Deep Learning

Dmytro Fishman (dmytro.fishman@ut.ee)
<table>
<thead>
<tr>
<th>Assignment</th>
<th>Date of assignment</th>
<th>Deadline (midnight 23:59)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>Sep 5</td>
<td>Sep 18</td>
</tr>
<tr>
<td>HW2</td>
<td>Sep 19</td>
<td>Oct 2</td>
</tr>
<tr>
<td>HW3</td>
<td>Oct 3</td>
<td>Oct 16</td>
</tr>
<tr>
<td>Paper summary</td>
<td>Oct 10</td>
<td>Oct 23</td>
</tr>
<tr>
<td>HW4</td>
<td>Oct 17</td>
<td>Oct 30</td>
</tr>
<tr>
<td>HW5</td>
<td>Oct 31</td>
<td>Nov 13</td>
</tr>
<tr>
<td>HW6</td>
<td>Nov 21</td>
<td>Dec 4</td>
</tr>
<tr>
<td>Project</td>
<td>TBA</td>
<td>Dec 12 - 14</td>
</tr>
</tbody>
</table>

*All deadlines are subject to change, check out Slack and website for updates*
Hierarchical clustering

- Parameters: linkage type and the distance
- Using a dendrogram can extract any number of clusters
- Hierarchical clustering is very slow ($N^3$)

K-means

- Parameters: $K$ and the distance
- You will have exactly $K$ clusters
- You need to know $K$
- Although NP-hard, it works surprisingly fast

DBSCAN

- Parameters: minPoints and epsilon
- Does not depend on $K$
- Parameters are tedious to pick up
- Noise
At first all points are considered to be separate clusters.

Euclidean distance is calculated between all clusters.

Agglomerative strategy:

Repeat as many times as there are clusters.

Until there is one cluster left.

Two closest clusters are merged into one.
At first all points are considered to be one cluster.

Repeat as many times as there are clusters.

Use some other clustering algorithm (e.g. K-Means) to split clusters.

Divisive strategy

Until there all points are considered separate clusters.
Randomly initialise $K$ cluster centres

Assign all points to closest cluster centres

K-means algorithm

K-means is computationally difficult (NP-hard)*

but it has been shown to converge in finite number of steps

*https://cseweb.ucsd.edu/~avattani/papers/kmeans_hardness.pdf
Any good way to choose $K$?

The rule of thumb is to choose $\sqrt{\frac{n}{2}}$ as $K$.

Elbow method: increase $K$ until it does not help to describe data better.

Overall silhouette score is an average of silhouette scores for all points.
Lonely points, are considered noise by DBSCAN.
Overview of ML

Supervised Learning
- Nearest Neighbour Classifier
- Linear regression
- Decision trees
- Overfitting
- Tran/val split
- Cross Validation
- Model selection
- Model implementation
- Data preprocessing
- Classification vs regression

Unsupervised Learning
- K-means
- Hierarchical clustering
- DBSCAN
- PCA
- T-SNE
- UMAP

Deep Learning
- Artificial neuron
- Forward path
- Activation Functions
- Gradient descent
- Backpropagation algorithm
- Vanishing Gradients
- Convolutional NNs
- Training loop
- Data augmentation
- L1 & L2 regularisation
- Lasso regression
- Ridge regression
- Weight decay
- Dropout
- Basic ensembling
- Bagging
- Boosting
- Random Forest
- XGBoost
- Stacking & Blending

Ensemble learning
- Bagging
- Boosting
- Random Forest
- XGBoost
- Stacking & Blending

Other
- Regularisation
- Accuracy
- Recall, precision
- F1-score
- Confusion matrix

Performance metrics
- Six homeworks
  (10 points each)
- Paper review
  (15 points)

Now you are ready for real-life tasks
Machine Learning
Machine Learning

Supervised Learning

Unsupervised Learning
Supervised Learning

Unsupervised Learning

Reinforcement Learning

Machine Learning
Supervised Learning

Unsupervised Learning

Reinforcement Learning

Machine Learning
Supervised Learning

Unsupervised Learning

Reinforcement Learning

Machine Learning
Machine Learning

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
Machine Learning

Deep Learning

Supervised Learning

Unsupervised Learning

Reinforcement Learning
This handsome man will talk to you real Deep Learning (soon)

I am here to scratch the surface :)

I am here to scratch the surface :)
The image shows a graph with two axes, labeled $X_1$ and $X_2$. The graph plots points at various coordinates. A vector with coordinates $(5, 5)$ is shown, indicating that the sum of inputs $(X_1, X_2)$ is $(5, 5)$. The graph also includes a notation for the sum of inputs.
Sum of inputs

\[ X_1 \quad 5 \]

\[ X_2 \quad 5 \]

\[ \rightarrow 10 \]
Weights
Weights are connections between nodes of the network.
Weights are connections between nodes of the network. Weights are not necessarily positive integers.

$$X_1 \times 0.12$$

$$X_2 \times -0.1$$

Sum of inputs
Sum of inputs

\[ X_1 \times 0.12 \]

\[ X_2 \times -0.1 \]
Sum of inputs $x 0.12 \quad x -0.1 \quad \rightarrow 0.1$
The diagram represents a neural network function. The inputs $X_1$ and $X_2$ are multiplied by weights $0.12$ and $-0.1$ respectively, and then summed to produce an output of $0.1$. The activation function is applied to this sum to produce the final output.
The graph shows the relationship between the inputs $X_1$ and $X_2$ and the output of the sum of these inputs multiplied by 0.12 and -0.1, respectively. The activation function applied to the sum of inputs is a step function, where if the sum is greater than 0, the output is 0.1. This is illustrated by the points on the graph and the step function diagram.
The diagram illustrates a classification task in machine learning. The inputs $X_1$ and $X_2$ are plotted on the graph. The sum of the inputs is calculated as:

$$\text{Sum of inputs} = X_1 \times 0.12 + X_2 \times -0.1$$

This sum is then passed through a step activation function, which is a non-linear function that outputs 0.1 if the sum is greater than 0. The diagram also shows a step function for the activation, indicating the decision boundary between the blue class and the red class.
The image shows a graph with two axes, $X_1$ and $X_2$, and a step function with the following explanation:

- **Step Activation Function:**
  - The sum of inputs is calculated as $X_1 \times 0.12 + X_2 \times -0.1$.
  - If the sum is greater than 0, it is activated as 0.1.

The graph illustrates a plot of points on the $X_1$ and $X_2$ axes, with the red class and blue class indicated by different colors.
X_1 \times 0.12 \\
X_2 \times -0.1 \\
Sum of inputs \\
Step activation function \\
> 0 \\
Red class

Activation function
Artificial neuron (perceptron)

Sum of inputs

\[ \times 0.12 \]

\[ \times -0.1 \]

\[ + \rightarrow 0.1 \]

Activation function

Step activation function

\[ > 0 \]

Red
**Biological neuron**

- Dendrites
- Axon

**Artificial neuron** (perceptron)

- Sum of inputs
  - $x \times 0.12$
  - $x \times -0.1$
- $\rightarrow 0.1$
- Step activation function
  - $> 0$
- Activation function
Sum of inputs $x 0.12$ $x -0.1$.

Activation function:
- Step function

Red

Activation function
Sum of inputs \times 0.12 \times -0.1 > 0

Activation function

Step function

Activation function
The diagram illustrates a step activation function with inputs $X_1$ and $X_2$. The inputs are multiplied by weights: $X_1 \times 0.12$ and $X_2 \times -0.1$. The sum of these inputs is calculated as $X_1 \times 0.12 + X_2 \times -0.1 = -0.16$. The step activation function then applies a threshold of $0$, with the output being $0$ if the sum is greater than $0$. In this case, the output is $-0.16$. The graph shows the position of the inputs on the $X_1$ and $X_2$ axes.
Activation function

Step activation function

Sum of inputs

$X_1 	imes 0.12$

$X_2 	imes -0.1$

-0.16

> 0

Blue

Activation function
Same procedure should be performed for **all data** points.
Same procedure should be performed for **all data** points.

\[ X_1 \times 0.12 \]
\[ X_2 \times -0.1 \]

**Sum of inputs**

\[ \text{Step activation function} \]

> 0

**Activation function**
Same procedure should be performed for **all data** points.

For each data point, the inputs $X_1$ and $X_2$ are calculated as follows:

- $X_1$ is multiplied by 0.12.
- $X_2$ is multiplied by -0.1.

