Backpropagation

Glenn Bruns
CSUMB

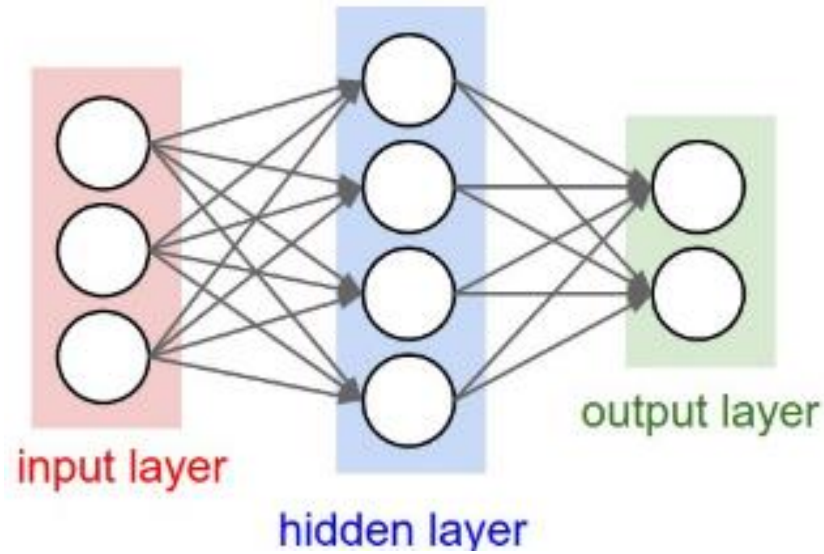
Much material in this deck from Géron, Hands-on Machine Learning with Scikit-Learn and TensorFlow

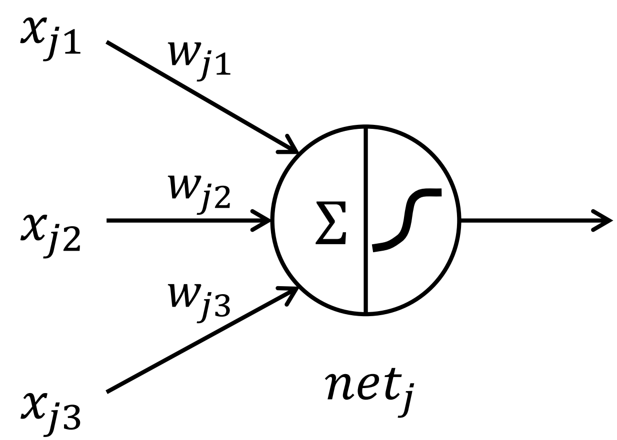
Learning outcomes

After this lecture you should be able to:

- Manually perform backpropagation on a simple network

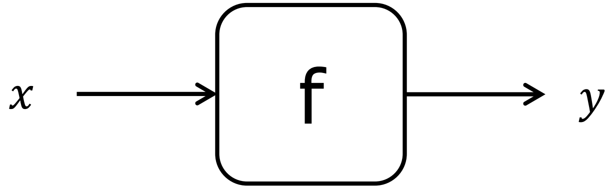
Training neural nets



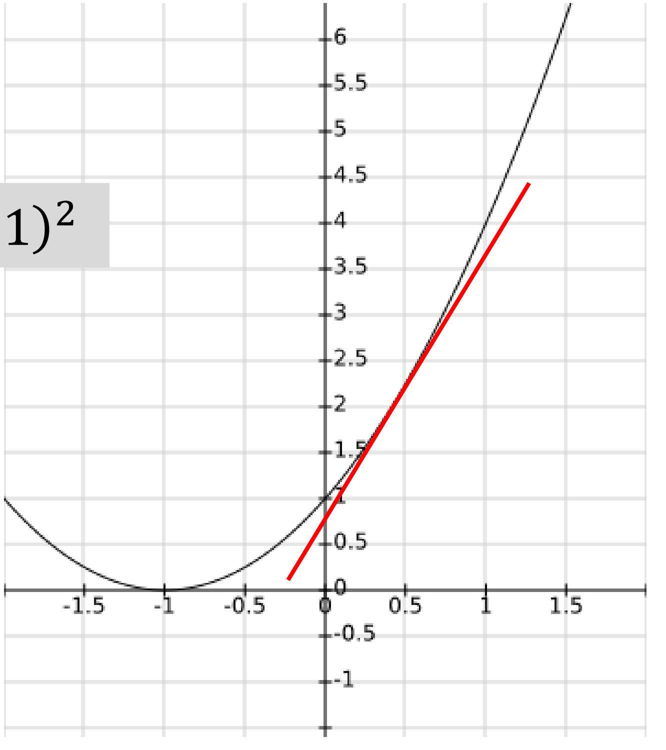


- The parameters of a neural net are its weights.
- In training, training data is fed to the net, and the weights are adjusted.
- Question: how does this work?

