Predicting the behavior of immune cell populations from single cell proteomic data

By

Angelova Nelina

“Προβλέποντας τη συμπεριφορά πληθυσμών ανοσοποιητικών κυττάρων από δεδομένα πρωτεϊνικής κυτταρομετρίας”

Supervisor: Prof. Tsamardinos Ioannis, UoC

Committee Members:

Dr. Lagani Vincenzo, ISU

Dr. David Gomez – Cabrero, King’s College

A thesis submitted in conformity with the requirements for

the degree of *Master of Science* in

Bioinformatics

Department of Medicine, University of Crete (UOC)

Heraklion, Crete

2019

Declaration

I, Nelina Angelova, declare that this thesis and the work presented in it are my own and has been generated by me, with the kind guidance of Prof. Tsamardinos Ioannis as a result of my own original research.

I confirm that:

1. This work was done wholly while in candidature for a Master of Science degree at UOC

2. Where any part of this thesis has previously been submitted for a degree or any other qualification at UOC or any other institution, this has been clearly stated

3. Where I have consulted the published work of others, this is always clearly attributed

4. Where I have quoted from the work of others, the source is always given

5. I have acknowledged all main sources of help

_____

Angelova Nelina,

2019

Copyright Notice

Acknowledgments

I would also like to thank my advisor, Dr. Papoutsoglou Georgios, for his guidance, Dr.Tsagris Michalis for his precious advices, and the members of MensXMachina Lab of UoC, for their help and support through my research.

Περίληψη

Από τις αρχές τις ανθρωπότητας, το είδος μας αντιμετωπίζει ασθένειες των οποίων τους μηχανισμούς είναι δύσκολο να αποκρυπτογραφήσει. Ο καρκίνος και η άνοια, είναι λίγα από τα απτά αυτά παραδείγματα, μόνο που σήμερα, οι άνθρωποι έχουν ένα πλεονέκτημα σε σχέση με το παρελθόν: τις έξυπνες μηχανές. Η μηχανική μάθηση, κλάδος του πεδίου της τεχνητής νοημοσύνης, στοχεύει στη δημιουργία συστημάτων ικανών να μάθουν και να αριστεύσουν σε μία δοσμένη εργασία. Ερευνητές που δουλεύουν στο πεδίο της Βιοπληροφορικής, μετατρέπουν αυτή την εργασία σε εύρεση βιοδεικτών, κατηγοριοποίηση ασθενών ή αναλύσεις επιβίωσης.

Πέραν όμως της προόδου στο πεδίο της μηχανικής μάθησης, ταυτόχρονα υπάρχουν εξελίξεις και στα πεδία των επιστημών ζωής. Το πρόβλημα είναι πως οι εξελίξεις των δύο αυτών κατευθύνσεων δε συγχρονίζονται πάντα. Αυτό συμβαίνει και στην περίπτωση της τεχνολογίας της κυτταρομετρίας, μίας αυτοματοποιημένης μεθόδου μέσω της οποίας επιτυγχάνεται η παρατήρηση, ποσοτικοποίηση και ο διαχωρισμός μεμονωμένων κυττάρων. Η πληθώρα πληροφοριών που η τεχνική αυτή αποδίδει είναι υψίστης σημασίας για την κατανόηση των πολύπλοκων ασθενειών που η ιατρική κοινότητα πασχίζει να αντιμετωπίσει, η μηχανική μάθηση όμως, δεν διαθέτει ακόμα τους κατάλληλους αλγορίθμους για την πλήρη και αποτελεσματική ανάλυση και κατανόηση τέτοιου τύπου δεδομένων. Παρ'όλη την προσπάθεια των τελευταίων χρόνων, και τη δημιουργία νέων ή την προσαρμογή παλαιών αλγορίθμων στην δοθείσα πρόκληση, η κοινότητα των επιστημόνων που ασχολούνται με το συγκεκριμένο πρόβλημα είναι σε βρεφικό στάδιο και η ανάγκη για νέες ιδέες και προσεγγίσεις είναι άμεση.

Η παρούσα διπλωματική προτείνει μία νέα μέθοδο ανάλυσης των δεδομένων κυτταρομετρίας, και συγκεκριμένα της δίτιμης κατηγοριοποίησης δειγμάτων. Η μέθοδος εμπεριέχει τη σύγκριση πινάκων κυτταρομετρίας μέσω της μέτρησης της ομοιότητας των κατανομών τους. Δύο αλγόριθμοι προτείνονται, αναπτύσσονται και συγκρίνονται μεταξύ τους αλλά και με άλλους, ήδη υπάρχοντες, καινοτόμους αλγορίθμους, μέσω οκτώ διαφορετικών σετ δεδομένων: ο MMD-KNN, που ακολουθεί τη λογική του αλγορίθμου του πλησιέστερου γείτονα (KNN) και έχει για μετρική του τον MMD (Maximum- Mean Discrepancy), και ο KBCsvm, ένας υβριδικός αλγόριθμος, που μετράει τις ίδιες αποστάσεις αλλά σε άλλη μαθηματική διάσταση (new feature space) μέσω χρήσης Kernel μετασχηματισμών, και ενσωματώνει τις αποστάσεις αυτές σε έναν SVMs αλγόριθμο για την κατηγοριοποίηση των δειγμάτων. Τα αποτελέσματα των συγκρίσεων δείχνουν πως οι αλγόριθμοι πετυχαίνουν αρκετά υψηλές αποδόσεις, ανταγωνίζονται επάξια άλλες σύγχρονες μεθόδους ανάλυσης, και ταυτόχρονα, διαθέτουν χαρακτηριστικά που τους καθιστούν εύχρηστους και κατάλληλους και για περεταίρω είδη αναλύσεων, όπως αυτή της παλινδρόμησης, με ελάχιστες προσαρμογές.

Abstract

Since the beginning of humanity, the mankind struggles with diseases whose mechanisms are hard to be deciphered. Diseases such as cancer and dementia are the open wounds of the civilization of our time, but today, humanity has an advantage that never had before: the aid of smart machines. Machine learning (ML), an artificial intelligence based field, aims at the creation of systems capable of learning and excellence in a given task. Scientists that work within the field of bioinformatics, turn this task into biomarker discovery, patient classification or time-to event predictions.

Besides the advances in the field of ML, there are also advances in the field of Biology and Medicine at the same time. The problem is that sometimes, the advances of these two fields are not synchronized. One of the biggest breakthroughs of our era, is the single-cell technologies, that can measure multi-parameters of one cell at a time, for up to thousands of cells from a given sample, in just one run. The information provided by a single-cell experiment is of great interest and can empower our knowledge about many abnormal conditions in the human body and beyond, but ML lacks the needed algorithms and approaches that could truly understand and model single-cell data. There are some approaches that have emerged in the past few years, each having its advantages and disadvantages in comparison to others, but the community of scientists that try to create and advance them is in its infancy and there is a need for new proposals for sure.

In this thesis, a new approach for single-cell classification tasks is added to the list: an assumption free approach that compares the matrices themselves through the dissimilarities of their distributions. Two algorithms are proposed, a KNN like one that uses Maximun Mean Discrepancy (MMD) as its distance metric, and Kernel Based Custom SVMs (KBCsvm), which can be thought as the advancement of the former. KBCsvm is a supervised approach that combines both the generative and discriminative natures of ML to create a hybrid model, that uses MMD, kernels and Support Vector Machines (SVMs) in its core. The validation of these methods and their comparison with other approaches took place with the use of 8 datasets, and showed that the algorithms perform equally or, in some cases, even better than some existing solutions. In the cases where other algorithms may outperform them, MMD-KNN and KBCsvm may offer better interpretability, assumption-free calculations and lesser human intervention, plus the much needed ability to be extended and adapted to different environments and tasks such as regression and biomarker discovery, following the steps of the state of the art algorithms emerged in the past five years.

Contents

1. Introduction

1.1 Towards a better healthcare with the aid of machine learning

In the era of technology and science, humanity still tries to deal with various diseases whose outbreak cannot be stopped. The causes of various types of cancer, or neurodegenerative diseases, remain a mystery to the doctors, who fail to prognose, diagnose and cure their patients in time. Computer scientists, bioinformaticians and professionals from many other related fields, have brought machines and systems to the doctors' aid, learning them to make predictions, classify patient samples and simulate conditions, as fast and as reliable as possible, in order to turn the odds in their favour.

Machine learning (ML), builds computer systems capable of learning from data, building mathematical models and making predictions based on patterns and inferences, without too many human interruptions. The main goal of this field is to build systems that progressively improve upon a given set task. There are three main types ML: supervised learning, unsupervised learning and learning through reinforcement.

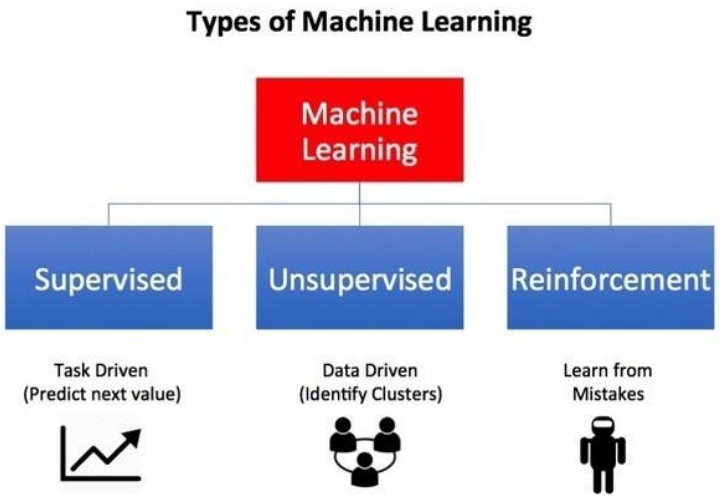


Figure 1: Types of ML(Source).

1.1.1 Unsupervised learning

The machine learning algorithms learn the properties of the data and perform tasks such as grouping, clustering and generally organizing, without the use of labels (disease status, trait, characterization, e.t.c.).They are totally data driven and the human intervention in the learning process is minimized. The imprint of such systems in everyday life, can be seen in areas such as the recommendation systems of YouTube.

1.1.2 Reinforcement learning

Reinforcement learning is as simple as its name: reinforcing the system by providing signals that associate good behaviour with a positive signal and bad behaviours with a bad one, help the algorithm to learn from its mistakes and perform better with time. A typical paradigm is the learning process of mastering at a game play, such as the one learned by the famous AlphaGo application of Google.

1.1.3 Supervised Learning

Supervised learning is the most studied and popular form of learning. Given data in the form of labeled samples from different categories, the system learns the relationship between them and tries to use this knowledge when seeing and classifying new, unseen and unlabeled samples to one of the classes. With the term *sample*, we refer to an instance of a dataset, i.e. a patient of a trial or a student of a class, alongside its information, which is collected and kept in a *structure* (sometimes mentioned as *object*), such as a vector. A sample can be also called an *observation*. There are algorithms that can learn the differences of the blood sample of a cancer patient with the one of a cancer-free individual, and then categorize a new patient accordingly. Some algorithms use a *discriminative* approach, meaning that they learn the decision boundary between the two classes (cancer, non-cancer), and some follow a *generative* approach, modeling the actual distribution of each class. The later handle best uncertainty and missing data, but the discriminative models are perfect for the construction of flexible decision boundaries in recognition problems. Some applications of these powerful algorithms are i.e. in spam classification and face recognition problems.

The samples given to the algorithm in the first fase, where it learns the relationships among them, are called *training examples*, as they are in fact used to train the algorithm. The samples which are given after the learning phase, to test its final distinguishing abilities, are called *test* examples. Given the difficulty of obtaining and creating a whole new dataset for the testing part, the scientist often cut a dataset in folds (chunks), give some data as training and keep the other hidden, an approach known as the *holdout* method. If you repeat the procedure, different folds can be holden out and tested afterwards, in order to get an average estimation of the test error an algorithm scores. If these folds are equally partitioned in k chunks, and each time you keep one out and train with the k-1 ones, it is called a *k-fold cross-validation*.

If the algorithm has parameters to be tuned, i.e. constants which are user defined, play a role in the learning phase and have to be carefully chosen after repeating the learning phase, the training data is cut into two parts: one for training, where the algorithm learns, and one for validation, where the algorithm predicts while tuning the parameters. The testing part defines the algorithm's final performance after the model is created and the parameters are tuned. Once again, the validation and training folds can be exchanged in a k-times repeating process.

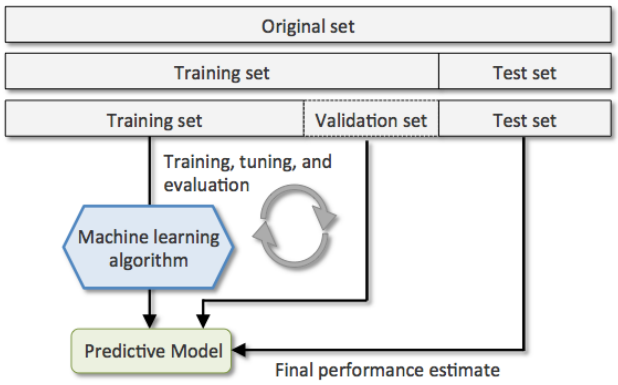


Figure 2: The k-fold Cross Validation Method inside a holdout method. A part is kept for testing (holdout), and with the remaining parts, a k-fold validation is performed so that the best model can be found after tuning the parameters. After a model is chosen, train with all the data except the holdout with the parameters selected, and test on the holdout. If this is repeated with k-folding the test set in an outer loop, the procedure is called «nested» (Source).

1.4 Measures of performance

One can estimate the performance of an algorithm or a method in many ways and by using many known metrics, such as the Accuracy, the Area Under the Curve (AUC), the Sensitivity and Specificity metrics, the F score, and many others.

