349:Machine Learning

Fall 2024

Neural Networks
Part 2

Training

Step 1: The neural network is presented with a series input/output pairs $\langle \bar{x}_s, y_s \rangle \in S$ from the training set.

Step 2: The neural network is used to perform inference (a forward pass) to generate an estimated output $\hat{y}_s$.

Step 3: A cost function (see next slides) is used to quantify how well (or poorly) the neural network performed by comparing $\hat{y}_s$ and y_s .

Step 4: Back propagation (see next slides) is performed by calculating gradients with respect to each trainable variable, which are used to update trainable variables:

$$\hat{y}_k = g\left(\sum_{j=0}^{M} w_{jk} f\left(\sum_{i=0}^{N} w_{ij} x_i + b_j\right) + b_k\right)$$

The updates can be performed as:

- Stochastic Gradient Descent: Network parameters are updated for each training example, or
- Batch Gradient Descent: Gradients are accumulated over the entire training set and applied all at once.

Cost Functions

Certain cost functions are associated with typical output layers, but they are unconstrained and are influenced by the neural network architecture:

- Regression uses Mean Square Error (MSE):

$$J = \frac{1}{2} \sum_{S} \left(\hat{y}_s - y_s \right)^2$$

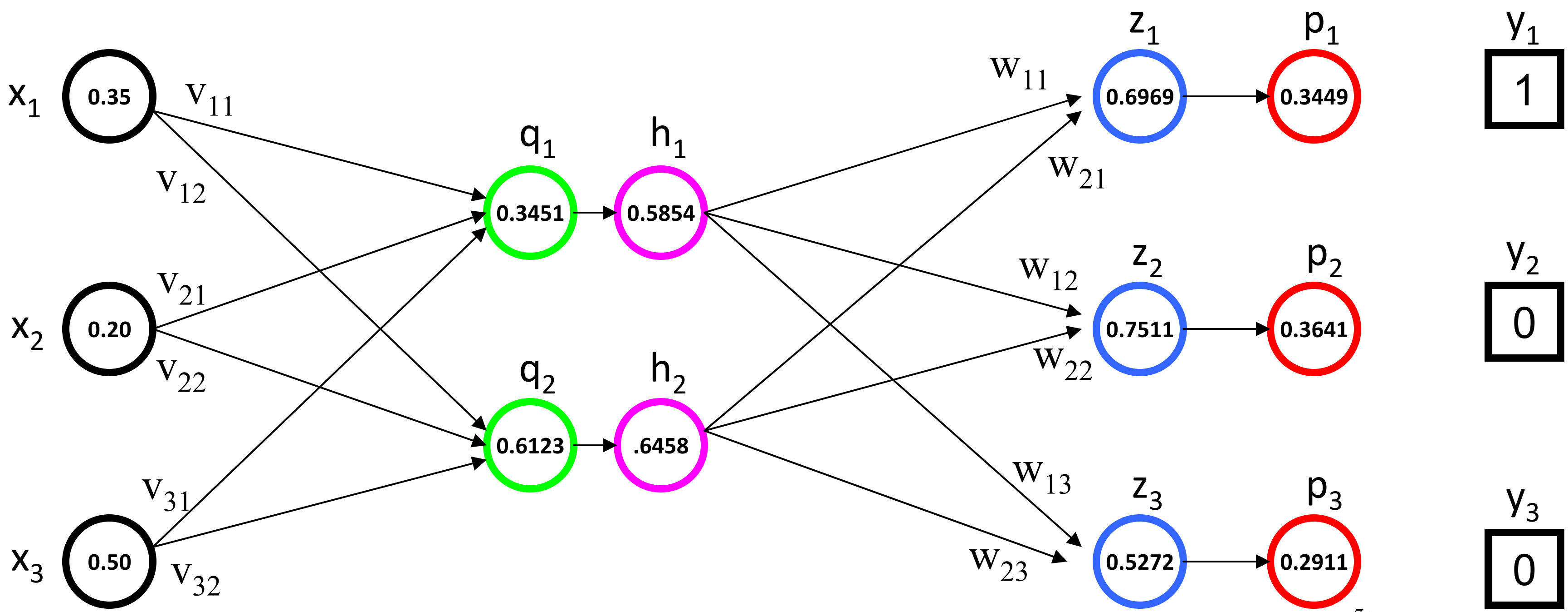
- Binary-Classification uses two-class cross-entropy:

$$J = -\left(y_s \ln(\hat{p}_s) + (1 - y_s) \ln(1 - \hat{p}_s) \right)$$

- Multi-Class Classification uses multi-class cross-entropy:

$$J = -\sum_{c \in C} y_c^{(s)} \ln(\hat{p}_c^{(s)}) + (1 - y_c^{(s)}) \ln(1 - \hat{p}_c^{(s)})$$

