CS 3100
Data Structures and Algorithms 2
Lecture 25: Intro to Machine Learning, Part 2

Co-instructors: Robbie Hott and Tom Horton

Fall 2023

Announcements

- Upcoming deadlines
  - PS5 (Max Flow, Reductions, ML), due December 5, 2023 at 11:59pm
  - PA5 (Tiling Dino) due December 5, 2023 at 11:59pm
- Interested in being a TA? Form coming later this week
- Quiz 5 (and Quiz 1-4 Retakes) on December 12, 2023 at 7pm
  - In our normal room
  - Final Exam conflict form – see announcement on Canvas
    - Only if you have an exam scheduled *at the same time* as our Quiz + Retakes
  - If you are an SDAC student, please schedule with them ASAP
- Course email (comes to both professors and head TAs):
  cs3100@cshelpdesk.atlassian.net

Where Are We?

1. This overview, and how many ML algorithms work with data
2. Clustering as an example of unsupervised learning
   - Single-link clustering (like PA3)
   - **Another algorithm: *k* means clustering**
3. A simple classification algorithm to show supervised learning
   - This algorithm only uses concepts you've learned already
   - Then a brief overview of other techniques
4. A brief intro to reinforcement learning

Another Algorithm: *k*-Means Clustering

- Several algorithms for this, but we'll examine the "naïve algorithm" (AKA Lloyd's algorithm)
- An **iterative refinement strategy** that works in steps
  1. At each step observations are assigned to one of *k* clusters
  2. Then assignments are updated to improve the clustering
     - How? Explained below
  3. Stop this iterative process when results converge and no improvement happens

Clustering Based on Centroids

We'll use the idea of a **cluster centroid**

- The average position of all the cluster's observations
- Calculated as the average value of each of the observations' coordinates in the *n*-dimensional space

In *k*-means clustering, each observation will belong to the cluster with the nearest centroid

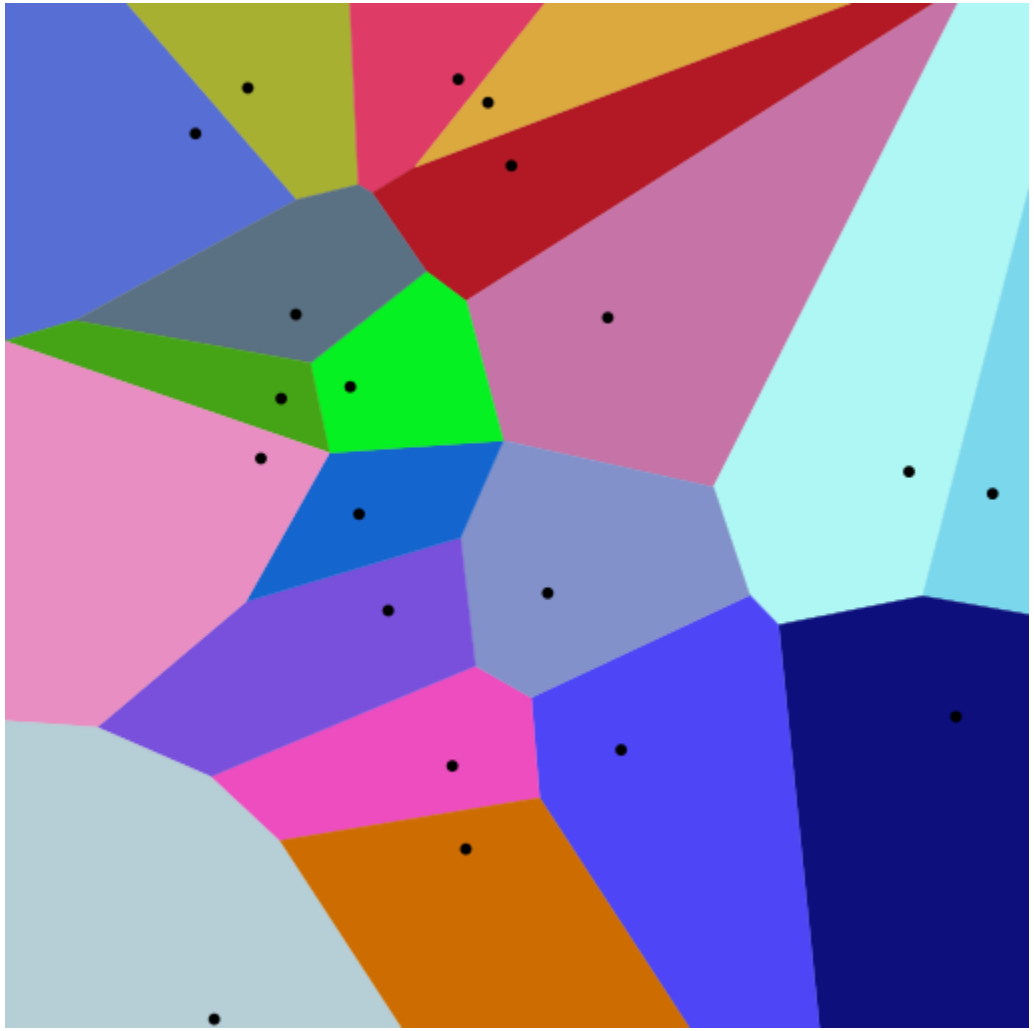
k-means clustering minimizes within-cluster variances of observations in the clusters

- Contrast with earlier single-link clustering algorithm, which maximizes the poorest of the intra-cluster distances

