Searching for Optimal Projections of Multidimensional Datasets with Applications to Classification Algorithms

Dominic Dent

9567718

In Collaboration With Karan Muhki

Supervised by Professor Stephen Watts

University of Manchester

School of Physics and Astronomy

MPhys Project

May 2019

Abstract

Four optimisation algorithms are implemented on a $n \times p$ matrix manifold known as the Stiefel manifold. The matrices constrained on the Stiefel manifold are optimised such that they minimise a single-valued, continuous cost function defined over the domain. Optimal matrices are used to project from an n-dimensional space to a p-dimensional space. These optimisation algorithms are applied to high dimensional, classification datasets such that a linear support vector machine (SVM) hinge loss function is minimised when the data is projected into 2-dimensions. This technique has applications to dimensionality reduction.

Contents

1 Introduction

Data is becoming an ever-increasing key resource in the digital world. Through the use of algorithms and smart analysis, great value can be found in data. One prominent example of the power of data science is the increasing accuracy of automated cancer prognosis and prediction[1]. With the exponential growth of computational power and the creative innovation that is so often seen in the technology sector, new, more extravagant ways of extracting value from data are being utilised that were previously above the limitations of computation.

During semester 1[2], the Grand Tour algorithm[3] was investigated as a method for finding "good" 2D projections of high dimensional data. At each iteration of the Grand Tour, a support vector machine (SVM), neural network and decision tree algorithm was applied and the accuracy was evaluated. However, the Grand Tour was deemed a highly inefficient way to find optimal 2D projections and hence a new optimisation algorithm for finding 2D projects was implemented.

In this project, different optimisation algorithms were implemented on the Stiefel manifold. The motivation behind implementing these optimisation algorithms was to find the 2-dimensional projection of a general n-dimensional feature space, that produces the highest classification accuracy when a Support Vector Machine (SVM) algorithm is applied to the data in the subspace. Optimisation is the process of finding the maximum or minimum value of a particular single-valued function, called the cost function. The optimisation algorithms have been implemented in such a way that they are generalised to project into any $k < n$ subspace and also optimise for any single-valued continuous cost function; the cost function is not necessarily required to be one that acts on a classified dataset.

The Stiefel manifold, $V_k(\mathbb{R}^n)$ or $V_{n,k}$, is a matrix manifold embedded in $\mathbb{R}^{n\times k}$ space where $k \leq n$. Formally, the manifold represents all orthonormal k-frames of a n-dimensional space, where a k-frame is an ordered set of k linearly independent vectors[4]. By finding points on the Stiefel manifold, the corresponding $\mathbb{R}^{n\times k}$ matrix $\mathbf{Y} \in V_{n,k}$ can be used to project the n-dimensional data $\mathbf{X} \in \mathbb{R}^n$ into the subspace by the operation $\mathbf{Y^T X} = \mathbf{XY} = \mathbf{X'}$, where $\mathbf{X'} \in \mathbb{R}^k$ is the result of n-dimensional data being projected into a k-dimensional subspace. Note that for a sample size $m > 1$, the transformation $\mathbf{Y^T X} = \mathbf{X'}$ can still be used where $\mathbf{X} \in \mathbb{R}^{n\times m}$ and $\mathbf{X'} \in \mathbb{R}^{k\times m}$.

By using these optimisation methods we can quantify how important certain features of a dataset are when the goal is to distinguish the difference between the classifications in the data. Hence, these methods can be used for identifying redundant features and dimensionality reduction. A classified dataset requires that for each data point in the set, there is a corresponding class or label. For example, in a dataset with two different classes, a linear combination of variables can be found to construct two new axes. By using one of the optimisation algorithms, along with a SVM (with a linear kernel), a projection can be found that will maximally separate the two classes in the 2-dimensional subspace.

2 Mathematical Theory

2.1 Stiefel Manifold

As mentioned in the previous section, the Stiefel manifold, $V_k(\mathbb{R}^n)$ or $V_{n,k}$, is the set of all orthonormal k-frames in $\mathbb{R}^n$ space. Each point on the manifold is represented by a $\mathbb{R}^{n\times k}$ matrix and hence the Stiefel manifold is a submanifold of $\mathbb{R}^{n\times k}$ space. For a $n \times k$ matrix, $\mathbf{Y} \in V_{n,k}$, representing a point on the Stiefel manifold, the elements have to satisfy the constraint condition,

$$\mathbf{Y}^T\mathbf{Y} = \mathbf{I}_k \tag{1}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix.

The Stiefel manifold is closely related to the Grassmannian or Grassmann manifold, $G_{n,k}$, which parameterises all k-dimensional linear subspaces of an n-dimensional space. By considering a single point on the Stiefel manifold, $\mathbf{Y} \in V_{n,k}$, any rotation from the orthogonal group O_n, applied to $\mathbf{Y}$, produces another point on the Stiefel manifold. A single point on the Grassmannian is equivalent to the set of all points on the Stiefel manifold that can be found from the transformation of an orthogonal rotation applied to a single point $\mathbf{Y}$. Formally, the Grassmannian, $G_{n,k}$, is the quotient set or quotient space arising from $V_{n,k}$ and the orthogonal group O_n; this can be represented using the notation $G_{n,k} = V_{n,k}/O_n$. Hence, a point on the Grassmannian is not uniquely defined by a single $n \times k$ matrix. The Stiefel manifold can also be defined as the quotient space arising from the orthogonal group via $V_{n,k} = O_n/O_{n-k}$[5].

There are two special cases of Stiefel manifolds that can be helpful to consider: $k = 1$ and $k = n$. $V_{n,1}$ is simply equivalent to the unit n-sphere, whereas $V_{n,n}$ is equivalent to the orthogonal group O_n.

2.2 Uniform Random Distribution on the Stiefel Manifold

Generating points uniformly at random is a useful tool for implementing optimisation algorithms. Due to the matrix representation of points on Stiefel manifolds having constraints on the matrix elements, this is a non-trivial task.

The method employed in this project was as follows[6][7]:

1. Generate $n \times k$ independent random variables from $N(0, 1)$ - the normal distribution with a mean of 0 and variance 1.
2. Arrange the variables into a $n \times k$ matrix $\mathbf{Y}$.
3. Return $\mathbf{X} = \mathbf{Y}(\mathbf{Y}^\mathbf{T}\mathbf{Y})^{-1/2}$; a random matrix that follows the uniform distribution on $V_{n,k}$.[1]

2.3 Tangent Space

The tangent space of a point, x, on a manifold, M, is a vector space that contains all the directions which are tangential to position x denoted by $T_x(M)$. Through differentiating the constraint given in equation (1), it follows that $\mathbf{Y}^T\dot{\mathbf{Y}} + \dot{\mathbf{Y}}^T\mathbf{Y} = 0$, where $\dot{\mathbf{Y}}$ is a matrix in the tangent space of the

[1]Where a square $k \times k$ matrix $\mathbf{Y}^{-1/2}$ is calculated through diagonalisation by finding the k eigenvalues and corresponding eigenvectors of the matrix $\mathbf{Y}$, such that $\mathbf{Y} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1}$, where $\mathbf{V}$ is the matrix constructed of the k column eigenvectors, and $\mathbf{D}$ is the diagonal matrix made up of eigenvalues along the diagonal. $\mathbf{Y}^{-1/2}$ is then calculated by the equality $\mathbf{Y}^{1/2} = \mathbf{V}\mathbf{D}^{1/2}\mathbf{V}^{-1}$ and then simply taking the inverse of $\mathbf{Y}^{1/2}$.

Stiefel manifold[5][8]. This condition imposes $p(p+1)/2$ constraints on the matrix $\dot{\mathbf{Y}}$ and hence the dimensionality of the tangent vector space is

$$np - \frac{p(p+1)}{2} = \frac{p(p-1)}{2} + p(n-p). \tag{2}$$

where p represents the number of dimensions to project into. Parallel transport is a way of moving geometrical objects along a smooth curve while preserving the initial local structure. An approximate method is used as no known analytical expression exists[9]. A formula for the projection of any $\mathbf{Z} \in \mathbb{R}^{n\times p}$ into the tangent space at $\mathbf{U}$ is given by

$$\pi_T(\mathbf{Z}) = \mathbf{U}(\mathbf{U}^T\mathbf{Z} - \mathbf{Z}^T\mathbf{U})/2 + (\mathbf{I} - \mathbf{U}\mathbf{U}^T)\mathbf{Z}. \tag{3}$$

For the approximate parallel transport method of transporting a vector from $T_{\mathbf{A}}(V_{n,p})$ to $T_{\mathbf{B}}(V_{n,p})$, where $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times p}$, we do the following[9]:

Subroutine 1:

1. Define ϵ, a stopping parameter to prevent numerical instability.
2. Let $l = \|\Delta\|$, where $\Delta \in T_{\mathbf{A}}(V_{n,p})$.
3. Using equation (3), project Δ to $T_{\mathbf{B}}(V_{n,p})$, to give a value for $\bar{\Delta}$.
4. **If** $\|\bar{\Delta}\| > \epsilon$, return $\bar{\Delta} = \bar{\Delta}\, l/\|\bar{\Delta}\|$,
   else return $\bar{\Delta} = 0$.

2.4 Geodesics on the Stiefel Manifold

For various algorithms, we needed to navigate the space in such a way to find the equivalent of straight line between points, hence the use of geodesics. For example, in the particle swarm algorithm, we required a contribution to the displacement in the direction of the current best scoring point on the manifold. A geodesic technique was also implemented in the genetic algorithm, during the creation of a child point from two parent points.

For a Riemannian manifold, if the differential geometric properties of a manifold M are known, a unique geodesic can be calculated. The Stiefel manifold is a Riemannian manifold as we can introduce an inner product in its tangent space[8][9]. There exists two inner products in the space that we want to consider, the Euclidean inner product[2] and the canonical inner product.

An exponential map is a mapping from the tangent space of a manifold to another point on the manifold, $\alpha : T_x(M) \to M$[10]. The exponential mapping from a point $\mathbf{Y}$ takes a shooting vector, $\dot{\mathbf{Y}} \in T_Y(V_{n,p})$, and maps it to a point on the geodesic. This exponential mapping on the Stiefel manifold has an analytic form and hence a geodesic equation can be found using a starting point and a shooting vector. The logarithmic mapping is the inverse of the exponential mapping[3], by using this we can calculate a shooting vector from a starting point and an endpoint. These two mappings can be used to analytically find an endpoint geodesic. However, the Stiefel manifold has no known analytic form for the logarithmic mapping and hence no known analytical formula exists for endpoint geodesics in $V_{n,p}$ space.

