Robot Learning

General course information
Basics of robotics
Fundamentals of machine learning

USC University of Southern California

LiraLab

Team

Prof. Erdem Bıyık
Instructor
biyik@usc.edu

Office hours

- Erdem:
  - Friday, 9:00am – 10:00am, GCS SB3 (Floor: LL2)

Online resources

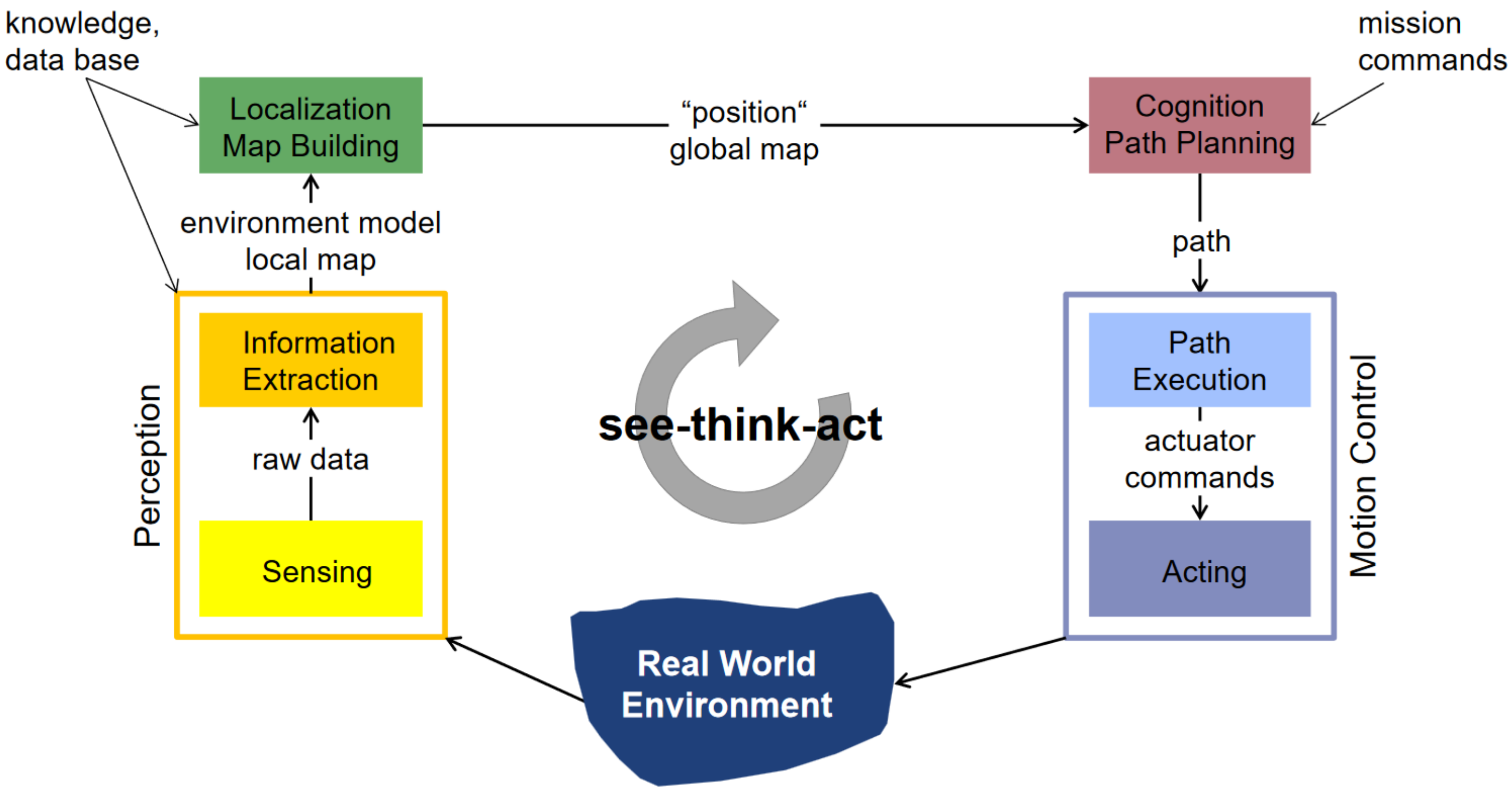
- Course website: https://liralab.usc.edu/csci699/
- Piazza: https://piazza.com/usc/fall2025/csci699specialtopics
- Gradescope: https://www.gradescope.com/courses/1073531

What is a robot?

- An embodied artificial intelligence
- A machine that can autonomously carry out useful work
- An artificial device that can sense its environment and purposefully act on or in that environment

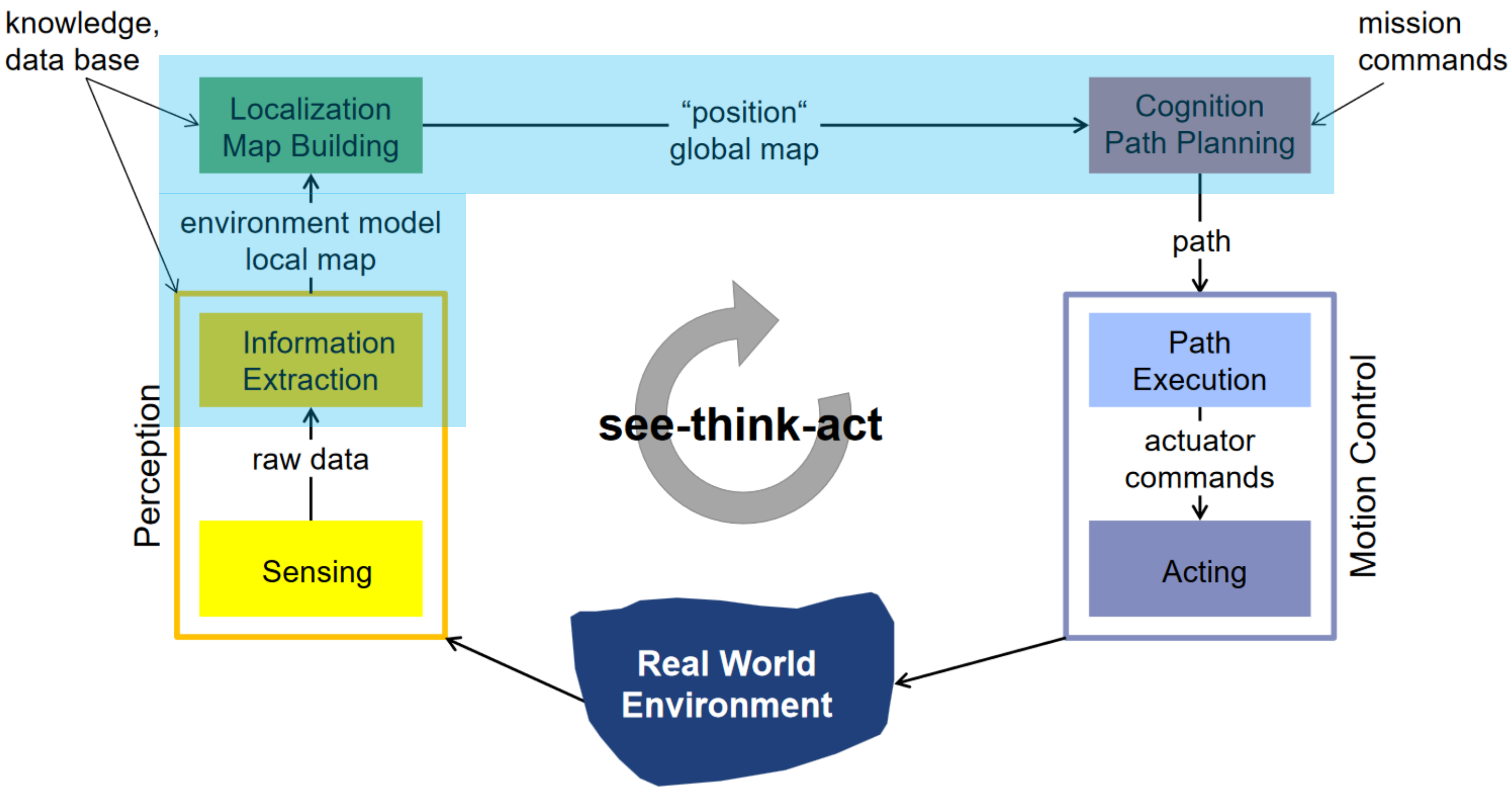
Robotics: A Very Short Introduction
Winfield, 2012

See-think-act cycle



Autonomous Mobile Robots
Siegwart et al.

Robot learning



Autonomous Mobile Robots
Siegwart et al.

Why robot learning?

Designing controllers is hard

- Requires good understanding of the system
- Doesn’t scale well to high-dimensional systems
- *“Manipulation breaks all the rigorous/reliable approaches I know for control.”*
  – Russ Tedrake (MIT / TRI)

Prerequisites

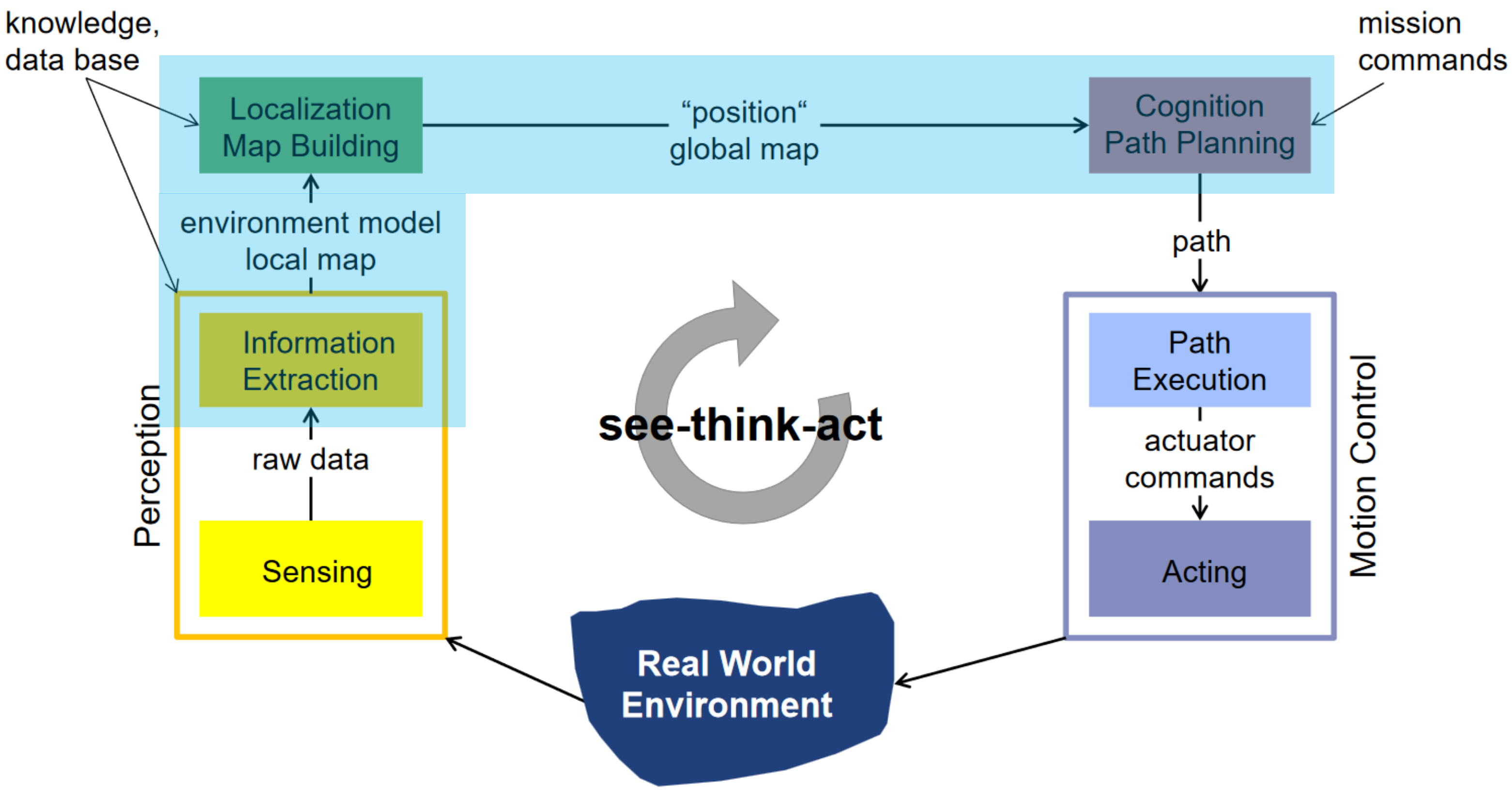
- Probability theory
- Calculus
- Linear algebra
- At least one programming language (preferably, Python)
    - Programming assignments will be in Python.

- Recommended:
    - Familiarity with basic concepts in machine learning

What’s covered?

- Basics of…
  - Robotics
  - Machine learning
  - Computer vision
- Representation learning
- Reinforcement learning
- Imitation learning / IRL
- Learning from human feedback
- Sim-to-real transfer
- Meta-learning
- Safe and robust learning
- Multi-agent learning
- Robot learning using natural language

What’s NOT covered?



Autonomous Mobile Robots
Siegwart et al.

What's NOT covered?

- Robot operating system (ROS)
  - CSCI 545: Robotics
- Simultaneous localization and mapping (SLAM)
  - CSCI 545: Robotics
- Grasping and manipulation
  - CSCI 699: Deep Learning for Robotic Manipulation
- Haptics
  - CSCI 649: Haptic Interfaces and Virtual Environments

Textbook & Readings

- No textbook is required.

- All readings will be available on course website.

- If I were to recommend textbooks for this class…
  - Reinforcement Learning: An Introduction by Sutton and Barto
  - Modern Adaptive Control and Reinforcement Learning (MACRL) by Bagnell, Boots, and Choudhury
  - Principles of Robot Autonomy by Lorenzetti and Pavone

Assignments

- One class project (40%)
- One homework assignment (20%)
- Four paper presentations (4 x 10%)

Homework assignments

- Three homework assignments in total – you choose which one you would like to do!
  - Basics of robotics & machine learning & computer vision
  - Reinforcement learning
  - Imitation learning & intent inference & shared autonomy
- Both theoretical and programming components
- Programming parts will be in Python
- No ROS knowledge required
- The submissions will be online, due at 12 midnight.

Homework assignments

- The class does not have a TA or a grader.
- They told me to be creative about how to handle grading.

- After every deadline, I will share the solution key with you.
- Everyone grades themselves in two weeks.
- I randomly pick n submissions and grade them myself.
  - If I realize the student cheated by increasing their score by x, I give a penalty of kx/n points where k is the number of students who submitted the homework.
- I repeat until I find n fairly graded assignments.

Class presentation

- 15-25 minutes presentation, depending on the week/paper
- Should include an extensive discussion of the paper
  - Motivation
  - Prior work
  - Methods
  - Results
  - Discussion
    - Both the positive and the negative aspects of the paper!
- 5 minutes Q&A

Course project

- The project will be done in groups of 2 or 3.
- Feel free to reach out to me if you have a good reason to do it individually or as a group of more than 3 students.

Course project

The project must have both robotics and machine learning components.

Examples:

- Application-dependent improvements over an existing robot learning method
- A new application of an existing robot learning technique
- A novel method that **may** have potential benefits

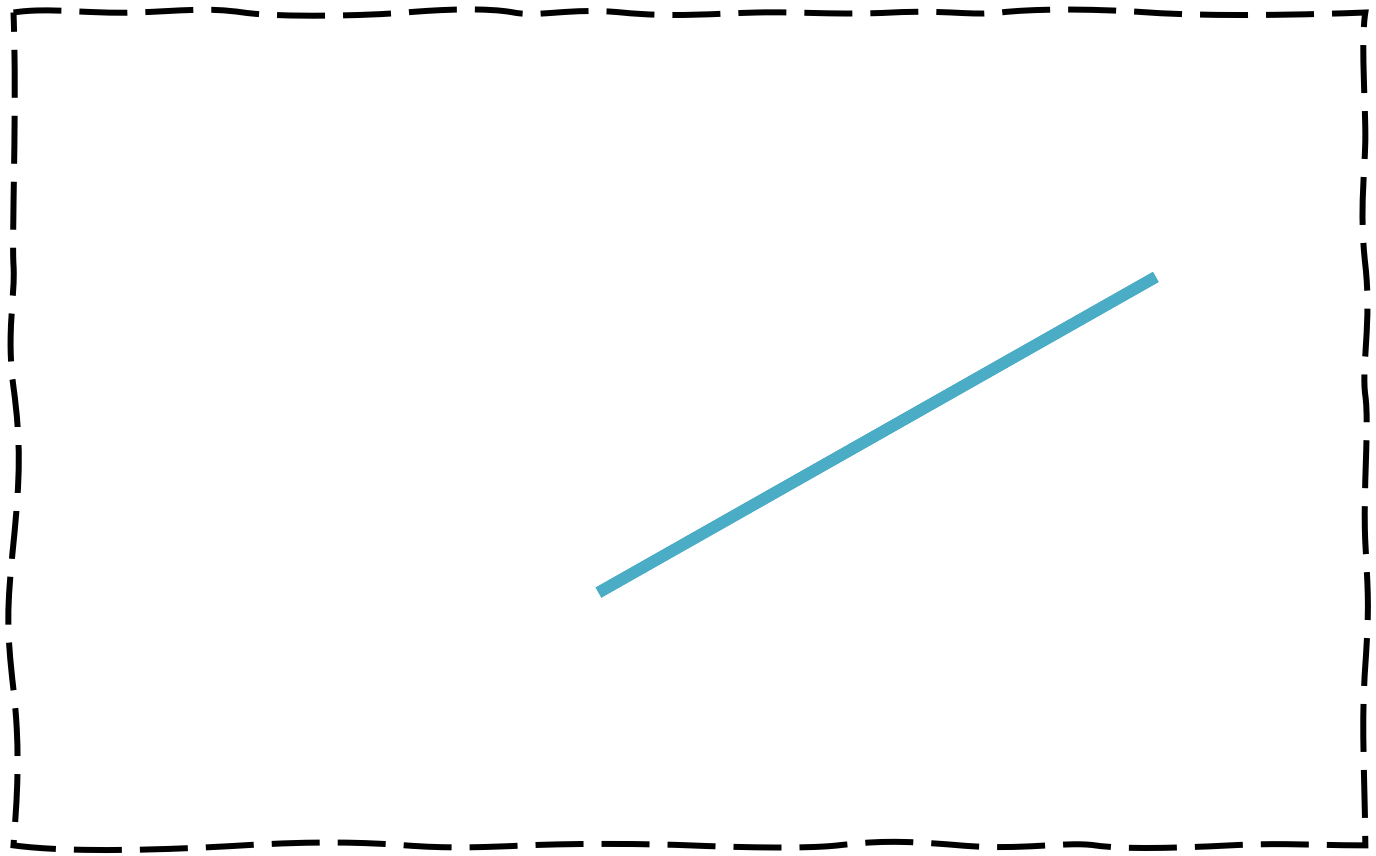
Course project

Component	Contribution to Grade	
Project Proposal Report	5%	October 19th
Project Milestone Report	5%	November 16th
Project Presentation (Possibly with Demo)	10%	December 5th
Final Project Report	15%	December 7th
Peer Review	5%	December 14th
Total	40%	