Accuracy

Accuracy is actually the fraction of predictions the model got right. More formally, it is the number of correct predictions made divided by the total number of predictions made. For binary classification, this could be also calculated as:

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \qquad (1)$$

Although accuracy can tell a lot about the performance of a model, it cannot and should not be trusted alone, due to a problem known as the *Accuracy Paradox*. When one deals with unbalanced datasets, meaning that the dataset that has to be analyzed has more negative than positive samples or vice-versa, although the accuracy he might get may be too good and too high, this does not really reflect the predictive power of the algorithm. For example, when one deals with a rare disease and has 2 samples of cases and 98 of controls, even a naïve ZeroR approach, which is an algorithm that just puts all the samples to the most frequent class, will score an accuracy of 98%, but that does not mean that it actually models something or have any predictive power. This is a typical paradigm of a *baseline* model. By the term *baseline*, we mean a simple method that uses randomness, simple statistics, frequentness, and other naïve methods for classification. The goal is to make methods that outrun baseline approaches in performance and have actual predictive power.

Area under the curve

An ROC curve (Receiver Operating Characteristic Curve) is a graph that shows the performance of a classification model at all classification thresholds. It plots two parameters, the True Positive Rate (TPR), also called Sensitivity, and the False Positive Rate, known as 1-Specificity.

- $TPR = \frac{TP}{TP+FN}$
- $FPR = \frac{FP}{FP+TN}$

Lowering the classification threshold classifies more samples as positives, thus increasing both FP and TP.

AUC measures the two-dimensional area underneath the ROC curve and ranges from 0 to 1. An AUC <=0.5, indicates that the algorithm is no better than one that depends on random guessing. An AUC >=0.8 is very good, and one that

approaches 1 is great. AUC is preferable than Accuracy, since it measures the quality of the predictions rather than their absolute values. Here is a figure that illustrates how a ROC curve looks like, and how different values of AUC can be explained.

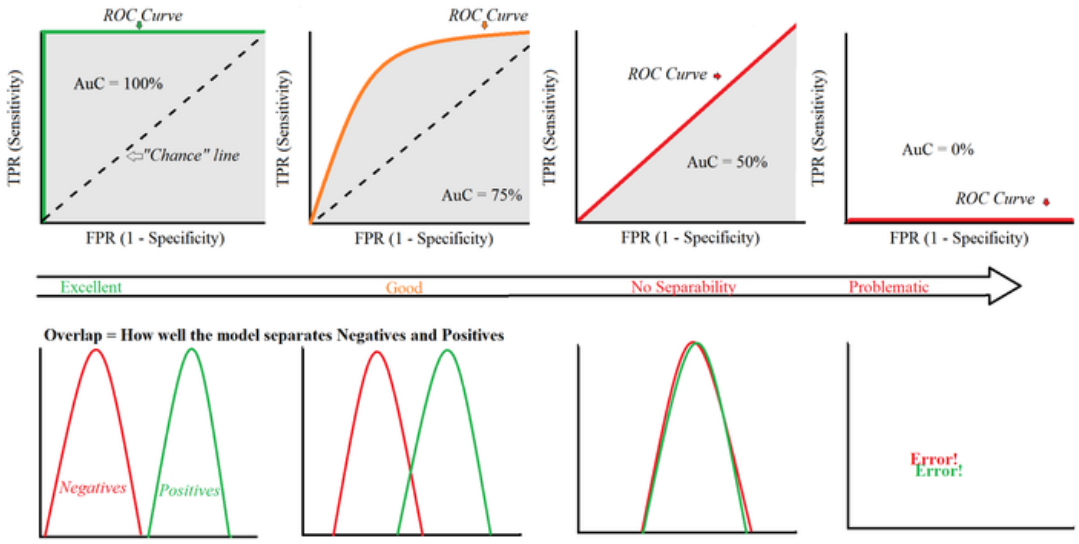


Figure 3: Four different ROC curves and how their AUC can be interpreted, by means of predictive power. (Source)

1.2 Single- cell technologies: Cytometry

Alongside many other scientific breakthroughs, cytometry is a technology developed during the last decades, that offers a great advantage: It can measure properties of the cells of a patient, one cell at a time. In the previous years, all the scientist had was an approximation of the levels of various elements of the blood, bone marrow or some tissue of a patient, and had to make decisions based solely on that information. Now, one can see what is happening on a single-cell level of a patient that carries a disease or a phenotype. Variables that can be measured include cell size, cell count, cell shape, cell structure, cell cycle phase, DNA content, the existence or absence of specific proteins in the cytoplasm or the cell surface, and other. To make such measurements a cytometer employs fluorescent labeling to detect specific antigens using antibodies, intracellular ions using indicator dyes, fluorescent reporter molecules such as green fluorescent protein (GFP), and DNA and RNA using acid- specific probes. Other optical signals can also be measured, including light scatter. Cytometry has played a crucial role in advancing the frontiers of biology, medicine, and technology. Although the term "cytometry" can apply to any method used to extract quantitative information from individual cells, the most common examples are flow cytometry and image cytometry, which are primarily optical methods[1].

Image Cytometry

In *image cytometry* the cell samples are fixed and static. They are usually analyzed by means of microscopy and computational image processing and analysis. Recent trends in image cytometry include high speed imaging, super-resolution imaging, kinetic image cytometry, and machine learning for image analysis and interpretation.

Flow Cytometry

In *flow cytometry*, a large number of cells flow in liquid and are analyzed by a laser beam. It is generally used for analyzing individual cells in a suspension. Its hallmarks are analysis speed, detection sensitivity, the simultaneous measurement of only few variables, and the necessity of sorting individual cells. Recent trends include single cell spectroscopy, single cell mass spectrometry, and the imaging of individual cells in flow. Flow cytometers provide less data than image cytometers, but have a significantly higher throughput, which establishes it as the dominating cytometric method since the mid-1950s.

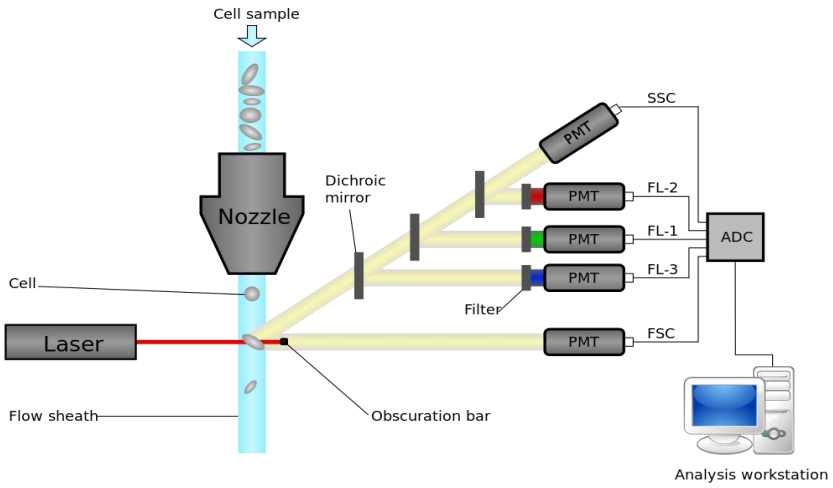


Figure 4: A Typical Flow Cytometer Layout (Source).

The output of a cytometry experiment is a matrix for each sample, with rows as many as the cells examined from it, and columns as many as the variables measured on a single cell level. The variables can be also referred as features. Such information is enormously rich having become the center of research in the field of Machine Learning and Biomedicine lately.

1.3 The problem that arises: Classification of cytometry - generated objects

Supervised learning is strongly related to the medical decisions to be improved. Through it, algorithms that separate healthy from non- healthy patients could be made and are being made over the last decades. The main factors responsible for this kind of separation are studied, and established as biomarkers and targets for drugs. In such classic classification problems, each sample's information that is later fed to the algorithm, is held in a vector. Commonly, the dataset is a matrix made of vectors, where each vector holds the information of a sample (e.g. patient). The columns of this matrix are the variables measured for each sample and form its information. Figure 5 shows an example of such a dataset, related to weather conditions, in which the target (label) is to play or not to play tennis based on the weather conditions of each day. The algorithm learns the relations of the features and then, when fed a new example, it classifies it accordingly as *Yes* or *No* based on the sample's similarities to what it has already seen.

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Figure 5: The weather dataset (Source).

Unfortunately, that is not the case when one tries to classify more complicated objects, such as images, videos, speech (voice recognition), or in this case, cytometry data, where the information for an object is no longer kept in a vector, but in a matrix. That means that now there is not one, but thousands of

vectors representing the same sample (Figure 6). There are approaches that try to tune classic techniques to the nuances of such datasets, but they fail to take into consideration the nature and true complexity of the data.

Vector Sample:

	Var 1	Var 2	Var3	Var 4	
Patient 003	0.430	0.566	1.220	0.765	Normal

Matrix Sample:

Patient 003	V1	V2	V3	V4	V5	V6	
Cell 1	0.55	0.67	0.76	0.77	1.00	0.63	
Cell 2	0.54	0.60	0.07	0.47	0.98	0.24	Normal
...	...	...	...	...	...	...	
Cell n	0.65	0.36	0.08	0.73	0.87	0.50	

Figure 6: The sample of a Normal (class) patient for both cases. The first one is an illustration of its information held in a vector and used in classic algorithms such as KNN. The variables (Var) could be anything, i.e. counts of white cells. The second one is an illustration of the information of a patient's sample, coming from an cytometry experiment, held in a matrix. n is the number of cells measured from that particular sample.

What the community needs, is the construction of new methods able to take into account the complexity of the distributions of such datasets. Thus, the goal is to develop an algorithm that could learn from populations of cytometry data and classify cytometry samples, in a computationally non-intensive, quick, and effective way (performance) by taking advantage of all the available information. At the same time, the approach should be easily interpretable by the scientists using it, making it possible for them to change or further develop it, if and when needed.

1.4 Overview of Suggested solution

There are two main approaches for one to follow, when it comes to the analysis and binary classification of the samples of interest. The first, is based on an explicit new feature construction out of each feature of a matrix independently. The new features are used in vectors that compose the input of algorithms known and used in classic machine learning tasks. The second, is based on the same concept of feature constructions, but the feature construction in this case is performed by looking the data multivariantly. In both cases, there are new features to be constructed, and a feature construction approach may involve assumptions, generalizations, data depletion, and perhaps excessive algorithmic steps or parameterizations.

In this thesis, a new approach for the analysis of cytometry data is proposed. This approach involves the comparison of two matrices (samples) through the

comparison of their underlying distributions. The calculation of the similarity or dissimilarity of the distributions, that yield nothing more than a distance, may take place either in the input space either in a new feature space. The first algorithm proposed uses the distances found through a given metric, in the simple distance based classification method known as the k-nearest neighbor algorithm. The second one is a hybrid algorithm, that uses the distances as a generative input for a discriminative approach, that can work with distances in higher dimensional spaces, known as Support vector machines (SVMs). Discriminative learning algorithms perform better in binary classification problems, but are typically trained from large collections of vectorial training examples. On the other hand, the generative methods model the distributions of the actual classes, which help in many ways. In some cases, it may be preferable to represent the data not as individual samples but as probability distributions. Representing each matrix as a collection of low dimensional feature vectors, which in turn are most naturally represented by a probability density function defined on the space of localized feature vectors, comes handy in many classification problems, but by representing each matrix as a probability distribution, one can cope with multiple drawbacks arising from the nature of the data itself and the experimental conditions under its gathering. It can reduce uncertainty coming from missing data or the variability coming from repeated noisy experiments such as cytometry itself, it can lower the time and space's complexity that can arise from coping with high dimensional data and storing enormous datasets, and thankfully, be used to learn from and create hybrid generative - discriminative approaches that have been the state of the art for the past decade. A joint approach copes with the limitations of both categories. The marrying of the generative and discriminative approaches, involves using kernels to integrate the generative models within a discriminative learning paradigm. The parameters of the generative part, i.e. the median and variance of a distribution found during the first step, serve as features that can be plugged into standard models. Kernels can deal with the nonlinearity of the data and represent the relationships of the samples as good as possible, and a discriminative approach that will use this knowledge will build strong performing classification models.

In the next two chapters, the two already known approaches of univariate and multivariate feature constructions are presented and being discussed, and the newly proposed method alongside the two constructed algorithms are being presented and analyzed. In the forth chapter, the validation and comparison of the methods takes place, with the use of eight cytometry datasets. Finally, chapter five includes the conclusions and future prospects of this work.

