Syllabus for the Datamining Class

Gener Avilés R

2017-04-14

Contents

1 Introduction **5**

2 Principal Components Analysis **7**
2.1 What does PCA do? 7
2.2 PCA Step by Step 7
2.3 Pros and Cons of PCA 10

3 Canonical Correlation Analysis (CCA) **13**
3.1 What is CCA? 13
3.2 Assumptions for CCA 13
3.3 Objectives of CCA 13
3.4 Terms used in the context of a CCA analysis 14
3.5 Interpreting canonical variates 14
3.6 Considerations when working with CCA 14
3.7 Limitations of CCA 14

4 Self Organizing Maps (SOM) **17**
4.1 Other names: 17
4.2 Generalities 17
4.3 Penfield´s Homunculus 18
4.4 Algorithm 18
4.5 Example 20

Chapter 1

Introduction

This course is taught in the ***Maestría y Doctorado en Ciencias e Ingeniería*** (MyDCI) programm of Facultad de Ingeniería, Arquitectura y Diseño of Universidad Autónoma de Baja California in the Ensenada Campus.

The course is taught by Dr. María de los Ángeles Cosío León.

Chapter 2

Principal Components Analysis

2.1 What does PCA do?

This methods tries to explain the correlation structure of a set of predictor variables using a smaller set o linear combinations of these variables called ***components***, note that components are not variables, rather indicators of linear combinations between variables. Given a dataset with m variables a set of k linear combinations can be used to represent it (meaning that k contains almost as much information as the m variables), also $k << m$.

2.2 PCA Step by Step

2.2.1 1. Getting the dataset and things ready.

Before starting the process of dimensionality reduction one should make sure the data is standardized, this is done to avoid bised in the results by values to large or to small when compared to each other.

2.2.2 2. Centering the points

- The **standardization process** is acomplished when the mean for each variable = 0 and the standard deviation = 1. The following formula can be followed to acomplish this process:

$$Z_i = \frac{(X_i - \mu_i)}{\sigma_{ii}}$$

Where: μ_i equals the mean of X_i and σ_{ii} equals the standard deviation of X_i.

- If the values are given as a set of points the process can be acomplished with the following formula:

$$x_{i,a} = x_{i,a} - \mu_a$$

This move will facilitate the calculations down the road.

2.2.3 3. Compute covariance ($\sigma_{X,Y}$) matrix

The **covariance** is a measure of the degree to which two variables vary together. Positive covariance indicates that when one variable increases, the other tends to increase. Negative covariance indicates that when one variable increases, the other tends to decrease. The covariance measure **is not scaled**.

In a $2x2$ matrix:

$$\begin{vmatrix} 2.0 & 0.8 \\ 0.8 & 0.6 \end{vmatrix}$$

Since the mean (μ) is equal to $\emptyset$ thanks to centering the values in the previous step, the formula to calculate the covariance of the values in the matrix is:

$$cov(x_1, x_2) = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}x_{i,2}$$

The way to interpret *covariance* is to understand it's results as information about how one attribute changes as the other one changes.

It is important to remember that, if we multiply a vector by the covariance matrix or $\sum$ the resulting vector will turn towards the direction of the variance.

Changing the units of measure would change the results, this is an inconvenience and is addressed by calculating the ***correlation coefficient*** r_{ij}:

r_{ij} scales the covariance by each of the standard deviations:

$$r_{ij} = \frac{\sigma_{ij}^2}{\sigma_{ii}\sigma_{jj}}$$

The r_{ij} gives us a value with reference to know how much of a correlation exists between two variables.

2.2.4 4. Eigenvectors + Eigenvalues

Define a **new set of dimentions** by:

1. Taking the dataset and looking for the direction of the data, looking to draw a line in which, along it, there is the **greatest amount of variance** σ^2 in the data, this line will be called the **principal component 1 (PC1)**.

$$\sigma^2 = \frac{\sum(X-\mu)^2}{N} \text{or} \sigma^2 = \frac{\sum X^2}{N} - \mu^2$$

In the previous formula σ^2 is defined as the sum of the squared distances of each term in the distribution from the mean (μ^2) divided by the number of terms in the distribution (N). In simple words: σ^2 measures **how far a set of random numbers are spread out from their mean**.

2. Once PC1 is determined, it will established the next dimension by drawing an ***orthogonal*** (perpendicular) line in relation to PC1, the exact area where the line will be drawn is determined by the same process of finding the gratest σ^2 of the remaining data, once this is done PC2 is ready.

3. This will be done iteratively until all the dimensions (d) of the dataset are covered and components (m) are generated for every single d.

- The first $m << d$ components become m new dimensions.
  - Coordinates from every datapoint will be changed to these "new" dimensions.
- **Greatest variability** is pursued to maintain the smoothness assumption of dimensions.

Eigenvectors and eigenvalues are mathematically expressed as:

$$A\overrightarrow{v} = \lambda\overrightarrow{v}$$

Where A represents transformation, $\overrightarrow{v}$, a vector, also known as **eigenvector**, that comes out of the matrix being analysied and λ, a scalar value also known as **eigenvalue**.

Principal components = eigenvectors with largest eigenvalues.

2.2.4.1 Finding Eigenvalues and Eigenvectors

In order to exemplify the process of finding these values and vector steps are presented for a $2x2$ matrix, but this can be done with any matrix of nxn dimensions following the rules of matrix algebra.

