Lecture 09

Multilayer Perceptrons

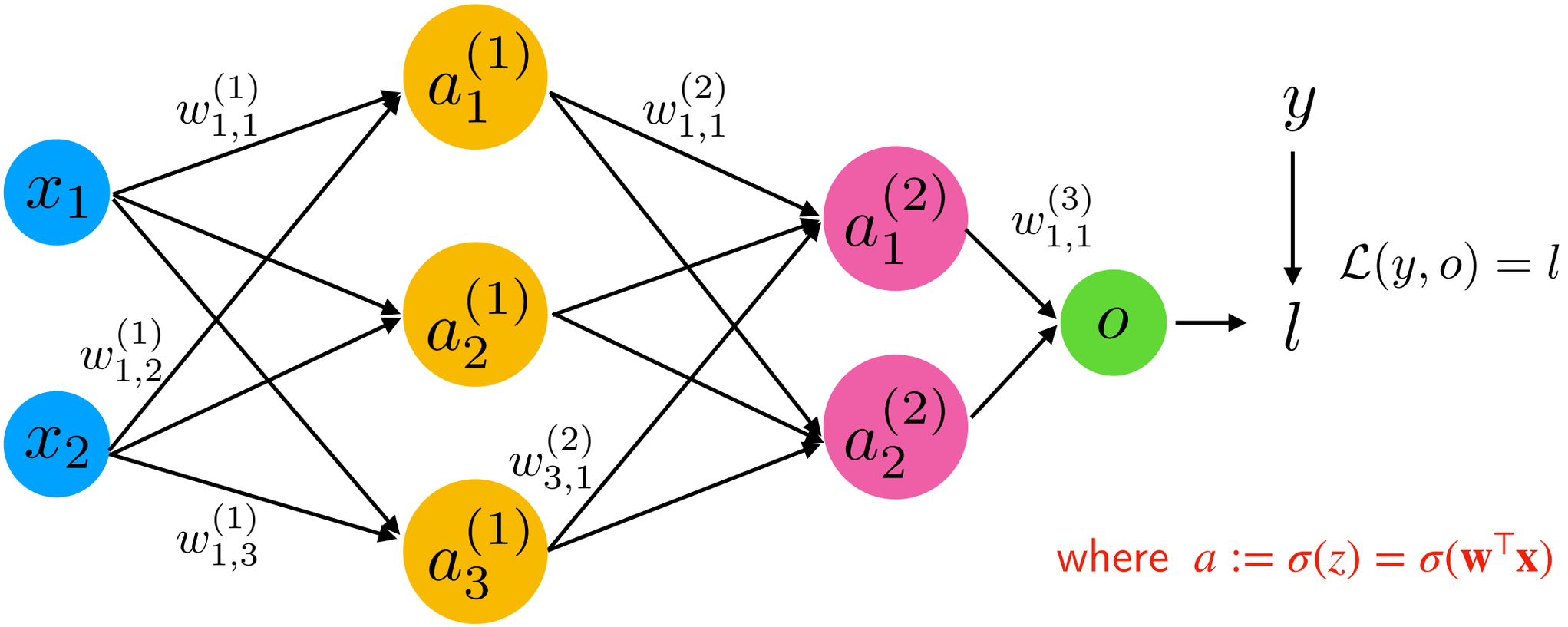
STAT 479: Deep Learning, Spring 2019

Sebastian Raschka

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

Graph with Fully-Connected Layers = Multilayer Perceptron

Nothing new, really

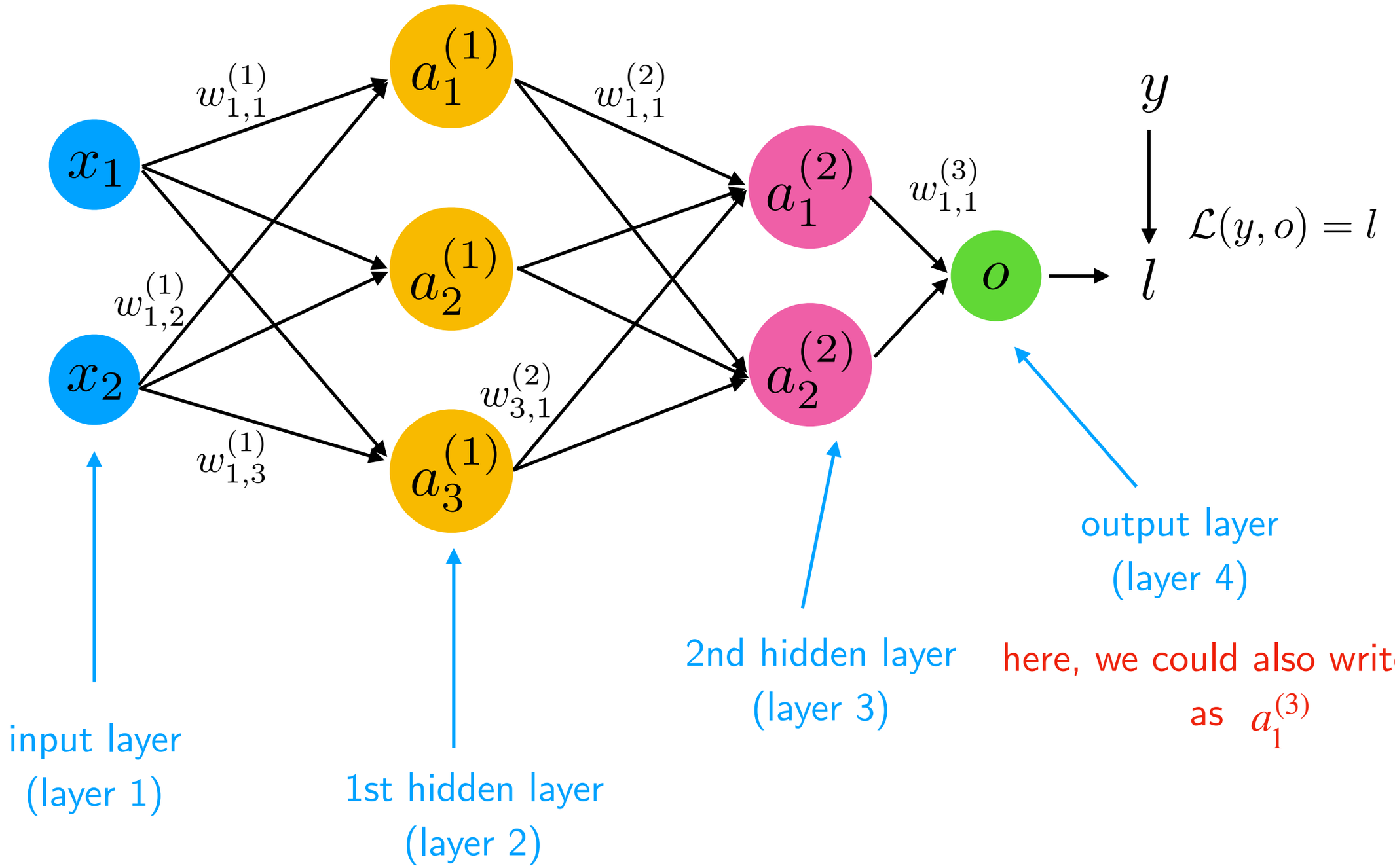


where $a := \sigma(z) = \sigma(\mathbf{w}^\top \mathbf{x})$