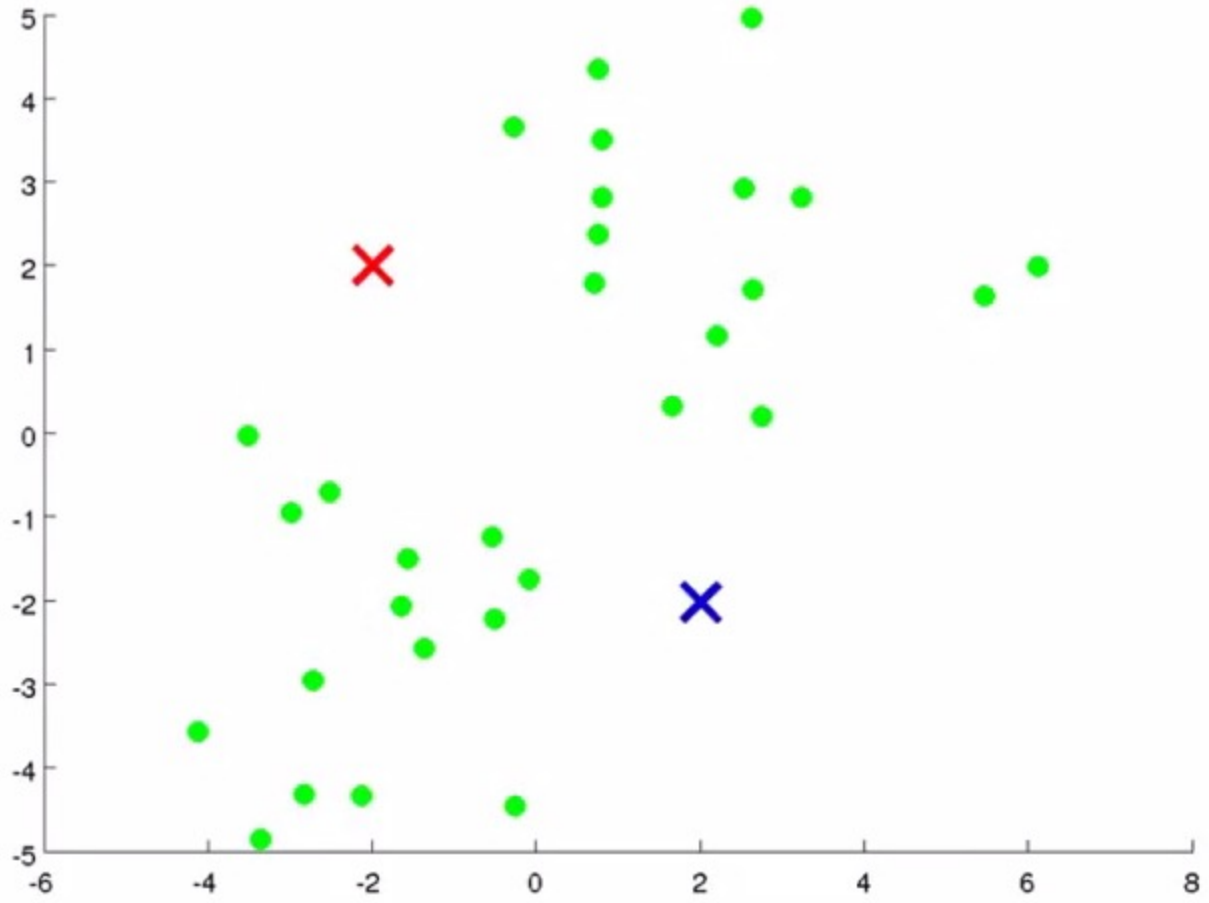
A related concept: Voronoi Cells

- A partitioning of planes into regions
- Each black dot is a *centroid*
- Every color represents the region with points closest to the centroid in that region

- *k*-means strategy reflects this idea, and the naïve algorithm works like this:
  - Initialize exactly *k* centroids
  - Assign each observation to the closest centroid
  - Update centroid locations because cluster membership has changed
  - Update assignments and repeat

