Using Genetic Programming to Find Functional Mappings for UMAP Embeddings

Finn Schofield
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
schofifinn@ecs.vuw.ac.nz

Andrew Lensen
School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
andrew.lensen@ecs.vuw.ac.nz

***Abstract*—Manifold learning is a widely used technique for reducing the dimensionality of complex data to make it more understandable and more efficient to work with. However, current state-of-the-art manifold learning techniques — such as Uniform Manifold Approximation and Projection (UMAP) — have a critical limitation. They do not provide a functional *mapping* from the higher dimensional space to the lower-dimensional space, instead, they produce only the lower-dimensional embedding. This means they are "black-boxes" that cannot be used in domains where explainability is paramount. Recently, there has been work on using genetic programming to perform manifold learning with functional mappings (represented by tree/s), however, these methods are limited in their performance compared to UMAP. To address this, in this work we propose utilising UMAP to create functional mappings with genetic programming-based manifold learning. We compare two different approaches: one that uses the embedding produced by UMAP as the target for the functional mapping; and the other which directly optimises the UMAP cost function by using it as the fitness function. Experimental results reinforce the value of producing a functional mapping and show promising performance compared to UMAP. Additionally, we visualise two-dimensional embeddings produced by our technique compared to UMAP to further analyse the behaviour of each of the algorithms.**

***Index Terms*—Manifold learning, Genetic Programming, Dimensionality Reduction, Feature Construction, Feature Selection.**

I. Introduction

Dimensionality reduction is an important technique for simplifying data to make it more understandable and easier to analyse [1]. It is an area that has received a great deal of interest, especially in recent years. One area that has received particular attention is nonlinear dimensionality reduction, also known as manifold learning [2]. Manifold learning algorithms seek to find an embedded manifold that the data lies on in the high-dimensional space, that can then be represented in a lower-dimensional space. While linear dimensionality reduction techniques rely on finding an embedding in a linearly transformed subspace [3], manifold learning techniques do not have this constraint.

Manifold learning techniques can be divided into two categories: *mapping* and *non-mapping* techniques. Mapping techniques learn a function to map data from the high dimensional space to the low-dimensional space, while non-mapping techniques simply provide the low-dimensional embedding. Currently, non-mapping techniques, such as Uniform Manifold Approximation and Projection (UMAP) [4], are the state-of-art in terms of embedding quality. However, mapping techniques produce functions that can be readily applied (reused) on new data, as well as offering a more transparent way of understanding the manifold and how it relates to the original data.

Genetic Programming (GP) is an evolutionary learning technique in which computer programs are evolved over generations [5]. GP's functional structure lends itself well to be used as a mapping technique for manifold learning, and there has been some work combining the two [6]. However, these techniques are still outperformed in embedding quality by the non-mapping state-of-the-art techniques.

To address the shortcomings of mapping-based manifold learning techniques, this paper will propose two new GP techniques which incorporate UMAP in their fitness evaluation: one which will seek to directly reproduce the embedding found by UMAP, and one which will indirectly replicate UMAP by directly optimising the UMAP objective function. These two approaches are indicative of a bigger question in GP: is it better to try and match a known good output (minimising error), or to directly optimise the fitness function that achieved that output? By utilising the power of a proven non-mapping method to find a functional mapping, we hope to combine the performance of UMAP with the functional representation of GP. Specifically, we will:

- Propose two new fitness functions for GP based on UMAP. The first will directly evaluate the individual against a precomputed UMAP embedding, and the second will seek to maximise the UMAP cost directly.
- Evaluate our proposed methods against pure UMAP through classification performance, as well as compare their performance to each other.
- Analyse selected two-dimensional visualisations produced by GP individuals, and compare them to visualisations produced directly by UMAP.
- Analyse a selected GP individual to understand the functional mappings produced by our techniques.

II. Background

A. Dimensionality Reduction

The dimensionality of a dataset is the number of features (or attributes) used to represent each instance. However, as this dimensionality increases, the data becomes more sparse and less efficient to perform computations over — an instance of the curse of dimensionality [7]. Hence, there is a need for techniques that can reduce the dimensionality of the data, while still maintaining its intrinsic structure. These methods, known as dimensionality reduction techniques, are used to find lower-dimensional embeddings of data as a preprocessing step for more efficient and effective use in other machine learning algorithms [1]. Another common use for dimensionality reduction techniques is using them to reduce data to 2 or 3 dimensions for visualisation. Simple techniques use linear transformations of the data from the high-dimensional to the low-dimensional space, such as Principal Component Analysis (PCA) [8] and Linear Discriminant Analysis (LDA) [9].

1) Feature Manipulation: A major paradigm of dimensionality reduction is feature manipulation (FM) [10]. Within FM, feature selection (FS) chooses w of the features from the original v-dimensional space to form a subspace; feature construction (FC), in contrast, *constructs* w new features by combining features from the v-dimensional space. There are three classes of FM techniques: wrapper, filter, and embedded. Wrapper methods directly use a machine learning algorithm (e.g. naïve Bayes) to evaluate the quality of a feature set. Filter methods evaluate the quality of a feature set through predefined metrics. In the third class, embedded methods, FM is an intrinsic part of the learning algorithm itself (e.g. decision tree learning) [11].

