Dimensionality Reduction for Studying Diffuse Circumgalactic Medium

Name: Jakub Rybak

University: Imperial College London

Email: jakub.rybak18@imperial.ac.uk

Location: London, UK

Summary

In the era of big data, datasets in a number of areas exhibit high dimensionality, that is a large number of explanatory variables are available. Such situations offer a lot of promise, as unless there is a high number of redundant features, these datasets are endowed with a lot of information potentially relevant for the response.

High-dimensionality also poses significant challenges. Firstly, due to the curse of dimensionality, performance and theoretical guarantees of many standard methods deteriorate. Secondly, it is much harder to interpret or visualise such data. Reducing the dimensionality of the set of features has the potential to alleviate all of these problems, as well as aid with data compression and speed-up certain algorithms.

This project will seek to identify dimensionality-reduction methods that achieve a reduction in the number of features while maintaining predictive performance of the relevant models. This will be done in the context of astrophysical data, however, the problem of many features is common in many areas, including genomics, economics and imaging. Thus, the applicability of our analysis will extend beyond the realm of physics.

Benefits to community

The project will have two main outcomes, which, given the omnipresence of this problem in many areas, will benefit a wide range of practitioners.

1. Creation of a Python functionality for identifying the most suitable dimensionality reduction tool for a given predictive method and dataset, which will provide a single point of access to dimensionality-reduction methods currently spread across multiple packages.

2. Provision of a clear overview of some of the main dimensionality-reduction methods, along with case studies based on publicly available datasets, making these tools more accessible to the general audience.

Time permitting, we will aim to implement methods of dimensionality reduction that are currently lacking in the Python machine learning libraries.

Introduction and motivation

The presence of many features creates a difficulty in applications of statistical and machine learning techniques in many areas. In such situations, data visualisation and interpretation of results become difficult, the speed of algorithms whose time complexity scales with input dimensionality deteriorates, and performance of many standard models can rapidly fall. Dimensionality-reduction, by which we mean a reduction in the number of features from, say, to , , is a powerful tool to alleviate these problems, as well as to aid with data compression.

This project will approach dimensionality-reduction with an objective of maximising model performance. Specifically, we will explore methods that reduce the number of features as much as possible, while maintaining (or enhancing) the original predictive performance of the model.

The analysis will be done on the simulated dataset of quasar absorption spectra, which arises in the study of the circumgalactic medium in astrophysics. However, such problems arise across many areas, for example in genomics (Bickel et al. 2009), in the analysis of imaging data, for example in ecology (Landgrebe, 2008), or in economics and finance (Fan and Li, 2006), and hence the results of this project will be widely applicable.

While most practitioners are familiar with unsupervised dimensionality-reduction methods, such as principal component analysis, methods which take into account a target variable enjoy much less popularity, despite their great promise in prediction tasks. Moreover, the implementation of these methods is split across multiple packages, often with very limited documentation. Thus, there is often a lack of clear an easy-to-access explanations of these methods as well as examples illustrating their usage.

This project will seek to fill this gap, as well as to provide a Python functionality that allows for automatic discrimination between various dimensionality-reduction methods for any given predictive model. Thus, our objective will be twofold.

Firstly, we will seek to provide a clear overview and case studies of selected supervised dimensionality-reduction methods. The aim will be to create a common entry point, explaining different methods and connections between them, on both an introductory and more advanced level, and illustrate their applications based on real datasets. This will make these tools more accessible to non-academic audiences.

Secondly, we will develop a Python functionality (module) that will allow users to automatically select the most appropriate supervised dimensionality-reduction tool which maintains the performance of a chosen predictive model. This will provide an easy access to methods implemented across different packages, as well as a way to discriminate between them.

Challenges (and promises) of dimensionality reduction

There are two main settings to be distinguished here, as each of them poses its own challenges. Firstly, if the sample size is much larger than the number of features, we are able to estimate complex models. Successful dimensionality reduction needs to be able to capture these patterns in the data in order to avoid performance deterioration.

On the other hand, if the sample size and the number of features are of similar orders of magnitudes, the curse of dimensionality leads to performance deterioration of many standard models and dimensionality reduction needs to be able to (and has a potential to) alleviate these issues. Below we will discuss these two settings in more detail.