[2]The Euclidean inner product can be used as the manifold is embedded in $\mathbb{R}^{n\times p}$ Euclidean space. However, in this project the canonical inner product was used.

[3]This is allowed when the exponential mapping is locally invertible.

This motivates the use of numerical methods to generate endpoint geodesics. To construct this geodesic numerically, we first define the geodesic equation[9],

$$\alpha(t) = [\alpha(0)\ \dot{\alpha}(0)]\ expm \left\{ t \begin{bmatrix} \mathbf{A} & -\mathbf{S}(0) \\ \mathbf{I} & \mathbf{A} \end{bmatrix} \right\} \mathbf{I}_{2p,p}\ expm\left\{-t\mathbf{A}\right\} \tag{4}$$

where $\alpha(0) \in V_{n,p}$ is the starting point on the manifold, $\dot{\alpha}(0) \in T_{\alpha(0)}(V_{n,p})$ is a shooting vector in the tangent space at $\alpha(0)$, t is a step-size parameter, $\mathbf{I}_{2p,p}$ is the $p \times p$ identity matrix stacked on top of a $p \times p$ zero matrix, $\mathbf{A} = \alpha(0)^T\dot{\alpha}(0)$ and is skew-symmetric and $\mathbf{S}(0) = \dot{\alpha}(0)^T\dot{\alpha}(0)$ is a symmetric matrix. $\mathbf{A}$ and $\mathbf{S}$ are found from the integrals of motion from the Euler-Lagrange equation for calculus of variations and is fully described by Edelman et al.[5].

From the geodesic equation (4), we now need to find a suitable shooting vector that will parameterise a geodesic from a specified point, to a another specified point. As no known logarithmic mapping exists, we will need to “guess” the shooting vector, and alter it based on the geodesic endpoint it gives. Bryner[9] provides two numerical schemes to compute the desired geodesic. The first is called the shooting method and the second is a path-straightening method. For this project, we use the shooting method, which has the advantage of producing an exact geodesic, whereas the path-straightening method produces an approximate geodesic but with the exact desired endpoints.

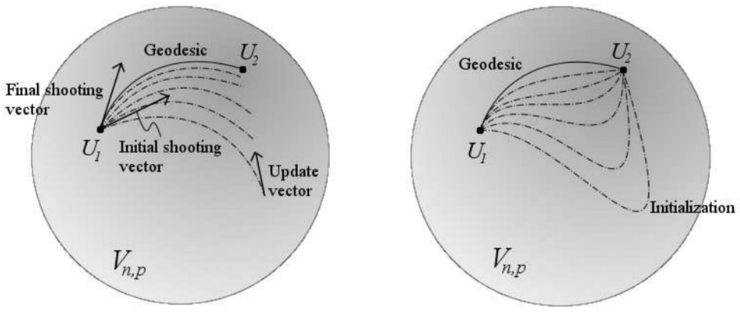

Figure 1: A figure showing a graphical representation of the two approximate geodesic methods[9]. The left image represents the shooting vector method and the right image represents the path straightening method.

In using the shooting point method, a way of generating a new, more accurate shooting vector is required. To do this we use the method stated in Bryner[9]. After one guess of a shooting vector, we can project the error vector into the tangent space via **Subroutine 1**. The algorithm to find the appropriate shooting vector is stated in Algorithm 1.

2.5 Cayley Transformations

During an iterative process on the Stiefel manifold, it can be computationally expensive to conserve the constraints needed on a $\mathbb{R}^{n\times p}$ matrix. However, one can apply a Cayley transformation to deal with the complexity[11][4] of generating new matrices that satisfy equation (1). During a gradient descent algorithm, we use the Cayley transformation to translate through the space on a gradient descent curve while preserving the constraint given by equation (1)[8]. We thus define the curve as the following,

$$\mathbf{Y}(\tau) = \left(\mathbf{I} + \frac{\tau}{2}\mathbf{W}\right)^{-1}\left(\mathbf{I} - \frac{\tau}{2}\mathbf{W}\right)\mathbf{X} \tag{5}$$

where $\mathbf{X} \in V_{n,p}$, τ is a step-size quantity. We set $\mathbf{W} = \mathbf{G}\mathbf{X}^T - \mathbf{X}\mathbf{G}^T$ to make the curve a descent curve on $V_{n,p}$, where $\mathbf{G}$ is a gradient in the space at $\mathbf{X}$ defined by

Algorithm 1 Shooting Vector Calculation Algorithm

```
Given two points $\mathbf{A}$ and $\mathbf{B}$ on the manifold $V_{n,p}$, choose the number of sample points along a geodesic,
$T$, and assign a value for the iteration step size $\delta$.
Assign a value for the convergence tolerance, $\tau$.
Assign the maximal number of iterations, $k_{max}$
Make a first approximation of the shooting vector, $\Delta \in T_A(V_{n,p})$
for $i = 1, \dots, T$ do Compute sample point $i$ using the shooting vector $\Delta$ at $\mathbf{A}$, and the geodesic
    equation (4) with $\alpha(t_i)$ where $t_i = i/T$

while $k < k_{max}$ or $\|\omega\| > \tau$ do
    Assign: $k := k + 1$
    Compute the error of the endpoint of the geodesic $\omega = \alpha(1) - \mathbf{B}$, and project $\omega$ into the tangent
    space $T_{\alpha(1)}(V_{n,p})$ using equation (3).
    for $i = T, T-1, \dots, 1$ do Using Subroutine 1, parallel transport $\omega$ from $T_{\alpha(t_i)}(V_{n,p})$ to
        $T_{\alpha(t_{i-1})}(V_{n,p})$

    Update $\Delta := \Delta - \delta\omega$
    for $i = 1, \dots, T$ do Calculate $\alpha(t_i)$ at $\mathbf{A}$ using equation (4) with updated $\Delta$
```

$$\mathbf{G} = \left[\frac{\partial F}{\partial \mathbf{X}_{i,j}}\right] \in \mathbb{R}^{n \times p} \tag{6}$$

where F is the single-valued cost function evaluated at $\mathbf{X}$, $F : \mathbb{R}^{n\times p} \to \mathbb{R}$. In practice, each element of the matrix was altered by a small value δ and the cost function was evaluated for each matrix applied to the dataset, hence this was one of the most computationally expensive parts of the algorithms. The tangent vector at $\mathbf{X_t}$, is thus $\mathbf{Y}'(t) = -\mathbf{W}_t\mathbf{X}_\tau$. As equation (4) requires the inversion of the term $\left(\mathbf{I} + \frac{\tau}{2}\mathbf{W}\right)^{-1}$ we can implement the use of the Sherman-Morrison-Woodbury (SMW) formula[12]; a computationally more efficient way of inverting matrices of a certain form[8]. The SMW formula is defined as

$$\left(\mathbf{B} + \alpha\mathbf{U}\mathbf{V}^T\right)^{-1} = \mathbf{B}^{-1} - \alpha\mathbf{B}^{-1}\mathbf{U}\left(\mathbf{I} + \alpha\mathbf{V}^T\mathbf{B}^{-1}\mathbf{U}\right)^{-1}\mathbf{V}^T\mathbf{B}^{-1} \tag{7}$$

where $\mathbf{U} = [\mathbf{G} \quad \mathbf{X}]$ and $\mathbf{V} = [\mathbf{X} \quad -\mathbf{G}]$, so therefore $\mathbf{W} = \mathbf{U}\mathbf{V}^T$. By substituting equation (7) into equation (5) we get a computationally simpler version of the descent curve,

$$\mathbf{Y}(\tau) = \mathbf{X} - t\mathbf{U}\left(\mathbf{I} + \frac{t}{2}\mathbf{V}^T\mathbf{U}\right)^{-1}\mathbf{V}^T\mathbf{X} \tag{8}$$

where the term $\left(\mathbf{I} + \frac{\tau}{2}\mathbf{V}^T\mathbf{U}\right)^{-1}$ is the inversion of a $2p \times 2p$ matrix and thus more computationally efficient that the $n \times n$ counterpart $\forall\ 2p < n$ matrices.

3 Algorithm Theory

3.1 Gradient Descent

A gradient descent algorithm is an algorithm that uses the gradient at a point $\mathbf{X}$ to define a direction, that maximally decreases a single-valued cost function $F(\mathbf{X})$, in which to move. The main disadvantage of gradient descent is that the iteration of points can often fall into regions of a space that form a basin of attraction, these can be saddle points in the space or local minima[4][13]. This problem motivates the stochastic gradient descent algorithm that will be discussed in the next subsection.

From the geometric properties of the Stiefel manifold defined in section 2, an implementation of a gradient descent algorithm has added complexities in comparison to a regular Euclidean space. To preserve the constraint from equation (1), the Cayley transformation (8) is used to define the descent curve though the space $V_{n,p}$. Algorithm 2 describes the method used.

Algorithm 2 Gradient Descent Algorithm

```
Evaluate the cost function $F(\mathbf{X}_0)$, and assign as the new minimum.
Assign a step size $\tau$, and a maximum amount of iterations $k_{max}$.
Assign a stopping condition for when $\|\mathbf{WX}\|$ is below a scalar value $\epsilon$.
while $\|\mathbf{WX}\|_k > \epsilon$ or $k < k_{max}$ do
    Assign: $k := k + 1$
    Calculate $\mathbf{W}_k = \mathbf{UV}^T|_k$ from the relevant values of $\mathbf{U}, \mathbf{V}, \mathbf{G}, \mathbf{X}$
    Calculate $\mathbf{Y}_k(\tau)$ from equation (7)
```

3.2 Stochastic Gradient Descent

As was described in the previous subsection, the disadvantage of the gradient descent is its tendency to get stuck in local minima or at saddle points. A stochastic gradient descent involves adding extra stochastic terms in an attempt to perturb out of a basin of attraction.

