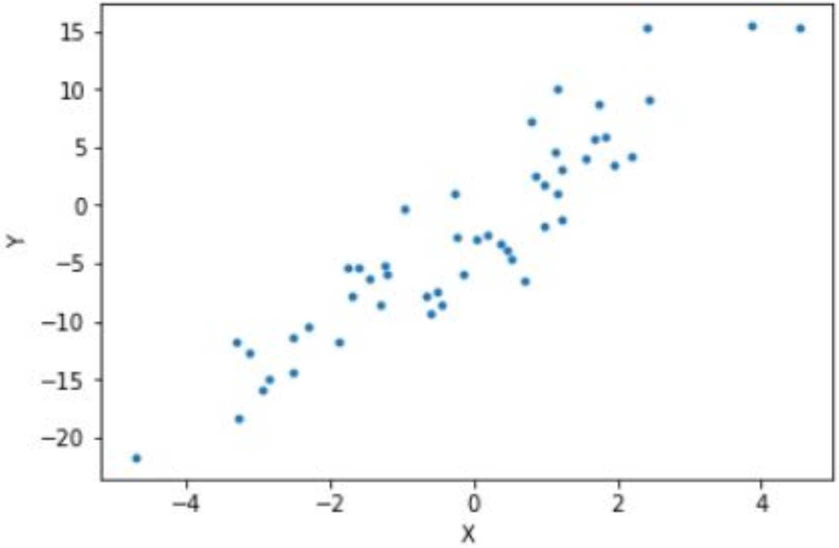
Backprop

Outline

- Simple example (OLS regression)
- Chain rule
- Backprop
- Deriving gradients for a neural network

Simple Regression

$$\hat{Y} = \beta_0 + \beta_1 X$$



Yes, we know how to estimate the coefficients of this model analytically. But we'll use this as a simple example of how to minimize a Loss function through gradient descent.

Squared Error Loss

$$L = \sum_i (y_i - \hat{y}_i)^2$$

$$= \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2$$

$$\theta = \{\beta_0, \beta_1\}$$

$$\min_{\beta_0, \beta_1} L : \nabla_\theta L = 0$$

Minimizing Loss

$$
\begin{aligned}
\frac{\partial L}{\partial \beta_0} &= \frac{\partial}{\partial \beta_0} \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= \sum_i \frac{\partial}{\partial \beta_0} (y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= 2 \sum_i (y_i - (\beta_0 + \beta_1 x_i)) \frac{\partial}{\partial \beta_0} (y_i - (\beta_0 + \beta_1 x_i)) \\
&= 2 \sum_i (y_i - (\beta_0 + \beta_1 x_i))(-1) \\
&= -2 \sum_i (y_i - (\beta_0 + \beta_1 x_i)) \\
&= -2 \sum_i e_i
\end{aligned}
$$

Minimizing Loss

$$
\begin{aligned}
\frac{\partial L}{\partial \beta_1} &= \frac{\partial}{\partial \beta_1} \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= \sum_i \frac{\partial}{\partial \beta_1} (y_i - (\beta_0 + \beta_1 x_i))^2 \\
&= 2 \sum_i (y_i - (\beta_0 + \beta_1 x_i)) \frac{\partial}{\partial \beta_1} (y_i - (\beta_0 + \beta_1 x_i)) \\
&= 2 \sum_i (y_i - (\beta_0 + \beta_1 x_i))(-x_i) \\
&= -2 \sum_i x_i (y_i - (\beta_0 + \beta_1 x_i)) \\
&= -2 \sum_i e_i x_i
\end{aligned}
$$

Minimizing Loss

$$L = \sum_i (y_i - (\beta_0 + \beta_1 x_i))^2$$

$$\nabla_\theta L = \{-2 \sum_i e_i, -2 \sum_i e_i x_i\}$$

Exercise : Use the accompanying notebook to estimate regression coefficients using gradient descent.

Minimizing Loss

- For our simple regression model, calculating the gradient was simple enough algebra. This is because the model is so simple.

- For neural network models, the model outputs tend to be a sum of several composed functions. Computing the gradient is still straightforward, but requires a little more thought and book-keeping.

Chain Rule

- Computing derivatives of composed functions

$$x \xrightarrow{f(x)} y \xrightarrow{g(y)} z$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

Example:

$$y = x^2$$
$$z = sin(y)$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$
$$= cos(y)2x$$

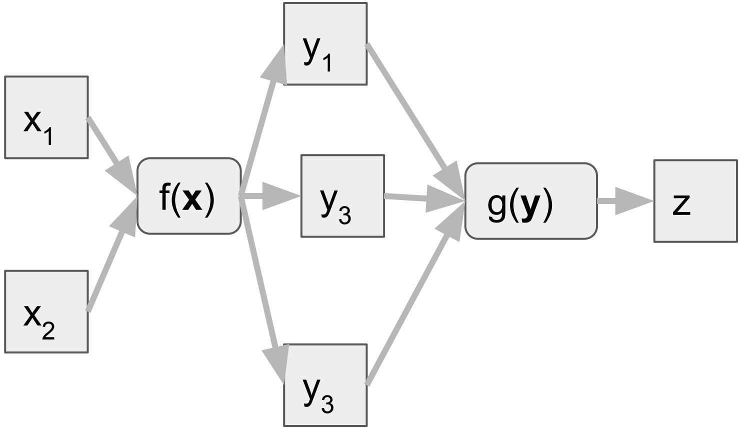
Chain Rule

- Generalize to vector-valued functions

$$\mathbf{x} \xrightarrow{f(\mathbf{x})} \mathbf{y} \xrightarrow{g(\mathbf{y})} \mathbf{z}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}.$$