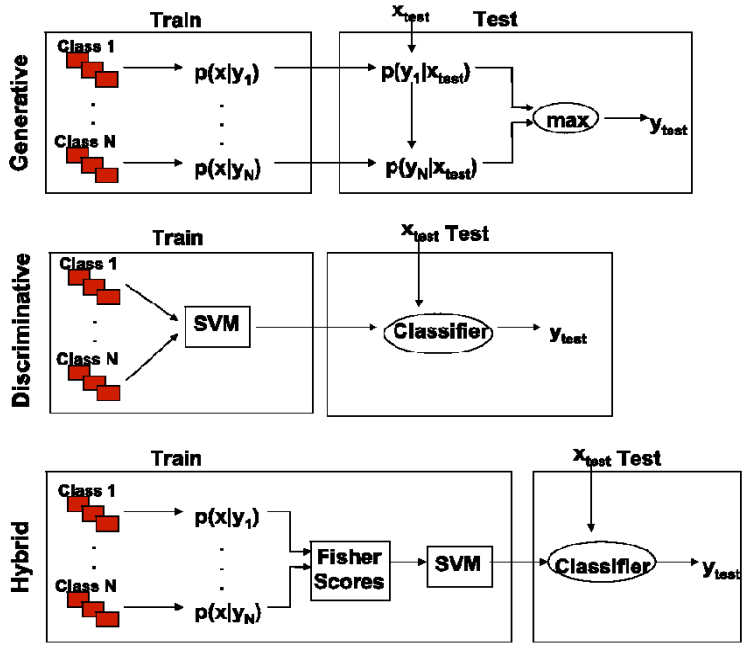


Figure 7: The three models discussed. The Generative model here, uses a Naïve Bayes approach, the Discriminative model uses a SVM, and the Hybrid model incorporates the Fisher scores of the first into the second one to form the decision rule. (Source)

2. Simple methods and State of the art

2.1 Univariate feature construction

The problem can be viewed as a binary classification problem between normal and abnormal cytometry matrices. The first and naivest thing that one could try in order to solve the above task, is to vectorize the available matrices and then use classic supervised classification approaches for the classification part. The vectorization part can be done by constructing new features out of the ones that already exist. For example, given a matrix, one could construct a vector of new features by measuring the mean, median, skewness or any other descriptive statistic of each of the columns, and use this new information for the classification process. By using these new summary features, each sample is now represented by a vector and not a matrix, and classic classification algorithms could work from there. The problem with such a quick but naïve approach is that by summarizing the true information of the data in measures such as a simple median, powerful information is lost and the result is a generalization that fails to capture the complexity of the data. More importantly, the univariate, explicit feature construction, does not take into account the data jointly, and important information about the relationships of

the features may be lost. The approach may partly work for phenotypes with strongly defined differences among cases and controls and when working with a lot of data for training, but that, most of the time, is not the case in the field of bioinformatics, and surely does not work for rare diseases of complex mechanisms. As a part of this research, and in order to check the power of such an algorithm in comparison with other techniques, each cytometry matrix of all the datasets is vectorized, by calculating new features (mean, median, skewness, kurtosis and variance) of each column. For each matrix $M_i^{r\times c}$, where i is the sample, r the number of rows and c the number of columns, the result is a vector of c*5 length. Then, the cases from the controls of each dataset are being distinguished using three different classic classification approaches, KNN, Lasso, and SVMs.

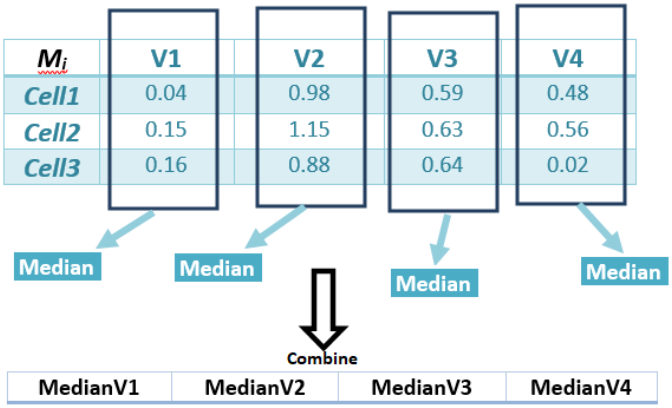


Figure 8: The vectorization method for the median on a matrix of 3×4 dimensions. The same will follow for the mean, skewness, kurtosis and variance statistics.

KNN

K- nearest neighbors is an algorithm widely used in the field of machine learning for both classification (2 or more classes) and regression (one target class) problems. It's a non-parametric method, known for its low computation time and the easy interpretation of its results. In its classification form, the training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm stores the feature vectors and class labels of the training samples. In the classification phase, an unlabeled test example is classified by assigning to it the label which is most frequent among its k nearest training samples (neighbors). Nearest, means those examples whose distance (usually Euclidean or Manhattan) to the new, testing example is the shortest one. k is a user-defined constant, most commonly tuned during a validation step, using a cross-validation approach that tests various numbers as candidates.

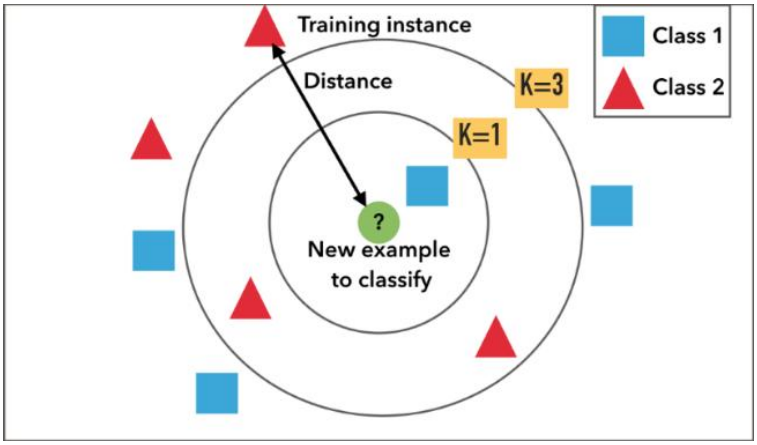


Figure 9: The KNN algorithm for k=1 and k=3. If k=1, the new example will be classified as Class 1 (blue), since its nearest neighbour is blue. If k=3, it will be classified as Class 2 (red), since its 3 nearest neighbours are one blue and two red ones. (Source)

Lasso (GLM)

Logistic regression is a method used to model the probability of a certain class or event existing, when there are one or more independent variables that determine an outcome, through the use of a bit complex cost function known as the "logistic" or "sigmoid" function. This function maps any real value into a value in the range [0,1], and in machine learning, it is used to map predictions to probabilities. It's the perfect predictive analysis to conduct when the dependent variable is binary, as it uses a linear regression equation to produce discrete binary outputs. The goal, is to minimize the cost function and the classification error with it, by using Gradient Descent. Lasso, (Least Absolute Shrinkage and Selection Operator), is a form of such a regression, in which L1 type regularization penalties are introduced. Regularization reduces the magnitude of the coefficients of the features, to stop the model of being driven by few features of big magnitude in an attempt to make a better automatic feature selection. Lambda, is the hyper-parameter of the model. The larger it is, the bigger the shrinkage of the coefficients.

2.2 State of the Art: From univariate to multivariate feature construction

2.2.1 FLOW- CAP Challenges

The analysis of cytometry datasets is indeed very important to the medical community, and as the years go by, researches understand and value the greatness

of the information it can offer more and more. Therefore, they have established whole events just for the reviewing and discussion of the later advances of the filed.

To this direction, the *Flow Cytometry: Critical Assessment of Population Identification Methods* (FLOWCAP) challenges[9], were established to compare the performance of existing methods on two tasks:

1. to determine if automated algorithms can reproduce expert manual cell population identification

2. sample classification, to determine if analysis pipelines can identify characteristics that correlate with external variables (e.g., clinical outcome).

Population Identification challenge

A key analysis of flow cytometry (FCM) data is the grouping of individual cell data records into discrete populations based on similarities in light scattering and fluorescence. This analysis is usually accomplished by sequential manual partitioning of cell events into populations through visual inspection of plots in one or two dimensions at a time (known as gating). In the past two decades, efforts are made for the automation of this procedure. The algorithms able to do such a task are unsupervised approaches that actually cluster the data in a similar way as a human expert. The first FLOWCAP challenge was dedicated to discussing, evaluating and comparing the existed algorithms at the time. As shown by it, machines can perform such tasks with equal and even better performances when compared to human efforts, saving both time and money to the ones interested in such experiments.

Sample classification challenge

The second task one can perform on FCM data, is the classification of the cytometry samples. The second FLOW-CAP challenge event was dedicated to algorithms that try to solve this problem.

The main algorithms of the challenge constructed new features, but some did it in a more sophisticated way, by looking at the features jointly. Cluster the data, or compute 2D histograms as in gating, and extract new features from this information, which can be later used in a classic classification algorithms. Many algorithms used for the population identification challenge, was used as feature constructors, and after the new features were made, they were used as inputs in classic algorithms. For example, flowCore performs hierarchical clustering, extracts features, and then uses Decision Trees. flowPeaksvm performs K-means and later SVMs, and PRAMS

clusters the data and uses the newly constructed features in a L1 penalized logistic regression. So the vectorization approach constructs new features by looking each feature individually, while most of the algorithms here, construct new features by looking them jointly, clustered for example. The following figure displays the list of the algorithms that participated in this challenge, and the comparative results of their performances over three distinct binary classification tasks (check [9] for more information).

Table 3 | Performance of algorithms in the sample-classification challenges on the validation cohort[a]

	Recall	Precision	Accuracy	*F*-measure	Recall	Precision	Accuracy	*F*-measure	Recall	Precision	Accuracy	*F*-measure
	Challenge 1: HEUvsUE				Challenge 2: AML				Challenge 3: HVTN			
FlowCAP												
2DhistsSVM[b]	0.50	0.091	0.50	**0.15**	0.00	0.95	0.99	**0.97**				
EMMIXCYTOM					0.95	0.95	0.99	**0.95**				
flowBin	0.012	0.00	0.45	**0.00**	0.10	0.30	0.92	**0.46**				
flowCore-flowStats	0.56	0.455	0.55	**0.50**					1.00	1.00	1.00	**1.00**
flowPeakssvm					1.00	1.00	1.00	**1.00**				
flowType	0.58	0.636	0.59	**0.61**	0.95	0.95	0.99	**0.95**	0.88	0.71	0.81	**0.79**
flowType-FeaLect	0.55	0.545	0.55	**0.55**	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
Kmeanssvm									1.00	1.00	1.00	**1.00**
PBSC	0.33	0.273	0.36	**0.30**	0.75	0.75	0.94	**0.75**	0.95	0.95	0.95	**0.95**
PRAMS									1.00	1.00	1.00	**1.00**
Pram Spheres	0.36	0.364	0.36	**0.36**					0.90	0.90	0.90	**0.90**
Random Spheres					0.95	0.95	0.99	**0.95**				
SPADE					1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
SWIFT	0.67	0.545	0.64	**0.60**					1.00	1.00	1.00	**1.00**

Figure 10: Performance of algorithms in the sample classification challenges during FLOWCAPII [9]. F-test is a measure that averages precision and recall and resembles AUC.

2.3.2 Citrus

Citrus[12] is a well known and used algorithm that is thought as the advancement of the algorithms that FlowCAP shared. It follows the logic presented above: cluster, construct new features, combine them and classify accordingly. The figure below, demonstrates how Citrus works: Cells from all samples (i) are combined and clustered by using hierarchical clustering (ii). Descriptive features of identified cell subsets are calculated on a per-sample basis (iii) and used in conjunction with additional experimental metadata (iv) to train a model predictive of the experimental endpoint (v). Predictive subset features are plotted as a function of experimental endpoint (vi), along with scatter or density plots of the corresponding informative subset (vii). In this example, the abundance of cells in subset A was found to differ between healthy and diseased samples (vi; H, subset A abundance in healthy patients; D, subset A abundance in diseased patients). Scatter plots show that cells in subset A have high expression of marker 1 and low expression of marker 2 relative to all measured cells (shown in gray).

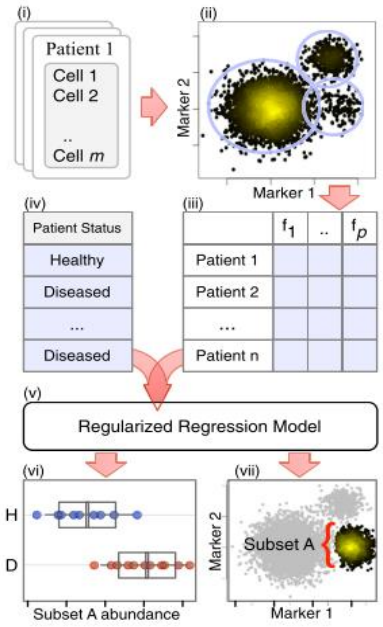


Figure 11: An overview of Citrus[12].

But, although Citrus is a powerful tool that combines un-supervised and supervised approaches for the analysis of FCM data, it has it's drawbacks: excessive stochasticity and too much human intervention (i.e. through the hyperparameters). Additionally, it is not suitable for experiments having a limited number of samples, or for experiments that aim at the identification of really rare subsets (i.e. antigen specific T-cells).

2.3.3 CellCNN

The state of the art methods are developed as approaches that could do both the analyses of classification and population identification stated above in one step. Citrus offers this opportunity, but the most recent and well established such method is called CellCNN [11]. It uses convolutional neural networks (CNN) in its core, a type of artificial neural networks famous for its applications in image processing. CellCNN takes multi-cell inputs (e.g. patient's blood or tissue) associated with a clinical outcome, and associates these inputs with the phenotypes by means of a convolutional network, which learns a concise cell population representation in terms of molecular profiles (filters) of individual cells whose presence or frequency is associated with a phenotype. Simply put, CellCNN treats the frame (matrix) that one gets from a cytometry experiment as an image, and image patches correspond to individual cell measurements. The filter response is evaluated for every cell measurement, which serves as a patch, and then, pooling is performed for each

filter. With pooling, the filter response is taken as an average or as a maximum, for all patches together, and represents the response for each filter on a matrix level. The Pooling layer is connected to the output layer, and the network training optimizes the weights to match the training data set phenotype.

