

# Day 6: Neural Networks

## Summer STEM: Machine Learning

Department of Electrical and Computer Engineering  
NYU Tandon School of Engineering  
Brooklyn, New York

# Outline

1 Review

2 Multiclass Classification

3 Neural Networks

4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

$$u = [u_1, u_2, \dots, u_D]^T$$

$L_{\infty}$  norm  $\max_i |u_i|$

$L^0$  norm the number of non-zero elements in the vector

# Machine Learning Problem Pipeline

- 1** Formulate the problem: regression, classification, or others?
- 2** Gather and visualize the data
- 3** Design the model and the loss function
- 4** Train your model
  - (a) Perform feature engineering
  - (b) Construct the design matrix
  - (c) Choose regularization techniques
  - (d) Tune hyper-parameters using a validation set
  - (e) If the performance is not satisfactory, go back to step (a).
- 5** Evaluate the model on a test set

# Outline

1 Review

2 Multiclass Classification

3 Neural Networks

4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

# Multiclass Classification

$$\begin{aligned} \mathbf{x} &= [x_1 \ x_2]^T \\ \underline{\phi(x)} &= [1 \ x_1 \ x_2]^T \end{aligned}$$

$$\mathbf{w}^T \underline{\phi(x)} = w_0 + w_1 x_1 + w_2 x_2$$

- Previous model:  $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$

- Representing Multiple Classes:

- One-hot / 1-of-K vectors, ex : 4 Class
- Class 1 :  $\mathbf{y} = [1, 0, 0, 0]$
- Class 2 :  $\mathbf{y} = [0, 1, 0, 0]$
- Class 3 :  $\mathbf{y} = [0, 0, 1, 0]$
- Class 4 :  $\mathbf{y} = [0, 0, 0, 1]$

$\hat{y}$  should be a  
vector of size 4

$\hat{y}$  is no longer the predicted label  
it is just  $\hat{y} = f(\mathbf{x})$  the output of the  
model.

$\hat{y} = [\underbrace{0.6, 0.2, 0.1, 0.1}]$  assign to the class  
with the highest value

# Multiclass Classification

$$\mathbf{w}^T \phi(\mathbf{x})$$

- Previous model:  $f(\mathbf{x}) = \sigma(\mathbf{w}^T \phi(\mathbf{x}))$  gives a scalar
- Representing Multiple Classes:  $\phi(\mathbf{x})$  has D elements
  - One-hot / 1-of-K vectors, ex : 4 Class
  - Class 1 :  $\mathbf{y} = [1, 0, 0, 0]^T$
  - Class 2 :  $\mathbf{y} = [0, 1, 0, 0]^T$
  - Class 3 :  $\mathbf{y} = [0, 0, 1, 0]^T$
  - Class 4 :  $\mathbf{y} = [0, 0, 0, 1]^T$
- Multiple outputs:  $f(\mathbf{x}) = \text{softmax}(W^T \phi(\mathbf{x}))$
- Shape of  $W^T \phi(\mathbf{x})$  :  $(K, 1) = (K, D) \times (D, 1)$  What would be the shape of  $W^T$  ?
- $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$

$$W^T \phi(\mathbf{x})$$

$$( \text{design mat. } X = \begin{bmatrix} \phi^T(x_1) \\ \phi^T(x_2) \\ \vdots \\ \phi^T(x_N) \end{bmatrix} )$$

$$\text{Ex} \quad x_i = \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} \end{bmatrix}$$

design matrix

if  $\phi(x_i)$  is a matrix of shape  $(3, 1)$

how can we get  $\hat{y}_i$  with a shape of  $(4, 1)$ ?

