Lecture 2: Machine Learning and Deep Learning Basics

Arsalan Mosenia

Chinmay Hegde

Linear Regression

- Now consider input: $x \in R^{\mathrm{d}}$ and output $y \in R$ the goal is to learn

$$\boldsymbol{y = f(x) = w_d x_d + \ldots + w_1 x_1 + w_0}$$

Linear Regression

Given training dataset $X \in R^{\mathrm{N}\times\mathrm{d}}$ and $Y \in R^{N}$

Training sample

$$\begin{pmatrix} 1 & x_{11} & x_{12} .. & x_{1d} \\ 1 & x_{21} & x_{22} .. & x_{d} \\ & & & \\ 1 & x_{N1} & x_{N2} .. & x_{Nd} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \\ w_d \end{pmatrix} = \begin{pmatrix} \widehat{y_1} \\ \widehat{y_2} \\ \\ \widehat{y_N} \end{pmatrix}$$

$$\hat{Y} = XW$$

Note: for simplicity we will assume that X includes a column of 1s

Linear Regression

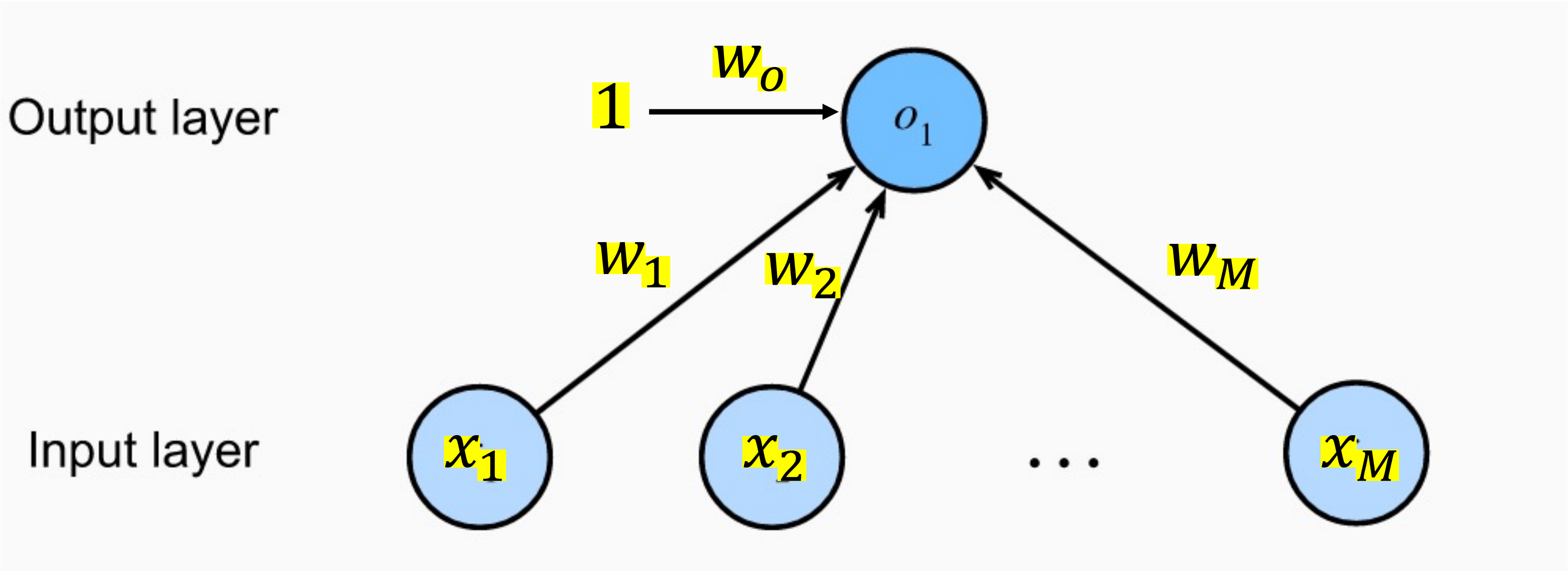
$$MSE = \sum (y - \hat{y})^2 = (Y - \hat{Y})^T . (Y - \hat{Y}) = (Y - XW)^T . (Y - XW)$$

Objective: $\min_W (Y - XW)^T . (Y - XW)$

Set derivate with respect to W to zero!

Solution: $W^* = (X^T X)^{-1} X^T Y$

“Connectionist" Perspective



- Each input is connected to the output node via an edge
- Output node sums inputs multiplied with their respective weights

A Biological Perspective

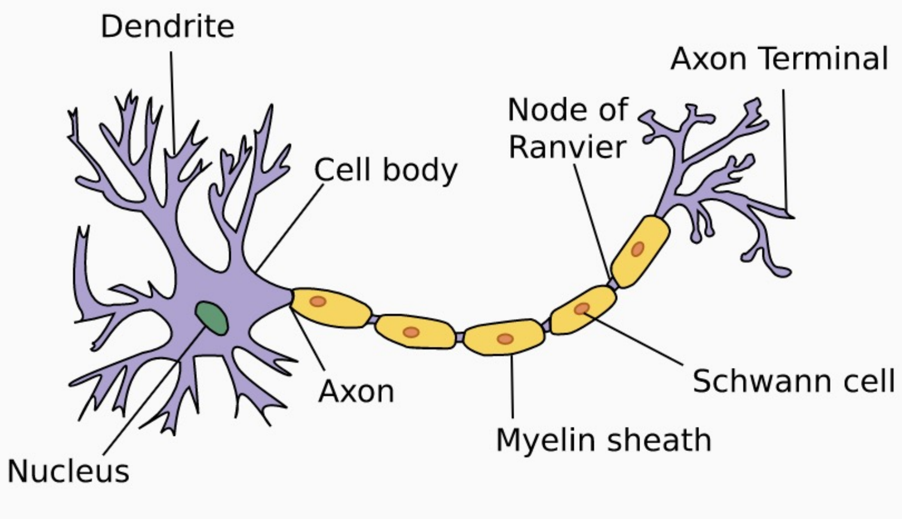


Fig. 3.1.3 The real neuron.

- **Dendrites**: accumulate inputs from neighboring neurons (inputs)
- **Nucleus**: performs a weighted sum of inputs, based on strengths of "synapses" + some non-linear activation
- **Axon**: output

$$y = f(x) = w_d x_d + \ldots + w_1 x_1 + w_0$$

Activation Functions

$o = \sigma(y)$ Non-linear "activation" function (example: "threshold")

For regression, the activation function is just the identity fn.

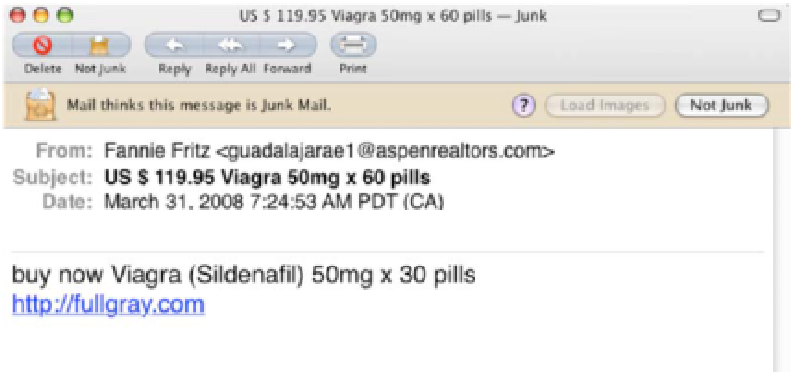
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{w}_{\boldsymbol{d}}\boldsymbol{x}_{\boldsymbol{d}} + \ldots + \boldsymbol{w}_{\boldsymbol{1}}\boldsymbol{x}_{\boldsymbol{1}} + \boldsymbol{w}_{\boldsymbol{0}}$$

- Non-linear "activation" function (example: "threshold")

Classification

Task (T):

- Emails $x \in$ all possible emails and y $\in \{spam, non\_spam\}$ find



SPAM

$$f: x \rightarrow y$$

Training Data:

A ”training dataset” emails marked as “spam” or “non-spam”

“Supervised Learning (Classification)”

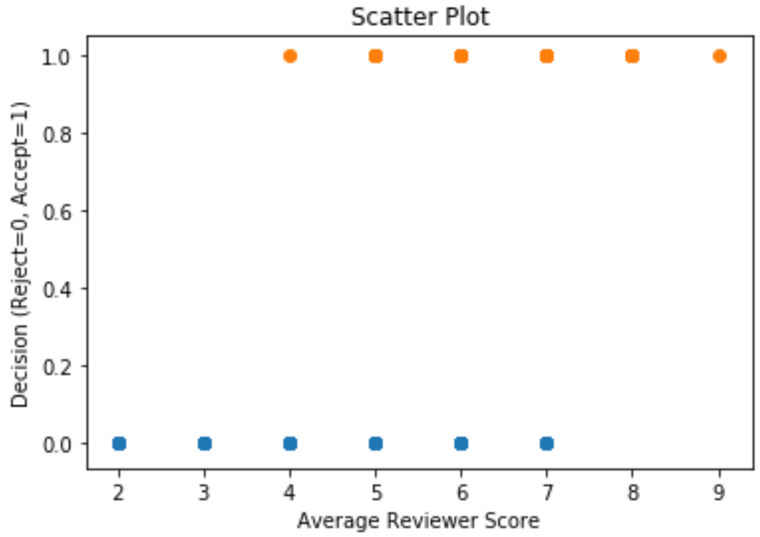
Binary Classification

- Simplest example where $x \in R$ and $y \in \{0,1\}$ "Categorical variable"

