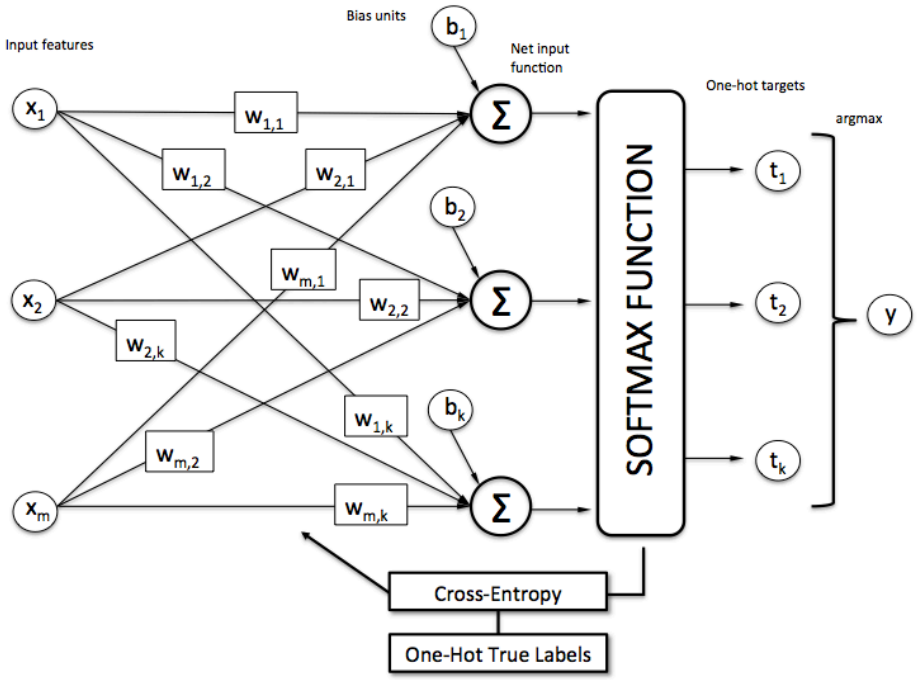
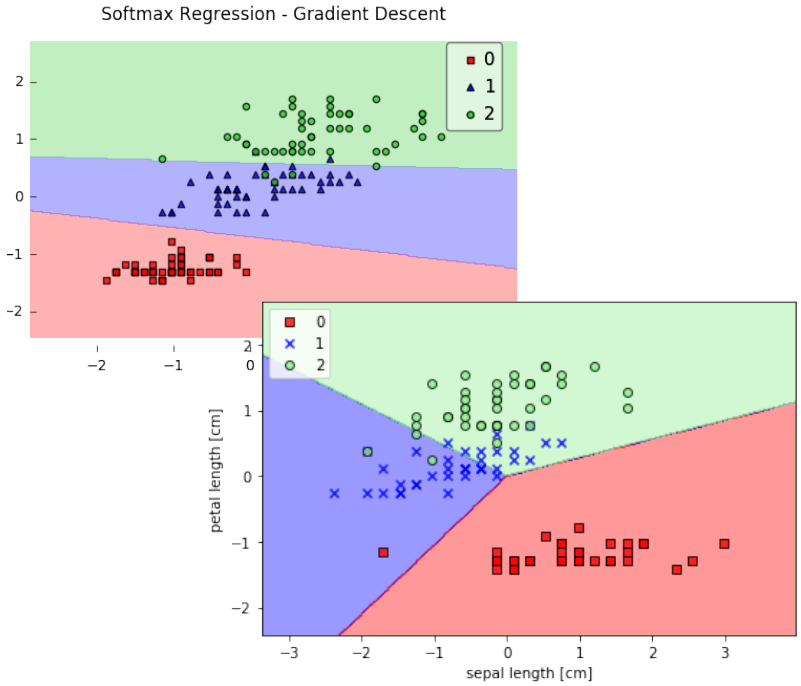
Logistic Regression

Deep Learning & Applications

Logistic Regression

- A better loss function for classification (cross entropy instead of MSE)
- Extending neurons to multi-class classification (multiple output notes + softmax)