Step 1.  $z_i = \underbrace{w^\top \phi(x_i)}_{\text{ }} \quad (w^\top, (4, 3))$

Step 2  $\hat{y}_i = \text{softmax}(z_i)$

Assume we have a vector  $z = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_d \end{bmatrix}$

$\text{softmax}(z) \rightarrow$  a vector of the same size as  $z$

$$\begin{bmatrix} \text{softmax}(z)_1 \\ \text{softmax}(z)_2 \\ \vdots \\ \text{softmax}(z)_d \end{bmatrix} \quad \text{where } \text{softmax}(z)_k = \frac{e^{z_k}}{\sum_{j=1}^d e^{z_j}}$$

$\left( e \approx 2.718 \right)$   
 $\text{np.exp}(x) = e^x$

# Multiclass Classification

- Multiple outputs:  $f(\mathbf{x}) = \text{softmax}(\mathbf{z})$  with  $\mathbf{z} = \underbrace{\mathbf{W}^T \phi(\mathbf{x})}_{\mathbf{z}}$

$$\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$$

- Softmax example: If  $\mathbf{z} = \begin{bmatrix} -1 \\ 2 \\ 1 \\ -4 \end{bmatrix}$  then,

$\text{sigmoid}(\mathbf{z}) \rightarrow p \quad \text{positive}$

$1-p \quad \text{negative}$

$$\text{softmax} \left( \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \right) \quad \text{softmax}(\mathbf{z}) =$$

$= \begin{bmatrix} p \\ 1-p \end{bmatrix} \quad \text{positive}$   
 $\quad \quad \quad \quad \quad \text{negative}$

$$\begin{bmatrix} e^{-1} \\ e^{-1}+e^2+e^1+e^{-4} \end{bmatrix}$$

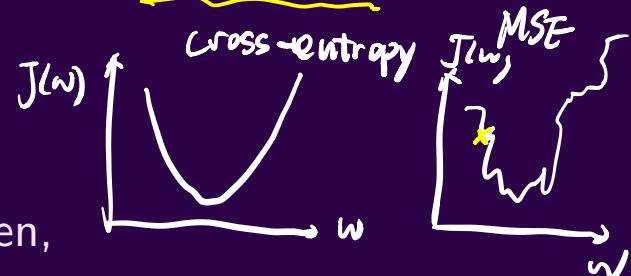
$$\begin{bmatrix} e^2 \\ e^{-1}+e^2+e^1+e^{-4} \end{bmatrix}$$

$$\begin{bmatrix} e^1 \\ e^{-1}+e^2+e^1+e^{-4} \end{bmatrix}$$

$$\begin{bmatrix} e^{-4} \\ e^{-1}+e^2+e^1+e^{-4} \end{bmatrix}$$

$$\approx \begin{bmatrix} 0.035 \\ 0.704 \\ 0.259 \\ 0.002 \end{bmatrix}$$

$$= \hat{y}$$



# Cross-entropy

- Multiple outputs:  $\hat{\mathbf{y}}_i = \text{softmax}(W^T \phi(\mathbf{x}_i))$
- Cross-Entropy:  $J(W) = - \sum_{i=1}^N \sum_{k=1}^K \mathbf{y}_{ik} \log(\hat{\mathbf{y}}_{ik})$
- Example :  $K = 4$

If,  $\mathbf{y}_i = [0, 0, 1, 0]$  then,  $\sum_{k=1}^K \mathbf{y}_{ik} \log(\hat{\mathbf{y}}_{ik}) = \underbrace{\log(\hat{\mathbf{y}}_{i3})}_{\log(0.259)}$

$$\begin{aligned} y_{i1} &= 0 & y_{i1} \log(y_{i1}) &= 0 \\ y_{i2} &= 0 & + y_{i2} \log(y_{i2}) &= 0 \\ y_{i3} &= 1 & + y_{i3} \log(\hat{y}_{i3}) &= \underbrace{\log(\hat{y}_{i3})}_{\log(0.259)} \\ y_{i4} &= 0 & + y_{i4} \log(\hat{y}_{i4}) &= 0 \end{aligned}$$

