SIGGRAPH 2020 S2020.SIGGRAPH.ORG THINK BEYOND

Walt Disney ANIMATION STUDIOS

JUNE 2020 WEBINAR

Hands-on Workshop: Machine Learning and Neural Networks

SIGGRAPHNOW

RAJESH SHARMA
SOFTWARE ENGINEER
Walt Disney Animation Studios

WALT DISNEY
ANIMATION STUDIOS

9

Machine Learning

Rajesh Sharma

Today

- Recap
  - Recurrent Neural Networks
- Reinforcement Learning
- Summary & Resources

Questions?

How are the weights selected for dropout? Is is random?	Greg Klar
The weights of the GAN are being modified in the gradient descent part of the training step, is that correct?	Bobby Bodenheimer
What forces the generator to diversity?	Greg Klar
How to make sure that our discriminator is not "too powerful" at the beginning of the training?	Ivan Puhachov
Why use GAN instead of VAE? Don't the VAE well behaved latent space offer more control over the generation of new samples?	Anonymous Attendee
Can you gnerate 3D objects with them ?	Ingeborg Tastl
Is an RNN also useful for generating video?	Marijn Eken
Does it matter what algorithm is used to convert text into numbers ?	Ingeborg Tastl

What are RNNs?

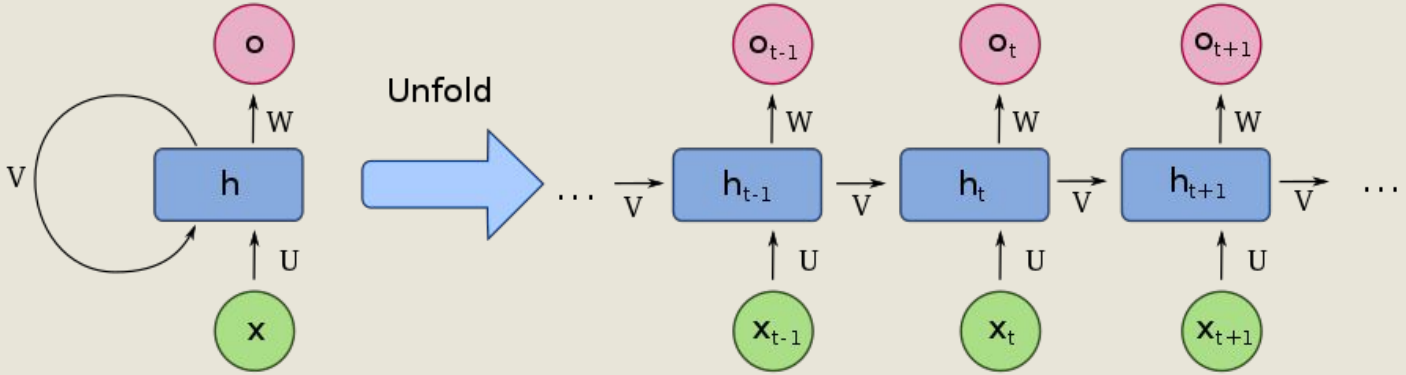
Information from previous timestep is passed forward

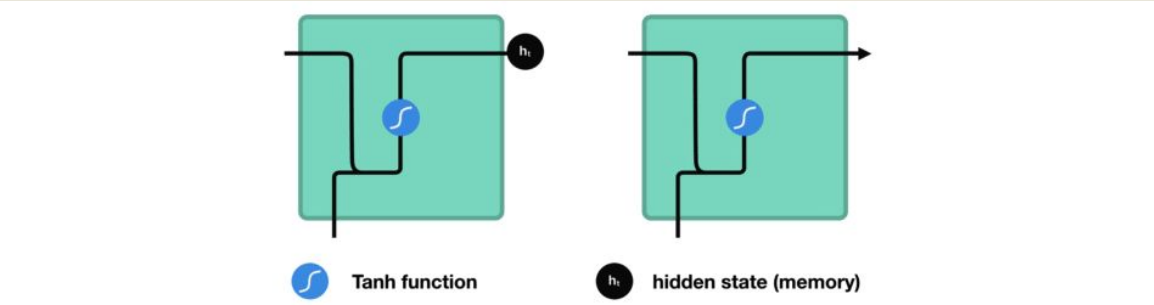
Useful for time-dependent data:

- Sequence prediction problems
- Language Translation: Speech, text, music

What are RNNs?