To begin with the example we will declare a matrix:

$$A = \begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix}$$

Now the steps:

1. **Multiply an nxn identity matrix by the scalar λ:** $IA\lambda$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} * \lambda = \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}$$

2. **Substract the identity matrix multiple from matrix A:** $A - \lambda I$

$$\begin{bmatrix} 7 & 3 \\ 3 & -1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix}$$

3. **Find the determinant of the matrix obtained in previous step:** $det(A - \lambda I)$

$$det \begin{bmatrix} 7-\lambda & 3 \\ 3 & -1-\lambda \end{bmatrix} = (7-\lambda)(-1-\lambda) - (3*3)$$

$$= -7 - 7\lambda + \lambda + \lambda^2 = -16 - 6\lambda + \lambda^2$$
$$= \lambda^2 - 6\lambda - 16$$

4. **Solve for the values of λ that satisfy the equation** $det(A - \lambda I) = 0$ Solving for $\lambda^2 - 6\lambda - 16 = 0$ will result in:

$$(\lambda - 8)(\lambda + 2) = 0$$

   Therfore $\lambda_1 = 8$ and $\lambda_2 = -2$ **these are the eigenvalues for matrix A.**

5. **Solve for the corresponding vector to each λ**

Solving for $\lambda = 8$, in this process we will call the matrix with substituted values: B.

$$\begin{bmatrix} 7-(8) & 3 \\ 3 & -1-(8) \end{bmatrix} = \begin{bmatrix} -1 & 3 \\ 3 & -9 \end{bmatrix}$$

We will assume the following $B\overline{X} = 0 \therefore$

$$\begin{bmatrix} -1 & 3 \\ 3 & -9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Applying row reduction $3R_1 + R_2 = R_2$ to:

$$\left[\begin{array}{cc|c} -1 & 3 & 0 \\ 3 & -9 & 0 \end{array}\right] = \left[\begin{array}{cc|c} -1 & 3 & 0 \\ 0 & 0 & 0 \end{array}\right] \therefore -x_1 + 3x_2 = 0$$

From the previous operation we obtain $3x_2 = x_1$, at this point we can choose a value for any x, we will go for $x_2 = 1$ to keep it simple.

$$3x_2 = x_1 \therefore 3(1) = x_1 \therefore x_1 = 3$$

Now we know that the eigenvalue $\lambda = 8$ $ corresponds to the eigenvector $\overline{X} = (3, 1)$.

Solving for $\lambda - 2$, generating matrix C.

$$C = \begin{bmatrix} 7 - (-2) & 3 \\ 3 & -1 - (-2) \end{bmatrix}$$

$C\overline{X} = 0 \therefore$

$$\begin{bmatrix} 9 & 3 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Applying row reduction $3R_2 - R_1 = R_1$:

$$\left[\begin{array}{cc|c} 0 & 0 & 0 \\ 3 & 1 & 0 \end{array}\right] \therefore 3x_1 + x_2 = 0$$

Assigning $x_1 = 1$:

$$x_2 = -3x_1 \therefore x_2 = -3(1)$$

The eigenvalue $\lambda = 8$ corresponds to the eigenvector $\overline{X} = (1, -3)$

2.2.5 5. Pick $m < d$ eigenvectors with highest eigenvalues.

In other words, usually the **2** eigenvectors with the highest scalars, or λ, will be selected to represent the whole dataset as Principal Component 1 and Principal Component 2.

2.2.6 6. Project datapoints to those eigenvectors.

One or the algoritm has to project the datapoints to these new set of dimensions so they can be analyized.

2.2.7 7. Perform analysis as needed according to study.

2.3 Pros and Cons of PCA

This algorithm, as all, is better suited for specific circumstances and performs poorly in others. The following list trys to summarize this idea:

2.3.0.1 Pros

- Reduction in size of data.
- Allows estimation of probabilites in high-dimensional data.
- It renders a set of components that are uncorrelated.

2.3.0.2 Cons

- It has a high computational cost, therefore it cannot be applied to very large datasets.
- Not good when working with fine-grained classes.

Chapter 3

Canonical Correlation Analysis (CCA)

3.1 What is CCA?

- Seeks the weighted linear composit for each variate (sets of D.V. or I.V.) to maximize the overlap in their distributions.
- Labeling of DV and IV is arbitrary. The procedure looks for relationships and not causation.
- Goal is to **maximize the correlation** (not the variance extracted as in most other techniques).
- Lacks specificity in interpreting results, that may limit its usefulness in many situations.

CCA helps us answer the questions:

- ***What is the best way to understand how the variables in two sets are related?*** , mathematically speaking:
  - ***what linear combinations of the set X variables (u) and the set Y variables (t) will maximize their correlation?***

3.1.1 Canonical R (R_c)

It represents the overlapping variance between two variates which are linear composites of each set of variables.

3.2 Assumptions for CCA

- Multiple continuous variables for DVs and IVs or categorical with dummy coding.
- Assumes **linear relationship** between any two variables and between variates.
- Multivariate normality is necessary to perform CCA.
- Multicollinearity in either variate **confounds** interpretation of canonical results.

3.3 Objectives of CCA

- Determine the magnitude of the relationships that may existe between two sets of variables.

- Derive a variate(s) for each set of criterion and predictor variables such that the variate(s) of each set is maximally correlated.
- Explain the nature of whatever relationships exist between the sets of criterion and predictor variables.
- Seek the max correlation of shared variance between the two sides of the equation.

3.4 Terms used in the context of a CCA analysis

- **Canonical correlation:** Correlation between two sets; the largest possible correlation that can be found between linear combinations.
- **Canonical variate:** The linear combinations created from the IV set and DV set.
- **Canonical weights:** weights used to create the liniear combinations; interpreted like regression coefficients.
- **Canonical loadings:** correlations between each variable and its variate; interpreted like loadings in PCA.
- **Canonical cross-loadings:** Correlation of each observed independent or dependent variable with opposite canonical variate.

3.5 Interpreting canonical variates

- Canonical weights
  - larger wight contributes more to the function.
  - negative weight indicates an inverse relationship with other variables.
  - always look out for multicollinearity, it can skew the whole analysis.
- Canonical Loadings.
  - A direct assessment of each variable´s contribution to its respective canonical variate.
  - Larger loadings are interpreted as more important to deriving the canonical variate.
  - Correlation between the original variable and its canonoical variate.
- Canonical Cross-Loadings
  - Measure of correlation of each original D.V. with the independent canonical variate.
  - Direct assessment of the relationship between each D.V. and the independent variate.
  - Provides a more pure measure of the dependent and independent variable relationship.
  - Preferred approach to interpretation.

