Finding Features: Semi Supervised Learning

Main Project

Group 20

Project Mentor: Matthew Thorpe

Jack Hodgkinson (10727774), George Hudson (10615602), Houlin Wang (10749248) and Mark Francis Rogers (10452200)

School of Mathematics
University of Manchester
6th May 2022

Abstract

The process of finding features using semi-supervised learning is completed by the application of techniques from machine learning. Semi-supervised learning combines methods from unsupervised learning (the use of unlabelled data) and supervised learning (data which has labels). The aim of finding features in a data set is identifying characteristics that capture the important trends and patterns in the data. We will discuss machine learning and more specifically semi-supervised learning, how we clean data, some methods of dimensionality reduction and apply the techniques we have discussed to the data set we have been provided.

Contents

Introduction

We live in a world where the manipulation and interpretation of big data is in high demand but beyond human capabilities [1]. Due to this, techniques have been developed in order to help us study data by using computers to learn patterns and make statistical predictions. This is known more commonly as machine learning, which is a part of artificial intelligence. Machine Learning models have been able to achieve close-human performance in a wide domain of tasks recently (GPT-3) as well as beyond-human performance in several highly challenging, narrow tasks (AlphaGo). Because of the usefulness of it, many profess that artificial intelligence, and within that also machine learning, will have a transformative effect on our economies and how we live [2].

Our semi-supervised learning pipeline, when simplified, follows a four-step process. First, data cleaning. Second, dimensionality reduction. Third, support vector machines. Finally, interpretation of the results.

Our main aim in this project is to understand the concept of finding features and semi supervised learning, with an application process to find labels for the unlabelled data in our data set using techniques that are discussed throughout the paper.

1 Machine learning

1.1 What is machine learning?

Machine learning (often abbreviated to ML) is a branch of artificial intelligence and computer science that imitates the ways in which humans learn [3]. ML involves the use of statistical methods to train algorithms on big data sets to make classifications or predictions. This uncovers key insights within data mining projects, and can increase precision. Machine learning is a broad term which contains further sub-fields such as a deep learning, semi-supervised learning and neural networks which all follow a similar method but have varying levels of human intervention.

Fundamentally, machine learning works by a three-step method:

1. **The decision process**: an algorithm which provides an estimate model for patterns from the input data, which can be labelled or unlabelled.
2. **Error function**: this is applied after the decision process and it evaluates the accuracy of the estimate model. This can be done through comparison of known examples.
3. **Model optimisation process**: weights are adjusted to reduce the amount of discrepancy between the known example and the model estimate from the decision process. This process is repeated until a threshold of accuracy has been met [4].

1.2 Why is machine learning important?

The importance of machine learning often is not fully understood as it has taken many years for the subject to come into its own due to complications of technology. However developments in technology have meant that we are now able to easily fix issues such as the lack of processing power and the expense of data storage. As a result, machine learning can now produce better results than it could do previously. Now we are at a stage where applications in machine learning can provide value to many industries and be involved in the activity of businesses [1].

1.3 Semi-supervised learning

Semi-supervised learning is halfway between supervised and unsupervised learning. Human intervention provides the algorithms with supervision information which provides labels for the data. However, this may only be a small proportion of the data. Lets consider the data set $X = (x_i)_{i \in \mathbb{N}}$, which can be divided into two parts: the points $X_l := (x_1, ..., x_l)$, for which labels $Y_i := (y_1, ..., y_l)$ are provided and the points

$X_u := (x_{l+1}, ..., x_{l+u})$, the labels of which are not known [5]. The aim of a semi-supervised learning algorithm is to take advantage of the combination of labelled and unlabelled data and hopefully provide information about the unlabelled data. This is very useful in situations where labelled data is scarce or expensive [6].

2 Dimensions and dimensionality reduction

2.1 Dimensions in machine learning

Now that we have discussed what is and ML and its importance, we will examine the data that we feed into our ML algorithms and how the amount of dimensions in the data set will affect the performance of these algorithms. Dimensions in a data set are often referred to as features and the number of features refers to the number of attributes in the data.

Suppose that we have a data set that is organised into rows and columns. This way of organisation is seen frequently in data sets, and is commonly found on spreadsheets. If we interpret the data set as an n-dimensional space, n being the number of columns. The **columns** become **input variables/features** fed to a model which predict the target variable and we interpret the **rows** in the data sets as **points located in the space**.

