University of Cantabria

Department of Communications Engineering

Kernel Methods for Nonlinear Identification, Equalization and Separation of Signals

Author: Steven Van Vaerenbergh

Supervisor: Ignacio Santamaría Caballero

2009

Métodos Kernel para Identificación, Igualación y Separación no Lineal de Señales

Tesis que se presenta para optar al título de

Doctor por la Universidad de Cantabria

Autor: Steven Van Vaerenbergh

Director: Ignacio Santamaría Caballero

Programa Oficial de Posgrado en Tecnologías de la Información
y Comunicaciones en Sistemas de Telecomunicación

Grupo de Tratamiento Avanzado de Señal

Departamento de Ingeniería de Comunicaciones

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

Universidad de Cantabria

2009

Nur droben, wo die Sterne,
Gibt’s Kirschen ohne Kerne.

—*Heinrich Heine*[1]

[1] Quoted after [Schölkopf and Smola, 2002].

Affiliations

Advanced Signal Processing Group
Department of Communications Engineering
University of Cantabria, Spain

This work was supported by FPU grant AP2005-5366 of the Spanish Ministry of Education and Science (MEC).

Acknowledgements

This thesis is the result of five years of brewing ideas and a few months of actually cooking them up into a single document. It would not have been possible without the support and assistance of numerous friends and colleagues.

First and foremost I wish to thank my supervisor, Dr. Ignacio Santamaría, for his excellent guidance and valuable advice that helped me focus on the core issues in this research. His talent to explain things clearly helped me to understand complex problems in an easy way. Second, I would like to thank my colleagues at the GTAS lab; Dr. Javier Vía for the fruitful discussions about almost all problems in signal processing (and about how they can all be solved by CCA), David Ramírez who was always in the mood for a pleasant chat, my former supervisor Dr. Luis Vielva for helping me with my first steps as a Ph.D. student, my office colleagues Victor, Jesus and Fouad for keeping the working spirit up, and all current and previous lab members I have had the pleasure to work with.

I am also very grateful to Dr. José C. Príncipe of the Computational NeuroEngineering Laboratory at the University of Florida for inviting me to be a visiting scholar at his lab, and for the insights he shared with me. I remember very inspiring and perspective-broadening discussions with many of the CNEL members about a multitude of ideas in machine learning, especially with Il "Memming" Park, Sohan Seth, and Luis Sánchez Giraldo.

Furthermore, I was fortunate enough to collaborate with Dr. Umut Ozertem, Dr. Weifeng Liu, Dr. Paolo Emilio Barbano and Dr. Deniz Erdogmus on various machine learning techniques treated in this thesis. I am grateful for the knowledge they shared with me during numerous interesting discussions.

I would also like to thank my family and especially my parents for giving me support, and my friends in Belgium and Spain, in particular David, Luisín and Mon, for taking me out to try new challenges whenever the lab was closed.

Finally, and most importantly, I thank Ángela, for always encouraging me and for inspiring me every day to be a better person. I am looking forward to write new chapters with her.

Steven Van Vaerenbergh

Santander, 29 September 2009

Abstract

In the last decade, kernel methods have become established techniques to perform nonlinear signal processing. Thanks to their foundation in the solid mathematical framework of reproducing kernel Hilbert spaces (RKHS), kernel methods yield convex optimization problems. In addition, they are universal nonlinear approximators and require only moderate computational complexity. These properties make them an attractive alternative to traditional nonlinear techniques such as Volterra series, polynomial filters and neural networks. Kernel methods also exhibit certain drawbacks that must be addressed properly in every application, including complexity issues for large data sets and overfitting problems.

In this work we propose a set of kernel-based algorithms to solve a number of related, nonlinear problems in signal processing and communications. In particular, we deal with the identification and equalization of nonlinear systems, and with nonlinear blind source separation (BSS).

First, the identification of nonlinear systems is addressed. After discussing supervised kernel-based techniques for identifying black-box nonlinear systems, we focus on the family of online kernel algorithms, which are usually posed as adaptive filtering algorithms in the kernel feature space. Nevertheless, most online kernel methods show difficulties that are not encountered in classical adaptive filtering and have not been satisfactorily solved yet. Specifically, they require a growing memory and are unable to track time-varying nonlinear systems. As a first contribution we present a set of kernel recursive least-squares (KRLS) algorithms that deal with both problems by fixing the memory size and adjusting the stored data adequately. These algorithms are also used as building blocks in later chapters.

In order to limit the complexity of the nonlinear mapping we then study the block-based nonlinear Wiener and Hammerstein systems. Despite their limited modeling capability, these systems are sufficient to represent many nonlinearities that appear in practice. By applying a suitable restriction in the chosen identification diagram, we show how a kernel canonical correlation analysis (KCCA) solution emerges. Additionally, by including the previously proposed KRLS techniques in this framework, we obtain a set of adaptive KCCA algorithms suitable for online identification and equalization of Wiener and Hammerstein systems.

After a further analysis of block-based systems, we demonstrate how oversampling allows blind identification and blind equalization of Wiener systems by applying a KCCA-based technique. The proposed technique is inspired by a linear blind identification method which we extend into feature space, and it can be applied to any scenario where multiple Wiener systems are excited by the same input signal.

In the second part of this thesis we treat blind source separation problems that allow for clustering approaches. This is the case when the source signals either belong to a finite alphabet or show a high degree of sparseness. However, when the mixture process is time-varying or nonlinear, the algorithms based on classical clustering do not hold. We study two such problems and we show that they can be tackled by designing specific versions of spectral clustering, which has an interpretation as kernel principal component analysis.

The first scenario is found in the blind decoding problem of fast time-varying multiple-input multiple-output (MIMO) systems. While the scatter plot data form overlapping clusters here, which prohibits the application of conventional clustering algorithms, we observe that they can be untangled by including temporal information. For the resulting problem we present a spectral clustering algorithm whose kernel is designed to favor the expected cluster shape and to exploit the constellation geometry. The second scenario is a nonlinear blind source separation problem in which the sources are sparse. We deal with the very restrictive underdetermined case, in which the number of available mixtures is less than the number of sources, and we develop a clustering algorithm capable of identifying the nonlinear mixture process, which allows to recover the original source signals.

In summary, this dissertation presents several techniques for related applications in nonlinear signal processing and communications. The proposed methods contribute to the state of the art in nonlinear system identification and equalization, and in nonlinear blind source separation.

Resumen

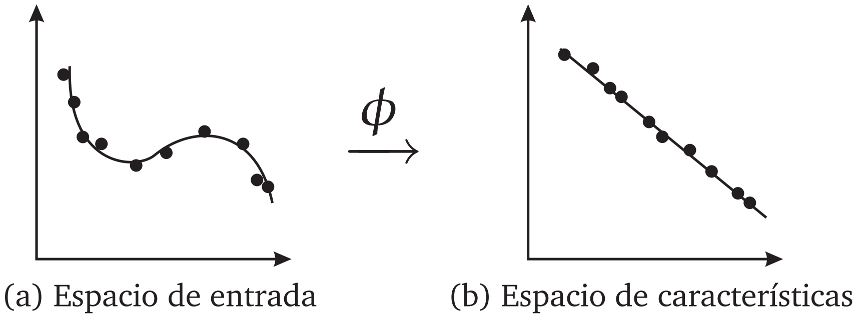
En la última década, los métodos kernel (métodos núcleo) han demostrado ser técnicas muy eficaces en la resolución de problemas no lineales. Parte de su éxito puede atribuirse a su sólida base matemática dentro de los espacios de Hilbert generados por funciones kernel ("reproducing kernel Hilbert spaces", RKHS); y al hecho de que resultan en problemas convexos de optimización. Además, son aproximadores no lineales universales y la complejidad computacional que requieren es moderada. Gracias a estas características, los métodos kernel constituyen una alternativa atractiva a las técnicas tradicionales no lineales, como las series de Volterra, los filtros de polinómicos y las redes neuronales. Los métodos kernel también presentan ciertos inconvenientes que deben ser abordados adecuadamente en las distintas aplicaciones, por ejemplo, las dificultades asociadas al manejo de grandes conjuntos de datos y los problemas de sobreajuste ocasionados al trabajar en espacios de dimensionalidad infinita.

En este trabajo se desarrolla un conjunto de algoritmos basados en métodos kernel para resolver una serie de problemas no lineales, dentro del ámbito del procesado de señal y las comunicaciones. En particular, se tratan problemas de identificación e igualación de sistemas no lineales, y problemas de separación ciega de fuentes no lineal ("blind source separation", BSS). La motivación de este trabajo se basa en la observación de que los métodos kernel son capaces de resolver problemas no lineales de manera eficiente y precisa con un gasto computational razonable, lo cual posibilita su implementación en tiempo real.

Esta tesis se divide en tres partes. La primera parte consiste en un estudio de la literatura sobre los métodos kernel y sus aplicaciones al procesado de señal. Las contribuciones de esta tesis al estado de arte se detallan en la segunda y tercera partes. Puesto que los problemas tratados están muy relacionados, se dividen las contribuciones basándose en las técnicas aplicadas. Específicamente, en la parte II se aplican técnicas de regresión con kernels y en la parte III se desarrollan métodos basados en el agrupamiento espectral.

I. Métodos Kernel y el Espacio de Características

El análisis de la estructura de un conjunto de datos es un problema central en el aprendizaje máquina. Por ejemplo, en problemas de regresión se dispone de una serie de datos de entrenamiento que representan las entradas y las correspondientes salidas de un sistema lineal o no lineal. El objetivo de la regresión es descubrir la relación funcional entre la entrada y la salida de este sistema, para poder así predecir



(a) Espacio de entrada (b) Espacio de características

Figura 1: La idea básica de los métodos kernel. El mapeo ϕ transforma los puntos de entrada (puntos negros) a un *espacio de características* de alta dimensión, donde corresponden a un modelo aproximadamente lineal (linea recta continua). Este modelo en el espacio de características corresponde a un modelo no lineal en el espacio de entrada (linea curvada continua).

la salida del sistema cuando se le presenta un dato de entrada nuevo. Una idea clave aquí es que las salidas correspondientes a entradas similares también deben ser similares. Para ello, se hace uso de la noción de *kernels* (núcleos). Un kernel representa una medida de similitud entre dos datos de entrada. Es una función simétrica que tiene como entrada dos datos (que pueden ser vectoriales) y produce un número real que mide la similitud entre estos datos. Estos métodos no se limitan sólo a los datos numéricos, dado que se pueden aplicar siempre y cuando la similitud entre los diferentes objetos se pueda medir como un escalar.

Los métodos kernel se basan en el marco matemático de los espacios de Hilbert generados por kernels (RKHS). Como consecuencia práctica, permiten calcular los productos escalares en un espacio de dimensión alta, posiblemente infinita, llamado *espacio de características* ("feature space"), como una función kernel en el espacio de entrada. Esta propiedad sencilla y elegante, conocida como "kernel trick", permite transformar cualquier algoritmo basado en productos escalares a un espacio de dimensión mucho mayor mediante la sustitución de los productos escalares en funciones kernel. La ventaja de moverse a un espacio de dimensión mucho mayor supone que, cuando el conjunto de datos observados muestra una cierta estructura no lineal, es más probable que los datos transformados a este espacio de característica correspondan a un modelo lineal (ver Fig. 1) gracias a la alta dimensionalidad de este espacio. Además, aunque la dimensionalidad de este espacio impide resolver el problema transformado de manera explícita, el kernel trick permite calcular la solución de manera implícita, es decir, en función de los datos disponibles.

Recientemente se han propuesto un gran número de técnicas basadas en métodos kernel para resolver problemas en clasificación, regresión no lineal, agrupamiento ("clustering") y predicción de series temporales no lineales. Estas técnicas incluyen las máquinas de vectores de soporte ("support vector machines", SVM), los procesos Gaussianos ("Gaussian processes", GP), el análisis discriminante mediante kernels ("kernel discriminant analysis", KDA), la regresión kernel regularizada ("kernel ridge regression", KRR), el análisis de componentes principales mediante kernels ("kernel principal component analysis", KPCA), el análisis de correlaciones canónicas mediante kernels ("kernel canonical component analysis", KCCA) y métodos de agrupamiento espectral ("spectral clustering"). Las técnicas elaboradas en esta tesis están relacionadas sobre todo con regresión kernel, KPCA, KCCA y agrupamiento espectral.

Mientras que los métodos kernel permiten obtener resultados sorprendentes en problemas no lineales usando sólo operaciones algebraicas sencillas, también conllevan algunos inconvenientes. En primer lugar, puesto que los métodos kernel son aproximadores universales, corren el peligro de sobreajustarse a un conjunto de datos de entrenamiento. La capacidad de generalización se puede mejorar entre otro mediante la técnica de regularización. En segundo lugar, el mapeo no lineal basado en kernels se construye en general como una expansión ponderada de kernels de un conjunto de vectores de soporte. Gracias al Teorema de Representación ("Representer Theorem") estos vectores de soporte se pueden elegir como los datos de entrenamiento. Sin embargo, cuando el conjunto de entrenamiento es grande, esta representación puede ser problemática, dado que muchos métodos kernel tienen complejidades cuadráticas o cúbicas en el número de datos usado. Como solución se suele recurrir a técnicas que reducen el número de vectores soporte, las cuales permiten una representación dispersa de la solución. Otros problemas de interés a la hora de aplicar métodos kernel son la elección de una función kernel adecuada (que debe representar la información a priori sobre el problema), y la determinación de los parámetros de esta función kernel.

II. Identificación e Igualación de Sistemas no Lineales

En la primera de las líneas de trabajo dentro de esta tesis se han desarrollado métodos kernel para la identificación supervisada y ciega de sistemas no lineales. En entornos en que se conocen todos los datos del problema de antemano, se pueden aplicar las técnicas de regresión en bloque ("batch") basadas en kernels que se estudian en el capítulo 2. Las contribuciones de esta parte de la tesis se centran por un lado en el desarrollo de nuevos algoritmos kenel online, y por otro lado en su aplicación a la identificación de sistemas de Wiener y Hammerstein, en entornos batch y online. Finalmente, también se trata el problema de la identificación ciega de estos sistemas.

Identificación Supervisada y Adaptativa de Sistemas no Lineales

En el capítulo 3 se estudia el estado del arte de las técnicas adaptativas basadas en kernels que permiten mejorar o actualizar su solución a medida que se van recibiendo más datos. Estos métodos adaptativos se obtienen en base a las técnicas lineales de filtrado adaptativo a las cuales se aplica el kernel trick.

El mayor problema al desarrollar métodos kernel online es el tamaño del soporte de la solución, que crece linealmente con el número de datos procesados. Puesto que el Teorema de Representación dicta que la solución exacta de los problemas tratados

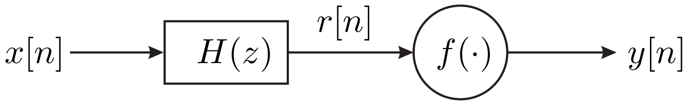


Figura 2: Diagrama de bloques de un sistema de Wiener. $H(z)$ representa el filtro lineal, y $f(\cdot)$ representa la no linealidad estática.

se puede expresar como una expansión kernel en todos los puntos del conjunto de datos de entrenamiento, un método kernel online requiere una memoria y una complejidad computacional que crecen en cada iteración. Por esta razón se suele limitar el crecimiento del soporte, añadiendo sólo los datos más significativos.

En la literatura se han sugerido algunas estrategias para construir una memoria dispersa de vectores soporte en métodos online, incluyendo el método de asignación de recursos en redes de funciones de base radial ("resource allocating networks", RAN) [Platt, 1991] y el criterio de dependencia lineal aproximada ("approximate linear dependency", ALD) [Engel et al., 2004]. En el capítulo 4 se presenta un enfoque distinto, que añade un dato en cada iteración a la memoria, pero también descarta un dato, para mantener así el tamaño de la memoria fijo. La primera técnica propuesta basada en esta filosofía es una versión kernel del algoritmo recursivo de mínimos cuadrados ("recursive least-squares", RLS) que limita la memoria sólo a los datos más recientes mediante una ventana deslizante ("sliding window"). Este método, llamado "sliding-window kernel recursive least-squares" (SW-KRLS) es capaz de ajustar su solución satisfactoriamente en entornos dinámicos gracias a que olvida los datos más antiguos. Por otra parte, este método no tiene en cuenta la importancia de cada dato descartado. Por lo tanto, cada vez que se desprende de un dato importante el rendimiento de este método empeora bruscamente. El segundo método propuesto parte de la misma idea de una memoria de tamaño fijo, pero en vez de descartar en cada iteración el dato más antiguo, es capaz de determinar cuál de los datos en su memoria es el menos importante para la regresión, y a continuación eliminarlo. Este método, denominado "fixed-budget kernel recursive least-squares" (FB-KRLS) obtiene un mejor rendimiento para identificar sistemas no lineales tanto estacionarios como variantes en el tiempo.

En conclusión, las técnicas estudiadas y propuestas en los capítulos 3 y 4 son sistemas de aprendizaje que pueden "sorprenderse" al ver datos nuevos que les aportan información, y pueden "olvidar" datos antiguos o menos relevantes. Gracias al marco de los métodos kernel estas técnicas presentan una complejidad moderada y su solución se obtiene mediante problemas de optimización convexa.

Identificación Supervisada de Sistemas no Lineales de Wiener o Hammerstein

En muchas aplicaciones prácticas, las no linealidades observadas pueden ser modeladas como modelos más restrictivos basados en bloques, como los sistemas de Wiener (un filtro lineal seguido por una no linealidad estática, ver Fig. 2) y Hammerstein (una no linealidad estática seguida por un filtro lineal). A continuación el estudio se centra en la identificación e igualación de este tipo de sistemas no lineales.

En el capítulo 5 se propone un marco basado en el análisis de correlaciones canónicas con kernels ("kernel canonical correlation analysis", KCCA) que surge de manera natural al resolver problemas de identificación con este tipo de modelos no lineales en cascada. Para sistemas de Wiener, este método busca la proyección lineal de la señal de entrada y la proyección no lineal de la señal de salida que maximicen la correlación. La señal obtenida corresponde a una estima de la señal intermedia y desconocida del sistema de Wiener. La técnica de KCCA se obtiene para este problema al aplicar una restricción en la energía de la señal intermedia, en vez de la habitual restricción sobre la energía de los coeficientes de los filtros estimados. Mediante distintos experimentos se demuestra que esta restricción proporciona una mayor robustez ante ruido. Dada la similaridad entre los sistemas Wiener y Hammerstein, se puede aplicar la misma técnica para identificar sistemas de Hammerstein. Finalmente también se presentan extensiones de cada algoritmo que permiten igualar sistemas de Wiener y Hammerstein.

A continuación se desarrolla una ampliación de las técnicas propuestas a entornos adaptativos. Basándose en una solución adaptativa del problema del análisis de correlaciones canónicas (CCA) que aplica dos algoritmos RLS acoplados, se presenta una solución online del problema de KCCA. Esta nueva técnica se basa en el acoplo de dos algoritmos KRLS, lo cual permite reutilizar todos los métodos KRLS estudiados y propuestos en los capítulos 3 y 4. Al elegir un kernel lineal y otro no lineal en los algoritmos KRLS se obtiene un algoritmo capaz de identificar de manera adaptativa un sistema Wiener o Hammerstein.

Identificación Ciega de Sistemas de Wiener

En el capítulo 6 se considera el problema de identificación ciega de sistemas no lineales que se pueden modelar como sistemas de Wiener. Dentro de este problema, un resultado de carácter científico destacable es que se demuestra que el sobremuestreo permite la identificación ciega aplicando técnicas de KCCA. El algoritmo desarrollado puede aplicarse para identificar de manera ciega sistemas no lineales con varias salidas, tales como redes de sensores no lineales o canales no lineales sobremuestrados, siempre que sus relaciones de entrada-salida se pueden modelar como sistemas de Wiener (ver Fig. 3). El algoritmo propuesto estima de manera alternada las partes lineales y no lineales del sistema y, en contraste con otras técnicas, no impone restricciones en la señal de entrada al sistema. Además, el número mínimo de salidas necesarias para identificar el sistema es de 2, mientras que otras técnicas, que identifican sistemas Volterra de manera ciega, requieren un mayor número de salidas.

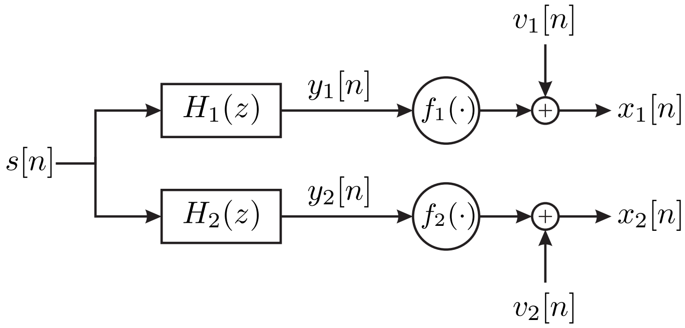


Figura 3: Un sistema no lineal con una entrada y dos salidas, en el cual cada relación entrada-salida se puede modelar como un sistema de Wiener. Los componentes $v_1[n]$ y $v_2[n]$ representan ruido aditivo.

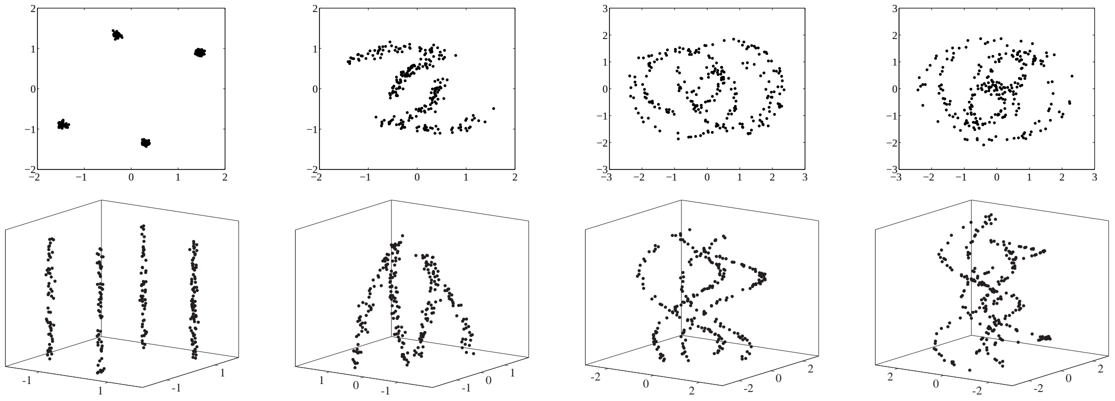
III. Separación Ciega de Fuentes Mediante Técnicas de Agrupamiento Espectral

En la tercera parte de esta tesis se estudian problemas de identificación ciega y de separación ciega que permiten ser tratados con algoritmos de agrupamiento ("clustering"). En particular, el tipo de problemas en los que las técnicas de clustering ofrecen una alternativa interesante son aquellos en los que las señales fuente forman parte de una alfabeto finito o cuando son dispersas. Después de una breve introducción a las técnicas de agrupamiento en el capítulo 7, se estudian dos de estos problemas para los que las técnicas tradicionales presentan deficiencias, y se proponen algoritmos basados en agrupamiento espectral ("spectral clustering") para resolver estos problemos.

Decodificación Ciega de Sistemas MIMO Rápidamente Variantes en el Tiempo

En primer lugar se estudia un problema de notable complejidad en el campo de las comunicaciones como es la decodificación ciega o semiciega (empleando únicamente un número muy reducido de símbolos piloto) de canales de múltiples entradas y múltiples salidas ("multiple-input multiple-output", MIMO) rápidamente variantes en el tiempo. Estos canales, que tienen una propagación Doppler ("Doppler spread") normalizada muy elevada, no permiten la aplicación de técnicas de decodificación ciega convencionales, dado que varían demasiado durante la transmisión de una trama de datos (ver Fig. 4, arriba). Este problema puede ocurrir cuando los transmisores o receptores de un sistema MIMO se mueven a gran velocidad o bien cuando la tasa de transmisión es muy baja en un sistema que se mueve lentamente. Es en estos dos casos cuando se puede producir solapamiento entre los conjuntos de puntos o clusters (que corresponden a los distintos símbolos transmitidos) en el scatter plot.

La técnica propuesta en el capítulo 8 incorpora la información temporal al scatter plot, lo cual convierte las nubes de puntos solapadas en hilos alargados que se



(a) $f_d T = 0$ (b) $f_d T = 0.001$ (c) $f_d T = 0.005$ (d) $f_d T = 0.01$

Figura 4: Efecto de la propagación Doppler normalizada en los símbolos recibidos. Arriba: Scatter plots de los datos recibidos por una antena de un sistema BPSK MIMO con 2 antenas transmisoras, para distintas propagaciones Doppler normalizadas $f_d T$. Abajo: Los mismos scatter plots a los cuales se ha añadido un eje temporal. Como resultado de los cambios en los canales durante la transmisión, se pueden observar hilos curvados en estos diagramas.

entrelazan sin cruzarse (ver Fig. 4, abajo). Además, para evitar el agrupamiento incorrecto que puede ocurrir cuando un cluster no contiene suficientes puntos, el método propuesto aprovecha la estructura de la constelación M-PSK utilizada dentro del algoritmo de agrupamiento.

Las técnicas de agrupamiento espectral propuestas emplean un kernel Gaussiano en el diseño de la matriz de afinidad. En una contribución adicional estudiamos alternativas que permiten incorporar más información a priori sobre el problema. Específicamente, el empleo de un nuevo kernel basado en caminos entre cada dos puntos (el denominado “connectivity kernel”) permite obtener soluciones mucho mejores que las obtenidas con un kernel Gaussiano.

En conclusión, las técnicas propuestas requieren un número de pilotos mucho más bajo comparado con técnicas adaptativas para la decodificación ciega. Dados el mismo número de símbolos piloto, estas técnicas obtienen resultados notablemente mejores en canales con Dopplers normalizados muy elevados.

Separación Ciega de Fuentes no Lineal en el Caso Indeterminado

En segundo lugar se estudian los problemas no lineales (“post-nonlinear”) de separación ciega de fuentes (“blind source separation”, BSS), que consisten en recuperar un número de fuentes que han sido mezcladas linealmente y después transformadas por una no linealidad sin memoria. Mientras que en la literatura existe un gran número de técnicas que tratan el problema post-nonlinear cuando el número de mezclas observadas es igual a, o mayor que el número de fuentes originales, en el capítulo 9 se trata el caso “indeterminado” de este problema, en el cual el número de mezclas es menor que el número de fuentes. Este problema se encuentra por ejemplo cuando un número de sensores que exhiben una no linealidad (como una saturación) miden un número más grande de indicadores.

La técnica propuesta requiere que exista un dominio en que las fuentes originales son señales dispersas. En estos casos, se puede aplicar la técnica de agrupamiento espectral al scatter plot para identificar instantes de tiempo en que sólo una fuente es activa. A continuación se aplica este conocimiento para estimar las transformaciones no lineales que se han aplicado en cada una de las mezclas no lineales. Una vez identificadas estas transformaciones no lineales, el problema se reduce a un problema de separación ciega lineal, para el cual existe un gran número de técnicas convencionales.

IV. Conclusiones

En esta tesis se han propuesto varias técnicas para aplicaciones de procesado de señal no lineal y comunicaciones, planteando mejoras al estado del arte de la identificación e igualación de sistemas no lineales, y de la separación ciega de fuentes no lineal.

Los resultados obtenidos en esta tesis han dado lugar a la publicación de un capítulo de libro, siete publicaciones en revistas internacionales y seis artículos en congresos internacionales, habiendo recibido en uno de ellos (European Signal Processing Conference 2007) el premio al mejor artículo de estudiante ("Best Student Paper Award"). En el apéndice G se presenta una lista completa de todas las publicaciones procedentes de este trabajo.

Así mismo, durante el desarrollo de esta tesis se ha realizado una estancia de tres meses entre septiembre de 2008 y diciembre de 2008 en el Computational Neuro-Engineering Laboratory (CNEL) de la University of Florida bajo la supervisión del Profesor Dr. José C. Principe. También se han realizado colaboraciones puntuales con otros investigadores como Dr. Paolo Emilio Barbano de la Yale University, Dr. Umut Ozertem de Yahoo! Research, el Profesor Dr. Deniz Erdogmus de la Northeastern University y Dr. Weifeng Liu de Amazon.com, en la temática de técnicas de aprendizaje máquina.

Notation and Acronyms

Used Notation

a	Scalar (lowercase)
$\mathbf{a}$	Column vector (lowercase boldface)
$\mathbf{a}[i]$	Instance of vector $\mathbf{a}$ at time step i
a_i	i-th component of vector $\mathbf{a}$
$\mathbf{a}_i$	i-th instance of $\mathbf{a}$, or the instance at time step i
$\mathbf{A}$	Matrix (uppercase boldface)
$\mathbf{A}_i$	Row i or column i of matrix $\mathbf{A}$ (depending on context)
A_{ij}	Element (i, j) of matrix $\mathbf{A}$
a^*	Complex conjugate of a
$\mathbf{A}^T$	Transpose
$\mathbf{A}^H$	Hermitian
$\mathbf{A}^\dagger$	Moore-Penrose pseudo-inverse of $\mathbf{A}$, i.e. $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$
$\|\mathbf{a}\|$	Euclidian norm of a vector, $\|\mathbf{a}\| = (\sum_i a_i^2)^{1/2}$
$\|\mathbf{A}\|_F$	Frobenius norm of a matrix, $\|\mathbf{A}\|_F = (\sum_i \sum_j A_{ij}^2)^{1/2}$
$\hat{\mathbf{a}}$	Estimate of $\mathbf{a}$
$E[x]$	Mathematical expectation of a random variable x
$\mathbb{R}$	Set of real numbers
$\mathbb{N}$	Set of natural numbers
$\mathcal{N}(\mu, \sigma)$	Normal distribution with mean μ and variance σ
$\langle \mathbf{x}_i, \mathbf{x}_j \rangle$	Inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$, also $\mathbf{x}_i^T \mathbf{x}_j$
$h[n] * s[n]$	Convolution of h and s, i.e. $h[n] * s[n] = \sum_{i=-\infty}^{+\infty} h[i]s[n-i]$
$f(\mathbf{x})$	Evaluation of a function $f = f(\cdot)$ in a point $\mathbf{x}$
$\phi(\mathbf{x})$, also $\tilde{\mathbf{x}}$	Transformation of $\mathbf{x}$ into feature space
$\mathcal{S}$	Set of points
$\|\mathcal{S}\|$	Cardinality of $\mathcal{S}$

Specific Variables

$\mathbf{I}_p$	Identity matrix of dimensions $p \times p$
$\mathbf{I}$	Identity matrix of appropriate dimensions
$\mathbf{0}$	Zero-matrix of appropriate dimensions

$\mathbf{K}$	Kernel matrix, $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$
α_i	Expansion coefficients
c	Regularization constant
λ	Forgetting factor
μ	Learning rate
N	Number of observations
n	Dimension of the observations, $n = \dim(\mathbf{x})$
m, also M	Reduced data dimension

Acronyms

AHC	Agglomerative Hierarchical Clustering
ALD	Approximate Linear Dependency
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase-Shift Keying
BSS	Blind Source Separation
CCA	Canonical Correlation Analysis
FB	Fixed-Budget
FIR	Finite Impulse Response
GEV	Generalized Eigenvalue [Problem]
ICA	Independent Component Analysis
ICD	Incomplete Cholesky Decomposition
i.i.d.	independently and identically distributed
IIR	Infinite Impulse Response
ITL	Information Theoretic Learning
KCCA	Kernel Canonical Correlation Analysis (also “kernel CCA”)
KDE	Kernel Density Estimation
KLMS	Kernel Least Mean Square (also “kernel LMS”)
KNN	K Nearest Neighbors
KPCA	Kernel Principal Component Analysis (also “kernel PCA”)
KRLS	Kernel Recursive Least-Squares (also “kernel RLS”)
KRR	Kernel Ridge Regression
LS	Least-Squares
LS-SVM	Least-Squares Support Vector Machine
LTI	Linear Translation-Invariant
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MMSE	Minimum Mean Square Error
M-PSK	M Phase Shift Keying
MSE	Mean Square Error

OSTBC	Orthogonal Space-Time Block Coding
PCA	Principal Component Analysis
PCSI	Perfect Channel State Information
pdf	Probability Density Function
PNL	Post-Nonlinear
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RBF	Radial Basis Function
RKHS	Reproducing Kernel Hilbert Space
RLS	Recursive Least-Squares
SIMO	Single-Input Multiple-Output
SISO	Single-Input Single-Output
SNR	Signal to Noise Ratio
SPCL	Spectral Clustering
STBC	Space-Time Block Coding
SVD	Singular Value Decomposition
SVM	Support Vector Machine
SW	Sliding-Window
V-BLAST	Vertical Bell Laboratories Layered Space-Time
ZF	Zero Forcing

Contents

Part I

Introduction and Background

Chapter 1
Introduction

Nonlinear phenomena are encountered in many engineering problems. Traditional signal processing techniques are linear, which makes them unable to extract the complex, nonlinear patterns that may lie in the data available in such scenarios. Therefore, problems concerning nonlinear data analysis have traditionally been tackled by polynomial filters [Mathews and Sicuranza, 2000], which provide straightforward extensions of many linear methods, or by neural network approaches [Haykin, 1999], which are able to *learn* nonlinear relationships.

In [Haykin, 1999], learning was defined as “the process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded”. Hence, the goal of a learning process is to make the network respond in a certain desired way to its environment. The learning process itself ensures that the performance of the network improves with experience.

This definition of learning applies to a broad class of *machine learning* systems. In general, machine learning concerns learning processes that are not based on a set of predefined rules, but learn relations from the data itself. There exist many types of learning, including *supervised* learning, in which the desired responses for all training data are given, *unsupervised* (or *blind*) learning, in which a data set is provided without the corresponding desired responses, and reinforcement learning, where only indirect feedback on the performance is available.

Two important concepts to keep in mind when designing a learning machine are *capacity* and *generalization* [Vapnik, 1995]. The capacity of a learning machine refers to the capability of this machine to represent complex and highly nonlinear functions. The generalization capability allows a learning machine to generalize beyond the training data to new, unseen data. Clearly, there exists a trade-off between the capacity and the generalization capability of a learning machine, as a high capacity will allow to represent the patterns in the training data very accurately, but it will usually not generalize well to new data.

In contrast to linear techniques, which typically offer elegant formulations and efficient algorithms, nonlinear learning machines such as neural networks require more computation and often involve nonlinear optimization problems with multiple local minima. An attractive alternative framework is offered by *kernel methods* [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004]. Kernel methods are powerful machine learning techniques that exhibit a less complex architecture and provide a straightforward approach to transform nonlinear problems into convex optimization problems. Common analysis tasks in kernel-based learning compromise classification, regression and clustering. In this thesis, we will focus on the latter two, and we will study the application of kernel methods to three basic problems in signal processing, specifically nonlinear system identification, nonlinear system equalization and nonlinear blind source separation.

1.1 Kernels as Similarity Measures

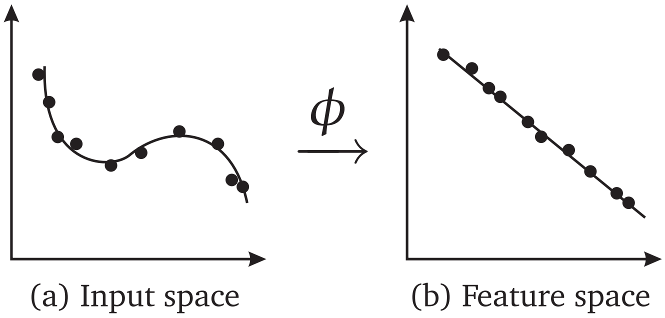
A central problem in machine learning is to discover structure in data. In regression, for instance, we are given a series of training data pairs $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ that correspond to the input data $\mathbf{x}_i$ of an observed, in general nonlinear system and its corresponding output y_i, for $i = 1, \ldots, N$. The goal of regression is to discover the underlying functional relationship between input and output of this system, with the goal of being able to predict the system's output when a new input data point is presented.

A key idea here is that similar inputs need to lead to similar outputs. When a new point is presented to the algorithm, it will compare this new point with all known previous points in order to produce an output. To this end, we will make use of the notion of kernels. A kernel can be thought of as a measure of input data similarity. It is a symmetric function that takes as an input two data points and produces a real number that measures the similarity between those points. Interestingly, kernel methods are not restricted to numerical data alone, as they can be applied whenever similarity between different objects can be measured as a scalar.

Kernel methods are based on the rigorous mathematical framework of reproducing kernel Hilbert spaces (RKHS), which will be discussed in the next chapter. The practical consequence of using this framework is that it allows to compute inner products in a high, possibly infinite-dimensional *feature space* as kernel functions in the input space. This property allows to transform linear inner product based techniques into a high-dimensional space by simply changing their inner products into kernels. When solving a kernel-based algorithm, one implicitly solves the linear algorithm in feature space, where the transformed data is more likely to correspond to a linear model (see Fig. 1.1).

1.2 Why use Kernel Methods?

Both neural networks and kernel methods are universal function approximators (see for instance [Hornik et al., 1990] and [Micchelli et al., 2006], respectively), which means that they can approximate a nonlinear mapping with any given accuracy. However, neural networks usually require a high number of parameters, and their optimal configuration is found by performing an iterative nonlinear optimization process, often implemented through back-propagation. For a multitude of



(a) Input space (b) Feature space

Figure 1.1: The basic idea of kernel methods. The mapping ϕ transforms the input data points (black dots) into a high-dimensional *feature space*, where they can be described by a linear model (straight solid line). The linear model found in feature space corresponds to a nonlinear model in the input space (curved solid line).

problems, this training procedure is slow, and it does not guarantee convergence to the optimal solution but rather encounters one of the multiple local minima [Boyd and Vandenberghe, 2004].

Kernel methods, on the other hand, generally admit a more elegant solution which stems from the framework of RKHS and the convexity of the resulting optimization problem. Therefore, much kernel-based algorithms have a unique global solution that can be found by solving a convex optimization problem. As a result, although kernel methods are only a decade old, they now represent an established framework to solve machine learning problems and they are backed by an extensive list of experimental accomplishments. Some of the best known kernel methods are support vector machines (SVM) [Vapnik, 1995], kernel principal component analysis (kernel PCA) [Schölkopf et al., 1998], kernel-based regression techniques [Schölkopf and Smola, 2002], kernel canonical correlation analysis (kernel CCA) [Bach and Jordan, 2002, Hardoon et al., 2003], kernel Fisher discriminant analysis (KFD) [Mika et al., 1999] and spectral clustering [Ng et al., 2001]. Successful applications of these algorithms have been reported in many fields, such as image processing, computational biology, bioinformatics, communications and medicine.

1.3 Goal of the Thesis

The aim of this thesis is to design a set of kernel-based algorithms to solve a number of related nonlinear problems in signal processing and communications. The motivation for this work lies in the fact that kernel methods are capable of efficiently and accurately solving nonlinear problems at the cost of only moderate computational and memory complexities, which makes them suitable for possible real-time implementations.

First, the problem of kernel-based nonlinear adaptive filtering will be treated. Although the concept of performing adaptive filtering in RKHS is promising, several difficulties need to be addressed, including overfitting problems that arise from the high dimensionality of the RKHS, and the growing support of online kernel methods.

In many practical applications, the observed nonlinearities can be modeled as more restricted block-based models, such as Wiener and Hammerstein systems. Therefore, the second problem in this thesis will consist in developing kernel-based supervised identification and equalization algorithms for these model-based nonlinear systems. The obtained algorithms will be extended to online scenarios by incorporating the developed kernel adaptive filtering techniques.

Next, a number of blind problems will be treated that cannot be solved by traditional linear algorithms, including blind identification of nonlinear systems and nonlinear blind source separation. Techniques will be proposed for blind identification of Wiener systems and blind decoding of fast time-varying multiple-input multiple-output channels, which stem from the area of communications. Lastly, the problem of underdetermined post-nonlinear blind source separation will be dealt with, which emerges when a certain number of signals is measured by a smaller number of sensors that show some nonlinear behavior.

1.4 Outline and Contributions

While there is a close relationship between most of the problems dealt with, this thesis will be divided into different parts that correspond to the different techniques used to construct the proposed algorithms.

The first part presents the required background on kernel methods and a literature survey of the most relevant previous work. More specifically, it contains the following chapters.

- In **chapter 1** we present a motivation for this work.
- **Chapter 2** explains the theoretical framework on which kernel methods are based. We describe the concept of regularization and present two examples of kernel algorithms that will be fundamental for the development of other algorithms in this thesis.
- In **chapter 3** we give an overview of state-of-the-art online kernel methods. We show how they are derived as kernel-based versions of classical adaptive filtering methods, and we provide the outline of each algorithm.

In the rest of this thesis we present our own contributions to the field. First, we treat the problems of nonlinear regression and nonlinear system identification with kernels, in part II.

- **Chapter 4** starts with a general discussion of nonlinear system identification. We choose the class of Wiener and Hammerstein systems to work with in the following chapters, which are block-based nonlinear systems that provide a trade-off between simplicity and modeling capacity. As a first contribution, we introduce a family of online kernel learning algorithms that implement a recursive least-squares (RLS) algorithm with fixed memory size in feature space. Unlike most online algorithms presented in chapter 3, these algorithms are capable of forgetting older or irrelevant data, which allows them to track time-varying systems.

- In **chapter 5** we propose a supervised method to identify Wiener and Hammerstein systems. We demonstrate how a kernel canonical correlation analysis (kernel CCA) solution emerges for a properly chosen cost function and minimization criterion. We derive batch and online versions, with different regularization options and show how they can be used for supervised identification and equalization of both systems.

- In **chapter 6** we present an algorithm for blind equalization of single-input multiple-output (SIMO) nonlinear systems, in which each nonlinear channel is a Wiener system. The proposed method combines ideas from blind linear SIMO identification with kernel canonical correlation analysis (kernel CCA) to identify the nonlinearities. It is shown that the blind identification problem can be solved in an iterative manner, alternating between a CCA problem (to estimate the linear filters) and a kernel CCA problem (to estimate the memoryless nonlinearities). The resulting technique can be applied to nonlinear SIMO systems with two or more outputs.

Part III of this thesis focusses on problems of blind decoding and nonlinear blind source separation that can be treated with spectral clustering based techniques.

- In **chapter 7** we introduce the problem of data clustering. In particular, we discuss the technique of spectral clustering, which will be used as the core technique of the clustering problems in the following two chapters.

- **Chapter 8** deals with clustering techniques for mixtures of signals that belong to a finite alphabet, such as communication symbols. When the mixture process changes in time, the clusters form complex shapes that cannot be recovered by classical clustering algorithms. Therefore, such time-varying problems are usually treated with adaptive algorithms that require a training sequence. We propose a spectral clustering algorithm that is capable of performing this task in a semi-blind fashion. Specifically, the only training data it needs consist of the minimal number of bits to remove ambiguity. We also present extensions that increase the algorithm's robustness by optimizing the kernel function.

- In **chapter 9** we consider underdetermined mixture problems in which the source signals are sparse. For linear mixtures a number of techniques exist that exploit the sparseness to identify the mixing process and recover the sources. We show how spectral clustering can be applied in nonlinear scenarios to retrieve the nonlinear directions in the measurement scatter plot, thereby identifying the mixing process.

Part IV summarizes the main conclusions of this thesis, in **chapter 10**, along with a number of future directions for the proposed techniques. We include a number of appendices and bibliographic references.

The relationship between the different chapters and parts of this thesis is summarized in the diagram of Fig. 1.2.

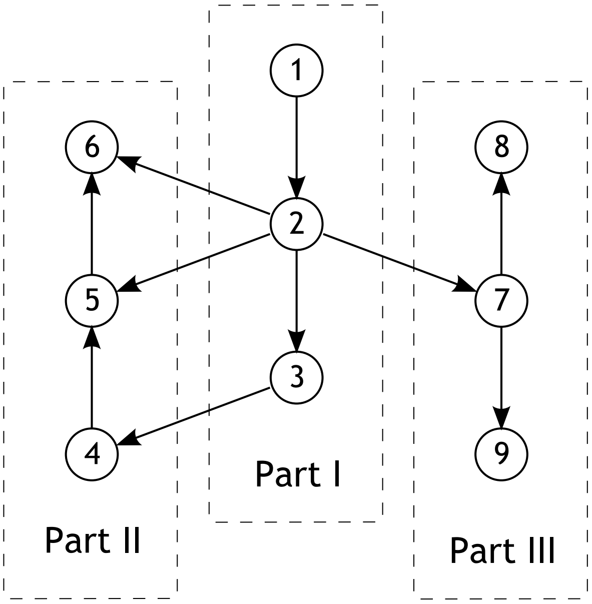


Figure 1.2: The structure of this thesis. Each circle represents a chapter, and the arrows represent direct relationships between the different chapters. Starting from chapter 1, various paths can be followed to read this thesis.

The contributions of this thesis are summarized at the end of each corresponding chapter. Additionally, a full listing of the resulting publications is included in appendix G. The obtained results have lead to the publication of one book chapter, seven papers in international journals and six papers presented at international conferences. One of these publications was awarded a Best Student Paper Award at the 2007 European Signal Processing Conference.

During the development of this thesis, the author has paid a scholarly visit of three months between September 2008 and December 2008 to the Computational NeuroEngineering Laboratory (CNEL) of the University of Florida, under supervision of Professor Dr. José C. Principe. A number of collaborations have also been established on the subject of machine learning techniques with other researchers including Dr. Paolo Emilio Barbano of Yale University, Dr. Umut Ozertem of Yahoo! Research, Professor Dr. Deniz Erdogmus of Northeastern University and Dr. Weifeng Liu of Amazon.com.

Chapter 2 Kernel Methods

This chapter is an introduction to the theoretical framework on which kernel methods are built. We discuss the mathematical concepts of reproducing kernel Hilbert spaces, the nonlinear mapping of data into feature space, its associated kernel function and regularization. We also illustrate the introduced concepts with two basic kernel algorithms that will be at the core of many techniques proposed in this thesis.

2.1 Kernels and Feature Spaces

We start by introducing some definitions and the used terminology. In general lines, we will follow the exposition given in [Schölkopf and Smola, 2002].

2.1.1 Basic definitions

Definition 2.1 ((Positive Definite) Kernel). *A* kernel *[Aronszajn, 1950] is a continuous, symmetric function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that operates on data in an input space $\mathcal{X}$. A kernel is called* positive definite *if for any set of input data points $\{\mathbf{x}_i\}_{i=1}^N \in \mathcal{X}$ it satisfies the condition*

$$\sum_{i,j=1}^{N} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \quad \forall \alpha_i \in \mathbb{R}. \tag{2.1}$$

A number of common kernels, such as the Gaussian kernel $\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$ are listed in section 2.1.5.

Definition 2.2 (Kernel Matrix). *For a given set of N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the $N \times N$ matrix $\mathbf{K}$ with elements $K_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ is called the* kernel matrix *(or* Gram matrix*) of κ with respect to the data, for $i, j = 1, \ldots, N$.*

A *positive definite kernel matrix* is obtained by constructing a kernel matrix using a positive definite kernel [1].

[1] Some authors prefer to call the corresponding matrices positive *semi*-definite, and reserve the term positive definiteness only for matrices that are constructed using a *strictly* positive definite kernel, i.e. a kernel for which (2.1) is strictly positive.

Definition 2.3 (Positive Definite Matrix). *A square real-valued matrix* **K** *satisfying*

$$\sum_{i,j=1}^{N} \alpha_i \alpha_j K_{ij} \geq 0, \qquad \forall \alpha_i \in \mathbb{R}, \tag{2.2}$$

is called a positive definite *matrix.*

This condition is equivalent to requiring that $\boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha} \geq 0, \forall \boldsymbol{\alpha} \in \mathbb{R}^N$. For the kernel matrix this translates into all of its eigenvalues being nonnegative.

2.1.2 Reproducing kernels and Hilbert spaces

It can be shown that a *feature space* can be found that is associated with a positive definite kernel such that the kernel is an inner product in that feature space. With the aim of constructing such a feature space, let us start by defining a *feature mapping* from $\mathcal{X}$ into the space of functions $\mathcal{H}$, for a given positive definite kernel κ,

$$\begin{aligned} \phi \ : \ & \mathcal{X} \to \mathcal{H} \\ & \mathbf{x} \mapsto \kappa(\mathbf{x}, \cdot). \end{aligned} \tag{2.3}$$

The function $\phi(\mathbf{x})(\cdot)$ assigns the value $\kappa(\mathbf{x}, \mathbf{x}')$ to the input point $\mathbf{x}'$. By interpreting the kernel function as a similarity function, this mapping represents every input point $\mathbf{x}$ by its similarity $\kappa(\mathbf{x}, \cdot)$ to all other points on the domain $\mathcal{X}$.

In order to construct a feature space associated with ϕ, the image of ϕ must be turned into a vector space and endowed with an inner product [Schölkopf and Smola, 2002]. A possible vector space can be defined by taking linear combinations of the form

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{x}_i, \cdot), \tag{2.4}$$

where m, α_i and $\mathbf{x}_i$ are chosen arbitrarily, and $i = 1, \ldots, m$. The inner product between f and another function $g(\cdot) = \sum_{j=1}^{m'} \beta_j \kappa(\mathbf{x}'_j, \cdot)$ in this space is defined as

$$\langle f, g \rangle := \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j \kappa(\mathbf{x}_i, \mathbf{x}'_j). \tag{2.5}$$

An interesting property that follows directly from the definition of ϕ is that all functions of the form (2.4) satisfy

$$\langle \kappa(\mathbf{x}, \cdot), f \rangle = f(\mathbf{x}). \tag{2.6}$$

In other words, κ is the representer of the evaluation of f. In particular, the kernel κ possesses the *reproducing property* [Aronszajn, 1950],

$$\langle \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}', \cdot) \rangle = \kappa(\mathbf{x}, \mathbf{x}'). \tag{2.7}$$

Therefore, positive definite kernels are also called *reproducing kernels*.

The previous description shows that any positive definite kernel has an associated feature space where it can be thought of as a inner product,

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle. \tag{2.8}$$

However, notice that this mapping only defines one possible way of constructing a feature space, and we will later discuss a different, explicit feature mapping that leads to another feature space.

Kernel trick

Up till this point we showed how a feature map can be constructed from a kernel. Interestingly, the converse also holds. For every mapping ϕ from the input space $\mathcal{X}$ to an inner product space, a positive definite kernel is obtained from (2.8) (see [Schölkopf and Smola, 2002] for the proof). This duality between positive definite kernels and feature spaces gives rise to the property that is commonly called the "kernel trick": *"Given an algorithm that is formulated in terms of a positive definite kernel κ, one can construct an alternative algorithm by replacing κ by another positive definite kernel κ'."*

The best known application of the kernel trick, and also the main idea behind many techniques used in this thesis, follows directly from the case where one kernel is chosen as the inner product in the input space. In this situation, the kernel trick states that we can transform any inner product based algorithm to an alternative algorithm by replacing the inner products with a nonlinear kernel. According to identity (2.8), performing this alternative kernel-based algorithm is equivalent to applying the original inner product based algorithm in the feature space. The latter algorithm is usually difficult or even impossible to carry out explicitly due to the high dimensionality of the feature space. However, thanks to (2.8), we can simply apply the equivalent kernel-based algorithm in the input space. The strength of this simple and elegant idea is that while the obtained solution is a nonlinear function of the input data (through the used nonlinear kernel), it is obtained by performing a convex optimization problem *implicitly* in the feature space.

The advantages of moving to a higher-dimensional space in classification problems were already shown by the theorem of Cover [Cover, 1965], which demonstrates that it is more probable that a problem is linearly separable in such a space.

Theorem 2.4 (Cover's Theorem)**.** *There are $C(N, n)$ possible linear separations of N points in a an n-dimensional Euclidian space, where*

$$C(N, n) = 2 \sum_{i=0}^{n-1} \binom{N-1}{i}. \tag{2.9}$$

If the dimensionality of the space is higher than the number of points, $n > N$, then all 2^N separations are possible. Note that this theorem is only a guideline and there exist specific configurations that do not satisfy it, for instance when all points lie on a low-dimensional manifold.

Reproducing kernel Hilbert spaces

By completing the previously-defined inner product spaces with a corresponding norm $\|f\| := \sqrt{\langle f, f \rangle}$, one can turn them into *Hilbert spaces*, which have some interesting mathematical properties. Formally, a Hilbert space $\mathcal{H}$ is an inner product space that is complete in the sense of Cauchy, i.e. a space in which every Cauchy sequence $\{\mathbf{x}^{(k)}\}$ necessarily converges to a unique $\mathbf{x} \in \mathcal{H}$. Examples of Hilbert spaces are the vector space of n-tuples $\mathbb{R}^n$ equipped with the inner product $\langle \mathbf{x}, \mathbf{x}' \rangle = \mathbf{x}^T \mathbf{x}'$, and the set L_2 of all square-integrable functions, whose inner product is $\langle f, g \rangle = \int f(\mathbf{x}) g(\mathbf{x}) d\mathbf{x}$. Note that the latter is a case of an infinite-dimensional Hilbert space. By introducing the concept of reproducing kernels, one obtains the following family of spaces:

Definition 2.5 (Reproducing Kernel Hilbert Space). *Let $\mathcal{X}$ be a non-empty set and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a* reproducing kernel Hilbert space (RKHS) *endowed with the inner product $\langle \cdot, \cdot \rangle$ and the norm $\|f\| := \sqrt{\langle f, f \rangle}$, if there exists a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties:*

1. *κ has the reproducing property*

$$\langle f, \kappa(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}), \quad \forall f \in \mathcal{H}, \forall \mathbf{x} \in \mathcal{X}, \tag{2.10}$$

and in particular, $\langle \kappa(\mathbf{x}, \cdot), \kappa(\mathbf{x}', \cdot) \rangle = \kappa(\mathbf{x}, \mathbf{x}')$.

2. *κ spans $\mathcal{H}$, i.e. every $f \in \mathcal{H}$ can be written as*

$$f(\cdot) = \sum \alpha_i \kappa(\mathbf{x}_i, \cdot), \tag{2.11}$$

where α_i are some real-valued coefficients.

Following this definition, the RKHS $\mathcal{H}$ uniquely determines a kernel κ. However, the opposite is not true, as will be illustrated in the following.

2.1.3 Mercer feature mapping

In this section, we will describe a procedure based on *Mercer's theorem* to obtain a specific Hilbert space associated to a given kernel κ. Although Hilbert spaces are isometrically isomorphic and therefore the obtained space will be equivalent to any other constructed RKHS, this theorem is interesting since it provides insight toward the understanding of kernel methods.

Theorem 2.6 (Mercer's Theorem). *Given a compact input space $\mathcal{X}$ and the set L_2 of all square integrable functions, i.e. $\int f(\mathbf{x})^2 d\mathbf{x} < \infty$, $\forall f(\cdot) \in L_2$. If κ is a symmetric real-valued function such that for all $f(\cdot) \in L_2$ we have*

$$\int_{\mathcal{X}} \kappa(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mathbf{x} d\mathbf{x}' \geq 0, \tag{2.12}$$

then it can be expanded as

$$\kappa(\mathbf{x},\mathbf{x}') = \sum_{i=1}^{n_{\mathcal{H}}} \lambda_i \psi_i(\mathbf{x})\psi_i(\mathbf{x}'), \tag{2.13}$$

where ψ_i and λ_i are the eigenfunctions and (nonnegative) eigenvalues of κ, respectively, and $g(\cdot) \in L_2$ [Mercer, 1909].

The requirement (2.12) in this theorem is called *Mercer's conditions*, and the obtained kernel is called a *Mercer kernel*. Notice that from (2.12) it follows that a Mercer kernel is also positive definite.

An important consequence of the Mercer theorem is that it allows us to construct an explicit *Mercer kernel map* ϕ of a data point $\mathbf{x}$ as

$$\phi(\mathbf{x}) = \left[\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots, \sqrt{\lambda_{n_{\mathcal{H}}}}\psi_{n_{\mathcal{H}}}(\mathbf{x})\right]^T. \tag{2.14}$$

The dimensionality of this space depends on the number of nonnegative eigenvalues, $n_{\mathcal{H}}$, which in case of the Gaussian kernel is infinite.

It can be shown that for every Mercer kernel κ there exists an RKHS $\mathcal{H}$ of functions defined over $\mathcal{X}$ for which κ is the reproducing kernel. To this end, we consider a kernel that satisfies (2.12) and the Hilbert space $\mathcal{H}$ containing the functions

$$f(\mathbf{x}) = \sum_{i=1}^{\infty} \alpha_i \kappa(\mathbf{x}_i,\mathbf{x}) = \sum_{i=1}^{\infty} \alpha_i \sum_{j=1}^{n_{\mathcal{H}}} \lambda_j \psi_j(\mathbf{x}_i)\psi_j(\mathbf{x}). \tag{2.15}$$

By constructing an inner product

$$\langle f, \kappa(\mathbf{x},\cdot)\rangle = \sum_{i=1}^{\infty} \alpha_i \sum_{j,n=1}^{n_{\mathcal{H}}} \lambda_j \psi_j(\mathbf{x}_i)\langle \psi_j, \psi_n\rangle \lambda_n \psi_n(\mathbf{x}), \tag{2.16}$$

and choosing $\langle \psi_j, \psi_n\rangle = \delta_{jn}/\lambda_j$, the Mercer kernel κ becomes a reproducing kernel for this Hilbert space, i.e. $\langle f, \kappa(\mathbf{x},\cdot)\rangle = f(\mathbf{x})$.

Interestingly, the Mercer kernel map and the RKHS kernel map lead to different feature spaces. However, since they are both Hilbert spaces, they are isometrically isomorphic, i.e. there exists a one-to-one linear mapping between them. Therefore, in this thesis we will not distinguish between them.

2.1.4 Universal approximation and the Representer Theorem

An interesting property of kernel methods is that there exist several kernels that can approximate any continuous function arbitrarily well [Steinwart, 2001, Micchelli et al., 2006]. Formally, a kernel κ possesses the *universal approximation property* if for any continuous mapping $f : \mathcal{X} \to \mathbb{R}$ and any accuracy $\epsilon > 0$, there exists a set of input points $\{\mathbf{c}_i\}_{i=1}^m \in \mathcal{X}$ and real numbers $\{\alpha_i\}_{i=1}^m$, with $m \in \mathbb{N}$, such that

$$\|f(\cdot) - \sum_{i=1}^{m} \alpha_i \kappa(\mathbf{c}_i,\cdot)\| \le \epsilon. \tag{2.17}$$

Furthermore, in the Hilbert space $\mathcal{H}$ associated with κ we can define a corresponding vector $\tilde{\mathbf{h}} \in \mathcal{H}$ as

$$\tilde{\mathbf{h}} = \sum_{i=1}^{m} \alpha_i \phi(\mathbf{c}_i). \tag{2.18}$$

By using $\tilde{\mathbf{h}}$ as a linear operator in $\mathcal{H}$,

$$\|f(\cdot) - \tilde{\mathbf{h}}^T \boldsymbol{\phi}(\cdot)\| \leq \epsilon, \tag{2.19}$$

it follows that the linear model in $\mathcal{H}$ possesses the universal approximation property. Note that we will adopt a tilde to denote variables in feature space.

In order to determine what points $\mathbf{c}_i$ should be used in the expansion (2.18), one can make use of the Representer Theorem [Kimeldorf and Wahba, 1971]. This theorem states that the solutions of certain risk minimization problems involving a quadratic regularizer can be written as expansions in terms of the training data. In [Schölkopf et al., 2001] a more general Representer Theorem was proposed in the context of kernel methods:

Theorem 2.7 (Representer Theorem). *Denote by $\mathcal{X}$ a set, by κ a positive definite kernel on $\mathcal{X} \times \mathcal{X}$, by $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\} \in \mathcal{X} \times \mathbb{R}$ a data set, by $L : (\mathcal{R} \times \mathbb{R})^N \to \mathbb{R}$ an arbitrary cost function and by $\Omega : \mathbb{R} \to \mathbb{R}$ any strictly monotonic increasing function. Then, each $f \in \mathcal{H}$ that minimizes the regularized minimization problem*

$$\min_{f \in \mathcal{H}} J(f) = L\left((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \cdots, (\mathbf{x}_N, y_N, f(\mathbf{x}_N)\right) + \Omega(\|f\|_{\mathcal{H}}) \tag{2.20}$$

can be written as a kernel expansion of the input data points, of the form

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}). \tag{2.21}$$

In practice, we will mostly use a pointwise mean squared loss function L for which the minimization problem (2.20) reads

$$\min_{f \in \mathcal{H}} J(f) = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 + \Omega(\|f\|_{\mathcal{H}}). \tag{2.22}$$

The strength of the Representer Theorem is that it allows to find the possibly infinite-dimensional solution $\tilde{\mathbf{h}}$ of a minimization problem in feature space as a finite combination of transformed data points $\tilde{\mathbf{x}}_i$. In this manner, the *primal problem* of finding an optimal vector $\tilde{\mathbf{h}} \in \mathcal{H}$ is converted into the computationally much more attractive *dual problem* of finding the N optimal expansion coefficients $\alpha_i \in \mathbb{R}$ of $\tilde{\mathbf{h}}$.

2.1.5 Common kernel functions

In this section we list some commonly used kernel functions. Note that a kernel function represents an inner product in some feature space, but it does not require the input data to have a vector representation.

Polynomial kernel. The polynomial kernel or order p is obtained as

$$\kappa(\mathbf{x}, \mathbf{x}') = \left(\langle \mathbf{x}, \mathbf{x}' \rangle + c\right)^p, \tag{2.23}$$

where c is a nonnegative constant, usually 1. For low orders p, it is easy to obtain an explicit feature mapping ϕ of $(\langle \mathbf{x}, \mathbf{x}' \rangle + c)^p$ by decomposing it as a weighted sum of monomials, using the binomial theorem.

Linear kernel. As a special case of the polynomial kernel of order 1, the linear kernel is sometimes used

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \mathbf{x}, \mathbf{x}' \rangle = \mathbf{x}^T \mathbf{x}'. \tag{2.24}$$

Gaussian kernel. The Gaussian kernel (or radial basis function (RBF) kernel), is a commonly used kernel function in kernel density estimation (KDE), such as Parzen density estimation [Parzen, 1962]. It is calculated as

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right), \tag{2.25}$$

where σ is a scaling constant. It can be shown that the Gaussian kernel spans an infinite-dimensional feature space. This does not pose a computational problem, though, since the kernel trick allows us to calculate the scalar product between two points in this infinite-dimensional space by simply calculating the kernel function (2.25) of the data in input space. Note that the factor 2 in the denominator is sometimes left out, in which case it is assumed that it is included in the factor σ^2.

Sigmoid kernel. Much like the RBF kernel, the sigmoid kernel stems from neural network theory. It is obtained as

$$\kappa(\mathbf{x}, \mathbf{x}') = \tanh(a\langle \mathbf{x}, \mathbf{x}' \rangle + c), \tag{2.26}$$

where $a, c \in \mathbb{R}$ and are suitable constants. Note that this kernel is not positive definite.

Sequence kernels. Sequence kernels compute the similarity between two sequences of strings taken from some alphabet [Lodhi et al., 2002]. These sequences do not necessarily need to have the same length. Sequence kernels are commonly used in bioinformatics (for instance in biological sequence analysis [Sonnenburg et al., 2005, Ben-Hur et al., 2008]) and the learning of formal languages [Kontorovich et al., 2008].

Time-series kernels. These kernels extend the concept of sequence kernels to time-series [Rüping, 2001, Cuturi et al., 2007]. They are applied for instance in the problem of dynamic time-warping (DTW).

2.2 Example Algorithms

Some of the most powerful kernel-based algorithms can be found in the areas of classification, regression and clustering. In this work we will focus mainly on applications of kernel regression (in part II) and clustering (in part III).

In this section we illustrate two common kernel algorithms, which are obtained directly by mapping linear algorithms into feature space. The first one is nonlinear regression with kernels, which is a classical supervised learning problem. The second one is a kernel-based implementation of principal component analysis (PCA), which deals with the extraction of principal directions along which the data lie. No labels are provided in the second example, which makes this fundamentally a blind algorithm.

Care must be taken in constructing these algorithms, since the high dimensionality of the feature space leads to certain difficulties that need to be addressed correctly. We discuss these difficulties and a few measures that can be taken to overcome them in section 2.3.

2.2.1 Kernel regression

Ridge regression Suppose we are given a set of data points $\mathbf{x}_i \in \mathbb{R}^n$ and their target images $y_i \in \mathbb{R}$, and we are asked to retrieve a function that models the underlying mapping $y = f(\mathbf{x})$. This is the well-known problem of regression, and a linear solution can be found by modeling this function as $y = \mathbf{x}^T\mathbf{h} + \epsilon$, where ϵ is an error term. If the square norm of this error is minimized, we obtain the least-squares (LS) criterion [Sayed, 2003], which is a widely used method in signal processing. The solution is found as the vector $\mathbf{h} \in \mathbb{R}^n$ that minimizes the functional

$$\min_{\mathbf{h}} J(\mathbf{h}) = \|\mathbf{y} - \mathbf{X}\mathbf{h}\|^2, \tag{2.27}$$

in which we introduced the data matrix $\mathbf{X} \in \mathbb{R}^{N\times n}$ that contains the N input patterns $\mathbf{x}_i$ stacked as its rows, i.e. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$, and the vector $\mathbf{y} \in \mathbb{R}^N$ that contains all images y_i. Typically, the number of observations N is larger than the number of unknowns n, in which case the LS problem is *overdetermined*. Then, if $\mathbf{X}^T\mathbf{X}$ is a nonsingular matrix, the unique minimum-norm solution of the least-squares problem is given by

$$\mathbf{h} = \mathbf{X}^\dagger\mathbf{y} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{2.28}$$

To improve generalization, the L_2 norm of the solution can be used as a penalty term

$$\min_{\mathbf{h}} J(\mathbf{h}) = \|\mathbf{y} - \mathbf{X}\mathbf{h}\|^2 + c\mathbf{h}^T\mathbf{h}, \tag{2.29}$$

where c is a regularization constant that controls the smoothness of the solution. This type of regression, commonly known as *ridge regression*, was introduced by Tikhonov [Tikhonov, 1963] as a remedy for ill-posedness. Essentially, it establishes a tradeoff between fitting the training data and reducing the norm of the solution. Taking the derivative of the cost function with respect to $\mathbf{h}$ yields

$$\mathbf{X}^T\mathbf{X}\mathbf{h} - \mathbf{X}^T\mathbf{y} + c\mathbf{h} = \mathbf{0}, \tag{2.30}$$

which is solved by

$$\mathbf{h} = \left(\mathbf{X}^T\mathbf{X} + c\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{2.31}$$

where $\mathbf{I}$ is the identity matrix[2].

As mentioned earlier, the Representer Theorem [Kimeldorf and Wahba, 1971] states that the solution $\mathbf{h}$ of minimization problems such as (2.29) can be written as a linear expansion of the training data points. Specifically, we can rewrite (2.30) in terms of $\mathbf{h}$ as

$$\mathbf{h} = c^{-1}\mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{h}), \tag{2.32}$$

or, by introducing a set of coefficients α_i,

$$\mathbf{h} = \mathbf{X}^T\boldsymbol{\alpha} = \sum_{i=1}^{N}\alpha_i\mathbf{x}_i. \tag{2.33}$$

The optimal coefficients can be obtained as follows:

$$\boldsymbol{\alpha} = c^{-1}(\mathbf{y} - \mathbf{X}\mathbf{h}) \tag{2.34}$$

$$\Rightarrow c\boldsymbol{\alpha} = \mathbf{y} - \mathbf{X}\mathbf{X}^T\boldsymbol{\alpha}. \tag{2.35}$$

Finally, we obtain

$$\boldsymbol{\alpha} = \left(\mathbf{X}\mathbf{X}^T + c\mathbf{I}\right)^{-1}\mathbf{y}, \tag{2.36}$$

which is similar to (2.31). This yields two different forms of solving the ridge regression problem (2.29). Equation (2.31) is called the *primal solution*. It obtains the weight vector explicitly, based on the computation of the covariance matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{n\times n}$. On the other hand, equation (2.36) gives the solution as a linear combination of the training data. This solution is known as the *dual solution*. Notice that it is based on the Gram matrix $\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{N\times N}$. Although it requires more computation than the primal solution since we usually have $n \ll N$, the information of the training samples is given in inner products, which allows to apply the kernel trick. As we will see in the following, when performing ridge regression in feature space it is more worthwhile to solve the dual problem, as the dimension of the feature space, n', is usually much higher than the number of training data points, $n' \gg N$.

[2] Notice that the images y_i do not necessarily need to be one-dimensional. If m-dimensional images are used, the regression problem can be seen as m independent one-dimensional problems. The combined solution $\mathbf{h} \in \mathbb{R}^{n\times m}$ of these problems is still obtained by (2.31).

