**MACHINE LEARNING**

**ASSIGNMENT II**

On:

Active learning, Cluster analysis and SOM

Prepared by:

Patel Darsh Rajesh 2018B4A70532P

Manan Malani 2018B4A70709P

Anuraj Som 2018A1PS0037P

**Active Learning**

Active learning is a special case of [machine learning](https://en.wikipedia.org/wiki/Machine_learning) in which a learning algorithm can interactively query a user (or some other information source) to label new data points with the desired outputs. In statistics literature, it is sometimes also called [optimal experimental design](https://en.wikipedia.org/wiki/Optimal_experimental_design). The information source is also called *teacher* or *oracle*.

There are situations in which unlabeled data is abundant but manual labeling is expensive. In such a scenario, learning algorithms can actively query the user/teacher for labels. This type of iterative supervised learning is called active learning. Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. With this approach, there is a risk that the algorithm is overwhelmed by uninformative examples. Recent developments are dedicated to multi-label active learning, hybrid active learning and active learning in a single-pass (on-line) context combining concepts from the field of machine learning (e.g. conflict and ignorance) with adaptive, [incremental learning](https://en.wikipedia.org/wiki/Incremental_learning) policies in the field of [online machine learning](https://en.wikipedia.org/wiki/Online_machine_learning). Whole model is being from scratch without any help from active learning libraries.

*(In our Model various instances of one saved model have been created for the reason to maintain uniformity among the prediction states)*

**Dataset**

Dataset used for performing active learning is the famous **mnist** dataset , reason for choosing this dataset was it was a multiclass dataset and it we can easily use plot images in the dataset to check the problem in our model.The dataset contains 60,000 images for training

So according to problem question 1(A) in our dataset.

:- We have taken about 10% (6000 images) for initial labeling of the model so there are about (54,000 images) to be removed which will eventually work as human annotators or oracles.

**Pool based Active learning**

In this scenario, instances are drawn from the entire data pool and assigned an informative score, a measurement of how well the learner “understands” the data. The system then selects the most informative instances and queries the teacher for the labels.

In this what we have done is we have considered remaining dataset as a pool and query the dataset on the basis of different methods

* Least Confidence (Uncertainty Sampling)
* Margin Sampling (Uncertainty Sampling)
* Entropy (Uncertainty Sampling)
* Vote Entropy
* KL divergence

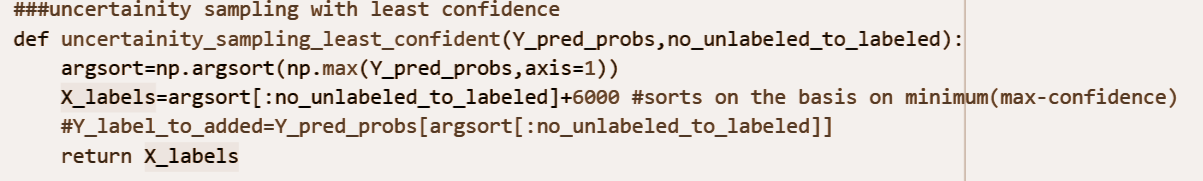
**Stream based Active Learning**

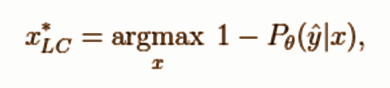
Here, each unlabeled data point is examined one at a time with the machine evaluating the informativeness of each item against its query parameters. The learner decides for itself whether to assign a label or query the teacher for each datapoint.

**Uncertainty Sampling**

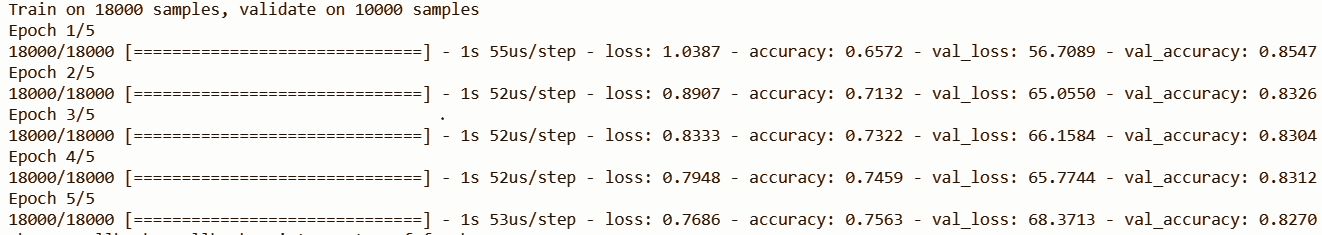
*(please note your output will vary a little with ours)*

1. **Least Confidence**: Probability of point existing has been calculated for different classes of data and the example of the data are sorted in a way which has minimum probability are taken above and choosed to query.



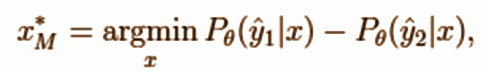


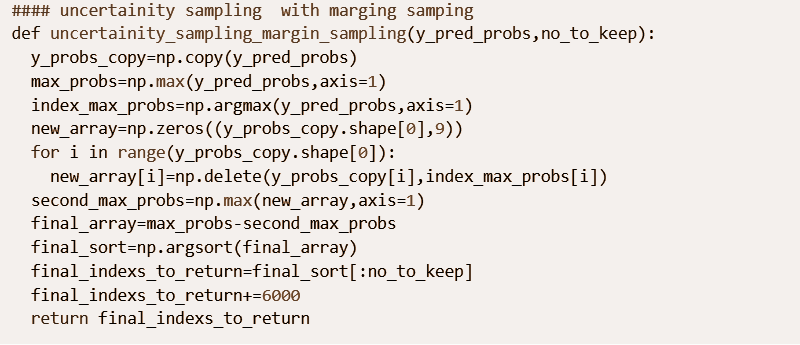
Formula Used for this is :-



As we can see results are not so bad as training accuracy low in the initial epoch can be justified as model is not able to do well as it will take some more epochs for model to getted fitted to such data (which have low confidence predictions value for even the maximum class) but main point to see is that val accuracy of the model is still 0.827 after the final epoch which is quite decent for just training over 24,000 (18000+6000)examples in comparison to training on 60,000 examples and giving val\_accuracy of 0.86 But this just show a better model for query strategy can be used.

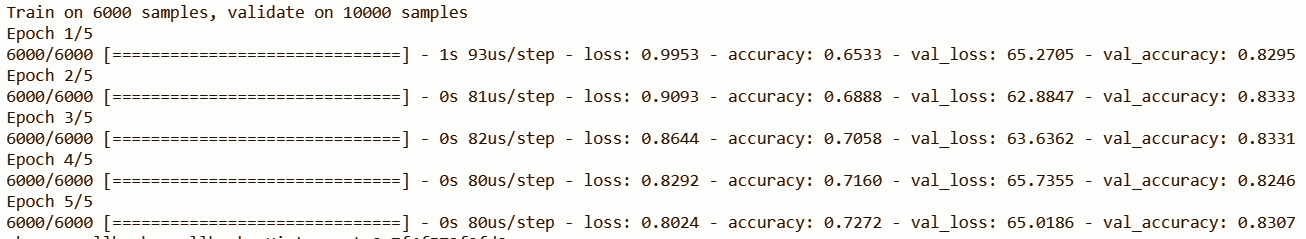
1. **Margin Sampling**: In Margin Sampling Rather than Taking minimum probability among the (max eg prob dist) like we did in least confidence we take difference max and second max probability classes and take difference between i.e out of all examples the one with minimum difference is chosen to query.



