Towards a universal molecular classifier

Stefanos Fafalios

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Prof. *Ioannis Tsamardinos*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 617393

University of Crete
Computer Science Department

Towards a universal molecular classifier

Thesis submitted by
Stefanos Fafalios
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Stefanos Fafalios

Committee approvals: _____
Ioannis Tsamardinos
Professor, Thesis Supervisor

_____
Ioannis G. Tollis
Professor, Committee Member

_____
Ana C. Conesa
Professor, Committee Member

Departmental approval: _____
Antonios Argyros
Professor, Director of Graduate Studies

Heraklion, Dec 2019

Towards a universal molecular classifier

Abstract

Despite the plethora of biological datasets, it is difficult to gather a satisfactory amount of biological data and derive safe conclusions regarding disease diagnosis. On top of that, a challenging problem of disease diagnosis and their treatment is to identify their complex characteristics that span to tens of thousands. We mathematically shape these characteristics using publicly available datasets, from disease studies, via a low dimensional space representation. In essence, using sophisticated statistical methods, suitable for very high dimensional data, we are able to provide a list of the most probable diseases for a given patient, but also include don't-know-predictions (cases with uncertain disease diagnosis). The benefit of this diagnostic framework is that it can assist medical doctors by directing them towards the most probable diseases for a given patient and allows for timely disease diagnosis and treatment. The analysis is time-efficient and can be conducted in real time, with very little computational power and low ram and disk memory requirements.

Περίληψη

Παρά την πληθώρα των βιολογικών δεδομένων, είναι δύσκολο να συγκεντρωθεί μια ικανοποιητική ποσότητα αρκετά ομοιογενών δεδομένων αυτού του είδους, ώστε να προκύψουν ασφαλή συμπεράσματα που θα οδηγούσαν στη διάγνωση κάποιας νόσου. Επιπλέον, στο πρόβλημα διάγνωσης και θεραπείας ασθενειών, επιφέρει πρόσθετη πολυπλοκότητα η δυσκολία προσδιορισμού και ο εντοπισμός των πολύπλοκων χαρακτηριστικών τους, που εκτείνεται σε δεκάδες χιλιάδες. Διαμορφώνουμε μαθηματικά αυτά τα χαρακτηριστικά, χρησιμοποιώντας δημόσια διαθέσιμα δεδομένα από μελέτες ασθενειών, μέσω αναπαραστάσεων χαμηλότερων διαστάσεων. Στη συνέχεια, χρησιμοποιώντας εκλεπτυσμένες στατιστικές μεθόδους, κατάλληλες για δεδομένα πολλών διαστάσεων, είμαστε σε θέση να παράσχουμε μια λίστα πιο πιθανών ασθενειών για έναν συγκεκριμένο ασθενή, αλλά επίσης να συμπεριλάβουμε περιπτώσεις μη πρόβλεψης (περιπτώσεις στις οποίες η διάγνωση ασθένειας είναι αβέβαιη). Το όφελος αυτού του διαγνωστικού εργαλείου συνίσταται στη βοήθεια που μπορεί να παρέξει στους ιατρούς, υποδεικνύοντας τις πιθανότερες ασθένειες ενός ασθενή ώστε να επιτρέψει έγκαιρη-αξιόπιστη διάγνωση και συνεπώς θεραπεία. Η ανάλυση είναι χρονικά αποδοτική. Μπορεί να πραγματοποιηθεί σε πραγματικό χρόνο, με ελάχιστη υπολογιστική ισχύ και χαμηλές απαιτήσεις μνήμης.

Acknowledgements

First and foremost I would like to thank my supervisor Prof. Ioannis Tsamardinos for his support throughout my program, as well as for giving me the opportunity to be engaged in a such interesting scientific field.

I would also like to thank Prof. Georgios Georgakopoulos and Dr. Klio Lakiotaki for their invaluable cooperation, comments and coaching. A special thanks to Ass. Prof. Michail Tsagris for dedicating part of his valuable time, going through the script and providing valuable comments.

Mostly, I would like to thank my family and friends for their unconditional love and support throughout all these years.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Every day huge amounts of blood and tissue samples are collected from patients from all over the world and are being analyzed. The gathering, categorization and even the digital storage of samples in an everyday computer has been an ordinary task for doctors, given the many scientific advances. The natural next step would be, the creation of an easily scalable data analysis framework, able to analyze datasets such as those, in real time, without the need of investing in expensive hardware, capable of categorizing new profiles to any known possible outcome (such as diseases, phenotypes etc). Such a framework could be used as a tool, towards the enhancement of the accuracy and the minimization of the time needed for disease diagnosis of a patient. With the employment of such a tool, a hospital could become a lot more competitive in terms of diagnostic performance and of course treat patients better. Even though research on distinguishing samples with a certain disease from samples without it, based on gene expression data, has progressed, the problem of identifying any disease still remains a relatively virgin research area. Previous research on such problems focused only on correctly differentiating samples with a specific disease from samples without that particular disease [Salem et al., 2017, Bittner et al., 2000, Dudoit et al., 2002, Ben-Dor et al., 2000, Shahjaman et al., 2017, Lee et al., 2008] , or on correctly differentiating subtypes of a specific disease [Khan et al., 2001, Kang et al., 2019]. Furthermore, most published papers lack a large scale experimental evaluation and validation.
We, have identified only one paper [Lee et al., 2019] that attempts to tackle the problem of identifying any disease. [Lee et al., 2019] first model an individual support vector machine (SVM)[Hearst, 1998] for each disease and then train each SVM to differentiate samples with a specific disease from all other samples. The trained models are then integrated into a Bayesian network whose structure is based on prior knowledge from the MeSH disease hierarchy [Lipscomb, 2000]. Finally, in order to estimate the posterior probabilities of the final output, they used the loopy belief propagation algorithm [Pearl, 1982]. Through this framework, they were able to outperform the best documented single genes on 30 out of 32 diseases that were used for evaluation. While their work seems able of addressing the task of

identifying any disease, the framework they employed is rather over-complicated, difficult to implement and not easily scalable on new data and diseases.
Our proposed approach is much simpler and computationally efficient. In effect, we propose a dimensionality reduction framework for classifying high dimensional data. We first apply principal component analysis [Pearson, 1901], that enables us to capture the underlying characteristics of each available dataset in a lower dimensional space. We then adopt an appropriate metric and measure the similarity between one sample and a dataset. The framework is easily scalable to any number of different classes and has been shown to be able to identify a patient's disease, from a single gene expression array, provided that there exist "clean datasets"[1] from that disease. In contrast to typical machine learning approaches, new datasets and classes can easily be added to our classifier without the need of retraining a model on the previously analyzed data. New diseases and datasets can be analyzed and included in the classification of new samples with little to no effort.

[1]Datasets that consist of samples with the exact disease that has to be identified.

Chapter 2

Materials and Methods

2.1 About the data

2.1.1 Gene-expression data

When a biological study takes place, biologists gather either tissue or blood samples from various subjects (e.g. human patients), extract their RNA and place it in an appropriate chip which will produce the expression of each gene that the chip can measure for each specific sample. The produced expression for a specific gene comes in the form of different colors according to the expression level (see figure 2.1). With the appropriate preprocessing on the colored gene expression output, data appropriate for statistical analysis is created. The analysis of this kind of data has many applications. For instance, in drug response analysis, for the categorization and classification of samples to different diseases, for the detection of unknown biological similarities among subjects or diseases, for the creation of a disease network that shows relations between different diseases and among others, for the quality testing of new samples. Our current work is sorely focused on the categorization part, but could also be extended to other uses, such as quality control of new samples.

2.1.2 The Biodataome database

The data were downloaded from Biodataome Lakiotaki et al. [2018] (fig 2.2). Biodataome is a database of genomic and epigenomic data. All available datasets have already been uniformly preprocessed and automatically annotated with Disease-Ontology terms. Each such dataset is a whole biological study (usually on a specific disease) accompanied with a publication. Being able to process the vast amount of information that exists in these datasets will output insights on a huge amount of diseases. For our main experiments, we used datasets from Biodataome that have the following characteristics: created with the GPL570 technology, have no duplicates (i.e. no datasets have common samples), have adequate sample size (sample size ≥ 40) and are labeled with a popular disease (label that appears on

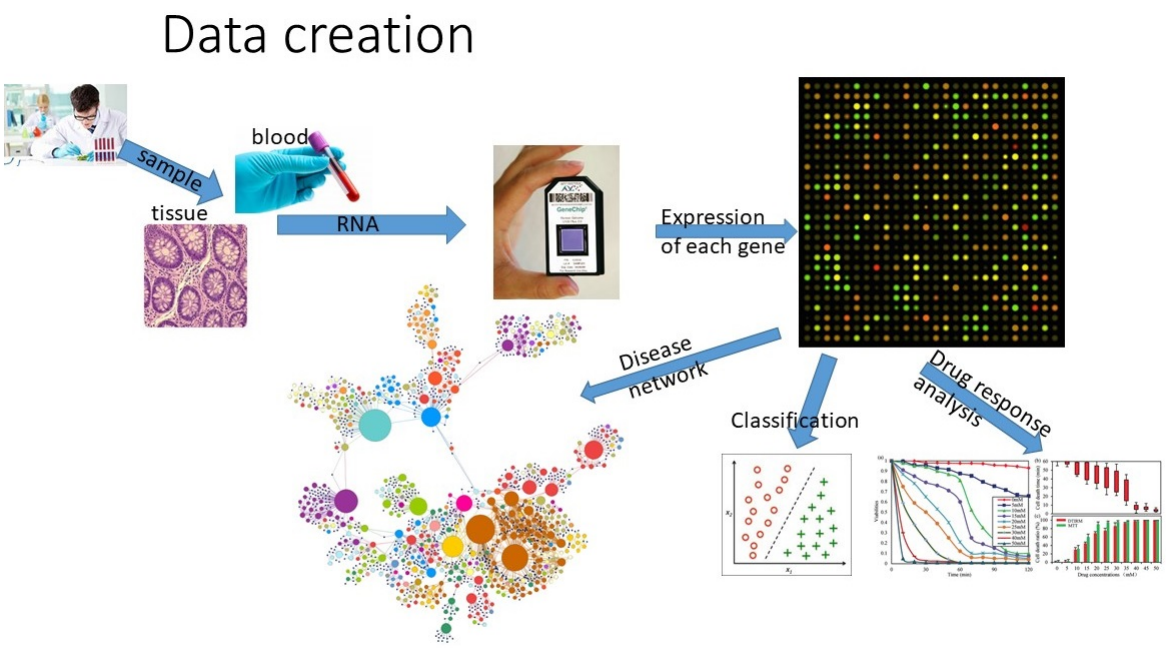

Figure 2.1: Creation and uses of gene-expression data

at least three datasets from GPL570 that have at least 40 samples). These specifications led to 139 datasets with 55,000 features in each dataset, and a total of 13,457 samples.

2.1.3 Manual Curation

The initial dataset labels (those that were provided by Biodataome) are not as disease specific as our experiments would require. Since the problem that we are trying to solve is, categorizing a new sample to a specific disease, we also have to make our datasets disease specific. We hence manually labeled each of the above datasets' samples to a specific disease label. More specifically, by using the disease-ontology terms or the cell line id codes from the information that was provided in the metadata of each such dataset, we were able to label each sample to a specific disease, or to healthy (if the sample was healthy or normal). Using the aformentioned labels, we created a dataset pool containing disease-specific datasets, by splitting each original dataset to newly formed datasets that only consist of samples that have the same disease. Some samples were removed from each dataset, from this newly created pool and were used for evaluation purposes.

