The Academic College of Tel Aviv-Yaffo School of Computer Science

Ensemble Classification via Kernel-PCA Dimensionality Reduction

Submitted by Chen Zvi

Under the supervision of Dr. Alon Schclar

Ensemble Classification via Kernel-PCA Dimensionality Reduction

Abstract

Over the last few decades, memory storage, data gathering, and computational capabilities have been significantly improved. Machine learning algorithms had to evolve to handle large quantities of data in a high dimension.

Ensemble methods are learning algorithms that construct a set of classifiers whose predictions are combined, usually by voting or an averaged weighting scheme, when classifying new instances.

In this paper, we describe a new ensemble classification algorithm, based on Kernel PCA. First, we reduce the dimension by Kernel PCA, in order to make use of the dimensionality reduction benefits, and to achieve the required diversity for the ensemble construction. Second, we train the ensemble members based on the dimension reduced data. Test samples are classified by first embedding them into the dimension reduced spaces of the ensemble members and then applying the ensemble members to the embedded sample.

Acknowledgements

I would like to to express my profound gratitude appreciation to my supervisor, Dr.Alon Schclar, of the Computer Science School at The Academic College of Tel Aviv-Yaffo. Dr.Alon Schclar provided valuable encouragement, positive attitude and meaningful guidance during the research process. The completion of this thesis would not have been possible without his dedication, constant availability and professional support as I refined my research.

Finally, I want to take this opportunity to express my sincerely gratitude to my mother, Shulamit, for providing me unfailing support and continuous encouragement during my years of study and writing this thesis. Thank you.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Ensembles Of Classifiers

Ensemble of classifiers conceptualize the wisdom of the crowd principle, which claims that a group of individual's collective opinion, is better than a single expert's opinion, when it is used for problem solving, decision making, innovation and prediction [1]. Ensemble methods are learning algorithms that train a group of classifiers, and predict a new sample by conducting a vote or, by weighting the classifiers predictions.

The idea of implementing the wisdom of the crowd principle to machine learning models has been in use for a long time; Since the 1990s many researchers have investigated this method of combining, by voting or aggregating, results of multiple base classifiers to a single outcome in many areas like: Data mining, Machine learning, Neural Networks, Pattern Recognition and Statistics [2][3][4].

Many researchers achieved significant performance improvements over single classifiers using ensemble methods [5][6] along with empirical researches that demonstrated superiority over single classifiers [7][8][9][10]. Besides the performance, ensemble methods have a huge advantage in Big Data since they can be executed in parallel and reduce the execution time when they are applied to [11][12].

Successful implementations of ensemble methods can be found in a wide range of areas such as Image and Video processing [13][14][15][16], Intelligent Transportation systems [17][18], Bioinformatics [19][20][21][22], Remote sensing [23][24] Manufacturing [25], Finance [26], to name a few.

1.1 The Ensemble of Classifiers Framework

An ensemble of classifiers consists of a set of individually trained classifiers which can be any kind of induction algorithms like Decision Trees, Nearest Neighbour, Neural

Network etc.. Typical pipeline of an Ensemble for classification is composed of three stages: selection and application of a diversity strategy to the data, training the base classifiers and answer derivation by voting or aggregate.

Figure 1.1 illustrates the basic ensemble architecture.

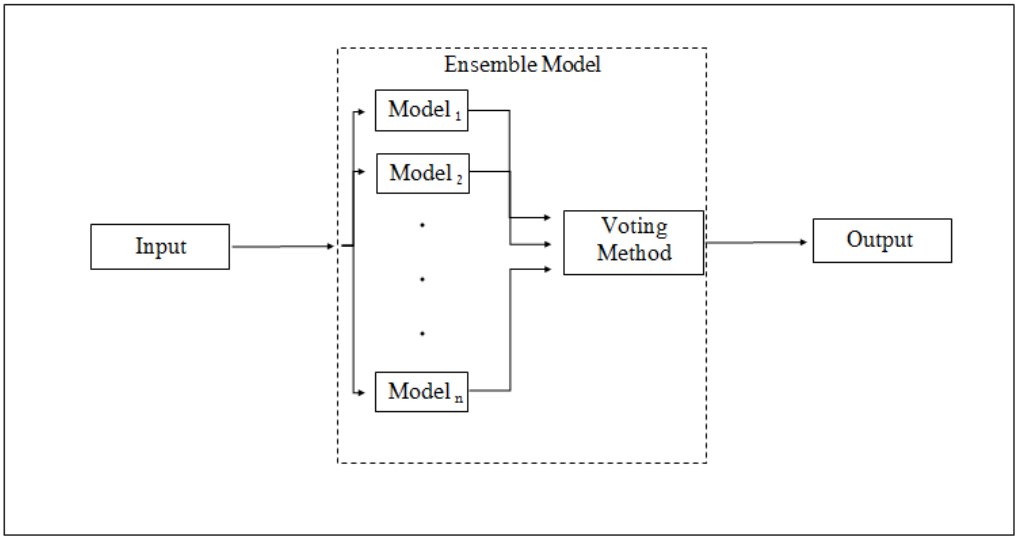

FIGURE 1.1: Basic Ensemble Architecture

There are many methods for constructing ensembles of classifiers. Most of them use a single induction algorithm to produce the ensemble members. This type of ensembles is known as single-model ensembles or ensembles of weak learners as opposed to multi-model ensembles that use several induction algorithms. The proposed method in this paper falls into the former category. Schapire [27] proved that weak learners‘ accuracy can be at least as good as strong learners, this research motivates using ensembles of weak learners. This method yielded some of the famous ensemble methods such as AdaBoost [28], Bagging [29], etc.

An effective ensemble classifier should adhere to two contradicting principles: Accuracy and Diversity [30]. Accuracy requires each individual classifier to have a better error rate than a classifier that randomly classifies a new sample. Diversity means that the generalization errors of the ensemble members should be uncorrelated. In other words, combination of several classifiers is only useful if they disagree on some inputs [7][31]. Several approaches have been proposed to achieve diversity for classifier ensembles. In our proposed algorithm, we introduce a novel method for the manipulation of the data by using Kernel PCA, in order to obtain the required diversity.

There are two popular and widely used ensemble learning techniques for creating accurate and diverse ensembles: Bagging [29] and Boosting [27][28]. These methods achieve the diversity by constructing a different training set for each of the ensemble members. Bagging uses subsampling while Boosting uses weighting.

1.2 Bootstrap Aggregation - Bagging

Bagging [29] is one of the earliest, simplest and effective single-model ensemble classifiers. The Bagging algorithm achieves the required diversity by generating different subsets (bootstraps) for the ensemble members. Each bootstrap is constructed by randomly choosing, with replications data instances, from the original training set. After generating the bootstraps, a base classifier model, like decision tree, is trained using each training set. The final result is calculated by applying voting (for classification) or averaging (for regression) to the results of the individual ensemble member.

Standard Bagging implementation, generates bootstraps whose size equals to that of the original training dataset. Because of the replications, any sample may appear in a bootstrap more than once, whereas some samples may not appear at all.

Each bootstrap sample contains approximately 63.2% unique samples from the training set. The probability of a sample being selected randomly from the training set to a bootstrap of size n, is $1-(1-1/n)^n$. For large n, this is about $1-1/e = 63.2\%$ [32].

Bagging can easily run in parallel i.e. the ensemble members can be trained in parallel. This method has been successfully applied to a wide range of problems such as Spam detection [33] and analysis of Gene expressions [34], to name a few.

Algorithm 1 summarizes the Bagging algorithm.

Algorithm 1 Bagging Algorithm

Input:
$D = \{\mathbf{d_1}, \mathbf{d_2}, ..., \mathbf{d_N}\}$- Training set
B- Number of bootstrap samples
L- Base learning algorithm
Output :
C*- bagging ensemble classifier

1: **for** i=1 **to** B **do**
2: $S_i \leftarrow$bootstrap sample of size n, from D
3: $C_i \leftarrow$Train a base classifier L(S_i)
4: **end for**
5: $C^*(x) = \mathbf{argmax_y} \sum_i^B (C_i(x) = y)$

1.2.1 Random Forest

Random Forest [35] is an extension of Bagging, uses modified decision trees as the inductions algorithm. The first projects the training set of each member onto a random subset of the original feature. The second uses a random procedure to split the subset of instances at each node.

Random Forest can be used for both classification and regression problems. In order to classify a new instance in case of regression, Random Forest averages the results of all individual decision trees, while in case of classification, random forest will use a majority vote.

In addition to the diversity obtained from constructing each tree by using a different subset of the data, Random Forest also change how the classification or the regression trees are constructed.

Random Forest is one of the most popular and most powerful machine learning algorithms. It has excellent performance, even without hyper-parameter tuning, over fitting control, application versatility. It can be found in many areas in daily life such as remote sensing and geographic data [36], bioinformatics [37], banking [38], stock market [39][40] etc.

1.2.2 Rotation Forest

Another state-of-the-art algorithm is the Rotation Forest [41] whose members, similarly to Random Forest, are decision trees. However, it uses a different algorithm to construct the individual training sets. Rotation Forest achieves diversity by constructing each training set using the following steps. First, it applies bootstrap sampling to the training set. Then it randomly splits the original set of features into subsets. Then each subset is rotated using PCA [42].

1.3 Boosting

Boosting [43][44] is a family of machine learning algorithms that are able to convert a set of weak learners to strong learners. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor.

