# CLUSTERING BEYOND K-MEANS

David Kauchak
CS 158 – Fall 2016

## Administrative

Final project

- Presentations on Tuesday
  - **4** minute max
  - 2-3 slides. *E-mail me by 9am on Tuesday*
  - What problem you tackled and results
- Paper and final code submitted on Wednesday

Final exam next week

## Midterm 2

| | Midterm 1 | Midterm 2 |
|---|---|---|
| Mean | 86% (37) | 85% (30) |
| Quartile 1 | 81% (35) | 80% (28) |
| Median (Q2) | 88% (38) | 87% (30.5) |
| Quartile 3 | 92% (39.5) | 93% (32.5) |

## K-means

Start with some initial cluster centers

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

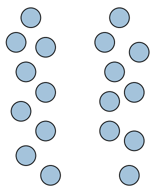
## Problems with K-means

Determining K is challenging

Hard clustering isn't always right

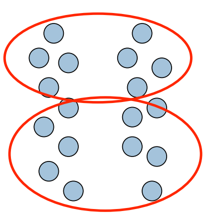
Greedy approach

## Problems with K-means



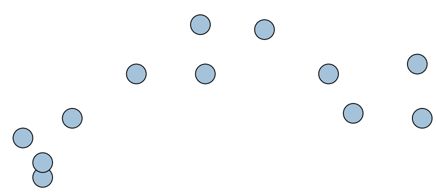
What would K-means give us here?

## Assumes spherical clusters

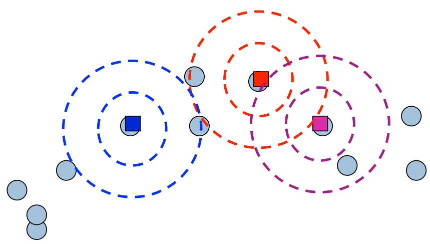


k-means assumes spherical clusters!

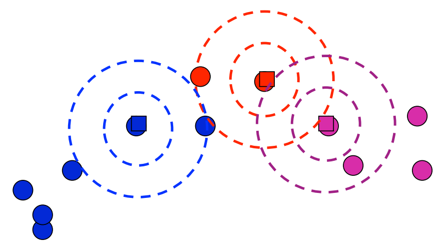
## K-means: another view



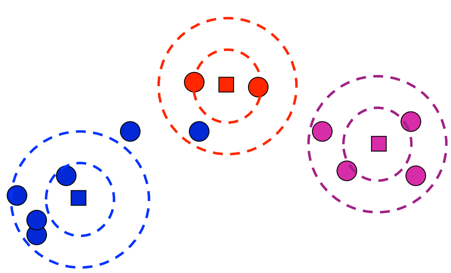
## K-means: another view



## K-means: assign points to nearest center



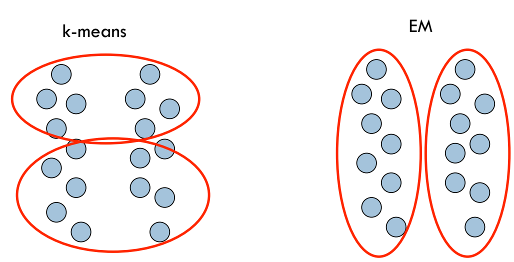
## K-means: readjust centers



Iteratively learning a collection of spherical clusters

## EM clustering: mixtures of Gaussians

Assume data came from a mixture of *Gaussians* (**elliptical data**), assign data to cluster with a certain *probability* (soft clustering)



## EM clustering

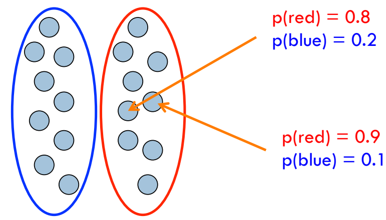
Very similar at a high-level to K-means

Iterate between assigning points and recalculating cluster centers

Two main differences between K-means and EM clustering:

1. We assume elliptical clusters (instead of spherical)
2. It is a "soft" clustering algorithm

## Soft clustering




## EM clustering

Start with some initial cluster centers

*Iterate:*

- **soft assign** points to each cluster

  Calculate: $p(\theta_c | x)$

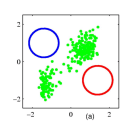
  the probability of each point belonging to each cluster

- recalculate the cluster centers

  Calculate new cluster parameters, $\theta_c$

  maximum likelihood cluster centers given the current soft clustering

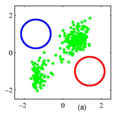
## EM example



Start with some initial cluster centers

Figure from Chris Bishop

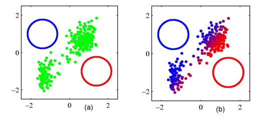
## Step 1: soft cluster points



Which points belong to which clusters (soft)?

Figure from Chris Bishop

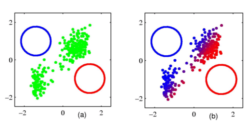
## Step 1: soft cluster points



Notice it's a soft (probabilistic) assignment

Figure from Chris Bishop

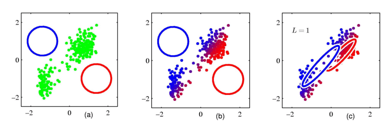
## Step 2: recalculate centers



What do the new centers look like?

Figure from Chris Bishop

## Step 2: recalculate centers



Cluster centers get a **weighted** contribution from points

Figure from Chris Bishop

## keep iterating…

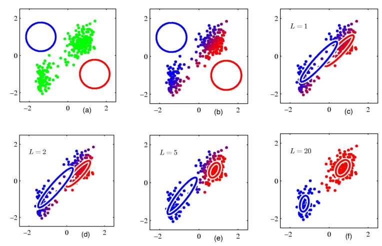


Figure from Chris Bishop

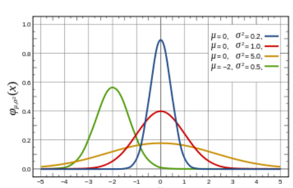
## Model: mixture of Gaussians

How do you define a Gaussian (i.e. ellipse)?
In 1-D?
In m-D?

## Gaussian in 1D

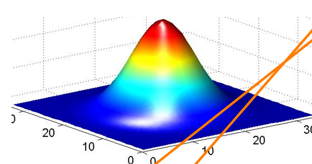
$$f(x;\sigma,\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

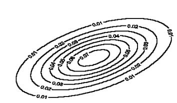
parameterized by the mean and the standard deviation/variance



## Gaussian in multiple dimensions

$$N[x;\mu,\Sigma] = \frac{1}{(2\pi)^{d/2}\sqrt{\det(\Sigma)}}\exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]$$

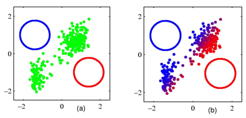




Covariance determines the shape of these contours

We learn the means of each cluster (i.e. the center) and the covariance matrix (i.e. how spread out it is in any given direction)

## Step 1: soft cluster points



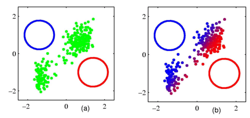
- soft assign points to each cluster

  Calculate: $p(\theta_c|x)$

  the probability of each point belonging to each cluster

How do we calculate these probabilities?

## Step 1: soft cluster points



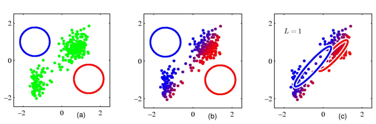
- soft assign points to each cluster

  Calculate: $p(\theta_c|x)$

  the probability of each point belonging to each cluster

Just plug into the Gaussian equation for each cluster!
(and normalize to make a probability)

## Step 2: recalculate centers



Recalculate centers:

calculate new cluster parameters, $\theta_c$

maximum likelihood cluster centers given the current soft clustering

How do calculate the cluster centers?

