UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

Centro de Investigación en Tecnoloxías da Información

Tesis doctoral

MACHINE LEARNING ALGORITHMS FOR PATTERN VISUALIZATION IN CLASSIFICATION TASKS AND FOR AUTOMATIC INDOOR TEMPERATURE PREDICTION

Presentada por:

Sadi A. M. Alawadi

Dirigida por:

Manuel Fernández Delgado

David Mera Pérez

Xaneiro de 2018

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Manuel Fernández Delgado, Profesor de Universidad del Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Santiago de Compostela

David Mera Pérez, Investigador Posdoctoral de la Universidad de Santiago de Compostela

HACEN CONSTAR:

Que la memoria titulada **MACHINE LEARNING ALGORITHMS FOR PATTERN VISUALIZATION IN CLASSIFICATION TASKS AND FOR AUTOMATIC INDOOR TEMPERATURE PREDICTION** ha sido realizada por **D. Sadi A.M. Alawadi** bajo nuestra dirección en el Centro Singular de Investigación en Tecnoloxías da Información de la Universidad de Santiago de Compostela (CiTIUS), y constituye la Tesis que presenta para optar al título de Doctor.

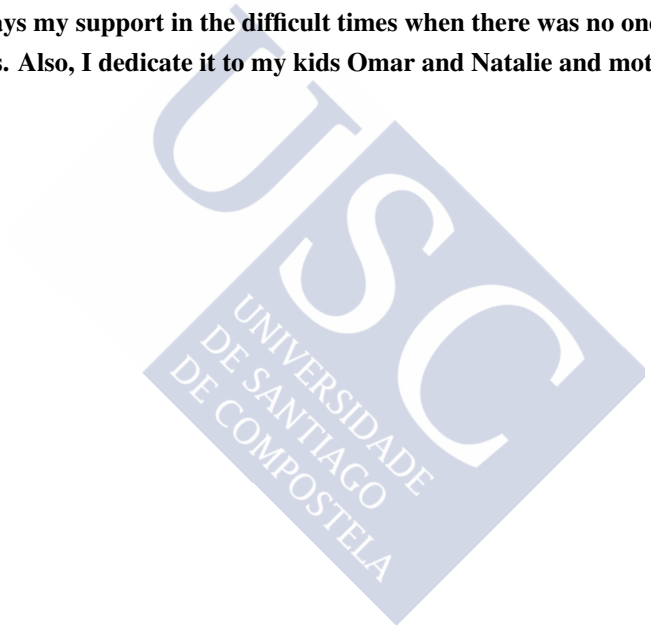
Xaneiro de 2018

Manuel Fernández Delgado
Codirector de la tesis

David Mera Pérez
Codirector de la tesis

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

I dedicate this work to my parents, my wife Rania who spent sleepless nights with me and was always my support in the difficult times when there was no one to answer my queries. Also, I dedicate it to my kids Omar and Natalie and mother-in-law and father-in-law.



USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Storytelling is ultimately a creative act of pattern recognition. Through characters, plot and setting, a writer creates places where previously invisible truths become visible. Or the storyteller posits a series of dots that the reader can connect.

Douglas Coupland

Creativity and insight almost always involve an experience of acute pattern recognition: the eureka moment in which we perceive the interconnection between disparate concepts or ideas to reveal something new.

Jason Silva

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Acknowledgments

During the last few years, I had a lot of challenges that made me more determined to achieve my goals. The major goal amongst them is to present my Ph.D. dissertation. The completion of this doctoral thesis was possible with the support of several people. Therefore, I would like to express my special appreciation to all of them. My Ph.D. main advisor **Dr. Manuel Fernández Delgado** and the co-advisor **Dr. David Mera Pérez**, I am extremely grateful to your valuable guidance, scholarly inputs, and consistent encouragement that I have received throughout the research work. You both have always made yourselves available to clarify my doubts despite your busy schedule, and I consider it as an excellent opportunity to do my doctoral programme under your guidance and to learn from you research expertise.

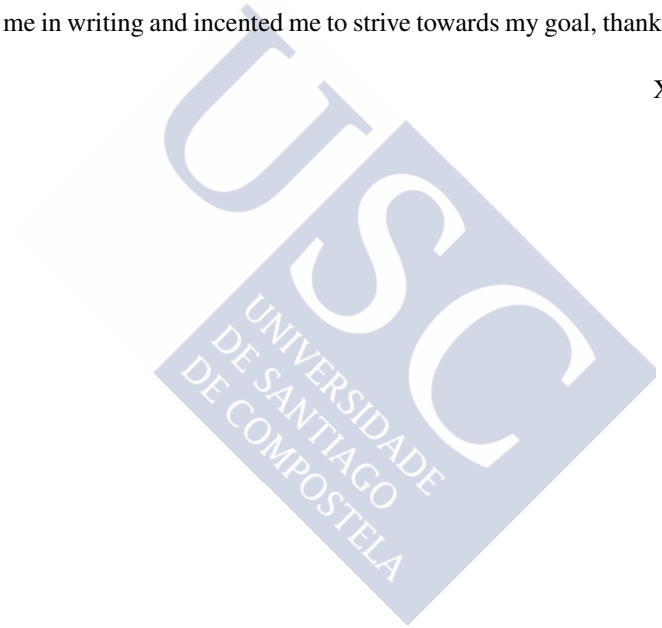
In fact, **Manuel** is the guy who you will immediately love and never forget him. Moreover, he's the warm-hearted advisor and one of the cleverest people I have met. He's always enthusiastic, energetic, and motivated, and he can command an audience. **Manuel**, I would like to thank you for guiding my dissertation and for allowing me to learn from you and grow as a researcher. Your advice on both research as well as on my career has been priceless.

I would also like to show my gratitude for my co-advisor **David** for his support to accomplish my dissertation and for his scientific advice, patience, motivation, immense knowledge and many insightful discussions and suggestions. He is one of my primary resources for getting my scientific questions answered. Furthermore, his influence was not only seen in my scientific work but also in my social life. **Manuel and David**, I would like to thank you again for helping me in research and writing my dissertation. So, without your guidance to the right direction, I wouldn't able successfully complete this work.

In other hand, I'm thankful to our research center Centro Singular de Investigación en Tecnoloxías da Información (CiTiUS) staff for their logistic and technical supports. And more special thanks for **Dr. José Manuel Cotos Yáñez** who handed me all kind of help during this period.

Special thanks to my family. My mother-in law and father-in-law, words and expressions cannot convoy how grateful I am to you. Also, my parents for all of your sacrifices that you have made on my behalf. Your prayers for me were what sustained me thus far. All my friends who supported me in writing and incented me to strive towards my goal, thank you very much.

Xaneiro de 2018



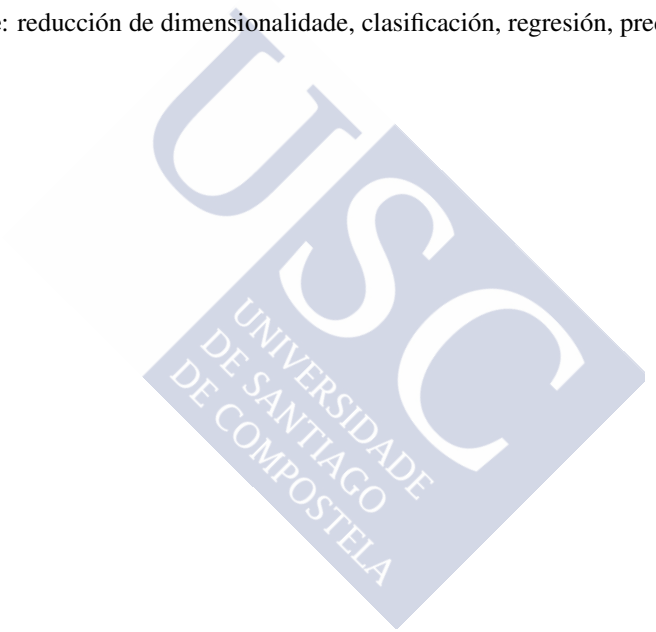
Resumo en galego

Esta tese sitúase no ámbito da aprendizaxe automática (*Machine Learning*), concretamente na clasificación de patróns e na regresión ou aproximación de funcións. Aínda que existen moitos métodos para a clasificación de patróns multi-dimensionais, en xeral compórtanse como "caixas negras" nas que a explicación do seu funcionamento resulta difícil ou imposíbel. Esta tese desenvolve métodos de reducción da dimensionalidade para proxectar problemas de clasificación multi-dimensionais sobre un espazo de dúas dimensións (un plano). Os clasificadores poden así usarse para aprender os datos proxectados e crear mapas bi-dimensionais dos problemas de clasificación que, por tratarse de gráficos, resulten intuitivos e fáciles de entender. Logo de realizar unha revisión das técnicas existentes para a reducción da dimensionalidade, propóñense varios métodos para proxectar os datos multi-dimensionais sobre o plano, minimizando a superposición entre clases. Estos métodos permiten proxectar patróns novos, non usados durante a aprendizaxe da proxección, combinando 8 tipos de proxeccións lineares, cadráticas e polinómicas con 4 medidas da superposición entre clases. As técnicas propostas compáranse con outros 34 métodos de reducción da dimensionalidade existentes na literatura sobre unha ampla colección de 71 problemas de clasificación. Os resultados mostran que a proxección Polynomial kernel discriminant analysis de grao 2 (PKDA2) acada resultados competitivos, creando mapas visuais e auto-explicativos dos problemas de clasificación sobre os que un clasificador de referencia (a máquina de vectores de soporte, ou SVM) acerta só lixeiramente menos que sobre os datos multi-dimensionais orixinais. Proporciónase tamén unha interfaz web e unha aplicación local, desenvolvidas nas linguaxes de programación PHP e Matlab respectivamente, que permiten aplicar todas estas proxeccións para visualizar os mapas 2D de calquera problema de clasificación.

No ámbito da regresión aplicouse unha ampla colección de regresores para a predicción automática de temperaturas en sistemas de climatización (HVAC). Estos sistemas teñen un impacto directo tanto no consumo de enerxía como no confort dos edificios, polo que un modelado preciso e fiábel dos mesmos é o punto de partida para o desenvolvemento de planos de eficiencia enerxética. O emprego de regresores para predecir a evolución da temperatura interior dos edificios en base ás condicións internas y externas (climáticas) permitirían avaliar o impacto das modificacións nos sistemas HVAC dende o punto de vista do confort. Co obxectivo de desenvolver un modelo eficiente para os sistemas HVAC, nesta tese avaliáronse 40 regresores empregando un conxunto de datos reais xerados por un edificio intelixente, o

Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS) da Universidade de Santiago de Compostela. Además desenvolvéronse e compararonse diferentes modelos baseados en redes neuronais que permiten o re-entrenamento automático. Esta característica aporta robustez aos modelos e permítelles: 1) afrontar circunstancias nunca vistas no entrenamento orixinal debidas a situacións climáticas excepcionais; e 2) soportar alteracións nas compoñentes dos sistemas producidas por erros ou cambios nos sistemas de sensorización.

Palabras chave: reducción de dimensionalidade, clasificación, regresión, predicción de temperaturas.



Resumen en castellano

La actual tesis se sitúa en el ámbito del aprendizaje automático (*Machine Learning*), concretamente en la clasificación de patrones y en la regresión o aproximación de funciones. Aunque existen muchos métodos de clasificación de patrones multi-dimensionales, en general se comportan como "cajas negras" en las que la explicación de su funcionamiento resulta difícil o imposible. Esta tesis desarrolla métodos de reducción en la dimensionalidad de los datos para proyectar problemas de clasificación multi-dimensionales sobre un espacio de dos dimensiones (un plano). Los clasificadores pueden así usarse para aprender los datos proyectados y crear mapas bi-dimensionales de los problemas de clasificación que, por tratarse de gráficos, resultan intuitivos y fáciles de entender. Después de realizar una revisión de las técnicas existentes para la reducción de dimensionalidad, se proponen varios métodos para proyectar los datos multi-dimensionales sobre el plano, minimizando la superposición entre clases. Estos métodos permiten proyectar patrones nuevos, no usados durante el aprendizaje de la proyección. Se combinan 8 tipos de proyecciones lineales, cuadráticas y polinómicas con 4 medidas de superposición entre clases. Estas proyecciones se comparan con otros 34 métodos de reducción de dimensionalidad existentes en la literatura sobre una amplia colección de 71 problemas de clasificación. Los mejores resultados han sido obtenidos por la proyección Polynomial kernel discriminant analysis de grado 2 (PKDA2), que crea mapas visuales y auto-explicativos de los problemas de clasificación sobre los que un clasificador de referencia (la máquina de vectores de soporte, o SVM) acierta sólo ligeramente menos que sobre los datos multi-dimensionales originales. Se proporciona también una interfaz web y unha aplicación local, desarrolladas en los lenguajes de programación PHP y Matlab respectivamente, que permiten aplicar estas proyecciones para visualizar los mapas 2D de cualquier problema de clasificación.

En el ámbito de la regresión se ha aplicado una amplia colección de regresores para la predicción automática de temperaturas en sistemas de climatización (HVAC). Estos sistemas tienen un impacto directo tanto en el consumo de energía como en el confort de los edificios, por lo que un modelado preciso y fiable de los mismos es el punto de partida para el desarrollo de planes de eficiencia energética. El empleo de regresores para predecir la evolución de la temperatura interior de los edificios en base a las condiciones internas y externas (climáticas) permitirían evaluar el impacto de las modificaciones en los sistemas HVAC desde el punto de vista del confort. Con el objetivo de desarrollar un modelo eficiente para los sistemas

HVAC, en esta tesis se han evaluado 40 regresores empleando un conjunto de datos reales generados por un edificio inteligente, el Centro Singular de Investigación en Tecnologías de la Información (CiTIUS), de la Universidad de Santiago de Compostela. Además se han desarrollado y comparado diferentes modelos basados en redes neuronales que permiten el re-entrenamiento automático. Esta característica aporta robustez a los modelos y les permite: 1) afrontar circunstancias nunca vistas en el entrenamiento debidas a situaciones climáticas excepcionales; y 2) soportar alteraciones en los componentes de los sistemas producidas por errores o cambios en los sistemas de sensorización.

Palabras clave: reducción de dimensionalidad, clasificación, regresión, predicción de temperaturas.

Summary in English

The current thesis falls in the scope of Machine Learning, specifically it deals with pattern classification and regression or function approximation. Despite there are many approaches for high-dimensional pattern classification, most of them behave as "black boxes" whose operation mode is difficult or even impossible to explain. This thesis develops methods of dimensionality reduction in order to project or map high-dimensional classification problems into a two-dimensional space (i.e., a plane). Classifiers can thus be used to learn the mapped data and to create two-dimensional maps of the classification problems whose graphic nature makes intuitive and easy to understand. After reviewing the existing methods for dimensionality reduction, several approaches are proposed to map high-dimensional data into the 2D space while minimizing the class overlap. These methods allow to map new patterns, not used during the mapping creation. Eight types of linear, quadratic and polynomial mappings are combined with four class overlap measures. These mappings are compared with other 34 dimensionality reduction methods existing in the literature over a wide collection of 71 classification problems. The best results are achieved by the mapping named Polynomial kernel discriminant analysis with degree 2 (PKDA2), which creates visual and self-explaining maps of the classification problems where a reference classifier (the support vector machine, or SVM) achieves an accuracy only slightly lower than using the original high-dimensional patterns. A web and a standalone graphical interface, developed in the programming languages PHP and Matlab, respectively, are also provided in order to visualize the 2D maps for any classification problem.

In the scope of regression, a wide collection of regressors has been applied for the automatic temperature forecasting in household climate systems (HVAC). These systems have a direct impact both in the energy consumption and in the building comfort, so their exact and reliable modeling is very important for the development of energy efficiency plans. The use of regression approaches to forecast the temperature evolution based on internal and external (climatic) conditions would allow to evaluate the impact of changes in HVAC systems from the point of view of comfort. In order to develop an efficient model for the HVAC systems, the current thesis evaluates 40 regressors using a real data set generated in a smart building, the Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS) of the Universidade de Santiago de Compostela. Moreover, different models based on neural networks which allow the automatic re-training have also been developed and compared. This feature

brings robustness to the models and allows them: 1) to learn circumstances never seen during training caused by exceptional climatic situations; and 2) to support alterations in the system components caused by errors or changes in the sensor devices.

Keywords: dimensionality reduction, classification, regression, temperature forecasting.

Resumen extendido

La presente tesis doctoral se centra en el desarrollo y aplicación de técnicas de aprendizaje automático (Machine Learning) para clasificación y regresión. En el primer aspecto, la tesis se centra en la visualización en imágenes 2D de datos numéricos multi-dimensionales en problemas de clasificación. Esta parte del trabajo se ha publicado en la revista internacional Neural Computing and Applications, incluida en el JCR e indexada en en 2^o cuartil de Computer Science - Artificial Intelligence [2]. En el segundo aspecto, la tesis ha aplicado una variada colección de algoritmos de regresión para la predicción de los valores de la temperatura en el interior de edificios. Esta parte del trabajo se ha publicado en el International Conference on Smart Cities (Smart-CT 2017), celebrado en 2017 en Málaga (España) [3].

La clasificación de patrones es uno de los apartados de más interés en el campo del aprendizaje automático. El término "patrones" se refiere a vectores numéricos multi-dimensionales formados por valores numéricos correspondientes a distintas magnitudes que se suponen son relevantes para su clasificación. Por otra parte, se asume la existencia de una colección de clases perfectamente diferenciadas, de modo que cada patrón pertenece a una de estas clases. El objetivo de un método de aprendizaje automático (clasificador) en un problema de clasificación consiste en predecir la clase a la que pertenece cada patrón, con una probabilidad de acierto aceptable, a partir de los valores numéricos del patrón. Para realizar esta predicción, el clasificador debe aprender a partir de una colección de ejemplos, esto es, patrones adecuadamente etiquetados con las clases a las que pertenecen. Existen multitud de clasificadores, la mayoría de los cuales tienen una naturaleza de "caja negra", es decir, que no ofrecen una explicación inteligible sobre su modo de operación. Para resolver esta limitación, las técnicas de minería de datos tratan de aportar técnicas de clasificación que además sean explicativas. Esta tesis trata de aportar métodos para explicar, por una parte, la naturaleza del problema de clasificación (principalmente en qué medida las clases están poco o muy

entremezcladas) y, por otra, el conocimiento adquirido por cualquier clasificador, y no sólo aquéllos que son auto-explicativos. Con este objetivo, la tesis propone el uso de técnicas de reducción de dimensionalidad para proyectar los patrones *n*-dimensionales sobre un espacio bi-dimensional (un plano) en el cual se puedan visualizar como en un mapa. Usando estos patrones 2D proyectados, cualquier clasificador puede ser entrenado para después clasificar los patrones correspondientes a todos los puntos del plano. De este modo, el clasificador compone un mapa donde las regiones del plano asignadas por el clasificador a cada clase se identifican con un color. Este mapa representaría al problema de clasificación, permitiendo visualizar como en un mapa convencional el número y extensión de las regiones asociadas a cada clase, junto con el grado de superposición entre ellas. También permitiría entender el problema de clasificación aprendido por el clasificador a partir de los patrones proyectados en el plano y, para cada patrón nuevo, su localización en el mapa indicaría a qué clase debería asignarse. Por supuesto, para que esto funcione el método de proyección de patrones *n*-dimensionales a 2D debería tener en cuenta la clase a la que pertenece el patrón, es decir, debe ser supervisado. La proyección también debe tener en cuenta la superposición entre los patrones de distintas clases de modo que el mapa resultante evite mostrar las distintas clases completamente entremezcladas.

El capítulo 2 de la tesis realiza una revisión bibliográfica de las técnicas existentes en la literatura para la reducción de dimensionalidad. Estas técnicas se organizan en varios grupos en base a sus características: lineales frente a no lineales, supervisadas frente a no supervisadas, locales frente a globales. Estos grupos son los siguientes:

1. Métodos clásicos, que incluyen linear discriminant analysis (LDA) y generalized discriminant analysis (GDA), principal component analysis (PCA) y Kernel PCA, probabilistic PCA (ProbPCA), Gaussian process latent variable model (GPLVM) y Factor Analysis (FA).

2. Métodos no lineales que conservan propiedades globales de los datos, tales como las distancias entre patrones. Incluyen la proyección de Sammon, Multi-dimensional Data Scaling (MDS), Stochastic Proximity Embedding (SPE), Stochastic Neighbor Embedding (SNE), Symmetric SNE (SSNE), t-distribution SNE (t-SNE), Isomap, Landmark Isomap, Diffusion map y Multi-layer autoencoder.

3. Técnicas no lineales locales, que conservan propiedades locales, es decir, relativas a pequeños entornos de los patrones. Esta categoría incluye Locally Linear Embedding

(LLE), Hessian LLE (HLLE), Neighborhood Preserving Embedding (NPE), Local Tangent Space Alignment (LTSA), Linear LTSA (LLTSA), Laplacian eigenmap, Locality Preserving Projection (LPP), Maximum Variance Unfolding (MVU), Fast MVU, Landmark MVU, Conformal Eigenmap (CCA), Locally linear coordination (LLC), Manifold charting y Coordinated Factor Analysis (CFA).

4. Otros métodos supervisados, que incluye Neighborhood Component Analysis (NCA), Maximally collapsing metric learning (MCML) y Large-margin nearest neighbor (LMNN).

La mayoría de estos métodos de reducción de la dimensionalidad son no supervisados y su objetivo es conservar propiedades geométricas de los datos originales, tales como distancias entre patrones (globales) o combinaciones lineales de los vecinos más cercanos (locales). Sólo unos pocos son supervisados y tienen en consideración las clases a las que pertenecen los patrones originales. Por otra parte, muchos de ellos no proporcionan una proyección propiamente dicha, como función entre los patrones originales y proyectados, sino que sólo dan las versiones proyectadas de los patrones originales. Esto es muy importante, porque en tales casos estos métodos no son capaces de proyectar adecuadamente patrones nuevos no usados durante el aprendizaje de la proyección (conocidos como patrones *out-of-sample* en la literatura).

En el capítulo 3 de esta tesis se proponen nuevos métodos de reducción de dimensionalidad diseñados para proyectar un problema de clasificación multi-dimensional sobre el espacio bi-dimensional. Para obtener un mapa donde la distribución de los patrones de las distintas clases sea interpretable, estos métodos intentan minimizar la superposición 2D entre estos patrones. Dado que existen en la literatura varias medidas para la superposición entre clases (el índice de separabilidad de Thornton, la separabilidad directa de clases y el índice J), se ha propuesto una nueva medida de superposición (la distancia media entre clases) y se han desarrollado varios métodos proyección que usan dichas medidas. Por otra parte, se han probado distintas expresiones matemáticas para proyectar datos desde un espacio n-dimensional (con $n > 2$) hasta un espacio de dimensión 2: funciones lineales, cuadráticas con monomios cruzados y polinómicas sin monomios cruzados con grados desde 2 hasta 7. Los desarrollos matemáticos incluidos en este capítulo permiten calcular las matrices que definen la proyección sobre 2D a partir de los patrones multi-dimensionales originales. Este cálculo resulta extremadamente eficiente desde el punto de vista computacional, pués se reduce a calcular autovalores y autovectores de matrices construidas a partir de los datos originales,

con expresiones analíticas cerradas y evitando cálculos iterativos. La complejidad de estos métodos es muy inferior a la de los métodos descritos en el capítulo 2, la mayoría de los cuales requieren procesos iterativos con un coste computacional muy superior. Por otra parte, los métodos propuestos permiten proyectar cualquier patrón no incluido en el cálculo de la proyección (es decir, patrones *out-of-sample*). Es esta una de sus propiedades más importantes, porque muchos de los métodos existentes no permiten proyectar patrones nuevos, siendo necesario calcular nuevamente la proyección incluyendo los patrones nuevos, proceso que resulta computacionalmente costoso.

Las combinaciones de distintas funciones y medidas de superposición han generado un total de 31 proyecciones, que se han comparado en el capítulo 4 entre sí y con los 34 métodos de reducción de dimensionalidad descritos en el capítulo 2. Los primeros se han implementado en Matlab, y los segundos se han ejecutado usando la implementación proporcionada por la Dimensionality Reduction Toolbox, desarrollada en Matlab para su uso público. La comparación se realiza usando un clasificador estándar para aprender los problemas de clasificación en 2D creados por los distintos métodos de proyección, evaluándose la calidad de cada proyección en base a la calidad de el clasificador entrenado con los datos proyectados por la misma. También se evalúa el clasificador en el problema de clasificación multi-dimensional original. Aunque existen bastantes excepciones, en general el clasificador funciona peor en los problemas proyectados, de modo que hay una cierta pérdida de calidad asociada a la proyección, pérdida que se puede considerar el precio a pagar por obtener una explicación gráfica del funcionamiento del clasificador y del problema de clasificación. La medida de calidad empleada ha sido la denominada “Kappa de Cohen”, que evalúa el grado de coincidencia entre las clases verdaderas y las predichas por el clasificador descartando las coincidencias casuales debidas a desbalanceos entre clases. El clasificador empleado ha sido la Support Vector Machine (SVM), considerada uno de los mejores clasificadores disponibles actualmente. La comparativa se ha desarrollado usando 71 conjuntos de datos (*benchmarks*) disponibles en el repositorio de aprendizaje automático de la University of California at Irving (UCI Machine Learning repository), usados comunmente en la evaluación de algoritmos de clasificación. Para realizar la comparativa entre las distintas proyecciones sobre los 71 problemas de clasificación, hemos empleado el ranking de Friedman, que ordena para cada problema las proyecciones por valores decrecientes de kappa y evalua luego el ranking de cada proyección como su posición en promedio sobre todos los problemas, de modo que la mejor proyección es la de menor ranking.

Los resultados mostrados en el capítulo 4 revelan que la proyección sobre un mapa 2D provoca una pérdida de calidad (medida en términos de kappa) sorprendentemente reducida, de modo que en el 53.5% de los problemas alguna proyección obtiene valores de kappa superiores a los obtenidos en el problema multi-dimensional original. Sin embargo, la proyección que supera al original los datos no es siempre la misma, de modo que seleccionando la proyección que resulta globalmente mejor sí que hay una cierta pérdida con respecto a los datos originales. Esta mejor proyección es el Polynomial Kernel Discriminant Analysis de grado 2 (PKDA2), que combina una proyección polinómica de grado 2 usando como medida de superposición entre clases el índice J. El valor de kappa usando PKDA2 alcanza el 82% del obtenido usando los datos originales en promedio sobre todos los problemas. Existen importantes diferencias entre problemas: para el 64.8% de ellos, la diferencia en términos de kappa entre usar los datos originales y los datos 2D proyectados usando PKDA es negativa (es decir, PKDA2 mejora la calidad, lo cual ocurre en el 11.3% de los problemas) o positiva pero menor del 10%, y sólo para el 22.5% de los problemas esta diferencia supera el 20%. De hecho, en términos de *accuracy* (porcentaje de acierto) la diferencia supera el 20% sólo para el 11% de los problemas. Por tanto, se puede decir que la obtención de un mapa 2D que explique gráficamente el problema de clasificación en términos de regiones asociadas a las distintas clases se consigue con una reducida pérdida en la calidad de la clasificación. No olvidemos que esta pérdida de calidad evalúa el grado en que el mapa representa al problema de clasificación original: diferencias elevadas sugieren que el mapa no está demasiado relacionado con dicho problema. Otras proyecciones que también obtienen resultados aceptables son LDA, PCA y ProbPCA, aunque 17 de las primeras 20 posiciones en el ranking de Friedman corresponden a proyecciones propuestas en esta tesis. De hecho, entre las proyecciones supervisadas, sólo el LDA obtiene una buena posición (la 7ª), situándose las restantes (NCA, MCML y LMNN) muy lejos de las primeras posiciones (kappa medios de 9.60%, 8.82% y 8.14%, frente al 47.45% obtenido por PKDA2).

En términos de coste computacional, los tiempos de ejecución requeridos por las mejores proyecciones han sido bastante reducidos en general, siendo PCA, MDS, LDA y PKDA2 las más rápidas con tiempos promedio entre 0.001 y 0.01 s. Otra de las contribuciones de la tesis ha sido el desarrollo de una interfaz web, utilizando el lenguaje PHP, para la proyección de problemas de clasificación usando las técnicas propuestas y las descritas en el capítulo 2. Esta interfaz está pensada para que de forma pública y anónima cualquier investigador pueda proyectar su problema de clasificación en un mapa 2D para así entender la distribución de

clases. También se ha desarrollado una interfaz *standalone* en Matlab que proporciona una funcionalidad semejante, pero sin la necesidad de una infraestructura web (servidor, base de datos, etc.).

El capítulo 5 presenta las contribuciones de la tesis en la aplicación de técnicas de regresión aplicadas a la predicción de temperaturas en el interior de edificios inteligentes. Este trabajo se enmarca en el ámbito de las iniciativas lideradas por la Unión Europea (UE) enfocadas al desarrollo de nuevos planes de eficiencia energética que contribuyan a una reducción del consumo de energía y de las emisiones de gases de efecto invernadero en los países miembros. Las actividades asociadas a dichos objetivos se enmarcan dentro de la acción "*Climate Action*" dirigida a la lucha contra el cambio climático en la que la UE se ha propuesto como reto reducir las emisiones de gases y aumentar la eficiencia energética en un 20% para el año 2020 respecto a los valores de 1990. La gestión eficiente de los edificios es considerada clave para conseguir las metas marcadas, ya que se estima que éstos representan el 40% del consumo energético y el 36% de las emisiones de CO_2 dentro de la UE. La Universidad de Santiago de Compostela (USC), en el marco de la "*Climate Action*", ha participado en el proyecto Opere, que ha permitido desplegar una red de sensores sobre 45 edificios universitarios. Esta red obtiene, de forma síncrona, más de 10.000 señales que se pueden emplear para el desarrollo de medidas orientadas a la eficiencia energética. La contribución de esta tesis se centró en el análisis de los datos generados por la red de sensores desplegada en el Centro Singular de Investigación en Tecnologías de la Información de la USC (CiTIUS), que genera 667 señales cada 10 segundos, y más concretamente en la gestión eficiente de su sistema de climatización (HVAC, por sus siglas en inglés), dado que este tipo de sistemas tiene un gran impacto tanto en el confort como en el consumo energético de cualquier edificio.

El empleo de algoritmos de aprendizaje automático para predecir la evolución de la temperatura interior de los edificios en base a las condiciones internas y externas (climáticas) permitiría evaluar el impacto de las modificaciones en los sistemas HVAC desde el punto de vista del confort y así afrontar con garantías el desarrollo de nuevos planes energéticos. En esta tesis, y con el objetivo de modelar de manera fiel y eficiente el sistema HVAC del CiTIUS, se analizaron qué características medidas por la red de sensores podían tener relevancia en dicho sistema. De este estudio se seleccionaron 10 características que posteriormente fueron empleadas en el desarrollo y evaluación de 40 regresores pertenecientes a 20 familias diferentes:

1. Linear regression.

2. Generalized linear regression, familia que incluye los regresores glm, penalized glm y glmnet (LASSO y elastic-net regularized GLM).
3. Non-negative least squares (NNLS).
4. Partial least squares: regresores sparse partial least squares (SPLS) y SIMPLS.
5. Least absolute shrinkage and selection operator (LASSO).
6. Ridge regression with forward, backward and sparse input selection (FOBA).
7. Neural networks: multi-layer perceptron (MLP), ensemble de redes neuronales (avNNet), generalized regression neural networks, deep learning neural network y extreme learning machine (ELM).
8. Support vector regression (SVR) con núcleos gausianos.
9. Regression trees: recursive partitioning y M5.
10. Bagging ensembles de árboles de regresión, multi-variate adaptive regression splines (MARS) y CART.
11. Boosting ensemble de GLMs.
12. Gradient boosting machines, con regresores lineales, smoothing splines, regression trees y generalized boosted models.
13. Random forests, quantile randon forests (QRF) y extremely randomized regression trees (extraTrees).
14. Regresión basada en prototipos (cubist).
15. Modelos bayesianos: Bayesian GLM, Bayesian regularized neural network y Bayesian additive regresion tree.
16. Independent component regression (ICR).
17. Generalized additive model (GAM).
18. Regresión mediante procesos gausianos, con kernel lineal y polinómico.
19. Regression LASSO quantil (RQLASSO).

20. Otros métodos: multi-variate adaptive regression splines (MARS) y projection pursuit regression (PPR).

El objetivo de los regresores es predecir con 1, 2 y 3 horas de antelación la temperatura interior de uno de los despachos del CiTIUS. Para ello se emplea un conjunto de datos formado por casi 40.000 patrones correspondientes a los períodos invernales de los años 2015-2016 y 2016-2017. Las métricas empleadas para comparar los regresores son el Mean Square Error (MSE) y el coeficiente de correlación de Pearson (R-coefficient). Los resultados obtenidos destacan la eficiencia de los regresores pertenecientes a la familia Random Forest, siendo extraTrees (extremely randomized regression trees) el que obtuvo las mejores métricas (p. ej. MSE: 0.058 y R-coefficient 0.97 en la predicción a 3 horas).

En un segundo experimento se evalúan 3 regresores basados en redes neuronales que incluyen métodos de aprendizaje incremental. Esta característica aporta robustez a los modelos y les permite afrontar circunstancias nunca vistas en el entrenamiento original, tales como situaciones climáticas excepcionales, y también soportar alteraciones en los componentes de los sistemas producidas por errores o por cambios en los sistemas de sensorización. Para comprobar la robustez de los modelos evaluados, éstos se entrenan exclusivamente con los datos del primer período invernal (2015-2016) y se testean con los del otro período (2016-2017). Por otra parte, se incorporan diferentes niveles de ruido aleatorio en los datos para simular errores o variaciones en los sensores. A modo comparativo, también se entrenó y evaluó, bajo las mismas circunstancias, un regresor extraTrees (sin aprendizaje incremental). Los experimentos demostraron que los regresores adaptativos permiten modelar de forma más precisa los sistemas HVAC y manejar variaciones y situaciones excepcionales (internas y externas) de forma eficiente, ya que las redes neuronales on-line obtuvieron mejores resultados que con extraTrees. El regresor "on-line learning adaptive multilayer perceptron" (OAMLP) obtuvo las mejores métricas en la comparativa (p. ej. MSE 0.021 y R-coefficient 0.975 en la predicción a 3 horas sin ruido), mostrando que la mejor configuración viene dada por un algoritmo que se adapta con cada nuevo patrón, ya que permite adaptarse rápidamente a nuevos escenarios, y que además se reentrena de forma automática e incremental cada semana con todo el conjunto de datos acumulado. Esto se debe a que su actualización permite suavizar las alteraciones en el modelo debidas a escenarios puntuales, e incluso proporciona la posibilidad de adaptar la arquitectura del modelo para mejorar su eficiencia a medida que aumentan los datos de entrenamiento.

Contents

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 1

INTRODUCTION

The current thesis belongs to the scope of pattern classification, which can be defined as the task of assigning on an input pattern $\mathbf{x}$ to a class in pre-specified set of C classes $c_1, \ldots, c_C$. Usually, each pattern $\mathbf{x}$ is a vector of numeric values $(x_1, \ldots, x_n)$, which can be seen as a vector in an n-dimensional space $\mathbb{R}^n$, whose components are usually called inputs, attributes or features. On the other hand, each class, from which we do not know anything more than the patterns which belong to it, can be considered as a region R of $\mathbb{R}^n$ including all these patterns. A classifier can be considered as a function which assigns each pattern $\mathbf{x}$ to the class associated to the region R in which $\mathbf{x}$ is placed. In fact, we can refer to the n-dimensional surfaces which separate the regions $R_1, \ldots, R_C$ associated to different classes. These surfaces can be considered the "borders between classes". This is a very intuitive idea of a classification process, which might provide information about the relative position of the classes, the overlap among them and the complexity of the class borders. In fact, this complexity might provide a qualitative measure, and provide an explanation of, the difficulty of the classification problem. Unfortunately, the high dimensionality n of the input pattern avoids to develop such a graphical visualization of the classification problem. Additionally, this dimensionality causes undesirable effects on the data (e.g., the so-called "Curse of dimensionality") which highly reduce the performance of many classification methods. The scientists try to mitigate these effects by transfering the data from the high-dimensional original space to a somehow meaningful lower-dimensional space (i.e., by finding a function $\mathbb{R}^n \to \mathbb{R}^d$, with $d < n$) where the data keep their "properties" (whatever this word means), using the so-called "dimensionality reduction methods" [36]. Most of these methods (see chapter 2) suppose that data in the origi-

nal high-dimensional space lie on a low-dimensional embedded space (or manifold), trying to discover this space and to somehow represent the data in that low-dimensional space. Moreover, these methods often try to discover this low-dimensional structures in an unsupervised way.

However, the objective of the work developed in the current thesis is slighly different. We are interested in data visualization for classification, so we start from a intrinsically supervised problem, where each pattern belongs to a class. Besides, we try to overcome the limitation imposed in visualization by the pattern high-dimensionality. Specifically, our objective is to formulate a mapping or projection method which transforms a set of patterns in $\mathbb{R}^n$ to $\mathbb{R}^2$ in order to allow the visualization of the class regions, and the borders among them, in a two-dimensional map. Although 2D visualization is a particular case of dimensionality reduction, most methods: 1) are not oriented for classification tasks, so they do not use the class labels, do not operate in a supervised way, and optimize measures which are generally not related to classification; and 2) they reach their full potenciality when $d > 2$, i.e., they reduce the data dimensionality but the new dimension d is larger than 2. Therefore, although we will apply the existing dimensionality reduction methods, we feel that there is a need of methods that project the patterns into a 2D space in such a way that the set of 2D patterns reflects, or keeps a relation to, or reproduces in some way, the original n-dimensional classification problem. From this collection of the 2D projected patterns, it is possible to create a 2D map of the high-dimensional classification problem, composed by the mapped training and test patterns, the regions of $\mathbb{R}^2$ assigned by a classifier to each class and the borders among these regions. This map should be intelligible, in the sense that it must show regions clearly defined for each class, avoiding class overlap as possible. It should also be possible to train a classifier using the 2D-mapped patterns. Different classifiers would create different maps, which might be used to compare classifiers on the same 2D mapped data. The best classifier would be the one which creates the map that is easier to understand. A classifier trained on the 2D mapped patterns should be able to classify a set of new 2D patterns, not seen during the classifier training nor during the mapping creation (called **out-of-sample** or test patterns). The classifier performance for these patterns should not be much lower than using the original n-dimensional training and test patterns, although some lose in accuracy may happen in exchange for data visualization. Otherwise, a high difference in the classifier performance using high-dimensional and 2D patterns would suggest that the latters do not correspond to the original ones from the classifier point of view (in other words, the map does not describe

the original n-dimensional classification problem with reliability). The mappings proposed in the current thesis (see chapter 3) optimize measures of the class overlap, so they are designed to cope with classification tasks, and allow to project out-of-sample n-dimensional patterns, as opposed to some existing methods of dimensionality reduction, which do not provide such an explicit mapping. These methods just create a 2D data set which expectedly represents the original set, but they are not able to map out-of-sample patterns, unless the mapping is calculated again including the new patterns. Chapter 4 compares a relevant collection of existing dimensionality reduction methods with the proposed mappings, describing the experimental work, discussing the results and measuring the performance level which is expectable using the 2D mapped patterns compared to the performance on the original high-dimensional classification problems.

Chapter 5 of the current thesis explores the application of regression methods for the automatic forecasting of temperatures in the context of the development of household climate systems (HVAC). An accurate temperature prediction is very important for an accurate management of these systems in terms of energy efficiency and building comfort. Specifically, a large collection of 40 regression approaches belonging to 20 different families is implemented and compared using real data generated in a smart building, the Centro Singular de Investigación en Tecnoloxías da Información (CiTIUS) of the Universidade de Santiago de Compostela, during two winter seasons (years 2015-2016 and 2016-2017). These data contain 10 inputs: underfloor heating status and temperature; air conditioning status, temperature and humidity; outdoor humidity and temperature; solar radiation; indoor and previous indoor temperature. The prediction was developed considering three forecasting horizons: within 1, 2 and 3 hours. The thesis also develops different neural network models based on the classical multi-layer perceptron which incorporate dynamic training with new patterns. Several versions are considered, which combine weekly re-training and on-line pattern updating. These properties allows the neural networks to learn new patterns never seen during training (e.g., exceptional climatic situations), adapting themselves to system errors and sensor changes. The main contributions of this thesis and the future research lines are compiled by chapter 6.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 2

METHODS OF DIMENSIONALITY REDUCTION

In this chapter we present some of the most popular methods for dimensionality reduction, organized in the groups usually considered in the literature [114]. We distinguish between the classical and modern approaches, and the latter are separated in global and local techniques [24]. A final group includes other supervised methods, which are the most related to the current work, oriented to supervised pattern classification. All over the text, we will denote by $\mathbf{x}_i \in \mathbb{R}^n$ the original high-dimensional (input) pattern and by $\mathbf{y}_i \in \mathbb{R}^d$ the low-dimensional (output) pattern, with $d < n$. We will also denote by $\phi(\cdot)$ the cost function to optimize.

2.1 Classical approaches

This section groups those methods which have been traditionally used for dimensionality reduction. Some of them use the class labels of the original patterns, being therefore supervised, but most of them do not use the class information.

