AI: Strategy + Marketing (MGT 853)

Deep and Reinforcement Learning (Session 3)

Vineet Kumar

Yale School of Management

Spring 2024

Agenda for Today's Session

- Deep Learning

Agenda for Today's Session

- Deep Learning
  - What is DL? What are the types of DL?

Agenda for Today's Session

- Deep Learning
  - What is DL? What are the types of DL?
  - why "deep" needed?

Agenda for Today's Session

- Deep Learning
  - What is DL? What are the types of DL?
  - why "deep" needed?
- Generative Models

Agenda for Today's Session

- Deep Learning
  - What is DL? What are the types of DL?
  - why "deep" needed?
- Generative Models
- Reinforcement Learning

Agenda for Today's Session

- Deep Learning
  - What is DL? What are the types of DL?
  - why "deep" needed?
- Generative Models
- Reinforcement Learning
- In-class Exercise on using (U)nsupervised, (S)upervised and (R)einforcement Learning

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data ("data seen by algo") as well as **test** data ("data *unseen* by algo")

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data ("data seen by algo") as well as **test** data ("data *unseen* by algo")
- More complex models can fit training data better but not necessarily test data

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data (“data seen by algo”) as well as **test** data (“data *unseen* by algo”)
- More complex models can fit training data better but not necessarily test data
- Imagine a learner who memorizes past SAT or GMAT exams.

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data ("data seen by algo") as well as **test** data ("data *unseen* by algo")
- More complex models can fit training data better but not necessarily test data
- Imagine a learner who memorizes past SAT or GMAT exams.
  - Great performance on training data (practice problems), but what about test data (actual exam)?

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data ("data seen by algo") as well as **test** data ("data *unseen* by algo")
- More complex models can fit training data better but not necessarily test data
- Imagine a learner who memorizes past SAT or GMAT exams.
  - Great performance on training data (practice problems), but what about test data (actual exam)?
- Appeal of linear models (e.g. regression) is that it avoids overfitting

What is the Learning Goal?

- We want to learn a good f so that $\boxed{y \approx f(x)}$ both within **training** data ("data seen by algo") as well as **test** data ("data *unseen* by algo")
- More complex models can fit training data better but not necessarily test data
- Imagine a learner who memorizes past SAT or GMAT exams.
  - Great performance on training data (practice problems), but what about test data (actual exam)?
- Appeal of linear models (e.g. regression) is that it avoids overfitting
- Challenge is that the **true function** f can be anything, not necessarily linear

ML Pipeline

ML Pipeline :

0. Identify Data (y, x)
1. Pre-processing
   Centering, Missing Data/Quality, Outliers, Normalization
2. Data Visualization
   Correlation, Distribution, Model
3. Data Splitting — Test
   Training Validation
4. Feature Engineering
5. Model Selection
6. Hyper parameter Selection
   Human Analyst Decides
7. Learning (Training)
   $y = f(x)$
8. Validation
9. Interpretation
10. Testing ...

Model Complexity: How can we $\uparrow$ or $\downarrow$ it?

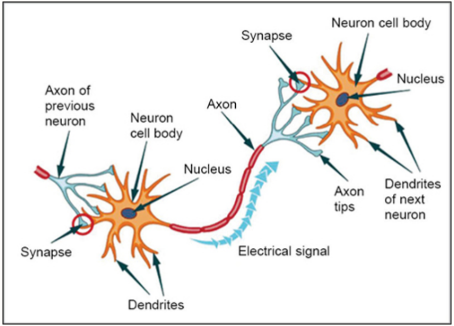
Assume data does not change, same set of y and X variables

- Number of parameters
- Number of layers in DL model
- Quadratic terms
- Interaction terms %item

can each impact model complexity.

Deep Learning

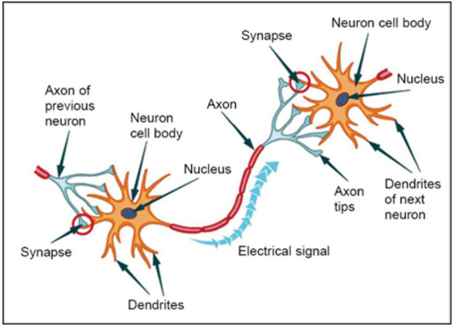
- Foundation of Deep Learning built in the 1950s-1960s



http://www.humanagingcentral.com/brain_page.html

Deep Learning

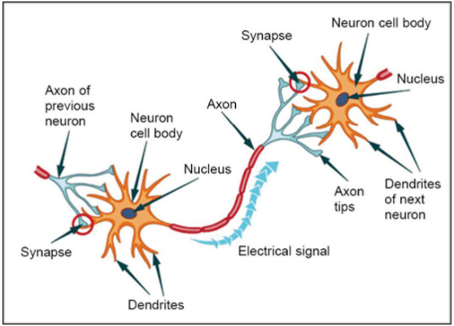
- Foundation of Deep Learning built in the 1950s-1960s
- *Perceptron:* Mathematical Model of a biological neuron (Rosenblatt 1958)



http://www.humanagingcentral.com/brain_page.html

Deep Learning

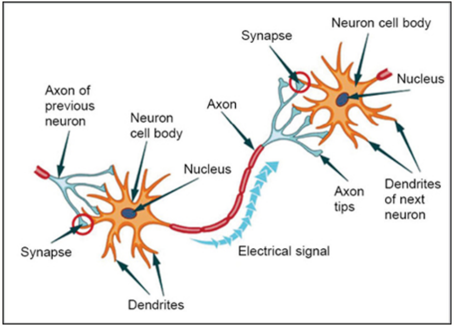
- Foundation of Deep Learning built in the 1950s-1960s
- *Perceptron:* Mathematical Model of a biological neuron (Rosenblatt 1958)
- *Neurons* are part of the nervous system and carry information to and from the brain