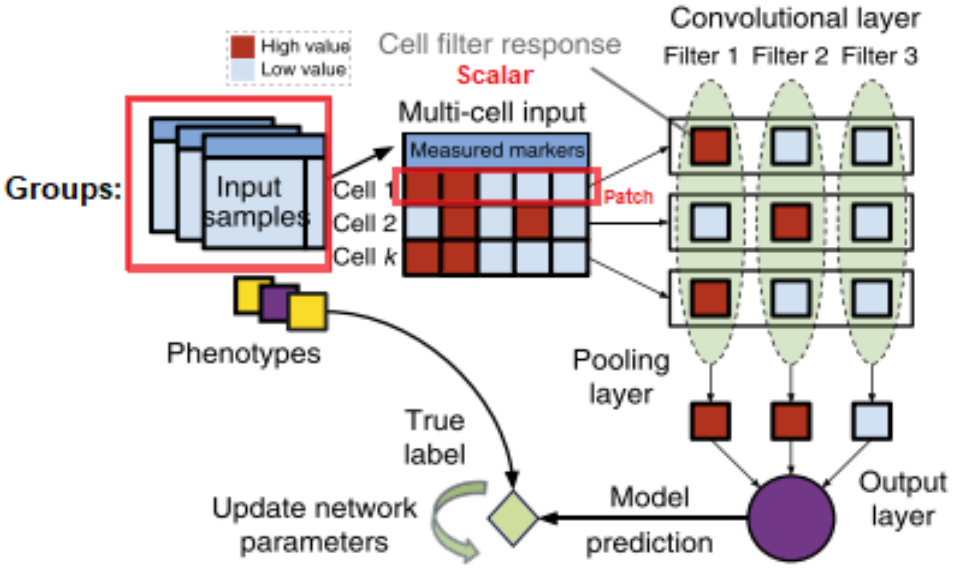


Figure 12: CellCNN's training overview [11].

When the network is trained, the optimized filter weights correspond to molecular profiles of relevant cell subsets and allow for assignment of the cell subset membership of individual cells (cell filter response). When the cell filter response of a test sample is computed, the data is projected and a density-based clustering of high cell filter response regions reveals potential cell-subsets, associated with the phenotype.

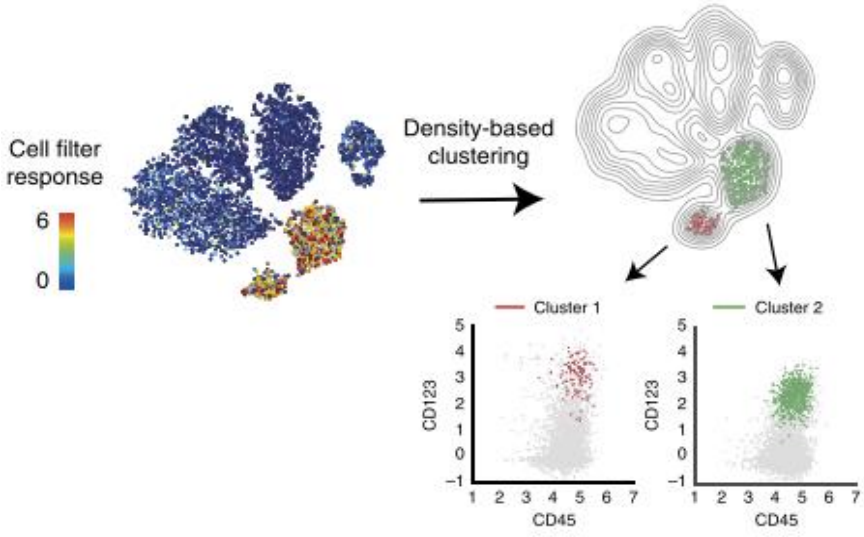


Figure 13: CellCNN's cell filter response overview [11]

The main idea is as follows: model the differences of the two classes, find the features that are more responsible for the decision made, find the cells that have extreme values of that feature, cluster them and check whether they belong to the same cell population or not. If so, components of this population represent a potential drug-target. CellCNN is also a two step procedure that involves both classification and clustering, but these two are now performed vice-versa. It does not yield a class response in a direct way, but it gives filter responses as an outcome. It's superiority to other approaches lie in the fact that it is characterized as an algorithm suitable for the detection of really rare cell-subsets, an ability that Citrus lacks, and its speed, due to the use of artificial neural networks (ANN) techniques. But as the power of the algorithm rises, it's interpretability falls. ANNs are known for their very good performances, but are treated as a black box from researches and scientists, something that limits their use and adaptability.

3. Developing a new approach

3.1 Working with distances of distributions

A different approach, that might capture the nuances of the data more efficiently is the following: instead of vectorising each matrix and end up classifying vectors, or constructing features from clusters or other more sophisticated structures, measure the difference among each pair of matrices through dissimilarity measures and by looking at their distributions, while making as few assumptions as possible. It should be mentioned that although this could be more insightful, the calculations among matrices with thousands of elements could be computationally expensive, and the number of rows of a matrix produced by a cytometry experiment could approach a million. Once the distances are found, they can be used in distance based algorithms such as KNN, or embedded in an abstract Hilbert space through kernelizations. In these new spaces, algorithms that work with kernels, such as SVMs, can classify the samples.

3.2 Generative Part: Kernel Embedding of Distributions

The distributions (P and Q) for the two classes of the binary classification problem are unknown. Some try to first model them and then build kernels upon that knowledge [2,3], but that may be firstly computationally intensive and secondly tricky when it comes to choosing the best way to select and calculate a suitable probability density function (pdf) for the data. The probability distribution selected can and should be based on the user's knowledge of the problem, its calculation can be accomplished using density estimation techniques such as Parzen windows, or, its parameters can be learned using maximum estimation techniques. Its form may directly affect the existence of a closed-form kernel, a kernel that ensures the existence of an optimum solution for the task [2]. Since the computations that involve the modeling of distributions and their comparison can be computationally intense and risky, in the sense that the modeling chosen may not work for the data after all, a solution coming from a mathematical theory known as the *kernel embedding* is used instead. Through it, one can take a measure of dissimilarity between two distributions, without having to know anything about them in the first place.

Kernel embedding of distributions, also called the kernel mean or mean map, is a parametric-free class of methods which represent a probability distribution as an element of a reproducing kernel Hilbert space (RKHS) [4]. The backbone of its theory and success emerge from the simpler notion of the *kernel function (k)*, a positive-definite function introduced half a century ago and gaining enormous popularity in the last two to three decades. Through it, one can perform an inner product $< x, y>$ in a high-dimensional space $\mathcal{H}$ for some data points $x, y \in X$. The existence and positive definiteness of such a function guarantees that there is a dot product space $\mathcal{H}$ and a non –linear transformation (mapping) $\varphi : X \rightarrow \mathcal{H}$ s.t. $k(x, y) = < \varphi(x), \varphi(y) >_{\mathrm{H}}$ without having to compute φ explicitly, a tactic known as the *"kernel trick"*[5]. With the use of this trick, the kernel function can be applied to any algorithm that can be expressed entirely in terms of dot products. Now, if the feature map φ is extended to the space of probability distributions, we could represent each distribution P as a mean function:

$$\varphi(P) = \mu_P := \int_X k(x,\cdot)\, dP(x) \qquad (2)$$

where $k : X \times X \rightarrow R$ is a symmetric and positive definite function. For a class of functions known as characteristic kernels (see Table 1), the mean map $P \rightarrow \mu_P$ is injective, implying that $\left\|\mu_P - \mu_Q\right\|_H = 0$ if and only if P = Q. Consequently, the kernel mean representation can be used to define a metric over the space of

probability distributions and fulfill the needs for the compactness hypothesis to hold, which states that objects that are similar, are also close in their representations, and thus a classification problem may be solved through representing objects by their mutual proximities [6]. This hypothesis puts a constraint on a dissimilarity measure d, which has to be such that d(r, s) for objects r and s is small if the objects are similar, i.e. it should be much smaller for similar objects than for objects that are very different. If we demand that d(r, s) = 0, if and only if the two objects are identical, this implies that they belong to the same class. This can be extended somewhat by assuming that all objects s with a small distance to the object r, i.e. for which d(r, s) is smaller than a sufficiently small threshold, are so similar to r, that they belong to the same class. Consequently, the dissimilarities of r and s to all objects x under consideration should be about the same, i.e. d(r, x) ≈ d(s, x). We conclude, therefore, that for dissimilarity representations satisfying the above continuity, the reverse of the compactness hypothesis holds: objects that are close in their representation are also similar in reality and belong, thereby, to the same class. As a result, classes do not overlap and the classification error may become zero for large training sets. Thus, if P and Q are close in their distances, then μ_P is also close to μ_Q and vice-versa. The functions which can be used as kernel functions are defined by Mercer's theorem, that states that:

A function that can be considered as a kernel, must:

- Be symmetric:
  A kernel function is an inner product of the mapping function; thus we can write:
  $$k(x_i, x_j) = \varphi(x_i)^T \cdot \varphi(x_j) \Rightarrow \varphi(x_i) \cdot \varphi(x_j)^T = k(x_j, x_i)$$
- Be Positive semi-definite:
  Now, if k is a kernel function, a kernel matrix K with $K_{i,j} = k(x_i, x_j)$ can be defined. This matrix is also known as Gram matrix, and merges all the information needed for the learning algorithm, the data points and the mapping function fused into the inner product. For K to be a Kernel matrix, it is necessary to be a positive semi-definite matrix:
  $$\sum_{i,j=1}^{n} c_i c_j \; k(x_i, x_j) \geq 0$$
  for any n ∈ N, any choice of $x_1, \ldots, x_n \in X$ and any $c_1, \ldots, c_n \in R$.

A positive definite kernel defines a space $\mathcal{H}$ of functions from X to R, called reproducing kernel Hilbert space (RKHS), with the following properties:

1. $\forall\, x \in X$, the function $k(x,\cdot)$: $y \to k(x,y)$ is an element of $\mathcal{H}$.
2. an inner product in $\mathcal{H}$ satisfies the reproducing property, i.e. for all $f \in \mathcal{H}$ and x ∈ X, $f(x) = < f, k(x,\cdot) >_H$

For any space $\mathcal{H}$ of functions on X with reproducing kernel k, a linear combination $\sum_{i=1}^{n} a_i\, k(x_i,\cdot)$ forms a dense subset of $\mathcal{H}$, for X of n elements and $a_i, \ldots, a_n$ non-zero real numbers. Since the true distributions P and Q are unknown, it is impossible to compute $\mu_P := \int_X k(x,\cdot)\, dP(x)$ and $\mu_Q := \int_X k(x,\cdot)\, dQ(x)$ directly, and thus, an empirical estimation of mean embeddings should be made instead. Given n i.i.d. samples {x_1,…, x_n}, the most common empirical estimate $\hat{\mu}_P$ of the kernel mean μ_P is:

$$\hat{\mu}_P := \frac{1}{n}\sum_{i=1}^{n} k(x_i,\cdot) \qquad (3)$$

$\hat{\mu}_P$ is an unbiased estimate of μ_P and it converges to it as n approaches infinity (law of large numbers).

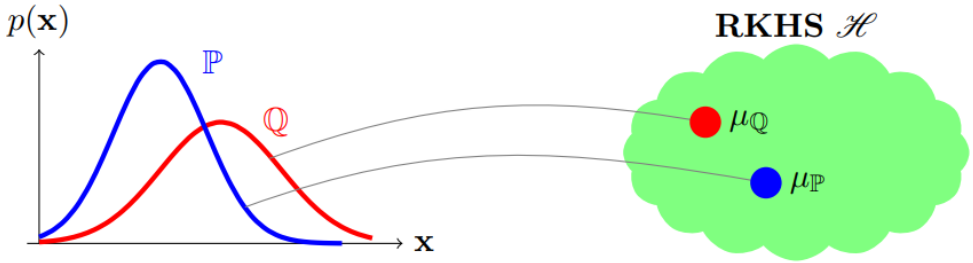


Figure 8: Embedding of marginal distributions: Each distribution is mapped into a RKHS via an expectation operation [4].

A dissimilarity measure that can serve as a metric is needed, and the characteristics of a suitable metric are covered through the compactness theory. What is left is to see how can these embeddings be used to define the metric of need. A metric defined in terms of mean embeddings can be considered a particular instance of an integral probability metric (IPM). Given P and Q as two probability measures on a measurable space X, an IPM is defined as:

$$\gamma[F,P,Q] = sup_{f\in F}\{\int f(x)dP(x) - \int f(y)dQ(y)\} \qquad (4)$$

where F is a space of real-valued bounded measurable functions on X and it fully characterizes the IPM.

Maximum – Mean Discrepancy (MMD)

When the supremum in the IPM function is taken over functions in the unit ball in an RKHS space (i.e. $F \coloneqq \{f | \|f\|_{\mathrm{H}} \leq 1\}$), the resulting metric is known as the Maximum mean discrepancy (MMD).

MMD measures distances between two probability distributions[7]. If x and x' are two i.i.d. samples coming from a distribution P, and y and y' as i.i.d. samples coming from a distribution Q, then the squared MMD is:

$$MMD^2(P,Q) = E(X,X')k(X,X') + E(Y,Y')k(Y,Y') - E(X,Y')k(X,Y') - E(X',Y)k(X',Y) \quad (5)$$

,where k(x,x') is the similarity of x and x' and $E(X,X')k(X,X')$is the average expected similarity of two i.i.d. samples coming from P. Since k(x,x') is a kernel, and a kernel is simply a dot product in a new feature space, its value is large when the samples are similar and small when they're not. The same goes for the terms referring to Y and Y' from Q. The final two terms give the average similarity between a Q generated and a P generated sample. If Q and P are identical, all terms will be same and will cancel, resulting in an MMD of zero.