2.1.1 Linear discriminant analysis (LDA)

The LDA [5, 38, 49] is a traditional linear technique used for dimensional reductions and classification tasks (subspace learning), which attempts in $\mathbb{R}^d$ to preserve as much as possible the class structure of the pattern in $\mathbb{R}^n$. The LDA maximizes the Fisher criterion in $\mathbb{R}^d$ in order to project the high dimensional data into the low dimensional space. In order to

transform $\mathbf{x}_i \in \mathbb{R}^n$ into $\mathbf{y} = \mathbf{W}^T\mathbf{x} \in \mathbb{R}^d$, the LDA searches a matrix $\mathbf{W}$ that maximizes the sum of distances between classes, measured by the between-class scatter matrix $\mathbf{S}_b$:

$$\mathbf{S}_b = \sum_{k=1}^{C} n_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T \tag{2.1}$$

where C is the number of classes, n_k is the number of patterns (population) of class c_k, while $\mathbf{m}_k$ is the mean vector of class k and $\mathbf{m}$ is the overall mean vector of the original patterns $\mathbf{x}_i$. Moreover, the LDA tries to minimize the distance within classes, measured by the within-class Scatter matrix $\mathbf{S}_w$:

$$\mathbf{S}_w = \sum_{k=1}^{C} \sum_{i \in c_k} (\mathbf{x}_i - \mathbf{m}_k)(\mathbf{x}_i - \mathbf{m}_k)^T \tag{2.2}$$

Thereby, LDA maximizes the Fisher criterion, defined as the following quotient:

$$\phi(\mathbf{W}) = \frac{\mathbf{W}^T\mathbf{S}_b\mathbf{W}}{\mathbf{W}^T\mathbf{S}_w\mathbf{W}} \tag{2.3}$$

For classification problems with more than two classes, the LDA finds $C-1$ separating hyperplanes and the LDA becomes multi-linear discriminant analysis (MDA). Afterwards, LDA optimizes the scatter matrix to get the required optimal transformation by solving the following generalized eigenvalue problem.

$$\mathbf{S}_b\mathbf{v} = \lambda\mathbf{S}_w\mathbf{v}, \qquad \lambda \neq 0 \tag{2.4}$$

where $\mathbf{v}$ represents the eigenvectors that form the $\mathbf{W}$ matrix columns, and λ are their corresponding eigenvalues. The mapped pattern $\mathbf{y}$ is given by:

$$\mathbf{y} = \mathbf{W}^T\mathbf{x} \tag{2.5}$$

where $\mathbf{W}$ includes the d eigenvectors associated to the largest eigenvalues (denoted henceforth as main eigenvectors). Classical LDA has an intrinsic limitation that requires at least one of the scatter matrices to be non-singular in order to avoid the so-called "under-sampled problem" [126].

2.1.2 Generalized discriminant analysis (GDA)

The GDA [6, 127, 76] is a derivation of LDA which uses the kernel trick to solve non-linearly separable classification problems, which can not be correctly separated by LDA. This technique uses a non-linear kernel function to project the patterns in the input space into a feature (or Hilbert) space $\mathbf{F}$, where the dot product is calculated, in order to learn non-linear

classification functions. The GDA maximizes the scatter between classes and minimizes the scatter within a class, similarly to the LDA, but in the high-dimensional space **F**, while the LDA maximizes the Fisher criterion in the original $\mathbb{R}^n$ space. The GDA uses the kernel PCA algorithm in the first stage to remove noise from the input patterns. Then, it computes the $N \times N$-order kernel matrix **K**, where N is the number or patterns, whose elements are $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j) = \vec{\varphi}(\mathbf{x}_i)^T \vec{\varphi}(\mathbf{x}_j)$. These elements are centered in the feature space by using the following formula:

$$k_{ij} = k_{ij} - \frac{1}{N}\sum_{l=1}^{N} k_{il} - \frac{1}{N}\sum_{l=1}^{N} k_{jl} + \frac{1}{N^2}\sum_{l=1}^{N}\sum_{m=1}^{N} k_{lm} \tag{2.6}$$

The GDA solves the eigenproblem:

$$\mathbf{K}\mathbf{v} = \lambda \mathbf{v} \tag{2.7}$$

creating the matrix **P** with the N eigenvectors $\mathbf{v}_i$ by columns. It has been shown that maximization of the Fisher criterion in the **F** space is equivalent to find the vector **a** that maximizes the Rayleight quotient:

$$\phi(\mathbf{a}) = \frac{\mathbf{a}^T\mathbf{K}\mathbf{W}\mathbf{K}\mathbf{a}}{\mathbf{a}^T\mathbf{K}\mathbf{K}\mathbf{a}} \tag{2.8}$$

where **W** is an $N \times N$ blockdiagonal matrix $\mathbf{W} = \{\mathbf{W}_c\}_{c=1}^{C}$, and the $n_c \times n_c$-order matrix $\mathbf{W}_c$ is $\frac{1}{n_c}\mathbf{I}$. This solution **a** is given by the d normalized eigenvectors $\mathbf{v}_i$ of the matrix $\mathbf{P}^T\mathbf{W}\mathbf{P}$ associated to the largest eigenvalues:

$$a_i = \frac{|\mathbf{v}_i|}{\sqrt{\mathbf{v}_i^T\mathbf{K}\mathbf{v}_i}}, \qquad i = 1, \ldots, d \tag{2.9}$$

and the i-th component y_i of the mapped pattern $\mathbf{y} \in \mathbb{R}^d$ is given by:

$$y_i(\mathbf{x}) = a_i \sum_{j=1}^{N} K(\mathbf{x}_j, \mathbf{x}), \qquad i = 1, \ldots, d \tag{2.10}$$

Note that this method does not provide an explicit mapping for new out-of-sample patterns, so an interpolation method is required to map them into $\mathbb{R}^d$. This also happens with other methods of dimensionality reduction, as we shall see.

2.1.3 Principal component analysis (PCA)

The PCA [59, 55] is a very popular statistical algorithm for exploratory data analysis which is also used for data pre-processing, image compression, data reduction and many other things. It develops an orthogonal transformation that linearly projects the patterns from $\mathbb{R}^n$ to $\mathbb{R}^d$ using the formula $\mathbf{y} = \mathbf{W}^T\mathbf{x}$, where the matrix $\mathbf{W}$ is composed by d principal components, which are vectors defining the directions with the maximum variability in $\mathbb{R}^n$. The subset of the principal components associated to the d largest eigenvalues define the low dimensional space $\mathbb{R}^d$. After standarize the data (i.e., pre-process the patterns substracting the mean and dividing by the standard deviation of each input, in such a way that the pre-processed patterns have zero mean and standard deviation one for each input), PCA computes the $n \times n$-order covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T$ and solves the following eigenvalue problem:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v} \tag{2.11}$$

The n eigenvectors $\mathbf{v}_i$ are sorted by decreasing order of their eigenvalues, which measure the data variance on the direction of each eigenvector. The eigenvalues near to zero identify directions in $\mathbb{R}^n$ where the data variance is low, they are less informative and thus can be dropped. The d eigenvectors whose eigenvalues are above certain threshold are selected as columns for the $\mathbf{W}$ matrix which define the mapping. When d is not pre-defined, the principal components are often selected in order to account more than 95% of the data variance.

2.1.4 Kernel PCA

This method [78, 98] is a nonlinear version of PCA that projects the patterns $\mathbf{x}_i \in \mathbb{R}^n$ to a feature space $\mathbf{F}$ of a dimension which may be infinite $\vec{\varphi} : \mathbb{R}^n \to \mathbf{F}$, defined implicitely by a kernel $K(\mathbf{x},\mathbf{y}) = \vec{\varphi}(\mathbf{x})^T\vec{\varphi}(\mathbf{y})$ using the kernel trick. The PCA is applied to the $N \times N$-order kernel matrix $\mathbf{K}$, whose elements are $k_{ij} = K(\mathbf{x}_i,\mathbf{x}_j)$, and the following eigenvector problem is solved:

$$\mathbf{K}\mathbf{v} = \lambda N\mathbf{v} \tag{2.12}$$

The KPCA extracts the principal components which lead to the projection of the patterns orthogonally into $\mathbb{R}^d$. Furthermore, the eigenvectors lie in the space expanded by $\vec{\varphi}(\mathbf{x}_1),\ldots,\vec{\varphi}(\mathbf{x}_N)$ and can be expressed as:

$$\mathbf{v} = \sum_{i=1}^{N} a_i \vec{\varphi}(\mathbf{x}_i) \tag{2.13}$$

Sometimes the $\mathbf{F}$ space does not have a zero-mean and it is necessary to center the elements of the $\mathbf{K}$ matrix by modifying its entries using eq. 2.6. Besides, the coefficients a_i associated to the nonzero eigenvalues are normalized by requiring the corresponding eigenvectors in the feature space to be also normalized using:

$$a_i = \frac{|\mathbf{v}_i|}{\sqrt{\lambda_i}}, \qquad i = 1, \ldots, N \tag{2.14}$$

In order to construct the low dimensional pattern $\mathbf{y} \in \mathbb{R}^d$, the KPCA projects a test pattern $\mathbf{x}$ using the d principal components (i.e., the d eigenvectors associated to the largest eigenvalues) extracted by the traditional PCA as in eq. 2.10. The KPCA does not develop any iterative optimization to reduce the pattern dimension. It can also represent out-of-sample patterns not included in the training set, and it has been succesfully applied to extraction of pattern structures, classification [63], face recognition [62], regression [92], de-noising and speech recognition [4].

2.1.5 Probabilistic PCA (ProbPCA)

The ProbPCA [112] is an iterative extension of PCA that uses a probabilistic Gaussian latent variable[1] model to solve the limitations inherent in the regular PCA, such as dealing with missing data patterns and lack of an explicit generative model. This method is formulated in a linear form as $\mathbf{x}_i = \mathbf{W}\mathbf{y}_i + \mathbf{m} + \vec{\varepsilon}_i$, where the $n \times d$-order matrix $\mathbf{W}$ develops the linear mapping from $\mathbb{R}^d$ to $\mathbb{R}^n$, while $\mathbf{y}_i \in \mathbb{R}^d$ is the latent (unobserved) input that describes the relation between observed patterns in $\mathbb{R}^d$ (i.e., the pattern projected to the low-dimensional space), $\mathbf{m}$ is the mean of the observed variables $\mathbf{x}_i$ (the original high-dimensional patterns), and $\vec{\varepsilon}_i$ is the error or data noise. It is assumed that both $\mathbf{y}_i$ and $\vec{\varepsilon}_i$ follow two normal Gaussian distributions $\mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the $d \times d$ identity matrix, and $\vec{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, with isotropic variance σ. Generally, ProbPCA uses an iterative algorithm called Expectation Maximization (EM) to estimate the unobserved variables $\mathbf{y}_i$ in two steps [80]:

1. **E-step** (expectation): it performs the estimation $\langle \mathbf{y}_i \rangle$ of the unobserved variable $\mathbf{y}_i$ (projected pattern in $\mathbb{R}^d$) from the observed variable $\mathbf{x}_i$ (original pattern in $\mathbb{R}^n$) and the

[1] A latent variable or input is an unobservable variable that describes the relationship between observed patterns in the high-dimensional space and represents those patterns in the low-dimensional space.

current value of parameters ($\mathbf{W}$,σ^2):

$$\langle \mathbf{y}_i \rangle = (\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1}\mathbf{W}^T(\mathbf{x}_i - \mathbf{m}) \tag{2.15}$$

$$\langle \mathbf{y}_i\mathbf{y}_i^T \rangle = \sigma^2(\mathbf{W}^T\mathbf{W} + \sigma^2\mathbf{I})^{-1} + \langle \mathbf{y}_i \rangle \langle \mathbf{y}_i \rangle^T \tag{2.16}$$

2. **M-step** (maximization): it estimates $\mathbf{W}$ and σ^2 that maximize the expectation of log-likelihood $\mathscr{L}$ with respect to the unknown variable $\mathbf{y}_i$ given the known variable $\mathbf{x}_i$:

$$\mathscr{L} = -\frac{N}{2}\{n\ln(2\pi) + \ln|\mathbf{C}| + \text{trace}(\mathbf{C}^{-1}\mathbf{S})\} \tag{2.17}$$

where $\mathbf{C} = \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I}$ is the observed covariance matrix and $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$ represents the sample covariance matrix. The matrix $\mathbf{W}$ which provides the maximum likelihood is calculated as:

$$\mathbf{W} = \mathbf{U}(\Lambda - \sigma^2\mathbf{I})^{1/2}\mathbf{R} \tag{2.18}$$

where $\mathbf{U}$ is a $n \times d$-order matrix whose columns are the d main eigenvectors of $\mathbf{S}$; the $d \times d$-order diagonal matrix Λ contains the d largest eigenvalues of $\mathbf{S}$; and the $d \times d$-order matrix $\mathbf{R}$ is an arbitrary orthogonal rotation. Finally, ProbPCA extends its scope to deal with patterns which are noisy or lack some inputs, and can be utilized as a general Gaussian density model [128].

2.1.6 Gaussian process latent variable model (GPLVM)

This is an unsupervised technique [65] that uses statistical models based on Gaussian proceses, and solves the non-linearity limitation of PCA. The GPLVM preserves the dissimilarity between patterns in $\mathbb{R}^d$ as it is in $\mathbb{R}^n$, so that if the patterns are far apart in $\mathbb{R}^n$ they will be far apart in $\mathbb{R}^d$. This technique has shown its ability to deal with the situations that have only a small number of training patterns including sparse input set.

The GPLVM maps the data onto $\mathbb{R}^d$ by maximizing the Gaussian process likelihood with respect to the latent variable $\mathbf{y} \in \mathbb{R}^d$ (the projected pattern). A Gaussian kernel function is used to map the original patterns $\mathbf{x} \in \mathbb{R}^n$ to $\mathbb{R}^d$. The scaled conjugate gradient (SCG) method is used to solve the eigenvalue problem by optimizing iteratively each latent variable, marginalising the likelihood parameters to represent the patterns in $\mathbb{R}^d$. Besides, sparsification is used to speed up the computation process by selecting a subset of patterns called "active

set" using the informative vector machine (IVM) method, which selects patterns sequentially according to the reduction in the subsequent process's entropy that they provide. This method maps the original patterns $\mathbf{x}_i$, but it does not provide an explicit mapping for out-of-sample patterns.

2.1.7 Factor analysis (FA)

Factor Analysis has been widely used in psychology, economy and operations research, and also for dimensional reduction in order to preserve the correlation between observed data [111, 105]. The FA represents the patterns in $\mathbb{R}^d$ estimating an underlying (latent) variable $\mathbf{y}_i \in \mathbb{R}^d$, called **Factor**, and describing the observed factors $\mathbf{x}_i \in \mathbb{R}^n$ as a linear combination of the underlying factors and a stochastic error term [60]:

$$\mathbf{x}_i = \mathbf{L}\mathbf{y}_i + \vec{\varepsilon}_i, \quad i = 1,\ldots,N \tag{2.19}$$

or in extended form:

$$x_{ij} = \sum_{k=1}^{d} l_{jk} y_{ik} + \varepsilon_{ij}, \qquad i = 1,\ldots,N; j = 1,\ldots,d \tag{2.20}$$

where l_{ij} is an element of the $n \times d$-order factor loading matrix $\mathbf{L}$ and $\vec{\varepsilon}_i$ is unobserved error term with $\langle \vec{\varepsilon}_i \rangle = \mathbf{0}$. The FA searches for a matrix $\mathbf{L}$ which provides the maximum likelihood on the observed covariance matrix [74]. The EM algorithm (see section 2.1.5) is used in order to iteratively find the matrix $\mathbf{L}$, starting from a random $n \times d$-order matrix $\mathbf{A}$ and a covariance matrix $\Sigma = \mathbf{I}_n$, the $n \times n$-order identity matrix, we have:

1. **Expectation** step: $\mathbf{C} = (\mathbf{A}\mathbf{A}^T + \Sigma)$, $\mathbf{M} = \mathbf{A}^T\mathbf{C}^{-1}\mathbf{X}$ and $\mathbf{S} = N(\mathbf{I}_d - \mathbf{A}^T\mathbf{C}^{-1}\mathbf{A}) + \mathbf{M}\mathbf{M}^T$
2. **Maximization** step: $\mathbf{A} = \mathbf{A}\mathbf{M}^T\mathbf{S}^{-1}$, $\Sigma = \dfrac{\mathbf{X}\mathbf{X}^T - \mathbf{A}\mathbf{M}\mathbf{X}^T}{N} + \varepsilon\mathbf{I}_n$. The new log-likelihood is calculated as $\mathscr{L} = \dfrac{1}{2}\left[\log(\det \mathbf{C}^{-1}) + \dfrac{\mathrm{trace}(\mathbf{C}^{-1}\mathbf{S})}{N}\right]$

The process finishes when the log-likelihood $\mathscr{L}$ does not change or a number of iterations is reached. Finally, the mapped pattern in $\mathbb{R}^d$ can be found as:

$$\mathbf{y} = \mathbf{A}^T\mathbf{x} \tag{2.21}$$

2.2 Global nonlinear methods

These methods try to preserve global properties of the pattern set, usually the pairwise distance between patterns, but allowing non-linear mappings between $\mathbb{R}^n$ and $\mathbb{R}^d$.

2.2.1 Sammon mapping

The Sammon mapping is one of the most classical data reduction algorithm,which maps the data from $\mathbb{R}^n$ to $\mathbb{R}^d$ while preserving the structure of inter-pattern distance [95], measured by an error function (the Sammon or Kruskal stress) defined by:

$$\phi = \frac{\sum\limits_{i<j} \frac{(\delta \mathbf{x}_{ij} - \delta \mathbf{y}_{ij})^2}{\delta \mathbf{x}_{ij}}}{\sum\limits_{i<j} \delta \mathbf{x}_{ij}} \tag{2.22}$$

Where $\delta \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j \in \mathbb{R}^n$ and analogous for $\delta \mathbf{y}_{ij} \in \mathbb{R}^d$. The Sammon stress is minimized using gradient descent. It is widely known in the literature that the Sammon mapping is not able to deal with complex high-dimensional data structures, nor to handle big data sets with lots of patterns. One of the reasons is that all the patterns share similar weights on the Sammon stress in high-dimensional spaces, so this mapping mixes distant and nearby patterns [67], leading to the well-known **crowding** problem. Besides, it just maps the patterns $\mathbf{x}_i$, but it does not provide an explicit mapping for new patterns. This also happens to the following methods (MDS, SPE, SNE, SSNE and t-SNE).

2.2.2 Multidimensional data scaling (MDS)

The MDS [113, 14] maps pairwise patterns from $\mathbb{R}^n$ onto $\mathbb{R}^d$ while retaining the inter-distance between patterns as much as possible in both spaces, using a dissimilarity matrix that describes the geometric structure of the pattern set. Thus, it only requires a matrix with the pairwise distances between patterns, and not the whole patterns themselves, which is an advantage for computing and storing requirements. This method is used for exploring the patterns, dimensionality reduction and more applications. There are two variants depending on the way to compute the dissimilarity matrix:

1. **Metrical MDS**, which uses distances, although classical MDS transforms distances into inner product. The loss function of the distances is called stress (ϕ), and it measures

the difference between the pairwise distances in $\mathbb{R}^n$ and $\mathbb{R}^d$. The Sammon (eq. 2.22) and raw stress functions are the most important. The latter is defined by:

$$\phi(\mathbf{Y}) = \sum_{i<j} (|\delta \mathbf{x}_{ij}| - |\delta \mathbf{y}_{ij}|)^2 \tag{2.23}$$

where $\mathbf{Y}$ is the matrix with the mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$, while $\delta \mathbf{x}_{ij}$ and $\delta \mathbf{y}_{ij}$ are defined in section 2.2.1 (Manhattan and maximum distances are also used by some MDS versions). The Sammon stress emphasizes more on keeping distances which are originally small. Other versions of MDS use more general stress functions, given by:

$$\phi(\mathbf{Y}) = \sqrt{\frac{1}{s} \sum_{i<j} [f(\delta \mathbf{x}_{ij}) - \delta \mathbf{y}_{ij}]^2} \tag{2.24}$$

where s is a scale factor. With Euclidean distances and f identity, we achieve the PCA solutions without correlation scaling.

2. **Non-metric MDS**, which use the rank ordering of the distances instead of the distance itself (although [15] states that MDS is metric when the above $f(\delta \mathbf{x}_{ij})$ function is linear, and non-metric otherwise).

The stress function is minimized using a eigendecomposition of the pairwise dissimilarity matrix, the conjugate gradient or a pseudo-Newton method. The literature reports some defficiencies of MDS to represent small distances (local structure of the patterns). Besides, using Euclidean distances it does not take into account the distribution of the neighboring patterns. Thus, when the patterns fall in a curved manifold (e.g., a spiral line in $\mathbb{R}^2$), two patterns can be considered nearby MDS, whereas their distance alongside the manifold (spiral) is comparative large.

2.2.3 Stochastic proximity embedding (SPE)

The SPE [1] is a self-organizing iterative technique that preserves the Euclidean distance between patterns in the low dimension space with respect the input space, and it is therefore based on the proximity relationship. It minimizes the raw stress function of MDS, re-written as:

$$\phi(\mathbf{Y}) = \sum_{i<j}(\delta \mathbf{y}_{ij} - r_{ij})^2, \quad r_{ij} = \frac{|\delta \mathbf{x}_{ij}|}{\max\limits_{k<l}\{|\delta \mathbf{x}_{kl}|\}} \tag{2.25}$$

The SPE uses an iterative process which starts with randomly selected N patterns $\mathbf{y}_i \in \mathbb{R}^d$, with $y_{ij} \in [0,1]$, and it uses an efficient rule to update them: s pairs of patterns $(\mathbf{y}_i, \mathbf{y}_j)$ are selected, and the matrix $\mathbf{Y}$ with their Euclidean pairwise distances is calculated. Then, $\mathbf{y}_i$ and $\mathbf{y}_j$ are updated in such a way that the difference between the distance r_{ij} in $\mathbb{R}^n$ and $\delta \mathbf{y}_{ij}$ in $\mathbb{R}^d$ is decreased:

$$\mathbf{y}_i(t+1) = \mathbf{y}_i(t) + \alpha(t)\delta \mathbf{y}_{ij}(t) \tag{2.26}$$

$$\mathbf{y}_j(t+1) = \mathbf{y}_j(t) + \alpha(t)\delta \mathbf{y}_{ji}(t) \tag{2.27}$$

$$\alpha(t) = \lambda \frac{r_{ij} - |\delta \mathbf{y}_{ij}(t)|}{2|\delta \mathbf{y}_{ij}(t)| + \varepsilon} \tag{2.28}$$

where λ and ε are the learning rate and the regularization parameter to avoid divisions by zero, respectively. The process is repeated until the stress $\phi(\mathbf{Y})$ decreases below some threshold, or until a number of iterations is reached.

2.2.4 Stochastic neighbor embedding (SNE)

The SNE [53] is a non-linear data reduction technique maps patterns from $\mathbb{R}^n$ to $\mathbb{R}^d$ preserving a neighborhood identity while retaining the pairwise Euclidean distance between the original patterns $\mathbf{x}_i \in \mathbb{R}^n$ as much as possible. This is accomplished by computing a symmetric probability (p_{ij}) for the original patterns using the following equation.

$$p_{j|i} = \frac{\mathrm{e}^{-|\delta \mathbf{x}_{ij}|^2/2\sigma^2}}{\sum\limits_{k\neq i}\mathrm{e}^{-|\delta \mathbf{x}_{ik}|^2/2\sigma^2}} \tag{2.29}$$

In $\mathbb{R}^d$ the SNE calculates the induced probability q_{ij} between each pattern $\mathbf{y}_i$ and its nearest neighbor $\mathbf{y}_j$.

$$q_{j|i} = \frac{\mathrm{e}^{-|\delta \mathbf{y}_{ij}|^2}}{\sum\limits_{k\neq i}\mathrm{e}^{-|\delta \mathbf{y}_{ik}|^2}} \tag{2.30}$$

The SNE minimizes a cost function different to MDS, given by the sum of Kullback-Leibler divergences of p_{ij} and q_{ij} between all neighbors:

$$\phi(\mathbf{Y}) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{2.31}$$

The $\phi(\mathbf{Y})$ is minimized in the original SNE by a gradient descent method, with Gaussian jitter to avoid local minima. Specifically, the gradient has the form:

$$\frac{\partial \phi}{\partial \mathbf{y}_i} = 2 \sum_{j=1}^{N} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}) \delta \mathbf{y}_{ij} \tag{2.32}$$

and the updating rule for $\mathbf{y}_i$, including a momentum term, is given by:

$$\mathbf{Y}(t+1) = \mathbf{Y}(t) + \eta \frac{\partial \phi}{\partial \mathbf{Y}} + \alpha(t) \left[\mathbf{Y}(t-1) - \mathbf{Y}(t-2) \right] \tag{2.33}$$

where η and $\alpha(t)$ are the learning rate and momentum parameter, respectively. The trust-region algorithm has been also used as an alternative minimization method [83].

2.2.5 Symmetric SNE (SSNE)

The symmetric SNE is a variant of SNE that maps the data from $\mathbb{R}^n$ to $\mathbb{R}^d$, with $d \ll n$, based on the pairwise similarity matrix attempting to preserve the neighbor identity [23, 125]. The SSNE computes $q_{ij} = q_{j|i}$ as in SNE in $\mathbb{R}^d$, but in $\mathbb{R}^n$ it defines p_{ij} according to:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n} \tag{2.34}$$

in order to avoid problems due to outliers. The stress function to be minimized and its gradient are given by:

$$\phi = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \quad \frac{\partial \phi}{\partial \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij}) \delta \mathbf{y}_{ij} \tag{2.35}$$

The use of a symmetric cost function, whose gradient is simpler than SNE, and the addition of momentum terms speed up the optimization with respect to SNE. However, the patterns mapped by SSNE in $\mathbb{R}^d$ are often compressed near to the center, leading to the crowding problem.

2.2.6 T-distribution SNE (t-SNE)

The t-SNE is a variation of SSNE [115] which uses a student t-distribution with a single degree of freedom, as a distribution in $\mathbb{R}^d$ with a tail more heavy than a Gaussian distribution

in order to avoid the crowding problem of SSNE. This, the joint probabilities q_{ij} in $\mathbb{R}^d$ are defined as:

$$q_{ij} = \frac{(1+|\delta \mathbf{y}_{ij}|^2)^{-1}}{\sum_{k \neq l}(1+|\delta \mathbf{y}_{kl}|^2)^{-1}}, \qquad i,j = 1,\ldots,N \tag{2.36}$$

The gradient of the stress function $\phi(\mathbf{W})$, defined as SNE, is given by:

$$\frac{\partial \phi}{\partial \mathbf{y}_i} = 4\sum_j \frac{p_{ij}-q_{ij}}{1+|\delta \mathbf{y}_{ij}|^2}\delta \mathbf{y}_{ij}, \qquad i = 1,\ldots,N \tag{2.37}$$

The minimization of ϕ uses the same update equation 2.33 as SNE. The t-SNE produces better data visualization than existing techniques and it preserves in $\mathbb{R}^d$ the local and global data structure of the original patterns in $\mathbb{R}^n$.

2.2.7 Isomap

This approach is a combination of MDS and PCA (subsections 2.2.2 and 2.1.3) which preserves the intrinsic geometry of the data [109], and specifically the pairwise geodesic distance between patterns, i.e., the distance alongside the curvilinear manifold which best describes the pattern set. These geodesic distances are calculated by creating a neighborhood graph G which connects each original pattern $\mathbf{x}_i$ to its k nearest neighbors (in some versions, to the neighbors within a fixed-radius). The geodesic distance between two patterns is estimated as the length of the shortest path between them in G, evaluated using Dijkstra's or Floyd's shortest-path algorithms. A geodesic distance matrix is composed with all the geodesic distances between patterns. The mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ are calculated by applying MDS on this matrix. Therefore, Isomap minimizes the 2-norm of the difference between the geodesic matrix and the pairwise distance matrix between the mapped patterns $\mathbf{y}_i$. It has been shown that Isomap may be topologically unstable.

2.2.8 Landmark Isomap

This method [101] is an extension of Isomap thats addresses its computational bottlenecks: 1) computation of the $N \times N$-order shortest path matrix between patterns $\mathbf{x}_i \in \mathbb{R}^n$; and 2) computation of the $N \times N$-order eigenvalue matrix, with $O(N^3)$ complexity, in order to visualize the mapped data in $\mathbb{R}^d$. Thereby, landmark Isomap performs a random selection of a small subset of n patterns $\mathbf{x}_i$, called landmarks, with $d \leq n \ll N$. These landmarks are used to compute the

shortest path matrix, of order $n \times N$ instead of $N \times N$ for the full geodesic matrix. Then, from the geodesic matrix it computes the matrix with the main eigenvectors in order to produce the data representation in $\mathbb{R}^d$ [86]. These steps reduce the execution time, while preserving locally and globally the Euclidean distances between patterns in $\mathbb{R}^n$ with respect to $\mathbb{R}^d$.

2.2.9 Diffusion map

A diffusion map [20] creates a graph of the patterns using Markov random walks as the first step. All the nodes in the graph are connected together, and the weight of the edges in the graph between two patterns $(\mathbf{x}_i, \mathbf{x}_j)$ is computed using the Gaussian (or diffusion) kernel [25]:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{e}^{-|\delta \mathbf{x}_{ij}|^2/\alpha} \tag{2.38}$$

Jumping form pattern $\mathbf{x}_i$ to a nearby pattern $\mathbf{x}_j$ is a one-time step of the random walk. Hence the diffusion distance is defined by:

$$D^t(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{N} \frac{(p_{ik}^t - p_{jk}^t)^2}{\varepsilon(\mathbf{x}_k)^0} \tag{2.39}$$

where p_{ik}^t is the probability of jumping from $\mathbf{x}_i$ to $\mathbf{x}_k$ at time step t, computed in the diffusion process through the whole pattern set at different time scales; and $\varepsilon(\mathbf{x}_k)^0$ is $\varepsilon(\mathbf{x}_k)$ calculated at $t = 0$. This prior step produces the diffusion matrix. Then, eigenvectors and eigenvalues of the diffusion matrix are calculated to represent the patterns $\mathbf{y}_i$ in $\mathbb{R}^d$ by selecting directions in the diffusion space associated to the largest eigenvalues. The diffusion map attempts to retain the diffusion distance, being robust to noise perturbation, and its computational cost is relatively low. Note that no explicit mapping is available for new out-of-sample patterns.

2.2.10 Multi-layer autoencoder

An autoencoder [54] is a feed-forward neural network with an odd number of hidden layers that is trained to give on its output the same pattern as the input. The autoencoder is trained using the original patterns $\mathbf{x} \in \mathbb{R}^n$, so during training they minimize the mean squared error between the desired and predicted output. The low-dimensional mapped pattern $\mathbf{y} \in \mathbb{R}^d$ is codified in the middle hidden layer for each input pattern. As shown by Figure 2.1, the $\mathbf{x}_i$ pattern is passed as input to the first part of the network, called **encoder**. The encoder reduces its dimension, applying a transformation $\phi : \mathbb{R}^n \to \mathbb{R}^d$. The middle hidden layer (**code**) captures

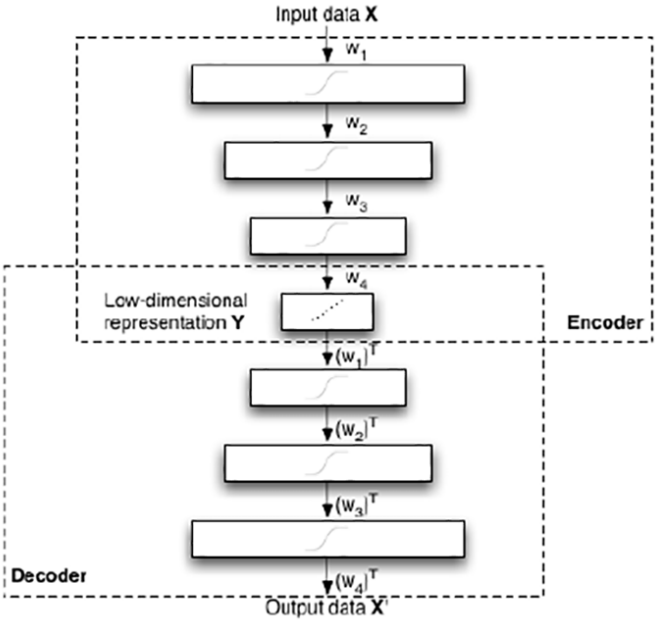


Figure 2.1: Schematic structure of a multi-layer autoencoder (extracted from [71]).

the important features that represent the pattern in $\mathbb{R}^d$. Finally, the last part (called **decoder**) restores on the output layer the input pattern from the **code** into the original dimension, developing a transformation $\psi : \mathbb{R}^d \rightarrow \mathbb{R}^n$. The autoencoder method is able to learn the meaningful features from $\mathbf{x}$, inducing the intrinsic low-dimensional structure of patterns [27]. Linear and non-linear mappings are available, depending on the activation function (linear or sigmoid) of the neurons in the network, being similar to PCA in the former case. The usual training of neural networks, based on back-propagation, is very slow for autoencoders, given its usually high number of connections. Therefore, it is usual to pre-train the network using a Restricted Boltzmann Machine (RBM), a kind of neural network with binary neurons, without connections between hidden neurons, and then weights are tuned using back-propagation or a variant of simulated annealing.

2.3 Local nonlinear mapping

The methods presented in this section are non-linear mappings oriented to preserve properties which are valid only in small neighborhoods around the patterns, to they are called local non-linear mappings. The literature [9] has shown that some local methods can be considered as variants of KPCA using different kernels.

2.3.1 Locally linear embedding (LLE)

The LLE [93] creates a representation graph of patterns, but preserving the local structure by describing the original patterns $\mathbf{x}_i$ as a linear combination of their k nearest neighbors $\mathbf{x}_{ij}$, with $j = 1,\ldots,k$:

$$\mathbf{x}_i = \sum_{j=1}^{k} w_{ij}\mathbf{x}_{ij}, \qquad i = 1,\ldots,N \tag{2.40}$$

where w_{ij} is the weight of the j-th nearest neighbor of $\mathbf{x}_i$. This method calculates the $N \times k$-order weight matrix $\mathbf{W}$ from the original patterns $\mathbf{x}_i$. This description is invariant under rescaling, translation and rotation, so that LLE maps the patterns to $\mathbb{R}^d$ in such a way that it preserves the local geometry of the original patterns $\mathbf{x}_i$ in the manifold, i.e., to keep the same weights as in $\mathbb{R}^n$, so much as possible. Specifically, LLE calculates the low-dimensional mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ in order to minimize the following cost function:

$$\phi(\mathbf{Y}) = \sum_{i=1}^{N} \left| \mathbf{y}_i - \sum_{j=1}^{k} w_{ij}\mathbf{y}_{ij} \right|^2 = \mathbf{Y}^T(\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})\mathbf{Y} \tag{2.41}$$

To calculate $\mathbf{y}_i \in \mathbb{R}^d$, the LLE computes the eigenvectors of matrix $\mathbf{I}-\mathbf{W}$ associated to the smallest d non-zero eigenvalues. The LLE has been reported in the literature [114, 93] as a low-performing method with fails in the visualization of several popular data sets.

2.3.2 Hessian LLE (HLLE)

This method is a variant of LLE which uses Hessian instead of Laplacian operator [31]. Supposing that the high-dimensional patterns $\mathbf{x}$ lie on a manifold embedded in $\mathbb{R}^n$, the HLLE maps them into a low-dimensional space $\mathbb{R}^d$ which is locally isometric to that manifold. This space can be not convex, so HLLE is more general than Isomap. The HLLE performs an eigen analysis of the Hessian matrix ($\mathbf{H}$) which contains the means of the Hessian over all

the patterns [124] and describes the curviness of the manifold around the patterns. It starts by searching the k nearest neighbors of each pattern $\mathbf{x}_i$, assuming local linearity for the manifold over the neighborhood. Applying PCA on this set of neighbors and selecting the d principal eigenvectors we achieve a basis for the local tangent space at pattern $\mathbf{x}_i$, which is centered by substracting its average. In order to estimate the Hessian matrix $\mathbf{H}^i$ of the manifold at $\mathbf{x}_i$, a matrix $\mathbf{Z}^i$ with the d cross products of the d eigenvectors of the basis is created and orthonormalized. Then, $\mathbf{H}^i$ is estimated as the transpose of the last $d(d+1)/2$ columns of $\mathbf{Z}^i$. Using all the local Hessian matrices $\mathbf{H}^i$, with $i = 1,\ldots,N$, the global Hessian matrix $\mathscr{H}$ is created with elements $\mathscr{H}_{kl}$ calculated as:

$$\mathscr{H}_{kl} = \sum_i \sum_j H^i_{jk} H^i_{jl} \tag{2.42}$$

This matrix $\mathscr{H}$ measures the curviness of the manifold where the data lie in $\mathbb{R}^n$. The d smallest non-zero eigenvectors of $\mathscr{H}$ are selected as the N mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ which define the d-dimensional space minimizing the manifold curvature. There is no explicit mapping for out-of-sample patterns.

2.3.3 Neighborhood preserving embedding (NPE)

The NPE is a linear mapping that attempts to preserve the local structure of the manifold where patterns are supposed to lie, being related to LLE [50]. It starts by constructing an adjacency graph that represents the relationship between input patterns $\mathbf{x}_i \in \mathbb{R}^n$ using the k-nearest neighbor (directed graph) or ε-neighborhood (undirected graph) methods. Then, it computes the $N \times N$-order weight matrix $\mathbf{W}$ of the edges similarly to LLE, and it solves the following eigenproblem:

$$(\mathbf{X} - \bar{\mathbf{X}})^T (\mathbf{I} - \mathbf{W})^T (\mathbf{I} - \mathbf{W})(\mathbf{X} - \bar{\mathbf{X}})\mathbf{v} = \lambda (\mathbf{X} - \bar{\mathbf{X}})^T (\mathbf{X} - \bar{\mathbf{X}})\mathbf{v} \tag{2.43}$$

where $\mathbf{I}$ is the N-order identity matrix, the $n \times N$-order matrix $\mathbf{X}$ contains the original patterns $\mathbf{x} \in \mathbb{R}^n$ by columns, and all the columns of the $\bar{\mathbf{X}}$ matrix are equal, containing the means of the n inputs. Denoting by $\mathbf{A}$ the $n \times d$-order matrix with the d smallest non-zero eigenvalues $\mathbf{v} \in \mathbb{R}^n$ by columns, the $d \times N$-order matrix $\mathbf{Y}$ with the low-dimensional mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ by columns is given by:

$$\mathbf{Y} = \mathbf{A}^T (\mathbf{X} - \bar{\mathbf{X}}) \tag{2.44}$$

2.3.4 Local tangent space alignment (LTSA)

The LTSA [132] describes the local properties of patterns in $\mathbb{R}^n$ using the local tangent space Θ_i of each pattern $\mathbf{x}_i$, similarly to HLLE. Assuming that patterns lie on a manifold which is locally linear, there are two linear mappings, one from $\mathbf{x}_i$ and other from its low-dimensional version $\mathbf{y}_i \in \mathbb{R}^d$, both into Θ_i. The LTSA proceeds by aligning both mappings in order to construct the LTS from $\mathbb{R}^d$, searching simultaneously for $\mathbf{y}_i$ and for the mappings (or matrices) $\mathbf{L}_i$ from $\mathbf{y}_i$ to Θ_i. In the beginning, LTSA applies PCA on the set of k neighbors of $\mathbf{x}_i$, creating a mapping from this set to Θ_i. Then, it minimizes:

$$\phi(\mathbf{Y}) = \sum_{i=1}^{N} |\mathbf{J}_k \mathbf{y}_i - \mathbf{L}_i \Theta_i|^2 \tag{2.45}$$

where $\mathbf{J}_k$ is the centering matrix of size k, so LTSA minimizes the sum, over all the patterns, of the squared norms of the differences between the local tangent space in $\mathbb{R}^n$, given by $\mathbf{L}_i\Theta_i$, and the centered low-dimensional pattern $\mathbf{y}_i$. The mapped patterns in $\mathbb{R}^d$ are the d smallest non-zero eigenvectors of the matrix $\dfrac{\mathbf{B}+\mathbf{B}^T}{2}$, where $\mathbf{B}$ is an alignment matrix calculated through an iterative complex procedure. No explicit mapping is available for out-of-sample patterns.

2.3.5 Linear LTSA (LLTSA)

This method [131] also aims to preserve in $\mathbb{F}^d$ the local geometric structure of the patterns in the original space $\mathbb{R}^n$, minimizing the cost function of LTSA. Similarly to LPP and NPE, the LLTSA defines a neighborhood graph on the patterns $\mathbf{x}_i \in \mathbb{R}^n$ and estimates its local tangent space Θ_i to represent the local geometry, estimating the same matrix $\mathbf{B}$ as LTSA, and solving the following eigenproblem:

$$(\mathbf{X}-\bar{\mathbf{X}})^T \mathbf{B} (\mathbf{X}-\bar{\mathbf{X}})\mathbf{v} = \lambda (\mathbf{X}-\bar{\mathbf{X}})^T (\mathbf{X}-\bar{\mathbf{X}})\mathbf{v} \tag{2.46}$$

where $\bar{\mathbf{X}}$ is the average of the pattern matrix $\mathbf{X}$. Denoting as $\mathbf{V}$ the matrix with the d smallest non-zero eigenvalues of the previous eigenproblem, the matrix $\mathbf{Y}$ with the low-dimensional mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ is given by $\mathbf{Y} = (\mathbf{X}-\bar{\mathbf{X}})\mathbf{V}$.

