Neural Networks and Markov Decision Processes

CMPT 882

Mar. 13

Outline

- Neural networks
  - Forward and backward propagation
  - Typical structures
- Markov Decision Processes
  - Definitions
  - Example
  - Objective in reinforcement learning

Neural Networks

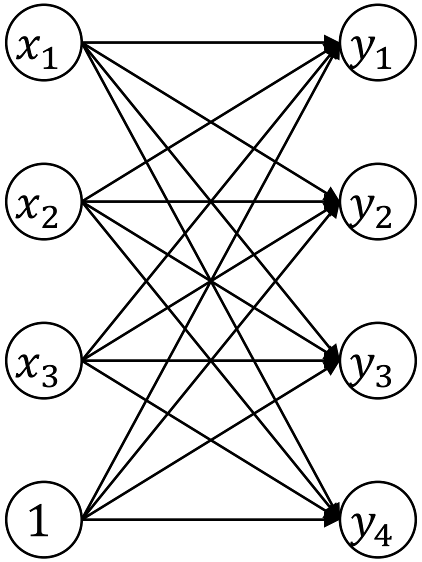
- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$

- Forward propagation
  - Evaluation of $f_\theta(x)$

- Backpropagation
  - Computation of $\frac{\partial l}{\partial \theta}$, where l is the loss function

Neural Networks

- A specific form of $f_\theta(x)$



$$y_1 = f(x^\top w_1 + b_1)$$

$$y_2 = f(x^\top w_2 + b_2)$$

$$y_3 = f(x^\top w_3 + b_3)$$

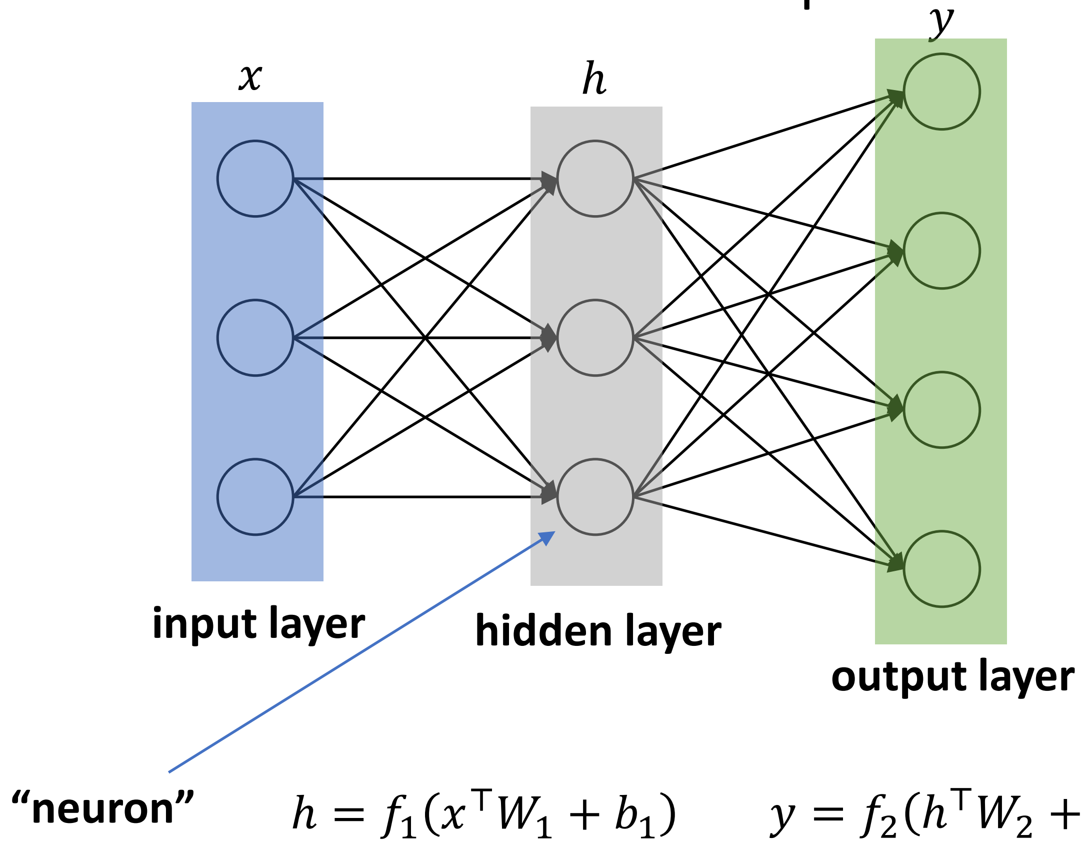
$$y_4 = f(x^\top w_4 + b_4)$$

$$y = f(x^\top W + b)$$

- Parameters θ are W and b
- "**Weights**"

Neural Networks

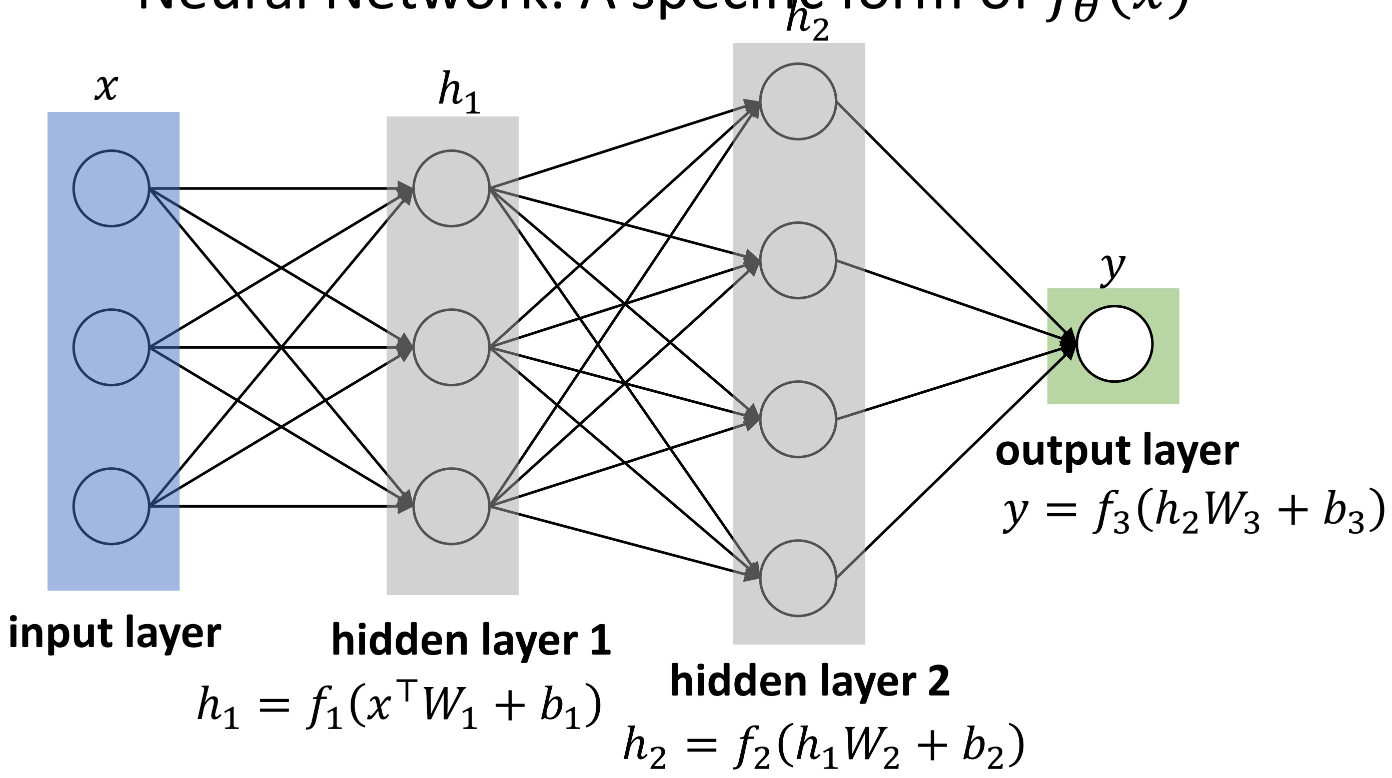
- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$