Bagging followed the research by Kearns and Valiant [45] research, who investigated the theoretical question, whether the complexity classes of weakly learnable and strongly

learnable problems, are equal or in other words, Is a set of weak learners equivalent to a single strong learner [46]. Schapire proved that the answer to their question is positive, and confirmed it through boosting.

Boosting achieves its diversity by manipulating the training set like Bagging [10]. However, the major difference between them is that it trains the ensemble members in a sequence, on a weighted version of the data, instead of randomly choosing data instances, the instances are chosen based on a distribution that is updated in each iteration according to the classification results of each member. The classification of a new sample takes into account the classification/misclassification rates of the individual members.

Algorithm 2 presents the general Boosting procedure.

Algorithm 2 Boosting Algorithm

Input:
D- Sample distribution
R- Number of learning rounds
L- Base learning algorithm
Output : C- Boosting ensemble classifier

1: $D_1 \leftarrow D$ - Initialize first distribution
2: **for** i=1 **to** R **do**
3: $C_i \leftarrow L(D_i)$- Train a base learner from distribution D_i
4: $\epsilon_i \leftarrow C_i(x) \neq f(x)$- Evaluate the error of C_i
5: $D_{i+1} \leftarrow AdjustDistribution(D_i, \epsilon_i)$
6: **end for**
7: $C(x) = CombineOutputs(h_1(x), ..., h_i(x))$

The most common implementation of the Boosting approach is the award winning Adaboost [28], which stands for adaptive boosting.

1.3.1 AdaBoost

The Adaptive Boosting (AdaBoost) algorithm [10] is a popular ensemble method that uses an iterative process. AdaBoost, by Yoav Freund and Robert Schapire won the Gödel Prize, an annual prize for outstanding papers in the area of theoretical computer science in 2003. Similar to Bagging, AdaBoost can be used for both classification and regression problem.

Adaboost combines multiple weak learners, such as decision trees and decision stumps into a single strong learner. The main idea behind this algorithm is to be more sensitive, through weights, to samples that are harder to classify. The weights represent the distribution of the training instances. A higher the weight the higher the chance of being selected to the training set of the next member.

The iterative process works as follow, at the first iteration, all samples have the same weight i.e. have the same chance of being selected. In each iteration the weights are updated. The weight of samples that were incorrectly classified, will (boosted) be increased compared to than those that were correctly classified, in order to have higher focus on them in the next iteration by increasing the chance of choosing them. The member classifier is built using a bootstrap sample based on the updated weights, which are no longer equal, and so on. At the end of the iterative process, the final result is defined as a weighted sum of the member results where the weights are defined according the error rate of the individual members.

1.3.2 Gradient Boosting

Gradient Boosting is another boosting algorithm for both regression and classification. Just like AdaBoost, Gradient Boosting iteratively constructs a strong learner from an ensemble of weak learners, where each member corrects the errors of its predecessor. The different between those to boosting algorithms is how they create the weak learners during the iterative process.

Gradient boosting visualizes the boosting problem as an optimisation problem, i.e it defines the classification/regression error as a loss function and minimizes it. This idea was first introduced by Leo Breiman [47]. Two years later, Friedman, J. H published the explicit regression gradient boosting algorithms [48][49].

In each iteration a member is constructed and the loss function is calculated. The loss represents the differetns between the actual value and the predicted value. In each iteration the predictions are updated to minimize the residuals.

Chapter 2

Kernels

2.1 Kernels and kernel trick

Kernel methods are a class of algorithms that are use for pattern recognition and analysis. The main goal of pattern analysis is to discover and learn types of relations, such as classifications, clusters, principal components and correlations, among samples in a data set. The kernel approach was first introduced by Aizerman et al [50] for pattern recognition using potential functions. However, it was first utilized for machine learning in [51]. From the 2000s kernel methods have become rather popular and widely used by learning algorithms. These methods have rigorous foundations and therefore they are popular in many domains.

There are many machine learning algorithms that can only handle linear datasets. However, in most cases the data sets are nonlinear. Kernel methods implicitly embed the data into a higher dimensional space, in order to convert non-linear structures into linear structures. These methods are tailored for algorithms that use inner products. Instead of employing the inner products in the original space the inner product is calculated in some high dimensional space where the dataset is "hopefully" linearly separable. This is achieved by calculating a kernel function

$$K(x_i, x_j) =< \phi(x_i), \phi(x_j) > \tag{2.1}$$

where $\phi : R^D \rightarrow E$ is the mapping of the original data points into the high dimensional space. Equation (2.1) is known as the kernel trick [52]. Namely, every kernel K that obeys certain rules, corresponds to a mapping into a high-dimensional space. Therefore, there is no need to explicitly find the mapping - only to calculate the kernel.

A kernel is required to be symmetric and positive (semi-) definite, i.e. for all $\mathbf{x_i}, \mathbf{x_j} \in R^D, K(\mathbf{x_i}, \mathbf{x_j}) = K(\mathbf{x_j}, \mathbf{x_i})$, and for all nonzero $\mathbf{y} \in \mathbf{R^n}, \mathbf{y'Ky} \geq \mathbf{0}$ is positive-semidefinite.

The Kernel trick is a powerful tool that easily provides a non-linear version of all algorithms that are described solely in terms of inner products between two vectors. The Kernel trick stems from the assumption that data with a non-linear structure can have a linear structure when they are embedded into a high dimensional space.

The chosen algorithm can be employed in the high dimensional space without explicitly projecting the data into this space, but rather by simply computing the inner products between the vectors of all pairs of data in the original space. This trick is highly used because it is allows to project the data into an infinite-dimensional feature space which is unfeasible to compute.

Kernel methods [53] are a powerful tool in machine learning. They have been successfully applied to many data types e.g. graphs [54][55], text categorization [56], images and bioinformatics [57]. Additionally, kernel methods achieved excellent results in natural language processing [58].

2.2 Popular kernels

There are three commonly used kernels: The RBF kernel, the Polynomial kernel and the Sigmoid.

2.2.1 The Radial Basis Function - RBF

The Radial Basis Function kernel which is also known as Gaussian kernel, or RBF, have been successfully applied to many pattern classification tasks. It is one of the most popular and versatile kernels in Euclidean spaces, and is extensively used in various kernel learning algorithms. The RBF kernel maps the data points to an infinite dimensional Hilbert space, and therefore can produce a very rich representation of the data. As we mentioned above a kernel function can be used in many cases as a measure of similarity between vectors. The RBF kernel provides a similarity measure between two vectors based on their Euclidean distance. If the Euclidean distance is small, the Gaussian tends to 1 and when the vector are far apart, the Gaussian tends to zero. Formally, the Gaussian takes the following form:

$$k(x, y) = -\exp(\sigma \left\|x - y\right\|^2)$$

The adjustable parameter σ endows a neighborhood notion for the similarity measure. When σ is large, the kernel will act almost linearly and produce values that are near 1. A small σ will produce a kernel that is highly sensitive to noise in the training data. σ also defines the width of the Gaussian bell-shaped curve. Larger values correspond to a narrow bell while higher values correspond to a broad bell. Figure 2.1 illustrated the influence of σ on the Gaussian bell-shape.

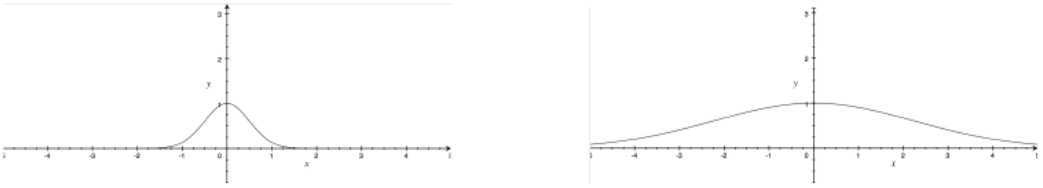

FIGURE 2.1: Gaussian bell
(a) Gaussian bell with large σ. (b) Gaussian bell with small σ .

2.2.2 Polynomial kernel

The Polynomial kernel is a simple kernel and is commonly used in Support Vector Machines . It is defined as:

$$k(x, y) = (\gamma x^T y + c)^d$$

The free parameters are d the polynomial degree, which is the order of the kernel, γ known as slope and $c >= 0$ which is a constant that responsible to trade off the influence of the higher and lower order terms. When $c = 0$ the kernel is a homogeneous polynomial [59], a polynomial whose all nonzero terms have the same degree.

Although any degree is applicable, the most commonly used degree is $d = 2$ [60] since higher degrees tend to cause overfitting. In general, polynomial kernels are applied to normalized data, because otherwise they result in kernels that are numerically unstable. Polynomial kernels are successfully applied in Speech Recognition [61] and natural language processing [62].

2.2.3 The Sigmoid kernel

The Sigmoid kernel or the Hyperbolic Tangent kernel owes its popularity to artificial neural networks where it is often used as an activation function. However, it is not widely used since the kernel matrix may not be positive semi-definite, and therefore its behavior is unexpected. In 1995, Vapnik first published the idea that kernel matrices

may not be positive semi-definite for all values of the free parameters [63]. The SVM algorithm uses the sigmoid kernel most of the time as a proxy for neural networks, Furthermore, a SVM model with a sigmoid kernel function is equivalent to a two-layer, perceptron neural network. The Sigmoid kernel is expressed as:

$$k(x, y) = \tanh(\gamma x^T y + c)$$

The kernel is composed of two free parameters, γ and c. A common value for c is $\frac{1}{N}$, where N is the dimension of the original data. The parameter c can be viewed as a shifting parameter that controls the threshold of mapping, and the parameter γ is a scaling factor [64].