The curse of dimensionality

It is widely documented that performance of many standard prediction models deteriorates when the number of features gets large (for example see Zou et al. (2005 for the case of linear model, Zhu et al. (2004) for SVMs, or Hastie et al. (2009) for a more general discussion). This is caused by the fact that high-dimensionality impacts many traditional statistical quantities on which both statistical models are based. For example, commonly used methods, such as linear regression, linear discriminant analysis and others, rely on an estimate of the covariance matrix. Estimating the covariance matrix involves estimating parameters for a set of features. Clearly, the number of parameters is of a squared order, and hence even a moderate number of features can invalidate such procedures.

Dimensionality-reduction can alleviate this problem, as by its nature it reduces the number of features in the dataset. However, the reduction methods also rely on the same statistical quantities as the models. For example, principal component analysis is based on a decomposition of the sample covariance matrix.

This illustrates that the curse of dimensionality is essentially moved from model-estimation to model pre-processing, but is still present (see e.g. Johnstone and Lu, 2009). Thus, care must be taken if we hope to obtain competitive model performance.

A common solution to this problem is to either amend the estimators of the underlying statistical quantities, for example by using a shrinkage estimator of the covariance matrix in linear discriminant analysis, or by regularisation.

Complex patterns

It is general practice in prediction problems to seek feature transformations, either based on visual analysis of data, or based on model estimation. For example, it is often unlikely that a feature has a linear impact on the response, and hence a better model can be obtained by including a polynomial term of the feature.

Presence of many features complicates this step, as data visualisation becomes more challenging, and subject-matter knowledge is unlikely to extend to hundreds of different features. With a large sample size, this can be circumvented by estimating flexible models, such as non-parametric and deep models.

Hence, in order to maintain the predictive performance, dimensionality-reduction techniques would ideally be able to incorporate the information from these extended feature sets.

This can be done either by the kernel trick, resulting in methods such as kernel linear discriminant analysis and kernel canonical correlation analysis, or by incorporating non-parametric methods into the classical techniques, such as in the case of flexible linear discriminant analysis.

Tools for dimensionality reduction

Dimensionality reduction seeks to find a smaller features space, consisting of, say features. This can be achieved in three ways; either by selecting a subset of the original features that are most important for the response, by projecting the original data onto a lower () dimensional subspace, or the combination of the two.

Projections

To reduce the number of features of the dataset, we can seek a new feature set, derived from the original one, which will capture the characteristics of the original data that we deem important.

Mathematically, this corresponds to projecting the -dimensional data onto a -dimensional subspace (), in a way that retains as much (of the important) information as possible.

For example, PCA seeks to find linear combinations of existing variables which successively maximise their variance, while being uncorrelated with each other. If we can find variables which capture most of the variability in the original dataset, then we can work with features instead of the original -features.

Linear Discriminant Analysis (LDA), on the other hand, seeks directions under which the data from different classes are well separated. Other supervised methods include, for example, partial least squares and canonical correlation analysis.

An important distinction is between the supervised and unsupervised settings, that is situations where a response variable is either present or not. While a reduction in unsupervised settings reduces the dimension of the input space, there is no guarantee that this reduction will preserve the specific information contained in the features that is relevant for predicting the target variable (for example, in the case of PCA, it may be the case that directions of maximum variance in the feature set are not related to the target variable). On the other hand, by considering the response in the reduction procedure, supervised methods seek to retain the information relevant for predicting it as much as possible.

Identifying a subset of features

An alternative to projecting data into a lower-dimensional space is to identify unnecessary (noise) features and eliminate them from the dataset. A wide range of models has been proposed for this problem, most famously Lasso (Tibshirani, 1996), which has enjoyed great popularity in applied statistics and a wide range of theoretical guarantees. This class of estimators can be encompassed by a family of regularized M-estimators (Negahban et al., 2012). The key component of this class of models is regularisation, which penalises large or non-zero coefficients, thus encouraging a large number of weights to be estimated as zero.

This method essentially assumes sparsity of the underlying model, that is, we believe that many of the features have no influence on the response, or that the true model is well-approximated by a model of this form. If the true model is indeed

sparse, these techniques can help us uncover the true set of active features, aiding interpretation.

