An ANALYSIS on VARIATIONS of INPUT PATTERN DISTRIBUTIONS in SELF-ORGANIZING MAPS in 2D

Kemal Taşkın
Cognitive Sciences Program
Institute of Informatics
Middle East Technical University
Ankara, TURKEY

e-mail: mail@kemaltaskin.net
20.04.2004

Abstract

Self-organizing mapping is an unsupervised learning paradigm used in pattern classification and hence artificial intelligence. This paradigm is based on modifying the class features via the incoming input stimuli. Its exciting part is that it introduces concepts such as neighborhood or mapping. Hence the results obtained from this paradigm highly depend on the selected neighborhood and mapping schemas as well as the distribution of the input stimuli on the feature space. We implemented a self-organizing map simulation tool and simulated the paradigm with variations on these parameters. This paper contains the analysis of the simulations and the discussion of the results.

Key words: Self-organizing maps, unsupervised learning, artificial neural networks, neighborhood

1. INTRODUCTION

All the paradigms of unsupervised learning involve independent class representations; they have no relationships at all. So they are not sensitive to the input distributions and hence subject to an erroneous discrimination. On the other hand in a self-organizing map there are spatial relationships\(^1\) or orders\(^2\) between the classes. These relationships can be considered as in-layer connections of output nodes of a multi-layer perceptron.

The concept is helpful in recognizing the semi-dependent classes visualized as different distributions in the feature space. However it introduces new problems such as the choice of neighborhood and the network-update schemas.

With the spatial relationships defined, the class set becomes a map and the process becomes updating this map with each training data. Technically, with each data the nearest class is calculated and updated with its neighbors such that they become more like that data gradually:

\(l \in [0,1] \) (learning rate)

For each training data \(t_i\)
An ANALYSIS on VARIATIONS of INPUT PATTERN DISTRIBUTIONS in SELF-ORGANIZING MAPS in 2D

- Calculate the nearest node \( n_c \) (representing a class) with a distance function such as Euclidean distance
- Update all the network with the formula:
  \[ d_{ij} = n_i \ast (1 - d) + t_i \ast d \]
  \( d \) is \( d' \ast l \ast g \) where \( d' \) is the Gaussian distance of \( n_i \) and \( n_c \) with respect to the neighborhood schema and \( g \) is the gamma factor (discussed later)
- Decrease learning rate

The problem here is to define the neighborhood and the network update procedure which includes the gradual update of the neighbors and the learning rate of the process. We simulated a self-organizing map (SOM) with two different neighborhood schemas (grid-form & line-form), three different input data distribution in 2D with varying network-update criterion. The results of these simulations are presented in the results section.

2. SIMULATION TOOL

For the simulations we implemented a SOM simulator\(^3\) (SOMSim-2D) in which the simulation parameters can be passed as parameters via a user interface (Figure 2.1). This interface also includes a display panel for the results.

![Figure 2.1: SOMSim-2D user interface. “Loop Count” is the # of the training data. “Gamma” is the Gaussian coefficient for the neighbor distances. “Learning Rate Std. Dev.” is the learning rate coefficient of the network. “Centralization” is the measure of data distribution in terms of centeredness.](image)

3. TESTS & RESULTS

With this simulation tool we first applied an equally distributed training data to the network with different feature space boundaries: Square, triangle, circle. The resultant mappings after 100000 data are below (Figure 3.1).
An ANALYSIS on VARIATIONS of INPUT PATTERN DISTRIBUTIONS in SELF-ORGANIZING MAPS in 2D

Figure 3.1: The resultant network states for three different boundary examples with grid-form neighborhood and gamma=3.0.

The system behaved as expected in square and triangle cases however it shaped a diamond in the circle case which is unexpected. The reasons of this will be discussed in detail in discussions section.

Figure 3.2: Network states for three different data distributions after 1000, 10000, 100000 training data. The system is run with gamma=3.0 and centralization=0.3.
An ANALYSIS on VARIATIONS of INPUT PATTERN DISTRIBUTIONS in SELF-ORGANIZING MAPS in 2D

Figure 3.3: Network states for three different data distributions after 1000, 10000, 10000 training data. The system is run with gamma=1.3 instead of 3.0 and centralization=0.3.