3.6 Considerations when working with CCA

- Small samples sizes may have an adverse effect.
- Suggested minimun sample size = 10 * # of values.
- Selection of variables to be included:
- Select them with domain knowledge or theoretical basis.
- Inclusion of irrelevant or deletion of relevant variables may adversely affect the entire canonical solution.
- All I.V.s must be interrelated and all D.V.s must be interrelated.
- Composition of D.V. and I.V. variates is critical to producing practical results.

3.7 Limitations of CCA

- R_c (canonical R) reflects only the variance shared by the linear composites, not the variances extracted from the variables.
- Canonical weights are subject to a great deal of instability, particularly when there is multicollinearity.

- Interpretation difficult because rotation is not possible.
- Precise statistics have not been developed to interpret canonical analysis.

This chapter is under construction.

Figure 3.1:

Chapter 4

Self Organizing Maps (SOM)

4.1 Other names:

- Self-Organizing Feature Map (SOFM).
- Kohonen Map.
- Kohonen Networks.

4.2 Generalities

- **S**elf **O**rganizing **M**aps belong to the family of **Artificial Neural Networks**.
- In the subgroup of **Unsupervised Learning**,
- To function they use a **competitive learning strategy** (winner takes all).
- They are considered to be a **non-linear** implementation of the Principal Components Analysis (PCA) algorithm.

Self Organizing Maps (SOM) were first described by Teuvo Kohonen (Kohonen, 1995), others have extended his work and modified SOMs to tackle specific problems.

"The Self-Organizing Map is inspired by postulated feature maps of neurons in the brain comprised of feature-sensitive cells that provide ordered projections between neuronal layers, such as those that may exist in the retina and cochlea. For example, there are acoustic feature maps that respond to sounds to which an animal is most frequently exposed, and tonotopic maps that may be responsible for the order preservation of acoustic resonances." (Brownlee, 2011) Different sensory inputs are maped into corresponding areas of the cerebral cortex in an orderly way. The map generated in the cerebral cortex is called a **topographic map** and it has two very important properties, (Bullinaria, 2015):

1. At each stage of representation, or processing, each piece of incoming information is kept in its proper context/neighborhood.
2. Neurons dealing with closely related pieces of information are kept close together so that they can interact via short synaptic connections.

Following the principles observed in the sensory input processing by neurological structures , the previous two properties should be kept in an artificial intelligence algorithm looking to reproduce this phenomenon. In shorter words: the principle of topographic map formation is the escence of this process, where:

> "The spatial location of an output neuron in a topographic map corresponds to a particular domain or feature drawn from the input space." (Bullinaria, 2015)

4.3 Penfield´s Homunculus

4.4 Algorithm

4.4.1 Conditions

- The data given to the algorithm must be continuous.
- The algorithm will perform better when fed high dimensional data.
- This algorithm will help in the process of

4.4.2 Steps of the algorithm

This section is based on the "Introduction to Neural Computation" course, taught by Dr John A Bullinaria at the University of Birmingham.

Self Organizing Maps follow the principle of **self organization**, composed of the following 4 steps:

1. Initialization.
2. Competition.
3. Cooperation.
4. Adaptation.

4.4.2.1 Initialization

All the connections weights (neurons or centroids) are initialized with small random values. Other authors use the the principal component (eigenvector) of the data in this step.

4.4.2.2 Competition:

For each input pattern, the neurons compute their respective values of a discriminant function which provides the basis for competition. The particular neuron with the smallest value of the discriminant function is declared the winner.

- Given the input space is D dimensional, the input patterns can be expressed as:

$$x = \{x_i : i = 1, ..., D\}$$

- The connection weights between the input units i and the neurons j in the computation layer can be expressed as:

$$w_j = \{w_{ji} : j = 1, ..., N; i = 1, ..., D\}$$

```
Where N is the total number of neurons.
```

Discriminant Function

Now that the dimensions of the data and how they are related to the lattice in the SOM are clearer, the next natural step in the process is to define the discriminant function, SOM bases this on the squared Euclidean distance:

$$d_j(x) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

By doing this, the neuron whose weight vector comes closest to the input vector (most similar to it) is "declared" the winner.

This process allows us to **map a continuous space given by the input to a discrete output space of neurons** by a process of competition and a "winner takes all" approach.

4.4.2.3 Cooperation:

The winning neuron determines the spatial location of a topological neighborhood of excited neurons, thereby providing the basis for cooperation among neighbouring neurons. This **lateral interaction** within a set of neurons has been observed by neurobiologist in human and other brains. When one neuron fires, its closest neighbours tend to get excited more than those further away. This phenomenon allows the formation of a **topological neighborhood** that decays with distance in relation to the excited neuron known as the **Best Matching Unit (BMU)** in a SOM.

If S_ij is the lateral distance between neurons i and j on the grid of output neurons, the topological neighborhood will then be expressed as:

$$T_{ij,I(x)} = exp\left(\frac{-S^2_{j,I(x)}}{2\sigma^2}\right)$$

Where $I(x)$ = the index of the winning neuron.

This has the following key properties:

- Maximal at the BMU.
- It decreases monotonically to zero as the distance goes to infinity.
- It is translation invariant, therefore independent of the location of the BMU.

Important note:

The process of self organization will work best if the size of σ of the neighborhood decreases with time. A popular approach has been an exponential decay:

$$\sigma(t) = \sigma_0 exp(\frac{-t}{\tau_\sigma})$$

4.4.2.4 Adaptation:

The excited neurons decrease their individual values of the discriminant function in relation to the input pattern through suitable adjustment of the associated connection weights, such that the response of the winning neuron to the subsequent application of a similiar input pattern is enhanced.

