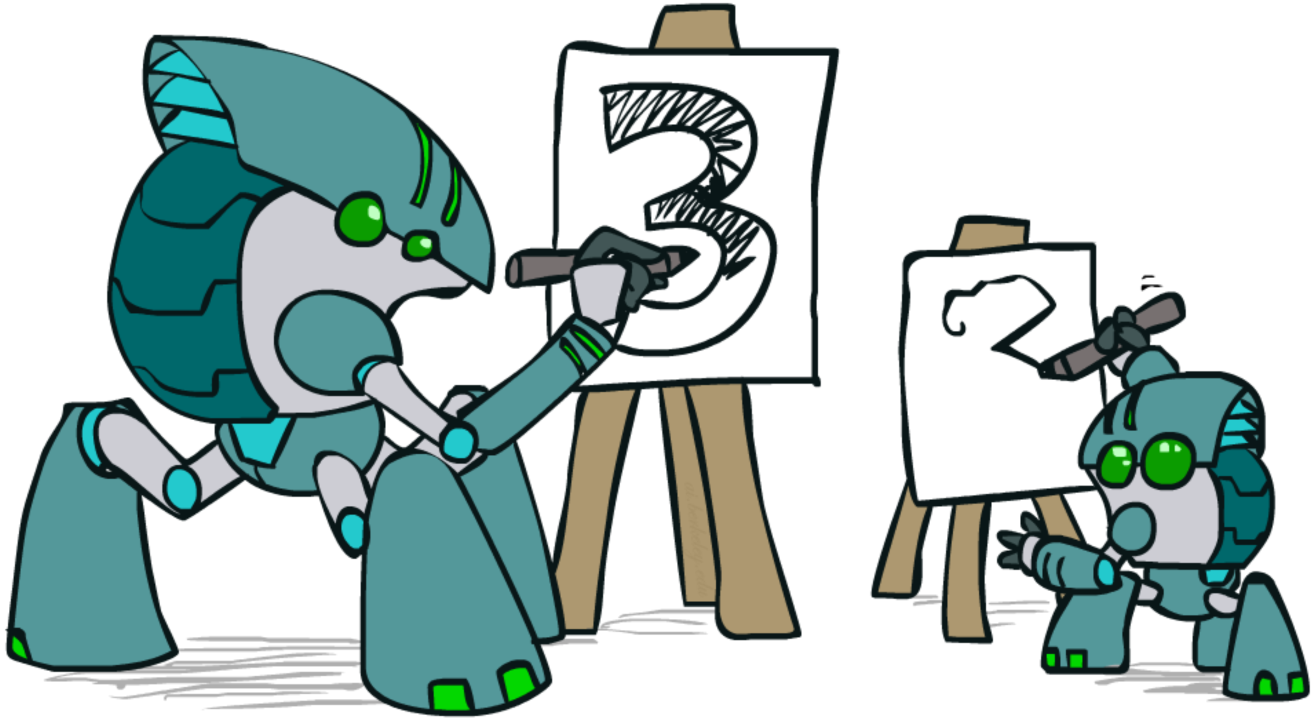
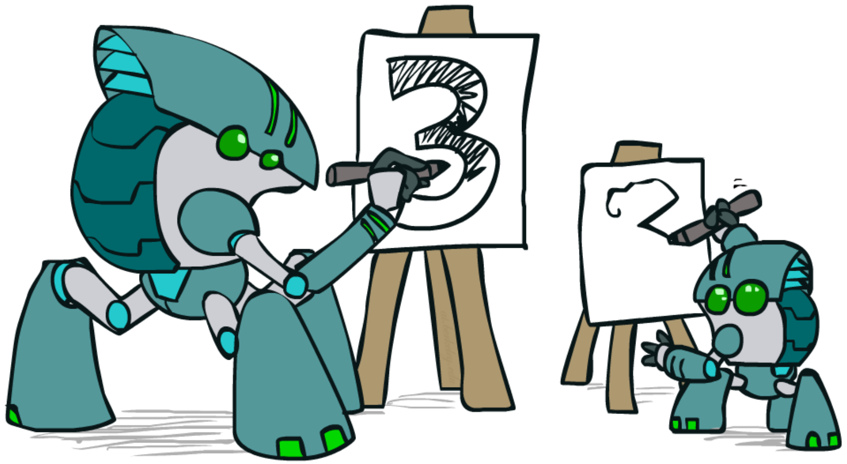
Apprenticeship Learning
Nearest Neighbor, Clustering



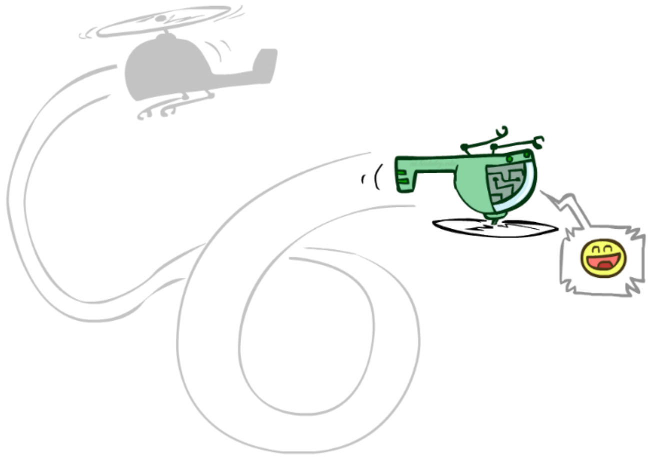
These slides are based on the slides created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley - http://ai.berkeley.edu.

Apprenticeship Learning



Apprenticeship learning is useful when we are dealing with dynamic and complex scenarios where there is no obvious reward function/goal.

Example: autonomous helicopter performing acrobatic maneuvers (flips, rolls, loops).

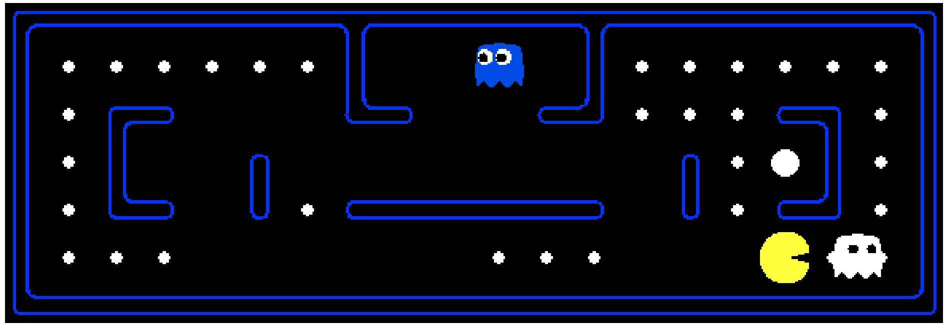


Call in the expert to perform the task and learn!

We can use the perceptron classifier for apprenticeship learning.

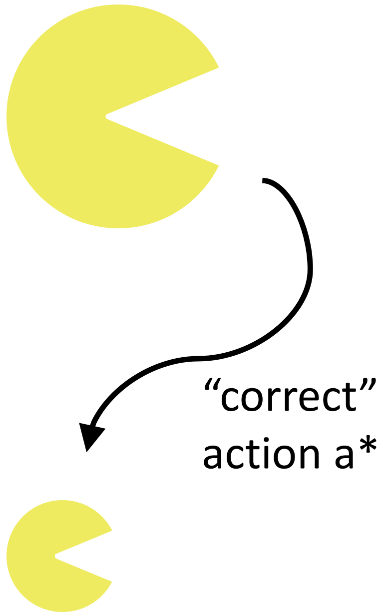
Pacman Apprenticeship

- Examples are game states s



- Labels are actions
- "Correct" actions: those taken by expert
- Features defined over (s,a) pairs: f(s,a)
- Score of a state-action combination (s,a) given by:

$$w \cdot f(s, a)$$



$$\forall a \neq a^*,$$
$$w \cdot f(a^*) > w \cdot f(a)$$

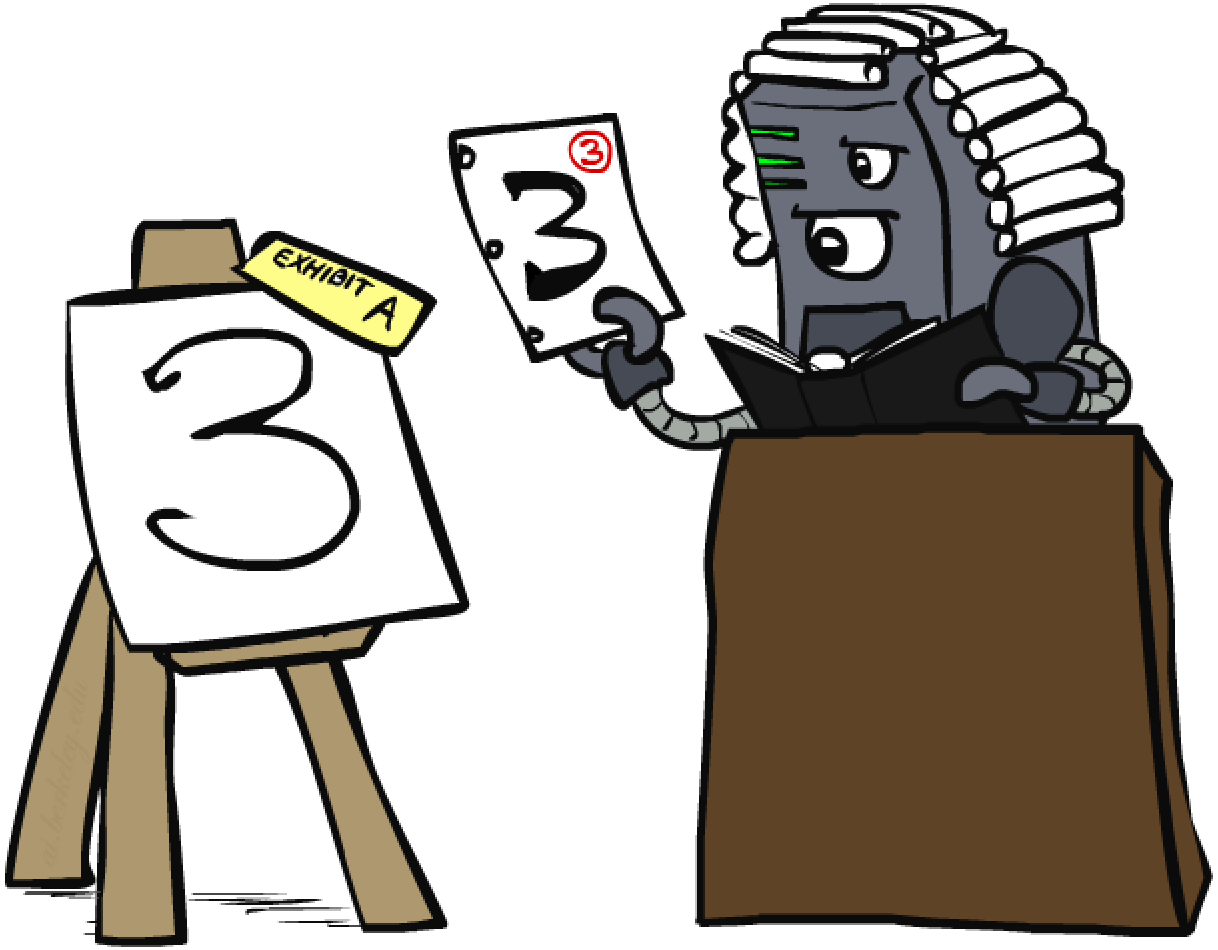
Pacman Apprenticeship

- States and actions of an expert Pacman playing the game have been recorded and saved in a pickle file (.pkl).
- The file is read and the features are extracted for you. We are only using the foodCount in this assignment but you can experiment with other features.
- The data (trainingData and validationData) contains a list of (features, legal moves) tuples.
- You don't have to use the validationData parameter in the *train* method in this assignment. Typically you would use it for tuning your parameters (number of iterations).