The stochastic gradient descent is implemented in a method similar to the regular gradient descent, however, the $\mathbf{W}$ term in equation (4) is altered before the Cayley transformation is applied[14]. The stochastic alteration to $\mathbf{W}$ is defined as $\mathbf{A}$, but first we define $\mathbf{Z} \in \mathbb{R}^{n\times p}$, a modified version of $\mathbf{G}$ given by the following equation

$$\mathbf{Z} = -\delta\mathbf{G} + \sigma(\mathbf{I}_n - \beta\mathbf{XX}^T)\mathbf{B}_\delta \tag{9}$$

where $\mathbf{G}$ is the gradient as defined by equation (5), $\beta = 1 - \sqrt{2}/2$, $\mathbf{B}_\delta \in \mathbb{R}^{n\times p}$ is a $n \times p$ matrix with elements drawn from an independent normal distribution $N(0, \delta)$ at each iteration, and $\delta, \sigma \in \mathbb{R}$ are constant scalar parameters. We can now define $\mathbf{A} = \mathbf{ZX}^T - \mathbf{YZ}^T$. The algorithm is then implemented in the same form as the gradient descent, Algorithm 1, with the modified $\mathbf{U}$, $\mathbf{V}$ given by $\mathbf{U} = [\mathbf{Z} \;\; \mathbf{X}]$, $\mathbf{V} = [\mathbf{X} \;\; -\mathbf{Z}]$, with a unique $\mathbf{B}_\delta$ being generated at each iteration.

3.3 Genetic Algorithm

A genetic algorithm is a partial search algorithm that draws inspiration from the biological process of natural selection[15]. There are five phases in a standard genetic algorithm:

[4]In high dimensional spaces, local minima are usually less frequent than saddle points. This is discussed in [13].

1. **Initialise population**: Generate a population distributed uniformly at random over the domain.
2. **Fitness evaluation**: Evaluate the cost of all members in the population.
3. **Selection**: Select the best scoring (lowest cost function values) points, other points are removed.
4. **Crossover**: Pair up the surviving points in the population and create child points. Child points were constructed by taking the midpoint of a geodesic between the two parents.
5. **Mutation**: The mutation stage adds a stochastic element to the child, moving the point randomly in a small vicinity.

There has been no previous implementation of a genetic algorithm on a Stiefel manifold to the best of the author's knowledge. The first three phases of a genetic algorithm implemented on a Stiefel manifold are easy to implement on any space, however the crossover and mutation phases are more challenging.

Once two parents are selected, a child point is produced by using the geodesic equation (3). When the geodesic equation has been found, the halfway point is selected before mutation. At the mutation phase, the geodesic equation is used again with a randomised shooting vector. The algorithm is implemented as described in Algorithm 3:

Algorithm 3 Genetic Algorithm

```
Generate the initial population of $N$ matrices from a uniform random distribution over the Stiefel manifold.
Assign the fraction of the population that survive each generation, $\gamma$, such that $\gamma N \in \mathbb{N}$ ($\gamma N$ is a positive integer).
Assign the maximal number of generations to compute, $T$.
Assign a value for the variance of the mutations, $\sigma^2$.
while $t < T$ do
    Assign: $t := t + 1$
    Evaluate the cost for each point in the generation $F(\mathbf{X}_i)$ where $i = 1, 2, \ldots, N$
    Choose the best scoring $\gamma N$ points from the population, and remove the remaining points
    for $i = 1, 2, \ldots, \gamma N$ do
        Complete preferential attachment to connect with another member of the surviving population (probability of attachment is proportional to the inverse of the cost)
        Using the geodesic algorithm, find the midpoint on the geodesic between the parents
        Mutate the point using a shooting vector along with the geodesic equation with elements drawn from a normal distribution $N(0, \sigma^2)$
        Assign the new population as surviving parent points and new child points
```

3.4 Particle Swarm Optimisation Algorithm (PSO)

The particle swarm optimisation algorithm initially fills the space with N particles or initial points. Each particle/point is transformed in three directions at each iteration:

1. Firstly, in the direction of the best scoring (lowest cost function value) particle out of the entire swarm.
2. Secondly, in the direction of that particle's previous best scoring position, during its lifetime.
3. Finally, the particle moves in the direction decided by the regular gradient descent scheme discussed in section 3.1.

The aim of a particle swarm is that by adding lots of points at different positions uniformly at random in the space, that one will be more likely to find a global minimum as each particle will try to explore its local space, but also gradually tend towards the current best minima. A particle is defined as being the point on the manifold $V_{n,p}$. If all the particles start converging towards the current minimum, the idea is that these other particles will also be able to explore other minima, and then eventually the neighbouring space as they converge on the best found cost.

One complication that we had during the building of this algorithm was during the implementation of moving in the three different directions. Ideally you would move once in a single, combined direction, however, because of the geometry of the space, moving in different directions is dependent on the order the transformations are applied. The algorithm is stated explicitly in Algorithm 4.

Algorithm 4 Particle Swarm Optimisation Algorithm

```
Generate the initial population of $N$ matrices from a uniform random distribution over the Stiefel manifold. Note $N$ is a dynamic value that can change as particles are removed from the swarm.
Assign values for geodesic parameters $\alpha$ and $\beta$.
Evaluate the cost of each particle in the population and assign the best scoring cost as the swarm best $F_{best}$.
while $k < k_{max}$ or $\mathbf{X}_i \approx \mathbf{X}_j \ \forall\, i, j$ (equivalent to $N = 1$) do
    Assign: $k := k + 1$
    If any particles in the population are at or very near the same point, defined as $\|\mathbf{X}_i - \mathbf{X}_j\| < \epsilon$, then remove the worst scoring point (or either of the points if the score is the same)
    Evaluate the best scoring particle and if necessary, assign new max particle $j$
    for Particles $i = 1, 2, \ldots, N$ do
        Evaluate if $i$ is at a new personal best, and if necessary assign as $i_{best}$
        if $i \neq j$ then Calculate the geodesic between $i$ and $j$, and move in that direction with a normalised shooting vector with a scalar value $\alpha$
        if $i \neq i_{best}$ then Calculate the geodesic between $i$ and $i_{best}$, and move in that direction with a normalised shooting vector multiplied by a scalar value $\beta$
        Perform a Cayley transformation as given in equation (7) to apply a gradient descent translation
```

3.5 Support Vector Machine (SVM)

Support vector machine, or SVM, is a supervised learning model that finds a decision boundary that maximises the margin between two or more classes[16]. There are two key ways that the success of the SVM algorithm was evaluated. The first is the accuracy - how many points were correctly classified divided by how many points there were in total. Although this was what we wanted to maximise, the accuracy function is not continuous over the space. This motivated the second way of quantifying the success of the algorithm, the hinge loss. The hinge loss is a measure of how far the points are away from the decision boundary (the boundary at which points on one side are predicted to be of one class, and predicted to be a different class on the other side).

The SVM algorithm finds the hyperplane that gives the maximum margin size. For linearly separable data, the (hard) margin M is defined as the distance from the boundary to the surface parallel to the boundary line that excludes all data points. Figure 2 shows an example of a SVM on a linearly separable dataset. Very often the data is not perfectly linearly separable, and some points will have to be on the wrong side of the margin. These points are then given a slack variable ε_i measured as the distance inside the margin, and hence the margin M is maximised with the constraints as follows,

$$\begin{gathered} y_i(x_i^T\beta + \beta_0) \geq M(1-\varepsilon_i),\ \forall i \\ \sum_{i=1}^{m} \varepsilon_i = \text{constant} \end{gathered} \tag{10}$$

where x_i is the n-dimensional data and y_i is the target variable for a single data point i, formally $y_i \in \{-1, 1\}$. The sample size is defined as m, and β, β_0 are the coefficients of the plane where $M = \frac{1}{||\beta||}$[17].

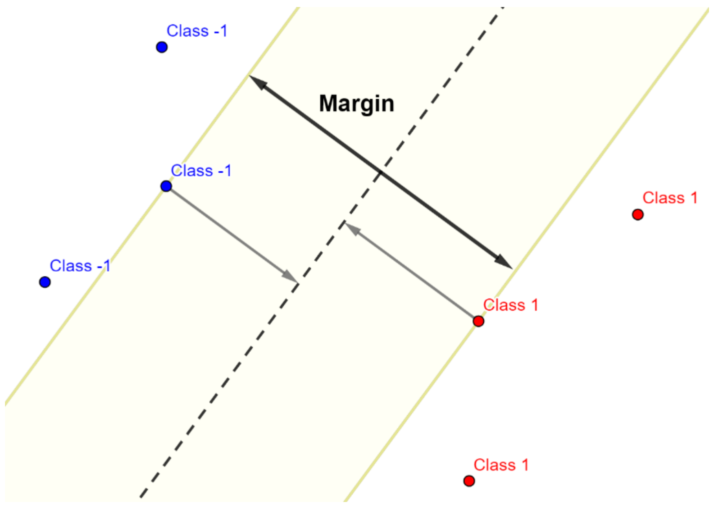

Figure 2: Figure showing a margin (black line) between two classes of data. The variable M is shown by the grey lines.

A linear kernel was used throughout the project, however other non-linear kernels exist. The kernel describes the type of function used to create the transformation to form another dimension.

The SVM evaluates a binary decision, however, when more than two classes are being evaluated the problem needs to be simplified for the computation of the boundary. Often what is used is a "One-Vs-Rest" strategy, in which one class is treated as normal but the remaining classes are combined together.

Hinge loss is a cost (or loss) function that penalises points on the wrong side of the margin boundary. If points are on the correct side of the boundary and beyond the margin, they contribute 0 to the cost function. However, any point that is either within the margin or on the wrong side of the boundary contributes to the hinge loss. The amount individual points contribute to the hinge loss is proportional to the distance from the margin on the correct side of the decision boundary. Hence, we define the hinge loss l as

$$l(x, y) = \sum_{i=1}^{m} max(0, 1 - y_i(x_i^T \beta + \beta_0)) \tag{11}$$

where the variables have the same meanings as equation (10) and the sum is over all samples in the dataset.