- Dataset of ICLR'18 review scores vs. accept/reject decisions

	TL;DR	_bibtex	abstract	authorids	authors	conf_1	conf_2	conf_3	decision	review	review_1	review_2	review_3	title
	None	@article{\nsharma2018hyperedge2vec:,\ntitle={H...	Data structured in form of overlapping or non-...	[sharm170@umn.edu, srjoty@ntu.edu.sg, himanshu...	[Ankit Sharma, Shafiq Joty, Himanshu Kharkwal,...	3.0	3.0	4.0	Reject	5.000000	5.0	5.0	5.0	Hyperedge2vec: Distributed Representations for...
	Query-based black-box attacks on deep neural n...	@article{\nnitin2018exploring,\ntitle={Explori...	Existing black-box attacks on deep neural netw...	[abhagoji@princeton.edu, _w@eecs.berkeley.edu,...	[Arjun Nitin Bhagoji, Warren He, Bo Li, Dawn S...	4.0	3.0	4.0	Reject	6.000000	5.0	6.0	7.0	Exploring the Space of Black-box Attacks on De...
	A theory and algorithmic framework for predict...	@article{\nd.2018learning,\ntitle={Learning We...	Predictive models that generalize well under d...	[fredrikj@mit.edu, kallus@cornell.edu, urish22...	[Fredrik D. Johansson, Nathan Kallus, Uri Shal...	3.0	3.0	4.0	Reject	6.666667	5.0	8.0	7.0	Learning Weighted Representations for Generali...
	We prove that DNN is a recursively approximate...	@article{\nzheng2018understanding,\ntitle={Und...	Deep learning achieves remarkable generalizati...	[zhenggh@mail.ustc.edu.cn, jtsang@bjtu.edu.cn,...	[Guanhua Zheng, Jitao Sang, Changsheng Xu]	3.0	3.0	2.0	Reject	3.666667	2.0	3.0	6.0	Understanding Deep Learning Generalization by

Binary Classification



Can you fit a linear model to this data?

TL;DR	_bibtex	abstract	authorids	authors	conf_1	conf_2	conf_3	decision	review	review_1	review_2	review_3	title
None	@article{\nsharma2018hyperedge2vec:,\ntitle={H...	Data structured in form of overlapping or non-...	[sharm170@umn.edu, srjoty@ntu.edu.sg, himanshu...	[Ankit Sharma, Shafiq Joty, Himanshu Kharkwal,...	3.0	3.0	4.0	Reject	5.000000	5.0	5.0	5.0	Hyperedge2vec: Distributed Representations for...
Query-based black-box attacks on deep neural n...	@article{\nnitin2018exploring,\ntitle={Explori...	Existing black-box attacks on deep neural netw...	[abhagoji@princeton.edu, _w@eecs.berkeley.edu,...	[Arjun Nitin Bhagoji, Warren He, Bo Li, Dawn S...	4.0	3.0	4.0	Reject	6.000000	5.0	6.0	7.0	Exploring the Space of Black-box Attacks on De...
A theory and algorithmic framework for predict...	@article{\nd.2018learning,\ntitle={Learning We...	Predictive models that generalize well under d...	[fredrikj@mit.edu, kallus@cornell.edu, urish22...	[Fredrik D. Johansson, Nathan Kallus, Uri Shal...	3.0	3.0	4.0	Reject	6.666667	5.0	8.0	7.0	Learning Weighted Representations for Generali...
We prove that DNN is a recursively approximate...	@article{\nzheng2018understanding,\ntitle={Und...	Deep learning achieves remarkable generalizati...	[zhenggh@mail.ustc.edu.cn, jitsang@bjtu.edu.cn,...	[Guanhua Zheng, Jitao Sang, Changsheng Xu]	3.0	3.0	2.0	Reject	3.666667	2.0	3.0	6.0	Understanding Deep Learning Generalization by

Logistic Regression

- First, let $p = \Pr\{y = 1|x\}$ Pr{Decision=Accept|Score}
- Consider the following function: $g = \log(\frac{p}{1-p})$

- g is called the "logits" function
- Logistic Regression: fit logits function using a linear model!

$$g = \log(\frac{p}{1-p}) = w_1 x + w_0$$

Logistic Regression

$$g = \log\left(\frac{p}{1-p}\right) = w_1 x + w_0$$

- Pr{Decision=Accept|Score}
- What is Pr{Decision=Reject|Score}

$$p = \frac{1}{1 + e^{-(w_1 x + w_0)}}$$

$$1 - p = \frac{e^{-(w_1 x + w_0)}}{1 + e^{-(w_1 x + w_0)}}$$

Decision Boundary?

How to Find Parameters? W1, W0

- What is Maximum Likelihood Estimation (MLE)?
    - Estimating the parameters of an assumed probability distribution
    - Given some observed data.

This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable.

Maximum Likelihood Estimation (MLE)

- Goal: Determine parameters for which the observed data have the highest joint probability.

- $L_n(w) = L_n(w; y) = f_n(y; w)$ *where* $Y = (y_1, y_2, y_3, \ldots, y_n)$
- $Find\ w \rightarrow argmaxL_n(w; Y)$

For Independent and identically distributed random (iid) variables:

$$f_n(\mathbf{y};\theta) = \prod_{k=1}^{n} f_k^{\mathsf{univar}}(y_k;\theta) \, .$$

Maximum Likelihood Estimation (MLE)

- What is the likelihood that the dataset came from our model?

$$p = \frac{1}{1 + e^{-(w_1 x + w_0)}}$$

$$1 - p = \frac{e^{-(w_1 x + w_0)}}{1 + e^{-(w_1 x + w_0)}}$$

#	X	Y
1	$x_1 = 3$	$y_1 = 0$
2	$x_2 = 8$	$y_2 = 1$
..		
..		
N	$x_N = 6$	$y_N = 1$

$$Likelihood = \frac{e^{-(3w_1 + w_0)}}{1 + e^{-(3w_1 + w_0)}} * \frac{1}{1 + e^{-(8w_1 + w_0)}} * \ldots \frac{1}{1 + e^{-(6w_1 + w_0)}}$$

Maximum Likelihood Estimation (MLE)

- $\hat{Y}_i = P(X_i) = P(Y_i = 1\,|X_i)$
- Likelihood $=\prod_{Y_i=1} P(X_i) * \prod_{Y_i=0}\left(1 - P(X_i)\right)$

$=\prod_{for\ all\ samples} P(X_i)^{y_i} * \left(1 - P(X_i)\right)^{1-y_i}$

MLE ≡ Maximizing Log of L

Log of Likelihood $= \sum_{i=1}^{n}[y_i *\ Log\left(P(x_i)\right) + (1 - y_i) * Log\left(1 - P(x_i)\right)]$

Maximum Likelihood Estimation (MLE)

Log of Likelihood = $\sum_{i=1}^{n}[y_i * \ Log(\hat{Y}_i) + (1 - y_i) * Log(1 - \hat{Y}_i)]$

MLE ≡ Maximizing (Log of L) ≡ Minimizing (–Log of L)

-Log(Likelihood) = -$\sum_{i=1}^{n}[y_i * \ Log(\hat{Y}_i) + (1 - y_i) * Log(1 - \hat{Y}_i)]$

Cost function? -Log(Likelihood)

Loss function? -$[y_i * \ Log(\hat{Y}_i) + (1 - y_i) * Log(1 - \hat{Y}_i)]$

Binary Cross-entropy Loss = Log-Loss = Logistic Regression Loss

Maximum Likelihood Estimation (MLE)

- What is the likelihood that the dataset came from our model?

#	X	Y
1	$x_1 = 3$	$y_1 = 0$
2	$x_2 = 8$	$y_2 = 1$
..		
..		
N	$x_N = 6$	$y_N = 1$

$$Likelihood = \frac{e^{-(3w_1+w_0)}}{1+e^{-(3w_1+w_0)}} * \frac{1}{1+e^{-(8w_1+w_0)}} * \ldots \frac{1}{1+e^{-(6w_1+w_0)}}$$

Maximize Likelihood

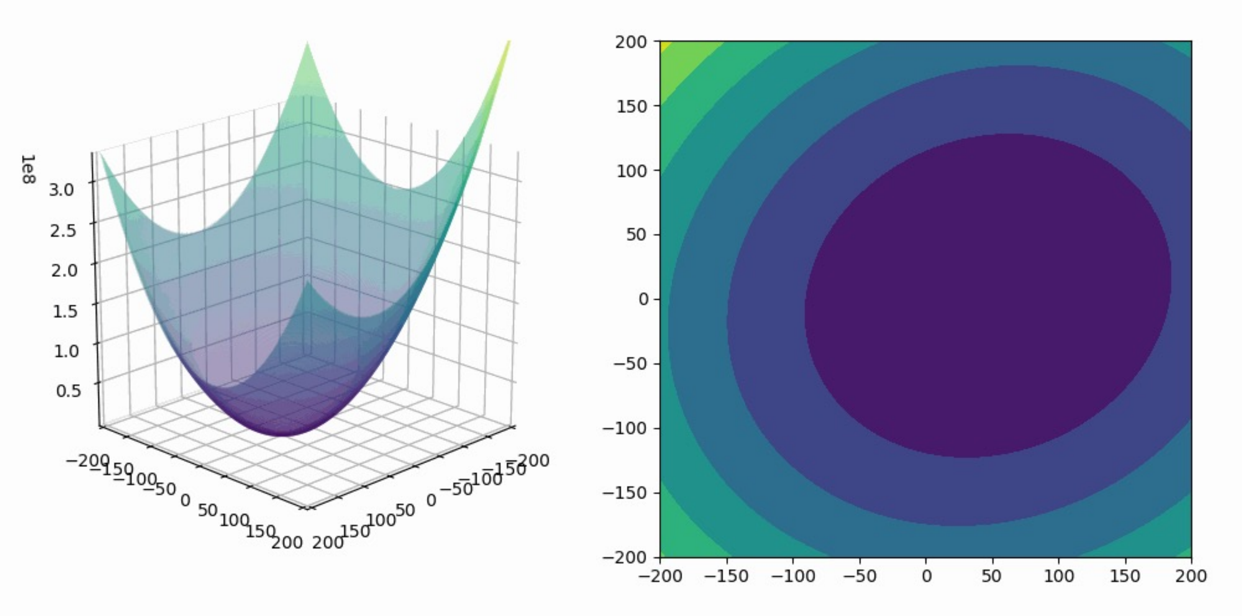
= Maximize Log-Likelihood

= Minimize (-Log-Likelihood)

"Cost"

How to Minimize the Cost Function?

