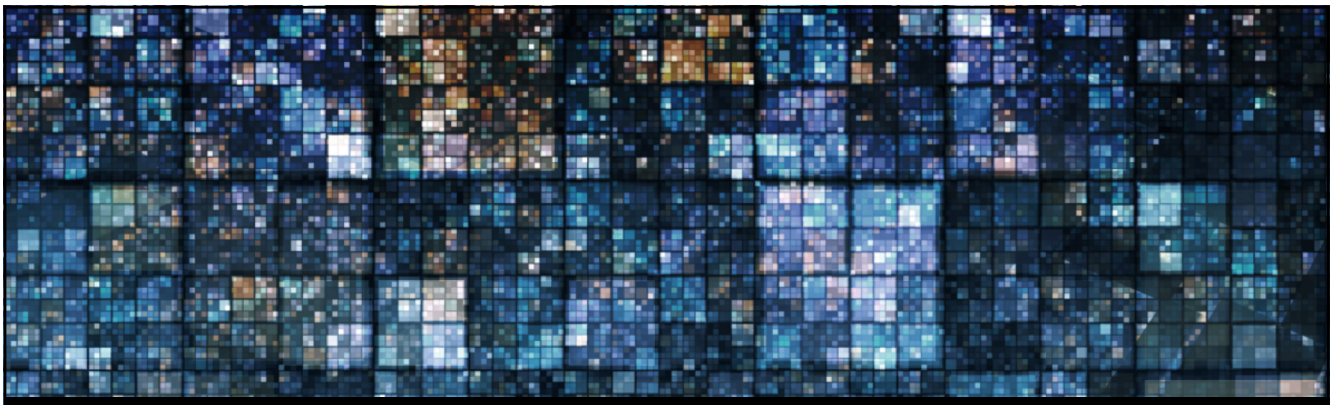


COMP47590
Advanced Machine Learning
Deep Learning - ANNs 1

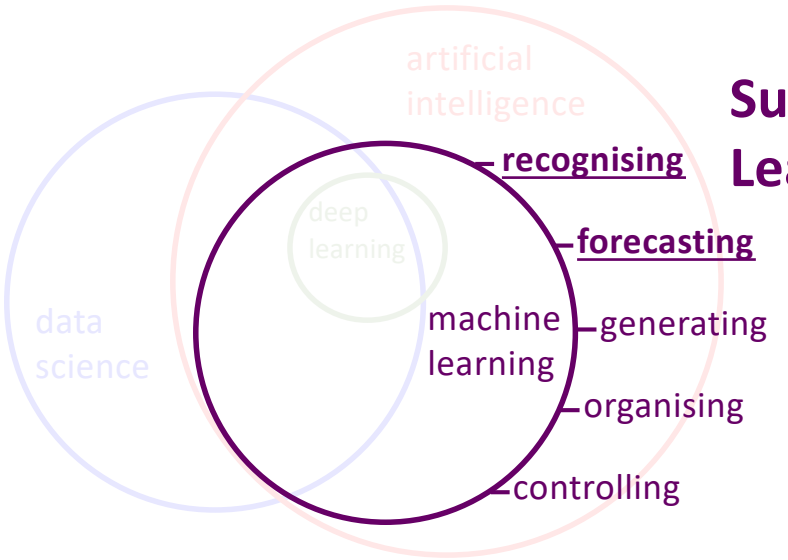
Dr. Brian Mac Namee

Information

Email: Brian.MacNamee@ucd.ie

Course Materials: All material posted on UCD CS moodle https://csmoodle.ucd.ie/moodle/course/view.php?id=756

Enrolment key **UCDAdvML2019**



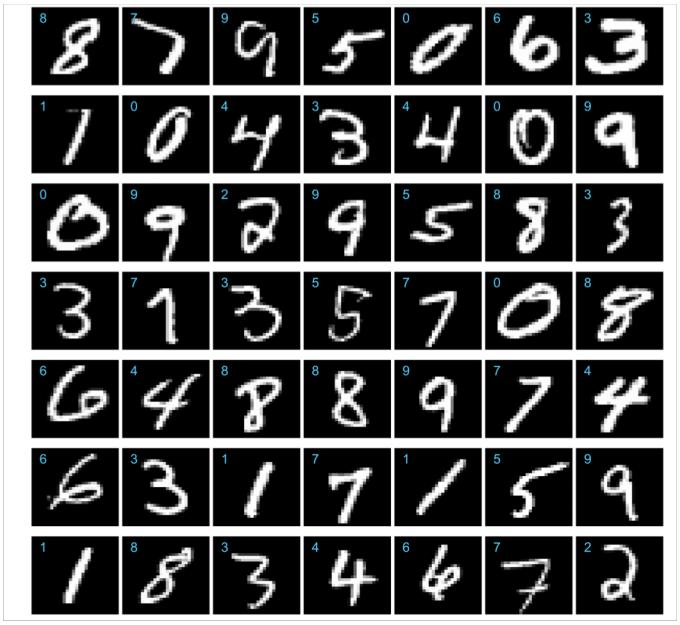
Supervised Learning

ARTIFICIAL NEURAL NETWORKS & DEEP LEARNING

Engineering representations, is one of the most important and time consuming jobs in most predictive analytics projects, and needs a blend of technical expertise and domain expertise

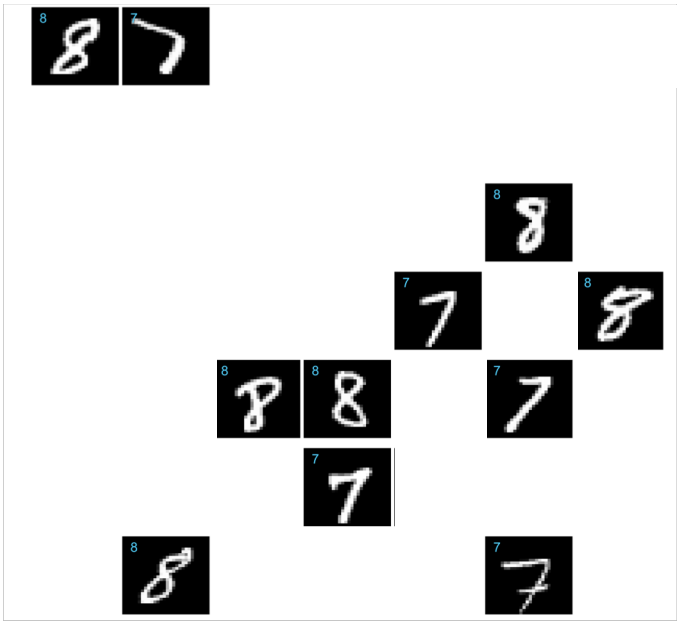
Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification

[LeCun etal, 2014]



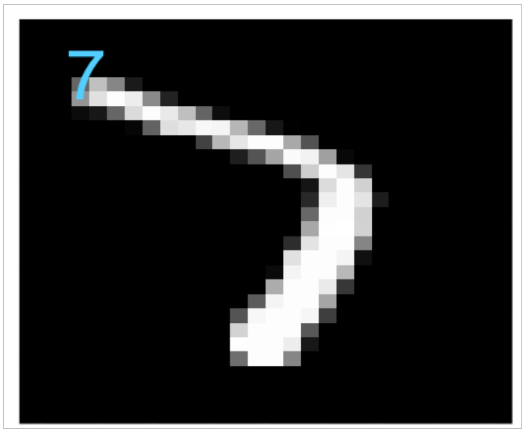
MNIST Dataset from Yan LeCun
http://yann.lecun.com/exdb/mnist/index.html

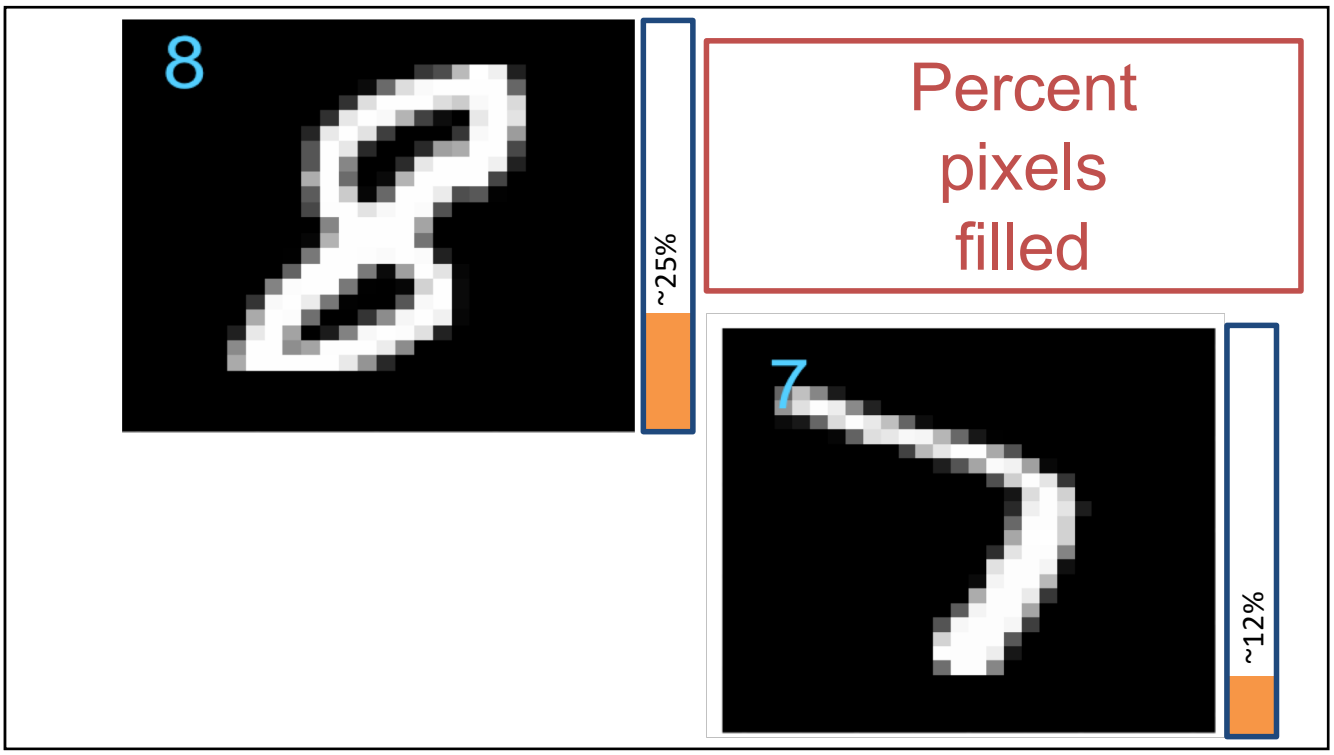
Kaggle Digit Recogniser Contest
https://www.kaggle.com/c/digit-recognizer

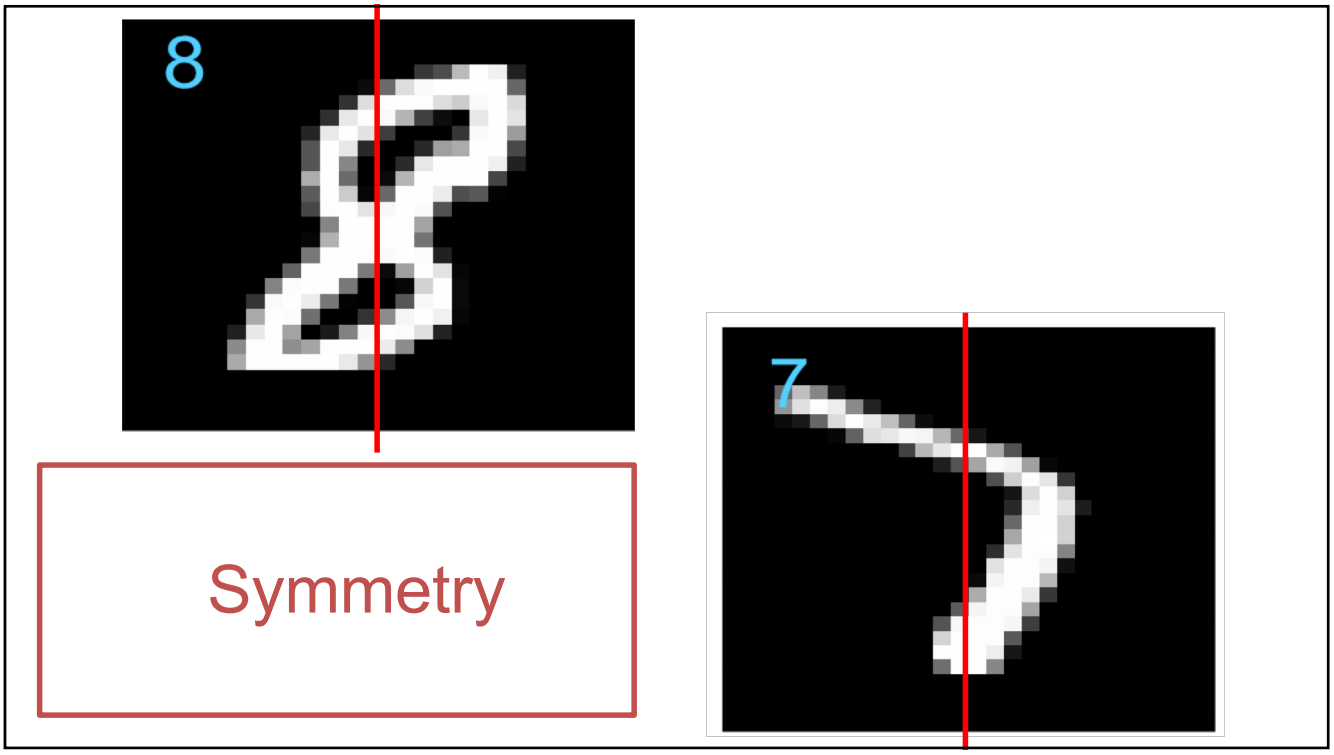


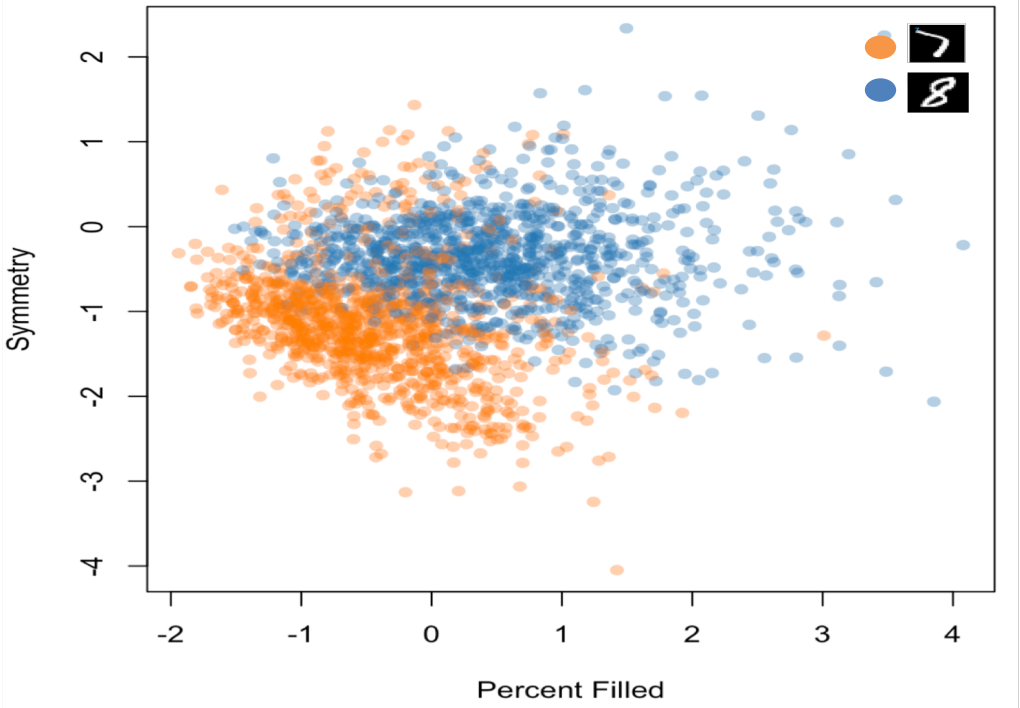
MNIST Dataset from Yan LeCun
http://yann.lecun.com/exdb/mnist/index.html

