LECTURE 01

Introduction

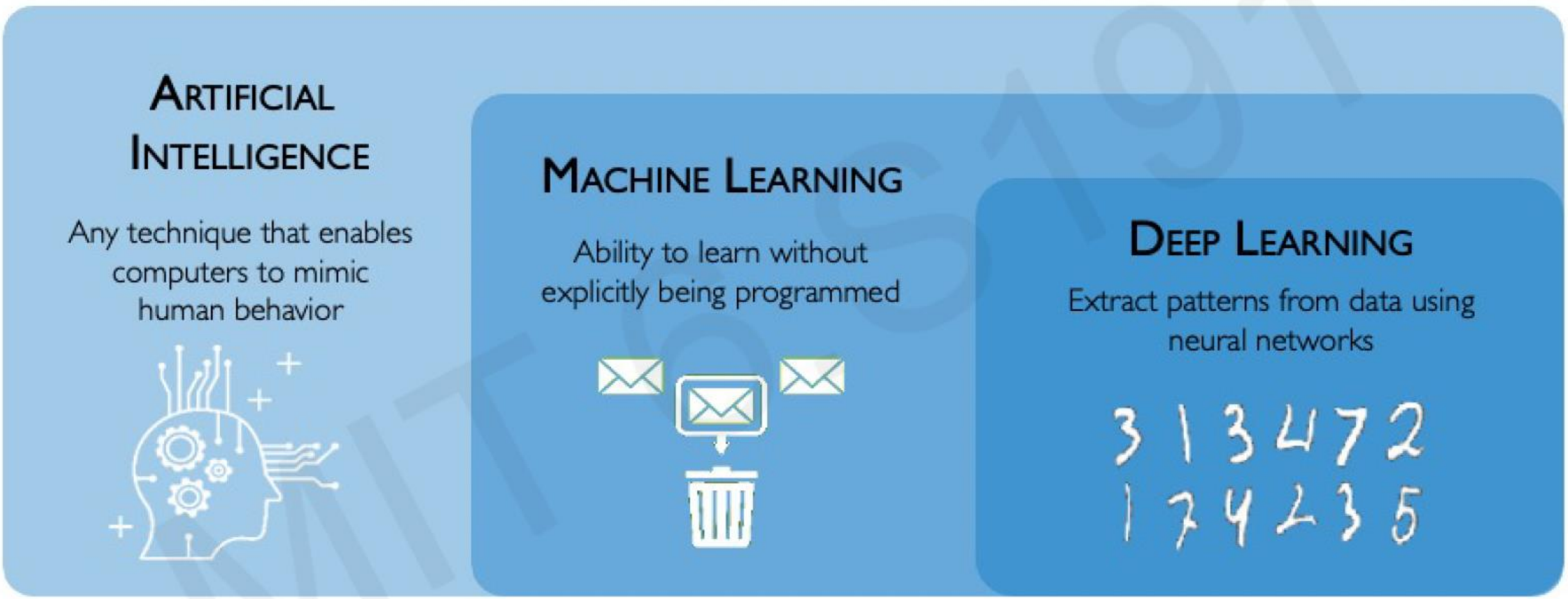
Course Logistics, Overview, Learning & Overfitting, Linear Regression as NN

CS/DS 541: Deep Learning, Fall 2025 @ WPI

Fabricio Murai

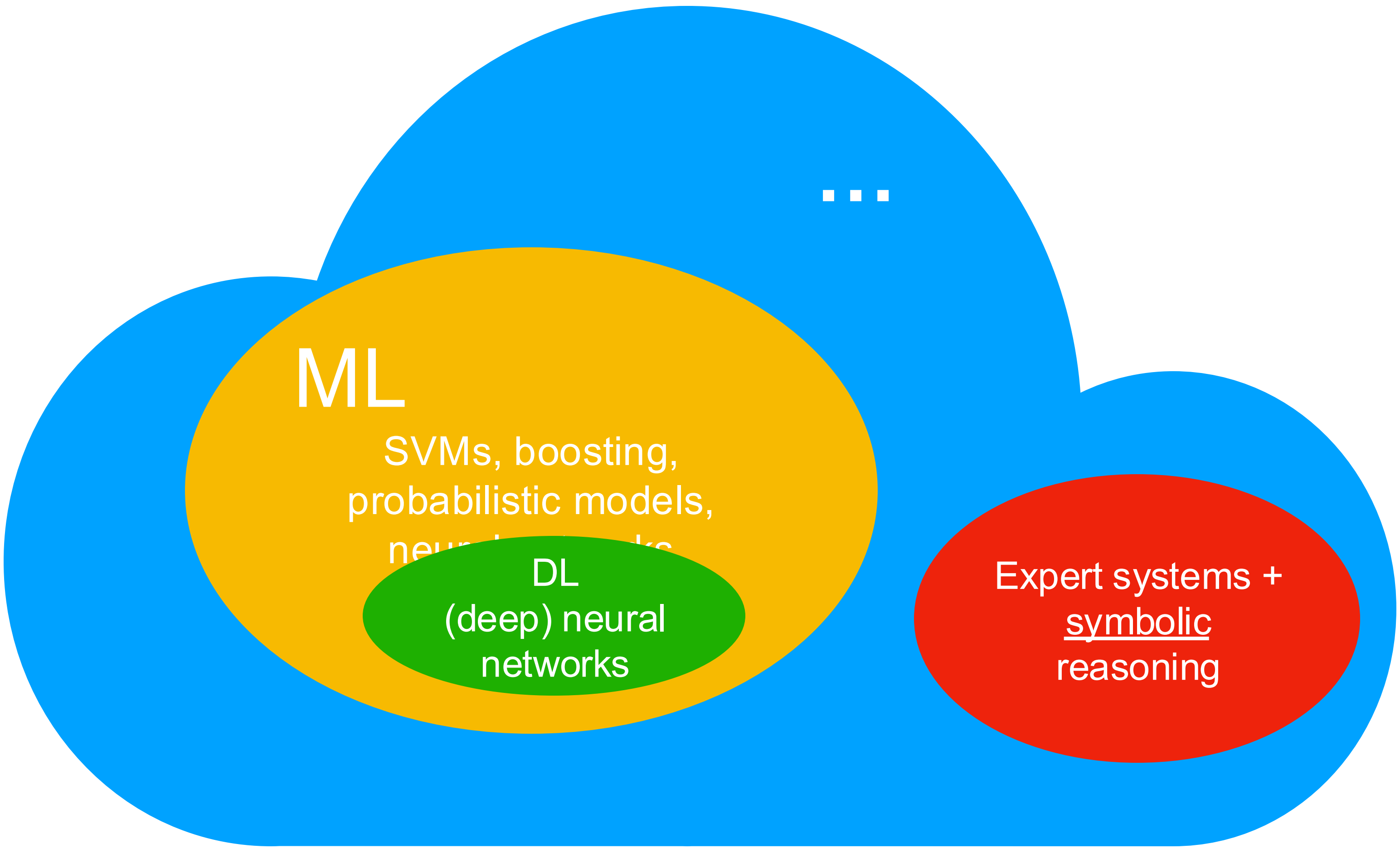
AI, ML & DL

What is Deep Learning?