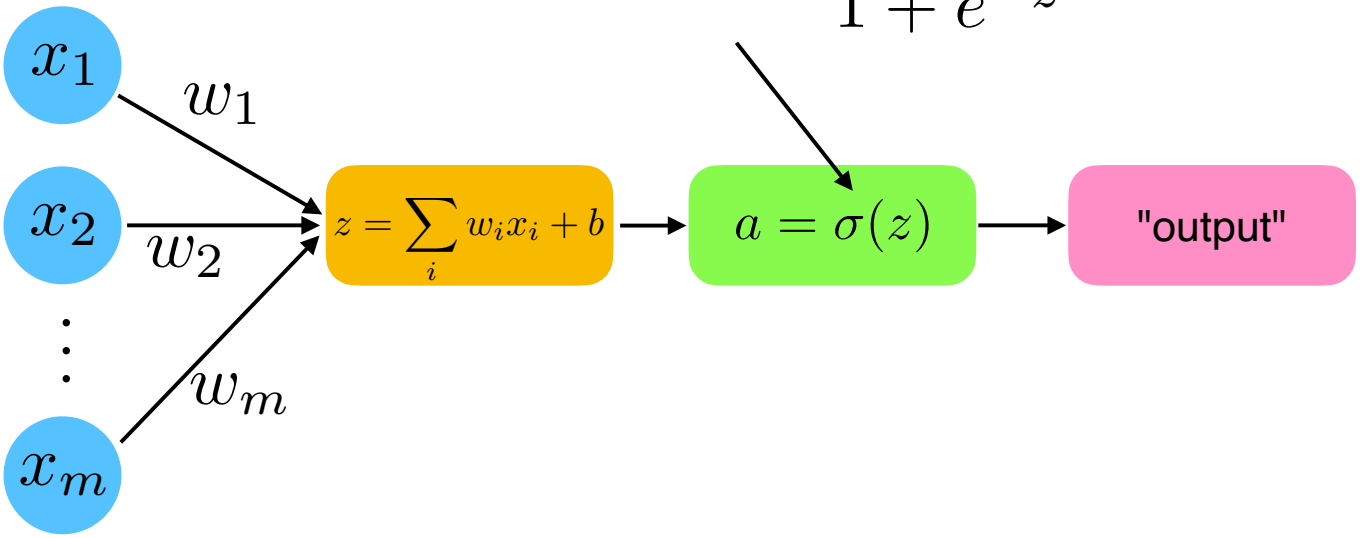




Logistic Regression Neuron

For binary classes $y \in \{0, 1\}$

"logistic sigmoid" $\sigma(z) = \dfrac{1}{1 + e^{-z}}$



In ADALINE, the activation function was an identity function, $\sigma(z) = z$

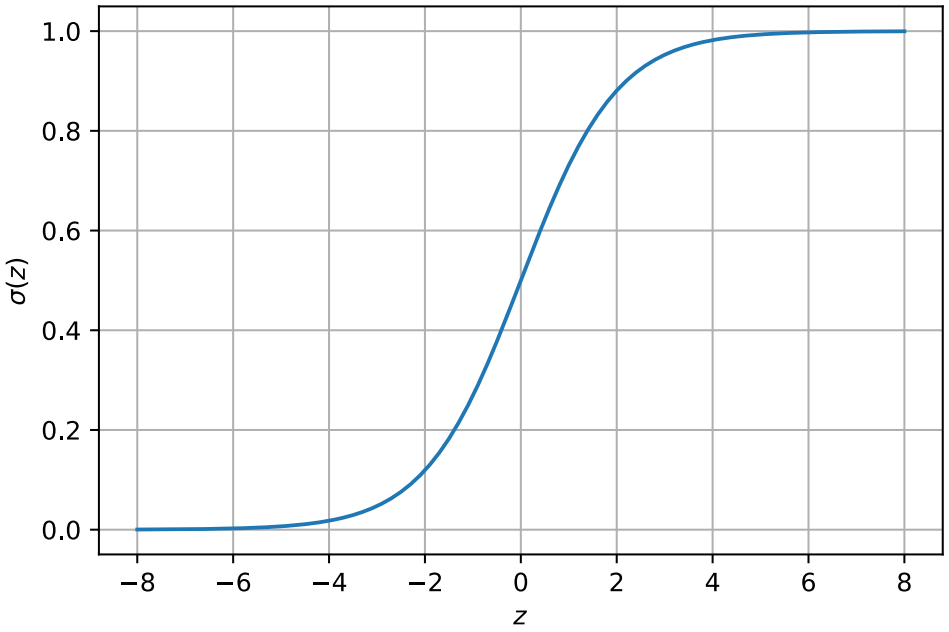
ADALINE we used MSE as loss function: $\mathrm{MSE} = \dfrac{1}{n} \sum_i (a^{[i]} - y^{[i]})^2$

We will use a different loss function for logistic regression

Logistic Sigmoid Function

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

activation



net input

Logistic Regression

Given the output:

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

here, the "hypothesis" is just our activation function output

$$h(\mathbf{x}) = a$$

We compute the posterior as:

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } y = 1 \\ 1 - h(\mathbf{x}) & \text{if } y = 0 \end{cases}$$

For a binary class problem (0 and 1), we want these probabilities to be:

$$P(y = 0|\mathbf{x}) \approx 1 \quad \text{if } y = 0$$

$$P(y = 1|\mathbf{x}) = 1 - P(y = 0|\mathbf{x}) \approx 1 \quad \text{if } y = 1$$

Logistic Regression

Given the output:

$$h(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$$

here, the "hypothesis" is just our activation function output

$$h(\mathbf{x}) = a$$

We compute the posterior as:

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{if } y = 1 \\ 1 - h(\mathbf{x}) & \text{if } y = 0 \end{cases}$$

rewrite more compactly

$$P(y|\mathbf{x}) = a^y (1-a)^{(1-y)}$$

Logistic Regression

And for multiple training examples, we want to maximize:

$$P\left(y^{[i]}, ..., y^{[n]} | \mathbf{x}^{[1]}, ..., \mathbf{x}^{[n]}\right) = \prod_{i=1}^{n} P\left(y^{[i]} | \mathbf{x}^{[i]}\right)$$

by finding *good/optimal* parameters

You may remember this as *Maximum Likelihood Estimation* from your other stats classes.

Likelihood "Loss"

$$
\begin{aligned}
L(\mathbf{w}) &= P(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \\
&= \prod_{i=1}^{n} P\big(y^{(i)} \mid x^{(i)}; \mathbf{w}\big) \\
&= \prod_{i=1}^{n} \Big(\sigma\big(z^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(z^{(i)}\big)\Big)^{1-y^{(i)}}
\end{aligned}
$$

Log-Likelihood "Loss"

$$
\begin{aligned}
L(\mathbf{w}) &= P(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) \\
&= \prod_{i=1}^{n} P\big(y^{(i)} \mid x^{(i)}; \mathbf{w}\big) \\
&= \prod_{i=1}^{n} \Big(\sigma\big(z^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(z^{(i)}\big)\Big)^{1-y^{(i)}}
\end{aligned}
$$

In practice, it is easier to maximize the (natural) log of this equation, which is called the log-likelihood function:

$$
\begin{aligned}
l(\mathbf{w}) &= \log L(\mathbf{w}) \\
&= \sum_{i=1}^{n} \Big[y^{(i)} \log\big(\sigma\big(z^{(i)}\big)\big) + \big(1 - y^{(i)}\big) \log\big(1 - \sigma\big(z^{(i)}\big)\big)\Big]
\end{aligned}
$$

Negative Log-Likelihood Loss

In practice, it is even more convenient to minimize negative log-likelihood instead of maximizing log-likelihood:

$$\mathcal{L}(\mathbf{w}) = -l(\mathbf{w})$$

$$= -\sum_{i=1}^{n} \left[y^{(i)} \log \left(\sigma\left(z^{(i)}\right) \right) + \left(1 - y^{(i)}\right) \log \left(1 - \sigma\left(z^{(i)}\right)\right) \right]$$

(in code, we also usually add a $1/n$ scaling factor for further convenience, where n is the number of training examples or number of examples in a minibatch)

Logistic Regression Loss

- So, "doing logistic regression" is similar to what we have done before in ADALINE, we minimized the MSE loss:

$$\text{MSE} = \frac{1}{n}\sum_i (a^{[i]} - y^{[i]})^2$$

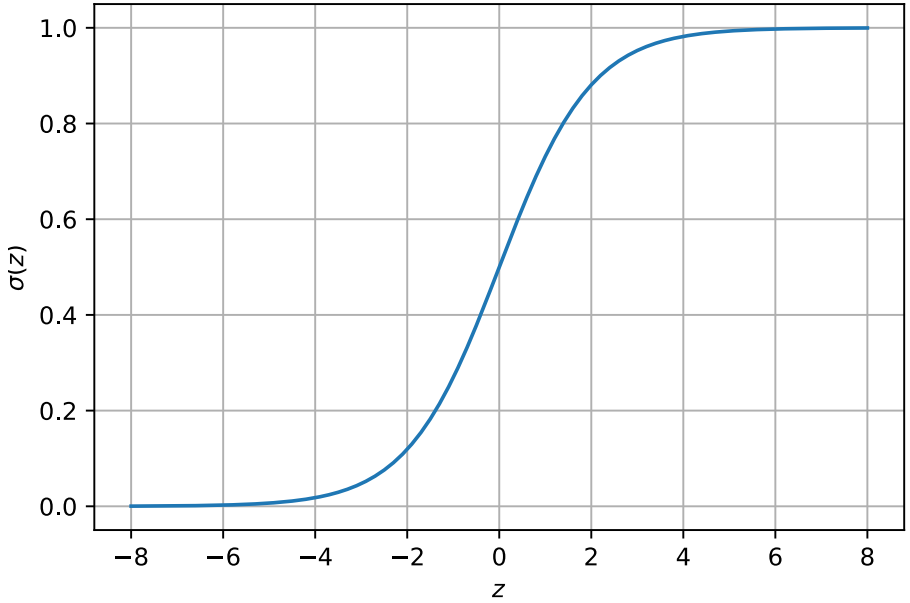
- However, the difference is that in Logistic Regression, we maximize the likelihood
- Maximizing likelihood is the same as maximizing the log-likelihood, but the latter is numerically more stable
- Maximizing the log-likelihood is the same as minimizing the negative log-likelihood, which is convenient, so we don't have to change our code and can still use gradient descent (instead of gradient ascent)