Kaggle Digit Recogniser Contest
https://www.kaggle.com/c/digit-recognizer




8
~25%
Percent pixels filled
7
~12%


8
Symmetry
7


Symmetry
Percent Filled
-4
-3
-2
-1
0
1
2
-2
-1
0
1
2
3
4

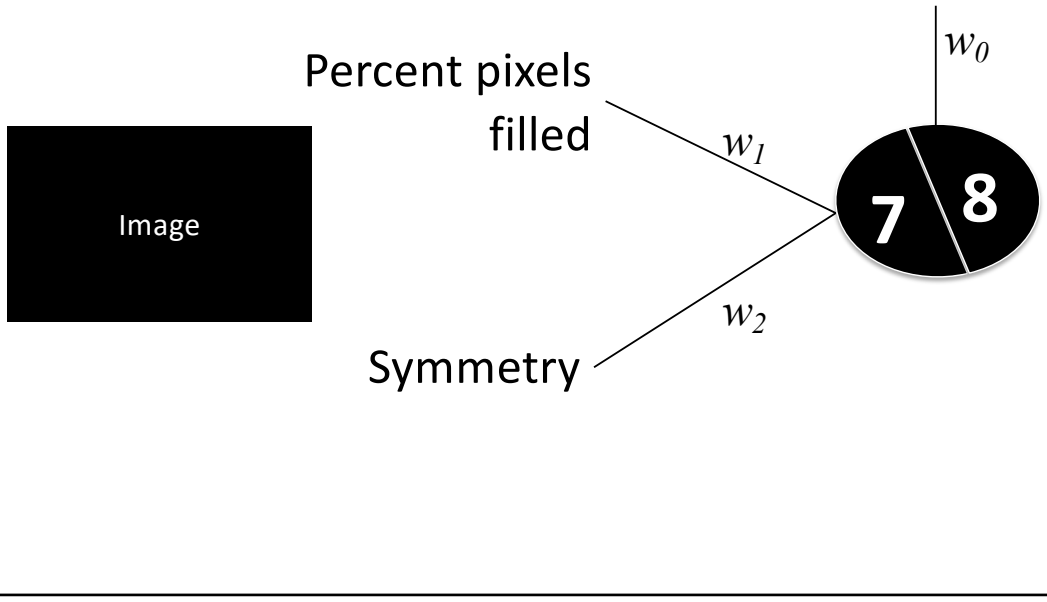
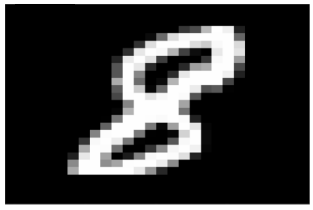
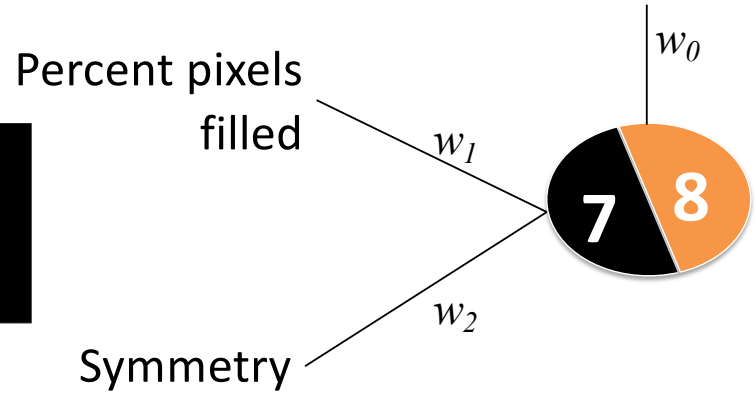
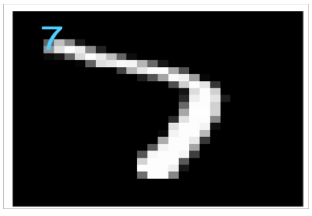
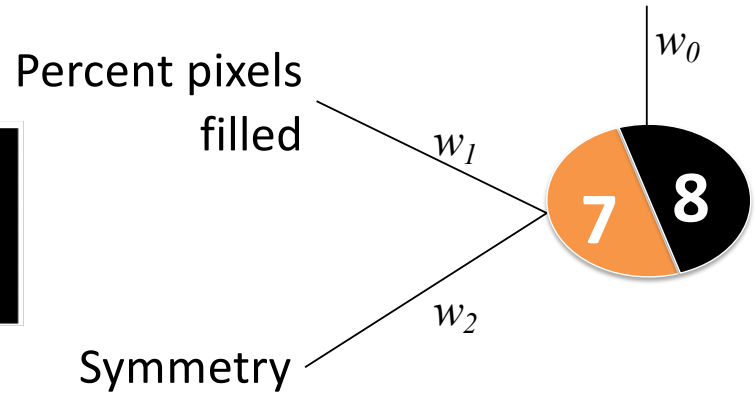


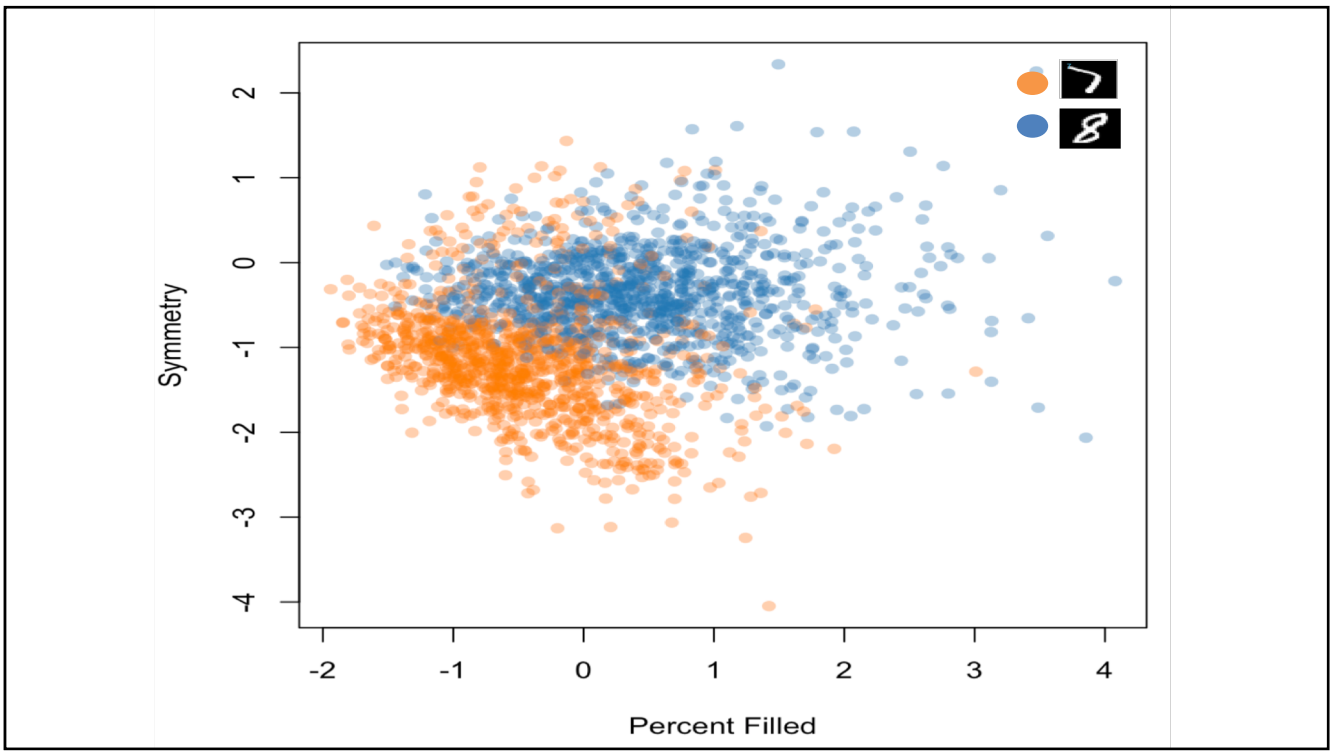
Image
Percent pixels filled
Symmetry
w_0
w_1
w_2
7
8

