MyShake Seismic Data Classification

Siqi Qin
s7qin@eng.ucsd.edu

Jinhan Zhang
jiz530@eng.ucsd.edu

Xiaoye Liu
lxiaoye@eng.ucsd.edu

Abstract

Early Earthquake Warning has become an important issue as the active earth activities increase. MyShake from UC Berkeley is one of the most famous earthquake prediction App in nowadays smart phone. This paper attempts to improve the classification accuracy of the human and earthquake based on the previous works on MyShake dataset. Both time and frequency domain features are used for the model training. Four classification methods are tried and models are evaluated with accuracy, balanced error rate and precision&recall. Boosting gives the best accuracy of 99.28 % and balanced error rate of 0.95 %.

1. Introduction

1.1. Background

Earthquake is a kind of natural catastrophe that can kill or hurt a large number of people, and cause huge and unrecoverable destruction to constructions, and results in huge financial losses. Earthquake Early Warning(EEW) system is able to detect magnitude and location of an earthquake a few seconds ahead of its damaging waves arrive [1]. It can issue the warning to the affected area which allows people to react and do some preparation to avoid further hurt such as finding a safe place, opening the emergency exit and many other things. Since smart phones are very prevalent and have its own built-in sensors and communications modules, the development of EEW application in smart phone becomes very useful. *Myshake* from UC Berkeley [2] is an well-known earthquake prediction app nowadays. Besides earthquake, shake can also be produced by human motions when the phone is put inside the pocket. Based on the dataset given by *Myshake*, the objective of this project is to try different classification models and improve the classification performance between these two classes.

1.2. Previews Works

Many people have tried to improve classification accuracy using different models and classification method. Some of the final projects from STATS 215 in UC Berkeley give good results. One of them used SuperLearner for classification task and found that it gives excellent predictive performance. Another group proposes two ways, incorporating class weights into the RF classifier and combining the sampling technique and the ensemble idea respectively, both of which are based on the Random Forest algorithm. In this project, different model and classifier are tried referenced to these methods.

2. Data Engineering

To better select useful features from the dataset for the classification task, an insight into the raw data is necessary and some preprocessing is required. This section attempts to explain the characteristics of the raw dataset and the preprocessing methods used to solve the potential problems and cast it into more useful features for the classification task.

2.1. Dataset Overview and Preprocessing

The MyShake dataset is kindly provided by Peter. The dataset includes 192 trials for the table experiments, 992 trials for the earthquake simulation and 26343 trials for the human activity. In each trial, the raw data are the x, y and z components of acceleration with unit in g. Also, the sample rate and related time information is given. In the classification problem, the trials from table experiment and earthquake simulation will be used to represent the earthquake class and the trials from the human activity are used to represent the human class. The statistics of the dataset for this classification problem is shown in **Table 1**.

Table 1. Statistics of Dataset

Class	Data Size
Earthquake	1184
Human	26343

One potential problem here is that the size of data in x, y, z may be different, so the first thing to do here is to cast them into the same length so that it will be more convenient for the future combination of these 3 dimensions. Next, the features will be extracted from these 3 dimension accelerations.

2.2. Feature Extraction

Most of the previous works from UC Berkeley are classifying earthquake and human based on the time domain features. In this project, both the time domain and frequency domain features are used.

In the time domain, only some statistical features are used. These features in x, y and z dimension are combined as the features in time domain. Time domain features for each dimension are listed in **Table 2**.

Table 2. Time domain features

Feature	Significance
Max, Min	peak values
Entropy	information contained
Mean	Average Energy of the signal
Variance	2nd order value of vibration
Range Count	distribution of energy
Mean Absolute Change	1st order value of vibration
Mean 2nd order derivative	Specific value of vibration

Overall, there are 24 features extracted in the time domain.

In the frequency domain, the features extracted are the band power of the raw data. To fix the number of features in all dimension, a 512 point Fast Fourier Transform(FFT) is performed in both x, y and z dimension raw data. Considering that all of these 3 raw data are real, the FFT result will be symmetrical with the DC axis. Thus, only the second half of the FFT result will be extracted(point 256 to 512). Then, these 256 points are binned with a size of 16, and the normalized energy in each bin are used as the features in frequency domain. Normalization is performed by being divided by the overall energy of these 256 points. Figure 1 gives a more direct visualization of this process.