A simple example



$y = (x + 1)^2$

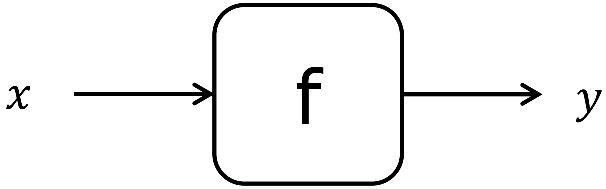


Suppose input x = 0.5, and we want to adjust x to make output y smaller.

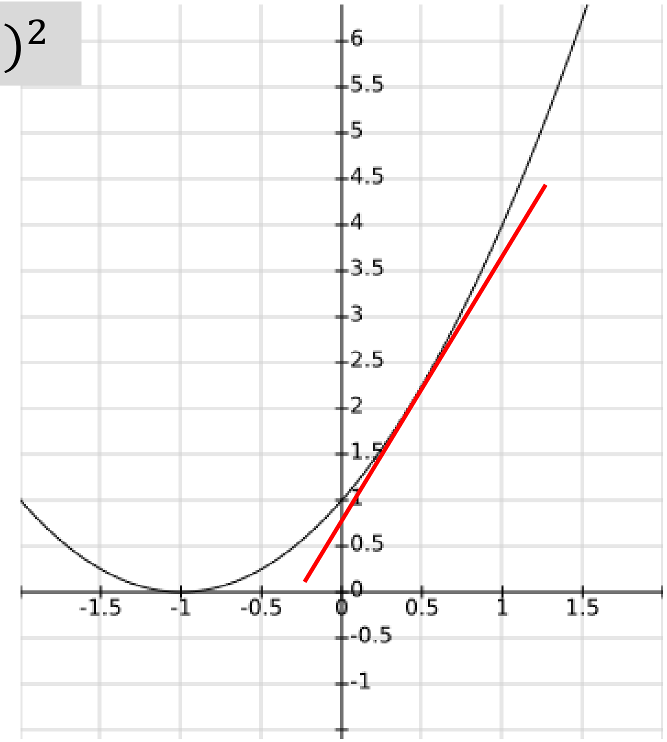
Look at the slope of the function at x = 0.5

