VILNIUS UNIVERSITY

Povilas Daniušis

FEATURE EXTRACTION VIA DEPENDENCE STRUCTURE OPTIMIZATION

Doctoral dissertation
Physical sciences, informatics (09P)

Vilnius, 2012

The dissertation work was carried out at Vilnius University from 2008 to 2011.

Scientific supervisor:

doc. dr. Pranas Vaitkus (Vilnius University, physical sciences, mathematics- 01P).

VILNIAUS UNIVERSITETAS

Povilas Daniušis

POŽYMIŲ IŠSKYRIMAS OPTIMIZUOJANT PRIKLAUSOMUMO STRUKTŪRĄ

Daktaro disertacija
Fiziniai mokslai, informatika (09P)

Vilnius, 2012

Disertacija rengta 2008 - 2011 metais Vilniaus universitete.

Mokslinis vadovas:

doc. dr. Pranas Vaitkus (Vilniaus universitetas, fiziniai mokslai, matematika - 01P).

Acknowledgements

I am very grateful to my scientific supervisor dr. Pranas Vaitkus for introducing me into machine learning, and for the freedom he gave me during the studies. I kindly thank dr. Dominik Janzing, prof. Bernhard Schölkopf, and other colleagues from Tübingen for allowing to work with them. It was wonderful experience and great honor to work with you. I sincerely thank prof. Darius Plikynas and mr. Sigitas Kryžius for freedom and support I experienced when working with you. My warm thanks go to Indrė Žliobaitė, for interesting discussions, and for kindly sharing the source code of her thesis.

Ačiū mamai ir močiutei už visokeriopą pagalbą, kurios niekada nepritrūko. Ačiū tau Dalia, ne tik už supratingumą ir palaikymą, bet ir už visas akimirkas praleistas kartu. Ačiū draugams Artūrui, Donatui, Matui, Ugniui už draugystę.

Povilas Daniušis

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine learning is a branch of computer science, which focuses on automatically making decisions from empirical observations. The classification of diseases based on symptoms or predicting the future stock value from historical data are typical examples of machine learning problems. Machine learning algorithm is trained by presenting to it a set of instances, drawn from the usually unknown probability distribution (e.g. pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ is a vector of symptoms, and y_i is a type of a disease). In the training process, the algorithm is expected to learn essential characteristics of the data and to generalize to examples, which were not seen during training. Data sets and the corresponding learning algorithms can be broadly classified into three categories.

Definition 1 (Unsupervised learning)**.** We call a finite collection of i.i.d. observations $\mathbf{x}_i \in \mathcal{X}$ an unsupervised data set. Learning of the model from such a data set is called unsupervised learning. The set $\mathcal{X}$ is called input space.

Definition 2 (Supervised learning)**.** If every instance of an unsupervised data set is associated with some $\mathbf{y}_i \in \mathcal{Y}$, where $\mathbf{y}_i$ depends on $\mathbf{x}_i$, such a data set is called a supervised data set. Supervised learning algorithm estimates the model from such a data set. The set $\mathcal{Y}$ is called output space.

Definition 3 (Semi-supervised learning)**.** If only part of an unsupervised data set is associated with the dependent variable $\mathbf{y}_i \in \mathcal{Y}$, we have a semi-supervised data set. Correspondingly, in semi-supervised learning, the model is constructed from semi-supervised data set.

In many important real world applications, the initial representation of data is inconvenient, or even prohibitive for further analysis. For example, in image analysis, text analysis and computational genetics, high-dimensional, massive, structural, incomplete, and noisy data sets are common. Therefore, *feature extraction*, or the revelation of informative features from raw data is one of the fundamental machine learning problems. Efficient feature extraction helps to understand data and the process that generates it, reduce costs for future measurements and data analysis. The representation of the structured data as a compact set of informative numeric features allows applying well studied machine learning techniques instead of developing new ones.

On the other hand, finding a good representation of data is very domain and application specific. Naturally, a universal and efficient solution does not exist, there is a tradeoff between generality and efficiency: general feature extraction methods (such as principal component or discriminant analysis (73)) often sacrifice the efficiency, while efficient methods are usually not general (e.g. SIFT features in image analysis (54)).

1.1 Application examples

The examples provided below illustrate the practical importance of feature extraction.

Bioinformatics. Computationally, genes are sequences $\mathbb{S} = \{x_1, x_2, ..., x_{n_\mathbb{S}}\}$, where $x_i \in \{A, C, G, T\}$. A, C, G and T corresponds for four nucleotides: adenine, cytosine, guanine and thymine. Proteins can be described by analogous sequence of 20 amino acids. Bioinformatics focuses on the analysis of such high dimensional sequences with the aim of revealing the information on diseases, efficiency of drugs, and other properties of interest. However, dimensionality of such sequences is usually tens of thousands, exceeding the number of observations many times. Therefore, feature extraction is a very important step in analysis. Review of feature extraction methods for bioinformatics is provided in article (70).

Finance. In order to predict a future change in the value of some financial instrument (such as stock or currency), to estimate the probability of bankruptcy, various indicators are considered. Such data can be highly diverse in structure (e.g. textual news, time

series, graphs describing business connections of the company). Determining which of these factors are important helps to make profitable investments or estimate the reliability of a client.

Image analysis, biometrics and robotics. Image is another example of a high dimensional object. However, in recognition problems, training sets are often small, consisting of only a few images per class. In such cases, feature exctraction is often application dependent. For example, in fingerprint recognition, the minutiae and ridge locations are considered as features, SIFT (54) representation of an image is applied for object recognition tasks. In mobile robot navigation and manipulation problems, sensor information is used to build a world model and operate within it. However, high dimensionality and low density of useful information in sensor output requires intensive preprocessing in order to extract useful and interpretable features (e.g. landmarks, obstacles, objects).

Medicine Determining relevant factors, influencing an outcome of treatment, automated diagnostics, analysis of ECG, EEG, MRI and x-ray data are the examples of machine learning problems from medicine, where feature extraction is important.

Internet Most of the information on the web is stored in textual form. Text is an example of a highly structured and high dimensional data object, and its analysis requires specific methods. Web page analysis performed by search engines relies on the link structure, which results in graphical data sets. Another important application is intrusion detection systems, where data are also high dimensional and structured. A review of related algorithms is provided in the article (13).

1.2 Thesis Focus

The dissertation focuses on supervised and semi-supervised feature extraction methods, which optimize the dependence structure of features. The following aspects are taken into consideration:

Relevant features. Quantitative characterization of feature relevance poses the first problem, which should be solved in order to design a feature extraction algorithm. From a general point of view, the notion of informative features intuitively can be related to the notion of dependence. In this dissertation we will follow this approach. Therefore, we need to choose an efficient dependence measure, define a feature relevance functions in terms of this measure, and finally to design an algorithms for feature relevance optimization. According to our approach, the feature relevance function is described by a structure of dependencies. In this thesis two types of dependence structures are analysed: in the first case, we seek features which maximize the dependence on the dependent variable (Section 4.2.1), and in the second one, we additionally minimize the mutual dependence of features (Section 4.3).

Universality. In the introduction we overviewed several examples where feature extraction plays important role. We saw that the data sets may be associated with various machine learning problems (e.g. classification or regression), have complex internal structure (e.g. texts or genetic sequences), and be otherwise inconvenient to work with. In practice, there are many situations when labeled data are expensive to get (e.g. medical and engineering diagnosis, credit scoring etc.). In such a cases, the semi-supervised (see Definition 3) data analysis, which exploits unlabeled training examples is applied. Extension of the suggested algorithms so as to handle such situations is another problem we attempt to solve.

Experimental evaluation. Another issue we focus on is empirical investigation of the efficiency of the considered feature extraction approaches. It is evident that there is no universal way to measure the quality of features. In order to do that, it is useful to assume that the features are the inputs to other machine learning algorithm (e.g. classifier or regression model), which performance is easy to evaluate. In our experiments, the extracted features were classified by k nearest neighbor classifier, and their quality is evaluated by classification performance measures.

1.2.1 Problem statement

Based on the above considerations, we now formulate the problem statement:

Propose and investigate universal supervised and semi-supervised feature extraction algorithms, based on an optimization of dependence structure.

The problem statement of this dissertation covers the following research problems:

Research problem RP1: Propose supervised feature extraction algorithms, based on dependence structure optimization.

Research problem RP2: Construct and experimentally investigate a semi-supervised versions of the suggested algorithms.

Research problem RP3: Experimentally investigate how the quality of features is affected by the estimation of dependence.

Research problem RP4: Experimentally investigate an efficiency of different dependence structures.

Research problem RP5: Experimentally investigate how the suggested algorithms cope with structured and non-linear data sets, and compare them with the existing feature extraction methods.

1.3 Research Methodology

The study of the research problems is conducted using theoretical and empirical research methodology. We will conduct theoretical analysis by reviewing the basic mathematical tools and methods, which will be used in our research. Further, we will review the related work of the other authors, and finally, study the corresponding research problems. The experimental part of the analysis is performed by discussion of the scenario of experiments, analyzed data sets, performance measures, repeatability and parameter selection issues. Afterwards, we provide the empirical results, and analyze how they contribute to the corresponding research problems. Finally, we conclude the thesis by discussing the research problems in light of all the achieved results.

1.4 The Main Contributions and Scientific Novelty

The main contributions of this thesis to the machine learning field are the following:

1. Two new dependence based supervised feature extraction algorithms (HBFE and HSCA) have been suggested;
2. Semi-supervised variants of HBFE and HSCA have been derived;
3. Suggested algorithms have been investigated and compared with alternative ones experimentally, using free access data sets.

This thesis is based on articles (17) and (18). Other related publications by the author are devoted for regression and classification models for matrix data (19), (15), (16), and causal inference (20), (42).

1.5 Statements Presented for the Defence

1. Two new feature extraction algorithms (HBFE and HSCA), based on dependence structure optimization have been suggested;
2. Experiments with binary classification data sets demonstrate that suggested algorithms in certain cases are more efficient than PCA or LDA;
3. Biasedness/unbiasedness of HSIC estimator are essential factor, influencing the efficiency of HBFE and HSCA. Experiments with multi-label classification data demonstrate that $HBFE_1$ is more effient than $HBFE_0$;
4. Experiments with binary classification data show that additional feature interdependency minimization (HSCA) improves their discriminative quality;
5. Experiments with semi-supervised data sets show that in certain cases Laplacian regularization improves the efficiency of HBFE and HSCA.

1.6 Structure of the Thesis

The thesis is structured in the following way. The Chapter 1 introduces the reader to the topic of this dissertation. Mathematical techniques, extensively used in our research are reviewed in Chapter 2. Therein, we briefly restate basic definitions and facts from the theory of positive definite kernels and reproducing kernel Hilbert spaces (RKHS's). Afterwards, we discuss several dependence measures, and introduce the main ingredient of our algorithms, the Hilbert-Schmidt independence criterion (HSIC) (28). Laplacian regularization and multilayer perceptron neural networks are also reviewed therein. Chapter 3 is devoted to a problem of feature extraction. After formulation of core concepts we review existing feature extraction algorithms as examples, illustrating the application of theory from Chapter 2, and discuss practical aspects of performing feature extraction on real data. In Chapter 4, we suggest and discuss new feature extraction techniques based on the optimization of the dependence structure. Semi-supervised and non-linear generalizations using positive definite kernels and artificial neural networks are constructed. Chapter 5 is devoted to computer experiments with proposed feature extraction approaches. We compare various configurations of the proposed schemes and analyze their efficiency on several dozen ordinary and multi-label classification data sets. The thesis is concluded by Chapter 6.

Chapter 2

Mathematical Tools

In this chapter we review mathematical techniques that will be useful in further analysis. We will start with the introduction of kernel methods in Section 2.1, which are extensively used throughout the thesis. Dependence measures are discussed in Section 2.2. Laplacian regularizer, which will be used for deriving semi-supervised extensions of suggested HBFE and HSCA algorithms (Section 4.4), is reviewed in Section 2.3. Finally, in Section 2.4, we shortly overview multilayer perceptron neural networks, as alternative functional models, since in certain situations multilayer perceptrons are advantageous.

2.1 Kernel methods

The theory of positive definite kernels provides a convenient and elegant framework for extending many linear machine learning techniques into nonlinear ones (73). An even more important benefit of kernel methods is that they are applicable for structured data. We will start with the definition of a positive definite kernel.

2.1.1 Positive definite kernels

Definition 4 (Positive definite kernel)**.** Let $\mathcal{X}$ be a set. A real valued, symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel if $\forall x_1, x_2, ..., x_n \in \mathcal{X}$, $\forall \mathbf{c} \in \mathbb{R}^n$,

the inequality $\mathbf{c}^T\mathbf{K}\mathbf{c} \geq 0$ is valid, where $\mathbf{K}$ ($\mathbf{K}_{ij} = k(x_i, x_j)$) is a kernel (or Gram) matrix.

Examples of positive definite kernels are:

1. linear $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T\mathbf{x}'$,
2. polynomial $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + a)^d$, $d \in \mathbb{N}$ and some $a \geq 0$,
3. Gaussian $k(\mathbf{x}, \mathbf{x}') = exp(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{2\cdot\sigma^2})$, $\sigma \in \mathbb{R}$,
4. Laplacian $k(\mathbf{x}, \mathbf{x}') = \frac{c}{2}exp(-c||\mathbf{x} - \mathbf{x}'||)$, $c \in (0, \infty)$,
5. inverted multiquadric $k(\mathbf{x}, \mathbf{x}') = \frac{1}{\sqrt{||x-y||^2+c^2}}$,
6. subset kernel $k(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{m}(1 + \mathbf{x}_i\mathbf{x}'_i)$, where $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^D$;
7. if $(\Omega, \mathcal{F}, P)$ is a probability space, random event kernel $\kappa : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ is defined as $\kappa(A, B) = P(A \cap B) - P(A)P(B)$, where $A, B \in \mathcal{F}$,

and others (see e.g. (25), (9)). The definition of the positive definite kernel does not specify the structure of the data. This property was exploited by various authors, and kernel methods for structured data were proposed (e.g. texts (3),(7), graphs (9), strings (52), tensors (75), etc.). Occasionally, for the sake of brevity, we will refer to positive definite kernels simply as a kernels.

2.1.2 RKHS'es and positive definite kernels

Positive definite kernels are closely related to the reproducing kernel Hilbert spaces (RKHS'es). Let $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Dirac evaluation functional over $\mathcal{H}$ is defined as a linear functional $\delta_x : f \to f(x)$.

Definition 5 (Reproducing kernel Hilbert space (33))**.** $\mathcal{H}$ is called the reproducing kernel Hilbert space (RKHS), if all Dirac evaluation functionals are bounded and continuous.

Theorem 1 (Riesz representation theorem (67))**.** *Suppose $\mathcal{H}$ is a Hilbert space. Then any continuous linear functional ϕ defined on functions of $\mathcal{H}$ admits the following unique representation of the form $\phi(f) = \langle f, g\rangle$ of some $g \in \mathcal{H}$.*

Let us assume, that $\mathcal{H}$ is a RKHS. We will apply Riesz representation theorem for Dirac evaluation functional δ_x. Hence, for any $x \in \mathcal{X}$, and any $f \in \mathcal{H}$ there exists unique $k(x, .) \in \mathcal{H}$, satisfying $f(x) = \delta_x(f) = \langle f, k(x, .)\rangle_{\mathcal{H}}$. Since $k(x, .)$ itself is an element of $\mathcal{H}$ and can be evaluated at every point, by the same argument $\mathcal{H}$ uniquely determines a so called reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ by $k(x, x') = \delta_x(k(x', .)) = \langle k(x', .), k(x, .)\rangle_{\mathcal{H}}$. As an inner product, a reproducing kernel satisfies the definition of positive definite kernel.

On the other hand, it can be shown, that positive definite kernel also uniquely determines RKHS.

Theorem 2 (Moore-Aronszajn theorem (1))**.** *Suppose $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel. Then there exists a unique reproducing kernel Hilbert space $\mathcal{H}$ of functions on $\mathcal{X}$, whose reproducing kernel is k.*

The so-called universal kernels form an important class of positive kernels.

Definition 6 (Universal kernel)**.** The positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called universal if the $span\{k(x, .), x \in \mathcal{X}\}$ is dense in a set of continuous functions, defined on $\mathcal{X}$.

The Gaussian kernel is an example of such a kernel (82).

2.1.3 Kernel trick

A positive definite kernel maps initial data points $x \in \mathcal{X}$ into elements of the corresponding RKHS via the feature map $x \to \phi(x) := k(x, .)$. According to the properties of $\phi(x)$ mentioned above the inner products between the mappings of any $x_i, x_j \in \mathcal{X}$ can be calculated by evaluating a positive definite kernel (i.e. $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j)\rangle$), which is widely exploited in numerous statistical and machine learning techniques and is known as the kernel trick.

Example. Let us consider a quadratic kernel

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2 = \sum_{i,j=1}^{m} x_i x_j x'_i x'_j + \sum_{i=1}^{m} \sqrt{2c}x_i\sqrt{2c}x'_i + c^2, \tag{2.1}$$

where $\mathbf{x}$,$\mathbf{x}' \in \mathbb{R}^{D_x}$, and $c \in \mathbb{R}$.

The corresponding feature map is

$$\phi(\mathbf{x}) = (x_1x_1, ..., x_1x_{D_x}, \sqrt{2c}x_1, ..., \sqrt{2c}x_{D_x}, c)^T.$$

It maps D_x-dimensional inputs $\mathbf{x}$ to $D_x^2 + D_x + 1$-dimensional quadratic feature space. According to kernel trick, $\langle\phi(\mathbf{x}), \phi(\mathbf{x}')\rangle = k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^2$ and therefore only $D_x + 1$ multiplications are required to compute inner products in the feature space.

2.1.4 Representer theorem

It is evident from previous considerations that any learning algorithm, which depends only on inner products in linear space, can be transformed into the one that depends only on inner products in RKHS, which are given by values of a corresponding positive definite kernel. The following theorem states that for a wide range of optimization problems, an optimal solution in possibly infinite dimensional RKHS'es can be represented as finite weighted sums of kernel functions.

Theorem 3 (Representer theorem (72))**.** *Suppose we are given a non-empty set $\mathcal{X}$, a positive definite kernel k, $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$, a training sample $(x_i, y_i)_{i=1}^m \in (\mathcal{X} \times \mathbb{R})^m$, a strictly monotonously increasing function g with values in $[0, \infty]$, cost function $c : (\mathcal{X} \cup \mathbb{R}^2) \to \mathbb{R} \cup \{\infty\}$, and a class of functions:*

$$\mathcal{F} = \{f \in \mathbb{R}^{\mathcal{X}} | f(.) = \sum_{i=1}^{\infty} \beta_i k(., z_i), \beta_i \in \mathbb{R}, z_i \in X, ||f||_{H_k} < \infty\}.$$

Then any $f \in \mathcal{F}$, minimizing the regularized risk functional

$$c((x_1, y_1, f(x_1)), ..., (x_m, y_m, f(x_m))) + g(||f||_{H_k})$$

admits the following representation:

$$f(.) = \sum_{i=1}^{m} \alpha_i k(., x_i),$$

where $\alpha_i \in \mathbb{R}, x_i \in \mathcal{X}$.

2.2 Dependence Measures

It is known from the basic probability theory, that random variables $X \sim F_X$ and $Y \sim F_Y$, $X, Y \in \mathbb{R}^k$ are independent if joint cumulative distribution function can be factored $F_{XY}(x, y) = F_X(x)F_Y(y)$. This property generalizes to more than two random variables. On the other hand, quantitative characterization of dependence is much more difficult.

Quantitative characterization of the dependence between two random variables is an important problem both from theoretical and practical points of view. Several dependence measures have been proposed by various authors (see e.g. thesis (46)). Further we will review several examples, mainly focusing on kernel-based measures, which we will use for feature extraction in Chapter 4.

2.2.1 Mutual information and total correlation

Let X and Y be two random variables with densities p_X, p_Y and the joint density $p_{X,Y}$. The dependence between them can be characterized by the mutual information, expressed by

$$I(X, Y) = D_{KL}(p_{X,Y}||p_X p_Y), \tag{2.2}$$

where $D_{KL}(p, q) = \int p(x) \log \frac{p(x)}{q(x)} dx$ is Kullback-Leibler divergence (14).

If X and Y are Gaussian, the mutual-information is given by

$$I(X, Y) = \frac{1}{2} \log \left(\frac{|\mathbf{C}_{xx}||\mathbf{C}_{yy}|}{|\mathbf{C}|} \right), \tag{2.3}$$

where $\mathbf{C}_{xx}$, $\mathbf{C}_{xx}$, $\mathbf{C}$ are covariance matrices of X, Y and $[X, Y]^T$, and $|.|$ is a determinant.

Mutual information is equal to zero if and only if X and Y are independent.

In the case of random vector, an analogous measure is often referred to as a total correlation. Let $p_{X_1,...,X_n}$ be a joint distribution of n random variables $X_1, ..., X_n$. Then, the total correlation is given by $C(X_1, ..., X_n) = D_{KL}(p_{X_1,...,X_n}||p_{X_1}, ..., p_{X_n})$. Despite popularity of mutual information in broad spectrum of applications, it fails to cope with structured data, and even in vectorial case a density estimation becomes more difficult as dimensionality grows.

2.2.2 Distance correlation

Having three i.i.d. pairs of random vectors, (X, Y), (X', Y') and (X'', Y''), distance correlation is defined as (84), (29)

$$dCorr(X, Y) = \frac{dCov(X, Y)}{\sqrt{dVar(X) \cdot dVar(Y)}}, \tag{2.4}$$

where distance covariance and distance variance are given by

$$dCov(X, Y)^2 = \mathbb{E}||X-X'||||Y-Y'||+\mathbb{E}||X-X'||\mathbb{E}||Y-Y'||-2\mathbb{E}||X-X'||||Y-Y"||, \tag{2.5}$$

and

$$dVar(X)^2 = \mathbb{E}||X - X'|| + \mathbb{E}||X - X''|| - 2\mathbb{E}||X - X'||||X - X''||. \tag{2.6}$$

Distance covariance (and distance correlation) also becomes zero if and only if the random variables are independent (84). Moreover, by construction of the measure X and Y may have different dimensions.

2.2.3 Correntropy

The linear dependence between two scalar random variables X and Y is measured by the covariance: $cov(X,Y) = \mathbb{E}_{XY}XY - \mathbb{E}_X X \mathbb{E}_Y Y$, or the correlation coefficient $\rho(X,Y) = \frac{cov(X,Y)}{\sqrt{Var(X)Var(Y)}}$, where $Var(X)$ denotes the variance of X. A non-linear generalization of covariance can be achieved by centered correntropy (93), (94).

Let k be a positive definite kernel with a feature map $\phi : X \to k(X,.)$. Since XY is an inner product, by properties of RKHS (see 2.1), we have $XY \to \langle k(X,.), k(Y,.)\rangle = k(X,Y)$. Therefore a covariance may be generalized (93) to the correntropy:

$$U(X,Y) = \mathbb{E}_{xy}k(x,y) - \mathbb{E}_x\mathbb{E}_x k(x,y). \tag{2.7}$$

Similarly to $\rho(X,Y)$, a correntropy coefficient is defined as

$$\eta(X,Y) = \frac{U(X,Y)}{\sqrt{U(X)U(Y)}}. \tag{2.8}$$

Note, that correntropy is defined only in a scalar case. From the practical viewpoint it is very important to be able to work with random vectors, and even more with general data structures.

2.2.4 Hilbert-Schmidt independence criterion

The Hilbert-Schmidt independence criterion (HSIC) is a more general kernel-based dependence measure proposed and investigated in (28), (77). Thereinafter, we will briefly re-introduce the basic concepts of the HSIC.

Let us denote by $\mathcal{X}$ and $\mathcal{Y}$ two sets of arbitrary objects (e.g. real vectors, graphs, strings etc.), from which observations (x,y) are drawn. Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be two positive definite kernels with feature maps $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{Y} \to \mathcal{G}$.

The cross-covariance operator between ϕ and ψ is defined as a linear operator C_{xy} :

$\mathcal{G} \to \mathcal{F}$ (see (38), (77)):

$$C_{xy} := \mathbb{E}_{xy}(\phi(x) - \mathbb{E}_x\phi(x)) \otimes (\psi(y) - \mathbb{E}_y\psi(y)), \tag{2.9}$$

where $\otimes$ is a tensor product, and expectations $\mathbb{E}_{xy}$, $\mathbb{E}_x$ and $\mathbb{E}_y$ are taken according to the joint probability measure P_{xy} and the marginal probability measures P_x and P_y.