The sum of these inputs is then calculated:

$$X_1 \times 0.12 + X_2 \times (-0.1)$$

The **activation function** used is a step function, which is evaluated as follows:

- If the sum of inputs is greater than 0, then the output is 1.
- Otherwise, the output is 0.

The graph shows the relationship between $X_1$ and $X_2$ for data points labeled as **Blue** and **Red**.
Blue

Incorrectly predicted points

Red

\[ X_1 \times 0.12 \]

\[ X_2 \times -0.1 \]

Sum of inputs

Step activation function

\[ > 0 \]

Activation function
Activation function

Step activation function

Decision boundary separating classes is controlled by weights

Sum of inputs

$x \times 0.12$

$x \times -0.1$

Activation function

$> 0$
By changing weights we can **increase** or **decrease** the influence of input features, hence **rotate** the **decision boundary**.

**Decision boundary** separating classes is controlled by weights.

By changing weights we can **increase** or **decrease** the influence of input features, hence **rotate** the **decision boundary**.
By changing weights we can **increase** or **decrease** the influence of input features, hence **rotate** the decision boundary.

Decision boundary separating classes is controlled by weights.

Good weights are found using **backpropagation** algorithm, which we will examine further.
In this case **linear decision boundary** won’t help.

**Decision boundary** separating classes is controlled by weights.

**Sum** of inputs

- $X_1 \times 0.12$
- $X_2 \times -0.1$

**Step activation function**

$$> 0$$

**Activation function**
In this case linear decision boundary won’t help.
In this case **linear decision boundary** won’t help.

Using more **neurons** and …
In this case **linear decision boundary** won’t help.

Using more **neurons** and **non-linear** activation function

\[
sig(x) = \frac{1}{1 + e^{-x}}
\]
Sigmoid activation function

\[ \text{sig}(x) \]
$\text{sig}(x) = \frac{1}{1 + e^{-x}}$
$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$
\[ \text{sig}(x) = \frac{1}{1 + e^{-x}} \]
The Sigmoid activation function is defined as:

\[ \text{sig}(x) = \frac{1}{1 + e^{-x}} \]
The Sigmoid activation function is defined as:

$$\text{sig}(x) = \frac{1}{1 + e^{-x}}$$
$\text{sig}(x) = \frac{1}{1 + e^{-x}}$
The Sigmoid activation function is defined as:

\[ \text{sig}(x) = \frac{1}{1 + e^{-x}} \]
No matter how big your \( x \) can grow, your \( \text{sig}(x) \) will always be between 0 and 1.
In this case **linear decision boundary** won’t help.

Using more **neurons** and **non-linear** activation function

\[ R(x) = \max(0, x) \]
$R(x) = \max(0, x)$
In this case **linear decision boundary** won’t help.

Using more **neurons** and **non-linear** activation function.
In this case **linear decision boundary** won’t help.

Using more **neurons** and **non-linear** activation function

It is possible to obtain non-linear **decision boundary**
Consider this 2D example
Consider this **2D** example

\[
(X_1, X_2)
\]
Consider this **2D** example.
Consider this 2D example.
Consider this 2D example
Point coordinates as input

$$(x_1, x_2)$$
Point coordinates as input

Input layer

$(x_1, x_2)$
Point coordinates as **input**
Point coordinates as input

\[(x_1, x_2)\]
Point coordinates as input

Hidden layer
Point coordinates as input

Hidden layer

Sigmoidal activation function

\[ \text{sig}(x) \]
Point coordinates as input

$(x_1, x_2)$

$X_1$ $X_2$
Point coordinates as input

Output layer
Score for the **first** class

Class scores

\[(x_1, x_2)\]
Score for the **first** class

Score for the **second** class

Class scores

out_{o1}

out_{o2}
If $\text{out}_{o1} > \text{out}_{o2}$ neural network predicts the first class.
Our input point belongs to the second class.

If $\text{out}_{o1} > \text{out}_{o2}$ neural network predicts the first class.

If out$_{o1} >$ out$_{o2}$ neural network predicts the first class.
Our input point belongs to the second class.

If \( \text{out}_{o1} > \text{out}_{o2} \) neural network predicts the first class.

Therefore for this point we expect \( \text{out}_{o2} > \text{out}_{o1} \).
Biases
What is the role of **bias** in Neural Networks?
What is the role of **bias** in Neural Networks?
What is the role of bias in Neural Networks?

Source: http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
What is the role of **bias** in Neural Networks?

![Diagram showing the role of bias in Neural Networks](http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf)
What is the role of **bias** in Neural Networks?

Input \( x \) \( \xrightarrow{w_1} \) Output \( \text{sig}(w_1 \ast x) \)

\( w_1 = [0.5, 1.0, 2.0] \)

Source: http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
What is the role of **bias** in Neural Networks?

\[ w_1 = [0.5, 1.0, 2.0] \]

Source: http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
What is the role of bias in Neural Networks?

Input: $\mathbf{x}$

Output: $\text{sig}(w_1 \ast x)$

$w_1 = [0.5, 1.0, 2.0]$

Bias: $b_1 = [-4.0, 0.0, 4.0]$  \quad \text{and} \quad w_1 = [1.0]$

Source: http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
What is the role of **bias** in Neural Networks?

Bias helps to **shift** the resulting curve

Source: http://www.uta.fi/sis/tie/neuro/index/Neurocomputing2.pdf
This particular architecture is **arbitrary** and meant as an **example**.

In practice, people use already existing architectures (e.g. **ResNet**, **GPT-3**, **U-Net** etc.)
When you create a new neural network, first you need to **initialise** weights and biases.
When you create a new neural network, first you need to **initialise** weights and biases.

Sample **normal** distribution

```python
import numpy as np
W = np.random.randn(2,2)
b = np.random.randn(1,2)
```
When you create a new neural network, first you need to **initialise** weights and biases.

Sample **normal** distribution

```python
import numpy as np
W = np.random.randn(2,2)
b = np.random.randn(1,2)
```
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.
Let's calculate **class scores** for each class using **feed-forward path algorithm**

Coordinates of the point

(0.05, 0.1)
Let's calculate **class scores** for each class using **feed-forward path** algorithm.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $in(h_1)$?

Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $in(h_1)$?
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer \( \text{in}(h_1) \)?

\[
\text{in}(h_1) = \ldots + \ldots + \ldots
\]
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

$$\text{in}(h_1) = x_1 \cdot w_1 + x_2 \cdot w_2 + b_{11}$$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

$$\text{in}(h_1) = x_1 \times 0.15 + x_2 \times 0.2 + 0.35$$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $in(h_1)$?

$$in(h_1) = 0.05 \times 0.15 + 0.1 \times 0.2 + 0.35$$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

$\text{in}(h_1) = 0.38$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $in(h_1)$?

$$in(h_1) = 0.38$$

What is the output of the first neuron of the hidden layer $out(h_1)$?

$$out(h_1) = \ldots$$

In the diagram, the input to the first neuron is $0.38$. The output of the first neuron is not explicitly shown, but it can be calculated using the weights and biases as specified in the image.
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

$$\text{in}(h_1) = 0.38$$

What is the output of the first neuron of the hidden layer $\text{out}(h_1)$?

$$\text{out}(h_1) = \text{sig}(\text{in}(h_1))$$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

$\text{in}(h_1) = 0.38$

What is the output of the first neuron of the hidden layer $\text{out}(h_1)$?

$\text{out}(h_1) = \text{sig}(0.38)$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer $\text{in}(h_1)$?

What is the output of the first neuron of the hidden layer $\text{out}(h_1)$?

$\text{in}(h_1) = 0.38$

$\text{out}(h_1) = \text{sig}(0.38)$

$\text{sig}(x)$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the hidden layer \( \text{in}(h_1) \)?