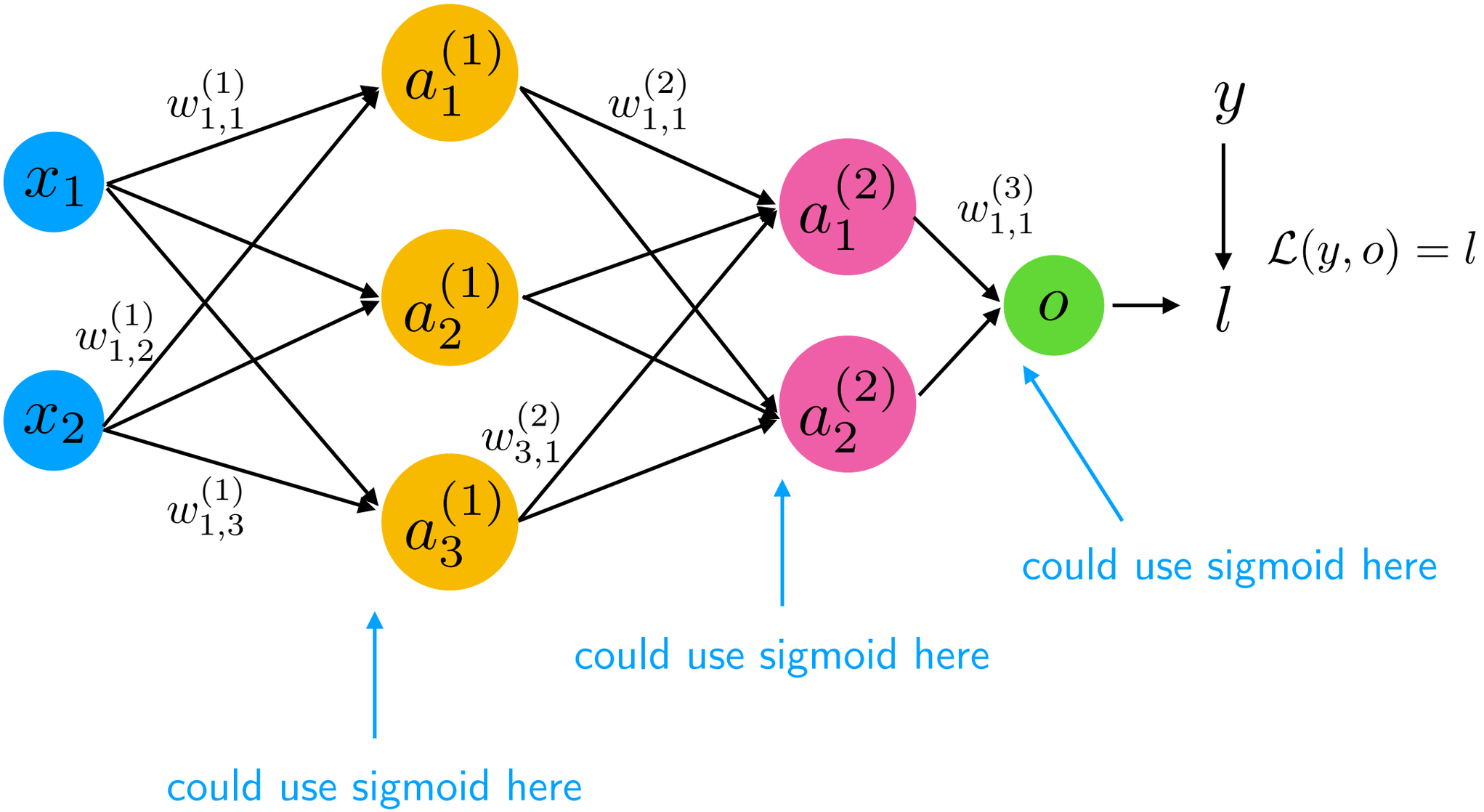
$$\frac{\partial l}{\partial w_{1,1}^{(1)}} = \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}} + \frac{\partial l}{\partial o} \cdot \frac{\partial o}{\partial a_2^{(2)}} \cdot \frac{\partial a_2^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{1,1}^{(1)}}$$