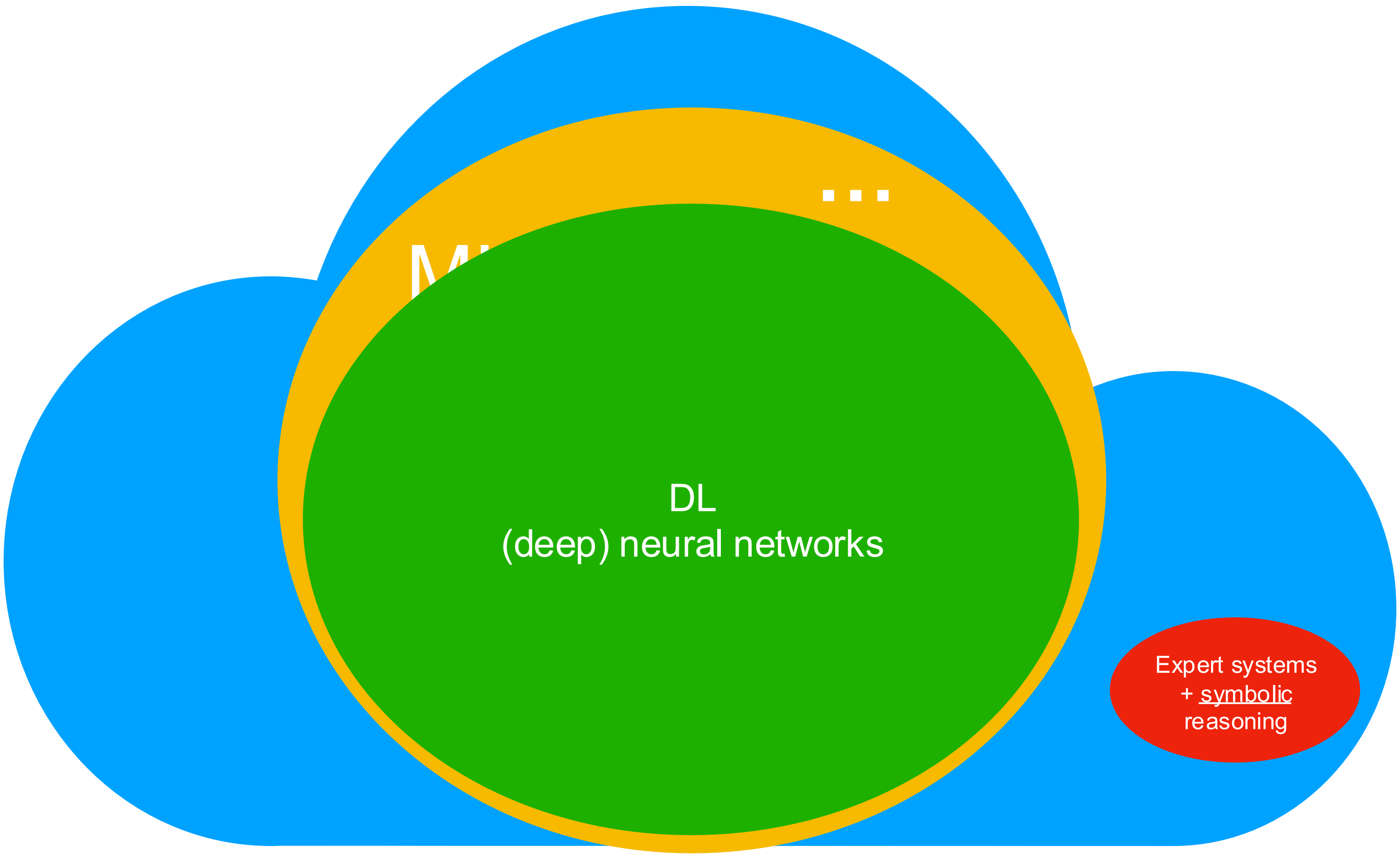


Teaching computers how to **learn a task** directly from **raw data**

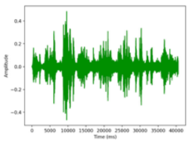
$DL \subset ML \subset AI$



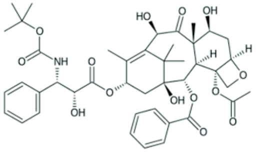
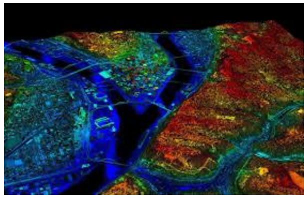
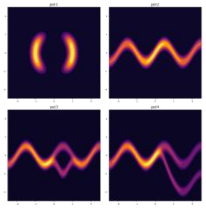
What people mean by “AI"



Domains

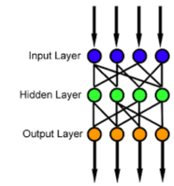
- Images 
- Audios 
- Videos 

- Sequences (e.g., text, market data)

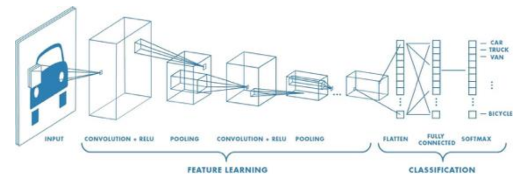
  `There once was a person from Nantucket…`
- Graphs (e.g., social networks, molecules)  
- Sets (e.g., point-cloud of LiDAR measurements) 
- Probability distributions 

Models and algorithms

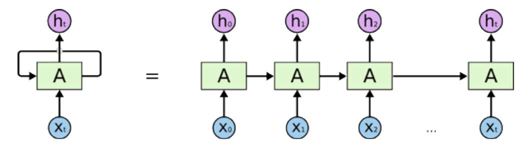
- Feed-forward neural networks (FFNN/DNN)



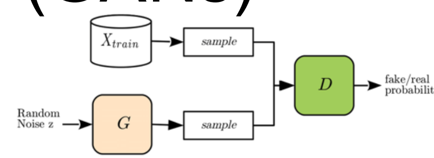
- Convolutional neural networks (CNN)



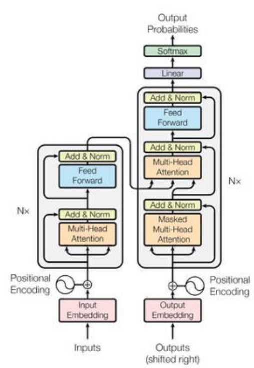
- Recurrent neural networks (RNN)



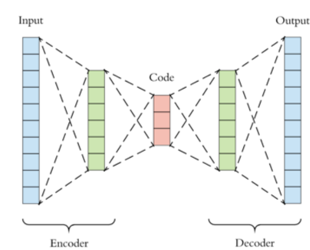
- Generative-adversarial networks (GANs)



- Transformers



- Auto-encoders (AEs) & variational AEs



Training regimes

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Few-shot learning
- Adversarial learning
- Generative models

CS541 Deep Learning: Course Outline
Part I: Lectures, Homeworks & Quizzes

9/1 No class

Changes & Rationale:

- Now lectures cover key concepts in ½ semester (before was 2/3); helps with ideas & time to work on project
- Shorter homeworks to be done in regular, small intervals (one week)

Date		Lecture	Events	Deadlines
Thu	8/21	Overview, Learning & Overfitting, Linear Regression as NN	HW 0 out	
Mon	8/25	Modern ML, Neural Nets as Universal Approximators, Gradient Descent & SGD	Quiz 0	
Thu	8/28	Training NNs, Convergence Issues, Loss Surfaces, Momentum	HW 1 out	HW 0 due
Mon	9/1	NO CLASS		
Thu	9/4	Classification, Representation, Autoencoders		Paper Choice due
Mon	9/8	Optimizers & Reg., Loss Functions, BatchNorm, Dropout	Quiz 1	
Thu	9/11	BackProp & Calculus of BackProp	HW 2 out	HW 1 due
Mon	9/15	Conv Layers	Quiz 2	
Thu	9/18	CNNs, ResNets & Learning Paradigms	HW 3 out	HW 2 due
Mon	9/22	Invariances & RNNs	Quiz 3	
Thu	9/25	BackProp Through Time, seq2seq models, RNN challenges	HW 4 out	HW 3 due
Mon	9/29	LSTM, Neural Attention	Quiz 4	
Thu	10/2	Transformer Models & Training	HW 5 out	HW 4 due
Mon	10/6	GenAI: VAEs & GANs	Quiz 5	
Thu	10/9	Diffusion Models		HW 5 due
Mon	10/13	BREAK		
Thu	10/17			