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z,$$



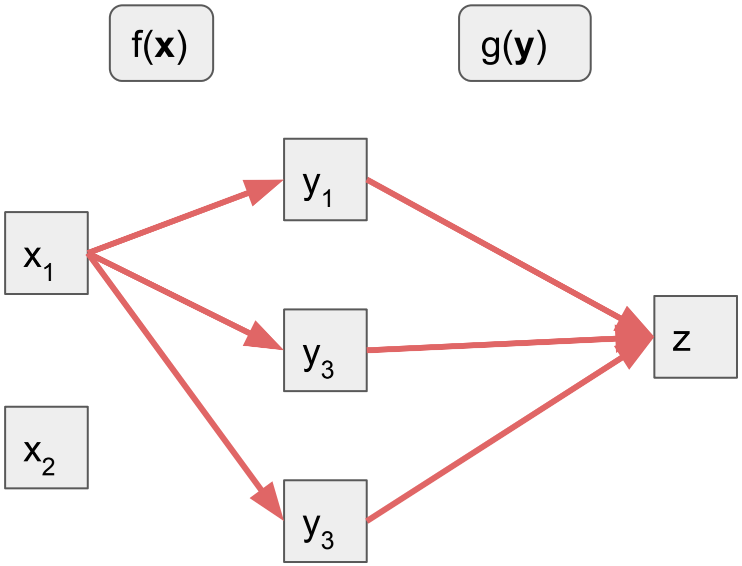
Chain Rule

- Generalize to vector-valued functions

$$\mathbf{x} \xrightarrow{\quad f(\mathbf{x})\quad} \mathbf{y} \xrightarrow{\quad g(\mathbf{y})\quad} \mathbf{z}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}.$$

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z,$$



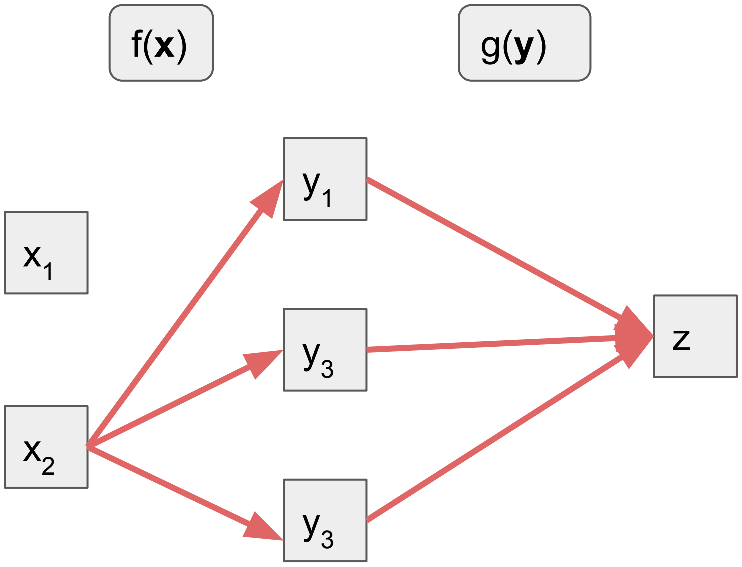
Chain Rule

- Generalize to vector-valued functions

$$\mathbf{x} \xrightarrow{f(\mathbf{x})} \mathbf{y} \xrightarrow{g(\mathbf{y})} \mathbf{z}$$

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}.$$

$$\nabla_{\boldsymbol{x}} z = \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)^{\top} \nabla_{\boldsymbol{y}} z,$$



Backpropagation

After the forward computation, compute the gradient on the output layer:

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

for $k = l, l-1, \ldots, 1$ **do**

Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$

Compute gradients on weights and biases (including the regularization term, where needed):

$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$

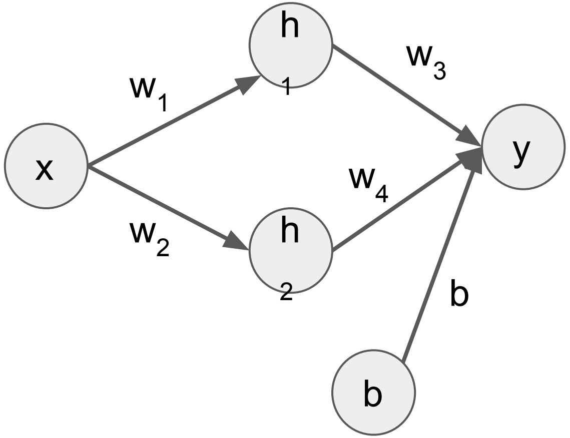
$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$

Propagate the gradients w.r.t. the next lower-level hidden layer's activations:

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top}\, \boldsymbol{g}$

end for

Example - Regression Network



$$L = \frac{1}{2}(y_i - \hat{y}(x_i))^2$$

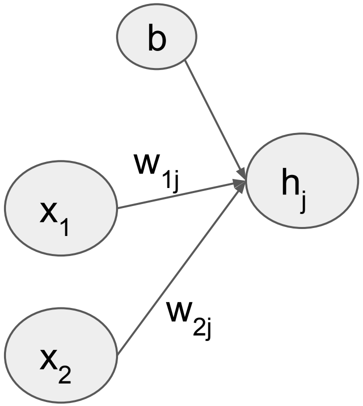
$$\theta = \{w_1, w_2, w_3, w_4, b\}$$

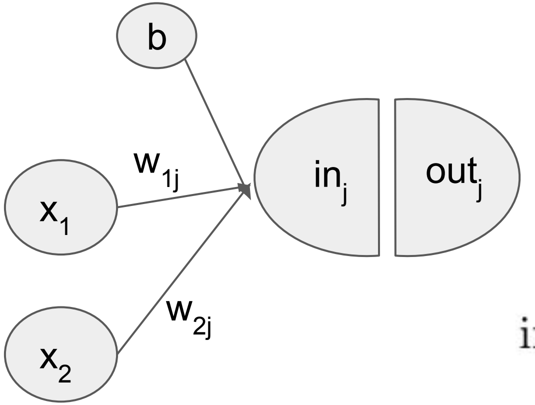
$$\nabla_\theta = \{\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4}, \frac{\partial L}{\partial b}\}$$

We'll use sigmoid activation

Aside - Notational Convenience

When deriving gradients, we will use a more explicit representation of hidden units, one that allows us to represent (and name) every transformation that occurs.



