18-661 Introduction to Machine Learning

Clustering, Part I

Spring 2020

ECE – Carnegie Mellon University

Announcements

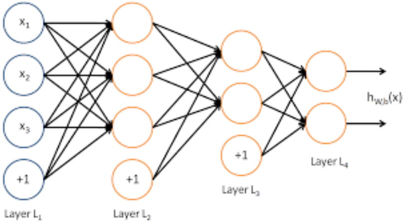
- **Homework 5:** due on April 1st

Outline

1. Review of Neural Networks
2. Clustering
3. k-means
4. k-means++

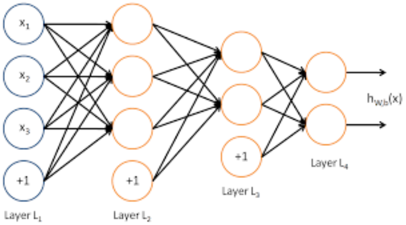
Review of Neural Networks

Neural Network Compress the Set of Features



- Start with feature vector **x** containing all pixels in the image
- Layer 1: distill the edges of the image
- Layer 2: distill triangles, circles, etc.
- Layer 3: recognize pointy ears, fur style etc.
- Layer 4: performs logistic regression on the features in layer 3

Neural Network Compress the Set of Features



- Start with feature vector **x** containing all pixels in the image
- Layer 1: distill the edges of the image
- Layer 2: distill triangles, circles, etc.
- Layer 3: recognize pointy ears, fur style etc.
- Layer 4: performs logistic regression on the features in layer 3

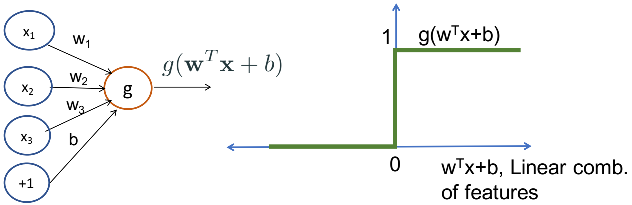
We cannot directly control what each layer learns; this depends on the training data

Perceptron: Rosenblatt (1957)

- The perception is a single-unit neural network with the heavyside activation function or sign(x)

Perceptron: Rosenblatt (1957)

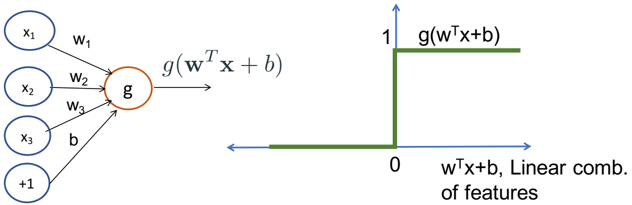
- The perception is a single-unit neural network with the heavyside activation function or sign(x)
- It considers a linear binary classification problem to distinguish between two classes $\{-1, +1\}$.



Perceptron

Perceptron: Rosenblatt (1957)

- The perception is a single-unit neural network with the heavyside activation function or sign(x)
- It considers a linear binary classification problem to distinguish between two classes $\{-1, +1\}$.

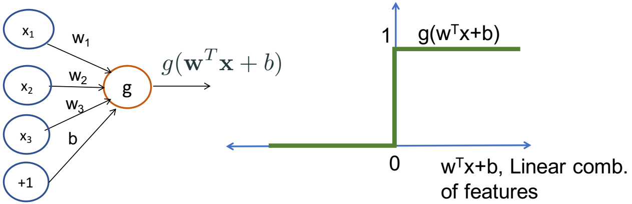


Perceptron

- Assign label sign($\boldsymbol{w}^\top \boldsymbol{x} + b$) to a new sample

Perceptron: Rosenblatt (1957)

- The perception is a single-unit neural network with the heavyside activation function or sign(x)
- It considers a linear binary classification problem to distinguish between two classes $\{-1, +1\}$.



Perceptron

- Assign label sign($\boldsymbol{w}^\top \boldsymbol{x} + b$) to a new sample
- Notation change: Merge b into the vector $\mathbf{w}$ and append 1 to the vector $\mathbf{x}$

How to learn the weights w?

The objective is to learn $\mathbf{w}$ that minimizes the number of errors on the training dataset. That is, minimize

$$\varepsilon = \sum_n \mathbb{I}[y_n \neq \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)]$$

Algorithm: For a randomly chosen data point $(\mathbf{x}_n, y_n)$ make small changes to $\boldsymbol{w}$ so that

$$y_n = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)$$

How to learn the weights w?

The objective is to learn **w** that minimizes the number of errors on the training dataset. That is, minimize

$$\varepsilon = \sum_n \mathbb{I}[y_n \neq \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)]$$

Algorithm: For a randomly chosen data point $(\mathbf{x}_n, y_n)$ make small changes to $\boldsymbol{w}$ so that

$$y_n = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)$$

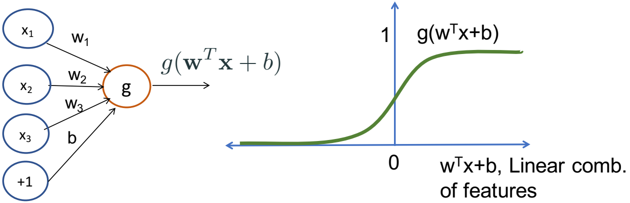
Two cases

- If $y_n = \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)$, do nothing.
- If $y_n \neq \text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_n)$,

$$\boldsymbol{w}^{\text{NEW}} \leftarrow \boldsymbol{w}^{\text{OLD}} + y_n \boldsymbol{x}_n$$

Binary Logistic Regression

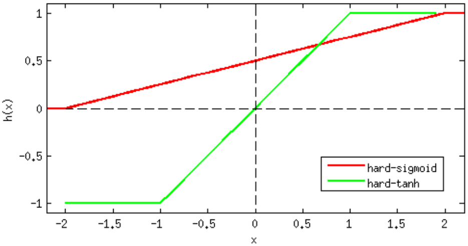
- Suppose g is the sigmoid function $\sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}}$
- We can find a linear decision boundary separating two classes. The output is the probability of $\mathbf{x}$ belonging to class 1.