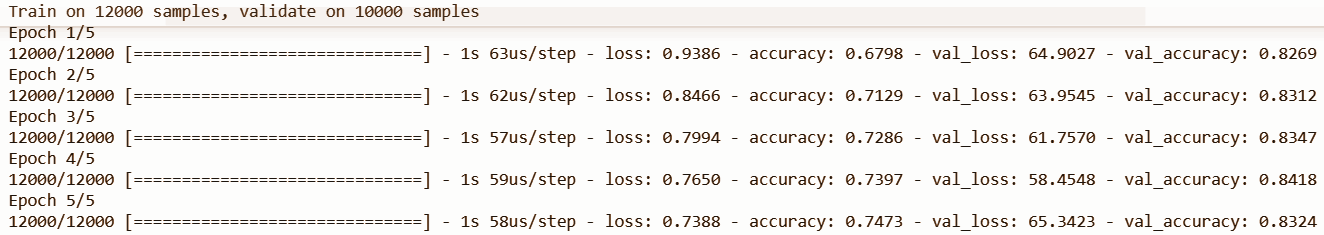


Results of this model are:-

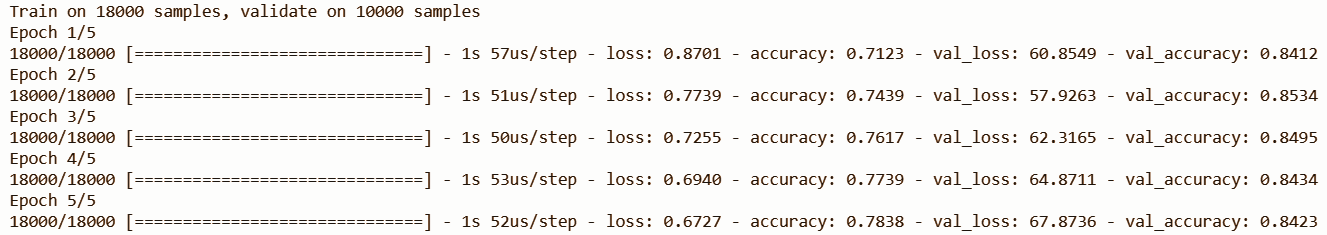
10% additional labels trained



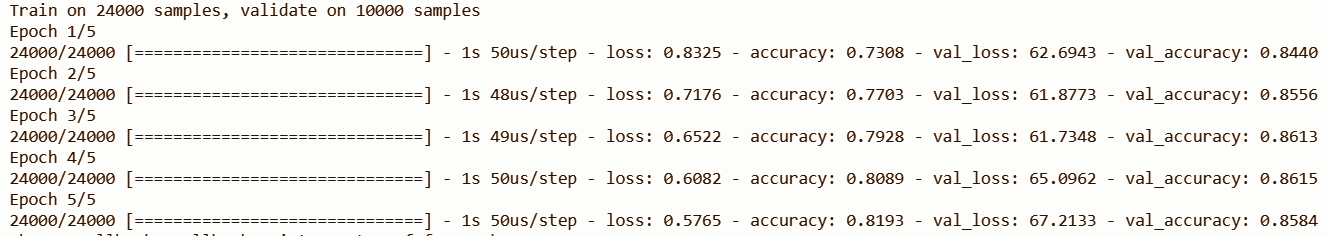
20% additional labels trained



30% additional labels trained



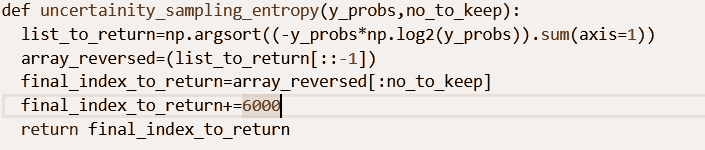
40% additional labels trained



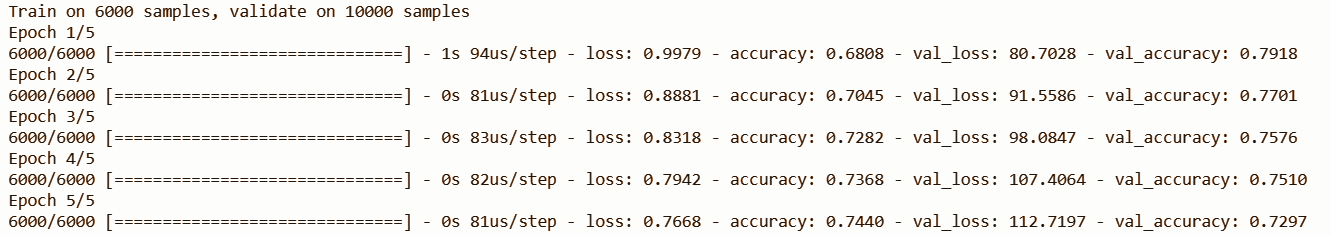
As we can see final val accuracy is quite well 0.85 which in comparison to a model trained on a complete dataset(0.88) is not quite bad and quite impressive.

1. **Entropy Sampling**: This query strategy method samples and sort the data on the basis of entropy of the samples and the sample which has the largest entropy is being queried first.

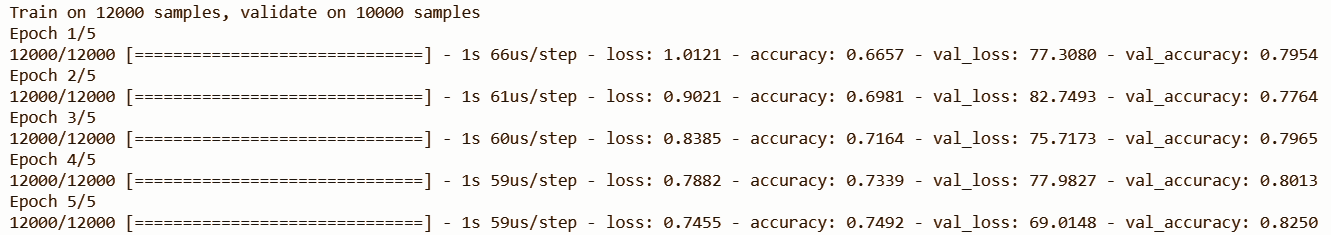




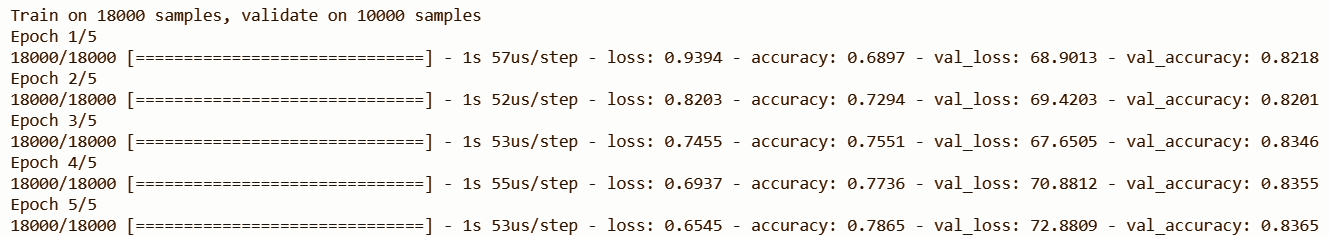
10%



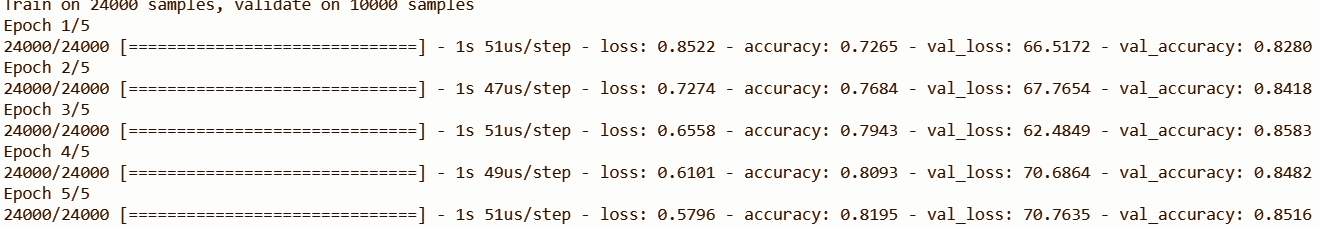
20%



30%



40%



So in this we can see that final validation accuracy is 0.8516 which is pretty decent and the model performed fairly showing how the use of Active learning can reduce the amount of points to labeled thus improving cost and efficiency.

**Query by committee**:

*Query by committee* is another popular active learning strategy, which alleviates many disadvantages of uncertainty sampling. For instance, uncertainty sampling tends to be biased towards the actual learner and it may miss important examples which are not in the sight of the estimator. This is fixed by keeping several hypotheses at the same time, selecting queries where disagreement occurs between them.

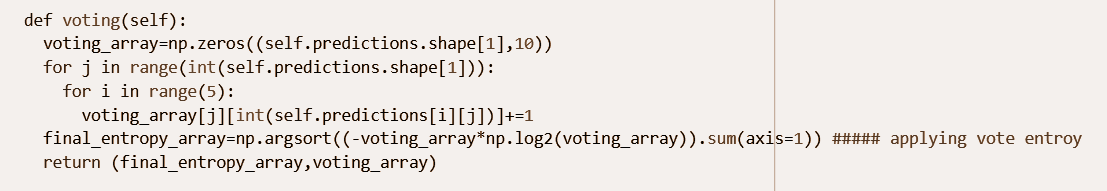
A five membered committee has been created with following classifier models:-

* SVM
* Random Forest
* Logistic Regression
* Decision tree
* Neural network

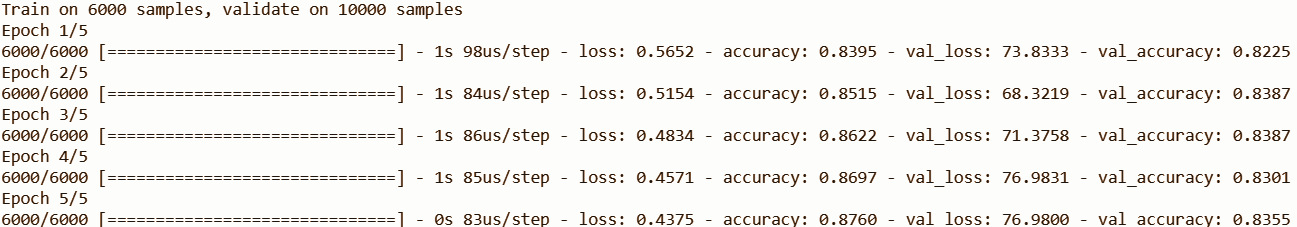
**Voting Entropy**

So with the help of this 5 models voting array was made which shows the votes provided to a given example by these five different models

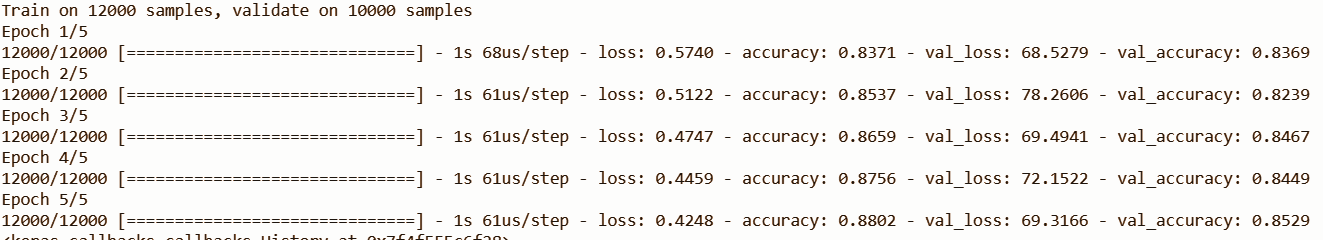
Using this voting array for all the unlabeled points we created voting entropy to query the points as the one example with largest entropy is to be queried first.



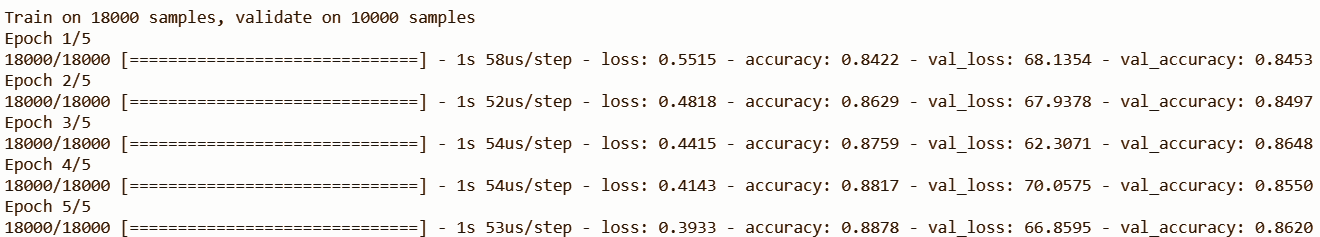
10%



20%



30%



Model is giving pretty decent validation accuracy of 0.8566 in comparison

And quite better in comparison to least confidence

**Kl divergence**

The [Kullback-Leibler Divergence](https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence) score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution.