There is an inherent trade-off between wrapper and filter methods. Generally, filter methods are faster as they do not require the training of a model to evaluate a feature sets quality as wrapper methods do. However, wrapper methods are generally better at producing the best feature set for a specific learning algorithm.

2) Manifold Learning: Reducing the dimensionality of data through linear transformations is often insufficient for complex data. Manifold learning, or non-linear dimensionality reduction, assumes that the data is sampled from a lower-dimensional manifold in the original space. These techniques seek to produce an embedding of the data onto this manifold. Manifold learning algorithms can be divided broadly into two classes, mapping and non-mapping methods.

The benefits of mapping techniques are their interpretability and reusability. The functional mapping itself can be interpreted to better understand which features and feature interactions are most important to reducing the dimensionality of the data, providing more insight into the nature of the data itself. Additionally, once a functional mapping has been found for a dataset, additional unseen data from the same domain can have the functional mapping applied to it without having to re-run the (often expensive) manifold learning algorithm.

Manifold learning techniques can be seen as a type of unsupervised filter-based feature construction. Instead of optimising the low-dimensional space for a specific machine learning algorithm, manifold learning instead tries to preserve the underlying structure of the data.

There are various manifold learning algorithms available, including the canonical t-Distributed Stochastic Neighbourhood Embedding (t-SNE) [12], and a non-linear variant of PCA [13]. Uniform Manifold Approximation and Projection (UMAP) is currently considered the state-of-the-art manifold learning technique [4]. UMAP works by first constructing a graph-based representation of the data in the high dimension space, where each edge of the graph corresponds to a probability that the instances are in the same n-neighbourhood. A second graph-based representation is then created in the low-dimensional space that is then optimised to be as similar to the high-dimensional graph as possible. This produces the low-dimensional embedding of the data.

B. Genetic Programming

GP is an evolutionary computation technique that represents solutions to a problem as evolvable computer programs [5]. GP begins with a population of randomly initialised individuals. The most common GP representation is a tree-like structure, where the leaf nodes represent problem-specific values and ephemeral random constants, while the internal nodes represent functions that take their children as inputs, producing an output that propagates up the tree.

The tree-like structure of GP lends itself naturally to representing interpretable functional mappings for manifold learning. A GP tree that reduces an instance to a single value can be thought of as a dimensionality reduction, where the original data is reduced to a single dimension. Existing work has used a multitree representation, where each tree represents a functional mapping of the data in the high-dimensional space to a single dimension in the low-dimensional space [14]. The multitree representation provides a straightforward functional mapping, where the importance of the original features to the low dimension features have the potential to be easily understood.

While GP has been used for manifold learning, existing techniques have relied on ad hoc measures of manifold quality. We believe that there is clear potential for improvement by instead using the properties of UMAP to measure the fitness in GP-based manifold learning. Through this, we expect that the strengths of the two approaches can be combined.

C. Related Work

There has been a wide range of research into GP for FC [14], and recent work looking more specifically into GP for manifold learning [6], [15]. This research has clearly demonstrated the power of GP for dimensionality reduction. The strength in GP techniques has been demonstrated to be its ability to produce interpretable and reusable solutions. There has been relevant work on using GP specifically for interpretable visualisation models, which is essentially two-dimensional manifold learning [16]. The use of GP for various techniques tangential to manifold learning has also been investigated, such as for auto-encoding [17] and feature learning [18].

Some work has been done into adapting UMAP to produce functional mappings, such as Parametric UMAP [19]. Parametric UMAP uses a deep neural network to learn a functional mapping between the graphical representation of the high-dimensional data produced by UMAP and a low-dimensional embedding. Parametric UMAP has been shown to perform similarly to UMAP while proving a functional mapping. However, due to the use of a deep neural network, this mapping is still very complex with many parameters, making it difficult to interpret.

III. Proposed Approaches

We propose two GP approaches for recreating the UMAP embedding, each using a different fitness function.

A. GP Representation

Both of our proposed methods use the same multitree representation, where each tree corresponds to a functional mapping of the high-dimensional data to a single dimension in the low-dimensional space. This multi-tree representation is well-suited to the task of dimensionality reduction. The function set used contains the standard binary arithmetic operators: $+$, $-$, $\times$, $\div$, where $\div$ is protected such that any time a division by zero is attempted, 1 is returned. Additionally, a $5+$ operator is used that takes 5 inputs and returns the sum of all. This is used to encourage trees with a higher branching factor, as this tends to produce trees with fewer nodes compared to very deep trees. The logical operators max, min and if are also used. max and min simply return the maximum or minimum of their inputs, respectively, while if takes three inputs: it returns the second if the first is less than zero, otherwise, it returns the third input. Finally, two nonlinear unary operators are used: the sigmoid function and the Rectified Linear Unit (ReLU). By including nonlinear operators, we allow our individuals to be able to capture the nonlinear structure of the manifolds.

The terminal set used comprises all the features of the original data, in addition to ephemeral random constants — randomly generated values that remain constant through evolution once initialised.

