ABSTRACT

ACKNOWLEDGEMENTS

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Learning curves may reflect different underlying learning processes. For example, in insight learning, it may follow slightly fluctuated level and sudden dramatic increase at some points. Because the learning is sometimes the result of experience through personal interactions with the environment. And following the occurrence of insight, abrupt realisation of how to solve the problem can be repeated in future similar situations. This means that the associated experiences with insights can be parameterised, and it can be linked to future behaviours.

Therefore, there are several possible functions which might fit learning curves and represent different learning processes; step function, powerlaw, piecewise powerlaw and etc. Through possible functions and learning curves, people can be grouped into several learning patterns. Moreover, by inspecting people in those groups, 'how they practice' to get the learning curve and 'which features' having an affect upon individuals can be found. Then suggestions for better learning for individuals may be categorised and proposed.

For this, online games are a reasonable instrument. As it involves "rapid perception, decision making, and motor responding", as well as it gives rich details of practice history, in order to find individuals' learning curve and investigate their features (Stafford and Dewar, 2013)

Text...

1.2 Problem Definition

There are three main issues in traditional reinforcement learning curve. Because it has smooth power law with diminishing gain derived from average value, all possible individual learning curve are squashed into one learning curve. Thus, it cannot tell which learning curve will belong to a task, behaviour, and learning process. Furthermore, even though identifying bad or good on learnings are important, but it cannot provide sufficient information about those.

Below is the list of problems on current reinforcement learning curve.

- Not being able to categorise learning process.
- Averaging possible individual learning curves
- Not enough information for identification of learning success

1.3 Aims and Objectives

These are the concepts I think you should cover in your introduction:

Skill acquisition & expertise - factors which are known to influence skill acquisition - practice amount, practice spacing, others

Using games to study skill acquisition

Learning curves - different possible underlying functions - problems with averaging over

The ultimate aim, of course, is to identify individuals who learn faster or for longer, and try and relate this to how they practice The aim of this paper is to fit different functions to the Axon game data. In the data, there are 854,064 individuals data about when, where they did the game and other informations, as well as scores of game according to their attempts. And so identify the underlying learning process. Furthermore, it is to inspect features which may have influences on the learning curves.

- Understand several possible functions; step function, powerlaw, piecewise powerlaw and etc.
- Understand different underlying learning processes; insight learning, associative learning, multi-component/process learning and etc.

- Fit functions to individual Axon game data and compare those to identify.
- Test theories of what makes learning most effective, exploiting unsupervised learning, establish which parameter is important for getting learning curve from the data.
- Design more effective learning practices.

1.4 Project Management

Text...

Chapter 2

Data Curation

A large number of data were acquired through the online game named Axon which is developed by Preload for Welcome trust (?, ?). Totally 1,201,516 machine identities (or players) played the game over 4 million times. The raw data set is fundamentally comprised by the score, date, time of plays and so on in accordance with machine identities which may represent each individual. Undoubtedly, it can be possible to extract information from the data set on how people practiced to get a higher score. For example, it shows how much time they played for each score. However, the raw data set of on-line game seems quite noisy to discern those factors which affect player's learning. It contained many 'undefined' or 'unrepresentable values', and some values are not valid, such as starting play time is later than the time play finished. For this reason, the information on what the analysis needs cannot be identified directly from it.

Thus, the data source needs refining processes until the gemstone of data set could gleam with the evidence of valuable information. In other word, the big data set has to be curated. Stonebraker et al. stated that "data curation is the act of discovering a data source(s) of interest, cleaning and transforming the new data, semantically integrating it with other local data sources, and deduplicating the resulting composite" (?, ?).

Below are the tasks in which steps of data curation are largely expected,

- Cleaning raw data set,
- Incrementally accommodating new data entities,
- Normalising scores grouped by the same players,

- Clustering players, showing the similar patterns.

Especially, data curation tasks followed methods of two previous studies (?, ?, ?). Accordingly, the curation tasks of cleaning, grouping by machine IDs and accommodating will be explained at the following section. After then, another tasks for data deformation and clustering will be examined.

2.1 Data Cleaning

What has to be noted before data cleaning is that it has to be implemented under the grouped data by the same machine IDs. Totally 4,038,802 plays were sorted by players having the same machine IDs, and then 1,201,516 players were detected. The criteria according to the previous two studies (?, ?, ?) are as follow,

1. Players who did not play at least 15 games,
2. Players who attempted more than 300 times,
3. Data which lack valid longitude or timing information for each attempt.

The reason of discarding those who played less than 15 attempts is because those cannot show a causal relationship between attempts and scores. In this step, xxx individuals were deleted. And the reason of second step is that plays more than 300 attempts did not show many differences against after 300 attempts. ??? out of total number of players attempted the game more than 300 times, and also they were removed. Finally, the discarded players having unrecognisable time information were ???. As a result, originally 4, 038,802 plays with xxx players were filtered, and reduced to ??? plays and ??? individuals decreased by ???.

Write about discontinuouty

2.2 Data accommodating

The next step of data curation for the analysis is to incrementally accommodate hidden but obvious features from the previous cleaned data. The data set is constituted by basic 8 categories and 7 increments.

2.2.1 Original variables in the data source

- Machine identities:

  When players access the on-line game, tracking code written by Preloaded recorded machine identities. Machine IDs might be considered as representing individuals who actually played the game. They inserted tracking code that recorded a machine identity each time the game was loaded and kept track of the score and the date and time of play

- Scores and Attempt numbers:

  Each player's attempts at the game and scores were recorded in the order of the time.

