Master's thesis 2015

Performance Evaluation of Feature Learning For Stroke Classification In A Microwave-based Medical Diagnostic System

Yimeng Hou

CHALMERS
UNIVERSITY OF TECHNOLOGY

Master of Communication Engineering
Department of Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden

Abstract

In recent years, *stroke* has become an important issue to consider medically and socially since it poses high rate of human mortality and disability worldwidely. It typically results from impaired blood supply in the brain. Pathologically, it can be classified into two types: ischemic stroke and hemorrhagic stroke. While early treatment can effectively save lives, the treatment to each type are different and wrong treatment may deteriorate patients' condition. Therefore, it is necessary to diagnose the type of stroke before any treatment is performed. Medfield Diagnostics proposed a solution for stroke diagnostics in pre-hospital scenarios where a medical instrument called *Strokefinder* can be used to diagnose stroke type. This instrument uses microwave antennas to transmit and receive response signals and acquire the data for each patient. After that, machine learning algorithms are used to process the data and perform classification. Earlier methods already show high classification accuracy for predicting the correct type of stroke.

This thesis is a successive research based on earlier development from Medfield Diagnostics and aims to investigate whether new methods would further improve the performance of stroke classification. Specifically, a methodology named feature learning will be evaluated. One of the common feature learning algorithms is autoencoder, which is a reconstruction-based unsupervised learning algorithm with the purpose of dimensionality reduction. Since the data from stroke diagnostic instrument is massive and high-dimensional, applying autoencoder before a classifier will lower the dimension of acquired data and may possibly improve the classification performance.

The thesis starts with basic theory of autoencoder and support vector machine (SVM) classifier, introduces the complete pipeline for classification which includes preprocessing, autoencoder, SVM and performance evaluation. In addition to normal autoencoder, the idea of class-specific autoencoder will be brought up and implemented. Essentially, the performance of three schemes are compared in detail, which covers SVM without autoencoder, SVM with normal autoencoder and SVM with class-specific autoencoder. The performance results contain classification Accuracy and AUC with various cross-validation methods by testing stroke datasets from lab simulation in Medfield Diagnostics and other external datasets such as CIFAR-10 and MNIST.

Keywords: stroke, microwave technique, classification, support vector machine, autoencoder, class-specific autoencoder

Acknowledgement

Successful work would never be carried out alone especially when I'm fresh to field I stepped into. So I'd like to express my gratitude to my supervisor and examiner Professor Tomas McKelvey for bringing me involved in this project and detailed technical guidance. I'd like to thank my supervisor Stefan Candefjord for kindest support and guidance. I'd appreciate both your time to my thesis, the tolerance to my faults, without which I could not make progress and finish my thesis.

I'd like to thank you all in Medfield Diagnostics for providing a cozy working environment and kind support. And I'd like to thank my parents for their funding and support to my study. Additional thank to the owner of CIFAR-10, MNIST and Wine Datasets. Time flies and six months are just like blinking eyes and I did learn much from this project. The completion of the thesis also means the end of my two-year study in Chalmers, which is a valuable experience I will cherish forever.

Yimeng Hou

Nomenclature

AUC	Area Under Curve, specifically, the area under ROC
AE	Autoencoder, one of the feature learning algorithms
ANN	Artificial Neural Network, one of the multi-layer learning network
BP	BackPropagation, an algorithm for calculating partial derivatives
CSAE	Class-Specific Autoencoder, distinguish from NAE
CIFAR-10	An open image dataset for machine learning
CT	X-ray Computed Tomography, medical imaging technique
FN	True Negative, one of the classification outcomes
FP	True Positive, one of the classification outcomes
L-BFGS	Limited-memory Broyden–Fletcher–Goldfarb–Shanno method, one method used for solving optimization problem.
MNIST	An open image dataset for machine learning
MRI	Magnetic Resonance Imaging, medical imaging technique
NAE	Normal Autoencoder, distinguish from CSAE
NCG	Nonlinear Conjugate Gradient, a method used for solving optimization problem.
numTr	A variable which is short for the number of training examples
p	Number of neurons in the hidden layer in autoencoder
ROC	Receiver Operation Curve, a 2D graph showing classification performance
SVM	Support Vector Machine, one of the discriminative classifiers
TN	True Negative, one of the classification outcomes
TP	True Positive, one of the classification outcomes
λ	Regularization term in the cost function of autoencoder

Contents

1 Introduction

1.1 What is Stroke?

Stroke, or precisely brain stroke, occurs when blood supply is impaired in human brain. It is officially defined by World Health Organization as neurological deficit of cerebrovascular cause that persists beyond 24 hours or is interrupted by death within 24 hours. Currently, it is the second-most common cause of death. 6.7 million people died of stroke in 2012, which is increased by 1 million comparing to 2000 [1]. Although extensive clinical effort has been made in the prevention and treatment of stroke during the past decades, there is still a long road to explore solutions for decreasing the mortality caused by stroke.

Pathologically, strokes are classified into 2 types: Ischemic stroke (IS) and Hemorrhagic stroke (HS). IS occurs when brain cells suffer from the shortage of blood supply, which may be resulted from the prevention of the normal flow of blood by clots. HS may result from the rupture of vessels. The bleeding may cause rising of intracranial pressure, inflammation or shortage of blood supply for brain cells [2].

As a solution, clots dissolving treatment is typically applied to IS patients within a time window of several hours from the onsets of the symptoms. However, this procedure is not applicable to HS patients since it may be life-threatening. To suit the remedy according to the case, it is important to identify whether the brain bleeds before any treatments are initiated. Nowadays, one common procedure to differentiate the types of stroke relies on CT or MRI medical tomography technique, which is expensive, immobile and not widely and full-time available.

1.2 Solution from Medfield Diagnostics

A microwave-based diagnostic solution is developed in close collaboration between Medfield Diagnostics AB and Chalmers University of Technology. The aim of the solution is to provide stroke diagnosis in prehospital scenarios, which is implemented by a portable diagnostic instrument named *Strokefinder* [1]. A picture of Strokefinder can be found in Figure 1.

Differing from CT and MRI instruments which are based on penetration and magnetic resonance respectively, Strokefinder uses microwaves to discover valuable information in brain matters. It is implemented by microwave antennas around the target area. During the measurement stage, some antennas will transmit the microwave signal in turns while others will receive the response signals. A data matrix contains measurements on various channels and frequencies will be acquired [2]. With some signal processing and machine learning algorithms, it is already possible to diagnose the types of stroke with high accuracy based on the data collected by Strokefinder [3][4]. The current method employed by Medfield Diagnostics are direct classification without any intermediate feature learning step [5], which already yielded good performance.

[1] Common prehospital scenarios may include ambulance, clinic and emergency room etc.

[2] A channel represents an arrangement of transmitting or receiving status of the antennas.

Figure 1: *A picture of Strokefinder MD100. It's a headrest stroke diagnostic instrument where measurements are performed in a supine position by eight broadband microwave antennas with various combination of channels.*

1.3 Assumption

In fact, the raw data from microwave measurement contains high-dimensional features from which bleeding information may not be easily identified by a classifier. If appropriate features learning or dimension-reduction methods could be applied before the classifier, more discriminative features may be generated and used for classification. Based on this assumption, we will introduce another category of machine learning algorithms: feature learning, as an additional step of the ordinary data processing procedure. The entire pipeline will include data preprocessing, feature learning, classifier and performance evaluation, which is shown in Figure 2.

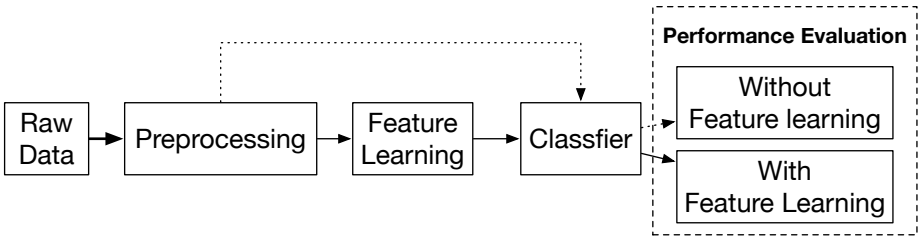

Figure 2: *The system block diagram. The solid lines represent the work flow with feature learning while dashed lines represent that without feature learning.*

1.4 Purpose and goal

The purpose of the thesis is to evaluate one specific feature learning method: *autoencoder* and to verify whether adding autoencoder can improve the classi-

fication performance. Specifically, the goals of the thesis are:

- Explore appropriate preprocessing methods for raw data from Strokefinder.
- Implement normal autoencoder and class-specific autoencoder
- Fit learned features into a support vector machine classifier.
- Evaluate this methodology on data from laboratory measurements of stroke by comparing the performance between with and without applying autoencoder.
- Evaluate this methodology on other external datasets.

1.5 Scope

For the purpose of convenient research, the implementations are carried out in Matlab, as it covers a collection of toolboxes for various developed algorithms. Specifically, Statistics and Machine Learning Toolbox gives efficient implementation of SVM classifiers and Cross-validation method. An external Matlab Toolbox named *minFunc* is invoked as an efficient solution of optimization [6]. The autoencoder is implemented manually based on the starter code from Stanford University [7].

An autoencoder may contain one or several hidden layers. While autoencoders with different number of hidden layers may work in similar method, the one with several hidden layers will require an additional pre-training step as prerequisite to work efficiently. However, the investigation of pre-training need considerable time and effort and cannot be covered in this project due to limited time.

Support vector machine was used as the only classifier as other classifier available from Medfield Diagnostics cannot yield good results due to unexpected reasons.

2 Theory

2.1 Preprocessing

Preprocessing is an initial data processing stage which is applied on raw data to improve the efficiency and ease of subsequent learning process. Typical preprocessing methods usually includes data cleaning, integration, reduction and transformation [8].

The task of data cleaning and integration is usually data correction, denoise, dealing with missing value,etc., which is used to maintain the integrity and consistency of the data. For numerical data containing features from different examples, the data is preferred to be transformed into a standard 2-D real matrix in which each column presents an attribute while each row presents an example. Data reduction is used to reduce the volume of data, hence yield a simplified representation, with the purpose of improving the efficiency of processing data. Data transformation is always used to reinforce the quality of data or change the form of data so that the pattern inside data may be easier to discover and features are more discriminative. In many cases, data transformation captures particular interest since the given dataset may be already in good condition.