Today…

- General course information
- Basics of robotics
- Fundamentals of machine learning

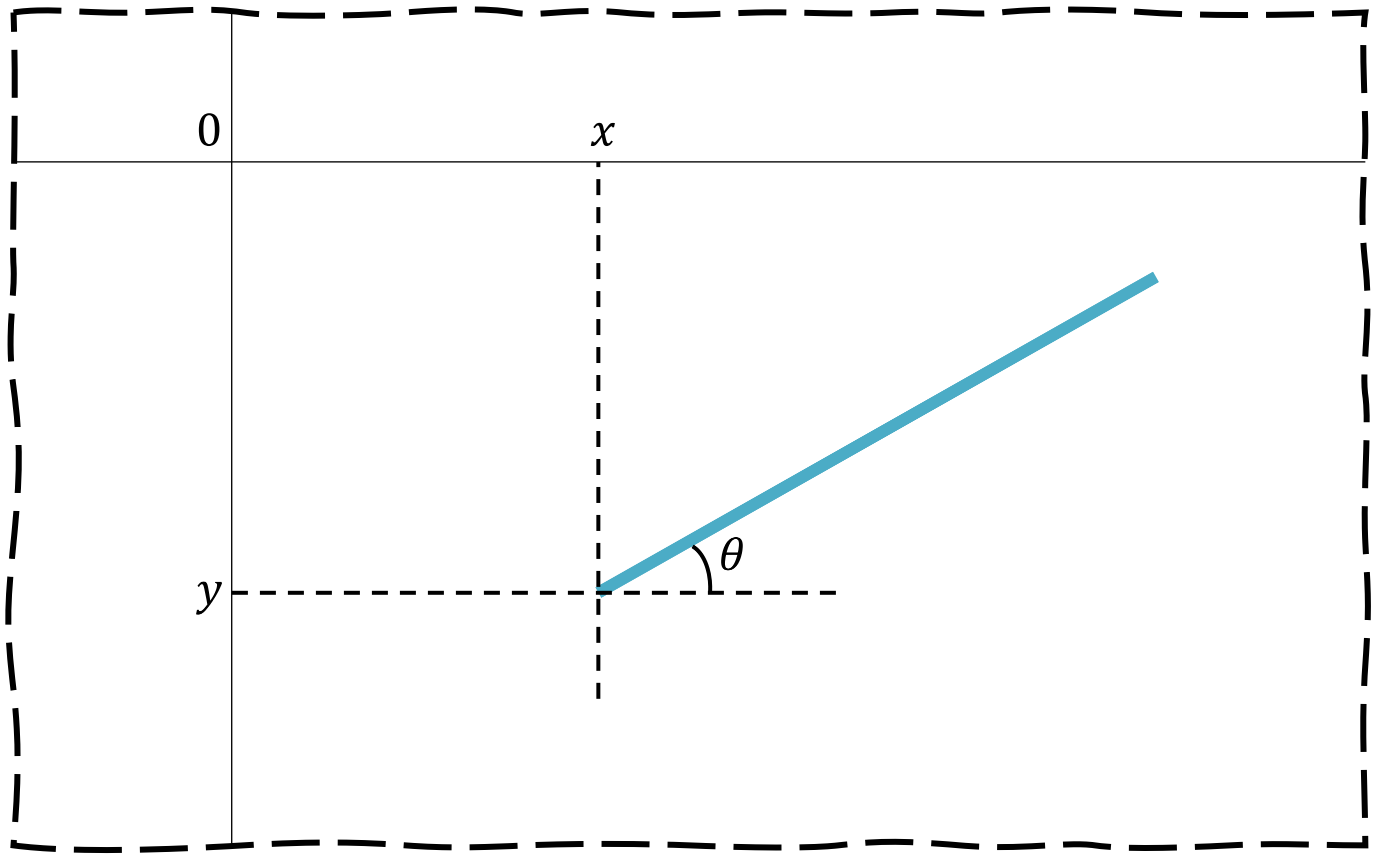
A rigid body in 2D space



This rigid body is free to move and rotate in any direction.

How many variables do we need to fully describe the configuration of this rigid body?

A rigid body in 2D space

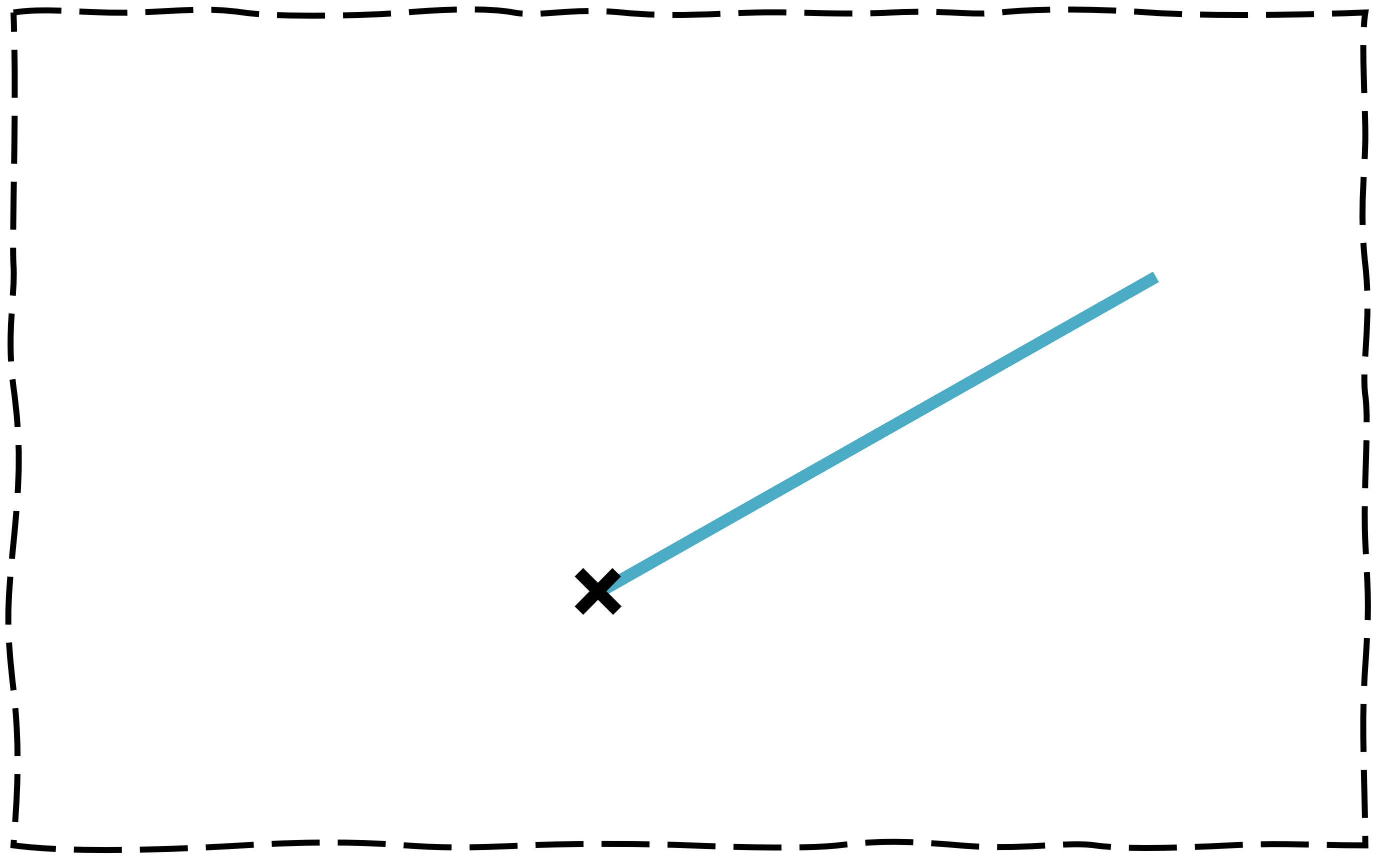


This rigid body is free to move and rotate in any direction.

How many variables do we need to fully describe the configuration of this rigid body?

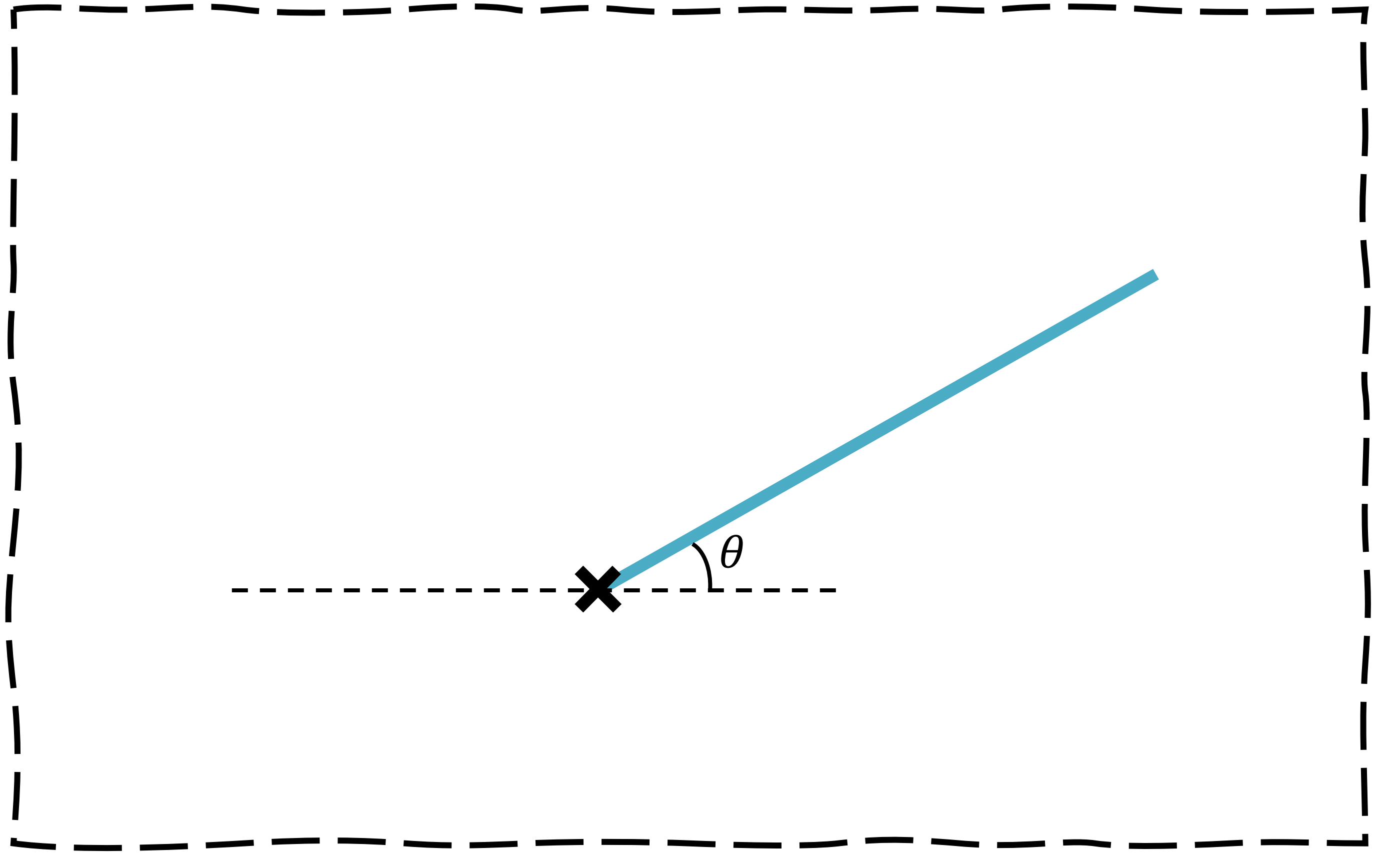
The answer is 3 variables: (x, y, θ)

A rigid body in 2D space



What if one of the end points is fixed?

A rigid body in 2D space



What if one of the end points is fixed?

Two of the variables are now fixed by two constraints:

$$x = \bar{x}$$
$$y = \bar{y}$$

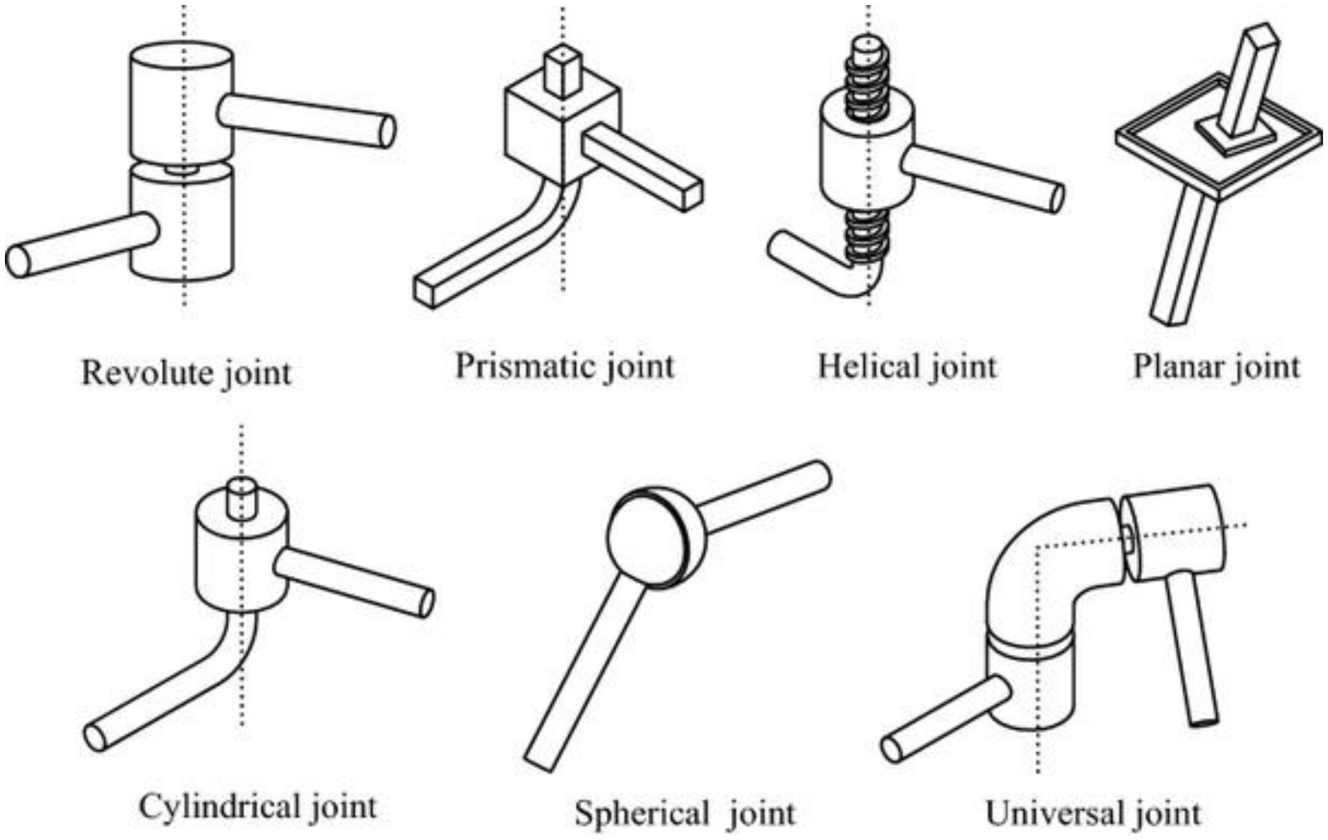
We only need one variable: θ

This is called the **degree-of-freedom (DoF)** of the body.

A rigid body in 3D space

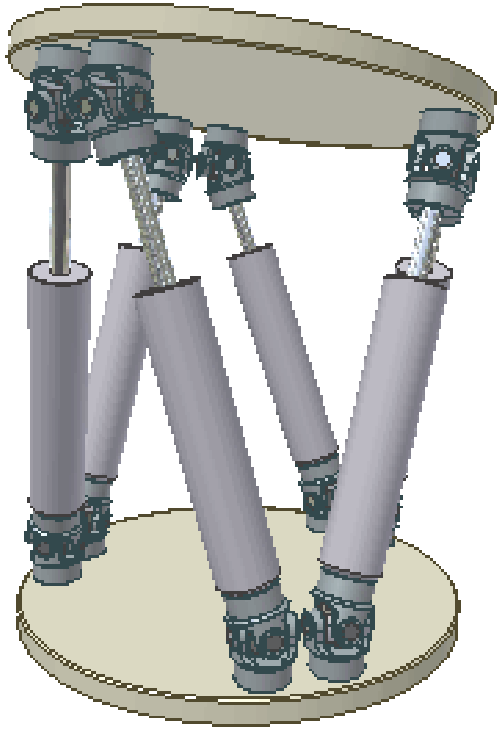
- Requires 6 degrees of freedom:
  - Three for position
  - Three for orientation

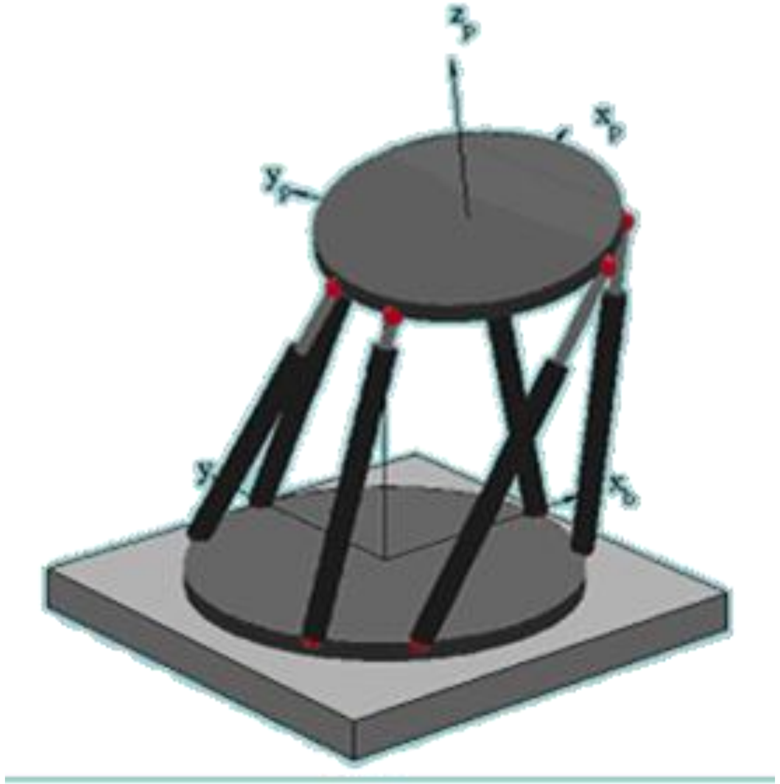
Common joints



Kinematic Design (Engineering Haptic Devices)
Andreas Röse

Degrees of freedom of a robot





Left: https://en.wikipedia.org/wiki/Stewart_platform
Right: From Flavio Firmani, University of Virginia