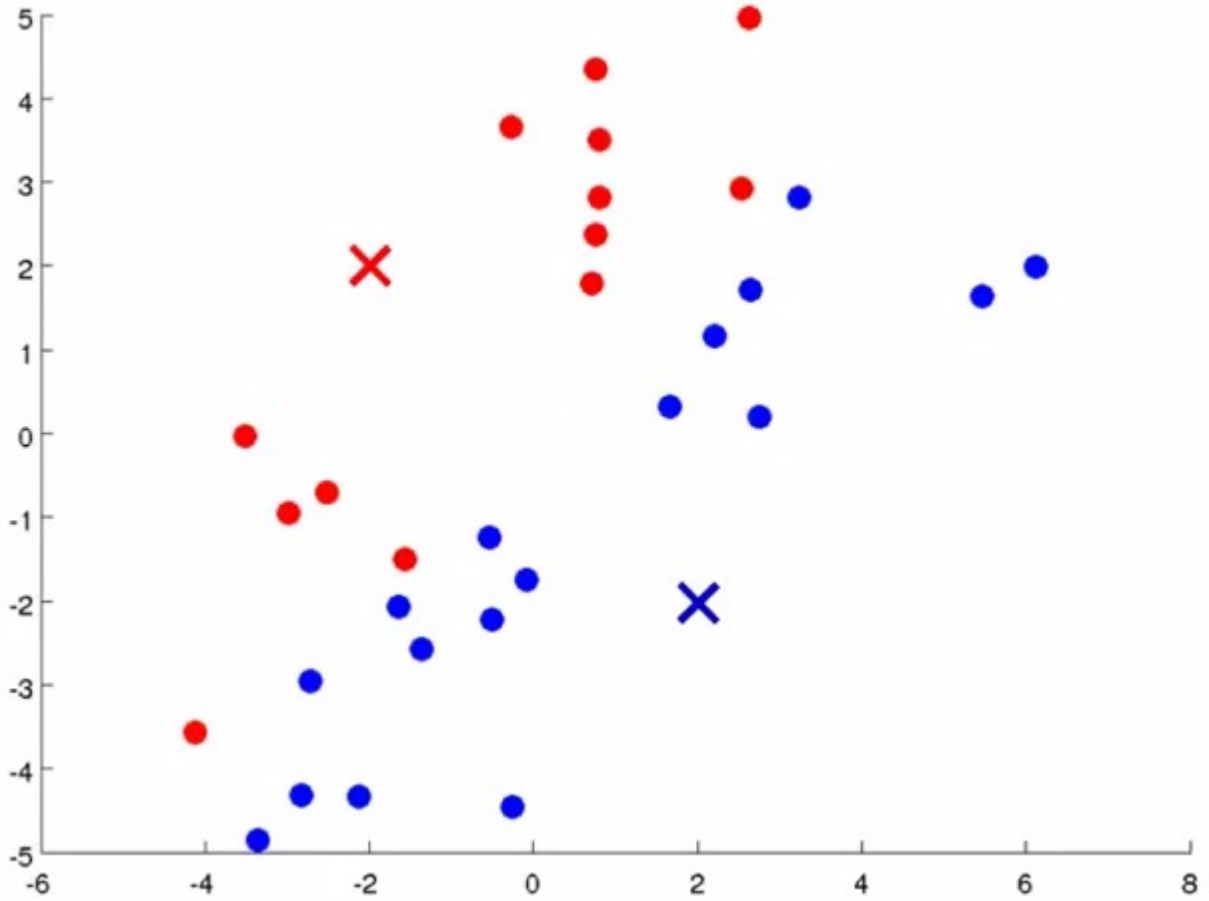


How it Works: The Initial State



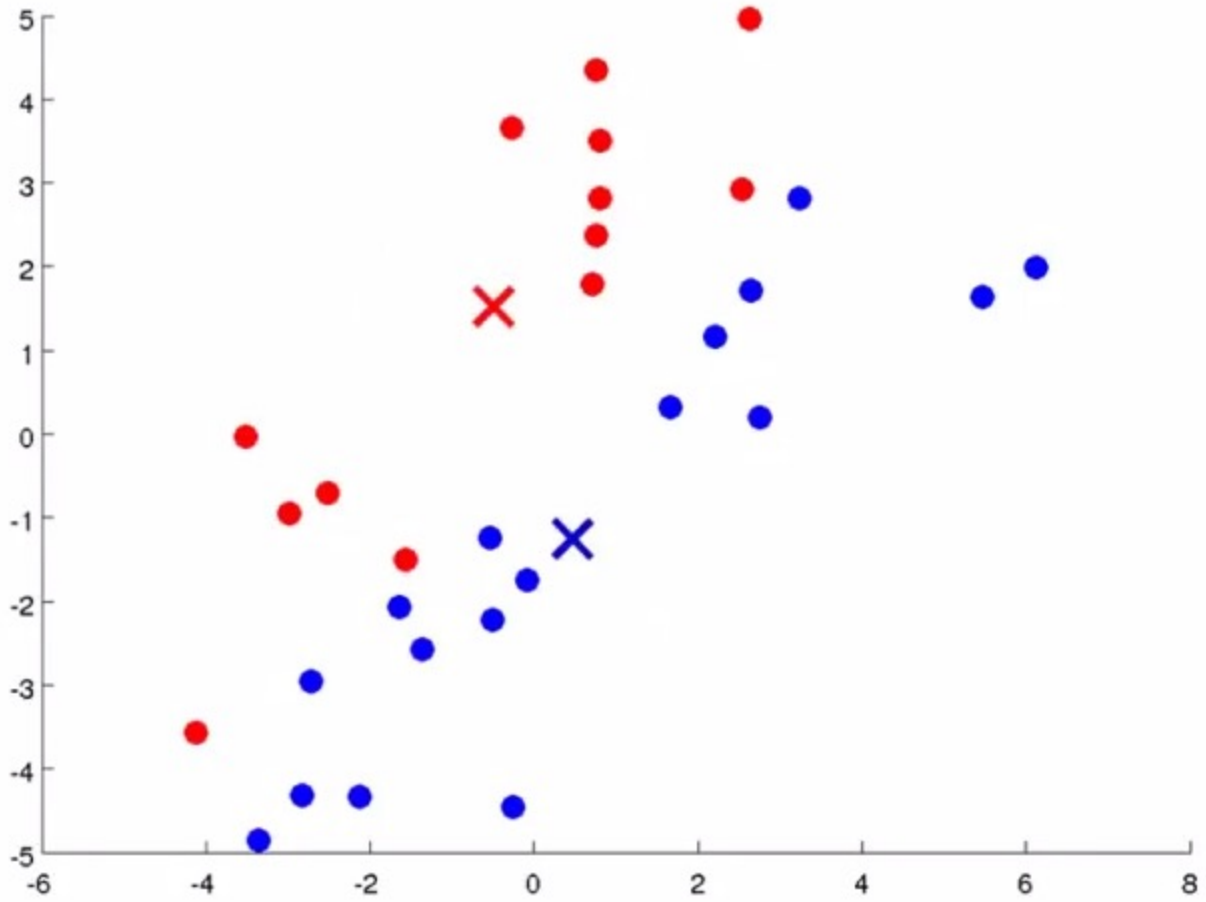
- *k* is 2
- Two initial centroid points chosen
  - Algorithm works no matter how they're chosen
  - *k r*andom locations, or *k* observations

Initial Assignment to Clusters



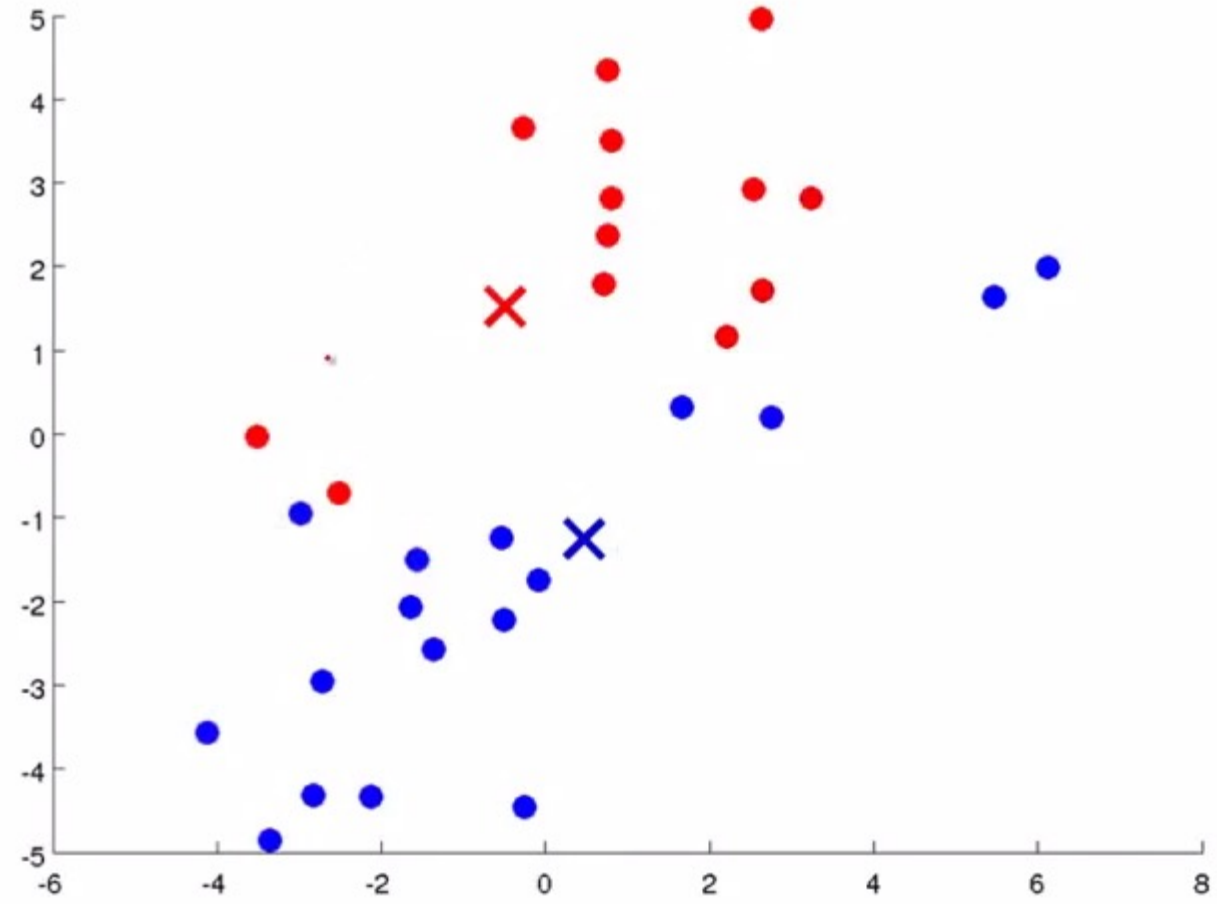
- Using initial centroids, observations assigned to clusters based on nearest centroid