Kernel ridge regression If the data show nonlinear relationships, the previous linear regression technique will be unable to model them adequately. However, a nonlinear solution can be found by moving to feature space. Kernel ridge regression (KRR) [Saunders et al., 1998, Evgeniou et al., 2000] is a technique that finds a nonlinear mapping f represented as a kernel expansion by minimizing the functional

$$\min_{f \in \mathcal{H}} J(f) = \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i))^2 + c\|f\|_{\mathcal{H}}^2, \tag{2.37}$$

where $\mathcal{H}$ is the RKHS associated with the Mercer kernel κ and c is a regularization parameter. Kernel ridge regression exploits the universal approximation capability of kernel methods, which allows it to model the unknown mapping with any given accuracy.

The Representer Theorem 2.7 shows that each minimizer $f \in \mathcal{H}$ of (2.37) has the form

$$f(\cdot) = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \cdot), \tag{2.38}$$

where $\mathbf{x}_i$ are the training data points and α_i are the corresponding coefficients, for $i = 1, \ldots, N$. Therefore, by substituting (2.38) in (2.37), one obtains

$$\min_{\alpha} J(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_i) \right)^2 + c \sum_{i,j=1}^{N} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j). \tag{2.39}$$

After introducing the kernel matrix $\mathbf{K}$ and the coefficients vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_N]^T$, we can write (2.39) as

$$\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \|\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}\|^2 + c\boldsymbol{\alpha}^T \mathbf{K}\boldsymbol{\alpha}, \tag{2.40}$$

whose solution is

$$\boldsymbol{\alpha} = (\mathbf{K} + c\mathbf{I})^{-1} \mathbf{y}, \tag{2.41}$$

which is the dual solution obtained in (2.36).

Notice that the regularization term in (2.40) cannot simply be left out (i.e., $c = 0$) when working with kernels that have the universal approximation property, such as the Gaussian kernel. For these kernels, the dimensionality of the feature space is much higher than the number of observations, hence it is always possible to find a solution that perfectly fits the training data. By introducing regularization, the complexity of the solution will be limited, and as a result, it will generalize better to new data points. These concepts will be discussed later in more detail.

In Fig. 2.1 the case of one-dimensional kernel ridge regression is illustrated. The 25 data patterns (x_i, y_i) in this example, represented by white dots, were generated through to the model

$$y_i = \frac{\sin(\pi x_i)}{\pi x_i} + v_i, \tag{2.42}$$

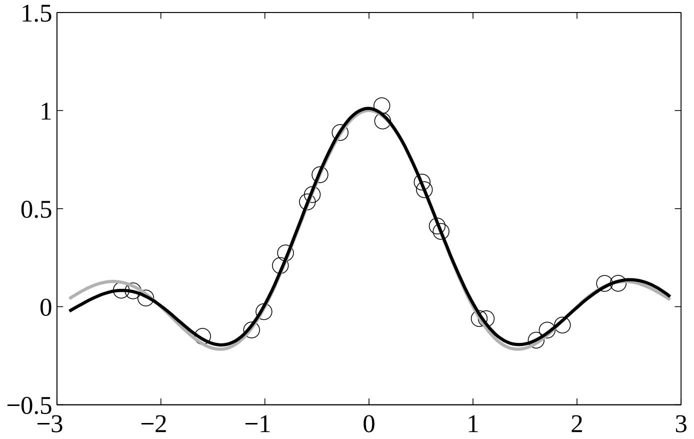


Figure 2.1: Example of one-dimensional kernel regression. The white dots represent the input data x_i versus the noisy labels $y_i + v_i$. The estimated kernel regressor function is shown as a solid black line, while the true underlying function is drawn as a solid grey line. The smoothness of the estimated function is imposed by regularizing the solution.

where v_i represents a small additive white Gaussian noise component. Once a regressor $\boldsymbol{\alpha}$ is obtained through (2.41), the image of any new point x can be found as

$$y = \mathbf{k}_x^T \boldsymbol{\alpha} = \tilde{\mathbf{x}}^T \tilde{\mathbf{X}} \boldsymbol{\alpha} = \sum_{i=1}^{N} \kappa(x_i, x)\alpha_i, \tag{2.43}$$

where $\tilde{\mathbf{X}}$ holds the transformed training data points, $\tilde{\mathbf{x}}$ is the transformed new test point, and $\mathbf{k}_x$ contains the kernels between the training data and the test point. This allows to trace the entire regressor function, which is depicted by the solid black line in Fig. 2.1. For this example a Gaussian kernel with $\sigma = 1$ and a regularization constant $c = 10^{-4}$ were used.

2.2.2 Kernel principal component analysis

Kernel principal component analysis (KPCA or kernel PCA) is a nonlinear generalization of principal component analysis (PCA), a classical technique in dimensionality reduction. Before addressing this “kernelized” version of PCA, we will first discuss some basic concepts of the linear technique.

Principal component analysis

Principal component analysis (PCA) [Diamantaras and Kung, 1996, Jolliffe, 2002] is an often used unsupervised technique in statistical data analysis, feature extraction and data compression. Given a set of measurements of a multidimensional random variable, the goal of PCA is to find a linear projection of this data onto a subspace such that the error between the original and the projected data is as small as possible in the least-square sense. Mathematically, this is equivalent to finding the best low rank approximation of the data via singular value decomposition (SVD).

Let us denote by $\mathbf{x}$ a zero-mean n-dimensional random variable. PCA will minimize the cost function

$$\underset{\mathbf{W},\mathbf{x}'}{\arg\min}\, J_{PCA} = E\left[\|\mathbf{x} - \mathbf{W}\mathbf{x}'\|^2\right] \quad \text{s.t. } \mathbf{w}_i^T\mathbf{w}_j = \delta_{ij}, \tag{2.44}$$

where $\mathbf{x}'$ represents the projection of $\mathbf{x}$ onto a subspace of dimension $m \leq n$, and the columns $\mathbf{w}_i$ of $\mathbf{W} \in \mathbb{R}^{n\times m}$ are the basis vectors of this subspace. The solution to this problem is found by diagonalizing the covariance matrix of $\mathbf{x}$. Given N observations $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$, an estimate of this covariance matrix can be obtained as

$$\mathbf{C} = \frac{1}{N}\mathbf{X}^T\mathbf{X}, \tag{2.45}$$

where the matrix $\mathbf{X} \in \mathbb{R}^{N\times n}$ contains the data stacked as rows. PCA is performed by solving the following eigenvalue problem

$$\frac{1}{N}\mathbf{X}^T\mathbf{X}\mathbf{v}_k = \lambda_k\mathbf{v}_k. \tag{2.46}$$

The solutions $\mathbf{v}_k$ are called *principal directions* of the data. They constitute the orthonormal basis vectors of the PCA transform. After projecting the data onto these directions, the *principal components* $\mathbf{u}_k$ are obtained, which are the coordinates of the data in the eigenvector basis:

$$\mathbf{u}_k = \frac{1}{N\sqrt{\lambda_k}}\mathbf{X}\mathbf{v}_k. \tag{2.47}$$

Often, a small number of principal components are sufficient to account for most of the variance in the data.

By combining the two previous equations, we find that the principal components $\mathbf{u}_k$ can also be obtained by solving another eigenvalue problem,

$$\frac{1}{N}\mathbf{X}\mathbf{X}^T\mathbf{u}_k = \lambda_k\mathbf{u}_k. \tag{2.48}$$

The matrix $\mathbf{X}\mathbf{X}^T \in \mathbb{R}^{N\times N}$ that appears here is the Gram matrix of the data. Interestingly, the same PCA information is contained in both eigenvalue problems (2.46) and (2.48), which are related through (2.47). Thanks to this relationship, the principal directions and the principal components of the PCA problem can be retrieved by solving any of the eigenvalue problems and taking into account (2.47). When choosing which of the eigenvalue problems to solve, one should take into account the different computational complexities involved with each of them. If the number of data points is much larger than the data dimensionality ($N \gg n$), it is recommended to solve the first eigenvalue problem (2.46) which has dimensions $n \times n$. This case is frequently encountered for instance in applications in communications. On the other

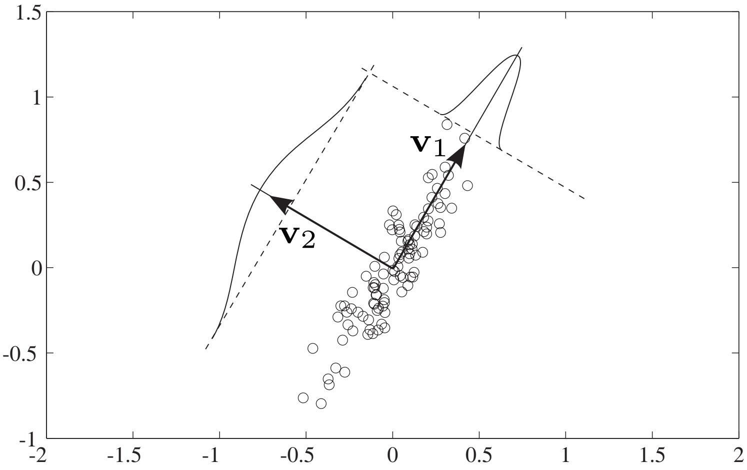


Figure 2.2: Example of PCA. For a given two-dimensional data set (white dots), PCA finds the principal directions $\mathbf{v}_1$ and $\mathbf{v}_2$. The first principal direction $\mathbf{v}_1$ indicates the direction of maximal data variance, as can be seen by the depicted probability density curves. Projecting the data onto this direction will yield the best possible linear one-dimensional representation in the MSE sense.

hand, when working with relatively few data points of very high dimension ($n \gg N$), solving the second problem (2.48) will create a smaller computational burden.

PCA obtains the optimal m-dimensional representation of the data by taking the m first principal components, corresponding to the m largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$. They have the following properties:

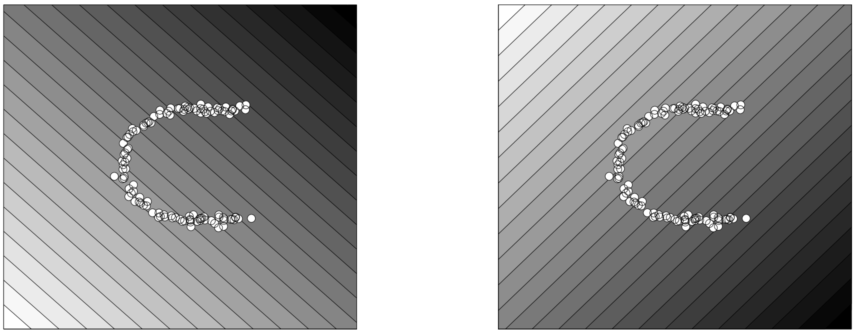
- The first m principal components contain more variance than any other m orthogonal directions.

- The mean-squared approximation error between the observations and the m principal components is minimal.

- The principal components are uncorrelated.

This analysis motivates the use of PCA in data compression and noise reduction applications.

An example of PCA on a simple two-dimensional data set is illustrated in Fig. 2.2. These data have a clear direction of maximal variance, which is identified correctly by PCA as the first eigenvector $\mathbf{v}_1$. If the data have a more complex, nonlinear structure, PCA will not be capable of finding an equally useful description of the data. This case is illustrated in Fig. 2.3, where the two eigendirections of a "U"-shaped data set are displayed. In this case we need an algorithm that is capable of extracting "nonlinear eigendirections".



(a) $\lambda_1 = 1$ (b) $\lambda_2 = 0.0318$

Figure 2.3: Linear PCA applied on a "U"-shaped data set. Lines of constant projection onto the first (a) and second (b) principal directions are shown, in which darker fill colors represent higher values. As can be seen, linear PCA is not able to identify the underlying nonlinear structure in the data.

Kernel PCA

Kernel principal component analysis is the procedure of performing PCA in feature space [Schölkopf et al., 1998]. First, this requires to map the data points $\mathbf{x}_i$ into feature space, obtaining $\tilde{\mathbf{x}}_i$. Assuming that the transformed points are centered in feature space[3], i.e. $\sum_{i=1}^{N} \tilde{\mathbf{x}}_i = \mathbf{0}$, the covariance matrix is

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T = \frac{1}{N} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}}. \tag{2.49}$$

However, this covariance matrix is an $n' \times n'$ matrix, where n' is the high, possible infinite dimensionality of the feature space. The standard PCA problem formulation in this space consists in solving

$$\mathbf{C}\tilde{\mathbf{v}} = \lambda \tilde{\mathbf{v}}, \tag{2.50}$$

which is now difficult or even impossible due to the high dimensionality of the used matrices. Fortunately, another way of obtaining the PCA information can be obtained by transforming this problem to its dual, Gram matrix-based formulation, as was mentioned in the description of the linear PCA problem. The dimensions of the Gram matrix (or kernel matrix) depend on the number of data points, N, and therefore as long as this number is reasonably low this seems to be a good strategy.

Let us redefine the PCA problem in terms of the kernel matrix. According to (2.49) we can write

$$\mathbf{C}\tilde{\mathbf{v}} = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{\mathbf{x}}_i^T \tilde{\mathbf{v}} \right) \tilde{\mathbf{x}}_i, \tag{2.51}$$

which implies that all solutions $\tilde{\mathbf{v}}$ must lie in the span of the transformed data $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_N$. Note that this is in accordance with the Representer Theorem. As

[3] In general the transformed data is not zero-mean in feature space. A procedure to obtain centered data is described in appendix C.

a consequence, there exist coefficients α_i such that

$$\tilde{\mathbf{v}} = \sum_{i=1}^{N} \alpha_i \tilde{\mathbf{x}}_i = \tilde{\mathbf{X}}^T \boldsymbol{\alpha}. \tag{2.52}$$

Substituting (2.52) and (2.49) in (2.50) yields

$$\frac{1}{N} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \boldsymbol{\alpha} = \lambda \tilde{\mathbf{X}}^T \boldsymbol{\alpha}. \tag{2.53}$$

Pre-multiplying this with the data matrix $\tilde{\mathbf{X}}$ yields

$$\frac{1}{N} \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \boldsymbol{\alpha} = \lambda \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T \boldsymbol{\alpha}, \tag{2.54}$$

which can be simplified by introducing the kernel matrix $\mathbf{K} = \tilde{\mathbf{X}} \tilde{\mathbf{X}}^T$, as

$$\frac{1}{N} \mathbf{K} \mathbf{K} \boldsymbol{\alpha} = \lambda \mathbf{K} \boldsymbol{\alpha}. \tag{2.55}$$

To find solutions of (2.55), we solve the eigenvalue problem

$$\frac{1}{N} \mathbf{K} \boldsymbol{\alpha} = \lambda \boldsymbol{\alpha}. \tag{2.56}$$

for nonzero eigenvalues. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ denote these eigenvalues, and $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_N$ the corresponding set of eigenvectors. To impose the unit norm of the eigenvectors in feature space, $\|\tilde{\mathbf{v}}_k\|^2 = 1$, we can see from (2.52) and (2.56) that the eigenvectors $\boldsymbol{\alpha}_k$ further need to be normalized as

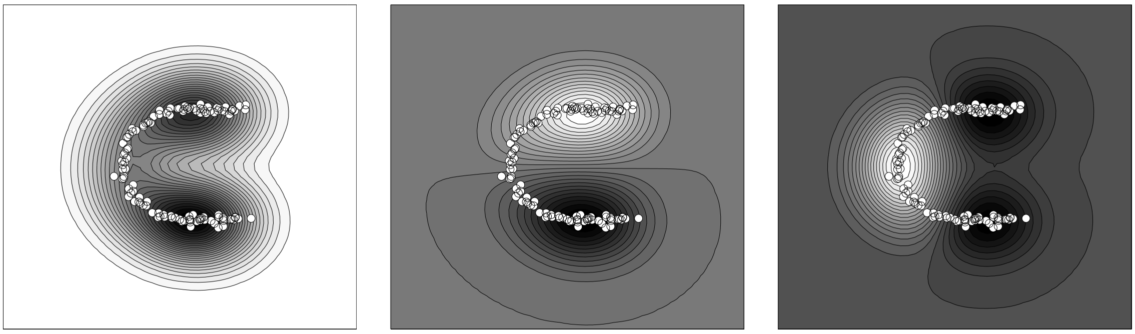
$$N \lambda_k \boldsymbol{\alpha}_k^T \boldsymbol{\alpha}_k = 1. \tag{2.57}$$

The entire KPCA procedure consists in solving the problem (2.56) and normalizing the obtained eigenvectors as in (2.57). Although these are simple algebraic operations, they allow to extract nonlinear features from the data.

Note that kernel PCA relies on solving the dual problem, i.e. instead of calculating the eigenvectors in feature space directly, it obtains their expansion in terms of the transformed data. Therefore, in order to obtain the principal components of a test point $\mathbf{x}$ it is necessary to calculate its projection in feature space onto the eigenvectors $\tilde{\mathbf{v}}_k$. In particular, we obtain

$$\tilde{\mathbf{x}}^T \tilde{\mathbf{v}}_k = \tilde{\mathbf{x}}^T \tilde{\mathbf{X}} \boldsymbol{\alpha}_k = \mathbf{k}_x^T \boldsymbol{\alpha}_k. \tag{2.58}$$

An example of the nonlinear feature extraction capabilities of kernel PCA is given in Fig. 2.4. Kernel PCA with a Gaussian kernel is performed here on the same "U"-shaped data set that was used in Fig. 2.3. It can be observed that kernel PCA successfully retrieves the nonlinear features present in the data. Moreover, since it implicitly operates in a high-dimensional feature space, the number of features that



(a) $\lambda_1 = 1$ (b) $\lambda_2 = 0.9768$ (c) $\lambda_3 = 0.7474$

Figure 2.4: Feature extraction with kernel PCA, performed on the data set from Fig. 2.3. The contours of constant projection on the first (a), second (b) and third (c) principal directions are plotted. Darker colors represent higher values, and the color at the border of the plots represents the zero-value. The first principal direction of KPCA correctly identifies the main nonlinear structure of the data, and the remaining eigenvectors can be used to provide further insight.

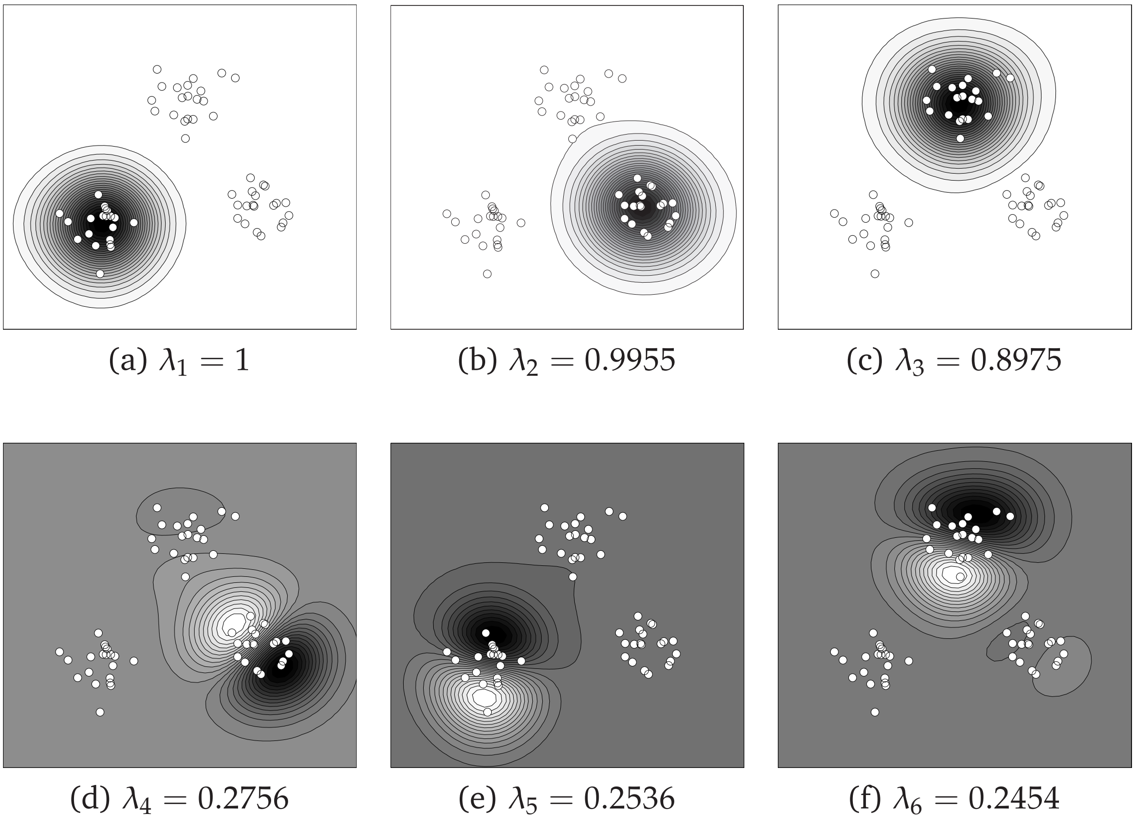
can be extracted is not limited to the input data dimension as in linear PCA. Instead, up to N features can be extracted.

Another example is given in Fig. 2.5, which illustrates the capabilities of kernel PCA to detect zones of high data density. As will be seen in the chapter 7, this property of kernel PCA can be exploited to construct clustering algorithms.

In general, kernel PCA can be employed to find directions in which a certain feature is maximally present in the data set. For a linear kernel, the directions of *maximal data variance* are detected, resulting in an algorithm that is equivalent to linear PCA. The Gaussian kernel, on the other hand, is well-known as the kernel used in Parzen window estimation for kernel density estimation (KDE), and therefore it should not come as a surprise that kernel PCA with the Gaussian kernel can capture “directions of maximal data density”. This idea is also reflected in [Girolami, 2002], where it was shown how KDE can be performed based only on the solutions of the kernel PCA eigenvector problem using a Gaussian kernel.

2.3 Regularization

Although the use of a high dimensional Hilbert space provides kernel methods with a very high degree of flexibility in solving minimization problems, it is the same flexibility that easily causes the solution to overfit to the training data. Specifically, if a sufficiently “rich” kernel function is used and no precautions are taken, the solution can have enough complexity to perfectly fit the given training data set while it will not generalize well to new, unseen data. Additionally, this can also lead to numerical instabilities.



(a) $\lambda_1 = 1$

(b) $\lambda_2 = 0.9955$

(c) $\lambda_3 = 0.8975$

(d) $\lambda_4 = 0.2756$

(e) $\lambda_5 = 0.2536$

(f) $\lambda_6 = 0.2454$

Figure 2.5: Kernel PCA applied on a three-cluster data set. The contour plots of constant projection on the first six principal directions are shown, along with the corresponding eigenvalues λ_i. A Gaussian kernel was used.

In order to prevent this overfitting, the solution should be regularized, which is commonly achieved by limiting its complexity. In regression, for instance, this can be interpreted as smoothing out the obtained functional representation. In this section we discuss three basic techniques to address overfitting.

2.3.1 L_2 regularization

The most common form of regularization is given in the Representer Theorem (2.20), which includes a *regularization function* Ω. A popular choice of regularization is to choose Ω such that it penalizes large expansion coefficients α_i, for instance by penalizing the L_2 norm of the solution as

$$\Omega[f] := \sum_{i=1}^{N} \alpha_i^2 = \|\boldsymbol{\alpha}\|^2. \tag{2.59}$$

This type of regularization, which was already touched upon in section 2.2.1, is known as ridge regression or quadratic regression, and was introduced by Tikhonov

[Tikhonov, 1963]. In the feature space, the entire minimization problem can be written as

$$\min_{\tilde{\mathbf{h}}} J(\tilde{\mathbf{h}}) = \|\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{h}}\|^2 + c\tilde{\mathbf{h}}^T\tilde{\mathbf{h}}. \tag{2.60}$$

Introducing the kernel matrix $\mathbf{K} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^T$ and taking into account that $\tilde{\mathbf{h}} = \tilde{\mathbf{X}}^T\boldsymbol{\alpha}$, it was shown in section 2.2.1 that the solution is found as

$$\boldsymbol{\alpha} = (\mathbf{K} + c\mathbf{I})^{-1}\,\mathbf{y}. \tag{2.61}$$

2.3.2 Sparsification

A second technique to lower the complexity is based on limiting the functional representation of the solution. According to the Representer Theorem, the solution can be represented as an expansion of all transformed input data points. Sparsification consists in discarding the less relevant input data points from this expansion. Apart from improving computational efficiency, it is also commonly known that a sparse model usually gives better generalization ability [Platt, 1991, Vapnik, 1995, Engel et al., 2004].

In order to select a compact subset of the training data points, different approaches have been proposed, including novelty criterion [Platt, 1991], approximate linear dependency (ALD) [Engel et al., 2004], the surprise information measure [Liu et al., 2010], and greedy algorithms [Cawley and Talbot, 2002]. The general idea of these methods is to construct a *dictionary* of points of interest, for which the kernel expansion and kernel matrix will be constructed. In most cases the dictionary can be kept reasonably small. Since these methods are closely related to building networks in online problem settings, they will be discussed in detail in chapter 3, which deal with online kernel methods.

2.3.3 Low-rank approximation

From the viewpoint of the feature space, the previous sparsification methods construct a subspace spanned by the transformed dictionary points. By constraining the solution to lie in this subspace, they inherently restrict the complexity of the solution. A different manner to obtain such a subspace is found by directly constructing a low-rank approximation of the transformed data, for instance by applying PCA in feature space [Schölkopf et al., 1998]. As was shown in the example of section 2.2.2, this consists in decomposing the kernel matrix into

$$\mathbf{K} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T, \tag{2.62}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times m}$ is a diagonal matrix containing the m principal eigenvalues and $\mathbf{U} \in \mathbb{R}^{N\times m}$ contains the corresponding eigenvectors. In practice there exist several techniques that are capable of computing this decomposition efficiently, even for large data sets, as will be shown in the next section.

2.4 Practical Considerations

Many kernel methods require computing the entire kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, whose dimensionality grows quadratically with the number of data points N. Furthermore, a wide range of operations on this matrix require up to $O(N^3)$ computations, for instance the computation of its inverse or SVD decomposition. As a result, the complexity of kernel methods can become prohibitive in problems where a large set of training data is involved. In this section we discuss a few techniques that allow kernel methods to scale to such problems.

A first category of kernel algorithms that can deal with large-scale problems is the family of iterative optimization methods. A few examples of these techniques include *sequential minimization* [Platt, 1999], which breaks up the large minimization problem into smaller sub-problems, and *sparse approximation* techniques, which iteratively choose random subsets of the data to approximate the solution [Vincent and Bengio, 2002]. Furthermore, in chapter 3 we will discuss *online kernel methods*, which achieve low complexity by iteratively updating the solution as data becomes available.

A second set of kernel methods for large-scale problems focus on approximating the kernel matrix. This is usually achieved by finding a suitable low-rank decomposition that satisfies

$$\mathbf{K} \approx \mathbf{U}\mathbf{U}^T, \tag{2.63}$$

where $\mathbf{U} \in \mathbb{R}^{N \times m}$ is a “tall” but “narrow” matrix of rank m. For most common kernels, the eigenspectrum of the kernel matrix declines rapidly and therefore an efficient decomposition can be obtained with $m \ll N$. Note that (2.63) is related to the kernel PCA decomposition. Specifically, these methods can be employed to perform kernel PCA efficiently on large data sets, as shown in appendix B.3. Here, we limit the discussion to an overview of the most common kernel matrix decomposition techniques:

- **Incomplete Cholesky decomposition** (ICD) [Bach and Jordan, 2002, Kulis et al., 2006, Bach and Jordan, 2005] obtains a dimensionality reduction based on an efficient pivot-based scheme, that allows to compute the decomposition without accessing all elements of $\mathbf{K}$ [4].

- **Nyström approximation** finds a numerical approximation to the eigendecomposition of the kernel matrix from randomly chosen rows and columns. It is based on the Nyström method [Williams and Seeger, 2000].

- **Sparse greedy matrix approximation** constructs an approximation of the kernel by consecutively adding columns to its estimate of $\mathbf{U}$ [Schölkopf and Smola, 2002]. Although it is similar to ICD, it includes a random factor in its selection procedure, and it has a slightly higher complexity than the two previous techniques.

[4] In appendix B an efficient way of calculating KPCA based on this form of ICD is discussed, which is used throughout this work to alleviate the computational burden of KPCA.

Low-rank approximation methods generally have time complexity $O(Nm^2)$ and memory complexity $O(Nm)$. Since the rank of the approximation, m, depends on the desired accuracy of approximation, ϵ, these methods usually allow to fix either m or ϵ.

A different approach to approximate the kernel matrix is followed by the *empirical feature map* [Xiong et al., 2005, Abe and Shigeo, 2007]. This method constructs an "empirical feature space", which is a space spanned by the mapped training data that gives approximately the same kernel value as that of the original feature space.

2.5 Choice of the Kernel Function

A problem inherent to every kernel method is the question of what kernel should be used. For most kernels, this includes choosing one or more suitable kernel parameters. Ideally, the kernel should reflect as much prior knowledge about the problem as possible. By interpreting kernels as similarity measures between data points, the choice of a kernel is often pointed out by the nature of the problem. For instance, if data density is important, the Gaussian kernel will be a good starting point.

2.5.1 Kernel design

The design of kernels is a very active topic in the machine learning community, which has led to a large number of techniques to construct positive definite kernels, or even *learn* them. The construction of positive definite kernels can be as simple as combining several different positive definite kernels. Examples of such kernels are the linear combination of kernels with nonnegative coefficients, and the product of two kernels. In part III of this thesis we will design kernels to include information about the local structure of the data for specific applications. For an overview of kernel design methods we refer to [Schölkopf and Smola, 2002]. More recently, the concept of hyperkernels was introduced [Tsang and Kwok, 2004, Ong et al., 2005], which define an RKHS on the space of kernels itself in order to learn a suitable kernel function.

2.5.2 Kernel parameters

A number of techniques have been proposed to automatically select the kernel parameter, including rules-of-thumb as well as more theoretically founded approaches. Common approaches include cross-validation and Silverman's rule for the Gaussian kernel [Silverman, 1986], which reads

$$\sigma = 0.9AN^{-1/5}, \tag{2.64}$$

where N is the number of data points and $A = \min(d, \frac{q_3 - q_1}{1.34})$ is the minimum of the empirical standard deviation d of the data and the data interquartile range scaled by 1.34.

2.6 Conclusions

In this chapter we provided the theoretical background for kernel methods and we laid out a number of related concepts, such as regularization, implementation issues and the design of the kernel itself. The examples provided in this chapter will be used as building blocks in the following chapters, in order to construct more elaborate algorithms.

Chapter 3
Online Kernel Methods

Online kernel methods are kernel-based algorithms capable of operating in online settings where the data arrive sequentially, while maintaining their computational load moderate during each iteration. In general, online kernel methods aim to sequentially uncover a function f that maps a series of input patterns $\mathbf{x}_n$ onto their respective desired images d_n. Similar to other online methods, they are useful in situations where the amount of data is too high to apply batch algorithms and the solution has to be updated sequentially to account for all processed data. A second scenario that requires online updating of the solution occurs in dynamic environments, where the solution changes over time.

Typically, online learning consists of two basic steps, that are repeated at each time instant n. First, the online algorithm receives an observation $\mathbf{x}_n$ for which it calculates the estimated image y_n, based on its current estimate of f. Next, the algorithm receives the desired image d_n, which allows it to calculate the estimation error $e_n = d_n - y_n$ and update its estimate of f.

Online kernel methods are often based on linear adaptive filtering techniques that are taken into feature space. The design of these algorithms requires to deal with challenges such as overfitting and computational complexity issues. In this chapter, we will focus on the most important online kernel methods, which are direct implementations of linear adaptive filtering techniques in feature space. We give a brief overview of some classical adaptive filtering algorithms, and discuss several issues that emerge when formulating these algorithms in feature space.

3.1 Adaptive Filtering in the Input Space

In this section we review some basic concepts of linear adaptive filtering theory. We will denote the input to a filter on time instant n as x_n, its output as y_n, and the desired output as d_n. The input signal x_n is assumed to be zero-mean, and to allow for a compact notation we will denote the *time-delay vector* of L taps of this signal on time instant n as $\mathbf{x}_n = [x_n, x_{n-1}, \dots, x_{n-L+1}]^T$.

3.1.1 Linear FIR filtering

Consider a zero-mean time series $\{x_1, \dots, x_N\}$ used as the input for a linear filter of length[1] L that is characterized by the impulse response $\mathbf{w} = [w_0, w_1, \dots, w_{L-1}]^T$. The output of this filter on time instant n is

$$y_n = \sum_{i=0}^{L-1} w_i x_{n-i} = \mathbf{x}_n^T \mathbf{w}, \tag{3.1}$$

for $n = 1, 2, \dots, N$. In a batch problem setting, the entire sequence of input-output patterns $\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots, \ (\mathbf{x}_N, d_N)\}$ is used to calculate the linear filter $\mathbf{w}$. By denoting the *estimation error* as $e_n = y_n - d_n$, the goal of optimal linear filtering is to minimize the mean-square error (MSE) cost

$$J(\mathbf{w}) = E[e_n^2]. \tag{3.2}$$

This cost function can be calculated in terms of the available data as

$$J(\mathbf{w}) = \sigma_d^2 + \mathbf{w}^T \mathbf{R}_x \mathbf{w} - 2\mathbf{w}^T \mathbf{p}, \tag{3.3}$$

where σ_d^2 is the variance of the desired signal, $\mathbf{R}_x$ is the input data covariance matrix $E\left[\mathbf{x}_n \mathbf{x}_n^T\right]$, $\mathbf{p}$ is the cross-covariance between the input and the desired signal $E\left[d_n \mathbf{x}_n\right]$. The gradient of (3.3) is given by

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = 2\mathbf{R}_x \mathbf{w} - 2\mathbf{p}. \tag{3.4}$$

Setting this gradient to zero leads to the well-known minimum mean-square error (MMSE) Wiener filter solution

$$\mathbf{w}_{opt} = \mathbf{R}_x^{-1} \mathbf{p}. \tag{3.5}$$

This filtering problem can also be solved iteratively by applying the steepest descent method. Given $\mathbf{R}_x$ and $\mathbf{p}$, the update equation of the steepest descent method reads

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \frac{\mu}{2} \nabla_{\mathbf{w}} J(\mathbf{w})|_{\mathbf{w}_n} = \mathbf{w}_n + \mu \left(\mathbf{p} - \mathbf{R}_x \mathbf{w}_n\right), \tag{3.6}$$

where μ is the *step-size* of the algorithm. Repeatedly applying this filter update will converge to the Wiener filter solution if a small enough step-size is chosen.

3.1.2 Least mean square algorithm

The LMS algorithm is a classical stochastic gradient algorithm often used in adaptive filtering. It was proposed by Widrow and Hoff in 1960 [Widrow and Hoff, 1960, Widrow et al., 1975] and it is widely used as a reference algorithm due to its simplicity and robustness.

[1] In this thesis, we denote the *length* of linear filters as their number of taps L. The corresponding filter *order* is $L - 1$.

Algorithm 3.1 Least Mean Square (LMS)

```
initialize
  $\mathbf{w}_0 = \mathbf{0}$.
for $n = 1, 2, \ldots$ do
  $y_n = \mathbf{x}_n^T \mathbf{w}_{n-1}$.
  $e_n = d_n - y_n$.
  $\mathbf{w}_n = \mathbf{w}_{n-1} + \mu \mathbf{x}_n e_n$.
end for
Output $\mathbf{w}_n$.
```

The LMS algorithm aims to solve the MSE filtering problem (3.2) in an online manner, updating its solution one step at a time as a new data pattern $(\mathbf{x}_n, d_n)$ becomes available. In practice, $\mathbf{R}_x$ and $\mathbf{p}$ are seldom known, and therefore the steepest descent method (3.6) cannot be readily applied. To estimate these statistics, the LMS algorithm uses the simplest choice: the *instantaneous estimates*, based on sample values of the input and desired signal

$$\hat{\mathbf{R}}_x = \mathbf{x}_n \mathbf{x}_n^T \tag{3.7}$$

$$\hat{\mathbf{p}} = d_n \mathbf{x}_n. \tag{3.8}$$

After substituting these estimates in the steepest-descent method (3.6), the well-known update rule of the LMS filter is obtained:

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \mathbf{x}_n \left(d_n - \mathbf{x}_n^T \mathbf{w}_n \right) = \mathbf{w}_n + \mu \mathbf{x}_n e_n, \tag{3.9}$$

where e_n is the *a-priori error* and $\mathbf{x}_n e_n$ represents an instantaneous estimate of the gradient.

The resulting LMS algorithm is summarized in Alg. 3.1. It is very fast, yielding a time complexity of $O(L)$. Moreover, thanks to its instantaneous estimate of the gradient, it can perform tracking of non-stationary systems. On the downside, its main disadvantage is that its instantaneous estimate of the gradient is noisy. As a consequence, it never converges in static environments, but rather oscillates around a solution with MSE J_∞. The misadjustment of this solution, compared with the optimal Wiener solution $J_{opt} < J_\infty$, depends on the step-size μ: A lower step-size will lead to a lower misadjustment, but it will also slow down the convergence.

Notice, with the application of kernel methods and the kernel trick in mind, that all operations of the LMS algorithm can be expressed solely in terms of inner products. The repeated application of the weight-update equation yields

$$\mathbf{w}_n = \mu \sum_{i=1}^{n} e_i \mathbf{x}_i, \tag{3.10}$$

which allows to write the filter output at instant n for a given input $\mathbf{x}$ as

$$y = \sum_{i=1}^{n} e_i \left(\mathbf{x}_i^T \mathbf{x} \right). \tag{3.11}$$

Meanwhile, the error e_i is obtained in terms of inner products as

$$e_n = d_n - \mu \sum_{i=1}^{n-1} e_i \left(\mathbf{x}_i^T \mathbf{x}_i \right). \tag{3.12}$$

A large number of modifications to the original LMS algorithm have been proposed to improve its different aspects [Sayed, 2003]. Convergence, for instance, can be improved by regularizing the squared norm of the solution. The resulting *leaky LMS* algorithm aims to minimize the cost function

$$\min_{\mathbf{w}_n} J(\mathbf{w}_n) = E\left[e_n^2\right] + c\|\mathbf{w}_n\|^2, \tag{3.13}$$

where c is a regularization constant. This translates into the instantaneous update rule

$$\mathbf{w}_{n+1} = (1 - \mu c)\mathbf{w}_n + \mu \mathbf{x}_n e_n. \tag{3.14}$$

3.1.3 Recursive least-squares algorithm

A comprehensive introduction to the method of *least squares* (LS) can be found in [Haykin, 2001]. Some of the basic concepts of least squares were also mentioned in section 2.2.1, which dealt with regression. As the emphasis in this chapter will be on deriving kernel algorithms, we will only require a description of the basic concepts of least squares.

In the method of least-squares, the tap weights of the filter $\mathbf{w}$ are chosen such as to minimize the cost function

$$\min_{\mathbf{w}} J_{LS}(\mathbf{w}) = \sum_{n=1}^{N} |e_n|^2 = \sum_{n=1}^{N} |d_n - \mathbf{x}_n^T \mathbf{w}|^2. \tag{3.15}$$

This method is deterministic in approach, in that it depends specifically on the number of data samples used in the computation. A regularization term can be included in the cost function, resulting in

$$\min_{\mathbf{w}} J_{LS}(\mathbf{w}) = \sum_{n=1}^{N} |d_n - \mathbf{x}_n^T \mathbf{w}|^2 + c\|\mathbf{w}\|^2, \tag{3.16}$$

whose solution is found as

$$\hat{\mathbf{w}} = \mathbf{X}^\dagger \mathbf{y} = \left(\mathbf{X}^T \mathbf{X} + c\mathbf{I} \right)^{-1} \mathbf{X}^T \mathbf{y}. \tag{3.17}$$

Given the least-squares estimate of the filter $\mathbf{w}_n$ at iteration n, the recursive least-squares (RLS) algorithm deals with the problem of updating this estimate when a new data point $\mathbf{x}_{n+1}$ and desired response d_{n+1} become available [Haykin, 2001]. Apart from a regularization term, the cost function of RLS also commonly includes

Algorithm 3.2 Exponentially-Weighted Recursive Least-Squares (RLS)

```
initialize
  $\mathbf{w}_0 = \mathbf{0}$.
  $\mathbf{P}_0 = c^{-1}\mathbf{I}$.
for $n = 1, 2, \ldots$ do
  $r_n = 1 + \lambda^{-1}\mathbf{x}_n^T\mathbf{P}_{n-1}\mathbf{x}_n$.
  $\mathbf{k}_n = \lambda^{-1}\mathbf{P}_{n-1}\mathbf{x}_n/r_n$.
  $e_n = d_n - \mathbf{x}_n^T\mathbf{w}_{n-1}$.
  $\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{k}_n e_n$.
  $\mathbf{P}_n = \lambda^{-1}\mathbf{P}_{n-1} - \lambda^{-1}\mathbf{P}_{n-1}\mathbf{x}_n\mathbf{x}_n^T\mathbf{P}_{n-1}/r_n$.
end for
Output $\mathbf{w}_n$.
```

a weighting factor, for instance an exponential weighting that limits the contribution of older data points. The resulting exponentially-weighted cost function is

$$\min_{\mathbf{w}_n} J_{RLS}(\mathbf{w}_n) = \sum_{i=1}^{n} \left(\lambda^{n-i} \left| d_i - \mathbf{x}_i^T \mathbf{w}_n \right|^2 + c\lambda^n \|\mathbf{w}_n\|^2 \right), \tag{3.18}$$

where $0 < \lambda \leq 1$ is called the *forgetting factor*. The exponentially weighted RLS algorithm aims to minimize this cost function in a recursive manner: Instead of determining afresh the LS solution when a new data point becomes available, it updates the solution in an efficient manner. In particular, in order to avoid the costly inversion of the covariance matrix, the RLS algorithm makes use of the matrix inversion lemma (see Lemma 3.1), which allows to calculate the inverse matrix in a recursive manner. Given the covariance matrix at iteration n, $\mathbf{R}_n$, and its inverse, $\mathbf{R}_n^{-1}$, it obtains the inverse of the next covariance matrix $\mathbf{R}_{n+1}^{-1} = \frac{1}{n+1}(n\mathbf{R}_n + \mathbf{x}_{n+1}\mathbf{x}_{n+1}^T)^{-1}$ in $O(L^2)$ operations.

Lemma 3.1 (Matrix Inversion Lemma)**.** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two positive-definite matrices of size* $L \times L$ *that satisfy*

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T, \tag{3.19}$$

where $\mathbf{D}$ *is a positive-definite* $M \times M$ *matrix and* $\mathbf{C}$ *is an* $L \times M$ *matrix. The inverse of matrix* $\mathbf{A}$ *can be expressed as*

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{B}. \tag{3.20}$$

This relationship is also known in the literature as *Woodbury's formula* or the *Sherman-Morrison-Woodbury* formula [Golub and Van Loan, 1996, Hager, 1989].

The entire exponentially-weighted RLS algorithm is summarized in Alg. 3.2. Here, $\mathbf{P}_n$ can be interpreted as the inverse of the regularized data covariance matrix, $\mathbf{k}_n$ is called the gain vector and e_n is the prediction error. For $\lambda = 1$, the forgetting factor has no influence, and assuming that the signals are ergodic the algorithm will converge towards the solution of the regularized LS problem (3.17).

3.2 Adaptive Filtering in RKHS

In recent years there have been some efforts in the literature to “kernelize" adaptive filters. The resulting algorithms combine the adaptive characteristics of traditional linear adaptive filters with the capabilities of kernel methods to resolve nonlinear problems by means of a convex learning process with no local minima. Although there have been numerous “adaptive” methods that use all training data, such as the deterministic gradient method of kernel ADALINE [Frieß and Harrison, 1999], we are more interested in algorithms that are capable of operating truly *online*, in the sense that they allow to sequentially receive new data while training.

Before providing detailed algorithm descriptions in sections 3.3 and 3.4, we first discuss the most common bottlenecks in the design of online kernel methods, which are similar to the issues encountered by block-based (batch) kernel methods (see section 2.3).

Computational complexity Traditionally, algorithms based on kernel methods were not capable of operating online since their representation of the functional mapping f becomes more complex as the number of observations increases [Kivinen et al., 2004]. This is a consequence of the Representer Theorem [Kimeldorf and Wahba, 1971, Schölkopf et al., 2001], which states that each minimizer $f \in \mathcal{H}$ of the regularized LS cost function (2.37) can be expressed as a kernel expansion

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{x}_i, \cdot), \tag{3.21}$$

where $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is a set of stored examples called *basis vectors*, and $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ are the stored coefficients. Without any measure taken, the number of required kernel functions grows linearly with the number of observations. This also causes the kernel matrix to grow without bound, resulting in a superlinearly computational complexity. Therefore, the main challenge in designing an online kernel method lies in limiting the growth of this functional representation [Platt, 1991, Vapnik, 1995, Engel et al., 2004].

Overfitting Additionally, online kernel methods suffer from overfitting, which is inherent to kernel methods due to the high dimensionality of the induced Hilbert space. Therefore some form of regularization is needed. Below we discuss the different options to apply regularization and to limit the computational complexity in online kernel methods. Notice that some methods address both problems at the same time.

3.2.1 Online regularization

To avoid overfitting, the complexity of the solution f can be penalized by limiting its L_2-norm (see section 2.3). The resulting minimization problem becomes

[Evgeniou et al., 2000]

$$\min_{f \in \mathcal{H}} J(f) = \sum_{i=1}^{n} L\left(d_i, f(\mathbf{x}_i)\right) + c\|f\|_{\mathcal{H}}^2, \tag{3.22}$$

where $L(\cdot, \cdot)$ is a loss function, $\mathcal{H}$ is the RKHS associated with the Mercer kernel $\kappa(\cdot, \cdot)$ and c is a regularization parameter.

3.2.2 Online sparsification

The general idea behind sparsification methods is to construct a sparse *dictionary* of points of interest, for which the kernel matrix will be constructed. These points should allow to represent the remaining points in a fairly simple way, for instance as a linear combination in feature space. As a general rule in learning theory, it is desirable to design a network with as few processing elements as possible. Sparsity reduces the complexity in terms of computation and memory, and it usually gives better generalization ability to unseen data [Platt, 1991, Vapnik, 1995]. Apart from avoiding overfitting, sparsification procedures are also capable of limiting the computational complexity of online algorithms: Since their dictionary can usually be kept reasonably small, the growth of the kernel matrix is restricted.

Suppose the dictionary at time instant $n-1$ is $\mathcal{D}_{n-1} = \{\mathbf{c}_i\}_{i=1}^{m_{n-1}}$, where $\mathbf{c}_i$ is the i-th stored center and m_{n-1} is the cardinality at this instant. When a new input-output pair $(\mathbf{x}_n, d_n)$ is presented, a decision is made whether or not $\mathbf{x}_n$ should be added to the dictionary as a center. We shortly discuss two possible criteria to construct such dictionaries.

Novelty Criterion The novelty criterion is an early data selection method, introduced by Platt [Platt, 1991]. It is used to construct *resource allocating networks* (RAN), which are essentially growing radial basis function networks. When a new data point $\mathbf{x}_n$ is obtained by the network, the novelty criterion calculates the distance of this point to the present dictionary. If this distance is smaller than some preset threshold, $\mathbf{x}_n$ will not be added to the dictionary. Otherwise, it computes the prediction error, and only if this error is larger than another preset threshold, $\mathbf{x}_n$ will be accepted as a new center.

Approximate Linear Dependency Criterion The approximate linear dependency (ALD) test introduced in [Engel et al., 2004] requires some more computation. When a new data pair $\{\mathbf{x}_n, y_n\}$ is presented, the algorithm tests whether the transformed input point $\phi(\mathbf{x}_n)$ is *approximately linearly dependent* on the dictionary vectors, by calculating the residual error

$$\delta_n = \min_{\mathbf{a}} \left\| \sum_{i=1}^{m_{n-1}} a_i \phi(\mathbf{c}_i) - \phi(\mathbf{x}_n) \right\|^2, \tag{3.23}$$

where $\mathbf{a} = [a_1, \ldots, a_{m_{n-1}}]^T$ is a vector containing the expansion coefficients of the linear combination. If δ_n does not exceed a certain threshold ν, the new data point $\mathbf{x}_n$ can be approximated sufficiently well in feature space by a linear combination of the centers stored in the current dictionary. On the other hand, if $\delta_n > \nu$, the current dictionary does not represent the new data point sufficiently well and it must be expanded. In this case, a new center $\mathbf{c}_n = \mathbf{x}_n$ is added to the dictionary, yielding $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{c}_n\}$ and $m_n = m_{n-1} + 1$. ALD-based approaches generally have complexity $O(m^2)$ at each time step, where m is the dictionary size.

A number of alternative methods have been proposed to achieve sparseness by creating a basis dictionary and storing the corresponding coefficients. A Bayesian framework for active data selection in online kernel methods was addressed in [Liu et al., 2010]. By measuring the information a new data point can contribute to a learning system, this criterion is capable of determining the data points that “surprise” the system and should therefore be stored in its memory. A more straightforward approach with a fixed-size memory was followed in the sliding-window and fixed-budget kernel methods from [Van Vaerenbergh et al., 2006b, Van Vaerenbergh et al., 2010b], which not only add points to the dictionary but also prune unnecessary points. These concepts will be discussed in detail in chapter 4.

3.2.3 Online low-rank approximation

As mentioned in section 2.3, imposing sparsification can be interpreted as constraining the solution to lie in a subspace of the feature space. While dictionary-based methods achieve this by selecting a reduced set of input data points, this constraint can also be accomplished by directly seeking an optimal subspace of the feature space, for instance by applying kernel PCA. This will both lower the computational complexity, since it reduces the dimensionality of the involved matrices, and avoid overfitting, since the solution is restricted to lie in this lower-dimensional subspace.

Online kernel PCA In [Hoegaerts et al., 2007] an algorithm was proposed that allows to track the kernel eigenspace dynamically, making it possible to perform online kernel PCA. Unlike iterative PCA approaches that recursively calculate the PCA solution for a given data set, this method is truly online in that it accepts data points sequentially and updates the KPCA solution in every iteration. Specifically, it updates the eigenvectors and eigenvalues of the kernel matrix as new data points are added to the problem. It has the possibility to exclude the influence of older observations in a sliding-window fashion by only considering the last N observations, which makes it suitable for time-varying problem settings. This implementation has time and memory complexity $O(NM^2)$, where N is the number of data points in the observed window and M is the number of eigenvectors used to span the subspace.

3.3 Least Mean Squares Techniques in RKHS

Recently, several extensions of LMS into feature space have been proposed. Specifically, Kivinen et al. proposed an algorithm called NORMA (Naive Online regularized Risk Minimization Algorithm) that directly differentiates the regularized functional to get the stochastic gradient [Kivinen et al., 2004], which is equivalent to a kernel version of the leaky least mean square algorithm (3.14). On the other hand, Liu et al. presented a related technique that does not use explicit regularization [Liu and Príncipe, 2008].

3.3.1 NORMA

Kivinen et al. proposed an algorithm to perform stochastic gradient descent in reproducing kernel Hilbert space [Kivinen et al., 2004]. In order to learn a mapping f, one can solve the following *empirical risk* minimization problem [Schölkopf and Smola, 2002]

$$\min_{f\in\mathcal{H}} J(f) = \sum_{i=1}^{n} L\left(d_i, f(\mathbf{x}_i)\right), \tag{3.24}$$

where $L(\cdot)$ is a suitable loss function. To avoid overfitting, NORMA penalizes the norm of the mapping by introducing a regularization term, leading to the *regularized risk* minimization problem

$$\min_{f\in\mathcal{H}} J(f) = \sum_{i=1}^{n} L\left(d_i, f(\mathbf{x}_i)\right) + c\|f\|^2_{\mathcal{H}}, \tag{3.25}$$

which is the expression presented in (3.22). Inspired by the LMS algorithm, an online approach was designed to find a solution by minimizing an *instantaneous regularized risk*. In practice, LMS can be performed in feature space by replacing the inner products in the LMS update equations (3.10), (3.11) and (3.12) by kernels. The resulting algorithm, NORMA (Naive Online regularized Risk Minimization Algorithm), is equivalent to a kernel version of leaky LMS.

By representing the estimate of the nonlinear mapping f as

$$f_n(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}), \tag{3.26}$$

the update rule of the coefficients α_i in NORMA is as follows

$$\alpha_i = \begin{cases} -\mu L'\left(y_i, d_i)\right), & \text{for } i = n \\ (1-\mu c)\alpha_i, & \text{for } i < n, \end{cases}$$

where L' represents the derivative of the loss function with respect to f. A major drawback to this update scheme is that the kernel expansion will contain n terms at instant n. In NORMA, this is solved by truncating the kernel expansion: since at

Algorithm 3.3 Naive Online regularized Risk Minimization Algorithm (NORMA)

```
initialize
    $y_1 = 0.$
    $\alpha_1 = -\mu L'(y_1, d_1).$
    $\beta_i = (1 - \mu c)^i$, for $i = 0, \dots, \tau.$
for $n = 2, 3, \dots$ do
    $y_n = \sum_{i=\max_{1,n-\tau}}^{n-1} \alpha_i \beta_{n-i-1} \kappa(\mathbf{x}_i, \mathbf{x}_n).$
    $\alpha_n = -\mu L'(y_n, d_n).$
end for
Output $\alpha_i$, for $i = \max(1, n - \tau), \dots, n.$
```

each instant n, the coefficients α_i with $i < n$ are scaled by $(1 - \mu c)$, the smallest terms can be dropped without incurring significant error. In practice, it was shown that for a given truncation parameter $\tau \in \mathbb{N}$, the truncation error by dropping all terms that are at least τ steps old is bounded. This truncation scheme fundamentally converts NORMA into a *sliding-window* kernel LMS algorithm (see Fig. 1 in [Kivinen et al., 2004]). The entire algorithm is summarized in Alg. 3.3.

3.3.2 Kernel least mean square algorithm

Recently, Liu et al. showed that the LMS algorithm can be well-posed in RKHS without the need of an extra regularization term in the finite training data case [Liu and Príncipe, 2008], because the solution is always forced to lie in the subspace spanned by the input data. The proposed algorithm was named *kernel least-mean squares algorithm* (KLMS).

The lack of an explicit regularization term leads to two important advantages. First of all, it has a simpler implementation than NORMA, as the update equations are straightforward kernel versions of (3.10), (3.11) and (3.12). Second, it can potentially provide better results because regularization biases the optimal solution.

In particular, it was shown that a small enough step-size can provide a sufficient "self-regularization" mechanism. Moreover, since the space spanned by $\{\phi(\mathbf{x}_i)\}_{i=1}^N$ is possibly infinite-dimensional, the projection error of the desired signal d_n could be very small, as is well known from Cover's theorem [Haykin, 1999]. On the downside, the speed of convergence and the misadjustment also depend upon the step-size. As a consequence, they conflict with the generalization ability.

In online scenarios where data is continuously being received, the size of the KLMS network will continuously grow, posing implementation challenges. Therefore a sparsification technique such as ALD can be applied. A method involving Gaussian elimination steps on the KLMS gram matrix was recently proposed in [Pokharel et al., 2009]. In Alg. 3.4 the basic kernel LMS algorithm without sparsification is summarized.

Algorithm 3.4 Kernel Least Mean Squares (KLMS)

```
initialize
  $y_1 = 0.$
  $\alpha_1 = -\mu L'(y_1, d_1).$
for $n = 2, 3, \ldots$ do
  $y_n = \sum_{i=1}^{n-1} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}_n).$
  $\alpha_n = -\mu L'(y_n, d_n).$
end for
Output $\alpha_i$, for $i = 1, \ldots, n$.
```

3.4 Recursive Least-Squares Techniques in RKHS

Different types of kernel-based RLS algorithms have been proposed in the past few years. Engel et al. presented a kernel recursive least-squares (KRLS) algorithm in [Engel et al., 2004] that was a straightforward implementation of the RLS algorithm in feature space. To limit the size of the kernel matrix it applies an ALD sparsification procedure. In [Van Vaerenbergh et al., 2006b] we presented a sliding-window based kernel RLS approach (SW-KRLS) that uses an updating procedure to keep the dimensions of the kernel matrix fixed. Recently, Liu et al. proposed a kernel version of the extended RLS algorithm [Liu and Príncipe, 2008] (EX-KRLS), and we presented an extension of the sliding-window approach to a more general fixed-budget (FB-KRLS) principle [Van Vaerenbergh et al., 2010b]. To better distinguish our own contributions, we will discuss only the ALD-based KRLS and EX-KRLS algorithms in this preliminary chapter, leaving the SW-KRLS and FB-KRLS algorithms for chapter 4 in part II of this thesis.

3.4.1 Kernel recursive least-squares algorithm

Kernel recursive least squares (KRLS or kernel RLS) is the technique of performing standard RLS in feature space. At every time instant n, kernel RLS aims to minimize the LS cost function

$$\min_{\boldsymbol{\alpha}_n} J = \|\mathbf{K}_n \boldsymbol{\alpha}_n - \mathbf{d}_n\|^2, \tag{3.27}$$

where $\mathbf{K}_n$ is the kernel matrix of all seen data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{K}_n = \tilde{\mathbf{X}}_n \tilde{\mathbf{X}}_n^T$, and $\mathbf{d}_n$ is a vector containing all seen desired outputs $\mathbf{d}_n = [d_1, \ldots, d_n]^T$. To simplify the notation, we will leave out the index n in the following.

The theoretical solution, $\boldsymbol{\alpha} = \mathbf{K}^\dagger \mathbf{d}$, suffers from a number of problems including overfitting and growing computational complexity. Engel et al. proposed a sequential sparsification procedure based on an approximate linear dependency (ALD) criterion to avoid these problems [Engel et al., 2004]. Specifically, it approximates this solution by using a reduced kernel matrix $\check{\mathbf{K}} \in \mathbb{R}^{M \times M}$, obtained only from the points stored in its dictionary $\mathcal{D}$. By denoting $\tilde{\mathbf{C}}$ as the data matrix containing the transformed dictionary centers, the original matrix of transformed data can be approximated as

$$\tilde{\mathbf{X}} \approx \mathbf{A}\tilde{\mathbf{C}}, \tag{3.28}$$

where $\mathbf{A} \in \mathbb{R}^{n \times M}$ contains the coefficients of the approximate linear combinations. This allows to approximate the original kernel matrix as

$$\mathbf{K} \approx \mathbf{A}\breve{\mathbf{K}}\mathbf{A}^T, \tag{3.29}$$

and the cost function (3.27) can be rewritten as

$$\min_{\breve{\boldsymbol{\alpha}}} J = \|\mathbf{A}\breve{\mathbf{K}}\breve{\boldsymbol{\alpha}} - \mathbf{d}\|^2, \tag{3.30}$$

where $\breve{\boldsymbol{\alpha}} = \mathbf{A}^T\boldsymbol{\alpha}$ contains a reduced set of coefficients. The solution to this LS problem is found as

$$\breve{\boldsymbol{\alpha}} = (\mathbf{A}\breve{\mathbf{K}})^\dagger\mathbf{d} = \breve{\mathbf{K}}^{-1}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{d}, \tag{3.31}$$

where we have exploited the fact that $\breve{\mathbf{K}}$ is non-singular by construction of the dictionary $\mathcal{D}$.

Given the solution $\breve{\boldsymbol{\alpha}}_n$ of (3.30) at a certain instant n, ALD-KRLS provides an efficient way to update this solution as a new data pair $\{\mathbf{x}_{n+1}, d_{n+1}\}$ is received [Engel et al., 2004]. In particular, it stores the matrices $\breve{\mathbf{K}}_n^{-1}$, $(\mathbf{A}_n^T\mathbf{A}_n)^{-1}$, and the vector $\mathbf{A}_n^T\mathbf{d}_n$ and it updates them every time a new data pair is received. The dictionary $\mathcal{D}_n$ is either preserved or expanded with the input point $\mathbf{x}_{n+1}$.

The entire ALD-KRLS algorithm is reproduced in Alg. 3.5. It is efficient with $O(M^2)$ time complexity and $O(M^2)$ memory complexity per time step, and it has been successful in nonlinear problems including regression and time-series prediction. In contrast to linear RLS, which can be extended easily to exclude the influence of older data, it is not readily suitable for tracking time-varying environments, since it assumes a static model. Solutions to this problem will be explored in chapter 4 by removing less relevant points from the dictionary. Notice that ALD-KRLS can be extended by including a forgetting factor, but in order to adjust to variations in time it also requires modifications to the ALD criterion that allow either to remove stored data points or to modify the labels corresponding to these dictionary points.

3.4.2 Extended kernel recursive least-squares algorithm

From a state-space viewpoint, the standard RLS algorithm implicitly assumes that the data satisfy the model

$$\begin{aligned} \mathbf{s}_{n+1} &= \mathbf{s}_n \\ y_n &= \mathbf{x}_n^T\mathbf{s}_n + v_n, \end{aligned} \tag{3.32}$$

where $\mathbf{s}_n$ represents the state of the system, which is fixed in this case, $\mathbf{x}_n$ is the input to the system, y_n represents the observations and v_n is Gaussian white observation noise, for all n [Haykin, 2001]. The first equation of (3.32) is called the *state model*, while the second one is called the *observation model*. Due to its fixed state, this model, which also lies at the basis of the ALD-KRLS algorithm, does not perform well for time-varying systems.

Algorithm 3.5 Kernel Recursive Least-Squares with Approximate Linear Dependency criterion (ALD-KRLS)

```
initialize
  First center $\mathbf{c}_1 = \mathbf{x}_1$.
  Dictionary $\mathcal{D}_1 = \{\mathbf{c}_1\}$.
  $m = 1$.
  $\breve{\mathbf{K}}_1 = \kappa(\mathbf{x}_1, \mathbf{x}_1)$, $\breve{\mathbf{K}}_1^{-1} = 1/\kappa(\mathbf{x}_1, \mathbf{x}_1)$.
  $\breve{\boldsymbol{\alpha}}_1 = d_1/\kappa(\mathbf{x}_1, \mathbf{x}_1)$, $\mathbf{P}_1 = 1$.
for $n = 2, 3, \ldots$ do
  Receive $(\mathbf{x}_n, d_n)$.
  Kernels of dictionary and new point: $\breve{\mathbf{k}}_n = [\kappa(\mathbf{c}_1, \mathbf{x}_n), \cdots, \kappa(\mathbf{c}_m, \mathbf{x}_n)]^T$.
  Optimal linear expansion coefficients: $\mathbf{a}_n = \breve{\mathbf{K}}_{n-1}^{-1}\breve{\mathbf{k}}_n$.
  ALD error: $\delta_n = \kappa(\mathbf{x}_n, \mathbf{x}_n) - \breve{\mathbf{k}}_n^T\mathbf{a}_n$.
  if $\delta_n > \nu$ then
    New dictionary center: $\mathbf{c}_m = \mathbf{x}_n$.
    Update dictionary: $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\mathbf{c}_m\}$.
    Update inverse kernel matrix: $\breve{\mathbf{K}}_n^{-1} = \frac{1}{\delta_n}\begin{bmatrix} \delta_n\breve{\mathbf{K}}_{n-1}^{-1} + \mathbf{a}_n\mathbf{a}_n^T & -\mathbf{a}_n \\ -\mathbf{a}_n^T & 1 \end{bmatrix}$.
    Update projection matrix: $\mathbf{P}_n = \begin{bmatrix} \mathbf{P}_{n-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$.
    Update KRLS expansion coefficients: $\breve{\boldsymbol{\alpha}}_n = \begin{bmatrix} \breve{\boldsymbol{\alpha}}_{n-1} - \frac{\mathbf{a}_n}{\delta_n}\left(d_n - \breve{\mathbf{k}}_n^T\breve{\boldsymbol{\alpha}}_{n-1}\right) \\ \frac{1}{\delta_n}\left(d_n - \breve{\mathbf{k}}_n^T\breve{\boldsymbol{\alpha}}_{n-1}\right) \end{bmatrix}$.
    $m \leftarrow m + 1$.
  else
    Preserve dictionary: $\mathcal{D}_n = \mathcal{D}_{n-1}$.
    Calculate $\mathbf{q}_n = \frac{\mathbf{P}_{n-1}\mathbf{a}_n}{1+\mathbf{a}_n^T\mathbf{P}_{n-1}\mathbf{a}_n}$.
    Update projection matrix: $\mathbf{P}_n = \mathbf{P}_{n-1} - \mathbf{q}_n\mathbf{a}_n^T\mathbf{P}_{n-1}$.
    Update KRLS expansion coefficients: $\breve{\boldsymbol{\alpha}}_n = \breve{\boldsymbol{\alpha}}_{n-1} + \breve{\mathbf{K}}_n^{-1}\mathbf{q}_n\left(d_n - \breve{\mathbf{k}}_n^T\breve{\boldsymbol{\alpha}}_{n-1}\right)$.
  end if
end for
Output $\mathcal{D}_n$ and $\breve{\boldsymbol{\alpha}}_n$.
```

A more general linear state-space model is used in the *extended* recursive least-squares method

$$\begin{aligned} \mathbf{s}_{n+1} &= \mathbf{A}\mathbf{s}_n + \mathbf{n}_n \\ y_n &= \mathbf{x}_n^T\mathbf{s}_n + v_n, \end{aligned} \tag{3.33}$$

where $\mathbf{A}$ is the state transition matrix, and $\mathbf{n}_n$ is Gaussian white state noise. Kalman proposed a two-step sequential procedure to update the state estimate, commonly known as the *Kalman filter* [Kalman, 1960], which sequentially minimizes the fol-

Algorithm 3.6 Extended Kernel Recursive Least-Squares (EX-KRLS)

```
initialize
    $\mathbf{a}_1 = \frac{\alpha d_1}{\mathbf{c}\beta + \kappa(\mathbf{x}_1, \mathbf{x}_1)}$.
    Concomitant variable $\rho_1 = \frac{c\beta}{\alpha^2\beta + cq}$.
    Concomitant matrix $\mathbf{Q}_1 = \frac{\alpha^2}{(c\beta + k_{1,1})(\alpha^2 + c\beta q)}$.
for $n = 2, 3, \ldots$ do
    Kernels of stored points and new point: $\mathbf{k}_n = [\kappa(\mathbf{x}_1, \mathbf{x}_n), \cdots, \kappa(\mathbf{x}_{n-1}, \mathbf{x}_n)]^T$.
    $\mathbf{z}_n = \mathbf{Q}_{n-1}\mathbf{h}_n$.
    $r_n = \beta^n \rho_{n-1} + \kappa(\mathbf{x}_n, \mathbf{x}_n) - \mathbf{h}_n^T \mathbf{z}_n$.
    Algorithm output: $y_n = \mathbf{k}_n^T \mathbf{a}_{n-1}$.
    A-prior error: $e_n = d_n - y_n$.
    Update expansion coefficients: $\mathbf{a}_n = \alpha \begin{bmatrix} \mathbf{a}_{n-1} - \mathbf{z}_n r_n^{-1} e_n \\ r_n^{-1} e_n \end{bmatrix}$.
    $\rho_n = \frac{\rho_{n-1}}{\alpha^2 + \beta^n q \rho_{n-1}}$.
    $\mathbf{Q}_n = \frac{\alpha^2}{r_n(\alpha^2 + \beta^n q \rho_{n-1})} \begin{bmatrix} \mathbf{Q}_{n-1} r_n + \mathbf{z}_n \mathbf{z}_n^T & -z_n \\ -z_n^T & 1 \end{bmatrix}$.
end for
Output the expansion vector $\mathbf{a}_n$ containing the coefficients corresponding to
$\kappa(\mathbf{x}_i, \cdot)$, for $i = 1, \ldots, n$.
```

lowing LS cost function

$$J(\mathbf{w}) = \sum_{j=0}^{N} \left(\lambda^{N-j} \left| d_j - \mathbf{x}_j^T \mathbf{w} \right|^2 + c\lambda^N \|\mathbf{w}\|^2 + \lambda^{N-j} q^{-1} \|\mathbf{n}_n\|^2 \right) \quad \text{s.t. } \mathbf{x}_{j+1} = \mathbf{A}\mathbf{x}_j + \mathbf{n}_n, \tag{3.34}$$

where the parameter q provides a trade-off between the modeling variation and measurement noise. This problem is solved recursively by the *extended RLS algorithm* [Sayed, 2003] which is similar in form to Alg. 3.2.

Recently, Liu et al. proposed an extended kernel recursive least-squares (EX-KRLS) algorithm [Liu and Príncipe, 2008], that recursively obtains the solution of (3.34) in feature space. The matrix inversion lemma cannot be applied directly here because of the complicated constrained least-squares cost function (3.34). Therefore, a special case was considered where the state transition operator has the form $\mathbf{A} = a\mathbf{I}$. In [Liu and Príncipe, 2008] it was shown how this model, which indicates either an attenuating or amplifying model, is suitable for tracking problems such as slow fading communication channels. Moreover, to curb the growth of kernel matrix size, an ALD criterion was applied. The core algorithm (without ALD) is summarized in Alg. 3.6.

3.5 Conclusions

In this chapter we introduced the concept of online kernel methods. We pointed out the two most common drawbacks in taking kernel methods to online environments, specifically, overfitting problems and high computational and memory problems, along with a list of common solutions. We also discussed the most important online kernel methods, and showed how they are obtained directly by taking linear adaptive filtering techniques into feature space.