Let S_i denote the pool of these removed samples, and let $Pool_i$ denote the pool of the curated datasets with the remaining samples. $Pool_i$ contains 166 datasets, from 37 different diseases, while there are a total of 234 curated datasets spread

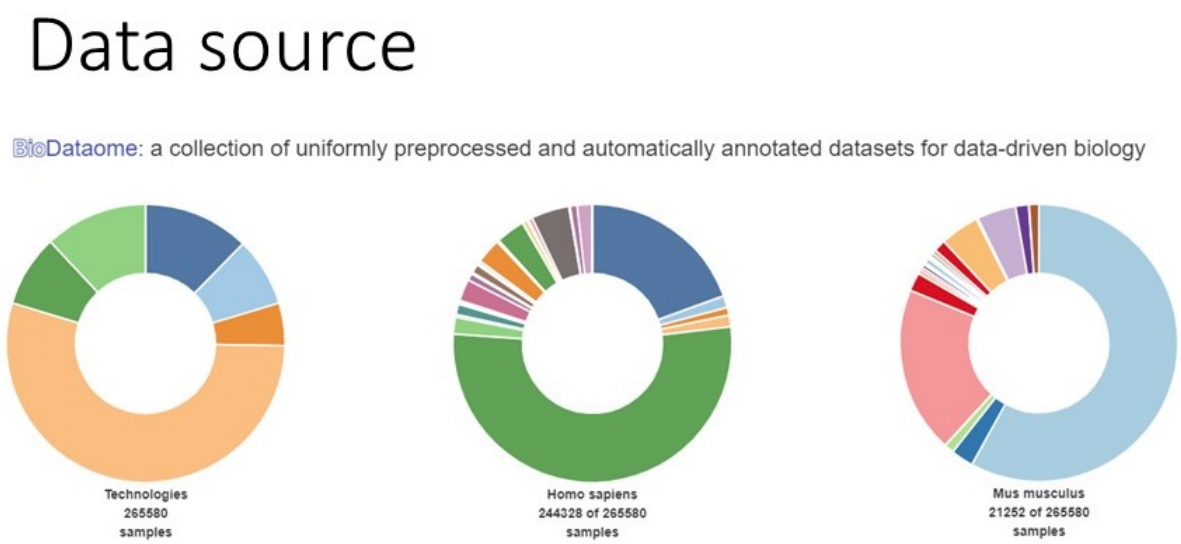

Figure 2.2: Distribution of data in **Biodataome**

among 47 different diseases. In order to assign a label to a sample, we can consult the label of the dataset that appears most similar to the sample in question. However, for the purpose of removing possible dataset bias (such as batch effect bias, etc.) and to adequately solve the problem of predicting a sample to a specific disease, when we are labeling a sample from S_i to a dataset from $Pool_i$ we remove from the possible results, the datasets that originate from the same study as the sample in question. Table 2.1 shows the distribution of samples and datasets of the curated disease specific datasets. As shown in this table, the two most prevalent classes are breast cancer, with 3,629 samples, spread among 38 datasets, which is then followed by the "healthy" (a.k.a. "normal tissue") class which contains 1,297 samples spread among 46 datasets.

2.1.4 Unseen disease cases

Many of the original 139 datasets contained samples of different diseases. This resulted in some disease labels having only one dataset under their name. In cases where a dataset from $Pool_i$ has too few samples, those samples were only used for classification. For these aforementioned reasons, there exist some samples in the classification phase, that are not possible to correctly classify (i.e. there does not exist a dataset in $Pool_i$ with the same label as these samples, or there exists exactly one dataset in $Pool_i$ with the same label, but it originates from the same initial dataset as the samples). These samples were generally excluded from the results, unless stated otherwise.

Disease	Number of samples	Number of datasets
Breast cancer	3629	38
Healthy	1297	46
Colorectal cancer	998	16
Lung cancer	940	11
Asthma	702	6
Lymphoblastic leukemia	638	6
Melanoma	585	10
Glioma	502	8
Lymphocytic leukemia	471	8
Ovarian cancer	391	7
Prostate cancer	337	7
Viral infectious disease	265	3
Rheumatoid arthritis	233	4
Endometrial cancer	175	4
Myeloid leukemia	170	3
Cancer	168	2
Myelodysplasia	159	1
Head and neck cancer	147	3
Hereditary spherocytosis	135	2
Pancreatic cancer	127	3
Gastric cancer	126	6
Ishikawa cells	120	2
Renal cell carcinoma	120	4
Hepatitis c	108	1
Bladder cancer	97	2
Thyroid cancer	94	2
Neuroblastoma	86	1
Ulcerative colitis	82	2
Hepatocellular carcinoma	65	4
Infants born to arsenic exposed mothers	64	1
Multiple myeloma	63	2
Cellular senescence	48	1
Pilocytic astrocytoma	41	1
Squamous cell carcinoma	41	2
Cervical cancer	38	3
Burkitt lymphoma	34	1
Human papillomavirus	30	1
Epithelial carcinoma	27	2
Fibrosarcoma	20	1
Hypernephroma	18	1
Leukemia	18	1
Inflammatory bowel disease	15	1
Osteosarcoma	15	1
Actinic keratosis	10	1
Anaplastic astrocytoma	7	1
Meningioma	1	1

Table 2.1: Total sample and dataset count per disease.

2.1.5 Splitting and dataset pool creation

Let **pop** be a pool of disease specific datasets, and $i = 1, \cdots, 30$ indicate a single run of our algorithms. At each iteration we create a pool of randomly drawn samples from **pop** ($\mathbf{S_i}$) that will be used for evaluation and a pool of datasets ($\mathbf{Pool_i}$) that consist of the rest of the samples that are not present in $\mathbf{S_i}$. More specifically, for each iteration approximately 10% of the samples of each dataset in **pop** are used for evaluation, while the rest of the samples are used for the creation of $\mathbf{Pool_i}$. If a dataset from $\mathbf{Pool_i}$ has too few samples (less than 20), then all the samples from that dataset will be placed in $\mathbf{S_i}$ instead.

2.2 Our Frameworks

We developed two frameworks for the classification of gene-expression data. A log-likelihood based ranker [Fortet and Mourier, 1953] and a Maximum Mean Discrepancy (MMD)[Gretton et al., 2012, Borgwardt et al., 2006] based ranker. The first is a linear method, based on a combination of Principal Component Analysis (PCA)[Pearson, 1901] and multivariate Gaussian probability distribution [Tong, 1990], while the latter, is a non-parametric and not necessarily linear method devised for computing distribution distances in the reproducing kernel Hilbert space [Berlinet and Thomas-Agnan, 2004]. Both frameworks rank, for each given sample, all the available datasets, from the most similar to the least similar. Among them, we favor the log-likelihood based framework, because, through dimensionality reduction, it is very computationally and memory efficient. The MMD based framework was used more as a competing framework. MMD does not perform dimensionality reduction, and we were interested to see how such a framework would perform on the same data. Both frameworks consist of three phases[1]:

- The precomputation phase, where all the datasets are analyzed and appropriate computations for later steps are stored.
- The main phase, where the distances between unlabeled samples and the datasets that were used in the previous phase are computed.
- The prediction phase, were a disease label is assigned to each unlabeled sample according to the distances computed in the previous phase.

After the precomputation phase, a bootstrap phase was later added. Bootstrap provides statistical significance for the distances that were computed in the main phase, and increases the accuracy and robustness of the results[2]. We evaluated the results using the $\mathbf{rank_i}$ probability which indicates the probability of a true

[1] Additional explanation on these phases will be found in section 2.2.3
[2] Elaboration on this phase will be found in section 2.2.3

label of a classification sample to appear in the i^{th} place of the produced disease similarity list.

2.2.1 The log-likelihood classifier

The log-likelihood classifier is based on estimating the probability density of a given sample originating from a specific distribution. This is achieved through **PCA** and the multivariate Gaussian probability density estimation.

2.2.1.1 Log-likelihood classifier for high dimensional data

We assume that each sample from our data follows a multivariate Gaussian distribution $X \sim \mathcal{N}_k(\mu_X, \Sigma_X)$ where X is a random dataset from $Pool_i$, k is the dimentionality of X, $\mu_X = E[X]$, and $\Sigma_X = E[(X - \mu_X)(X - \mu_X)^T]$

Then, for a given sample $x = (x_1, \cdots, x_k)^T$ we can compute the probability that $\mathbf{x}$ comes from the distribution of $\mathbf{X}$ whose probability density function is given by

$$p(x|X) = \frac{\exp\left(-\frac{1}{2}(x - \mu_X)^T \Sigma_X^{-1}(x - \mu_X)\right)}{\sqrt{(2\pi)^k |\Sigma_X|}}$$

The difficulty of using this equation in our case, is that $k \approx 5,5 \cdot 10^4$ and hence the covariance matrix Σ_X will consist of $\mathbf{k \cdot k = 3,025 \cdot 10^9}$ **elements**. Storing such a big matrix requires more than 24Gb of available RAM memory, which can limit the capabilities of a regular computer. Memory limitations aside, another problem that would arise with the above equation, is that Σ_X would be rank deficient, meaning that it would be not invertible. To overcome this computational and mathematical burden, we reduced the dimensionality of each dataset using Principal Component Analysis (PCA). We performed PCA on each dataset, and stored the n_X most significant eigenvectors (P_X^i) and eigenvalues (λ_X^i)[3].

The covariance matrix can be estimated by: $\Sigma_X = P_X L_X P_X^T + I\sigma_X^2$, where I is an identity matrix, L_x is a diagonal matrix containing the eigenvalues λ_X^i and σ_X^2 is the variance that is not explained by the most significant principal components. The multivariate Gaussian probability density function now becomes

$$p_X(x_1, \cdots, x_k) = \frac{\exp\left(-\frac{1}{2}(x - \mu_X)^T P_X M_X P_X^T (x - \mu_X)\right)}{\sqrt{(2\pi)^k |\Sigma_X|}}$$

where M_X is a diagonal matrix, whose diagonal elements are $\frac{\lambda_X^i}{\lambda_X^i + \sigma_X^2}$.

[3]The eigenvectors whose cumulative eigenvalues account for approximately 50% of the variance of the dataset.

The determinant of the covariance matrix can be estimated by the product of the eigenvalues $|\Sigma_X| = \det(\Sigma_X) = \Pi_{i=1}^{n_X}\lambda_i$. Since not all eigenvalues of Σ_X are stored, the determinant could be rewritten as:

$$|\Sigma_X| = \Pi_i^{n_X}(\lambda_X^i + \sigma_X^2) \times \sigma_X^{2(k-n_X)},$$

where n_X the number of the most significant principal components.

Using the equation based on the principal components and the eigen values, enables making computations in such an order that a $k \times k$ matrix will never be created. The biggest intermediate matrixes that would have to be created, would be of size $k \times n_X << k \times k$.
For simplicity we assumed **standard** multivariate normal distribution and in order to avoid numerical overflows (zero roundings), we used the natural logarithm of the probability density function.

$$\ln\left(p_X(x_1, \cdots, x_k)\right) = -\frac{1}{2}k\ln(2\pi) - \frac{1}{2}[(k-n_X)\ln(\sigma_X^2) + \sum_i \ln(\lambda_X^i + \sigma_X^2)] - \frac{1}{2}(x^T P_X)M_X(P_X^T x).$$

2.2.1.2 Tuning the log-likelihood hyperparameter

To employ the log-likelihood classifier, we would need to specify how many principal components should be used from each dataset. For the sake of estimating the optimal hyperparameter, we used all the Biodataome datasets from the GPL570 technology, that were not included in other experiments and contained at least 20 samples. These datasets were not used elsewhere either because they were not labeled with a disease, or because they contained duplicates. The datasets were split in 90% training samples and 10% testing samples. We then estimated how well the log-likelihood classifier can assign test samples to their original dataset, for various percentages of explained variance. As seen in figure 2.3, using only the principal components that explain $\sim 50\%$ of the variance of a given dataset, seemed to be the best choice. This threshold seems to enjoy the best achieved accuracy on the above experiment (left), as well as, the best achieved ranking of the target class on average (right), for this kind of data.