2.3.6 Laplacian eigenmap

This method [7] maps the patterns from $\mathbb{R}^n$ to $\mathbb{R}^d$ by preserving the local properties, in this case the pairwise Euclidean distances between neighbors, of the manifold where the patterns

lie. The mapped patterns $\mathbf{y}_i$ are calculated in such a way that they minimize the distance between $\mathbf{y}_i$ and its nearest neighbors. In the cost function, distances are weighted decreasingly with the neighbor order. First, it creates a graph where each pattern $\mathbf{x}_i$ is connected to its k nearest neighbors. The edge weight between $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by the Gaussian kernel function:

$$w_{ij} = \mathrm{e}^{-|\delta \mathbf{x}_{ij}|^2/2\sigma^2}, \qquad i,j = 1,\ldots,N \tag{2.47}$$

where $\delta \mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. The cost function which is minimized to calculate $\mathbf{y}_i \in \mathbb{R}^d$ is given by:

$$\phi(\mathbf{Y}) = \sum_{i.j} w_{ij} |\delta \mathbf{y}_{ij}|^2 \tag{2.48}$$

where $\delta \mathbf{y}_{ij} = \mathbf{y}_i - \mathbf{y}_j$. The degree matrix $\mathbf{M}$ is calculated as a diagonal matrix whose elements are the row sums of $\mathbf{W}$, and the Laplacian matrix, similarly to analytical mechanics in Physics, is defined as $\mathbf{L} = \mathbf{M} - \mathbf{W}$. Therefore, the cost function ϕ can be written as $\phi = 2\mathbf{Y}^T\mathbf{L}\mathbf{Y}$, which is minimized by solving the eigenproblem:

$$\mathbf{L}\mathbf{v} = \lambda \mathbf{M}\mathbf{v} \tag{2.49}$$

The $N \times d$-order matrix $\mathbf{V}$ with the d eigenvectors associated to the smallest non-zero eigenvalues (by columns) defines the low-dimensional patterns $\mathbf{y}_i \in \mathbb{R}^d$, with $i = 1,\ldots,N$, being each mapped pattern $\mathbf{y}_i$ one row of $\mathbf{V}$. The Laplacian eigenmaps have been successfully applied in many field like clustering and face recognition.

2.3.7 Locality preserving projection (LPP)

This projection is a linear approximation to the nonlinear Laplacian eigenmap [51] that uses the Laplacian to project the data from $\mathbb{R}^n$ to $\mathbb{R}^d$, with $d \ll n$. Linear approaches are useful when a fast method is required to map out-of-sample patterns or when patterns must be reconstructed from $\mathbb{R}^d$. A nearest neighbor graph is created connecting each pattern to its k neighbors. The graph weights w_{ij}, calculated using eq. 2.47, compose the $N \times N$-order weight matrix $\mathbf{W}$. Analogously to Laplacian eigenmaps, the Laplacian is calculated as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where the N-order diagonal matrix $\mathbf{D}$ has elements D_{ii} equal to the column sums of $\mathbf{W}$. In order to calculate $\mathbf{y}_i \in \mathbb{R}^d$, the LPP solves the generalized eigenproblem:

$$\mathbf{X}\mathbf{L}\mathbf{X}^T\mathbf{v} = \lambda \mathbf{X}\mathbf{D}\mathbf{X}^T\mathbf{v} \tag{2.50}$$

where the $n \times N$-order matrix $\mathbf{X}$ stores the original patterns $\mathbf{x} \in \mathbb{R}^n$, after substracting their means, by columns. It can be shown that the $n \times d$-order matrix $\mathbf{A}$ which minimizes the Laplacian eigenmap cost function (eq. 2.48) is composed by the eigenvectors $\mathbf{v}_i \in \mathbb{R}^n$ associated to the d smallest non-zero eigenvalues, by columns. Finally, the $d \times N$-order matrix $\mathbf{Y}$ with the low-dimensional mapped patterns $\mathbf{y} \in \mathbb{R}^d$ is given by $\mathbf{Y} = \mathbf{A}^T\mathbf{X}$.

2.3.8 Maximum variance unfolding (MVU)

This method [118, 119] attempts to preserve in $\mathbb{R}^d$ the local data properties of $\mathbb{R}^n$ by requiring $|\mathbf{y}_i - \mathbf{y}_j|^2 = |\mathbf{x}_i - \mathbf{x}_j|^2$, with $i = 1, \ldots, N$ and $j > i$, using LLE to provide a starting solution. The idea of MVU is to represent the data in a lower dimension by pulling as much as possible the pairwise apart and maximize their total sum of distances. Thereby, MVU starts from a intermediate space $\mathbb{R}^p$, with $d < p < n$, generated using LLE, and creates a connected graph G whose arrows link each pattern $\mathbf{x}_i \in \mathbb{R}^p$ to its k-nearest neighborhoods (k is a free tunable hyper-parameter), keeping the distances between $\mathbf{x}_i$ and its neighbors in both spaces. In order to map the patterns to $\mathbb{R}^d$, the MVU reformulates the previously obtained results as a quadratic program to represent the patterns in $\mathbb{R}^d$, by solving the following optimization problem:

$$\begin{aligned} &\text{Maximize } \sum_{i,j \in G} |\delta\mathbf{y}_{ij}|^2 \text{ subject to:} \\ &1)\ |\delta\mathbf{y}_{ij}|^2 = |\delta\mathbf{x}_{ij}|^2 \qquad \forall i, j \in G \\ &2)\ \sum_i \mathbf{y}_i = \mathbf{0} \end{aligned} \tag{2.51}$$

where the last constraint means that the mapped patterns are centered. During this optimization the MVU may suffer from spurious local minima. The MVU does not provide an explicit mapping for new patterns.

2.3.9 Fast MVU

The FastMVU is similar to Isomap in that it describes a neighborhood graph on the original patterns and holds distances between pairs of patterns in the resulting graph. It works to "unfold" the data while maximizing the pairwise distances between patterns without changing the distances in the neighborhood graph (i.e., without distorting the local geometry of the data manifold). The resulting problem can be best solved using semi-definite programming [121].

The FastMVU begins building a neighborhood graph G, by connecting each pattern $\mathbf{x}_i$ to its k closest neighbors. After that, FastMVU squares the Euclidean distances between all patterns in order to increase the sum to its maximum, while keeping unchanged the distances between patterns within G. Therefore, FastMVU transforms the optimization problem of MVU (eq. 2.51) into a semi-definite programming problem by defining the $\mathbf{Y}$ matrix with the low-dimensional patterns $\mathbf{y}_i \in \mathbb{R}^d$, creating a semi-definite positive $N \times N$-order kernel matrix $\mathbf{K}$ with elements $K_{ij} = \mathbf{y}_i^T \mathbf{y}_j$, and maximizing trace($\mathbf{K}$) subject to [116]:

$$\begin{aligned} &1)\ K_{ii} - 2K_{ij} + K_{jj} = |\delta \mathbf{x}_{ij}|^2 \qquad i,j \in G \\ &2)\ \sum_{ij} K_{ij} = 0 \end{aligned}$$

The $\mathbf{Y}$ matrix with the low-dimensional patterns is achieved by $\mathbf{Y} = \mathbf{K}^{1/2}$, which always exists because $\mathbf{K}$ is semi-definite positive.

2.3.10 Landmark MVU

Since MVU solves a semi-definite programming problem, its computational cost is very high specially for N high, so Landmark MVU was proposed to accelerate this process [26, 58]. First, it applies MVU on a reduced subset of m patterns called landmarks, randomly selected, extending hereafter the dimensionality reduction to the whole pattern set and allowing online learning of new patterns. The landmark MVU supposes that the original patterns $\mathbf{x}_i$ lie on a d-dimensional manifold, which is spanned by the low-dimensional mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ corresponding to the m landmarks. Let $\mathbf{X}_1$, $\mathbf{Y}_1$ be the sets of original and mapped landmark patterns, respectively; $\mathbf{X}_2$ and $\mathbf{Y}_2$ be the sets of original and mapped non-landmark patterns, respectively. We have that $\mathbf{Y} = h(\mathbf{X}) = h(\mathbf{X}_1)\mathbf{Q}^T$, where h is the mapping from $\mathbb{R}^n$ to $\mathbb{R}^d$ and $\mathbf{Q}$ is a $N \times m$-order matrix. Considering that $\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2]$, that $\mathbf{Y} = h(\mathbf{X}) = [h(\mathbf{X}_1), h(\mathbf{X}_2)] = [\mathbf{Y}_1, \mathbf{Y}_2]$ and that $\mathbf{Q}^T = [\mathbf{Q}_1, \mathbf{Q}_2]$, we have:

$$[\mathbf{Y}_1, \mathbf{Y}_2] = [\mathbf{Y}_1\mathbf{Q}_1, \mathbf{Y}_1\mathbf{Q}_2] \tag{2.52}$$

so that $\mathbf{Q}_1 = \mathbf{I}$ and:

$$\mathbf{Y}_2 = \mathbf{Y}_1\mathbf{Q}_2 \tag{2.53}$$

Each pattern $\mathbf{x}_i \in \mathbf{X}_2$ can be approximately described as a linear combination of its k nearest neighbor patterns $\mathbf{x}_j$ with reconstruction weights $\mathbf{w}_{ij}$:

$$\mathbf{x}_i \approx \sum_{j \in \mathscr{N}_i} w_{ij}\mathbf{x}_j, \quad \forall \mathbf{x}_i \in \mathbf{X}_2 \tag{2.54}$$

where $\mathscr{N}_i$ denotes the neighborhood of $\mathbf{x}_i$. In matrix form, it can be written as:

$$\mathbf{X}_2 \approx \mathbf{X}_1\mathbf{W}_1 + \mathbf{X}_2\mathbf{W}_2 \tag{2.55}$$

being $\mathbf{W}_1$ and $\mathbf{W}_2$ the weight matrices for landmarks and non-landmarks, respectively. In order to keep the reconstruction weights in $\mathbb{R}^d$, it must be:

$$\mathbf{Y}_2 \approx \mathbf{Y}_1\mathbf{W}_1 + \mathbf{Y}_2\mathbf{W}_2 \Rightarrow \mathbf{Y}_2 = \mathbf{Y}_1\mathbf{W}_1(\mathbf{I}-\mathbf{W}_2)^+ \tag{2.56}$$

where $(\mathbf{I}-\mathbf{W}_2)^+ = (\mathbf{I}-\mathbf{W}_2)^T[(\mathbf{I}-\mathbf{W}_2)(\mathbf{I}-\mathbf{W}_2)^T]^{-1}$ is the generalized pseudo-inverse of $\mathbf{I}-\mathbf{W}_2$. Comparing eqs. 2.53 and 2.56, we see that:

$$\mathbf{Q}_2 = \mathbf{W}_1(\mathbf{I}-\mathbf{W}_2)^+ \tag{2.57}$$

Once $\mathbf{Q} = [\mathbf{I}, \mathbf{Q}_2]$ is known and denoting $\mathbf{K} = \mathbf{QLQ}^T$, linear programming techniques are used to find a matrix $\mathbf{L}$ such maximizes trace$(\mathbf{K})$ subject to $\sum_{i,j=1}^{N} K_{ij} = 0$, to $K_{jj} + K_{ll} - 2K_{jl} = |\delta\mathbf{x}_{jl}|$ for $j,l \in \mathscr{N}_i$ and $\mathbf{x}_i \in \mathbf{X}_1$ (original landmark patterns), being $\mathbf{L}$ semi-definite positive. Similarly to MVU, the matrix $\mathbf{Y}$ with low-dimensional mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ is given by $\mathbf{Y} = \mathbf{K}^{1/2}$.

2.3.11 Conformal eigenmap (CCA)

This method [99] starts by using LLE or Laplacian eigenmaps to map the patterns $\mathbf{x}_i$ from $\mathbb{R}^n$ to patterns $\mathbf{z}_i \in \mathbb{R}^t$, with $d < t < n$, and then it creates a conformal mapping to $\mathbb{R}^d$, i.e., a mapping which is maximally angle-preserving, so that:

$$\frac{|\delta\mathbf{z}_{hi}|^2}{|\delta\mathbf{x}_{hi}|^2} = \frac{|\delta\mathbf{z}_{ij}|^2}{|\delta\mathbf{x}_{ij}|^2} = \frac{|\delta\mathbf{z}_{hj}|^2}{|\delta\mathbf{x}_{hj}|^2} \tag{2.58}$$

Thus, the CCA mapping reduces the pattern dimension while preserving the angles between neighboring patterns. The conformality of the triangles at pattern $\mathbf{x}_h$ is given by:

$$D_h(s_h) = \sum_{ij} \eta_{hi}\eta_{ij}(|\delta\mathbf{z}_{ij}|^2 - s_h|\delta\mathbf{x}_{ij}|^2)^2 \tag{2.59}$$

where $\eta_{ij} = 1$ only if $\mathbf{x}_i$ and $\mathbf{x}_j$ are neighbors, and s_h corrects for scalings in the transformation from $\mathbb{R}^n$ to $\mathbb{R}^d$. The conformal eigenmap searches the linear mapping (associated to a matrix $\mathbf{L}$) from $\mathbf{z}_i \in \mathbb{R}^t$ to $\mathbf{y}_i = \mathbf{L}\mathbf{z}_i \in \mathbb{R}^d$ which maximizes:

$$\sum_h \sum_{ij} \eta_{hi} (|\mathbf{L}\delta\mathbf{z}_{ij}|^2 - s_h |\delta\mathbf{x}_{ij}|^2)^2 \tag{2.60}$$

subject to $\text{trace}(\mathbf{L}^T\mathbf{L}) = 1$, in order to avoid the trivial solution $\mathbf{L} = \mathbf{0}$. The CCA does not provide an explicit mapping for out-of-sample patterns.

2.3.12 Locally linear coordination (LLC)

The LLC [108] captures a manifold curved structure in order to map the patterns from $\mathbb{R}^n$ to $\mathbb{R}^d$ in two steps: 1) to compute a mixture of K local coordinate linear models (or dimensionality reducers) of the patterns using expectation maximization; and 2) to perform a global coordinate alignment of these reducers using a linear mapping from them to the low-dimensional patterns $\mathbf{y}_i \in \mathbb{R}^d$ that minimizes the cost of LLE. The k-th reducer maps to a dimension d_k and it has a responsibility r_{ik} for each pattern $i = 1, \ldots, N$, which represents the reliability of reducer k to represent $\mathbf{x}_i$ in $\mathbb{R}^{d_k}$, so that $\sum_k r_{ik} = 1$. Let also be $\mathbf{z}_{ik}$ the pattern mapped by the k-th reducer. The global alignment is done by applying a linear mapping given by the matrix $\mathbf{L}_k$ and an offset $\mathbf{l}_k$ for the k-th reducer, so that the mapped pattern $\mathbf{y}_i$ is given by:

$$\mathbf{y}_i = \sum_k r_{nk}(\mathbf{L}_k\mathbf{z}_{ik} + \mathbf{l}_k) = \sum_k \sum_{m=0}^{d_k} r_{ik} z_{jk}^m \mathbf{l}_k^m = \sum_j u_{ij}\mathbf{l}_j, \quad i = 1, \ldots, N \tag{2.61}$$

where $j = j(m,k)$, while $\mathbf{l}_k^m$ is the m-th column of matrix $\mathbf{L}_k$. Besides, z_{ik}^m is the m-th component of $\mathbf{z}_{ik}$ and $u_{ij} = r_{ik}\mathbf{z}_{ik}^m$. In matrix form, we have $\mathbf{Y} = \mathbf{UL}$. The whole process is illustrated in Figure 2.2. The global alignment requires to select the matrix $\mathbf{L}$ in order to minimize the LLE convex cost function:

$$\varepsilon(\mathbf{X}, \mathbf{W}) = \sum_i \left| \mathbf{x}_i - \sum_{m \in \mathcal{N}_i} w_{im}\mathbf{x}_m \right|^2 = \text{trace}\left[\mathbf{X}^T(\mathbf{I} - \mathbf{W}^T)(\mathbf{I} - \mathbf{W})\mathbf{X}\right] \tag{2.62}$$

where $\mathbf{X}$ is the matrix with the original input patterns, w_{im} the weight of the m-th neighbor of $\mathbf{x}_i$, subject to $\sum_m w_{im} = 1$, stored in matrix $\mathbf{W}$, and $\mathbf{I}$ is the $N \times N$-order identity matrix. Since the weights w_{im} summarize the local pattern geometry, the objective will be to minimize the same cost in $\mathbb{R}^d$:

$$\begin{aligned} &\text{Maximize } \phi(\mathbf{X}, \mathbf{W}) = \text{trace}\left[\mathbf{Y}^T(\mathbf{I} - \mathbf{W}^T)(\mathbf{I} - \mathbf{W})\mathbf{Y}\right] \text{ subject to:} \\ &1)\ \mathbf{1}^T\mathbf{Y} = \mathbf{0} \quad (\phi \text{ must be rotation and translation invariant}) \\ &2)\ \mathbf{Y}^T\mathbf{Y} = \mathbf{I} \quad (\phi \text{ must scale as } \mathbf{Y}) \end{aligned} \tag{2.63}$$

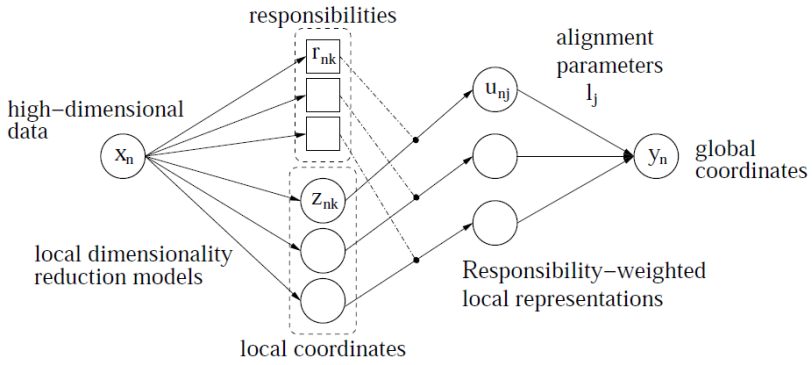


Figure 2.2: Global alignment of patterns **x** in LLC using the responsibility-weighted reductors (Figure extracted from [108]).

Replacing $\mathbf{Y} = \mathbf{UL}$ (remember that $\mathbf{U}$ is known, because $u_{ij} = r_{ik} z_{ik}^m$), this problem leads to the eigenvalue problem:

$$N\mathbf{U}^T(\mathbf{I}-\mathbf{W}^T)(\mathbf{I}-\mathbf{W})\mathbf{U}\mathbf{v} = \lambda \mathbf{U}^T\mathbf{U}\mathbf{v} \tag{2.64}$$

The $d+1$ smallest nonzero eigenvalues compose the columns of matrix $\mathbf{U}$, and the low-dimensional patterns $\mathbf{y}_i \in \mathbb{R}^d$ are achieved as $\mathbf{Y} = \mathbf{UL}$. The main weakness of LLC is that the fitting of mixture functions may find some local maxima in the log-likelihood function. Besides, it does not provide an explicit mapping for out-of-sample patterns.

2.3.13 Manifold charting

This nonlinear method [10] maps to $\mathbb{R}^d$ the original patterns in $\mathbb{R}^n$ by aligning a mixture of linear factor analyzers. First, the mixture of m analyzers is created using expectation-maximization, achieving $\mathbf{z}_{ij} \in \mathbb{R}^{d_j}$ for pattern i and analyzer j, and responsibilities r_{ij}. Then, it minimizes the following convex cost function, which measures the global difference between the mapped pattern $\mathbf{y}_i$ and the sum of patterns $\mathbf{y}_{ik}$ mapped by the analyzers, weighted by their responsibilities r_{ik}:

$$\phi(\mathbf{Y}) = \sum_{i=1}^{N}\sum_{j=1}^{m} r_{ij}|\mathbf{y}_i - \mathbf{y}_{ik}|^2 \tag{2.65}$$

where $\mathbf{y}_i = \sum_{k=1}^{m} r_{ik}\mathbf{y}_{ik}$. Rewriting $\phi(\mathbf{Y}) = \sum_{i=1}^{N}\sum_{j=1}^{m}\sum_{k=1}^{m} r_{ij}r_{ik}|\mathbf{y}_{ij} - \mathbf{y}_{ik}|^2$ and considering $\mathbf{Y} = \mathbf{UL}$, where $u_{ij} = r_{ij}[\mathbf{z}_{ij}, 1]$ similarly to LLC, the cost function $\phi(\mathbf{L})$ can be expressed in matrix form as:

$$\phi(\mathbf{L}) = \mathbf{L}^T(\mathbf{D} - \mathbf{U}^T\mathbf{U})\mathbf{L} \tag{2.66}$$

where $\mathbf{L}$ is the matrix which transforms the matrix $\mathbf{Z}$ with the patterns mapped by the analyzers into $\mathbf{Y}$. The matrix $\mathbf{D}$ is composed by the sum of covariances of the patterns $\mathbf{z}_{ij}$ mapped by the analyzers, weighted by the responsibilities r_{ij}. The columns of the $m \times d$-order matrix $\mathbf{L}$ which maximizes $\phi(\mathbf{L})$ are the d eigenvectors associated to the smallest non-zero eigenvalues of the following eigenvector problem:

$$(\mathbf{D} - \mathbf{U}^T\mathbf{U})\mathbf{v} = \lambda\mathbf{U}^T\mathbf{U}\mathbf{v} \tag{2.67}$$

Finally, denoting by $[\mathbf{Z}, \mathbf{1}]$ the $(m-1) \times (m-1)$-order matrix $\mathbf{Z}$ with a column vector with ones pasted on its right, the $d \times N$-order matrix $\mathbf{Y}$ with the mapped patterns $\mathbf{y}_i \in \mathbb{R}^d$ by columns is given by:

$$\mathbf{Y} = \mathbf{L}^T[\mathbf{Z}, \mathbf{1}]^T \tag{2.68}$$

2.3.14 Coordinated factor analysis (CFA)

This method [117] maps the patterns to $\mathbb{R}^d$ by globally aligning a mixture of factor analyzers (linear models), combining in the same process the creation and alignment of these separate linear models, so they can be tuned to optimize the global alignment. The CFA uses EM optimizing the normal log-likelihood function of the mixture of factor analizers, penalized by a negative Kullback-Leibler divergence which weights the similarity between the mixture and a Gaussian distribution, i.e., the agreement between the low- and high-dimension patterns $\mathbf{y}_i \in \mathbb{R}^d$ and $\mathbf{x}_i \in \mathbb{R}^n$, respectively. This penalized log-likelihood is given by:

$$\mathscr{L} = \sum_{i=1}^{N}\{\log p(\mathbf{x}_i) - D[q_i(\mathbf{y})\|p(\mathbf{y}|\mathbf{x}_i)]\}) \tag{2.69}$$

where $\log p(\mathbf{x}_i)$ is the log-likelihood of the mixture of the factor analyzers, $D(\cdot\|\cdot)$ is the Kullback-Leibler divergende and $q_i(\mathbf{y}) = \mathscr{N}(\mathbf{y} : \mathbf{y}_i, \Sigma_i)$ is a Gaussian distribution of mean $\mathbf{y}_i$ and covariance Σ_i. The optimization of $\mathscr{L}$ is complex, although CFA provides closed expressions for all the steps, but an out-of-sample extension is not available. The literature has reported performance limitations when many local maxima exist for $\mathscr{L}$.

2.4 Other supervised methods

Most of the previous methods, excepting LDA and GDA, do not use the class labels and therefore can be considered unsupervised from the point of view of classification. There are also three additional methods that have been specifically designed for classification tasks, so they use these labels and can be considered as supervised dimensionality reduction methods.

2.4.1 Neighborhood component analysis (NCA)

The NCA is a linear supervised method used for the visualization, classification patterns and dimensionality reductions problems [48]. It aims to learn the Mahalanobis metric distance associated to a $d \times n$-order transformation matrix $\mathbf{A}$ which maximizes the average leave-one-out (LOO) error of the k-nearest neighbor (KNN) classifier in the transformed space $\mathbb{R}^d$. This distance metric is calculated as:

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x}_j)^T(\mathbf{A}\mathbf{x}_i - \mathbf{A}\mathbf{x}_j) = \delta\mathbf{x}_{ij}^T\mathbf{A}^T\mathbf{A}\delta\mathbf{x}_{ij} = \delta\mathbf{x}_{ij}^T\mathbf{Q}\delta\mathbf{x}_{ij} \tag{2.70}$$

where $\mathbf{Q} = \mathbf{A}^T\mathbf{A}$. Since infinitesimal changes in $\mathbf{A}$ may lead to finite changes in the LOO error, this value is a discontinuous function of $\mathbf{A}$, so the NCA uses a stochastic neighbor assignment with a differentiable cost function instead of this error. For pattern $\mathbf{x}_i$, the probability p_{ij} of pattern $\mathbf{x}_j$ to be its neighbor is given by the softmax over Euclidean distance in the transformed space:

$$p_{ij} = \frac{e^{-|\mathbf{A}\delta\mathbf{x}_{ij}|^2}}{\sum_{k\neq i} e^{-|\mathbf{A}\delta\mathbf{x}_{ik}|^2}}, \quad p_{ii} = 0, \qquad i, j = 1, \ldots, N \tag{2.71}$$

Based on the stochastic selection rule, the probability p_i of the pattern $\mathbf{x}_i$ to be correctly classified can be computed as:

$$p_i = \sum_{j\in C_i} p_{ij} \tag{2.72}$$

where $C_i = \{j | c_i = c_j\}$ represents the set of indices of the patterns $\mathbf{x}_j$ that belong to the same class as $\mathbf{x}_i$. The NCA is seeks to maximize the expected number of patterns correctly classified under the scheme that produces the transformation matrix $\mathbf{A}$, which is given by:

$$\phi(\mathbf{A}) = \sum_{i=1}^{N}\sum_{j\in C_i} p_{ij} = \sum_i p_i(\mathbf{A}) \tag{2.73}$$

The gradient rule shown below is a result differentiating ϕ with respect to $\mathbf{A}$:

$$\frac{\partial \phi}{\partial \mathbf{A}} = 2\mathbf{A}\sum_{i=1}^{N}\left(p_i\sum_{k=1}^{N}p_{ik}\delta\mathbf{x}_{ik}\delta\mathbf{x}_{ik}^T - \sum_{j\in C_i}p_{ij}\delta\mathbf{x}_{ij}\delta\mathbf{x}_{ij}^T\right) \tag{2.74}$$

The NCA uses iterative gradient-base optimizer such as delta-bar-delta or conjugate gradients in order to optimize the cost function ϕ. Because this function is not convex, some care is needed during the training to avoid the local maxima. The low-dimensional pattern $\mathbf{y} \in \mathbb{R}^d$ is given by $\mathbf{y} = \mathbf{A}\mathbf{x}$.

2.4.2 Maximally collapsing metric learning (MCML)

The fundamentals of this method [46] are that a good metric distance for a given classification task should place patterns of the same class near each other, and far from patterns of other classes. The MCML generates such distance by optimizing a convex problem which tries to collapse each class in a single pattern placed far away from the patterns of other classes. This provides a compact low-dimensional representation of the original patterns. Given $d_{ij}^{\mathbf{A}} = \delta\mathbf{x}_{ij}^T\mathbf{A}\delta\mathbf{x}_{ij}$, we define a conditional distribution for $i \neq j$ as:

$$p^{\mathbf{A}}(j|i) = \frac{\mathrm{e}^{-d_{ij}^{\mathbf{A}}}}{\sum_{k\neq i}\mathrm{e}^{-d_{ij}^{\mathbf{A}}}} \tag{2.75}$$

The MCML searches for a positive semi-definite matrix $\mathbf{A}$ that minimizes the Kullback-Leibler divergence between $p^{\mathbf{A}}(j|i)$ and $p_0(j|i)$, defined as $p_0(j|i) = 1$ only if $\mathbf{y}_i = \mathbf{y}_j$ and $p_0(j|i) = 0$ otherwise. This divergence, adequately rewritten, is given by:

$$\phi(\mathbf{A}) = -\sum_{\mathbf{y}_i=\mathbf{y}_j}\log p(j|i) = \sum_{\mathbf{y}_i=\mathbf{y}_j}d_{ij}^{\mathbf{A}} + \sum_i \log\sum_{k\neq i}\mathrm{e}^{-d_{ik}^{\mathbf{A}}} \tag{2.76}$$

The minimization of $\phi(\mathbf{A})$, which is a convex problem, starts by initializing $\mathbf{A}$ to some random or heuristically selected positive semi-definite matrix, and iterates the following two-step process:

1. Update $\mathbf{A}$: $\mathbf{A}(t+1) = \mathbf{A}(t) - \varepsilon\nabla\phi\left[\mathbf{A}(t)\right]$, where $\nabla\phi\left[\mathbf{A}(t)\right] = \sum_{i,j}[p_0(j|i) - p(j|i)]\delta\mathbf{x}_{ji}\delta\mathbf{x}_{ji}^T$.
2. Calculate the eigen-decomposition of $\mathbf{A}$: $\mathbf{A}(t+1) = \sum_k \lambda_k\mathbf{u}_k\mathbf{u}_k^T$ and then set $\mathbf{A}(t+1) = \sum_k \max(\lambda_k, 0)\mathbf{u}_k\mathbf{u}_k^T$

The low-dimensional mapped pattern $\mathbf{y} \in \mathbb{R}^d$ is calculated as $\mathbf{y} = \mathbf{B}\mathbf{x}$, with $\mathbf{B} = \mathbf{C}^{1/2}$ and matrix $\mathbf{C}$ contains the first d columns of $\mathbf{A}$ with the eigenvectors associated to the d largest eigenvalues.

2.4.3 Large-margin nearest neighbor (LMNN)

The LMNN calculates a positive semi-definite matrix $\mathbf{M}$ for the Mahalanobis distance $d(\mathbf{x}_i, \mathbf{x}_j) = \delta\mathbf{x}_{ij}^T\mathbf{M}\delta\mathbf{x}_{ij}$ and the k-nearest neighbor classifier [120]. The objective is to minimize a cost function which weights two terms:

1. The pull term ε_{pull} penalizes large distances between $\mathbf{x}_i$ and the set $\mathcal{N}_i$ of its nearest neighbors $\mathbf{x}_j$, and is given by:

$$\varepsilon_{pull}(\mathbf{M}) = \sum_i \sum_{j \in \mathcal{N}_i} |\mathbf{M}\delta\mathbf{x}_{ij}|^2 \tag{2.77}$$

2. The push term ε_{push} penalizes small distances between patterns of different classes:

$$\varepsilon_{push}(\mathbf{M}) = \sum_i \sum_{j \in \mathcal{N}_i} \sum_{c_l \neq c_i} \left[1 + |\mathbf{M}\delta\mathbf{x}_{ij}|^2 - |\mathbf{M}\delta\mathbf{x}_{il}|^2\right]_+ \tag{2.78}$$

where c_i is the class of pattern $\mathbf{x}_i$ and $[z]_+ = \max(z, 0)$ is the usual hinge loss function. The optimization problem is to find a positive semi-definite matrix $\mathbf{M}$ that:

$$\begin{aligned}
&\text{Minimize } (1-\alpha)\varepsilon_{pull}(\mathbf{M}) + \alpha\varepsilon_{push}(\mathbf{M}) \text{ subject to:} \\
&1)\ \delta\mathbf{x}_{il}^T\mathbf{M}\delta\mathbf{x}_{il} - \delta\mathbf{x}_{ij}^T\mathbf{M}\delta\mathbf{x}_{ij} \geq 1 - \xi_{ijl} \\
&2)\ \xi_{ijl} \geq 0
\end{aligned}$$

where $0 \leq \alpha \leq 1$ is the regularization parameter, with recommended value $\alpha = 0.5$, and ξ_{ijl} are the slack variables, similarly to the Support Vector Machine (SVM). Similarly to MCML, the low-dimensional mapped pattern $\mathbf{y} \in \mathbb{R}^d$ is given by $\mathbf{y} = \mathbf{A}\mathbf{x}$, where the $d \times n$-order matrix $\mathbf{A}$ is composed by the d eigenvectors associated to the largest eigenvalues of $\mathbf{M}$, by rows.

2.5 Remarks

After describing the dimensionality reduction methods, we can compile the features of the different methods in the Table 2.1. From this table, some mappings included in the local non-linear group are linear versions of other local non-linear methods (e.g., LLTSA is the linear

Method	Features
Classical methods	
LDA	Supervised
GDA	Supervised, kernel, not explicit
PCA	Linear, unsupervised
KernelPCA	Unsupervised, non-linear, kernel
ProbPCA	Unsupervised, Expectation-Maximization (EM), Gaussian processes
GPLVM	Unsupervised, Gaussian processes, not-explicit
FA	Latent variables, EM
Global non-linear methods	
Sammon, MDS, SPE, SNE, SSNE, t-SNE	Preserve pairwise distance, not explicit
Isomap, Landmark Isomap	Preserve pairwise geodesic distance, neighborhood graph
Diffusion map	Preserves distance, diffusion kernel, neighborhood graph, not explicit
Multilayer autoencoder	Neural network
Local non-linear methods	
LLE	Linear, Preserves neighbor weights, neighbor graph
Hessian LLE	Preserves neighbor weights, not explicit
NPE	Linear approach to LLE, preserves neighbor weights
LTSA	Local tangent space, nearest neighbors, not explicit
LLTSA	Linear approach to LTSA
Laplacian eigenmap	Preserves pairwise distances, neighbor graph
LPP	Linear approach to Laplacian eigenmap, neighbor graph
MVU	Preserves pairwise distances, starts from LLE, not explicit
FastMVU	Preserves distances, neighbor graph
Landmark MVU	Starts from MVU
CCA	Neighbor angle preserving, starts from LLE or Laplacian eigenmap, not explicit
LLC	Linear, mixture of linear models, global alignment, not explicit
Manifold charting	Mixture of linear factor analyzers
CFA	Mixture of linear factor analyzers, not explicit
Other supervised methods	
NCA, MCML, LMNN	Linear, supervised, Mahalanobis metric

Table 2.1: Main features of the dimensionality reduction methods. Those methods which do not provide an explicit mapping for out-of-sample patterns are labeled as "non-explicit".

approach to LTSA, which is non-linear), but we included them here in order to group together those methods which are related. We can see that most methods preserve distances, eigher in a global or local style. Many methods use nearest neighbors (e.g. Hessian LLE and LTSA), sometimes with a neighbor graph (e.g. Isomap, Landmark Isomap, Diffusion map, LLE, LPP and FastMVU). Some methods use Gaussian processes (e.g. ProbPCA and GPLVM) or a mixture of linear models or factor analyzers (e.g. LLC, Manifold charting, CFA), and some others use Expertation-Maximization (e.g. ProbPCA, FA and CFA). Besides, most methods are unsupervised, excepting LDA, GDA and "other supervised methods". Finally, the table

also identifies as “non-explicit” those methods which do not provide an explicit mapping for out-of-sample patterns and therefore require an interpolation method in order to map (or to reduce the dimensionality of) new test patterns, not used during the execution of the dimensionality reduction method.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 3

PROPOSED 2D MAPPINGS

In the current chapter we present our proposals to project (or to map, or to reduce the dimensionality of the) patterns from $\mathbb{R}^n$ to $\mathbb{R}^2$ in order to build a 2D map for classification tasks, so that in the following we will consider $d = 2$. This work and its experimental validation, which is described in chapter 4, have been published in the journal paper [2]. Let us be $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ as the input pattern in the original space, and let $\mathbf{y} \in \mathbb{R}^2$ be the output in the 2-dimensional space. We search for an unknown function $\Phi : \mathbb{R}^n \to \mathbb{R}^2$ which maps $\mathbf{x}$ to $\mathbf{y}$ so that $\mathbf{y} = \Phi(\mathbf{x})$. We will consider that function Φ can be written as a polynomial function of $\mathbf{x}$, since the monomials of any degree of the components of $\mathbf{x}$ are a basis of the function space. We tried several alternative mappings, given by functions of increasing complexity, given by $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{z}(\mathbf{x})$, where the matrix $\mathbf{A}$ must be calculated to maximize some measure of class overlap, and $\mathbf{z}(\mathbf{x})$ is a vector polynomial function of $\mathbf{x}$ (see below). The order of matrix $\mathbf{A}$ and the function $\mathbf{z}(\mathbf{x})$ are different for each case. The mappings developed are the following:

1. **Linear**: $\mathbf{z}(\mathbf{x}) = \mathbf{x}$, so that $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{x}$, being $\mathbf{A}$ a matrix of order $2 \times n$. This is the simplest mapping and the fastest one to compute.

2. **Full quadratic**: $\mathbf{z}(\mathbf{x}) = (x_1, \ldots, x_n, x_1^2, \ldots, x_1x_n, x_2^2, \ldots, x_2x_n, \ldots, x_{n-1}^2, x_{n-1}x_n, x_n^2) \in \mathbb{R}^p$, being $p = \dfrac{n(n+1)}{2}$, is a quadratic vector function with linear terms $x_i, i = 1, \ldots, n$, and also with 2nd degree crossed products x_ix_j with $i = 1, \ldots, n$ and $j \geq i$. In this case, the matrix $\mathbf{A}$ is of order $2 \times p$, a size which is larger than the linear mapping, so it requires more computations and time.

3. **Polynomial**: $\mathbf{z}(\mathbf{x}) = (x_1,\ldots,x_n,\frac{x_1^2}{2!},\ldots,\frac{x_n^2}{2!},\ldots,\frac{x_1^D}{D!},\ldots,\frac{x_n^D}{D!}) \in \mathbb{R}^{nD}$ is a polynomial of degree D without crossed products (only with pure terms) of the form $\frac{x_i^d}{d!}$, with $i = 1\ldots,n$, $d = 1,\ldots,D$,divided by $d!$ to avoid scale problems. We consider $D = 2,\ldots,7$ (six values). The matrix $\mathbf{A}$ is of order $2 \times p$, with $p = nD$, so its size is also high and it may require much time, depending on the order n of the original space and on the degree D of the polynomial.

Note that the previous polynomials do not include an offset term because, as we will see, the calculations imply differences which lead to cancel this offset. The mapping Φ is defined by a matrix $\mathbf{A}$ of size $2 \times p$, which is different in each version, which must be calculated from the training patterns. The value p is the dimension of the space expanded from the original space, being $p = n$ for the linear, $p = (n(n+1))/2$ for the full quadratic and $p = nD$ for the polynomial ■ function. Since our objective is to map a classification problem to $\mathbb{R}^2$, we are interested in calculating $\mathbf{A}$ so that it minimizes the overlap in $\mathbb{R}^2$ between the patterns of the different classes. We will considered three overlap measures defined in the literature, described in the following subsections. Let us denote $\delta\mathbf{z}_{ij} = \mathbf{z}_i - \mathbf{z}_j$, for $i, j = 1,\ldots,N$, while $\delta z_{ij}^k = z_i^k - z_j^k$ for $k = 1,\ldots,p$, and $\delta\mathbf{z}_{ij} = [\delta z_{ij}^1,\ldots,\delta z_{ij}^p]^T$, where the superscript ^{T} denotes matrix transposition, and p is the number of columns of matrix $\mathbf{A}$, i.e., the dimension of the expanded space. Note also that:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \ldots & a_{1p} \\ a_{21} & \ldots & a_{2p} \end{bmatrix} \tag{3.1}$$

and that consequently $|\mathbf{A}\delta\mathbf{z}_{ij}|^2 = (\mathbf{A}\delta\mathbf{z}_{ij})^T\mathbf{A}\delta\mathbf{z}_{ij} = (\delta\mathbf{z})_{ij}^T\mathbf{A}^T\mathbf{A}\delta\mathbf{z}_{ij}$.