Assuming that the RKHS'es $\mathcal{F}$ and $\mathcal{G}$ have orthonormal bases $(u_i)_{i\geq 1}$ and $(v_j)_{j\geq 1}$ respectively, let us define the Hilbert-Schmidt norm of the linear operator $C : \mathcal{G} \to \mathcal{F}$.

Definition 7 ((28))**.** Let $C : \mathcal{G} \to \mathcal{F}$ be a linear operator. Then, provided the sum converges, the Hilbert-Schmidt norm of C, $||C||^2_{HS}$, is defined as

$$||C||^2_{HS} := \sum_{i,j} \langle Cv_i, u_j\rangle^2_{\mathcal{F}}. \tag{2.10}$$

It is evident that the Hilbert-Schmidt norm extends the notion of the Frobenius norm on matrices. The existence of orthonormal bases is guaranteed if $\mathcal{X}$ and $\mathcal{Y}$ are separable (e.g. $\mathbb{R}^n$), and the corresponding positive definite kernels k and l are continuous (33). The Hilbert-Schmidt independence criterion is defined as follows ((28), (77)):

$$HSIC(\mathcal{F}, \mathcal{G}, P_{xy}) := ||C_{xy}||^2_{HS}. \tag{2.11}$$

HSIC can be expressed in an equivalent form by the following formula (28):

$$\begin{aligned} &HSIC(\mathcal{F}, \mathcal{G}, P_{xy}) = \\ &\quad \mathbb{E}_{xx'yy'}[k(x, x')l(y, y')] + \mathbb{E}_{xx'}[k(x, x')]\mathbb{E}_{yy'}[l(y, y')] - \\ &\quad -2\mathbb{E}_{xy}[\mathbb{E}_{x'}[k(x, x')]\mathbb{E}_{y'}[k(y, y')]], \end{aligned}$$

where (x', y') is an independent copy of (x, y). In the case when both feature maps are linear, HSIC is equivalent to the Frobenius norm of the cross-covariance matrix, but in general, when universal kernels (82) are considered, theoretically, HSIC can detect any nonlinear dependence (28).

For the sake of convenience let us further assume that both input and output data is vectorial. Let us denote by $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_m]$ and $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_m]$ two matrices, consisting of i.i.d observations $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$ drawn according to unknown distribution P_{xy}. Let $\mathbf{K} : K_{i,j} = k(x_i, x_j)$, and $\mathbf{L} : L_{i,j} = l(y_i, y_j)$ $(i, j = 1, 2, ..., m)$ be two corresponding Gram matrices. Let us define by $\mathbf{H} = \mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^T$ a centering matrix. There are two empirical estimators of HSIC proposed (see (28), (77)) [1]:

$$HSIC_0(\mathbf{X}, \mathbf{Y}) := (m-1)^{-2} Tr(\mathbf{KHLH}), \tag{2.12}$$

and

$$HSIC_1(\mathbf{X}, \mathbf{Y}) := \frac{1}{m(m-3)} \left(Tr\widetilde{\mathbf{K}}\widetilde{\mathbf{L}} + \frac{\mathbf{1}^T\widetilde{\mathbf{K}}\mathbf{1}\mathbf{1}^T\widetilde{\mathbf{L}}\mathbf{1}}{(m-1)(m-2)} - \frac{2}{m-2}\mathbf{1}^T\widetilde{\mathbf{K}}\widetilde{\mathbf{L}}\mathbf{1}) \right), \tag{2.13}$$

where $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{L}}$ are kernel matrices, with diagonal elements set to 0 (i. e. $\widetilde{\mathbf{K}} = \mathbf{K} - \mathrm{diag}(\mathbf{K})$).

Both estimators are concentrated in sense of the following theorem (see (28) and (77)).

Theorem 4. *Let us assume that k, and l are bounded almost everywhere by 1, and are non- negative. Then for $m > 1$ and all $\delta > 0$, with the probability at least $1 - \delta$ for all P_{xy}*

$$|HSIC_0(\boldsymbol{X}, \boldsymbol{Y}) - HSIC(\mathcal{F}, \mathcal{G}, P_{xy})| \leq \sqrt{\frac{log(\frac{6}{\delta})}{\alpha^2 m}} + \frac{C}{m},$$

and

$$|HSIC_1(\boldsymbol{X}, \boldsymbol{Y}) - HSIC(\mathcal{F}, \mathcal{G}, P_{xy})| \leq 8\sqrt{\frac{log(\frac{2}{\delta})}{m}},$$

where α^2 and C are constants.

The (2.12) is biased with an $O(m^{-1})$ bias, and the (2.13) is an unbiased estimator of HSIC (28), (77).

[1] Further if biasedness will not an essential aspect, we also occasionally denote an estimator of HSIC as $\widehat{HSIC}$

Note, that HSIC corresponds to a sum of squared singular values of cross-covariance operator. Another similar dependence measure, COCO (39), corresponds to largest singular value of cross covariance operator. An interpretation of COCO is maximal covariance between $f(X)$ and $g(Y)$, $f \in \mathcal{F}$, and $g \in \mathcal{G}$.

2.3 Laplacian Regularization

In this section we will review the Laplacian regularization, which will be useful in Section 4.4 for an analysis of RP2. Let $T = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{m_l} \cup (\mathbf{x}_i)_{i=m_l+1}^{m}$ be a semi-supervised data set, where m_l denotes a number of labeled instances. We will assume that the variable of interest (e.g. dependent variable, features of $\mathbf{x}$, etc.) is modeled by the functional mapping $\mathbf{f} : \mathbf{x} \rightarrow \mathbf{f}(\mathbf{x})$.

In semi-supervised learning, some form of dependence between input density $p(\mathbf{x})$ and $\mathbf{f} : \mathbf{x}_i \rightarrow \mathbf{f}(\mathbf{x}_i)$ usually is postulated. *Cluster assumption* is one of most popular. It states, that inputs of the same cluster are likely to have similar outputs $\mathbf{f}(\mathbf{x}_i)$ (95) . Let us denote by $\mathbf{F} = [\mathbf{f}(\mathbf{x}_1), ..., \mathbf{f}(\mathbf{x}_m)]^T$, and $\mathbf{D} = \mathrm{diag}(\mathbf{W}\mathbf{1}^T)$. Let $N_p(\mathbf{x}_i)$ be a set of p-nearest neighbors of $\mathbf{x}_i$.

Let us map the input instances $\mathbf{x}_i$ into the vertices of undirected, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$. The edge between two vertices $\mathbf{x}_i$ and $\mathbf{x}_j$ is created if they are considered to be sufficiently similar (e.g. $\mathbf{x}_i \in N_p(\mathbf{x}_j)$ and vice versa). Additionally, the similarity scores are assigned to the edges, e.g.

$$\mathbf{W}_{i,j} = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2), \sigma > 0, \tag{2.14}$$

$$\mathbf{W}_{i,j} = \begin{cases} 1 & \text{if } ||\mathbf{x}_i - \mathbf{x}_j|| \leq \epsilon \\ 0 & \text{otherwise,} \end{cases} \tag{2.15}$$

$$\mathbf{W}_{i,j} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in N_p(\mathbf{x}_j) \text{ or } X_j \in N_p(\mathbf{x}_i) \\ 0 & \text{otherwise.} \end{cases} \tag{2.16}$$

Hence, G represents the training instances $\mathbf{x}_i$ and their similarity structure. Such a

representation is referred to as a *graph embedding* (12). Laplacian regularization relies on the Laplacian matrix of $\mathcal{G}$ by restricting the growth of the regularizers

$$\Omega_0(\mathbf{F}) = \frac{1}{2}\sum_{i,j=1}^{m} ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||^2 \mathbf{W}_{i,j} = Tr(\mathbf{F}\Delta_0\mathbf{F}^T), \tag{2.17}$$

where $\boldsymbol{\Delta}_0 = \mathbf{D} - \mathbf{W}$, or

$$\Omega_1(\mathbf{F}) = \frac{1}{2}\sum_{i,j=1}^{m} ||\frac{f(\mathbf{x}_i)}{\sqrt{\mathbf{D}_{i,i}}} - \frac{f(\mathbf{x}_j)}{\sqrt{\mathbf{D}_{j,j}}}||^2 \mathbf{W}_{i,j} = Tr(\mathbf{F}\Delta_1\mathbf{F}^T), \tag{2.18}$$

where $\boldsymbol{\Delta}_1 = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$.

Δ_0 is called a *graph Laplacian*, and Δ_1 - a *normalized graph Laplacian*. These regularizers are included in the cost function of some supervised learning algorithm. For example, in the case of linear regression we minimize

$$\arg\min_{\mathbf{P}} \sum_{i=1}^{m_l} ||\mathbf{P}^T\mathbf{x}_i - \mathbf{y}_i||^2 + \lambda\Omega_j(\mathbf{P}^T\widetilde{\mathbf{X}}), \tag{2.19}$$

where $\widetilde{\mathbf{X}} = [\mathbf{x}_1, ..., \mathbf{x}_m]$ is full matrix of input instances, $j \in 0, 1$, and $\lambda > 0$. Hence, the Laplacian regularizer penalizes the solution $\mathbf{P}$ more, if for nearby $\mathbf{x}_i$ and $\mathbf{x}_j$ the corresponding features are far from each other.

Laplacian regularization is used in various classification, regression and feature extraction algorithms (e.g. (12), (95), (79)) for deriving semi-supervised modifications.

2.4 Multilayer Perceptron Neural Networks

Another functional model, which we will use in this dissertation, belongs to the class of models known as neural networks. Multilayer perceptron (MLP) is widely applied for various recognition, forecasting, modeling and other problems (60), (63). It stems from early attempts to model biological neurone (65), (58). MLP and it's training backpropagation algorithm was proposed by (68).

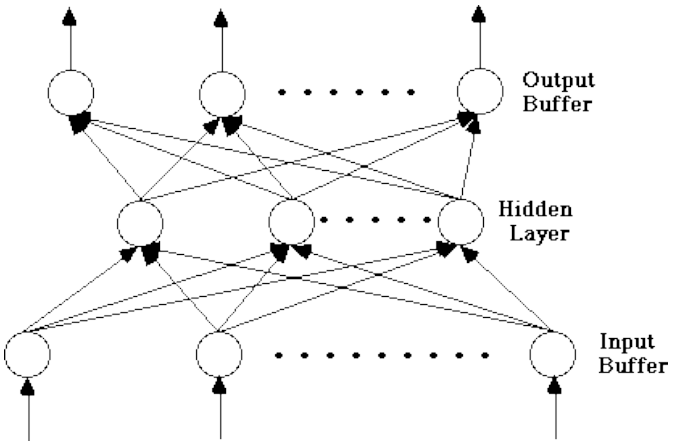

Figure 2.1: An illustration of MLP neural network structure.

The graphical structure of the network is shown in Fig. 2.1. The input layer node receives input vectors $\mathbf{x}$, and passes them into the hidden layer. Each i-th hidden layer node computes activation $\sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i)$, which is further received by output nodes, where weighted sums of hidden layer activations are computed and returned as MLP output. Therefore, MLP's hidden layer can be interpreted as a feature extraction mechanism, which produces features, suitable for linear analysis. Mathematically, MLP with a single hidden layer is defined by the following formula:

$$MLP_j(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{i,j} \sigma(\langle \mathbf{w}_i, \mathbf{x} \rangle + b_i), \tag{2.20}$$

where $\mathbf{w}_{i,j} \in \mathbb{R}^k, \alpha_{i,j}, b_{i,j} \in \mathbb{R}$, σ is the so called activation function, $\langle ., . \rangle$ - an inner product. Popular activation functions are sigmoid or hyperbolic tangent, defined as

$$\sigma_1(x) = \frac{1}{1 + e^{-x}}, \tag{2.21}$$

and

$$\sigma_2(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.22}$$

respectively.

MLP's are the so called universal approximators (61) in the sense that for any continuous function f, defined on a compact set $\mathcal{C} \in \mathbb{R}^n$, and for any continuous, non-polynomial activation function σ, defined on C, it is always possible to find single

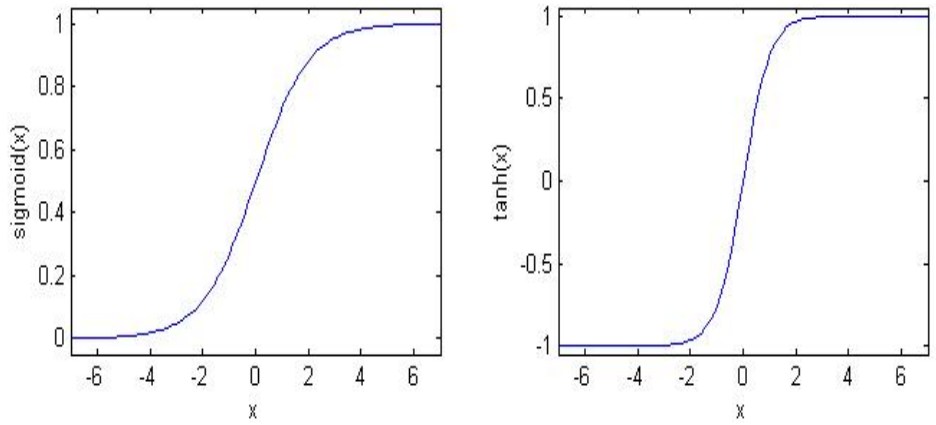

Figure 2.2: The graph of logistic sigmoid and hyperbolic tangent.

hidden layer MLP, such as

$$\sup_{\mathbf{x}\in\mathcal{C}} ||\mathbf{f}(\mathbf{x}) - \mathbf{MLP}(\mathbf{x})|| < \epsilon \tag{2.23}$$

for any fixed $\epsilon > 0$. Despite a universal approximation property, MLP's with two or even more hidden layers sometimes can be practically useful (e.g. autoencoder neural networks discussed in Chapter 3). The parameters of MLP's are usually estimated by gradient-like optimization (e.g. back-propagation algorithm (68), (31), (35)) minimizing some error function. In the case of non differentiable error functions, evolutionary and other biologically inspired algorithms are popular (62), (44). Various MLP modifications are proposed for matrix inputs (15), interval inputs (66), and other structured data.

In general, MLP neural networks do not have such elegant mathematical interpretation as kernel methods 2.1, but in some practical situations they are more attractive. For example, in large-scale learning problems, stochastic gradient updates (31) are very convenient, while the computation of a large kernel matrix can be prohibitively time consuming, MLP makes it possible to easily control the complexity of the model and does not impose any special requirements for the activation function.

Chapter 3

Feature Extraction Algorithms

In this chapter we focus on the general feature extraction problem. We start with reviewing several existing approaches, which also illustrate the theory discussed in Chapter 2. In Section 3.1 and Section 3.2 we discuss one of the most popular unsupervised and supervised feature extraction approaches - principal component analysis and linear discriminant analysis, both in linear and kernel forms. Section 3.3 reviews autoencoder neural networks. Examples of manifold learning algorithms - Laplacian eigenmaps and locally linear embedding (LLE) are discussed in Section 3.4 and Section 3.5. Section 3.6 deals with FOHSIC/BAHSIC as an example of supervised HSIC-based feature selection algorithm. Finally, in Section 3.7 finalizing remarks are provided.

In many important applications (see Chapter 1) the data are high-dimensional (*curse of dimensionality* (36)), structured, non-numerical or has other inconvenient representation, which can result in poor generalization or even make a particular problem intractable. In such a cases, feature extraction can increase the efficiency of various machine learning algorithms, and consequently it is often the first step in computer data analysis. We will start with the definition.

Definition 8 (Feature extraction)**.** Let $\mathcal{X}$ be an input space. A feature extraction is a mapping $\mathcal{M}_{f,d} : x \to \mathbb{R}^d$, where $x \in \mathcal{X}$, $d \in \mathbb{N}$, which is learned from the training data by maximizing an objective function f. Depending on the definition of f, we will call such an algorithm unsupervised, semi-supervised or supervised.

An objective or cost function f imposes desirable properties of the features $\mathcal{M}(x)$. The cost functions are based on very different heuristics. For example, mutual information

and information theoretic approaches are analyzed in (4), (8), (87), papers (10), (69) assumes that the data lie in some low dimensional manifold, (11) extends the manifold assumption to a discriminative scenario, meanwhile, articles (8), (17), (18), (80), (87), and (99) focus on approaches related to dependence maximization.

In the case when $\mathcal{M}$ only selects a subset of x it is called feature selection. Feature selection and extraction approaches are broadly classified as wrappers, filters and hybrid ones (30), (21). Wrappers are methods coupled with particular learning algorithm (e.g. classifier), ant their cost function is associated with its performance. Filters are more general algorithms that do not rely on any knowledge about learning algorithm to be used further. Hybrid algorithms, conceptually, are somewhere between them. They use evaluation criteria from both models. Dimensionality reduction (55) is another term synonymous to feature extraction, but rather less general, since it emphasizes that the aim is to reduce data dimensionality.

For the convenience, further we will assume that $\mathcal{X}$ and $\mathcal{Y}$ consists of real vectors. Let $T = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m$ be a supervised training set of i.i.d. observations, $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$.

Let us denote the matrix of input instances $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]$, the matrix of output instances $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_m]$, and the matrix of extracted features $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_m]$. Let $\mathbf{X}_{\mathcal{S},:}$ be a sub-matrix of $\mathbf{X}$, constructed of the rows of $\mathbf{X}$, which have the indices from $\mathcal{S}$, $1 \leq \mathcal{S}_i \leq D_x$. Similarly, $\mathbf{X}_{:,\mathcal{S}}$ denotes a sub-matrix of columns.

Thereinafter we will review several examples of existing feature extraction algorithms. The reader can find quite a comprehensive review in the book (30).

3.1 Principal Component Analysis

Let us assume that zero mean i.i.d. observations $\mathbf{x}_i$ are drawn from the multivariate distribution $p(\mathbf{x})$. In principal component analysis (43) we seek an orthogonal $D_x \times d_x$, $d_x \leq D_x$ projection matrix $\mathbf{P}$, which maximizes the variance of projected d_x dimensional data (see Fig. 3.1). Having an observation matrix $\mathbf{X}$, the objective function of PCA is represented by formula

$$\arg\max_{\mathbf{P}} \frac{1}{2m} Tr(\mathbf{X}^T \mathbf{P}\mathbf{P}^T \mathbf{X}), s.t. \mathbf{P}^T\mathbf{P} = \mathbf{I}. \tag{3.1}$$

An optimal $\mathbf{P}$ can be found by using the Lagrangian method. The Lagrangian function is given by

$$L(\mathbf{P}) = \frac{1}{2m} Tr(\mathbf{X}^T \mathbf{P}\mathbf{P}^T \mathbf{X} - \mathbf{\Lambda}(\mathbf{P}^T\mathbf{P} - \mathbf{I})), \tag{3.2}$$

where $\mathbf{\Lambda}$ is the diagonal matrix of Lagrange multipliers.

Setting $\nabla L(\mathbf{P}) = \frac{1}{m}\mathbf{X}\mathbf{X}^T\mathbf{P} - \mathbf{\Lambda}\mathbf{P}$ equal to zero, we obtain that the solution consists of eigenvectors of the empirical covariance matrix $\frac{1}{m}\mathbf{X}\mathbf{X}^T$, corresponding to d_x largest eigenvalues. Features of PCA, $\mathbf{F} = \mathbf{P}^T\mathbf{X}$, are called *principal components*. Principal components can be found iteratively by an Algorithm 1 (22).

Algorithm 1 $IterativePCA(\mathbf{X}, d_x)$

Input: Data $\mathbf{X}$, dimensionality d_x.
Output: Projection matrix $\mathbf{P}$.

1. Set $\mathbf{X}_1 = \mathbf{X}$
2. **for** $i \in \{1, ..., d_x\}$
   (a) select $\mathbf{p}_i$ as a principal eigenvector of $\mathbf{X}_i\mathbf{X}_i^T$
   (b) set $\mathbf{X}_{i+1} = (\mathbf{I} - \frac{\mathbf{p}_i\mathbf{p}_i^T}{\mathbf{p}_i^T\mathbf{p}_i})\mathbf{X}_i$
3. **end**;
4. Output projection matrix $\mathbf{P} = \left[\mathbf{p}_1, ..., \mathbf{p}_{d_x}\right]$.

Step (a) can be efficiently carried out by using power iteration (26). Step (b) is known as deflation and performs the projection of the residual matrix $\mathbf{X}_i$ onto subspace, orthogonal to $\mathbf{p}_i$:

$$\mathbf{X}_{i+1}\mathbf{X}_{i+1}^T = \mathbf{X}_i \left(\mathbf{I} - \frac{\mathbf{p}_i\mathbf{p}_i^T}{\mathbf{p}_i^T\mathbf{p}_i}\right)\left(\mathbf{I} - \frac{\mathbf{p}_i\mathbf{p}_i^T}{\mathbf{p}_i^T\mathbf{p}_i}\right)^T \mathbf{X}_i^T = \mathbf{X}_i\mathbf{X}_i^T - \lambda_i\frac{\mathbf{p}_i\mathbf{p}_i^T}{\mathbf{p}_i^T\mathbf{p}_i}, \tag{3.3}$$

and therefore $\mathbf{X}_{i+1}\mathbf{X}_{i+1}^T\mathbf{p}_i = \mathbf{0}$. For $j \neq i$ due to orthogonality of eigenvectors $\mathbf{X}_{i+1}\mathbf{X}_{i+1}^T\mathbf{p}_j = \mathbf{X}_i\mathbf{X}_i^T\mathbf{p}_j$ and therefore Algorithm 1 indeed computes eigenvectors and eigenvalues of $\mathbf{X}\mathbf{X}^T$.

3.1.1 Kernel PCA

In kernel PCA (73) input observations $\mathbf{x}_i$ are mapped into elements of RKHS, and principal component analysis is performed there. Let us consider a positive definite kernel k, with feature map ϕ, which maps input observations $\mathbf{x}_i$ into $\phi(\mathbf{x}_i)$. Since mapped data not necessarily have zero mean, additional care must be taken to perform centering in the feature space. Let us map input instance $\mathbf{x}_i$ into $\phi(\mathbf{x}_i) - m^{-1}\sum_{j=1}^{m}\phi(\mathbf{x}_j)$, and let $\mathbf{H} = \mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^T$ be the centering matrix. Therefore the matrix of input instances $\mathbf{X}$ is mapped into $\mathbf{\Phi H}$ and the objective of PCA in RKHS becomes

$$\arg\max_{\mathbf{P}} \frac{1}{2m} Tr(\mathbf{H}\mathbf{\Phi}^T\mathbf{P}\mathbf{P}^T\mathbf{\Phi}\mathbf{H}), s.t. \mathbf{P}^T\mathbf{P} = \mathbf{I}. \tag{3.4}$$

According to Theorem 3 the solution admits the form $\mathbf{P} = \mathbf{\Phi H Q}$. By kernel trick $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{K}$, and therefore (3.4) can be solved by maximizing Lagrangian

$$L(\mathbf{Q}) = \frac{1}{2m} Tr(\mathbf{HKHQQ}^T\mathbf{HKH} - \Lambda(\mathbf{Q}^T\mathbf{HKHQ} - \mathbf{I})), \tag{3.5}$$

and optimal $\mathbf{Q}$ consists of normalized eigenvectors of $\mathbf{HKH}$, and can be determined incrementally by Algorithm 2.

Algorithm 2 $IterativeKPCA(\mathbf{K}, d_x)$

Input: Kernel matrix $\mathbf{K}$, dimensionality d_x.
Output: Projection matrix $\mathbf{Q}$.

1. Set $\mathbf{K}_1 = \mathbf{HKH}$
2. for $i = 1, ..., d_x$
   (a) select $\mathbf{q}_i$ as principal eigenvector of $\mathbf{K}_i$
   (b) set $\mathbf{K}_{i+1} = (\mathbf{I} - \mathbf{K}_i \frac{\mathbf{q}_i\mathbf{q}_i^T}{\mathbf{q}_i^T\mathbf{K}_i\mathbf{q}_i})\mathbf{K}_i$
3. end;
4. Output projection vectors $\frac{\mathbf{q}_i}{\sqrt{\mathbf{q}_i^T\mathbf{HKHq}_i}}$.