Commonly-used methods in data transformation includes mean reduction, standardization and rescaling [9]. *Mean reduction* simply calculates the mean of each attribute and subtracts the entire matrix by the mean vector, μ. *Standardization* performs similarly but divide the data matrix by the the standard deviation, σ. The data matrix will become a zero-mean, unit-variance matrix across each training example. The intuition behind these two steps is to equalize the weights of different attributes to the predictions and to avoid some particular features dominating the results. Rescaling typically map the numerical range of original data from $\mathbb{R}$ into either $[0, 1]$ or $[-1, 1]$ subject to the requirement of subsequent algorithm.

The rule for customising preprocessing methods for specific learning algorithm are flexible. The quality of preprocessing methods are often evaluated by end performance. For example, in classification problem, good preprocessing methods is the one that gives lowest classification error. Note that it is essential to maintain the consistency of ways of preprocessing training and testing data, i.e. apply same mean, sigma and rescaling factor calculated from training dataset to testing dataset. Otherwise, there will be a apparent degradation of the classification performance [10].

2.2 Autoencoder

Feature learning is a process of selecting, converting or combining known features from one representation into another, usually with the purpose of dimension reduction. It may be jointly used with preprocessing to provide subsequent classifier with enhanced features, since part of effort in machine learning task always goes into the design of representation of the data [11]. However, human engineering of feature design is time-consuming and prone to mistakes. Although the features generated by automatic procedures may be difficult to interpret by human, it may yield surprising good learning results. *Autoencoder*(AE) is one of those automatic procedures. It is a reconstruction-based unsupervised learning algorithm which is constructed based on layered network,

where the learning process can be performed layer-by-layer. Given the data in the input layer, an AE encodes it into intermediate representation in the hidden layers and try to reconstruct the original data in the output layer. It's an iterative process in which the reconstruction quality would be improved with more iterations.

A typical AE usually has an hourglass structure: $n < p$ where p and n are the number of units in the hidden and output layer, respectively. This AE will learn a compressed representation of the input data. However, the case when $n > p$ may be of interest. In this case, the AE will enlarge the feature spaces from $\mathbb{R}^n$ to $\mathbb{R}^p$, which may be applicable to cases when the original feature space is small. No matter which case it is, the AE will automatically find patterns or higher-level representations of the data.

Due to randomly initialized weights and the numerical accuracy of the optimization algorithm, the autoencoder will learn uncertain output features given certain input features each time it run, which means it's a high-variability model.

Although AE is a multi-layer learning network, it has a fundamental difference from traditional artificial neural network (ANN): It cannot be used for regression or classification directly in the output layer comparing to ANN since the output layer is merely used for evaluate how well an AE is trained and is not actual output of interest [12].

2.2.1 Structure

AE is formulated by multi-layer network. Each layer contains several neurons, which are the basic arithmetic units. Except the output layer, each layer also contains an bias unit regarded as an intercept term for calculation. The structure of AE is shown in Figure 3.

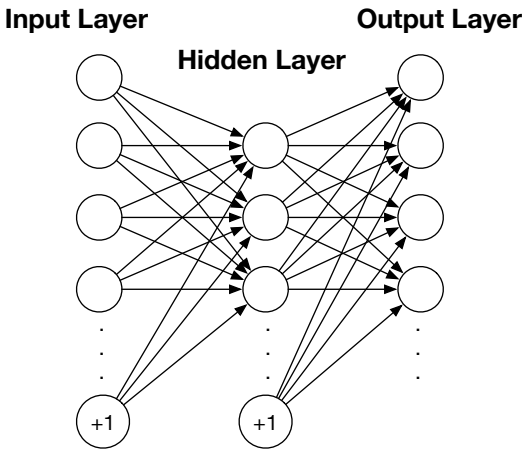

Figure 3: *fully connected AE with three layers. In each layer, several neurons are deployed, which is denoted by a blank circle. The circles with +1 inside are bias units. The arrows indicates the connectivity of units and data flows. Data is processed in one direction from the input to the output layer.*

The calculation of an AE is proceeded layer-by-layer from the input to the

output layer. The value of each neurons in layer $l+1$ is the linear combination from neurons from layer l. A mathematical representation is shown in Equation 2. M and N are the number of units in layer l and $l+1$, respectively. W_{ij} denotes the weight for the connection from i^{th} units in Layer L to j^{th} units in Layer L+1, the superscript l denotes the l^{th} layer.

$$z_i^{(l+1)} = \sum_{j=1}^{M} W_{ij} x_j^{(l)} + b_i^{(l)} \tag{1}$$

$$a_i = f(z_i) \tag{2}$$

$$\text{where } i = 1, 2, ..., N$$

The values z after linear combination will be mapped non-linearly into a narrower numerical range to calculate the activations denoted by a. Common choice for the mapping function are sigmoid function and hyperbolic tangent function, which map the input data into $\mathbb{R} \in [0, 1]$ and $[-1, 1]$, respectively. The selection of the appropriate mapping functions is based on the numerical range of input data. For example, non-negative data should particularly choose sigmoid function while real-valued data may choose tanh function.

$$f(z) = \frac{1}{1+e^{-z}} \qquad \text{Sigmoid Function}$$

$$f(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}} \qquad \text{Hyperbolic Tangent Function}$$

The layer-by-layer, feed-forward process will continue until the output layer is reached. The hypothesis of reconstructed data $h(x)$ is shown in Equation 3, where L is the number of layers in AE and each a_j is a column vector including all training examples,

$$h(x_j) = a_j^{(L)} \tag{3}$$

2.2.2 Training

It seems obvious that the weights of the AE regarding to data is unknown before training. Therefore, applying feed-forward process once is not expected to get the exactly perfect reconstruction in the output layer. In fact, the strategy of training AE is to iteratively adjust the weights so that the reconstruction approaches optimum gradually. To quantitatively model the difference between the input data and reconstruction data, a quadratic loss function is used, which is denoted by J_{direct} and is shown in Equation 4. Note that K is the number of training examples in the dataset.

$$J_{direct}(W, b) = \frac{1}{2K} \sum_{k=1}^{K} \left\| h(x_k) - x_k \right\|^2 \tag{4}$$

However, numerous solutions to the loss function may exist. To simplify the model and prevent overfitting, regularization should be applied into the model [13]. In an AE, it is implemented by adding an extra term called weight penalty, which is denoted by $J_{weights}$ and is shown in Equation 5. It is calculated by summing up all the squared weights in the AE. λ is a coefficient for weight

penalty which controls the level of regularization. When $\lambda = 0$, the AE work perfectly to current data but poorly when generalizing to unknown data.

$$J_{weight}(W,b) = \frac{\lambda}{2}\sum_{l=1}^{L}\sum_{i=1}^{M}\sum_{j=1}^{N}(W_{ij}^{l})^2 \tag{5}$$

The total cost function can be presented as:

$$J_{total}(W,b) = J_{direct}(W,b) + J_{weight}(W,b) \tag{6}$$

Now the problem can be viewed as minimizing a nonlinear function J with respect to the weights W and b. There are plenty of optimization methods to perform this task. *Nonlinear Conjugate Gradient* (NCG) is one of them for solving multivariate nonlinear equation. It has advantages over other methods like L-BFGS especially for high-dimensional training data [14]. The operation of NCG requires two set of parameters in each iteration: function value and the gradients of variables. Specifically, given a function $f(x)$ with n variables to minimize, NCG works in the following way:

In the first iteration $n = 1$, use steepest descent direction as searching direction $d_1 = -g_1$ and use line search to find best step size α and update the variable value.

For the following $n \geq 2$ iterations, it will [15][16]

1. Calculate β using Hestenes-Stiefel method $\beta_n = \frac{g_n^T(g_n - g_{n-1})}{(g_n - g_{n-1})^T d_{n-1}}$
2. Update conjugate direction $d_{n+1} = -g_n + \beta_n d_n$
3. Line-searching to determine best updating step size α, $\alpha > 0$
4. Update variable $x_{n+1} = x_n + \alpha d_n$

However, Hestenes-Stiefel method is gradient-based optimization method which need information for the gradient of each weights in each iteration, which is mathematically presented as:

$$\nabla J(W,b) = \{\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b); \frac{\partial}{\partial b^{(l)}}J(W,b)\} \tag{7}$$

Since the input dataset contains numerous training examples, the overall partial derivatives of each weight are the sum-averaged ones from each examples plus the weight penalty term, which is shown in Equation 8 and 9. Suppose that the data contains features from K training examples.

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b) = \frac{1}{K}\sum_{k=1}^{K}\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b)_k + \lambda W_{ij}^{(l)} \tag{8}$$

$$\frac{\partial}{\partial b^{(l)}}J(W,b) = \frac{1}{K}\sum_{k=1}^{K}\frac{\partial}{\partial b^{(l)}}J(W,b)_k \tag{9}$$

Instead of other direct mathematical methods , *backpropagation*(BP) is commonly used as an efficient calculation of those partial derivatives [17]. As required in BP, a deviation term in each unit will be calculated. In each iteration,

the algorithm will calculate the deviation between theoretical output values (i.e. input values) and observed values for each units. The deviation will be allocated backward to the units in the former layer based on the weights. This operation will continue until it reaches second layer [3].

Note that, for learning network with more than one hidden layer, the algorithms may not work well [18]. This is because the gradients for former layers may be quantitatively too small to tune the weights efficiently and is likely to stuck in the local minimas [19]. As a solution, pre-training is usually introduced before AE to provide a initial estimation of the weights [20]. However, pre-training is unnecessary in our project since the AE used has only one hidden layer.

By definition, the error terms for the output layer and other layers can be calculated in Equation 10 and 11, respectively. Note that $f'(z)$ is the derivative of nonlinear mapping function $f(z)$ and $\bullet$ notation denotes element-wise multiplication [13][21].

$$\delta^{(L)} = -[h(x) - a] \bullet f'(z^{(l)}) \tag{10}$$

$$\delta^{(l)} = W^T \delta^{(l+1)} \bullet f'(z^{(l)}) \tag{11}$$

Known the error terms of each units, the gradients of each example k can be calculated via Equation 12 and 13

$$\sum_{k=1}^{K} \frac{\partial}{\partial W_{ij}^{(l)}} J(W,b)_k = \delta_i^{(l+1)} (a_j^{(l)})^T \tag{12}$$

$$\sum_{k=1}^{K} \frac{\partial}{\partial b^{(l)}} J(W,b)_k = \delta_i^{(l+1)} \tag{13}$$

Substitute Equation 12 and 13 to Equation 8 and 9, all information is known to proceed iterative training process.

A overview of optimization process is shown in Figure 4. To begin with, the weights W and b will be assigned with random values close to zero, which is a common way to initialize variables. In each iteration of the entire process, the AE will calculate the cost and gradients by feedforward and backpropagation process, respectively. These process will be iteratively performed until the gradients are quantitatively too small or maximum number of iterations is reached. Although it does not guarantee to find the global minimum, previous researches indicate the reconstructed error is already low enough for the AE which contains less than 2 hidden layers [18].