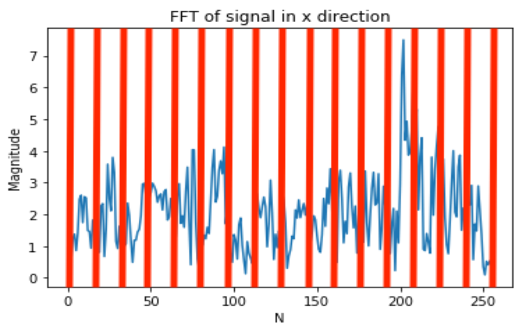

Figure 1. Feature Extraction in Frequency Domain

For each of the dimension, there will be 16 features in the frequency domain. Thus, in total, there are 72 features for the classification task.

2.3. Feature Selection

Combining the 16 features in frequency domain and the 8 features in time domain for each of x, y and z direction, there are 72 features in total for training the model. However, some of these features should be redundant. To select the most useful features for the model, KBest methods and Tree Based features selection methods are used here.

2.3.1 KBest Feature Selection

Univariate feature selection works by selecting the best features based on univariate statistical tests [3]. Chi-squared statistics between each non-negative feature and class are computed and the scores are used to select the top 10 features with highest values for the test chi-squared statistic from the input features.

The top 10 features selected overall and their corresponding scores are shown in **Table 3**.

Table 3. KBest Top 10 Features

Feature	Score
X Range Count	248.19747915794642
Y Range Count	240.42413597998859
Z Entropy	57.931555342198919
X Entropy	52.214238430489537
Y Mean	43.117537869501476
X Mean	41.700174402145997
Y Entropy	40.866437388414006
Z Max	28.621408062615217
Y Mean Absolute Change	19.442295887542997
X Mean Absolute Change	17.739705664715959

It can be found that the best features selected with KBest are all in time domain. It seems that the features extracted in frequency domain are not preferred in this feature selection method.

2.3.2 Tree Based Feature Selection

Another feature selection method used in this project is tree based feature selection. Methods that use ensembles of decision trees in python, such as random forest and extra tree classifier can also be used to compute the importances of each attribute, which in turn can be used to discard irrelevant features [4]. This can be realized by setting importance threshold for all the features.

The top 10 features selected overall and their corresponding importance are shown in **Table 4**.

Compared with KBest method, the tree based feature selections methods seems to be more preferable to the frequency domain features. Thus, both of these selected features will be used for the model training to compare the performance of two pairs of 10 features.

Table 4. Tree Based Top 10 Features

Feature	Importance
Y Bin6	0.069951000858849588
X Bin3	0.056770193478084797
Z Bin8	0.043527418795959581
Y Bin1	0.042234899021278527
Z Bin1	0.039207346146501287
X Bin7	0.035142537352107338
X Bin1	0.033746800624808879
X Range Count	0.033255561121663127
Y Range Count	0.032437465139782469
Y Mean Absolute Change	0.030606836865436842

3. Method

In this section, several kinds of classifiers are introduced and explored. There are mainly 4 kinds of classifiers used to do the classification: Fisher's Linear Discrimination, Support Vector Machine, Random Forest and Gradient Boosting. They will be explained in detail during this part.

3.1. Fisher's Linear Discrimination

Fisher's Linear Discrimination, also called as Linear Discriminant Analysis(LDA), is a widely-used method for supervised learning [5]. It tries to find a linear combination of features which can best represent each class or separate multiple classes. This combination of features can be further used to do dimensionality reduction or as a classifier [6].

The main idea for LDA is to separate points from different classes most while make points with the same label close. Thus LDA aims to maximize the objective function below:

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

where S_B is the between-class scatter matrix, S_W is the within-class scatter matrix and w is the projection matrix. The mathematic definition of the scatter matrices are as below:

$$S_B = \sum_c (\mu_c - \overline{x})(\mu_c - \overline{x})^T$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T$$

where c is the number of different classes. The visualization of Fisher's Linear Discrimination is shown as Figure 2. From the objective function above, the optimal projection matrix in the problem of two class can be obtained by:

$$w = S_W^{-1}(\mu_1 - \mu_2)$$

where μ_1 and μ_2 are the mean of two classes repectively.