Recent studies on PCA includes probablistic PCA (85), sparse kernel PCA (22), robust PCA (48), (88) and others. Independent component analysis (ICA) (5), (41) is

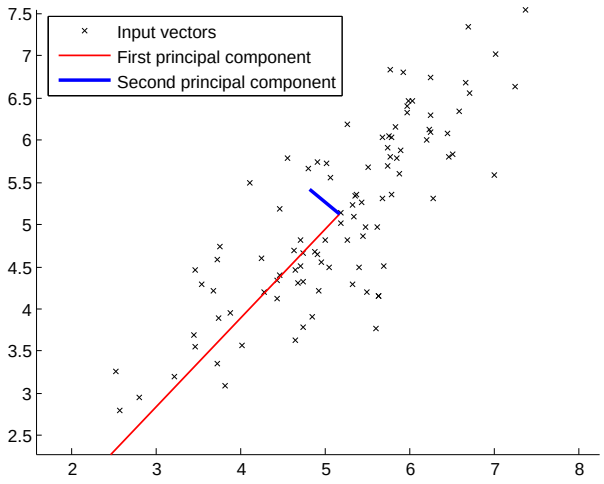

Figure 3.1: PCA projects data into directions of maximum variance.

another related technique. Instead considering uncorrelatedness it seeks independent projections of the data.

3.2 Linear Discriminant Analysis

We now assume supervised scenario with discrete dependent variables, which represents class labels $y_k \in \{1, 2, ..., c\}$. Linear discriminant analysis (LDA) (37), (40) seeks a directions on which the features with different labels are separated while ones with the same label to be close to each other, Fig. 3.2.

Let m_k be a number of observations with label k, $\boldsymbol{\mu}_k = \frac{1}{m_k} \sum_{y_i=k} \mathbf{x}_k$, $\boldsymbol{\mu} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{x}_i$ be intra-class and global means respectively. Between-class, within-class and total scatter matrices are defined

$$\mathbf{S}_b = \sum_{k=1}^{c} m_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T, \tag{3.6}$$

$$\mathbf{S}_w = \sum_{k=1}^{c} \sum_{i:y_i=k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T, \tag{3.7}$$

and $\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w$.

The directions of LDA are obtained by maximizing the Rayleigh quotient:

$$J(\mathbf{p}_i) = \frac{\mathbf{p}_i^T \mathbf{S}_b \mathbf{p}_i}{\mathbf{p}_i^T \mathbf{S}_t \mathbf{p}_i}. \tag{3.8}$$

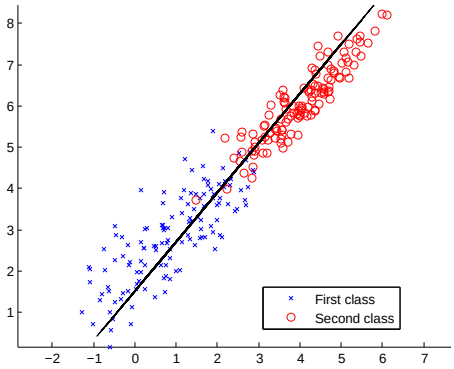

Figure 3.2: LDA projects the data into a linear subspace so that the observations with different labels are separated, and ones with the same label are close to each other.

3.2.1 Kernel LDA

Let us consider positive definite kernel k with feature map ϕ. According to Theorem 3 the solution belongs to linear span of $\mathbf{\Phi}$, $\mathbf{p} = \mathbf{\Phi q}$. Let $\mathbf{S}_b^\phi$ be a map of $\mathbf{S}_b$, and $\mathbf{S}_t^\phi$ be a map of $\mathbf{S}_t$. By the kernel trick, the objective 3.8 in feature space becomes

$$J_\phi(\mathbf{q}_i) = \frac{\mathbf{q}_i^T \mathbf{K}\, \mathbf{W}_b \mathbf{K} \mathbf{q}_i}{\mathbf{q}_i^T \mathbf{K}\, \mathbf{H}\, \mathbf{K} \mathbf{q}_i}, \tag{3.9}$$

where $\mathbf{K}$ is Gram matrix, $\mathbf{W}_b = \sum_{k=1}^{c} m_k (\frac{1}{m_k}\mathbf{e}_k - \frac{1}{m}\mathbf{1})(\frac{1}{m_k}\mathbf{e}_k - \frac{1}{m}\mathbf{1})^T$, centering matrix $\mathbf{H} = \mathbf{I} - m^{-1}\mathbf{1}\mathbf{1}^T$, and binary vectors

$$e_i^k = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{otherwise,} \end{cases} \tag{3.10}$$

3.8 and 3.9 are maximized by solving generalized eigenproblems $\mathbf{S}_b \mathbf{p}_i = \lambda_i \mathbf{S}_t \mathbf{p}_i$ and $\mathbf{W}_b \mathbf{q}_i = \alpha_i \mathbf{W}_t \mathbf{q}_i$. The number of LDA solutions is bounded by $c - 1$.

More efficient modifications of LDA were proposed and investigated by several authors. For example, (12) suggested incremental, semi-supervised, locally sensitive versions

of LDA, sparse LDA was proposed by (71), (22), (79) proposed robust discrimination algorithm, which exploits the structure of the manifold on which data lies.

3.3 Autoencoder Neural Network

Interesting application of MLP neural network for dimensionality reduction are so called autoencoders (34), which can be applied both in unsupervised and supervised scenarios. Let us start with the former. Consider a MLP neural network with three layers, having $N_1 = D_x$, $N_2 = d_x < D_x$, and $N_3 = D_x$ neurones respectively (see Figure 3.3, here $D_x = 5$, $d_x = 3$). Minimizing least squares error

$$\sum_i ||\mathbf{x}_i - \mathbf{MLP}(\mathbf{x}_i)||^2,$$

the data is compressed into d_x numerical features, specified by the outputs of hidden layer. Intuitively, if one could achieve good reconstruction of inputs $\mathbf{x}_i$, the such a features should carry almost the same information, as $\mathbf{x}_i$. In linear case autoencoder is almost identical to PCA.

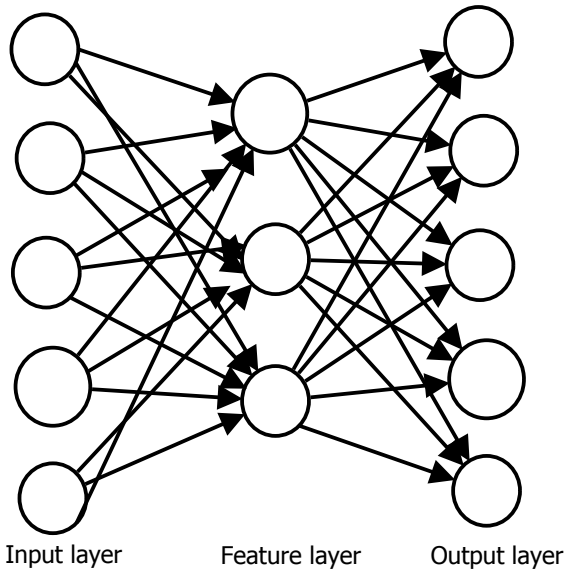

Figure 3.3: Graphical representation of autoencoder.

Conventional MLP with single hidden layer also can be interpreted as an autoencoder, which forms non-linear features in the hidden layer, while in the output layer simultane-

ously classification or regression model is constructed. Recent results on autoencoders and related methods include studies of sparse (50), semi-supervised (76), ensemble (89) models, and restricted Boltzmann machines (34).

3.4 Laplacian Eigenmap and Locality Preserving Projections

The method of Laplacian eigenmap (10) is an example of so called *manifold learning* algorithms. In the cases when the data lie on some non-linear low dimensional manifold embedded into a high dimensional feature space, global techniques such as PCA are inefficient. Laplacian eigenmap is closely related to the Laplacian regularizer Δ_i, reviewed in 2.3. It seeks a minimizer of

$$\Omega_i(\mathbf{F}) = Tr(\mathbf{F}\Delta_i\mathbf{F}^T), i \in \{0, 1\}, \tag{3.11}$$

(see 2.3), with the constraint $Tr(\mathbf{FDF}^T) = \mathbf{I}$, which removes the scaling factor. The locality preserving projections is a linear approximation of Laplacian eigenmap, which assumes that $\mathbf{F} = \mathbf{P}^T\mathbf{X}$, and, therefore, an optimal $\mathbf{P}$ can be deduced by solving a generalized eigenproblem

$$\mathbf{X}\Delta_i\mathbf{X}^T\mathbf{P} = \mathbf{\Lambda X\ D\ X}^T\mathbf{P}, \tag{3.12}$$

and taking d eigenvectors with smallest eigenvalues. This corresponds to a minimum of Laplacian regularizer. Here $\mathbf{\Lambda}$ is a diagonal matrix, consisting of the eigenvalues.

3.4.1 Kernel case

Applying the same argument as in kernel PCA and kernel LDA we map the data instances into a feature space defined by some positive definite kernel k. The kernel

variant of Laplacian eigenmap minimizes objective function

$$\Omega_i(\mathbf{Q}) = Tr(\mathbf{Q}^T\mathbf{K}\boldsymbol{\Delta}_i\mathbf{K}\,\mathbf{Q} - \boldsymbol{\Lambda}\mathbf{K}\mathbf{D}\mathbf{K}), i \in \{0, 1\}, \tag{3.13}$$

and the solution is achieved by eigenvectors of the generalized eigenproblem

$$\mathbf{K}\boldsymbol{\Delta}_i\mathbf{K}\mathbf{Q} = \boldsymbol{\Lambda}\mathbf{K}\mathbf{D}\mathbf{K}\mathbf{Q}, \tag{3.14}$$

which corresponds to the lowest eigenvalues. The idea of Laplacian eigenmap is efficiently exploited in various feature extraction approaches (see e.g. (79)).

3.5 Locally Linear Embedding

Another manifold learning algorithm, locally linear embedding (LLE, (69)), aims to approximate each training instance $\mathbf{x}_i$ by the linear combination of it's neighbors in such a way, that the projections of $\mathbf{x}_i$ are also a well approximated by the linear combination of the projections of $\mathbf{x}_i$'s neighbors.

Mathematically LLE seeks a minimizers of

$$J_0(\mathbf{W}) = \sum_{i=1}^{m} ||\mathbf{x}_i - \sum_{j=1}^{m} \mathbf{M}_{i,j}\mathbf{x}_j||^2, s.t. \mathbf{W}_{i,i} = 0, \sum_j \mathbf{W}_{i,j} = 1, \tag{3.15}$$

and

$$J_1(\mathbf{F}) = \sum_{i=1}^{m} ||\mathbf{f}_i - \sum_{j=1}^{m} \mathbf{M}_{i,j}\mathbf{f}_j||^2, s.t. m^{-1}\mathbf{F}\mathbf{F}^T = \mathbf{I}. \tag{3.16}$$

Eq. is a constrained least squares problem. Entries of $\mathbf{W}$ can be calculated individually (64) by the following procedure.

1. Calculate $m \times m$ distance matrix $\boldsymbol{\Delta}$.

2. For each $\mathbf{x}_i$ store the indices of k nearest neighbors of $\mathbf{x}_i$ into a $m \times k$ matrix $\mathbf{\Gamma}$ (i.e. $\mathbf{\Gamma}_{i,j}$ contains the index of j-th nearest neighbor of $\mathbf{x}_i$).

3. For each $\mathbf{x}_i$ construct a matrix $\mathbf{Q}$ with entries $\mathbf{Q}_{j,l} = \frac{1}{2}(\mathbf{\Delta}_{i,\mathbf{\Gamma}_{i,j}} + \mathbf{\Delta}_{i,\mathbf{\Gamma}_{i,l}} - \mathbf{\Delta}_{\mathbf{\Gamma}_{i,j},\mathbf{\Gamma}_{i,l}})$. Calculate $\boldsymbol{R} = (\mathbf{Q} + \beta\mathbf{I})^{-1}$, where β is suitably chosen regularization parameter (64).

4. Set $\mathbf{W}_{i,\Gamma_{i,j}} = \gamma \sum_{t=1}^{k} \mathbf{R}_{j,m}$, where $\gamma = (\mathbf{1}^T\mathbf{R}\mathbf{1})^{-1}$.

Final embedding of LLE is constructed using the minimizer of Eq. 3.16, which can be found by solving eigenproblem

$$(\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})\mathbf{F} = \mathbf{\Lambda}\mathbf{F}. \tag{3.17}$$

and taking eigenvectors, which correspond to smallest eigenvalues. Here Λ is also a diagonal matrix of eigenvalues. After constructing $\mathbf{F}$ it can be embedded into a linear or non-linear spaces using an appropriate kernel and applying kernel trick.

3.6 FOHSIC and BAHSIC

Forward feature selection procedures start with an empty set and incrementally add features, which optimize some cost function. Conversely, in backward feature selection one begins with all the features and progressively eliminated last useful ones. Both schemes are robust and quite fast. However, they may lead to a different set of features (30). Further, we will review an example of forward/backward selection (77), which maximize the estimator of HSIC between selected features and a dependent variable.

BAHSIC starts with all the features and gradually removes irrelevant ones, whereas FOHSIC starts with an empty set of features and adds those which enhance dependence most. Since both algorithms are analogous, in Algorithm 3 we provide the pseudo-code only for BAHSIC.

To maintain a balance between speed and performance authors (77) recommend at each step consider size of $\mathcal{I}$ so that about 10% of current features are removed. In practice backward procedures tend to perform better than forward ones, possibly due to that they starts with all the information, contained in a training set.

Algorithm 3 BAHSIC($\mathbf{X}$,$\mathbf{Y}$,d_x)
Input: Data matrices $\mathbf{X}$ and $\mathbf{Y}$, feature dimensionality d_x.
Output: Ordered set of features $\mathcal{S}^+$.

1. $\mathcal{S} = \{1, 2, ..., D_x\}$.
2. $\mathcal{S}^+ = \emptyset$.
3. **repeat**
4. $\mathcal{I} := argmax_{\mathcal{I} \in \mathcal{S}} \widehat{HSIC}(\mathbf{X}_{\mathcal{S}/\{\mathcal{I}\}}, \mathbf{Y})$.
5. $\mathcal{S} := \mathcal{S}/\mathcal{I}$.
6. $\mathcal{S}^+ := \mathcal{S}^+ \cup \mathcal{I}$.
7. **until** $|\mathcal{S}^+| = d_x$.

3.7 Conclusion

In this section we reviewed several feature extraction algorithms. PCA and LDA were included as classical examples of unsupervised and supervised feature extraction methods. Despite that they are among the earliest data analysis methods, many important applications benefit from both of them. Autoencoders were reviewed as an application of MLP neural network for supervised-feature extraction. Aforementioned algorithms are global as they do not rely on the local similarities of the data. On the other side, Laplacian eigenmap, LPP, and LLE are local algorithms. They aim to reveal low dimensional representation of the data, that preserve local similarity structures. Finally, FOHSIC/BAHSIC algorithms illustrates an application of HSIC for feature selection. Further we will discuss some practical aspects of feature extraction.

3.7.1 Practical aspects

Determining meta-parameters From practical point of view, the dimension of features, the kernel type and it's parameters (e.g. σ in the case of Gaussian kernel), and other meta-parameters usually strongly affect the performance of further analysis. One of most widely used approaches to estimating such parameters is k-fold cross-validation (31). The idea of k-fold cross validation is to split training data into k folds, train learn-

ing algorithm on $k-1$ folds and test on the remaining ones, repeating the process until the testing of all the training instances has been completed. In the case when $k = m-1$, the process is called leave one out cross validation. Although often efficient in practice, cross validation usually is not computationally cheap procedure. For larger data sets it may become prohibitive. In the cases when cross validation is not acceptable, meta-parameters may be determined quite efficiently using the following approaches:

- When Gaussian kernel is used, to start an analysis by setting the width parameter σ equal to the median of the observation distance (77);
- Several techniques are proposed to estimate the intrinsic dimension of the data (see e.g. (51)). For example, maximum likelihood approach of (51) is based on the idea, that in a small sphere $S_x(R)$ around the observation $\mathbf{x}$, the density $f(x)$ is constant, and treats the observations as a homogeneous Poisson process in $S_x(R)$. The correlation dimension (27) is estimated by regressing the logarithm of $C_m(r) = \frac{2}{m(m-1)} \sum_{i=1}^{m} \sum_{j=i+1}^{m} \delta(||\mathbf{x}_i - \mathbf{x}_j|| < r)$ against $\log r$ over the linear part (here $\delta(.)$ is an indicator function). Despite that cross-validation in many cases is more efficient, dimensionality estimation methods can be helpful for reducing the range of cross validation.
- In most cases simple linear kernel is sufficient. However, in the case of structural or non-linear data sets kernel selection may strongly affect the quality of the features. The kernel function may be chosen according to practitioners intuition, or automatically learned form the data. In such a case kernel is expressed as convex linear combination of different basis kernels, and the weights are learned from the data (see e.g. (49)). This approach is known as multi-kernel learning.

Sparse solutions Kernel feature extraction problems usually are posed as the eigen-decomposition of some data dependent $m \times m$ matrix. Such an operation scales cubically in m. With large m it may become computationally hard, lead to overfitting and other inconveniences. Large kernel matrix is associated with similar effects for most kernel methods as well.

Instead of representing a solution as a sum over all the training examples, only a sparse subset of informative training instances may be analyzed (see Theorem 3). For achieving this purpose, various techniques are suggested.

For example, the algorithm of (6) greedily selects the training instances, which minimize the distance to the linear span of already selected ones. Let us denote the indices of selected data instances by $\mathcal{I}$. The solution of some kernel optimization problem is assumed to admit the form $\mathbf{\Phi}_{:,\mathcal{I}}\mathbf{Q}$.

Similar algorithms are investigated in (24), (92), (74). The approach of (92) generates an index set $\mathcal{I}$ of length d. The entries $\mathcal{I}$ are chosen without replacement from $\{1, 2, ..., m\}$. The approximation of kernel matrix is expressed by

$$\mathbf{K}^* = \mathbf{K}_{:,\mathcal{I}}\mathbf{K}_{\mathcal{I},\mathcal{I}}^{-1}\mathbf{K}_{:,\mathcal{I}}^T, \tag{3.18}$$

where $\mathbf{K}_{:,\mathcal{I}}$ is a sub-matrix of $\mathbf{K}$ with unchanged rows, and columns from $\mathcal{I}$, and $\mathbf{K}_{\mathcal{I},\mathcal{I}}$ is $d \times d$ sub-matrix of $\mathbf{K}$ with both columns an rows from $\mathcal{I}$. (74) suggest to greedily select $\mathcal{I}$, which minimizes Frobenius norm (26) of $\mathbf{K} - \mathbf{K}^*$. The article (24) proposes a generalization of the method of (92), which is based on sampling with replacement form the columns of $\mathbf{K}$. Thesis (22) is devoted to sparse feature extraction.

Chapter 4

HBFE and HSCA algorithms

In this chapter we suggest and discuss several supervised and semi-supervised feature extraction algorithms based on the optimization of the dependence structure. In Section 4.1, we define a dependence structure. In Section 4.2 and Section 4.3, we focus on feature extraction by optimizing two different dependence structures. In Section 4.4, we apply the Laplacian regularizer to derive a semi-supervised versions of suggested algorithms, and in Section 4.5 we discuss non-linear modeling by multilayer perceptron neural networks. Conclusive remarks are provided in Section 4.6.

The results of this chapter are based on publications by the author (17) and (18). The results of Section 4.3 and Section 4.4 are new and previously unpublished.

Suppose we have a supervised training set $T = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m$, where $\mathbf{x}_i \in \mathbb{R}^{D_x}$ are input observations, and $\mathbf{y} \in \mathbb{R}^{D_y}$ are dependent variables. Let us denote the data matrices $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m]$, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_m]$, and feature matrix $\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_m]$.

4.1 Dependence Structure

Let us denote the input and output spaces by $\mathcal{X}$ and $\mathcal{Y}$, and let $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R}^{d_X}\}$ be a space of feature functions. Let $\{d_{\mathcal{F},\mathcal{X}}\}$, $\{d_{\mathcal{F},\mathcal{F}}\}$, $\{d_{\mathcal{F},\mathcal{Y}}\}$ be three sets of dependence measures, defined on the pairs of corresponding sets. Each such a set consists of the dependence measures, that allow to focus on certain dependencies. A triple $\mathcal{D} = \{d_{\mathcal{F},\mathcal{X}}, d_{\mathcal{F},\mathcal{F}}, d_{\mathcal{F},\mathcal{Y}}\}$ defines a *dependence structure*. In order to perform feature

extraction we construct parametrized dependence structure and use it to define and maximize a feature relevance function $R : \mathcal{D} \to \mathbb{R}$.

4.1.1 Dependence measure

The idea of using dependence as a criterion of feature relevance is not new. However, traditionally, dependence-based feature construction relies on information theoretic cost functions (4), (8), (86). Measuring and optimizing such a cost functions often poses serious inconveniences. For example, in mutual information-based algorithms density estimation is required. For high dimensional or structured data, it becomes problematic or prohibitive. Even for cost functions based on lower order non-linear statistics (e.g. quadratic information (87)), the solution is achieved only by numerical methods, and the problem of local optima and meta-parameter selection persists.

We will use an estimator of HSIC as a dependence measure (see Section 2.2.4). The choice of HSIC is motivated by its useful theoretical properties (28), (77), and promising empirical results achieved by various HSIC-based algorithms (e.g. supervised (17, 18), (99), and unsupervised (80), (90) feature extraction, feature selection (47), (57), (77), clustering (59), (78). However, later work of (83) points out that since HSIC is a dependence measure and not conditional dependence measure, it is not capable of performing sufficient dimensionality reduction (i.e. achieve that the input and output variable are conditionally independent given the set of features), which statistically is more well grounded than heuristic approaches.

4.2 HSIC-based Feature Extraction

Further we will analyze a dependence structure, which contains single dependence $d_{\mathcal{F},\mathcal{Y}} := \widehat{HSIC}(\mathbf{f}(\mathbf{X}), \mathbf{Y})$. We will call the corresponding feature extraction method *HSIC-based feature extraction* (HBFE).

4.2.1 HBFE: biased estimator case

The starting point of our work (17) was the method of (99). It seeks linear projections which maximize the biased HSIC estimator (2.12) between features and outputs.

4.2.1.1 Linear case

Let us assume, that for the inputs $\mathbf{K} = \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}$, and for outputs, the kernel can be chosen arbitrarily. The maximum of

$$Tr(\mathbf{KHLH}) = Tr(\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{XHLH}) = \sum_{i=1}^{D_x} \mathbf{p}_i^T\mathbf{XHLHX}^T\mathbf{p}_i, s.t. \mathbf{P}^T\mathbf{P} = \mathbf{I} \tag{4.1}$$

can be achieved by $d_x \leq D_x$ principal eigenvectors of

$$\mathbf{X}\mathbf{M}_0\mathbf{X}^T\mathbf{p} = \lambda\mathbf{p}, \tag{4.2}$$

where

$$\mathbf{M}_0 := \mathbf{HLH}. \tag{4.3}$$

4.2.1.2 Kernel case

Let us now project $\mathbf{x}_i$'s onto RKHS defined by positive definite kernel k with a corresponding feature map $\phi : \mathbf{X}, \rightarrow \Phi$ and maximize 4.1 in the RKHS. According to Theorem 3, the solution admits the form $\mathbf{P} = \Phi\mathbf{Q}$. By the kernel trick we obtain that 4.1 becomes

$$Tr(\mathbf{K}\,\mathbf{Q}\,\mathbf{Q}^T\mathbf{K}\,\mathbf{HLH}) \rightarrow \max_{\mathbf{Q}} s.t. \mathbf{Q}^T\mathbf{KQ} = \mathbf{I}, \tag{4.4}$$

and the corresponding Lagrangian is equal to

$$L(\mathbf{Q}) = Tr(\mathbf{K}\,\mathbf{Q}\,\mathbf{Q}^T\mathbf{KHLH} - \Lambda\mathbf{Q}^T\mathbf{KQ}), \tag{4.5}$$

where $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_{d_x})$.

Having equated $\nabla_{\mathbf{Q}} L(\mathbf{Q})$ to zero, we obtain that optimal $\mathbf{Q}$ consists of eigenvectors of the generalized eigenvalue problem:

$$\mathbf{K}\mathbf{M}_0\mathbf{K}\mathbf{q} = \lambda\mathbf{K}\mathbf{q}, \tag{4.6}$$

where $\mathbf{M}_0$ is defined in Eq. 4.3. If $\mathbf{K}$ is invertible, the solution can be obtained by solving an ordinary eigenvalue problem $\mathbf{M}_0\mathbf{K}\mathbf{q} = \lambda\mathbf{q}$. The maximal value of $HSIC_0$ is equal to $(m-1)^{-2}\sum_{i=1}^{d_x}\lambda_i$.