2.3 Classification

2.3.1 Introduction

Classification is one of the most common tasks in statistics and machine learning. It discriminates the examples from different classes based on the attributes they

[3]The input layer and output layer are regarded as first and last layer, respectively. The sequence of other layers are in a ascending order from the input layer to output layer.

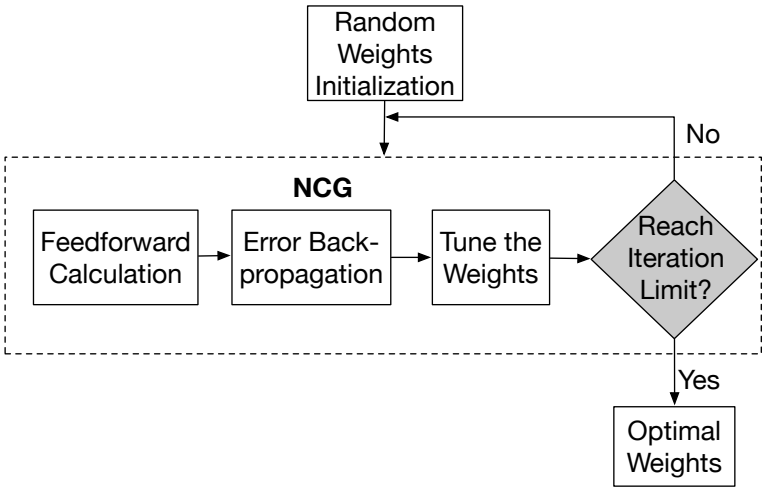

Figure 4: *A block diagram of the entire optimization process. The optimization starts from random weights initialization and ends up until it reaches iteration limits.*

own [4]. It is usually performed in a supervised learning way in which the classifier assigns classes to unlabeled examples based on information of given examples. Suppose each example has a vector of n attributes $\{x_1, x_2, x_3, ..., x_n\}$ and a label y indicating which class it belongs to regarding to the specific dimension we concern. To perform supervised classification, this known dataset will be used as training data for building classification models. When new unlabeled data (also known as testing data) is coming, the classifier will assign labels to the testing examples based on the rules previously defined. A schematic view of supervised classification procedure is shown in Figure 5.

Another type of classification is unsupervised learning. It refer to the scenario where no label knowledge for each example is available. Therefore, the algorithm is trying to 'discover' instead of 'learn' the pattern. The classification is usually performed by clustering analysis which is based on modeling the distribution of examples in multi-dimensional feature space. The examples which belong to the same cluster may share similarity in particular aspects. Unsupervised classification is popular in applications such as Business intelligence, user behavior analysis,etc.

2.3.2 Performance evaluation

The performance of a classifier is a measurement of how well it performs the classification task. A good classifier should predict the labels accurately with preferably little time (Some application requires real-time processing). Consider binary classification problems in which labels of the examples $y \in \{+1, -1\}$. Correct prediction means the predicted label of an example by the classifier $\hat{y}$ is identical with the true label y.

[4]The terms: attributes, features or predictors have the same meaning and are interchangeable in this thesis.

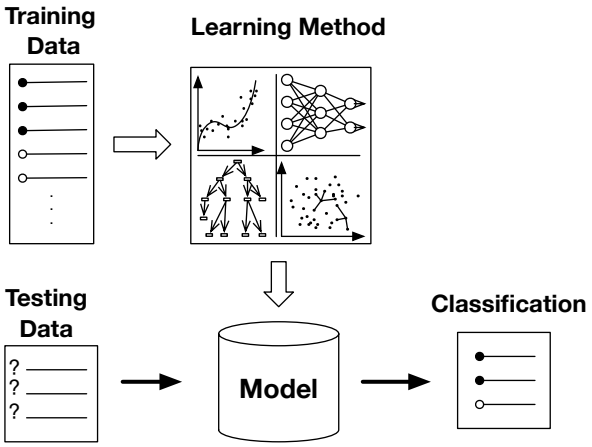

Figure 5: *General supervised classification process. It 'teaches' a model with appropriate learning methods based on the training examples so that the model could predict the label of unknown data.*

$$y = \hat{y} \qquad \text{True}$$

$$y \neq \hat{y} \qquad \text{False}$$

Specifically, there are 4 possible outcomes.

$$\hat{y} = +1 | y = +1 \qquad \text{True Positive (TP)}$$

$$\hat{y} = -1 | y = -1 \qquad \text{True Negative (TN)}$$

$$\hat{y} = +1 | y = -1 \qquad \text{False Positive (FP)}$$

$$\hat{y} = -1 | y = +1 \qquad \text{False Negative (FN)}$$

After classification, the counts in each scenario will be collected and presented in a Confusion Matrix, which is the basis for further study of the classification. The most commonly used methods in performance evaluation is *Accuracy* and *ROC*. Accuracy is an intuitively straightforward evaluation of classification performance. It is a ratio of correct number of classification over total number of classification, which is shown mathematically in Equation 14.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

Receiver Operating Curve (ROC) is a graphical tool of evaluating binary classification results. Each point in the graph is one classification outcome. For classifiers which yield continuous classification results, it shows a smooth curve generated by varying discrimination threshold. For discrete classifier, stair-like curve is plotted based on the posterior probability of each class [5].

[5]Posterior probability is the conditional probability of Event B happens after Event A is observed, which can be described mathematically as $P(B = b_i | A = a)$

The horizontal and vertical axises are the False Positive Rate and True Positive Rate, respectively. The optimal classification point lies in $(0, 1)$ where all the positive labels are classified correctly with no errors. Comparing to ACC which sometimes provides evaluation, ROC offers more comprehensive information about the classification.

Area Under Curve (AUC) is a scalar metric of evaluating classification performance. It is calculated by summing up or integrating the area under ROC. Similar with ACC, high AUC value means high overall classification performance.

2.3.3 A binary classification example

Suppose that two types of wine were to be classified by several tasters from a big, messy cellar where it contains a number of wine with different labels. To be efficient and novel, they decided to use machine learning algorithms to perform the task. To start with, they brought some basic chemical equipments to analyze and extract attributes from each label of wine. They set up 13 attributes they want to test on each bottle of wine. For examples, the content of alcohol, acid, magnesium or phenols, the color intensity, etc. [22]. After they have applied this test on wine with all existing labels, it is easy to classify the wines only by identifying the labels of the rest of wines.

They regularized all the information in standard data form. In this case, the size of the data matrix is 100*13, each row in the matrix corresponds to an *example* of wine while each column corresponds to an *attribute*. They carefully cleaned, smoothed and normalized the data to ensure the *quality* of the dataset. This step is known as *preprocessing*, which is an indispensable part of classification procedure. Since they were uncertain about how well the classifier would work. They randomly set aside 20 examples from 110 total examples as the *testing dataset* to evaluate their method. This step is known as *cross-validation*, which is usually combined with classification performance to evaluate a model. After they applied some classification methods, a graphical view of the classification process is shown in Figure 6.

The method they used in the wine classification problem constructed a boundary for separating each class space, which is termed a *discriminative* model. After they applied this model to testing dataset and varying the threshold of separation [6], the corresponding ROC is shown in Figure 7.

2.3.4 Support vector machine

Support Vector Machine(SVM) is one of the discriminative models for supervised classification problem. It constructs a hyperplane in multi-dimensional feature space and separates data of different labels. Given training data in which labels are specified, SVM will firstly construct a model where it is trained and classification specifics such as decision boundaries, kernels, number of features are defined. When new observations is coming, they are classified based on which side of hyperplane they belong to.

For the case when two classes of examples are separable by a hyperplane, SVM finds the optimal hyperplane that yield largest margins between any instances of different classes [23]. This hyperplane is defined in Equation 15 and

[6] Varying the threshold can yield different confusion matrices.

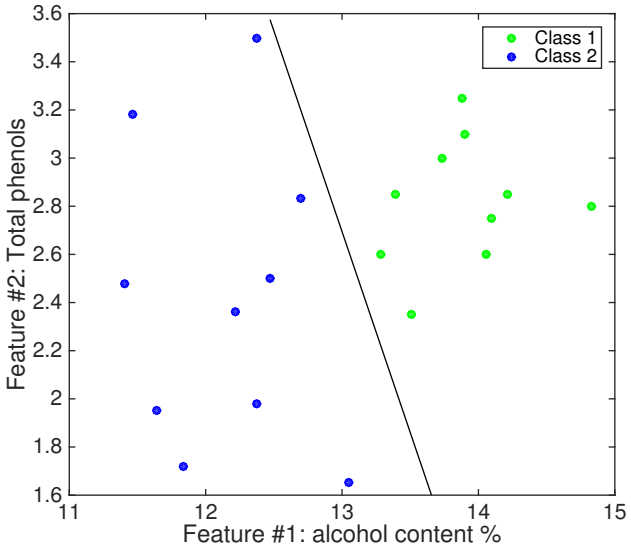

Figure 6: *A 2-D graph showing the classification process, Two random features from 13 features are visualized.*

16.

$$h(x) = x^T\theta + \theta_0 = 0 \tag{15}$$

$$\gamma^{(i)} = y^{(i)} sgn(h(x)) \tag{16}$$

where θ is the vector from hyperplane to each example x, β_0 is intercept term. The margin of each example to the hyperplane is presented as $\gamma^{(i)}$. The optimization problem can be represented in Equation 17.

$$arg \max_{\gamma,\theta,\theta_0} \gamma \tag{17}$$

In case the hyperplane may not fully separate data, SVM grands certain tolerance to false classification which can be presented in Equation 18 [24]. A schematic view of a SVM shows in Figure 8.

$$y^{(i)} sgn(h(x)) \geq \gamma(1-\xi) \tag{18}$$

$$\text{subject to } \textstyle\sum_i^K \xi^{(i)} \leq C$$

Kernel methods are often applied to generate more features. They use inner-product operations which map original features to higher dimensional feature space. Suppose each original example i contains a vector of features $\{x_1, x_2, ..., x_m\}$. After applied kernel method, each example will have $\{\hat{x_1}, \hat{x_2}, ..., \hat{x_n}\}$ features, where $m < n$. There are several commonly used kernel functions, which is shown below:

- Linear kernel : $K(x, x') = \langle x, x' \rangle + c$
- Gaussian-RBF kernel : $K(x, x') = exp(-\gamma \|x - x'\|^2)$