Update Centroids



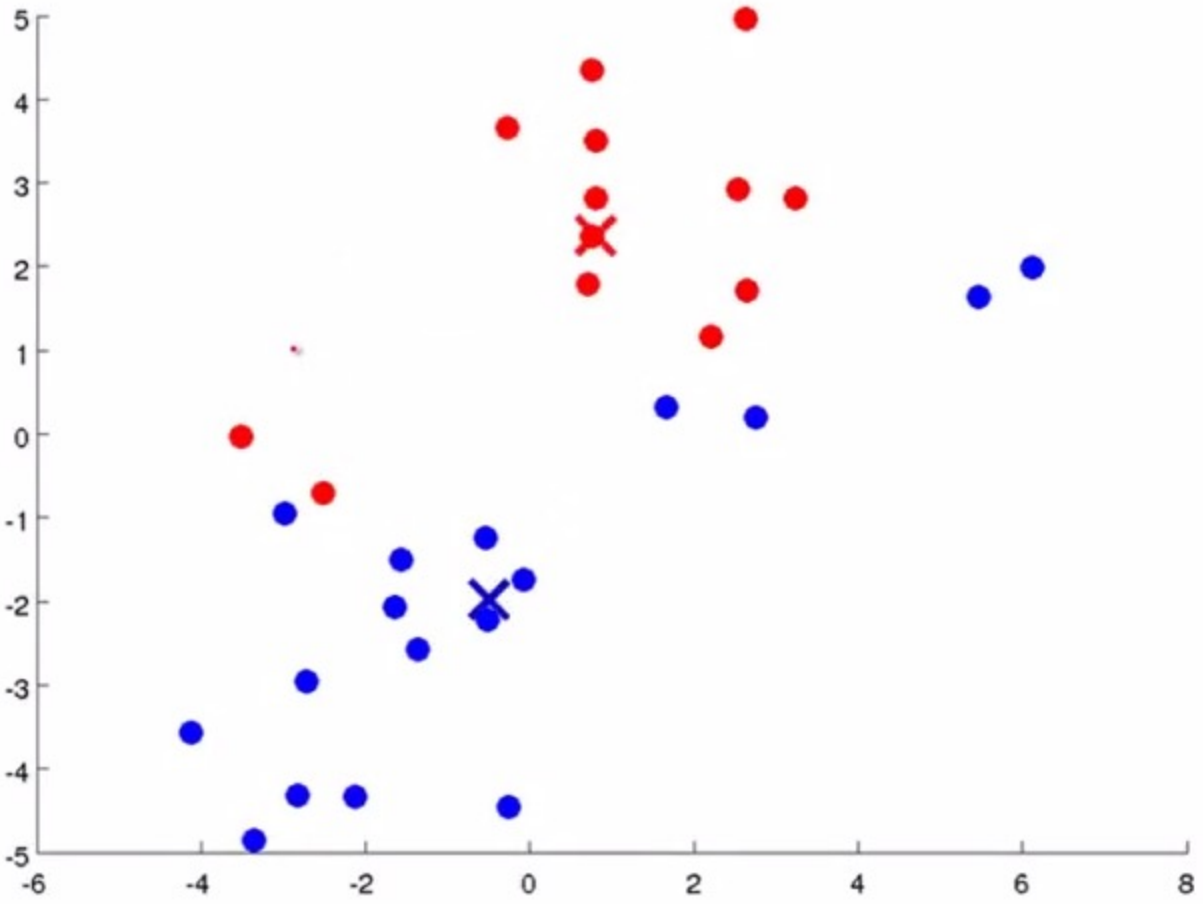
- After assignment, calculate new location for each centroid
  - Using all observations in a cluster, average each of the *n*-dimensional coordinates