\( \text{in}(h_1) = 0.38 \)

What is the output of the first neuron of the hidden layer \( \text{out}(h_1) \)?

\( \text{out}(h_1) = \text{sig}(0.38) \)

\( \text{sig}(x) \)
Let's calculate class scores for each class using feed-forward path algorithm.

What is the **input** of the first neuron of the hidden layer $\text{in}(h_1)$?

$\text{in}(h_1) = 0.38$

What is the **output** of the first neuron of the hidden layer $\text{out}(h_1)$?

$\text{out}(h_1) = \text{sig}(0.38) = 0.59$

$sig(x)$
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

\[
\begin{align*}
\text{out}(h_1) &= 0.59 \\
\end{align*}
\]
Let's calculate class scores for each class using feed-forward path algorithm.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

What is the input of the second neuron of the hidden layer $in(h_2)$?
Let's calculate **class scores** for each class using **feed-forward path algorithm**

Let's calculate class scores for each class using feed-forward path algorithm.

\[
\text{out}(h_1) = 0.59
\]

What is the input of the second neuron of the hidden layer \(\text{in}(h_2)\)?

\[
\text{in}(h_2) = 0.05 \times 0.25 + 0.1 \times 0.3 + 0.35
\]
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

What is the **input** of the second neuron of the hidden layer \( h_2 \)?

\[
in(h_2) = 0.39
\]
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the second neuron of the hidden layer \( \text{in}(h_2) \)?

\( \text{in}(h_2) = 0.39 \)

What is the output of the second neuron of the hidden layer \( \text{out}(h_2) \)?

\( \text{out}(h_2) = \ldots \)
Let's calculate **class scores** for each class using **feed-forward path algorithm**

Let's calculate class scores for each class using feed-forward path algorithm

\[
\text{out}(h_1) = 0.59
\]

\[
\text{in}(h_2) = 0.39
\]

\[
\text{out}(h_2) = \text{sig}(0.39)
\]

What is the **input** of the second neuron of the hidden layer \(\text{in}(h_2)\)?

What is the **output** of the second neuron of the hidden layer \(\text{out}(h_2)\)?
Let's calculate class scores for each class using feed-forward path algorithm.

Let's calculate class scores for each class using feed-forward path algorithm.

out($h_1$) = 0.59

What is the input of the second neuron of the hidden layer $in(h_2)$?

in($h_2$) = 0.39

What is the output of the second neuron of the hidden layer $out(h_2)$?

out($h_2$) = 0.6
Let's calculate class scores for each class using feed-forward path algorithm.

\[ \text{out}(h_1) = 0.59 \]
\[ \text{out}(h_2) = 0.6 \]
Let's calculate **class scores** for each class using **feed-forward path algorithm**

$$\text{out}(h_1) = 0.59$$

$$\text{out}(h_2) = 0.6$$
Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the output layer in(o1)?

out(h1) = 0.59

out(h2) = 0.6

in(o1) = ?

in(o2) = ?
Let's calculate class scores for each class using feed-forward path algorithm.

Let's calculate class scores for each class using feed-forward path algorithm.

What is the input of the first neuron of the output layer in(o₁)?

out(h₁) = 0.59

out(h₂) = 0.6

in(o₁) = ...

out(h₁) = 0.59

0.15 (w₁)

0.20 (w₂)

0.30 (w₄)

0.35 (w₆)

0.40 (w₅)

0.45 (w₆)

0.50 (w₇)

0.55 (w₈)

0.6 (b₂₁)

0.35 (b₁₁)

0.35 (b₁₂)

0.6 (b₂₂)

0.0

0.0

0.0

0.0

0.05

0.05

0.05

0.05

0.1

0.1

0.1

0.1

x₁

x₂

0.0

0.0

0.0

0.0

(0.05, 0.1)

(0.05, 0.1)

What is the input of the first neuron of the output layer in(o₁)?
What is the **input** of the first neuron of the output layer?

\[
\text{in}(o_1) = 1.1
\]

\[
\text{in}(o_1) = 1.1
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_2) = 0.6
\]

\[
\text{out}(h_1) = 0.59
\]

\[
\text{out}(h_1) = 0.59
\]
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Let's calculate class scores for each class using feed-forward path algorithm.

\[
\begin{align*}
\text{out}(h_1) &= 0.59 \\
\text{in}(o_1) &= 1.1 \\
\text{out}(h_2) &= 0.6 \\
\end{align*}
\]
Let's calculate class scores for each class using feed-forward path algorithm.

Let's calculate the class score for the first class:

\[ \text{out}(o_1) = \text{in}(o_1) \times \text{out}(h_1) \]

\[ \text{in}(o_1) = 1.1 \]
\[ \text{out}(h_1) = 0.59 \]

\[ \text{out}(o_1) = 1.1 \times 0.59 \]

\[ \text{out}(o_1) = 0.649 \]

What is the output of the first neuron of the output layer, \( \text{out}(o_1) \)?
What is the output of the first neuron of the output layer out(o₁)?

Let's calculate class scores for each class using feed-forward path algorithm.

out(h₁) = 0.59
out(h₂) = 0.6
in(o₁) = 1.1

What is the output of the first neuron of the output layer out(o₁)?

Again sigmoid.
Let's calculate class scores for each class using feed-forward path algorithm.

What is the output of the first neuron of the output layer \( \text{out}(o_1) \)?

\[
\text{in}(o_1) = 1.1
\]

Again sigmoid

\[
\text{out}(o_1) = \text{sig}(1.1)
\]
Let's calculate class scores for each class using feed-forward path algorithm.

What is the **output** of the first neuron of the output layer, \( \text{out}(o_1) \)?

\[
\begin{align*}
\text{in}(o_1) &= 1.1 \\
\text{out}(o_1) &= 0.75
\end{align*}
\]

Again, **sigmoid**
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

\[
\text{out}(h_1) = 0.59 \quad \text{in}(o_1) = 1.1
\]
\[
\text{out}(o_1) = 0.75
\]

Let's calculate **class scores** for each class using **feed-forward path** algorithm.
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using matrices.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

![Diagram showing feed-forward path](image)

**Input matrix**

**Often in practice, feed-forward path is performed using matrices**
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let’s calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Often in practice, feed-forward path is performed using **matrices**.
Let's calculate **class scores** for each class using **feed-forward path** algorithm.

Let's calculate class scores for each class.

- **out**(h) = 0.59
- **in**(o) = 1.1
- **out**(o) = 0.75
- **out**(o) = 0.77

\[
\begin{align*}
\text{out}(h_1) &= 0.59 \\
\text{in}(o_1) &= 1.1 \\
\text{out}(o_1) &= 0.75 \\
\text{out}(o_2) &= 0.77
\end{align*}
\]
How good are the class predictions (scores)?

\[\text{out}(h_1) = 0.59\]
\[\text{in}(o_1) = 1.1\]
\[\text{out}(o_1) = 0.75\]

\[\text{out}(h_2) = 0.6\]
\[\text{out}(o_2) = 0.77\]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Our input point belongs to the second class.
How good are the class **predictions** (scores)?

Our input point belongs to the **second** class.

Therefore for this point we **expect**

out(o₁) to be very close to 0

out(o₂) to be very close to 1
How good are the class predictions (scores)?

Our input point belongs to the second class.

Therefore for this point we expect:

- \( \text{out}(o_1) = 0.75 \)
- \( \text{out}(o_2) = 0.77 \)
- \( \text{out}(o_2) \) to be very close to 1
- \( \text{out}(o_1) \) to be very close to 0

These expected values for each input point are stored in a variable called ground truth or truth for short. These are labels in supervised learning terms.
How good are the class **predictions** (scores)?

Let’s calculate the **error** for these predictions (**MSE**).

Our input point belongs to the **second** class.
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = E_{o1} + E_{o2} \]

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>truth = 0</td>
</tr>
<tr>
<td>0.77</td>
<td>truth = 1</td>
</tr>
</tbody>
</table>
How good are the class **predictions** (scores)?

Our **input** point belongs to the **second** class.