Our multi-tree approach requires unique crossover and mutation operators. For crossover, the standard crossover is applied to each pair of trees between the two selected individuals. This all-index crossover approach allows certain trees to specialise within an individual. Mutation is performed by simply selecting a random tree of the individual to be mutated and performing standard mutation to it.

An example of a multi-tree individual is provided in Fig. 1. This individual represents a functional mapping of 10-dimensional data on a 5-dimensional manifold. Of these trees, (a) and (c) represent simple feature selection; while the other trees are more sophisticated functions.

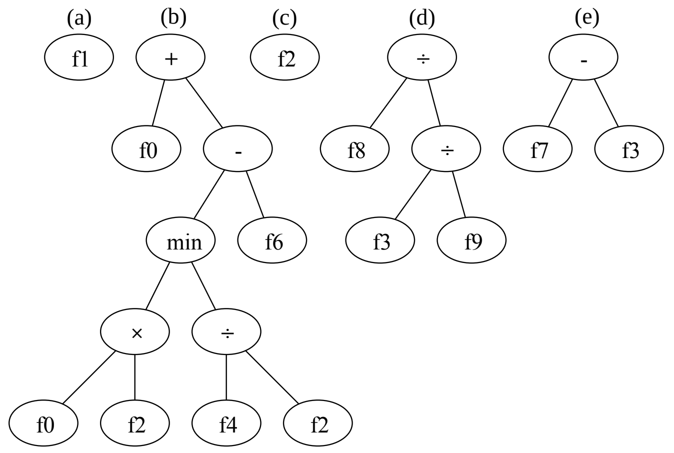


Fig. 1: An example of a multi-tree GP individual. This individual represents a functional mapping of 10-dimensional data to a 5-dimensional embedding, where each tree represents a mapping to a single constructed feature in the embedding.

B. UMAP Cost Fitness

Our first proposed GP approach uses the UMAP cost function as the fitness function in GP. This method seeks to produce individuals which produce embeddings with the lowest UMAP cost. Through this technique, we seek to create embeddings that would be judged as high-quality by the UMAP algorithm.

The UMAP cost of an embedding is calculated in several steps [4]. First, $\mathbf{v}$, the high-dimensional graph representation, is calculated:

$$v_{ij} = \exp[-(r_{ij} - \rho_i/\sigma_i)] \tag{1}$$

where v_{ij} is the probability that an edge exists between instances i and j. r_{ij} are the distances between the instances in the high-dimensional space (using Euclidean distance). p_i is the distance to the nearest neighbour of i. σ_i, which controls the degree of interaction between neighbours, is found through a binary search such that $\Sigma_j v_{ij} = log_2 k$. $\mathbf{v}$ is then symmetrised in Eq. (2), to ensure that the neighbours are equally related. As $\mathbf{v}$ only concerns the data in the high-dimensional space (the original data), this only needs to be calculated once at the start of the evolutionary process.

$$\mathbf{v}_{symm} = \mathbf{v} + \mathbf{v}^T - \mathbf{v} \circ \mathbf{v}^T \tag{2}$$

Given a low-dimensional embedding (i.e. the output of a GP individual), $\mathbf{w}$ is calculated as in Eq. (3). $\mathbf{w}$ is similar to $\mathbf{v}$ — it represents the low-dimensional embedding as a graph. w_{ij} represents the probability that an edge exists between instances i and j in the low-dimensional space. d is the distance between i and j in the low-dimensional space, and a and b are hyper-parameters. The UMAP recommended defaults of $a = 1.929$ and $b = 0.7915$ are used.

$$w_{ij} = 1/(1 + ad_{ij}^{2b}) \tag{3}$$

Finally, the dissimilarity between the high- and low-dimensional spaces is calculated based on $\mathbf{v}$ and $\mathbf{w}$. This is done as in Eq. (4), which provides us with the cost (fitness) of the embedding.

The UMAP cost (C_{UMAP}) is always nonnegative, with 0 representing an embedding with perfect retention of structure. Thus, this cost function should be minimised.

$$C_{UMAP} = \sum_{i \neq j} v_{ij} \log(\frac{v_{ij}}{wij}) + (1 - v_{ij}) \log(\frac{1 - v_{ij}}{1 - w_{ij}}) \quad (4)$$

C. NRMSE Fitness

Our Normalised Root Mean Square Error (NRMSE) fitness technique uses a reference UMAP embedding as the target output for GP. This technique offers an obvious way to produce a manifold with a functional mapping, as we work backwards by trying to find a functional mapping for an already existing (known good) embedding. As this is a simple error calculation, it is also computationally more efficient than optimising the UMAP cost as in our first method. Additionally, UMAP only needs to be run once at the start of the evolution to learn the embedding we try to reproduce.

The NRMSE fitness works by first using the GP individual to produce the low-dimensional embedding from the original data. The error between this embedding and the low-dimensional UMAP embedding can then be calculated, first by calculating the root mean square error (RMSE) as in Eq. (5).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}} \quad (5)$$

where y_i is the output of the i^{th} GP tree and $\hat{y}_i$ is the corresponding dimension of the UMAP embedding. n is the number of trees. Finally, the normalised RMSE is calculated with Eq. (6). We use the normalised form to allow for easier comparison across different datasets with varied domains.

