Tutorial on k-means, neural networks and support vector machines

Introduction

- An outline on k-means, back-propagation and support vector machines are given.
- A brief description of each approach is given, with some algorithmic layouts provided.
- For the practicals, please provide your own code and answers

Note that text in blue are web references that can be clicked.

k-means: Data Mining Algorithm

- k-means is used to cluster or group data, such as drawing a circle around data that looks similar
- Formally, to partition data into said groups minimize a cluster sum of squares $\Sigma_{j=1}^{k}\Sigma_{i=1}^{n}||x_i^j - c_j||^2$ where c is the center point or centroid.
- Note that k-means is not an optimal clustering technique since clustering cannot be well defined.
- Also note that the *CalculateCentroid* and *UpdateCluster* steps are optimal.
- For the practicals, please provide your own code and answers

k-means algorithm

A visualisation of how the centroid move during the update centroid step can be seen in Centroid Update Visualisation.

Please note that the centroid of each group or cluster of data defines each group associated with that centroid. Said differently, the centroid c_1 is closest to the data associated with centroid c_1.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)

Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

foreach $c_i \in C$ **do**
  $c_i \leftarrow e_j \in E$ (e.g. random selection)
end
foreach $e_i \in E$ **do**
  $l(e_i) \leftarrow argminDistance(e_i, c_j) j \in \{1 \ldots k\}$
end

$changed \leftarrow false$;
$iter \leftarrow 0$;
repeat
  foreach $c_i \in C$ **do**
    $UpdateCluster(c_i)$;
  end
  foreach $e_i \in E$ **do**
    $minDist \leftarrow argminDistance(e_i, c_j)\ j \in \{1 \ldots k\}$;
    if $minDist \neq l(e_i)$ **then**
      $l(e_i) \leftarrow minDist$;
      $changed \leftarrow true$;
    end
  end
  $iter++$;
until $changed = true\ and\ iter \leq MaxIters$;

Figure 1: k-means retrieved from `wikibooks.com` in pseudo-code[1]

```
#!/usr/bin/python3

import random
studentno = int(input('Please enter your student number'))
random.seed(studentno % 10000)
col1 = [random.randint(0,studentno) for i in range(0,200)]
random.seed(studentno % 1000)
col2 = [random.randint(0,studentno) for i in range(0,200)]
sdata = open('data.txt', 'w')
[print("%s,%s" % (str(col1[i]), str(col2[i])), file=sdata) for i in range(0,200)]
```

Figure 2: Data generation algorithm for the practical hand-in

Practical 1: Implement k-means

Please read to the end for handing in information.

- Task 1

  Implement k-means Figure 1 from page in python. Python via Jupyter Notebook can be used for this and subsequent tasks.

  Note: installing extra python modules can be done via `pip install <module> --user` if python works on the command line. Otherwise consult the web for more information. Regardless, the following packages are of use: pandas, sklearn, numpy, matplotlib, seaborn. See eg 'pip3 install seaborn –user' on linux.

- Task 2

  Use your student number and run the code snippet in Figure 2 on your student number to generate the text file listed. Use the data in the *generated* text file to find the centroids.

Read the next page

Practical 1: Implement k-means

Please read to the end for handing in information.

- Task 3

  Explain in words your code and why and how you implemented your code the way you did.

- Task 4 Explain in words how and why the elbow method is used and where the method is useful.

- For handing in:

  Please hand in: your implementation in python of k-means, the coordinates describing your *student number generated data* centroid, your word-based explanation, and an explanation of the elbow method.

Neural Networks: Back-propagation

- For this tutorial use Back-propagation Background for a basic understanding of back-propagation [2]
- In short, back-propagation neural networks are multi-layer recurrent network perceptron based

Back-propagation Outline

Basic outline for backprop:

- An *activation function* "triggers" when a sufficiently large value passes through the activation function in the perceptron
- The first layer is the *input* layer
- The calculated activation function values are then collected, added in some way and then fed through to the next *hidden* layer of the perceptron
- Multiple hidden layers are acceptable
- The last layer is the *output* layer that gives a prediction value

Back-propagation Algorithm Notes

- Please make sure to understand the basics of the activation function before attempting to use the algorithm
- Deep learning is not the focus of what we are considering
- Key to the activation function:
  - A distance measure
  - A usable error based on using the distance measure on the data that provide a predictable output in some way
  - Some calculable minimum used for local and global mimima optimization

Back-propagation Algorithm Implementation Notes

- Backprop consists of:
  - A feed-forward component
  - Backpropagation to the output layer; ”trains” the activation function
  - Backpropagation to the hidden layer; ”trains” the activation function
  - Weight updates (actual training of the network)
- If the computed *error* is small then the algorithm ceases training the weights of the network

Practical 2: Implement Back-propagation

Please read to the end for handing in information.

- Task 1

  Explain in words the back-propagation neural network algorithm with a bulleted list of steps in an outline fashion, and explain the reason for the last data column in Figure 3.

- Task 2 Adapt your explanation of Practical 2, Task 1 to a pseudo-code implementation

- Task 3 Implement your pseudo-code and explain the basics of weight updates. Make sure the weight updates are clearly explained. There is no need to fully implement back-propagation for this specific task; please make sure that your code is clean and easy to read. Matrix algebra operations libraries may be assumed for this task in the style of `numpy` or `scikit`.

- Task 4 Use your python back-prop implementation in (say) jupyter notebook with the generated data from Figure 3 to generate five *scores* and include these in a separate file called `back-prop-scores.txt` with your submission. Please use the code from Complete Example to generate the cross-validation scores with your use of `scikit` (there should be five values) [3].

- For handing in:

  Please hand in: A text file explaining the bulleted backprop algorithm, your pseudo-code implementation in python of back-prop, delta-update code for the neural network weights with properly commented code (your word-based explanation), your data generated file with the back-prop-scores.txt values

```python
#!/usr/bin/python3

import random

from sklearn.cluster import KMeans as kmeans
import numpy as np
import matplotlib.pyplot as plot

DATAPOINTS=20

studentno = 1234567
studentno = int(input('Please enter your student number: '))
random.seed(studentno % 10000)
col1 = [random.randint(0,studentno) for i in range(0,DATAPOINTS)]
random.seed(studentno % 1000)
col2 = [random.randint(0,studentno) for i in range(0,DATAPOINTS)]
random.seed(studentno % 100)
col3 = [random.randint(0,studentno) for i in range(0,DATAPOINTS)]
random.seed(studentno % 10)
col4 = [random.randint(0,studentno) for i in range(0,DATAPOINTS)]
random.seed(studentno % 100000)
col5 = [random.randint(0,studentno) for i in range(0,DATAPOINTS)]
col6 = [i % 3 for i in range(0,DATAPOINTS)]
sdata = open('data.txt', 'w')
[print("%s,%s,%s,%s,%s,%s" % (str(col1[i]), str(col2[i]), str(col3[i]),
 str(col4[i]), str(col5[i]), str(col6[i])), file=sdata) for i in range(
0,DATAPOINTS)]
```

Figure 3: Data generation algorithm for Practical 2 hand-in

References

[1] Wikibooks, "Data mining algorithms in r/clustering/k-means."

[2] R. Rojas, *Neural Networks - A Systematic Introduction*. Springer-Verlag, New York, 1996.

[3] J. Brownlee, "How to code a neural network with backpropagation in python (from scratch)," 2019. `https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/`.