UNIVERSITÉ DE MONCTON

DÉPARTEMENT DE MATHÉMATIQUES ET STATISTIQUE

Identifying Gambling Personae Through Machine Learning Algorithms

Éloïse SOUCY
Salah EL ADLOUNI
Sophie LÉGER AUFFREY
Jalila JBILOU

Presented to :
MATHEMATICS STATISTICS AND COMPUTER SCIENCE
GRADUATE STUDENT CONFERENCE
CHARLOTTETOWN, PRINCE EDWARD ISLAND
OCTOBER 13-14, 2023

1 Introduction

There is a growing need for gambling disorder studies. Especially with the easy access of online gaming which made gambling more accessible than ever [2]. This research aims to identify distinct gambling personae through machine learning algorithms and could potentially lead to a model that could detect potential pathological gamblers in the early stage by identifying the playstyle.

2 Related theories

2.1 Uniform Manifold Approximation and Projection(UMAP)

UMAP is a recent dimension reduction technique that constructs a theoretical framework based on Riemannian geometry and algebraic topology. Dimensionality reduction is the process of reducing the number of features (and dimensions) in a dataset while keeping as much of the variation in the original dataset as possible [16]. We usually perform dimensionality reduction before training a model. This can increase efficacity of the model.

The first part of UMAP is the construction of a fuzzy topological representation. The second part is the optimization of a low dimensional representation that most resembles the fuzzy representation as possible, measured by cross entropy [9][7]. UMAP is similar to another technique called t-SNE [5] because they both build a neighbours graph in the original space of the data and tries to find a similar graph in lower dimensions. Instead, other techniques sush as PCA [15] factorize a matrix characterising the data to reduce the dimension [9][10].

First, let's construct a fuzzy representation. We use simplicial complexes to construct topological spaces out of groups of points. K-simplex are constructed in a geometrical way with $k+1$ independent points. Each point form a 0-simplex and they can be connected to a neighbour within a fix radius to form a 1-simplex. Three points connected together form a 2-simplex and so on [14].

Figure 1: Low-dimension simplices [9]

The formed simplices are put together to form simplicial complexes κ. To form the simplex, we need to learn about the topology of the space. Assuming that the data are drawn from an underlying topological space, we need to generate an open cover of it. In a metric space, we can approximate an open cover as simply creating balls of some fixed radius centered at each point. Simple neighborhood-graph and laying it into a lower dimension doesn't provide a theoretical justification as using the topological approach provides [10]. Another advantage, of using the topological approach is that it leads to easier computation. A theorem called Nerve theorem proves that simplicial complex can be projected in a lower dimensionality graph by being equivalent to the union cover [11].

Thus, working with finite data, the fixed radius does not always cover the whole manifold. If the data is uniformly distributed along the manifold, then a radius the size of a little above half the average distance between points should be enough. In most real world problems, the data will not be uniformly distributed [10]. As a result, in low density regions, the gap between points could be larger than the radius, thus not being able to form a simplex. In high density regions, there could be a lot of neighbours in a given radius making everything far too connected. The way to fix these problems is to use a variable radius; smaller in high density regions and bigger in low density regions. A variable radius is mathematically supported by the definition of a Riemannian metric on the manifold. The choice of a hyperparameter k, which is the number of neighbours, will determine the local radius by

estimating the density. If the choice of k is high, the global structure will be more conserved and the local radius will be larger. In contrary, if we choose a smaller k, then the local radius will be smaller, and the local structure will be preserved [9].

A local notion of distance is calculated for each point by assuming the data is uniformly distributed on the manifold. The radius of the ball is stretched until it reaches the k-th nearest neighbour of the point. Each point has its own distance function. Consequently, there is a single radius that will respect each point's local distance.

By working with a Riemannian metric-based approach; having a local distance for each point, we can weigh the edges of the graph by how far the points are on the edges. In other words, we can work with a fuzzy topology where an open set is not a binary yes or no, but a fuzzy value between zero and 1 [3]. It is not required that all points are connected together but no points should be completely isolated. The open set should extend as far as the closest neighbour and the fuzzy confidence can decay beyond the first neighbor.

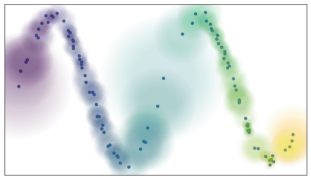

Figure 2: Fuzzy open balls representation with a locally varying metric [9]

$$\text{similarity score} = e^{-(\text{raw distance} - \text{distance to nearest neighbor})/\sigma}$$

The similarity score with the nearest neighbour will always be 1. The parameter σ will ensure that the sum of the similarity scores is equal to the logarithm base 2 of the number of neighbours.