2.2.2 The MMD metric

In order to get good insights on how accurate the log-likelihood's framework performance is, we constructed another similar framework, whose core relies upon the MMD in order to compute the distance between a sample and a given disease. The MMD is given by

$$MMD(X,Y)^2 = E_{x,x'}[K(x,x')] + E_{y,y'}[K(y,y')] - 2E_{x,y}[K(x,y)]$$

where X, Y are two distributions, x, x' are independent random variables from distribution X, y, y' are independent random variables from distribution Y, and

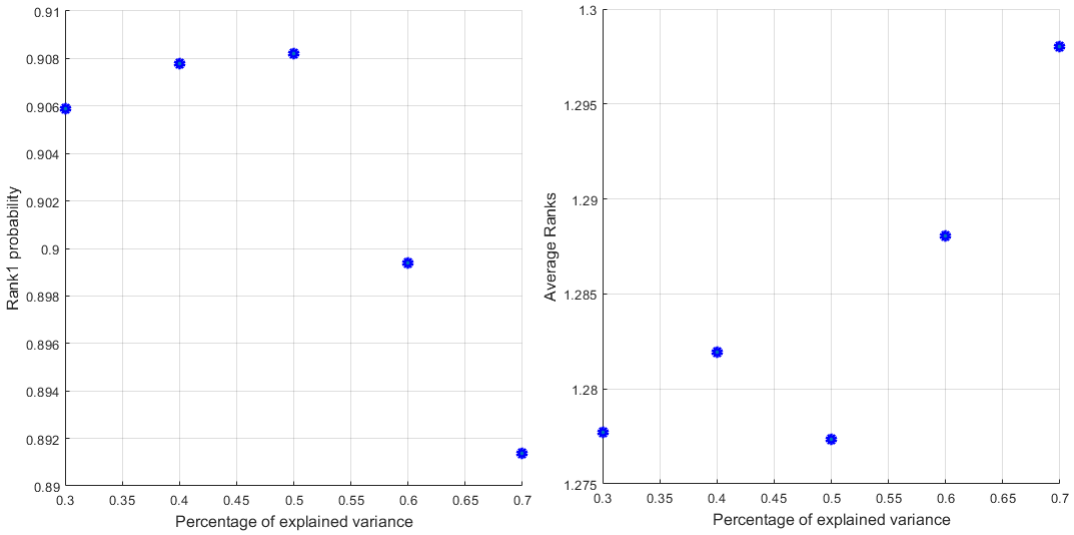

Figure 2.3: Tuning log-likelihood's percentage of principal components to keep Probability of finding a sample's original dataset for different log-likelihood hyperparameters (left). Expected rank value for different log-likelihood hyperparameters (right)

K is an appropriate kernel fucntion. In the case where X, Y are two datasets, the above equation, translates to:

$$MMD(X,Y)^2 = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} K(x_i,x_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} K(y_i,y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} K(x_i,y_j)$$

where n, m are the number of samples in X, Y respectively.
For the purpose of our experiments K was a Gaussian kernel given by:

$$K(x,y) = \exp(-\frac{\parallel x-y \parallel^2}{2\gamma^2}),$$

which is a non-linear symmetric kernel where γ is a hyper-parameter whose value should be tuned appropriately[4].

In the case where X and Y are single multidimensional points (i.e. vectors) and X is identical to Y, then the expected value of the Gaussian kernel will be 1. As a result, if we assume in the MMD equation that X is a dataset with more than 1 samples, and Y only contains a single sample ($Y = y$) [Aytekin et al., 2018], then MMD becomes

$$MMD(X,y)^2 = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} K(x_i,x_j) + 1 - \frac{2}{m}\sum_{i=1}^{m} K(x_i,y)$$

[4]This kernel was initially used. Later, the logarithm of this kernel was found to be more appropriate (see section 2.2.3.1).

Furthermore, as stated in [Sutherland et al., 2016], there is no need to include the values of $K(x_i, x_i)$ in the estimation of the expected value. Regarding the hyperparameter (γ) of the Gaussian kernel, [Gretton et al., 2012] proposed as a rule of thumb value, the median of the distances between the random points of the joint distribution ($X \cup Y$). In contrast to this option, in [Sutherland et al., 2016] it was stated that the optimal hyperparameter, is the one that maximizes the ratio of MMD divided by its variance. In order to estimate γ this way, for each MMD computation, it would be essential to try a range of numerous different hyperparameters and select the one that maximizes the aforementioned ratio. While the rule of thumb value is clearly not the optimal solution (in terms of finding the best γ), the latter is extremely time consuming and hence computationally prohibitive. As a result, for the purpose of our experiments the former method was used to estimate γ, with the difference that we only use X to estimate the median of the distances in the kernel space, instead of using both X and Y, since in our case, Y is a single point. Finally, as was described in section 2.2.3.1, the kernel that was used in our main experiments was the natural logarithm of the radial basis function kernel (log rbf i.e. logarithmic Gaussian kernel). As a result, the final form of the MMD equation that was used in this work is described by:

$$MMD(X,y)^2 = \frac{2}{m(m-1)} \sum_{i=1}^{m} \sum_{j>i}^{m} -\frac{\| x_i - x_j \|^2}{2\gamma_X^2} - \frac{2}{m} \sum_{i=1}^{m} -\frac{\| x_i - y \|^2}{2\gamma_X^2}$$

2.2.3 Phases of the frameworks

1. The precomputation phase
   In this phase, all the datasets from **Pool$_i$** are being analyzed, and necessary precomputations are made according to each of the two classifiers.

   - The log-likelihood framework standardizes each dataset, computes the eigenvectors and eigenvalues of each dataset through PCA and then stores the standardization parameters (mean and standard deviation of each variable) along with the most significant eigenvalues and eigenvectors (those that explain $\sim 50\%$ of the variance of a given dataset), so that they can be used in the next step.
   - The MMD framework standardizes each dataset, estimates the γ hyperparameter of the Gaussian kernel, the first term of the MMD equation ($E[K(x, x')]$) as well as an intermediate matrix for the last term of the MMD equation ($E[K(x, y)]$), and stores all such values for each dataset, so that they can be used in the next step.

2. The bootstrap phase

   Various errors of earlier versions of those frameworks occurred because some datasets in **Pool$_i$**, produced high similarity to many of the samples, regardless of each sample's disease. These datasets produced large likelihood values

and small MMD distances, even when they were compared to seemingly random samples. This indicates that our estimated distances scale differently among different datasets and studies.
To make our classifiers more robust to such datasets, we estimate whether the distance between a specific sample from $\mathbf{S_i}$ and a dataset is, statistically, similar to distances between samples from the joint distribution and the dataset. Specifically, for each distance between a sample in $\mathbf{S_i}$ and a dataset from $\mathbf{Pool_i}$, we test the hypothesis that the sample follows the joint distribution of all other diseases except for the specific disease of that given dataset using bootstrap.
For a given dataset in $\mathbf{Pool_i}$, we estimate the similarity or dissimilarity distribution (log-likelihood or MMD), between the dataset and samples from $\mathbf{Pool_i}$, which do not have the same label as the given dataset. We then fit the Fisher–Tippett distribution [Coles, 2001] (commonly known as Generalized Extreme Value Distribution (**GEV**)) on the computations that correspond to the 25% most similar random samples[5]. By the end of this bootstrap phase, we will have estimated the distribution of the distances of the **most similar**, but also **random** samples, to each dataset[6].
Figure 2.4 depicts an example on how this kind of bootstrapping can help improve the results. Let **A** and **B** be two datasets whose distribution we estimated with random samples. Also, let **x** be a sample which we would like to classify to one of the two datasets. The difficulty with classifying sample **x** in this example is that its distance from both distributions is **2**. Bootstrap allows to see that, for dataset **A**, distances near **2** are most probably distances of random samples with regards to that dataset, while for dataset **B**, this distance seems to be statistically important (small p-value). This results in classifying sample **x** to dataset **B**.

3. The main phase

   In this phase using the precomputations from the previous steps, each framework computes its respective distances, between each sample in $\mathbf{S_i}$ and each dataset in $\mathbf{Pool_i}$. Using the precomputations for each dataset in $\mathbf{Pool_i}$, each sample from $\mathbf{S_i}$ is standardized with the standardization parameters that were used in the current dataset from $\mathbf{Pool_i}$, and all the distances between the samples from $\mathbf{S_i}$ and the current dataset are computed (As previously stated, from this list of distances, in order to exclude any bias, we remove the computations that correspond to the distances between samples and datasets that originated from the same study).
   Then, using the observed distribution that was estimated in the bootstrap

[5]The **GEV** is a type of distribution that combines three simpler distributions into a single form and is often used to model the smallest or largest value among a large set of independent, identically distributed random values, which exactly fits our case.
[6]For further elaboration on the bootstrap phase, consult section 2.2.3.1.

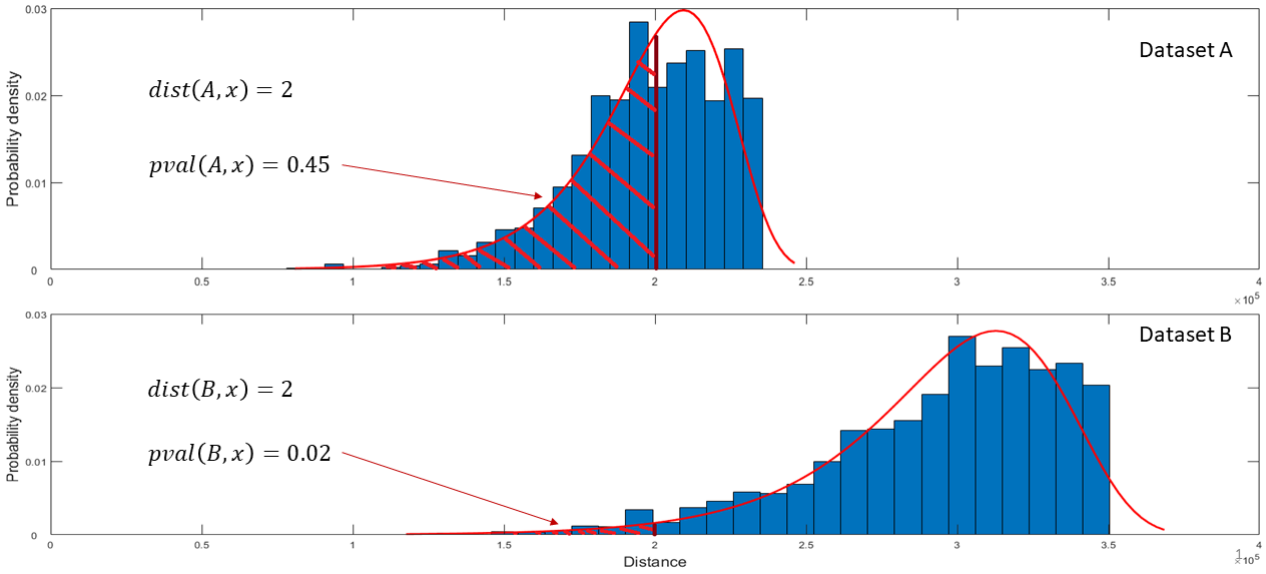

Figure 2.4: Classification example with bootstrapping.

phase, for the current dataset, we can compute how significantly more similar each sample is to the dataset, than the dataset is to some random samples. As a result, the output of the main phase is a distance matrix (in the form of p-values), in which, each row corresponds to a different sample from $\mathbf{S_i}$, and each column corresponds to a computed distance between $\mathbf{S_i}$ and a dataset from $\mathbf{Pool_i}$.

