Lecture 08

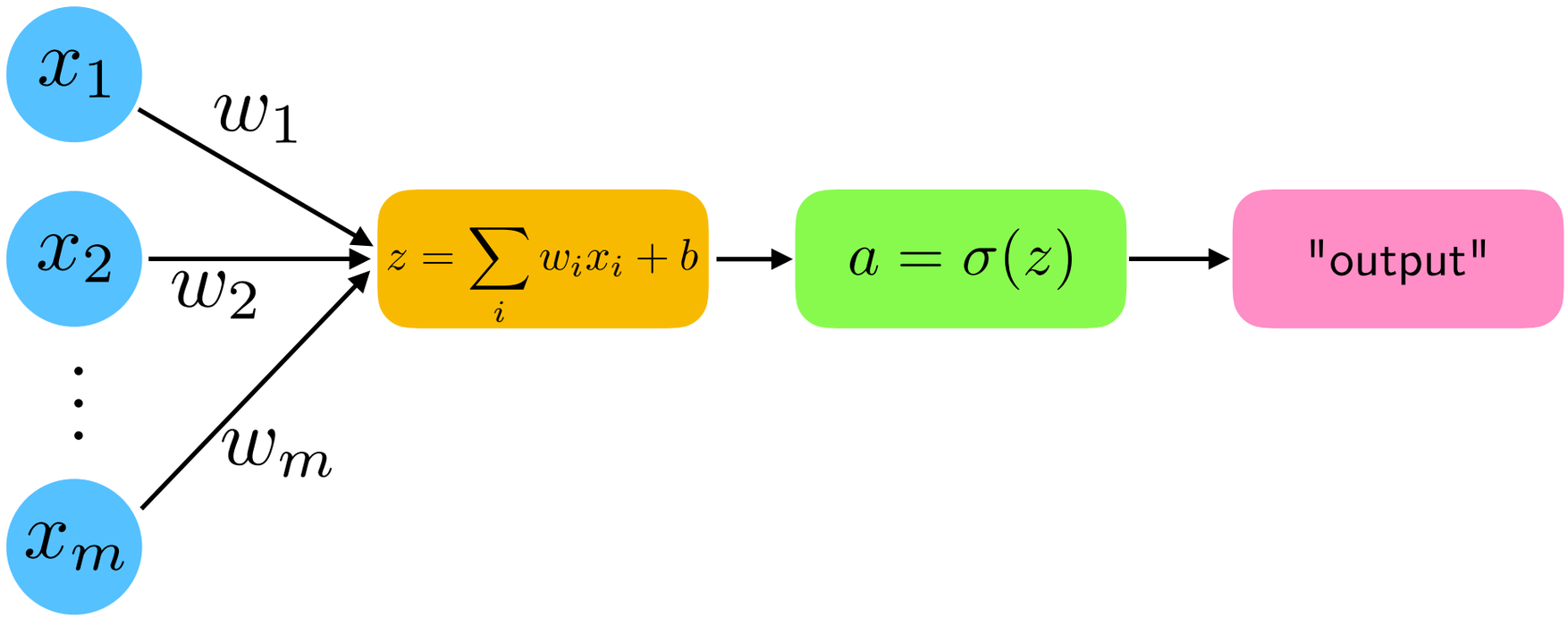
Logistic Regression and Multi-class Classification

STAT 479: Deep Learning, Spring 2019

Sebastian Raschka

http://stat.wisc.edu/~sraschka/teaching/stat479-ss2019/

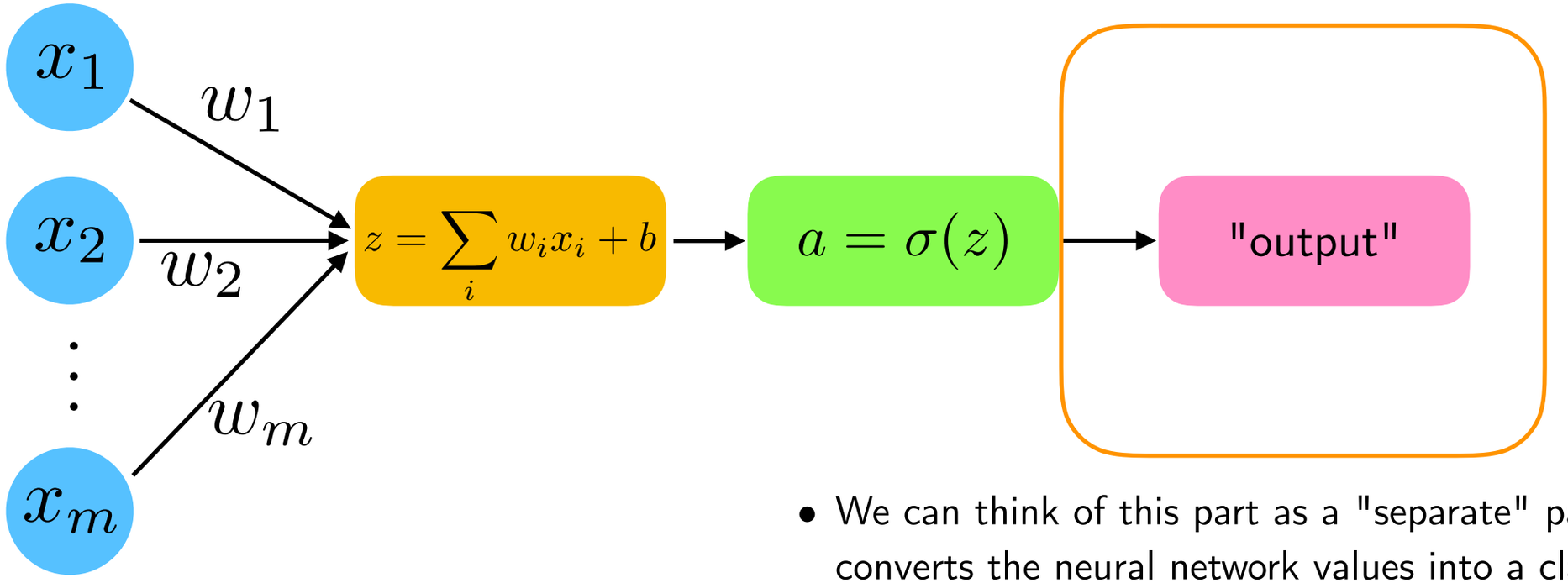
Homework 2 Recap



Generic Neuron representation

(Perceptron, ADALINE, Linear Regression, and later today: Logistic Regression)

Homework 2 Recap

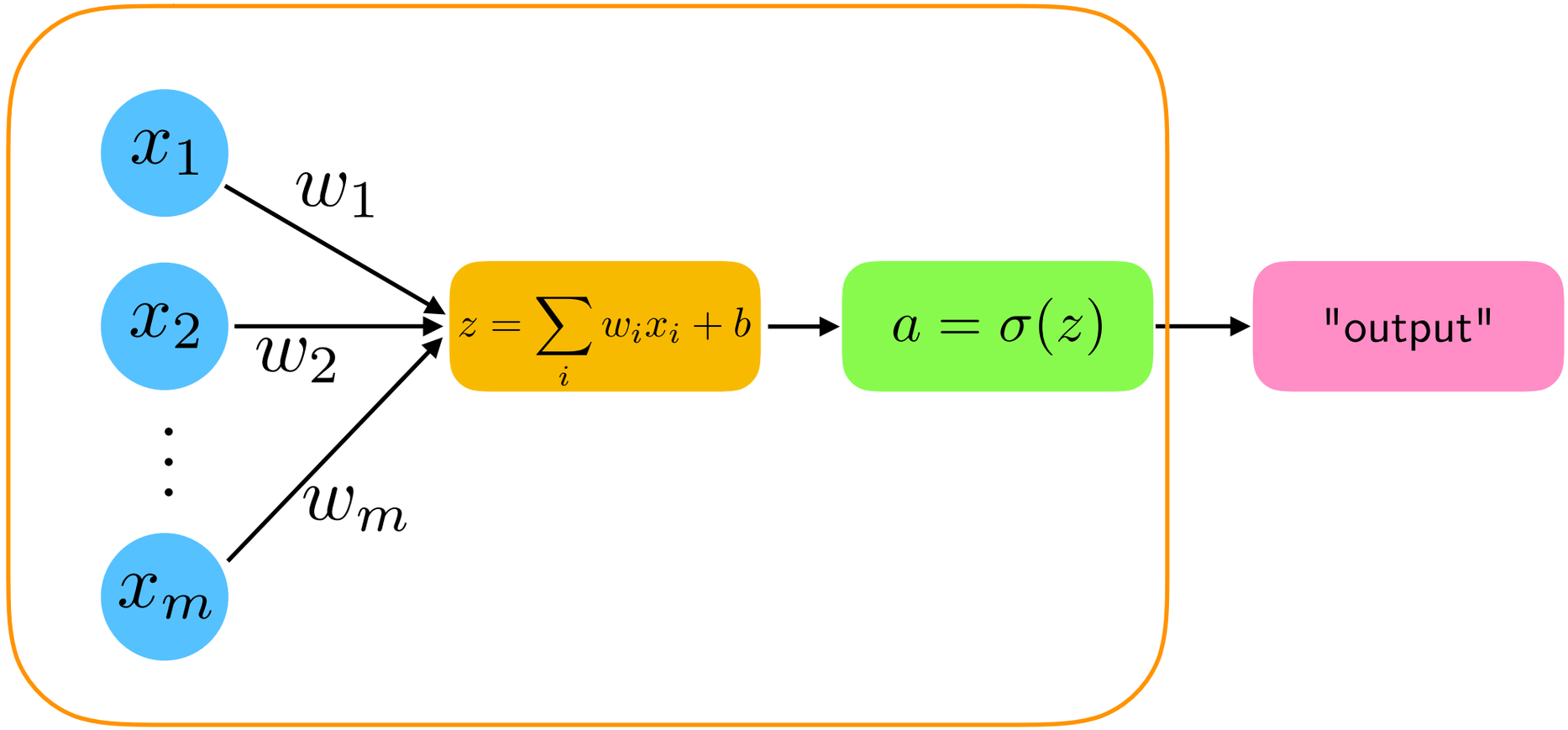


- We can think of this part as a "separate" part that converts the neural network values into a class label, for example; e.g., via a threshold function

