ClusteringSolutions

November 12, 2018

1 Solutions to the Clustering Task

1.1 Problem 1

In [1]:
```
Pkg.add("RDatasets")
using RDatasets
iris = dataset("datasets", "iris")
```

```
INFO: Package RDatasets is already installedINFO: METADATA is out-of-date  you may not have the
```

Out[1]:
```
150Œ5 DataFrames.DataFrame
 Row  SepalLength  SepalWidth  PetalLength  PetalWidth  Species

 1    5.1          3.5         1.4          0.2         "setosa"
 2    4.9          3.0         1.4          0.2         "setosa"
 3    4.7          3.2         1.3          0.2         "setosa"
 4    4.6          3.1         1.5          0.2         "setosa"
 5    5.0          3.6         1.4          0.2         "setosa"
 6    5.4          3.9         1.7          0.4         "setosa"
 7    4.6          3.4         1.4          0.3         "setosa"
 8    5.0          3.4         1.5          0.2         "setosa"
 9    4.4          2.9         1.4          0.2         "setosa"
 10   4.9          3.1         1.5          0.1         "setosa"
 11   5.4          3.7         1.5          0.2         "setosa"

 139  6.0          3.0         4.8          1.8         "virginica"
 140  6.9          3.1         5.4          2.1         "virginica"
 141  6.7          3.1         5.6          2.4         "virginica"
 142  6.9          3.1         5.1          2.3         "virginica"
 143  5.8          2.7         5.1          1.9         "virginica"
 144  6.8          3.2         5.9          2.3         "virginica"
 145  6.7          3.3         5.7          2.5         "virginica"
 146  6.7          3.0         5.2          2.3         "virginica"
 147  6.3          2.5         5.0          1.9         "virginica"
 148  6.5          3.0         5.2          2.0         "virginica"
 149  6.2          3.4         5.4          2.3         "virginica"
 150  5.9          3.0         5.1          1.8         "virginica"
```

In [4]:
```
using Clustering
features = Array(iris[:,[1,3,4]])'
result = kmeans( features, 3 )
```

Out[4]:
```
Clustering.KmeansResult{Float64}([6.81 5.006 5.89667; 5.7075 1.462 4.37167; 2.075 0.24
```

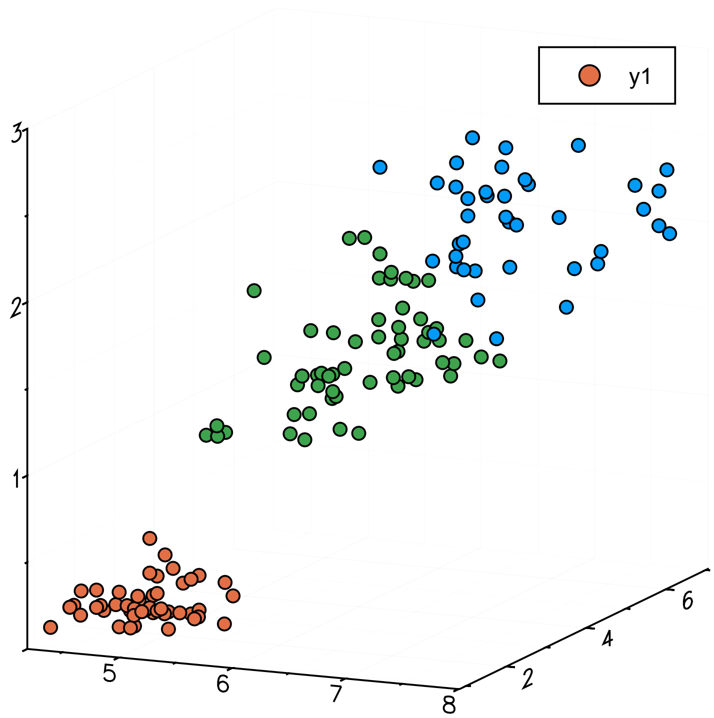
In [9]:
```
features'
```

Out[9]:
```
150Œ3 Array{Float64,2}:
 5.1  1.4  0.2
 4.9  1.4  0.2
 4.7  1.3  0.2
 4.6  1.5  0.2
 5.0  1.4  0.2
 5.4  1.7  0.4
 4.6  1.4  0.3
 5.0  1.5  0.2
 4.4  1.4  0.2
 4.9  1.5  0.1
 5.4  1.5  0.2
 4.8  1.6  0.2
 4.8  1.4  0.1

 6.0  4.8  1.8
 6.9  5.4  2.1
 6.7  5.6  2.4
 6.9  5.1  2.3
 5.8  5.1  1.9
 6.8  5.9  2.3
 6.7  5.7  2.5
 6.7  5.2  2.3
 6.3  5.0  1.9
 6.5  5.2  2.0
 6.2  5.4  2.3
 5.9  5.1  1.8
```