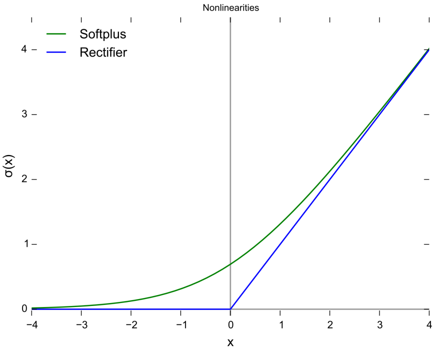
Neuron with Sigmoid activation

Choice of Activation Function

- Sigmoid unit $\sigma(z) = \frac{1}{1+e^{-z}}$
- Tanh Unit $tanh(z) = 2\sigma(2z) - 1$
  - Both are squashing type non-linearity
  - Problem: Saturate across most of their domain, strongly sensitive only when z is closer to zero
- To avoid the problem of vanishing gradients we can use piece-wise linear approximations to these functions
- This significantly reduces the computation complexity because gradients can take only one a few values

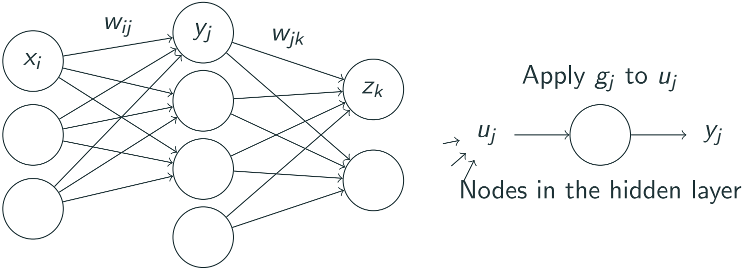


Rectified Linear Units



- Approximates the softplus function which is $\log(1 + e^z)$
- ReLu Activation function is $g(z) = max(0, z)$ with $z \in R$
- Similar to linear units. Easy to optimize!
- Give large and *consistent* gradients when active
- Modifications: Leaky ReLUs

The Back-propagation Algorithm



- w_{ij}: weights connecting node i in layer $(\ell - 1)$ to node j in layer ℓ.
- b_j, b_k: bias for nodes j and k.
- u_j, u_k: inputs to nodes j and k (where $u_j = b_j + \sum_i x_i w_{ij}$).
- g_j, g_k: activation function for node j (applied to u_j) and node k.
- $y_j = g_j(u_j)$, $z_k = g_k(u_k)$: output/activation of nodes j and k.
- t_k: target value for node k in the output layer.

The Back-propagation Algorithm

Back-propagation the error. Given parameters w, b:

- Step 1: Forward-propagate to find z_k in terms of the input (the "feed-forward signals").
- Step 2: Calculate output error E by comparing the predicted output z_k to its true value t_k.
- Step 3: Back-propagate E by weighting it by the gradients of the associated activation functions and the weights in previous layers.
- Step 4: Calculate the gradients $\frac{\partial E}{\partial w}$ and $\frac{\partial E}{\partial b}$ for the parameters w, b at each layer based on the backpropagated error signal and the feedforward signals from the inputs.
- Step 5: Update the parameters using the calculated gradients $w \leftarrow w - \eta\frac{\partial E}{\partial w}$, $b \leftarrow b - \eta\frac{\partial E}{\partial b}$ where η is the step size.

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size

Avoiding Overfitting

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size
- Learning Rate

Avoiding Overfitting

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size
- Learning Rate
- Momentum

Avoiding Overfitting

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size
- Learning Rate
- Momentum
- Width of each layer

Avoiding Overfitting

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size
- Learning Rate
- Momentum
- Width of each layer

Avoiding Overfitting

- Regularizing the loss function

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

- Mini-batch size
- Learning Rate
- Momentum
- Width of each layer

Avoiding Overfitting

- Regularizing the loss function
- Choosing the right network depth

Optimizing SGD For Faster Convergence

Speed of convergence depends on parameters such as

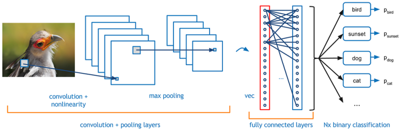
- Mini-batch size
- Learning Rate
- Momentum
- Width of each layer

Avoiding Overfitting

- Regularizing the loss function
- Choosing the right network depth
- Dropout– Ensemble of several models

Deep Convolutional Networks

- Deep supervised neural networks are generally too difficult to train
- **One notable exception:** Convolutional neural networks (CNN)
- Convolutional nets were inspired by the visual system's structure
- Have much fewer connections and parameters and so they are easier to train



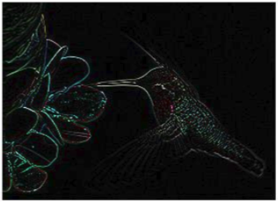
2-Dimensional Convolution

$$f[x.y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot [x - n_1, y - n_2]$$

https://graphics.stanford.edu/courses/cs178/applets/convolution.html

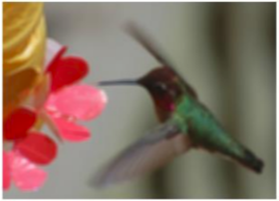
Filter

0.00	0.00	0.00	0.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	-2.00	8.00	-2.00	0.00
0.00	0.00	-2.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00

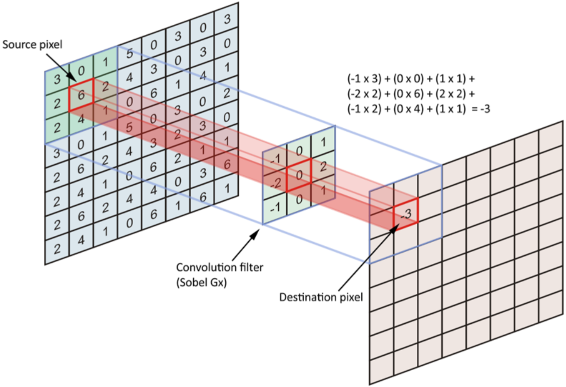


Original

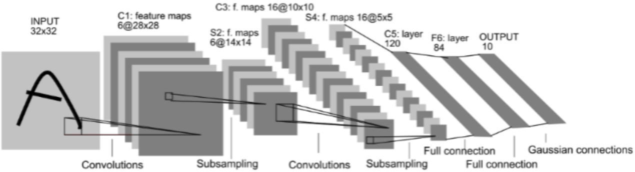
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04
0.04	0.04	0.04	0.04	0.04



Convolve subsets of an image with a small filter. Each pixel in the output image is a weighted sum of the filter and a subset of the input.