Let’s calculate the **error** for these predictions (**MSE**).

\[
E_{total} = E_{o1} + E_{o2}
\]

\[
E_{o1} = 
\]

\[
E_{o1} = 0.75
\]

truth = 0

prediction = 0.75

error

\[
in(o_1) = 1.1
\]

\[
out(h_1) = 0.59
\]

\[
out(h_2) = 0.6
\]

\[
in(o_2) = 1.1
\]

\[
out(h_2) = 0.6
\]

\[
out(h_2) = 0.6
\]

\[
in(o_2) = 1.1
\]

\[
out(h_2) = 0.6
\]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = E_{o1} + E_{o2} \]

\[ E_{o1} = \frac{1}{2} (\text{truth} - \text{out}_{o1})^2 \]
How good are the class **predictions** (scores)?

Our input point belongs to the **second** class.

Let's calculate the **error** for these predictions (**MSE**)

\[
E_{total} = E_{o1} + E_{o2}
\]

\[
E_{o1} = \frac{1}{2} (true - out_{o1})^2
\]
How good are the class **predictions** (scores)?

Our **input** point belongs to the **second** class.

Let’s calculate the **error** for these predictions (**MSE**)

$$E_{total} = E_{o1} + E_{o2}$$

$$E_{o1} = \frac{1}{2}(\text{truth} - 0.75)^2$$
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[
E_{\text{total}} = E_{o_1} + E_{o_2}
\]

\[
E_{o_1} = \frac{1}{2} (0 - 0.75)^2
\]

prediction | error
---|---
0.75 | truth = 0
0.77 | truth = 1
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[ E_{total} = E_{o1} + E_{o2} \]

\[ E_{o1} = 0.28 \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[ E_{total} = 0.28 + E_{o2} \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = 0.28 + E_{o2} \]

Error on the first output neuron.

- Prediction: 0.75
  - Error: 0.28 (truth = 0)

- Prediction: 0.77
  - Error: 0.28 (truth = 1)
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = 0.28 + E_{o2} \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[ E_{total} = 0.28 + E_{o2} \]

\[ E_{o2} = \frac{1}{2}(truth - out_{o2})^2 \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = 0.28 + E_{o2} \]

\[ E_{o2} = \frac{1}{2}(1 - 0.77)^2 \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[ E_{\text{total}} = 0.28 + E_{o2} \]

\[ E_{o2} = 0.025 \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE).

\[ E_{total} = 0.28 + 0.025 \]
How good are the class predictions (scores)?

Our input point belongs to the second class.

Let's calculate the error for these predictions (MSE):

\[ E_{total} = 0.305 \]
What can we do to make the error **smaller**?
What can we do to make the error smaller?

We need to make predictions match the truth.

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Error</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.28</td>
<td>0</td>
</tr>
<tr>
<td>0.77</td>
<td>0.025</td>
<td>1</td>
</tr>
</tbody>
</table>
What can we do to make the error **smaller**?

\[
\begin{align*}
\text{in}(o_1) &= 1.1 \\
\text{out}(h_1) &= 0.59 \\
\text{prediction} &= 0.75 \\
\text{error} &= 0.28 \\
\text{truth} &= 0 \\
\text{prediction} &= 0.77 \\
\text{error} &= 0.025 \\
\text{truth} &= 1
\end{align*}
\]
What can we do to make the error **smaller**?

The only way is to change **weights** and **biases**.

The diagram illustrates the neural network's computation, showing the inputs, weights, bias, and outputs. The prediction errors for each output are calculated as follows:

- For output $h_1$, the prediction is $0.59$, and the error is $0.28$ (truth = 0).
- For output $h_2$, the prediction is $0.6$, and the error is $0.025$ (truth = 1).

The network's weights and biases are labeled for clarity.
What can we do to make the error **smaller**?

Can we change the inputs?

<table>
<thead>
<tr>
<th>prediction</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>truth = 0</td>
<td>0.28</td>
</tr>
<tr>
<td>truth = 1</td>
<td>0.025</td>
</tr>
</tbody>
</table>
What can we do to make the error **smaller**?

Can we change the **inputs**?

Yes, but this will compromise the data.

<table>
<thead>
<tr>
<th>prediction</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.28</td>
</tr>
<tr>
<td>truth = 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>prediction</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.77</td>
<td>0.025</td>
</tr>
<tr>
<td>truth = 1</td>
<td></td>
</tr>
</tbody>
</table>
What can we do to make the error **smaller**?

The only way is to change **weights** and **biases**.
What can we do to make the error **smaller**?

Let’s try to reduce **total error** by changing $w_5$.
What can we do to make the error **smaller**?

Let’s try to reduce **total error** by changing $w_5$.
What can we do to make the error **smaller**?

Let’s try to reduce **total error** by changing $w_5$

$$E_{total} = 0.28 + 0.025$$
What can we do to make the error **smaller**?

Let’s try to reduce **total error** by changing $w_5$

$$E_{\text{total}} = 0.28 + 0.025$$
What can we do to make the error **smaller**?

Let’s try to reduce **total** error by changing $w_5$

$$E_{total} = 0.29 + 0.025$$
What can we do to make the error **smaller**?

Let’s try to reduce total error by changing $w_5$

$$E_{total} = 0.28 + 0.025$$
What can we do to make the error **smaller**?

Let's try to reduce **total error** by changing $w_5$

$$E_{total} = 0.278 + 0.025$$
What can we do to make the error smaller?

Let's try to reduce total error by changing $w_5$

$$E_{\text{total}} = 0.273 + 0.025$$
What can we do to make the error **smaller**?

Let’s try to reduce **total** error by changing $w_5$

$$E_{total} = 0.273 + 0.025$$

Every time we change a weight, have to recompute the **feed-forward path**
What can we do to make the error smaller?

Every time we change a weight, have to recompute the feed-forward path.
We want a way to **efficiently update all our weights** so that **total error decreases** most substantially.
We want a way to **efficiently update all our weights** so that **total error decreases** most substantially.

We want to compute the **gradient**

\[ E_{\text{total}} \]

\[ \text{weight - 1} \quad \text{weight + 1} \]
We want a way to **efficiently update all our weights** so that **total error decreases** most substantially.

We want to compute the **gradient**

**Gradient** shows the direction of the **greatest increase** of the function.
We want a way to **efficiently update all our weights** so that **total error decreases** most substantially.

\[
\frac{\partial E}{\partial w}
\]

**Gradient** shows the direction of the **greatest increase** of the function.
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$.
Let's try to reduce total \textbf{error} by changing $w_5$

We need to find a \textbf{gradient} $E_{\text{total}}$ with respect to $w_5$

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = \ldots
\]
Let's try to reduce total error by changing $w_5$.

We need to find a gradient $E_{total}$ with respect to $w_5$.

$$\frac{\partial E_{total}}{\partial w_5} = \ldots$$

Will show the direction of the increase of the function.
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

We need to find a gradient $E_{total}$ with respect to $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = \ldots$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

We need to find a gradient $E_{total}$ with respect to $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \ldots$$
Let’s try to reduce total error by changing $w_5$.

We need to find a gradient $E_{\text{total}}$ with respect to $w_5$:

$$\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial E_{o_1}} \ast \ldots$$

The backpropagation algorithm:

$$E_{\text{total}} = E_{o_1} + E_{o_2}$$

Prediction 0.75, truth 0.28
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

We need to find a gradient $E_{total}$ with respect to $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \ast \ldots$$

$$E_{total} = E_{o1} + E_{o2}$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

We need to find a gradient $E_{total}$ with respect to $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \ldots$$
Let’s try to reduce total error by changing $w_5$.

We need to find a gradient $E_{total}$ with respect to $w_5$.