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \quad (6)$$

where y_{max} and y_{min} are the maximum and minimum outputs across the GP individual. NRMSE fitness is always nonnegative, where a value of 0 indicates that the embedding produced by the GP individual is identical to the UMAP embedding.

IV. Experiment Design

To evaluate the quality of our two proposed methods, we focus primarily on the classification accuracy that can be attained when using the learned embeddings. If the classification accuracy achieved using the embeddings produced by our techniques remains competitive with that of the embeddings produced by UMAP, we can infer that our embeddings have retained a similar level of structure from the original dataset. By evaluating our methods through classification a single time at the end of the evolution, we avoid any potential bias that would result from using a measure related to that used by one of the fitness functions.

TABLE I: GP Parameters

Parameter	Setting
Generations	1000
Population Size	1024
Mutation	20%
Crossover	80%
Elitism	top 10
Selection	Tournament
Min. Tree Depth	2
Max. Tree Depth	8
Tournament Size	7
Pop. Initialisation	Half-and-half

TABLE II: Classification Datasets Used

Dataset	Instances	Features	Classes
Breast Cancer	683	9	2
COIL20	1440	1024	20
Dermatology	258	34	6
Ionosphere	351	34	2
MFEAT	2000	649	10
MNIST	2000	748	2
Movement Libras	360	90	15
Segmentation	2310	19	7
Wine	178	13	3

The classification algorithm used for evaluating classification performance is the scikit-learn Random Forest implementation with 100 trees. Random Forest is a reasonably cheap classification algorithm that performs well across a range of datasets and is, therefore, a good candidate [20]. 10-fold cross-validation is used to evaluate the embedding produced by the final evaluation. For each dataset, 30 runs are performed for each method to account for stochasticity in the evolutionary process.

For each method on each dataset, we perform our experiments to a reduction to one, two, three, five, and 10 dimensions. One, two, and three dimensions were selected as they represent the most extreme reductions in dimensionality. The two-dimensional case is also easily analysed through visualisation. Additionally, five and 10 dimension reduction is also selected to analyse how the methods compare on easier manifold learning problems.

Standard GP parameters are used as presented in Table I. The default UMAP parameters are used to generate the UMAP embedding for NRMSE fitness.

A. Datasets

The selected datasets for the experiments are presented in Table II. These datasets are mostly real-world datasets from the UCI Repository. The datasets have been selected as they represent a range of values in both instances, features and classes. This allows us to evaluate the robustness of our approach to a range of real-world data of varying characteristics. Understanding how our methods perform on high-dimensional data is especially important, as these datasets are the ones where dimensionality reduction is most needed.

TABLE III: Classification Accuracy Results

Method	Breast.	COIL.	Derm.	Iono.	MFEAT.	MNIST.	Move.	Seg.	Wine
UCOST1	0.920−	**0.244**−	0.675−	0.846	0.271−	0.617−	**0.339**−	**0.615**−	0.810−
NRMSE1	**0.925**−	0.213−	**0.717**−	0.855+	**0.321**−	**0.885**−	0.323−	0.471−	**0.885**−
UMAP1	0.958	0.844	0.950	0.843	0.969	0.975	0.652	0.814	0.945
UCOST2	**0.959**−	**0.575**−	0.870−	0.872+	0.603−	0.815−	**0.513**−	**0.757**−	0.922−
NRMSE2	0.912−	0.453−	0.857−	0.875+	**0.639**−	**0.905**−	0.454−	0.718−	0.925−
UMAP2	0.970	0.897	0.961	0.857	0.980	0.986	0.721	0.844	0.961
UCOST3	**0.967**−	**0.693**−	**0.923**−	0.886+	0.737−	0.863−	**0.619**−	**0.817**−	**0.946**−
NRMSE3	0.945−	0.605−	0.860−	0.886+	0.746−	**0.918**−	0.535−	0.797−	0.931−
UMAP3	0.972	0.910	0.964	0.869	0.981	0.987	0.749	0.852	0.962
UCOST5	**0.968**−	**0.797**−	**0.948**−	0.897+	**0.844**−	0.902−	**0.706**−	**0.871**+	**0.962**
NRMSE5	0.958−	0.718−	0.900−	0.893+	0.808−	**0.917**−	0.611−	0.840−	0.912−
UMAP5	0.973	0.922	0.966	0.865	0.982	0.987	0.745	0.857	0.962
UCOST10	0.969−	**0.883**−	**0.954**−	0.906+	**0.908**−	**0.921**−	**0.757**	**0.902**+	**0.959**
NRMSE10	0.967−	0.820−	0.918−	0.903+	0.831−	0.896−	0.688−	0.887+	0.904−
UMAP10	0.973	0.929	0.966	0.868	0.981	0.986	0.748	0.855	0.961

B. *Evaluation Measures*

Our techniques are measured and compared to UMAP using classification accuracy. Classification accuracy is represented as a real number in the range $[0, 1]$, such that 1 represents all instances in a dataset being correctly classified and 0 represents no instances in a dataset being correctly classified. As such, classification accuracy is a measure we seek to maximise.