http://www.humanagingcentral.com/brain_page.html

Deep Learning

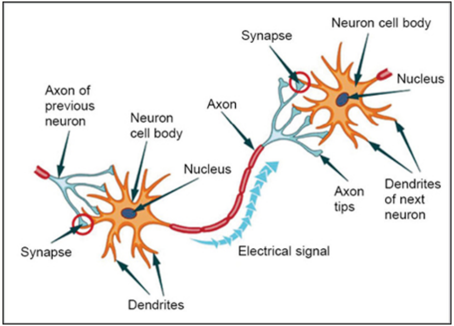
- Foundation of Deep Learning built in the 1950s-1960s
- *Perceptron:* Mathematical Model of a biological neuron (Rosenblatt 1958)
- *Neurons* are part of the nervous system and carry information to and from the brain
- *Soma (cell body)* receives the signal (information) through dendrites



http://www.humanagingcentral.com/brain_page.html

Deep Learning

- Foundation of Deep Learning built in the 1950s-1960s
- *Perceptron:* Mathematical Model of a biological neuron (Rosenblatt 1958)
- *Neurons* are part of the nervous system and carry information to and from the brain
- *Soma (cell body)* receives the signal (information) through dendrites
- *Axon* transmits signal to another neuron

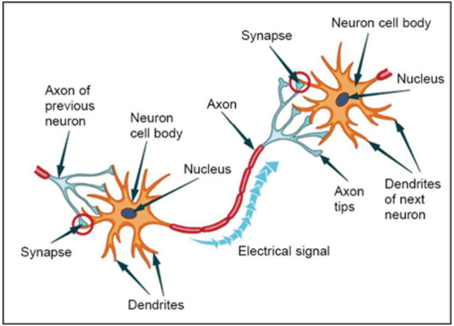


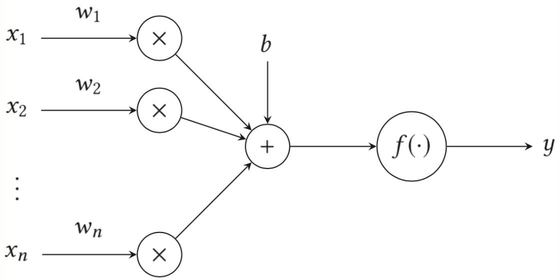
http://www.humanagingcentral.com/brain_page.html

Deep Learning

Connecting Brain Neurons to Artificial Neurons

Perceptrons were motivated by the human brain, which was built up of billions of neurons





Rosenblatt's Perceptron

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

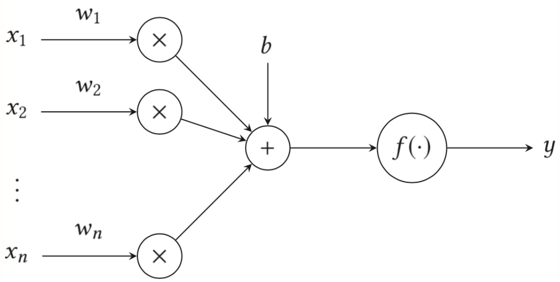
– The New York Times, late 1950s

https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-

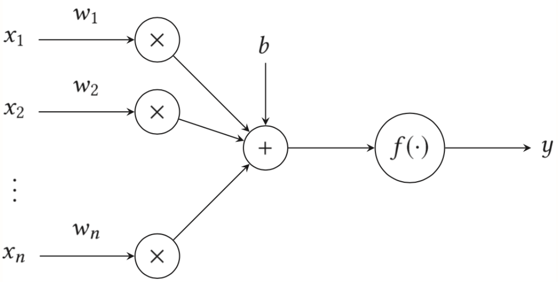
Deep Learning

Perceptron – Basic Element of DL

- Algorithm for Supervised Learning



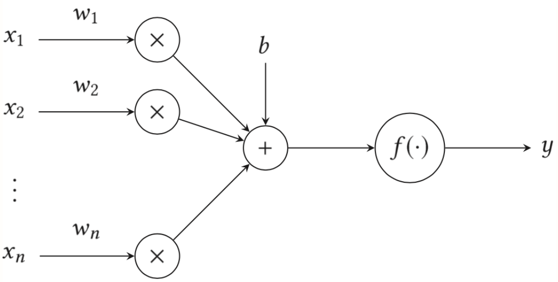
- Algorithm for Supervised Learning
  - Classification: $y = 0$ or 1



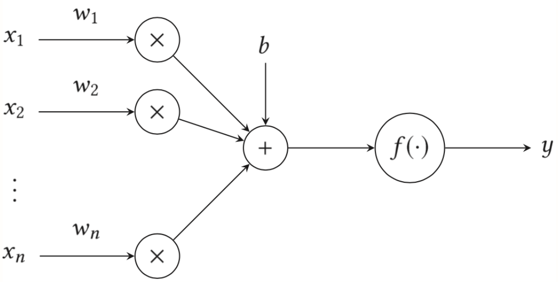
Deep Learning

Perceptron – Basic Element of DL

- Algorithm for Supervised Learning
  - Classification: $y = 0$ or 1
- Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)