In [20]:
```
using Plots; gr()
scatter(features[1,:], features[2,:], features[3,:], color = result.assignments)
```

Out[20]:



1.2 Problem 2 (Advanced)

The main clustering package for julia, is unexpectedly, named Clustering.jl - It supports K-means, K-medoids, Affinity Propagation, DBSCAN - It also supports hierarchical clustering, but that is not currently in the docs.

You'll also want Distances.jl for all your distance metric needs. It is traditional with word2vec to use cosine distance.

1.2.1 Affinity Propagraion

If you set the availability right, it can get a breakdown where the ball-sports and clustered seperately from the other sports. Though you may have problems with some of the cities being classes as sports, as this word2vec repressentation was trained on a dump of wikipedia taken in 2014, and there are a lot of sports pages talking about the Athens and Beijing olypics.

2 First we loadup some data

For the the example presented here, we will use a subhset of Word Embedding, trained using Word2Vec.jl. These are 100 dimentional vectors, which encode syntactic and semantic information about words.

```
In [1]: using Embeddings
        countries = ["Afghanistan", "Algeria", "Angola", "Arabia", "Argentina", "Australia", "
```

```
        usa_cities = ["Albuquerque", "Atlanta", "Austin", "Baltimore", "Boston", "Charlotte", "
        world_capitals = ["Accra", "Algiers", "Amman", "Ankara", "Antananarivo", "Athens", "Bag
        animals = ["alpaca","camel","cattle","dog","dove","duck","ferret","goldfish","goose","
        sports = ["archery","badminton","basketball","boxing","cycling","diving","equestrian","

        words_by_class = [countries, usa_cities, world_capitals, animals, sports]
        all_words = reduce(vcat, words_by_class)
        embedding_table = load_embeddings(Word2Vec; keep_words = all_words)
        @assert Set(all_words) == Set(embedding_table.vocab)

        embeddings = embedding_table.embeddings
        all_words = embedding_table.vocab
        classes = map(all_words) do word
            findfirst(col -> word  col, [countries, usa_cities, world_capitals, animals, sports
        end;
```

```
In [3]: display(all_words)
        embeddings
```

```
185-element Array{String,1}:
 "China"
 "field"
 "Iraq"
 "Washington"
 "India"
 "South"
 "football"
 "Canada"
 "London"
 "England"
 "Australia"
 "Japan"
 "Pakistan"

 "taekwondo"
 "goldfish"
 "Las"
 "llama"
 "pentathlon"
 "alpaca"
 "Bogotá"
 "yak"
 "Antananarivo"
 "Brasília"
 "Usa"
 "Yaoundé"
```

```
Out[3]: 300Œ185 Array{Float32,2}:
```

```
 -0.0269732   -0.129657    0.0646846      -0.0523314    0.0111792
  0.0499904    0.0643991   0.042243         0.0302394    0.0545651
  0.0401002    0.0540952  -0.0221116        0.132239     0.0819807
  0.0305697   -0.128798    0.0202965       -0.0261657    0.099548
 -0.0471133    0.0152411  -0.0963669        0.00556217   0.0638811
 -0.0837969   -0.143395   -0.0346525      -0.011281     -0.0351346
  0.0557447    0.0097672   0.0396028       -0.0651793   -0.0915629
 -0.0168133   -0.129657   -0.118808        -0.0783405   -0.0460476
 -0.0240961    0.0119675  -0.0321773       -0.0463776   -0.0883688
  0.0126774    0.0403568   0.0676548        0.0152764    0.0282141
 -0.0517886    0.115918    0.0103132      -0.0523314   -0.0294119
 -0.00867639  -0.0755616  -0.00198014       0.100903     0.077722
  0.0906301    0.0807135  -0.000959131     -0.0927551    0.00367649

  0.00429324   0.0414301   0.105608        -0.0357232   -0.0670751
  0.0697708   -0.0157778   0.0295371        0.0457508   -0.0417889
  0.0517886    0.0328435  -0.0339924       0.0305528   -0.0117781
  0.0321881    0.104327   -0.0260719        0.0288293    0.000827626
  0.0561044   -0.139961   -0.041583        -0.0927551   -0.0123104
  0.0293109    0.0202857   0.0653447       -0.0460642   -0.0129758
  0.00233768   0.0880121  -0.0518137       -0.0399536   -0.0243547
 -0.0139362    0.0244717   0.0301972      -0.055465      0.0377963
 -0.0676129    0.0547392  -0.0587442       -0.0311795   -0.097951
  0.0507097   -0.0328435  -0.0184813       -0.0463776   -0.0351346
  0.0345258   -0.0139531   0.102307         0.0429306   -0.0473785
 -0.0293109    0.0377808   0.0358076        0.0374467    0.0686722
```