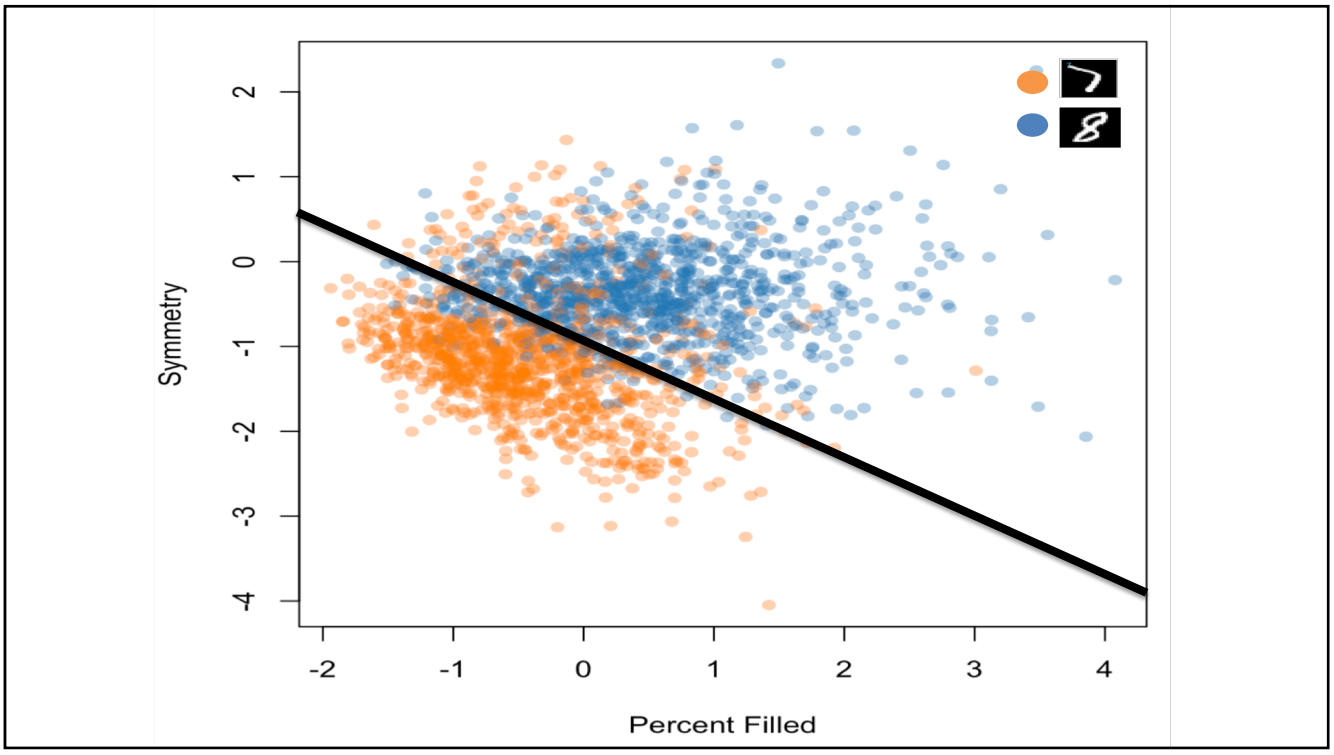


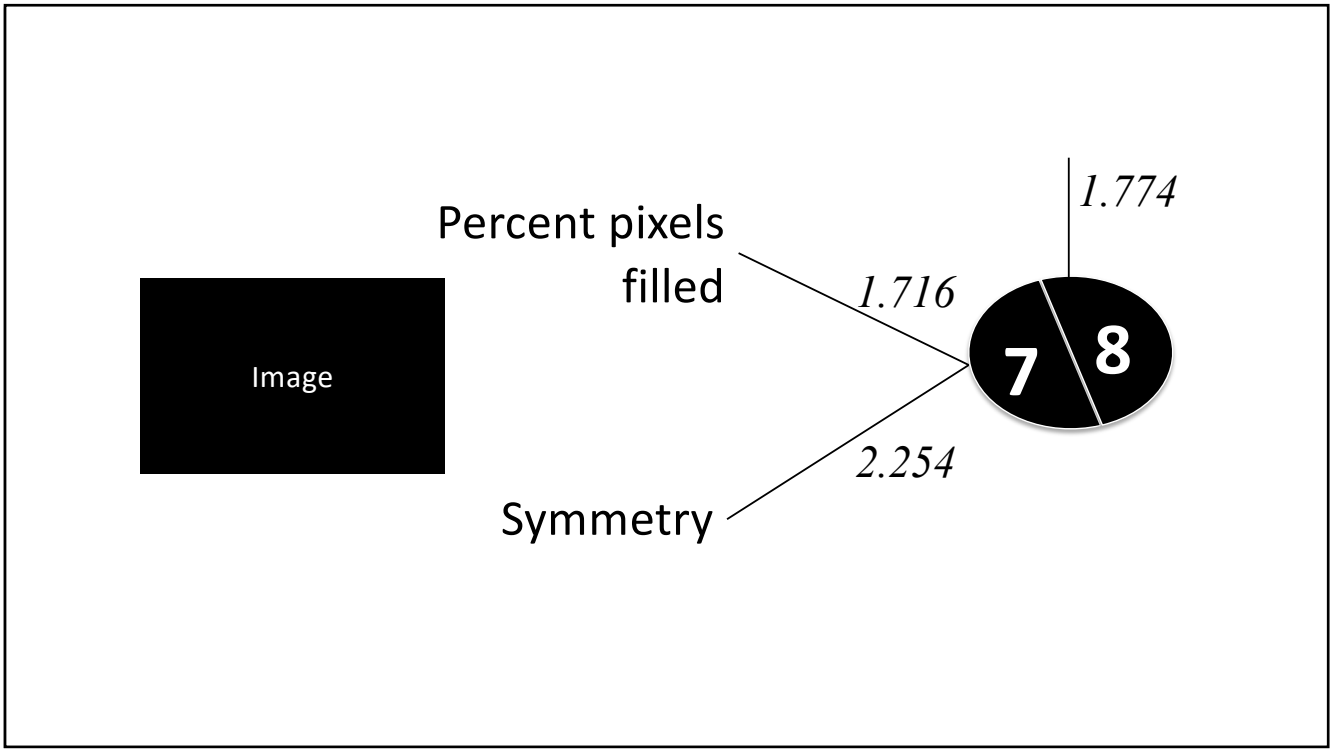

Percent pixels filled
Symmetry
w_0
w_1
w_2
7
8

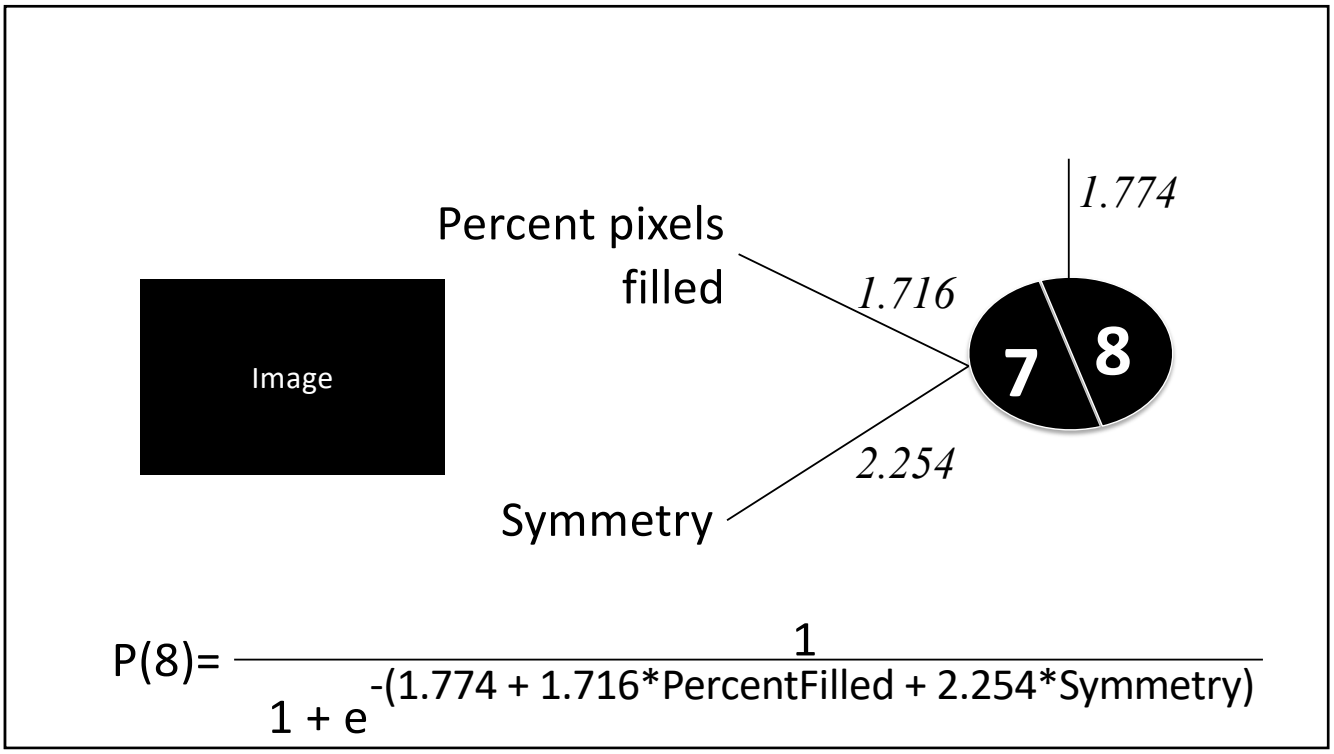



Percent pixels filled
Symmetry
w_0
w_1
w_2
7
8

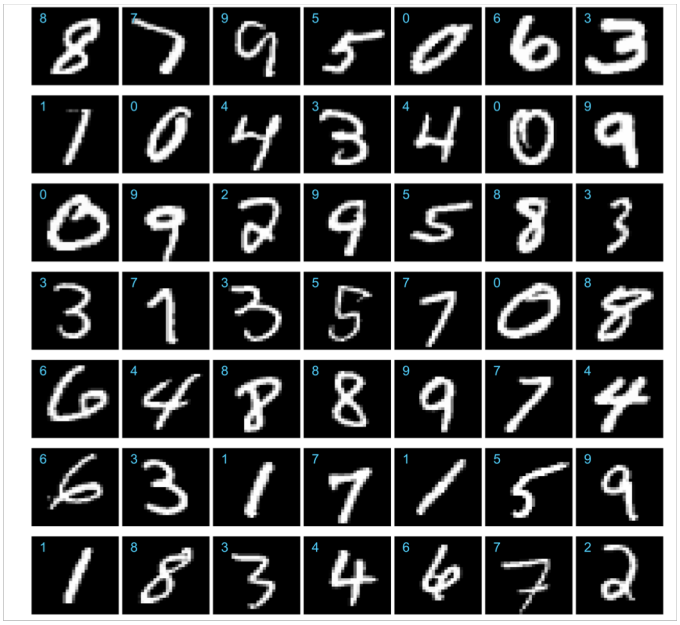

Symmetry
2
1
0
-1
-2
-3
-4
-2
-1
0
1
2
3
4
Percent Filled


Symmetry
2
1
0
-1
-2
-3
-4
-2
-1
0
1
2
3
4
Percent Filled



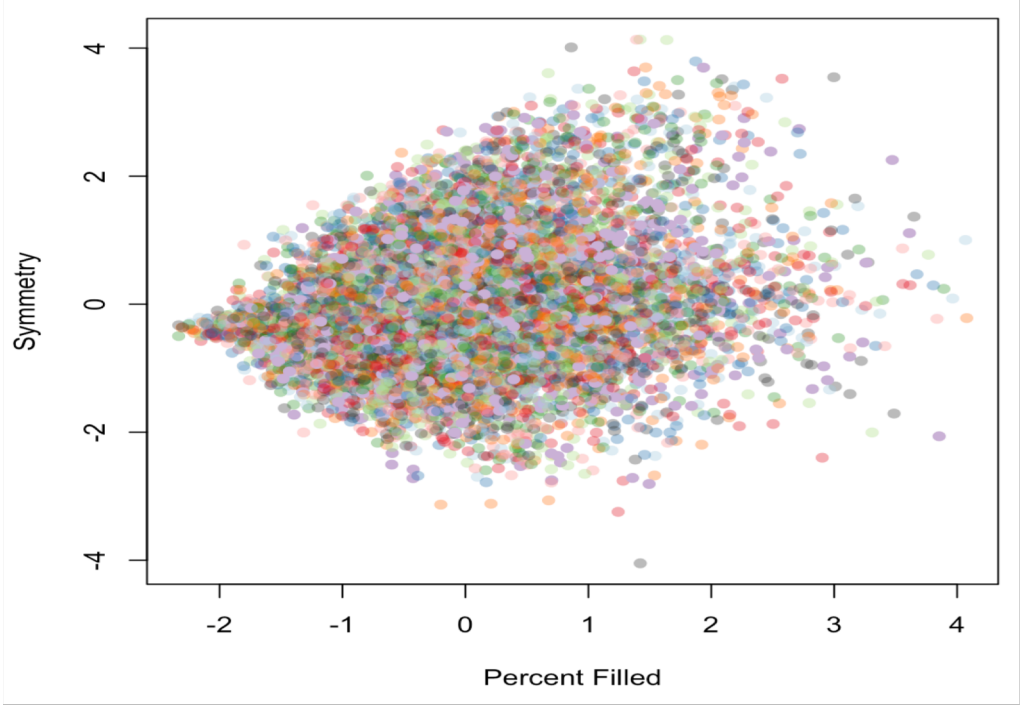


$$P(8)= \frac{1}{1 + e^{-(1.774 + 1.716*\text{PercentFilled} + 2.254*\text{Symmetry})}}$$

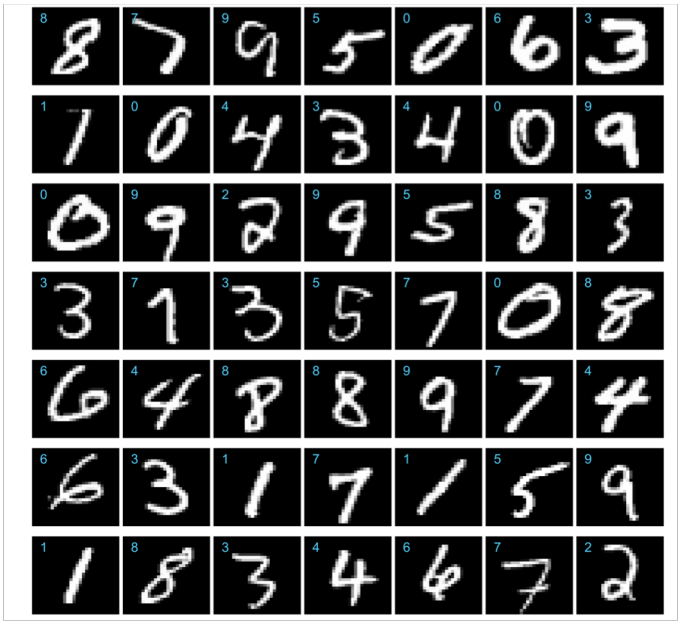


MNIST Dataset from Yan LeCun
http://yann.lecun.com/exdb/mnist/index.html

Kaggle Digit Recogniser Contest
https://www.kaggle.com/c/digit-recognizer

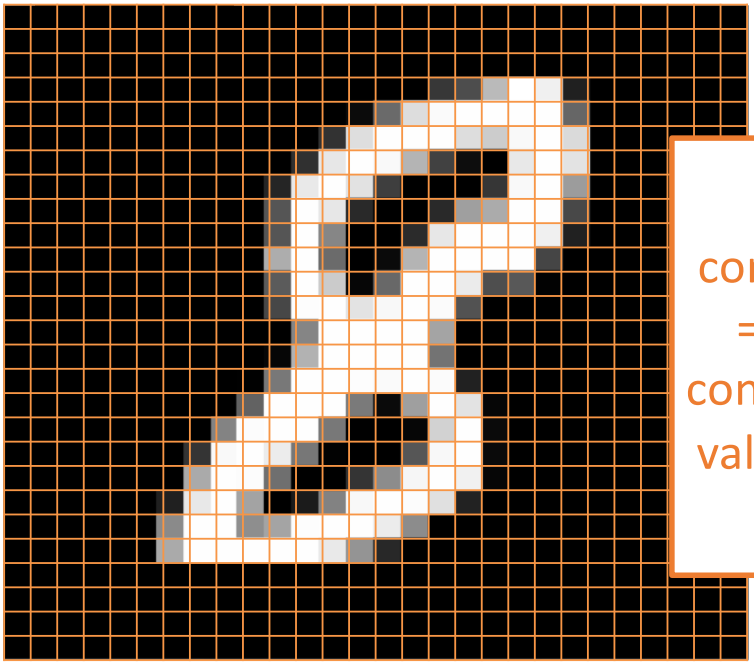


Rosenblatt's perceptron from 1957 was an early example of **representation learning**, and one of the first **artificial neural networks**



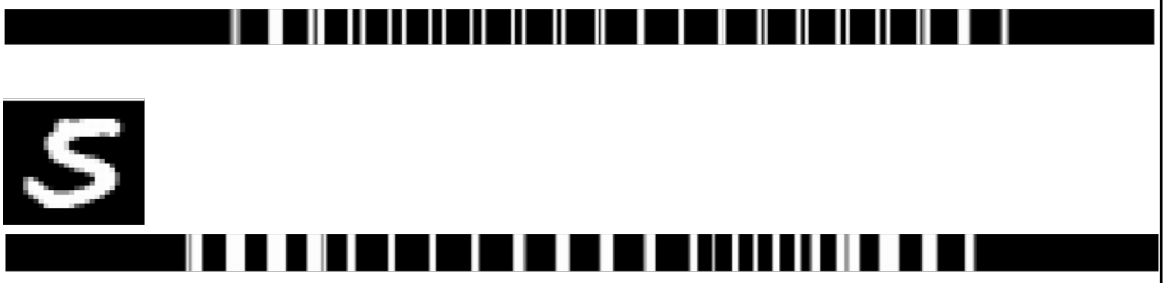
MNIST Dataset from Yan LeCun
http://yann.lecun.com/exdb/mnist/index.html

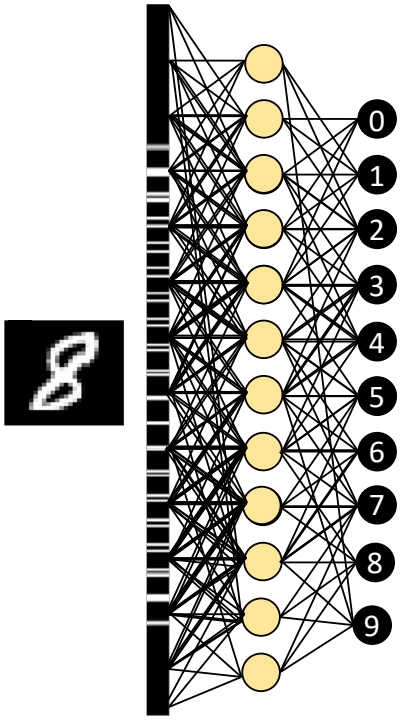
Kaggle Digit Recogniser Contest
https://www.kaggle.com/c/digit-recognizer



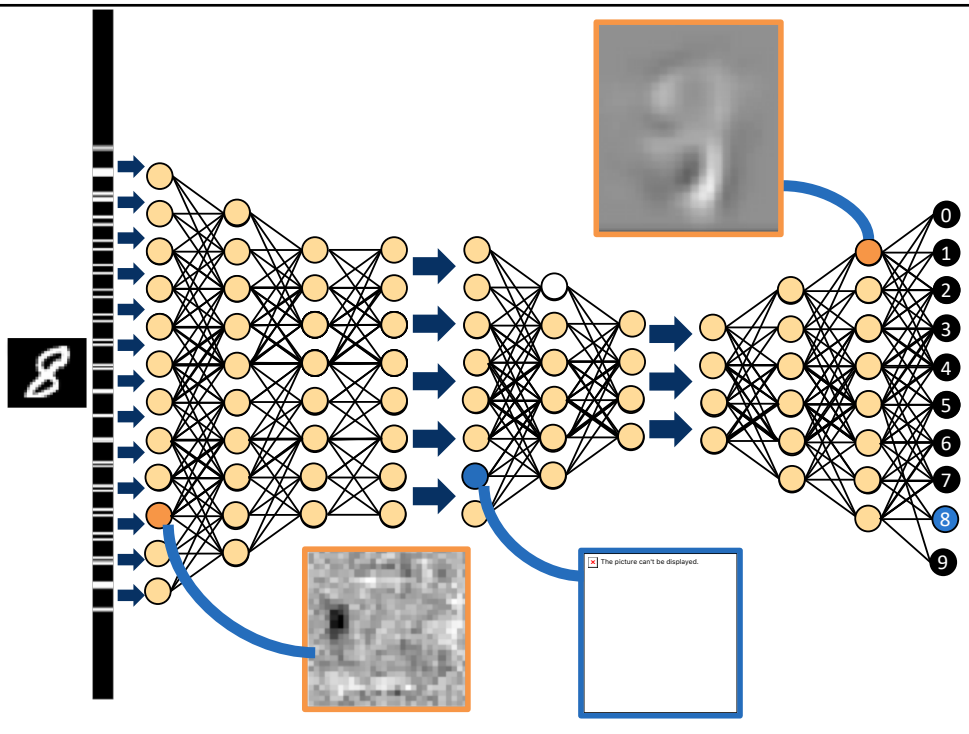
Each image is composed of 28 x 28 = 784 pixels each containing a grayscale value between 0 and 255

This **low-level representation** uses a vector of 784 features, each with values between 0 and 255

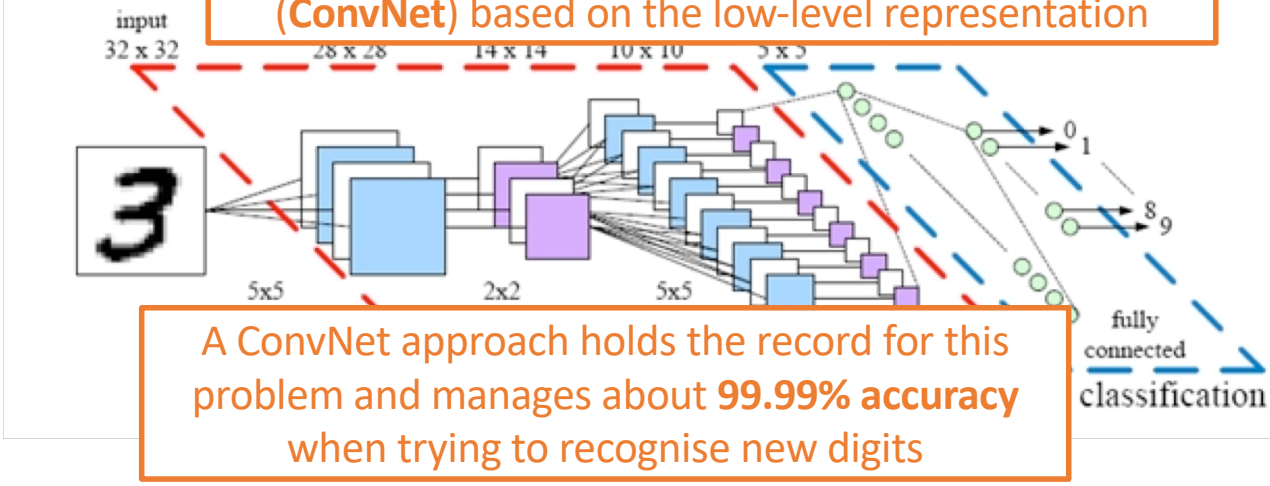




A simple representation learning approach to the digit recognition problem could use a **multi-layer perceptron** to make predictions using the low-level representation

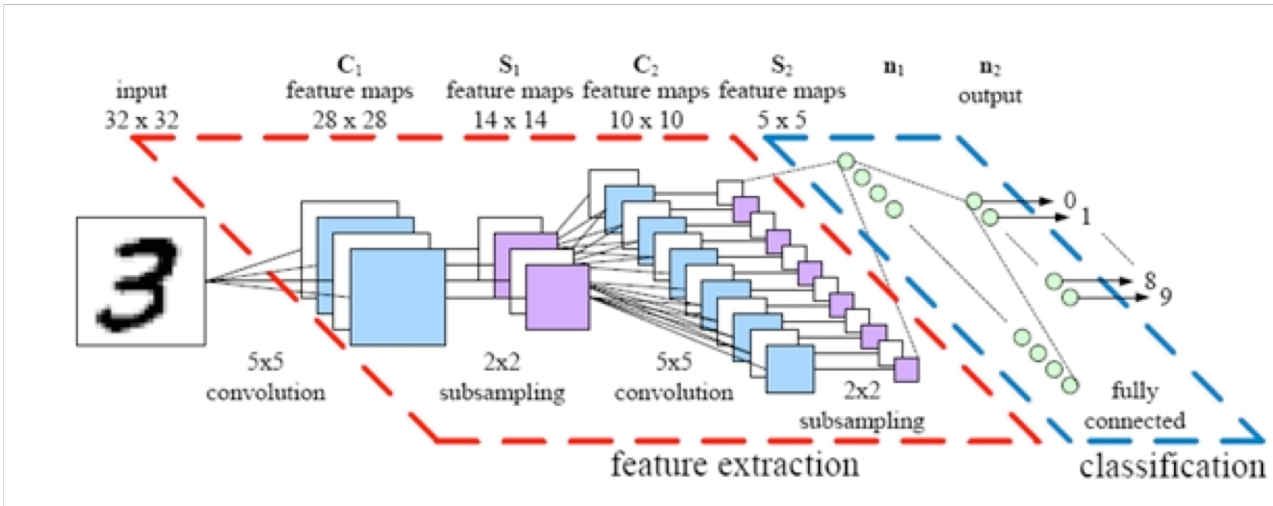


A **deep learning** approach to the digit recognition problem could use a **convolutional neural network** (**ConvNet**) based on the low-level representation

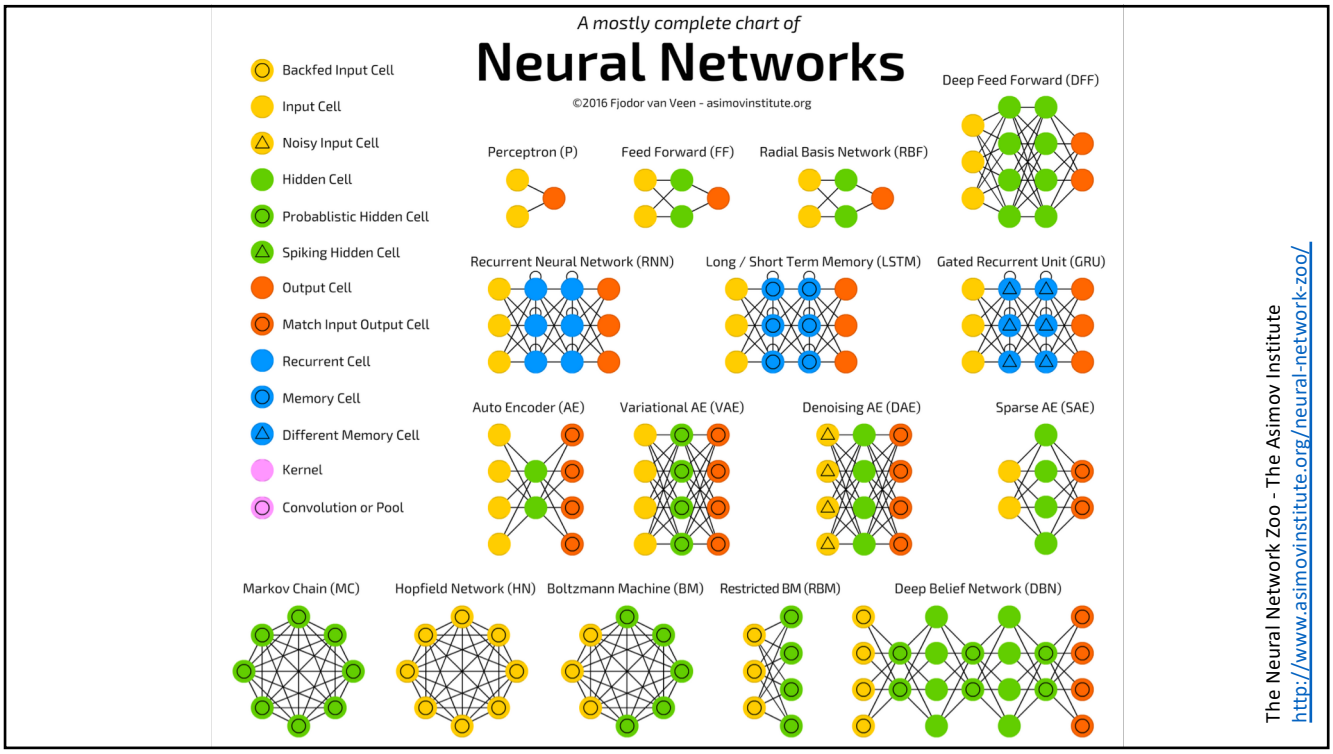


A ConvNet approach holds the record for this problem and manages about **99.99% accuracy** when trying to recognise new digits

Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster
Jonathan Ward, Sergey Andreev, Francisco Heredia, Bogdan Lazar, Zlatka Manevska
http://parse.ele.tue.nl/education/cluster2



Efficient mapping of the training of Convolutional Neural Networks to a CUDA-based cluster
Jonathan Ward, Sergey Andreev, Francisco Heredia, Bogdan Lazar, Zlatka Manevska
http://parse.ele.tue.nl/education/cluster2



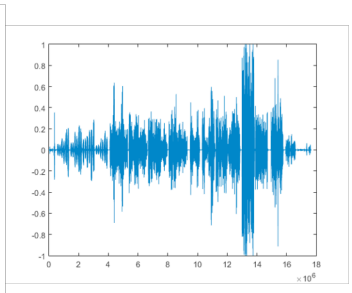
The Neural Network Zoo - The Asimov Institute
http://www.asimovinstitute.org/neural-network-zoo/

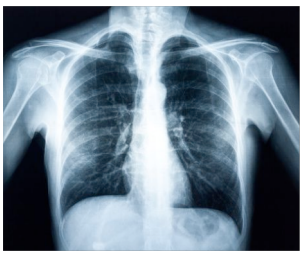
Structured Data

Table 1.2
A more complex credit scoring dataset.