2.1.1 Visualising the idea of dimensions

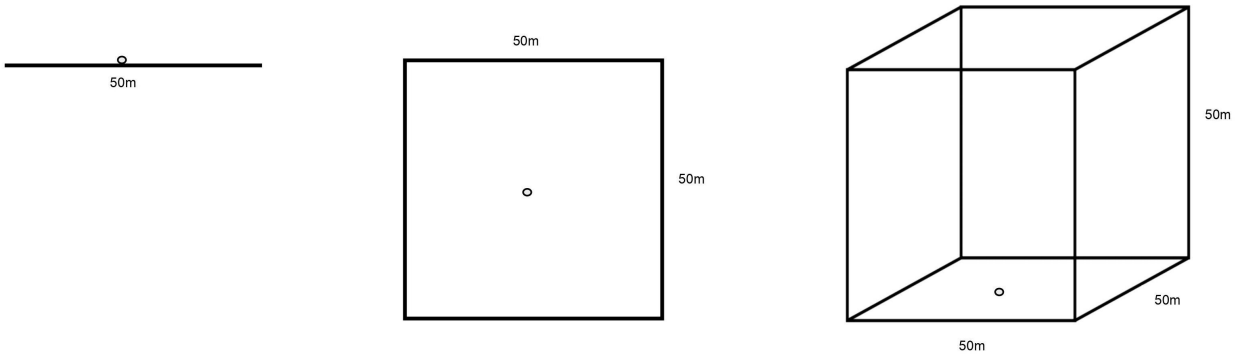

Figure 1: Visualising the idea of dimensions

Suppose you walk in a straight line for 50 metres, and somewhere along that line you drop a penny. If you restrict your search area along that 50 metres, you would find your penny quickly. If you decide to expand your search area to a $50m$ by $50m$ square, it would take a longer time for you to find that penny. Let us expand that search area again, now to a $50m$ by $50m$ by $50m$ cube. You might want to say goodbye to that penny!

In summary, the more dimensions that are in the data set, the more that the algorithm has to search to find the estimate model.

2.2 What is dimensionality reduction and why is it important?

Dimensionality reduction is simply the reduction of the amount of dimensions in the data set from a high-dimension to a lower dimension. As we have seen, having a data set with a high-dimension in turn means that the ML model will perform poor. We therefore carry out dimensionality reduction to reduce the chance of over-fitting and to ensure that the machine learning model performs better on the real data.

There are advantages to completing dimensionality reduction. The advantages include:

- Fewer dimensions means less complexity in the data set.

- A reduced data set will require less storage space.
- The accuracy of the model increases as the amount of anomalous data is decreased.
- Fewer data means that the algorithm will train faster.
- Reducing the features of the data set means that data visualisation can be completed faster.
- Noise and redundant features can be removed from the data set [7].

3 Data cleaning

3.1 Why do we perform data cleaning?

Data cleaning is crucial to improving the performance of machine learning algorithms and in our case semi-supervised methods. It follows the principle of a famous phrase: "garbage in, garbage out." Without accurate and clean data, the accuracy of our result decreases [8].

3.2 A few methods to clean data

1. Standardisation We need to transform our data into a standard form before we can proceed. Standardisation involves transforming the data so that it fits a normal distribution. This is important for types of Linear Discriminant Analysis, as an assumption of the LDA method of dimensionality reduction is that the data set follows a normal distribution. Among the various methods which is used to convert data to a normal distribution, typically the z-score method is used. This can be seen by re-scaling the variables using the formula

$$z = \frac{x - \mu}{\sigma}$$

where μ stands for the average of the variable, and σ denotes the standard deviation.

2. Normalisation This involves making the range of the data be between 0 and 1. This is crucial for the K Nearest Neighbours method to reduce features as we are finding the nearest distance between data points. If the scale for each column of data is different, then the weight for each column would be different as a result. For example, if we have a data set with 3 columns, column 1 has data from $0 - 10$, column 2 has data from $0 - 50$ and column 3 has data from $0 - 500$, then after applying KNN methods the result is more likely to be in column 3 or 2.

3. Outlier Extreme values (often called outliers or anomalies) in a data set can be toxic, resulting in peaks in our data therefore making the processing inaccurate. In order to remove outlying values from the general pattern of the data, we plot our data graphically and then remove any data that is anomalous.

4. Repetition of data and missing data These inaccuracies cause significant problems and make the data unreliable. We delete any repetitions of data and for missing data, we can either delete the incomplete data or fill in with data constructed using statistical techniques such as mean, median and mode [9].

4 Some methods of dimensionality reduction

4.1 Principal Component Analysis

4.1.1 What is Principal Component Analysis?

Principal Component Analysis (PCA) is one of the most commonly used dimensionality reduction techniques. But how can we retain the covariance structure [5] while reducing the dimension of data? PCA satisfies these two conditions (to an extent) by projecting the data to lower dimensions, principal components (PC). PCA has four majors steps: (1) normalising if the variables have different scales, (2) calculating the covariance matrix, (3) obtaining the eigenvalues and eigenvectors, and (4) making a feature vector. Because the labels themselves do not play a role in this method, it belongs to unsupervised learning.