If MMD is expressed as the distance between mean embeddings in $\mathcal{H}$, we get:

$$\begin{aligned} &MMD[\mathrm{H}, \mathrm{P}, \mathrm{Q}] = sup_{\|f\| \leq 1}\{\int f(x)dP(x) - \int f(y)dQ(y)\} = \\ &sup_{\|f\| \leq 1}\{\langle f, \int k(x,\cdot)dP(x) \rangle - \langle f, \int k(y,\cdot)dQ(y) \rangle\} = \\ &sup_{\|f\| \leq 1}\{\langle f, \mu_P - \mu_Q \rangle\} = \|\mu_P - \mu_Q\|_{\mathrm{H}} \end{aligned} \quad (6)$$

through using the reproducing property of $\mathcal{H}$ and the linearity of the inner product. Thus, MMD can be also expressed in terms of the associated kernel function as:

$$MMD^2[\mathrm{H}, \mathrm{P}, \mathrm{Q}] = E_{X,\tilde{X}}\left[k\left(X,\tilde{X}\right)\right] - 2E_{X,Y}[k\,(X,Y)] + E_{Y,\tilde{Y}}\left[k\left(Y,\tilde{Y}\right)\right] \quad (7)$$

where $\tilde{X}, X \sim P$ and $\tilde{Y}, Y \sim Q$ are independent copies. If $\mathcal{H}$ is characteristic, then MMD =0 if and only if P = Q, and that could make MMD a suitable metric based on the compactness hypothesis.

Since the true distributions are unknown, an empirical estimation of MMD can be learned directly from the samples. Given i.i.d samples X= {x_1,…, x_m} and Y = {y_1,…, y_n} from P and Q respectively, a biased empirical estimate of MMD can be obtained as:

$$\widehat{MMD^2}_b[H,X,Y] \coloneqq sup_{\|f\|_H \leq 1}\{\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{j=1}^{n} f(y_i)\} \qquad (8)$$

Moreover, an unbiased estimate of the MMD entirely in terms of k has as follows:

$$\widehat{MMD^2}_u[H,X,Y] = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i,x_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k(y_i,y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=i}^{n} k(x_i,y_j) \qquad (9)$$

A natural application of the MMD is two-sample testing: a statistical hypothesis test for equality of two samples. During this application, the null hypothesis $H_0 : \|\mu_P - \mu_Q\|_H = 0$ is tested against the alternative hypothesis $H_1 : \|\mu_P - \mu_Q\|_H \neq 0$. One of the bigger pros of this test include the lack of distributions, which means less assumptions, but one of its bigger cons it's the computationally intense calculations that take place and make its computation quadratic in time, which makes MMD almost useless for large-scale problems.

RBF Kernel (Gaussian)

The kernel used is truly important for all these theorems to hold. As said, for the compactness hypothesis to be supported, the kernel must be a characteristic kernel that ensures that no information is lost when mapping the distribution into $\mathcal{H}$ (injective mapping). Other characteristics, such as universality, translation-invariance and positive-definiteness are also important. RBF, a type of Gaussian kernel function, satisfies all these conditions, contrary to other known and widely used kernels such as the polynomial kernel (see Table 1 below), and thus is a great choice for building the approach. Moreover, it's a commonly used kernel for a MMD test on R^d. The Radial basis function kernel (RBF), or Gaussian kernel, is a kernel in the form of a Gaussian function, and is defined as:

$$K_{RBF}(x, x') = \exp[-\|x - x'\|^2/2\sigma^2] \quad (10)$$

where σ^2 is a bandwidth parameter that sets the "spread" of the kernel. Sometimes, the parameter of RBF is called gamma (γ), where γ = $\frac{1}{2\sigma^2}$, and thus:

$$K_{RBF}(x, x') = \exp[-\gamma\|x - x'\|^2] \quad (11)$$

More specifically, the function is of the form of a bell-shaped curve and γ sets its width. The larger the γ the narrower the bell. If x and x' are close together, $\|x - x'\|^2$ will be small. So if γ> 0, it follows that $-\gamma\|x - x'\|^2$ will be larger. Thus, closer x and x' have larger RBF kernel than farther ones. So how could one calculate the dissimilarities between a cohort of matrices using RBF based MMD?

Kernel Function	$k(\mathbf{x}, \mathbf{y})$	Domain $\mathcal{X}$	U	C	TI	SPD
Dirac	$\mathbf{1}_{\mathbf{x}=\mathbf{y}}$	$\{1, 2, \ldots, m\}$	✓	✓	✗	✓
Discrete	$\sum_{s\in\mathcal{X}} w_s \#_s(\mathbf{x})\#_s(\mathbf{y})$ with $w_s > 0$ for all s	$\{s_1, s_2, \ldots, s_m\}$	✓	✓	✗	✓
Linear	$\langle \mathbf{x}, \mathbf{y} \rangle$	$\mathbb{R}^d$	✗	✗	✗	✗
Polynomial	$(\langle \mathbf{x}, \mathbf{y} \rangle + c)^p$	$\mathbb{R}^d$	✗	✗	✗	✗
Gaussian	$\exp(-\sigma\|\mathbf{x} - \mathbf{y}\|_2^2),\ \sigma > 0$	$\mathbb{R}^d$	✓	✓	✓	✓
Laplacian	$\exp(-\sigma\|\mathbf{x} - \mathbf{y}\|_1),\ \sigma > 0$	$\mathbb{R}^d$	✓	✓	✓	✓
Rational quadratic	$(\|\mathbf{x} - \mathbf{y}\|_2^2 + c^2)^{-\beta},\ \beta > 0, c > 0$	$\mathbb{R}^d$	✓	✓	✓	✓
B_{2l+1}-splines	$B_{2l+1}(\mathbf{x} - \mathbf{y})$ where $l \in \mathbb{N}$ with $B_i := B_i \otimes B_0$	$[-1, 1]$	✓	✓	✓	✓
Exponential	$\exp(\sigma\langle \mathbf{x}, \mathbf{y} \rangle),\ \sigma > 0$	compact sets of $\mathbb{R}^d$	✗	✓	✗	✓
Matérn	$\frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|_2}{\sigma}\right) K_\nu \left(\frac{\sqrt{2\nu}\|\mathbf{x}-\mathbf{x}'\|_2}{\sigma}\right)$	$\mathbb{R}^d$	✓	✓	✓	✓
Poisson	$1/(1 - 2\alpha\cos(\mathbf{x} - \mathbf{y}) + \alpha^2),\ 0 < \alpha < 1$	$([0, 2\pi), +)$	✓	✓	✓	✓

Table 1: Various characterizations of well-known kernel functions. The columns marked 'U', 'C', 'TI', and 'SPD' indicate whether the kernels are universal, characteristic, translation-invariant, and strictly positive definite, respectively, w.r.t. the domain X. [4]

First, compute the squared Euclidean distances between any two matrices x_i and x_j and between the two matrices with themselves. The result gives 3 symmetric distance matrices: $D(x_i,x_j)$, $D(x_i,x_i)$ and $D(x_j,x_j)$. The following figure illustrates that for two matrices x^{4*3} and x^{4*3}:

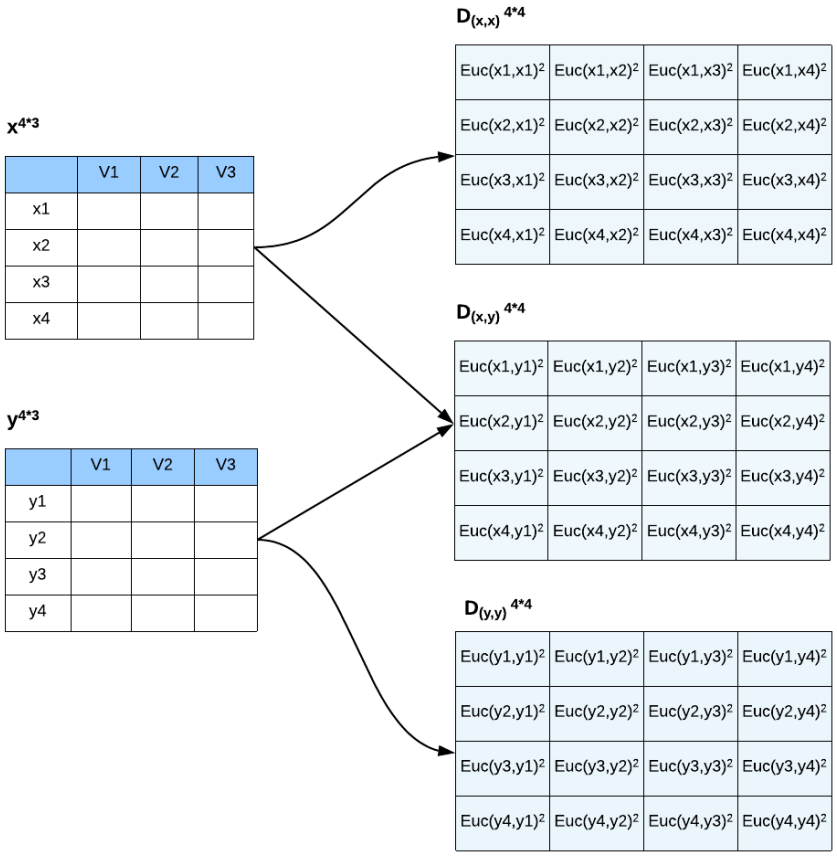


Figure 14: Calculating the distances among each pair of matrices trough the squared Euclidean Distance.

Continuing, all the elements of each of the three matrices $D(x_i,x_j)$, $D(x_i,x_i)$ and $D(x_j,x_j)$ are multiplied with $-\gamma$ and exponentiated.

The trickiest part of using a kernel function, is arguably the tuning of its parameters. As already said, RBF uses the γ parameter, whose value has a great impact at the resulting kernel matrix and the classification task that follows. The function of RBF goes to zero when the distance between x and y is larger than $\frac{1}{\sqrt{\gamma}}$ and that means that gamma defines the region near a fixed point. If its set low, a given datapoint has a kernel greater than zero relative to any example in the set of support vectors, and we get a smooth decision boundary, that may underfit. If its set high, the locality of the support vector expansion increases and that leads to a curvy decision line, which leads to overfitting. More simply, γ defines how far the influence of a training datapoint reaches. Low gamma means “far”, and points far away from the decision line are still included in the calculations of it, while high gamma means “close”, and only the nearest points play a role in the decision procedure.

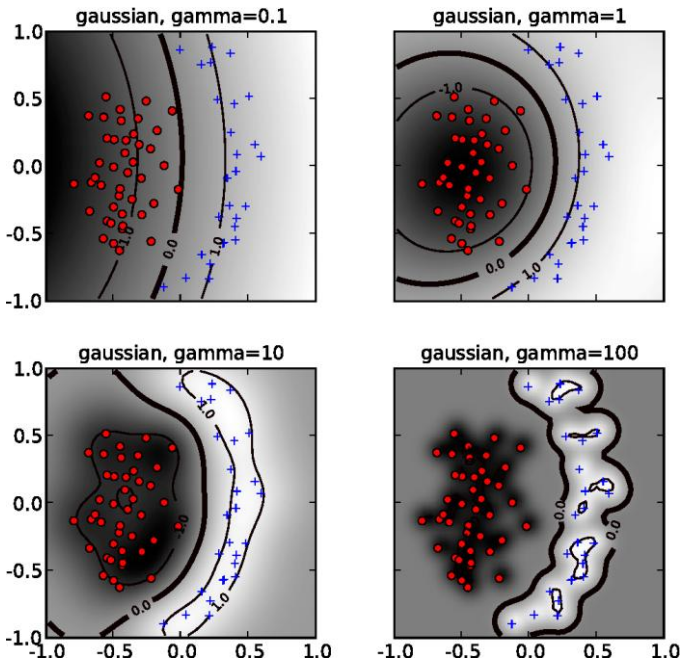


Figure 15: The influence of gamma in the decision line calculations (Source).

Since choosing a gamma for a given problem may be hard and ML aims at a minimum human intervention, the community has proposed several heuristic ways of choosing γ straight from your data. The information of the influence of gamma into the decision process may help one understand why a heuristic solution is preferable: A certain value of γ determines a boundary for the RBF kernel in which the kernel of a point will be larger than a certain value. This boundary determines which points have a saying in the classification process. By choosing it heuristically, usually according to quantiles on the pairwise distances, you ensure that a certain percentage of the datapoints lies within that boundary. This, consequently, ensures that changing a datapoint will only affect the decision function for a certain percentage of datapoints, avoiding the change of the decision function for all or only one datapoint. Precisely, γ is usually obtained through the theory of the *median heuristic*[13], where you first order the elements of the $D_{(x,y)}$ object in an increasing order and then take the median of the result. In practice, the scientists use and other empirical quantiles such as 0.1 and 0.9.

Once the decision upon gamma is made, and all the elements of the three matrices involved are exponentiated, the MMD among i and j can be computed according to (9), which in fact gives a number ranging from 0 to 1. The procedure is repeated for each possible pair of matrices from the cohort. By finding the MMD between each pair of matrices and saving it in each iteration, the algorithm ends up with a matrix D_{MMD}^{n*n}, where each D_{MMDij} holds the MMD distance between the

matrices i and j, for i,j ranging from 1 to n. Now, one will either use these in a KNN approach, which brings us to our MMD- KNN algorithm presented later, or will take it one step further and use them as the generative part of a hybrid model, which is the idea behind KBCsvm.

3.3 Discriminative Part: Custom SVMs

The math behind the powerful SVMs can show how kernels could be actually introduced in a discriminative environment, how the discriminative algorithm changes with the use of a kernel method, and give a hint about how the kernel matrix should be used justifying this use mathematically.

Linear and non-linear SVMs

Support vector machines are discriminative learning models for classification and regression analyses, that belong to the family of supervised machine learning algorithms. Even though it can be extended to more than one classes (multiclass SVMs), it's binary classification capabilities are one of the most used and studied in the field.

Linear SVM

Given a set of labelled training examples (i.e. "case", "control"), an SVM model is a representation of the examples as points in space, mapped in a way such that the examples of the two categories are divided by a clear road that can be thought as a gap, as wide as possible. The model assigns testing examples to one category or the other, based on their mapping into that same space and whether this mapping makes them fall on the one or the other side of the road. The samples on the edge of the boundary lines that separates the two classes, are known as Support Vectors. They are the most difficult samples to classify, their existence and position is crucial for finding the optimum location of the dividing line, and their change affects the road's position. The dividing line will be set in the middle of the distance between the boundary (dotted) lines.