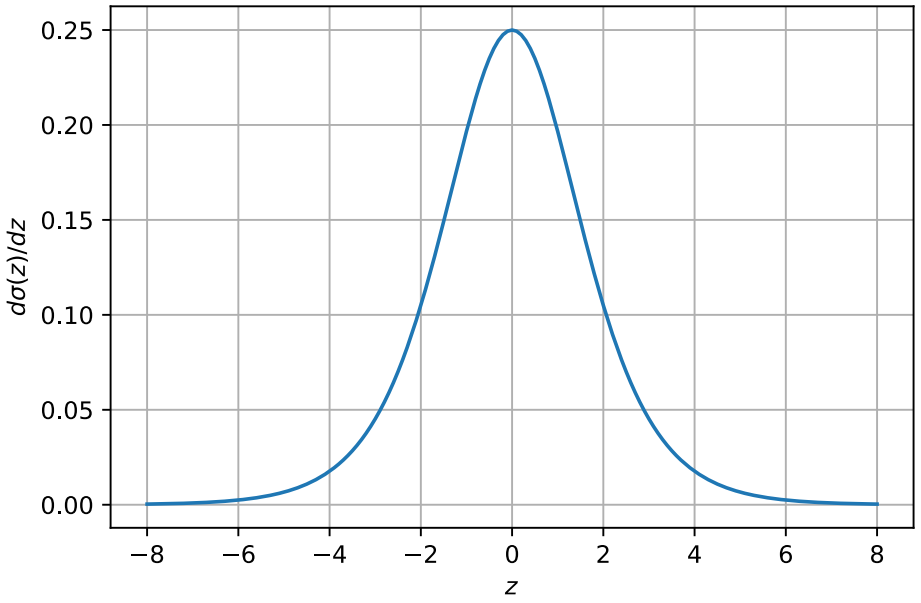
Logistic Sigmoid Derivative

Generally, another nice property of the logistic sigmoid (in multi-layer nets) is that it has nice derivatives!

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

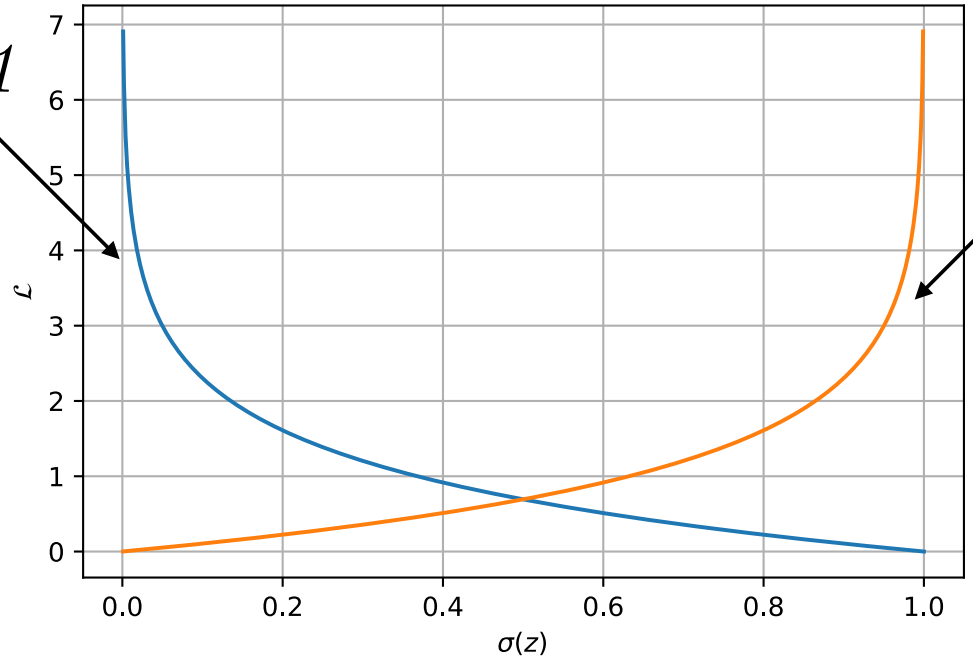
$$\frac{d}{dz}\sigma(z) = \frac{e^{-z}}{(1 + e^{-z})^2} = \sigma(z)(1 - \sigma(z))$$





Loss for a Single Training Example

if $y = 1$



if $y = 0$

$$\mathcal{L}(\mathbf{w}) = -y^{(i)} \log\left(\sigma\left(z^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - \sigma\left(z^{(i)}\right)\right)$$

Learning Rule

Same gradient descent rule as before, for which we need

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial a} \frac{da}{dz} \frac{\partial z}{\partial w_j}$$

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{a - y}{a - a^2}$$

$$\frac{da}{dz} = \frac{e^{-z}}{(1 + e^{-z})^2} = a \cdot (1 - a)$$

$$\frac{\partial z}{w_j} = x_j$$

Learning Rule for Logistic Regression

Same gradient descent rule as for ADALINE & Linear Regression, for which we need

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{\partial \mathcal{L}}{\partial a} \frac{da}{dz} \frac{\partial z}{\partial w_j}$$

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{a - y}{a - a^2}$$

$$\frac{da}{dz} = \frac{e^{-z}}{(1 + e^{-z})^2} = a \cdot (1 - a)$$

$$\frac{\partial z}{w_j} = x_j$$

$$\frac{\partial \mathcal{L}}{\partial z} = a - y$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = (a - y) x_j$$

Learning Rule for Logistic Regression

Remember the Linear Regression & ADALINE Lectures?