Grübler's formula

$N =$ # of bodies (including ground)
$J =$ # of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

Grübler’s formula

$N = \#$ of bodies (including ground)
$J = \#$ of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

$$\mathrm{dof} = \mathrm{m}(\mathrm{N} - 1) - \sum_{i=1}^{J} c_i$$

Grübler's formula

N = # of bodies (including ground)
J = # of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

$$\text{dof} = \text{m}(\text{N} - 1) - \sum_{i=1}^{J} c_i$$

Number of independent joint constraints

Grübler's formula

N = # of bodies (including ground)
J = # of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

$$\text{dof} = \text{m}(\text{N} - 1) - \sum_{i=1}^{J} c_i$$

$$= \text{m}(\text{N} - 1) - \sum_{i=1}^{J} (m - f_i)$$

Grübler's formula

N = # of bodies (including ground)
J = # of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

$$\mathrm{dof} = \mathrm{m}(\mathrm{N}-1) - \sum_{i=1}^{J} c_i$$

$$= \mathrm{m}(\mathrm{N}-1) - \sum_{i=1}^{J} (m - f_i) = \mathrm{m}(\mathrm{N}-1-\mathrm{J}) + \sum_{i=1}^{J} f_i$$

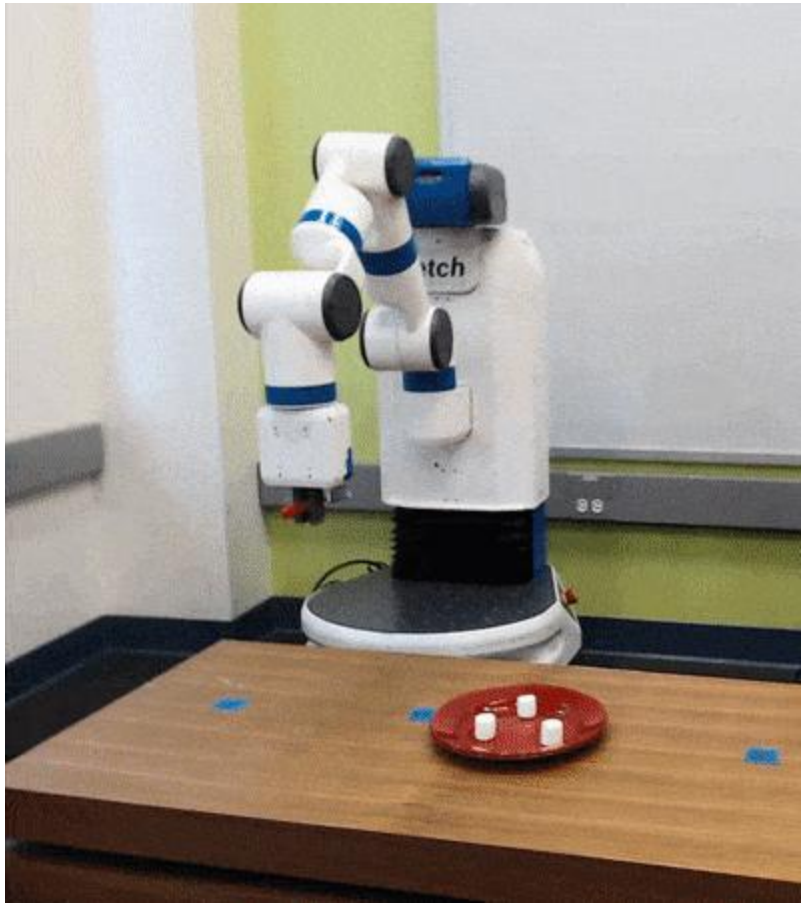
Grübler's formula

N = # of bodies (including ground)
J = # of joints

$$m = \begin{cases} 3, & \text{if planar} \\ 6, & \text{if spatial} \end{cases}$$

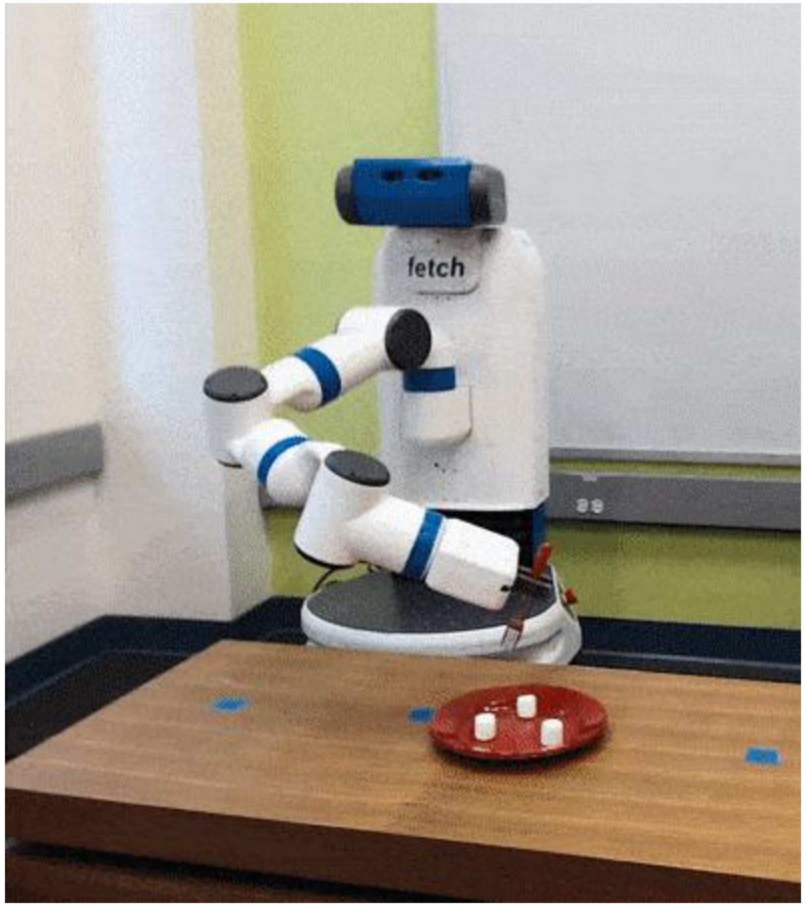
$$\mathrm{dof} = \mathrm{m}(\mathrm{N} - 1 - \mathrm{J}) + \sum_{i=1}^{J} f_i$$

An open-chain robot arm



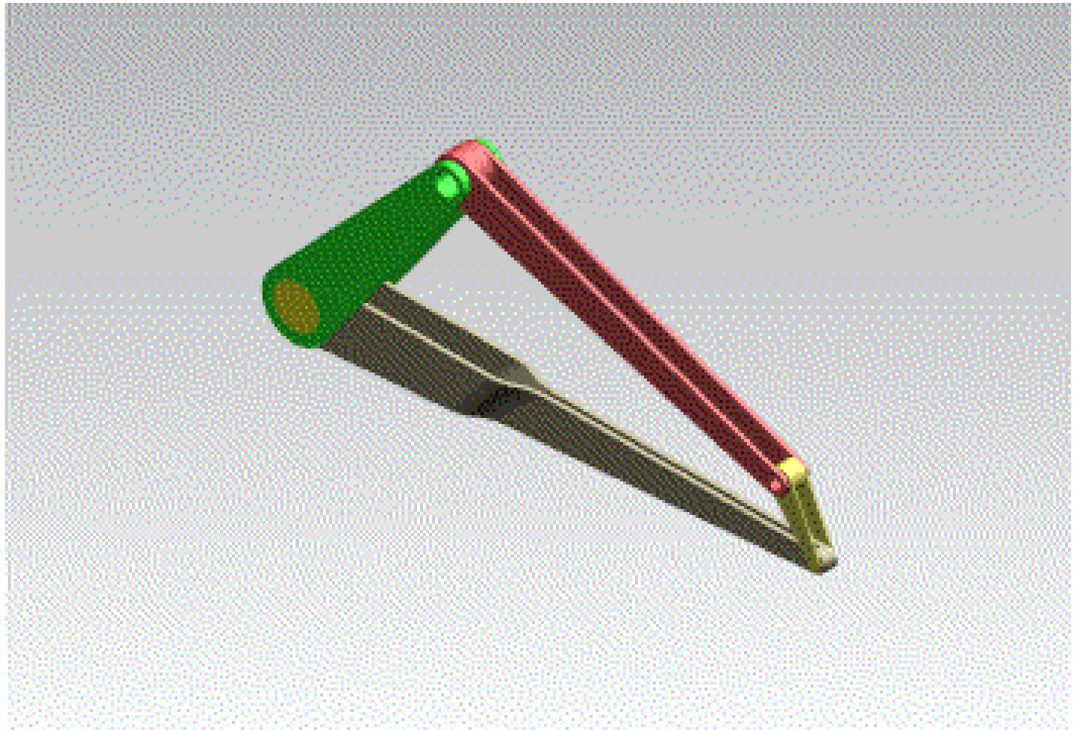
Controlling Assistive Robots with Learned Latent Actions
Dylan Losey et al., ICRA 2020

An open-chain robot arm



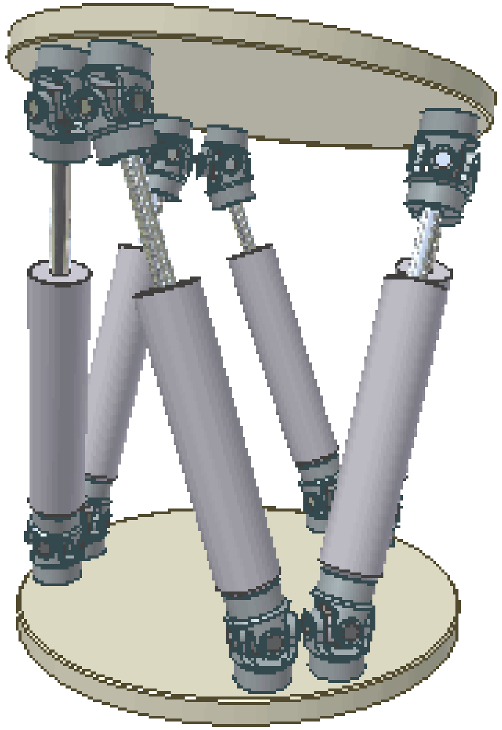
Controlling Assistive Robots with Learned Latent Actions
Dylan Losey et al., ICRA 2020

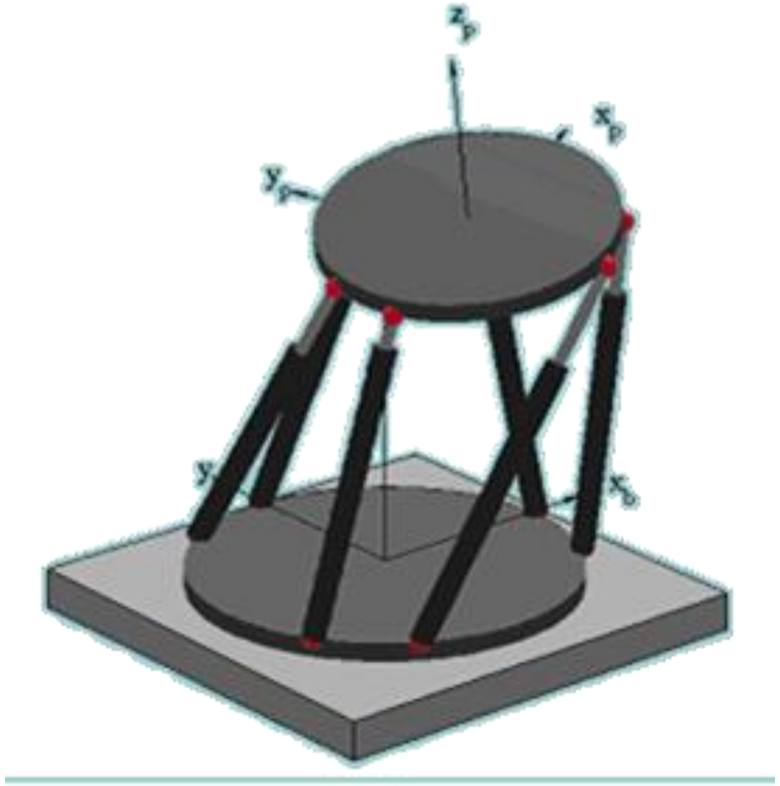
Four-bar closed-chain mechanism



From Rupesh Dewangan, GrabCAD

Stewart platform





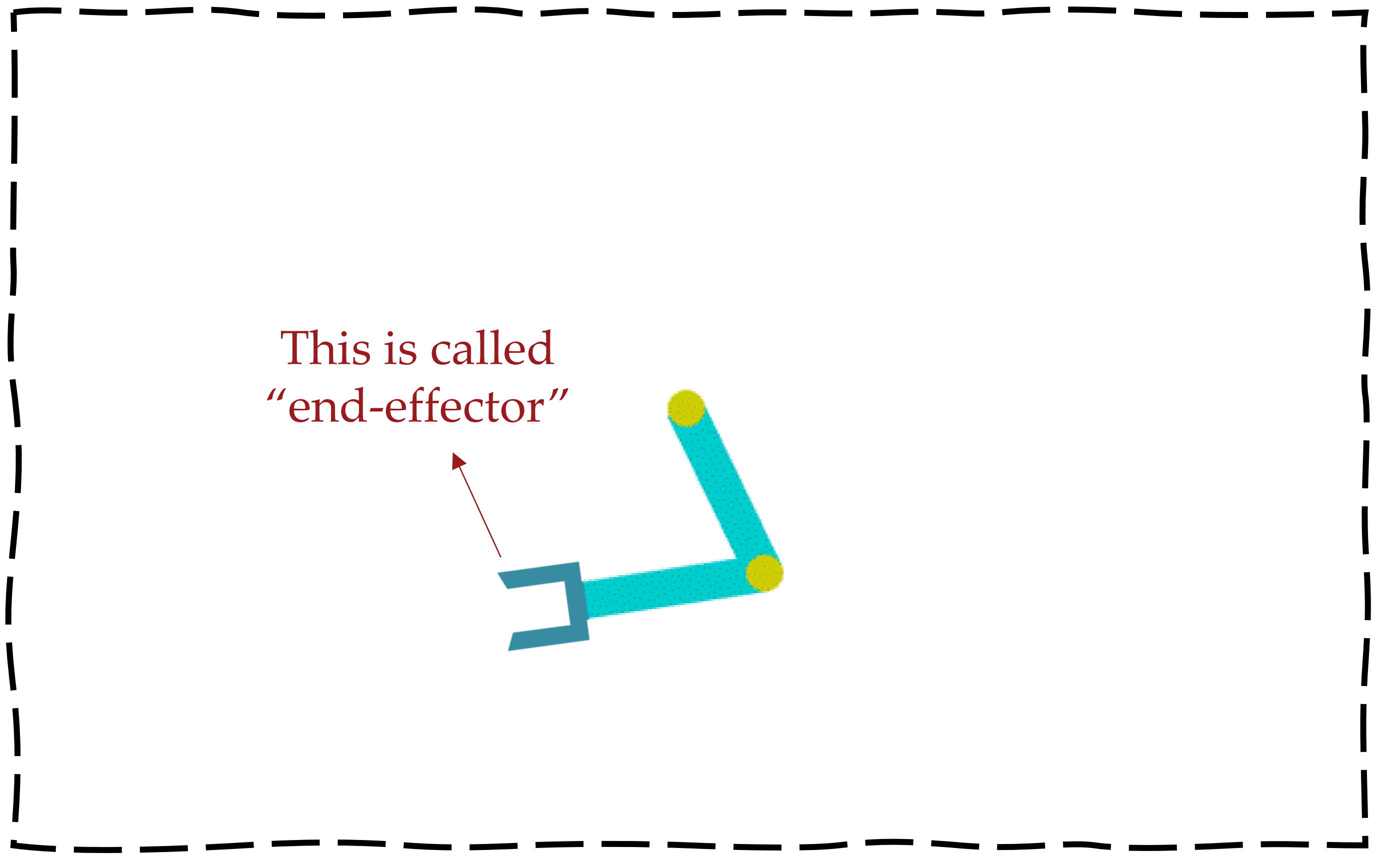
Left: https://en.wikipedia.org/wiki/Stewart_platform
Right: From Flavio Firmani, University of Virginia

From: Wikipedia

Acrobot

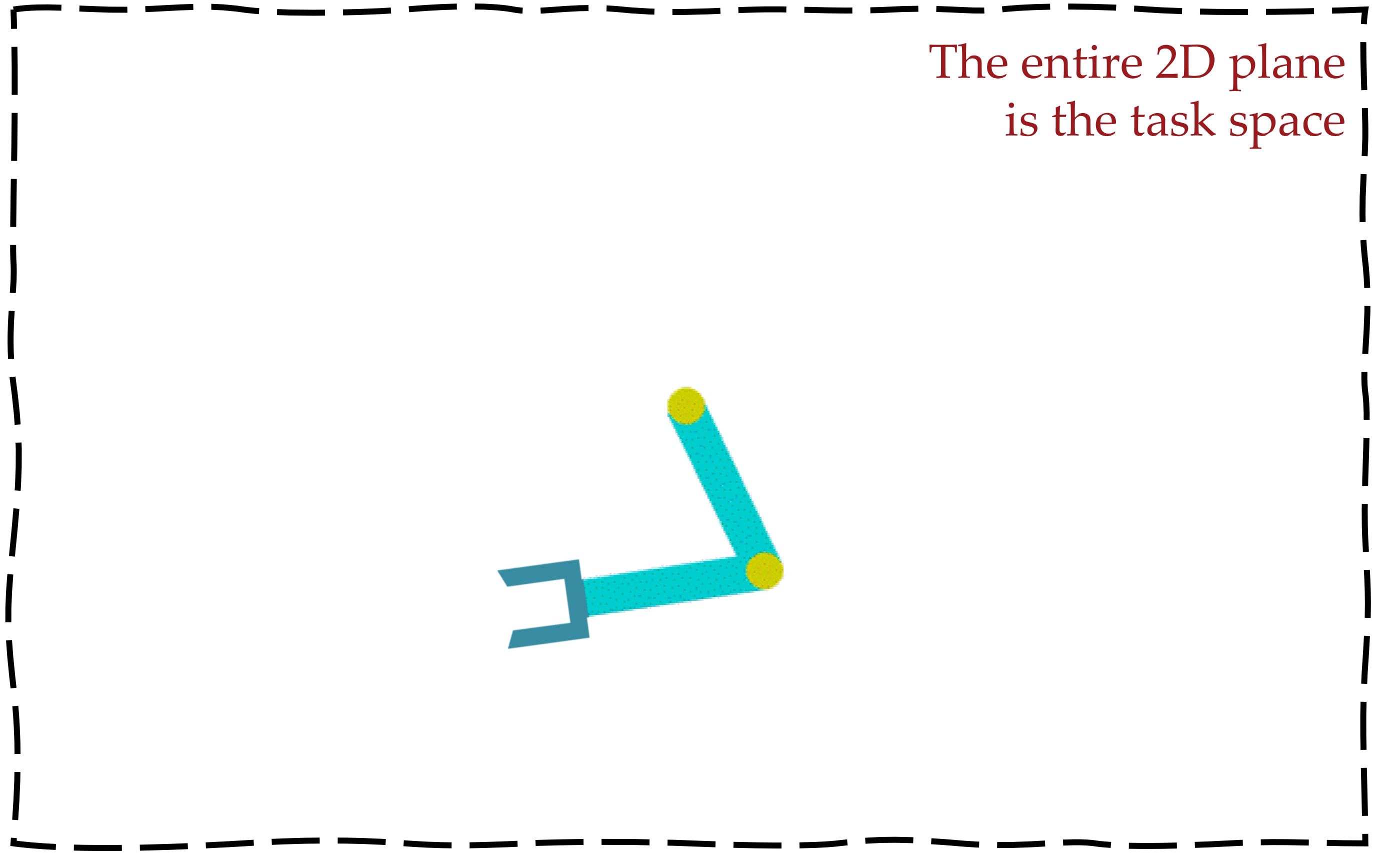
From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