- Predicted class labels are not used during training (except by the Perceptron)

- ADALINE, Logistic Regression (later today), and all common types of multi-layer neural networks don't use predicted class labels for optimization as a threshold function is not smooth

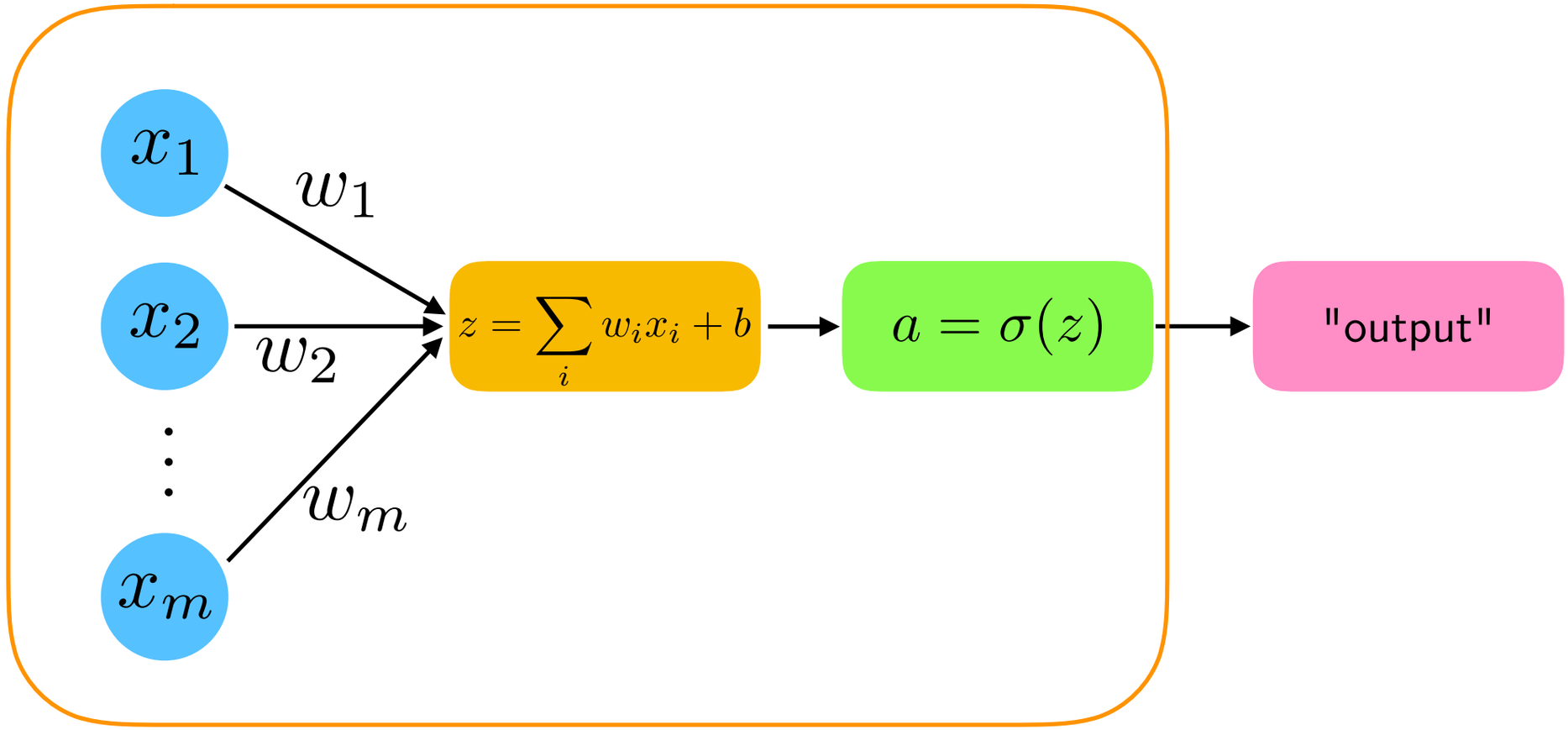
Homework 2 Recap



- For training, only this part matters!

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

Homework 2 Recap

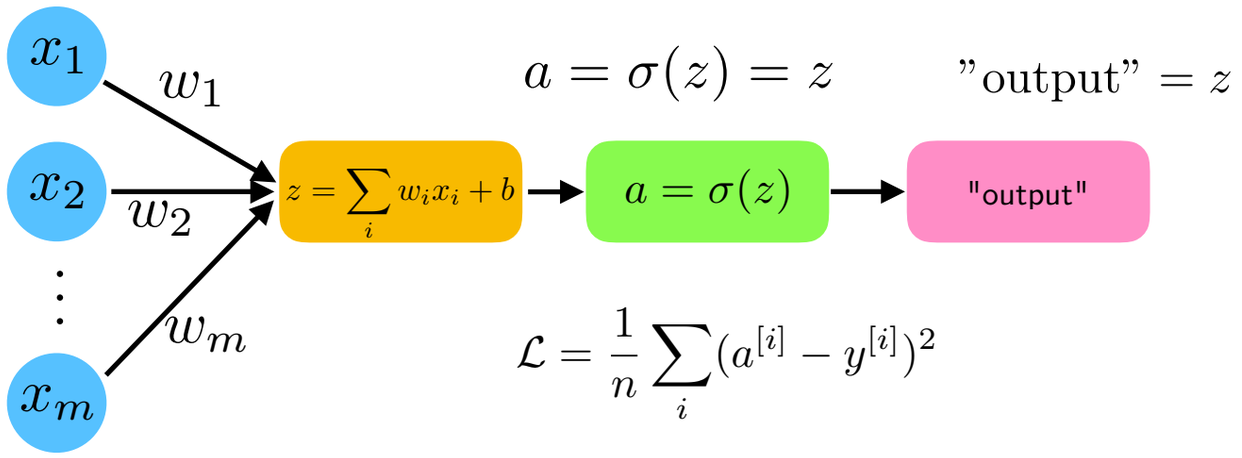


- For training, only this part matters!

$$\mathcal{L} = \frac{1}{n} \sum_i (a^{[i]} - y^{[i]})^2 \quad \text{Want:} \quad \frac{\partial \mathcal{L}}{\partial w_i} \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial b}$$

