CS6375: Machine Learning

Gautam Kunapuli

Final Exam Review

THE UNIVERSITY OF TEXAS AT DALLAS
Erik Jonsson School of Engineering and Computer Science

Final Exam

Date: December 13, 2018
Time: 5:00pm-7:45pm
Location: JO 3.516

Topics:
- **Ensemble methods**
  - Bagging
  - Boosting
- **Clustering**
  - K-Means
  - Hierarchical clustering
  - Soft clustering & Gaussian Mixture Models
- **Dimensionality Reduction**: Principal Component Analysis
- **Neural Networks**
- **Reinforcement Learning**

Ensemble Methods

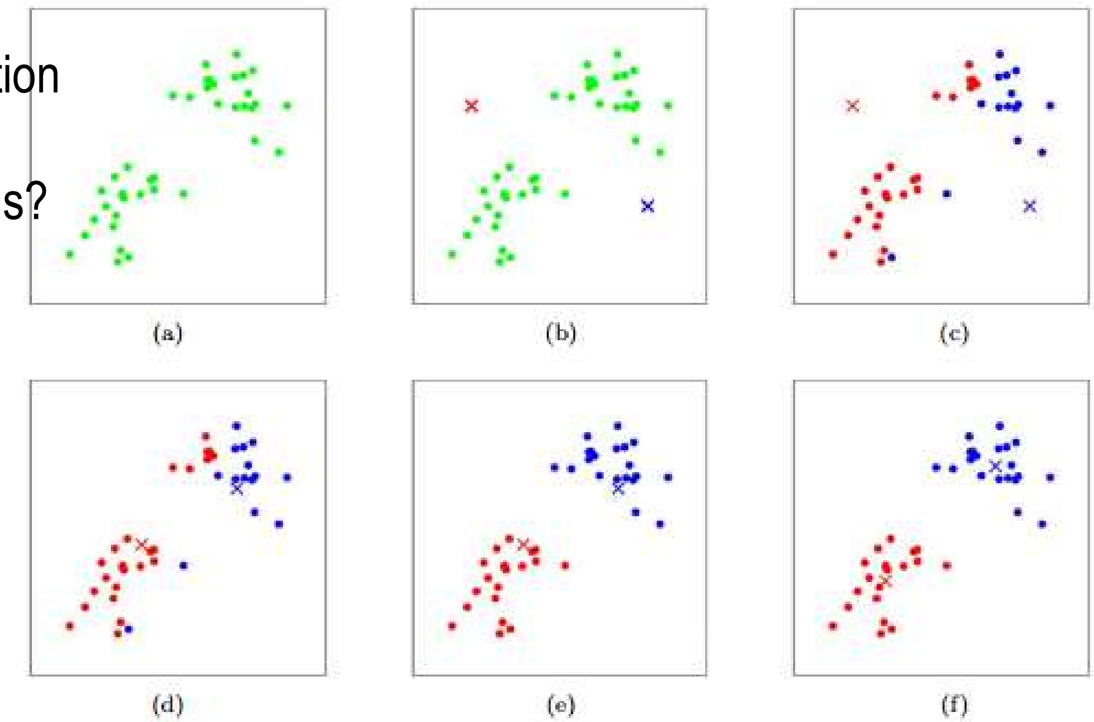
- Error = Variance + Bias^2 + Noise^2
  - **Intuition**: When combining multiple independent decisions, random errors cancel each other out
- **Two main methods** – Bagging and Boosting
- **Bagging**: Combines several learned models that are learned *independently* from **bootstrap replicates** of the same data set.
  - What does bagging remove? Bias or variance?
    - *Reduce variance without increasing bias by averaging*
  - What are the best ones to bag? Trees? Linear regression? Nearest neighbors?
- **Boosting**: Learns a weighted combination of classifiers. Focuses on the **incorrectly classified part** of the data set
  - What does boosting address? Bias or variance?
    - *Reduce bias and variance*
  - What is AdaBoost?
    - How does AdaBoost weight examples?
    - What are good weak learners?
  - How does boosting avoid overfitting?
    - Margins!
    - Remember, boosting is **not** immune to overfitting
- How do ensemble methods work with stable algorithms? Outliers?