If we have n random variables whom we project into an m-dimensional subspace, then

$$\epsilon_{PCA} = \sum_i \left\| x_i - \sum_{a=1}^{m} (x_i \cdot e_\alpha) e_\alpha \right\|^2 ,$$

where e_α are the orthonormal basis of the input space. From this it can be shown easily that the subspace with maximum variance is also the one with minimum reconstruction error [5].

4.1.2 The method of Principle Component Analysis

1. **Preprocessing** A preprocessing step involving standardisation and normalisation as discussed in **??** is necessary. Subtracting the mean of the variable μ is necessary, dividing by the standard deviation should be done to reduce the weight in the overall prediction of larger fluctuating variables, for instance in the case of anatomical measurements [10].

2. **Covariance Matrix** Let's assume that we have a data set with n variables, each of whom having k observations. This will be called our data matrix D:

$$D = \begin{pmatrix} d_{11} & \dots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{k1} & \dots & d_{kn} \end{pmatrix}.$$

Because we already have a centred positive matrix, to obtain the covariance matrix up to the constant k it is enough to multiply our data matrix D with its transpose:

$$D^T D.$$

3. **Eigenvectors and eigenvalues** The eigenvectors define directions in which the data is skewed, while the eigenvalues represent the extent to which they are represented in that direction and magnitude, respectively.

The matrix $D^T D$ is the covariance matrix of D up to a constant. We assume knowledge that A and cA, c being a constant have the same eigenvalues as A multiplied by c and the same eigenvectors as the actual covariance matrix of D. As a result of this, it can be decomposed into the set of eigenvectors and eigenvalues

$$X \lambda X^{-1},$$

where λ is a diagonal matrix with the entries corresponding to the eigenvalues and X is the matrix of eigenvectors.

4. **Feature selection** Finally, we have to select the relevant features. Here we are making a payoff between dimensionality and precision. Having a too high dimensionality will be sub-optimal for later, but discarding too many features can also be detrimental for performance. Sometimes, when we want to make a 2D plot the choice of dimensions must be 2, but at other times it is a more subtle matter. For the latter case, the proportion of variance explained is a vital tool. The proportion of variance explained by λ_1 is:

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \dots + \lambda_n}.$$

4.2 Linear Discriminant Analysis

4.2.1 What is Linear Discriminant Analysis?

Linear Discriminant Analysis (LDA) is not just a dimension reduction tool, but also a robust classification method [11]. Robert Fisher in 1936 was the first to use the method of discrimination to classify between different types of flowers [12]. The goal of the LDA method is to project a data set onto a lower-dimensional space and classify the data set into groups.

LDA is similar to the PCA method we have previously discussed, as both of them use the idea of matrix decomposition - more specifically eigenvalue decomposition. The method of LDA is applicable to many dimensions but to help us visually understand what is going on we will focus on a two dimensional case. Let us consider two samples plotted against each other say X_1 and X_2 which belong to corresponding classes say C_1 and C_2. We aim to find a line of the form $\vec{y} = \vec{w}^T\vec{x}$ which after projection successfully separates the samples into their corresponding classes. $\vec{w}$ is the weights unit vector onto which the data points are to be projected [13].

It may seem intuitive to say, why don't we just project everything down to the x-axis? But looking at the left hand image of 2, it is clear this results in an overlap of the classes. Alternatively a line is chosen based on maximisation of the projected means and the minimisation of the projected variances to hopefully separate the data into classes, details of which will be spoken about in 4.2.2. See the right hand image of 2 to help motivate this idea.