CS541 Deep Learning: Course Outline
Part II: Exam, Project, Paper & Project Presentations

Changes & Rationale:

- Exam at beginning of 2^{nd} half
- Individual paper presentation: values self-learning, practices ability to read papers & present, whiteboarding skills
- Small changes to grading

11/27 No class

12/8 No class

Mon	10/20	EXAM		
Thu	10/23	Example Paper Presentation + Exam Discussion		Project Proposal Due
Mon	10/27	Paper Presentation - Day 1		
Thu	10/30	Paper Presentation - Day 2		Reply to Proposal Comments
Mon	11/3	Paper Presentation - Day 3		
Thu	11/6	Paper Presentation - Day 4		
Mon	11/10	Paper Presentation - Day 5		
Thu	11/13	Paper Presentation - Day 6		Proposal Progress Report due
Mon	11/17	Paper Presentation - Day 7		
Thu	11/20	Paper Presentation - Day 8		Reply to Progress Report Comments
Mon	11/24	Paper Presentation - Do Overs		
Thu	11/27	NO CLASS		
Mon	12/1	Project Presentation - Day 1		
Thu	12/4	Project Presentation - Day 2		
Mon	12/8	NO CLASS		
Thu	12/11	Wrap-up and Advanced Topics		Project Write-Up Due

Prerequisites

- Course assumes some knowledge in: linear algebra, calculus, programming, ML, probability & stats.
- No single topic is too hard by itself.
- But we will cover and touch upon many topics and this is what makes the course hard.
- Programming
  - You should be able to write non-trivial programs (in Python)
  - Familiarity with PyTorch is a plus.

Course Logistics

- Class meets Mon and Thu 4:00-5:20pm ET *in person*
  - Videos of the lectures will be available on Canvas right after class
- Structure of lectures:
  - Thursdays: 80min of lecture with Q&A interactions
    I will try to do ungraded in-class activities
  - Mondays: 20min of quiz followed by
    60min of lecture with Q&A interactions

Logistics: Office Hours

- What are Office Hours (OHs)?
  - Opportunity to connect with the instructor and the TAs, ask questions about lectures, assignments, discuss project, papers, or even just chat.
  - It does not need to be a pressing matter. Every student in this class is welcome.

- OHs will be either in person or virtual
  - We will have OHs every day until BREAK, starting from 2nd week
  - Prof. Murai's OH: 30 minutes after each lecture
    Mon/Thu 5:20-5:50pm
    In classroom (UH 420)
  - Noushin Largani's (TA) OH:
    Tue/Wed/Fri 11:00-12:00pm
    UH 341 (shared lab)

How to contact me?

- Instructor: Prof. Fabricio Murai
- Office Hours: 30 minutes after each lecture
  Same classroom (UH 420)
- Email: ***Please do NOT use email for this course***
- In-person location: Office UH365 (please schedule first)

Join Discord Server here

Work for Course: Grading

In-class Quizzes (first 6 Mondays, in-person only): 9% (1x1% + 4x2%).
There will be make-up opportunities in case of sickness, family emergency or other excusable reasons*.
Lowest Quiz grade is dropped.

Homework assignments (1+5 coding & theory): 28% (1x3% + 5x5%).
HW0 is individual; HWs 1-5 can be done in groups of up to 2 students.
Lowest HW grade is dropped.

Slack Days: Students can distribute up to 7 days of lateness across homeworks.

Paper presentation: 5%.
An individual presentation of a seminal or recent paper published in a top Deep Learning conference.

Engagement with Presenters: 5%.
Attend peer presentations and engage with them meaningfully by asking questions.

Exam: 18% [in class, Oct 20].

Final project: 35%.
Students will be able to define teams of 3-4 members on their own. Project consists of:

Class participation: 3% extra points on final grade.
Students will have opportunities to contribute to class discussions.

Work for Course: Submitting

- How to submit?
  - Upload via Canvas
  - Posted deadline is 5:59pm to remind you to ask questions before 6pm
  - True deadline is 11:59pm
  - Homeworks
    - **Code:** (python script OR jupyter notebook) <u>AND</u> PDF export
    - **Math:** typed OR handwritten solutions OR jupyter notebook
- Total of <u>7 Slack Days</u> per student can be used <u>towards</u> <u>HWs</u>
  - Late submissions count against slack days of all team members
  - (Corollary) No HW can be submitted more than 7 days late
  - Paper & Project deliverables do NOT benefit from slack days

Work for Course: Quizzes

- First 6 Mondays, in-class.
  - Administered on paper.
  - Quiz 0: 1% of final grade
  - Quizzes 1 through 5: 2% of final grade
  - Lowest quiz is dropped (total 9% of grade)
  - Make-up oppts in case of sickness, family emergency or other excusable reasons.
  - Content
    - Lecture slides & in-class activities
  - Attention! First quiz next Monday (syllabus + Lec 1)

Work for Course: Exam

- Single in-class exam, 1-sheet notes, timed
  - Administered on paper.
  - Duration: 80 minutes.
  - Date: Monday, October 20th.
  - Content
    - Lecture slides & in-class activities
    - Concepts seen in HWs
    - 50-60% multiple choice questions
    - Remainder is open-ended
    - More details to come!

Code of Conduct

- We strictly enforce WPI Student Code of Conduct
  - **First time incidents will incur a Departmental Agreement; no exceptions will be made.**

Make sure you read and understand it!

- What is your position re using LLMs on graded activities?
  - Watching lectures & videos will give you the impression that you are learning, but you won't know whether you did until you attempt to solve problems
  - It will be hard, but most learning happens through "productive struggles"
  - Quizzes are formative assessments: low stakes, aim to check learning; it is good for you to know what you don't know
  - Homeworks: course staff and I will strive to provide you no more, no less than what you need for learning and solving homework problems
  - In contrast, if you ask an LLM a problem, you will get answers (some incorrect)
  - Exam is a summative assessment; when the same concepts are evoked in Exam, you may not be able to recall or utilize them properly
  - Think about your future as a professional

What is Machine Learning?

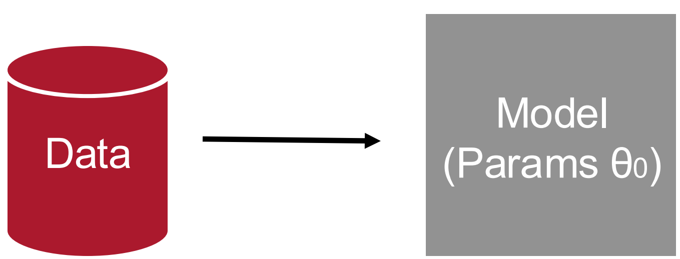
and

Is Learning = Optimization?

