example:

$$\mathbf{x}_1 = \begin{pmatrix} 0.7 \\ \vdots \\ 0.1 \end{pmatrix} \rightarrow$$

$$\mathbf{x}_2 = \begin{pmatrix} 0.4 \\ \vdots \\ 0.9 \end{pmatrix} \rightarrow$$



# What is a Neural Network

*Training Data*

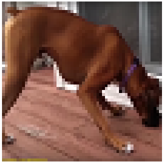
We denote training data by

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{d}$$

Example:

$$\mathbf{x}_1 = \begin{pmatrix} 0.7 \\ \vdots \\ 0.1 \end{pmatrix} \rightarrow$$

$$\mathbf{x}_2 = \begin{pmatrix} 0.4 \\ \vdots \\ 0.9 \end{pmatrix} \rightarrow$$



$$\mathbf{y}_1 = 1 \qquad \mathbf{y}_2 = 0$$

# What is a Neural Network

*Training and Evaluation*

Given data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{d}$

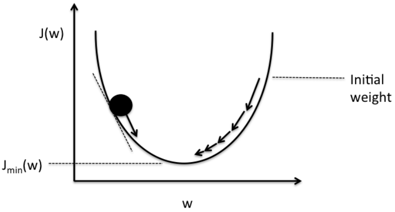
$$L(N, \mathcal{D}) = \frac{1}{d} \sum_{i=1}^{d} \|\mathbf{y}_i - N(\mathbf{x}_i)\|^2$$ – mean squared error

# What is a Neural Network

*Training and Evaluation*

Aim to find $\mathcal{W}, \mathcal{B}$ such that $L$ is minimised.

Achieve this via SGD and backpropagation.




**Schematic of gradient descent.**

# What is a Neural Network

*The Complete Process*

- We have a NN with parameters $\mathcal{W}$ and $\mathcal{B}$
- Adjust these parameters such that $L$ is minimum given our data
- Use this new NN to obtain our prediction given $\mathbf{x}$, $N(\mathbf{x})$

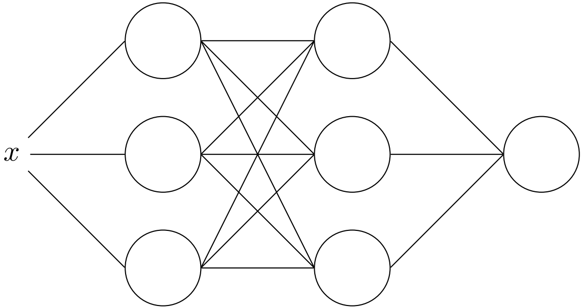
# What is a Neural Network

*Black Box Problem*


"A black box is a device, system or object which can be viewed in terms of its inputs and outputs, without any knowledge of its internal workings"

# What is a Neural Network

*So why are they so mysterious?*



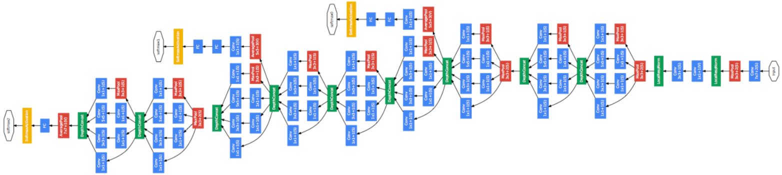
# What is a Neural Network

*So why are they rubbish?*

$$
\begin{aligned}
N(x) =&\phi(w\phi(w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx+b)+b)\\
&+w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx+b)+b)+w\phi(w\phi(wx+\\
&w\phi(wx+b)+w\phi(wx+b)+b)+b)+w\phi(w\phi(w\phi(wx+b)+w\phi(wx\\
&+b)+w\phi(wx+b)+b)+w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx\\
&+b)+b)+w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx+b)+b)+b)\\
&+w\phi(w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx+b)+b)+\\
&w\phi(w\phi(wx+b)+w\phi(wx+b)+w\phi(wx+b)+b)+w\phi(w\phi(wx+b)\\
&+w\phi(wx+b)+w\phi(wx+b)+b)+b)+b)
\end{aligned}
$$