Stochastic gradient descent

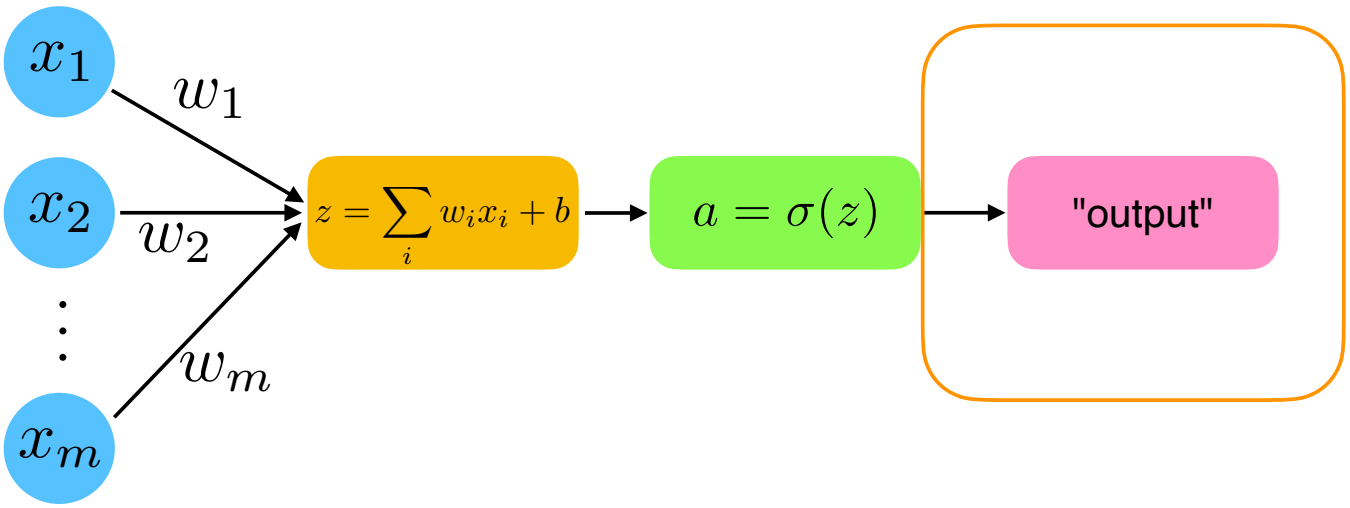
1. Initialize $\mathbf{w} := \mathbf{0} \in \mathbb{R}^m$, $\mathbf{b} := 0$
2. For every training epoch:
   A. For every $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D}$
      (a) $\hat{y}^{[i]} := \sigma\big(\mathbf{x}^{[i]T}\mathbf{w} + b\big)$
      (b) $\nabla_{\mathbf{w}}\mathcal{L} = -\big(y^{[i]} - \hat{y}^{[i]}\big)\mathbf{x}^{[i]}$
      $\nabla_b\mathcal{L} = -\big(y^{[i]} - \hat{y}^{[i]}\big)$
      (c) $\mathbf{w} := \mathbf{w} + \eta \times (-\nabla_{\mathbf{w}}\mathcal{L})$
      $b := b + \eta \times \underbrace{(-\nabla_b\mathcal{L})}_{\text{negative gradient}}$

learning rate

Note

$a - y \Leftrightarrow -\big(y^{[i]} - \hat{y}^{[i]}\big)$

Class Label Predictions with Logistic Regression

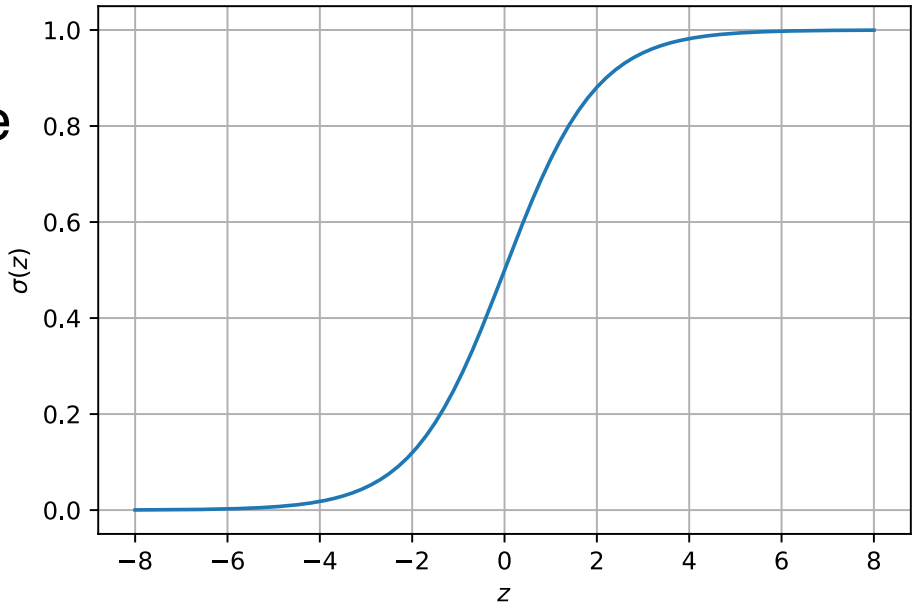


In logistic regression, we can use

$$\hat{y} := \begin{cases} 1 & \text{if } \sigma(z) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

which is the same as

$$\hat{y} := \begin{cases} 1 & \text{if } z > 0.0 \\ 0 & \text{otherwise} \end{cases}$$



About the Term "Logits"

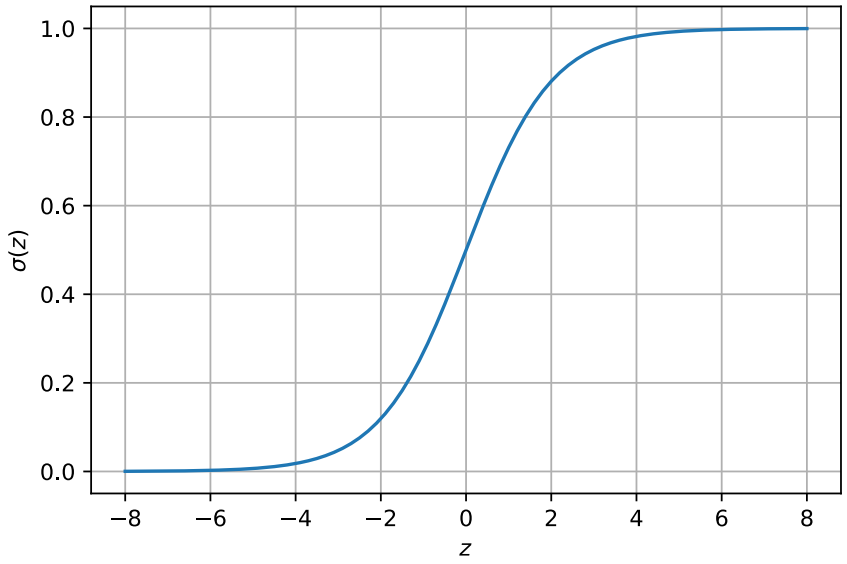
"Logits" is a very commonly used Deep Learning jargon; probably inspired by the logits for logistic regression

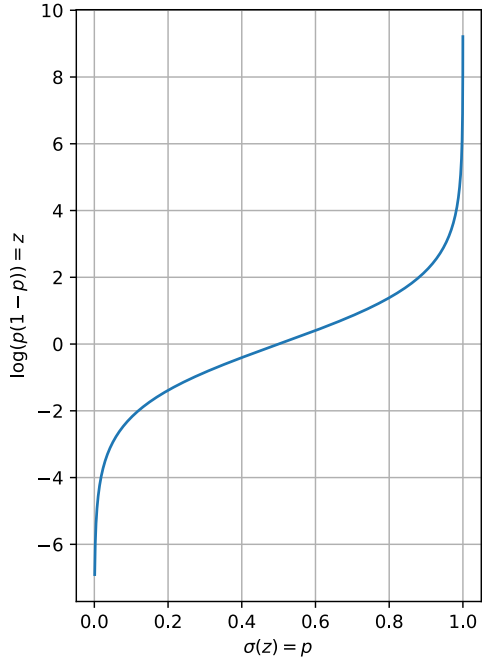
- In deep learning, the *logits* are the *net inputs* of the the last neuron layer
- In statistics, we call the *log-odds* the *logits*
- In logistic regression, the *logits* are naturally $\mathbf{w}^\top \mathbf{x}$ ...

