UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 12: Review and Decision Making in Physical Systems

Yuanyuan Shi

Assistant Professor, ECE

University of California, San Diego

- Project Milestone Report is due next Monday, May 9th
- HW2 Written Part posted on Gradescope and Canvas.
- Due in 2 weeks on May 16th

Course Outline

UC San Diego

Part 0: Introduction & Fundamentals

- Lecture 2: Supervised Learning Setup
- Lecture 3: Neural Networks and Backpropagation
- Lecture 4: Optimization & Regularization in Deep Learning

Part 1: Modeling and Learning in Physical Systems

- Lecture 5-6: Convolutional Neural Networks
- Lecture 7: Recurrent Neural Networks
- Lecture 8-9: Graph Neural Networks
- Lecture 10-11: Physics Informed Neural Networks and Neural Operator

Lecture 2: Supervised Learning Setup

UC San Diego

- **Linear Regression**

  Given $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}, x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \mathbb{R}$, we build a prediction model

  $$\hat{y}^{(i)} = f(x^{(i)}) = b + \sum_{j=1}^{n} w_j x_j^{(i)} = w^T x^{(i)} + b$$

  Cost function:

  $$L(w, b) = \frac{1}{2} \sum_{i=1}^{m} \left(w^T x^{(i)} + b - y^{(i)}\right)^2$$

  - Gradient descent to find the best w
  - Why using squared loss? Maximum likelihood interpretation

- **Logistic Regression**

  Given $\{x^{(i)}, y^{(i)}\}_{i=1}^{N}, x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \{0, 1\}$, we build a prediction model

  $$\hat{y}^{(i)} = \sigma\left(w^T x^{(i)} + b\right), \sigma(z) = \frac{1}{1 + e^{-z}}, \text{sigmoid function}$$

  Cost function:

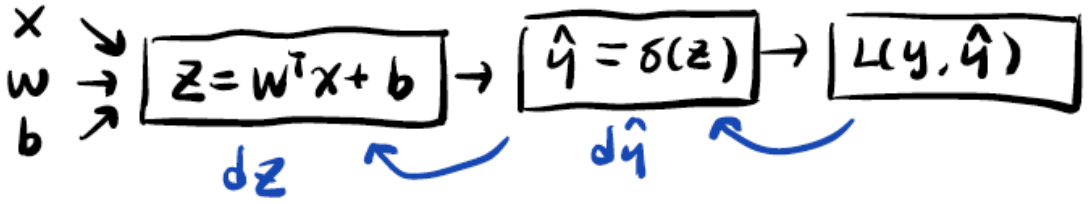
  $$L(w, b) = -\sum_{i=1}^{m} [y^{(i)} log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

Lecture 3: Neural Networks and Backpropagation

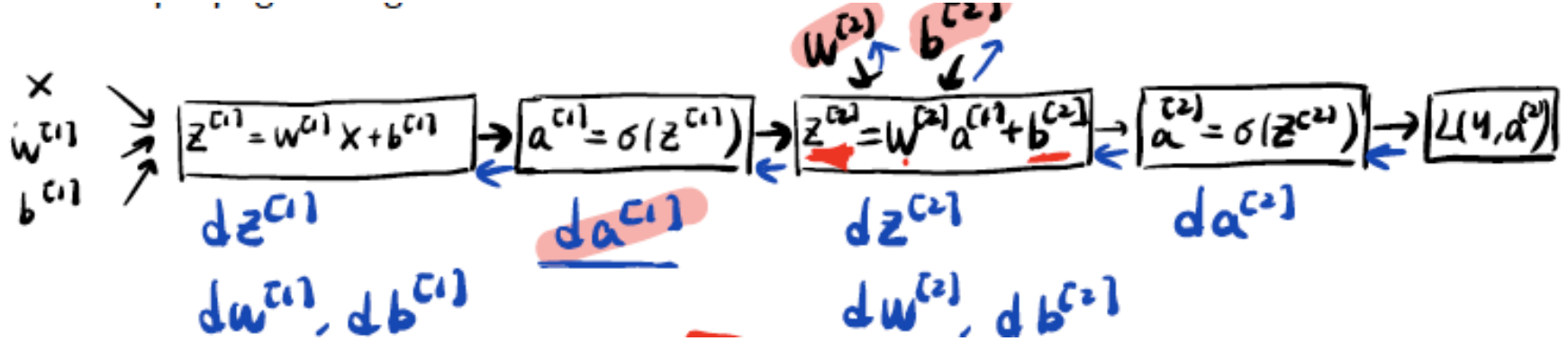
UC San Diego

- Computation Graph: Forward and Backward Computation

Logistic Regression



One hidden layer MLP

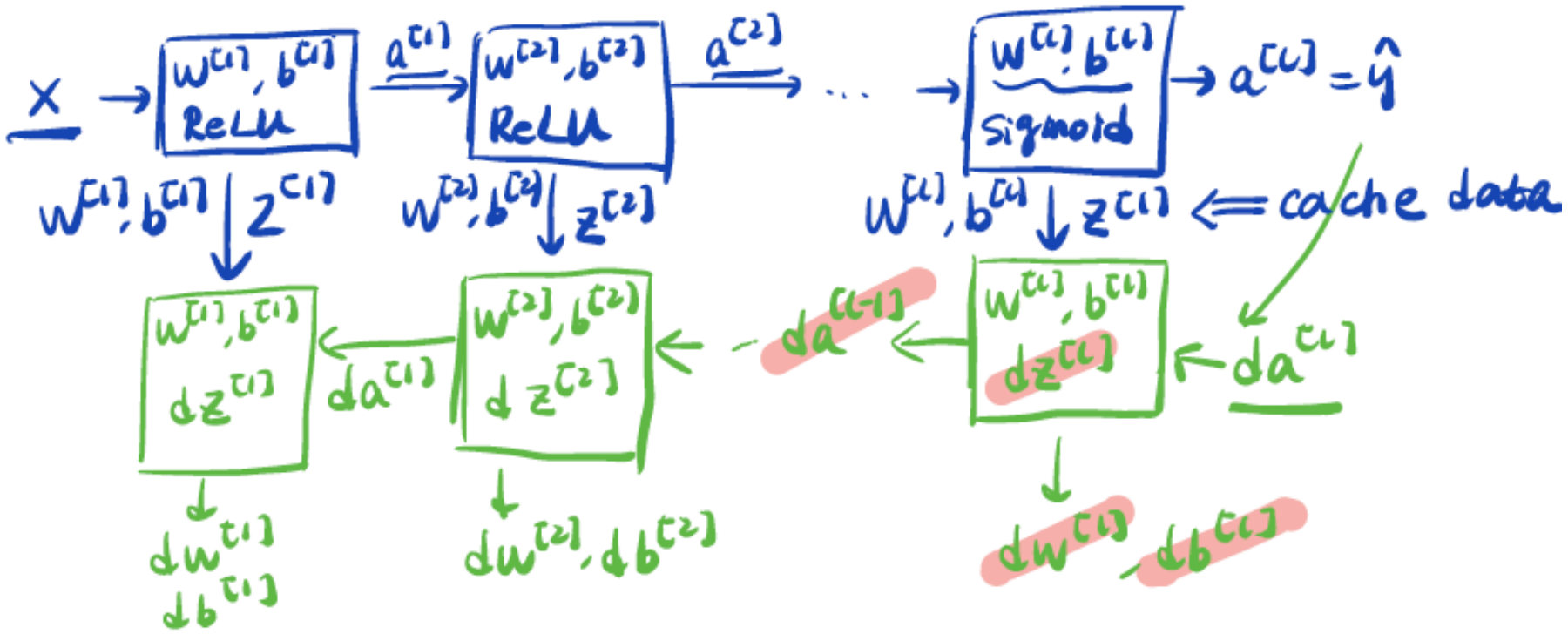


- Activation functions $\sigma(\cdot)$: sigmoid, tanh, ReLU, LeakyReLU; and their derivatives

Lecture 3: Neural Networks and Backpropagation

UC San Diego

- Forward and Backward Computation of L-layer MLP



- Universal function approximation (1-layer NN) and why deep?