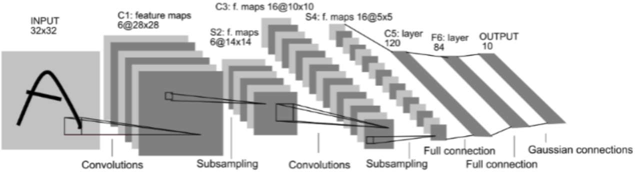


LeNet 5, LeCun 1998



- **Input**: 32×32 pixel image. Largest character is 20×20 (All important info should be in the center of the receptive field of the highest level feature detectors)
- **Cx**: Convolutional layer (C1,C3,C5)
- **Sx**: Sub-sample layer (S2,S4)
- **Fx**: Fully connected layer (F6)

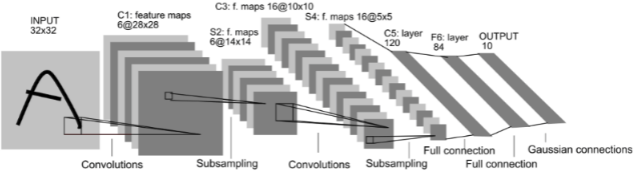
LeNet 5, Layer C1



C1: Convolutional layer with 6 feature maps of size 28X28 $C1^k(k = 1..6)$

Each unit of C1 has 5x5 receptive field in the input layer.

LeNet 5, Layer C1



C1: Convolutional layer with 6 feature maps of size 28X28 $C1^k (k = 1..6)$

Each unit of C1 has 5x5 receptive field in the input layer.

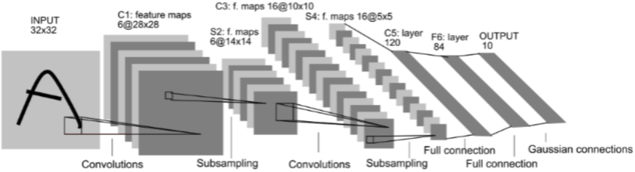
- Topological structure
- Sparse connections
- Shared weights

$(5 * 5 + 1) * 6 = 156$ parameters to learn

Connections: $28 * 28 * (5 * 5 + 1) * 6 = 122304$

If it was fully connected, we had (32*32+1)*(28*28)*6 parameters

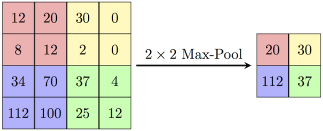
LeNet 5, Layer S2



S2: Sub-sampling layer with 6 feature maps of size 14×14

2×2 non-overlapping receptive fields in C1

These days, we typically use



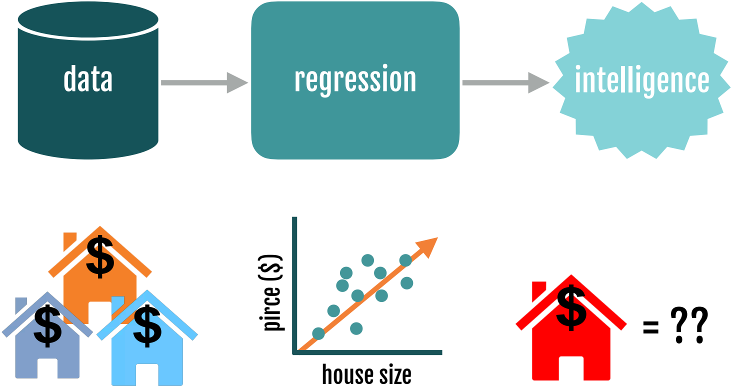
Outline

1. Review of Neural Networks
2. Clustering
3. k-means
4. k-means++

Clustering

Supervised Learning: Regression

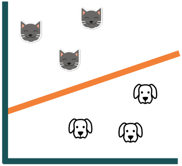
How much should you sell your house for?

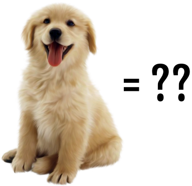


input: houses & features **learn**: $x \to y$ relationship **predict**: y (*continuous*)

Supervised Learning: Classification

Cat or dog?





input: cats and dogs

learn: $x \rightarrow y$ relationship

predict: y (*categorical*)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

- Learning algorithm must find patterns from features alone

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

- Learning algorithm must find patterns from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

- Learning algorithm must find patterns from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

- Learning algorithm must find patterns from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

- Learning algorithm must find patterns from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
    - Clustering (today)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_n, y_n)\}$

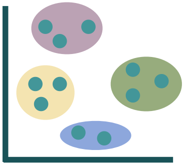
- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$

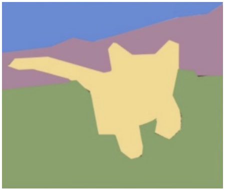
- Learning algorithm must find patterns from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
  - Clustering (today)
  - Dimensionality Reduction: Transform an initial feature representation into a more concise representation

How to segment an image?

input: raw pixels $\{x\}$



separate: $\{x\}$ into sets



output: cluster labels $\{z\}$

Clustering

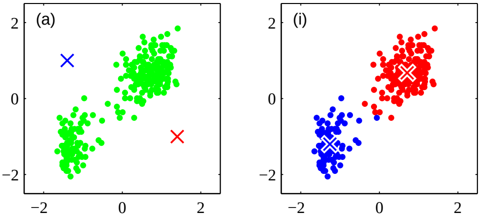
Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ and K, we want to output:

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$: prototypes of clusters

Toy Example Cluster data into two clusters.