- Date, Hour and Minute:

  Time information when the game was loaded were recorded. **(Q. what's the information of this?)**, **(Q. what's the information of this?)**

- Latitude and Longitude:

  The information of locations where the machine ID accessed the game was collected approximately at the maximum of city-block level.

2.2.2 Hidden variables calculated from the original variables

- Time set:

  Date, hour and minute information were combined together to obtain timing between attempts.

- Time difference:

  Time differences were calculated by comparing time to the time of its previous attempt. Therefore, the first attempts do not have values of time difference.

- Local time:

  Using the formula, "$local\ time = UTCtime + (longitude24 \div 360), modulo24$ **(Q. understand the formula)**", local times for each play were calculated (Stafford et al., 2016) in seconds. There is data that local time could not be calculated because of missing latitude and longitude information. Calculating the local time facilitates comparison of players in different time zone.

- Gap types:

  On the basis of time difference, Stafford et al. (2016) classified individuals into 4 groups, ”according to the nature of the timing of their first 15 attempts at the game”. First group is ’no gap’ for those who had less than 15 minute gap between each play. Second group is ’wake’ in which players are assumed having a rest in there waking hours if they have ”a single gap between 7 and 12 hours”. Working hours is time between 5 am and 12 pm. And those who rested for between 7 and 12 hours in working hours were categorised as ’sleep’. Finally, all other individuals are classified as ”no category”.

- Long gap, Short gap and Sleep gap:

  For more detailed analysis, ’rest’ which is defined in ’Gap types’ categorisation was divided with 3 types. Long gap, short gap and sleep gap were determined when players had a rest ’between 7 hours and 11 hours’, ’below 15 minutes’, and ’not in working hours’ respectively.

- The number of total plays:

As the term of hidden variables, there are possibilities that more factors can be mined from the data set and be incrementally accommodated to it if appropriate methods are implemented.

2.3 Data Deformation

CSV format has been used as a basic data file format for the previous data curation tasks. The main roles of CSV format are loading data-set from the data storage space, processing, reprocessing, and storing. Simply, the data set needs to be stored somewhere in data storage unit of machine(s) where the analysis processes would operate. CSV is an abbreviation of Comma-Separated Values. A data record of CSV format file is a line formed by one or more elements, and commas separate those element values to be distinguishable (Shafranovich, 2008). Saving data in the CSV format has several benefits most of the time: ease to generate, read and edit manually, versatility in most programming languages, and so on (Idris 2014). However, these are true only when data size is manageable. Because fitting all data-set of such a big data in computer memory takes a lot of spaces. In some case loading on the memory cannot be possible depending on computer capability and setup. And taking large space would mean processing will be slow. Thus, CSV format is not very efficient way to manage such a big data (Idris 2014). To solve with those problems, two approaches of being able to reduce storing spaces and of enabling faster speed for analysing were applied; converting CSV file format into Pickle format, and splitting data set into several pieces.

2.3.1 Pickle file format

There are various file formats which can provide ”a high level of data compression such as zip, bzip and gzip” (Idris 2014).

2.3.2 Hash table

Hash table, **cut 15 attempts**.

2.4 K-Means Clustering with Competitive Learning Algorithm

In order to identify learning curves of individuals, the first thing to do is clustering players into several groups in which they show similar learning patterns. Traditional ways of identifying individual learning curves is either averaging all data of test subjects or fitting different possible underlying functions to observed data-set (Newell & Rosenbloom 1981; Gallistel et al., 2004; Donner & Hardy, 2015). Well represented learning curve in psychology may be a ”smooth power law of diminishing gains” (Donner & Hardy, 2015). For example, Howard (2014) tested function fitting on chess performance and verified a ”power function fit best”.

However, in contrast, Gaschler et al. (2014) stated that ”the exponential function was better than the power function fitting” for learning curves of chess players. This different results in the same observation of chess game would stem from that the power law is an artifact which averages many divergent shapes of exponential curves (Heathcote et al., 2000; Murre & Chessa, 2011) as well as other possible underlying curves.

Therefore, before analysing the on-line game data set, it would be better assumption that there are many different shapes of learning curves depicting individuals' learning best, rather than only one learning curve.

In contrast with classification tasks that classify labeled data-set, clustering is an unsupervised classification task which automatically groups similar data without predefined and labeled classes (Hackeling, 2014), and it generates the same result as classification algorithm does (Harrington, 2012). Among several clustering methods, K-Means clustering was used with artificial neural network algorithm, competitive learning. It is because K-Means clustering algorithm in high dimensions is computationally very slow and the key method of K-Means; comparing distance in 2 and 3 dimensions, would not work in high dimensions (Daume III, 2012).

Hence, there is possibility that the data-set may not properly be clustered, as the on-line game data-set is comprised of 15 dimensions. Compensating K-Means clustering with neural network algorithm might provide one of solutions for the high dimensions data-set.

2.4.1 K-Means clustering

The name of K-Means is because it discovers k unique clusters, and the mean values of that clusters represent the centres of each cluster. The key idea of K-Means clustering algorithm is to compare distances between all data and centres of clusters. If centroids of clusters were found, each data can be assigned to its nearest centres. In the same manner, centres of clusters can be computed after each data is allocated to clusters.

Again, there are no labels or predefined information of classes in the data set. Thus, the information of which approach has to be implemented first between two approaches is unknown. The solutions are to start with initial random point, to iterate calculation of centre position, and to rearrange elements in clusters. Through iterating these three approaches until stopping criteria is satisfied, k means eventually converges to local optimisation, and comprise k clusters.