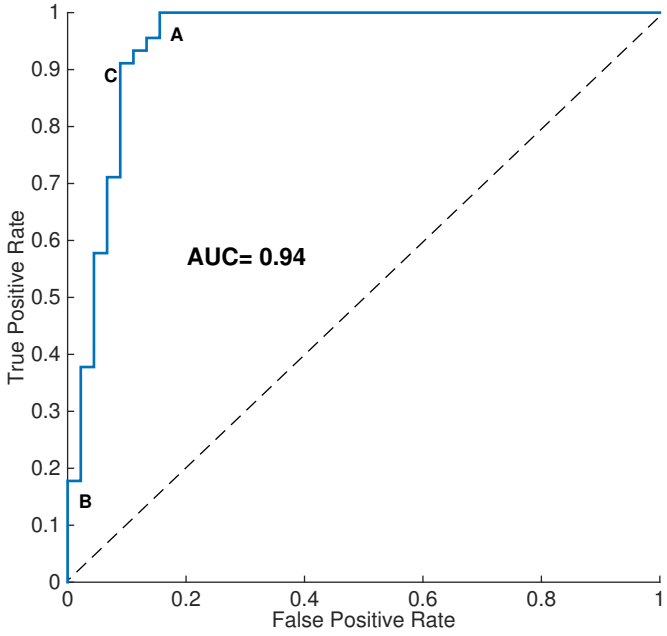

Figure 7: *The ROC of of wine classification. Each discrete point means a observation of the classification. Point A indicates that all positive class examples are correctly classified while point B indicates the opposite. Point C is the observation which has optimal accuracy since it has minimum euclidean distance to point (0,1).*

- i-th degree polynomial kernel : $K(x, x') = exp(r + \gamma\langle x, x'\rangle)^i$

The RBF kernel is suitable to more applications than polynomial kernel and it has fewer numerical problem [10]. SVM with linear kernel may not get as good results as with RBF kernel, but it is computationally efficient and is especially applicable to the case when the number of features are much greater than the number of training examples.

2.4 Cross-validation

Cross-validation (CV) can be viewed a statistical method of measuring the performance of a predictive model, i.e. how well current model fits new data. It divides known data into two independent subsets: training set and validation set. The former one is used for building model for specific tasks while the latter one is used to evaluate the performance of that model. Since the performance of a parametric model always depends on the parameters, cross-validation is usually performed with the purpose of selecting optimal parameters for current model. It is an important idea that is widely applied in model-based learning methods.

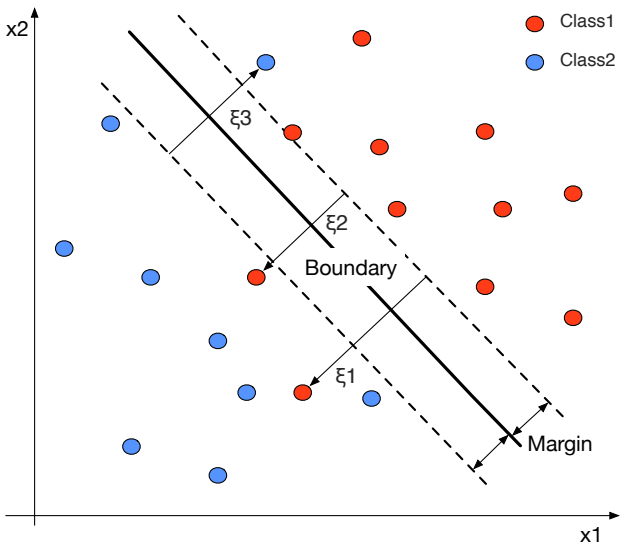

Figure 8: *A classification example by SVM classifier in Two-dimensional feature space. Each circle means a training example in SVM. The examples from two different classes are separated by the boundary. SVM grants certain tolerance for false classification, i.e. allow some examples in incorrect side of the hyperplane. The cost for false classification is numerically denoted by* ξ_i.

2.4.1 Leave-p-out

Suppose the dataset has n total examples, *leave-p-out cross-validation* method picks p $(0 < p < n; n, p \in \mathbb{Z})$ examples as training data while the rest as testing data. It is an exhaustive and unbiased validation method as it places all possible combination of distributing the dataset. However, the computational cost is not certainly beneficial to the evaluation of a model [25]. Therefore, it may not be considered the best method as it apparently seems.

2.4.2 k-Fold

k-Fold Cross-validation method randomly divides the dataset into k exclusive partitions with approximate equal size. In each fold, one partition of the data will be the testing dataset while other partitions together act as training dataset. The entire validation process will be performed k times until each partition has used as testing dataset. In this case, each example in the dataset will be used for prediction exactly once.

2.4.3 Random sub-sampling

This method randomly divides dataset into training and testing subsets, each subset contains certain number of examples. To lower the bias, the random sub-sampling will always be proceeded several times. In this case, some examples are probable to be used as testing data several times whereas some may never been used.

2.4.4 Stratified sampling

Since data contains examples from different classes, the way that randomly divide the data sets may yield unbalanced number of training examples from different classes in a dataset, which may bias and degrade the performance of the model. *Stratified sampling* is a method of dividing training data set in which it ensures approximately equal number of examples from each class. It is worth mention that all the datasets in this project are divided in a stratified manner.

3 Method

3.1 Preprocessing

Since AE and SVM cannot handle complex number as input data, each element in data matrix will be took absolute value. After converted into real matrix, the new features from each observation will be unrolled into a vector. The entire matrix will be normalized and rescaled into numerical range of $\mathbb{R} \in [-1, 1]$ without which AE will be hard to converge in given iterations. A schematic view of complete procedures of preprocessing is shown in Figure 9.

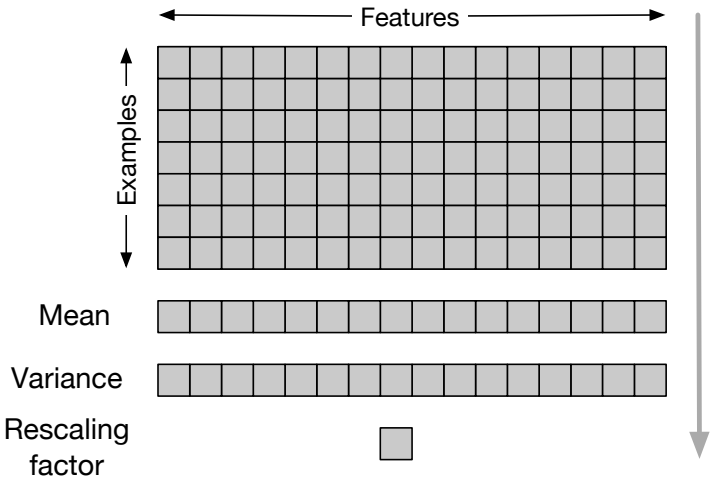

Figure 9: *A approximately graphical view of preprocessing. Small rectangulars are elements in data matrix. The mean and variance are vectors while rescaling factor is a scalar.*

After several trials with different combinations of different preprocessing methods, the result shows that no promising classification results can be produced from the entire huge matrix, i.e. either ACC or AUC is approximate to 0.5. Except for the poor performance, several other reasons why giving up using the entire matrix can be concluded below:

- High time expense and memory pressure for calculation.
- High dimensional data makes it hard for AE to learn useful features.
- Overfitting is more possible to occur.
- The optimization process is hard to converge.

As a result, feature selection is necessary during pre-processing step. This step relies on exhaustively conduct performance evaluation until performance rises upon a acceptable threshold [26]. With some suggestions from Medfield Diagnostics, the data measured by side antennas in low frequency band is manually selected. A schematic view of feature selection is shown in Figure 10.

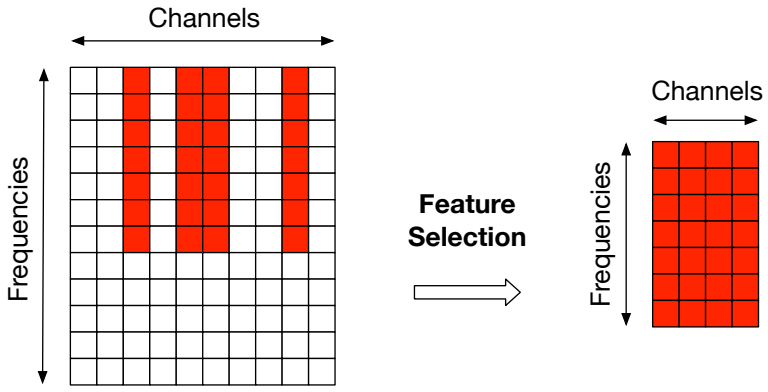

Figure 10: *Feature selection. Note that the left and right rectangulars all refer to the same observation, i.e each observation has a frequency-by-channel matrix.*

3.2 Normal and Class-Specific AE

Two method of using AE are proposed: Normal AE (NAE) and Class-Specific AE (CSAE). Given a dataset with examples from multiple classes, a NAE will learn a representation based on all the examples regardless of their labels. In that case, the weights learned by AE are universal to multiple classes. It may be problematic sometimes since the features after AE may not be representative and specific to class. CSAE is one possible solution to multi-class training problem. It essentially splits the training dataset by class and trains each class of training examples with AE individually. For example, for n-class classification problem, the class set $Y \in \{c_1, c_2, c_i, ..., c_n\}$ where c_i denotes class i. After class-specific training, the output set $W \in \{w_1, w_2, w_i, ..., w_n\}$ where w_i denotes a vector of AE weights for Class i. These weights will be substituted into the entire original dataset independently and generate the class-specific features. Finally, these class-specific features will be combined together and used to build classification model. A view of working flow of NAE and CSAE is shown in Figure 11.

3.3 AE model selection

Two important parameters: p and λ can influence the performance of the AE. The choice of selecting p is flexible since there is no hard threshold for reconstruction error when continuously varying p. Detailed discussion is presented in Result Section.