Gradient Descent -- Architecture



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

$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

Inputs X (1 x 3)

0.3500	
0.2000	
0.5000	

Weights V (3 x 2)

0.4963	0.7682
0.0885	0.1320
0.3074	0.6341

Linear Q (1 x 2)

0.3451
0.6123

Hidden H (1 x 2)

0.5854
0.6485

Weights W (2 x 3)

0.4901	0.8964	0.4556
0.6323	0.3489	0.4017

Logits Z (1 x 3)

0.6969
0.7511
0.5272

Probs P (1 x 3)

34.5%
36.4%
29.1%

Labels Y (1 x 3)

Good
Neutral
Bad

Gradient Descent -- Equations

Cost Function: Multi-Class Cross-Entropy

$$L = -\sum_{i=0}^{C} y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i)$$

Activation Function: Sigmoid

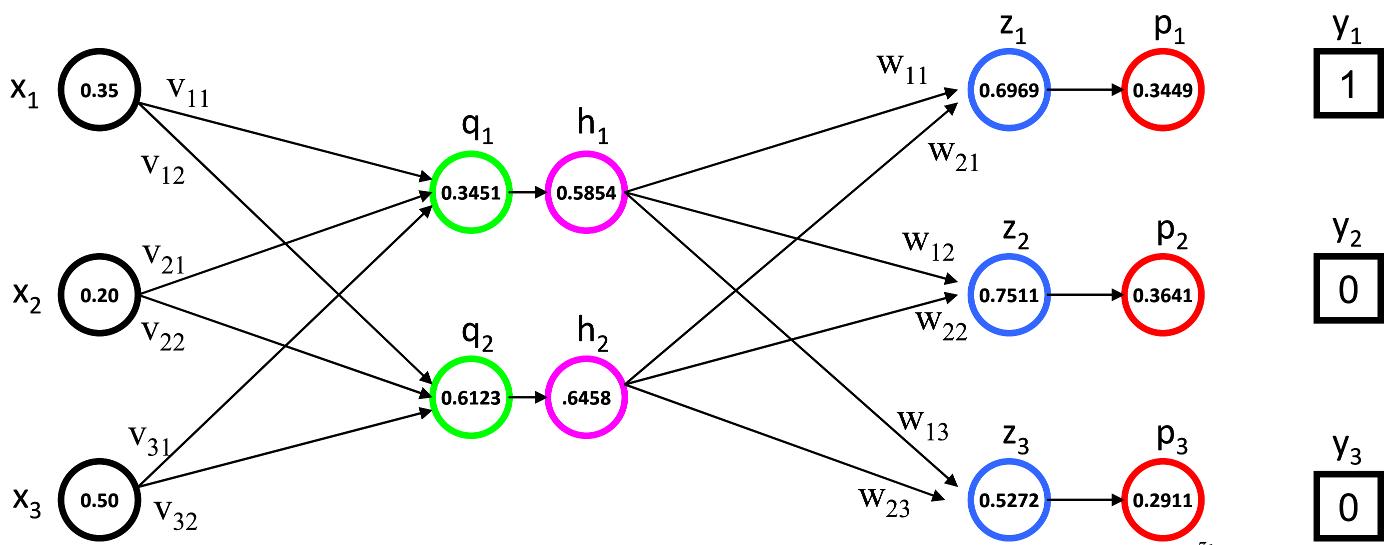
$$\sigma(q) = \frac{1}{1 + e^{-q}}$$

Softmax:

$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

$$= \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

Gradient Descent -- Forward Pass Calculations



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

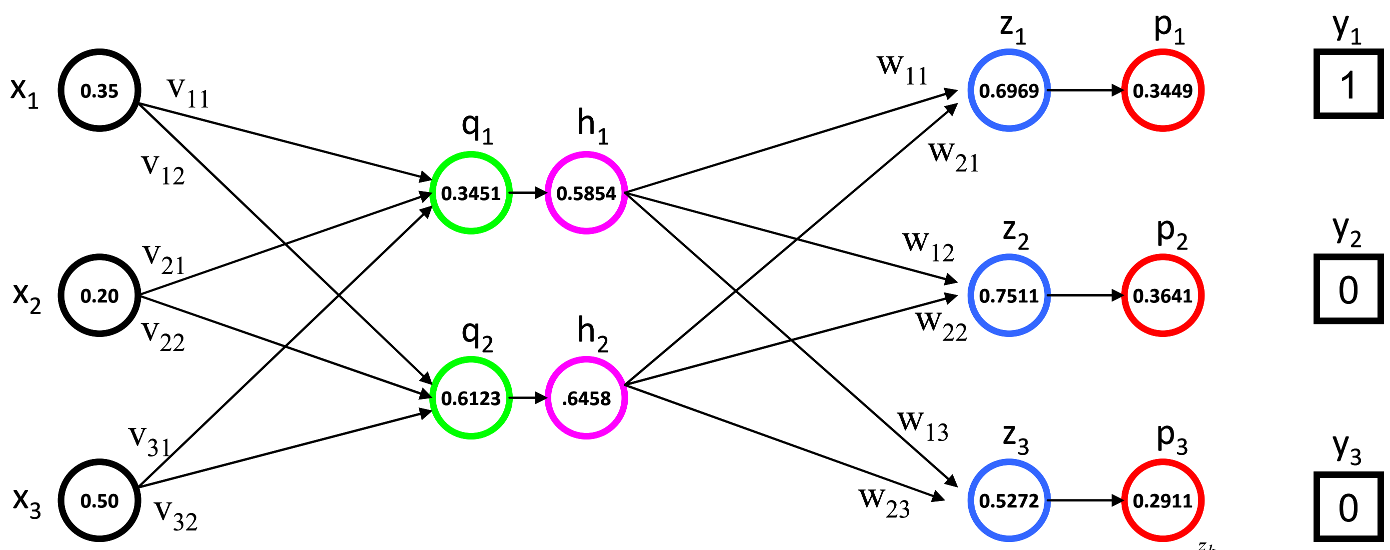
$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

Inputs X (1 x 3)	Weights V (3 x 2)	Linear Q (1 x 2)	Hidden H (1 x 2)	Weights W (2 x 3)	Logits Z (1 x 3)	Probs P (1 x 3)	Labels Y (1 x 3)
0.3500	0.4963 0.7682	0.3451	0.5854	0.4901 0.8964 0.4556	0.6969	34.5%	Good
0.2000	0.0885 0.1320	0.6123	0.6485	0.6323 0.3489 0.4017	0.7511	36.4%	Neutral
0.5000	0.3074 0.6341				0.5272	29.1%	Bad

$$\begin{bmatrix} 0.3500 & 0.2000 & 0.500 \end{bmatrix} \bullet \begin{bmatrix} 0.4963 & 0.7682 \\ 0.0885 & 0.1320 \\ 0.3074 & 0.6341 \end{bmatrix} = \begin{bmatrix} 0.3451 \\ 0.6123 \end{bmatrix}$$

0.4963 X 0.3500 + 0.0885 X 0.2000 + 0.3074 X 0.500 = 0.3451

Gradient Descent -- Forward Pass Calculations



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

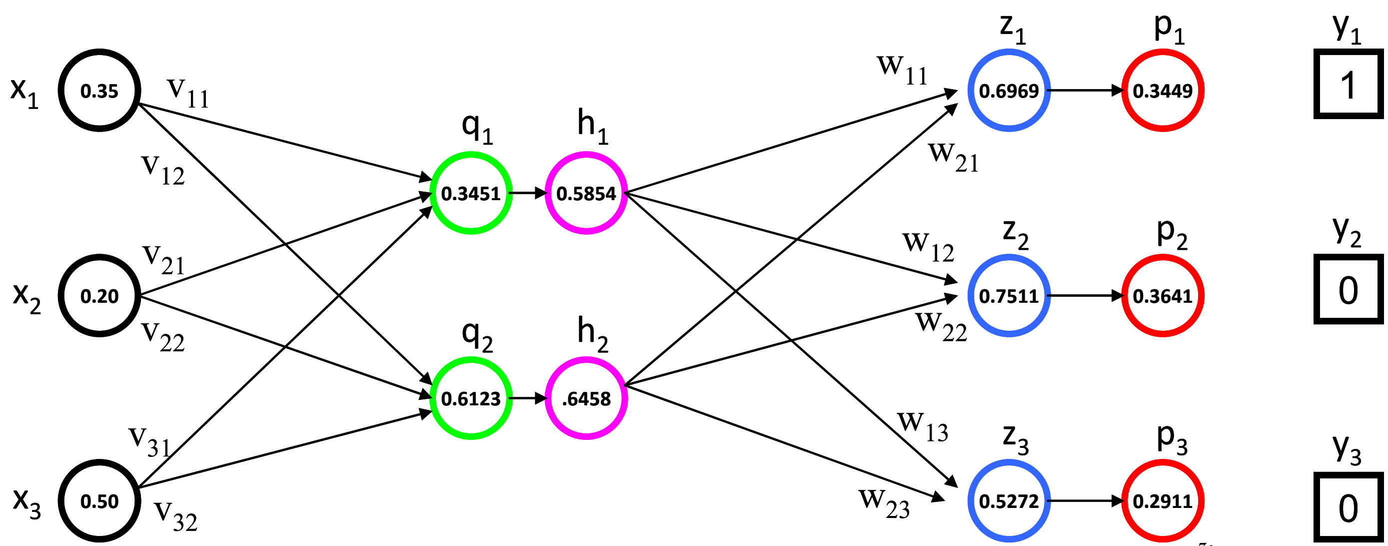
$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