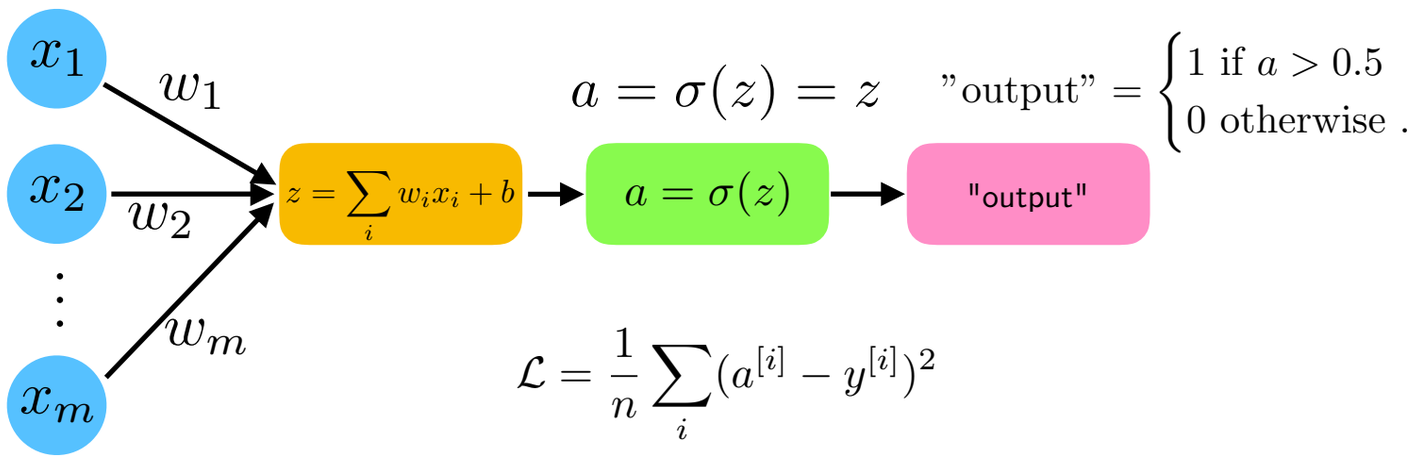
Homework 2 Recap

a) Linear regression



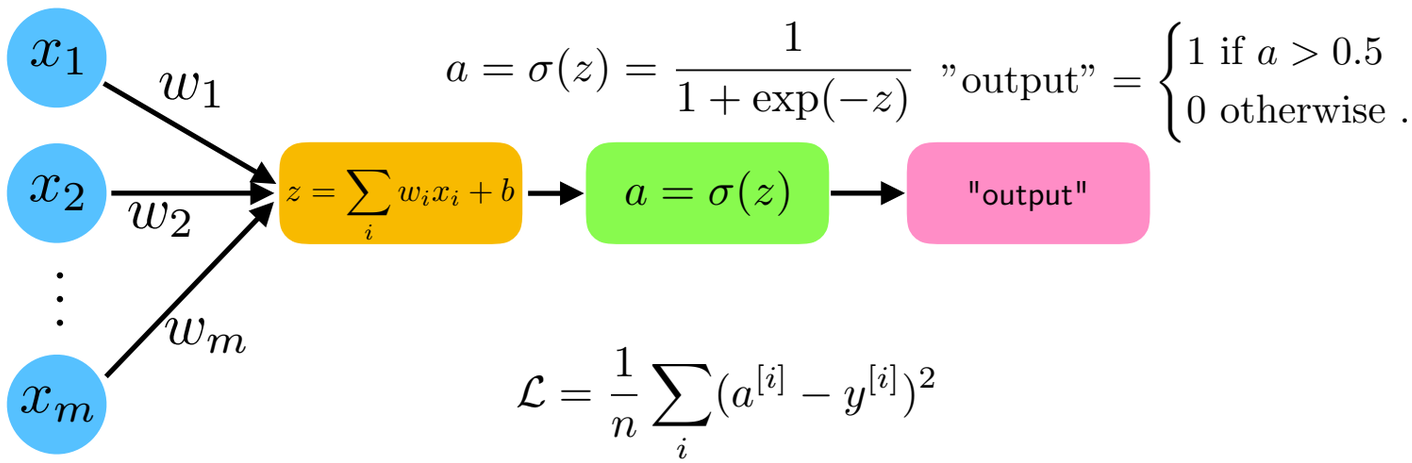
---

b) ADALINE



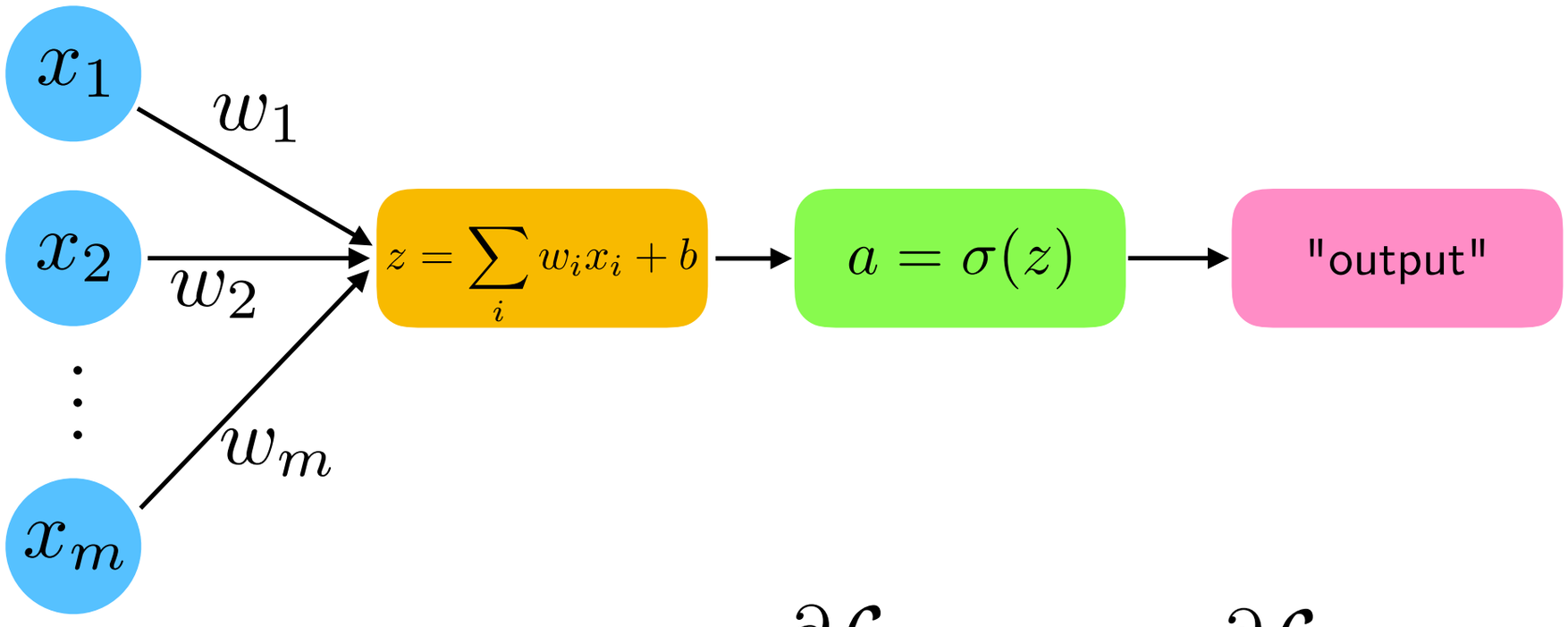
---

c) HW2 Neuron



Homework 2 Recap

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

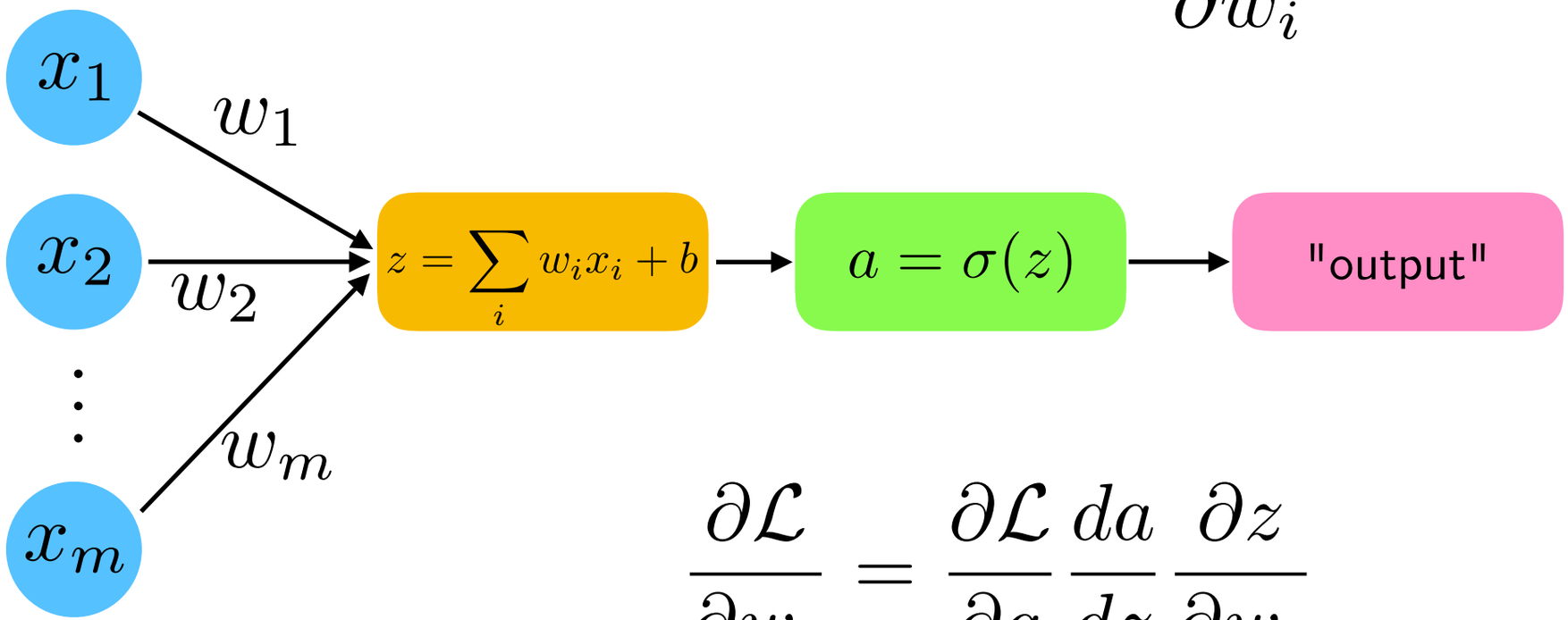


Want: $\frac{\partial \mathcal{L}}{\partial w_i}$ and $\frac{\partial \mathcal{L}}{\partial b}$

Homework 2 Recap

$\mathcal{L} = (a^{[i]} - y^{[i]})^2$ For simplicity, suppose we have only 1 training example