(Assume network for binary classification)

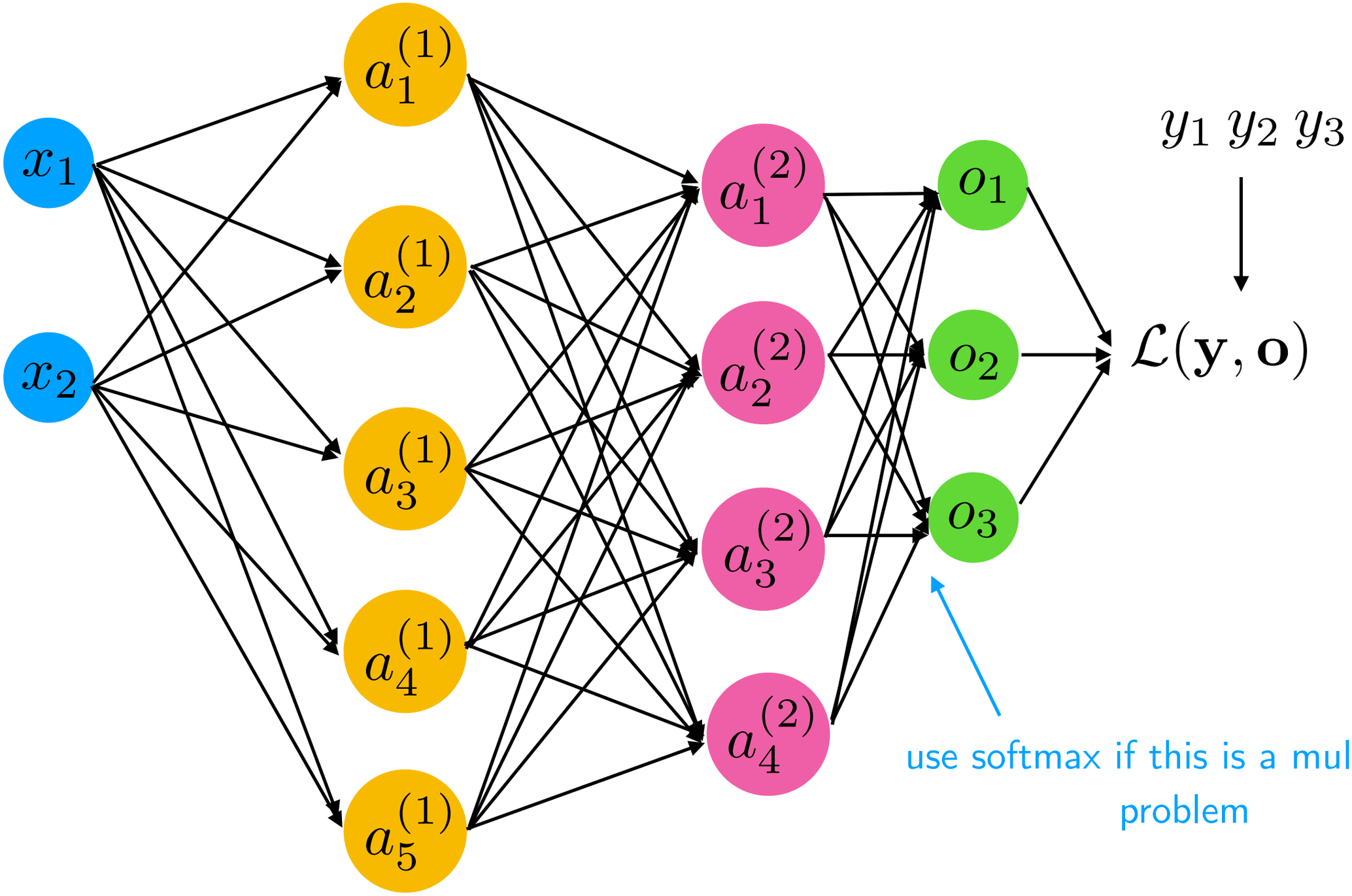
Graph with Fully-Connected Layers = Multilayer Perceptron



Graph with Fully-Connected Layers = Multilayer Perceptron



Graph with Fully-Connected Layers = Multilayer Perceptron

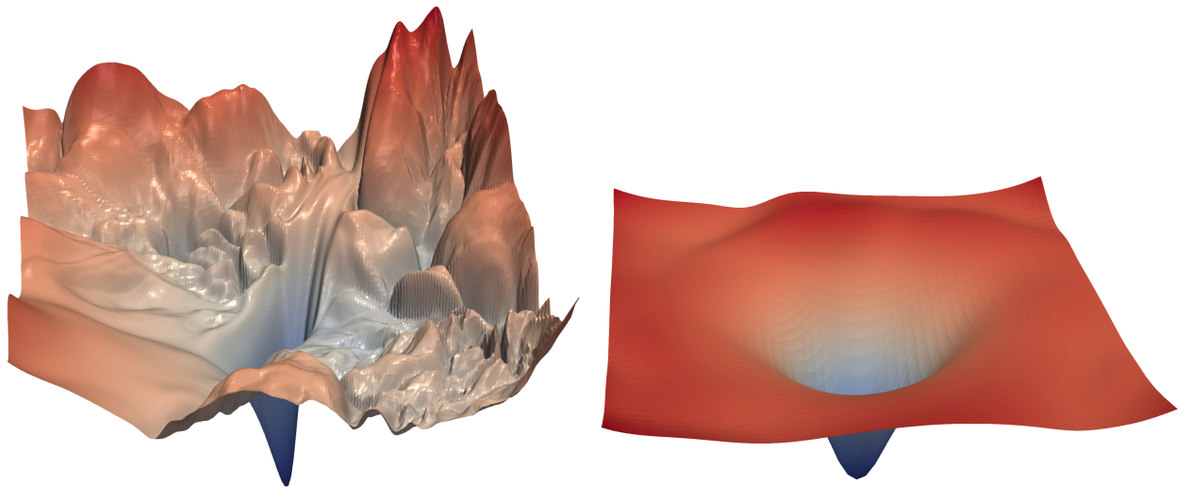


Note That the Loss is Not Convex Anymore

- Linear regression, Adaline, Logistic Regression, and Softmax Regression had convex loss functions with respect to the weights