Inputs X (1 x 3)	Weights V (3 x 2)	Linear Q (1 x 2)	Hidden H (1 x 2)	Weights W (2 x 3)	Logits Z (1 x 3)	Probs P (1 x 3)	Labels Y (1 x 3)
0.3500	0.4963 0.7682	0.3451	0.5854	0.4901 0.8964 0.4556	0.6969	34.5%	**Good**
0.2000	0.0885 0.1320	0.6123	0.6485	0.6323 0.3489 0.4017	0.7511	36.4%	**Neutral**
0.5000	0.3074 0.6341				0.5272	29.1%	**Bad**

$$\sigma\left(\begin{matrix} 0.3451 \\ 0.6123 \end{matrix}\right) = \begin{matrix} 0.5854 \\ 0.6458 \end{matrix}$$

1 / (1 + exp(-0.3451) = 0.5854

Gradient Descent -- Forward Pass Calculations



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

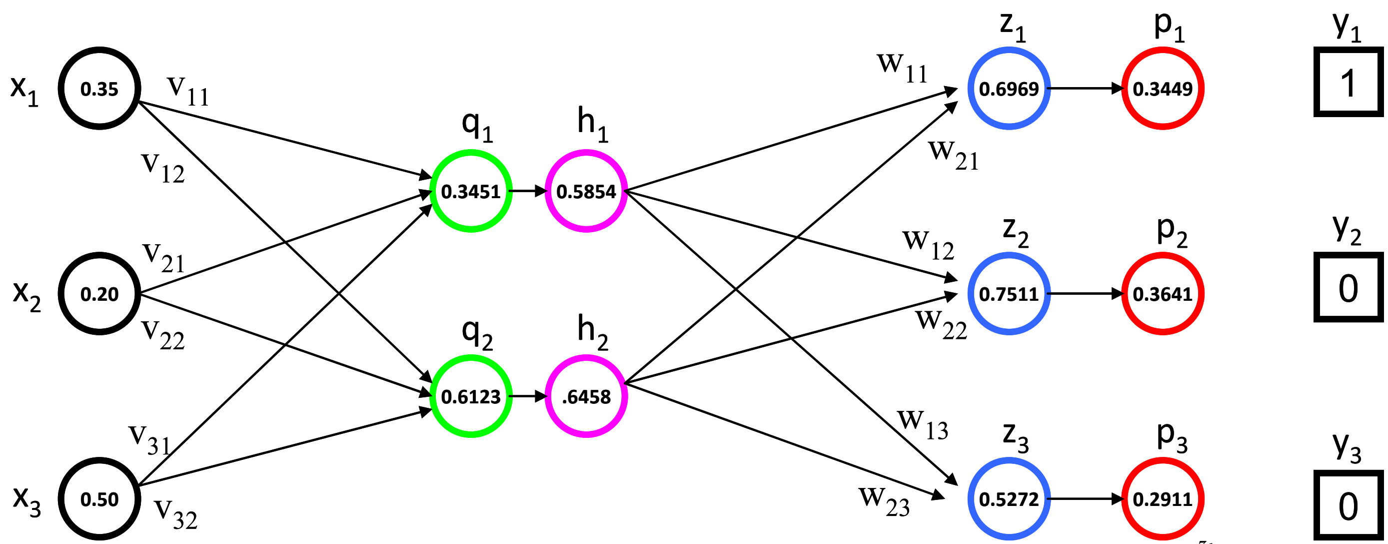
$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