In a topographic neighborhood not only the winning neuron gets its weights updated, but its neighbors will have their weights updated as well, although by not as much as the winner itself. In practice, the appropriate weight update equation is:

$$\Delta w_{ji} = \eta(t) * T_{j,I(x)}(t) * (x_i - w_{ji})$$

Where there is a time (t, epoch) dependent learning rate $\eta(t) = \eta_0 exp(\frac{-t}{\tau_\eta})$, and the updates are applied for all the training patterns x over many epochs.

Each learning weight update will move the weight vectors w_i of the BMU (winning neuron) and it's neighbors towards the input vector x. Repeated presentations of the training data thus leads to topological ordering.

4.5 Example

We will take the dataset `wines` found in the package `kohonen`, which shows the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

4.5.1 Summary of the dataset

```
library(kohonen)
data(wines)
summary(wines)
```

```
##     alcohol        malic acid        ash        ash alkalinity
##  Min.   :11.03   Min.   :0.74   Min.   :1.360   Min.   :10.60
##  1st Qu.:12.36   1st Qu.:1.60   1st Qu.:2.210   1st Qu.:17.20
##  Median :13.05   Median :1.87   Median :2.360   Median :19.50
##  Mean   :12.99   Mean   :2.34   Mean   :2.366   Mean   :19.52
##  3rd Qu.:13.67   3rd Qu.:3.10   3rd Qu.:2.560   3rd Qu.:21.50
##  Max.   :14.83   Max.   :5.80   Max.   :3.230   Max.   :30.00
##    magnesium       tot. phenols     flavonoids    non-flav. phenols
##  Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300
##  1st Qu.: 88.00   1st Qu.:1.740   1st Qu.:1.200   1st Qu.:0.2700
##  Median : 98.00   Median :2.350   Median :2.130   Median :0.3400
##  Mean   : 99.59   Mean   :2.292   Mean   :2.023   Mean   :0.3623
##  3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.860   3rd Qu.:0.4400
##  Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600
##     proanth         col. int.         col. hue        OD ratio
##  Min.   :0.410   Min.   : 1.280   Min.   :0.480   Min.   :1.270
##  1st Qu.:1.250   1st Qu.: 3.210   1st Qu.:0.780   1st Qu.:1.930
##  Median :1.550   Median : 4.680   Median :0.960   Median :2.780
##  Mean   :1.587   Mean   : 5.055   Mean   :0.957   Mean   :2.604
##  3rd Qu.:1.950   3rd Qu.: 6.200   3rd Qu.:1.120   3rd Qu.:3.170
##  Max.   :3.580   Max.   :13.000   Max.   :1.710   Max.   :4.000
##     proline
##  Min.   : 278.0
##  1st Qu.: 500.0
##  Median : 672.0
##  Mean   : 745.1
##  3rd Qu.: 985.0
##  Max.   :1680.0
```

4.5.2 Training Phase

```
library(kohonen)
#SOM grid/lattice
som_grid <- somgrid(xdim = 6, ydim = 6, topo = "hexagonal")

#Standardazing dataset
wines.sc <- scale(wines)

#Setting the seed
```

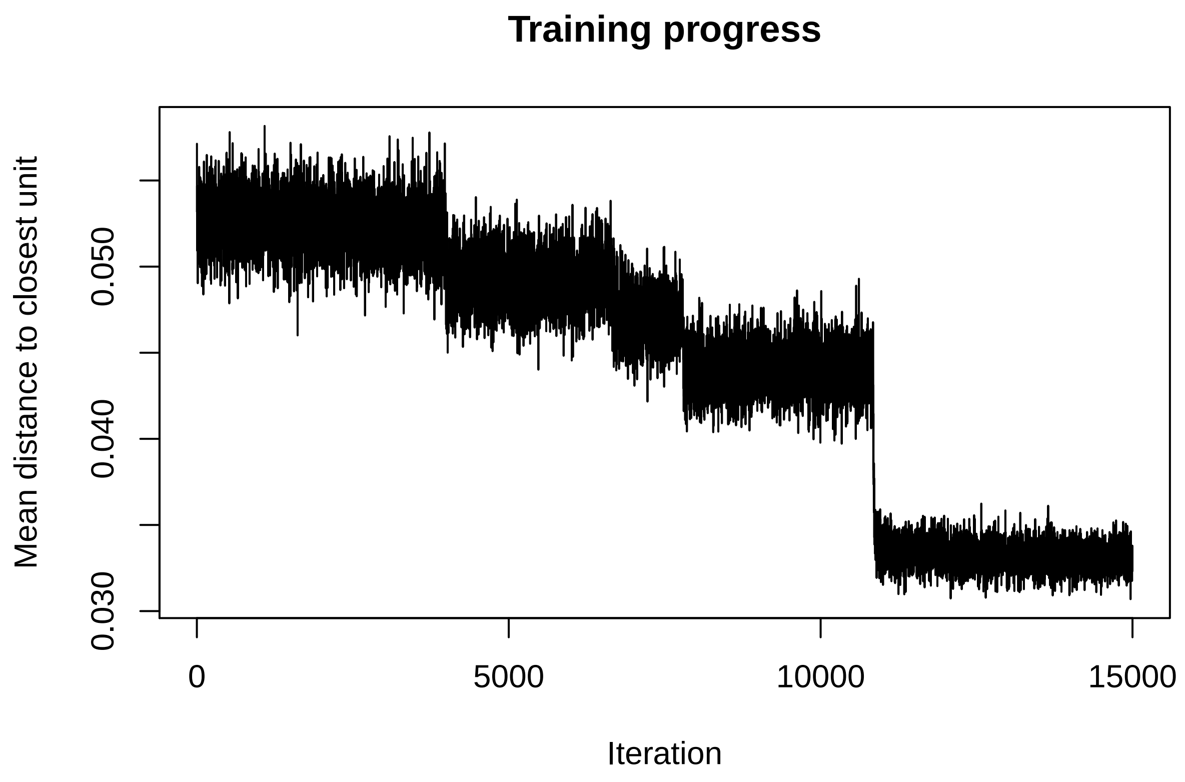
```r
set.seed(123)

#Training SOM
som_model <- som(wines.sc,
                 grid = som_grid,
                 rlen = 15000,
                 alpha = c(0.05,0.01),
                 keep.data = TRUE)
```