4 Results

There are two classes of results presented in this report. Firstly, the success and effectiveness of the optimisation algorithms. Secondly, how well the optimisation results can be used for dimensionality reduction. The second results are clearly dependent on the first results. In order to give a sense of the likelihood of getting certain values for the cost function, 10^6 Stiefel matrices were generated uniformly at random and their cost function and accuracy were evaluated on the "Wine" dataset, this can be seen in Figure 3. None of the 10^6 matrices produced an accuracy of 100%, and the hinge loss had a mean of $\mu = 0.6942$ and a standard deviation of $\sigma = 0.1878$. Although accuracy and hinge loss are not perfectly correlated, there is a strong correlation between the two variables, and due to the requirement of having a continuous cost function, hinge loss was used.

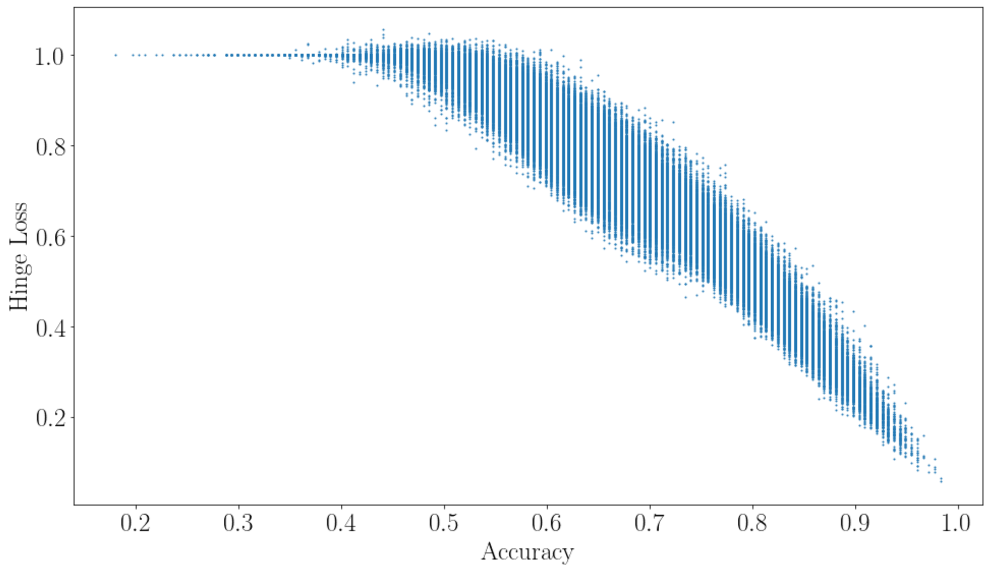

Figure 3: A figure showing a scatter plot of the hinge loss values and corresponding accuracy for 10^6 matrices on the Stiefel manifold, generated uniformly at random. This was applied to the "Wine" dataset. The discrete lines on the vertical are observed due to the accuracy measure being discrete.

4.1 Optimisation Algorithms

In general, the main focus was a correct implementation of the algorithms and computational efficiency was less important. All these algorithms would adapt well to parallel computing using graphical processing units (GPU). For some algorithms, Python's multiprocessing library was used to improve the speed of the computation. The speed analysis was run on a Dell XPS 9560 laptop with 16GB RAM and an Intel i7-7700HQ 4 core CPU @ 2.80GHz.

Due to the stochastic elements in the various algorithms, when conducting analysis, the algorithms were ran multiple times and an average was taken. This was to avoid any discrepancies from effects such as randomly picking a low initial cost or alternatively, a point that had an unusually good path to a low cost.

The "Wine" dataset contains 13 variables and 3 classes, each class represents a region from where a particular wine has been sourced from. Figure 3 shows a projection of the "Wine" dataset with a low hinge loss value with a 100% SVM classification accuracy, an example of a high hinge loss can be found in the appendix. The "Particle" dataset contains 7 variables and 2 classes - background and signal.

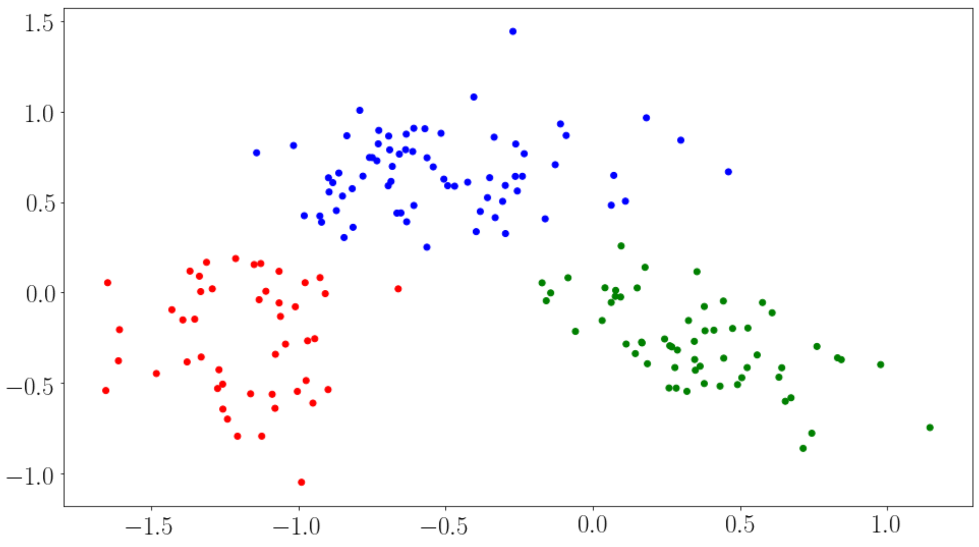

Figure 4: Figure showing a projection of the "Wine" dataset with a low hinge loss, and 100% accuracy. The x and y axes are a combination of the original variables defined by the matrix $\mathbf{X} \in V_{n,p}$.

4.2 Parameter Tuning

Various tests needed to be made in order to find the best parameters for each algorithm on a specific dataset. In the project, a grid search or parameter sweep was performed. Other, more efficient methods do exists[18][19] but they were not employed in this project. This meant exhaustively testing each combination of parameters. This fine tuning was extremely computationally expensive as many loops needed to be run for each set of parameters, and also each parameter needed to be tested for all other possible parameter combination, as the parameters may not be independent of each other. Due to the computational cost, this analysis was only performed on the "Wine" dataset.

For the gradient descent, the value that was chosen to be best for the step size, was $\tau = 0.05$. The test for the cost against time for different τ values is plotted in Figure 5. Other tuning graphs can be found in the appendix. The final values used for parameters of the stochastic gradient descent were $\tau = 1$, $\delta = 0.05$, and $\sigma = 0.1$; the graphs that show why these values were chosen are included in the appendix. For the genetic algorithm, the values for the parameters were chosen as population size $N = 2^8$, variance $\sigma^2 = 0.05$, and the reciprocal of the fraction of surviving points $\gamma = 7$. The graphs are also included in the appendix.

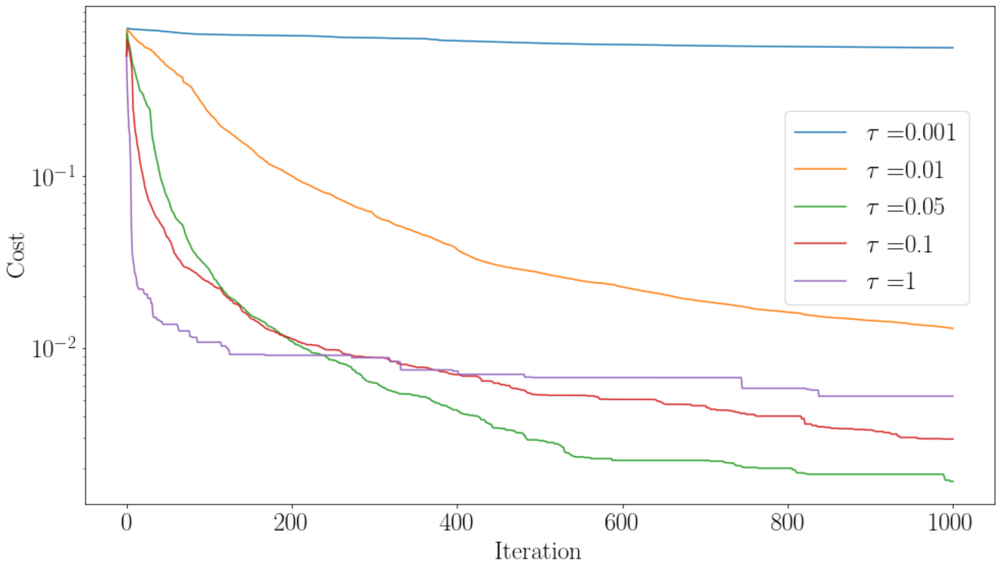

Figure 5: Figure showing how different values of τ effected the number of iterations it took for convergence in the gradient descent algorithm.

Unfortunately, some errors were arising in the particle swarm algorithm that meant a full analysis could not be done. After around 70-120 iterations, the A, and S values seemed to blow up and eventually become too large for meaningful calculations to be made. The initial guess for why this problem occurs in the calculation of the shooting vector might be caused when the approximate shooting vector is being calculated for points that are too close to each other for the approximate method to be valid. As this was the last algorithm to be implemented, there was not enough time to correct this issue. For the intial iterations, results were seen to perform better than the stochastic algorithm on the "Wine" dataset but slightly worse than the genetic and gradient descent algorithms.

4.3 Performance Comparisons

The "No free lunch" theorem for optimisation, states that all algorithms that search for optima of a cost function perform the same when averaged over all possible cost functions and datasets[20]. This varying efficiency of the different algorithms when testing on different datasets was observed with the Stiefel optimisation algorithms.

For the "Wine" dataset, the gradient descent algorithm performed the best on average; this can be seen in Figure 6. It was relatively easy to find a perfect linear separation for the dataset with all the algorithms, but the lowest costs were usually found by the gradient descent. This is probably due to the scarcity of local minima over the domain, due to relatively small sample size of 177 and the large dimensionality. Increases in dimensionality tend to see a decrease in the number of local minima, this is discussed in Duaphin[13].