Chapter 3

Dimensionality Reduction

Real-world data, such as videos, images, speech signals and text, resides in high-dimensional space. High-dimensional datasets introduce many mathematical challenges. These challenges include the "curse of dimensionality" [65], for traditional algorithms using machine learning and pattern recognition applications. In order to handle such high dimensional datasets in machine learning algorithms, their dimensionality needs to be reduced. Dimensionality reduction resolves this problem as well as improves the efficiency and accuracy of data analysis.

Dimensionality reduction, in the context of machine learning, refers to the process of embedding data in a D-dimensional space into a d-dimensional space where $d \ll D$, while ensuring that the low-dimensional embedding concisely conveys similar information. In other words, it is a transformation of data in high-dimensional into an equally informative representation of reduced dimensionality. The reduced data, should still represent the correlation between the samples, therefore this dimension is the minimum number of parameters that are needed to account for the observed properties of the data [66].

By reducing the dimensionality of the data, both the time and space complexities of algorithms that are employed are reduced. This facilitates the application of algorithms whose execution is not feasible in the original space. Furthermore, it can decorrelate the data and remove unnecessary features. Finally, it can reduce noise and therefore improve the performance and accuracy. To allow visualization of the dataset, the target dimension is usually set to 2 or 3.

Dimensionality reduction can be classified into two major categories, feature selection and feature extraction [67]. Feature selection uses a subset of the original features to create a data model. The major strategies for feature selection are: filter, wrapper

and embedded [68][69]. Feature extraction, unlike feature selection, produces a new set of features, that are a function (usually non-linear) of the original features. Common feature etraction methods include PCA 3.1, Kernel PCA 3.2 and Isomap 3.3. In the following we describe some of the most popular dimensionality reduction techniques. A review of the most commonly used dimensionality reduction methods can be found in [70].

3.1 Principal Component Analysis - PCA

Principal component analysis (PCA) [71][42] is probably the most widely used linear technique for dimensionality reduction. It reduces the dimension by embedding the data into a linear subspace of a lower dimension. The algorithm finds the directions of the maximum variance in the high-dimensionality data, and projects the data onto these directions to retain as much of the variance as possible.

The linear subspace can be specified by vectors, which are orthogonal, linear transformations of the original data points, named principal components. The principal components are extracted in such a way that the first principal component describe the direction of the maximum variance in the dataset, the second principal component is direction of the next highest variance in the dataset, and is uncorrelated to the first one, and so on. The number of principle components can be grater than the size of the original dimension. However, the assumption is that only $n << N$ (N the number of principle components) are needed to represent most of the information in the dataset.

PCA is mostly used as a tool in exploratory data analysis and for making predictive models. It works well when the data approximately lies near a hyperplane. However, PCA fails when the structure of the data is not linear.

Algorithm 3 summarizes the PCA algorithm.

Algorithm 3 PCA Algorithm

Input: $X = \{\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N}\}, \mathbf{x_i} \in R^D$- Training set
d- new lower dimension, d $\ll$ D

1: $\tilde{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ - Compute the mean
2: $Cov(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \tilde{x}_i)(x_i - \tilde{x}_i)^T$
3: Find decomposition of $Cov(x)$, obtaining $\xi_1, \xi_2, ..., \xi_D$- the eigenvectors and their corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_D \geq 0$
4: $y = (\xi_1^T(x - \tilde{x}), \xi_2^T(x - \tilde{x}), ..., \xi_d^T(x - \tilde{x}))^T \in R^d$ - The new lower dimension presentation $\forall x \in R^d$.
5: **Output** : $x \approx \tilde{x} + (\xi_1^T(x - \tilde{x}))\xi_1 + (\xi_2^T(x - \tilde{x}))\xi_2 + ... + (\xi_d^T(x - \tilde{x}))\xi_d$

3.2 The Kernel PCA (KPCA) Dimensionality Reduction Algorithm

Kernel PCA [72] is a nonlinear extension of the PCA algorithm. PCA is a powerful technique for linear dimensionality reduction and feature extraction. However, it is not effective for data whose structure is not linear. KPCA makes use of the "kernel trick" by implicitly projecting linearly inseparable data onto a higher dimensional space, where the data becomes linearly separable. KPCA was first proposed by B. Schölkopf. [72]. It has been utilized in a wide range of problems such as face recognition [73] and handwritten digits recognition [74]. The next paragraphs will describe the Kernel PCA algorithm.

Suppose we are given a nonempty set of vectors $\{\mathbf{x_1}, ..., \mathbf{x_n}\}, \mathbf{x_i} \in R^D$. The set is mapped by $\phi : R^D \rightarrow E$, to a high dimensional feature space E, so the mapping data can be written as $\phi(\mathbf{x_1}), ..., \phi(\mathbf{x_n}), \mathbf{x_i} \in E$.

The target space may be infinite dimensional. We use the "kernel trick" which is a function that calculates the dot product in E, for all $\mathbf{x_i}, \mathbf{x_j} \in R^D$ under $\phi, k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})\phi(\mathbf{x_j})^T$. Kernel PCA computes the principal components of the mapped data.

Lets assume that the projected data have zero mean.

$$\frac{1}{n}\sum_{i=1}^{\mathbf{n}} \phi(\mathbf{x_i}) = 0 \tag{3.1}$$

The formula of the covariance matrix for PCA, after the projection is:

$$C = \frac{1}{n}\sum_{i=1}^{\mathbf{n}} \phi(\mathbf{x_i})(\mathbf{x_i})^T \tag{3.2}$$

In order to find the principal components of the data in E, we calculate the eigenvectors and eigenvalues of the covariance matrix C.

$$Cv = \lambda v \tag{3.3}$$

By substituting (3.2) into (3.3) and dividing by λ, v can be expressed as:

$$v = \sum_{i=1}^{\mathbf{n}} \mathbf{a_i}\phi(\mathbf{x_i}) \tag{3.4}$$

We note that for $\lambda \neq 0$, all the solutions for v must be a part of the span of $\phi(\mathbf{x_1}), ..., \phi(\mathbf{x_n})$ Finding the eigenvectors is equivalent to finding the coefficients $\mathbf{a_i}$. Substituting (3.2) and (3.4) into (3.3) we get:

$$\frac{1}{n}\sum_{i=1}^{\mathbf{n}} \phi(\mathbf{x_i})\phi(\mathbf{x_i})^T \sum_{l=1}^{\mathbf{n}} \mathbf{a}_{jl}\phi(\mathbf{x_l}) = \lambda_{\mathbf{j}} \sum_{l=1}^{\mathbf{n}} \mathbf{a}_{jl}\phi(\mathbf{x_l}) \tag{3.5}$$

Rewrite (3.5) as:

$$\frac{1}{n}\sum_{i=1}^{\mathbf{n}} \phi(\mathbf{x_i}) \sum_{l=1}^{\mathbf{n}} \mathbf{a}_{jl}\phi(\mathbf{x_i})^T\phi(\mathbf{x_l}) = \lambda_{\mathbf{j}} \sum_{i=1}^{\mathbf{n}} \mathbf{a}_{jl}\phi(\mathbf{x_l}) \tag{3.6}$$

We define the kernel function, $k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_j})^T\phi(\mathbf{x_i})$ and multiply both sides of (3.6) by $\phi(\mathbf{x_k})^T$

$$\frac{1}{n}\sum_{i=1}^{\mathbf{n}} k(\mathbf{x_k}, \mathbf{x_i}) \sum_{l=1}^{\mathbf{n}} \mathbf{a}_{jl}k(\mathbf{x_i}, \mathbf{x_l}) = \lambda_{\mathbf{j}} \sum_{l=1}^{\mathbf{n}} \mathbf{a}_{jl}k(\mathbf{x_k}, \mathbf{x_l}) \tag{3.7}$$

Let K be the matrix $\mathbf{K}_{i,j} = k(\mathbf{x_i}, \mathbf{x_j})$. By substituting K into (3.7) and rearranging it, we get:

$$K^2\mathbf{a_j} = n\lambda_{\mathbf{j}}K\mathbf{a_j} \tag{3.8}$$

To solve $\mathbf{a_j}$ we have to solve the eigenvalue problem of (3.9) for nonzero eigenvalues.

$$K\mathbf{a_j} = n\lambda_{\mathbf{j}}\mathbf{a_j} \tag{3.9}$$

Removing K from both sides of the equation (3.8), affects only the eigenvectors with zero eigenvalue, which will not be principle components anyway.

$$\mathbf{v_j^T v_j} = 1 \Rightarrow \sum_{k,l=1}^{\mathbf{n}} \mathbf{a}_{jl}\mathbf{a}_{jk}\phi(\mathbf{x_l})^T\phi(\mathbf{x_k}) = 1 \Rightarrow \mathbf{a_j}^T K\mathbf{a_j} = 1 \tag{3.10}$$