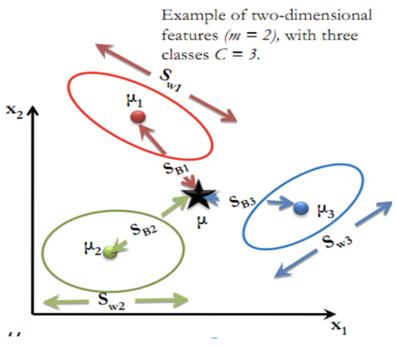

Figure 2. Fisher's linear discrimination for 3 classes

In summary, Fisher's Linear Discrimination attempts to express dependent variable as a linear combination of other features [7][8], by explicitly trying to model the difference between the classes of data. LDA works with the case that measurements made on independent variables for each observation are continuous quantities. However, if the problem contains categorical independent variables, discriminant correspondence analysis [9] [10] may be used to help with this case.

3.2. Support Vector Machine

Support vector machine [11] is a supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis [12]. Support vector machine constructs a hyperplane in a high-dimensional space which has the largest distance to the nearest training-data point of any class. Generally, its idea is to construct a classifier which focuses on points close to other classes by minimizing the misclassification error. Thus a classifier of the following form is desirable:

$$y_i = \begin{cases} 1 & if \quad X_i\theta - \alpha > 0 \\ -1 & otherwise \end{cases}$$

Then the objective function which aims to minimize the number of misclassifications would be:

$$argmin_\theta \sum_i \delta(y_i(X_i\theta - \alpha) \leqslant 0)$$

If the data to train is linearly separable, the classifier can simply choose two parallel hyperplanes that separate the two classes of data, and the distance between these hyperplanes is the largest. This is called SVM with hard margin, and the optimization problem would become:

$$argmin_{\theta,\alpha} \|\theta\|_2^2 \quad with \quad \forall_i y_i(X_i\theta - \alpha) \geqslant 1$$

However, this idea is not alway good since we may set the boundary of classes as shown in Fig 3.

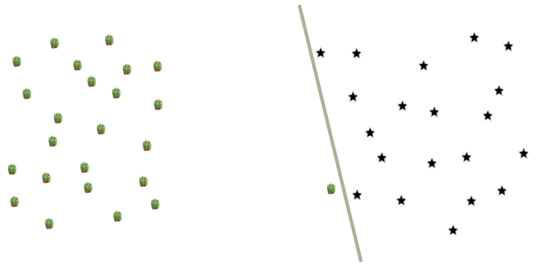

Figure 3. SVM with hard-margin

In order to make SVM compatible for the case that the data is not linearly separable, we use the hinge loss function which has the form as below:

$$max(0, 1 - y_i(x_i\theta - \alpha))$$

This function is zero if the output matches the label (i.e. the point lies on the correct side of the margin). While for the other case, the hinge loss would be proportional to the distance from the margin. Apply the hinge loss into our optimization process, we then wish to minimize the objective function below:

$$argmin_{\theta,\alpha}[\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(X_i\theta - \alpha))] + \lambda\|\theta\|^2$$

where λ determines the tradeoff between increasing the margin-size and minimizing the error number. The result of SVM with soft-margin is shown as Fig 4.

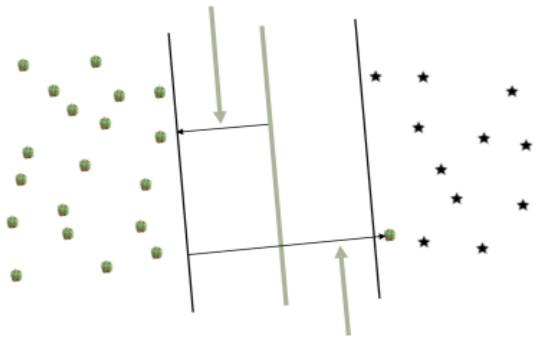

Figure 4. SVM with soft-margin

All discussed above is about performing linear classification, however, SVM can also perform non-linear classification using the kernel trick. The main idea for kernel trick is to implicitly map inputs into higher-dimensional feature spaces[13].

3.3. Random Forest

Bagging is the basis of random forest, so we will introduce bagging first here. Bagging (Breiman, 1996), also called bootstrap aggregating, is a type of ensemble meta-estimator that fits base classifiers. It fits many trees to bootstrap-resampled versions of the training data, and classify by majority vote. Suppose $C(S, x)$ is a classifier, such as a tree, based on our training data S, producing a predicted class label at input x. To bag the classifier, we bootstrap (randomly select) samples to sets $S^1, ...S^M$ to replace the training data.

$$C_{bag}(x) = \arg\max_{m}\{C(S^m, x)\}_{m=1}^{M}$$

Bagging introduces randomness to the training samples, reduces the variance of unstable classifiers and improves the performance [14]. The random forest (Breiman 2001) classifier is a substantial modification of bagging that builds a big collection of de-correlated trees and then averages them. It introduces more randomness to the model. At each tree split, it randomly selects m features from all the features and only considers those features for splitting [15].