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$

Backpropagation algorithm
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o_1} + E_{o_2}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o_1}} \cdot \frac{\partial E_{o_1}}{\partial o_{out_1}} \cdot \frac{\partial o_{out_1}}{\partial o_{in_1}} \cdot \frac{\partial o_{in_1}}{\partial w_5}$$
Let's try to reduce total error by changing $w_5$.
Backpropagation algorithm

Let’s try to reduce **total error** by changing \( w_5 \)

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}
\]

\[ E_{total} = E_{o1} + E_{o2} \]
Let’s try to reduce **total error** by changing \( w_5 \)

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}
\]

\[
E_{total} = E_{o1} + E_{o2}
\]
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{\text{total}} = E_{o_1} + E_{o_2}$$

$$\frac{\partial E_{\text{total}}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial E_{o_1}} * \frac{\partial E_{o_1}}{\partial \text{out}_o} * \frac{\partial \text{out}_o}{\partial \text{in}_o} * \frac{\partial \text{in}_o}{\partial w_5}$$

$$\frac{\partial E_{\text{total}}}{\partial E_{o_1}} = \frac{\partial}{\partial E_{o_1}}(E_{o_1} + E_{o_2})$$
Let's try to reduce total error by changing $w_5$.
Backpropagation algorithm

Let's try to reduce **total error** by changing \( w_5 \)

\[
E_{total} = E_{o1} + E_{o2}
\]

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}
\]

\[
f(x, y) = x + y
\]

\[
\frac{\partial E_{total}}{\partial E_{o1}} = \frac{\partial}{\partial E_{o1}}(E_{o1} + E_{o2})
\]
Let's try to reduce total error by changing $w_5$.

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$

$$f(x, y) = x + y$$

$$\frac{\partial f(x, y)}{\partial x} = \frac{\partial}{\partial x}(x + y)$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o_1}} \ast \frac{\partial E_{o_1}}{\partial out_{o_1}} \ast \frac{\partial out_{o_1}}{\partial in_{o_1}} \ast \frac{\partial in_{o_1}}{\partial w_5}
\]

\[
f(x, y) = x + y
\]

\[
\frac{\partial f(x, y)}{\partial x} = \frac{\partial}{\partial x} x + \frac{\partial}{\partial x} y
\]
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o_1} + E_{o_2}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o_1}} \times \frac{\partial E_{o_1}}{\partial out_{o_1}} \times \frac{\partial out_{o_1}}{\partial in_{o_1}} \times \frac{\partial in_{o_1}}{\partial w_5}$$

$$f(x, y) = x + y$$

$$\frac{\partial f(x, y)}{\partial x} = \frac{\partial}{\partial x} x + 0$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \cdot \frac{\partial E_{o1}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial in_{o1}} \cdot \frac{\partial in_{o1}}{\partial w_5}$$

$$f(x, y) = x + y$$

$$\frac{\partial f(x, y)}{\partial x} = 1$$
Backpropagation algorithm

Let’s try to reduce **total error** by changing \( w_5 \)

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o_1}} \ast \frac{\partial E_{o_1}}{\partial out_{o_1}} \ast \frac{\partial out_{o_1}}{\partial in_{o_1}} \ast \frac{\partial in_{o_1}}{\partial w_5}
\]

\[
E_{total} = E_{o_1} + E_{o_2}
\]

\[
\frac{\partial E_{total}}{\partial E_{o_1}} = 1
\]
**Backpropagation algorithm**

Let’s try to reduce **total error** by changing $w_5$

$E_{total} = E_{o1} + E_{o2}$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$

What does this **actually** say?

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial E_{o1}} = 1$$
If we increase $E_{o1}$ by 1 what does this say?

$E_{\text{total}} = E_{o1} + E_{o2}$

$\frac{\partial E_{\text{total}}}{\partial E_{o1}} = 1$
If we increase $E_{o1}$ by 1

\[ E_{total} = (E_{o1} + 1) + E_{o2} \]

\[ \frac{\partial E_{total}}{\partial E_{o1}} = 1 \]
If we increase $E_{o1}$ by 1

$$E_{total} = (E_{o1} + E_{o2}) + 1$$

$$\frac{\partial E_{total}}{\partial E_{o1}} = 1$$
What does this actually say?

If we increase $E_{o1}$ by 1

$$E_{total} = E_{total} + 1$$

$E_{total}$ would also increase by 1

$$\frac{\partial E_{total}}{\partial E_{o1}} = 1$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \star \frac{\partial E_{o1}}{\partial out_{o1}} \star \frac{\partial out_{o1}}{\partial in_{o1}} \star \frac{\partial in_{o1}}{\partial w_5}$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

\[
\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial E_{o1}} \times \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}
\]

We know this value
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$
\frac{\partial E_{total}}{\partial w_5} = 1 \cdot \frac{\partial E_{o_1}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} \cdot \frac{\partial in_{o_1}}{\partial w_5}
$$

$$
E_{o_1} = \frac{1}{2} (\text{truth} - out_{o_1})^2
$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 * \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial in_{o1}} * \frac{\partial in_{o1}}{\partial w_5}$$

$$E_{o1} = \frac{1}{2}(\text{truth} - out_{o1})^2$$

$$\frac{\partial E_{o1}}{\partial out_{o1}} = \frac{\partial}{\partial out_{o1}} \left(\frac{1}{2}(\text{truth} - out_{o1})^2\right)$$
Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \ast \frac{\partial E_{o1}}{\partial out_{o1}} \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$

$$E_{o1} = \frac{1}{2}(x - y)^2$$

$$\frac{\partial E_{o1}}{\partial y} = \frac{\partial}{\partial y} \left( \frac{1}{2}(x - y)^2 \right)$$

Backpropagation algorithm
Backpropagation algorithm

Let's try to reduce total error by changing \( w_5 \)

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times \frac{\partial E_{o1}}{\partial out_{o1}} \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}
\]

\[
E_{o1} = \frac{1}{2} (x - y)^2
\]

\[
\frac{\partial E_{o1}}{\partial y} = \left( \frac{1}{2} \times 2(x - y) \right) \times \frac{\partial}{\partial y} (x - y)
\]
Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 * \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial in_{o1}} * \frac{\partial in_{o1}}{\partial w_5}$$

$$E_{o1} = \frac{1}{2} (x - y)^2$$

$$\frac{\partial E_{o1}}{\partial y} = \left( \frac{1}{2} * 2(x - y) \right) * (-1)$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \ast \frac{\partial E_{o1}}{\partial \text{out}_{o1}} \ast \frac{\partial \text{out}_{o1}}{\partial \text{in}_{o1}} \ast \frac{\partial \text{in}_{o1}}{\partial w_5}$$

$$E_{o1} = \frac{1}{2} (x - y)^2$$

$$\frac{\partial E_{o1}}{\partial y} = (x - y) \ast (-1)$$
Let’s try to reduce total error by changing $w_5$.

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 1 \times \frac{\partial E_{o_1}}{\partial \text{out}_{o_1}} \times \frac{\partial \text{out}_{o_1}}{\partial \text{in}_{o_1}} \times \frac{\partial \text{in}_{o_1}}{\partial w_5}
\]

\[
E_{o_1} = \frac{1}{2} (x - y)^2
\]

\[
\frac{\partial E_{o_1}}{\partial y} = y - x
\]

Backpropagation algorithm

\[
E_{\text{total}} = E_{o_1} + E_{o_2}
\]

$\text{in}(o_1) = 1.1$

$\text{out}_{o_1} = 0.75$

$E_{o_1} = 0.28$

truth = 0
Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 * \frac{\partial E_{o1}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial in_{o1}} * \frac{\partial in_{o1}}{\partial w_5}$$

$$E_{o1} = \frac{1}{2}(truth - out_{o1})^2$$

$$\frac{\partial E_{o1}}{\partial y} = out_{o1} - truth$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$
\frac{\partial E_{\text{total}}}{\partial w_5} = 1 \cdot \frac{\partial E_{o_1}}{\partial \text{out}_{o_1}} \cdot \frac{\partial \text{out}_{o_1}}{\partial \text{in}_{o_1}} \cdot \frac{\partial \text{in}_{o_1}}{\partial w_5}
$$

$$
E_{o_1} = \frac{1}{2} (\text{truth} - \text{out}_{o_1})^2
$$

$$
\frac{\partial E_{o_1}}{\partial y} = 0.75 - 0
$$
Let's try to reduce total error by changing $w_5$