For example, in the context of quasar absorption spectra data, the light intensity at subsequent wavelengths may be strongly correlated. As a result, given a signal at wavelength , signal at wavelength may not add that much additional information. Thus, the presence of features corresponding to the wavelength may not add much to model performance.

Existing work

This section will review methods of dimensionality reduction and their implementations in Python that will be explored further in this project, and discuss how this project builds on an extends the resources already available.

Unsupervised projection methods

In practice, dimensionality reduction for predictions problems has often been done using unsupervised methods, such as principal component analysis, independent component analysis, and their extensions. Although for problems involving a target variable, supervised dimensionality-reduction methods might be expected to perform better, this is not always the case (see e.g. Weinberger and Saul (2009)). Given this and the wide-spread use of unsupervised methods we will include them in our analysis. Note that PCA, kernel PCA, as well as sparse PCA, which is designed to handle high-dimensional data, are implemented in the Python Scikit-learn package (Pedregosa et al., 2011), facilitating such analysis. An extension of unsupervised non-linear dimensionality reduction methods is manifold learning, an approach that seeks to find low-dimensional manifolds embedded in high-dimensional spaces. This includes for example the Isomap method (Tenenbaum et al., 2000) and local linear embeddings (Roweis and Saul, 2000). These are implemented in the manifold module of Scikit-learn (Pedregosa et al., 2011).

Supervised projection methods

Two main types of supervised dimensionality-reduction techniques for classification problems will be discussed here. The first class of methods is based on discriminant functions (Hastie et al., 2009), and includes methods such as linear and quadratic discriminant analyses. The second-class of methods are distance-based methods, of which -nearest neighbours (kNN) is the most prominent example.

Discriminant methods

The basic supervised dimensionality-reduction techniques, such as LDA, QDA and PLS, have been around for a long time, and provide a natural starting point for our empirical investigation. These are implemented in the Python Scikit-learn package (Pedregosa et al., 2011).

LDA (Fisher, 1936) seeks a projection to a sub-space in which all data points from the same class are close together while class centroids are as far apart as possible (maximising the between-class variance relative to the within-class variance). LDA assumes Gaussian distributions within each class, with equal covariance matrices across classes. Relaxing the assumption of common class covariance matrices leads to the quadratic discriminant analysis. LDA and QDA are implemented in the Scikit-learn package (Pedregosa et al., 2011). Importantly, LDA methods allow the user to use the shrinkage estimator of the covariance matrix, which is suitable for high-dimensional problems.

For classification problems, we seek a -dimensional subspace that allows the separation of classes as much as possible. LDA is a linear dimensionality-reduction technique and hence seeks such subspace in relation to a separating hyperplane. This technique is thus unable to capture non-linearities in the feature space. An extension is to enlarge the set of features to allow non-linearities to be captured. Such approach gains flexibility at the cost of increasing dimensionality of the problem, exacerbating the curse of dimensionality and the risk of overfitting. To counteract this, regularisation methods need to be used. Therefore, a natural extension is to use non-parametric regression methods which generate an extended feature space through basis functions and then fit a regularised model. In the context of LDA, this approach leads to the flexible discriminant analysis model (Hastie et al., 1994) and its generalisation, the penalised discriminant analysis (Hastie et al., 1995). Clearly, these methods provide a powerful extension of the LDA procedure. Another approach is to use the kernel trick, leading to a kernel LDA (Mika, 1999), which is implemented in the pyDML package (Suarez et al., 2020) and the regularised kernel canonical correlation analysis, implemented in the Pyrcca package (Bilenko and Gallant, 2016).

Another drawback of LDA is the assumption of a Gaussian distribution within each class, which, among other things, makes it unsuitable for multimodal data. Also, the dimensionality of the sub-space (that is, the number of features to which we are able to reduce our dataset), is limited to at most , where is number of classes (see e.g. Fukunaga, 1990 for more details). In the case of multiple modes, an alternative is to seek a sub-space in which data points within a single class that are close together in the original features space are still close, data points from different classes are separated, but data points from the same class that are far apart from each other in the original space are not imposed to be close together in the transformed space. This is an approach taken by the Local LDA (Sugiyama, 2007). Again, a non-linear

feature space can be considered via the kernel trick. Both LLDA and kernel LLDA were recently implemented in Python by Suarez et al. (2020).