To choose optimal λ, a line-searching method combined with cross-validation will be performed. To begin with, a linearly or Exponentially growing sequences of variables is set up. For example, $\lambda = 0.1^n$, $n = 1, 2, 3, .., 10$. For each λ, a cross-validation process can be applied. To begin with, the training dataset is randomly divided into sub-training set and validation set. Most common strategy is to randomly assign one-third of total examples for sub-training set and the rest for validation set. Concretely, the weights trained by using sub-training data are substituted into validation dataset and the reconstruction error

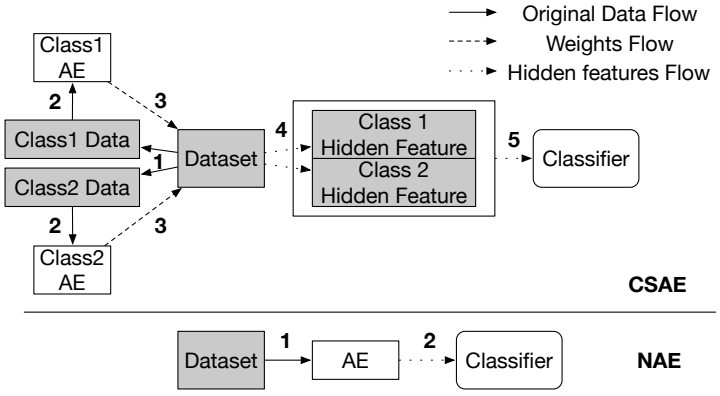

Figure 11: *Block diagram showing the mechanism of NAE and CSAE in binary classification. The procedure always begins with the dataset after preprocessing and ends up with classifier. The figures from* 1 *to* 5 *showing detailed sequence of each part.*

are essentially observed. These errors are the indication of how the weights generalized to exclusive dataset. The optimal λ is the one that yields lowest reconstruction error in the validation set. The AE will be re-trained using all the training examples with that optimal λ.

3.4 SVM model selection

Similar with AE, SVM also relies on parameter searching to get optimal classification performance as the optimal parameters are unknown for a given problem. An interval cross-validation is therefore performed. The specific steps are shown below:

- Apply cross-validation methods to divide training data into sub-training and sub-validation sets
- Grid-searching optimal parameter with lowest cross-validation loss in 3-dimensional parameter spaces [10], i.e. C, γ and kernel function.
- Re-train the SVM model with corresponding optimal parameters
- Predict the labels of testing sets using SVM model.

Figure 12 clearly shows entire procedure of classification.

3.5 Performance evaluation

To evaluate the performance in a comprehensive way, ACC and AUC values are calculated with various parameters in different schemes. A list of parameters to be selected are shown in Table 1. For fair comparison of those three schemes, same training and testing data is used. The performance will be characterised by mean and standard deviation with several times of measurement.

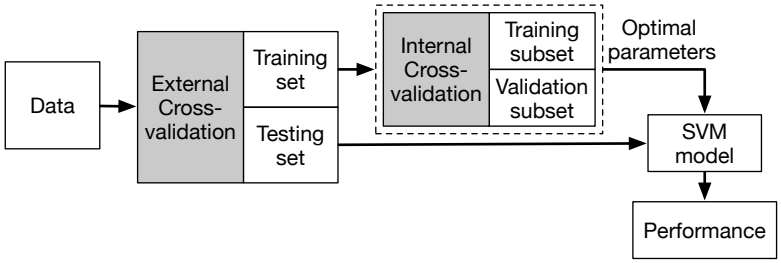

Figure 12: *A schematic view of SVM classification. The data went through cross-validation twice: The external cross-validation is a stragety of evaluation the classification performance with known data. The internal cross-validation is a model-selection process where optimal parameters are found,*

Schemes	Parameters
SVM	numTr, kernel, C, γ
NAE+SVM	numTr, p, λ, kernel, C, γ
CSAE+SVM	numTr, p, λ, kernel, C, γ

Table 1: *Schemes and parameters for testing, numTr is the number of training examples, p is the number of hidden units, λ is weight decay parameter, C and γ are two internal parameters in SVM. A grid-searching method was used to find the optimal parameters.*

4 Results and Discussion

This chapter will cover all the test results based on current methods. Specially, it will start with the performance of AE, which can be described by time and reconstruction error with the number of iteration, training examples and hidden layers as variables.

The results can be divided into two parts: The first part presents the classification performance on external datasets including MNIST and CIFAR-10. The second part shows stroke classification performance on the two datasets from Medfield Diagnostics lab measurement.

The performance will be evaluated using Accuracy, AUC value as metrics. Both mean and standard deviation over several times of tests will be presented.

4.1 Autoencoder

4.1.1 Time with iterations

Two major parts of the model contribute to the total running time. The first part is training a autoencoder, which mainly depends on how many variables (in this case, weights) in the optimization function. Given a data set with m features, n examples and p hidden units, the number of weights W can be calculated in Equation 19.

$$W = 2mp + m + p = (2m + 1)p + m \tag{19}$$

With fixed data set, the time spent grows linearly with more hidden units as it shows in Figure 13.

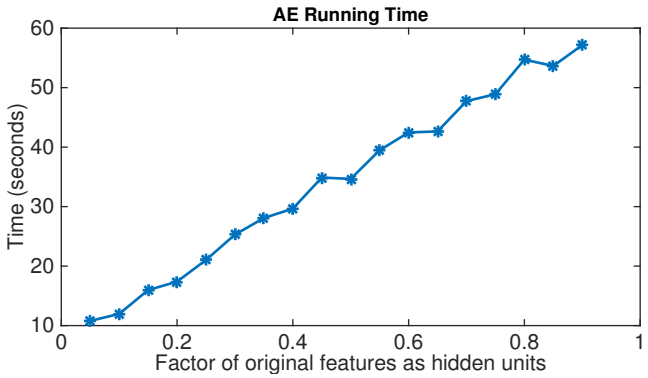

Figure 13: *AE training time with different number of hidden units . The max number of iteration is 400. This test is performed in laptop with a CPU of Intel I5 2.8GHz.*

The second part is interval cross-validation process, which can be complicated since number of original features or training examples, cross-validation folds, ranges of the parameters in line-searching and kernels all contributes to the running time. A general estimation of the system running time shows in Figure 14, which shows the decomposition of the entire system running time. The result is conducted by testing various numbers of training examples and

fixed p. The time cost for running AE grows roughly linearly. It is concluded that AE will help improving the time efficiency by generating less number of refined features for fast classification.

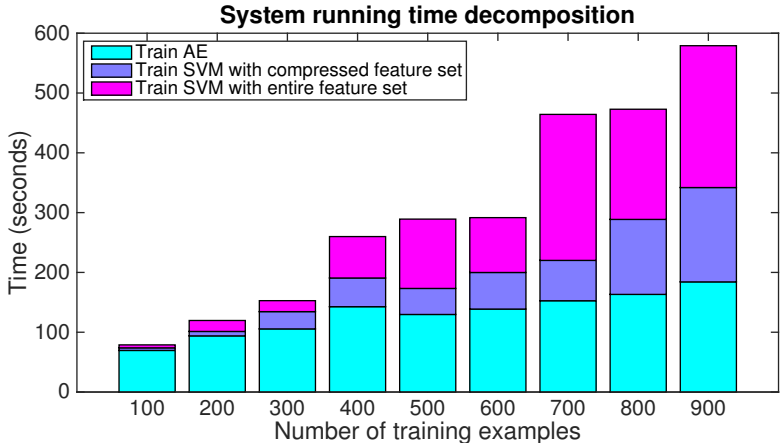

Figure 14: *System running time decomposition with training examples varying from 100 to 900 and with fixed p = 15% of the number of original features, the number of testing example is 100, maximum iteration in AE is 400. A 10-fold internal cross-validation for grid-searching optimal parameters for SVM. The test environment is the same as Figure 13.*

4.1.2 Cost with iterations

The optimization function will return the total cost J_{tot} in each iteration. It's a general metric for evaluating the working performance of the autoencoder. It seems apparent that more iterations mean less errors the autoencoder made in the reconstruction. In addition, other factors such as number of hidden layers or number of train examples will also effect the optimization process of autoencoder.

An example from CIFAR-10 dataset will be used to explain how AE performs with different parameters [27]. The testing data contains 100 training examples with 1024 features in each example. The following figures shows how cost changes with number of hidden units, number of training examples and number of original features as variables.

If too few hidden units is used, the AE would be short of 'basis' to represent the reconstructed data, as it is illustrated in Figure 15 where 5% of the original features as hidden units could not reconstruct the data very well.

According to Equation 4, the cost is averaged over training examples. Therefore, only the number of features influences the cost value. These assumptions can be proved in Figure 15 and Figure 16. In addition, due to high variability of data, no patterns can be concluded from curves in Figure 16.

It is concluded that lowering p may significantly increase the cost of autoencoder, which results in poor reconstruction. Large p is not recommended since the margin benefit for improving the training is trivial while time cost grows rapidly. 10% to 20% of the original features as hidden units would be

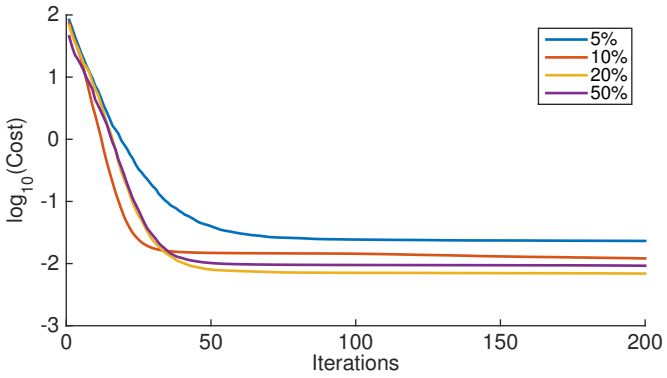

Figure 15: *The cost with different number of hidden units. The legends are the percentage of original features as hidden units. With more hidden units in hidden layer, the reconstruction quality is improving.*

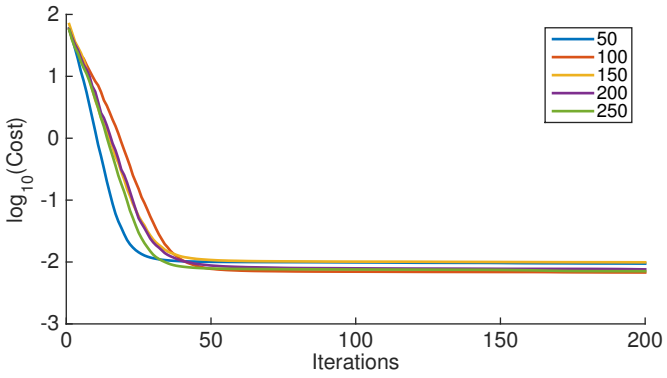

Figure 16: The cost with different number of training examples.

appropriate choice to start testing with.

4.2 Performance on external datasets

4.2.1 MNIST

MNIST dataset contains greyscale images of handwritten digits with different styles, each images composes of 28*28 pixels. The intensity of the pixels are unrolled into a vector and used as features. The dataset includes 2000 images labeled with digit *1* and another 2000 images labeled with digit *2*. Figure 18 shows the estimated cost with different λ. It indicates that $\lambda_{opt} = 0.01$.

Using optimal parameters, four images before and after autoencoder is shown in Figure 19.

After running the three schemes, the performance of this dataset shows in Table 2.