- Sometimes, we cannot derive a formula to minimize loss. (For which model and loss function were we able to do this? ______)
- Using Gradient Descent
- Repeated steps in the opposite direction of the gradient at the current point, because this is the direction of steepest descent.



Gradient Descent for Minimizing Cost

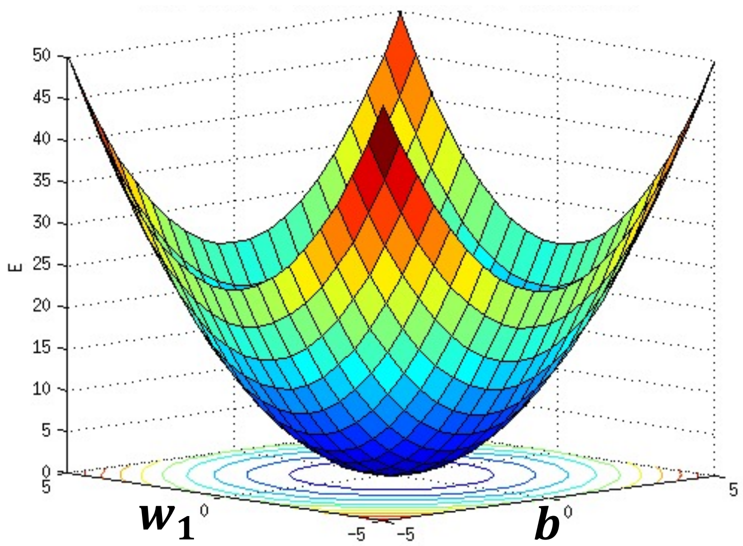
$$J(\boldsymbol{w}, b) = \sum_{i=1}^{N} l_i(\boldsymbol{w}, b)$$

Gradient Update:

Step size

Partial derivative

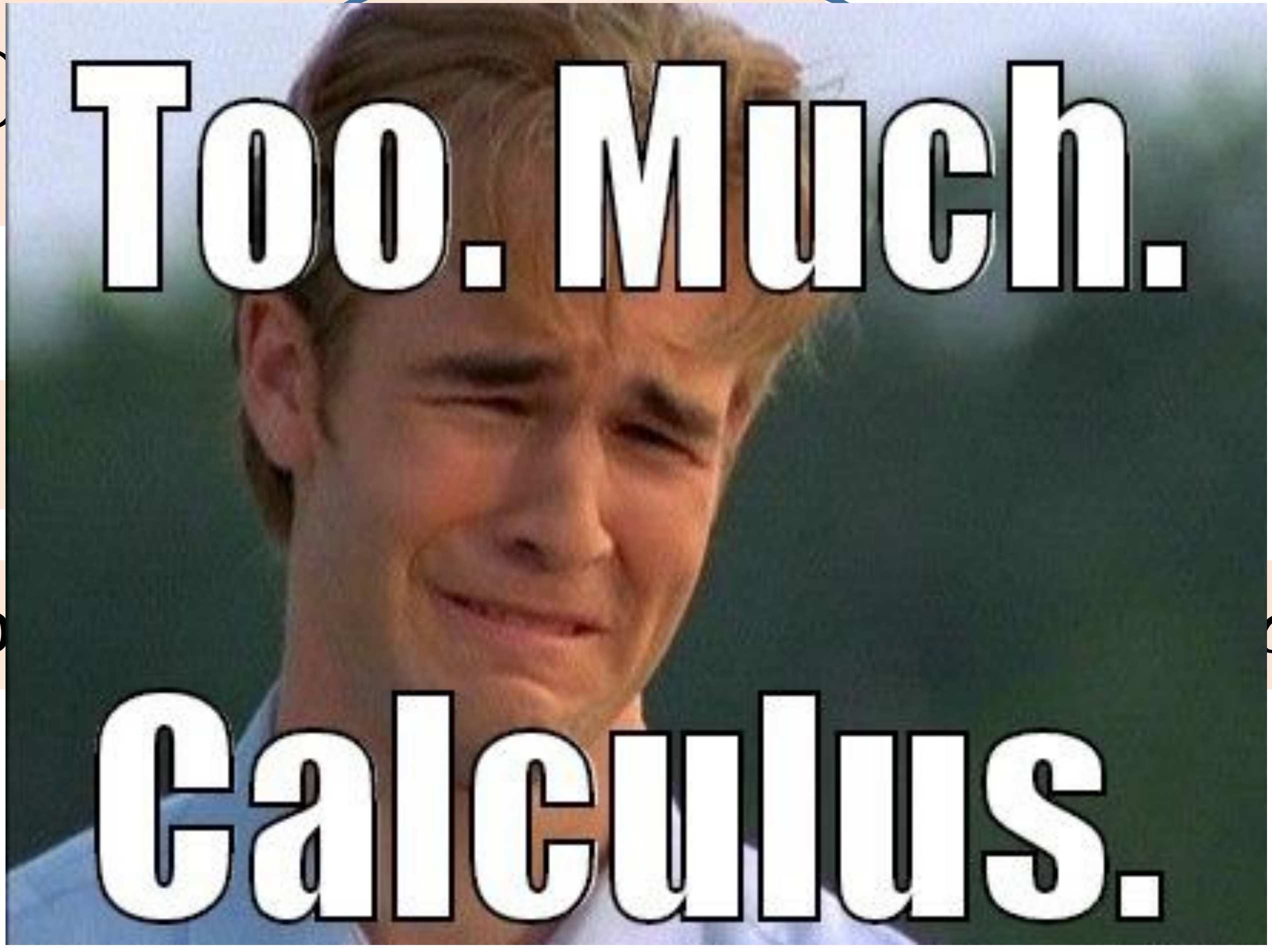
$$(\boldsymbol{w}, b) \leftarrow (\boldsymbol{w}, b) - \frac{\eta}{N} \sum_{i=1}^{N} \partial_{(\boldsymbol{w},b)} l_i(\boldsymbol{w}, b)$$



Computing the Partial Derivative

N

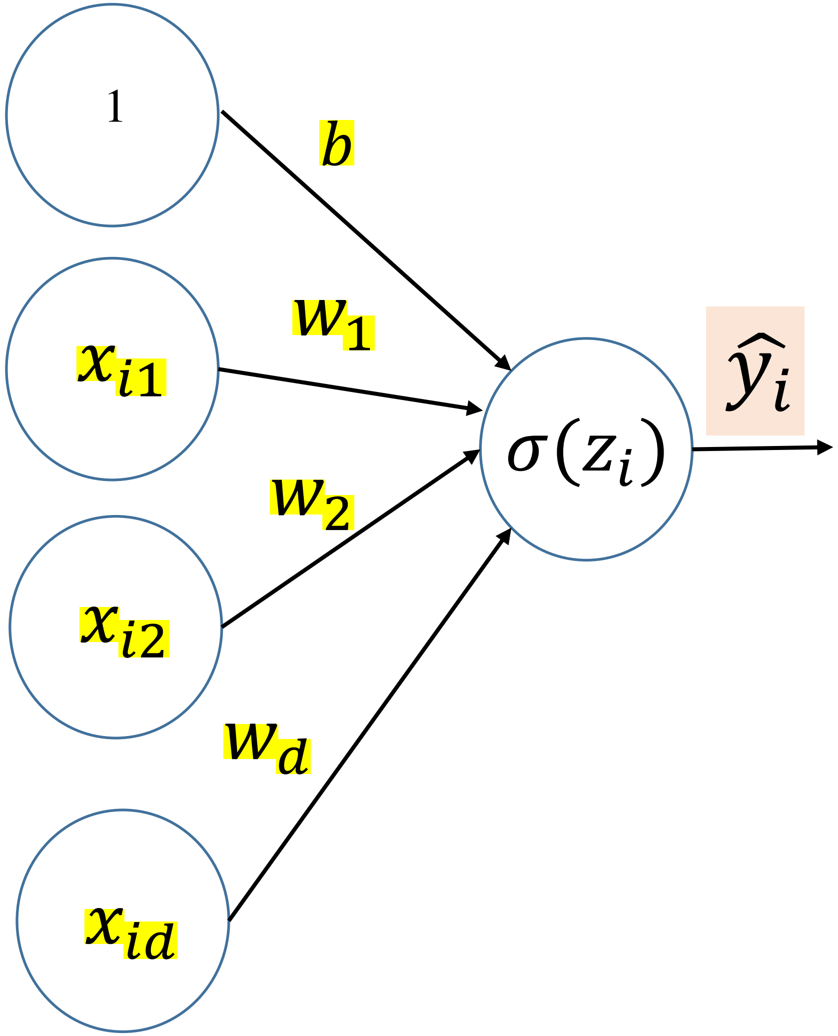
$(\boldsymbol{w}, b) \leftarrow (\boldsymbol{w}, b$