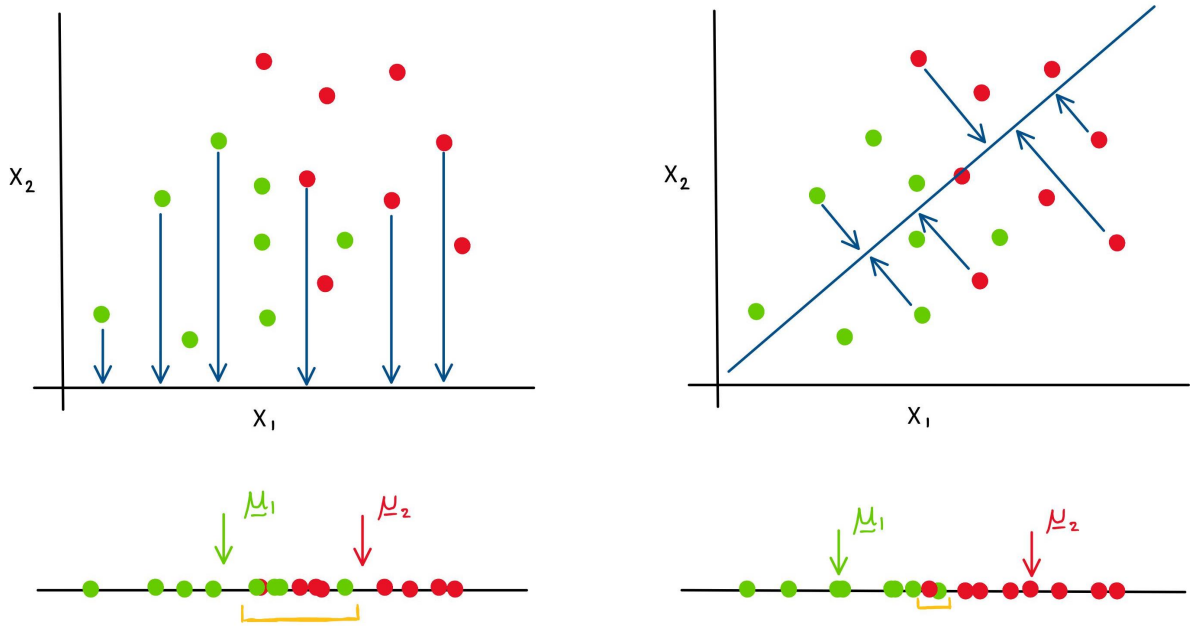

Figure 2: Explaining the separation of data classes using Linear Discriminant Analysis.

4.2.2 The method of Fisher LDA in 2D case

Fisher LDA makes no assumptions about the distribution of the classes or the equality of the class covariance [14]. Let us suppose we have two observations with the corresponding means $\vec{\mu_0}$, $\vec{\mu_1}$ and covariances Σ_0, Σ_1.

After projection, we can then calculate the corresponding mean $\vec{w}^T\vec{\mu_i}$ and variance $\vec{w}^T\Sigma_i\vec{w}$ for $i = 0, 1$. As discussed above, we want to find a line which maximises the projected means and minimizes the projected variances so we choose the distance between the projected means as our objective function.

$$\left(\vec{w}\cdot(\vec{\mu_0}-\vec{\mu_1})\right)^2 = \vec{w}^T(\vec{\mu_0}-\vec{\mu_1})(\vec{\mu_0}-\vec{\mu_1})^T\vec{w} = \vec{w}^T S_b\vec{w}.(*)$$

Here we define $S_b = (\vec{\mu_0} - \vec{\mu_1})(\vec{\mu_0} - \vec{\mu_1})^T$ as the between class scatter. Unfortunately the distance between the projected means is not a good enough measure to fully maximise the separation as seen in 2. Thus Fisher proposed a solution by which we also consider the maximum standard deviation between the classes, which is often referred to as scatter between projected samples.

$$\vec{w}^T\Sigma_0\vec{w} + \vec{w}^T\Sigma_1\vec{w} = \vec{w}^T(\Sigma_0 + \Sigma_1)\vec{w} = \vec{w}^T S_w\vec{w}.(**)$$

Where $S_w = (\Sigma_0 + \Sigma_1)$ is defined to be within class scatter.

Combining the $(*)$ and $(**)$, Fisher derived the maximisation of separation to be the linear combination of features $\vec{w}^T\vec{x}$ which maximises the criterion function defined to be:

$$\mathrm{J}(\vec{w}) = \frac{\left(\vec{w}\cdot(\vec{\mu_0} - \vec{\mu_1})\right)^2}{\vec{w}^T(\Sigma_0 + \Sigma_1)\vec{w}} = \frac{\vec{w}^T S_b\vec{w}}{\vec{w}^T S_w\vec{w}}$$

We now have a function expressed in terms of $\vec{w}$ and we know maximum separation occurs when the derivative of $J(\vec{w})$ with respect to $\vec{w}$ is equal to zero. This will give us an eigenvalue problem with eigenvector $\vec{w}^*$, namely the optimal weights vector.

$$\frac{d}{d\vec{w}}[J(\vec{w})] = \frac{d}{d\vec{w}}\left[\frac{\vec{w}^T S_b\vec{w}}{\vec{w}^T S_w\vec{w}}\right] = 0$$

$$\Rightarrow \left[\vec{w}^T S_w\vec{w}\right]\frac{d\left[\vec{w}^T S_b\vec{w}\right]}{d\vec{w}} - \left[\vec{w}^T S_b\vec{w}\right]\frac{d\left[\vec{w}^T S_w\vec{w}\right]}{d\vec{w}} = 0$$

$$\Rightarrow \left[\vec{w}^T S_w\vec{w}\right]2S_b\vec{w} - \left[\vec{w}^T S_b\vec{w}\right]2S_w\vec{w} = 0$$

We divide by $\vec{w}^T S_w\vec{w}$:

$$\left[\frac{\vec{w}^T S_w\vec{w}}{\vec{w}^T S_w\vec{w}}\right]S_b\vec{w} - \left[\frac{\vec{w}^T S_b\vec{w}}{\vec{w}^T S_w\vec{w}}\right]S_w\vec{w} = 0$$

$$\Rightarrow S_b\vec{w} - vS_w\vec{w} = 0$$

$$\Rightarrow S_w^{-1}S_b\vec{w} = v\vec{w}$$

So we get a generalised eigenvalue problem: $(\Sigma_0 + \Sigma_1)^{-1}(\vec{\mu_0} - \vec{\mu_1})(\vec{\mu_0} - \vec{\mu_1})^T\vec{w} = v\vec{w}$, with v as the eigenvalue, with $\vec{w}$ the eigenvector.