Figure 3.4: Network state comparisons between grid-form neighborhood and line-form neighborhood for de-centralized data after 1000, 10000, 100000 training data with gamma=3.0.

The second class of tests we simulated included the distribution variances of the training data. This class of tests was applied only with the square feature space in order to fix that variable. There were three different distributions: Random, centered and de-centered. The generation of
An ANALYSIS on VARIATIONS of INPUT PATTERN DISTRIBUTIONS in SELF-ORGANIZING MAPS in 2D

the randomized data was a simple algorithm in which the data was stretched towards the centre or outer direction proportional to the “Centralization” parameter on the user interface. The difference in training data distribution yielded different states such that the regions of the feature space with high data density pulled the classes more. The visualization of the network states of these three distributions after 1000, 10000, 100000 data are in Figure 3.2.

The third class of tests was designed to observe the effect of gamma factor. The procedure was the same with the previous class with a gamma value of 1.3 instead of the default value 3.0. The obtained results are in Figure 3.3. The test is then the comparison of the last row of the previous figure with this one. To remind, gamma is the factor used in determining the neighbor pool of a single class. So the choice of gamma decreases or increases the pulling effect of the training data, resulting different convergence states for the network.

The fourth and the last class of tests was designed to observe the effect of the neighborhood schema. We compared the previously used grid-form with the line-form neighborhood. We visualized the difference using the de-centered data distribution. The states of the networks after 1000, 10000, 100000 data are visualized in Figure 3.4.

4. DISCUSSIONS

Although not having an effect on the other tests the failure of the circle boundary example should still be discussed. We used Cartesian distance in calculating the distances between the training data and the classes in the implementation. Using the angle difference in polar coordinates did not give the desired solution either. So we moved to another point of view. Our data distribution algorithm was simply randomizing the data in the polar coordinates:

Point in polar coordinates: (φ,r)
Point in Cartesian coordinates: (x,y)
φ = random(-1.0,1.0)*PI
r = random(0.0,1.0)
(x,y)desired = polar_to_cartesian(φ,r)

We believe that this process makes a centralized randomization. We will be working on an improvement on this algorithm for the next version of SOMSim-2D.

We saw the effect of the distribution change in the second class of tests which were as expected. Moreover we observed that the gamma factor had an effect on the smoothness of the network. The network would act as a whole if a single training data had a larger area of effect (which was proportional to gamma). So a smaller gamma would make the classes act alone and the network would be less smooth and less sensitive to distribution effects. That is actually what to be observed in Figure 3.3. It is not very observable in the centralized case since the centralization effect is so strong that it overrides everything.

5. CONCLUSION

From the results obtained we had the chance of observing the effects of various criteria of self-organizing maps. First, we saw that the distribution change had an effect on the shape of
the network. The region of the feature space with more training data pulls the classes onto itself. Fine corollaries can be driven from this point. For example one can configure and run a SOM before a feature mapping or prototyping process and ease the determination of classes. Moreover one can use a SOM for weight initialization in a multi-layer perceptron.

Second, we observed the effect of the gamma factor. In literal terms, gamma is the collaboration factor of the network. With large gamma values the network will act as a single unit and the nodes (classes) move freely otherwise. In our meta-problem of distribution differentiation it did not have a primary effect. This is because the choice of gamma factor is highly problem dependent. With the example at hand we believe that gamma should be inversely proportional to the number of class expected in a problem since the number of nodes per class will have to decrease with the increased size of classes and the nodes will have to move more independent.

Third, we observed that the choice of neighborhood does not change the convergence of the system if it is inductive (or translative). It only has an effect on performance and visualization. For example the line-form visualization of the de-centralization effect is much better than the grid-form.

To sum up, we visualized the effects of different parameters of self-organizing maps and derived conclusions about them. Not interestingly, just as the other fields of the computer science the parameter setting turned out to be a problem dependent process.

6. REFERENCES

2: Honkela, T. (1998), Self-Organizing Maps In Natural Language Processing
3: SOMSim-2D v1.0 (Self-Organizing Map Simulation in 2-D) [http://thesis.kemaltaskin.net/]