V. Results

The mean classification accuracies from the 30 runs performed of each experiment are presented in Table III. We use NRMSE to represent the NRMSE fitness approach, UCOST to represent the UMAP cost fitness approach, and UMAP to represent the standard UMAP. For each method, we also indicate the number of dimensions the data is being reduced to: UMAP1 is UMAP reducing to a single dimension, UCOST2 is UCOST reducing to two dimensions, etc.

A Mann-Whitney significance test was performed with a p-value of 0.05. The tests were performed across all experiments between each method and the UMAP baseline, as well as between each other. A $+$ or $-$ is presented next to a UCOST or NRSME result if it was found to be statistically better or worse than UMAP, respectively. Neither sign appearing indicates no significant difference was found. If either UCOST or NRMSE performed significantly better than the other on the dataset, then the mean is presented in bold. If neither are bold, no significant difference in performance was found.

A. *UMAP Cost vs. UMAP*

From the results, generally, the UMAP Cost fitness method is outperformed in classification accuracy by UMAP. However, there are several datasets where UMAP cost outperforms UMAP in some dimensions, as discussed below.

The strongest result is in UMAP Cost's performance on the Ionosphere dataset. UMAP Cost performed significantly better than UMAP in all dimensions, except in a single dimension where no significant difference was found. Additionally, UMAP Cost outperformed UMAP on the Segmentation dataset in 5 and 10 dimensions. On the Wine dataset in 5 and 10 dimensions and Movement Libras in 10 dimensions, no significant difference in classification performance was found.

Even though UMAP cost, in general, has poorer classification accuracy than UMAP, the fact that on some datasets UMAP Cost can outperform UMAP (and on others, come close) while producing a functional mapping is promising.

B. *NRMSE vs. UMAP*

As with UMAP Cost, NRMSE is generally outperformed by UMAP. In all one-, two-, three- and five-dimensional experiments, NRMSE performs significantly worse except for the Ionosphere dataset, where NRMSE outperforms UMAP in all dimensions. In 10 dimensions, NRMSE also outperforms UMAP on the Segmentation dataset.

As NRMSE is trying to recreate UMAP directly, it is not surprising that it is generally outperformed. Therefore, the fact that on certain datasets in certain dimensions NRMSE does outperform UMAP is interesting to observe, as it suggests that UMAP is struggling to optimise effectively in some cases. This result is further analysed through visualisations in Section VI.

C. *UMAP Cost vs. NRMSE*

Comparing our two GP techniques, we see that when reducing to a single dimension, NRMSE outperforms UMAP Cost on five of the nine datasets. On Movement Libras, Segmentation and COIL20, UMAP Cost outperforms NRMSE, while there was no significant difference found between the two techniques on the Ionosphere dataset.

However, as the dimensions of the embedding increase, NRMSE's slight performance advantage disappears. When reducing to two dimensions, NRMSE only outperforms UMAP Cost on the MFEAT and MNIST datasets.

When reducing to three and five dimensions, the same trend continues. UMAP Cost is only outperformed on the MNIST and has better classification accuracy on all other datasets except Ionosphere and MFEAT.

Finally, when reducing the dimensionality to ten dimensions, UMAP Cost outperforms NRMSE on all datasets except Breast Cancer, where there is no longer a significant difference between the performance of the two methods across the dataset.

Comparing the performance of the two methods, a general trend of UMAP Cost's performance increasing relative to NRMSE as the dimensions increase is clear. This, in combination with NRMSE being outperformed more often by UMAP, indicates that UMAP Cost is, in general, the superior GP method. This may be as NRMSE is constrained by trying to directly recreate a UMAP embedding that is not guaranteed to be easily represented in a depth-limited tree. UMAP Cost, on the other hand, is indirectly recreating UMAP by optimising the preservation of the structure in the low-dimensional space. In this way, GP has more freedom to discover good embeddings that are viable to represent with a tree-based mapping.

VI. Visualisation Analysis

As visualisation — especially in two dimensions — is a common use of dimensionality reduction, it is useful to compare the visualisations produced by our methods and compare them to each other and those produced by UMAP. The two-dimensional embeddings with median classification performance produced by each method for the Dermatology, MFEAT and Ionosphere datasets are presented in Fig. 2. Ionosphere is selected for analysis as it is the only dataset our techniques were able to significantly outperform UMAP on over a range of dimensions. MFEAT is selected due to its high dimensionality with 649 features, while Dermatology is selected as a lower-dimensional dataset, having 34 features, while still being high enough such that visualisation is valuable.

Looking at the Dermatology visualisations in Fig. 2a, Fig. 2b, and Fig. 2c, we see that the GP methods do not cluster the data as tightly as the UMAP embeddings. However, both GP methods are sufficient in capturing the same overall global structure as UMAP, particularly in the case of NRMSE. In some ways, this is a limitation of UMAP: it tends to sacrifice local structure (the relationship within a neighbourhood) to separate clusters as much as possible. By using a mapping-based approach, GP is less likely to lose neighbourhood structure.

The MFEAT visualisations are shown Fig. 2d, Fig. 2e, and Fig. 2f. Both the GP methods struggle to separate the classes, resulting in a single large cluster, with classes spread around the cluster. As with Dermatology, the same general global structure is captured, only with more overlap and less separation than UMAP.