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



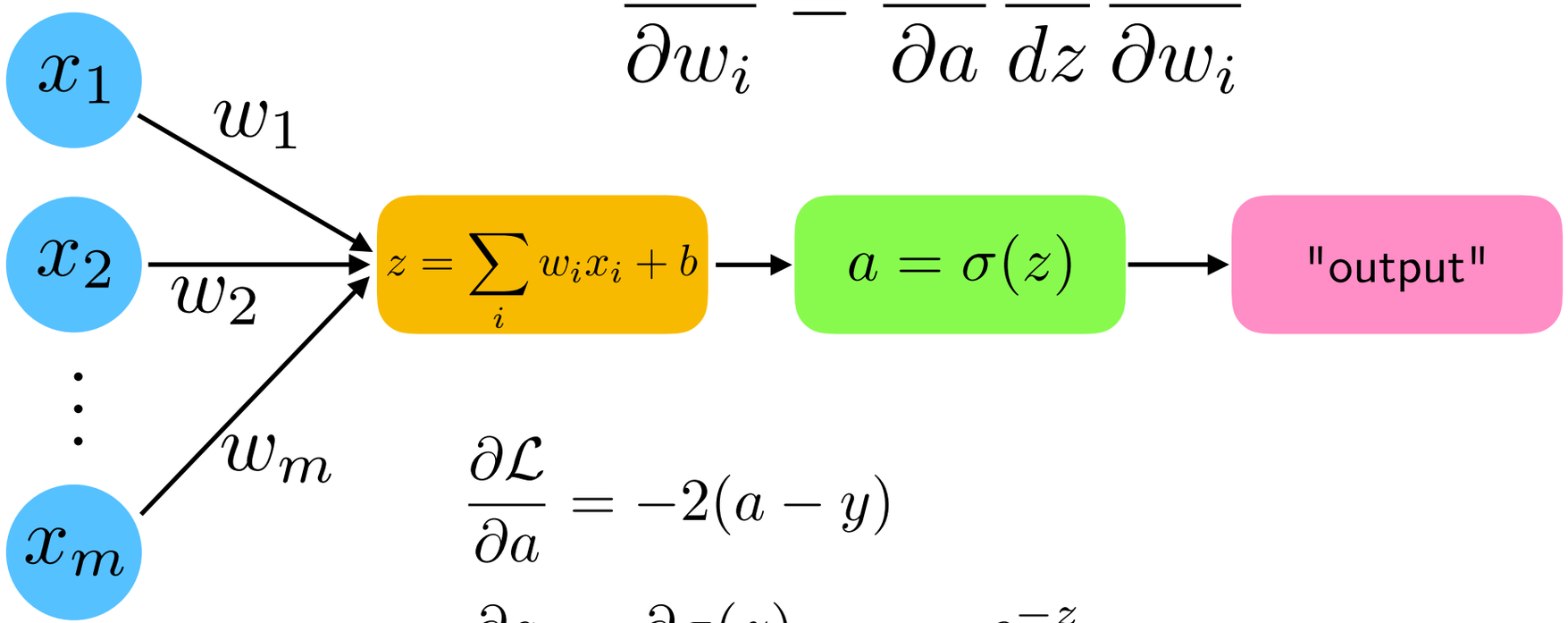
$$\frac{\partial \mathcal{L}}{\partial w_i} = \frac{\partial \mathcal{L}}{\partial a} \frac{da}{dz} \frac{\partial z}{\partial w_i}$$

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

Homework 2 Recap

$\mathcal{L} = (a^{[i]} - y^{[i]})^2$ For simplicity, suppose we have only 1 training example

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



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

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

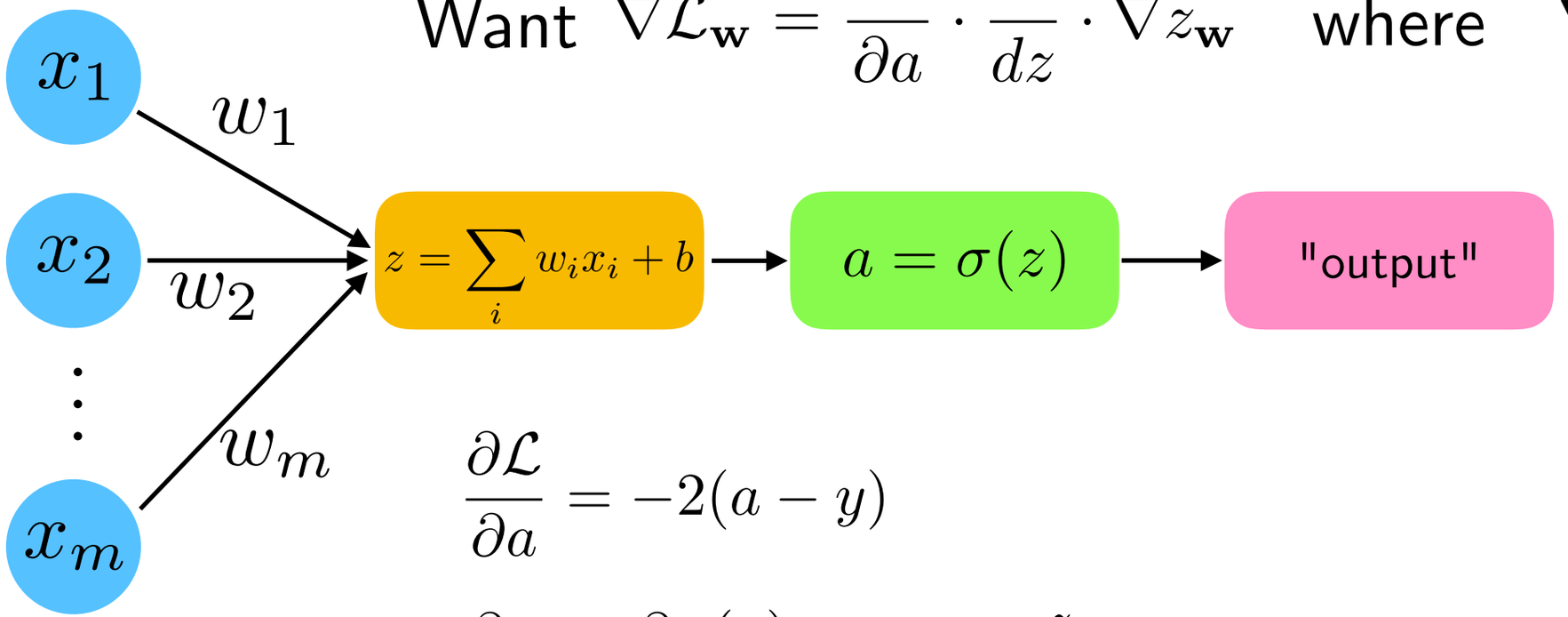
$$\frac{\partial z}{\partial w_i} = x_i$$

Homework 2 Recap

$\mathcal{L} = (a^{[i]} - y^{[i]})^2$ For simplicity, suppose we have only 1 training example

but multiple weights

Want $\nabla \mathcal{L}_{\mathbf{w}} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{da}{dz} \cdot \nabla z_{\mathbf{w}}$ where $\nabla \mathcal{L}_{\mathbf{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{w_1} \\ \frac{\partial \mathcal{L}}{w_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{w_i} \end{bmatrix}$



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

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

$$\nabla_{\mathbf{w}} z = [x_1 \quad x_2 \quad \dots \quad x_m]^\top$$

Homework 2 Recap

Multiple training examples:

$$\mathcal{L} = (a^{[i]} - y^{[i]})^2 \longrightarrow \mathcal{L} = \frac{1}{n}\sum_i (a^{[i]} - y^{[i]})^2$$

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

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n}\sum_{i=1}^{n} -2(a^{[i]} - y^{[i]}) \cdot \sigma(z^{[i]}) \cdot (1 - \sigma(z^{[i]})) \cdot x_j^{[i]})$$

Only this part is new (compared to 1 training example)!

