Neural Networks Q&A

Oana Cocarascu & Helen Yannakoudakis

Department of Informatics
King's College London

Delta rule vs generalised delta rule

- Q: Difference between delta rule and generalised delta rule.
- The error in the delta rule is defined in terms of s not $g(s)$.

$$E(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{M}(t_j - s_j)^2$$

- Generalised delta rule keeps the transfer function (which has to be differentiable).

$$E(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{M}(t_j - g(s_j))^2$$

Dot product

- Q: What does the dot product measure?
- A: We use the dot product to calculate the weighted sum of the input.
- The dot product of the input (**x**) and weight (**w**) vectors:

$$w_0 x_0 + w_1 x_1 + w_2 x_2$$

- Easier notation, can be computed more efficiently. In practice, matrix multiplication is used as it looks at an entire layer which can be done efficiently using libraries.

Universal function approximator theorem

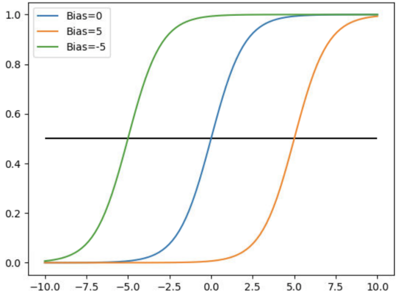
- The universal function approximator theorem means that we would be able to represent continuous functions, but there is no guarantee that we will learn that function (i.e., layer too large and may fail to generalise).

Bias

- Q: How do we find w_0?
- Q: Why would we want to introduce a bias, if the values in the network can be fine-tuned?
- Q: Relation between w_0 and threshold (on values in slide 14).

Bias

- Bias shifts the activation function (to left or right).

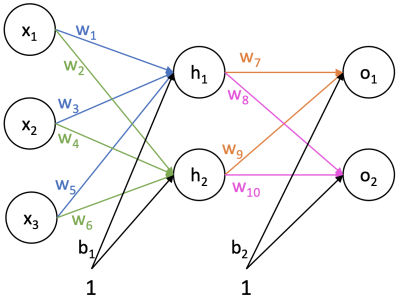


- Oversimplification: a NN that needs to output 3 when $x = 0$. How will you do that since $xw = 0$? Add bias 3.

Bias

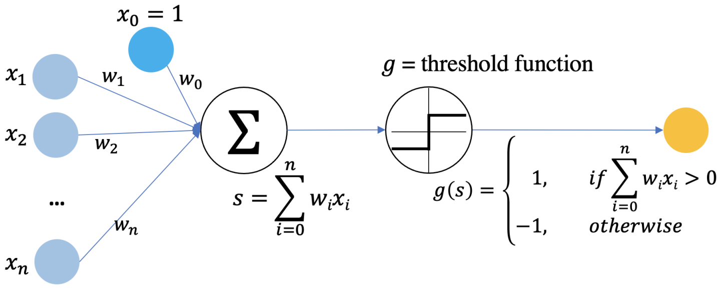
- Bias shifts the activation function (to left or right).
- Intuition: $y = ax + b$ equivalent to $y = ax + 1 \times b$
- Bias is added to the product of inputs and weights.
- Example2: a NN with 2 input nodes ($x_1 = 1, x_2 = 2$), one node in the hidden layer with $ReLU(x) = max(0, x)$.
  Random weights $w_1 = -0.5; w_2 = 0.2$.
  Then $ReLU(1 \times -0.5 + 2 \times 0.2) = ReLU(-0.1) = 0$
  What if we want to shift our threshold and fire at -1 instead?
  Add bias 1 $\Rightarrow ReLU(1 \times -0.5 + 2 \times 0.2 + 1) = 0.9$

Bias



- Bias is added to the product of inputs and weights.
- The bias gradients are not affected by the feedforward signal, only by the error.
- Common practice to initialise biases with 0. For ReLU, biases are initialised to a small value (e.g. 0.01) so that they fire in the beginning.

Transfer/activation/threshold function



$x_0 = 1$
x_1
x_2
...
x_n
w_0
w_1
w_2
w_n
Σ
$s = \sum_{i=0}^{n} w_i x_i$
g = threshold function
$g(s) = \begin{cases} 1, & if \sum_{i=0}^{n} w_i x_i > 0 \\ -1, & otherwise \end{cases}$

- Examples of activation functions:
  - Sigmoid
  - Hyperbolic Tangent
  - Rectified Linear Unit

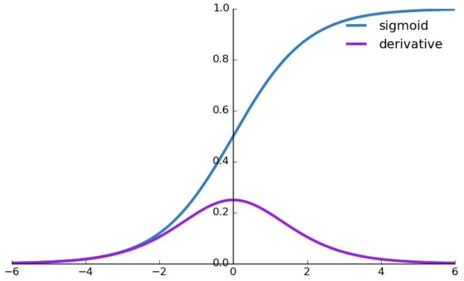
Activation functions

- Q: Why should the activation function be differentiable?
- Q: On slide 3, the slides say that the activation function is non-linear.
- Q: How to choose the activation function.