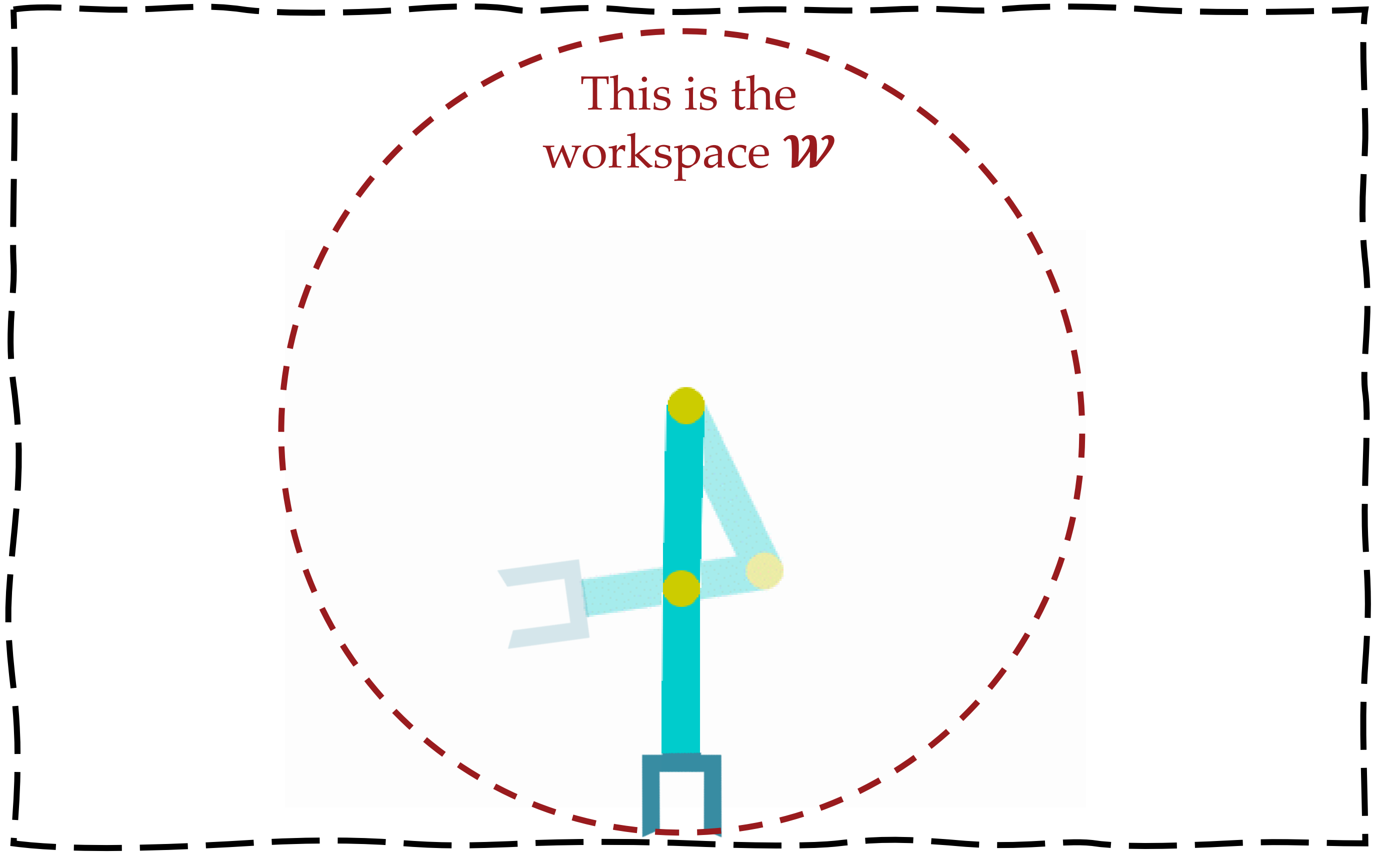
From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



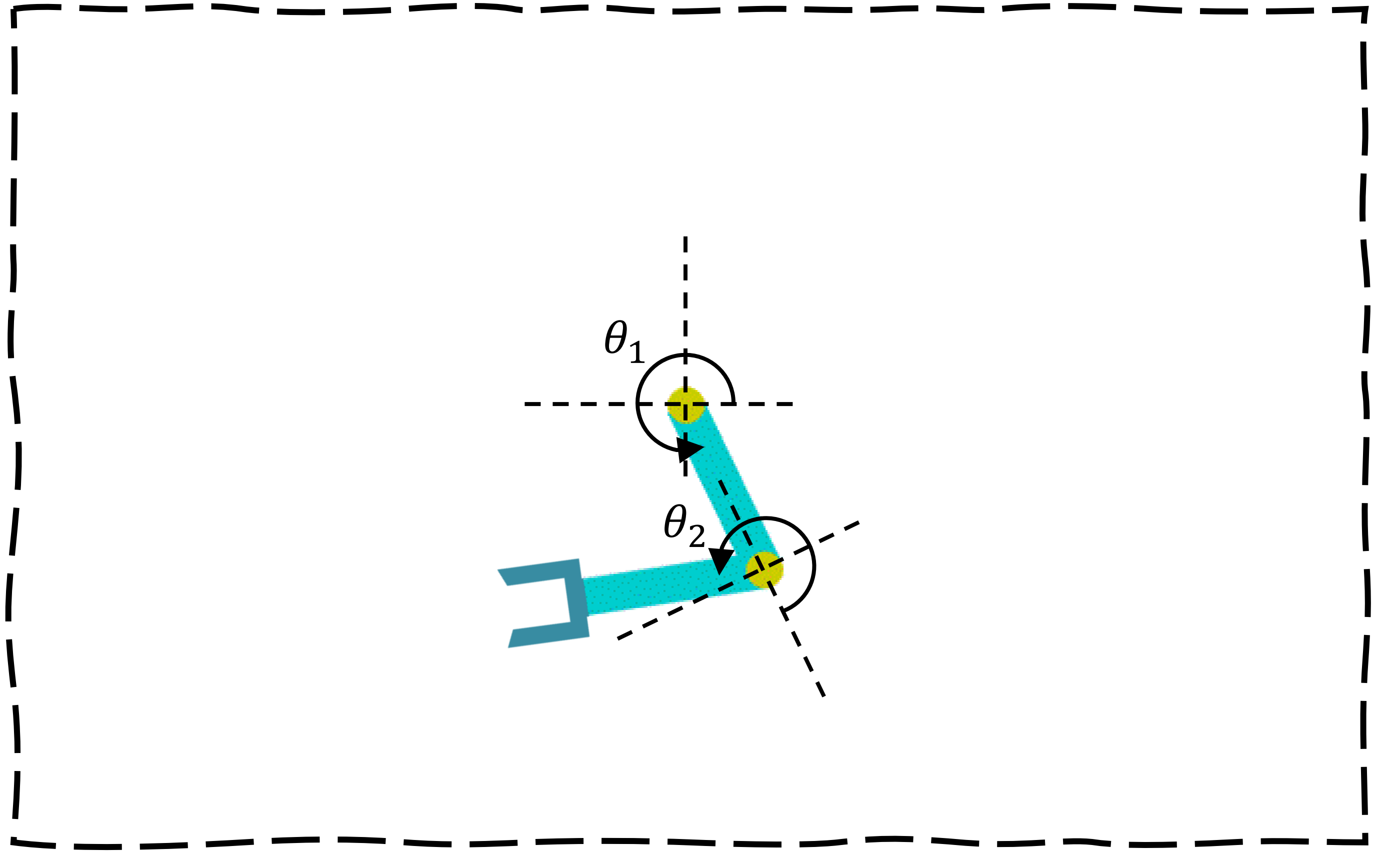
From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples
Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



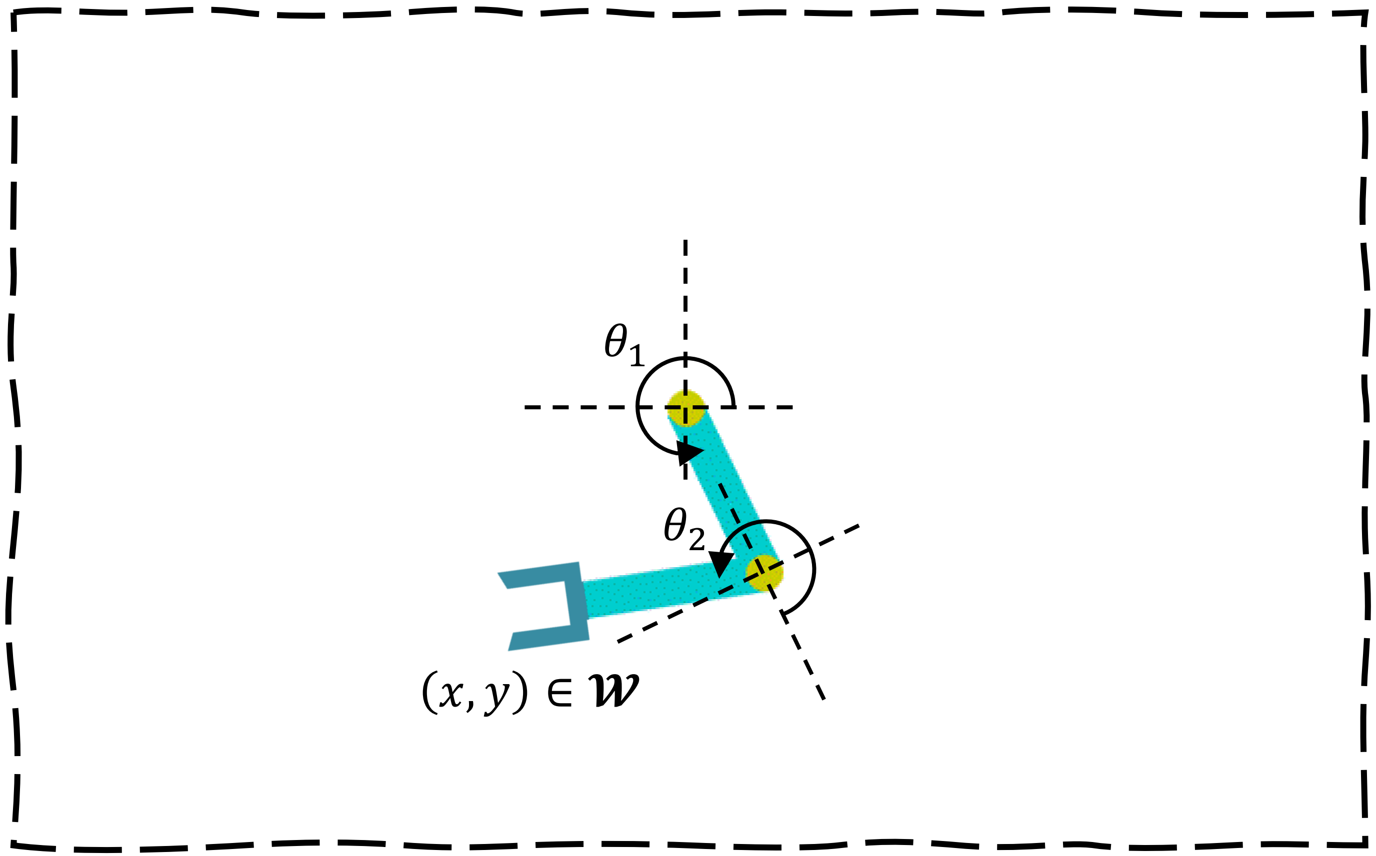
The robot's current configuration is:

$$(\theta_1, \theta_2) \in \boldsymbol{\mathcal{C}}$$

This is called the configuration space

From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



The robot's current configuration is:

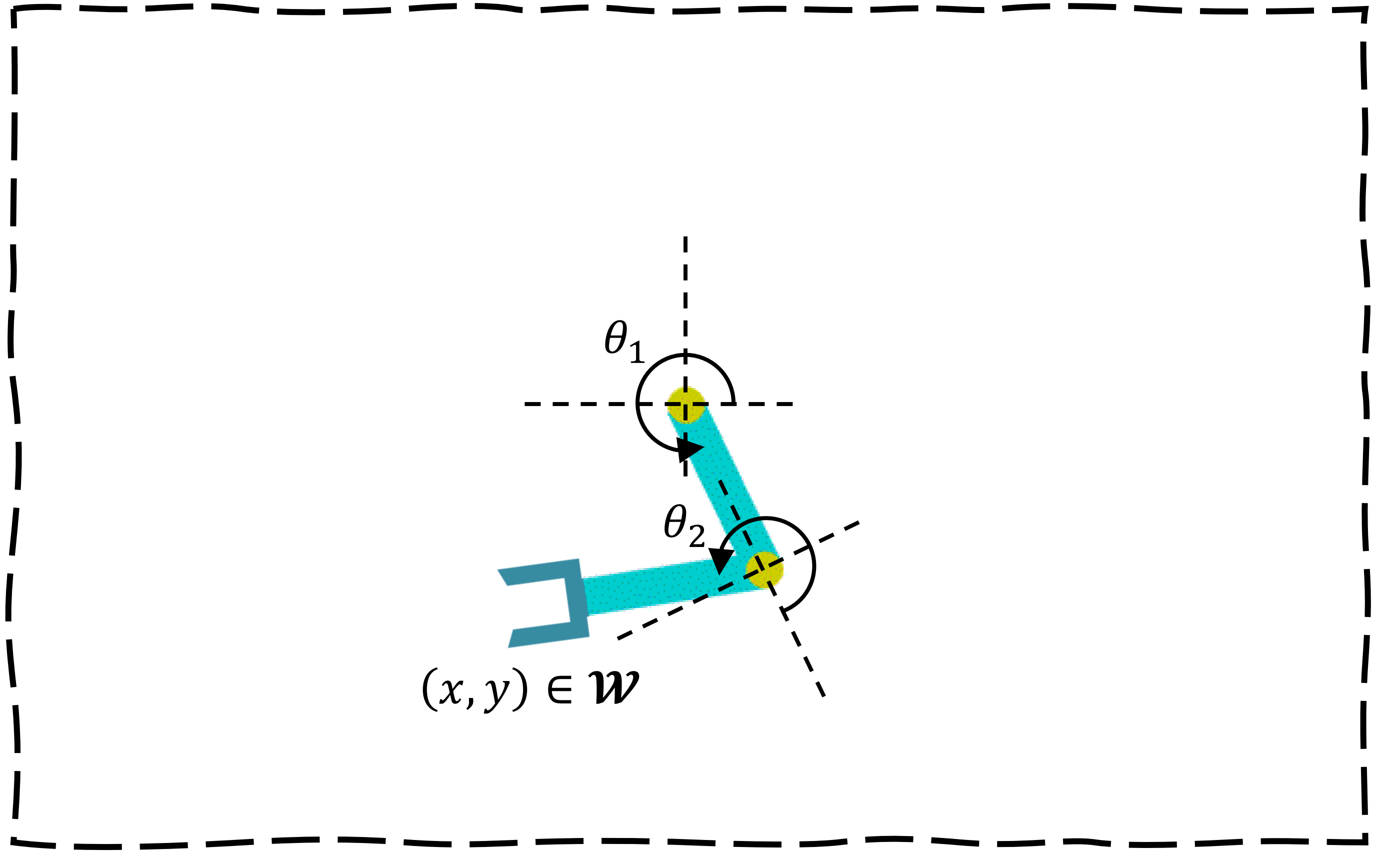
$$(\theta_1, \theta_2) \in \boldsymbol{\mathcal{C}}$$

<u>Forward kinematics</u>: FK: $\boldsymbol{\mathcal{C}} \rightarrow \boldsymbol{\mathcal{W}}$

$$\text{FK}\big((\theta_1, \theta_2)\big) = (x, y)$$

From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



The robot's current configuration is:

$$(\theta_1, \theta_2) \in \mathcal{C}$$

<u>Forward kinematics</u>: FK: $\mathcal{C} \rightarrow \mathcal{W}$

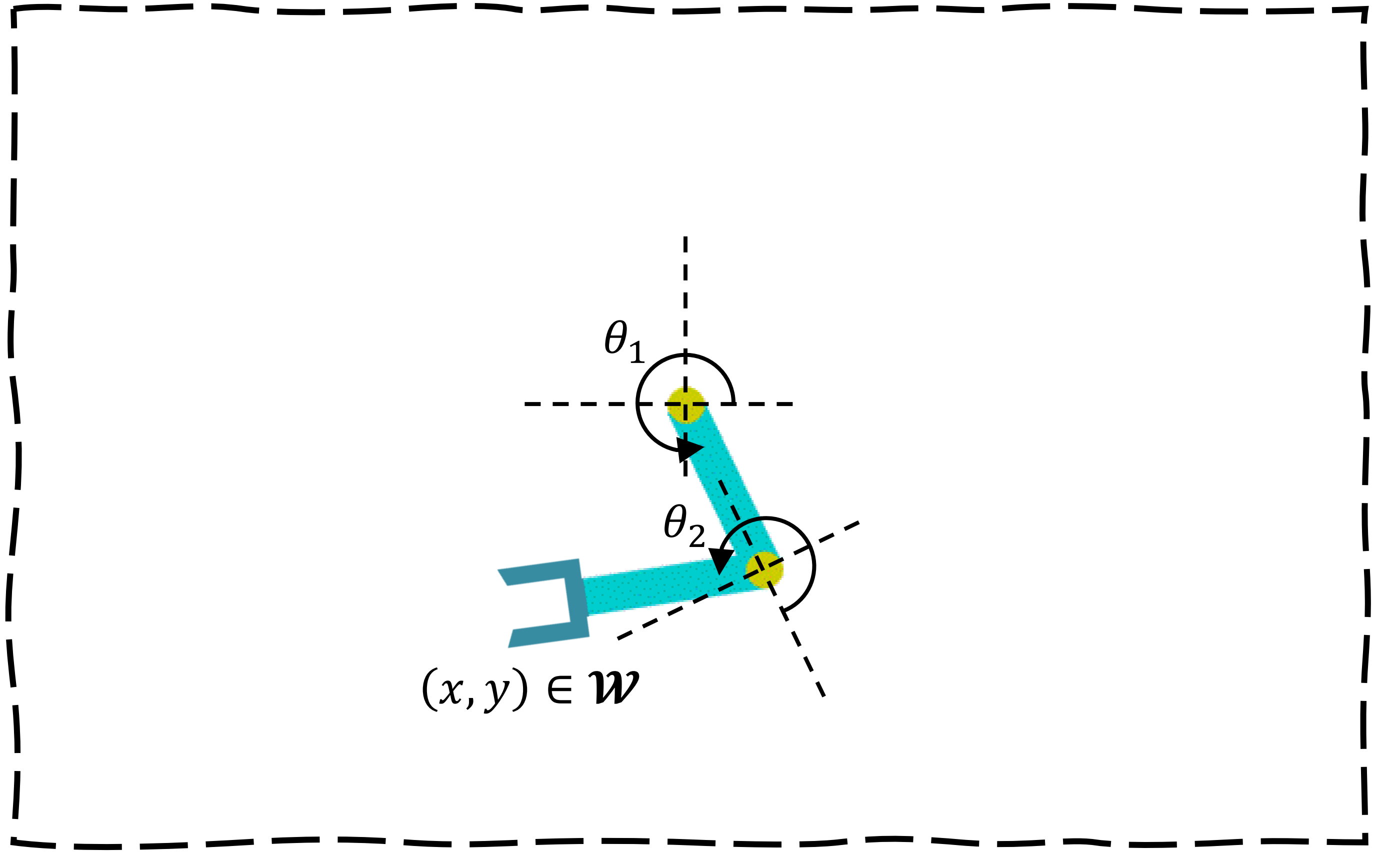
$$\mathrm{FK}\big((\theta_1, \theta_2)\big) = (x, y)$$