ID	Amount	Salary	Loan-Salary Ratio	Age	Occupation	Property	Type	Outcome
1	245,100	66,400	3.69	44	industrial	farm	stb	repay
2	90,600	75,300	1.20	41	industrial	farm	stb	repay
3	195,600	52,100	3.75	37	industrial	farm	ftb	default
4	157,800	67,600	2.33	44	industrial	apartment	ftb	repay
5	150,800	35,800	4.21	39	professional	apartment	stb	default
6	133,000	45,300	2.94	29	industrial	farm	ftb	default
7	193,100	73,200	2.64	38	professional	house	ftb	repay
8	215,000	77,600	2.77	17	professional	farm	ftb	repay
9	83,000	62,500	1.33	30	professional	house	ftb	repay
10	186,100	49,200	3.78	30	industrial	house	ftb	default
11	161,500	53,300	3.03	28	professional	apartment	stb	repay
12	157,400	63,900	2.46	30	professional	farm	stb	repay
13	210,000	54,200	3.87	43	professional	apartment	ftb	repay
14	209,700	53,000	3.96	39	industrial	farm	ftb	default
15	143,200	65,300	2.19	32	industrial	apartment	ftb	default
16	203,000	64,400	3.15	44	industrial	farm	ftb	repay
17	247,800	63,800	3.88	46	industrial	house	stb	repay
18	162,700	77,400	2.10	37	professional	house	stb	repay
19	213,300	61,100	3.49	21	industrial	apartment	ftb	default
20	284,100	32,300	8.80	51	industrial	farm	ftb	default
21	154,000	48,900	3.15	49	professional	house	stb	repay
22	112,800	79,700	1.42	41	professional	house	ftb	repay
23	252,000	59,700	4.22	27	professional	house	stb	default
24	175,200	39,900	4.39	37	professional	apartment	stb	default
25	149,700	58,600	2.55	35	industrial	farm	stb	default

Unstructured Data

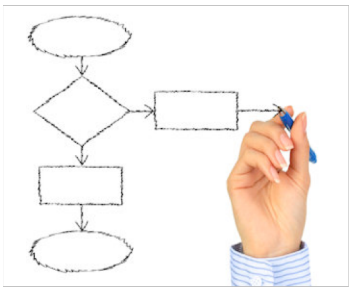




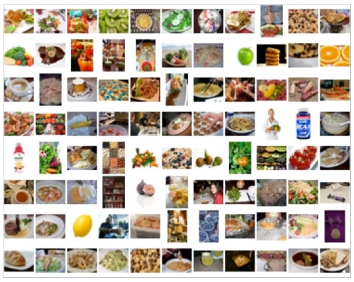
Apple Inc (AAPL) on Tuesday reported fiscal first-quarter net income of $18.02bn.

The Cupertino, California-based company said it had profit of $3.06 per share.

The results surpassed Wall Street expectations. The maker of iPhones, iPads and other products posted revenue of $74.6bn in the period, also exceeding Street forecasts. Analysts expected $67.38bn…"



ALGORITHMS



DATA

HARDWARE

APPLICATIONS

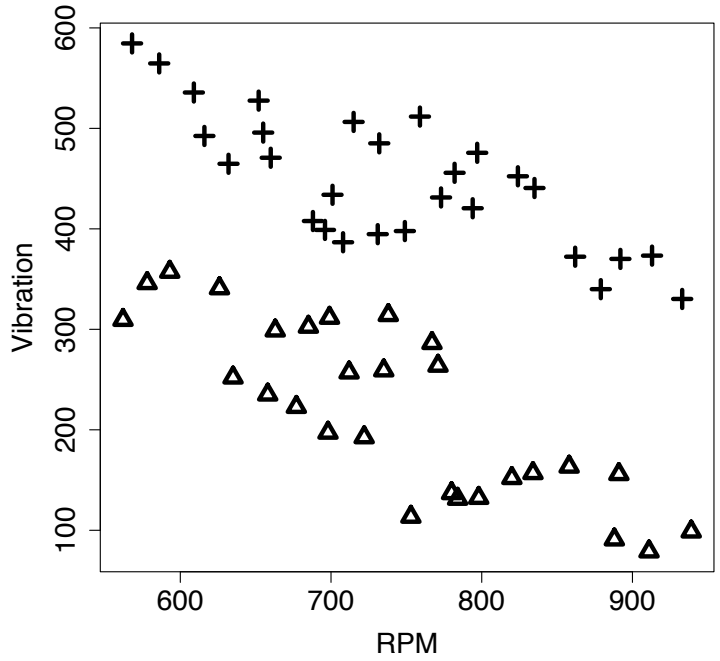
Google

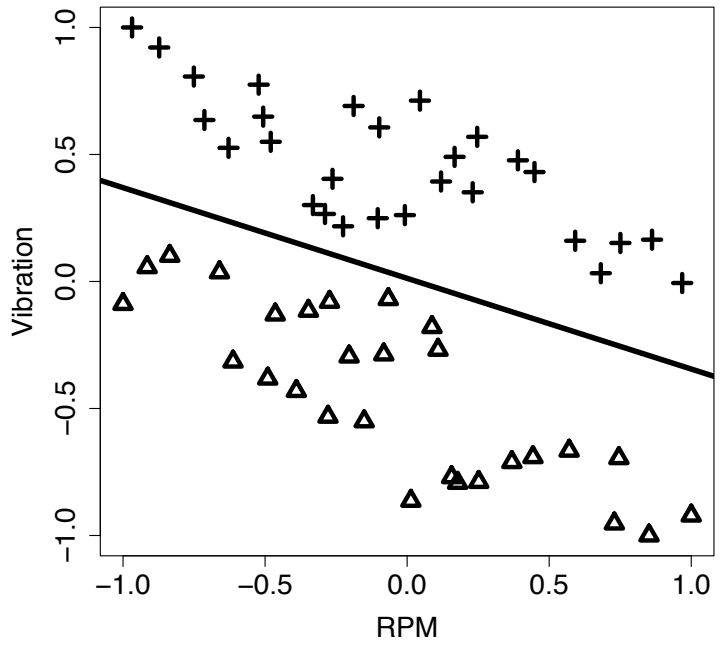
NETFLIX

facebook

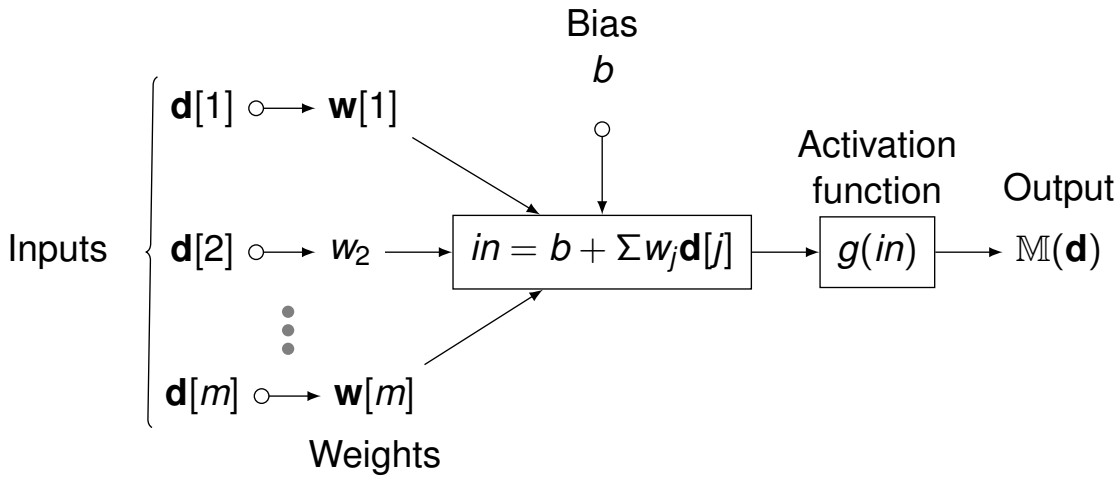
Baidu 百度

PERCEPTION


Vibration
100
200
300
400
500
600
600
700
800
900
RPM


Vibration
−1.0
−0.5
0.0
0.5
1.0
−1.0
−0.5
0.0
0.5
1.0
RPM

Perceptron

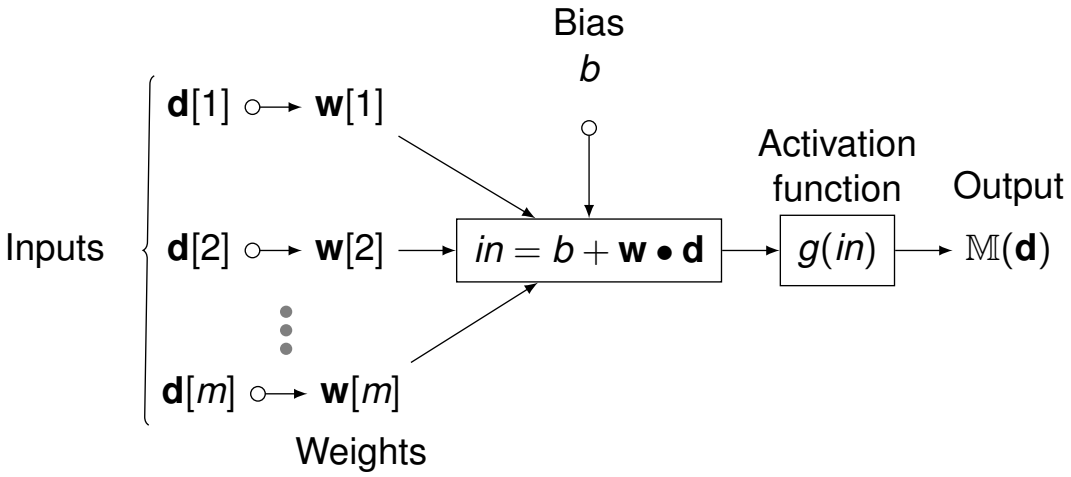


Perceptron

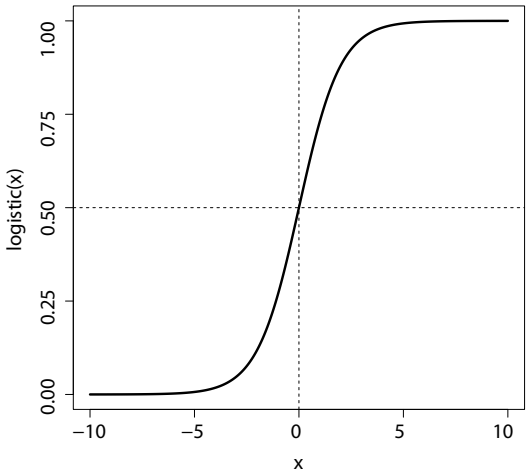
$$\mathbb{M}(\mathbf{d}) \leftarrow g(in_i)$$

$$in_i \leftarrow \left(\sum_j \mathbf{w}[j] * \mathbf{d}[j] \right) + b$$

Perceptron

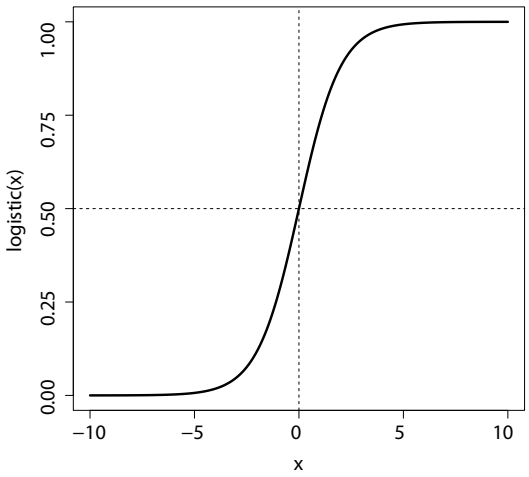


Activation Function



There are lots of different activation functions that we can use in neural networks; for example, the **logistic function** is an example of a suitable activation function

Activation Function



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Perceptron

$$\mathbb{M}(\mathbf{d}) \leftarrow g(in_i)$$

$$in_i \leftarrow \left(\sum_j \mathbf{w}[j] * \mathbf{d}[j] \right) + b$$

Perceptron

$$\mathbb{M}(\mathbf{d}) \leftarrow \frac{1}{1 + e^{-\left(\left(\sum_j \mathbf{w}[j] * \mathbf{d}[j]\right) + b\right)}}$$

Loss Function

$$\mathscr{L}\left(\mathbb{M}(\mathbf{d}), t\right) = \frac{1}{2}\left(\mathbb{M}(\mathbf{d}) - t\right)^2$$

Squared error makes sense as a loss function

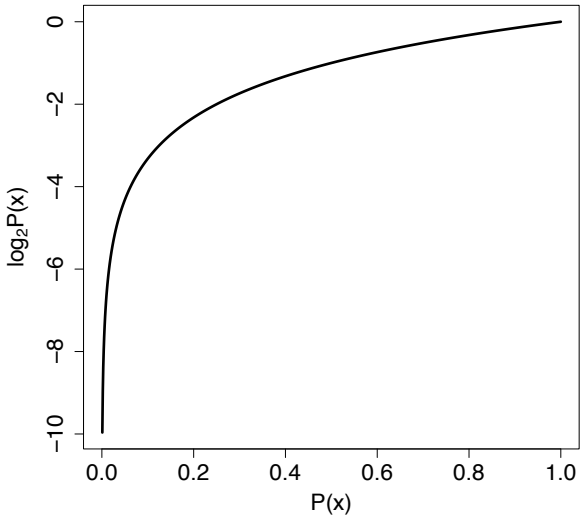
- It causes problems in the shape of the error surface when performing gradient descent
- Usually not used in gradient descent

Loss Function

$$\mathscr{L}\left(\mathbb{M}(\mathbf{d}), t\right) = -\left(\left(t \times log\left(\mathbb{M}\left(\mathbf{d}\right)\right) + \left(1 - t\right) \times log\left(1 - \mathbb{M}\left(\mathbf{d}\right)\right)\right)\right)$$

Log loss function is a more well behaved loss function for binary classification problems

Loss Function



Cost Function

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right)$$

The cost function is simply the average loss across a training datset

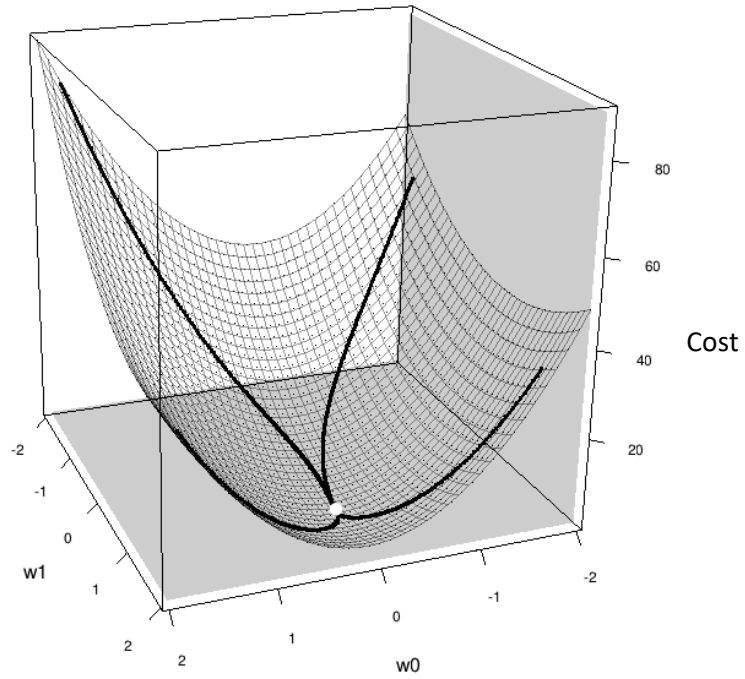
- We can add more terms later on for regularisation

