CHAPTER II

BACKGROUND

2.1 Conventions

We state here the definition that will commonly be used in the following sections. We will use an uppercase character such as X to refer to a matrix, while any lowercase character such as x will refer to a constant. The boldface letter such as $\mathbf{x}$ will refer to a vector, and we will denote it as a matrix with one column. Another commonly used convention will be I_m which will refer to an identity matrix with m rows. For a dataset X, each column of X represents each data where each row represents the value of each attribute.

2.2 Customization

Customization or user-adaptation is an approach in modifying/improving the model that is previously trained on a dataset to be better fit to a new dataset. The most useful thing about this approach is that it does not require having the previous data and should be more efficient in the time of operation. However, one obvious drawback could be in the performance of the model resulted from performing customization with the new dataset, comparing with the model that is retrained using both the new dataset and the previous dataset from scratch.

In customization, there will be a problem of how to weight the importance between the previously learned model and customization data appropriately in the case that both of these datasets come from the same distribution. However, it could be proved as a useful approach if the distribution migrates overtime. Each time we perform customization on a model, it will act as trying to make the model be better fit to the new dataset and at the same time slowly making the model forget what it has learned from the previous dataset by a degree. This is why it seems as a good approach to adapt a well learned model to another distribution with some similarity, and it also can be viewed as training an existing model so that it will yield better result with another dataset.

2.3 Classification

The task of classification is to predict the value of an attribute of a data, based on the value of other attributes of data as input. There are two tasks which fit this description, namely classification and regression, but the difference between them is in the possible value of the predicted attribute, i.e. the value from classification is discrete and the value from regression is continuous.

2.3.1 Weighted Nearest Neighbor

The original k-nearest neighbor algorithm (k-NN) (Mitchell, 1997; Hastie et al., 2001; Michie et al., 1994; Han and Kamber, 2000) is considered the simplest algorithm for classification in discussion. The algorithm is based on the idea of voting for the resulted class among k of the most similar training examples, measured in the feature space. In order to improve this algorithm for better accuracy and also for smoother regression in the probability of the prediction, each chosen candidate for voting will be given with its impact on the classification result, based on how close they are to the target of the prediction. One can also view k-nearest neighbor as a special case of weighted nearest neighbor with the weight function $lim_{k\to\infty}x^k$ when x is the Euclidean distance measured between the training data and input data, or to say the infinite norm ($||\cdot||_\infty$) of the measured distance.

2.3.2 Max Wins

Max Wins (Friedman, 1996), so called voting scheme, is a well-known approach to combine many binary classifiers into one multiclass classifier. This method could be easily described as performing voting between all binary classifiers. The most voted class will be the result of the classification for the multiclass classifier.

2.3.3 Adaptive Directed Acyclic Graph

Adaptive Directed Acyclic Graph (ADAG) (Kijsirikul et al., 2002) is another approach to combine many binary classifiers into one multiclass classifier.

The method would generally be portrayed as traversing an inverse binary decision tree with each of possible classes as its leaf, as shown in Figure 2.1. The prerequisite requirement of this algorithm is to have existing binary classifiers between each pair of possible classes. At each node, the binary classifier for the pair of respective classes will be used on the data, and the unlikely class will be eliminated from consideration while the more likely one will be passed on along the tree and be used again in the comparison at the next node, until the root is reached with the most likely class. Compared to Max Wins, the main advantage of ADAG is that it requires $n - 1$ times of classification by binary classifiers to eliminate $n - 1$ unlikely answers instead of $n(n-1)/2$ times of classification by using Max Wins. However, its prediction result would still be inferior to Max Wins. Another characteristic of ADAG is that, given that there is an improvement on each of subclassifiers, the expected improvement on the classifier created by ADAG should yield greater improvement than Max Wins, since the classification would improve on each of the classifiers along the path from the correct class comparing to linear improvement from Max Wins.

Note that though there is an improved version of ADAG called Reordering Adaptive Directed Acyclic Graph (RADAG) (Phetkaew et al., 2003) which could yield even better result than Max Wins, but its characteristic in relying on error estimation for each of the binary classifiers would make it be more complicated to use on any classifiers other than support vector machines (SVMs), requiring k-fold cross validation to estimate the error itself, and hence it will not be used in this dissertation.