  ... because *log-odds* is just short for "logarithm of the odds": log($\mathrm{p}/(1 - \mathrm{p})$)
- In other words, the *logits* are the inverse of the logistic sigmoid function

About the Term "Logits"

"Logits" is a very commonly used Deep Learning jargon; probably inspired by the logits for logistic regression





$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

$$logit(\sigma(z)) = \log(\sigma(z)/(1 - \sigma(z))) = z$$

About the Term "Binary Cross Entropy"

- Negative log-likelihood and binary cross entropy are equivalent
- They are just formulated in different contexts
- Cross entropy comes from the "information theory" (computer science) perspective

$$H_{\mathbf{a}}(\mathbf{y}) = -\sum_{i} \left(y^{[i]} \log(a^{[i]}) + (1 - y^{[i]}) \log(1 - a^{[i]}) \right)$$

Binary Cross Entropy

This assumes one-hot encoding where the y's are either 0 or

$$H_{\mathbf{a}}(\mathbf{y}) = \sum_{i=1}^{n} \sum_{k=1}^{K} -y_k^{[i]} \log \left(a_k^{[i]} \right)$$

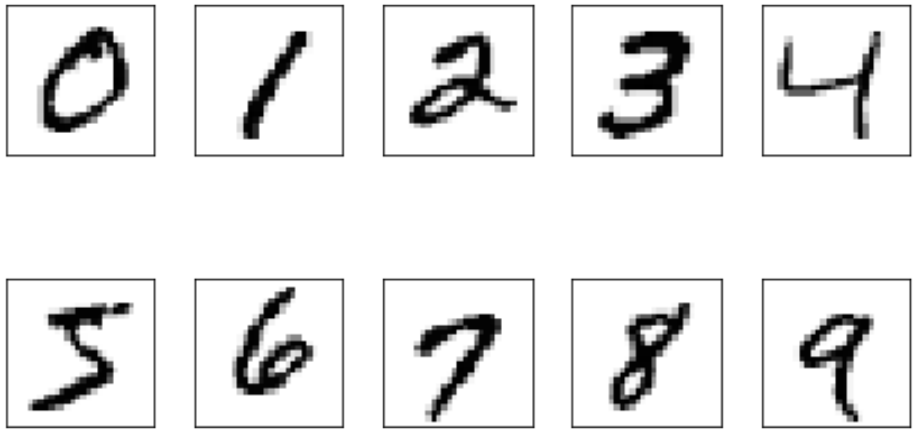
(Multi-category) Cross Entropy

for K different class labels

Generalization to Multiple classes: Softmax Regression

MNIST - 60k Handwritten Digits

http://yann.lecun.com/exdb/mnist/



Balanced dataset:

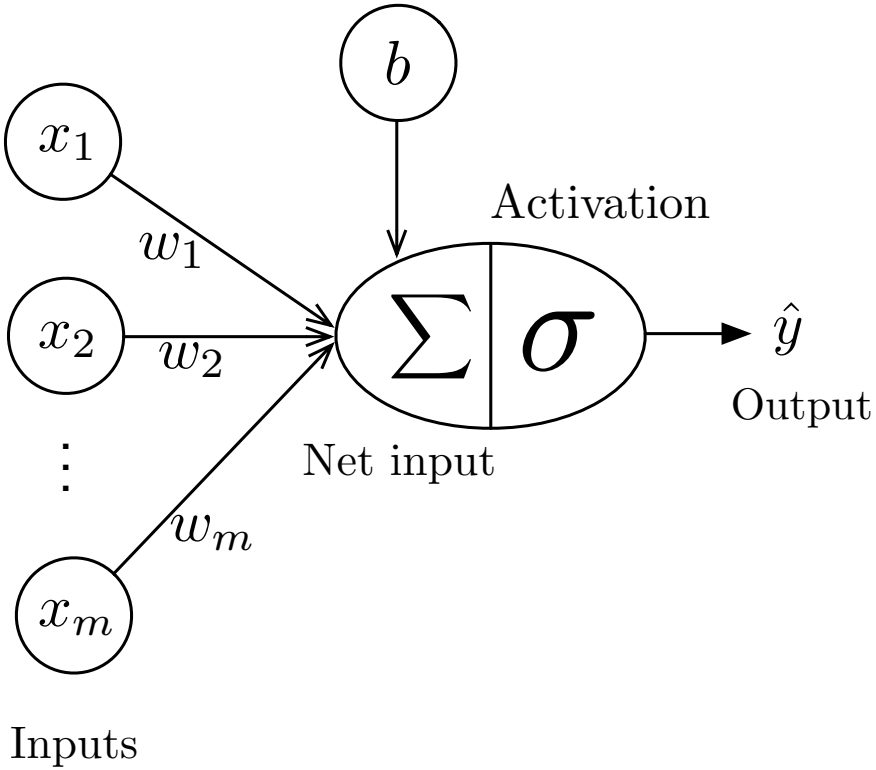
- 10 classes (digits 0-9)
- 6k digits per class

Image dimensions: 28x28x1

In NCHW, an image batch of 128 examples would be a tensor with dimensions (128, 1, 28, 28)

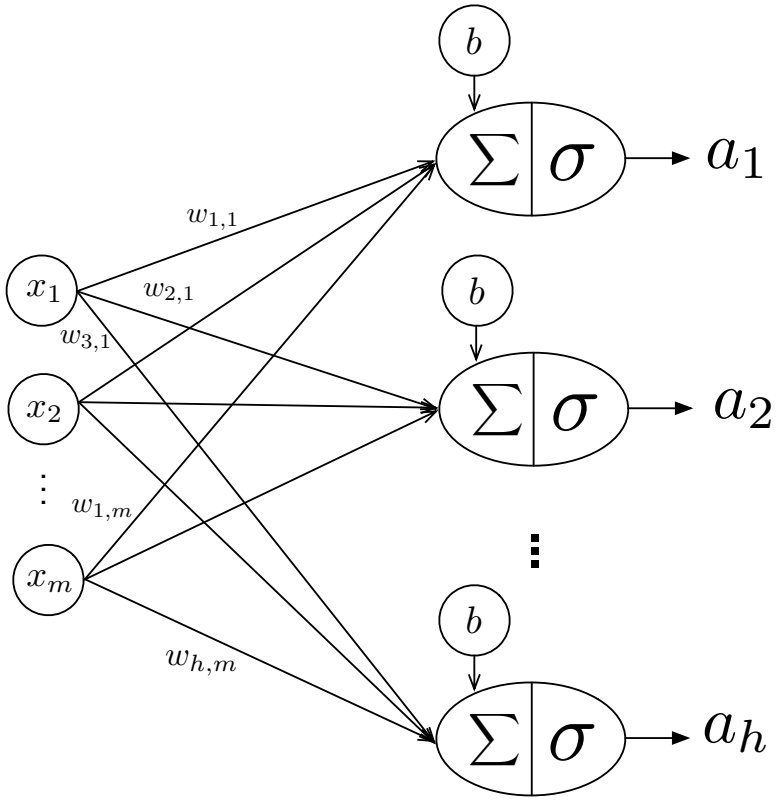
- **Training set images**: train-images-idx3-ubyte.gz (9.9 MB, 47 MB unzipped, and 60,000 examples)
- **Training set labels**: train-labels-idx1-ubyte.gz (29 KB, 60 KB unzipped, and 60,000 labels)
- **Test set images**: t10k-images-idx3-ubyte.gz (1.6 MB, 7.8 MB, unzipped and 10,000 examples)
- **Test set labels**: t10k-labels-idx1-ubyte.gz (5 KB, 10 KB unzipped, and 10,000 labels)

Previous Approach for Binary Classes



In logistic regression, the activation can be interpreted as $p(y=1/x)$

Another Approach Could be ...