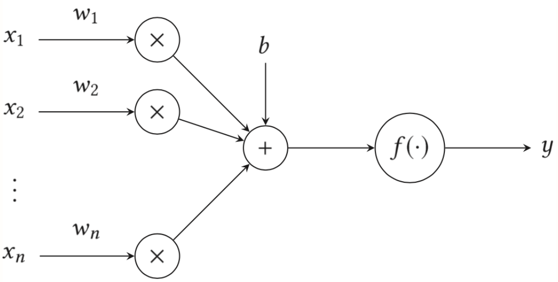
- Algorithm for Supervised Learning
  - Classification: $y = 0$ or 1
- Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)
- Activation function $f(\cdot)$



Deep Learning

Perceptron – Basic Element of DL

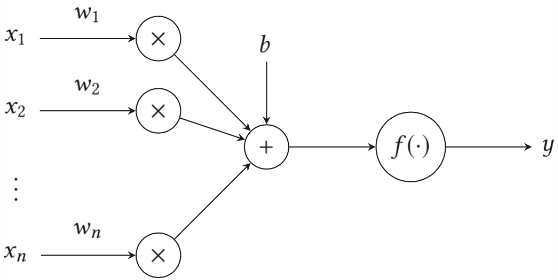
- Algorithm for Supervised Learning
  - Classification: $y = 0$ or 1
- Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)
- Activation function $f(\cdot)$
- Proved that Perceptron would automatically learn weights



Deep Learning

Perceptron – Activation Function

- *Activation function* is what is being learned (nothing else)

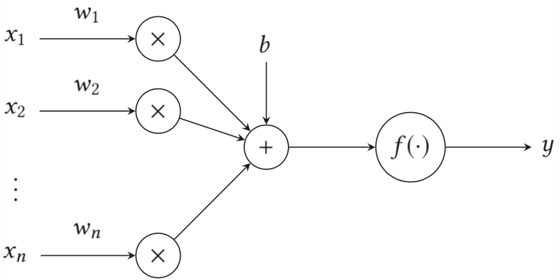


Activation functions like Sigmoid, ReLU, or even a step function provide non-linearity that helps approximate general functions

Deep Learning

Perceptron – Activation Function

- *Activation function* is what is being learned (nothing else)
  - Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)

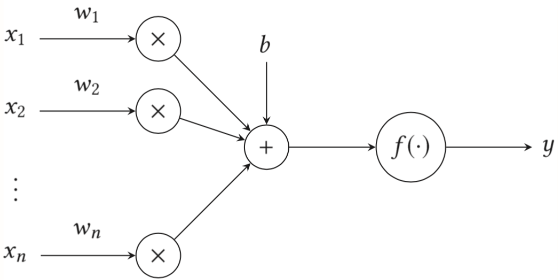


Activation functions like Sigmoid, ReLU, or even a step function provide non-linearity that helps approximate general functions

Deep Learning

Perceptron – Activation Function

- *Activation function* is what is being learned (nothing else)
  - Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)
- Can we just have a simple linear activation function?

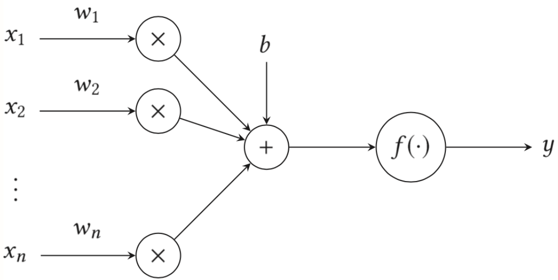


Activation functions like Sigmoid, ReLU, or even a step function provide non-linearity that helps approximate general functions

Deep Learning

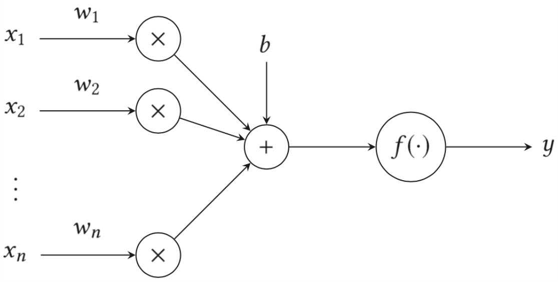
Perceptron – Activation Function

- *Activation function* is what is being learned (nothing else)
  - Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)
- Can we just have a simple linear activation function?
- $\boxed{f(x) = w_0 + w_1 x_1 + \ldots w_n x_n}$



Activation functions like Sigmoid, ReLU, or even a step function provide non-linearity that helps approximate general functions

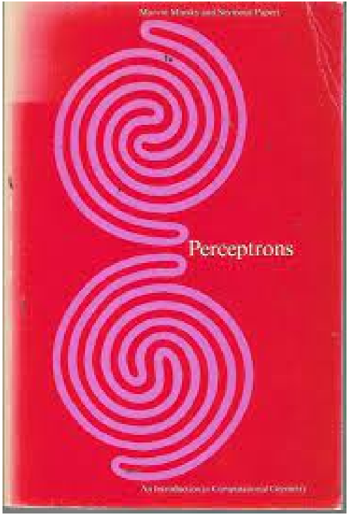
- *Activation function* is what is being learned (nothing else)
  - Parameters: Bias (b or w_0), Weights ($w_1, \ldots, w_n$)
- Can we just have a simple linear activation function?
- $\boxed{f(x) = w_0 + w_1 x_1 + \ldots w_n x_n}$
- Turns out you need *some* non-linearity



Activation functions like Sigmoid, ReLU, or even a step function provide non-linearity that helps approximate general functions