Information from previous timestep is passed forward

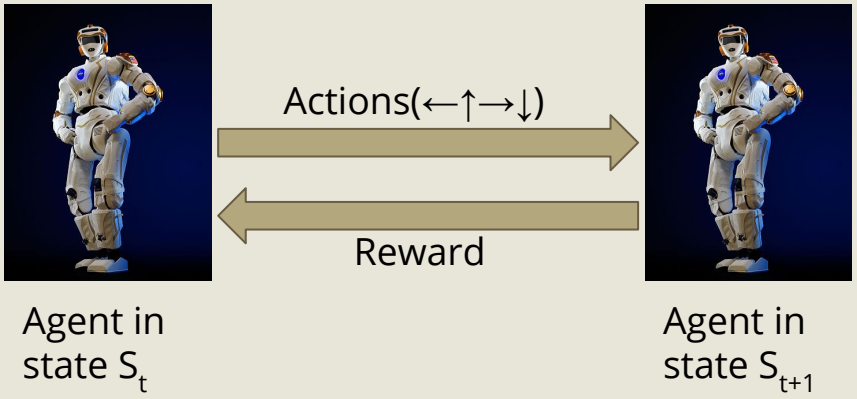


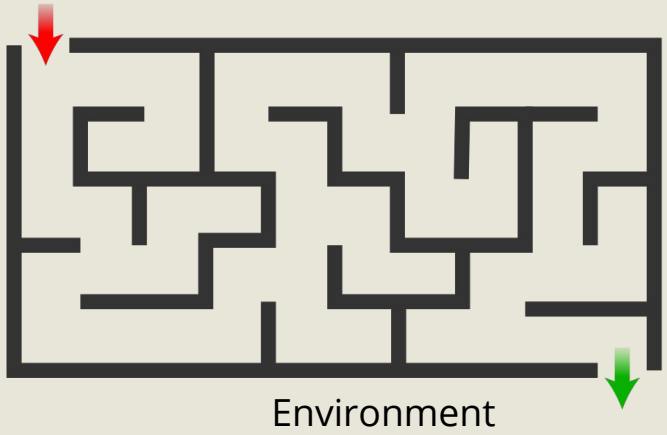


Today:

Reinforcement Learning:

Environment, states, actions, rewards





Goal: Maximize Total Reward

Challenges:

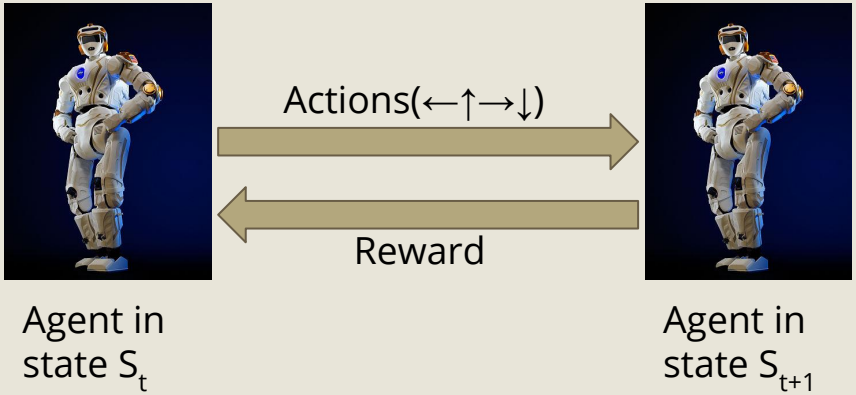
Actions & States

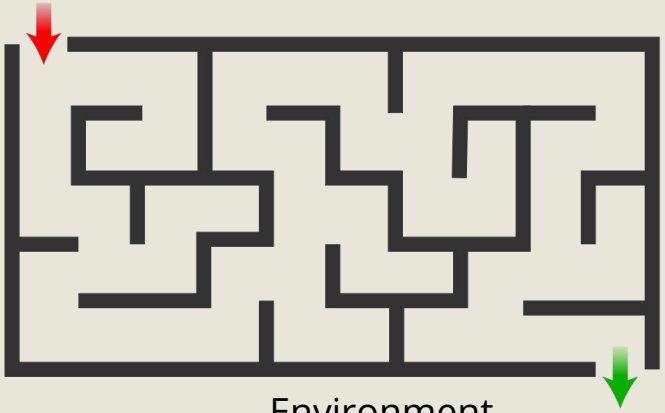
Model of environment

Policy for choosing next action

Reward at each step

Value Function





Goal: Maximize Total Reward

N-arm bandit problem

10-arm bandit problem:

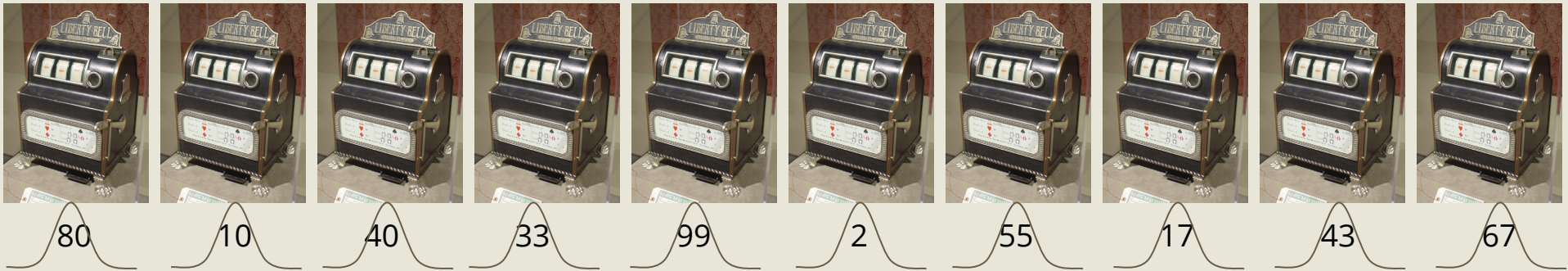
-- Given 10 choices for action

-- Each action has a fixed* reward (unknown)

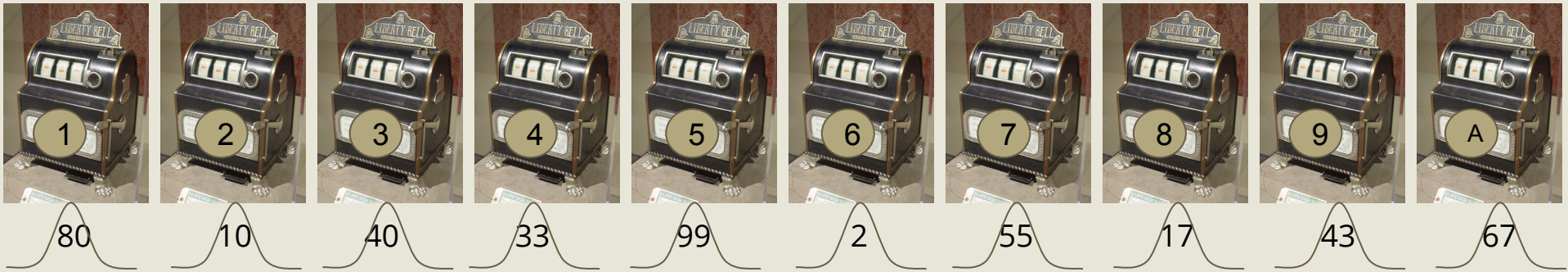
*Reward for each action varies but each action has a ‘fixed’ mean reward

Goal: Maximize total reward over a number of actions

10-arm bandit problem:



10-arm bandit problem:



Strategy 1: Naive-Randomly pick an action

Gets an average reward over all actions

Hands on...