# What is a Neural Network

*So why are they rubbish?*



# What is a Neural Network

*Why are they great?*

**Theorem 2.3.1 (universal approximation theorem):**
*Let $\varphi(.)$ be an arbitrary activation function. Let $X \subseteq \mathrm{R}^m$ and $X$ is compact. The space of continuous functions on $X$ is denoted by $C(X)$. Then $\forall f \in C(X)$, $\forall \varepsilon > 0$: $\exists n \in \mathrm{N}$, $a_{ij}$, $b_i$, $w_i \in \mathrm{R}$, $i \in \{1..n\}$, $j \in \{1..m\}$:*

$$(A_n f)(x_1, \ldots, x_m) = \sum_{i=1}^{n} w_i \varphi(\sum_{j=1}^{m} a_{ij} x_j + b_i)$$

*as an approximation of the function $f(.)$; that is*

$$\|f - A_n f\| < \varepsilon$$

# The Learning Process

*Introduction*

$$\frac{1}{d}\sum_{i=1}^{d}(Y_i - N(X_i))^2 \to E\left[(Y - N(X))^2\right]$$

as $d \to \infty$

# The Learning Process

*Bias/Variance Tradeoff*

$$\mathcal{D} \rightarrow \mathcal{W}, \mathcal{B} \rightarrow N(\mathbf{x})$$

# The Learning Process

*Bias/Variance Tradeoff*

$$\mathcal{D} \to \mathcal{W}, \mathcal{B} \to N(\mathbf{x})$$

$$E_{\mathcal{D}} \left( E \left[ (y - N(\mathbf{x}))^2 \right] \right) = E \Bigg[ (y - E_{\mathcal{D}}[N(\mathbf{x})])^2 + Var_{\mathcal{D}}(N(\mathbf{x})) \Bigg]$$

# The Learning Process

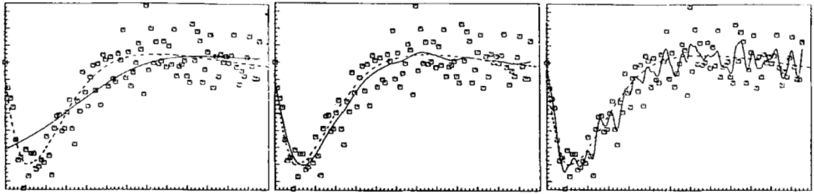
*Bias/Variance Tradeoff*

$$\mathcal{D} \rightarrow \mathcal{W}, \mathcal{B} \rightarrow N(\mathbf{x})$$

$$E_{\mathcal{D}}\left(E\left[(y - N(\mathbf{x}))^2\right]\right) = E\Big[\underbrace{(y - E_{\mathcal{D}}[N(\mathbf{x})])^2}_{\text{squared bias}} + \underbrace{Var_{\mathcal{D}}(N(\mathbf{x}))}_{\text{variance}}\Big]$$

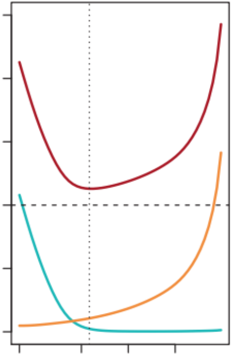
# The Learning Process

## *Bias/Variance Tradeoff*



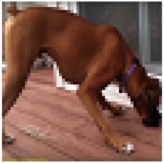
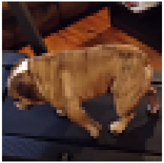
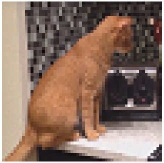
# The Learning Process

*Bias/Variance Tradeoff*



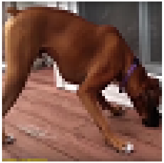
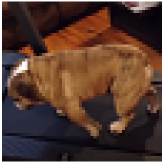
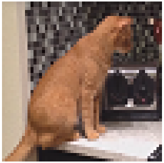
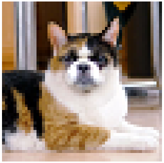
# The Learning Process