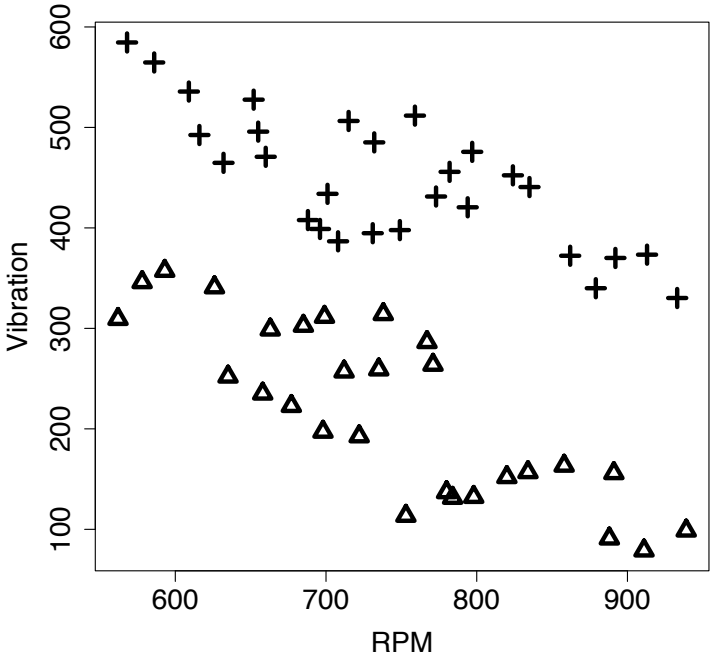
Loss Functions & Cost Functions

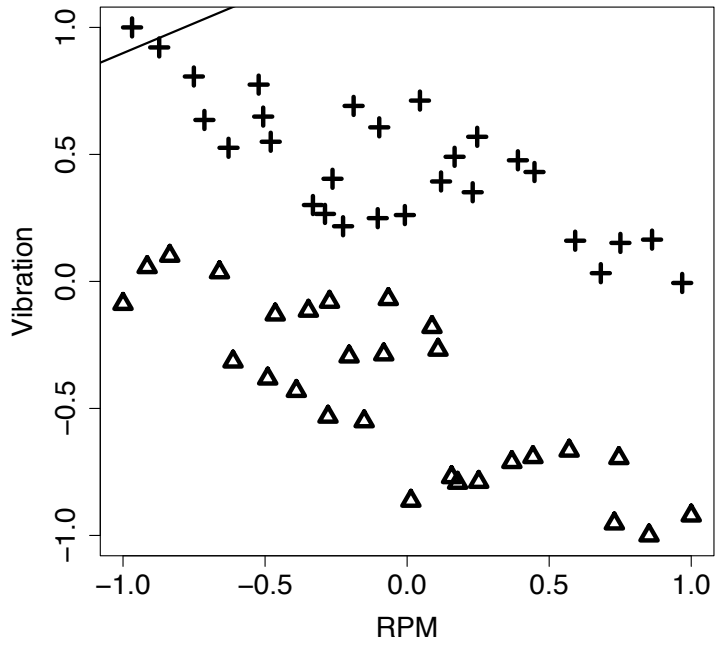
What is the difference between a loss function and a cost function?

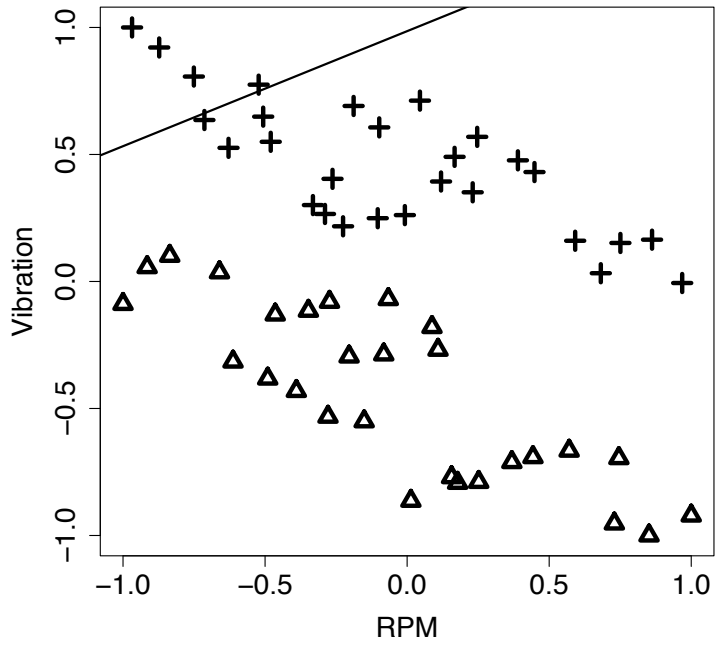
- A **loss function** is a measure of the prediction error on a single training instance
- A **cost function** is the measure of the average prediction error across a set of training instances
- Cost functions allow us to add in **regularisation**

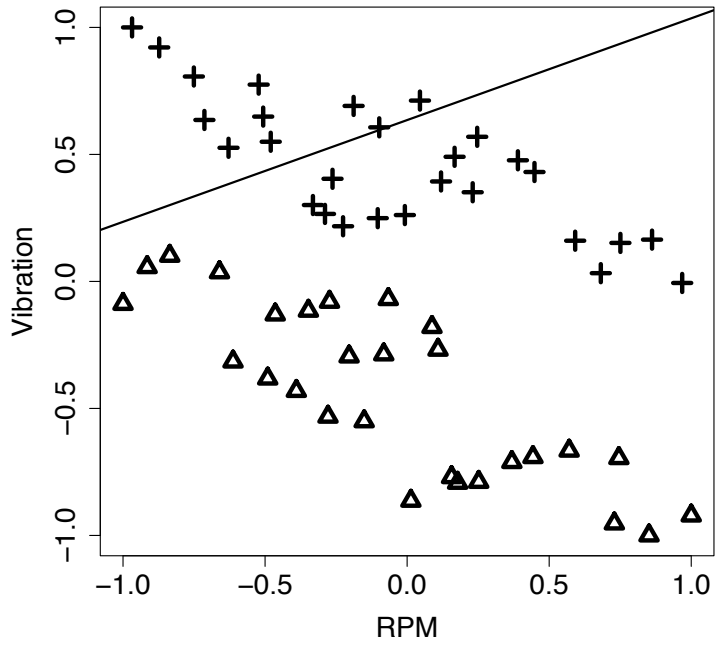
GRADIENT DESCENT

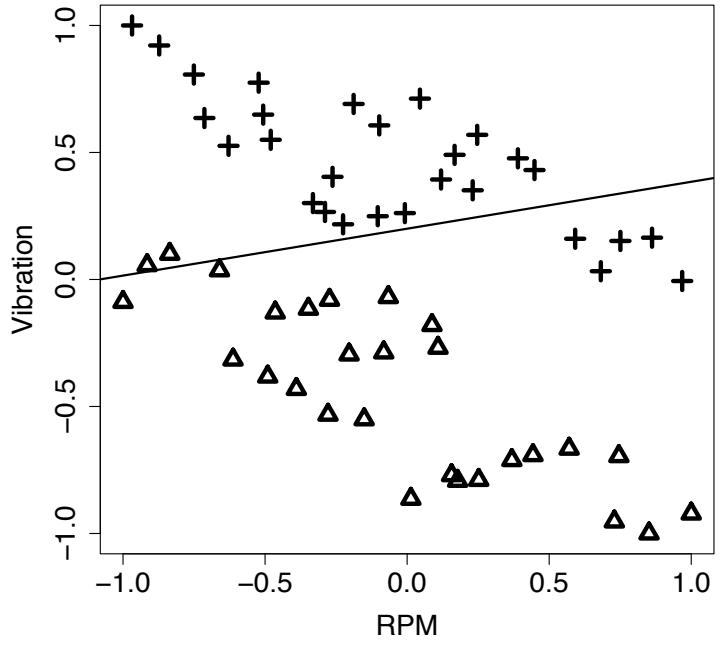

Cost
80
60
40
20
w1
-2
-1
0
1
2
w0
2
1
0
-1
-2

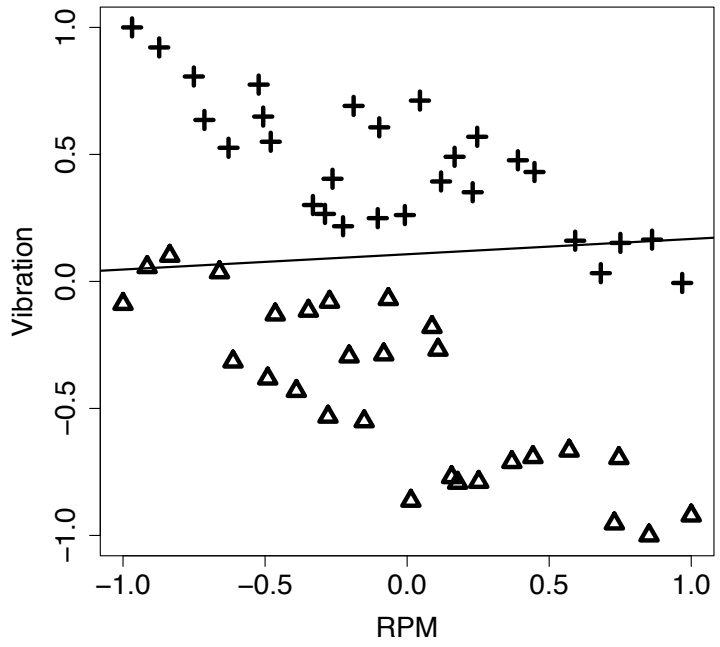

Vibration
100
200
300
400
500
600
RPM
600
700
800
900

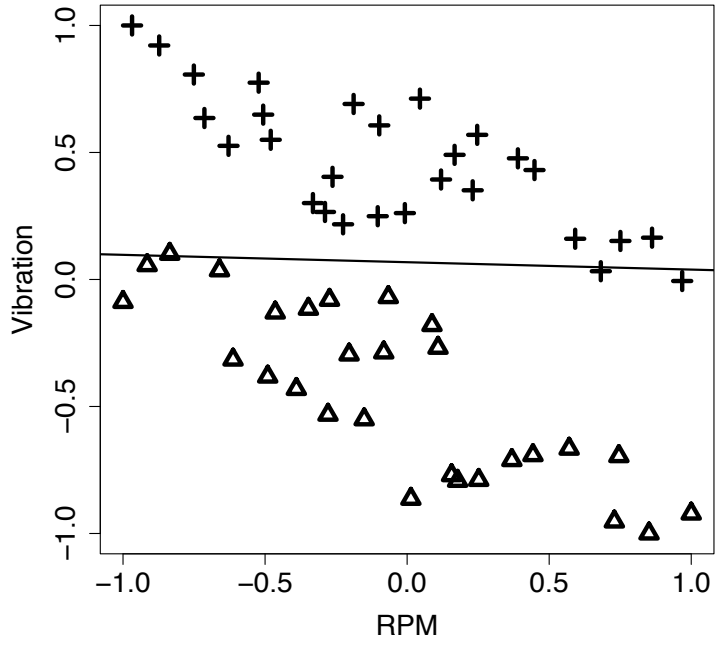

Vibration
RPM
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

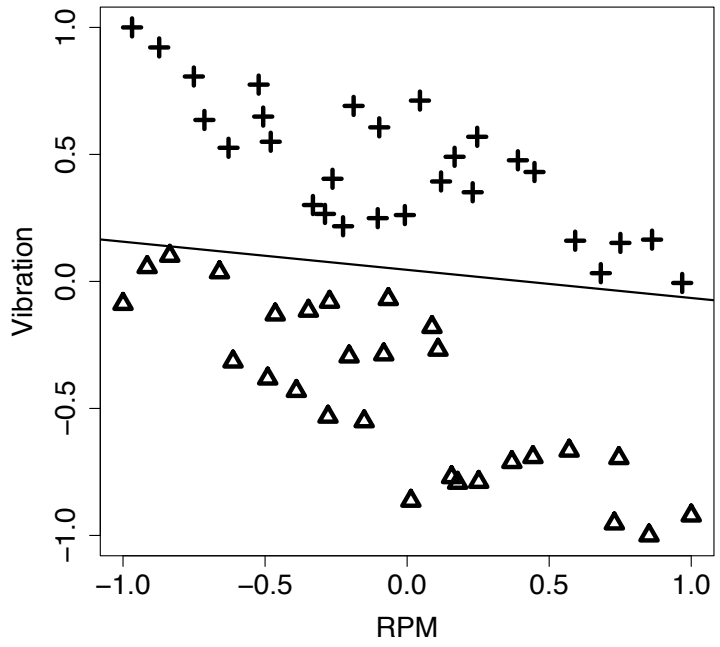

Vibration
RPM
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

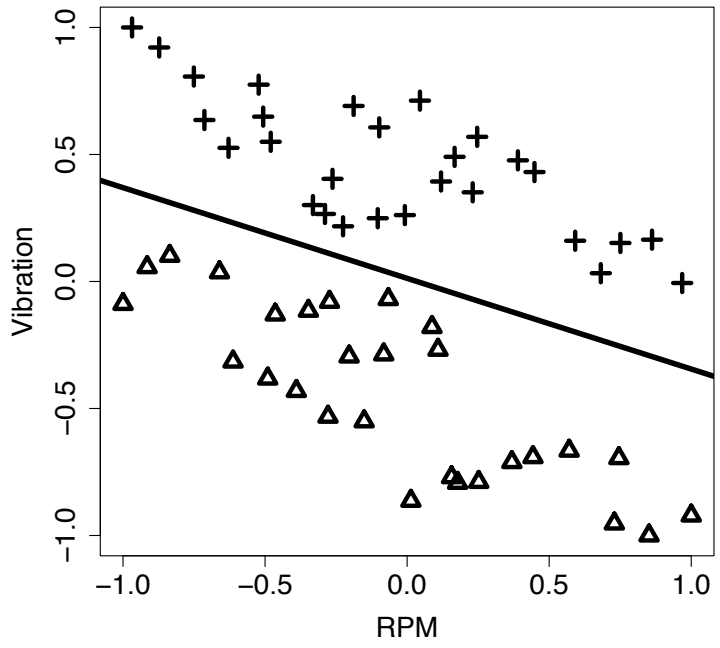

Vibration
RPM
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

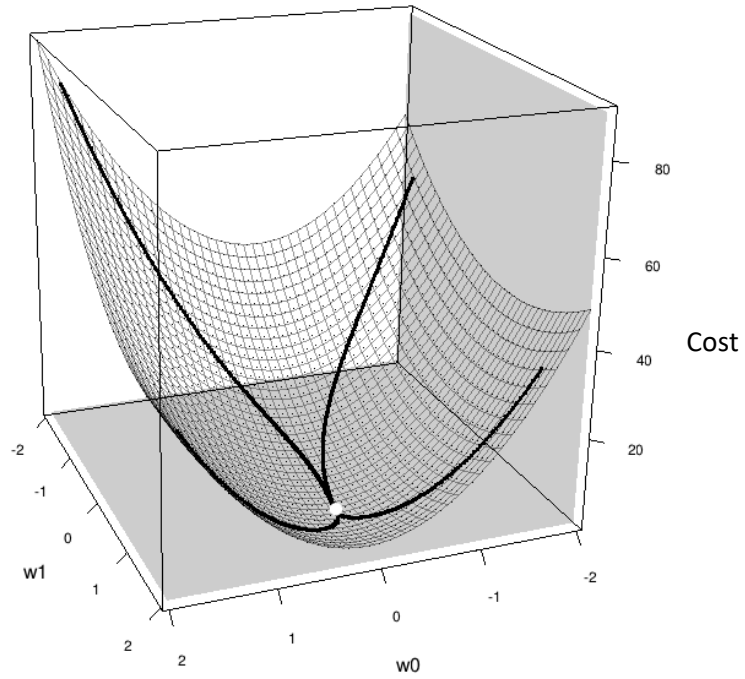

Vibration
RPM
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0


Vibration
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
RPM


Vibration
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
RPM


Vibration
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
RPM


Vibration
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
RPM



Gradient Descent Algorithm

Choose random weights

Until convegence

- Set all gradients to 0
- For each training instance
  - Calculate model output
  - Calculate loss
  - Update gradient sum for each weight and bias term
- Update weights and biases using weight update rule