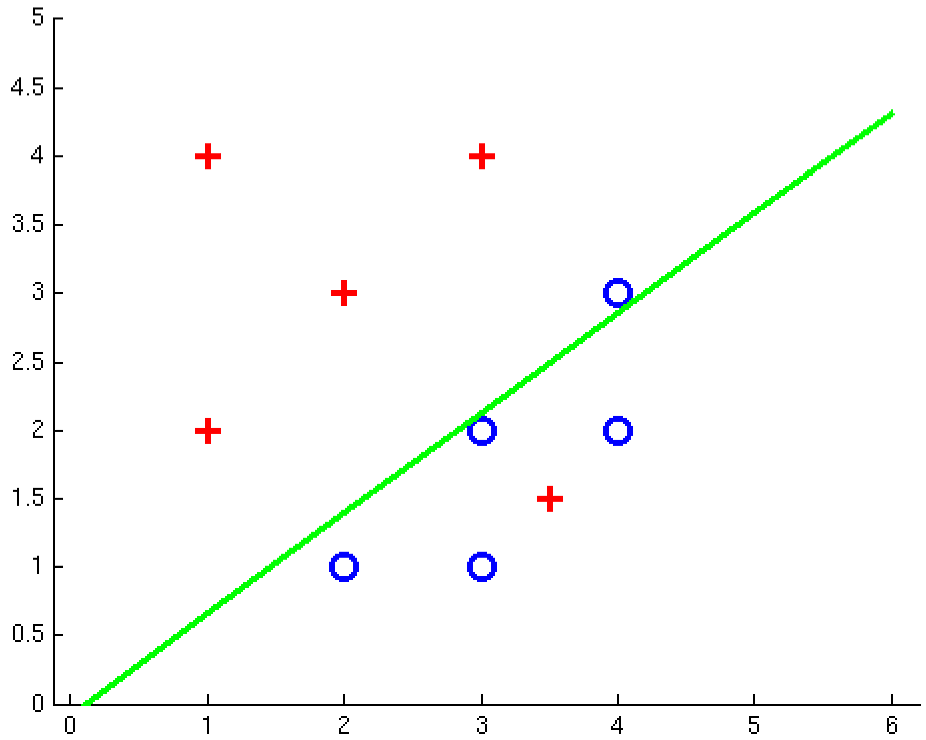
Pacman Apprenticeship!

- *features* represents the features corresponding to one state.
- There is a feature vector for each action from that state
  You can print *features* to see what it looks like.
- Here's an example:
  {'West': {'foodCount': 96}, 'East': {'foodCount': 96}, 'Stop': {'foodCount': 97}}
- To access the feature vector corresponding to the action 'East': we write: features['East'] – that is f(s, 'East‘)
- In general, for a given example state s, features[action] is f(s, action)
- Look in util.py for the Counter class and use its methods!

Case-Based Learning



Non-Separable Data



Case-Based Reasoning

- Classification from similarity
  - Case-based reasoning
  - Predict an instance's label using similar instances

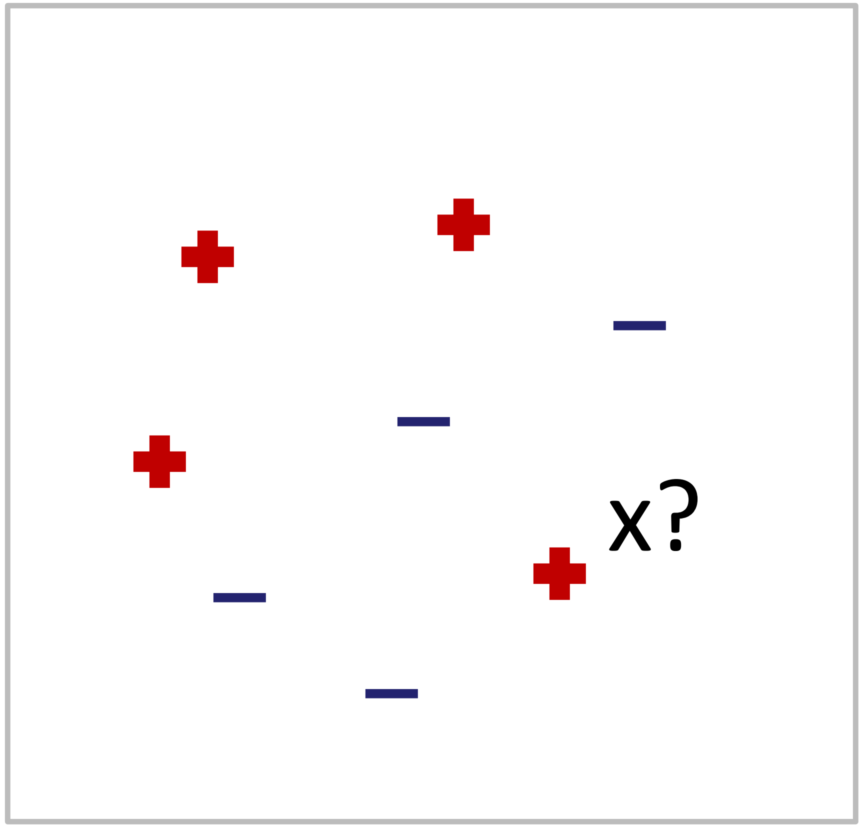
Nearest Neighbor Classification

- Nearest-neighbor classification
  - 1-NN: copy the label of the closest data point
  - Closest: most similar
  - How do we determine similarity?

  - What would the label of X be, using 1-NN?

A. +

B. -



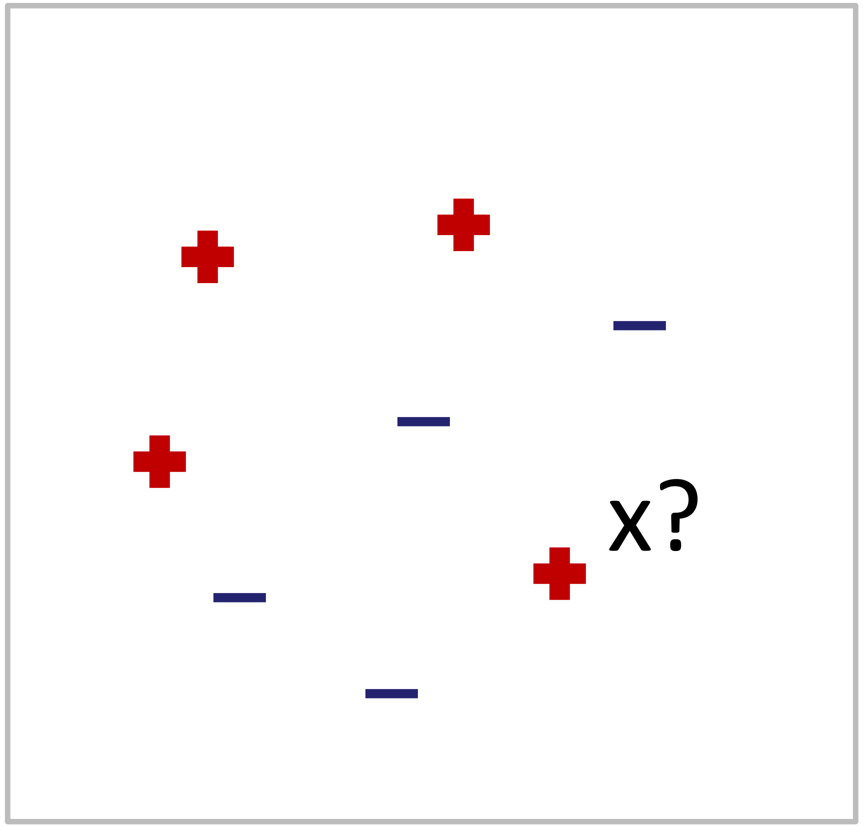
Nearest Neighbor Classification

- Nearest-neighbor classification
  - K-NN: vote the k nearest neighbors
  - Large k results in smoother classification

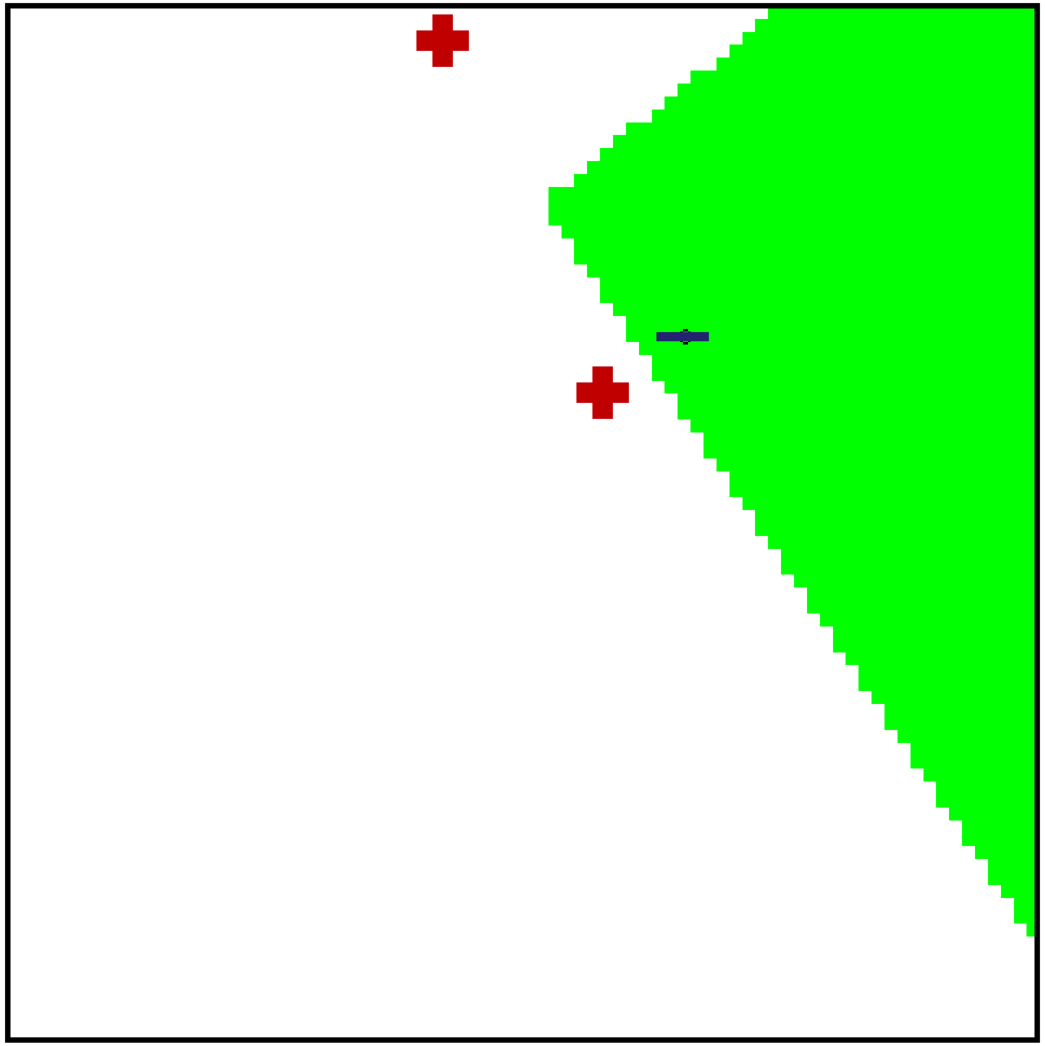
  - What would the label of X be, using 3-NN?