Part II

Identification and Equalization of Nonlinear Systems

Chapter 4
Supervised Identification of Nonlinear Systems

In this chapter we give an introduction to nonlinear system identification. After discussing the most important types of nonlinear systems, we introduce a specific family of block-based models, which will be used throughout the rest of this thesis. We then provide a classification of nonlinear identification scenarios and discuss the different kernel-based identification approaches. As a first contribution, we extend the previously described family of online kernel methods with a set of fixed memory size KRLS algorithms, and we show how they have certain advantages over other online kernel methods in time-varying environments.

4.1 Volterra and Wiener theory of Nonlinear Systems

The field of nonlinear system identification has been studied for many years and is still an active research area [Billings, 1980, Kalouptsidis and Theodoridis, 1993, Sjöberg et al., 1995, Nelles, 2000, Giannakis and Serpedin, 2001]. In general, system identification consists in trying to infer the functional relationship between a system's input and output, based on observations of the in- and outgoing signals.

For *linear* systems, identification follows a unified approach thanks to the *superposition principle*. This principle states that the output of a linear combination of input signals to a linear system is the same linear combination of the outputs of the system corresponding to the individual components. Or, simply put, the following holds for the response $\mathcal{L}$ of every linear system:

$$\mathcal{L}(\alpha\mathbf{x}_1 + \beta\mathbf{x}_2) = \alpha\mathcal{L}(\mathbf{x}_1) + \beta\mathcal{L}(\mathbf{x}_2), \tag{4.1}$$

where α and β are scalars and $\mathbf{x}_1$ and $\mathbf{x}_2$ represent inputs to the system. As a consequence, if a system is linear and time-invariant (LTI) it can be characterized completely by its impulse response.

Nonlinear systems, however, do not satisfy the superposition principle, and there does not exist an equivalent canonical representation for all nonlinear systems. Consequently, the traditional approach to studying nonlinear systems is to consider only one class of systems at a time, and to develop a parametric description that fits this class and allows to analyze it.

One of the earliest approaches to parameterize nonlinear systems was introduced by Volterra, who proposed to extend the standard convolutional description of linear systems by a series of polynomial integral operators H_i with increasing degree of nonlinearity [Volterra, 1887]. For discrete-time systems, this description becomes[1]

$$y[n] = H_0 + \sum_{p=1}^{\infty} H_p(x[n]), \tag{4.2}$$

where $x[n]$ and $y[n]$ are the input and output signals. Assuming the described nonlinear system is causal and of finite memory, the functionals H_p can be expanded as

$$H_p(x[n]) = \sum_{m_1=0}^{M-1} \cdots \sum_{m_p=0}^{M-1} h^{(p)}_{m_1,\ldots,m_p} x[n-m_1] \cdots x[n-m_p], \tag{4.3}$$

where $h^{(p)}_{m_1,\ldots,m_p}$ is the p-th order Volterra kernel of the system. The entire Volterra kernel H_p is then described by its M^p coefficients $h^{(p)}_{m_1,\ldots,m_p}$ [Alper, 1965]. Note that *Volterra series* are directly related to Taylor series, but they extend this concept by allowing the represented system's output to depend on past inputs.

If the input signals are restricted to a suitable subset of the input function space, it can be shown that any continuous, nonlinear system can be uniformly approximated up to arbitrary accuracy by a Volterra series of finite order [Fréchet, 1910, Brilliant, 1958, Boyd and Chua, 1985]. This is a generalization of the Stone-Weierstraß theorem, which states that every continuous function of a variable x defined on an interval $[a, b]$ can be approximated with arbitrary precision as a polynomial in x. Thanks to this approximation capability, Volterra series have become a well-studied subject [Schetzen, 1980, Giannakis and Serpedin, 2001, Franz and Schölkopf, 2006] with applications in numerous fields. In particular, finite-order Volterra series that have finite memory (such as the truncated series of Eq. (4.3)) lie at the basis of the field of *polynomial signal processing* [Mathews and Sicuranza, 2000].

In order to identify a nonlinear system ideally, the system response has to be measured for all possible input functions. Volterra theory dictates that the appropriate system representation can be found by minimizing the L_∞ norm between the true output and the system output. Wiener proposed some relaxations to this identification scheme [Wiener, 1958, Papoulis, 1984] by introducing a framework of so-called G-functionals, which are directly related to the original Volterra operators. He showed that if the input to the system is white and Gaussian, the G-functionals are mutually orthogonal and the entire system can be identified in the mean squared error sense (by minimizing the L_2-norm). This idea, which was exploited in the popular cross-correlation method of Lee and Schetzen [Lee and Schetzen, 1965], allows to measure the Volterra kernels directly.

[1] Since the variables in this chapter often require subscripts, we will use brackets to indicate the time index of a point, such as $\mathbf{x}[n]$, from this chapter on.

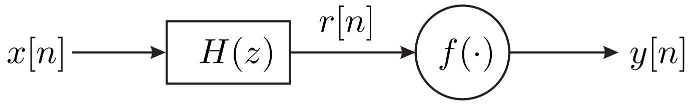


Figure 4.1: Block diagram of a Wiener system.

The main drawback to representing nonlinear systems as Volterra series is that for growing degrees of nonlinearity and input dimension, the corresponding Volterra representation requires to estimate an exponentially growing number of terms. Therefore their application is limited to low-dimensional systems with mild nonlinearities. In the sequel we will focus on simplified nonlinear system models that represents a trade-off between system complexity and nonlinear approximation capabilities.

4.2 Wiener and Hammerstein Systems

Although the functional series expansion of Volterra series provides an adequate representation for a large class of nonlinear systems, practical identification schemes based on this description often result in an excessive computational load. It is for this reason that several authors have considered the identification of specific configurations of nonlinear systems, notably cascade systems composed of linear subsystems with memory and continuous zero-memory nonlinear elements [Narendra and Gallman, 1966, Gardiner, 1973].

A first such system, known as the Wiener system [Billings and Fakhouri, 1977], consists of a linear filter followed by a static memoryless nonlinearity, as illustrated in Fig. 4.1. In case a finite impulse response (FIR) filter is chosen as the linear part, represented as

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + \cdots + h_{L-1} z^{-L+1}, \tag{4.4}$$

the system's output for a given input signal $x[n]$, $n = 0, 1, 2, \ldots$ is obtained as

$$y[n] = f\left(\sum_{i=0}^{L-1} h_i x[n-i]\right), \tag{4.5}$$

where h_i, $i = 0, \ldots, L-1$ represents the impulse response of the linear filter and $f(\cdot)$ is the nonlinearity. This notation can be shortened by introducing the time-delay vector $\mathbf{x}[n]$, as

$$y[n] = f\left(\mathbf{x}[n]^T \mathbf{h}\right). \tag{4.6}$$

The Wiener model is a much more simplified version of Wiener's original nonlinear system characterization [Wiener, 1958]. However, despite its simplicity, it has been used successfully to describe a number of nonlinear systems appearing in practice. Common applications include biomedical engi-

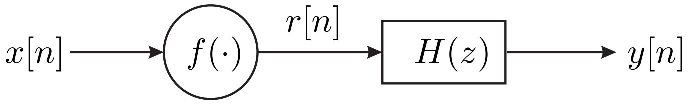


Figure 4.2: Block diagram of a Hammerstein system.

neering [Hunter and Korenberg, 1986, Westwick and Kearney, 1998], control systems [Billings and Fakhouri, 1982, Greblicki, 2004], digital satellite communications [Feher, 1983], digital magnetic recording [Sands and Cioffi, 1993], optical fibre communications [Kawakami Harrop Galvão et al., 2007] and chemical processes [Pajunen, 1992].

The inverse configuration, a series connection of a static memoryless nonlinearity and a linear filter, is known as the Hammerstein system [Billings and Fakhouri, 1979] (see Fig. 4.2). In case its linear part is represented by a FIR filter, the output of a Hammerstein system is obtained as $y[n] = \sum_{i=0}^{L-1} h_i f\left(x[n-i]\right)$. Hammerstein systems are encountered for instance in electrical drives [Balestrino et al., 2001], acoustic echo cancelation [Ngia and Sjobert, 1998], heat exchangers and biomedical modeling [Westwick and Kearney, 2000].

Wiener and Hammerstein systems form particular types of block-oriented structures [Chen, 1995, Giannakis and Serpedin, 2001]. Other popular cascade models include the so-called "sandwich" models that combine more than two blocks, for instance the Hammerstein-Wiener model, which consists of a regular Hammerstein system followed by an additional nonlinearity.

4.2.1 Review of identification techniques

Many identification approaches for Wiener and Hammerstein systems have been reported in the literature since the late seventies. Most of them are supervised techniques, although in the last decade a number of blind identification methods have been proposed as well.

Traditional supervised methods followed a black-box approach, which does not make any assumption about the system structure. Nonlinear equalization or identification was tackled by considering nonlinear structures such as multilayer perceptrons (MLPs) [Erdogmus et al., 2001a], recurrent neural networks [Kechriotis et al., 1994] or piecewise linear networks [Adali and Liu, 1997]. Other black-box identification methods include techniques based on orthogonal least-squares expansion [Korenberg, 1989, Chen et al., 1989] and separable least-squares [Bruls et al., 1999].

To improve identification results, a number of algorithms were introduced that exploit the Wiener or Hammerstein's structure explicitly. This can be done for instance by an identification scheme that mimics the unknown system, and estimates its parameters iteratively [Narendra and Gallman, 1966, Billings and Fakhouri, 1977, Billings and Fakhouri, 1982, Hunter and Korenberg, 1986, Wigren, 1994, Greblicki, 1997, Westwick and Kearney, 2000, Haykin, 2001, Greblicki, 2004, Dempsey and Westwick, 2004].

Another approach that exploits the system's structure consists of a two-step procedure that consecutively estimates the linear part and the nonlinearity of the Wiener or Hammerstein systems. Most proposed two-step techniques are based on predefined test signals [Bai, 1998, Pawlak et al., 2007, Wang et al., 2007].

An even different proposition is found in [Aschbacher and Rupp, 2005], where both blocks of the nonlinear system are estimated simultaneously through a coupled regression on the unknown intermediate signal. In chapter 5 we will generalize this technique and present a family of robust identification and equalization techniques based on a kernel canonical correlation analysis (kernel CCA) framework [Van Vaerenbergh et al., 2006a, Van Vaerenbergh et al., 2008a].

Extensions to the standard identification settings have also been proposed, for instance to account for complex signals [Cousseau et al., 2007] and multiple-input multiple output (MIMO) Wiener or Hammerstein systems [Goethals et al., 2005].

Although all above-mentioned techniques are supervised approaches (i.e., input and output signals are known during estimation), recently, there have also been a few attempts to identify Wiener and Hammerstein systems *blindly*. Most of these techniques make certain assumptions about the input signal, for instance requiring it to be Gaussian and white [Gómez and Baeyens, 2007, Vanbeylen et al., 2008]. Taleb et al. presented a less restrictive method that only assumed the input signal was white (i.i.d.) [Taleb et al., 2001]. The resulting technique aims to recover the input signal by minimizing the mutual information of the inversion system output. Recently, we proposed a different approach based on oversampling of the nonlinear system's output, which does not make any assumption about the input signal's statistics. This technique will be discussed in detail in chapter 6.

4.3 Nonlinear System Identification with Kernels

In this part of the thesis we are interested in nonlinear system identification techniques that are built specifically on the RKHS framework. These methods exhibit the interesting property that they allow to construct universal approximators in the form of expansions of kernel functions. In this section we will describe the different scenarios of system identification that will be treated in this work, and we will introduce the corresponding kernel-based identification approaches.

In the most common scenario of nonlinear system identification, input and output signals are available beforehand and no further information is given about the system. An appropriate kernel-based identification technique for this scenario is kernel least-squares regression, which was already discussed in section 2.2.1. If the system's input and output data are not known beforehand, but instead received sequentially, the online kernel methods from chapter 3 could be employed.

In general, to determine what type of identification algorithm is needed, we should ask the following three questions:

- Are all data readily available to the algorithm, or do some data arrive after a first solution is obtained? If all data are available, a *batch* (or *block-based*) algorithm can be constructed. If this is not the case, an *adaptive* (or *online*) algorithm is required, which is capable of updating the solution based on the newly arriving data.
- Are both the input and output signals available, or only the output signal? The first scenario allows for *supervised* identification, while in the second case only *blind* identification is possible.
- Can the nonlinear system be described by a certain model? If this is the case, such structure should be exploited to achieve a more accurate identification algorithm. If not, a standard *black-box* algorithm is required.

Based on these criteria, we can distinguish among a number of different identification scenarios.

1. **Batch supervised black-box identification.** This is the standard case of nonlinear identification, as introduced in section 2.2.1. All data is available, but no information about the nonlinear mapping is given.
2. **Online supervised black-box identification.** In this scenario an online algorithm is required, such as the kernel methods presented in chapter 3.
3. **Supervised identification of model-based nonlinear systems.** In chapter 5 we will discuss several identification and equalization algorithms for this scenario, both for batch and online problems settings. To this end, we will develop a framework of kernel canonical correlation analysis (kernel CCA). The nonlinear system models chosen in this work will be the Wiener and the Hammerstein systems.
4. **Blind identification of model-based nonlinear systems.** Blind techniques aim to identify a system given only its output signal. In doing so, it is required to make assumptions about either the input signal statistics, the system's structure, or both. While it is common to make assumptions about both, we will present a technique that allows to identify and equalize a Wiener system blindly by only exploiting its structure. This will be the topic of chapter 6. Although the proposed technique is a batch algorithm, it performs a recursive estimate and therefore it is readily extendible to online problem settings.

Before addressing the identification of specific model-based nonlinear systems in the next chapter, the rest of this chapter will be dedicated to a set of online kernel-based techniques we proposed for supervised black-box identification (the second scenario in the previous list). These technique mark the first contribution of this thesis, and they take up the overview of adaptive kernel filtering techniques where we left it in chapter 3. Moreover, they will establish building blocks for the algorithms introduced in the next chapter.

4.4 Sliding-Window Kernel RLS

The ALD-KRLS algorithm [Engel et al., 2004] discussed in section 3.4 provides an efficient means to calculate the solution to a batch least-squares nonlinear identification problem. In order to achieve a low computational complexity, it starts with a solution valid for only one or a few data points and recursively updates this solution to account for more data. It can also be applied in an online scenario where not all data is available beforehand, although it requires that the nonlinearity to be identified is static.

Unlike the linear RLS algorithm, ALD-KRLS is not able to operate in a time-varying environment. In linear RLS this is made possible for instance by including a forgetting factor, yielding an *exponentially weighted* RLS algorithm. ALD-KRLS, however, is based on the ALD criterion, which does not have a mechanism to exclude the influence of already stored data points. As the nonlinear mapping changes in a time-varying environment, input-output pairs stored by the ALD criterion will become invalid and provide erroneous information to the algorithm. Although one could try to lower the influence of such outdated input-output pairs, for instance by applying a forgetting factor, the ALD criterion will not admit newer, possibly more accurate input-output pairs to be stored if their input data coincides with any of the older stored data points.

Another aspect of the ALD-KRLS algorithm that could be improved is its growing memory. Although this growth is curbed by the ALD criterion and it will eventually come to a halt, in many applications it would be interesting to limit the size of the final memory. While this can be done in ALD-KRLS by manually stopping the growth, the optimality of the obtained memory is not guaranteed.

In the following we will first introduce a simple KRLS algorithm with a fixed memory size, that can operate as a tracker. Afterwards, we will improve this algorithm by equipping it with a criterion that allows to construct its memory optimally.

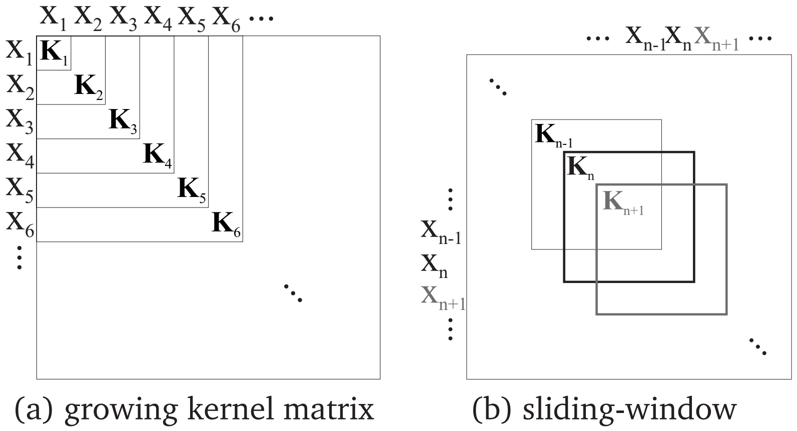
4.4.1 A sliding-window approach

In an online scenario, the solution of the regularized kernel LS regression problem (2.41) reads

$$\boldsymbol{\alpha}_n = \left(\mathbf{K}_n^{reg}\right)^{-1} \mathbf{d}_n, \tag{4.7}$$

at every time instant n. Here, $\boldsymbol{\alpha}_n$ contains the kernel regression coefficients, $\mathbf{K}_n^{reg} = \mathbf{K}_n + c\mathbf{I}$ is the regularized kernel matrix with regularization constant c, and $\mathbf{d}_n$ represents the vector of desired system outputs.

In order to exclude less important data points contained in (4.7), we presented an approach that only considers the last M data points in every time step [Van Vaerenbergh et al., 2006b]. This approach applies a *sliding window* of length M to the online data stream of input-output pairs $\{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \dots\}$. For the n-th sliding window, the observation matrix $\mathbf{X}_n = [\mathbf{x}_n, \mathbf{x}_{n-1}, \dots, \mathbf{x}_{n-M+1}]^T$ is formed, and the corresponding kernel matrix $\mathbf{K}_n^{reg} = \tilde{\mathbf{X}}_n\tilde{\mathbf{X}}_n^T + c\mathbf{I}$ is obtained from it (see Fig. 4.3). Similarly, the M last desired outputs are grouped into the vector



(a) growing kernel matrix (b) sliding-window

Figure 4.3: (a) Kernel matrices $\mathbf{K}_n$ of growing size. (b) Kernel matrices $\mathbf{K}_n$ of fixed size, obtained by only considering the data in windows of fixed size.

$\mathbf{d}_n = [d_n, d_{n-1}, \ldots, d_{n-M+1}]^T$. As a consequence, this method represents the nonlinear mapping at every time step n as the nonlinear regressor (4.7) obtained only from the M data points contained in the sliding window:

$$f_n(\mathbf{x}) = \sum_{i=1}^{M} \alpha_{n,i}\kappa(\mathbf{x}_{n-M+i}, \mathbf{x}) = \mathbf{k}_n(\mathbf{x})^T \boldsymbol{\alpha}_n, \tag{4.8}$$

where $\alpha_{n,i}$ represents the i-th coefficient of $\boldsymbol{\alpha}_n$ and $\mathbf{k}_n(\mathbf{x})$ contains the kernel functions between all points in the window and the test point $\mathbf{x}$.

In the following, we will present an efficient scheme to update the solution (4.7) when a new input-output data pair is received.

4.4.2 Updating the inverse of the kernel matrix

Given the solution of (4.7) at time instant $n-1$, $\boldsymbol{\alpha}_{n-1}$, the sliding-window approach requires to obtain the inverse of the new regularized[2] kernel matrix $\mathbf{K}_n^{-1}$ and a new output vector $\mathbf{d}_n$. Obtaining the new output vector is straightforward, as it only consists in adding the new data point d_n to $\mathbf{d}_{n-1}$ and removing the oldest data point d_{n-M} from it. The calculation of the new kernel matrix requires the calculation of the $M \times M$ inverse matrix $\mathbf{K}_n^{-1}$. This is costly both in time (requiring $O(M^3)$ operations) and in memory ($O(M^2)$). To avoid the direct computation of the matrix inverse, we proposed an update algorithm in [Van Vaerenbergh et al., 2006b] that allows to obtain $\mathbf{K}_n^{-1}$ solely from the knowledge of the data of the current window $\{\mathbf{X}_n, \mathbf{d}_n\}$ and the previous inverse kernel matrix $\mathbf{K}_{n-1}^{-1}$.

Given the regularized kernel matrix $\mathbf{K}_{n-1}$, the new regularized kernel matrix $\mathbf{K}_n$ can be constructed in two steps. First, the first row and column of $\mathbf{K}_{n-1}$ are removed.

[2] With slight abuse of notation, we will denote by $\mathbf{K}_n$ the regularized matrix that includes the regularization term $c\mathbf{I}$, wherever no confusion is possible.

Algorithm 4.1 Sliding-Window Kernel Recursive Least-Squares (SW-KRLS)

```
initialize
    Obtain $\mathbf{K}_0$.
    Calculate and store $\mathbf{K}_0^{-1}$.
for $n = 1, 2, \ldots$ do
    Obtain $\mathbf{d}_n$: remove the oldest point from $\mathbf{d}_{n-1}$ and add $d_n$ to it.
    Downsize $\mathbf{K}_{n-1}^{-1}$: Obtain $\bar{\mathbf{K}}_{n-1}$ out of $\mathbf{K}_{n-1}$, and compute $\bar{\mathbf{K}}_{n-1}^{-1}$ with (C.4).
    Upsize $\bar{\mathbf{K}}_{n-1}^{-1}$: Obtain $\mathbf{K}_n$ as in (4.9), and compute $\mathbf{K}_n^{-1}$ according to (C.3).
    Update the solution: $\boldsymbol{\alpha}_n = \mathbf{K}_n^{-1}\mathbf{d}_n$.
end for
```

This step is referred to as *downsizing* the kernel matrix, and the resulting matrix is denoted as $\bar{\mathbf{K}}_{n-1}$. In the second step, kernels of the new data are added as the last row and column to this matrix:

$$\mathbf{K}_n = \begin{bmatrix} \bar{\mathbf{K}}_{n-1} & \bar{\mathbf{k}}_{n-1}(\mathbf{x}_n) \\ \bar{\mathbf{k}}_{n-1}(\mathbf{x}_n)^T & k_{nn} + c \end{bmatrix}, \tag{4.9}$$

where $\bar{\mathbf{k}}_{n-1}(\mathbf{x}_n) = [\kappa(\mathbf{x}_{n-M+1}, \mathbf{x}_n), \ldots, \kappa(\mathbf{x}_{n-1}, \mathbf{x}_n)]^T$ and $k_{nn} = \kappa(\mathbf{x}_n, \mathbf{x}_n)$. This step is referred to as *upsizing* the kernel matrix. Notice that every element of the diagonal of this matrix contains the regularization term c.

Calculating the inverse kernel matrix $\mathbf{K}_n^{-1}$ is also done in two steps, using two inversion formulas derived in appendices C.1 and C.2. First, given the previous kernel matrix $\mathbf{K}_{n-1}$ and its inverse $\mathbf{K}_{n-1}^{-1}$, the inverse of the *downsized* $(M-1) \times (M-1)$ matrix $\bar{\mathbf{K}}_{n-1}$ is calculated according to Eq. (C.4). Then, the matrix $\bar{\mathbf{K}}_{n-1}$ is *upsized* to obtain $\mathbf{K}_n$, and based on the knowledge of $\mathbf{K}_n$ and $\bar{\mathbf{K}}_{n-1}^{-1}$ the inversion formula from Eq. (C.3) is applied to obtain $\mathbf{K}_n^{-1}$. Note that these formulas do not calculate the inverse matrices explicitly, but rather derive them from known matrices maintaining an overall time complexity of $O(M^2)$ of the algorithm, where M is the window length.

To initialize the algorithm, we can either start with an empty window (of length 0) and let this grow to length M during the first M iterations by only applying the upsizing operations, or apply zero-padding of the data to fill an initial window of length M. In the latter case, the initial observation matrix $\mathbf{X}_0$ consists entirely of zeros and the regularized kernel matrix $\mathbf{K}_0$ corresponding to these data and its inverse can easily be computed. In case a Gaussian or polynomial kernel function is used, this matrix is $\mathbf{K}_0 = \mathbf{1} + c\mathbf{I}$. The complete algorithm is summarized in Alg. (4.1).

4.5 Experiments with Sliding-Window Kernel RLS

To show the advantages of kernel-based methods over classical nonlinear algorithms, the performance of the sliding-window kernel RLS algorithm was compared with a multi-layer perceptron (MLP) in [Van Vaerenbergh et al., 2006b, Van Vaerenbergh et al., 2007b], for online identification tasks. In this section we reproduce those results.

The supervised identification tasks were performed on static and time-varying Wiener systems, which were considered black-box nonlinear systems during identification. Different amounts of additive noise were considered at the output of the system, which is a standard practice in identification applications[3]. Furthermore, the linear part of the Wiener systems is modeled as a FIR filter.

4.5.1 Identification of a Wiener System with an abrupt channel change

We consider a supervised identification problem of a static Wiener system, in which at a given time instant the linear channel coefficients are changed abruptly. This allows to compare the tracking capabilities of both algorithms: During the first part of the simulation, the linear channel is $H_1(z)$ and after receiving 500 symbols it is changed into $H_2(z)$, with

$$\begin{aligned} H_1(z) &= 1 - 0.3668z^{-1} - 0.4764z^{-2} + 0.8070z^{-3} \\ H_2(z) &= 1 - 0.8326z^{-1} + 0.6656z^{-2} + 0.7153z^{-3}. \end{aligned}$$

A binary signal $x_n \in \{-1, +1\}$ is sent through this channel, after which the signal is transformed nonlinearly according to the nonlinear function $f(\cdot) = \tanh(\cdot)$. Finally, 20dB of additive white Gaussian noise (AWGN) is added. In these experiments, the Wiener system is treated as a black box of which only the input and output are known. To fill the initial sliding window, zero-padding of the signals was applied. The MLP was trained in an online manner, i.e. in every iteration one new data point was used to update the net weights.

Comparison to MLP

System identification was first performed by an MLP with 8 neurons in its hidden layer and learning rate 0.1, and then by using the sliding-window kernel RLS with window size $N = 150$ and using a polynomial kernel of order $p = 3$. For both methods we applied time-embedding techniques assuming that the length L of the linear channel was known. More specifically, the used MLP was a time-delay MLP with L inputs, and the input vectors for the kernel RLS algorithm were time-delayed vectors of length L, $\mathbf{x}_n = [x_n, \ldots, x_{n-L+1}]^T$. The length of the used linear channels H_1 and H_2 was $L = 4$.

At iteration n, the input-output pair $(\mathbf{x}_n, d_n)$ is fed into the identification algorithm. The performance is then evaluated by estimating the next output sample $y_{n+1} = \hat{d}_{n+1}$, given the next input vector x_{n+1}, and comparing it to the actual output d_{n+1}. For both methods and all subsequent experiments in this section, the mean square error (MSE) was averaged out over 250 Monte-Carlo simulations.

The MSE for both approaches is shown in Fig. 4.4. From iteration 500 to 650 it is observed that the algorithm needs exactly $N = 150$ iterations to adjust completely to

[3] Some models suggest adding the noise component to the signal on the intermediate of the Wiener system. We will not specifically consider such models.

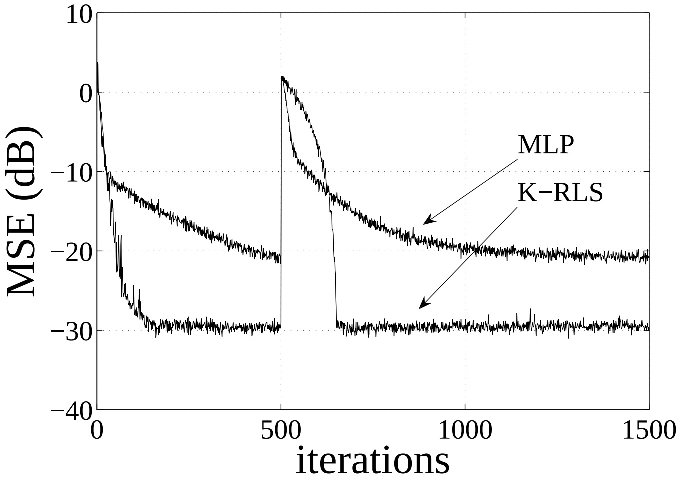


Figure 4.4: MSE of the identification of the nonlinear Wiener system of Fig. 4.1, for the standard method using an MLP and for the window-based kernel RLS algorithm with window length $N = 150$. A change in filter coefficients of the nonlinear Wiener system was introduced after sending 500 data points.

the channel change. For the chosen learning rate, the MSE of the MLP initially drops fast but it converges to a higher MSE.

Recently, a modification to the original sliding-window kernel RLS technique was presented [Julian, 2009]. Based on the observation that an abrupt change in the observed nonlinear system causes the MSE performance to peak, this algorithm proposes to downsize the kernel matrix by more than one sample if a certain error threshold is exceeded. By excluding multiple older data points, which account for a large share of the error, it reduces an important part of the error peaks after an abrupt change. By restricting the maximum number of computations per iteration to be the same as for the original sliding-window KRLS algorithm, this algorithm determines the maximal number of points that can be removed by the operation of downsizing, which is computationally less expensive than upsizing. To recover the original kernel matrix size, the algorithm can either omit the downsize operation during a number of iterations, or recursively upsize the kernel matrix with the remaining (older) data in the memory. Later in this chapter, we will present a different extension of the original sliding-window kernel RLS algorithm based on a "fixed-budget" criterion.

In the following, the influence of the original algorithm's parameters is discussed. As a reference, we use to the basic setup with $N = 150$, $L = 4$ and 20dB SNR.

Influence of parameters

Window length Fig. 4.5 shows the performance of the kernel RLS algorithm for different sliding-window lengths N. First, it is observed that a larger window corresponds to slower convergence of the algorithm after the channel change. This occurs because N iterations are needed to replace the data in the kernel matrix with data

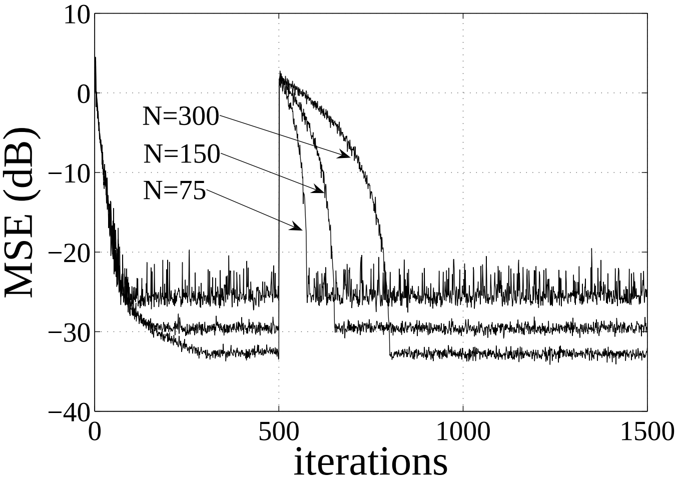


Figure 4.5: Comparison of the identification results of the kernel RLS algorithm for different window lengths for Wiener system identification. Larger windows obtain a better MSE. In general, the number of iterations needed for convergence is of the order of the window length.

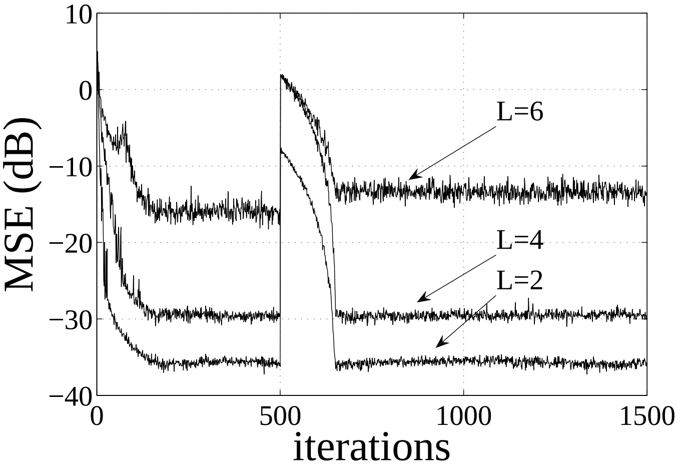


Figure 4.6: Comparison of the identification results of the kernel RLS algorithm for different Wiener system channel lengths. Since the same sliding-window length was used in the three situations, the best results are obtained for the shortest channel. Note that the channel length was known while performing identification.

corresponding to the new channel. Second, for small windows, peaks appear in the MSE. The kernel method generally combines the images of the data points within one window to represent the output of the nonlinear system. For small windows (such as $N = 75$) the data points within one window are often insufficient to sample the entire data range. Whenever a new point $\mathbf{x}_n$ is observed that falls outside of the

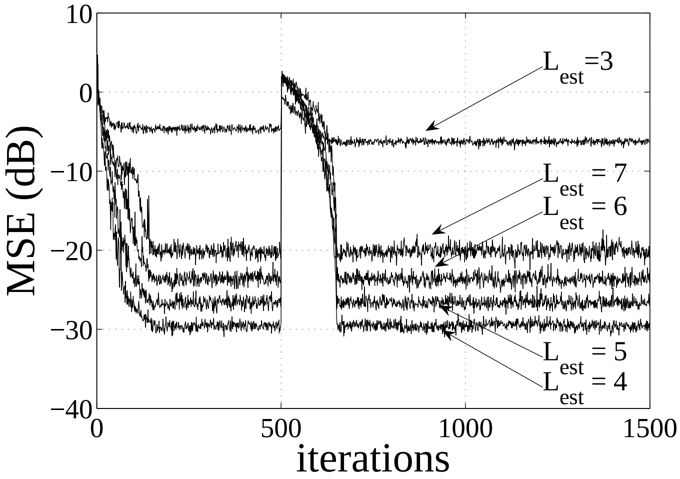


Figure 4.7: Identification results when the correct channel length $L = 4$ is not known. The curve $L_{est} = 3$ shows that underestimation of the channel leads to a worse performance than overestimation, which corresponds to $L_{est} > 4$.

sampled region, the performance shows an error peak. This also explains why the algorithm converges to a lower error for larger windows.

Channel length If a longer channel is used in a Wiener system, resulting in a higher input dimension, more basis vectors are needed by the sliding-window kernel RLS algorithm to sample the input space adequately. As can be seen in Fig. 4.6, performance drops if the same sliding-window length $N = 150$ is used in identification of Wiener systems with different channel lengths.

If the correct channel length L is not known, an estimate L_{est} can be made. The simulations of Fig. 4.7 show that the algorithm is very sensitive to underestimations of the channel length ($L_{est} < L$). In case the correct channel length is not known, it is preferred to overestimate this length slightly ($L_{est} > L$). Note that the computation time is hardly affected by an increase of the estimated channel length, since the input data dimension is of little importance once the kernel functions are computed.

Noise level Fig. 4.8 shows the MSE curves for different SNR values for the white Gaussian additive noise, which reflect the variance of the noise directly.

Regularization constant Although the regularization constant is necessary to avoid overfitting, the algorithm is robust to changes in its actual value, as can be seen in Fig. 4.9. For very large values ($c > 20$) the effect of the regularization becomes dominant.

Kernel function All previous tests were performed using a polynomial kernel function, which yielded satisfactory results, given the polynomial shape of the observed

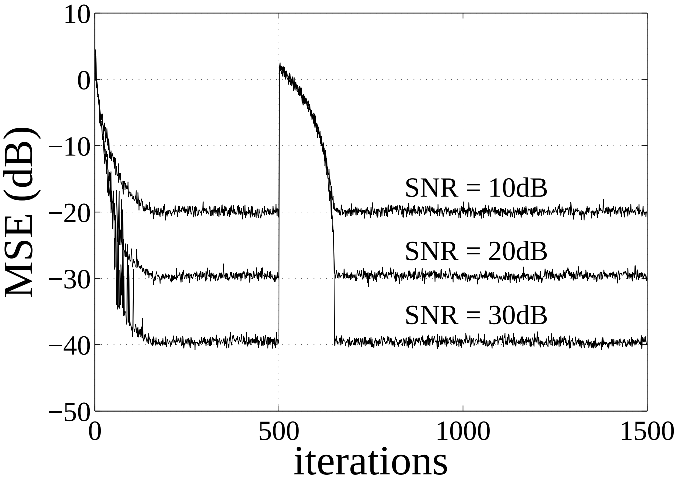


Figure 4.8: Identification results of the kernel RLS algorithm for different system SNR values in Wiener system identification.

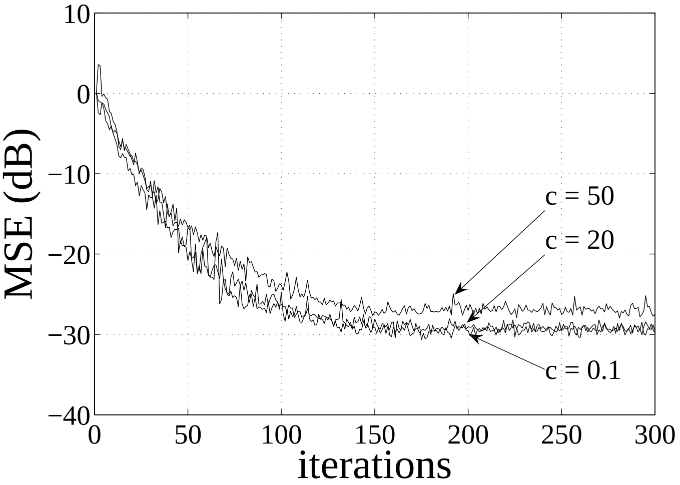


Figure 4.9: Influence of the regularization constant on the algorithm's performance. For $c \leq 20$ the obtained performance is very similar. For larger values, such as $c = 50$, regularization has a dominating effect on the performance.

nonlinearity. When the Gaussian kernel was used, the algorithm's performance dropped slightly.

4.5.2 Identification of a slowly time-varying Wiener System

In *linear* systems that are varying in time, the RLS algorithm with forgetting factor (see Alg. 3.2) can be applied to identify the system and track the system changes. Here, we show that a sliding-window kernel RLS algorithm is also capable of tracking time variations. Fig. 4.10 shows the MSE curve for a Wiener system in which

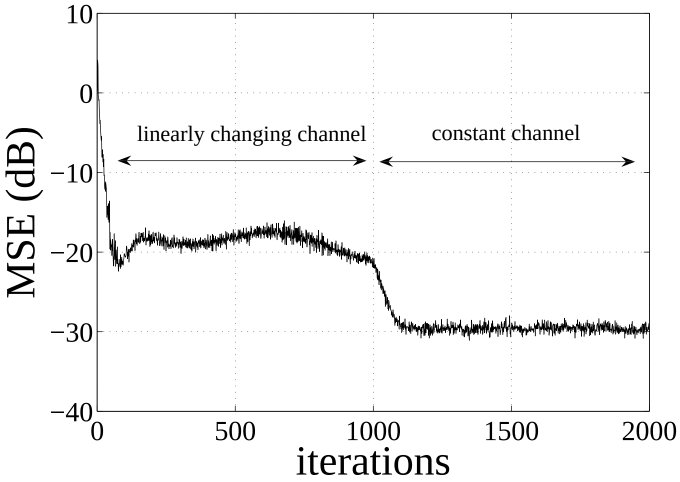


Figure 4.10: MSE for time-tracking of a Wiener system in which the channel varies linearly.

the channel varies linearly over the first 1000 iterations from $H_1(z)$ to $H_2(z)$ and is static over the next 1000 iterations. When the channel is varying in time, the sliding-window kernel RLS algorithm obtains a slightly worse MSE than for the static channel.

4.6 Fixed-Budget Kernel RLS

In sliding-window kernel RLS, the entire window is used as the kernel support to represent the unknown nonlinear mapping. In every time step, the *oldest point* is discarded from this memory, without considering the contribution of this point to the current the kernel expansion. As a result, the performance is strongly affected every time a significant point is discarded. An improvement to this procedure consists in pruning the *least significant* point from the memory instead. Thanks to this modification, the sliding-window technique can be seen as a special case of a broader family of *fixed-budget* kernel RLS algorithms [Van Vaerenbergh et al., 2010b].

A central idea in fixed-budget kernel RLS is that not all training data are equally significant. By letting the algorithm decide what data to maintain for its kernel expansion, we are giving it a more active role. For that reason, fixed-budget kernel RLS belongs to the class of *active learning* algorithms.

4.6.1 Network pruning

To achieve a sparse network, one can either construct a network by adding representative centers as shown in sections 2.3.2 and 3.2.2, or start from a large network and prune less significant centers. Sparseness is preferred since it improves generalization and lowers time and memory complexity. Pruning techniques have been well studied in the context of neural network design, for instance in optimal brain damage [Le Cun et al., 1989] and optimal brain surgeon [Hassibi et al., 1993], which are based on an analysis of the Hessian of the error surface. In [de Kruif and de Vries, 2003, Hoegaerts et al., 2004] a number of different, easier to evaluate criteria were presented to prune least-squares support vector machines (LS-SVM).

A few methods have been proposed that combine growing and pruning procedures, for instance generalized growing and pruning (GGAP) [Huang et al., 2005], the forgetron [Dekel et al., 2008], and sliding-window algorithms [Kivinen et al., 2004, Van Vaerenbergh et al., 2006b], which limit the memory to the M newest data points. These approaches are interesting with practical applications in mind, such as implementations on a microchip, since they allow to put an exact upper bound on the memory size and the number of computations needed. Moreover, they are capable of forgetting past data, which makes them suitable for operating in time-varying environments. However, they could be improved by applying a more intelligent discarding criterion.

Given the system memory of M points $(\mathbf{x}_i, d_i)$ at iteration $n-1$, we first add the new point $(\mathbf{x}_n, d_n)$ to the memory. A simple approach to determine the least significant point consists in performing kernel regression to map the $M+1$ inputs $\mathbf{x}_i$ onto the $M+1$ labels d_i by applying (4.7), which can be calculated efficiently in a recursive manner (see section 4.4.2). Since each of the obtained coefficients α_i represents the contribution of the i-th data point to the nonlinear mapping, the least significant point in the current regression is found as the one that has the lowest absolute regression coefficient $|\alpha_i|$. It is easily seen that the evaluation of this criterion yields a computational complexity of $O(M)$ in each iteration.

In [de Kruif and de Vries, 2003] a criterion was proposed that prunes the point that bears least error after it is omitted, in the context of least-squares support vector machines (LS-SVM). This error can be obtained as

$$d(\mathbf{x}_i, d_i) = \frac{|\alpha_i|}{[\breve{\mathbf{K}}_n^{-1}]_{i,i}}, \tag{4.10}$$

where $\breve{\mathbf{K}}_n^{-1}$ is the inverse regularized kernel matrix consisting of $M+1$ elements (before downsizing), and $[\breve{\mathbf{K}}_n^{-1}]_{i,i}$ denotes the i-th element on its diagonal. Although obtaining the inverse kernel matrix requires an additional computation when used to prune a LS-SVM, this matrix is readily available in fixed-budget kernel RLS, as it is calculated to update the KRLS solution. Therefore, the complexity of this criterion is also $O(M)$. Moreover, according to [de Kruif and de Vries, 2003] this criterion obtains significant better performance than the regression-based criterion. For these reasons we choose to use this criterion in fixed-budget kernel RLS.

Note that many other, more sophisticated criteria can be designed to prune the memory. However, most of them are computationally much more expensive. We will discuss some of them in the future research lines of this thesis (see chapter 10).

4.6.2 Inverse matrix update

Once the least significant point has been determined, it is discarded from the memory. This also requires to recalculate the new kernel and inverse kernel matrices. To obtain the downsized inverse kernel matrix, the matrix update procedure of the sliding-window kernel RLS algorithm can be extended in a straightforward fashion. Previously, we showed how the inverse kernel matrix could be obtained in an efficient manner after the first row and column have been removed. In the current case, not the first but the L-th row and column need to be removed, where L is the index of the least significant point in memory. As is shown in appendix C.2.2, this can be obtained by combining the downsizing formula for removing the first row and column with a few simple permutations (see Alg. C.1).

4.6.3 Label update for tracking time-varying mappings

The above described procedure is capable of identifying a static nonlinear mapping, by selecting points to store in memory and performing regression on these points. If the nonlinear mapping changes over time, however, it is likely that the memory contains points that do not reflect the current mapping well. Since regression is performed only on the memory, these invalid points can remain in the memory and affect the algorithm's performance.

On the other hand, it is reasonable to assume that after a number of iterations the input space will be sufficiently well sampled. Since the change in the observed system's response is reflected only on the desired data d_i, we only need to adjust the data d_i stored in the memory in order to achieve tracking capability. We propose to use the following update for all stored data labels, whenever a new input-output point $(\mathbf{x}_n, d_n)$ is received

$$d_i \leftarrow d_i - \mu\kappa(\mathbf{x}_i, \mathbf{x}_n)(d_i - d_n), \quad \forall i, \tag{4.11}$$

where $\mu \in [0, 1]$ is a step-size parameter.

The "relabeling" equation (4.11) takes into account the similarities in input *and* output space, measured respectively by the kernel function and the difference $d_i - d_n$. As a consequence, it only affects points $\mathbf{x}_i$ that are close enough to the new point $\mathbf{x}_n$ in the sense measured by the kernel. Concordantly, the change in the labels will be proportional to $d_i - d_n$. For instance, when using a Gaussian kernel, the kernel $\kappa(\mathbf{x}_i, \mathbf{x}_n)$ will be close to zero if the new point $\mathbf{x}_n$ is far away from a stored point $\mathbf{x}_i$, and the label y_i will not be changed. On the other hand, if the new point $\mathbf{x}_n$ coincides with some stored $\mathbf{x}_i$ and its label d_i is very different from d_n, this label will be changed proportionally to the difference $d_i - d_n$. Notice that if $\mu = 0$ this update has no effect, and the algorithm assumes the observed nonlinear system to be static.

The entire fixed-budget kernel RLS algorithm is summarized in Alg. 4.2. In every iteration, it first adds the new input-output pair $(\mathbf{x}_n, d_n)$ to its memory and subsequently decides which point to prune. The algorithm starts with an empty memory, and during the first M iterations it skips the pruning step. The upsized matrices and

Algorithm 4.2 Fixed-Budget Kernel Recursive Least-Squares (FB-KRLS)

```
initialize
    Store $\{\mathbf{x}_1, d_1\}$ in memory.
    Calculate $\mathbf{K}_1^{-1}$, and $\boldsymbol{\alpha}$ with (4.7).
for $n = 2, 3, \ldots$ do
    Update all stored labels $d_i$ with (4.11).
    Add $(\mathbf{x}_n, d_n)$ to memory and obtain $\breve{\mathbf{K}}_n^{-1}$ with (C.3).
    if memory size $> M$ then
        Determine least significant stored point $(\mathbf{x}_L, d_L)$ with (4.10).
        Prune $(\mathbf{x}_L, d_L)$ from memory and obtain $\mathbf{K}_n^{-1}$ with Alg. C.1.
    end if
    Obtain KRLS solution based on updated memory, with (4.7).
end for
Output $\alpha_i$ and $(\mathbf{x}_i, d_i)$, for all $i$ stored in the memory.
```

vectors are denoted with a breve here, for instance $\breve{\mathbf{K}}_n$. Since the inverse kernel matrix can be updated by Eq. (C.3) and Alg. C.1, which are simple algebraic operations, this algorithm has the same computational complexity as the original sliding-window kernel RLS algorithm.

4.7 Comparison of Kernel-Based RLS Algorithms

In this section we present an experimental comparison of different kernel RLS algorithms. The first experiment analyzes the performance on the identification of a static system, in particular the prediction of a nonlinear time-series. In the second experiment, the tracking capability of the algorithms is compared on a time-varying nonlinear channel.

4.7.1 Prediction of a steady-state nonlinear time-series

We perform one-step prediction on the nonlinear Mackey-Glass time-series with a number of online algorithms. The algorithms are trained online on 500 points of this series, and in each iteration the MSE performance is calculated on a test set of 100 points. A time-embedding of 7 is chosen, i.e. $\mathbf{x}_n = [x_n, \ldots, x_{n-6}]^T$ and the desired output is $y_n = x_{n+1}$. The data are corrupted by zero-mean Gaussian noise with 0.001 variance.

We test the performance of the following RLS-based algorithms.

- Linear RLS.
- Sliding-window kernel RLS (SW-KRLS) with a window of 50 samples.
- ALD-based KRLS (ALD-KRLS) with accuracy parameter $\nu = 0.43$. The accuracy parameter will limit the dictionary growth. Its value is chosen here to obtain a final memory size of approximately 50 samples.

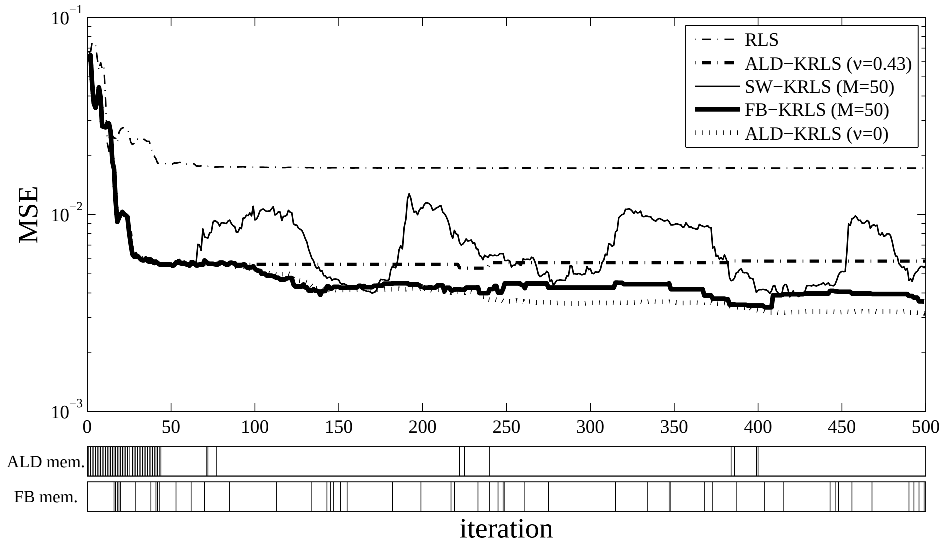


Figure 4.11: Top: Learning curves for one-step prediction on the Mackey-Glass time-series. Bottom: indices of the points stored in memory by ALD-KRLS ($\nu = 0.43$) and FB-KRLS. Note that the final memory of FB-KRLS consists of points selected over the entire time series.

- Fixed-budget kernel RLS (FB-KRLS) with a window of 50 samples.
- ALD-KRLS with accuracy parameter $\nu = 0$. This case is included to provide a lower bound on the performance. With this setting, the dictionary accepts every data point and the resulting computational complexity is $O(n^2)$, where n indicates the current iteration number.

For all kernel-based methods, the Gaussian kernel with $\sigma = 1$ is used, and a regularization term $c = 0.1$ is included.

The learning curves of the different algorithms are shown in Fig. 4.11. It is remarkable that the FB-KRLS technique obtains results that are very close to the lower bound. By setting $\nu = 0.43$, ALD-KRLS stores 53 points in memory, which is similar to FB-KRLS. Nevertheless, in this case ALD-KRLS performs worse. In the lower part of Fig. 4.11, the indices of the points in memory are shown, for ALD-KRLS with ($\nu = 0.43$) and FB-KRLS. Thanks to its pruning mechanism, FB-KRLS is capable of selecting the most significant data points over the entire time series.

4.7.2 Identification of a time-varying nonlinear system

For the second experiment, we consider a nonlinear communications channel composed of a linear filter followed by a static nonlinearity (a Wiener system). As the system input we choose a binary signal, $x_i \in \{-1, +1\}$, and 30dB of white Gaussian noise is added at its output. The binary input signal is time-embedded with 4 taps, resulting in the input space showing 16 clusters. The static nonlinearity is the

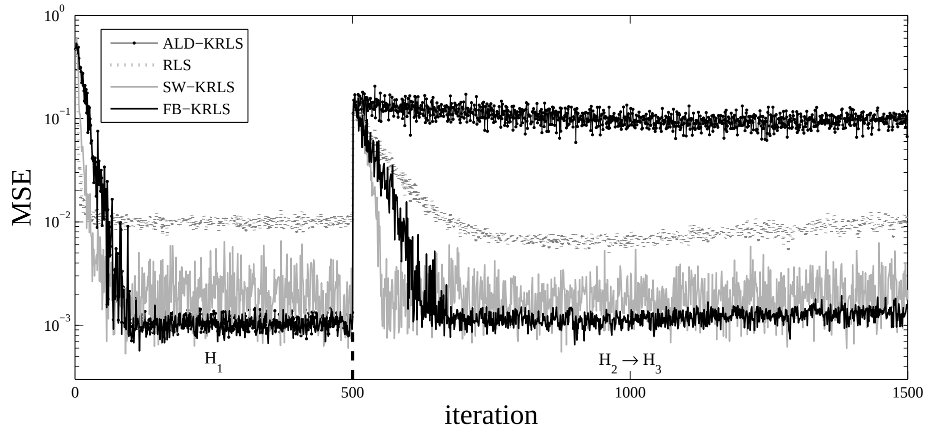


Figure 4.12: Performance on a time-varying Wiener system. In the first zone ("H_1") all algorithms reach optimal steady-state performance, and ALD-KRLS and FB-KRLS coincide. After the abrupt change at iteration 500, only the tracking algorithms are able to recover.

saturation function $f(x) = \tanh(x)$. For the linear filter the following channels are used:

$$\begin{aligned} H_1(z) &= 1 - 0.2663z^{-1} - 0.5541z^{-2} + 0.1420z^{-3} \\ H_2(z) &= 1 + 0.1050z^{-1} - 0.3760z^{-2} - 0.4284z^{-3} \\ H_3(z) &= 1 - 0.4326z^{-1} - 0.1656z^{-2} - 0.3153z^{-3}. \end{aligned}$$

During the first 500 iterations, the linear filter is fixed as $H_1(z)$. On iteration 501, the channel is abruptly switched to $H_2(z)$, which is then changed linearly until becoming $H_3(z)$ on the 1500-th iteration.

The algorithms use the following parameters: A Gaussian kernel with $\sigma = 0.1$ and $\lambda = 0.01$ are chosen for ALD-KRLS and FB-KRLS. Both methods only require 16 points in their dictionary, which is obtained for ALD-KRLS by setting $\nu = 0.1$, and for FB-KRLS by fixing $M = 16$. Note that since the input space shows only 16 clusters, the final dictionary of ALD-KRLS will contain only 16 points even for smaller values of ν. The SW-KRLS algorithm must be given a memory size of $M > 16$ to assure that it contains most of the 16 possible input points. In this experiment we chose $M = 50$ for the SW-KRLS algorithm, and to account for situations in which the memory lacks some point, it uses a Gaussian kernel with $\sigma = 2$, which should cover the entire input space even if points are missing from the memory. Finally, ALD-KRLS and RLS use a forgetting factor $\beta = 0.99$, and for FB-KRLS μ is set to 0.8.

The results, averaged out over 100 Monte Carlo simulations, are shown in Fig. 4.12. During the first 500 iterations, all algorithms reach their steady-state performance. The MSE curves for ALD-KRLS and FB-KRLS overlap here. FB-KRLS obtains better performance than SW-KRLS, since the latter does not actively select significant points. After the abrupt change in the linear channel, the tracking algorithms RLS, SW-KRLS and FB-KRLS are capable of recovering their steady state performance.

Table 4.1: Performance comparison of final MSE values.

Algorithm	MSE	final memory size
ALD-KRLS	0.0951 ± 0.1363	16
RLS	0.0084 ± 0.0099	n/a
SW-KRLS	0.0020 ± 0.0087	50
FB-KRLS	0.0012 ± 0.0018	16

Since the channel is slowly changing, however, their MSE varies slightly over time. On the other hand, ALD-KRLS performs very bad after the channel switch, due to the fact that it is not designed to be a tracking algorithm. The ALD criterion selects its dictionary entirely during the first 500 iterations, and since the input data points $\mathbf{x}_i$ are still the same after the channel switch (only the labels y_i change), it is not capable of adjusting its nonlinear mapping. Table 4.1 displays the MSE averaged out over the last 500 iterations.

4.8 Conclusions

In this chapter we discussed different types of nonlinear systems, in order to design kernel-based identification algorithms. Since nonlinear system identification generally requires models whose number of parameters grows exponentially with the system complexity, we decided to focus on the family of Wiener and Hammerstein system models in the rest of this part of the thesis. These systems provide at the same time a simple and mathematically attractive structure, and they are found in a large number of problems.

The rest of this chapter was dedicated to a class of online kernel-based identification algorithms we recently presented. The main features of these algorithms are the introduction of L_2 regularization against overfitting and a fixed memory size. We introduced efficient matrix inversion formulas to keep the complexity of the problem bounded.

The proposed sliding-window KRLS algorithm is very easy to implement and it is capable of achieving satisfactory performance in both static and time-varying environments. The fixed-budget kernel KRLS algorithm has the same computational complexity but it takes a more active role in its data selection. In static environments it outperforms other kernel-based adaptive filtering algorithms including ALD-KRLS, given similar memory requirements. In a second experiment it was also shown that it is capable of performing tracking, obtaining slightly better results compared with sliding-window KRLS while using less memory.

The publications that have contributed to this chapter are

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. “A sliding-window kernel RLS algorithm and its application to nonlinear channel identification”. In *Proceed-*

ings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). Toulouse, France, 2006.

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. “Nonlinear system identification using a new sliding-window kernel RLS algorithm”. *Journal of Communications*, volume 2, no. 3, pages 1–8, 2007.
- S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe. “Fixed-budget kernel recursive least-squares”. *Submitted to the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

Chapter 5
Supervised Identification of Wiener and Hammerstein Systems

In this chapter we introduce identification and equalization algorithms that exploit the structure of Wiener and Hammerstein systems. In particular, we follow a supervised identification approach that simultaneously identifies both parts of the nonlinear system. Given the correct restrictions on the identification problem, we show how kernel canonical correlation analysis (KCCA) emerges as the logical solution to this problem. We then extend the proposed identification algorithm to an adaptive version that allows to deal with time-varying systems. In order to avoid overfitting we discuss and compare three possible regularization techniques for both the batch and the adaptive versions of the proposed algorithm.

5.1 Problem Statement

Recently, an iterative gradient identification method was presented for Wiener systems that exploits the cascade structure by jointly identifying the linear filter and the inverse nonlinearity [Aschbacher and Rupp, 2005]. It uses a linear estimator $\hat{H}(z)$ and a nonlinear estimator $\hat{g}(\cdot)$, represented by a polynomial, that respectively model the linear filter $H(z)$ and the inverse of the nonlinearity $f(\cdot)$ of the Wiener system[1], as depicted in Fig. 5.1. The estimator models are adjusted by minimizing the error $e[n]$ between their outputs $r_x[n]$ and $r_y[n]$. In the noiseless case, it is possible to find estimators whose outputs correspond exactly to the reference signal $r[n]$ (up to an unknown scaling factor which is inherent to this problem).

In order to avoid the trivial solution, $\hat{H}(z) = 0$ and $\hat{g}(\cdot) = 0$, a constraint should be applied to the solution. For this purpose, it is instructive to look at the expanded form

$$\|\mathbf{e}\|^2 = \|\mathbf{r}_x - \mathbf{r}_y\|^2 = \|\mathbf{r}_x\|^2 + \|\mathbf{r}_y\|^2 - 2\mathbf{r}_x^T\mathbf{r}_y, \tag{5.1}$$

where $\mathbf{e}$, $\mathbf{r}_x$ and $\mathbf{r}_y$ are vectors that contain all elements $e[n]$, $r_x[n]$ and $r_y[n]$, respectively, with $n = 1, \dots, N$.

[1] Note that this approach assumes that the nonlinearity $f(\cdot)$ is invertible in the output data range.

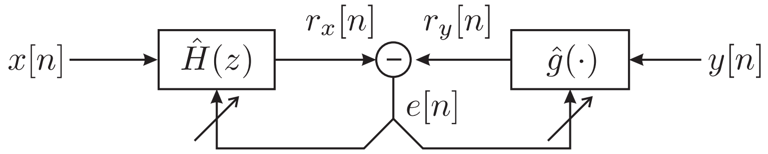


Figure 5.1: The used Wiener system identification diagram.

Typically, a unit norm or a linear constraint is applied on the filter coefficients. In [Aschbacher and Rupp, 2005], a linear restriction was proposed: the first coefficient of the linear filter $\hat{H}(z)$ was fixed to 1, which removes the scaling ambiguity. With the estimated filter represented as $\mathbf{h} = [h_1, \dots, h_L]$ the minimization problem reads

$$\min \|\mathbf{r}_x - \mathbf{r}_y\|^2 \quad \text{s.t.} \quad h_1 = 1. \tag{5.2}$$

However, from (5.1) it is easy to see that there exist situations in which this restriction will not prevent the terms $\|\mathbf{r}_x\|^2$ and $\|\mathbf{r}_y\|^2$ from going to zero. For instance, if a low-pass signal is fed into the system, the cost function (5.2) will not exclude the possibility that the estimated filter $\hat{H}(z)$ exactly cancels out this signal, as could do a high-pass filter. This problem is not limited to this restriction only, but it is inherent to all restrictions applied only on filter coefficients [Vía et al., 2007a]. As a result, such restrictions can lead to noise enhancement problems.

A second and more sensible restriction to minimize (5.1) is to fix the energy of the output signals $\mathbf{r}_x$ and $\mathbf{r}_y$ while maximizing their correlation $\mathbf{r}_x^T\mathbf{r}_y$, which is obtained by solving

$$\min \|\mathbf{r}_x - \mathbf{r}_y\|^2 \quad \text{s.t.} \quad \|\mathbf{r}_x\|^2 = \|\mathbf{r}_y\|^2 = 1. \tag{5.3}$$

Since the norms of $\mathbf{r}_x$ and $\mathbf{r}_y$ are now fixed, the zero-solution is excluded by construction. To illustrate this, a performance comparison between batch identification algorithms based on filter coefficient constraints and this signal energy constraint will be given in section 5.4.1.

5.2 Kernel Canonical Correlation Analysis for Wiener System Identification

We will now construct an identification algorithm based on the proposed signal power constraint (5.3). To represent the linear and nonlinear estimated filters, many different approaches can be used. For instance, the linear part can be represented by finite impulse response (FIR), rational [Wigren, 1994], Laguerre [Wahlberg, 1991], Kautz [Wahlberg, 1994], or linear state-space models [Hagenblad, 1999]. We will represent the linear part of the system as a FIR model. For the nonlinear static part, a number of parametric models can be used, including power series, Chebyshev polynomials, wavelets and piecewise linear (PWL) functions, as well as some nonparametric methods including neural networks [Wigren, 1994, Chen and Chang, 1996, Cousseau et al., 2007]. Nonparametric approaches do not assume that the nonlinearity corresponds to a given model, but rather let the training data decide which characteristic fits them best. We will apply a nonparametric identification approach based on kernel methods. More specifically, the nonlinearity will be represented as a kernel expansion.

5.2.1 Identification algorithm

To identify the linear channel of the Wiener system, we estimate a FIR filter $\mathbf{h} \in \mathbb{R}^L$ whose output is given by

$$r_x[n] = \sum_{i=0}^{L-1} h_i x[n-i] = \mathbf{x}[n]^T \mathbf{h}, \tag{5.4}$$

where $\mathbf{x}[n] = \big[x[n], x[n-1], \cdots, x[n-L+1]\big]^T \in \mathbb{R}^L$ is a time-embedded vector. The nonlinearity can be represented by a kernel expansion $\hat{g}(\cdot) = \sum_{i=1}^{N} \alpha_i \kappa(y[i], \cdot)$, yielding

$$r_y[n] = \hat{g}(y[n]) = \sum_{i=1}^{N} \alpha_i \kappa(y[i], y[n]), \tag{5.5}$$

where α_i are the expansion coefficients and $\kappa(\cdot, \cdot)$ is a suitable kernel function.

Although these representations of the linear and nonlinear part seem to be unrelated, they are in fact both projections, although in different spaces. On the linear side of the system, the input vector $\mathbf{x}[n]$ is projected linearly onto the estimated channel $\hat{\mathbf{h}}$, yielding $r_x[n]$. On the nonlinear side, a linear projection is performed in the feature space, which corresponds to a nonlinear projection in input space: the transformed vector $\tilde{\mathbf{y}}[n]$ is projected onto a vector $\tilde{\mathbf{h}}_y$ in feature space, yielding

$$r_y[n] = \tilde{\mathbf{y}}[n]^T \tilde{\mathbf{h}}_y. \tag{5.6}$$

Since $\tilde{\mathbf{h}}_y$ can be found as the solution to a quadratic minimization problem, the Representer Theorem states that it can also be written as a linear combination of the transformed input patterns, in accordance with (5.5). Specifically, $\tilde{\mathbf{h}}_y$ can be expanded as

$$\tilde{\mathbf{h}}_y = \sum_{i=1}^{N} \alpha_i \tilde{\mathbf{y}}[i], \tag{5.7}$$

which allows to rewrite (5.6) as

$$r_y[n] = \sum_{i=1}^{N} \alpha_i \tilde{\mathbf{y}}[i]^T \tilde{\mathbf{y}}[n] = \sum_{i=1}^{N} \alpha_i \kappa(y[i], y[n]), \tag{5.8}$$

where we applied the kernel trick in the second equality.

To find the optimal linear and nonlinear estimators, it is convenient to formulate (5.3) in terms of matrices. By $\mathbf{X} \in \mathbb{R}^{N \times L}$, we will denote the data matrix containing

$\mathbf{x}[n]$ as rows. The vector containing the corresponding outputs of the linear filter is then obtained as

$$\mathbf{r}_x = \mathbf{X}\mathbf{h}. \tag{5.9}$$

In a similar fashion, the transformed data points $\tilde{\mathbf{y}}[n]$ can be stacked as rows of the transformed data matrix $\tilde{\mathbf{Y}} \in \mathbb{R}^{N\times m'}$. The vector containing all outputs of the nonlinear estimator is

$$\mathbf{r}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{h}}_y. \tag{5.10}$$

Using (5.8), this can be rewritten as

$$\mathbf{r}_y = \mathbf{K}_y\boldsymbol{\alpha}, \tag{5.11}$$

where $\mathbf{K}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T$ is the kernel matrix and $\boldsymbol{\alpha}$ contains the coefficients of the kernel expansion $\tilde{\mathbf{h}}_y = \tilde{\mathbf{Y}}^T\boldsymbol{\alpha}$. With the obtained data representation, the minimization problem (5.3) is rewritten as minimizing

$$\min \|\mathbf{X}\mathbf{h} - \mathbf{K}_y\boldsymbol{\alpha}\|^2 \quad \text{s.t.} \quad \|\mathbf{X}\mathbf{h}\|^2 = \|\mathbf{K}_y\boldsymbol{\alpha}\|^2 = 1. \tag{5.12}$$

This problem is a particular case of kernel canonical correlation analysis (KCCA or kernel CCA) [Lai and Fyfe, 2000, Bach and Jordan, 2002, Hardoon et al., 2003] in which one of the used kernels is linear. In general, it has been proved [Hardoon et al., 2003] that minimizing (5.12) is equivalent to maximizing

$$\rho = \max \frac{\mathbf{r}_x^T\mathbf{r}_y}{\|\mathbf{r}_x\|\|\mathbf{r}_y\|} = \max_{\mathbf{h},\boldsymbol{\alpha}} \frac{\mathbf{h}^T\mathbf{X}^T\mathbf{K}_y\boldsymbol{\alpha}}{\sqrt{\mathbf{h}^T\mathbf{X}^T\mathbf{X}\mathbf{h}\boldsymbol{\alpha}^T\mathbf{K}_y^T\mathbf{K}_y\boldsymbol{\alpha}}}, \tag{5.13}$$

which is the problem of maximizing the correlation between the projected signals $\mathbf{X}\mathbf{h}$ and $\mathbf{K}_y\boldsymbol{\alpha}$. If both kernels were linear, this problem would reduce to standard canonical correlation analysis (CCA), which is an established statistical technique to find linear relationships between two data sets [Hotelling, 1936]. A brief introduction to CCA can be found in appendix D.

The minimization problem (5.12) can be solved by the method of Lagrange multipliers, yielding the following generalized eigenvalue (GEV) problem [Hardoon et al., 2003, Van Vaerenbergh et al., 2006a]

$$\frac{1}{2}\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{K}_y \\ \mathbf{K}_y^T\mathbf{X} & \mathbf{K}_y^T\mathbf{K}_y\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \boldsymbol{\alpha}\end{bmatrix} = \beta\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_y^T\mathbf{K}_y\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \boldsymbol{\alpha}\end{bmatrix}, \tag{5.14}$$

where $\beta = (\rho+1)/2$ is a parameter related to a principal component analysis (PCA) interpretation of CCA [Vía et al., 2005a]. In practice, it is sufficient to solve the slightly less complex GEV

$$\frac{1}{2}\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{K}_y \\ \mathbf{X} & \mathbf{K}_y\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \boldsymbol{\alpha}\end{bmatrix} = \beta\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_y\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \boldsymbol{\alpha}\end{bmatrix}, \tag{5.15}$$

since, as can be easily verified, the GEV problem (5.15) is transformed into (5.14) by pre-multiplication with a block-diagonal matrix containing the unit matrix and $\mathbf{K}_y^T$. Hence, any pair $(\mathbf{h}, \boldsymbol{\alpha})$ that solves (5.15) will also be a solution of (5.14).