Reassign Observations for Updated Centroids



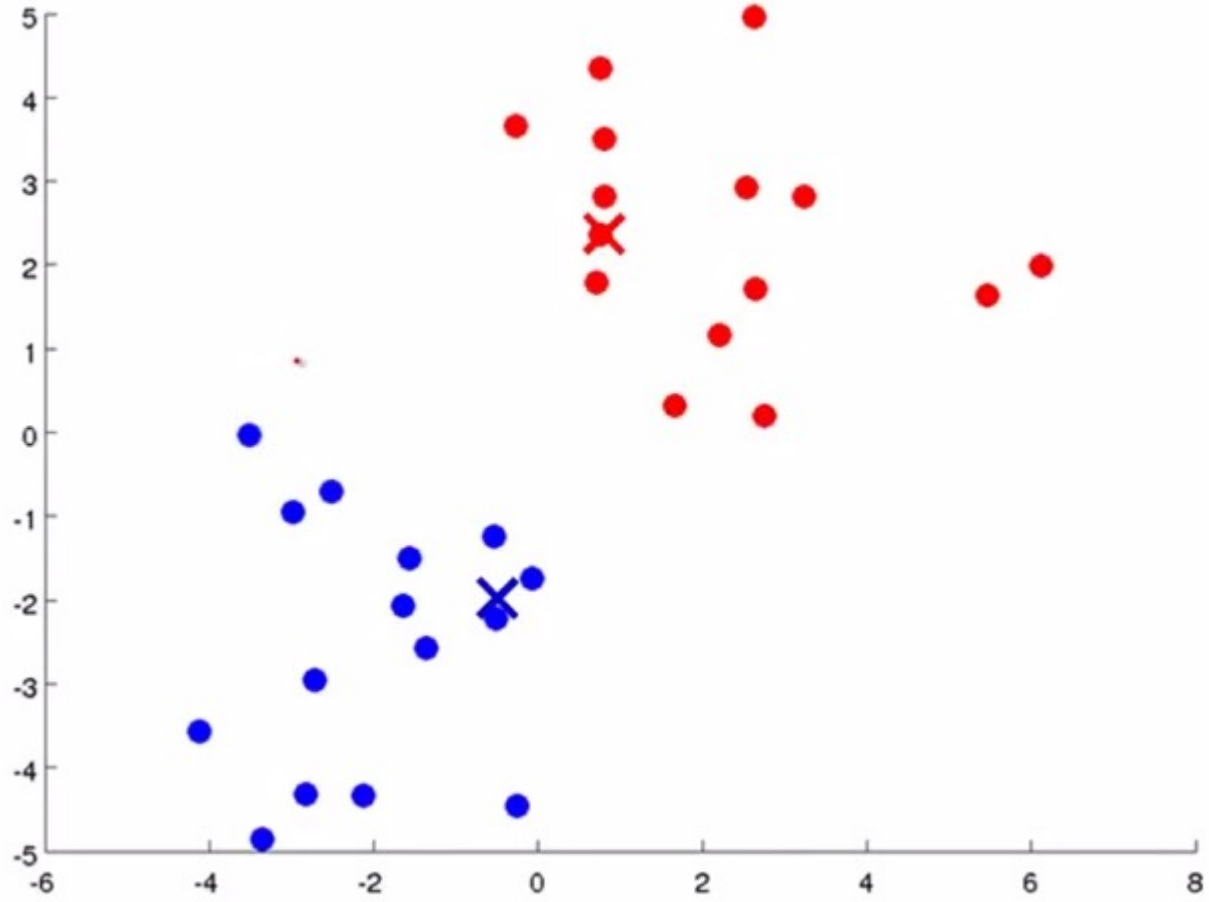
- Using the updated centroids, reassign each observation to a cluster based on nearest centroid
  - Comparison with previous slide:
    - Centroids are the same
    - Cluster assignments are updated

Update Centroids



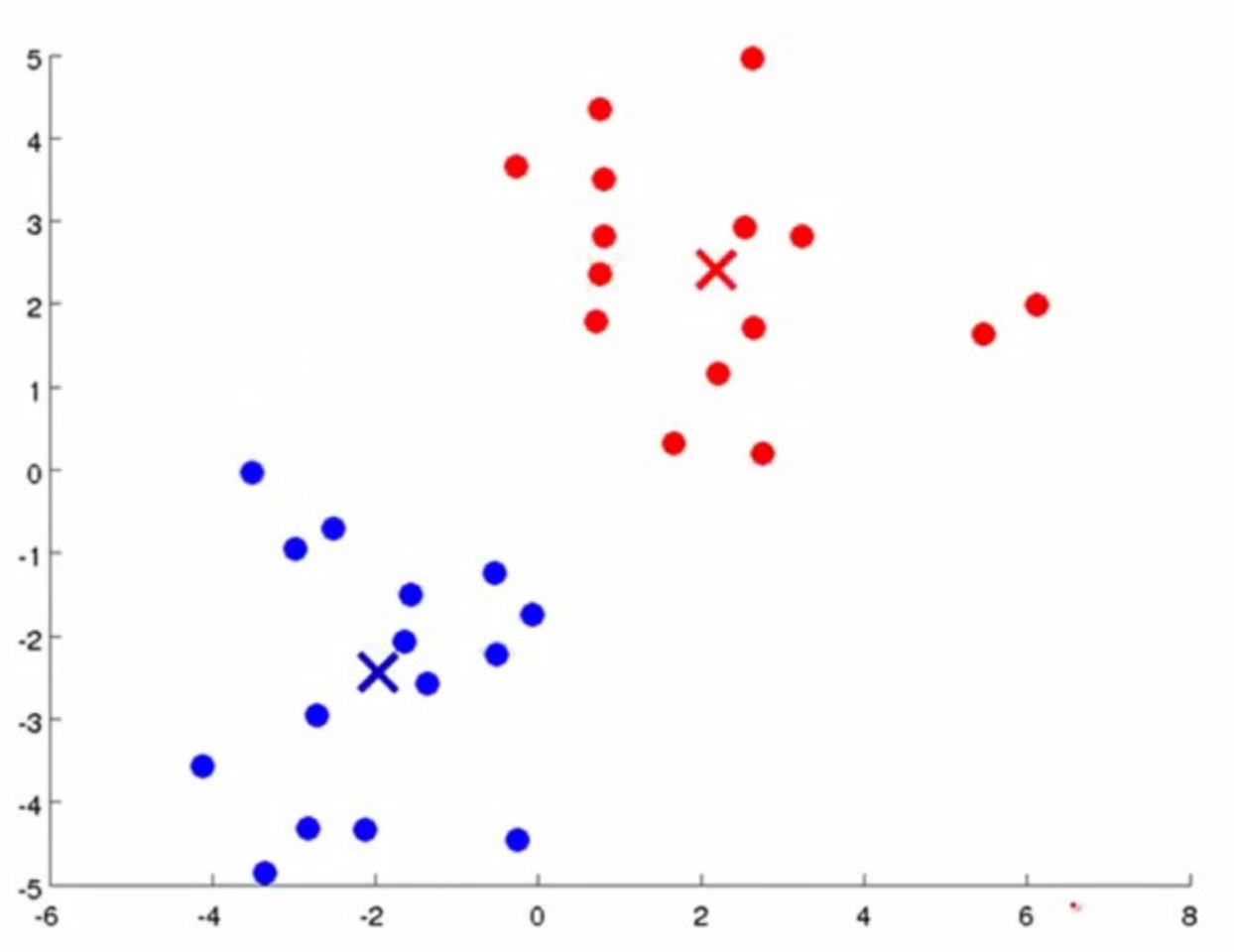
- Centroids updated based on reassignment done in last step