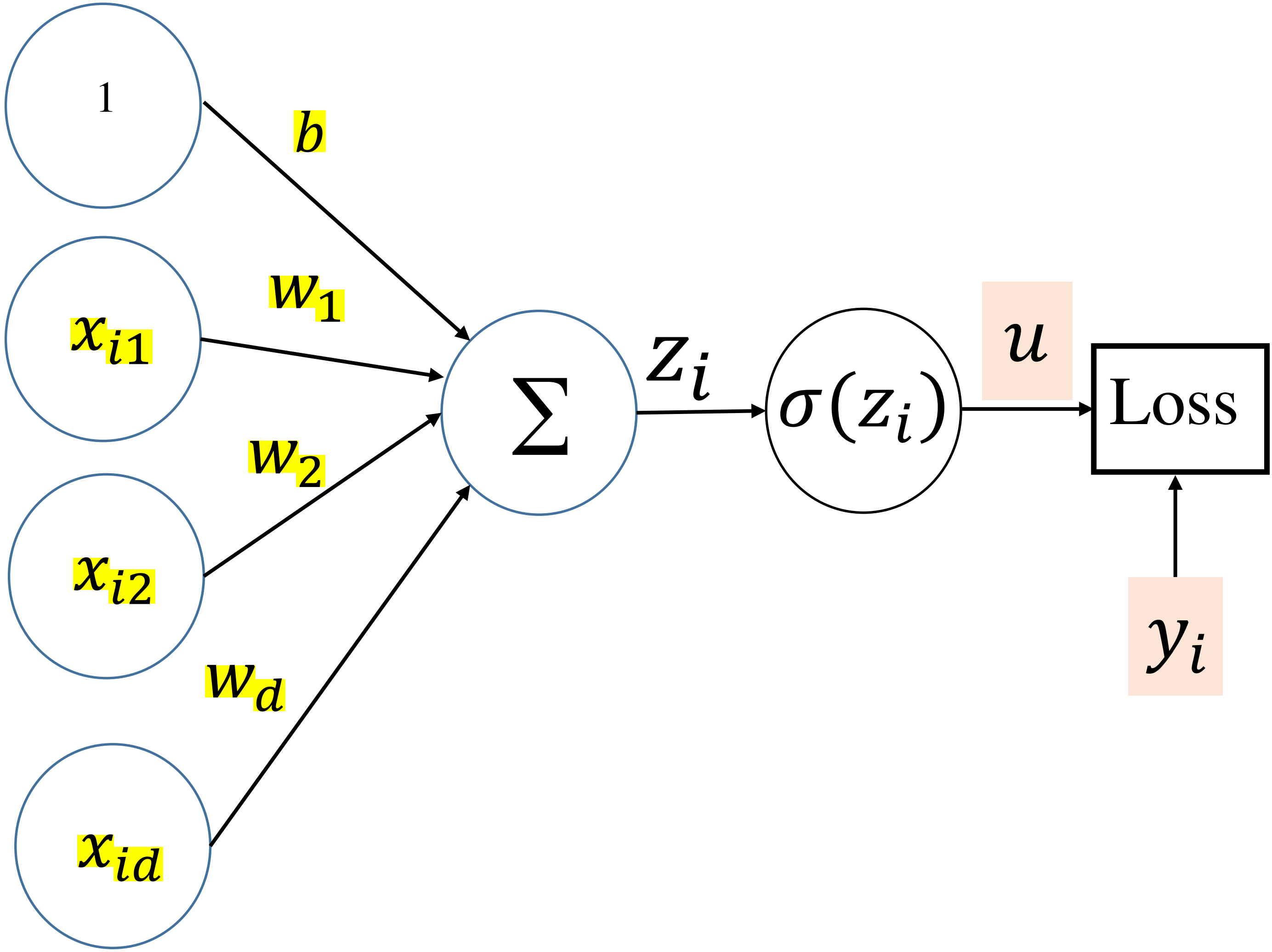


$\partial_{(\boldsymbol{w},b)}\, l_i(\boldsymbol{w}, b)$

$= \partial_{(\boldsymbol{w},b)}[-y_i \log$ … $\sigma(\boldsymbol{w}^T \boldsymbol{x}_i + b)]$