$h = f_1(x^\top W_1 + b_1)$ $\quad$ $y = f_2(h^\top W_2 + b_2)$

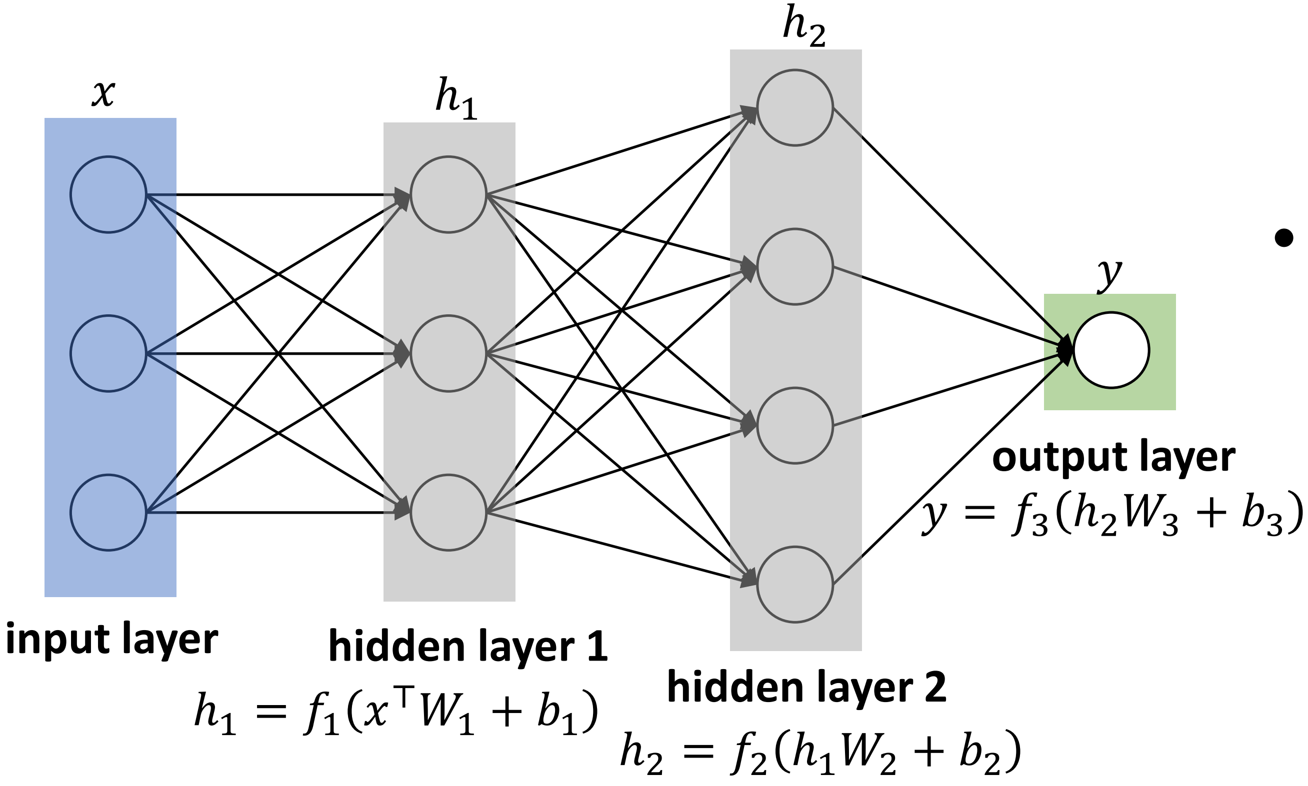
Neural Networks

- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$



Neural Networks

- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$



- Parameters θ are the weights W_i and b_i

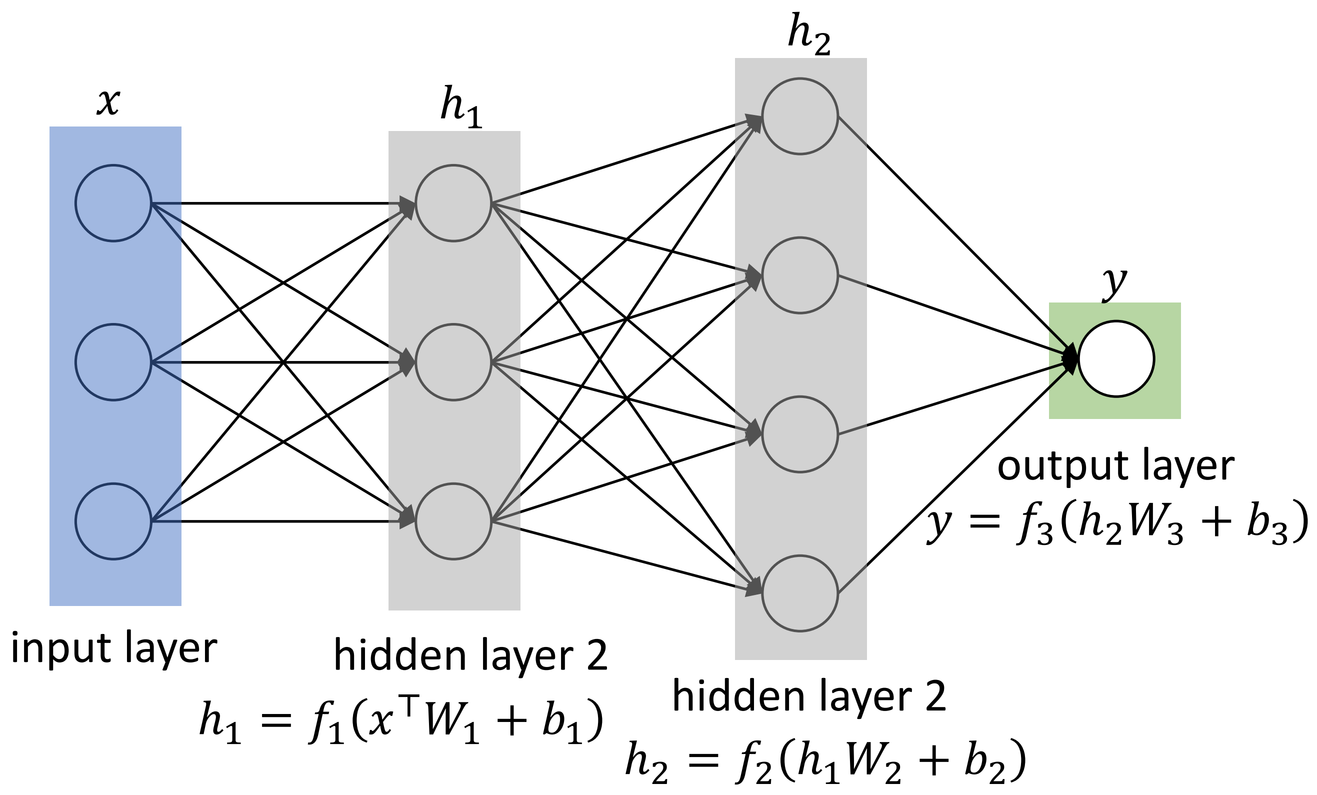
- f_1, f_2, f_3 are nonlinear
  - Otherwise f would just be a single linear function:

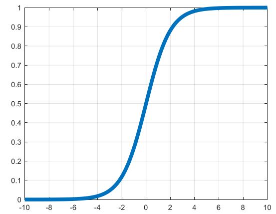
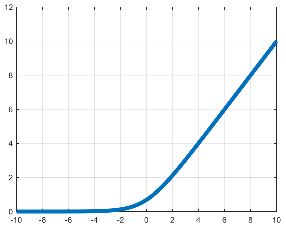
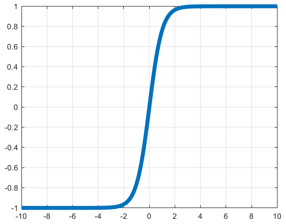
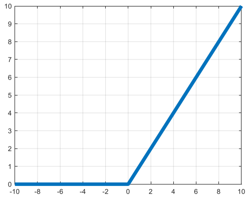
$$\begin{aligned} y &= \big((x^\top W_1 + b_1)W_2 + b_2\big)W_3 + b_3 \\ &= x^\top W_1 W_2 W_3 + b_1 W_2 W_3 + b_2 W_3 + b_3 \end{aligned}$$

  - “Activation functions”

Neural Networks

- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$

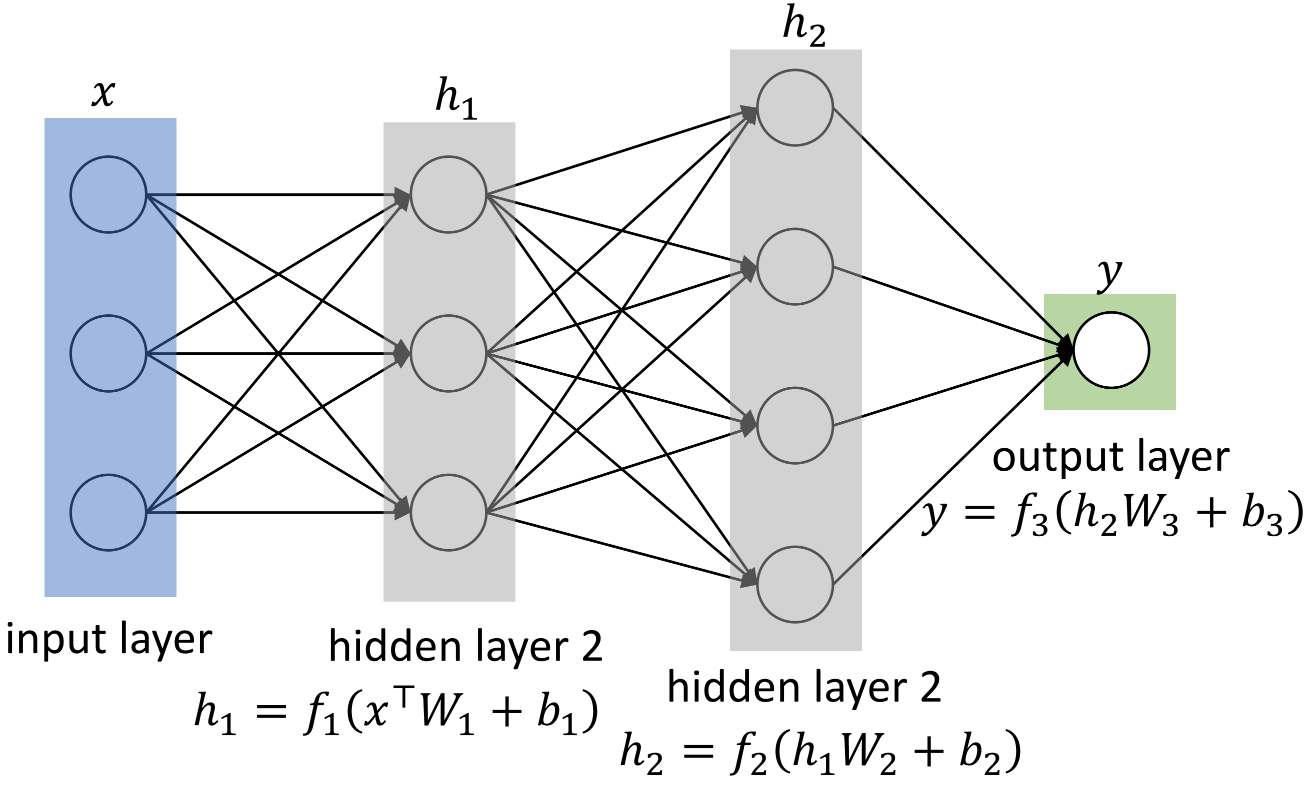


- Common choices of activation functions
  - Sigmoid:
    $$\frac{1}{1 + e^{-x}}$$
    