4.2.2 HBFE: unbiased estimator case

In (17) we extended the work of (99), by analyzing an unbiased (2.13) estimator. We will call the corresponding method $HBFE_1$.

4.2.2.1 Linear case

As in previous section we will assume that $\mathbf{K} = \mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}$, and $\mathbf{L} = \{l(\mathbf{y}_i, \mathbf{y}_j\}_{i,j=1}^m$, where l is an arbitrary positive definite kernel. Let us denote matrices

$$\mathbf{A} = \mathbf{1}\mathbf{1}^T, \mathbf{H} = \mathbf{I} - m^{-1}\mathbf{A},$$

$$\widetilde{\mathbf{K}} = \mathbf{K} - \mathrm{diag}(\mathbf{K}),$$

$$\widetilde{\mathbf{L}} = \mathbf{L} - \mathrm{diag}(\mathbf{L}),$$

and

$$\mathbf{Z} = \widetilde{\mathbf{K}}\widetilde{\mathbf{L}} + \frac{\widetilde{\mathbf{K}}\mathbf{A}\widetilde{\mathbf{L}}\mathbf{A}}{(m-1)(m-2)} - \frac{2\widetilde{\mathbf{K}}\widetilde{\mathbf{L}}\mathbf{A}}{m-2}.$$

We seek a projection matrix $\mathbf{P}$, which projects D_x-dimensional inputs $\mathbf{x}$ onto d_x-dimensional space ($d_x < D_x$), and maximizes the unbiased estimator $HSIC_1$ (2.13). Since

$$HSIC_1(\mathbf{X}, \mathbf{Y}) = \frac{Tr(\mathbf{Z})}{m(m-3)} = \frac{Tr(\mathbf{Z}^T)}{m(m-3)} = \frac{Tr(\frac{\mathbf{Z}+\mathbf{Z}^T}{2})}{m(m-3)}, \tag{4.7}$$

to get an optimal projection matrix, we have to maximize the following constrained objective function:

$$
\begin{aligned}
Tr\left(\frac{\mathbf{Z}+\mathbf{Z}^T}{2}\right) = Tr\big((\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}-\mathbf{D}_K)\widetilde{\mathbf{L}} + \frac{(\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}-\mathbf{D}_K)\mathbf{A}\widetilde{\mathbf{L}}\mathbf{A}}{(m-1)(m-2)} - \\
\frac{(\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}-\mathbf{D}_K)\widetilde{\mathbf{L}}\mathbf{A}+\mathbf{A}\widetilde{\mathbf{L}}(\mathbf{X}^T\mathbf{P}\mathbf{P}^T\mathbf{X}-\mathbf{D}_K)}{m-2}\big) = \\
\sum_{i=1}^{D_x} Tr\big(\mathbf{X}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{X}\widetilde{\mathbf{L}} + \frac{(\mathbf{X}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{X})\mathbf{A}\widetilde{\mathbf{L}}\mathbf{A}}{(m-1)(m-2)} - \frac{\mathbf{X}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{X}\widetilde{\mathbf{L}}\mathbf{A}+\mathbf{A}\widetilde{\mathbf{L}}\mathbf{X}^T\mathbf{p}_i\mathbf{p}_i^T\mathbf{X}}{m-2}\big) - \\
\frac{\mathbf{1}^T\widetilde{\mathbf{L}}\mathbf{1}}{(m-1)(m-2)}\sum_{i=1}^{D_x}\mathbf{p}_i^T\mathbf{X}\mathbf{X}^T\mathbf{p}_i + \frac{2}{m-2}\sum_{i=1}^{D_x}\mathbf{p}_i^T\Big(\sum_{j=1}^{m}\mathbf{x}_j\mathbf{x}_j^T\sum_{k=1}^{m}\widetilde{\mathbf{L}}_{jk}\Big)\mathbf{p}_i = \\
\sum_{i=1}^{D_x}\mathbf{p}_i^T\mathbf{X}\big(\widetilde{\mathbf{L}} + \frac{\mathbf{A}\widetilde{\mathbf{L}}\mathbf{A}-\mathbf{1}^T\widetilde{\mathbf{L}}\mathbf{1}\mathbf{I}}{(m-1)(m-2)} - \frac{\widetilde{\mathbf{L}}\mathbf{A}+\mathbf{A}\widetilde{\mathbf{L}}-2\,\mathrm{diag}(\widetilde{\mathbf{L}}\mathbf{A})}{m-2}\big)\mathbf{X}^T\mathbf{p}_i \\
s.t. \mathbf{P}^T\mathbf{P} = \mathbf{I},
\end{aligned}
\tag{4.8}
$$

where $\mathbf{D}_K = \mathrm{diag}(\mathbf{K})$. Like previously, it can be maximized by the Lagrangian method. Let us define

$$
\mathbf{M}_1 := \widetilde{\mathbf{L}} + \frac{\mathbf{A}\widetilde{\mathbf{L}}\mathbf{A}-\mathbf{1}^T\widetilde{\mathbf{L}}\mathbf{1}\mathbf{I}}{(m-1)(m-2)} - \frac{\widetilde{\mathbf{L}}\mathbf{A}+\mathbf{A}\widetilde{\mathbf{L}}-2\,\mathrm{diag}(\widetilde{\mathbf{L}}\mathbf{A})}{m-2}. \tag{4.9}
$$

It is evident from 4.8 that the solution to this problem consists of d_x eigenvectors of the eigenproblem

$$
\mathbf{X}\mathbf{M}_1\mathbf{X}^T\mathbf{p} = \lambda\mathbf{p}, \tag{4.10}
$$

corresponding to the largest d_x eigenvalues. Eq. 4.7 implies that all the eigenvalues are real. The corresponding optimal value of $HSIC_1$ is equal to $\frac{1}{m(m-3)}\sum_{i=1}^{d_x}\lambda_i$, where λ_i is i-th largest eigenvalue.

4.2.2.2 Kernel case

To derive a kernelized variant of HBFE, let us map original inputs $\mathbf{x}$ to $\boldsymbol{\phi}(x)$ in RKHS associated to some kernel k with feature map $\boldsymbol{\phi} : \mathbf{X} \rightarrow \boldsymbol{\Phi}$. Again, according to Theorem 3, $\mathbf{P} = \boldsymbol{\Phi}\mathbf{Q}$. Therefore, by the kernel trick, the solution can be obtained by solving generalized eigenvalue problem:

$$\mathbf{K}\mathbf{M}_1\mathbf{K}\mathbf{q} = \lambda\mathbf{K}\mathbf{q}. \tag{4.11}$$

In the case when $\mathbf{K}$ is non-singular, the solution is equivalent to the solution of ordinary eigenvalue problem $\mathbf{M}_1\mathbf{K}\mathbf{q} = \lambda\mathbf{q}$.

4.3 Hilbert-Schmidt Component Analysis

Intuitively is clear that good features should be informative and relevant to the variable of interest. On the other hand, the consideration of all such features comes at a price: it increases the dimensionality of the features, and thereby an information becomes coded into a large number of possibly irrelevant, inefficient or redundant patterns. In most cases, redundant features are less robust, have negative effect on the performance of further analysis, increases the storage space and have other undesirable properties (30). Minimum-redundancy maximum-relevance framework (23), (97) encompasses approaches which seeks the features that are not only relevant, but also non-redundant.

Further we will analyze a sequence of dependences $\{d_{\mathcal{F},\mathcal{Y}}, d^t_{\mathcal{F},\mathcal{F}}\}$, where $d_{\mathcal{F},\mathcal{Y}} := \widehat{HSIC}(\mathbf{f}(\mathbf{X}), \mathbf{Y})$, $d^t_{\mathcal{F},\mathcal{F}} = \widehat{HSIC}(\mathbf{f}(\mathbf{X}), [\mathbf{f}_1(\mathbf{X}), ..., \mathbf{f}_{t-1}(\mathbf{X})])$, and t denotes an iteration. We will call the corresponding method *Hilbert-Schmidt component analysis*.

4.3.1 Linear case

Let us first assume that the kernel for inputs is linear. Following the above approach, we will study another feature extraction scheme, which iteratively seeks $d_x \leq D_x$ linear projections, maximizing dependence on the dependent variable y, and simultaneously

minimizing dependence on the already computed projections. In other words, for the t-th feature, we seek a projection vector $\mathbf{p}$, which maximizes the ratio

$$\eta_t(\mathbf{p}) = \frac{\widehat{HSIC}(\mathbf{p}^T\mathbf{X}, \mathbf{Y})}{\widehat{HSIC}(\mathbf{p}^T\mathbf{X}, \mathbf{P}_t^T\mathbf{X})}, \tag{4.12}$$

where $\mathbf{P}_t = [\mathbf{p}_1, ..., \mathbf{p}_{t-1}]$ are projection vectors extracted in previous $t-1$ steps. We will call this approach *Hilbert-Schmidt component analysis* (HSCA).

At the first iteration, only $\widehat{HSIC}(\mathbf{p}^T\mathbf{X}, \mathbf{Y})$ is maximized. Assuming that the kernel matrix $\mathbf{K}$ is linear and plugging an estimator (2.12) into (4.12), we have to maximize the following generalized Rayleigh quotient:

$$\eta_t(\mathbf{p}) = \frac{Tr(\mathbf{X}^T\mathbf{p}\mathbf{p}^T\mathbf{X}\mathbf{H}\mathbf{L}\mathbf{H})}{Tr(\mathbf{X}^T\mathbf{p}\mathbf{p}^T\mathbf{X}\mathbf{H}\mathbf{L}_f\mathbf{H})} = \frac{\mathbf{p}^T\mathbf{X}\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X}^T\mathbf{p}}{\mathbf{p}^T\mathbf{X}\mathbf{H}\mathbf{L}^f\mathbf{H}\mathbf{X}^T\mathbf{p}}, \tag{4.13}$$

where the kernel matrix of features $\mathbf{L}^f_{i,j} = l_f(\mathbf{P}_{t-1}^T\mathbf{x}_i, \mathbf{P}_{t-1}^T\mathbf{x}_j)$, and l_f is arbitrarily chosen positive definite kernel. The maximizer is principal eigenvector of the generalized eigenproblem

$$\mathbf{X}\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X}^T\mathbf{p} = \lambda\mathbf{X}\,\mathbf{H}\mathbf{L}_f\mathbf{H}\mathbf{X}^T\mathbf{p}. \tag{4.14}$$

In certain cases to avoid numerical instabilities instead 4.16 may be useful to consider

$$\mathbf{X}\,\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X}^T\mathbf{p} = \lambda(\mathbf{X}\,\mathbf{H}\mathbf{L}_f\mathbf{H}\mathbf{X}^T + \alpha\mathbf{I})\mathbf{p}, \tag{4.15}$$

where $\alpha \geq 0$ is the regularization parameter.

Having two matrices $\mathbf{A}, \mathbf{B}$, and the vector $\mathbf{u}$ of appropriate dimensionality, let us denote by $\mu(\mathbf{A}, \mathbf{B})$ the principal eigenvector of the generalized eigenproblem $\mathbf{A}\mathbf{u} = \lambda\mathbf{B}\,\mathbf{u}$. On the basis of the observations made above, now we will formulate HSCA algorithm. For the sake of simplicity of mathematical expression, we consider a biased estimator of HSIC (2.12), although the case of an unbiased estimator (2.12) is straightforward.

Algorithm 4 $HSCA(\mathbf{X}, \mathbf{Y}, d_x)$

Input: Data matrices $\mathbf{X}, \mathbf{Y}$, dimensionality d_x.
Output: Projection matrix $\mathbf{P}$.

1. Construct $m \times m$ kernel matrix $\mathbf{L} : \mathbf{L}_{i,j} = l(\mathbf{y}_i, \mathbf{y}_j)$, set $\mathbf{M} = \mathbf{I}$, and $\mathbf{P} = \emptyset$.
2. **for** $t \in \{1, ..., d_x\}$ **do**
3. Update $\mathbf{P} := [\mathbf{P}, \mu(\mathbf{XHLHX}^T, \mathbf{M})]$.
4. Construct $m \times m$ kernel matrix of already extracted projections $\mathbf{L}^f_{i,j} = l_f(\mathbf{P}^T\mathbf{x}_i, \mathbf{P}^T\mathbf{x}_j)$.
5. $\mathbf{M} := \mathbf{XHL}^f\mathbf{HX}^T$.
6. **end**
7. Output the projection matrix $\mathbf{P}$. The features for the input $\mathbf{x}$ are $\mathbf{P}^T\mathbf{x}$.

4.3.2 Kernel case

Let k be a positive definite kernel with a feature map $\boldsymbol{\phi}$ which maps the original inputs $\mathbf{x}$ into the RKHS associated with k. Thus, the input data matrix $\mathbf{X}$ in the feature space is $\boldsymbol{\Phi} = [\boldsymbol{\phi}(x_1), ..., \boldsymbol{\phi}(x_m)]$. According to Theorem 3, in each iteration the maximizer admits the form $\mathbf{p} = \boldsymbol{\Phi}\mathbf{q}$. By the kernel trick, Eq. 4.16 becomes

$$\mathbf{K\,HLH\,Kp} = \lambda\mathbf{K\,HL}^f\mathbf{H\,Kp}, \tag{4.16}$$

and, therefore, Algorithm 5 can be used to compute projections in the kernel case.

Algorithm 5 $KHSCA(\mathbf{X}, \mathbf{Y}, d_x)$

Input: Data matrices $\mathbf{X}, \mathbf{Y}$, dimensionality d_x.
Output: Projection matrix $\mathbf{P}$.

1. Construct $m \times m$ kernel matrices $\mathbf{K} : K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{L} : \mathbf{L}_{i,j} = l(\mathbf{y}_i, \mathbf{y}_j)$, set $\mathbf{M} = \mathbf{K}$, and $\mathbf{P} = \emptyset$.
2. **for** $t \in \{1, ..., d_x\}$ **do**
3. Update $\mathbf{P} := [\mathbf{P}, \mu(\mathbf{KHLHK}, \mathbf{M})]$
4. Construct $m \times m$ kernel matrix of already extracted projections $\mathbf{L}^f_{i,j} = l_f(\mathbf{P}^T\mathbf{x}_i, \mathbf{P}^T\mathbf{x}_j)$.
5. $\mathbf{M} := \mathbf{KHL}^f\mathbf{HK}$.
6. **end**
7. Output the projection matrix $\mathbf{P}$. The features for the input $\mathbf{x}$ are $\mathbf{P}^T[k(\mathbf{x}, \mathbf{x}_1), k(\mathbf{x}, \mathbf{x}_2), ..., k(\mathbf{x}, \mathbf{x}_m)]^T$.

4.4 Semi-supervised HBFE and HSCA

In the following we will address research problem RP2. Let $\mathbf{x}_{m+1}, ..., \mathbf{x}_{m+M}$ be a sequence of additional M observations, for which the dependent variables are not available. Let us denote an extended input matrix by $\widetilde{\mathbf{X}} = [\mathbf{X}, \mathbf{x}_{m+1}, ..., \mathbf{x}_{m+M}]$. Let $\mathbf{W}$ be an $(m + M) \times (m + M)$ similarity matrix, defined as $\mathbf{W}_{i,j} = w(\mathbf{x}_i, \mathbf{x}_j)$, where w is some similarity measure (e.g. Gaussian kernel, if $\mathbf{x}_i \in \mathbb{R}^{D_x}$), and let $\mathbf{D}$ be a diagonal matrix, $\mathbf{D}_{i,i} = \sum_j \mathbf{W}_{i,j}$.

To derive a semi-supervised extension, we will use graph Laplacian 2.3 approach, which imposes additional Laplacian regularization for the projection matrix. Therefore, the aim is to optimize the dependence structure, at the same time preserving the manifold, on which the data lies. In this section we will consider only the linear versions of algorithms, since kernel variants can be analyzed in a completely analogous way as in supervised cases. Let us denote the graph Laplacian by $\mathbf{\Delta}$.

4.4.1 Semi-supervised HBFE

In semi-supervised HBFE, we seek a projection matrix $\mathbf{P}$, which maximizes

$$\widehat{HSIC}((1-\beta)\mathbf{P}^T\mathbf{X}, \mathbf{Y}) - \beta\Omega(\mathbf{P}^T\widetilde{\mathbf{X}}), s.t. \mathbf{P}^T\mathbf{P} = \mathbf{I}, \tag{4.17}$$

where Ω and $0 \leq \beta \leq 1$ are Laplacian regularizer, and regularization parameter. The maximum can be achieved by d_x leading eigenvectors of

$$((1-\beta)\mathbf{X}\mathbf{M}_i\mathbf{X}^T - \beta\widetilde{\mathbf{X}}\boldsymbol{\Delta}\widetilde{\mathbf{X}}^T)\mathbf{p} = \lambda\mathbf{p}, \tag{4.18}$$

where $\mathbf{M}_0$ and $\mathbf{M}_1$ are matrices **??** or **??** in the case of a biased and unbiased estimator respectively.

4.4.2 Semi-supervised HSCA

By incorporating Laplacian regularizer into (4.12), for each feature, we seek a maximizer of

$$\eta_t(\mathbf{p}) = \frac{(1-\beta)\widehat{HSIC}(\mathbf{p}^T\mathbf{X}, \mathbf{Y}) - \beta\Omega(\mathbf{p}^T\widetilde{\mathbf{X}})}{\widehat{HSIC}(\mathbf{p}^T\mathbf{x}, \mathbf{P}_t^T\mathbf{X})}, \tag{4.19}$$

where Ω and $0 \leq \beta \leq 1$ are Laplacian regularizer, and regularization parameter. When plugging an (2.12) estimator of HSIC, for each feature, we have to maximize

$$\frac{\mathbf{p}^T((1-\beta)\mathbf{X}\,\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X}^T - \beta\widetilde{\mathbf{X}}\boldsymbol{\Delta}\widetilde{\mathbf{X}}^T)\mathbf{p}}{\mathbf{p}^T\mathbf{X}\,\mathbf{H}\mathbf{L}^f\mathbf{H}\mathbf{X}^T\mathbf{p}}, \tag{4.20}$$

which can be achieved by the principal eigenvector of the generalized eigenproblem:

$$((1-\beta)\mathbf{X}\mathbf{H}\mathbf{L}\mathbf{H}\mathbf{X}^T - \beta\widetilde{\mathbf{X}}\boldsymbol{\Delta}\widetilde{\mathbf{X}}^T)\mathbf{p} = \lambda\mathbf{X}\,\mathbf{H}\mathbf{L}^f\mathbf{H}\mathbf{X}^T\mathbf{p}. \tag{4.21}$$

Therefore, we now can formulate semi-supervised HSCA (Algorithm 6).

Algorithm 6 $SSLHSCA(\mathbf{X}, \mathbf{Y}, d_x, \beta)$

Input: Data matrices $\mathbf{X}, \mathbf{Y}$, dimensionality d_x.
Output: Projection matrix $\mathbf{P}$.

1. Construct $m \times m$ kernel matrices $\mathbf{K} : K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{L} : \mathbf{L}_{i,j} = l(\mathbf{y}_i, \mathbf{y}_j)$, the graph Laplacian $\mathbf{\Delta}$, set $\mathbf{M} = \mathbf{I}$, and $\mathbf{P} = \emptyset$.
2. **for** $t \in \{1, ..., d_x\}$ **do**
3. Update $\mathbf{P} := [\mathbf{P}, \mu((1-\beta)\mathbf{XHLHX}^T - \beta\widetilde{\mathbf{X}}\mathbf{\Delta}\widetilde{\mathbf{X}}^T, \mathbf{M})]$
4. Construct $m \times m$ kernel matrix of already extracted projections $\mathbf{L}^f_{i,j} = l^f(\mathbf{P}^T\mathbf{x}_i, \mathbf{P}^T\mathbf{x}_j)$.
5. $\mathbf{M} := \mathbf{XHL}^f\mathbf{HX}^T$.
6. **end**
7. Output the projection matrix $\mathbf{P}$. The features for the input $\mathbf{x}$ are $\mathbf{P}^T\mathbf{x}$.

4.5 NeuroHBFE and NeuroHSCA

In this section we will describe a modifications of HBFE and HSCA, where non-linear modeling is achieved by using multilayer perceptron neural networks. In article (18) we suggested and discussed feature extraction schemes, conceptually parallel to HBFE and HSCA.

4.5.1 NeuroHBFE

Let $\mathbf{MLP}(.|\boldsymbol{\theta})$ be a multilayer perceptron neural network with D_x input neurones and d_X output neurones. In NeuroHBFE we maximize

$$\widehat{HSIC}(\mathbf{MLP}(\mathbf{X}|\boldsymbol{\theta}), \mathbf{Y}), \tag{4.22}$$

with respect to parameter the vector $\boldsymbol{\theta}$.

4.5.2 NeuroHSCA

Let $\mathbf{MLP}(.|\boldsymbol{\theta})$ be a multilayer perceptron neural network with D_x input neurones and one output neurones. In NeuroHSCA we iteratively maximize

$$\eta_t(\boldsymbol{\theta}) = \frac{\widehat{HSIC}(\mathbf{MLP}(\mathbf{X}|\boldsymbol{\theta}), \mathbf{Y})}{\widehat{HSIC}(\mathbf{MLP}(\mathbf{X}|\boldsymbol{\theta}), \mathbf{F}_t)}, \tag{4.23}$$

where $\mathbf{F}_t = [\mathbf{MLP}(\mathbf{X}|\boldsymbol{\theta}_1), ..., \mathbf{MLP}(\mathbf{X}|\boldsymbol{\theta}_{t-1})]$ denotes the matrix of previously extracted features.

In both cases the features are given by the outputs of the corresponding MLP's. The optimization of (4.22) and (4.23) rely on numerical methods, which often is problematic due to local optima and meta-parameter selection issues. In (18) we suggested the optimization algorithm, which combines gradient ascent (as a local optimizer) and simulated annealing (45) (as the global one), since such an approach can help to avoid a local optimums. On the other side, in the case of large data sets stochastic gradient updates may be more convenient. Therefore, NeuroHBFE and NeuroHSCA may have certain advantages comparing to their kernel analogues.

4.6 Conclusions

In this chapter we suggested HBFE (HSIC-based feature extraction) and HSCA (Hilbert-Schmidt component analysis) algorithms, which optimize two different dependence structures of the features and the dependent variable. The dependence is evaluated using either biased or unbiased estimator of HSIC. Being a kernel-based measure, HSIC is applicable for non-linear or structural data, it's estimators are convenient for optimization. HBFE seeks features, which maximize the dependence with the dependent variable, while HSCA additionally minimizes the interdependence of the features. Both algorithms are formulated in terms of eigenproblems. We extend HBFE and HSCA for semi-supervised data sets by using Laplacian regularization. In such a case the cost functions of HBFE and HSCA are altered so, that features preserve distance similarity (i.e. similar data instances should be mapped to similar features). We additionally discussed an alternative NeuroHBFE and NeuroHSCA algorithms, where non-linear modeling is achieved by using multilayer perceptron (MLP) neural networks.

Chapter 5

Computer Experiments

This chapter is devoted to computer experiments with feature extraction approaches, described in Chapter 4. A collection of ordinary and multi-label classification data sets is analyzed. The experiments involve the statistical analysis of the classification performance when inputs of the classifier are the features, generated by different methods. The main objective of this chapter is empirical analysis of RP2, RP3, RP4, and RP5:

1. Empirical aspects of RP2 is investigated by conducting the experiments with semi-supervised variants of HBFE and HSCA,
2. RP3 is investigated by studying how the biasedness of HSIC estimator affects the efficiency of HBFE and HSCA,
3. RP4 is analyzed by comparing HBFE and HSCA,
4. RP5 is studied by conducting the experiments with kernel variants of HBFE and HBFE, and by comparing the suggested algorithms with alternative feature extraction methods.

Section 5.1 contributes to the analysis of RP2, RP3, RP4 and RP5 by analyzing the experiments with binary classification data sets obtained from chemistry, finance, medicine, physics and other fields. Section 5.2 is devoted to the experiments with multi-label classification data sets, and elucidates certain aspects of RP3. Section 5.3 focuses on the experiments with feature extraction from structured data using string kernel (52), and contributes mostly to RP5. Section 5.4 concludes the chapter.

Table 5.1: Data set statistics.

Data set statistics: m - sample size, D_x - a dimensionality, p^+, p^- - class distribution.