A. +

B. -

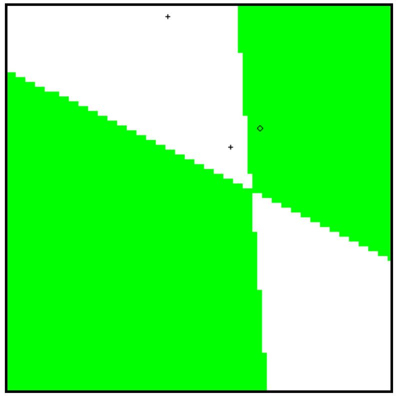


Nearest Neighbor Classification



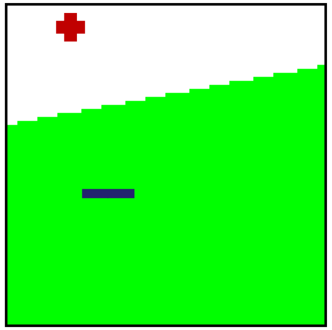
Parametric / Non-Parametric

- Parametric models:
  - Fixed set of parameters
  - More data means better settings
- Non-parametric models:
  - Complexity of the classifier increases with data
- (K)NN is non-parametric

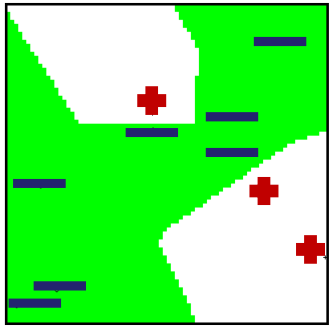


Truth

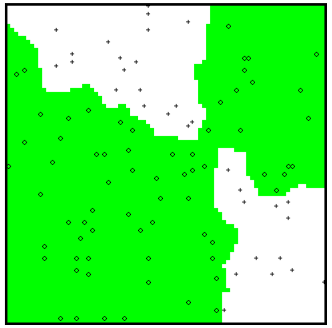
2 Examples



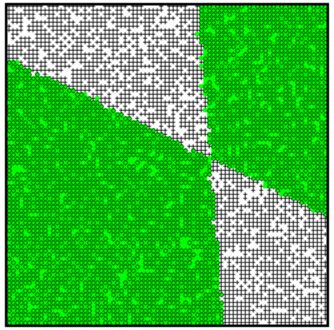
10 Examples



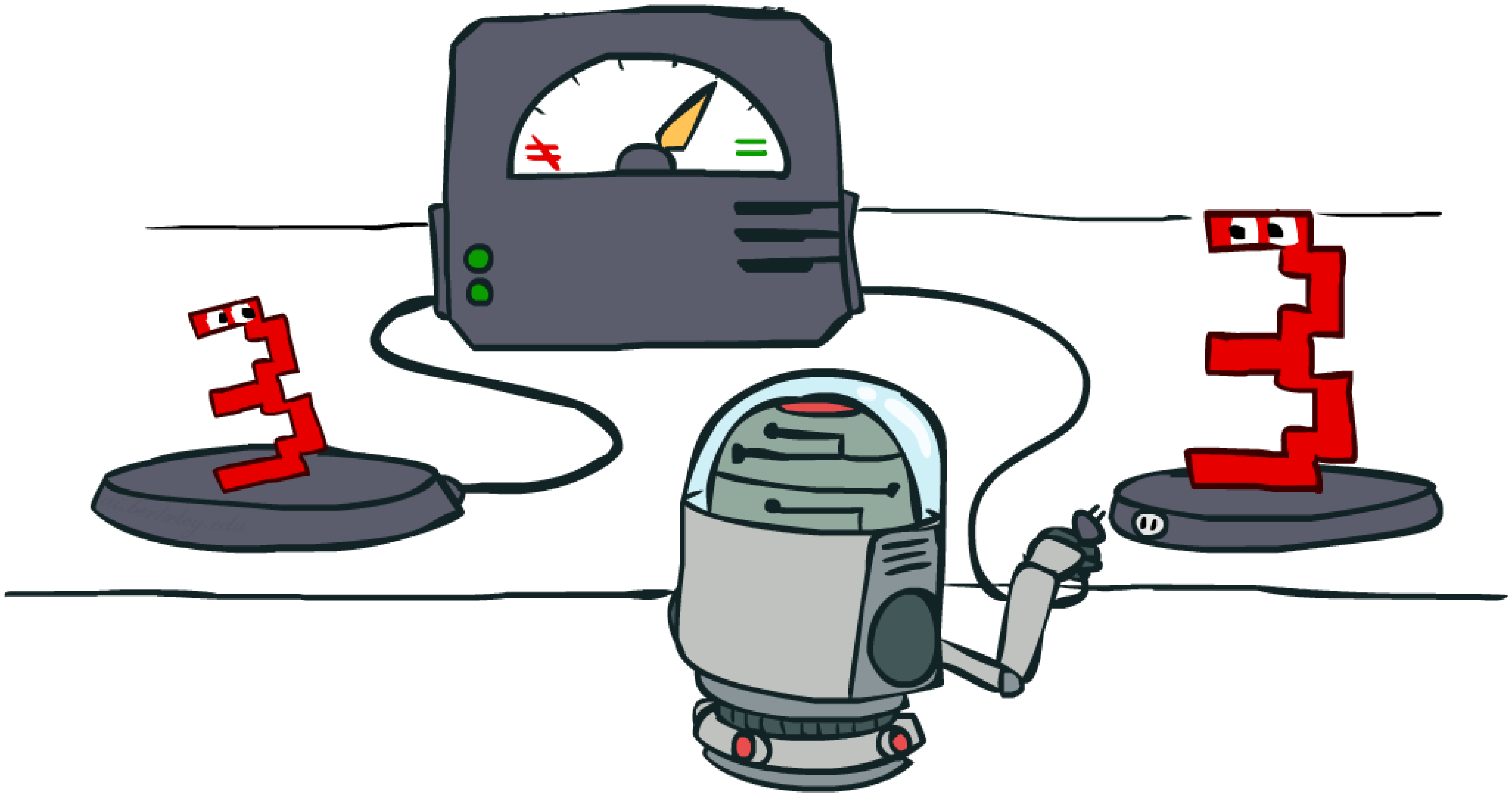
100 Examples



10000 Examples

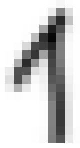


Similarity Functions



Nearest-Neighbor Classification

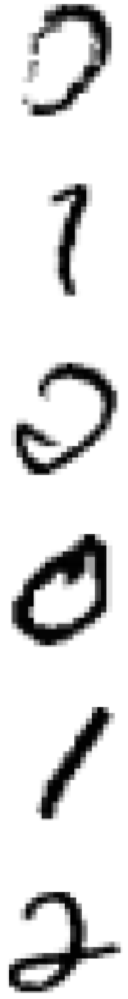
- Nearest neighbor for digits:
  - Take new image
  - Compare to all training images
  - Assign based on closest example



- Encoding: image is vector of intensities:

 $= \langle 0.0 \;\; 0.0 \;\; 0.3 \;\; 0.8 \;\; 0.7 \;\; 0.1 \ldots 0.0 \rangle$

- What's the similarity function?



0

1

2

0

1

2

Basic Similarity

- Many similarities based on feature dot products:

$$\text{sim}(x, x') = f(x) \cdot f(x') = \sum_i f_i(x) f_i(x')$$

- If features are just the pixels (we're looking for the pixels in common):

$$\text{sim}(x, x') = x \cdot x' = \sum_i x_i x'_i$$

- Note: not all similarities are of this form

Invariant Metrics

- Better similarity functions use knowledge about vision
- Example: invariant metrics:
  - Similarities are invariant under certain transformations
  - Rotation, scaling, translation, stroke-thickness…
  - E.g:

  - How can we incorporate such invariances into our similarities?

Quiz

Which classification is faster?

A. Perceptron

B. Nearest neighbor

Quiz

Which classification is faster?

A. Perceptron
B. Nearest neighbor

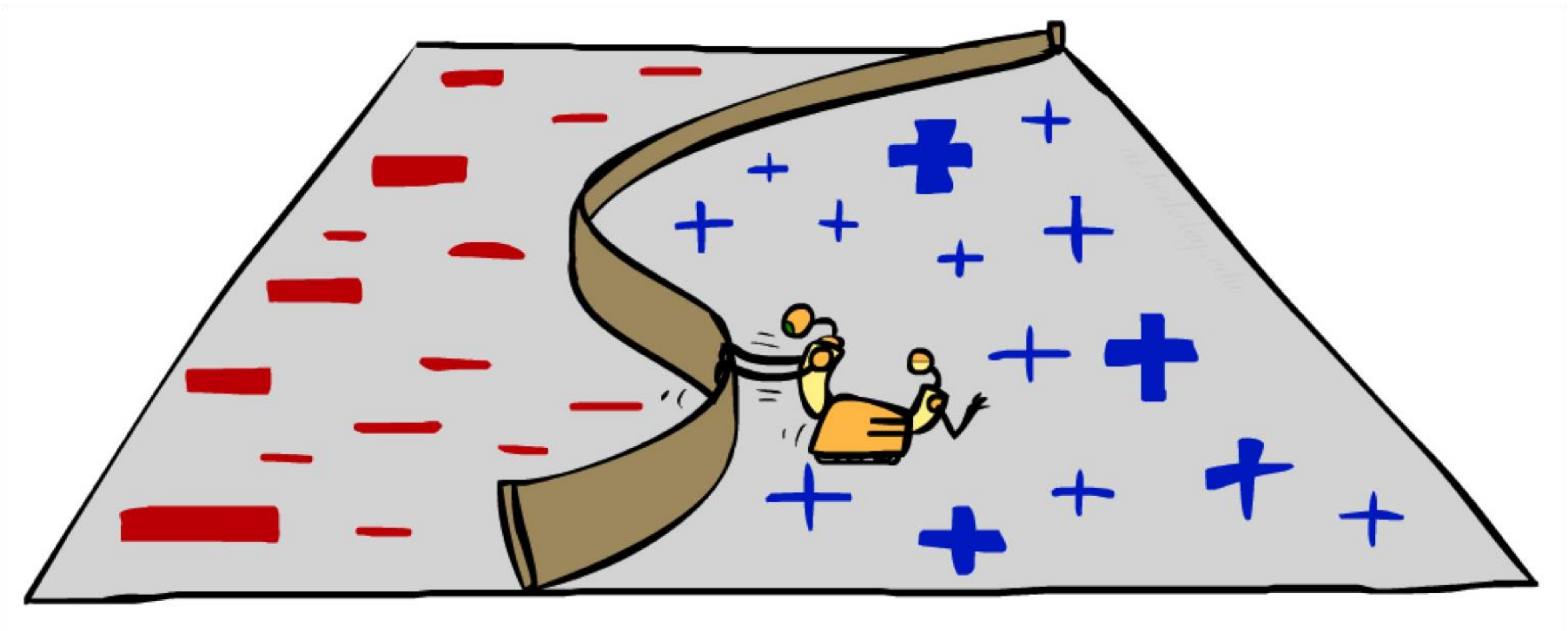
Lengthy searches for nearest neighbors. We have to search through the whole training data set.