Backpropagation algorithm

$$E_{total} = E_{o_1} + E_{o_2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \cdot \frac{\partial E_{o_1}}{\partial out_{o_1}} \cdot \frac{\partial out_{o_1}}{\partial in_{o_1}} \cdot \frac{\partial in_{o_1}}{\partial w_5}$$

$$E_{o_1} = \frac{1}{2} (truth - out_{o_1})^2$$

$$\frac{\partial E_{o_1}}{\partial y} = 0.75$$
Let’s try to reduce **total error** by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \cdot \frac{\partial E_{o1}}{\partial out_{o1}} \cdot \frac{\partial out_{o1}}{\partial in_{o1}} \cdot \frac{\partial in_{o1}}{\partial w_5}$$

We know this value
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \ast 0.75 \ast \frac{\partial out_{o1}}{\partial in_{o1}} \ast \frac{\partial in_{o1}}{\partial w_5}$$

We know this value
Backpropagation algorithm

Let's try to reduce **total error** by changing $w_5$

$$
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}
$$
Let's try to reduce **total error** by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \cdot 0.75 \cdot \frac{\partial out_{o1}}{\partial in_{o1}} \cdot \frac{\partial in_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-in_{o1}}}$$

$$\frac{\partial out_{o1}}{\partial in_{o1}} = \frac{\partial}{\partial in_{o1}} \left( \frac{1}{1 + e^{-in_{o1}}} \right)$$
Let's try to reduce total error by changing $w_5$.

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-in_{o1}}}$$

There is a shortcut:

$$\frac{\partial out_{o1}}{\partial in_{o1}} = \frac{\partial}{\partial in_{o1}} \left( \frac{1}{1 + e^{-in_{o1}}} \right)$$
**Backpropagation algorithm**

Let’s try to reduce **total error** by changing $w_5$

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}
\]

\[
out_{o1} = \frac{1}{1 + e^{-in_{o1}}}
\]

There is a shortcut

\[
\frac{\partial out_{o1}}{\partial in_{o1}} = out_{o1} \times (1 - out_{o1})
\]
Let's try to reduce total error by changing $w_5$

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times \frac{\partial out_{o1}}{\partial in_{o1}} \times \frac{\partial in_{o1}}{\partial w_5}
\]

\[
out_{o1} = \frac{1}{1 + e^{-in_{o1}}}
\]

There is a shortcut

\[
\frac{\partial out_{o1}}{\partial in_{o1}} = 0.75 \times (1 - 0.75)
\]
**Backpropagation algorithm**

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \cdot 0.75 \cdot \frac{\partial out_{o1}}{\partial in_{o1}} \cdot \frac{\partial in_{o1}}{\partial w_5}$$

$$out_{o1} = \frac{1}{1 + e^{-in_{o1}}}$$

There is a shortcut

$$\frac{\partial out_{o1}}{\partial in_{o1}} = 0.186$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \cdot 0.75 \cdot 0.186 \cdot \frac{\partial in_{o1}}{\partial w_5}$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5}$$
Backpropagation algorithm

Let’s try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5}$$
Let’s try to reduce **total error** by changing \( w_5 \)

**Backpropagation algorithm**

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o_1}}{\partial w_5}
\]

\[
in_{o_1} = out_{h_1} \times w_5 + out_{h_2} \times w_6 + b_{21}
\]
Let's try to reduce **total error** by changing $w_5$.

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o_1}}{\partial w_5}
\]

\[
in_{o_1} = out_{h_1} \times w_5 + out_{h_2} \times w_6 + b_{21}
\]

\[
\frac{\partial in_{o_1}}{\partial w_5} = \frac{\partial E_{\text{total}}}{\partial w_5}
\]
Backpropagation algorithm

Let's try to reduce **total error** by changing $w_5$

\[
\frac{\partial E_{\text{total}}}{\partial w_5} = 1 * 0.75 * 0.186 * \frac{\partial in_{o1}}{\partial w_5}
\]

\[
in_{o1} = out_{h1} * w_5 + out_{h2} * w_6 + b_{21}
\]

\[
\frac{\partial in_{o1}}{\partial w_5} = \frac{\partial}{\partial w_5} (out_{h1} * w_5 + out_{h2} * w_6 + b_{21})
\]
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

$$E_{total} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{total}}{\partial w_5} = 1 * 0.75 * 0.186 * \frac{\partial in_{o1}}{\partial w_5}$$

$$in_{o1} = out_{h1} * w_5 + out_{h2} * w_6 + b_{21}$$

$$\frac{\partial in_{o1}}{\partial w_5} = \frac{\partial}{\partial w_5}(out_{h1} * w_5) + \frac{\partial}{\partial w_5}(out_{h2} * w_6) + \frac{\partial}{\partial w_5}(b_{21})$$
Let’s try to reduce total error by changing $w_5$.

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5}
\]

\[
in_{o1} = out_{h1} \times w_5 + out_{h2} \times w_6 + b_{21}
\]

\[
\frac{\partial in_{o1}}{\partial w_5} = \frac{\partial}{\partial w_5} (out_{h1} \times w_5) + \frac{\partial}{\partial w_5} (out_{h2} \times w_6) + 0
\]
Let’s try to reduce total error by changing $w_5$.

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5}$$

$$in_{o1} = out_{h_1} \times w_5 + out_{h_2} \times w_6 + b_{21}$$

$$\frac{\partial in_{o1}}{\partial w_5} = \frac{\partial}{\partial w_5} (out_{h_1} \times w_5) + 0 + 0$$

Backpropagation algorithm
Backpropagation algorithm

Let’s try to reduce total error by changing \( w_5 \)

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5}
\]

\[
in_{o1} = out_{h1} \times w_5 + out_{h2} \times w_6 + b_{21}
\]

\[
\frac{\partial in_{o1}}{\partial w_5} = out_{h1} + 0 + 0
\]
Let’s try to reduce total error by changing $w_5$

\[ E_{total} = E_{o1} + E_{o2} \]

\[ \frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times \frac{\partial in_{o1}}{\partial w_5} \]

\[ \frac{\partial in_{o1}}{\partial w_5} = 0.59 \]

\[ in_{o1} = out_{h1} \times w_5 + out_{h2} \times w_6 + b_{21} \]
Let's try to reduce **total error** by changing $w_5$

$$ \frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59 $$
Let's try to reduce total error by changing $w_5$

$$E_{\text{total}} = E_{o1} + E_{o2}$$

$$\frac{\partial E_{\text{total}}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59$$
Let's try to reduce total error by changing $w_5$

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59 = 0.082$$
Backpropagation algorithm

Let's try to reduce total error by changing $w_5$

What does this actually say?

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59 = 0.082$$
Let's try to reduce the total error by changing \( w_5 \). What does this actually say?

\[
\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59 = 0.082
\]

If we increase \( w_5 \), \( E_{total} \) will increase.
Let's try to reduce total error by changing $w_5$.

What does this actually say?

$$\frac{\partial E_{total}}{\partial w_5} = 1 \times 0.75 \times 0.186 \times 0.59 = 0.082$$

If we increase $w_5$, $E_{total}$ will increase.

So we need to decrease it!
\[ w_5 = w_5 - \frac{\partial E_{\text{total}}}{\partial w_5} \]

So we need to **decrease** it!
If update by a lot, we can miss the optimum

\[ w_5 = w_5 - \frac{\partial E_{\text{total}}}{\partial w_5} \]

So we need to **decrease** it!
So we need to decrease it!

\[ w_5 = w_5 - \frac{\partial E_{total}}{\partial w_5} \]

Thus, we make a bit smaller step
\[ w_5 = w_5 - \eta^* \frac{\partial E_{total}}{\partial w_5} \]

So we need to **decrease** it!