3.1 Thornton's separability index

The *Thornton separability index* [110] or *geometric separability* is defined as the proportion of patterns and its nearest neighbor they belong to the same class. Its computation requires to find the nearest neighbor of each training pattern in $\mathbb{R}^2$, which requires to use the minimum function. Unfortunately, this function is not differentiable, so that standard calculus can not be used in order to maximize this index and to find the optimal mapping Φ. Therefore, we will limit to calculate, for each pattern $\mathbf{x}_i \in \mathbb{R}^n$, its nearest neighbor $\mathbf{x}_{j(i)}$, and then we will maximize the sum of squared distances in $\mathbb{R}^2$ between $\mathbf{y}_i$ and $\mathbf{y}_{j(i)}$ when their class labels c_i

and $c_{j(i)}$ differ. Let be $M = \{i = 1,\ldots,N : c_i \neq c_{j(i)}\}$, i.e., the set of training patterns whose nearest neighbor has a class label different to the training pattern. Using that $\mathbf{y} = \Phi(\mathbf{x}) = \mathbf{A}\mathbf{z}(\mathbf{x})$, this overlap measure, which we will denote as ψ_T, can be calculated as:

$$\psi_T = \sum_{i \in M} |\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_{j(i)})|^2 \tag{3.2}$$

Replacing $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{z}(\mathbf{x})$ in eq. 3.2 and denoting $\delta\mathbf{z}_i = \mathbf{z}_i - \mathbf{z}_{j(i)}$ we achieve:

$$\psi_T = \sum_{i \in M} |\mathbf{A}\mathbf{z}_i - \mathbf{A}\mathbf{z}_j|^2 = \sum_{i \in M} |\mathbf{A}(\mathbf{z}_i - \mathbf{z}_j)|^2 = \sum_{i \in M} |\mathbf{A}\delta\mathbf{z}_i|^2 \tag{3.3}$$

Substituting the value of $|\mathbf{A}\delta\mathbf{z}_i|^2$ in eq. 3.3 we get:

$$\psi_T = \sum_{i \in M} \delta\mathbf{z}_i{}^T \, \mathbf{A}^T \, \mathbf{A}\delta\mathbf{z}_i \tag{3.4}$$

In order to calculate ψ_T, we can develop $\delta\mathbf{z}_i^T\mathbf{A}^T$ as follows:

$$\delta\mathbf{z}_i{}^T \, \mathbf{A}^T = \left[\delta z_i^1, \ldots, \delta z_i^p\right] \begin{bmatrix} a_{11} & a_{21} \\ \vdots & \vdots \\ a_{1p} & a_{2p} \end{bmatrix} = \left[\delta z_i^1 a_{11} + \cdots + \delta z_i^p a_{1p}, \delta z_i^1 a_{21} + \cdots + \delta z_i^p a_{2p}\right] =$$

$$= \left[\sum_{k=1}^{p} \delta z_i^k a_{1k}, \sum_{k=1}^{p} \delta z_i^k a_{2k}\right]$$

On the other hand:

$$\mathbf{A}\delta\mathbf{z}_i = \begin{bmatrix} a_{11} & \ldots & a_{1p} \\ a_{21} & \ldots & a_{2p} \end{bmatrix} \begin{bmatrix} \delta z_i^1 \\ \vdots \\ \delta z_i^p \end{bmatrix} = \begin{bmatrix} a_{11}\delta z_i^1 + \ldots & a_{1p}\delta z_i^p \\ a_{21}\delta z_i^1 + \ldots & a_{2p}\delta z_i^p \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{p} a_{1k}\delta z_i^k \\ \sum_{k=1}^{p} a_{2k}\delta z_i^k \end{bmatrix}$$

Therefore, we have:

$$\delta\mathbf{z}_i{}^T\mathbf{A}^T\mathbf{A}\delta\mathbf{z}_i = \left[\sum_{k=1}^{p} \delta z_i^k a_{1k} \quad \sum_{k=1}^{p} \delta z_i^k a_{2k}\right] \begin{bmatrix} \sum_{k=1}^{p} a_{1k}\delta z_i^k \\ \sum_{k=1}^{p} a_{2k}\delta z_i^k \end{bmatrix} =$$

$$= \left[\left(\sum_{k=1}^{p} \delta z_i^k a_{1k}\right)^2 + \left(\sum_{k=1}^{p} \delta z_i^k a_{2k}\right)^2\right] = \sum_{l=1}^{2} \left(\sum_{k=1}^{p} \delta z_i^k a_{lk}\right)^2$$

So that $\psi_T(\mathbf{A})$ can be expanded as:

$$\psi_T(\mathbf{A}) = \sum_{i\in M} \delta\mathbf{z}_i^T \mathbf{A}^T \mathbf{A} \delta\mathbf{z}_i = \sum_{i\in M} \sum_{l=1}^{2} \left(\sum_{k=1}^{p} \delta z_i^k a_{lk} \right)^2 \tag{3.5}$$

We want to calculate the matrix $\mathbf{A}$ which minimizes the overlap ψ_T. However, ψ_T depends on the absolute value of the elements of matrix $\mathbf{A}$, so that for $\mathbf{A}' = \alpha\mathbf{A}$, we see that $\psi_T(\mathbf{A}') = \alpha^2 \psi_T(\mathbf{A})$. Therefore, in order to find a minimum of $\psi_T(\mathbf{A})$ we need to introduce a constraint on the values of the elements of matrix $\mathbf{A}$: in our case, we will require that the sum of squared elements, which is the squared norm, must be 1, so that:

$$\sum_{i=1}^{2} \sum_{k=1}^{p} a_{ik}^2 = |\mathbf{A}|^2 = 1 \tag{3.6}$$

Now we can use the method of Lagrange multipliers[1] to maximize ψ_T subject to $|\mathbf{A}|^2 = 1$. Denoting by λ the Lagrange multiplier associated to the constraint, the Lagrange function $\mathscr{L}_T(\mathbf{A}, \lambda)$ is given by:

$$\mathscr{L}_T(\mathbf{A}, \lambda) = \sum_{i\in M} \sum_{l=1}^{2} \left(\sum_{k=1}^{p} \delta z_i^k a_{lk} \right)^2 - \lambda \left(\sum_{l=1}^{2} \sum_{k=1}^{p} a_{lk}^2 - 1 \right) \tag{3.7}$$

Deriving with respect to a_{rs}, with $r = 1, 2$, and $s = 1, \ldots, p$, and equaling to zero we have:

$$\frac{\partial \mathscr{L}}{\partial a_{rs}} = 2 \sum_{i\in M} \left(\sum_{k=1}^{p} \delta z_i^k a_{rk} \right) \delta z_i^s - 2\lambda a_{rs} = 0, \quad r = 1, 2; s = 1, \ldots, p \tag{3.8}$$

Note that the derivative with respect to a_{rs} of the sum over l is only non-zero for $l = r$. Ordering the sumations on the first term and dividing the whole equation by 2 we achieve:

$$\sum_{k=1}^{p} \left(\sum_{i\in M} \delta z_i^k \delta z_i^s \right) a_{rk} - \lambda a_{rs} = 0, \quad r = 1, 2; s = 1, \ldots, p \tag{3.9}$$

Let us define the $p \times p$-order matrix $\mathbf{B}_T$ as:

$$\mathbf{B}_T = \sum_{i\in M} \delta\mathbf{z}_i \delta\mathbf{z}_i^T \tag{3.10}$$

so that its element b_{uv}, with $u, v = 1, \ldots, p$, is given by:

[1] https://en.wikipedia.org/wiki/Lagrange_multiplier

$$b_{uv} = \sum_{i \in M} \delta z_i^u \delta z_i^v, \quad u,v = 1,\ldots,p \tag{3.11}$$

Note that $\mathbf{B}_T$ is symmetric. It is easy to show that:

$$(\mathbf{B}_T - \lambda \mathbf{I}_p)^T \mathbf{A}^T = \mathbf{0}_p \tag{3.12}$$

is the matrix form of equations 3.9, where $\mathbf{I}_p$ is the identity matrix of order p and $\mathbf{0}_p$ is the zero $p \times 2$-order matrix. The element u,v of this matrix equation, for $u,v = 1,\ldots,p$, is:

$$\left(\begin{bmatrix} \sum_{i \in M} \delta z_i^1 \delta z_i^1 & \ldots & \sum_{i \in M} \delta z_i^p \delta z_i^1 \\ \ldots & \ldots & \ldots \\ \sum_{i \in M} \delta z_i^1 \delta z_i^1 & \ldots & \sum_{i \in M} \delta z_i^p \delta z_i^1 \end{bmatrix} - \lambda \begin{bmatrix} 1 & \ldots & 0 \\ \ldots & \ldots & \ldots \\ 0 & \ldots & 1 \end{bmatrix} \right) \begin{bmatrix} a_{11} & a_{21} \\ \ldots & \ldots \\ a_{1p} & a_{2p} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ \ldots & \ldots \\ 0 & 0 \end{bmatrix} \tag{3.13}$$

Developing the elements of this matrix equation for $r = 1,2$, $s = 1,\ldots,p$, we achieve:

$$\sum_{k=1}^{p} \left(\sum_{i \in M} \delta z_i^k \delta z_i^s \right) a_{rk} - \lambda a_{rs} = 0, \quad r = 1,2; s = 1,\ldots,p \tag{3.14}$$

which is equal to eq. 3.9. Since both $\mathbf{B}_T$ and $\mathbf{I}_p$ are symetric, they are equal to their transposes, so that from eq. 3.12 we achieve:

$$(\mathbf{B}_T - \lambda \mathbf{I}_p)\mathbf{A}^T = \mathbf{0}_p \tag{3.15}$$

This is an example of eigenvalue problem. It is well-known from the literature that, in order to find the solution $\mathbf{A}$ to this minimization problem, the two rows of matrix $\mathbf{A}$ must be the eigenvectors corresponding to the largest eigenvalues of matrix $\mathbf{B}_T$. Therefore, in order to calculate $\mathbf{A}$ which provides the lowest ψ_T subject to $|\mathbf{A}^2| = 1$ we just have to calculate the matrix $\mathbf{B}_T$ and the rows of $\mathbf{A}$ must be its two main eigenvectors.

3.2 Direct class Separability (DS)

This measure [82], which will be denoted as ψ_{DS} is calculated as the difference between the sum of distances between patterns of different classes (inter-class distances D_B) and the sum

of the distances of patterns within the same class (intra-class distances D_W). Denoting by C the number of classes, we have:

$$D_B = \sum_{c=1}^{C} \sum_{i \in c} \sum_{j \notin c} |\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)|^2, \quad D_W = \sum_{c=1}^{C} \sum_{i \in c} \sum_{j \in c} |\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)|^2$$

$$\psi_{DS} = \sum_{c=1}^{C} \sum_{i \in c} \left[\sum_{j \notin c} |\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)|^2 - \sum_{j \in c} |\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_j)|^2 \right] \tag{3.16}$$

We are going to substitute $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{z}(\mathbf{x})$ in 3.16 and we achieve:

$$\psi_{DS} = \sum_{c=1}^{C} \sum_{i \in c} \left[\sum_{j \notin c} |\mathbf{A}\mathbf{z}(\mathbf{x}_i) - \mathbf{A}\mathbf{z}(\mathbf{x}_j)|^2 - \sum_{j \in c} |\mathbf{A}\mathbf{z}(\mathbf{x}_i) - \mathbf{A}\mathbf{z}(\mathbf{x}_j)|^2 \right] \tag{3.17}$$

By choosing $\mathbf{A}$ as common factor, and simplifying ψ_{DS} the we achieve :

$$\begin{aligned} \psi_{DS} &= \sum_{c=1}^{C} \sum_{i \in c} \left[\sum_{j \notin c} |\mathbf{A}(\mathbf{z}(\mathbf{x}_i) - \mathbf{z}(\mathbf{x}_j))|^2 - \sum_{j \in c} |\mathbf{A}\left(\mathbf{z}(\mathbf{x}_i) - \mathbf{z}(\mathbf{x}_j)\right)|^2 \right] = \\ &= \sum_{c=1}^{C} \sum_{i \in c} \left[\sum_{j \notin c} |\mathbf{A}\delta\mathbf{z}_{ij}|^2 - \sum_{j \in c} |\mathbf{A}\delta\mathbf{z}_{ij}|^2 \right] = \\ &= \sum_{c=1}^{C} \sum_{i \in c} \left[\sum_{j \notin c} \left(\delta\mathbf{z}_{ij}^T \mathbf{A}^T \mathbf{A}\delta\mathbf{z}_{ij}\right) - \sum_{j \in c} \left(\delta\mathbf{z}_{ij}^T \mathbf{A}^T \mathbf{A}\delta\mathbf{z}_{ij}\right) \right] \end{aligned} \tag{3.18}$$

Remember that $\mathbf{z}_i = \mathbf{z}(\mathbf{x}_i)$ and analogous for $\mathbf{z}_j$, and that $\delta\mathbf{z}_{ij} = \mathbf{z}_i - \mathbf{z}_j$. Equation 3.18 has inside the brackets two similar terms, one relative to patterns of class c (with negative sign), and other relative to patterns which do not belong to class c (with positive sign). This equation can be simplified by adding a flag $\beta_{jc} = -1$ for patterns j of class c and $\beta_{jc} = +1$ for patterns j which do not belong to class c.

$$\psi_{DS} = \sum_{c=1}^{C} \sum_{j=1}^{N} \beta_{jc} \sum_{i \in c} \delta\mathbf{z}_{ij}^T \, \mathbf{A}^T \, \mathbf{A}\delta\mathbf{z}_{ij} \tag{3.19}$$

Replacing the value of $\delta\mathbf{z}_{ij}^T \mathbf{A}^T \mathbf{A}\delta\mathbf{z}_{ij}$ in eq. 3.5, we achieve:

$$\psi_{DS} = \sum_{c=1}^{C} \sum_{j=1}^{N} \beta_{jc} \sum_{i \in c} \sum_{l=1}^{2} \left(\sum_{k=1}^{p} \delta z_{ij}^k a_{lk} \right)^2 \tag{3.20}$$

Similarly to the previous section, in order to minimize ψ_{DS} we must introduce a constraint on the norm of matrix $\mathbf{A}$, otherwise the value of ψ_T will depend on scale factors of the matrix

elements. Therefore, we want to minimize ψ_{DS} subject to the constraint $|\mathbf{A}|^2 = 1$, using the Lagrange multipliers method. The Lagrange function is given by:

$$\mathscr{L}_{DS} = \sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}\sum_{l=1}^{2}\left(\sum_{k=1}^{p}\delta z_{ij}^{k}a_{lk}\right)^2 - \lambda\left(\sum_{l=1}^{2}\sum_{k=1}^{p}a_{lk}^2 - 1\right) \tag{3.21}$$

Deriving $\mathscr{L}_{DS}$ with respect to a_{rs} (note that the derivative of the sum over l is non-zero only for $l = r$) and equaling to zero we achieve:

$$\frac{\partial \mathscr{L}_{DS}}{\partial a_{rs}} = \sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}2\left(\sum_{k=1}^{p}a_{rk}\delta z_{ij}^{k}\right)\delta z_{ij}^{s} - 2\lambda a_{rs} =$$

$$= 2\sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}\delta z_{ij}^{s}\sum_{k=1}^{p}a_{rk}\delta z_{ij}^{k} - 2\lambda a_{rs} = 0, \quad r = 1,2; s = 1,\ldots,p \tag{3.22}$$

Dividing by 2 and ordering the sums in the first term we achieve:

$$\sum_{k=1}^{p}\left(\sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}\delta z_{ij}^{s}\delta z_{ij}^{k}\right)a_{rk} - \lambda a_{rs} = 0, \quad r = 1,2; s = 1,\ldots,p \tag{3.23}$$

Let us define b_{sk} as:

$$b_{sk} = \sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}\delta z_{ij}^{s}\delta z_{ij}^{k}, \quad s,k = 1,\ldots,p \tag{3.24}$$

Note that $b_{sk} = b_{ks}$. It is easy to show that b_{sk} is the (s,k)-th element of the $p \times p$-order symmetric matrix $\mathbf{B}_{DS}$ defined as:

$$\mathbf{B}_{DS} = \sum_{c=1}^{C}\sum_{j=1}^{N}\beta_{jc}\sum_{i\in c}\delta \mathbf{z}_{ij}\delta \mathbf{z}_{ij}^{T} \tag{3.25}$$

Besides, eq. 3.23 can be written as:

$$\sum_{k=1}^{p}b_{sk}a_{rk} - \lambda a_{rs} = 0, \quad r = 1,2; s = 1,\ldots,p \tag{3.26}$$

Similarly to eq. 3.12 in the previous section, these equations can be written in matrix form as:

$$(\mathbf{B}_{DS} - \lambda \mathbf{I}_p)^T\mathbf{A}^T = \mathbf{0} \tag{3.27}$$

where $\mathbf{I}_p$ is the $p \times p$-order identity matrix and $\mathbf{0}$ is the $p \times 2$-order zero matrix. Given that $\mathbf{B}_{DS}$ and $\mathbf{I}_p$ are symmetric, we achieve:

$$(\mathbf{B}_{DS} - \lambda \mathbf{I}_p)\mathbf{A}^T = \mathbf{0} \tag{3.28}$$

We achieve again an eigenvalue problem, which is easily solved by calculating the eigenvalues of matrix $\mathbf{B}_{DS}$. Specifically, to find $\mathbf{A}$ that minimizes ψ_{DS} subject to $|\mathbf{A}|^2 = 1$ we must select the two main eigenvectors of $\mathbf{B}_{DS}$.

3.3 J-index

This measure [34] is defined as the ratio between the norms of the between-class scatter $\mathbf{S}_B$ and the within scatter $\mathbf{S}_W$ matrices, defined by:

$$\begin{aligned} \mathbf{S}_B &= \sum_{c=1}^{C} \alpha_c \left[\Phi(\mathbf{m}_c) - \Phi(\mathbf{m})\right] \left[\Phi(\mathbf{m}_c) - \Phi(\ \mathbf{m})\right]^T \\ \mathbf{S}_W &= \sum_{c=1}^{C} \alpha_c \sum_{i \in c} \left[\ \Phi(\mathbf{x}_i) - \ \Phi(\mathbf{m}_c)\right] \left[\ \Phi(\mathbf{x}_i) - \ \Phi(\mathbf{m}_c)\right]^T \end{aligned} \tag{3.29}$$

Where $\alpha_c = n_c/(N-1)$. Here, n_c is the number of patterns of class c; $\mathbf{m}_c$ is the mean of patterns of class c; and $\mathbf{m} = \dfrac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$ is the mean of the whole training set. This J-index is optimized for $\mathbf{z}(\mathbf{x}) = \mathbf{x}$ by the well-known Linear Discriminant Analysis (LDA). Using that $\Phi(\mathbf{x}) = \mathbf{A}\mathbf{z}(\mathbf{x})$, and denoting $\delta\mathbf{z}_c = \mathbf{z}(\mathbf{m}_c) - \mathbf{z}(\mathbf{m})$ and $\delta\mathbf{z}_{ic} = \mathbf{z}(\mathbf{x}_i) - \mathbf{z}(\mathbf{m}_c)$, we achieve:

$$\begin{aligned} \mathbf{S}_B &= \sum_{c=1}^{C} \alpha_c \mathbf{A}\delta\mathbf{z}_c \delta\mathbf{z}_c^T \mathbf{A}^T = \mathbf{A}\mathbf{T}_B\mathbf{A}^T, \quad \mathbf{T}_B = \sum_{c=1}^{C} \alpha_c \delta\mathbf{z}_c \delta\mathbf{z}_c^T \\ \mathbf{S}_W &= \sum_{c=1}^{C} \alpha_c \sum_{i \in c} \mathbf{A}\delta\mathbf{z}_{ic} \delta\mathbf{z}_{ic}^T \mathbf{A}^T = \mathbf{A}\mathbf{T}_W\mathbf{A}^T, \quad \mathbf{T}_W = \sum_{c=1}^{C} \alpha_c \sum_{i \in c} \delta\mathbf{z}_{ic} \delta\mathbf{z}_{ic}^T \end{aligned} \tag{3.30}$$

Note that $\mathbf{T}_B$ and $\mathbf{T}_W$ are two matrices of order $p \times p$. In order to maximize J, the matrix $\mathbf{A}$ must be selected in such a way that:

$$\mathbf{A} = \arg\max_{\mathbf{A}} \frac{|\ \mathbf{S}_B|}{|\ \mathbf{S}_W|} = \arg\max_{\mathbf{A}} \frac{|\ \mathbf{A}\ \mathbf{T}_B\ \mathbf{A}^T|}{|\ \mathbf{A}\ \mathbf{T}_W\ \mathbf{A}^T|} \tag{3.31}$$

The standard literature about LDA (see e.g. [34]) explains that the solution of this optimization problem is the same as the generalized eigenvalue problem $(\mathbf{T}_B - \lambda\mathbf{T}_W)\mathbf{A}^T = \mathbf{0}$. The

two rows of the matrix $\mathbf{A}$ which is the solution to this eigenvalue problem are the generalized eigenvectors corresponding to the two largest eigenvalues (in absolute value) of the matrix $\mathbf{T}_W^{-1}\mathbf{T}_B$, so the matrix $\mathbf{T}_W$ must be non-singular in order to achieve a valid solution. Using a linear Φ mapping, we achieve to the well-known LDA classifier. On the other hand, a full quadratic or polynomial mapping (see the previous subsection) are special cases of Kernel Discriminant Analysis (KDA). However, in these cases the covariance matrix $\mathbf{T}_W$ is singular, so the calculation of $\mathbf{T}_W^{-1}$ to solve the generalized eigenvalue problem is ill-conditioned. This drawback can be solved [77] using $\mathbf{T}_W + \mathbf{I}$ instead of $\mathbf{T}_W$. Therefore, the rows of the matrix $\mathbf{A}$ which maximizes J are the two main eigenvectors of the matrix:

$$\mathbf{B}_{DA} = (\mathbf{T}_W + \alpha\mathbf{I})^{-1}\mathbf{T}_B \tag{3.32}$$

with $\alpha = 0$ (resp. $\alpha = 1$) for linear (resp. quadratic and polynomial) kernel. The combination of J index as overlap measure and the different mappings will be denoted as LDA (DA with linear mapping), QKDA (discriminant analysis with full quadratic kernel mapping) and PKDA2,..., PKDA7 (DA with polynomial kernel mapping without crossed products, varying the degree D from 2 to 7).

3.4 Class Mean Distance

This overlap measure, denoted as ψ_{CD} is proposed by us [2] and defined as the sum of the squared distances between class means in $\mathbb{R}^2$ (distances are squared in order to achieve simpler matrix expressions). The maximization of this measure is intended to make classes maximally separated in $\mathbb{R}^2$, because the sum of squared distances between the class means should be high. The value of ψ_{CD} can be computed as:

$$\psi_{CD} = \sum_{c=1}^{C}\sum_{i\neq c} |\Phi(\mathbf{m}_c) - \Phi(\mathbf{m}_i)|^2 \tag{3.33}$$

where $\mathbf{m}_c$ and $\mathbf{m}_i$ are the class means (in $\mathbb{R}^n$) of classes c and i respectively. In order to calculate ψ_{CD}, we are going to replace $\Phi(\mathbf{m}_c) = \mathbf{A}\mathbf{z}(\mathbf{m}_c)$ in equation 3.33 as shown below.

$$\psi_{CD} = \sum_{c=1}^{C}\sum_{i\neq c} |\mathbf{A}\mathbf{z}(\mathbf{m}_c) - \mathbf{A}\mathbf{z}(\mathbf{m}_i)|^2 = \sum_{c=1}^{C}\sum_{i\neq c} |\mathbf{A}(\mathbf{z}(\mathbf{m}_c) - \mathbf{z}(\mathbf{m}_i))|^2 = \sum_{c=1}^{C}\sum_{i\neq c} |\mathbf{A}\delta\mathbf{m}_{ci}|^2$$

where, similarly to $\delta\mathbf{z}_{ij}$ in the section 3.2, we define $\delta\mathbf{m}_{ci} = \mathbf{m}_c - \mathbf{m}_i$. Given that:

$$|\mathbf{A}\delta\mathbf{m}_{ci}|^2 = \delta\mathbf{m}_{ci}^T\mathbf{A}^T\mathbf{A}\delta\mathbf{m}_{ci} = \sum_{l=1}^{2}\left(\sum_{k=1}^{p}\delta m_{ci}^k a_{lk}\right)^2 \tag{3.34}$$

we achieve that:

$$\psi_{CD} = \sum_{c=1}^{C}\sum_{i\neq c}\sum_{l=1}^{2}\left(\sum_{k=1}^{p}\delta m_{ci}^k a_{lk}\right)^2 \tag{3.35}$$

We are going to use the Lagrange function to maximize the ψ_{CD} subject, as in the previous sections, to the constraint on the norm of $\mathbf{A}$. The Lagrange function $\mathscr{L}_{CD}(\mathbf{A},\lambda)$ is given by :

$$\mathscr{L}_{CD}(\mathbf{A},\lambda) = \sum_{c=1}^{C}\sum_{i\neq c}\sum_{l=1}^{2}\left(\sum_{k=1}^{p}\delta m_{ci}^k a_{lk}\right)^2 - \lambda\left(\sum_{l=1}^{2}\sum_{k=1}^{p}a_{lk}^2 - 1\right) \tag{3.36}$$

Deriving equation 3.36 with respect to a_{rs} (note that sum over l gives only non-zero terms for $l = r$), equaling to zero and simplifying, we have:

$$\begin{aligned}\frac{\partial\mathscr{L}_{CD}}{\partial a_{rs}} &= \sum_{c=1}^{C}\sum_{i\neq c}2\left(\sum_{k=1}^{p}\delta m_{ci}^k a_{rk}\right)\delta m_{ci}^s - 2\lambda a_{rs} = 0\\ \Rightarrow \sum_{k=1}^{p}\left(\sum_{c=1}^{C}\sum_{i\neq c}\delta m_{ci}^k\delta m_{ci}^s\right)a_{rk} - \lambda a_{rs} &= 0, \quad r = 1,2; s = 1,\ldots,p\end{aligned} \tag{3.37}$$

Defining b_{sk} similarly to the section 3.2:

$$b_{sk} = \sum_{c=1}^{C}\sum_{i\neq c}\delta m_{ci}^k\delta m_{ci}^s, \quad s,k = 1,\ldots,p \tag{3.38}$$

so that $b_{sk} = b_{ks}$, the eqs. 3.37 can be written as:

$$\sum_{k=1}^{p}b_{sk}a_{rk} - \lambda a_{rs} = 0, \quad r = 1,2; s = 1,\ldots,p \tag{3.39}$$

Defining the $p\times p$-order symmetric matrix $\mathbf{B}_{CD}$ as:

$$\mathbf{B}_{CD} = \sum_{c=1}^{C}\sum_{i\neq c}\delta\mathbf{m}_{ci}\delta\mathbf{m}_{ci}^T \tag{3.40}$$

the eqs. 3.39 can be expressed in matrix form as:

$$(\mathbf{B}_{CD} - \lambda\mathbf{I}_p)^T\mathbf{A}^T = \mathbf{0} \tag{3.41}$$

Label	Mapping	Overlap	Function
DSOLM	Direct class Separability Optimization Linear Mapping	DS	Linear
TOLM	Thornton Opt. Linear Map.	Thornton	
CDOLM	Class Distance Opt. Linear Map.	CD	
QKDA	DA with Quadratic Kernel map.	J-index	Quadratic
DSOQM	Direct class Separability Opt. Quadratic Map.	DS	
TOQM	Thornton Opt. Quadratic Map.	Thornton	
CDOQM	Class Distance Opt. Quadratic Map.	CD	
PKDA(D)	DA with Polynomial Kernel map.	J-index	Polynomial ($D = 2,\ldots,7$)
DSOPM(D)	Direct class Separability Opt. Pol. Map.	DS	
TOPM(D)	Class Distance Opt. Pol. Map.	Thornton	
CDOPM(D)	Class Distance Opt. Pol. Map.	CD	

Table 3.1: Mappings varying the overlap measure and the function type. The LDA, corresponding to J-index and linear function, is not listed in the current table, because it is already provided by the Drtoolbox (table 3.1).

which is the same eigenvalue problem as in sections 3.1 and 3.2 but for matrix $\mathbf{B}_{CD}$, so the matrix $\mathbf{A}$ which maximizes ψ_{CD} is the one whose rows are the main eigenvectors of matrix $\mathbf{B}_{CD}$.

Using the previous four overlapping measures (Thornton index, direct class separability, J-index and class mean distance), combined with the eight Φ functions that we proposed in the beginning of this chapter (linear, quadratic and polynomial with degree $D = 2,\ldots,7$), and removing LDA (corresponding to J index and linear function, which is already provided by Drtoolbox) we achieve $4 \times 8 - 1 = 31$ mapping methods, which are summarized by Table 3.1.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 4

EXPERIMENTS WITH 2D MAPPINGS

In the current chapter we apply the 31 mappings included in Table 3.1 of chapter 3 in order to transform the original patterns in $\mathbb{R}^n$ to $\mathbb{R}^2$ for the 71 classification benchmark data sets of Table 4.1. The mapping will be created using a set of training patterns. A mapping will be better for classification tasks if the set of 2D patterns that it creates by applying it on a separate set of test patterns, not included in the training set, is easier to classify than the 2D sets created by other mappings. Therefore, it will be necessary to use a classifier to process the set of 2D test patterns created by the mapping, and the best mapping will be the one which provides the best classifier performance. In our case, the classifier used is a Support Vector Machine (SVM) with Gaussian kernel implemented by the SVM library [18], due to its high performance widely tested in the literature.

4.1 Experimental setting

Before starting the experiments, we pre-processed the data sets in Table 4.1 by making the mean equal to zero and the standard deviation one and transforming all the discrete inputs to numerical values. We also removed the repeated patterns in order to avoid the data redundancy during the mapping process because its not functional to transform the same pattern many times, and to avoid repeated values in the matrices $\mathbf{B}$, which would become singular difficulting the eigenvector calculation. We also create the partitions corresponding to 4 folds, each one composed by training, validation and test set. The partitions are generated in such a way that, for each class in the data set, 50%, 25% and 25% of the patterns are used for

No.	dataset	#pat.	#inp.	#cl.	No.	dataset	#pat.	#inp.	#cl.
1	acute-inflammation	99	6	2	37	lung-cancer	32	56	3
2	acute-nephritis	99	6	2	38	lymphography	142	18	2
3	annealing*	786	31	5	39	mammographic	642	5	2
4	balance-scale	625	4	3	40	molec-biol-promoter	106	57	2
5	balloons	16	4	2	41	monks-1*	124	6	2
6	blood	502	4	2	42	monks-2*	169	6	2
7	breast-cancer	266	9	2	43	monks-3*	122	6	2
8	breast-cancer-wisc	463	9	2	44	oocytes-merluccius-nucleus-4d	1022	41	2
9	breast-cancer-wisc-diag	569	30	2	45	oocytes-merluccius-states-2f	1022	25	3
10	breast-cancer-wisc-prog	198	33	2	46	oocytes-trisopterus-nucleus-2f	912	25	2
11	breast-tissue	105	9	6	47	oocytes-trisopterus-states-5b	912	32	3
12	congressional-voting	342	16	2	48	parkinsons	195	22	2
13	conn-bench-sonar-mines-rocks	208	60	2	49	pima	768	8	2
14	credit-approval	690	15	2	50	pittsburg-bridges-MATERIAL	102	7	3
15	cylinder-bands	511	35	2	51	pittsburg-bridges-rel	99	7	3
16	dermatology	366	34	6	52	pittsburg-bridges-span	89	7	3
17	echocardiogram	131	10	2	53	pittsburg-bridges-tord	98	7	2
18	ecoli	332	6	6	54	pittsburg-bridges-type	101	7	6
19	energy-y1	768	8	3	55	planning	176	12	2
20	energy-y2	768	8	3	56	post-operative	72	8	2
21	fertility	99	9	2	57	seeds	210	7	3
22	flags	190	28	6	58	spect*	62	22	2
23	glass	213	9	6	59	spectf*	80	44	2
24	haberman-survival	283	3	2	60	statlog-australian-credit	690	14	2
25	hayes-roth*	51	3	3	61	statlog-german-credit	1000	24	2
26	heart-cleveland	303	13	5	62	statlog-heart	270	13	2
27	heart-hungarian	293	13	2	63	statlog-vehicle	846	18	4
28	heart-switzerland	123	12	5	64	synthetic-control	600	60	6
29	heart-va	199	12	5	65	teaching	106	5	3
30	hepatitis	155	19	2	66	tic-tac-toe	958	9	2
31	horse-colic *	300	25	2	67	trains	10	32	2
32	ilpd-indian-liver	570	10	2	68	vertebral-column-2classes	310	6	2
33	image-segmentation*	210	18	7	69	vertebral-column-3classes	310	6	3
34	ionosphere	350	33	2	70	wine	178	13	3
35	iris	147	4	3	71	zoo	55	16	6
36	lenses	20	4	2					

Table 4.1: Collection of 71 data sets from the UCI data base and our real problems. It shows the number of patterns (#pat.), inputs (#inp.),and classes (#cl.). * These data set has two separated data files one for training, and the other one for testing

training, validation and test sets, respectively, rotating the partitions for each fold. The classes with less than 5 patterns are not considered.

We compare the performance achieved by SVM using: 1) the 31 proposed mappings, summarized in Table 3.1; 2) the 34 mappings provided by the Drtoolbox (Table 4.2), described in section 2; and 3) the original n-dimensional data (henceforth named as svmNd). Overall,

we used 66 methods and 71 data sets, which give a total of 66 $\times$ 71 = 4686 runs, occurring errors in 126 cases (2.68% of the cases). For each fold, each one of the 65 mappings is created, i.e., the corresponding **A** matrix is calculated by solving the associated eigenvalue problem, described in chapter 3). The matrices $\mathbf{B}_T$, $\mathbf{B}_{DS}$, $\mathbf{B}_{DA}$ and $\mathbf{B}_{CD}$ are calculated using only the training patterns, excluding the validation and test patterns. The mapping defined by matrix **A** is used to transform the validation and test patterns from $\mathbb{R}^n$ to $\mathbb{R}^2$. The proposed mappings (Table 3.1) are already functions from $\mathbb{R}^n$ to $\mathbb{R}^2$ and thus can map validation and test patterns (also called out-of-sample patterns in the literature), while the Drtoolbox mappings (Table 4.2) use the `out_of_sample(...)`, for mappings labeled with an asterisk in that table, or `out_of_sample_est(...)` functions of this toolbox (the last one is used for projections without an explicit mapping, and it develops an interpolation to give a mapped version of the out-of-sample pattern).

Linear Discriminant Analysis (textbfLDA)*	Locally Linear Embedding (LLE)*
Generalized Discriminant Analysis (**GDA**)	Hessian LLE
Principal Component Analysis (PCA)*	Neighborhood Preserving Embedding (NPE)*
Kernel PCA*	Local Tangent Space Alignment (LTSA)
Probabilistic PCA*	Linear Local Tangent Space Alignment (LLTSA)*
Gaussian Process Latent Variable Model (GPLVM)	Laplacian Eigenmap*
Factor Analysis (FA)*	Linearity Preserving Projection (LPP)*
Sammon mapping	Maximum Variance Unfolding (MVU)
Multidimensional scaling (MDS)	Fast MVU*
Stochastic Proximity Embedding (SPE)	Landmark MVU*
Stochastic Neighbor Embedding (SNE)	Conformal Eigenmap (CCA)
Symmetric SNE	Locally Linear Coordination (LLC)
t-Distributed SNE	Manifold Charting*
Isomap*	Coordinated Factor Analysis (CFA)
Landmark Isomap*	Large Margin Nearest Neighbor (**LMNN**)*
Diffusion map	Maximally Collapsing Metric Learning (**MCML**)*
Autoencoder*	Neighborhood Components Analysis (**NCA**)*

Table 4.2: Mappings provided by the Drtoolbox. The supervised mappings (in bold) use the class labels, and are expected to provide better classification results. The mappings with (resp. without) an asterisk use the out_of_sample (resp. out_of_sample_est) function to map out-of-sample patterns.

A Gaussian kernel SVM is trained using the 2D mapped training patterns of each fold. The two SVM metaparameters (regularization C and Gaussian spread γ) are tuned using values in the ranges $\{2^i\}_{-5}^{1}5$ and $\{2^i\}_{-15}^{3}$ respectively, values recommended by the LIBSVM developers. We select the values which provide the best average performance on the 2D validation sets. Finally, the SVM with the selected metaparameter values is trained and tested on the

No.	$\kappa(\mathbb{R}^n)$	$\kappa(\mathbb{R}^2)$	$\kappa(\mathbb{R}^n) - \kappa(\mathbb{R}^2)$	Mapping	No.	$\kappa(\mathbb{R}^n)$	$\kappa(\mathbb{R}^2)$	$\kappa(\mathbb{R}^n) - \kappa(\mathbb{R}^2)$	Mapping
1	100.00	100.00	0.00	LDA	37	34.02	38.39	-4.37	PCA
2	100.00	100.00	0.00	LDA	38	64.49	66.99	-2.50	PCA
3	73.75	70.52	3.23	PKDA7	39	62.63	57.81	4.82	ProbPCA
4	96.67	84.73	11.94	DSOPM2	40	61.54	69.23	-7.69	DSOPM7
5	62.50	75.00	-12.50	LDA	41	46.67	61.67	-15.00	TOPM2
6	32.92	25.56	7.36	CDOLM	42	42.10	36.15	5.95	DSOPM2
7	30.62	34.36	-3.74	CDOPM7	43	78.33	80.00	-1.67	DSOPM3
8	91.72	92.15	-0.43	PKDA2	44	66.09	56.78	9.31	LDA
9	93.16	93.57	-0.41	PKDA2	45	81.98	79.97	2.01	PKDA3
10	33.49	31.85	1.64	LDA	46	66.05	52.80	13.25	LDA
11	65.91	59.27	6.64	NPE	47	86.48	82.61	3.87	LDA
12	83.89	89.58	-5.69	LDA	48	76.11	63.80	12.31	PKDA5
13	66.34	59.62	6.72	PKDA2	49	46.49	50.20	-3.71	LDA
14	71.57	71.87	-0.30	PKDA7	50	54.95	59.90	-4.95	DSOPM4
15	47.99	42.06	5.93	QKDA	51	33.08	41.27	-8.19	TOPM5
16	96.22	84.55	11.67	LDA	52	27.17	47.78	-20.61	DSOPM5
17	61.77	63.51	-1.74	PKDA2	53	14.07	49.17	-35.10	QKDA
18	80.57	72.86	7.71	DSOLM	54	44.70	41.84	2.86	QKDA
19	91.56	93.89	-2.33	QKDA	55	-0.96	13.15	-14.11	ProbPCA
20	91.56	93.89	-2.33	QKDA	56	-2.55	17.54	-20.09	MVU
21	-4.55	30.76	-35.31	PKDA3	57	90.44	95.59	-5.15	LDA
22	37.97	30.13	7.84	PKDA3	58	20.40	35.23	-14.83	CDOPM7
23	51.76	48.18	3.58	TOLM	59	52.50	55.00	-2.50	DSOPM4
24	8.57	19.54	-10.97	TOPM4	60	5.70	8.04	-2.34	GDA
25	61.95	48.73	13.22	DSOQM	61	40.82	41.11	-0.29	LDA
26	34.93	39.48	-4.55	PKDA3	62	68.74	69.41	-0.67	DSOPM2
27	62.00	63.13	-1.13	CDOPM3	63	76.65	65.07	11.58	LDA
28	9.87	18.23	-8.36	PKDA2	64	98.80	86.80	12.00	CDOPM2
29	10.93	12.55	-1.62	CDOPM3	65	37.16	27.11	10.05	KernelPCA
30	41.23	61.30	-20.07	ProbPCA	66	98.13	96.25	1.88	LDA
31	65.34	54.87	10.47	CDOPM5	67	0.00	100.00	-100.00	CDOLM
32	25.02	21.13	3.89	ProbPCA	68	60.75	57.62	3.13	CDOQM
33	87.50	75.60	11.90	PKDA2	69	77.65	70.79	6.86	LDA
34	87.21	87.90	-0.69	CDOQM	70	98.24	97.37	0.87	CDOPM3
35	95.83	93.75	2.08	LPP	71	91.96	91.99	-0.03	PKDA2
36	62.50	87.50	-25.00	PKDA2					

Table 4.3: Value of κ achieved by SVM using the original patterns in $\mathbb{R}^n$ (column $\kappa(\mathbb{R}^n)$, corresponding to svmNd in the test) and using the patterns mapped to $\mathbb{R}^2$ (column $\kappa(\mathbb{R}^2)$, and difference $\Delta = \kappa(\mathbb{R}^n) - \kappa(\mathbb{R}^2)$, alongside with the name of the mapping which achieved the best κ in $\mathbb{R}^2$.

four 2D training and test sets, respectively, and the average performance is reported for the 65 mappings and svmNd. The performance measure that we use is the Cohen kappa[1], denoted as κ and measured in %, which excludes the probability of classifier success by chance:

[1] https://en.wikipedia.org/wiki/Cohen_kappa

$$\kappa = 100\frac{p_a - p_e}{s - p_e}; \quad p_a = \sum_{i=1}^{C} n_{ii}; \quad p_e = \frac{1}{s}\sum_{i=1}^{C}\left(\sum_{j=1}^{C} n_{ij}\right)\left(\sum_{k=1}^{C} n_{ki}\right); \quad s = \sum_{i=1}^{C}\sum_{j=1}^{C} n_{ij}$$

where n_{ij} is the number of patterns of class i assigned by the classifier to class j, for $i, j = 1, \ldots, C$. In order to select the best mapping over the collection of 71 data sets, we create a ranking of the methods under comparison using the Friedman rank test [100]. This rank is decreasing with the performance, so that lower rank means better results. Given that the classifier is the same for all the mappings, higher κ for a certain mapping means that the mapped data are easier to classify, so the mapping can be considered more useful for classification tasks.