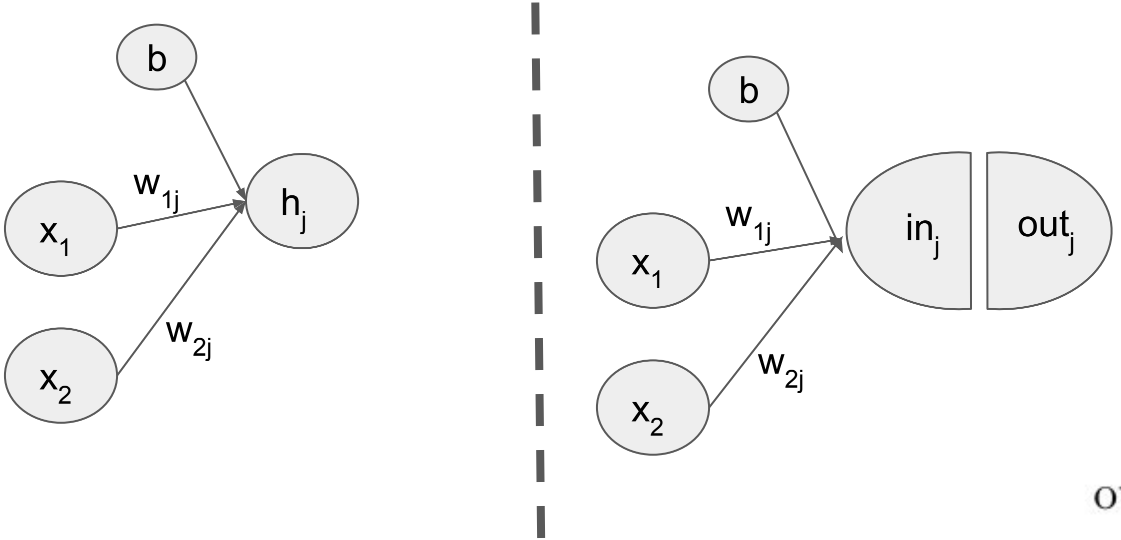


$$\mathrm{in}_j = x_1 w_{1j} + x_2 w_{2j} + b$$

$$\mathrm{out}_j = f(\mathrm{in}_j) = \frac{1}{1 + \exp(-\mathrm{in}_j)}$$

Aside - Notational Convenience

When deriving gradients, we will use a more explicit representation of hidden units, one that allows us to represent (and name) every transformation that occurs.

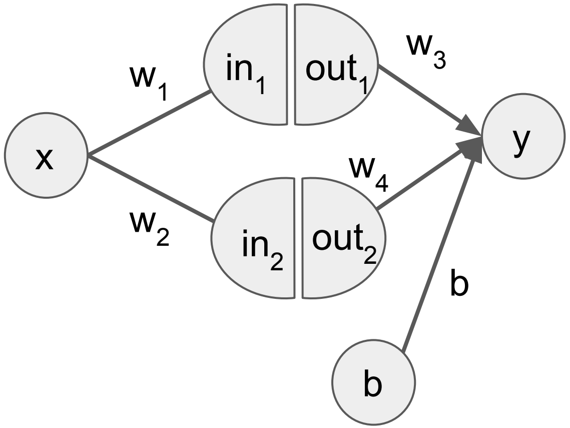


Note how out_j is a composite function with respect to x or any w

This is why the chain rule comes in

$$\mathrm{out}_j = \frac{1}{1 + \mathrm{e}^{-(x_1 w_{1j} + x_2 w_{2j} + b)}}$$

Example - Regression Network



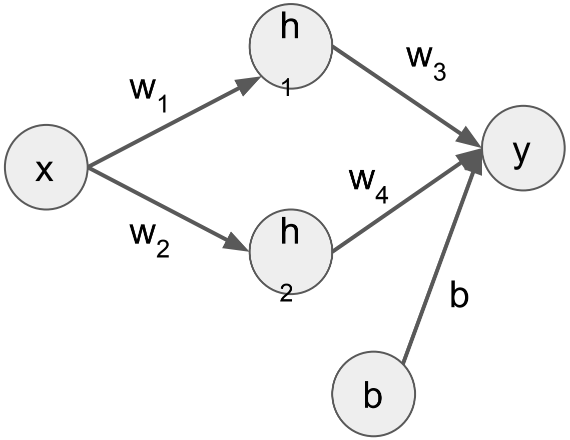
This notation makes it clear how we will compute derivatives of composite functions. Some useful properties:

(1) $$\frac{d\mathrm{out}_j}{d\mathrm{in}_j} = \frac{d}{dx}f(x) = \frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$$

(2) $$\frac{d\mathrm{in}_j}{dw_k} = \frac{d}{dw_k}[b + w_1x_1 + w_2x_2] = x_k$$

(3) $$\frac{d\mathrm{out}_j}{dw_k} = \frac{d\mathrm{out}_j}{d\mathrm{in}_j}\frac{d\mathrm{in}_j}{dw_k} = \sigma(\mathrm{in}_j)(1-\sigma(\mathrm{in}_j))x_k$$

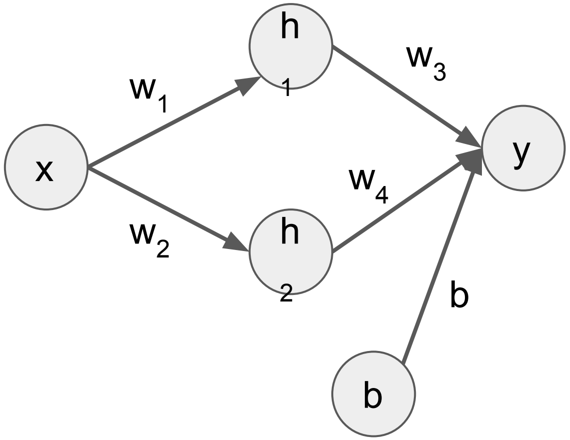
Example - Regression Network



With these preliminaries, let's restate our goals

- Construct a loss function
- Derive the gradient vector for that loss function
- Use gradient descent to optimize that loss function

Example - Regression Network



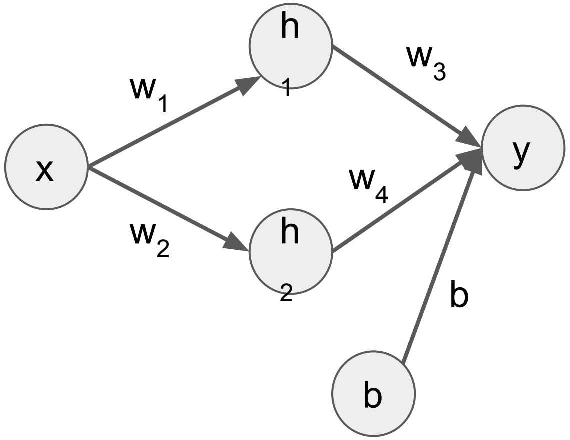
- Construct a loss function

$$L = \frac{1}{2}(y_i - \hat{y}(x_i))^2$$

$$\theta = \{w_1, w_2, w_3, w_4, b\}$$

$$\nabla_\theta = \{\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4}, \frac{\partial L}{\partial b}\}$$