4. The prediction phase

   This phase takes as input the distance matrix that was computed in the previous phase. The values of each row are sorted in such a way that the first values will indicate highest similarity. We can then classify each sample to the disease of the dataset that corresponds to the first value of each such sorted list.
   In order to evaluate how well our frameworks perform, we can consult the rank of a sample's disease. For example, if the first dataset that has the same disease label as a sample that we are classifying, is ranked 1^{st}, then that sample's disease was found in $rank_1$. If the fist dataset that has the same disease as the sample is ranked 2^{nd}, then the sample's disease was found in $rank_2$, etc.
   Having computed the ranks for all the samples that exist in $\mathbf{S_i}$, we can compute the probability of each rank. For instance, estimating the $rank_1$ probability, would be equivalent to estimating the overall accuracy.
   In this phase, if the ratio of datasets per disease was more balanced, it would also make sense to consult the top-K most similar datasets in order to make

a prediction of the sample's disease.

2.2.3.1 Improving the bootstrap phase

In order to make the experiments more robust, we decided to add a bootstrap step to our frameworks. More specifically for each given dataset in the train set, we estimated its log-likelihood distribution with **random** samples from the whole train population [7]. An appropriate theoretical distribution was then fitted, on the above computations for each dataset. Using the fitted theoretical distribution of a given dataset, the level of statistical significance of the distance between each sample and the dataset can be estimated (i.e. how much statistically more similar a given sample is to a dataset, than a random sample). We employed a similar approach for the MMD classifier. Finally, using these distributions, the classification of each sample can be based on the estimated significances, derived from the aforementioned fitted distributions of each dataset. While this method has helped improve the results for the log-likelihood classifier, adding a bootstrap step did not increase the accuracy of the MMD classifier as much. This was probably due to the fact that the theoretical distribution did not always fit the empirical MMD distribution accurately enough. This can be seen in figures 2.5 and 2.6. While the observed log-likelihood distributions are being adequately fitted, there are cases of some datasets where the MMD computations can not be fitted correctly. For example in figure 2.6 the fitting on the first dataset seems to be accurate, but the fitted distribution is not very representative of the observed distribution on the second dataset.

The rbf kernel's values, which MMD uses at it's core, are bounded in [0,1]. By using the logarithm of the rbf's function we unbounded those results (the kernel's values will now lie between $-\infty$ and 0). This way we ended up with a distribution that is easier to be fitted. Further, we figured that it is not necessary to simulate the whole distribution of the random samples. Only the part of the distribution that plays the biggest role during the classification phase needs to be accurately estimated. As a result, we fitted a generalized extreme value distribution on the 25% most significant part of each observed distribution, for both the MMD and the log-likelihood classifier (i.e. we estimated only the most significant part of each dataset's distribution for each algorithm). Figure 2.7 shows how well the **GEV** can fit the most important part of each distribution for both metrics. The top row shows the whole random distribution for the **MMD** metric on a specific dataset, and the fitting of the **GEV** on the 25% most important part of that distribution, while the next row, depicts the equivalent distribution fitting on the same dataset for the **log-likelihood** metric.

[7] With the term random we mean samples that do not share the same distribution as the given dataset.

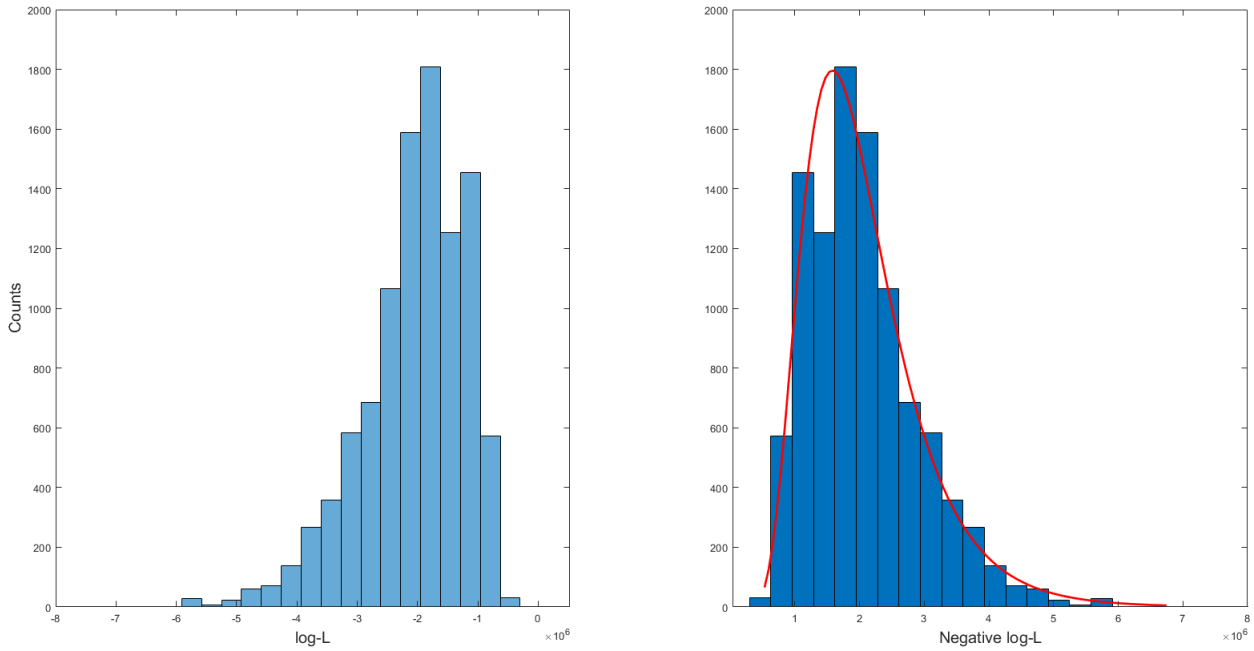

Figure 2.5: Fitting a log-normal distribution on the log-likelihood estimations The fitting is employed between random samples and a randomly selected dataset.

2.3 T-SNE representation

In order to have a better understanding of what our frameworks "see" when we are classifying a sample to a disease, we decided to represent our datasets in a two-dimensional space. Then, representing a sample in the same two-dimensional space, would provide visual interpretation of our results, and possibly a better understanding of what goes wrong when missclassifications happen. We have already described a way to compute "distances" between a sample and datasets. To achieve our desired visualizations we also need a way to compute dataset-to-dataset "distances". The curated Symmetric Kullback–Leibler divergence (cSKL)[Lakiotaki et al., 2019], is a data-driven approach for estimating the distance between two high dimensional distributions, from lower space representations. Since log-likelihood's math was based on the above work, the cSKL seemed like a fitting approach to achieve accurate visualizations. The log-likelihood "distances" are expressed in the form of statistical significances (p-values). As a result, we created a boostrapping phase for the cSKL algorithm as well, so that both kinds of distances will be in the same scale.

To estimate the statistical significance of the computed cSKL between two different statistical distributions, DS_X and DS_Y that correspond to two datasets X and Y, we estimate the cSKL between DS_X and 20 randomly formed datasets

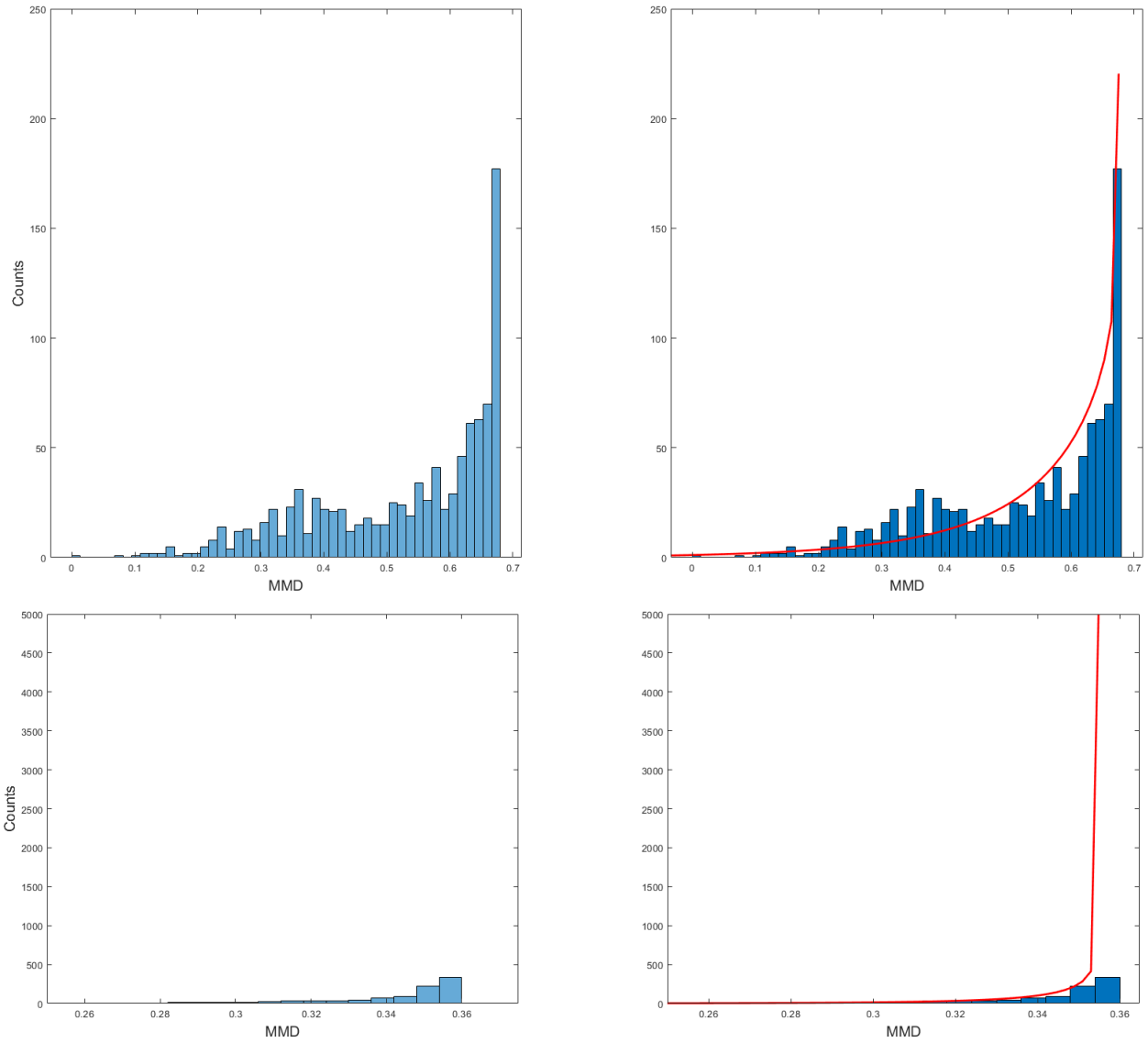

Figure 2.6: Fitting a GEV distribution on the MMD estimations
The fitting is employed between random samples and two different datasets.

