Classification on Fatty cattle data

Jing Deng

September 15, 2011

Abstract

This report first introduced the techniques used for data analysis. Then the results will be given in two main sections. First, the support vector machine (SVM) and nonlinear Fisher discriminant analysis (NFDA) will be employed to build the classifier on the original data. The aim is to construct an classifier which can be used to prediction the fatty liver by using measurable protein levels. Second, as the original data is highly imbalanced, the Synthetic minority over-sampling technique (SMOTE) is used to randomly generate some negative samples (Normal cattle without fatty liver). The same process with then implemented on the new data set. A brief conclusion of the experimental results and suggested future work is given in the last section.

Keywords: Nonlinear Fisher discriminant analysis, support vector machine, Two-stage selection, leave-one-out cross validation, Synthetic minority over-sampling technique

1 Introduction

Classification is the problem of identifying the sub-population to which a new observation belongs. As a universal technique, classification has been widely applied in statistical analysis, pattern recognition and machine learning [1]. Most techniques are originally proposed for two-class classification, while their generalized variants may be used for multi-class cases [2]. Nevertheless, two-class classification still covers a wide range of practical applications.

An important issue with two-class classification in practice is the imbalanced problem which means the instances in one class outnumber the instances of the other classes. Such an imbalance in the data represents the so-called between-class imbalance. Actually, imbalanced problems widely exist in the fields of medical diagnosis, science and engineering, and some examples includes surveillance of nosocomial infection, cardiac care and elucidating protein-protein interactions as well as fraud detection, network intrusion detection and telecommunication management. Normally, the instances in the majority class are referred to as negative, while in its counterpart, the minority class, the instances are referred to as positive. The most convenient and effective way to deal with imbalanced learning problems is the re-sampling approach which only adjust the original training data set instead of modifying the learning algorithm. It has been shown that balanced data sets provides better classification performance than imbalanced ones [3]. Among the most studied re-sampling approaches, the synthetic minority over-sampling

technique (SMOTE) [4] is a well acknowledged over-sampling method, in which the minority class is over-sampled by creating synthetic instances in the feature space formed by other minority instances and their K-nearest neighbours.

To assess the generalization ability, the resultant classifier is usually applied to a set of test data. This involves splitting all the available measured data into different sets. However, the amount of data available is often limited, so it is desirable to use all the data available to train the model without sacrificing any generalization performance. A typical way of doing this is to employ cross validation. The limited data set is split into s parts in which $s-1$ parts are used for model training, and the single remaining part is used for model testing. This procedure is continued until all s different possible combinations has been implemented. The extreme case of cross validation is known as the Leave-One-Out (LOO) method [5, 6], where only one sample is used for testing and the rest left for training. The overall model error or LOO error is then the average test error of each data sample. Though this approach can achieve improved model generalization, its computational complexity is extremely high. Generally, it is therefore feasible only for very small data set, because the computational complexity is proportional to the total number of data samples. However, if the model has a Linear-In-The-Parameters structure, it has been shown that LOO error can be calculated without splitting the data set explicitly [6]. In this work, the data set is small, so LOO cross validation is used directly.

With the training data and test data sets been pre-processed, the classifier is then developed based on the object's characteristics (also known as features). Early work was mainly concentrated on a linear classifier, non-linear solutions are the major interest nowadays. More specifically, a linear classifier is defined as one where the classification decision is made based on the value of a linear combination of the characteristics. The best known method is linear discriminant analysis (LDA), which assumes that the collected data samples are normally distributed in each class. Fisher's linear discriminant [7] is derived from LDA.

Clearly, a linear discriminant may not be complex enough for real world applications. More sophisticated methods are available for non-linear problems, such as neural networks [8]. However, by using the kernel approach a linear discriminant can still be employed for non-linear classification [9].

The kernel idea was originally proposed for support vector machines (SVM) [10, 11] and non-linear principal component analysis (NPCA) [12]. It maps the input data into a high (or even infinite) dimensional feature space where the original problem becomes linear. The so-called *kernel trick* is involved here as the non-linear mapping function doesn't need to be known explicitly. The algorithm only uses dot products (also known as kernels) of the mapped data, instead of their explicit position in the feature space. Figure 1 illustrates the non-linear projection. Possible choices for kernel are the Gaussian radial basis function (RBF) or a polynomial function [9].

Fisher's linear discriminant can also be easily generalized to non-linear classification using such a kernel projection [2], leading to the more flexible kernel Fisher discriminant (KFD) [9]. The main issue that affects practical implementation is the computational complexity which scales with the number of training samples. Thus, KFD is not recommended for use on large data sets.