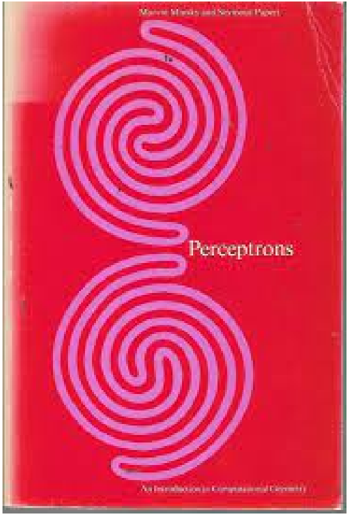
Minsky: The Exclusive OR (XOR) Problem

- History of AI was altered by a book titled “Perceptrons” (1969)



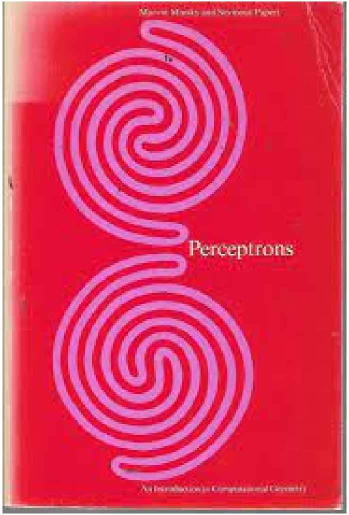
Minsky: The Exclusive OR (XOR) Problem

- History of AI was altered by a book titled "Perceptrons" (1969)
  - written by MIT Professor Minsky and Seymour Papert



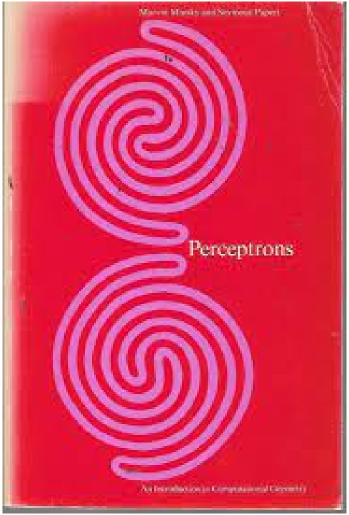
Minsky: The Exclusive OR (XOR) Problem

- History of AI was altered by a book titled "Perceptrons" (1969)
  - written by MIT Professor Minsky and Seymour Papert
- Proved that a Perceptron could not learn the XOR function



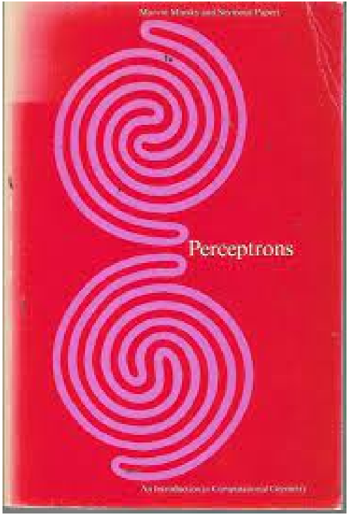
Minsky: The Exclusive OR (XOR) Problem

- History of AI was altered by a book titled "Perceptrons" (1969)
  - written by MIT Professor Minsky and Seymour Papert
- Proved that a Perceptron could not learn the XOR function
  - Challenge because we cannot be sure what was possible to learn and what was not



Minsky: The Exclusive OR (XOR) Problem

- History of AI was altered by a book titled "Perceptrons" (1969)
  - written by MIT Professor Minsky and Seymour Papert
- Proved that a Perceptron could not learn the XOR function
  - Challenge because we cannot be sure what was possible to learn and what was not
- Many attribute this book with causing an AI winter



What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function
- Even if you had essentially an infinite amount of data, you could not obtain parameters that would make $f(X)$ close to $y = XOR(x_1, x_2)$

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function
- Even if you had essentially an infinite amount of data, you could not obtain parameters that would make $f(X)$ close to $y = XOR(x_1, x_2)$
- However, the problem does not hold when we have multiple layers of neurons

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function
- Even if you had essentially an infinite amount of data, you could not obtain parameters that would make $f(X)$ close to $y = XOR(x_1, x_2)$
- However, the problem does not hold when we have multiple layers of neurons
    - Even a relative small number

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function
- Even if you had essentially an infinite amount of data, you could not obtain parameters that would make $f(X)$ close to $y = XOR(x_1, x_2)$
- However, the problem does not hold when we have multiple layers of neurons
  - Even a relative small number
- Which is why we need deep neural networks

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

What is the Exclusive OR (XOR) Problem?

- Perceptrons were proven to be incapable of learning this function
- Even if you had essentially an infinite amount of data, you could not obtain parameters that would make $f(X)$ close to $y = XOR(x_1, x_2)$
- However, the problem does not hold when we have multiple layers of neurons
  - Even a relative small number
- Which is why we need deep neural networks

x_1	x_2	$y = XOR(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Universal Approximation

Can approximate any “reasonable” function

Neural Networks, Vol. 2, pp. 359–366, 1989
Printed in the USA.

0893-6080/89 $3.00

ORIGINAL CONTRIBUTION

Multilayer Feedforward Networks are Universal Approximators

KURT HORNIK
Technische Universität Wien

MAXWELL STINCHCOMBE AND HALBERT WHITE
University of California, San Diego

(*Received* 16 *September* 1988; *revised and accepted* 9 *March* 1989)

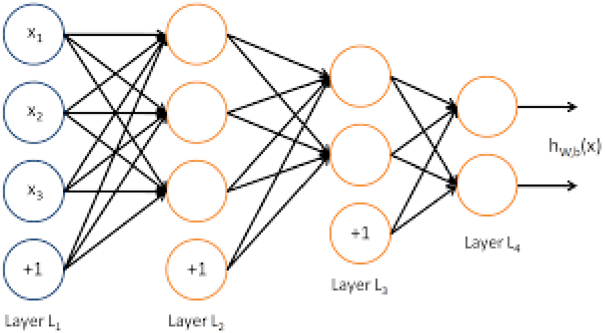
Abstract—*This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available. In this sense, multilayer feedforward networks are a class of universal approximators.*

Keywords—Feedforward networks, Universal approximation, Mapping networks, Network representation capability, Stone-Weierstrass Theorem, Squashing functions, Sigma-Pi networks, Back-propagation networks.

https://cognitivemedium.com/magic_paper/assets/Hornik.pdf

Universal Approximation

Can approximate any “reasonable” function



Deep Learning: Digit Recognition

- How to recognize digits – why?