Each tree in bagging is identically distributed so the expectation of an average of M trees is equal to the expectation of any one tree in the sets. This means the bias of bagging is equal to individual trees which means it only reduces the variance and cannot reduce the bias. Consider an average of M i.d. random variables with positive pairwise correlation ρ, each with σ^2 variance, then the variance of the average is $\rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$. For bagging, the M increases, the second term disappears, but the first term still remains. However, for random forest, it also reduces the correlation between the trees to improve the variance reduction.

3.4. Gradient Boosting

Gradient boosting is an ensemble learning method, which produces a classifier in the form of an ensemble of weak prediction models. The details of this algorithm are introduced as followed [16].

```
**Initialize** model with a constant value:
  $F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$
**For m = 1 to M**:
  compute pseudo residuals:
    $r_{im} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)}$
  Fit a base learner $h_m(x)$ with $(x_i, r_{im})_{i=1}^{n}$
  Computer multiplier $\gamma_m$:
  $\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$
  Update the model:
    $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$
**Output** $F_M(x)$
```

Algorithm 1: Pseudo code for Gradient Boost

The gradient boosting is similar to the gradient descent method in model optimization. For the update of parameter γ, all data points have the same weight at the beginning. But as the procedure goes on, the misclassified points are given more and more weights(if we want to get the minimum sum of the loss functions, we should try our best to modify the one that have bigger residuals. Thus, the misclassified points will dominate in the parameter selection of γ) [17]. Finally, the parameters for the model we train will give us less misclassified labels, which enhances the performance.

4. Result Analysis

Those 4 models introduced in **Section 3** are tested for the classification task with only time domain features, only frequency domain features and the combination of them two. Both the training set and test set have 500 data for earthquake class and 10000 data for human class. Due to the length limitation of this paper, only the results for the one combining time and frequency domain features will be discussed here. The results for the other two can be found in the provided code. Here, three evaluation methods are used, which includes Accuracy, Balanced Error Rate and Precision & Recall.

4.1. Accuracy

In this part, models are assessed primarily with accuracy. The result for the classification performance of these 4 models are shown in **Table 5**. Trivially, accuracy can reflect the percentage of correctly classified data versa overall dataset. But for this specific dataset, it may not work well. Take SVM as an example, it will give 95.2381% correct rate with tree based features and 95.4095% correct rate with KBest features, which are pretty good results for accuracy. But when insight into the distribution of the misclassified data, the problem reflects out. For the Tree based features, there are 500 false human, which means that 500 earthquakes are classified as human. But remember that there are only 500 data for the earthquake here! In this way, SVM just classified all the earthquake data into human. This is disastrous because it is much costly to mis-classify earthquake into human. It is earthquake prediction!

Table 5. Performance Summary, Unbalanced Dataset

Method	Tree Based			KBest		
	Accu	F EQ	F Man	Accu	F EQ	F Man
LDA	0.980571	82	122	0.951429	95	415
SVM	0.952381	0	500	0.954095	5	477
Random Forest	0.992762	12	64	0.998762	4	9
Boosting	0.992762	19	57	0.998952	9	2

This results from the unbalance of the dataset. Considering the size of earthquake is too small compared to the human class (500 vs 10000), the earthquake data in the model training have much less dominance, thus misclassifying the minority is less costly for the model as it is always easy to follow the majority. To fix this problem, the earthquake data are given 20 times weight compared to the human ones in the training to balance their dominance in the model. The performance of the models trained with weighted data are shown in **Table 6**. It is easy to find that after given higher weight, the number of false human gets smaller, even based on the accuracy, the models get ”worse”.

Table 6. Performance Summary, Weighted Model

Method	Tree Based			KBest		
	Accu	F EQ	F Man	Accu	F EQ	F Man
LDA	0.97	230	85	0.898952	1053	8
SVM	0.773143	2380	2	0.94181	606	5
Random Forest	0.991714	14	73	0.998952	4	7
Boosting	0.985619	149	2	0.998381	16	1

Based on the performance summary of the weighted models, boosting gives the best classification result as it gives a relatively high accuracy and has small number of false human. Random forest also performs well compared with simple models like LDA and SVM.