Example - Regression Network

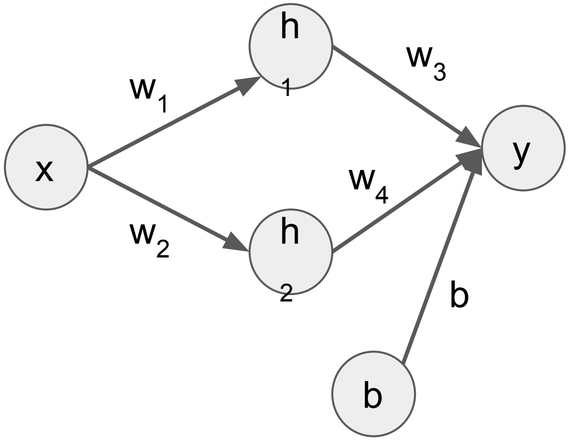


- Derive gradients

$$\begin{aligned}
\frac{\partial L}{dw_3} &= \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_3} \\
&= \frac{\partial L}{\partial \hat{y}} \frac{\partial}{\partial w_3} (w_3 \text{out}_1 + w_4 \text{out}_2 + b) \\
&= \frac{\partial L}{\partial \hat{y}} \text{out}_1 \\
&= \text{out}_1 \frac{\partial}{\partial \hat{y}} (\frac{1}{2}(y_i - \hat{y}(x_i))^2) \\
&= \text{out}_1 (y_i - \hat{y}(x_i))(0 - 1) \\
&= -\text{out}_1 (y_i - \hat{y}(x_i))
\end{aligned}$$

$$\frac{\partial L}{\partial w_3} = -\text{out}_1 e_i$$

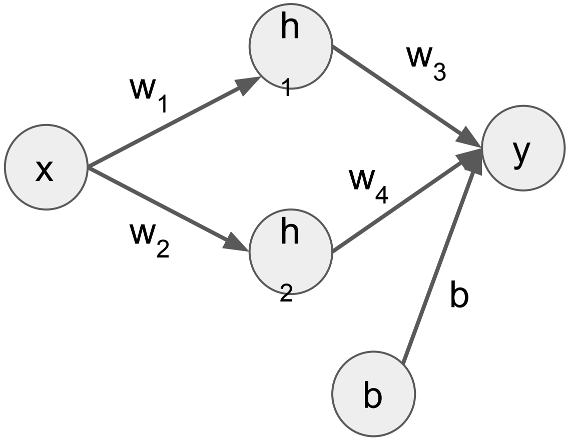
Example - Regression Network



- Derive gradients

$$\frac{\partial L}{\partial w_4} = -\mathrm{out}_2 e_i$$

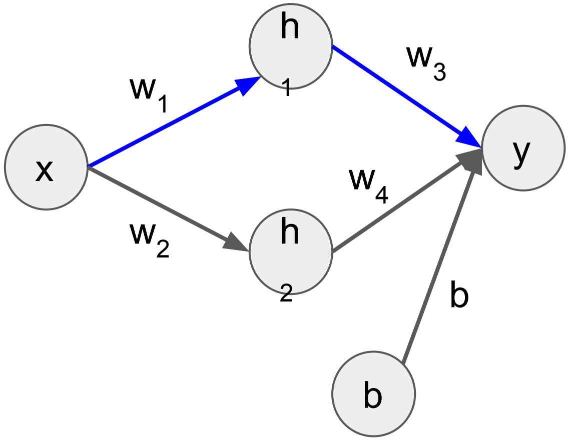
Example - Regression Network



- Derive gradients

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b}$$
$$= -(y_i - \hat{y}(x_i)) \frac{\partial}{\partial b} \hat{y}$$
$$= -e_i \frac{\partial}{\partial b} (w_3 \text{out}_1 + w_4 \text{out}_2 + b)$$
$$= -e_i$$

Example - Regression Network



- Derive gradients

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathrm{out}_1} \frac{d\mathrm{out}_1}{d\mathrm{in}_1} \frac{\partial \mathrm{in}_1}{\partial w_1}$$

$$\frac{\partial L}{\partial \hat{y}} = -(y_i - \hat{y}(x_i))$$

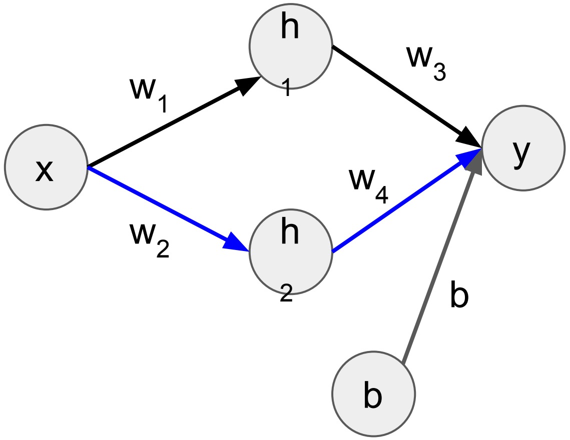
$$\frac{\partial \hat{y}}{\partial \mathrm{out}_1} = w_3$$

$$\frac{d\mathrm{out}_1}{d\mathrm{in}_1} = \sigma(\mathrm{in}_1)(1 - \sigma(\mathrm{in}_1))$$

$$\frac{\partial \mathrm{in}_1}{\partial w_1} = \frac{d}{dw_1}(xw_1) = x$$

$$\frac{\partial L}{\partial w_1} = -x_i e_i w_3 \sigma(\mathrm{in}_1)(1 - \sigma(\mathrm{in}_1))$$

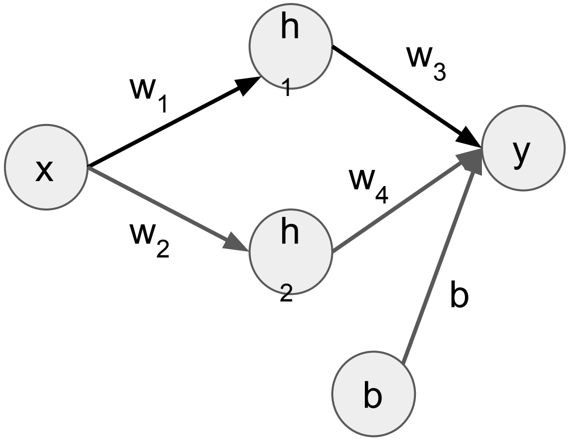
Example - Regression Network



- Derive gradients

$$\frac{\partial L}{\partial w_2} = -x_i e_i w_4 \sigma(\mathrm{in}_2)(1 - \sigma(\mathrm{in}_2))$$