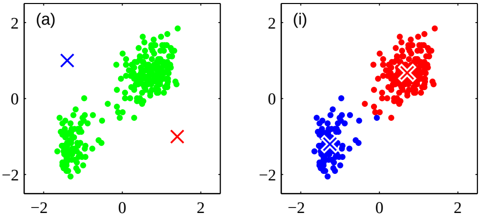


Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

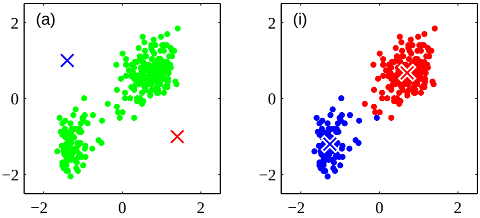


Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

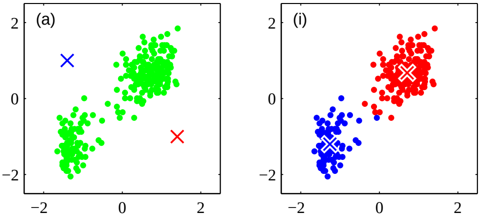


Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.



Example Applications

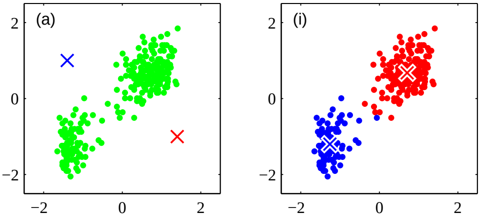
- Identify communities within social networks

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.



Example Applications

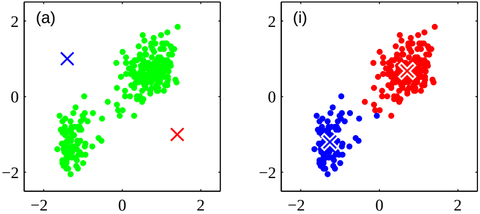
- Identify communities within social networks
- Find topic groups in news stories

Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output:

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership

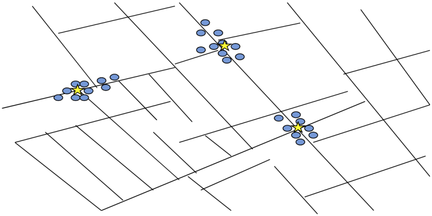
Toy Example Cluster data into two clusters.



Example Applications

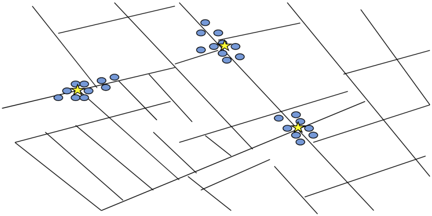
- Identify communities within social networks
- Find topic groups in news stories
- Group similar sequences into gene families

First? Example of Clustering



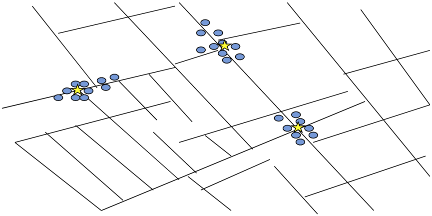
- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.

First? Example of Clustering



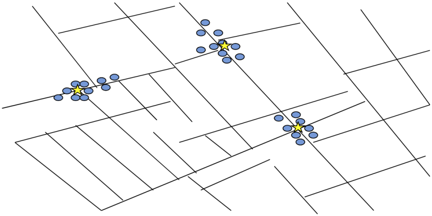
- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.

First? Example of Clustering



- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells – thus exposing both the problem and the solution.

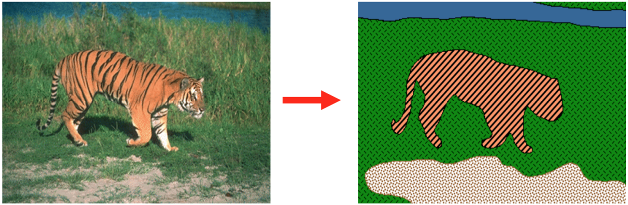
First? Example of Clustering



- John Snow, a London physician plotted the location of cholera deaths on a map during an outbreak in the 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells – thus exposing both the problem and the solution.
- This story is all the more relevant today as we are trying to overcome the COVID-19 outbreak