While a large number of these methods have been implemented in Python, the documentation often lacks a clear explanation of the methods and illustrations of their usage. For example, the pyDML package simply refers to the original academic articles, which are not easily accessible to the general audience. Filling this gap could help practitioners in applying cutting-edge tools as well as raising awareness about them.

Metrics-based methods

A very simple, yet often effective method for classification is a -nearest neighbours classification, which seeks to classify each data point based on the classes of its neighbours. This method is non-parametric, giving it a great flexibility to adopt to data. However, kNN has two challenges. Firstly, it is not obvious which metric should be used to measure distances between points. While kNN traditionally uses a Euclidean distance, different metrics may be more suitable for different types of datasets. This has led to the development of metric-learning techniques (see Bellet et al. (2013) for an overview). Secondly, with many features the sample space is sparsely populated, which often leads to performance deterioration in kNN. An alternative is to learn a suitable Mahalanobis distance, for example using neighbourhood component analysis (Goldberger et al. 2004). This approach corresponds to learning a linear transformation of the data, such that the kNN performs well in the transformed space. By learning a transformation onto a smaller space, which corresponds to learning a low-rank metric, we can reduce the dimensionality of the dataset (Goldberger et al. 2004). Large margin nearest neighbour classification (LMNN) of Weinberger and Saul (2009) follows a similar approach. LMNN seeks to learn a linear transformation in the input space, such that neighbourhoods contain samples with the same label, by penalising large distances between nearby inputs with the same labels and small distances between inputs from different classes. Similarly to support vector machines, the method seeks a margin between neighbourhoods, and hence around kNN decision boundaries. Again, learning a lower-rank transformation results in dimensionality reduction. LMNN and its extension, kernel LMNN are implemented in pyDML package (Suarez et al., 2020).

Of course, the success of any dimensionality reduction will depend on the prediction methods used. For example, metric-based dimensionality reduction methods will be more suitable for metric-based prediction models, such as kNN, while linear dimensionality reduction methods will be more appropriate for methods seeking linear decision boundaries. For this reason, it is important to start with a wide range

of dimensionality-reduction methods and use validation to discriminate between them.

Variable selection

The above-mentioned methods have all projected data onto a lower-dimensional subspace. Another alternative is to perform a feature selection step, either by a way of regularisation or using greedy approaches, such as forward or backward elimination.

Regularisation can be achieved by penalising coefficients by either L1 penalty, resulting in Lasso (Tibshirani, 1996), or a combination of L1 and L2 penalties, resulting in an elastic net (Zou and Hastie, 2005). On the other hand, greedy approaches aim to successively eliminate (or add) the feature least (or most) relevant for the target variable. The disadvantage of greedy approaches relative to regularisation is that, firstly, they are not robust to the curse of dimensionality, as for large feature sets the estimated model will be invalid, invalidating in turn any elimination based on the estimated coefficients. Secondly, even for small feature sets, greedy approaches lack the desirable theoretical properties, such as convergence to an optimum point.

A summary of the methods described here can be found in Table 1. As has been mentioned, most of these methods have been implemented in Python (some of them only very recently), which makes their exploration feasible in the time given.

Two important methods have not been implemented in Python yet, namely flexible and penalised discriminant analysis. Adding both of these methods would provide a valuable extension of Python's machine learning toolkit.

