Day 6: Neural Networks

Summer STEM: Machine Learning

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

Outline

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

L-infinity 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

Multiclass Classificaiton

$x = [x_1 \; x_2]^T$

$\phi(x) = [1 \; x_1 \; x_2]^T$

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

- 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 $\hat{y} = f(x)$ the output of the model.

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

NYU TANDON SCHOOL OF ENGINEERING

Multiclass Classificaiton

- 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 : $\mathbf{y} = [1, 0, 0, 0]^T$
  - Class 2 : $\mathbf{y} = [0, 1, 0, 0]^T$
  - Class 3 : $\mathbf{y} = [0, 0, 1, 0]^T$
  - Class 4 : $\mathbf{y} = [0, 0, 1, 0]^T$
- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(W^T \phi(\mathbf{x}))$
- Shape of $W^T \phi(\mathbf{x})$: $(K, 1) = (K, D) \times (D, 1)$
- $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$

$w^T \phi(x)$ gives a scalar

$\phi(x)$ has D elements

$W^T \phi(x)$

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

What would be the shape of W^T?

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

$\phi(x_i) = \begin{bmatrix} 1 \\ x_{i1} \\ x_{i2} \end{bmatrix}$

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. $z_i = \underbrace{W^T \phi(x_i)}$ $\quad (W^T, (4, 3))$

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

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

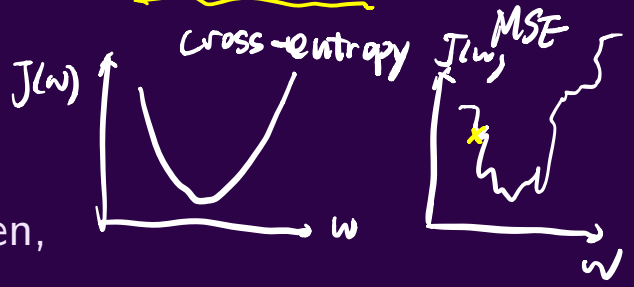
Softmax (z) $\rightarrow$ a vector of the same size as z

$\begin{bmatrix} \text{softmax}(z)_1 \\ \text{softmax}(z)_2 \\ \vdots \\ \text{softmax}(z)_D \end{bmatrix}$ where $\text{softmax}(z)_k = \dfrac{e^{z_k}}{\sum_{j=1}^{D} e^{z_j}}$

$(e \approx 2.718$
$np.exp(x) = e^x)$

Multiclass Classificaiton

- Multiple outputs: $f(\mathbf{x}) = \text{softmax}(\mathbf{z})$ with $\mathbf{z} = W^T \phi(\mathbf{x})$
- $\text{softmax}(\mathbf{z})_k = \frac{e^{z_k}}{\sum_j e^{z_j}}$
- Softmax example: If $\mathbf{z} = \begin{bmatrix} -1 \\ 2 \\ 1 \\ -4 \end{bmatrix}$ then,



sigmoid(z) → p positive
1−p negative

$$\text{softmax}\left(\begin{bmatrix} z_1 \\ z_2 \end{bmatrix}\right) = \begin{bmatrix} p \\ 1-p \end{bmatrix} \quad \begin{matrix} \text{positive} \\ \text{negative} \end{matrix}$$

$$\text{softmax}(z) = \begin{bmatrix} \frac{e^{-1}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{2}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{1}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \\ \frac{e^{-4}}{e^{-1}+e^{2}+e^{1}+e^{-4}} \end{bmatrix} \approx \begin{bmatrix} 0.035 \\ 0.704 \\ 0.259 \\ 0.002 \end{bmatrix} = \hat{y}$$

Cross-entropy

- Multiple outputs: $\hat{\mathbf{y}}_\mathbf{i} = \text{softmax}(W^T \phi(\mathbf{x}_i))$
- 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, $\sum_{k=1}^{K} \mathbf{y}_{ik} log(\hat{\mathbf{y}}_{ik}) = log(\hat{\mathbf{y}}_{i3})$