- This is not the case anymore; in practice, we usually end up at different local minima if we repeat the training (e.g., by changing the random seed for weight initialization or shuffling the dataset while leaving all setting the same)

- In practice though, we WANT to explore different starting weights, however, because some lead to better solutions than others



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Image Source: Li, H., Xu, Z., Taylor, G., Studer, C. and Goldstein, T., 2018. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems* (pp. 6391-6401).

Multilayer Perceptron with Sigmoid Activation and MSE Loss (from scratch)

https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L09_mlp/code/mlp-fromscratch__sigmoid-mse.ipynb

Multilayer Perceptron with Sigmoid Activation and MSE Loss

vs

Multilayer Perceptron with Softmax Activation and Cross Entropy Loss (in PyTorch)

https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L09_mlp/code/mlp-pytorch.ipynb

About Softmax & Sigmoid in the Output Layer and Issues with MSE

- Sigmoid activation + MSE has the problem of very flat gradients when the output is very wrong i.e., 10^{-5} probability and class label 1

$$\frac{\partial \mathcal{L}}{\partial w_j} = -\frac{2}{n}(\mathbf{y} - \mathbf{a}) \odot \sigma(\mathbf{z}) \odot (1 - \sigma(\mathbf{z}))\mathbf{x}_j^\top$$ (from HW2: sigmoid + MSE neuron)

- Softmax (forces network to learn probability distribution over labels) in output layer is better than sigmoid because of the mutually exclusive labels as discussed in the Softmax lecture; hence, in output layer, better than sigmoid

About the DataLoader Class ...

- Example showing how you can create your own data loader to efficiently iterate through your own collection of images
  (pretend the MNIST images there are some custom image collection)

https://github.com/rasbt/stat479-deep-learning-ss19/tree/master/L09_mlp/code/custom-dataloader

Your 3rd Homework

- Experiment with a multi-layer perceptron on a subset of the QuickDraw dataset (due next Friday, March 8th 11:59 pm)

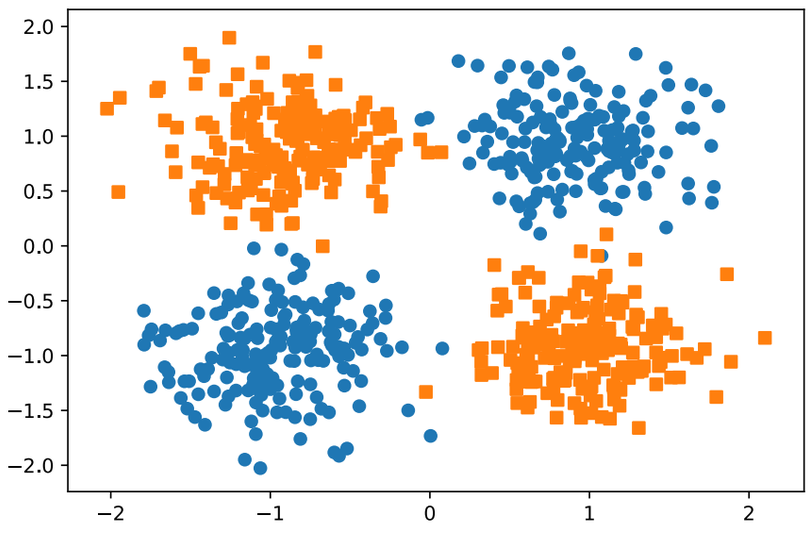
  https://github.com/rasbt/stat479-deep-learning-ss19/tree/master/L09_mlp/code/custom-dataloader

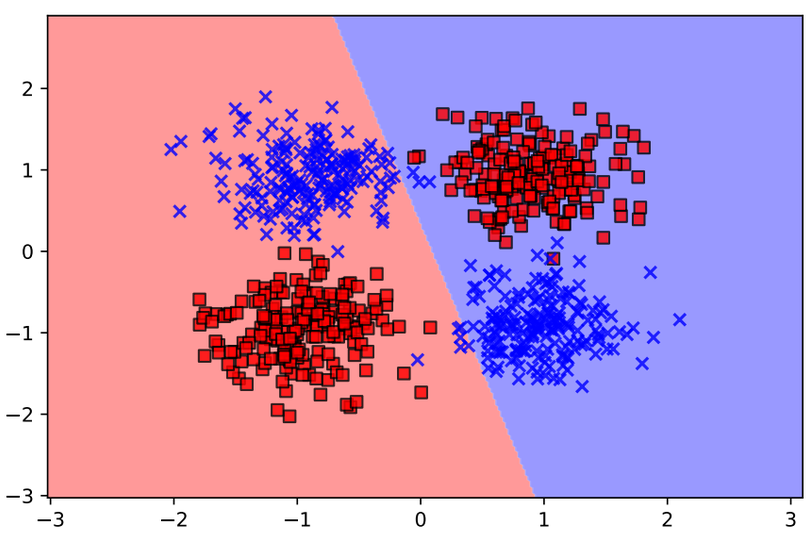
What happens if we initialize the multi-layer perceptron to all-zero weights?