The KL divergence between two distributions Q and P is often stated using the following notation:

* KL(P || Q)

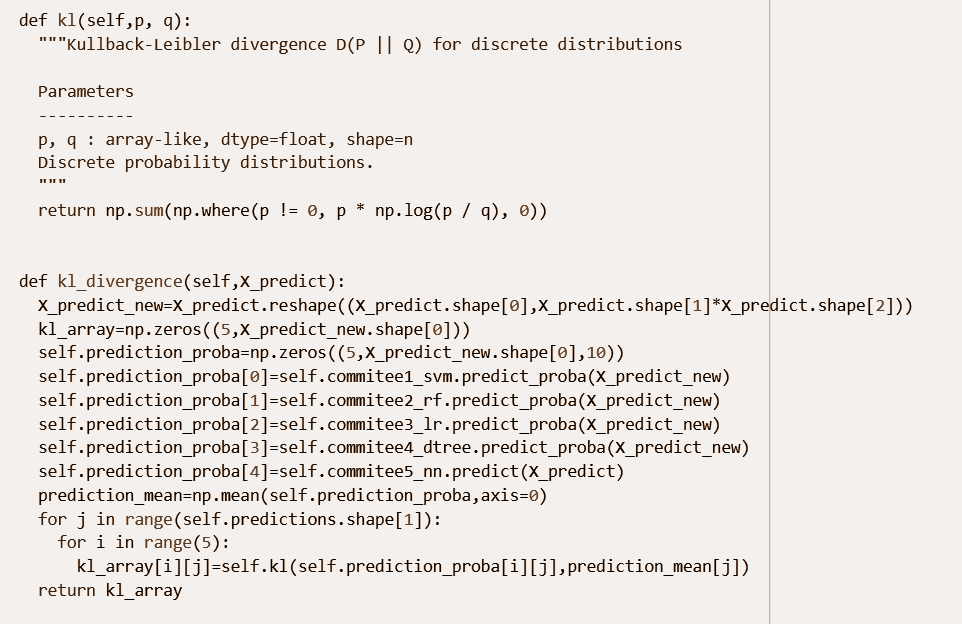
Where the “||” operator indicates “*divergence*” or Ps divergence from Q.

KL divergence can be calculated as the negative sum of probability of each event in P multiplied by the log of the probability of the event in Q over the probability of the event in P.

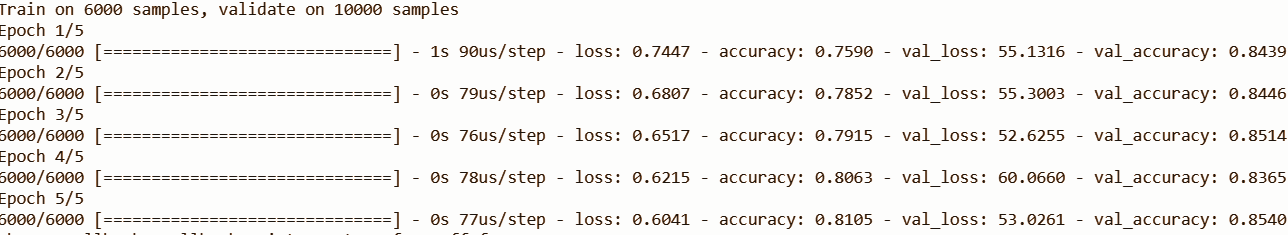
* KL(P || Q) = – sum x in X P(x) \* log(Q(x) / P(x))

The value within the sum is the divergence for a given event.

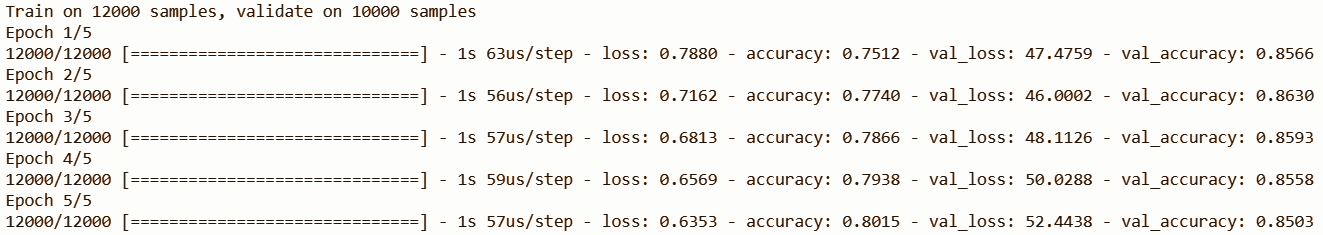
This is the same as the positive sum of probability of each event in P multiplied by the log of the probability of the event in P over the probability of the event in Q (e.g. the terms in the fraction are flipped). This is the more common implementation used in practice.



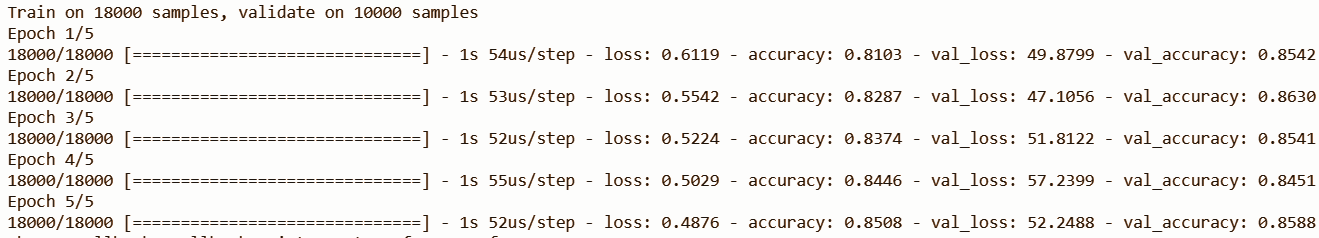
10%



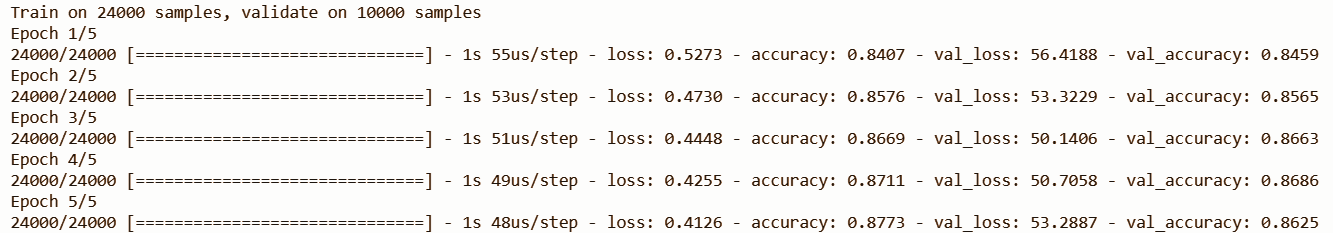
20%



30%



40%



As we can see we get a final validation accuracy of 0.8625 which is much higher than others as kl divergence has performed really well compared to other examples

**Version Space**

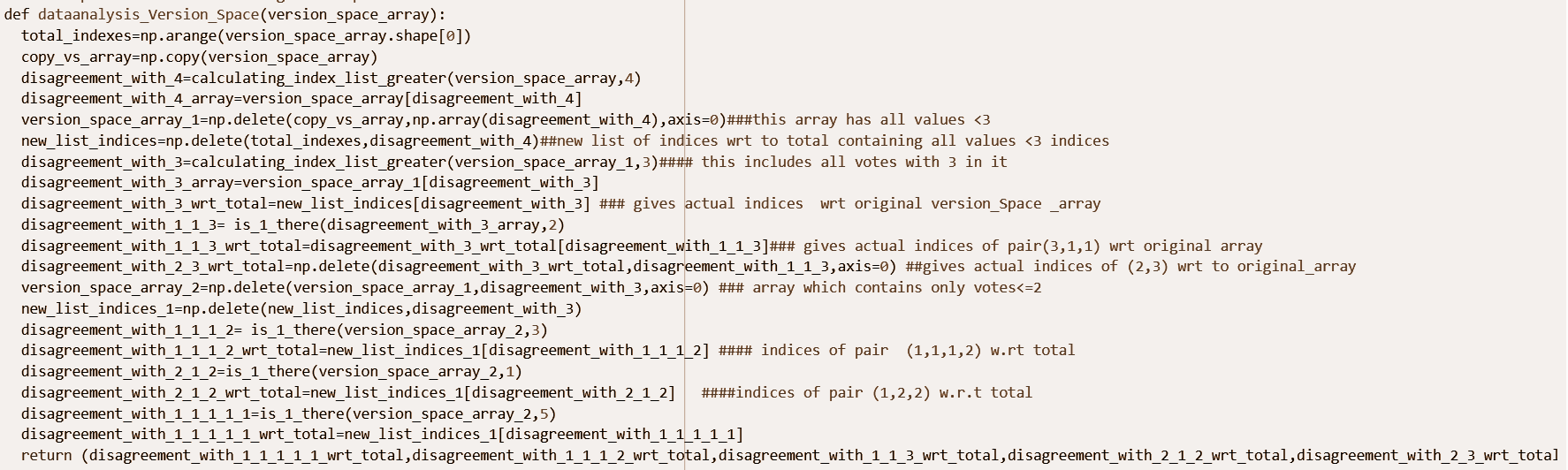
That portion of the predicted space where at least some disagreement takes place . So for the committee of five members version space was calculated which turns out to be

26534 images(please remember that prediction and fitting of the model are random every time we run so the result may vary a bit when you will run this) out of 54000 images have at least one disagreement .

So for the committee of five members total pairs of disagreements possible are (1,4) ,(2,3) ,(2,1,1,1) ,(3,1,1),(1,1,1,1,1)and ,(2,1,2).