Homework 2 Recap

Multiple training examples:

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

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n} \sum_{i=1}^{n} -2(a^{[i]} - y^{[i]}) \cdot \sigma(z^{[i]}) \cdot (1 - \sigma(z^{[i]})) \cdot x_j^{[i]})$$

Vectorized:

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{2}{n}(\mathbf{a} - \mathbf{y}) \odot \sigma(\mathbf{z}) \odot (1 - \sigma(\mathbf{z}))\mathbf{x}_j^\top$$

Homework 2 Recap

Multiple training examples:

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

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{n} \sum_{i=1}^{n} -2(a^{[i]} - y^{[i]}) \cdot \sigma(z^{[i]}) \cdot (1 - \sigma(z^{[i]})) \cdot x_j^{[i]})$$

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{2}{n}(\mathbf{a} - \mathbf{y}) \odot \sigma(\mathbf{z}) \odot (1 - \sigma(\mathbf{z}))\mathbf{x}_j^\top$$

Multiple training examples and features:

$$\nabla_{\mathbf{w}} \mathcal{L} = \Big(-\frac{2}{n}(\mathbf{a} - \mathbf{y}) \odot \sigma(\mathbf{z}) \odot (1 - \sigma(\mathbf{z}))^\top \mathbf{X} \Big)^\top \text{ where } \mathbf{x} \in \mathbb{R}^{m \times 1}$$

$$\nabla_{\mathbf{w}} \mathcal{L} = \mathbf{X}^\top \Big(-\frac{2}{n}(\mathbf{a} - \mathbf{y}) \odot \sigma(\mathbf{z}) \odot (1 - \sigma(\mathbf{z})) \Big) \text{ where } \mathbf{X} \in \mathbb{R}^{n \times m}$$

Homework 2 Recap

$$\mathcal{L} := 0, \quad w_1 := 0, \quad w_2 := 0, \quad b := 0$$

for i =1 to i=n:

$$z^{[i]} = w^T x^{[i]} + b$$

$$a^{[i]} = \sigma\left(z^{[i]}\right)$$

$$\mathcal{L} \pm (y^{[i]} - a^{[i]})^2$$

$$\frac{\partial \mathcal{L}}{\partial z^{[i]}} = -2(a^{[i]} - y^{[i]}) \cdot \sigma(z^{[i]})(1 - \sigma(z^{[i]})$$

$$\frac{\partial \mathcal{L}}{\partial w_1} \pm x_1^{[i]} \frac{\partial \mathcal{L}}{\partial z^{[i]}}$$

$$\frac{\partial \mathcal{L}}{\partial w_2} \pm x_2^{[i]} \frac{\partial \mathcal{L}}{\partial z^{[i]}}$$

$$\frac{\partial \mathcal{L}}{\partial b} \pm \frac{\partial \mathcal{L}}{\partial z^{[i]}}$$

$$\mathcal{L} := \mathcal{L}/n$$

$$\frac{\partial \mathcal{L}}{\partial w_1} := \frac{\partial \mathcal{L}}{\partial w_1}/n, \quad \frac{\partial \mathcal{L}}{\partial w_2} := \frac{\partial \mathcal{L}}{\partial w_2}/n, \quad \frac{\partial \mathcal{L}}{\partial b} := \frac{\partial \mathcal{L}}{\partial b}/n$$

If still confused, think of the previous slide as a for-loop;
we simply "vectorized" it :)

Homework 2 Recap

Lastly, about the sigmoid derivative ...

$$
\begin{aligned}
\frac{d}{dz}\sigma(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{d}{dz}\left(1+e^{-z}\right)^{-1} \quad \text{[apply chain rule]} \\
&= -(1+e^{-z})^{-2} \cdot \frac{d}{dz}(1+e^{-z}) \quad \text{[apply sum rule]} \\
&= -(1+e^{-z})^{-2} \cdot \left(\frac{d}{dz}1 + \frac{d}{dz}e^{-z}\right) \\
&= -(1+e^{-z})^{-2} \cdot \frac{d}{dz}e^{-z} \quad \text{[apply chain rule]} \\
&= -(1+e^{-z})^{-2} \cdot e^{-z}\frac{d}{dz}(-z) \\
&= -(1+e^{-z})^{-2} \cdot \left(-e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})^2} \cdot e^{-z} \\
&= \frac{e^{-z}}{(1+e^{-z})^2}
\end{aligned}
$$

Homework 2 Recap

Simplifying the sigmoid derivative ...

$$\begin{aligned}
\frac{e^{-z}}{(1+e^{-z})^2} &= \frac{e^{-z}}{1+e^{-z}} \cdot \frac{1}{1+e^{-z}} \\
&= \frac{-1+1+e^{-z}}{1+e^{-z}} \cdot \frac{1}{1+e^{-z}} \\
&= \left(\frac{-1}{1+e^{-z}} + \frac{1+e^{-z}}{1+e^{-z}}\right) \cdot \frac{1}{1+e^{-z}} \\
&= \left(\frac{-1}{1+e^{-z}} + 1\right) \cdot \frac{1}{1+e^{-z}} \\
&= \left(1 - \frac{1}{1+e^{-z}}\right) \cdot \frac{1}{1+e^{-z}} \\
&= \big(1-\sigma(z)\big) \cdot \sigma(z)
\end{aligned}$$

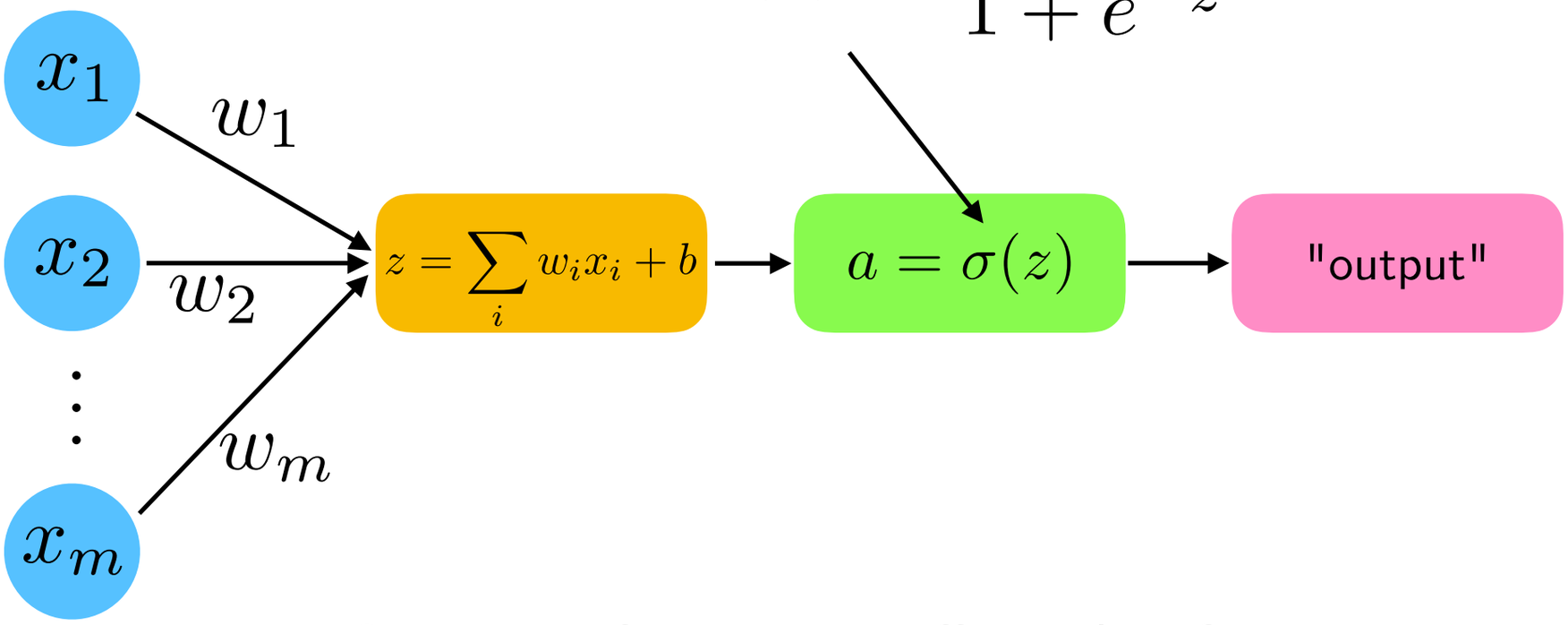
$$\frac{d\sigma(z)}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = \sigma(z) \cdot (1-\sigma(z))$$