## Fitting a Gaussian

What is the "best"-fit Gaussian for this data?

10, 10, 10, 9, 9, 8, 11, 7, 6, …

Recall this is the 1-D Gaussian equation:

$$f(x;\sigma,\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

## Fitting a Gaussian

What is the "best"-fit Gaussian for this data?

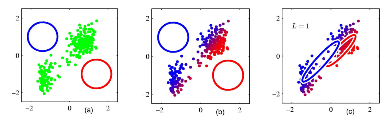
10, 10, 10, 9, 9, 8, 11, 7, 6, …

The MLE is just the mean and variance of the data!

Recall this is the 1-D Gaussian equation:

$$f(x;\sigma,\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

## Step 2: recalculate centers



Recalculate centers:

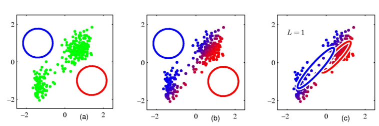
Calculate $\theta_c$

maximum likelihood cluster centers given the current

***soft clustering***

How do we deal with "soft" data points?

## Step 2: recalculate centers



Recalculate centers:

Calculate $\theta_c$

maximum likelihood cluster centers given the current

***soft clustering***

Use fractional counts!

## E and M steps: creating a better model

EM stands for Expectation Maximization

**Expectation**: Given the current model, figure out the expected probabilities of the data points to each cluster

$p(\theta_c|x)$ What is the probability of each point belonging to each cluster?

**Maximization**: Given the probabilistic assignment of all the points, estimate a new model, $\theta_c$

Just like NB maximum likelihood estimation, except we use fractional counts instead of whole counts

## Similar to *k*-means

Iterate:

Assign/cluster each point to closest center

Expectation: Given the current model, figure out the expected probabilities of the points to each cluster

$p(\theta_c|x)$

Recalculate centers as the mean of the points in a cluster

Maximization: Given the probabilistic assignment of all the points, estimate a new model, $\theta_c$

## E and M steps

**Expectation**: Given the current model, figure out the expected probabilities of the data points to each cluster

**Maximization**: Given the probabilistic assignment of all the points, estimate a new model, $\theta_c$

***Iterate:***

each iterations increases the likelihood of the data and is guaranteed to converge (though to a local optimum)!

## EM

EM is a general purpose approach for training a model when you don't have labels

Not just for clustering!

- K-means is just for clustering

One of the most general purpose unsupervised approaches

- can be hard to get right!

## EM is a general framework

Create an initial model, θ'

- Arbitrarily, randomly, or with a small set of training examples

Use the model θ' to obtain another model θ such that

$\sum_i \log P_\theta(data_i) > \sum_i \log P_{\theta'}(data_i)$ i.e. better models data (increased log likelihood)

Let θ' = θ and repeat the above step until reaching a local maximum

- Guaranteed to find a better model after each iteration

Where else have you seen EM?

## EM shows up all over the place

Training HMMs (Baum-Welch algorithm)

Learning probabilities for Bayesian networks

EM-clustering

Learning word alignments for language translation

Learning Twitter friend network

Genetics

Finance

Anytime you have a model and unlabeled data!

## Finding Word Alignments

… la maison … la maison bleue … la fleur …

… the house … the blue house … the flower …

In machine translation, we train from pairs of translated sentences

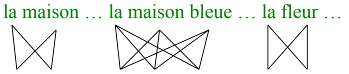
Often useful to know how the words align in the sentences

Use EM!

- learn a model of P(french-word | english-word)

## Finding Word Alignments

… la maison … la maison bleue … la fleur …



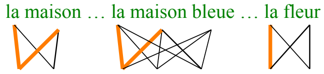
… the house … the blue house … the flower …

All word alignments are equally likely

All P(french-word | english-word) equally likely

## Finding Word Alignments

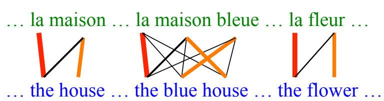
… la maison … la maison bleue … la fleur …



… the house … the blue house … the flower …

"la" and "the" observed to co-occur frequently,
so P(la | the) is increased.