  - Softplus:
    $$\log(1 + e^{x})$$
    
  - Hyperbolic tangent:
    $$\tanh x$$
    
  - Rectified linear unit (ReLU):
    $$\max(0, x)$$
    

- Key feature: easy to differentiate

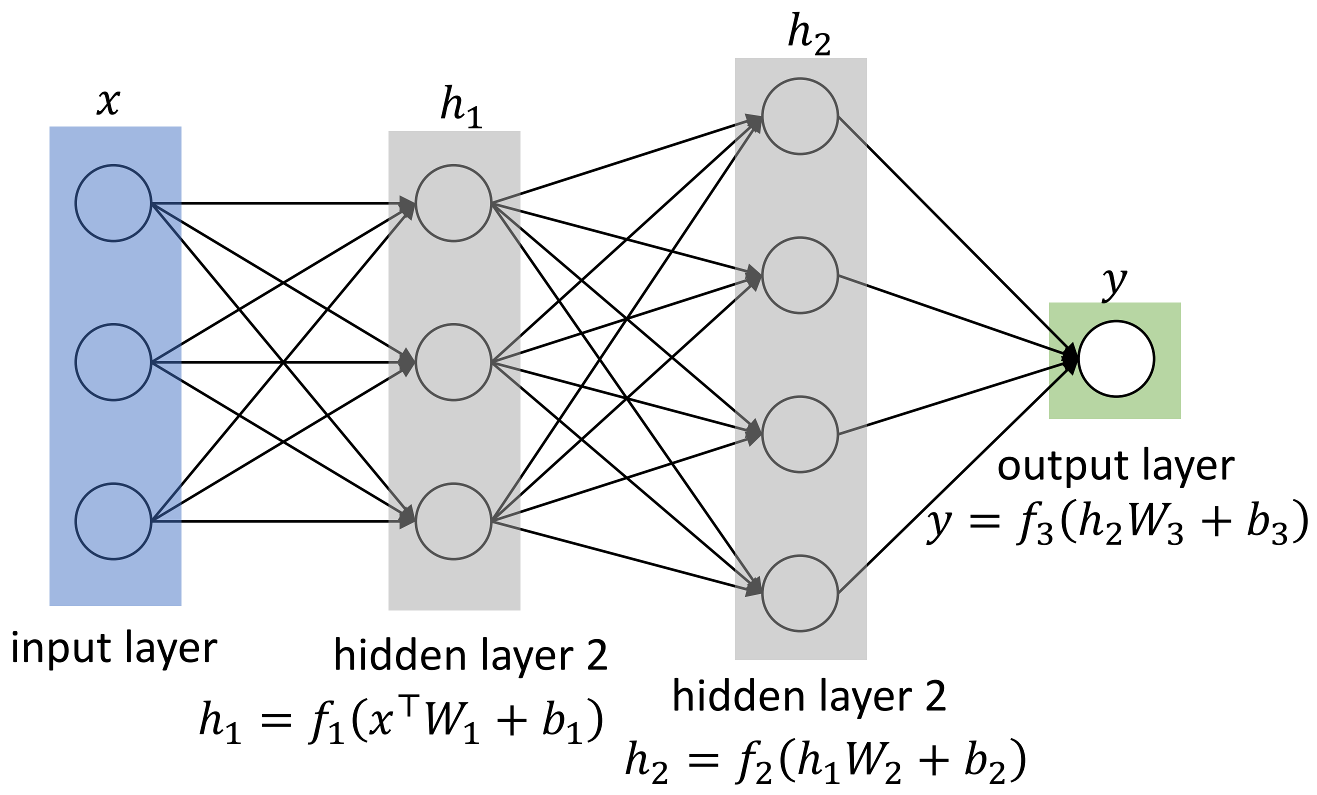
Training Neural Networks and Backpropagation

- Regression: Choose θ such that $y \approx f_\theta(x)$
  - Neural Network: A specific form of $f_\theta(x)$



- Given current θ, X, Y, compute $f_\theta(X)$ to then obtain loss, $l(\theta; X, Y)$
  - $l(\theta; X, Y)$ compares $f_\theta(X)$ with ground truth Y
  - Evaluation of f: “**Forward propagation**”
- Minimize $l(\theta; X, Y)$
  - Stochastic gradient descent
  - Evaluation of $\frac{\partial l}{\partial W}$: “**Backpropagation**”
    - Example: $\frac{\partial y}{\partial W_1} = \frac{\partial y}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W_1}$
    - Just the chain rule

Backpropagation



$y = f_3(h_2 W_3 + b_3)$,

- where $h_2 = f_2(h_1 W_2 + b_2) = h_2(W_2)$
- So $y = f_3(h_2(W_2) W_3 + b_3) = f_3(W_2, W_3)$

But $h_2 = f_2(h_1 W_2 + b_2)$,

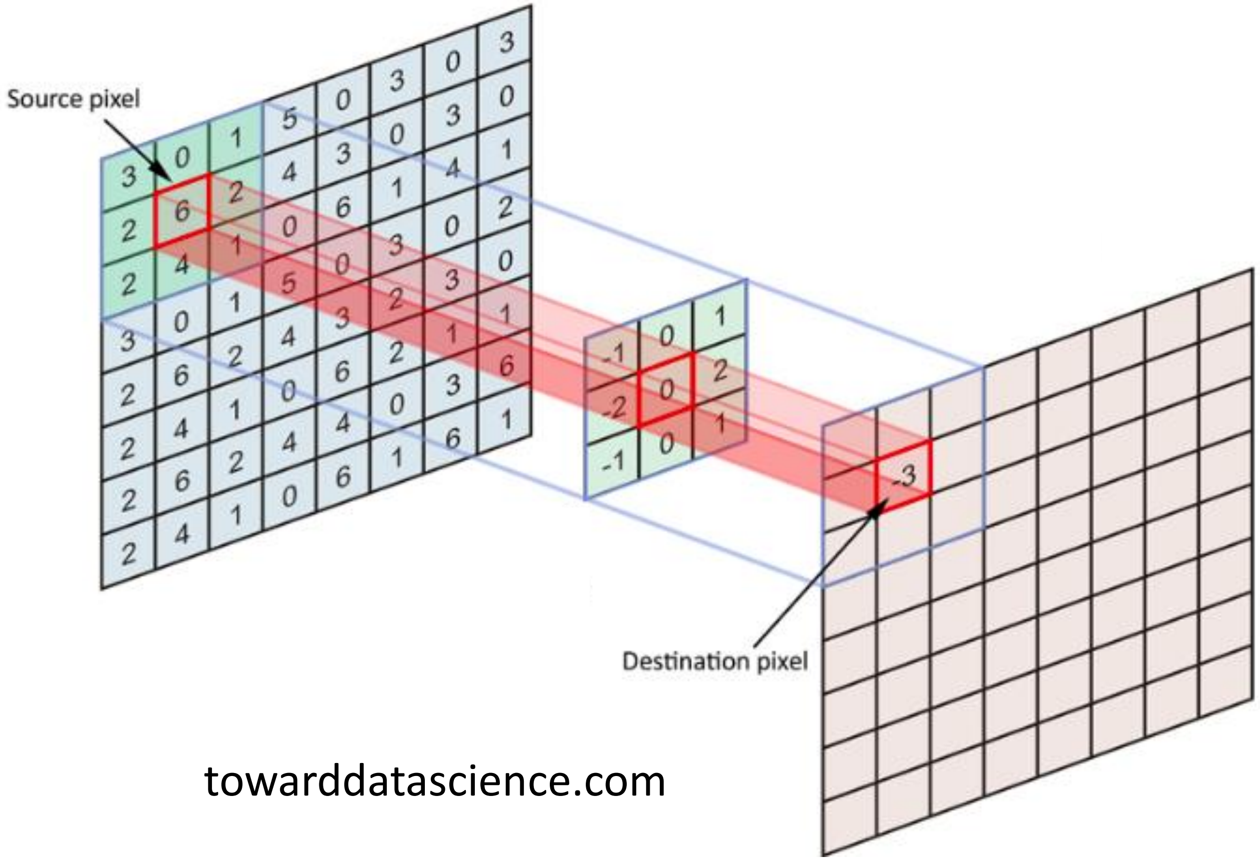
- where $h_1 = f_1(x^\top W_1 + b_1) = h_1(W_1)$
- So $h_2 = f_2(h_1(W_1) W_2 + b_2) = f_2(W_1, W_2)$
- So $y = f_3(h_2(W_1, W_2) W_3 + b_3) = f_3(W_1, W_2, W_3)$