Machine Learning Refresher

- What is **Machine Learning**?
  - Study of **algorithms** (models) that improve their **performance** at a **task** through **experience** (data).
- **Real-World Applications:** image recognition, recommendation systems, anomaly detection, and more.
- Two moments:

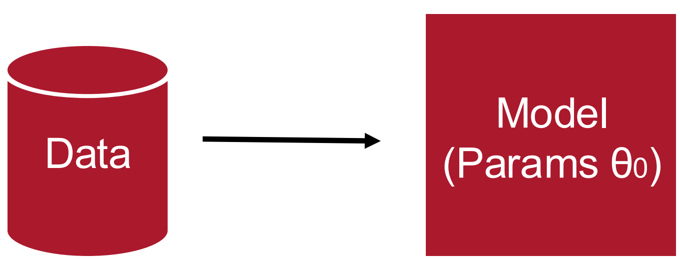
1. Training



Machine Learning Refresher

- What is **Machine Learning**?
  - Study of **algorithms** (models) that improve their **performance** at a **task** through **experience** (data).
- **Real-World Applications:** image recognition, recommendation systems, anomaly detection, and more.
- Two moments:

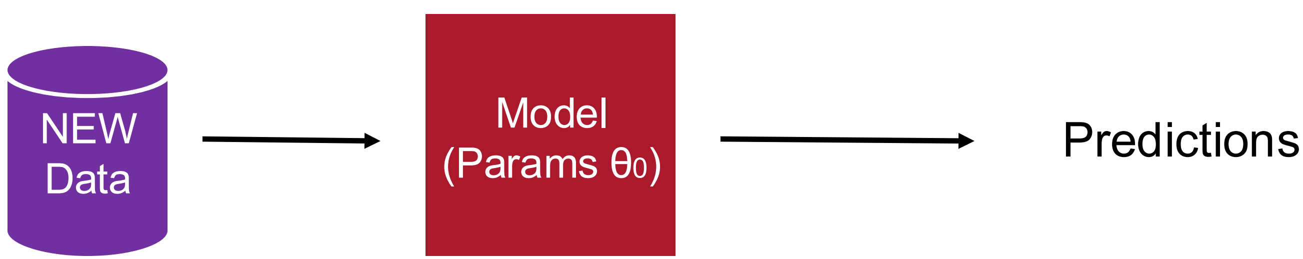
1. Training



Machine Learning Refresher

- What is **Machine Learning**?
  - Study of **algorithms** (models) that improve their **performance** at a **task** through **experience** (data).
- **Real-World Applications:** image recognition, recommendation systems, anomaly detection, and more.
- Two moments:

2. Testing/Inference

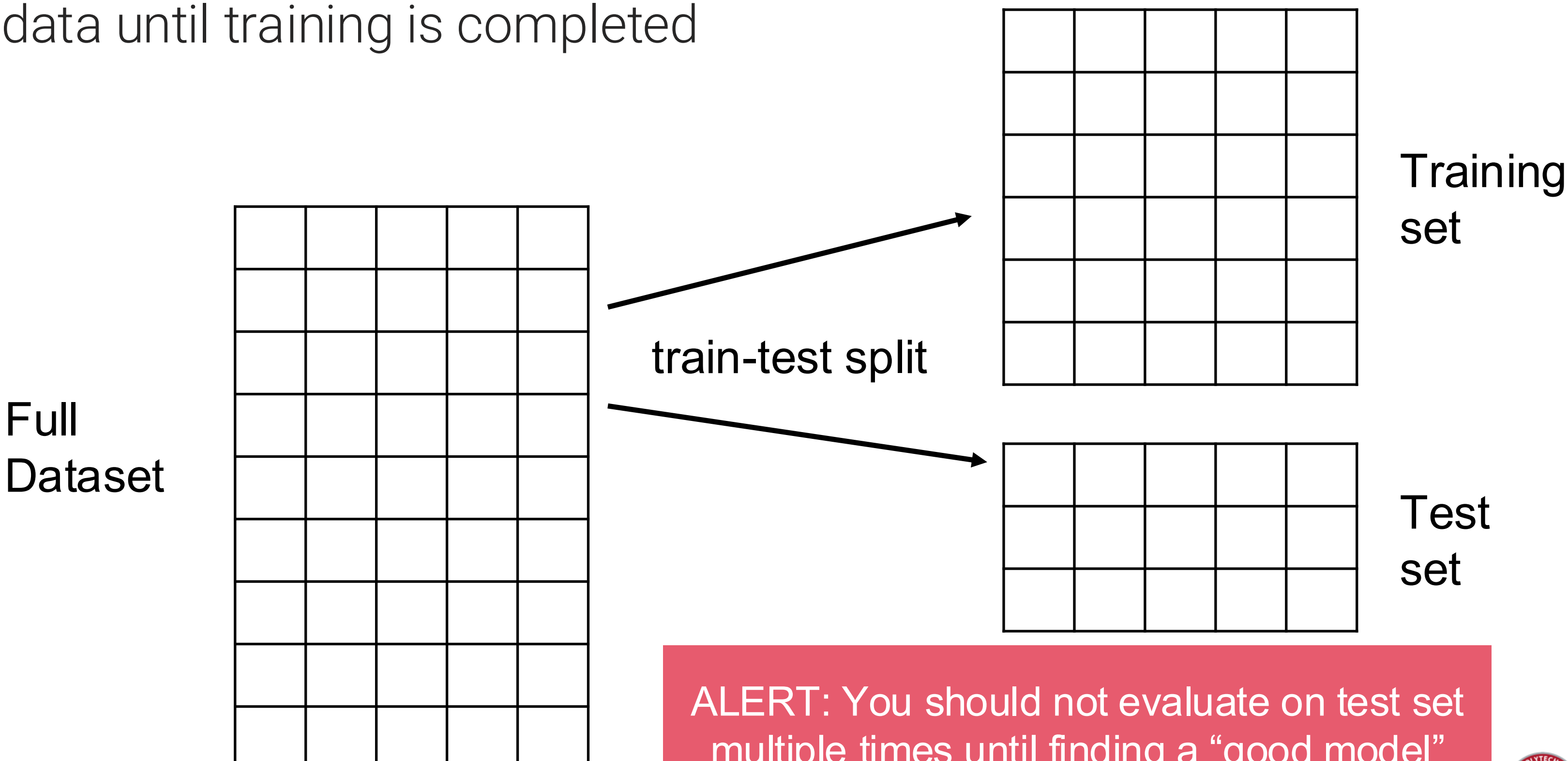


What is a good model?