We obtain solution to the eigenvalue problem as $\vec{w}^* = S_w^{-1}(\vec{\mu_0} - \vec{\mu_1})$.

4.2.3 A numerical example of LDA

Here we can use an example to show how to calculate by hand:

If we have two sets of data:

$$X_1 = (x_1, x_2) = \{(4,1),(2,4),(2,3),(3,6),(4,4)\}$$
$$X_2 = (x_1, x_2) = \{(9,10),(6,8),(9,5),(8,7),(10,8)\}$$

The class statistics are:

$$\Sigma_0 = \begin{bmatrix} 0.80 & -0.40 \\ -0.40 & 2.60 \end{bmatrix}; \Sigma_1 = \begin{bmatrix} 1.84 & -0.04 \\ -0.04 & 2.64 \end{bmatrix}$$

$$\vec{\mu}_0 = \begin{bmatrix} 3.00 & 3.60 \end{bmatrix}^T, \vec{\mu}_1 = \begin{bmatrix} 8.40 & 7.60 \end{bmatrix}^T$$

The within and between-class scatter are:

$$S_b = \begin{bmatrix} 29.16 & 21.60 \\ 21.60 & 16.00 \end{bmatrix}; S_w = \begin{bmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{bmatrix}$$

The LDA projection is then obtained as the solution of the generalised eigenvalue problem:

$$S_w^{-1} S_b \vec{v} = \lambda \vec{v} \Rightarrow \left| S_w^{-1} S_b - \lambda \mathrm{I} \right| = 0 \Rightarrow \begin{vmatrix} 11.89 - \lambda & 8.81 \\ 5.08 & 3.76 - \lambda \end{vmatrix} = 0 \Rightarrow \lambda = 15.65$$

$$\begin{bmatrix} 11.89 & 8.81 \\ 5.08 & 3.76 \end{bmatrix} \begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \end{bmatrix} = 15.65 \begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \end{bmatrix} \Rightarrow \begin{bmatrix} \mathrm{v}_1 \\ \mathrm{v}_2 \end{bmatrix} = \begin{bmatrix} 0.91 \\ 0.39 \end{bmatrix}$$

Finally we have:

$$\vec{w}^* = S_w^{-1} \left(\vec{\mu_0} - \vec{\mu_1} \right) = \begin{bmatrix} -0.91 & -0.39 \end{bmatrix}^\top$$

Thus, we have found the optimal projection.

5 Support Vector Machines

5.1 What is a support vector machine?

A support vector machine (often abbreviated to SVM) is a supervised learning algorithm that learns by example to assign labels to objects [15]. An SVM classifies data by finding the best hyperplane that separates all data points of one class from the other. The optimal hyperplane for an SVM is the one with the largest margin between the two classes. Margin is defined to be the maximal width of the slab parallel to the hyperplane that has no interior data points [16]. An illustration of this can be seen below in 3:

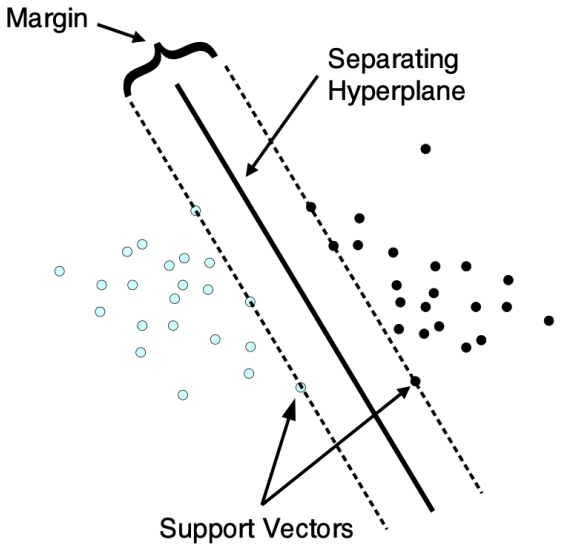

Figure 3: A visual representation of support vector machines, from [16].

5.2 The steps of the support vector machine algorithm

1. **Define the hyperplane** We have discusses that the purpose of SVMs is to separate data points into different groups. A way to do this mathematically is to create a hyperplane.