4.2 Discussion

Table 4.3 reports the κ achieved by the SVM using the original patterns in $\mathbb{R}^n$ and using the mapped patterns in $\mathbb{R}^2$, with the name of the mapping which provided the best κ in $\mathbb{R}^2$. Surprisingly, for 38 (resp. 2) of 71 data sets the performance κ is better (resp. equal) in $\mathbb{R}^2$ than in $\mathbb{R}^n$, exhibiting negative (res. zero) differences. In the remaining 31 data sets, κ is better in $\mathbb{R}^n$ than in $\mathbb{R}^2$, but these positive differences are not very high. In fact, for data sets with positive differences, their highest value is 13.25 (data set no. 46, oocutes-trisopterus-nucleus-2f), and the mean and median are 6.99 and 6.72, respectively. On the other hand, in data sets with negative differences their highest value is -100, and their mean and median are -10.55 and -4.46, respectively. Therefore, the conclusion is that mapping to $\mathbb{R}^2$, which allows a graphical view of the classification problem, does not reduce dramatically performance compared to the original n-dimensional problem. The appendix A reports the whole confusion matrices, alongside with the κ achieved by SVM using the best mapping for each of the 71 data sets (for space reasons, the confusion matrices only report the data set number, as in Table 4.1). Figure 4.1 (upper panel) shows the κ achieved by SVM in $\mathbb{R}^n$ (denoted previously as svmNd) and the best κ achieved by SVM in 2D. We can see that, excepting the outlier in position 4 of the plot, which corresponds to data set trains, where SVM achieves κ=0% and κ=100% with patterns in $\mathbb{R}^n$ $\mathbb{R}^2$ patterns, respectively, almost all the red points are inside the ± 10 band around the blue line, excepting 10 data sets where red points are only sligthly below the ± 10 band (in fact, the highest positive difference is 13.25, as we reported above). This means that the best κ in 2D is never too far from the κ in nD. The lower panel of Figure 4.1 shows the differences between both κ's: their values are positives

only after position 40 of 71, and the positive values never overcome 15%, while the negative differences are below -10 for 12 data sets.

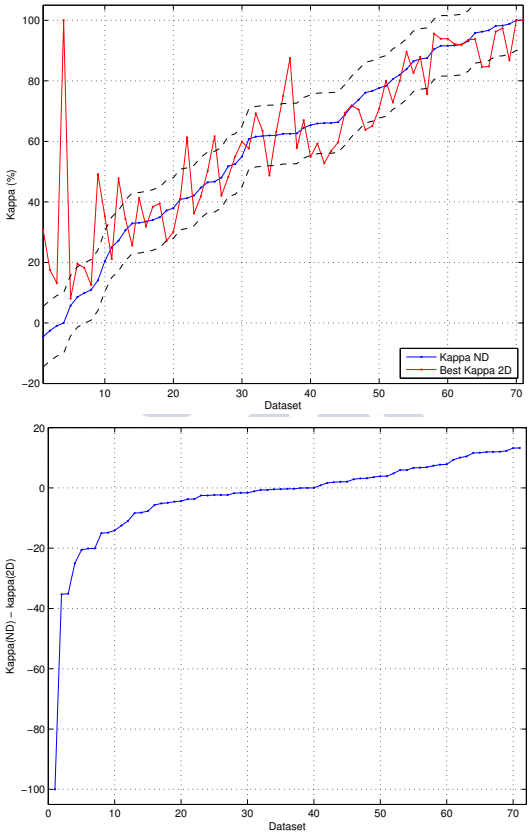


Figure 4.1: Upper panel: κ (in %) achieved by svmNd (in blue) and best κ achieved by SVM using the 2D mapped patterns (in red), for each data set, sorted by increasing κ (in nD). Black dashed lines limit the region where κ= $\kappa(n\text{D}) \pm 10$. Lower panel: Difference $\kappa(n\text{D})$ - best $\kappa(2\text{D})$ for each data set, sorted increasingly.

However, we must emphasize that we are comparing $\mathbb{R}^n$ to the best $\mathbb{R}^2$ mapping, which is not the same for all the data sets. Below we will select the globally best mapping, and we will evaluate the difference between κ in $\mathbb{R}^n$ and $\mathbb{R}^2$ for this best mapping. By the moment, we have seen that for all the data sets there is at least one mapping which does not reduce the performance too much compared to the original data, i.e., there is a mapping which projects

to $\mathbb{R}^2$ without destroying the classification problem, i.e., keeping in $\mathbb{R}^2$ the "structure" of the original classification problem in $\mathbb{R}^n$.

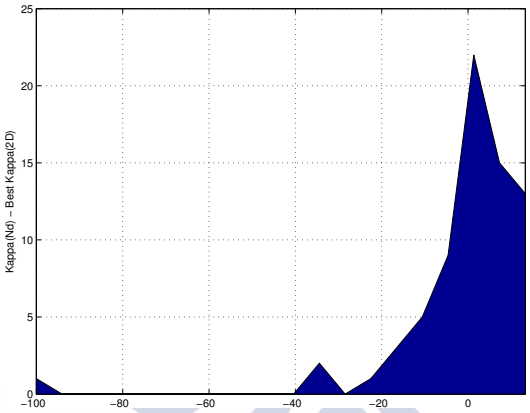


Figure 4.2: Histogram of the difference $\kappa(\mathbb{R}^n) - \kappa(\mathbb{R}^2)$ achieved by SVM for all the data sets.

Figure 4.2 shows the histogram of these differences: for 37 data sets the difference is negative, which means that $\kappa(\mathbb{R}^2)$ overcomes $\kappa(\mathbb{R}^n)$, while for other 31 data sets the difference is positive, so the mapping to $\mathbb{R}^2$ reduced the κ. For other 3 data sets the difference is zero. The mean and median differences are -2.25 and -0.29, respectively. Considering only those data sets where the difference is positive, which means that $\kappa(\mathbb{R}^n) > \kappa(\mathbb{R}^2)$, the mean and median values are 6.99 and 6.72, respectively, which means that even when the mapping reduces the classifier performance, the reduction is not too high (the maximum difference is 13.25).

The upper panel of Figure 4.3 shows the average κ of the mappings in Table 3.1 (including LDA, which is listed in Table 4.2) varying the overlap measure (DA, DS, T and CD) and the function (linear, quadratic and polynomial with degree $D = 2 \ldots 7$). Given that a polynomial with high degrees fits better complex mappings, it is expectable that κ were increasing with the function degree, analogously to a Taylor series. However, the linear versions already achieve good κ values, while the quadratic work much worse, and the polynomial mappings decrease or keep almost constant increasing D in the four measures. The bad results of quadratic compared to polynomial mappings means that the crossed terms $x_i x_j$ with $j \geq i$ are not useful to create this kind of mappings, and that pure powers x_i^d are much more efficient. Comparing the overlap measures, DA achieves the best results, followed by CD, while DS usually has

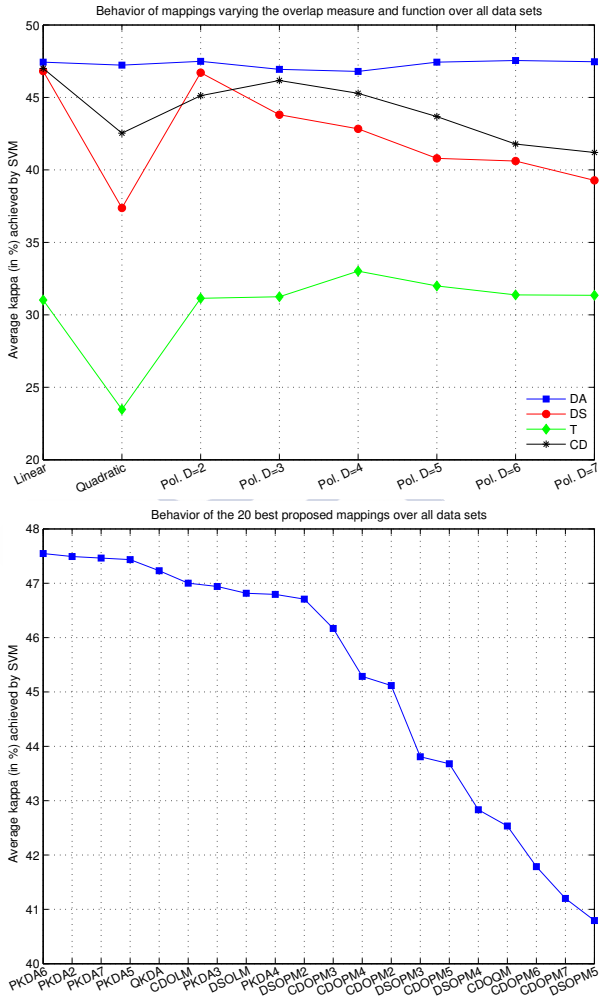


Figure 4.3: Upper panel: average κ (in %) achieved by SVM over all the data sets for the proposed mappings (Table 3.1), varying the function and overlap measure. Lower panel: average κ achieved by the 20 best of the proposed mappings, sorted decreasingly.

lower performances and the Thornton index (T) is clearly the worst one. When the polynomial degree D is increased, DA is almost constant, while CD and DS clearly decrease, and T keeps almost constant with a maximum for D=4. The lower panel of Figure 4.3 sorts the 20 best proposed mappings by decreasing κ. The PKDA versions achieve the first positions, with the

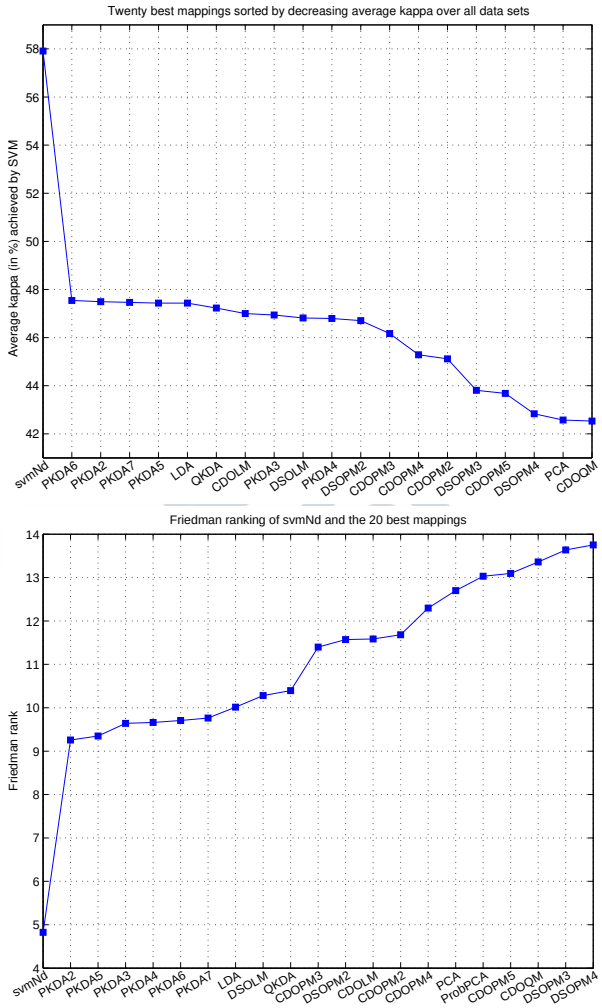


Figure 4.4: Upper panel: average κ of svmNd and the 20 best mappings (according to κ) over all the data sets. Lower panel: Friedman ranks of the 20 best mappings.

best result for D=6. The linear mappings CDOLM and DSOLM achieve the best positions for CD and DS overlap measures, respectively. The DS and CD polynomial mappings are in the second half of the plot. The quadratic and Thornton mappings work very bad, being in the last positions or outside the Figure 4.3.

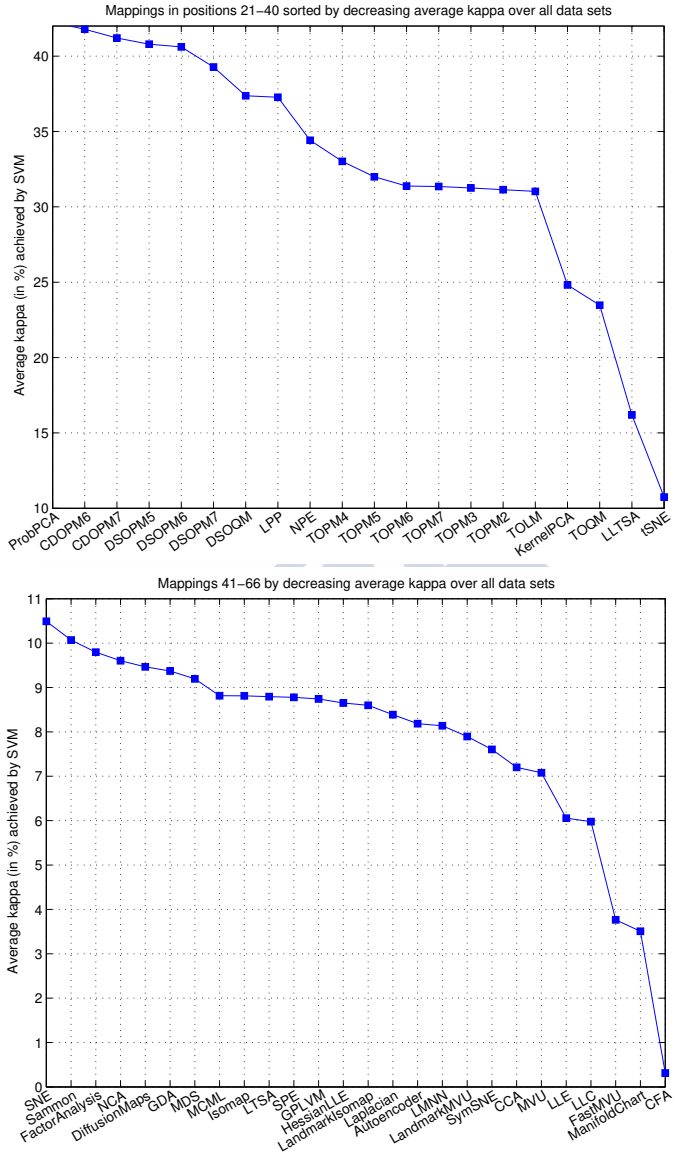


Figure 4.5: Average κ (in %) achieved by SVM over all the data sets for the mappings 21-40 (upper panel) and 41-66 (lower panel).

Figure 4.4 shows the average κ (upper panel) and the Friedman ranking (lower panel) achieved by svmNd and the 20 best mappings. The svmNd achieves the best average κ (57.91%), while PKDA6 (κ=47.55%), PKDA2 (47.49%), PKDA7 (47.46%) and PKDA5 (47.43%) are better than LDA (47.38%), which is the best Drtoobox mapping. The other linear mapping in the top-20 is CDOLM, and the only quadratic mappings are QKDA, right after LDA, and CDOQM (last position). Considering ranks (lower panel), the proposed mappings achieve 17 out of 20 best (lowest) ranks. The PKDA2 achieves the best rank, despite PKDA6 achieves highest average κ (in fact, PKDA5, PKDA3 and PKDA4 also have better ranks than PKDA6), so PKDA2 can be considered the best mapping for classification. Anyway, PKDA with all the degrees have better ranks than LDA (8th position), which is followed by QKDA and by several CD and DS mappings. Besides LDA, the only Drtoolbox mappings included in the 20 best ranks are PCA and ProbPCA, with much lower κ values (42.58% and 42.25% respectively).

No.	svmNd	PKDA2	LDA	No.	svmNd	PKDA2	LDA	No.	svmNd	PKDA2	LDA
1	100	100	100	25	62.0	39.5	31.2	49	46.5	39.7	50.2
2	100	100	100	26	34.9	37.8	30.5	50	55.0	55.4	48.2
3	73.8	65.5	57.9	27	62.0	59.7	53.7	51	33.1	36.7	38.8
4	96.7	80.3	83.9	28	9.9	18.2	11.1	52	27.2	43.0	16.8
5	62.5	-2.5	75.0	29	10.9	7.3	5.8	53	14.1	16.7	17.5
6	32.9	11.8	18.6	30	41.2	56.9	36.9	54	44.7	35.5	36.9
7	30.6	22.4	17.6	31	65.3	51.9	54.5	55	-1.0	-13.9	0.0
8	91.7	92.2	90.4	32	25.0	1.3	7.4	56	-2.5	1.2	-2.5
9	93.2	93.6	90.5	33	87.5	75.6	66.1	57	90.4	89.7	95.6
10	33.5	20.3	31.9	34	87.2	85.9	68.7	58	20.4	11.4	-4.9
11	65.9	53.9	49.6	35	95.8	87.5	89.6	59	52.5	35.0	-7.5
12	83.9	84.2	89.6	36	62.5	87.5	54.2	60	5.7	1.3	-0.0
13	66.3	59.6	38.9	37	34.0	-6.6	2.3	61	40.8	33.9	41.1
14	71.6	70.0	71.5	38	64.5	63.0	59.0	62	68.7	68.6	68.8
15	48.0	27.1	38.8	39	62.6	52.4	53.4	63	76.7	36.2	65.1
16	96.2	76.5	84.5	40	61.5	67.3	9.6	64	98.8	84.2	77.0
17	61.8	63.5	61.1	41	46.7	40.0	28.3	65	37.2	19.2	19.3
18	80.6	65.8	57.2	42	42.1	-6.0	-5.5	66	98.1	28.2	96.2
19	91.6	89.5	84.5	43	78.3	56.7	63.3	67	0.0	25.0	-25.0
20	91.6	89.5	84.5	44	66.1	19.7	56.8	68	60.8	30.1	51.7
21	-4.5	-3.7	-3.9	45	82.0	79.8	79.5	69	77.7	50.5	70.8
22	38.0	30.0	18.2	46	66.0	34.8	52.8	70	98.2	95.6	97.4
23	51.8	42.4	40.5	47	86.5	60.3	82.6	71	92.0	92.0	83.9
24	8.6	4.6	4.3	48	76.1	49.8	55.8				
Avg.	57.91	47.49	47.43	#best	49	16	11				

Table 4.4: Values of κ (in %) of svmNd, PKDA2 and LDA for each data set. The last row reports the average κ over all the data sets and the number of data sets where each method achieves the best κ (the sum is 76 instead of 71 due to ties for some data sets).

The Figure 4.5 shows the κ of the mappings from positions 21 to 40 (upper panel) and 41-66 (lower panel). Most of the mappings in the upper panel are proposed by us (DS and all the T mappings), although they achieve κ values much lower than mappings in positions 1-20. The quadratic mappings (DSOQM, TOQM and QKDA) are in this range of κ's. Among the Drtoolbox mappings, only LPP (37.37%), NPE (34.41%), KernelPCA (24.81%), LLTSA (16.19%) and t-SNE (10.74%) are included in the upper panel, with very low κ values. The lower panel, with κ below 11%, includes the remaining Drtoolbox mappings. Among the supervised methods of the Drtoolbox, only LDA is in the top-20, while NCA (9.60%), GDA (9.37%), MCML (8.82%) and LMNN (8.14%) achieve results below several unsupervised mappings (SNE, Sammon and Factor Analysis, among others), being in the lower panel of Figure 4.5.

No.	svmNd	PKDA2	LDA	No.	svmNd	PKDA2	LDA
4	98.08	88.78	90.5	39	81.56	76.25	76.9
5	83.33	41.67	91.7	42	73.81	51.19	48.8
6	77.82	70.36	73.4	46	83.59	69.16	77.3
10	75.00	75.52	75.5	58	65.00	58.33	51.7
15	75.39	67.32	71.3	59	76.25	67.50	46.2
16	96.98	81.32	87.6	63	82.50	52.02	73.8
22	52.72	46.74	36.4	64	99.00	86.83	80.8
23	65.38	59.13	57.7	65	58.00	46.00	46.0
25	79.55	65.91	61.4	66	99.16	69.46	98.3
31	84.46	78.72	80.1	68	82.14	69.16	77.9
32	72.89	71.30	70.2	69	86.04	70.13	81.8
37	60.71	32.14	35.7				

Table 4.5: Accuracy (in %) of PKDA2, svmNd and LDA in those data sets where the svmNd outperforms PKDA in terms of κ. The data set number is listed in Table 4.1.

Additional experiments developed by us showed that when the Drtoolbox mappings are calculated using the whole data set (including training, validation and test patterns) the 2D mapped patterns are much easier to classify for SVM, achieving higher κ values. However, when the mapping is learnt using only training patterns, the classification results using validation (for parameter tuning) and test patterns are much worse. This means that Drtoolbox mappings as LPP, NPE, t-SNE or MDS, which are very useful to visualizate data, exhibit over-training in such a way that they build 2D data sets that do not extrapolate to out-of-sample patterns. We confirmed this conclusion comparing the average κ of the Drtoolbox techniques which use the `out_of_sample` (labeled with an asterisk in Table 4.2) and the `out_of_sample_est` functions. The formers provide an explicit mapping from $\mathbb{R}^n$ to $\mathbb{R}^2$, while the latters create a 2D pattern set representing the high-dimensional training patterns,

and afterwards use estimation techniques to map out-of-sample (i.e., validation and test) patterns. The average κ values are 17.7% and 8.2% for the methods with explicit mapping and for the others, respectively. This confirms our expectation that the classification is worse for those mappings which only create a 2D data set, which may eventually be hardly interpolable to out-of-sample patterns.

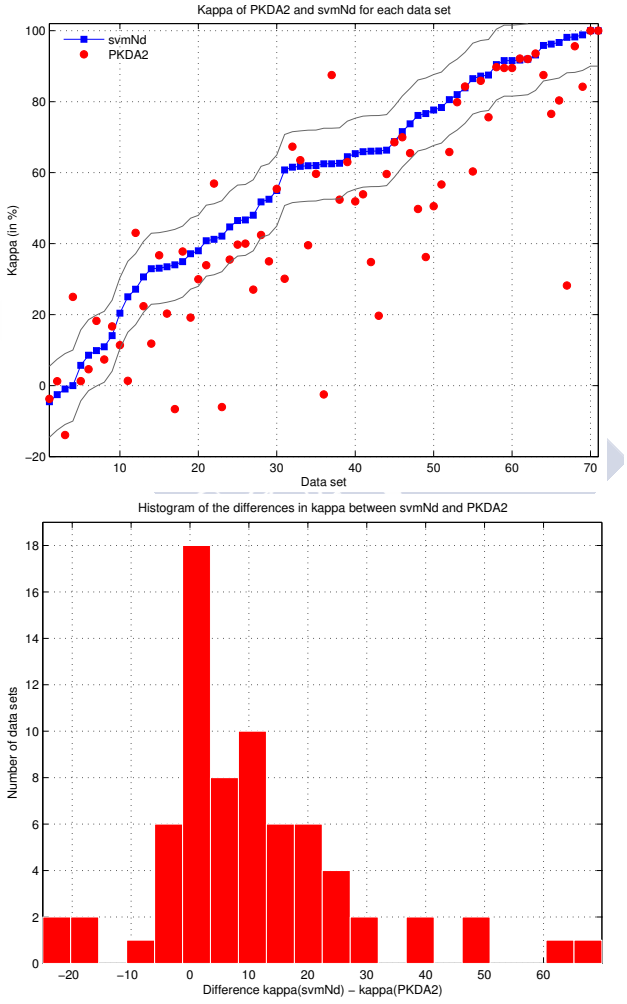


Figure 4.6: Upper panel: comparison of the κ (in %) achieved by SVM using PKDA2 and svmNd, ordered by increasing κ of svmNd. Lower panel: histogram of differences between svmNd and PKDA2.

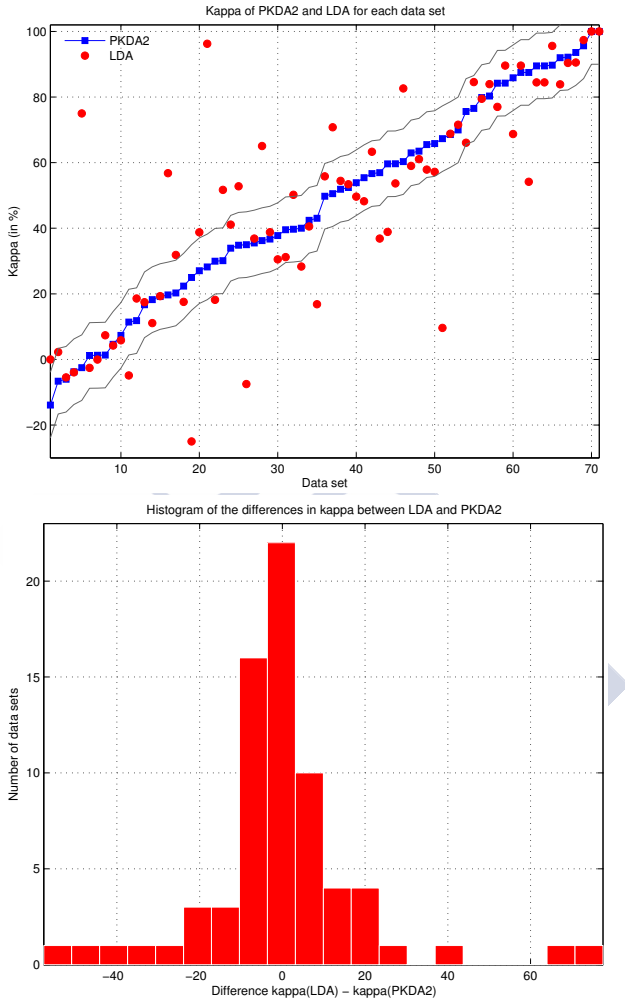


Figure 4.7: Upper panel: comparison of the κ achieved by PKDA2 and LDA, ordered by increasing κ of PKDA2. Lower panel: histogram of differences between LDA and PKDA2.

In order to compare svmNd and PKDA2, Table 4.4 reports the κ achieved with svmNd, PKDA2 and LDA for each data set, the average κ and the number of sets where each method achieves the best κ (column #best). There are big differences among data sets: in 8 data sets the PKDA2 outperforms svmNd. This is surprising, since the classifier using mapped patterns should work equally or worse than using the original patterns. In other 38 data sets the svmNd

outperforms PKDA2 by less than 10%. Therefore, for an important group of data sets (64.8% of the total) PKDA2 works better, or it does not work much worse, than svmNd. And only for the remaining 25 data sets (35.2% of the total) the svmNd outperforms PKDA2 by more than 10%. The differences between both in terms of κ are big for the 23 data sets listed in Table 4.5, which reports the accuracies of PKDA2, svmNd and LDA for those data sets. However, for most of them the difference in terms of accuracy is much smaller than in terms of κ. Only for 8 of 23 data sets (5, 16, 25, 37, 42, 63, 66 and 68) the difference in accuracy is also high.

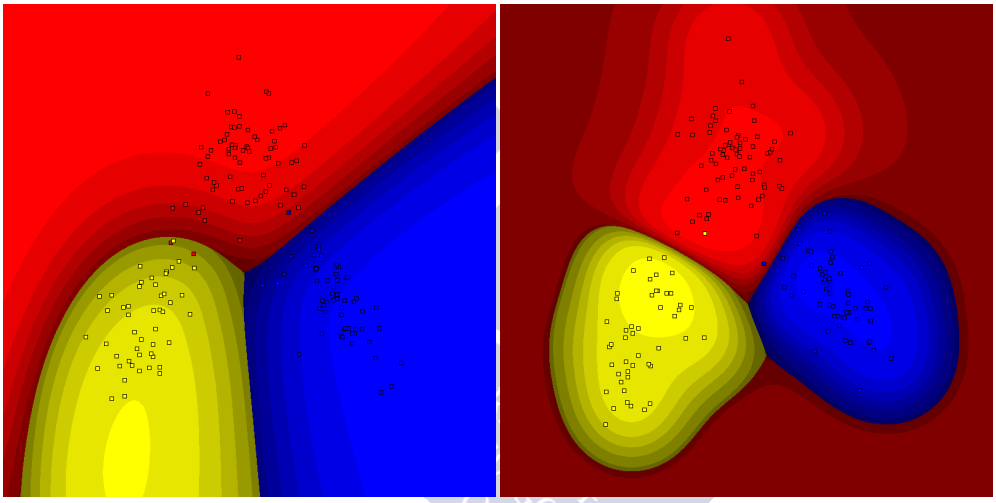


Figure 4.8: Training, validation and test patterns (small squares) in $\mathbb{R}^2$ mapped by PKDA2 (left panel) and LDA (right panel) and class regions learnt by SVM using both mappings for data set 70 (wine), with 178 patterns, 13 inputs and 3 classes (colors blue, red and yellow, graded by the classification probability given by SVM).

Figure 4.6 (upper panel) compares the κ of PKDA2 and svmNd for each data set. Two grey lines show a margin of $\pm 10\%$ around the κ achieved by svmNd. While four red points (PKDA2) are clearly over the 10%-margin (which means that PKDA2 outperforms svmNd), and other 40 points are within this margin (where svmNd and PKDA2 work similarly), the remaining 27 red points are clearly below the margin (which means that svmNd clearly outperforms PKDA2). The lower panel shows the histogram of the differences between svmNd and PKDA2: although there are 6 sets with differences above 30%, most of the differences are under 30%, and many of them are under 10%, while some sets have negative difference (which means that PKDA2 works better than svmNd).

Comparing PKDA2 and LDA, the former outperforms the latter in 36 data sets, being worse than LDA in 33 data sets, and both lie in 2 data sets. Considering the 23 data sets where svmNd clearly outperforms PKDA2 (Table 4.5), the LDA works in general similarly to PKDA2, except for 5 sets (5, 46, 63, 66 and 69). The upper panel of Figure 4.7 compares the κ of PKDA2 and LDA for each data set. There are 10 red points above the margin (data sets where LDA outperforms PKDA2), 50 points inside the margin (where both work similarly) and 11 points below the margin (where PKDA2 outperforms LDA). The histogram of differences between LDA and PKDA2 (lower panel of this Figure) shows that most differences are under 20%, there are only 4 exceptions with differences above 20%, and many sets with negative difference (meaning that PKDA2 outperforms LDA). A T-test shows that svmNd is significantly better than PKDA2 (p-value=3.02E-6) and LDA (p=2.49E-9), but PKDA2 is not better than LDA (p=0.96).

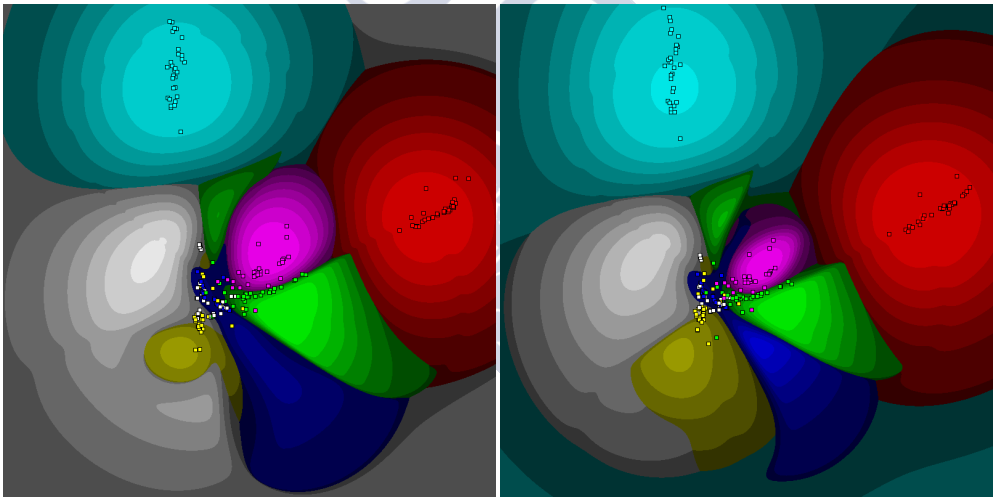


Figure 4.9: Class map learnt by SVM using the 2D patterns mapped by PKDA2 (left panel) and PKDA4 (right panel) for data set 33 (image-segmentation), with 210 patterns, 18 inputs and 7 classes (colors blue, red, yellow, green, white, magenta and cyan).

Since the proposed methods allow to visualize classification problems in 2D, one of the most important objectives of this thesis is to show the class maps created by SVM for a given mapping and data set, including the training, validation and test patterns, the regions assigned by SVM to each class and the borders among them. Figure 4.8 shows this information using PKDA2 (left panel) and LDA (right panel), which are the globally best-performing map-

pings, for data set 70 (wine). For this data set, SVM achieves κ=98.24% using the original n-dimensional patterns (svmNd), κ=95.61% using the 2D patterns mapped by PKDA2 and κ=97.36% using LDA. The fact that the κ values do not decrease too much from $\mathbb{R}^n$ to $\mathbb{R}^2$ suggest that the 2D map of this data set is highly representative of the original classification problem in $\mathbb{R}^n$, i.e., the 2D map is a valid projection of the original high-dimensional data. The region associated to each class is painted with the color associated to that class, but this color is degraded in order to show the probability of assigning a pattern to that class. This is calculated using the utility of the LIBSVM library to provide a probability of assigning a pattern to a given class. Specifically, the color in the classification map of Figure 4.8 is darker (resp. brighter) with lower (resp. higher) probabilities.

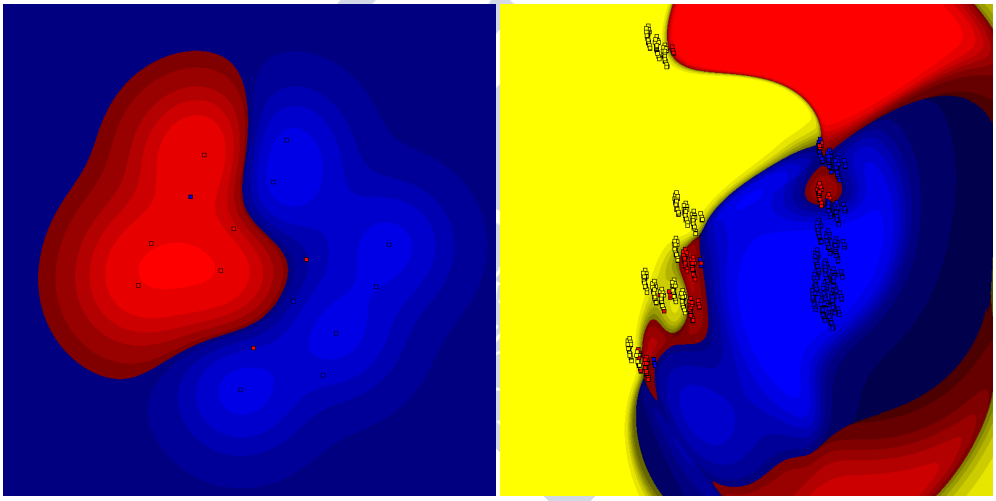


Figure 4.10: Class map created by SVM using the 2D patterns mapped by PKDA2 for data sets balloons (2 classes) and energy-y1 (3 classes).

We can also check that most squares (which correspond to the training, validation and test patterns) are placed in regions of their same color, which means that they are correctly classified. This happens for this data set, where the κ with 2D patterns is high both using PKDA2 and LDA. As an example of the opposite behavior, Figure 4.9 shows the map of data set 33 (image-segmentation) where the performance using the 2D patterns mapped by PKDA2 (κ=75.60%) and by LDA (66.07%) are worse than using the original patterns (svmNd, 87.5%). Consequently, there are more squares of a color placed in a region of a

different color (e.g., white patterns inside the blue region). Note in this case that the class "white", due to the probability color degradation, seems to be gray except in the small region where the probability is considered high. It is also interesting the relative positions of the regions associated to the different classes, and the number of patterns of one class which are mapped to regions associated to other classes (this can also measured by the non-diagonal terms in the confusion matrix). Figures 4.10, 4.11 and 4.12 show the maps created by PKDA2 for other data sets varying the number of classes (due to limitations on the number of colors, we excluded data sets with more than 8 classes).

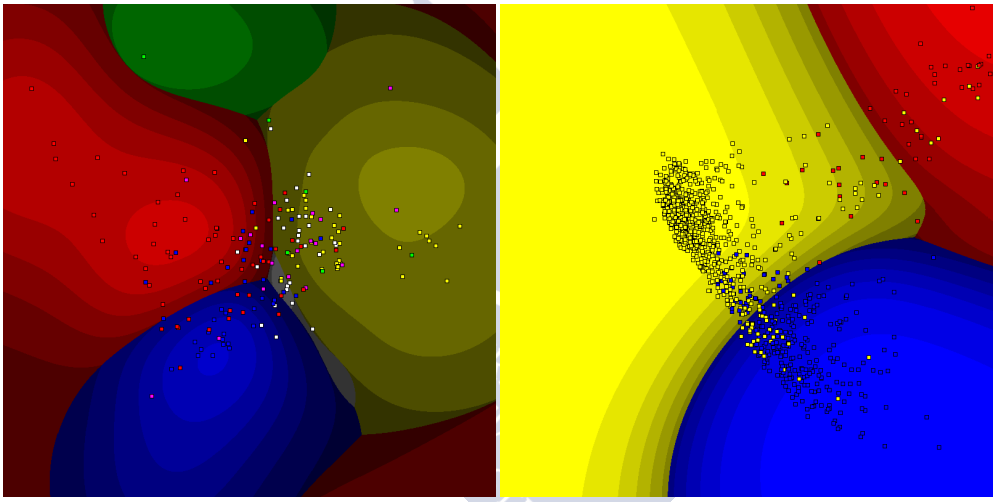


Figure 4.11: Class map created by SVM using the 2D patterns mapped by PKDA2 for data sets flags (6 classes) and oocytes_merluccius_states_2f (3).

Regarding to the computational efficiency, Figure 4.13 shows (upper panel) the times, in a logarithmic scale, spent by the proposed mappings for each measure (DA, DS, T and CD) and function (linear, quadratic and polynomial varying $D = 2, \ldots, 7$). All the mappings were run using Matlab 7.12 (R2011a) on a Linux Debian 3.2 computer (64 bits). Comparing measures, CD, DA and T are two orders of magnitude faster than DS. Comparing functions, the four measures behave similarly, with similar but translated plots. The linear function is the fastest and the quadratic is the slowest (by 2-3 orders). This shows the high computational cost of the cross products, which as we saw also reduces largely the performance. The polynomial functions slowly increase with D, as it might be expected, but the 2nd-degree polynomial is

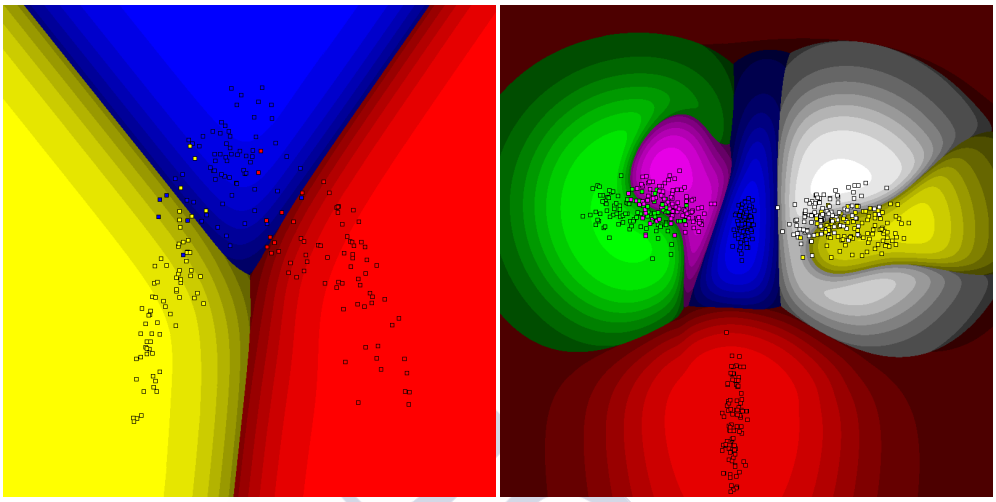


Figure 4.12: Class map created by SVM for data sets seeds (3 classes) and synthetic-control (6).

almost so fast as the linear function. The lower panel of Figure 4.13 sorts the times of the 20 fastest mappings. The PCA is the fastest one, but CDOLM, whose κ=47.0% is similar to LDA and much better than PCA (42.6%), is the second fastest, followed by MDS and LDA (the four mappings have times about 0.001 s.). The PKDA2, despite being quadratic, is almost so fast as LDA, which is linear. This means that the polynomial with pure powers, alongside with its good result, is relatively efficient. It is also relevant that, similarly to the κ ranking (Figure 4.4, lower panel), the majority (16) of the 20 fastest mappings are methods proposed by us (Table 3.1), so they are efficients and work well simultaneously.

4.3 Graphical interfaces

We developed two application programs which allow to map multi-dimensional classification data sets into 2D using the proposed methods and the ones provided by the Drtoolbox (Tables 3.1 and 4.2, respectively). The first program is a web application developed in the PHP programming language, so it can be accessed from anywhere, while the second is a standalone desktop application developed using the Matlab graphical user interface designer.