One solution is to transform KFD into a least-squares problem [13], and then to adopt a forward selection algorithm, such as orthogonal least squares (OLS) [14, 15] or the fast recursive algorithm (FRA) [16], to produce a compact classifier [13, 17]. Though

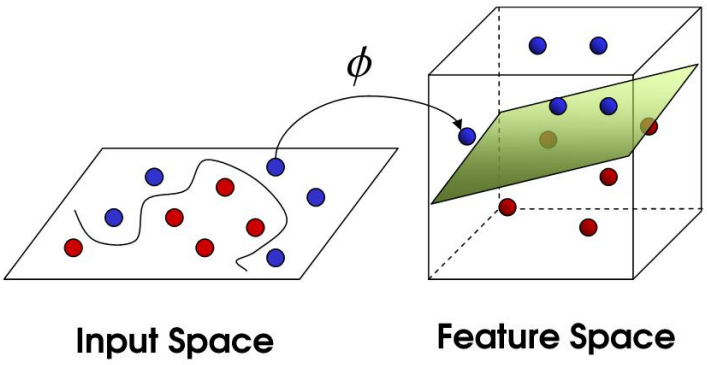

Figure 1: Non-linear mapping in kernel approach

a forward construction scheme can efficiently build a sparse model from a large candidate term pool, the final model is not optimal [18], since the calculation of the contribution of any new term depends on previously selected ones [19]. To reduce this constraint, genetic search has been suggested to refine the model structure [20], but at the expense of a very high computational complexity. In [19], a two-stage stepwise construction method was recently proposed which combined both forward and backward construction procedure. This retains the computational efficiency of the FRA while improving the model compactness.

In the above two-stage construction algorithm, an initial model is first constructed by the FRA where the contribution of a particular term of interest is measured by its reduction in an appropriate cost function. The significance of each selected term is then reviewed at a second stage of model refinement, and insignificant terms are replaced. Specifically, if the contribution of a previously selected term is less than any from the candidate pool, it will be replaced with this. Thus, the cost function can be further reduced without increasing the model size. This checking cycle is iterated until no insignificant model term exists in the trained model, resulting in an optimized model structure with improved performance.

Another widely used technique is the well-known support vector machine (SVM)[21, 22] which has been utilized widely and achieved a strong track record in various biological applications, ranging from the gene classification of microarray data [23, 24], tumor classification [25] to the identification of hormone misuses in cattle [26]. Nevertheless, an unfavourable fact with the SVM algorithm is its expensive computational cost. Training a SVM involves a quadratic programming subject to linear inequality constraints. The resultant classifier also suffers from the curse of dimension. It usually requires more support vectors than Non-linear Fisher discriminant analysis to achieve satisfactory accuracy. The experimental result in the following sections will confirm this.

2 Preliminaries

2.1 Support Vector Machine

The support vector machine tries to find a separating hyperplane in this space with which the training data are correctly arranged into their intended class and the distance between

boundaries of the two classes, which is known as the margin, is maximized. However, in practice, most real-life problems are not trivial so that the separating hyperplane cannot be located. This problem is solved by mapping the data to a space of higher dimensions where they become separable. The new space is termed as feature space, as opposed to the input space which contains the training data. However, one obvious drawback is that the data mapping from the input space to the feature space imposes more computational cost. Nonetheless, in the case that the feature space is infinite in dimensions, great difficulties arise in implementing the data transformation. Fortunately, in SVM algorithms, a favourable fact is that its optimal hyperplane requires only the dot product between training data in the feature space. By introduction of the kernel technique which performs the dot products between vectors of feature space, the search for the optimal hyperplane can be formulated even without explicit knowledge of the feature space. This feature of kernel function spares SVMs the computation of the exact representation of the training data in the feature space. For ill-posed classification problems, it is not practical to have a hyperplane of the optimal margin with which each training sample falls into the supposed group. SVMs then settle for a separating hyperplane which allows classification errors on training errors. It then require the solution of the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\Phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{1}$$

where the sample $\mathbf{x}_i$ are mapped into a higher dimensional space by the kernel function Φ. The parameter C is to be chosen to impose penalties on errors, which actually acts as the trade-off between the training errors and the margin. Therefore, the establishment of a support vector machine is started with proper settings of the penalty constant and also parameter(s) of the chosen kernel function, which depend heavily upon the specific training data set. Experiments in this report all opted for Gaussian radial basis function (RBF) which is the most widely used one and takes the form of

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\sigma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2) \tag{2}$$

where σ is a constant and normally pre-determined. In this report, the parameter σ as well as the penalty constant, denoted as C, for an SVM is tuned by implementing grid-search and is set as the values which produce the best leave-one-out cross validation (LOOCV) accuracy. The SVM were implemented by LIBSVM [27].