Table 1: Dimensionality-reduction methods in Python

Method	Implementation	Documentation & example	Property
Unsupervised Methods			
Principal Component Analysis	Scikit-learn	Yes	-
Sparse PCA	Scikit-learn	Yes	High-dim
Kernel PCA	Scikit-learn	Yes	Flexible
Independent Component Analysis	Scikit-learn	Yes	-
Manifold learning			
Isomap	Scikit-learn	Yes	-
Local linear embeddings	Scikit-learn	Yes	-
Supervised Methods			
Discriminant methods			
Linear Discriminant Analysis	Scikit-learn	Yes	-
LDA with robust covar. matrix	Scikit-learn	Yes	High-dim
Kernel LDA	pyDML	Yes	Flexible
Regularised kernel CCA	Pyrcca	Yes	Flexible & high-dim
Flexible and penalised DA	Not implemented	N/A	Flexible
Local LDA	pyDML	N/A	-
Kernel Local LDA	pyDML	N/A	Flexible
Metric-based methods			
Neighborhood Component Analysis	pyDML	N/A	-
Large margin nearest neighbor	pyDML	N/A	-
Kernel LMNN	pyDML	N/A	Flexible
Variable selection methods			
Lasso (L1 penalty)	Scikit-learn	Yes	High-dim
Elastic net (L1 + L2 penalty)	Scikit-learn	Yes	High-dim

Column "Property" indicates methods that are designed specifically to either capture complex patterns ("flexible") or to be robust to the curs eof dimensionality ("high-dim").
Scikit-learn package is due to (Pedregosa et al., 2011), pyDML package was introduced in Suarez et al. (2020) and Pyrcca package by (Bilenko and Gallant, 2016).

Deliverables

This is a high-level overview of the main deliverables. More details on how these will be achieved can be found in “Implementation” section and break-down into even smaller tasks with specific deadlines can be found in the “Timeline” section.

1. Analysis of dimensionality-reduction techniques in the context of astrophysical data.

2. Open-source tool allowing automatic discrimination between various dimensionality-reduction approaches.

3. Series of blog articles with documentation and case studies for methods reviewed above, namely:
   - LDA (+ kernel LDA)

- Canonical correlation analysis (+kernel version)
- Local LDA (+ kernel version)
- Neighbourhood component analysis
- Large margin nearest neighbour (+ kernel version)

Implementation

The project will have two objectives, that will be pursued in parallel:

1. Provide a framework for selecting dimensionality-reduction methods in prediction problems.

2. Document tools for supervised dimensionality reduction and provide case studies for each of them.

Below is an overview of these objectives, also see the timeline for a breakdown into smaller tasks.

Framework for dimensionality-reduction in supervised problems

This problem will be explored using quasar absorption spectra data. In later stages, this analysis will be the foundation for the creation of a general tool which will allow discriminating between the different reduction methods for a wide class of prediction models.

1. The astrophysical context

- Explore the connection between prediction models and reduction methods. Perform an empirical investigation of which methods work well with which prediction models.
- Start with selecting a few prediction models without dimensionality reduction to obtain a performance benchmark. Explore dimensionality-reduction techniques described above in the context of these.
- Each dimensionality-reduction technique needs to be tuned on its own (e.g. selecting the number of features it returns). The resulting reduced feature sets can be then combined. A further dimensionality reduction can be achieved by considering feature importance.
- Analyse the merits of the multi-stage approach, for example firstly performing variable selection and then further dimensionality reduction (or vice-versa).
- Using the results of multiple dimensionality-reduction techniques has the potential to improve prediction performance, as each method picks up different aspects of data. However, it may complicate interpretation. Explore

how much performance we sacrifice using a single or just a few reduction methods, and which ones are worth using in conjunction.

2. **Generalised framework**

- Build on the analysis from the previous part to generalise these concepts.
- For a user-defined prediction method and a selected list of reduction methods, identify a reduction method that achieves the desired dimensionality reduction at a minimal performance cost.
  Creating a generalised framework will be based on the fact that model and dimensionality-reduction objects in Python often belong to the same or similar classes and share many attributes. This should simplify the approach.
- See Table 2 for an overview of how this method could work. However, working through the astrophysical data this is likely to be further improved or revised.

Documentation and case studies

Supervised dimensionality reduction methods are less well-known than unsupervised methods. We will seek to fill this gap by providing an overview of supervised methods mentioned in “Existing work” section, with focus on methods for which documentation is not available in the corresponding Python package.

For each method we will include the following:

- An explanation of how dimensionality reduction is achieved intuitively (without mathematical details), to make the material more accessible.
- A visual illustration (using simulated data or suitable illustrations from academic articles).
- A mathematical formulation, seeking a balance between clarity, brevity and completeness.
- A case study using a publicly available dataset. This will explain how to use the corresponding Python function, interpret the output etc.
- A mention of whether extensions are available/possible.
- A list of references for users interested in more details (and, of course, to avoid plagiarism).