We will define the hyperplane H such that:

$$\vec{W}^T \vec{x} + b = 0$$

where $\vec{W}$ is a weight vector, $\vec{x}$ is an input vector and b is the bias.

2. **Finding the optimal hyperplane** After defining the hyperplane H, we want to find the optimal hyperplane, which is done by finding a plane that has the maximum margin, i.e the maximum distance between data points of both classes [17]. To maximise this margin we consider the loss function:

$$min\lambda \left\|W\right\|^2 + \sum_{i=1}^{n}(1 - y_i \left\langle x_i, W\right\rangle)_+$$

We now take partial derivatives with respect to the weights to obtain their gradients:

$$\frac{\partial}{\partial W_k}\lambda \left\|W_k\right\|^2 = 2\lambda W_k,$$

$$\frac{\partial}{\partial W_k}(1 - y_i \left\langle x_i, W\right\rangle)_+ = \begin{cases} 0 & if y_i \left\langle x_i, W\right\rangle \geq 1 \\ -y_i x_{ik} & else \end{cases}$$

Thus we have two cases for the gradient update: the first is when we have no misclassification thus we update the gradient from the regularisation parameter, $W = W - \alpha \cdot (2\lambda W)$. Otherwise we have a misclassification and so we include the loss along with the regularisation parameter to perform gradient update, $W = W + \alpha \cdot (y_i \cdot x_i - 2\lambda W)$ [17].

6 Applying the methods discussed to our given data set

6.1 Information about the data set

We have been provided with a data set of blood samples containing the measurements from a number of tests, of which a subset are labelled either healthy or sick. We will apply techniques that have been discussed throughout the project on the data set to try and identify which of the unlabelled patients are healthy and which are sick.

We are provided with 5000 samples, and each sample contains 100 features. There are 5000 labels given, some which are labelled and the majority are unlabelled. Each label is either 0, 1 or NaN. We are not told which of 0 and 1 is healthy and which is sick. The aim of our code is to distinguish between each.

6.2 How much can we trust the data?

By performing data cleaning methods as discussed in 3.2, we are increasing the reliability of the data set. With any given data set, you cannot be certain that all the information provided is accurate. This is why before we run our data through ML algorithms, we must clean it. We will apply the four methods of data cleaning discussed to our data as seen below [18], [19]:

[1]:
```
# Import data
import pandas as pd
import numpy as np
# %matplotlib inline
import matplotlib.pyplot as plt
```

[2]:
```
# Load the features
data = pd.read_csv('/Users/jack/Documents/Group Project/Main Project/finding_features/
 ↪features10.csv')
# Load the labels
labels = pd.read_csv('/Users/jack/Documents/Group Project/Main Project/
 ↪finding_features/knownlabels10.csv')
```

[4]:
```
#Drop Repeating Data
data = data.drop_duplicates()
```

[5]:
```
#Standardise all data to follow the standard normal distribution
data = (data-data.mean())/(data.std())
```

6.3 Using a method of dimensionality reduction on the data

Now that we have data without noise, we will apply a method of dimensionality reduction. Because the labels for a large chunk of the data set are missing, having an unsupervised dimensionality reduction technique is desirable, where in the second step one performs a further classification technique on the data set with reduced dimensionality. PCA satisfies these conditions as it is an unsupervised dimensionality reduction technique, while LDA is a supervised method. For this reason, we have decided to use the PCA method of dimensionality reduction on our data set.

Afterwards, we will look at the results by using the explained variance, the ratio of the variance of each principal component over the total variance.

[6]:
```
# Normalisation can be very important as PCA is scale-sensitive.
# Now we will perform the necessary preprocessing step.
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA

#Apply the PCA model
pca = PCA(n_components=9)
pipe = Pipeline([('scaler', StandardScaler()), ('pca', pca)])
data_t = pipe.fit_transform(data)
```

Having run the model several times with different principal components, we decided that 9 is the optimal number.

[7]:
```
#See explained variance
print(pca.explained_variance_)
```

```
[1.25101974 1.24394059 1.22739924 1.21844824 1.20020646 1.1920235
 1.18386342 1.17025869 1.16554249]
```

[8]:
```
#See transformed data dimensions
data_t.shape
```

[8]:
```
(4999, 9)
```

We can see that we managed to reduce the 5000 times 100 data set to a 5000 times 9 data set.

6.4 The SVM algorithm on the data

Now that we have applied PCA to the data, we will run an SVM to help label the remaining unlabelled data points. Because the observations have to be classified as either 0 or 1 (binary classification), our data set is well-suited for the hyperplane drawn by the SVM method.

[9]:
```
#Use the principal components for the SVM
data_t = pd.DataFrame(data_t) #transform array into data set
data_t["Outcome"] = labels
#Drops those rows where the Outcome is not NaN
test_data_t = data_t.dropna()
```

We will use our 500 correctly labelled observations to train and test the model. 90% will be used for training and the residual 10% for testing.