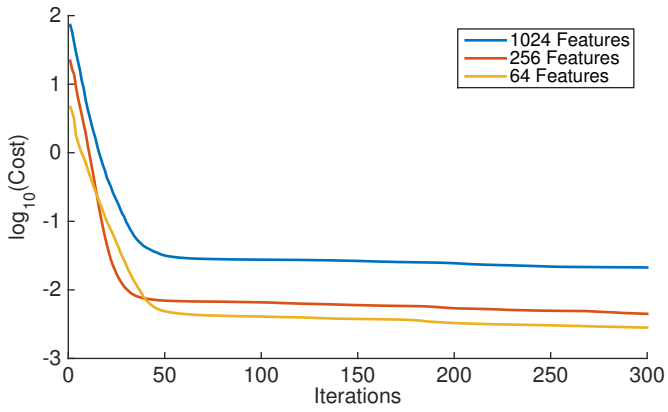

Figure 17: *The cost with different number of features and 50 hidden units. For images, lowering the resoluton of the image can yield less features comparing to original image.*

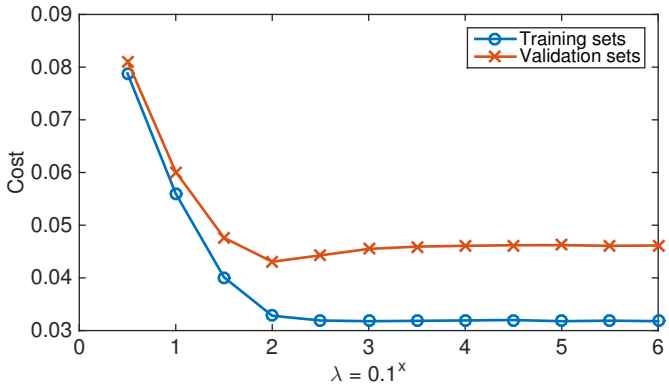

Figure 18: *Finding optimal λ with line-searching method. The results is acquired by training 1000 examples of digit '2' with different λ. The cost is the Mean absolute error (MAE) between input image and reconstructed image.*

Schemes	Accuracy	AUC
SVM	0.9925	0.9986
NAE+SVM	0.9815	0.9990
CSAE+SVM	0.9925	0.9995

Table 2: *Mean ACC and AUC evaluation by using 10-Fold cross-validation. The results are running by 5 times repeatedly and calculated mean value. RBF kernel is used in SVM. The standard deviation is numerically neglectable. A reference ACC from external source is 0.986 [28].*

4.2.2 CIFAR-10

This data set is a collection for $32 * 32$ pixels grey-scale images, with ordinary objects or creatures such as cars and horses. Using a subset of this dataset in the

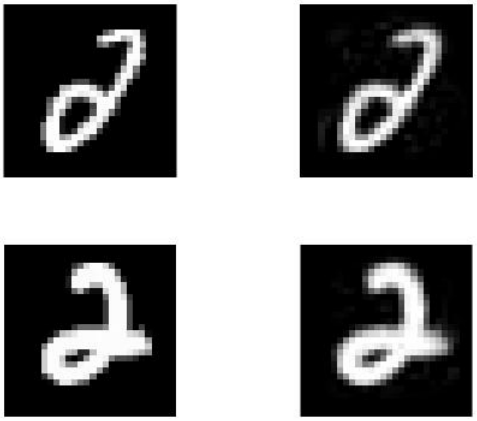

Figure 19: *Two random examples of from MNIST dataset. The left column shows the original images. The right column shows the corresponding reconstructed images by AE.*

tests, the entire data matrix has a size of $1024 * 500$, which means 1024 unrolled pixel intensity value as features and 250 examples in each class. The test also starts with a line-searching for the optimal parameter λ. Figure 20 intuitively shows the original and reconstruction images. Since the image data contains much redundancy (irrelevant objects, patterns or different backgrounds), the autoencoder will learn a noisy representation of the features, which may explain why the autoencoder generalize poorly to other images of same class.

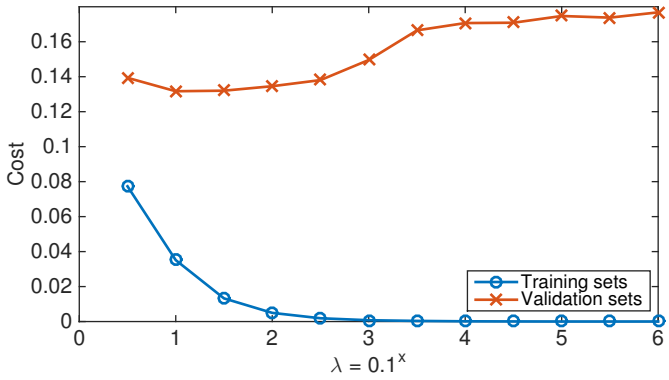

Figure 20: *Finding optimal λ by line-searching method. Optimal C lies in the bottom of 'U' shape curve from validation sets. The cost denotes MAE.*

Figure 21 shows two examples for original and reconstructed images. Figure 22 shows the 2-D errors surface between original and reconstructed data. Both perspectives indicate the autoencoder is well-trained with minor reconstruction errors.

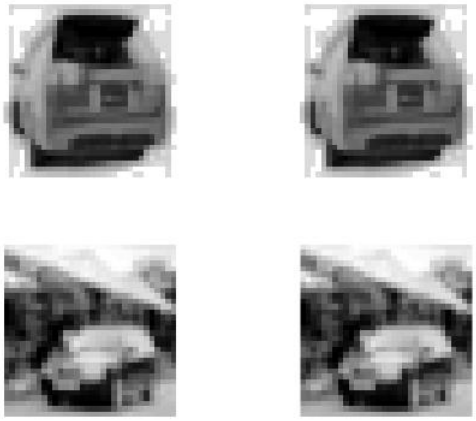

Figure 21: *Two examples on original and reconstructed images. The left column shows the original images while the right colomn shows the reconstructed images by AE. Intuitively, there is no sensible errors for these images.*

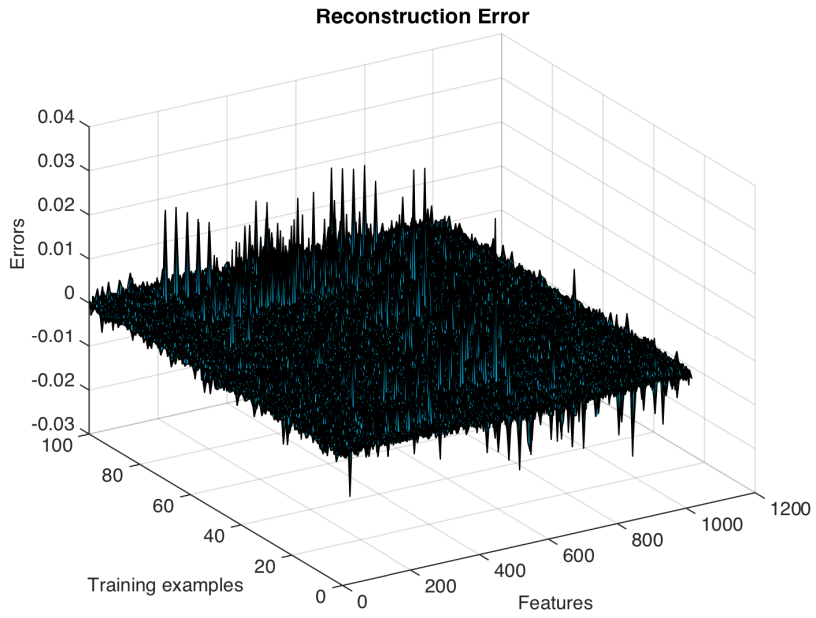

Figure 22: *The errors surface which contains 100 training examples with 1024 features. The features are translated directly from image pixel intensity so that it has a numerical value between 0 and 1. The errors are calculated by taking the subtraction of reconstruction value and original value. The peaks indicate tiny deviations.*

Thereafter, the data with size of 154*500 will be feed into SVM to perform classification. The performance results is shown in Table 3. It shows that SVM with CSAE can achieve equivalent performances.

Schemes	Accuracy	AUC
SVM	0.80	0.90
NAE+SVM	0.78	0.89
CSAE+SVM	0.81	0.88

Table 3: *Mean ACC and AUC evaluation by using 10-Fold cross-validation. The results are acquired by repeatedly running 5 times and calculated mean value. RBF kernel is used in SVM. The standard deviation is numerically negligible.*

4.3 Performance on stroke datasets

4.3.1 AE Optimization

Applied the model selection methods illustrated before, Figure 23 shows the reconstruction errors with different λ, which indicates the optimal $\lambda \in [0, 0.0001]$. The 'L' shape curve of validation data suggests the weights learned by AE will not cause overfitting. Figure 24 intuitively shows the deviation of reconstruction for validation data. It is concluded that AE works generally well except the parts where the numerical scale is very small, as it is seen in the valley of the curves, which may be explained by the limited accuracy of optimization in AE. However, there is no evidence shows this deviation will significantly influence subsequent classification performance.

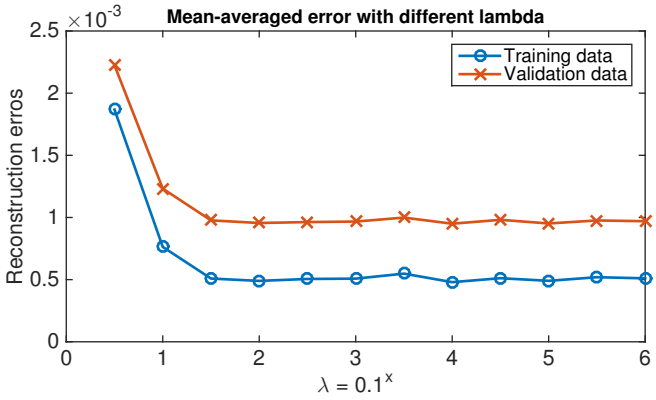

Figure 23: *Search optimal λ by line-searching method. Note the scale of vertical axis.*

4.3.2 SVM Optimization

Tests in early stage of this project show that when applying RBF or Polynomial kernels to SVM, the classifier gives poor results, which only yield $AUC_{max} \approx 0.6$. Possible explanation could be mapping high medical data to higher dimensional feature space would not yield useful features for classification. As a result, linear

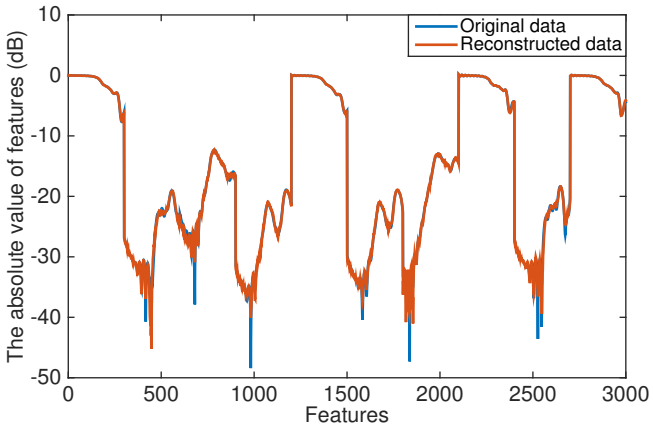

Figure 24: *A comparison between original features and reconstructed features on testing dataset. A random example from the testing dataset is used. Note that the features are the absolute value of original complex value. The vertical axis is in logarithm scale. The test is performed with* $p = 15\%$ *of original features,* $\lambda = 0.0001$ *and* 400 *iterations in AE.*

kernel will be eventually used as kernel function for SVM. The SVM model is constructed by line-searching for optimal C and using automatic kernels scale from Matlab built-in function *fitcsvm*.