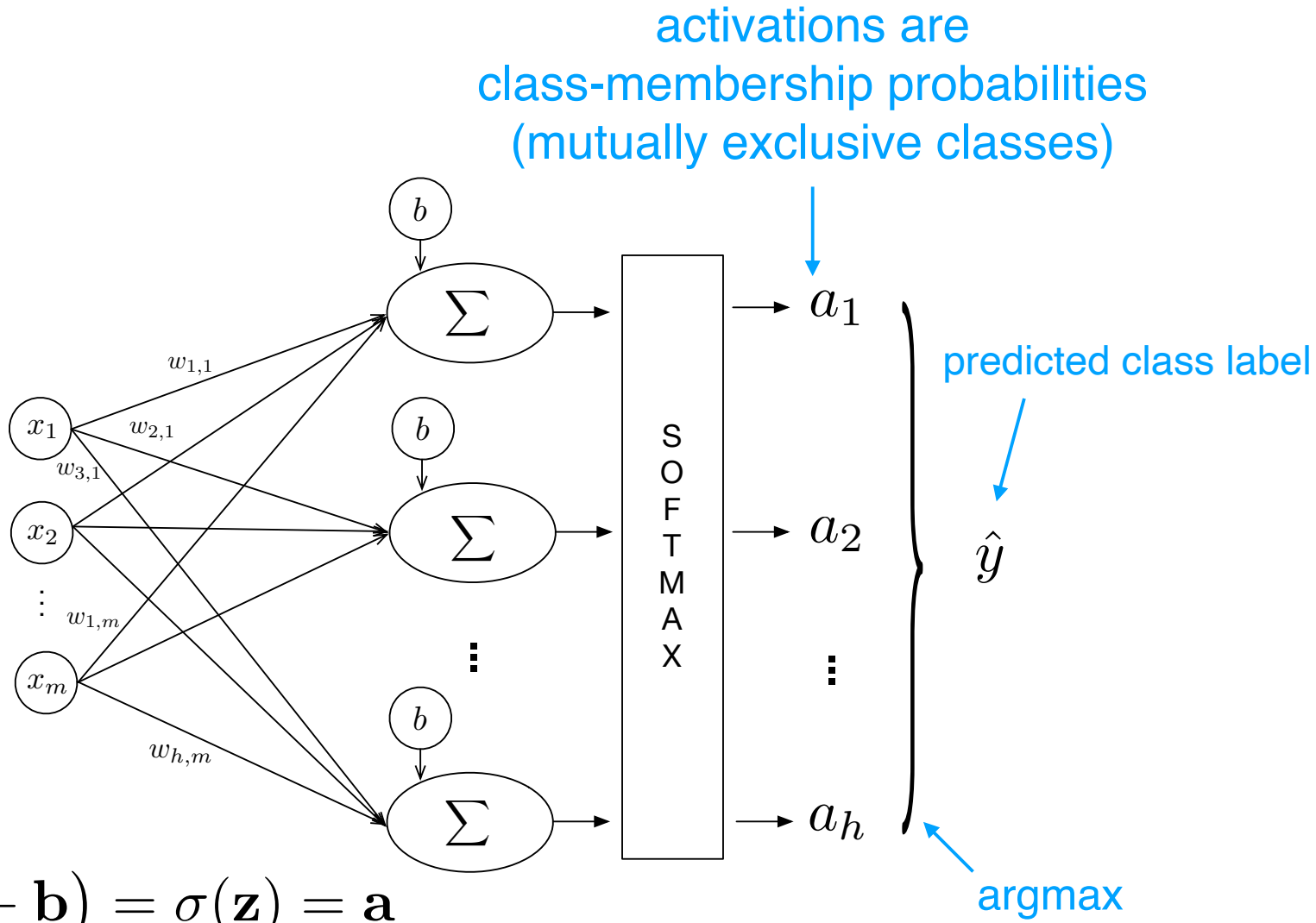
activations are class-membership probabilities (NOT mutually exclusive classes)

$$\sigma\big(\mathbf{W}\mathbf{x} + \mathbf{b}\big) = \sigma(\mathbf{z}) = \mathbf{a}$$

$$\mathbf{W} \in \mathbb{R}^{h \times m}$$

where h is the number of classes

Multinomial Logistic Regression / Softmax Regression



$$\sigma\big(\mathbf{W}\mathbf{x} + \mathbf{b}\big) = \sigma(\mathbf{z}) = \mathbf{a}$$

Softmax Activation

$$P(y = t \mid z_t^{[i]}) = \sigma_{\text{softmax}}(z_t^{[i]}) = \frac{e^{z_t^{[i]}}}{\sum_{j=1}^{h} e^{z_j^{[i]}}}$$

$t \in \{j...h\}$

h is the number of class labels

(Essentially, softmax is just an exponential function that normalizes the activations so that they sum up to 1)

Onehot Encoding Needed

class labels
0
1
3
2

class_0	class_1	class_2	class_3
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0

Loss Function

(Multi-category) Cross Entropy

for h different class labels

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{h} -y_j^{[i]} \log \left(a_j^{[i]} \right)$$

This assumes one-hot encoded labels!

Cross Entropy Loss Function Example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

Cross Entropy Loss Function Example

1 training example

$$\mathbf{Y}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

(4 training examples, 3 classes)

$$\mathcal{L} = \sum_{i=1}^{n} \sum_{j=1}^{h} -y_j^{[i]} \log\left(a_j^{[i]}\right)$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

Cross Entropy Loss Function Example

$$\mathbf{A}_{\text{onehot}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &+ [(-1) \cdot \log(0.4147)] \\ &+ [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &+ [(-0) \cdot \log(0.2248)] \\ &+ [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &+ [(-0) \cdot \log(0.2978)] \\ &+ [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

Cross Entropy Loss Function Example

$$\text{onehot} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_{\text{softmax outputs}} = \begin{bmatrix} 0.3792 & 0.3104 & 0.3104 \\ 0.3072 & 0.4147 & 0.2780 \\ 0.4263 & 0.2248 & 0.3490 \\ 0.2668 & 0.2978 & 0.4354 \end{bmatrix}$$

$$\begin{aligned} \mathcal{L}^{[1]} &= [(-1) \cdot \log(0.3792)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &+ [(-0) \cdot \log(0.3104)] \\ &= 0.969692... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[2]} &= [(-0) \cdot \log(0.3072)] \\ &+ [(-1) \cdot \log(0.4147)] \\ &+ [(-0) \cdot \log(0.2780)] \\ &= 0.880200... \end{aligned}$$

$$\begin{aligned} \mathcal{L}^{[3]} &= [(-0) \cdot \log(0.4263)] \\ &+ [(-0) \cdot \log(0.2248)] \\ &+ [(-1) \cdot \log(0.3490)] \\ &= 1.05268... \end{aligned}$$