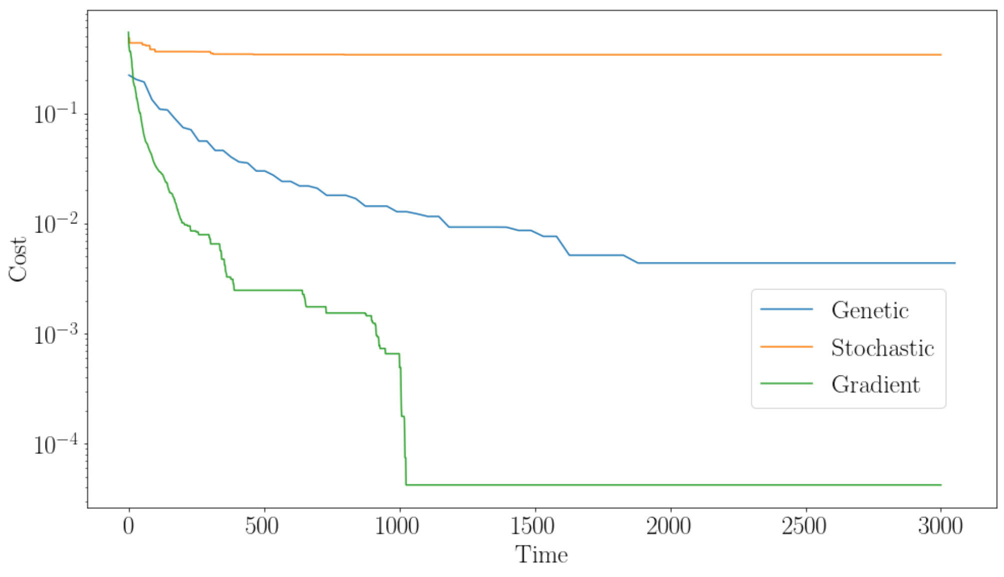

Figure 6: Figure showing the relative performances of the algorithms on the "Wine" dataset. The hinge loss is plotted on the y-axis and the time in seconds on the x-axis.

An example of where the genetic algorithm exceeds above the others, is the particle dataset. Figure 7 shows how the hinge loss varied with time on the "Particle" dataset. When datasets are not perfectly linearly separable, and have lots of variance in the hinge loss over the domain, the gradient descent can easily get stuck at local minima or saddle points. The stochastic gradient descent can normally overcome these stationary points where the regular gradient descent might get stuck, but is still somewhat susceptible to these regions. The genetic algorithm performs the best on these types of spaces.

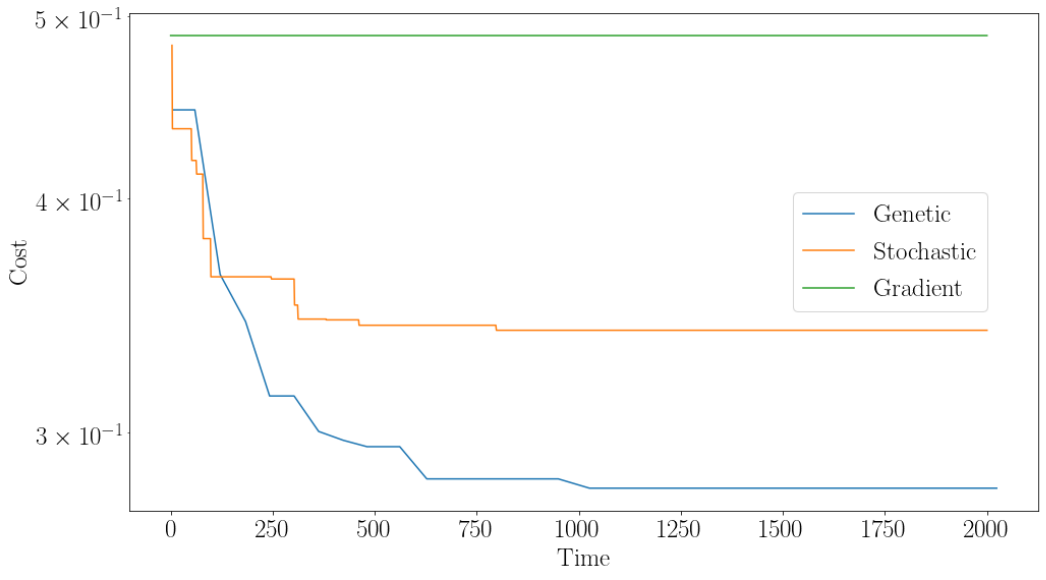

Figure 7: Figure showing how the SVM hinge loss changed with time (in seconds) when applied to the "Particle" dataset. This was for one run of each algorithm. The hinge loss is plotted on the y-axis and the time in seconds on the x-axis.

4.4 Dimensionality Reduction

Dimensionality reduction describes the concept of removing dimensions from a feature space while trying to preserve as much key information in the data. Other techniques for this can either reduce discrete features, or construct new dimensions containing a combination of lesser prevalent features.

To find the contributions of each variable to the final projection, the square of each element in a row is added, and the square root is taken of the sum,

$$C_i = \sqrt{\sum_{j=1}^{p} \mathbf{X}_{i,j}^2} \tag{12}$$

where $\mathbf{X}_{i,j}$ is the i^{th} row and j^{th} column of the matrix $\mathbf{X} \in V_{n,p}$.

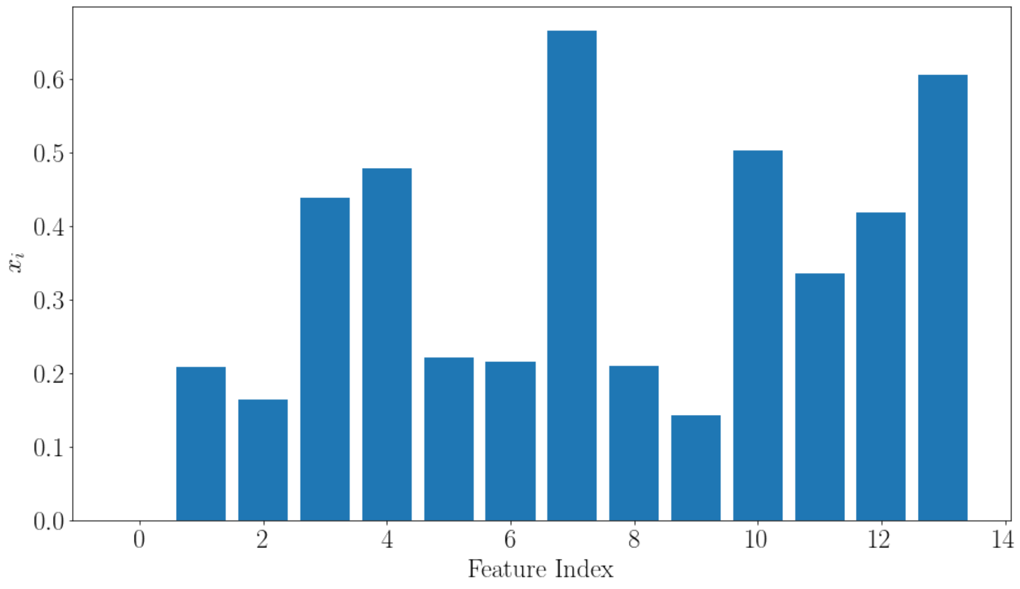

Figure 8: A figure showing the prevalence of each variable (in both the x and y axes) from the "Wine" dataset featured in an optimal 2D projection. Calculated using equation (12).

Figure 8 shows the relative weights for an optimal project in 2D; from this we can interpret that feature 7, flavonoids, and feature 13, proline, are the most important variables when trying to linearly separate each class of data. Alternatively, if one allows for a linear combination of many variables to construct new variables, then the weightings are found from the two optimal rotation matrix columns.

4.5 Applied Example

To test the accuracy of the technique, a breast cancer dataset was used that contained 30 variables, and 2 classes. The classes were either a diagnosis of a tumor begin benign or malignant. The dataset had a sample size of 569, in which 80% of the data was used as training data to use in the algorithm, and 20% was used to test the accuracy of the SVM model at the optimal projection. A gradient descent model was used for 200 iterations to find an optimal projection matrix, this took 56.02 seconds. An accuracy of 98.2% was observed when applying the test data to the optimal projection and SVM model, and the projected test data can be seen in Figure 9.

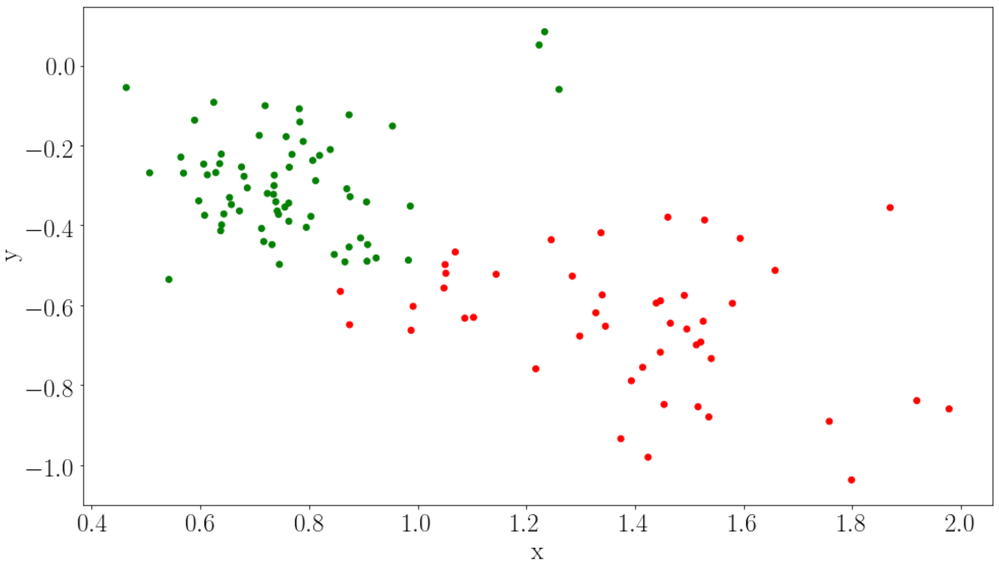

Figure 9: This figure shows breast cancer test data on an optimal projection found from separate training data. The linear SVM had an accuracy of 98.2%. The red points represent malignant tumors and the green points represent benign tumors.

For bench-marking this method, the SVM was calculated in the 30-dimension feature space. The accuracy was found to be identical at 98.2% for the 30-dimensional SVM, which implies that the information regarding the linear separation has been preserved when reducing the dimensions of the space by 28. Figure 10 shows the percent each variable was incorporated into the two axes.