# Outline

1 Review

2 Multiclass Classification

3 Neural Networks

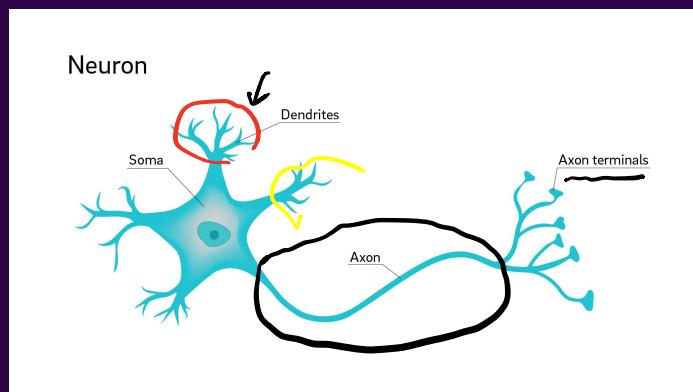
4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

# Biological Neuron

Neural Spike model



Source: David Baillot/UC San Diego



- message
  - electrical signals
  - dendrites
- 
- A graph showing Voltage (mV) on the vertical axis versus time on the horizontal axis. The membrane potential (yellow line) starts at -70 mV and fires three action potentials, each reaching a peak above the 30 mV threshold. The third spike is highlighted in red. Handwritten text next to the graph states: "'analyze' : 'process'?"

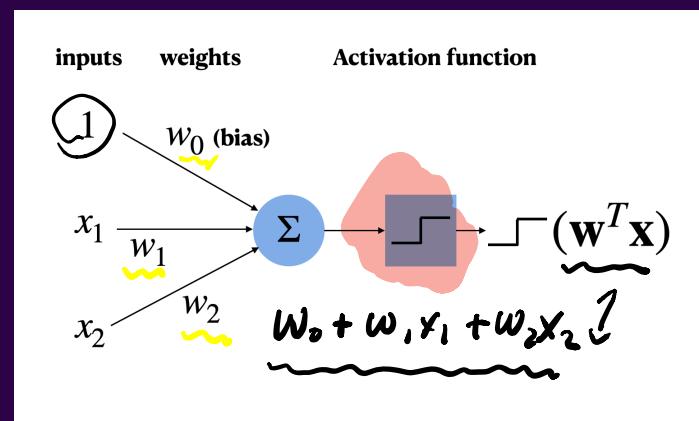
# Mathematical Neuron: Perceptron

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

Biological Neuron:

- A neuron can receive electrochemical signals from other neurons;
- A neuron fires once its accumulated electric charge passes a certain threshold.
- Neurons that fire together wire together.

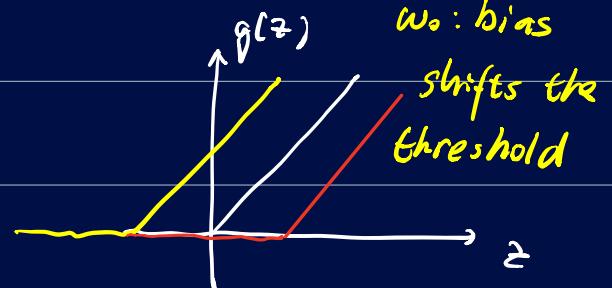
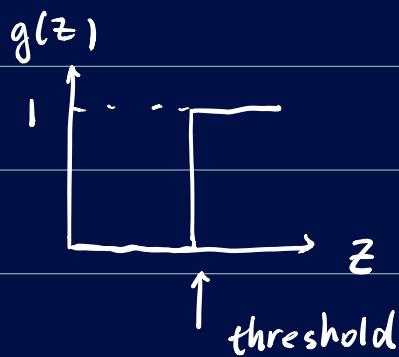
Mathematical Neuron  
(Perceptron)