Example: gradient with respect to W_1

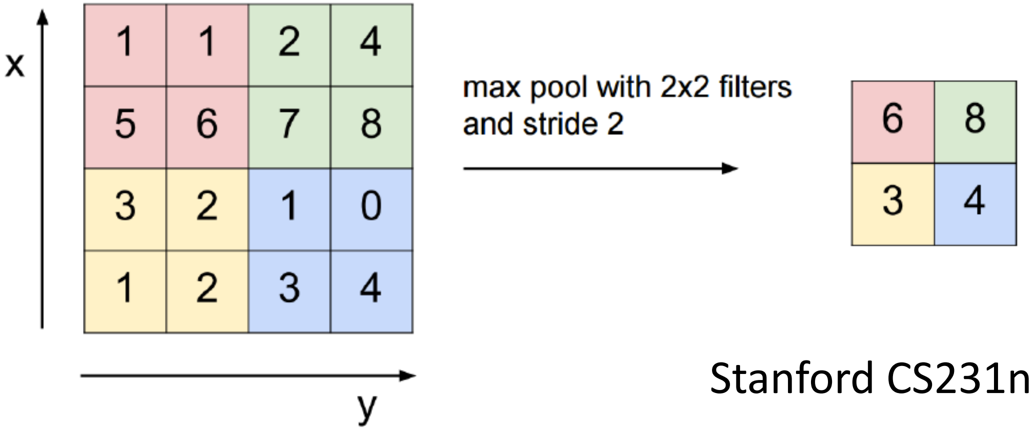
- $\frac{\partial y}{\partial W_1} = \frac{\partial f_3}{\partial h_2} \frac{\partial h_2}{\partial W_1} = \frac{\partial f_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W_1}$
- Each term is a tensor, which results from taking the gradient of a vector w.r.t. a matrix
- Software like TensorFlow performs this (and other operations common in machine learning) efficiently

Common Operations

- Fully connected (dot product)
- Convolution
  - Translationally invariant
  - Controls overfitting
- Pooling (fixed function)
  - Down-sampling
  - Controls overfitting
- Nonlinearity layer (fixed function)
  - Activation functions, e.g. ReLU

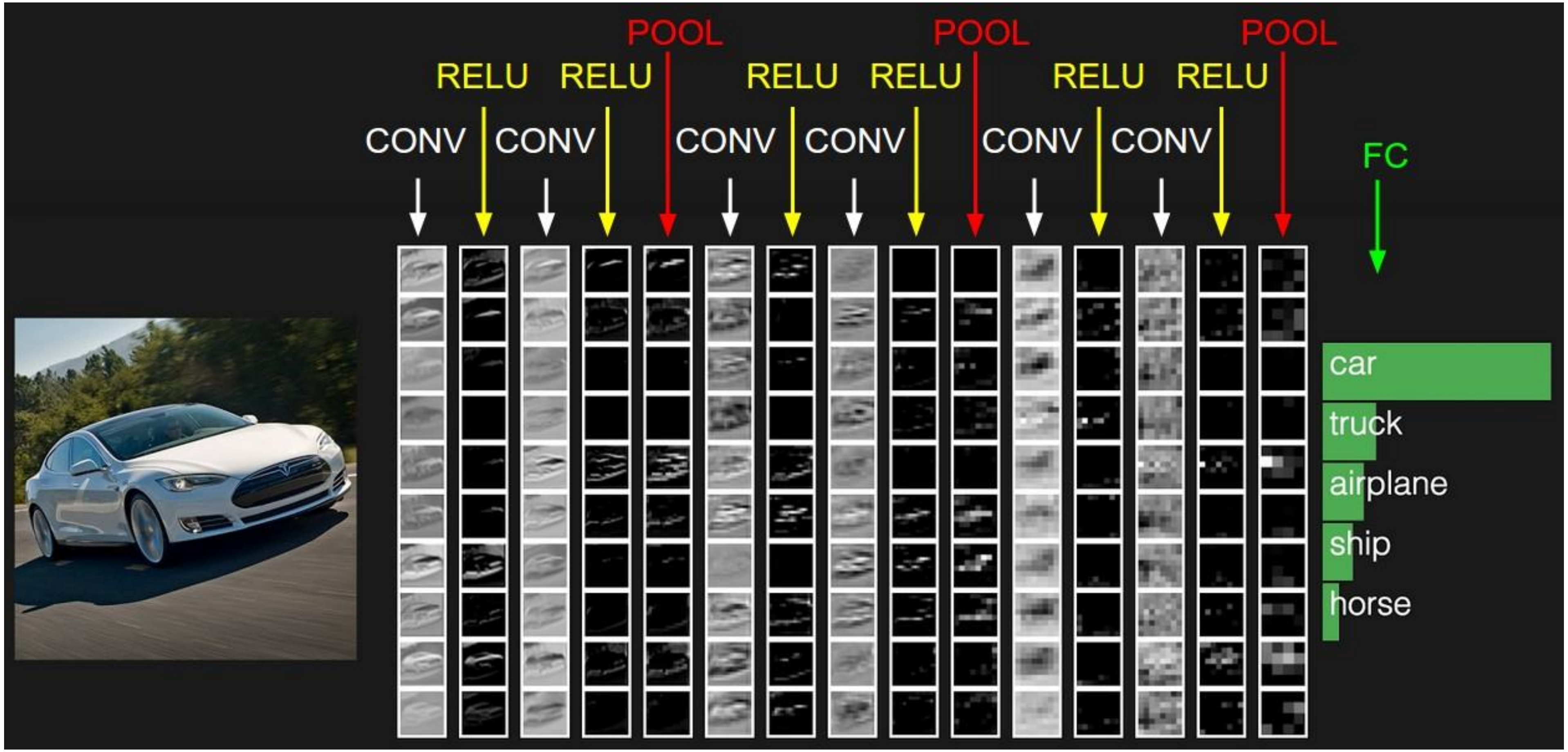


towarddatascience.com

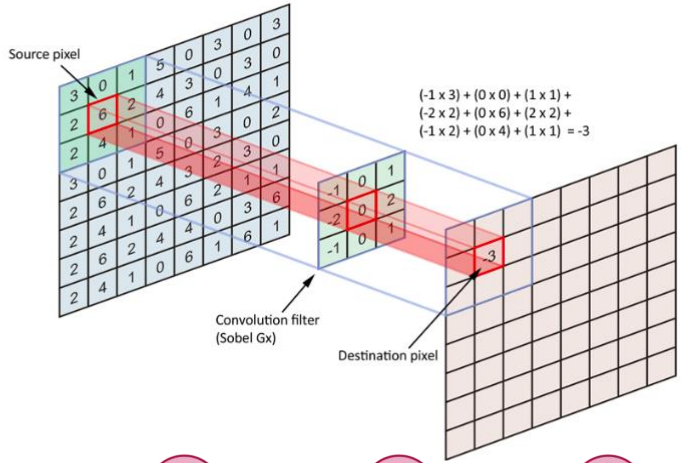


Stanford CS231n

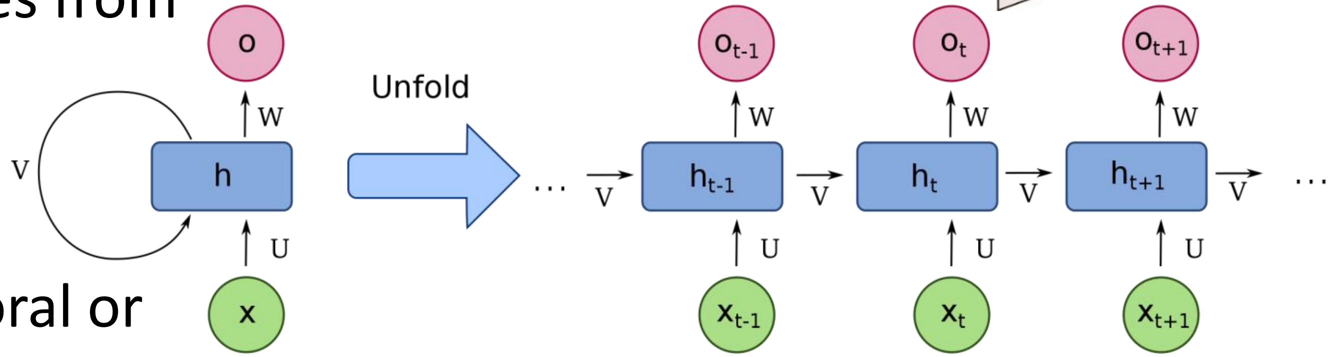
Example: Small VGG Net From Stanford CS231n