The slope of the function at x is positive, so make x a little smaller

Details on adjusting x



$y = (x+1)^2$



1. What is the slope at x = 0.5 ?

$$\frac{dy}{dx} = 2(x+1)$$

this is a <u>function</u>; the slope depends on x

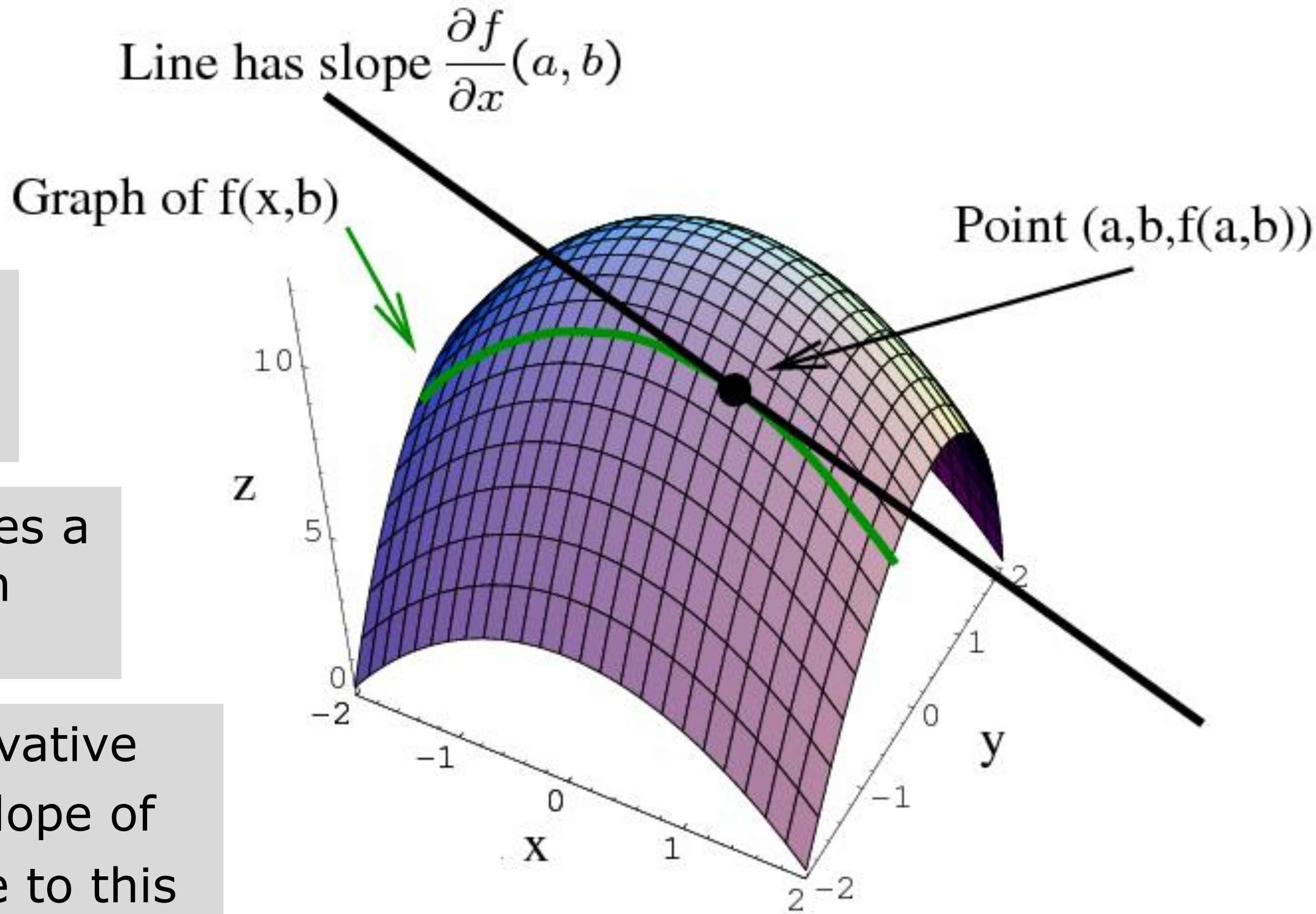
The slope at x = 0.5 is
2(0.5 + 1) = 3

2. How exactly to adjust x?

$$x_1 = x_0 - \eta \frac{dy}{dx}(x_0)$$

if $x_0 = 0.5$, and $\eta = 0.01$, then x_1 is ?

Recall: partial derivatives



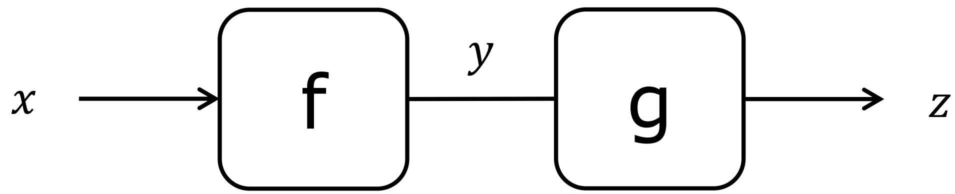
The graph of $f(x, y)$ is a surface.

Fixing $y = b$ gives a curve (shown in green

The partial derivative $\frac{\delta f}{\delta x}(a, b)$ is the slope of the tangent line to this curve at the point where $x = a$.

source: https://mathinsight.org/partial_derivative_limit_definition

An example with two nodes



$z = y^2$

$y = x + x\text{^}3$

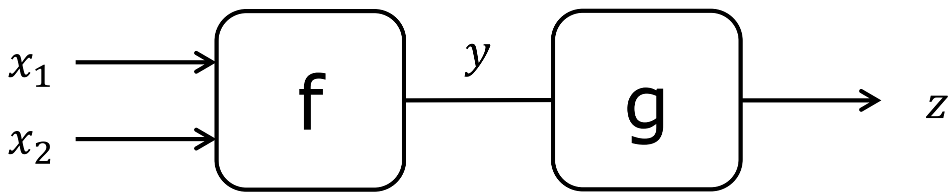
Approach 1: combine f and g to get $z = (x + x^3)^2$ and then do the same thing as the last example.