A Tale of Two Approaches…

- Nearest neighbor-like approaches
  - Can use fancy similarity functions
  - Don’t actually get to do explicit learning

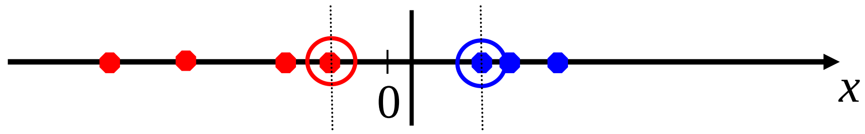
- Perceptron-like approaches
  - Explicit training to reduce empirical error
  - Usually linear classification
  - Faster classification

Non-Linearity

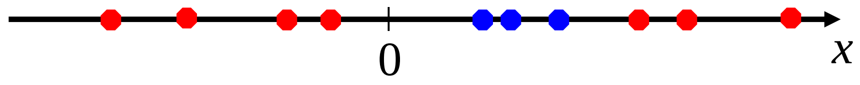


Non-Linear Separators

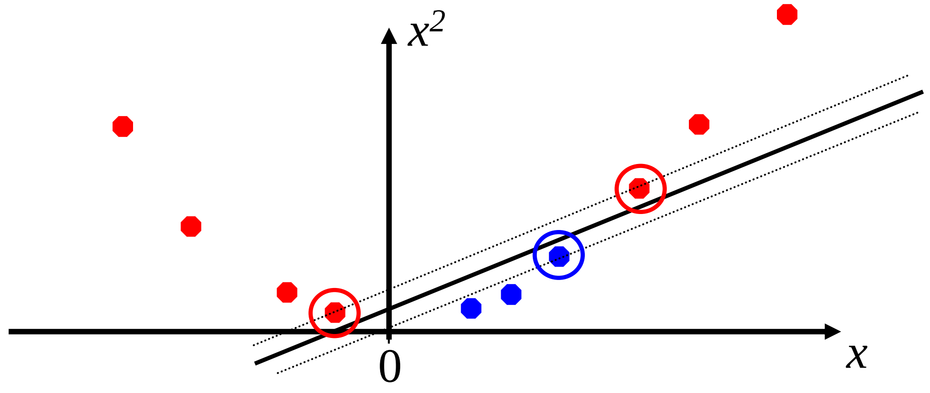
- Data that is linearly separable works out great for linear decision rules:



- But what are we going to do if the dataset is just too hard?



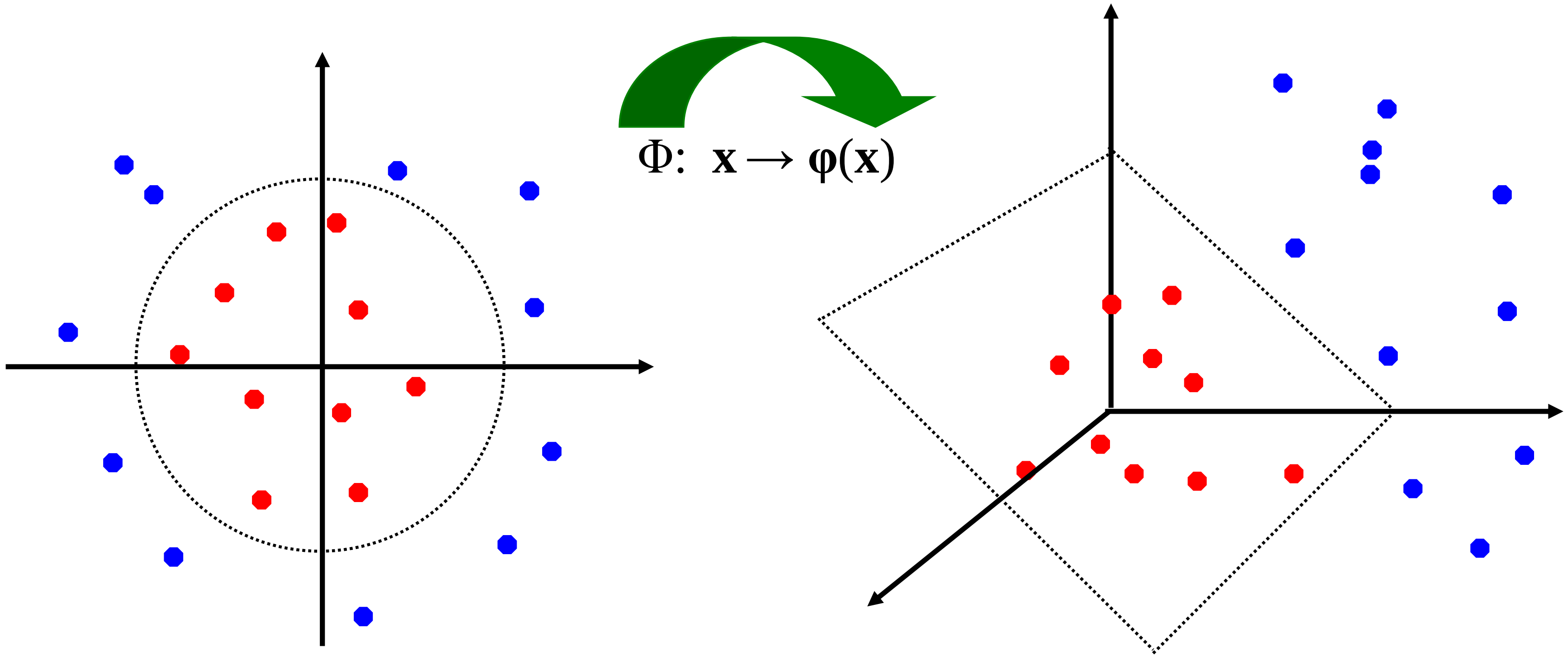
- How about… mapping data to a higher-dimensional space:



This and next few slides adapted from Ray Mooney, UT

Non-Linear Separators

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



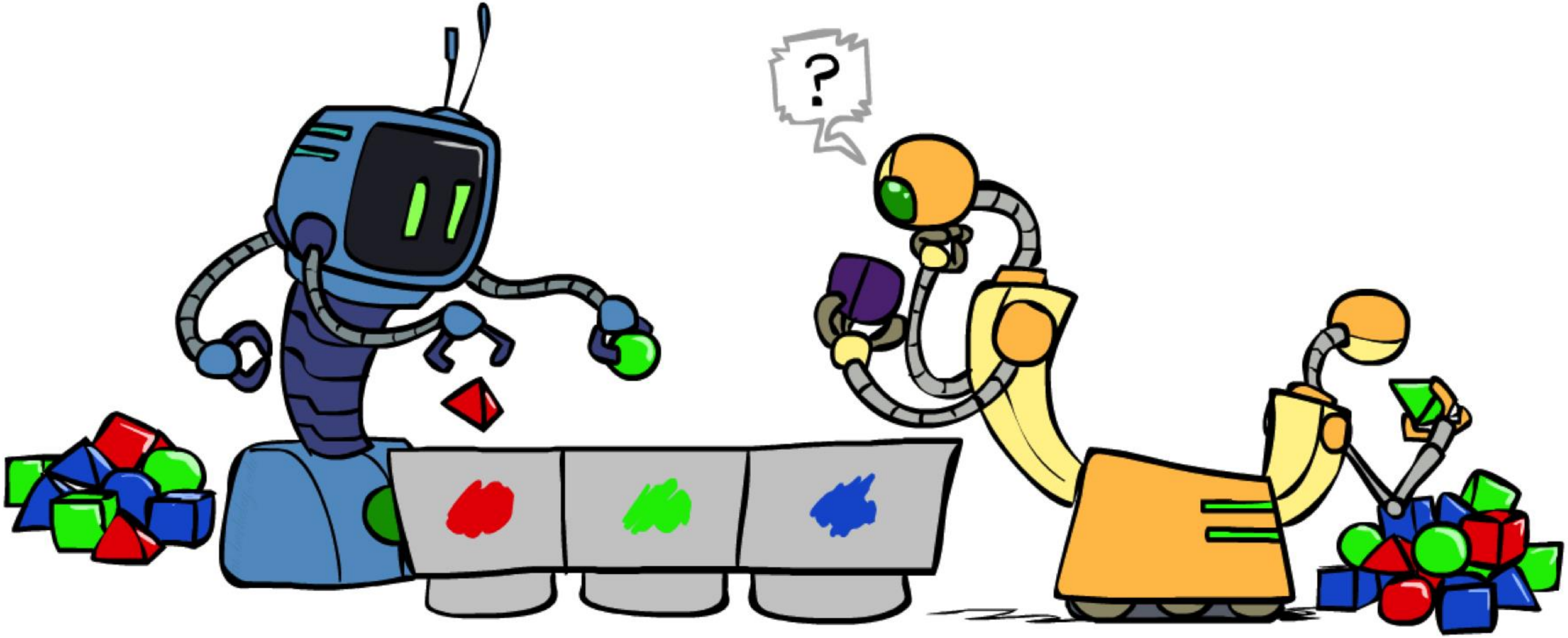
Recap: Classification

- Supervised learning
- Make a prediction given evidence
- We've seen several methods for this
- Useful when we have labeled data