*Example: Classification*

| 1 | 0 | 0 | 1 |
|---|---|---|---|
|  |  |  |  |
| 0.93 | 0.01 | 0.08 | 0.96 |

## The Learning Process

*Example: Classification*

| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
|  |  |  |  |  |
| 0.93 | 0.01 | 0.08 | 0.96 | 0.42 |

# The Learning Process

*Example: Classification*

Split the input space into volumes $V(v)$:

$V(v)$ – volume of size $\mathrm{d}v$ and centered at a point $v$

# The Learning Process

*Example: Classification*

Split the input space into volumes $V(v)$:

$V(v)$ – volume of size $\mathrm{d}v$ and centered at a point $v$

By continuity, $N(v) = N(x) \, \forall x \in V(v)$

# The Learning Process

*Example: Classification*

Let $p(v) := p(y = 1|x \in V(v))$

$$E_{V(v)}\left[(y - N(x))^2\right] = [1 - N(v)]^2 p(v) + N(v)^2[1 - p(v)]$$

# The Learning Process

*Example: Classification*

$$E\left[(y - N(x))^2\right] = \sum_V \left[(1 - N(v))p(v) + N(v)^2(1 - p(v))\right] \mathrm{d}v$$

# The Learning Process

*Example: Classification*

$$E\left[(y - N(x))^2\right] = \sum_V \left[(1 - N(v))p(v) + N(v)^2(1 - p(v))\right] \mathrm{d}v$$

Differentiating each term w.r.t. $N(v)$ and setting to $0$ gives

$$-2p(v)[1 - N(v)] + 2(1 - p(v))N(v) = 0$$

From this we deduce $p(v) = N(v)$ minimises the error in any $V(v)$.

# The Learning Process

*Generalisation Error*

Suppose we are creating a classifier, i.e. $y \in \{0, 1\}$.

The VC dimension, $h$, is the largest number of data points we can always expect the NN to fit to.

# The Learning Process

*Generalisation Error*

Suppose we are creating a classifier, i.e. $y \in \{0, 1\}$.

The VC dimension, $h$, is the largest number of data points we can always expect the NN to fit to.

$$\nu = \text{average rate of error}$$
$$\nu_{emp} = \text{rate of error on training data}$$

# The Learning Process

*Generalisation Error*

If $\nu(\mathcal{W}, \mathcal{B})$ and $\nu_{emp}(\mathcal{W}, \mathcal{B})$ are sufficiently close to each other with probability $\alpha$ then

$$\nu(\mathcal{W}, \mathcal{B}) < \nu_{emp}(\mathcal{W}, \mathcal{B}) + \varepsilon_1(h, d, \alpha)$$

# The Learning Process

*Generalisation Error*

If $\nu(\mathcal{W},\mathcal{B})$ and $\nu_{emp}(\mathcal{W},\mathcal{B})$ are sufficiently close to each other with probability $\alpha$ then

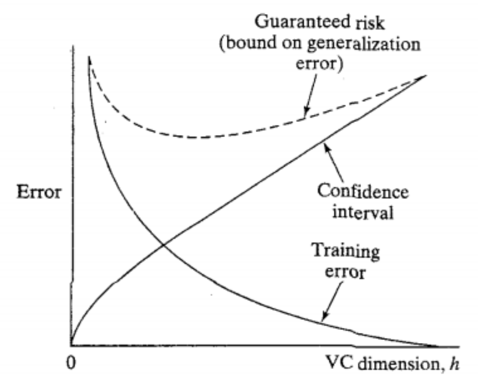
$$\nu(\mathcal{W},\mathcal{B}) < \nu_{emp}(\mathcal{W},\mathcal{B}) + \varepsilon_1(h,d,\alpha)$$

$$\varepsilon_1(h,d,\alpha) = 2\varepsilon_0^2\left(1+\sqrt{1+\frac{\nu_{emp}(\mathcal{W},\mathcal{B})}{\varepsilon_0^2}}\right)$$

$$\varepsilon_0 = \sqrt{\frac{h}{d}\left[\ln\frac{2d}{h}+1\right]-\frac{1}{d}\ln\alpha}$$