There are two terminating criteria in K-Means clustering algorithm. The first one is using a threshold for the difference of cost functions (2.2) through subsequent iterations (2.1).

$$J_i - J_{i-1} < \theta \text{where, } j \in K, \tag{2.1}$$

and cost function J is

$$J(\mu, \mathbf{C}; \mathbf{D}) = \sum_{n}^{N} ||x_n - \mu_{C_k}||^2 = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2, \tag{2.2}$$

where μ is mean of cluster, **D** is (the number of parameters) data-set, **C** is clusters in the data-set, x is data, N is the number of data, and K is the number of clusters. The other one is to use a threshold for the difference of centre positions between successive iterations (2.3).

$$x_i - x_{i-1} < \theta \quad \text{, where } i \text{ in } C_k. \tag{2.3}$$

Below is pseudo-code of K-Means algorithm,

```
for all $k$ such that $0 \leq k < K$ do
    randomly initialise $\mu_k$
end for
while stopping criteria (2.1) or (2.3) do
    for all $k \in K$ do
        for all $i$ such that $0 \leq i < N$ do
            $\mathbf{C}_k \leftarrow \arg\min_x ||\mu_k - x_i||^2$ ,
        end for
    end for
    for all $k \in K$ do
        $\mu_k \leftarrow \text{mean}(\mathbf{C}_k)$
    end for
end while
return $\mathbf{C}, \mu$
```

At first, the positions of centroids are randomly initialised. Then, the first for loop in the while loop assigns each data to nearest clusters. The second for loop calculates the centres of clusters. By iterating these two tasks, all centroid of clusters move slightly towards the centres of each cluster, and eventually converges to local optima. For online game data analysis, stopping criterion (2.3) was used. This K-Means clustering algorithm was improved with neural network, competitive learning algorithm, in order to cluster the on-line game data-set in high dimensions.

2.4.2 Competitive learning

Competitive learning is unsupervised learning approaches in which the output neurons contend against each other for the one firing to react from the given pattern in inputs, and therefore only one unit take winner upon all output units (2.4.2). This exclusive phenomenon in synaptic neurons is called 'winner-take-all units', and sometimes they also called 'grandmother cells'. Which means a neuron fires on a specific stimulus. This neural networks has an advantage of clustering unlabeled data, searching the correlations of the input data. One of the most significant application of competitive learning might be vector quantisation. In which, input data-set is a bunch of vectors in data space. And input vectors are compressed and stored by transmitted into prototype vectors.

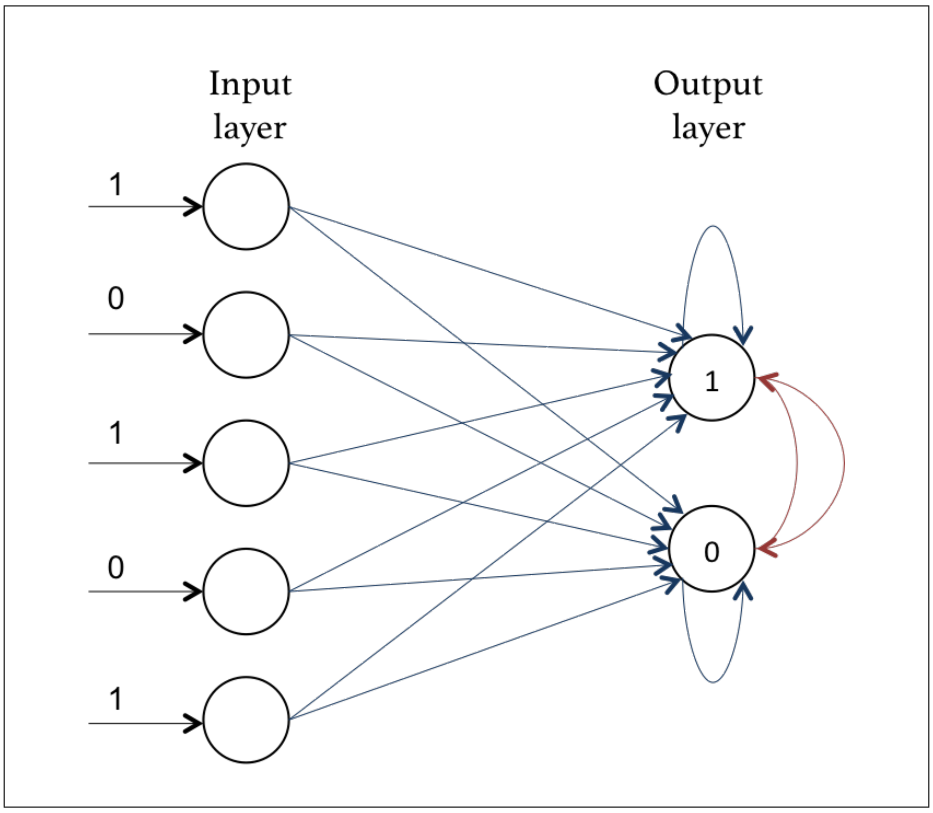

Figure 2.1: Competitive neuron diagram. When an input enters the network, it generates output values according to weights between an input and output neuron (blue arrows). Output neurons compete each others (red arrows), then only one output fires on an input pattern. Output neurons are considered only having binary 0/1 output

It has very similar concept to K-Means clustering algorithm, and the prototype vectors can be interpreted as centroids in K-Means clustering. Besides, vector quantization employs Euclidean metric to assign an input vector to the nearest prototype, weight vector. When an input data, usually represented as a vector, is applied at the input of the network, the winner neuron indicates the appropriate class, and if all input data are entered to the network, the input space organises a Voronoi tessellation. To be specific, Voronoi tessellation which was used in the analysis is Centroid Voronoi tessellation, in which means are positioned in each Voronoi cell (center of mass). The weight vectors (or centroid in K-Means) are the vectors of prototypes.