Clustering: K-Means

- Euclidean distance is used as a metric and variance is used as a measure of cluster scatter
  - What are other distance measures?
- k is an input parameter: inappropriate choice of k may yield poor results.
- Convergence to local minimum
  - may produce counterintuitive results

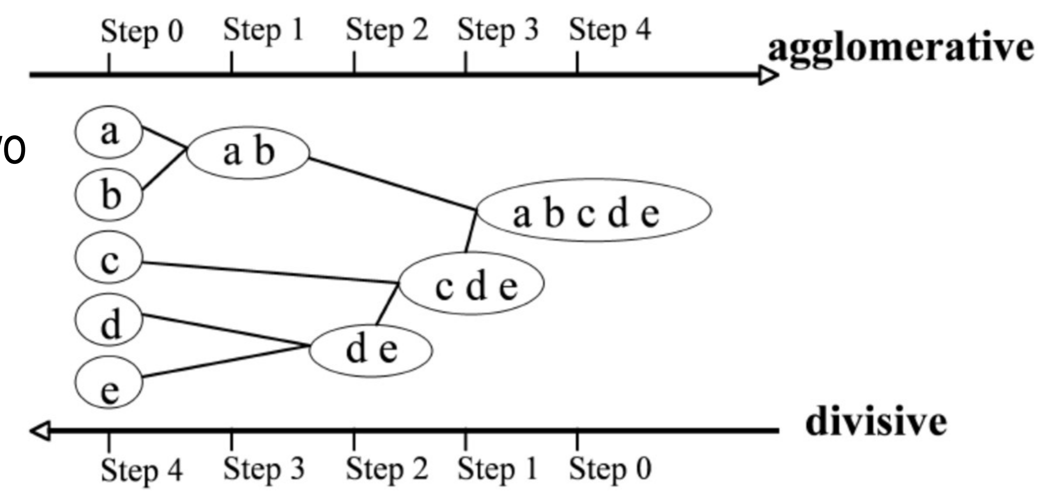
- What are responsibilities? What is the loss function of k-means clustering?
- What is the computational complexity of k-means?
- What type of clusters does k-means generate?

k-means Clustering
Given: Unlabeled data, $\boldsymbol{x}_i, i = 1, \ldots, n$
Initialize: Pick k random points as cluster centers $\mu_j, j = 1, \ldots, k$
while not converged do
- **Assign** data points $\boldsymbol{x}_i$ to **closest** cluster center μ_j
- **Update** the cluster centers μ_j to the mean (average) of the points assigned to that cluster
- if the assignments no longer change, **converged** = **true**



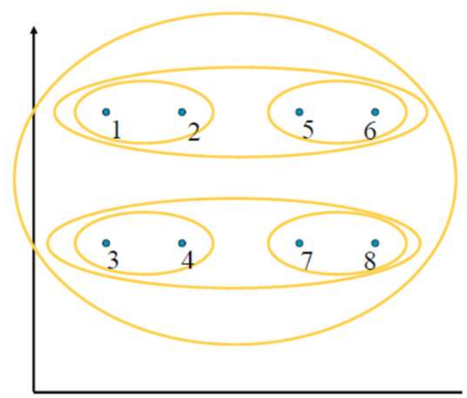
Hierarchical Clustering

- **Bottom-up (agglomerative)**: Recursively merge two groups with the smallest **between-cluster similarity**
- **Top-down (divisive)**: Recursively split a **least-coherent** (e.g. largest diameter) cluster

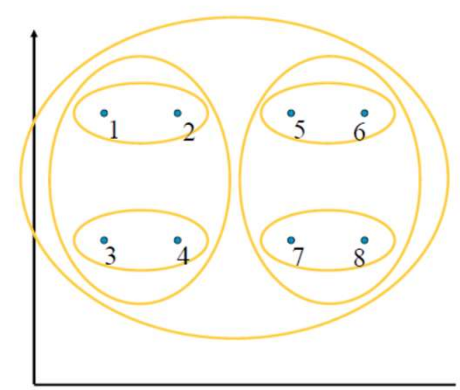


Differences between different types of linkages:
Closest pair (single-link clustering) tends to yield elongated clusters
Farthest pair (complete-link clustering) tends to yield rounder, more spherical clusters
Average of all pairs trades-off between single and complete link