Multiply (3.9) by $(\mathbf{a_j})$ and using (3.10) we get:

$$\lambda_{\mathbf{j}} n\mathbf{a_j^T a_j} = 1, \forall j \tag{3.11}$$

The dimensionality reduction of the data is achieved by projecting the data onto the first m eigenvectors $\mathbf{v_j}$ when they are ordered in decreasing order according to the magnitude of the eigenvalues as follows:

$$\phi(x)^T \mathbf{v_j} = \sum_{i=1}^{\mathbf{n}} \mathbf{a}_{j_i} \phi(x)^T \phi(\mathbf{x_i}) = \sum_{i=1}^{\mathbf{n}} \mathbf{a}_{j_i} K(x, \mathbf{x_i}) \tag{3.12}$$

At the beginning we assumed that the projected data have zero mean. However, if the data is not centered in D we can use the centralized version of $\widetilde{K}$:

$$\widetilde{K} = K - \mathbf{1_N} K - K \mathbf{1_N} + \mathbf{1_N} K \mathbf{1_N} \tag{3.13}$$

To summarize, the Kernel PCA dimensionality reduction algorithm is composed of the following steps:

1. Choose a kernel function and compute the kernel matrix.
2. Center the kernel matrix.
3. Solve the eigenvalue problem of the centered matrix.
4. Project the data onto the first m eigenvectors $\mathbf{v_j}$

3.3 ISOMAP

The ISOMAP algorithm [75] is fundamental dimensionality reduction algorithm. ISOMAP constructs a low-dimensional representation which best preserves the pair-wise geodesic distances among the data points.The geodesic distance is the weight of the shortest path which goes through points in the dataset. ISOMAP calculated the pair-wise geodesic distances matrix and then it applies the MDS algorithm to this matrix.

Multidimensional Scaling (MDS)[76] is another widely used approach that maps dataset from the original high dimensional space to a lower dimensional space, but does so by preserving all pair-wise distances among the points in the data. MDS finds a low dimensional representation of a dataset given a proximity matrix (similarity or dissimilarity) of the dataset.

ISOMAP algorithm has been successfully applied to many domains such as face recognition [77], video and image analyses [78] and more.

Chapter 4

Ensemble classifiers via Kernel PCA

4.1 The proposed algorithm

We present a new framework for the construction of ensemble classifiers using KPCA. Our algorithm consists of two stages. The first one reduces the dimension of the data by applying KPCA, in various manners to obtain a set of different dimension-reduced versions of the original dataset. The second trains the ensemble members using a single induction algorithm. We use the Nearest Neighbors [79] and the CART Decision Tree [80] induction algorithms, however, other induction algorithms can be used. Each member is trained based on a unique dimension-reduced version of the original dataset.

Fig. 4.1 illustrates the training process of our algorithm.

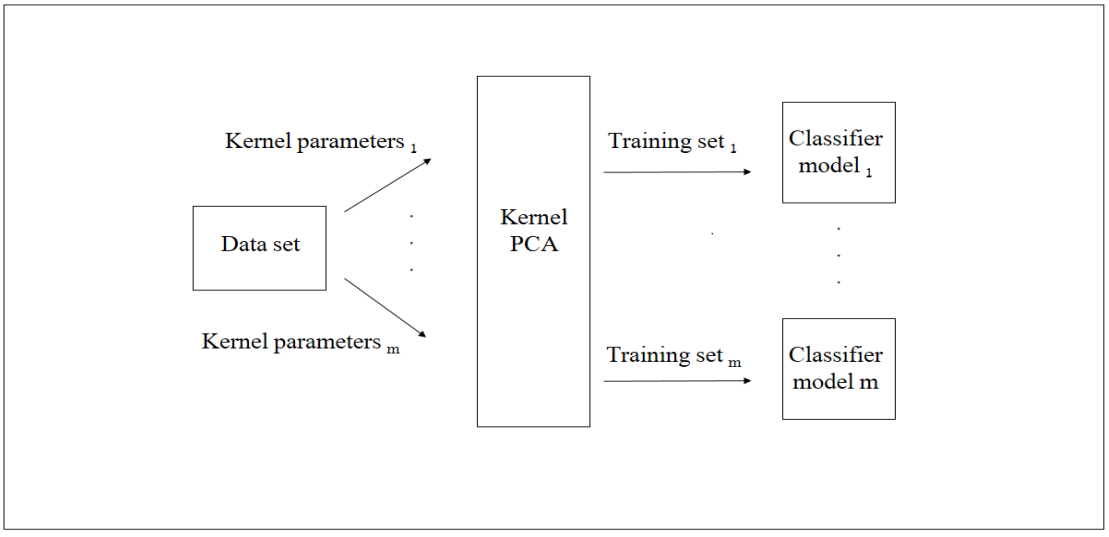

FIGURE 4.1: Training pipeline

Test samples are classified by first embedding them into the dimension reduced spaces of the ensemble members and then applying each ensemble member to the embedded sample. The final classification is obtained via voting. The classification process of a test sample is depicted in Fig. 4.2

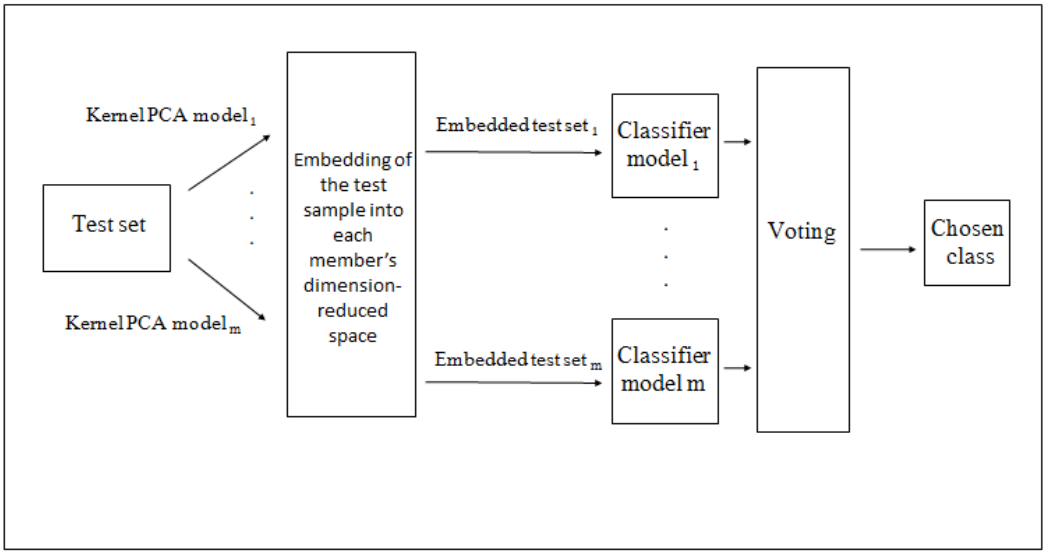

FIGURE 4.2: Prediction pipeline

Cross-validation is a statistical well known method which is a resampling procedure used to evaluate machine learning models on a limited data sample. In order to estimate the skill of the proposed algorithm we used Cross-validation.

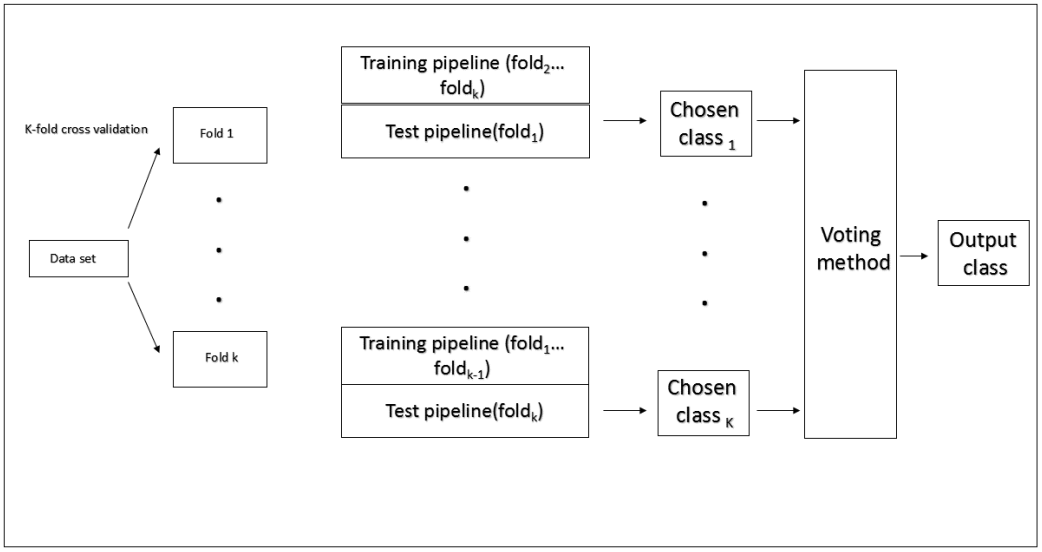

FIGURE 4.3: The testing process

4.2 Different versions of the proposed algorithm

All the experiments we conducted shared the following details (these settings were empirically derived after we tested a wide range of experimental settings): (a) the number of ensemble members was set to ten; (b) the reduced dimension was set to either ten or to half of the dimension of the original space. All experiments were ran using stratified 10-fold cross validation. The key difference between the experiments was the kernel we used. As mentioned Sec in 1.1, accuracy and diversity are critical components for constructing an effective ensemble classifier. Applying KPCA can achieve both. We tested three kernels: the RBF, the Polynomial and the Sigmoid. The kernel parameter values were selected using a data-driven scheme that was tailored for each kernel.