Output neurons O_i in (2.4) are the units having the net input in an pattern at a time,

$$O_i = \sum_j w_{ij} x_j = \vec{\mathbf{w}}_\mathbf{i} \cdot \vec{\mathbf{x}}, \tag{2.4}$$

for an input vector $\mathbf{x}$, and winner neurons are found by (2.5),

$$\vec{\mathbf{w}}_k^T \vec{\mathbf{x}}^\mu > \vec{\mathbf{w}}_i^T \vec{\mathbf{x}}^\mu \quad \text{for all i}, \tag{2.5}$$

where winner neuron is $\vec{\mathbf{w}}_k^T$, and $\vec{\mathbf{x}}^\mu$ is an input of pattern μ. (2.5) tells winning unit is the biggest. And if the weight vectors connecting between each input and output are normalized, $\vec{\mathbf{w}}_\mathbf{i} = 1$, then (2.4) can be redefined as below,

$$|\vec{x}^\mu - \vec{w}_k^T| \leq |\vec{x}^\mu - \vec{w}_i^T|, \quad \text{only if } |\vec{w} = 1| \text{ for all i}. \tag{2.6}$$

(2.6) shows similarity to the key idea of K-Means clustering, which only consider distances between every data and centroids.

Competitive learning only consider the output neuron with the maximum output from net value of inner product between input and weight. Biologically, output neurons are connected with other all output neurons with lateral inhibition, as well as excitatory connection itself, and therefore winner takes all units.

For competitive learning, the thing has to be remembered is that weights have to be normalised in order for all units to have a unit length.

2.4.3 Normalisation

For most machine learning algorithm, normalisation is an important for analysing data with different scale to be compared and then to derive information in K-Means clustering (Michael 2015). To be specific about normalisation for competitive learning, network exploit the inner product for each input with weight vectors in order to identify winner output unit. If looking inside of (2.5), the equation for inner product is

$$\vec{\mathbf{w}}^T \cdot \vec{\mathbf{x}}^\mu = \vec{\mathbf{w}}^T \vec{\mathbf{x}}^\mu \cos(\theta), \tag{2.7}$$

with respect to the lengths of both units and a direction between input pattern and weight. Non-normalised one has no way for an criterion of competing output units with weight /vectors. Because the neuron which has the most largest value in length always wins, and that means the algorithm will finish without any learning or no informative outcome through iterations in the both case of not-normalised data-set and weight vectors.

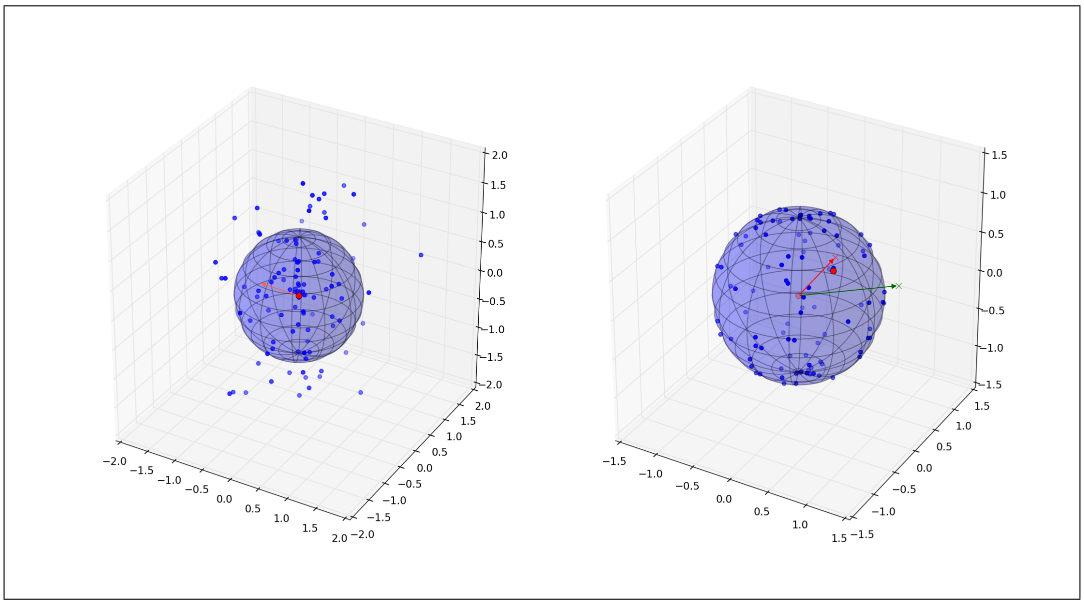

Figure 2.2: Examples of not-normalised data and weight in 3 dimensions. In both left and right figures, sphere represents unit length area. Left is the plot of original data with a weight (red arrow). The right one is the figure that has normalised data-set onto the unit length area, but not-normalised weight vectors (green arrow).

For the solution of that, normalising input and weight vectors simply transforms (2.7) into the equation computing only a direction between those vectors, which tells how closely weight vector is from the input vector.