Clustering

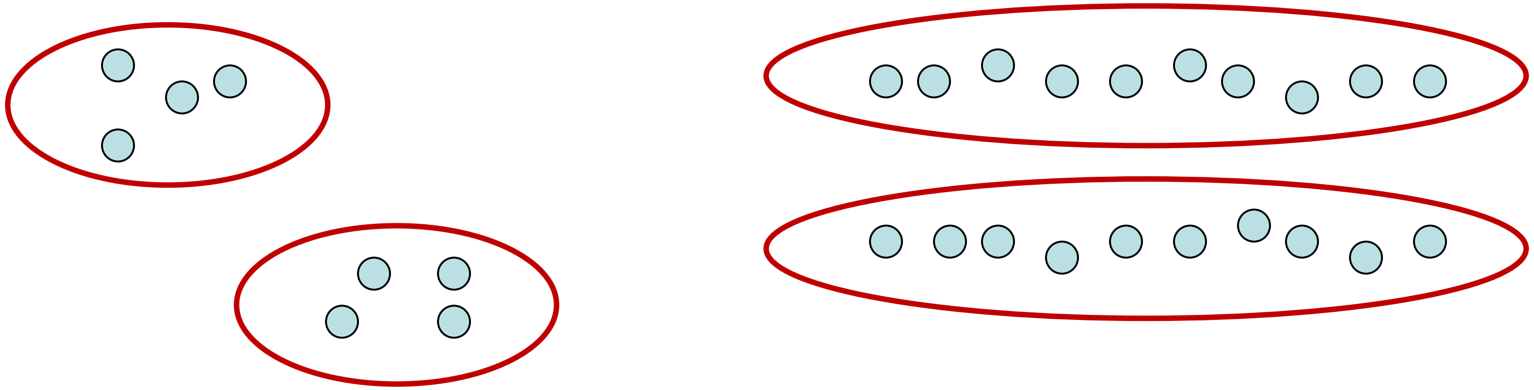
- Unsupervised learning
- Detect patterns in unlabeled data
  - group emails, search results, images
  - find categories of customers
  - detect anomalous/malicious program executions
  - blind source separation
- Useful when we don't know what we're looking for
- Requires data, but no labels
- Often get gibberish

Clustering



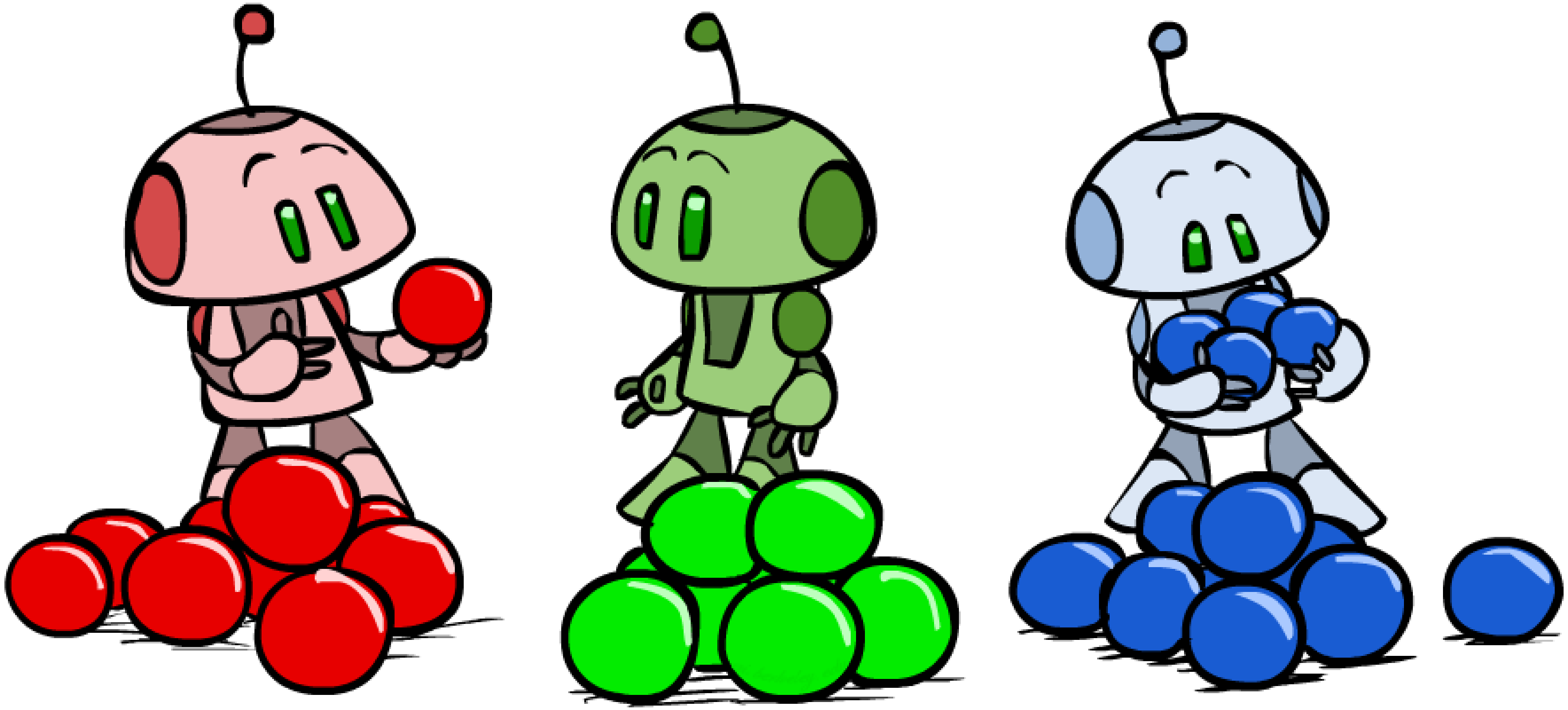
Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



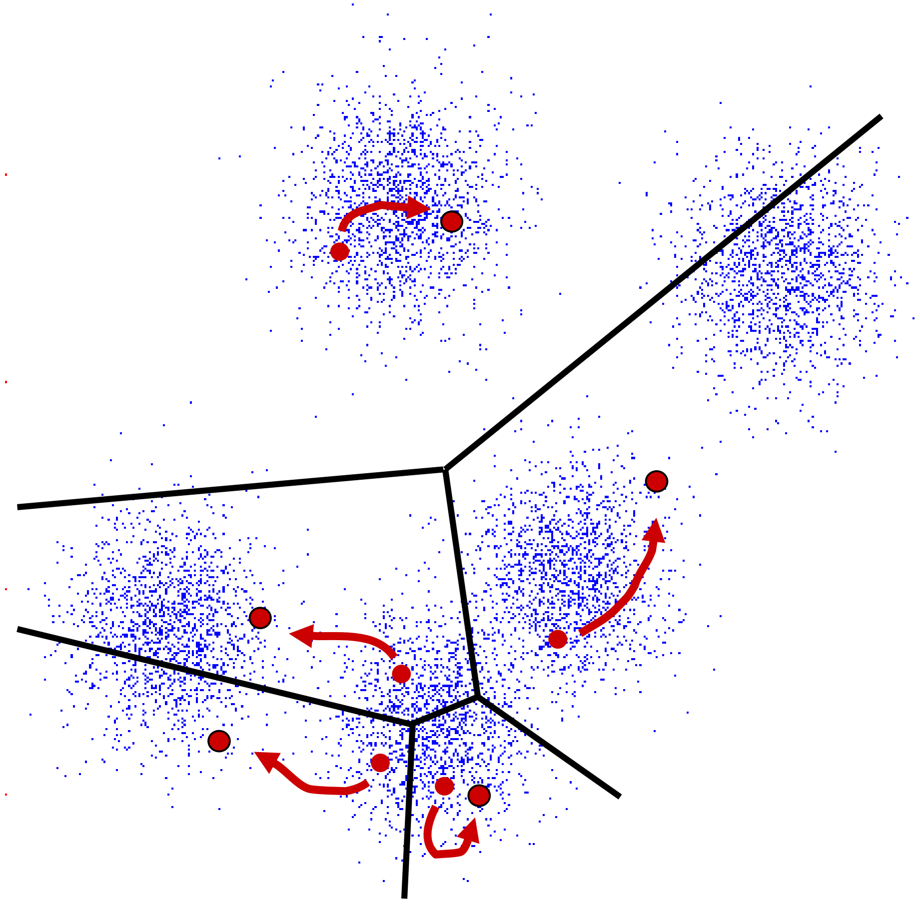
- What could "similar" mean?
  - One option: small (squared) Euclidean distance

K-Means



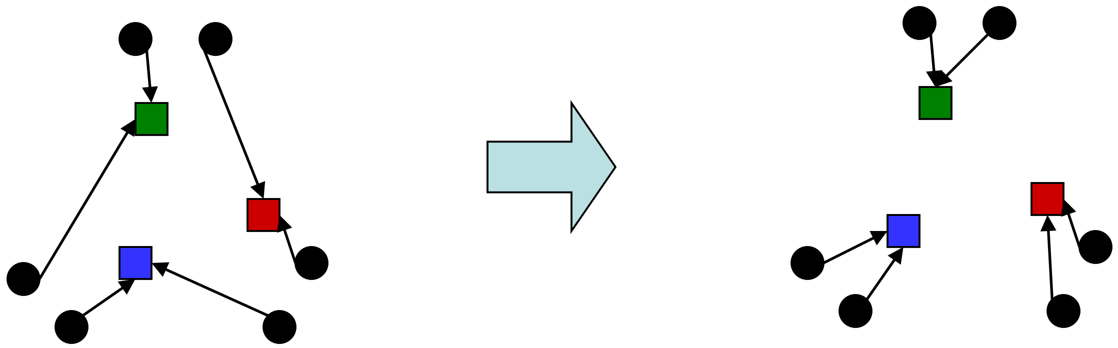
K-Means

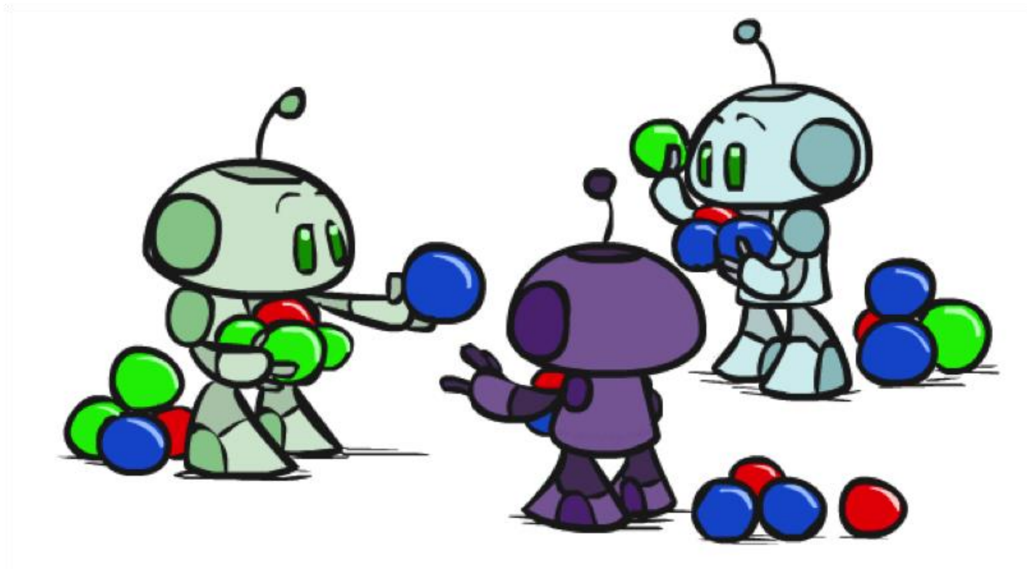
- An iterative clustering algorithm
  - Pick K random points as cluster centers (means)
  - Repeat:
    - Assign data instances to closest mean
    - Assign each mean to the average of its assigned points
  - Stop when no points' assignments change