Deep Learning: Digit Recognition

- How to recognize digits – why?
    - Standardized into a 28 x 28 pixel grid

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%
- Deep Learning models (CNN) achieve 0.5% error rate

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%
- Deep Learning models (CNN) achieve 0.5% error rate
  - Other ML models require a lot of feature engineering done by hand

Deep Learning: Digit Recognition

- How to recognize digits – why?

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%
- Deep Learning models (CNN) achieve 0.5% error rate

Deep Learning: Digit Recognition

- How to recognize digits – why?
  - Standardized into a 28 x 28 pixel grid
- 60,000 training and 10,000 test images
- Human error rate is about 2%
- Deep Learning models (CNN) achieve 0.5% error rate
  - Other ML models require a lot of feature engineering done by hand

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels
    - An image is not noise, it is made up of components

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels
    - An image is not noise, it is made up of components
- Use a special operation called a "convolution" that averages out information in parts of the image

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels
  - An image is not noise, it is made up of components
- Use a special operation called a "convolution" that averages out information in parts of the image
- Can identify local features

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels
  - An image is not noise, it is made up of components
- Use a special operation called a "convolution" that averages out information in parts of the image
- Can identify local features

Convolutional Neural Network

CNN

- Specialized type of Deep NN for image like data
- What is special about an image?
- Locally (pixel-to-pixel) there is continuity between pixels
  - An image is not noise, it is made up of components
- Use a special operation called a "convolution" that averages out information in parts of the image
- Can identify local features

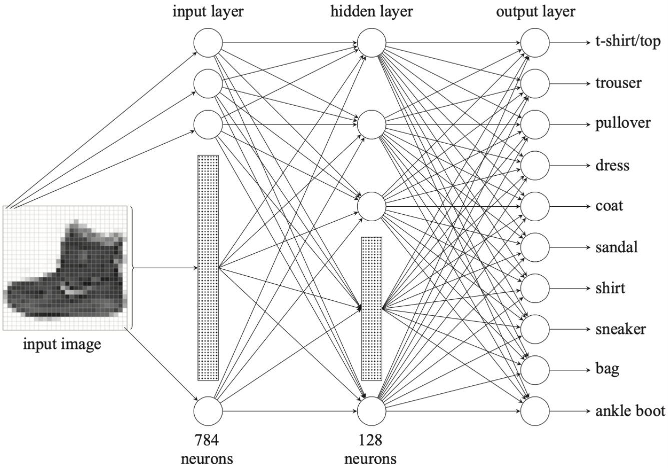
Convolution in Action

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

CIFAR10: Clothing Data Set

Source: Algorithms by Louridas, Chapter 6

CIFAR10: Deep Neural Net



Source: Algorithms by Louridas, Chapter 6

Deep Net Architecture

Regularization

What is Regularization?

Regularization is a set of techniques for reducing model complexity and increasing generalization ability

- Deep Learning models can become complex very quickly

Deep Net Architecture

Regularization

What is Regularization?

Regularization is a set of techniques for reducing model complexity and increasing generalization ability

- Deep Learning models can become complex very quickly
  - Current models can easily have *millions* of parameters and work great with training data

What is Regularization?

Regularization is a set of techniques for reducing model complexity and increasing generalization ability

- Deep Learning models can become complex very quickly
  - Current models can easily have *millions* of parameters and work great with training data
- Achieve higher performance on test data by making a model simpler

Deep Net Architecture

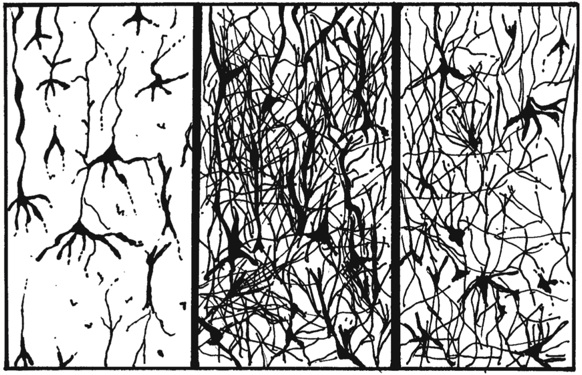
Regularization

What is Regularization?

Regularization is a set of techniques for reducing model complexity and increasing generalization ability

- Deep Learning models can become complex very quickly
  - Current models can easily have *millions* of parameters and work great with training data
- Achieve higher performance on test data by making a model simpler
- Is there a connection to how humans learn?