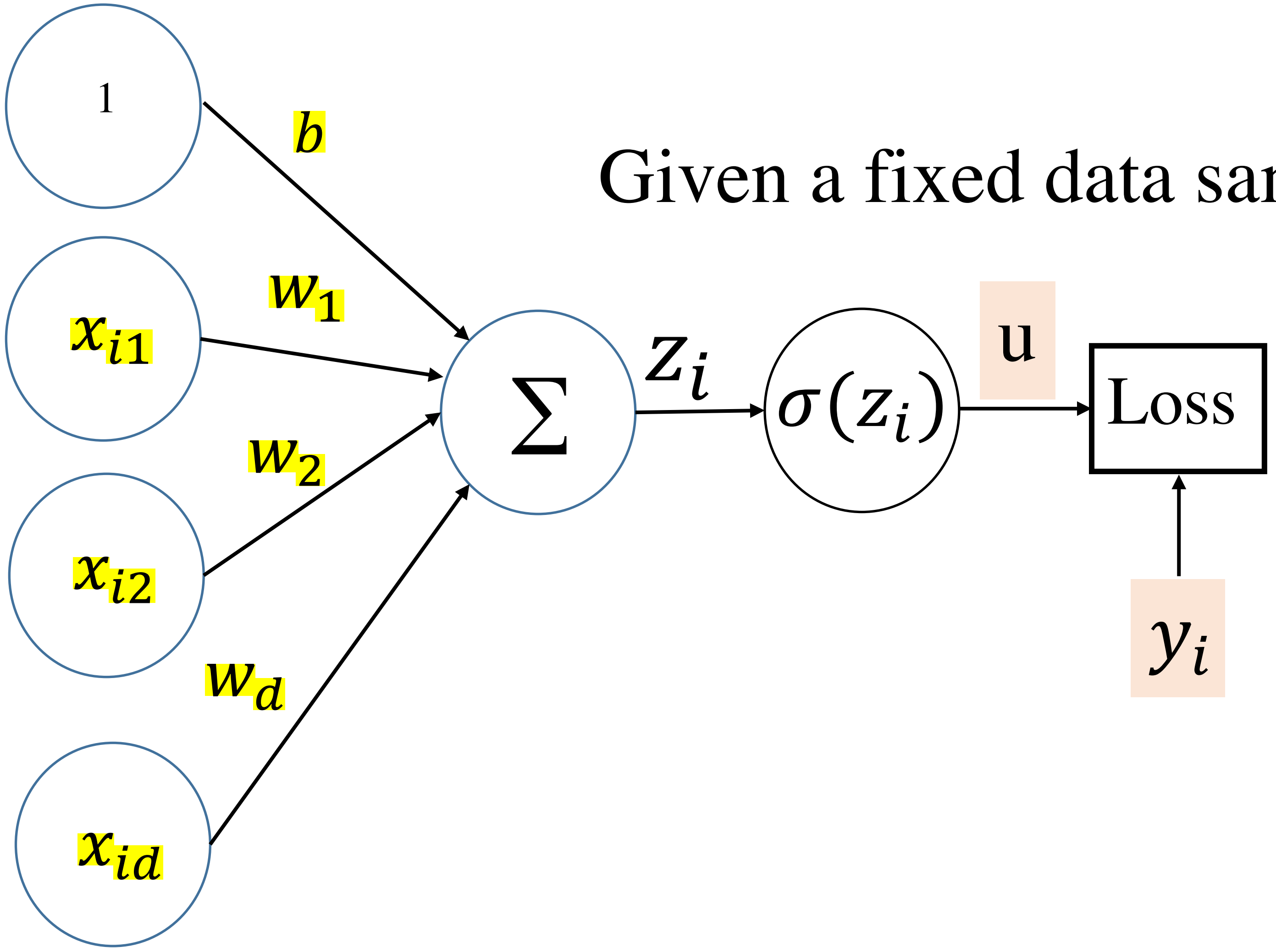
Back Propagation: Computational Graph



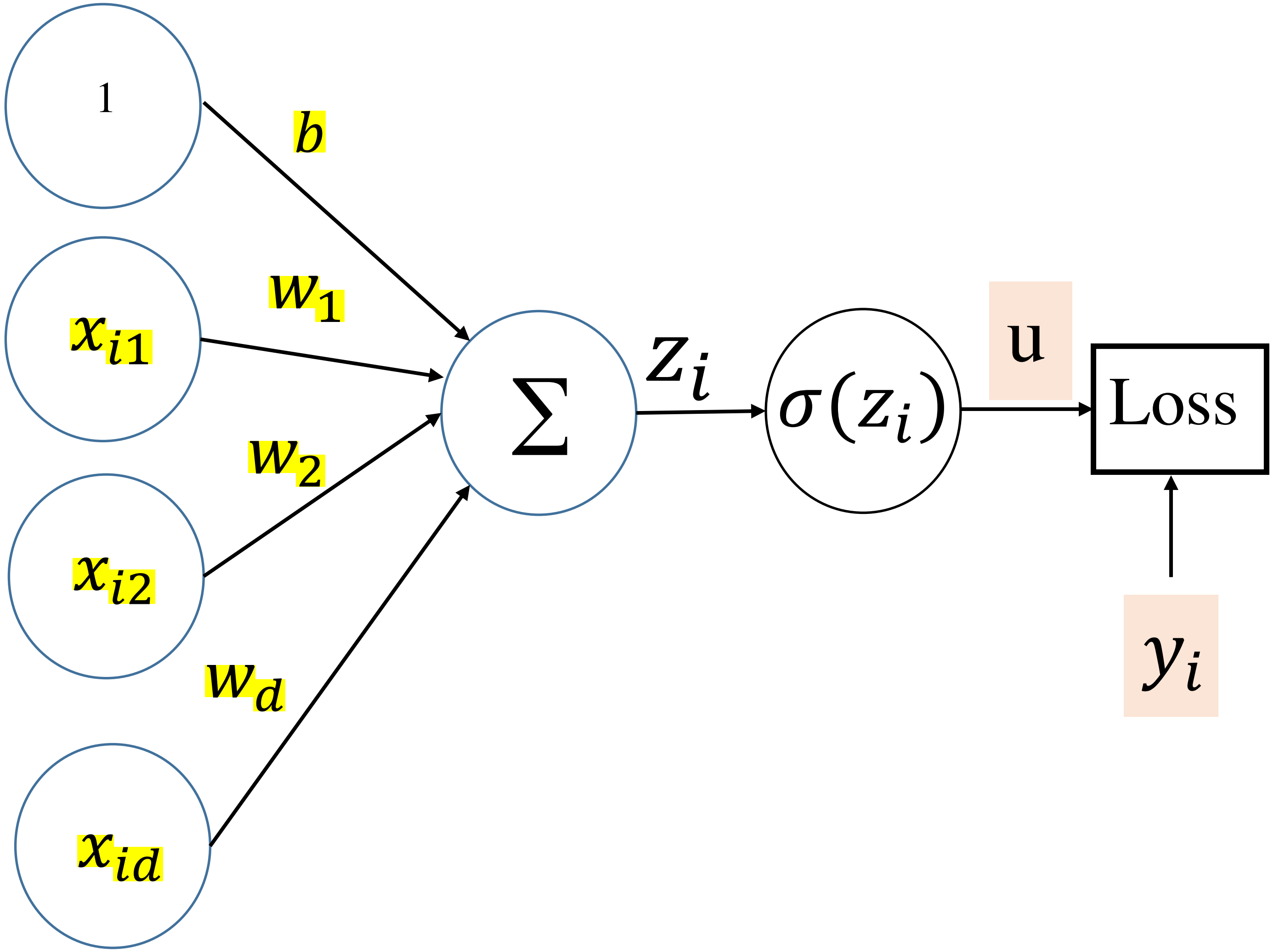


Back Propagation: Forward ()

Given a fixed data sample (x_i, y_i)

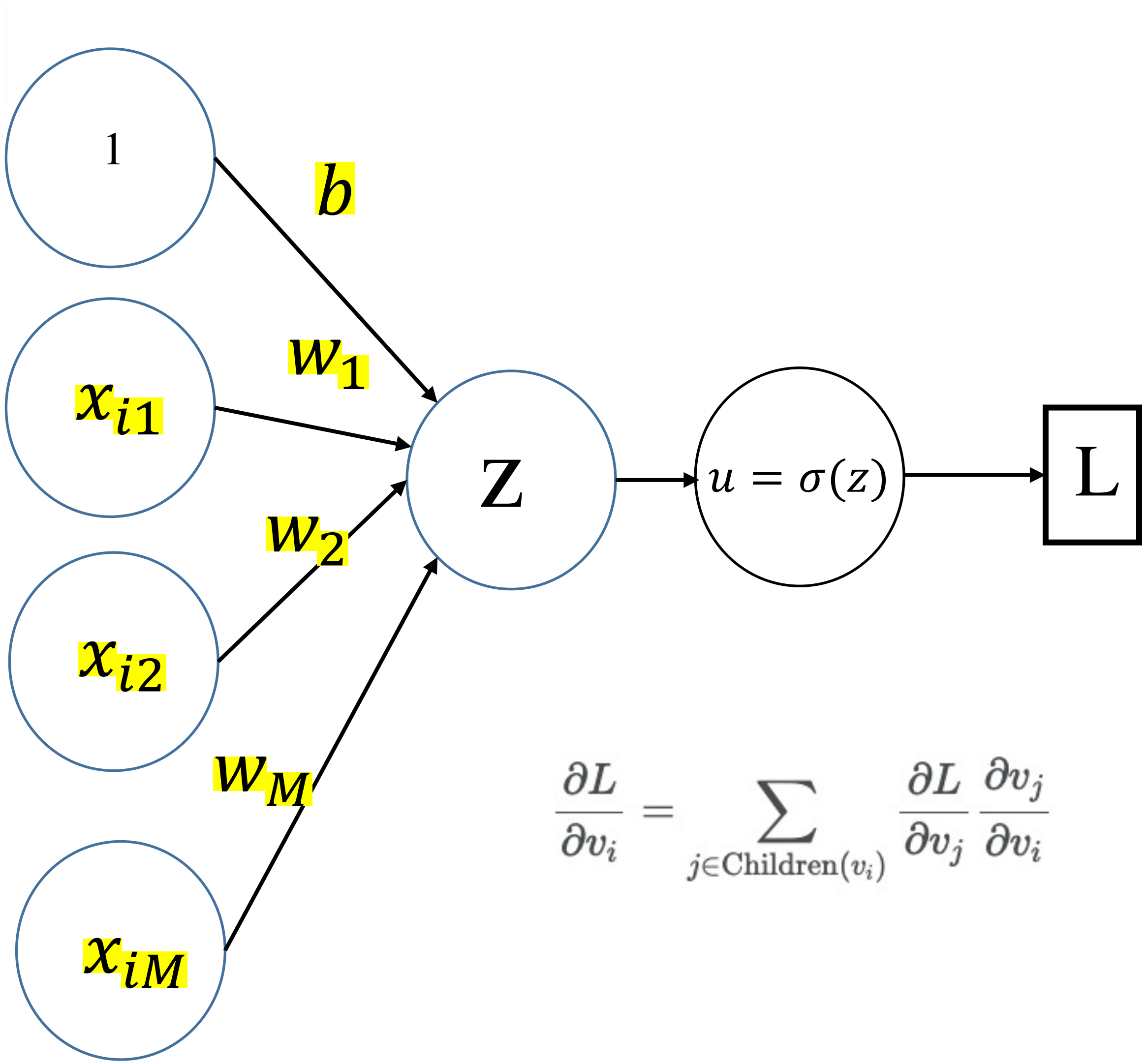


Back Propagation: Backward()



$$\frac{\partial L}{\partial v_i} = \sum_{j \in \text{Children}(v_i)} \frac{\partial L}{\partial v_j} \frac{\partial v_j}{\partial v_i}$$

Back Propagation: Example



$$\frac{\partial L}{\partial v_i} = \sum_{j \in \text{Children}(v_i)} \frac{\partial L}{\partial v_j} \frac{\partial v_j}{\partial v_i}$$

$$z = xw + b$$
$$u = \sigma(z)$$
$$Loss = -y\log(u) - (1-y)\log(1-u)$$

- (∂l)/∂l = 1
- (∂l)/∂u =[(∂l)/∂l] * [(∂l)/∂u] $= -y * \left(\frac{1}{u}\right) + (1-y) * \frac{1}{1-u}$
- (∂l)/∂z =[(∂l)/∂u] *[(∂u)/∂z]

  Note $\sigma'(z) = \sigma(z) * \big(1 - \sigma(z)\big)$

(∂l)/∂z = $\left[-y * \left(\frac{1}{u}\right) + (1-y) * \frac{1}{1-u}\right] * u(1-u)$