<u>Inverse kinematics</u>: IK: $\mathcal{W} \rightarrow \mathcal{C}$

$$\mathrm{IK}\big((x, y)\big) = (\theta_1, \theta_2)$$

From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

Acrobot



The robot's current configuration is:

$$(\theta_1, \theta_2) \in \boldsymbol{\mathcal{C}}$$

Forward kinematics: FK: $\boldsymbol{\mathcal{C}} \rightarrow \boldsymbol{\mathcal{W}}$

$$\mathrm{FK}\big((\theta_1, \theta_2)\big) = (x, y)$$

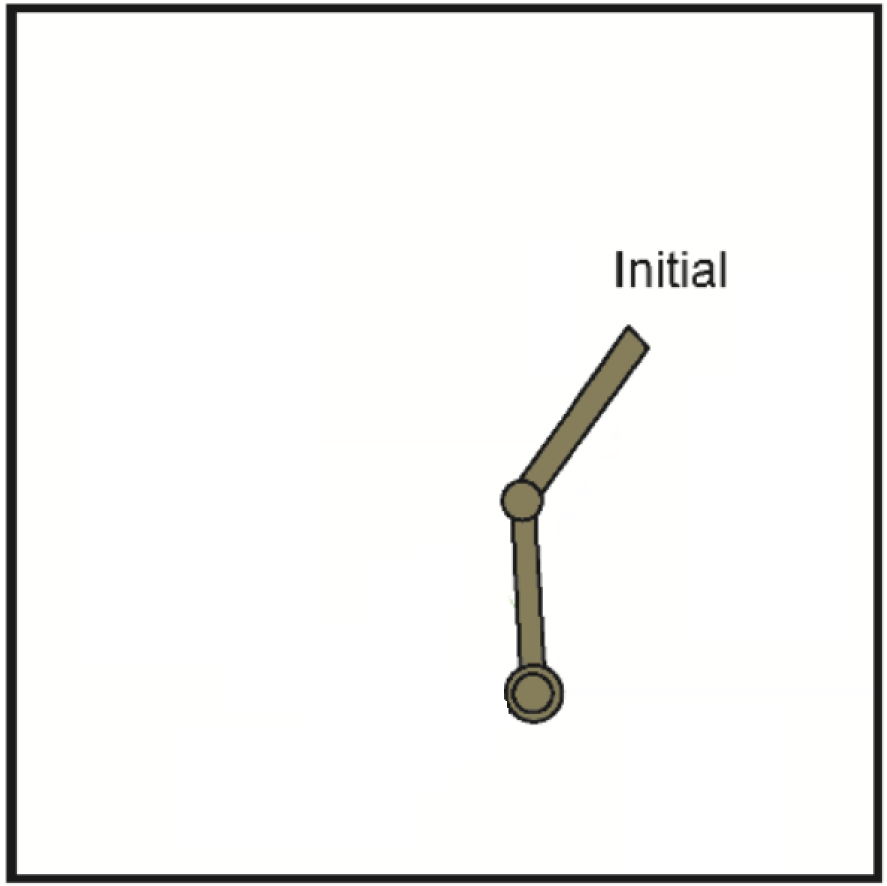
Inverse kinematics: IK: $\boldsymbol{\mathcal{W}} \rightarrow \boldsymbol{\mathcal{C}}$

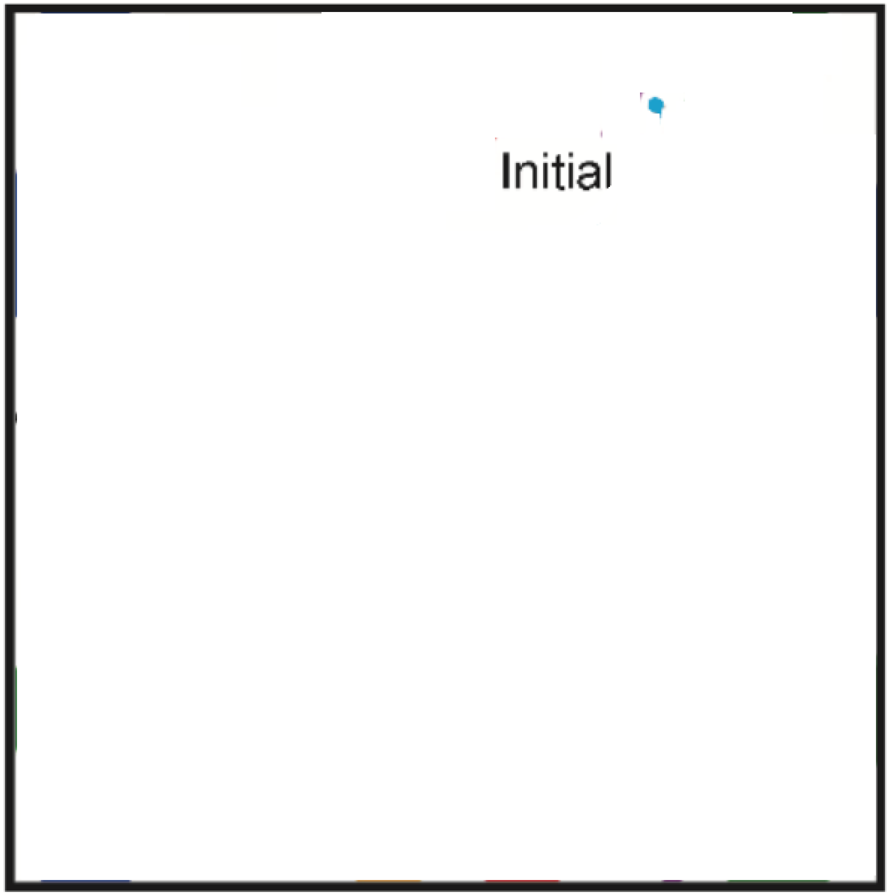
$$\mathrm{IK}\big((x, y)\big) = (\theta_1, \theta_2)$$

This is often not a proper function.
Because many configurations may
lead to the same end-effector pose.

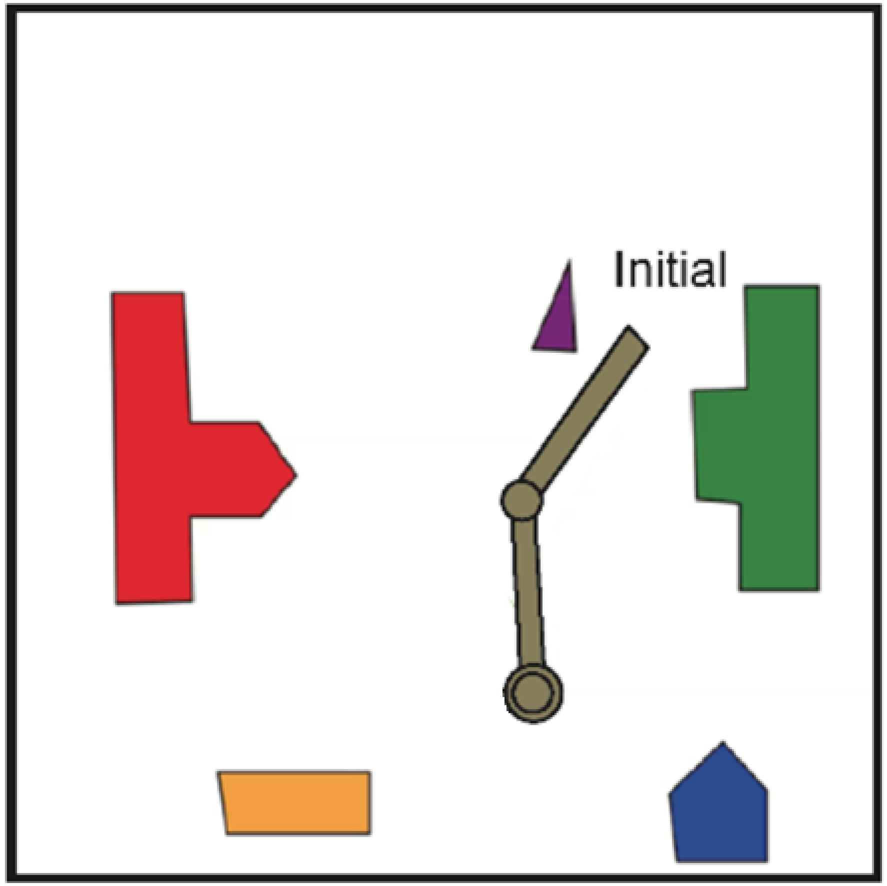
From: Gymnasium, Farama Foundation
Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding
Richard S. Sutton, NeurIPS 1995

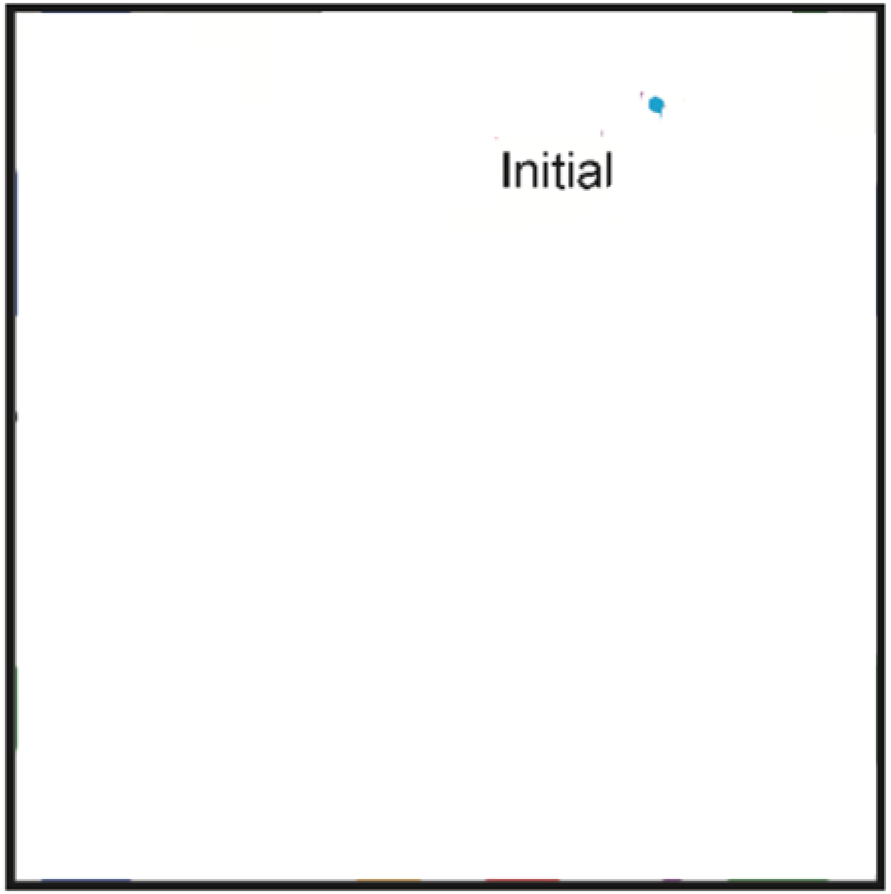
Okay, but why?



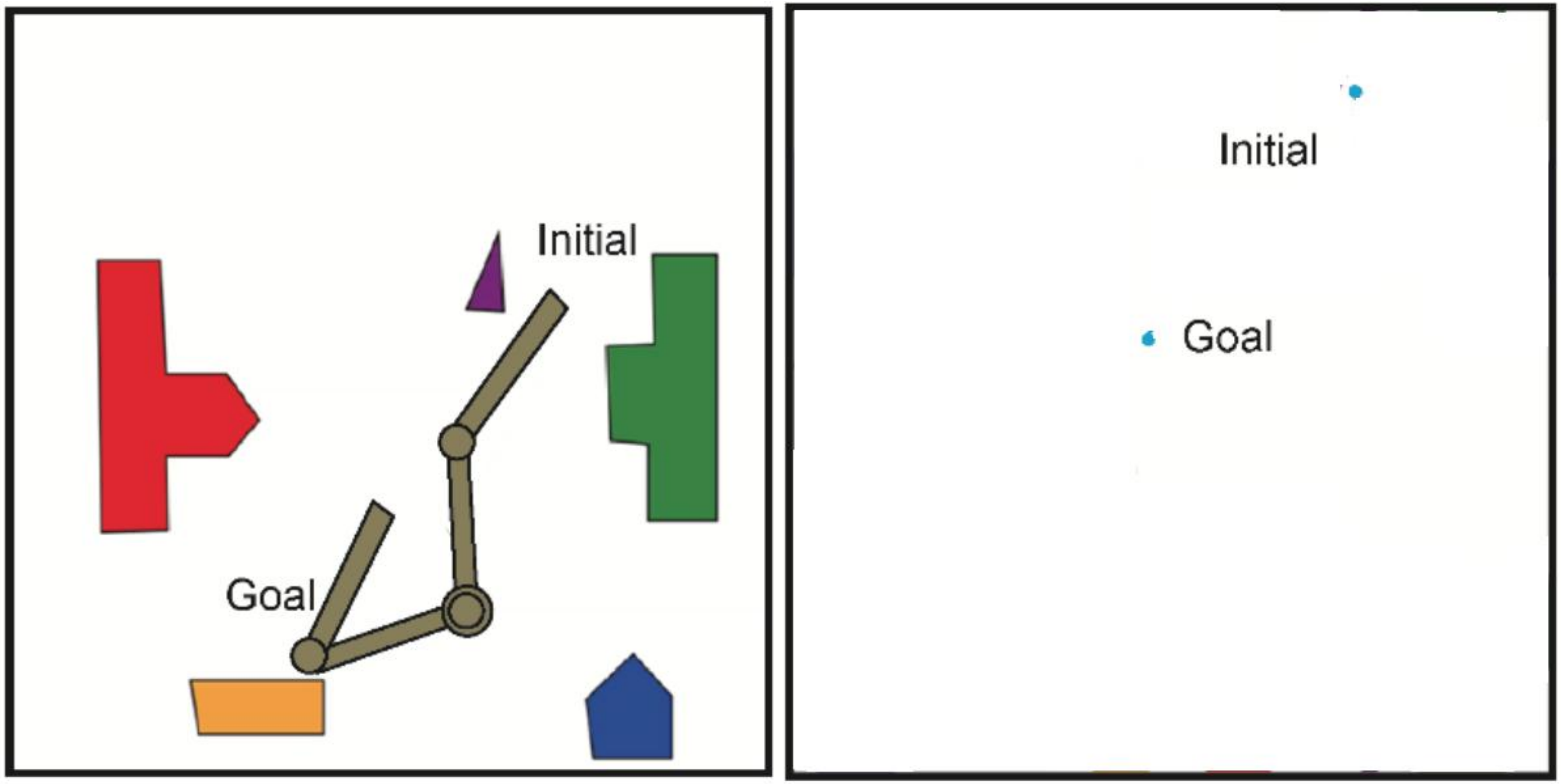


Okay, but why?

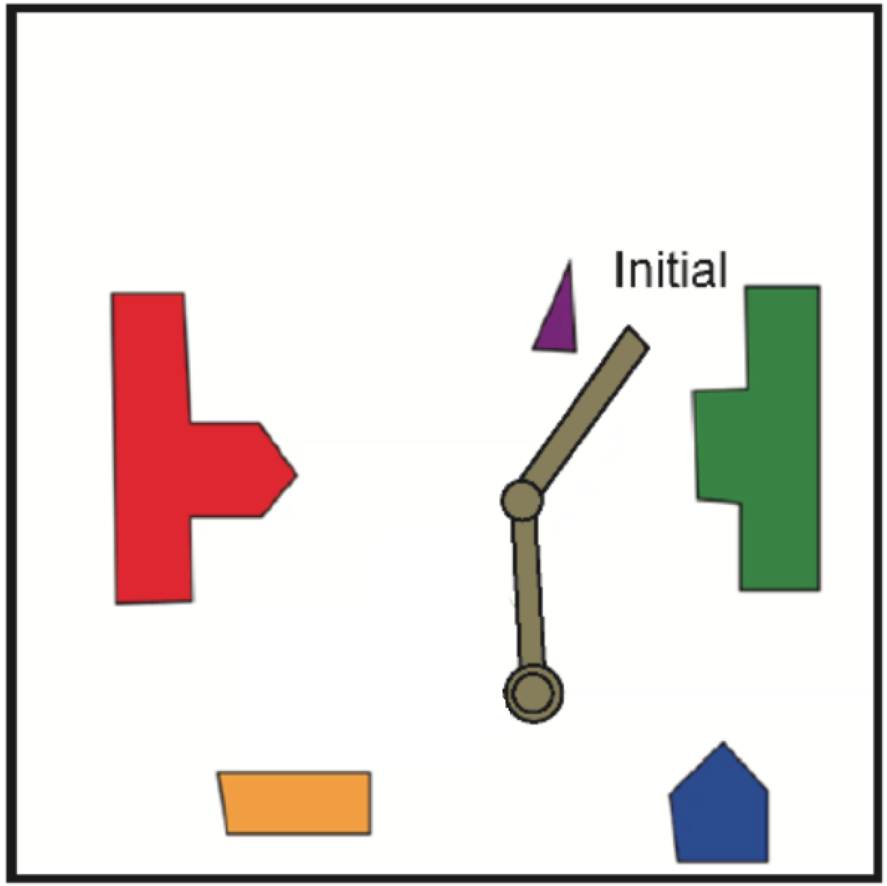


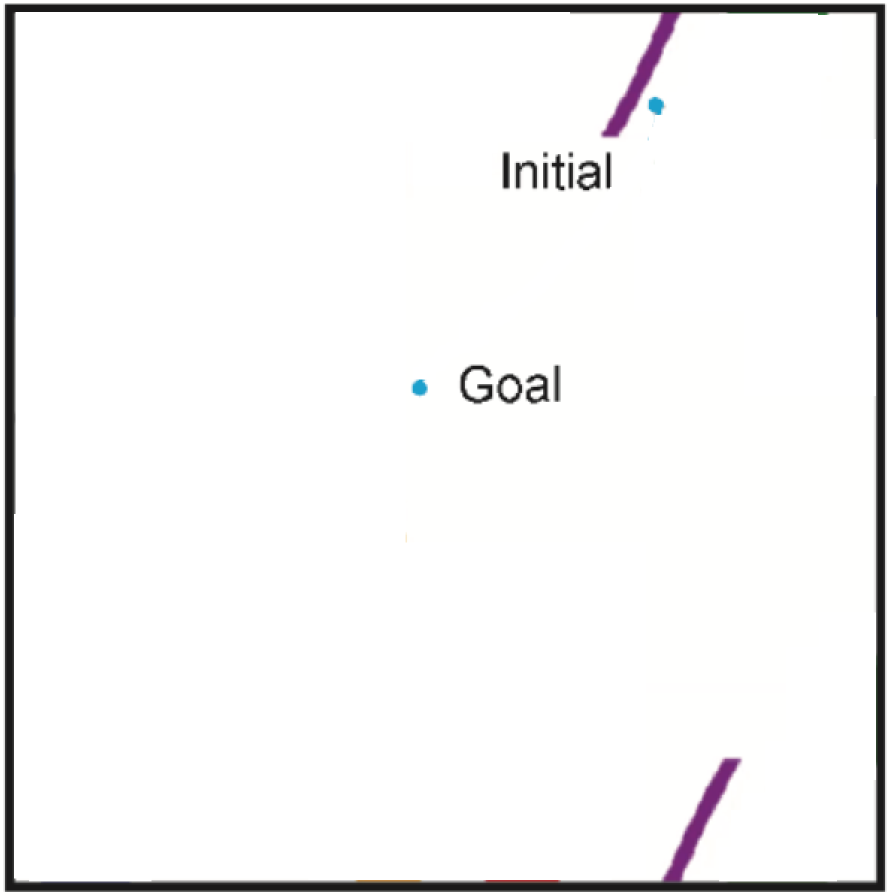


Okay, but why?