4.5.3 Visualization

4.5.3.1 Training progress

Once a stable plateu is shown in the plot, one can assume that the Euclidean distance from each node's weights to the samples in the database represented by that node (or neuron) has reached it's lower value. If the curve in the plot is continually decreasing more iterations will be needed.

```r
plot(som_model, type = "changes")
```



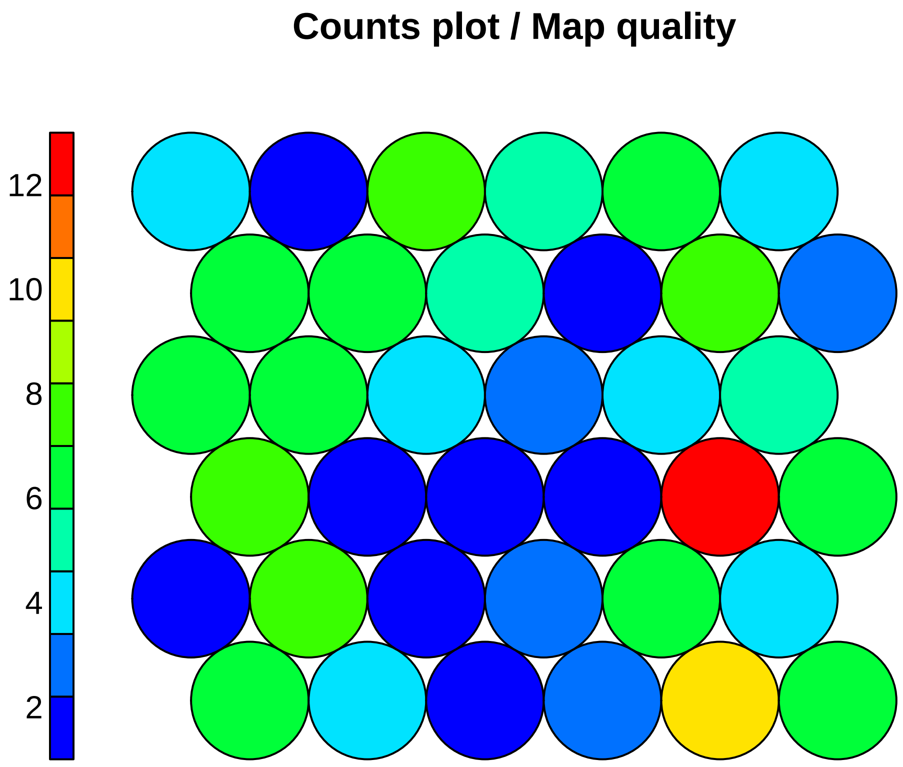
4.5.3.2 Node Counts

This steps helps us visualize how many samples are mapped to each neuron on the map. This can be used as an indicator of **map quality**. Some key points:

- Sample distribution should be relatively uniform (colors or shades similar) throughout the map.

- Large values in a map area are a sign that a larger map is needed.
- Empty neurons are a sign that the map is to large.
- It is costumary to aim for 5 to 10 samples per neuron for an ideal map.

```r
source('coolBlueHotRed.R')
plot(som_model,
     type = "count",
     palette.name = coolBlueHotRed,
     main = "Counts plot / Map quality")
```

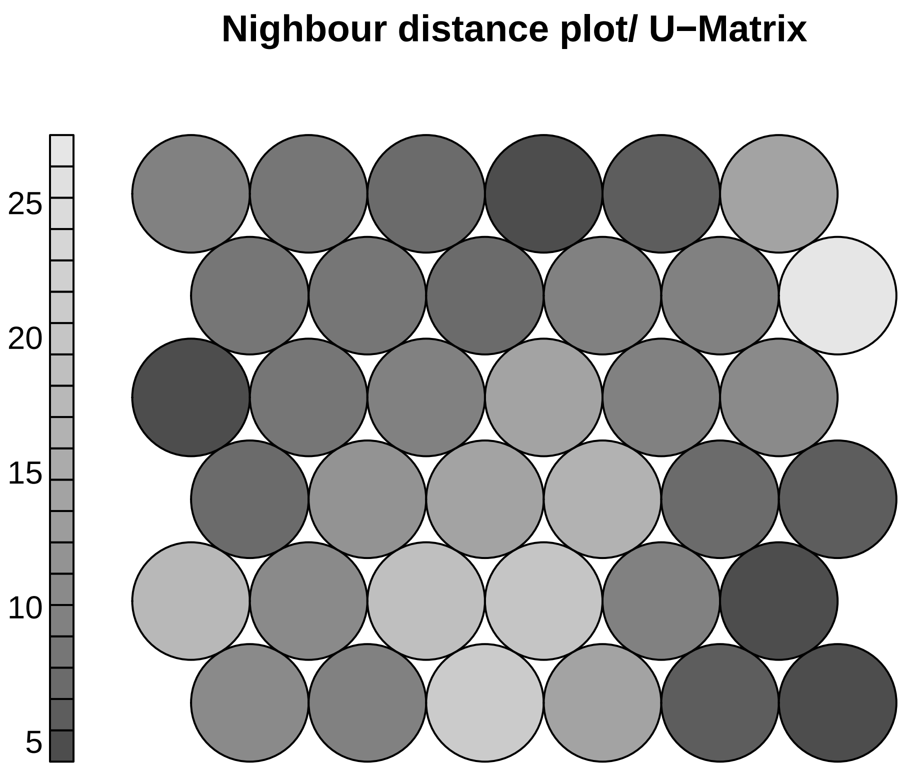


4.5.3.3 Neighbor Distance

This is also called the ***U-Matrix*** and it is a visualization of the distance between each neuron and it's neihgbors. Key points:

- It's usually visualized using a grayscale.
- Areas of low neighbor distance indicate groups of similar neurons.
- Areas of high nighbor distance indicate that neurons are more dissimilar.
- The U-Matrix can be used to identify clusters.

```r
plot(som_model,
     type = "dist.neighbours",
     palette.name = gray.colors,
     main = "Nighbour distance plot/ U-Matrix")
```



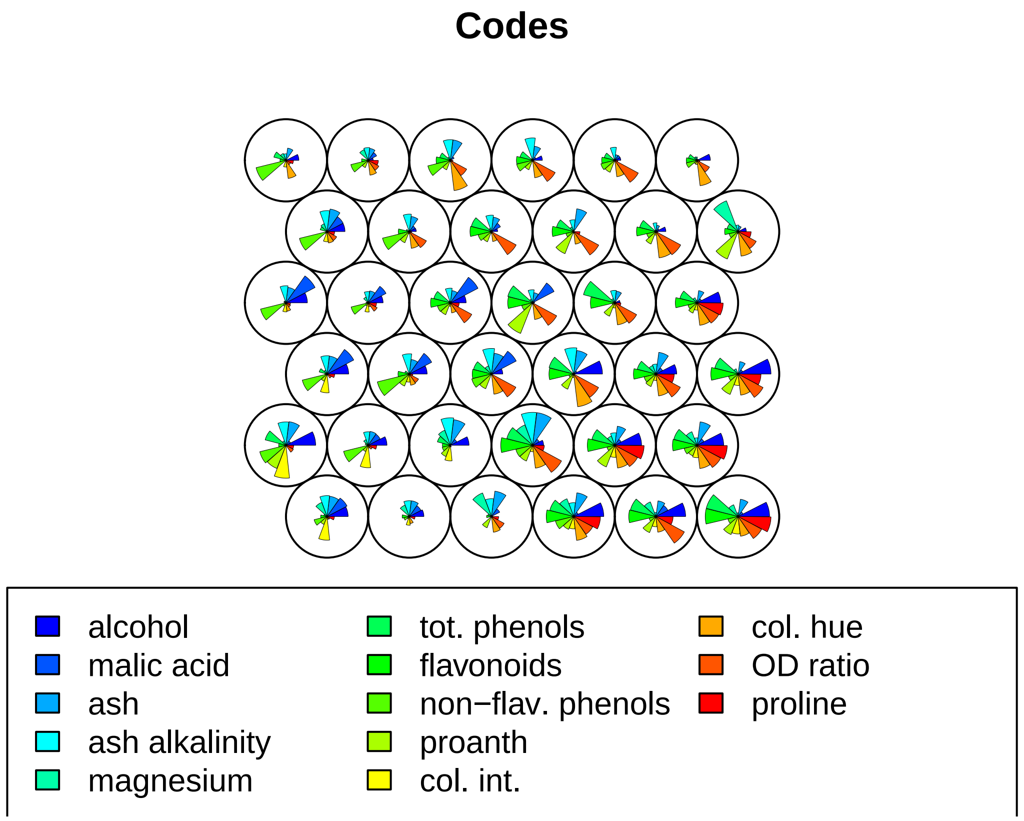
4.5.3.4 Weight vectors / Codes

These codes are made up of normalised values of the original variables from the dataset used to train the SOM. Each node weight vector is similar (representative) of the samples mapped to that node. By doing this visualization it is possible to see patterns in the distribution of the values of variables.

Key points:

- The visualization of >7 variables with this approach is usually of not much use.

```r
plot(som_model,
     type = "codes",
     main = "Codes",
     palette.name = coolBlueHotRed)
```