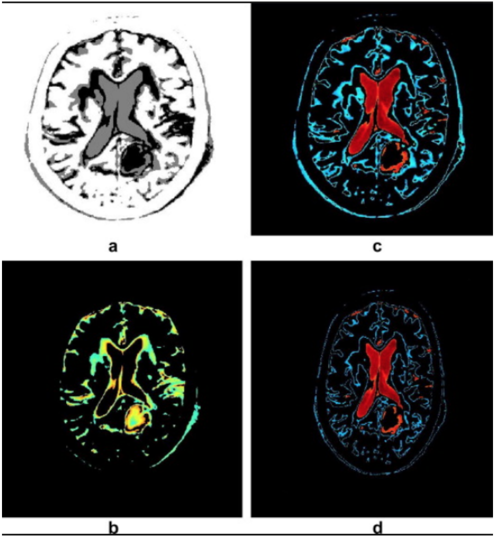
More examples

Image segmentation into foreground and background



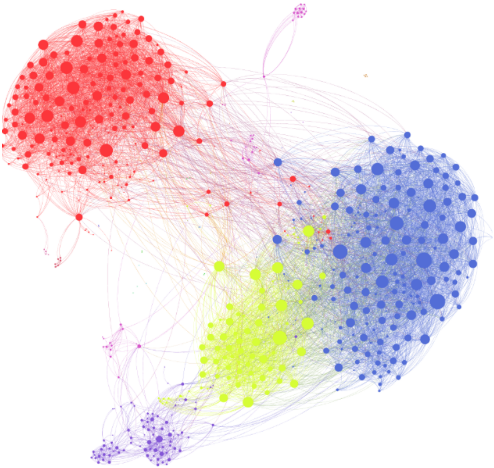
More examples

Detecting brain lesions from MRI Scans



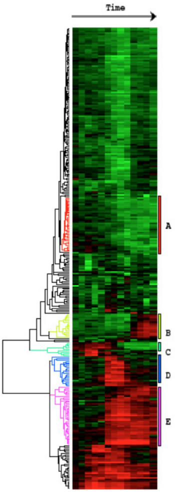
More examples

Social network analysis



More examples

Clustering gene expression data



Clustering

Today we will cover two methods for clustering

- k-means
- k-means++

k-means

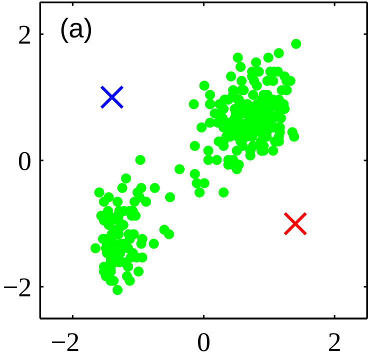
k-means

k-means: an iterative clustering method

High-level idea:

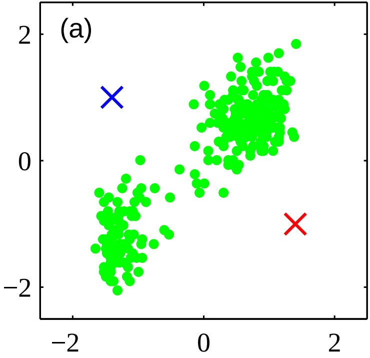
- Initialize: Pick k random points as cluster centers, $\{\mu_1, \ldots, \mu_k\}$
- Alternate:
  1. Assign data points to closest cluster center in $\{\mu_1, \ldots, \mu_k\}$
  2. Change each cluster center to the average of its assigned points
- Stop: When the clusters are stable

- Initialize: Pick k random points as cluster centers

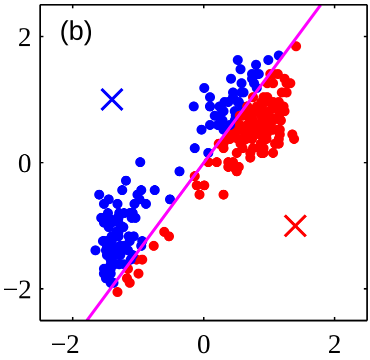


k-means example

- Initialize: Pick k random points as cluster centers
- (Shown here for $k=2$)

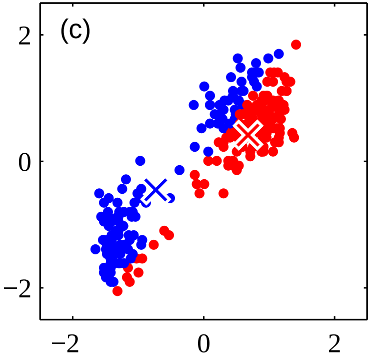


- Alternating Step 1: Assign data points to closest cluster center



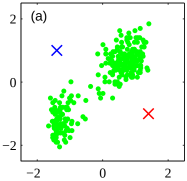
k-means example

- Alternating Step 2: Change the cluster center to the average of the assigned points

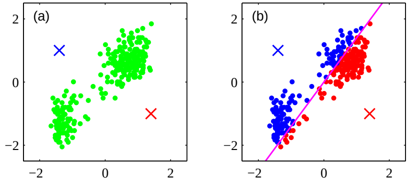


Then: Repeat . . .

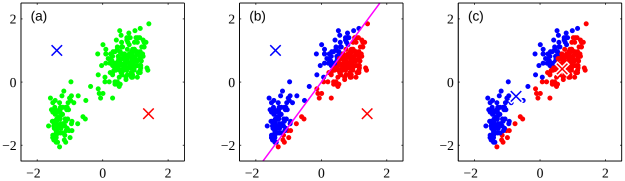
k-means example (several iterations)



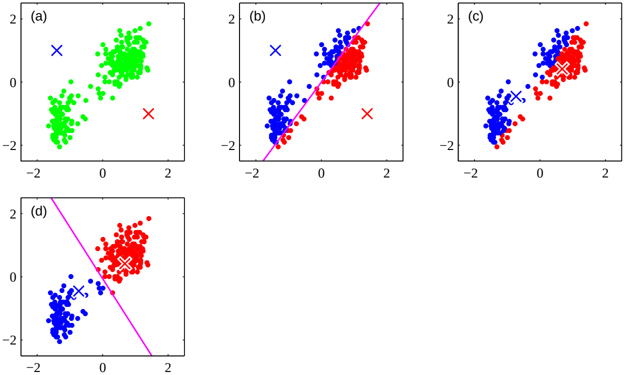
k-means example (several iterations)



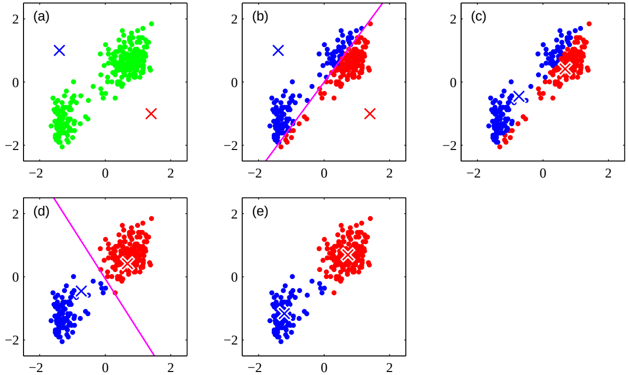
k-means example (several iterations)