Phase I: Update Assignments

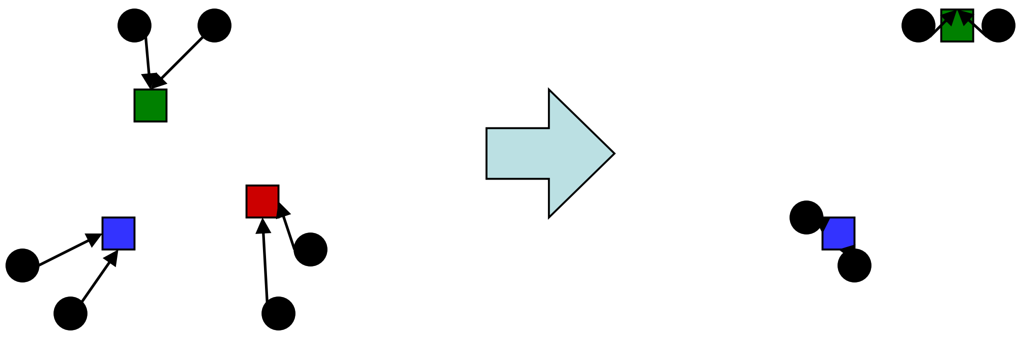
- For each point, re-assign to closest mean:

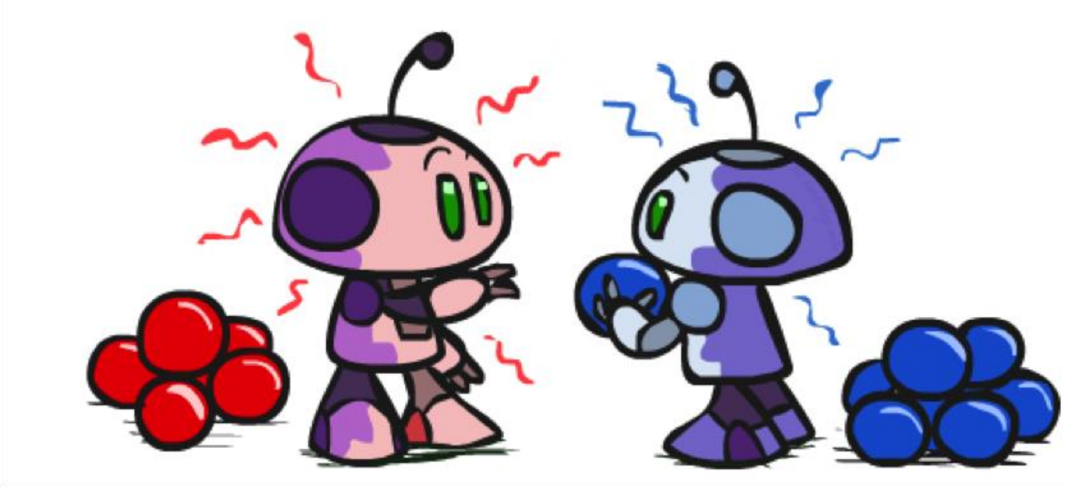




Phase II: Update Means

- Move each mean to the average of its assigned points:

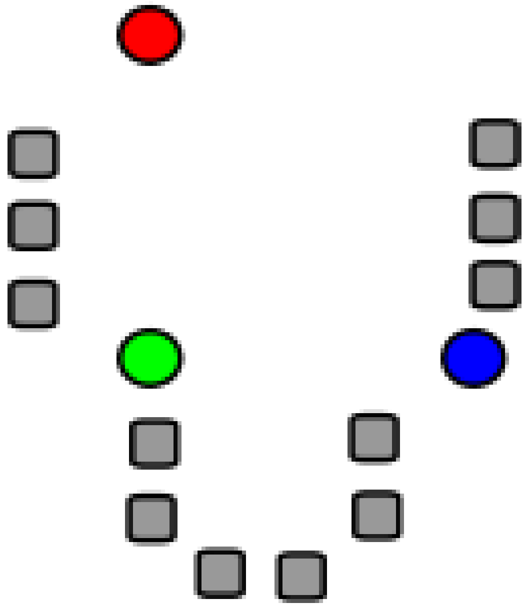




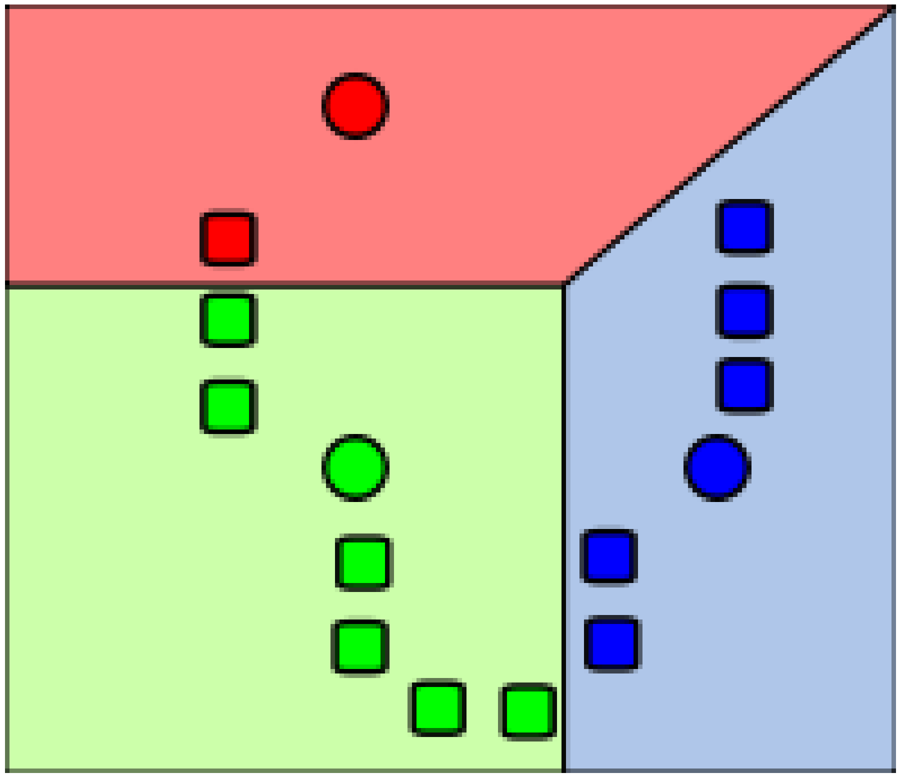
K-Means Demo

- An iterative clustering algorithm
  - Pick 2 random points as cluster centers (means): red and green
  - Repeat:
    - Assign data instances to closest mean: show red or green card
    - Move each mean to the average of its assigned points
  - Stop when no points' assignments change