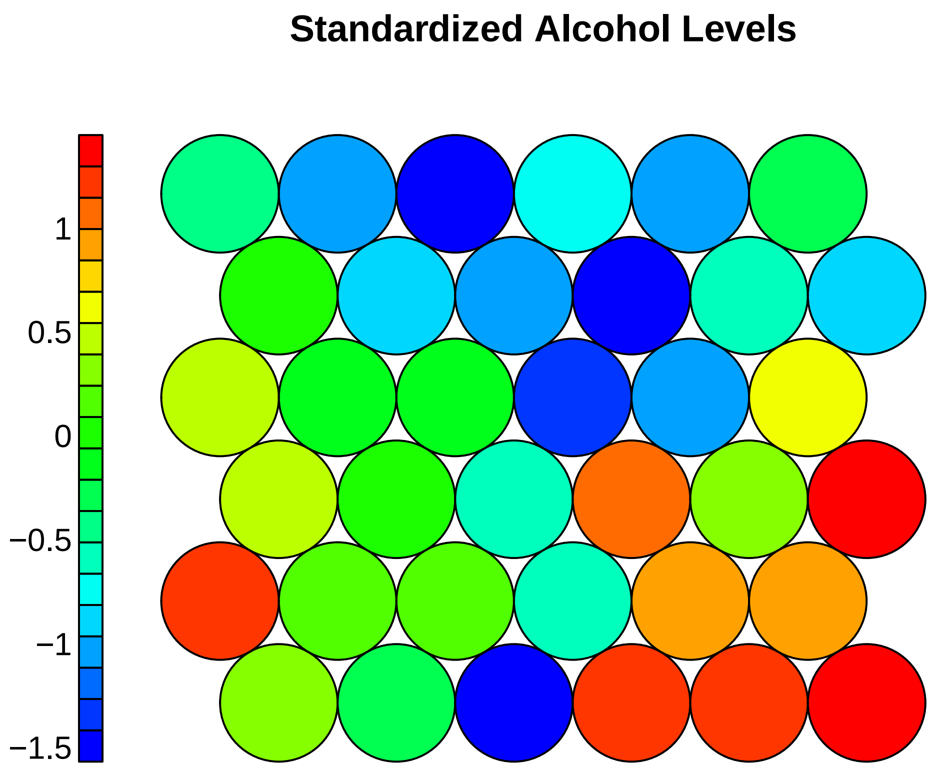


4.5.3.5 Heatmaps

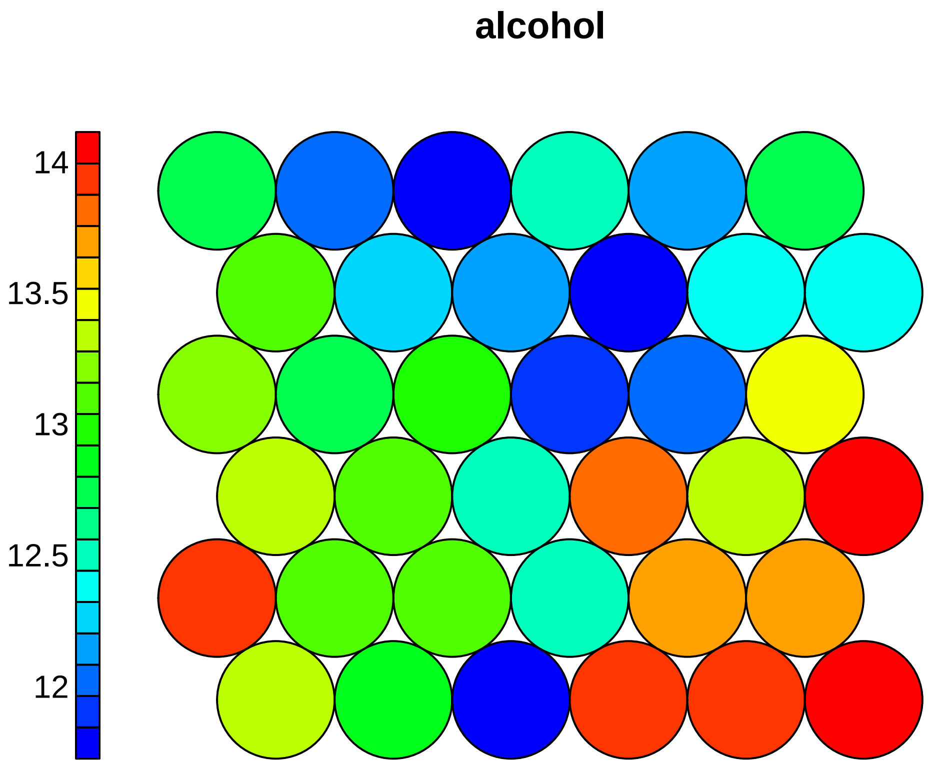
One of the most important visualization of a SOM. It allows for a weight space view of a single variable (distribution). Usually, multiple heatmaps are produced based on different variables and then compared to see if areas and/or patterns of interest are discovered. Key points:

- When generating multiple visualizations of the same SOM, individual sample positions do not move form one visualization to another, the map is simpleyu colored by different variables.

```r
codes <- as.data.frame(as.list(som_model$codes))

#Standardized values
plot(som_model,
     type = "property",
     property = codes$alcohol,
     palette.name = coolBlueHotRed,
     main = "Standardized Alcohol Levels")
```



Heat maps are usually presented to non-technicall people, therefore it is prefered to use the normal values as they appeared on the original dataset. The following heatmaps are presented with values previous to standardization.



4.5.3.6 Drawing Conclusions

Understanding how a Self Organizing Map is generated is of key value to run these analysis. Once a clear understanding of the mathematics and algorithmia is achieved then the process of training the SOM can start. Special attention must be put in the quality of the SOM, the right amount of ephocs and the size of the grid as well.

In the previous example one can see that wines with high levels of alcohol tend to have low OD ratios and high levels of total phenols. Combined with domain knowledge these assosiations can be put to good use in the field.