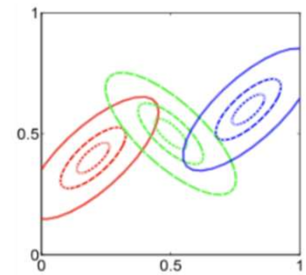
Closest pair
(single-link clustering)

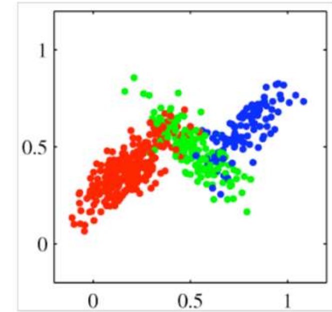


Farthest pair
(complete-link clustering)



Gaussian Mixture Models





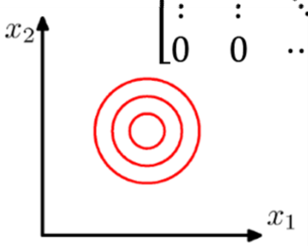
Probabilistic clustering (generative model):

Each cluster is associated with a **Gaussian distribution**. To **generate** data,

- randomly choose a cluster j with probability $P(y = j)$
  - *distribution over the clusters is modeled as a multinomial distribution*
- generate from the distribution of the j-th cluster:
  - *distribution over each cluster is modeled as a multivariate Gaussian distribution*

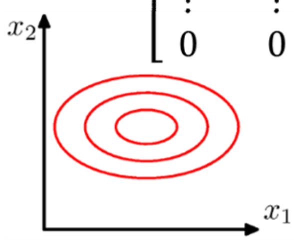
$$P(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$

$$\Sigma = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$



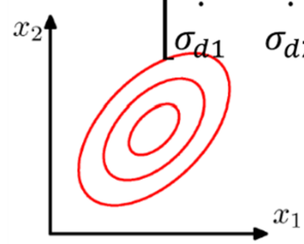
identity matrix

$$\Sigma = \begin{bmatrix} \sigma_{11} & 0 & \cdots & 0 \\ 0 & \sigma_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{dd} \end{bmatrix}$$

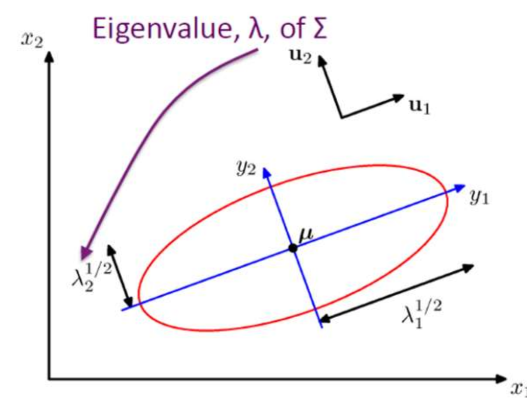


diagonal matrix: Gaussian Naïve Bayes

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_{dd} \end{bmatrix}$$



arbitrary positive semi-definite matrix: eigenvectors specify rotation, eigenvalues specify (relative) elongation