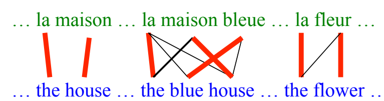
## Finding Word Alignments




"house" co-occurs with both "la" and "maison", but P(maison | house) can be raised without limit, to 1.0, while P(la | house) is limited because of "the"

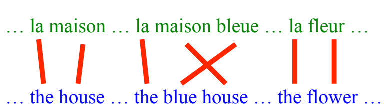
(pigeonhole principle)

## Finding Word Alignments




settling down after another iteration

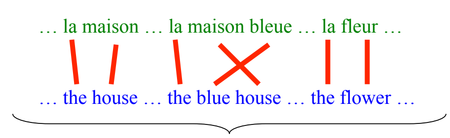
## Finding Word Alignments




**Inherent hidden structure revealed by EM training!**
For details, see

- "A Statistical MT Tutorial Workbook" (Knight, 1999).
  - 37 easy sections, final section promises a free beer.
- "The Mathematics of Statistical Machine Translation" (Brown et al, 1993)
- Software: GIZA++

## Statistical Machine Translation




P(maison | house ) = 0.411
P(maison | building) = 0.027
P(maison | manson) = 0.020
…

Estimating the model from training data

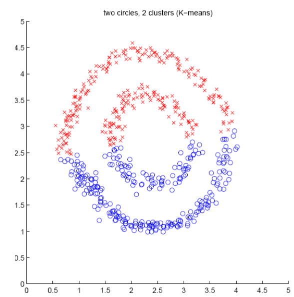
## Other clustering algorithms

K-means and EM-clustering are by far the most popular for clustering

However, they can't handle all clustering tasks

What types of clustering problems can't they handle?

## Non-Gaussian data

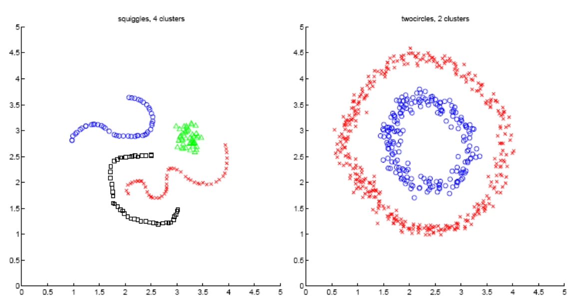



What is the problem?

Similar to classification: global decision (linear model) vs. local decision (K-NN)

Spectral clustering

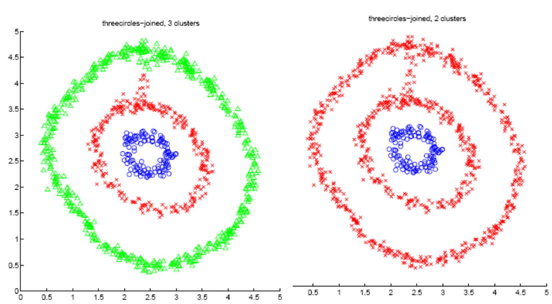
## Spectral clustering examples




(d) (e)

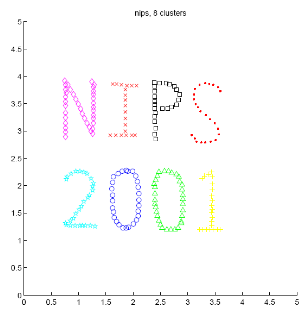
Ng et al On Spectral clustering: analysis and algorithm

## Spectral clustering examples




Ng et al On Spectral clustering: analysis and algorithm

## Spectral clustering examples




Ng et al On Spectral clustering: analysis and algorithm

## What Is A Good Clustering?

Internal criterion: A good clustering will produce high quality clusters in which:

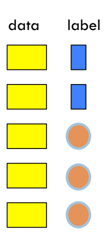
- the intra-class (that is, intra-cluster) similarity is high
- the inter-class similarity is low

How would you evaluate clustering?