Logistic Regression for Binary Classification

Logistic Regression Neuron

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

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

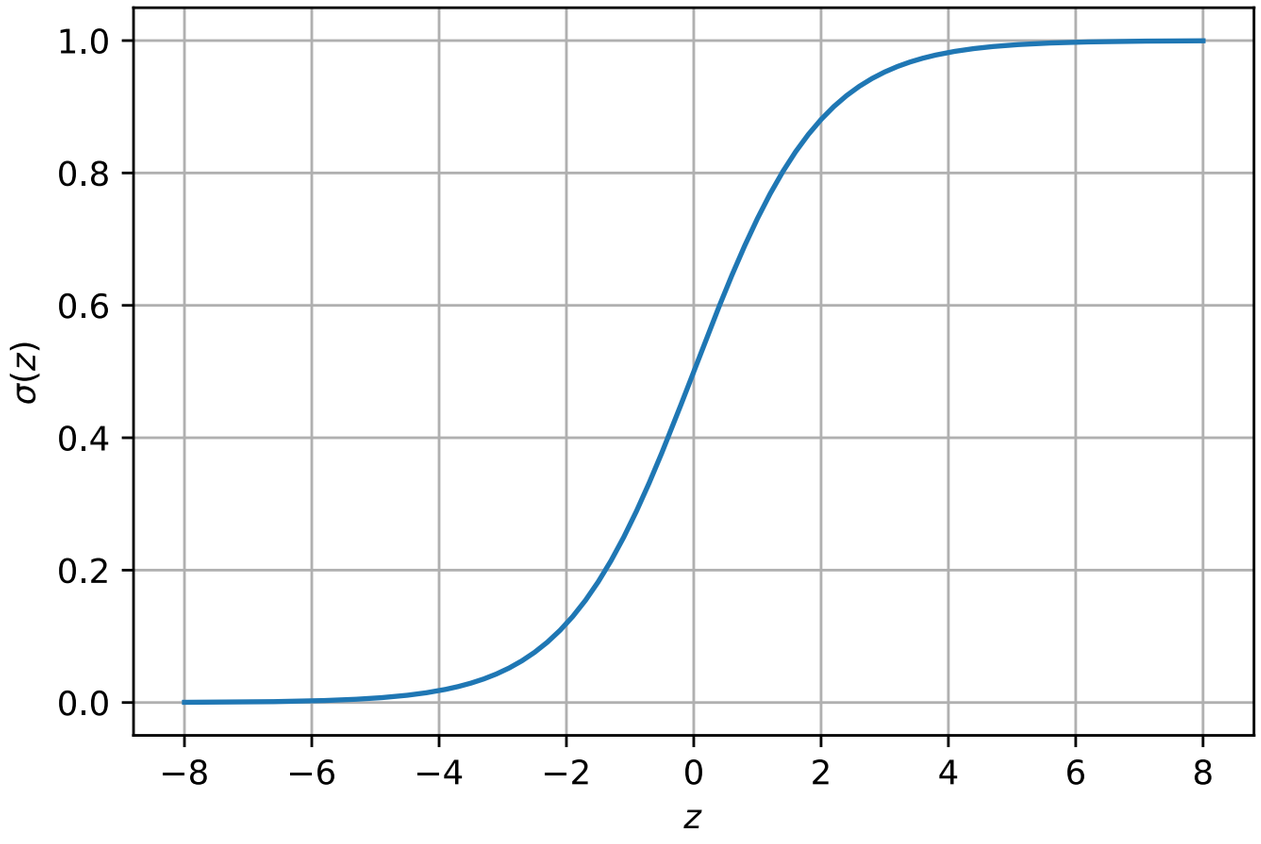


- Same as before, except different loss function
- in ADALINE and HW, we minimized the MSE loss:

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

Logistic Sigmoid Function

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



Logistic Regression

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

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

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

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

Logistic Regression

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

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

here, hypothesis is just our activation function output

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

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

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

Logistic Regression

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

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

rewrite more compactly

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

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

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

Logistic Regression

And for multiple training examples, we want to maximize:

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

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

Likelihood "Loss"

$$L(\mathbf{w}) = P(\mathbf{y} \mid \mathbf{x}; \mathbf{w})$$

$$= \prod_{i=1}^{n} P\big(y^{(i)} \mid x^{(i)}; \mathbf{w}\big)$$

$$= \prod_{i=1}^{n} \Big(\sigma\big(z^{(i)}\big)\Big)^{y^{(i)}} \Big(1 - \sigma\big(z^{(i)}\big)\Big)^{1-y^{(i)}}$$

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 and HW2, we minimized the MSE loss:

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

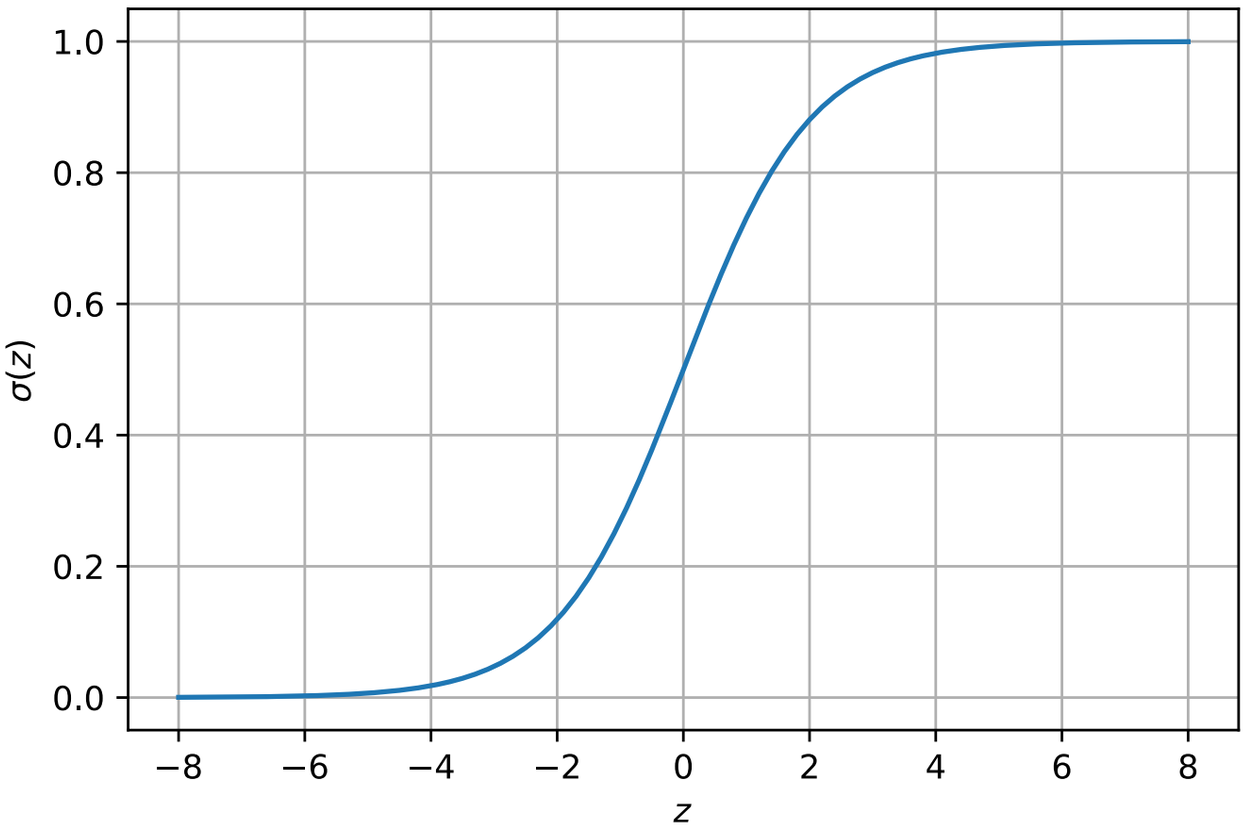
- However, the difference is that in Logistic Regression, we maximize the likelihood
- Maximizing likelihood is the same as maximizing the log-likelihood, but the latter is numerically more stable
- Maximizing the log-likelihood is the same as minimizing the negative log-likelihood, which is convenient,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