- Help me out…

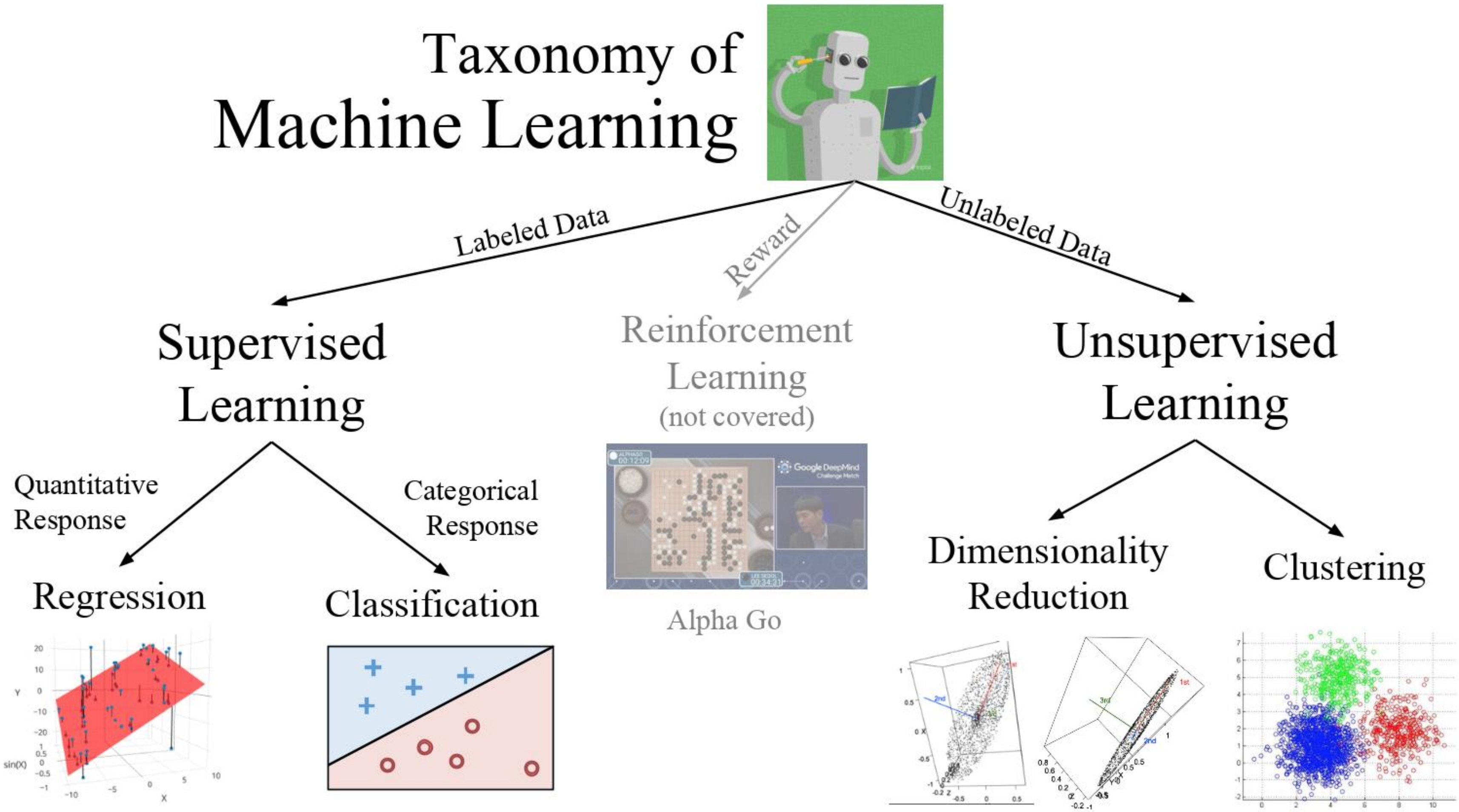
Learning ≠ Optimization

- In practice, we want to optimize model performance on *unseen data* (i.e., generalization).
- Optimization is a <u>means</u> for Learning
- We can estimate generalization performance by holding out part of the data until training is completed



ALERT: You should not evaluate on test set multiple times until finding a “good model”

A helpful detour: Linear Regression



The Modeling Process

1. Choose a model	How should we represent the world?
2. Choose a loss function	How do we quantify prediction error?
3. Fit the model	How do we choose the best parameters of our model given our data?
4. Evaluate model performance	How do we evaluate whether this process gave rise to a good model?

Linear regression

- Denote dataset by $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
  - Boston Housing example.
    Each row represents a suburb/town.
    medv: median value of owner-occupied homes in $1K

1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4

$\mathbf{x}^{(1)}$ $y^{(1)}$

- Prediction: want to create a "machine" to estimate $y^{(i)}$ for new $\mathbf{x}^{(i)}$ with high accuracy.

Linear regression

- Denote dataset by $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
  Suppose $\mathbf{x}^{(i)} \in \mathbb{R}^m$.

- Define machine as function g (with parameters $\mathbf{w}$) whose output $\hat{y}$ is linear in its inputs:

$$y^{(i)} = g(\mathbf{x}^{(i)}; \mathbf{w}) = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_m x_m^{(i)}$$

Ingredient #1

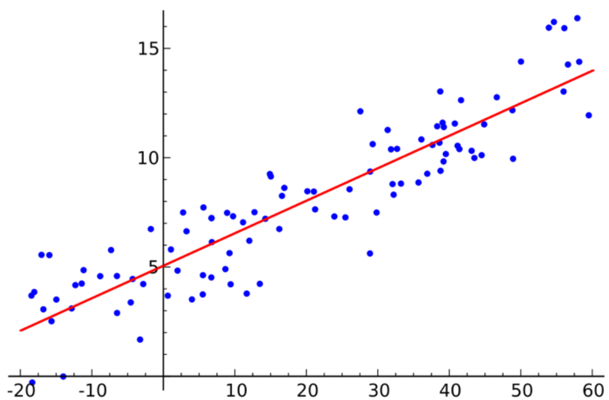


Image Credits: Sewaqu

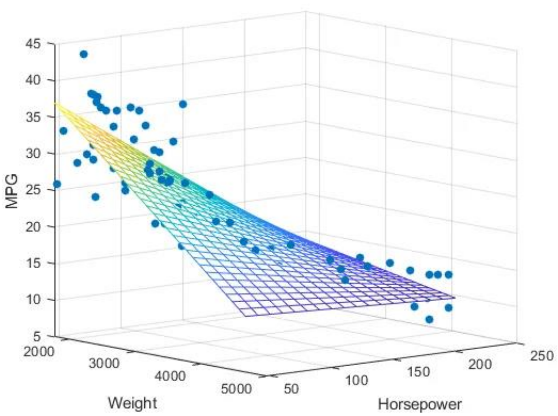


Image Credits: Lekha Priya

Linear regression

- Denote dataset by $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
  Suppose $\mathbf{x}^{(i)} \in \mathbb{R}^m$.

- Define machine as function g (with parameters $\mathbf{w}$) whose output $\hat{y}$ is linear in its inputs:

$$y^{(i)} = g(\mathbf{x}^{(i)}; \mathbf{w}) = w_0 + w_1 x_1^{(i)} + w_2 x_2^{(i)} + w_m x_m^{(i)}$$

Ingredient #1

Drop $^{(i)}$ + vectorize, to simplify notation:

$$\hat{y} = \sum_{j=1}^{m} w_j x_j + b$$

*w*0 is called intercept or bias; we will denote as *b*

Linear regression

- Given a dataset D, we want to optimize **w**.
- Model definition: $\hat{y} = g(\mathbf{x}; \mathbf{w}, b) = \sum_{j=1}^{m} w_j x_j + b$
- Let's choose each "weight" w_j to minimize the **mean squared error** (MSE) of our predictions.
- We can define the **loss** function that we seek to minimize:

Ingredient #2

$$f_{\text{MSE}}(y, \hat{y}; \mathbf{w}, b) = \frac{1}{n} \sum_{i=1}^{n} (g(\mathbf{x}^{(i)}; \mathbf{w}, b) - y^{(i)})^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} ({\mathbf{x}^{(i)}}^\top \mathbf{w} + b - y^{(i)})^2$$

Note: can choose other loss functions, but MSE is most common

WPI

Linear regression: exact solution

- $\mathbf{w}$ is an unconstrained real-valued vector; hence, we can use differential calculus to find the minimum of f_{MSE}.
- Just derive the gradient of f_{MSE} w.r.t. $\mathbf{w}$ and b, set to 0, and solve.
- Since f_{MSE} is a convex function, we are guaranteed that this critical point is a global minimum.

Solving for b

- MSE loss:

$$f_{\mathrm{MSE}}(y,\hat{y};\mathbf{w},b)=\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)}{}^{\top}\mathbf{w}+b-y^{(i)})^2$$

- Taking derivative wrt to b:

$$\frac{\partial f_{\mathrm{MSE}}}{\partial b}=\frac{1}{n}\sum_{i=1}^{n}2(\mathbf{x}^{(i)}{}^{\top}\mathbf{w}+b-y^{(i)})\cdot 1$$

$$=\frac{2}{n}\sum_{i=1}^{n}(\widehat{y}^{(i)}-y^{(i)})$$

What can we say about the sum of the residuals when b is optimal?