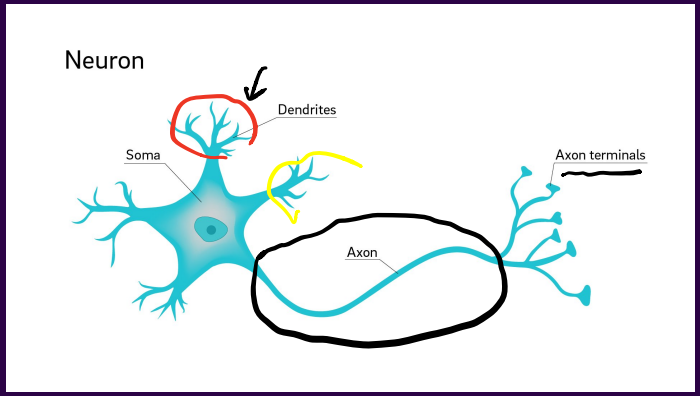
$log(0.259)$

$y_{i1} = 0$
$y_{i2} = 0$
$y_{i3} = 1$
$y_{i4} = 0$

$y_{i1} log(\hat{y}_{i1})$ → 0
$+ y_{i2} log(\hat{y}_{i2})$ → 0
$+ y_{i3} log(\hat{y}_{i3}) + y_{i4} log(\hat{y}_{i4})$ → 0

Outline

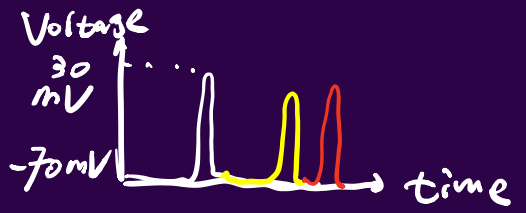
Biological Neuron



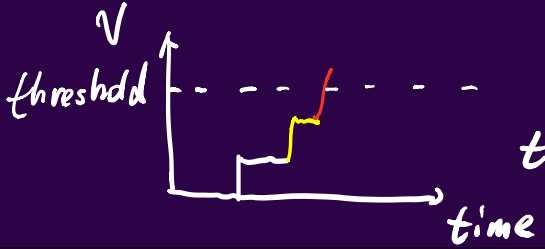
Source: David Baillot/UC San Diego

Neural Spike model.

- message
- electrical signals
- dendrites



- "analyze"? "process"?



Once the charge passes the threshold, it will "fire"

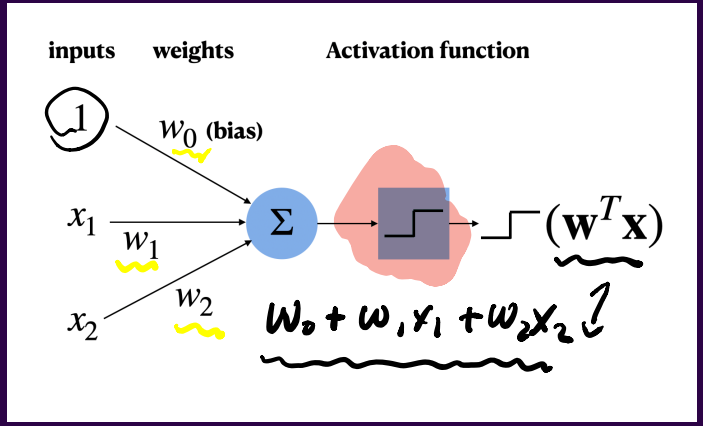
Mathematical Neuron: Perceptron

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

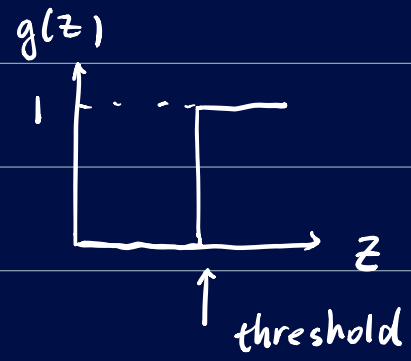
Biological Neuron:

- 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.

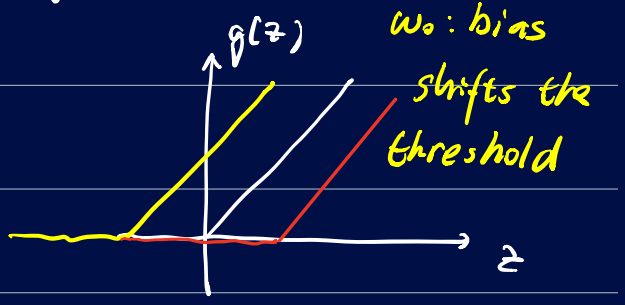
Mathematical Neuron (Perceptron)