Activation Functions

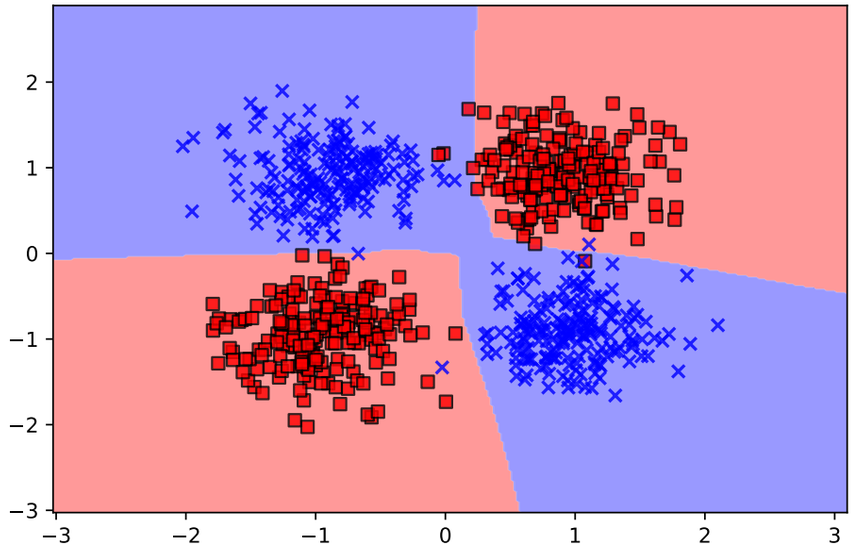
Question: What happens if we don't use non-linear activation functions?

Solving the XOR Problem with Non-Linear Activations





1-hidden layer MLP
with linear activation function

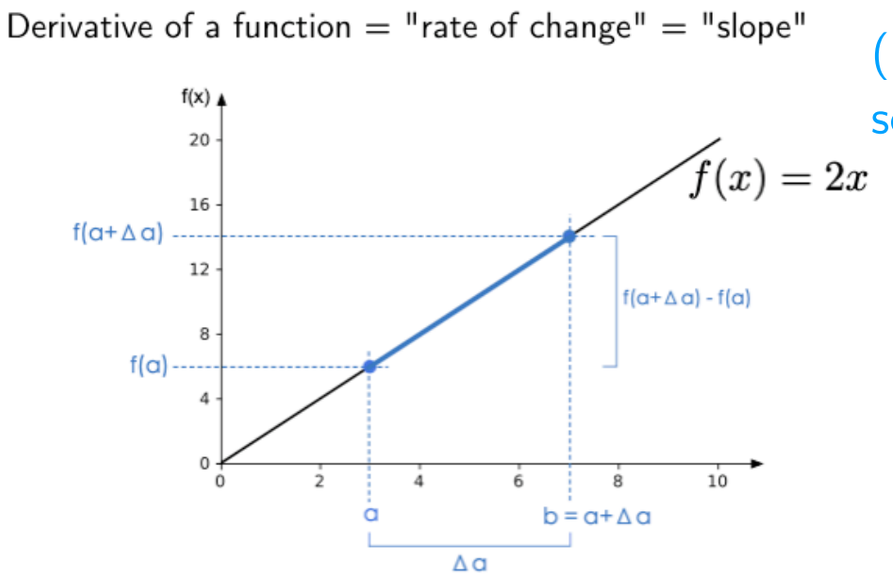


1-hidden layer MLP
with non-linear activation function (ReLU)

https://github.com/rasbt/stat479-deep-learning-ss19/blob/master/L09_mlp/code/xor-problem.ipynb

Gradient Checking

- Back in the day, we usually checked our gradients manually during debugging (note that this is super slow!)



$$\text{Slope} = \frac{f(a+\Delta a) - f(a)}{a + \Delta a - a} = \frac{f(a+\Delta a) - f(a)}{\Delta a}$$

(remember this from the calculus refresher section?)

Usually, a centered version works better, where epsilon is a very small value:

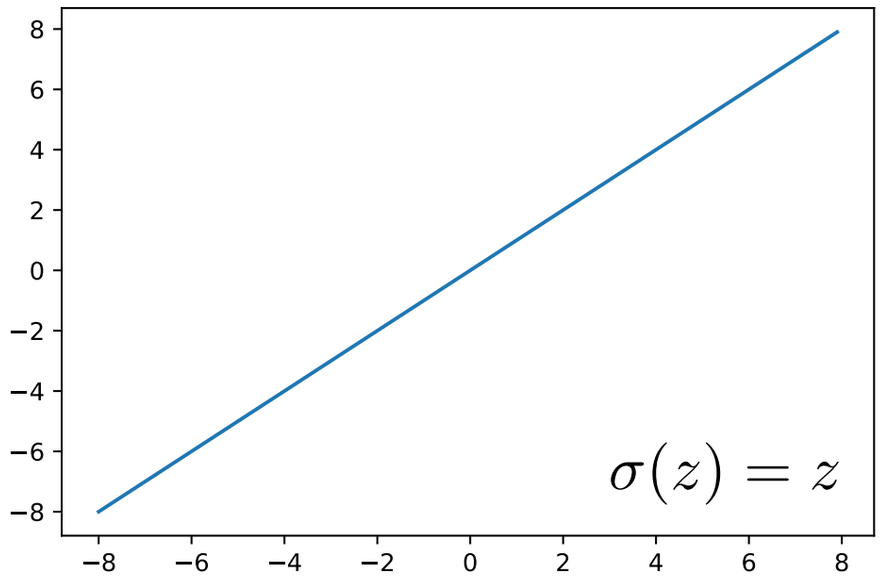
$$\frac{\mathcal{L}\left(w_{i,j}^{(l)} + \varepsilon\right) - \mathcal{L}\left(w_{i,j}^{(l)} - \varepsilon\right)}{2\varepsilon}$$