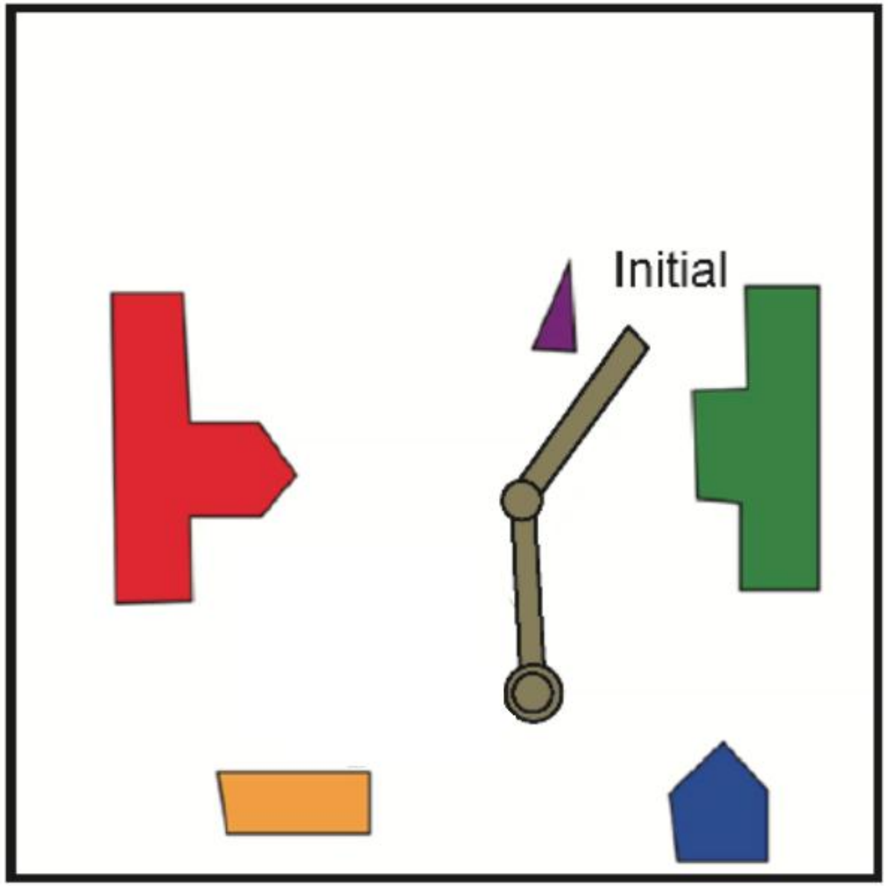


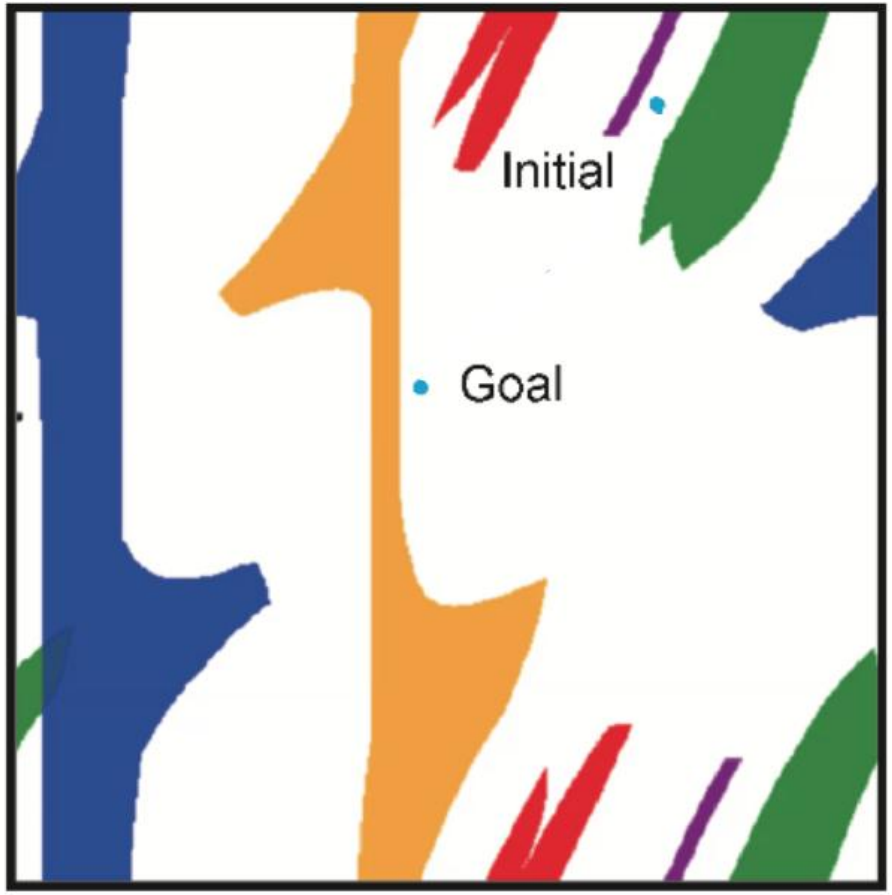
Okay, but why?



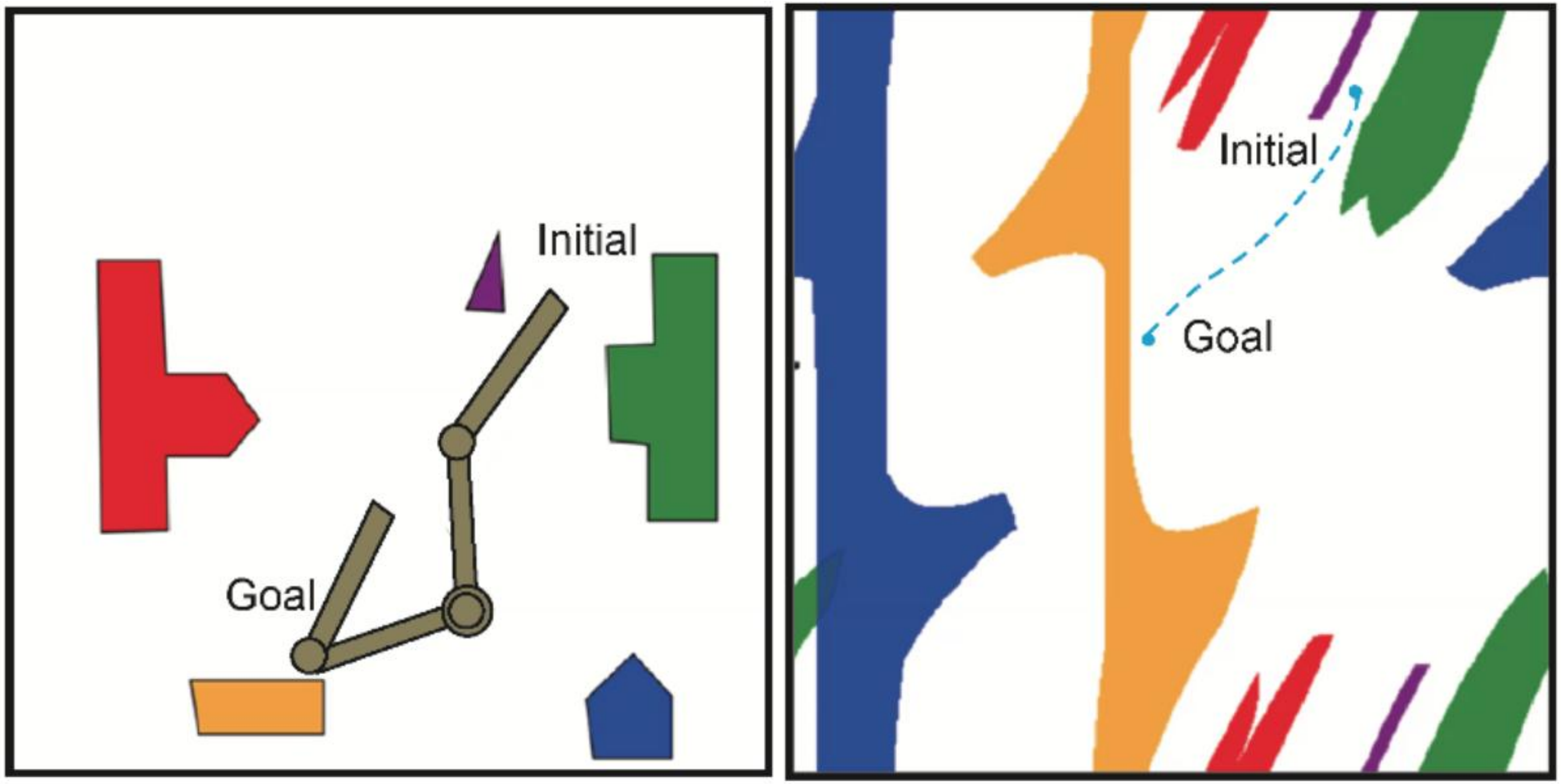


Okay, but why?





Okay, but why?



Today…

- General course information
- Basics of robotics
- Fundamentals of machine learning

Machine learning

Supervised learning

Given $\{(x^i, y^i)\}_{i=1}^n$, find a function

$$f(x) = y$$

data point
predictor
...

value
label
...

(classification, regression)

Unsupervised learning

Machine learning

Supervised learning

Given $\{(x^i, y^i)\}_{i=1}^{n}$, find a function

$$f(x) = y$$

data point
predictor
…

value
label
…

(classification, regression)

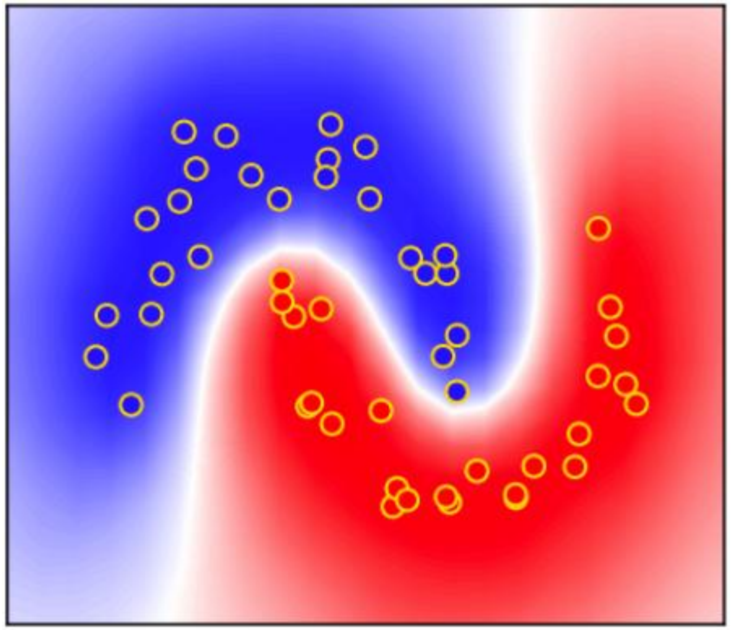
Unsupervised learning

Given $\{x^i\}_{i=1}^{n}$, find patterns

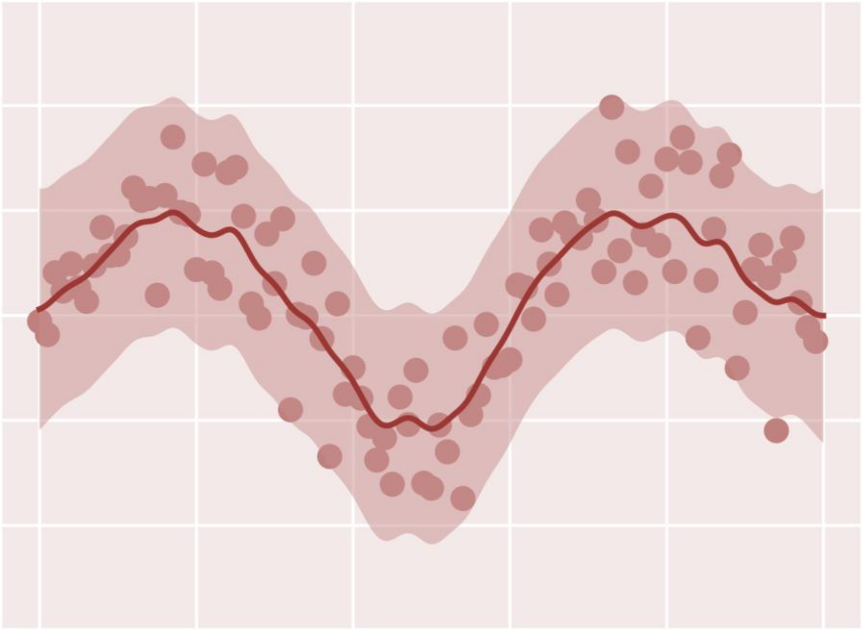
(clustering, compression, dimensionality reduction)

Supervised learning

Classification



Regression



Gaussian Process Classification Model in various PPLs, Arthur Lui
Gaussian Process Regression From First Principles, Ryan Sander

Learning models

- Parametric models:

$$y = f_\theta(x)$$

  Examples: naïve Bayes, logistic regression, neural networks

Learning models

- Parametric models:

$$y = f_\theta(x)$$

Examples: naïve Bayes, logistic regression, neural networks

- Non-parametric models:

dataset

$$y = f(x; D)$$

Examples: K-nearest neighbors, Gaussian process regression

Loss functions

A loss function evaluates the quality of fit in $f(x) \approx y$ or the quality of patterns in an unsupervised learning problem.

Examples:

ℓ^2 loss: $L(\theta) = \sum_{(x^i, y^i) \in D} \left(y^i - f_\theta(x^i)\right)^2$

Cross-entropy loss: $L(\theta) = -\sum_{(x^i, y^i) \in D} \left(y^i\right)^\top \log f_\theta\left(x^i\right)$

Minimizing the loss

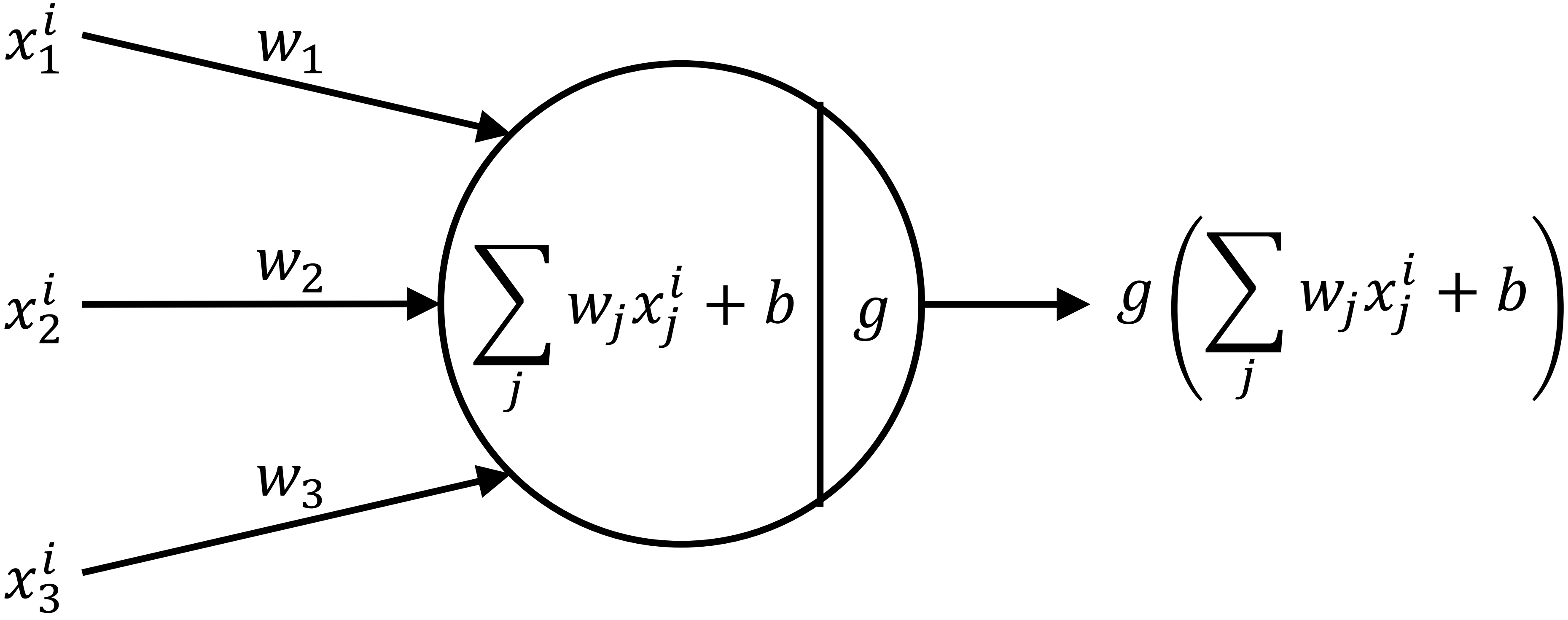
- Analytical solution
  - Use exact methods to find $\theta^* = \arg\min_\theta L(\theta)$
  - Occasionally possible, e.g., linear regression

- Numerical optimization
  - Numerically minimize $L(\theta)$, e.g., gradient descent by computing $\nabla L(\theta)$
  - Much more common in robot learning research
  - Stochastic optimization is often necessary for efficiency

Neural networks

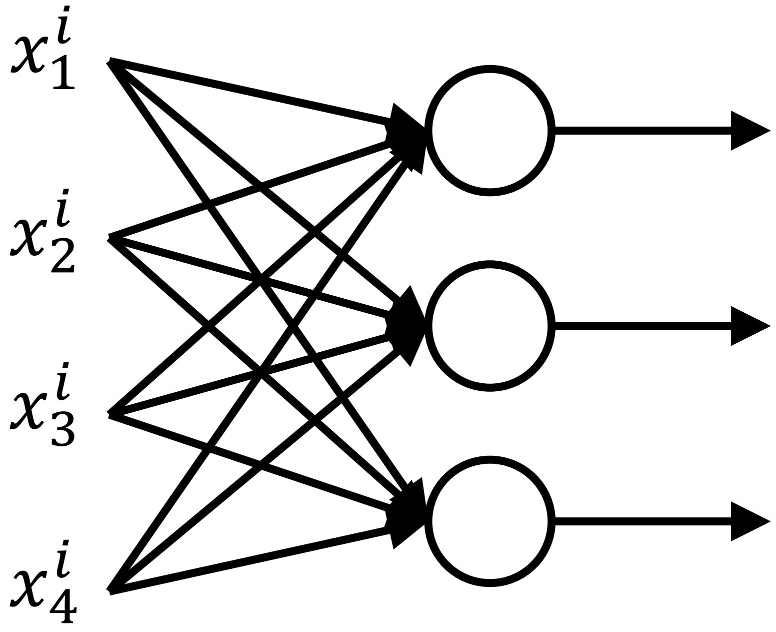
This is not the first model taught in a machine learning class. But we will almost never use other models.