Table 2: Suggested form of the generalised procedure

Inputs:

- Prediction method (object)
- Train and validation sets (dataframes)
- Dim-red. methods to consider (dictionary)
- Single method (boolean): True means that we are seeking a single dimensionality-reduction method from the given list, rather than a combination of them.
- Variable selection (Boolean): indicating whether variable selection (using regularised model) should be done as part of analysis.

Process:

1. Auto-tune dimensionality-reduction methods individually. Produce the reduced feature sets for each method.

2. If single_method == True:
    - If variable_selection == True: reduce the dimension (for each reduction method separately) further using Lasso, else pass
    - Use validation to choose between different reduction methods/feature sets.

3. Else:
    - Combine the reduced feature sets to one dataframe.
    - If variable_selection == True: reduce this set further using penalised regression, else pass.
    - Fit the prediction model on the new set of features.

4. Use feature importance to reduce the number of features further.

Output:

- The most realistic option is to output a set of performances for different dimensionality of the feature set (this will be obtained from the last step, where feature importance is used to reduce the number of features), as reduction in the dimensionality will come at a performance cost. Hence the user should be the one deciding on the trade-off she/he is willing to take.

Timeline

Community bonding period:

- Preliminary investigation. Explore mathematical underpinnings, merits and connections between the methods to be studied. While I have certain grasp of these topics, writing about them clearly and concisely will be easier after refreshing and exploring the material in more depth.
- Getting to know the organization's processes on how contributions are to be made.

Weeks 1 - 2:

- Data processing.
- Fit and evaluate prediction methods to obtain performance benchmarks.
- Incorporate dimensionality reduction into prediction models as a pre-processing step. Focus on the dimensionality tools in Scikit-learn.
- Create an overview introducing LDA, both from the standard perspective as well as Fisher's perspectives, and CCA, as well as their kernel-based extensions.

Weeks 3 - 4:

- Explore Pyrcca and pyDML packages. What are the common attributes with Scikit-learn and which aspects differ.
- Incorporate these methods into the predictive frameworks.
- Discriminate between different dimensionality-reduction tools, for different types of prediction models.
- Provide an overview of the Local LDA method (+ kernel extension), along with case studies, and regularised kernel CCA method.

Week 5:

- Bring the work together in the context of astrophysical data. Asses which dimensionality-reduction methods are most suitable (this will likely differ for different predictive models).

  This will be done with a view toward generalisation of this procedure, which will be pursued in the upcoming weeks.

Weeks 6-7:

- Dimensionality-reduction functionality: start creating generalised tool, which, for a given predictive method, select appropriate dimensionality-reduction tools.
- Create an auto-tuning functionality (later to be used in the generalised tool). We will need to explore how the dimensionality-reduction procedures can be auto-tuned. For most reduction methods, we need to select number of components. For some methods, such as PCA, this is often done using rules of thumb (e.g. looking for an inflection point in a plot), whereas for other

methods, for example kernel-based methods, cross validation is often used (which is computationally expensive if done for many methods or many different component numbers).
- Provide documentation and case studies for metric-based methods.

Weeks 8-9:

- Incorporate the auto-tuning functionality into a more general tool.
- Allow a user to choose whether variable selection by regularised regression should be used.
- Create a tool which, for a wide class of predictive methods identifies the least important features (for example, for random forests this can be done by reduction in impurity due to individual features). This would allow another layer of dimensionality reduction.
- Combine variable selection, autotuning of reduction methods, and reduction by feature importance into one tool.
- Write the documentation, including methodological details.

Week 10:

- Test and remove any remaining bugs.
- Finish the documentation.

Please note that I am planning to take two extended weekends off during summer, one at the beginning of July and another one in late July or early April.

After GSoC

The topic of this project and the methods explored are important for researchers across many areas. For these reasons, I would like to stay involved in this area after GSoC ends. Below are a some of the ways in which dimensionality-reduction methods and their use can be enhanced via open-source projects.