Ames	2495	377	0.3523	0.6477
Australian	690	14	0.55507	0.44493
Breast cancer	683	10	0.34993	0.65007
Covertype	581	54	0.45783	0.54217
Derm	358	34	0.31006	0.68994
German	1000	24	0.7	0.3
Heart	270	13	0.55556	0.44444
Ionosphere	351	34	0.64103	0.35897
Sonar	208	60	0.46635	0.53365
Spambase	4601	57	0.39404	0.60596
Specft	80	44	0.5	0.5
Wdbc	569	30	0.62742	0.37258

5.1 Experiments with Binary Classification Data

We will analyze twelve binary classification data sets summarized in Table **??**. Eleven of them are from the UCI machine learning repository (2), and the *Ames* data set is from chemometrics. Details about this data set is not available due to agreement with the provider.

We denote $Tr = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{m_{train}}$, and $Te = (\mathbf{x}_j, \mathbf{y}_j)_{j=1}^{m_{test}}$ training and testing sets, where $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^{D_x}$ are input variables, and $\mathbf{y}_i \in \{-1, 1\}$ are categorical variables, corresponding to a class label. A rule $f : f(\mathbf{x}) \to y$, which maps the input instance $\mathbf{x}$ to a label $y \in \{-1, 1\}$ is called a classifier. The measure of efficiency we will analyze therein is accuracy over the testing set:

$$\text{Accuracy}(f) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \delta(f(\mathbf{x}_i) = y_i), \tag{5.1}$$

where $\delta(.)$ is an indicator function of the predicate π:

$$\delta(\pi) = \begin{cases} 1 & \textit{if } \pi \textit{ is true}, \\ 0 & \textit{if } \pi \textit{ is not true}. \end{cases} \tag{5.2}$$

All the data was standardized by subtracting the mean and dividing them by the stan-

dard deviation, given that it is non-zero. Components with zero standard deviation were removed. For the sake of simplicity we will use the k nearest neighbor (k-NN) classifier with Euclidean metric

$$d_E(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}')}. \tag{5.3}$$

For the input instance $\mathbf{x}$, the k-NN classifier computes k nearest neighbors from the training set and assigns the class which corresponds to the most frequent class label. The parameter k allows to control smoothness of the decision surface: larger k allows to reduce the effect of noise, but make the boundaries between the classes less distinct: in degenerate case, when k approaches to infinity, the classifier just assigns the most frequent class label. In order to avoid ties, we analyze odd number of neighbors $k \in \{1, 3, 5\}$.

The following feature types were examined:

1. Unmodified inputs (denoted as $Full$) as a baseline (in such a case first d_x features were selected),
2. HBFE features based on the unbiased $HSIC_1$ estimator (denoted as $HBFE_1$),
3. HBFE features based on the biased $HSIC_0$ estimator (denoted as $HBFE_0$),
4. HSCA features based on the unbiased $HSIC_1$ estimator (denoted as $HSCA_1$),
5. HSCA features based on the biased $HSIC_0$ estimator (denoted as $HSCA_0$),
6. PCA features,
7. LDA features.

The choice of PCA and LDA is motivated by their popularity, existence of kernelized versions, and in the case of LDA also a semi-supervised variants (see (12)). Since PCA is an unsupervised technique, it was considered as an indicator how much additional performance we can achieve using a supervised technique. LDA is another well known feature extraction algorithm. The performance of LDA was interesting to us, since it is a supervised technique for classification data sets, and therefore was expected to perform

well in our case. On the other hand, HSIC-based approaches HBFE and HSCA does not exploit categorical nature of the dependent variable, and therefore were expected to perform between PCA and LDA.

The following procedure was adopted when conducting experiments. Fifty random partitions of the data set into training and testing sets of equal size was generated, and feature extraction was performed using all the above-mentioned methods. The projection matrices of the feature extraction methods were estimated using only the training data. The features generated from the testing set then were classified using k-NN classifier. The feature dimensionality was selected using a training data and 3-fold cross validation. Wilcoxon's sign rank test (91) with the standard p-value threshold of 0.05 was applied to the samples of corresponding classification accuracies. The following methods were compared, indicating the statistical significance in the tables:

1. $HBFE_1$ with $HBFE_0$, and $HSCA_1$ with $HSCA_0$ (better one indicated in **bold** text);
2. The most efficient method with the remaining ones (statistically significant cases are reported in underlined text);
3. HSCA with HBFE (data sets where HSCA was more efficient are indicated with $\bullet$, and $\circ$ means that it turned out to be less efficient);
4. The most efficient HSIC-based algorithm (i.e. $HBFE_0$, $HBFE_1$, $HSCA_0$ or $HSCA_1$) with the remaining ones ($\diamond$ means that HSIC-based algorithm outperformed other ones, and $\star$ means that PCA, LDA or unmodified inputs were more efficient).

5.1.1 Linear kernel case

We will start the experiments with the linear versions of HBFE and HSCA. In order to avoid singularities, in the case of HSCA, we will use a constant l_2 regularizer equal to 10^{-5}. The results for the feature dimensionalities optimized by 3-fold cross validation are presented in Table 5.3. Table 5.2 summarizes statistically significant cases presented in Table 5.3. Fig. 5.1, Fig. 5.2 show how the feature dimensionality of HBFE and

Table 5.2: Summary of the experiments with *linear* kernel.

Number of statistically significant cases versus change in accuracy (derived from Table 5.3)

Indicator	$k=1$		$k=3$		$k=5$	
$\bullet$	4	1.4339%	6	2.0554%	6	2.0674%
$\circ$	0	-	0	-	0	-
$\diamond$	3	2.1189%	4	3.1537%	5	2.6018%
$\star$	0	-	0	-	0	-
$HSIC_1 > HSIC_0$	3	2.0487%	0	-	3	0.8523%
$HSIC_1 < HSIC_0$	6	2.1726%	6	2.0896%	6	2.5953%

HSCA influences the accuracy of the classifier (in the case $k = 5$). Additionally, 0.95-confidence intervals are provided.

Table 5.2 shows that HSCA algorithm was more efficient than HBFE ($\bullet$), which contributes to RP4 by demonstrating that feature redundancy minimization improves their quality. The experiments demonstrated 2% improvement in the classification accuracy. However HSCA is more computationally demanding, and for large data sets may be less convenient.

Fig. 5.1, Fig. 5.2, and Table 5.3 contribute to RP3 by revealing, that biasedness of HSIC estimator may be important for HBFE and HSCA. It is quite surprising, since estimators (2.12) and (2.13) estimate the same (2.11) measure, and differ only by $O(m^{-1})$ bias. This difference seems to be important even with quite large training samples. Table 5.2 demonstrates, that biasedness/unbiasedness of the HSIC estimator may change the classification accuracy by $0.8 - 2\%$.

The results of Table 5.3 also contributes to RP5. It demonstrates that either HBFE or HSCA outperformed other methods by 2% - 3% ($\diamond$), or their performance was statistically equal.

Although HBFE and HSCA are general approaches, not optimized for any specific learning problem (e.g. classification or regression), their performance was generally better or similar to that of LDA, which maximizes the class separability. PCA was not as efficient as supervised techniques in most of the cases. Table 5.3 also shows that classification accuracy tends to be higher with larger values of k.

Table 5.3: Classification accuracy using *linear* kernel.

Dataset	*Full*	$HBFE_1$	$HBFE_0$	$HSCA_1$	$HSCA_0$	*PCA*	*LDA*
1-NN classifier							
Ames • ◇	0.7753	0.7589	**0.7765**	0.7826	**0.8012**	0.7786	0.7714
Australian	0.7933	0.7987	0.8045	0.8093	0.8095	0.7868	0.8114
Breastcancer	0.9558	0.9553	0.9543	0.9553	0.9595	0.9566	0.9562
Covertype ◇	0.6868	**0.6956**	0.6732	0.7086	0.7014	0.6748	0.6756
Derm	0.9906	0.9973	0.9973	0.9973	0.9971	0.9949	0.9971
German	0.6698	**0.6841**	0.6730	0.6833	0.6910	0.6700	0.6851
Heart	0.7618	0.7600	0.7677	0.7612	0.7627	0.7520	0.7698
Ionosphere • ◇	0.8421	0.8555	**0.8683**	0.8686	0.8773	0.8581	0.8171
Sonar	0.8146	**0.7819**	0.7538	0.7427	**0.8046**	0.8146	0.6938
Spambase •	0.8975	0.8993	0.8979	0.9116	0.9056	0.9015	0.8680
Specft	0.6770	0.6690	0.7030	0.6730	0.7020	0.6630	0.5370
Wdbc •	0.9503	0.9355	**0.9455**	0.9476	**0.9570**	0.9506	0.9528
3-NN classifier							
Ames • ◇	0.7872	0.7753	**0.7852**	0.7897	**0.8158**	0.7878	0.7853
Australian •	0.8328	0.8329	0.8314	0.8431	0.8423	0.8227	0.8393
Breastcancer	0.9618	0.9642	0.9656	0.9661	0.9674	0.9650	0.9660
Covertype	0.6796	0.6720	**0.6956**	0.7046	0.7063	0.6672	0.7032
Derm	0.9926	0.9973	0.9973	0.9973	0.9973	0.9937	0.9980
German	0.7036	0.7131	0.7078	0.7181	0.7208	0.7031	0.7169
Heart	0.8003	0.7828	0.7959	0.7905	0.7988	0.7876	0.7938
Ionosphere • ◇	0.8313	0.8542	**0.8656**	0.8715	0.8763	0.8615	0.8384
Sonar •	0.7908	0.7450	0.7373	0.7542	**0.7996**	0.7962	0.6923
Spambase • ◇	0.8993	0.9087	0.9051	0.9158	0.9128	0.9065	0.8930
Specft ◇	0.6810	0.7350	0.7550	0.6780	0.7130	0.6780	0.5370
Wdbc •	0.9597	0.9482	0.9528	0.9539	**0.9628**	0.9577	0.9575
5-NN classifier							
Ames • ◇	0.7916	**0.7882**	0.7819	0.7961	**0.8199**	0.7900	0.7904
Australian	0.8445	0.8421	0.8463	0.8499	0.8500	0.8322	0.8529
Breastcancer	0.9641	0.9666	0.9646	0.9668	0.9663	0.9653	0.9665
Covertype	0.6661	0.6843	0.7007	**0.7157**	0.7029	0.6561	0.7113
Derm	0.9911	0.9971	0.9973	0.9971	0.9971	0.9951	0.9984
German •	0.7113	0.7262	0.7209	0.7288	0.7343	0.7127	0.7302
Heart	0.8133	0.8033	0.8068	0.8145	0.8136	0.8062	0.8065
Ionosphere • ◇	0.8279	0.8571	0.8610	0.8622	**0.8763**	0.8539	0.8398
Sonar •	0.7512	0.7108	**0.7369**	0.7488	**0.7812**	0.7631	0.6912
Spambase • ◇	0.8970	0.9057	0.9086	**0.9186**	0.9128	0.9083	0.9003
Specft ◇	0.6680	0.7340	0.7470	0.6800	**0.7240**	0.6860	0.5370
Wdbc • ◇	0.9593	0.9485	0.9528	0.9523	**0.9675**	0.9577	0.9570

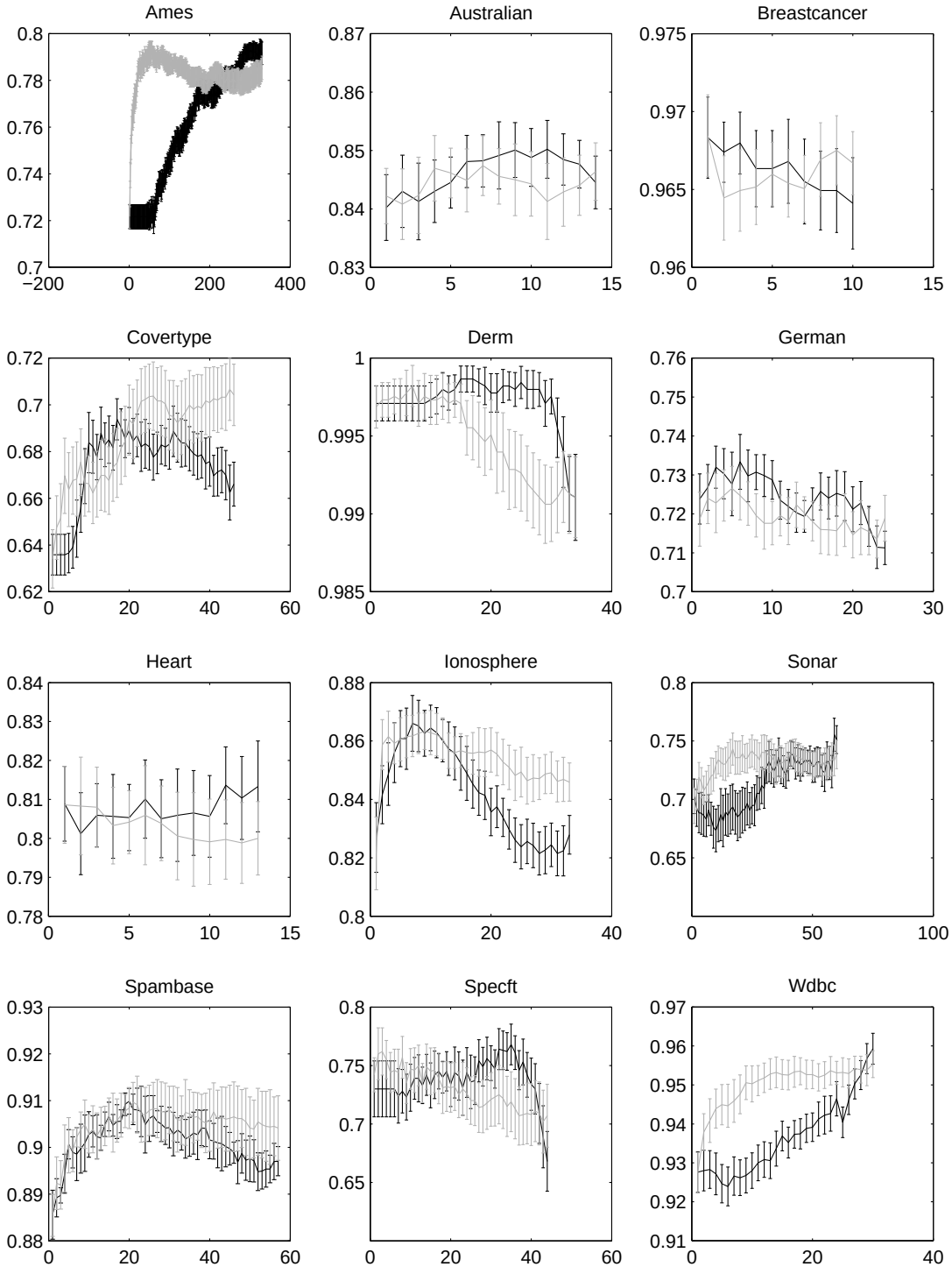

Figure 5.1: Results for UCI data and linear HBFE.

Dimensionality versus accuracy of $HBFE_1$ (black) and $HBFE_0$ (gray). The number of neighbors $k = 5$ in all the cases.

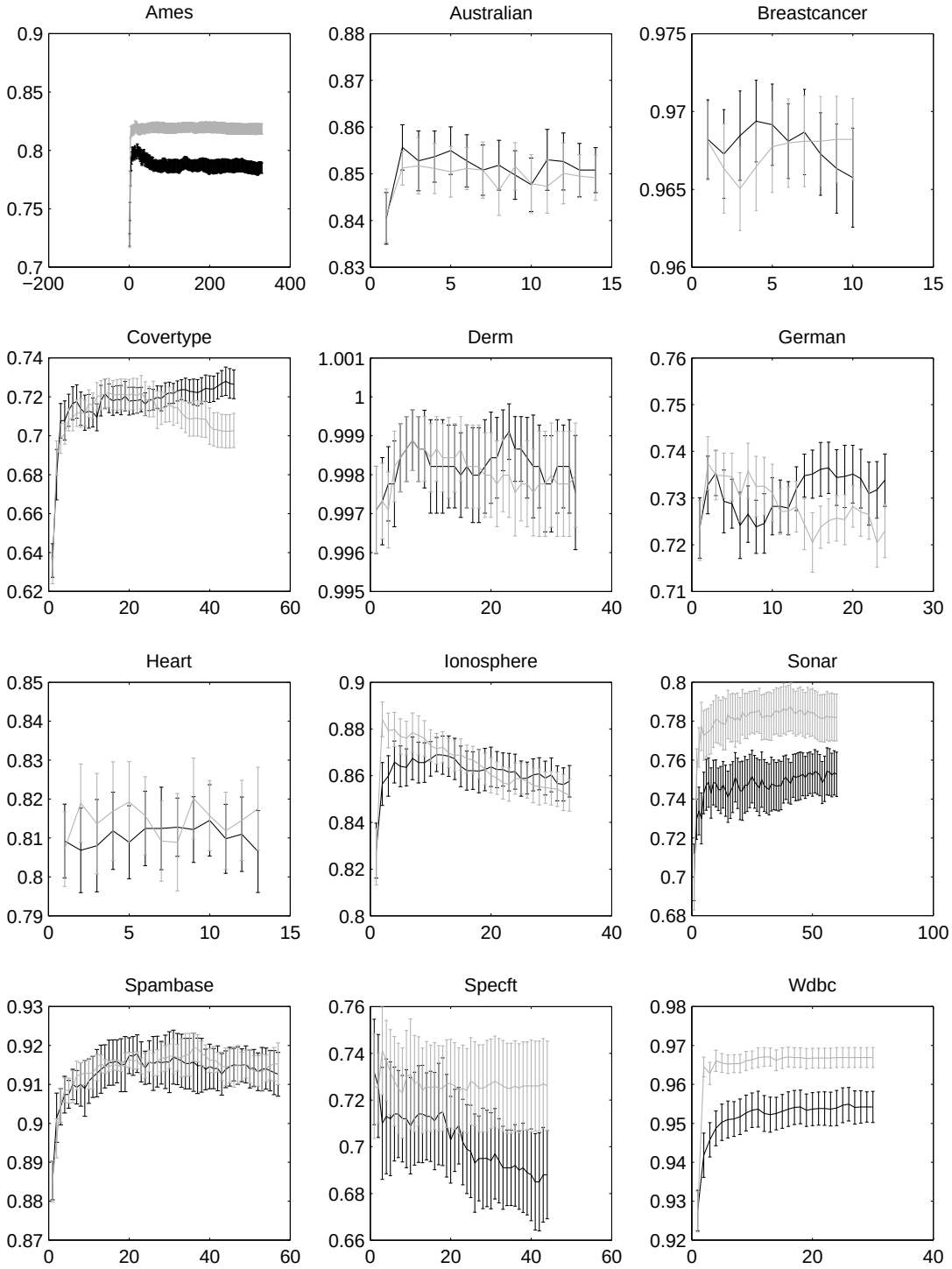

Figure 5.2: Results for UCI data and linear HSCA.

Dimensionality versus accuracy of $HSCA_1$ (black) and $HSCA_0$ (gray). The number of neighbors $k = 5$ in all the cases.

5.1.2 Gaussian kernel case

Further we will look into the non-linear aspects of the data using Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2)$. Therefore, input data are embedded into the RKHS via non-linear feature map, associated with Gaussian kernel. The kernel width parameter $\sigma \in \{(1 + \frac{a}{10}) \cdot median(d_E(\mathbf{x}_i, \mathbf{x}_j)), a \in \{-2, -1, 0, 1, 2\}\}$ was selected by 3-fold cross validation for each feature extraction method individually. The linear kernel was selected for the outputs, since the output data are very simple (only binary class labels). In the case of HSCA, the Gaussian kernel was also used for extracted features. PCA and LDA were also replaced by their kernel variants with Gaussian kernel.

The experimenting scenario with random splits was basically the same as in the linear case. Since in kernel case the solution of kernel algorithm is expressed as the weighted sum of all the training instances, m parameters should be estimated. In order to reduce the number of parameters, and to shorten computations, we select a subset of training instances $\mathcal{I}$ using basis selection algorithm of (6). The stopping criteria of the basis selection process is singularity of input kernel matrix $\mathbf{K}_{\mathcal{I},\mathcal{I}}$. Using full kernel matrix often leaded into numerical instabilities, especially in the case of HSCA. We ommit *Ames* data set due to prohibitively long computations.

The numeric results are presented in Table 5.5 (as in linear case, the dimensionality of the features was optimized using 3-fold cross validation), and statistically significant cases are summarized in Table 5.4. Fig. 5.3 and Fig. 5.4 show how the classification accuracy is influenced by the dimensionality of features, generated by HBFE and HSCA (in the case $k = 5$).

Table 5.4 show that biasedness of HSIC estimator may alter the classification accuracy by up to 5%. Althought the biased estimator appeared to be more efficient in most of the cases, the determination of conditions when one estimator is more efficient than the other requires further investigation.

Table 5.4 contributes to RP4 by demonstrating that the tendency of HSCA to be more efficient than HBFE is preserved (•) with average 2% improvement in classification accuracy.

Table 5.5 and Table 5.4 show that kernel LDA with Gaussian kernel often was more efficient than the other methods. We speculate that it may be influenced by the interac-

Table 5.4: Summary of the experiments with *gaussian* kernel.

Number of statistically significant cases versus change in accuracy (derived from Table 5.5)

Indicator	$k=1$		$k=3$		$k=5$	
$\bullet$	5	2.1779%	5	2.5181%	3	2.9840%
$\circ$	0	-	0	-	0	-
$\diamond$	1	1.5378%	4	2.8429%	2	3.8315%
$\star$	3	3.5171%	3	3.3857%	4	3.3192%
$HSIC_1 > HSIC_0$	1	1.1600%	2	0.7960%	2	0.4770%
$HSIC_1 < HSIC_0$	6	5.3356%	6	3.7074%	4	3.8270%

tion of LDA cost function and non-linear kernel, since it, being more flexible than linear one, allows LDA to transform the data into separable clusters, which are suitable for nearest neighbor classifier. Table 5.5 demonstrates that in certain cases ($\diamond$) either kernel HBFE or kernel HSCA allowed to improve the classification accuracy up to 3.8%.

Table 5.5 contributes to RP5 by showing that Gaussian kernel may improve classification accuracy up to 5% comparing to the linear kernel (e.g. *Ionosphere* and *Heart* data sets). Although we do not compared Gaussian and linear algorithms explicitly, the statistical reliability is reflected by Fig. 5.3, Fig. 5.4 and Fig. 5.1, Fig. 5.2.

However, for most data sets Gaussian kernel was not advantageous. This may be influenced by inefficiently selected σ, or structural properties of the data. As in the case of linear kernel, the classification also tend to be more accurate with larger values of k.

Table 5.5: Classification accuracy using *Gaussian* kernel.