2.2 Nonlinear Fisher discriminant analysis

Nonlinear Fisher Discriminant (NFD) analysis is a generalization of linear Fisher discriminant analysis produced by adapting the kernel method. Specifically, the data samples are first mapped into some high dimensional feature space F using a nonlinear function ϕ, with linear discriminant analysis subsequently performed in this feature space. The advantage of the kernel approach is that the mapping function does not need to be known exactly. Only the dot product of the mapped data is involved in solving the problem, and this can be represented by a suitable kernel function (e.g. a Gaussian kernel). Thus

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}} \cdot \phi(\mathbf{x}_j) \tag{3}$$

Normally, a nonlinear discriminant function $\mathbf{w} \in F$ can be obtained by maximising the following function

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B^\phi \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W^\phi \mathbf{w}} \tag{4}$$

where

$$\mathbf{S}_B^\phi = (\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)(\mathbf{m}_1^\phi - \mathbf{m}_2^\phi)^T \tag{5}$$

and

$$\mathbf{S}_W^\phi = \sum_{i=1,2} \sum_{\mathbf{x} \in \mathbf{c}_i} \left(\phi(\mathbf{x}) - \mathbf{m}_i^\phi\right) \left(\phi(\mathbf{x}) - \mathbf{m}_i^\phi\right)^T \tag{6}$$

where $\mathbf{m}_i^\phi$ is the mean value of the samples in each class. Following [9], it can be shown that the discriminant function $\mathbf{w}$ is given by

$$\mathbf{w} = (\mathbf{S}_W^\phi)^{-1}(\mathbf{m}_1^\phi - \mathbf{m}_2^\phi) \tag{7}$$

and the projection of a new sample $\mathbf{x}_i$ onto $\mathbf{w}$ is given by

$$\hat{y}_i = \phi(\mathbf{x}_i)^T \mathbf{w} \tag{8}$$

According to [17], the nonlinear discriminant function obtained using a minimum squared-error cost function has the same direction as the Fisher discriminant solution. Specifically, suppose that a set of N data samples belongs to two categories. The first N_1 samples are collected from class 1 with label value y_1, and the remaining N_2 samples are from class 2 with label value y_2. By letting the output $y_1 = N/N_1$ and $y_2 = N/N_2$ (N_i is the number of samples belonging to class i), it can be shown that the NFD is directly related to least-squares problems.

Assuming N data samples are available for training, equation (8) can be written as:

$$\begin{pmatrix} 1 & \boldsymbol{\phi}(\mathbf{x}_1)^T \\ \vdots & \vdots \\ 1 & \boldsymbol{\phi}(\mathbf{x}_{N_1})^T \\ 1 & \boldsymbol{\phi}(\mathbf{x}_{N_1+1})^T \\ \vdots & \vdots \\ 1 & \boldsymbol{\phi}(\mathbf{x}_N)^T \end{pmatrix} \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} + \begin{pmatrix} e_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ e_N \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_1 \\ y_2 \\ \vdots \\ y_2 \end{pmatrix} \tag{9}$$

where w_0 is the threshold, and $e_i, i = 1 \ldots N$ are the model errors. Equation (9) can be re-written in matrix form as:

$$\mathbf{X}\mathbf{w} + \mathbf{e} = \mathbf{y} \tag{10}$$

Here, w_0 is already included in $\mathbf{w}$. The least-squares method solves this equation by minimizing the cost function

$$J(\hat{\mathbf{w}}) = \| \mathbf{e} \|^2 = ||\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}||^2 \tag{11}$$

leading to the solution

$$\mathbf{X}^T\mathbf{X}\hat{\mathbf{w}} = \mathbf{X}^T\mathbf{y} \tag{12}$$

such that

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{13}$$

In [17], it is shown that

$$\hat{\mathbf{w}} = \eta(\mathbf{S}_W^{\phi})^{-1}(\mathbf{m}_1^{\phi} - \mathbf{m}_2^{\phi}) \tag{14}$$

where η is a constant. It is clear that (7) and (14) are identical except for an unimportant constant.

Unfortunately, the mapping function in equation (9) still needs to be known exactly. This is difficult to calculate or may not be available in practice. The following shows that the calculation of Φ can be avoided by adopting the kernel method.

It is assumed that $\mathbf{w} \in F$, so that it can be spanned by all the training samples in F. Thus, $\mathbf{w}$ can be expressed as

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x}_i) \tag{15}$$

Substituting Eq.(15) into Eq.(9) and replacing $\phi(x_i)^T \cdot \phi(x_j)$ with $k(x_i, x_j)$ gives

$$\begin{bmatrix} 1 & k(\mathbf{x}_1, \mathbf{x}_1) & \dots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & k(\mathbf{x}_N, \mathbf{x}_1) & \dots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \begin{bmatrix} w_0 \\ \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_1 \\ y_2 \\ \vdots \\ y_2 \end{bmatrix} \tag{16}$$

By writing the above equation in matrix form

$$\mathbf{P}\boldsymbol{\theta} + \mathbf{e} = \mathbf{y} \tag{17}$$

(where $\boldsymbol{\theta}$ is the parameter vector and $\boldsymbol{e}$ is the error vector) the Fisher discriminant analysis is converted to a least-squares formulation with the regression matrix $\mathbf{P}$ already known, leading to the solution

$$\hat{\boldsymbol{\theta}} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y} \tag{18}$$