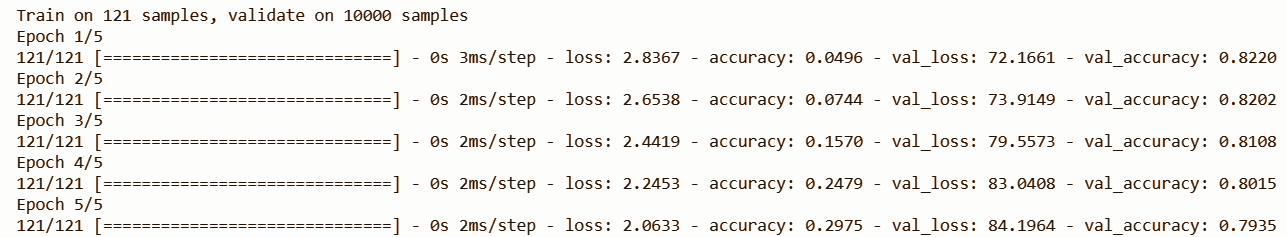
So to decrease the version space the most we will try to order the labels in following order:

* (1,1,1,1,1)
* (2,1,1,1)
* (2,1,2)
* (3,1,1)
* (2,3)
* (1,4)

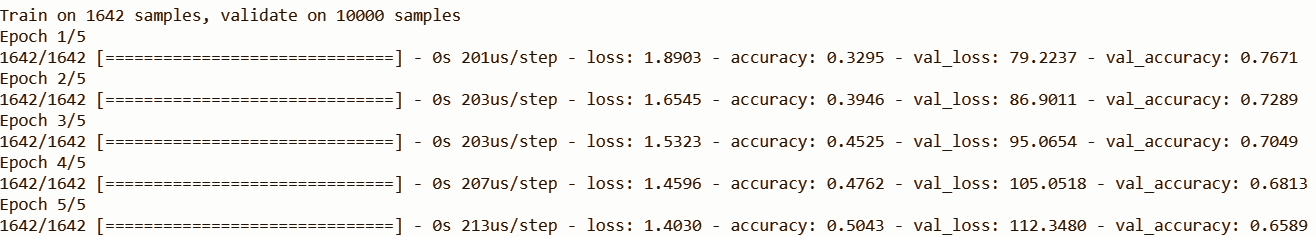


So after fitting the model through different disagreements results we get are (Note same model is being fitted and refitted again)

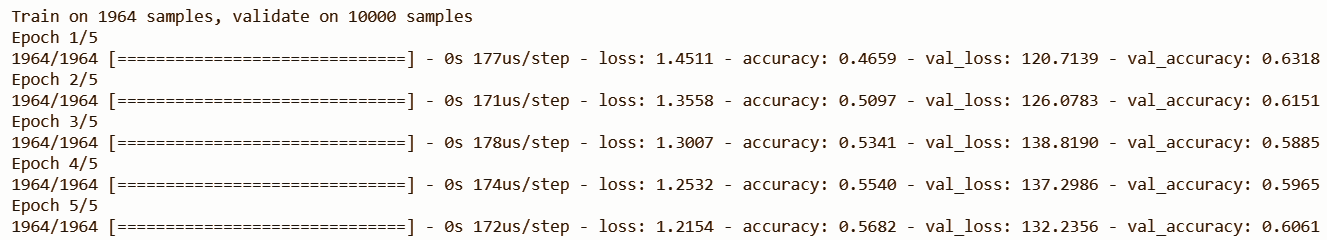
Pair(1,1,1,1)



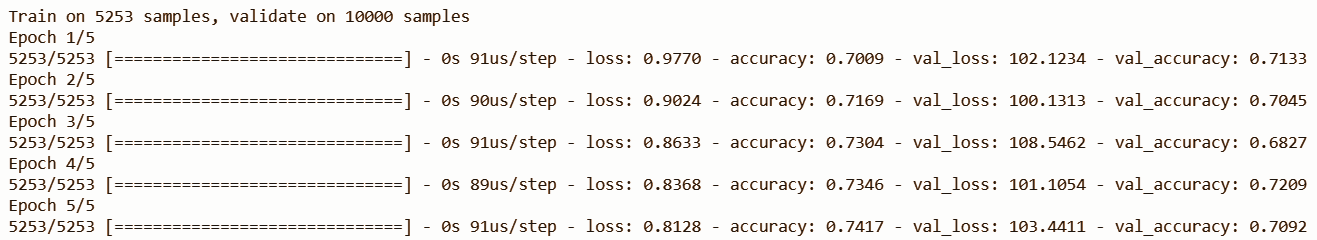
Pair(1,1,1,2)



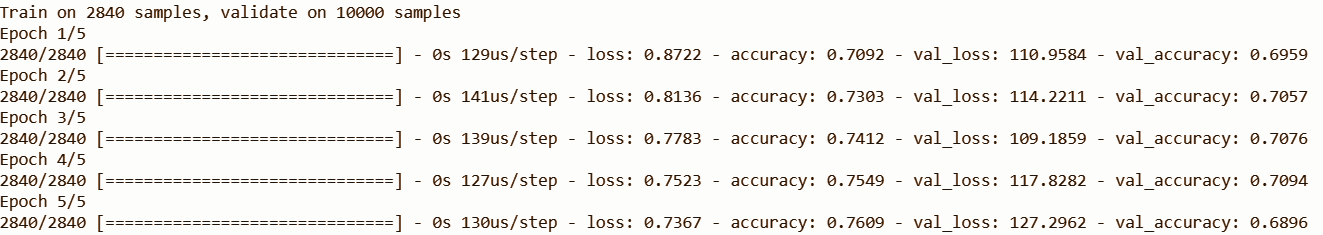
Pair (2,1,2)



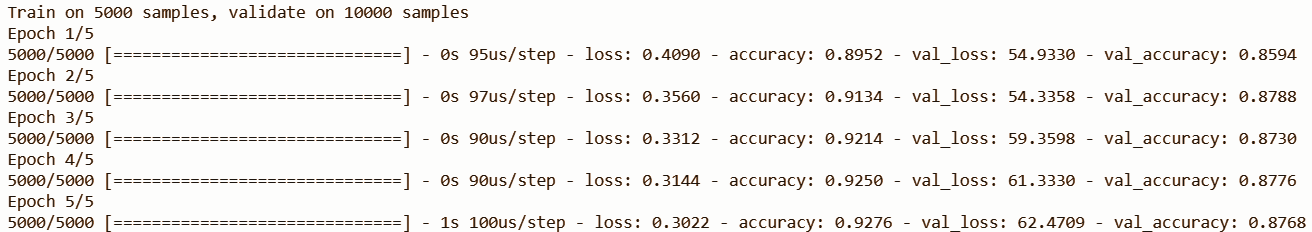
Pair (1,1,3)



Pair(2,3)



Pair(1,4)



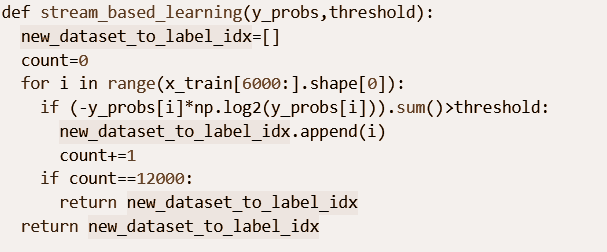
So validation accuracy of 0.87 is been performed by model through the order with just about 25%(15,000) of additional labels which is really good

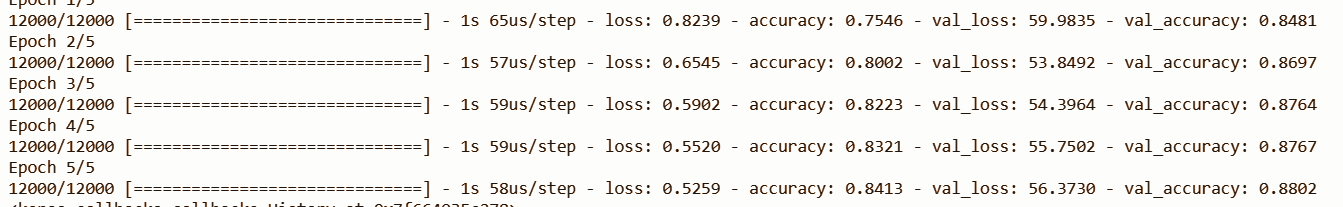
But there has been a decrease in model’s performance in initial stages. The reason is maybe due to presence of outliers as we all know active learning does not perform well in case of outliers so maybe the outliers like image of label 1 looking like two and similar stuff can decrease the performance of the model.

**Stream Based Learning**

Here, each unlabeled data point is examined one at a time with the machine evaluating the informativeness of each item against its query parameters. The learner decides for itself whether to assign a label or query the teacher for each datapoint.

So we have decided to put a query threshold of 1 for data points using entropy as a query strategy and querying a total of 12,000 additional images.



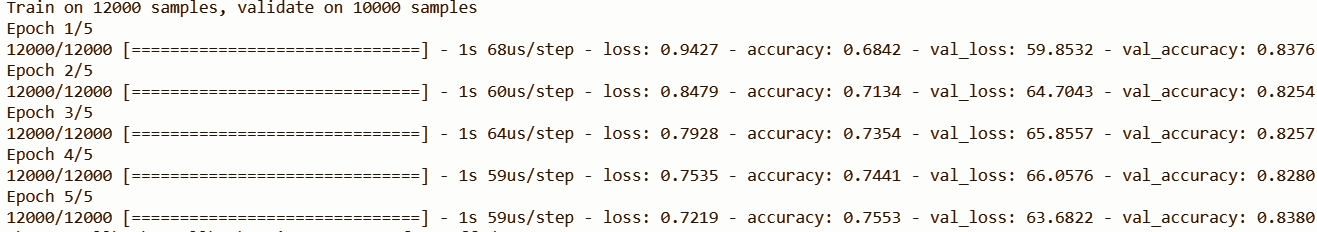


Validation accuracy (0.88) we get is extremely good which shows how impressive our model’s performance is.

**Comparison with random model**

*(Your results may vary with ours)*

Random model, trained on additional 30% data labels



So as you can see our best active learning has performed much better(0.8627) than randomly labelling data which has performed a val\_accuracy of 0.8380.

Cluster Analysis

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group are more similar (in some sense) to each other than to those in other groups. These groups are called clusters. For implementing these similarities to form clusters, certain types of distances between the objects are used. For effective clustering, intra-cluster distances or the distances between the objects in the same cluster are minimized and inter-cluster distances or the distances between several clusters are maximized. It is a common technique used in statistical data analysis, in many fields like pattern recognition and image processing.