Approach 2: get $\frac{dz}{dx}$ by using the chain rule: $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

Example: Let the input x be 1 (so y is 2 and z is 4). How to adjust x to make z smaller?

Work out that $\frac{dz}{dy}$ is $2y$, and $\frac{dy}{dx}$ is $1 + 3x^2$. Then the chain rule says that the slope of the f,g combined, at x=1, is:
$\frac{dz}{dy}(2) * \frac{dy}{dx}(1) = 4 * 4 = 16.$ For new x, use $x - \eta \frac{dz}{dx}(x)$

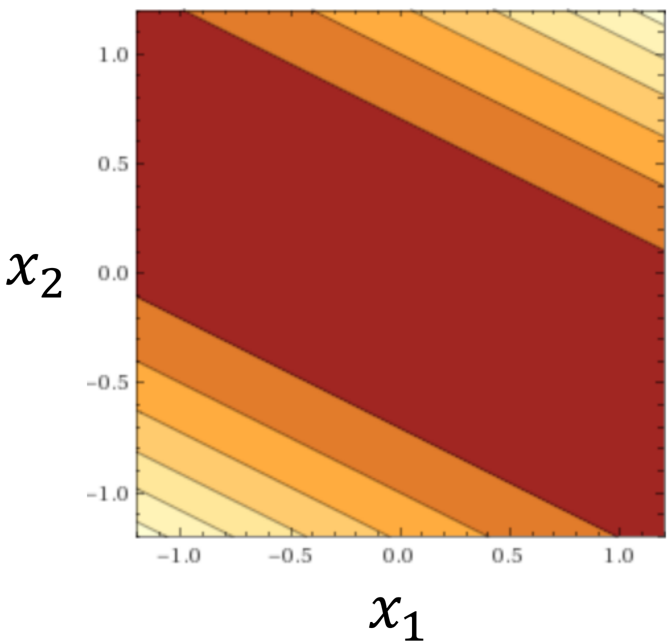
An example with a multi-input node

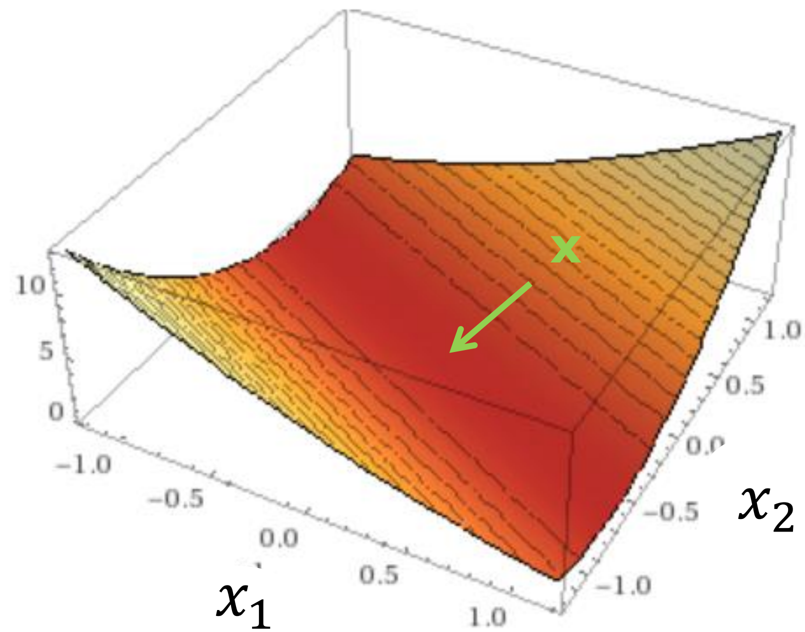


$z = y^2$

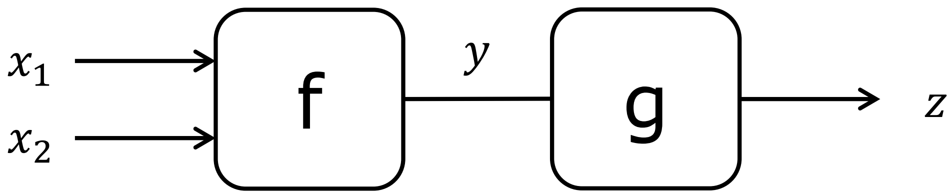
$y = x_1 + 2\,x_2$

Now we have to think about how x_1 affects z and how x_2 affects z





An example with multi-input node



$z = y^2$

$y = x_1 + 2\,x_2$

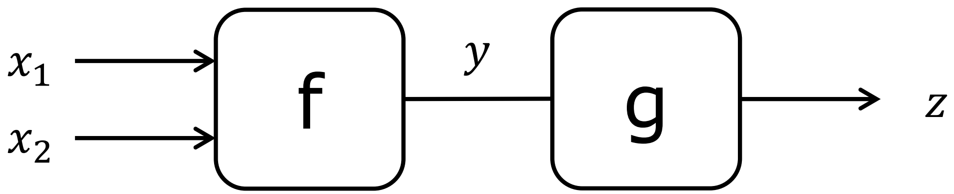
What is $\frac{\partial z}{\partial x_1}$? What is $\frac{\partial z}{\partial x_2}$?

Use the multi-variable chain rule: $\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_1}$ (similarly for x_2)

Example: Let $x_1 = 1$ and $x_2 = 3$ (so $y = 7$). How to adjust x_1 to lower z?

1. $\frac{\partial z}{\partial y}$ is $2y$, and $\frac{dy}{dx_1}$ is 1