## Common approach: use labeled data

Use data with known classes

- For example, document classification data



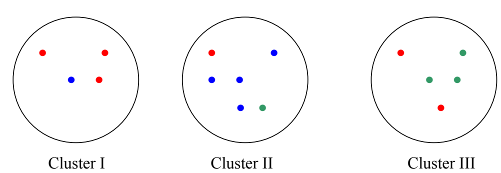

If we clustered this data (ignoring labels) what would we like to see?

Reproduces class partitions

How can we quantify this?

## Common approach: use labeled data

**Purity**, the proportion of the dominant class in the cluster




Cluster I: Purity = (max(3, 1, 0)) / 4 = 3/4

Cluster II: Purity = (max(1, 4, 1)) / 6 = 4/6

Cluster III: Purity = (max(2, 0, 3)) / 5 = 3/5

Overall purity?

## Overall purity

Cluster I: Purity = (max(3, 1, 0)) / 4 = 3/4

Cluster II: Purity = (max(1, 4, 1)) / 6 = 4/6

Cluster III: Purity = (max(2, 0, 3)) / 5 = 3/5

Cluster average:

$$\frac{\frac{3}{4}+\frac{4}{6}+\frac{3}{5}}{3}=0.672$$

Weighted average:

$$\frac{4*\frac{3}{4}+6*\frac{4}{6}+5*\frac{3}{5}}{15}=\frac{3+4+3}{15}=0.667$$

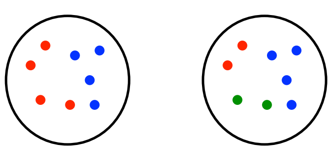
## Purity issues…

**Purity**, the proportion of the dominant class in the cluster

Good for comparing two algorithms, but not understanding how well a single algorithm is doing, why?

- Increasing the number of clusters increases purity

## Purity isn't perfect



Which is better based on purity?

Which do you think is better?

Ideas?

## Common approach: use labeled data

**Average entropy** of classes in clusters

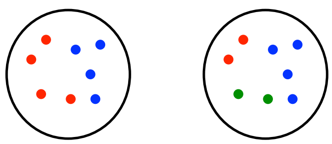
$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

where $p(class_i)$ is proportion of class *i* in cluster

## Common approach: use labeled data

**Average entropy** of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

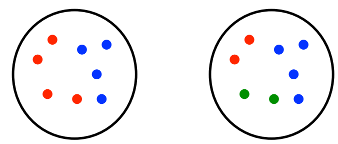


entropy?

## Common approach: use labeled data

**Average entropy** of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$



$-0.5\log 0.5 - 0.5\log 0.5 = 1$

$-0.5\log 0.5 - 0.25\log 0.25 - 0.25\log 0.25 = 1.5$

## Where we've been!

How many slides?

1367 slides

## Where we've been!

Our ML suite:

How many classes?

How many lines of code?

## Where we've been!

Our ML suite:

29 classes

2951 lines of code

## Where we've been!

Our ML suite:

- Supports 7 classifiers
  - Decision Tree
  - Perceptron
  - Average Perceptron
  - Gradient descent
    - 2 loss functions
    - 2 regularization methods
  - K-NN
  - Naïve Bayes
  - 2 layer neural network
- Supports two types of data normalization
  - feature normalization
  - example normalization
- Supports two types of meta-classifiers
  - OVA
  - AVA

## Where we've been!

Hadoop!

- 532 lines of hadoop code in demos

## Where we've been!

Geometric view of data

Model analysis and interpretation (linear, etc.)

Evaluation and experimentation

Probability basics

Regularization (and priors)

Deep learning

Ensemble methods

Unsupervised learning (clustering)