There are several types of approaches to clustering dependent on how a cluster is defined. These include:

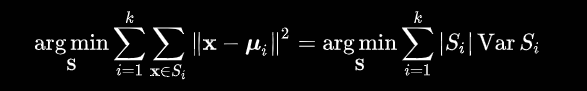
* *Hierarchical clustering*: This is based on the distances between objects which is used to categorise them into clusters. This makes the use of certain distance metrics like Euclidean distance, Squared Euclidean distance and Manhattan distance
* *Partitional clustering*: It involves the construction of a partition of a database of certain objects into a set of *k* clusters that will optimise the chosen partitioning criterion. This could be Global or Heuristic methods. Heuristic methods include k-means and k-medoids. In k-means, each cluster is represented by the center of the cluster. In k-medoids or PAM (Partition around medoids), each cluster is represented by one of the objects in the cluster.
* *Distribution-based clustering*: Clusters are defined as objects belonging most likely to the same distribution.
* *Density-based clustering*: Clusters are defined as areas of higher density than the remainder of the data set. Objects in the sparse areas, required to separate clusters, are usually considered to be noise and border points.
* *Grid-based clustering*: This is used for multi-dimensional datasets. The comparisons for clusters are done using grids or cells in a grid structure.

Here, we are going to be implementing k-means clustering.

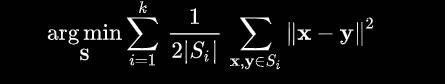
**K-means clustering:**

K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (here means are the cluster centers or cluster centroids) that define the cluster. It can be applied when we know k, the number of clusters we want to find k-means clustering minimizes intra-cluster variances, in terms of squared Euclidean distances.

Given a set of observations (x1, x2, ..., x*n*), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into *k* (≤ *n*) sets S = {*S*1, *S*2, ..., *Sk*} so as to minimize the intra-cluster sum of squares, or variance. This is computed in terms of finding:



where *μi* is the mean of points in *Si*. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:



**Algorithm for clustering:**

* If there are k classes, randomly pick k points as the centroids of the k clusters
* For each point, distances using the defined distance metrics, are calculated from the centroids. Each point is assigned to the cluster whose centroid it is closest to.
* Cluster centroids are recalculated using the cluster data points.
* Again distances are measured from these cluster centroids for each point and the entire procedure continues in a loop.
* The loop continues until there is no change in clusters and cluster centroids between two consecutive iterations.

**Majority-based labelling:**

In order to label the clusters that are created using the clustering algorithm, we can use majority-based labelling. Here the cluster with similar objects are labelled according to the label which has the majority in the cluster for similar objects which helps us in analysing the clustering extent.

**Dataset:**

The dataset being used is the mnist dataset.

**Procedure:**

Here we will be implementing k-means on our dataset. Then we will create the clusters by using 40% of the data. From the clusters that are created, use 20% of the points and their labels in order to label all the points. Then from there, we calculate accuracy. We also calculate the advantages of using Clustering in terms of money and time saved from labelling.

**Implementation in Python:**

* We first import the required files and the dataset
* Then we construct the k-means clustering model as km with a number of clusters as 10 which is equal to the number of classes in the dataset.

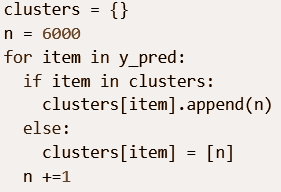
We do our clustering on 40% of the data from x\_train.



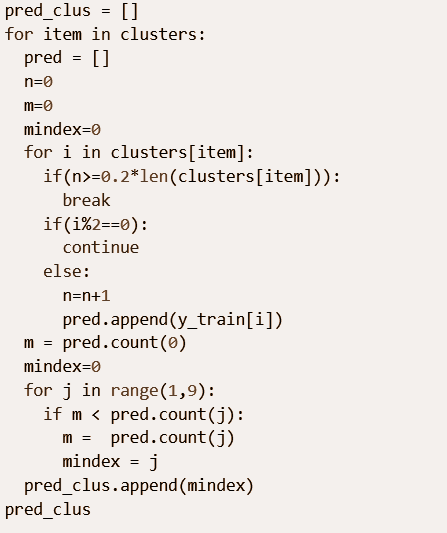
* We fit the model and predict in the data to be clustered and store the values in y\_pred

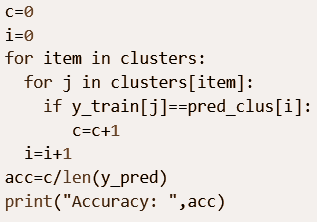


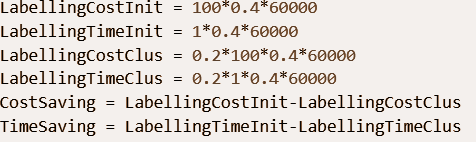
* We then create a visualisation in the form of a table for clusters by storing the cluster number and along with that the subsequent data points in that cluster in the form of their row numbers from the original dataset.



* The code then takes 20% of data points from each cluster and notes their labels. It then uses majority-based learning to then assign the predicted label of each cluster by taking the label of the data point that is the majority for that cluster. This is done for each and every cluster. Then the predicted cluster labels for each cluster is stored in pred\_clus.



* We then calculate the accuracy of the cluster-based labelling.  
    
    
  
* Then the cost in terms of money and time are calculated for the initial case of labelling the points and also for the case of labelling some points and using cluster analysis and majority-based learning. The amount of money and time saved is then calculated.



* The values calculated are as follows:

|  |  |
| --- | --- |
| *Accuracy:* | 0.5886574074074075 |
| *Cost Saved is:* | Rs. 1920000.0 |
| *Time Saved is:* | 19200.0 hours |

Self-Organizing Maps

A self-organizing map or kohonen map is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. Self-organizing maps differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighborhood function to preserve the topological properties of the input space.

**SOM’s Architecture**

Each data point in the data set recognizes themselves by competing for representation. SOM mapping steps starts from initializing the weight vectors. From there a sample vector is selected randomly and the map of weight vectors is searched to find which weight best represents that sample. Each weight vector has neighboring weights that are close to it. The weight that is chosen is rewarded by being able to become more like that randomly selected sample vector. The neighbors of that weight are also rewarded by being able to become more like the chosen sample vector. This allows the map to grow and form different shapes. Most generally, they form square/rectangular/hexagonal/L shapes in 2D feature space.

**The Algorithm**

1. Each node’s weights are initialized.

2. A vector is chosen at random from the set of training data.

3. Every node is examined to calculate which one’s weights are most like the input vector. The winning node is commonly known as the Best Matching Unit (BMU).

4. Then the neighborhood of the BMU is calculated. The amount of neighbors decreases over time.

5. The winning weight is rewarded with becoming more like the sample vector. The neighbors also become more like the sample vector. The closer a node is to the BMU, the more its weights get altered and the farther away the neighbor is from the BMU, the less it learns.

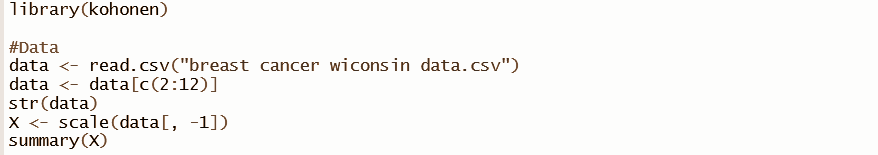
6. Repeat step 2 for N iterations.

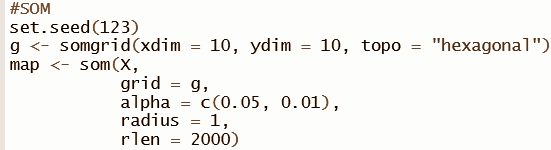
**Best Matching Unit** is a technique which calculates the distance from each weight to the sample vector, by running through all weight vectors. The weight with the shortest distance is the winner. There are numerous ways to determine the distance, however, the most commonly used method is the Euclidean Distance, and that’s what is used in the following implementation.

**Dataset**

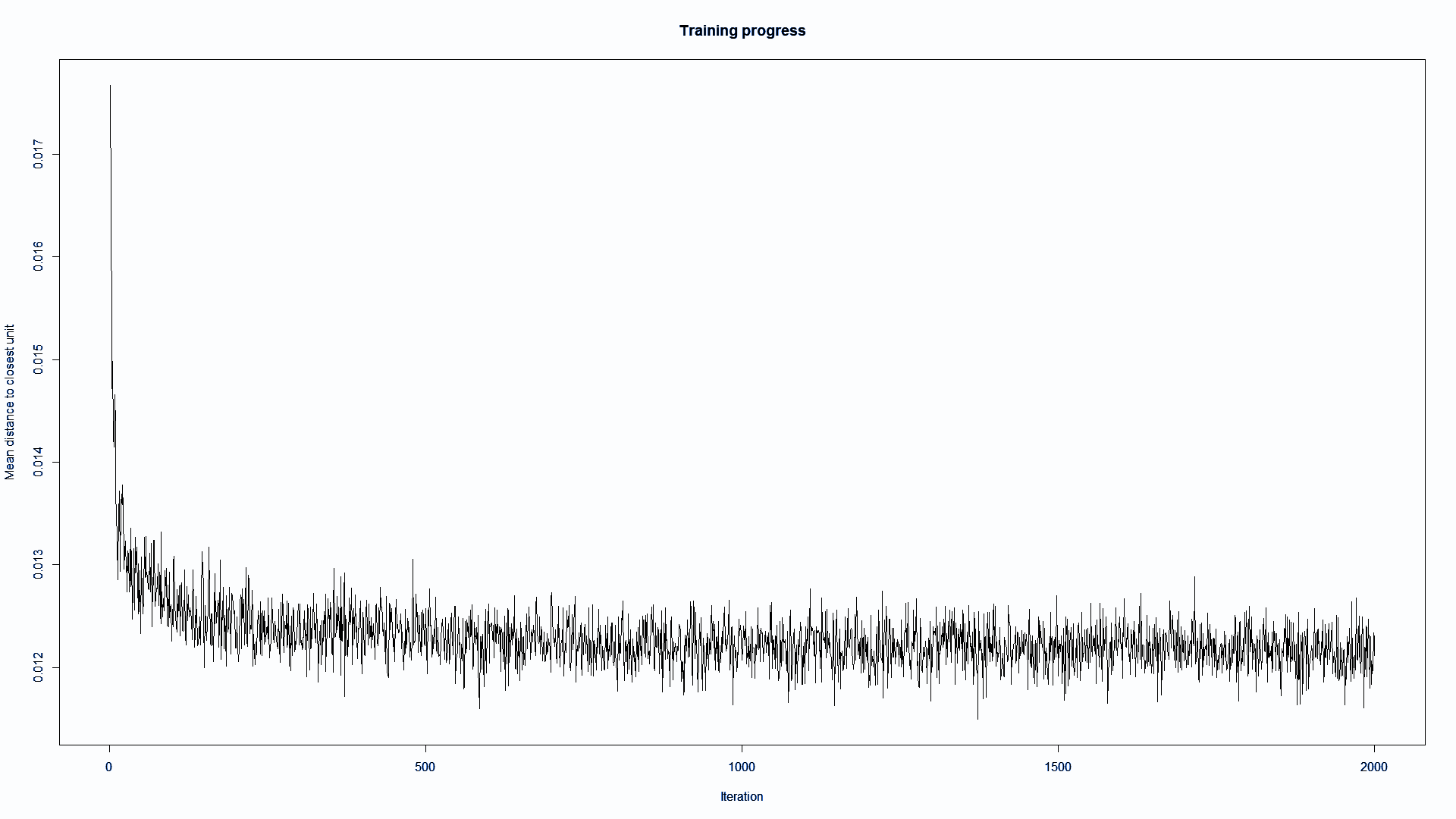
The dataset used for implementing SOM is the Breast Cancer Wisconsin dataset as breast cancer is the most common cancer after skin cancer in women and it affects nearly 2.1Million women each year killing nearly 600K women of them. This dataset contains 569 instances with 10 features which is sufficient to easily visualize kohonen maps.