- Gradient descent / mini-batch gradient descent
- Gradient descent with momentum

$$\boldsymbol{V_{dw} = \beta V_{dw} + (1-\beta)dw, V_{db} = \beta V_{db} + (1-\beta)db}$$

$$w^{(k+1)} = w^{(k)} - \eta V_{dw}, b^{(k+1)} = b^{(k)} - \eta V_{db}$$

- RMSprop (similar as Adagrad)

$$\boldsymbol{S_{dw} = \beta_2 S_{dw} + (1-\beta_2)(dw)^2, S_{db} = \beta_2 S_{db} + (1-\beta_2)(db)^2}$$

$$w^{(k+1)} = w^{(k)} - \eta \frac{dw}{\sqrt{S_{dw}} + \varepsilon}, b^{(k+1)} = b^{(k)} - \eta \frac{db}{\sqrt{S_{db}} + \varepsilon}$$

- Adam Optimizer

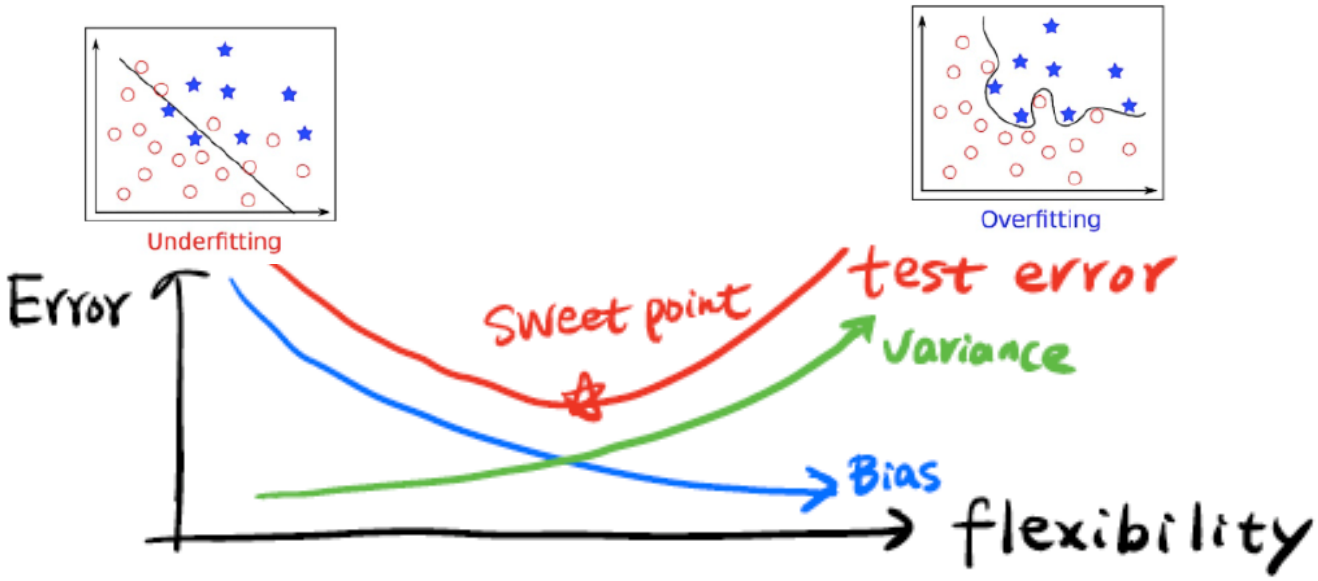
$$w^{(k+1)} = w^{(k)} - \eta \frac{\boldsymbol{V_{dw}}}{\sqrt{\boldsymbol{S_{dw}}} + \boldsymbol{\varepsilon}}, b^{(k+1)} = b^{(k)} - \eta \frac{\boldsymbol{V_{db}}}{\sqrt{\boldsymbol{S_{db}}} + \boldsymbol{\varepsilon}}$$

Lecture 4: Optimization & Regularization in Deep Learning

UC San Diego

- Random weight initialization (and why we need it?)
- Learning rate decay
- Some recent optimization landscape conjecture in deep learning
  - No poor local minima
  - Need to escape saddle points effectively

- Regularization in Deep Learning
  - Bias and variance tradeoff
  - Dropout
  - **Batch normalization**
  - Early stopping
  - Weight regularization



Course Outline

UC San Diego

Part 0: Introduction & Fundamentals

- Lecture 2: Supervised Learning Setup
- Lecture 3: Neural Networks and Backpropagation
- Lecture 4: Optimization & Regularization in Deep Learning

Part 1: Modeling and Learning in Physical Systems

- Lecture 5-6: Convolutional Neural Networks
- Lecture 7: Recurrent Neural Networks
- Lecture 8-9: Graph Neural Networks
- Lecture 10-11: Physics Informed Neural Networks and Neural Operator

Lecture 5-6: Convolutional Neural Networks

UC San Diego

- Convolution Layer
  - Filter (f)
  - Padding (p)
  - Stride (s)

Input (6×6)

10	10	10	10	10	10
10	10	10	10	10	10
10	10	10	10	10	10
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Filter (3×3)

1	1	1
0	0	0
-1	-1	-1

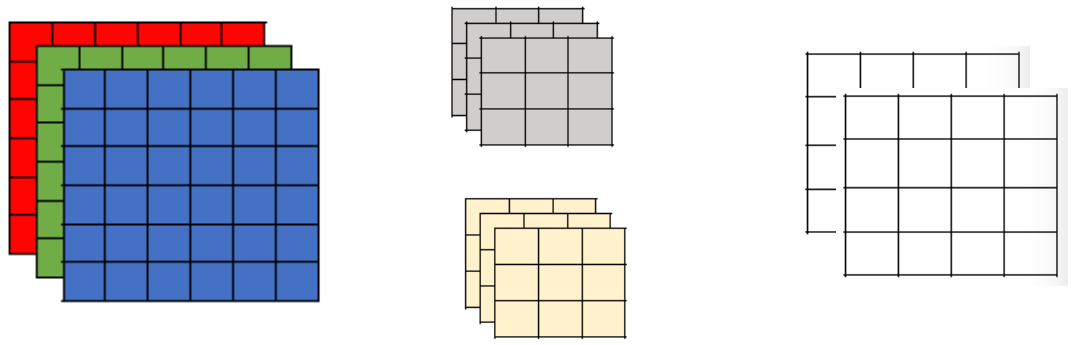
=

Output (4×4)

0	0	0	0
30	30	30	30
30	30	30	30
0	0	0	0

$$\text{output size: } \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

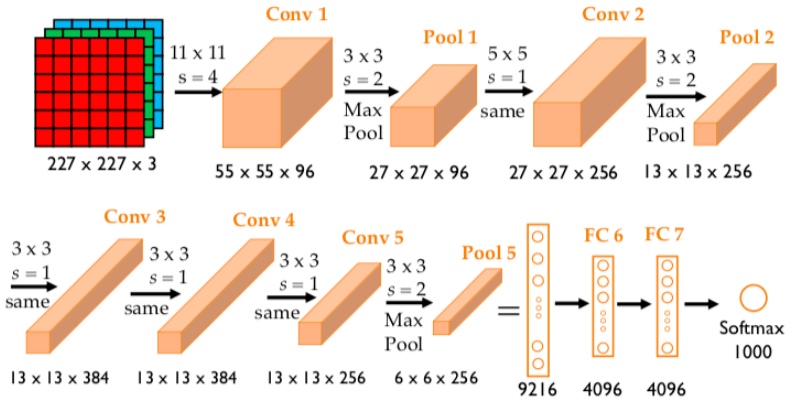
- Convolution over different channels
- More filters