# The Learning Process

*Generalisation Error*



# The Learning Process

## *Generalisation Error*

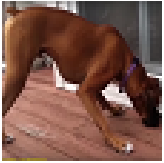
$$\nu_{gene}(\mathcal{W}, \mathcal{B}) = \text{rate of error on new data}$$

$$P(\nu_{gene}(\mathcal{W}, \mathcal{B}) < \nu_{train}(\mathcal{W}, \mathcal{B}) + \varepsilon_1) > 1 - \alpha$$

# Bayesian Neural Networks

*Introduction*

$1$ $\quad$ $0$

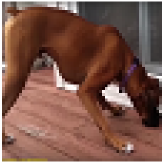


$0.93$ $\quad$ $0.01$

# Bayesian Neural Networks

*Introduction*

$1$ $\qquad$ $0$

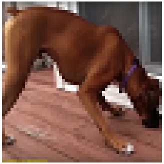


$0.93$ $\qquad$ $0.01$ $\qquad$ $0.78$

# Bayesian Neural Networks

*Introduction*

$1$ $\quad$ $0$

$0.93$ $\quad$ $0.01$ $\quad$ $0.78$

No notion of uncertainty!

# Bayesian Neural Networks

*Introduction*

"Bayesian learning is distinguished by its use of probability to express all forms of uncertainty"[RADFORD]

$$\mathcal{W} \quad \mathcal{B}$$

# Bayesian Neural Networks

*Introduction*

$$p(\mathcal{W}, \mathcal{B})$$

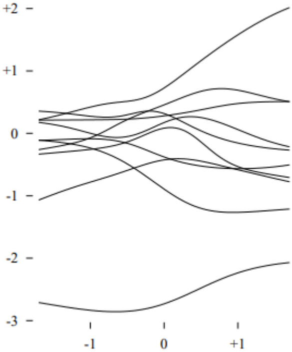
# Bayesian Neural Networks

*Introduction*

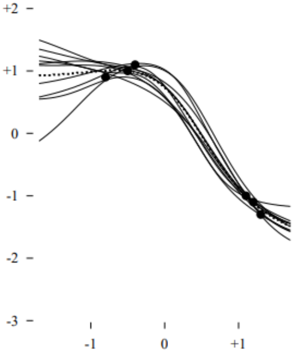
$$p(\mathcal{W}, \mathcal{B}) \rightarrow p(\mathcal{W}, \mathcal{B}|\mathcal{D})$$

# Bayesian Neural Networks

*Example*




$\mathcal{W}, \mathcal{B}$




$\mathcal{W}, \mathcal{B} \mid \mathcal{D}$

# Bayesian Neural Networks

*Posterior Distribution*

$$p(\mathcal{W}, \mathcal{B}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{W}, \mathcal{B})p(\mathcal{W}, \mathcal{B})}{\int p(\mathcal{D}|\mathcal{W}', \mathcal{B}')\mathrm{d}\mathcal{W}'\,\mathrm{d}\mathcal{B}'}$$

$$\propto p(\mathcal{D}|\mathcal{W}, \mathcal{B})p(\mathcal{W}, \mathcal{B})$$

# Bayesian Neural Networks

*Uncertainty*

Suppose $\mathbf{y}_{pred}$ is the prediction obtained from the BNN.

We can evaluate the uncertainty of such a prediction by analysing the distribution of $\mathbf{y}|\mathbf{x}, \mathcal{D}$ around $\mathbf{y} = \mathbf{y}_{pred}$

# Bayesian Neural Networks

*Uncertainty*

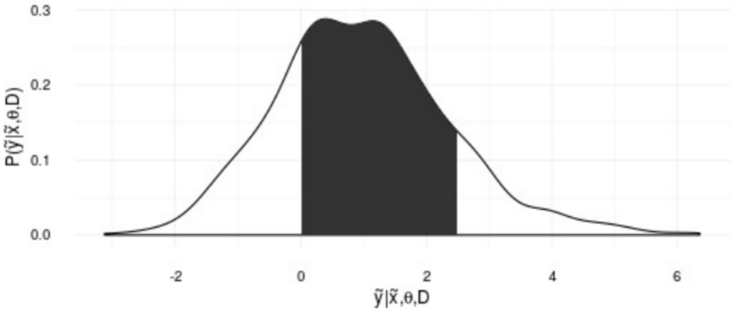
Suppose $\mathbf{y}_{pred}$ is the prediction obtained from the BNN.