[10]:
```
#Separate the labelled data to training () and testing set.
x_train = test_data_t.iloc[0:450, 1:9]
y_train = test_data_t.iloc[0:450, 9]
x_test = test_data_t.iloc[450:500, 1:9]
y_test = test_data_t.iloc[450:500, 9]
```

[11]:
```
#Import the SVM package and the accuracy score
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

Kernel functions are symmetric functions that serve are a method to transform input into processing data [5]. The easiest way to find the right kernel is to try out several different kernel functions and test their performance empirically (for instance, looking at the accuracy rate). We have found that the linear kernel is well-suited for the task.

[12]:
```
#Apply a linear kernel
clf = SVC(kernel='linear')
clf.fit(x_train,y_train.values.ravel())
y_pred = clf.predict(x_test)
print(f"Accuracy score: {accuracy_score(y_test,y_pred)}")
```

```
Accuracy score: 0.62
```

We define the accuracy score as the fraction of correct predictions / all predictions. Our accuracy score of 0.62 is acceptable.

Now we will apply the model only on the NaN data. We will hence predict the labels for the unlabelled data set (NaN), which can be then used for another model.

[13]:
```
#Now use only the unlabelled data set
#The next line drops those rows where the Outcome is not NaN
NaNData = data_t[data_t['Outcome'].isnull()]
```

[14]:
```
x_test = NaNData.iloc[:, :8]
y_pred_unlabelled = clf.predict(x_test)
```

[15]:
```
#Set the outcome to the one predicted
NaNData["Outcome"] = y_pred_unlabelled
```

[16]:
```
NaNData
```

[16]:

	0	1	2	3	4	5	6	7	8	Outcome
0	-0.075880	-1.463210	-0.716981	-0.229734	0.773517	2.023123	0.888324			
1	2.189416	0.594882	-0.214223	-3.078477	-0.594746	-0.436441	2.243179			
2	0.282966	-0.804971	-0.281567	1.811123	-0.079522	0.130032	-1.389806			
3	0.330520	-0.241726	-0.733050	1.583563	-0.479666	-0.359959	-1.735051			
4	1.120919	2.028620	-1.472610	0.849413	1.416831	-0.281703	-0.468698			
...	...	...	...	...	...	...	...			
4988	-1.129619	-1.843601	0.789952	-0.582246	-0.373582	0.074332	0.334915			
4989	-0.745463	2.051640	-0.551076	-1.477879	0.338358	-0.032159	-0.579395			
4994	1.070011	-0.522073	-0.233809	-0.736957	1.736542	1.952166	0.402093			
4995	-1.381192	-0.347502	-1.644984	0.440618	-1.526446	-1.704203	0.943328			
4998	-0.871681	-1.412968	-1.736183	-1.722709	-0.434000	0.344940	0.189543			

```
0     0.252478  1.601064      0.0
1     1.717278 -0.381131      0.0
2     0.526583  0.593338      1.0
3     1.075430  0.259641      1.0
4     1.384900 -0.067231      1.0
...        ...       ...      ...
4988 -0.776655  1.090754      0.0
4989 -1.295597  1.018476      0.0
4994  0.041264  0.855725      0.0
4995 -0.664638 -0.663427      1.0
4998  0.276469 -0.316071      0.0