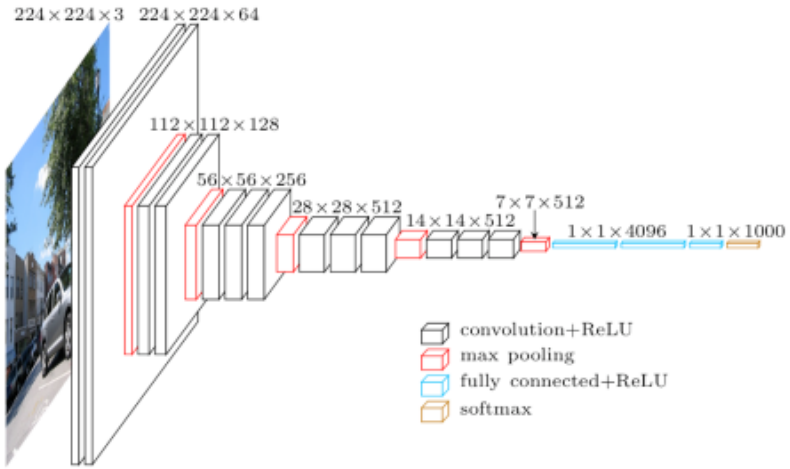


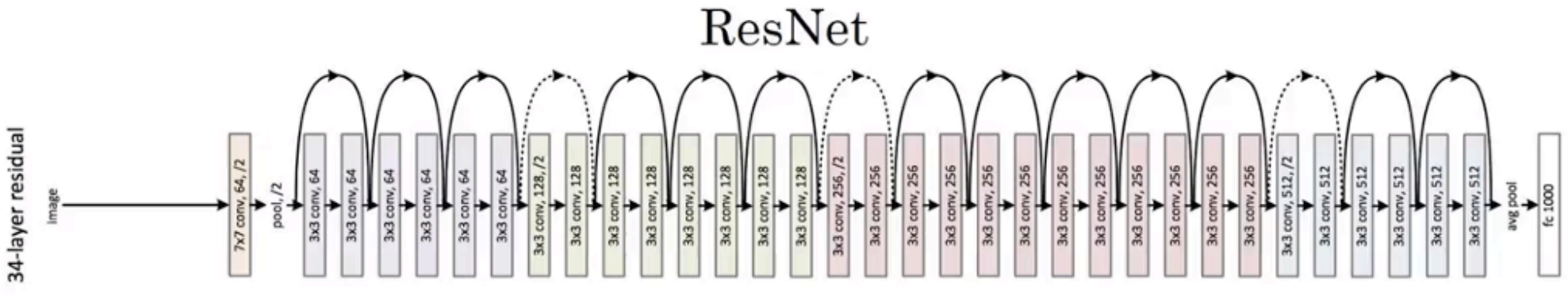
Lecture 5-6: Convolutional Neural Networks

UC San Diego

- Pooling Layer
- Classic Networks
  - LeNet – 5
  - AlexNet
  - VGG-16
  - ResNet



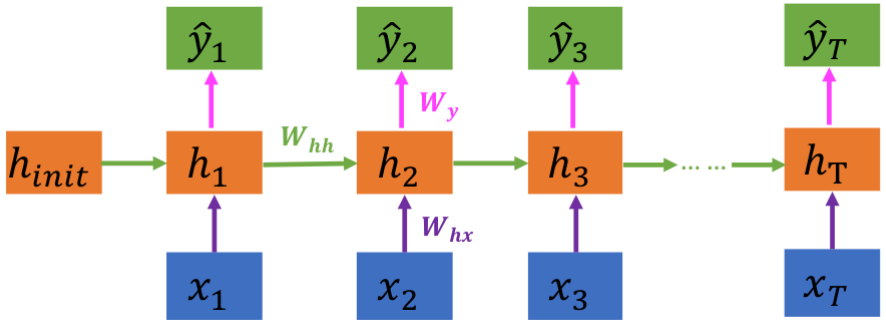




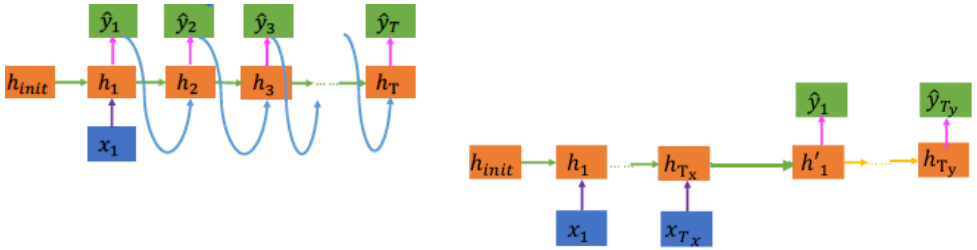
Lecture 7: Recurrent Neural Networks

UC San Diego

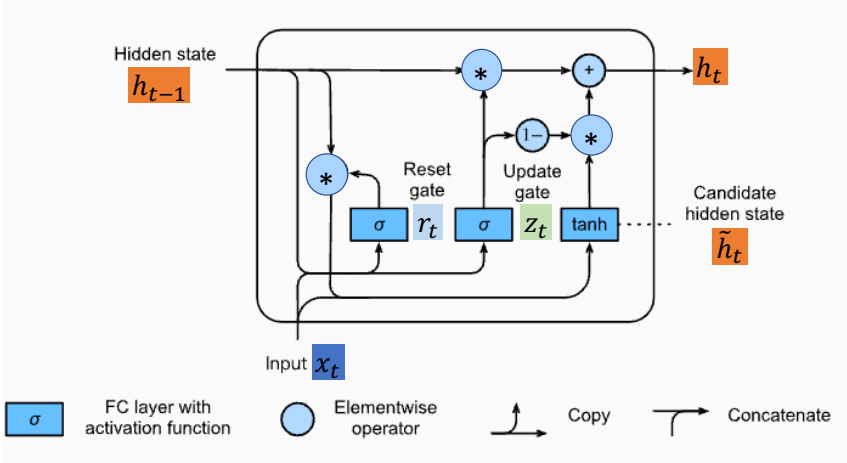
- Recurrent Neural Network: forward and backward computation through time

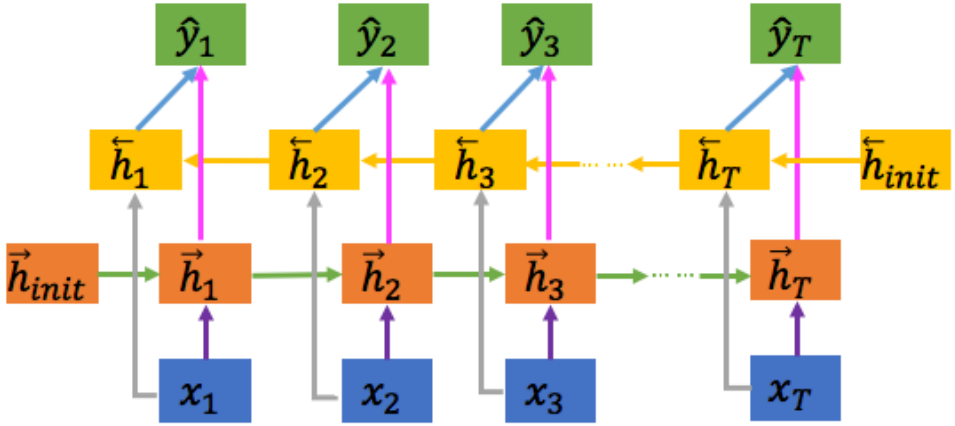


- One to many
- Many to one
- Many to many
- Many to many (input and output of different length)