WPI

Solving for w

- The gradient of f_{MSE} wrt $\mathbf{w}$ is thus:

$$\nabla_{\mathbf{w}} f_{\text{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) = \nabla_{\mathbf{w}} \left[\frac{1}{2n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right]$$

Added 2 in denominator to make derivation easier

Solving for w

- The gradient of f_{MSE} wrt **w** is thus:

$$\begin{aligned}\nabla_{\mathbf{w}} f_{\mathrm{MSE}}(\mathbf{y}, \hat{\mathbf{y}}; \mathbf{w}) &= \nabla_{\mathbf{w}} \left[\frac{1}{2n} \sum_{i=1}^{n} \left(\mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right] \\ &= \frac{1}{2n} \sum_{i=1}^{n} \nabla_{\mathbf{w}} \left[\left(\mathbf{x}^{(i)\top} \mathbf{w} - y^{(i)} \right)^2 \right]\end{aligned}$$

Hint: gradient ∇ will have the same shape as params; details on HW1

What if there is no closed form solution?

- Alternatively, linear regression can be solved numerically using gradient descent.
- **Numerical solution**: need to iterate (according to some algorithm) many times to *approximate* the optimal value.
- Gradient descent is more laborious to code than the exact solution, but it generalizes to a wide variety of ML models.

Evaluating Models

What are some ways to determine if our model was a good fit to our data?

In this case, minimizing MSE is equivalent to minimizing RMSE; however, in general, loss function & evaluation don't need to be equivalent

1. Compute statistics:
   - Compute column means, standard deviation.
   - If fitting a simple linear mode, compute correlation r.

2. Example of performance metric: Root Mean Square Error (RMSE)

$$\text{RMSE}(y, \hat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

   - It is the square root of MSE, which is loss that we've been minimizing to determine optimal parameter models.
   - RMSE is in the same unit as y.
   - Lower RMSE indicates more "accurate" predictions (lower "average loss" across data)

3. Visualization:
   - Look at a residual plot of $e_i = y_i - \hat{y}_i$ to visualize the difference between actual and predicted values.

Recap

- Learning ≠ Optimization
- Supervised learning
  - Training: use labeled data to learn model params
  - Testing/Inference: use learned model to make predictions for new data
  - REMEMBER to split data and not to evaluate on test data more than once
- Modeling Process
  1. Choose a model
  2. Choose a loss function
  3. Fit the model
  4. Evaluate model performance

Recap

- Linear Regression: used in regression tasks
  - Model $\hat{y} \doteq g(\mathbf{x};\mathbf{w}) \doteq \sum_{j=1}^{m} x_j w_j = \mathbf{x}^\top \mathbf{w}$
  - Loss $f_{\mathrm{MSE}}(y,\hat{y};\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)\top}\mathbf{w} - y^{(i)})^2$
  - Exact solution

$$\mathbf{w} = \left(\sum_i \mathbf{x}^{(i)}\mathbf{x}^{(i)\top}\right)^{-1} \sum_i \mathbf{x}^{(i)} y^{(i)}$$

Why do we need DL? What is so special about neural nets?

Regression Task

- **Regression**: predict a numerical value given some input. Learning algorithm must output a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$
- **Example 1**: predict the price of houses in Boston
- **Linear regression**: $f(\mathbf{x}) = w_1x_1 + w_2x_2 + \ldots + w_mx_m + b$
  - x_1: area
  - x_2: bedrooms
  - x_3: baths
  - x_4: age
  - $x_5 \ldots x_{27}$: neighborhood

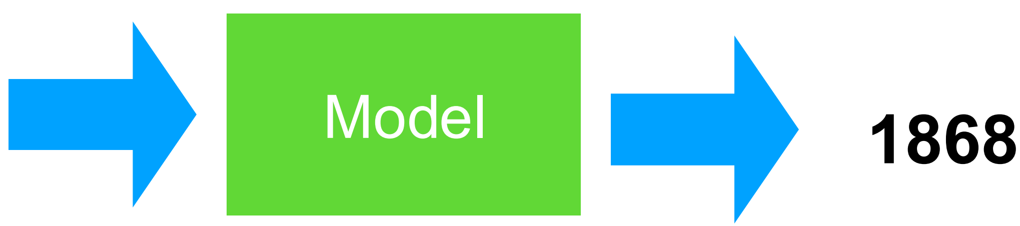
What types of things this model CANNOT account for?

Limitations of classic ML approaches

- Some don’t capture interactions between features
- Some assume linear relationship between $\mathbf{x}_i$ and y
- Require (manual) feature engineering

- **Example 2:** predict building construction date from

Boynton Hall

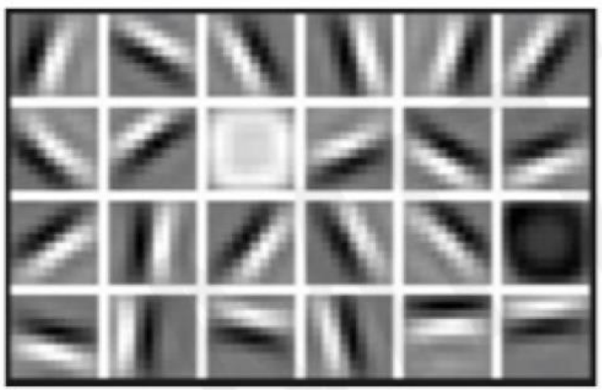


Why Deep Learning?

Hand engineered features are time consuming, brittle, and not scalable in practice

Can we learn the **underlying features** directly from data?

Low Level Features



Lines & Edges

Mid Level Features

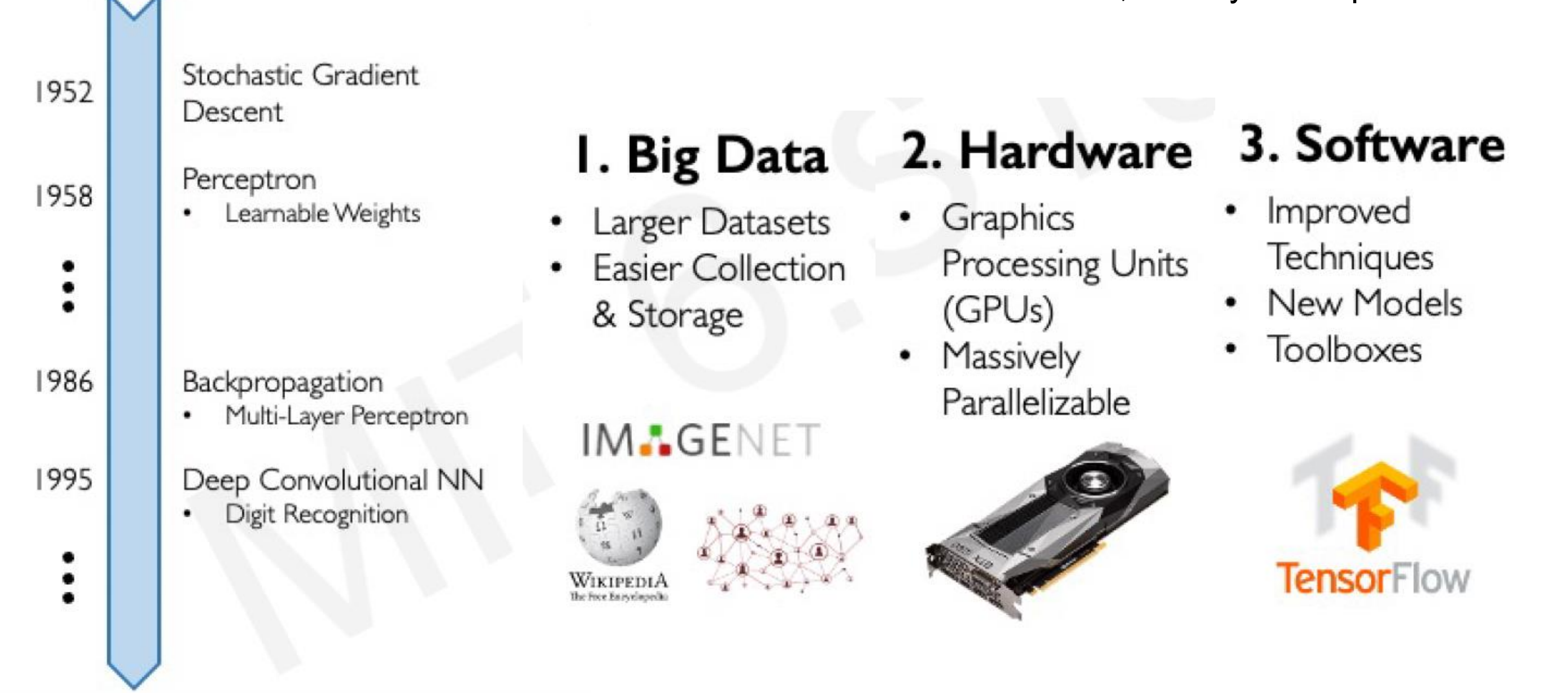
Eyes & Nose & Ears

High Level Features

Facial Structure

Why Now?