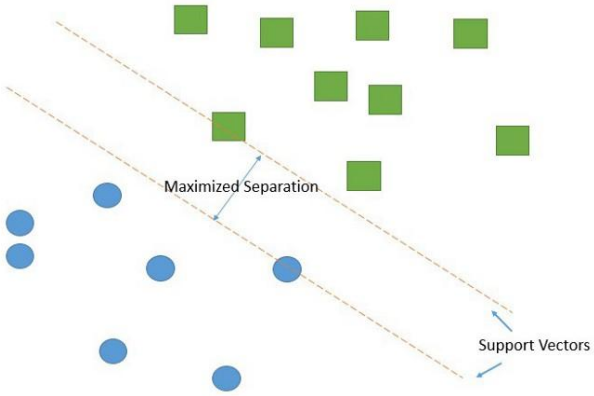


Figure 16: Linear Support Vector Machines (Source).

The above information makes clear that the SVMs are a constrained optimization problem. Optimization problem, because we have to find the line from which the support vectors are maximally separated, a line formally called "margin". Constrained, because the support vectors should be away from the road and not on the actual road.

To solve the problem, the following mathematical concepts should be used:

Lagrange Multipliers

The method of Lagrange multipliers is a strategy for finding the local maxima and minima of a function f, subject to equality constraints. The basic idea behind the Lagrange multiplier theorem, is to convert a constrained problem into a form s.t. the derivative test of an unconstrained problem can still be applied. Suppose we are given a function *f(x,y,..)* and we are looking for its extrema. *f* is subject to the condition *g(x,y,..) = k*. The gradient of the objective function *f,* lines up either in parallel or anti-parallel direction to the gradient of the constraint *g,* at an optimal point. In other words, the gradients should be multiple of one another.

Example:

Obtain the extrema of $f(x,y) = 8x^2 - 2y$ under the constraint $x^2 + y^2 = 1$.

Using a Lagrange multiplier: $\nabla f = \lambda \nabla g; g(x,y) = (x^2 + y^2)$

Equating component wise: $16x = 2\lambda x$; $x(16-2\lambda) = 0$

$$-2 = 2\lambda y;\ y = -\frac{1}{\lambda}$$

From the first part we get x = 0 or λ= 8. Using the constraint, we get y = ± 1; If λ= 8, then:

$$x^2 + \frac{1}{\lambda^2} = 1; x = \pm\frac{3\sqrt{7}}{8} \quad \text{; for } \lambda = 8$$

Thus, the solutions are: $(-\frac{3\sqrt{7}}{8}, -\frac{1}{8})$, $(\frac{3\sqrt{7}}{8}, -\frac{1}{8})$, *(0,1) , (0, -1)*

The function values at these points are : $f(\pm\frac{3\sqrt{7}}{8}, -\frac{1}{8})$ = 8.125 , *f (0,1) =-2, f (0,-1) = 2*

These are the unique minimum, two maxima and the saddle point of the function, and as shown in the following figure, the extrema of a constrained function *f* lie on the surface of the constraint *g,* as our margin should lie on the dotted lines provided by our support vectors.

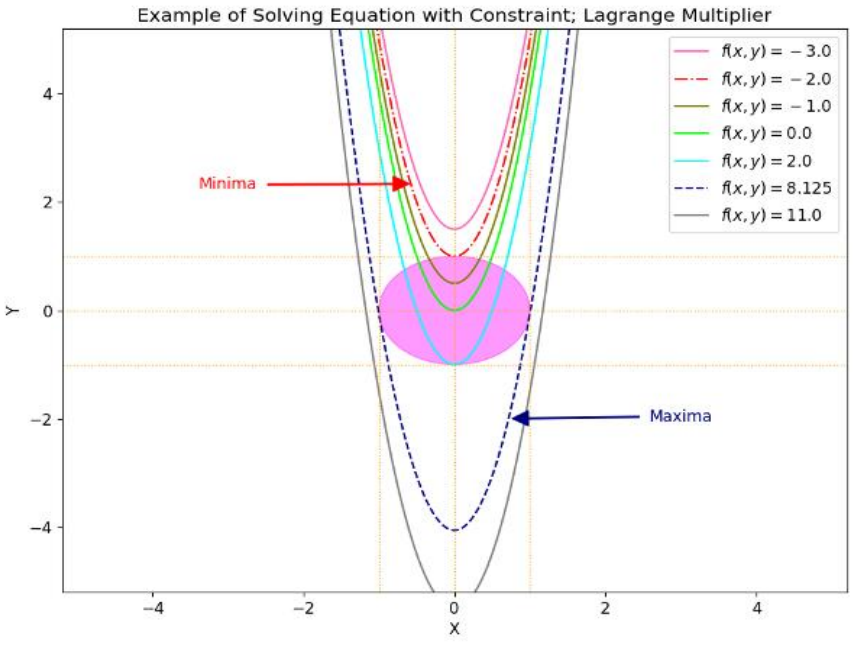


Figure 17: Example of solving equation with constraint: Lagrange Multiplier (Source).

Back to the problem:

Here is a vector W, perpendicular to the median line, and an unknown sample represented by x.

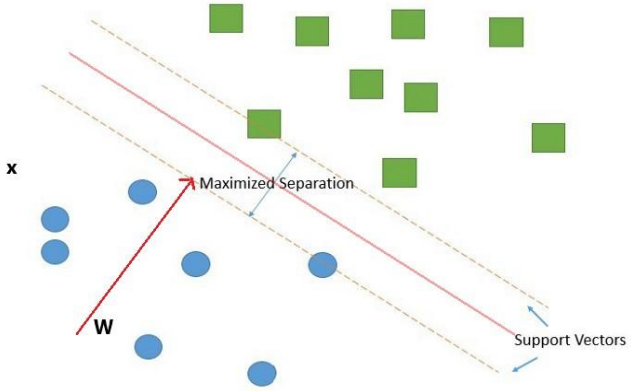


Figure 18: Linear Support Vector Machines, classification of new sample.

To determine on which side of the separating line the new sample lies, a projection of x along the perpendicular to the median line is taken, which is w. If this projection is greater than a number b, called bias, then the new sample is on the right side of the line. And this condition forms our decision rule:

Condition for a sample to be on the right side of the median line:

$$\overrightarrow{w} \cdot \overrightarrow{x} \geq b\,;\ \overrightarrow{w} \cdot \overrightarrow{x} + b \geq 0\,; \qquad (12)$$

To determine w and b, we consider known samples and insist on conditions as below:

$$\overrightarrow{w} \cdot \overrightarrow{x_r} + b \geq 1\,;\ \overrightarrow{w} \cdot \overrightarrow{x_l} + b \leq 1\,; \quad (13)$$

Where *r* and *l* are for right and left respectively. The equality holds for samples on the boundary line, aka the support vectors. Let's also introduce a variable positive for the samples on the right and negative for samples on the left:

$$y_i = \begin{cases} +1; & for\ x_r \\ -1; & for\ x_l \end{cases} \qquad (14)$$

Then, by L.H.S. multiplying the previous equations gives:

$$y_i\,(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) \geq 1; \quad (15)$$

$$y_i\,(\overrightarrow{w} \cdot \overrightarrow{x_i} + b) - 1 \geq 0; \qquad (16)$$

In order to define the distance between the two parallel lines (width of the margin), any two support vectors on either side can be selected, and their difference vector is calculated. The width of the margin will be the dot product of this difference vector with a unit vector perpendicular to the separating line, w. So the width of the road can be formulated as:

$$\text{Width} = (\vec{x}_{svr} - \vec{x}_{svl}) \cdot \frac{\vec{w}}{|w|} \quad (17)$$

And since the goal is to maximize the width of the margin, this gives:

$$\text{Minimize } \frac{1}{2}|w|^2 \quad (18)$$

When applying the Lagrange's method to these conclusions, one gets:

$$a_i(y_{SVi}(\vec{w} \cdot \vec{x}_{svi} + b) - 1) \quad (19)$$

The Lagrangian to be minimized can be written as:

$$L = \frac{1}{2}|w|^2 - \sum_i a_i(y_{SVi}(\vec{w} \cdot \vec{x}_{svi} + b) - 1) \quad (20)$$

Expanding the above expression:

$$L = \frac{1}{2}|w|^2 - \sum_i a_i y_{SVi}(\vec{w} \cdot \vec{x}_{svi} + b) + a_i \quad (21)$$

The independent variables here are w and b, so:

- $\frac{\partial L}{\partial \vec{w}} = 0$; differentiate w.r.t. a vector
- $\frac{\partial L}{\partial b} = 0$; differentiate w.r.t. a scalar

$$\frac{\partial L}{\partial \vec{w}}|w|^2 = \frac{\partial L}{\partial \vec{w}}(\vec{w} \cdot \vec{w}) \Rightarrow \frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_i a_i y_i (\vec{x}_{svi}) = 0 \quad (22)$$

and $\Rightarrow \frac{\partial L}{\partial b} = \sum_i a_i y_i = 0$

From these expressions, one can conclude that the normal vector w is linear combination of the support vectors.

If w and b are getting ridden, one gets:

$$L = \frac{1}{2}(\sum_i a_i y_i (\vec{x}_{svi}))(\sum_j a_j y_j (\vec{x}_{svj})) - \sum_i a_i y_i (\vec{x}_{svi})) \cdot (\sum_j a_j y_j (\vec{x}_{svj})) - \sum_i a_i y_i b + \sum_i a_i \quad (23)$$

Reducing, while knowing that $\sum_i a_i y_i = 0$:

$$L = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i\, a_j\, y_i\, y_j\, \vec{x}_{svi} \cdot \vec{x}_{svj} \qquad (24)$$

and the expression above gives the biggest hint and the most important conclusion:

The maximization depends only on the dot product of pairs of support vectors. Plus, whether a new sample will be on the right of the road or not, depends on the dot product of the support vectors and the new sample:

$$\sum_i a_i y_i (\vec{x}_{svi}) \cdot \vec{u} + b \geq 0 \qquad (25)$$

Support Vector Machines need also tuning. The most important parameter is "cost". Cost is a kind of regularization parameter, that controls the amount of misclassifications the algorithm allows by penalizing them. For large C, the optimization will create a small margin if that helps on getting all the training points correctly classified, but that may lead to overfitting. Small costs on the other hand, may encourage linear separation, but tolerates training errors and may lead to under-fitting.

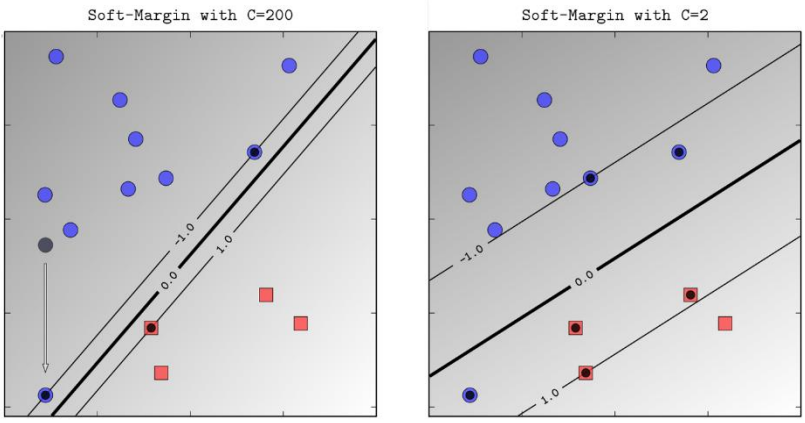


Figure 19: The cost parameter of SVMs (Source).

Non – Linear SVMs: The Kernel Idea

All this information is important and works when the data are linearly separable, and it actually forms the whole idea behind the linear SVMs algorithms. But when one deals with higher dimensional data, which is most of the time, the mathematics won't work, and need a concept known as the "kernel trick"[5]. As

already said, the idea behind kernels is to map a non- linear separable dataset into a higher dimensional space. There, a hyperplane can be found (instead of a line), that separates the samples. If the mapping function (ϕ) that maps the data into that higher space is used, then the maximization and decision rules will depend on the dot products of the mapping function for different samples:

$$\text{Maximization: } L = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i\, a_j\, y_i\, y_j\, \varphi(\vec{x}_{svi}) \cdot \varphi(\vec{x}_{svj}) \qquad (26)$$

$$\text{Decision Rule: } \sum_i a_i y_i\, \varphi(\vec{x}_{svi}) \cdot \varphi(\vec{u}) + b \geq 0 \qquad (27)$$

If we have a kernel function *k,* defined as $k(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$, then one just needs to know *k* and not the mapping function itself.

3.4 Distance measures in a KNN approach: MMD - KNN

As a first attempt, an algorithm which uses the MMD distances directly was constructed, for comparing and classifying the matrices of the data.