Reassign Observations for Updated Centroids



- Reassign observations using centroids from previous slide
  - Comparison with previous slide:
    - Centroids are the same
    - Cluster assignments are updated

Update Centroids, and We've Converged



- Centroids updated based on reassignment done in last step
- Then, observations are reassigned
  - But none of the assignments change from the last step
  - So the algorithm stops

Online Visualization

Explore how this works at
https://hckr.pl/k-means-visualization/

- You can add random points, but more interesting to add points manually in clusters (or close to clusters) to see how it handles these
- Specify how many clusters you want
- On right side, use two buttons ***Reassign data points*** and ***Update centroids' positions*** to step through algorithm
- Note:
  - To use same points but clear clustering and centroids, hit ***Remove All***
  - To clear the points when you want to start over, refresh page in your browser

Time Complexity

- Let's talk about time-complexity!
- What factors affect the time-complexity?
  - Obviously *n*, the number of observations
  - And what else?

Time Complexity

What factors affect the time-complexity?

1. Obviously *n*, the number of observations
2. *k*, the number of clusters
3. *i*, the number of iterations before we converge
4. *d*, the dimensionality of the observations

Think about when each of these affect the amount of work done. Can you say what part of the algorithms is affected by each of these?

Time Complexity

Updating the centroids

- for each of *O(n)* observations, calculate the average of each of *d* dimensions
- So this part alone is *O(nd)* each time we need to do it

Reassigning observations

- for each of *n* observations, calculate the distance to each of *k* centroids (and remember the smallest value)
- So this part alone is *O(ndk)* each time we need to do it

And we repeat all of this for *i* iterations!

Final Words on *k*-Means

Overall, the algorithm is *O(ndki)*

- This is pseudo-polynomial because of *k* (but often *k* is not large)

Wikipedia has a good article on this problem and this algorithm
https://en.wikipedia.org/wiki/K-means_clustering
See that for another illustration of it working, and for more about things like:

- Issues related to convergence
- Issues related to choosing initial centroids
- More details on time-complexity
- Details on improvements on the naïve algorithm shown here

Where Are We?

1. This overview, and how many ML algorithms work with data
2. Clustering as an example of unsupervised learning
   - Single-link clustering (like PA3)
   - Another algorithm: *k*-means clustering
3. A simple classification algorithm to show supervised learning
   - ***k*-nearest neighbor classification**
     - This algorithm only uses concepts you've learned already
   - Then a brief overview of other techniques
4. A brief intro to reinforcement learning

Supervised Learning and Classification

Supervised Learning and Classification

- Reminders
  - Supervised learning makes decisions based on a set of data observations for which we know some ”truth” about each observation
- Classification (one type of supervised learning problem)
  - We know there are 2 or more classes of observations
  - We have observations where we know the class each belongs to
    - This is called the **training set**
  - **Goal:** Build a model or algorithm that allows us to accurately predict which class a new unclassified data point belongs to

Earlier Example

- Data for patients at risk for a type of cancer
- Features might be:
  - Weight, body-mass index, cholesterol levels, gender, ethnicity, socioeconomic category, zipcode, level of alcohol consumption, …
- For observations in training set, we know whether or not the patient has had this cancer
  - Two classes: *had cancer* and *didn't have cancer*
- Goal: for a new patient with values for those features, predict which class they should be in
  - Perhaps with some level of probability

Another Example: Grade Prediction

Student	Earned an A in 2130?	Male?	Works Hard?	Drinks?	Will earn A in 3130?
Richard	Yes	Yes	No	Yes	No
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	No
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No
Alex	Yes	No	Yes	No	???

- Can we predict a student's grade in CS3130?
- Table shows a training set with 6 observations
  - Last column is label for the class
- What could we predict for Alex?
- Can we derive some rules from the training set?
  - If so, why might these rules work or not work on unclassified data?

Another Example: Grade Prediction

Student	Earned an A in 2130?	Male?	Works Hard?	Drinks?	Will earn A in 3130?
Richard	Yes	Yes	No	Yes	No
Allen	Yes	Yes	Yes	No	Yes
Alison	No	No	Yes	No	No
Jeff	No	Yes	No	Yes	No
Gail	Yes	No	Yes	Yes	Yes
Simon	No	Yes	Yes	Yes	No
Alex	Yes	No	Yes	No	**Yes**

- Some possible learned rules?
  - You will get an A in CS 3130 if:

    Earned an A in 2130 && Work Hard

    -- or --

    (Male && Don't Drink) || (Female && Drink)
- Will these hold up if given a new test set? Probably not…
  - Training set is very small
  - Our rules are ***biased*** on this data set and probably will not ***generalize***

Classification Process

- Get a good **training set**
  - Need a large number of observations
    - How large? As large as possible! ☺ Seriously, this is an important topic
- Develop an algorithm, perhaps based on some kind of model
- Evaluate your model with a **test set**
  - Another set of observations with known assignments to classes
  - Not used during training, not used to develop your model
    - Prevents **bias** or **overfitting your model** to the training set
  - Accuracy evaluated by percentage of test set that's correctly classified
- Refine your model or algorithm, and re-test
- Finally, apply to unclassified observations to make predictions