Another problem occurs when points have their own distance metric. From point a's perspective the distance to point b could be 1.5, and on the perspective of point b, the distance to a could be 0.4. Having fuzzy simplicial sets, the best way to find the fuzzy union is to combine the weights like this: $a + b - a \cdot b$. These combined weights give us a probability that at least one of the edges (a with b or b with a) exist [3].

Now that we have a fuzzy representation of the data, let's see how we can put this in a low-dimensional representation. UMAP initialize a low-dimensional graph using spectral embedding. This time, the low dimensional representation will be lying on the Euclidean manifold and the distance metric will not be variable. UMAP calculate the similarity scores using a fixed, symmetrical curve based on a t-distribution.

$$\text{Low d. score} = \frac{1}{1 + \alpha d^{2\beta}}$$

α and β give us control on how packed we want our points in the low-dimensional graph. UMAP randomly selects two points in a neighbourhood proportionally to their high-dimensional similarity score. And then, randomly pick one of them to move closer to the other by calculating the low-dimensional similarity score. That similarity score is the "neighbour" score. UMAP selects a third random point, but in a different neighbourhood and not proportionally to their high-dimensional similarity score. The low-dimensional similarity score is calculated between the last point selected and the moving point. That similarity score is the "not neighbour" score. Those two low-dimensional scores help UMAP evaluate where to move the moving point using the minima of a cost function.

$$\text{Cost} = log\left(\frac{1}{\text{neighbour}}\right) + log\left(\frac{1}{1 - \text{not neighbour}}\right)$$

Knowing there are a lot of points to move, UMAP is using stochastic gradient descent to find the optimal low-dimensional graph. The randomness caused by using Stochastic Gradient Descent means that each time the UMAP algorithm is run, the graph might look a little different [1].

2.2 *K*-means

Clustering algorithms are unsupervised learning models that find a division in the unlabeled data where similar data tends to be put into the same clusters [13]. *K*-means is a clustering algorithm that assign the data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid is at the minimum [8].

Let's denote the centers μ_k with $k = 1, ..., k$, x_n the data points and r_{nk} their belonging to cluster k:

$$r_{nk} = \begin{cases} 1 & \text{if } x_n \text{ is in } k \\ 0 & \text{otherwise} \end{cases}$$

Let's consider the objective function

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2$$

The steps of *k*-means are [4]:

Step 1 The selection of k random data points to be considered as the center of the k clusters (μ_k).

Step 2 The distances between centers and all the other data points is calculated. Each data points are assigned to the closest center point's cluster. In another perspective, μ_k is fixed and we find r_{nk} by differentiate J with respect to r_{nk}.

$$\frac{\partial J}{\partial r_{nk}} = \sum_{n=1}^{N} \sum_{k=1}^{K} ||x_n - \mu_k||^2 \qquad r_{nk} = \begin{cases} 1, & \text{if } k = argmin_j ||x_n - \mu_j||^2 \\ 0, & \text{otherwise} \end{cases}$$

Step 3 For each cluster, we replace the center for the point that is closer to the real cluster's gravity center. In other words, the r_{nk} is fixed and μ_k is calculated by differentiating J with respect to μ_k.

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \qquad \mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}}$$

Steps 2 and 3 are repeated until:

- The in-class variances stop decreasing.
- The interclass variances stop increasing.
- The data points stopped getting assigned to another cluster.
- A maximum iteration limit was set and reached.

In practice, the iteration should not be numerous. When one of the first three conditions are met, it means that the algorithm converged to a local optimum but not necessarily optimal. Therefore, it is recommended to run *K*-means a few times and select the one that has the lower sum of squared distance. If one of the clusters has gone empty during the algorithm process, we can pick a random point to form a new cluster's center [13].

2.3 Support Vector Machine (SVM)

Classification methods are supervised machine learning, meaning that the training of a classification model is based on labeled data. The algorithm predicts which class a new observation belongs to based on the observed data and their membership [13].

Support Vector Machines are simple algorithms that can be used for both regression and classification tasks. The objective of the Support Vector Machine algorithm is to find a hyperplane in an N-dimensional space that will seperates the points.The function of a linear decision boundary function between two groups, in a two dimensional graph is:

$$\sum_{j=1} x_j w_{ij} + w_{i,0} = 0$$

with the weights w_{ij}.