The objective of both applications is to provide a user-friendly interface and straightforward application that allows to create a 2D classification map for any data set, specially as

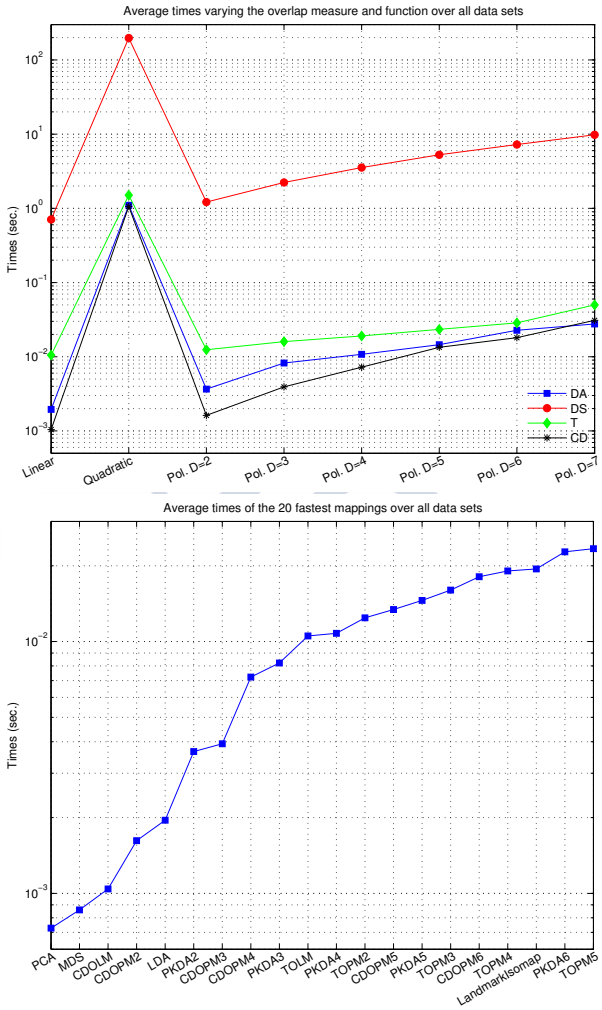


Figure 4.13: Upper panel: average times (in sec.) spent by the proposed mappings varying the overlap measure and function. Lower panel: times spent by the 20 fastest mappings.

a tool for data inspection and understanding for classification tasks. The user can inmediately transform patterns from $\mathbb{R}^n$ to $\mathbb{R}^2$, evaluating and comparing the classification results achieved by SVM in both spaces. The data set is imported in the standard comma-separated-values (CSV) format. All the data items must be numeric with each feature in a column, being

the last column the class label (also numeric, with values starting from 1). Before the mapping to 2D, the data are standarized in order to have zero mean and standard deviation one for each feature. The 71 classification benchmark data sets included in Table 4.1 are provided as trial data in order to allow users to run and test both PHP and Matlab interfaces.

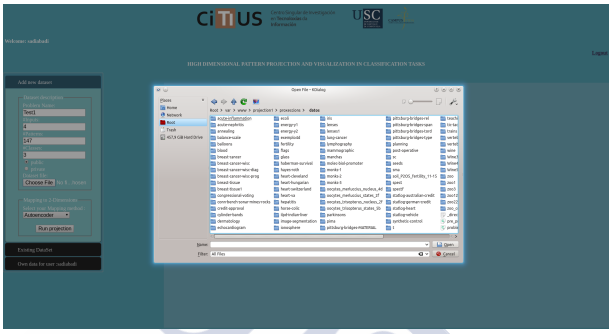


Figure 4.14: Uploading a new data set (lenses1) with 4 inputs,20 patterns and 2 classes using PHP web interface. These required information must be filled in the left panel using the "add new data set" menu.

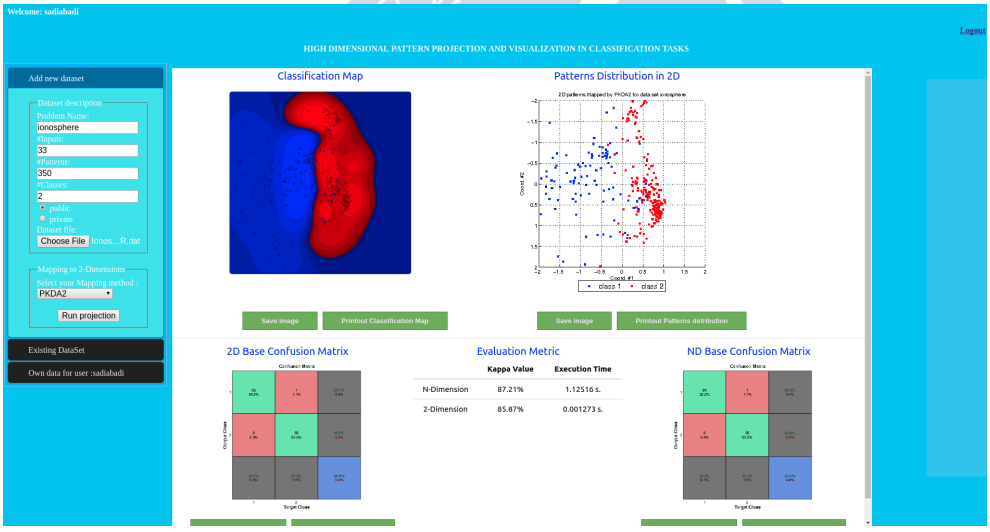


Figure 4.15: Screenshot of the PHP interface after projecting the UCI data set ionosphere using PKDA2.

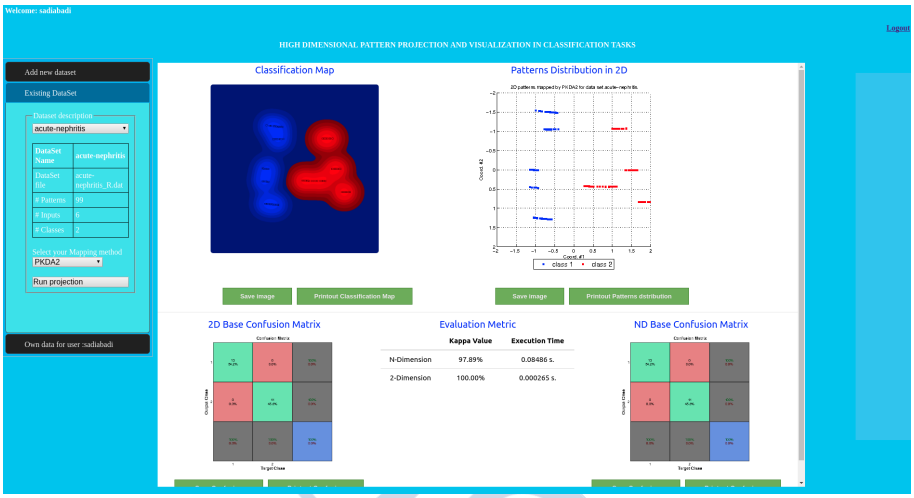


Figure 4.16: Screenshot of the PHP interface after mapping data set acute-nephritis using PKDA2.

4.3.1 PHP web interface

The PHP webapp was developed using PHP5, JavaScript, CSS3, HTML5 as developing languages for the graphical user interface, Matlab 2012 for the numerical computation and MySQL as database management system. The interface is shown in Figure 4.15 (the screenshot has been taken with the browser in full-screen visualization). After login into the system, the user must specify the name of the data set and the numbers of inputs, patterns and classes (see left part of Figure 4.14, panel "Add new data set"). These data items will be saved in the corresponding database table for the current user, which can retrieve it from the "Own data for user" panel (lower part of the left panel in Figure 4.14). The users have the option to make their own data public and share them with other users, or make it private for his own use only. The PHP interface captures the user inputs and uploads the data set file to the server, which is mapped to 2D by the Matlab program. This program also runs the SVM classifier both on the multi-dimensional and on the 2D mapped patterns.

After the execution of the Matlab projection program, the PHP interface displays the results (Figure 4.15 shows these results for the UCI data set ionosphere mapped using PKDA2). These results contain the classification map (top left) created by the SVM using the 2D projected patterns, including the regions of the 2D space assigned by SVM to each class; the 2D mapped patterns mapped by the selected projection (top right); the confusion matrix (bot-

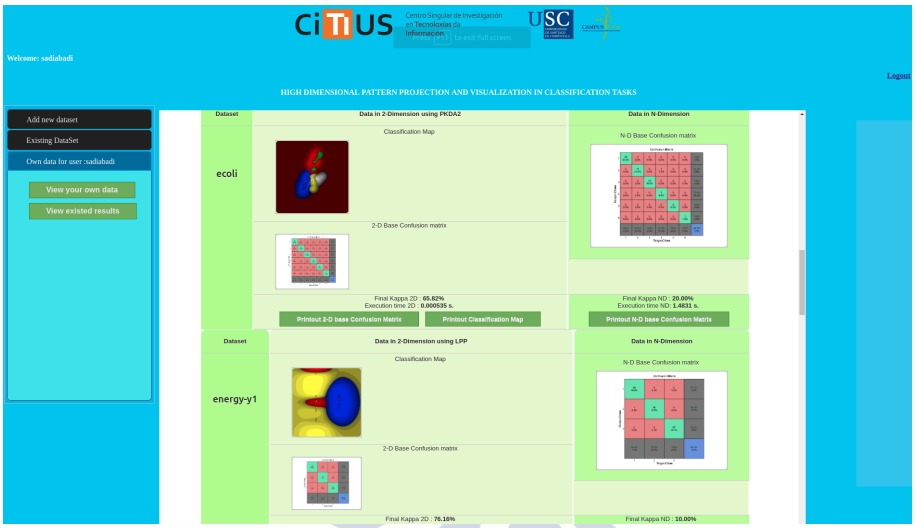


Figure 4.17: Screenshot with the compilation of results for a given user in the PHP interface.

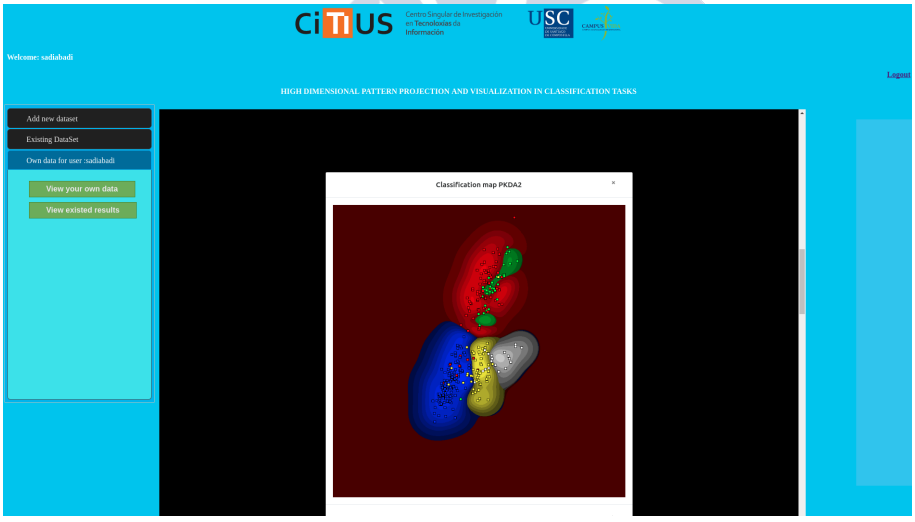


Figure 4.18: Zooming-in the classification map of balloon data set using PKDA2 algorithm.

tom left) achieved by SVM on these 2D data; the evaluation metric (bottom middle), which includes the κ achieved by SVM in the original n-dimensional and in the mapped 2D data,

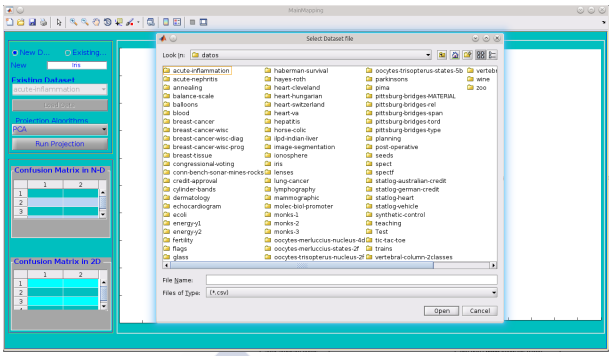


Figure 4.19: Screenshot of the desktop Matlab interface uploading a new data set using the left panel. The file explorer only requires the data set file and the remaining details are extracted from it.

alongside with both execution times; and the nD confusion matrix (bottom right) of SVM on the original n-dimensional patterns. In this example, where the number n of inputs is 33, the κ only decreases from 87.21% to 85.87% in $\mathbb{R}^n$ and $\mathbb{R}^2$, respective, and the elapsed time spent by the SVM in 2D is reduced about 1000 times with respect to $\mathbb{R}^n$. Figure 4.16 shows another example of classification map, achieved using PKDA2 to map into 2D the UCI data set acute-nephritis. In this case the κ slightly increases from 97.89% in $\mathbb{R}^n$, with $n = 6$ for this data set, to 100% in 2D. The PHP platform allows the user to save and printout the results, and to compile the user results (classification maps and κ for nD and 2D patterns) from previous executions (Figure 4.17), and to zoom in the classification map canvas (Figure 4.18).

4.3.2 Matlab standalone interface

The equivalent of Figure 4.14 in the Matlab interface is shown by Figure 4.19, which in order to upload a data set only requires the name of the CSV file, because the remaining details are extracted from the data set by the Matlab program. The aspect of this interface after mapping a data set (Figure 4.20) is very similar to the PHP interface, but without the user management funcionalities. The Matlab interface includes the 2D classification map created by the SVM, the 2D mapped patterns, and the confusion matrices achieved by SVM using the nD and 2D patterns, alongside with their respective κ values. In the Figure 4.20, the classification map is achieved by PKDA2 for the UCI data set breast-tissue, with $n = 9$ inputs and C=6 classes. There is a reduction of 15.9% in the κ value from 64.43% to 48.46% in $\mathbb{R}^n$ and 2D, respectively, but the 2D map is quite explicative, because the classes are fairly separated,

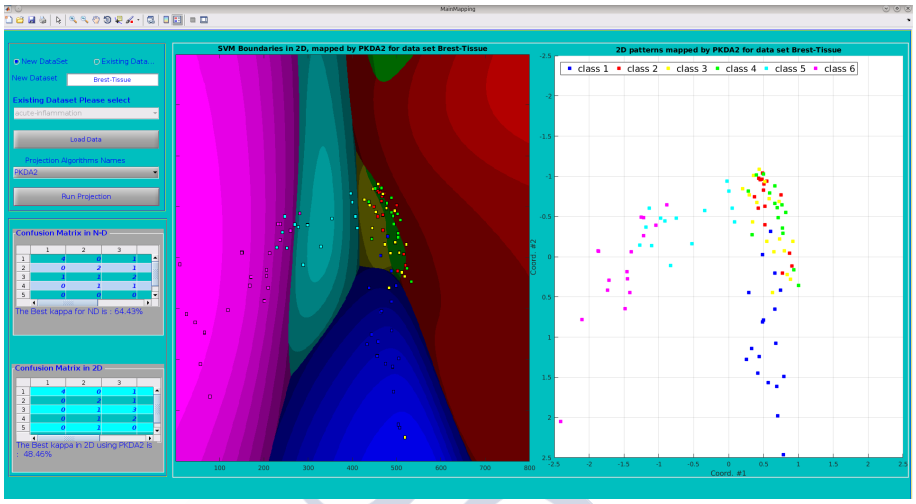


Figure 4.20: Screenshot of the Matlab interface after mapping a new data set (breast-tissue) using PKDA2.

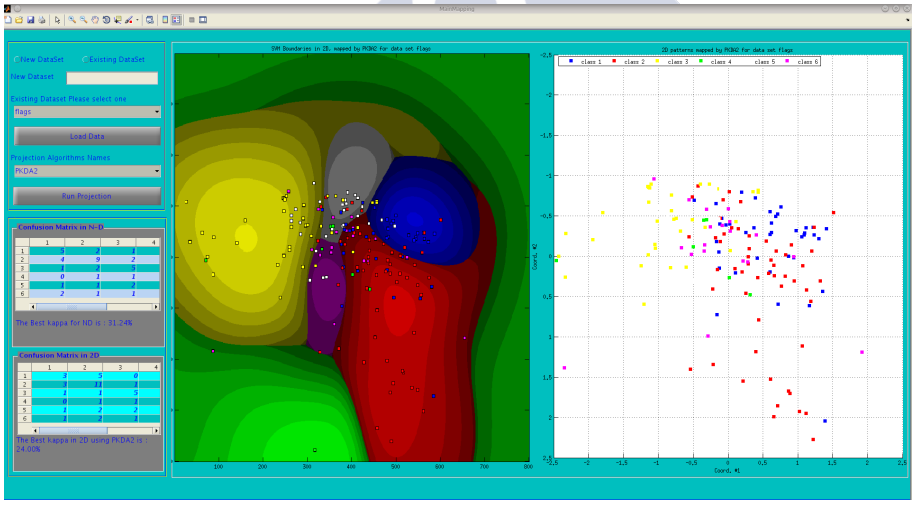


Figure 4.21: Screenshot of the Matlab interface after mapping the UCI data set flags using PKDA2.

although there is some overlap between classes 2, 3 and 4 (red, yellow and light green). It is also interesting the evolution (from left to right) from class 6 (magenta) to 5 (dark green) and then to 1 (blue), downwards to the right, and 2, 4 and 3 (yellow, light green and red)

upwards to the right. Figure 4.21 shows another example achieved mapping the UCI data set flags (n=28 inputs and C=6 classes) with the PKDA2 mapping. In this case, there is a slight reduction in the κ from 31.2% to 24% using nD and 2D patterns respectively.

Chapter 5

Automatic prediction of indoor building temperature

The demand for energy has drastically increased in the last few decades and large amounts of fossil fuel have to be burned to generate the required energy. The burning of coal, fuel oil, and natural gas is the largest source of carbon emissions, which causes the greenhouse effect, contributes to global warming and has subsequently affected the environment negatively. Moreover, the last few years has seen an overall increase in household consumption, especially in space heating systems [33], which has increased gas emission levels from the burning fossil fuels.

Because the cities of the world have the highest level of energy consumption, and thus produce huge amounts of greenhouse gas emissions, they constitute one of the most important factors contributing to climate change and its efects. As climate change affects all the world regions, it requires serious international actions to mitigate its footprint. The EU is currently leading several initiatives focused on reducing its effects. Specifically, the "Climate Action" aims to decrease the greenhouse gas emissions by improving the energy efficiency to reduce the consumption. The objectives are to reduce the greenhouse gas emissions by at least 20% and 40%, and the amount of energy used by 20% and 27%, by 2020 and 2030, respectively, below the 1990 levels [22].

As a consequence, the European Climate Action initiatives have been set up with the aim of tackling the negative effects of climate change and reducing energy consumption. Urban infrastructures have been identified as the key to execute the road-map of the EU, because

they have the highest energy demand, being responsible for a high percentage of global gas emissions. The development of efficient building energy management systems is a key factor in achieving the objectives because buildings represent 40% of energy consumption and 36% of total CO_2 emissions within the Union [133]. In order to achieve the objectives of the road-map, accurate models of the current energy systems and efficient energy management plans for buildings must be developed. Specially, modeling the HVAC (Heating, Ventilation, and Air Conditioning) systems is essential, since they have a relevant impact on both energy consumption and building comfort (around 70% of this consumption in a typical household is devoted to space heating).

The University of Santiago de Compostela (USC), in the framework of the "Climate Action" initiative, has been involved in the European Opere project [70] with the objective of improving the university energy management systems. This project enabled to deploy a sensor network in 45 university buildings that generates more than 10,000 signals. In particular, our research interest is focused on the Centro Singular de Investigación en Tecnoloxías da Información da Universidade de Santiago de Compostela (CiTIUS), which is a research building with a medium-size sensor network that produces 667 signals every 10 seconds. This raw data set constitutes a rich source of information that can be used to improve energy efficiency, to detect system failures and to optimize resources.

In this chapter we present two experiments, whose common goal is to forecast the indoor temperature in a CiTIUS office for three horizons: one, two and three consecutive hours. The data set used in both experiments are provided by CiTIUS sensors and the closest Meteogalicia weather station[1] (Santiago-EOAS). The first experiment compares 40 regressors belonging to 20 families over these three horizons to choose the most accurate regressor, which will be used in next experiment. The second experiment develops three neural networks for modelling the HVAC system of the CiTIUS. These networks are dynamic and perform online learning, and were trained in two different weather conditions, a rainy and a dry winter corresponding to years 2015-2016 and 2016-2017, respectively. They have been evaluated in terms of prediction accuracy as well as tolerance to both climate variability and sensor noise, in order to implement an energy plan for the whole building for the next three consecutive hours [3].

The remainder of this chapter is organized in the following manner. The next section reviews some previous work in this field. Section 5.2 demonstrates the regression algorithms used to develop the experiments. Sections 5.3 and 5.4 report the data set used in the ex-

[1] http://meteogalicia.es

periments and the experimental setup. Finally the results and discussion are shown in section 5.4.2.

5.1 Related work

The management of HVAC systems is necessary for optimizing building heating systems and to improve the energy efficiency. Previous studies have shown that machine learning algorithms can be used to model energy system in general and HVAC systems in particular [33]. Simon et al. incorporated a neural network into an intelligent model that modulates the building cooling load. This allows to forecast and to analyze the energy demand of the building [64]. Furthermore, it allows to find critical factors that have a vital influence on the energy consumption. The study reveals that the use of building occupancy plays a significant factor to forecast the cooling load of the HVAC system. Nguyen et al. investigated the impact of user's activities and behaviors on potential energy saving in smart buildings [84]. This work categorizes the user's most valuable activities and their influences in energy demands into three primary subsystems: HVAC, light, and plug load systems. Moreover, Varick et al. studied the influence of building occupancy in energy saving using real-time data. He succeeded in developing an occupancy model that could be integrated into a building HVAC system through Markov Chains. The study showed that 42% of consumed energy can be saved annually [37]. Zhao also discussed the impact of external factors on the building energy performance by summarizing various energy prediction approaches that were implemented through machine learning (ML) algorithms, and reviewing the engineering and statistical methods that were used to predict the building energy consumption [133]. A new prediction model based on Support Vector Regression (SVR) was developed to forecast hourly cooling load inside office buildings [69]. The model parameters were tuned to obtain the optimal settings to get the best temperature prediction. The comparison between the developed model and the classical multi-layer perceptron neural network (MLP) demonstrated that the SVR got the highest prediction accuracy and lower mean squared error (MSE). The study of Dong investigated the feasibility of applying SVR for regression in the forecasting building energy consumption, and the influence of different SVR parameters on the prediction accuracy [30]. The study reveals that the SVR get highest results compared with other related research methods using neural networks and genetic programming.

The influence of the external weather variables on the indoor temperature prediction has been examined through autoregressive model (ARX) and Autoregressive Moving Average model (ARMAX), where the selection of the appropriate structure of both models has been determined to achieve the best forecasting accuracy. The dynamic structure of these models can become an adaptable controller, which allows to increase the occupant's comfort level in the building and to reduce the energy consumption of HVAC systems [90]. The results showed that the ARX model obtained the best prediction accuracy. Abdullatif et al. proposed a new cooling load prediction model for buildings using the generalized regression neural network (GRNN). Occupancy and orientational characteristics were considerated in order to optimize HVAC thermal energy storage in buildings [8]. In the research conducted by Catalina, the researcher developed polynomial regression models using neural networks to predict heating demand for residential buildings monthly, considering the residential constructional structure [16]. The developed models were validated with 270 different scenarios to find the best approach. Several other recent studies proposed approaches using different ML algorithms for predicting building energy consumption [35, 96, 133, 81]. In these studies, external factors such as building structure, orientation, isolation, environmental variables and multiple parameters were considered. The numerical results revealed that these factors have a significant effect on the indoor temperature prediction and the energy consumption in a building.

The GA-ANFIS was another model used to forecast the indoor building temperature [68]. This approach uses genetic GA to optimize the fuzzy if-then rule base by finding the best configuration of the subtractive clusters. The adaptive network based fuzzy inference system (ANFIS) adjusts the premise and subsequent parameters to match the training data. The results demonstrated that GA-ANFIS achieved higher performance levels compared to neural networks in terms of prediction accuracy. Neural networks were used to forecast long-term energy demand during the day for Swedish buildings based on short-term data measurements [85], and to predict short-term indoor temperatures aiming at reducing the HVAC energy consumption [129]. In this work, the initial model parameters were estimated by using a random model or unbiased prior knowledge. The integration of the on-line learning process made this module adaptable to new patterns. The experiments shown that these two algorithms achieved better forecasting performance compared to neural networks. Recently, Rodríguez-Mier et al. used FRULER-GFS (fuzzy rule learning through evolution for regression-genetic fuzzy system) to develop a rule-based model for predicting the indoor building temperature. The knowledge bases learned by FRULER comprise Takagi-Sugeno-Kang fuzzy rules that

correctly forecast the temperature dynamics measured by a number of different predictors collected from both indoor and outdoor the building. The experiment results illustrated that FRULER-GFS achieved the best accuracy rate compared with ElasticNet and random forest regressors [91].

Doukas et al. developed an integrated decision support system based on a set of rules designed to improve the building energy management system [32]. The system allows central control over energy consumption in the building, which makes it extremely flexible. Furthermore, expert knowledge was used in the system to create a reliable energy profile. The HVAC control optimization (On/Off) provided the system with the ability to detect and eliminate any wrong decision. The study demonstrates that the expert knowledge has a significant impact on improving the energy management of the building.

5.2 Regression methods

We selected 40 regressors belonging to 20 different families (Table 5.1) to develop this comparative study. The majority of these regressors were chosen from the Classification And Regression Training (caret)[2] package of the R statistical computing language[3]. Rather than use the interface provided by the caret package, we executed the regressors directly using the corresponding R packages detailed below in order to control and tune the model execution. We also used other three popular methods implemented on other platforms: support vector regression (SVR) using the LibSVM library implemented in C++ [4]; GRNN and extreme learning machine (ELM) with Gaussian kernels, both implemented in the Matlab scientific language[5]. The hyper-parameters for each regressor were tuned in order to find the best values to train the regressor, trying the values proposed by the caret documentation. Their corresponding tunable hyper-parameters, number of values used for tuning and required packages are reported in Table 5.3. The specific values used for tuning each hyper-parameter are listed by utilities of the caret R package, and they should be adequated for each regressor and hyper-parameter. A brief description of these 40 regressors is shown in the following list, grouped by families of regressors.

[2] http://topepo.github.io/caret/train-models-by-tag.html.
[3] http://r-project.org.
[4] https://www.csie.ntu.edu.tw/∼cjlin/libsvm.
[5] http://mathworks.com.

No.	Family	Regressors	No.	Family	Regressors
1	Linear regression	lm	11	Boosting ensembles	randomGLM
2	Generalized linear regression	glm penalized glmnet	12	Gradient boosting machines	BstLm bstSm bstTree gbm
3	Least squares	nnls	13	Random forests	rf qrf extraTrees
4	Partial least squares	spls simpls	14	Prototype models	cubist
5	Least absolute shrinkage	lasso	15	Bayesian models	bayesglm brnn bartMachine
6	Ridge regression	foba	16	Independent Component Analysis	icr
7	Neural networks	MLP avNNet grnn elm dnn elm-kernel	17	Generalized additive models	gam
8	Support vector regression	svmRadial	18	Gaussian processes	gaussprLinear gaussprPoly
9	Regression trees	rpart M5	19	Quantile regression	rqlasso
10	Bagging ensembles	bag bagEarth treebag	20	Other methods	earth ppr

Table 5.1: Regressors considered in this work, grouped by families.

1. ***Linear regression***: this family includes the linear model **lm** [17]. It is widely used to do regression, variance analysis and covariance analysis. Collinear inputs show undetermined regression factors so they are discarded for this model and other regressors. We assume the response variable to be normally distributed and the link function to be the identity function.

2. ***Generalized linear regression***: includes the following regressors:

   2.1 **glm** is the generalized linear model, which combine a number of statistical models that include linear, logistic and Poisson regression [29]. The GLM is used for analyzing linear and non-linear effects of continuous and/categorical predictors on a discrete or continuous response variable. It allows the model to be related to the response variable via a link function and the distribution to be a function of its predictor.

2.2 **penalized** is the penalized linear regression, where a penalty added to control properties of the regression coefficients in order to optimize the likelihood instead of just maximizing it. The penalty L1 (the least absolute shrinkage and selection operator - lasso), penalizes the sum of absolute values of the coefficients, thus shrinking the irrelevant coefficient values to get close to zero. The L2 penalty (the "ridge" penalty), penalizes the sum of squared coefficients, reducing the impact of input collinearity. The regression is regularized by weighting both penalties [47].

2.3 **glmnet** is the lasso and elastic-net regularized generalized linear models [102]. The glmnet model fits a generalized linear model via penalized maximum likelihood. It is an extremely fast and efficient algorithm. It fits linear regression, logistic and multinomial, Poisson, Cox, grouped multinomial and multiple-response Gaussian regression models. The only tunable hyper-parameter is the elastic-net penalty mixing parameter α, which brings LASSO and ridge regression for α=1 and α=0, respectively.

3. ***Least squares***: the **nnls** is the non-negative least squares regression, which finds arg $\min_{\mathbf{x}} |\mathbf{Ax} - \mathbf{b}|$ subject to $\mathbf{x} \geq 0$ using the method proposed by [66].

4. ***Partial least squares***, composed by:

   4.1 **spls** is the sparse partial least squares regression, a dimension reduction method which allows both sparse variable selection and dimension reduction simultaneously. It introduces sparse linear combinations of the inputs in order to avoid lack of consistency with high dimensional patterns [19].

   4.2 **simpls** is the Statistically Inspired Modification of PLS [94]. This method directly calculates the PLS factors as linear combinations of the inputs thus truly maximizing the covariance criterion under orthogonality and normalization constraints.

5. ***Least absolute shrinkage and selection operator (LASSO)***. It diminishes the sum of squared errors with a limit on the sum of the absolute coefficients. It uses the `enet` function in the `elasticnet` R package.

6. ***Ridge (or Tikhonov) regression*** is is the most commonly used method of approximating answers for problems that do not have a unique solution. We used the **foba** regressor, which develops regression with forward, backward and sparse input selection [130].

This method does a backward step when the ridge penalized risk increase is less than ν (by default= 0.5) times the ridge penalized risk reduction in the matching backward step.

7. ***Neural networks***. This family includes the following regressors:

   7.1 **MLP** is the classical multi-layer perceptron, a feed-forward neural network used for pattern classification and regression, composed by three layers of nodes (input, hidden and output) and trained using the back-propagation learning algorithm. We used the MLP implementation provided by the Matlab neural network toolbox.

   7.2 **avNNet** is an averaged neural network committee (`caret` package). It fits an ensemble of 5 feed-forward neural networks, whose weights are randomly initialized with different seeds for each network in the ensemble. For regression, the outputs are averaged over the 5 networks. The hyper-parameters are the network size and the weight decay.

   7.3 **grnn** is the generalized regression neural network implemented by the Matlab neural network toolbox [106]. It is a one-pass learning algorithm with an extremely parallel structure, which uses Gaussian functions for the output estimation.

   7.4 **elm** is the extreme learning machine (ELM), a classical feed-forward neural network with a single hidden layer [56]. Its unique characteristics include very fast training, good generalization, and the ability to perform universal classification and function approximation. The tunable hyper-parameters are the number of hidden neurons and the activation function.

   7.5 **elm-kernel** is the extreme learning machine (ELM)[6] neural network with Gaussian kernel implemented in Matlab [56].

   7.6 **dnn** is the deep belief neural network implemented by the `DeepNet` R package, with three hidden layers whose number of neurons are tuned [52].

8. ***Support vector regression***, named **svmRadial** in the current study [103]. It uses a Gaussian kernel, tuning the kernel spread σ and regularization parameter C.

9. ***Regression trees***, including:

[6] http://extreme-learning-machines.org

9.1 **rpart** is the recursive partitioning and regression tree [13], which iteratively creates a regression tree by splitting the inputs an unspecified number of times until an ending condition is reached.

9.2 **M5** is the multivariate linear tree-based model (`RWeka` package) created using the M5 algorithm [89].

10. ***Bagging ensembles***, composed by:

10.1 **bag** is the bagging ensemble of conditional inference regression trees [11], averaging their outputs in order to produce the result for a test pattern.

10.2 **bagEarth** is the bagged multivariate adaptive regression splines (MARS) a bagging ensemble of MARS base regressors which computes an `earth` model for each bootstrap sample of the training set (25 samples by default). The only hyper-parameter is the maximum number of values in the pruned regression model.

10.3 **treebag** is a bagging ensemble of regression trees (CART), eacn one trained on a different bootstrap sample (25 samples by default).

11. ***Boosting ensembles***, includes **randomGLM**, a boosting ensemble of generalized linear models [104], trained each one on a bootstrap sample (100 samples), and randomly selecting inputs and interaction terms among them depending on the maximum interaction order.

12. ***Gradient boosting machines***, includes:

12.1 **BstLm** is the gradient boosting machine with linear regressors as base learners regressors, tuning number of boosting iterations [41, 73, 21].

12.2 **bstSm** is the gradient boosting machine with smoothing splines as base regressors with the same tunable parameter [97].

12.3 **bstTree** is the gradient boosting with regression base trees. It creates a sequence of simple trees which are built so that they predict the residuals of the previous trees [73, 21].

12.4 **gbm** is the generalized boosting model, which iteratively adds basis functions in a greedy fashion so that each additional basis function further reduces the selected loss function [41]. The tunable hyper-parameters are the maximum depth of input

interactions and the number of trees for prediction (5 values each one). We use a Gaussian distribution and shrinkage equal to 0.1.

13. ***Random forests***, composed by:

    13.1 **rf** is the random forest ensemble of random regression trees, whose output is the average of the regression trees outputs [12]. Its only hyper-parameter is the number of inputs randomly selected at each tree (`mtry`).

    13.2 **qrf** is the quantile regression forest, a tree-based ensemble which generalizes random forest in order to estimate conditional quantile functions [75]. The only tunable hyper-parameter is `mtry`.

    13.3 **extraTrees** is the ensemble of extremely randomized regression trees [45] . Its tunable hyper-parameters are `mtry` and the minimum sample size to split a node (`numRandomCuts`).

14. ***Prototype models***: this family includes **cubist**, a M5 rule-based model with corrections based on nearest neighbors in the training set [88]. Its hyper-parameters are the numbers of training committees and neighbors for prediction.

15. ***Bayesian models*** include:

    15.1 **bayesglm** is the Bayesian GLM, which uses the expectation maximization method to update the β values in GLM at each iteration, representing the prior information with an augmented regression [44]. The coefficients are calculated using a student-t prior distribution.

    15.2 **brnn** is the Bayesian regularized neural network, which determines the weights of two terms (squared error and squared sum of network weights) based on inference techniques [39, 72]. The weights are not normalized, and the number of hidden neurons is a hyper-parameter.

    15.3 **bartMachine** is the Bayesian additive regression tree [61]. The tunable hyper-parameters are the prior boundary (K) and the base value (α) in tree prior to decide if a node is terminal or not.

16. ***Independent component regression*** (**icr**), which fits a linear regression model using independent component analysis instead of the original inputs [57]. The only hyper-parameter is the number of independent components (`n.comp`).

17. ***Generalized additive model*** with splines (**gam**), which tunes the hyper-parameter is `select`, a boolean flag which decides whether an extra wiggliness penalty term to each function [123].

18. ***Gaussian processes regression*** [122]. This family includes:

    18.1 **gaussprLinear**, with linear kernel.

    18.2 **gaussprPoly**, with polynomial kernel, tuning the polynomial `degree` and `scale`.

19. ***Quantile regression***. This family includes the **rqlasso** regressor, which develops quantile regression with LASSO penalty [79]. This method fits a quantile regression model with the LASSO penalty, tuning the regularization hyper-parameter λ.

20. ***Other regression methods*** are:

    20.1 **earth** is the multivariate adaptive regression spline [40], whose only hyper-parameter is the maximum number of terms in the model (`nprune`).

    20.2 **ppr** is the projection pursuit regression [42], which tunes the number of terms (`nterms`) to be included in the final model.

5.3 Data acquisition

The data used to develop the experiments are provided by both sensor measurements linked to the HVAC system of the CiTIUS and by the closest Meteogalicia weather station. The patterns were obtained every 10 minutes sequentially during two time periods: from 1st October 2015 to 31st March 2016 (26,321 patterns) and from 1st November 2016 to 31st January 2017 (13,083 patterns). Both periods correspond to the CiTIUS HVAC winter working mode, which has the highest energy demand to warm the building. It must be noted that the second period corresponds to an unusually dry winter season in Galicia. Thus, the weather conditions in both time periods are very different.

The patterns are composed of 10 variables, 7 of them provided by the CiTIUS and the remaining are from Meteogalicia weather station. These variables are listed in Table 5.3. The data set is available on-line[7], so that the experiments can be repeated, if required.

[7] https://gitlab.citius.usc.es/cograde/HVAC-model.

Features	Abbr.	Type	Description
Underfloor Heating Status *	UHS	Binary	Status of the underfloor heating system in the office
Underfloor Heating Temperature *	UHT	Continuous	Temperature of the water linked to the underfloor heating system
Air Conditioning Status *	ACS	Binary	Status of the air conditioning system in the office
Air Conditioning Temperature*	ACT	Continuous	Temperature of the main air conditioning system.
Air Conditioning Humidity *	ACH	Continuous	Percentage of the humidity linked to the main air conditioning system
Humidity +	OutH	Continuous	Degree of the outdoor relative humidity
Temperature +	OutT	Continuous	Outdoor temperature
Solar radiation +	SR	Continuous	Level of solar radiation
Indoor temperature *	t	Continuous	Indoor temperature at time t
Previous indoor temperature *	$t-1$	Continuous	Indoor temperature at time $t-1$

Table 5.2: Pattern features, where (*) and (+) represent features from the CiTIUS HVAC and from Meteogalicia, respectively.

5.4 Experiments

We developed two different experiments with different objectives. In the first experiment (section 5.4.1), the 40 regressors described in section 5.2 were compared to select the most accurate one. In the second experiment (section 5.4.2), we developed new ML models based on neural networks that perform on-line learning and can be adapted at runtime in order to deal with both sensor noise and weather variability. Training and test were developed in both experiments using the data set described in section 5.3, that corresponds to two different time periods (9 months) and contains 39,404 patterns. The experiments were repeated 10 times for each regressor using different seeds for the random number generator, in order to generate new training and validation partitions each time. In the first experiment we randomly selected 2,000 patterns for training, 760 patterns for validation and the remaining patterns (36,644) for test. The data used to train and validate the models in the second experiment correspond to the first time period. The partitions were generated randomly in such a way that 85% and 15% of the patterns were used for training and validating the models, respectively, while the testing partitions correspond to the second time period (2016-2017). The data in both experiments were normalized to have 0 mean and 1 standard deviation. Moreover, the hyper-parameters of the regressors were tuned using the values listed in Table 5.3. The selected final values for the hyper-parameter are those which maximize the average performance (see below) over the validation sets.

Regressor	Hyperp. (values)	Packages	Regressor	Hyperp. (values)	Packages
lm	–	MASS	randomGLM	maxInterationOrder(3)	randomGLM
glm	–	gbm, plyr	bag	–	caret
penalized	$\lambda_1(5)$, $\lambda_2(4)$	penalized	BstLm	mstop(10)	bst, plyr
glmnet	$\alpha(7)$, $\lambda(3)$	glmnet	bstSm	mstop(10)	bst, plyr
nnls	–	nnls	bstTree	mstop(4), maxdepth(5)	bst, plyr
spls	$K(3)$,η, $\kappa(7)$	spls	gbm	n.trees(5) interaction.depth(5)	gbm, plyr
simpls	ncomp(10)	pls	rf	mtry(10)	randomForest
lasso	–	elasticnet	qrf	mtry(2)	quantregForest
foba	$k(2)$,$\lambda(10)$	foba	extraTrees	mtry(10) numRandomCuts(2)	extraTrees
MLP	n.hidden(20)	nnet	cubist	committees(3) neighbors(3)	Cubist
avNNet	size(7) decay(3)	nnet	bayesglm	–	arm
grnn	spread(14)	Matlab	brnn	neurons(15)	brnn
elm	nhid(20) actfun(4)	elmNN	bartMachine	$K(3)$,$\alpha(3)$	bartMachine
dnn	layer1(10) layer2(10) layer3(10)	deepnet	icr	ncomp(10)	fastICA
elm-kernel	$\sigma(25)$,$C(25)$	Matlab	gam	select(2)	gam
svmRadial	$\sigma(5)$,$C(4)$	kernlab	gausssprLinear	–	kernlab
rpart	complexity(10)	rpart	gausssprPoly	degree(5),scale(3)	kernlab
M5	pruned(2) smoothed(2) rules(2)	RWeka	rqlasso	$\lambda(10)$	rqPen
bagEarth	nprune(10)	caret	earth	nprune(15)	earth
treebag	–	ipred, plyr e1071	ppr	nterms(10)	stats

Table 5.3: List of the regressors, with their tunable hyper-parameters, values tried and packages.

The performance of the tested regressors was evaluated using the Pearson correlation, also known as R-coefficient, and the Mean Squared Error (MSE). The former is defined as:

$$\rho(\hat{Y},Y) = \frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{\overline{\hat{Y}_i - \mu_{\hat{Y}}}}{\sigma_{\hat{Y}}}\right)\left(\frac{Y_i - \mu_Y}{\sigma_Y}\right)$$

where $\mu_{\hat{Y}}$ and $\sigma_{\hat{Y}}$ are the mean and standard deviation of the regressor outputs $\hat{Y}$, respectively, while μ_Y represents the mean of the real output, σ_Y is the standard deviation of the real output Y and N is the number of test patterns. The MSE is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2$$

The average of both MSE and R-coefficient over the 10 repetitions have been calculated as a final performance measurement for all the compared approaches in both experiments.