Classification Algorithms

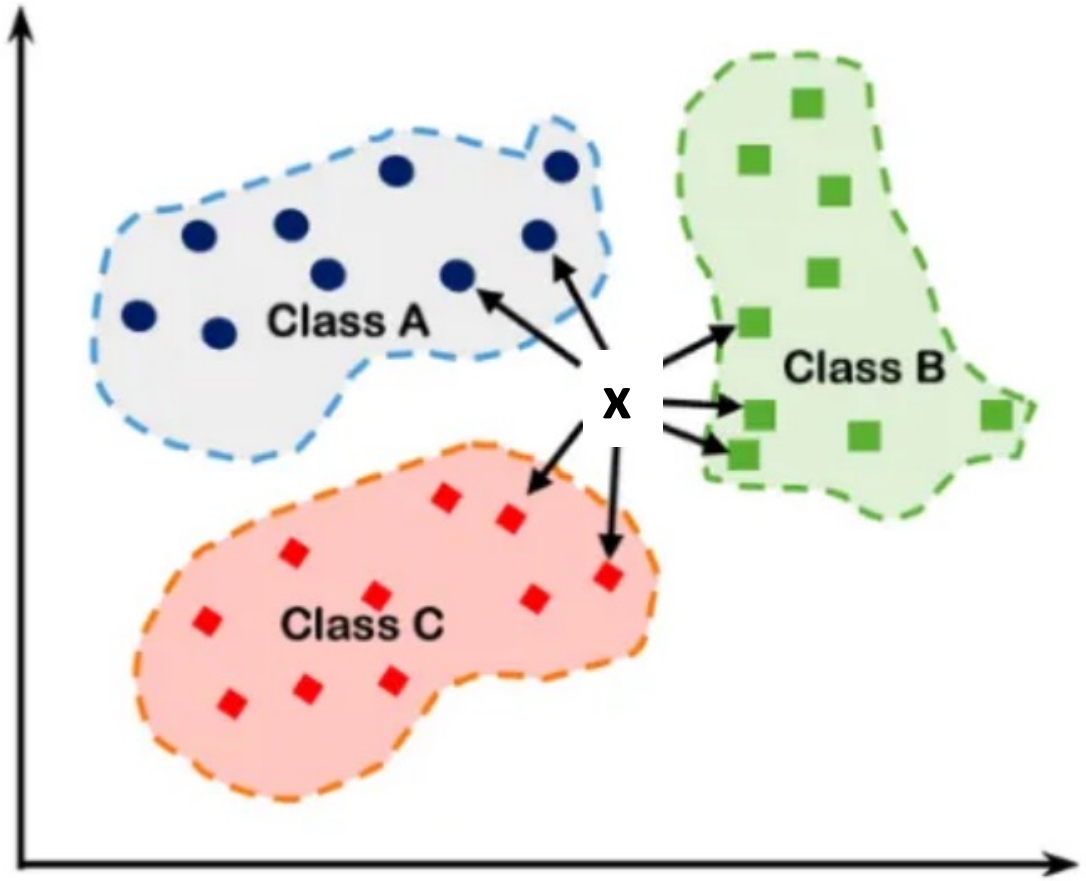
- The process described above can be used with many algorithms or approaches to classification
  - Linear regression, Support Vector Machines (SVM), Decision Trees, Regression approaches, Probability Estimation, Neural Networks, …
  - Some of these depend on certain statistical distributions of the data
  - Overview: https://en.wikipedia.org/wiki/Statistical_classification
- We'll look at one very simple approach (and leave the others for your future study of ML):

  k-nearest neighbor classification

k-Nearest Neighbors

- The classified observations in the training set are treated as points in an *n*-dimensional space
- A value of *k* is chosen
  - Often an odd number for two classes
  - Could be 1 (a nearest-neighbor classifier)
  - Note that here *k* will mean something different than in *k*-means clustering
- Find the *k* observations in the training set that are closest to an unclassified observation x
- The class of those *k* observations act like a vote used to assign x to a class
  - Say k=5, and 3 neighbors are Class A, and 2 are Class B. x is assigned to Class A.

k-Nearest Neighbors Example



- Here *k*=7, and there are 3 classes
  - Nicely separated! The real world is often messier!
- Unclassified observation x is assigned to Class B
  - 3 nearest neighbors for B, and only 2 for Class A and 2 for Class C
- If it was a tie vote? Then we can't classify this observation
  - For two classes, use an odd *k*

Image from: https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2

k-Nearest Neighbors Details

- First, if the data has certain properties, this approach works surprisingly well!
- Issues
  - If size of the classes represented in the test set are very different, ”majority wins” gives a skewed result
    - Can address this with some kind of weighting (of votes, or of distances)
  - Choice of *k*: is bigger better?
    - One option: try different values on your test set and choose which gives best results on that data
  - Too many features and curse of dimensionality
    - **Feature extraction** is one technique to address this
    - Again, you can evaluate possible improvements using the test set
  - Note that *k*-NN does not depend on any statistical distribution of the data
    - Being “non-parametric” is an advantage for some problems
- Many good write-ups on the web about *k*-NN including Wikipedia

Implementation and Time Complexity

- To classify an observation x for a training set of size *n*
  1. Must calculate distance from x to all *n* items: *O(nd)* overall
  2. Find the *k* smallest of these
  3. Count the "votes" for those k values: *O(k)*
- For step 2 above, sort the *n* distances and choose smallest *k*?
  - You know better than that! We can do better than *O(n log n)* to solve this! How? What data structure?

Implementation and Time Complexity

- To classify an observation x for a training set of size *n*
  1. Must calculate distance from x to all *n* items: *O(nd)* overall
  2. Find the *k* smallest of these
  3. Count the "votes" for those *k* values: *O(k)*
- For step 2 above, as we find distances, we need to keep track of the *k* smallest
  - Let's just store *k* items, the smallest seen so far
  - To update, if the next distance calculated is good enough, it should replace one of the *k* items we've stored
  - What item? What data structure to use?
  - **Answer:** if the next distance is smaller than the largest of the *k* items, replace it! What data structure is good for this?
  - **Answer:** a max-heap! If we had sorted, we'd have done *O(n log n).* Now we're doing *O(n log k)* where $k \ll n$

Reinforcement Learning (RL)

Reinforcement Learning Definitions

Reinforcement learning *is learning what to do — how to map situations to actions — so as to maximize a numerical reward signal.*

- Sutton & Barto, 2018

Reinforcement learning *(**RL**) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.*

- Wikipedia

https://en.wikipedia.org/wiki/Reinforcement_learning

Reinforcement learning

- RL can address a family of problems that involve **sequential decision making**

Game playing

Self-driving car

Conversational System

Key Concepts of RL

- Main characters in a RL problem: the **agent** and the **environment**
  - Agent lives in and interacts with the environment
- At each step of an interaction
  - Agent sees the state of the world (perhaps partial)
  - Agent decides what action to take from a set of possible actions
  - Environment may change as a result of an action (or on its own)

- Agent gets a reward signal from the environment
- Agent's goal is to maximize ***return***, the cumulative reward