The solution of the KCCA problem is given by the eigenvector corresponding to the largest eigenvalue of the GEV (5.15). However, if $\mathbf{K}_y$ is invertible, it is easy to see from (5.12) that for *each* $\mathbf{h}$ satisfying $\|\mathbf{Xh}\|^2 = 1$, there exists an $\boldsymbol{\alpha} = \mathbf{K}_y^{-1}\mathbf{Xh}$ that solves the minimization problem and, therefore, also the GEV problem (5.15). This is a consequence of overfitting, which occurs for sufficiently "rich" kernel functions, i.e. kernels that correspond to feature spaces whose dimension m' is much higher than the number of available data points N. For instance, in case the Gaussian kernel is used, the feature space will have dimension $m' = \infty$. With N unknown coefficients α_i and $m' \gg N$ degrees of freedom, Eq. (5.15) potentially suffers from an overfitting problem. Therefore, the GEV should be regularized.

5.2.2 Regularization techniques

In section 2.3, three basic regularization techniques were discussed. Here, we will show how each of them can be applied to regularize the GEV (5.15).

L_2 regularization

Ridge regression is a commonly applied regularization technique in kernel CCA [Bach and Jordan, 2002, Suykens et al., 2002, Hardoon et al., 2003, Lai and Fyfe, 2000]. It consists in adding a constraint on the L_2 norm of the solution $\tilde{\mathbf{h}}_y$, after which the restriction (5.12) becomes $\|\mathbf{K}_y\boldsymbol{\alpha}\|^2 + c\|\tilde{\mathbf{h}}_y\|^2 = 1$, where c is regularization constant. In appendix D.3 we discuss a few different GEVs that obtain the solution of this L_2-regularized problem. Here, we will work with the GEV (D.18) that is obtained by introducing regularization only in the right-hand side of the GEV:

$$\frac{1}{2}\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{K}_y \\ \mathbf{X} & \mathbf{K}_y\end{bmatrix}\begin{bmatrix}\mathbf{h} \\ \boldsymbol{\alpha}\end{bmatrix} = \beta\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_y^{reg}\end{bmatrix}\begin{bmatrix}\mathbf{h} \\ \boldsymbol{\alpha}\end{bmatrix}. \tag{5.16}$$

Sparsification of the solution

In the sparsification approach, the solution is represented in terms of a dictionary of M significant points, which can be obtained for instance in an online manner by the ALD criterion [Engel et al., 2004]. Once a dictionary of points is found according to a reasonable criterion, the complete set of data points $\tilde{\mathbf{Y}}$ can be expressed in terms of the transformed dictionary as $\tilde{\mathbf{Y}} \approx \mathbf{A}\tilde{\mathbf{D}}$, where $\mathbf{A} \in \mathbb{R}^{N\times M}$ contains the coefficients of these approximate linear combinations, and $\tilde{\mathbf{D}} \in \mathbb{R}^{M\times m'}$ contains the dictionary points row-wise. This also reduces the expansion coefficients vector to $\breve{\boldsymbol{\alpha}} = \mathbf{A}^T\boldsymbol{\alpha}$, which now contains M elements. Introducing the reduced kernel matrix $\breve{\mathbf{K}}_y = \tilde{\mathbf{D}}\tilde{\mathbf{D}}^T$, the following approximation can be made:

$$\mathbf{K}_y = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T \approx \mathbf{A}\breve{\mathbf{K}}_y\mathbf{A}^T. \tag{5.17}$$

Substituting $\mathbf{K}_y$ by $\mathbf{A}\breve{\mathbf{K}}_y\mathbf{A}^T$ in the minimization problem (5.12) leads to the GEV

$$\frac{1}{2}\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{A}\breve{\mathbf{K}}_y \\ \mathbf{A}^T\mathbf{X} & \mathbf{A}^T\mathbf{A}\breve{\mathbf{K}}_y\end{bmatrix}\begin{bmatrix}\mathbf{h} \\ \breve{\boldsymbol{\alpha}}\end{bmatrix} = \beta\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^T\mathbf{A}\breve{\mathbf{K}}_y\end{bmatrix}\begin{bmatrix}\mathbf{h} \\ \breve{\boldsymbol{\alpha}}\end{bmatrix}. \tag{5.18}$$

In section 5.3 we will adopt this sparsification procedure to design a regularized adaptive version KCCA algorithm.

Low-dimensional approximation

Kernel PCA can be applied to obtain a low-rank approximation of the kernel matrix

$$\mathbf{K}_y \approx \mathbf{V\Sigma V}^T, \tag{5.19}$$

where $\mathbf{\Sigma}$ is a diagonal matrix containing the M largest eigenvalues s_i and $\mathbf{V}$ contains the corresponding eigenvectors $\mathbf{v}_i$ column-wise. Introducing $\bar{\boldsymbol{\alpha}} = \mathbf{V}^T\boldsymbol{\alpha}$ as the projection of $\boldsymbol{\alpha}$ onto the M-dimensional subspace spanned by the eigenvectors $\mathbf{v}_i$, the GEV problem (5.15) reduces to

$$\frac{1}{2}\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{X}^T\mathbf{V\Sigma}\\ \mathbf{V}^T\mathbf{X} & \mathbf{\Sigma}\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \bar{\boldsymbol{\alpha}}\end{bmatrix} = \beta\begin{bmatrix}\mathbf{X}^T\mathbf{X} & \mathbf{0}\\ \mathbf{0} & \mathbf{\Sigma}\end{bmatrix}\begin{bmatrix}\mathbf{h}\\ \bar{\boldsymbol{\alpha}}\end{bmatrix}, \tag{5.20}$$

where we have exploited the fact that $\mathbf{V}^T\mathbf{V} = \mathbf{I}_M$.

5.2.3 Unifying Wiener and Hammerstein system identification and equalization

To identify the linear channel and the inverse nonlinearity of the Wiener system, any of the regularized GEV problems (5.16), (5.20), or (5.18) can be solved. Moreover, given the symmetric structure of the Wiener and Hammerstein systems (see figures 4.1 and 4.2), it should be clear that the same approach can be applied to identify the blocks of the Hammerstein system. To do so, the linear and nonlinear estimators of the proposed kernel CCA algorithm need to be switched. The obtained Hammerstein system identification algorithm estimates the direct static nonlinearity and the inverse linear channel, which is retrieved as a FIR filter.

Full identification of an unknown system provides an estimate of the system output given a certain input signal. To fully identify the Wiener system, the presented KCCA algorithm needs to be complemented with an estimate of the direct nonlinearity $f(\cdot)$. This nonlinearity can be obtained by applying any nonlinear regression algorithm on the signal in between the two blocks (whose estimate is provided by the KCCA-based algorithm) and the given output signal $\mathbf{y}$. In particular, to stay within the scope of this work, we propose to obtain $\hat{f}(\cdot)$ as another kernel expansion

$$\hat{f}(\cdot) = \sum_{n=1}^{N} \beta_n \kappa(\cdot, r[n]), \tag{5.21}$$

where $r[n] = (r_x[n] + r_y[n])/2$. In section 5.4 the full identification process is illustrated with some examples.

Apart from Wiener system identification, a number of algorithms can be based directly on the presented KCCA algorithm. In case of the Hammerstein system, KCCA readily obtains an estimate of the direct nonlinearity and the inverse linear channel. To fully identify the Hammerstein system, the direct linear channel needs to be estimated, which can be done by applying standard filter inversion techniques [Haykin, 2001].

At this point, it is interesting to note that the inversion of the estimated linear filter can also be used to equalize a Wiener system (see [Van Vaerenbergh et al., 2006a]) when combined with an estimate of the inverse nonlinearity, which is obtained by the KCCA algorithm obtains in the first place. To come full circle, a Hammerstein system equalization algorithm can be constructed based on the inverse linear channel estimated by KCCA and the inverse nonlinearity that can be obtained by performing nonlinear regression on the appropriate signals.

The main advantage of these techniques compared to black-box methods is that by exploiting the specific model of the nonlinear system, the solution has less degrees of freedom and will likely be more accurate. Specifically, if a Wiener system is identified in a black-box manner by kernel LS regression as $f_{bb} : \mathbb{R}^L \rightarrow \mathbb{R}$, the entire L-dimensional input space range must be represented in the training data, requiring a large number of input points. By exploiting the system structure, on the other hand, the inverse nonlinearity $g : \mathbb{R} \rightarrow \mathbb{R}$ is a one-dimensional function whose estimation requires far less data, and the linear filter is represented only by L additional coefficients.

5.3 Adaptive Solution

As discussed in chapter 3, in a number of situations it is desirable to have an adaptive algorithm that can update its solution according to newly arriving data. Standard scenarios include problems where the amount of data is too high to apply a batch algorithm, and environments that change over time. In this section, we discuss an adaptive version of kernel CCA that can be used for online identification of Wiener and Hammerstein systems.

5.3.1 Formulation of KCCA as coupled RLS problems

The special structure of the GEV problem (5.15) has recently been exploited to obtain efficient CCA and KCCA algorithms [Vía et al., 2007b, Pezeshki et al., 2005, Van Vaerenbergh et al., 2006a]. Specifically, the GEV problem in CCA can be viewed as two coupled least-squares regression problems, which, in case of kernel CCA, become

$$\begin{cases} \beta \mathbf{h} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\mathbf{r}} \\ \beta \mathbf{K}_y \boldsymbol{\alpha} = \hat{\mathbf{r}}, \end{cases} \tag{5.22}$$

where $\hat{\mathbf{r}} = (\mathbf{r}_x + \mathbf{r}_y)/2 = (\mathbf{X}\mathbf{h} + \mathbf{K}_y\boldsymbol{\alpha})/2$. This idea has been used in [Vía et al., 2005b, Van Vaerenbergh et al., 2006a] to develop an algorithm based on the solution of these regression problems iteratively: at each iteration t two LS regression problems are solved using

$$\hat{\mathbf{r}}(t) = \frac{\mathbf{r}_x(t) + \mathbf{r}_y(t)}{2} = \frac{\mathbf{X}\mathbf{h}(t-1) + \mathbf{K}_y\boldsymbol{\alpha}(t-1)}{2} \tag{5.23}$$

as desired output.

Furthermore, this LS regression framework was exploited directly to develop an adaptive CCA algorithm based on the recursive least-squares algorithm (RLS), which was shown to converge to the CCA solution [Vía et al., 2005b]. For Wiener and Hammerstein system identification, the adaptive solution of (5.22) can be obtained by coupling one *linear* RLS algorithm with one *kernel RLS* algorithm. To regularize this kernel RLS algorithm, any of the three discussed regularization techniques can be used (see section 3.2). We will illustrate each of them with a suitable kernel RLS algorithm, and show how they fit into the KCCA framework. Although we will discuss only the three kernel RLS methods considered in [Van Vaerenbergh et al., 2008a], the procedure is identical for any other kernel RLS algorithm.

Note that the algorithm adaptively updates its estimation of both parts of the system, and therefore shortly after initialization the signal $\hat{r}[n]$ will not yet represent an accurate estimate of the real intermediate signal $r[n]$. Hence, the linear and kernel RLS algorithm used in training must be capable of forgetting the influence of this initially erroneous estimate. For the linear part, this is solved by simply using an RLS algorithm with a forgetting factor. The nonlinear part requires a kernel RLS algorithm with tracker capabilities, which is less obvious to implement, as was seen in chapter 3. We will discuss this further in the experiments.

L_2 regularization: sliding-window kernel RLS

The sliding-window method of section 4.4 creates buffers that capture the last N input points, $\mathbf{y} = [y[n], \ldots, y[n-N+1]]^T$, and the last estimated points of the reference signal, $\hat{\mathbf{r}} = [\hat{r}[n], \ldots, \hat{r}[n-N+1]]^T$, which are used as the method's output[2]. In feature space, the input data $\mathbf{y}$ corresponds to a transformed data matrix $\tilde{\mathbf{Y}}$, which is used to calculate the regularized kernel matrix $\mathbf{K}_y^{reg} = \tilde{\mathbf{Y}}\tilde{\mathbf{Y}}^T + c\mathbf{I}$. This allows to solve the LS regression problem

$$\boldsymbol{\alpha} = \left(\mathbf{K}_y^{reg}\right)^{-1}\hat{\mathbf{r}}. \tag{5.24}$$

The sliding-window approach provides this kernel RLS algorithm with the capability to recalculate the solution $\boldsymbol{\alpha}$ as new input-output pairs $\{y[n], \hat{r}[n]\}$ are received. Since it discards older input-output pairs, it is capable to operate as a tracker. At any moment the estimated output r_y corresponding to a new input point y can be calculated, for a given estimate of $\boldsymbol{\alpha}$, as

$$r_y = \sum_{n=1}^{N} \alpha_n \tilde{\mathbf{y}}[n]\tilde{\mathbf{y}} = \sum_{n=1}^{N} \alpha[n]\kappa(y[n], y) = \mathbf{k}_{\mathbf{y}}^T\boldsymbol{\alpha}, \tag{5.25}$$

[2] As was mentioned in the previous chapter, the sliding-window kernel RLS algorithm shows better tracker capabilities than the fixed-budget algorithm. Therefore, it is preferred in this kernel CCA framework.

where $\mathbf{k_y}$ is a vector containing the elements $\kappa(y[n], y)$ and $y[n]$ corresponds to the points in the input data buffer. This allows to obtain the identification error of the algorithm.

When this algorithm is used as the kernel RLS algorithm in the adaptive kernel CCA framework for Wiener system identification, the coupled LS regression problems (5.22) become

$$\begin{cases} \beta\mathbf{h} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\mathbf{r}} \\ \beta\boldsymbol{\alpha} = \left(\mathbf{K}_y^{reg}\right)^{-1}\hat{\mathbf{r}}. \end{cases} \tag{5.26}$$

Note that the more recent fixed-budget kernel RLS algorithm could also be used in this framework. Nevertheless, since the main idea in this chapter is to perform regression on the unknown reference signal $\hat{r}[n]$, any nonlinear RLS algorithm can be used. Therefore, the L_2-regularized method chosen in this chapter will be the sliding-window kernel RLS algorithm, which was also used in the original study.

Sequential sparsification: ALD-KRLS

The dictionary-based kernel RLS algorithm recursively obtains the solution to the LS problem by efficiently solving

$$\breve{\boldsymbol{\alpha}} = (\mathbf{A}\breve{\mathbf{K}}_y)^{\dagger}\mathbf{r}_y = \breve{\mathbf{K}}_y^{-1}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{r}_y, \tag{5.27}$$

where $\mathbf{r}_y$ contains all output data and $\breve{\mathbf{K}}_y$ is the reduced kernel matrix from (5.17). After plugging this kernel RLS algorithm into (5.22), the coupled LS regression problems become

$$\begin{cases} \beta\mathbf{h} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\mathbf{r}} \\ \beta\breve{\boldsymbol{\alpha}} = \breve{\mathbf{K}}_y^{-1}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\hat{\mathbf{r}}. \end{cases} \tag{5.28}$$

Given an estimate of $\breve{\boldsymbol{\alpha}}$, the estimated output r_y corresponding to a new input point y can be calculated as

$$r_y = \sum_{i=1}^{M} \breve{\alpha}_i \kappa(c[i], y) = \mathbf{k}_{\mathbf{cy}}^T \breve{\boldsymbol{\alpha}}, \tag{5.29}$$

where $\mathbf{k_{cy}}$ contains the kernel functions of the points $c[i]$ in the dictionary and the data point y.

Notice that the ALD criterion is not suitable for time-varying environments since it cannot fully exclude the influence of already stored data points. Therefore, if the estimate of the reference signal $\hat{r}$ is unreliable at a certain moment in time, the performance of this algorithm will be affected at all later instants.

Low-rank approximation: online kernel PCA-based RLS

The online kernel PCA algorithm from [Hoegaerts et al., 2007] (see section 3.2) can be used to approximate the second LS regression problem from (5.22), leading to the following set of coupled problems:

$$\begin{cases} \beta\mathbf{h} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\hat{\mathbf{r}} \\ \beta\bar{\boldsymbol{\alpha}} = \boldsymbol{\Sigma}^{-1}\mathbf{V}^T\hat{\mathbf{r}}, \end{cases} \tag{5.30}$$

Algorithm 5.1 Adaptive Kernel CCA for Wiener System Identification

```
initialize
    Initialize the RLS and KRLS algorithm.
for $n = 1, 2, \dots$ do
    Receive $\{\mathbf{x}[n], y[n]\}$.
    Compute the outputs of RLS and KRLS, $r_x[n]$ and $r_y[n]$.
    Estimated reference signal: $\hat{r}[n] = (r_x[n] + r_y[n])/2$.
    Use the input-output pairs $\{\mathbf{x}[n], \hat{r}[n]\}$ and $\{y[n], \hat{r}[n]\}$ to update the RLS and
      KRLS solutions $\mathbf{h}$ and $\boldsymbol{\alpha}$.
    Normalize the solutions with $\beta = \|\mathbf{h}\|$, i.e. $\mathbf{h} \leftarrow \mathbf{h}/\beta$ and $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha}/\beta$.
end for
```

where we used the notations of (5.19). Furthermore, the estimated output r_y of the nonlinear filter corresponding to a new input point y is calculated by this algorithm as

$$r_y = \sum_{i=1}^{N} \sum_{j=1}^{M} \kappa(y_i, y) V_{ij} \bar{\alpha}_i = \mathbf{k}_{\mathbf{y}}^T \mathbf{V} \bar{\boldsymbol{\alpha}}, \tag{5.31}$$

where V_{ij} denotes the i-th element of the eigenvector $\mathbf{v}_j$.

5.3.2 Adaptive identification algorithm

The entire adaptive kernel CCA algorithm is summarized in Alg. 5.1. It couples a linear RLS algorithm and a kernel RLS algorithm, as in (5.22). Depending on the used type of regularization, the system of coupled equations it solves can take the form of (5.26), (5.28) or (5.30). Notice the normalization step at the end of each iteration, which fixes the scaling factor of the solution.

5.4 Experiments

In this section, we reproduce some experiments from [Van Vaerenbergh et al., 2008a] in which the proposed kernel CCA algorithms were experimentally tested. We begin by comparing three algorithms based on different error minimization constraints, in a batch experiment. Next, we conduct a series of online identification tests including a static Wiener system, a time-varying Wiener system, and a static Hammerstein system.

To compare the performance of the used algorithms, two different MSE values can be analyzed. First, the kernel CCA algorithms' success can be measured directly by comparing the estimated signal $\hat{r}$ to the real internal signal r of the system, resulting in the error $e_r = r - \hat{r}$. Second, as shown in section 5.2.3, the proposed KCCA algorithms can be extended to perform full system identification and equalization. In that case, the full system identification error e_y is obtained as the difference between estimated system output and real system output, $e_y = y - \hat{y}$.

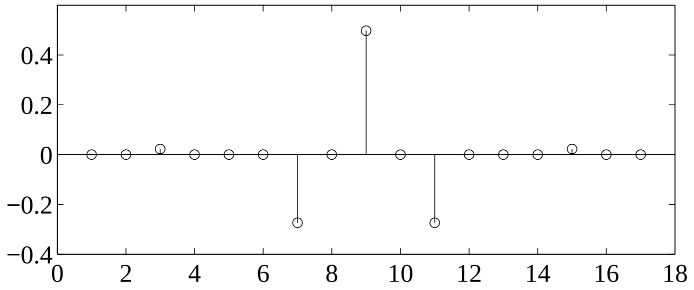


Figure 5.2: The 17 taps bandpass filter used as the linear channel in the Wiener system, generated in Matlab as `fir1(16,[0.25,0.75])`.

The input signal for all experiments consists of a Gaussian with distribution $\mathcal{N}(0,1)$, and to the output of the Wiener or Hammerstein system additive zero-mean white Gaussian noise is added. Two different linear channels and two different nonlinearities are used. The exact setup is specified in each experiment, and the length of the linear filter is supposed to be known in all cases. In [Van Vaerenbergh et al., 2006a] it was shown that the performance of the kernel CCA algorithm for Wiener identification is hardly affected by overestimation of the linear channel length. Therefore, if the exact filter length is not known, it can be overestimated without significant performance loss.

5.4.1 Batch identification

In the first experiment, we compare the performance of the different constraints to minimize the error $\|\mathbf{r}_x - \mathbf{r}_y\|^2$ between the linear and nonlinear estimates in the simultaneous identification scheme from section 5.2. The identification of a static Wiener system is treated here as a batch problem, i.e. all data points are available beforehand.

The Wiener system used for this setup consists of the static linear channel from [Aschbacher and Rupp, 2005] representing a FIR bandpass filter of 17 taps (see Figure 5.2) and a static nonlinearity given by $f(x) = 0.2x + \tanh(x)$. 500 samples are used to identify this system. To represent the inverse nonlinearity, a kernel expansion is used, based on a Gaussian kernel with kernel size $\sigma = 0.2$. In order to avoid overfitting of the kernel matrix, L_2 regularization is applied by adding a constant $c = 10^{-4}$ to its diagonal.

Three different identification approaches are applied, using different constraints to minimize the error $\|\mathbf{e}\|^2$. As discussed in section 5.1, these constraints can be based on the filter coefficients or the signal energy. In a first approach, we apply the filter coefficient norm constraint (5.2), which fixes $h_1 = 1$. The corresponding optimal estimators are found by solving a simple LS problem. If, instead, we fix the filter norm $\|\mathbf{h}\|^2 + \|\boldsymbol{\alpha}\|^2 = 1$, we obtain the following problem

$$\min \|\mathbf{r}_x - \mathbf{r}_y\|^2 \quad \text{s.t.} \quad \|\mathbf{h}\|^2 + \|\boldsymbol{\alpha}\|^2 = 1, \tag{5.32}$$

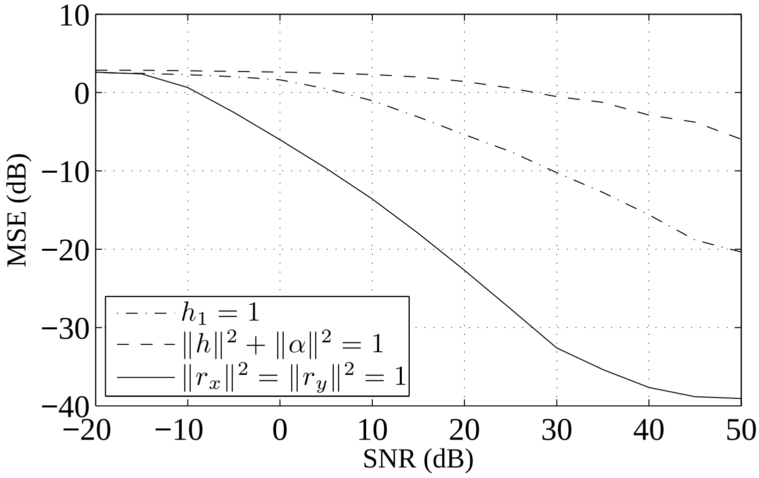


Figure 5.3: MSE $\|\mathbf{e}_r\|^2$ on the Wiener system's internal signal. The algorithms based on filter coefficient constraints (dotted and dashed lines) perform worse than the proposed KCCA algorithm (solid line), which is based on a signal power constraint.

which, after introducing the substitutions $\mathbf{L} = \left[\mathbf{X}, -\mathbf{K}_y\right]$ and $\mathbf{v} = \left[\mathbf{h}^T, \boldsymbol{\alpha}^T\right]^T$, becomes

$$\min \|\mathbf{L}\mathbf{v}\|^2 = \min \|\mathbf{v}^T\mathbf{L}^T\mathbf{L}\mathbf{v}\| \quad \text{s.t.} \quad \|\mathbf{v}\|^2 = 1. \tag{5.33}$$

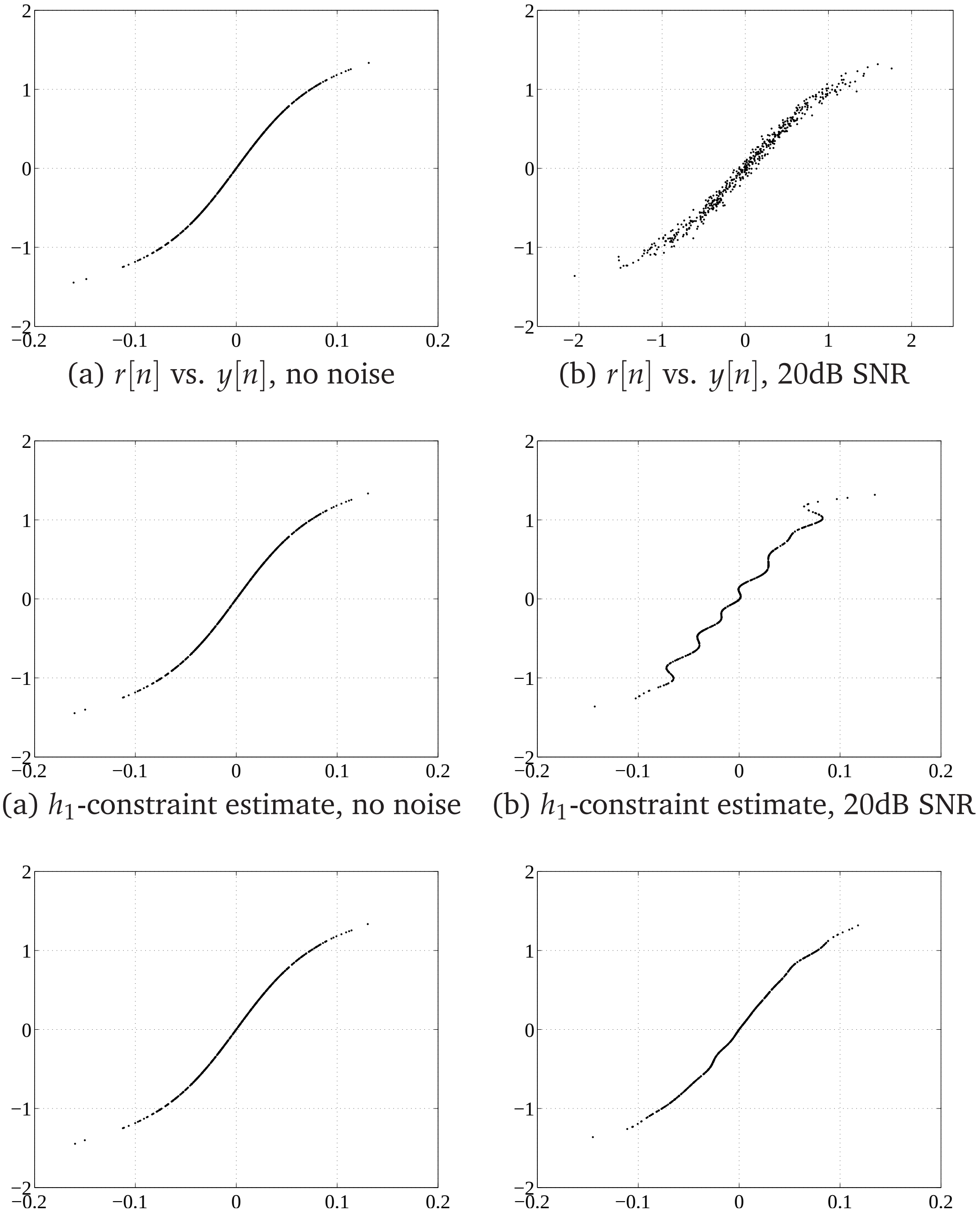
The solution $\mathbf{v}$ of this second approach is found as the eigenvector corresponding to the smallest eigenvalue of the matrix $\mathbf{L}^T\mathbf{L}$. As a third approach, we apply the signal energy-based constraint (5.3), which fixes $\|\mathbf{r}_x\|^2 = \|\mathbf{r}_y\|^2 = 1$. The corresponding solution is obtained by solving the GEV (5.16).

In Fig. 5.3, the performance results are shown for the three approaches and for different noise levels. To calculate the error $\mathbf{e}_r = \mathbf{r} - \hat{\mathbf{r}}$, both $\mathbf{r}$ and $\hat{\mathbf{r}}$ have been normalized to compensate for the scaling indeterminacy of the Wiener system. The MSE is obtained by averaging out $\|\mathbf{e}_r\|^2$ over 250 runs of the algorithms. As can be observed, the algorithms based on the filter coefficient constraints perform clearly worse than the proposed KCCA algorithm.

Figure 5.4 compares the real inverse nonlinearity with the estimate of this nonlinearity for the solution based on the h_1 filter coefficient constraint and to the estimate obtained by regularized KCCA. For 20dB of output noise, the results of the first algorithm are dominated by noise enhancement problems (Figure 5.4 (d)). This further illustrates the advantage of the signal power constraint over the filter coefficient constraint.

In the second experiment, we compare the full Wiener system identification results for the KCCA approach with two black-box neural network methods, specifically a radial basis function (RBF) network and a multilayer perceptron (MLP). The Wiener system setup and used input signal are the same as in the previous experiment.

For a fair comparison, the used solution methods should have similar complexity. Since complexity comparison is difficult due to the significant architectural differences between kernel and classical neural network approaches



(a) $r[n]$ vs. $y[n]$, no noise

(b) $r[n]$ vs. $y[n]$, 20dB SNR

(a) h_1-constraint estimate, no noise

(b) h_1-constraint estimate, 20dB SNR

(a) KCCA estimate, no noise

(b) KCCA estimate, 20dB SNR

Figure 5.4: Estimates of the nonlinearity in the static Wiener system for different constraints on the solution. The top row shows the true signal $r[n]$ versus the points $y[n]$ representing the system nonlinearity, for a noiseless case in (a) and a system that has 20dB white Gaussian noise at its output, in (b). The second and third row show $r_y[n]$ versus $y[n]$ obtained by applying the filter coefficient constraint $h_1 = 1$ and the signal power constraint from KCCA, respectively.

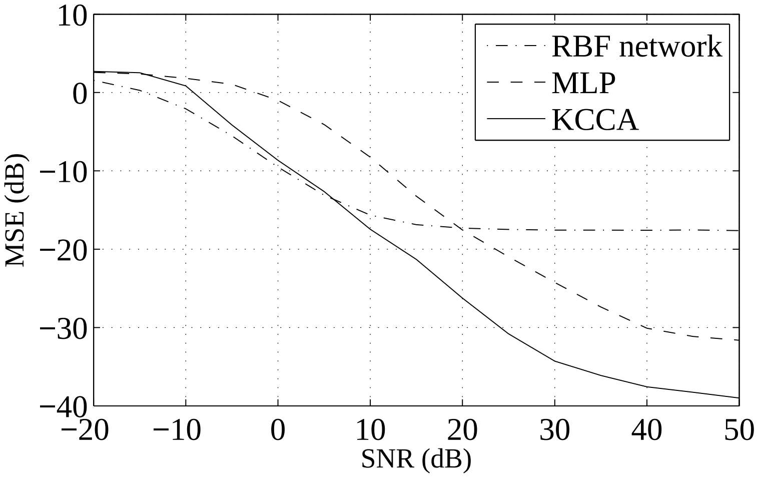


Figure 5.5: Full identification MSE $||\mathbf{e}_y||^2$ of the Wiener system, using two black-box methods (RBF network and MLP) and the proposed KCCA algorithm. All three methods had a similar complexity (in terms of the number of parameters).

[Schölkopf and Smola, 2002], we compare the identification methods when simply given a similar number of parameters. The KCCA algorithm requires 17 parameters to identify the linear channel and 500 parameters in its kernel expansion, totalling 517. When the RBF network and the MLP have 27 neurons in their hidden layer, they obtain a comparable total of 514 parameters, considering they use a time-delay input of length 17. For the MLP, however, better results were obtained by lowering its number of neurons, and therefore, we only assigned it 15 neurons. The RBF network was trained with a sum-squared error goal of 10^{-6} and the Gaussian function of its centers had a spread of 10. The MLP used a hyperbolic tangent transfer function, and it was trained over 50 epochs with the Levenberg-Marquardt algorithm.

The results of the batch identification experiment can be seen in Fig. 5.5. The KCCA algorithm performs best due to its knowledge of the internal structure of the system. Note that by choosing the hyperbolic tangent function as the transfer function, the MLP's structure closely resembles the used Wiener system and, therefore, also obtains good performance.

5.4.2 Online identification

In a second set of simulations, we compare the identification performance of the three adaptive kernel CCA-based identification algorithms from section 5.3. In all online experiments the optimal parameters as well as the kernel functions were determined by an exhaustive search, and the fact that the different algorithms should have comparable computational complexity was taken into account.

Static Wiener system identification

The Wiener system used in this experiment contained the same linear channel as in the previous batch example, followed by the nonlinearity $f(x) = \tanh(x)$. No output noise was added in this first setup.

We applied the three proposed adaptive kernel CCA-based algorithms with the following parameters:

- kernel CCA with standard regularization, $c = 10^{-3}$, and a sliding window of 150 samples, using the Gaussian kernel function with kernel width $\sigma = 0.2$;
- kernel CCA with ALD-based sparsification from [Engel et al., 2004], with a polynomial kernel function of order 3 and accuracy parameter $\nu = 10^{-4}$. This parameter controls the level of sparsity of the solution;
- kernel CCA based on kernel PCA using 15 eigenvectors calculated from a 150-sample sliding window, applying the polynomial kernel function of order 3.

The linear RLS algorithm used in all three cases was a standard exponentially weighted RLS algorithm [Haykin, 2001] with a forgetting factor of 0.99.

The obtained MSE $e_r^2[n]$ for the three algorithms can be seen in Fig. 5.6. Most notable is the slow convergence of the dictionary-based kernel CCA implementation. This is explained by the fact that the ALD criterion used in the dictionary-based kernel RLS algorithm from [Engel et al., 2004] is lacking a forgetting mechanism and, therefore, the influence of the initially erroneous reference signal $\hat{r}$ takes a large number of iterations to decrease. The kernel PCA-based algorithm obtained its optimal performance for a polynomial kernel, while the standard regularized kernel CCA algorithm performs slightly better, with the Gaussian kernel.

A comparison of the results of the sliding-window KCCA algorithm for different noise levels is given in Fig. 5.7. A different Wiener system was used, with linear channel

$$H(z) = 1 + 0.3668z^{-1} - 0.5764z^{-2} + 0.2070z^{-3},$$

and nonlinearity $f(x) = \tanh(x)$.

Figure 5.8 shows the full online system identification results obtained by an MLP and the proposed KCCA algorithm on this Wiener system. The used MLP has learning rate 0.01 and was trained at each iteration step with the new data point. The KCCA algorithm again has L_2 regularization with $c = 10^{-3}$, $\sigma = 0.2$, and a sliding window of 150 samples. Both the inverse nonlinearity and the direct nonlinearity were estimated with the sliding-window kernel RLS technique. Although this algorithm converges slower, it is clear that its knowledge of the internal structure of the Wiener system implies a considerable advantage over the black-box approach.

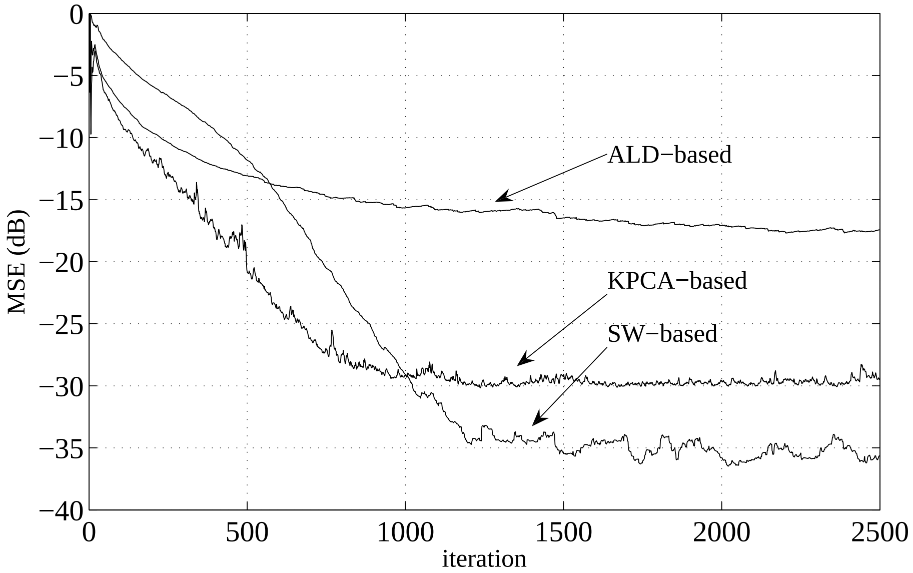


Figure 5.6: MSE $e_r^2[n]$ on the Wiener system's internal signal $r[n]$ for adaptive kernel CCA-based identification of a static noiseless Wiener system.

Dynamic Wiener system identification

In a second experiment, the tracking capabilities of the discussed algorithms were tested. Therefore, an abrupt change in the Wiener system was triggered[3]: During the first part, the Wiener system uses the 17-coefficient channel from the previous tests, but after receiving the 1000-th data point, its channel is changed to

$$H(z) = 1 + 0.3668z^{-1} - 0.5764z^{-2} + 0.2070z^{-3}.$$

The nonlinearity was $f(x) = \tanh(x)$ in both cases. Moreover, 20dB of zero-mean white Gaussian noise was added to the output of the system during the entire experiment.

The parameters of the applied identification algorithms were chosen as follows.

- for kernel CCA with standard regularization, we used $c = 10^{-3}$, a sliding window of 150 samples and the polynomial kernel of order 3;
- for kernel CCA with ALD-based sparsification, we used accuracy parameter $\nu = 10^{-3}$ and a polynomial kernel of order 3;
- the kernel CCA algorithm based on kernel PCA was used with 15 eigenvectors, a sliding window of 150 samples, and the polynomial kernel of order 3.

The length of the estimated linear channel was fixed as 17 during this experiment, resulting in an overestimated channel estimate in the second part.

[3] Note that, although only the linear filter is changed, the proposed adaptive identification method allows both parts of the Wiener system to be varying in time.

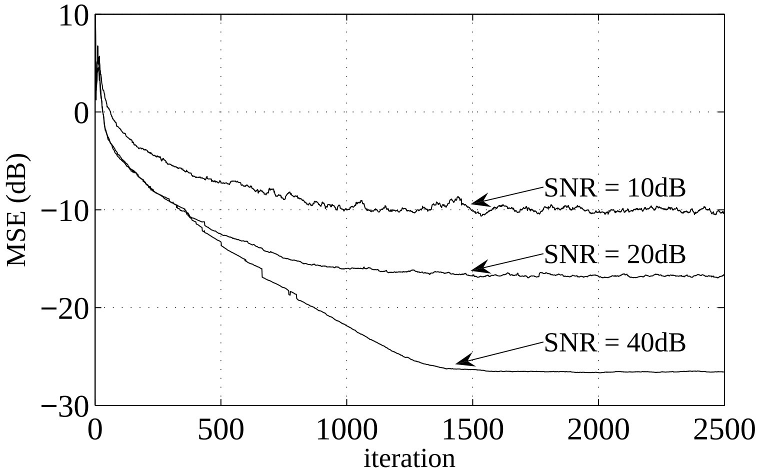


Figure 5.7: MSE $e_r^2[n]$ on the Wiener system's internal signal $r[n]$ for various noise levels, obtained by the adaptive KCCA algorithm.

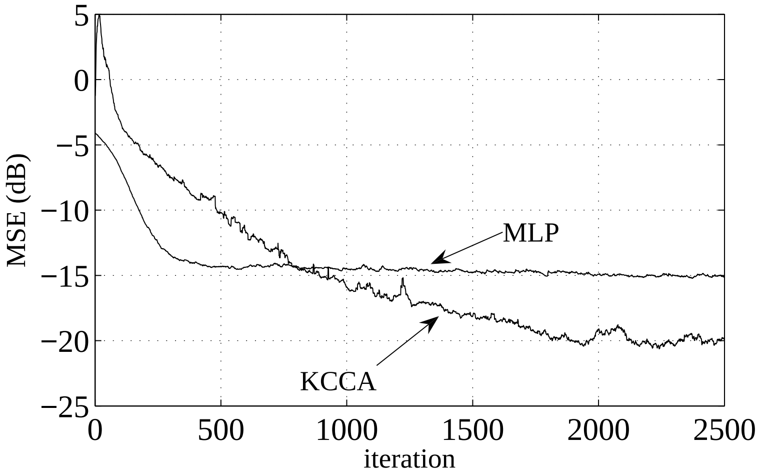


Figure 5.8: MSE $e_y^2[n]$ for full system identification of the Wiener system, using a black-box method (MLP) and the proposed KCCA algorithm.

The identification results can be seen in Fig. 5.9. As in the case of the static Wiener system, the ALD-based kernel CCA algorithm obtains the worst performance, for reasons discussed earlier. The algorithms based on SW-KRLS and kernel PCA based KRLS obtain very similar performance.

Static Hammerstein system identification

In this setup, we considered a static Hammerstein system consisting of the nonlinearity $f(x) = \tanh(x)$ followed by the linear channel

$$H(z) = 1 - 0.4326z^{-1} + 0.3656z^{-2} - 0.3153z^{-3}. \tag{5.34}$$

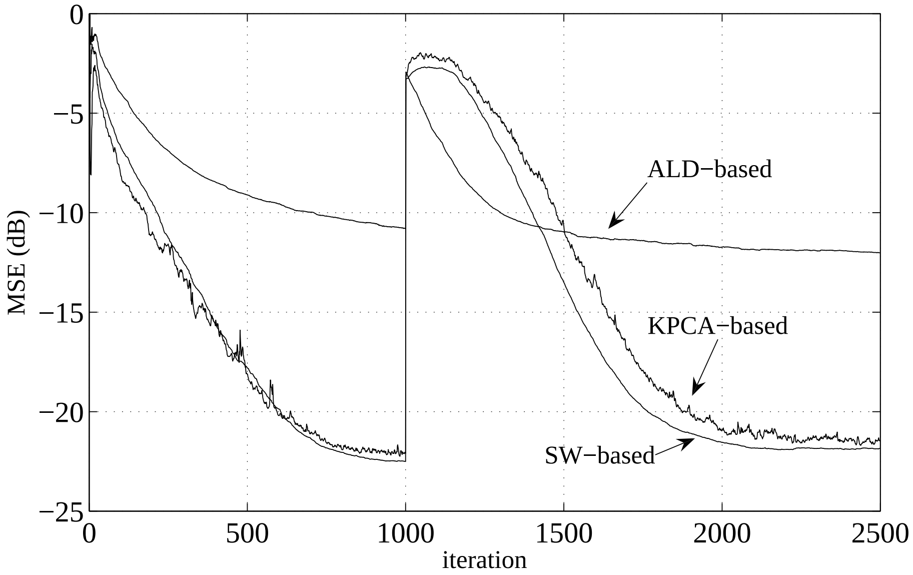


Figure 5.9: Wiener system MSE $e_r^2[n]$ obtained by adaptive identification of a Wiener system that exhibits an abrupt change and contains additive noise.

To the output of this system, 20dB zero-mean additive white Gaussian noise was added. When applying the proposed kernel CCA-based algorithms to identify this system, the *direct* nonlinearity is estimated and a FIR estimate is made of the *inverse* linear channel, which corresponds to an IIR filter. To adequately estimate this channel, the length of the direct FIR filter estimate was considerably increased.

The adaptive kernel CCA algorithms were applied with the following parameters:

- kernel CCA with standard regularization, $c = 10^{-2}$, and a sliding window of 150 samples, using the Gaussian kernel function with kernel width $\sigma = 0.2$;
- kernel CCA with ALD-based sparsification, using accuracy parameter $\nu = 10^{-2}$ and the Gaussian kernel function with kernel width $\sigma = 0.2$;
- kernel CCA based on kernel PCA using 10 eigenvectors, a 150-sample sliding window the Gaussian kernel function with kernel width $\sigma = 0.2$.

In all three algorithms, the inverse linear channel was approximated as a FIR channel of length 15.

The MSE results for the Hammerstein system identification can be found in Fig. 5.10. The observed MSE performances are similar to the observations already made for the previous examples. However, due to the different setup and the presence of noise, the obtained results are not as good as those of the identification of a static noiseless Wiener system (see Fig. 5.6). Nevertheless, with the chosen parameters, the L_2 regularization-based kernel CCA algorithm is capable of attaining the 20dB noise floor.

In all previous examples, the length N of the sliding windows for the L_2 regularization-based kernel CCA and the kernel PCA-based kernel CCA was fixed

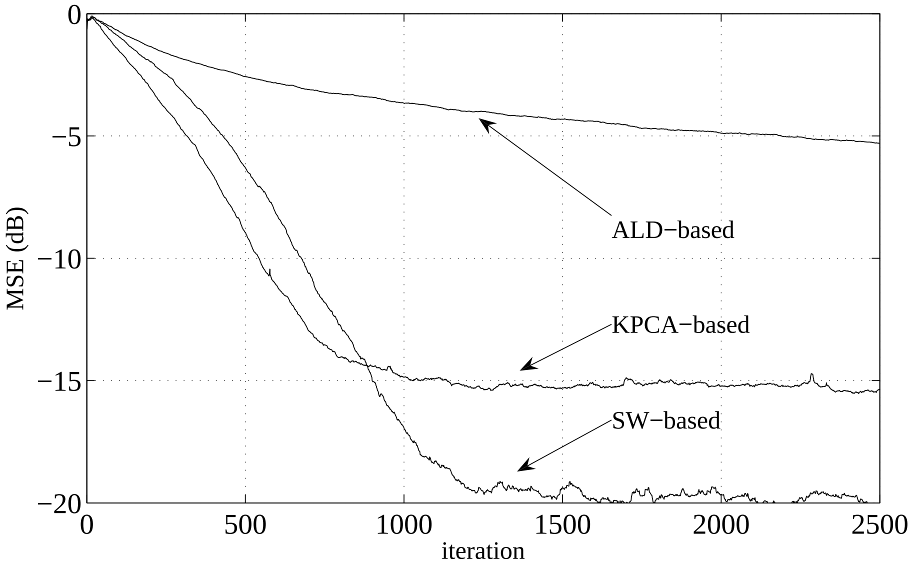


Figure 5.10: MSE $e_r^2[n]$ on the Hammerstein system's internal signal $r[n]$ for the three adaptive kernel CCA-based algorithms.

as 150. Taking into account the number of eigenvectors used by the latter, both obtain a very similar computational complexity. The ALD-based algorithm, on the other hand, is computationally more attractive, but it is not capable of obtaining the same performance levels in non-stationary environments.

5.5 Conclusions and Discussion

In this chapter we focussed on the problem of supervised Wiener and Hammerstein system identification, taking into account explicitly the system's structure. To this end, we proposed to simultaneously estimate the linear and nonlinear parts in a fashion inspired by the identification technique from [Aschbacher and Rupp, 2005]. However, we showed that by applying a more robust restriction to the solution, the proposed kernel CCA algorithm emerges as the logical solution to identify these nonlinear systems. We then discussed a kernel CCA framework that encompasses different types of regularization to avoid overfitting.

In the second contribution of this chapter, we showed how the identification problem can be solved in an iterative manner, by formulating the kernel CCA problem as a set of two coupled least-squares regression problems. This allowed us to develop adaptive versions of the proposed KCCA algorithms, for the three different types of regularization, which are capable of identifying systems that change over time.

The proposed algorithms were compared in a series of batch and online experiments. Experimentally it was verified that an ALD-based algorithm such as the dictionary-based kernel RLS from [Engel et al., 2004] is not readily capable of performing tracking, and therefore cannot be used in this kernel CCA framework. The kernel CCA algorithm using L_2 regularization and a sliding window obtains similar performance and computational cost as the kernel PCA-based algorithm. These two algorithms showed to be successful in identifying both static and time-varying Wiener and Hammerstein systems.

The publications that have contributed to this chapter are

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Online kernel canonical correlation analysis for supervised equalization of Wiener systems". In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Vancouver, Canada, 2006.

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Adaptive kernel canonical correlation analysis algorithms for nonparametric identification of Wiener and Hammerstein systems". *EURASIP Journal on Advances in Signal Processing*, volume 1, Article ID 875351, 13 pages, 2008.

Chapter 6
Blind Identification and Equalization of Wiener Systems

In this chapter we consider the problem of blind identification and equalization of nonlinear Wiener systems. We follow a well-known linear blind identification approach that can be applied to linear single-input multiple-output (SIMO) systems which, under certain conditions, can be obtained by oversampling a single-input single-output (SISO) system. The SIMO systems considered in this chapter are modeled as SIMO Wiener systems, which contain multiple Wiener systems that are excited by a common input signal. By applying a kernel transform to the output of these systems, we are able to identify the linear channels and inverse nonlinearities with an iterative CCA-based approach.

6.1 Introduction

In the last decade there has been a great interest in blind identification and equalization methods. In digital communications, blind methods permit channel identification or equalization without the need to send known training signals, thus saving bandwidth. In particular, the problem of blind identification of single-input multiple-output (SIMO) linear channels has received considerable attention [Xu et al., 1995, Vía et al., 2006]. In this case, blind identification can be accomplished by resorting only to the second-order statistics (SOS) of the channel output.

While a lot of attention has gone to the analysis of linear SIMO systems, many real-life systems exhibit nonlinear characteristics. Recently, a growing amount of research has been conducted on nonlinear system identification. For an overview we refer to chapter 4. Due to the nonlinear dynamics of such systems, their blind identification is a particularly challenging problem.

Blind methods generally assume some knowledge on the input signal statistics and/or the channel model. For blind identification of Wiener system, few methods have been proposed. In [Gómez and Baeyens, 2007, Vanbeylen et al., 2008] techniques were proposed that require the input signal to be i.i.d. and Gaussian. A less restrictive approach was followed by Taleb et al. in [Taleb et al., 2001], where the input signal to the Wiener system is only required to be i.i.d.

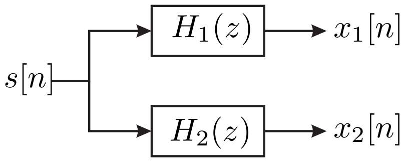


Figure 6.1: A linear SIMO system.

In this chapter we present a blind method to identify and equalize nonlinear SIMO Wiener systems that consist of various Wiener systems. These systems could represent a sensor array in which every sensor exhibits some nonlinear behavior, or they could be obtained by oversampling the output of a nonlinear communication channel that is modeled as a conventional SISO Wiener system [Xu et al., 1995]. In general, the presence of multiple output channels increases the performance of equalization methods, and therefore this scenario has been studied frequently in linear environments [Xu et al., 1995, Ding and Li, 2001]. For the more challenging scenario of blind equalization of *nonlinear* SIMO systems, a few methods have been proposed, particularly for multichannel Volterra models [Giannakis and Serpedin, 1997, López-Valcarce and Dasgupta, 2001].

Before addressing the blind identification problem for SIMO Wiener systems, we will briefly review the blind identification method for linear SIMO systems proposed in [Xu et al., 1995].

6.2 Blind Identification of a Linear SIMO System

For simplicity, we will first treat the case of two outputs. Consider a system that consists of two linear channels $H_1(z)$ and $H_2(z)$ that share the same input signal, $s[n]$, as depicted in Fig. 6.1. The output of each channel can be written as

$$x_i[n] = \sum_{j=0}^{L-1} h_i[j]s[n-j] = h_i[n] * s[n], \tag{6.1}$$

where L is the maximal channel length (which we assume to be known), the coefficients $h_i[j]$ represent the impulse response of each channel $H_i(z)$, i.e. $\mathbf{h}_i = [h_i[0], \dots, h_i[L-1]^T$, and $h_i[n] * s[n]$ denotes the convolution between the filter $\mathbf{h}_i$ and the input signal $s[n]$. Such a system can be obtained, for instance, by oversampling a single linear channel given that the source signal has some excess bandwidth, which is the bandwidth occupied by the signal beyond the Nyquist frequency $1/2T$.

The identification method presented by Xu et al. exploits the commutativity of the convolution, in particular

$$h_2[n] * (h_1[n] * s[n]) = h_1[n] * (h_2[n] * s[n]). \tag{6.2}$$

This property inspired the design of the identification diagram shown in Fig. 6.2, which allows to find estimates of the channels, $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$, by minimizing the follow-

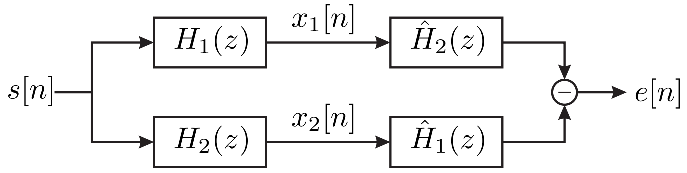


Figure 6.2: A blind identification scheme for the linear SIMO system. If the estimated $\hat{H}_1$ and $\hat{H}_2$ correspond to the real H_1 and H_2, the error $e[n]$ will be zero.

ing cost function

$$\begin{aligned}\min_{\hat{\mathbf{h}}_1,\hat{\mathbf{h}}_2} J &= \frac{1}{2}\sum_{n=1}^{N} |e[n]|^2 \\ &= \frac{1}{2}\sum_{n=1}^{N} |\hat{h}_2[n] * x_1[n] - \hat{h}_1[n] * x_2[n]|^2 \\ &= \frac{1}{2}\sum_{n=1}^{N} |\hat{h}_2[n] * (h_1[n] * s[n]) - \hat{h}_1[n] * (h_2[n] * s[n])|^2.\end{aligned} \tag{6.3}$$

In order to solve this minimization problem, let us adopt a matrix-based notation. Define the matrix

$$\mathbf{X}_i = \begin{bmatrix} x_i[n+L-1] & \cdots & x_i[n] \\ \vdots & \ddots & \vdots \\ x_i[n+N-1] & \cdots & x_i[n+N-L] \end{bmatrix}, \tag{6.4}$$

for $i = 1, 2$. By denoting the estimate of the channel impulse response vectors as

$$\hat{\mathbf{h}}_i = \left[\hat{h}_i[0], \ldots, \hat{h}_i[L-1]\right]^T, \tag{6.5}$$

it can be easily verified that in a noiseless case the solution should satisfy

$$\mathbf{X}_1\hat{\mathbf{h}}_2 = \mathbf{X}_2\hat{\mathbf{h}}_1, \tag{6.6}$$

as illustrated in the identification diagram of Fig. 6.2. Correct identification is guaranteed when the channels $H_i(z)$ do not share any common zeros, for $i = 1, 2$. Also, the *linear complexity* of the input sequence, which is a measure of its diversity, should be sufficiently high. For communication signals this is generally satisfied.

In order to avoid the zero-solution $\hat{\mathbf{h}}_i = \mathbf{0}$, a restriction must be applied. Typical restrictions in communications are either to fix the norm of the filters $\hat{\mathbf{h}}_i$, or to fix the norm of the output signal $\mathbf{X}_i\hat{\mathbf{h}}_j$.

A restriction on the filter norm was used in [Xu et al., 1995] to develop a LS method. With this restriction, the minimization problem (6.3) becomes

$$\min_{\hat{\mathbf{h}}_1,\hat{\mathbf{h}}_2} J_{LS} = \frac{1}{2}\left\|\mathbf{X}_1\hat{\mathbf{h}}_2 - \mathbf{X}_2\hat{\mathbf{h}}_1\right\|^2 \quad \text{s.t. } \|\hat{\mathbf{h}}_1\|^2 + \|\hat{\mathbf{h}}_2\|^2 = 1, \tag{6.7}$$

whose solution can be obtained from the following eigenvalue problem

$$\begin{bmatrix} \mathbf{X}_1^T\mathbf{X}_1 & -\mathbf{X}_1^T\mathbf{X}_2 \\ -\mathbf{X}_2^T\mathbf{X}_1 & \mathbf{X}_2^T\mathbf{X}_2 \end{bmatrix} \hat{\mathbf{h}} = \beta\hat{\mathbf{h}}, \tag{6.8}$$

where $\hat{\mathbf{h}} = [\hat{\mathbf{h}}_2^T, \hat{\mathbf{h}}_1^T]^T$ is found as the eigenvector corresponding to the smallest eigenvalue.

If, instead, the constraint is applied to the norm of output signals (as in [Vía et al., 2006]), the cost function to minimize becomes

$$\min_{\hat{\mathbf{h}}_1,\hat{\mathbf{h}}_2} J_{CCA} = \frac{1}{2}\left\|\mathbf{X}_1\hat{\mathbf{h}}_2 - \mathbf{X}_2\hat{\mathbf{h}}_1\right\|^2 \quad \text{s.t. } \|\mathbf{X}_1\hat{\mathbf{h}}_2\|^2 = \|\mathbf{X}_2\hat{\mathbf{h}}_1\|^2 = 1. \tag{6.9}$$

This is a canonical correlation analysis (CCA) problem, and its solution is given by the principal eigenvector of the following generalized eigenvalue problem (GEV) (see appendix D).

$$\begin{bmatrix} \mathbf{X}_1^T\mathbf{X}_1 & \mathbf{X}_1^T\mathbf{X}_2 \\ \mathbf{X}_2^T\mathbf{X}_1 & \mathbf{X}_2^T\mathbf{X}_2 \end{bmatrix} \hat{\mathbf{h}} = \beta \begin{bmatrix} \mathbf{X}_1^T\mathbf{X}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_2^T\mathbf{X}_2 \end{bmatrix} \hat{\mathbf{h}}. \tag{6.10}$$

Once the channels $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ have been estimated by solving either of the eigenvector problems, system equalization can be performed by applying the zero-forcing (ZF) or the minimum mean square error (MMSE) approach. Note that both the LS algorithm and the CCA-based algorithm require knowledge of the maximum channel length[1] L, and they assume that the linear channels share no common zeros.

Although we only discussed these methods for a 1×2 linear SIMO system, their generalization to multiple output channels is straightforward, as shown in [Xu et al., 1995] and [Vía et al., 2006]. Systems with more than two outputs can be encountered for instance in sensor networks, or by oversampling a SISO system with sufficient excess bandwidth. In applications in communications, the raised-cosine filter is widely used for pulse-shaping. When a raised cosine pulse with a roll-off factor $\beta = 1$ is used, the excess bandwidth amounts to 100% [Proakis, 1983], and by oversampling with a factor 2 the signal diversity is already exploited completely.

6.3 Blind Identification and Equalization of SIMO Wiener Systems

Let us now consider a nonlinear SIMO system, in which each channel is modeled as a Wiener system. In Fig. 6.3 an example with only two outputs is shown, as encountered for instance in [Chen, 1995]. We will call this system a *SIMO Wiener system*, since each of its channels consists of a Wiener system. It can be found in a blind source separation context, where it represents a specific *post-nonlinear* mixture model of one source signal that is measured by multiple sensors that show a certain nonlinear behavior.

[1] Note that we choose to specify the channel length L, which fixes the channel order as $L - 1$.

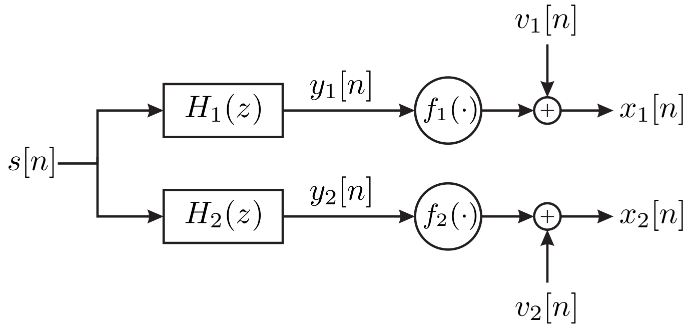


Figure 6.3: A SIMO system consisting of two Wiener systems. Additive noise is added to the output of each channel, in the form of $v_1[n]$ and $v_2[n]$.

In general, this system can also be obtained by oversampling the output of a single Wiener system, similar to the linear case of section 6.2. The validity of this model can be shown as follows. Imagine that one had access to the Wiener system's internal signal and could sample it. Since this signal is the output of a linear channel, by oversampling it one could represent the oversampled linear channel output by the equivalent two-channel diagram from section 6.2. On the other hand, notice that the Wiener system's nonlinearity is memoryless and therefore applies to the signal on a sample-by-sample basis. Therefore, it does not matter if one oversamples the internal signal or the output signal, and hence the equivalent diagram corresponds exactly to the one given in Fig. 6.3. Furthermore, in this case both nonlinearities $f_1(\cdot)$ and $f_2(\cdot)$ are identical.

6.3.1 Problem setting and identification scheme

A SIMO Wiener system with 2 outputs can be modeled as

$$
\begin{aligned}
y_i[n] &= \sum_{j=0}^{L-1} h_i[j]s[n-j] && (6.11)\\
x_i[n] &= f_i(y_i[n]) + v_i[n], && (6.12)
\end{aligned}
$$

for $i = 1, 2$, where $s[n]$ represents the input symbol sent at time instant n, $h_i[j]$ is the j-th coefficient of the i-th linear FIR channel $H_i(z)$, $f_i(\cdot)$ is the nonlinearity of channel i and $v_i[n]$ represents additive Gaussian noise, for $i = 1, 2$ and $n = 1, \ldots, N$. Without loss of generality, L represents the maximum channel length.

The blind equalization problem consists in recovering the transmitted signal $s[n]$ when only the output signals $x_i[n]$ are observed. The proposed solution is mainly based on the linear identification method from section 6.2. First of all, note that, if we are capable of canceling out the nonlinearities, for instance by applying the inverse nonlinearities to the outputs $x_i[n]$, this problem reduces to the linear case which can be solved by one of the discussed techniques.

The proposed identification diagram for a 1×2 SIMO Wiener system is represented in Fig. 6.4. In this diagram, the nonlinearities $\hat{g}_i(\cdot)$ represent the inverse of

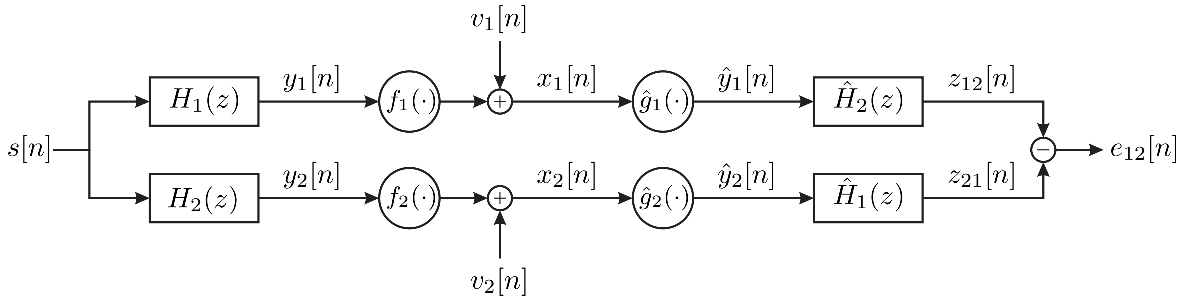


Figure 6.4: The proposed identification diagram for a SIMO system consisting of two Wiener subsystems.

the nonlinearities $f_i(\cdot)$. We will estimate these inverse nonlinearities as kernel expansions $\hat{g}_i(\cdot) = \sum_{m=1}^{M} \hat{\alpha}_i[m]\kappa(x_i^s[m], \cdot)$, where $x_i^s[m]$ are some basis points used for the nonlinear representation. The corresponding output of the nonlinearity becomes

$$\hat{y}_i[n] = \hat{g}_i(x_i[n]) = \sum_{m=1}^{M} \hat{\alpha}_i[m]\kappa(x_i^s[m], x_i[n]). \tag{6.13}$$

In the following we will use the variable $k_i^s(m,n) = \kappa(x_i^s[m], x_i[n])$ to simplify the notation. In a first approach, all available points $x_i[n]$ will be used as basis vectors, i.e., $M = N$.

Given only the outputs $x_i[n]$ of the system, direct estimation of the nonlinearities is not possible in general, as no information on the input signal $s[n]$ is available. Therefore, we will design an algorithm that allows us to obtain both the linear filters and the nonlinearities simultaneously, through a single cost function.

6.3.2 Proposed cost function

First, we will treat the case where the observed system has only two outputs. Given the representation of the estimated nonlinearity $\hat{g}_i(\cdot)$ as in (6.13), the first output of the proposed identification scheme (see Fig. 6.4) can be written as

$$z_{12}[n] = \sum_{j=0}^{L-1} \sum_{m=1}^{M} \hat{h}_2[j] k_1^s(m, n-j) \hat{\alpha}_1[m]. \tag{6.14}$$

In matrix notation, this becomes

$$z_{12}[n] = \hat{\mathbf{h}}_2^T \mathbf{K}_1[n] \hat{\boldsymbol{\alpha}}_1, \tag{6.15}$$

where the l-th row of $\mathbf{K}_1[n]$ contains the elements from $k_1^s(n+l-1,1)$ till $k_1^s(n+l-1, M)$. The expression for $z_{21}[n]$ is found in the same manner, in particular

$$z_{21}[n] = \hat{\mathbf{h}}_1^T \mathbf{K}_2[n] \hat{\boldsymbol{\alpha}}_2. \tag{6.16}$$

In order to estimate the coefficients $\hat{h}_i[j]$ and $\hat{\alpha}_i[m]$, $i = 1, 2$, we propose to minimize the MSE between the system's output signals $z_{12}[n]$ and $z_{21}[n]$

$$\min_{\hat{\alpha},\hat{\mathbf{h}}} J_2 = \|\mathbf{z}_{12} - \mathbf{z}_{21}\|^2 \quad \text{s.t.} \ \|\mathbf{z}_{12}\|^2 = \|\mathbf{z}_{21}\|^2 = 1, \tag{6.17}$$

where the vectors $\mathbf{z}_{12}$ and $\mathbf{z}_{21}$ contain N output samples of each channel, and by $\hat{\alpha}$ and $\hat{\mathbf{h}}$ we denote parameter sets that respectively contain all coefficients of the nonlinear expansion, $\hat{\alpha}_i[m]$, and all coefficients of the linear filters, $\hat{h}_i[j]$, for $i = 1, 2$. The expanded form of this cost function is obtained by introducing the definitions of $z_{12}[n]$ and $z_{21}[n]$, namely

$$\begin{aligned} \min_{\hat{\mathbf{h}},\hat{\alpha}} J_2 &= \sum_{n=1}^{N} |z_{12}[n] - z_{21}[n]|^2 \\ &= \sum_{n=1}^{N} \left| \sum_{j=0}^{L-1} \sum_{m=1}^{M} \hat{h}_2[j] k_1^s(m, n-j) \hat{\alpha}_1[m] - \sum_{j=0}^{L-1} \sum_{m=1}^{M} \hat{h}_1[j] k_2^s(m, n-j) \hat{\alpha}_2[m] \right|^2 \\ \text{s.t.} \ & \sum_{n=1}^{N} |z_{12}[n]|^2 = \sum_{n=1}^{N} |z_{21}[n]|^2 = 1. \end{aligned} \tag{6.18}$$

6.3.3 Iterative solution

The cost function (6.17) needs to be minimized w.r.t. two parameter sets, $\boldsymbol{\alpha}$ and $\mathbf{h}$, but it has no closed-form analytical solution. However, if $\hat{\boldsymbol{\alpha}}_1$ and $\hat{\boldsymbol{\alpha}}_2$ were available, it would be possible to obtain the corresponding optimal filters $\hat{\mathbf{h}}_2$ and $\hat{\mathbf{h}}_1$ by applying linear CCA. Moreover, since the nonlinearities $g_1(\cdot)$ and $g_2(\cdot)$ are represented as linear combinations of kernels, a similar operation can be carried out to estimate these: if $\hat{\mathbf{h}}_2$ and $\hat{\mathbf{h}}_1$ are given, (6.17) can be solved to find the optimal coefficients of the kernel expansions $\hat{\boldsymbol{\alpha}}_\mathbf{1}$ and $\hat{\boldsymbol{\alpha}}_\mathbf{2}$. This suggests an iterative scheme that alternates between updating the linear channels $\hat{\mathbf{h}}_i$ and the memoryless nonlinearities $\hat{\boldsymbol{\alpha}}_\mathbf{i}$. Convergence is guaranteed because each update may either decrease or maintain the cost [Bezdek and Hathaway, 2003, Stoica and Selen, 2004].

This procedure also motivates the use of a signal energy based restriction in (6.17). In both iteration types, the signal energy can be restricted, in contrast to the restriction of (6.7) which can only be fulfilled when optimizing the linear filters.

Iteration 1: given $\hat{\boldsymbol{\alpha}}_i$, obtain $\hat{\mathbf{h}}_i$

If estimates of $\hat{\boldsymbol{\alpha}}_1$ and $\hat{\boldsymbol{\alpha}}_2$ are given, Eq. (6.14) shows that the output $z_{12}[n]$ of the identification scheme can be obtained as

$$z_{12}[n] = \sum_{i=0}^{L-1} \hat{h}_2[i] \hat{y}_1[n-i], \tag{6.19}$$

where $\hat{y}_1[n-i]$ is calculated with (6.13). In matrix form this can be written as $\mathbf{z}_{12} = \hat{\mathbf{Y}}_1 \hat{\mathbf{h}}_2$, where the n-th row of the matrix $\hat{\mathbf{Y}}_1$ contains the elements from $\hat{y}_1[n]$

until $\hat{y}_1[n+L-1]$. The minimization problem (6.17) can be rewritten as minimizing

$$\min_{\hat{\mathbf{h}}} J_2 = \|\hat{\mathbf{Y}}_1\hat{\mathbf{h}}_2 - \hat{\mathbf{Y}}_2\hat{\mathbf{h}}_1\|^2 \quad \text{s.t. } \|\hat{\mathbf{Y}}_1\hat{\mathbf{h}}_2\|^2 = \|\hat{\mathbf{Y}}_2\hat{\mathbf{h}}_1\|^2 = 1, \tag{6.20}$$

which can be solved by standard linear CCA.