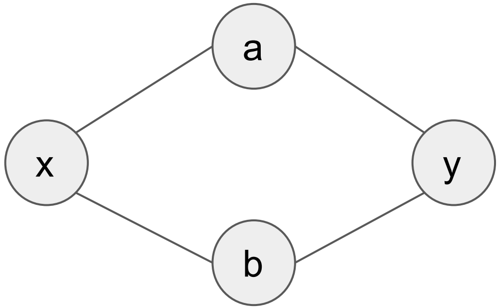
Example - Regression Network



- Derive gradient

$$\nabla_\theta = \begin{bmatrix} -x_i e_i w_3 \sigma(\mathrm{in}_1)(1-\sigma(\mathrm{in}_1)) \\ -x_i e_i w_4 \sigma(\mathrm{in}_2)(1-\sigma(\mathrm{in}_2)) \\ -\mathrm{out}_1 e_i \\ -\mathrm{out}_2 e_i \\ -e_i \end{bmatrix}$$

Chain Rule - Sum over paths

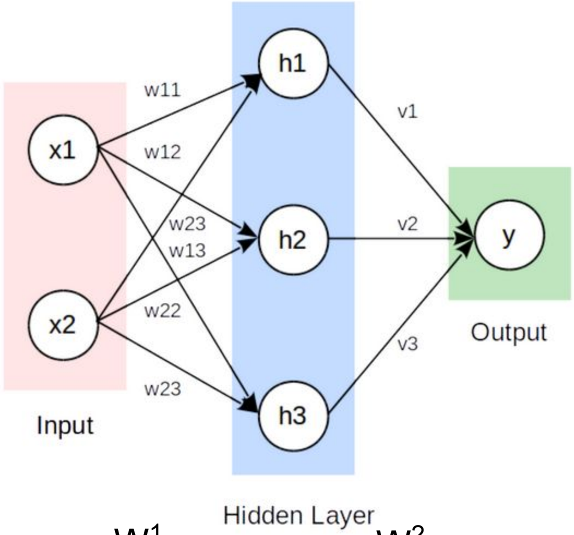


$$\frac{dy}{dx} = \frac{dy}{da}\frac{da}{dx} + \frac{dy}{db}\frac{db}{dx}$$

- To compute the total contribution of one variable to another, we must sum over all paths in the computation graph that connect the two
- We didn't encounter this in our previous examples, but is important to keep in mind with deeper networks

Generic Formulation for MLPs

- Each layer has a weight matrix **W** and a bias vector **b**.
- The layer is typically associated with a single type of activation function.
- Borrowing from Nielsen, we'll use a notational convention
  - The *error* due to unit j from layer l
  - δ^l_j



W^1 W^2

b^1 b^2

Generic Formulation for MLPs

- Starting from the output layer
- The error due to unit j in the output layer

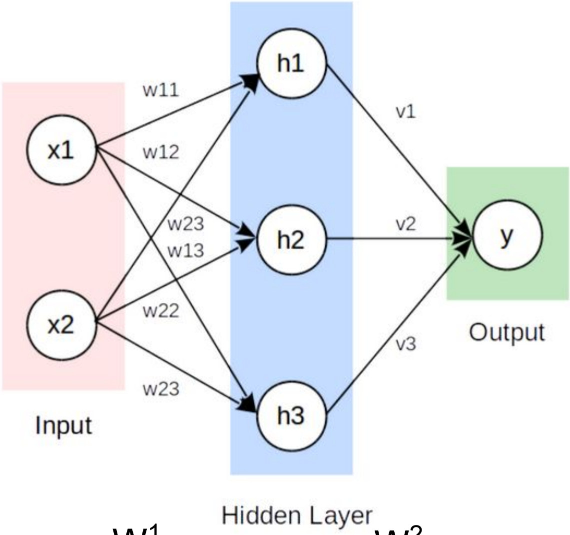
$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L).$$

- In vector form

$$\delta^L = \nabla_a C \odot \sigma'(z^L).$$

- For example, squared error Loss function, this should look familiar

$$\delta^L = (a^L - y) \odot \sigma'(z^L)$$



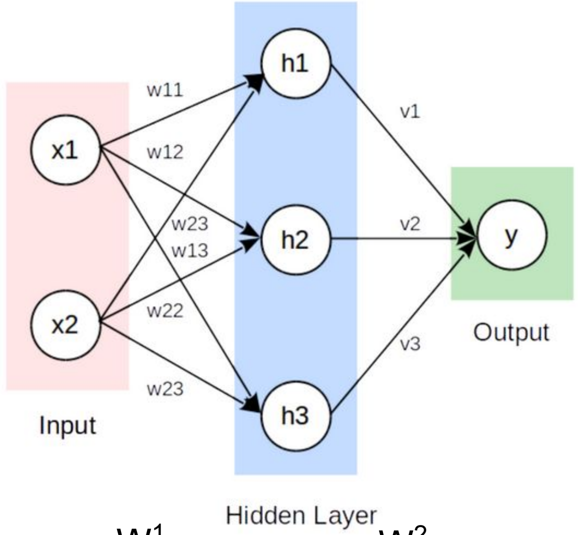
W^1 W^2

b^1 b^2

Generic Formulation for MLPs

- Then recurse from right to left.
- We want expressions relating the weights in each layer and the error in the adjacent (to the right) layer

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l),$$



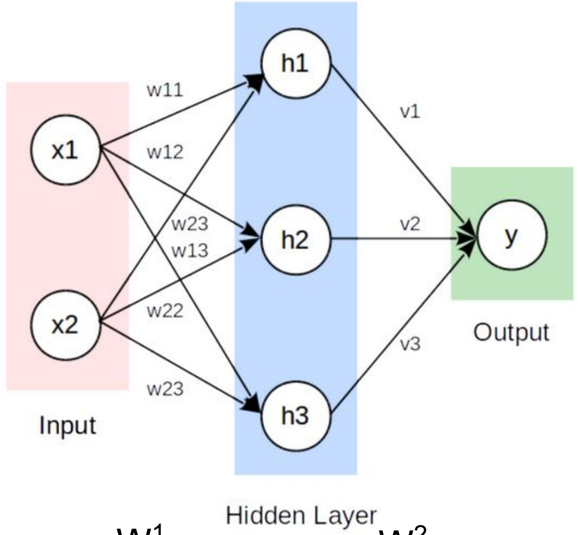
W^1 W^2

b^1 b^2

Generic Formulation for MLPs

- Now we have the ingredient for out partial derivatives
- For bias weights (for any layer):