Synaptic Density in Humans varies with age

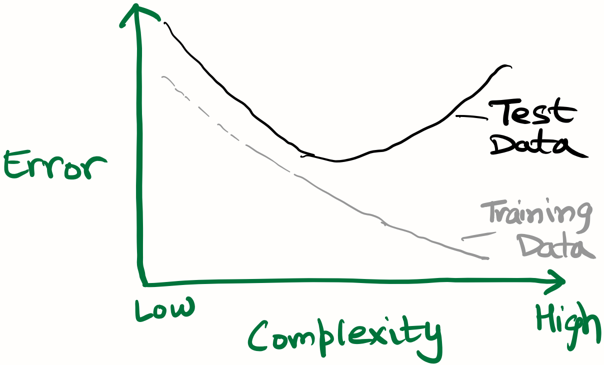


at a child's birth | at 7 years of age | at 15 years of age

Understanding Brain Development in Young Children, Brotherson, 2022

Children typically have more synapses (connections between neurons) than adults

Generalization Error and Model Complexity



Complex models explain *training* data better but not *test* data $\implies$ we need validation and test data that is separate (no leakage) from training data

Recurrent Neural Network

RNN

- Images have a spatial regularity (local contiguity)

Recurrent Neural Network

RNN

- Images have a spatial regularity (local contiguity)
- Language (written or speech) has sequence regularity

Recurrent Neural Network

RNN

- Images have a spatial regularity (local contiguity)
- Language (written or speech) has sequence regularity
- Next word prediction leverages this:

Recurrent Neural Network

RNN

- Images have a spatial regularity (local contiguity)
- Language (written or speech) has sequence regularity
- Next word prediction leverages this:
    - Start a sentence with the word “I” (very different from ”Our”)

Recurrent Neural Network

RNN

- Images have a spatial regularity (local contiguity)
- Language (written or speech) has sequence regularity
- Next word prediction leverages this:
  - Start a sentence with the word "I" (very different from "Our")
- RNNs accommodate and model these patterns by altering the structure of the feedforward networks

Recurrent Neural Network

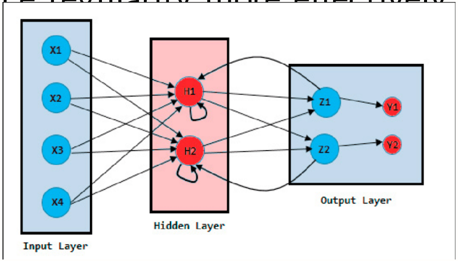
RNN

- Images have a spatial regularity (local contiguity)
- Language (written or speech) has sequence regularity
- Next word prediction leverages this:
  - Start a sentence with the word “I” (very different from ”Our”)
- RNNs accommodate and model these patterns by altering the structure of the feedforward networks
- The output of neurons in “later” layers is given as input to same “earlier” layers

Recurrent Neural Network

RNN

- Not just feedforward: Allow connections to go “backwards”
- Can learn sequence regularity more effectively



https://www.researchgate.net/figure/Basic-Architecture-of-Recurrent-Neural-Network-22-23_fig2_325668211

Generative AI Idea

Generative AI Models for Text – Large Language Models (LLMs)

LLMs

How do they work?

Some questions with LLMs:

- What are they trained on?

LLMs

How do they work?

Some questions with LLMs:

- What are they trained on?
- Why is language predictable?

LLMs

How do they work?

Some questions with LLMs:

- What are they trained on?
- Why is language predictable?
- Are they trained with supervised or unsupervised learning?

LLMs

How do they work?

Some questions with LLMs:

- What are they trained on?
- Why is language predictable?
- Are they trained with supervised or unsupervised learning?
- Why do they show different outputs each time you ask the same thing?

LLMs

How do they work?

Some questions with LLMs:

- What are they trained on?
- Why is language predictable?
- Are they trained with supervised or unsupervised learning?
- Why do they show different outputs each time you ask the same thing?
- Why do they hallucinate – create random or nonsense text?

Generative AI Model – Images

Generative AI

- Can AI actually be part of the creation process?

Generative AI

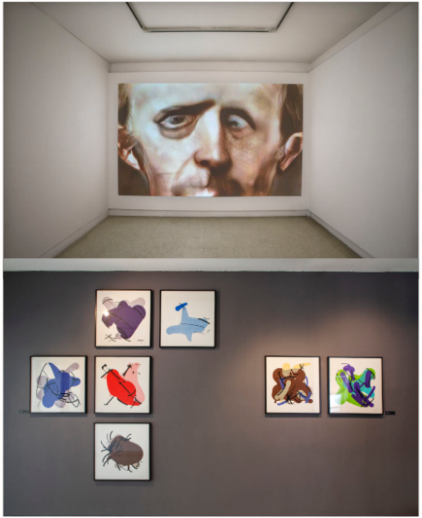
- Can AI actually be part of the creation process?
  - AI can do not just prediction but also generation

Generative AI

- Can AI actually be part of the creation process?
    - AI can do not just prediction but also generation
    - AI can create or co-create with humans

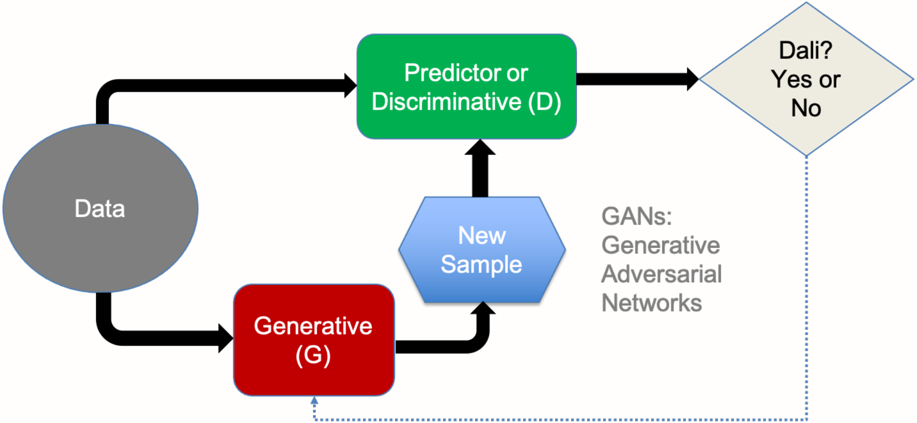
Generative AI

Real Paintings by Dali



Generative AI

Generative AI



GANs: Generative Adversarial Networks

Reinforcement Learning

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997
- AlphaGO defeated leading Go player in 2017

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997
- AlphaGO defeated leading Go player in 2017
  - How was AlphaGO trained?