Iteration 2: given $\hat{\mathbf{h}}_i$, obtain $\hat{\boldsymbol{\alpha}}_i$

If estimates of $\hat{\mathbf{h}}_1$ and $\hat{\mathbf{h}}_2$ are given, Eq. (6.14) shows that the output $z_{12}[n]$ of the identification scheme can be obtained as

$$z_{12}[n] = \sum_{m=1}^{M} w_{12}[n,m]\hat{\alpha}_1[m], \tag{6.21}$$

where the variable $w_{12}[n,m] = \sum_{i=0}^{L-1} \hat{h}_2[i]k_1(n-i,m)$ is introduced. In matrix form this can be written as $\mathbf{z}_{12} = \mathbf{W}_{12}\hat{\boldsymbol{\alpha}}_1$, where the n-th row of the matrix $\mathbf{W}_{12}$ contains the elements $w_{12}[n,1]$ until $w_{12}[n,M]$. The minimization problem (6.17) can be rewritten as minimizing

$$\min_{\hat{\boldsymbol{\alpha}}} J_2 = \|\mathbf{W}_{12}\hat{\boldsymbol{\alpha}}_1 - \mathbf{W}_{21}\hat{\boldsymbol{\alpha}}_2\|^2 \quad \text{s.t. } \|\mathbf{W}_{12}\hat{\boldsymbol{\alpha}}_1\|^2 = \|\mathbf{W}_{21}\hat{\boldsymbol{\alpha}}_2\|^2 = 1, \tag{6.22}$$

which establishes a CCA problem that accounts for the estimation of the nonlinearities. However, if all data points $x_i[n]$ are used as support vectors in the kernel expansion (6.13), i.e., if $M = N$ (which implies $\hat{\boldsymbol{\alpha}}_i \in \mathbb{R}^N$), the dimensionality of this problem is significantly higher than its linear counterpart (6.20). This will lead to overfitting problems and high computational complexity, as discussed in the previous chapters. To avoid these issues, we propose to apply PCA on the kernel matrices $\mathbf{K}_i \in \mathbb{R}^{N\times N}$, and approximate them as

$$\mathbf{V}_i\boldsymbol{\Sigma}_i\mathbf{V}_i^T \approx \mathbf{K}_i, \tag{6.23}$$

where $\boldsymbol{\Sigma}_i \in \mathbb{R}^{M\times M}$ is a diagonal matrix containing the M largest eigenvalues of $\mathbf{K}_i$ and $\mathbf{V}_i \in \mathbb{R}^{N\times M}$ contains the M corresponding eigenvectors. This allows us to redefine the variable $w_{12}[n,m]$ as

$$w_{12}[n,m] = \sum_{i=0}^{L-1} h_2[i]v_1(n-i,m), \tag{6.24}$$

where $v_1(n,m)$ is the n-th element of the m-th eigenvector[2] in $\mathbf{V}_1$.

Since the eigenspectrum of kernel matrices usually decays quickly, the dimensions of the matrices $\mathbf{W}_{ij}$ in (6.22) can be reduced to $N \times M$, with $M \ll N$, which reduces the computational complexity of the CCA problem (6.22) correspondingly from $O(N^3)$ to $O(NM^2)$. In practice, selection of the rank M can be done by inspecting the decay of the eigenvalues, or by simply defining a percentage of the

[2] Since the eigenvalues can be absorbed in the expansion coefficients of $\hat{\alpha}_1$ and $\hat{\alpha}_2$, we do not need to take them into account here.

Algorithm 6.1 Alternating KCCA for Blind Equalization of SIMO Wiener Systems

```
initialize
    Obtain $\hat{\mathbf{h}}_i$ by solving the LS problem (6.8).
    Construct the kernel matrices $\mathbf{K}_i$ from $x_i[n]$.
    Perform kernel PCA to obtain the rank-reduced matrices $\mathbf{W}_{ij}$.
repeat
    CCA1: With given $\hat{\mathbf{h}}_i$, update $\hat{\boldsymbol{\alpha}}_i$ by solving (6.22).
    CCA2: With given $\hat{\boldsymbol{\alpha}}_i$, update $\hat{\mathbf{h}}_i$ by solving (6.20).
until convergence
Obtain $s[n]$ from $\hat{y}_i[n]$ and $\hat{\mathbf{h}}_i$ by applying linear ZF or MMSE equalizers.
```

signal energy that needs to be maintained. Kernel PCA can be performed efficiently even on very large data sets by following the implementation described in appendix B.3. Furthermore, note that this operation also avoids overfitting by restricting the solution space to be low-rank.

6.3.4 Initialization

Analogously to many other iterative techniques, the proposed cyclic minimization algorithm could suffer from local minima. In practice, nevertheless, local minima can be avoided by means of a proper initialization technique. A straightforward initialization consists in fixing $\hat{\boldsymbol{\alpha}}_i$ such that the kernel expansions represent the identity function $g(x) = x$, and obtaining the initial estimate of the linear channels $\hat{\mathbf{h}}_i$ by solving the linear LS problem (6.8) for the system outputs $x_i[n]$.

A more accurate initialization scheme can be obtained by combining all products of $\hat{\mathbf{h}}_p$ and $\hat{\boldsymbol{\alpha}}_p$ that occur in the cost function (6.18) into a single "solution vector", and minimizing the cost function w.r.t. to this vector. Since the solution should contain the Kronecker products between $\hat{\boldsymbol{\alpha}}_1$ and $\hat{\mathbf{h}}_2$ on one hand, and $\hat{\boldsymbol{\alpha}}_2$ and $\hat{\mathbf{h}}_1$ on the other hand, a good estimate can be obtained by imposing this structure on the solution.

However, the proposed blind equalization method requires the nonlinearity to be invertible, and therefore in practice a linear initialization is close enough to the global minimum. The entire iterative technique for 2 output channels is summarized in Alg. 6.1.

6.3.5 Solution for systems with multiple outputs

In the general case of a system with P sensors, the cost function needs to take into account the correlations between each pair of outputs.

The cost function (6.17) for a system with 2 output channels is generalized for P outputs as

$$\min_{\hat{\boldsymbol{\alpha}},\hat{\mathbf{h}}} J_P = \sum_{\substack{i,j=1 \\ i\neq j}}^{P} \|\mathbf{z}_{ij} - \mathbf{z}_{ji}\|^2 \quad \text{s.t. } \|\mathbf{z}_{ij}\|^2 = 1, \tag{6.25}$$

where $\mathbf{z}_{ij}$ denotes the signal obtained by transforming the output signal $\mathbf{x}_i$ by $g_i(\cdot)$ and filtering it by $\mathbf{h}_j$. The resulting algorithm is analogous to the two-channel case of Alg. 6.1. The extensions of (6.22) and (6.20) to P outputs are obtained as follows. Given estimates of $\hat{\mathbf{h}}_i$, a set of estimates of $\hat{\boldsymbol{\alpha}}_i$ is found by minimizing

$$\min_{\hat{\boldsymbol{\alpha}}} J_P = \sum_{\substack{i,j=1\\ i\neq j}}^{P} \|\mathbf{W}_{ij}\hat{\boldsymbol{\alpha}}_i - \mathbf{W}_{ji}\hat{\boldsymbol{\alpha}}_j\|^2 \quad \text{s.t. } \|\mathbf{W}_{ij}\hat{\boldsymbol{\alpha}}_i\|^2 = 1, \tag{6.26}$$

where the n-th row of $\mathbf{W}_{ij}$ contains the elements $w_{ij}[n,1]$ until $w_{ij}[n,M]$. Here, $w_{ij}[n,m]$ is defined as $w_{ij}[n,m] = \sum_{l=0}^{L-1} h_j[l] v_i(n-l,m)$, and $v_i(n,m)$ is the n-th element of the m-th eigenvector of $\mathbf{K}_i$.

The solution to the minimization problem (6.26) can be found as the principal eigenvector of the corresponding GEV. After some calculations, this GEV reads

$$\mathbf{R}_{\hat{\mathbf{h}}}\hat{\boldsymbol{\alpha}} = \beta \mathbf{D}_{\hat{\mathbf{h}}}\hat{\boldsymbol{\alpha}}, \tag{6.27}$$

in which

$$\mathbf{R}_{\hat{\mathbf{h}}} = \begin{bmatrix} \mathbf{0} & \mathbf{W}_{12}^T\mathbf{W}_{21} & \cdots & \mathbf{W}_{1P}^T\mathbf{W}_{P1} \\ \mathbf{W}_{21}^T\mathbf{W}_{12} & \mathbf{0} & \cdots & \mathbf{W}_{2P}^T\mathbf{W}_{P2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_{P1}^T\mathbf{W}_{1P} & \mathbf{W}_{P2}^T\mathbf{W}_{2P} & \cdots & \mathbf{0} \end{bmatrix}, \tag{6.28}$$

$\mathbf{D}_{\hat{\mathbf{h}}}$ is a block-diagonal matrix whose i-th block on the diagonal is $\sum_{j=1;j\neq i}^{P} \mathbf{W}_{ij}^T\mathbf{W}_{ij}$, for $i = 1,\ldots,P$, and $\hat{\boldsymbol{\alpha}} = [\hat{\boldsymbol{\alpha}}_1, \hat{\boldsymbol{\alpha}}_2, \ldots, \hat{\boldsymbol{\alpha}}_P]^T$. Details of how this GEV is obtained can be found in appendix D.

Subsequently, the parameters $\hat{\boldsymbol{\alpha}}_i$ are fixed and new estimates of $\hat{\mathbf{h}}_i$ are obtained by minimizing

$$\min_{\hat{\mathbf{h}}} J_P = \sum_{\substack{i,j=1\\ i\neq j}}^{P} \|\hat{\mathbf{Y}}_i\hat{\mathbf{h}}_j - \hat{\mathbf{Y}}_j\hat{\mathbf{h}}_i\|^2 \quad \text{s.t. } \|\hat{\mathbf{Y}}_i\hat{\mathbf{h}}_j\|^2 = 1, \tag{6.29}$$

where the n-th row of the matrix $\hat{\mathbf{Y}}_i$ contains the elements $\hat{y}_i[n]$ until $\hat{y}_i[n+L-1]$, and $\hat{y}_1[n]$ is calculated with (6.13). Again, the minimization problem (6.29) can be solved by retrieving the principal eigenvector of the corresponding GEV. This GEV, which is identical to Eq. (D.10), is found as

$$\frac{1}{P}\mathbf{R}_{\hat{\alpha}}\hat{\mathbf{h}} = \beta \mathbf{D}_{\hat{\alpha}}\hat{\mathbf{h}}, \tag{6.30}$$

in which

$$\mathbf{R}_{\hat{\alpha}} = \begin{bmatrix} \hat{\mathbf{Y}}_1^T\hat{\mathbf{Y}}_1 & \hat{\mathbf{Y}}_1^T\hat{\mathbf{Y}}_2 & \cdots & \hat{\mathbf{Y}}_1^T\hat{\mathbf{Y}}_P \\ \hat{\mathbf{Y}}_2^T\hat{\mathbf{Y}}_1 & \hat{\mathbf{Y}}_2^T\hat{\mathbf{Y}}_2 & \cdots & \hat{\mathbf{Y}}_2^T\hat{\mathbf{Y}}_P \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{Y}}_P^T\hat{\mathbf{Y}}_1 & \hat{\mathbf{Y}}_P^T\hat{\mathbf{Y}}_2 & \cdots & \hat{\mathbf{Y}}_P^T\hat{\mathbf{Y}}_P \end{bmatrix}, \tag{6.31}$$

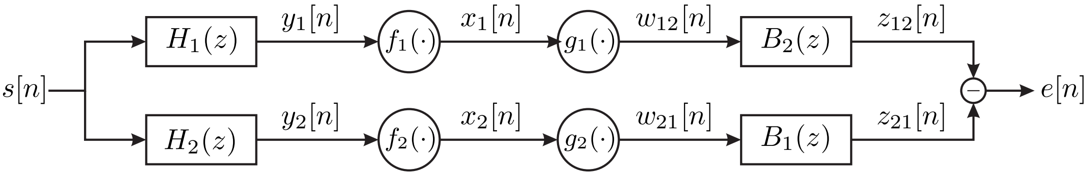


Figure 6.5: The proposed identification diagram for a SIMO Wiener system in a noiseless situation.

$\mathbf{D}_{\hat{\alpha}}$ is a block-diagonal matrix whose i-th block on the diagonal is $\hat{\mathbf{Y}}_i^T\hat{\mathbf{Y}}_i$, and the solution $\hat{\mathbf{h}}$ contains the different estimated filters $\hat{\mathbf{h}} = [\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_P]^T$.

Finally, we must note that when the SIMO system is obtained by oversampling, the P nonlinearities will be the same. Obviously, this can be exploited to obtain a more accurate estimate. The corresponding GEV can be found in a similar manner. Notice also that oversampling multiple times implies that the noise components $v_i[n]$ become correlated, which puts practical restrictions on the use of this algorithm.

6.4 Uniqueness of the Solution

In this section we consider the identifiability conditions of the proposed blind SIMO Wiener system identification diagram. Although we only treat the case of two output channels, the generalization to multiple outputs is straightforward.

Theorem 6.1 (Sufficient condition). *The blind identification problem has a unique solution if*

1. *The polynomials $\{H_i(z)\}_{i=1}^2$ do not share any common zeros.*
2. *The linear complexity of the input signal is at least $2L+1$, where L is the maximum length of the linear channels.*
3. *The memoryless nonlinearities $\{f_i(\cdot)\}_{i=1}^2$ are invertible and infinitely derivable.*
4. *The kernel function $\kappa(\cdot,\cdot)$ is a universal kernel.*

The two first conditions are identical to the sufficient conditions of the linear blind method from [Xu et al., 1995]. The third condition guarantees that the nonlinearities of the system can be canceled out and that they are continuous. The last condition ensures that the representations of $g_1(\cdot)$ and $g_2(\cdot)$ as kernel expansions can represent any infinitely derivable nonlinearities without restrictions.

Consider the diagram of Fig. 6.5, which has no additive noise component. We will prove that the cost function $J_2 = \sum_{n=1}^N e[n]^2$ only reaches its minimal value of zero when the blocks $g_1(\cdot)$, $g_2(\cdot)$, $B_1(z)$ and $B_2(z)$ correspond to

$$\begin{cases} g_1(\cdot) = f_1^{-1}(\cdot); & B_1(z) = H_2(z); \\ g_2(\cdot) = f_2^{-1}(\cdot); & B_2(z) = H_1(z), \end{cases} \tag{6.32}$$

up to some arbitrary scalars, which are inherent to this identification problem. Although it will not be explicitly stated, in the following we will assume that all equalities and inequalities are considered up to such scalars.

First of all, note that the solution (6.32) is always a valid solution. Specifically, its nonlinearities $g_1(\cdot)$ and $g_2(\cdot)$ cancel out $f_1(\cdot)$ and $f_2(\cdot)$, respectively, which yields $w_{12}[n] = y_1[n]$ and $w_{21}[n] = y_2[n]$. In this case the diagram of Fig. 6.5 reduces to the linear diagram of Fig. 6.2, for which it was shown in [Xu et al., 1995] that the unique solution consists of $B_1(z) = H_2(z)$ and $B_2(z) = H_1(z)$.

We now show that the indicated solution is the only possible solution for which $J = 0$. Denote by $\rho_i(x)$ the function composition

$$\rho_i(x) = (g_i \circ f_i)(x), \tag{6.33}$$

for $i = 1, 2$. When $g_i(\cdot) = f_i^{-1}(\cdot)$, as for instance in the solution (6.32), this function equals the identity function $\rho_i(x) = x$. We consider the two cases in which the linear and nonlinear blocks differ from the ideal solution (6.32).

Case 1: $g_1(\cdot) = f_1^{-1}(\cdot)$ **and** $g_2(\cdot) = f_2^{-1}(\cdot)$**, but** $B_1(z) \neq H_1(z)$ **and/or** $B_2(z) \neq H_2(z)$**.** In this case the nonlinearities fulfill (6.32) but one or both linear channels do not. Since $\rho_1(x) = x$ and $\rho_2(x) = x$ here, this identification problem reduces to the linear problem of [Xu et al., 1995], for which the only solution that yields $J = 0$ is $B_1(z) = H_2(z)$ and $B_2(z) = H_1(z)$. Therefore, if the nonlinearities fulfill (6.32), the linear channels must also do so.

Case 2: $g_1(\cdot) \neq f_1^{-1}(\cdot)$ **and/or** $g_2(\cdot) \neq f_2^{-1}(\cdot)$**.** In the second case, assume that the minimal cost $J = 0$ is obtained while at least one of the nonlinearities is different from the solution (6.32). Here, either $\rho_1(x) \neq x$, $\rho_2(x) \neq x$, or they are both different from the identity function. According to the Stone-Weierstraß theorem, the continuous real-valued functions $\rho_i(x)$ on a compact interval can be uniformly approximated by sums of monomials. Without loss of generality, we consider only functions of up to the second order

$$\begin{aligned}\rho_1(x) &= a_1 x + a_2 x^2 \\ \rho_2(x) &= b_1 x + b_2 x^2\end{aligned}$$

After taking Fourier transforms, the signals before B_1 and B_2 become

$$\begin{aligned}W_{12}(\omega) &= S(\omega)H_1(\omega) + a_2 S(\widetilde{\omega)H_1}(\omega) \\ W_{21}(\omega) &= S(\omega)H_2(\omega) + b_2 S(\widetilde{\omega)H_2}(\omega),\end{aligned}$$

where $H_i(\omega) = \mathcal{F}(a_1 B_1[n])$ and $S(\widetilde{\omega)H_i}(\omega) = \mathcal{F}\left((s[n] * H_i[n])^2\right)$ represents the circular convolution. By denoting $G_i(S,\omega) = S(\widetilde{\omega)H_i}(\omega)$, for $i = 1, 2$, the output of the diagram shown in Fig. 6.5 is obtained as

$$\begin{aligned}E(\omega) &= S(\omega)\left[H_1(\omega)B_2(\omega) - H_2(\omega)B_1(\omega)\right] + \left[aG_1(S,\omega)B_2(\omega) - bG_2(S,\omega)B_1(\omega)\right] \\ &= \left[S(\omega)H_1(\omega) + a_2 G_1(S,\omega)\right]B_2(\omega) - \left[S(\omega)H_2(\omega) + b_2 G_2(S,\omega)\right]B_1(\omega).\end{aligned}$$

When $E(\omega) = 0$, the length of all filters between brackets cannot be inferior to L, due to Bézout's identity [Xu et al., 1995]. On the other hand, the second condition in this theorem requires that their lengths do not exceed L. Hence, the lengths of these filters are L, and we obtain

$$\begin{cases} B_1(\omega) = H_1(\omega) + a_2 G_1(S,\omega)/S(\omega) \\ B_2(\omega) = H_2(\omega) + b_2 G_2(S,\omega)/S(\omega). \end{cases} \tag{6.34}$$

Since $B_1(\omega)$ and $B_2(\omega)$ depend on $S(\omega)$, Eq. (6.34) only holds for an arbitrary input $S(\omega)$ when $a_2 = b_2 = 0$, in which case the solution reduces to (6.32). □

6.5 Experiments

We experimentally tested the proposed algorithm with some numerical examples. All tests were conducted on data sets of $N = 256$ data symbols. The fraction of the signal energy discarded by the kernel PCA procedure in the initialization phase was fixed as 10^{-14}. The resulting number of kept eigenvectors was between $M = 11$ and $M = 15$. In all experiments convergence was obtained in less than 20 iterations.

The first system used is a 1×3 Wiener SIMO system with linear filters

$$\begin{aligned} H_1(z) &= 0.6172 + 0.6247z^{-1} + 0.3373z^{-2} - 0.0349z^{-3} - 3.2957z^{-4} \\ H_2(z) &= -0.8601 + 0.1532z^{-1} - 0.1888z^{-2} - 0.6264z^{-3} + 0.9985z^{-4} \\ H_3(z) &= 1.3271 - 0.1472z^{-1} - 0.4786z^{-2} + 0.6682z^{-3} + 0.0045z^{-4}, \end{aligned}$$

respectively. The nonlinearity was the same for all the channels, namely $f_i(x) = \tanh(0.8x) + 0.1x$.

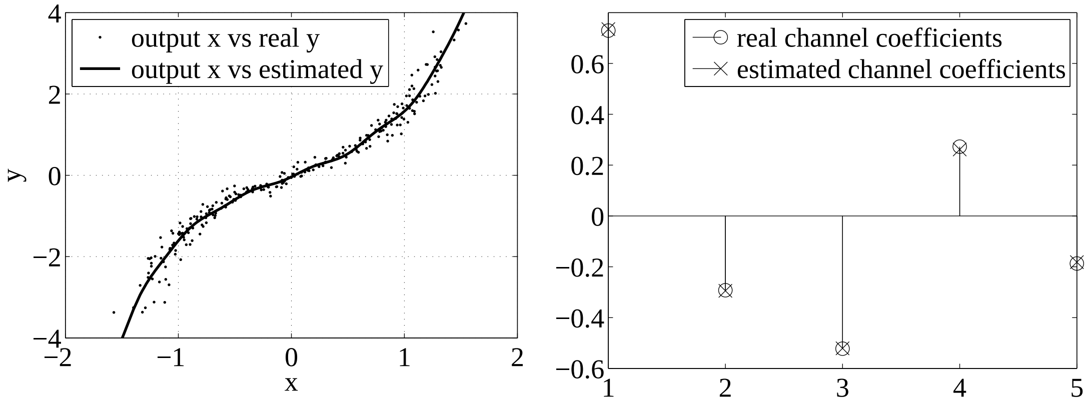
In the first experiment, the source signal was drawn from $\mathcal{N}(0,1)$ and 20dB of Gaussian white noise was added after the nonlinearities. The proposed method was used to identify this system, with a kernel with $\sigma = 1$. Fig. 6.6 shows the true and estimated linear filter and nonlinearity for one of the branches of the Wiener SIMO system, after 15 iterations of the algorithm. Similar results were obtained for the other branches of the system.

In a second experiment, we compared the performance of a number of CCA-based equalizers. The results are shown in Fig. 6.7.

1. As a benchmark, we applied the blind linear CCA-based equalizer with zero-forcing from [Vía et al., 2006] on a system that only contained the *linear* channels $H_1(z)$, $H_2(z)$ and $H_3(z)$. Its performance is shown by the solid curve with black dots.

2. Then, we applied the same blind linear method on the SIMO Wiener system from the first example. Since this system shows nonlinearities, the obtained MSE is obviously very bad (as shown by the solid curve with white squares).

3. The performance of the proposed blind KCCA-based method is shown as the solid curve with white circles.



(a) Nonlinearity estimate

(b) Linear filter coefficients

Figure 6.6: Identification results on the 1 × 3 Wiener SIMO system. (a) shows the noisy output $x_3[n]$ vs. the real internal signal $y_3[n]$, and $x_3[n]$ vs. the estimated $\hat{y}_3[n]$. (b) shows the estimated filter coefficients of $\mathbf{h}_3$ vs. the real coefficients.

4. As a second benchmark, we included the results of the supervised equalization method from chapter 5, using the same kernel parameters. This method was performed in batch mode, on each system branch separately.

Averages were taken over 50 independent Monte-Carlo simulations, and the MSE was calculated between the true and the estimated input signal. As can be observed, the proposed method obtains very satisfactory results on this system. The MSE is even very close to the one obtained by the related supervised method.

For the third test we compared three SIMO Wiener systems with different numbers of outputs. System 1 was a 1 × 2 SIMO Wiener system with $H_1(z)$ and $H_2(z)$ as defined in the first experiment. System 2 was the discussed 1 × 3 system, and system 3 was a 1 × 4 SIMO Wiener system that included all three previous Wiener systems and a new linear channel

$$H_4(z) = -0.1155 - 0.9170z^{-1} + 0.5605z^{-2} + 0.4862z^{-3} - 0.8004z^{-4} \tag{6.35}$$

in its fourth branch. The nonlinearity was maintained, and we exploited the fact that it was the same for each channel. The results are shown in Fig. 6.8. As can be observed, the performance improves when channels are added, which can be explained by the fact that the algorithm is given more information to retrieve the source signal. Nevertheless, this improvement quickly slows down as more channels are added.

Finally, we repeated the third test for a system with a binary input $s[n] \in \{-1, 1\}$, but now we did not exploit the information that the nonlinearity was the same for each channel. Very good results are obtained, as shown in Fig. 6.9, mostly due to the fact that the input signal is binary and we only measure the bit error rate (BER).

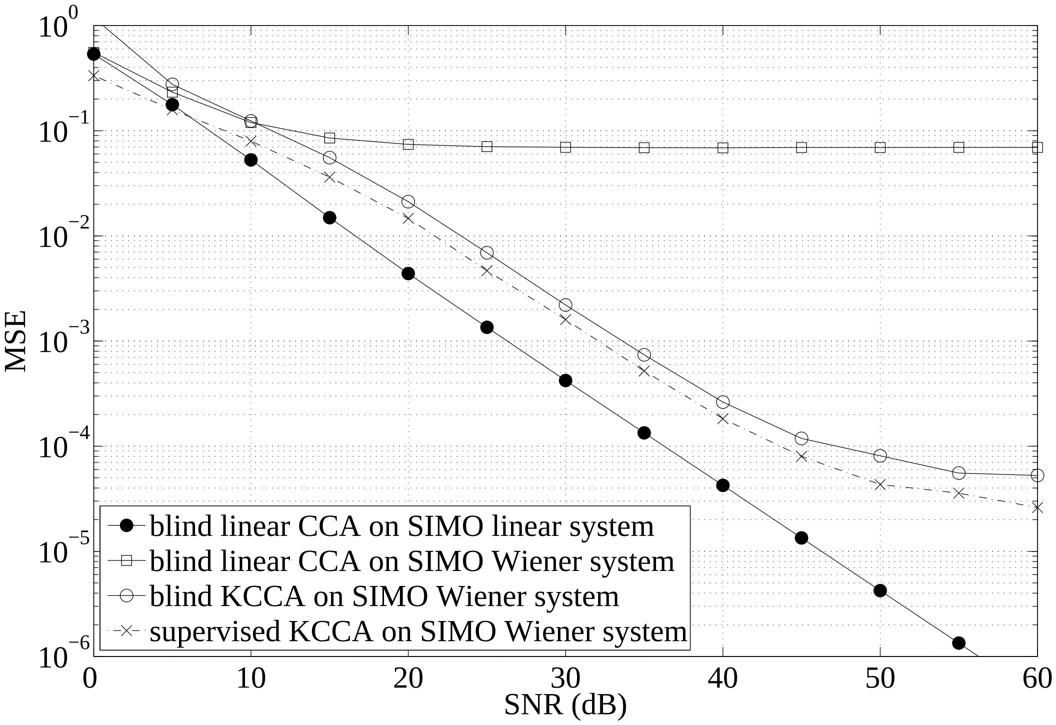


Figure 6.7: MSE comparison for different algorithms.

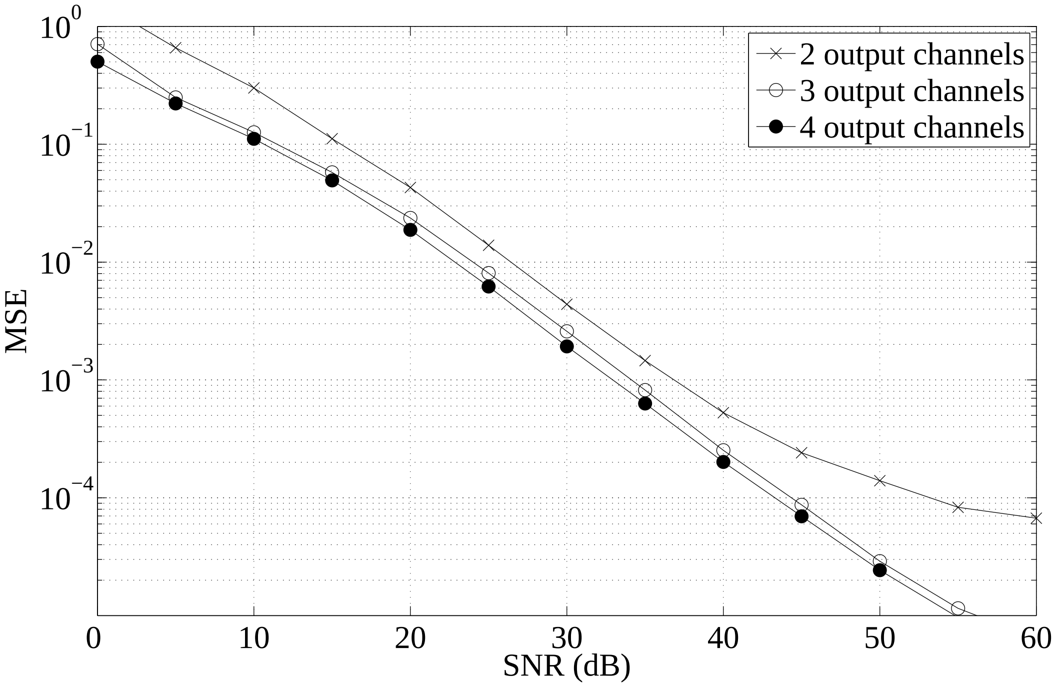


Figure 6.8: Blind identification results for different SIMO Wiener systems with a Gaussian input, where the algorithm took into account that all channels have the same nonlinearity.

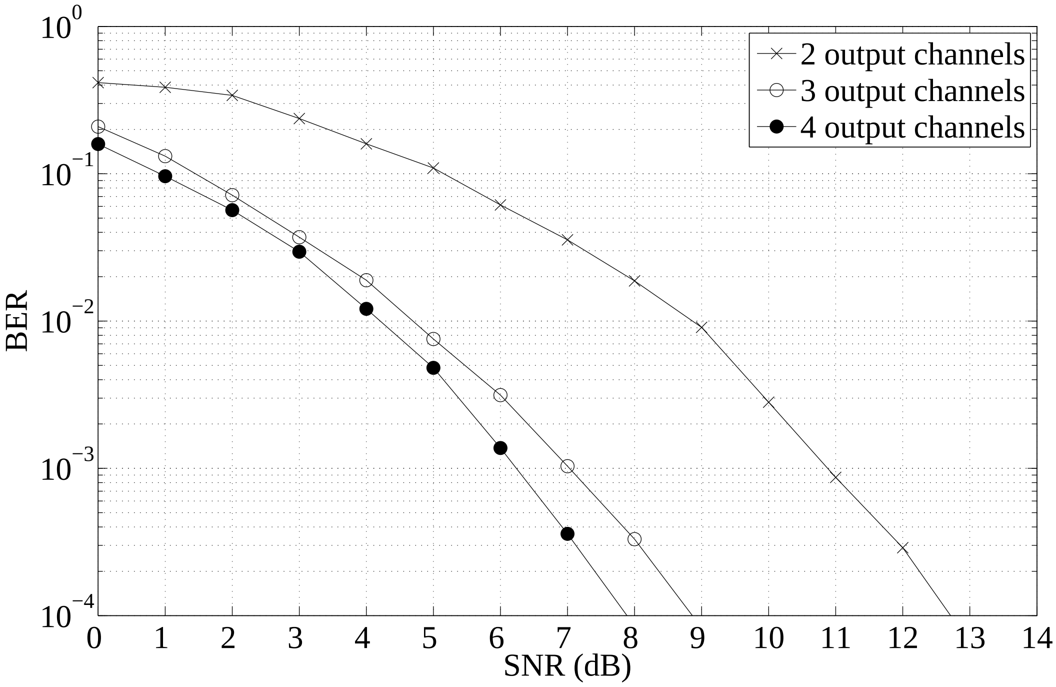


Figure 6.9: Results for SIMO Wiener systems with a binary input.

6.6 Conclusions

We proposed a blind equalization algorithm for nonlinear SIMO systems in which every channel is a Wiener system. The method iterates between a CCA algorithm for estimating the linear channel and a KCCA algorithm for estimating the memoryless nonlinearities. The proposed method is capable of operating correctly when only two output channels are available, which is in contrast to other blind nonlinear equalization methods, for equalization of Volterra systems, such as [Taleb et al., 2001] and [Giannakis and Serpedin, 1997, López-Valcarce and Dasgupta, 2001] that require at least three output channels. Results show that this iterative algorithm converges fast and achieves performance that is very close to a related supervised method.

The publications that have contributed to this chapter are

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. "A kernel canonical correlation analysis algorithm for blind equalization of oversampled Wiener systems". In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2008)*, pages 20–25. 2008.

- S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Blind equalization of post-nonlinear SIMO systems". *Submitted to IEEE Transactions on Signal Processing*, volume 58, pages 1–8, 2010.

Part III

Applications of Spectral Clustering in Source Separation

Chapter 7
Spectral Clustering Techniques

This part of the thesis is dedicated to applications of spectral clustering to blind source separation problems. In a typical source separation scenario, one or more source signals are transformed through a mixture process, and only the final mixtures are observed. In case the source signals are sparse signals or if they are drawn from a finite constellation, the mixture data will often show a number of discrete data groups that can be clustered using a suitable algorithm. In the next two chapters we will discuss two source separation scenarios that lead to difficult clustering problems, which require a sophisticated clustering algorithm such as spectral clustering. In this introductory chapter we will discuss the technique of spectral clustering itself, along with one of its most common implementations.

7.1 Data Clustering

Clustering is an important problem in unsupervised learning. The goal of clustering is to partition a given set of "objects" such that the objects in the same group are as similar as possible to each other, and at the same time as dissimilar as possible to objects that belong to other groups. Traditional clustering algorithms include k-means, hierarchical clustering [Hartigan, 1975], fuzzy c-means [Bezdek, 1981], and expectation maximization (EM) learning [Dempster et al., 1977]. These algorithms are easy to implement but their application is limited to rather simple clustering problems.

Recently, the technique of *spectral clustering* has become a very popular clustering method [Ng et al., 2001, von Luxburg, 2006]. Spectral clustering is easy to implement since it relies only on standard algebraic operations, yet it often outperforms traditional algorithms on more complex clustering tasks. Specifically, it consists in analyzing the spectrum of a modified kernel matrix.

Before introducing spectral clustering, we will first discuss two standard clustering techniques which are used later in this thesis.

Algorithm 7.1 *K*-Means Clustering.

```
Given the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ and the number of clusters to retrieve, $K$.
initialize
    Choose $\boldsymbol{\mu}_j \in \mathcal{X}$ randomly, for $j = 1, 2, \ldots, K$.
repeat
    Assign $\mathbf{x}_i$ to the partition $\mathcal{S}_j$ with the closest center $\boldsymbol{\mu}_j$, for $i = 1, 2, \ldots, N$.
    Recalculate the $\boldsymbol{\mu}_j$ as the center of cluster $\mathcal{S}_j$, for $j = 1, 2, \ldots, K$.
until there is no change in the assignments.
```

7.1.1 Common clustering algorithms

k-means clustering

The k-means clustering procedure is one of the most commonly used clustering techniques. Given a data set $\mathbf{x}_i \in \mathcal{X}$, $i = 1, 2, \ldots, N$ and the number of clusters to retrieve, K, it aims to find the K partitions $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K\}$ so as to minimize the within-cluster sum of squares

$$\arg\min_{\mathcal{S}} \sum_{j=1}^{K} \sum_{\mathbf{x}_i \in \mathcal{S}_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2, \tag{7.1}$$

where $\boldsymbol{\mu}_j = \frac{1}{|\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_j} \mathbf{x}_i$ is the mean vector of partition S_j. Although this problem is NP-hard, a good solution can be obtained in general by using an iterative refinement technique called the "k-means algorithm" (see algorithm 7.1).

Due to its iterative nature, the k-means algorithm is prone to converging to local minima, and it is sensitive to its initialization. Thanks to its low computational complexity, however, in practice it is often sufficient to choose the best clustering result of multiple, randomly initialized runs. In general a more solid initialization is required, and a number of techniques have been proposed for this purpose. For instance, recently it was shown that the continuous (relaxed) solution of k-means clustering is given by the PCA principal components [Ding and He, 2004]. This suggests that a robust "PCA-guided" k-means clustering algorithm can be constructed by initializing the standard k-means algorithm with the PCA relaxed solutions.

Kernel k-means The k-means algorithm can be carried out readily in feature space by describing the problem entirely in terms of inner products [Schölkopf et al., 1996]. Recently, it was shown that the resulting *kernel k-means procedure* is closely linked to spectral clustering [Dhillon et al., 2004]. Many other kernel-based clustering algorithms have been proposed in the past decade, for instance in [Jenssen et al., 2004] which employs an information theoretic learning (ITL) criterion.

Algorithm 7.2 Agglomerative Hierarchical Clustering.

```
Given the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ and the number of clusters to retrieve, $K$.
initialize
    Create $N$ clusters $\mathcal{S}_j$ and assign one data point $\mathbf{x}_i$ to each of them.
repeat
    Calculate $D(\mathcal{S}_j, \mathcal{S}_l)$, for $j, l = 1, \ldots, N$.
    Merge the clusters for which $D(\mathcal{S}_j, \mathcal{S}_l)$ is minimal.
until $K$ clusters are obtained.
```

Hierarchical clustering

Hierarchical clustering is a heuristic procedure of building hierarchies of clusters. This can either be done “bottom up” or “top down”. The bottom-up method consists in first assigning one cluster to each data point and then merging these clusters iteratively. This is known as “agglomerative” clustering. The top-down method, also known as “divisive” clustering, starts with one cluster that contains all points and then recursively splits it while moving down the hierarchy. In algorithm 7.2 we summarize the agglomerative technique, which is far more commonly used.

In order to decide which clusters should be combined in agglomerative clustering, a measure of similarity or dissimilarity between clusters is required. Usually this is a distance measure D. In each iteration, the clusters that are closest according to this measure are merged. The most commonly used linkage criteria are

- Complete linkage clustering: $D_{cl}(\mathcal{S}_i, \mathcal{S}_j) = \max\{d(\mathbf{x}_i, \mathbf{x}_j), \mathbf{x}_i \in \mathcal{S}_i, \mathbf{x}_j \in \mathcal{S}_j\}$. The distance between two clusters is defined as the distance between the most distant pair of points, one from each cluster.

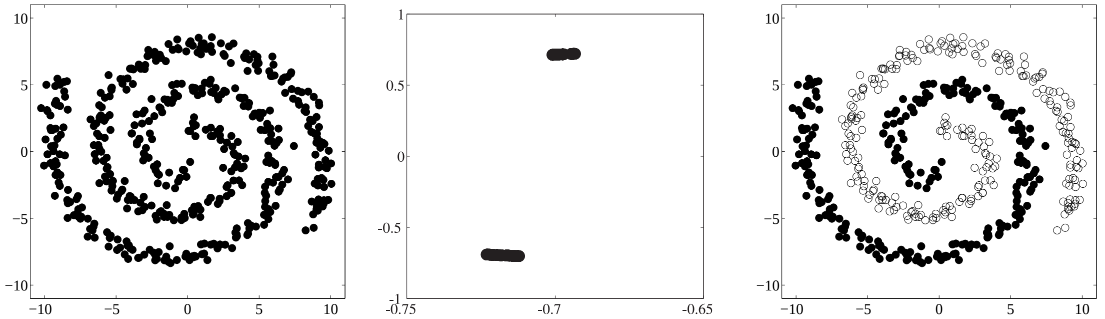
- Single linkage clustering: $D_{sl}(\mathcal{S}_i, \mathcal{S}_j) = \min\{d(\mathbf{x}_i, \mathbf{x}_j), \mathbf{x}_i \in \mathcal{S}_i, \mathbf{x}_j \in \mathcal{S}_j\}$. The distance between two clusters is defined as the distance between the closest pair of points, one from each cluster.

- Average linkage clustering: $D_{al}(\mathcal{S}_i, \mathcal{S}_j) = \frac{1}{|\mathcal{S}_i||\mathcal{S}_j|} \sum_{\mathbf{x}_i \in \mathcal{S}_i} \sum_{\mathbf{x}_j \in \mathcal{S}_j} d(\mathbf{x}_i, \mathbf{x}_j)$. The distance between two clusters is defined as the average of distances between all pairs of points, where each pair is made up of one point from each cluster.

7.2 Spectral Clustering

Spectral clustering [Ng et al., 2001, von Luxburg, 2006] is a recently proposed technique that performs data clustering based on a spectral analysis of point-to-point similarities. Specifically, by calculating the eigenvectors of a modified kernel matrix it reduces the original hard clustering problem to an easier clustering problem which can be solved by conventional clustering algorithms in the eigenspace of this matrix.

The basic form of spectral clustering can be found by solving a graph bipartitioning problem [Chung, 1997], which consists in separating the graph into two



(a) Data points. (b) Eigenvector data. (c) Clustered data points.

Figure 7.1: Example of spectral clustering. (a) A set of data points $\{\mathbf{x}_i\}_{i=1}^N$ representing two intertwined data groups, difficult or impossible to cluster with conventional clustering algorithms. (b) A graph Laplacian of these points is calculated and the rows of its two principal eigenvectors, which we call $\{\mathbf{y}_i\}_{i=1}^N$, are shown as black dots. As can be seen in this plot, the eigenvector data form compact clusters compared with the original data. The resulting points $\{\mathbf{y}_i\}_{i=1}^N$ are clustered with K-means. (c) The obtained cluster membership information is applied to the original data. Note that the spectral clustering algorithm is sensitive to the choice of its kernel scale (here $\sigma = 0.03$).

sets. A detailed deduction of spectral clustering from graph theory can be found in appendix E. Furthermore, direct interpretations of spectral clustering as random walks [Meila and Shi, 2000] and as kernel PCA [Alzate and Suykens, 2006] (see section E.3) have been discovered. Compared with traditional clustering algorithms, spectral clustering is especially attractive since it is capable of solving complex clustering problems by only performing fairly simple algebraic operations. Therefore, the literature on spectral clustering is abundant (see for instance [Cristianini et al., 2000, Bach and Jordan, 2003, Kannan et al., 2004]) and it has already been applied successfully to problems in a large number of areas including machine learning, image segmentation, data mining and bioinformatics.

Although there exist various spectral clustering techniques that have slightly different implementations, they all share the same basic procedure [Ng et al., 2001, von Luxburg, 2006]. Assume we are given a set of data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ and a similarity measure $\kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0$. In accordance to the previous chapters, this similarity can be represented by a kernel function, and it should be chosen in such a way that it reflects the information about the problem as much as possible. For instance, if the goal is to retrieve dense clusters of connected points, the Gaussian kernel can be used. Subsequently, spectral clustering is performed as follows:

1. First, construct a similarity graph from the data points and obtain the corresponding kernel matrix $\mathbf{K}$ (here also called "similarity" or "affinity" matrix).

2. Second, compute a graph Laplacian matrix $\mathbf{L}$ from the similarity matrix. Graph Laplacians are matrices studied in the field of *spectral graph theory*

Algorithm 7.3 NJW Spectral Clustering.

```
Given the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ and the number of clusters to retrieve, $K$.
Calculate the affinity matrix $\mathbf{K}$: $K_{i,j} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ and $K_{ii} = 0$, for $i, j = 1, \ldots, N$.
Calculate the diagonal matrix $\mathbf{D}$: $D_{ii} = \sum_{j=1}^{N} K_{ij}$.
Obtain the graph Laplacian $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$.
Construct $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K]$ containing the first $K$ eigenvectors of $\mathbf{L}$.
Denote by $\mathbf{y}_i$ the rows of $\mathbf{V}$ and normalize them to unit length.
Eigenvector clustering: cluster the points $\mathbf{y}_i$ with $k$-means.
Assign the original point $\mathbf{x}_i$ to cluster $j$ only if $\mathbf{y}_i$ was assigned to cluster $j$.
```

[Fiedler, 1975, Chung, 1997]. In order to construct a graph Laplacian, denote by $\mathbf{D}$ the diagonal matrix whose i-th element is the sum of all similarities with the point $\mathbf{x}_i$, i.e. $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$, where $d_i = \sum_j^N \kappa(\mathbf{x}_i, \mathbf{x}_j)$. Commonly used graph Laplacians include the *unnormalized graph Laplacian* $\mathbf{L} = \mathbf{D} - \mathbf{K}$ and the *symmetric normalized graph Laplacian* $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$.

3. Third, calculate the K first eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ of the graph Laplacian, corresponding to the K largest eigenvalues, where K is the number of clusters to retrieve. Construct a matrix $\mathbf{V} \in \mathbb{R}^{N \times K}$ that contains the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ as columns, and let $\mathbf{y}_i \in \mathbb{R}^K$ be the vectors corresponding to the i-th row of $\mathbf{V}$, for $i = 1, \ldots, N$.

4. Finally, cluster the points $\mathbf{y}_i$ with the k-means algorithm into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$. The original point $\mathbf{x}_i$ is assigned to cluster j if and only if $\mathbf{y}_i \in \mathcal{C}_j$.

The basic spectral clustering algorithm is illustrated in Fig. 7.1. Although the last step of spectral clustering consists of a standard k-means algorithm, notice that k-means alone would not succeed on the data of Fig 7.1 (a) without the described preprocessing. Specifically, spectral clustering transforms the data $\mathbf{x}_i$ into the points $\mathbf{y}_i$ which are much easier to cluster, thanks to the properties of the graph Laplacian [Chung, 1997].

7.2.1 NJW algorithm

A popular implementation of spectral clustering is the Ng-Jordan-Weiss (NJW) algorithm, introduced in [Ng et al., 2001]. This algorithm uses the symmetric normalized graph Laplacian $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{K}\mathbf{D}^{-1/2}$, which penalizes small clusters, and it carries out an additional normalization of $\mathbf{y}_i$ to increase the robustness of the final k-means clustering stage. The resulting NJW algorithm is effective in a large number of different clustering problems, and similar to other spectral clustering algorithms it is also very easy to implement. It is summarized in Alg. 7.3. We refer to appendix E for more details on this algorithm.

7.2.2 Parameter choice and implementation

Although spectral clustering algorithms allow to solve very complex clustering tasks with straightforward algebraic operations, they show a number of difficulties. In particular, they have a high computational complexity and require the choice of the kernel parameter, which are problems related to all kernel methods. Additionally, they require to determine the number of clusters, which is a problem inherent to clustering.

The high computational complexity of spectral clustering is a problem exhibited by most kernel methods. In particular, spectral clustering requires the calculation of the SVD of an $N \times N$ matrix, which involves a computational complexity of $O(N^3)$. Recently several methods have been proposed to speed up spectral clustering, such as those based on the incomplete Cholesky decomposition method from [Alzate and Suykens, 2008], or the "parallel spectral clustering" algorithm from [Song et al., 2008], which uses a sparse similarity matrix (nearest neighbors). Alternatively, any of the complexity-limiting techniques presented in section 2.4 could be used.

Another important issue is the determination of a suitable kernel parameter, such as the kernel "width" σ in case of the Gaussian kernel. As has been shown in a number of publications [Ng et al., 2001, Zelnik-Manor and Perona, 2004], the clustering result is highly sensitive to the value of this parameter. This problem is directly related to the more general problem of kernel density estimation (KDE). A number of well-founded algorithms have been proposed to determine this parameter including Silverman's rule (2.64), local scaling [Zelnik-Manor and Perona, 2004] and others [Ng et al., 2001, Bach and Jordan, 2003].

A few techniques have been proposed to determine the number of clusters to retrieve, including some rules-of-thumb (see for instance [Bach and Jordan, 2003]), although in a lot of applications this number is known. In this work the number of clusters to retrieve is known in all considered clustering problems.

7.2.3 Self-tuning spectral clustering

As mentioned in the introductory chapter 2 of this thesis, the design of a kernel can include information about a point's local neighborhood. To this end, in [Zelnik-Manor and Perona, 2004] the elements of the affinity matrix are calculated as

$$K_{ij} = \exp(-d^2(\mathbf{x}_i, \mathbf{x}_j)/(\sigma_i \sigma_j)) \tag{7.2}$$

where the "local scale" $\sigma_i = d_m(\mathbf{x}_i, \mathbf{x}_L)$ is calculated as the distance between $\mathbf{x}_i$ and its L-th closest neighbor. This ad-hoc measure determines for each point a scale that adjusts to its own neighborhood, and therefore it can be used to cluster points together if they belong to "similar adjacent neighborhoods". According to [Zelnik-Manor and Perona, 2004], the selection of L is independent of scale and it is a function of the data dimension of the embedding space. Furthermore, a slightly more robust algorithm can be obtained by selecting a higher L and calculating σ_i as the median distance between $\mathbf{x}_i$ and its L-th neighbor. We will use this measure in the following chapters.

7.3 Clustering in Source Separation

In source separation, clustering can be used to solve mixture problems in which the source signals either belong to a finite alphabet or where they are sparse signals.

The first scenario occurs for instance in multiple-input multiple-output (MIMO) communication systems, in which one or more antennas are used at transmitter and receiver side. Since the transmitted symbols belong to a finite alphabet (the constellation), the received data points will form clusters in the scatter plot corresponding to the sent data symbols. A number of algorithms have been proposed to retrieve these clusters.

In a standard setting, variations of the communication channel during the transmission of one block of symbols are so small that they can be ignored or easily compensated for. In chapter 8 we will focus on situations in which the MIMO system is *fast time-varying*. Such situations can occur in mobile communications where transmitter and/or receiver are moving at high speed. Depending on the *Doppler spread* of the channels, the clusters in the system's scatter plot will show different degrees of overlap. Due to this overlap, traditional clustering algorithms will no longer be able to decode the sent symbols. In chapter 8 we will design decoding procedures based on spectral clustering for this task.

The second scenario we will study is the underdetermined blind source separation problem in which the sources are sparse signals. For such signals, a scatter plot of the mixtures reveals axes that correspond the columns of the mixing matrix. If the number of mixtures is insufficient to apply standard blind source separation techniques, conventional clustering techniques can be applied to retrieve the directions of these axes and identify the mixture process. However, if the mixture process of sparse signals is nonlinear, the scatter plot of the mixed signals will show curves instead of straight lines. Here, spectral clustering can be applied to find the position of these curves, after which the problem can be reduced to a standard linear mixture problem. This will be the topic of chapter 9.

7.4 Conclusions

In this chapter we introduced the problem of clustering and we discussed a number of techniques for this task. In particular, we highlighted some of the properties of the recently proposed spectral clustering technique. This kernel-based algorithm is capable of performing complex clustering tasks with basic algebraic operations. We also discussed two standard scenarios in source separation where clustering algorithms are required and pointed out situations that can no longer be treated by conventional clustering algorithms. In the next chapters we will study these problems in detail and design spectral clustering based techniques to solve them.

Chapter 8
Blind Decoding of Fast Time-Varying MIMO Channels

In this chapter we address the problem of blind decoding of multiple-input multiple-output (MIMO) systems that use M-PSK modulations. Since the transmitted symbols belong to a finite alphabet (the *constellation*), the received data points will form clusters corresponding to the transmitted data symbols. If such a MIMO system is time-varying, the clusters in its scatter plot can overlap, and algorithms based on traditional clustering cannot be applied.

We show that specific versions of spectral clustering can be designed to tackle these problems. By adding a temporal dimension to the received data, intertwined threads appear in its scatter plot. These can be clustered by a spectral clustering algorithm that relies on the geometry of the used constellation. After obtaining clusters that represent the different transmitted data symbols, the decoding problem consists of assigning a symbol to each cluster, based on known (pilot) symbols.

8.1 Introduction

In the last decades, multiple-input multiple-output (MIMO) wireless communication technology has gained considerable attention due to its potential to significantly increase spectral efficiency compared with traditional single-input single-output (SISO) technology.

A number of computationally efficient algorithms have been proposed for reliable symbol detection in flat-fading MIMO systems, based on the assumption that the MIMO channel is static and known at the receiver side, such as the Vertical Bell Laboratories Layered Space-Time (V-BLAST) architecture [Foschini et al., 1999]. Nevertheless, their direct application in time-varying environments is difficult, due to the need of perfect state information at the receiver side [Karami and Shiva, 2006]. A few adaptive equalization algorithms have been proposed to resolve this issue [Choi et al., 2005, Rontogiannis et al., 2006, Karami and Shiva, 2006, Kekatos et al., 2007] or the model-based approach in [Liu and Giannakis, 1998]. All of these techniques are *supervised* equalization algorithms, requiring an initialization phase in which a number of pilot symbol slots are transmitted.

An alternative approach to equalization can be based on clustering, exploiting the fact that, for channels excited by signals belonging to a finite alphabet, the noisy observations tend to cluster around a finite number of *channel states*. Since the pioneering work by Chen, Mulgrew et al. [Chen et al., 1993, Chen et al., 1995] these techniques have been widely applied in communications. Once the channel states or cluster centroids have been estimated (typically by applying any supervised clustering technique based on a known training sequence), channel equalization reduces to a classification problem, which can be solved efficiently using a radial basis function network [Chen et al., 1993, Cid-Sueiro et al., 1994, Lee et al., 1999a], or a support vector machine [Sebald and Bucklew, 2000, Mitchinson and Harrison, 2002, Pérez-Cruz et al., 2001, Santamaría et al., 2003]. From a more general point of view we can say that the problem of separating linear mixtures of discrete-alphabet inputs reduces to the problem of finding and labeling a finite (and known in advance) number of clusters. It is also worth mentioning that these clustering methods consider only time-invariant environments.

The main drawback of these algorithms is that the number of clusters grows exponentially as M^{N_t}, where M is the constellation size and N_t is the number of transmit antennas. During the last years, a lot of work has been directed towards reducing the number of clusters to be estimated. This reduction can be achieved by estimating only those cluster centers that lie close to the decision border [Lee et al., 1999a, Santamaría et al., 2003], or by exploiting the symmetries of the channel states inherited from the symmetries of the input constellation [Montalvão Filho et al., 2002, Kopsinis and Theodoridis, 2003]. Also, some extensions to multiple-input single-output channels have been recently proposed in [Diamantaras, 2006, Diamantaras and Papadimitriou, 2006].

In time-varying systems, the variations in the mixing matrix provoke a movement of the cluster centers and, consequently, the clusters adopt non-convex shapes and overlap each other. Conventional clustering algorithms such as k-means, fuzzy k-means and expectation-maximization (EM) learning, which typically require well-separated clusters, would fail. A promising alternative is the recently proposed spectral clustering technique [Ng et al., 2001], which is capable of clustering non-convex data sets (see chapter 7).

In [Van Vaerenbergh et al., 2007a, Van Vaerenbergh and Santamaría, 2008, Van Vaerenbergh et al., 2009] we initially presented and subsequently improved a clustering technique that can deal with *fast* time-varying systems. Such systems show normalized Doppler frequencies $f_d T \geq 0.001$, where f_d is the Doppler frequency and T the transmission rate. The proposed technique relies on three key principles. First, by adding a temporal dimension to the scatter plot, it converts the overlapping clusters into elongated threads, which can be clustered using a standard form of spectral clustering. Second, the geometry of the constellation is taken into account to reduce the number of clusters to retrieve, which, under normal circumstances, grows exponentially. This procedure exploits the symmetries of the cluster centers to reduce the number of clusters, therefore reducing the computational cost and

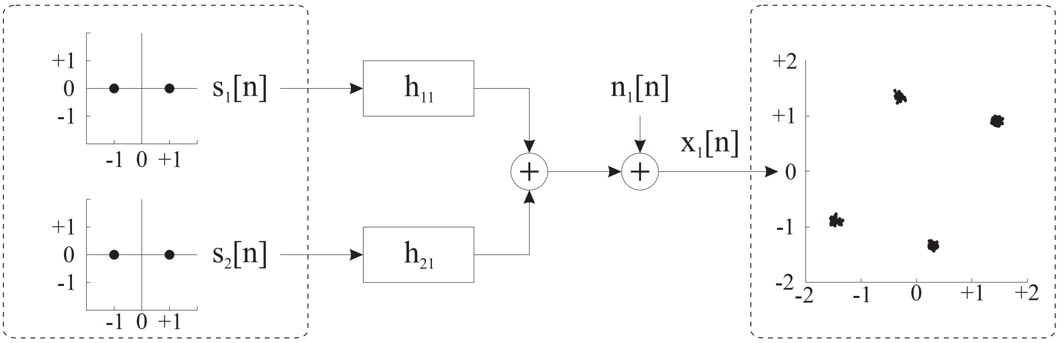


Figure 8.1: A BPSK MIMO system with constant, flat-fading channels. A transmitter with two antennas is shown on the left. For clarity, only one receiver antenna is included in this diagram. Its scatter plot, depicted on the right, shows compact clouds of points, which correspond to the transmitted data symbols.

extending the applicability of the method. And third, a specific path-based similarity function, the *connectivity kernel*, is adopted to incorporate more information on the problem into the clustering procedure. In particular, this kernel function favors elongated clusters. The proposed clustering technique is fully unsupervised in that no knowledge of a training sequence is required (i.e., a blind technique). Only a few pilots (known symbols) are needed to label the clusters and to decode the transmitted data. The obtained spectral clustering method is capable of finding clusters in sequential data, and it achieves better results than state-of-the-art MIMO decoding techniques for time-correlated channels such as the generalized decision feedback equalizer proposed by Choi et al. [Choi et al., 2005].

In this chapter we discuss this technique in detail, and we apply it to the problem of decoding time-varying multiple-input multiple-output (MIMO) channels. We present an efficient scheme to calculate the connectivity kernel for sequential data, as well as a procedure to select its kernel scale. Some extensions to this method will also be discussed. For instance, the use of orthogonal space-time block coded (OSTBC) MIMO schemes, such as the popular Alamouti encoding [Alamouti, 1998], allows us to exploit the structure imposed by the code in the clustering stage. For fast time-varying channels, up to certain Doppler spreads, we obtain better results than a space-time differential code. Also, further enhancements can be obtained by optimizing the final clustering stage of the spectral clustering algorithm. In particular, the order of the data can be taken into account again in this final clustering stage to avoid invalid solutions.

8.2 Problem Statement

In a typical MIMO flat-fading system with N_t transmit and N_r receive antennas (such as in Fig. 8.1), the N_r-dimensional received vector $\mathbf{x}[n] = [x_1[n], \ldots, x_{N_r}[n]]^T$ at time n is expressed as

$$\mathbf{x}[n] = \mathbf{H}[n]\mathbf{s}[n] + \mathbf{v}[n] \tag{8.1}$$

where $\mathbf{H}[n]$ is the complex $N_r \times N_t$ channel matrix whose elements represent independent flat-fading SISO channels, $\mathbf{s}[n]$ contains the N (in general, complex) symbols transmitted by the N_t antennas at time n, and $\mathbf{v}[n]$ represents spatially and temporally white complex zero-mean Gaussian noise. The goal of blind symbol decoding is to estimate the symbols $\mathbf{s}[n]$ given only the received data points $\mathbf{x}[n]$.

In MIMO systems with *block-fading* channels, variations of the channel during the transmission of one block of symbols are so small that they can be ignored. Hence the channel matrix $\mathbf{H}[n] = \mathbf{H}$ is considered constant during transmission of one block of symbols. This is not the case for MIMO systems with *fast time-varying* channels, where the channel matrix changes from symbol to symbol due to the Doppler spread caused by the movement of the transmitter and/or receiver. In such systems, depending on the Doppler spread, the channel matrices $\mathbf{H}[n]$ are temporally correlated. These variations can be modeled for instance by the Clarke and Gans Fading Model [Rappaport, 2001] which states that if a vertical $\lambda/4$ antenna with uniform power distribution is used to transmit a single tone, the received spectrum is

$$S_{E_z}(f) = \frac{1.5}{\pi f_m \sqrt{1 - \left(\frac{f-f_c}{f_m}\right)^2}}, \tag{8.2}$$

where f_c and f_m are the carrier frequency and the maximum Doppler shift, respectively.

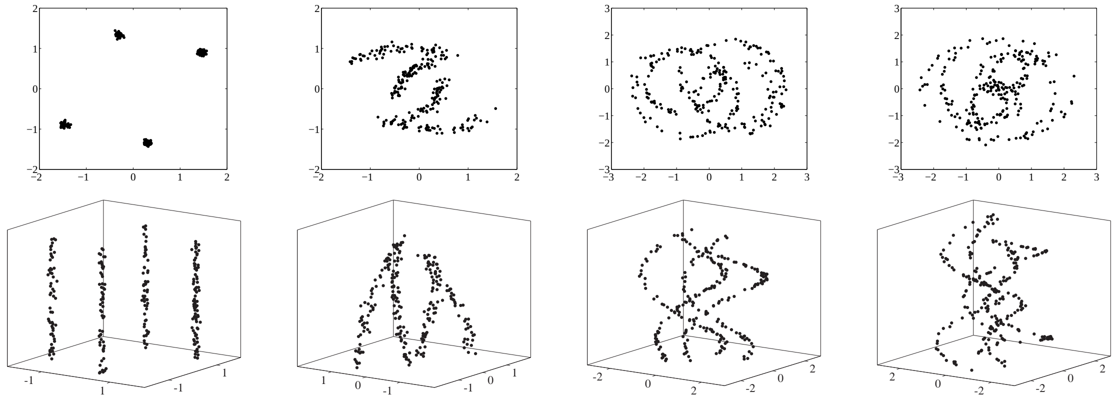
The top row of Fig. 8.2 illustrates the effect of different Doppler spreads on the scatter plot of a received data block in a typical binary phase-shift keying (BPSK) MIMO system, for which the basic constellation points are $d \in \{+1, -1\}$. Whereas the received data in a static system contains clearly separable clusters (Fig. 8.2(a) top), in a time-varying system these will overlap (Fig. 8.2(b), (c) and (d) top) and classical clustering algorithms that operate directly on the data will fail.

8.3 Key Geometrical Ideas of the Clustering Approach

As discussed in Section 8.2, the data points in a time-varying MIMO communication system may form non-convex and even overlapping clusters in the scatter plots, impossible to cluster with classical algorithms. Spectral clustering can deal with the non-convexity of these groups, but it will fail when the data clusters overlap. In [Van Vaerenbergh et al., 2007a] a simple workaround was presented to circumvent this problem, consisting in making use of the temporal index, which is usually discarded for scatter plot analysis. It is discussed in detail in the following section.

A second problem arises when the number of points per cluster is too small. On one hand, the clustering algorithm deals with one block of data at a time, containing typically $N = 256$ data symbols. On the other hand, the number of clusters to retrieve is M^{N_t}, where M is the size (the *cardinality*) of the source alphabet and N_t is number of transmit antennas[1]. Both M and N_t need to be small to guarantee a

[1] Note that the channel length $L = 1$ since we are dealing with flat-fading channels.



(a) $f_d T = 0$ (b) $f_d T = 0.001$ (c) $f_d T = 0.005$ (d) $f_d T = 0.01$

Figure 8.2: Effect of different normalized Doppler spreads on the received data symbols. Top row: Scatter plots of the data received by one receive antenna for a BPSK MIMO system with 2 transmit antennas, for different values of the normalized Doppler frequency $f_d T$. Bottom row: Scatter plots to which the time index was added as an additional vertical axis. Due to the channel changes during the transmission of the data block, curved threads appear in this plot.

sufficient number of points per cluster, which is usually not the case. However, as will be demonstrated, it is possible to exploit the constellation geometry of M-PSK systems to make the data clusters more dense and consequently easier to retrieve.

Finally, we would like to restrict the retrieved clusters to structures that are elongated and thread-like. Although it is difficult to implement this into spectral clustering as a hard constraint, a kernel that *favors* such elongated structures can easily be designed. The proposed approach uses the *connectivity* kernel from [Fischer et al., 2003] which is optimized to take into account the order of the data. This kernel function will be discussed in section 8.4.

8.3.1 Adding the temporal dimension into the clustering problem

The received data $\mathbf{x}[n]$ in a fast time-varying MIMO system can be preprocessed for spectral clustering by simply adding the temporal dimension into the vector of observations. The combined vector representing one data point and its temporal index $t[n]$ can be denoted as

$$\mathbf{x}^0[n] = \left[\mathbf{x}[n]^T, t[n]\right]^T, \tag{8.3}$$

which is a complex vector with $N_r + 1$ elements[2]. When this extra dimension is added to the scatter plots of Fig. 8.2 (top), threads appear due to the temporal correlation between consecutive channel matrices (see Fig. 8.2 (bottom)). Thanks to its capabilities to cluster non-convex data sets, spectral clustering should be able to retrieve these different threads from $\mathbf{x}^0[n]$.

[2] The super-index 0 is used to distinguish it from rotated versions of this vector, see Sec. 8.3.2.

8.3.2 Exploiting the input constellation symmetries

In clustering it is important that, in order to be detected, each cluster should have a certain minimum number of points. A rule of thumb is to have at least 10 samples per cluster. And since spectral clustering is a computationally costly procedure, this number of clusters, which is M^{N_t}, should be reasonably low. Taking into account that most commercial MIMO systems use up to $N_t = 4$ transmit antennas, we will only treat BPSK systems ($M = 2$) and QPSK systems ($M = 4$) in this work.

Clusters should also be well connected, i.e., the distance between neighboring points of the same thread should not be larger than the distance between points of different threads. This requires a rescaling of the temporal dimension to match the scale of the spatial dimensions, for instance $t[n] = n/256$, with $n = 1, \dots, 256$ for blocks of 256 symbols. Moreover, this means that if a symbol is not transmitted during a considerable time, one thread might be incorrectly identified as two separate threads. However, as will be shown in the next section, both difficulties can be reduced by using information derived from the geometric properties of the constellation.

In this section we show that the geometrical symmetries of the transmitted constellation can be used to design a two-phase clustering algorithm, in which only a reduced number of clusters needs to be detected during each phase. Before dealing with the general case of $N_t \times N_r$ M-PSK MIMO systems, the proposed algorithm is illustrated on a simple 2×2 BPSK MIMO system.

Case of BPSK MIMO systems

In the noiseless case ($\mathbf{v}[n] = \mathbf{0}$), Eq. (8.1) can be written as

$$\mathbf{x}[n] = \mathbf{H}[n]\mathbf{s}[n]. \tag{8.4}$$

For a 2×2 BPSK MIMO system, there will be 4 symbol clusters to detect in the data $\mathbf{x}[n]$, corresponding to the transmitted symbol vectors $[+1, +1]^T$, $[+1, -1]^T$, $[-1, +1]^T$ and $[-1, -1]^T$. In Fig. 8.2 we can observe that for any cluster following a certain trajectory, there is always another cluster following a trajectory symmetric with respect to the origin. This observation is confirmed by (8.4): since a BPSK system can generate both $\mathbf{s}[n]$ and $-\mathbf{s}[n]$, the data point $\mathbf{x}[n]$ as well as its opposite $-\mathbf{x}[n]$ can be received. When transmitted through a time-varying channel, these data points lie in clusters that follow symmetric trajectories with respect to the introduced "temporal axis". This property will be exploited to improve the spectral clustering stage, by first grouping together the data points that follow symmetric trajectories.

This geometrical property is not limited to 2×2 systems. In a general $N_t \times N_r$ BPSK MIMO system with fast time-varying channels, for any cluster following a certain trajectory in time, there is always another cluster following the symmetric trajectory. Combining the data of two such clusters will provide a more robust clustering problem. This observation leads to the following two-phase algorithm: In the first phase, groups of symmetric clusters are detected. One clustering problem needs to be solved here to find 2^{N_t-1} clusters. In the second phase, each group of symmetric clusters is separated into two different clusters. This second phase consists of 2 independent problems.

Clustering procedure for $N_t \times N_r$ BPSK MIMO systems

Phase 1: Grouping of symmetric clusters. The following analysis shows how spectral clustering can be extended to find clusters consisting of two symmetric "subclusters" at a time. Similar to (8.3) we can introduce the "symmetric" data point

$$\mathbf{x}^1[n] = \left[-\mathbf{x}[n]^T, t[n]\right]^T, \tag{8.5}$$

which contains the data point opposite to $\mathbf{x}^0[n]$ with respect to the temporal axis. This new data point does not correspond to any real data point and can therefore be considered a "virtual" pattern. It will only be used to facilitate the clustering process and the superscript 1 refers to the fact that it is the *first* of these virtual patterns. Other constellations such as QPSK need more virtual patterns per original data point, as will be seen below.

Consider the distance measure

$$d(\mathbf{x}[i], \mathbf{x}[j]) = \min\left(\left|\mathbf{x}^0[i] - \mathbf{x}^0[j]\right|^2, \left|\mathbf{x}^0[i] - \mathbf{x}^1[j]\right|^2\right). \tag{8.6}$$

This measure is small in two cases: firstly for points that are very close to each other, and secondly for points that are very close to opposite of each other. If a Gaussian kernel is used with this distance measure for spectral clustering, neighboring points as well as opposite points will be grouped together, leading to 2^{N_t-1} clusters. This first phase avoids the incorrect clustering that might occur when some of the threads have a low number of data points, by combining the information of symmetric clusters.