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$



W^1 W^2

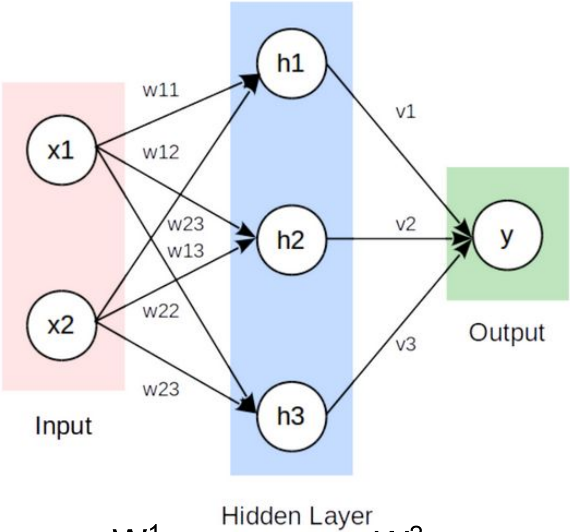
b^1 b^2

Generic Formulation for MLPs

- Now we have the ingredient for out partial derivatives
- For connection weights (for any layer):

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j.$$

$$\frac{\partial C}{\partial w} = a_{\text{in}} \delta_{\text{out}}.$$



W^1 W^2

b^1 b^2

Generic Formulation for MLPs

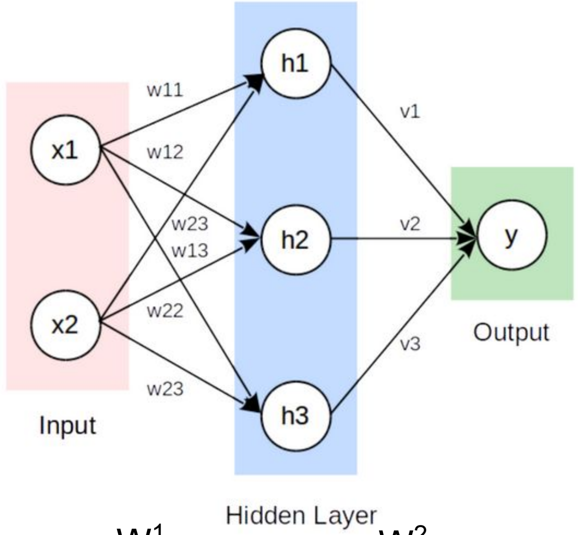
- Now we have the ingredient for out partial derivatives
- For connection weights (for any layer):

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j.$$

$$\frac{\partial C}{\partial w} = a_{\rm in} \delta_{\rm out}.$$

How much the output of this unit changes, as its input changes (ie. the amount of input)

How much the Loss changes when the output of this unit changes



W^1 W^2

b^1 b^2

Generic Formulation for MLPs

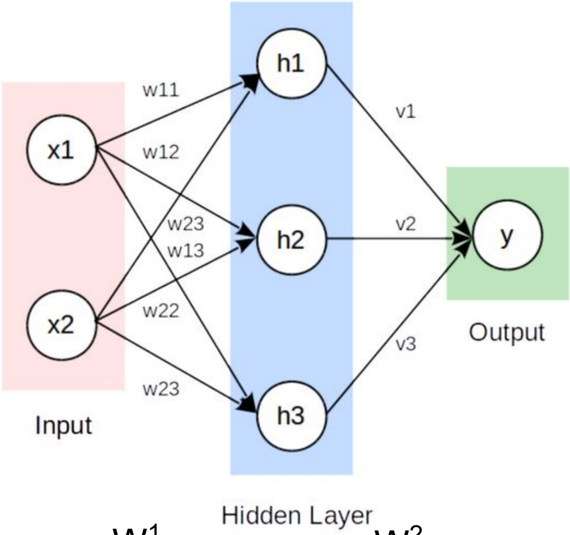
Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \quad \text{(BP3)}$$

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \quad \text{(BP4)}$$



W^1 W^2

b^1 b^2

Generic Formulation for MLPs

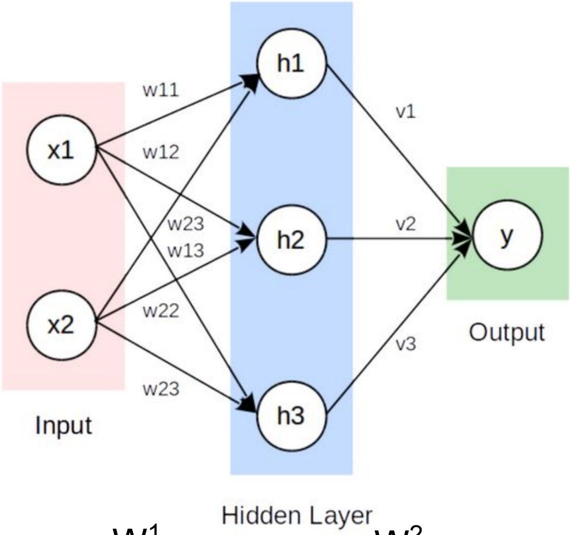
Summary: the equations of backpropagation

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad \text{(BP1)}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \quad \text{(BP2)}$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_j \quad \text{(BP3)}$$

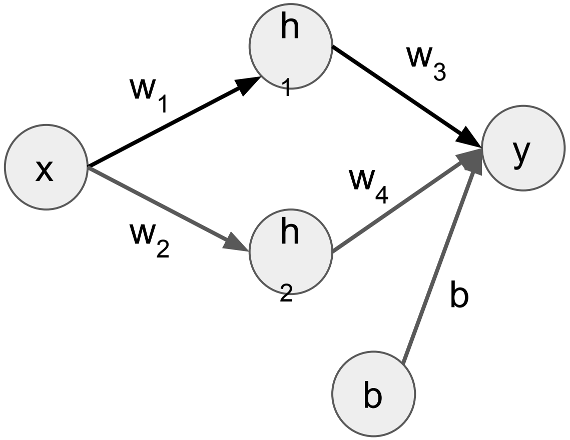
$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \quad \text{(BP4)}$$