The Ionosphere visualisations in Fig. 2g, Fig. 2h, and Fig. 2i show us an example of visualisations where our GP methods outperformed UMAP. We can see that UMAP maintains the denser, more well-separated clusters compared to GP we have observed in the previous examples. However, these clusters do not correspond to the class labels to anywhere near the same degree as in Dermatology and MFEAT. The UMAP visualisation divides the data into a group of distinct clusters, two comprised of a mix of yellow and purple labels, one with yellow labels and a single purple label, and the rest being comprised solely of yellow labels. Both GP visualisations do not contain clear clusters, however, they do separate the purple labels from the yellow in a more consistent way. UCOST presents a horizontal division at roughly 0.45 on the y-axis, with all purple labels being below this line. While there are still yellow labels grouped with purple labels, the majority are above 0.45. NRMSE presents the data in a slightly different way. All purple labels are grouped at 0.0 on the x-axis while varying along the y-axis. Again, some yellow labels are mixed in with the purple labels, however, the majority are clearly separated.

Our analysis of the visualisations indicates that generally, UMAP is superior at finding a manifold that captures the structure of the high-dimensional data in two-dimensional space. However, in the case of Ionosphere, UMAP seems to also find structure where it doesn't exist, leading to inferior classification results compared to the GP methods.

VII. Evolved Individual Analysis

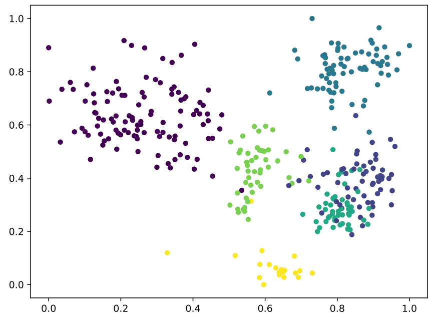
As the utility of mapping-based manifold learning techniques is the ability to interpret the mapping from the high-dimensional space to the lower one, it is useful to look at what an evolved individual looks like. An example of an evolved individual is presented in Fig. 3. This is the smallest individual produced from the 30 experiments performed on the Ionosphere dataset using the NRSME GP method with a single dimension. This individual is comprised of 127 nodes and uses 27 of the original 34 features. It achieved a classification performance of 0.863 — the mean of 0.855.

Analysing the tree, we can see that the non-linear ReLU function is used four times. Each of these times, the input comes directly from a feature node — $f29$, $f32$, and $f12$, which occurs twice. The use of ReLU in the individual indicates some non-linear relationship between the features in the original space and on the manifold, especially in $f12$.

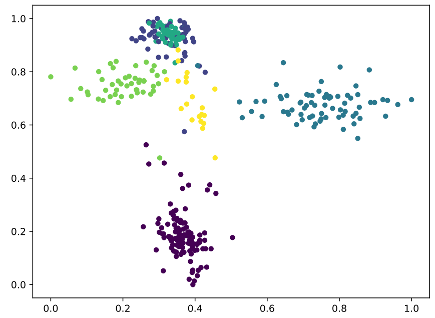
Even though the tree is quite large, and therefore represents a fairly complex mathematical function, it is promising to produce an interpretable functional mapping that can outperform UMAP. For example, from this individual, we can determine the features of the original data that are or are not important for the reduction in dimensionality by whether or not they have been included. Future efforts to incorporate parsimony pressure or other such bloat reduction techniques would likely create even more interpretable mappings.

VIII. Conclusion

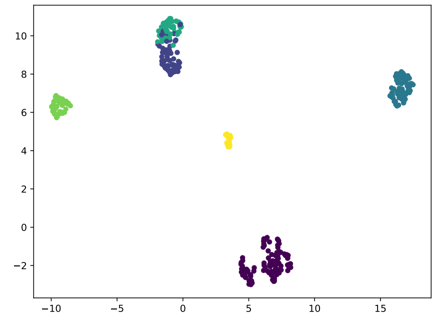
This work presented two new GP approaches to manifold learning that incorporate components of the current state-of-the-art UMAP method. By using GP, we can perform mapping-based manifold learning, while utilising the performance of