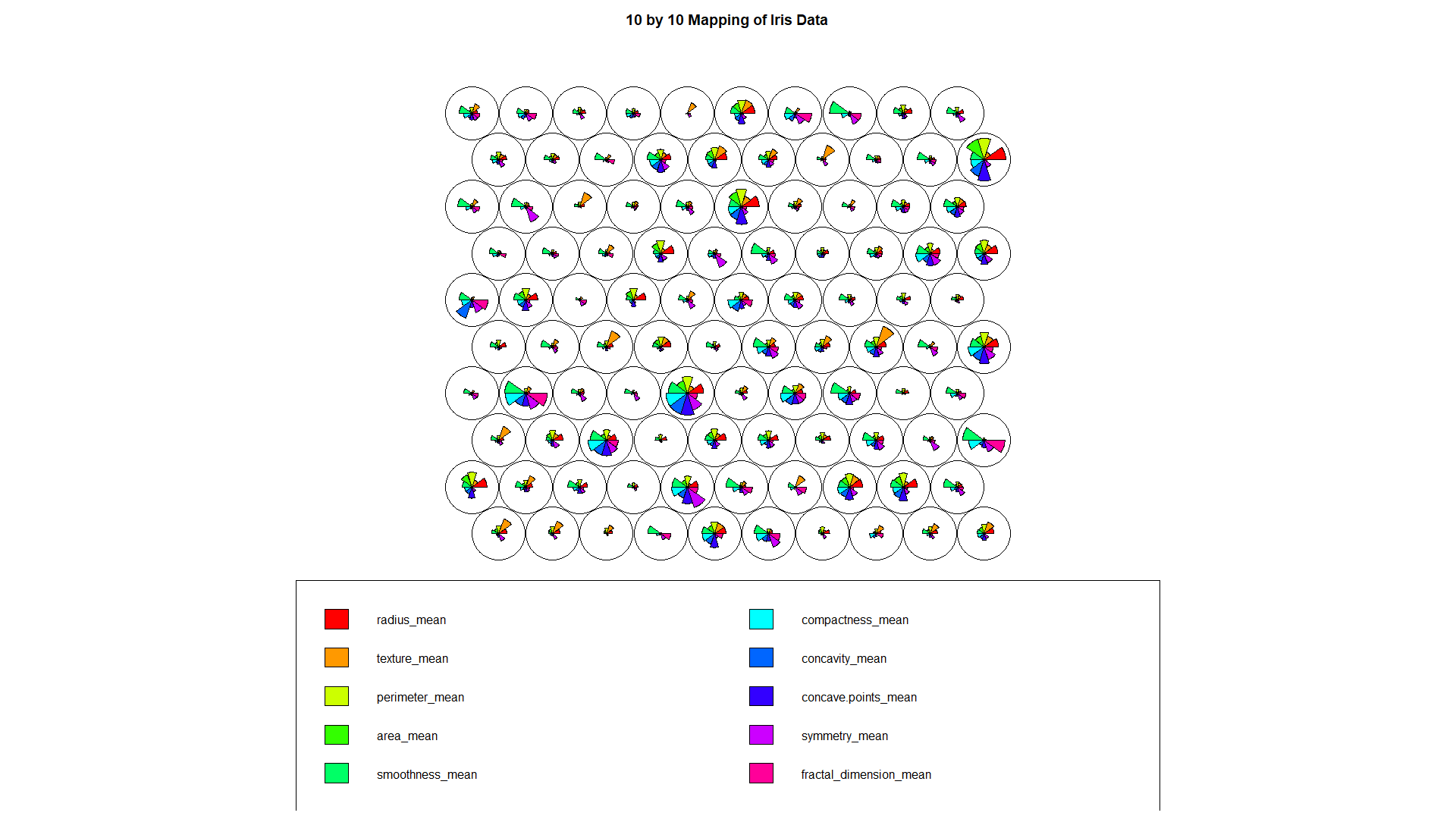
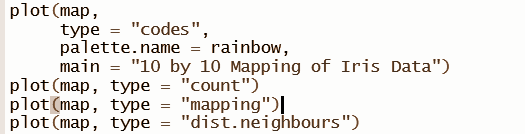
**Implementation (in R)**