We can evaluate the uncertainty of such a prediction by analysing the distribution of $\mathbf{y}|\mathbf{x}, \mathcal{D}$ around $\mathbf{y} = \mathbf{y}_{pred}$

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \mathcal{W}, \mathcal{B}) \, p(\mathcal{W}, \mathcal{B}|\mathcal{D}) \, \mathrm{d}\mathcal{W} \, \mathrm{d}\mathcal{B}$$

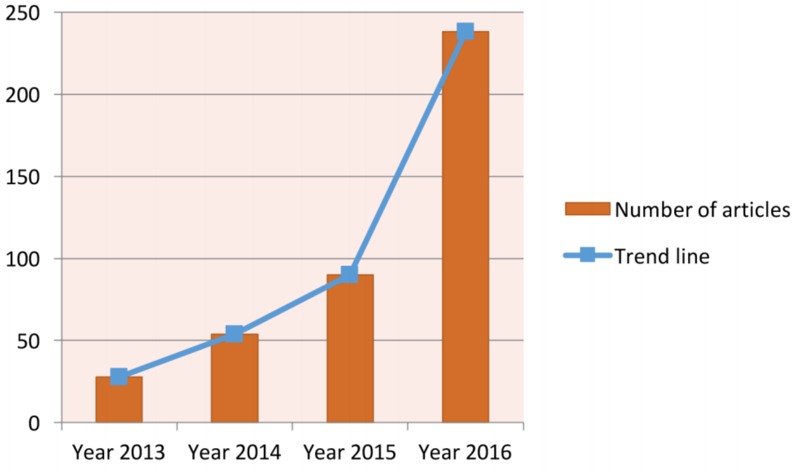
# Bayesian Neural Networks

*Uncertainty*



# Bayesian Neural Networks

*Uncertainty*



# Bayesian Neural Networks

## Traditional approach

- We have a NN with parameters $\mathcal{W}$ and $\mathcal{B}$
- Adjust these parameters such that $L$ is minimum given our data
- Use this new NN to obtain our prediction given $\mathbf{x}$, $N(\mathbf{x})$

# Bayesian Neural Networks

## Traditional approach

- We have a NN with parameters $\mathcal{W}$ and $\mathcal{B}$
- Adjust these parameters such that $L$ is minimum given our data
- Use this new NN to obtain our prediction given $\mathbf{x}$, $N(\mathbf{x})$

## Bayesian approach

- Set the prior distributions and sample from $\mathcal{W}, \mathcal{B}|\mathcal{D}$
- Average value of $N(\mathbf{x})$ over the sampled parameters is $\mathbf{y}_{pred}$
- Evaluate distribution of $\mathbf{y}|\mathbf{x}, \mathcal{D}$ to evaluate uncertainty

# Bayesian Neural Networks

*Prior Distribution*

$$w, b \sim Normal\left(0, \sigma^2\right) \quad \text{for all } w \in \mathcal{W}, b \in \mathcal{B}$$