2. Using the chain rule: $\frac{\partial z}{\partial x_1}$ at (1,3) is $\frac{dz}{dy}(7) * \frac{dy}{dx_1}(1{,}3) = 14 * 1 = 14$
3. For new value of x_1, use $x_1 - \eta\,\frac{dz}{dx_1}(1{,}3)$

Exercise: calculate new x_2



$z = y^2$

$y = x_1 + 2\,x_2$

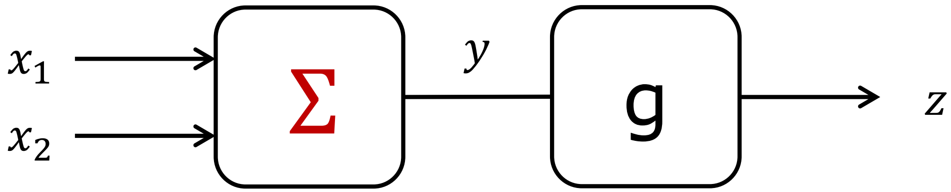
Use the multi-variable chain rule: $\frac{\partial z}{\partial x_2} = \frac{\partial z}{\partial y}\,\frac{\partial y}{\partial x_2}$ (similarly for x_2)

Let $x_1 = 1$ and $x_2 = 3$ (so $y = 7$)

1. $\frac{\partial z}{\partial y}$ is $2y$, and $\frac{\delta y}{\delta x_2}$ is 2

2. Calculate new value of x_2, use $x_2 - \eta\,\frac{\delta z}{\delta x_2}(1{,}3)$ (let η be 0.01)

A weighted sum node



$z = y^2$

$y = \mathrm{w}_1\, x_1 + w_2\, x_2$

Node Σ outputs the weighted sum of its inputs. The coefficient values are $w_1 = 0.5$ and $w_2 = 2.0$.

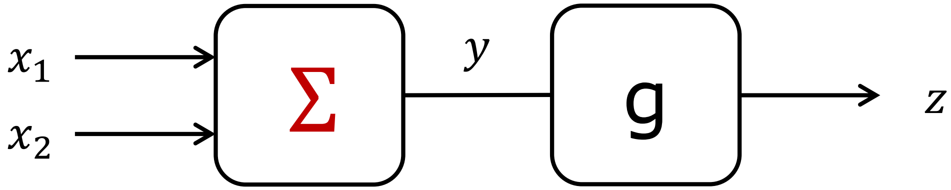
How to modify the coefficients of Σ to make z smaller? (Now treat x_1 and x_2 as constants.)

Use the multi-variable chain rule: $\dfrac{\partial z}{\partial w_1} = \dfrac{\partial z}{\partial y}\dfrac{\partial y}{\partial w_1}$ (similarly for w_2)

Example: Let $x_1 = 1$ and $x_2 = 3$ (so $y = 0.5 + 6 = 6.5$).

