Chapter 29. Unsupervised Learning

Background

- Focusing on clustering
- Used less often than supervised learning
  - Harder to apply and measure success
  - The curse of dimensionality

Use Cases

- Pattern recognition
- Finding anomalies in data
- Topic modeling

Clustering model scalability

Table 29-1. Clustering model scalability reference

Model	Statistical recommendation	Computation limits	Training examples
k-means	50 to 100 maximum	Features x clusters < 10 million	No limit
Bisecting *k*-means	50 to 100 maximum	Features x clusters < 10 million	No limit
GMM	50 to 100 maximum	Features x clusters < 10 million	No limit
LDA	An interpretable number	1,000s of topics	No limit

Dataset

```
1  // Create Vector Assembler.
2
3  // in Scala
4  import org.apache.spark.ml.feature.VectorAssembler
5  val va = new VectorAssembler()
6  .setInputCols(Array("Quantity", "UnitPrice"))
7  .setOutputCol("features")
```

```
import org.apache.spark.ml.feature.VectorAssembler
va: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_89e2f29b104b
```

Command took 0.24 seconds -- by simpson145@live.missouristate.edu at 11/26/2018, 4:06:29 PM on My Cluster

Cmd 2

```
1  // Data Set for Chapter 29
2  val sales = va.transform(spark.read.format("csv")
3  .option("header", "true")
4  .option("inferSchema", "true")
5  .load("/databricks-datasets/definitive-guide/data/retail-data/by-day/*.csv")
6  .limit(50)
7  .coalesce(1)
8  .where("Description IS NOT NULL"))
9  sales.cache()
10
```

▸ (5) Spark Jobs

▸ sales: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 7 more fields]

```
sales: org.apache.spark.sql.DataFrame = [InvoiceNo: string, StockCode: string ... 7 more fields]
res14: sales.type = [InvoiceNo: string, StockCode: string ... 7 more fields]
```

Dataset Shown

```
1 sales.show()
```

▸ (1) Spark Jobs

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	features
580538	23084	RABBIT NIGHT LIGHT	48	2011-12-05 08:38:00	1.79	14075.0	United Kingdom	[48.0,1.79]
580538	23077	DOUGHNUT LIP GLOSS	20	2011-12-05 08:38:00	1.25	14075.0	United Kingdom	[20.0,1.25]
580538	22906	12 MESSAGE CARDS ...	24	2011-12-05 08:38:00	1.65	14075.0	United Kingdom	[24.0,1.65]
580538	21914	BLUE HARMONICA IN...	24	2011-12-05 08:38:00	1.25	14075.0	United Kingdom	[24.0,1.25]
580538	22467	GUMBALL COAT RACK	6	2011-12-05 08:38:00	2.55	14075.0	United Kingdom	[6.0,2.55]
580538	21544	SKULLS WATER TRA...	48	2011-12-05 08:38:00	0.85	14075.0	United Kingdom	[48.0,0.85]
580538	23126	FELTCRAFT GIRL AM...	8	2011-12-05 08:38:00	4.95	14075.0	United Kingdom	[8.0,4.95]
580538	21833	CAMOUFLAGE LED TORCH	24	2011-12-05 08:38:00	1.69	14075.0	United Kingdom	[24.0,1.69]
580539	21479	WHITE SKULL HOT W...	4	2011-12-05 08:39:00	4.25	18180.0	United Kingdom	[4.0,4.25]
580539	84030E	ENGLISH ROSE HOT ...	4	2011-12-05 08:39:00	4.25	18180.0	United Kingdom	[4.0,4.25]
580539	23355	HOT WATER BOTTLE ...	4	2011-12-05 08:39:00	4.95	18180.0	United Kingdom	[4.0,4.95]
580539	22111	SCOTTIE DOG HOT W...	3	2011-12-05 08:39:00	4.95	18180.0	United Kingdom	[3.0,4.95]
580539	21115	ROSE CARAVAN DOOR...	8	2011-12-05 08:39:00	1.95	18180.0	United Kingdom	[8.0,1.95]
580539	21411	GINGHAM HEART DO...	8	2011-12-05 08:39:00	1.95	18180.0	United Kingdom	[8.0,1.95]
580539	23235	STORAGE TIN VINTA...	12	2011-12-05 08:39:00	1.25	18180.0	United Kingdom	[12.0,1.25]
580539	23239	SET OF 4 KNICK KN...	6	2011-12-05 08:39:00	1.65	18180.0	United Kingdom	[6.0,1.65]
580539	22197	POPCORN HOLDER	36	2011-12-05 08:39:00	0.85	18180.0	United Kingdom	[36.0,0.85]
580539	22693	GROW A FLYTRAP OR...	24	2011-12-05 08:39:00	1.25	18180.0	United Kingdom	[24.0,1.25]
580539	22372	AIRLINE BAG VINTA...	4	2011-12-05 08:39:00	4.25	18180.0	United Kingdom	[4.0,4.25]
580539	22375	AIRLINE BAG VINTA...	4	2011-12-05 08:39:00	4.25	18180.0	United Kingdom	[4.0,4.25]