Eventually, the inner product in (2.6) becomes the criterion which detects the shortest Euclidean distance by comparing angles between input and weight vectors, (2.8).

$$\begin{aligned} \vec{\mathbf{w}}_k^T \cdot \vec{\mathbf{x}}^\mu &> \vec{\mathbf{w}}_i^T \cdot \vec{\mathbf{x}}^\mu \\ \cos(\theta_k) &\geq \cos(\theta_i). \end{aligned} \tag{2.8}$$

The firing output neuron is still has the largest value among output units when the θ is smaller than others based on the competition with proximity of the input vector.

(2.4.3) shows not-normalised issues in competitive learning. In the tot-normalised data-set and weight vector, (2.7) always compare only the length, not the closest one in the distance between data and weight vector. It is obvious that the most long vector always win, which cannot be Representative for input patters, and therefore weights would not move to the centre of Voronoi cell.

2.4.4 Procedure of the algorithm

After normalising data-set and setting weight vectors with unit length, the competitive learning algorithm for pattern classifier exploiting (2.4) and (2.8) follows as below,

1. choose weight vector with random values:
   Similar to K-Means clustering, weight vectors which implies center of mass for clusters (or Voronoi cells) are initialised with random values. However, the issue of dead unit would be caused by randomly initialised weight vectors. Dead unit is the unit which is very far from any input patterns, and therefore never fires. And other problem is the optimal number of weight vectors. This problem will be discussed in the next section. The last thing to be noted is that weight vector must have unit length.

2. apply input patterns to the network in turn or in random order:
   After setting initial weights, input data vector as an pattern is entered to the network. It does not matter applying it in turn or in random order. The advantage of competitive learning over K-Means is that K-Means comparing all data, then must be slow, but competitive learning can run in random order input data until it approaches the sufficient learning rate. Therefore it is much faster than K-Means, on condition that how learning rate is prescribed.

3. discover a winner output unit for each input:
   When an input vector is applied to the network, which output unit is likely to

take winner position for each normalised input pattern is found through (2.8). For a given input patter, the winner unit has the largest value among other output neurons. In other world, winner-take-all unit is the most closest vector to the input vector with the smallest angle. Inhibitory and self-exitatory connection are definded biologically, but alternatively using python command, **numpy.argmax**, just maximum value is searched.

4. Update the winner neuron's weight to be closes to the input vector:
   For each input, winner $\mathbf{w_k}$ is found among the outputs, and then it is updated to be closer by an allowed amount with learning factor η to the current input vector. After continued update of the weight for the winner unit, weight vectors will be positioned at the center of Vornoi cells which shows similar patterns of inputs. Below is on-line and batch update rule,

$$\begin{aligned} \text{on-line rule}: \Delta\vec{w}_k &= \eta(\vec{x}^\mu - \vec{w}_k) \\ \text{batch rule}: \Delta\vec{w}_k &= \eta\Sigma_{\mu\in C_k}(\vec{w}_k - \vec{x}^\mu) \end{aligned} \tag{2.9}$$

   On-line learning rule updates the change of one data point at a time. However, batch learning rule accumulates weights until the all inputs in a cluster are passed to the network, and sums all of those constantly. Both converges the same centroids of clusters, but batch rule is slower than on-line learning rule which slightly fluctuates towards local optima when updating.

As mentioned in procedure 1, choose weight vector with random values, the two possible problem; the number of clusters and dead units, of competitive learning have to be handled with proper method. The next section will introduce breakthroughs which can tackle those two issues.

2.4.5 Local optima and optimal number of clusters

Sometimes, a number of mean values might be randomly initialised very nearly to each other. And centroids would converge the same point in the same cluster. In this case, clusters would not describe data-set informatively. This issue is called local optima problem. Unfortunately, there is ”no way of knowing what the right answer is” for the outcome of clustering algorithm, but ”repeating the algorithm dozens or even more times" can increase reliability of clustering result (Hackeling, 2014). By repeating clustering algorithm several times, the starting random data points will be set differently and converge different or the same local optima as subsequent outcomes. Clustering results are compared relatively with repeated results, and the most frequent similar results can be presumed as reliable result.

The another crux of K-Means clustering is that the optimal number of clusters is unknown. There are XXX players in the on-line game data-set, and it has no predefined information of how many groups which shows similar learning patterns are. To estimate optimal number of clusters, Elbow method with Akaike Information Criteria (eq:AICandBIC) was used. The criteria which can determine the number of cluster is as follow,

$$\begin{aligned} &\text{Bayes Information Criteria (BIC)}: \\ &\qquad \arg\min_K (J_K + K \log D) \\ &\text{Akaike Information Criteria (AIC)}: \\ &\qquad \arg\min_K (J_K + 2KD). \end{aligned} \tag{2.10}$$

More a number of clusters, less the value of the cost function, J_K. Because, increasing K can divide the data-set into smaller pieces of clusters than the actual number of how many clusters necessarily exists in it. Usually, using the Elbow method shows changes of the value of cost function. It go down significantly at the first some K, and suddenly slow down the change. By inspecting the point showing from dramatic to gradual change, generally the optimal number of K is founded. However, when elbow method for the on-line data-set is plot, this inflection point cannot be determined. Because all costs decrease gradually, Fig. 2.4.5. Therefore, AIC was exploited with elbow method. If plotting AIC values in different number of K, it decreases by the certain amount that is not worth to decrease K more, and then start to increase. It is because 2KD or $K \log D$ proportionally compensates decreasing the value of cost function for K. In high dimensions, AIC criteria gives "high penalty for many clusters" (Daume III, 2012). As a result, K value which minimizes AIC or BIC (2.10) could be considered as the optimal number of clusters.