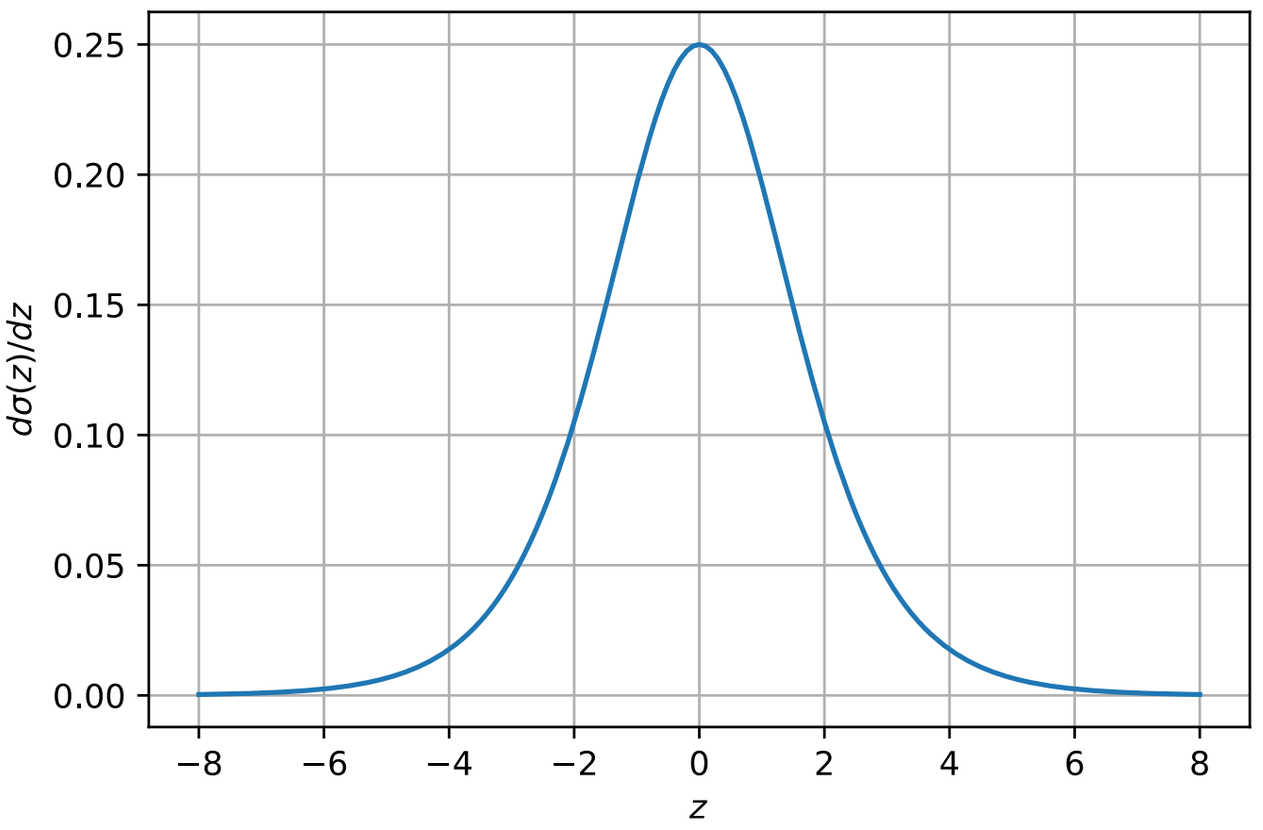
Logistic Sigmoid Derivative

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

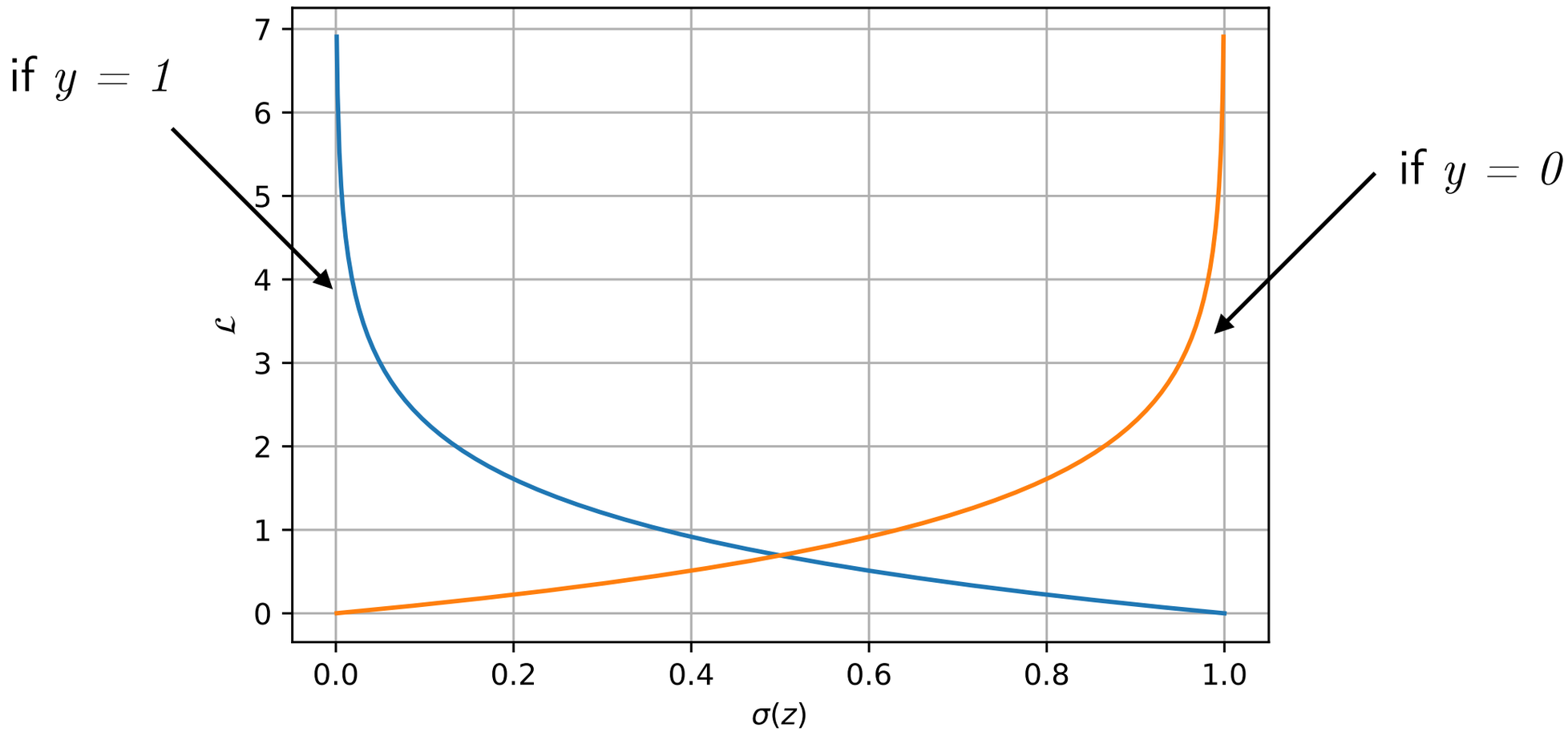
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

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





Loss for a Single Training Example



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

Learning Rule

Same gradient descent rule as before, for which we need

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

$$\frac{\partial \mathcal{L}}{\partial a} = -\left(y \frac{1}{a} - (1-y) \frac{1}{1-a} \right)$$

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

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

Learning Rule

Same gradient descent rule as before, for which we need

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

$$\frac{\partial \mathcal{L}}{\partial a} = -\left(y \frac{1}{a} - (1-y) \frac{1}{1-a} \right)$$

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

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

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

Learning Rule for Logistic Regression

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

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

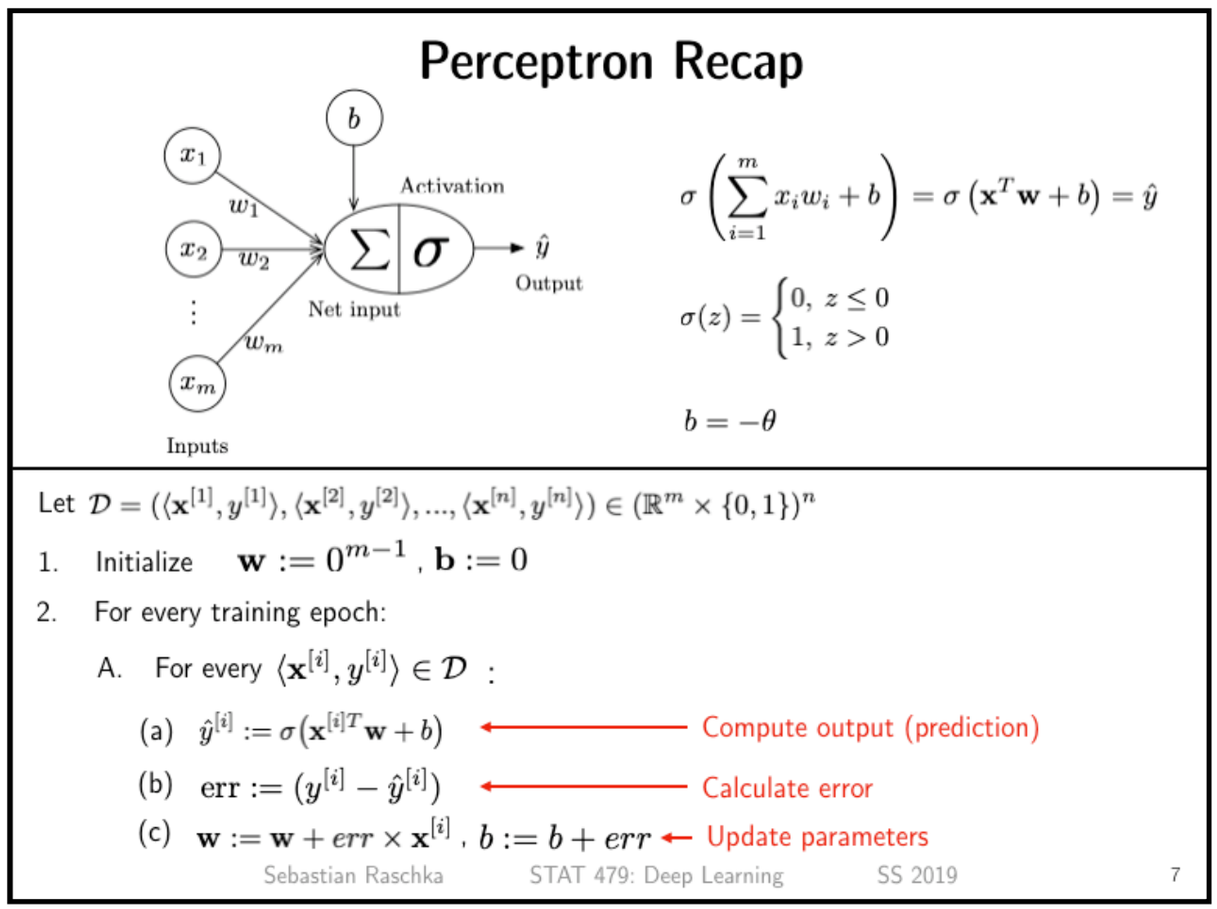
$$\frac{\partial \mathcal{L}}{\partial a} = -\left(y \frac{1}{a} - (1-y) \frac{1}{1-a} \right)$$

$$\frac{da}{dz} = \frac{e^{-z}}{(1+e^{-z})^2} = a \cdot (1-a) \longrightarrow \frac{\partial \mathcal{L}}{\partial z} = a - y \longrightarrow \frac{\partial \mathcal{L}}{\partial w_j} = (a-y)x_j$$

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

Learning Rule for Logistic Regression

Remember the Perceptron & ADALINE Lectures?