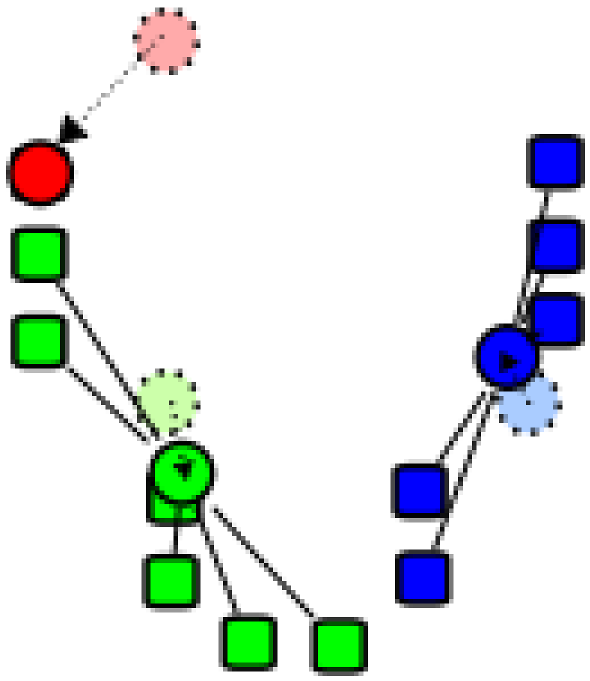
K-Means Example



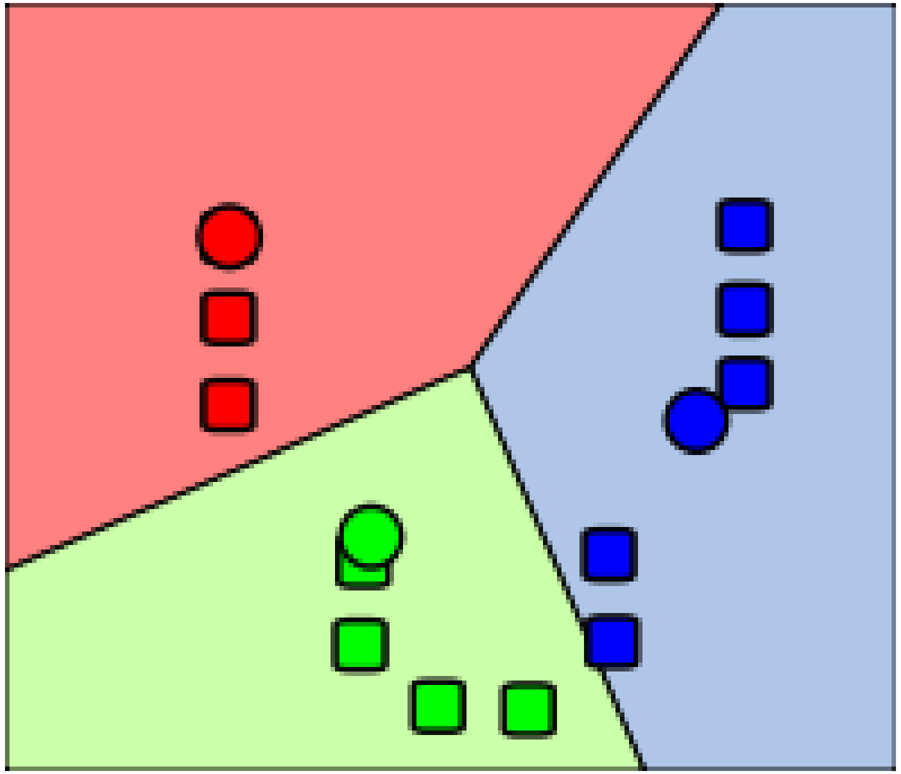
K-Means Example



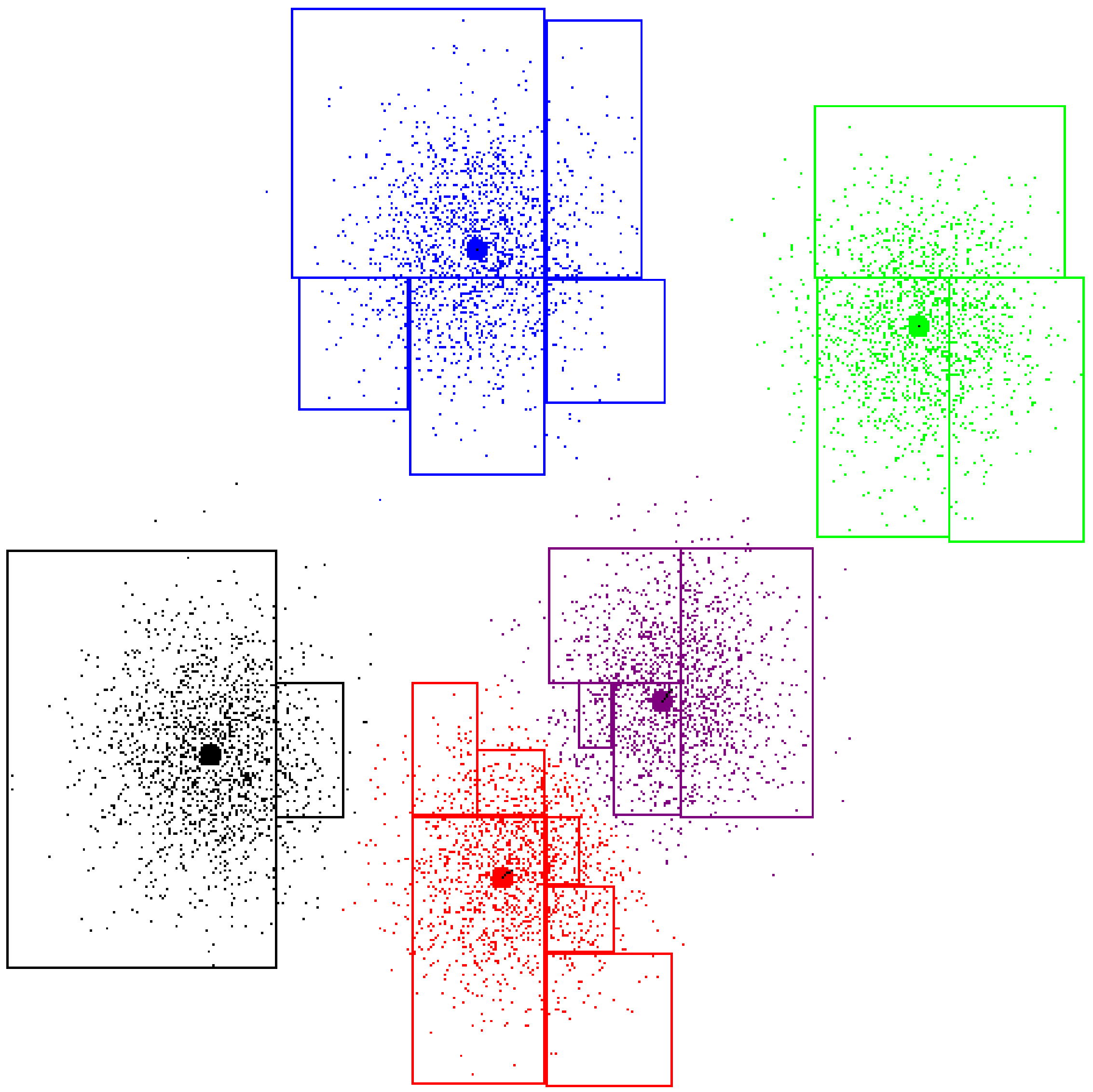
K-Means Example



K-Means Example



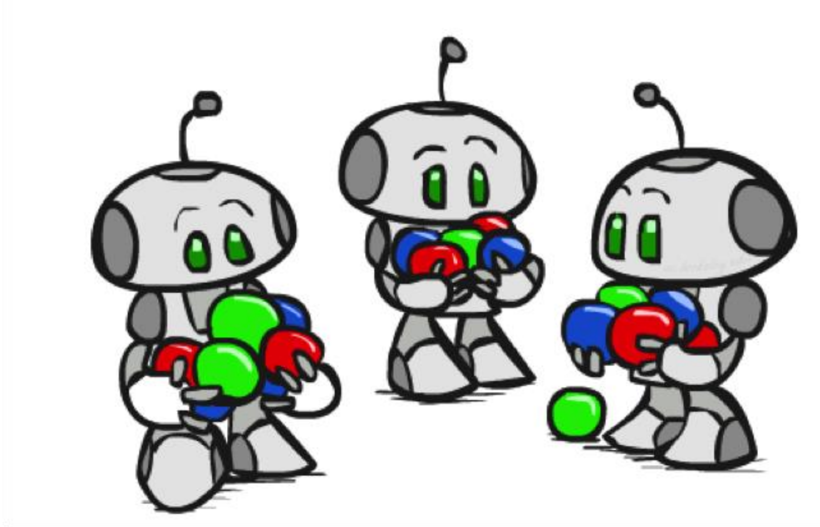
K-Means Example



Initialization

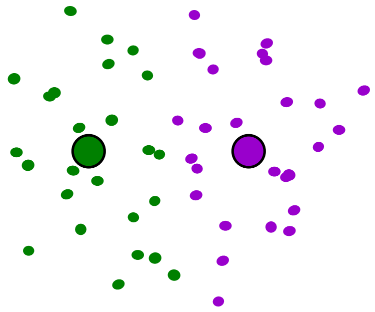
- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?

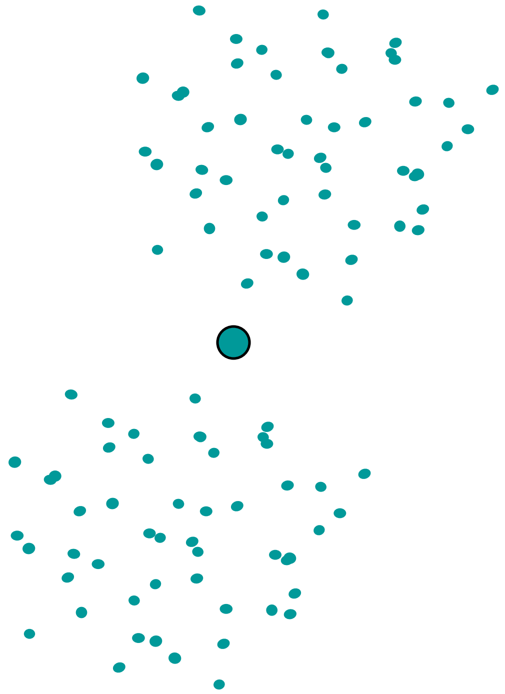
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



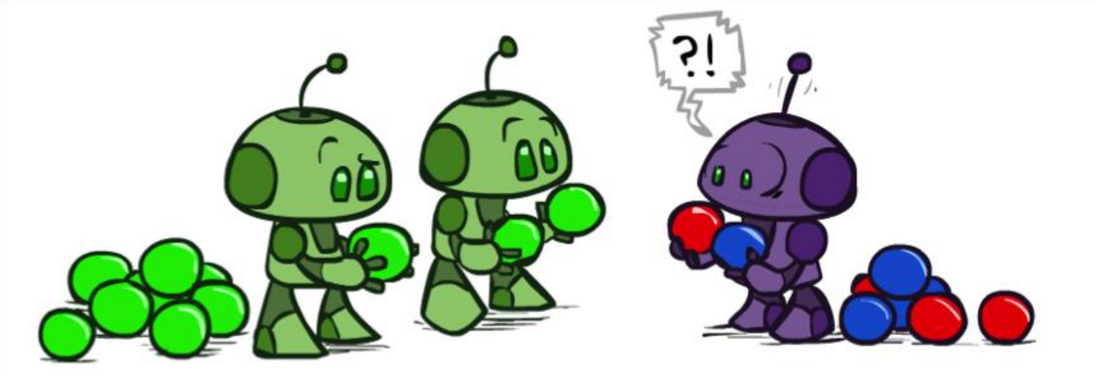
K-Means Getting Stuck

- A local optimum:



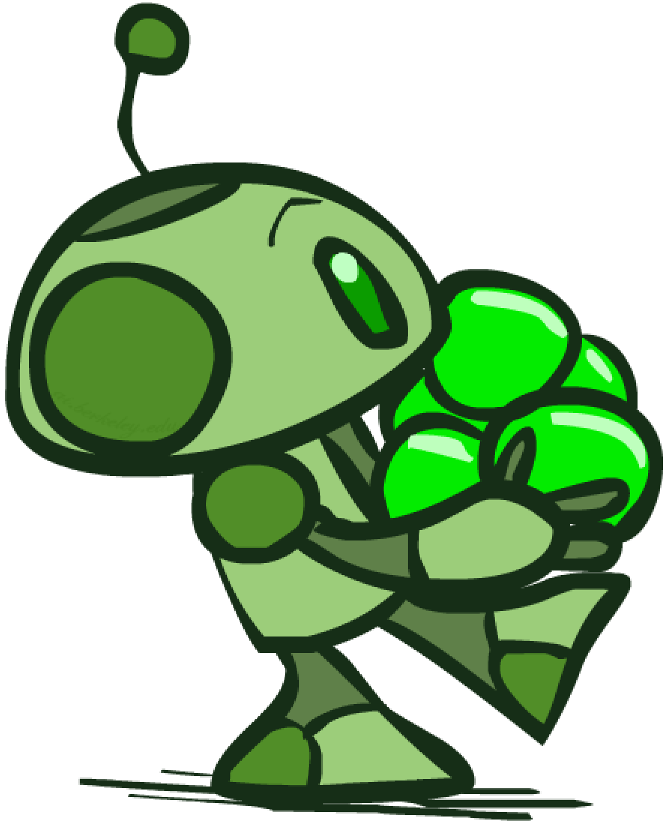


Why doesn't this work out like the earlier example, with the purple taking over half the blue?