Fig. 2.4.5 shows the information of elbow method with AIC and BIC for the on-line data-set. It can be seemed that the data-set can be clustered into n clusters. Difference

AIC_BIC_Elbow.png

Figure 2.3: Comparison between AIC and BIC according to the different number of k. AIC is more strict method than BIC to identify the number of clusters.

between xx K and xx K is xx and yy K and yy K is yy. If the value of ccc is set as a threshold for these differences, increasing K beyond 11 would not improve the result of clustering.

2.4.6 Dealing with dead unit

Dead unit problem may occur when weights are initialised very far from any input vectors. Because those cannot win against those which are closer to input vectors than them. (2.4.6) shows geometrically more precise understanding. There are two arrows with different colours. Red arrows are initialised very close to two input groups than green arrow. When those are implied to the network, green weight will never be updated, while red weight will move to the centre between two input groups.

There are several solutions to deal with dead units problem.

1. Leaky learning (LL): LL adjusts learning rate of both winer and loser, and give opportunity for loser to fire in the future, by moving towards more compact cluster.

$$\begin{gathered}
w_{ij}^{new} = w_{ij}^{old} + \eta_{winner}(x_i - w_{ij}^{old}), \\
w_{ij}^{new} = w_{ij}^{old} + \eta_{loser}(x_i - w_{ij}^{old}), \\
\text{where } \eta_{winner} \gg \eta_{loser}.
\end{gathered} \tag{2.11}$$

   find reference.

2. Add bias (AB): AB adds bias to repeated winner outputs so that they are hard to fire again after as the weight updates iterate. In [3], the equations for AB were referenced,

$$\begin{aligned}
f_j^{new} &= f_j^{old} + \beta(z_j - f_j^{old}), \\
b_j &= \gamma(1/n - f_j), \\
w_{ij}^{new} &= w_{ij}^{old} + \alpha(x_i - w_{ij}^{old})z_j, \\
y_j &= \Sigma_i w_{ij} x + b_j,
\end{aligned} \tag{2.12}$$

   where z_j: selector having 0 for loser or 1 for winner, γ, β: constant factor, n: the total number of nodes, f: output neuron's firing rate, b: bias. Performance of competitive learning can be controlled by regulating the parameters[3].

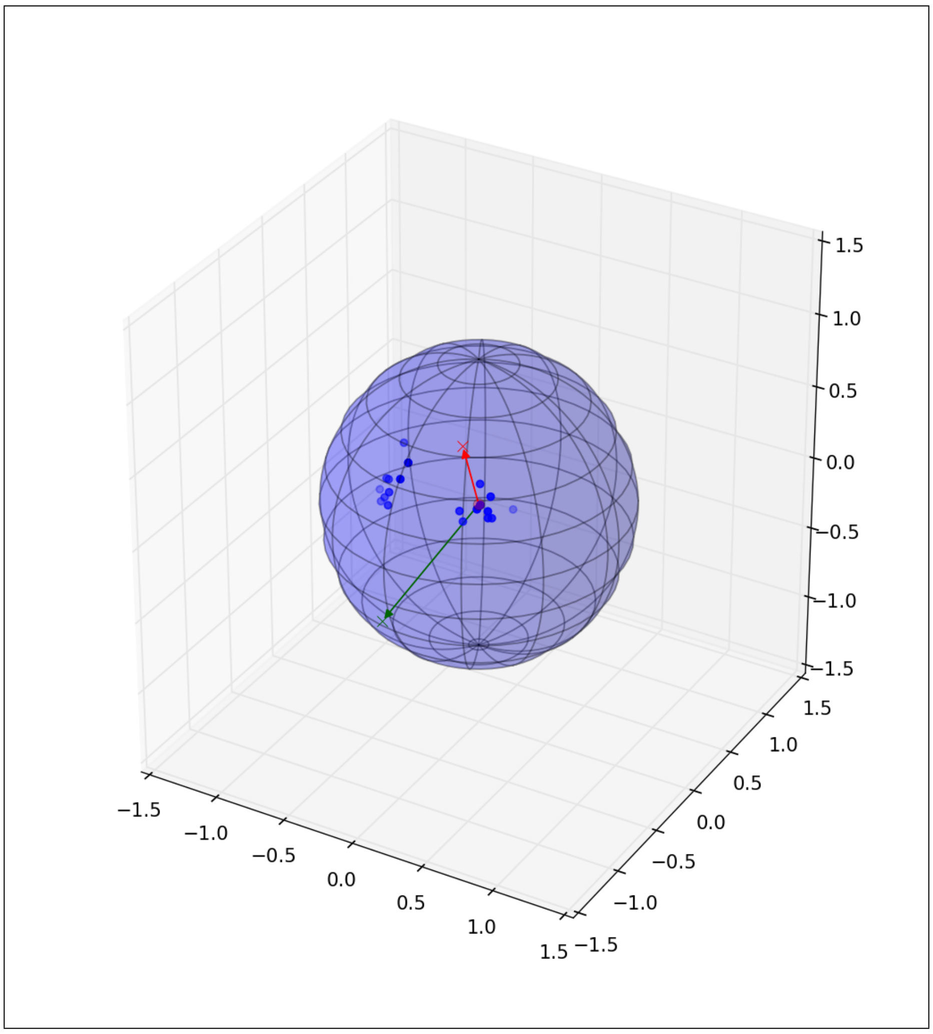

Figure 2.4: An example of dead unit problem. Red and Green arrows are initialised weight before applying into the network. Green arrow is set far from both two input clusters, and it is assumed as a dead unit. Unfortunately, green arrow will never fire through the learning, and which means that input vectors are not properly clustered.