5 Discussion

In this project, we have designed two new optimisation algorithms on the Stiefel manifold: the genetic algorithm and the particle swarm optimisation algorithm[5]. We have also successfully implemented two algorithms that have previously been studied on the manifold: the gradient descent and stochastic gradient descent. Using these optimisation algorithms on the Stiefel manifold, we have shown that they can be used to produce 2D plots of high dimensional data in such a way that the data can often be linearly separated. In finding these projections, we have also shown that the optimal projection matrix can be used to investigate how important each variable is if the goal is to linearly separate the data, hence, a new method of dimensionality reduction. The practical implication for this, is that in classification problems, we can see which features are redundant and hence, the important features can be focused on. Alternatively, the final optimal 2D projection is by definition another form of dimensionality reduction.

From looking at a classification dataset of breast cancer tumors with the associated labels of malignant or benign tumors, when applying these algorithms, we can find an optimal projection to find a linear separation between the two classes of tumors. Then, by studying the rotation matrix for that projection, we can see which variables are most important for discriminating between benign and malignant tumors. This can shape future diagnosis tests by removing the tests for features that were included in

[5]Although some bugs for this still need to be fixed, the PSO algorithm still calculates results for $\sim$100 iterations. An example of this is included in the appendix.

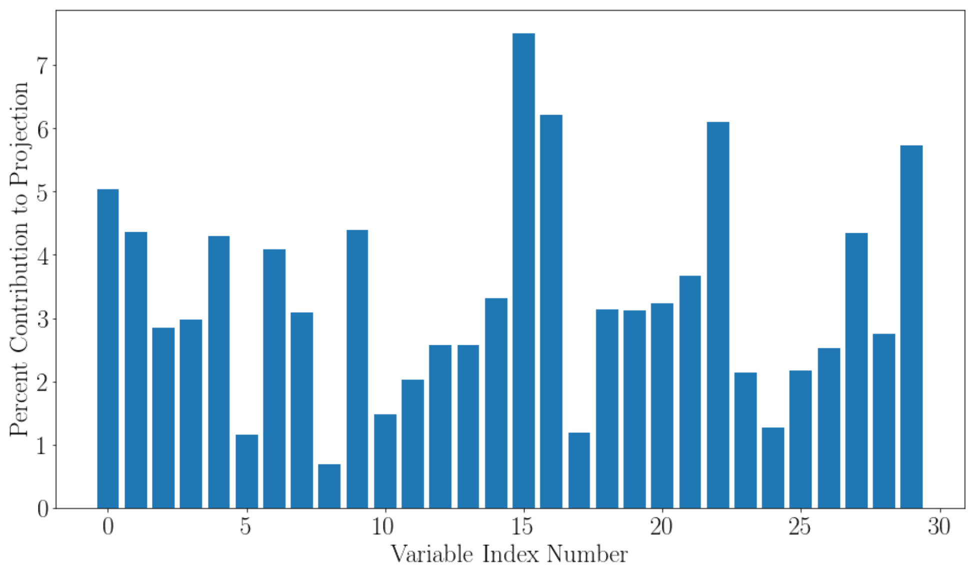

Figure 10: For the optimal 2D projection of the breast cancer data, this figure shows which variables were most prevalent in the two axes, indicating importance of each variable for linear separability.

the dataset yet had negligible input for discriminating between the two classes; hence saving time and costs, and more accurately diagnosing tumors. There is a wide range of applicability for these algorithms, and as computational technology becomes more powerful bigger and bigger datasets can make use of these techniques. A study analysing the accuracy of clinical examinations of breast lumps for determining malignancy showed that clinical assessments provided an accuracy of 90.8%[21], whereas the technique used in this project provided an accuracy of 98.2%. However, it should be noted that this comparison is made between two different samples, and thus to make an accurate comparison you should carry out these tests on the same sample. The implementation of this was using a linear SVM loss function, but if a more complex learning function, or a non-linear SVM kernel was used, it is likely that this accuracy would be able to be improved. As indicated by the matching optimum 2D projection SVM accuracy and native feature space SVM accuracy, there may be a link between the exact hyperplane found in the native feature space and the contribution of the axes in the 2D space.

A method that is commonly used for dimensionality reduction in data science today is principle component analysis (PCA)[22][23]. PCA works by identifying linearly uncorrelated variables , the principal components are ordered in such a way that the first component has the maximum variance. The dimensions of the space can then be reduced by only taking the first p ($p < n$ for a n-dimensional dataset) components as the components will have maximal variance between them whereas the first removed component would have had the maximal correlation between variables, hence less useful for discrimination.

By defining a cost function acting on a projected dataset, the way in which the data can be optimised and subsequently reduced in dimensionality is entirely up to the user. With the current personal computer resources, the algorithms presented in this report are computationally expensive and work on larger timescales when compared with most optimisation algorithms in $\mathbb{R}^n$ space. The speed of algorithms involving a form of gradient descent is very dependent on the number of dimensions of the Stiefel manifold being considered ($V_{n,p}$ where n is the number of features and p is dimensionality of the projection space). However, the Genetic algorithm scales a lot better with the number of dimensions due to not having to calculate the derivative. The calculation of the derivative involves altering each element in the matrix a small amount, moving it away from its current position, then each one of the matrices involves applying the projection to the data and subsequently evaluating the SVM of that projection. During implementing these algorithms, Python's multiprocessing library was used to speed up the calculations required for computing the derivative. However, we were not able to implement parallel computing with the use of general purpose computing on graphical processing units (GPGPU). It should be noted that all the algorithms have components that would greatly benefit from being implemented on parallel computing platforms.

As previously mentioned, any single-valued, continuous cost function can be used. Because of the flexibility in the way these algorithms have been implemented, the cost function can be completely different, or alternatively, modifications can be made to an existing cost function for a specific goal or purpose. For example, when using this algorithm for dimensionality reduction, extra terms can be added to the hinge loss to penalise features that contribute a "medium" amount to the projection, hence to encourage searching for projections that have their axes made up of fewer key variables. Obviously this would require the added term to be scaled correctly in order to not overly discriminate "medium" contributing features such that features can't switch from being a high (low) contributor to a low (high) contributor in one of the axes.

There are many areas in which these algorithms can be improved. A simple way of improving these algorithms is to have adaptive parameters. There is a plethora of research on adaptive or dynamic parameters that are altered to improve speed of convergence. The learning rate or the step-size in the algorithms that use gradient descent is one such parameter. Among the most commonly used methods is the Adam optimiser[24], which dynamically alters the step-size by using estimates of lower order moments. The advantages of using an adaptive learning rate is that there would not be any need for tuning the parameters by running lots of tests through a grid search method, with the added benefit of having a speed increase in finding optima. Alternatively, some other tuning methods could have been used[18][19].

Another place for improvement would be to implement or understand the mathematics needed to combine the three directions in the PSO. Ideally, there would be only one step in the PSO, however, due to our limited knowledge of the geometry of the space, we were not able to find a method to condense the three steps into one. The operation analogous to this in Euclidean space, can be found by adding the three displacement vectors.

The genetic algorithm could potentially be improved by evaluating multiple points neighbouring the midpoint of the geodesic and picking the best, and doing this evaluation without the mutation component. This would be implemented with a percent of the points not having the mutation phase of the algorithm and these points that do not mutate would perform a more detailed search of the loss function values around the midpoint of the geodesic. Preliminary tests of this evaluated the cost over the entirety of the geodesic, however, this did not perform well as the majority of the time, points would be very close to one of the parents. This then doesn't allow for genetic diversity, as points will start grouping together which defeats the point of the algorithm.

Although improvements can be implemented, it is worth noting that all algorithms were easily able to find projections of 100% accuracy on the "Wine" dataset, and found these projections quickly. In the previous semesters work, the data was split into training and test data, however, due to the aim of the project it was decided that we wouldn't include this step as the focus was on building the optimisation algorithms. Further tests should be made using randomly split test and training data, and for any cost functions that are prone to over-fitting, this is crucial. It was unrealistic to test this in the time-frame given, as this method requires datasets with larger sample sizes and hence the algorithms would be slower to run. Also, a linear SVM is one algorithm in which over-fitting is less of a problem in comparison with other learning algorithms or non-linear kernels.

The Stiefel manifold, as discussed in section 2.1, contains a set of points on $V_{n,p}$ that are uniquely defined by a single point on the Grassmann manifold $G_{n,p}$. In other words, when finding one projection from the Stiefel manifold, any rotation matrix from the orthogonal group O_n can be applied to the matrix without changing the point on the Grassmannian. Certain cost functions are independent of this rotation, hence in that case, an optimisation on the Grassmannian would be preferable to reduce the number of points on the manifold that are equivalent with regards to the cost function.

6 Conclusion

In conclusion, four optimisation algorithms have been implemented on the Stiefel manifold. Two of which (gradient descent and stochastic gradient descent) have been already studied in this space[11][14]. The other two (genetic algorithm and particle swarm) were designed from using previous research into the mathematical geometry of the space and aspects of other algorithms. The application of these optimisation methods can be used to find a lower dimensional projection of a higher dimensional feature space, in such a way, that the projection has an optimal scoring for a loss function. The algorithms were implemented in a generic way such that any cost/loss function can be used, and any lower dimensional projection can be found.

The specific application that was focused on in this project was 2-dimensional projections of a dataset, evaluated by a linear SVM and hinge loss function. This method finds the projection that best linearly separates out the data. In the process of finding this projection, the features that are more prevalent in the two final axes, are features that are more important to discriminate between the classes used in the dataset. Practical applications of this are to better identify features that are redundant for predicting a certain classification, and thus identifying variables that are not required in further data gathering. Because the cost/loss function can be general, a user can customise the way in which they want to identify how redundant features are for a specific purpose. This method has shown useful potential for identifying redundant features and performing dimensionality reduction.

References

[1] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and structural biotechnology journal*, vol. 13, pp. 8–17, 2015.

[2] D. Dent and K. Mukhi, "Data mining using the grand tour algorithm," tech. rep., MPhys Report, The University of Manchester, 2019.

[3] D. Asimov, "The grand tour: a tool for viewing multidimensional data," *SIAM journal on scientific and statistical computing*, vol. 6, no. 1, pp. 128–143, 1985.

[4] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds.* Princeton University Press, 2009.