Activation functions

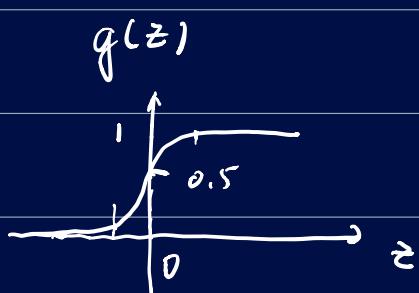
$$g(z)$$

$$z = \underline{w^T x}$$

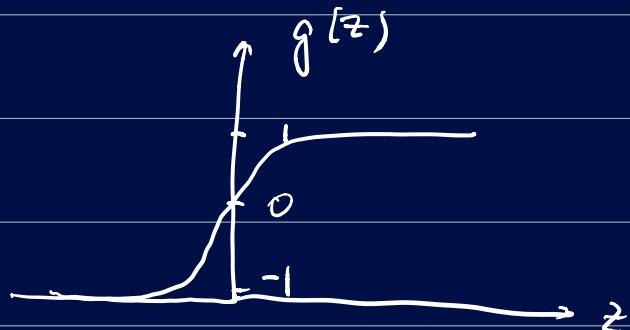


Step function

$$\text{ReLU: } g(z) = \max(0, z)$$



Sigmoid



Tanh

perceptron

$$\approx z$$

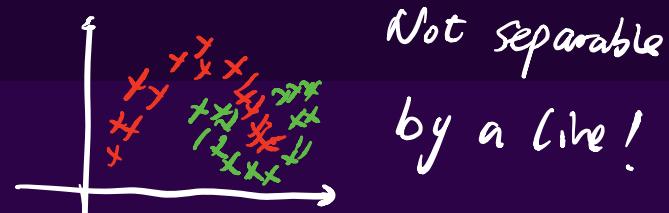
$$f(x) = g(\underline{w^T x})$$

What if I use Sigmoid?

$$f(x) = \text{Sigmoid}(w^T x)$$

logistic regression

# Relation to Logistic Regression



What if we use the sigmoid function as the activation?

$$f(\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x})$$

Decision boundary is a line: how is this supposed to revolutionize machine learning?

Decision boundary

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\begin{aligned} \text{Sigmoid } (\mathbf{w}^T \mathbf{x}) &= 0.5 \\ \Rightarrow \quad \mathbf{w}^T \mathbf{x} &= 0 \\ \Rightarrow \quad w_0 + w_1 x_1 + w_2 x_2 &= 0 \end{aligned}$$

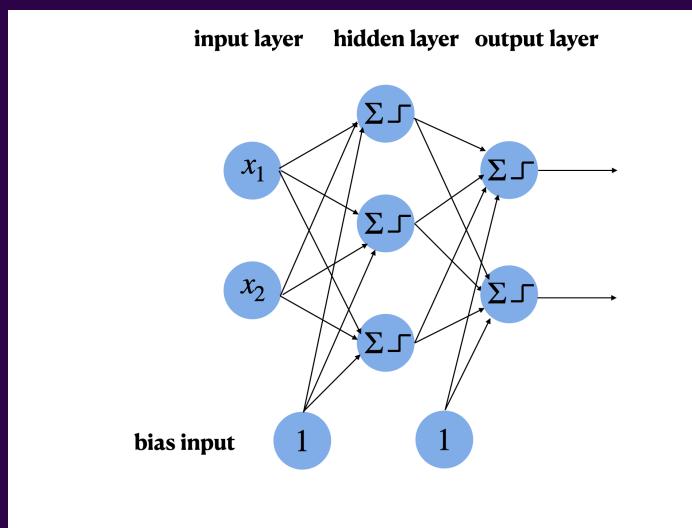
# Multi-layer Perceptron (MLP)

*We need more neurons!*

We need more neurons and we need to connect them together!

- Many ways to do that...
- Today: multi-layer perceptron/fully connected feed-forward network.

# MLP Example



- What is the shape of the input and output?
- How many parameters does this model have? 17
- What activation function would you use for the output layer? Why? *depends on the task*

what are the model parameters for a NN?

- weights, bias

Hyper - parameters ?

- # neurons per layer
- # layers
- the type of activation function

# More about MLPs

- Many choices for the activation function: Sigmoid, Tanh, ReLU, Swish, etc.
- Many choices for the number of hidden layers and the number of neurons per layer.
- MLPs can approximate any continuous function given enough data.
- MLPs can overfit, but we know many effective ways of regularization.