Neural Network Architectures



- Convolutional neural network (CNN)
  - Has translational invariance properties from convolution
  - Common used for computer vision



- Recurrent neural network RNN
  - Has feedback loops to capture temporal or sequential information
  - Useful for handwriting recognition, speech recognition, reinforcement learning
  - Long short-term memory (LSTM): special type of RNN with advantages in numerical properties
- Others
  - General feedforward networks, variational autoencoders (VAEs), conditional VAEs,

Training Neural Networks

- Data preprocessing
  - Removing bad data
  - Transform input data (e.g. rotating, stretching, adding noise)

- Training process (optimization algorithm)
  - L1 and L2 regularization
  - Dropout: randomly set neurons to zero in each training iteration
  - **Learning rate** (step size) and other hyperparameter tuning

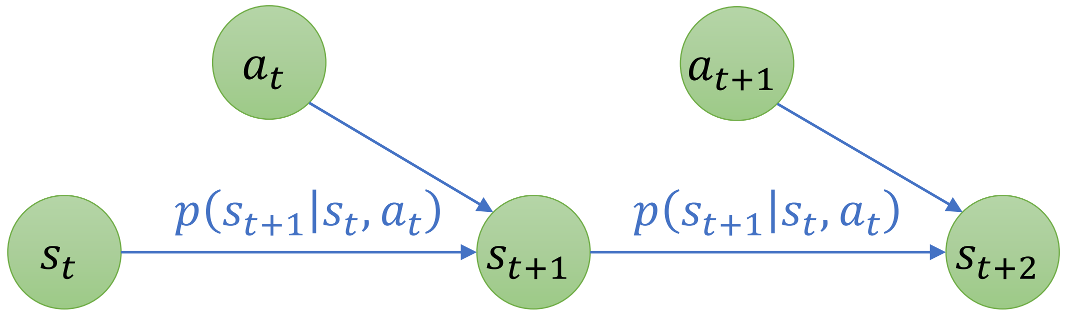
- Software packages: efficient gradient computation
  - Caffe, Torch, Theano, TensorFlow

Outline

- Neural networks
  - Forward and backward propagation
  - Typical structures
- Markov Decision Processes
  - Definitions
  - Example
  - Objective in reinforcement learning

Markov Decision Process

- Probabilistic model of robots and other systems
- State: $s \in \mathcal{S}$, discrete or continuous
- Action (control): $a \in \mathcal{A}$, discrete or continuous
- Transition operator (dynamics): $\mathcal{T}$
  - $\mathcal{T}_{ijk} = p(s_{t+1} = i | s_t = j, a_t = k)$ ← a tensor (multidimensional array)



State in MDPs and Reinforcement Learning

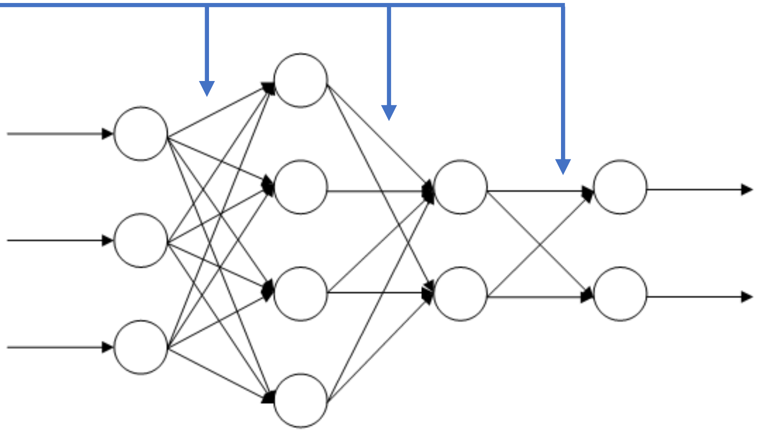
- In optimal control, state usually represents internal states of a robot
- In RL, state usually also include the internal states of a robot, but often also include
  - State of other robots
  - State of the environment
  - Sensor measurements
- Distinction between state and observation can be blurred
- In general, the state contains all variables other than actions that determine the next state through the transition probability $p(s_{t+1}|s_t, a_t)$

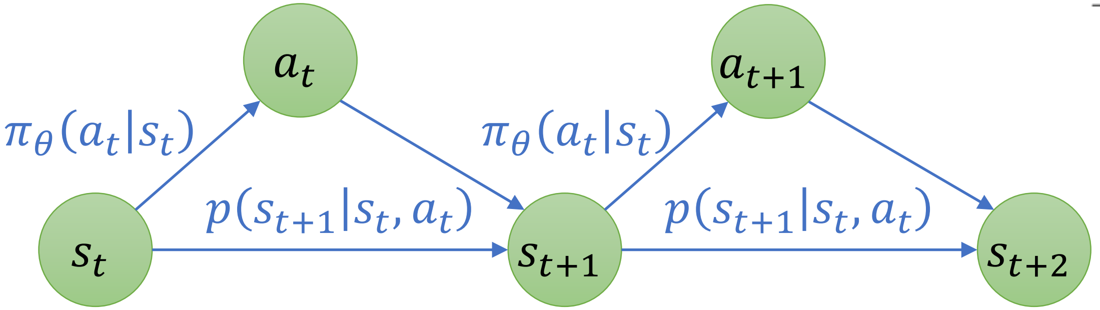
Policy and Reward

- Control policy (feedback control): $\pi(a|s)$
  - Parametrized by some parameters

$$\theta : \pi_\theta(a|s) \coloneqq p(a|s)$$

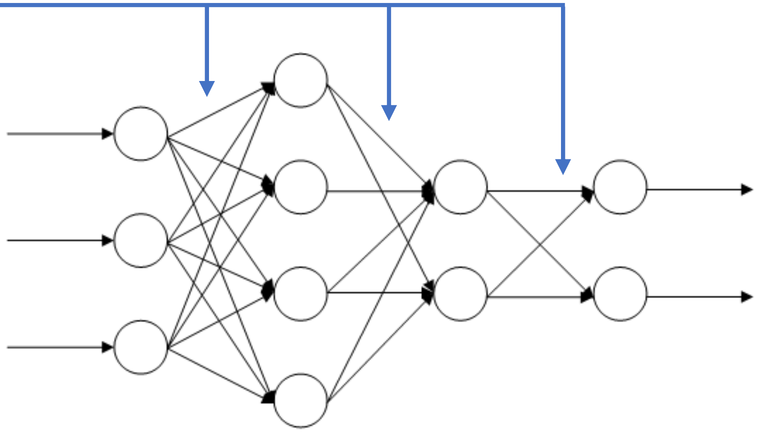
  - Can be stochastic: probability of applying action a at state s