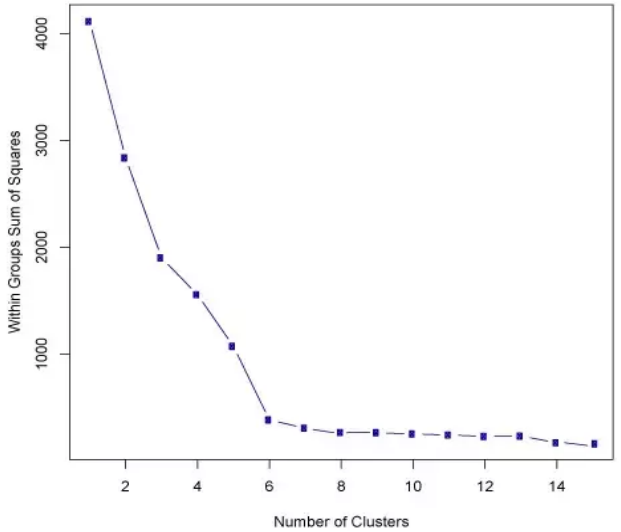
K-means

- One of the most popular clustering algorithms
- Centroid
- Terminate condition
  - Finite number of iterations
  - Centroid locations does not update anymore
- https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

How to choose the number of clusters?

- Elbow method
  - The idea is to find the K at which the SSE decreases most abruptly.

$$SSE = \sum_{i=1}^{K} \sum_{x \in c_i} dist(x, c_i)^2$$



Hyperparameters

Params	Description
K	number of clusters

Training Parameters

Params	Values	Description
initMode	random and k-means (default)	the starting locations of the centroids
initSteps	greater than 0 (default is 2.)	number of steps for K-means
maxIter	default is 20.	Total number of iterations over the data before stopping.
tol	default is 0.0001	threshold by which changes in centroids show that the model is optimized

Cmd 4

```scala
// Kmeans Example

// in Scala
import org.apache.spark.ml.clustering.KMeans
val km = new KMeans().setK(5)
println(km.explainParams())
val kmModel = km.fit(sales)
```

```scala
val summary = kmModel.summary
summary.clusterSizes // number of points
kmModel.computeCost(sales)
println("Cluster Centers: ")
kmModel.clusterCenters.foreach(println)
```

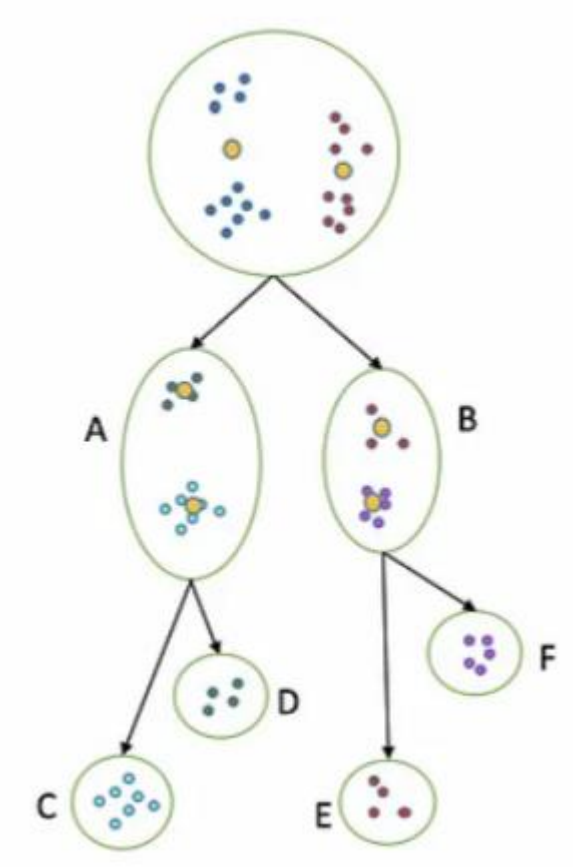
Metrics Summary

- val summary = kmModel.summary summary.clusterSizes // number of points
- kmModel.computeCost(sales)
- println("Cluster Centers: ")
- kmModel.clusterCenters.foreach(println)

Bisecting K-means

- Bisecting K-means is a top-down clustering method.
- It will start by creating a single group and then splitting that group into smaller groups in order to end up with the number of clusters specified by the user.
- *usually* a faster method than K-means and will yield different results.