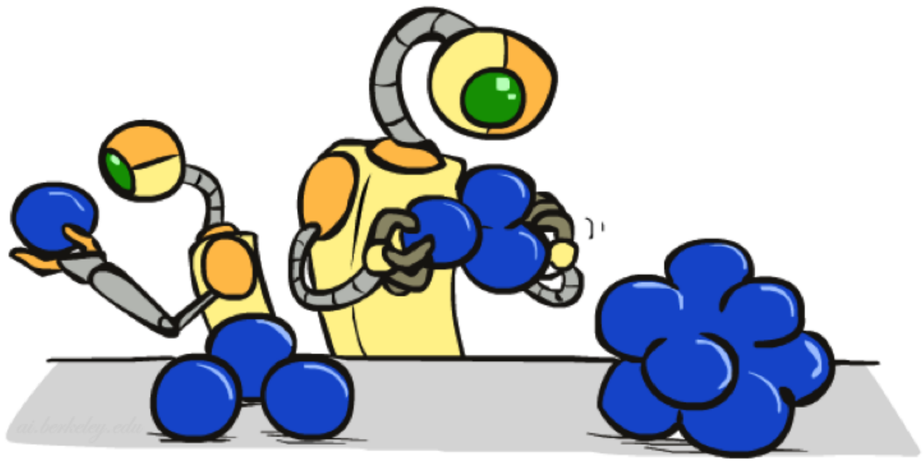


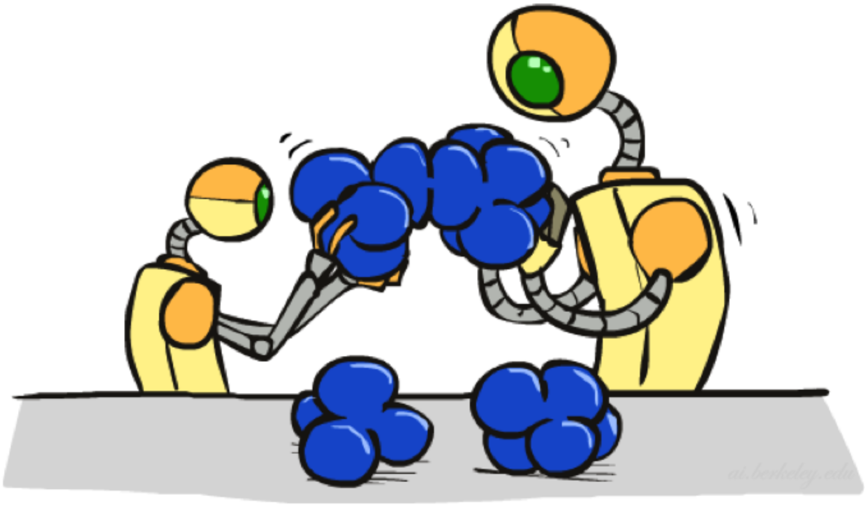
K-Means Questions

- Will K-means converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
  - If the patterns are very very clear?
- Will it find something interesting?
- Do people ever use it?
- How many clusters to pick?



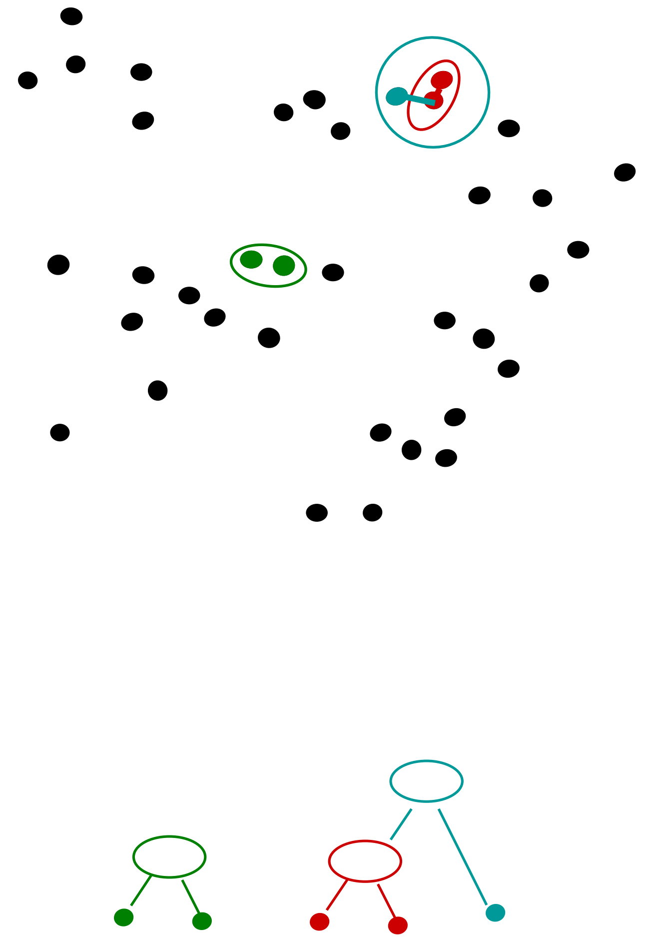
Agglomerative Clustering





Agglomerative Clustering

- First merge very similar instances
- Incrementally build larger clusters out of smaller clusters
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left



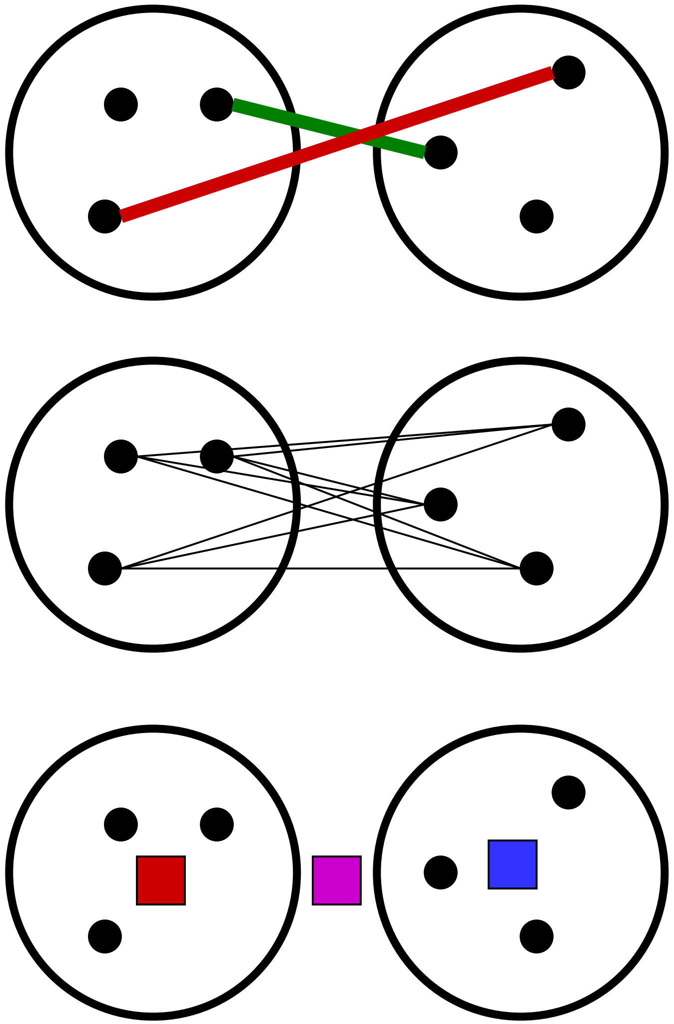
- Produces not one clustering, but a family of clusterings represented by a dendrogram

Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?

- Many options
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward’s method (min variance, like k-means)

- Different choices create different clustering behaviors



Example: Google News

artificial intelligence

All News Videos Images Books More Search tools

About 943,000 results (0.47 seconds)

White House Wants to Discuss the Pros and Cons of **Artif**...
Fortune - 14 hours ago
The goal of the workshops is to promote discussion about the pros and cons of **artificial intelligence**. AI opportunities include smarter machines ...

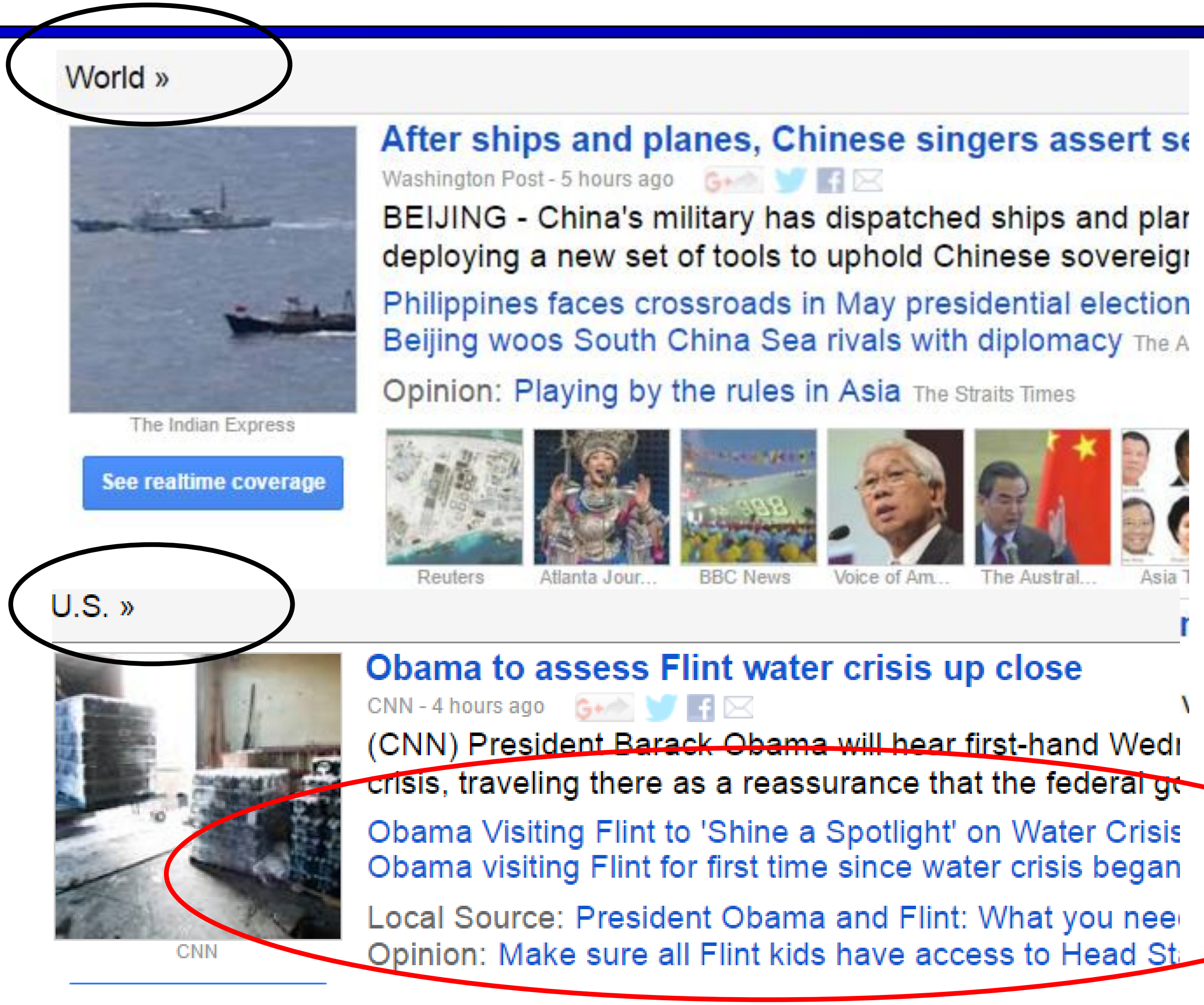
The White House Has Realized **Artificial Intelligence** Is Very Important
Popular Science - 15 hours ago
Explore in depth (5 more articles)

Elon Musk Opens $1 Billion "Robot Gym" for **Artificial Int**...
Nature World News - 1 hour ago
Elon Musk developed a $1 billion-program "Robot Gym" to train **artificial intelligence**. He says he wanted the robots to be as versatile as ...

Elon Musk's Research Company Develops a "Training Gym" for ...
Electronic Design - 22 hours ago
Explore in depth (2 more articles)

Story groupings: unsupervised clustering

Example: Google News



Top-level categories:
supervised classification

Story groupings:
unsupervised clustering