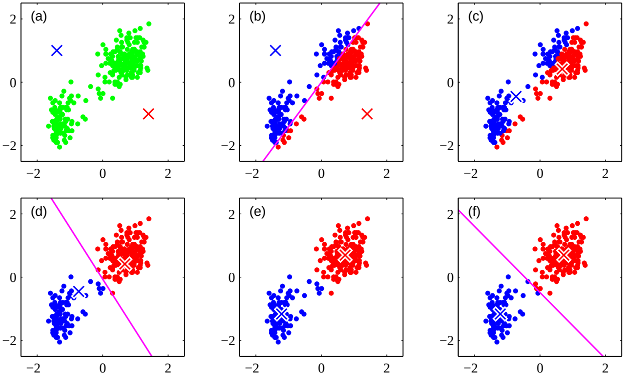
k-means example (several iterations)



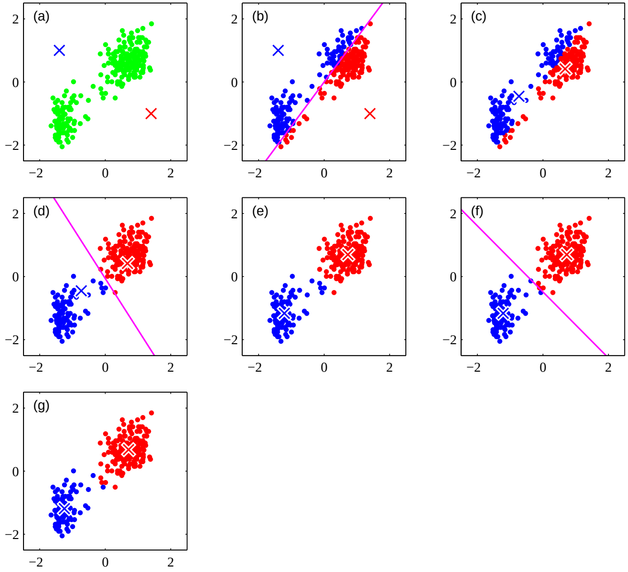
k-means example (several iterations)



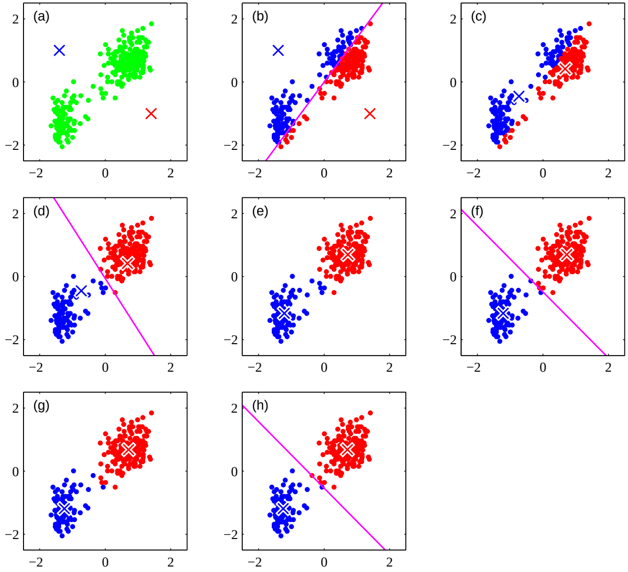
k-means example (several iterations)



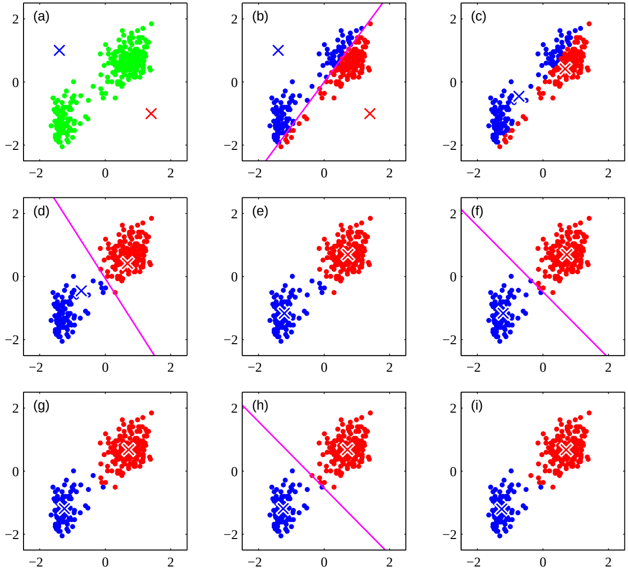
k-means example (several iterations)



k-means example (several iterations)



k-means example (several iterations)



k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2 = \sum_{k=1}^{K} \underbrace{\sum_{n:A(\boldsymbol{x}_n)=k} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2}_{\text{spread within the } k\text{th cluster}}$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2 = \sum_{k=1}^{K} \underbrace{\sum_{n:A(\boldsymbol{x}_n)=k} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2}_{\text{spread within the } k\text{th cluster}}$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Notes:

- Distance measure: $\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$ calculates how far $\boldsymbol{x}_n$ is from the cluster center $\boldsymbol{\mu}_k$

k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype $\boldsymbol{\mu}_k$

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2 = \sum_{k=1}^{K} \underbrace{\sum_{n:A(\boldsymbol{x}_n)=k} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2}_{\text{spread within the } k\text{th cluster}}$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Notes:

- Distance measure: $\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$ calculates how far $\boldsymbol{x}_n$ is from the cluster center $\boldsymbol{\mu}_k$
- Canonical example is the 2-norm, i.e., $\|\cdot\|_2^2$, but could be some other distance measure!

Algorithm

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values

Algorithm

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Fix $\{\boldsymbol{\mu}_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \text{argmin}_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2$$

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Fix $\{\boldsymbol{\mu}_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Fix $\{r_{nk}\}$ and minimize over $\{\boldsymbol{\mu}_k\}$ to get this update:

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}$$

- **Step 3** Return to Step 1 unless stopping criterion is met

Properties of k-means algorithm

Does it converge?