1. A perceptron



Neural networks

2. A single layer neural network



$$y_1^i = g\left(w_1^\top x^i + b_1\right)$$

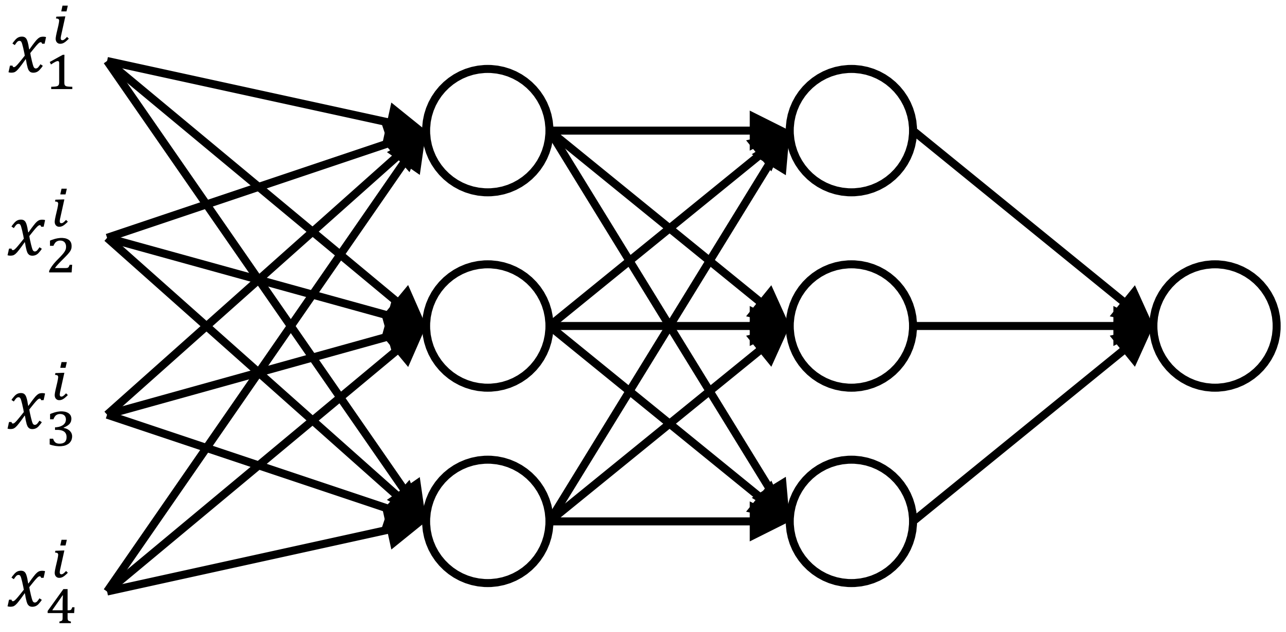
$$y_2^i = g\left(w_2^\top x^i + b_2\right)$$

$$y_3^i = g\left(w_3^\top x^i + b_3\right)$$

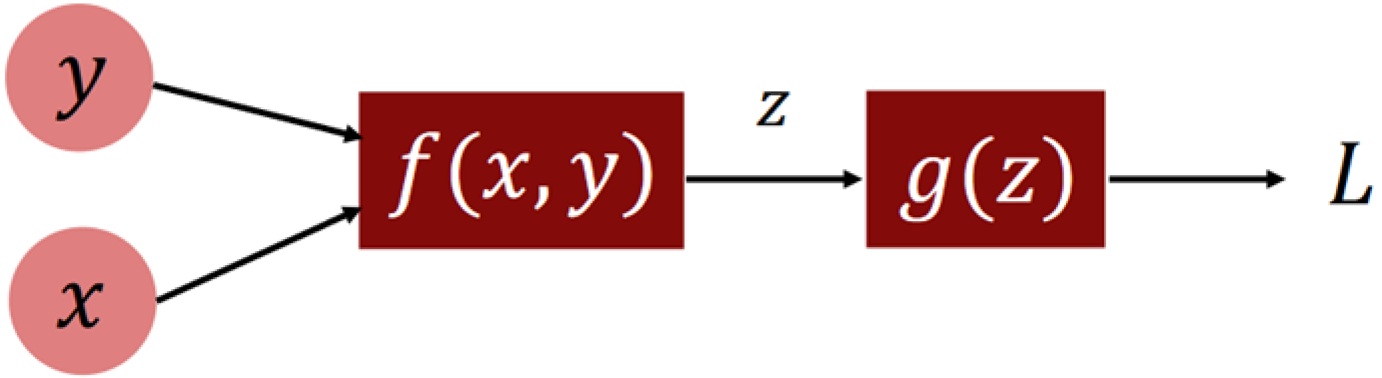
$$\Rightarrow \quad y = g(W^\top x + b)$$

Neural networks

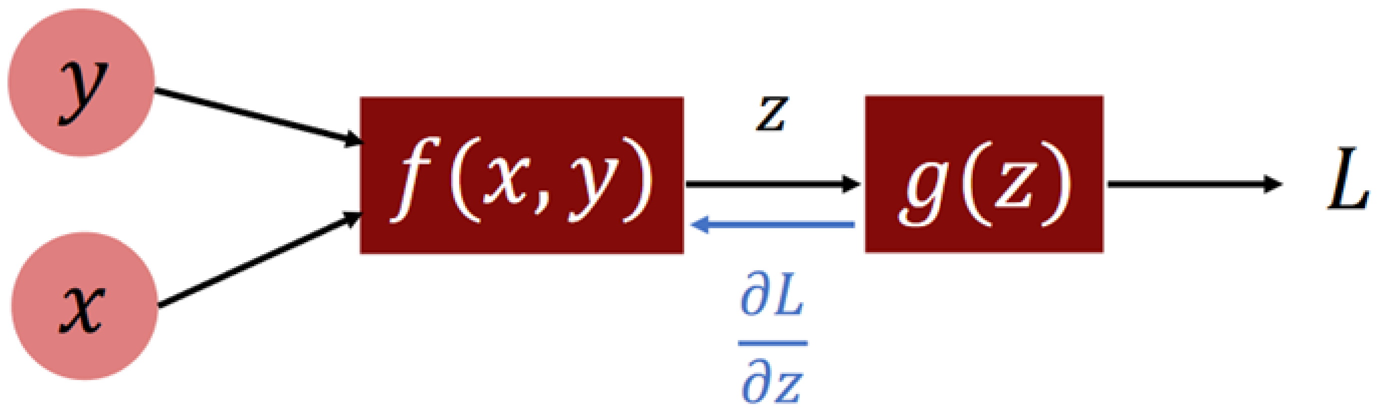
3. A deep neural network



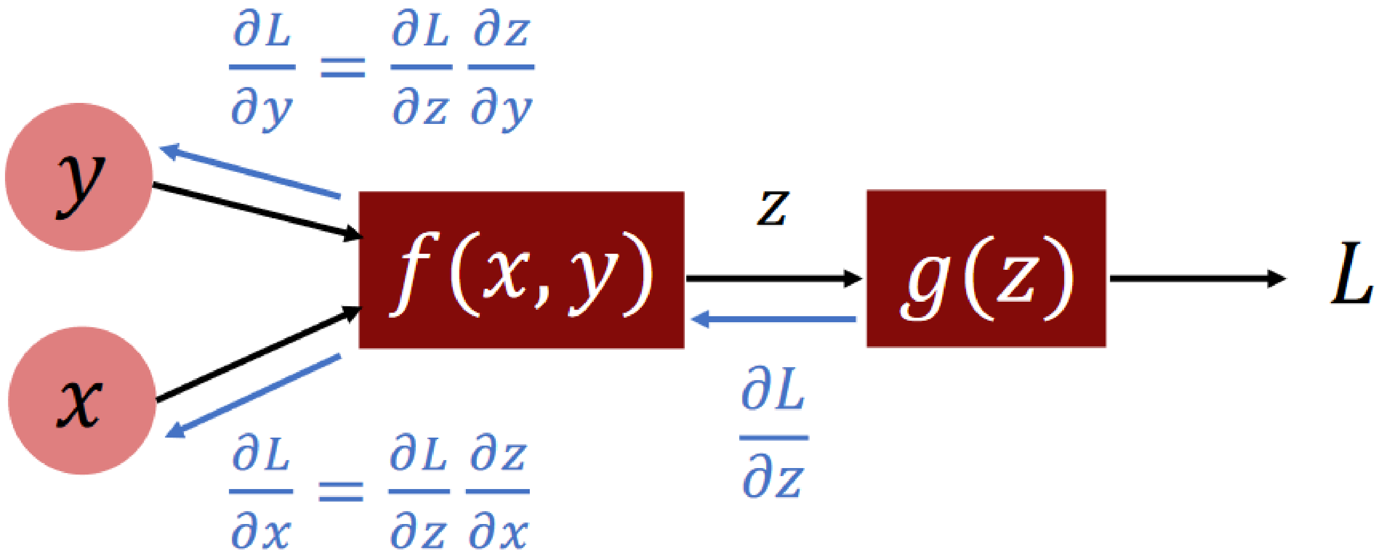
Backpropagation



Backpropagation

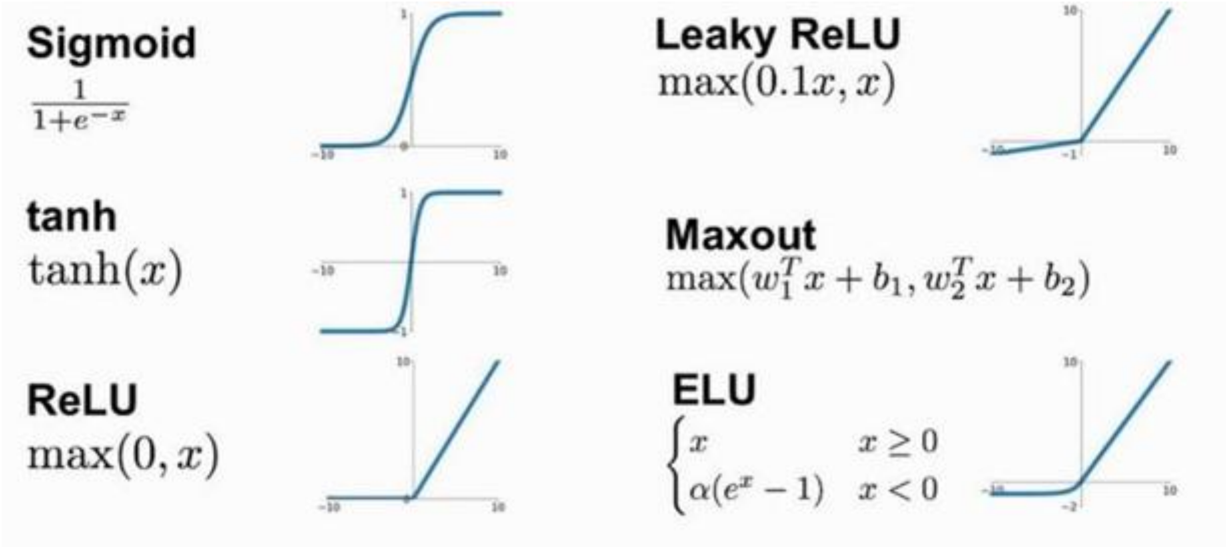


Backpropagation



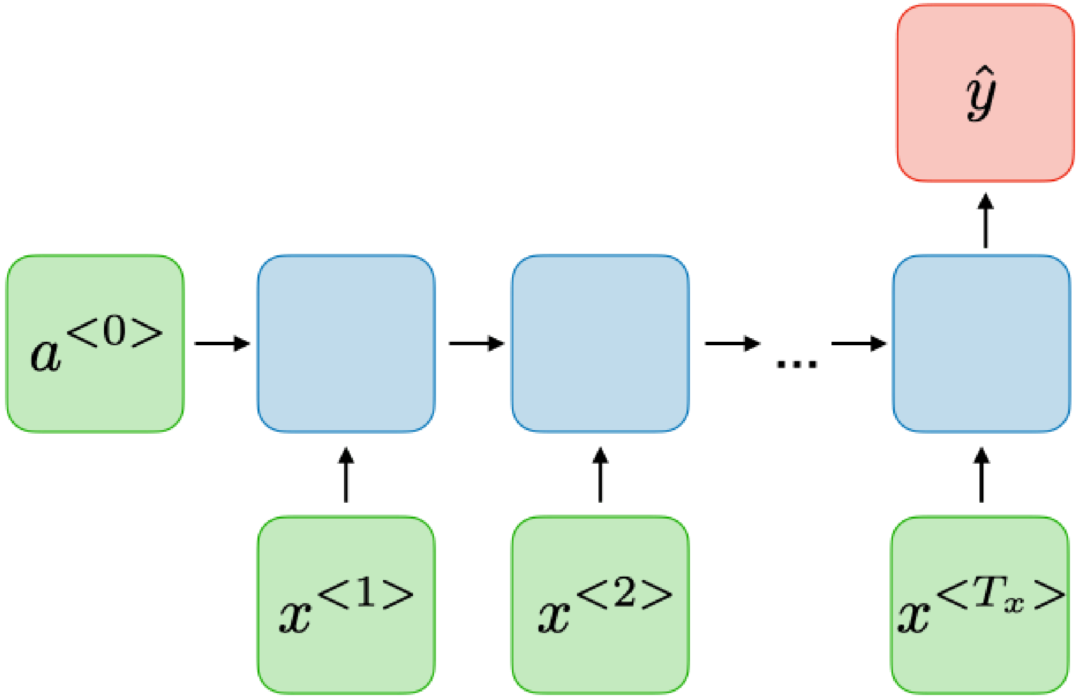
Activation functions

g should not be a linear function.



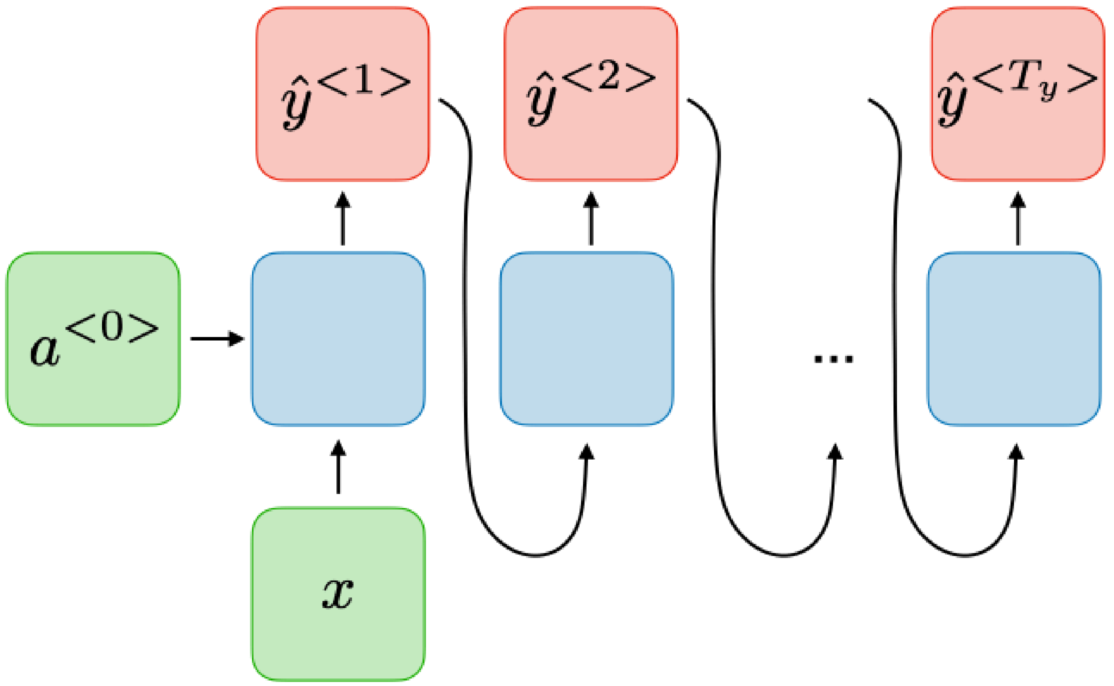
Recurrent neural networks (RNN)

Many-to-one



Recurrent neural networks (RNN)