$DS_{X'}$ with the same sample size as DS_X. Then a normal distribution is fitted on these computed cSKLs. The samples that were randomly drawn to create these datasets, where samples from $\mathbf{Pool_1}$ that do not have the same disease as X. Similarly we compute the cSKL between DS_Y and 20 randomly formed datasets $DS_{Y'}$, then the distribution of these computations is estimated.
To get the final significance value of $d_{XY} = cSKL(DS_X, DS_Y)$, we estimate the p-value of d_{XY} and DS_X, and the p-value of d_{XY} and DS_Y and the maximum

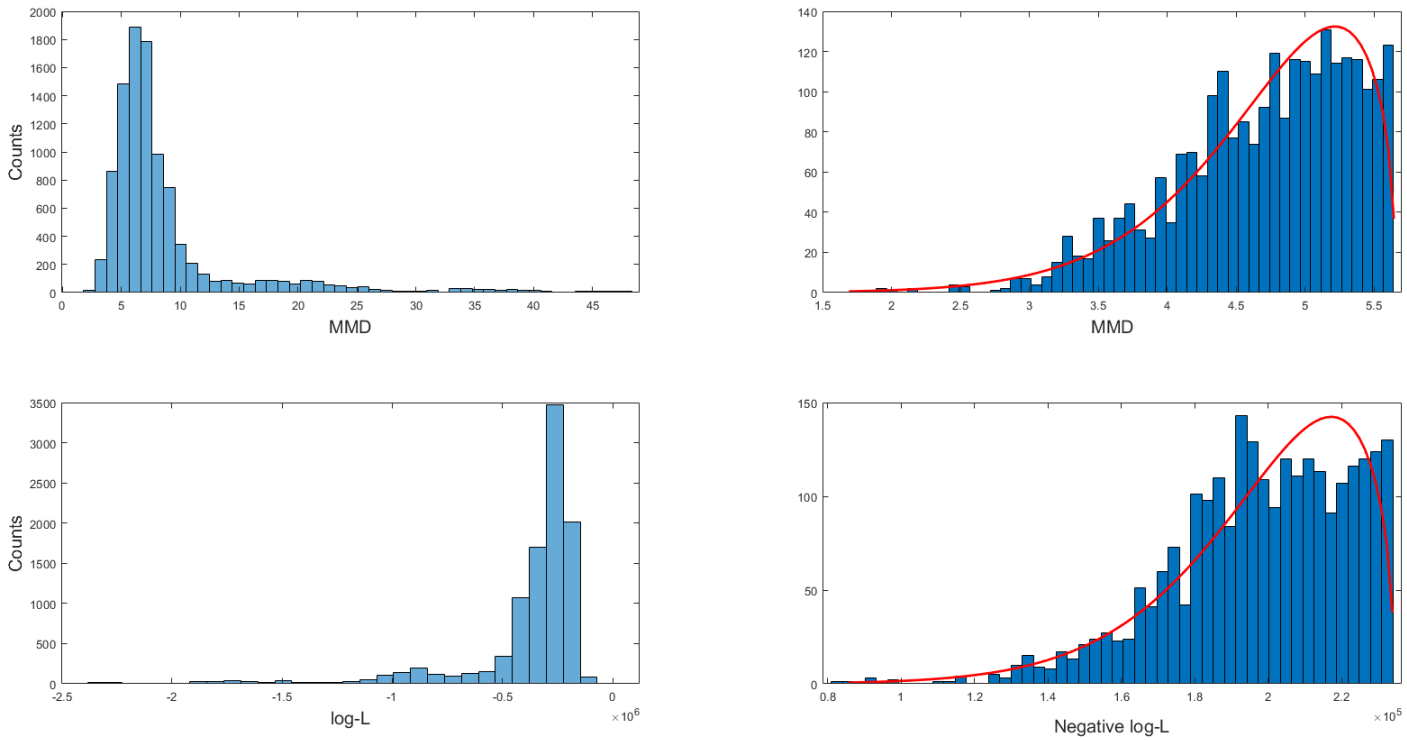

Figure 2.7: Improved distribution fitting.

value between the two is used.
As a result, the similarity between each pair of datasets in $\mathbf{Pool_1}$ is defined by a p-value. All these similarities can be represented in the form of a distance matrix. Each row of this matrix, represents a point (a given dataset) in multidimensional space. Afterwards with the help of the t-SNE algorithm [van der Maaten and Hinton, 2008] which is an algorithm for visualizing multidimensional points in lower dimensions, we can visualize where a specific sample would be placed in the same two dimensional plane as our datasets.

Chapter 3

Results

3.1 Early experiments and data setup

In one of our most notable early experiments we used all the datasets from the initial dataset pool. Let us denote the initial pool of removed samples $\mathbf{S_0}$, and the pool of datasets with the remaining samples as $\mathbf{Pool_0}$.
Using these datasets we tried to categorize each sample from $\mathbf{S_0}$ to a specific disease. To implement this, initially, using our algorithms, we assigned the most similar dataset to each sample, and then labeled (prediction) each sample with the disease that best describes the dataset that was assigned to it. We then compared the predicted label of each sample to the label of the dataset from which, each sample from $\mathbf{S_0}$ originated (The label of each dataset was already provided from Biodataome, which had been automatically extracted using text-mining). While this tactic produced excellent results (we were almost always able to predict a sample to the label of the dataset of origin), this method does not necessarily display how successful we are in finding a given sample's disease. Instead, it portrays, how successful we are in finding a given sample's original dataset.
Figures 3.1 and 3.2 depict how the disease specific datasets would be classified, if the original dataset was not removed from each sample's ranked list. The histograms in figure 3.1 depict probability of rank, without the use of bootstrapping, while figure 3.2 shows how the use of bootstrapping affects the accuracy of the results. While both the algorithms seem to be performing quite well, achieving very small average rank values and excellent accuracy, we suspect that both algorithms are managing to correctly identify a sample's original dataset, instead of correctly identifying a sample's disease. In any case, these results are valuable in showing how accurately our frameworks would function as a dataset retrieval system. A query that this system would take as input would be a single sample, and it would return a list of the most similar datasets with regards to that sample.

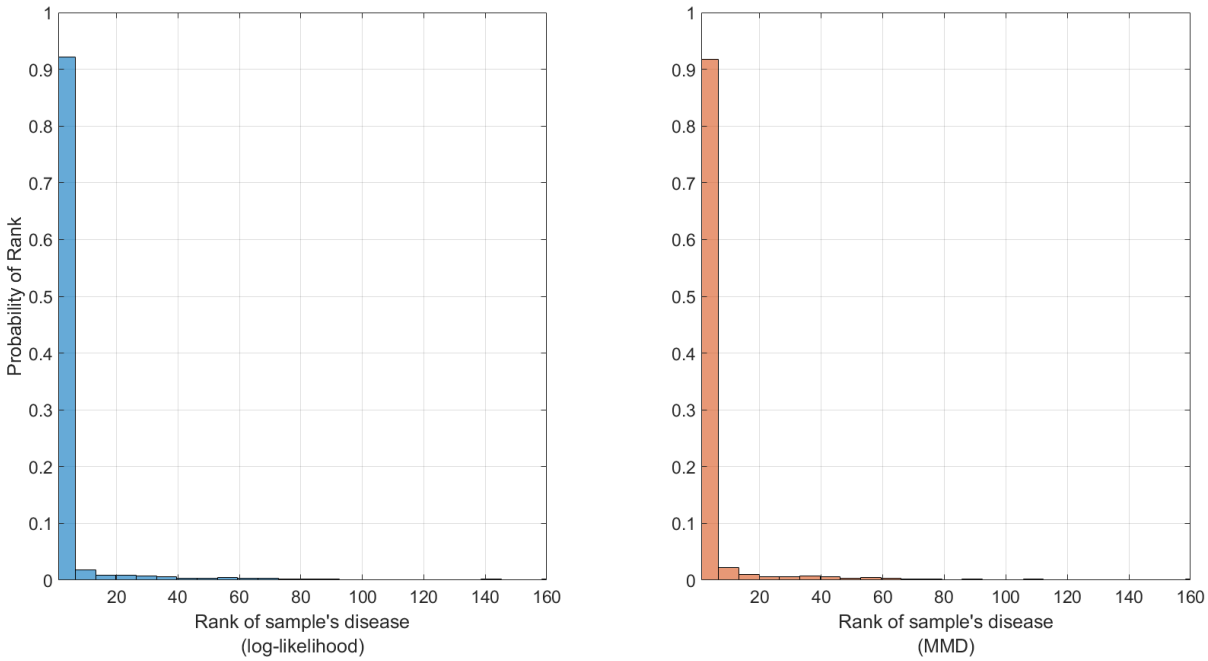

Figure 3.1: Rank probability for the log-likelihood classifier and the MMD classifier

3.2 Main experiments

In our main results, as was described in section 2.1.3, instead of using the dataset of origin for labeling each sample, we decided to manually label them (This has also made it possible to be more specific with the disease of a sample, and to label samples to healthy, when no disease is present, or when the tissue that is sampled is normal). Then, in order to ensure that we have removed the aforementioned dataset bias, we removed from each ranked list that was produced for each sample in $\mathbf{S_i}$, the computations that refer to datasets from $Pool_i$ that share the same dataset of origin as the sample in question.

3.2.1 Comparing results of the two frameworks

In order to produce accurate estimations, we applied our frameworks multiple times, on the curated dataset pool, while bias removal was employed as well (described in section 2.1.3). As described in figure 3.3, each curated dataset is randomly seperated. 90% of each dataset ends up in $\mathbf{Pool_i}$ and 10% in $\mathbf{S_i}$. Furthermore, if a dataset contains less than 20 samples, all it's samples will be placed in $\mathbf{S_i}$. The precomputation phase and the bootstrapping phase are employed on the datasets in $\mathbf{Pool_i}$. Then, in the main phase all the distances and statistical significances between datasets in $\mathbf{Pool_i}$ and samples in $\mathbf{S_i}$ are computed, while bias removal is also employed in the current step. Afterwards, in the next step, each

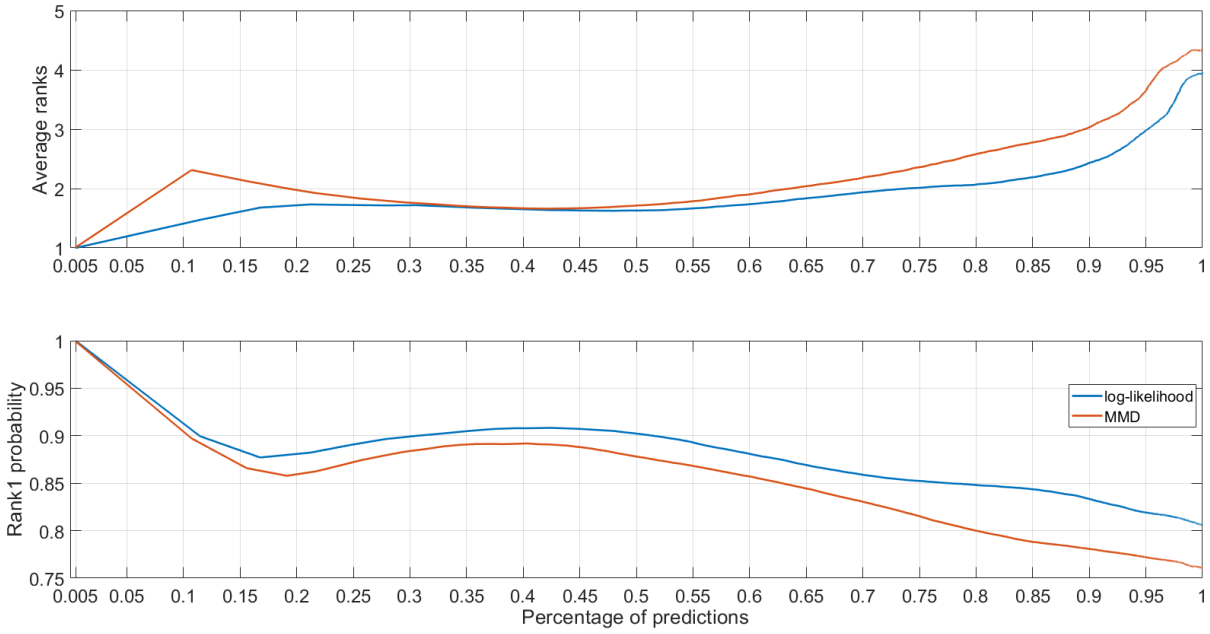

Figure 3.2: Average ranks and $Rank_1$ probability for various percentages of predictions