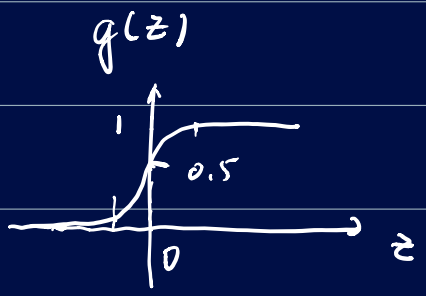
Activation functions $g(z)$ $z = w^T x$



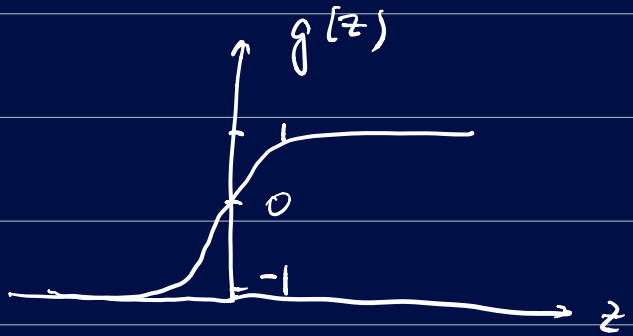
Step function



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



Sigmoid



Tanh

perceptron $= z$

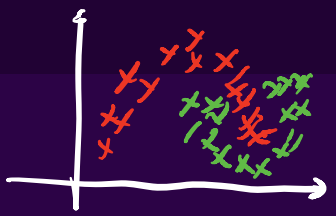
$$f(x) = g(w^T x)$$

What if I use sigmoid ?

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

logistic regression

Relation to Logistic Regression



Not separable by a line!

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?

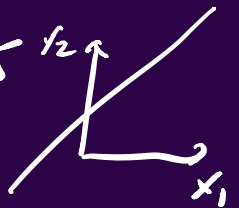
Decision boundary

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

$$\text{sigmoid}(w^T x) = 0.5$$

$$\Rightarrow \quad w^T x = 0$$

$$\Rightarrow \quad w_0 + w_1 x_1 + w_2 x_2 = 0$$



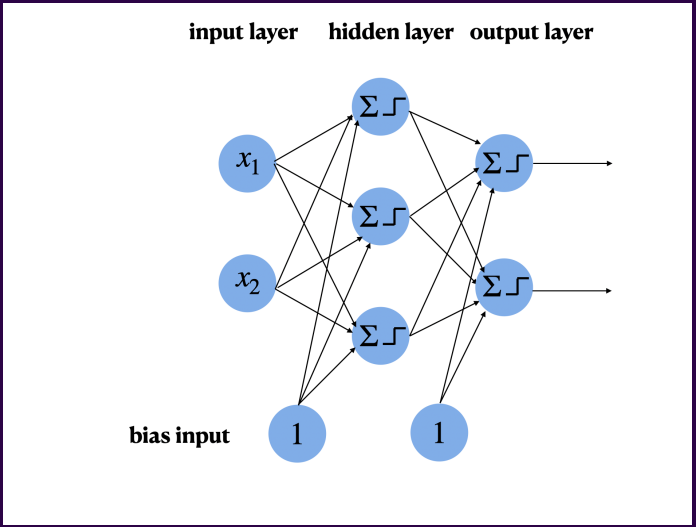
Multi-layer Perceptron (MLP)

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

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

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

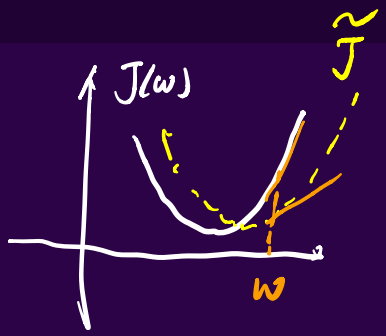
- Update Rule

  Repeat{

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

  }

Yet Another Hyper-Parameter



This gives a noisy gradient

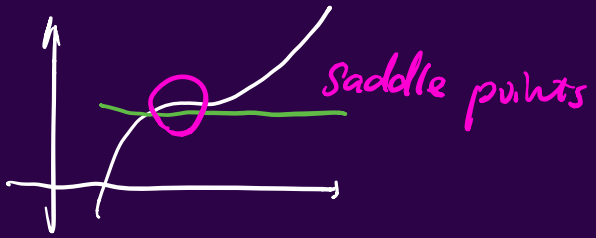
$$\nabla \tilde{J}(\mathbf{w}) = \nabla J(\mathbf{w}) + \epsilon$$