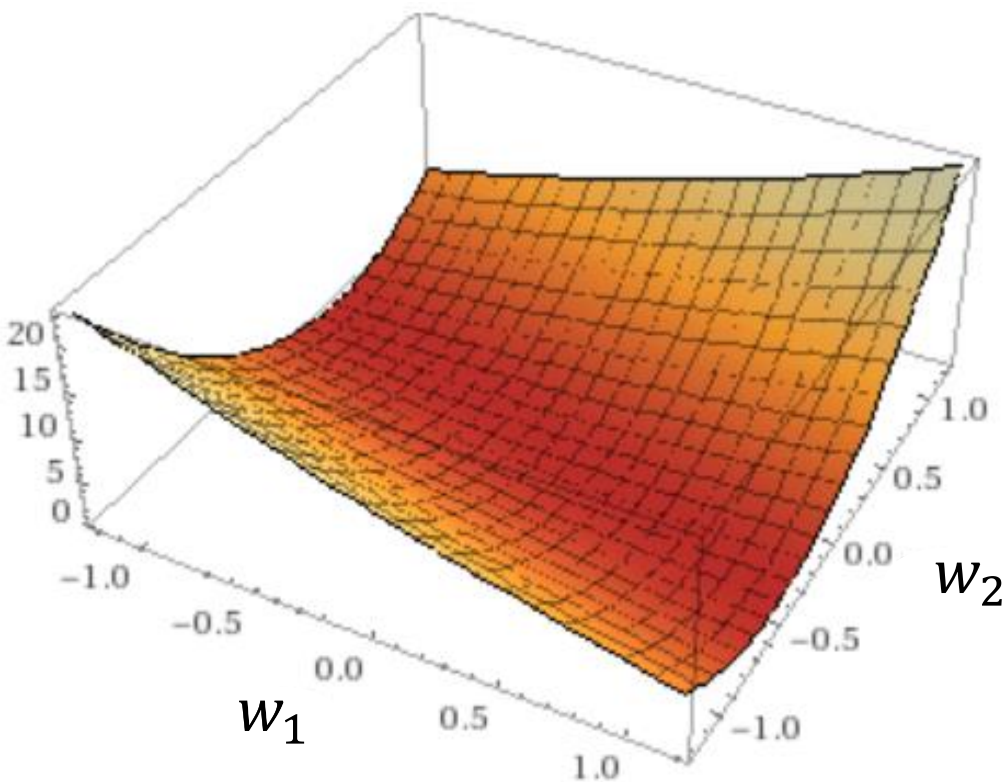
1. $\frac{\partial z}{\partial y}$ is $2y$, and $\frac{\delta y}{\delta w_1}$ is x_1 (note: $\frac{\delta y}{\delta w_1}$ does not depend on w_1 !)
2. By chain rule: $\frac{\partial z}{\partial w_1}$ at w_1, w_2 is $\frac{dz}{dy}(6.5) * \frac{dy}{dw_1}(0.5{,}2.0) = 13 * 1 = 13$
3. For new value of w_1, use $w_1 - \eta\, \frac{\delta z}{\delta w_1}(0.5, 2.0)$

A weighted sum node



$z = y^2$

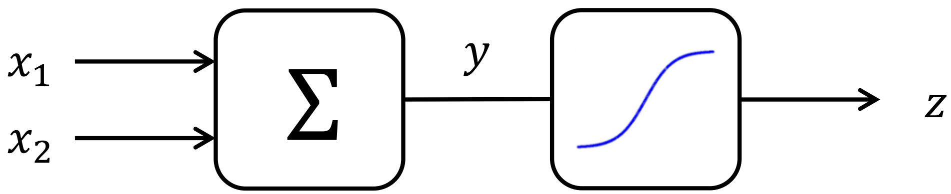
$y = \mathrm{w}_1\, x_1 + w_2\, x_2$



A plot of the combined functions.

x_1 and x_2 are treated as constants ($x_1 = 1$ and $x_2 = 3$).