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
    - Key idea: k-means is an alternating optimization approach
    - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
    - *However*, may converge to a *local minimum* (objective is non-convex)

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

Properties of *k*-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
    - Key idea: k-means is an alternating optimization approach
    - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
    - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

- **Running time per iteration**:

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

- **Running time per iteration**:
  - Assume: n data points, each with d features, and k clusters

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

- **Running time per iteration**:
  - Assume: n data points, each with d features, and k clusters
  - Assign data points to closest cluster: $O(ndk)$

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

- **Running time per iteration**:
  - Assume: n data points, each with d features, and k clusters
  - Assign data points to closest cluster: $O(ndk)$
  - Re-compute cluster centers: $O(ndk)$

Properties of k-means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
  - Key idea: k-means is an alternating optimization approach
  - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
  - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

- **Running time per iteration**:
  - Assume: n data points, each with d features, and k clusters
  - Assign data points to closest cluster: $O(ndk)$
  - Re-compute cluster centers: $O(ndk)$
- **Thus, total runtime is**: $O(ndki)$, where i is the number of iterations

Practical Issues with k-means

- How to select k?

Practical Issues with k-means

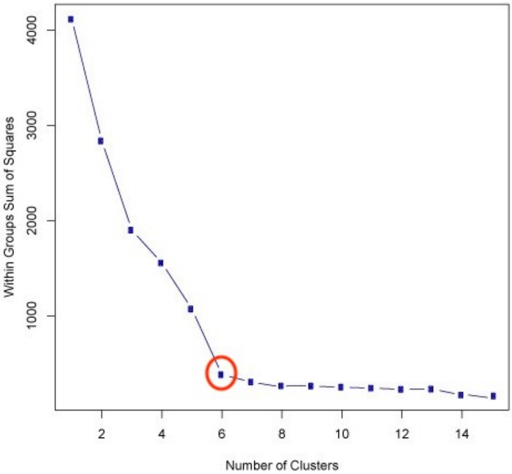
- How to select k?
  - Prior knowledge

Practical Issues with k-means

- How to select k?
  - Prior knowledge
  - Heuristics (e.g., elbow method)

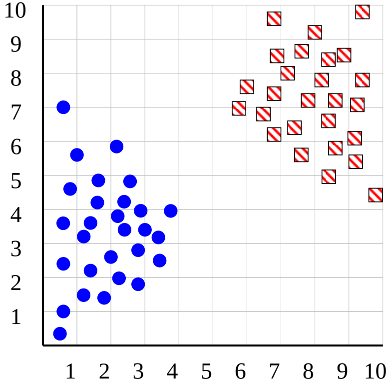
Elbow method

Key idea: select a small value of k that adding a new cluster doesn't reduce the within-cluster distances much

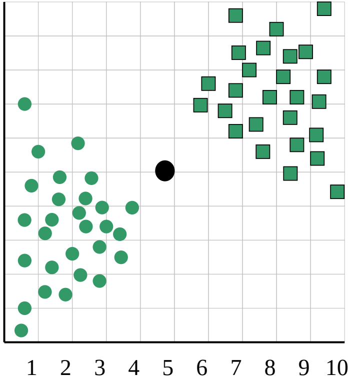


How can we tell the *right* number of clusters?

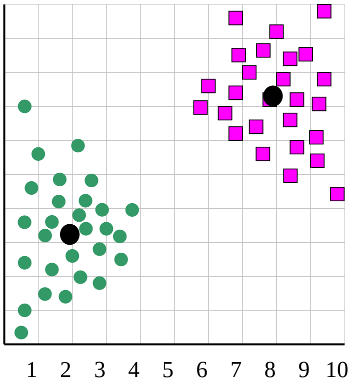
In general, this is a unsolved problem. However there are many approximate methods. In the next few slides we will see an example.



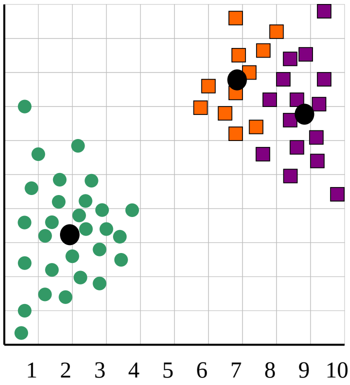
When k = 1, the objective function is 873.0



When k = 2, the objective function is 173.1



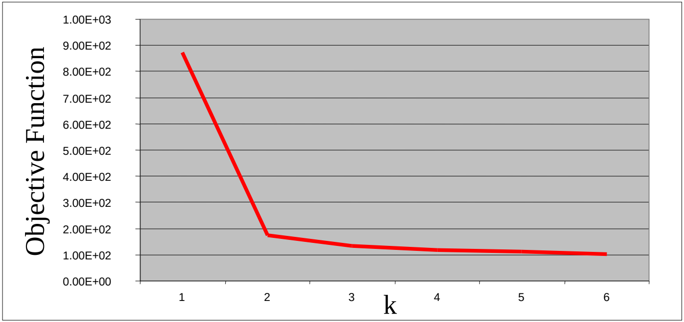
When k = 3, the objective function is 133.6



Elbow Method

We can plot the objective function values for k equals 1 to 6…

The abrupt change at k = 2, is highly suggestive of two clusters in the data. This technique for determining the number of clusters is known as “knee finding” or “elbow finding”.

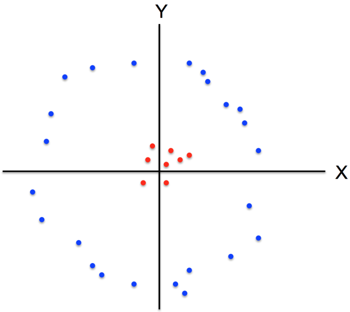


Note that the results are not always as clear cut as in this toy example

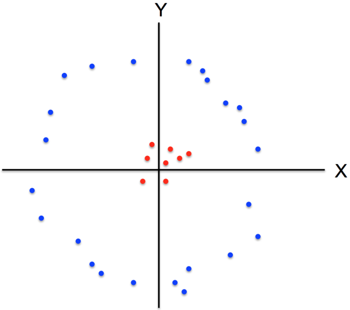
Practical Issues with k-means

- How to select k?
  - Prior knowledge
  - Heuristics (e.g., elbow method)
- How to select distance measure?
  - Often requires some knowledge of problem
  - Some examples: Euclidean distance (for images), Hamming distance (distance between two strings), shared key words (for websites)

How to get k-means to work on this data?