$Dataset$	$Full$	$HBFE_1$	$HBFE_0$	$HSCA_1$	$HSCA_0$	PCA	LDA
1-NN classifier							
Australian •	0.7924	0.7900	0.7927	0.7878	**0.8185**	0.7807	0.8110
Breastcancer	0.9508	0.9505	0.9458	0.9472	0.9466	0.9522	0.9487
Covertype • ⋆	0.6932	0.6480	**0.6738**	0.6855	0.6860	0.6777	0.7091
Derm	0.9872	0.9985	0.9989	0.9993	0.9989	0.9935	0.9984
German • ◇	0.6717	0.6668	0.6784	**0.6956**	0.6797	0.6737	0.6802
Heart	0.7638	0.7689	0.7679	0.7575	0.7669	0.7588	0.7366
Ionosphere	0.8521	0.8895	**0.9128**	0.9025	0.9158	0.9083	0.8927
Sonar •	0.8358	0.6955	**0.7942**	0.7204	**0.8344**	0.8387	0.8258
Spambase • ⋆	0.8570	0.7994	0.7903	0.8119	0.8129	0.8471	0.8779
Specft	0.6778	0.7283	0.7317	0.7650	0.7400	0.6370	0.7370
Wdbc ⋆	0.9510	0.9148	**0.9425**	0.9320	0.9424	0.9510	0.9599
3-NN classifier							
Australian • ◇	0.8362	0.8344	0.8311	0.8233	**0.8491**	0.8193	0.8275
Breastcancer	0.9607	0.9595	0.9548	0.9554	0.9587	0.9617	0.9558
Covertype • ⋆	0.6874	0.6522	0.6731	**0.6989**	0.6752	0.6784	0.7241
Derm	0.9882	0.9993	0.9993	0.9996	0.9996	0.9960	0.9981
German • ◇	0.7006	0.6987	**0.7111**	0.7269	0.7120	0.6913	0.6880
Heart	0.8082	0.8104	0.8257	0.8133	0.8188	0.8012	0.7436
Ionosphere ◇	0.8314	0.8994	**0.9261**	0.9204	0.9230	0.9013	0.8971
Sonar •	0.8100	0.7256	**0.7814**	0.7312	**0.8093**	0.8186	0.8244
Spambase ⋆	0.8630	0.8189	0.8124	0.8276	0.8266	0.8497	0.8883
Specft • ◇	0.6722	0.7450	0.7400	**0.7867**	0.7367	0.6648	0.7370
Wdbc ⋆	0.9596	0.9200	**0.9437**	0.9401	0.9465	0.9560	0.9620
5-NN classifier							
Australian	0.8514	0.8460	0.8375	0.8399	**0.8567**	0.8406	0.8351
Breastcancer	0.9643	0.9632	0.9597	0.9543	0.9586	0.9646	0.9565
Covertype • ⋆	0.6748	0.6703	0.6699	**0.7048**	0.6768	0.6659	0.7315
Derm	0.9850	0.9993	0.9993	0.9996	0.9996	0.9966	0.9981
German ◇	0.7123	0.7159	0.7251	0.7360	0.7285	0.7004	0.6948
Heart	0.8169	0.8321	0.8296	0.8212	0.8281	0.8111	0.7403
Ionosphere	0.8216	0.9078	0.9135	0.9204	0.9223	0.9098	0.9003
Sonar ⋆	0.7878	0.7263	**0.7821**	0.7369	**0.7928**	0.7935	0.8215
Spambase • ⋆	0.8637	0.8223	0.8141	0.8374	0.8355	0.8528	0.8953
Specft • ◇	0.6833	0.7400	0.7500	**0.7900**	0.7400	0.6926	0.7370
Wdbc ⋆	0.9589	0.9242	**0.9488**	0.9380	0.9481	0.9573	0.9682

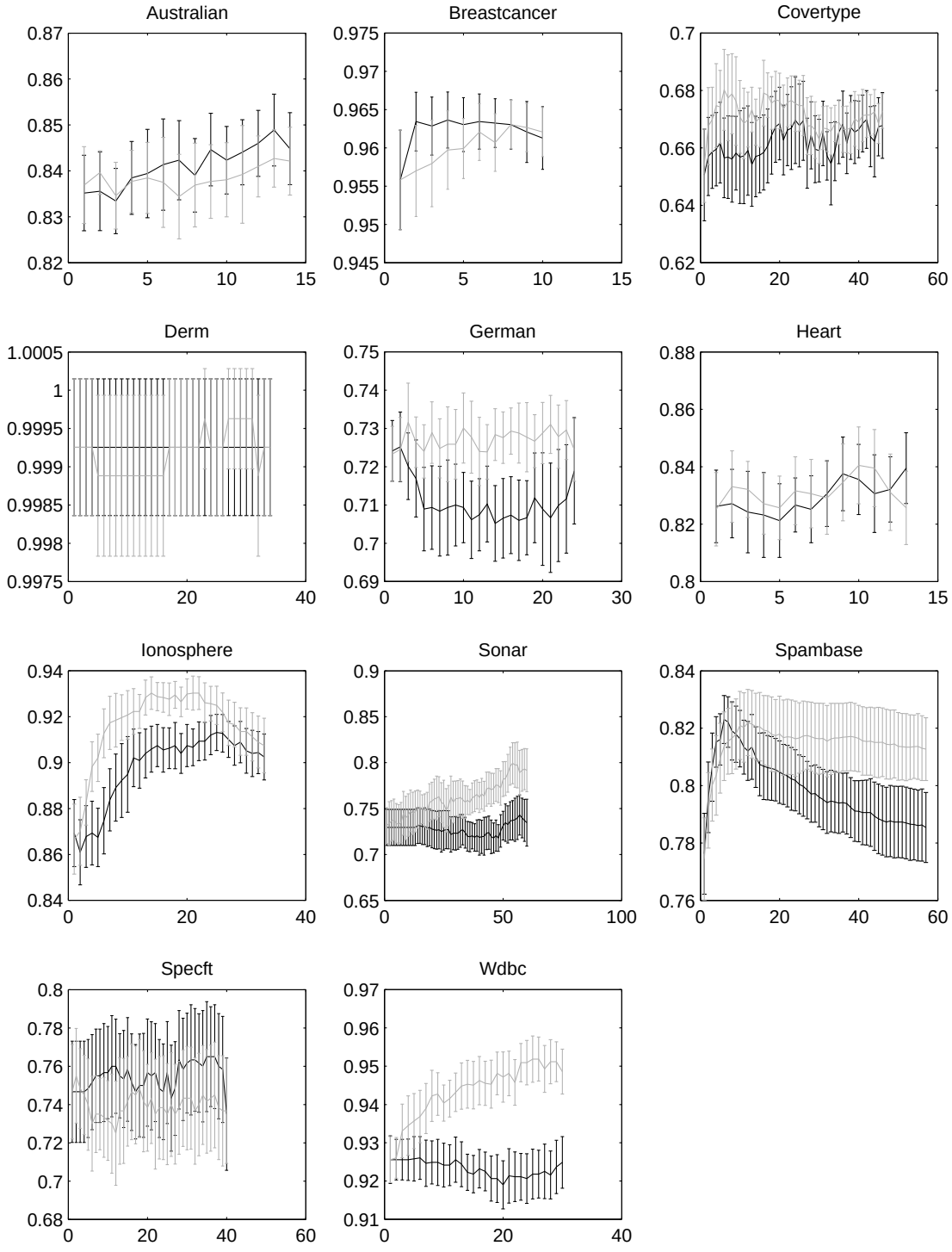

Figure 5.3: Results for UCI data and Gaussian HBFE.

Dimensionality versus accuracy of $HBFE_1$ (black) and $HBFE_0$ (gray). The number of neighbors $k = 5$ in all the cases.

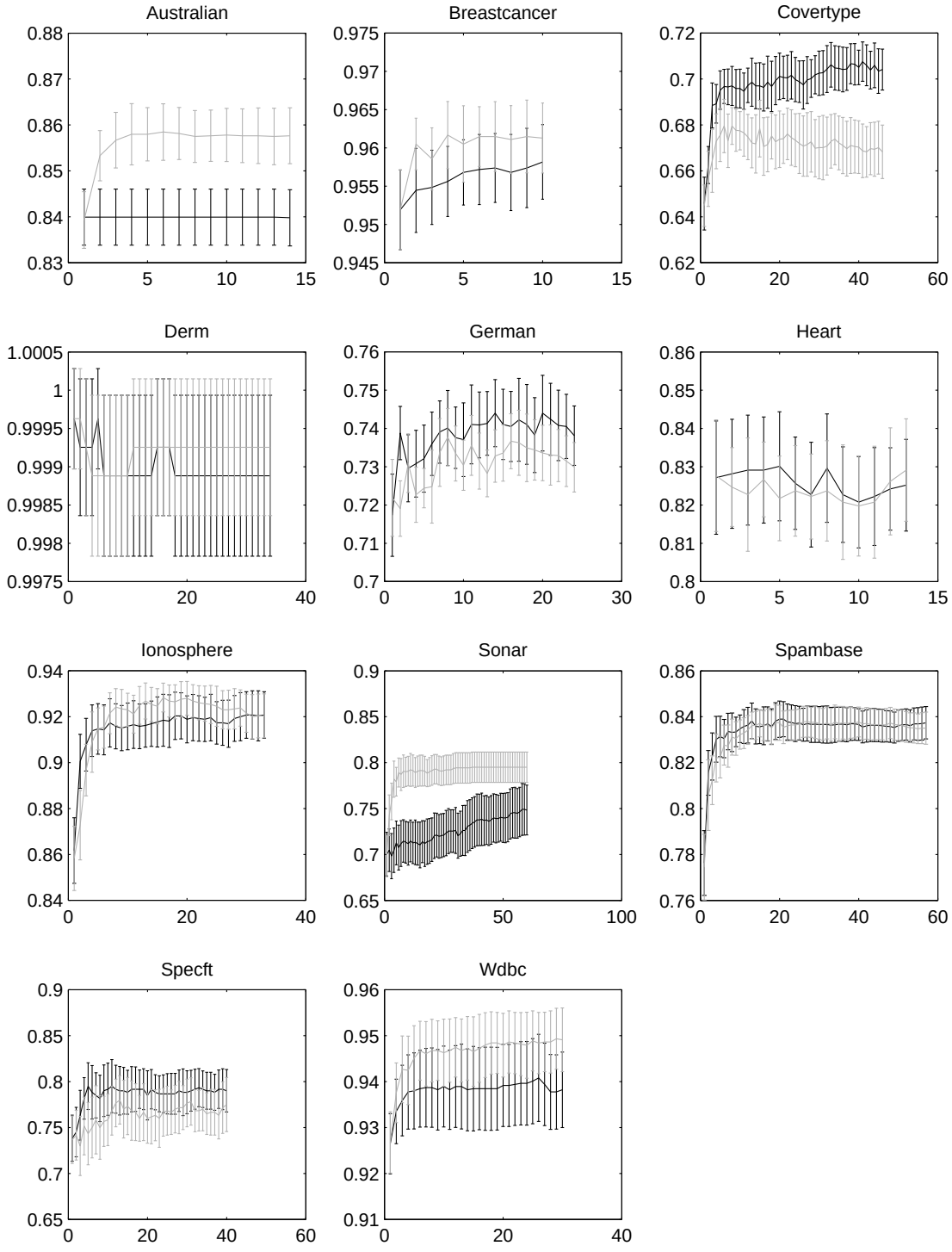

Figure 5.4: Results for UCI data and Gaussian HSCA.

Dimensionality versus accuracy of $HBFE_1$ (black) and $HBFE_0$ (gray). The number of neighbors $k = 5$ in all the cases.

5.1.3 Semi-supervised case

In this section we will conduct empirical investigation of RP2 by analyzing semi-supervised variants of HBFE and HSCA. Following the same scheme as above, now we will assume that in each random split 30 percent of training instances are labeled, and labels for the remaining ones are not known. We will investigate how the classification performance is influenced by graph Laplacian regularizer (2.17) with Gaussian weights

$$\mathbf{W}_{i,j} = exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma^2). \tag{5.4}$$

The parameter of Laplacian regularization $\beta \in [0, 0.05, ..., 0.25, 0.3]$, $\sigma \in \{(1 + \frac{a}{10}) \cdot median(d_E(\mathbf{x}_i, \mathbf{x}_j)), a \in \{-2, -1, 0, 1, 2\}\}$, and feature dimensionality $1 \leq d_x \leq D_x$ were selected by 3-fold cross validation. In this section we will statistically compare analyzed algorithms when $\beta = 0$ (Laplacian regularization was not used), and when the regularization parameter β was selected by cross-validation. In order to avoid numerical instabilities we also used

Kernels for inputs, outputs and features (in the case of HSCA) were linear. The Laplacian regularizer was added to the HBFE, HSCA, and LDA (for semi-supervised LDA see (12)). Since cross validation significantly increased computations, in order to reduce computations we omited *Ames*, and *Spambase* data sets. *Derm* data set was excluded because the classification accuracy already is near to 1. We also ommit PCA since it is purely unsupervised method, and is out of scope of this section.

Numeric results are provided in Table 5.6. Fig. 5.5 (HBFE) and Fig. 5.6 (HSCA) show how the change in classification accuracy is influenced by feature dimensionality. Additionally, for each dimensionality 0.95-confidence intervals are provided.

Table 5.6 demonstrates that in certain cases it is possible to achieve the improvement in the classification accuracy of up to 3.8%, if Laplacian regularizer was used. However, we also observe opposite cases when Laplacial regularization leads into decrease of classification performance. We speculate that at least to some extent this depends on whether hypothesis of Laplacian regularization holds. In general, Fig. 5.5 and Fig. 5.6 show the tendency of semi-supervised modifications of HBFE and HSCA to be efficient.

Table 5.6: Classification accuracy in the case of semi-supervised scenario.

Dataset	*Full*	$HBFE_1$	$HBFE_0$	$HSCA_1$	$HSCA_0$	*LDA*
			1-NN classifier			
Australian	0.779	0.791	0.787	0.803	$0.801^{-1.43}$	0.810
Breastcancer	0.950	0.953	0.951	0.951	0.953	0.950
Covertype	0.629	$0.620^{0.52}$	0.641	0.645	0.646	$0.650^{-3.40}$
German	0.664	0.673	$0.660^{1.03}$	$0.685^{-0.45}$	0.679	0.680
Heart	0.764	0.755	0.759	0.756	0.765	0.749
Ionosphere	0.802	$0.772^{3.83}$	$0.840^{-1.79}$	$0.791^{2.21}$	0.823	0.774
Sonar	0.720	0.656	0.689	0.674	0.709	0.643
Wdbc	0.938	$0.916^{0.18}$	0.929	0.924	0.948	$0.919^{2.01}$
			3-NN classifier			
Australian	0.819	0.831	0.823	0.839	$0.840^{-1.27}$	0.838
Breastcancer	0.958	0.962	0.962	0.961	0.961	0.958
Covertype	0.624	$0.626^{0.48}$	$0.650^{0.18}$	0.657	0.654	$0.668^{-2.81}$
German	0.689	$0.696^{0.31}$	$0.688^{0.99}$	0.713	0.707	0.707
Heart	0.791	0.787	0.791	$0.783^{0.83}$	$0.793^{-1.30}$	$0.758^{3.14}$
Ionosphere	0.768	$0.784^{1.80}$	$0.832^{-1.25}$	$0.801^{2.14}$	0.812	0.774
Sonar	0.674	0.671	0.685	0.682	0.704	0.643
Specft	0.620	0.660	$0.653^{1.61}$	0.636	0.640	$0.639^{-2.50}$
Wdbc	0.941	0.927	0.937	0.930	0.951	$0.922^{1.90}$
			5-NN classifier			
Australian	0.835	0.841	$0.835^{0.81}$	0.848	$0.848^{-0.79}$	0.850
Breastcancer	0.959	0.964	0.964	0.964	0.963	0.962
Covertype	0.621	$0.627^{0.76}$	$0.650^{0.96}$	0.662	0.655	$0.668^{-2.19}$
German	0.705	$0.708^{0.64}$	$0.703^{0.95}$	$0.723^{0.62}$	0.718	0.716
Heart	0.801	0.802	$0.801^{0.24}$	$0.794^{1.06}$	$0.802^{-0.80}$	$0.769^{1.60}$
Ionosphere	0.735	$0.787^{2.14}$	$0.818^{-1.54}$	$0.802^{1.20}$	$0.804^{0.77}$	1.774
Sonar	0.660	$0.677^{0.78}$	$0.681^{0.51}$	0.690	0.706	0.643
Specft	0.589	0.665	$0.642^{2.30}$	$0.614^{1.68}$	0.635	$0.637^{-2.55}$
Wdbc	0.941	0.930	0.936	$0.931^{0.21}$	$0.948^{0.18}$	$0.923^{1.99}$

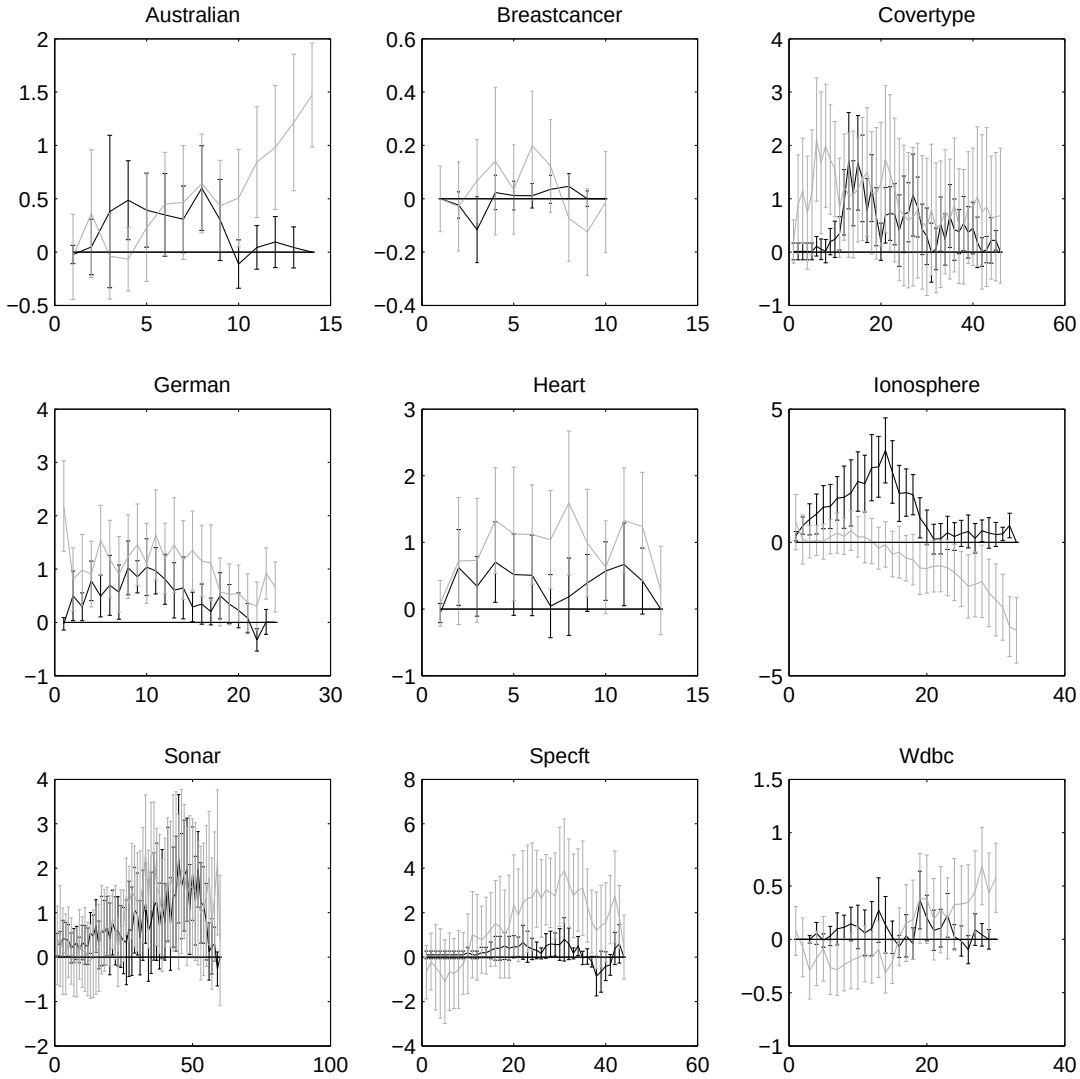

Figure 5.5: Results for UCI data and semi-supervised HBFE.

Dimensionality versus change in accuracy (in percent) for semi-supervised HBFE features based on $HBFE_1$ (black) and $HBFE_0$ (gray). The number of neighbors $k = 5$ in all the cases.

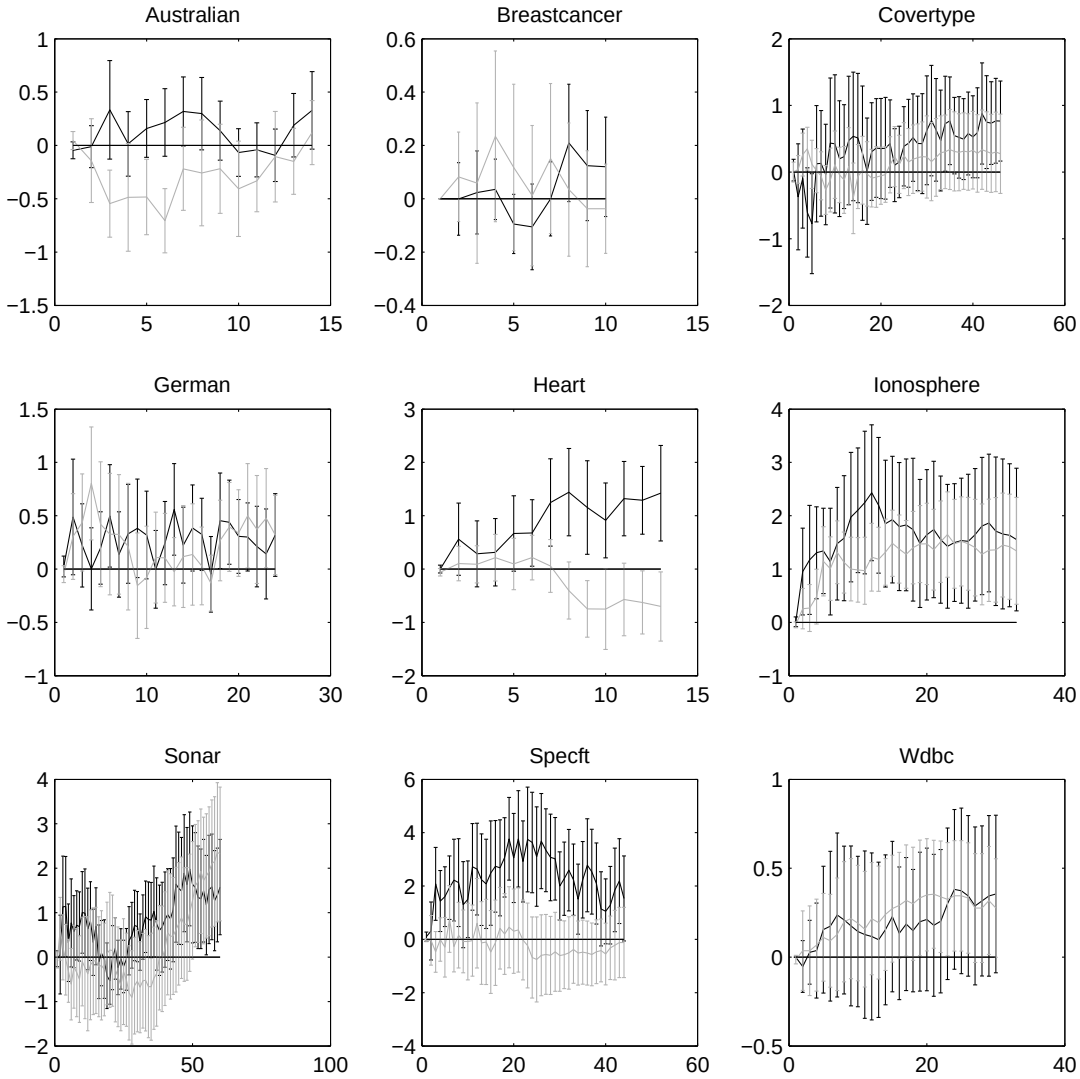

Figure 5.6: Results for UCI data and semi-supervised HSCA.

Dimensionality versus change in accuracy (in percent) for semi-supervised HSCA features based on $HSCA_1$ (black) and $HSCA_0$ (gray). The number of neighbors $k = 5$ in all the cases.

5.2 Experiments with Multi-label Yahoo Data

This section focuses on RP3 and describes the experiments with feature extraction for multi-label classification data. We will start with brief introduction of basic terminology from the topic. A survey on multi-label learning can be found in (81).

5.2.1 Multi-label classification

In multi-label classification problems an object can belong to several classes simultaneously. Such a situations arise in many important areas: web page, text, image classification and other ones. Treating every combination of class labels as a single class and applying standard classification and feature extraction techniques (e.g. LDA) quickly becomes prohibitative due to combinatorial explosion. In contrary, HSIC-based methods does not assume any structure of the dependent variable and therefore are easily applicable in multi-label case.

Let us assume that the input instances are drawn from a set $\mathcal{X} = \mathbb{R}^{D_x}$ and let $\mathcal{Y} = \{1, 2, ..., n_l\}$ be a set of class labels. In multi-label classification problems, the input $\mathbf{x}_i$ is associated with a subset $\mathbf{y}_i \subseteq \mathcal{Y}$.

The goal of multi-label classification is to learn a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, which measures how likely a label y belongs to a set of true labels of $\mathbf{x}_i$, i.e. $f(\mathbf{x}_i, y) > f(\mathbf{x}_i, y')$ where $y \in \mathbf{y}_i$ and $y' \notin \mathbf{y}_i$. Based on such a function the ranking function $\mathrm{rank}_f : \mathcal{X} \times \mathcal{Y} \to \{1, 2, ..., n_l\}$ is defined (98):

$$f(\mathbf{x}_i, y) > f(\mathbf{x}_i, y') \Rightarrow \operatorname*{rank}_f(\mathbf{x}_i, y) < \operatorname*{rank}_f(\mathbf{x}_i, y'). \tag{5.5}$$

The multi-label classifier itself is given by $h(\mathbf{x}_i) = \{y \in \mathcal{Y} : f(\mathbf{x}_i, y) > \theta(\mathbf{x}_i)\}$, where $\theta(.) \in \mathbb{R}$ is a threshold function. In most cases $\theta(.)$ is set to zero constant. In further experiments we use special k-NN classifier for multi-label data (see (98) for details).