A logistic activation function node



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

$$y = \mathrm{w}_1\, x_1 + w_2\, x_2$$

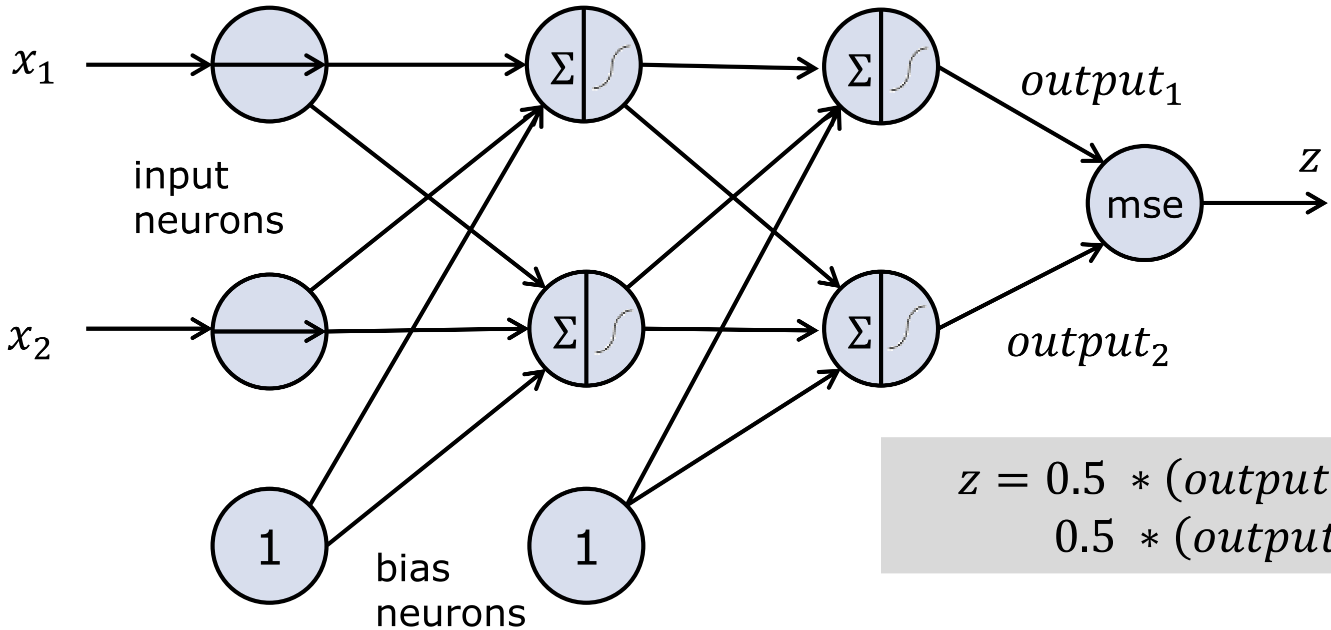
Node f outputs the weighted sum of its inputs. The coefficients of f are $w_1 = 0.5$ and $w_2 = 2.0$.

Use the multi-variable chain rule: $\frac{\partial z}{\partial w_1} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial w_1}$ (similarly for w_2)

Example: Let $x_1 = 1$ and $x_2 = 3$ (so $y = 0.5 + 6 = 6.5$).

1. $\frac{\partial z}{\partial y}$ is $g(y)(1 - g(y))$, and $\frac{dy}{dw_1}$ is x_1
2. By chain rule: $\frac{\partial z}{\partial x_1}$ at w_1, w_2 is $\frac{dz}{dy}(6.5) * \frac{dy}{dw_1}(0.5{,}2.0) = 0.0015 * 1 = 0.0015$
3. For new value of w_1, use $w_1 - \eta \frac{dz}{dw_1}(0.5{,}2.0)$

A more realistic example



$$z = 0.5 * (output_1 - y_1)^2 + 0.5 * (output_2 - y_2)^2$$

Here there are 8 weights, 2 for each of the nodes in the hidden layers. We want to minimize the error z.

Backpropagation takes some x_1, x_2 as input, and outputs $(\frac{\partial z}{\partial w_1}, \frac{\partial z}{\partial w_2}, \frac{\partial z}{\partial w_3}, \frac{\partial z}{\partial w_4}, \frac{\partial z}{\partial w_4}, \frac{\partial z}{\partial w_6}, \frac{\partial z}{\partial w_7}, \frac{\partial z}{\partial w_8})$

Notes from Goodfellow et al

- Multilayer perceptrons (MLPs) are also called deep feedforward networks
- back-propagation is not gradient descent – it refers only to the method for computing the gradient
- back-propagation is a very general technique:
  - is not limited to the gradient of a cost function with respect to its parameters
  - not limited to neural networks

source: Deep Learning, by Goodfellow, Bengio, and Courville

Summary

- During training of a neural net you repeatedly tweak node weights to reduce the value of a cost function.
- With backprop you compute the partial derivatives needed in the optimization process.
- The examples in these slides show the core ideas, but modern neural nets use much more advanced algorithms.

Bonus content

www.emergentmind.com/neural-network