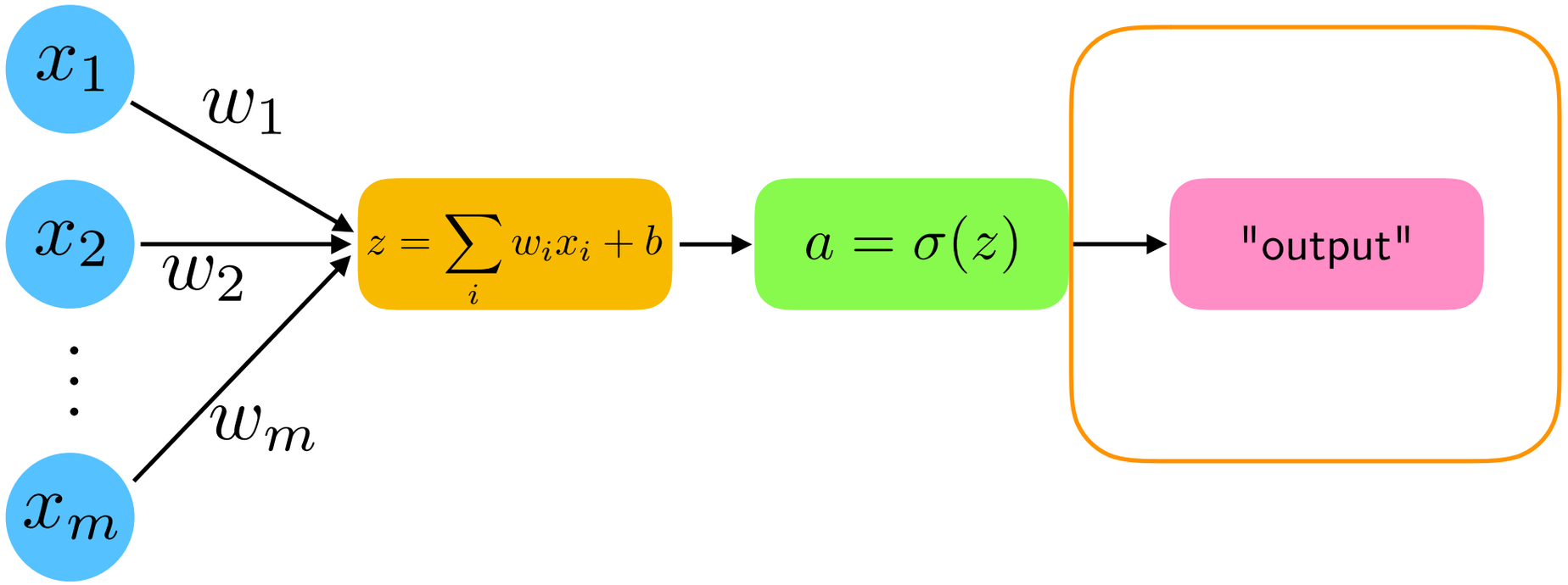
$$\frac{\partial \mathcal{L}}{\partial w_j} = (a - y)x_j$$

$$\begin{aligned}\Delta w :=& \text{ "negative gradient"} \\ =& -\frac{\partial \mathcal{L}}{\partial w_j} \\ =& (y - a)x_j\end{aligned}$$

$$w := \eta \cdot \Delta w$$

Update for each batch/minibatch with learning rate

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

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

- We can think of this part as a "separate" part that converts the neural network values into a class label, for example; e.g., via a threshold function
- Predicted class labels are not used during training (except by the Perceptron)
- ADALINE, Logistic Regressio, and all common types of multi-layer neural networks don't use predicted class labels for optimization as a threshold function is not smooth

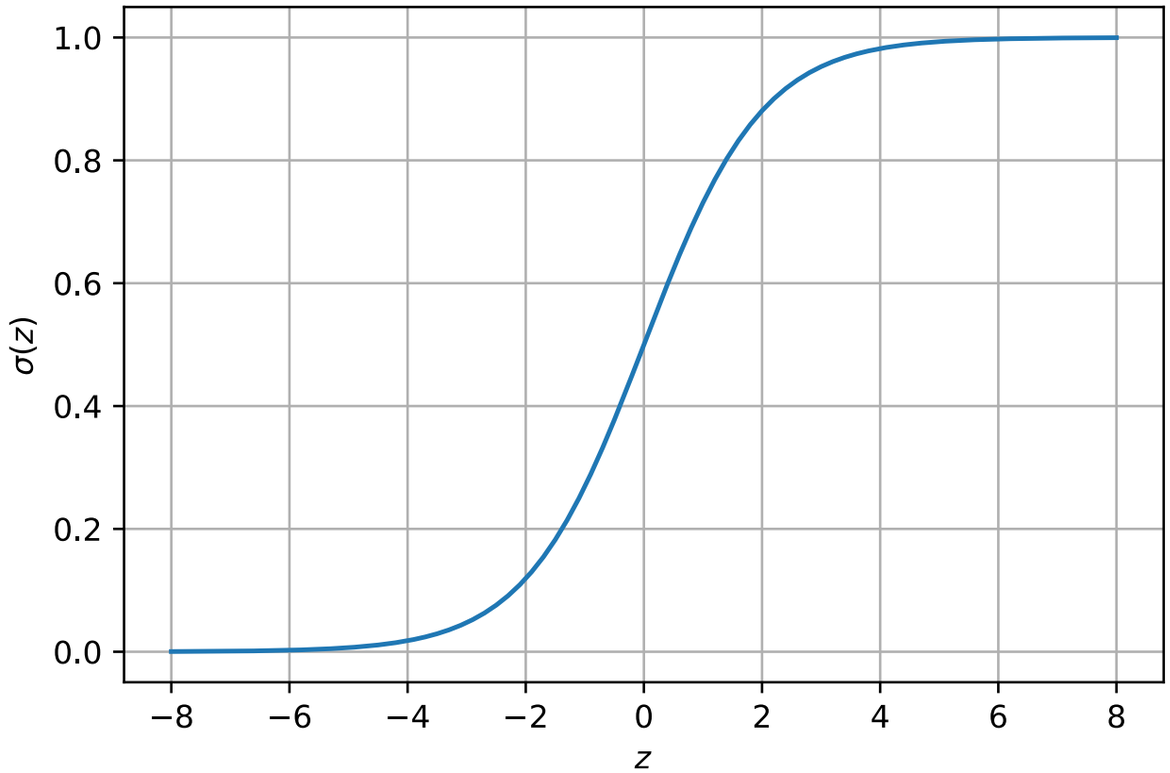
About the Term "Logits"

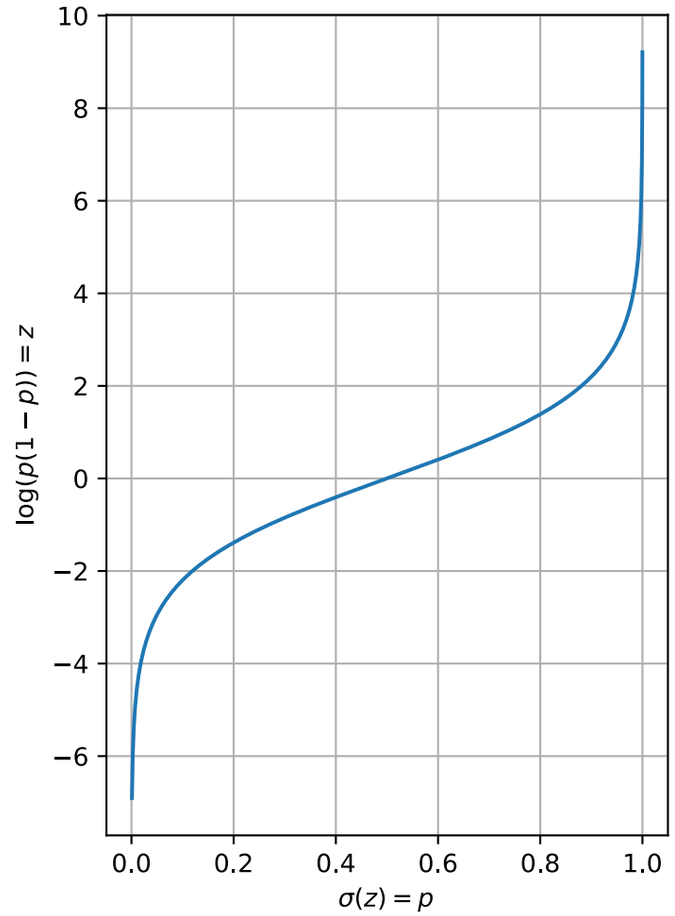
"Logits" is a very commonly used Deep Learning jargon;
probably inspired by the logits for logistic regression

- 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
- For those who took my previous STAT479: Machine Learning class, we have seen regular entropy in the context of decision trees (but with $\log_2$ instead of the natural log)

https://github.com/rasbt/stat479-machine-learning-fs18/blob/master/06_trees/06_trees_notes.pdf

$$H_{\mathbf{a}}(\mathbf{y}) = -\sum_i \Big(a^{[i]} \log(y^{[i]}) + (1 - a^{[i]}) \log(1 - y^{[i]}) \Big)$$

Binary Cross Entropy

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

(Multi-category) Cross Entropy
for K different class labels

To be continued ...

... Code example will be added before next lecture

Logistic Regression for Multi-class Classification:

Multinomial Logistic Regression / Softmax Regression

Logistic Regression

Cross entropy

$$E(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} \log \left(1 + \exp(-y^{[i]} \cdot \mathbf{w}^T \mathbf{x})\right)$$

... Slides will be added before next lecture