- Modern Architectures: Gated Recurrent Units (GRU), LSTM, Bidirectional RNN

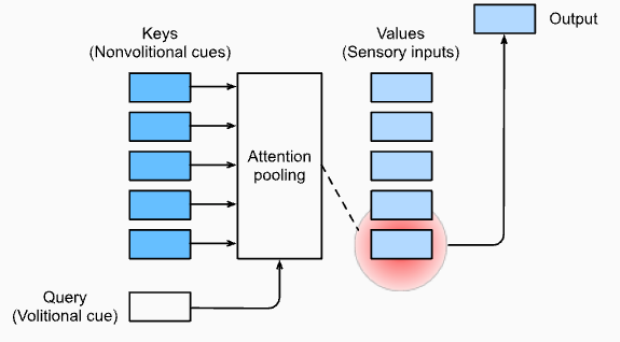




Lecture 7: Recurrent Neural Networks

UC San Diego

- Attention Mechanism



Query: q; Keys and value pairs: $(k_1, v_1), (k_2, v_2), \ldots, (k_m, v_m)$

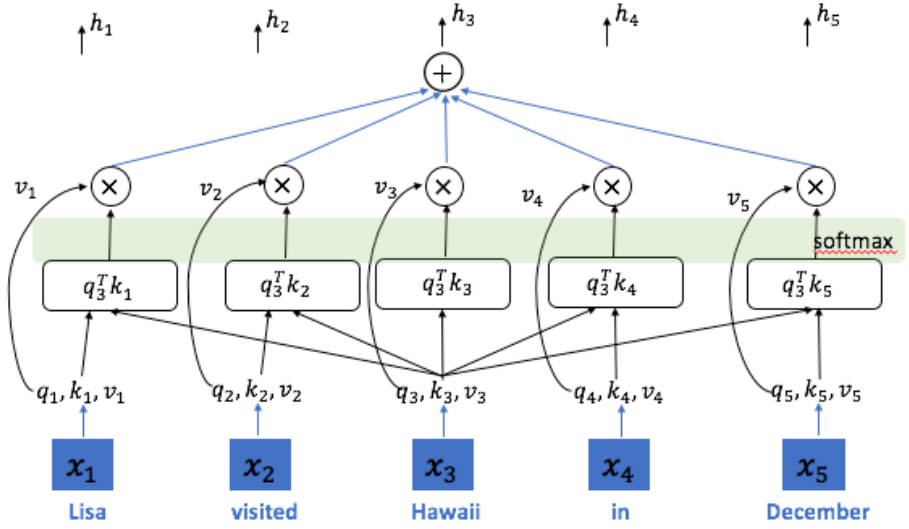
$$f(q) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$

$\alpha(q, k_i) = \frac{exp(a(q,k_i))}{\sum_{j=1}^{m} exp(a(q,k_j))} \in (0,1)$, $a(\cdot,\cdot)$ is the attention score function.

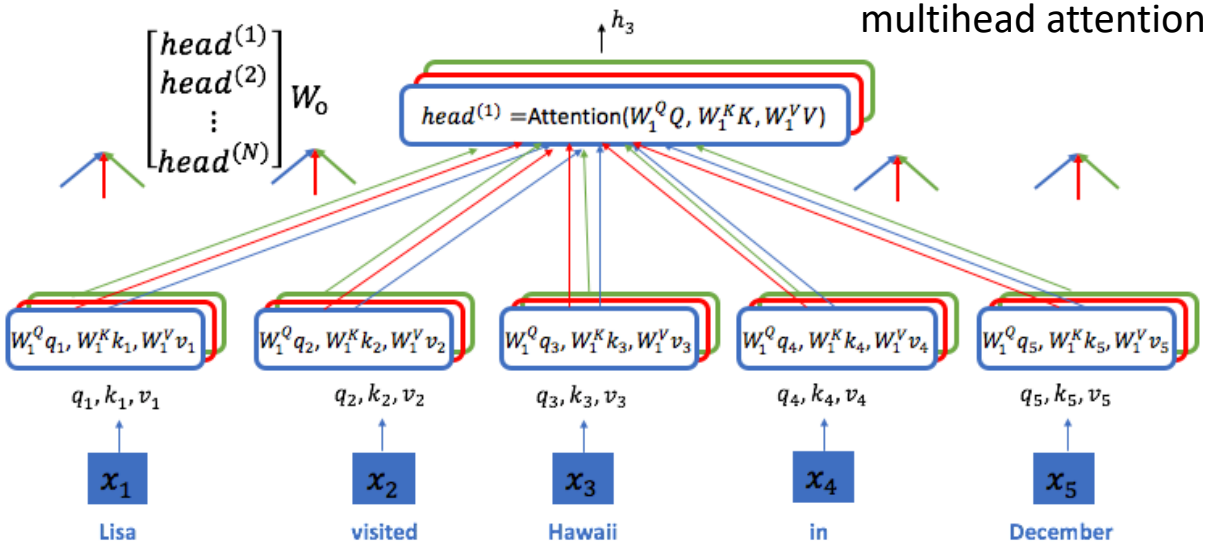
$$a(q, k_i) = \frac{k_i^T q}{\sqrt{d_k}} \; or \; a(q, k_i) = W_a \sigma(W_q q + W_k k_i)$$

- Transformer

self attention



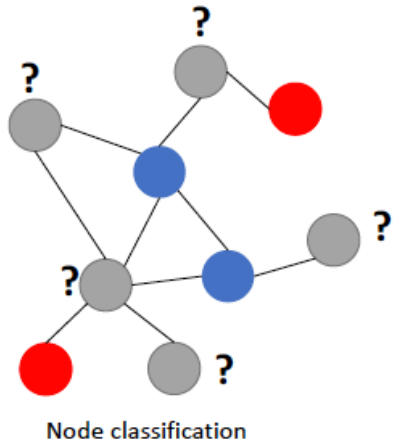
multihead attention



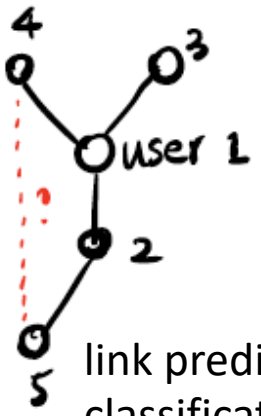
Lecture 8-9: Graph Neural Networks

UC San Diego

- Node embedding is the key for different graph tasks

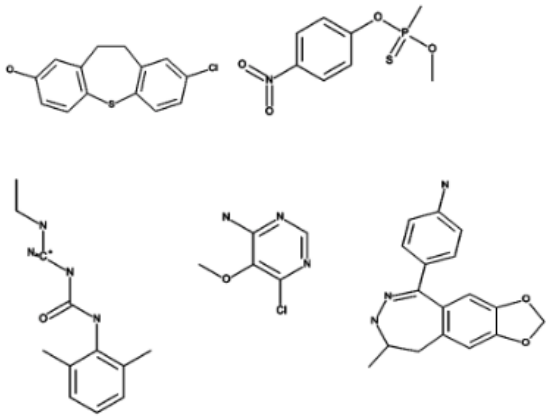


Node classification

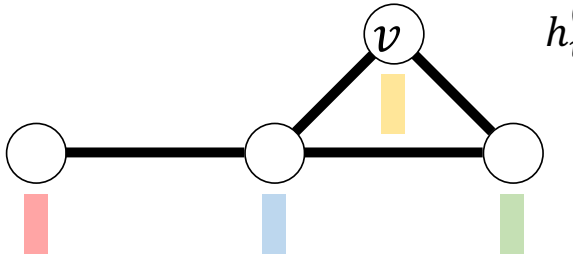


link prediction or classification

Graph-level task: toxic molecule or not?