- (∂l)/∂w = (∂l)/∂z * (∂z)/∂w

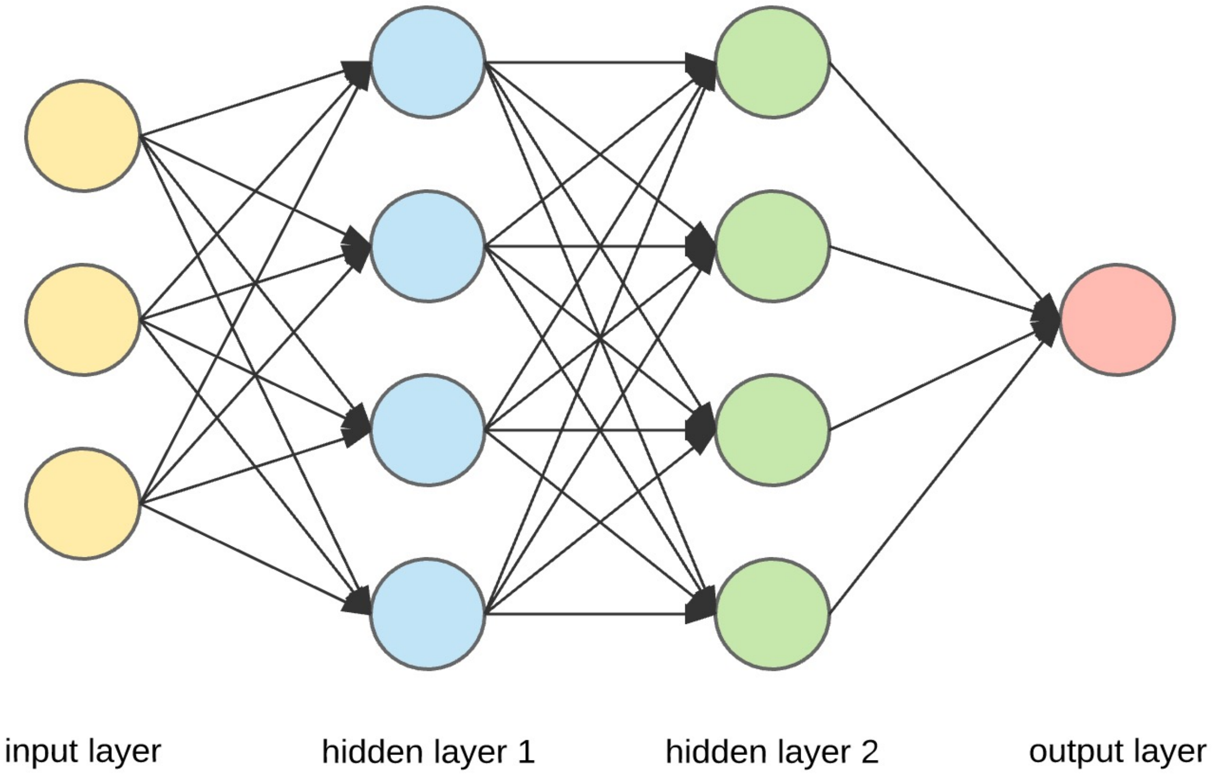
  $= \left[-y * \left(\frac{1}{u}\right) + (1-y) * \frac{1}{1-u}\right] * u(1-u)$ * (∂z)/∂w

  $= \left[-y * \left(\frac{1}{u}\right) + (1-y) * \frac{1}{1-u}\right] * u(1-u)$ * x

- (∂l)/∂b = (∂l)/∂z * (∂z)/∂b

  $= \left[-y * \left(\frac{1}{u}\right) + (1-y) * \frac{1}{1-u}\right] * u(1-u) * 1$

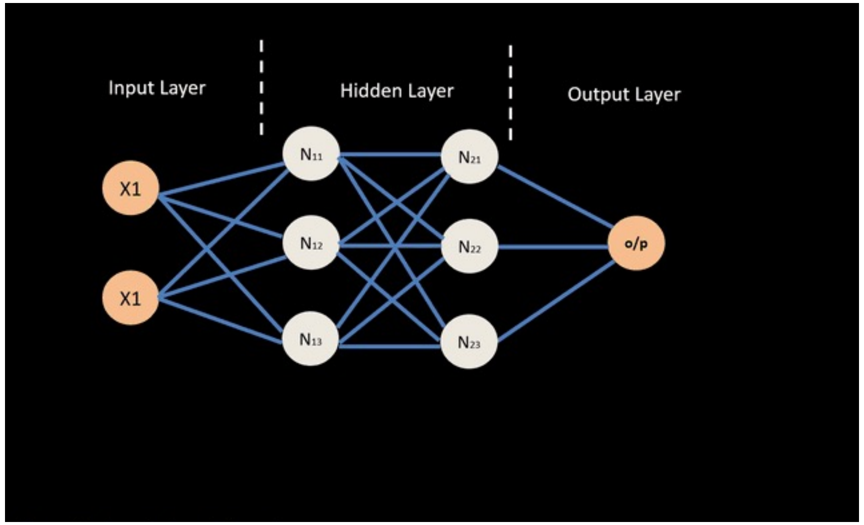
Neural Networks

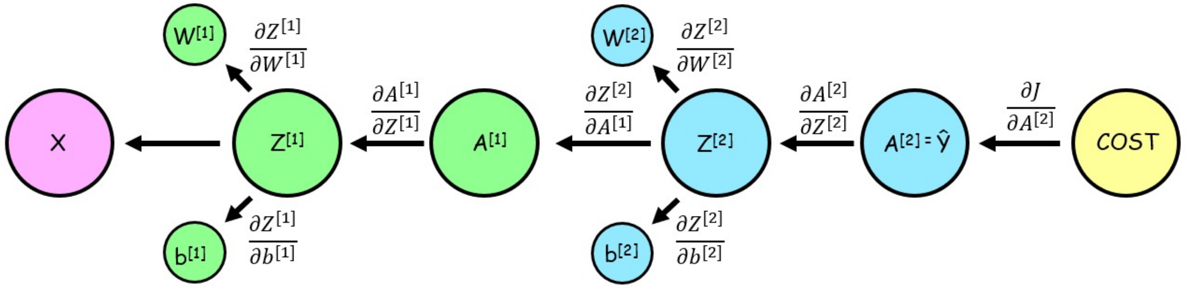


- 2-hidden layer network or 3-layer network
- **Params:** Weights/Biases
- Non-linearity? Activation functions
    - Relu, Leaky Relu, Tanh, Sigmoid

Revisiting Back Propagation

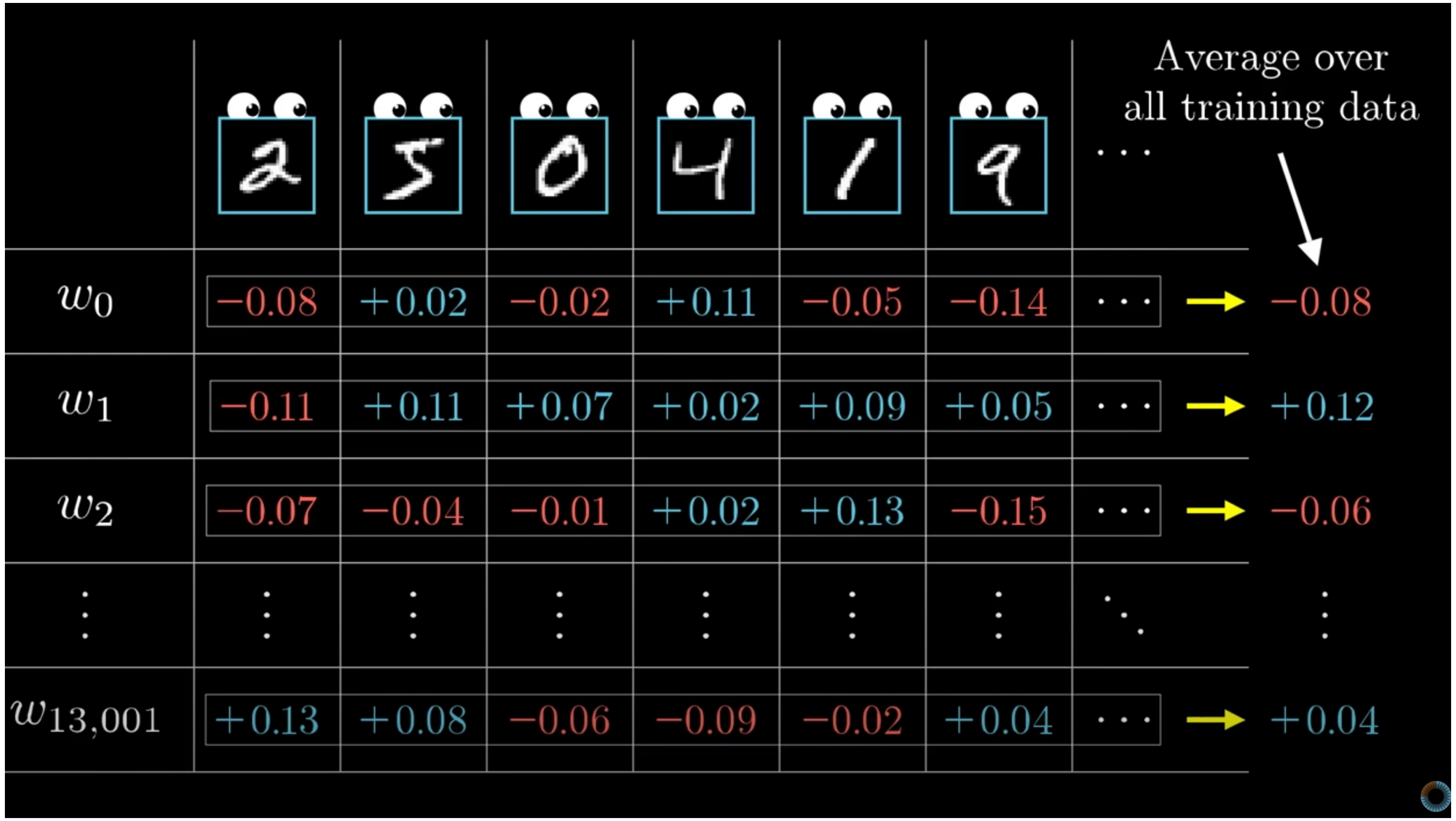
- Every partial derivative depends cleanly on partial derivatives of child nodes
- The computation is very structured, and everything can be calculated by traversing once through the graph.
  - Gradients can be computed efficiently.