W^1 W^2

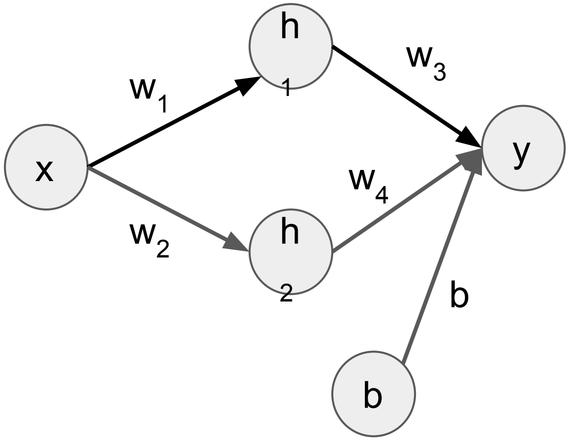
b^1 b^2

Example - Regression Network



- W^1, W^2, b

Example - Regression Network



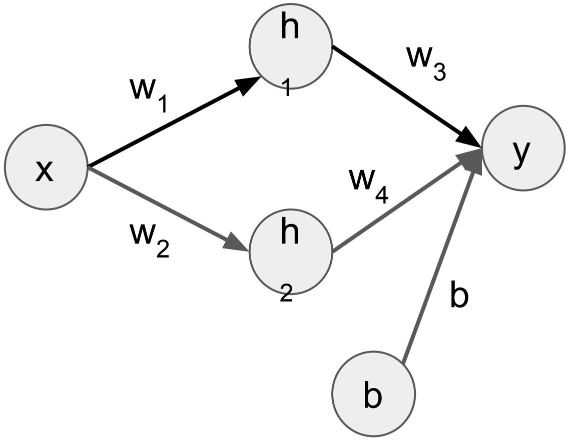
- 1. Bias

$$\delta^L = \frac{\partial C}{d\hat{y}} = \frac{\partial}{\partial \hat{y}} \frac{1}{2}(y_i - \hat{y}_i)^2 = (y_i - \hat{y}_i) = e_i$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial b} = e_i$$

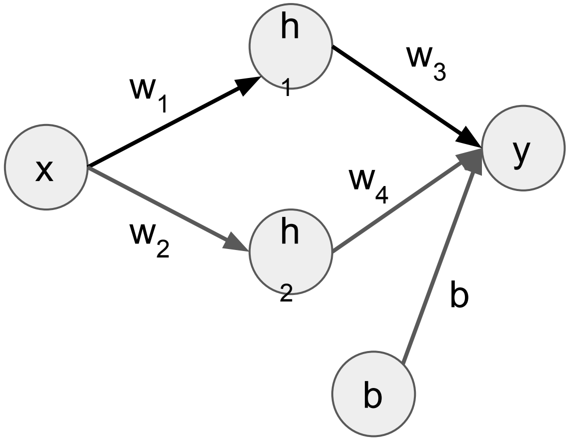
Example - Regression Network



- 2. Output layer weights W^2

$$\frac{\partial C}{\partial W^2} = a^{l-1}\delta^L$$
$$= \begin{bmatrix} \mathrm{out}_1 \\ \mathrm{out}_2 \end{bmatrix} \cdot e_i$$
$$= \begin{bmatrix} \mathrm{out}_1 e_i \\ \mathrm{out}_2 e_i \end{bmatrix}$$

Example - Regression Network



- 3. Hidden layer weights W^1

$$\frac{\partial C}{\partial W^l} = a^{l-1}\delta^l$$

$$a^0 = x$$

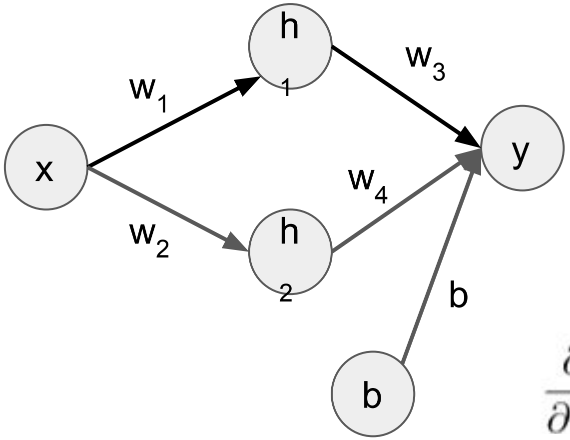
$$\delta^1 = (W^2\delta^L) \odot \sigma'(\text{in}_j)$$

$$= \begin{bmatrix} w_3 \\ w_4 \end{bmatrix}^T \cdot e_i \odot [\sigma(\text{in}_j)(1-\sigma(\text{in}_j))]$$

$$= \begin{bmatrix} w_3 e_i \sigma(\text{in}_1)(1-\sigma(\text{in}_1)) \\ w_4 e_i \sigma(\text{in}_2)(1-\sigma(\text{in}_2)) \end{bmatrix}$$

$$\frac{\partial C}{\partial W^1} = \begin{bmatrix} x w_3 e_i \sigma(\text{in}_1)(1-\sigma(\text{in}_1)) \\ x w_4 e_i \sigma(\text{in}_2)(1-\sigma(\text{in}_2)) \end{bmatrix}$$

Example - Regression Network



- Compare to previous result

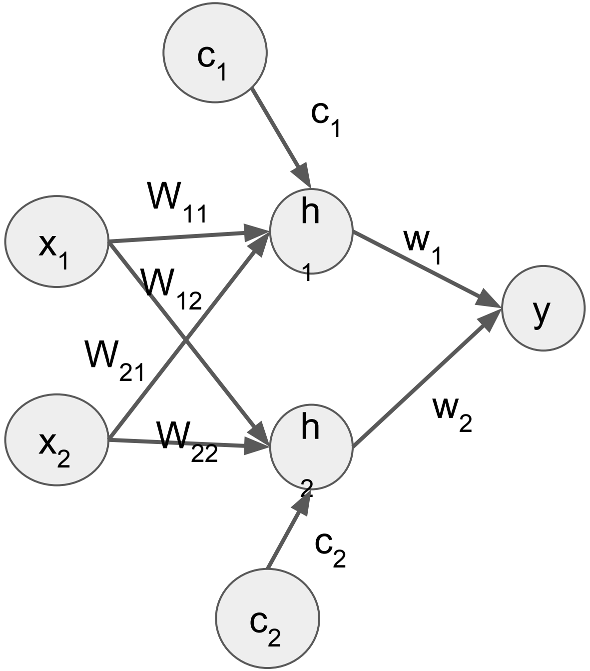
$$\nabla_\theta = \begin{bmatrix} -x_i e_i w_3 \sigma(\text{in}_1)(1-\sigma(\text{in}_1)) \\ -x_i e_i w_4 \sigma(\text{in}_2)(1-\sigma(\text{in}_2)) \\ -\text{out}_1 e_i \\ -\text{out}_2 e_i \\ -e_i \end{bmatrix}$$