In [6]:
```julia
using Clustering
using Distances
using LinearAlgebra

similarity = 1f0 .- pairwise(CosineDist(), embeddings)
availability = 0.01*ones(size(similarity,1))
# tweaking availability is how you control number of clusters
# it is the diagonal of the similarity matrix
similarity[diagind(size(similarity)...)] = availability
aprop = affinityprop(similarity)
```

Out[6]:
```
AffinityPropResult([10, 28, 29, 34, 40, 52, 56, 62, 63, 77    114, 123, 124, 139, 143,
```

In [8]:
```julia
for (cluster_ii, examplar_ind) in enumerate(aprop.exemplars)
    println("-"^32)
    println("Exemplar: ", all_words[examplar_ind])
    cluster_member_inds = findall(assignments(aprop).==cluster_ii)
    println(join(getindex.([all_words], cluster_member_inds), ", "))
end
```

```
--------------------------------
Exemplar: England
```

```
London, England, Britain, Ireland, Wales
--------------------------------
Exemplar: Atlanta
Detroit, Houston, Atlanta, Philadelphia, Dallas, Charlotte, Indianapolis, Memphis, Columbus, Na
--------------------------------
Exemplar: Baghdad
Iraq, Afghanistan, Baghdad, Kabul, Cairo, Beirut, Riyadh, Amman
--------------------------------
Exemplar: Seattle
Washington, Chicago, Boston, Seattle, Baltimore, Portland, Milwaukee, Sacramento
--------------------------------
Exemplar: soccer
field, football, basketball, golf, soccer, hockey, tennis, rugby, wrestling, volleyball, handba
--------------------------------
Exemplar: Moscow
Russia, Moscow, Ukraine, Baku, Minsk, Kyiv, Tashkent
--------------------------------
Exemplar: Thailand
China, Japan, Singapore, Vietnam, Indonesia, Thailand, Malaysia, Philippines, Myanmar, Bangkok
--------------------------------
Exemplar: Argentina
South, Canada, Australia, Germany, Spain, Italy, Brazil, Argentina, Venezuela, Colombia, Peru,
--------------------------------
Exemplar: Tehran
Iran, Tehran, Damascus, Ankara
--------------------------------
Exemplar: Bangladesh
India, Pakistan, Bangladesh, Nepal, Uzbekistan, Dhaka
--------------------------------
Exemplar: Seoul
Beijing, Tokyo, Korea, Seoul, Pyongyang, Jakarta, Taipei, Hanoi
--------------------------------
Exemplar: Uganda
Nigeria, Sudan, Kenya, Ghana, Uganda, Ethiopia, Tanzania, Nairobi, Mozambique, Kampala
--------------------------------
Exemplar: Morocco
France, Egypt, Yemen, Algeria, Morocco, Arabia, Rabat
--------------------------------
Exemplar: Santiago
Madrid, Manila, Santiago, Havana, Lima, Francisco, San, Caracas, Quito, Las, Bogotá
--------------------------------
Exemplar: Albuquerque
Mexico, Denver, Phoenix, Austin, Fresno, Tucson, Omaha, Mesa, Vegas, Albuquerque
--------------------------------
Exemplar: rat
mouse, duck, rat, goose, pigeon, ferret, goldfish
--------------------------------
Exemplar: dove
```

shooting, diving, dove

Exemplar: Budapest

Paris, Poland, Rome, Berlin, Athens, Vienna, Stockholm, Warsaw, Budapest, Bucharest

----

Exemplar: rowing

swimming, cycling, sailing, fencing, gymnastics, rowing, equestrian, triathlon, kayak, pentath

----

Exemplar: Kinshasa

Congo, Khartoum, Pretoria, Accra, Madagascar, Algiers, Luanda, Kinshasa, Antananarivo, Brasíli

----

Exemplar: judo

boxing, archery, badminton, weightlifting, judo, taekwondo

Exemplar: llama

dog, cattle, camel, llama, alpaca, yak