Policy and Reward

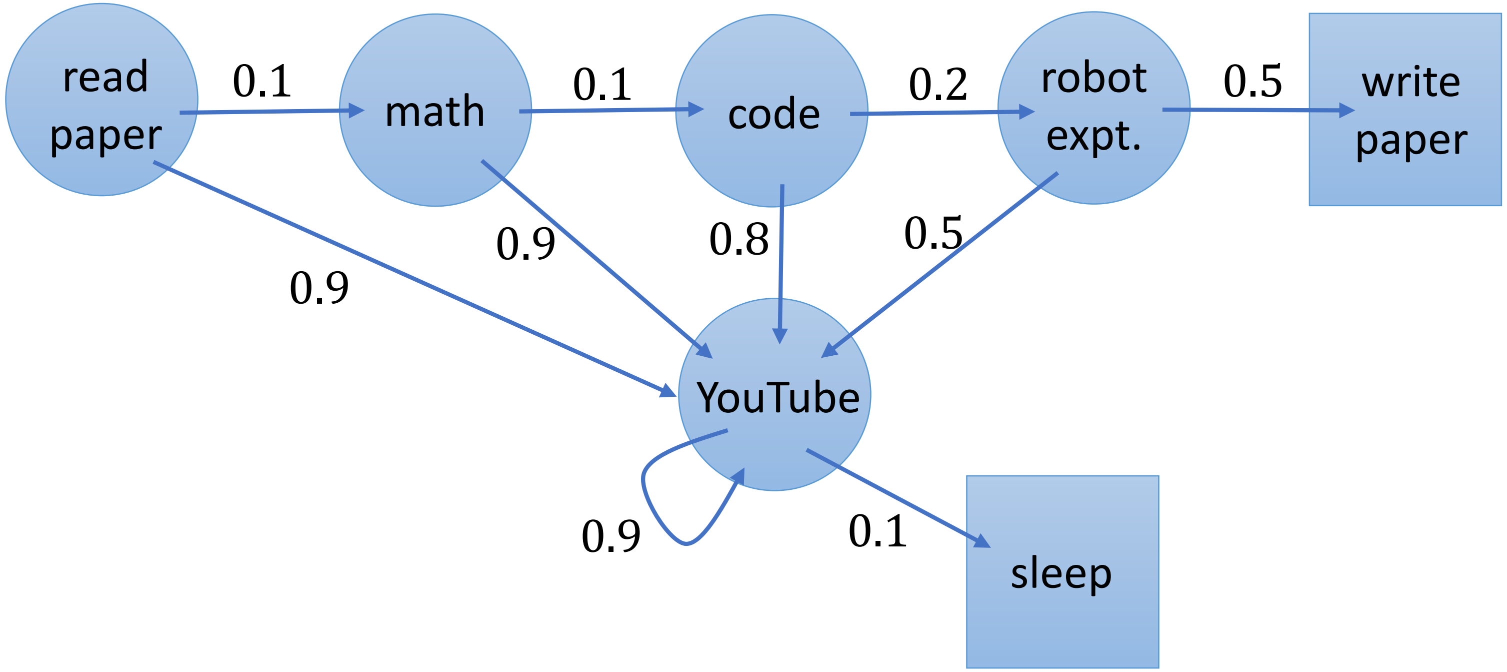
- Control policy (feedback control): $\pi(a|s)$
  - Parametrized by some parameters
$$\theta: \pi_\theta(a|s) \coloneqq p(a|s)$$
  - Can be stochastic: probability of applying action a at state s



- Reward function: $r(s_t, a_t)$
  - Reward received for being at state s_t and applying action a_t
  - Analogous to the cost in optimal control

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



State space includes

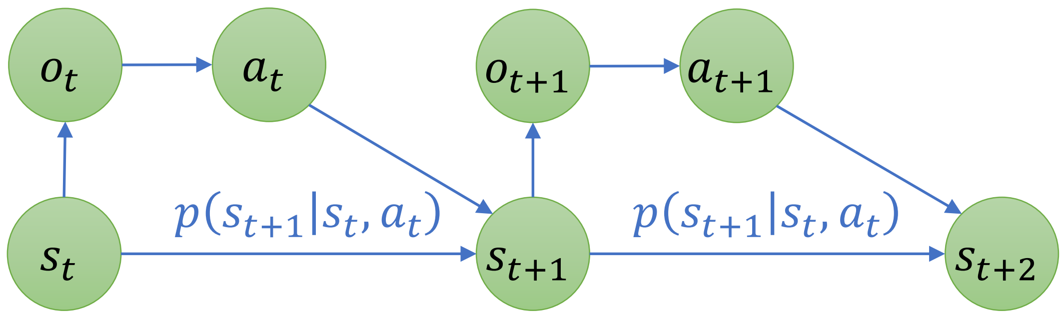
- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.1 & & & 0.9 & & \\ & & 0.1 & & 0.9 & & \\ & & & 0.2 & 0.8 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.9 & & 0.1 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Extensions of Problem Setup

- Partially observability
  - Partially Observable Markov Decision Process (POMDP)
  - State not fully known; instead, act based on observations



  - Policy: $\pi_\theta(a|o)$
- In this class, state s will be synonymous with observation o.

Reinforcement Learning Objective

- Given: an MDP with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathcal{T}$, and reward function $r(s, a)$

- Objective: Maximize discounted sum of rewards ("return")

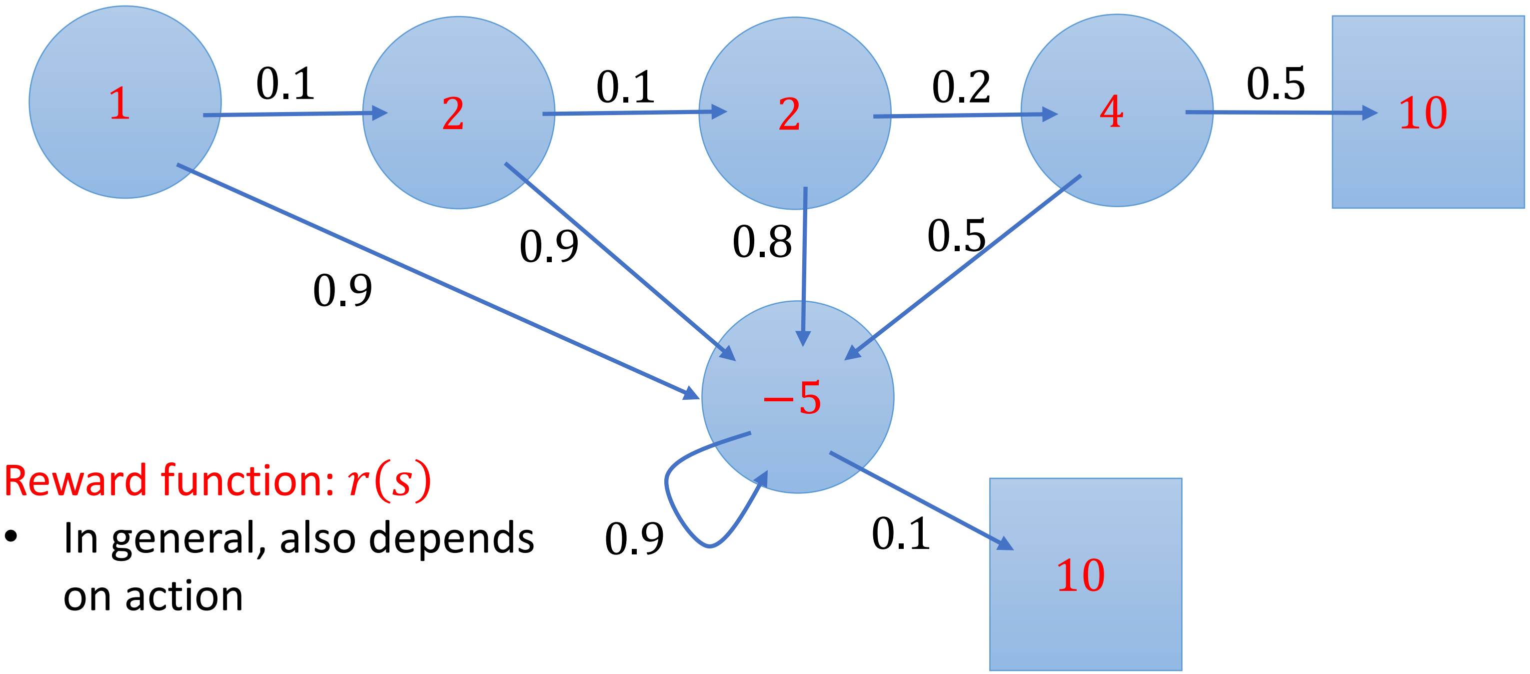
$$\underset{\pi_\theta}{\text{maximize}}\, \mathbb{E} \sum_t \gamma^k r(s_t, a_t)$$