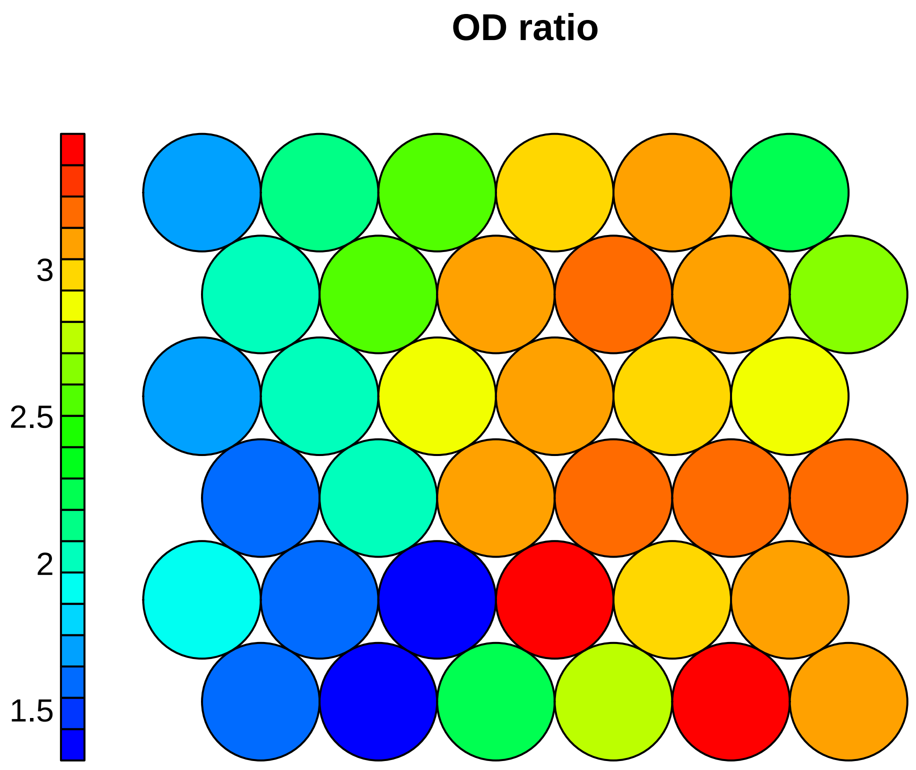


Figure 4.1: OD Ratio in Wines

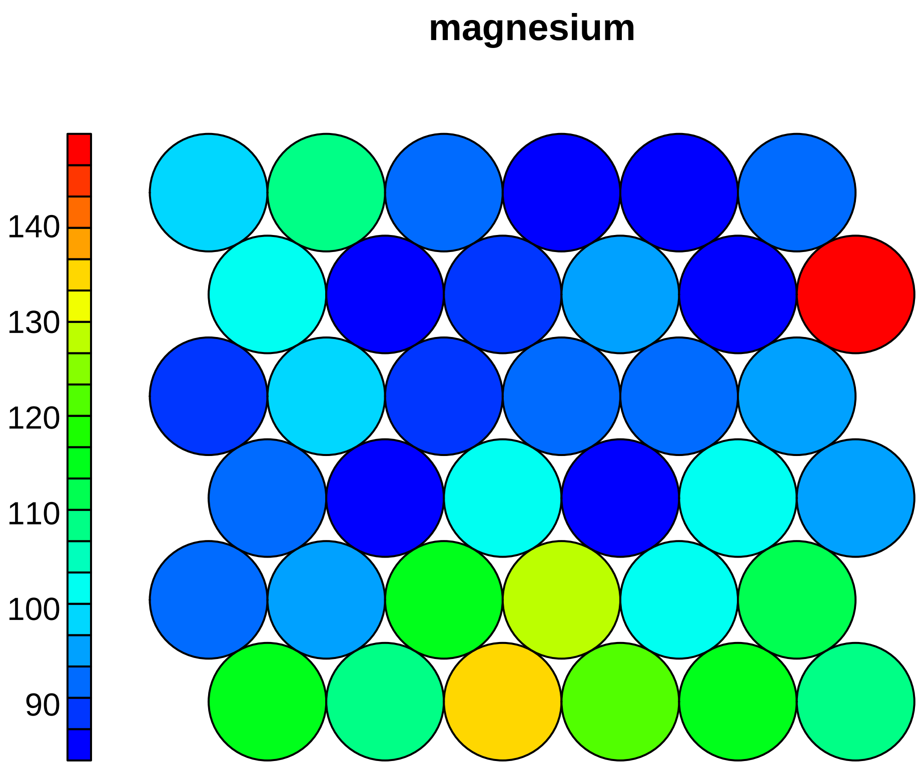


Figure 4.2: Magnesium levels in Wines

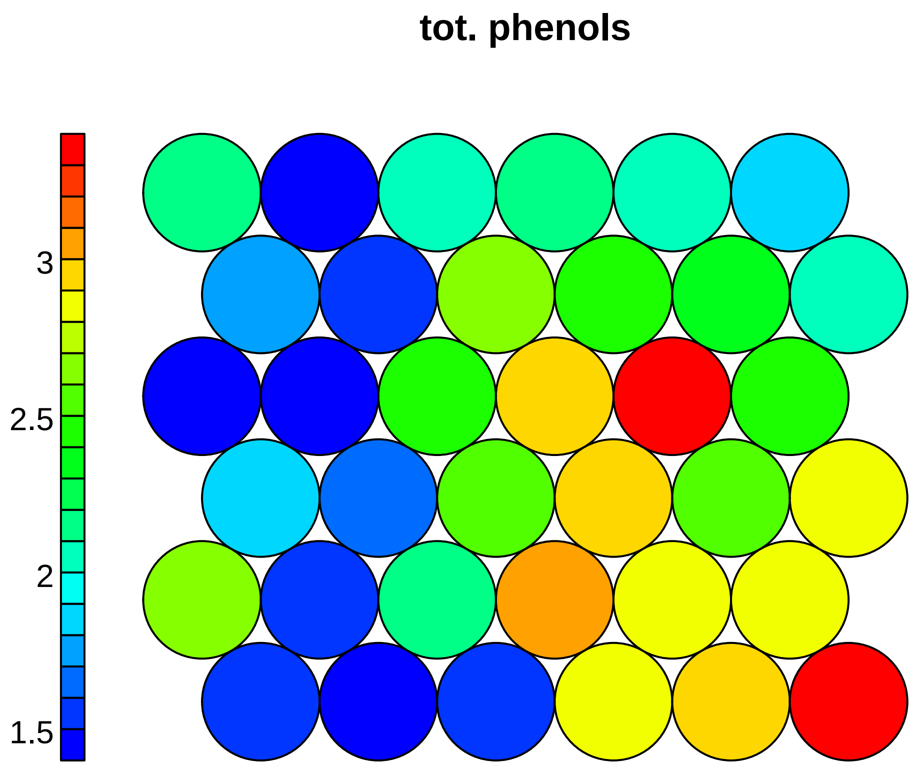


Figure 4.3: Total Phenols levels in Wines

Bibliography

Brownlee, J. (2011). Clever algorithms: nature-inspired programming recipes. Jason Brownlee.

Bullinaria, J. A. (2015). Self organizing maps: Fundamentals.

Kohonen, T. (1995). Self-organizing maps, volume 30 of springer series in information sciences.