Gradient Descent with Back Propagation

- GD is a generic way to find the local minimum
- Back propagation provides an efficient way to compute derivatives
- GD takes average of the weights suggested by examples for each step

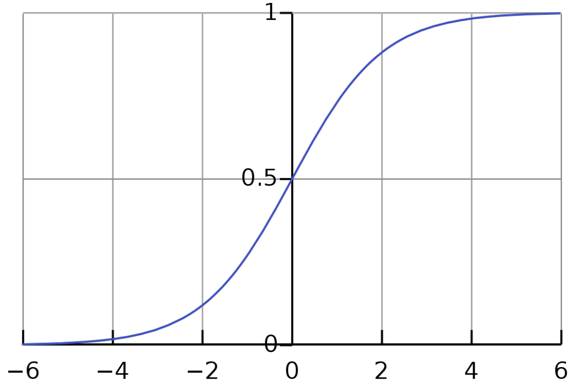


	2	5	0	4	1	9	...	Average over all training data
w_0	−0.08	+0.02	−0.02	+0.11	−0.05	−0.14	...	−0.08
w_1	−0.11	+0.11	+0.07	+0.02	+0.09	+0.05	...	+0.12
w_2	−0.07	−0.04	−0.01	+0.02	+0.13	−0.15	...	−0.06
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮
$w_{13,001}$	+0.13	+0.08	−0.06	−0.09	−0.02	+0.04	...	+0.04

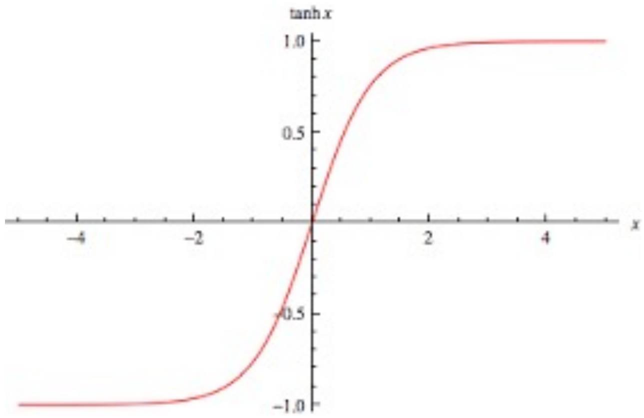
Revisiting Activation Functions

Non-linearity? Activation functions

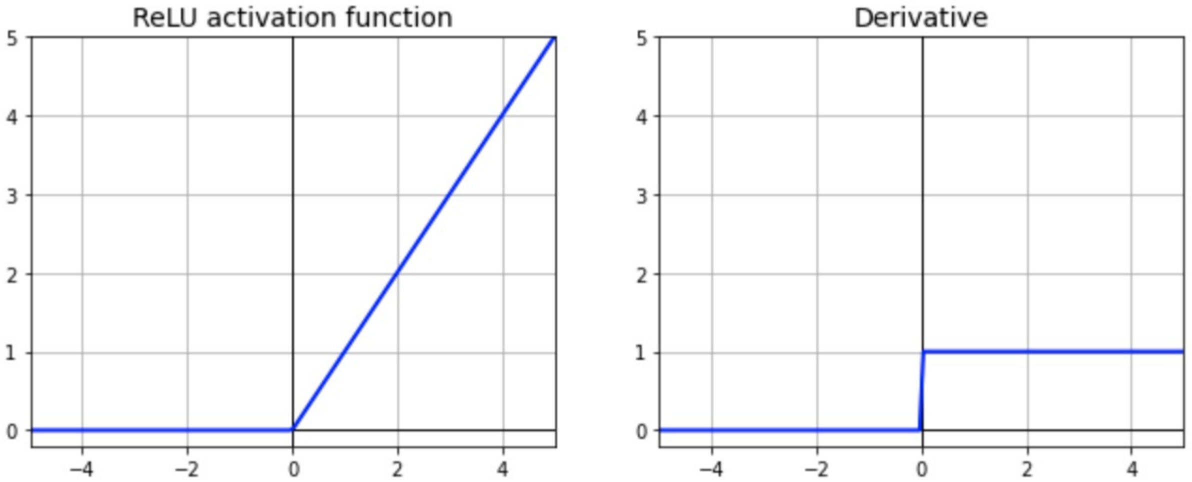
- Sigmoid



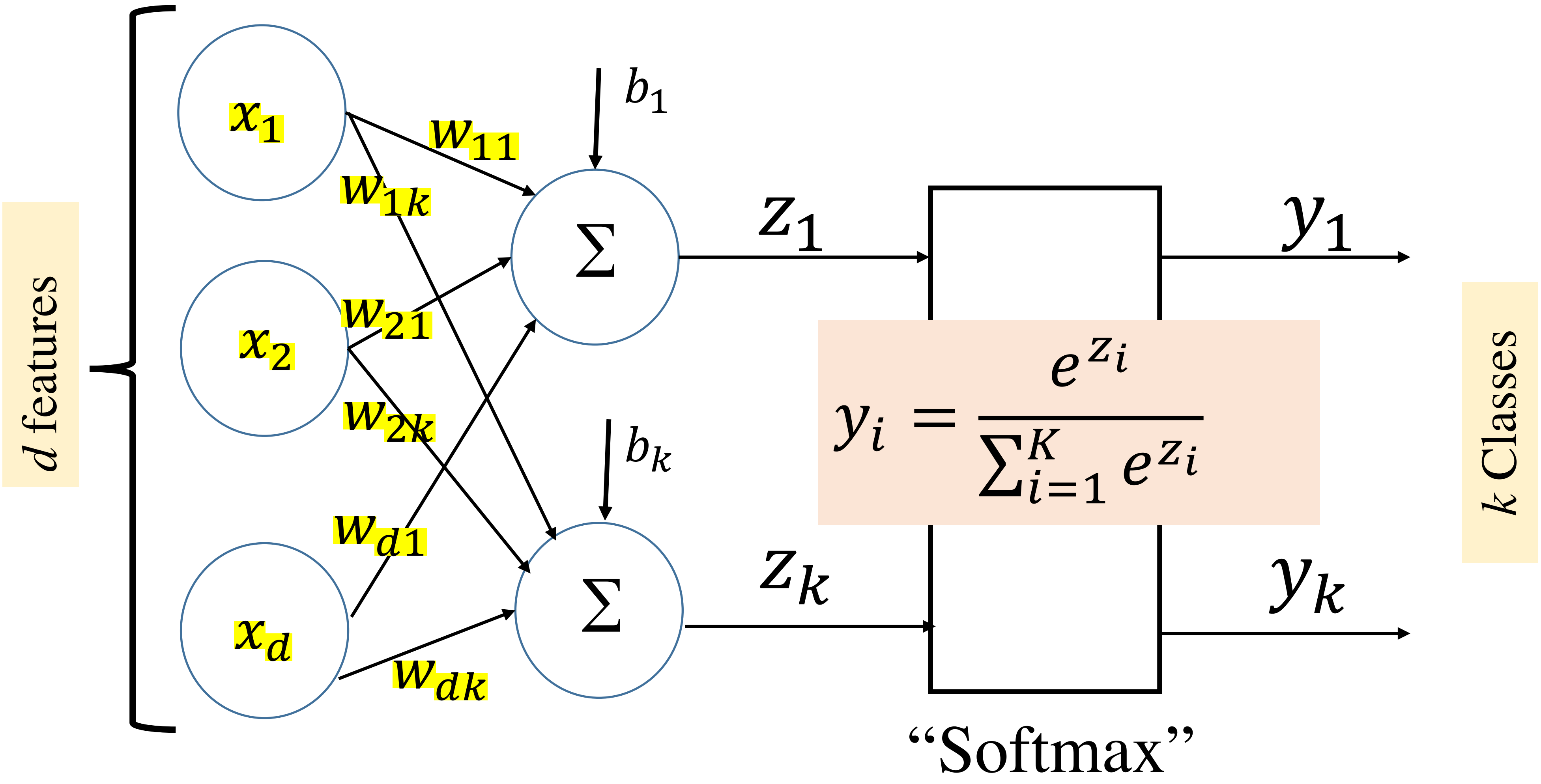
- Tanh



- Rectified Linear Unit (Relu)