5.4.1 Regressor comparison

An accurate and reliable planning of HVAC system leads to and efficient consumption of energy and higher comfort levels inside buildings. Machine learning algorithms that predict the development of the interior temperature of buildings based on internal and external climate conditions can be used to evaluate the effect of modifications in the HVAC systems to improve the building comfort. In order to develop an efficient model for HVAC systems, we compared the 40 regressors of the previous collection for temperature forecasting in a CiTIUS office using the three previous horizons (one, two and three hours). We used the set of real indoor and outdoor temperature measurements described in section 5.3.

We computed the Friedman rank [43] of the MSE and R-coefficient for the whole list of regressors in order to compare all the regressors accross the data set. This rank examines the current position of each regressor in average over all the horizons. The regressors must be sorted by decreasing performance (e.g., by increasing MSE or by decreasing R-coefficient) for each data set, and the Friedman rank of each regressor is its average position over the horizons. Figure 5.1 shows the Friedman rank for MSE and R-coefficient, by increasing rank (i.e., by decreasing performance). Two regressors of the random forest family (extraTrees and rf) achieve the best results for both performance measurements. In fact, there are small differences between both figures, just in the position of some regressors, such as cubist and averaged neural network committee (avNNet), and also between the generalized boosting model (gbm) and the Bayesian regularized neural network (brnn). For more details, Table 5.4 lists the ranks both for MSE and R-coefficient for each regressor, where the column "Avg. R-coefficient." reports the average R-Coefficient for each regressor over all the three horizons.

We also developed a post-hoc Friedman-Nemenyi [28] statistical test comparing the R-coefficient of the best regressor (extraTrees) with the remaining ones. This test is implemented in R by the `PMCMR` package [87]. The hypothesis tests are used to test the validity of a claim that is made about a population, which is called the null hypothesis. The alternative hypothesis is the one you would believe if the null hypothesis is concluded to be untrue. All the statistical hypothesis tests ultimately use the so-called p-value, which measures the degree of evidence of the null hypothesis. The p-value is a number between 0 and 1 which must be interpreted in the following way: a small p-value (typically $p < 0.05$) indicates strong evidence against the null hypothesis, which must be rejected within a 5% tolerance, because 0.05 is the 5% tolerance threshold commonly considered. Conversely, a large p-value (> 0.05) indicates weak evidence against the null hypothesis, which must be accepted. Finally, p-values very

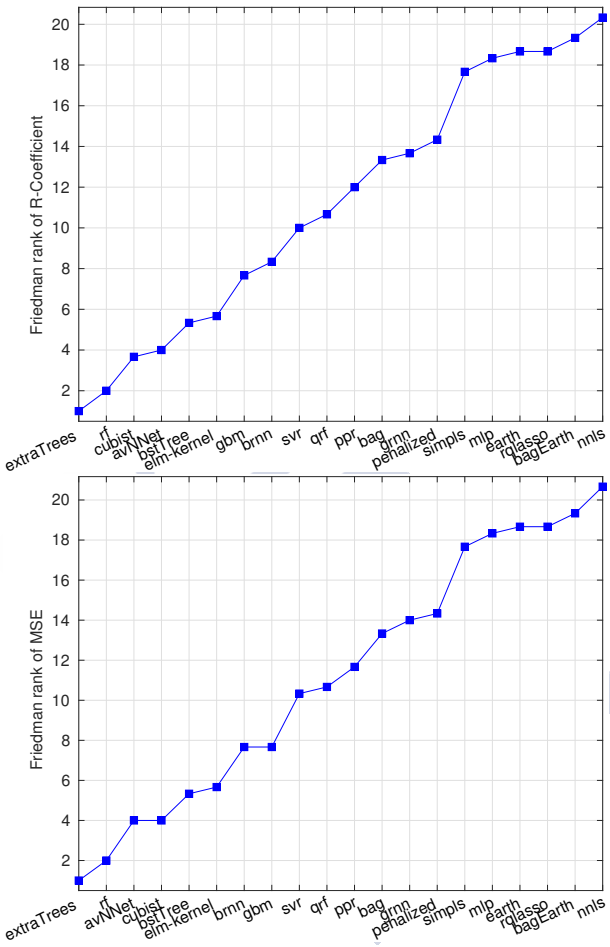


Figure 5.1: Friedman rank of R-coefficient (upper panel) and MSE (lower panel) for the 20 best regressors.

close to the threshold (0.05) are considered to be marginal (the decision might be any). The p-value of a test should always be reported in order to give a measurement of the conclusion reliability. In our case, we are comparing the best regressor (extraTrees) to the remaining ones, and the null hypothesis is that extraTrees and the other regressor behave similarly (i.e., their R-coefficients are not significantly different). Therefore, a p-value above (resp. below) 0.05 means that the null hypothesis must be accepted (resp. rejected), i.e., that extraTrees is not (resp. is) significantly better. The p-values listed in Table 5.4 show that extraTrees is only

	MSE rank		R-coefficient rank			
Order	Regressor	Rank	Regressor	Rank	R-coefficient Avg.	p-value
1	**extraTrees**	**1**	**extraTrees**	**1**	**0.97052**	**–**
2	rf	2	rf	2	0.96916	1.0000
3	avNNet	4	cubist	3.7	0.96801	1.0000
4	cubist	4	avNNet	4	0.96727	1.0000
5	bstTree	5.3	bstTree	5.3	0.96738	1.0000
6	elm-kernel	5.7	elm-kernel	5.7	0.96673	1.0000
7	brnn	7.7	gbm	7.7	0.96595	1.0000
8	gbm	7.7	brnn	8.3	0.96506	1.0000
9	svr	10.3	svr	10	0.96505	1.0000
10	qrf	10.7	qrf	10.7	0.96503	1.0000
11	ppr	11.7	ppr	12	0.96154	1.0000
12	bag	13.3	bag	13.3	0.96023	1.0000
13	grnn	14	grnn	13.7	0.9614	1.0000
14	penalized	14.3	penalized	14.3	0.95398	1.0000
15	simpls	17.7	simpls	17.7	0.9503	0.99972
16	mlp	18.3	mlp	18.3	0.95268	0.99934
17	earth	18.7	earth	18.7	0.93754	0.99903
18	rqlasso	18.7	rqlasso	18.7	0.95003	0.99903
19	bagEarth	19.3	bagEarth	19.3	0.92212	0.99797
20	nnls	20.7	nnls	20.3	0.94505	0.99457
21	BstLm	20.7	BstLm	21.3	0.94431	0.98728
22	lasso	21.7	lasso	21.3	0.94468	0.98728
23	bayesglm	25.3	bayesglm	25	0.93695	0.88219
24	elm	26	glm	26	0.93687	0.74914
25	glm	26.3	gam	27	0.93687	0.74914
26	spls	27.3	spls	27.3	0.89188	0.65318
27	gaussprLinear	27.3	gaussprLinear	27.3	0.93688	0.65318
28	gam	27.3	elm	27.3	0.93927	0.65318
29	M5	27.7	M5	27.7	0.91652	0.62295
30	lm	28.3	lm	28	0.93687	0.60281
31	treebag	29	treebag	29	0.93359	0.57645
32	rpart	29.3	rpart	29.3	0.92993	0.54622
33	icr	29.3	icr	29.3	0.93426	0.54622
34	randomGLM	29.7	randomGLM	29.7	0.84641	0.51607
35	foba	30	foba	30	0.93617	0.48616
36	dnn	35.7	dnn	35.7	0.70223	0.11552
37	bstSm	37	gaussprPoly	37.0	0.00101	0.07419
38	glmnet	38	bstSm	38.0	0.00000	0.05202
39	bartMachine	39	bartMachine	39.0	-0.00095	0.03580
40	gaussprPoly	40.0	glmnet	40.0	-0.00116	0.02420

Table 5.4: Friedman rank of the MSE (left) and R-Coefficient (right). The p-value (last column) of the Posthoc Friedman Nemenyi test compares the best regressor to the remaining ones.

significantly better than the 2 last regressors (bartMachine and glmnet), for which $p < 0.05$, so the differences between extraTrees and the first 37 regressors is not statistically significant.

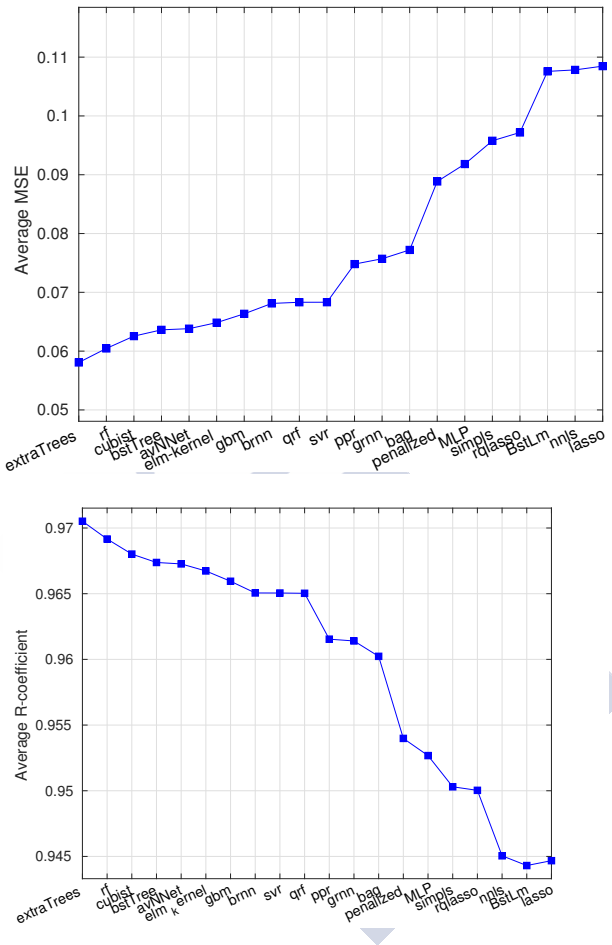


Figure 5.2: Average values of MSE and R-Coefficient over the data sets of the 20 best regressors to forecast three consecutive hours.

The average MSE and R-coefficient of the best 20 regressors over the three prediction horizons is shown in Figure 5.2, sorted decreasingly. ExtraTrees achieves the best average MSE (0.058) and R-coefficient (0.97), followed by rf and cubist. Both plots are similar to Figure 5.1, with bstTree swapped with avNNet to become 4th and 5th respectively. Also, the positions of bag and grnn are swapped to become 12th and 13th respectively. In the last 4 positions, rqlasso and nnls also improved their positions. Moreover, earth and bagearth

regressors disappear from the top 20, while lasso and BstLm replace them in the last two positions.

	One hour	Two hours	Three hours
MSE	0.04041	0.06011	0.07370
R-Coefficient	0.97958	0.96951	0.96245

Table 5.5: The best R-Coefficient and MSE are achieved by extraTrees for the forecasting horizon.

The conclusion of this experiment is that extraTrees achieves the best results in terms of Friedman rank, and also in terms of average values, for MSE and R-coefficient (Table 5.5). The difference between the R-coefficient achieved by extraTrees in the three prediction horizons is quite small (around 0.01), and the same happens for MSE, so its performance does not reduce very fast increasing the horizon. However, the statistical Friedman-Nemenyi post-hoc test shows that its difference with respect to the remaining regressors is not statistically significant. Other regressors with good performance are random forest, average neural network committee (avNNet), cubist, gradient boosting of regression trees (bstTree) and kernel ELM (elm-kernel). There is a high agreement between average values and Friedman ranks in the results. This comparison might be useful for any smart building in order to build a prediction model for improving the building management, efficiency and to assist in reducing energy consumption.

5.4.2 Online learning approaches

We also applied neural networks that allow automatic re-training to the temperature prediction. Automatic re-training provides robustness to the models and allows them to handle circunstances not encountered in the original training, such as exceptional climatic situations, or even withstand certain alterations on the system components resulting from errors or changes in the sensor devices. Concretely, these models were tuned to forecast three hours of the indoor temperature in a CiTIUS office. After that, we compared these models with extraTrees, which was the best regressor obtained from the experiments described on the previous subsection, using the same experimental methodology. We used different ways to train them in order to compare their accuracies in a real testing environment that is affected by different weather conditions and noise levels:

1. Adaptive multi-layer perceptron (AMLP). This approach was initially trained with historical data, but afterwards the AMLP was iteratively trained and adapted as it was processing new data. Concretely, this approach was trained every week with an incremental data set that includes both weekly and old data. Thus, the AMLP adapts to new scenarios, even changing the network architecture, in each training iteration.

2. Online learning multi-layer perceptron (OMLP). Trained initially with historical data, it is updated with each new pattern for a faster learning. It must be noted that there is a delay between the temperature prediction for a pattern and the use of that pattern to update the network. The reason is that, in order to use that pattern to update the network, the true indoor temperature associated to that pattern must be known, so it is necessary to wait for this value.

3. Online learning adaptive multi-layer perceptron (OAMLP). This is a combination of OMLP and AMLP: the network is updated with each new pattern, similarly to OMLP, and also with an incremental data set once a week, similarly to AMLP.

Experimental setup

We evaluated the three previous approaches using Matlab implementations and the data set described in section 5.3. Specifically, we used the first period (from 1^{st} October 2015 to 31^{st} March 2016) to both train and validate the models, and the second period (from 1^{st} November 2016 to 31^{st} January 2017) to test them. This methodology allowed us to evaluate the models with different weather conditions. Moreover, we also artificially generated other alternative test sets adding noise to the original one in order to evaluate different noise scenarios. Train and validation partitions were randomly generated in such a way that 85% of the patterns were used for training and the remainder for validation. Taking in mind a fair comparison, all the models were tuned with the same parameters during the training phase, listed in Table 5.6) and in the link in the above footnote 7.

All the developed models (AMLP, OMLP and OAMLP) are based on a classical MLP composed by three layers: input layer, hidden layer and output layer. The number of neurons linked to the hidden layer was fixed for each approach through empirical tests based on the "*ad hoc*" rule that this number should not be higher than the double of neurons of the input layer [107]. Each architecture was validated using the validation set. Finally, for each approach, the architecture with the best performance was selected for the final model. Given that these are

Parameter	Value
Network implementation function	`fitnet`
Input layer neurons	10
Output layer neurons	1
Hidden layer neurons	Empirical evaluation from 1 to 20 ($2 \times$ input_neurons)
Learning rate	0.01
Epochs	1000
Training function	Gradient descent backpropagation (traingd)
Activation function (hidden layer)	Sigmoid function
Activation function (output layer)	Linear function
Divide function	Divideind (index available in the link.)
Divide mode	Sample

Table 5.6: Training parameters used to develop the evaluated neural networks models

adaptive models, they had specific training processes. The AMLP approach was iteratively and incrementally trained every week using both weekly data (1,008 patterns) and historical data. Weekly patterns were distributed in each iteration between the train and validation sets (80% and 20%, respectively). These new enriched data sets were used to train and adapt the neural networks models. The OMLP approach was updated at runtime using the input patterns once the true indoor temperature was available. The architecture of this network remained unchanged during this process. Finally, the OAMLP approach was a hybrid model between the AMLP and the OMLP since it was updated at runtime and also trained and adapted once a week.

The noise sensitivity of the developed neural networks models was examined adding different noise levels (5%, 10%, 15%, 20% and 25%) to the test set (see the link in the above footnote 7), in order to simulate noise and temporary failures related to the sensors. Specifically, this noise was added using the Matlab Gaussian noise function (`awgn`). We repeated the experiments 10 times for each model using the same data set but using different random seeds (from 1 to 10) for generating different partitions and network initializations. After that, we averaged the results for each model.

Results and discussion

As a case-study, we have focused on the 3-hours indoor temperature forecast of a CiTIUS office. Figure 5.3 shows the MSE for each forecasting horizon (1, 2 and 3 hours) and approach (extraTrees, AMLP, OMLP and OAMLP) , which increases as the horizon grows. The lowest

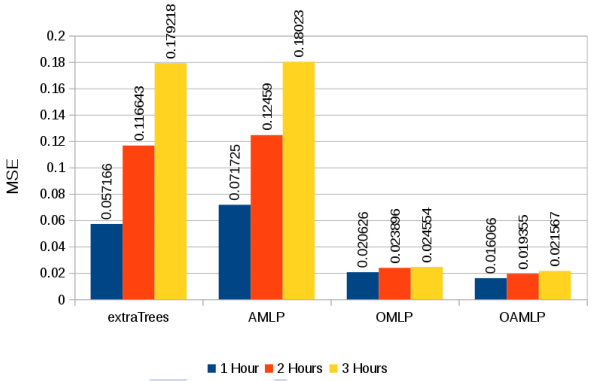


Figure 5.3: Value of MSE of the evaluated approaches without noise according to the forecasting horizon.

MSE is achieved by OAMLP in all the horizons, which is the approach with larger adaptation abilities. The OMLP achieves MSE only slightly higher, while extraTrees and AMLP achieve much higher MSE values. Therefore, it seems that the pattern updating is the feature which raises the performance for OAMLP and OMLP, while the weekly updating (AMLP) does not decrease MSE with respect to extraTrees, which is not updated at all, in fact extraTrees work slightly better than AMLP for the three forecast horizons.

Figure 5.4 plots the results of the four tested approaches in terms of R-Coefficient and MSE (left and right columns, respectively), for the noise levels considered. Results show that the approaches achieved a reasonable perfomance (high R-coefficient and low MSE) in all the evaluated scenarios, which decreases slowly with the noise level (decreasing R-coefficient and increasing MSE), thus exhibiting a high stability against noise. The best performance is achieved by OAMLP, followed very nearly by OMLP, while extraTrees, and specially AMLP, work much worse. Increasing the forecast horizon, the difference between OAMLP and OMLP reduces, in fact they behave equally for 3 hours. The difference between extraTrees and AMLP also reduces increasing the horizon. Increasing the noise level reduces slighly the performance, but this influence is reduced when the horizon increases, in fact the plots are almost constant with noise for 3 hours.

The performance improvement of OAMLP and OMLP with respect to extraTrees and AMLP is considerable, because the formers are continuously adapted using the patterns, which seems more useful in terms of performance than the weekly training of AMLP. Thus,

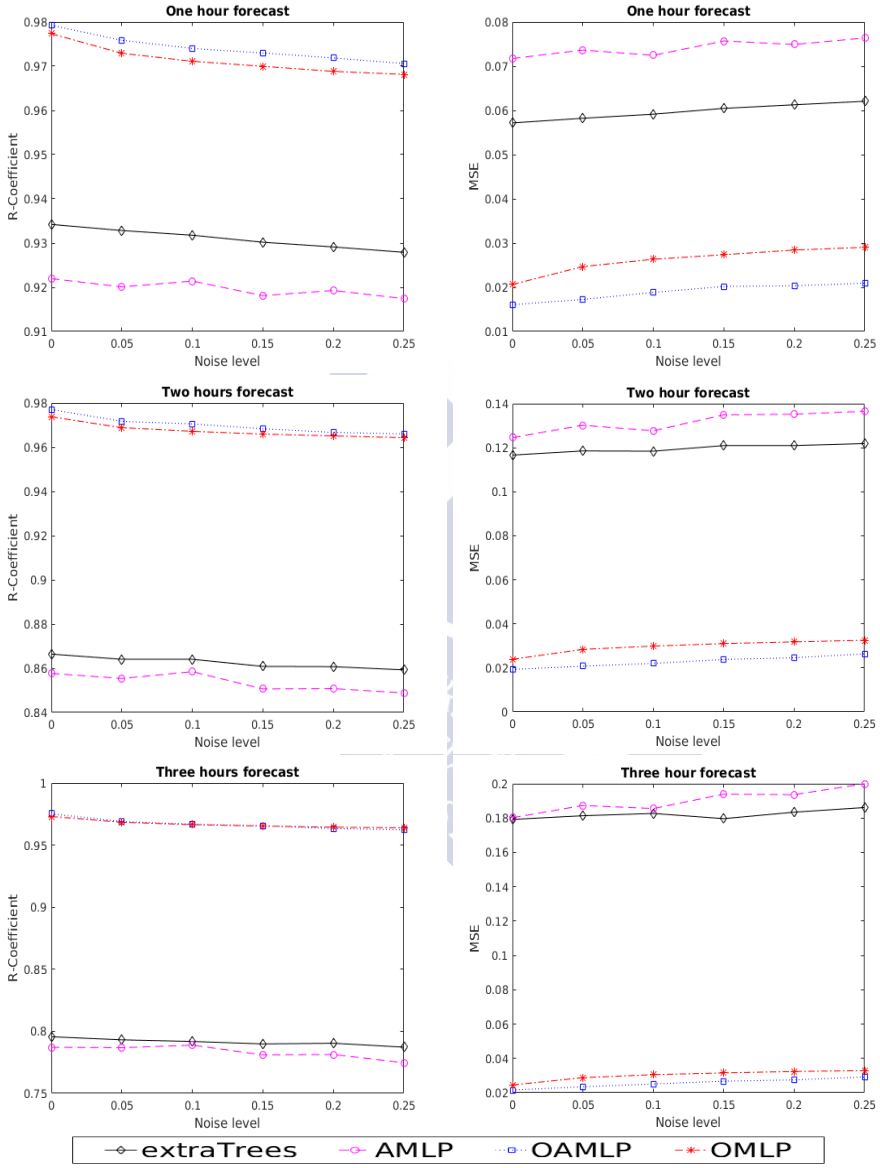


Figure 5.4: R-Coefficient (left column) and MSE (right column) of the evaluated approaches according to the forecasting horizon.

Noise %	Iteration											
	1	2	3	4	5	6	7	8	9	10	11	12
0	6	6	6	6	6	6	6	6	6	16	6	6
5	6	7	6	6	16	6	6	6	6	6	6	6
10	6	6	6	6	6	16	6	16	6	6	6	6
15	7	7	6	6	6	6	6	16	6	16	6	6
20	6	6	6	6	6	6	6	6	6	6	6	6
25	6	6	6	6	6	6	6	6	16	16	6	6

Table 5.7: Changes in the number of hidden neurons of OAMLP during the training.

the updating procedure of OAMLP and OMLP is able to deal with new scenarios, changing external conditions and noise. The slightly difference between OAMLP and OMLP must be caused by the absence of weekly re-training in OMLP, which sometimes fails under the influence of outliers and anomalies. It must be highlighted that both models OAMLP and AMLP can automatically change their architecture in each training iteration. Table 5.4.2 shows the different architectures used during the OAMLP training process. Despite 6 hidden neurons is the most used architecture, some iterations required a different number of neurons. We consider this a relevant characteristic since it allows models to be automatically improved as more data are obtained.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

CHAPTER 6

CONCLUSIONS

In the current thesis, we worked in the 2D visualization of multi-dimensional data for classification tasks, alongside with the application of a diverse collection of regression methods for the automatic prediction of indoor temperatures in smart buildings. With respect to the first objective, we propose 31 several mappings from $\mathbb{R}^n$ to $\mathbb{R}^2$ which allow the training of classifiers using the 2D mapped patterns. The proposed methods also map out-of-sample patterns, not seen during the mapping calculation, which can be classified without a high degradation in the performance. The mappings combine functions with different degrees (linear, quadratic including crossed products and polynomial with pure powers until degree 7) and minimize several measures of class overlap in $\mathbb{R}^2$: the J-index, which leads to Linear and Kernel Discriminant Analysis (DA), Direct class Separability (DS), Thornton (T) separability index and Class mean Distance (CD). These mappings only require the calculation of the leading eigenvectors of a matrix, defined by the overlap measure, whose order depends on the function degree, so they are very efficient. For each mapping, a Gaussian kernel LIBSVM classifier is trained on the 2D patterns mapped using the 31 proposed approaches (Table 3.1), and the Cohen κ is evaluated. These κ values are compared to the ones achieved using the original patterns and the 2D patterns mapped by the 34 techniques provided by the Matlab Toolbox for Dimensionality Reduction (Table 4.2). The mapping with the best κ is supposedly more useful for classification, since its mapped patterns are easier to classify. The experimental work shows that LIBSVM achieves the best results using the 2D data mapped by the 2nd-degree Polynomial Kernel Discriminant Analysis (PKDA2), with average κ=47.49%, which represents 82% of the κ achieved using the original n-dimensional data (57.9%). In fact, for 43 of

71 data sets (60.6% of the sets) the difference between svmNd and PKDA2 is below 10%. The LDA (κ=47.43%), provided by the Drtoolbox, and the CDOLM (linear CD mapping, 47.0%) also work well, but the remaining CD (46.1%), DS (43.8%) and T (33.0%) mappings are sub-optimal. Excepting LDA, the remaining Drtoolbox methods (both supervised or not) are also not competitive (PCA is the best with κ=42.58%). The CDOLM and PKDA2 are among the fastest mappings, similar to PCA and LDA respectively (although PKDA2 has degree 2 and LDA is linear), and much faster than most of the remaining Drtoobox mappings. These results show that PKDA2 visualizes 2D classification maps which: 1) retaining an important part (82%) of the performance with the original n-dimensional patterns; and 2) using the 2D mapped patterns, there is a probability around 0.6 of achieving κ between 90% and 100% of the performance using the original patterns. The future work includes to develop mappings with alternative overlap measures, in order to reduce the accuracy gap between the classification in $\mathbb{R}^n$ and $\mathbb{R}^2$; and to design a mapping update procedure which includes new training patterns, thus avoiding a complete re-calculation of the mapping.

With respect to the second objective, we compared a large collection composed by 40 regression methods of 20 different regressor families (Table 5.1) for indoor temperature forecasting, using the data set provided by both CiTIUS indoor sensors and the closest Meteogalicia weather station, over three forecasting horizons (temperature within 1, 2 and 3 hours). This comparison showed the good performances of extraTrees both for R-coefficient and MSE, and both in terms of average values and Friedman ranks. Other regressors with good performances are random forest, ensemble of multi-layer perceptron (MLP) neural networks, cubist, gradient boosted machines with regression trees and extreme learning machine with Gaussian kernels. Alongside with this comparison, three different machine learning approaches based on neural networks were developed to model the CiTIUS HVAC system and to forecast the indoor temperature of a CiTIUS office for the three horizons and for two weather scenarios (a rainy and a dry winter). These approaches are OMLP, AMLP and OAMLP, dynamic versions of MLP which allow adaptation using new patterns. Their performances were studied and compared taking into account different weather conditions and sensor noise levels. According to the results, an adaptive training approach is essential to model an HVAC system and to forecast the indoor temperature of the linked facility. The regressors trained off-line like extraTrees cannot deal with new environmental conditions and different noise levels. Nevertheless, adaptive approaches such as AMLP, OMLP and OAMLP can be automatically updated to different scenarios. However, we must take in mind that the use of on-line approaches

like OMLP can fall under the influence of outliers. Our results show that a hybrid approach between an on-line learning procedure and an incremental weekly training process achieve the best performance and is more stable in presence of noise. On the one hand, it can be fast updated to new scenarios, even when they are temporary. On the other hand, the deviations caused by short-lived events, outliers and model anomalies can be fixed when the network is trained once a week. Moreover, this approach also allows to change its architecture during the iterative training process, thus improving the model as more information is obtained. The future work includes the deployment of this model in all the CiTIUS facilities, in order to get a full system where different HVAC configurations can be tested and their effects can be evaluated. After that, we will develop an algorithm to automatically optimize the HVAC configuration.

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

APPENDIX A

CONFUSION MATRICES

This appendix compiles the confusion matrices achieved by SVM using the 2D mapped patterns using the best mapping for each data set.

No.	Confusion matrix $\mathbb{R}^N$	Confusion matrix $\mathbb{R}^2$
1	K=100 | 1 | 2 1 | 13 | 0 2 | 0 | 11 Se | 100.0% | 100.0% Pp | 100.0% | 100.0%	Method name = [PKDA2,LDA] K=100 | 1 | 2 1 | 13 | 0 2 | 0 | 11 Se | 100.0% | 100.0% Pp | 100.0% | 100.0%
2	K=100 | 1 | 2 1 | 13 | 0 2 | 0 | 11 Se | 100.0% | 100.0% Pp | 100.0% | 100.0%	Method name = [PKDA2,LDA] K=100 | 1 | 2 1 | 13 | 0 2 | 0 | 11 Se | 100.0% | 100.0% Pp | 100.0% | 100.0%
3	K=73.8 | 1 | 2 | 3 | 4 | 5 1 | 0.75 | 0.00 | 0.25 | 0.00 | 0.00 2 | 0.00 | 13.50 | 7.50 | 0.00 | 0.00 3 | 0.50 | 8.50 | 137.75 | 1.00 | 1.25 4 | 0.00 | 0.00 | 0.75 | 14.25 | 0.00 5 | 0.00 | 0.00 | 0.50 | 0.00 | 7.50 Se | 75.0% | 64.3% | 92.4% | 95.0 | 93.8 Pp | 60.0% | 61.4% | 93.9% | 93.4 | 85.7	Method name = [PKDA7] K=70.52 | 1 | 2 | 3 | 4 | 5 1 | 0.25 | 0.00 | 0.50 | 0.00 | 0.25 2 | 0.00 | 6.00 | 15.00 | 0.00 | 0.00 3 | 0.50 | 0.25 | 147.75 | 0.75 | 0.25 4 | 0.00 | 0.00 | 0.75 | 14.25 | 0.00 5 | 0.00 | 0.00 | 1.00 | 0.00 | 7.00 Se | 25.0% | 28.6% | 98.8% | 95.0 | 87.5 Pp | 33.3% | 96.0% | 89.5% | 95.0 | 93.3

Continued on next page

Table A.1 – *Continued from previous page*

4	K=96.7 | 1 | 2 | 3 1 | 11.75 | 0.25 | 0.00 2 | 1.50 | 70.50 | 0.00 3 | 1.25 | 0.00 | 70.75 Se | 97.9% | 97.9% | 98.3% Pp | 81.0% | 99.6% | 100.0%	Method name = [DSOPM2] K=84.73 | 1 | 2 | 3 1 | 8.25 | 2.25 | 1.50 2 | 4.00 | 67.00 | 1.00 3 | 3.00 | 2.00 | 67.00 Se | 68.8% | 93.1% | 93.1% Pp | 54.1% | 94.0% | 96.4%
5	K=62.5 | 1 | 2 1 | 1.75 | 0.25 2 | 0.25 | 0.75 Se | 87.5% | 75.0% Pp | 87.5% | 75.0%	Method name = [LDA] K=75 | 1 | 2 1 | 2.00 | 0.00 2 | 0.25 | 0.75 Se | 100.0% | 75.0% Pp | 88.9% | 100.0%
6	K=32.9 | 1 | 2 1 | 85.00 | 7.00 2 | 20.50 | 11.50 Se | 92.4% | 35.9% Pp | 80.6% | 62.2%	Method name = [CDOLM] K=25.56 | 1 | 2 1 | 85.00 | 7.00 2 | 22.50 | 9.50 Se | 92.4% | 29.7% Pp | 79.1% | 57.6%
7	K=30.6 | 1 | 2 1 | 39.25 | 7.75 2 | 9.75 | 8.25 Se | 83.5% | 45.8% Pp | 80.1% | 51.6%	Method name = [CDOPM7] K=34.36 | 1 | 2 1 | 41.50 | 5.50 2 | 10.25 | 7.75 Se | 88.3% | 43.1% Pp | 80.2% | 58.5%
8	K=91.7 | 1 | 2 1 | 52.00 | 4.00 2 | 0.75 | 58.25 Se | 92.9% | 98.7% Pp | 98.6% | 93.6%	Method name = [PKDA2] K=92.15 | 1 | 2 1 | 52.25 | 3.75 2 | 0.75 | 58.25 Se | 93.3% | 98.7% Pp | 98.6% | 94.0%
9	K=93.2 | 1 | 2 1 | 87.75 | 1.25 2 | 3.25 | 49.75 Se | 98.6% | 93.9% Pp | 96.4% | 97.5%	Method name = [PKDA2] K=93.57 | 1 | 2 1 | 87.50 | 1.50 2 | 2.75 | 50.25 Se | 98.3% | 94.8% Pp | 97.0% | 97.1%
10	K=33.5 | 1 | 2 1 | 30.00 | 7.00 2 | 5.00 | 6.00 Se | 81.1% | 54.5% Pp | 85.7% | 46.2%	Method name = [LDA] K=31.85 | 1 | 2 1 | 30.75 | 6.25 2 | 5.50 | 5.50 Se | 83.1% | 50.0% Pp | 84.8% | 46.8%

Continued on next page

Table A.1 – *Continued from previous page*

11

K=65.9	1	2	3	4	5	6
1	3.75	0.75	0.50	0.00	0.00	0.00
2	0.00	2.00	0.75	0.25	0.00	0.00
3	1.00	1.25	1.25	0.50	0.00	0.00
4	0.00	0.50	0.25	2.25	0.00	0.00
5	0.00	0.00	0.00	0.00	2.75	0.25
6	0.00	0.00	0.00	0.00	0.50	4.50
Se	75.0%	66.7%	31.2%	75.0%	91.7%	90.0%
Pp	78.9%	44.4%	45.5%	75.0%	84.6%	94.7%

Method name = [NPE]						
K=59.27	1	2	3	4	5	6
1	4.50	0.25	0.00	0.25	0.00	0.00
2	0.00	1.75	0.50	0.75	0.00	0.00
3	0.75	1.25	1.50	0.50	0.00	0.00
4	0.00	0.50	0.50	2.00	0.00	0.00
5	0.25	0.00	0.25	0.00	2.00	0.50
6	0.25	0.00	0.00	0.00	1.25	3.50
Se	90.0%	58.3%	37.5%	66.7%	66.7%	70.0%
Pp	78.3%	46.7%	54.5%	57.1%	61.5%	87.5%

12

K=83.9	1	2
1	52.00	4.00
2	2.25	26.75
Se	92.9%	92.2%
Pp	95.9%	87.0%

Method name = [LDA]		
K=89.58	1	2
1	53.75	2.25
2	1.75	27.25
Se	96.0%	94.0%
Pp	96.8%	92.4%

13

K=66.3	1	2
1	24.00	3.00
2	5.50	18.50
Se	88.9%	77.1%
Pp	81.4%	86.0%

Method name = [PKDA2]		
K=59.62	1	2
1	22.00	5.00
2	5.25	18.75
Se	81.5%	78.1%
Pp	80.7%	78.9%

14

K=71.6	1	2
1	67.75	8.25
2	16.00	79.00
Se	89.1%	83.2%
Pp	80.9%	90.5%

Method name = [PKDA7]		
K=71.87	1	2
1	64.25	11.75
2	12.00	83.00
Se	84.5%	87.4%
Pp	84.3%	87.6%

15

K=48.0	1	2
1	33.25	15.75
2	15.50	62.50
Se	67.9%	80.1%
Pp	68.2%	79.9%

Method name = [QKDA]		
K=42.06	1	2
1	28.75	20.25
2	13.75	64.25
Se	58.7%	82.4%
Pp	67.6%	76.0%

16

K=96.2	1	2	3	4	5	6
1	28.00	0.00	0.00	0.00	0.00	0.00
2	0.00	13.50	0.00	1.50	0.00	0.00
3	0.00	0.00	17.75	0.25	0.00	0.00
4	0.00	1.00	0.00	11.00	0.00	0.00
5	0.00	0.00	0.00	0.00	13.00	0.00
6	0.00	0.00	0.00	0.00	0.00	5.00
Se	100.0%	90.0%	98.6%	91.7%	100.0%	100.0%
Pp	100.0%	93.1%	100.0%	86.3%	100.0%	100.0%

Method name = [LDA]						
K=84.55	1	2	3	4	5	6
1	26.75	0.50	0.00	0.00	0.75	0.00
2	0.00	11.00	0.00	2.50	1.50	0.00
3	0.00	0.00	18.00	0.00	0.00	0.00
4	0.00	2.75	0.00	9.25	0.00	0.00
5	0.25	0.50	0.00	1.50	10.75	0.00
6	0.00	0.00	0.00	0.00	1.00	4.00
Se	95.5%	73.3%	100.0%	77.1%	82.7%	80.0%
Pp	99.1%	74.6%	100.0%	69.8%	76.8%	100.0%

Continued on next page

Table A.1 – *Continued from previous page*

17

K=61.8	1	2
1	21.25	0.75
2	4.00	6.00
Se	96.6%	60.0%
Pp	84.2%	88.9%

Method name = [PKDA2]

K=63.51	1	2
1	21.50	0.50
2	.00	6.00
Se	97.7%	60.0%
Pp	84.3%	92.3%

18

K=80.6	1	2	3	4	5	6
1	34.00	0.00	0.75	0.25	0.00	0.00
2	1.00	14.50	0.00	3.25	0.00	0.25
3	1.25	0.25	11.25	0.00	0.25	0.00
4	0.00	2.50	0.00	5.25	0.00	0.25
5	0.00	0.00	1.00	0.00	3.75	0.25
6	0.00	0.00	0.00	0.00	0.00	1.00
Se	97.1%	76.3%	86.5%	65.6%	75.0%	100.0%
Pp	93.8%	84.1%	86.5%	60.0%	93.8%	57.1%

Method name = [DSOLM]

K=72.86	1	2	3	4	5	6
1	34.00	0.25	0.75	0.00	0.00	0.00
2	1.75	16.75	0.25	0.25	0.00	0.00
3	1.25	0.25	11.00	0.00	0.50	0.00
4	0.00	6.50	0.25	1.25	0.00	0.00
5	0.00	0.00	3.25	0.00	1.75	0.00
6	0.00	0.00	0.00	0.00	0.00	1.00
Se	97.1%	88.2%	84.6%	15.6%	35.0%	100.0%
Pp	91.9%	70.5%	71.0%	83.3%	77.8%	100.0%

19

K=91.6	1	2	3
1	87.75	2.25	0.00
2	3.00	27.75	3.25
2	0.00	1.50	65.50
Se	97.5%	81.6%	97.8%
Pp	96.7%	88.1%	95.3%

Method name = [QKDA]

K=93.89	1	2	3
1	88.75	1.25	0.00
2	0.50	29.25	4.25
2	0.00	1.25	65.75
Se	98.6%	86.0%	98.1%
Pp	99.4%	92.1%	93.9%

20

K=91.6	1	2	3
1	87.75	2.25	0.00
2	3.00	27.75	3.25
2	0.00	1.50	65.50
Se	97.5%	81.6%	97.8%
Pp	96.7%	88.1%	95.3%

Method name = [QKDA]

K=93.89	1	2	3
1	88.75	1.25	0.00
2	0.50	29.25	4.25
2	0.00	1.25	65.75
Se	98.6%	86.0%	98.1%
Pp	99.4%	92.1%	93.9%

21

K=-4.5	1	2
1	21.00	1.00
2	2.00	0.00
Se	95.5%	0.0%
Pp	91.3%	0.0%

Method name = [PKDA3]

K=30.76	1	2
1	20.75	1.25
2	1.25	0.75
Se	94.3%	37.5%
Pp	94.3%	37.5%

22

K=38.0	1	2	3	4	5	6
1	5.50	2.50	0.50	0.00	1.00	0.50
2	2.00	10.25	1.00	0.00	1.25	0.50
3	0.50	2.25	5.00	0.00	1.25	0.00
4	0.00	1.00	0.25	0.00	0.00	0.75
5	0.50	1.25	1.00	0.00	2.75	0.50
6	1.00	1.50	0.50	0.00	0.25	0.75
Se	55.0%	68.3%	55.6%	0.0%	45.8%	18.8%
Pp	57.9%	54.7%	60.6%	0.0%	42.3%	25.0%

Method name = [PKDA3]

K=30.13	1	2	3	4	5	6
1	4.50	3.50	0.75	0.00	1.25	0.00
2	3.50	9.50	0.50	0.00	1.50	0.00
3	1.00	1.25	5.25	0.00	1.25	0.25
4	0.00	0.75	1.00	0.00	0.00	0.25
5	1.00	1.25	1.50	0.00	1.75	0.50
6	0.75	1.00	1.00	0.00	0.75	0.50
Se	45.0%	63.3%	58.3%	0.0%	29.2%	12.5%
Pp	41.9%	55.1%	52.5%	0.0%	26.9%	33.3%

Continued on next page

Table A.1 – *Continued from previous page*

23

K=51.8	1	2	3	4	5	6
1	12.50	3.25	1.25	0.00	0.00	0.00
2	4.75	12.50	0.25	1.00	0.25	0.25
3	1.25	1.75	1.00	0.00	0.00	0.00
4	0.25	1.00	0.00	1.25	0.00	0.50
5	0.00	0.75	0.00	0.25	1.00	0.00
6	0.00	0.50	0.00	0.50	0.25	5.75
Se	73.5%	65.8%	25.0%	41.7%	50.0%	82.1%
Pp	66.7%	63.3%	40.0%	41.7%	66.7%	88.5%

Method name = [TOLM]

K=48.18	1	2	3	4	5	6
1	13.50	3.50	0.00	0.00	0.00	0.00
2	4.25	13.00	0.00	0.25	0.00	1.50
3	2.25	1.00	0.00	0.25	0.00	0.50
4	0.50	1.50	0.00	0.75	0.00	0.25
5	1.00	1.00	0.00	0.00	0.00	0.00
6	0.25	0.50	0.00	0.00	0.25	6.00
Se	79.4%	68.4%	0.0%	25.0%	0.0%	85.7%
Pp	62.1%	63.4%	0.0%	60.0%	0.0%	72.7%