Inputs X (1 x 3)	Weights V (3 x 2)	Linear Q (1 x 2)	Hidden H (1 x 2)	Weights W (2 x 3)	Logits Z (1 x 3)	Probs P (1 x 3)	Labels Y (1 x 3)
0.3500	0.4963 0.7682	0.3451	0.5854	0.4901 0.8964 0.4556	0.6969	34.5%	Good
0.2000	0.0885 0.1320	0.6123	0.6485	0.6323 0.3489 0.4017	0.7511	36.4%	Neutral
0.5000	0.3074 0.6341				0.5272	29.1%	Bad

$$\begin{bmatrix} 0.5854 & 0.6485 \end{bmatrix} \bullet \begin{bmatrix} 0.4901 & 0.8964 & 0.4556 \\ 0.6323 & 0.3489 & 0.4017 \end{bmatrix} = \begin{bmatrix} 0.6969 \\ 0.7511 \\ 0.5272 \end{bmatrix}$$

0.4901 X 0.5854 + 0.6323 X 0.6458 = 0.6969

Gradient Descent -- Forward Pass Calculations



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

$$p_k = \frac{e^{z_k}}{\sum_{c \in C} e^{z_c}}$$

Inputs X (1 x 3)	Weights V (3 x 2)		Linear Q (1 x 2)	Hidden H (1 x 2)	Weights W (2 x 3)			Logits Z (1 x 3)	Probs P (1 x 3)	Labels Y (1 x 3)
0.3500	0.4963	0.7682	0.3451	0.5854	0.4901	0.8964	0.4556	0.6969	34.5%	**Good**
0.2000	0.0885	0.1320	0.6123	0.6485	0.6323	0.3489	0.4017	0.7511	36.4%	**Neutral**
0.5000	0.3074	0.6341						0.5272	29.1%	**Bad**

$$\text{softmax}\left(\begin{matrix}0.6969\\0.7511\\0.5272\end{matrix}\right) = \begin{matrix}34.5\%\\36.4\%\\29.1\%\end{matrix}$$

exp(0.6969)/[exp(0.6969) + exp(0.7511) + exp(0.5272)] = 0.3449

Gradient Descent -- Some Derivatives

Cost Function: Multi-Class Cross-Entropy

$$\frac{\partial L}{\partial p_i} = -\frac{1}{p_i}$$ for y =1 $$\frac{\partial L}{\partial p_i} = -\frac{1}{1-p_i}$$ for y =0

Activation Function: Sigmoid

$$\frac{\partial \sigma(q)}{\partial q} = \sigma(q)(1-\sigma(q))$$

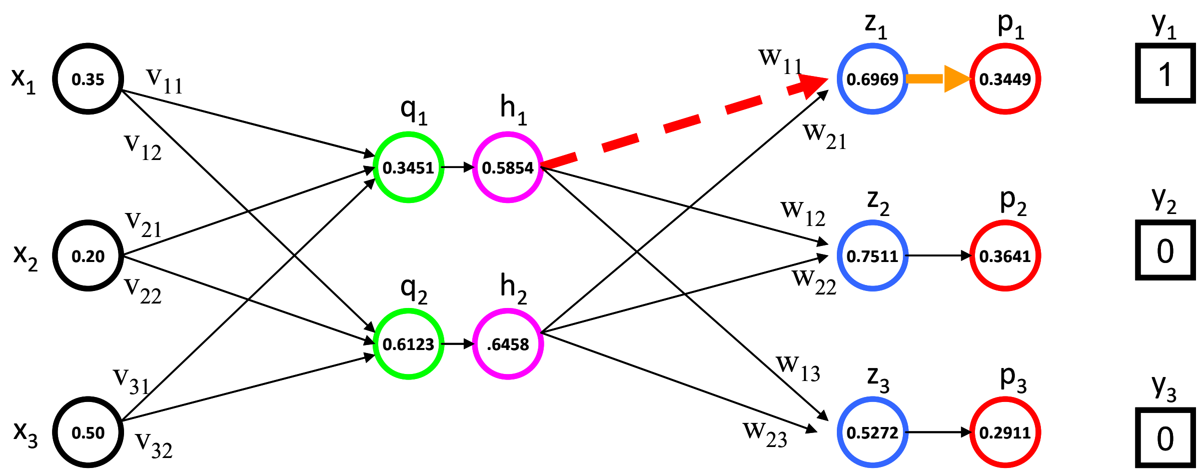
Softmax:

$$\frac{\partial \texttt{softmax}(z_k)}{\partial z_k} = \frac{e^{z_k}(e^{z_0} + e^{z_1} + \cdots + e^{z_c} - e^{z_k})}{(e^{z_0} + e^{z_1} + \cdots + e^{z_c})^2}$$