Phase 2: Retrieving the individual clusters. After having identified the 2^{N_t-1} groups of symmetric clusters, the two individual threads for each group need to be retrieved. Since these sub-clusters are now separated from the other clusters, the clustering problem is greatly simplified. Problems might occur, however, if one or both clusters contain few data points. Luckily, the constellation geometry can be exploited again in this second phase of the problem, making use once more of the symmetry of the sub-cluster's trajectories.

Specifically, we can increase the number of points in each cluster by adding the symmetric virtual patterns $\mathbf{x}^1[n]$ to the data points $\mathbf{x}^0[n]$ that need clustering. As shown above, these symmetric points will lie in one of the two clusters to retrieve, and their presence will only make the original clusters more dense. As a result, the performance of the clustering algorithm will improve. Taking this into account, the second clustering phase consists of the following steps:

1. Expand the set of points to cluster, $\mathbf{x}^0[n]$, with their (opposite) virtual points $\mathbf{x}^1[n]$.
2. Apply spectral clustering to these points.
3. Discard the virtual points to obtain clusters consisting only of the original points $\mathbf{x}^0[n]$.

Thanks to this procedure, even if one of the clusters has few nearby samples, tends to have “holes”, or is even empty, it is still possible to apply spectral clustering to retrieve the two clusters. In the case of an “empty” cluster the second step should correctly leave one of the clusters empty and only identify its position.

Generalization to M-PSK MIMO systems

The described procedure to exploit the constellation geometry can be easily extended to M-PSK constellations. In this general case, M^{N_t} clusters need to be retrieved, which boils down to finding M^{N_t-1} clusters in the first phase and M sub-clusters in the second phase.

In case of BPSK systems the first phase of the clustering algorithm consisted in grouping together clusters that follow symmetric trajectories in time. The main difference for M-PSK systems is that now the clusters to retrieve should follow trajectories that are rotated over a certain angle α with respect to the introduced temporal axis. For QPSK systems this angle will be $\alpha = \pi/2$, as can be deduced easily from its constellation. The special case of BPSK systems is found for $\alpha = \pi$. In general, we have that $\alpha = 2\pi/M$, where M is the size of the constellation.

The clustering algorithm consists of the same two phases as for BPSK systems. The only difference is that not 1 *symmetric* virtual pattern but $M-1$ *rotated* virtual patterns should be taken into account in both phases. Instead of (8.6), the distance measure to be used in phase 1 is

$$d(\mathbf{x}[i], \mathbf{x}[j]) = \min_k \left[\|\mathbf{x}^0[i] - \mathbf{x}^k[j]\|^2 \right], \quad k = 0, \ldots, M-1, \tag{8.7}$$

where

$$\mathbf{x}^k[n] = \left[e^{j \cdot 2k\pi/M} \mathbf{x}^T[n], t[n] \right]^T \tag{8.8}$$

are rotated versions of the data point $\mathbf{x}^0[n]$ with respect to the temporal axis. For $k > 0$ the points $\mathbf{x}^k[n]$ represent the virtual patterns of $\mathbf{x}^0[n]$. In phase 2, all $M-1$ virtual patterns per data point should be added to the clustering problem.

8.3.3 Symbol decoding

Once the symbol clusters have been successfully retrieved, the original time-varying problem has been reduced to a simpler decoding problem, which is the only supervised part of the proposed algorithm. Each cluster should be mapped to a constellation symbol, and to this end a small number of pilot symbols $\mathbf{s}[n]$ are transmitted at the start of the symbol block. Specifically, N_t pilots must be transmitted during N_t slots[3]. Note that these pilot symbol slots are not needed for the clustering process. Therefore it is not required that they are included at the beginning of the data block. In practice, better results are obtained if the pilots are transmitted in the middle of the block.

Defining the matrix of pilot symbols $\mathbf{S}_p = [\mathbf{s}[i_1], \mathbf{s}[i_2], \ldots, \mathbf{s}[i_{N_t}]]$ and the matrix of corresponding received data $\mathbf{X}_p = [\mathbf{x}[i_1], \mathbf{x}[i_2], \ldots, \mathbf{x}[i_{N_t}]]$, an approximation of the channel matrix $\mathbf{H}$ around the position of the pilot symbols can be obtained as

$$\hat{\mathbf{H}} = \mathbf{X}_p \mathbf{S}_p^{-1}. \tag{8.9}$$

The symbol decoding stage of the algorithm consists in assigning the symbol slot $\mathbf{s}$ to the cluster that contains the data points closest to the vector $\hat{\mathbf{H}}\mathbf{s}$, taken into account the extra temporal dimension.

An overview of the complete algorithm will be given in Section 8.5.

8.4 Optimizing the Kernel Function

The previously described spectral clustering based technique obtains very satisfactory results. In cases of high noise or high Doppler spread, however, it does not exclude invalid solutions in which clusters are bifurcated or two threads are clustered as one. Therefore, a restriction needs to be built into the clustering procedure to avoid clusters that do not consist of single threads. In this section we present a kernel function that favors clusters of elongated shape.

8.4.1 Path-based spectral clustering

Path-based clustering [Fischer et al., 2003, Ozertem et al., 2008] is a recently developed technique for clustering groups of points that are elongated in addition to being dense. In graph theory, a path p in a graph $\mathcal{G}$ is an alternating sequence of vertices and edges, beginning and ending with vertices, in which all vertices are distinct and each edge is incident with the vertex immediately preceding it and with the vertex immediately following it.

Let us denote by $\mathcal{P}_{i,j}$ the set of all paths from vertex i to vertex j. When dealing with elongated structures, two points should be considered similar if there is a clear path between them, in the sense that all of its edges are short. For instance, in Fig. 8.3, the vertices i_1 and j belong to the same cluster. This is reflected in the fact that there is a path between them that consists only of short edges. On the other hand, the vertices i_2 and j belong to different clusters, and any path connecting them will contain at least one longer edge.

Based on this observation, in [Fischer et al., 2003] a kernel function was proposed that calculates the similarity between two vertices i and j based on the *weakest*

[3] The number N_t is a theoretical minimum. In order to improve performance in certain situations, more pilot symbols could be used.

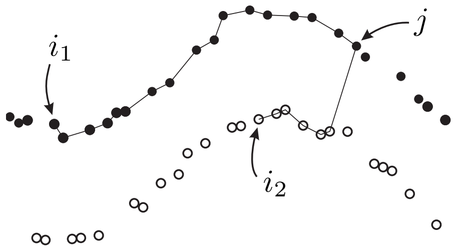


Figure 8.3: Path-based similarity: Due to the optimal paths generally following dense regions of data, vertices i_1 and j are considered more similar than i_2 and j.

link of the best path between them. The *weakest link* of a path p is considered to be its longest edge. We will denote the length of this edge as the "effective distance" of the path, which can be written as

$$\bar{d}^p_{i,j} = \max_{(k,l)\in p} d_{k,l}. \tag{8.10}$$

The *best path* between the two vertices will be the one whose effective distance is shortest, and we denote the *effective distance between the two vertices* as

$$\bar{d}_{i,j} = \min_{p\in\mathcal{P}_{i,j}} \bar{d}^p_{i,j}. \tag{8.11}$$

Based on this metric, the similarity between vertices i and j can be expressed using the Gaussian kernel as

$$\begin{aligned}\kappa_c(\mathbf{x}_i, \mathbf{x}_j) &= \exp\left(-\frac{\bar{d}^2_{i,j}}{\sigma^2}\right) &(8.12)\\ &= \exp\left(-\frac{1}{\sigma^2}\min_{p\in\mathcal{P}_{i,j}}\max_{(k,l)\in p} d^2_{k,l}\right).\end{aligned}$$

This kernel function is called "connectivity" kernel.

Although the computation of the kernel function (8.12) contains a min-max operation over all possible paths between two vertices, a recursive scheme can easily be applied to calculate the kernel function based on the result for shorter segments, for instance by using Dijkstra's shortest path algorithm [Dijkstra, 1959] (as discussed in appendix F). In the following we present an efficient procedure to calculate this kernel in the special case of sequential data.

8.4.2 Adaptation for sequential data

The application considered in this chapter deals with data symbols from communications, which are transmitted and received sequentially. While the previous section described the *general* path-based clustering method, the problem treated here deals with ordered data and therefore it will benefit from incorporating temporal information into the clustering process. Note that most kernel methods and clustering algorithms do not take into account the order of the training data.

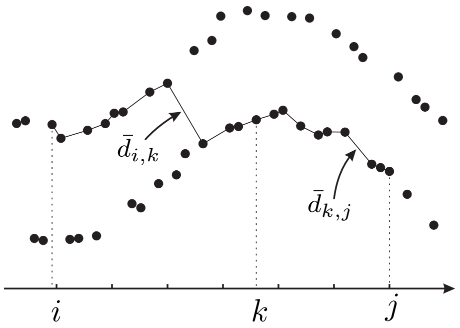


Figure 8.4: The effective distance between vertices i and j can be calculated by considering all combinations of two shorter optimal paths, from i to k and from k to j, with $k = i, \ldots, j$.

The connectivity kernel can respect the data order by only considering paths that are "monotonic" in the temporal dimension. In other words, the paths used in kernel (8.12) should either consist *only* of edges $(k, l) \in p$ that fulfill $l > k$, or *only* of edges that fulfill $k > l$. Moreover, this restriction greatly reduces the total number of paths that need to be taken into account, which is very convenient for problems from the area of communications.

The following scheme describes how to efficiently calculate the effective distances between all pairs of data points, resulting in an effective distance matrix $\bar{\mathbf{D}}$. Once obtained, the similarity between points can be obtained by calculating (8.12). Let us denote by $\delta = j - i$ the "temporal separation" between points $\mathbf{x}[i]$ and $\mathbf{x}[j]$, for $i, j = 1, \ldots, N$. We will fill the effective distance matrix $\bar{\mathbf{D}}$ one diagonal at a time, in an inductive manner:

1. $\delta = 0$: Elements on the diagonal of $\bar{\mathbf{D}}$ correspond to pairs of identical points, and therefore $\bar{d}_{i,i} = 0$, $\forall i$.

2. $\delta = 1$: Elements on the first upper diagonal are consecutive data points. In this case, the effective distance equals the real distance $\bar{d}_{i,i+1} = d_{i,i+1}$, $\forall i$.

3. $\delta = l$: Elements on the l-th upper diagonal can be calculated based on the results obtained for $\delta < l$. The optimal path between $\mathbf{x}[i]$ and $\mathbf{x}[j]$ is either a direct connection of $\mathbf{x}[i]$ and $\mathbf{x}[j]$, or a combination of two shorter paths, as illustrated in Fig. 8.4. The effective distance of the direct connection is simply its Euclidian distance, while the effective distance of a combination of shorter paths can be calculated as the maximal effective distance of its parts. The resulting effective distance between $\mathbf{x}[i]$ and $\mathbf{x}[j]$ is the minimum of the effective distances over these paths:

$$\bar{d}_{i,j} = \min\left(d_{i,j}, \max(\bar{d}_{i,i+1}, \bar{d}_{i+1,j}), \ldots, \max(\bar{d}_{i,j-1}, \bar{d}_{j-1,j})\right). \tag{8.13}$$

4. $\delta < 0$: Since the distance measure is symmetric, the lower part of $\bar{\mathbf{D}}$ can be copied from the upper part: $\bar{d}_{j,i} = \bar{d}_{i,j}$.

Algorithm 8.1 Spectral Clustering based Decoding of Time-Varying MIMO Channels.

```
initialize
    Obtain the data points $\mathbf{x}^0[n]$, for $i = 1, \ldots, N$.
    Construct the virtual patterns $\mathbf{x}^k[n]$, as in (8.8), for $k = 1, \ldots, M-1$.
spectral clustering phase 1:
    Use the connectivity kernel (8.12) to obtain the kernel matrix $\mathbf{K}$.
    Apply the NJW algorithm to retrieve $M^{N_t-1}$ clusters (see Alg. 7.3).
spectral clustering phase 2:
for each obtained cluster do
    Create a set with the data points $\mathbf{x}^0[n]$ from this cluster and all corresponding
    virtual patterns.
    Obtain the kernel matrix of this set, using the connectivity kernel (8.12).
    Apply the NJW algorithm to retrieve $M$ final clusters (see Alg. 7.3).
    Discard the virtual patterns from each cluster.
end for
Decode the data symbols: assign symbols to the clusters as in Sec. 8.3.3.
```

8.5 Proposed Algorithm

The final algorithm to decode fast time-varying M-PSK MIMO systems is summarized in Algorithm 8.1. Given the original data points $\mathbf{x}[n]$, it first adds the temporal index $t[n]$ to obtain the patterns $\mathbf{x}^0[n]$, and then constructs the $M-1$ virtual patterns $\mathbf{x}^k[n]$ per pattern by rotating them around the temporal axis. Next, it performs the described clustering algorithm in two phases, with the connectivity kernel, and finally it performs symbol decoding by assigning a data symbol slot to each cluster.

8.6 Extensions of the Core Algorithm

8.6.1 Exploiting additional information: OSTBC MIMO schemes

In the previous sections, the case of spatial multiplexing MIMO systems was discussed, in which N_t different symbols are transmitted by the N_t antennas at a given time instant. Recently, space-time block coding (STBC) [Alamouti, 1998] has emerged as one of the most promising techniques to exploit spatial diversity in multiple-input multiple- output (MIMO) systems. Among space-time coding schemes, the orthogonal space-time block coding (OSTBC) [Alamouti, 1998, Tarokh et al., 1999] is one of the most attractive because it is able to provide full diversity gain with very simple encoding and decoding. The most popular OSTBC is Alamouti coding [Alamouti, 1998]. It uses the following coding matrix to transmit a block of 2 symbols s_1 and s_2:

$$\mathbf{S} = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix}. \tag{8.14}$$

The Alamouti code uses 2 time slots to transmit 2 symbols, which makes it a *rate*-1 *code*. It uses 2 transmit antennas, where the first one sends the symbols $[s_1, \ -s_2^*]$ and the second one sends the symbols $[s_2, \ s_1^*]$. The sequences transmitted by both antennas are orthogonal, as can be easily verified:

$$[s_1, \ -s_2^*]^H \cdot [s_2, \ s_1^*] = s_1^* s_2 - s_2 s_1^* = 0. \tag{8.15}$$

The structure that the Alamouti coding adds to the data can be exploited in the clustering problem as follows. With Alamouti coding, each original symbol pair is transmitted in two consecutive time slots, where the second time slot contains a transformed version of these symbols. To exploit the redundancy in a clustering framework, new data points $\mathbf{x}^A[n]$ can be constructed as follows:

$$\mathbf{x}^A[n] = \begin{cases} \left[\mathbf{x}^T[n], \mathbf{x}^H[n+1]\right]^T, & \text{for } n = 1,3,5,\ldots \\ \left[\mathbf{x}^T[n], -\mathbf{x}^H[n-1]\right]^T, & \text{for } n = 2,4,6,\ldots \end{cases} \tag{8.16}$$

This transformation is based on the relationship of time slots: it adds extra components to the original data points $\mathbf{x}[n]$ that, due to the code structure, belong to the same "time-slot pair" and therefore contain the same information.

As with spatial multiplexing, the resulting data points $\mathbf{x}^A[n]$ will form data clusters, and it is not difficult to verify that the number of these clusters will be the same with or without transformation (8.16). However, the transformation causes an increase of the embedding dimensionality, which increases the distance between points of different clusters more than the distance between points of the same cluster. This greatly improves the clustering results, as will be illustrated in a practical experiment.

After embedding the received data according to (8.16), the two-phase spectral clustering process can be applied as previously indicated.

8.6.2 A self-tuning connectivity kernel

Since the clustering procedure is very sensitive to the kernel width σ, the algorithm proposed in [Van Vaerenbergh et al., 2007a] used a "local scaling" procedure for the Gaussian kernel. Instead of using a single global σ for all data points, a local kernel width σ_i was assigned to each point, equal to the median of distances to its K-th nearest neighbors [Zelnik-Manor and Perona, 2004], as described in Sec. 7.2.3. The same idea can be applied to increase the robustness of the connectivity kernel (8.12), leading to the following *locally scaled* connectivity kernel:

$$\kappa_{lc}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\bar{d}_{i,j}^2}{\sigma_i \sigma_j}\right). \tag{8.17}$$

This kernel is obtained in a straightforward manner by scaling each distance $d_{i,j}$ by the local scales σ_i and σ_j, before calculating the effective distance $\bar{d}_{i,j}$.

8.6.3 Optimizing the eigenvector clustering step of spectral clustering

The proposed algorithm favors elongated groups of points by making use of the connectivity kernel to calculate the kernel matrix. Ideally, each clustering procedure should retrieve thread-like clusters that span as much of the time range as possible, if the temporal dimension is taken into account. To rule out invalid solutions such as bifurcated threads, which can occur by plainly applying k-means at the final step of the spectral clustering algorithm, the temporal dimension can be added again to the obtained points and *hierarchical clustering* can be used to detect the clusters.

In agglomerative hierarchical clustering (AHC) (see section 7.1.1), each point is initialized as a cluster. Subsequently, the two closest clusters are joined, and this process is repeated until the desired number of final clusters is obtained. The distance between clusters is measured as the minimal Euclidian distance between any node of the first cluster and any node of the second cluster, which is known as "single linkage" hierarchical clustering. The additional constraint we add to avoid invalid solutions, is that if two clusters overlap in time, they are only linked if at least $m - 1$ additional clusters overlap in the same time fraction. If this is not fulfilled, the cluster pair is not merged but skipped in this iteration, and the next closest pair of clusters is considered.

8.7 Generalized Decision Feedback Equalizer

The generalized decision feedback equalizer (GDFE) algorithm from [Choi et al., 2005] is an adaptive algorithm for decoding fast time-varying MIMO systems, based on the V-BLAST architecture. For each time instant, the symbols are successively detected and canceled from the received data vector via decision feedback filtering. The filter tap weights and symbol detection order are updated using an RLS-based time- and order-update algorithm. Its complexity is $O(M^3)$ but it provides some savings compared with V-BLAST. During its training period, it needs to send a number of pilot symbols to initialize the algorithm. This method is known to suffer from numerical instabilities. Therefore, several improvements have been presented, for instance [Rontogiannis et al., 2006]. In the next section we will compare the performance of the original GDFE algorithm with the presented technique.

8.8 Simulation Results

A number of simulations were carried out to illustrate the performance of the proposed algorithms. In all experiments, the following parameters were assumed: the channels were independent Rayleigh flat-fading and the temporal variation of each channel between a transmit and receive antenna pair was based on the Clarke-Gans model [Rappaport, 2001]. The symbols $\mathbf{s}[n]$ were grouped into frames consisting of $N = 256$ slots.

8.8.1 Overview of the compared decoding techniques

A number of different decoding algorithms were compared, including a few variations of the proposed algorithm:

GDFE. The generalized decision feedback equalizer (GDFE) algorithm from [Choi et al., 2005], with $\lambda = 0.95$.

SPCL: KNN + *k*-means. A simplified version of the presented spectral clustering (SPCL) technique uses a K-nearest neighbor (KNN) kernel, the unnormalized graph Laplacian matrix (see section 7.2), and k-means eigenvector clustering.

SPCL: Gaussian kernel + *k*-means. The algorithm proposed in [Van Vaerenbergh et al., 2007a] uses a locally scaled Gaussian kernel, the symmetric normalized graph Laplacian matrix from the NJW algorithm (see section 7.2) and k-means.

SPCL: Connectivity kernel + *k*-means. For this algorithm the Gaussian kernel was replaced by a locally scaled connectivity kernel.

SPCL: Connectivity kernel + AHC. This algorithm uses spectral clustering with a locally scaled connectivity kernel, the normalized graph Laplacian from the NJW algorithm and agglomerative hierarchical clustering.

PCSI ML. As a reference lower bound error curve, we include results obtained by the optimum maximum likelihood (ML) decoding, which has perfect channel state information (PCSI).

For the adaptive GDFE method from [Choi et al., 2005], either N_t or 32 pilot symbols were used in the initialization phase. Its forgetting factor λ was fixed as 0.95. For all clustering algorithms, the optimal number of neighbors was determined experimentally as $K = 14$, and the scaling of the temporal axis was fixed as $t_n = 5 \cdot f_d T \cdot n$ with $n = 1, \ldots, 256$. Once the clusters were retrieved, the original symbols were decoded using N_t pilot symbols, placed in the middle of the frame.

8.8.2 BPSK systems with spatial multiplexing

In a first setup, a 2 × 2 system with a Doppler frequency of $f_d T = 0.005$ was considered. Fig. 8.5 shows the bit error (BER) rates for the compared algorithms. As can be

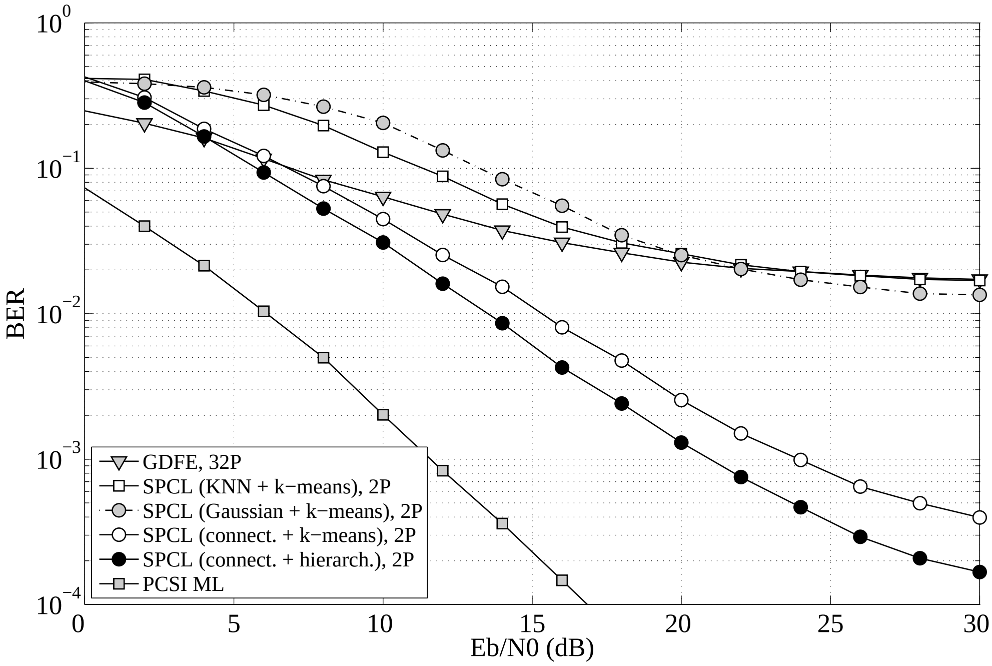


Figure 8.5: Performance comparison of different algorithms for a 2 × 2 BPSK system with $f_d T = 0.005$. The indications 2P and 32P stand for the number of pilot symbols used. As a lower bound for the blind algorithms, a BER curve for optimum maximum likelihood (ML) decoder is included, which has perfect knowledge of the channel state (PCSI).

observed, the algorithm from [Van Vaerenbergh et al., 2007a] (third curve) performs poorly for high noise levels, but reaches the same performance as the GDFE for low noise situations. Notice that the clustering algorithm only uses 2 pilot symbols (2P), while the GDFE algorithm uses 32 pilot symbols (32P) in this test. When the Gaussian kernel is replaced by the connectivity kernel, a dramatic increase in performance is seen, as shown by the fourth curve. An additional improvement in performance can be achieved when moreover the final *k*-means clustering method is replaced by the described hierarchical clustering method. Notice also that all compared algorithms have an error floor for higher SNR values. For the spectral clustering algorithms this is due to situations in which the randomly generated channel threads overlap and clustering fails. Such cases cannot be avoided even if no noise is present. An improvement in the clustering algorithm might consist in embedding gradient information in the used kernel.

We will now study the performance of these algorithms when one or more parameters of this basic problem setting are changed. Fig. 8.6 shows the change in performance due to different normalized Doppler spreads, for the spectral clustering algorithm using a Gaussian kernel with *k*-means clustering, and the connectivity kernel with hierarchical clustering. While there is an obvious decrease in MSE when the Doppler spread is lowered, notice also that the spectral clustering algorithm with

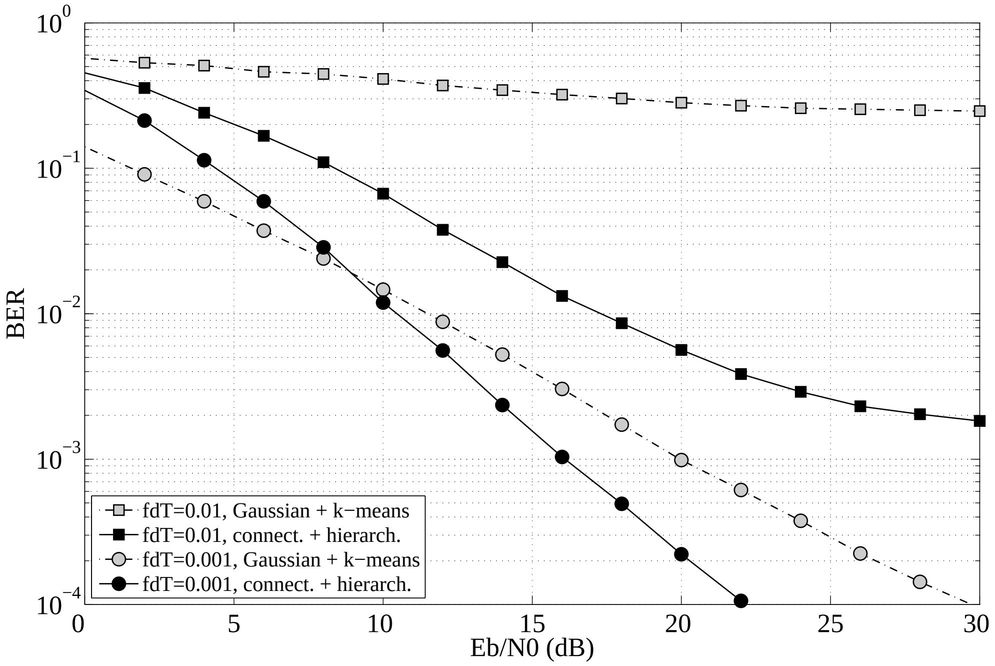


Figure 8.6: Comparison of the proposed method using a Gaussian kernel with k-means eigenvector clustering, and using the connectivity kernel with hierarchical eigenvector clustering, at different Doppler spreads, for a 2 × 2 BPSK system.

the connectivity kernel is much better at decoding the symbols for high normalized Doppler spreads and low noise ratios (third curve versus first curve). However, it is also observed that the connectivity kernel is less robust to noise at these higher Doppler spreads.

In Fig. 8.7 the effect of increasing the number of receive antennas N_t from 2 to 4 is shown. Note that that the performance of the GDFE degrades for lower noise levels in this case, due to numerical instabilities. This can be avoided for instance by the method described in [Kekatos et al., 2007].

The performance on a 4 × 4 system is shown in Fig. 8.8. Since this system represents a more complex scenario with $M^{N_t} = 16$ clusters to retrieve, we only present results for the (lower) normalized Doppler frequency $f_d T = 0.001$. When using only 4 pilot symbols, the GDFE algorithm performs considerably worse than the proposed spectral clustering methods.

8.8.3 QPSK systems with spatial multiplexing

In the second set of experiments, a QPSK signal was used in different MIMO system configurations. We only include results for the cases when the number of transmit antennas was $N_t = 2$, more specifically for a 2 × 2 and a 2 × 4 scenario. If 4 transmit antennas were used, the number of clusters would increase to $M^{N_t} = 4^4 = 32$ which

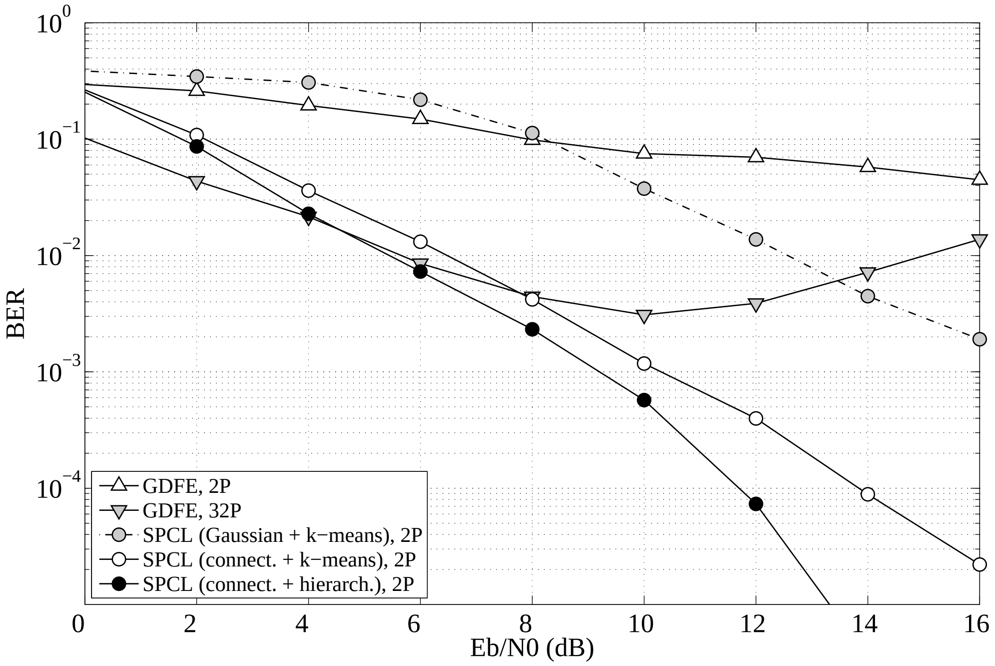


Figure 8.7: Performance comparison of different algorithms for a 2 × 4 BPSK system with $f_d T = 0.005$.

would result in clusters with an average of $256/32 = 8$ points. This would be insufficient to reliably detect clusters, even when exploiting the constellation geometry.

The proposed spectral clustering algorithms were applied as in the previous experiment, with 2 pilot symbol slots. In Fig. 8.9, their results are compared with the GDFE algorithm, for which first 2 pilot symbol slots were used, and this number of pilots was then changed to 32 to obtain similar performance.

In Fig. 8.10 the results for a 2 × 4 system are displayed, at different normalized Doppler spreads. For higher Doppler spreads and low noise ratios, these results show a clear advantage of using hierarchical clustering of the eigenvector data instead of *k*-means clustering. Nevertheless, for lower noise ratios it is less robust than when the *k*-means clustering is used. In this scenario the GDFE algorithm suffered from numerical instabilities (similar to those in the 2 × 4 BPSK case) and therefore its results are left out of this comparison.

8.8.4 MIMO systems with Alamouti coding

We now consider a 2 × 2 BPSK system with Alamouti-coded data. Here, the results of the spectral clustering algorithm were compared with the differential space-time block code detection (DSTBC) [Gao et al., 2002], which can also be decoded in a blind fashion (i.e. no channel state information is required). Both algorithms were used with N_t pilot symbols.

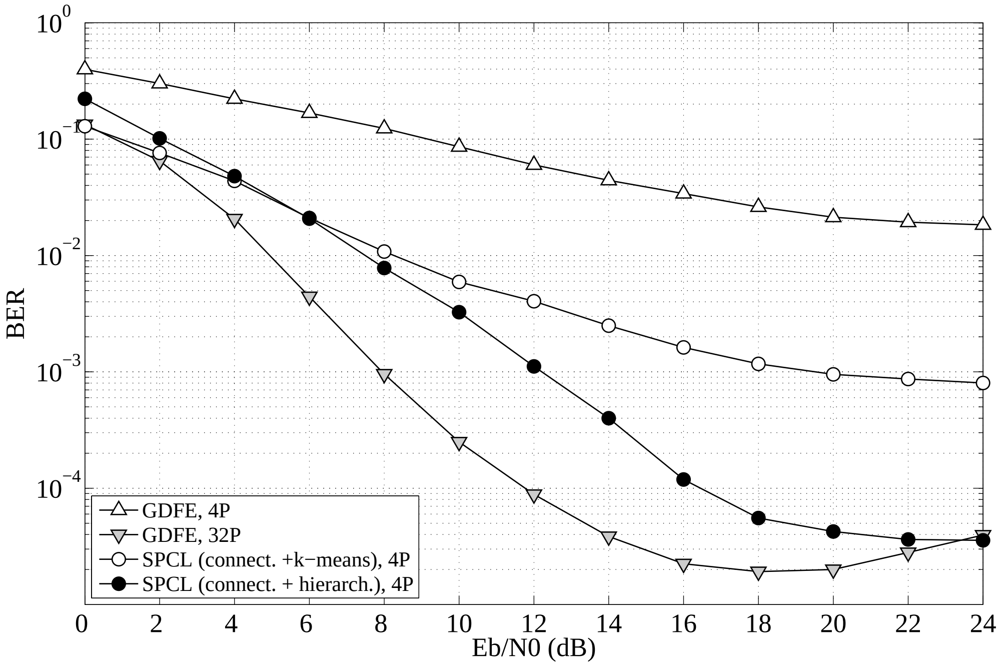


Figure 8.8: Performance comparison of different algorithms for a 4 × 4 BPSK system with $f_d T = 0.001$.

The results for a 2 × 2 BPSK system are displayed in Fig. 8.11. The proposed spectral clustering based is able to outperform the DSTBC algorithm only at certain high SNR levels.

8.8.5 Impulsive noise

As a final example, Fig. 8.12 shows the clustering result on a data set that is contaminated by impulsive noise. The noise pdf of this example is $p(v) = 0.9p_1 + 0.1p_2$, where p_1 and p_2 are zero-mean Gaussian white noise distributions with $Eb/N0 =$ 15dB and $Eb/N0 = -15$dB, respectively. If the outliers lie between the clusters, using the connectivity kernel can lead to incorrect clustering, since the outliers can constitute a path that connects two clusters [Fischer et al., 2003]. In such cases it would be more convenient to use a Gaussian kernel, whenever the normalized Doppler spread is acceptable. If the outliers do not lie in between the clusters, the connectivity kernel-based clustering is usually not disrupted. Fig. 8.12 illustrates the clustering in presence of impulsive noise. The top plot shows the result when 4 clusters are retrieved, and in the bottom plot 5 clusters are detected. In this last case the outliers are automatically grouped as a new cluster.

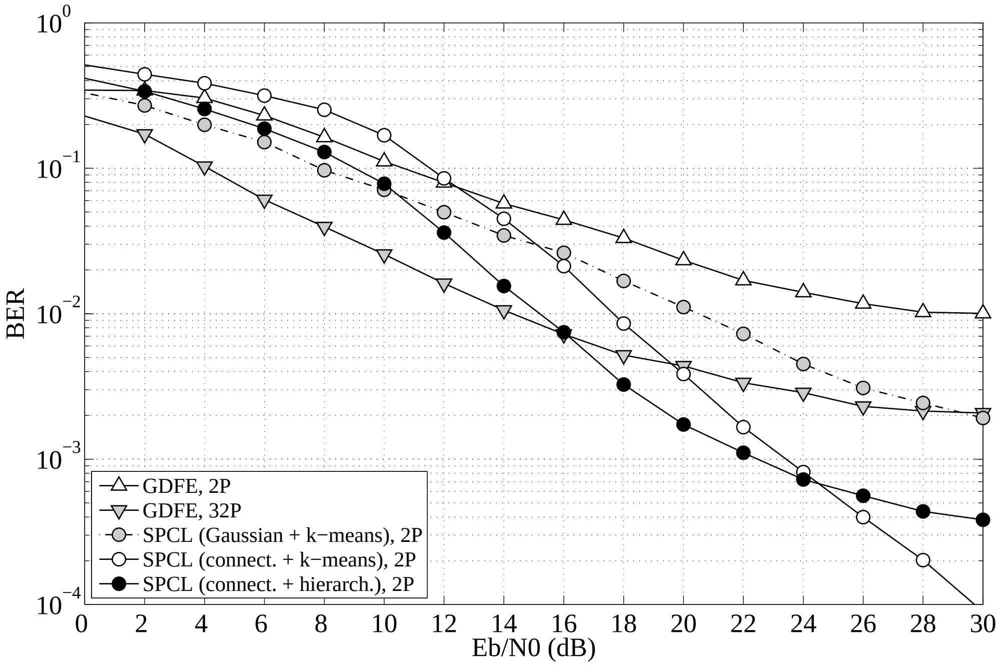


Figure 8.9: Performance comparison of different algorithms for a 2 × 2 QPSK system with $f_d T = 0.001$.

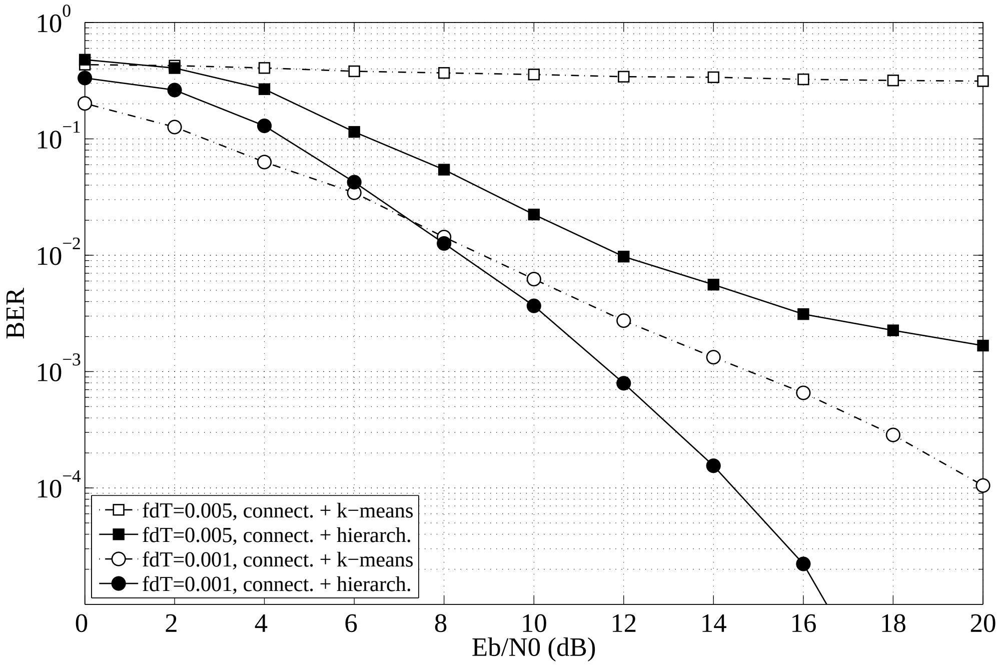


Figure 8.10: Comparison of the proposed algorithm with the connectivity kernel and two types of eigenvector clustering, at different Doppler spreads, for a 2 × 4 QPSK system.

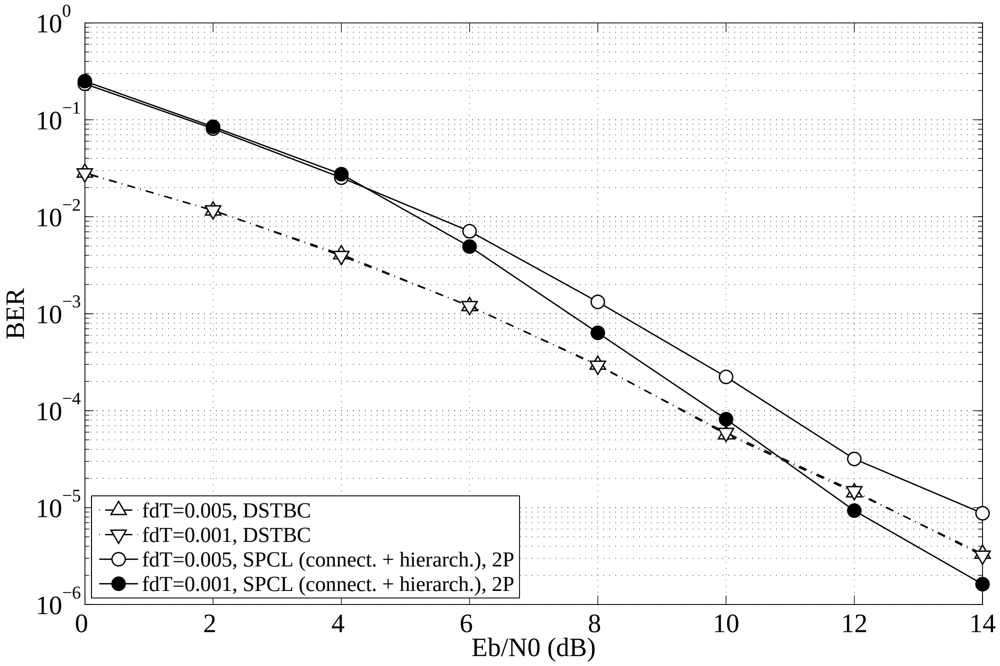


Figure 8.11: BER curves for a 2 × 2 BPSK MIMO system with Alamouti coding.

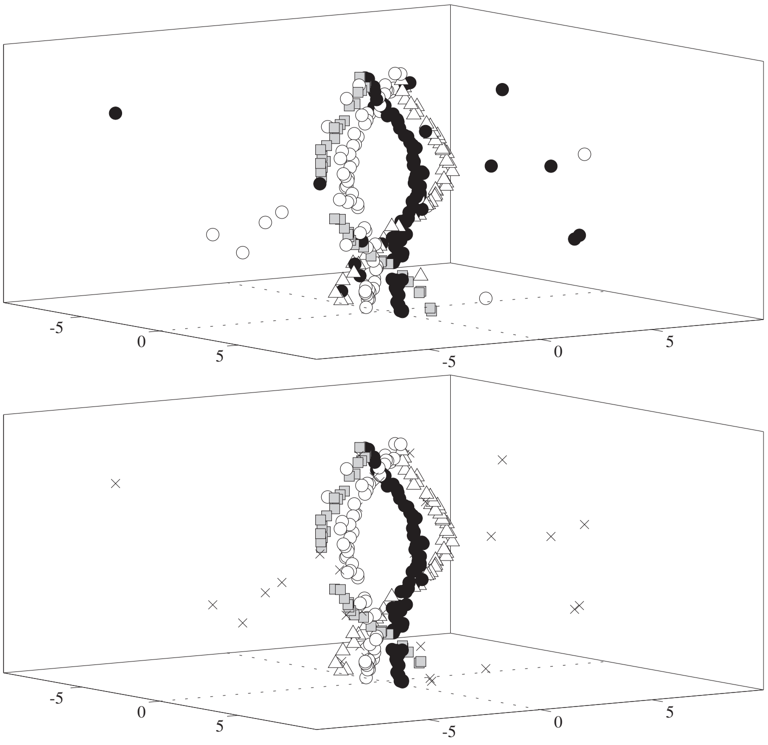


Figure 8.12: Clustering with impulsive noise. Top: retrieving 4 clusters. Bottom: retrieving 5 clusters.

8.9 Conclusions

We presented a novel clustering algorithm that is capable of decoding fast flat-fading time-varying M-PSK MIMO channels. This algorithm operates on the received data to which it adds a temporal dimension, and it exploits constellation geometry to retrieve symbol threads from it. The only supervised part of the presented clustering method is the final decoding phase.

The presented clustering algorithm is a spectral clustering method whose different steps were analyzed and adjusted to suit this particular problem. Specifically, it was shown that the connectivity kernel is more appropriate than the Gaussian kernel for this type of data. A number of extensions were also presented, including a hierarchical clustering procedure to avoid mistakes made by k-means in the final clustering step, and a method to exploit the additional structure introduced by the Alamouti code. Although the algorithm can be used for different M-PSK systems, in practice it is limited to moderate numbers of transmit antennas and BPSK and QPSK constellations. Simulation results show that the presented algorithm outperforms the adaptive GDFE method for high Doppler spreads using less pilot symbols.

The publications that have contributed to this chapter are listed below. The first publication was awarded a "Best Student Paper Award" at the 15th European Signal Processing Conference conference.

- S. Van Vaerenbergh, E. Estébanez, and I. Santamaría. "A spectral clustering algorithm for decoding fast time-varying BPSK MIMO channels". In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*. Poznań, Poland, 2007.

- S. Van Vaerenbergh and I. Santamaría. *Intelligent Systems: Techniques and Applications*, chapter A Spectral Clustering Approach for Blind Decoding of MIMO Transmissions over Time-Correlated Fading Channels, pages 351–377. Shaker Publishing, Maastricht, The Netherlands, 2008.

- S. Van Vaerenbergh, I. Santamaría, P. Barbano, U. Ozertem, and D. Erdogmus. "Path-based spectral clustering for decoding fast time-varying MIMO channels". In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2009)*. Grenoble, France, 2009.

- S. Van Vaerenbergh, I. Santamaría, P. E. Barbano, U. Ozertem, and D. Erdogmus. "Path-based clustering for decoding time-varying MIMO channels". *IEEE Transactions on Signal Processing, in preparation*, 2010.

Chapter 9

Underdetermined Post-Nonlinear Blind Source Separation

This chapter deals with a blind source separation scenario in which the number of mixtures is less than the number of sources. In this case, the sources cannot be recovered by standard blind source separation techniques. Nevertheless, there exist a number of clustering procedures to estimate the mixture matrix, by relying on the sparseness of the source signals.

We focus on the post-nonlinear case, in which the linear mixtures undergo an additional nonlinear transformation. Due to this nonlinearity, standard linear algorithms are not capable of dealing successfully with this problem. We will design a spectral clustering based technique which is able to estimate the inverse nonlinear transformations. Once estimated, the problem is reduced to a linear source separation scenario which can be solved by one of the existing linear techniques.

9.1 Introduction to Blind Source Separation

Blind source separation (BSS) is an important problem in the signal processing area, with a number of applications in communications, speech processing and biomedical signal processing. In general, the goal of BSS is to recover a number of source signals from their observed linear or nonlinear mixtures [Comon et al., 1991, Cardoso, 1998]. Depending on the type of the mixing process, the number of sources, n_s, and the number of available mixtures, m, several scenarios can be distinguished.

9.1.1 Standard problem setting

The most basic scenario of BSS assumes a simple linear model, in which the measurement random vector $\mathbf{y} \in \mathbb{R}^m$ can be described as an instantaneous mixture

$$\mathbf{y} = \mathbf{A}\mathbf{s} + \mathbf{v}, \tag{9.1}$$

where $\mathbf{s} \in \mathbb{R}^{n_s}$ is a zero-mean independent random vector representing statistically independent source, $\mathbf{A} \in \mathbb{R}^{m \times n_s}$ is the unknown mixing matrix, and $\mathbf{v} \in \mathbb{R}^m$ is an independent random vector representing sensor noise.

The BSS problems that are based on this model can be categorized depending on the number of mixtures, m, versus the number of sources, n_s. Many techniques have been developed for the case where as many mixtures as unknown sources are available ($m = n_s$). Most of them stem from the theory of independent component analysis (ICA) [Comon, 1994, Hyvärinen et al., 2001], a statistical technique whose goal is to represent a set of random variables as linear functions of statistically independent components. In the absence of noise, this reduces to the problem of estimating the "unmixing matrix" $\mathbf{W} = \mathbf{A}^{-1}$, which allows to retrieve the estimated sources as $\hat{\mathbf{x}} = \mathbf{W}\mathbf{y} = \mathbf{W}\mathbf{A}\mathbf{s}$. The unmixing matrix $\mathbf{W}$ is found by minimizing the dependence between the elements of the transformed vector $\hat{\mathbf{x}}$. In order to estimate this matrix, a suitable dependence measure or contrast function must be defined and minimized with respect to $\mathbf{W}$.

Most ICA contrast functions can be derived using the maximum likelihood (ML) principle. By denoting the pdf of each source s_i as p_{s_i}, the pdf of the random source vector $\mathbf{s} = [s_1, \ldots, s_{n_s}]^T$ is given by $P_s = \prod_{i=1}^{n_s} p_{s_i}(s_i)$ and the pdf of the observations for a given mixing matrix $\mathbf{A}$ is

$$P(\mathbf{x}|\mathbf{A}, P_s) = |\mathbf{A}|^{-1} P_s(\mathbf{A}^{-1}\mathbf{x}), \tag{9.2}$$

where $\mathbf{A}^{-1}$ is the unmixing matrix that needs to be estimated. Given a set of N realizations of $\mathbf{x}$, their normalized log-likehood is given by

$$L_N(\mathbf{A}, P_s) = \frac{1}{N} \sum_{n=1}^{N} \log P(\mathbf{x}[n]|\mathbf{A}, P_s), \tag{9.3}$$

and the ML estimate of the mixing matrix is $\mathbf{A} = \arg\max_{\mathbf{A}}(L_N(\mathbf{A}, P_s))$. For this problem, an impressive amount of algorithms have been proposed in the literature. We will shortly review some of the most interesting techniques[1].

Infomax

The Infomax algorithm [Bell and Sejnowski, 1995] is based on the principle that network entropy maximization, or "infomax", is equivalent to maximum likelihood estimation. Infomax maximizes the log-likehood function given in Eq. (9.3) with respect to $\mathbf{W} = \mathbf{A}^{-1}$, using a stochastic gradient descent. The original Infomax algorithm is rather restricted, since it assumes the sources have sub-Gaussian distributions. Furthermore, its convergence is slow due to the use of gradient techniques. Its main advantage is its simplicity of implementation and very low computational complexity.

JADE

The Joint Approximate Diagonalization of Eigenmatrices (JADE) algorithm [Cardoso and Souloumiac, 1993] is based on higher-order statistics. In particular,

[1] Notice that these methods only exploit the spatial structure of the mixtures, which is a suitable approach for mixtures of i.i.d. sources. If the sources have temporal structure, this information can also be exploited (see for instance [Molgedey and Schuster, 1994]).

it aims to obtain a cumulant matrix with equal eigenvalues. Cumulant matrices are four-dimensional extensions of the covariance matrix, using the fourth order cumulant $\text{cum}(x_i, x_j, x_k, x_l) = E\{x_i x_j x_k x_l\} - E\{x_i x_j\}E\{x_k x_l\} - E\{x_i x_k\}E\{x_j x_l\} - E\{x_i x_l\}E\{x_j x_k\}$. The contrast function of JADE consists of a sum of squared cross-cumulants of the estimated sources, which it minimizes by some efficient algebraic techniques. Although JADE is very competitive for small-scale problems, its use of cumulant tensors makes it unsuitable for large-scale applications.

FastICA

The FastICA algorithm [Hyvärinen and Oja, 1997] uses a fast "fixed-point" iterative scheme to maximize the non-Gaussianity of the estimated sources. Implementations were proposed that measure the non-Gaussianity either by approximating the kurtosis of the estimated sources or by their negentropy, although the latter is more robust. In order to ensure orthonormality of the optimal unmixing matrix, FastICA first decorrelates ("whitens") the mixtures by a PCA-based transformation. The FastICA algorithm is computationally very efficient, has low-memory requirements, and it is fairly unrestrictive with respect to the source distributions. Its main disadvantage is that its performance depends on the choice of a suitable nonlinearity to calculate the higher-order cumulants.

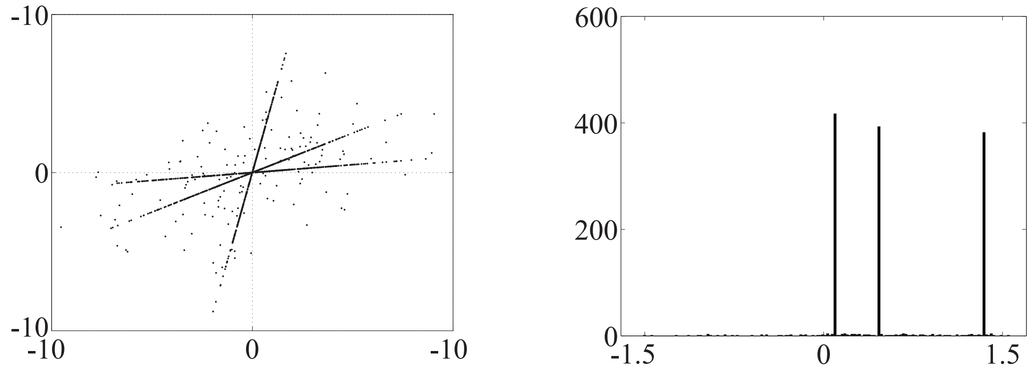
Kernel ICA

Kernel ICA [Bach and Jordan, 2002] uses a kernel CCA based contrast function to obtain the unmixing matrix $\mathbf{W}$. For "rich enough" kernels such as the Gaussian kernel, it was shown that the components of the random vector $\mathbf{x}$ are independent if and only if their first canonical correlation in the corresponding RKHS is equal to zero. Kernel ICA minimizes a KCCA-based contrast function by means of a gradient approach, making use of the fact that after whitening the mixtures, the optimal unmixing matrix $\mathbf{W}$ is orthonormal. Apart from outperforming the previous three algorithms, Kernel ICA is more robust to outliers and near-Gaussianity of the sources. However, these performance improvements come at a higher computational cost.

9.1.2 Underdetermined BSS

The previous methods dealt with the standard BSS problem where the number of mixtures and sources are the same. If more mixtures than sources are available ($m > n_s$), the redundancy in information can be used to extend existing techniques in order to achieve additional noise reduction [Joho et al., 2000].

If less mixtures than sources are available ($m < n_s$), a more complex scenario arises. This so-called *underdetermined* BSS problem can only be solved if one relies on *a priori* information about the sources. Most techniques that tackle this problem are designed as two-step procedures: In a first step the mixing matrix is estimated, often based on geometric properties. In a second step the original sources are recovered.



(a) Scatter plot. (b) Angle histogram.

Figure 9.1: Scatter plot and angle histogram of a linear mixture with 3 sparse sources, with $p = 0.9$, and 2 mixtures. In the scatter plot (a), most samples are aligned with a column of **A** due to the sparseness of the sources. This is reflected in the angle histogram (b) that shows three dominant angles.

An extreme case of underdetermined BSS problems occurs when only one mixture is available. In audio applications, this scenario is known as the single-microphone source separation problem [Roweis, 2000, Bach and Jordan, 2004]. In the present application, however, at least two mixtures will be available.

To estimate the mixing matrix, it is often assumed that the problem can be transformed to a domain in which the sources are *sparse*, meaning that they equal zero most of the time. For instance, in problems of audio separation it is usual to work in the time-frequency domain, where overlap of the sources is likely to be much smaller than in the time domain [Belouchrani and Amin, 1998]. In order to model sources with different degrees of sparsity we consider the probability density function

$$p_{s_i}(s_i) = p_i \delta(s_i) + (1 - p_i) f_{s_i}(s_i), \quad i = 1, \dots, n; \tag{9.4}$$

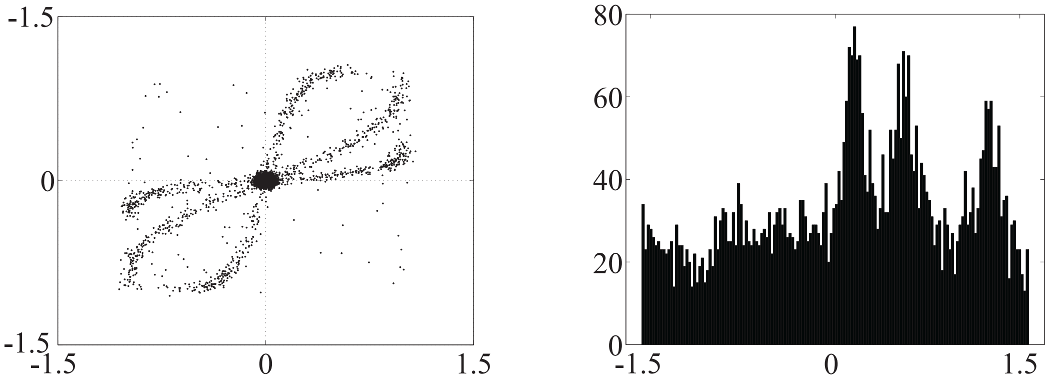
where p_i is the probability that a source s_i is inactive, $\delta(\cdot)$ denotes Dirac's delta and $f_{s_i}(s_i)$ is the pdf of the i-th source when it is active [Luengo et al., 2005]. If p_i is high, the corresponding source will be inactive most of the time.

In the following we will assume that the sources in the BSS problem are sparse, or that the problem has been transformed into a space where this is the case. When only the i-th source is active, the output vector can be written as

$$\mathbf{y} = \mathbf{A}_i s_i + \mathbf{v}, \tag{9.5}$$

where $\mathbf{A}_i$ is the i-th column of the mixing matrix and s_i is the active source. Thus, in the absence of noise this output signal will be aligned with the i-th column of **A**. In mixtures with sparse sources, only one or few sources will be active most of the time. Therefore, a large number of the output samples **y** will be aligned with one of the directions represented by each column $\mathbf{A}_i$, as can be seen in Fig. 9.1.

Using this geometrical insight, several algorithms were proposed to separate sparse sources in an underdetermined problem setting [Bofill and Zibulevsky, 2001,



(a) Scatter plot.

(b) Angle histogram.

Figure 9.2: (a) shows the scatter plot of the mixtures of Fig. 9.1 after each mixture is transformed nonlinearly and noise is added as in Eq. (9.6). The dominant directions of the scatter plot are now nonlinear curves. The histogram (b) still shows three, though less precise peaks, which only provide a linear estimation of the dominant directions.

Vielva et al., 2001, Lee et al., 1999b, Luengo et al., 2005]. In a first stage, these methods estimate the mixing matrix **A** by identifying the main vectors to which the samples in the scatter plot are aligned. A large number of estimators have been proposed for this purpose, among them a Parzen windowing-based method [Erdogmus et al., 2001b], a line spectrum estimation method [Vielva et al., 2002] and a kernel PCA based technique [Desobry and Févotte, 2006]. Once **A** is known, the original source samples are estimated. The most straightforward method to do this is by using the pseudoinverse of **A**. Better estimates can be obtained by the shortest-path (L_1-norm) method introduced in [Bofill and Zibulevsky, 2001] or the MAP estimator described in [Vielva et al., 2001].

9.1.3 Post-nonlinear BSS

A considerable amount of research has also been done on the so-called post-nonlinear BSS problem (PNL BSS), in which the sources are first mixed linearly and then transformed nonlinearly. In this scenario, the m sensors that measure the mixtures show some kind of saturation effect or another nonlinearity, which results in the extension of (9.1) to a post-nonlinear mixture model

$$\mathbf{x} = \mathbf{f}\left(\mathbf{A}\mathbf{s}\right) + \mathbf{v}, \tag{9.6}$$

where $\mathbf{f}(\cdot)$ is an $m \times 1$ vector of nonlinear functions and $\mathbf{x} \in \mathbb{R}^m$ is the measurement random vector.

In this model, each component $f_i(\cdot)$ of $\mathbf{f}(\cdot)$ affects only one mixture so that there are no cross-nonlinearities. In the underdetermined case ($m < n$), the addition of the post-nonlinear transformations means an additional difficulty for the estimation

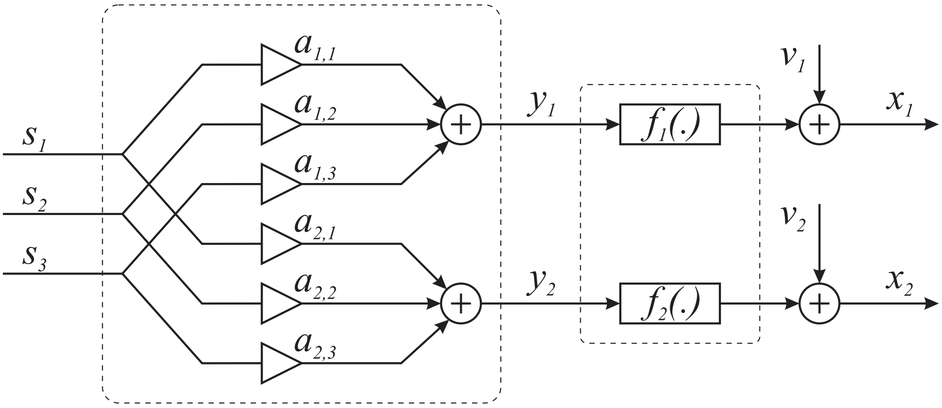


Figure 9.3: The post-nonlinear mixing model for $m = 2$ mixtures and $n = 3$ sources. The linear mixture model is represented by the large dashed block, in which the sources are multiplied by mixing factors $a_{i,j}$ and then summed. The smaller dashed block represents the nonlinearities that operate on each mixture.

of the mixing matrix, as can be seen in Fig. 9.2: the scatter plot now shows curves instead of straight lines. Therefore, it is not sufficient to apply a clustering technique on the angle histogram. The corresponding mixing diagram is shown in Fig. 9.3.

For an equal number of mixtures and sources ($m = n$), some algorithms have been proposed [Solazzi et al., 2001, Tan and Wang, 2001, Babaie-Zadeh et al., 2002, Taleb and Jutten, 1999] that treat the PNL problem. However, these algorithms cannot deal with the more restricted problem of *underdetermined* PNL BSS. An *underdetermined* PNL BSS algorithm was recently proposed in [Theis and Amari, 2004]. It nevertheless requires the number of active sources at each instant to be lower than the total number of available mixtures m and assumes noiseless mixtures. The approach presented in this chapter relaxes these restrictions on the sources and it is able to work with noisy mixtures.

9.1.4 Proposed approach

The described algorithm aims at estimating the inverse nonlinearities $\mathbf{g}(\cdot) = \mathbf{f}^{-1}(\cdot)$ of the underdetermined post-nonlinear blind source separation problem. Estimation of $\mathbf{g}(\cdot)$ allows to recover the linear mixtures $\mathbf{y}$, which can be used to estimate the original sources $\mathbf{s}$ relying on known methods for *linear* underdetermined BSS.

The algorithm consists of two main steps, as proposed in [Van Vaerenbergh and Santamaría, 2006]. After some de-noising in a preprocessing stage, we apply a spectral clustering technique that clusters the mixture samples into different sets corresponding to the different sources. Next, the inverse nonlinearities are estimated by training a set of multilayer perceptrons (MLP) with the clustered data by minimizing a specifically designed cost function. Finally, each mixture is transformed by its corresponding inverse nonlinearity. The resulting linear underdetermined BSS problem can be solved using any of the existing methods. The different steps of this method are described in detail in the next sections.

A related approach for a much simpler BSS scenario was proposed simultaneously in [Babaie-Zadeh et al., 2006]. It treated the basic linear BSS problem with $m = n_s$ for sparse sources. In a first step, points were clustered into groups by a k-lines procedure, and in a second step they were fitted to lines representing the columns of the mixing matrix.

9.2 Clustering PNL BSS Data

In this section we show how spectral clustering can be applied to the problem of underdetermined post-nonlinear BSS. In particular, after identifying samples that correspond to time instants on which only one source is active, spectral clustering can be used to cluster them into groups corresponding to the different sources.

9.2.1 Preprocessing

Some preprocessing steps are taken to facilitate the spectral clustering stage. Specifically, the mixture samples are roughly reduced to those for which only one source was active at that instant. Apart from guaranteeing the overall efficiency of the algorithm, this reduction also lowers the computational cost.

Central samples are removed because they correspond to inactive sources and are almost unaffected by the nonlinearity. If $p_i = p, \forall i$, the probability of having no active sources at all according to the sparse source model (9.4) is p^n, so the $\nu_1 = p^n N$ samples closest to the origin can be removed. In addition, "non-sparse" samples, which are the result of multiple sources active at the same time, are also removed. When applying local scaling (see section 7.2.3), points with a higher local scale will likely correspond to multiple active sources, since the local scale is inversely proportional to the local density of points. Therefore, the samples that correspond time instants at which multiple sources are active can be estimated as the $\nu_2 = \left[1 - n(1-p)p^{n-1} - p^n\right] N$ samples with highest local scale. This can also be seen as a de-noising operation.

If the sources have different p_i-values, ν_1 and ν_2 can easily be calculated according to the previous description. In practice rough (over-) estimates can be used for ν_1 and ν_2. Especially when the p_i are unknown, ν_1 and ν_2 should be chosen so that after preprocessing the remaining samples can be clustered into non-overlapping clusters.

9.2.2 Identification and clustering limitations

The performance of the clustering algorithm will depend on the distance between points of different clusters, since the spectral clustering algorithm based on a Gaussian kernel is not capable of retrieving overlapping clusters correctly.

Furthermore, it is assumed that the different sources have double sided distributions. By applying spectral clustering, it is then possible to distinguish $2n_s$ clusters, one for each sidelobe of the n_s distributions. Since the nonlinearities are assumed to be linear for small input values, determining which pair of clusters correspond to the same source can be done by looking at which clusters have the same slopes close to the origin. Finally, n_S clusters are obtained, corresponding to the n_s sources. We will denote these clusters as $\mathcal{C}_i$, and assume they contain time indices n. A point $\mathbf{x}[n]$ is said to belong to a cluster $\mathcal{C}_i$ if $n \in \mathcal{C}_i$.

9.3 Estimating the Inverse Nonlinear Functions

The clustering process retrieves n_s curve-shaped clusters, where each cluster correspond to one active source. In this section we will discuss two strategies to determine the m inverse nonlinearities that should be applied on the m components of these curves to transform them into the straight lines that appear in the linear underdetermined BSS problem.

In [Theis and Amari, 2004], these nonlinearities were estimated by considering that, after transforming the cluster data with the estimated inverse nonlinearities, there should be a linear relationship between the *same component* of the points in *different clusters*, which can be calculated by resampling all obtained curves. In the approach proposed in [Van Vaerenbergh and Santamaría, 2006], we exploited the fact that there should be a linear relationship between the *different components* of the points in *same cluster*. Both approaches can lead to a reliable estimation of the inverse nonlinearities, but while the former requires resampling of each clustered curve to identify the corresponding components, the latter allows to operate directly on the available data. By doing so it avoids the resampling process needed in [Theis and Amari, 2004] that can be problematic in cases of strong nonlinearities or in the presence of noise.

9.3.1 Cost function

According to the proposed criterion, there should be a fixed linear relationship between any two components j, k of every point $\mathbf{x}[n]$ in a cluster $\mathcal{C}_i$. We will represent this linear relationship by a slope factor $\rho^i_{j,k}$. Therefore, in [Van Vaerenbergh and Santamaría, 2006] the following cost function was proposed

$$\min_{g,\rho} J = \sum_{i=1}^{n_s} \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} \sum_{n_i \in \mathcal{C}_i} \Big(g_j(x_j[n_i]) - \rho^i_{j,k} g_k(x_k[n_i]) \Big)^2 . \tag{9.7}$$

Here, $x_j[n]$ represents the j-th component of data point $\mathbf{x}[n]$, and $g_j(\cdot)$ is the nonlinear function that applies to the j-th component. Thanks to the introduction of the slope factors $\rho^i_{j,k}$, the cost will be zero when the first term, $g_j\left(x_j[n_i]\right)$, is mapped exactly onto the second term, $\rho^i_{j,k} g_k\left(x^i_k[n_i]\right)$, i.e., when there exists a linear relationship with slope $\rho^i_{j,k}$ between the components j and k of cluster $\mathcal{C}_i$. In order to avoid the zero-solution, the solutions were restricted by $\sum_{i=1}^{n_s} \sum_{j=1}^{m} \sum_{n_i \in \mathcal{C}_i} \|g_j(x_j[n_i])\|^2 = 1$.

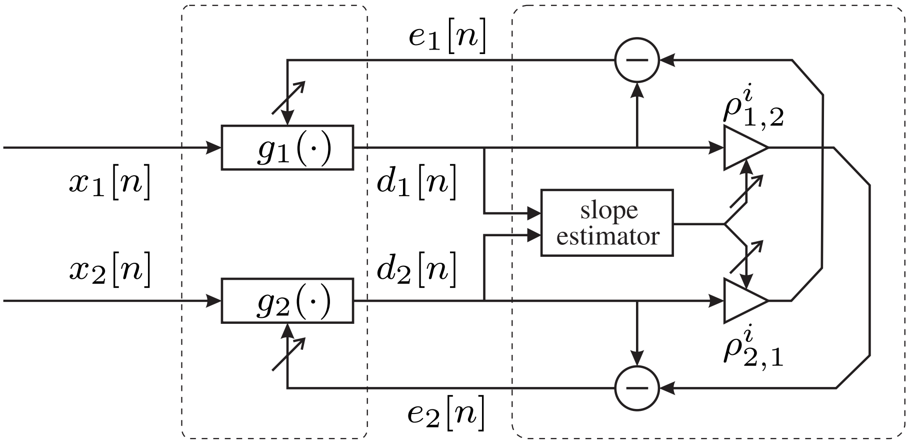


Figure 9.4: The block diagram used for the MLP parameter training for $m = 2$. The blocks labeled $g_1(.)$ and $g_2(.)$ represent the 2 MLPs. The slope estimator is used to estimate the slope $K^i_{1,2}$ of the curve formed by $(d^i_1(t), d^i_2(t))$. To train the upper MLP, we use as desired signal $K^i_{1,2}d^i_2(t)$. In this way the error signal $e^i_1(t) = d^i_1(t) - K^i_{1,2}d^i_2(t)$ measures the deviation from linearity of this curve. The same procedure is carried out for the lower MLP.