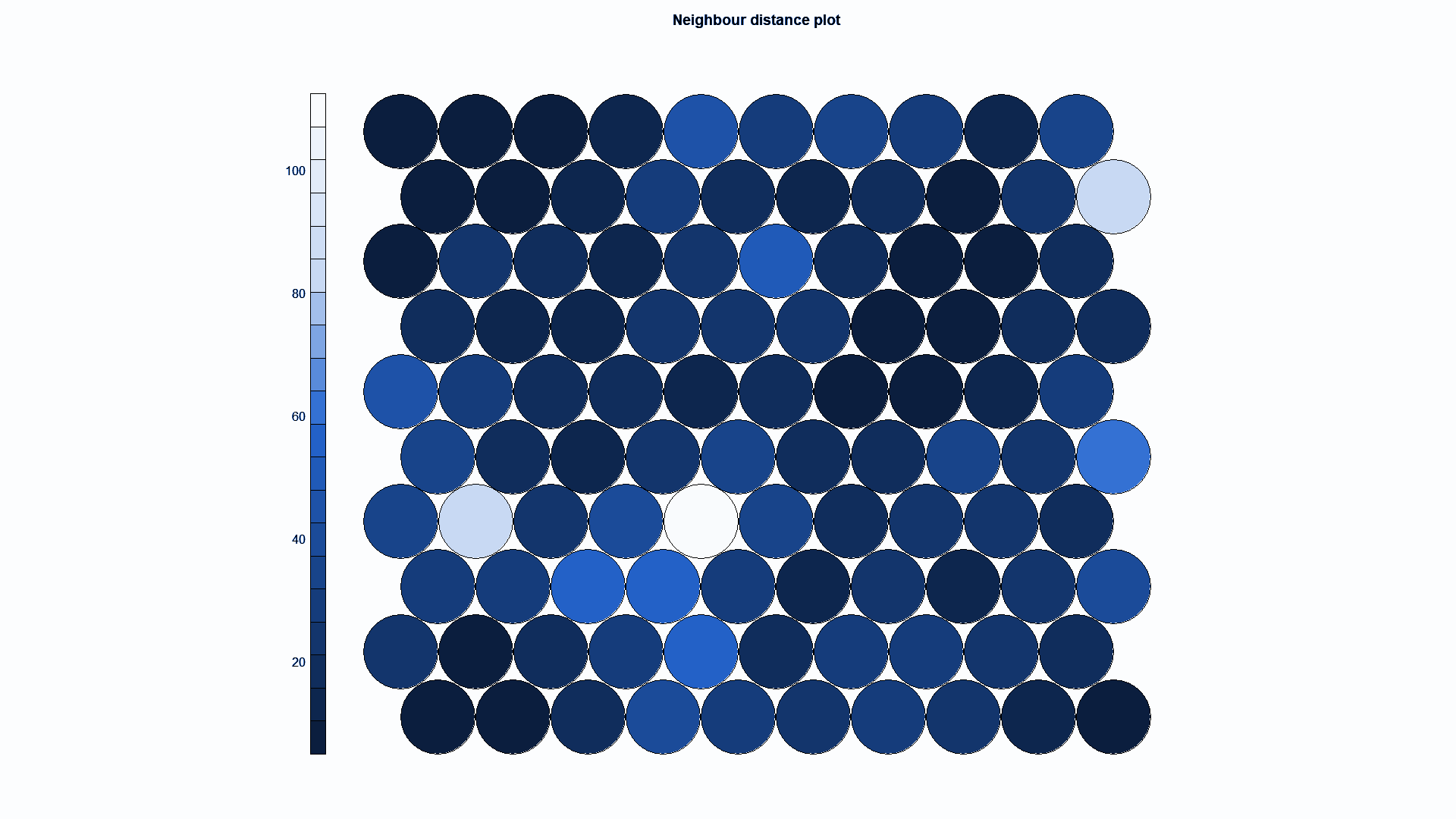
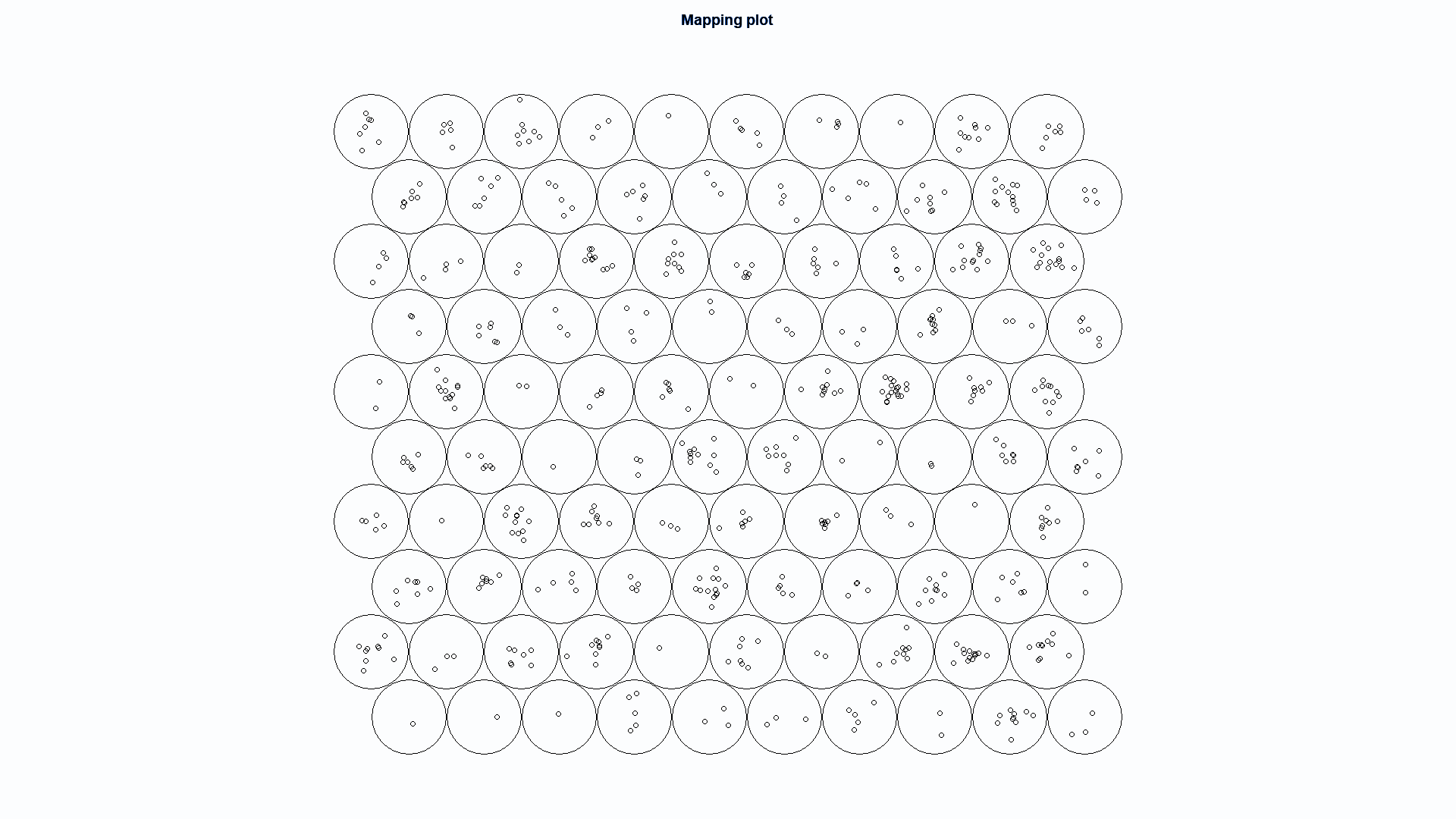
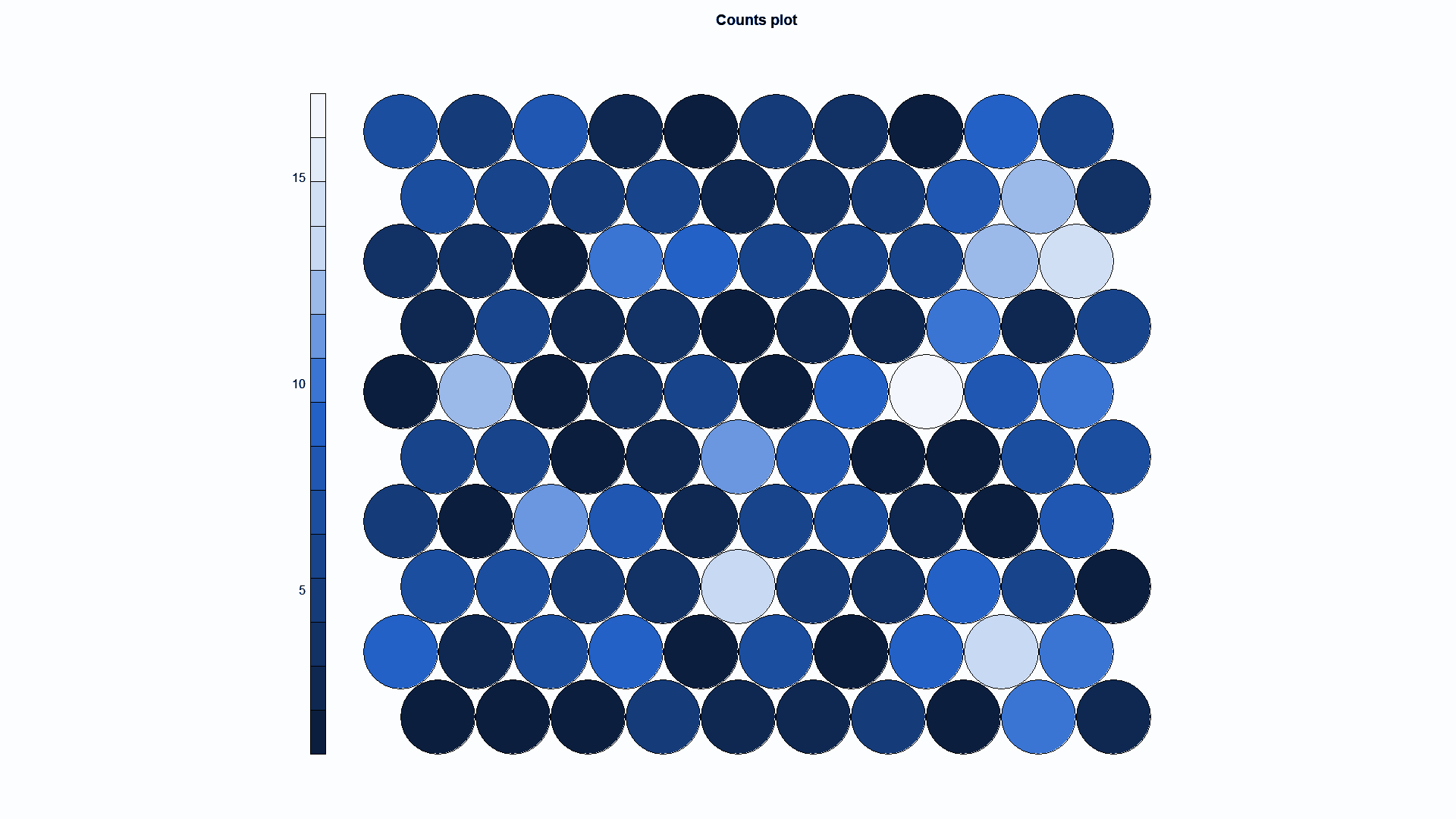
1. We start by importing and normalizing the data 
2. We then create a 10x10 hexagonal som grid. For creating the som map, we use the scaled data stored in ‘X’, use the som grid ‘g’, set the learning rate 0.05 which is linearly declining to 0.01, radius of neighborhood to 1 and the number of steps to 2000



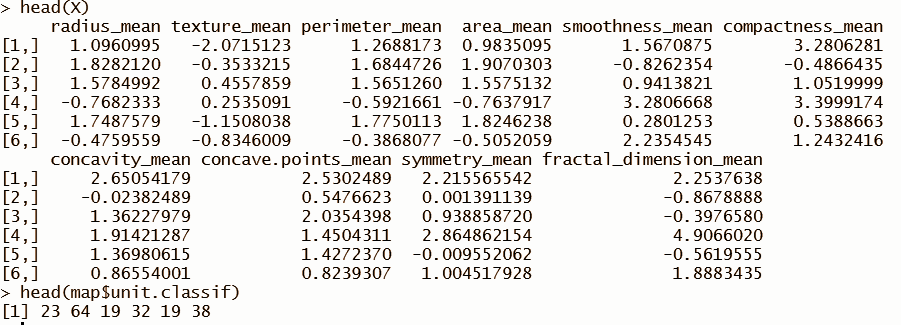
1. Then we plot the graph for changed over number of steps

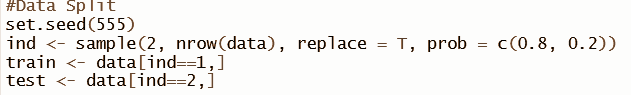
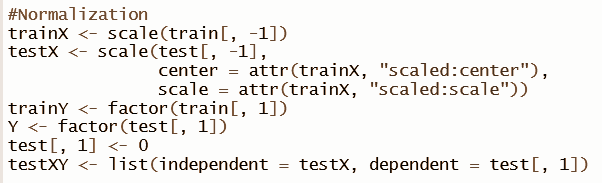
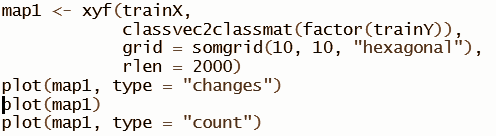