Linear Transform:

$$\frac{\partial w \cdot x}{\partial w} = x$$

Gradient Descent -- Gradient Calculation



Chain Rule:

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f(g(x))}{\partial g(x)} \cdot \frac{\partial g(x)}{\partial x}$$

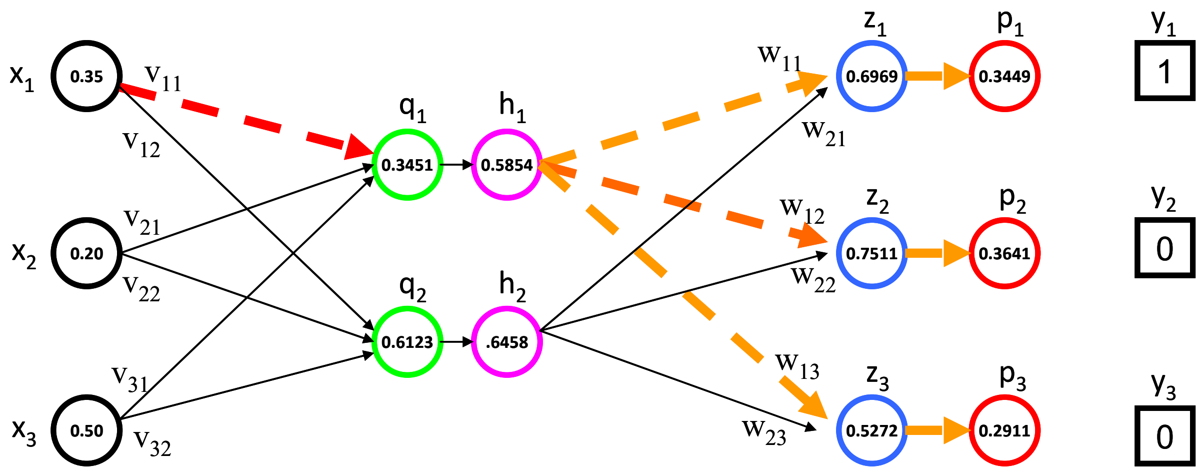
$$\frac{\partial L}{\partial w_{11}} = \frac{\partial L}{\partial p_1} \cdot \frac{\partial p_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{11}}$$

$$= -\frac{1}{p_1} \cdot \frac{e^{z_1} \cdot (e^{z_2} + e^{z_3})}{(e^{z_1} + e^{z_2} + e^{z_3})^2} \cdot h_1$$

$$= -\frac{1}{0.3449} \cdot \frac{e^{0.6969} \cdot (e^{0.7511} + e^{0.5272})}{(e^{0.6969} + e^{0.7511} + e^{0.5272})^2} \cdot 0.5854$$

$$= -0.3835$$

Gradient Descent -- Gradient Calculation



$$\frac{\partial L}{\partial v_{11}} = \frac{\partial L}{\partial p_1}\cdot\frac{\partial p_1}{\partial z_1}\cdot\frac{\partial z_1}{\partial h_1}\cdot\frac{\partial h_1}{\partial q_1}\cdot\frac{\partial q_1}{\partial v_{11}} + \frac{\partial L}{\partial p_2}\cdot\frac{\partial p_2}{\partial z_2}\cdot\frac{\partial z_2}{\partial h_1}\cdot\frac{\partial h_1}{\partial q_1}\cdot\frac{\partial q_1}{\partial v_{11}} + \frac{\partial L}{\partial p_3}\cdot\frac{\partial p_3}{\partial z_3}\cdot\frac{\partial z_3}{\partial h_1}\cdot\frac{\partial h_1}{\partial q_1}\cdot\frac{\partial q_1}{\partial v_{11}}$$

$$= -\frac{1}{p_1}\cdot\frac{e^{z_1}\cdot(e^{z_2}+e^{z_3})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}\cdot w_{11}\cdot\sigma(q_1)(1-\sigma(q_1))\cdot x_1 +$$