4.2.1 The RBF kernel

The first experiment investigated the RBF kernel which has one free parameter σ as mentioned in Sec 2.2.1. In order to create a diverse and accrue ensemble, we chose for each member in the ensemble a different value for σ. The σ was chosen as follows: first, we calculated the average of all pair-wise Euclidean distances. We denote it by avg. Next, we drew a number between 1.0 to 3.0 from a uniform distribution, number r. Finally, we set σ to be $1.0/avg^r$

4.2.2 The Polynomial kernel

The Polynomial kernel (Sec. 2.2.2), has three immutable parameters: d, γ and c. The values for these parameters were obtained in the following manner: first, we calculated the average- avg and the maximum- max of all pair-wise Euclidean distances. The degree d was set to 3 in all the experiments and γ was chosen to be $\frac{1}{0.5max}$. The diversity was obtained through randomization of the constant c. Specifically, after we calculated avg we multiplied it by a random number that is uniformly drawn in the range $[0.5, 1.5]$.

4.2.3 The Sigmoid kernel

The Sigmoid kernel (Sec. 2.2.3) contains two free parameters γ and c. These parameters generally have the best result when $\gamma > 0$ and $c < 0$ [64]. Here, we also calculated the average- avg of all pair-wise Euclidean distances in order to determine the γ value. We set γ to $1.0/avg^5$. In order to achieve the required diversity, c was uniformly drawn between -1 and 0..

4.2.4 Combinations of kernels

Each kernel is tailored to a specific geometry of dataset. However, sometimes the geometry is composed of different geometries. Accordingly, we investigated combinations of kernels to address such cases. There is an infinite number of kernel combinations that yield a kernel which is valid for KPCA i.e. symmetric and positive semi-definite [67]. In particular, sum and product of two kernels is a valid kernel. We tried three combinations: (a) the sum of the RBF and the Polynomial kernels; (b) the sum of the RBF and the Sigmoid kernels; (c) the product of the RBF and Polynomial kernels.

Chapter 5

Experimental Results

5.1 Data sets

We evaluated our approach using 19 popular data sets from the UCI Machine Learning Repository website [81], which contains benchmark datasets that are commonly used to evaluate machine learning algorithms. We compared the results to four ensemble algorithms: Bagging, Adaboost, Gradient Boosting and Random Forest. None of the datasets contained missing values. No prior knowledge of the datasets and their domains were used in the experiment. The list of datasets and their properties are summarized in Table 5.1.

5.2 Experimental Framework and Runtime environment

We ran the proposed algorithm on a Windows server, with 64 GB installed memory (RAM) and eight processors. We implemented the proposed algorithm using Python and the Scikit-learn [82] package. Scikit-learn harnesses the Python environment to provide state-of-the-art implementations of many well-known machine learning algorithms, while maintaining an easy-to-use interface tightly integrated into the Python language.

N.	Dataset name	Instances	Features	Labels	Content Referenc
1	diabetes	340590	20	4	https://archive.ics.uci.edu/ml/datasets/diabetes
2	Ecoli	335	7	8	https://archive.ics.uci.edu/ml/datasets/Ecoli
3	Hill Valley without noise	1212	100	2	http://archive.ics.uci.edu/ml/datasets/hill-valley
4	Hill Valley with noise	1212	100	2	http://archive.ics.uci.edu/ml/datasets/hill-valley
5	Ionosphere	351	34	2	http://archive.ics.uci.edu/ml/datasets/Ionosphere
6	iris	150	4	3	http://archive.ics.uci.edu/ml/datasets/iris
7	isolet	7797	617	26	http://archive.ics.uci.edu/ml/datasets/isolet
8	liver-disorders	345	7	2	https://archive.ics.uci.edu/ml/datasets/liver+disorders
9	MultFeat	2000	649	10	https://archive.ics.uci.edu/ml/datasets/Multiple+Features
10	Musk Version 1	476	166	2	https://archive.ics.uci.edu/ml/datasets/Musk+(Version+1)
11	Musk Version 2	6598	166	2	https://archive.ics.uci.edu/ml/datasets/Musk+(Version+2)
12	sat	285	17	2	http://archive.ics.uci.edu/ml/datasets/university
13	segment	2310	19	7	http://archive.ics.uci.edu/ml/datasets/image+segmentation
14	spambase	4601	57	2	https://archive.ics.uci.edu/ml/datasets/spambase
15	Statlog Landsat Satellite	6435	36	7	https://archive.ics.uci.edu/ml/datasets/Statlog+(Landsat+Satellite)
16	Waveform Generator with noise	5000	40	3	http://archive.ics.uci.edu/ml/datasets/waveform+database+generator+(version+2)
17	Waveform without noise	5000	21	3	http://archive.ics.uci.edu/ml/datasets/waveform+database+generator+(version+1)
18	Website Phishing	1353	10	3	https://archive.ics.uci.edu/ml/datasets/Website+Phishing
19	Wine	178	13	3	https://archive.ics.uci.edu/ml/datasets/wine

TABLE 5.1: Data sets details

5.3 Results

Tables 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7 present the results of the proposed algorithms. In each of the tables, the first column specifies the name of the tested dataset and the second to last contains the accuracy of each experiments. The last row is the average accuracy of each algorithm.

We use to following acronyms in tables 5.2-5.7: RBF stands for the ensemble that reduces the dimension using the RBF kernel. Poly stands for the ensemble that reduces the dimension using the polynomial kernel. NN is Nearest Neighbour induction algorithm and DT is Decision Tree (CART) induction algorithm. 10d represents experiments conducted with a reduced dimension of ten, 0.5d represents experiments where the target dimension was half the size of the original dimension. The name of each algorithm contains the kernel name followed by the induction algorithm that was used and the target dimension. For example, *Poly NN 0.5d* stands for the ensemble classifier which uses the nearest neighbor induction algorithm and reduced the dimension using the Polynomial kernel to half of the original dimension.

Table 5.2 shows the results of using the RBF kernel.

Dataset	**RBF NN (10d)**	**RBF NN (0.5d)**	**RBF DT (10d)**	**RBF DT (0.5d)**
diabetes	0.687	0.681	0.725	0.697
Ecoli	0.828	0.815	0.831	0.816
Hill Valley without noise	0.567	0.652	0.831	0.819
Hill Valley with noise	0.667	0.606	0.629	0.827
Ionosphere	0.952	0.940	0.949	0.946
iris	0.966	0.966	0.953	0.953
isolet	0.779	0.913	0.781	0.905
liver-disorders	0.634	0.611	0.699	0.618
MultFeat	0.931	0.953	0.913	0.938
Musk Version 1	0.853	0.874	0.853	0.861
Musk Version 2	0.963	0.956	0.970	0.963
sat	0.857	0.867	0.876	0.834
Statlog Landsat Satellite	0.901	0.913	0.903	0.912
Waveform Generator with noise	0.836	0.827	0.861	0.861
Waveform without noise	0.834	0.832	0.860	0.855
Website Phishing	0.870	0.866	0.867	0.862
segment	0.952	0.952	0.962	0.962
spambase	0.759	0.798	0.800	0.883
wine	0.742	0.723	0.894	0.769
Average Accuracy	**0.820**	**0.829**	**0.850**	**0.857**

TABLE 5.2: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the RBF kernel.

Table 5.3 shows the results obtained by using the Polynomial kernel.

Dataset	Poly NN (10d)	Poly NN (0.5d)	Poly DT (10d)	Poly DT (0.5d)
diabetes	0.680	0.667	0.719	0.693
Ecoli	0.820	0.801	0.783	0.773
Hill Valley without noise	0.528	0.603	0.966	0.593
Hill Valley with noise	0.525	0.522	0.836	0.599
Ionosphere	0.889	0.884	0.891	0.894
iris	0.953	0.967	0.960	0.940
isolet	0.769	0.904	0.733	0.841
liver-disorders	0.600	0.603	0.641	0.612
MultFeat	0.914	0.949	0.903	0.911
Musk Version 1	0.874	0.876	0.864	0.853
Musk Version 2	0.958	0.956	0.960	0.968
sat	0.864	0.873	0.859	0.840
Statlog Landsat Satellite	0.897	0.903	0.876	0.875
Waveform Generator with noise	0.807	0.787	0.846	0.842
Waveform without noise	0.809	0.801	0.843	0.832
Website Phishing	0.862	0.858	0.866	0.859
segment	0.936	0.936	0.926	0.926
spambase	0.767	0.815	0.800	0.916
wine	0.738	0.754	0.892	0.883
Average Accuracy	**0.799**	**0.814**	**0.851**	**0.824**

TABLE 5.3: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the Polynomial kernel.

Table 5.4 presents the results of using the Sigmoid kernel.

Dataset	Sigmoid NN (10d)	Sigmoid NN (0.5d)	Sigmoid DT (10d)	Sigmoid DT (0.5d)
diabetes	0.674	0.680	0.708	0.690
Ecoli	0.811	0.744	0.783	0.747
Hill Valley without noise	0.635	0.627	0.565	0.677
Hill Valley with noise	0.653	0.610	0.588	0.541
Ionosphere	0.886	0.889	0.911	0.921
iris	0.960	0.853	0.947	0.927
isolet	0.752	0.895	0.698	0.759
liver-disorders	0.623	0.627	0.641	0.631
MultFeat	0.930	0.950	0.886	0.898
Musk Version 1	0.834	0.855	0.815	0.794
Musk Version 2	0.955	0.955	0.951	0.962
sat	0.842	0.859	0.807	0.767
Statlog Landsat Satellite	0.902	0.906	0.884	0.879
Waveform Generator with noise	0.811	0.785	0.830	0.839
Waveform without noise	0.807	0.799	0.843	0.842
Website Phishing	0.861	0.857	0.858	0.842
segment	0.965	0.965	0.965	0.958
spambase	0.805	0.820	0.887	0.924
wine	0.741	0.763	0.899	0.876
Average Accuracy	**0.813**	**0.813**	**0.814**	**0.814**

TABLE 5.4: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the Sigmoid kernel.