4.2. Balanced Error Rate

For the unbalanced dataset, balanced error rate is a more reasonable parameter to assess the performance of the model [18]. Balanced Error Rate is based on the number of the true positive, true negative, false positive and false negative, which are shown in Figure 5 (Cited from UCSD CSE 258 material).

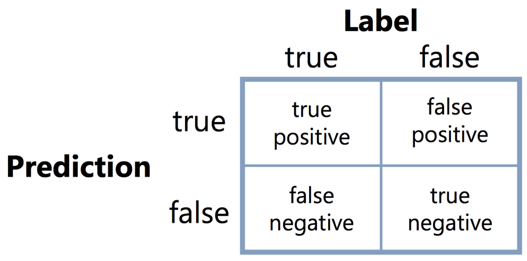

Figure 5. Classification result definition

Based on these definitions, the accuracy as introduced in last section should be $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$. From the equation, the problem of unbalance between different data points becomes clear. Here, true positive rate(TPR), true negative rate(TNR) and Balanced Error Rate(BER) are defined as followed.

$$TPR = \frac{TP}{TP+FN}$$
$$TNR = \frac{TN}{TN+FP}$$
$$BER = 1 - \frac{TPR+TNR}{2}$$

From the equation, we can find that the TNR and TPR represents the ”accuracy” in both class and then they are averaged with the same weight. Thus, the unbalance can be canceled with this assessment and the evaluation will become more reasonable. The result for the classification performance of these 4 models are shown in **Table 7**. The problem of SVM in misclassifying all earthquake into human becomes more evidently as the TPR are 0 and the BER is 0.5 high.

Table 7. Performance Summary, Unbalanced Dataset

Method	Tree Based			KBest		
	TPR	TNR	BER	TPR	TNR	BER
LDA	0.756	0.9918	0.1261	0.17	0.9905	0.41975
SVM	0.0	1.0	0.5	0.046	0.9995	0.47725
Random Forest	0.872	0.9988	0.0646	0.982	0.9996	0.0092
Boosting	0.886	0.9981	0.05795	0.996	0.9991	0.00245

Giving weights to the models, the performance becomes better as shown in **Table 8**.

Table 8. Performance Summary, Weighted Model

Method	Tree Based			KBest		
	TPR	TNR	BER	TPR	TNR	BER
LDA	0.83	0.977	0.0965	0.984	0.8947	0.06065
SVM	0.996	0.762	0.121	0.99	0.9394	0.0353
Random Forest	0.854	0.9986	0.0737	0.986	0.9996	0.0072
Boosting	0.996	0.9851	0.00945	0.998	0.9984	0.0018

Based on the performance summary of the weighted models, boosting is still the best one among these 4 models as it give a BER less than 0.01. Random forest also performs well compared with those simple models.

4.3. Precision & Recall

The last method to assess these models is precision & re all. After classification, each data will have a score given by the model. For example in SVM, the score is the distance to the boundary. Assuming the distance in the positive class is positive, then the higher the score, the more confident it is for the classification result. Precision and recall are based on these scores. Given these scores, true class labels are sorted with the score as key. And in the sorted version of labels, precision represents the accuracy of how many true positives are returned in a certain length of dataset and recall shows how many true positives are returned within all the true positives [19]. Assuming that 6 true positive are returned in the top 10 scored data and there are only 8 true positives overall. Here, the precision is $\frac{6}{10}$ and the recall is $\frac{6}{8}$.

Based on the definition above, it is desirable that both the precision and the recall are high. But as a result, they two can not always maintain in a high level. Thus, maximizing the product of precision and recall can be a reasonable way to evaluate the models. Figure 6, Figure 7 and Figure 8 shows the precision-recall curve of the models(random forest has no scores, thus it will not appear in this section).

As the figures shows, Boosting is also the best one among these three models as it gives the max product of precision and recall, which is represented with the area enveloped by the curve.