# Exercise: PyTorch Basics

Open `demo_pytorch_basics.ipynb`

# Outline

1 Review

2 Multiclass Classification

3 Neural Networks

4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

# Deep Learning

What does deep learning stand for?

- Deep: Neural network architectures with many hidden layers.
- Learning: Optimizing model parameters given a dataset.

In general, the deeper the model is, the more parameters we need to learn and the more data is needed.

# Large-Scale Machine Learning

For deep learning systems to perform well, large datasets are required

- COCO 330K images
- ImageNet 14 million images

Challenges:

- Memory limitation: GeForce RTX 2080 Ti has 11 GB memory, while ImageNet is about 300 GB.
- Computation: Calculating gradients for the whole dataset is computationally expensive (slow), and we need to do this many times.

# Stochastic Gradient Descent

Idea: Instead of calculating the gradients from the whole dataset, do it only on a subset.

- Randomly select  $B$  samples from the dataset
- The loss for this subset

$$\tilde{J}(\mathbf{w}) = \frac{1}{B} \sum_{i=1}^B \|y - \hat{y}_i\|^2$$

- Update Rule

*Repeat*{

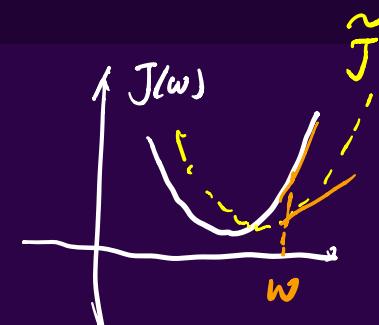
$$\mathbf{w}_{new} = \mathbf{w} - \alpha \nabla \tilde{J}(\mathbf{w})$$

}

# Yet Another Hyper-Parameter

This gives a noisy gradient

$$\nabla \tilde{J}(\mathbf{w}) = \nabla J(\mathbf{w}) + \underbrace{\epsilon}_{\text{wavy}}$$

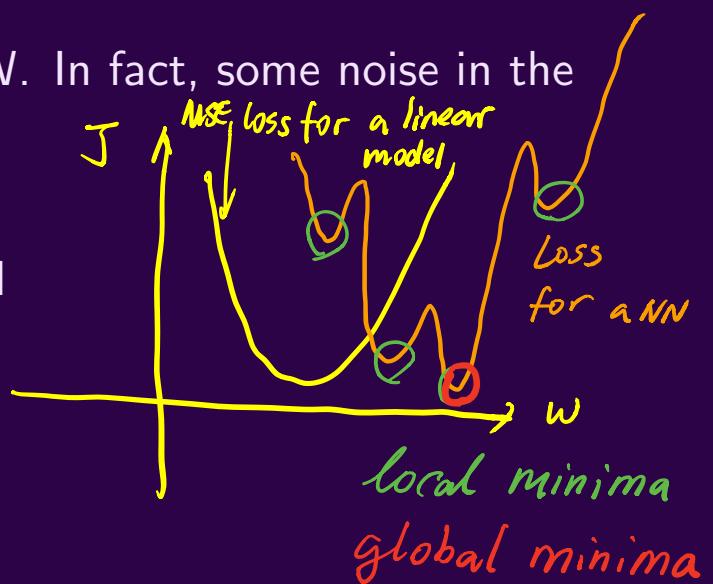
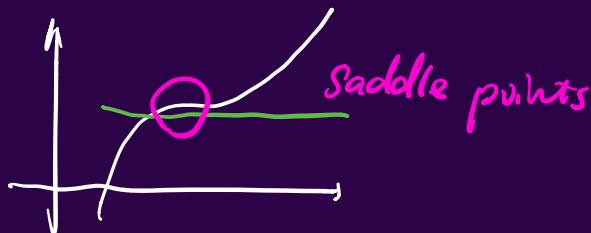


- **SGD:**  $B = 1$ , gives very noisy gradients
- **(batch) GD:**  $B = N$ ,  $\epsilon = 0$ , expensive to compute
- **Mini-batch GD:** Pick a small  $B$ , typical values are 32, 64, rarely more than 128 for image inputs

# Some Noise Helps

Even if we can, we rarely set  $B = N$ . In fact, some noise in the gradients might help to

- escape from local minima,
- escape from saddle points, and
- improve generalization.