Tables 5.5- 5.7 present the results of the ensemble classifiers that use a combination of kernels. Table 5.5 shows the results of using a kernel which is the sum of the RBF and Polynomial kernels.

Dataset	RBF + Poly NN (10d)	RBF + Poly NN (0.5d)	RBF +Poly DT (10d)	RBF +Poly DT (0.5d)
diabetes	0.686	0.684	0.687	0.710
Ecoli	0.825	0.778	0.568	0.687
Hill Valley without noise	0.546	0.619	0.662	0.652
Hill Valley with noise	0.523	0.507	0.651	0.641
Ionosphere	0.897	0.875	0.875	0.880
iris	0.960	0.973	0.553	0.56
isolet	0.769	0.903	0.696	0.808
liver-disorders	0.588	0.574	0.632	0.574
MultFeat	0.913	0.947	0.898	0.880
Musk Version 1	0.862	0.865	0.803	0.746
Musk Version 2	0.960	0.956	0.930	0.941
sat	0.874	0.874	0.785	0.752
Statlog Landsat Satellite	0.897	0.901	0.869	0.874
Waveform Generator with noise	0.803	0.782	0.840	0.842
Waveform without noise	0.817	0.803	0.834	0.840
Website Phishing	0.862	0.869	0.828	0.828
segment	0.937	0.937	0.929	0.929
spambase	0.764	0.817	0.823	0.864
wine	0.759	0.760	0.903	0.875
Average Accuracy	**0.802**	**0.812**	**0.777**	**0.796**

TABLE 5.5: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the sum of the RBF and Polynomial kernels.

Table 5.6 presents the results obtained by using a kernel which is the sum of the RBF and Sigmoid kernels.

Dataset	RBF + Sigmoid NN (10d)	RBF + Sigmoid NN (0.5d)	RBF + Sigmoid DT (10d)	RBF + Sigmoid DT (0.5d)
diabetes	0.688	0.668	0.708	0.690
Ecoli	0.826	0.833	0.783	0.747
Hill Valley without noise	0.558	0.601	0.565	0.677
Hill Valley with noise	0.641	0.616	0.588	0.541
Ionosphere	0.949	0.929	0.911	0.921
iris	0.960	0.960	0.947	0.927
isolet	0.796	0.912	0.698	0.759
liver-disorders	0.652	0.611	0.641	0.631
MultFeat	0.927	0.951	0.886	0.898
Musk Version 1	0.851	0.861	0.815	0.794
Musk Version 2	0.964	0.963	0.951	0.962
sat	0.882	0.885	0.807	0.767
Statlog Landsat Satellite	0.905	0.912	0.884	0.879
Waveform Generator with noise	0.844	0.812	0.830	0.839
Waveform without noise	0.823	0.822	0.843	0.842
Website Phishing	0.866	0.864	0.858	0.842
segment	0.941	0.941	0.965	0.958
spambase	0.757	0.785	0.887	0.924
wine	0.746	0.757	0.899	0.876
Average Accuracy	**0.820**	**0.825**	**0.814**	**0.814**

TABLE 5.6: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the sum of the RBF and Sigmoid kernels.

Table 5.7 shows the results of using a kernel which is the product of the RBF and Polynomial kernels.

Dataset	RBF * Poly NN (10d)	RBF * Poly NN (0.5d)	RBF * Poly DT (10d)	RBF * Poly DT (0.5d)
diabetes	0.692	0.673	0.686	0.707
Ecoli	0.687	0.646	0.673	0.610
Hill Valley without noise	0.615	0.574	0.705	0.777
Hill Valley with noise	0.513	0.534	0.556	0.673
Ionosphere	0.863	0.903	0.815	0.875
iris	0.506	0.547	0.507	0.534
isolet	0.747	0.877	0.747	0.844
liver-disorders	0.622	0.574	0.669	0.641
MultFeat	0.924	0.897	0.924	0.928
Musk Version 1	0.867	0.878	0.865	0.837
Musk Version 2	0.957	0.958	0.956	0.962
sat	0.840	0.859	0.874	0.813
Statlog Landsat Satellite	0.870	0.871	0.886	0.892
Waveform Generator with noise	0.829	0.788	0.852	0.854
Waveform without noise	0.818	0.826	0.846	0.841
Website Phishing	0.856	0.852	0.849	0.858
segment	0.940	0.940	0.953	0.953
spambase	0.754	0.766	0.795	0.839
wine	0.768	0.694	0.870	0.787
Average Accuracy	**0.772**	**0.771**	**0.791**	**0.801**

TABLE 5.7: Results of the ensemble classifier that uses the Nearest Neighbor or the Decision Tree (CART) induction algorithms and the product of the RBF and Polynomial kernels.

5.3.1 Results comparison

The results of the experimental study indicate that dimensionality reduction via Kernel PCA is a promising approach for the construction of ensembles of classifiers. We compared the various algorithms according to the steps described in [83]. First, for each dataset, we ranked the algorithm in descending order according to their accuracy. Next, we compared the algorithms according to their average rank. The ensemble classifier that achieved the best average rank was the one that used the RBF kernel, the (CART) decision tree and whose target dimension was ten (Table 5.8).

Using the adjusted Friedman test we rejected the null hypothesis that all methods obtain the same classification accuracy with a confidence level of 95% $\chi^2_{(23)} = 73.031, p - value < 0.0001$. Following the rejection of the null hypothesis, we employed the Holm–Bonferroni post-hoc test which compared the *RBF DT (10d)* algorithm with the other proposed ensemble classifiers (Tables 5.2-5.7). Table 5.8 summarizes this comparison. An asterisk marks the proposed algorithms that were significantly outperformed by the *RBF DT (10d)* algorithm. However, we could not reject at confidence level 95% the null hypothesis that the algorithms without asterisk and the *RBF DT (10d)* have the same accuracy.

Algorithm	Average Rank
RBF DT (10d)	5.76
RBF DT (0.5d)	6.79
Poly DT (10d)	9.66
RBF + Sigmoid DT (10d)*	9.87
RBF + Sigmoid DT (0.5d)*	10.08
RBF NN (0.5d)	10.08
RBF NN (10d)	10.21
RBF * Poly DT (0.5d)	11.18
Poly DT (0.5d)	12.61
RBF * Poly DT (10d)*	12.92
RBF + Poly NN (10d)*	12.95
Sigmoid DT (10d)*	12.95
RBF + Poly DT (0.5d)*	12.97
RBF + Poly NN (0.5d)	13.03
Sigmoid DT (0.5d)*	13.03
Sigmoid NN (0.5d)	13.29
Sigmoid NN (10d)	13.39
Poly NN (0.5d)	13.61
RBF + Poly DT (10d)*	14.42
Poly NN (10d)*	15.11
RBF + Sigmoid NN (0.5d)	15.97
RBF + Sigmoid NN (10d)	16.24
RBF * Poly NN (10d)*	16.76
RBF * Poly NN (0.5d)*	17.13

TABLE 5.8: The proposed algorithms rank

Ensemble classifiers via Kernel PCA rank, the best performing algorithm has the lowest average rank. The asterisk means the rejection the Holm–Bonferroni hypothesis with confidence level of 95%

After we found the proposed method with the best performance, *RBF DT (10d)*, we compared it with four well known ensemble algorithms: Bagging, AdaBoost, Random Forest and Gradient Boosting. All the algorithms were ran with the default values of their parameters. The results of the comparison are given in Table 5.9. We compared these algorithms according to their average rank as we did with the proposed algorithms. The average ranking is given in table 5.10. The *RBF DT (10d)* algorithm achieved the best rank. The null hypothesis that all methods have the same classification accuracy was rejected by the adjusted Friedman test with a confidence level of 95% $\chi^2_{(4)} = 13.63, p - value < 0.009$. Since the null hypothesis was rejected, we employed

the Holm–Bonferroni post-hoc test to compare between the the *RBF DT (10d)* algorithm and the other classifiers. Bagging and AdaBoost were significantly inferior to the *RBF DT (10d)* algorithm. The null hypothesis that the Random Forest and Gradient Boosting algorithms have the same accuracy as *RBF DT (10d)* could not be rejected at confidence level of 90% .