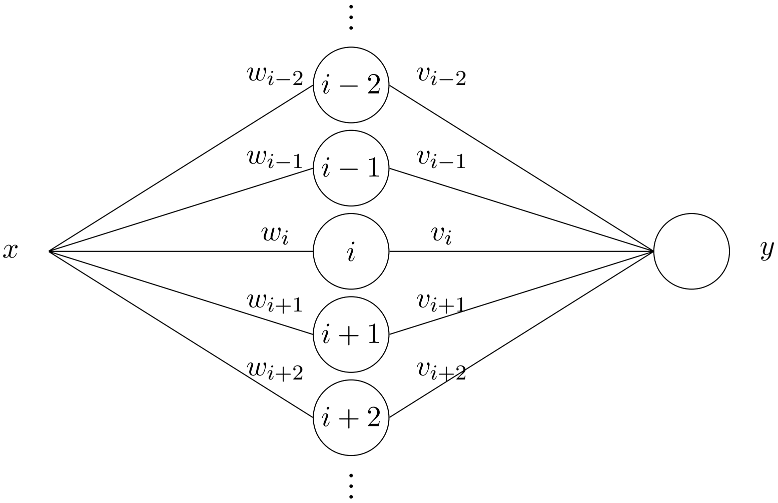
$$p(\mathcal{W}, B) = \prod p(w) \prod p(b)$$

$$-\log p(\mathcal{W}, B) = \sum \log p(w) + \sum \log p(b)$$

$$\propto \frac{1}{2\sigma^2}(|\mathcal{W}|^2 + |\mathcal{B}|^2)$$

# Bayesian Neural Networks

*Infinite Networks*



# Bayesian Neural Networks

*Infinite Networks*

$$w_i, v_i \sim N(0, H^{-1}\sigma_w^2) \quad b_i, c \sim N(0, \sigma_b^2)$$

$$N(x) = c + \sum_{i=1}^{H} v_i h_i(x)$$

$$h_i(x) = \phi\left(w_i x + b_i\right)$$

# Bayesian Neural Networks

By CLT,

$$N(\mathbf{x}) \xrightarrow{\mathcal{D}} Normal\left(0, \sigma_b^2 + \omega_w^2 V(x)\right) \quad \text{as } H \to \infty$$

# Bayesian Neural Networks

*Infinite Networks*

$$E(N(x_p), N(x_q)) = \sigma_b^2 + \sigma_w^2 C(x_p, x_q)$$

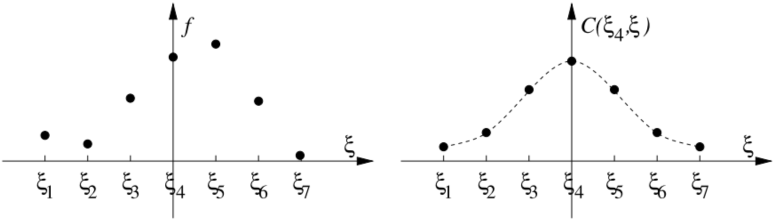
where $C(x_p, x_q) = E(h_i(x_p) h_i(x_q))$

Distributions over functions of this sort are known as

**Gaussian processes**

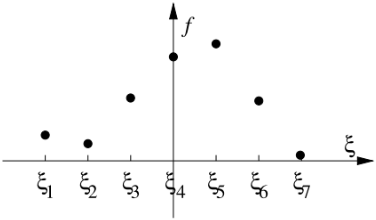
# Bayesian Neural Networks

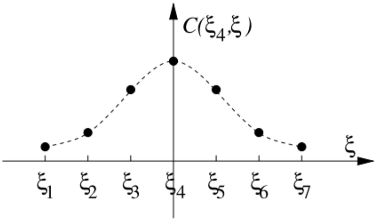
*Gaussian Process*



# Bayesian Neural Networks

*Gaussian Process*




Smooth, flexible functions! No overfitting as a result of increased flexibility

# Bayesian Neural Networks

*Pro's and Con's*

## Pro's

- Can evaluate uncertainty
- Don't need to worry about overfitting
- Can apply well-studied ideas in statistics

# Bayesian Neural Networks

*Pro's and Con's*

## Pro's

- Can evaluate uncertainty
- Don't need to worry about overfitting
- Can apply well-studied ideas in statistics

## Con's

- Very computationally expensive
- Difficult to establish a prior distribution
- Occasionally results rely heavily on choice of prior