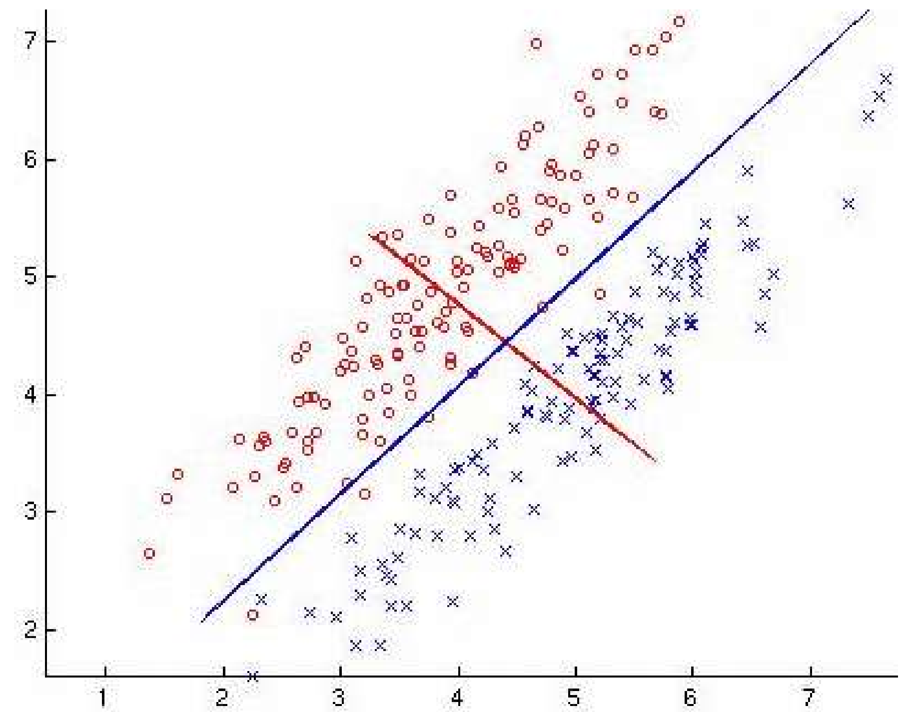


Solved using Expectation Maximization

- **soft** assignment of points to clusters
- maximize log likelihood.
- Can be used for non-spherical clusters with different probabilities

Principal Component Analysis

- Can be used to **reduce the dimensionality** of the data **while still maintaining a good approximation** of the sample mean and variance
- Can also be used for **selecting good features** that are combinations of the input features
- **Unsupervised** – just finds a good representation of the data in terms of combinations of the input features



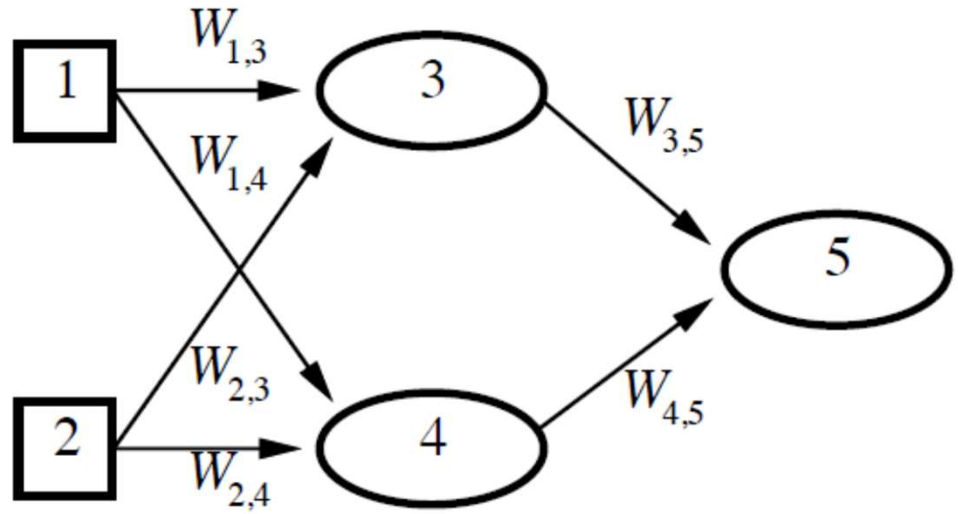
Principal Component Analysis identifies the principal compone in the **sample covariance matrix** of the data, $X^T X$ (*note that since our data is #examples (n) x features (d), the covariance matrix will be $d \times d$*)