# Outline

1 Review

2 Multiclass Classification

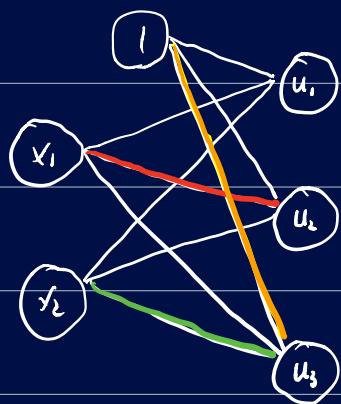
3 Neural Networks

4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

## Recap



weights

$$w_{ij} : x_j \rightarrow u_i$$

bias

$$w_{i0} : l \rightarrow u_i$$

$$w_{21}$$

$$x_1 \rightarrow u_2$$

$$w_{32}$$

$$x_2 \rightarrow u_3$$

$$w_{30}$$

$$l \rightarrow u_3$$

$$u_1 = l \cdot w_{10} + w_{11} x_1 + w_{12} x_2$$

$$u_2 = l \cdot w_{20} + w_{21} x_1 + w_{22} x_2$$

$$u_3 = l \cdot w_{30} + w_{31} x_1 + w_{32} x_2$$

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \\ w_{30} & w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} l \\ x_1 \\ x_2 \end{bmatrix}$$