(then compare this with the symbolic gradient and compute the difference, e.g., via L2 norm)

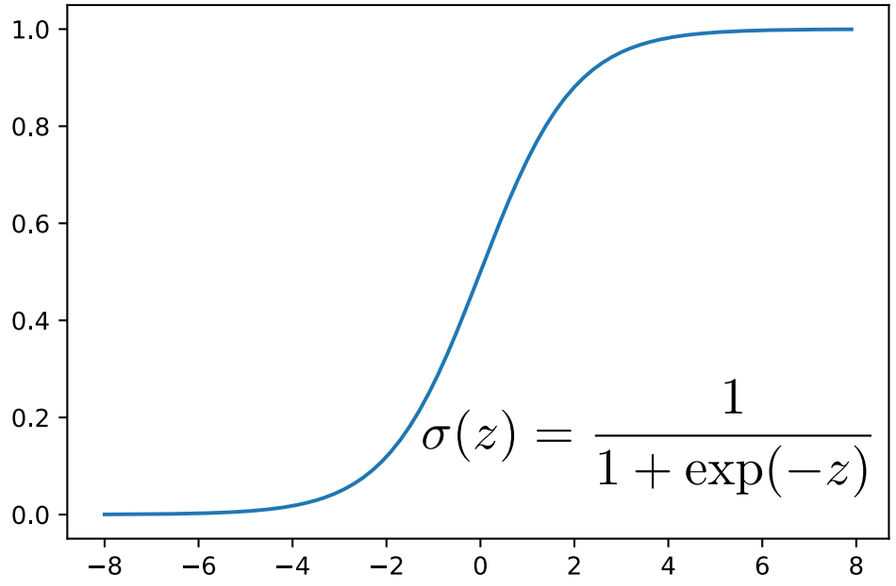
- Rarely done in practice anymore because we usually nowadays use autograd anyway, due to the complexity of deep neural networks

A Selection of Common Activation Functions (1)

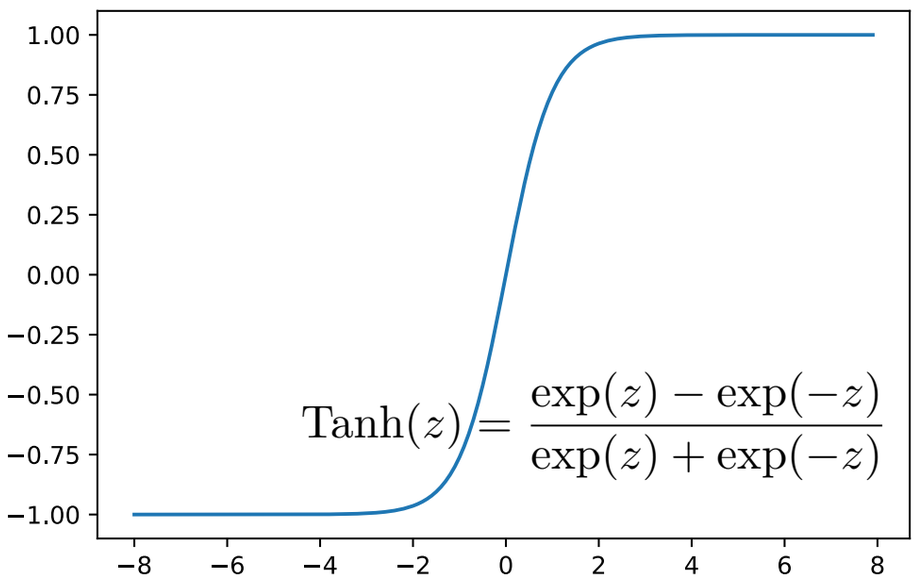
Identity



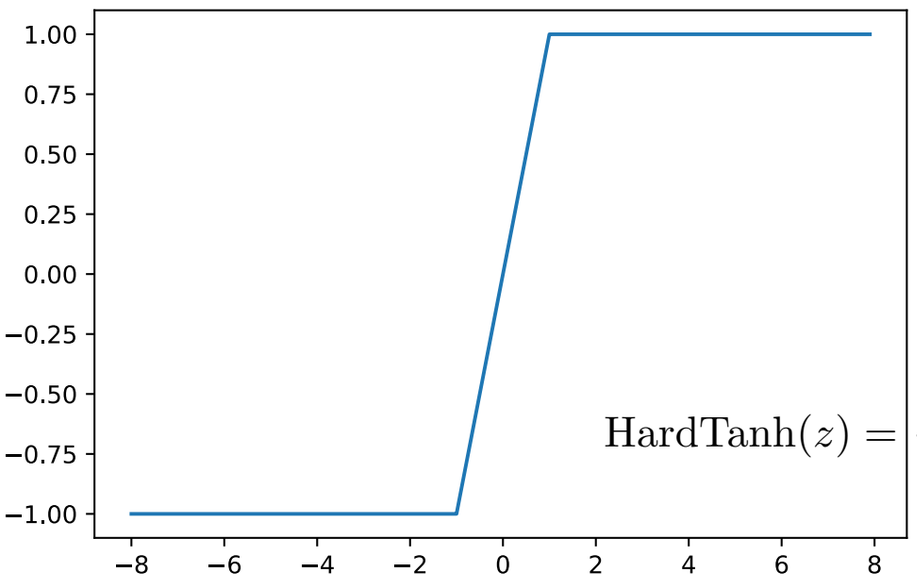
(Logistic) Sigmoid



Tanh ("tanH")