  - $\gamma \in (0,1]$: discount factor – larger roughly means "far-sighted"
    - Prioritizes immediate rewards
    - $\gamma < 1$ avoids infinite rewards; $\gamma = 1$ is possible if all sequences are finite

- Constraints: often implicit
  - Subject to transition matrix $\mathcal{T}$ (system dynamics)

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



Reward function: $r(s)$

- In general, also depends on action

State space includes

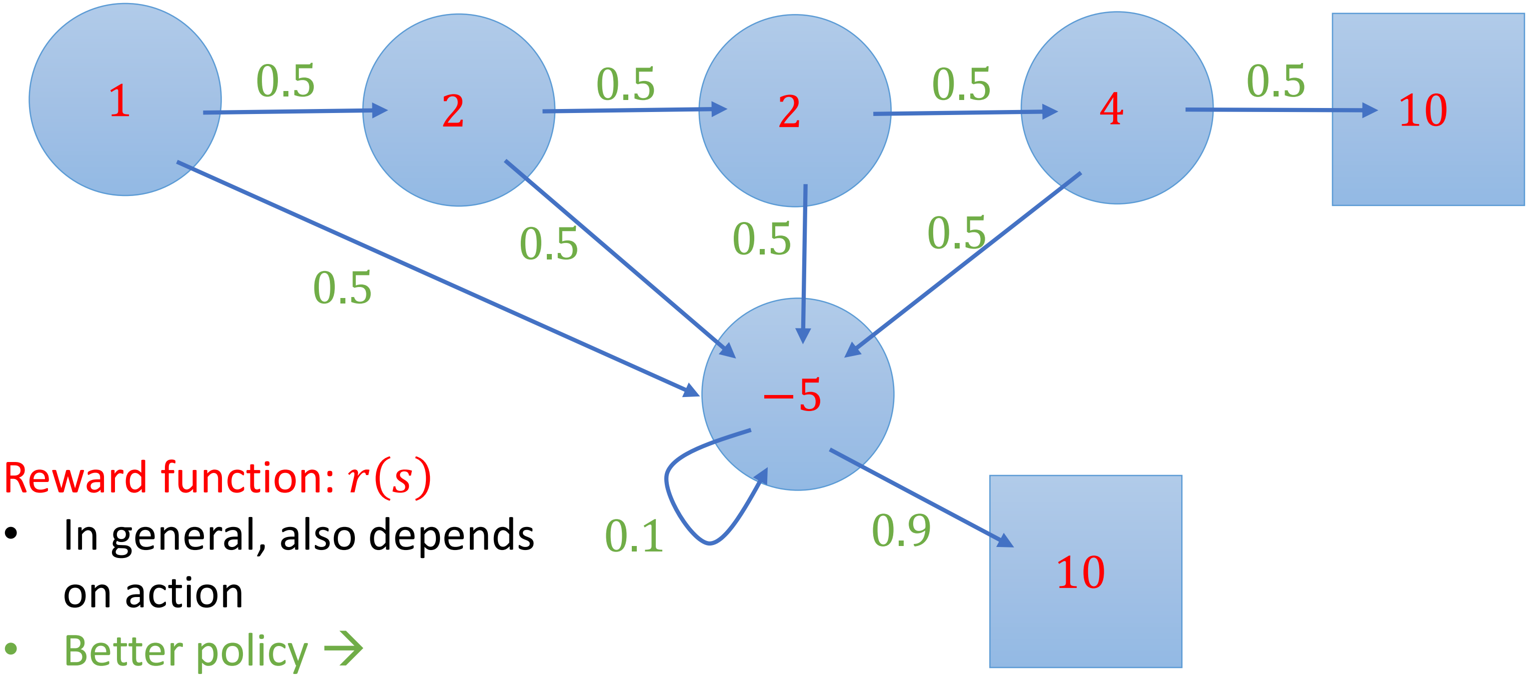
- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.1 & & & 0.9 & & \\ & & 0.1 & & 0.9 & & \\ & & & 0.2 & 0.8 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.9 & & 0.1 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Markov Decision Process

- An MDP with a particular policy results in a Markov chain: $p(s_{t+1}|s_t, a_t), a_t \sim \pi_\theta(a_t|s_t)$



Reward function: $r(s)$

- In general, also depends on action
- Better policy → different Markov chain → different reward

State space includes

- Reading paper
- Doing math
- Coding
- Doing robotic experiments
- Watching YouTube
- Writing paper
- Sleeping

Transition probabilities

$$\mathcal{T} = \begin{bmatrix} & 0.5 & & & 0.5 & & \\ & & 0.5 & & 0.5 & & \\ & & & 0.5 & 0.5 & & \\ & & & & 0.5 & 0.5 & \\ & & & & 0.1 & & 0.9 \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}$$

Reinforcement Learning vs. Optimal Control

- Reinforcement Learning

$$\underset{\pi_\theta}{\text{maximize}}\, \mathbb{E} \sum_t \gamma^k r(s_t, a_t)$$

  - Dynamics constraint is implicit
    - And not necessary needed
  - Typically, no other explicit constraints
  - Problem set up captured entirely in the reward
  - Probabilistic

- Optimal control

$$\begin{aligned} \underset{u(\cdot)}{\text{minimize}} \quad & l\big(x\big(t_f\big), t_f\big) + \int_0^{t_f} c(x(t), u(t), t)dt \\ \text{subject to } & \dot{x}(t) = f\big(x(t), u(t)\big) \\ & g\big(x(t), u(t)\big) \geq 0 \\ & x(t) \in \mathbb{R}^n, u(t) \in \mathbb{R}^m, x(0) = x_0 \end{aligned}$$

  - Explicit constraints
  - Can be continuous time
  - Not necessarily probabilistic