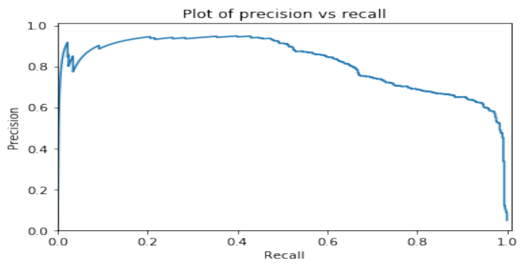

Figure 6. Precision-Recall Curve of SVM

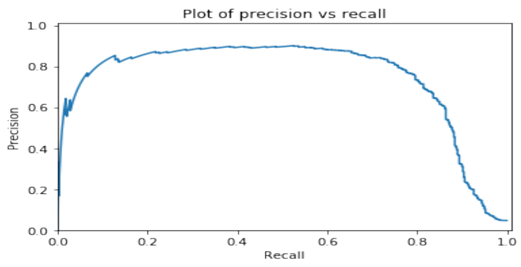

Figure 7. Precision-Recall Curve of LDA

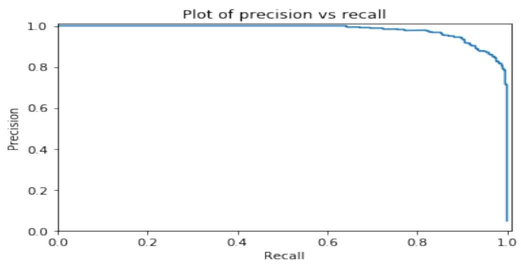

Figure 8. Precision-Recall Curve of Boosting

5. Conclusion and Future Directions

In this project, a complete data analysis procedure is performed from data preprocessing and feature extraction to the classification and analysis. Two machine learning methods covered in class(LDA and SVM) and two extra methods random forest and boosting are used to realize the classification task. Based on the result analysis, boosting gives the best performance among these 3 evaluation methods.

In the future, there are 3 directions to go. Firstly, except for earthquake signal and human activity, there are still many kinds of signals that may introduce 'noise' into the model [20]. Thus, the models need to be modified for multi-class classification problem(for example, the third class can be added to represent the noise). Secondly, the frequency feature extraction method can be improved to realize better performance. Finally, the prediction of the earthquake magnitude will be tried with regression methods.

References

[1] Richard M Allen and Hiroo Kanamori. The potential for earthquake early warning in southern california. *Science*, 300(5620):786–789, 2003.

[2] Qingkai Kong, Richard M Allen, Louis Schreier, and Young-Woo Kwon. Myshake: A smartphone seismic network for earthquake early warning and beyond. *Science advances*, 2(2):e1501055, 2016.

[3] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[4] V Sugumaran, V Muralidharan, and KI Ramachandran. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing*, 21(2):930–942, 2007.

[5] H. Yu and J. Yang. A direct lda algorithm for high-dimensional data with application to face recognition. *Pattern Recognition*, 34(10):2067–2069, 2001.

[6] P. Tahmasebi and M. Mortazavi A. Hezarkhani and. Application of discriminant analysis for alteration separation. *Australian Journal of Basic and Applied Sciences*, 6(4):564–576, 2010.

[7] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):197–188, 1936.

[8] G. J. McLachlan. Discriminant analysis and statistical pattern recognition. *Wiley Interscience*, 2004.

[9] H. Abdi. Discriminant correspondence analysis. *N.J. Salkind (Ed.): Encyclopedia of Measurement and Statistic*, pages 270–275, 2007.

[10] G. Perriere and J. Thioulouse. Use of correspondence discriminant analysis to predict the subcellular location of bacterial proteins. *Computer Methods and Programs in Biomedicine*, 70:99–105, 2003.

[11] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on computational learning theory*, page 144, 1992.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[13] B. Yekkehkhany, A. Safari1, S. Homayouni, and M. Hasanlou. A comparison study of different kernel functions for svm-based classification of multi-temporal polarimetry sar data. *The 1st ISPRS International Conference on Geospatial Information Research*, pages 15–17, 2014.

[14] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1):105–139, 1999.

[15] Mahesh Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, 2005.

[16] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[17] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[18] Dario Strbenac, Graham J Mann, Jean YH Yang, and John T Ormerod. Differential distribution improves gene selection stability and has competitive classification performance for patient survival. *Nucleic acids research*, 44(13):e119–e119, 2016.

[19] Jake Lever, Martin Krzywinski, and Naomi Altman. Points of significance: Classification evaluation. *Nature Methods*, 13(8):603–604, 2016.

[20] Stefan Bosse. Distributed machine learning with self-organizing mobile agents for earthquake monitoring. pages 126–132, 2016.