Note that the definition of the cost function uses only the factors ρ^i_{jk} for which $j < k$. In order to obtain a more symmetric cost function, one could fix the missing factors as $\rho^i_{jk} = 1$ for $j > k$. Also notice that terms with $j = k$ are omitted, since they do not contribute to the cost function.

Since the cost function (9.7) has two parameter sets, i.e., the nonlinearities $g_j(\cdot)$ and the slope factors $\rho^i_{j,k}$, we propose to minimize this cost function in an iterative manner, updating alternatingly each of the sets.

9.3.2 Estimation using an MLP-based model

In [Van Vaerenbergh and Santamaría, 2006] the inverse nonlinearities $g_j(.)$ are modeled as single-input single-output (SISO) multilayer perceptrons (MLPs), with one hidden layer of r neurons. For an input x, the output of the j-th MLP is obtained as

$$\begin{aligned} g_j(x) &= \sum_{k=1}^{r} w_{j,2}(k)\Phi\left(\sum_{l=1}^{r} w_{j,1}(l)x + b_{j,1}(l)\right) + b_{j,2} \\ &= \mathbf{w}^T_{j,2}\Phi(\mathbf{w}_{j,1}x + \mathbf{b}_{j,1}) + b_{j,2}, \end{aligned} \tag{9.8}$$

where $\mathbf{w}_{j,1}, \mathbf{w}_{j,2} \in \mathbb{R}^r$ are weight vectors, $\mathbf{b}_{j,1} \in \mathbb{R}^r$ and $b_{j,2} \in \mathbb{R}$ are biases and $\Phi(.)$ is a neuron activation function.

Parameter training

Fig. 9.4 illustrates the proposed training diagram corresponding to the cost function (9.7) for the case $m = 2$. For $m > 2$ a similar diagram is obtained.

A simple initialization procedure consists in setting the parameters of the MLPs such that they represent the identity function,

$$\mathbf{g}_j(x) = x, \quad \forall j. \tag{9.9}$$

By assuming that the nonlinearities are smooth and invertible, this initialization should allow to find the optimal solutions.

In each iteration of the actual training, the parameters of the MLP are adapted using a batch gradient approach to minimize (9.7). Similar to [Theis and Amari, 2004], we also assume that the nonlinearities pass through the origin, i.e., $g_j(0) = 0$. Therefore the bias of the output layer $b_{j,2}$ is fixed as $b_{j,2} = -\mathbf{w}_{j,2}^T \Phi\left(\mathbf{b}_{j,1}\right)$. In order to update the slope factors ρ_{jk}^i, the histogram-based estimator described in [Luengo et al., 2005] can be used. Ideally, there should be a linear combination between components j and k of cluster i, resulting in

$$\rho_{j,k}^i = \frac{x_j[n_i]}{x_k[n_i]}, \quad \forall n_i \in \mathcal{C}_i. \tag{9.10}$$

After this training the mixing matrix can be estimated in a straightforward way relying on the estimated $\rho_{j,k}^i$.

In the future research lines of this thesis, we also propose a KCCA-based technique that can replace the MLP-based nonlinearity estimation (see section 10.5.1).

9.4 Simulation Results

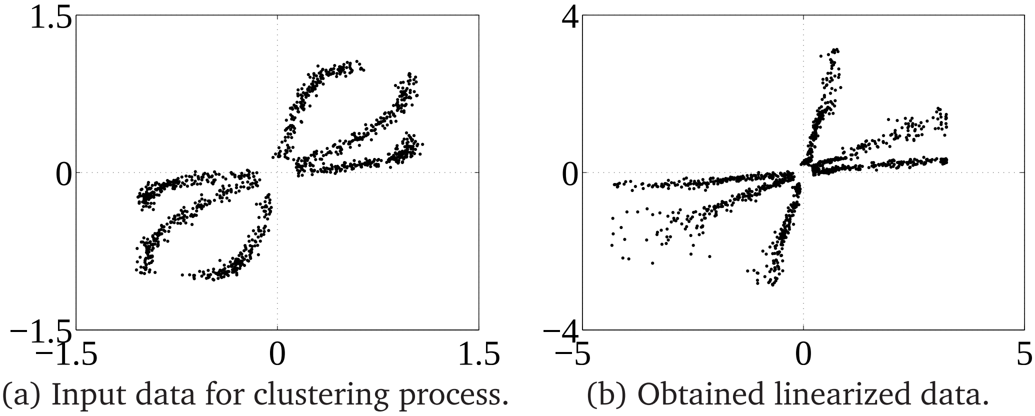
To illustrate the presented method we performed various experiments. In the first experiment, we used the mixing model with 3 sources and 2 mixtures from Fig. 9.3. The source distribution $f_{s_1}(s_i)$ was chosen as $\mathrm{N}(0,1)$ and the sparsity factor was $p = 0.9$. 5000 samples of these sources were mixed by

$$A = \begin{bmatrix} 0.2 & 0.8 & 1.0 \\ 0.9 & 0.4 & 0.1 \end{bmatrix}. \tag{9.11}$$

The obtained data points were transformed by the post-nonlinear functions $f_1(x) = \tanh(0.5x)$ and $f_2(x) = \tanh(0.5x)$ and 20dB of Gaussian white noise was added to the mixtures. The obtained measurements are the ones displayed in Fig. 9.2.

Self-tuning spectral clustering was applied to these data, with $K = 10$ neighbors[2] and next two MLPs with 9 hidden neurons were trained to estimate the two inverse

[2] Since the goal of the clustering stage is to retrieve elongated clusters, one could argue that the connectivity kernel from chapter 8 could be used. However, as the clusters in the present application are much more dense, the advantage of the connectivity kernel is almost negligible compared to the Gaussian kernel, in most cases.



(a) Input data for clustering process. (b) Obtained linearized data.

Figure 9.5: Scatter plots before and after the nonlinearity estimation. (a) shows the scatter plot after removing central samples and non-sparse samples. Six clusters can be distinguished, and they are combined to three clusters. (b) shows the output of the MLPs after training with this data. The estimated inverse nonlinearities (e) and (f) coincide well with the real inverse nonlinearities (c) and (d) except for scaling.

nonlinearities, with a learning rate of $\mu = 0.01$ and a maximum of 1000 epochs. The activation function of each neuron was the hyperbolic tangent, and the coefficients $w_{j,1}$ and $w_{j,2}$ were kept strictly positive in order to guarantee invertibility of the obtained nonlinearity. In Fig. 9.5 the training process is illustrated on the scatter plot. The estimated inverse nonlinearities, shown in Fig. 9.6, coincide well with the real inverse nonlinearities, up to a scaling factor which is inherent to this problem.

Next, the slope factors $\rho^i_{j,k}$ were used to estimate the mixing matrix

$$\hat{\mathbf{A}} = \begin{bmatrix} 0.99 & 0.23 & 0.88 \\ 0.12 & 0.97 & 0.44 \end{bmatrix}. \tag{9.12}$$

Up to scaling and a permutation inherent to all BSS problems, it represents an acceptable estimate of the original **A**. Finally, the source signals **s** were recovered using the underdetermined linear BSS algorithm proposed in [Bofill and Zibulevsky, 2001]. In this experiment, the evaluation criterion of the separation is the signal-to-noise (SNR) ratio[3], which compares the quality of the recovered signal with artifacts in the estimate due to other sources and additive noise. Specifically,

$$SNR = 10 \log_{10} \frac{||\hat{\mathbf{s}}||^2}{||\mathbf{s} - \hat{\mathbf{s}}||^2}, \tag{9.13}$$

in which the estimate $\hat{\mathbf{s}}$ and the source signal **s** were normalized for fair comparison. The obtained SNR values ratios amount 15dB, 11dB and 12dB respectively, compared with 6dB, 4dB and 4dB when this linear BSS algorithm was applied directly on the nonlinear mixtures. Although we achieved an improvement over the linear case, the obtained error is still rather high.

[3] In this experiment, both the noise level and the separation evaluation criterion are expressed in SNR. However, from the context it should be clear what value is referred to.

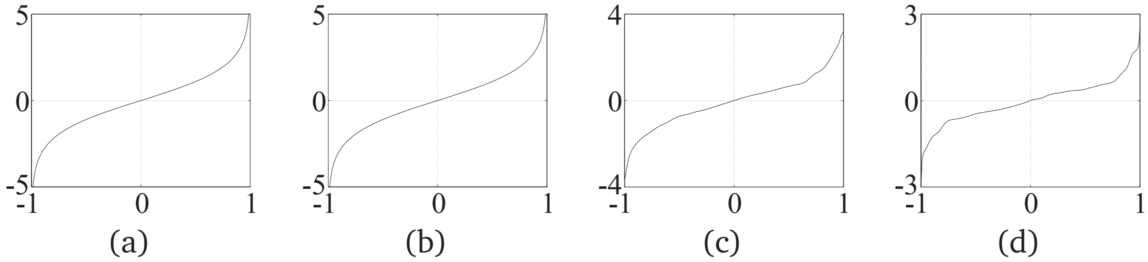
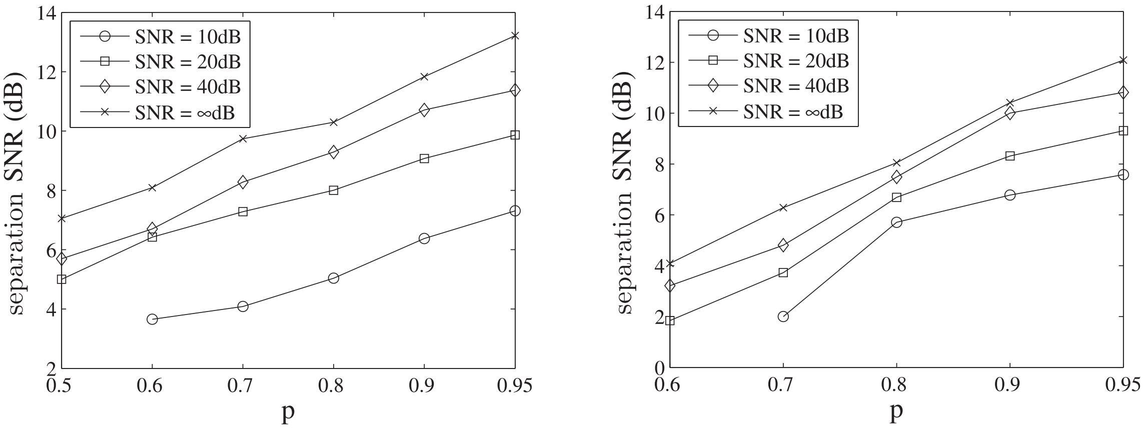


Figure 9.6: The estimated inverse nonlinearities (c) and (d) coincide well with the real inverse nonlinearities (a) and (b), up to a scaling factor inherent to BSS problems.

In a second experiment we conducted a number of Monte-Carlo simulations for signals with different sparsity and noise levels. The source signals were generated according to model (9.6) with a normal distribution $f_{s_1}(s_i)$ of zero mean and variance 10. For each sparsity and SNR level, 20 different mixing matrices were generated randomly by choosing the amplitudes of the basis vectors uniformly from $[0.1, 1]$ and the angles uniformly from $[-\pi, \pi]$ with a minimum angle of $\pi/10$ between every pair of basis vectors to avoid cluster overlapping. The number of samples in each case was $2500/(1 - \nu_1 - \nu_2)$ in order to restrict the clustering to 2500 samples. After mixing by **A**, the mixtures were transformed by the nonlinear functions $f_j(x) = \tanh(x)$. Finally Gaussian white noise was added to reach the specified SNR level.

A 2-measurement scenario with 3 mixtures ($m = 3$, $n_s = 2$) as well as a 3-measurement scenario with 5 mixtures ($m = 5$, $n_s = 3$) were simulated. Self-tuning spectral clustering was applied and m MLPs with $r = 15$ hidden neurons were trained to estimate the two inverse nonlinearities, with a learning rate of $\mu = 0.01$ and a maximum of 1000 epochs. For all the neurons in the hidden layers the hyperbolic tangent activation function was chosen.

After training, the basis vectors were estimated from $\rho^i_{j,k}$ and the source signals were estimated applying the shortest-path algorithm from [Bofill and Zibulevsky, 2001]. The results are shown in Fig. 9.7. Since no measures were taken to reduce the sensor noise, the obtained SNR values are highly dependent on the noise level. Although in most cases the inverse nonlinearity estimation can "linearize" the clusters sufficiently well (see for instance Fig. 9.5(b)), only a modest SNR value was obtained even for the noiseless case ($SNR = \infty$dB). This is due to the strong nonlinearity and to the fact that the MLPs only represent the inverse nonlinear functions well for input points that are in the training range. Points that are outside of it, such as the "non-sparse" samples, are estimated with greater error and therefore represent the main contribution in the error. Notice also that the SNR values obtained for the setting of the previous experiment (20dB noise and $p = 0.9$) are slightly higher here, which is due to the fact that the results are averaged out here over a number of random realizations of the mixing process.



(a) 3 sources, 2 mixtures

(b) 5 sources, 3 mixtures

Figure 9.7: Obtained SNR values for two different mixture problems, for different sparsity and additive noise levels.

9.5 Conclusions

We presented an algorithm based on spectral clustering to solve the post-nonlinear underdetermined blind source problem. The algorithm consists of two steps: First, a spectral clustering algorithm is applied to identify the active sources. Second, the inverse nonlinearities are estimated by a set of coupled MLPs. After these two steps, the obtained signals represent a "linearized" underdetermined BSS problem, which can be solved by traditional linear methods.

The presented method requires that the sources are sparse and the nonlinearities are invertible. As long as the clusters corresponding to the different sources do not overlap in the scatter plot of the mixtures, there is no restriction on the number of sources or mixtures. We included simulation results for the case of 3 sources and 2 mixtures, and for the case of 5 sources and 3 mixtures. The proposed method is used as a preprocessing step that linearizes the problem, after which a traditional linear method is applied. If the proposed method is omitted, the unmixing performance degrades significantly.

The publication that has contributed to this chapter is

- S. Van Vaerenbergh and I. Santamaría. "A spectral clustering approach to underdetermined postnonlinear blind source separation of sparse sources". *IEEE Transactions on Neural Networks*, volume 17, no. 3, pages 811–814, 2006.

Part IV

Conclusions and Bibliography

Chapter 10
Conclusions and Future Directions

Thanks to the advances during the last decade, the field of kernel methods has become an attractive framework for treating a wide range of problems in nonlinear signal processing. As machine learning techniques, kernel methods are universal approximators by nature, and their solid mathematical foundation guarantees that the resulting techniques are convex optimization problems. Additionally, a low computational complexity can be guaranteed by designing online kernel methods or by applying low-rank approximations to block-based techniques.

In this thesis we have studied the application of kernel methods to a number of problems in nonlinear signal processing and communications, in particular identification and equalization of nonlinear systems and nonlinear blind source separation. The proposed techniques are related through their foundation in kernel regression, kernel principal component analysis and kernel canonical correlation analysis. Furthermore, a lot of attention has gone to developing efficient online techniques.

The presented algorithms demonstrate that kernel methods are capable of being more accurate, more noise-robust, and algorithmically more interesting compared with traditional methods in nonlinear signal processing. These advantages can be attributed mostly to the used RKHS framework and to the flexibility of having a kernel, which allows to engineer kernel methods to fit specific problems.

The main contributions to the state of the art presented in this thesis are

- the introduction of a family of kernel recursive least-squares algorithms that use a fixed memory size, with application in nonlinear adaptive filtering;
- a kernel canonical correlation based framework for supervised identification and equalization of Wiener and Hammerstein systems;
- a blind equalization technique for Wiener systems based on oversampling;
- a spectral clustering based algorithm for blind decoding of fast time-varying MIMO systems.

In the following we summarize the main conclusions of each chapter. In addition, we list some guidelines for future work based on the methods proposed in this thesis.

10.1 Sliding-Window and Fixed-Budget KRLS

The sliding-window kernel RLS and fixed-budget kernel RLS algorithms of chapter 4 are low-complexity online methods that have proven to be of interest in a number of scenarios. For the identification of steady-state nonlinear systems, fixed-budget KRLS is capable of obtaining better results than ALD-KRLS given the same memory requirements. Additionally, both algorithms obtain acceptable results in time-varying scenarios, while other kernel-based methods such as ALD-KRLS are not capable of adjusting their solutions adequately to the changing environment. To maintain a fixed memory size, the sliding-window approach discards the oldest point in every iteration. The fixed-budget algorithm performs a more active type of learning, in that it determines the least significant point to prune in each iteration, based on an a-posteriori error minimization criterion.

Despite the fact that both methods have fairly simple architectures, they show a few advantages that could allow for some more extensions:

- Their fixed memory size is an attractive property for implementations on microprocessors, which have limited memory. This property is in stark contrast to many other methods that learn by building *growing* networks.

- Since they repeat the same basic operations in every iteration, and no growing memory needs to be taken into account, the number of computations per iteration is fixed, and the total complexity is predictable.

Since the sliding-window KRLS algorithm can be considered a special case of the fixed-budget algorithm, future research in this area will focus on improving the fixed-budget KRLS algorithm. In the following we list some interesting related future research topics.

10.1.1 Improving the discard criterion

The pruning criterion used in the original fixed-budget algorithm from chapter 4 aims to minimize the a-posteriori error, as inspired by pruning techniques for LS-SVM [de Kruif and de Vries, 2003]. It requires the calculation of the inverse kernel matrix, which supposes an expensive additional computation for most algorithms. However, KRLS already performs this computation as part of its own recursion, and hence the additionally required computational complexity is linear. Furthermore, although this criterion is closely related to optimal brain surgeon [Hassibi et al., 1993], their exact relationship remains to be studied.

More sophisticated discarding criteria can be designed by determining how well each point of the memory can be explained by the remaining points, in a “leave-one-out” strategy, which consists of the following steps. Denote by $\mathcal{D}_m$ the subset of the memory that contains all points from the memory except for the one with index m. For each possible data set $\mathcal{D}_m$, the goal of the leave-one-out criterion is to determine how much information the point-pair $(\mathbf{x}_m, y_m)$ could contribute to $\mathcal{D}_m$. Different measures of information can be applied here, such as the ALD criterion, which only takes into account the input data $\mathbf{x}_i$, or the *surprise information measure* as proposed in [Liu et al., 2010], which takes into account input data and labels. However, an effort needs to be made to maintain the computational complexity of these approaches low, as a naive implementation would yield $O(M^3)$ per iteration, where M is the number of points in memory.

10.1.2 Extensions to other kernel adaptive filtering techniques

The idea of pruning the data dictionary can be applied to other kernel-based adaptive filtering techniques, including extended KRLS [Liu and Príncipe, 2008]), kernel affine projections algorithms (KAPA) [Pokharel et al., 2008] and the kernel least mean squares (KLMS) algorithm [Liu et al., 2008]. Since EX-KRLS and KAPA are related to KRLS, they can use a similar pruning criterion as FB-KRLS, based on parameters that are already calculated in each iteration. In KLMS, on the other hand, no such parameters are available, and it would be desirable to maintain the overall linear complexity of the algorithm. Therefore, a straightforward pruning criterion for KLMS could only be based on the expansion coefficients of the kernel expansion, which are proportional to the corresponding error. The usefulness of such criteria remains to be studied.

10.1.3 Kalman filtering

Although the sliding-window and fixed-budget KRLS algorithms represent interesting techniques to perform efficient identification of static and time-varying nonlinear processes, a more elegant approach to perform nonlinear tracking would be to design a recursive state system in RKHS. While there have been some efforts in that direction (see for instance [Liu and Príncipe, 2008]), the design of a Kalman filter in feature space is still an open research topic.

10.2 Kernel CCA for Supervised Identification of Wiener and Hammerstein Systems

In chapter 5 we proposed a framework based on kernel CCA for identification of nonlinear Wiener and Hammerstein systems. We also provided an extension that allows to perform equalization of these systems. The presented algorithms included adaptive versions that allow to plug in any kernel-based implementation of the recursive least-squares algorithm. The only requirement of the chosen identification approach is that the second block in the diagram is invertible (in case of Wiener systems, the nonlinear part, and in case of Hammerstein systems, the linear filter). Fortunately, a lot of widely used models fulfill this restriction, such as Wiener systems whose nonlinearity is a weak saturation. We included simulations to demonstrate the performance of the proposed algorithms, and to show their higher robustness to noise compared with other algorithms.

Many directions for future research are open. The proposed methods can be used directly in problems with complex signals, for instance in the identification of nonlinear power amplifiers for OFDM systems [Santamaría et al., 2003]. Another possibility to explore is the application of kernel CCA to larger cascade models such as the three-block Wiener-Hammerstein systems. Also, by combining the ideas from this chapter and the techniques presented in chapter 6, a more accurate supervised identification algorithm for Hammerstein systems can be designed. In the following we present some preliminary results of this new algorithm, which we briefly explored during the preparation of this thesis.

10.2.1 Supervised identification of Hammerstein systems

An interesting property of many kernel-based algorithms is that once the kernel functions are calculated, all subsequent operations and optimizations are linear. This property is particularly interesting in problems where one needs to model a system that concatenates a nonlinear operation with one or several linear operations, such as the Hammerstein system. Consequently, a simple yet very accurate supervised identification method for Hammerstein systems can be constructed, based on kernel least-squares regression. Although a different supervised identification method for Hammerstein systems can be obtained as a specific configuration of the kernel CCA framework proposed in chapter 5 (see section 5.2.3), this method has the advantage that it imposes no restrictions on the system nor on the input data, and therefore it should perform better for the particular problem of supervised Hammerstein system identification. In this section we will briefly lay out the basic ideas behind this approach.

Consider the following kernel expansion with N basis vectors $\mathbf{x}_i$

$$\hat{y}[n] = \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}[n]) = \boldsymbol{\alpha}^T \mathbf{k}_x[n], . \tag{10.1}$$

By extending this nonlinear mapping with a convolution, one obtains a model that corresponds to the Hammerstein system

$$z[n] = h * \hat{y}[n] = \sum_{i=1}^{N} \sum_{j=0}^{L-1} h_j \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}[n-j]), \tag{10.2}$$

as illustrated in Fig. 10.1. In order to estimate the coefficients α_i and h_j, we propose to minimize the MSE between the system output $z[n]$ and the desired signal $d[n]$

$$\begin{aligned} \min_{\boldsymbol{\alpha},\mathbf{h}} J &= \frac{1}{N} \sum_{n=1}^{N} |z[n] - d[n]|^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} \left| \sum_{i=1}^{N} \sum_{j=0}^{L-1} h_j \alpha_i \kappa(\mathbf{x}_i, \mathbf{x}[n-j]) - d[n] \right|^2 , \end{aligned} \tag{10.3}$$

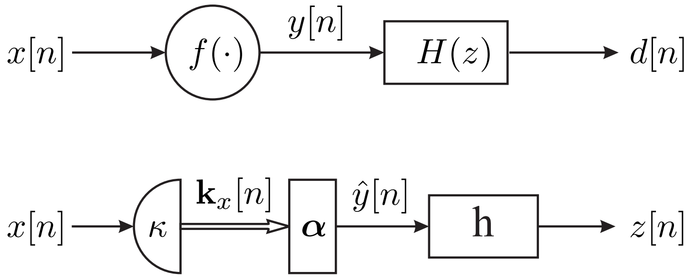
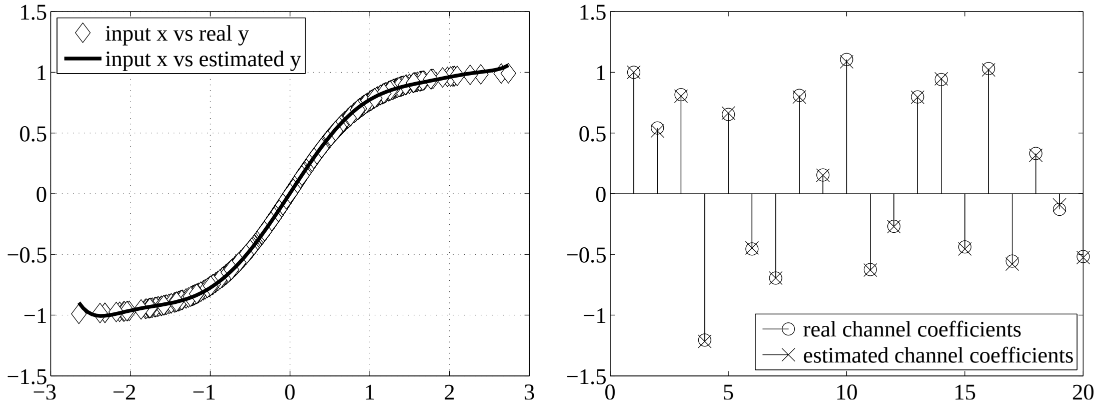


Figure 10.1: The block-diagram of a Hammerstein system (top) and the corresponding proposed identification scheme (bottom). The identification scheme contains a kernel expansion, consisting of the kernel transformation labeled κ and the expansion with coefficients α_i, followed by a FIR filter with impulse response $\mathbf{h}$. Black arrows indicate scalar variables, while the white arrow indicates that the output of the kernel transformation is a vector. The advantage of this scheme is that only the linear blocks $\boldsymbol{\alpha}$ and $\mathbf{h}$ need to be identified, while the kernel transformation guarantees that the real nonlinearity $f(\cdot)$ can be approximated as a kernel expansion up to any accuracy.

which is a function of two parameters, $\boldsymbol{\alpha}$ and $\mathbf{h}$. Inspired by the technique presented in chapter 6, this cost function can be minimized in a cyclic manner, by alternatingly fixing one of the parameters, and minimizing w.r.t. to the other. The procedure obtained in this way is an alternating kernel least-squares (AKLS) method, whose convergence is guaranteed since each step either maintains or lowers the cost. In most cases, local minima can be avoided by a smart initialization procedure. 5 In order to alleviate the computational burden of this algorithm, the dimensionality of the transformed data points can be reduced by applying centered kernel PCA before starting the iterative procedure, and maintaining only the M most significant components.

We implemented a preliminary version of this algorithm in [Van Vaerenbergh and Santamaría, 2010], which makes use of the ICD-based centered kernel PCA implementation from appendix B, and we conducted two simple experiments to demonstrate its usefulness. In a first test, the algorithm was applied to a Hammerstein system composed of the nonlinearity $f(\cdot) = \tanh(\cdot)$ followed by a 20-taps FIR filter with a randomly generated impulse response. For the input signal, 500 points were taken from $N(0,1)$ and 20dB white Gaussian noise was added to the system's output. The proposed algorithm was applied using a Gaussian kernel function with $\sigma = 0.5$. The fraction of the signal energy discarded by the kernel PCA procedure in the initialization was fixed as 10^{-14}, which resulted in discarding all but $m = 11$ eigenvectors. The obtained estimates for the nonlinearity and linear filter are shown in Fig. 10.2. Overall, the algorithm yielded a MSE of -14.5dB on the noisy data.

In a second experiment we applied this identification technique to practical data from human muscle stretch reflex dynamics. A description of how the input



(a) Nonlinearity estimate (b) Linear filter coefficients

Figure 10.2: Identification results on the Hammerstein system. (a) shows the input signal $x[n]$ vs. the real internal signal $y[n]$, and $x[n]$ vs. the estimated $\hat{y}[n]$. (b) shows the estimated channel coefficients of $\mathbf{h}$ vs. the real coefficients.

and output data was collected for system identification problem can be found in [Westwick and Kearney, 2000]. 10000 data points were used for this experiment. Although the linear filter in Hammerstein system identification is generally modeled as an IIR filter, we approximated this by a FIR filter of length 40 in a first implementation. Based on the vast literature available on linear filtering, extensions of this method that use IIR filters are straightforward.

First we applied kernel PCA using a Gaussian kernel with $\sigma = 0.01$. By fixing the discarded energy fraction to be 10^{-14}, kernel PCA maintained 257 eigenvectors. The iterative procedure was initialized by exploiting the Kronecker-like structure of the combined solution vector (see section 6.3). In Fig. 10.3, the normalized mean square error (NMSE) learning curve is displayed, which reaches a final NMSE of $9.05 \cdot 10^{-7}$, but already converged sufficiently after around 10 iterations. In terms of the number of iterations, this algorithm converges rather fast, compared with conventional methods which typically need several thousands of iterations to converge to obtain the same error. Given these positive preliminary results, we plan to investigate the different aspects of this method in future work.

10.3 Blind Equalization of Wiener Systems

The blind equalization technique from chapter 6 is based directly on the ideas introduced in chapter 5, where a kernel-based algorithm was matched to the structure of the studied system. In order to achieve a blind algorithm, we extended the blind LS equalization technique from [Xu et al., 1995] to nonlinear scenarios, and we showed how oversampling also allows to perform blind equalization of a certain class nonlinear systems. Specifically, the presented technique is capable of blindly equalizing oversampled Wiener systems or SIMO systems whose input-output relations can be

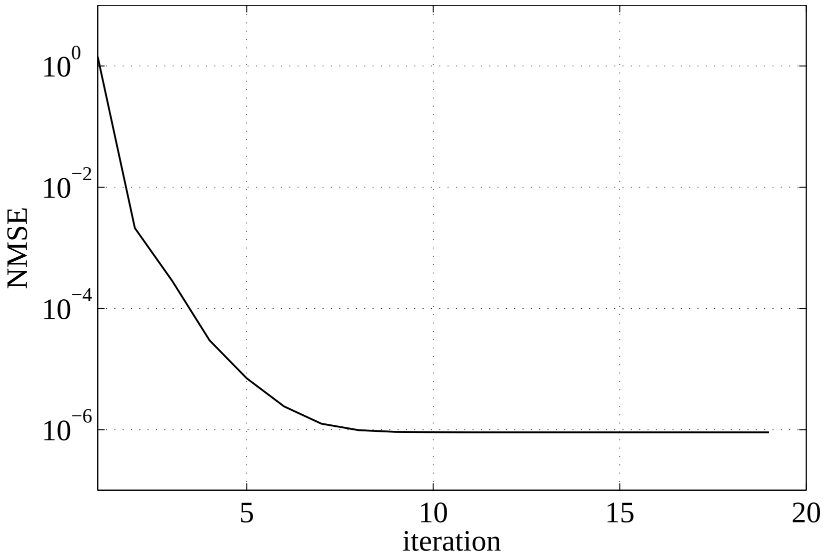


Figure 10.3: NMSE of the proposed method on the stretch reflex data set.

modeled as Wiener systems. Some other blind techniques have been proposed that require more than one output channel, with application in blind Volterra systems equalization. However, these techniques require at least 3 output channels, while the proposed method is capable of operating correctly when only 2 outputs are available.

In the following, we will briefly discuss several new algorithms based on the ideas presented in chapter 6.

10.3.1 Blind equalization of SISO Wiener systems

In [Taleb et al., 2001], a method is presented for blind equalization of Wiener systems. This method aims to invert a Wiener system by estimating a Hammerstein system that is composed of the inverse blocks of the original system. In order to estimate this inverse system, it minimizes the mutual information of the entire structure's output signal. To represent the nonlinearity, the authors chose to use a kernel expansion. The results obtained in chapter 6 suggest that some of the introduced ideas could improve the performance of this algorithm. Among others, the application of kernel PCA in a first stage could improve the noise-robustness and reduce the overall computational complexity.

The blind identification of Wiener systems remains an active research topic, as shown by the number of recent publications on this topic (see for instance the maximum-likelihood approach from [Vanbeylen et al., 2009]).

10.3.2 Post-nonlinear blind source separation

The mixture model of the main problem presented in chapter 6 is known as a *post-nonlinear* (PNL) mixture. In general, the post-nonlinear model is a well known mixture model in the field of blind source separation (BSS), that consists of a linear mixture followed by a componentwise nonlinear transformation. It is frequently used as a simplified nonlinear model, and it has the advantage that in some cases the nonlinearity can be inverted. This idea is also exploited in chapter 9.

In the proposed blind equalization method for Wiener systems from chapter 6, a specific post-nonlinear mixture model is used, in which only one source is present and the linear mixtures include convolutions. However, the ideas used to design this method could be exploited to solve other BSS problems, such as the more standard PNL-BSS scenario in which multiple sources are present and the linear mixing process is instantaneous.

10.4 Blind Decoding of Time-Varying MIMO Systems

In chapter 8 we treated the problem of blind decoding of fast time-varying MIMO channels, which is a complex problem in the area of communications. Since typical signals consist of symbols that belong to a finite alphabet here, we proposed a clustering approach to decode received data received in these MIMO systems. The proposed technique extends the scatter plot with a temporal dimension, and then applies spectral clustering to retrieve threads that correspond to the different data symbols. It also exploits the symmetry of the M-PSK constellation in different ways to improve robustness. In a second contribution, we optimized the kernel function used by the clustering method so that it favors thread-like structures. We also presented an extension for Alamouti-coded signals, and discussed a hierarchical-clustering procedure to replace the k-means clustering of the eigenvector data in the spectral clustering problem. This modification allows to take into account the temporal information again, which leads to higher robustness in a number of situations.

At high normalized Doppler frequencies, the proposed algorithm outperforms other adaptive techniques using less pilot symbols, thus allowing more information to be sent. However, it requires the calculation of the eigenvectors of its kernel matrix, which generally requires $O(N^3)$ operations. In most cases this can be lowered to $O(N^2)$ using any of the techniques for efficient implementation described in section 7.2.2, which, nevertheless, is still prohibitive in most real-time applications. This analysis suggests that the proposed spectral clustering based technique could be used as an initialization for an adaptive algorithm. Specifically, given only a few pilot symbol slots it can estimate a short symbol vector sequence which can be used as a pilot sequence for a (computationally more efficient) decision-based adaptive algorithm.

The problem addressed in chapter 8 applies spectral clustering to data that belong to a certain geometric model. As mentioned a few times in this thesis, an important guideline for kernel design in spectral clustering and kernel methods in general is that the kernel should reflect as much knowledge about the problem as possible. In that sense, the kernel represents the prior information about the addressed problem. While the connectivity kernel used in this chapter represents a fairly simple model, more complex parametric kernels can be designed for related problems. In the following we mention a few possible extensions to the presented method.

10.4.1 Incorporating gradient information in clustering

A first direction for future work based on chapter 8 consists of the inclusion of gradient information in the connectivity kernel, which determines the local gradient of the curve-shaped cluster for each point. Equipped with this information, the kernel should assign a high similarity only to close points that show a similar gradient.

First of all, this is very likely to increase the robustness of spectral clustering for thread-like clusters, since the gradient information will favor the detection of smooth curves. Second, this would also increase the possibilities of spectral clustering to detect overlapping clusters, such as two crossing threads. On the other hand, the computation of the gradient in a path-based kernel means a significant increase in computational complexity. Moreover, it is likely to introduce an additional smoothness parameter, which needs to be determined for each application, in addition to the other kernel parameters.

An interesting application of a gradient-based spectral clustering would be the problem of K-curves clustering, in which a number of points have to be grouped into K smooth curves.

10.4.2 Multi-target clustering

The blind MIMO decoding problem with fast time-varying channels considered that at each time instant there was only one observation that belonged *only to one source*. A different problem is obtained when at each time instant there is one observation for *every source*. The proposed clustering method based on the connectivity kernel only requires a minimal change to operate on such data: in the calculation of the connectivity kernel, we only have to exclude paths between points that occur at the same time instant. Such a situation can be encountered for instance in the problem of *multi-target clustering*, which is traditionally solved by adaptive methods such as Kalman filtering or particle filtering.

10.5 Underdetermined Post-Nonlinear Blind Source Separation

In chapter 9 we proposed a clustering method to treat the post-nonlinear BSS scenario in which fewer mixtures than sources are available. By exploiting the sparseness of the source signals, the proposed spectral clustering method detects samples that correspond to time instants on which only one source is active. With the obtained information, it is capable of compensating for the nonlinearities that operate on each mixture. For this purpose, we proposed an iterative estimator based on a coupled training of different MLPs. Once the nonlinearities are estimated, a linear underdetermined source separation problem is obtained, for which any linear technique can be applied.

While the technique from chapter 9 was chronologically the first contribution to this thesis, a future research topic could consist in improving it by applying the techniques presented in the rest of this thesis. Specifically, the MLP-based estimation of the nonlinearities could be replaced by a method based on kernel regression. Below we summarize the main ideas of this approach, which yields a kernel canonical correlation based estimation.

10.5.1 Estimation of the nonlinearities using kernel CCA

In chapter 9, each nonlinearity $g_j(\cdot)$ was represented by an MLP, whose parameters needed to be estimated through back-propagation. The framework of kernel methods allows for a more elegant estimation of these nonlinearities. To this end, the nonlinearities $g_j(\cdot)$ are first represented as kernel expansions

$$g_j(x) = \sum_i \alpha_i^j \kappa(x_i, x), \tag{10.4}$$

where x_i are some one-dimensional support points. Specifically, this support can be chosen as the set containing the j-th component of all points $\mathbf{x}[n]$ of all clusters, for $n = 1, \ldots, N$.

By plugging this representation of the nonlinearities into the cost function (9.7) and adopting a matrix notation, an algebraically more interesting formulation of the minimization problem can be obtained. Denote by $\mathbf{x}_j^i$ the vector containing the j-th component of all points in cluster $\mathcal{C}_i$. By transforming this vector into feature space, we obtain the data matrix $\tilde{\mathbf{X}}_j^i$. The kernel matrix between component j of cluster i_1 and cluster i_2 can now be written as $\mathbf{K}_j^{i_1,i_2} = \tilde{\mathbf{X}}_j^{i_1}(\tilde{\mathbf{X}}_j^{i_2})^T$. A large kernel matrix containing the contribution of the j-th component of every mixture point $\mathbf{x}[n]$ is then obtained as

$$\mathbf{K}_j = \begin{bmatrix} \mathbf{K}_j^{1,1} & \ldots & \mathbf{K}_j^{1,n_s} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_j^{n_s,1} & \ldots & \mathbf{K}_j^{n_s,n_s} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{X}}_j^1(\tilde{\mathbf{X}}_j^1)^T & \ldots & \tilde{\mathbf{X}}_j^1(\tilde{\mathbf{X}}_j^{n_s})^T \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{X}}_j^{n_s}(\tilde{\mathbf{X}}_j^1)^T & \ldots & \tilde{\mathbf{X}}_j^{n_s}(\tilde{\mathbf{X}}_j^{n_s})^T \end{bmatrix}. \tag{10.5}$$

This allows to rewrite (9.7) as the minimization problem

$$\begin{aligned} \min_{\boldsymbol{\alpha},\mathbf{P}} J &= \sum_{j=1}^{m}\sum_{k=1}^{m} \left\|\mathbf{K}_j\boldsymbol{\alpha}_j - \mathbf{P}_{jk}\mathbf{K}_k\boldsymbol{\alpha}_k\right\|^2 \qquad (10.6) \\ \text{s.t.} & \quad \sum_{j=1}^{m} \left\|\mathbf{K}_j\boldsymbol{\alpha}_j\right\|^2 = 1, \end{aligned}$$

where $\boldsymbol{\alpha}_j$ contains all elements α_i^j and $\mathbf{P}_{jk}$ is a diagonal matrix containing the scaling factors between the components j and k of each cluster

$$\mathbf{P}_{jk} = \begin{bmatrix} \mathbf{P}_{jk}^1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{jk}^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{jk}^{n_s} \end{bmatrix}. \tag{10.7}$$

Here, each matrix $\mathbf{P}^i_{jk}$ is a diagonal matrix containing only instances of the element ρ^i_{jk} on its diagonal[1].

Given an estimate of each slope factor ρ^i_{jk}, the minimization problem (10.6) can be rewritten as the following kernel CCA problem (see appendix D)

$$\frac{1}{n_s}\begin{bmatrix} \mathbf{P}_{1,1}\mathbf{K}_1 & \dots & \mathbf{P}_{1,n_s}\mathbf{K}_{n_s} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{n_s,1}\mathbf{K}_1 & \dots & \mathbf{P}_{n_s,n_s}\mathbf{K}_{n_s} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_{n_s} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{K}_1 & \dots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \dots & \mathbf{K}_{n_s} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_{n_s} \end{bmatrix}, \tag{10.8}$$

or, in short,

$$\mathbf{R}\boldsymbol{\alpha} = \lambda\mathbf{D}\boldsymbol{\alpha} \tag{10.9}$$

On the other hand, given an estimate of $\boldsymbol{\alpha}_j$, the CCA problem can be solved for ρ^i_{jk} by minimizing the cost function (10.8) with respect to ρ^i_{jk}, which yields a simple LS problem. In this manner a cyclic minimization process can be designed that switches between optimizing the nonlinearities and the scaling factor, until convergence is reached, similar to the technique introduced in chapter 6.

[1] Remind that $\rho^i_{jk} = 1$ for $j \geq k$.

Appendix A
Data Centering in Feature Space

A number of techniques, such as kernel PCA, require the data to be centered in feature space. While the explicit computation of the centered data in feature space represents a very complex problem for most kernels, it is fairly easy to calculate the *kernel matrix* of the centered data.

In this appendix, we first derive the most general case of data centering: A kernel matrix is to be calculated between two data sets, $\mathcal{A}$ and $\mathcal{B}$, that are centered in feature space with respect to, respectively, data sets $\mathcal{C}$ and $\mathcal{D}$; from $\mathcal{A}$ we remove the mean of $\mathcal{C}$, and from $\mathcal{B}$ we remove the mean of $\mathcal{D}$. Once the formula of this general case is obtained, more practical cases are derived from it, where some of these defined data sets coincide.

We denote the first data set as $\mathcal{A}$, its number of points data as $N_{\mathcal{A}}$, and the data points themselves as $\mathbf{x}_i^{\mathcal{A}}$, $i = 1, 2, \ldots, N_{\mathcal{A}}$, which after transformation to feature space become $\tilde{\mathbf{x}}_i{}^{\mathcal{A}}$. Similar notation is used for data sets $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$. The elements of the kernel matrix $K^{\mathcal{AB}}$ between data sets $\mathcal{A}$ and $\mathcal{B}$ are denoted as $k_{ij}^{\mathcal{AB}} = \kappa(\mathbf{x}_i^{\mathcal{A}}, \mathbf{x}_j^{\mathcal{B}}) = (\tilde{\mathbf{x}}_i^{\mathcal{A}})^T \tilde{\mathbf{x}}_j^{\mathcal{B}}$.

When data sets $\mathcal{A}$ and $\mathcal{B}$ are centered in feature space respectively w.r.t. data sets $\mathcal{C}$ and $\mathcal{D}$, the elements of the resulting centered kernel matrix are obtained as

$$\begin{aligned}
k_{ij}^{\mathcal{AB},\mathcal{CD}} &= \left(\tilde{\mathbf{x}}_i^{\mathcal{A}} - \frac{1}{N_{\mathcal{C}}}\sum_{k=1}^{N_{\mathcal{C}}} \tilde{\mathbf{x}}_k^{\mathcal{C}}\right)^T \left(\tilde{\mathbf{x}}_j^{\mathcal{B}} - \frac{1}{N_{\mathcal{D}}}\sum_{k=1}^{N_{\mathcal{D}}} \tilde{\mathbf{x}}_k^{\mathcal{D}}\right) \\
&= (\tilde{\mathbf{x}}_i^{\mathcal{A}})^T \tilde{\mathbf{x}}_j^{\mathcal{B}} - \frac{1}{N_{\mathcal{D}}}\sum_{k=1}^{N_{\mathcal{D}}} (\tilde{\mathbf{x}}_i^{\mathcal{A}})^T \tilde{\mathbf{x}}_k^{\mathcal{D}} - \frac{1}{N_{\mathcal{C}}}\sum_{k=1}^{N_{\mathcal{C}}} (\tilde{\mathbf{x}}_k^{\mathcal{C}})^T \tilde{\mathbf{x}}_j^{\mathcal{B}} \\
&\quad + \frac{1}{N_{\mathcal{C}} N_{\mathcal{D}}} \left(\sum_{k=1}^{N_{\mathcal{C}}} \tilde{\mathbf{x}}_k^{\mathcal{C}}\right)^T \left(\sum_{k=1}^{N_{\mathcal{D}}} \tilde{\mathbf{x}}_k^{\mathcal{D}}\right).
\end{aligned}$$

Denoting the all-ones matrix of size $N_{\mathcal{A}}$ by $N_{\mathcal{B}}$ as $\mathbf{1}^{\mathcal{AB}}$, we obtain the centered kernel matrix as

$$\mathbf{K}^{\mathcal{AB},\mathcal{CD}} = \mathbf{K}^{\mathcal{AB}} - \frac{1}{N_{\mathcal{C}}}\mathbf{1}^{\mathcal{AC}}\mathbf{K}^{\mathcal{CB}} - \frac{1}{N_{\mathcal{D}}}\mathbf{K}^{\mathcal{AD}}\mathbf{1}^{\mathcal{DB}} + \frac{1}{N_{\mathcal{C}} N_{\mathcal{D}}}\mathbf{1}^{\mathcal{AC}}\mathbf{K}^{\mathcal{CD}}\mathbf{1}^{\mathcal{DB}}. \tag{A.1}$$

In case the data sets $\mathcal{C}$ and $\mathcal{D}$ coincide, the centered kernel matrix is found as

$$\mathbf{K}^{\mathcal{AB},\mathcal{C}} = \mathbf{K}^{\mathcal{AB}} - \frac{1}{N_{\mathcal{C}}}\mathbf{1}^{\mathcal{AC}}\mathbf{K}^{\mathcal{CB}} - \frac{1}{N_{\mathcal{C}}}\mathbf{K}^{\mathcal{AC}}\mathbf{1}^{\mathcal{CB}} + \frac{1}{N_{\mathcal{C}}^2}\mathbf{1}^{\mathcal{AC}}\mathbf{K}^{\mathcal{CC}}\mathbf{1}^{\mathcal{CB}}. \tag{A.2}$$

When the data sets $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ coincide, the centered kernel matrix is found as

$$\mathbf{K}^{\mathcal{AB},\mathcal{B}} = \left(\mathbf{K}^{\mathcal{AB}} - \frac{\mathbf{1}^{\mathcal{AB}}}{N^{\mathcal{B}}}K^{\mathcal{BB}}\right)\left(\mathbf{I} - \frac{\mathbf{1}^{\mathcal{BB}}}{N^{\mathcal{B}}}\right), \tag{A.3}$$

where $\mathbf{I}$ is the unit matrix. In case all data sets coincide, $\mathcal{A} = \mathcal{B} = \mathcal{C} = \mathcal{D}$, the previous equation takes the form

$$\mathbf{K}^{\mathcal{AA},\mathcal{A}} = \left(\mathbf{I} - \frac{\mathbf{1}^{\mathcal{AA}}}{N^{\mathcal{A}}}\right)\mathbf{K}^{\mathcal{AA}}\left(\mathbf{I} - \frac{\mathbf{1}^{\mathcal{AA}}}{N^{\mathcal{A}}}\right), \tag{A.4}$$

as introduced for instance in [Schölkopf et al., 1998].

In practical cases, centering is mostly performed using (A.4), where we wish to obtain the kernel matrix for only one data set and remove its mean in feature space. For feature extraction, it is often necessary to compute the images of test patterns that are centered with respect to a given data set. This case corresponds to the centering formula obtained in (A.3). The remaining formulas, (A.2) and (A.1), represent cases that can occur in applications such as pre-imaging [Kwok and Tsang, 2004].

Appendix B
Incomplete Cholesky Decomposition for Kernel Matrix Computation

B.1 Incomplete Cholesky Decomposition

In this appendix we review a common technique to obtain a low-rank approximation of Gram matrices. For this purpose, we employ the incomplete Cholesky decomposition (ICD), as introduced in [Fine and Scheinberg, 2001, Bach and Jordan, 2002], which allows to approximate the entire Gram matrix in $O(M^2N)$ operations, where N is the number of data points and M is the rank of the approximated matrix.

Any $N \times N$ symmetric positive definite matrix $\mathbf{K}$ can be expressed as $\mathbf{K} = \mathbf{G}\mathbf{G}^T$, where $\mathbf{G}$ is an $N \times N$ lower triangular matrix with positive diagonal entries. This decomposition is known as the *Cholesky decomposition*, which is a special case of LU decomposition for symmetric positive definite matrices [Golub and Van Loan, 1996]. The goal of *incomplete* Cholesky decomposition, however, is to find a matrix $\bar{\mathbf{G}}$ of size $N \times M$, for small M, such that the difference $\mathbf{K} - \bar{\mathbf{G}}\bar{\mathbf{G}}^T$ becomes arbitrarily small. In practice, this approximation becomes possible when the eigenspectrum of $\mathbf{K}$ decays quickly.

In order to avoid the $O(N^2)$ operations required to calculate the kernel matrix $\mathbf{K}$ itself, $\bar{\mathbf{G}}$ must be computed without accessing all elements of $\mathbf{K}$. At each step i, incomplete Cholesky decomposition achieves this by constructing the matrix $\bar{\mathbf{G}}_i$ out of the columns of $\mathbf{K}$ for which the diagonal elements (the "pivots") of the residual $\mathbf{K} - \bar{\mathbf{G}}_i\bar{\mathbf{G}}_i^T$ are above a certain threshold. As a result, the only elements of $\mathbf{K}$ that are needed in memory are the diagonal elements. Most other elements are never used and those that are needed can be calculated as simple kernel evaluations on the input data.

The value of M can either be fixed or determined by requiring a certain approximation accuracy $\|\mathbf{K} - \bar{\mathbf{G}}\bar{\mathbf{G}}^T\| < \epsilon$, where ϵ is a small positive number. Algorithm B.1 presents the entire pivot-based algorithm to obtain incomplete Cholesky decomposition, for a given accuracy threshold ϵ, including a speedup procedure introduced by Seth et al. in [Seth and Príncipe, 2009]. This algorithm uses the notation $\bar{\mathbf{G}}_{a:b,c:d}$ and $\bar{\mathbf{G}}_{:,i}$ to refer to the matrix extracted from $\bar{\mathbf{G}}$ by taking the rows a to b and columns c

Algorithm B.1 Incomplete Cholesky Decomposition

```
initialize
    Permutation vector: $\mathbf{p} = [1, 2, \ldots, N]$.
    Diagonal of the residual matrix: $\mathbf{d} = \mathrm{diag}(\mathbf{K})$.
    First column of $\bar{\mathbf{G}}$: $\bar{\mathbf{G}}_{:,1} = \mathbf{K}_{:,1}$.
    $i = 1$.
while $\sum_{j=i}^{N} \mathbf{d}_j > \epsilon$ do
    if $i > 1$ then
        Update the residual diagonal: $\mathbf{d}_{i:n} = \mathrm{diag}(\mathbf{K}_{i:N,i:N}) - \bar{\mathbf{G}}_{i:N,1:i-1}\bar{\mathbf{G}}_{i:N,1:i-1}^T\mathbf{1}$.
    end if
    Find the new best element: $j^* = \mathrm{argmax}_{i \le j \le N}\mathbf{d}_j$.
    Update the permutation: $\mathbf{p}_i \leftrightarrow \mathbf{p}_{j^*}$.
    Permute rows $i$ and $j^*$: $\bar{\mathbf{G}}_{i,1:i-1} \leftrightarrow \bar{\mathbf{G}}_{j^*,1:i-1}$.
    Update the diagonal: $\bar{\mathbf{G}}_{i,i} = \sqrt{\mathbf{d}_{j^*,j^*}}$.
    Calculate the $i$-th column: $\bar{\mathbf{G}}_{i+1:N,i} = \frac{1}{\bar{\mathbf{G}}_{i,i}}(\mathbf{K}_{\mathbf{p}_{i+1:N},\mathbf{p}_i} - \bar{\mathbf{G}}_{i+1:N,1:i-1}\bar{\mathbf{G}}_{i,1:i-1}^T)$.
    $i \leftarrow i + 1$.
end while
Sort the rows of $\bar{\mathbf{G}}$ according to $\mathbf{p}$: $\bar{\mathbf{G}} \leftarrow \bar{\mathbf{G}}_{\mathbf{p},:}$.
Output $\bar{\mathbf{G}}$ and $M = i - 1$.
```

to d, and by taking the i-th column, respectively, and $\mathbf{1}$ denotes the all-ones vector of adequate length.

B.2 Low-rank Approximation of the Centered Kernel Matrix

The centered kernel matrix of a data set can be approximated easily after the incomplete Cholesky decomposition $\bar{\mathbf{G}}\bar{\mathbf{G}}^T$ has been obtained. Specifically, the centered low-rank approximation of the kernel matrix is found as

$$\begin{aligned}
(\bar{\mathbf{G}}\bar{\mathbf{G}}^T)_{centered} &= (\mathbf{I} - \mathbf{1}/N)\bar{\mathbf{G}}\bar{\mathbf{G}}^T(\mathbf{I} - \mathbf{1}/N) && \text{(B.1)}\\
&= (\bar{\mathbf{G}} - \mathbf{1}\bar{\mathbf{G}})(\bar{\mathbf{G}} - \mathbf{1}\bar{\mathbf{G}})^T && \text{(B.2)}\\
&= \bar{\mathbf{G}}_c\bar{\mathbf{G}}_c^T. && \text{(B.3)}
\end{aligned}$$

Not surprisingly, the approximate kernel matrix $\bar{\mathbf{G}}\bar{\mathbf{G}}^T$ can be centered in feature space by simply removing the column means from $\bar{\mathbf{G}}$. For more specific centering schemes, such as the ones presented in appendix A that involve different data sets, the centered low-rank kernel matrix can be obtained in a straightforward fashion based on the approximation for the asymmetric kernel matrix between different data sets, as discussed in [Seth and Príncipe, 2009].

B.3 Approximate Kernel PCA based on ICD

The described low-rank approximation technique is closely related to kernel PCA. In particular, ICD determines an M-dimensional subspace of the feature space, spanned by the columns of $\bar{\mathbf{G}}$, by considering only the most significant pivots. Although the columns of $\bar{\mathbf{G}}$ constitute a basis of this subspace, in general they are not orthogonal. Orthogonality (and more specifically, orthonormality) can be obtained by solving the following $M \times M$ eigenvalue problem

$$\frac{1}{N}\bar{\mathbf{G}}^T\bar{\mathbf{G}}\mathbf{v} = \lambda\mathbf{v}, \tag{B.4}$$

and using the coefficients of the resulting eigenvectors $\mathbf{v}$ as weights to combine the columns of $\bar{\mathbf{G}}$ into different basis vectors. Although this subspace is not obtained by the least-squares criterion necessary for PCA, in practice it is very close to the optimal PCA subspace and it only differs in the least significant dimension. Therefore, this procedure allows to construct a fast approximation of kernel PCA, especially useful in problems with large data sets, since it does not require to calculate the entire kernel matrix. It also allows to calculate the *centered* kernel PCA decomposition, by simply removing the column means from the obtained basis vectors (as in section B.2).

Appendix C
Matrix Inversion Formulas

In this appendix we present formulas to obtain the inverse of a matrix from which certain rows and columns are removed or to which rows and columns are added. Given the original matrix and its inverse, the matrix inversion lemma is applied to obtain efficient formulas that avoid inversion of the entire matrix. The overall computational complexity of each formula is $O(M^2)$, where M is the number of rows and columns of the original matrix.

The matrix inversion lemma states the following: Let **A** and **B** be two positive-definite matrices of size $M \times M$ that satisfy

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^T, \tag{C.1}$$

where **D** is a positive-definite $L \times L$ matrix and **C** is an $L \times M$ matrix. The inverse of matrix **A** can be expressed as

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^T\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{B}. \tag{C.2}$$

C.1 The Inverse of an Upsized Matrix

By *upsizing* a matrix we refer to adding one row and one column to it. In case a row and column are added *at the end*, we obtain the following formulas. Specifically, the matrix **A** shown below is upsized to the matrix **K**. Given the inverse matrix $\mathbf{A}^{-1}$ and the upsized matrix **K**, the inverse matrix $\mathbf{K}^{-1}$ can be obtained as follows:

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{b}^T & d \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{E} & \mathbf{f} \\ \mathbf{f}^T & g \end{bmatrix}$$

$$\Rightarrow \left\{ \begin{array}{lcl} \mathbf{A}\mathbf{E} + \mathbf{b}\mathbf{f}^T & = & \mathbf{I} \\ \mathbf{A}\mathbf{f} + \mathbf{b}g & = & \mathbf{0} \\ \mathbf{b}^T\mathbf{f} + dg & = & 1 \end{array} \right.$$

$$\Rightarrow \mathbf{K}^{-1} = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{I} + \mathbf{b}\mathbf{b}^T\mathbf{A}^{-1H}g) & -\mathbf{A}^{-1}\mathbf{b}g \\ -(\mathbf{A}^{-1}\mathbf{b})^Tg & g \end{bmatrix}, \tag{C.3}$$

with $g = (d - \mathbf{b}^T\mathbf{A}^{-1}\mathbf{b})^{-1}$.

C.2 The Inverse of a Downsized Matrix

By *downsizing* a matrix **K**, we denote the operation of removing a row and column of the matrix. We start with the basic case of removing the first row and column.

C.2.1 Removing the first row and column

When the first row and column of matrix **K** are removed, the matrix **D** is obtained. The inverse matrix $\mathbf{D}^{-1}$ can be expressed in terms of the known elements of $\mathbf{K}^{-1}$ as follows:

$$\mathbf{K} = \begin{bmatrix} a & \mathbf{b}^T \\ \mathbf{b} & \mathbf{D} \end{bmatrix}, \quad \mathbf{K}^{-1} = \begin{bmatrix} e & \mathbf{f}^T \\ \mathbf{f} & \mathbf{G} \end{bmatrix}$$

$$\Rightarrow \left\{ \begin{array}{lcl} \mathbf{b}e + \mathbf{D}\mathbf{f} & = & \mathbf{0} \\ \mathbf{b}\mathbf{f}^T + \mathbf{D}\mathbf{G} & = & \mathbf{I} \end{array} \right.$$

$$\Rightarrow \mathbf{D}^{-1} = \mathbf{G} - \mathbf{f}\mathbf{f}^T/e. \tag{C.4}$$

C.2.2 Removing an arbitrary row and column

The procedure that removes an arbitrary row and column is similar to operation needed to remove the first row and column. Given a matrix **L** and its inverse $\mathbf{L}^{-1}$, we remove the i-th row and column of **L** and would like to obtain the inverse of this matrix only in terms of **L** and $\mathbf{L}^{-1}$.

First of all, notice that it is possible to swap the i-th row and column of **L** with its first row and column by pre- and post-multiplying it with the following *symmetric permutation matrix* $\mathbf{P}_i$ of size M

$$\mathbf{P}_i = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{i-2} & \mathbf{0} & \mathbf{0} \\ 1 & \mathbf{0} & 0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{M-i} \end{bmatrix}, \tag{C.5}$$

where $\mathbf{I}_j$ is the unit matrix of size j and **0** is the all-zeroes matrix of adequate dimensions. This matrix is obtained by swapping the i-th and the first row and column of the unit matrix **I**. It has the important property that it is equal to its inverse $\mathbf{P}_i^{-1} = \mathbf{P}_i$. A related matrix also needed for this inversion procedure is

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{0} & \mathbf{I}_{i-1} & \mathbf{0} \\ 1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_{M-i} \end{bmatrix}, \tag{C.6}$$

which is orthogonal, since $\mathbf{Q}_i\mathbf{Q}_i^T = \mathbf{I}$. Pre- and post-multiplying a matrix by $\mathbf{Q}_i$ puts its i-th row and column in front of all others.

The entire procedure to obtain the inverse can be found in Alg. C.1. This procedure can be explained as follows. First of all, the first row and column of **K** correspond to the i-th row and column of the original matrix **L**. Second, all but one

Algorithm C.1 Procedure to obtain the inverse of a matrix whose i-th row and column are removed

```
Compute $\mathbf{K} = \mathbf{P}_i\mathbf{L}\mathbf{P}_i$ and $\mathbf{K}^{-1} = \mathbf{P}_i\mathbf{L}^{-1}\mathbf{P}_i$, with $\mathbf{P}_i$ from (C.5).
Remove the first row and column of $\mathbf{K}$ to obtain $\mathbf{D}$.
Calculate the inverse $\mathbf{D}^{-1}$ by applying (C.4).
Obtain $\mathbf{Q}_i\mathbf{D}^{-1}\mathbf{Q}_i$, with $\mathbf{Q}_i$ from (C.6).
```

of the elements of the original row and column are found as the $i-1$-th row and column of $\mathbf{D}$. In the final step this row and column are restored as the first row and column by placing them in front of the other rows and columns. Finally, notice that all multiplications with $\mathbf{P}_i$ and $\mathbf{Q}_i$ can be performed by simple row- and column-swap operations. In practice, $\mathbf{P}_i$ and $\mathbf{Q}_i$ never have to be calculated nor stored.

Appendix D
Canonical Correlation Analysis

D.1 Canonical Correlation Analysis for Two Data Sets

Consider two random zero-mean variables $\mathbf{x}_1$ and $\mathbf{x}_2$ that are multi-dimensional. Canonical correlation analysis (CCA) is the problem of finding basis vectors for these variables such that the correlation between the projections of the variables onto these basis vectors are mutually maximized [Hotelling, 1936, Hardoon et al., 2003]. Contrary to standard correlation analysis, which is dependent on the coordinate system in which the variables are described, CCA seeks a pair of optimal linear transformations for each of the sets of variables, such that the transformed variables are maximally correlated. Here, we will follow the analysis from [Hardoon et al., 2003].

Denote by $\mathbf{h}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{h}_2 \in \mathbb{R}^{n_2}$ the respective projection vectors, and by $y_1 = \mathbf{x}_1^T\mathbf{h}_1$ and $y_2 = \mathbf{x}_2^T\mathbf{h}_2$ the obtained transformed one-dimensional variables. Notice that the dimensionality of $\mathbf{x}_1 \in \mathbb{R}^{n_1}$ and $\mathbf{x}_2 \in \mathbb{R}^{n_2}$ does not need to be the same. Assume we are given N observations of each of the variables, $\mathbf{x}_1[1], \dots, \mathbf{x}_1[N]$ and $\mathbf{x}_2[1], \dots, \mathbf{x}_2[N]$, which can be stacked row-wise in order to obtain the corresponding data matrices $\mathbf{X}_1 \in \mathbb{R}^{N\times n_1}$ and $\mathbf{X}_2 \in \mathbb{R}^{N\times n_2}$. The obtained projections are combined in a similar fashion into data vectors $\mathbf{y}_1 = \mathbf{X}_1\mathbf{h}_1$ and $\mathbf{y}_2 = \mathbf{X}_2\mathbf{h}_2$.

With this notation, the function to be maximized is

$$\begin{aligned}\rho &= \max_{\mathbf{h}_1,\mathbf{h}_2} \frac{\mathbf{h}_1^T\mathbf{X}_1^T\mathbf{X}_2\mathbf{h}_2}{\sqrt{\mathbf{h}_1^T\mathbf{X}_1^T\mathbf{X}_1\mathbf{h}_1\mathbf{h}_2^T\mathbf{X}_2^T\mathbf{X}_2\mathbf{h}_2}} \\ &= \max_{\mathbf{h}_1,\mathbf{h}_2} \frac{\mathbf{h}_1^T\mathbf{R}_{12}\mathbf{h}_2}{\sqrt{\mathbf{h}_1^T\mathbf{R}_{11}\mathbf{h}_1\mathbf{h}_2^T\mathbf{R}_{22}\mathbf{h}_2}},\end{aligned} \tag{D.1}$$

where we have introduced the covariance matrices $\mathbf{R}_{ij} = \mathbf{X}_i^T\mathbf{X}_j$, with $i, j = 1, 2$. Notice that (D.1) is not affected by scaling $\mathbf{h}_1$ or $\mathbf{h}_2$, since

$$\frac{a\mathbf{h}_1^T\mathbf{R}_{12}\mathbf{h}_2}{\sqrt{a^2\mathbf{h}_1^T\mathbf{R}_{11}\mathbf{h}_1\mathbf{h}_2^T\mathbf{R}_{22}\mathbf{h}_2}} = \frac{\mathbf{h}_1^T\mathbf{R}_{12}\mathbf{h}_2}{\sqrt{\mathbf{h}_1^T\mathbf{R}_{11}\mathbf{h}_1\mathbf{h}_2^T\mathbf{R}_{22}\mathbf{h}_2}}. \tag{D.2}$$

Therefore, the CCA problem (D.1) is equivalent to maximizing the numerator subject to the constraint

$$\begin{cases} \mathbf{h}_1^T \mathbf{R}_{11} \mathbf{h}_1 = 1 \\ \mathbf{h}_2^T \mathbf{R}_{22} \mathbf{h}_2 = 1. \end{cases} \tag{D.3}$$

The corresponding Lagrangian is

$$\mathcal{L}(\mathbf{h}_1, \mathbf{h}_2, \lambda) = \mathbf{h}_1^T \mathbf{R}_{12} \mathbf{h}_2 - \lambda_1 \left(\mathbf{h}_1^T \mathbf{R}_{11} \mathbf{h}_1 - 1 \right) - \lambda_2 \left(\| \mathbf{h}_2^T \mathbf{R}_{22} \mathbf{h}_2 - 1 \right). \tag{D.4}$$

Deriving to both projection vectors leads to

$$\frac{\delta \mathcal{L}}{\delta \mathbf{h}_1} = \mathbf{R}_{12} \mathbf{h}_2 - 2\lambda_1 \mathbf{R}_{11} \mathbf{h}_1 = 0 \tag{D.5}$$

$$\frac{\delta \mathcal{L}}{\delta \mathbf{h}_2} = \mathbf{R}_{21} \mathbf{h}_1 - 2\lambda_2 \mathbf{R}_{22} \mathbf{h}_2 = 0. \tag{D.6}$$

Subtracting $\mathbf{h}_2^T$ times Eq. (D.6) from $\mathbf{h}_1^T$ times Eq. (D.5) yields

$$\begin{aligned} 0 &= \mathbf{h}_1^T \mathbf{R}_{12} \mathbf{h}_2 - 2\lambda_1 \mathbf{h}_1^T \mathbf{R}_{11} \mathbf{h}_1 - \mathbf{h}_2^T \mathbf{R}_{21} \mathbf{h}_1 + 2\lambda_2 \mathbf{h}_2^T \mathbf{R}_{22} \mathbf{h}_2 \\ &= 2\lambda_2 \mathbf{h}_2^T \mathbf{R}_{22} \mathbf{h}_2 - 2\lambda_1 \mathbf{h}_1^T \mathbf{R}_{11} \mathbf{h}_1. \end{aligned} \tag{D.7}$$

Together with the constraints (D.3), this implies that $\lambda_1 - \lambda_2 = 0$, and we can denote $\lambda = \lambda_1 = \lambda_2$. Therefore, the Lagrangian (D.4) can be maximized by solving the following generalized eigenvalue problem (GEV), which is obtained directly from (D.5) and (D.6)

$$\frac{1}{2} \begin{bmatrix} \mathbf{0} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix}. \tag{D.8}$$

The optimal CCA solution $\mathbf{h} = [\mathbf{h}_1^T, \mathbf{h}_2^T]^T$ is found as the eigenvector corresponding to the largest eigenvalue of this generalized eigenvalue problem (GEV). The encountered solution for the maximal eigenvalue corresponds to the *first canonical correlation* ρ_1. The subsequent canonical correlations are found as the ρ-values corresponding to the eigenvalues correspond to the successive eigenvalues of (D.8).

An equivalent GEV of the form $\mathbf{Rh} = \lambda' \mathbf{Dh}$ can be found as

$$\frac{1}{2} \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{R}_{21} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} = \lambda' \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix}, \tag{D.9}$$

where $\lambda' = \lambda + 1/2$.