Find and open:

tenArmBandit.ipynb

10-arm bandit problem:

Strategy 2: Greedy

- Keep a tally of actions and average reward
- Choose action with maximum current avg reward
  - Q-Action Table

Strategy 2: Greedy

80 10 40 33 99 2 55 17 43 67

10-arm bandit problem:

Strategy 3: Greedy-epsilon (exploit and explore)

Keep a tally of actions, average reward

Choose action with maximum current avg reward

But,

Once in a while,

choose another action randomly

10-arm bandit problem:

Strategy 4: Greedy-decaying-epsilon

- Keep a tally of actions, average reward
- Choose action with maximum current avg reward

  But,
  - Once in a while,
    - choose another action randomly
  - over time, do this less frequently

Hands on...

Find and open:

tenArmBandit.ipynb

Deep Q Learning (DQN)

-- DQN: Deep CNN to represent/learn Q

-- Experience Replay: uses a random sample of prior actions instead of the most recent action to proceed

-- Example: Space Invaders

-- Example: Walking

Policy Design, reward structure is important

Summary

-- Basics: regression, UAT, no free-lunch
-- Fully Connected: experiments, final layer
-- CNN: building block for image-based training
-- RNN, LTSM, GRU: time series, language, text
-- Unet, resNet: CNN-like with better detail transfer
-- Variational AutoEncoder: Generative:(mean,variance)
-- Transfer Learning: mt-cnn, facenet
-- GAN: Generative: direct sample
-- Reinforcement Learning: env, states, actions, rewards

Summary (Practice)

-- Bias vs Variance: Overtraining, Undertraining
-- Data: Lots of it, augment, un-biased
-- Hyperparameters: layers, nodes, optimizer, L-Rate
-- Loss function: logits (log-likelihood), L2, L1
-- Training: epochs, batches, tfds, plotting
-- Distributions: you are trying to find a sample
-- Work like a scientist:
    Hypothesis, experiment, observe, record, change:
    Repeat

Local Install (Linux/Mac)

```
# In Colab (after you have imported everything you need)
!pip freeze > requirements.txt
----------------------------------------

mkdir ~/myml
cd ~/myml

# copy the requirements.txt file from colab to ~/myml
# save these instructions as README.env in ~/myml

# create a python3 virtual environment
python3 -mvenv --system-site-packages mlenv

# activate the virtual environment
source mlenv/bin/activate

# upgrade the installer
pip install --upgrade pip

# Install all the software specified in requirements.txt
pip install -r requirements.txt

# test it out:
which python (this should point to the python in the virtual env)
```

Summary (not covered)

-- Cloud-based: Training and Deployment
-- Local: clusters, machines, environment
-- Tensorboard: for logging, visualizations
-- Intermediate layer visualization
-- Other methods: Random Forests, XGBoost
-- More theory

Where to go from here:

Deep Learning: https://www.deeplearningbook.org
and some excellent lectures to go along:
https://www.deeplearningbook.org/lecture_slides.html