$$\frac{\partial C}{\partial W^1} = \begin{bmatrix} x w_3 e_i \sigma(\text{in}_1)(1-\sigma(\text{in}_1)) \\ x w_4 e_i \sigma(\text{in}_2)(1-\sigma(\text{in}_2)) \end{bmatrix}$$

$$\frac{\partial C}{\partial W^2} = \begin{bmatrix} \text{out}_1 e_i \\ \text{out}_2 e_i \end{bmatrix}$$

$$\frac{\partial C}{\partial b} = e_i$$

Exercise - Classification Network

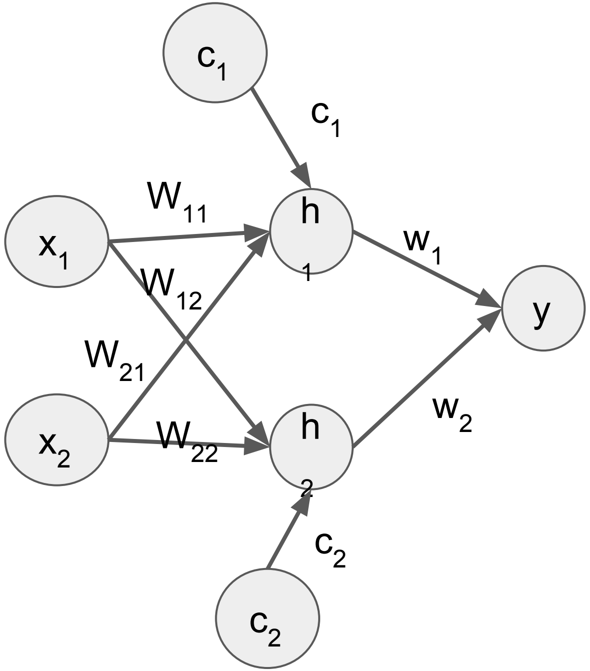


Recall this example from the book (XOR), where we have 2-D input data and need to classify them with with an obviously non-linear decision boundary.

This model has 8 free parameters. Assume ReLu activations in the hidden layer and a sigmoid output for the classifier. Loss function will be binary cross-entropy.

1. Derive the gradients for the free parameters in the model
2. Use gradient descent to fit our model to the 4 datapoint dataset.
3. Do your estimated weights match up with the ones in the book? Do they need to?

Exercise - Classification Network

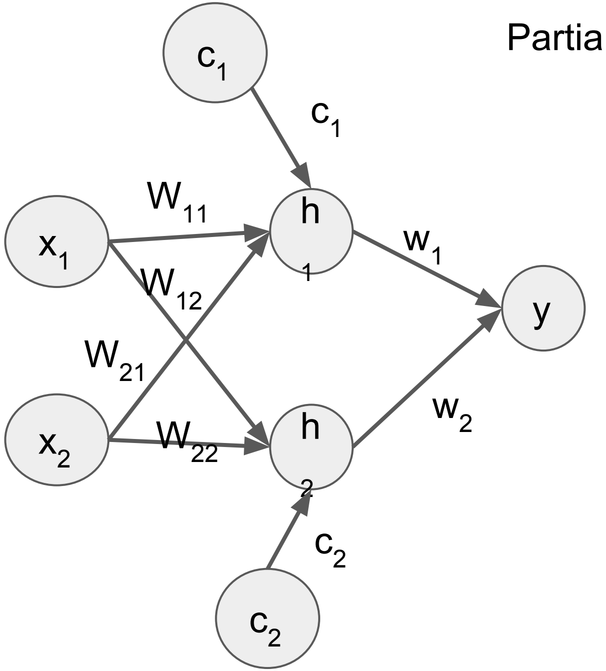


Helpful hint -

$$L = -y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)$$

$$\begin{aligned}
\frac{dL}{d\hat{y}} &= \frac{d}{d\hat{y}}(-y_i \log(\hat{y}_i) - (1 - y_i)\log(1 - \hat{y}_i)) \\
&= -\left(\frac{d(-y_i \log(\hat{y}_i))}{d\hat{y}} + \frac{d((1 - y_i)\log(1 - \hat{y}_i))}{d\hat{y}}\right) \\
&= -\left(\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i}\right)
\end{aligned}$$

Exercise - Classification



Partial Solution

$$\frac{dL}{d\hat{y}} = -\left(\frac{y_i}{\hat{y}_i} - \frac{1-y_i}{1-\hat{y}_i}\right)$$

$$\frac{\partial L}{\partial w_1} = \frac{dL}{d\hat{y}}\mathrm{out}_1$$

$$\frac{\partial L}{\partial w_2} = \frac{dL}{d\hat{y}}\mathrm{out}_2$$

$$\frac{\partial L}{\partial c_1} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_1}\frac{d\mathrm{out}_1}{d\mathrm{in}_1}\frac{\partial \mathrm{in}_1}{\partial c_1}$$

$$\frac{\partial L}{\partial c_2} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_2}\frac{d\mathrm{out}_2}{d\mathrm{in}_2}\frac{\partial \mathrm{in}_2}{\partial c_2}$$

$$\frac{\partial L}{\partial W_{11}} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_1}\frac{d\mathrm{out}_1}{d\mathrm{in}_1}\frac{\partial \mathrm{in}_1}{\partial W_{11}}$$

$$\frac{\partial L}{\partial W_{21}} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_1}\frac{d\mathrm{out}_1}{d\mathrm{in}_1}\frac{\partial \mathrm{in}_1}{\partial W_{21}}$$

$$\frac{\partial L}{\partial W_{12}} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_2}\frac{d\mathrm{out}_2}{d\mathrm{in}_2}\frac{\partial \mathrm{in}_2}{\partial W_{12}}$$

$$\frac{\partial L}{\partial W_{22}} = \frac{dL}{d\hat{y}}\frac{\partial \hat{y}}{\partial \mathrm{out}_2}\frac{d\mathrm{out}_2}{d\mathrm{in}_2}\frac{\partial \mathrm{in}_2}{\partial W_{22}}$$