Bisecting K-means



Hyperparameters

Params	Description
K	number of clusters

Training Parameters

Params	Values	Description
minDivisibleClusterSize	default is 1.0	The minimum number of points (if greater than or equal to 1.0) or the minimum proportion of points (if less than 1.0) of a divisible cluster
maxIter	default is 20.	Total number of iterations over the data before stopping.

Code Example

```
1  // Bisecting Kmeans
2
3  // in Scala
4  import org.apache.spark.ml.clustering.BisectingKMeans
5  val bkm = new BisectingKMeans().setK(5).setMaxIter(5)
6  println(bkm.explainParams())
7  val bkmModel = bkm.fit(sales)
```

▸ (17) Spark Jobs

```
featuresCol: features column name (default: features)
k: The desired number of leaf clusters. Must be > 1. (default: 4, current: 5)
maxIter: maximum number of iterations (>= 0) (default: 20, current: 5)
minDivisibleClusterSize: The minimum number of points (if >= 1.0) or the minimum proportion of points (if < 1.0) of a divisible cluster. (default: 1.0)
predictionCol: prediction column name (default: prediction)
seed: random seed (default: 566573821)
import org.apache.spark.ml.clustering.BisectingKMeans
bkm: org.apache.spark.ml.clustering.BisectingKMeans = bisecting-kmeans_5c958c0c6ad9
bkmModel: org.apache.spark.ml.clustering.BisectingKMeansModel = bisecting-kmeans_5c958c0c6ad9
```

Command took 1.66 seconds -- by simpson145@live.missouristate.edu at 11/26/2018, 4:06:30 PM on My Cluster

Bisecting K-means Summary

This includes information about the clusters created, as well as their relative sizes (number of examples):

```
// in Scala
val summary = bkmModel.summary
summary.clusterSizes // number of points
```

```
kmModel.computeCost(sales)
println("Cluster Centers: ")
kmModel.clusterCenters.foreach(println)
```

Bisecting K-Means Output

```scala
// Bisecting Kmeans Summary
// ERROR IN THE BOOK, FIXED HERE.

// in Scala
val summary = bkmModel.summary
summary.clusterSizes // number of points
bkmModel.computeCost(sales)
println("Cluster Centers: ")
bkmModel.clusterCenters.foreach(println)
```

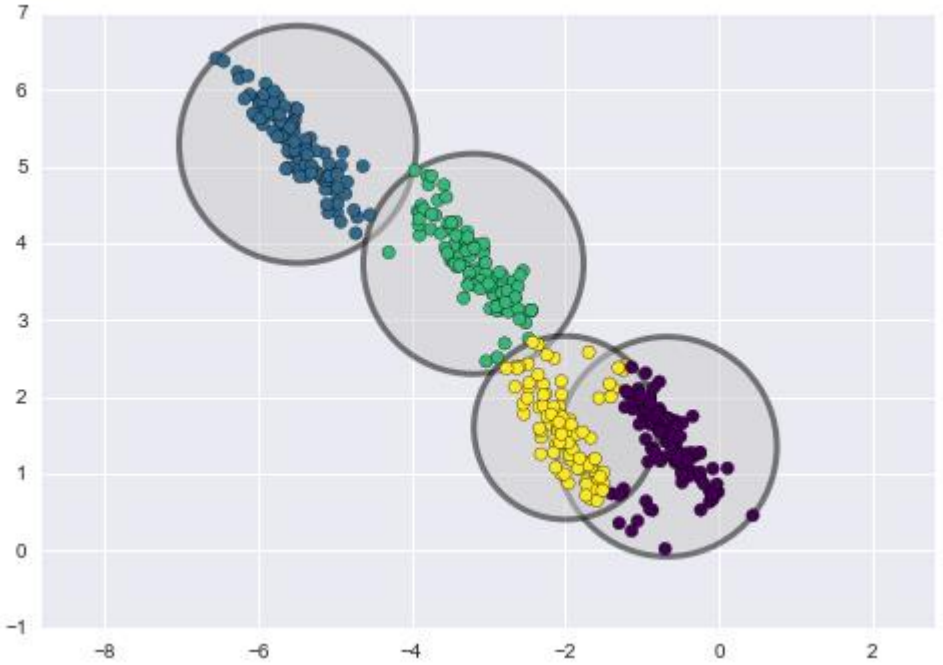
▸ (2) Spark Jobs

```
Cluster Centers:
[4.8125,4.095625]
[2.5,11.24375]
[10.923076923076923,1.1423076923076922]
[23.200000000000003,0.9560000000000001]
[44.0,1.1633333333333333]
summary: org.apache.spark.ml.clustering.BisectingKMeansSummary = org.apache.spark.ml.clustering.BisectingKMeansSummary@2920d042
```