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997
- AlphaGO defeated leading Go player in 2017
    - How was AlphaGO trained?
- AlphaGo Zero was introduced in 2017 in the journal Nature

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997
- AlphaGO defeated leading Go player in 2017
  - How was AlphaGO trained?
- AlphaGo Zero was introduced in 2017 in the journal Nature
- AlphaGo Zero was based on RL and taught itself mastery in 21 days

Power of RL in Games

- Deep Blue defeated world Chess champion in 1997
- AlphaGO defeated leading Go player in 2017
  - How was AlphaGO trained?
- AlphaGo Zero was introduced in 2017 in the journal Nature
- AlphaGo Zero was based on RL and taught itself mastery in 21 days
- Defeated AlphaGo 100–0

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions
- Does not require any data

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions
- Does not require any data
- Does require the ability to experiment or "trial and error"

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions
- Does not require any data
- Does require the ability to experiment or “trial and error”
- How does RL compare to typical A/B testing experimentation?

What is Reinforcement Learning (RL)?

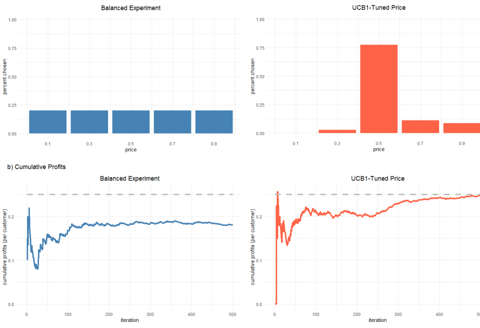
- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions
- Does not require any data
- Does require the ability to experiment or "trial and error"
- How does RL compare to typical A/B testing experimentation?
- Humans also use this mode of learning especially children

What is Reinforcement Learning (RL)?

- Algorithm designed to learn from interactions to maximize reward(s)
- Learns an optimal sequence of actions
- Does not require any data
- Does require the ability to experiment or "trial and error"
- How does RL compare to typical A/B testing experimentation?
- Humans also use this mode of learning especially children
- Critically important to specify the reward function properly

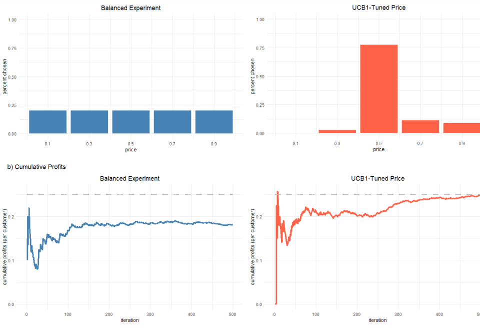
RL in Dynamic Pricing

- When you have a new product we don't know demand



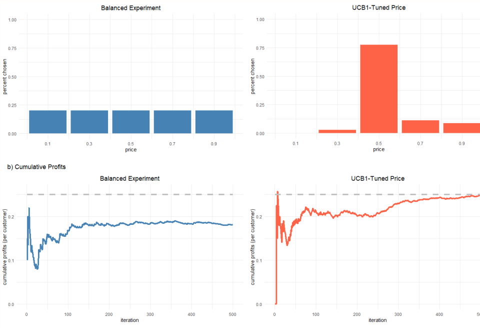
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?



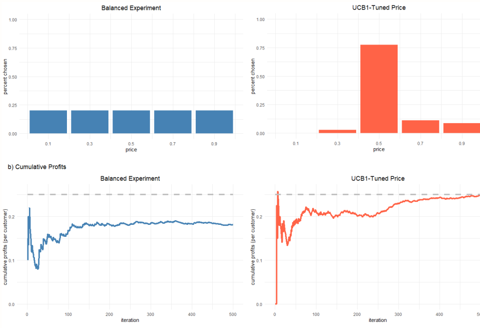
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up



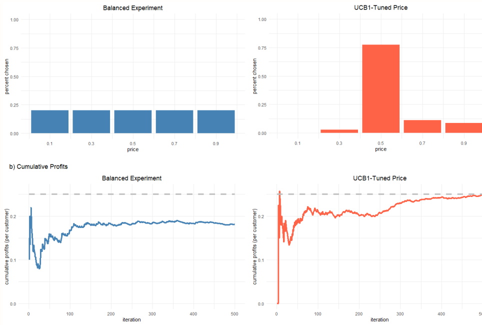
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up
    - Different from A/B testing?



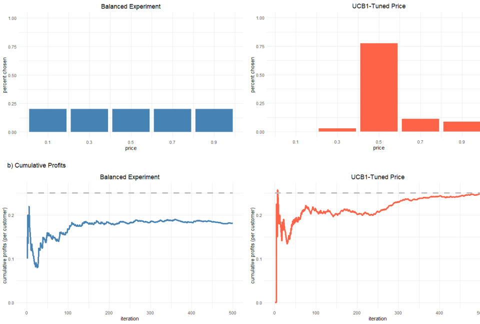
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up
  - Different from A/B testing?
- Actions, rewards and state?