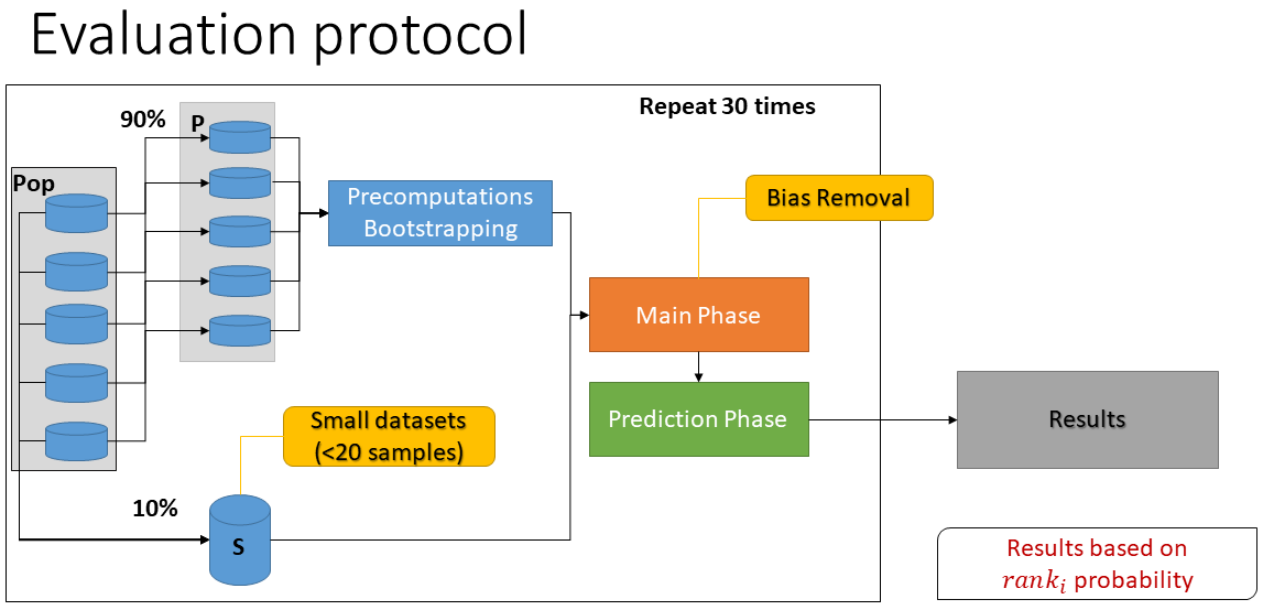

Figure 3.3: Evaluation protocol for main experiments

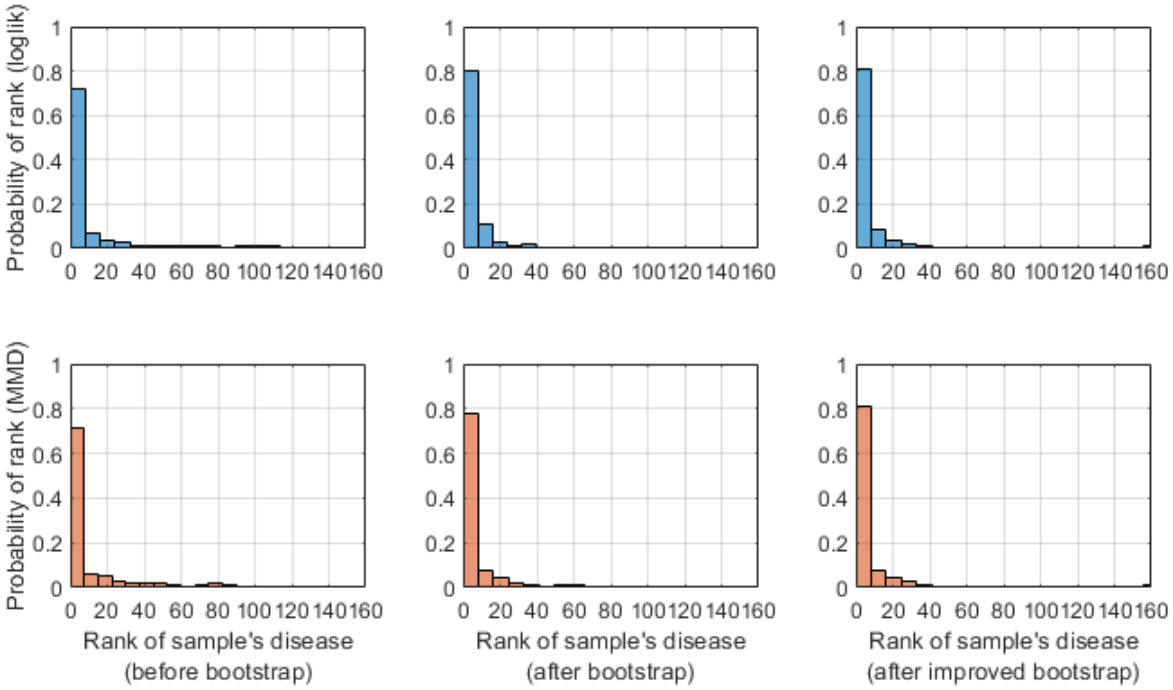

Figure 3.4: Rank probability histograms before and after bootstrapping. Before bootstrapping (left), after bootstrapping (middle), after improved bootstrapping (right).

sample from $\mathbf{S_i}$ is categorized to a specific disease, using the distances computed in the previous step. This process is repeated 30 times, and finally the average results are presented.

Figures 3.4, 3.5 and 3.6 depict how the overall performance of both algorithms can be boosted when an appropriate bootstrapping phase is employed. Figure 3.4 shows how probable the appearance of each rank would be, even when there is no use of bootstrapping, for both frameworks. The top row depicts the histograms before bootstrapping, after initial bootstrapping and after the improved bootstrapping (where the fitting is employed only on the 25% most significant part of the computations) for the log-likelihood metric while the bottom row depicts the same histograms for the MMD metric. As it can be observed, the use of bootstrapping has increased the probability of smaller ranks for both frameworks. The results have become more accurate and the sorted lists more accurately depict the similarity between samples and datasets. Furthermore, the improved bootstrapping slightly improves the results when compared to the initial bootstrapping, even when our frameworks make all the predictions. Figures 3.5 and 3.6, compare the initial boostrapping and the improved bootstrapping for the MMD metric and the log-likelihood metric respectively, in terms of accuracy ($rank_1$ probability), average ranks and percentage of predictions for various strictness levels. The first

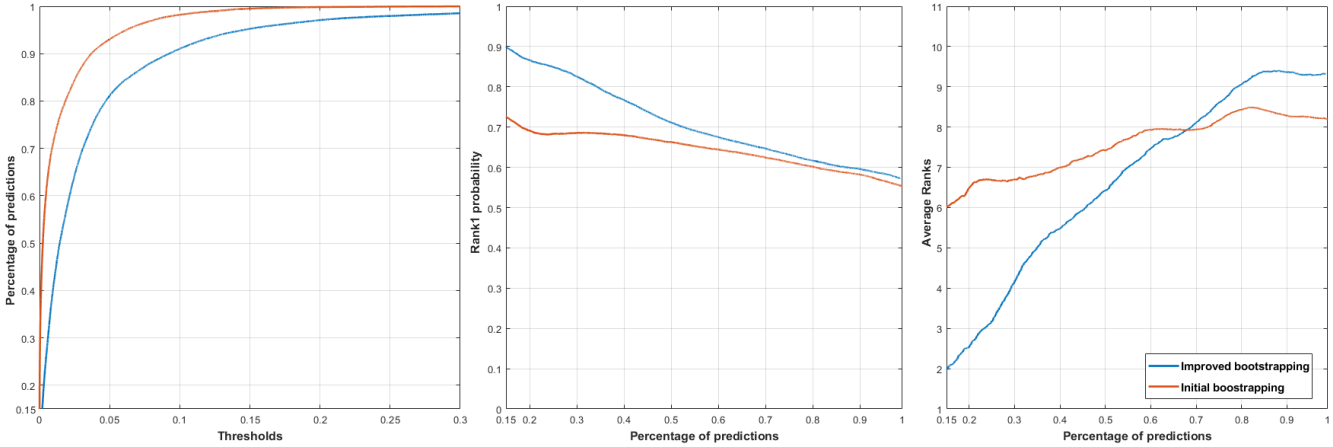

Figure 3.5: Initial vs Improved boostrapping (log-likelihood).

plot in each of the two figures depicts the correlation between different thresholds and percentage of predictions for both kinds of bootstrapping (i.e. how many predictions will our frameworks end up making if we decide to not predict the cases where the significance is above a specific threshold), while the next two plots show the accuracy and the average ranks of the cases that end up being predicted for various strictness levels (thresholds) that correspond to specific percentages of predictions. For the MMD framework, in the case of initial bootstrapping versus improved bootstrapping, the accuracy of the improved bootstrapping as well as the average ranks, are superior for every strictness level. In the case of the log-likelihood framework, the $rank_1$ probability of the improved bootstrapping is superior for every strictness level, while the average ranks seem to be somewhat better for the initial bootstrapping when relaxed strictness levels are used, but, in contrast to the improved bootstrapping, the average ranks do not seem to improve as the thresholds become stricter. When the log-likelihood framework makes less than $\sim 65\%$ of the total number of predictions, the average ranks of the two kinds of bootstrapping start to converge, with the average ranks of the improved bootstrapping being exceedingly better than those of the initial bootstrapping.

Figures 3.7, 3.8, 3.9, 3.10 also clearly depict the advantage of the use of bootstrapping. By allowing our frameworks to only make predictions when we are confident enough about the outcome (the smaller a p-value is between a given sample and a dataset, the more statistically confident we are that the sample is similar to a dataset), the predictive performance of the frameworks increases. Figure 3.7 compares the two classifiers when improved bootstrapping is employed, in terms of average ranks and accuracy for various percentages of predictions. Both classifiers have a starting accuracy near 55% where all the predictions are made, and reach more than 90% accuracy with an appropriate strictness level. In contrast, the best possible "dummy" classifier, that predicts everything to the most probable

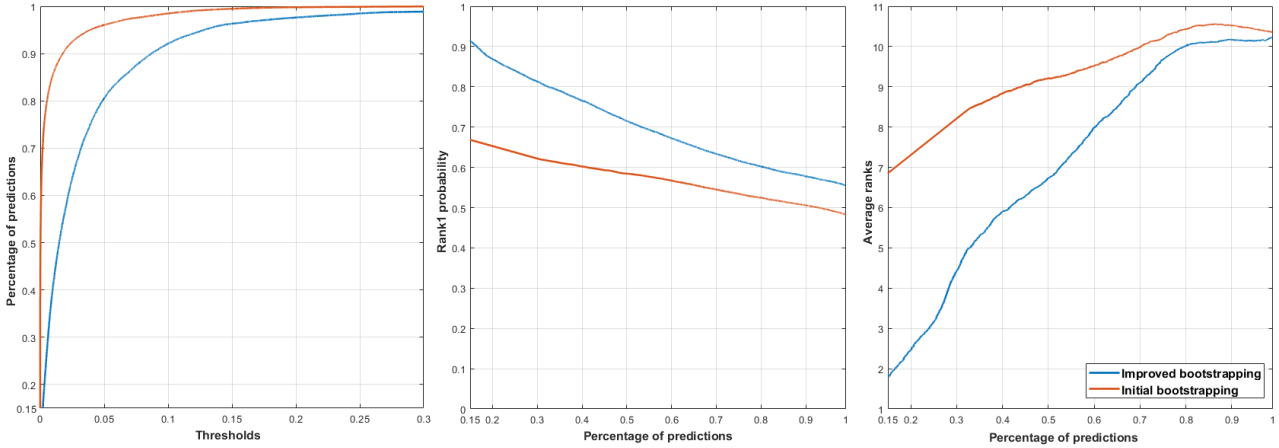

Figure 3.6: Initial vs Improved boostrapping (MMD).