2.3.4 Optical Character Recognition

Optical Character Recognition (OCR) is a kind of classification task where we want to predict a character given an image of that character. The most usual form of OCR input is the value in each pixel of the image with fixed size of the character. The process of this task usually begins by scanning a paper for the text. After scanning is done, the next step is to separate the whole image into several individual images of characters before performing OCR on each of them. After that, the results are then combined into words and processed back

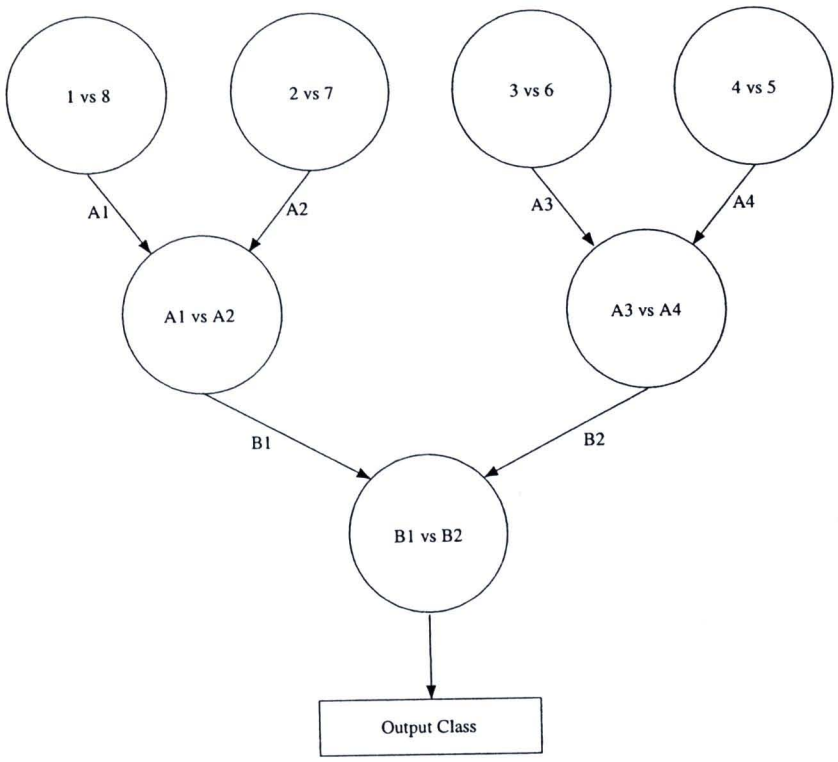


Figure 2.1: Visualization for Adaptive Directed Acyclic Graph.

into predicted text of the page.

2.3.5 Handwriting Recognition

Handwriting Recognition (HWR) is another kind of task in classification that we want to predict characters a person writes, with the information about how the writing tool has been stroked. The most general form of input of this task is the pen stroke that designates the position and traces how the pen is drawn and lift. Two-dimensional time series data is generated with the value designating the current position of the pen. However, in many cases, features from this time series will be extracted and be used as in normal classification.

2.4 Dimensionality Reduction

Dimensionality reduction refers to a process in reducing the number of attributes of data for further use. Namely, the algorithm for dimensionality reduction will yield the output as a function or process $\mu : \mathbf{R^n} \rightarrow \mathbf{R^m}$ when n is the number of attributes as the input and m is the desired number of attributes.

Instead of using a dataset M for that task as usual, we will use the result from the transformation, $\mu(M)$, as dataset for the task instead.

There are many motivations to perform dimensionality reduction. The most obvious is for the efficiency. Suppose that the task to be performed requires a lot of resource so it cannot satisfy some constraint in time and memory. The choice will be either to change the algorithm or to reduce the load by reducing the size of data. Instead of reducing the number of examples, dimensionality reduction removes some information about each data. The matter to be considered in dimensionality reduction is how to do it in the fashion that the resulted data will be as useful as possible for the task.

Another purpose of dimensionality reduction is not because of limitation, but to improve the result. Usually, a dataset may contain many useless information that could be considered as noise. Performing dimensionality reduction will act as attempting to eliminate the most useless information from the data, namely eliminate the noise, and hence it will be useful for the algorithm that is noise sensitive. Another example is in altering the space of decision for the algorithm. Dimensionality reduction may act as transformation of input space in which the algorithm for the task does not effectively perform, and thus alter the possible decision which the algorithm can produce. As a result, the algorithm will be able to handle more complicated task by using dimensionality reduction as preprocessing. A good example is performing nonlinear dimensionality reduction before applying an algorithm whose decision is based on a limited number of linear planes. In this way we could perform better on the dataset which requires more complicated space of decision.