COMPUTATION GRAPHS

Computation Graphs

The descriptive language of deep learning models

Functional description of the required computation

Can be used to do two types of computation

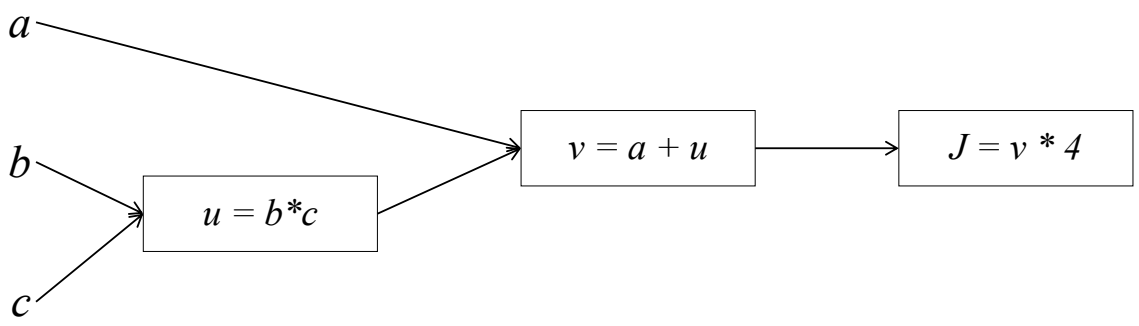
- Forward computation
- Backward computation

Computation Graphs

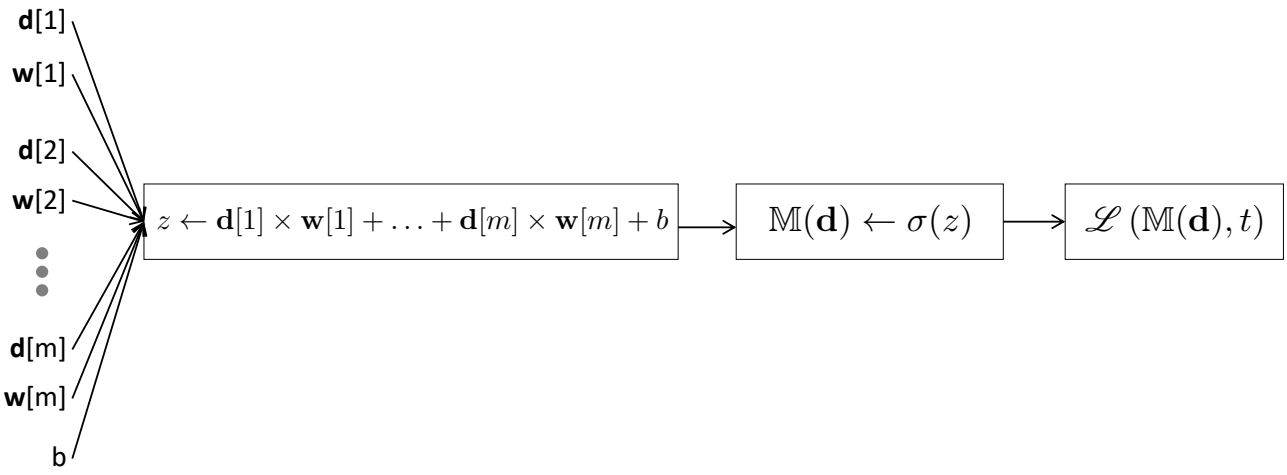
$$J(a, b, c) = (a + b*c) * 4$$

Computation Graphs

$$J(a, b, c) = (a + b*c) * 4$$

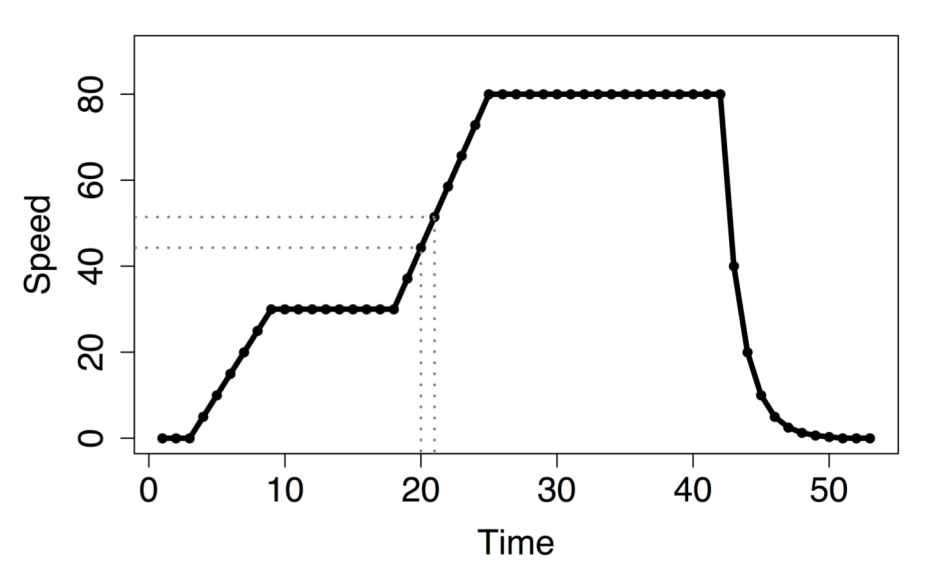


Computation Graphs

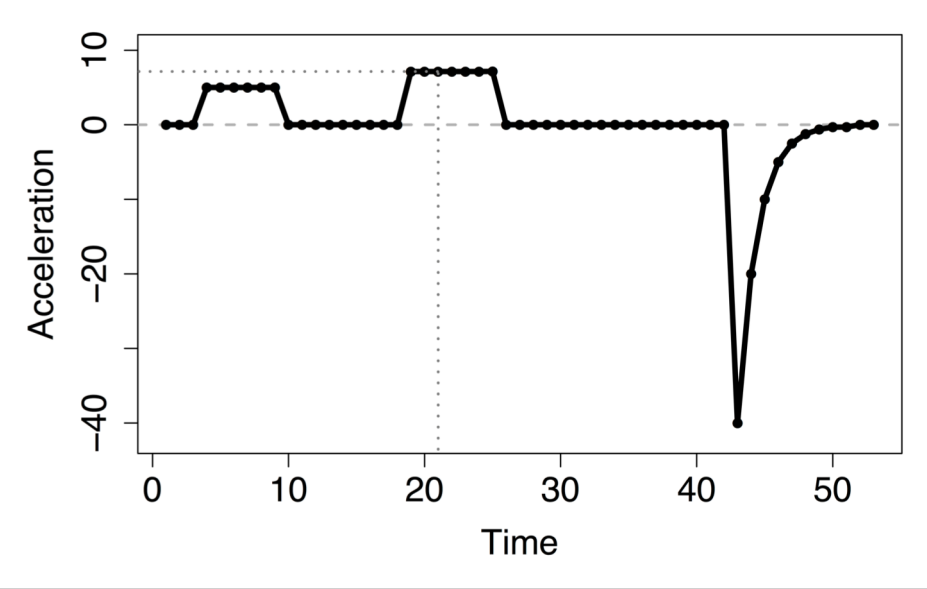


DERIVATIVES

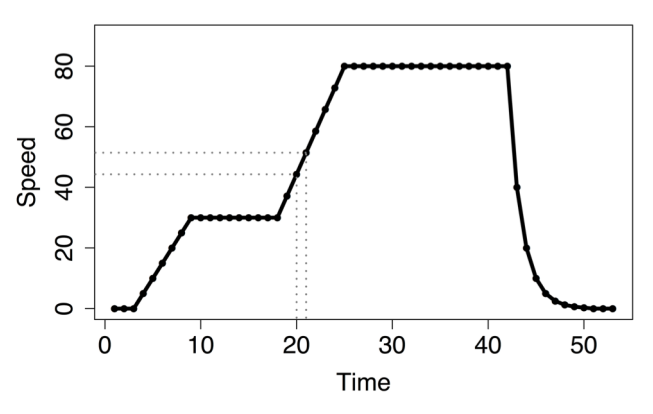
Derivatives

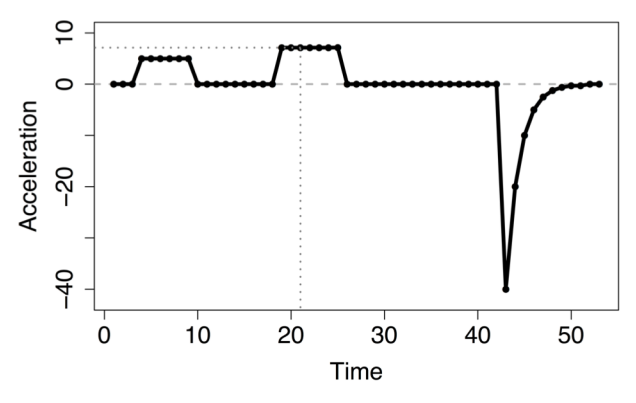


Derivatives

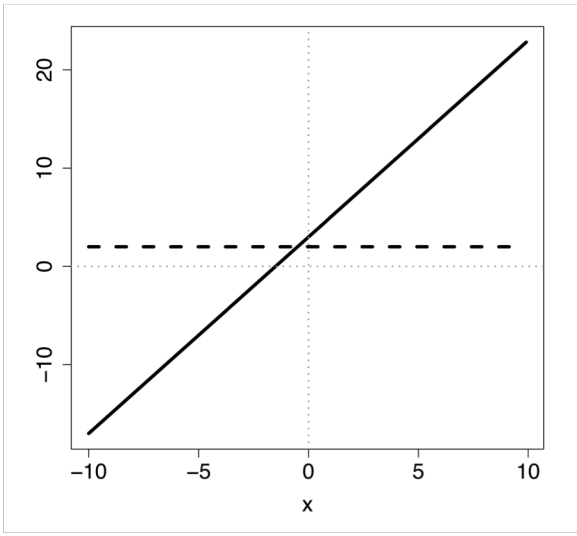


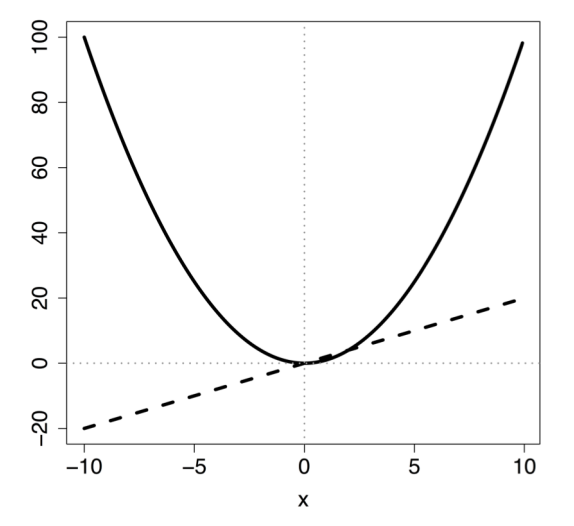
Derivatives



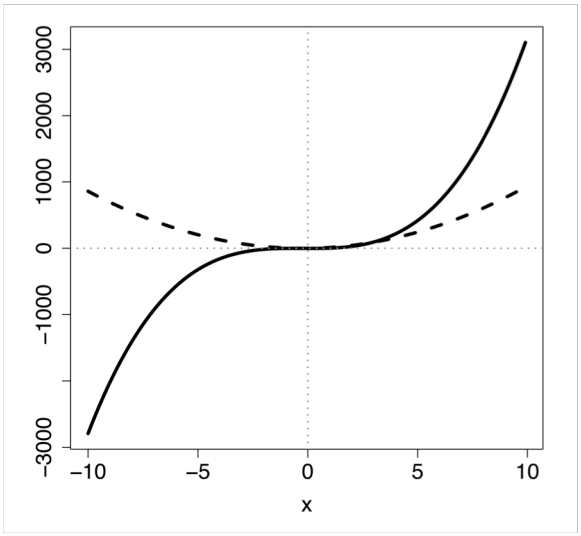


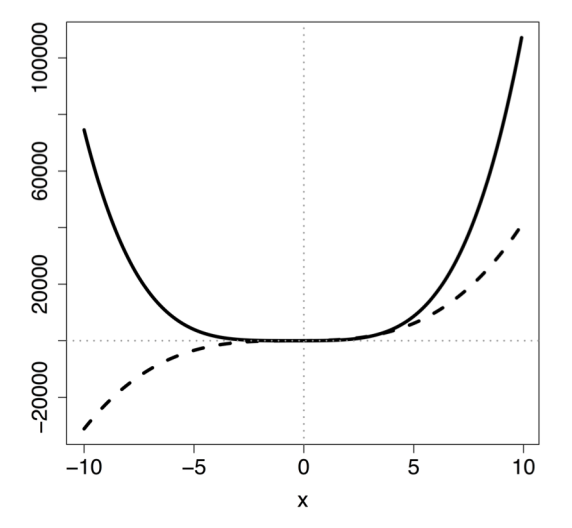
Derivatives





Derivatives





COMPUTATION GRAPHS & DERIVATIVES

Computation Graphs

The descriptive language of deep learning models

Functional description of the required computation

Can be used to do two types of computation

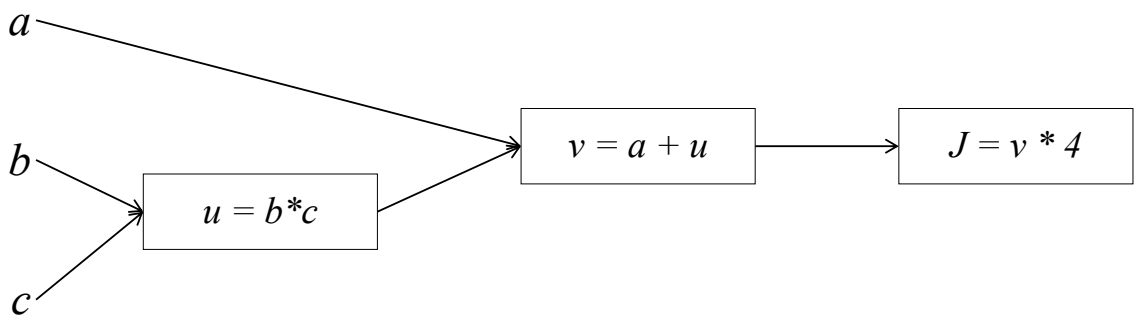
- Forward computation
- **Backward computation**