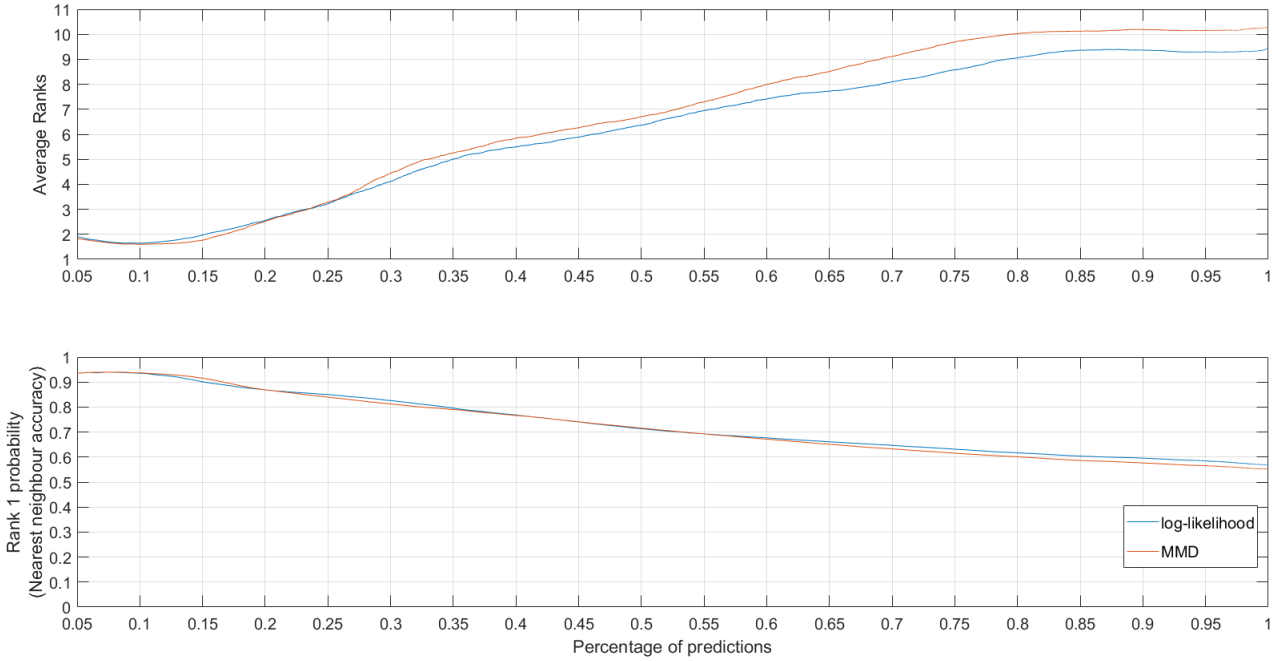

Figure 3.7: Performance comparison between MMD and log-likelihood. Depiction of how the results improve as we make less predictions based on a significance threshold.

disease (breast cancer), would achieve an accuracy of 20.66%. When we allow for don't-know-predictions, both frameworks achieve near 85% $rank_1$ probability

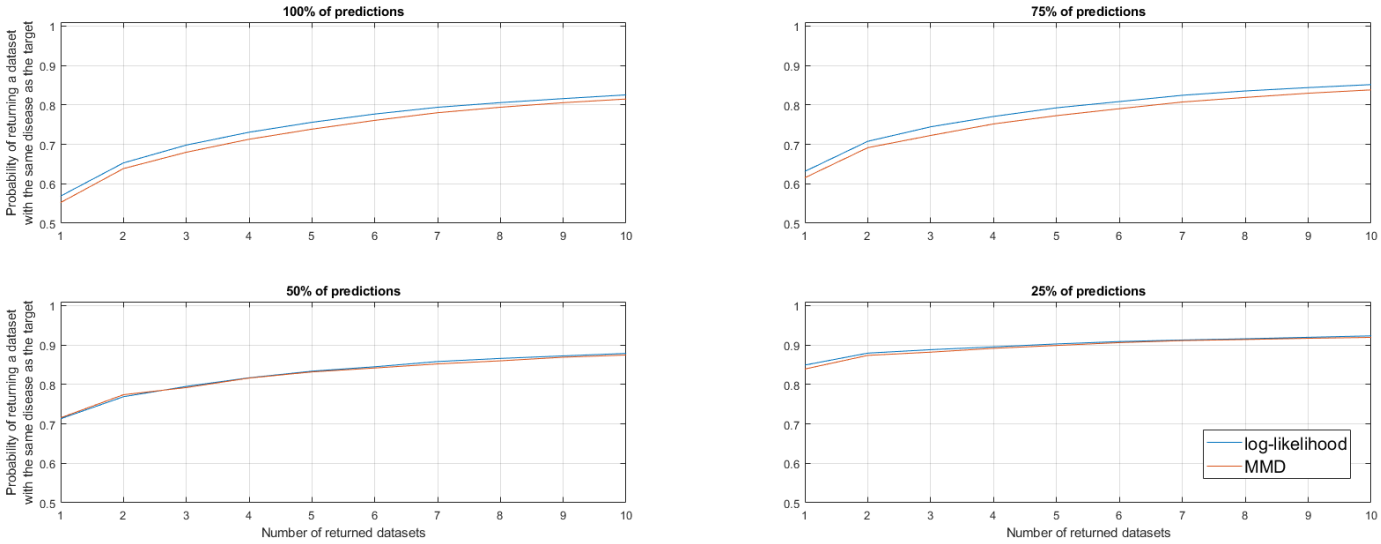

Figure 3.8: Cumulative rank probability for different percentages of predictions.

when approximately 25% of the predictions are made, and 95% $rank_1$ probability when about 10% of the predictions are made. In addition, the two metrics seem to be very comparable in terms of performance, with the log-likelihood classifier, seemingly performing slightly better than the MMD based classifier.

In figure 3.8 we investigate how many datasets need to be returned on average so that at least one dataset from the top-K datasets will be labeled with the target disease for a specific sample. When we make all the predictions there is $\sim 82\%$ chance that the relevant disease will be found in the first 10 datasets, while when we make 25% of the predictions, there is $\sim 90\%$ chance that the relevant disease will be found in the first 5 returned datasets on average. As shown in Figures 3.9, 3.10, the predictions that were discarded based on significance thresholds are mostly erroneous predictions. Another point that these figures seem to suggest is that, the more samples we have for a given disease, the more probable it is for samples that actually belong to that disease to be correctly classified.

3.2.2 Don't-know-predictions

For the purpose of investigating whether not making a prediction based on significance thresholds helps identify cases that are not possible to predict, or cases that will not be predicted correctly, we estimated the area under the receiving operator curves (AUC) [Bradley, 1997] for two different settings. In the first setting, we set as case 0 the samples that can be labeled (i.e. their true disease exists in at least two datasets in $Pool_i$), and case 1 the samples whose disease has not been previously observed (i.e. these samples are not possible to be correctly classified). As for the second setting, we set as case 0 the samples that have been

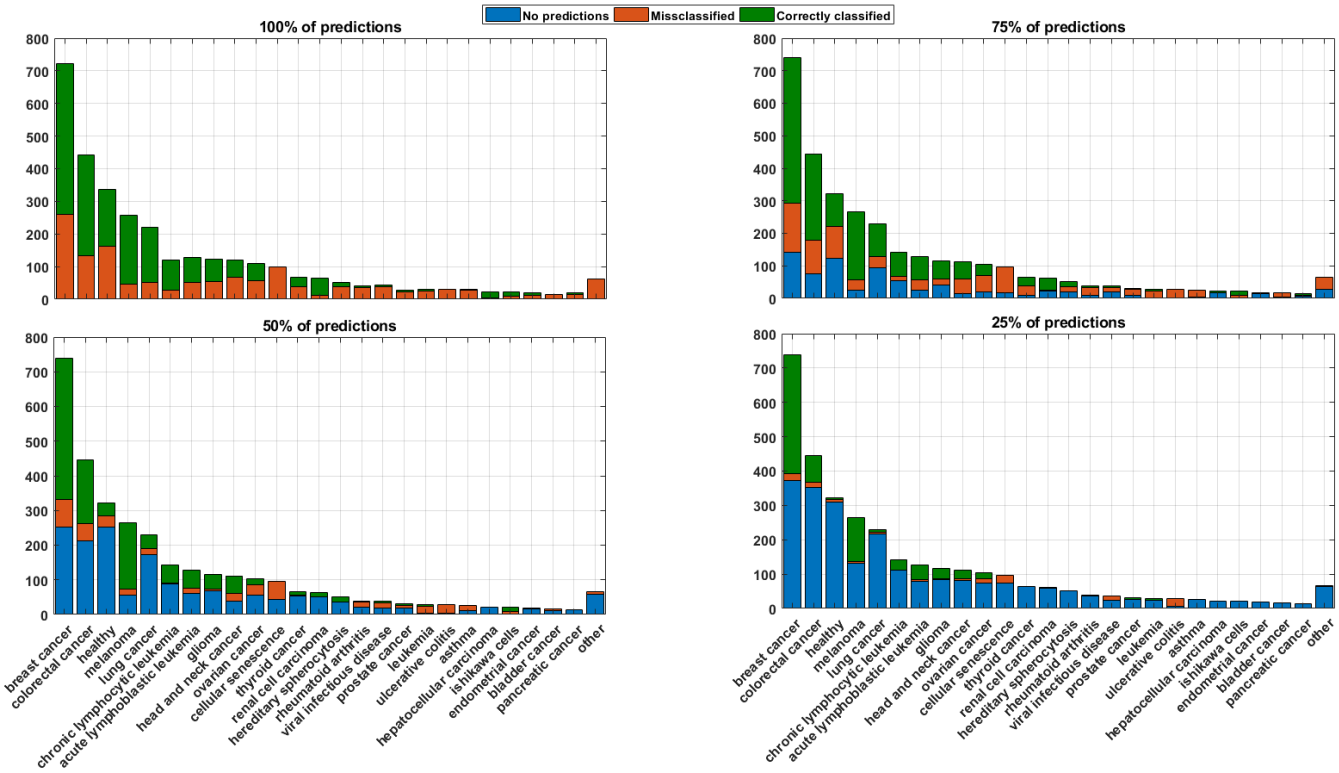

Figure 3.9: Prediction outcomes per disease for the log-likelihood classifier.

correctly labeled by the classifier, and as case 1 the samples that either have been incorrectly labeled, or that are not possible to predict. We then estimated the sensitivity and 1-specificity on both of these settings for various thresholds, and plotted the resulting curves. The reason we used AUC instead of accuracy for these experiments, is that in each of the two settings the two classes are unbalanced, and the AUC outcomes are not biased by unbalanced data, in contrast to the classic accuracy. Figure 3.11 depicts the resulting curves for the two settings (top: setting 1, bottom: setting 2). On the first setting the log-likelihood classifier achieves AUC = 0.661 and the MMD classifier achieves AUC = 0.653, while on the second setting the log-likelihood classifier achieves AUC = 0.716, while the MMD classifier achieves AUC = 0.723. The first curves, being above the diagonal, mean that we are truly able to identify to some extent cases that are impossible to predict (samples whose disease has not been previously seen) with both our frameworks. In addition, the fact that the curves of the second setting exceed those from the first setting, implies that through bootstrapping we are also able to predict cases where our frameworks' predictions will possibly be incorrect (cases where our classifiers will fail to find the correct label of a sample). Figures 3.12 and 3.13, are confusion-matrix like tables from the results of the log-likelihood classifier, which depict the average probability of predicting a sample of a specific