24

K=8.6	1	2
1	44.75	6.25
2	15.25	3.75
Se	87.7%	74.6%
Pp	19.7%	37.5%

Method name = [TOPM4]

K=19.54	1	2
1	43.00	8.00
2	12.75	6.25
Se	84.3%	32.9%
Pp	77.1%	43.9%

25

K=62.0	1	2	3
1	0.75	1.00	0.25
2	0.50	1.50	0.00
3	0.50	0.00	6.50
Se	37.5%	75.0%	92.9%
Pp	42.9%	60.0%	96.3%

Method name = [DSOQM]

K=48.73	1	2	3
1	0.50	0.50	1.00
2	1.25	0.75	0.00
3	0.00	0.00	7.00
Se	25.0%	37.5%	100.0%
Pp	28.6%	60.0%	87.5%

26

K=34.9	1	2	3	4	5
1	36.75	3.00	1.00	0.00	0.25
2	5.50	4.00	1.75	1.75	0.00
3	2.25	2.00	1.50	3.25	0.00
4	1.00	2.50	1.50	2.75	0.25
5	0.50	0.50	0.50	1.50	0.00
Se	89.6%	30.8%	16.7%	34.4%	0.0%
Pp	79.9%	33.3%	24.0%	29.7%	0.0%

Method name = [PKDA3]

K=39.48	1	2	3	4	5
1	37.50	1.75	1.00	0.75	0.00
2	6.00	3.25	1.00	2.75	0.00
3	1.75	2.00	1.75	3.50	0.00
4	1.25	0.50	1.50	4.50	0.25
5	0.25	0.25	0.25	2.25	0.00
Se	91.5%	25.0%	19.4%	56.2%	0.0%
Pp	80.2%	41.9%	31.8%	32.7%	0.0%

27

K=62.0	1	2
1	41.25	4.75
2	7.50	18.50
Se	89.7%	84.6%
Pp	71.2%	79.6%

Method name = [CDOPM3]

K=63.13	1	2
1	40.75	5.25
2	6.75	19.25
Se	88.6%	74.0%
Pp	85.8%	78.6%

Continued on next page

Table A.1 – *Continued from previous page*

28

K=9.9	1	2	3	4	5
1	0.25	1.25	0.25	0.25	0.00
2	1.00	6.25	3.50	1.25	0.00
3	0.25	4.00	3.00	0.75	0.00
4	0.00	3.00	2.00	2.00	0.00
5	0.00	0.25	0.00	0.75	0.00
Se	12.5%	52.1%	37.5%	28.6%	0.0%
Pp	16.7%	42.4%	34.3%	40.0%	0.0%

Method name = [PKDA2]					
K=18.23	1	2	3	4	5
1	0.00	1.75	0.00	0.25	0.00
2	0.50	9.75	0.50	1.25	0.00
3	0.25	5.50	0.75	1.50	0.00
4	0.00	3.25	0.25	3.50	0.00
5	0.00	0.25	0.00	0.75	0.00
Se	0.0%	81.2%	9.4%	50.0%	0.0%
Pp	0.0%	47.6%	50.0%	48.3%	0.0%

29

K=10.9	1	2	3	4	5
1	6.00	3.25	0.75	1.50	0.50
2	4.00	4.75	2.75	1.75	0.75
3	1.50	3.50	2.00	2.50	0.50
4	2.50	3.25	1.50	2.50	0.25
5	0.25	0.75	0.25	0.50	0.25
Se	50.0%	33.9%	20.0%	25.0%	12.5%
Pp	42.1%	30.6%	27.6%	28.6%	11.1%

Method name = [CDOPM3]					
K=12.55	1	2	3	4	5
1	5.75	5.50	0.50	0.25	0.00
2	4.00	8.50	1.00	0.50	0.00
3	1.75	4.75	2.00	1.50	0.00
4	2.00	6.00	1.50	0.25	0.25
5	0.00	0.25	0.75	0.50	0.50
Se	47.9%	60.7%	20.0%	2.5%	25.0%
Pp	42.6%	34.0%	34.8%	8.3%	66.7%

30

K=41.2	1	2
1	4.75	3.25
2	4.50	25.50
Se	59.4%	85.0%
Pp	51.4%	88.7%

Method name = [ProbPCA]		
K=61.3	1	2
1	5.25	2.75
2	2.00	28.00
Se	65.6%	93.3%
Pp	72.4%	91.1%

31

K=65.3	1	2
1	43.00	4.00
2	7.50	19.50
Se	91.5%	72.2%
Pp	85.1%	83.0%

Method name = [CDOPM5]		
K=54.87	1	2
1	42.50	4.50
2	10.25	16.75
Se	90.4%	62.0%
Pp	80.6%	78.8%

32

K=25.0	1	2
1	89.75	11.25
2	27.25	13.75
Se	88.9%	33.5%
Pp	76.7%	55.0%

Method name = [ProbPCA]		
K=21.13	1	2
1	77.00	24.00
2	22.50	18.50
Se	76.2%	45.1%
Pp	77.4%	43.5%

Continued on next page

Table A.1 – *Continued from previous page*

33

K=87.5	1	2	3	4	5	6	7
1	6.75	0.00	0.00	0.00	0.25	0.00	0.00
2	0.00	7.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	6.00	0.25	0.75	0.00	0.00
4	0.25	0.00	0.50	5.25	0.50	0.50	0.00
5	0.00	0.00	1.00	0.75	5.25	0.00	0.00
6	0.00	0.00	0.25	0.25	0.00	6.50	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	7.00
Se	96.4%	100.0%	85.7%	75.0%	75.0%	92.9%	100.0%
Pp	96.4%	100.0%	77.4%	80.8%	77.8%	92.9%	100.0%

Method name = [PKDA2]							
K= 75.6	1	2	3	4	5	6	7
1	5.25	0.00	0.00	0.00	1.75	0.00	0.00
2	0.00	7.00	0.00	0.00	0.00	0.00	0.00
3	1.75	0.00	5.00	0.00	0.25	0.00	0.00
4	0.75	0.00	0.00	5.50	0.50	0.25	0.00
5	2.75	0.00	1.00	0.25	3.00	0.00	0.00
6	0.25	0.00	0.00	0.75	0.00	6.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.00	7.00
Se	75.0%	100.0%	71.4%	78.6%	42.9%	85.7%	100.0%
Pp	48.8%	100.0%	83.3%	84.6%	54.5%	96.0%	100.0%

34

K=87.2	1	2
1	27.00	4.00
2	1.00	55.00
Se	87.1%	98.2%
Pp	96.4%	93.2%

Method name = [CDOQM]		
K=87.9	1	2
1	27.50	3.50
2	1.25	54.75
Se	88.7%	97.8%
Pp	95.7%	94.0%

35

K=95.8	1	2	3
1	12.00	0.00	0.00
2	0.00	11.25	0.75
3	0.00	0.25	11.75
Se	100.0%	93.8%	97.9%
Pp	100.0%	97.8%	94.0%

Method name = [LPP]			
K=93.75	1	2	3
1	12.00	0.00	0.00
2	0.00	11.50	0.50
3	0.00	1.00	11.00
Se	100.0%	95.8%	91.7%
Pp	100.0%	92.0%	95.7%

36

K=62.5	1	2
1	0.75	0.25
2	0.25	2.75
Se	75.0%	91.7%
Pp	75.0%	91.7%

Method name = [PKDA2]		
K=87.5	1	2
1	1.00	0.00
2	0.25	2.75
Se	100.0%	91.7%
Pp	80.0%	100.0%

37

K=34	1	2	3
1	0.25	1.75	0.00
2	0.00	3.00	0.00
3	0.00	1.00	1.00
Se	12.5%	100.0%	50.0%
Pp	100.0%	52.2%	100.0%

Method name = [PCA]			
K=38.39	1	2	3
1	1.00	0.75	0.25
2	1.00	2.00	0.00
3	0.00	0.75	1.25
Se	50.0%	66.7%	62.5%
Pp	50.0%	57.1%	83.3%

38

K=64.5	1	2
1	17.50	2.50
2	3.50	11.50
Se	87.5%	76.7%
Pp	83.3%	82.1%

Method name = [PCA]		
K=66.99	1	2
1	18.75	1.25
2	4.25	10.75
Se	93.8%	71.7%
Pp	81.5%	89.6%

Continued on next page

Table A.1 – *Continued from previous page*

39	K=62.6 | 1 | 2 1 | 74.75 | 12.25 2 | 17.25 | 55.75 Se | 85.9% | 76.4% Pp | 81.2% | 82.0%	Method name = [ProbPCA] K=57.81 | 1 | 2 1 | 70.25 | 16.75 2 | 16.75 | 56.25 Se | 80.7% | 77.1% Pp | 80.7% | 77.1%
40	K=61.5 | 1 | 2 1 | 9.75 | 3.25 2 | 1.75 | 11.25 Se | 75.0% | 86.5% Pp | 84.8% | 77.6%	Method name = [DSOPM7] K=69.23 | 1 | 2 1 | 10.75 | 2.25 2 | 1.75 | 11.25 Se | 82.7% | 86.5% Pp | 86.0% | 83.3%
41	K=46.7 | 1 | 2 1 | 11.25 | 3.75 2 | 4.25 | 10.75 Se | 75.0% | 71.7% Pp | 72.6% | 74.1%	Method name = [TOPM2] K=61.67 | 1 | 2 1 | 12.75 | 2.25 2 | 3.50 | 11.50 Se | 85.0% | 76.7% Pp | 78.5% | 83.6%
42	K=42.1 | 1 | 2 1 | 21.75 | 4.25 2 | 6.75 | 9.25 Se | 83.7% | 57.8% Pp | 76.3% | 68.5%	Method name = [DSOPM2] K=36.15 | 1 | 2 1 | 17.50 | 8.50 2 | 4.75 | 11.25 Se | 67.3% | 70.3% Pp | 78.7% | 57.0%
43	K=78.3 | 1 | 2 1 | 14.50 | 0.50 2 | 2.75 | 12.25 Se | 96.7% | 81.7% Pp | 84.1% | 96.1%	Method name = [DSOPM3] K=80 | 1 | 2 1 | 14.00 | 1.00 2 | 2.00 | 13.00 Se | 93.3% | 86.7% Pp | 87.5% | 92.9%
44	K=66.1 | 1 | 2 1 | 63.25 | 20.75 2 | 17.00 | 154.00 Se | 75.3% | 90.1% Pp | 78.8% | 88.1%	Method name = [LDA] K=56.78 | 1 | 2 1 | 57.00 | 27.00 2 | 20.75 | 150.25 Se | 67.9% | 87.9% Pp | 73.3% | 84.8%
45	K=82 | 1 | 2 | 3 1 | 57.25 | 0.00 | 6.75 2 | 0.00 | 13.00 | 2.00 3 | 10.75 | 1.75 | 162.50 Se | 89.5% | 86.7% | 92.9% Pp | 84.2% | 88.1% | 94.9%	Method name = [PKDA3] K=79.97 | 1 | 2 | 3 1 | 54.75 | 0.25 | 9.00 2 | 0.00 | 11.75 | 3.25 3 | 7.75 | 2.75 | 164.50 Se | 85.5% | 78.3% | 94.0% Pp | 87.6% | 79.7% | 93.1%

Continued on next page

Table A.1 – *Continued from previous page*

46	K=66 | 1 | 2 1 | 74.25 | 21.75 2 | 15.50 | 115.50 Se | 77.3% | 88.2% Pp | 82.7% | 84.2%	Method name = [LDA] K=52.8 | 1 | 2 1 | 64.75 | 31.25 2 | 20.25 | 110.75 Se | 67.4% | 84.5% Pp | 76.2% | 78.0%
47	K=86.5 | 1 | 2 | 3 1 | 126.25 | 0.00 | 4.75 2 | 0.00 | 2.50 | 0.50 3 | 8.50 | 1.50 | 83.00 Se | 96.4% | 83.3% | 89.2% Pp | 93.7% | 62.5% | 94.1%	Method name = [LDA] K=82.61 | 1 | 2 | 3 1 | 125.50 | 0.25 | 5.25 2 | 0.00 | 2.00 | 1.00 2 | 12.25 | 0.75 | 80.00 Se | 95.8% | 66.7% | 86.0% Pp | 91.1% | 66.7% | 92.8%
48	K=76.1 | 1 | 2 1 | 9.50 | 2.50 2 | 1.50 | 34.50 Se | 79.2% | 95.8% Pp | 86.4% | 93.2%	Method name = [PKDA5] K=63.8 | 1 | 2 1 | 7.75 | 4.25 2 | 1.75 | 34.25 Se | 64.6% | 95.1% Pp | 81.6% | 89.0%
49	K=46.5 | 1 | 2 1 | 111.25 | 13.75 2 | 30.25 | 36.75 Se | 89.0% | 54.9% Pp | 78.6% | 72.8%	Method name = [LDA] K=50.2 | 1 | 2 1 | 112.50 | 12.50 2 | 28.50 | 38.50 Se | 90.0% | 57.5% Pp | 79.8% | 75.5%
50	K=55 | 1 | 2 | 3 1 | 2.00 | 0.25 | 0.75 2 | 0.75 | 0.00 | 1.25 3 | 0.00 | 0.25 | 18.75 Se | 66.7% | 0.0% | 98.7% Pp | 72.7% | 0.0% | 90.4%	Method name = [DSOPM4] K=59.9 | 1 | 2 | 3 1 | 2.00 | 0.50 | 0.50 2 | 0.50 | 0.25 | 1.25 3 | 0.00 | 0.25 | 18.75 Se | 66.7% | 12.5% | 98.7% Pp | 80.0% | 25.0% | 91.5%
51	K=33.1 | 1 | 2 | 3 1 | 3.50 | 0.25 | 2.25 2 | 0.00 | 1.50 | 1.50 3 | 2.75 | 2.00 | 9.25 Se | 58.3% | 50.0% | 66.1% Pp | 56.0% | 40.0% | 71.2%	Method name = [TOPM5] K=41.27 | 1 | 2 | 3 1 | 3.50 | 0.25 | 2.25 2 | 0.50 | 0.50 | 2.00 3 | 1.00 | 0.50 | 12.50 Se | 58.3% | 16.7% | 89.3% Pp | 70.0% | 40.0% | 74.6%

Continued on next page

Table A.1 – *Continued from previous page*

52

K=27.2	1	2	3
1	0.25	1.25	0.50
2	0.75	10.50	0.75
3	0.00	4.25	2.75
Se	12.5%	87.5%	39.3%
Pp	25.0%	65.6%	68.8%

Method name = [DSOPM5]			
K=47.78	1	2	3
1	1.00	0.50	0.50
2	0.75	10.50	0.75
3	0.50	2.75	3.75
Se	50.0%	87.5%	53.6%
Pp	44.4%	76.4%	75.0%

53

K=14.1	1	2
1	19.25	1.75
2	2.25	0.75
Se	91.7%	25.0%
Pp	89.5%	30.0%

Method name = [QKDA]		
K=49.17	1	2
1	20.25	0.75
2	1.50	1.50
Se	96.4%	50.0%
Pp	93.1%	66.7%

54

K=44.7	1	2	3	4	5	6
1	2.00	0.75	0.25	0.00	0.00	0.00
2	0.50	0.25	0.50	0.50	0.25	0.00
3	0.75	0.25	9.00	0.25	0.75	0.00
4	0.00	0.00	0.75	1.50	0.50	0.25
5	0.00	0.00	0.75	0.50	0.75	0.00
6	0.25	0.00	0.00	0.75	0.50	0.50
Se	66.7%	12.5%	81.8%	50.0%	37.5%	25.0%
Pp	57.1%	20.0%	80.0%	42.9%	27.3%	66.7%

Method name = [QKDA]						
K=41.48	1	2	3	4	5	6
1	2.00	0.50	0.50	0.00	0.00	0.00
2	0.50	0.00	0.75	0.25	0.00	0.50
3	0.75	0.00	9.50	0.00	0.50	0.25
4	0.00	0.00	1.00	1.25	0.50	0.25
5	0.00	0.00	1.00	0.25	0.50	0.25
6	0.25	0.00	0.75	0.00	0.25	0.75
Se	66.7%	0.0%	86.4%	41.7%	25.0%	37.5%
Pp	57.1%	0.0%	70.4%	71.4%	28.6%	37.5%

55

K= -1	1	2
1	29.50	1.50
2	11.50	0.50
Se	95.2%	4.2%
Pp	72.0%	25.0%

Method name = [ProbPCA]		
K=13.15	1	2
1	24.50	6.50
2	8.00	4.00
Se	79.0%	33.3%
Pp	75.4%	38.1%

56

K= -2.5	1	2
1	12.75	0.25
2	5.00	0.00
Se	98.1%	0.0%
Pp	71.8%	0.0%

Method name = [MVU]		
K=17.54	1	2
1	10.25	2.75
2	3.00	2.00
Se	78.8%	40.0%
Pp	77.4%	42.1%

57

K=90.4	1	2	3
1	15.25	0.50	1.25
2	1.25	15.75	0.00
3	0.25	0.00	16.75
Se	89.7%	92.6%	98.5%
Pp	91.0%	96.9%	93.1%

Method name = [LDA]			
K=95.59	1	2	3
1	16.25	0.25	0.50
2	0.00	17.00	0.00
3	0.75	0.00	16.25
Se	95.6%	100.0%	95.6%
Pp	95.6%	98.6%	97.0%

Continued on next page

Table A.1 – *Continued from previous page*

58	K=20.4 | 1 | 2 1 | 7.75 | 1.25 2 | 4.00 | 2.00 Se | 86.1% | 33.3% Pp | 66.0% | 61.5%	Method name = [CDOPM7] K=35.23 | 1 | 2 1 | 7.50 | 1.50 2 | 3.00 | 3.00 Se | 83.3% | 50.0% Pp | 71.4% | 66.7%
59	K=52.5 | 1 | 2 1 | 8.25 | 1.75 2 | 3.00 | 7.00 Se | 82.5% | 70.0% Pp | 73.3% | 80.0%	Method name = [DSOPM4] K=55 | 1 | 2 1 | 8.50 | 1.50 2 | 3.00 | 7.00 Se | 85.0% | 70.0% Pp | 73.9% | 82.4%
60	K=5.7 | 1 | 2 1 | 8.25 | 46.75 2 | 12.00 | 105.00 Se | 15.0% | 89.7% Pp | 40.7% | 69.2%	Method name = [GDA] K=8.04 | 1 | 2 1 | 21.25 | 33.75 2 | 35.75 | 81.25 Se | 38.6% | 69.4% Pp | 37.3% | 70.7%
61	K=40.8 | 1 | 2 1 | 154.00 | 21.00 2 | 37.25 | 37.75 Se | 88.0% | 50.3% Pp | 80.5% | 64.3%	Method name = [LDA] K=41.11 | 1 | 2 1 | 155.50 | 19.50 2 | 38.00 | 37.00 Se | 88.9% | 49.3% Pp | 80.4% | 65.5%
62	K=68.7 | 1 | 2 1 | 33.50 | 3.50 2 | 6.75 | 23.25 Se | 90.5% | 77.5% Pp | 83.2% | 86.9%	Method name = [DSOPM2] K=69.41 | 1 | 2 1 | 33.75 | 3.25 2 | 6.75 | 23.25 Se | 91.2% | 77.5% Pp | 83.3% | 87.7%
63	K=76.7 | 1 | 2 | 3 | 4 1 | 52.00 | 0.75 | 0.00 | 1.25 2 | 0.25 | 36.75 | 15.25 | 0.75 3 | 0.50 | 14.25 | 38.75 | 0.50 4 | 1.25 | 1.50 | 0.50 | 45.75 Se | 96.3% | 69.3% | 71.8% | 93.4% Pp | 96.3% | 69.0% | 71.1% | 94.8%	Method name = [LDA] K=65.07 | 1 | 2 | 3 | 4 1 | 51.75 | 1.50 | 0.00 | 0.75 2 | 2.25 | 33.25 | 15.50 | 2.00 3 | 2.25 | 24.25 | 25.00 | 2.50 4 | 0.75 | 0.75 | 2.50 | 45.00 Se | 95.8% | 62.7% | 46.3% | 91.8% Pp | 90.8% | 55.6% | 58.1% | 89.6%

Continued on next page

Table A.1 – *Continued from previous page*

64

K=98.8	1	2	3	4	5	6
1	25.00	0.00	0.00	0.00	0.00	0.00
2	0.00	25.00	0.00	0.00	0.00	0.00
3	0.00	0.00	24.50	0.00	0.50	0.00
4	0.00	0.00	0.00	24.75	0.00	0.25
5	0.00	0.00	0.25	0.00	24.75	0.00
6	0.00	0.00	0.00	0.50	0.00	24.50
Se	100.0%	100.0%	98.0%	99.0%	99.0%	98.0%
Pp	100.0%	100.0%	99.0%	98.0%	98.0%	99.0%

Method name = [CDOPM2]						
K=86.8	1	2	3	4	5	6
1	25.00	0.00	0.00	0.00	0.00	0.00
1	0.00	25.00	0.00	0.00	0.00	0.00
1	0.00	0.00	22.00	0.00	3.00	0.00
1	0.00	0.00	0.00	22.75	0.00	2.25
1	0.00	0.00	5.25	0.00	19.75	0.00
1	0.00	0.00	0.00	6.00	0.00	19.00
Se	100.0%	100.0%	88.0%	91.0%	79.0%	76.0%
Pp	100.0%	100.0%	80.7%	79.1%	86.8%	89.4%

65

K=37.2	1	2	3
1	6.00	1.00	1.00
2	3.50	3.25	1.25
3	1.50	2.25	5.25
Se	75.0%	40.6%	58.3%
Pp	54.5%	50.0%	70.0%

Method name = [KernelPCA]			
K=27.11	1	2	3
1	4.75	0.75	2.50
2	2.50	1.50	4.00
3	1.25	1.00	6.75
Se	59.4%	18.8%	75.0%
Pp	55.9%	46.2%	50.9%

66

K=98.1	1	2
1	81.50	1.50
2	0.50	155.50
Se	98.2%	99.7%
Pp	99.4%	99.0%

Method name = [LDA]		
K=96.25	1	2
1	79.00	4.00
2	0.00	156.00
Se	95.2%	100.0%
Pp	100.0%	97.5%

67

K=0.0	1	2
1	0.75	0.25
2	0.75	0.25
Se	75.0%	25.0%
Pp	50.0%	50.0%

Method name = [CDOLM]		
K=100	1	2
1	1.00	0.00
2	0.00	1.00
Se	100.0%	100.0%
Pp	100.0%	100.0%

68

K=60.8	1	2
1	43.75	8.25
2	5.50	19.50
Se	84.1%	78.0%
Pp	88.8%	70.3%

Method name = [CDOQM]		
K=57.62	1	2
1	42.75	9.25
2	5.75	19.25
Se	82.2%	77.0%
Pp	88.1%	67.5%

69

K=77.7	1	2	3
1	10.50	4.00	0.50
2	3.50	20.50	1.00
3	0.50	1.25	35.25
Se	70.0%	82.0%	95.3%
Pp	72.4%	79.6%	95.9%

Method name = [LDA]			
K=70.79	1	2	3
1	9.50	4.50	1.00
2	2.50	20.25	2.25
3	1.25	2.50	33.25
Se	63.3%	81.0%	89.9%
Pp	71.7%	74.3%	91.1%

Continued on next page

Table A.1 – *Continued from previous page*

70

K=98.2	1	2	3
1	13.75	0.25	0.00
1	0.00	16.75	0.25
1	0.00	0.00	12.00
Se	98.2%	98.5%	100.0%
Pp	100.0%	98.5%	98.0%

Method name = [CDOPM3]			
K=97.37	1	2	3
1	14.00	0.00	0.00
2	0.00	16.25	0.75
3	0.00	0.00	12.00
Se	100.0%	95.6%	100.0%
Pp	100.0%	100.0%	94.1%

71

K=92	1	2	3	4	5	6
1	4.00	0.00	0.00	0.00	0.00	0.00
2	0.00	3.00	0.00	0.00	0.00	0.00
3	0.00	0.25	0.75	0.00	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.00	1.00	0.00
6	0.00	0.00	0.00	0.00	0.50	1.50
Se	100.0%	100.0%	75.0%	100.0%	100.0%	75.0%
Pp	100.0%	92.3%	100.0%	100.0%	66.7%	100.0%

Method name = [PKDA2]						
K=91.99	1	2	3	4	5	6
1	4.00	0.00	0.00	0.00	0.00	0.00
2	0.00	3.00	0.00	0.00	0.00	0.00
3	0.00	0.00	1.00	0.00	0.00	0.00
4	0.00	0.00	0.00	1.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.75	0.25
6	0.00	0.00	0.00	0.00	0.50	1.50
Se	100.0%	100.0%	100.0%	100.0%	75.0%	75.0%
Pp	100.0%	100.0%	100.0%	100.0%	60.0%	85.7%

Table A.1: The confusion matrix in $\mathbb{R}^N$,and in $\mathbb{R}^2$ for whole datasets, that mentioned the best mapping methods, where K is the kappa, Se is the sensitivity measure, and Pp the precision

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

Bibliography

[1] D.K. Agrafiotis. Stochastic proximity embedding. *Journal of Computational Chemistry*, 24(10):1215–1221, 2003.

[2] S. Alawadi, M. Fernández-Delgado, D. Mera, and S. Barro. Polynomial kernel discriminant analysis for 2d visualization of classification problems. *Neural Computing & Applications*, page In press, 2018.

[3] S. Alawadi, D. Mera, M. Fernández-Delgado, and J.A. Taboada. Comparative study of artificial neural network models for forecasting the indoor temperature in smart buildings. In *International Conference on Smart Cities*, pages 29–38. Springer, 2017.

[4] L. Amaro, Z. Heiga, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. On the use of kernel PCA for feature extraction in speech recognition. *IEICE Transactions on Information and Systems*, 87(12):2802–2811, 2004.

[5] S. Balakrishnama and A. Ganapathiraju. Linear discriminant analysis-a brief tutorial. *Institute for Signal and Information Processing*, 18, 1998.

[6] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[7] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[8] Abdullatif E Ben-Nakhi and Mohamed A Mahmoud. Cooling load prediction for buildings using general regression neural networks. *Energy Conversion and Management*, 45(13):2127–2141, 2004.

[9] Y. Bengio, N. Delalleau, N. Le Roux, J.F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 16(10):2197–2219, 2004.

[10] M. Brand. Charting a manifold. In *Advances in Neural Information Processing Systems*, pages 961–968, 2002.

[11] L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

[12] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[13] L. Breiman, J. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees*. Wadsworth and Brooks, 1984.

[14] A. Buja, D.F. Swayne, M.L. Littman, N. Dean, H. Hofmann, and L. Chen. Data visualization with multidimensional scaling. *Journal of Computational and Graphical Statistics*, 17(2):444–472, 2008.

[15] M.A. Carreira-Perpignan. A review of dimension reduction techniques. Technical report, Dept. of Computer Science, Univ. Sheffield, 1997.

[16] Tiberiu Catalina, Joseph Virgone, and Eric Blanco. Development and validation of regression models to predict monthly heating demand for residential buildings. *Energy and buildings*, 40(10):1825–1832, 2008.

[17] J.M. Chambers. *Linear models*, chapter 4. J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole, 1992.

[18] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[19] H. Chun and S. Keles. Sparse partial least squares for simultaneous dimension reduction and variable selection. *Journal of the Royal Statistical Society*, 72:3–25, 2010.

[20] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.

[21] ATLAS Collaboration et al. The evolution of boosting algorithms-from machine learning to statistical modelling. *Methods Inf Med*, 53(6):419–427, 2014.

[22] Europese Commissie. A roadmap for moving to a competitive low carbon economy in 2050. *Europese Commissie, Brussel*, 2011.

[23] J. Cook, I. Sutskever, A. Mnih, and G.E. Hinton. Visualizing similarity data with a mixture of maps. In *International Conference on Artificial Intelligence and Statistics*, volume 7, pages 67–74, 2007.

[24] J.P. Cunningham and Z. Ghahramani. Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Reseach*, 16:2859–2900, 2015.

[25] J. de la Porte, B.M. Herbst, W. Hereman, and S.J. van der Walt. An introduction to diffusion maps. In *Symposium of the Pattern Recognition Association of South Africa*, pages 15–25, 2008.

[26] V. de Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford University, 2004.

[27] D. DeMers and G.W. Cottrell. n–linear dimensionality reduction. *Advances in Neural Information Processing Systems*, 5:580–587, 1993.

[28] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[29] A.J. Dobson. *An introduction to generalized linear models*. Chapman and Hall, 1990.

[30] Bing Dong, Cheng Cao, and Siew Eang Lee. Applying support vector machines to predict building energy consumption in tropical region. *Energy and Buildings*, 37(5):545–553, 2005.

[31] D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10):5591–5596, 2003.

[32] Haris Doukas, Konstantinos D Patlitzianas, Konstantinos Iatropoulos, and John Psarras. Intelligent building energy management system using rule sets. *Building and environment*, 42(10):3562–3569, 2007.

[33] Anastasios I Dounis and Christos Caraiscos. Advanced control systems engineering for energy and comfort management in a building environment—a review. *Renewable and Sustainable Energy Reviews*, 13(6):1246–1261, 2009.

[34] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification, 2nd Ed.* Wiley-Interscience, 2001.

[35] Betul Bektas Ekici and U Teoman Aksoy. Prediction of building energy consumption by using artificial neural networks. *Advances in Engineering Software*, 40(5):356–362, 2009.

[36] D. Engel, L. Hüttenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *OASIcs-OpenAccess Series in Informatics*, volume 27. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[37] Varick L Erickson, Miguel Á Carreira-Perpiñán, and Alberto E Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269. IEEE, 2011.

[38] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[39] F.D. Foresee and M. T. Hagan. Gauss-Newton approximation to Bayesian regularization. In *International Joint Conference on Neural Networks*, pages 1930–1935, 1997.

[40] J.H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–141, 1991.

[41] J.H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[42] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76:817–823, 1981.

[43] S. García, A. Fernández, A.D. Benítez, and F. Herrera. Statistical comparisons by means of non-parametric tests: A case study on genetic based machine learning. In

Proceedings of the II Congreso Español de Informática (CEDI 2007). V Taller Nacional de Minería de Datos y Aprendizaje (TAMIDA), pages 95–104, 2007.

[44] A. Gelman, A. Jakulin, M.G. Pittau, and Y.S. Su. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, 2(4):1360–1383, 2009.

[45] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[46] A. Globerson and S.T. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, pages 451–458, 2005.

[47] J.J. Goeman. L-1 penalized estimation in the cox proportional hazards model. *Biometrical Journal*, 52:70–84, 2010.

[48] J. Goldberger, S. Roweis, G.E. Hinton, and R. Salakhutdinov. Neighborhood component analysis. In *Advances in Neural Information Processing Systems*, 2004.

[49] Q. Gu, Z. Li, and J. Han. Linear discriminant dimensionality reduction. In *Machine Learning and Knowledge Discovery in Databases*, pages 549–564. Springer, 2011.

[50] X. He, D. Cai, S. Yan, and H.J. Zhang. Neighborhood preserving embedding. In *IEEE International Conference on Computer Vision*, volume 2, pages 1208–1213. IEEE, 2005.

[51] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems*, volume 16, page 153. MIT, 2003.

[52] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[53] G.E. Hinton and S.T. Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems*, pages 833–840, 2002.

[54] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[55] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417, 1933.

[56] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Systs., Man, and Cybern.-Part B: Cybern.*, 42(2):513–529, 2012.

[57] A. Hyvarinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13:411–430, 2000.

[58] W. Jianzhong. *Geometric structure of high-dimensional data and dimensionality reduction*, chapter Maximum Variance Unfolding, pages 181–202. Springer Berlin Heidelberg, 2011.

[59] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[60] H.F. Kaiser. The application of electronic computers to factor analysis. *Educational and Psychological Measurement*, 1960.

[61] A. Kapelner and J. Bleich. bartMachine: machine learning with Bayesian additive regression trees. *Journal of Statistical Software*, 70(4):1–40, 2016.

[62] K.I. Kim, K. Jung, and H.J. Kim. Face recognition using kernel principal component analysis. *IEEE Signal Processing Letters*, 9(2):40–42, 2002.

[63] K.I. Kim, S.H. Park, and H.J. Kim. Kernel principal component analysis for texture classification. *IEEE Signal Processing Letters*, 8(2):39–41, 2001.

[64] Simon SK Kwok, Richard KK Yuen, and Eric WM Lee. An intelligent approach to assessing the effect of building occupancy on building cooling load prediction. *Building and Environment*, 46(8):1681–1690, 2011.

[65] N.D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in Neural Information Processing Systems*, 16(3):329–336, 2004.

[66] C.L. Lawson and R.J. Hanson. *Solving least squares problems*, volume 15 of *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), 1995.

[67] S. Lespinats, M. Verleysen, A. Giron, and B. Fertil. DD-HDS: A method for visualization and exploration of high-dimensional data. *IEEE Transactions on Neural Networks*, 18(5):1265–1279, 2007.

[68] Kangji Li, Hongye Su, and Jian Chu. Forecasting building energy consumption using neural networks and hybrid neuro-fuzzy system: A comparative study. *Energy and Buildings*, 43(10):2893–2899, 2011.

[69] Qiong Li, Qinglin Meng, Jiejin Cai, Hiroshi Yoshino, and Akashi Mochida. Applying support vector machine to predict hourly cooling load in the building. *Applied Energy*, 86(10):2249–2256, 2009.

[70] http://www.life-opere.org/en, (Retrieved January 2017).

[71] L. Maaten, E. Postma, and H. Herik. Dimensionality reduction: A comparative review. Technical report, Tilburg University, 2009.

[72] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[73] Kelly O Maloney, Matthias Schmid, and Donald E Weller. Applying additive modelling and gradient boosting to assess the effects of watershed and reach characteristics on riverine assemblages. *Methods in Ecology and Evolution*, 3(1):116–128, 2012.

[74] A.E. Maxwell. Recent trends in factor analysis. *Journal of the Royal Statistical Society. Series A (General)*, pages 49–59, 1961.

[75] N Meinshausen. Quantregforest: quantile regression forests. *R package version 0.2-2*, 2007.

[76] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. In *IEEE Signal Processing Society Workshop*, volume 9, pages 41–48, 1999.

[77] S. Mika, G. Ratsch, J. Weston, B. Schölkopf, and K. R. Mullers. Fisher discriminant analysis with kernels. In *Proc. IEEE Workshop in Neural Netw. for Signal Proc.*, pages 41–48, 1999.

[78] S. Mika, B. Schölkopf, A.J. Smola, K.R. Müller, M. Scholz, and G. Rätsch. Kernel pca and de-noising in feature spaces. In *Advances in Neural Information Processing Systems*, volume 4, page 7. Citeseer, 1998.

[79] I. Mizera and R. Koenker. Convex optimization in r. *Journal of Statistical Software*, 60(5):1–23, 2014.

[80] T.K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

[81] Petru-Daniel Moroşan, Romain Bourdais, Didier Dumur, and Jean Buisson. Building temperature regulation using a distributed model predictive control. *Energy and Buildings*, 42(9):1445–1452, 2010.

[82] L.S. Mthembu and J. Greene. A comparison of three class separability measures. In *Symposium of the Pattern Recognition Association of South Africa*, pages 63–67, 2004.

[83] K. Nam, H. Je, and S. Choi. Fast stochastic neightbor empedding: a trust region algorithm. In *Proc. IEEE International Joint Conference on Neural Networks*, pages 123–128, 2004.

[84] Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and buildings*, 56:244–257, 2013.

[85] Thomas Olofsson and Staffan Andersson. Long-term energy demand predictions based on short-term measured data. *Energy and Buildings*, 33(2):85–91, 2001.

[86] J. Platt. Fastmap, metricmap, and landmark mds are all nystrom algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2005.

[87] T. Pohlert. *The pairwise multiple comparison of mean ranks package (PMCMR)*, 2014. R package.

[88] R. Quinlan. Combining instance-based and model-based learning. In *Proc. Intl. Conf. on Machine Learning*, pages 236–243, 1993.

[89] R.J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, 1992.

[90] GJ Ríos-Moreno, M Trejo-Perea, R Castaneda-Miranda, VM Hernández-Guzmán, and G Herrera-Ruiz. Modelling temperature in intelligent buildings by means of autoregressive models. *Automation in Construction*, 16(5):713–722, 2007.

[91] Pablo Rodrıguez-Mier, Marc Fresquet, Manuel Mucientes, and Alberto Bugarın. Prediction of indoor temperatures for energy optimization in buildings. In *Conference of the Spanish Association for Artificial Intelligence*, pages 675–684, 2016.

[92] R. Rosipal, M. Girolami, L.J. Trejo, and A. Cichocki. Kernel pca for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10(3):231–243, 2001.

[93] S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[94] S. De Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and intelligent laboratory systems*, 18:251–263, 1993.

[95] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18(5):401–409, 1969.

[96] Olivia Guerra Santin, Laure Itard, and Henk Visscher. The effect of occupancy and building characteristics on energy use for space and water heating in dutch residential stock. *Energy and buildings*, 41(11):1223–1232, 2009.

[97] Matthias Schmid and Torsten Hothorn. Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2):298–311, 2008.

[98] B. Schölkopf, A. Smola K.R., and Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997.

[99] F. Sha and L.K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *International Conference on Machine Learning*, pages 784–791, 2005.

[100] D.J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2006.

[101] V.D. Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, pages 705–712, 2002.

[102] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for Cox's proportional hazards model via coordinate descent. *Journal of statistical software*, 39(5):1–13, 2011.

[103] A.J. Smola and Bernhard B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[104] L. Song, P. Langfelder, and S. Horvath. Random generalized linear model: a highly accurate and interpretable ensemble predictor. *BMC Bioinformatics*, 14(1):1–22, 2013.

[105] C. Spearman. General intelligence objectively determined and measured. *Americal Journal of Psichology*, 15:206–221, 1904.

[106] D.F. Specht. A general regression neural network. *IEEE Trans. on Neural Networks*, 2:568–576, 1991.

[107] Kevin Swingler. *Applying neural networks: a practical guide*. Morgan Kaufmann, 1996.

[108] Yee W Teh and Sam T Roweis. Automatic alignment of local representations. In *Advances in Neural Information Processing Systems*, pages 841–848, 2002.

[109] J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[110] C. Thornton. Separability is a learner's best friend. In *Neural Computation and Psychology Workshop*, pages 40–46. Springer, 1998.

[111] L.L. Thurstone. *Multiple factor analysis.* Univ. Chicago Press, Chicago (USA), 1947.

[112] M.E. Tipping and C.M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[113] W.S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[114] L. van der Maaten. An introduction to dimensionality reduction using Matlab. *Report*, 1201(07-07):62, 2007.

[115] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[116] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

[117] J. Verbeek. Learning nonlinear image manifolds by global alignment of local linear models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1236–1250, 2006.

[118] K.Q. Weinberger and L.K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *National Conference on Artificial Intelligence*, volume 6, pages 1683–1686, 2006.

[119] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[120] K.Q. Weinberger and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Reseach*, 10:207–244, 2009.

[121] K.Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems*, pages 1489–1496, 2007.

[122] C. K. I. Williams and D. Barber. Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.

[123] S.N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society*, 1(73):3–36, 2011.

[124] X. Xing, S. Du, and K. Wang. Robust hessian locally linear embedding techniques for high-dimensional data. *Algorithms*, 9(2):36, 2016.

[125] Z. Yang, I. King, Z. Xu, and E. Oja. Heavy-tailed symmetric stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 2169–2177, 2009.

[126] J. Ye, R. Janardan, and Q. Li. Two-dimensional linear discriminant analysis. In *Advances in Neural Information Processing Systems*, pages 1569–1576, 2004.

[127] Jieping Ye. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6(Apr):483–502, 2005.

[128] L. Yu, R.R. Snapp, T. Ruiz, and M. Radermacher. Probabilistic principal component analysis with expectation maximization (ppca-em) facilitates volume classification and estimates the missing data. *Journal of Structural Biology*, 171(1):18–30, 2010.

[129] F Zamora-Martínez, P Romeu, P Botella-Rocamora, and J Pardo. On-line learning of indoor temperature forecasting models towards energy efficiency. *Energy and Buildings*, 83:162–172, 2014.

[130] T. Zhang. Adaptive forward-backward greedy algorithm for learning sparse representations. *IEEE Trans. Inf. Theor.*, 57(7):4689–4708, 2011.

[131] T. Zhang, J. Yang, D. Zhao, and X. Ge. Linear local tangent space alignment and application to face recognition. *Neurocomputing*, 70(7):1547–1553, 2007.

[132] Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via local tangent space alignment. *SIAM Journal on Scientific Computation*, 26(1):313–338, 2004.

[133] Hai-xiang Zhao and Frédéric Magoulès. A review on the prediction of building energy consumption. *Renewable and Sustainable Energy Reviews*, 16(6):3586–3592, 2012.

List of Figures

USC
UNIVERSIDADE
DE SANTIAGO
DE COMPOSTELA

List of Tables