- 3 popular graph neural network architectures
  - Graph convolutional networks
  - Graph Sample and Aggregate (GraphSAGE)
  - Graph Attention Networks



$$m_v = \text{Agg}(\{\,\})$$

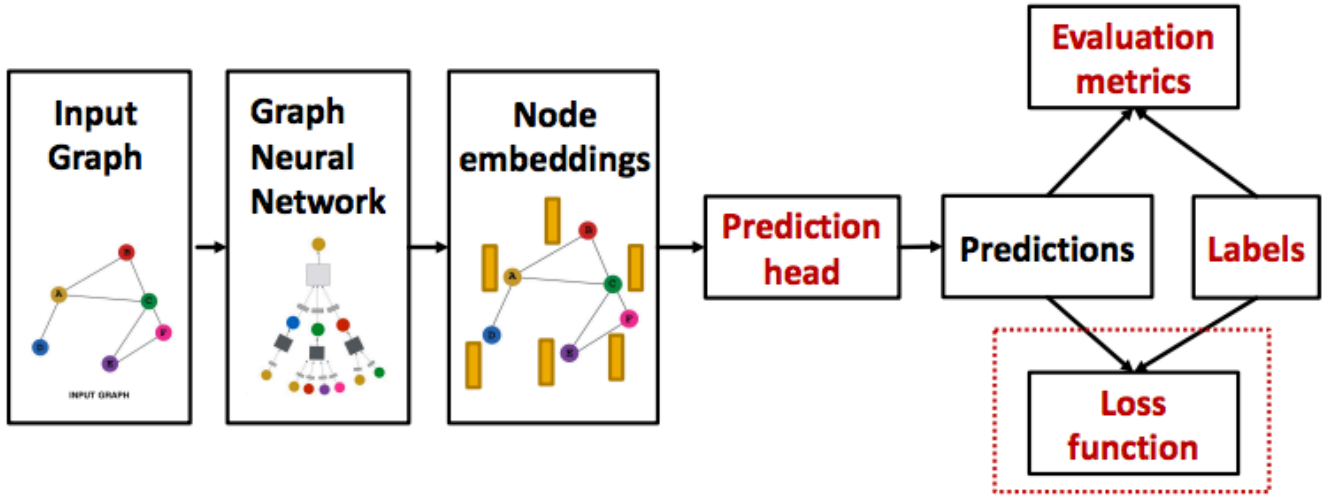
$$h_v^{(k)} = f^{(k)}[\text{combine}\,(x_v, m_v)\,]$$

1-layer of GNN

Lecture 8-9: Graph Neural Networks

UC San Diego

- Pipeline: end-to-end training of GNN and prediction models



Some example applications:

Node classification

$\hat{y}_v = \text{prediction model}\left(\text{h}_\text{v}^{(k)}\right)$

Loss function:

$L(\hat{y}_v, y_v) = -\sum_{c \in all\ classes} y_v^{(c)} \log \hat{y}_v^{(c)}$

Edge prediction

$\hat{y}_v = \text{prediction model}\left(\text{h}_\text{v}^{(k)}\right), P_{uv} = sigmoid(\hat{y}_v^T \hat{y}_u)$

Loss function:

$L(\hat{y}_v, \hat{y}_u, e_{uv}) = -e_{uv} log P_{uv} - (1 - e_{uv}) \log(1 - P_{uv})$

Graph level task

$G^{(k)} = \text{Agg/Pool}[h_{v1}^{(k)}, h_{v2}^{(k)}, \ldots, h_{vN}^{(k)}]$

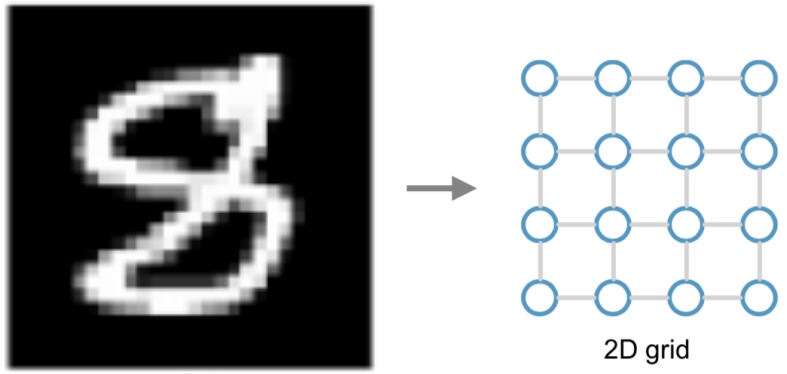
$\hat{y}_G = \text{prediction model}\left(G^{(k)}\right)$

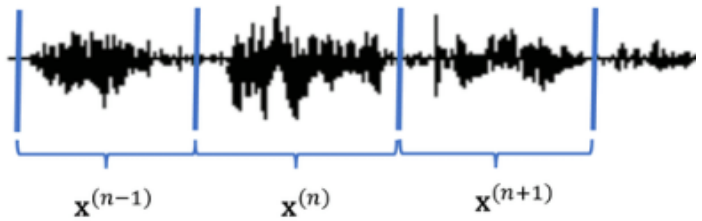
Loss function: $L(\hat{y}_G, y_G) = -\sum_{c \in all\ classes} y_G^{(c)} \log \hat{y}_G^{(c)}$

CNN, RNN, GNN

UC San Diego

basic convolutional neural networks, recurrent networks and graph neural networks can all be viewed as special cases of standard neural networks (MLP) with shared weights and skip connections

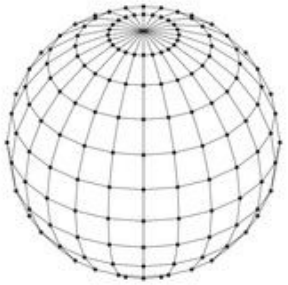




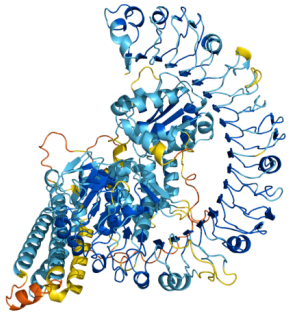
CNN operates on 2D spatial structure

RNN operates on 1D temporal structure

- GNN can operate on spatial data with complex geometry structure



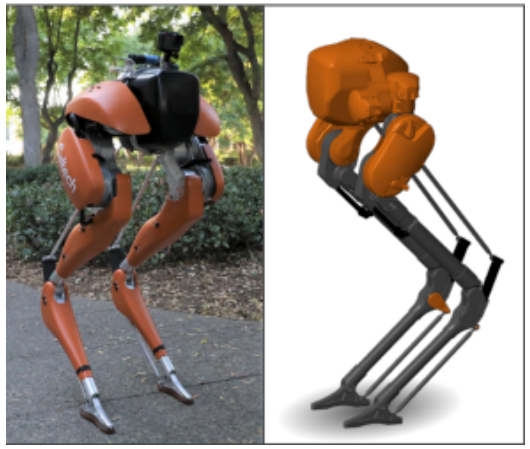
spherical



Lecture 10-11: Physics Informed Neural Networks and Neural Operator

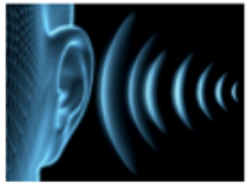
UC San Diego

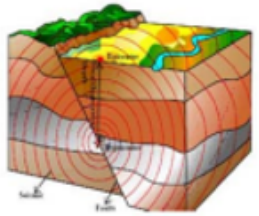
- Many physical systems are governed or described by ODEs and PDEs