4.3.3 Data description

The stroke datasets are 2-D complex number matrices including hundreds of observations measured in various frequency samples and channels. Two datasets are available for testing. They are acquired by artificial bleeding measurements on manikins in Medfield Diagnostics.

Dataset A contains 1000 observations: 500 from brain bleeding patients and 500 from healthy people. Each observation includes features collected from 36 channels and 1000 frequency point spanning from 3MHz to 3GHz.

Dataset B contains 985 sections. Each section is considered as an individual measure interval in which four bleeding sizes can be simulated while other parameters remain unchanged. The bleeding size ranges from no bleeding(healthy) up to 60 mm. The tests are based on the three following scenarios:

B-1 Pick the observations with *largest* bleeding size and no bleeding

B-2 Pick the observations with *smallest* bleeding size and no bleeding.

B-3 Pick the observations with *Random sampled* bleeding size and no bleeding.

4.3.4 Preparation

An overview of parameters and methods used in the tests is listed below:

- Apply feature selection so that each example contains 3000 complex number features from 10 channels 300 frequency points from 1MHz to 1GHz.

- Use the absolute values of the complex numbers as new features, with mean-reduction, standardization and rescaling as preprocessing methods.
- Assign $\lambda = 10^{-5}$, $p = 3000 * 15\%$ and 400 iterations in AE training
- Apply 10-Fold internal cross-validation for searching optimal C parameter, *Linear* kernel in SVM
- Apply 10-Fold and random sub-sampling as external cross-validation for performance evaluation
- Use the mean and standard deviation of Accuracy and AUC values as metrics for performance comparison.

4.3.5 Dataset A

The classification performance by using random sub-sampling based on the settings above shows in Figure 25 and 26. These results show the mean and standard deviation of ACC and AUC acquired by running *30* times repeatedly based on the same parameters. In each time, the medical data is randomly sub-sampled into training and testing sets in a *stratified* way. The training sets contains *variable* training examples whereas the testing set contains *100* testing examples. To make a fair comparison, three schemes are all based on *same training* and *testing examples* each time the data is sampled.

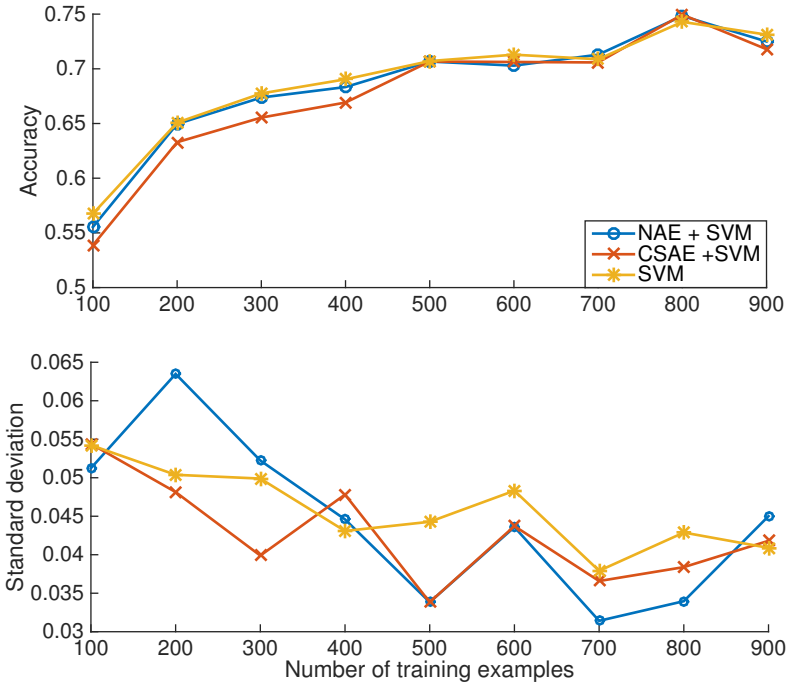

Figure 25: *Mean ACC of Dataset A with corresponding standard deviation.*

The performance using 10-Fold Cross-validation shows in Table 4 and 5 with same parameters as in random sub-sampling. Therefore, it may correspond to

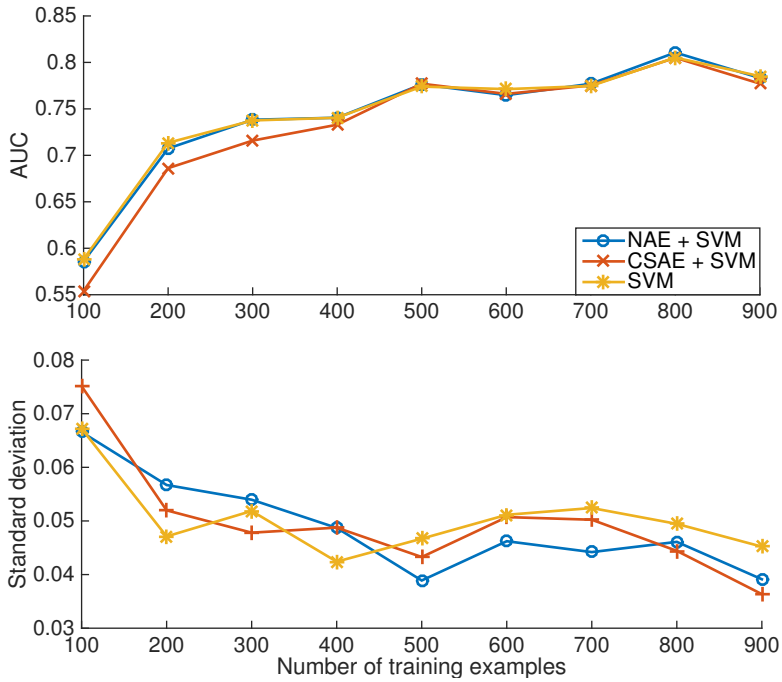

Figure 26: *Mean AUC of Dataset A with corresponding standard deviation.*

results when $numTr = 900$ in Figure 25 and 26. The Accuracy and AUC curves show a ascending trend while the standard deviation curves show a descending trend. And when $numTr = 800$, overfitting occurs. These results are reasonable in machine learning since more training examples should provide better and stable learning performance until overfitting reaches where the performance starts to oscillate or decline. Meanwhile, adding AE before SVM might offer tiny improvement comparing to the case when using SVM solely.

Schemes	Accuarcy	AUC
NAE+SVM	0.73±0.05	0.79±0.05
CSAE+SVM	0.73±0.05	0.79±0.03
SVM	**0.74**±0.06	**0.79**±0.03

Table 4: *A comparison of three schemes by 10-Fold Cross-validation. Each entry shows the mean Accuracy or AUC and its standard deviation by running 5 times repeatedly. The one with highest value is marked by bold font.*

Dataset B

The testing results of Dataset B is acquired by running *20* times repeatedly based on the same parameter setup as it is in Dataset A. The classification results for the three scenarios is shown below.

Schemes	Accuarcy	AUC
NAE+SVM	0.50±0.03	0.50±0.04
CSAE+SVM	0.50±0.03	0.50±0.03
SVM	0.50±0.03	0.50±0.04

Table 5: *The control evaluation of the schemes by 10-Fold Cross-validation. The parameter setting is the same as Table 4 except SVM is trained with randomized labels.*

4.3.6 Dataset B-1

The results in Figure 27 show the performance by using the examples with largest bleeding size and no bleeding, which gives strong discriminative features. It shows the highest performance as theoretical expectation. The slope of the AUC curves are decreasing with more training examples and approaching AUC = 0.8. However, the standard deviation remains around 0.05, which indicates the deviation de-correlates with the number of training examples.

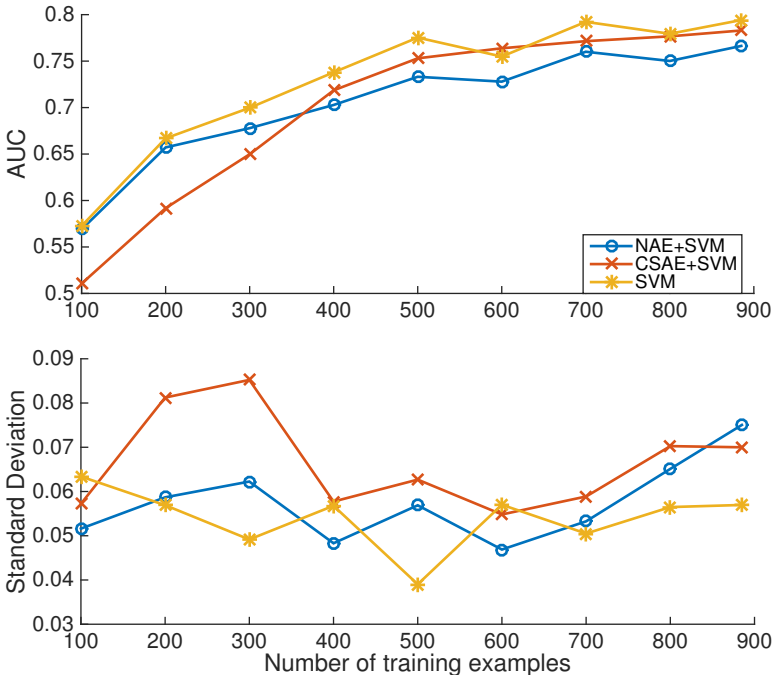

Figure 27: *Mean AUC and estimated standard deviation of Dataset B-1. Note the scales and label of each y axis.*

4.3.7 Dataset B-2

The AUC performance is shown in Figure 28. It is acquired by using the examples with smallest bleeding size and no bleeding, which gives weak discriminative features. The AUC grows slowly with more training examples with a maximum

AUC of 0.61. This test suggests the algorithms work not well when the size of bleeding is not conspicuous.