Dataset	RBF DT (10d)	Bagging	AdaBoost	Random Forest	Gradient Boosting
diabetes	0.725	0.717	0.750	0.732	0.759
Ecoli	0.831	0.835	0.828	0.836	0.856
Hill Valley without noise	0.831	0.560	0.534	0.577	0.512
Hill Valley with noise	0.629	0.527	0.517	0.539	0.528
Ionosphere	0.949	0.838	0.895	0.915	0.887
iris	0.953	0.946	0.953	0.946	0.960
isolet	0.781	0.894	0.809	0.891	0.862
liver-disorders	0.699	0.632	0.710	0.652	0.669
MultFeat	0.913	0.946	0.950	0.973	0.955
Musk Version 1	0.853	0.757	0.759	0.767	0.798
Musk Version 2	0.970	0.775	0.820	0.818	0.841
sat	0.876	0.821	0.816	0.794	0.813
Statlog Landsat Satellite	0.903	0.878	0.683	0.882	0.848
Waveform Generator with noise	0.861	0.810	0.781	0.807	0.826
Waveform without noise	0.860	0.822	0.785	0.818	0.819
Website Phishing	0.867	0.874	0.835	0.882	0.848
segment	0.962	0.942	0.528	0.969	0.952
spambase	0.800	0.787	0.917	0.929	0.908
wine	0.894	0.721	0.886	0.905	0.900
Average Accuracy	**0.850**	**0.794**	**0.777**	**0.823**	**0.818**

TABLE 5.9: Comparison between the *RBF DT (10d)* algorithm and the Bagging, AdaBoost, Random Forest and Gradient Boosting algorithms.

Algorithm	Average Rank
RBF DT (10d)	2.29
Random Forest	2.55
Gradient Boosting	2.74
Bagging*	3.71
AdaBoost*	3.71

TABLE 5.10: The rank of 5.9 algorithms

The best performing algorithm has the lowest average rank. The asterisk means the rejection of the Holm–Bonferroni hypothesis with confidence level of 90%

In order to compare all the ensemble classifiers via Kernel PCA versions, Bagging, AdaBoost, Random Forest and Gradient Boosting across all inducers and datasets we used the same stages as applied above. The null hypothesis that all methods have the same accuracy was rejected by the adjusted Friedman test with a confidence level of 95% $\chi^2_{(27)} = 73.45, p - value < 0.0001$. Following the rejection of the null hypothesis, we employed the Holm–Bonferroni post-hoc test 5.11.

Algorithm	Average Rank
RBF DT (10d)	7.05
RBF DT (0.5d)	8.18
RBF NN (0.5d)	11.34
Poly DT (10d)	11.37
RBF + Sigmoid NN (0.5d)	11.53
RBF + Sigmoid NN (10d)	11.63
Random Forest	11.66
RBF NN (10d)	11.79
RBF * Poly DT (0.5d)	13.08
Gradient Boosting	14.08
Poly DT (0.5d)	14.55
RBF + Sigmoid DT (10d)*	14.74
Sigmoid DT (10d)*	14.74
RBF * Poly DT (10d)*	14.76
RBF + Poly NN (0.5d)	15.03
RBF + Sigmoid DT (0.5d)*	15.03
Sigmoid DT (0.5d)*	15.03
Sigmoid NN (0.5d)	15.13
Sigmoid NN (10d)*	15.32
Poly NN (0.5d)	15.61
RBF + Poly NN (10d)*	16.76
Bagging*	16.92
AdaBoost	17.29
Poly NN (10d)*	17.61
RBF +Poly DT (0.5d)*	18.55
RBF +Poly DT (10d)*	18.61
RBF * Poly NN (10d)*	19.24
RBF * Poly NN (0.5d)*	19.39

TABLE 5.11: The rank of all experiments. The asterisk means the rejection of the Holm–Bonferroni hypothesis with confidence level of 90%

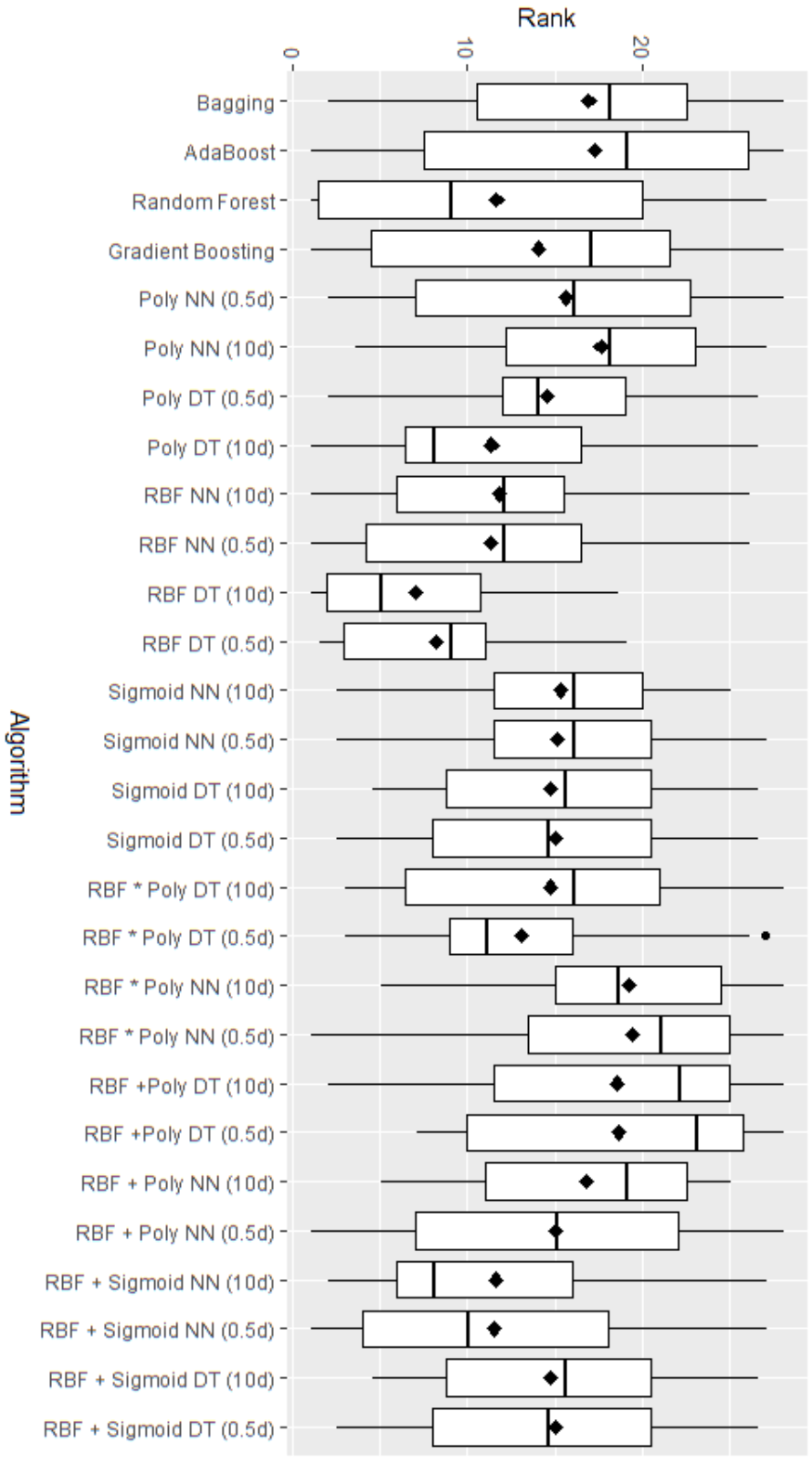

FIGURE 5.1: Ranks results graph

The bottom and top of each box are the 25th and 75th percentile (the lower and upper quartiles respectively). The inner line in each box is the 50th percentile (the median). The inner point is the average and the outer point is an outlier (single data point). The vertical lines that extend to the smallest and larges ranks are not outliers.

Chapter 6

Conclusin and future work

In this paper we presented a new technique for construction of ensembles of classifiers using Kernel PCA. Different dimension-reduced versions of the original training set are constructed using KPCA. The differences among the versions are achieved by running KPCA with different parameter values. Test samples are classified by first embedding them into the dimension reduced spaces of the ensemble members and then applying the ensemble member to the embedded samples. The algorithm is trivially parallel by running each member independently.

We investigated several kernels and used a data driven formula for the kernels free parameters: the RBF kernel, the Polynomial kernel and the Sigmoid kernel. We also investigated combinations of these kernels.

Out of all the models we investigated, the RBF kernel along with the decision tree classifier achieved the best results which are significantly superior compared to the algorithms we used for comparison. We chose the reduced dimension to be either 10 or half the size of the original dimension. However, other values should be investigated or alternatively, find the best dimension for each chosen set of kernel parameter values. Furthermore, we can create unlimited hybrid kernels based on mathematical operations on kernels, combinations and transformations. Finally, other induction algorithms can be investigated as well as a multi-model ensemble based on the proposed approach. This is currently being investigated by the authors. We are confident that this research constitutes the proverbial tip of the iceberg, with the vast potential behind this idea, still submerged, waiting to be discovered.

Bibliography

[1] MLA Polikar and Robi. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[2] Leo Breiman. Stacked regressions. *Machine learning 24.1*, 1:49–64, 1996.

[3] R Clemen. Combining forecasts. *A review and annotated bibliography. Journal of Forecasting*, 5:559–583, 1989.

[4] D. H. Wolpert. Stacked generalization. *Neural networks*, 5:241–259, 1992.

[5] E. Bauer and R Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999.

[6] D. Opitz and R Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11(1):169–198, 1999.

[7] L. K.Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 12(10):993–1001, 1990.

[8] L Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[9] Wyatt J. Harris R. Brown, G. and X Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, 2005.

[10] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[11] B. B Zhou P. Yang, Y. Hwa Yang and A. Y Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):296–308, 2010.

[12] Yoo P. D. Muhaidat S. Karagiannidis G. K. Taha K Al-Jarrah, O. Y. Efficient machine learning for big data: A review. *Big Data Research*, 2(3):87–93, 2015.