3. Add noise (AN) to outputs in updating: By adding randomly distributed gaussian distribution into output neutrons in every learning time, frequently winner neurons may be disturbed not to fire, and then even when a winner and losers are obvious by (4), a loser may take top place instead of the winner.

$$\begin{gathered} y_j = \Sigma_i w_{ij} x_i + \varepsilon, \\ where\ \varepsilon \sim N(0, \sigma^2) \end{gathered} \tag{2.13}$$

4. Set weights with random samples from input vectors (SV): solution 1), 2), and 3) above use initial weight values in random, there is the risk that some weights have the value very distant from any clusters. To prevent this problem, SV has initial values randomly taken inputs. Therefor, initial weights alway would always locate in some clusters, and dead unit would never be issued. However, there may be the problem that some initial weight values are sampled some same clusters, and other clusters would remain being excluded. Thus, Gaussian distributed random values with large (LSD) or small (SSD) standard deviation has to be added to the initial weight sampled from inputs so that its vectors have appropriate positions near to clusters of inputs or, sometimes, outside of those.

$$\begin{gathered} w_{ij}^{init} = w_{ij}^{random} + \varepsilon, \\ where\ \varepsilon \sim N(0, \sigma^2) \end{gathered} \tag{2.14}$$

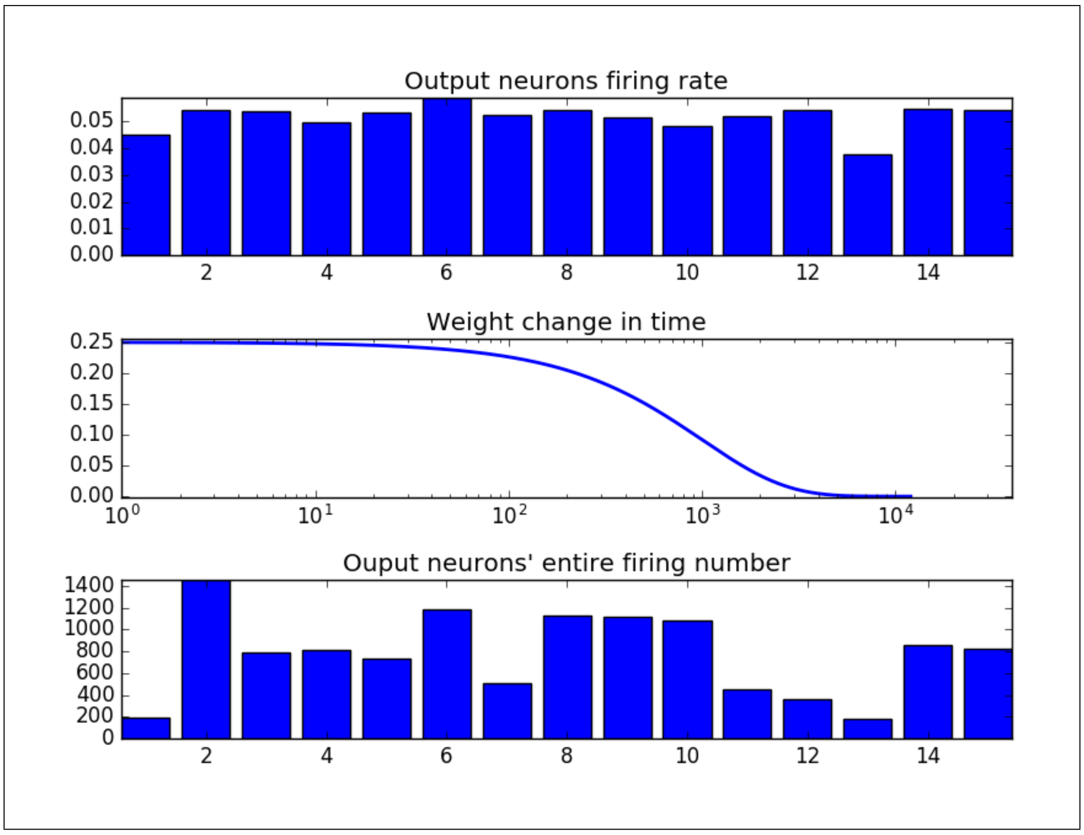

Figure 2.5: Comparison between AIC and BIC according to the different number of k. AIC is more strict method than BIC to identify the number of clusters.