1. Flexible discriminant analysis in an important extension of discriminant-based methods. Implementation of this method is lacking in Python and would be of great benefit. This would utilise the implementation of multivariate adapative splines (Py-earth package) and could draw inspiration from the R package “mda”.

2. A further extension to the penalised discriminant analysis would also benefit from open-source implementation.

3. The curse of dimensionality and robust covariance matrices: In Python, sklearn.covariance module implements a number of covariance matrices that are suitable for datasets with a large number of features. However, many methods

mentioned above do not have an option to use these metrices (LDA being an exception). Can some of these methods be extended by allowing the user to use these “robust” covariance matrices?

About me

I am a first-year PhD student in Statistics at Imperial College London. My research interests are centred on high-dimensional statistics and machine learning, and I am currently working on scalable SVM models with theoretical guarantees. Previously, I have completed MSc in Statistics at the same university, where I wrote a thesis on estimating covariance matrices in high-dimensional datasets, which resulted in the publication listed below.

Prior to my doctorate studies, I worked as an econometrician in commodities trading, where I have used ML and statistics to analyse oil markets. Projects I worked on included forecasting impacts of Covid epidemic on the oil markets, developing a forecasting model of refinery throughputs and building a forecast-evaluation framework. All work was done in Python. Prior to that, I worked at the European Central Bank and in Goldman Sachs.

Publications:

- Rybak, J. and Battey, H. S. (2021): Sparsity induced by covariance transformation: some deterministic and probabilistic results. *Proc. R. Soc. Lond. A.*, 477, 20200756.

References

Bellet, A., Habrard, A. and Sebban, M., (2013) A survey on metric learning for feature vectors and structured data. arXiv preprint arXiv:1306.6709

Bickel, P. J., Brown, J. B., Huang, H. and Li, Q. (2009). An overview of recent developments in genomics and associated statistical methods. Philos. Trans. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 367 4313–4337.

Bilenko, N.Y. and Gallant, J.L., (2016). Pyrcca: regularized kernel canonical correlation analysis in python and its applications to neuroimaging. Frontiers in neuroinformatics, 10, p.49.

Fan, J., & Li, R. (2006). Statistical challenges with high dimensionality: Feature selection in knowledge discovery. arXiv preprint math/0602133.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems, Eugen. 7: 179–188. 11.

Fukunaga, K. (1990). Introduction to Statistical Pattern Recognition. Academic Press, Inc., Boston, second edition

Goldberger, J., Hinton, G. E., Roweis, S., & Salakhutdinov, R. R. (2004). Neighbourhood components analysis. Advances in neural information processing systems, 17, 513-520.

Hastie, T., Buja, A. and Tibshirani, R. (1995). Penalized discriminant analysis, Annals of Statistics 23: 73–102.

Hastie, T., Tibshirani, R. and Buja, A. (1994). Flexible discriminant analysis by optimal scoring, Journal of the American Statistical Association 89: 1255–1270.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media.

Johnstone, I. M. and Lu, A. Y. (2009), 'On consistency and sparsity for principal components analysis in high dimensions', Journal of the American Statistical Association 104(486), 682–693.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., & Mullers, K. R. (1999, August). Fisher discriminant analysis with kernels. In Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468) (pp. 41-48). Ieee.

Negahban, S. N., Ravikumar, P., Wainwright, M. J., & Yu, B. (2012). A unified framework for high-dimensional analysis of $ M $-estimators with decomposable regularizers. Statistical science, 27(4), 538-557.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. science, 290(5500), 2323-2326.

Somorjai, R. L., Dolenko, B., & Baumgartner, R. (2003). Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. Bioinformatics, 19(12), 1484-1491.

Suárez, J. L., Garcıa, S., & Herrera, F. (2020). pyDML: a Python library for distance metric learning. Journal of Machine Learning Research, 21(96), 1-7.

Sugiyama, M. (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. Journal of Machine Learning Research, 8(5).

Tenenbaum, J. B., De Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500), 2319-2323.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. 58 267–288.

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. Journal of machine learning research, 10(2).

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the royal statistical society: series B (statistical methodology), 67(2), 301-320.

Zhu J, Rosset S, Hastie T, Tibshirani R. (2004) 1-norm support vector machines. Advances in Neural Information Processing Systems. 2004; 16(1):49–56