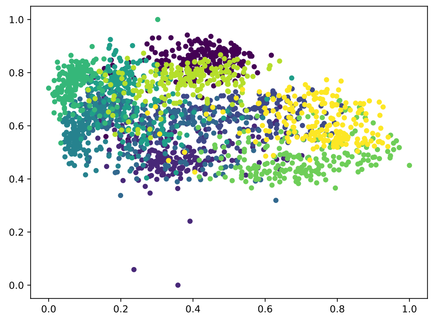
(a) Dermatology – UCOST



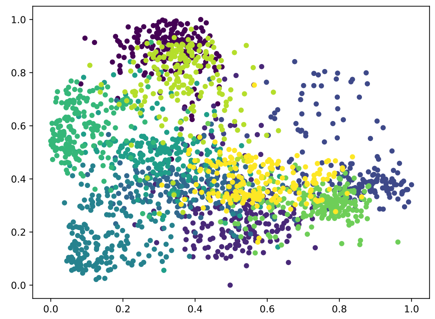
(b) Dermatology – NRMSE



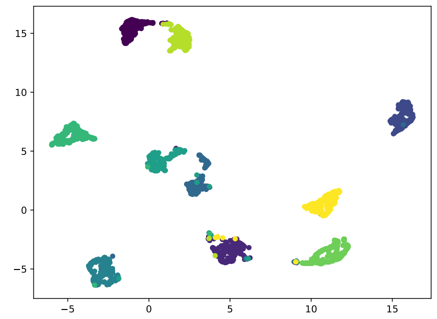
(c) Dermatology – UMAP



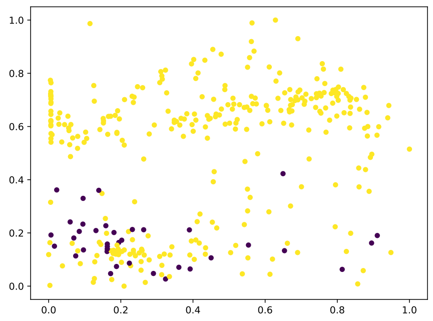
(d) MFEAT – UCOST



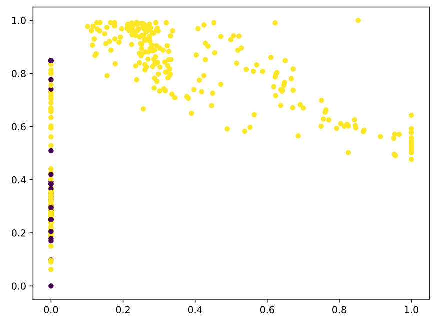
(e) MFEAT – NRMSE



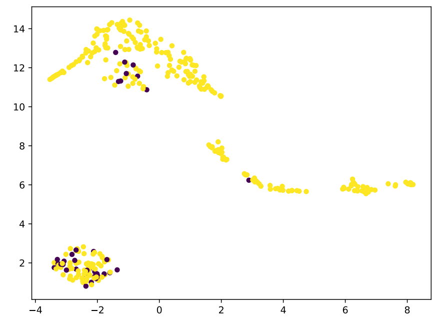
(f) MFEAT – UMAP



(g) Ionosphere – UCOST



(h) Ionosphere – NRMSE



(i) Ionosphere – UMAP

Fig. 2: Visualisations produced by the GP methods and UMAP on the Dermatology, MFEAT and Ionosphere datasets. The median result of each was chosen for visualisation.

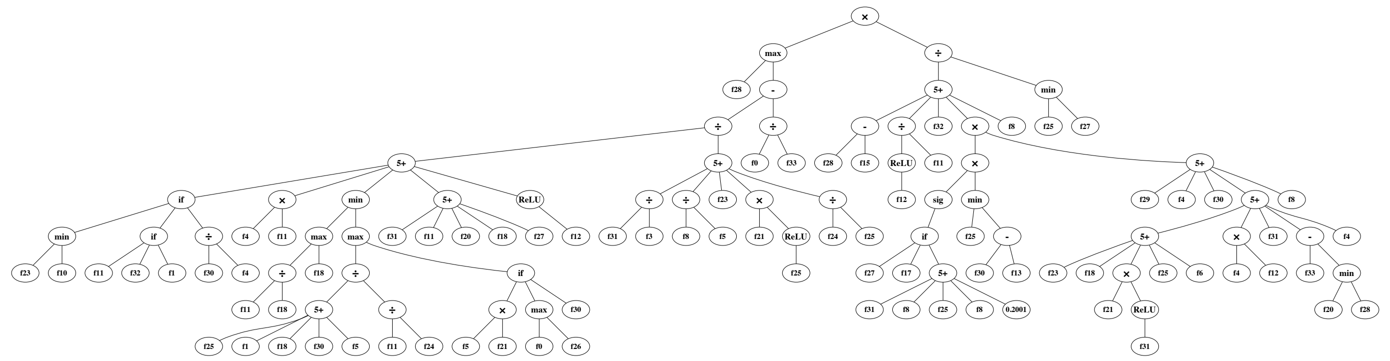


Fig. 3: A single dimension individual for the Ionosphere dataset. This individual presents an example of a single constructed feature that outperforms UMAP.

UMAP which is a non-mapping method. The first approach, NRMSE, sought to directly reproduce a UMAP embedding by evaluating the error of the GP-produced embedding against that of one produced by UMAP. The second approach, UMAP Cost, indirectly found an embedding by optimising the UMAP cost function. We evaluated the performance of one, two, three, five, and 10 dimension embeddings produced by these methods across nine datasets of varying characteristics. We compared these embeddings to each other, as well as to embeddings produced by UMAP as a baseline.

We have found that generally, the classification performance of embeddings produced by our methods are poorer than those produced by UMAP. This was not surprising, as some performance loss should be expected when trying to mimic a high-performance, non-mapping technique such as UMAP. Considering that the GP methods provide a functional mapping, they are a promising candidate for further work in using UMAP itself to find embeddings that are more transparent.