Computation Graphs

$$J(a, b, c) = (a + b*c) * 4$$

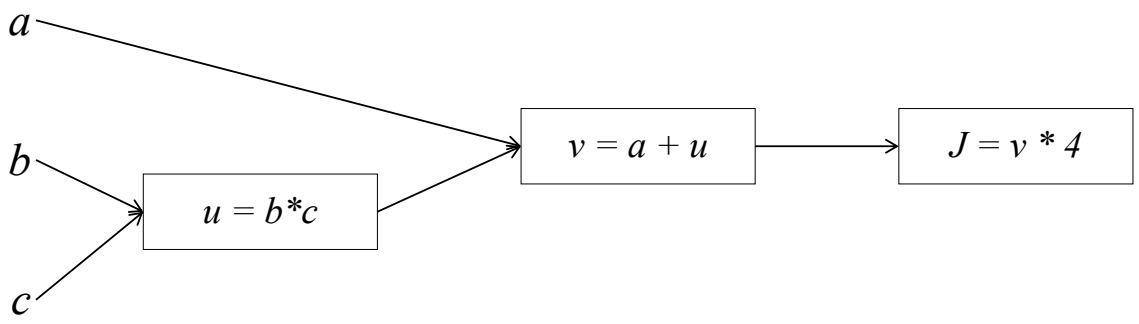
Computation Graphs

$$J(a, b, c) = (a + b*c) * 4$$



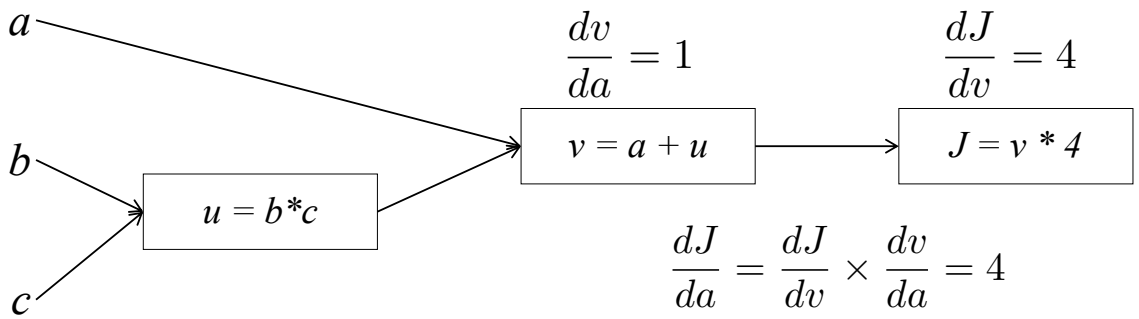
Computation Graphs & Gradients

$$J(a, b, c) = (a + b*c) * 4$$

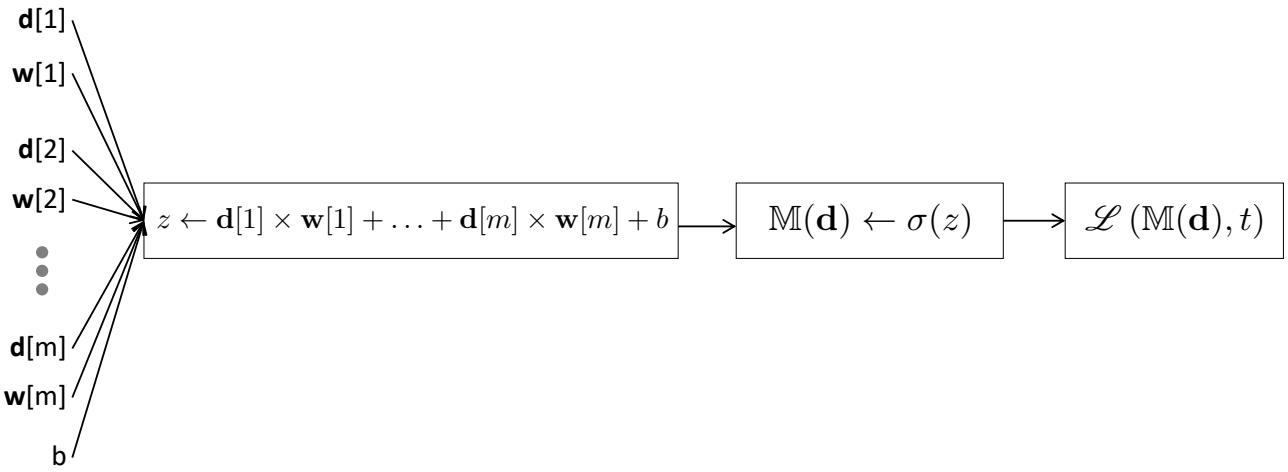


Computation Graphs & Gradients

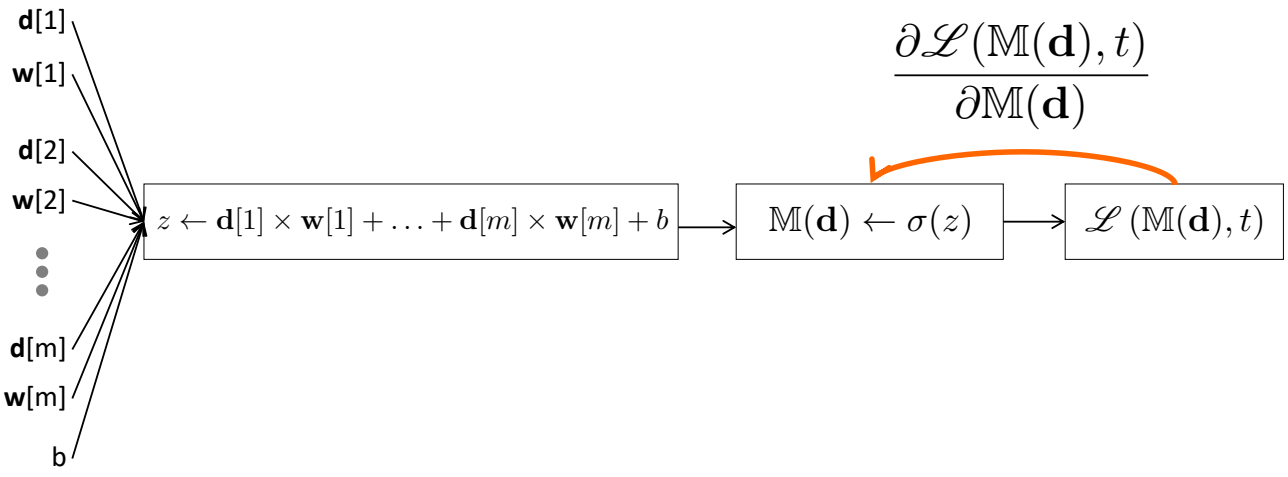
$$J(a, b, c) = (a + b*c) * 4$$



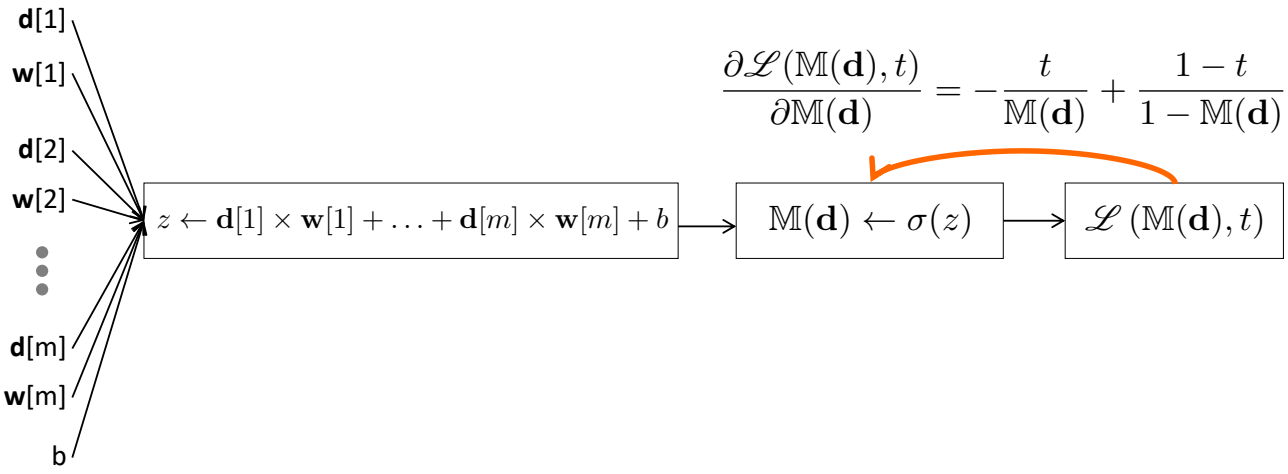
Computation Graphs



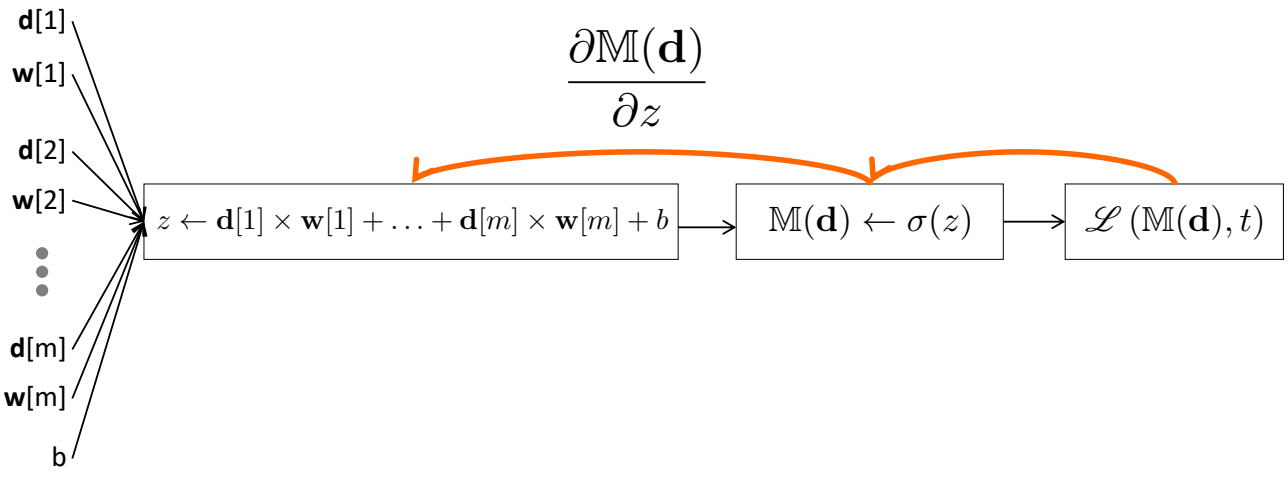
Computation Graphs



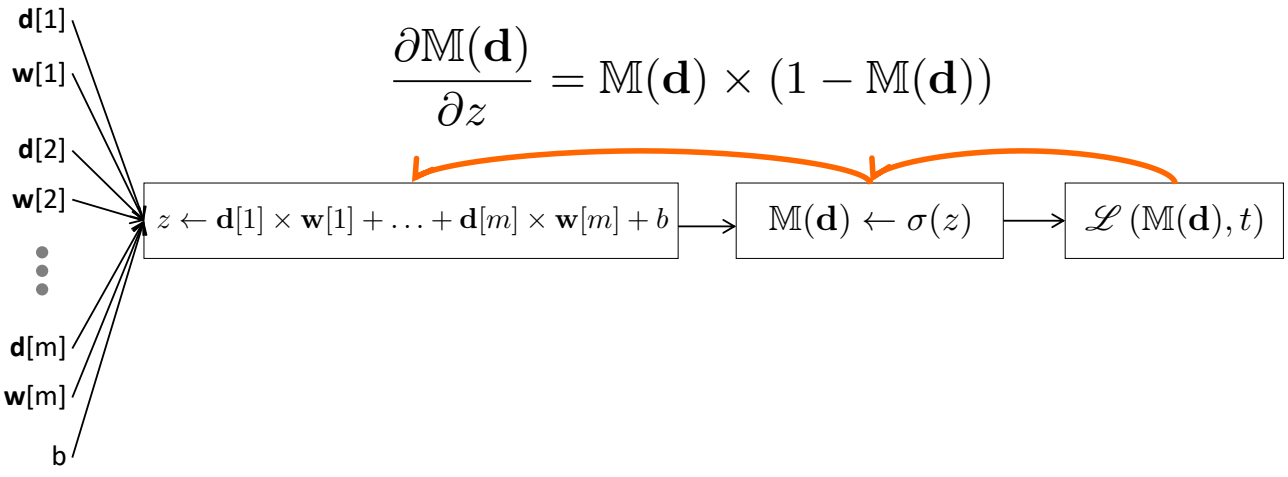
Computation Graphs



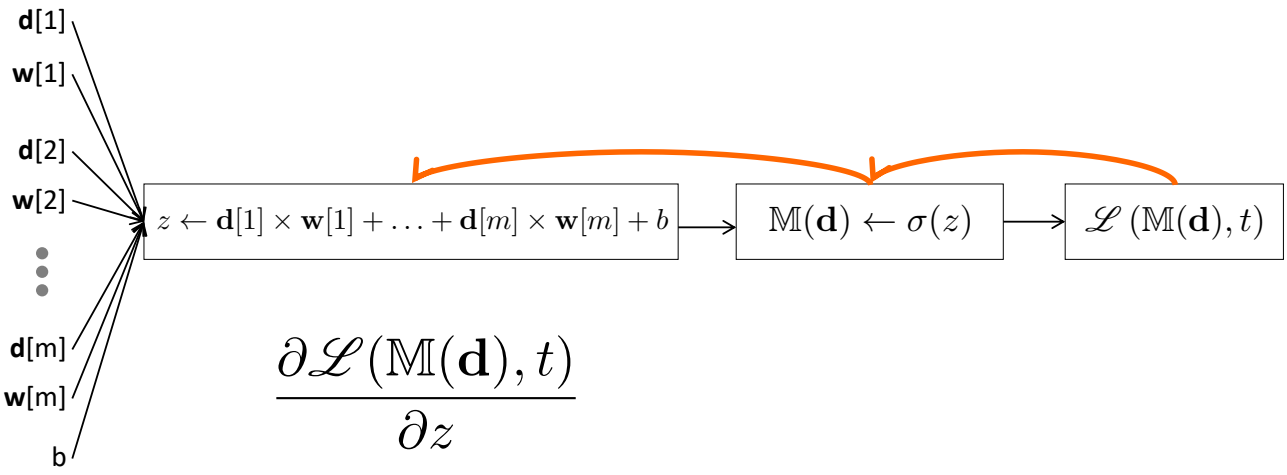
Computation Graphs



Computation Graphs

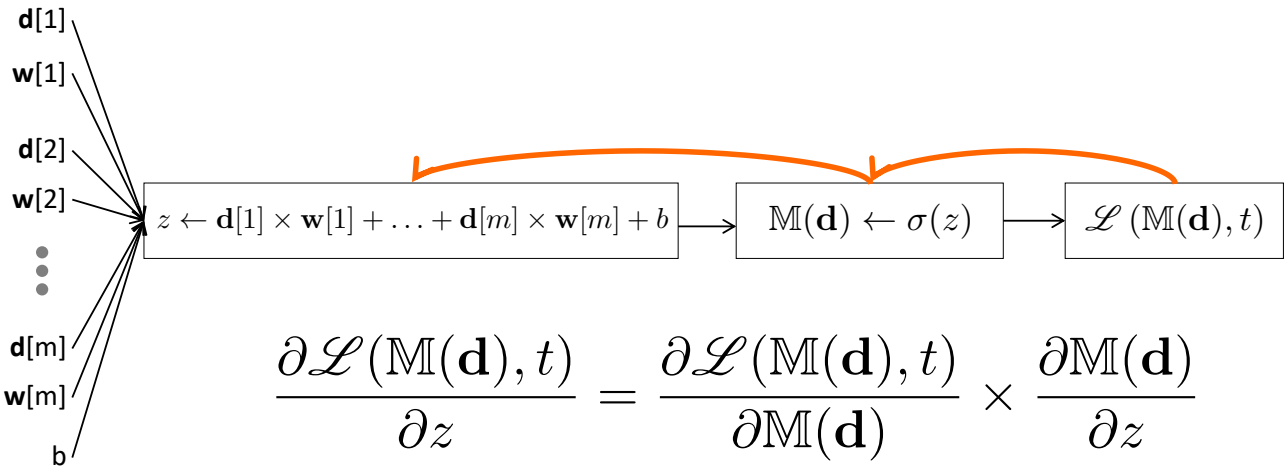


Computation Graphs



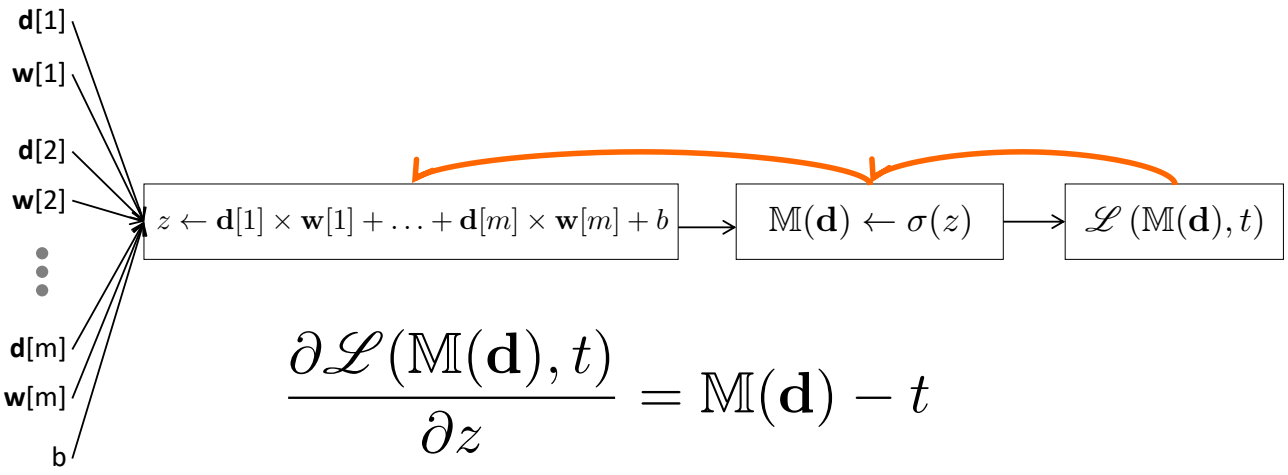
$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial z}$$

Computation Graphs



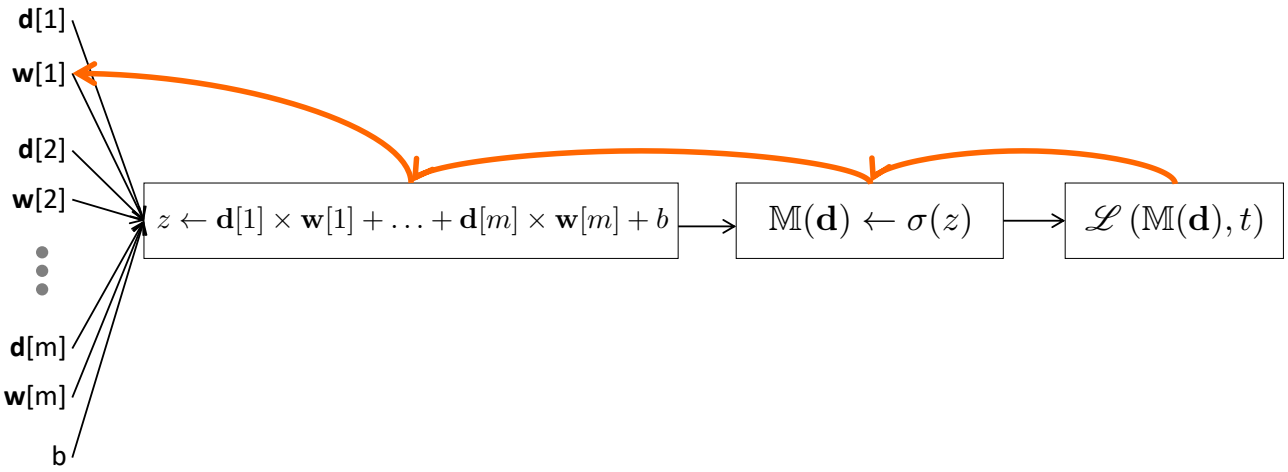
$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial z} = \frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbb{M}(\mathbf{d})} \times \frac{\partial \mathbb{M}(\mathbf{d})}{\partial z}$$