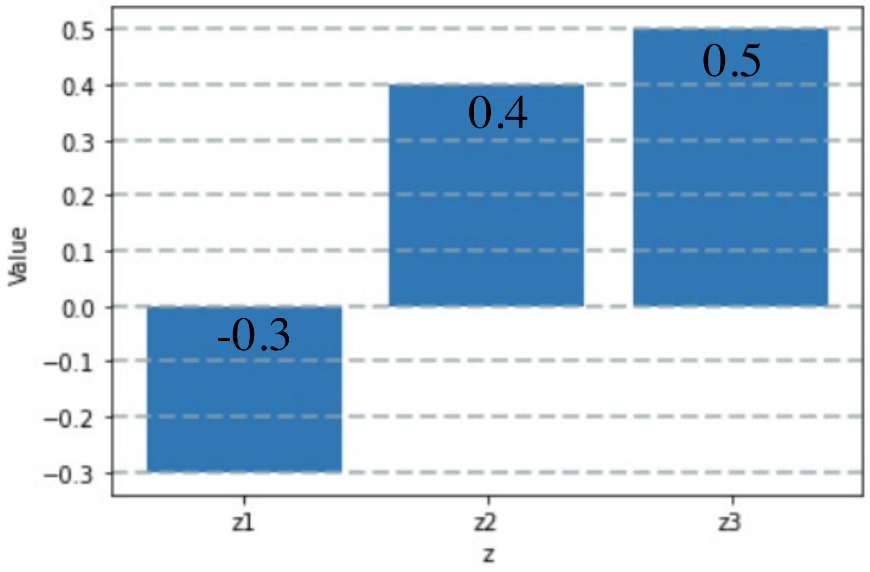


Multi-Class Classification

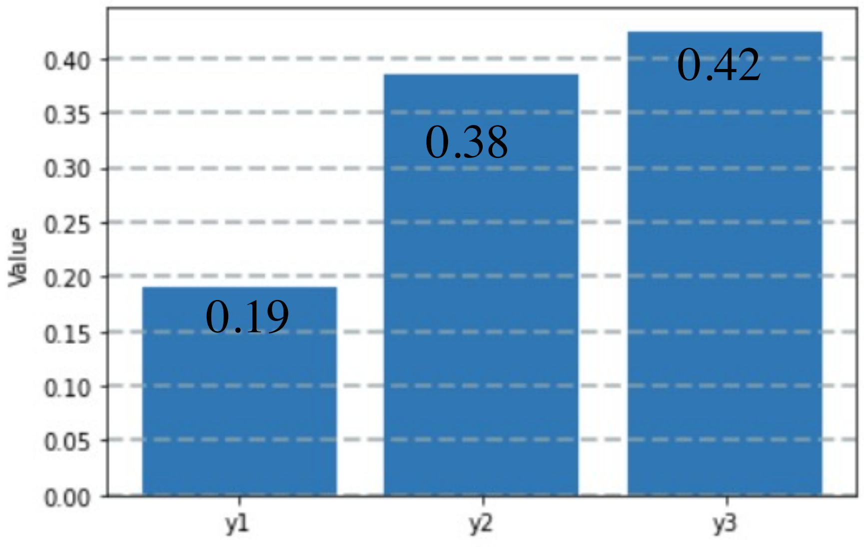


Understanding Softmax

$$y_i = \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_i}}$$



The z vector has real-valued entries, including both positive and negative values

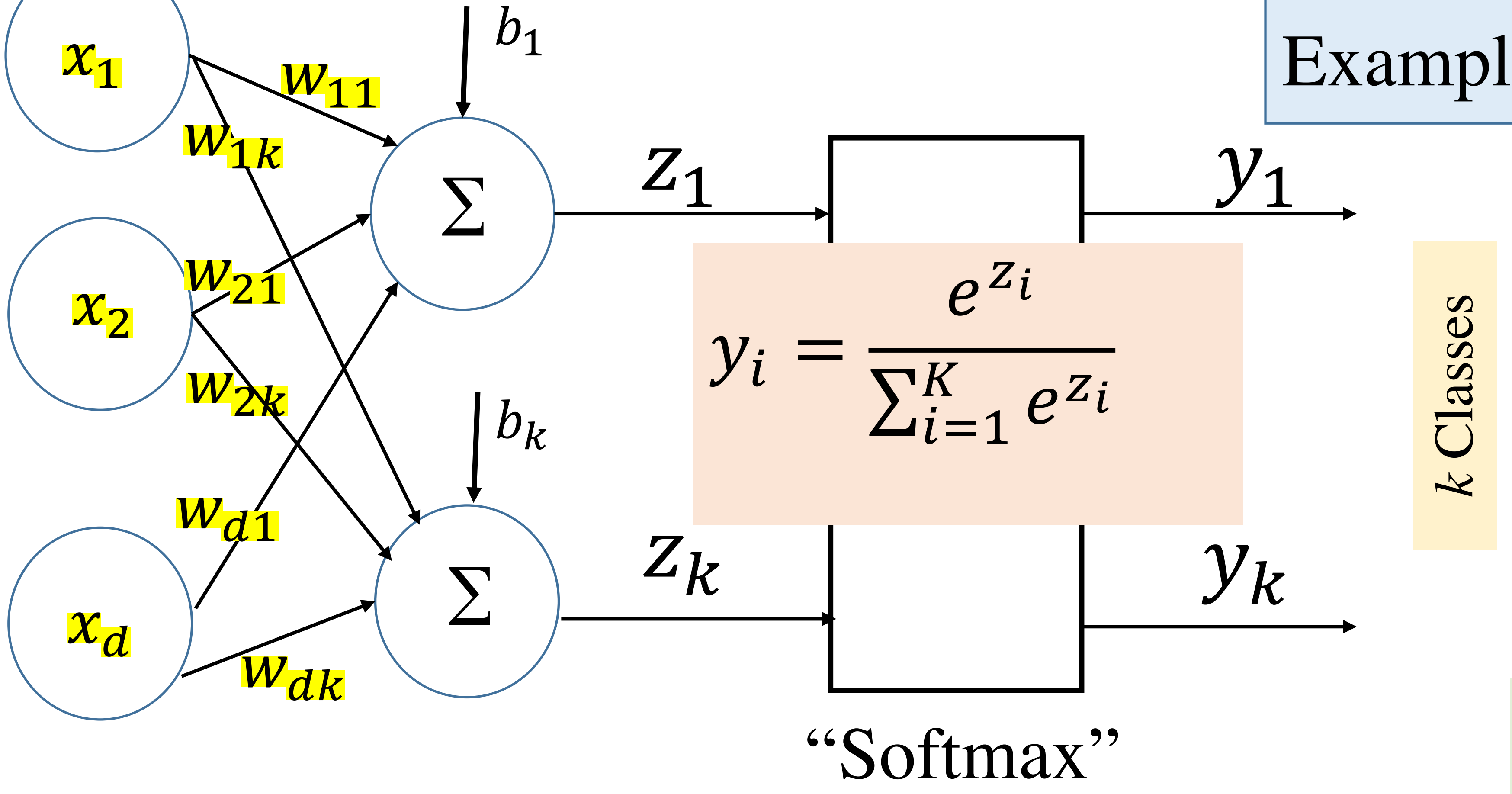


The y vector has:

- Positive real-valued entries
- Values sum to 1
- Preserves ordering of z

Can be viewed as a probability distribution over y!

Cross-entropy Loss



$$y_i = \frac{e^{z_i}}{\sum_{i=1}^{K} e^{z_i}}$$

Labels: $[y_1 \ldots . y_k]$

"One-hot Coded"

Example: [0 0 .. 1…0]

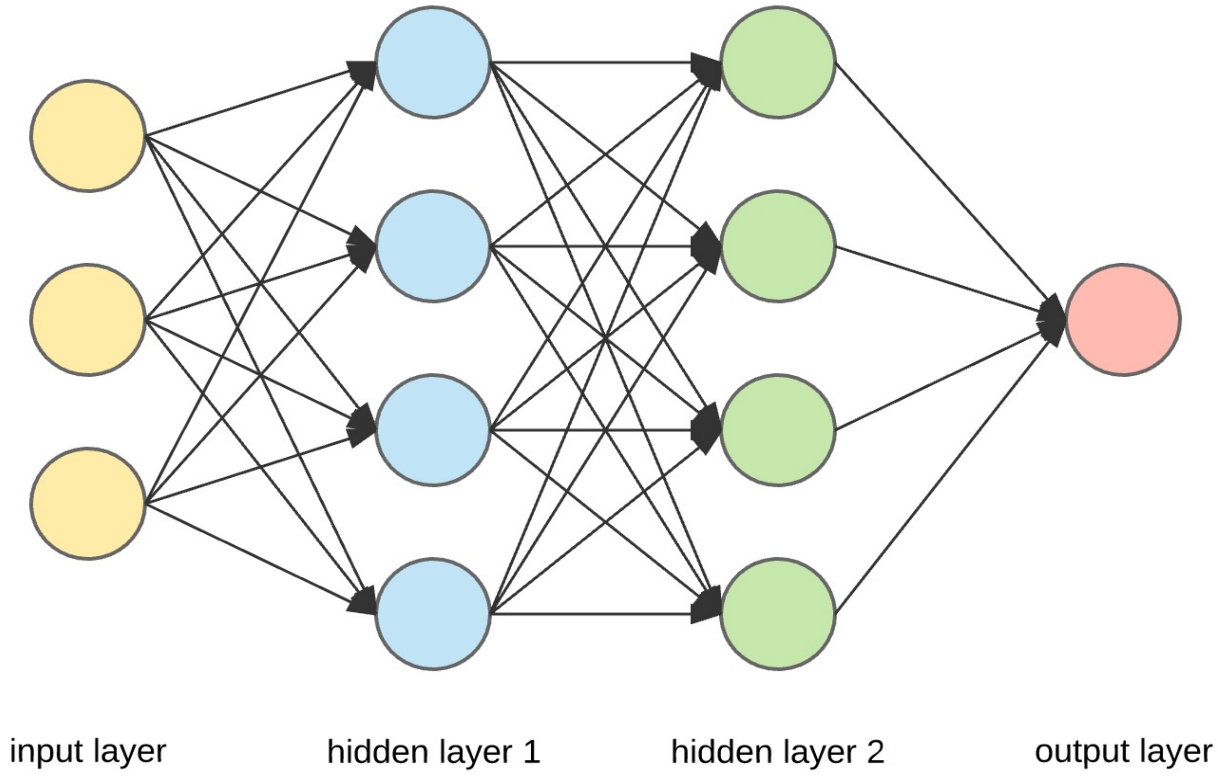
Cross-Entropy Loss

$$\sum_{i=1}^{K} -y_i log(1 - \hat{y}_i)$$

Summary

- Linear Regression
- Logistic Regression
  - MLE
  - Cross Entropy Loss
- Gradient Descent
- Backpropagation
- Neural Networks
  - Activation functions
- Multi-class Neural Networks
  - Softmax
  - One-hot encoding

Some Questions



- What happens if we initialize all params to 1?
- What happens if we initialize all params to 0?
- Why do we need non-linearity?
- Why does Relu make training faster?