D.2 Generalization to Several Data Sets

In 1971, Kettenring described a maximum-variance (MAXVAR) generalization of the canonical correlation analysis framework [Kettenring, 1971] to several data sets. CCA was implemented here through a two-layer neural network, where the first layer performs a constrained projection of the input data, and the second layer performs an extraction of the principal components. A summary of different alternative generalizations of CCA can be found in [Vía et al., 2007b]. In particular, Vía et al. proposed a generalization that is equivalent to MAXVAR CCA and allows for an elegant extension of the GEV (D.9), as

$$\frac{1}{M}\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \cdots & \mathbf{R}_{1M} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \cdots & \mathbf{R}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{M1} & \mathbf{R}_{M2} & \cdots & \mathbf{R}_{MM} \end{bmatrix}\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_M \end{bmatrix} = \lambda'' \begin{bmatrix} \mathbf{R}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}_{MM} \end{bmatrix}\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \vdots \\ \mathbf{h}_M \end{bmatrix}. \quad (D.10)$$

D.3 Kernel Canonical Correlation Analysis

In order to retrieve nonlinear relationships between data sets, the problem of CCA can be transformed into feature space. First of all, this requires transforming the data points as $\tilde{\mathbf{x}}_i[n] = \phi(\mathbf{x}_i[n])$, for $i = 1, 2$ and $n = 1, \dots, N$. The data matrices $\tilde{\mathbf{X}}_i$ are constructed by stacking these points as rows. After following a similar reasoning as in section D.1, we obtain the transformed version of the GEV (D.8),

$$\frac{1}{2}\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{X}}_1^T\tilde{\mathbf{X}}_2 \\ \tilde{\mathbf{X}}_2^T\tilde{\mathbf{X}}_1 & \mathbf{0} \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{h}}_1 \\ \tilde{\mathbf{h}}_2 \end{bmatrix} = \lambda \begin{bmatrix} \tilde{\mathbf{X}}_1^T\tilde{\mathbf{X}}_1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{X}}_2^T\tilde{\mathbf{X}}_2 \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{h}}_1 \\ \tilde{\mathbf{h}}_2 \end{bmatrix}. \quad (D.11)$$

By representing the solutions as expansions in terms of the data, $\tilde{\mathbf{h}}_1 = \tilde{\mathbf{X}}_1^T\boldsymbol{\alpha}_1$ and $\tilde{\mathbf{h}}_2 = \tilde{\mathbf{X}}_2^T\boldsymbol{\alpha}_2$, and pre-multiplying (D.11) with a suitable block-diagonal data matrix, we obtain

$$\frac{1}{2}\begin{bmatrix} \tilde{\mathbf{X}}_1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{X}}_2 \end{bmatrix}\begin{bmatrix} \mathbf{0} & \tilde{\mathbf{X}}_1^T\tilde{\mathbf{X}}_2 \\ \tilde{\mathbf{X}}_2^T\tilde{\mathbf{X}}_1 & \mathbf{0} \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{X}}_1^T\boldsymbol{\alpha}_1 \\ \tilde{\mathbf{X}}_2^T\boldsymbol{\alpha}_2 \end{bmatrix} = \lambda \begin{bmatrix} \tilde{\mathbf{X}}_1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{X}}_2 \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{X}}_1^T\tilde{\mathbf{X}}_1 & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{X}}_2^T\tilde{\mathbf{X}}_2 \end{bmatrix}\begin{bmatrix} \tilde{\mathbf{X}}_1^T\boldsymbol{\alpha}_1 \\ \tilde{\mathbf{X}}_2^T\boldsymbol{\alpha}_2 \end{bmatrix}. \quad (D.12)$$

At this point the kernel matrices $\mathbf{K}_i = \tilde{\mathbf{X}}_i\tilde{\mathbf{X}}_i^T$ can be introduced, for $i, j = 1, 2$, which are calculated as $K_i(m, n) = \kappa(\mathbf{x}_i[m], \mathbf{x}_i[n])$. This reduces the GEV to

$$\frac{1}{2}\begin{bmatrix} \mathbf{0} & \mathbf{K}_1\mathbf{K}_2 \\ \mathbf{K}_2\mathbf{K}_1 & \mathbf{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{K}_1^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2^2 \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix}. \quad (D.13)$$

However, for “rich enough” kernel functions κ such as the Gaussian kernel, the modeling capacity of the corresponding RKHS is so high that it allows to find perfect correlation between any two data sets. This *overfitting* can be avoided by regularizing the complexity of the solutions in feature space through ridge regression, which converts the energy constraints into

$$\begin{cases} \boldsymbol{\alpha}_1^T\mathbf{K}_1^2\boldsymbol{\alpha}_1 + c\boldsymbol{\alpha}_1^T\mathbf{K}_1\boldsymbol{\alpha}_1 = 1 \\ \boldsymbol{\alpha}_2^T\mathbf{K}_2^2\boldsymbol{\alpha}_2 + c\boldsymbol{\alpha}_2^T\mathbf{K}_2\boldsymbol{\alpha}_2 = 1. \end{cases} \quad (D.14)$$

The corresponding GEV is obtained as

$$\frac{1}{2}\begin{bmatrix} \mathbf{0} & \mathbf{K}_1\mathbf{K}_2 \\ \mathbf{K}_2\mathbf{K}_1 & \mathbf{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{K}_1(\mathbf{K}_1 + c\mathbf{I}) & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2(\mathbf{K}_2 + c\mathbf{I}) \end{bmatrix}\begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix}. \quad (D.15)$$

Similar to the linear case, the solutions of this GEV can also be obtained by solving

$$\frac{1}{2}\begin{bmatrix}\mathbf{0} & \mathbf{K}_2\\ \mathbf{K}_1 & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix} = \lambda\begin{bmatrix}\mathbf{K}_1 + c\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2 + c\mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}, \tag{D.16}$$

which requires slightly less operations to construct. An interesting GEV can also be obtained by adding $\frac{1}{2}\begin{bmatrix}\mathbf{K}_1^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2^2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}$ to both sides of equation (D.15), which yields

$$\begin{aligned}\frac{1}{2}\begin{bmatrix}\mathbf{K}_1^2 & \mathbf{K}_1\mathbf{K}_2\\ \mathbf{K}_2\mathbf{K}_1 & \mathbf{K}_2^2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix} &= \begin{bmatrix}\frac{1}{2}\mathbf{K}_1^2 + \lambda\mathbf{K}_1^2 + \lambda c\mathbf{K}_1 & \mathbf{0}\\ \mathbf{0} & \frac{1}{2}\mathbf{K}_2^2 + \lambda\mathbf{K}_2^2 + \lambda c\mathbf{K}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}\\ &= (\lambda + \frac{1}{2})\begin{bmatrix}\mathbf{K}_1^2 + \frac{\lambda c}{\lambda+\frac{1}{2}}\mathbf{K}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2^2 + \frac{\lambda c}{\lambda+\frac{1}{2}}\mathbf{K}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}.\end{aligned}$$

By substituting $\lambda' = \lambda + \frac{1}{2}$ and introducing $c' = \frac{\lambda c}{\lambda+\frac{1}{2}}$, this problem reduces to

$$\frac{1}{2}\begin{bmatrix}\mathbf{K}_1^2 & \mathbf{K}_1\mathbf{K}_2\\ \mathbf{K}_2\mathbf{K}_1 & \mathbf{K}_2^2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix} = \lambda'\begin{bmatrix}\mathbf{K}_1(\mathbf{K}_1 + c'\mathbf{I}) & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2(\mathbf{K}_2 + c'\mathbf{I})\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}, \tag{D.17}$$

whose solutions are also found by solving

$$\frac{1}{2}\begin{bmatrix}\mathbf{K}_1 & \mathbf{K}_2\\ \mathbf{K}_1 & \mathbf{K}_2\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix} = \lambda'\begin{bmatrix}\mathbf{K}_1 + c'\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{K}_2 + c'\mathbf{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_1\\ \boldsymbol{\alpha}_2\end{bmatrix}. \tag{D.18}$$

Therefore, by introducing a regularization term c' only in the right hand side of (D.18), we are imposing a regularization term $c = \frac{c'}{1-\frac{\lambda'}{2}}$ onto the norm of the CCA solution in feature space. Finally, note that (D.17) can be extended for multi-variate problems in a straightforward fashion thanks to its symmetry, similar to what was done for (D.10).

Appendix E

Deduction of Spectral Clustering from Graph Theory

E.1 Mincut and Normalized Cut

Although several algorithms for spectral clustering exist, they are all based on an eigenvector problem posed on the similarity matrix. This basic algorithm can be obtained in few different ways, for instance by describing the partitioning problem as a graph cut problem or a random walks optimization problem [von Luxburg, 2006]. In this appendix we will deduce spectral clustering from graph theory, following the analysis presented in [Alzate and Suykens, 2006].

Given an undirected graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ is the set of edges, the problem of graph bipartitioning consists in separating the graph into two sets $\mathcal{A}$, $\mathcal{B}$ by eliminating edges connecting the two sets. The sets should be disjoint such that: $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$. The total weight of the edges that have to be eliminated is called the *cut*:

$$cut(\mathcal{A}, \mathcal{B}) = \sum_{\mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}} w(\mathbf{a}, \mathbf{b}), \tag{E.1}$$

where $w(\mathbf{a}, \mathbf{b})$ is the weight between nodes $\mathbf{a} \in \mathcal{A}$ and $\mathbf{b} \in \mathcal{B}$. In general, we will denote the weight between the nodes with indices i and j as w_{ij}.

E.1.1 Mincut

One of the most elementary criteria to perform graph bipartitioning is the *mincut* criterion [Fiedler, 1975], which is formulated as follows:

$$\min_{q} J_{mincut} = cut(\mathcal{A}, \mathcal{B}) = \frac{1}{2} \sum_{i,j} w_{ij} (q_i - q_j)^2, \tag{E.2}$$

where q_i is a cluster membership indicator:

$$q_i = \begin{cases} 1, & \text{if } i \in \mathcal{A} \\ -1, & \text{if } i \in \mathcal{B}. \end{cases} \tag{E.3}$$

By defining $\mathbf{D} = \text{diag}(d_1, \ldots, d_N)$, where $d_i = \sum_j w_{ij}$ is the sum of the weights for all edges that connect to node i, and $\mathbf{W}$ as the symmetric kernel matrix with ij-th entry equal to w_{ij}, we can write the minimization problem (E.2) as

$$\begin{aligned} \min_{q} J_{mincut} &= \mathbf{q}^T(\mathbf{D} - \mathbf{W})\mathbf{q} \\ \text{s.t.} \quad & \mathbf{q} \in \{-1, +1\}^N. \end{aligned} \tag{E.4}$$

Here, $\mathbf{q}$ is a vector that contains all q_i values, and the matrix $\mathbf{D} - \mathbf{W}$ is the *Laplacian* of the graph. We will denote this matrix as $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

The original mincut problem is NP-hard due to the constraint on $\mathbf{q}$ [Chung, 1997]. However, a suboptimal solution can be found by relaxing this constraint and letting $\mathbf{q}$ take real values. The solution to the relaxed problem is found by solving the eigenvalue problem

$$\mathbf{L}\bar{\mathbf{q}} = \lambda\bar{\mathbf{q}}, \tag{E.5}$$

with the constraint $\bar{\mathbf{q}}^T\bar{\mathbf{q}} = 1$. The suboptimal solution $\bar{\mathbf{q}}$ is the eigenvector corresponding to the second smallest eigenvalue (also called the Fiedler vector). The cluster membership indicator q_i is obtained by binarizing $\bar{q}_i$ using a suitable threshold θ:

$$q_i = sign(\bar{q}_i - \theta), \quad i = 1, \ldots, N. \tag{E.6}$$

Due to the fact that there are no restrictions in (E.2) related to the cluster size, the mincut criterion tends to separate small sets of points. In the following we discuss a criterion that imposes a simple size restriction on the clusters.

E.1.2 Normalized cut

In [Shi and Malik, 2000], a graph bipartitioning criterion was introduced that normalizes the cost of the cut relative to total weight of each cluster, called *normalized cut*. It is defined as:

$$\min_{\mathbf{q}} J_{ncut} = \frac{cut(\mathcal{A}, \mathcal{B})}{d_{\mathcal{A}}} + \frac{cut(\mathcal{A}, \mathcal{B})}{d_{\mathcal{B}}}, \tag{E.7}$$

where $d_{\mathcal{A}} = \sum_{i \in \mathcal{A}} d_i$. Due to this normalization, the normalized-cut criterion penalizes small sets or isolated points.

Once again, the solution to this problem is NP-hard, but if $\mathbf{q}$ can take real values then J_{ncut} is minimized by the eigenvector corresponding to the second smallest eigenvalue of the following generalized eigenvalue (GEV) problem:

$$\mathbf{L}\bar{\mathbf{q}} = \lambda\mathbf{D}\bar{\mathbf{q}}. \tag{E.8}$$

E.2 NJW Algorithm

By pre-multiplying the normalized-cut criterion (E.8) by $\mathbf{D}^{-\frac{1}{2}}$, we obtain

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\bar{\mathbf{q}} = \lambda \mathbf{D}^{-\frac{1}{2}}\bar{\mathbf{q}}. \tag{E.9}$$

Defining $\hat{\mathbf{q}} = \mathbf{D}^{\frac{1}{2}}\bar{\mathbf{q}}$ leads to

$$\begin{aligned}
\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{q}} &= \lambda\hat{\mathbf{q}} & \text{(E.10)}\\
(\mathbf{I}-\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}})\hat{\mathbf{q}} &= \lambda\hat{\mathbf{q}} & \text{(E.11)}\\
\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{q}} &= \hat{\lambda}\hat{\mathbf{q}}, & \text{(E.12)}
\end{aligned}$$

where $\tilde{\lambda} = 1-\lambda$. By combining (E.12) with the norm constraint on $\hat{\mathbf{q}}$, we obtain the eigenvalue problem used in the Ng-Jordan-Weiss (NJW) algorithm [Ng et al., 2001]:

$$\mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{q}} = \tilde{\lambda}\hat{\mathbf{q}}. \tag{E.13}$$

E.3 Spectral Clustering as Weighted Kernel PCA

By defining an $N \times N$ weight matrix $\mathbf{V}$, the standard procedure of kernel PCA can be extended to into a *weighted kernel PCA* algorithm [Alzate and Suykens, 2006] that allows to lower the influence of outliers or emphasize certain points. Formally, the weighted kernel PCA problem is defined as

$$\mathbf{V}\mathbf{W}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}, \tag{E.14}$$

where $\mathbf{W}$ is the kernel matrix. By choosing $\mathbf{V} = \mathbf{D}^{-1}$ and $\boldsymbol{\alpha} = \mathbf{D}^{-1/2}\mathbf{q}$, we obtain the standard formulation of the eigenvalue problem (E.13) from the NJW algorithm.

Appendix F
Calculation of the Connectivity Kernel

The Dijkstra shortest path algorithm [Dijkstra, 1959] is a graph search algorithm that calculates the shortest path between one vertex of a graph and every other vertex. With a slight modification, the Dijkstra algorithm can be used to calculate the *effective distance* needed for the connectivity kernel.

By the *effective distance* $\bar{d}_{i,j}$ between two vertices i and j we denote the length of the *weakest link* of the *best path* between these vertices. In this definition, the *weakest link* refers to the longest edge, and the best path is the path whose longest edge is the shortest among all paths. Whereas the original Dijkstra algorithm calculates the total distance between two vertices, the effective distance only requires the maximum edge length. This can be obtained by simply replacing the summation of distances with a max operation in the original algorithm. The resulting algorithm is found below. It returns the effective distances between one vertex of a graph and every other vertex. Finally, the connectivity kernel can be obtained by calculating the Gaussian kernel using the effective distance.

Algorithm F.1 Modified Dijkstra Algorithm for Connectivity Kernel Calculation.

```
Given the nodes $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X}$ and the distances $d_{i,j}$ between all node pairs.
for all $i$ do
  Initialize effective distances $\bar{d}_{i,i} = 0$ and $\bar{d}_{i,j} = \infty$, for all $j \neq i$.
  Set node $i$ as current node $c$.
  Mark al nodes as “unvisited”.
  repeat
    for all unvisited nodes $n$ neighboring current node do
      Effective distance on current path: $\bar{d}^p_{i,n} = \max(\bar{d}_{i,c}, d_{c,n})$.
      Update effective distance: $\bar{d}_{i,n} = \min(\bar{d}_{i,n}, \bar{d}^p_{i,n})$.
    end for
    Mark current node as visited.
  until all nodes are visited.
end for
Output the connectivity kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\bar{d}^2_{i,j}/2\sigma^2)$.
```

Appendix G Publications

The detailed contributions of this work are mentioned at the end of each corresponding chapter. Here we provide the entire listing which includes some additional publications that were not included in the body of this thesis.

G.1 Book Chapter

1. S. Van Vaerenbergh and I. Santamaría. *Intelligent Systems: Techniques and Applications*, chapter A Spectral Clustering Approach for Blind Decoding of MIMO Transmissions over Time-Correlated Fading Channels, pages 351–377. Shaker Publishing, Maastricht, The Netherlands, 2008.

G.2 International Journals

2. S. Van Vaerenbergh, I. Santamaría, P. E. Barbano, U. Ozertem, and D. Erdogmus. "Path-based clustering for decoding time-varying MIMO channels". *IEEE Transactions on Signal Processing, in preparation*, 2010.

3. S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Blind equalization of post-nonlinear SIMO systems". *Submitted to IEEE Transactions on Signal Processing*, volume 58, pages 1–8, 2010.

4. S. Van Vaerenbergh and I. Santamaría. "Alternating kernel least-squares for supervised identification of Hammerstein systems". *Submitted to IEEE Signal Processing Letters*, volume 1, page 4, 2010.

5. L. Sánchez-Giraldo, S. Seth, S. Van Vaerenbergh, and J. C. Príncipe. "Efficient computation for information theoretic learning". *Submitted to IEEE Signal Processing Letters*, volume 1, 4 pages, 2010.

6. S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Adaptive kernel canonical correlation analysis algorithms for nonparametric identification of Wiener and Hammerstein systems". *EURASIP Journal on Advances in Signal Processing*, volume 1, Article ID 875351, 13 pages, 2008.

7. S. Van Vaerenbergh, J. Vía, and I. Santamaría. “Nonlinear system identification using a new sliding-window kernel RLS algorithm”. *Journal of Communications*, volume 2, no. 3, pages 1–8, 2007.

8. S. Van Vaerenbergh and I. Santamaría. “A spectral clustering approach to underdetermined postnonlinear blind source separation of sparse sources”. *IEEE Transactions on Neural Networks*, volume 17, no. 3, pages 811–814, 2006.

G.3 International Conferences

9. S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe. “Fixed-budget kernel recursive least-squares”. *Submitted to the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

10. S. Van Vaerenbergh, I. Santamaría, P. Barbano, U. Ozertem, and D. Erdogmus. “Path-based spectral clustering for decoding fast time-varying MIMO channels”. In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2009)*. Grenoble, France, 2009.

11. S. Van Vaerenbergh, J. Vía, and I. Santamaría. “A kernel canonical correlation analysis algorithm for blind equalization of oversampled Wiener systems”. In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2008)*, pages 20–25. 2008.

12. S. Van Vaerenbergh, E. Estébanez, and I. Santamaría. “A spectral clustering algorithm for decoding fast time-varying BPSK MIMO channels”. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*. Poznań, Poland, 2007.

13. S. Van Vaerenbergh, J. Vía, and I. Santamaría. “Online kernel canonical correlation analysis for supervised equalization of Wiener systems”. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Vancouver, Canada, 2006.

14. S. Van Vaerenbergh, J. Vía, and I. Santamaría. “A sliding-window kernel RLS algorithm and its application to nonlinear channel identification”. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Toulouse, France, 2006.

G.4 National Conferences

15. D. Ramírez, I. Santamaría, J. Vía, and S. Van Vaerenbergh. “An extension of SVM for piecewise regression”. In *Proceedings of the National Symposium of the International Union of Radio Science (URSI 2007)*. Tenerife, Spain, 2007.

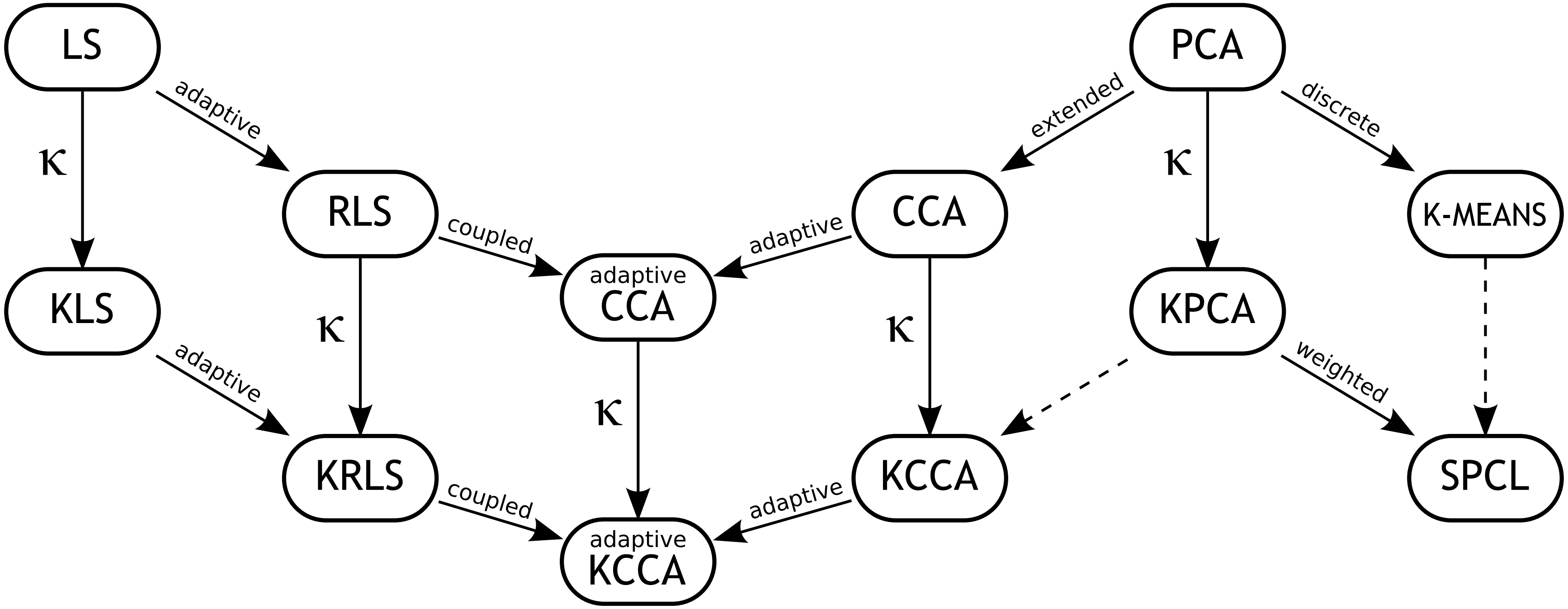


Figure: Relationship of the techniques used in this thesis. The core techniques, least-squares regression (LS) and principal component analysis (PCA), were introduced in chapter 2. The methods on the left-hand side are supervised, while on the right-hand side we find blind techniques. The arrows indicate relationships between the different techniques. Dashed arrows stand for a loose relationship.

List of Figures

List of Algorithms

Bibliography

[Abe and Shigeo, 2007] Abe and Shigeo. “Sparse least squares support vector training in the reduced empirical feature space”. *Pattern Analysis and Applications (PAA)*, volume 10, no. 3, pages 203–214, 2007.

[Adali and Liu, 1997] T. Adali and X. Liu. “Canonical piecewise linear network for nonlinear filtering and its application to blind equalization”. *Signal Processing*, volume 61, no. 2, pages 145–155, 1997.

[Alamouti, 1998] S. Alamouti. “A simple transmit diversity technique for wireless communications”. *IEEE Journal on Selected Areas in Communications*, volume 16, no. 8, pages 1451–1458, 1998.

[Alper, 1965] P. Alper. “A consideration of the discrete volterra series”. *IEEE Transactions on Automatic Control*, volume 10, no. 3, pages 322–327, 1965.

[Alzate and Suykens, 2006] C. Alzate and J. Suykens. “A weighted kernel PCA formulation with out-of-sample extensions for spectral clustering methods”. In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, pages 138–144. IEEE, Vancouver, Canada, 2006.

[Alzate and Suykens, 2008] C. Alzate and J. A. K. Suykens. “Sparse kernel models for spectral clustering using the incomplete cholesky decomposition”. In *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, pages 3556–3563. Hong Kong, China, 2008.

[Aronszajn, 1950] N. Aronszajn. “Theory of reproducing kernels”. *Transactions of the American Mathematical Society*, volume 68, pages 337–404, 1950.

[Aschbacher and Rupp, 2005] E. Aschbacher and M. Rupp. “Robustness analysis of a gradient identification method for a nonlinear Wiener system”. In *Proceedings of the 2005 IEEE/SP 13th Workshop on Statistical Signal Processing (SSP 2005)*. Bordeaux, France, 2005.

[Babaie-Zadeh et al., 2006] M. Babaie-Zadeh, C. Jutten, and A. Mansour. “Sparse ICA via cluster-wise PCA”. *Neurocomputing*, volume 69, no. 13-15, pages 1458–1466, 2006.

[Babaie-Zadeh et al., 2002] M. Babaie-Zadeh, C. Jutten, and K. Nayebi. “A geometric approach for separating post nonlinear mixtures”. In *Proceedings of the*

11th European Signal Processing Conference (EUSIPCO), volume II, pages 11–14. IEEE, Toulouse, France, 2002.

[Bach and Jordan, 2002] F. R. Bach and M. I. Jordan. “Kernel independent component analysis”. *Journal of Machine Learning Research*, volume 3, pages 1–48, 2002.

[Bach and Jordan, 2003] F. R. Bach and M. I. Jordan. “Learning spectral clustering”. Technical report, EECS Department, University of California, Berkeley, 2003.

[Bach and Jordan, 2004] F. R. Bach and M. I. Jordan. “Blind one-microphone speech separation: A spectral learning approach”. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS 2004)*, pages 65–72. Whistler, BC, Canada, 2004.

[Bach and Jordan, 2005] F. R. Bach and M. I. Jordan. “Predictive low-rank decomposition for kernel methods”. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 33–40. Bonn, Germany, 2005.

[Bai, 1998] E.-W. Bai. “An optimal two stage identification algorithm for Hammerstein-Wiener nonlinear systems”. In *Proceedings of the 1998 American Control Conference*, volume 5, pages 2756–2760. 1998.

[Balestrino et al., 2001] A. Balestrino, A. Landi, M. Ould-Zmirli, and L. Sani. “Automatic nonlinear auto-tuning method for Hammerstein modeling of electrical drives”. *IEEE Transactions on Industrial Electronics*, volume 48, no. 3, pages 645–655, 2001.

[Bell and Sejnowski, 1995] A. Bell and T. Sejnowski. “An information-maximization approach to blind separation and blind deconvolution”. *Neural computation*, volume 7, no. 6, pages 1129–1159, 1995.

[Belouchrani and Amin, 1998] A. Belouchrani and M. Amin. “Blind source separation based on time-frequency signal representations”. *IEEE Transactions on Signal Processing*, volume 46, no. 11, pages 2888–2897, 1998.

[Ben-Hur et al., 2008] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. “Support vector machines and kernels for computational biology”. *PLoS Computational Biology*, volume 4, no. 10, page e1000173, 2008.

[Bezdek, 1981] J. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Plenum Press, New York, NY, USA, 1981.

[Bezdek and Hathaway, 2003] J. C. Bezdek and R. J. Hathaway. “Convergence of alternating optimization”. *Neural, Parallel and Scientific Computation*, volume 11, no. 4, pages 351–368, 2003.

[Billings, 1980] S. Billings. “Identification of nonlinear systems: A survey”. *Proceedings of IEE, Part D*, volume 127, pages 272–285, 1980.

[Billings and Fakhouri, 1977] S. Billings and S. Fakhouri. “Identification of nonlinear systems using the wiener model”. *Electronics Letters*, volume 13, no. 17, pages 502–504, 1977.

[Billings and Fakhouri, 1979] S. A. Billings and S. Y. Fakhouri. “Nonlinear system identification using the Hammerstein model”. *International Journal of System Sciences*, volume 10, no. 5, pages 567–578, 1979.

[Billings and Fakhouri, 1982] S. A. Billings and S. Y. Fakhouri. “Identification of systems containing linear dynamic and static nonlinear elements”. *Automatica*, volume 18, pages 15–26, 1982.

[Bofill and Zibulevsky, 2001] P. Bofill and M. Zibulevsky. “Underdetermined blind source separation using sparse representations”. *Signal Processing*, volume 81(11), pages 2353–2362, 2001.

[Boyd and Chua, 1985] S. Boyd and L. Chua. “Fading memory and the problem of approximating nonlinear operators with volterra series”. *IEEE Transactions on Circuits and Systems*, volume 32, no. 11, pages 1150–1161, 1985.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[Brilliant, 1958] M. B. Brilliant. “Theory of the analysis of nonlinear systems”. RLE Technical Report 345, M.I.T., 1958.

[Bruls et al., 1999] J. Bruls, C. Chou, B. Haverkamp, and M. Verhaegen. “Linear and nonlinear system identification using separable least-squares”. *European Journal of Control Engineering Practice*, volume 5, no. 1, pages 116–128, 1999.

[Cardoso, 1998] J. Cardoso. “Blind signal separation: statistical principles”. *Proceedings of the IEEE. Special issue on blind identification and estimation*, volume 9, no. 10, pages 2009–2025, 1998.

[Cardoso and Souloumiac, 1993] J.-F. Cardoso and A. Souloumiac. “Blind beamforming for non Gaussian signals”. *IEE Proceedings-F*, volume 140, no. 6, pages 362–370, 1993.

[Cawley and Talbot, 2002] G. C. Cawley and N. L. C. Talbot. “Reduced rank kernel ridge regression”. *Neural Processing Letters*, volume 16, no. 3, pages 293–302, 2002.

[Chen and Chang, 1996] C.-T. Chen and W.-D. Chang. “A feedforward neural network with function shape autotuning”. *Neural Networks*, volume 9, no. 4, pages 627 – 641, 1996.

[Chen, 1995] H.-W. Chen. “Modeling and identification of parallel nonlinear systems: structural classification and parameter estimation methods”. *Proceedings of the IEEE*, volume 83, no. 1, pages 39–66, 1995.

[Chen et al., 1989] S. Chen, S. A. Billings, and W. Luo. “Orthogonal least squares methods and their application to non-linear system identification”. *International Journal of Control*, volume 50, pages 1873–1896, 1989.

[Chen et al., 1995] S. Chen, S. McLaughlin, P. Grant, and B. Mulgrew. “Multi-stage blind clustering equalizer”. *IEEE Transactions on Communications*, volume 43, no. 2, pages 701–705, 1995.

[Chen et al., 1993] S. Chen, B. Mulgrew, and P. Grant. “A clustering technique for digital communications channel equalization using radial basis function networks”. *IEEE Transactions on Neural Networks*, volume 4, no. 4, pages 570–590, 1993.

[Choi et al., 2005] J. Choi, H. Yu, and Y. Lee. “Adaptive MIMO decision feedback equalization for receivers with time-varying channels”. *IEEE Transactions on Signal Processing*, volume 53, no. 11, pages 4295–4303, 2005.

[Chung, 1997] F. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[Cid-Sueiro et al., 1994] J. Cid-Sueiro, A. Artés-Rodríguez, and A. R. Figueiras-Vidal. “Recurrent radial basis function networks for optimal symbol-by-symbol equalization”. *Signal Processing*, volume 40, no. 1, pages 53–63, 1994.

[Comon, 1994] P. Comon. “Independent component analysis - a new concept?”. *Signal Processing*, volume 36, pages 287–314, 1994.

[Comon et al., 1991] P. Comon, C. Jutten, and J. Herault. “Blind separation of sources, part II: Problem statement”. *Signal Processing*, volume 24, pages 11–21, 1991.

[Cousseau et al., 2007] J. Cousseau, J. Figueroa, S. Werner, and T. Laakso. “Efficient nonlinear Wiener model identification using a complex-valued simplicial canonical piecewise linear filter”. *IEEE Transactions on Signal Processing*, volume 55, no. 5, pages 1780–1792, 2007.

[Cover, 1965] T. M. Cover. “Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition”. *IEEE Transactions on Electronic Computers*, volume EC-14, no. 3, pages 326–334, 1965.

[Cristianini et al., 2000] N. Cristianini, J. Shawe-Taylor, and J. Kandola. “Spectral kernel methods for clustering”. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems (NIPS 2002)*, pages 649–656. Whistler, BC, Canada, 2000.

[Cuturi et al., 2007] M. Cuturi, J. P. Vert, O. Birkenes, and T. Matsui. “A kernel for time series based on global alignments”. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 413–416. Honolulu, Hawai’i, USA, 2007.

[de Kruif and de Vries, 2003] B. J. de Kruif and T. J. de Vries. “Pruning error minimization in least squares support vector machines”. *IEEE Transactions on Neural Networks*, volume 14, no. 3, pages 696–702, 2003.

[Dekel et al., 2008] O. Dekel, S. Shalev-Shwartz, and Y. Singer. “The forgetron: A kernel-based perceptron on a budget”. *SIAM Journal on Computing*, volume 37, no. 5, pages 1342–1372, 2008.

[Dempsey and Westwick, 2004] E. Dempsey and D. Westwick. “Identification of Hammerstein models with cubic spline nonlinearities”. *IEEE Transactions on Biomedical Engineering*, volume 51, no. 2, pages 237–245, 2004.

[Dempster et al., 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. *Journal of the Royal Statistical Society, Series B*, volume 39, no. 1, pages 1–38, 1977.

[Desobry and Févotte, 2006] F. Desobry and C. Févotte. “Kernel PCA based estimation of the mixing matrix in linear instantaneous mixtures of sparse sources”. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 5. Toulouse, France, 2006.

[Dhillon et al., 2004] I. S. Dhillon, Y. Guan, and B. Kulis. “Kernel k-means: spectral clustering and normalized cuts”. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD*, pages 551–556. ACM Press, New York, NY, USA, 2004.

[Diamantaras, 2006] K. I. Diamantaras. “A clustering approach for the blind separation of multiple finite alphabet sequences from a single linear mixture”. *Signal Processing*, volume 86, no. 4, pages 877–891, 2006.

[Diamantaras and Kung, 1996] K. I. Diamantaras and S. Y. Kung. *Principal component neural networks: theory and applications*. John Wiley & Sons, Inc., New York, NY, USA, 1996.

[Diamantaras and Papadimitriou, 2006] K. I. Diamantaras and T. Papadimitriou. “Blind deconvolution of multi-input single-output systems with binary sources”. *IEEE Transactions on Signal Processing*, volume 54, no. 10, pages 3270–3731, 2006.

[Dijkstra, 1959] E. W. Dijkstra. “A note on two problems in connexion with graphs”. *Numerische Mathematik*, volume 1, no. 1, pages 269–271, 1959.

[Ding and He, 2004] C. Ding and X. He. “K-means clustering via principal component analysis”. In *Proceedings of the 21st international conference on Machine learning (ICML)*, pages 225–232. Alberta, Canada, 2004.

[Ding and Li, 2001] Z. Ding and Y. Li. *Blind Equalization and Identification*. Mark Dekker, New York, 2001.

[Engel et al., 2004] Y. Engel, S. Mannor, and R. Meir. “The kernel recursive least-squares algorithm”. *IEEE Transactions on Signal Processing*, volume 52, no. 8, pages 2275–2285, 2004.

[Erdogmus et al., 2001a] D. Erdogmus, D. Rende, J. C. Príncipe, and T. F. Wong. “Nonlinear channel equalization using multilayer perceptrons with information-theoric criterion”. In *Proceedings of the XI IEEE Workshop on Neural Networks and Signal Processing (NNSP)*, pages 401–451. Falmouth, MA, USA, 2001.

[Erdogmus et al., 2001b] D. Erdogmus, L. Vielva, and J. C. Príncipe. “Nonparametric estimation and tracking of the mixing matrix for underdetermined blind source separation”. In *Proceedings of the 3rd International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2001)*, pages 189–194. San Diego, California, USA, 2001.

[Evgeniou et al., 2000] T. Evgeniou, M. Pontil, and T. Poggio. “Regularization networks and support vector machines”. In *Advances in Computational Mathematics*, volume 1, pages 1–50. MIT Press, 2000.

[Feher, 1983] K. Feher. *Digital Communications: Satellite/Earth Station Engineering*. Englewood Cliffs, N.J.: Prentice-Hall, 1983.

[Fiedler, 1975] M. Fiedler. “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory”. *Czechoslovak Mathematical Journal*, volume 25, pages 619–633, 1975.

[Fine and Scheinberg, 2001] S. Fine and K. Scheinberg. “Efficient SVM training using low-rank kernel representations”. *Journal of Machine Learning Research*, volume 2, pages 243–264, 2001.

[Fischer et al., 2003] B. Fischer, V. Roth, and J. M. Buhmann. “Clustering with the connectivity kernel”. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS 2003)*. Whistler, BC, Canada, 2003.

[Foschini et al., 1999] G. J. Foschini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky. “Simplified processing for high spectral efficiency wireless communication employing multi-element arrays”. *IEEE Journal on Selected Areas in Communications*, volume 17, no. 9, pages 1841–1852, 1999.

[Franz and Schölkopf, 2006] M. O. Franz and B. Schölkopf. “A unifying view of Wiener and Volterra theory and polynomial kernel regression”. *Neural Computation*, volume 18, no. 12, pages 3097–3118, 2006.

[Fréchet, 1910] M. Fréchet. “Sur les fonctionelles continues”. *Annales Scientifiques de L’École Normale Supérieure*, volume 27, pages 193–216, 1910.

[Frieß and Harrison, 1999] T.-T. Frieß and R. F. Harrison. “A kernel-based adaline”. In *Proceedings of the 7th European Symposium on Artificial Neural Networks (ESANN 1999)*, pages 245–250. Bruges, Belgium, 1999.

[Gao et al., 2002] C. Gao, D. Lao, and A. Haimovich. “Bit error probability for space-time block code with coherent and differential detection”. *Proceedings of the 56th Vehicular Technology Conference (VTC 2002-Fall)*, volume 1, pages 410–414, 2002.

[Gardiner, 1973] A. B. Gardiner. “Identification of processes containing single-valued nonlinearities”. *International Journal of Control*, volume 18, no. 5, pages 1029–1039, 1973.

[Giannakis and Serpedin, 1997] G. Giannakis and E. Serpedin. “Linear multichannel blind equalizers of nonlinear FIR volterra channels”. *IEEE Transactions on Signal Processing*, volume 45, no. 1, pages 67–81, 1997.

[Giannakis and Serpedin, 2001] G. Giannakis and E. Serpedin. “A bibliography on nonlinear system identification”. *Signal Processing*, volume 81, no. 3, pages 533–580, 2001.

[Girolami, 2002] M. Girolami. “Orthogonal series density estimation and the kernel eigenvalue problem”. *Neural Computation*, volume 14, no. 3, pages 669–688, 2002.

[Goethals et al., 2005] I. Goethals, K. Pelckmans, J. A. K. Suykens, and B. De Moor. “Identification of MIMO Hammerstein models using least squares support vector machines”. *Automatica*, volume 41, no. 7, pages 1263 – 1272, 2005.

[Golub and Van Loan, 1996] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 1996.

[Gómez and Baeyens, 2007] J. Gómez and E. Baeyens. “Subspace-based blind identification of IIR Wiener systems”. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*. Poznań, Poland, 2007.

[Greblicki, 1997] W. Greblicki. “Nonparametric approach to Wiener system identification”. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, volume 44, no. 6, pages 538–545, 1997.

[Greblicki, 2004] W. Greblicki. “Nonlinearity recovering in Wiener system driven with correlated signal”. *IEEE Transactions on Automatic Control*, volume 49, no. 10, pages 1805–1812, 2004.

[Hagenblad, 1999] A. Hagenblad. *Aspects of the Identification of Wiener Models*. Licentiate thesis no. 793, Department of Electrical Engineering, Linköping University, Linköping, Sweden, 1999.

[Hager, 1989] W. W. Hager. “Updating the inverse of a matrix”. *SIAM Review*, volume 31, no. 2, pages 221–239, 1989.

[Hardoon et al., 2003] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. “Canonical correlation analysis: An overview with application to learning methods”. Technical Report CSD-TR-03-02, Royal Holloway University of London, 2003.

[Hartigan, 1975] J. Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc. New York, NY, USA, 1975.

[Hassibi et al., 1993] B. Hassibi, D. G. Stork, and G. J. Wolff. “Optimal brain surgeon and general network pruning”. In *IEEE International Conference on Neural Networks*, volume 1, pages 293–299. 1993.

[Haykin, 1999] S. Haykin. *Neural Networks: A Comprehensive Foundation, 2nd. Ed.* Prentice Hall, Englewood Cliffs, NJ, USA, 1999.

[Haykin, 2001] S. Haykin. *Adaptive Filter Theory (4th Edition)*. Prentice Hall, 2001.

[Hoegaerts et al., 2007] L. Hoegaerts, L. De Lathauwer, I. Goethals, J. Suykens, J. Vandewalle, and B. De Moor. “Efficiently updating and tracking the dominant kernel principal components”. *Neural Networks*, volume 20, no. 2, pages 220–229, 2007.

[Hoegaerts et al., 2004] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. De Moor. “A comparison of pruning algorithms for sparse least squares support vector machines”. *Lecture Notes in Computer Science*, pages 1247–1253, 2004.

[Hornik et al., 1990] K. Hornik, M. Stinchcombe, and H. White. “Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks”. *Neural Networks*, volume 3, no. 5, pages 551–560, 1990.

[Hotelling, 1936] H. Hotelling. “Relations between two sets of variates”. *Biometrika*, volume 28, pages 321–377, 1936.

[Huang et al., 2005] G.-B. Huang, P. Saratchandran, and N. Sundararajan. “A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation”. *IEEE Transactions on Neural Networks*, volume 16, no. 1, pages 57–67, 2005.

[Hunter and Korenberg, 1986] I. W. Hunter and M. J. Korenberg. “The identification of nonlinear biological systems: Wiener and Hammerstein cascade models”. *Biological Cybernetics*, volume 55, no. 2, pages 135–144, 1986.

[Hyvärinen et al., 2001] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley Interscience, 2001.

[Hyvärinen and Oja, 1997] A. Hyvärinen and E. Oja. "A fast fixed-point algorithm for independent component analysis". *Neural Computation*, volume 9, no. 7, pages 1483–1492, 1997.

[Jenssen et al., 2004] R. Jenssen, T. Eltoft, and J. C. Príncipe. "Information theoretic spectral clustering". In *Proceedings of the 2004 International Joint Conference on Neural Networks (IJCNN)*, pages 111–116. Budapest, Hungary, 2004.

[Joho et al., 2000] M. Joho, H. Mathis, and R. Lambert. "Overdetermined blind source separation: Using more sensors than source signals in a noisy mixture". In *Proceedings of the 2nd International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2000)*, pages 81–86. Helsinki, Finland, 2000.

[Jolliffe, 2002] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2002.

[Julian, 2009] B. Julian. "Modifications to the sliding-window kernel RLS algorithm for time-varying nonlinear systems: Online resizing of the kernel matrix". In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Taipei, Taiwan, 2009.

[Kalman, 1960] R. E. Kalman. "A new approach to linear filtering and prediction problems". *Transactions of the ASME – Journal of Basic Engineering*, volume 82 (Series D), pages 35–45, 1960.

[Kalouptsidis and Theodoridis, 1993] N. Kalouptsidis and S. Theodoridis. *Adaptive system identification and signal processing algorithms*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1993.

[Kannan et al., 2004] R. Kannan, S. Vempala, and A. Vetta. "On clusterings: Good, bad and spectral". *Journal of the ACM (JACM)*, volume 51, no. 3, pages 497–515, 2004.

[Karami and Shiva, 2006] E. Karami and M. Shiva. "Decision-directed recursive least squares MIMO channels tracking". *EURASIP Journal on Wireless Communications and Networking*, volume 2006, Article ID 43275, 10 pages, 2006.

[Kawakami Harrop Galvão et al., 2007] R. Kawakami Harrop Galvão, S. Hadjiloucas, A. Izhac, V. Becerra, and J. Bowen. "Wiener-system subspace identification for mobile wireless mm-wave networks". *IEEE Transactions on Vehicular Technology*, volume 56, no. 4, pages 1935–1948, 2007.

[Kechriotis et al., 1994] G. Kechriotis, E. Zarvas, and E. S. Manolakos. "Using recurrent neural networks for adaptive communication channel equalization". *IEEE Transactions on Neural Networks*, volume 5, pages 267–278, 1994.

[Kekatos et al., 2007] V. Kekatos, A. Rontogiannis, and K. Berberidis. "Cholesky factorization-based adaptive BLAST DFE for wideband MIMO channels".

EURASIP Journal on Advances in Signal Processing, volume 2007, Article ID 45789, 11 pages, 2007.

[Kettenring, 1971] J. R. Kettenring. “Canonical analysis of several sets of variables”. *Biometrika*, volume 58, no. 3, pages 433–451, 1971.

[Kimeldorf and Wahba, 1971] G. S. Kimeldorf and G. Wahba. “Some results on Tchebycheffian spline functions”. *Journal of Mathematical Analysis and Applications*, volume 33, no. 1, pages 82–95, 1971.

[Kivinen et al., 2004] J. Kivinen, A. Smola, and R. Williamson. “Online learning with kernels”. *IEEE Transactions on Signal Processing*, volume 52, no. 8, pages 2165–2176, 2004.

[Kontorovich et al., 2008] L. A. Kontorovich, C. Cortes, and M. Mohri. “Kernel methods for learning languages”. *Theoretical Computer Science*, volume 405, no. 3, pages 223–236, 2008.

[Kopsinis and Theodoridis, 2003] Y. Kopsinis and S. Theodoridis. “A novel cluster based MLSE equalizer for M-PAM signaling schemes”. *Signal Processing*, volume 83, no. 9, pages 1905–1918, 2003.

[Korenberg, 1989] M. J. Korenberg. “A robust orthogonal algorithm for system identification and time-series analysis”. *Biological Cybernetics*, volume 60, no. 4, pages 267–276, 1989.

[Kulis et al., 2006] B. Kulis, M. Sustik, and I. Dhillon. “Learning low-rank kernel matrices”. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 505–512. Pittsburgh, PA, USA, 2006.

[Kwok and Tsang, 2004] J. T. Y. Kwok and I. W. H. Tsang. “The pre-image problem in kernel methods”. *IEEE Transactions on Neural Networks*, volume 15, no. 6, pages 1517–1525, 2004.

[Lai and Fyfe, 2000] P. L. Lai and C. Fyfe. “Kernel and nonlinear canonical correlation analysis”. *International Journal of Neural Systems*, volume 10, no. 5, pages 365–377, 2000.

[Le Cun et al., 1989] Y. Le Cun, J. S. Denker, and S. A. Solla. “Optimal brain damage”. In *Proceedings of the 2nd Annual Conference on Neural Information Processing Systems (NIPS 1989)*, pages 598–605. Denver, CO, USA, 1989.

[Lee et al., 1999a] J. Lee, C. Beach, and N. Tepedelenlioglu. “A practical radial basis function equalizer”. *IEEE Transactions on Neural Networks*, volume 10, no. 2, pages 450–455, 1999.

[Lee et al., 1999b] T. W. Lee, M. S. Lewicki, M. Girolami, and T. J. Sejnowski. “Blind source separation of more sources than mixtures using overcomplete representations”. *IEEE Signal Processing Letters*, volume 6, pages 87–90, 1999.

[Lee and Schetzen, 1965] Y. W. Lee and M. Schetzen. "Measurement of the Wiener kernels of a non-linear system by cross-correlation". *International Journal of Control*, volume 2, pages 237–254, 1965.

[Liu and Giannakis, 1998] H. Liu and G. Giannakis. "Deterministic approaches for blind equalization of time-varying channels with antenna arrays". *IEEE Transactions on Signal Processing*, volume 46, no. 11, pages 3003–3013, 1998.

[Liu et al., 2010] W. Liu, I. Park, and J. C. Príncipe. "An information theoretic approach of designing sparse kernel adaptive filters". *Accepted for publication in IEEE Transactions on Neural Networks*, 2010.

[Liu et al., 2008] W. Liu, P. P. Pokharel, and J. C. Príncipe. "The kernel least-mean-square algorithm". *IEEE Transactions on Signal Processing*, volume 56, no. 2, pages 543–554, 2008.

[Liu and Príncipe, 2008] W. Liu and J. C. Príncipe. "Extended recursive least squares in RKHS". In *Proceedings of the First Workshop on Cognitive Information Processing*. Santorini, Greece, 2008.

[Lodhi et al., 2002] H. Lodhi, C. Saunders, J. S. Taylor, N. Cristianini, and C. Watkins. "Text classification using string kernels". *Journal of Machine Learning Research*, volume 2, pages 419–444, 2002.

[López-Valcarce and Dasgupta, 2001] R. López-Valcarce and S. Dasgupta. "Blind equalization of nonlinear channels from second-order statistics". *IEEE Transactions on Signal Processing*, volume 49, no. 12, pages 3084–3097, 2001.

[Luengo et al., 2005] D. Luengo, I. Santamaría, and L. Vielva. "A general solution to blind inverse problems for sparse input signals". *Neurocomputing*, volume 69, no. 1-3, pages 198–215, 2005.

[Mathews and Sicuranza, 2000] V. J. Mathews and G. L. Sicuranza. *Polynomial signal processing*. New York: Wiley, 2000.

[Meila and Shi, 2000] M. Meila and J. Shi. "Learning segmentation by random walks". In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems (NIPS 2000)*, pages 873–879. Whistler, BC, Canada, 2000.

[Mercer, 1909] J. Mercer. "Functions of positive and negative type and their connection with the theory of integral equations". *Philosophical Transactions of the Royal Society, London*, volume A 209, pages 415–446, 1909.

[Micchelli et al., 2006] C. Micchelli, Y. Xu, and H. Zhang. "Universal kernels". *Journal of Machine Learning Research*, volume 7, pages 2651–2667, 2006.

[Mika et al., 1999] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. "Fisher discriminant analysis with kernels". In *Neural Networks for Signal Processing IX. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pages 41–48. Madison, WI, USA, 1999.

[Mitchinson and Harrison, 2002] B. Mitchinson and R. Harrison. “Digital communications channel equalization using the kernel adaline”. *IEEE Transactions on Communications*, volume 50, no. 4, pages 571–576, 2002.

[Molgedey and Schuster, 1994] L. Molgedey and H. Schuster. “Separation of a mixture of independent signals using time delayed correlations”. *Physical Review Letters*, volume 72, no. 23, pages 3634–3637, 1994.

[Montalvão Filho et al., 2002] J. Montalvão Filho, B. Dorizzi, and J. M. Mota. “Channel estimation by symmetrical clustering”. *IEEE Transactions on Signal Processing*, volume 50, no. 6, pages 1459–1469, 2002.

[Narendra and Gallman, 1966] K. Narendra and P. Gallman. “An iterative method for the identification of nonlinear systems using a Hammerstein model”. *IEEE Transactions on Automatic Control*, volume 11, no. 3, pages 546–550, 1966.

[Nelles, 2000] O. Nelles. *Nonlinear System Identification*. Springer-Verlag, Berlin, Germany, 2000.

[Ng et al., 2001] A. Y. Ng, M. I. Jordan, and Y. Weiss. “On spectral clustering: Analysis and an algorithm”. In *Proceedings of the 14th Annual Conference on Neural Information Processing Systems (NIPS 2001)*, pages 849–856. Whistler, BC, Canada, 2001.

[Ngia and Sjobert, 1998] K. Ngia and J. Sjobert. “Nonlinear acoustic echo cancellation using a Hammerstein model”. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 1229–1232. Seattle, WA, USA, 1998.

[Ong et al., 2005] C. S. Ong, A. J. Smola, and R. C. Williamson. “Learning the kernel with hyperkernels”. *Journal of Machine Learning Research*, volume 6, pages 1043–1071, 2005.

[Ozertem et al., 2008] U. Ozertem, D. Erdogmus, and M. Carreira-Perpiñán. “Density geodesics for similarity clustering”. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1977–1980. Las Vegas, NV, USA, 2008.

[Pajunen, 1992] G. A. Pajunen. “Adaptive control of Wiener type nonlinear systems”. *Automatica*, volume 28, no. 4, pages 781–785, 1992.

[Papoulis, 1984] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 1984.

[Parzen, 1962] E. Parzen. “On the estimation of a probability density function and mode”. *The Annals of Mathematical Statistics*, volume 33, no. 3, pages 1065–1076, 1962.

[Pawlak et al., 2007] M. Pawlak, Z. Hasiewicz, and P. Wachel. “On nonparametric identification of Wiener systems”. *IEEE Transactions on Signal Processing*, volume 55, no. 2, pages 482–492, 2007.

[Pérez-Cruz et al., 2001] F. Pérez-Cruz, A. Navia-Vázquez, P. Alarcón-Diana, and A. Artés-Rodríguez. “SVC-based equalizer for burst TDMA transmissions”. *Signal Processing*, volume 81, pages 1681–1693, 2001.

[Pezeshki et al., 2005] A. Pezeshki, L. L. Scharf, M. R. Azimi-Sadjadi, and Y. Hua. “Two-channel constrained least squares problems: solutions using power methods and connections with canonical coordinates”. *IEEE Transactions on Signal Processing*, volume 53, pages 121–135, 2005.

[Platt, 1991] J. Platt. “A resource-allocating network for function interpolation”. *Neural Computation*, volume 3, no. 2, pages 213–225, 1991.

[Platt, 1999] J. Platt. “Sequential minimal optimization: A fast algorithm for training support vector machines”. *Advances in Kernel Methods – Support Vector Learning*, volume 208, page 21, 1999.

[Pokharel et al., 2008] P. P. Pokharel, W. Liu, and J. C. Príncipe. “Kernel affine projection algorithms”. *EURASIP Journal on Advances in Signal Processing*, volume 2008, page 12 pages, 2008.

[Pokharel et al., 2009] P. P. Pokharel, W. Liu, and J. C. Príncipe. “Kernel least mean square algorithm with constrained growth”. *Signal Processing*, volume 89, no. 3, pages 257–265, 2009.

[Proakis, 1983] J. G. Proakis. *Digital communications*. McGraw-Hill, New York, 1983.

[Ramírez et al., 2007] D. Ramírez, I. Santamaría, J. Vía, and S. Van Vaerenbergh. “An extension of SVM for piecewise regression”. In *Proceedings of the National Symposium of the International Union of Radio Science (URSI 2007)*. Tenerife, Spain, 2007.

[Rappaport, 2001] T. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall PTR, 2001.

[Rontogiannis et al., 2006] A. Rontogiannis, V. Kekatos, and K. Berberidis. “A square-root adaptive V-BLAST algorithm for fast time-varying MIMO channels”. *IEEE Signal Processing Letters*, volume 13, no. 5, pages pp. 265–268, 2006.

[Roweis, 2000] S. T. Roweis. “One microphone source separation”. In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems (NIPS 2000)*, pages 793–799. Whistler, BC, Canada, 2000.

[Rüping, 2001] S. Rüping. "SVM kernels for time series analysis". In R. Klinkenberg, S. Ruping, A. Fick, N. Henze, C. Herzog, R. Molitor, and O. Schroder, editors, *LLWA 01 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivität*, Forschungsberichte des Fachbereichs Informatik der Universitat Dortmund, pages 43–50. Dortmund, Germany, 2001.

[Sánchez-Giraldo et al., 2010] L. Sánchez-Giraldo, S. Seth, S. Van Vaerenbergh, and J. C. Príncipe. "Efficient computation for information theoretic learning". *Submitted to IEEE Signal Processing Letters*, volume 1, 4 pages, 2010.

[Sands and Cioffi, 1993] N. P. Sands and J. M. Cioffi. "Nonlinear channel models for digital magnetic recording". *IEEE Transactions on Magnetism*, volume 29, pages 3996–3998, 1993.

[Santamaría et al., 2003] I. Santamaría, R. González, C. Pantaleón, and J. C. Príncipe. "Maximum margin equalizers trained with the Adatron algorithm". *Signal Processing*, volume 83, no. 3, pages 593–602, 2003.

[Santamaría et al., 2003] I. Santamaría, J. Ibáñez, M. Lázaro, C. Pantaleón, and L. Vielva. "Modeling nonlinear power amplifiers in OFDM systems from subsampled data: A comparative study using real measurements". *EURASIP Journal on Applied Signal Processing*, volume 12, pages 1219–1228, 2003.

[Saunders et al., 1998] C. Saunders, A. Gammerman, and V. Vovk. "Ridge regression learning algorithm in dual variables". In *Proceedings of the 15th International Conference on Machine Learning (ICML)*, pages 515–521. Madison, WI, USA, 1998.

[Sayed, 2003] A. Sayed. *Fundamentals of Adaptive Filtering*. Wiley, New York, NY, USA, 2003.

[Schetzen, 1980] M. Schetzen. *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Publishing Co., Inc., New York, NY, USA, 1980.

[Schölkopf et al., 2001] B. Schölkopf, R. Herbrich, and A. J. Smola. "A generalized representer theorem". In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory (COLT '01/EuroCOLT '01)*, pages 416–426. Springer-Verlag, London, UK, 2001.

[Schölkopf et al., 1996] B. Schölkopf, A. Smola, and K.-R. Müller. "Nonlinear component analysis as a kernel eigenvalue problem". Technical Report 44, Max-Planck-Institut für biologische Kybernetik, Tübingen, Germany, 1996.

[Schölkopf et al., 1998] B. Schölkopf, A. Smola, and K.-R. Müller. "Nonlinear component analysis as a kernel eigenvalue problem". *Neural Computation*, volume 10, no. 5, pages 1299–1319, 1998.

[Schölkopf and Smola, 2002] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, USA, 2002.

[Sebald and Bucklew, 2000] D. Sebald and J. Bucklew. "Support vector machine techniques for nonlinear equalization". *IEEE Transactions on Signal Processing*, volume 48, no. 11, pages 3217–3226, 2000.

[Seth and Príncipe, 2009] S. Seth and J. C. Príncipe. "On speeding up computation in information theoretic learning". In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Georgia, Atlanta, USA, 2009.

[Shawe-Taylor and Cristianini, 2004] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[Shi and Malik, 2000] J. Shi and J. Malik. "Normalized cuts and image segmentation". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, no. 8, pages 888–905, 2000.

[Silverman, 1986] B. W. Silverman. *Density estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, London, UK, 1986.

[Sjöberg et al., 1995] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky. "Nonlinear black-box modeling in system identification: a unified overview". *Automatica*, volume 31, pages 1691–1724, 1995.

[Solazzi et al., 2001] M. Solazzi, R. Parisi, and A. Uncini. "Blind source separation in nonlinear mixtures by adaptive spline neural networks". In *Proceedings of the 3rd International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2001)*, pages 254–259. San Diego, California, USA, 2001.

[Song et al., 2008] Y. Song, W. Chen, H. Bai, C.-J. Lin, and E. Y. Chang. "Parallel spectral clustering". In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 374–389. Antwerp, Belgium, 2008.

[Sonnenburg et al., 2005] S. Sonnenburg, G. Rätsch, and B. Schölkopf. "Large scale genomic sequence SVM classifiers". In *Proceedings of the 22nd international conference on Machine learning (ICML)*, pages 848–855. Bonn, Germany, 2005.

[Steinwart, 2001] I. Steinwart. "On the influence of the kernel on the consistency of support vector machines". *Journal of Machine Learning Research*, volume 2, pages 67–93, 2001.

[Stoica and Selen, 2004] P. Stoica and Y. Selen. "Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: a refresher". *IEEE Signal Processing Magazine*, volume 21, no. 1, pages 112–114, 2004.

[Suykens et al., 2002] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

[Taleb and Jutten, 1999] A. Taleb and C. Jutten. “Source separation in post-nonlinear mixtures”. *IEEE Transactions on Signal Processing*, volume 47, pages 2807–2820, 1999.

[Taleb et al., 2001] A. Taleb, J. Solé, and C. Jutten. “Quasi-nonparametric blind inversion of Wiener systems”. *IEEE Transactions on Signal Processing*, volume 49, no. 5, pages 917–924, 2001.

[Tan and Wang, 2001] Y. Tan and J. Wang. “Nonlinear blind source separation using higher order statistics and a genetic algorithm”. *IEEE Transactions on Evolutionary Computation*, volume 5, no. 6, pages 600–612, 2001.

[Tarokh et al., 1999] V. Tarokh, H. Jafarkhani, and A. R. Calderbank. “Space-time block codes from orthogonal designs”. *IEEE Transactions on Information Theory*, volume 45, no. 5, pages 1456–1467, 1999.

[Theis and Amari, 2004] F. J. Theis and S. Amari. “Postnonlinear overcomplete blind source separation using sparse sources”. In *Proceedings of the 5th International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2004)*, volume 3195 of *Lecture Notes in Computer Science*, pages 718–725. Granada, Spain, 2004.

[Tikhonov, 1963] A. Tikhonov. “Solution of incorrectly formulated problems and the regularization method”. In *Soviet Mathematics Doklady*, volume 4, pages 1035–1038. 1963.

[Tsang and Kwok, 2004] I. W. Tsang and J. T. Kwok. “Efficient hyperkernel learning using second-order cone programming”. In *Proceedings of the 15th European Conference on Machine Learning (ECML)*, pages 453–464. Pisa, Italy, 2004.

[Van Vaerenbergh et al., 2007a] S. Van Vaerenbergh, E. Estébanez, and I. Santamaría. “A spectral clustering algorithm for decoding fast time-varying BPSK MIMO channels”. In *Proceedings of the 15th European Signal Processing Conference (EUSIPCO)*. Poznań, Poland, 2007.

[Van Vaerenbergh and Santamaría, 2006] S. Van Vaerenbergh and I. Santamaría. “A spectral clustering approach to underdetermined postnonlinear blind source separation of sparse sources”. *IEEE Transactions on Neural Networks*, volume 17, no. 3, pages 811–814, 2006.

[Van Vaerenbergh and Santamaría, 2008] S. Van Vaerenbergh and I. Santamaría. *Intelligent Systems: Techniques and Applications*, chapter A Spectral Clustering Approach for Blind Decoding of MIMO Transmissions over Time-Correlated Fading Channels, pages 351–377. Shaker Publishing, Maastricht, The Netherlands, 2008.

[Van Vaerenbergh and Santamaría, 2010] S. Van Vaerenbergh and I. Santamaría. "Alternating kernel least-squares for supervised identification of Hammerstein systems". *Submitted to IEEE Signal Processing Letters*, volume 1, page 4, 2010.

[Van Vaerenbergh et al., 2009] S. Van Vaerenbergh, I. Santamaría, P. Barbano, U. Ozertem, and D. Erdogmus. "Path-based spectral clustering for decoding fast time-varying MIMO channels". In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2009)*. Grenoble, France, 2009.

[Van Vaerenbergh et al., 2010a] S. Van Vaerenbergh, I. Santamaría, P. E. Barbano, U. Ozertem, and D. Erdogmus. "Path-based clustering for decoding time-varying MIMO channels". *IEEE Transactions on Signal Processing, in preparation*, 2010.

[Van Vaerenbergh et al., 2010b] S. Van Vaerenbergh, I. Santamaría, W. Liu, and J. C. Príncipe. "Fixed-budget kernel recursive least-squares". *Submitted to the 2010 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[Van Vaerenbergh et al., 2006a] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Online kernel canonical correlation analysis for supervised equalization of Wiener systems". In *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Vancouver, Canada, 2006.

[Van Vaerenbergh et al., 2006b] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "A sliding-window kernel RLS algorithm and its application to nonlinear channel identification". In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Toulouse, France, 2006.

[Van Vaerenbergh et al., 2007b] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Nonlinear system identification using a new sliding-window kernel RLS algorithm". *Journal of Communications*, volume 2, no. 3, pages 1–8, 2007.

[Van Vaerenbergh et al., 2008a] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Adaptive kernel canonical correlation analysis algorithms for nonparametric identification of Wiener and Hammerstein systems". *EURASIP Journal on Advances in Signal Processing*, volume 1, Article ID 875351, 13 pages, 2008.

[Van Vaerenbergh et al., 2008b] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "A kernel canonical correlation analysis algorithm for blind equalization of oversampled Wiener systems". In *IEEE Workshop on Machine Learning for Signal Processing (MLSP 2008)*, pages 20–25. 2008.

[Van Vaerenbergh et al., 2010c] S. Van Vaerenbergh, J. Vía, and I. Santamaría. "Blind equalization of post-nonlinear SIMO systems". *Submitted to IEEE Transactions on Signal Processing*, volume 58, pages 1–8, 2010.

[Vanbeylen et al., 2008] L. Vanbeylen, R. Pintelon, and J. Schoukens. "Application of blind identification to nonlinear calibration". *IEEE Transactions on Instrumentation and Measurement*, volume 57, no. 8, pages 1771–1778, 2008.

[Vanbeylen et al., 2009] L. Vanbeylen, R. Pintelon, and J. Schoukens. "Blind maximum-likelihood identification of Wiener systems". *IEEE Transactions on Signal Processing*, volume 57, no. 8, pages 3017–3029, 2009.

[Vapnik, 1995] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[Vía et al., 2005a] J. Vía, I. Santamaría, and J. Pérez. "Canonical correlation analysis (CCA) algorithms for multiple data sets: Application to blind SIMO equalization". In *Proceedings of the 13th European Signal Processing Conference (EUSIPCO)*. Antalya, Turkey, 2005.

[Vía et al., 2005b] J. Vía, I. Santamaría, and J. Pérez. "A robust RLS algorithm for adaptive canonical correlation analysis". In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume IV, pages 365–368. Philadelphia, PA, USA, 2005.

[Vía et al., 2006] J. Vía, I. Santamaría, and J. Pérez. "Effective channel order estimation based on combined identification / equalization". *IEEE Transactions on Signal Processing*, volume 54, no. 9, pages 3518–3526, 2006.

[Vía et al., 2007a] J. Vía, I. Santamaría, and J. Pérez. "Deterministic CCA-based algorithms for blind equalization of FIR-MIMO channels". *IEEE Transactions on Signal Processing*, volume 55, no. 7, pages 3867–3878, 2007.

[Vía et al., 2007b] J. Vía, I. Santamaría, and J. Pérez. "A learning algorithm for adaptive canonical correlation analysis of several data sets". *Neural Networks*, volume 20, no. 1, pages 139–152, 2007.

[Vielva et al., 2001] L. Vielva, D. Erdogmus, and J. C. Príncipe. "Underdetermined blind source separation using a probabilistic source sparsity model". In *Proceedings of the 3rd International Conference on Independent Component Analysis and Blind Signal Separation (ICA 2001)*, pages 675–679. San Diego, California, USA, 2001.

[Vielva et al., 2002] L. Vielva, I. Santamaría, C. Pantaleón, J. Ibáñez, D. Erdogmus, and J. C. Príncipe. "Estimation of the mixing matrix for underdetermined blind source separation using spectral techniques". In *Proceedings of the 11th European Signal Processing Conference (EUSIPCO)*, volume 1, pages 557–560. EURASIP, Toulouse, France, 2002.

[Vincent and Bengio, 2002] P. Vincent and Y. Bengio. "Kernel matching pursuit". *Machine Learning*, volume 48, no. 1, pages 165–187, 2002.

[Volterra, 1887] V. Volterra. “Sopra le funzioni che dipendono da altre funzioni”. In *Atti della Reale Accademia dei Lincei*, volume 3, pages 97–105, 141–146, and 153–158. Rome, Italy, 1887.

[von Luxburg, 2006] U. von Luxburg. “A tutorial on spectral clustering”. Technical Report 149, Max Planck Institute for Biological Cybernetics, 2006.

[Wahlberg, 1991] B. Wahlberg. “System identification using Laguerre models”. *IEEE Transactions on Automatic Control*, volume 36, no. 5, pages 551–562, 1991.

[Wahlberg, 1994] B. Wahlberg. “System identification using Kautz models”. *IEEE Transactions on Automatic Control*, volume 39, no. 6, pages 1276–1282, 1994.

[Wang et al., 2007] J. Wang, A. Sano, T. Chen, and B. Huang. “Blind Hammerstein identification for MR damper modeling”. In *Proceedings of the American Control Conference (ACC 07)*. New York, NY, USA, 2007.

[Westwick and Kearney, 1998] D. Westwick and R. Kearney. “Nonparametric identification of nonlinear biomedical systems, part 1: Theory”. *Critical Reviews in Biomedical Engineering*, volume 26, pages 153–226, 1998.

[Westwick and Kearney, 2000] D. Westwick and R. Kearney. “Identification of a Hammerstein model of the stretch reflex EMG using separable least squares”. In *Proceedings of the 22nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 3, pages 1901–1904. 2000.

[Widrow et al., 1975] B. Widrow, J. R. Glover, J. M. Mccool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, R. C. Goodlin, and R. C. Goodlin. “Adaptive noise cancelling: Principles and applications”. *Proceedings of the IEEE*, volume 63, no. 12, pages 1692–1716, 1975.

[Widrow and Hoff, 1960] B. Widrow and M. E. Hoff. “Adaptive switching circuits”. In *Convention Records of the Western Conference of the Institute for Radio Engineers*, volume 4, pages 96–104. 1960.

[Wiener, 1958] N. Wiener. *Nonlinear Problems in Random Theory*. MIT Technology Press and John Wiley and Sons, New York, first edition, 1958.

[Wigren, 1994] T. Wigren. “Convergence analysis of recursive identification algorithms based on the nonlinear Wiener model”. *IEEE Transactions on Automatic Control*, volume 39, no. 11, pages 2191–2206, 1994.

[Williams and Seeger, 2000] C. Williams and M. Seeger. “Using the Nyström method to speed up kernel machines”. In *Proceedings of the 13th Annual Conference on Neural Information Processing Systems (NIPS 2000)*, pages 682–688. Whistler, BC, Canada, 2000.

[Xiong et al., 2005] H. Xiong, M. N. S. Swamy, and M. Omair Ahmad. "Optimizing the kernel in the empirical feature space". *IEEE Transactions on Neural Networks*, volume 16, no. 2, pages 460–474, 2005.

[Xu et al., 1995] G. Xu, H. Liu, L. Tong, and T. Kailath. "A least-squares approach to blind channel identification". *IEEE Transactions on Signal Processing*, volume 43, no. 12, pages 2982–2993, 1995.

[Zelnik-Manor and Perona, 2004] L. Zelnik-Manor and P. Perona. "Self-tuning spectral clustering". In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems (NIPS 2004)*, pages 1601–1608. Whistler, BC, Canada, 2004.