Having many possible decision boundaries, SVM uses a constraint so that there is only one outcome: the margin distance between points and the hyperplane is maximal [13] [6]. Maximizing the margin increases the predictive accuracy of a future data point.

In this research, we will discuss building a hyperplane between two groups. For cases where points can belong to 3+ groups, it's possible to build more than one hyperplane.

Let's consider the case when the data is linearly separable. Let x be the data related to a group $r\epsilon\{-1,+1\}$. Maximizing the margin ρ subject to minimizing the distance between the margin ρ and the data x can be written mathematically as:

$$\frac{w^t x^t + w_0}{||w||} \geq \rho$$

If we fix the distances between the closest point and the margin to 1 unit, $\rho||w|| = 1$, we will have the optimization problem:

minimize

$$\frac{1}{2}||w||^2 \tag{1}$$

subjected to

$$r^t(w^t x^t + w_0) \geq 1 \tag{2}$$

We can reformulate the problem using the Lagrange multipliers method to find the minima of a function subject to a constraint.

$$L_p = \frac{1}{2}||w||^2 - \sum_t \lambda^t [r^t(w^T x^t + w_0) - 1] \tag{3}$$

$$= \frac{1}{2}||w||^2 - \sum_t \lambda^t r^t (w^T x^t + w_0) + \sum_t \lambda^t$$

The partial derivatives are given by:

$$\frac{\partial L_p}{\partial w} = 0, \qquad \frac{\partial L_p}{\partial w_0} = 0.$$

$$\frac{\partial L_p}{\partial w} = w - \textstyle\sum_t \lambda^t r^t x^t = 0 \Leftrightarrow w = \textstyle\sum_t \lambda^t r^t x^t, \qquad \frac{\partial L_p}{\partial w_0} = \textstyle\sum_t \lambda^t r^t = 0.$$

We have a new formulation of the constrained optimization problem when we replace the partial derivatives in equation (3). The dual formulation is:

$$\text{maximize} -\frac{1}{2}\sum_t \sum_s \lambda^t \lambda^s r^t r^s (x^t)^T x^s + \sum_t \lambda^t$$

$$\text{subjected to } \sum_t \lambda^t r^t = 0 \text{ with } \lambda^t \geq 0, \forall t.$$

The Lagrange multipliers λ^t are identifying the support vectors. λ^t are positives for all t and most of them are equal to 0. The ones superior to 0, $\lambda^t > 0$, are associated with the data x^t that will have an impact on where the decision boundary goes.

The post-training data evaluation for y will be given by:

$$h(y) = \sum_t \lambda^t r^t (x^t)^T y + w_0$$

$$w_0 = E[r^t - w^T x^t] = \frac{1}{|M|} \sum_{\lambda^t \epsilon M} (r^t - \sum_{\lambda^s \epsilon M} \lambda^s r^s (x^t)^T x^s)$$

where M is the Lagrange multipliers set.

A little adjustment can be done with linearly separable data with noise. Slack variables ξ^t are introduced and associated with the training data x^t to permit flexibility of the decision boundary [13]. For x^t, $t = 1, .., i$:

1. $\xi^t = 0$ when there is no problem with x^t.
2. $0 < \xi^t < 1$ when x^t is on the right side of the decision boundary, but in the margin.
3. $\xi^t > 1$ when x^t is misclassified.

The adjustment is done on the optimization criterion.

$$r^t (w^T x^t + w_0) \geq 1 - \xi^t$$

When the data cannot be linearly separable in the original space, we use a kernel $K(\cdot, \cdot)$ to classify the data in a usually bigger dimensional space without having to work directly in the space. The resulting decision boundary in the original space will be non-linear.

The post-training data evaluation for a new data point y, using a kernel SVM is:

$$h(y) = \sum_t \lambda^t r^t K(x^t, y) + w_0$$

with

$$w_0 = \frac{1}{|M|} \sum_{\lambda^t \epsilon M} (r^t - \sum_{\lambda^s \epsilon M} \lambda^s r^s K(x^t, x^s))$$

where M is the Lagrange multipliers set.

3 Application

3.1 Uniform Manifold Approximation and Projection (UMAP)

We have reduced the dimensions of the data with UMAP. Manhattan and Euclidean have been both considered for the distance metric but we opted for Manhattan because it is better suited with high-dimension data [9].

3.2 *K*-means

k-means is an unsupervised machine learning method that will take as an input unlabeled data and return labeled data. We have used the elbow method to find the number of clusters.