Normally, the column terms in $\mathbf{P}$ are redundant and correlated, and the information matrix $(\mathbf{P}^T\mathbf{P})$ is ill-conditioned. The direct solution obtained from (18) is therefore not accurate. Among the numerical methods available for computing $\hat{\boldsymbol{\theta}}$, matrix decomposition methods are widely used [28] with Orthogonal Least Squares (OLS) being the best-known [14, 15]. An alternative, however, is the Fast Recursive Algorithm (FRA) which has proven to be more efficient and stable. In this work, improved two-stage selection (TSS) methods will be proposed and applied to the nonlinear Fisher discriminant analysis to obtain a more compact classifier with better generalization performance.

2.3 Two-stage selection

Stepwise selection is the recommended subset selection technique owing to its superior performance [5]. However, in conventional stepwise selection, all the terms already selected need to undergo some significance check before choosing a new term, and those regarded as insignificant are then removed from the model at each iteration. This inevitably increases the overall computation complexity. By contrast, the recently proposed two-stage selection algorithm [19], which includes a forward selection stage [16] and a second backward refinement stage, provides a more efficient alternative. Insignificant terms can be effectively removed at the second stage without much increasing the computation.

2.3.1 Forward recursive selection - first stage

The Fast Recursive Algorithm (FRA) is based on a recursive matrix $\mathbf{M}_k$ and a residual matrix $\mathbf{R}_k$ defined by

$$\mathbf{M}_k \triangleq \mathbf{\Phi}_k^T \mathbf{\Phi}_k \quad k = 1, \cdots, n \tag{19}$$

$$\mathbf{R}_k \triangleq I - \mathbf{\Phi}_k \mathbf{M}_k^{-1} \mathbf{\Phi}_k^T \qquad \mathbf{R}_0 \triangleq I \tag{20}$$

where $\mathbf{\Phi}_k \in \Re^{N \times k}$ contains the first k columns of the regression matrix $\mathbf{P}$ in (18). According to [16] and [19], the matrices $\mathbf{R}_k,\ k = 0, \cdots, n$ possesses the following attractive properties:

$$\mathbf{R}_{k+1} = \mathbf{R}_k - \frac{\mathbf{R}_k \boldsymbol{\phi}_{k+1} \boldsymbol{\phi}_{k+1}^T \mathbf{R}_k^T}{\boldsymbol{\phi}_{k+1}^T \mathbf{R}_k \boldsymbol{\phi}_{k+1}},\ k = 0, 1, \cdots, n-1 \tag{21}$$

$$\mathbf{R}_k^T = \mathbf{R}_k; \quad (\mathbf{R}_k)^2 = \mathbf{R}_k, \quad k = 0, 1, \cdots, n \tag{22}$$

$$\mathbf{R}_i \mathbf{R}_j = \mathbf{R}_j \mathbf{R}_i = \mathbf{R}_i, \quad i \geq j;\ i, j = 0, 1, \cdots, n \tag{23}$$

$$\mathbf{R}_k \boldsymbol{\phi}_j = \begin{cases} \mathbf{0}, & rank([\mathbf{P}_k, \boldsymbol{\phi}_j]) = k \\ \boldsymbol{\phi}_j^{(k)} \neq \mathbf{0}, & rank([\mathbf{P}_k, \boldsymbol{\phi}_j]) = k+1 \end{cases}, \quad j = 0, 1, \cdots, n \tag{24}$$

$$\mathbf{R}_{1,\cdots,p,\cdots,q,\ldots,k} = \mathbf{R}_{1,\cdots,q,\cdots,p,\ldots,k}, \quad p, q \leq k \tag{25}$$

Suppose the selected regressors are expressed as $\mathbf{p}_i$, $(i = 1, \cdots, n)$, equation (25) means that any change in the selection order of the $\mathbf{p}_i$ does not change the residual matrices $\mathbf{R}_k$. This property will help to reduce the computational effort in the second stage. The cost function in (4) can now be rewritten as:

$$J_k(\mathbf{P}_k) = \mathbf{y}^T \mathbf{R}_k \mathbf{y} \tag{26}$$

In this forward stage, the model terms are optimized one at a time. Suppose at the k^{th} step, one more term $\mathbf{p}_{k+1}$ is to be selected. The net contribution of $\mathbf{p}_{k+1}$ to the cost function can then be calculated as:

$$\begin{aligned} \Delta J_{k+1}(\mathbf{P}_k, \mathbf{p}_{k+1}) &= \mathbf{y}^T (\mathbf{R}_k - \mathbf{R}_{k+1}) \mathbf{y} \\ &= \frac{\mathbf{y}^T \mathbf{R}_k \mathbf{p}_{k+1} \mathbf{p}_{k+1}^T \mathbf{R}_k \mathbf{y}}{\mathbf{p}_{k+1}^T \mathbf{R}_k \mathbf{p}_{k+1}} \\ &= \frac{(\mathbf{y}^T \mathbf{p}_{k+1}^{(k)})^2}{\mathbf{p}_{k+1}^T \mathbf{p}_{k+1}^{(k)}} \end{aligned} \tag{27}$$