1D Wave Equation: $\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$


water wave


air wave

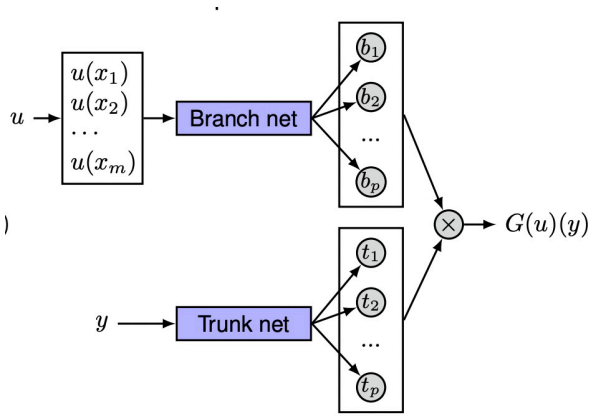

earth wave

- If we know the form of the differential equations
  - Could use PINN to solve and predict how the system trajectory will evolve
  - Or use PINN for learning/identifying the unknown parameters in the systems

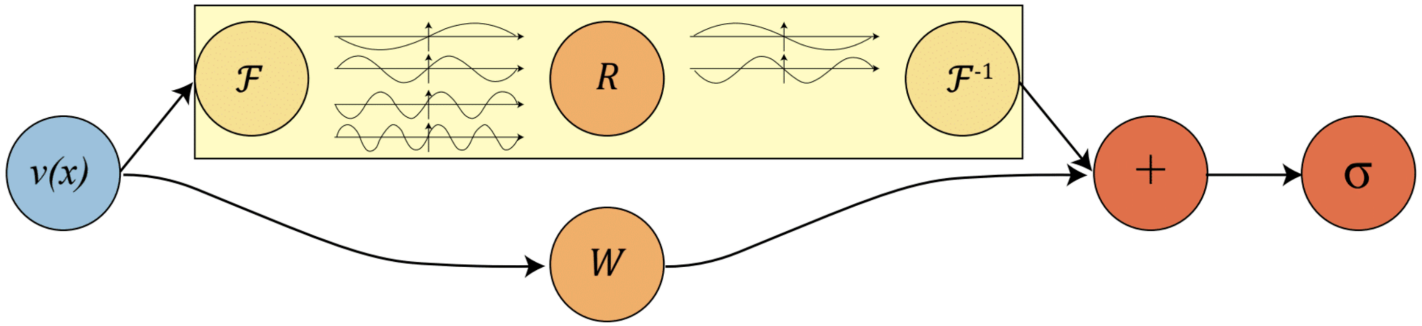
Lecture 10-11: Physics Informed Neural Networks and Neural Operator

UC San Diego

- If we only have data collected from the physical systems (governed or described by ODEs and PDEs), we can learn a neural operator to model and predict the system behaviors.



DeepONet



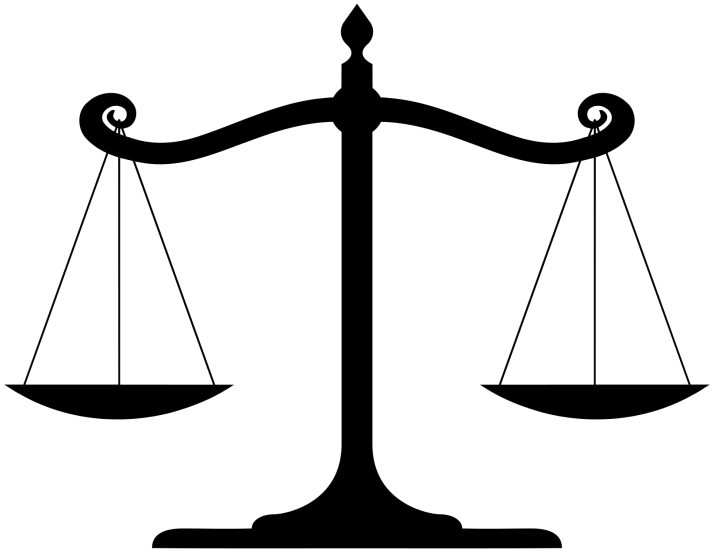
Fourier Neural Operator

Summary: Modeling and Learning in Physical Systems

UC San Diego

Priors: things assumed beforehand

- Which model to use?
- Which prior knowledge (physics law/constraints/…) to enforce?



Learning: things extracted from data

Basic supervised learning recipe

- Gather training data; decide model class
- Define loss function
- Improve the model by minimizing loss (optimization problem – often use gradient descent in deep learning)

- Balance between prior and learning:
  - Strong prior, minimal learning: can lead to bias if the prior isn't aligned with the data
  - Weak prior, much learning: may lead to high variance if overfitting to the noise in the data
- Choose the priors and collect data to obtain a model that achieves desired performance on your task

Course Outline

UC San Diego

Part 0: Introduction & Fundamentals

- Lecture 2: Supervised Learning Setup
- Lecture 3: Neural Networks and Backpropagation
- Lecture 4: Optimization & Regularization in Deep Learning

Part 1: Modeling and Learning in Physical Systems

- Lecture 5-6: Convolutional Neural Networks
- Lecture 7: Recurrent Neural Networks
- Lecture 8-9: Graph Neural Networks
- Lecture 10-11: Physics Informed Neural Networks and Neural Operator

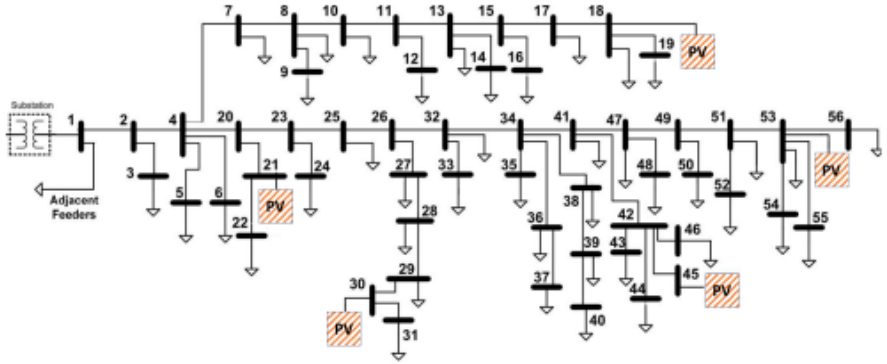
Part 2: Decision Making in Physical Systems

Decision Making in Physical Systems

UC San Diego

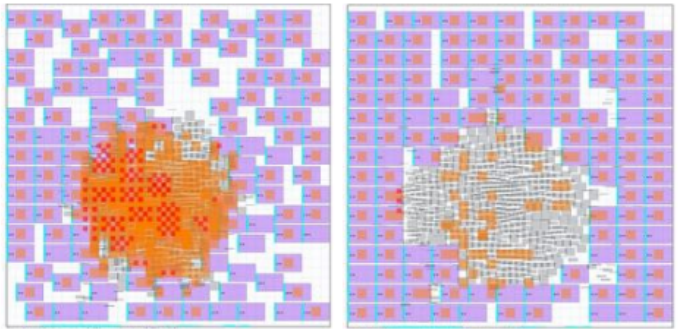
Robots: decide where to move the arm

Autonomous vehicles: to accelerate or slow down?