Thus, we make a bit **smaller step**
\[ w_5 = w_5 - \eta * 0.082 \]

Thus, we make a bit smaller step

So we need to decrease it!
$w_5 = 0.4 - \eta \times 0.082$

Thus, we make a bit **smaller step**

So we need to **decrease** it!
\[ w_5 = 0.4 - 0.5 \times 0.082 \]

Thus, we make a bit smaller step

So we need to decrease it!
\[ w_5 = 0.4 - 0.5 \times 0.082 \]

Step size is also called learning rate, and it can be really small

Thus, we make a bit smaller step

So we need to decrease it!
$w_5 = 0.3585$

Thus, we make a bit **smaller step**

So we need to **decrease** it!
What can we do to make the error \textbf{smaller}? 

\[ E_{\text{total}} = 0.28 + 0.025 = 0.305 \]
What can we do to make the error smaller?

\[ w_5 = 0.3585 \]

\[ E_{total} = 0.28 + 0.025 = 0.305 \]
What can we do to make the error **smaller**?

\[ w_5 = 0.3585 \]

\[ E_{total} = 0.28 + 0.025 = 0.305 \]
What can we do to make the error **smaller**?

$$w_5 = 0.3585$$

$$E_{total} = 0.28 + 0.025 = 0.303$$
What can we do to make the error **smaller**?

This looks fine for one weight \( w_5 \) how about all the rest?

\[
E_{\text{total}} = 0.28 + 0.025 = 0.303
\]
\[
\text{out}(h_1) = 0.59 \\
\text{out}(h_2) = 0.6 \\
in(o_1) = 1.1
\]

Prediction error:

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Error</th>
<th>Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.745</td>
<td>0.278</td>
<td>0</td>
</tr>
<tr>
<td>0.77</td>
<td>0.025</td>
<td>1</td>
</tr>
</tbody>
</table>
After updating all remaining weights total error = 0.131
After updating all remaining weights total error = 0.131
Repeating 1000 times decreases it to 0.001
In this example we were dealing with only one weight \(w_5\)

\[ w_5 = 0.3585 \]

This not how it is usually done in practice.

Usually updates for weights in the same layer are computed at the same time using matrix operations.

Let’s see how backpropagation works using \textbf{matrix} notation.
\[ X \times W + b \]

\[
\begin{array}{c|c|c|c|c}
 & x_1 & x_2 & w_1 & w_3 \\
\hline
x_1 & & & w_1 & \\
\hline
x_2 & & & w_2 & w_4 \\
\hline
b_1 & & & b_1 & \\
\hline
b_2 & & & b_2 & \\
\end{array}
\]
\[
X \ast W + b = P \rightarrow E_{total}
\]
How does $W$ influence $E_{total}$?

$X \ast W + b = P \rightarrow E_{total}$

$W = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix}$

$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

$P = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$

$E_T$
How does $W$ influence $E_{total}$?

We use the **chain rule** again!
How does $P$ influence $E_{total}$?

We use the chain rule again!

\[ X \ast W + b = P \]

Predictions ($P$) \hspace{2cm} Truth ($T$)

\[ E_{total} \]

\[ \sum \]

We use the chain rule again!
How does $P$ influence $E_{total}$?

We use the **chain rule** again!

$$X \ast W + b = P \rightarrow E_{total}$$

$$\frac{\partial E_{total}}{\partial P} = \frac{1}{2} \frac{\partial(T - P)^2}{\partial P}$$

Truth ($T$)

Predictions ($P$)

$E_{total}$
How does $P$ influence $E_{total}$?

We use the chain rule again!

$$\frac{\partial E_{total}}{\partial P} = \frac{1}{2} * 2 * (-1) * (T - P)$$

$$X * W + b = P$$
How does $P$ influence $E_{total}$?

We use the **chain rule** again!

\[
\frac{\partial E_{total}}{\partial P} = P - T
\]

We use the chain rule again!
How does $W$ influence $P$?

We use the **chain rule** again!

\[
\begin{align*}
\sum (w_1, x_2, x_1, x_2) &= \sum (w_2, w_3, w_4) \\
\sum &= Predictions \ (P) \\
\end{align*}
\]

\[
E_{total} = \frac{\partial E_{total}}{\partial P} = P - T
\]

\[
X \ast W + b = P
\]

Predictions \ ($P$)
How does $W$ influence $P$?

We use the **chain rule** again!

$$\frac{\partial P}{\partial W} = \frac{\partial (X \ast W + b)}{\partial W}$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$
How does $W$ influence $P$?

We use the **chain rule** again!

$$\frac{\partial P}{\partial W} = X^T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$X \ast W + b = P$

$P_1 \quad P_2$

$E_T$
How does $W$ influence $E_{total}$?

We use the **chain rule** again!

$$\frac{\partial E_{total}}{\partial W} = ?$$

$$\frac{\partial P}{\partial W} = X^T \quad \frac{\partial E_{total}}{\partial P} = P - T$$
How does $W$ influence $E_{total}$?

We use the **chain rule** again!

$$
\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} \ast \frac{\partial P}{\partial W}
$$

$$
\frac{\partial P}{\partial W} = X^T \quad \frac{\partial E_{total}}{\partial P} = P - T
$$

$$
X \ast W + b = P \quad \quad T \quad \quad E_{total}
$$
What is the problem here?

How does $W$ influence $E_{total}$?

$$\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} \ast \frac{\partial P}{\partial W}$$

$$\frac{\partial P}{\partial W} = X^T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$$X \ast W + b = P$$

$$P \rightarrow E_{total}$$
How does $W$ influence $E_{total}$?

What is the problem here?

$[?; ?] = [?; ?] * [?; ?]$

$$\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} * \frac{\partial P}{\partial W}$$

$$\frac{\partial P}{\partial W} = X^T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$$X * W + b = P \quad T \quad E_{total}$$
How does $W$ influence $E_{total}$?

What is the problem here?

$[?;?] = [?;?] * [2; N]$

$$\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} * \frac{\partial P}{\partial W}$$

$$\frac{\partial P}{\partial W} = X^T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$X \ast W + b = P \rightarrow E_{total}$
How does $W$ influence $E_{total}$?

What is the problem here?

\[ [?;?] = [?;?] * [2; N] \]

\[
\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} \times \frac{\partial P}{\partial W}
\]

\[
\frac{\partial P}{\partial W} = X^T \quad \frac{\partial E_{total}}{\partial P} = P - T
\]

\[
X \ast W + b = P 
\]

$E_{total}$
How does $W$ influence $E_{total}$?

What is the problem here?

$[?;?] = [?;2]*[2;N]$

$$\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} \cdot \frac{\partial P}{\partial W}$$

$$\frac{\partial P}{\partial W} = X^T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$$X \cdot W + b = P$$

$$T \rightarrow E_{total}$$
How does $W$ influence $E_{total}$?

What is the problem here?

$$[?;?] = [N; 2] *[2; N]$$

$$\frac{\partial E_{total}}{\partial W} = \left( \frac{\partial E_{total}}{\partial P} \right) * \frac{\partial P}{\partial W}$$

$$= X^T \frac{\partial P}{\partial W} = P - T$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$$X \ast W + b = P \rightarrow E_{total}$$

Truth ($T$)

Predictions ($P$)

$$E_{total}$$
How does $W$ influence $E_{total}$?

What is the problem here?

$[2; 2] = [N; 2]*[2; N]$

$$\frac{\partial E_{total}}{\partial W} = \frac{\partial E_{total}}{\partial P} \ast \frac{\partial P}{\partial W}$$

$$\frac{\partial P}{\partial W} = X^T \quad \frac{\partial E_{total}}{\partial P} = P - T$$

$X \ast W + b = P \quad T \rightarrow E_{total}$
We finally know how \( W \) influence \( E_{total} \! \)!

\[
\frac{\partial E_{total}}{\partial W} = \frac{\partial P}{\partial W} \cdot \frac{\partial E_{total}}{\partial P}
\]

\[
\frac{\partial P}{\partial W} = X^T \quad \frac{\partial E_{total}}{\partial P} = P - T
\]

\[
X \ast W + b = P \quad T \rightarrow \quad E_{total}
\]
How does $W$ influence $P$?