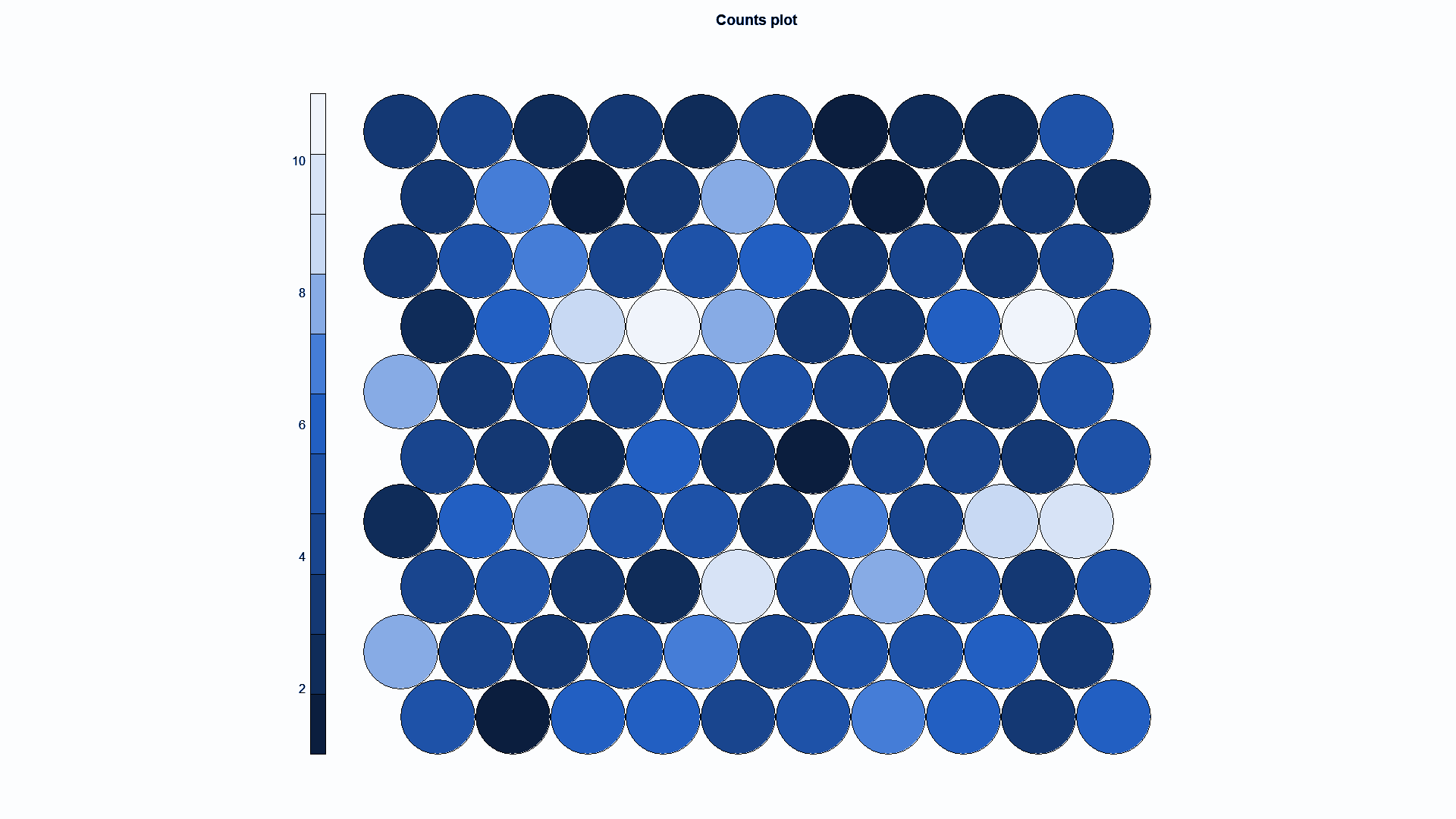
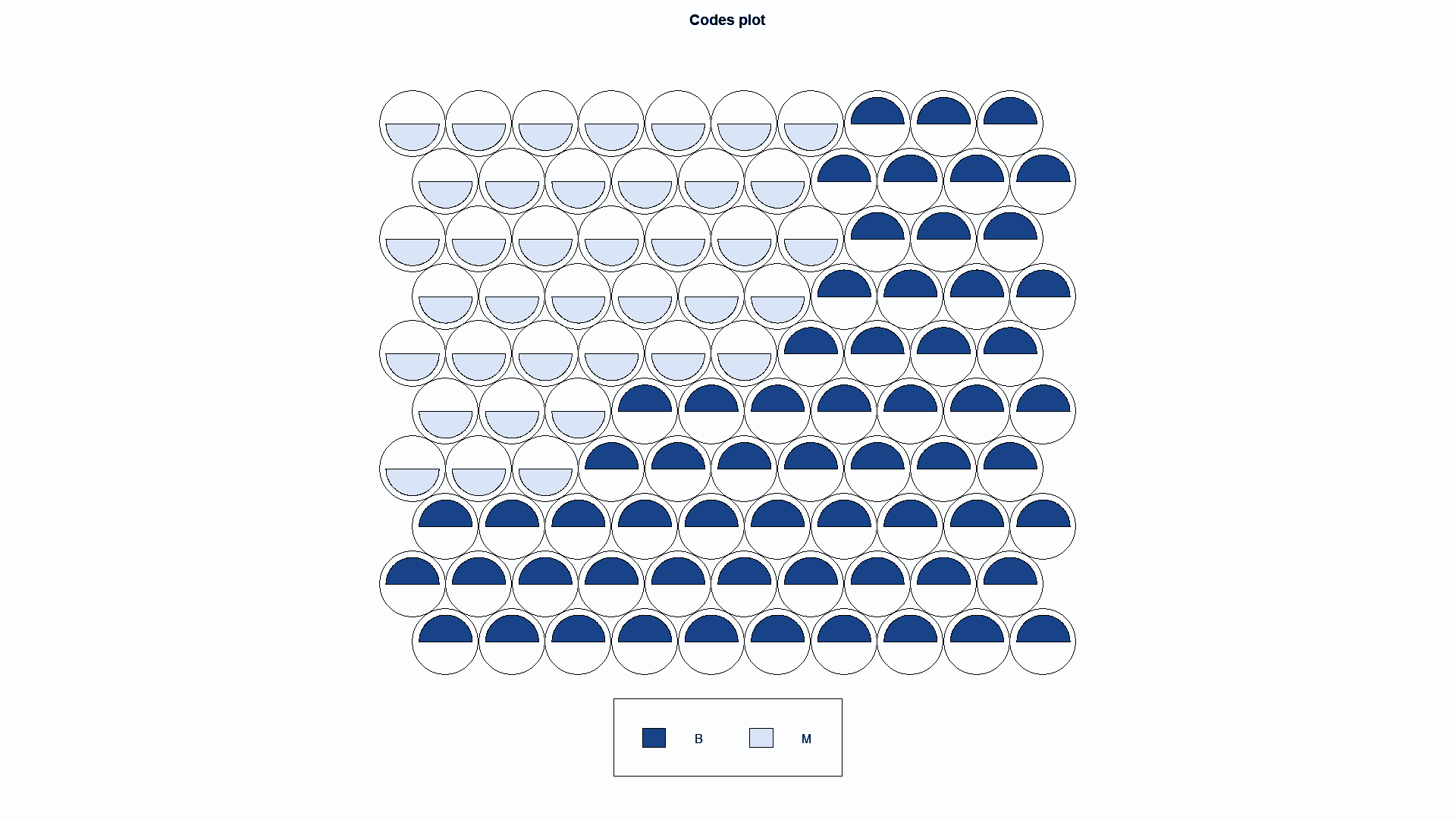
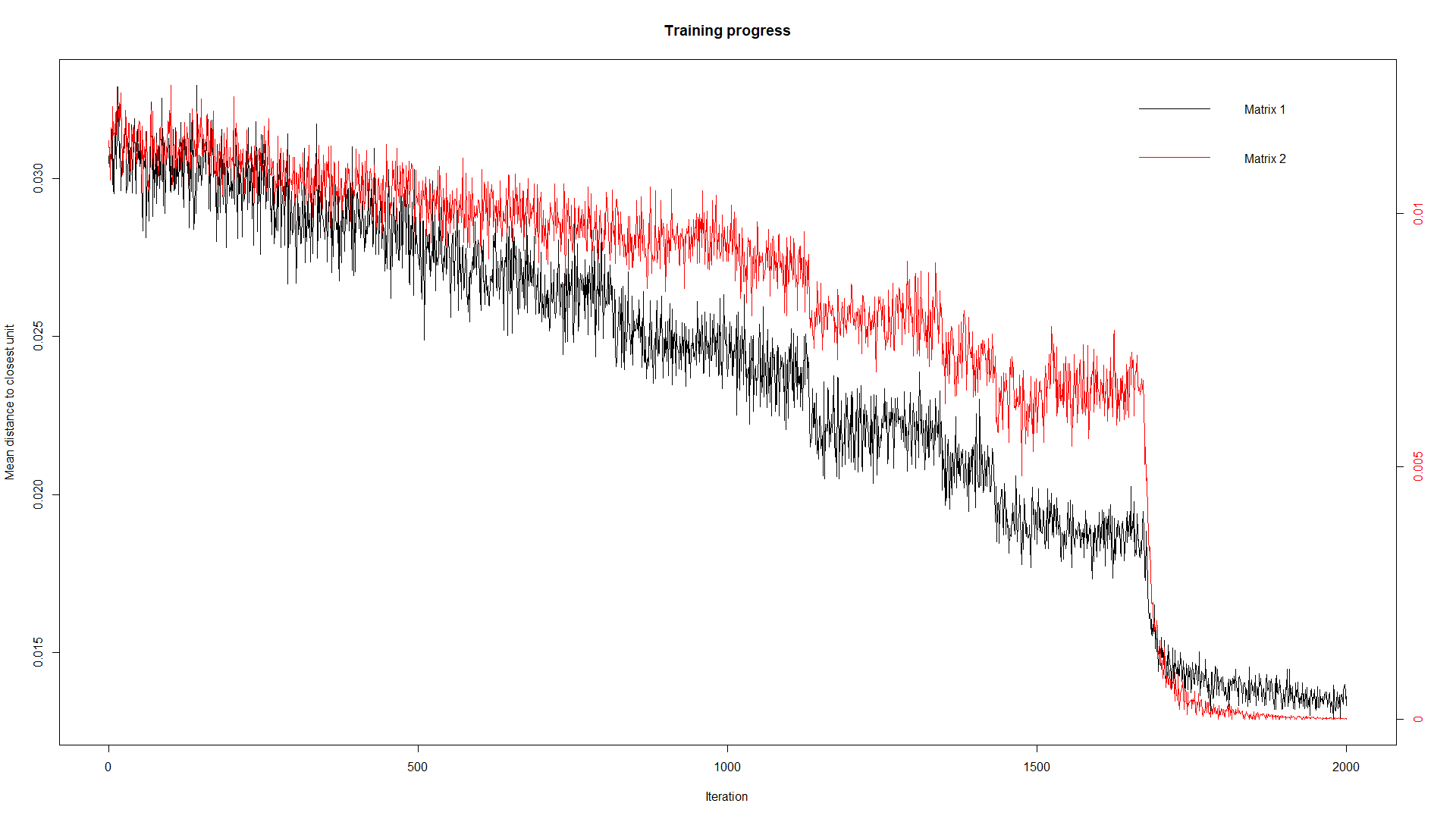
1. Then we plot codes plot, counts plot, mappings plot and distant neighbors plot of map to visualize the data



1. We can see from the image below how the data is plotted using the som, where X represents the data and unit.classif represents in which cell the instance is mapped.



1. For testing the som we split the data into two parts, for training and testing.
2. Then we normalize both training and testing data by removing class labels and make a factor tensor trainY, Y of train and test class labels respectively
3. We train as som named map1 using a 10x10 hexagonal somgrid for 2000 steps and plot the changes, codes and counts plot



1. We plot the codes plot and create a cluster boundary using cutree and hclust