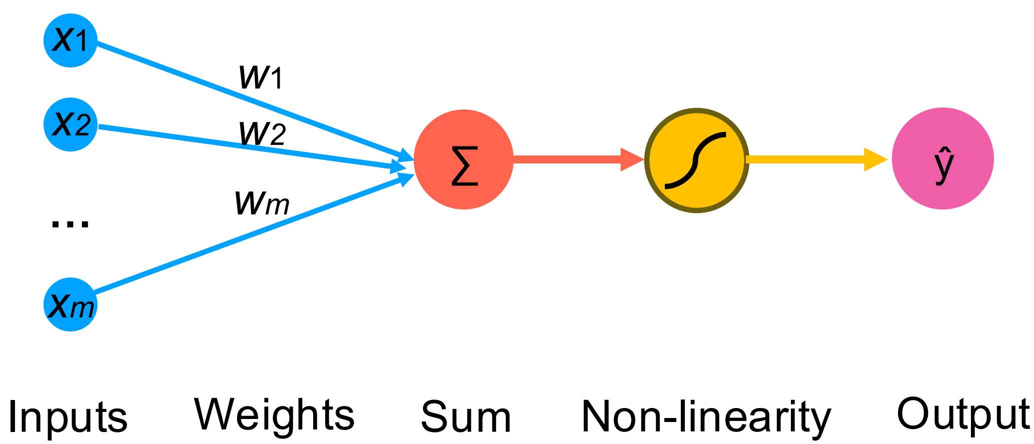
Neural Networks date back decades, so why the explosion?



The Perceptron

- Like other ML models, neural networks can be seen as mathematical **functions** $\hat{y} = f(\mathbf{x})$.

- Perceptron: the structural building block of deep learning models



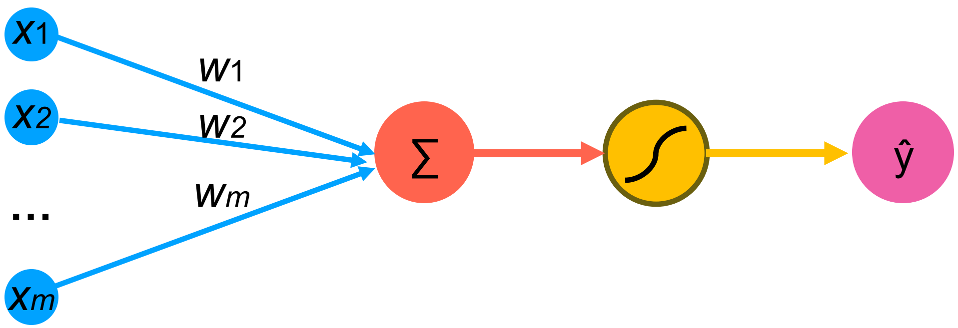
Linear combination of inputs

Output

$$\widehat{y} = g\left(\sum_{i=1}^{m} x_i\, w_i\right)$$

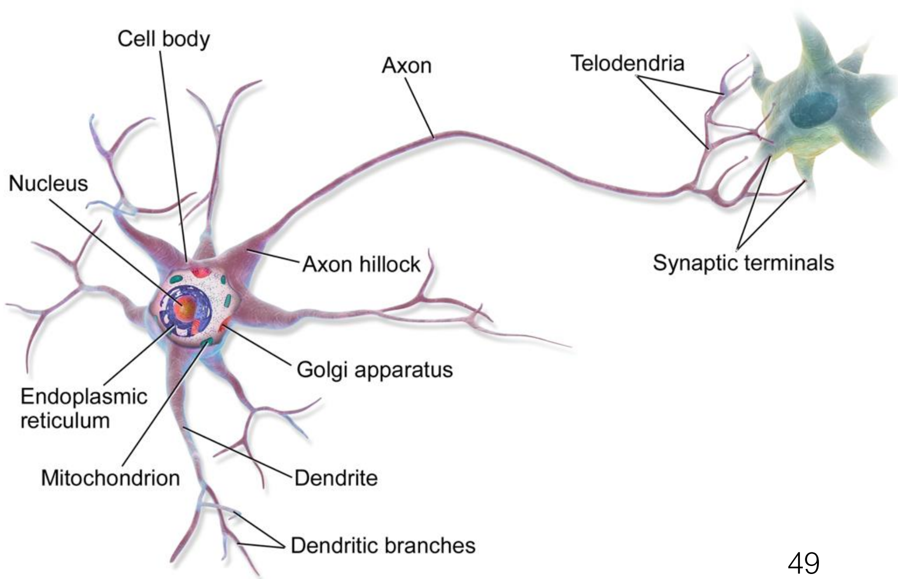
Non-linearity (aka activation function)

Compare that with Biological Neuron



$$\hat{y} = g\left(b + \sum_{i=1}^{m} x_i\, w_i\right)$$

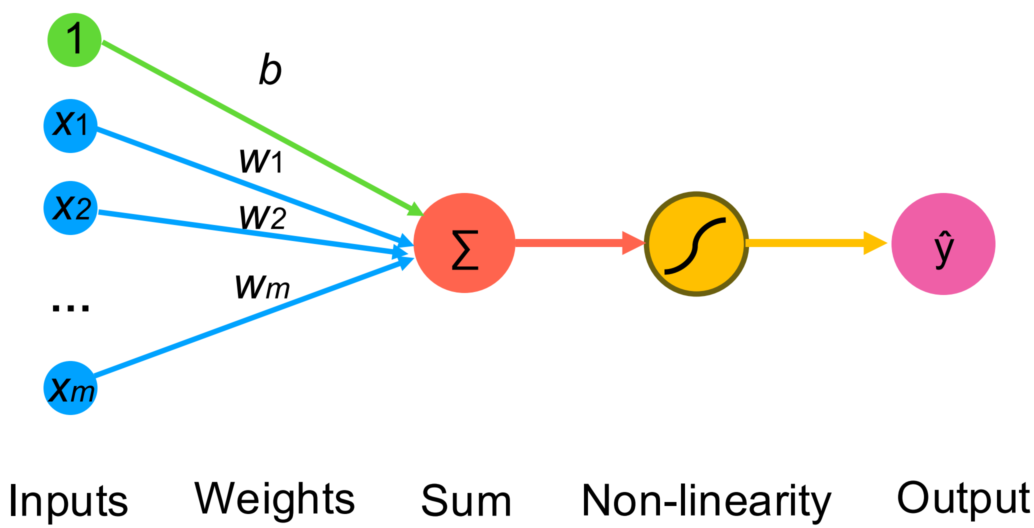
Inputs Weights Sum Non-linearity Output



In biological neural, activation g is all-or-none

The Perceptron

- It also has bias term b that shifts base level (i.e., result when all inputs are zero) even if not explicitly shown