- **RL methods are about how agents can learn behavior to achieve this goal**

A view of agent and environment interaction

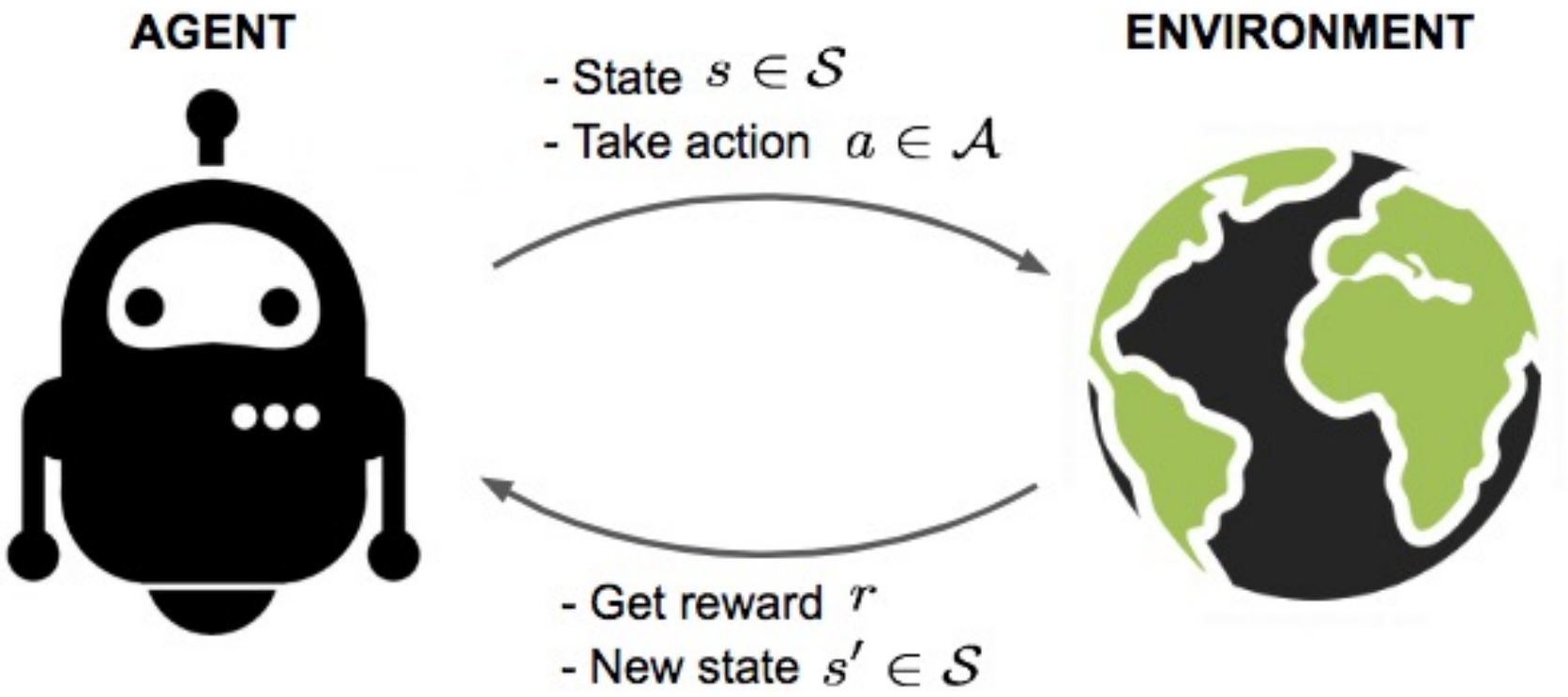


Image credit: Lil'Log

Reinforcement Learning: Example

- Suppose we want an agent that plays Mario
- A concept of reward
  - Going to the right = reward (maximize this)
- A set of actions an agent can take
  - Buttons on the controller (jump, etc.)
- A set of ways to observe the environment

Want to Explore More?

- PyTorch, a popular library for deep learning
- Tutorial: *Training a Mario-playing RL Agent* https://pytorch.org/tutorials/intermediate/mario_rl_tutorial.html
  - Learn more about RL from link near top on RL *Concepts*
  - The tutorial "walks you through the fundamentals of Deep Reinforcement Learning. At the end, you will implement an AI-powered Mario (using Double Deep Q-Networks) that can play the game by itself"

UVA CS Department and ML/AI

ML/AI and UVA CS Faculty

Faculty with Primary Area in ML/AI

Faculty Name	Primary Area	Secondary Area
Doryab, Afsaneh	ML/AI	HCI
Farnoud, Farzad	ML/AI	Theory/Alg
Fioretto, Nando	ML/AI	Theory/Alg
Fletcher, Tom	ML/AI	Graphics
Ji, Yangfeng	ML/AI	
Li, Jundong	ML/AI	
Meng, Yu	ML/AI	
Qi, Jane	ML/AI	
Vullikanti, Anil	ML/AI	Theory/Alg
Wei, Chen-Yu	ML/AI	Theory/Alg
Zhang, Aidong	ML/AI	
Zhang, Miaomiao	ML/AI	Image/Vision
Zhang, Shangtong	ML/AI	

Faculty with Secondary Area in ML/AI

Faculty Name	Primary Area	Secondary Area
Behl, Madhur	CPS	ML/AI
Feng, Lu	CPS	ML/AI
Iqbal, Tariq	CPS	ML/AI
Kuo, Yen-Ling	CPS	ML/AI
Cheng, Zezhou	Graphics	ML/AI
Basit, Nada	CS Education	ML/AI
Nguyen, Rich	CS Education	ML/AI

Note: CPS is Cyberphysical Systems

CS Courses in ML/AI (in last 2 years)

- CS 4710 Artificial Intelligence
- CS 4774 Machine Learning
- CS 4501 Optimization
- CS 4501 Machine Learning in Image Analysis
- CS 4501 Introduction to Reinforcement Learning
- CS 4501 Statistical Learning and Graphical Models
- CS 6316 Machine Learning
- CS 6666 Data Mining - Principles and Algorithms
- CS 6501 Machine Learning in Image Analysis
- CS 6501 Learning for Interactive Robots
- CS 6501 Reinforcement Learning
- CS 6501 Natural Language Processing
- CS 6501 Risks and Benefits of Generative AI and LLMs
- CS 6501 Learning in Robotics
- CS 6501 Probabilistic Machine Learning
- CS 6501 Computational Behavior Modeling