The algorithm has as follows: split the samples to training, validation and test sets. Create the D_{MMD}matrix (like already explained in the MMD section), while using all the data. To classify the validation samples, take their columns from D_{MMD}, delete the rows which represent the validation samples themselves as and the test samples, and keep actually only the training rows. The columns will represent the distances of each validation matrix to each training matrix. Short them in ascending order, keep the k minimum ones that represent the k closest neighbors and decide on the sample's label based on the majority vote of the k neighbors' labels, just like in KNN. During the validation stage, tune k (constant) which will serve as our number of neighbors the algorithm will take into account, from a given set: 1,3,5 or 7. In the testing phase, use all other matrices as training samples and exclude only the test rows. Use the best k while sorting the column of each test sample, and classify accordingly. Here is an illustration of the algorithm for a cohort of 10 samples:

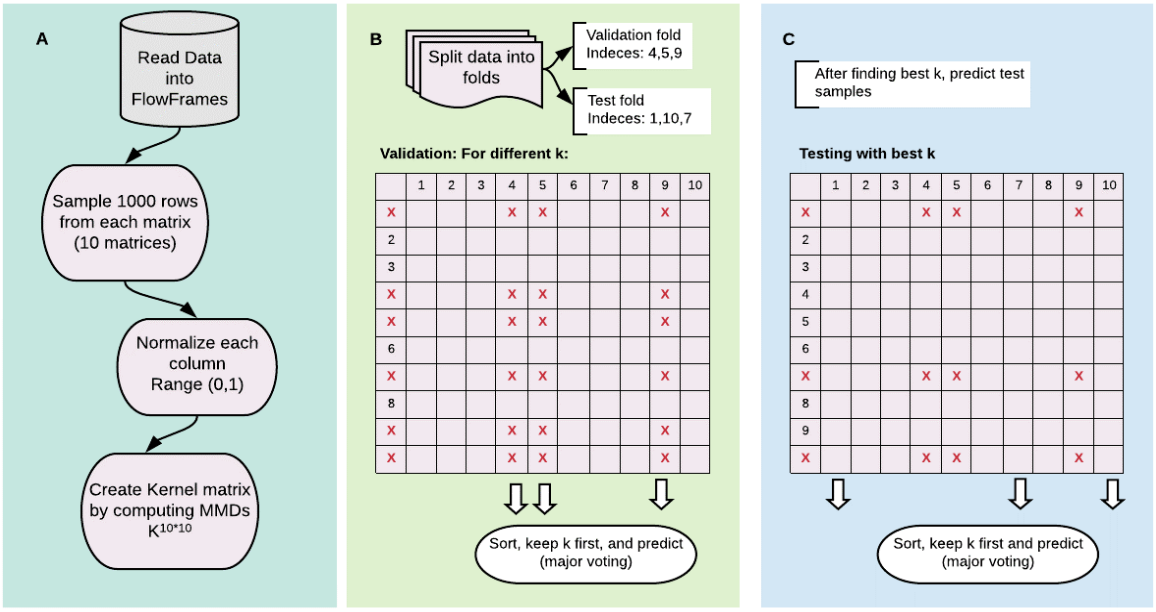


Figure 20: Illustration of the testing phase of the MMD based KNN like algorithm for 10 matrices. A: Reading and pre-processing data, B: Validation Process, C: Testing phase

3.5 A hybrid extended idea: KBCsvms

In order to extend this idea, and use the hybrid approach described in the Introduction section, Kernel Based Custom SVMs (KBCsvm) was created.

3.5.1 Building the generative part: Kernel matrix

The first part of the algorithm aims at the construction of a kernel matrix K out of the kernel functions among each pair of matrices. Each cell of the kernel matrix K, will hold the dissimilarity between two of the matrices translated in a Hilbert space. The questions that arise at this point are two: which metric will be used and how, in order for the compactness hypothesis to hold, and secondly, will the matrix K be a valid kernel, meaning semi-positive definite, in order for the Mercer's conditions to hold. Here comes the kernel embedding theory, alongside MMD and RBF. As said above, if $\mathcal{H}$ is characteristic, then MMD =0 if and only if P = Q, and that makes MMD a suitable metric that supports the compactness hypothesis, a metric that gives the dissimilarity and/or similarity between each pair of our objects. One can ensure that $\mathcal{H}$ is characteristic by choosing the right kernel to use, which in this case, is RBF.

Once the metric is decided, there is a suitable kernel function that ensures injectiveness and positive-definiteness, and the data are available, all one need is an empirical estimation of MMD coming directly from the samples, to measure the dissimilarity among each pair of objects. For a dataset of n matrices, a symmetric D_{MMD}^{n*n} matrix is created, just like in the MMD based KNN algorithm. When it comes to the gamma parameter that RBF uses, with the aim of having more than one gamma to choose from for optimization goals, KBCsvm creates 3 different D_{MMD}^{n*n}matrices: one with the 0.1, one with the 0.5 and one with the 0.9 quantile formed by the data (extension of the median heuristic).

Once one has the D_{MMD}^{n*n} matrix, he needs to clarify whether he has a valid kernel matrix or not to proceed to the discriminative part, and here, the answer is no. MMD gives only a hint about the distance among each pair of matrices, when it calculates the difference among the mean elements of their distributions embedded in an RKHS space. The result of MMD gives $\|\mu_P - \mu_Q\|_{\mathrm{H}}$, and RBF needs and $\|x - x'\|^2$ term, so the distance can be embedded in the kernel function. After columns and rows are normalized, KBCsvm squares the result of MMD, multiplies it by gamma, where gamma is found by applying the median heuristic on the training kernel matrix, and exponentiates it, for every cell of each of the three matrices created. In that way, the distance matrix D turns into a kernel matrix K, and can be finally fed to the discriminative part.

3.5.2 Building the Discriminative part: Custom SVMs

Having computed the kernel matrices, all is needed is to find a way of integrating them into an SVM and use them for the classification part. Since the classification of a new sample (matrix) depends solely on its dot product with the support vectors, one needs to find those support vectors from creating the model with training data, and then, use them to predict the labels of the testing data. Thus, the second, discriminative part is a two step procedure and has as follows:

Split each kernel matrix K to training and testing parts by choosing a mini-matrix of K for training, as shown in the following figure. Feed it to an SVM algorithm customized in a way that allows already formed kernel matrices as inputs, instead of just user defined kernel functions (kernlab R package). Create a model through a k-fold cross validation procedure, and once the SVs are determined, predict the class of each new matrix (test sample) by computing its dot product with the SVs.

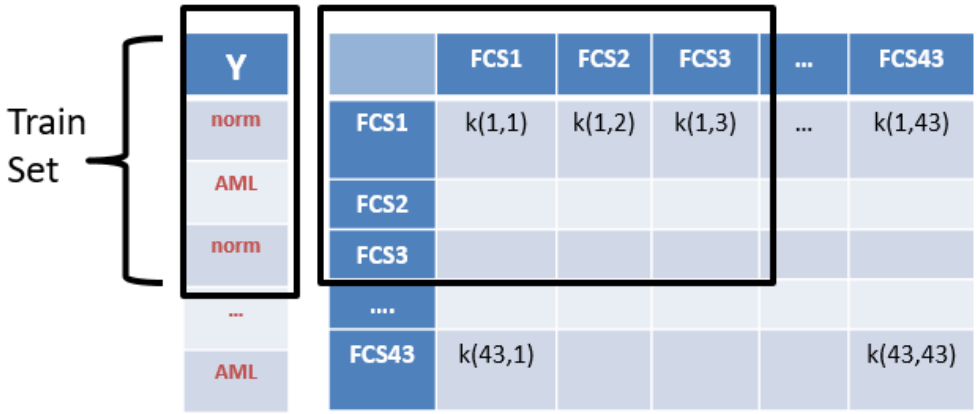


Figure 21: Create a training kernel matrix using a portion of K and train your model with it through 10 fold cross-validation.

Before procceding, one also needs to figure out how to incorporate the tuning part into the model. Besides γ, the cost parameter has also to be tuned, which here, is chosen among the values 0.1,1, or 10. Thus, the final algorithm has as follows:

Create 3 different distance matrices D_{MMD} with the 3 different γ while leaving some test samples as a holdout out of the procedure. Kernelize them, and run each of them in the custom SVM algorithm with all 3 of the cost values. Save the performance (AUC) of each run in a list. That will result in 3 matrices * 3 cost values = 9 entries. The best AUC theoretically reveals the best combination of γ and C for the dataset, so once found, call the MMD - kernel producing method again, to create a D_{MMD} using all the data plus the holdout fold. Once ound, call the custom SVM algorithm with the best cost, train with all the data minus the holdout entries, and finally, test on them. The following diagram shows all the steps of KBCsvm, and the different colours represent the 3 steps needed for the procedure: reading and preproccesing of data, kernel matrix creation and tuning, final model creation and testing.

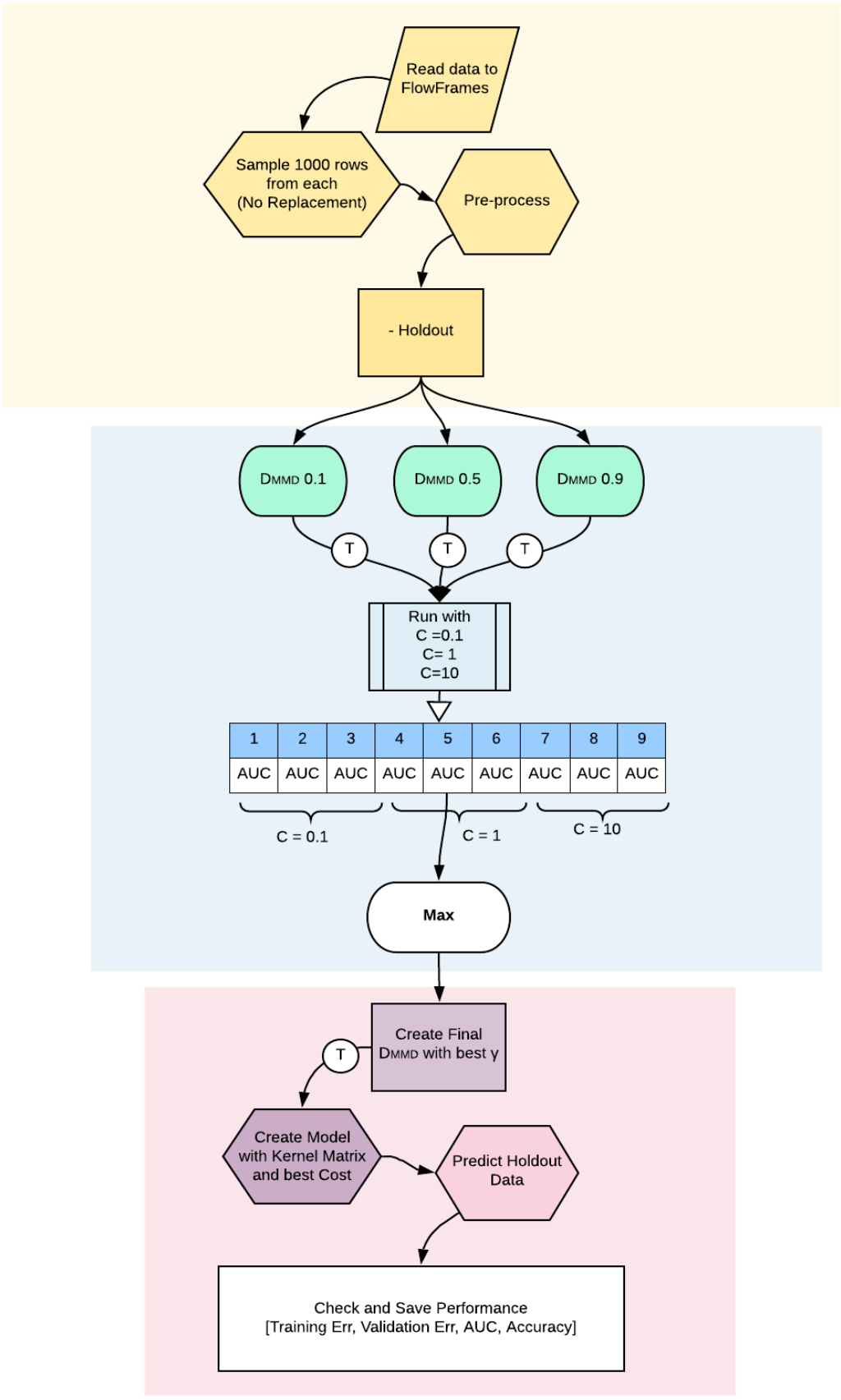


Figure 22: The pipeline of KBCsvm.

4. Results

4.1 Datasets

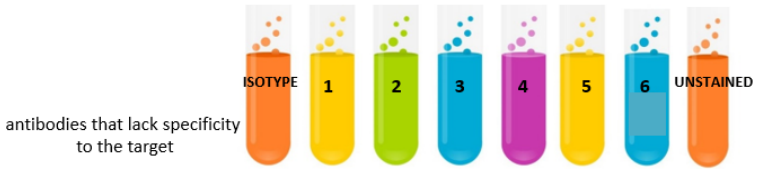
4.1.1 AML

AML is a cohort of 8 flow cytometry experiments, used to evaluate computational pipelines for identification of cell populations that can discriminate between leukemic and normal subjects, and sample classification challenges, of the second FLOWCAP event [9]. Its publicly available through *flowrepository.org*.

Peripheral blood or bone marrow aspirate samples were collected from 359 donors, (43 AML positives and 316 healthy ones) over a 1-year period. Each sample was split in 8 parts (8 tubes), and measured through flow cytometry with the use of different marker and antigen combinations (see following figure). Tube #1 is an isotype control and #8 is unstained. This results into 359*8 = 2.872 FCS files, 8 for each participant. The matrices produced in each file have 30.000 rows, as many as the single cells measured from each tube, and 8 observables: 7 reagents + time of measurement. Isotype controls were set up for each subject tested. Each experiment is highly imbalanced, with way more control samples than case samples.

For the experiments made in this thesis, the "Time" slot was neglected, since it provides irrelevant information for the tasks, as and the 8th tube, since it's unstained. Finally, in order to avoid having results influenced by the accuracy paradox, an equal number of normal observations to the test samples (43 in total) were sampled for running the validation and comparison routines of the algorithms for each of the 7 remaining experiments.