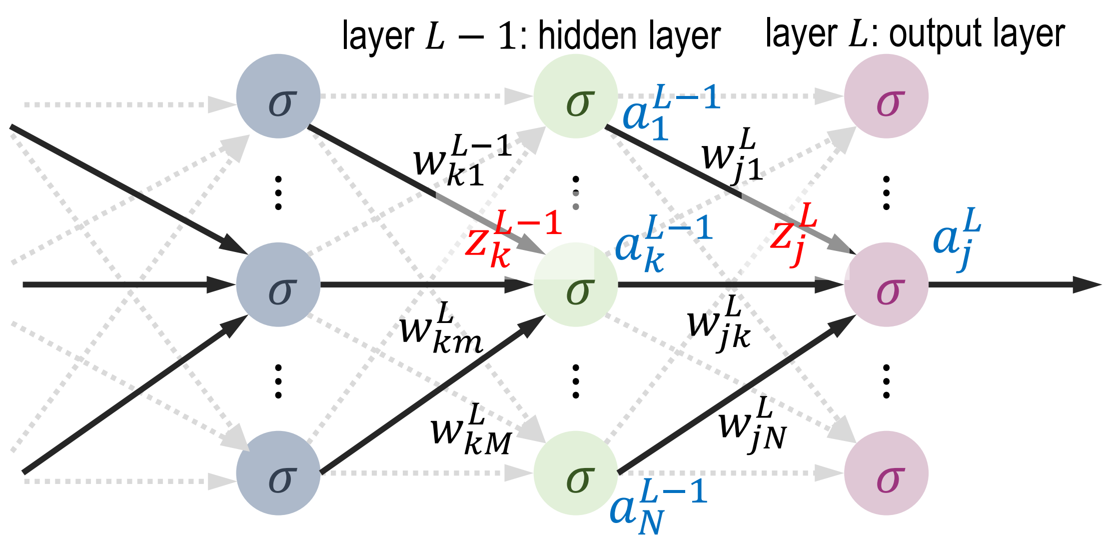
- PCA finds a set of **orthogonal vectors** that **best explain the variance of the sample covariance matrix**
- These are exactly the **eigenvectors** of the covariance matrix $X^T X$
- We can **discard the eigenvectors** corresponding to small magnitude eigenvalues to yield an approximation
- **Simple algorithm** to describe, MATLAB and other programming languages have built in **support** for eigenvector/eigenvalue computations

- How to geometrically identify principal components, projections and effectiveness for classification?
- How do we select an ideal number of principal components?
- What are the properties of eigenvalues and eigenvectors?

Neural Networks

- **Universal approximators**
  - Representing Boolean functions and logical formulas
  - Representing general mathematical functions
- Various propagation aspects of neural networks
  - functional representation of outputs and inputs
  - gradient expressions
  - activation functions and properties (sigmoid, tanh, RelU)
- **Trained using backpropagation**
  - forward propagation
  - backward propagation
- Overfitting in neural networks
  - regularization, dropout, other strategies
  - bias-variance tradeoff

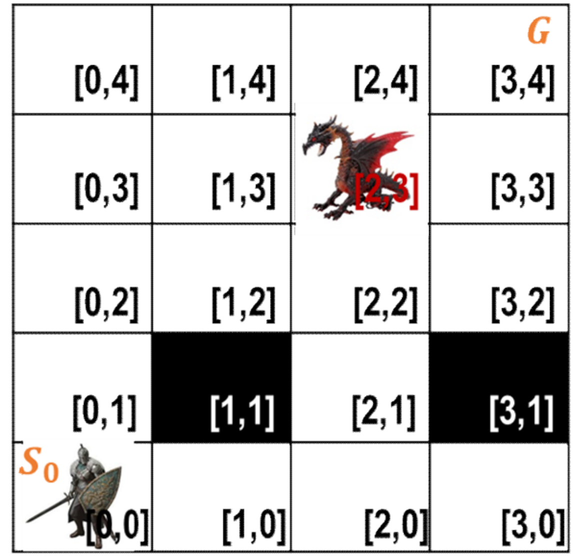




Reinforcement Learning

Modeling RL as a Markov Decision Process

- set of **states** S, set of **actions** A, **initial state** S_0
  - for grid world, can be cell coordinates
- **transition model** $P(s, a, s')$
  - $P(\,[1,1], \uparrow, [1,2]\,) \;=\; 0.8$
- **reward function** $r(s)$
  - $r(\,[3,4]\,) \;=\; +1$
- **discount factor**
- **learn a policy**: mapping from S to A
  - $\pi(s)$ or $\pi(s, a)$ (deterministic vs. stochastic)



Actions	Transition Probabilities
→ (right)	→ (60%), ↓ (40%)
↑ (up)	↑ (100%)
← (left)	← (100%)
↓ (down)	↓ (70%), ← (30%)

- **Value functions**
  - Which states and actions are better?
- Bellman equations, **Bellman optimality conditions**
- What is the difference between **value iteration** and **policy iteration**?
  - What is the value iteration update equation?
- What is the **exploration** vs. **exploitation** tradeoff?
- What is the **Q-learning**?
  - Why is Q-learning called model-free learning?