[5] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM journal on Matrix Analysis and Applications*, vol. 20, no. 2, pp. 303–353, 1998.

[6] Y. Chikuse, *Statistics on special manifolds*, vol. 1. Springer Science & Business Media, 2003.

[7] G. Camano-Garcia, "Statistics on stiefel manifolds," 2006.

[8] H. D. Tagare, "Notes on optimization on stiefel manifolds," tech. rep., Yale University, 2011.

[9] D. Bryner, "Endpoint geodesics on the stiefel manifold embedded in euclidean space," *SIAM Journal on Matrix Analysis and Applications*, vol. 38, no. 4, pp. 1139–1159, 2017.

[10] J. Jost and J. Jost, *Riemannian geometry and geometric analysis*, vol. 42005. Springer, 2008.

[11] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, 2013.

[12] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *The Annals of Mathematical Statistics*, vol. 21, no. 1, pp. 124–127, 1950.

[13] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Advances in neural information processing systems*, pp. 2933–2941, 2014.

[14] H. Yuan, X. Gu, R. Lai, and Z. Wen, "Global optimization with orthogonality constraints via stochastic diffusion on manifold," *arXiv preprint arXiv:1707.02126*, 2017.

[15] J. H. Holland, "Genetic algorithms," *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[16] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[17] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[18] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[19] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, pp. 2546–2554, 2011.

[20] D. H. Wolpert, W. G. Macready, *et al.*, “No free lunch theorems for search,” tech. rep., SFI-TR-95-02-010, Santa Fe Institute, 1995.

[21] C. Ravi and G. Rodrigues, “Accuracy of clinical examination of breast lumps in detecting malignancy: A retrospective study,” *Indian journal of surgical oncology*, vol. 3, no. 2, pp. 154–157, 2012.

[22] K. Pearson, “Liii. on lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[23] S. Wold, K. Esbensen, and P. Geladi, “Principal component analysis,” *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *ICLR*, 2015.

A Extra Figures

Figure 12 shows an example of a high hinge loss on the Wine dataset.

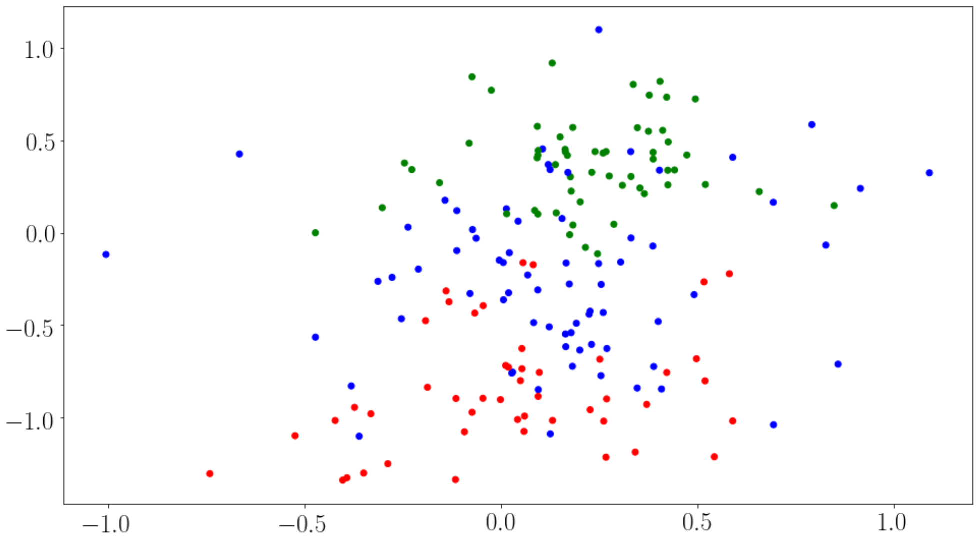

Figure 11: Figure showing a projection of the “Wine” dataset with a high hinge loss.

For the parameter tuning the following graphs were used for choosing the best fitting values in the stochastic gradient descent optimisation algorithm.

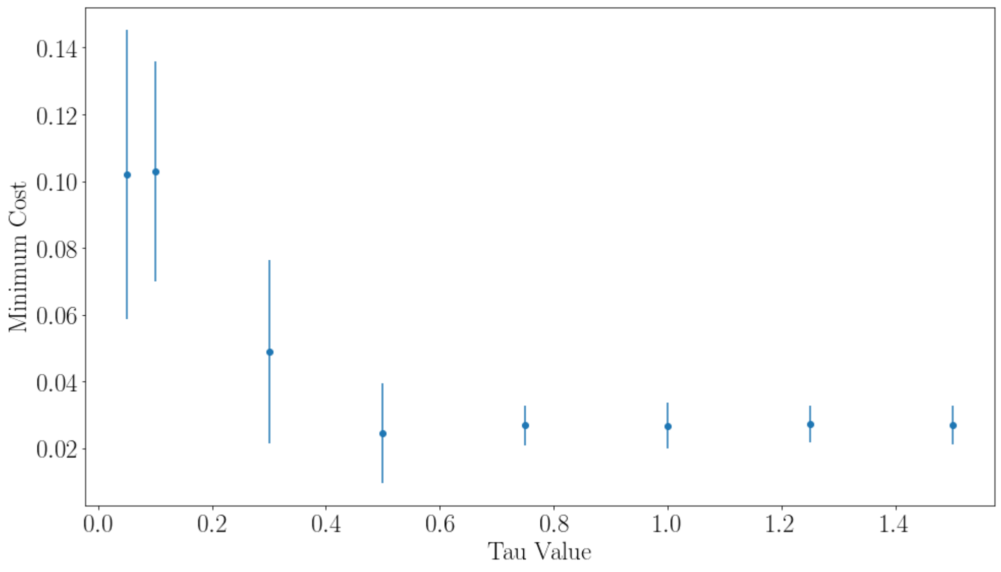

Figure 12: Figure showing the average minimum cost with error bars for choosing a τ parameter for the stochastic gradient descent. This was found on the “Wine” dataset.

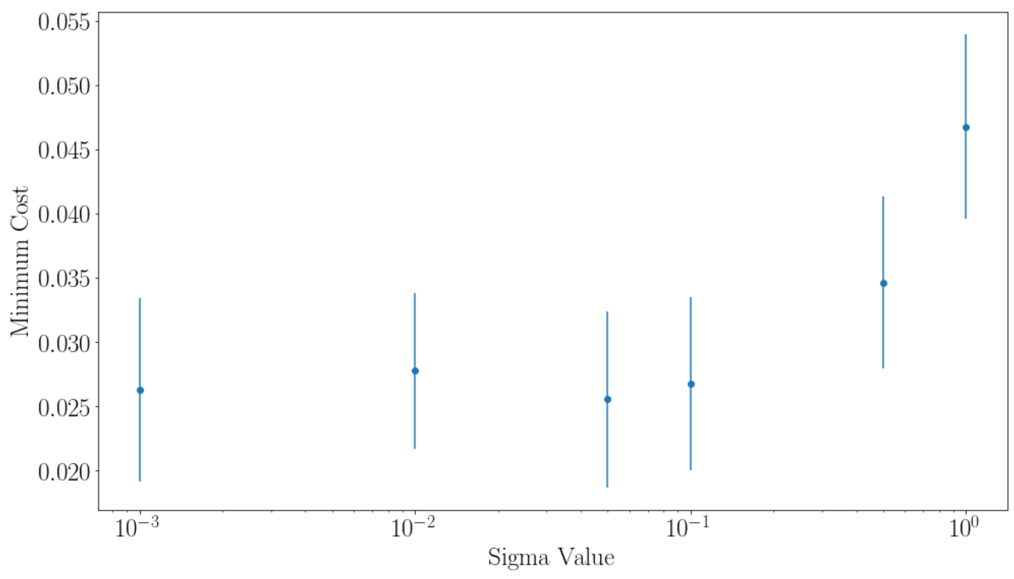

Figure 13: Figure showing the average minimum cost with error bars for choosing a σ parameter for the stochastic gradient descent. This was found on the “Wine” dataset.

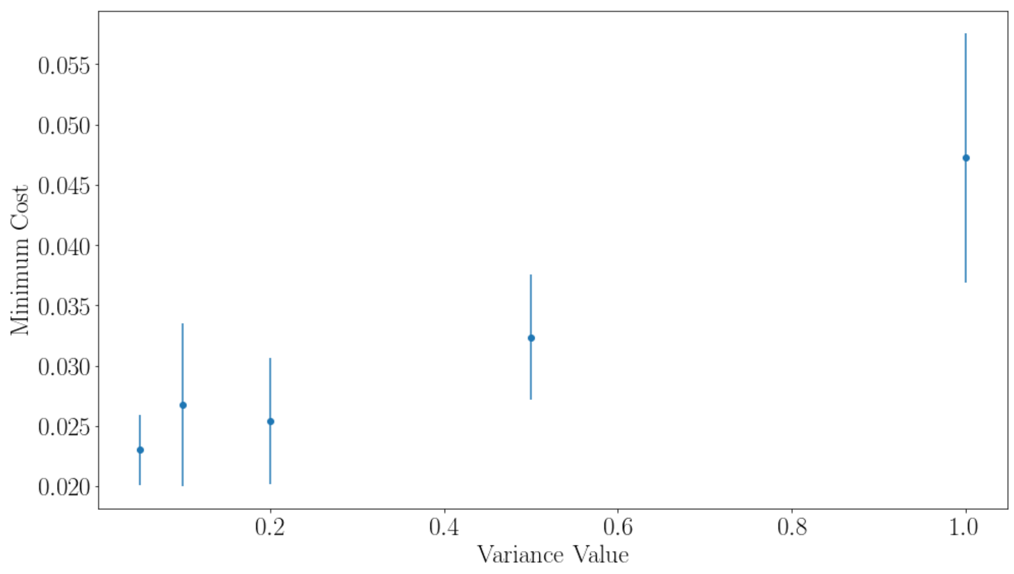

Figure 14: Figure showing the average minimum cost with error bars for choosing a variance for the stochastic gradient descent. This was found on the “Wine” dataset.