- A: The activation function should be non-linear.
  Why? To learn non-linear relationships in the data.
- The activation function should be differentiable.
  Why? To backpropagate the error when training to optimise the weights.

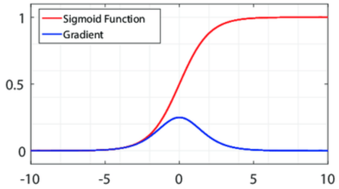
Vanishing gradient

- Q: Why are ReLU and Softplus better than Sigmoid?

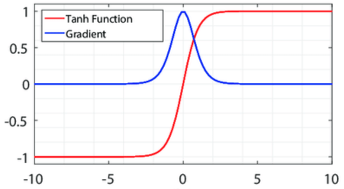


- Using the chain rule, the derivatives of each layer are multiplied from the final layer to the initial layer to compute the derivatives of the initial layers.
- As the number of layers increases, the number of products required to update the weights increases.
- If the gradients become very small, the updates to the weights will be negligible.

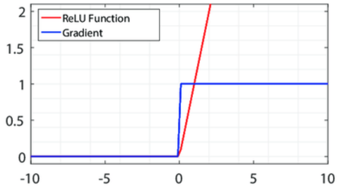
Plots of various activation functions



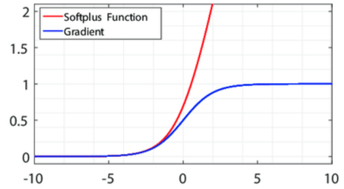
(a)



(b)



(c)



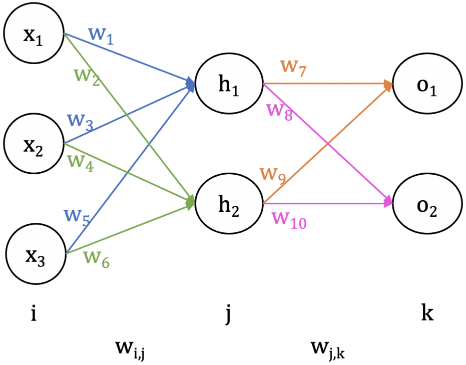
(d)

- Sigmoid is smooth (has smooth threshold) and differentiable at all points.
- Softplus is a smooth approximation to the ReLU function.

Backpropagation

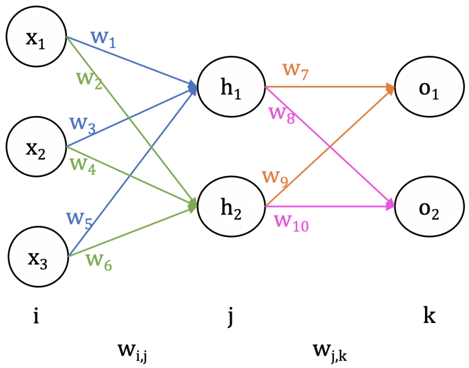
- Q: Are we expected to fully understand all the concepts and rules of calculating partial derivatives for the sake of preparing for the exam?
- Q: How deep should we go in the preparation for the exam with respect to knowing how to derivate?
- Q: Do we have to compute the partial derivative each time we need to update the weights?
- Q: Backpropagation example.

Backpropagation example



- We will initialise weights as follows (values chosen to keep the calculations simple):
  $w_1 = 0.1; \ldots; w_6 = 0.6; w_7 = 0.7; \ldots; w_9 = 0.9; w_{10} = 1$
- First instance in the training set:
  $x_1 = 1; x_2 = 2; x_3 = 3$ and $y_1 = 1; y_2 = 0$

Backpropagation example

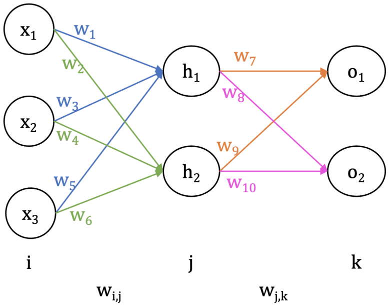


- We will use sigmoid as it has a simple to use derivative.

$$g(s) = \frac{1}{1+e^{-s}} \qquad \frac{\partial g(s)}{\partial s} = g(s)(1-g(s))$$

- First instance in the training set:
  $x_1 = 1; x_2 = 2; x_3 = 3$ and $y_1 = 1; y_2 = 0$

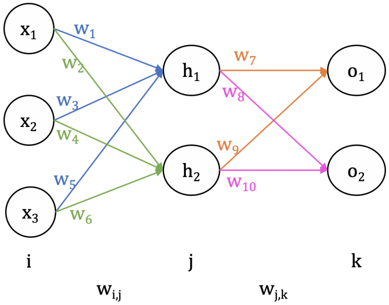
Backpropagation example



$$h_j = g\left(\sum_i x_i w_{i,j}\right) = g(s_j)$$

$$o_k = g\left(\sum_j h_j w_{j,k}\right) = g(s_k)$$

Backpropagation example



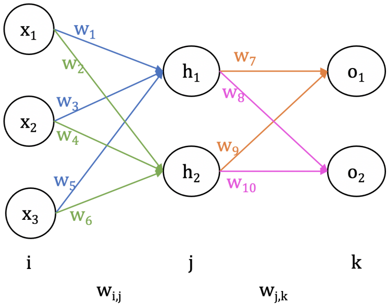
$$h_j = g\left(\sum_i x_i w_{i,j}\right) = g(s_j)$$