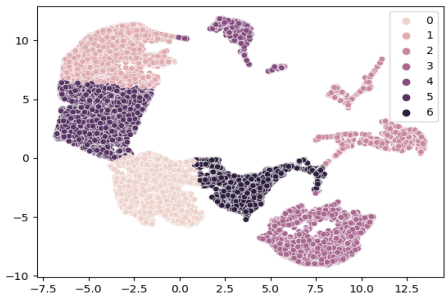

Figure 3: Clusters with *K*-means

With the labeled data, we illustrated the distribution of the data in respect to the features using box plots. This visualization tool offers general information about a group of data's symmetry, skew, variance and outliers which can be used to make comparisons between groups. Therefore, we were able to associate distinct gambling behaviours that differentiate one cluster from an other.

3.3 Support Vector Machine

We used the labels found with *k*-means and used kernel SVM algorithms on our gambling session data [12].

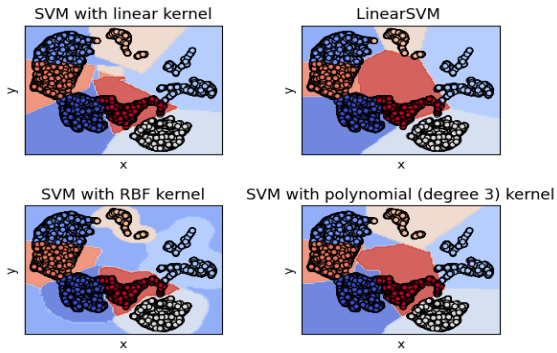

Figure 4: SVM algorithms with various kernels

We can use one of these kernel SVM to classify new, unobserved data into a group. Once a new data of a gambling session is classified into a group, we can deduce certain characteristics and gambling behaviours from the previous analysis.

4 Conclusion

This research has a solid base in data analysis and has a great list of related statistical theories. From understanding the problem, to data preprocessing, to data processing, we have learned how machine learning models are built and have selected parameters to achieve optimal results. Through this, we were able to identify distinct gambling behaviours for each cluster of gambling sessions.

References

[1] Jonathon P. et al. *Stochastic gradient descent*. URL: https://optimization.cbe.cornell.edu/index.php?title=Stochastic_gradient_descent. (accessed: 11.09.2023).

[2] LaBrie R.A. et al. "Assessing the playing field: A prospective longitudinal study of Internet sports gambling behavior". In: *Journal of Gambling Studies 23* (2007), pp. 347–362. DOI: 10899-007-9067-3.

[3] C.L Chang. "Fuzzy topological spaces". In: *Journal of Mathematical Analysis and Applications* 24.1 (1968), pp. 182–190. ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(68)90057-7. URL: https://www.sciencedirect.com/science/article/pii/0022247X68900577.

[4] Imad Dabbura. *K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks*. URL: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a. (accessed: 11.09.2023).

[5] Kemal Erdem. *t-SNE clearly explained*. URL: https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a. (accessed: 11.09.2023).

[6] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. (accessed: 07.09.2023).

[7] Arize IA and Leland McInnes. *How To Use UMAP and HDBScan To Surface Insights and Discover Issues*. URL: https://www.youtube.com/watch?v=g5DZuRDnOGY. (accessed: 11.09.2023).

[8] Scikit learn. *Clustering*. URL: https://scikit-learn.org/stable/modules/clustering.html. (accessed: 07.09.2023).

[9] Leland McInnes. *How UMAP Works*. URL: https://umap-learn.readthedocs.io/en/latest/how_umap_works.html. (accessed: 11.09.2023).

[10] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[11] Frédéric Meunier and Luis Montejano. "Different versions of the nerve theorem and colourful simplices". In: *Journal of Combinatorial Theory, Series A* 169 (2020), p. 105125. ISSN: 0097-3165. DOI: https://doi.org/10.1016/j.jcta.2019.105125. URL: https://www.sciencedirect.com/science/article/pii/S0097316519300998.

[12] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[13] Gilbert Saporta. *Probabilité, analyse des données et statistique*. 2e edition. Éditions Technip, France, 2006. ISBN: 2-7108-0814-5.

[14] Jonathan Schneider. *Geometry of Simplexes*. 2019. URL: http://homepages.math.uic.edu/~jschnei3/Writing/Simplexes. (accessed: 07.09.2023).

[15] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG].

[16] Bhasha Vachharajani and Dishant Pandya. "Dimension reduction techniques: Current status and perspectives". In: *Materials Today: Proceedings* 62 (2022). International Conference on Additive Manufacturing and Advanced Materials (AM2), pp. 7024–7027. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2021.12.549. URL: https://www.sciencedirect.com/science/article/pii/S2214785321083115.