[13] Lee D. J. Hong Y. Archibald J Chang, Y. Unsupervised video shot detection using clustering ensemble with a color global scale-invariant feature transform descriptor. *EURASIP Journal on Image and Video Processing*, 2008(1):860743, 2007.

[14] S Avidan. Ensemble tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):261–271, 2007.

[15] Rodríguez J. J. Plumpton C. O. Linden D. E. Johnston S. J et al Kuncheva, L. Random subspace ensembles for fmri classification. *Medical Imaging, IEEE Transactions on*, 29(2):531–542, 2010.

[16] Zheng Y.-T. Tang S. Zhou X. Zhang Y. Lin S. Song, Y. and T.-S Chua. Random subspace ensembles for fmri classification. *Localized multiple kernel learning for realistic human action recognition in videos*, 21(9):1193–1202, 2011.

[17] Nunes U. Oliveira, L. and P Peixoto. On exploration of classifier ensemble synergism in pedestrian detection. *Intelligent Transportation Systems, IEEE Transactions on*, 11(1):16–27, 2010.

[18] B Zhang. Reliable classification of vehicle types based on cascade classifier ensembles. *Intelligent Transportation Systems, IEEE Transactions on*, 14(1): 322–332, 2013.

[19] Lumini A.-Gupta D. Nanni, L. and A Garg. Identifying bacterial virulent proteins by fusing a set of classifiers based on variants of chou's pseudo amino acid composition and on evolutionary information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(2):467–475, 2012.

[20] D. West D P. Mangiameli and R. Rampal. Model selection for medical diagnosis decision support systems. *Decision Support Systems*, 32(3):247–259, 2004.

[21] P. Soda and G Iannello. Aggregation of classifiers for staining pattern recognition in antinuclear autoantibodies analysis. *Information Technology in Biomedicine, IEEE Transactions on*, 13(3):322–329, 2009.

[22] Deng Z. Wong H.-S. Yu, Z. and L Tan. Identifying protein-kinasespecific phosphorylation sites based on the bagging–adaboost ensemble approach. *NanoBioscience, IEEE Transactions on*, 9(2):132–143, 2010.

[23] B. Waske and J. A Benediktsson. Fusion of support vector machines for classification of multisensor data. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(12):3858–3866, 2007.

[24] P. Zhong and R Wang. A multiple conditional random fields ensemble model for urban area detection in remote sensing optical images. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(12):3978–3988, 2007.

[25] L. Rokach. Mining manufacturing data using genetic algorithm-based feature set decomposition. *International journal of intelligent systems technologies and applications*, 4(1-2):57–78, 2008.

[26] Russell Purvis Leigh, William and James M. Ragusa. Forecasting the nyse composite index with technical analysis, pattern recognizer, neural networks, and genetic algorithm: a case study in romantic decision support. *International journal of intelligent systems technologies and applications*, 32(4):361–377, 2002.

[27] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2): 197–227, 1990.

[28] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *Icml*, 96: 148 – 156, 1996.

[29] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123 – 140, 1996.

[30] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[31] Ghosh J Tumer, K. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996.

[32] Thomas G Dietterich. Ensemble methods in machine learning. *In International workshop on multiple classifier systems*, Springer, Berlin, Heidelberg:1–15, 2000.

[33] W. Xu Z. Yang, X. Nie and J. Guo. An approach to spam detection by naive bayes ensemble based on decision induction. *In: Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*, 2:861–866, 2006.

[34] M. Muselli G. Valentini and F. Ruffino. Bagged ensembles of support vector machines for gene expression data analysis. *Neural Networks, 2003. Proceedings of the International Joint Conference on*, 3:1844–1849, 2003.

[35] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[36] Benediktsson J. A.- Sveinsson J. R. Gislason, P. O. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294–300, 2006.

[37] De Andres-S. A. Díaz-Uriarte, R. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.

[38] Detken C. Alessi, L. Identifying excessive credit growth and leverage. *Journal of Financial Stability*, 35:215–225, 2018.

[39] Saha S. Dey-S. R. Khaidem, L. Predicting the direction of stock market prices using random forest. *arXiv preprint arXiv*, 1605.00003, 2016.

[40] Shah S. Thakkar P.- Kotecha K. Patel, J. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4): 2162–2172, 2015.

[41] L. I. Kuncheva J. J. Rodriguez and C. J. Alonso. Rotation forest. A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1619–1630, 2006.

[42] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417 – 441, 1933.

[43] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[44] Schapire R Freund, Y. Experiments with a new boosting algorithm. *In Proceedings of the Thirteenth International Conference on Machine Learning*, page 148–156, 1996.

[45] M. Kearns and L. G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *In Proceedings of the 21st Annual ACM Symposiumon Theory of Computing*, page 433–444, 1989.

[46] Z. H. Zhou. Ensemble methods: foundations and algorithms. *Chapman and Hall/CRC*, 2012.

[47] Leo Breiman. Arcing the edge. *Technical Report 486, Statistics Department, University of California at Berkeley*, 1997.

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

[49] Ridgeway G. The state of boosting. *Computing Science and Statistics*, pages 172–81, 1999.

[50] E. M. Braverman Aizerman, M. A. and L. I. Rozonoer. The probability problem of pattern recognition learning and the method of potential functions. *Automation and Remote Control*, 25:1175–1190, 1964.

[51] I. M. Guyon Boser, B. E. and V. N. Vapnik. A training algorithm for optimal margin classifiers. *In D. Haussler (Ed.), Proceedings Fifth Annual Workshop on Computational Learning Theory (COLT)*, ACM:144–152, 1992.

[52] Koutroumbas K Theodoridis, S. Pattern recognition. *IEEE Transactions on Neural Networks*, 19(2):172–81, 2008.

[53] Bernhard Schölkopf Hofmann, Thomas and Alexander J. Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008.

[54] Koji Tsuda Kashima, Hisashi and Akihiro Inokuchi. Marginalized kernels between labeled graphs. *Proceedings of the 20th international conference on machine learning*, ICML-03, 2003.

[55] Koji Tsuda Kashima, Hisashi and Akihiro Inokuchi. Kernels for graphs. *Kernel methods in computational biology*, 39.1:101–113, 2004.

[56] T Joachims. Learning to classify text using support vector machines: Methods, theory and algorithms. *Norwell: Kluwer Academic Publishers*, 24(186), 2002.

[57] Cristianini N. Duffy-N. Bednarski D. W. Schummer M. Haussler D. Furey, T. S. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 24(16(10)):906–914, 2000.

[58] Duffy N. Collins, M. Convolution kernels for natural language. *In Advances in neural information processing systems*, pages 625–632, 2002.

[59] A Shashua. Introduction to machine learning: Class notes 67577. *arXiv preprint arXiv:0904.3664*, 2009.

[60] Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. *In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM:239–247, 2013.

[61] Cho-Jui; Chang Kai-Wei; Ringgaard Michael; Lin Chih-Jen Chang, Yin-Wen; Hsieh. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11:1471–1490, 2010.

[62] Yoav Goldberg and Michael Elhadad. splitsvm: Fast, space-efficient, non-heuristic. *Polynomial Kernel Computation for NLP Applications*, Proc. ACL-08: HLT, 2008.

[63] V Vapnik. The nature of statistical learning theory. *New York, NY:Springer-Verlag*, 1995.

[64] Lin-C. J. Lin, H. T. A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. *submitted to Neural Computation*, 3:1–32, 2003.

[65] L. O. Jimenez and D. A. Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 28(1):39 – 54, 1998.

[66] K. Fukunaga. Introduction to statistical pattern recognitio. *Academic Press Professional, Inc. San Diego, CA, USA*, 1990.

[67] Pavel Pudil and Jana Novovičová. Novel methods for feature subset selection with respect to problem knowledge. *In Feature extraction, construction and selection Springer, Boston, MA*, pages 101–116, 1998.

[68] A. Elisseeff I. Guyon. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[69] S.D. Bay S. Hettich. The uci kdd archive http://kdd.ics.uci.edu. *University of California, Department of Information and Computer Science, Irvine, CA*, 1999.

[70] Eric Postma Van Der Maaten, Laurens and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[71] K. Pearson. On lines and planes of closest fit to systems of points in spaces. *Philiosophical Magazine*, 2:559–572, 1901.

[72] Alexander Smola Bernhard Schöllkopf and Klaus-Robert Mülller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10: 1299–1319, 1998.

[73] Keechul Jung Kwang In Kim and Hang Joon Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40 – 42, 2002.

[74] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern Recognition*, 40(3): 863–874, 2007.

[75] V. de Silva J. B. Tenenbaum and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[76] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1 – 27, 1964.

[77] M. H. Yang. Face recognition using extended isomap. *In Image Processing. 2002. Proceedings. 2002 International Conference on*, 2:II–II, 2002.

[78] Robert Pless. Image spaces and video trajectories: Using isomap to explore video sequences. *In ICCV*, 3:1433–1440, 2003.

[79] N. S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[80] Friedman J. H. Olshen R. A. Stone C. J. Breiman, L. Classification and regression trees. *Wadsworth International Group*, 1984.

[81] M. Lichman. Uci machine learning repository. 2013.

[82] Alexandre Gramfort Fabian Pedregosa, Gael Varoquaux and Vincent Michel. scikit-learn. 2010.

[83] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, pages 7:1–30, 2006.