where $\mathbf{p}_{k+1}^{(k)} \triangleq \mathbf{R}_k \mathbf{p}_{k+1}$. According to (21), this net contribution can be further simplified by defining an auxiliary matrix $\mathbf{A} \in \Re^{n \times M}$ and a vector $\mathbf{b} \in \Re^{M \times 1}$ with elements given by:

$$a_{i,j} \triangleq (\mathbf{p}_i^{(i-1)})^T \mathbf{p}_j,\ 1 \leq i \leq j,\ 1 \leq j \leq M \tag{28}$$

$$b_j \triangleq (\mathbf{p}_j^{(j-1)})^T \mathbf{y} \tag{29}$$

where $\mathbf{p}_j^{(0)} = \mathbf{p}_j$. The definitions here are changed from the original TSS or FRA, so that other techniques can be integrated without loosing efficiency. In [16], it is shown that $a_{i,j}$ and b_j can be updated recursively using:

$$a_{i,j} = \mathbf{p}_i^T \mathbf{p}_j - \sum_{l=1}^{i-1} a_{l,i} a_{l,j} / a_{l,l} \tag{30}$$

$$b_j = \mathbf{p}_j^T \mathbf{y} - \sum_{l=1}^{j-1} (a_{l,j} b_l) / a_{l,l} \tag{31}$$

Now, by substituting (28) and (29) into (27), the net contribution of a new model term $\mathbf{p}_{k+1}$ to the cost function can be expressed as:

$$\Delta J_{k+1}(\mathbf{p}_{k+1}) = \frac{b_{k+1}^2}{a_{k+1,k+1}} \tag{32}$$

This provides a formula for selecting the best model term from the candidate pool at each step. In practice, the calculation of $a_{j,j}^{(k+1)}$ and $b_j^{(k+1)}$ $(j = k+1, \cdots, M)$ can be further simplified by recursive updating instead of using (30) and (31)

$$a_{j,j}^{(k+1)} = a_{j,j}^{(k)} - a_{k,j}^2 / a_{k,k} \tag{33}$$

$$b_j^{(k+1)} = b_j^{(k)} - a_{k,j} b_k / a_{k,k} \tag{34}$$

Thus at the end of each selection, these terms are updated and stored for use in the next comparison or selection. By default, $a_{j,j}^{(k)}$ and $b_j^{(k)}$ will be written as $a_{j,j}$ and b_j in what follows. The selection procedure then continues until some termination criterion is met (e.g., Akaike's information criterion (AIC) [5]) or a desired model size is reached.

Finally, after a satisfactory non-linear model has been constructed, the coefficients of each term are computed recursively according to:

$$\hat{\theta}_j = \left(b_j - \sum_{i=j+1}^{n} \hat{\theta}_i a_{j,i} \right) / a_{j,j}, \quad j = n, n-1, \cdots, 1. \tag{35}$$

2.3.2 Model refinement - second stage

This involves the elimination of insignificant terms due to constraints introduced in forward selection. Noting that the last selected term in the forward construction is always maximally optimized for the entire model, the backward model refinement can be divided into two main parts; Firstly, a selected term $\mathbf{p}_k,\ k = 1, \cdots, n-1$ is shifted to the n^{th} position as it was the last optimized one. Then, the contributions of all the candidate terms are recalculated based on the new $n-1$ selected regressors and compared with the one at the n^{th} position. If the shifted term is less significant than anyone from the candidate pool, it will be replaced, leading to a reduced training error without increasing the model size. This review is repeated until all the selected model terms are more significant than those remaining in the candidate pool.

Re-ordering of selected terms

Suppose a selected model term $\mathbf{p}_k$ is to be moved to the n^{th} position in the regression matrix $\mathbf{P}_n$. This can be achieved by repeatedly interchanging two adjacent terms so that

$$\mathbf{p}_q^* = \mathbf{p}_{q+1}, \quad \mathbf{p}_{q+1}^* = \mathbf{p}_q, \quad q = k, \cdots, n-1 \tag{36}$$

where the $*$ is used to indicate the updated value. By noting the property in (25), it is clear that only $\mathbf{R}_q$ in the residual matrix series is changed at each step. This is updated using

$$\mathbf{R}_q^* = \mathbf{R}_{q-1} - \frac{\mathbf{R}_{q-1}\mathbf{p}_q^*(\mathbf{p}_q^*)^T\mathbf{R}_{q-1}^T}{(\mathbf{p}_q^*)^T\mathbf{R}_{q-1}\mathbf{p}_q^*} \tag{37}$$