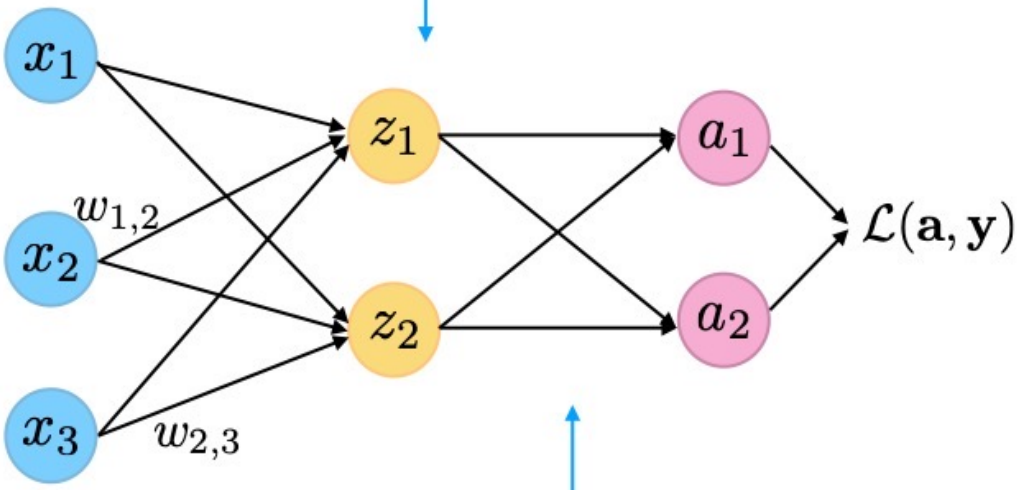
$$\begin{aligned} \mathcal{L}^{[4]} &= [(-0) \cdot \log(0.2668)] \\ &+ [(-0) \cdot \log(0.2978)] \\ &+ [(-1) \cdot \log(0.4354)] \\ &= 0.831490... \end{aligned}$$

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{h} -y_j^{[i]} \log\left(a_j^{[i]}\right)$$

$$\approx 0.9335$$

Softmax Regression Learning Rule

Note that I put the logits (net inputs) as the intermediate step



The activation function (softmax) would be applied here to obtain the activations

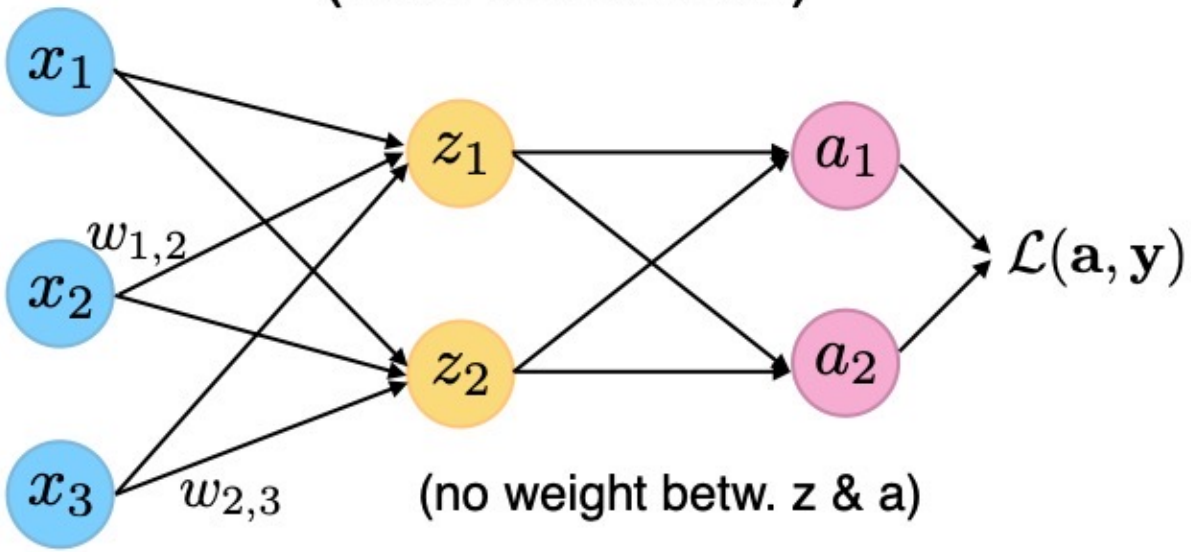
Want: $\frac{\partial \mathcal{L}}{\partial w_i}$ and $\frac{\partial \mathcal{L}}{\partial b}$

$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial a} \frac{da}{dz} \frac{\partial z}{\partial w_i}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial a} \frac{da}{dz} \frac{\partial z}{\partial b}$$

Softmax Regression Sketch

(bias not shown)



Multivariable chain rule

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$-\frac{y_1}{a_1} \quad a_1(1-a_1) \quad x_2 \qquad -\frac{y_2}{a_2} \quad -a_2 a_1 \quad x_2$$

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$-\frac{y_1}{a_1}$$

$$\frac{\partial L}{\partial a_1} = \frac{\partial}{\partial a_1}\left[\sum_{j=1}^{h} -y_j \log\left(a_j\right)\right]$$

$$= \frac{\partial}{\partial a_1}\left[-y_1 \log\left(a_1\right)\right]$$

$$= -\frac{y_1}{a_1}$$

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$\downarrow$$

$$a_1(1-a_1)$$

	Function	Derivative
Sum Rule	$f(x)+g(x)$	$f'(x)+g'(x)$
Difference Rule	$f(x)-g(x)$	$f'(x)-g'(x)$
Product Rule	$f(x)g(x)$	$f'(x)g(x)+f(x)g'(x)$
Quotient Rule	$f(x)/g(x)$	$[g(x)f'(x)-f(x)g'(x)]/[g(x)]^2$
Reciprocal Rule	$1/f(x)$	$-[f'(x)]/[f(x)]^2$
Chain Rule	$f(g(x))$	$f'(g(x))g'(x)$

$$\frac{\partial a_1}{\partial z_1} = \frac{\partial}{\partial z_1}\left[\frac{e^{z_1}}{\sum_{j=1}^{h} e^{z_j}}\right]$$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]\frac{\partial}{\partial z_1}e^{z_1} - e^{z_1}\frac{\partial}{\partial z_1}\left[\sum_{j=1}^{h} e^{z_j}\right]}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]e^{z_1} - e^{z_1}e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{e^{z_1}\left(\left[\sum_{j=1}^{h} e^{z_1}\right] - e^{z_1}\right)}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2} = \frac{e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \cdot \frac{\left[\sum_{j=1}^{h} e^{z_j}\right] - e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} = a_1(1-a_1)$$

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$\downarrow$$

$$-a_2 a_1$$

$$\frac{\partial a_2}{\partial z_1} = \frac{\partial}{\partial z_1}\left[\frac{e^{z_2}}{\sum_{j=1}^{h} e^{z_j}}\right]$$