The main issues about dimensionality reduction can be described as following. The first is how to determine if the result from optimization should perform well on the upcoming task. That is choosing the value to be optimized with the expectation that the resulted dataset will be good for the algorithm which performs the desired machine learning task. The next issue is how to find the function or create the process $\mu(M)$ that provides good score on that goal, namely the process of optimization. The last one is the constraint or limi-

tation in the space of dimensionality reduction.

2.4.1 Linear Dimensionality Reduction

Linear dimensionality reduction is dimensionality reduction with the constraint that $\mu(M)$ must be linear transformation. This can be viewed as projecting the dataset on a linear subspace. Linear dimensionality reduction is important because of the optimization method. Limiting the space of transformation to be linear transformation results in better formulation of the problem and produces the form of semidefinite programming or quadratic programming which could be optimized with a well-known efficient procedure. In such a case, the method yields the optimal solution, not just a good one.

2.4.1.1 Principal Component Analysis

Principal component analysis (PCA) (Han and Kamber, 2000) is an unsupervised linear dimensionality reduction algorithm and is usually considered as the simplest form of dimensionality reduction which is widely used. The goal of PCA is to maximize the variance of a transformed dataset with a fixed number of attributes.

Principal component analysis consists of two main steps. The first is to calculate a covariance matrix of the dataset, i.e. the generalization of variances in many dimensions. Each element of covariance matrix C in row i and column j is according to the formula $c_{ij} = \sum(x_i - \bar{x})(x_j - \bar{x_j})$, or if X is a matrix of the dataset with each row as each of the data that is translated to have zero mean then $C = XX^T$.

The second step of principal component analysis is to perform eigenvalue decomposition of the covariance matrix. Since the matrix XX^T is squared, symmetric and positive semidefinite, it follows that all the eigenvalues of the matrix are nonnegative real values. We can find the set of orthonormal eigenvectors of the matrix that span the entire space, and we can perform eigenvalue decomposition of the matrix. This means that we can find an orthogonal matrix U that $C = U\delta U^T$ where δ is the diagonal matrix of eigenvalues ordered by their sizes.

Transformation by principal component analysis to reduce the number of dimensions to m will be made by using the m eigenvectors with maximum eigenvalues as the new axis, in which the resulted new attributes are projection of the data on them. Let us consider this to have better understanding of principal component analysis. If we transform dataset X with orthogonal matrix U by projecting the data on its axis, then the new dataset will become $U^T X$. We can calculate the covariance of this new matrix as following:

$$\begin{aligned} U^T X (U^T X)^T &= U^T X X^T U, \\ &= U^T C U, \\ &= U^T U \delta U^T U, \\ &= I_m \delta I_m, \\ &= \delta. \end{aligned}$$

This means that the value of covariance matrix created from the transformed dataset will only exist in the diagonal elements. In another aspect, if we want a dimension that maximizes the variance along that direction, then it will be obvious that the first eigenvector will be the first column of U. By repeating this process of choosing the direction in the linear space that is orthogonal to all the already chosen directions, we will get the resulted axis of m-dimensional space as the m eigenvectors with the maximum eigenvalues of the covariance matrix.

2.4.1.2 Linear Discriminant Analysis

Linear discriminant analysis (LDA) is among one of the simplest supervised linear dimensionality reduction. The goal of linear discriminant analysis is to find a linear transformation that maximizes the discrimination between classes.

2.4.2 KPCA Methods of Mahalanobis Distance Learning for Nearest Neighbor

KPCA methods of Mahalanobis distance learning for nearest neighbor (KMMLN) (Chatpatanasiri et al., 2010, 2008) is a group of algorithms for dimensionality reduction. The idea of this type of algorithms is that given a linear dimensionality reduction, the algorithms make a nonlinear dimensionality reduction in the way that is similar to kernel trick of support vector machine by using Kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1998). The algorithm consists of 3 steps, i.e. finding optimal kernel, performing KPCA, then using linear dimensionality reduction algorithms.