In comparing our two GP methods, we found that overall, UMAP Cost tended to perform better in terms of classification accuracy, especially on higher dimension embeddings. UMAP Cost also outperformed UMAP on more experiments than NRMSE. This led us to conclude that UMAP was generally the more effective of our two proposed techniques. This is likely because UMAP Cost is less constrained in its optimisation, optimising the embedding directly rather than trying to indirectly recreate a specific UMAP embedding.

The two-dimensional visualisations produced by our methods were compared to each other and those produced by UMAP. It was found that while UMAP was superior in producing dense, well-separated clusters, the GP methods were able to capture the same general global structure. Additionally, analysis of the visualisations on the Ionosphere dataset provided more context on the GP methods' superior performance. A trade-off between the GP techniques and UMAP was observed, with GP having less defined visualisations while providing a functional mapping to better understand the learned manifold. Further analysis of a relatively small evolved individual on the Ionosphere dataset reinforced the potential for GP to perform interpretable manifold learning by properties of the UMAP algorithm.

A. Further Work

This work represents an initial exploration into using UMAP to complement GP for manifold learning. As such, there is plenty of further work than can be explored. While the strengths of the proposed approaches lie in the functional mappings associated with their low-dimensional embedding, no bloat control techniques beyond depth limiting have yet been used to help keep the functional mappings smaller and thus more interpretable. Further work could look into incorporating techniques such as parsimony pressure to produce more interpretable trees.

References

[1] L. Van Der Maaten, E. Postma, and J. Van den Herik, "Dimensionality reduction: a comparative," *J Mach Learn Res*, vol. 10, no. 66-71, p. 13, 2009.

[2] L. Cayton, "Algorithms for manifold learning," Tech. Rep. CS2008-0923, 2005.

[3] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learn. Res.*, vol. 16, pp. 2859–2900, 2015.

[4] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: uniform manifold approximation and projection," *J. Open Source Softw.*, vol. 3, no. 29, p. 861, 2018. [Online]. Available: https://doi.org/10.21105/joss.00861

[5] J. R. Koza, *Genetic programming - on the programming of computers by means of natural selection*, ser. Complex adaptive systems. MIT Press, 1993.

[6] A. Lensen, B. Xue, and M. Zhang, "Can genetic programming do manifold learning too?" in *Genetic Programming - 22nd European Conference, EuroGP 2019, Held as Part of EvoStar 2019, Leipzig, Germany, April 24-26, 2019, Proceedings*, 2019, pp. 114–130.

[7] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering," in *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases*, 1999, pp. 506–517.

[8] I. Jolliffe and J. Cadima, "Principal component analysis: A review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, p. 20150202, 04 2016.

[9] A. J. Izenman, *Linear Discriminant Analysis*. New York, NY: Springer New York, 2008, pp. 237–280.

[10] P. Sondhi, "Feature construction methods: a survey," Tech. Rep., 2009.

[11] L. Talavera, "An evaluation of filter and wrapper methods for feature selection in categorical clustering," in *Advances in Intelligent Data Analysis VI, 6th International Symposium on Intelligent Data Analysis, IDA 2005, Madrid, Spain, September 8-10, 2005, Proceedings*, ser. Lecture Notes in Computer Science, A. F. Famili, J. N. Kok, J. M. P. Sánchez, A. Siebes, and A. J. Feelders, Eds., vol. 3646. Springer, 2005, pp. 440–451.

[12] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[13] A. R. Webb, "An approach to non-linear principal components analysis using radially symmetric kernel functions," *Stat. Comput.*, vol. 6, no. 2, pp. 159–168, 1996.

[14] B. Tran, B. Xue, and M. Zhang, "Genetic programming for multiple-feature construction on high-dimensional classification," *Pattern Recognition*, vol. 93, pp. 404–417, 2019.

[15] P. Orzechowski, F. Magiera, and J. H. Moore, "Benchmarking Manifold Learning Methods on a Large Collection of Datasets," in *Proceedings of the European Conference on Genetic Programming (EuroGP)*, ser. Lecture Notes in Computer Science, vol. 12101, 2020, pp. 135–150.

[16] A. Lensen, B. Xue, and M. Zhang, "Genetic Programming for Evolving a Front of Interpretable Models for Data Visualization," *IEEE Trans. Cybern.*, pp. 1–15, 2020.

[17] J. McDermott, "Why Is Auto-Encoding Difficult for Genetic Programming?" in *Proceedings of the European Conference on Genetic Programming (EuroGP)*, ser. Lecture Notes in Computer Science, vol. 11451. Springer, 2019, pp. 131–145.

[18] S. Ruberto, V. Terragni, and J. H. Moore, "Image feature learning with genetic programming," in *Proceedings of the 16th International Conference on Parallel Problem Solving from Nature (PPSN XVI)*, ser. Lecture Notes in Computer Science, vol. 12270. Springer, 2020, pp. 63–78.

[19] T. Sainburg, L. McInnes, and T. Q. Gentner, "Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning," *CoRR*, vol. abs/2009.12981, 2020. [Online]. Available: https://arxiv.org/abs/2009.12981

[20] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and qsar modeling," *Journal of chemical information and computer sciences*, vol. 43, no. 6, pp. 1947–1958, 2003.