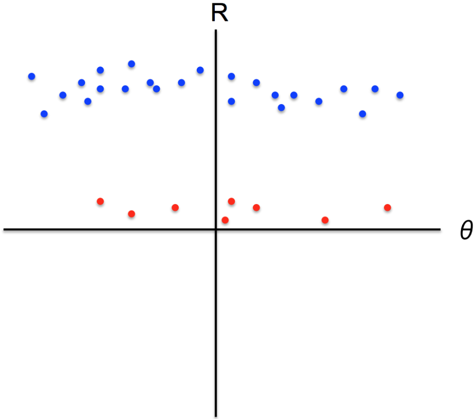
How to get k-means to work on this data?



Should look at the distance of the data points from the origin $\sqrt{x_n^2 + y_n^2}$

Distance measure

Changing features (distance measure) can help

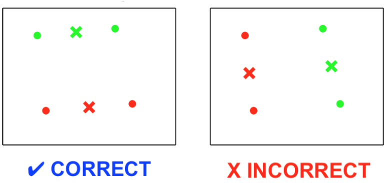


Practical Issues with k-means

- How to select k?
  - Prior knowledge
  - Heuristics (e.g., elbow method)
- How to select distance measure?
  - Often requires some knowledge of problem
  - Some examples: Euclidean distance (for images), Hamming distance (distance between two strings), shared key words (for websites)
- How to initialize cluster centers?
  - The final clustering can depend significantly on the initial points you pick!

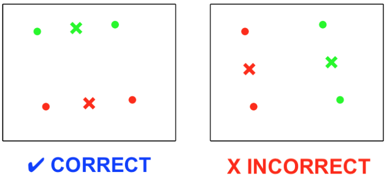
How to initialize cluster centers?

Random initialization can lead to *different results*

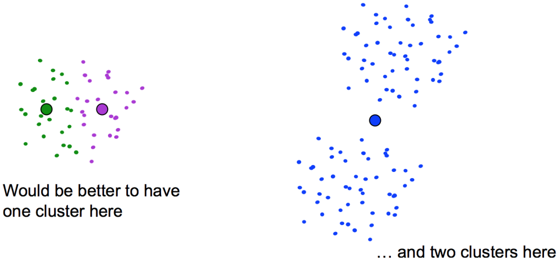


How to initialize cluster centers?

Random initialization can lead to *different results*



Choosing k is also non-trivial



k-means++

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \ldots, k$

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \ldots, k$
  - Choose μ_j among $x_1, \ldots, x_n$ with probability:

    $$P(\mu_j = x_i) \propto min_{j'<j} \|x_i - \mu_{j'}\|^2$$

    This means that if x_i is close to one of the already chosen cluster means $\mu_1, \ldots \mu_{j-1}$, then we assign a lower probability of selecting it as the next cluster mean.

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \ldots, k$
  - Choose μ_j among $x_1, \ldots, x_n$ with probability:

    $$P(\mu_j = x_i) \propto min_{j'<j} \|x_i - \mu_{j'}\|^2$$

    This means that if x_i is close to one of the already chosen cluster means $\mu_1, \ldots \mu_{j-1}$, then we assign a lower probability of selecting it as the next cluster mean.

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \ldots, k$
  - Choose μ_j among $x_1, \ldots, x_n$ with probability:

$$P(\mu_j = x_i) \propto min_{j'<j} \|x_i - \mu_{j'}\|^2$$

    This means that if x_i is close to one of the already chosen cluster means $\mu_1, \ldots \mu_{j-1}$, then we assign a lower probability of selecting it as the next cluster mean.

Initialization helps to get good coverage of the space

k-means++

Key idea: Run k-means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \ldots, k$
  - Choose μ_j among $x_1, \ldots, x_n$ with probability:

    $$P(\mu_j = x_i) \propto min_{j' < j} \|x_i - \mu_{j'}\|^2$$

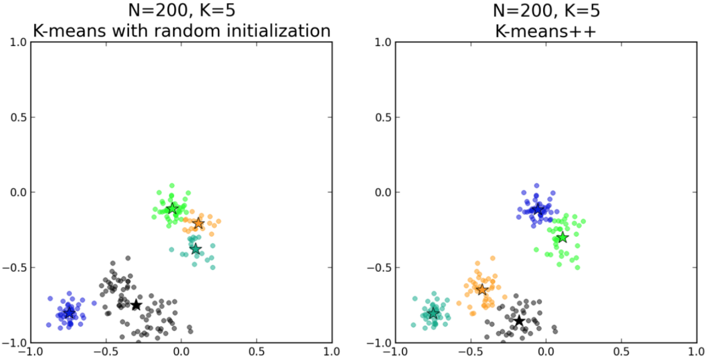
    This means that if x_i is close to one of the already chosen cluster means $\mu_1, \ldots \mu_{j-1}$, then we assign a lower probability of selecting it as the next cluster mean.

Initialization helps to get good coverage of the space

Theorem: k-means++ always obtains a $O(logk)$ approximation to the optimal solution in expectation.

Running k-means after this initialization can only improve on the result

k-means++



N=200, K=5
K-means with random initialization
N=200, K=5
K-means++
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

You should know ...

- What unsupervised learning is

You should know . . .

- What unsupervised learning is
- What clustering is

You should know ...

- What unsupervised learning is
- What clustering is
- How to cluster using k-means

You should know ...

- What unsupervised learning is
- What clustering is
- How to cluster using k-means
- Practical issues with k-means

You should know . . .

- What unsupervised learning is
- What clustering is
- How to cluster using k-means
- Practical issues with k-means
- How k-means++ improves on k-means