Power Grid: decide how much power to generate

Supply chain: decide how much inventory to replenish



Chip Placement

Uncertainty in the Physical Systems

UC San Diego

Robots: decide where the arm move to, but actuators can fail, hit unseen obstacles, etc

Renewable generators: decide how much to produce, but sun does not always shine (cloud) and wind does not always blow

Inventory management: decide what to replenish, but don't know the customer demand for various products

Outline

UC San Diego

- **Markov Decision Process**
- Value Iteration
- Policy Evaluation

Overview of Concepts

UC San Diego

- Markov Decision Process
  - State
  - Action
  - Transition Probability
  - Reward (and discount factor)
- Policy
- Return
- Value function

Markov Property

UC San Diego

- **Markov property:** A state S_t is Markov if and only if

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \dots, S_t]$$

- The future is independent of the past given the present

- **State transition probability:** from a Markov state s to next state s', the state transition probability is defined by:

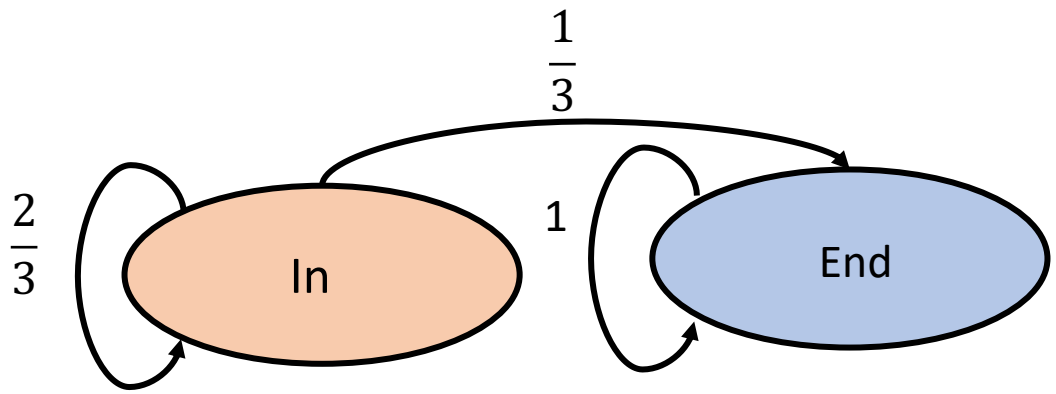
$$P_{ss'} = P[S_{t+1} = s'|S_t = s]$$

- **State transition matrix**

Markov Process & Example

UC San Diego

- A Markov Process (or Markov Chain) in a tuple $< S, P >$
  - S is a finite set of states
  - P is a state transition probability matrix, $P_{ss'} = P[S_{t+1} = s' | S_t = s]$



Dice Game Markov Chain

For each round r = 1,2, …

- Roll a 6-sided dice
- If the dice results in 1 or 2, game ends
- Otherwise, game continues to the next round

Markov Decision Process

UC San Diego

- A Markov Decision Process in a tuple $< S, A, P, R, \gamma >$
  - S is a finite set of states
  - A is a finite set of actions
  - P is a state transition probability matrix, $P_{ss'}^{a} = P[S_{t+1} = s' | S_t = s, A_t = a]$
  - R is a reward function, $R_s^a = R(R_{t+1} | S_t = s, A_t = a)$
  - γ is a discount factor $\gamma \in [0, 1]$

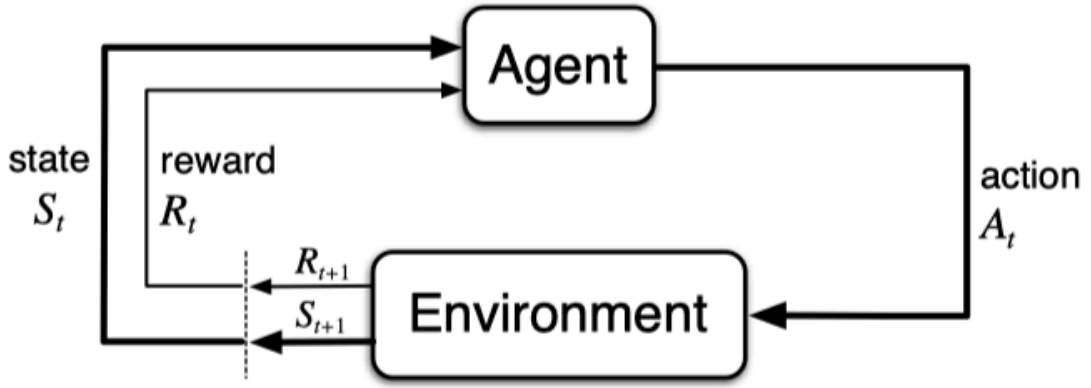
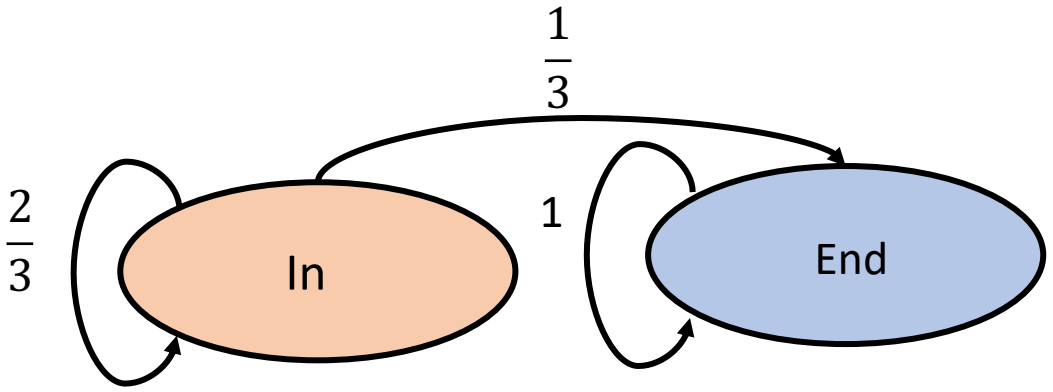


Figure 3.1: The agent–environment interaction in a Markov decision process.

Markov Decision Process Example

UC San Diego



Dice Game Markov Chain

For each round $r = 1,2, \ldots$

- You choose stay or quit
- If quit, you get $\$10$ and the game ends
- If stay, you get $\$4$ and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

Two Sets of Notations?

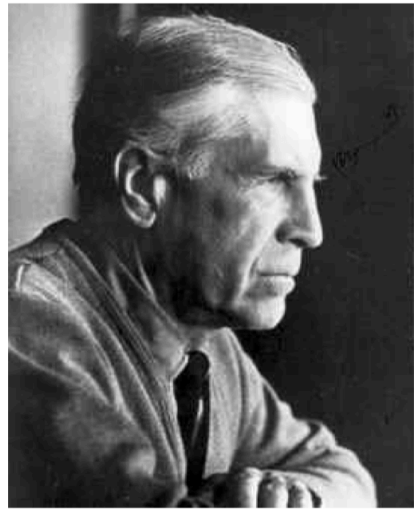
UC San Diego

$\mathbf{s}_t$ – state
$\mathbf{a}_t$ – action
$r(\mathbf{s}, \mathbf{a})$ – reward function

$\mathbf{x}_t$ – state
$\mathbf{u}_t$ – action
$c(\mathbf{x}, \mathbf{u})$ – cost function

$$r(\mathbf{s}, \mathbf{a}) = -c(\mathbf{x}, \mathbf{u})$$

Richard Bellman



Lev Pontryagin

Environment Model

UC San Diego

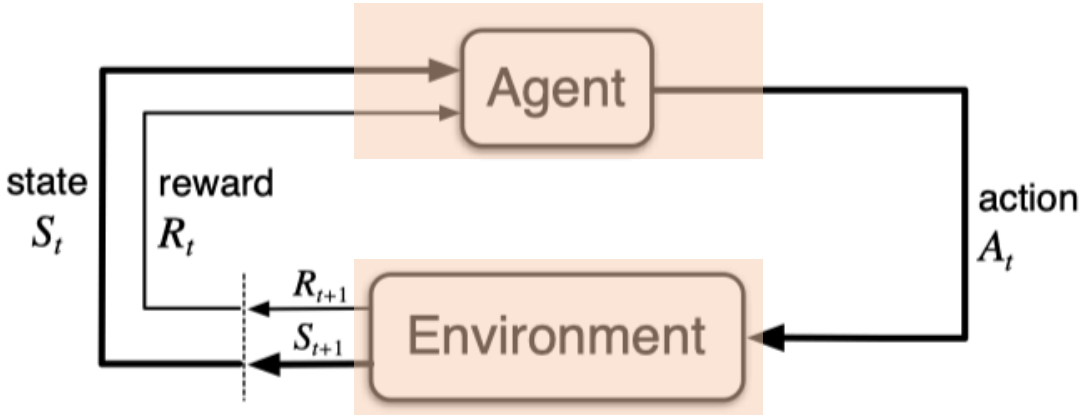


Figure 3.1: The agent–environment interaction in a Markov decision process.