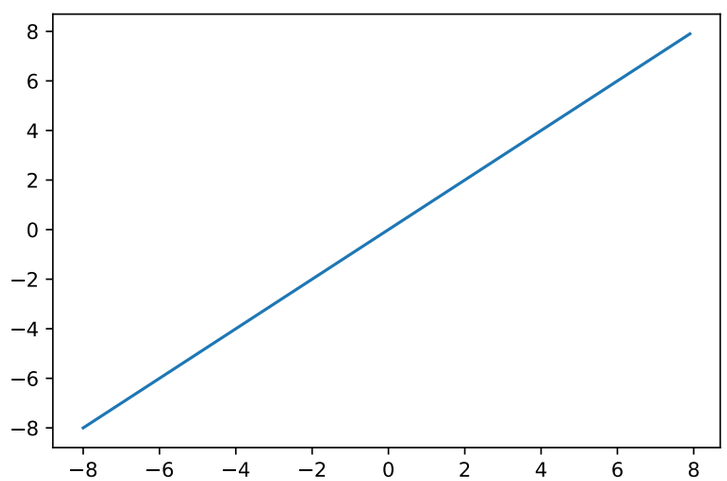


Hard Tanh

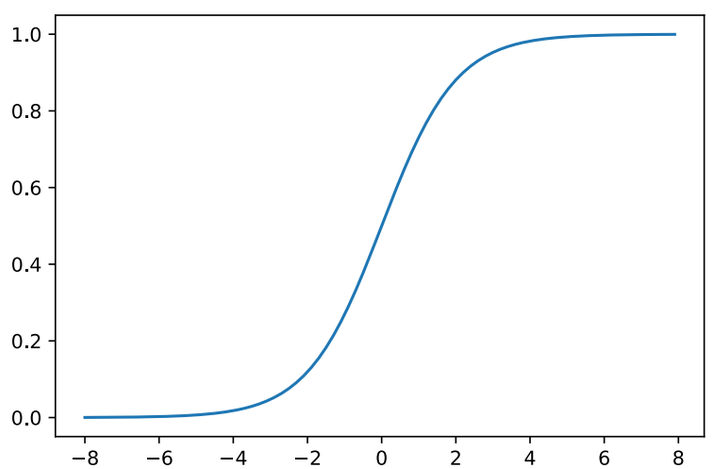


A Selection of Common Activation Functions (1)

Identity

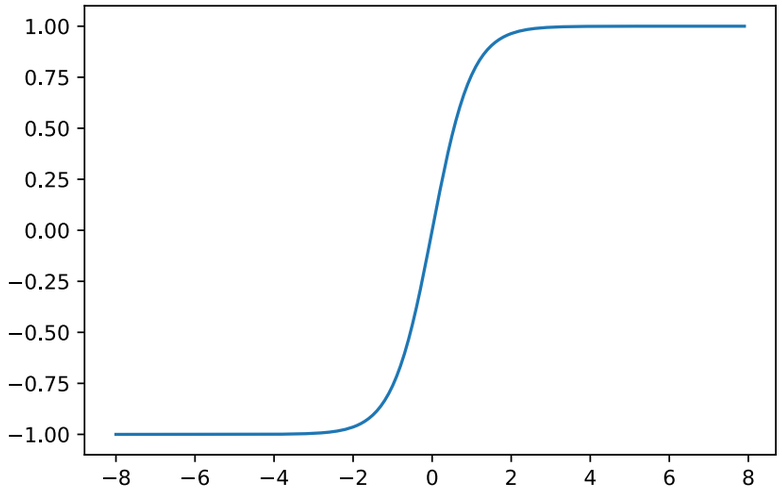


(Logistic) Sigmoid

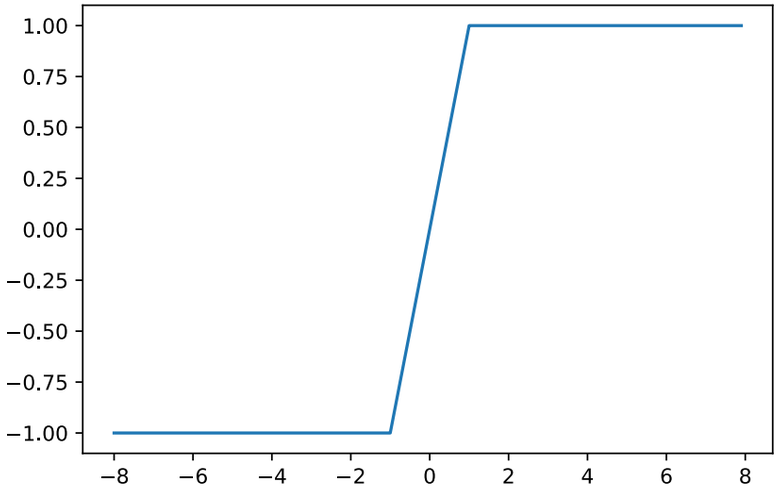


- Advantages of TanH
- Mean centering
- positive and negative values
- Larger gradients

Tanh ("tanH")