Meanwhile, the following terms also need to be updated:

- In matrix $\mathbf{A}$, only the upper triangular elements $a_{i,j}, i \leq j$ are used for regressor selection. The q^{th} and the $(q+1)^{th}$ columns, with elements from row 1 to $q-1$, need to be modified according to:

$$\begin{cases} a_{i,q}^* & = (p_i^{(i-1)})^T p_{q+1} & = a_{i,q+1} \\ a_{i,q+1}^* & = (p_i^{(i-1)})^T p_q & = a_{i,q} \end{cases}, \ i = 1, \cdots, q-1 \tag{38}$$

  The q^{th} row, with elements from column q to column n, is also changed using

$$a_{q,j}^* = \begin{cases} a_{q+1,q+1} + a_{q,q+1}^2/a_{q,q} & j = q \\ a_{q,q+1} & j = q+1 \\ a_{q+1,j} + a_{q,q+1}a_{q,j}/a_{q,q} & j \geqslant q+2 \end{cases} \tag{39}$$

  and the $(q+1)^{th}$ row $a_{q+1,j}$, for $j = q+1, \cdots, n$, is likewise changed to

$$a_{q+1,j}^* = \begin{cases} a_{q,q} - a_{q,q+1}^2/a_{q,q}^* & j = q+1 \\ a_{q,j} - a_{q,q+1}a_{q,j}^*/a_{q,q}^* & j \geqslant q+2 \end{cases} \tag{40}$$

- For the vector $\mathbf{b}$, only the q^{th} and the $(q+1)^{th}$ elements are altered. Thus

$$b_q^* = b_{q+1} + a_{q,q+1}b_q/a_{q,q} \tag{41}$$

$$b_{q+1}^* = b_q - a_{q,q+1}b_q^*/a_{q,q}^* \tag{42}$$

This procedure continues until the k^{th} term is shifted to the n^{th} position, the new regression matrix and the series of residual matrices then becomes

$$\mathbf{P}_n^* = [\mathbf{p}_1, \cdots, \mathbf{p}_{k-1}, \mathbf{p}_{k+1}, \cdots, \mathbf{p}_n, \mathbf{p}_k] \tag{43}$$

$$\{\mathbf{R}_k^*\} = [\mathbf{R}_1, \cdots, \mathbf{R}_{k-1}, \mathbf{R}_k^*, \cdots, \mathbf{R}_n^*] \tag{44}$$

Comparison of net contributions

As the model term $\mathbf{p}_k$ of interest has now been moved to the n^{th} position in the full regression matrix $\mathbf{P}_n$, its contribution to the cost function needs to be reviewed. The contribution of each candidate term is calculated based on the re-ordered terms $\mathbf{p}_j$ ($j = 1, \cdots, n-1$). Specifically, $a_{j,j}$ and b_j for $j = n+1, \cdots, M$ are updated using

$$a_{j,j}^* = a_{j,j}^{(n+1)} + (a_{n,j}^*)^2/a_{n,n}^* \tag{45}$$

$$b_j^* = b_j^{(n+1)} + b_n^* a_{n,j}^*/a_{n,n}^* \tag{46}$$

The significance of the shifted term $\mathbf{p}_k$ and those remaining in the candidate pool are reviewed and their contributions to the cost function being recalculated as:

$$\Delta J_n^*(\mathbf{p}_k) = \Delta J_n(\mathbf{p}_n^*) = (b_n^*)^2/a_{n,n}^* \tag{47}$$

$$\Delta J_n^*(\boldsymbol{\phi}_j) = (b_j^*)^2/a_{j,j}^* \tag{48}$$

Now, assuming $\Delta J_n^*(\boldsymbol{\phi}_s) = \max\{\Delta J_n^*(\boldsymbol{\phi}_j), j = n+1, \cdots, M\}$, and that $\Delta J_n^*(\boldsymbol{\phi}_s) > \Delta J_n(\mathbf{p}_n^*)$, then $\boldsymbol{\phi}_s$ will replace $\mathbf{p}_n^*$ in the regression matrix $\mathbf{P}_n^*$, and $\mathbf{p}_n^*$ will be returned to the candidate pool and will take the position of $\boldsymbol{\phi}_s$. Meanwhile, the following terms need to be updated according to this interchange:

- In the matrix $\mathbf{A}$, the following terms are updated

$$a_{i,n}^* = a_{i,s}, \quad a_{i,s}^* = a_{i,n} \quad (i = 1, \cdots, n-1) \tag{49a}$$

$$a_{n,j}^* = \begin{cases} a_{s,s} & j = n \\ a_{n,s} & j = s \\ \boldsymbol{\phi}_s^T \boldsymbol{\phi}_j - \sum_{l=1}^{n-1} a_{l,s} a_{l,j}/a_{l,l} & \forall j,\ j \neq n\ \&\ j \neq s \end{cases} \tag{49b}$$

$$(a_{j,j}^{(n+1)})^* = \begin{cases} a_{n,n} - (a_{n,s}^*)^2/a_{n,n}^* & j = s \\ a_{j,j}^* - (a_{n,j}^*)^2/a_{n,n}^* & j \neq s \end{cases} \tag{49c}$$