Computation Graphs

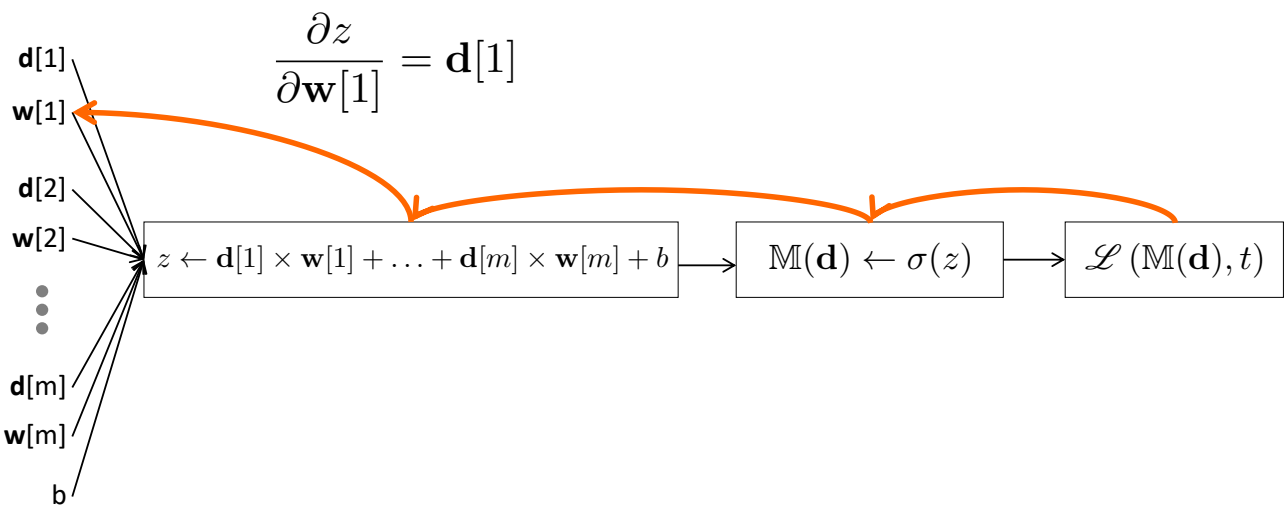


$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial z} = \mathbb{M}(\mathbf{d}) - t$$

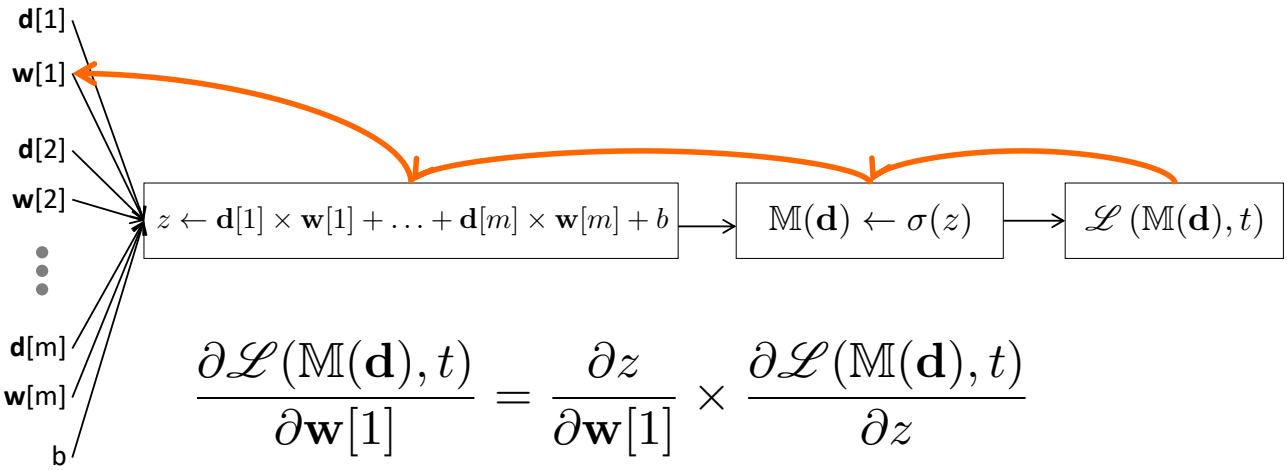
Computation Graphs



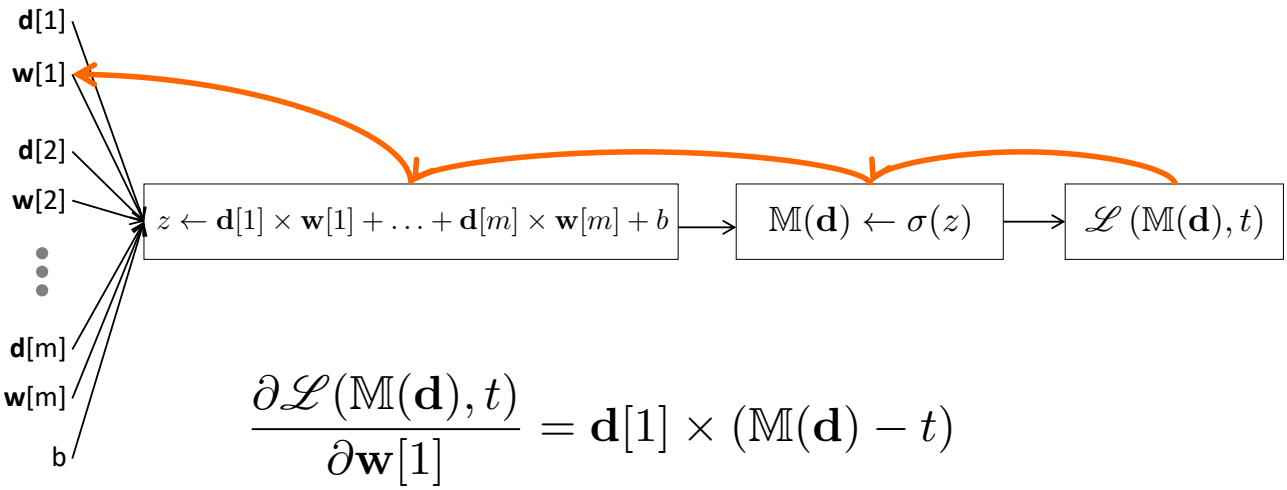
Computation Graphs



Computation Graphs

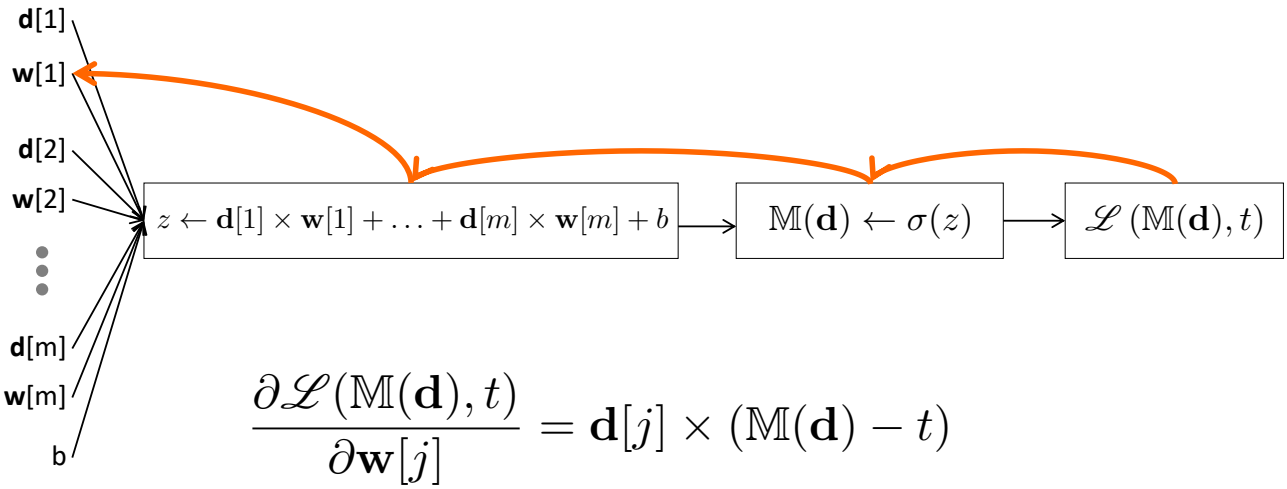


Computation Graphs



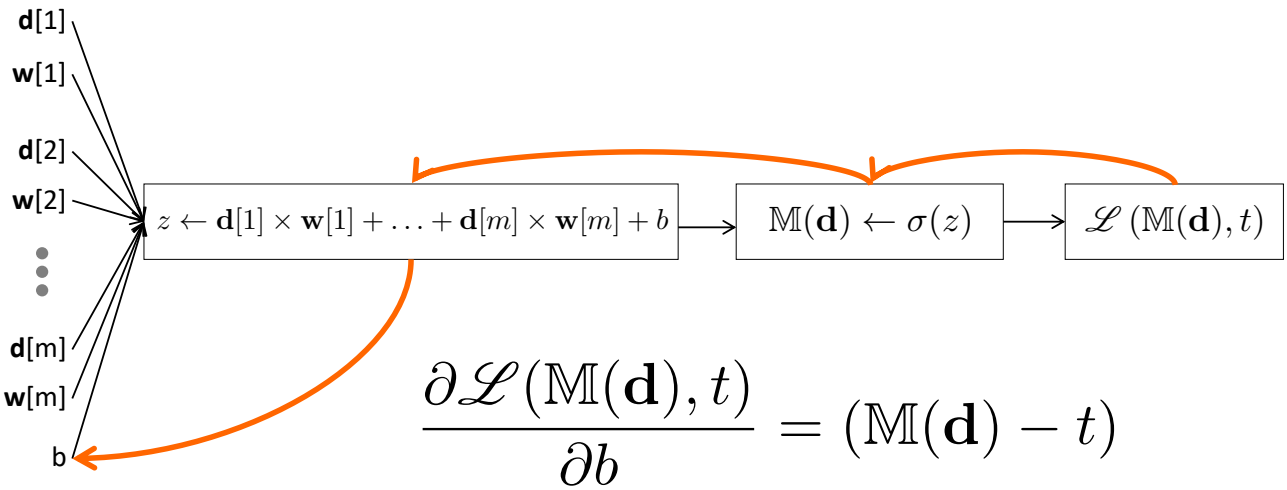
$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}[1]} = \mathbf{d}[1] \times (\mathbb{M}(\mathbf{d}) - t)$$

Computation Graphs



$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}[j]} = \mathbf{d}[j] \times (\mathbb{M}(\mathbf{d}) - t)$$

Computation Graphs



$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial b} = (\mathbb{M}(\mathbf{d}) - t)$$

Weight Update Rules

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \alpha \frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}[j]}$$

$$b \leftarrow b - \alpha \frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial b}$$

Weight Update Rules

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \alpha \times \mathbf{d}[j] \times (\mathbb{M}(\mathbf{d}) - t)$$

$$b \leftarrow b - \alpha \times (\mathbb{M}(\mathbf{d}) - t)$$

Weight Update Rules

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{w}[j]}$$

$$b \leftarrow b - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial b}$$

Derivative of Cost Function

$$J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right)$$

Derivative of Cost Function

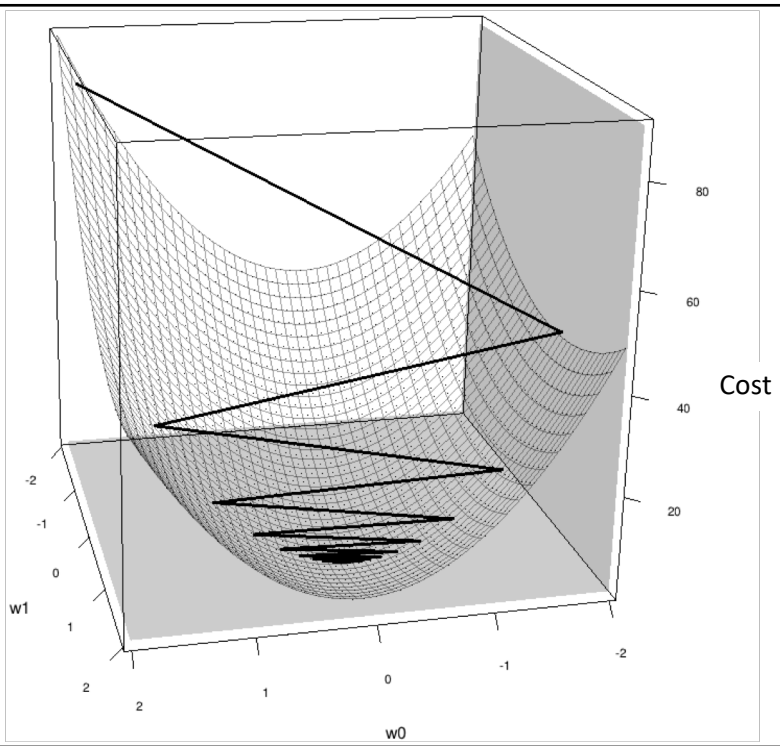
$$\frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{w}[j]} = \frac{1}{m} \sum_{i=1}^{m} \frac{\partial \mathscr{L}\left(\mathbb{M}(\mathbf{d}_i), t_i\right)}{\partial \mathbf{w}[j]}$$

Weight Udpate Rules

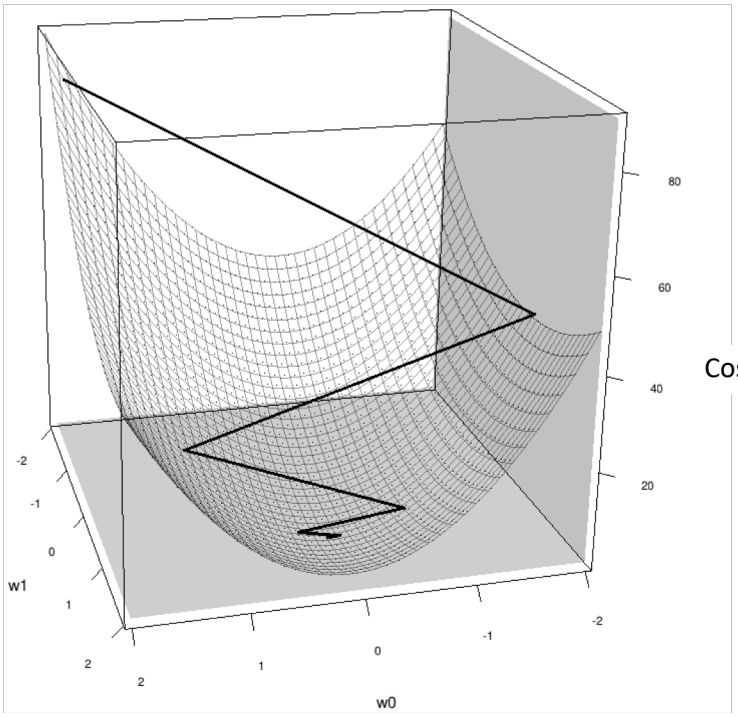
$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] - \alpha \times \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{d_i}[j] \times (\mathbb{M}(\mathbf{d_i}) - t_i) \right)$$

$$b \leftarrow b - \alpha \times \left(\frac{1}{m} \sum_{i=1}^{m} (\mathbb{M}(\mathbf{d_i}) - t_i) \right)$$

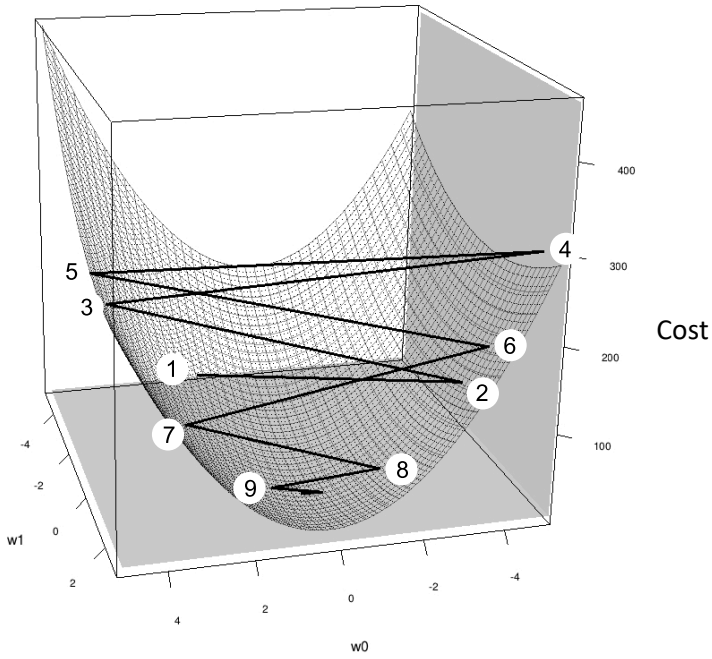
Learning Rate



Learning Rate



Learning Rate



SUMMARY

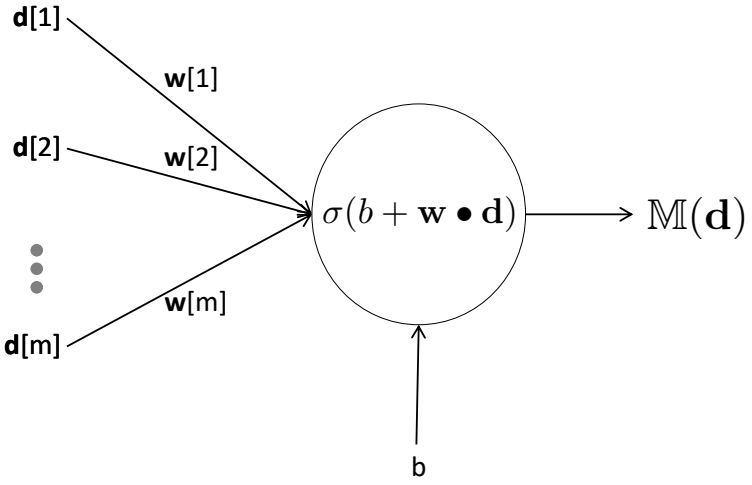
Summary

Recent developments in neural networks and deep learning have led to a step change in the performance of machine learning models - especially for problems with unstructured data

Gradient descent is the core algorithm behind deep learning

Computation graphs simplify the specification od deep networks

Perceptron



Questions

?