Our input and target: $x_1 = 1; x_2 = 2; x_3 = 3$ and $y_1 = 1; y_2 = 0$

$h_1 = g(s_{h_1}) = g(x_1 w_1 + x_2 w_3 + x_3 w_5) = g(2.2) = 0.90$

$h_2 = g(s_{h_2}) = g(x_1 w_2 + x_2 w_4 + x_3 w_6) = g(2.8) = 0.94$

Backpropagation example



$$o_k = g\left(\sum_j h_j w_{j,k}\right) = g(s_k)$$

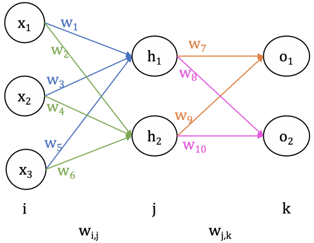
$$o_1 = g(s_{o_1}) = g(h_1 w_7 + h_2 w_9) = g(1.47) = 0.81$$

$$o_2 = g(s_{o_2}) = g(h_1 w_8 + h_2 w_{10}) = g(1.66) = 0.84$$

Backpropagation example

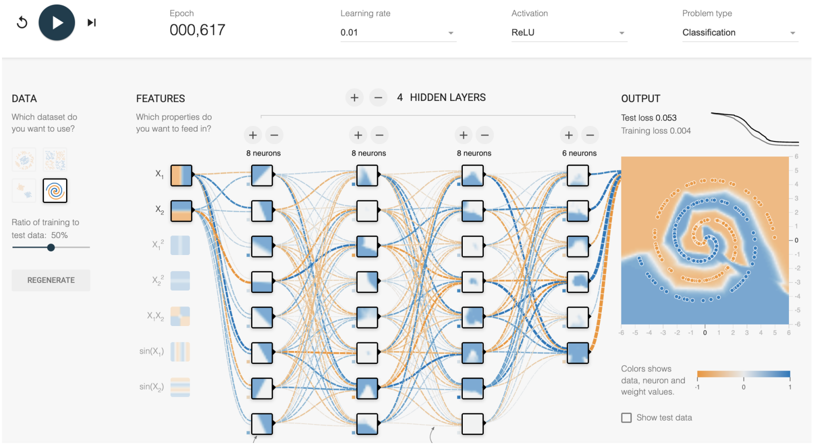
- The rule for updating weights is: $w_i \leftarrow w_i - \alpha \dfrac{\partial E}{\partial w_i}$
- We obtained: $\dfrac{\partial E}{\partial w_{j,k}} = -\delta_k h_j$
- Thus: $w_{j,k} \leftarrow w_{j,k} + \alpha \delta_k h_j$
- From the slides: $\delta_k = (y_k - o_k) g(s_k)(1 - g(s_k))$
- So, on our example:
  $w_7 \leftarrow w_7 + \alpha(y_1 - o_1) \times 0.81 \times (1 - 0.81) h_1$
  Replacing the values gives you the new w_7.
  Similar process for w_8, w_9, w_{10} in our example.

Backpropagation example



- The rule for updating weights is: $w_i \leftarrow w_i - \alpha \dfrac{\partial E}{\partial w_i}$
- We obtained: $\dfrac{\partial E}{\partial w_{i,j}} == -x_i \underbrace{g(s_j)(1 - g(s_j)) \sum_k \delta_k w_{j,k}}_{\delta_j} = -\delta_j x_i$
- Thus: $w_{i,j} \leftarrow w_{i,j} + \alpha \delta_j x_i$
- Replace with the appropriate values to calculate new $w_1, \ldots, w_6$ (e.g., for w_1, consider h_1 and weights w_7 and w_8).

Neural networks


Epoch 000,617

Learning rate 0.01

Activation ReLU

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

REGENERATE

FEATURES

Which properties do you want to feed in?

4 HIDDEN LAYERS

8 neurons 8 neurons 8 neurons 6 neurons

OUTPUT

Test loss 0.053

Training loss 0.004

Colors shows data, neuron and weight values.

Show test data

- Q: What’s happening in the hidden layers?
- A: Example with Tensorflow on the spiral dataset.

Deep neural networks

- Q: How do we know how many layers we need and what dimensionality does the data need to have on each layer?
- Q: How do I gather enough experience to understand how to build the architecture of the NN for my specific problem?
- Q: What do we have to demonstrate in order to come up with our own function?

- A: There is no exact answer.
- It depends on the number of inputs/outputs, amount of training data, complexity of the function to be learnt.
- You can experiment with different number of hidden layers and neurons per layer.
- How to start: Have a look at state-of-the-art on the dataset you are using and the task you are addressing, see existing implementations (papers also release the code).

Deep neural networks

- Q: How do you make existing models explainable?
- Q: What is symbolic AI?