5.2.2 Performance measures

Evaluating a performance of multi-label classifier inevitably is harder than in the single label case, because of the overlapping of the class labels: a result can be completely correct, partly correct, or completely wrong, and relevant measure of the performance may depend on the application. Therefore it is useful to take into account several performance measures (100), (53).

Let $T = (\mathbf{x}_i, \mathbf{y}_i)_{i=1}^m$ be a supervised training set, consisting of the input variables $\mathbf{x}_i \in \mathbb{R}^{D_x}$ and corresponding output variables $\mathbf{y}_i$, which describe a set of labels, assigned for the input instance. In the experiments with multi-label classification we use five performance measures (98).

One-error evaluates how often the top-ranked label was not in the set of ground truth labels. Therefore smaller one-error indicates better performance. The measure is defined as

$$\mathrm{One-error}(f) = \frac{1}{m}\sum_{i=1}^{m} \delta(\arg_{y\in\mathcal{Y}} \max f(\mathbf{x}_i, y) \notin \mathbf{y}_i). \tag{5.6}$$

Alternatively one may consider similar measure, which takes into account more best ranked labels.

Hamming Loss takes into account how many times the classifier predicts a label not belonging to an instance, or not predicts one, that belongs to it. Naturally, smaller Hamming loss also is associated with better classification performance. Having a multi-label classifier h, the measure is given by

$$\mathrm{Hamming-loss}(h) = \frac{1}{n_l m}\sum_{i=1}^{m}\sum_{y\in\mathcal{Y}} \delta(y \in h(\mathbf{x}_i) \wedge y \notin y_i) + \delta(y \notin h(\mathbf{x}_i) \wedge y \in y_i). \tag{5.7}$$

Coverage evaluates how far we need, on the average, to go down the list of the labels

in order to cover all the true labels.

$$\text{Coverage}(f) = \frac{1}{m}\sum_{i=1}^{m}\max_{y\in\mathbf{y}_i}\text{rank}_f(\mathbf{x}_i, y) - 1 \tag{5.8}$$

Therefore in ideal case the coverage is zero, and smaller coverage also implies better performance.

Ranking loss evaluates the average fraction of label pairs that are not correctly ordered. Let $\overline{\mathbf{y}}$ be a complementary set of $\mathbf{y}$ in $\mathcal{Y}$. Ranking loss is defined as

$$\text{Ranking}-\text{loss}(f) = \frac{1}{m}\sum_{i=1}^{m}\frac{|(y,y')\in\mathbf{y}_i\times\overline{\mathbf{y}}_i : f(\mathbf{x}_i,y)\leq f(\mathbf{x}_i,y')|}{|\mathbf{y}_i||\overline{\mathbf{y}}_i|}. \tag{5.9}$$

In perfect case ranking loss also equals to zero.

Average precision evaluates the average fraction of labels ranked above a particular label $y\in\mathbf{y}_i$ which actually belongs to $\mathbf{y}$:

$$\text{Average}-\text{precission}(f) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|\mathbf{y}_i|}\sum_{y\in\mathbf{y}_i}\frac{|y'\in\mathbf{y}_i : \text{rank}_f(\mathbf{x}_i,y') < \text{rank}_f(\mathbf{x}_i,y)|}{\text{rank}_f(\mathbf{x}_i,y)} \tag{5.10}$$

In worst case it is equal to 0, and in perfect case reaches 1.

5.2.3 Results for multi-label data sets

We now investigate eleven multi-label classification data sets, originating from web page classification. The input instances are based on statistics, derived from the content of web page, and the output consists of the set of labels, which were assigned to a particular page (e.g. commercial, advertisements, computers etc.). The data sets are available for downloading from[1]. All the data sets consists of 2000 observations. The input and output dimensionalities are given in Table 5.7. Since Yahoo data sets

[1] http://www.kecl.ntt.co.jp/as/members/ueda/yahoo.tar.gz

are quite large in dimensionality and sample size, experiments with non-linear kernels and HSCA were omitted due to quite high computational demand. We tested feature

Table 5.7: Yahoo data set statistics.

Data set	D_x	D_y
Arts	462	26
Business	438	30
Computers	681	33
Education	550	33
Entertainment	640	21
Health	612	32
Recreation	606	22
Reference	793	33
Science	743	40
Social	1047	39
Society	636	27

dimensionalities ranging from 2% to 100% in 5% interval following the same random split scheme as previously. The average results for individual data sets are provided in Table 5.8 and Table 5.9[1]. Fig. 5.7, Fig. 5.8, Fig. 5.9 and Fig. 5.10 reflects the classification performance for different feature dimensionalities. In the figures 0.95-confidence intervals are provided for each dimensionality. In order to find out whether the results are statistically significant Wilcoxon signed rank test was used to compare best performing approach with the remaining ones. The best and statistically significant results (p-value < 0.05) are marked with underlined text.

The results of this section contribute to RP3. They demonstrate that for multi-label classification unbiased $HSIC_1$ estimator is systematically more efficient, than biased one. This tendency is observed in Table 5.8 and Table 5.9, and Fig. 5.7, Fig. 5.8 and Fig. 5.9. Authors of (99) also studied the same data sets and found that HBFE with biased $HSIC_0$ estimator was more efficient than MRSI (96) and LPP (32). Our results also suggest that HBFE with unbiased estimator may be more efficient than two later approaches. In order to compare our results with that of (98), for classification we used the same $k = 10$ nearest neighbors.

[1] $\downarrow$ indicates, that the smaller value is better, and $\uparrow$ that the larger one is better.

Table 5.8: I: averaged results for individual Yahoo data sets.

Dataset	$Full$	$HBFE_1$	$HBFE_0$	PCA
		Average precision ↑		
Arts	0.5187	0.5350	0.5256	0.5175
Business	0.8815	0.8810	0.8807	0.8786
Computers	0.6256	0.6486	0.6418	0.6296
Education	0.5651	0.5700	0.5618	0.5620
Entertainment	0.6044	0.6088	0.6036	0.5981
Health	0.7275	0.7328	0.7314	0.7231
Recreation	0.4786	0.5132	0.5005	0.4902
Reference	0.6175	0.6486	0.6328	0.6208
Science	0.4991	0.5113	0.5022	0.4936
Social	0.7282	0.7258	0.7249	0.7250
Society	0.5865	0.5793	0.5814	0.5819
		Coverage ↓		
Arts	5.5342	5.4120	5.5244	5.5572
Business	2.2580	2.2864	2.2827	2.3072
Computers	4.3115	4.0565	4.1288	4.2584
Education	3.9407	3.9926	4.0168	4.0069
Entertainment	3.3802	3.3353	3.3857	3.3986
Health	2.9814	2.9518	2.9889	3.0028
Recreation	4.9598	4.6434	4.7830	4.8868
Reference	3.3839	3.1393	3.2749	3.3308
Science	6.7934	6.6753	6.8010	6.8859
Social	3.3135	3.6075	3.5178	3.4239
Society	5.6348	5.6643	5.6802	5.6290

Table 5.9: II: averaged results for individual Yahoo data sets.

Data file	$Full$	$HBFE_1$	$HBFE_0$	PCA
		Hamming loss ↓		
Arts	0.0598	0.0591	0.0592	0.0602
Business	0.0266	0.0266	0.0266	0.0267
Computers	0.0413	0.0391	0.0391	0.0405
Education	0.0408	0.0415	0.0411	0.0413
Entertainment	0.0593	0.0578	0.0583	0.0595
Health	0.0399	0.0388	0.0391	0.0401
Recreation	0.0614	0.0597	0.0597	0.0607
Reference	0.0308	0.0295	0.0300	0.0302
Science	0.0354	0.0360	0.0355	0.0356
Social	0.0226	0.0241	0.0230	0.0227
Society	0.0567	0.0588	0.0572	0.0564
		One error ↓		
Arts	0.6004	0.5781	0.5876	0.6026
Business	0.1143	0.1130	0.1131	0.1168
Computers	0.4514	0.4301	0.4354	0.4461
Education	0.5675	0.5644	0.5735	0.5721
Entertainment	0.5184	0.5103	0.5169	0.5275
Health	0.3484	0.3401	0.3370	0.3521
Recreation	0.6717	0.6199	0.6398	0.6556
Reference	0.4876	0.4542	0.4693	0.4848
Science	0.6163	0.5994	0.6105	0.6227
Social	0.3521	0.3569	0.3545	0.3557
Society	0.4602	0.4771	0.4678	0.4656
		Ranking loss ↓		
Arts	0.1527	0.1487	0.1530	0.1532
Business	0.0392	0.0406	0.0402	0.0410
Computers	0.0919	0.0851	0.0873	0.0907
Education	0.0899	0.0911	0.0923	0.0917
Entertainment	0.1237	0.1220	0.1246	0.1245
Health	0.0540	0.0529	0.0538	0.0544
Recreation	0.1865	0.1732	0.1794	0.1832
Reference	0.0892	0.0810	0.0859	0.0878
Science	0.1306	0.1277	0.1303	0.1325
Social	0.0623	0.0679	0.0673	0.0645
Society	0.1407	0.1425	0.1427	0.1423

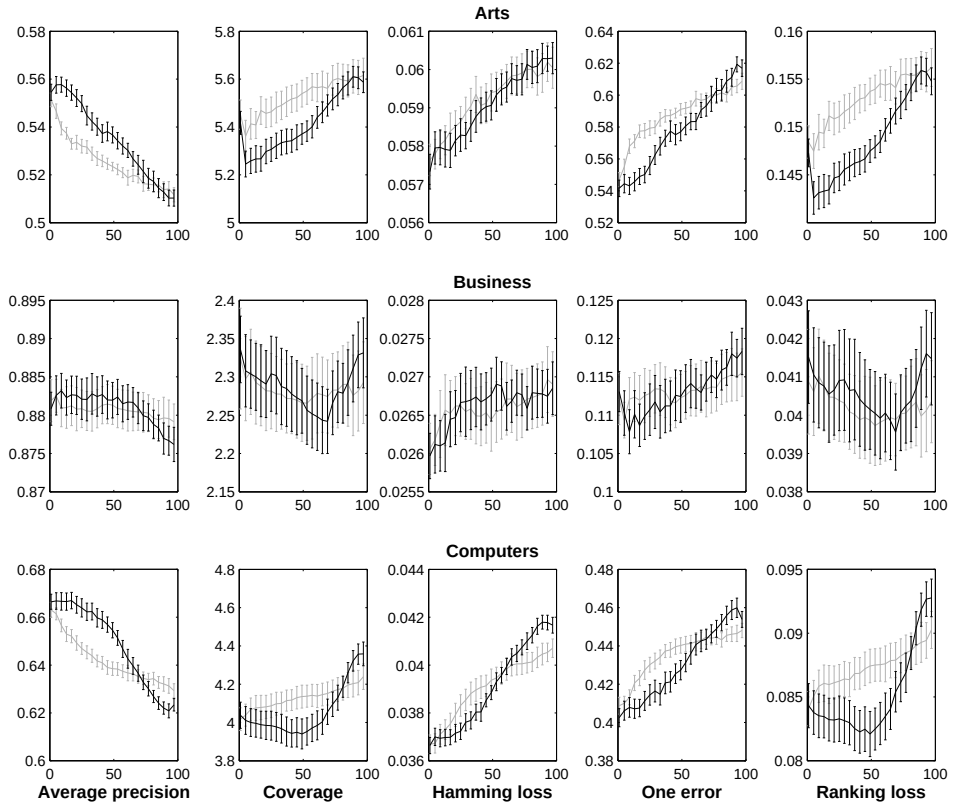

Figure 5.7: Results Yahoo data sets and linear HBFE, part I.

Yahoo data sets, part I. Number of features versus accuracy for the features based on $HBFE_1$ (black) and $HBFE_0$ (gray).

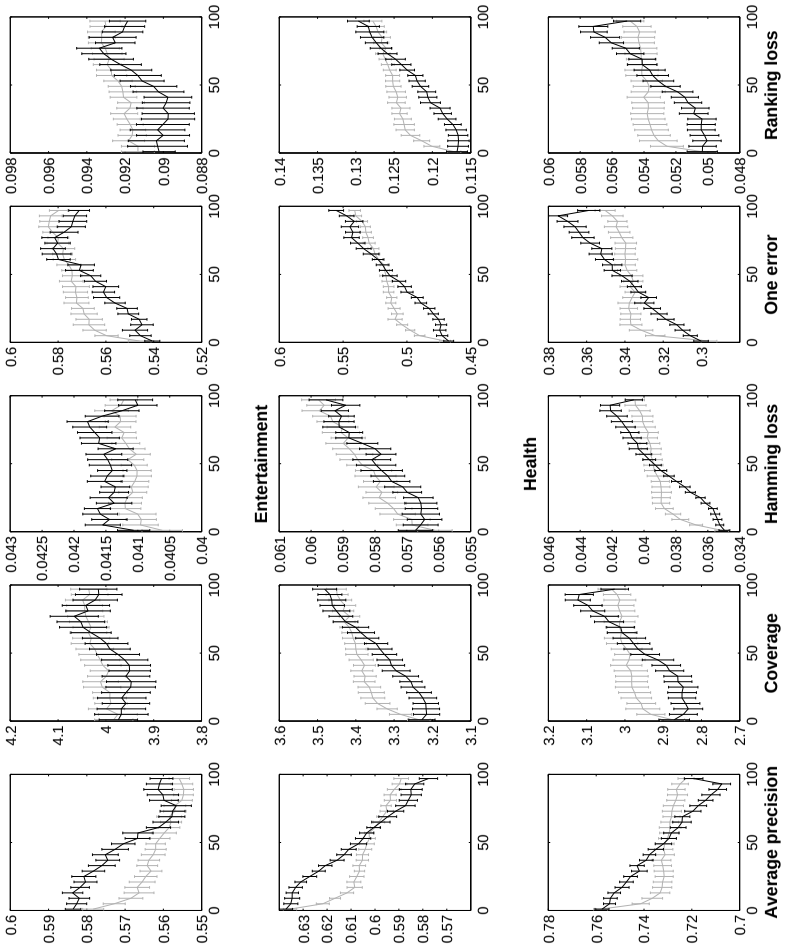

Figure 5.8: Results Yahoo data sets and linear HBFE, part II.

Yahoo data sets, part II. Number of features versus accuracy for the features based on $HBFE_1$ (black) and $HBFE_0$ (gray).

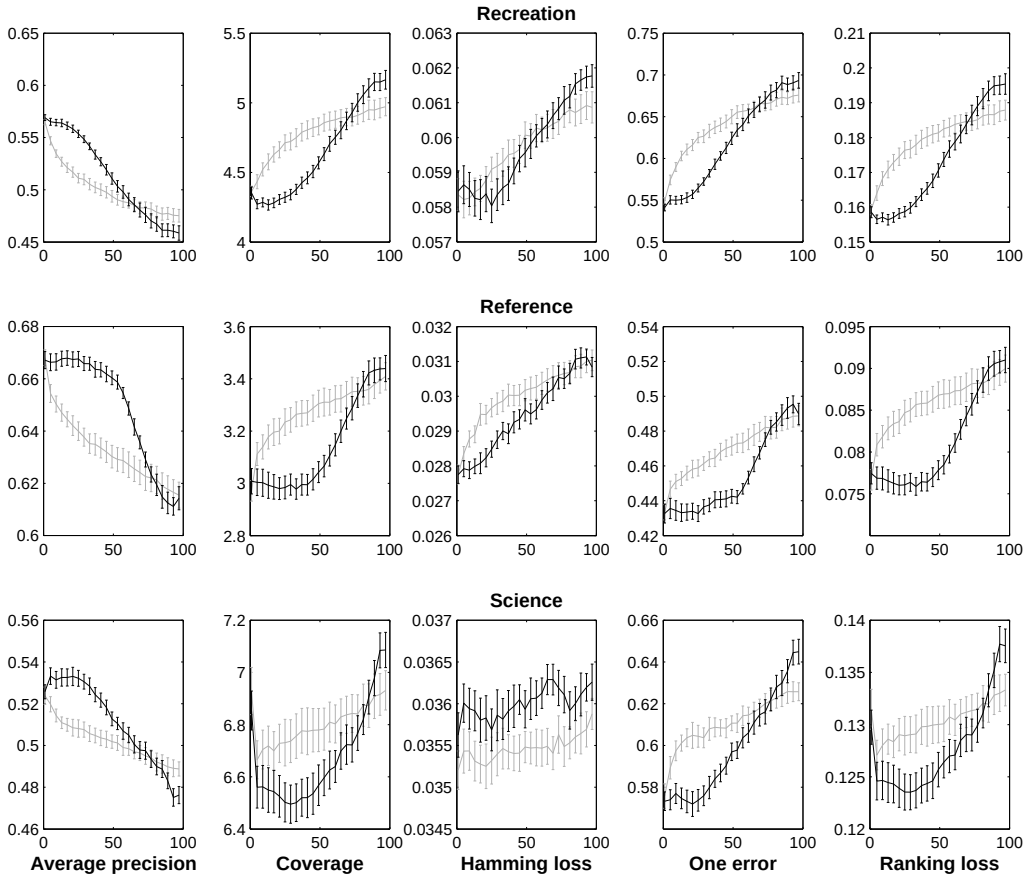

Figure 5.9: Results Yahoo data sets and linear HBFE, part III.

Yahoo data set 7-9, part III. Number of features versus accuracy for the features based on $HBFE_1$ (black) and $HBFE_0$ (gray).

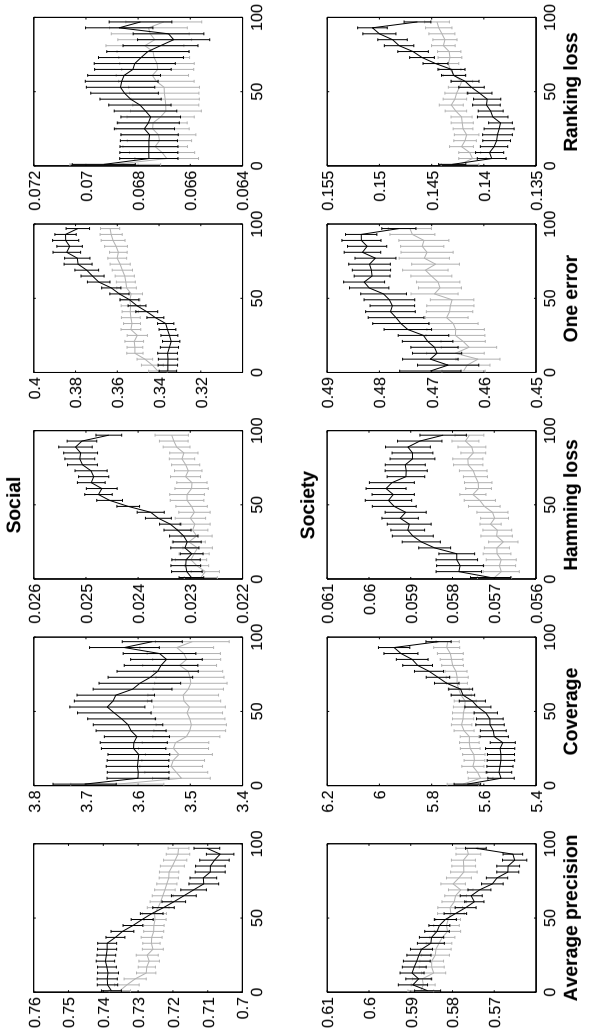

Figure 5.10: Results Yahoo data sets and linear HBFE, part IV.

Yahoo data set, part IV. Number of features versus accuracy for the features based on $HBFE_1$ (black) and $HBFE_0$ (gray).

5.3 Experiments with Structured Data

In this section we will investigate the UCI *Promoters* data set, originating from the field of bioinformatics. It poses a is binary classification problem, where each input instance is represented as string of A,C,G and T, corresponding to four nucleotides and must be classified as a promoter or a non-promoter [1]. The data sets consists of 106 57-dimensional instances, half of which are labeled as promoters and the second one as non-promoters. We investigated numerical (A,G,C,T encoded as 1,2,3,4 and standard kernels applied) and symbolic representation of the data. When dealing with the symbolic representation of the data, we analysed the string kernel (52). This choice was motivated by the intuitive notion that the discriminative information contained in the input should depend more on various combinations of sub strings, rather than just on distances of inner products between them, because such regions have a complex internal structure. Let us start with the definition.

Definition 9 (String kernel)**.** (52). Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. We denote by $|s|$ the length of string $s = s_1 s_2 ... s_{|s|}$. Having two strings s and t by st we denote the string obtained by concatenating the strings s and t. The string $s[i : j]$ is the substring $s_i ... s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_1, ..., i_{|}u|)$, with $1 = i_1 \leq ... \leq i_{|}u| \leq |s|$, such that $u_j = s_{i_j}$, for $j = 1, ..., |u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|}u| - i_1 + 1$. We denote Σ^n the set of all strings of length n, and let $\Sigma^* = \bigcup_{n=1}^{\infty} \Sigma^n$. We now define feature spaces $F_n = \mathbb{R}^{\Sigma^n}$. The feature map for a string s is given by defining u coordinate of $\phi_u(s)$ for each $u \in \Sigma^n$. Fixing some $0 \leq \lambda \leq 1$ we define:

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}. \tag{5.11}$$

These features measure the number of occurrences of subsequences in the strings weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to

[1] Promoter is a region of DNA that facilitates gene transcription.

their frequency of occurrence and lengths. String kernel is expressed by

$$K_{n,\lambda}(s,t) = \sum_{u \in \Sigma_n} \langle \phi_u(s), \phi_u(t) \rangle = \sum_{u \in \Sigma_n} \sum_{\mathbf{i}: u = s[\mathbf{i}]} \sum_{\mathbf{j}: u = t[\mathbf{j}]} \lambda^{(l(\mathbf{i}) + l(\mathbf{j}))}. \tag{5.12}$$

In our experiments we will use a normalized string kernel, which is given by

$$\overline{K}_{n,\lambda}(s,t) = \frac{K_{n,\lambda}(s,t)}{\sqrt{K_{n,\lambda}(s,s) K_{n,\lambda}(t,t)}}. \tag{5.13}$$

Following the same scheme as the one in Section 5.1 for the classification we use a k-nearest neighbor classifier with Euclidean metric ($k \in \{1, 3, 5\}$). The meta-parameters of the kernels were selected by 3-fold cross validation. We consider linear kernel for the outputs and the same one for the previously extracted features (in the case of HSCA).

The classification accuracies are reported in the Table 5.10 (the dimensionality of features was selected by 3-fold cross validation). Fig. 5.11 demonstrates how the accuracy is influenced by the dimensionality of the features.

Experiments with string kernel confirm our hypothesis, since they demonstrate $> 10\%$ improvement in the classification accuracy as compared to the results of the experiments with linear and Gaussian kernels. Table 5.10 shows that the performance of LDA also gradually increased with the complexity of the kernel, finally saturating at the same level as HBFE.