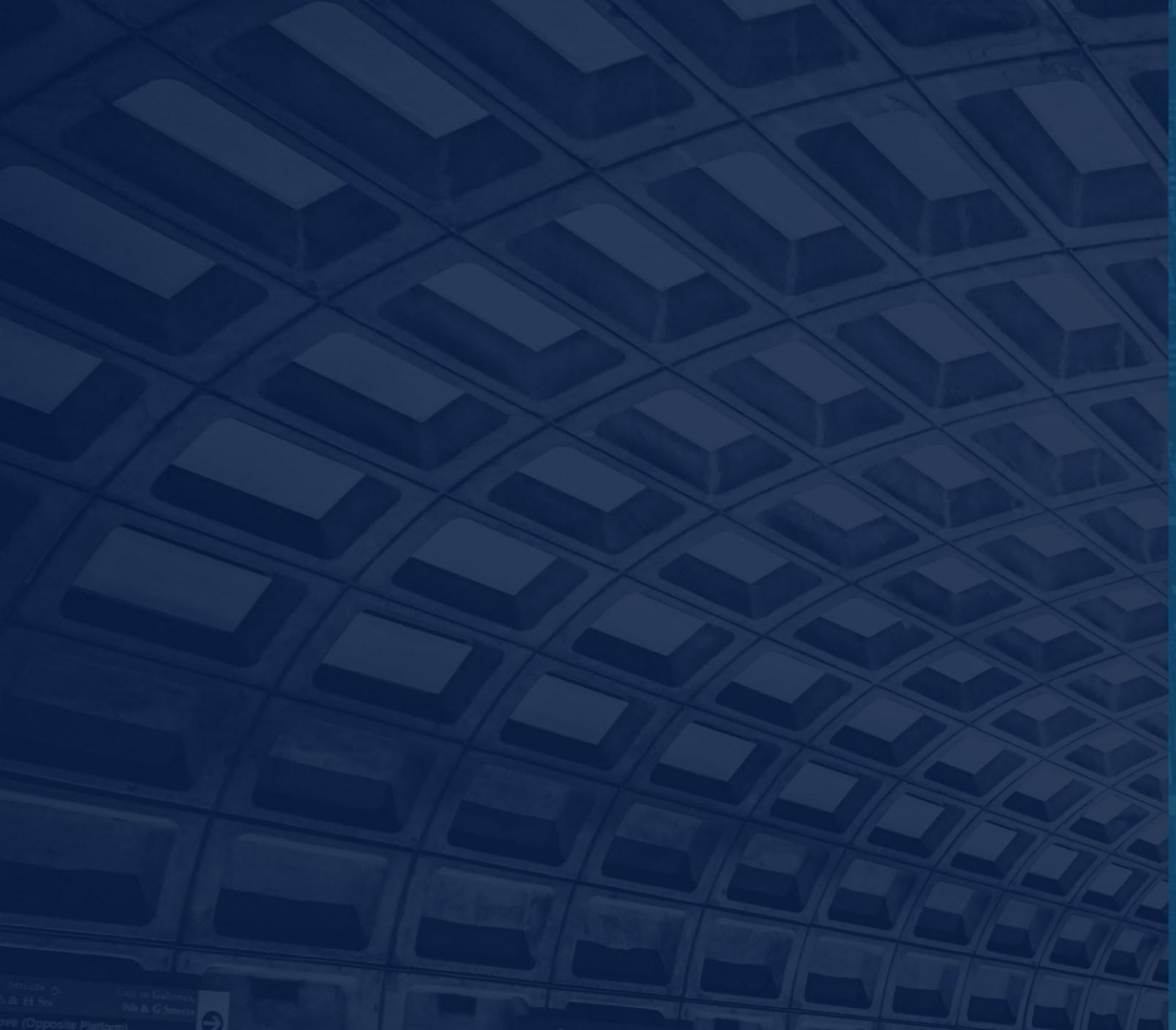
Reinforcement Learning:
http://incompleteideas.net/book/bookdraft2017nov5.pdf

Statistics:
https://link.springer.com/book/10.1007/978-0-387-21736-9

A roadmap to reading:
https://github.com/floodsung/Deep-Learning-Papers-Reading-Roadmap

More comprehensive list of resources:
https://www.kdnuggets.com/2020/03/24-best-free-books-understand-machine-learning.html

Video Tutorials (3Blue1Brown):
https://www.youtube.com/channel/UCYO_jab_esuFRV4b17AJtAw/videos



SIGGRAPH 2020 S2020.SIGGRAPH.ORG THINK BEYOND

QUESTIONS?

Submit now!

THANK YOU

WANT TO HEAR MORE FROM SIGGRAPH AND DISNEY?

Check out SIGGRAPH Now on YouTube

YouTube.com/user/ACMSIGGRAPH

Subscribe to the SIGGRAPH Spotlight podcast

SIGGRAPH SPOTLIGHT

STITCHER

iHeart PODCAST NETWORK

SOUNDCLOUD