- A model predicts what the environment will do next
- $\mathcal{P}$ predicts the next state
- $\mathcal{R}$ predicts the next (immediate) reward, e.g.

$$\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$
$$\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Policy

UC San Diego

- **A policy π** is a mapping from each state $s \in S$ to an action $a \in A$

$$\pi[a|s] = P[A_t = a|S_t = s]$$

- A policy full defines the behavior of an agent
- In MDP, policies only depend on the current state S_t, not the history
- Policy is stationary (time independent), $A_t \sim \pi(\cdot\,|S_t), \forall t > 0$

Example: policy "stay"

For each round r = 1,2, ...

- You choose stay or quit
- If quit, you get $10 and the game ends
- If stay, you get $4 and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

Sample **episodes** starting from $S_1 = \text{In}$

- [In (stay), End] $4
- [In (stay), In (stay), In (stay), End] $12
- [In (stay), In (stay), End] $8
- [In (stay), In (stay), In (stay), In (stay), End] $16
- ……

Policy

UC San Diego

- **A policy π** is a mapping from each state $s \in S$ to an action $a \in A$

$$\pi[a|s] = P[A_t = a|S_t = s]$$

- A policy full defines the behavior of an agent
- In MDP, policies only depend on the current state S_t, not the history
- Policy is stationary (time independent), $A_t \sim \pi(\cdot\,|S_t), \forall t > 0$

Example: policy "quit"

For each round r = 1,2, …

- You choose stay or quit
- If quit, you get $10 and the game ends
- If stay, you get $4 and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

Sample **episodes** starting from $S_1 = \text{In}$

- [In (quit), End] $10

Return

UC San Diego

- **The return G_t** is the total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The discount $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k + 1$ time-steps is $\gamma^k R$
- This values immediate reward above delayed reward
  - γ close to 0 leads to "greedy" evaluation (only care about current step)
  - γ close to 1 leads to "far-sighted" evaluation

Return

UC San Diego

- **The return G_t** is the total discounted reward from time step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

For each round r = 1,2, ...

- You choose stay or quit
- If quit, you get $10 and the game ends
- If stay, you get $4 and then we roll a 6-sided dice
  - If the dice results in 1 or 2, game ends
  - Otherwise, game continues to the next round

Discount $\gamma = 1$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 4 + 4 + 4 = \$16$

Discount $\gamma = 0$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 0 \cdot 4 + 0 \cdot 4 + 0 \cdot 4 = \$4$

Discount $\gamma = 1/2$: [In (stay), In (stay), In (stay), In (stay), End] $4 + 0.5 \cdot 4 + 0.5^2 \cdot 4 + 0.5^3 \cdot 4 = \$7.5$

Why Discount?

UC San Diego

- Mathematically convenient to use discount rewards
- Animal/human behavior shows preference for immediate reward
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Sometimes, it is possible to use undiscounted reward if all possible episodes are finite
- ……

Value function

UC San Diego

The **state-value function $v_\pi(s)$ of a policy π** in an MDP is the expected return starting from state s, and then following policy π

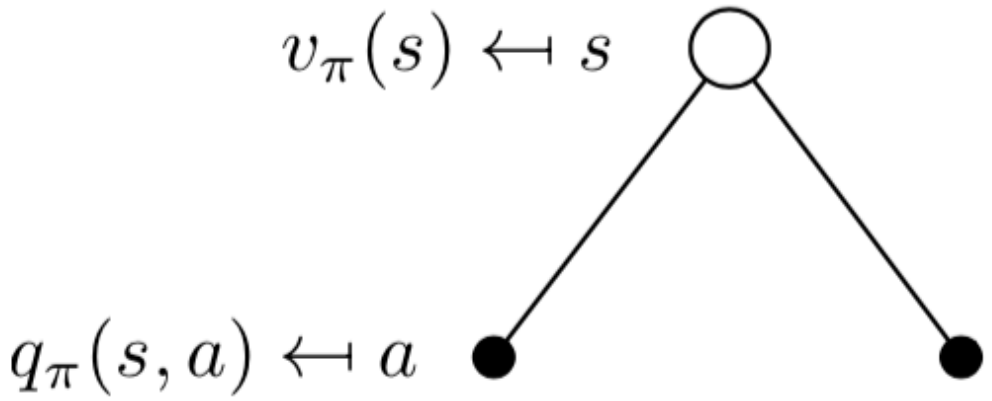
$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

The **state-action value function $q_\pi(s, a)$ of a policy π** (sometimes also refer to as action value function) is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Value function

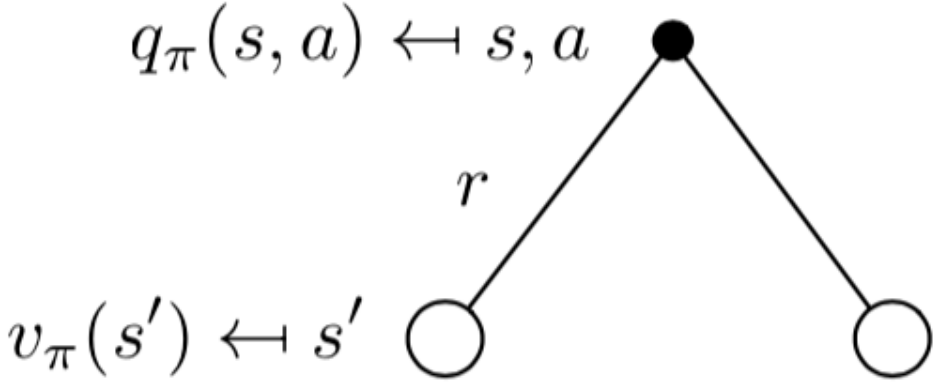
UC San Diego



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Value function

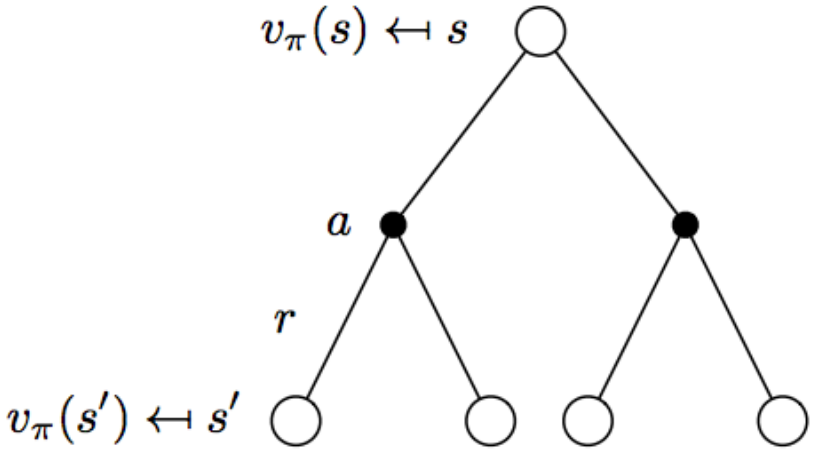
UC San Diego



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Value function

UC San Diego



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Summary

UC San Diego

- Markov Decision Process $< S, A, P, R, \gamma >$
  - State
  - Action
  - Transition Probability
  - Reward (and discount factor)
- Policy
- Return
- Value function: state value function $v_\pi(s)$, action value function $q_\pi(s, a)$