[4499 rows x 10 columns]
```

The NaNData dataframe above contains the 4500 unlabeled observations with the outcome predicted with the previously trained SVM. This is also called pseudo-labelled data [5].

Conclusion

In conclusion, we have been successful in finding features for our data-set using semi supervised learning techniques.

We began by providing a brief introduction to machine learning, touching upon how ML algorithms worked, then focused our attention to semi-supervised algorithms and the process of finding features in a set of labeled and unlabeled data.

We continue building our understanding of the topic by exploring the idea of dimensions in the data set and data cleaning. With the help of the coin example we hope it provided a simple explanation that too many dimensions can be detrimental to performance and reducing the dimensions simplifies the problem in many ways.

After providing some motivation to why we are doing all of this, we started to explore two common methods of dimensionality reduction. The first was PCA which allowed us to easily reduce the dimensions of the feature space by projecting the data to a lower dimension and giving us the principle components. Secondly, we explored the method of LDA which found a hyperplane of lower dimension to the feature space which separated the data into distinct classes after it was projected onto this plane. We found that the hyperplane was chosen based on maximisation of the projected mean and minimisation of the projected variance.

We then explained support vector machines which further separate our data into different groups using a hyperplane. The optimal hyperplane came at no surprise to be the one that separated the different groups the most and this was found by maximisation of the loss function.

However, support vector machines can have several limitations. They do not perform very well on large data sets meaning they would not work well on big data. Also, their performance in imbalanced data sets is relatively poor, as the hyperplanes can be biased to the minority class. Choosing the right kernel can be very difficult [20].

Finally, we wanted to apply everything we have learnt to the data which was provided to see what we could find. We successfully applied the method of PCA to the data to conclude that there were 9 important features. PCA can be problematic when one wants to preserve the interpretability of the features, or when the number of features is significantly greater than the number of observations. In the case of our data and goal (50 times more observations than features, binary classification), neither of the drawbacks was of concern. We then applied a SVM to predict the missing labels, and acquired an accuracy score of 0.62 on the test data set.

The acceptability of the accuracy score depends on the specific confidence interval expected. If, for instance, the hospital expects a 70% confidence interval, the expected accuracy for the pseudolabels (assuming it will fluctuate around 0.62) combined with the 500 correctly labeled observations yields an average accuracy of 0.658. This might be sufficient in itself, or could be used for training another model to improve the final accuracy score.

References

[1] *Why is Machine Learning So Important?* July 2021. URL: https://csuglobal.edu/blog/why-is-machine-learning-so-important (visited on 22nd Apr. 2022).

[2] Stuart J. (Stuart Jonathan) Russell. *Human compatible : artificial intelligence and the problem of control*. 2020, p. 336. ISBN: 0525558632.

[3] Issam El Naqa and Martin J. Murphy. *What Is Machine Learning?* Springer International Publishing, 2015, pp. 3–11. DOI: 10.1007/978-3-319-18305-3_1.

[4] *What is Machine Learning?* July 2020. URL: https://www.ibm.com/cloud/learn/machine-learning (visited on 22nd Apr. 2022).

[5] Olivier Chapelle, Bernhard Schölkopf and Ale Zien. *Semi-Supervised Learning*. Ed. by Oliver Chapelle, Bernhard Schölkopf and Ale Zien. The MIT Press, 2006, pp. 1–12.

[6] Andrew Goldberg and Xiaojin Zhu. "Introduction to Semi-Supervised Learning". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* (2009), pp. 9–21.

[7] *What is Dimensionality Reduction? Overview, and Popular Techniques*. June 2021. URL: https://www.simplilearn.com/what-is-dimensionality-reduction-article#why_dimensionality_reduction_is_important (visited on 3rd May 2022).

[8] Jerry R. Marlatt Daniel T. Brooks Brandon Becker. *Computer Applications in Particular Industries: Securities*. Third. Computers The Law, American Bar Association, Section of Science and Technology, 1981, pp. 250–253.

[9] Omar Elgabry. *The Ultimate Guide to Data Cleaning* |. Feb. 2021. URL: https://towardsdatascience.com/the-ultimate-guide-to-data-cleaning-3969843991d4 (visited on 3rd May 2022).

[10] I.T. Jolliffe. *Principal Component Analysis*. Ed. by Ian Jolliffe. Second. Springer US, 2002. ISBN: 978-0-387-95442-4. DOI: https://doi.org/10.1007/b98835. URL: https://link.springer.com/book/10.1007/b98835#about.

[11] Xiaozhou Yang. *Linear Discriminant Analysis, Explained*. May 2020. URL: https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b (visited on 3rd May 2022).

[12] Petros Xanthopoulos, Panos M. Pardalos and Theodore B. Trafalis. *Robust Data Mining*. Springer New York, 2013. ISBN: 978-1-4419-9877-4. DOI: 10.1007/978-1-4419-9878-1.

[13] Sunil Kumar Dash. *A Brief Introduction to Linear Discriminant Analysis*. Aug. 2021. URL: https://www.analyticsvidhya.com/blog/2021/08/a-brief-introduction-to-linear-discriminant-analysis/ (visited on 5th May 2022).

[14] Ronald Fisher. "The Use of Multiple Measurements in Toxonomic Problems". In: *Annals of Eugenics* 7 (2 Sept. 1936), pp. 179–188. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[15] William S Noble. *What is a support vector machine?* 2006. URL: http://www.nature.com/naturebiotechnology.

[16] David Meyer. *Support Vector Machines*. FH Technikum Wien, Aug. 2015. URL: https://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf.

[17] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. June 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (visited on 5th May 2022).

[18] *Principal Component Analysis for Visualization*. May 2020. URL: https://machinelearningmastery.com/principal-component-analysis-for-visualization/ (visited on 5th May 2022).

[19] Rohith Gandhi. *Support Vector Machine — Introduction to Machine Learning Algorithms*. June 2018. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 (visited on 5th May 2022).

[20] Wei Hao Khoong. *When Do Support Vector Machines Fail?* Aug. 2021. URL: https://towardsdatascience.com/when-do-support-vector-machines-fail-3f23295ebef2 (visited on 5th May 2022).