# Information Theory in Neural Networks

# Dropout Regularization

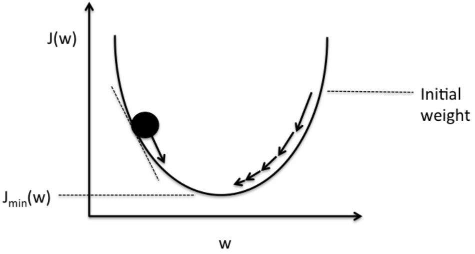
*Introduction*

A weird method to reduce overfitting

# Dropout Regularization

*Introduction*

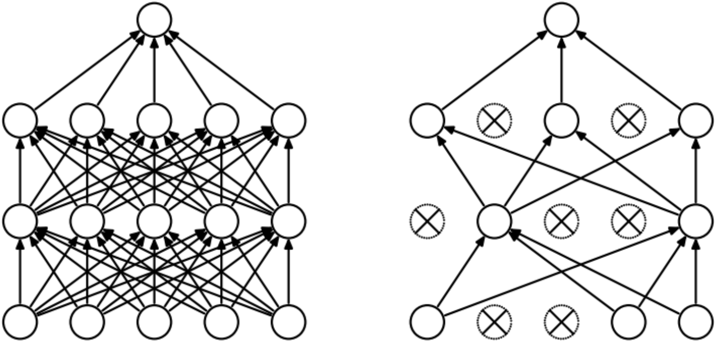
Recall SGD as our method for minimising $L$:




**Schematic of gradient descent.**

# Dropout Regularization

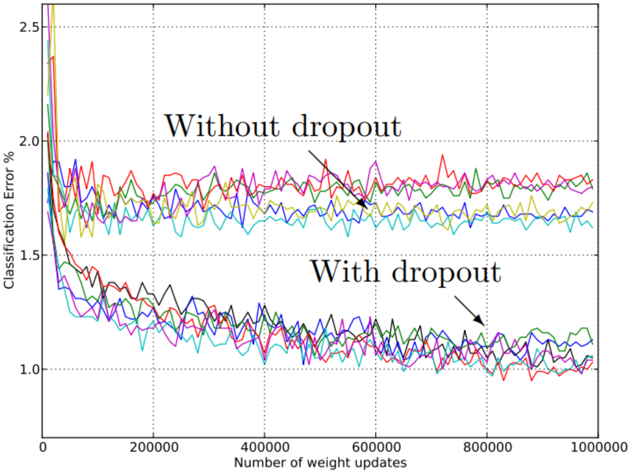
*Introduction*



(a) Standard Neural Net (b) After applying dropout.

# Dropout Regularization

*Introduction*



# Dropout Regularization

*First Paper*

## Improving neural networks by preventing co-adaptation of feature detectors

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

# Dropout Regularization

*First Paper*

Improving neural networks by preventing
co-adaptation of feature detectors

G. E. Hinton*, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov
Department of Computer Science, University of Toronto,
6 King's College Rd, Toronto, Ontario M5S 3G4, Canada

"Overfitting can be reduced by using dropout to prevent complex co-adaptations on the training data."

# Dropout Regularization

*Formulation*

If layer $i$ is a dropout layer

$$l_i(\mathbf{x}) = \frac{1}{p}\phi(W\mathbf{x} + \mathbf{b}) \circ \mathbf{r}_i$$

$$(\mathbf{r}_i)_j \sim \text{Bernoulli}(p_i)$$

# Dropout Regularization

*Suprising Property of Dropout*

Suppose we have the data set $\{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{1})\}$ and we are training a neural network of linear activation where all biases are 0 i.e.

$$N(\mathbf{x}) = W_K \dots W_1 \, \mathbf{x}$$

# Dropout Regularization

*Suprising Property of Dropout*

Suppose we have the data set $\{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{1})\}$ and we are training a neural network of linear activation where all biases are 0 i.e.

$$N(\mathbf{x}) = W_K \dots W_1\,\mathbf{x}$$

If we insert dropout into every layer, for the NN to be optimum there must be **at least one negative weight!**

# What's Next

# What’s Next

Who knows?

## Questions

https://bit.ly/2QV9vjg

https://bit.ly/2R0RWyq