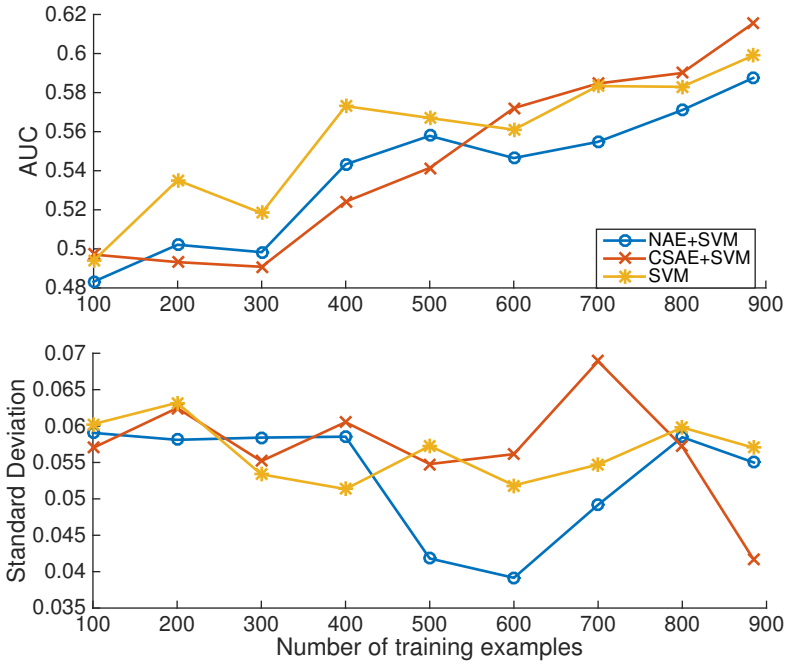

Figure 28: *Mean AUC and estimated standard deviation of Dataset B-2.*

4.3.8 Dataset B-3

The result in Figure 29 shows the performance by for the case in which the bleeding size is unknown. It shows the performance of this dataset is between B-1 and B-2 as expected.

4.4 Discussion

4.4.1 Performance

An interception could be noticed when comparing the performance of NAE with CSAE, which indicates that NAE performs better when the number of training examples is small and CSAE performs better as more training examples are used. The results from MNIST and CIFAR-10 datasets shown in Table 2 and 3 illustrated that when large number of examples are used in CSAE+SVM case, the performance is no worse than the performance when using SVM solely.

The difference regarding to performance for the three schemes is tiny. SVM shows stable performance in MNIST and CIFAR-10 datasets while shows high deviations in both Dataset A and B. Therefore, SVM may not be the best model for stroke patients classification.

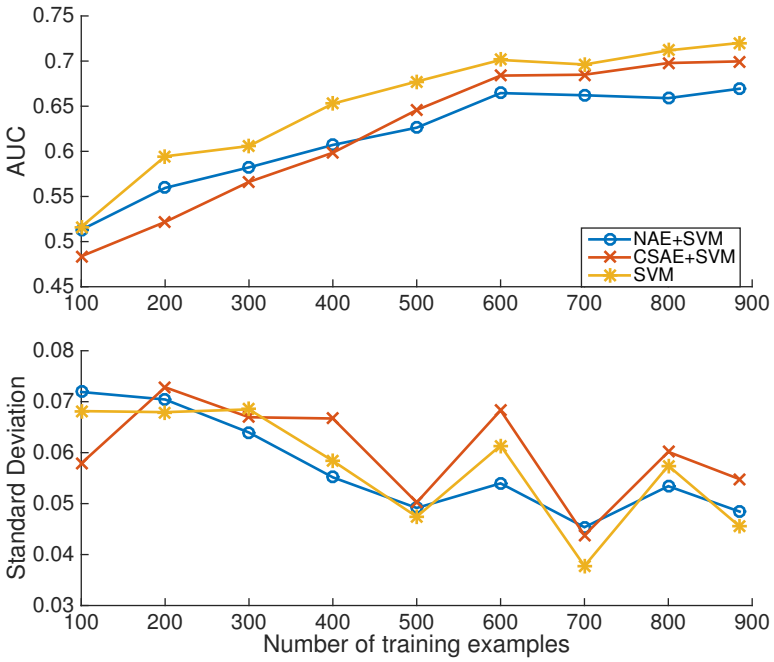

Figure 29: *Mean AUC and estimated standard deviation of Dataset B-3.*

4.4.2 Variability

AE is considered as high-variation model since it will learn different representation of the same data each time it runs. As a result, the classifier may yield different classification results. Therefore, higher deviation is expected when combining AE to classifier. This is reflected in Dataset B where the curve for SVM is more smooth and steady than the others.

The estimated standard deviation for all three schemes are all around 0.05, which is considered high . When analyzing the trend of standard deviation with training examples, the curves stabilized in Dataset A but continue oscillating in Dataset B. This indicates more simulations are necessary before the performance can be estimated accurately.

5 Conclusion

The tests of combining AE and SVM for classification is successfully carried out upon different datasets. For stroke classification, the performance using Linear kernel SVM with model selection can reach a peak AUC of more than 0.8 on both medical datasets. The combination of AE and SVM can hardly bring noticeable improvements in most cases. The conclusion is also revealed in external datasets where for most cases, the three schemes get indistinguishably similar results.

The results also have relatively high standard deviation especially in stroke datasets though it may decrease with more training examples. This may result from complex and high-dimensional nature of medical data since SVM has steady, low-deviation performance in external datasets. Therefore, it indicates that SVM may not be the best model for stroke classification problem.

It is worth noticing that, for datasets which contain large number of training examples, CSAE will outperform NAE regarding to performance. This information may be helpful as a reference when combining AE with other classifiers.

5.1 Limitation

In the planning stage, several feature selection methods are tested. For example, use real, imaginary or both parts, use the angle of the complex number as input features. Unfortunately, due to unknown reasons, none of above methods could bring promising classification results, i.e. the Accuracy or AUC remains 0.5, which means totally indiscriminate. Only the method that use the absolute value of the matrix could finally adopted as.

The training of AE and model selection in SVM may take hours to finish each time the Matlab code runs. Due to limited time for this project, the mean and standard deviation measurement of Accuracy and AUC for Dataset A and B are conducted only 20 or 30 times. As a result, the results may not converge perfectly.

5.2 Future work

More lab simulation data could be generated if possible since the classification performance in Dataset B shows the ascending trend of AUC curve with more training examples. In this case, the classification performance still has space to improve.

The AE in this project cannot strictly considered as deep learning method since it contains only one hidden layer [29]. It would be interesting to test the more complex case where several hidden layers are used, since it would be more natural to extract features from layer-by-layer process rather than find a straightforward representation by only one hidden layer [30].

It is difficult to evaluate the quality of features from AE regarding to the classification performance. Current method only uses the reconstruction error. It would be biased since this step naively considers well-trained AE would certainly improved the classification results. It would be necessary to build direct relation between the performance of AE and classification performance. This can be a complicated task since AE is of high variability which need numbers of simulations to model the distribution of the results.

For SVM classifier, heuristic methods could be applied instead of grid-searching methods as it could achieve similar results but be more time-efficient [10]. More classifiers including regression-oriented could be tested, since bleeding size in the brain is a continuous quantity. Binary classification may pose a problem in which the hard threshold may cause high distortion.

References

[1] "The top 10 causes of death," http://www.who.int/mediacentre/factsheets/fs310/en/, 2015.

[2] V. A. Thamburaj, *Textbook of contemporary neurosurgery.* Jaypee Bros., 2012.

[3] M. Persson, A. Fhager, H. Trefna, Y. Yu, and T. McKelvey, "Microwave-based stroke diagnosis making global prehospital thrombolytic treatment possible," *IEEE Transactions on Biomedical Engineering*, vol. 61, pp. 2806–2817, 2014.

[4] S. Semenov, "Microwave tomography: Review of the progress towards clinical applications," *The Royal Society*, vol. 367, no. 1900, pp. 3021–3042, 2009.

[5] A. Fhager, T. McKelvey, and M. Persson, "Stroke detection using a broadband microwave antenna system," *IEEE*, pp. 1–3, 2010.

[6] M. Schmidt, "minfunc: unconstrained differentiable multivariate optimization in matlab," http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html, 2015.

[7] A. Ng and J. Ngiam, "Learning color features with sparse autoencoders," http://deeplearning.stanford.edu/wiki/index.php/Exercise:Learning_color_features_with_Sparse_Autoencoders, 2011.

[8] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*, 3rd ed. Elsevier/Morgan Kaufmann, 2012.

[9] A. Ng and J. Ngiam, "Ufldl tutorial," http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial, 2015.

[10] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," National Taiwan University, Tech. Rep., 2010.

[11] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, vol. 35, no. 8, 2013.

[12] T. M. Mitchell, *Machine learning.* McGraw-Hill, 1997.

[13] A. Ng, "Sparse autoencoder," Stanford University, Tech. Rep., January 2011, cS294A Lecture notes.

[14] B. L. Quoc, J. Ngiam, and A. Coates, "On optimization methods for deep learning," Stanford University, Tech. Rep., 2011.

[15] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," School of Computer Science, Carnegie Mellon University, Tech. Rep., 1994.

[16] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergency property," *SIAM Journal on Optimization*, 1999.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, 2006.

[19] D. Erhan, Y. Bengio, and A. Courville, "Why does unsupervised pretraining help deep learning?" *Journal of Machine Learning Research 11*, 2010.

[20] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT Press, 2012.

[21] D. Serre, *Matrices: Theory and application.* Springer, 2002.

[22] M. Lichman, "UCI machine learning repository," https://archive.ics.uci.edu/ml/datasets/Wine, University of California, Irvine, School of Information and Computer Sciences, Tech. Rep., July 2015.

[23] A. Ng, "Support vector machine," Stanford University, Tech. Rep., January 2011, cS294A Lecture notes.

[24] H. Trevor, F. Jerome, and T. Robert, *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer Series in Statistics, 2001.

[25] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1137–1143, 1995.

[26] T. A. Runkler, *Data Analytics - Models and Algorithms for Intelligent Data Analysis.* Springer Vieweg, 2012.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, 1998.

[29] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[30] L. Deng and D. Yu, "Deep learning: Methods and applications," http://research.microsoft.com/pubs/209355/DeepLearning-NowPublishing-Vol7-SIG-039.pdf, 2014.