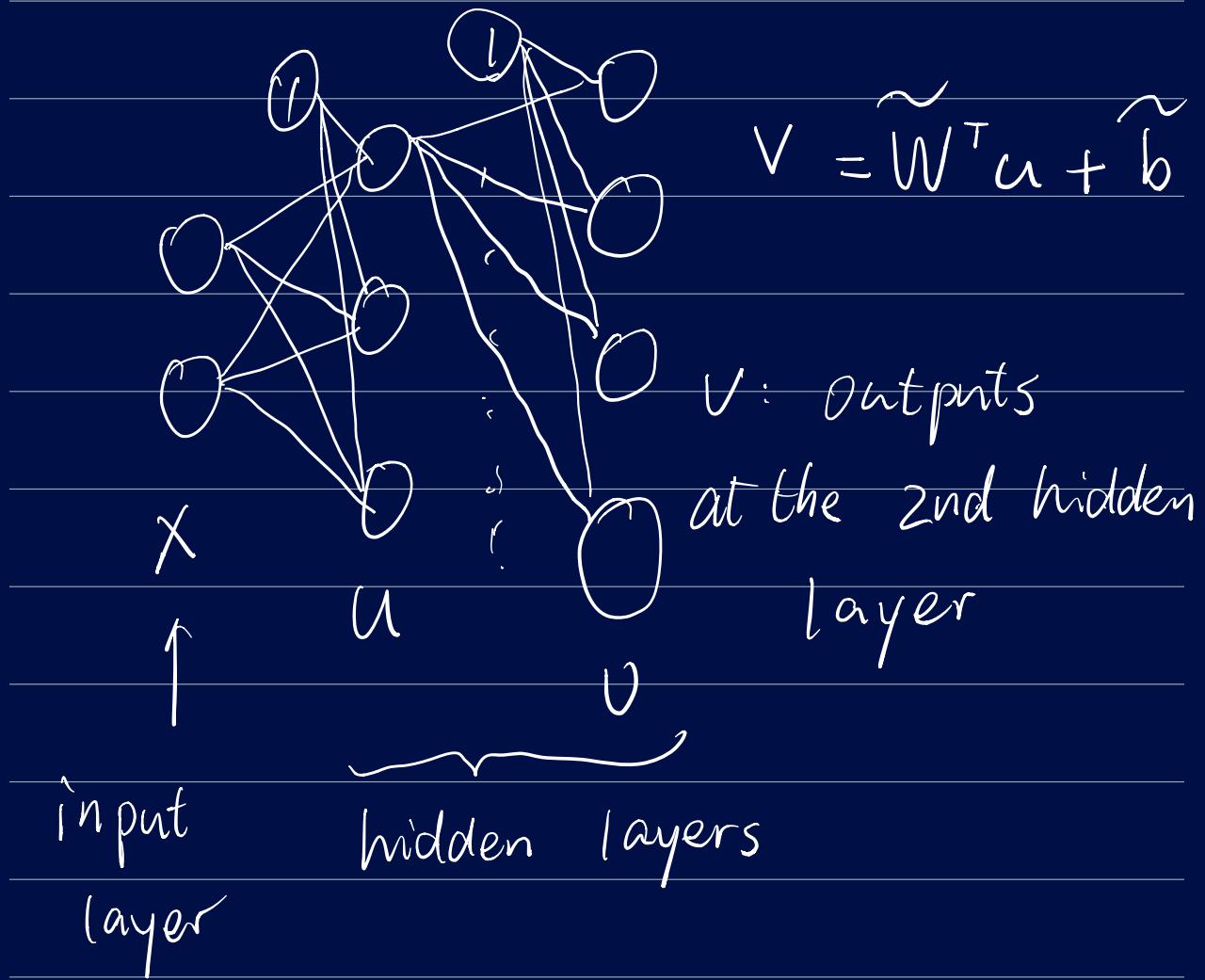
$u$

$W^T$

$b$

$$= \underbrace{\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}}_{\text{weights}} \underbrace{\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix}}_{\text{inputs}} + \underbrace{\begin{bmatrix} w_{10} \\ w_{20} \\ w_{30} \end{bmatrix}}_{\text{bias}}$$

$$u = W^T x + b$$



$$a = W^T x + b$$

$$v = \underbrace{\tilde{W}^T}_{\sim} u + \underbrace{\tilde{b}}_{\sim}$$

$$v = \tilde{W}^T (W^T x + b) + \tilde{b}$$

$$v = \underbrace{\tilde{W}^T W^T}_{\sim} x + \left( \underbrace{\tilde{W}^T b}_{\sim} + \underbrace{\tilde{b}}_{\sim} \right)$$

$$(4,1) \quad (4,2) \quad (4,1) \quad (4,1)$$

$$v = M x + m$$

This is a linear model

What is a local minimum?

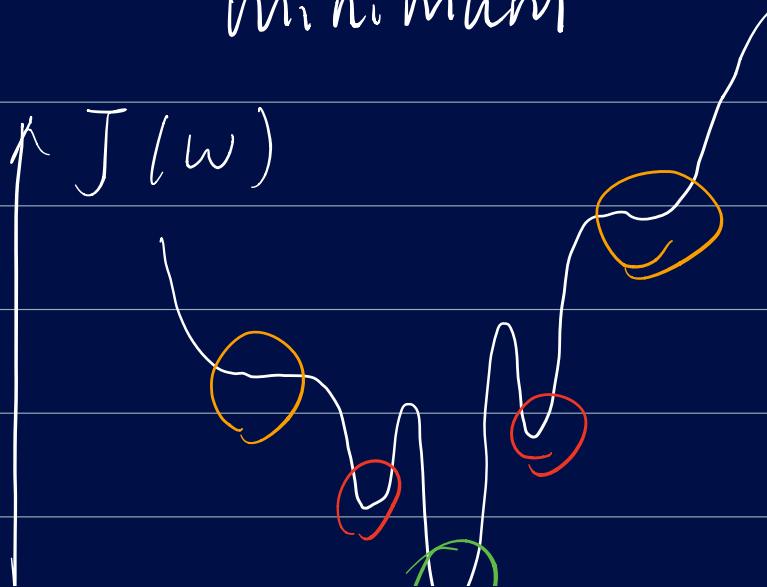
- $\nabla J(w) = 0$
- not a global minimum

- For scalar-valued  $w$

$\nabla J(w)$  flips the sign from  
- to +

Saddle point

- $\nabla J(w) = 0$
- not a global / local minimum



Y

## SGD vs. GD

SGD:  $B \leq N$

$$\tilde{J}(\omega) = \frac{1}{B} \sum_{i=1}^B \|y_i - \hat{y}_i\|^2$$

$$\omega_{\text{new}} = \omega - \alpha \nabla \tilde{J}(\omega)$$

$$\tilde{J}(\omega) \rightarrow \boxed{\quad} \rightarrow \nabla \tilde{J}(\omega)$$

GD

$$J(\omega) = \frac{1}{N} \sum_{i=1}^N \|y_i - \hat{y}_i\|^2$$

# Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.

# Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters

# Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters

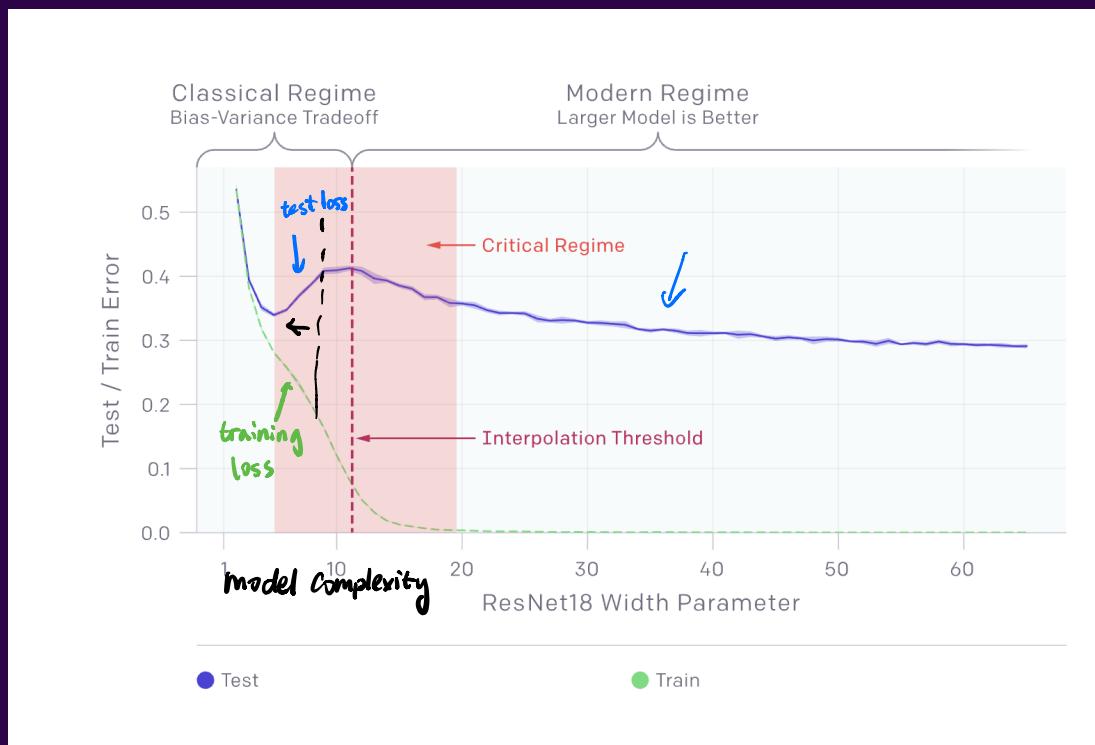
# Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters
- Conventional wisdom: Such models overfit.

# Overparameterized Models

- Modern deep learning models are heavily overparameterized, i.e. the number of learnable parameters is much larger than the number of the training samples.
  - ResNet: State-of-the-art vision model, 10-60 million parameters
  - GPT-3: State-of-the-art language model, 175 billion parameters
- Conventional wisdom: Such models overfit.
- It is not the case in practice!

# Double Descent Curve



Source: OpenAI

# Outline

1 Review

2 Multiclass Classification

3 Neural Networks

4 Stochastic Gradient Descent

5 Overparameterization

6 Lab

# Let's solve the mini-project with MLPs!

Open `lab_mlp_fish_market.ipynb`