2.4.7 Result

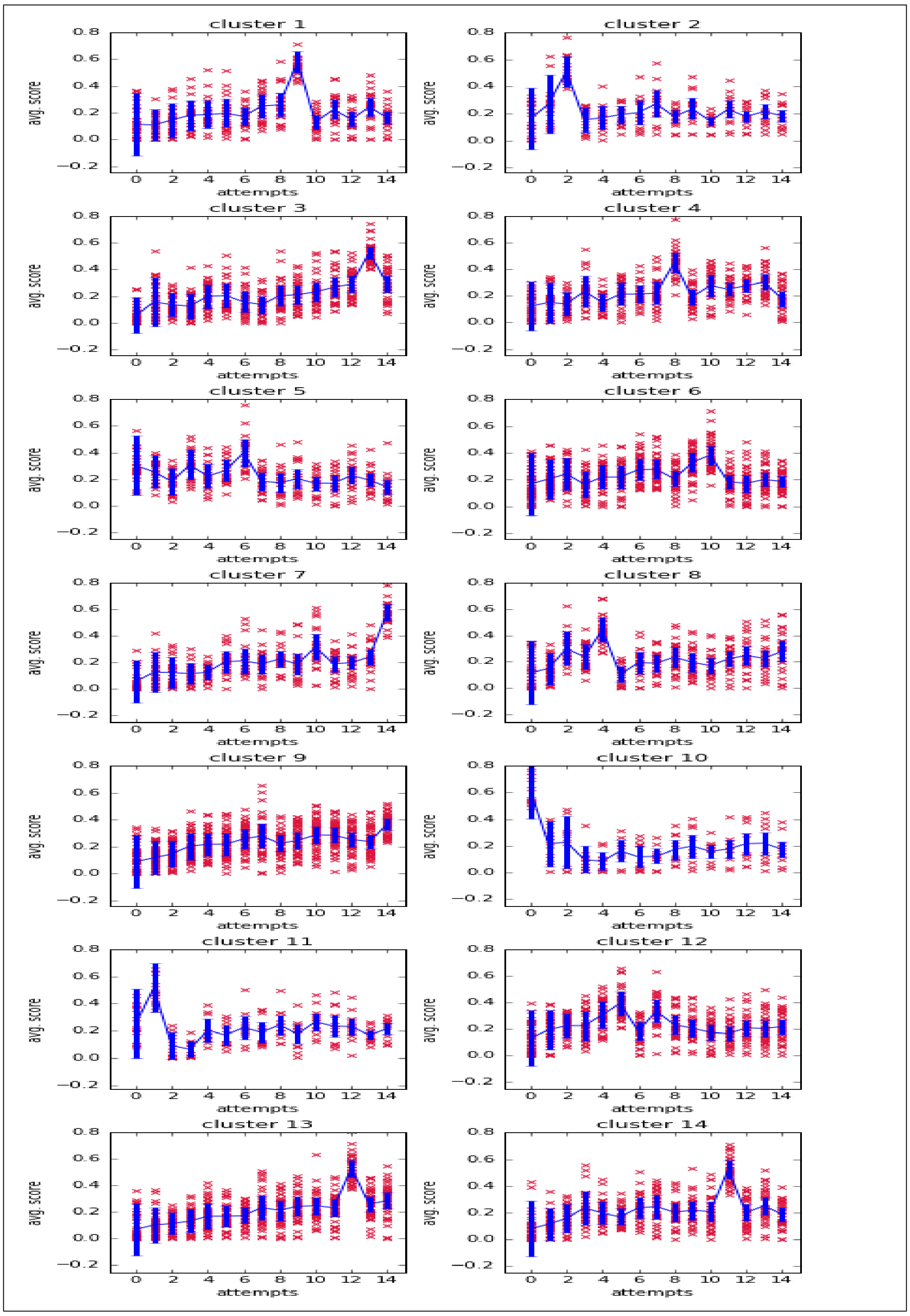

Figure 2.6: 10 Clusters

Chapter 3

Function Fitting

References

Bowles, M. (2015). *Machine learning in python: essential techniques for predictive analysis*. John Wiley & Sons.

Daumé III, H. (2012). A course in machine learning. *chapter*, *5*, 69.

Donner, Y., & Hardy, J. L. (2015). Piecewise power laws in individual learning curves. *Psychonomic bulletin & review*, *22*(5), 1308–1319.

Gallistel, C. R., Fairhurst, S., & Balsam, P. (2004). The learning curve: implications of a quantitative analysis. *Proceedings of the national academy of Sciences of the united States of america*, *101*(36), 13124–13131.

Gaschler, R., Progscha, J., Smallbone, K., Ram, N., & Bilalic, M. (2007). Playing off the curve-testing quantitative predictions of skill acquisition theories in development of chess performance. *Psychological perspectives on expertise*, *20*(40), 137.

Hackeling, G. (2014). *Mastering machine learning with scikit-learn*. Packt Publishing Ltd.

Harrington, P. (2012). *Machine learning in action* (Vol. 5). Manning Greenwich, CT.

Heathcote, A., Brown, S., & Mewhort, D. (2000). The power law repealed: The case for an exponential law of practice. *Psychonomic bulletin & review*, *7*(2), 185–207.

Howard, R. W. (2014). Learning curves in highly skilled chess players: a test of the generality of the power law of practice. *Acta psychologica*, *151*, 16–23.

Idris, I. (2014). *Python data analysis*. Packt Publishing Ltd.

Murre, J. M., & Chessa, A. G. (2011). Power laws from individual differences in learning and forgetting: mathematical analyses. *Psychonomic bulletin & review*, *18*(3), 592–597.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition*, *1*, 1–55.

on-line Python document. (n.d.). 12.1. pickle python object serialization [Computer software manual]. (Retrieved August 16, 2016, from https://docs.python.org/3/library/pickle.html)

Shafranovich, Y. (2008). Common format and mime type for comma-separated values (csv) files.[sl], 2005. *Disponıvel em:¡ http://www. ietf. org/rfc/rfc4180. txt*.

Stafford, T., & Dewar, M. (2014). Tracing the trajectory of skill learning with a very large sample of online game players. *Psychological Science*, *25*(2), 511–518.

Stafford, T., & Haasnoot, E. (2016). Testing sleep consolidation in skill learning: a field study using an online game. *Topics in Cognitive Science*.

Stonebraker, M., Bruckner, D., Ilyas, I. F., Beskales, G., Cherniack, M., Zdonik, S. B., . . . Xu, S. (2013). Data curation at scale: The data tamer system. In *Cidr.*

Stuart, P. (2009). Axon - a game for science [Computer software manual]. (Retrieved February 3, 2014, from http://preloaded.com/axon-game-science/)