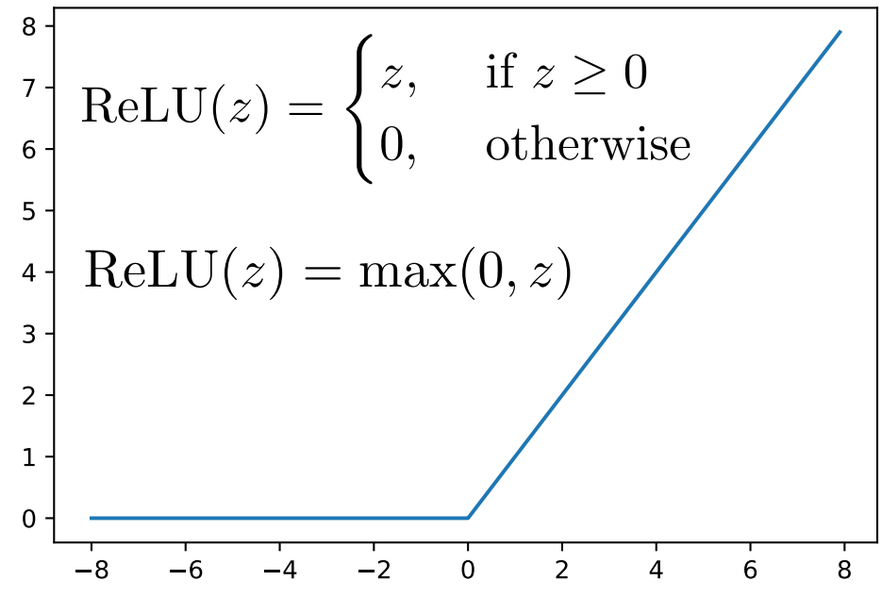


Hard Tanh

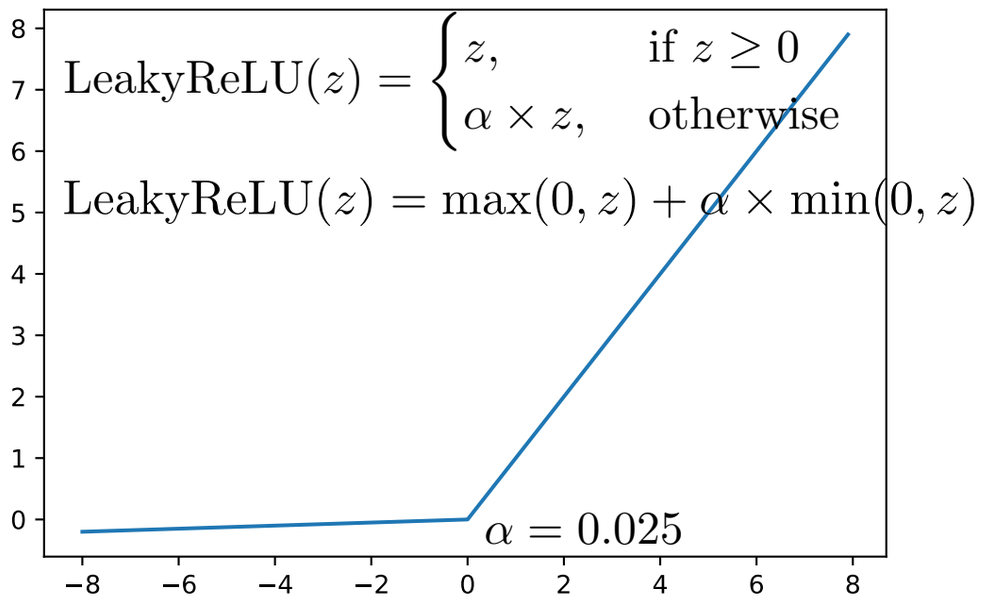


A Selection of Common Activation Functions (2)

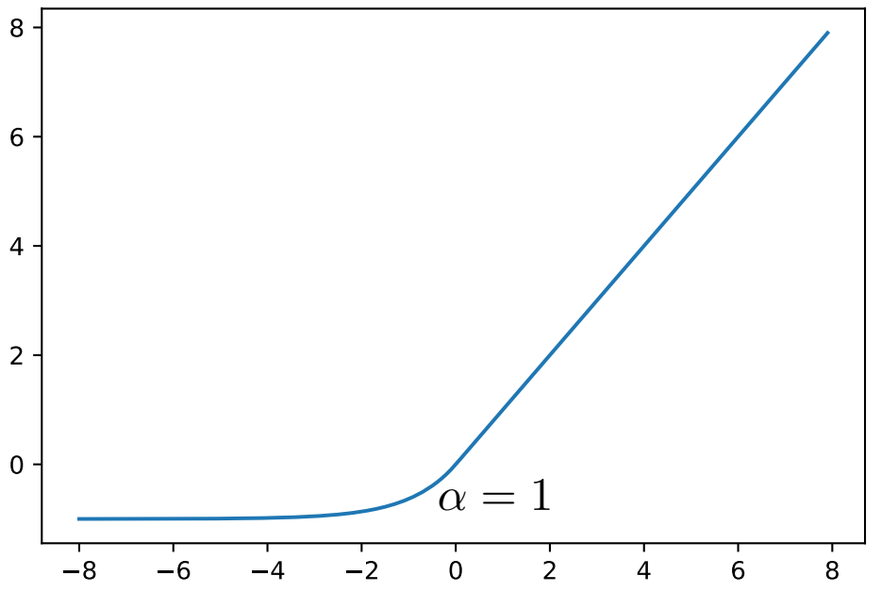
ReLU (Rectified Linear Unit)



Leaky ReLU



ELU (Exponential Linear Unit)



$$\mathrm{ELU}(z) = \max(0, z) + \min(0, \alpha \times (\exp(z) - 1))$$

PReLU (Parameterized Rectified Linear Unit)

here, alpha is a trainable parameter

$$\mathrm{PReLU}(z) = \begin{cases} z, & \text{if } z \geq 0 \\ \alpha z, & \text{otherwise} \end{cases}$$

$$\mathrm{PReLU}(z) = \max(0, z) + \alpha \times \min(0, z)$$

Pros and Cons of Different Activations

to be continued on Monday ...