$$\frac{1}{(1-p_2)}\cdot\frac{e^{z_2}\cdot(e^{z_1}+e^{z_3})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}\cdot w_{12}\cdot\sigma(q_1)(1-\sigma(q_1))\cdot x_1 +$$

$$\frac{1}{1-p_3}\cdot\frac{e^{z_3}\cdot(e^{z_1}+e^{z_2})}{(e^{z_1}+e^{z_2}+e^{z_3})^2}\cdot w_{13}\cdot\sigma(q_1)(1-\sigma(q_1))\cdot x_1$$

$$= -0.0118$$

Gradient Descent -- Apply Gradients

Initialize $\theta^{(0)}$

Repeat until convergence:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta^{(t)}} L(X, Y; \theta^{(t)})$$

Return $\theta^{(t_{max})}$

$$\underset{Weights\ W^{(t)}}{\begin{bmatrix} 0.4901 & 0.8964 & 0.4556 \\ 0.6323 & 0.3489 & 0.4017 \end{bmatrix}} - 0.5 \underset{\nabla_{w_{11}} L(X,Y,\theta^{(t)})}{\begin{bmatrix} -0.3835 & - & - \\ - & - & - \end{bmatrix}} = \underset{Weights\ W^{(t+1)}}{\begin{bmatrix} 0.6819 & 0.8964 & 0.4556 \\ 0.6323 & 0.3489 & 0.4017 \end{bmatrix}}$$

$$\underset{Weights\ V^{(t)}}{\begin{bmatrix} 0.4963 & 0.7682 \\ 0.0885 & 0.1320 \\ 0.3074 & 0.6341 \end{bmatrix}} - 0.5 \underset{\nabla_{v_{11}} L(X,Y,\theta^{(t)})}{\begin{bmatrix} -0.0118 & - \\ - & - \\ - & - \end{bmatrix}} = \underset{Weights\ V^{(t+1)}}{\begin{bmatrix} 0.5022 & 0.7682 \\ 0.0885 & 0.1320 \\ 0.3074 & 0.6341 \end{bmatrix}}$$

Gradient Descent -- Additional Resources

Detailed Numerical Example:

https://medium.com/@14prakash/back-propagation-is-very-simple-who-made-it-complicated-97b794c97e5c

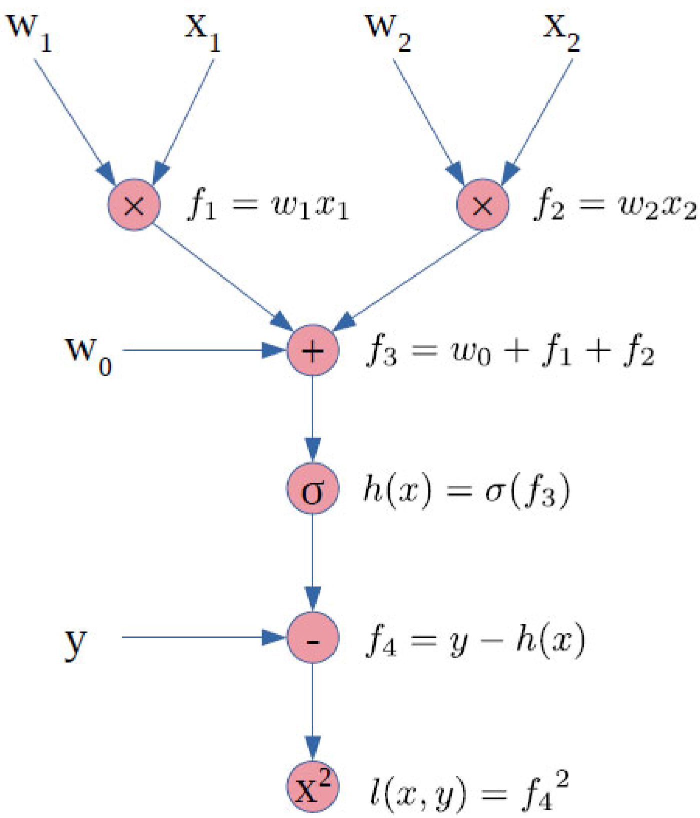
Gradient Calculation Video:

https://www.youtube.com/watch?v=lj7JYJSgpZI

Wolfram Alpha:

https://www.wolframalpha.com/

Calculation Graphs Forward Pass

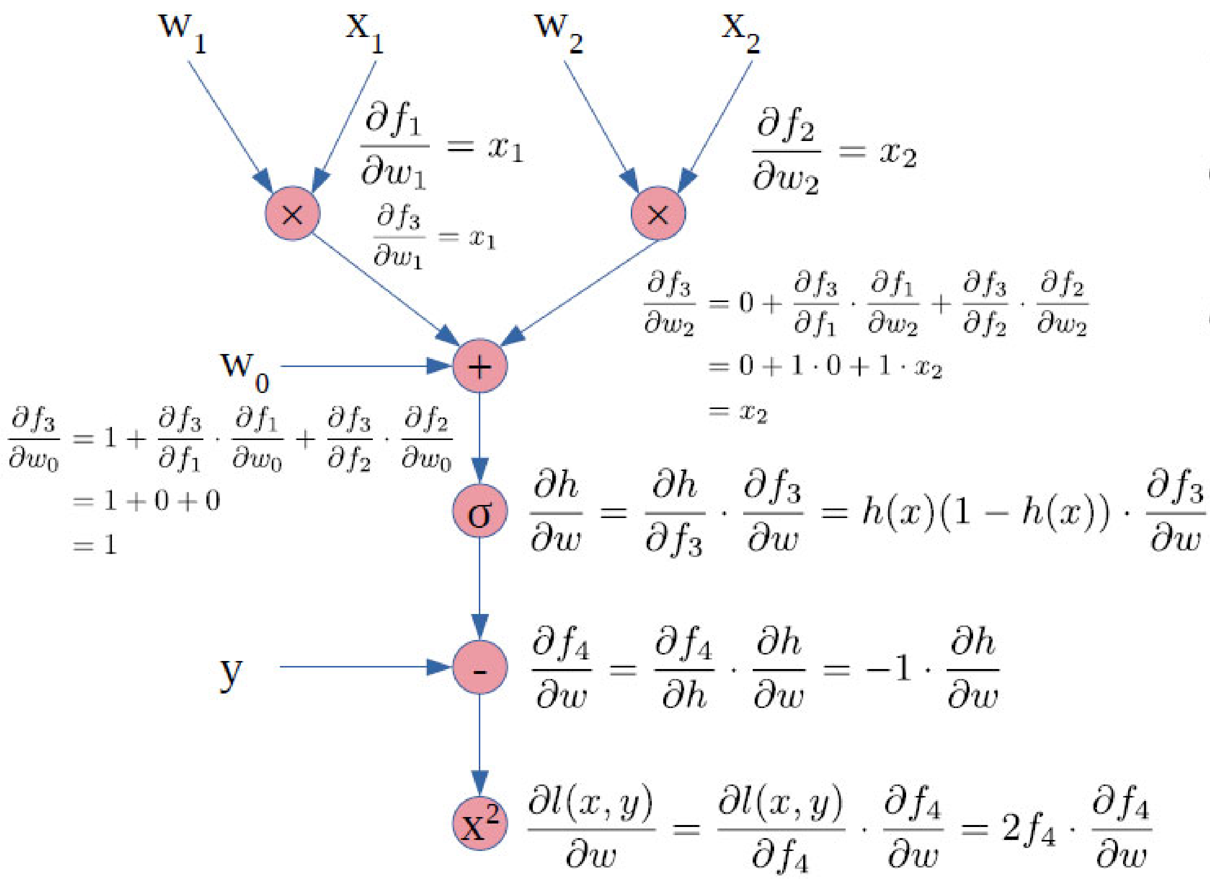


$$h(x) = \sigma\left(w_0 + w_1 x_1 + w_2 x_2\right)$$

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

$$l(x, y) = (y - h(x))^2$$

Calculation Graphs Gradients



$$h(x) = \sigma\left(w_0 + w_1 x_1 + w_2 x_2\right)$$

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

$$l(x, y) = (y - h(x))^2$$

Tunable Hyper-parameters

- Learning Rate is a “knob” that you can twist empirically
- Stochastic Gradient Descent vs. Batch Gradient Descent
- Drop Out randomly ignores certain weight dimensions during training
- Regularization minimizes a specific property of the neural network by modifying the cost function:

$$J = \frac{1}{2}\sum_{S}\left(\hat{y}_s - y_s\right)^2 + \lambda_1 \left\| w_{ij} \right\|_2 + \lambda_2 \left\| w_{jk} \right\|_2$$

- Scaling input values and weight initializations
- Gradient clipping and normalization
- Dynamic learning rate algorithms
- Many others