Linear combination of inputs

Output

$$\hat{y} = g\left(b + \sum_{i=1}^{m} x_i \, w_i\right)$$

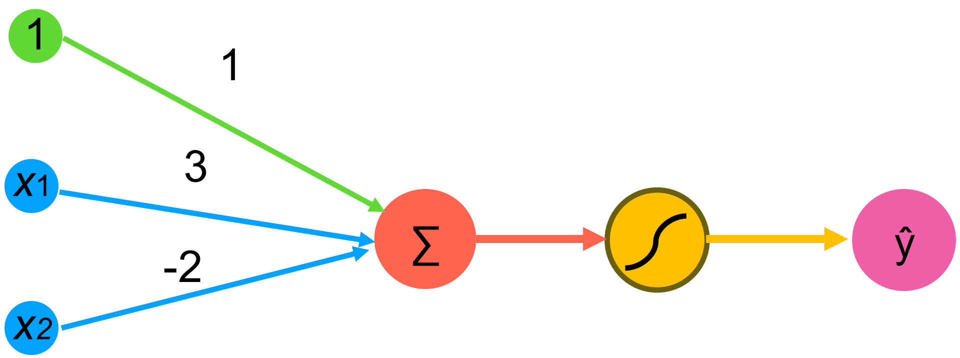
Non-linearity

bias

Vector notation (one instance): $\hat{y} = g\left(b + \mathbf{w}^\top \mathbf{x}\right)$

In this course, we assume vectors are "column", unless stated otherwise

The Perceptron: Forward Propagation

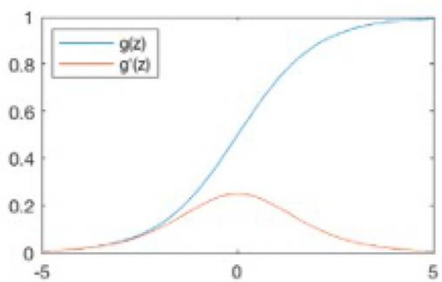


We have: b = 1 and $\mathbf{w} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\begin{aligned} \widehat{y} &= g\left(b + \mathbf{x}^\top \mathbf{w} \right) \\ &= g\left(1 + \begin{bmatrix} 3 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \\ &= g\left(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}} \right) \end{aligned}$$

Common Activation Functions

Sigmoid Function

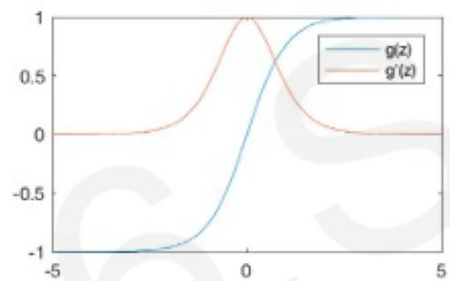


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

```
torch.sigmoid(z)
```

Hyperbolic Tangent

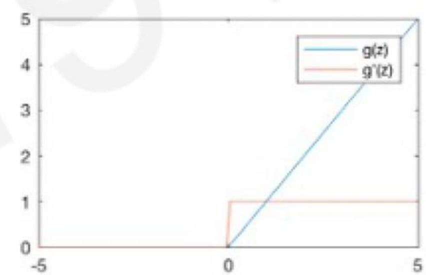


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

```
torch.tanh(z)
```

Rectified Linear Unit (ReLU)



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

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

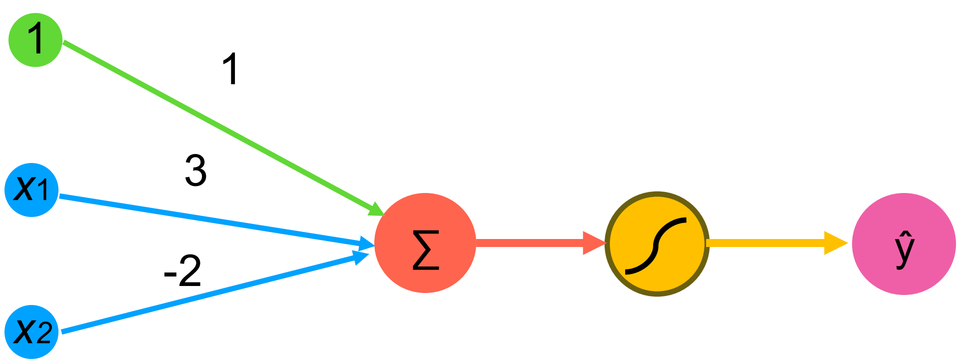
```
torch.relu(z)
```

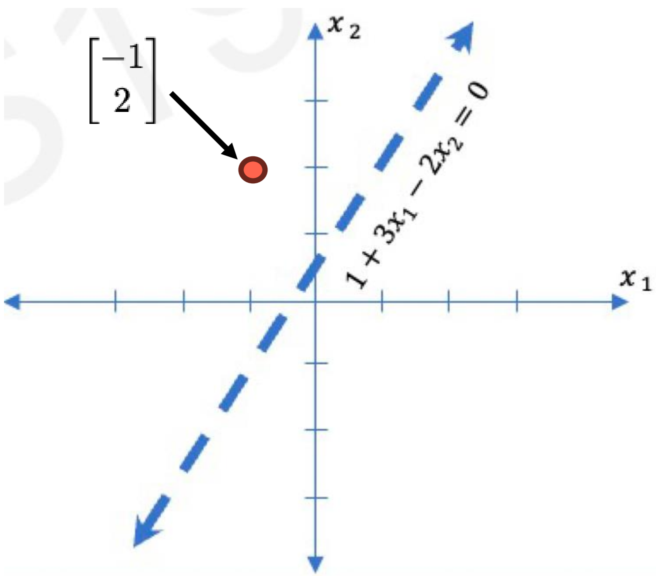
NOTE: All activation functions are non-linear

Pytorch code blocks

The Perceptron: Forward Propagation

$$\hat{y} = g\,(\,1 + 3x_1 - 2x_2)$$

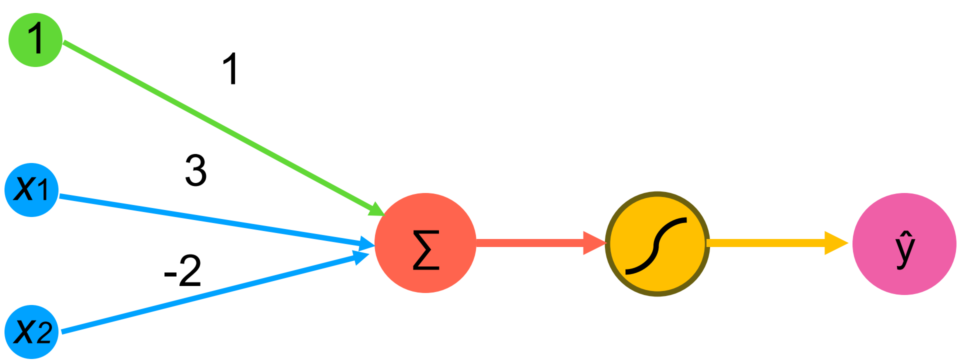




Assume we have input: $\mathbf{x} = \begin{bmatrix}-1\\2\end{bmatrix}$

$$\begin{aligned}\hat{y} &= g\left(1 + (3 * -1) - (2 * 2)\right)\\ &= g\,(-6) \approx 0.002\end{aligned}$$

The Perceptron: Forward Propagation



$$\hat{y} = g\,(\underbrace{1 + 3x_1 - 2x_2}_{z})$$