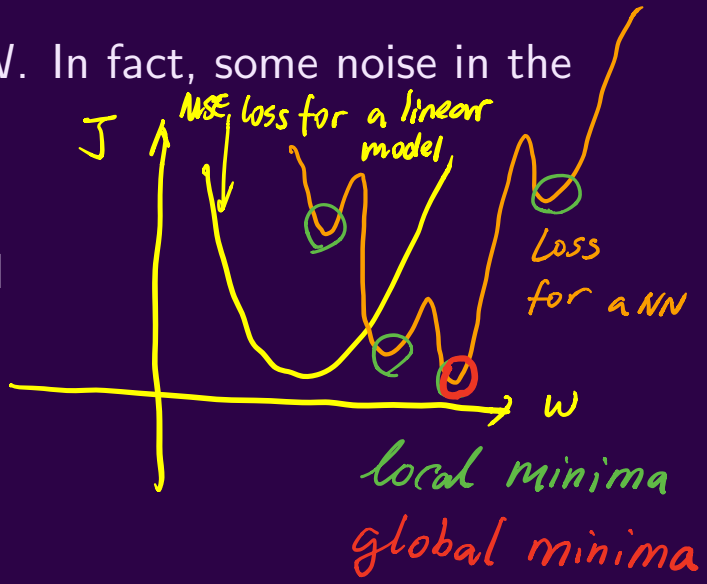
- **SGD**: $B = 1$, gives very noisy gradients
- **(batch) GD**: $B = N$, $\epsilon = 0$, expensive to compute
- **Mini-batch GD**: Pick a small B, typical values are $32, 64$, rarely more than 128 for image inputs

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

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

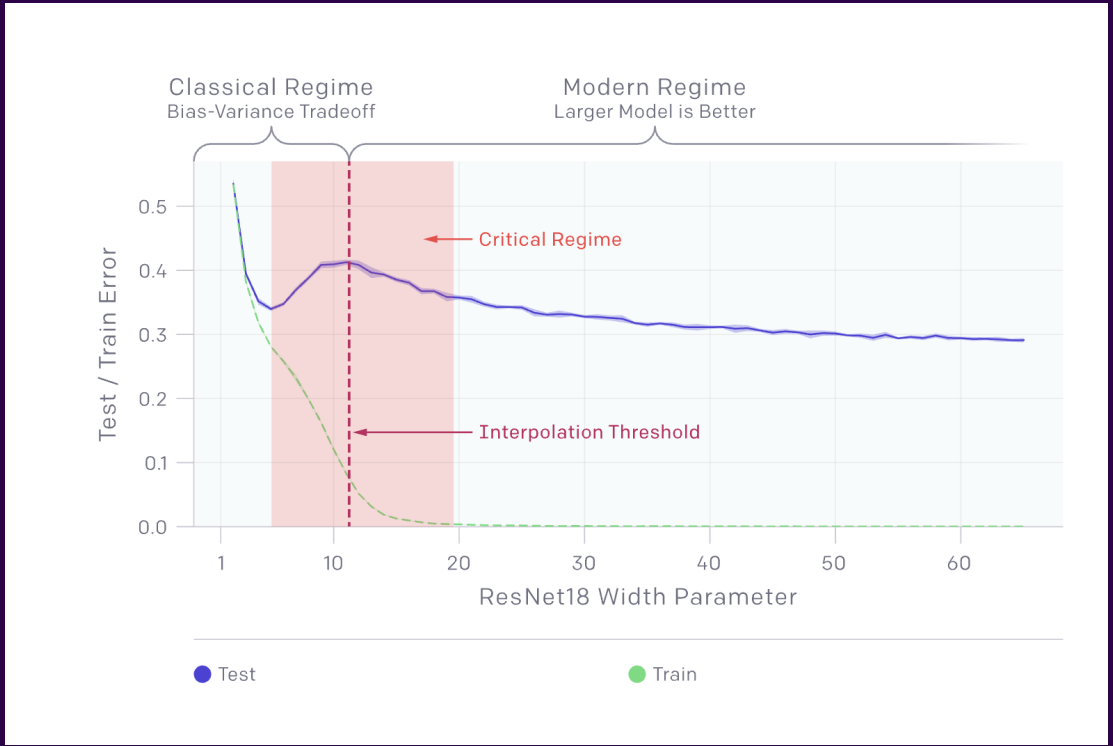
Overparameterized Models

- 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

Let's solve the mini-project with MLPs!

Open `lab_mlp_fish_market.ipynb`