We use the **chain rule** again!

$$\frac{\partial P}{\partial W} = \frac{\partial (X^* W + b)}{\partial W} \quad \frac{\partial E_{total}}{\partial P} = P - T$$

$$X \ast W + b = P$$

$E_{total}$
What else is there?

\[
X \ast W + b = P \quad \text{Predictions (P)}
\]

\[
E_{total}
\]

\[
\Sigma \rightarrow \text{Truth (T)}
\]

\[
(x_1, x_2)
\]

\[
(x_1, x_2)
\]

\[
\begin{array}{|c|c|}
\hline
x_1 & x_2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
w_1 & w_3 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
w_2 & w_4 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
b_1 & b_2 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|}
\hline
P_1 & P_2 \\
\hline
\end{array}
\]

\[
E_T
\]
What else is there? **Biases!**

\[
\Sigma \rightarrow \text{Predictions (P)} \rightarrow \text{Truth (T)}
\]

\[
\begin{align*}
X & \ast W + b = P \\
& \rightarrow E_{total}
\end{align*}
\]

\[
\begin{align*}
b_1 & \quad b_2 \\
P_1 & \quad P_2 \\
E_T &
\end{align*}
\]
In order to estimate how much \( b \) influence \( E_{total} \) we again need to know how much \( E_{total} \) is influenced by \( P \).
In order to estimate how much $b$ influence $E_{total}$ we again need to know how much $E_{total}$ is influenced by $P$.

We already know this!

\[
\frac{\partial E_{total}}{\partial P} = P - T
\]

\[
X * W + b = P \rightarrow E_{total}
\]
How does $b$ influence $P$?

$$
\frac{\partial P}{\partial b} = \frac{\partial(X^* W + b)}{\partial b}
$$

$$
\frac{\partial E_{total}}{\partial P} = P - T
$$

$$
X \ast W + b = P
$$

Truth ($T$)

Predictions ($P$)

$E_{total}$
How does $b$ influence $P$?

$$\frac{\partial P}{\partial b} = 1$$

$$\frac{\partial E_{total}}{\partial P} = P - T$$

$$X \ast W + b = P \rightarrow E_{total}$$
Estimate how much $b$ influence $E_{total}$

\[
\frac{\partial P}{\partial b} = 1
\]

\[
\frac{\partial E_{total}}{\partial P} = P - T
\]
Estimate how much $b$ influence $E_{\text{total}}$

$$\frac{\partial E_{\text{total}}}{\partial b} = \frac{\partial P}{\partial b} \ast \frac{\partial E_{\text{total}}}{\partial P}$$

$$\frac{\partial P}{\partial b} = 1$$

$$\frac{\partial E_{\text{total}}}{\partial P} = P - T$$

$$X * W + b = P \rightarrow E_{\text{total}}$$

Truth ($T$)  Predictions ($P$)  $E_{\text{total}}$
Now that we finally know $db$ and $dW$ we can update original weights and biases.

\[
\begin{bmatrix}
  w_1 & w_3 \\
  w_2 & w_4 \\
\end{bmatrix}
\]
Now that we finally know $db$ and $dW$ we can update original weights and biases

<table>
<thead>
<tr>
<th></th>
<th>$W$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1$</td>
<td>$W_3$</td>
<td></td>
</tr>
<tr>
<td>$W_2$</td>
<td>$W_4$</td>
<td></td>
</tr>
</tbody>
</table>

Updated $W = W - db$
Now that we finally know $db$ and $dW$ we can update original weights and biases.

\[
\begin{align*}
\text{Updated } W &= W - \frac{\partial E_{\text{total}}}{\partial W} \\
\begin{bmatrix}
w_1 & w_3 \\
w_2 & w_4
\end{bmatrix} &= \begin{bmatrix}
w_1 & w_3 \\
w_2 & w_4
\end{bmatrix} - \begin{bmatrix}
dw_1 & dw_3 \\
dw_2 & dw_4
\end{bmatrix}
\end{align*}
\]
Now that we finally know $db$ and $dW$ we can update original weights and biases.

$$\text{Updated} \quad W \quad = \quad W \quad - \quad \eta \quad \cdot \quad \frac{\partial E_{\text{total}}}{\partial W}$$

<table>
<thead>
<tr>
<th>Updated</th>
<th>$W$</th>
<th>$W$</th>
<th>$\frac{\partial E_{\text{total}}}{\partial W}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_1$</td>
<td>$W_3$</td>
<td>$W_1$</td>
<td>$W_3$</td>
</tr>
<tr>
<td>$W_2$</td>
<td>$W_4$</td>
<td>$W_2$</td>
<td>$W_4$</td>
</tr>
<tr>
<td>$\frac{dw_1}{dW}$</td>
<td>$\frac{dw_3}{dW}$</td>
<td>$\frac{dw_1}{dW}$</td>
<td>$\frac{dw_3}{dW}$</td>
</tr>
</tbody>
</table>

Learning rate
Now that we finally know $db$ and $dW$ we can update original weights and biases.

\[
W_{\text{ Updated}} = W - \eta * \frac{\partial E_{\text{total}}}{\partial W}
\]

\[
b_{\text{ Updated}} = b - \eta * \frac{\partial E_{\text{total}}}{\partial b}
\]
What would change if we had an activation function? (e.g. ReLu)
What would change if we had an **activation function**? (e.g. **ReLU**)
What would change if we had an **activation function**? (e.g. **ReLU**)
What would change if we had an activation function? (e.g. ReLu)

All negative predictions will turn to 0

ReLu (X \ast W + b) = P \rightarrow E_{total}

What would change if we had an activation function? (e.g. ReLu)

All negative predictions will turn to 0

ReLu (X \ast W + b) = P \rightarrow E_{total}
What would change if we had an activation function? (e.g. ReLu)

All **negative** predictions will turn to **0**

\[
\text{ReLu} \left( X \ast W + b \right) = P
\]

Backpropagation will also change
We finally know how $W$ influence $E_{total}$!

\[ \frac{\partial E_{total}}{\partial W} = \frac{\partial P}{\partial W} \ast \frac{\partial E_{total}}{\partial P} \]

\[ \frac{\partial P}{\partial W} = X^T \]

\[ \frac{\partial E_{total}}{\partial P} = P - T \]
How does $W$ influence $P$?

$$\frac{\partial P}{\partial W} =$$

$$X \ast W + b = P \rightarrow E_{total}$$

$$P_1 \quad P_2$$
How does $W$ influence $P$?

$\frac{\partial P}{\partial W} = \frac{\partial (\max(0, (X \ast W + b)))}{\partial W}$

$X \ast W + b = P$
How does $W$ influence $P$?

$$\frac{\partial P}{\partial W} = \begin{cases} \frac{\partial (X \ast W + b)}{\partial W} & \text{for } X > 0 \\ 0 & \text{for } X \leq 0 \end{cases}$$

$$X \ast W + b = P$$
How does \( \mathbf{W} \) influence \( \mathbf{P} \)?

\[
\frac{\partial \mathbf{P}}{\partial \mathbf{W}} = \begin{cases} 
X^T & \text{for } X > 0 \\
0 & \text{for } X \leq 0 
\end{cases}
\]

\[
X \ast \mathbf{W} + b = \mathbf{P} \quad \xrightarrow{T} \quad \mathbf{E}_{total}
\]
How does $W$ influence $P$?

This would be $X$ with 0s instead of negative values.

$$\frac{\partial P}{\partial W} = \begin{cases} X^T & \text{for } X > 0 \\ 0 & \text{for } X \leq 0 \end{cases}$$
Training Neural Networks
(part I)

http://playground.tensorflow.org/
2. Choose initialization method
Select an initialization method for the values of your neural network parameters.

- Zero
- Too small
- Appropriate
- Too large

3. Train the network.
Observe the cost function and the decision boundary.

Select whether to visualize the weights or gradients of the network above.

- Weight
- Gradient

https://www.deeplearning.ai/ai-notes-initialization/index.html
Resources:

**Brandon Rohrer’s** youtube video: How Deep Neural Networks Work (https://youtu.be/ILsA4nyG7I0)

Stanford University **CS231n** Convolutional Neural Networks for Visual Recognition (github.io version): http://cs231n.github.io/

**Matt Mazur’s**: A Step by Step Backpropagation Example (https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/)

**Andrej Karpathy’s blog post**: Yes you should understand backprop (https://medium.com/@karpathy/yes-you-should-understand-backprop-e2f06eab496b)

**Raul Vicente’s lecture**: From brain to Deep Learning and back (https://www.uttv.ee/naita?id=23585)

That's all Folks!