- In the vector $\mathbf{b}$,

$$b_n^* = b_s \tag{50a}$$

$$(b_j^{(n+1)})^* = \begin{cases} b_n - a_{n,s}^* b_n^*/a_{n,n}^* & j = s \\ b_j - a_{n,j}^* b_n^*/a_{n,n}^* & j \neq s \end{cases} \tag{50b}$$

The shifting and comparison procedures described above are repeated until no insignificant term remains in the selected model. Finally, after a satisfactory model has been constructed, the coefficients of each model term are computed recursively using (35).

2.4 Synthetic minority over-sampling technique

A major problem caused by imbalanced data set is that most classifier tend to attribute the minority class instances to the majority class due to insufficient minority class training instances in the decision region. As a result, the trained decision boundary tends to be far away from the majority class. The contribution of SMOTE is to enhance the significance of the small and specific region belonging to the minority class in the decision region, which leads to the better generalization of the classifier.

The SMOTE over-samples the minority data by creating synthetic instances based on the original minority data set. Suppose one minority data sample, denoted by x_0, was selected, the synthetic data points are randomly generated on the lines linking x_0 with some of its K nearest neighbours. K is pre-determined based on the sparsity of minority data set. A oversampling ratio of the original minority data size, denoted by $\beta\%$ can also be pre-determined. Thus, a new synthetic instance x_s can be given as

$$x_s = x_0 + \delta(x_{0,k} - x_0) \tag{51}$$

where $x_{0,k}$ is the k^{th} nearest neighbours of x_0 in the minority class, and $\delta \in [0, 1]$ is a random number. This procedure is repeated for all the samples in minority class.

3 Classification on original data sets

There are some sample values missing from the original data set. So some regression models (Radial basis function neural network model) were first constructed to approximate those missing values. All the following analysis are based on the full data set without any missing value.

3.1 Using support vector machine

the Matlab version of LIBSVM package was used for the classifier construction. As the number of data available is still small for data-driven modelling, both the 5-fold cross validation and leave-one-out cross validation were used to measure the generalization performance of resultant classifier. Table 1 shows the results, including the structure and parameters of SVM classifiers.

Table 1: Experimental results from SVM on original data set (C and σ are from (1) and (2), the aim is to use a small number of SVs to achieve as higher accuracy as possible)

cross validation	Number of SVs	C	σ	Accuracy
5-fold	57	2.64	0.57	83.72%
Leave-one-out	61	2.3	0.87	83.72%

3.2 Using Fisher discriminant analysis

The Fisher discriminant analysis initially uses all data samples available to construct the classifier, so the resultant classifier is usually very large and subset selection algorithm will then be employed to enhance its sparsity. The method used in this report is our proposed two-stage selection [19]. Further, in order to evaluate the generalization performance of constructed classifier, 2/3 of the original data is randomly selected for training with the remaining 1/3 reserved for testing. In order to compare the performance of NFD with SVM, the generated data set is the same as previous section. Table 2 shows the performances of different classifiers constructed by Fisher discriminant analysis. Gaussian function is used as the mapping function with $\sigma^2 = 0.5$ (by exhaustive search). The sensitivity and specificity are given in (52) and (53) respectively.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{52}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{53}$$

where TP is represents true positive number(the samples belongs to positive class, and the classifier predict it as positive class as well), FP is the false positive number (the samples that belongs to negative class, but the classifier predict it as positive class), TN is the tru negative numbers and FN denotes the false negative numbers. Normally, the specificity is more important, as the risk or treating sick as normal is very high.

According to table 2, it is clear that the increase of classifier size leads to the increase of both training accuracy and testing accuracy, but there is not much difference on test accuracy when the classifier size increased to 10. The test specificity sometimes becomes NaN. This may be caused by both the small number of test samples and small amount of negative classes (normal cattle), so their values were not given in table 2.

Table 2: The performances of different classifier based on original data set (The training specificity are all 100%, this is caused by the small number of negative samples)

Classifier size	Training Sensitivity	Training Specificity	Test Sensitivity
4	87.04%	100.00%	79.31%
6	88.68%	100.00%	79.31%
10	94.00%	100.00%	85.19%
17	97.92%	100.00%	82.14%
19	97.92%	100.00%	85.19%

4 Classification on balanced data sets

As mentioned in the introduction section, imbalanced data will lead to inaccurate training. In this section, the SMOTE method was used to randomly generate some negative samples (normal cattle) to rebalance the two classes. Originally, there are 70 positive samples (Lipids = 1, 2, 3, or 4), and 14 negative samples (Lipids = 0), so there are 70-14 = 56 negative samples need to be generated, leading to a total number of 140 data points.