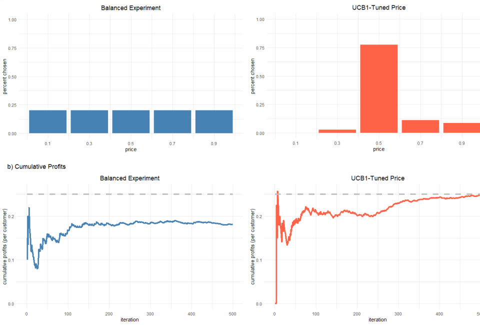
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up
  - Different from A/B testing?
- Actions, rewards and state?
- **Actions:** Charge price: $p = 1, 2, \ldots, 100$



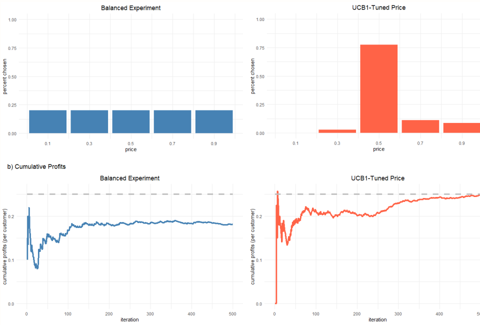
RL in Dynamic Pricing

- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up
  - Different from A/B testing?
- Actions, rewards and state?
- **Actions:** Charge price: $p = 1, 2, \ldots, 100$
- **Rewards:** $\Pi = p \times D(p)$



RL in Dynamic Pricing

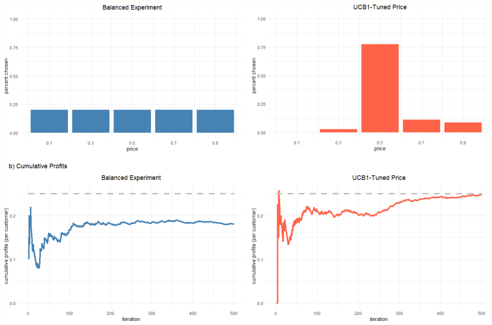
- When you have a new product we don't know demand
- How can you learn the demand curve?
- Specify as RL: Charge a different price each time a customer shows up
  - Different from A/B testing?
- Actions, rewards and state?
- **Actions:** Charge price: $p = 1, 2, \ldots, 100$
- **Rewards:** $\Pi = p \times D(p)$
- **State:** Sequence of prices tested and quantities sold



Now let's try this with Chess

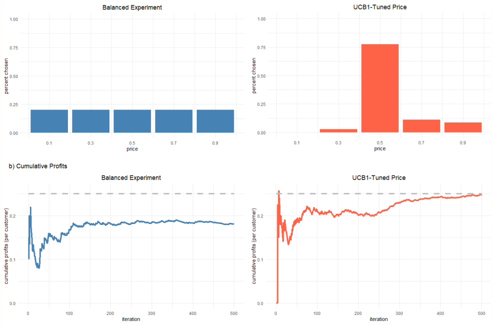
RL in Dynamic Pricing

- Blue is Balanced experiment (A/B test), Orange is RL



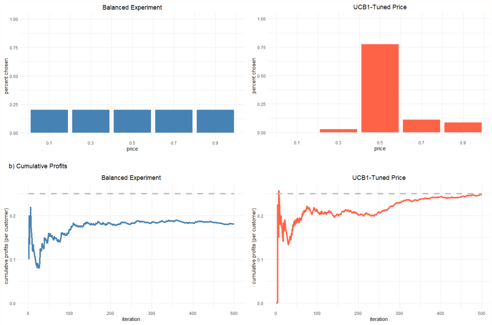
RL in Dynamic Pricing

- Blue is Balanced experiment (A/B test), Orange is RL
- RL chooses prices that are more "optimal"



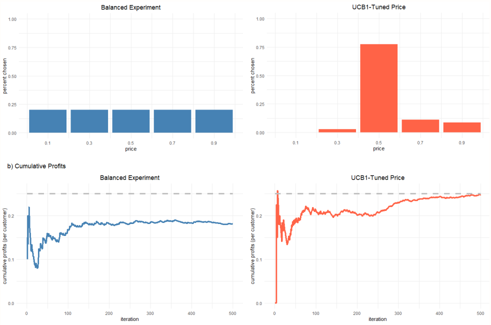
RL in Dynamic Pricing

- Blue is Balanced experiment (A/B test), Orange is RL
- RL chooses prices that are more "optimal"
- Learning *while* experimenting (RL) versus Learning *after* experimenting (A/B)



RL in Dynamic Pricing

- Blue is Balanced experiment (A/B test), Orange is RL
- RL chooses prices that are more "optimal"
- Learning *while* experimenting (RL) versus Learning *after* experimenting (A/B)
- RL obtains higher profit levels quicker



RL in Robotics

Next Class: Prediction to Decision

- Everything we have looked at up until now has focused on prediction problems
- What data is typically used, labels (y) and predictors (X)
  - Structured, Unstructured (Text, Images, Audio, Video,...)
- Building blocks of ML models
- Specifying prediction problems provides a foundation for **converting** a problem to a prediction problem

Specifying ML Problems (In class exercise)

Let's try this in Groups

- For each case, suggest an (U)nsupervised, (S)upervised **and** (R)einforcement learning approach.
- For (U) and (S) specify what variables you will use as y and X
- For (R), specify the actions (a), states (s), and reward function Π
- What is the goal of your ML model? What metric would you use to measure improvement?

3 Cases – Chose ONE

A) Social media platform like Instagram would like to increase engagement among its users

B) Content firm like Netflix or Spotify would like to recommend new content to its users

C) Apparel retailer like Patagonia would like to improve its product assortment