2.5 Halton Sequence

When there is the need to perform a well distributed sampling inside a close boundary of n-dimensional space, the easiest way will be to sample with uniform randomization. However, if the result from sampling is made in one or two dimensional box and the result of sampling is visualized by the scattering of the result when viewing the close boundary then, in most cases, one could easily feel that all the sampling result could be better distributed if hand-picked by man, as shown in Figure 2.2. In order to evaluate the sampling quality, there are two values called discrepancy and dispersion which serve as measurements, and can be formulated as following (Choset et al., 2005).

$$D(P,\mathbb{R}) = sup_{R\in\mathbb{R}} \left| \frac{\mu(R)}{\mu(X)} - \frac{|P\cap R|}{N} \right|$$

$$\delta(P,\rho) = sup_{x\in X} min_{p\in P} \rho(x,p)$$

where D is discrepancy, δ is dispersion, P is a set of point samples on the space X, N is the number of points in P, $\mathbb{R}$ is the set of possible spaces used in measuring the value, μ is a measure of space or size of the space and ρ is a metric measuring the distance. Both equations can be interpreted as following: discrepancy measures how the sampling is distributed, comparing the difference between ratio of sampled data and sampling space, while dispersion measures the size of the largest region in which none of the sampling could be found,

Table 2.1: First few terms in Halton sequence for two dimensional space.

n	n_2	n_3	$\Phi_2(n)_2$	$\Phi_3(n)_3$	$\Phi_2(n)$	$\Phi_3(n)$
1	1	1	.1	.1	1/2	1/3
2	10	2	.01	.2	1/4	2/3
3	11	10	.11	.01	3/4	1/9
4	100	11	.001	.11	1/8	4/9
5	101	12	.101	.21	5/8	7/9
6	110	20	.011	.02	3/8	2/9
7	111	21	.111	.12	7/8	5/9
8	1000	22	.0001	.22	1/16	8/9
9	1001	100	.1001	.001	9/16	1/27
10	1010	101	.0101	.101	5/16	10/27

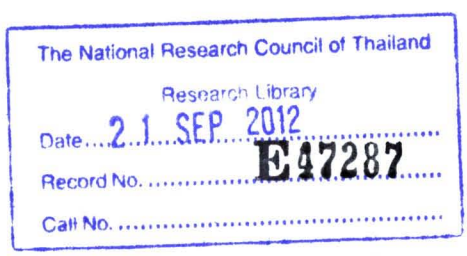
which is a ball in this equation.

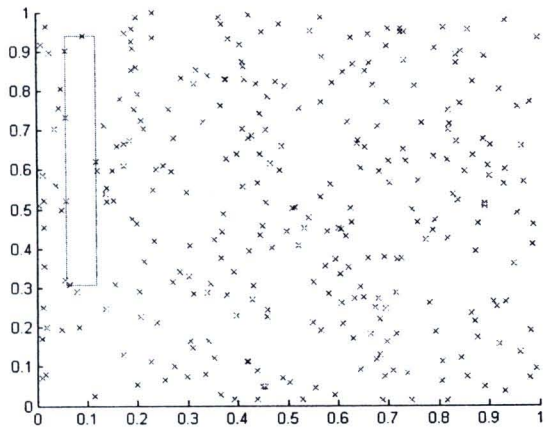
In order to achieve better result than randomization and to guarantee the worst case result, one would have to resort to quasirandom sampling, that is using deterministic scheme to determine the sampling sequence, which could be further perturbed by adding some small randomization. One of well-known methods is Halton sequence (Halton, 1960), which could be formulated as following.

$$p_n = (\Phi_{b_1}(n), \Phi_{b_2}(n), ..., \Phi_{b_d}(n)).$$

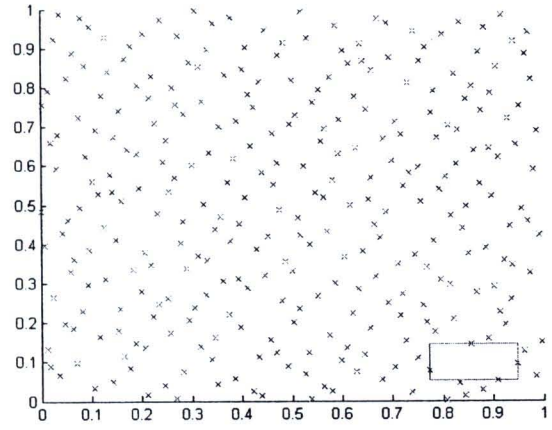
$$\Phi_{b_j}(n) = \sum a_{ij} b_j^{-(i+1)}.$$

Where d is the number of dimensions, b_j is the j^{th} prime, and a_{ij} is the i^{th} digit of number n when written as based j numeral and written backward from behind the decimal, as shown in Table 2.1.





(a) random number generator (dispersion=0.0383)



(b) Halton sequence (dispersion=0.0164)

Figure 2.2: 300 sampled data in two dimensions ($[0,1]^2$) and dispersion value measured with box space. The box shows the area which results in dispersion value.