Gaussian Mixture Models

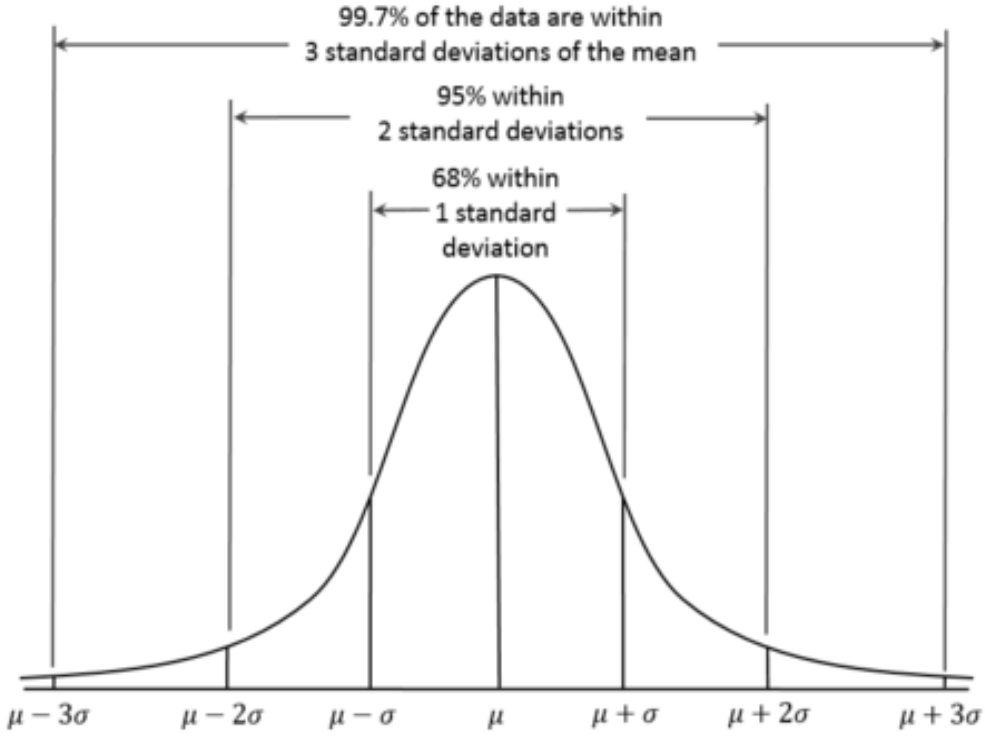
Weakness of the K-means algorithm

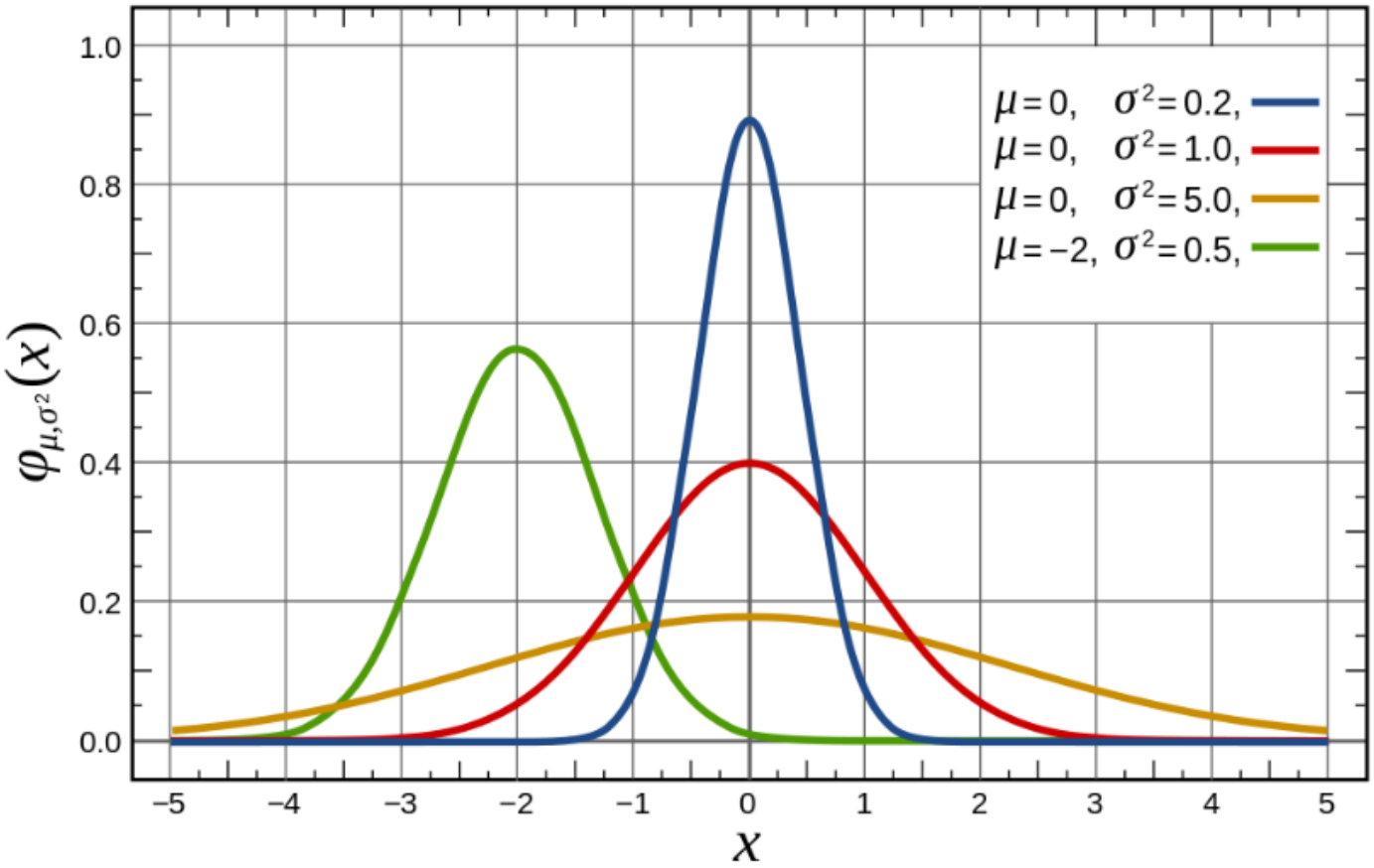
- Poorly handled overlapping element



Gaussian Mixture Models

- Clustering algorithm
- What is Gaussian distribution?
  - A function of a continuous random variable


99.7% of the data are within
3 standard deviations of the mean
95% within
2 standard deviations
68% within
1 standard
deviation
$\mu - 3\sigma$
$\mu - 2\sigma$
$\mu - \sigma$
μ
$\mu + \sigma$
$\mu + 2\sigma$
$\mu + 3\sigma$


$\mu = 0, \quad \sigma^2 = 0.2,$
$\mu = 0, \quad \sigma^2 = 1.0,$
$\mu = 0, \quad \sigma^2 = 5.0,$
$\mu = -2, \quad \sigma^2 = 0.5,$
$\varphi_{\mu,\sigma^2}(x)$
x
1.0
0.8
0.6
0.4
0.2
0.0
−5
−4
−3
−2
−1
0
1
2
3
4
5

Gaussian Mixture Models (Cont.)

- Probability distribution that consists of multiple probability distributions.
- Measures the **probability** that any point belongs to the given cluster
- Soft-boundaries

Hyperparameters

Params	Description
K	number of clusters

Training Parameters

Params	Values	Description
maxIter	default is 100.	Total number of iterations over the data before stopping.
tol	default is 0.0001	threshold by which changes in centroids show that the model is optimized

GMM Example

```
import org.apache.spark.ml.clustering.GaussianMixture
val gmm = new GaussianMixture().setK(5)
println(gmm.explainParams())
val model = gmm.fit(sales)
```

Gaussian Mixture Model Summary

```
val summary = model.summary
model.weights
model.gaussiansDF.show()
summary.cluster.show()
summary.clusterSizes
summary.probability.show()
```

References

- http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html
- https://jakevdp.github.io/PythonDataScienceHandbook/05.12-gaussian-mixtures.html
- https://github.com/llSourcell/Gaussian_Mixture_Models/blob/master/intro_to_gmm_%26_em.ipynb
- https://spark.apache.org/docs/latest/mllib-clustering.html#bisecting-k-means
- https://www.linkedin.com/pulse/initial-investigation-k-means-bisecting-algorithms-dave-blodgett/
- https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05