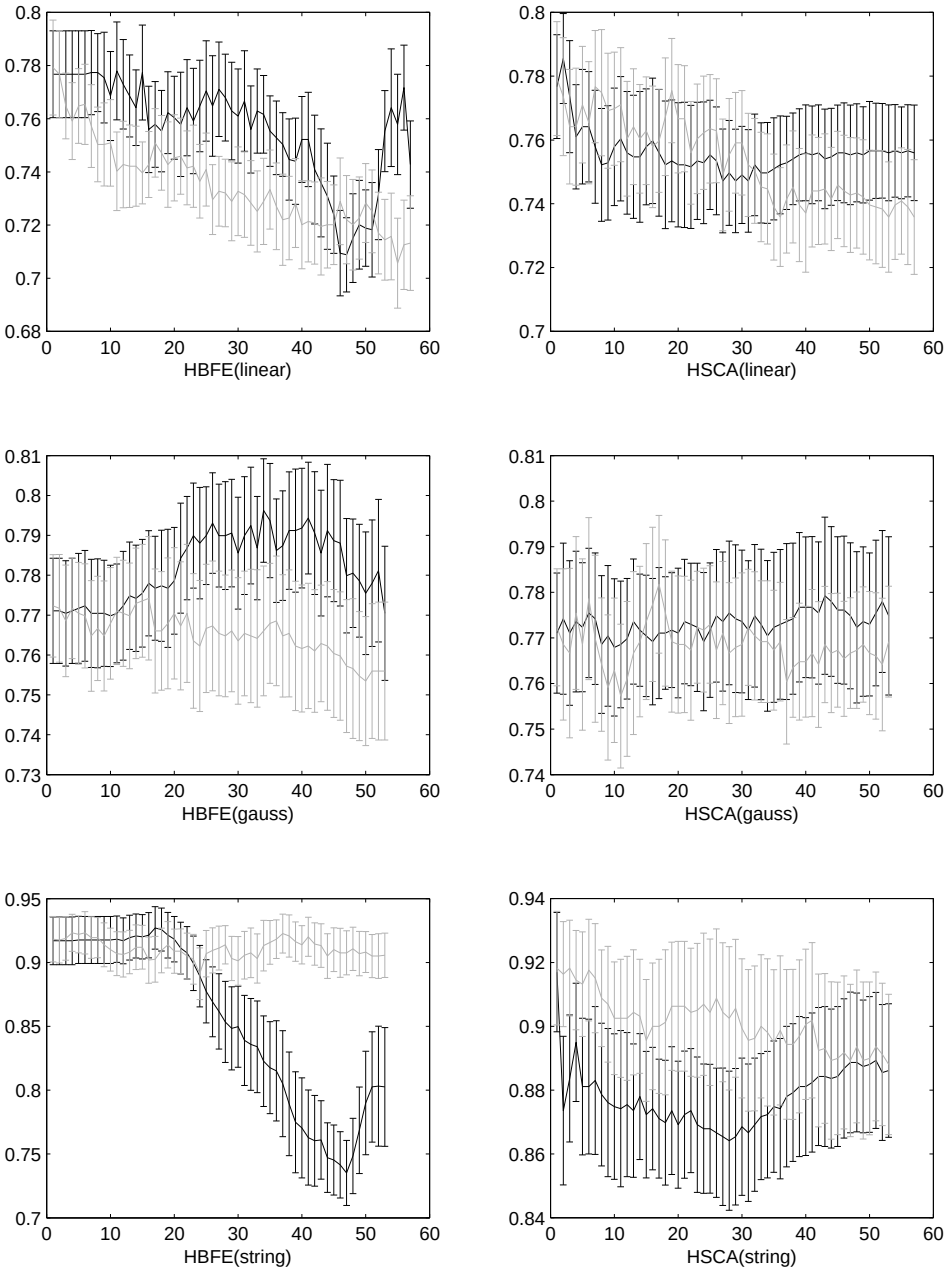

Figure 5.11: Results for Promoters data set.

Number of features versus accuracy for the features of $HBFE_1/HSCA_1$ (black) and $HBFE_0/HSCA_0$ (gray).

Table 5.10: Results for *Promoters* data.

Kernel	Full	$HBFE_1$	$HBFE_0$	$HSCA_1$	$HSCA_0$	PCA	LDA
			1-NN classifier				
Linear		0.7484	0.7553	0.6755	**0.7264**	0.7138	0.6031
Gaussian	0.7264	0.7711	0.7686	0.7774	0.7862	0.7195	0.7736
String		0.9145	0.9126	0.9145	0.9126	0.8226	0.9176
			3-NN classifier				
Linear		0.7635	0.7566	0.7340	**0.7623**	0.7623	0.6031
Gaussian $\diamond$	0.7635	0.7899	0.7792	0.7843	0.7836	0.7371	0.7736
String		0.9075	0.9082	0.9075	0.9082	0.7421	0.9176
			5-NN classifier				
Linear $\diamond$		0.7799	0.7642	0.7491	0.7616	0.7415	0.6031
Gaussian $\diamond$	0.7428	0.7868	0.7818	0.7887	0.7799	0.7365	0.7736
String		0.9170	0.9182	0.9170	0.9182	0.7459	0.9176

5.4 Conclusions

In this chapter we analyzed a scenario, in which extracted features were used as the inputs to a k nearest neighbor classifier. Following the random split scheme (see Section 5.1) we statistically compared considered feature extraction methods, and used the classification performance (accuracy, or other measures) as the measure of feature relevance. The results of the conducted experiments demonstrate practical usefulness of HBFE and HSCA, and contribute to empirical aspects of RP2-RP5.

Chapter 6

Conclusions

In this dissertation we addressed the problem of supervised and semi-supervised feature extraction, when a criteria of feature relevance is based on a dependence structure. Conceptually such an approach is general, since it allows to focus on various dependencies. In Section 1.2 we formulated the problem statement and decomposed it into five research problems RP1-RP5.

RP1. In Chapter 2 we reviewed several dependence measures. We choose to use HSIC as a measure of dependence because of its convenience both from theoretical and practical points of view. It is able to detect any statistical dependence, has simple estimators, and is applicable to any structured data, once appropriate positive definite kernel is known. In Section 4.1, we formalized the concept of dependence structure. Afterwards, we focused on two particular cases and proposed HBFE and HSCA algorithms. HBFE seeks features which maximize the dependence on the dependent variable, while HSCA additionally minimizes the interdependence of features. A kernel versions of HBFE and HSCA allow to analyze non-linear or structured data sets. Additionally, we discussed two feature extraction schemes, NeuroHBFE and NeuroHSCA, which are parallel to HBFE and HSCA. NeuroHBFE and NeuroHSCA use multilayer perceptron neural networks for modeling of the non-linear features.

RP2. In Section 2.3 we reviewed Laplacian regularizer, which penalizes the cost function if for two nearby data instances the corresponding features are distant. In Sec-

tion 4.4 we used Laplacian regularization for construction of semi-supervised variants of HBFE and HSCA. In Section 5.1.3 we investigated semi-supervised modifications of analyzed feature extraction algorithms. The results from Section 5.1.3 show that the suggested algorithms may be helpful for semi-supervised learning data sets. The experiments demonstrate slight improvement (up to 3%, but in most cases only a fraction of percent) in the classification accuracy as compared to the baseline algorithms without Laplacian regularization. On the other hand, such an improvement may be significant in certain situations, when the observation of dependent variable is expensive.

RP3. RP3 was analyzed using both theoretical and experimental research methodology. In HBFE and HSCA, the dependence is measured either by using the biased or unbiased estimator of HSIC 2.2.4. In Chapter 4 we derived the variants of HBFE and HSCA for both estimators. Empirical analysis of RP3 was conducted in Section 5.1, Section 5.2, and Section 5.3. We observed that biasedness of the estimator may strongly affect the classification accuracy (up to 5%). The experiments with binary classification data sets show that biased estimator of HSIC tends to be more efficient than the unbiased one. However, the experiments with multi-label classification (Section 5.2) reveal that in the case of HBFE the tendency is opposite. However, the determination of conditions for one estimator to be more efficient than the another, requires further investigation.

RP4. In order to answer to RP4, we compared HBFE and HSCE experimentally, since they are based on two different dependence structures. In all the cases where HBFE and HSCA was compared, HSCA either was more efficient, or their performances was statistically identical. In the former case the experiments demonstrate the average $\approx 2\%$ improvement in the classification accuracy.

RP5. Suggested HBFE and HSCA algorithms were compared to PCA, LDA and unmodified features as a baseline. Experiments show that in certain data sets $2 - 3\%$ improvement in the classification accuracy is possible as comparing to the alternative algorithms. On the other hand, we feel that further experimentations with more different algorithms, and especially data sets, associated to various machine learning problems, are needed to support our empirical results.

Experimental results from Section 5.1 show that linear kernel often is sufficient. However, for certain data sets where non-linearities or structural properties were important, other kernels turned out to be more efficient. The experiments demonstrated that Gaussian or string kernel allowed to achieve $5 - 10\%$ improvement in the classification accuracy as compared to linear algorithms. The experiments with different kernels show that suggested feature extraction schemes are efficient with non-linear or structural data sets.

Bibliography

[1] Aronszajn, N., 1950. Theory of reproducing kernels. Transactions of the American Mathematical Society, Vol. 68, pp. 337-404. 10

[2] Asuncion, A. and Newman, D.J., 2007. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/]. University of California, School of Information and Computer Science. 47

[3] Basili, R., Cammisa, M., and Moschitti, A., 2006. A semantic kernel to classify texts with very few training examples. Informatica, Vol. 30(2), pp. 163-172. 9

[4] Battiti, R., 1994. Using mutual information for selecting features in supervised neural net learning. Neural Networks, Vol. 5(4), pp. 537-550. 22, 35

[5] Bach, F.R., Jordan, M. Kernel., 2002. Independent component analysis. Journal of Machine Learnin Research. Vol. 3, pp. 1-48. 24

[6] Baudat, G., and Anouar, F., 2003. Feature vector selection and projection using kernels. Neurocomputing, Vol. 55, pp. 21-38. 33, 54

[7] Bloehdorn, S., Basili, R., Cammisa, M., and A. Moschitti., 2006. Semantic kernels for text classification based on topological measures of feature similarity. In Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 06), pp. 18-22. 9

[8] Bollacker, K.D., and Ghosh, J., 1996. Linear feature extractors based on mutual information. In Proc. 13th ICPR, pp. 720-724. 22, 35

[9] Borgwardt, K.M., 2007. Graph kernels. Doctoral dissertation. München. 9

[10] Belkin, M. and Niyogi, P., 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. Advances in Neural Information Processing System, pp. 585-591. 22, 28

[11] Cai, D., He, X., Zhou, K., Han, J., and Bao, H., 2007. Locality Sensitive Discriminant Analysis. Proc. 20th International Joint Conference on Artificial Intelligence (IJCAI), pp. 708-713. 22

[12] Cai, D., 2009. Spectral regression: a regression framework for efficient regularized subspace learning. Doctoral dissertation. Urbana, Illinois. 18, 26, 48, 59

[13] Y. Chen, Y. Li, X. Cheng, and L. Guo., 2006. Survey and taxonomy of feature selection algorithms in intrusion detection system. Inscrypt, pp. 153-167. 3

[14] Cover, TM, Thomas JA. Elements of Information Theory. New York: John Wiley and Sons. 12

[15] Daniušis, P., Vaitkus, Pr., 2008. Neural network with matrix inputs. Informatica, Vol. 19(4), pp. 477-486. 6, 20

[16] Daniušis, P., Vaitkus, Pr., 2008. Kernel regression on matrix data. Proceedings of Lith. mathematical society, Vol. 48-49, pp. 191-195. 6

[17] Daniušis, P., Vaitkus, Pr., 2009. Supervised feature extraction using Hilbert-Schmidt norms. Lecture Notes In Computer Science, Vol. 5788, pp. 25-33. 6, 22, 34, 35, 36, 37

[18] Daniušis, P., Vaitkus, Pr., 2009. A feature extraction algorithm based on the Hilbert-Schmidt independence criterion. Šiauliai Mathematical Seminar, Vol. 4(12), pp. 35-42. 6, 22, 34, 35, 44, 45

[19] Daniušis, P., Vaitkus, Pr., 2009. Matrix-based neural network with linear nodes. Electronics and Electrical engineering, Vol. 6(94), pp. 39-42. 6

[20] Daniušis P., Janzing D., Mooij J., Zscheischler J., Steudel B., Zhang K., and Schölkopf B., 2010. Inferring deterministic causal relations. Proceedings of UAI2010. 6

[21] S. Das, Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection, 2001. Proc. 18th International Conferenfe of Machine Learning, pp. 74-81. 22

[22] Dhanjal, C., 2008. Sparse kernel feature extraction. Doctoral dissertation. 23, 24, 27, 33

[23] Ding C, Peng H. Minimum redundancy feature selection from microarray gene expression data., 2003. Proceedings of the IEEE Conference on Computational Systems Bioinformatics. pp. 523-528. 39

[24] Drineas, P., and Mahoney, M.W., 2005. Approximating a Gram matrix for improved kernel-based learning, in Proceedings of the 18th Annual Conference on Learning Theory, pp. 323-337. 33

[25] Gärtner, T., 2003. A survey of kernels for structured data. SIGKDD Explorations, Vol. 5(1), pp. 49-58. 9

[26] Golub, G., and Van Loan, C., 1996. Matrix computations. The Johns Hopkins University Press, Baltimore, third edition. 23, 33

[27] Grassberger, P. and Procaccia, I. Measuring the strangeness of strange attractors. Physica D, Vol. 9, pp. 189-208. 32

[28] Gretton, A., Bousquet, O., Smola, A., Schölkopf B., 2005. Measuring statistical dependence with Hilbert-Schmidt norms. Proceedings of 16th International Conference on Algorithmic Learning Theory, pp. 63-77. 7, 14, 15, 16, 35

[29] Gretton, A., Fukumizu, K., Sriperumbudu K., B., 2009. Discussion of: Brownian distance covariance. Annals of Statistics. Vol. 3, No. 4, pp. 1285-1294. 13

[30] Guyon, I., Gunn, S., Nikravesh, M., and Zadeh, L., 2006. Feature Extraction, Foundations and Applications. Springer. 22, 30, 39

[31] Haykin, S., 1998. Neural Networks: A Comprehensive Foundation (2nd Edition) *Prentice Hall*. 20, 31

[32] He, X., and Niyogi, P., 2004. Locality preserving projections. In Advances in Neural Information Processing Systems, Cambridge, MA. 66

[33] Hein, M., Bousquet, O., 2004. Kernels, associated structures and generalizations. Technical report. 9, 15

[34] Hinton, G. E., and Salakhutdino R. R., 2006. Reducing the Dimensionality of Data with Neural Network. Science. Vol. 313 (5786), pp. 504-507. 27, 28

[35] Huang, G.B., Liang, N.Y., Rong H.J., Saratchandran, P. and Sundararajan N., 2005. On line sequential extreme learning machine. In The IASTED International Conference on Computational Intelligence. Calgary, Canada. 20

[36] Friedman, J.H., 1997. On bias, variance, 0/1-loss, and the curse-of-dimensionality. Data Mining and Knowledge Discovery, Vol. 1(1), pp. 55-77. 21

[37] Fisher, R.A., 1936. The use of multiple measurements in taxonomic problems. Annals of Eugenics, Vol. 7, pp. 179-188. 25

[38] Fukumizu, K., Bach F.R., Jordan, M.I., 2004. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. Journal of Machine Learning Research, Vol. 5, pp. 73-99. 15

[39] Fukumizu, K., Gretton, A., Sun, X., and Schölkopf B. Kernel measures of conditional dependence. NIPS 20, pp. 489-496. 17

[40] Fukunaga, K., 1990. Introduction to Statistical Pattern Recognition. Academic Press, 2nd edition. 25

[41] Hyväarinen, A., Karhunen, J., and Oja, E., 2001. Independent Component Analysis. New York: John Wiley and Sons. 24

[42] Janzing, D., Mooij, J., Zhang, K., Lemeire, J., Zscheischler J., Daniušis, P., Steudel, B., and Schölkopf B., 2012. Information-geometric approach to inferring causal directions. Artificial Intelligence, Elsevier. 6

[43] Jolliffe, I. T., 1986. Principal component analysis. Springer, Berlin. 22

[44] Kennedy, J., Eberhart, R. Particle swarm optimization, 1995. IEEE International Conference on neural networks. Vol. 4. pp. 1942-1948. 20

[45] Kirkpatrick, S., Gelatt, C. D., and Vecchi M. P. Optimization by Simulated Annealing., 1983. *Science*. Vol. 220(4598), pp. 671-680. 45

[46] Klami, A. Modelling of mutual dependencies, 2008. Doctoral dissertation. 12

[47] Kong, X., Yu, P.S., 2010. Multi-Label Feature Selection for Graph Classification. IEEE International Conference on Data Mining, pp. 274-283. 35

[48] Kriegel, H.P., Kröger, P., Schubert, E., and Zimek, A., 2008. A General Framework for Increasing the Robustness of PCA-based Correlation Clustering Algorithms. Proceedings of 20th Int. Conf. on Scientific and Statistical Database Management, Vol. 5069, pp. 418-435. 24

[49] Lanckriet, G.R.G., Cristianini, N., Bartlet, P., El Ghaoui, L., Jordan, M.I., 2004. Learning the Kernel Matrix with Semidefinite Programming. Journal of Machine Learning Research, Vol. 5, pp. 27-72. 32

[50] Lemme, A., Reinhart, R.F., and Steil J.J., 2010. Efficient online learning of a non-negative sparse autoencoder. ESANN 2010 proceedings, pp. 1-6. 28

[51] Levina, E., and Bickel, P.J., 2004. Maximum likelihood estimation of intrinsic dimension. NIPS 17, pp. 777-784. 32

[52] Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins C., 2002. Text classification using string kernels. Journal of Machine Learning Research, Vol. 2, pp. 419-444. 9, 46, 73

[53] Li, T., Zhang, C., and Zhu, S., 2006. Empirical studies on multi-label classification. In Proceedings of the 18th IEEE international conference on tools with artificial intelligence, pp. 86-92. 64

[54] Lowe, D.G., 1999. Object Recognition from Local Scale-Invariant Features. Proceedings of the International Conference on Computer Vision, Vol. 2, pp. 1150-1157. 2, 3

[55] Maaten, L.J.P., Postma, E.O., and Herik, H.J., 2009. Dimensionality Reduction: A Comparative Review. Preprint. 22

[56] Mackey, L. 2008. Defation Methods for Sparse PCA. In NIPS 21, pp. 1017-1024.

[57] Masaeli, M., Fung G., Dy J. G., 2010. From Transformation-Based Dimensionality Reduction to Feature Selection. International Conference on Machine Learning (ICML), Haifa, Israel. 35

[58] Minsky, M. and Papert, S., 1969. Perceptrons: An Introduction to Computational Geometry. MIT press. Cambridge, US. 18

[59] Niu, D., Dy, J. G, Jordan, M.I., 2011. Dimensionality Reduction for Spectral Clustering. 14th International Conference on Articial Intelligence and Statistics (AISTATS). 35

[60] M. Paliwal and U. A. Kumar., 2009. Neural networks and statistical techniques: A review of applications. Expert Syst. Appl., Vol. 36(1), pp. 2-17. 18

[61] Pinkus A., 1999. Approximation theory of the MLP model in neural networks. *Acta Numerica*, Vol. 8, pp. 143-195. 19

[62] Rajasekaran, S., Vijayalakshmi, P., 2003. Neural networks, fuzzy logic and genetic algorithms. New Delhi: Prentice Hall of India. 20

[63] Raudys, S., 2001. Statistical and Neural Classifiers - An Integrated Approach to Design. Springer, London. 18

[64] Ridder, D., Kouropteva, O., Okun, O., Pietikäinen, M., Duin, R.P.W., 2003. Supervised locally linear embedding. Proceedings of the 2003 joint international conference on Artificial neural networks and neural information processing, pp. 333-341. 29, 30

[65] Rosenblatt, Frank., 1957. The Perceptron - a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory. 18

[66] Rossi, F., Conan-Guez, B., 2002. Multi-layer perceptron on interval data. Classification, Clustering and Data Analysis (IFCS 2002), pp. 427-434. 20

[67] Rudin, W., 1986. Real and Complex Analysis, Third Edition. McGraw-Hill Science/Engineering/Math. 10

[68] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986. Learning representations by back-propagating errors. Nature, Vol. 323, pp. 533-536. 18, 20

[69] Roweis, S.T., and Saul, L. K., 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science, Vol. 290, pp. 2323-2326. 22, 29

[70] Saeys, Y., Inza, I., Larranaga, P., 2007. A review of feature selection techniques in bioinformatics, Bioinformatics, Vol. 23(19), pp. 2507-2517. 2

[71] Shao, J., Wang, Y., Deng, X., and Wang, S., 2011. Sparse linear discriminant analysis by thresholding for high dimensional data. The Annals of Statistics, Vol. 39(2), pp. 1241-1265. 27

[72] Schölkopf, B., Herbrich, R., and Smola, A.J., 2001. A generalized representer theorem. COLT/EuroCOLT. LNAI 2111, pp. 416-426. 11

[73] Schölkopf B. and Smola A.J., 2002. Learning with Kernels. MIT Press. 2, 8, 24

[74] Smola, A. J. and Scholkopf, B., 2000. Sparse greedy matrix approximation for machine learning. In Proc. Intl. Conf. Machine Learning, pp. 911-918. 33

[75] Signoretto, M., Lathauwerb, L., Suykens, J.A.K., 2011. Kernel-based Framework to Tensorial Data Analysis. Neural networks, Vol. 24(8), pp. 861-874. 9

[76] Socher, R., Pennington, J., and Huang, E. H., and Ng, A. Y., Manning, C.D., 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. Proceedings of Conference on Empirical Methods in Natural Language Processing. pp. 151-161. 28

[77] Song, L., Smola, A., Gretton, A., Borgwardt, K., and Bedo, J., 2007. Supervised feature selection via dependence estimation. In Proc. Intl. Conf. Machine Learning, pp. 823-830. 14, 15, 16, 30, 32, 35

[78] Song, L., Smola, A., Gretton, A., Borgwardt, K., 2007. A Dependence Maximization View of Clustering. ICML 24, pp. 815-822. 35

[79] Song, Y., Nie, F., Zhang, C., 2008. Semi-supervised sub-manifold discriminant analysis, Pattern recognition letters, Vol. 29(13), pp. 1806-1813. 18, 27, 29

[80] Song, L., Smola, A., Gretton, A., Borgwardt, K., 2008. Colored Maximum Variance Unfolding. NIPS 20, pp.1385-1392. 22, 35

[81] Sorower, M.S., 2010. A Literature Survey on Algorithms for Multi-label Learning. Preprint. 63

[82] Steinwart, I., 2002. On the influence of the kernel on the consistency of svms. Journal of Machine Learning Research, Vol. 2, pp. 67-93. 10, 15

[83] Suzuki T., Sugiyama, M., 2010. Sufficient Dimension Reduction via Squared-loss Mutual Information Estimation. Technical Report TR09-0005, Department of Computer Science, Tokyo Institute of Technology. 35

[84] Szekely, G. J., Rizzo, M.L., and Bakirov, N. K., 2007. Measuring and testing independence by correlation of distances. Annals of Statistics. Vol. 35, pp. 2769-2794. 13

[85] Tipping, M. E., and Bishop, C., 1999. Probabilistic Principal Component Analysis. Journal of the Royal Statistical Society, Vol. 61, pp. 611-622. 24

[86] Tishby, N., Pereira, F.C., and Bialek, W. The Information Bottleneck method. The 37th annual Allerton Conference on Communication, Control, and Computing, pp. 368Ű377. 35

[87] Torkkola, K., 2003. Feature Extraction by Non-Parametric Mutual Information Maximization. Journal of Machine Learning Research, Vol. 3, pp. 1415-1438. 22, 35

[88] Torre, F., and Black, M.J., 2001. Robust principal component analysis for computer vision. In ICCV, pp. 362-369. 24

[89] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol P.A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, Vol. 11, pp. 3371-3408. 28

[90] Wang, M. Sha, F., and Jordan, Michael I., 2010. Unsupervised kernel dimension reduction. NIPS. Vancouver, Canada. 35

[91] Wilcoxon, F. 1945. Individual comparisons by ranking methods. Biometrics, Vol. 1, pp. 80-83. 49

[92] Williams, C. K. I., and Seeger, M., 2001. Using the Nystrom method to speed up kernel machines. In Advances in Neural Information Processing Systems 13, pp. 682-688. 33

[93] Xu, J.W., Bakardjian H., Cichocki A., and Principe J.C., 2008. A new nonlinear similarity measure for multichannel signals. Neural Netowrks, Vol. 21, pp. 222-231. 14

[94] Xu, J. W., Paiva A. R. C., Park I., and Principe J. C., 2008. A reproducing kernel Hilbert space framework for informationtheoretic learning. IEEE Transactions on Signal Processing, Vol. 56(12), pp. 5891-5902. 14

[95] Yu, K., Tresp, V., and Zhou, D., 2004. Semi-supervised Induction with Basis Function. Technical Report No. 141, Max Planck Institute for Biological Cybernetics. Tüubingen, Germany. 17, 18

[96] Yu, K.; Yu, S., and Tresp, V., 2005. Multi-label informed latent semantic indexing. In SIGIR, pp. 258-265. 66

[97] Yu, L., Liu, H., 2004. Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research. Vol. 5, pp. 1205-1224. 39

[98] Zhang, M.L., and Zhou, Z.H., 2007. ML-kNN: A lazy learning approach to multi-label learning. Pattern Recognition, Vol. 40(7), pp. 2038-2048. 63, 64, 66

[99] Zhang, Y., Zhou, Zhi-Hua., 2008. Multi-label dimensionality reduction via dependence maximization. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, pp. 1503-1505. 22, 35, 36, 37, 66

[100] Zhang, Y., Zhou, Zhi-Hua., 2010. Multi-Label Dimensionality Reduction via Dependence Maximization. ACM Transactions on Knowledge Discovery from Data (TKDD), Vol. 4(3), pp. 1-21. 64