The results of SVM and NFD are given in table 3 and table 4 respectively. It is clear that the results of SVM classification on balanced data is much better than on the original data set. However, for the Fisher discriminant analysis, there are not much difference with the original one. This means that SVM method is more sensitive to the data balance while NFD is more robust on the data balance.

Table 3: Experimental results from SVM on balanced data set

cross validation	Number of SVs	C	σ	Accuracy
5-fold	77	3.03	1.32	98.51%
Leave-one-out	77	3.03	1.32	96.27%

Table 4: The performances of different classifier based on balanced data set (The data set used is the same as used in SVM experiment, so the algorithm runs only once)

Classifier	Training		Testing	
size	Sensitivity	Specificity	Sensitivity	Specificity
4	80.70%	87.50%	64.00%	80.00%
6	91.67%	85.37%	72.00%	90.00%
10	94.00%	92.31%	75.00%	90.48%
17	98.00%	97.44%	85.71%	91.67%
19	98.04%	100.00%	85.71%	91.67%

5 Conclusion and future work

Through the results, the classifier constructed by non-linear Fisher discriminant analysis (NFD) and two-stage selection is more compact than from support vector machine (SVM), and NFD is more robust on the imbalance data.

The future work includes:

- multi-class classification. In this report, all the cattle with larger than 0 Lipids are regarded as positive class. However, it will be more useful to predict the Lipid lever.

- Improved compactness of SVM. The Least-square support vector machine has similar performance with conventional SVM, but the resultant classifier is more compact. Our proposed two-stage selection can also be integrated with LS-SVM.

References

[1] C.M. Bishop. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[2] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–404, 2000.

[3] A. Estabrooks, T. Jo, and N. Japkowicz. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

[4] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.

[5] O. Nelles. *Nonlinear System Identification*. Springer, 2001.

[6] X. Hong, P. M. Sharkey, and K. Warwick. Automatic nonlinear predictive model-construction algorithm using forward regression and the press statistic. *IEE Proceedings: Control Theory and Applications*, 150(3):245–254, 2003.

[7] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Pr, 1990.

[8] B.D. Ripley. *Pattern recognition and neural networks*. Cambridge Univ Press, 2008.

[9] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. R. Muller. Fisher discriminant analysis with kernels. *Neural networks for signal processing IX*, pages 41–48, 1999.

[10] V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 2000.

[11] B. Schölkopf, C.J.C. Burges, and A.J. Smola. *Advances in kernel methods: support vector learning*. The MIT press, 1999.

[12] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[13] R.F. Harrison and K. Pasupa. A simple iterative algorithm for parsimonious binary kernel fisher discrimination. *Pattern Analysis and Applications*, 13(1):15–22, 2010.

[14] S. Chen, S.A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50(5):1873–1896, 1989.

[15] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.

[16] K. Li, J.X. Peng, and G.W. Irwin. A fast nonlinear model identification method. *IEEE Transactions on Automatic Control*, 50(8):1211–1216, 2005.

[17] S.A. Billings and K.L. Lee. Nonlinear fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm. *Neural Networks*, 15(2):263–70, 2002.

[18] A. Sherstinsky and R. W. Picard. On the efficiency of the orthogonal least squares training method for radial basis function networks. *IEEE Transactions on Neural Networks*, 7(1):195–200, 1996.

[19] K. Li, J. X. Peng, and E. W. Bai. A two-stage algorithm for identification of nonlinear dynamic systems. *Automatica*, 42(7):1189–1197, 2006.

[20] K.Z. Mao and S.A. Billings. Algorithms for minimal model structure detection in nonlinear dynamic system identification. *International Journal of Control*, 68(2):311–330, 1997.

[21] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[22] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge Univ Pr, 2000.

[23] M.P.S. Brown, W.N. Grundy, D. Lin, N. Cristianini, C.W. Sugnet, T.S. Furey, M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, 97(1):262, 2000.

[24] T.S. Furey, N. Cristianini, N. Duffy, D.W. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906, 2000.

[25] Y. Lee and C.K. Lee. Classification of multiple cancer types by multicategory support vector machines using gene expression data. *Bioinformatics*, 19(9):1132, 2003.

[26] R.T. Cunningham, M.H. Mooney, X.L. Xia, S. Crooks, D. Matthews, M. OKeeffe, K. Li, and C.T. Elliott. Feasibility of a clinical chemical analysis approach to predict misuse of growth promoting hormones in cattle. *Analytical Chemistry*, 81(3):977–983, 2009.

[27] C.W. Hsu, C.C. Chang, C.J. Lin, et al. A practical guide to support vector classification, 2003. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[28] K. Z. Mao. Fast orthogonal forward selection algorithm for feature subset selection. *IEEE Transactions on Neural Networks*, 13(5):1218 – 1224, 2002.