Per-subject: 8 tubes with different marker combinations:



AML 1		AML 2		AML 3	
Channel	Reagent	Channel	Reagent	Channel	Reagent
FSC	FSC	FSC	FSC	FSC	FSC
SSC	SSC	SSC	SSC	SSC	SSC
FL1	IgG1-FITC	FL1	Kappa-FITC	FL1	CD7-FITC
FL2	IgG1-PE	FL2	Lambda-PE	FL2	CD4-PE
FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD
FL4	IgG1-PC5	FL4	CD19-PC5	FL4	CD8-PC5
FL5	IgG1-PC7	FL5	CD20-PC7	FL5	CD2-PC7

AML 4		AML 5		AML 6		AML 7		AML 8	
Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent	Channel	Reagent
FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC	FSC
SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC	SSC
FL1	CD15-FITC	FL1	CD14-FITC	FL1	HLA DR-FITC	FL1	CD5-FITC	FL1	FL1
FL2	CD13-PE	FL2	CD11c-PE	FL2	CD117-PE	FL2	CD19-PE	FL2	FL2
FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD	FL3	CD45-ECD	FL3	7AAD
FL4	CD16-PC5	FL4	CD64-PC5	FL4	CD34-PC5	FL4	CD3-PC5	FL4	FL4
FL5	CD56-PC7	FL5	CD33-PC7	FL5	CD38-PC7	FL5	CD10-PC7	FL5	FL5

Figure 23: The 8 tubes with the different marker combinations involved in each of the independent experiments.

4.1.2 PBMC

PBMC is a mass cytometry dataset available at *Cytobank.org*[10]. The goal of its creation was to develop a cell multiplexing method to increase the sample throughput of mass cytometry, and to apply this method to human peripheral blood mononuclear cells (PBMCs) for the study of cell signalling dynamics and the variability between PBMC donors.

PBMCs from eight healthy human donors were used. Each sample was divided in two, resulting in 16 samples. The 8 of them were left unstimulated and the other 8 were stimulated for 30 minutes before measurement with BCR/fc receptor (FCR). Each FlowFrame has more than 2.000 cells measured and 35 variables (columns). The "Time" slot was neglected in all experiments.

4.2 Experimental Setup

4.2.1 Reading and preparation of data

In this section, the validation and comparison of 6 of the algorithms covered above is conducted: KNN, GLM and SVMs from the univariate, vectorization approach, Citrus from the state of the art section, and the algorithms MMD-KNN and KBCsvms. Before the algorithms can be run, the dataset must be loaded in a proper form and a few preprocessing steps must be made. First, the file of each sample was loaded as a flow frame through the flowCore package of R, and compensated if needed, to neglect the noise or overlap the cytometer may have produced and passed to the data. The data were split into testing and training folds, in order for all the algorithms to be tested upon the same samples. For the 3 latter algorithms, a 1000 of rows were sampled randomly and without replacement from each matrix, due to computational costs (sub-sampling). Again, these are the same 1000 rows for each algorithm. Now, for the pre-processing steps, make non-negative the negative numbers if any, by adding the absolute minimum number of each column to each of its entries. Then, shrink the ranges of each column by taking its log, and then normalize in a way that the values of that column will range from 0 to 1. After a holdout fold is kept for testing, all other data are passed to the training part of each algorithm.

AML

For the leukemia datasets, the algorithms have to distinguish among equal number of leukemic and non-leukemic samples, for a total of 86 samples, measured under 7 different conditions (different antigen combinations). For each of the algorithms, 20% of the data was left out as a holdout testing cohort. The 80% of the remaining data, was used to build a model, tuned for its parameters as following: KNN was tuned for its number of neighbors (choose k from 1 to 9), Lasso was tuned for its lambda parameter (default heuristic of the glmnet package of R), and SVMs were tuned for the cost parameter (c = 0.1,1 or 10) and the gamma parameter of RBF (median heuristic with 0.1,0.5 or 0.9 quantile) in a 7 fold cross validation. When it comes to Citrus, we specified the columns needed for clustering as they advised (surface markers such as CD4), the type of the task (binary classification through nearest shrinkage centroids), the number of rows to be sub-sampled (1000), and the type of the new features it constructs (Calculated Cluster Features = medians). All other parameters were left as they were (default), to avoid the excessive user intervention as much as possible. MMD-KNN was tuned for k (choose 1,3,5 or 7), and its MMD calculations were done using the median heuristic for RBF. KBCsvm was tuned for the gamma parameter of MMD (median heuristic with 0.1,0.5 or 0.9 quantile) and for the cost parameter of its SVMs part (c = 0.1,1 or 10) through a 7-fold cross validation. The kernelization took place with a gamma found through the median heuristic. The holdout samples were classified with the best model of each algorithm. The performance of the algorithms is shown in Table 2 below, and it is an average result of repeating 3 times each of the routines per experiment, in an attempt to eliminate the cherry picking effect.

PBMC

The dataset consists of 16 samples, 8 stimulated and 8 references. Due to the small number of samples, the data were split into 4 stratified folds. The three were used for a raining and tuning, and the 4th one for testing. All said above for the AML dataset still hold, about the preparation and tuning of the data, with the difference that here a 3-fold cross validation was applied (instead of 7), again, due to the small number of samples. The results for the three times repeated routines are summarized in Table 2 below.

4.3 Comparative evaluation results

Method	V-KNN		V-Lasso		V-SVMs		Citrus		MMDKNN		KBCsvm	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
AML1	0.55	0.55	0.98	0.98	0.88	0.88	0.59	0.6	0.68	0.68	0.68	0.68
AML2	0.66	0.66	1	1	0.92	0.92	0.77	0.77	0.79	0.79	0.74	0.74
AML3	0.66	0.66	0.94	0.94	0.96	0.96	0.88	0.88	0.94	0.94	0.9	0.9
AML4	0.61	0.61	1	1	0.98	0.98	0.83	0.83	0.88	0.88	0.71	0.71
AML5	0.61	0.57	0.96	0.96	0.86	0.87	0.7	0.71	0.8	0.78	0.77	0.8
AML6	0.75	0.75	0.94	0.95	1	1	0.96	0.95	0.84	0.83	0.75	0.72
AML7	0.63	0.65	1	1	1	1	0.94	0.94	1	1	1	1
PBMC	1	1	1	1	1	1	0.83	0.83	0.66	0.66	1	1

Table 2: The results of the 6 algorithms and their comparison

The naïve vectorization approaches introduced in section 2, and particularly the vectorization approach with Lasso, score very high performances. MMD- KNN also performs good and outperforms Citrus in many occasions, but it seems that it is not powerful enough when dealing with few data. Citrus performs well, but not great. It looks like it needs the excessive user intervention and like it relies fully on more pre-processing steps to score excellent accuracies. Moreover, it didn't score high in the small dataset of PBMC, which was the purpose behind the inclusion of this dataset in the experiments in the first place. Since Citrus is not good when there are only a few samples available, is the proposed approach here, and especially KBCsvm more suitable? The results show that this could be the case, since KBCsvm is the only one of the 3 more sophisticated approaches that scored good results, at least in these evaluations. At this point, it should be noted that an evaluation of the performance of CellCNN alongside the other algorithms couldn't be established, since to the best of our knowledge, its results export filter weights and not a class membership. With such limitations, a comparison in the context of this thesis could not be carried out.

5. Conclusions and future work

5.1 Discussion

KBCsvm and MMD-KNN use MMD, a metric suitable for directly comparing two matrices of data by looking at their distributions. This comparison is made with no assumptions upon the underlying distributions of the datasets. The difference among the two algorithms lies in the space in which these calculations take place. MMD-KNN works with distances in the input space and KBCsvm works with them in a new feature space constructed through kernels. The two algorithms, as created here, haven't any hyperparameters. The assumption free nature of KBCsvm, comes with the advantageous nature of a hybrid algorithm, that combines the good parts of both discriminative and generative approaches. There is no user-defined parameter, and its internal ones are tuned mostly through heuristic approaches that evolve directly through the underlying data. Finally, besides its good performance in general, the approach seems suitable for dealing with too small datasets too.

Now, although KBCsvm is clearly a sophisticated approach, there is a thing that may sometimes keep its performance from escalating: the great sampling the KBCsvm needs for computational reasons during its 1st phase, where only 1000 rows are taken into account from each sample. The number of rows neglected is high and there is a great possibility that the cells left behind hold even greater information than those kept by the sampling procedure. This problem though, can be easily solved with a more optimized KBCsvm algorithm or when the existing one runs in platforms not overwhelmed by the amount of data. The disadvantages of KBCsvm include the computational intense calculations that MMD needs too, calculations that escalate the time needed for the algorithm to run as the sample number increase. All these apply for MMD-KNN too. Finally, KBCsvm seems to be sensitive to the un-pre-processed form of the numeric data involved. If the dataset consists of features whose measurements are of great range, the pre-processing steps made here are not enough, and different approaches must be followed before one can use datasets of such nature.

5.2 Conclusions

KBCsvm is a one step, fully supervised algorithm, and is a kind of optimization of the also proposed MMD-KNN. Unlike other state of the art techniques, that use a second clustering-based unsupervised step in the beginning or the end of their

procedures, and then construct new features to use alongside classic supervised approaches, KBCsvm uses directly the matrices produced from a cytometry experiment to make its computations. The key steps of the pipeline involved are clearly stated and fully supported by mathematical theories born decades ago and fully used and extended until today. This makes it quite reliable and suitable for future optimizations. The performance of KBCsvm can be described as good, but there are two more main things that make KBCsvm a strong opponent of the state of the art algorithms in our days: First, it beats algorithms such as CellCNN in terms of interpretability. CellCNN builds an artificial neural network that is treated as a black box by the scientists using it, and although it may produce excellent results, our little understanding of it limits our chances of developing it further. Furthermore, it uses no assumptions and has no user-intervention, as approaches such as Citrus do. It offers a quite computationally intensive, but fully reliable and totally machine-driven modelling of the data. Thus, in conclusion, KBCsvm is an algorithm quite different than the ones that already exist, quite powerful, although there is room for improvement, and manageable enough, to be optimized or changed according to new needs, which bring us to our future directions.

5.3 Future work

Although this thesis is focused on creating a new approach for supervised binary classification tasks, it should be mentioned that the algorithms presented here can also be extended to solving regression tasks. Regression is a machine learning method, very similar to what we have already covered, with one, big difference: the final target we would like to predict is not a class among two or more classes, but a continuous variable, that in most cases is no other than time. Regression is suitable for survival analysis and other very important time-to event analyses, that simply try to predict right the amount of time someone or something has until something other happens, e.g. survival time of a patient or time until metastasis happens. Both CellCNN and Citrus include this feature, and KBCsvm can also include it quite easily, since support vector machines, the discriminative part of the algorithm, is suitable for regression tasks.

Furthermore, there is a way KBCsvm can be turned into a really important and helping tool for the field of life sciences. As already stated, the decision boundary found in SVMs depends solely on the samples that are chosen as support vectors during the modelling stage of the algorithm. These samples are the hardiest to classify, and thus do not identify as the most exemplary samples of their class. The most exemplary examples of each dataset are not used as SVs and thus are placed far away from the decision boundary. If one could collect these samples of each

class, verify that they're not far from the margin of the SVMs cause they're outliers, but they are indeed exemplary and strong paradigms of their category, could they be used to find cell populations that are strong correlated with their labels? And if yes, how. If this is applicable, KBCsvm could turn into an approach that does not only perform well (and with few optimizations or better systems extremely well), but could also include biomarker discovery in its core. Arvaniti's work is the state of the art method in our days [11], that makes it possible for such two in one methods to work, but it involves artificial neural networks (ANNs) in its core, which are known for their non- interpretability. Citrus also includes such a feature, but as already stated, it is way human driven. Besides that, its creators recommend that the data it uses to produce good results, must not be in their raw, initial form, but pre-processed in a wet lab with amplification techniques, which wash away the un-wanted cells from the tissue or blood sample in advance. This makes the task easier on the one hand, but on the other, may exclude cell types that could possibly carry traits that give hints about the patient's disease status. KBCsvm could be possibly extended to that, and since it is an algorithm quite free of parameters in comparison to others, and not based on so many assumptions as i.e. the underlying distribution of the data, it may eventually turn into a more suitable candidate for single-cell population identification tasks.

References

1. International Society for Advancement of Cytometry 2019. Cytometry Types.

2. Jebara, T. & Kondor, R. Bhattacharyya and Expected Likelihood Kernels.

3. Moreno, P. J., Ho Hewlett-Packard, P. P. & Vasconcelos, N. *A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications*.

4. Krikamol, M. et al. arXiv : 1605 . 09522v3 [stat . ML] 25 Jan 2017 Kernel Mean Embedding of Distributions : A review and beyond. (2017).

5. Schoikopf, B. The Kernel Trick for Distances.

6. Pekalska, Elzbieta, Paclik, Pavel, Duin, P. W. R. A Generalized Kernel Approach to Dissimilarity-based Classification. **2**, 175–211 (2001).

7. Gretton, A. A Kernel Two-Sample Test. **13**, 723–773 (2012).

8. Department of Mathematics, O. S. U. No Title. (1996). Available at: https://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/grad/grad.html.

9. Aghaeepour, N. et al. CRITICAL ASSESSMENT OF AUTOMATED FLOW CYTOMETRY. **10**, 228–238 (2014).

10. Bodenmiller, B. *et al.* NIH Public Access. **30**, 858–867 (2013).

11. Arvaniti, E. & Claassen, M. Sensitive detection of rare disease-associated cell subsets via representation learning. *Nat. Commun.* 1–10 (2017). doi:10.1038/ncomms14825

12. Bruggner, R. V., Bodenmiller, B., Dill, D. L., Tibshirani, R. J. & Nolan, G. P. Automated identification of stratifying signatures in cellular subpopulations. *Proc. Natl. Acad. Sci.***111**, E2770–E2777 (2014).

13. Garreau, D., Jitkrittum, W. & Kanagawa, M. Large sample analysis of the median heuristic. (2018).