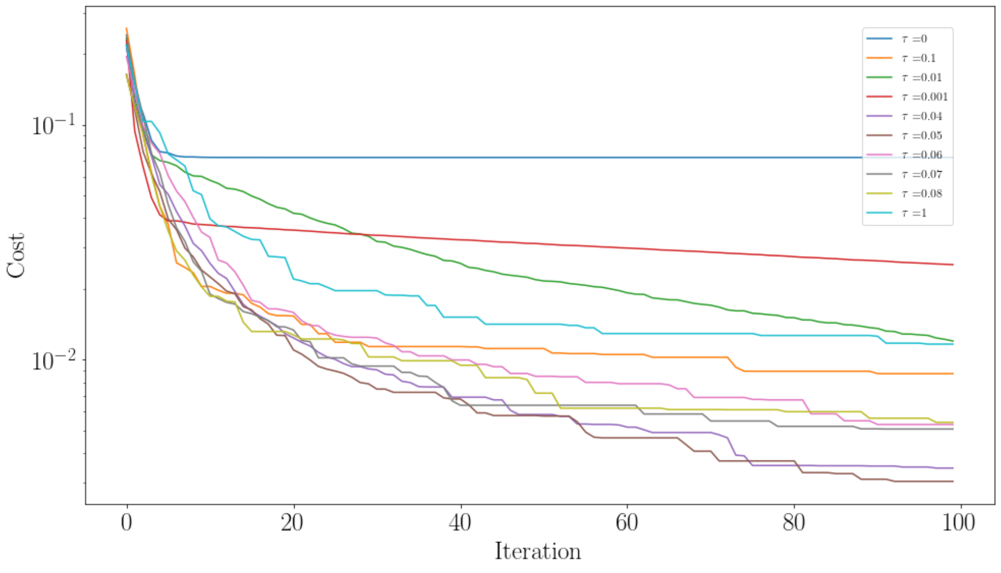

Figure 15: Figure showing the effect of altering the variance in the genetic optimisation algorithm. This was found on the "Wine" dataset.

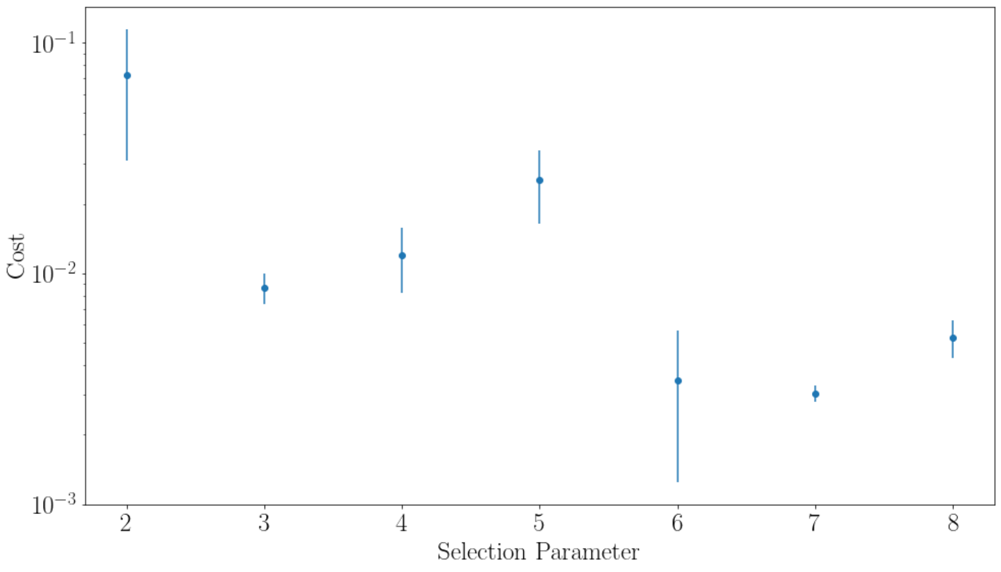

Figure 16: Figure showing the effect of altering the reciprocal of the fraction of population that survive and reproduce in the genetic algorithm. This was found on the "Wine" dataset.

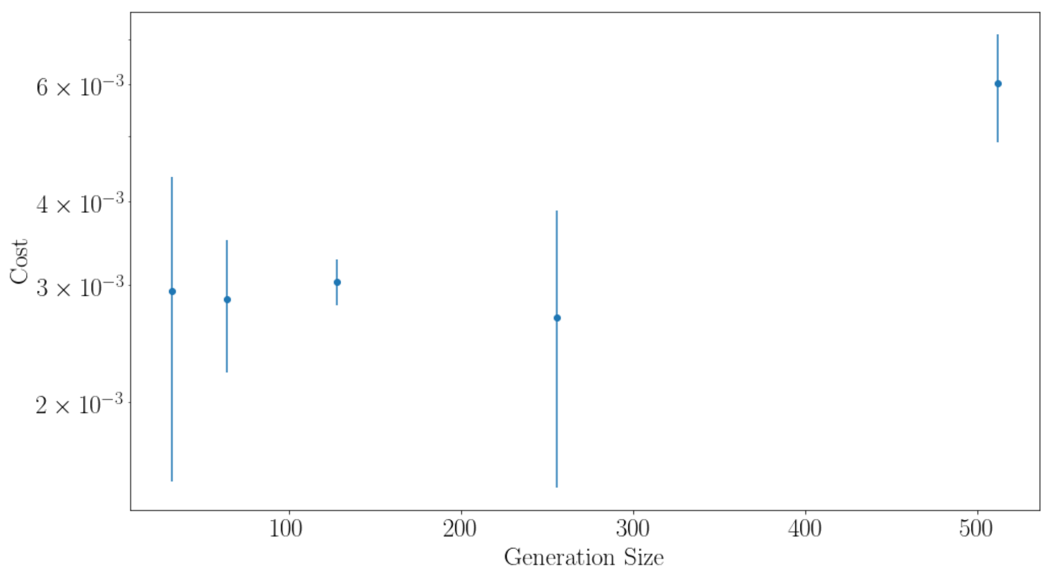

Figure 17: Figure showing the effect of altering the size of the intial population in the genetic algorithm. This was found on the "Wine" dataset.

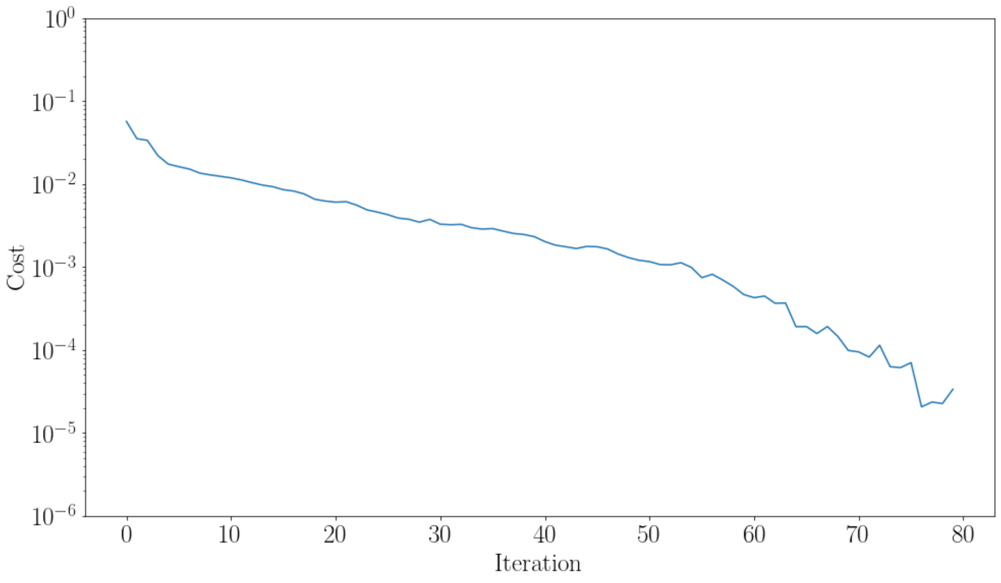

Figure 18: Figure showing the cost against each iteration from the PSO algorithm acting on the "Wine" dataset.

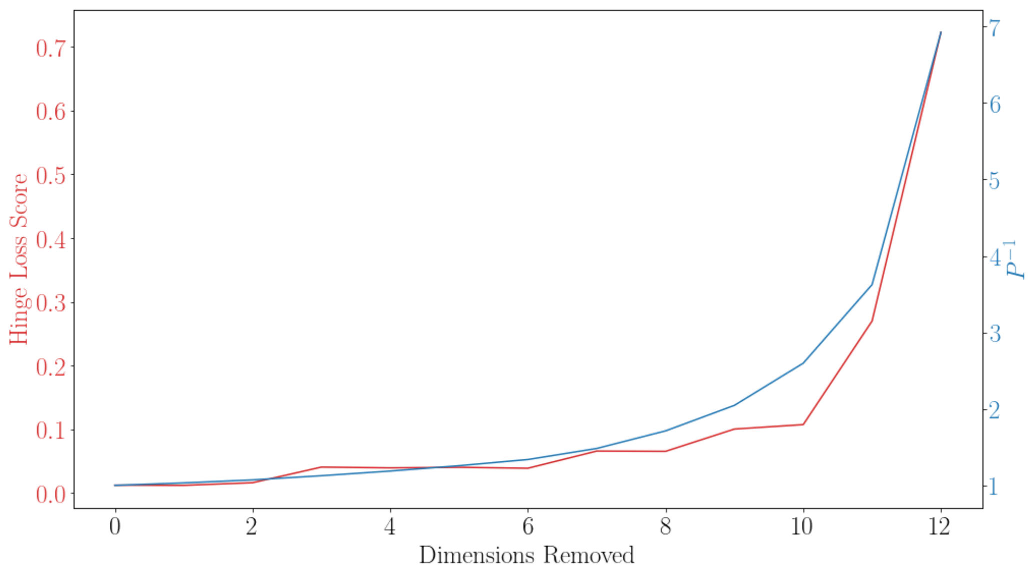

Figure 19: This figure shows the relationship between the value for hinge loss as the lowest contributing variables are removed, one at a time, from an optimal 2D projection of the "Wine" dataset. The red line corresponds to the hinge loss when x dimensions have been removed and the SVM is calculated on the $(n-x)$-dimensional space projection. The blue line represents the reciprocal of the sum of the magnitudes seen in the optimal projection in 2D.

B Code

The code for all the algorithms can be given on request.