	Function	Derivative
Sum Rule	$f(x)+g(x)$	$f'(x)+g'(x)$
Difference Rule	$f(x)-g(x)$	$f'(x)-g'(x)$
Product Rule	$f(x)g(x)$	$f'(x)g(x)+f(x)g'(x)$
Quotient Rule	$f(x)/g(x)$	$[g(x)f'(x)-f(x)g'(x)]/[g(x)]^2$
Reciprocal Rule	$1/f(x)$	$-[f'(x)]/[f(x)]^2$
Chain Rule	$f(g(x))$	$f'(g(x))g'(x)$

$$= \frac{\left[\sum_{j=1}^{h} e^{z_j}\right]\frac{\partial}{\partial z_1}e^{z_2} - e^{z_2}\frac{\partial}{\partial z_1}\left[\sum_{j=1}^{h} e^{z_j}\right]}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{0 - e^{z_2}e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]^2}$$

$$= \frac{-e^{z_2}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \cdot \frac{e^{z_1}}{\left[\sum_{j=1}^{h} e^{z_j}\right]} \quad = -a_2 a_1$$

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

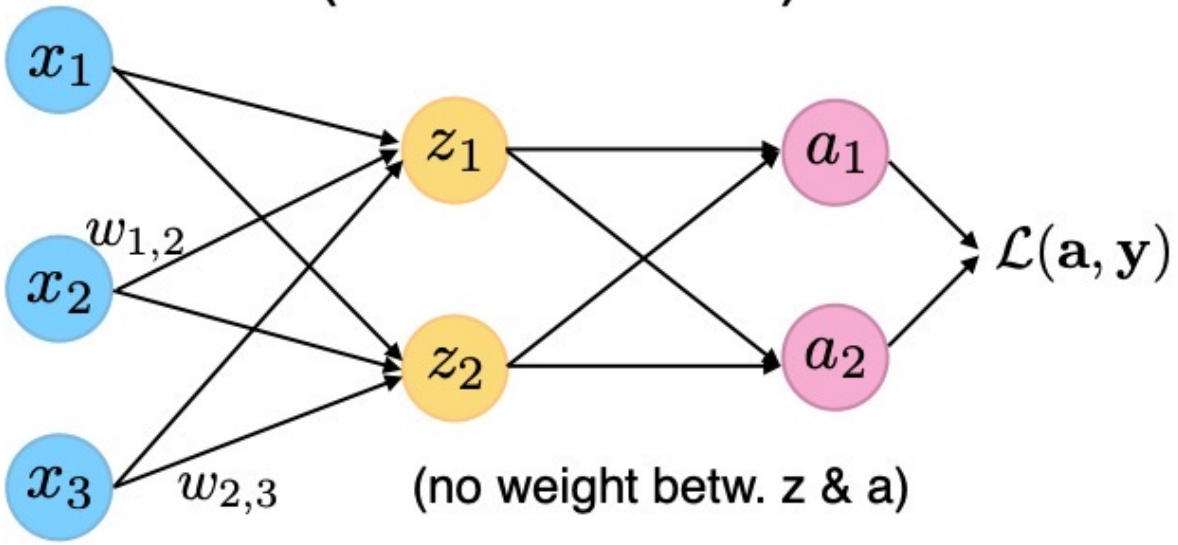
x_2

$$= \frac{\partial}{\partial w_{1,2}}\left[w_{1,2} \cdot x_2 + b\right]$$

$$= x_2$$

Softmax Regression Sketch

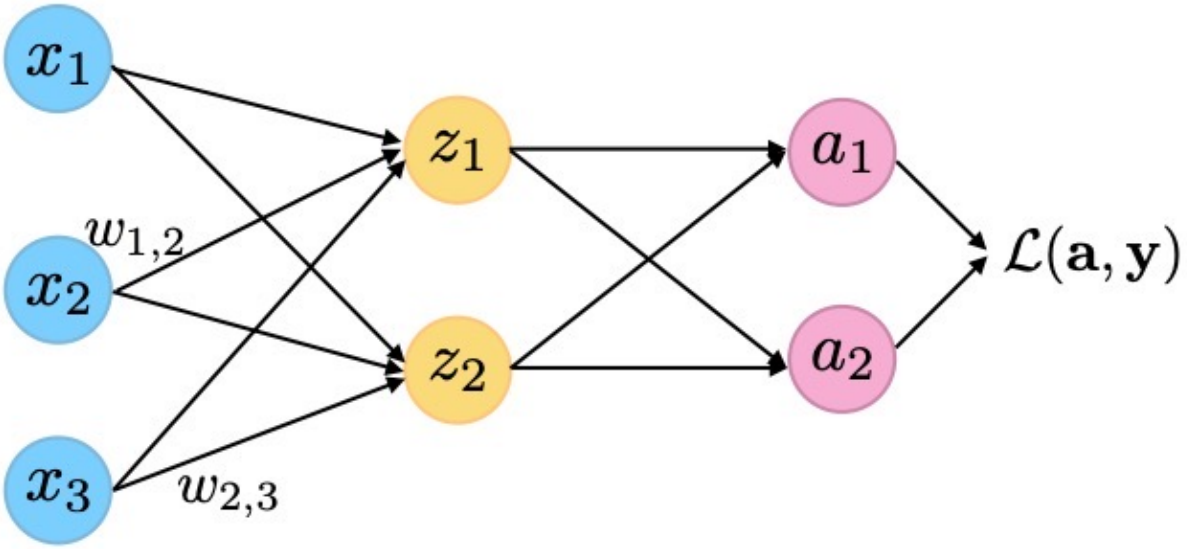
(bias not shown)



Multivariable chain rule

$$
\begin{aligned}
\frac{\partial L}{\partial w_{1,2}} &= \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} \\
&= \frac{-y_1}{a_1}[a_1(1-a_1)]x_2 + \frac{-y_2}{a_2}(-a_2 a_1)x_2 \\
&= (y_2 a_1 - y_1 + y_1 a_1)x_2 \\
&= (a_1(y_1+y_2) - y_1)x_2 \\
&= -(y_1 - a_1)x_2
\end{aligned}
$$

Softmax Regression Sketch



Vectorized Form:

$$\frac{\partial L}{\partial w_{1,2}} = \frac{\partial L}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}} + \frac{\partial L}{\partial a_2}\frac{\partial a_2}{\partial z_1}\frac{\partial z_1}{\partial w_{1,2}}$$

$$= \frac{-y_1}{a_1}[a_1(1-a_1)]x_2 + \frac{-y_2}{a_2}(-a_2 a_1)x_2$$

$$= (y_2 a_1 - y_1 + y_1 a_1)x_2$$

$$= (a_1(y_1 + y_2) - y_1)x_2$$

$$= -(y_1 - a_1)x_2$$

$$\nabla_{\mathbf{W}}\mathcal{L} = -(\mathbf{X}^\top(\mathbf{Y} - \mathbf{A}))^\top$$

where $\mathbf{W} \in \mathbb{R}^{k \times m}$

$\mathbf{X} \in \mathbb{R}^{n \times m}$

$\mathbf{A} \in \mathbb{R}^{n \times k}$

$\mathbf{Y} \in \mathbb{R}^{n \times k}$