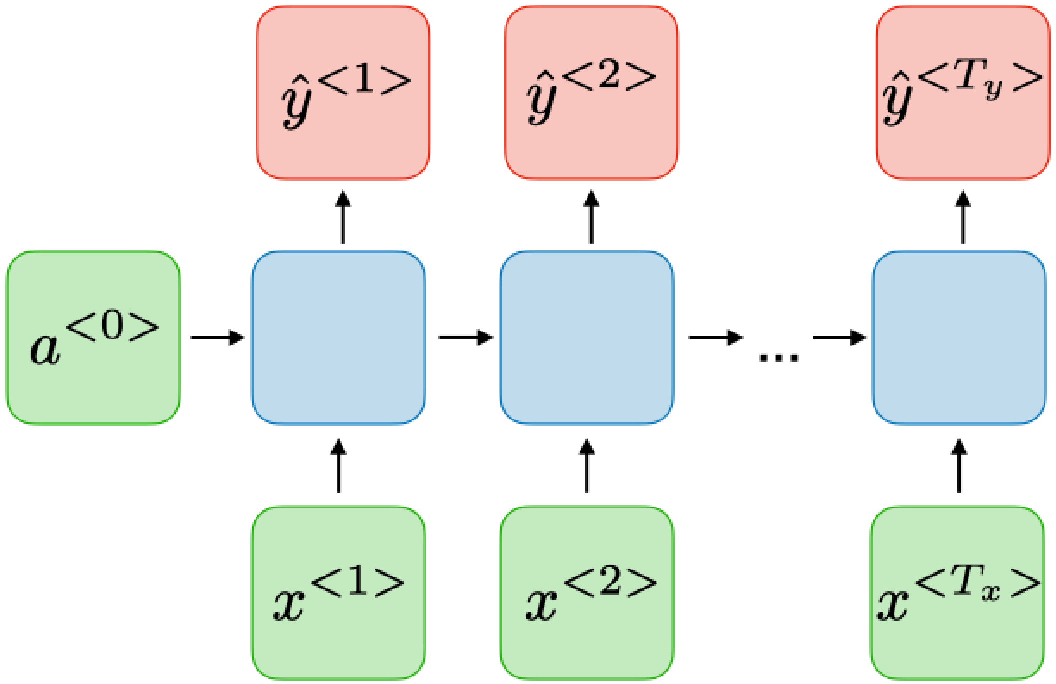
One-to-many



From: Stanford CS 230

Recurrent neural networks (RNN)

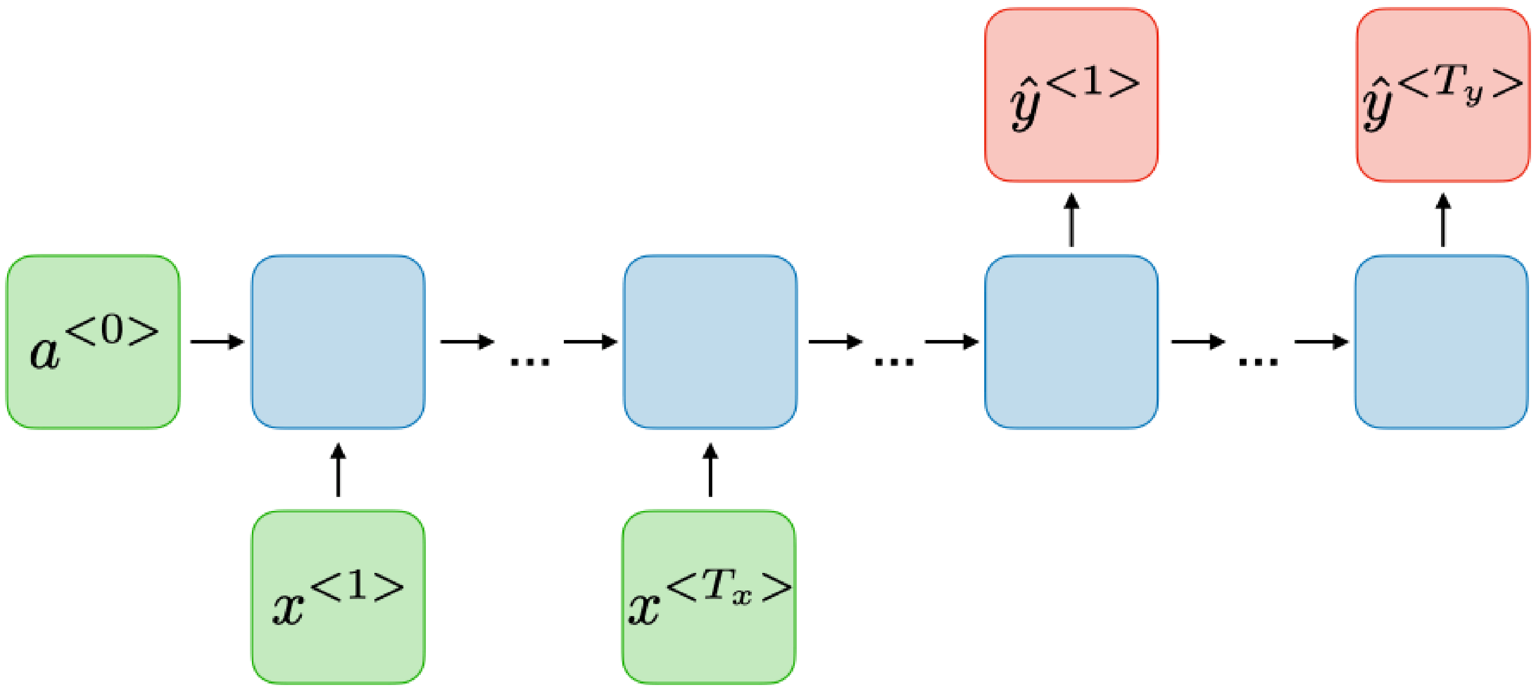
Many-to-many



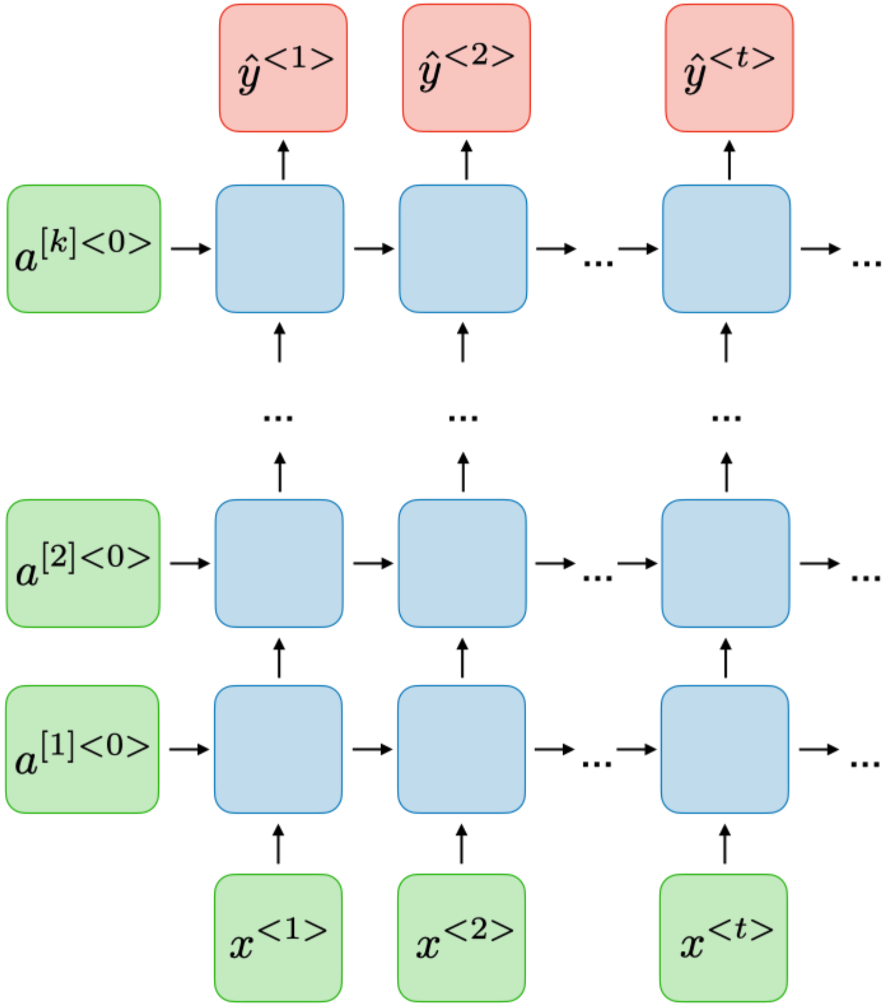
From: Stanford CS 230

Recurrent neural networks (RNN)

Many-to-many

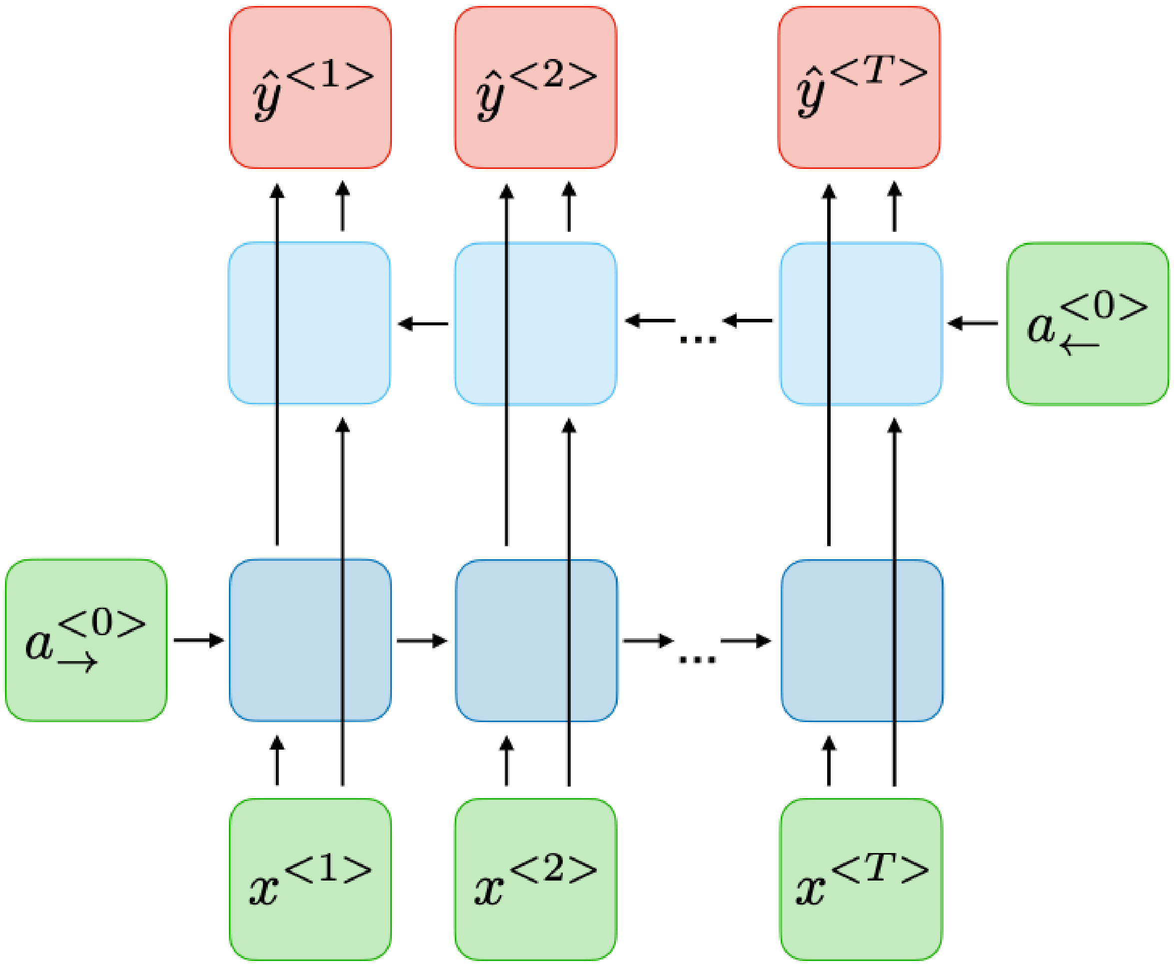


Deep RNNs



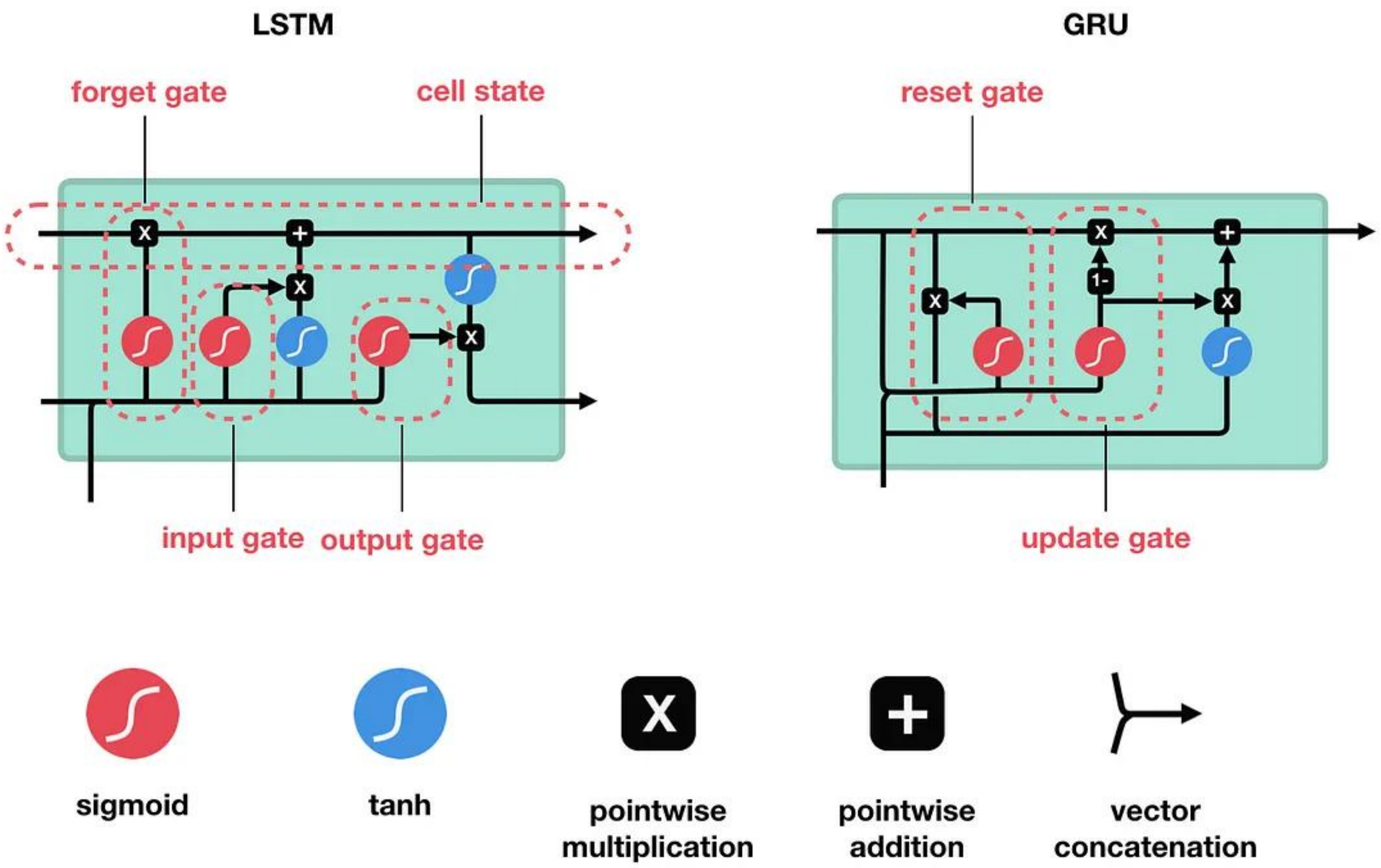
From: Stanford CS 230

Bidirectional RNNs



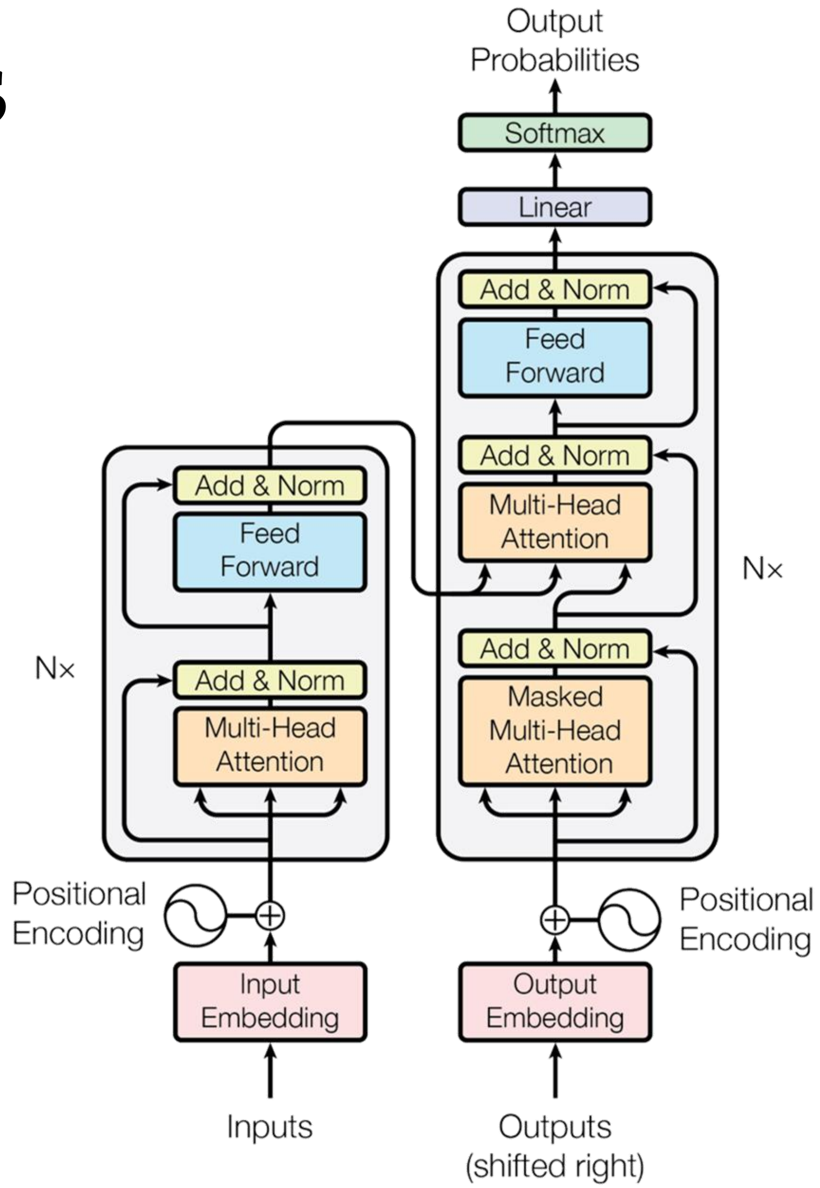
From: Stanford CS 230

LSTMs and GRUs



From: Michael Phi

Transformers



Attention is all you need
Vaswani et al., NeurIPS 2017

Today…

- General course information
- Basics of robotics
- Fundamentals of machine learning

Until next week…

Homework assignments will include programming with a machine learning library: PyTorch.

There are many online PyTorch tutorials. For what we covered today, check out:

- https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html
- https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html

Next time…

- Basics of computer vision for robotics

- Representation learning