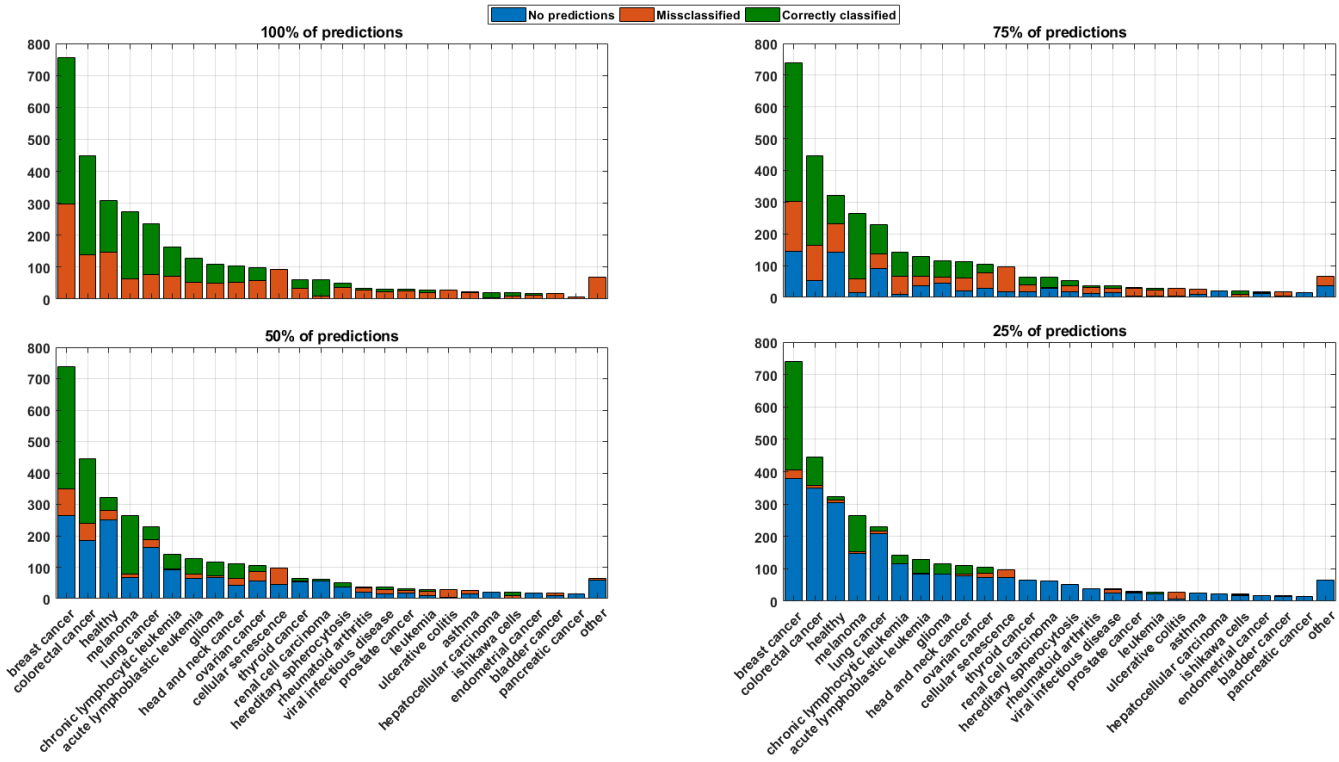

Figure 3.10: Prediction outcomes per disease for the MMD classifier.

disease to a specific outcome, when all the predictions are made, and when we make predictions on approximately 25% of the samples respectively (25% of the predictions corresponds to a pvalue thresholds of **0.0046**). The cells that fall on the diagonal which is signified by a white line, are the cases where correct predictions are made. The rows that exist below the diagonal refer to diseases which are present during the classification phase, but have not been previously seen by our classifier. In an ideal scenario, every cell on the main diagonal would be as bright as possible, having a value of '1', meaning that every prediction has been a correct one. While our case is not an ideal scenario, by comparing the two figures we can observe, that many of the incorrect predictions ended up being insignificant. Figure 3.13, can be used to examine interesting "difficult" cases, from the predictions that remained significant. For example, when 25% of the predictions where made, every myelodysplasia sample was predicted as acute lymphoblastic leukemia, most of the samples in the endometrial cancer class where predicted as breast cancer, every pilocytic astrocytoma sample was predicted as glioma, etc. Furthermore, in the case of the "healthy" samples, we can observe that the accuracy dropped when the most significant predictions were kept. While we speculate that many of these errors occur due to insufficient sample size on our datasets, there also exist samples that have been treated which probably affects their gene-expression significantly.

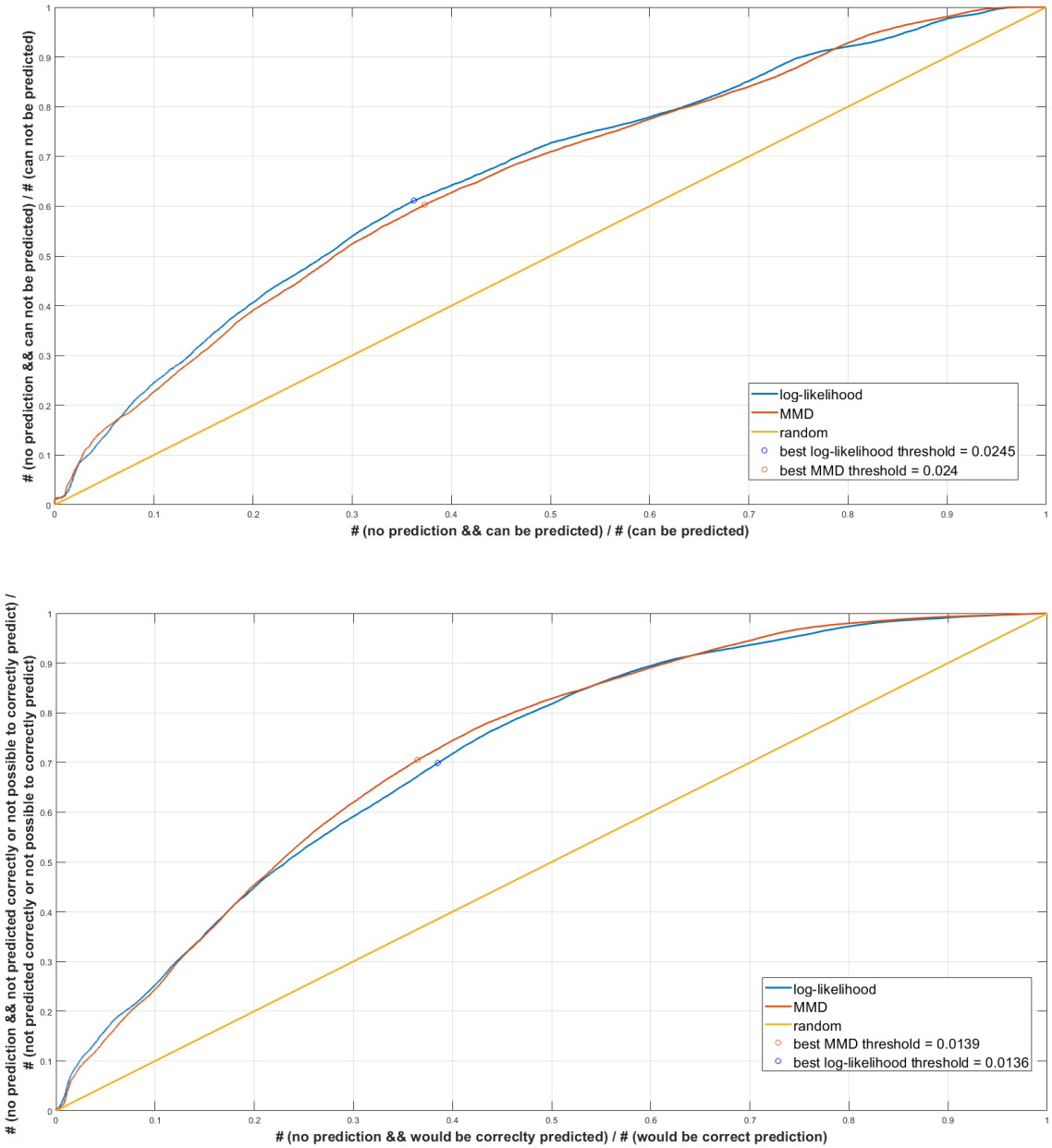

Figure 3.11: AUC: Predicting incorrect predictions

Finally, it is highly possible that some pathologies are not differentiable through gene-expression analysis.

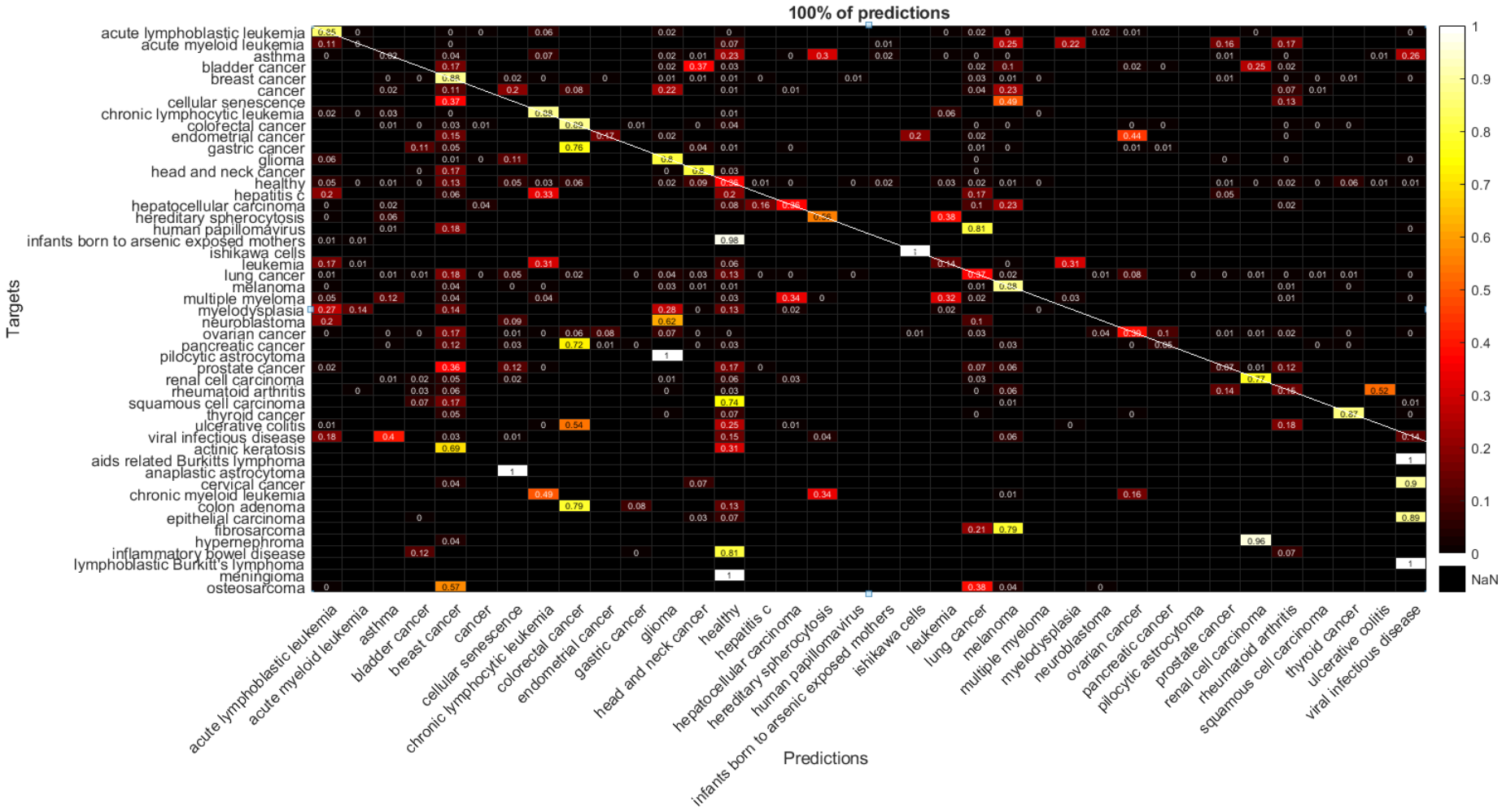

Figure 3.12: Probability of each possible outcome without using a significance threshold (log-lik).

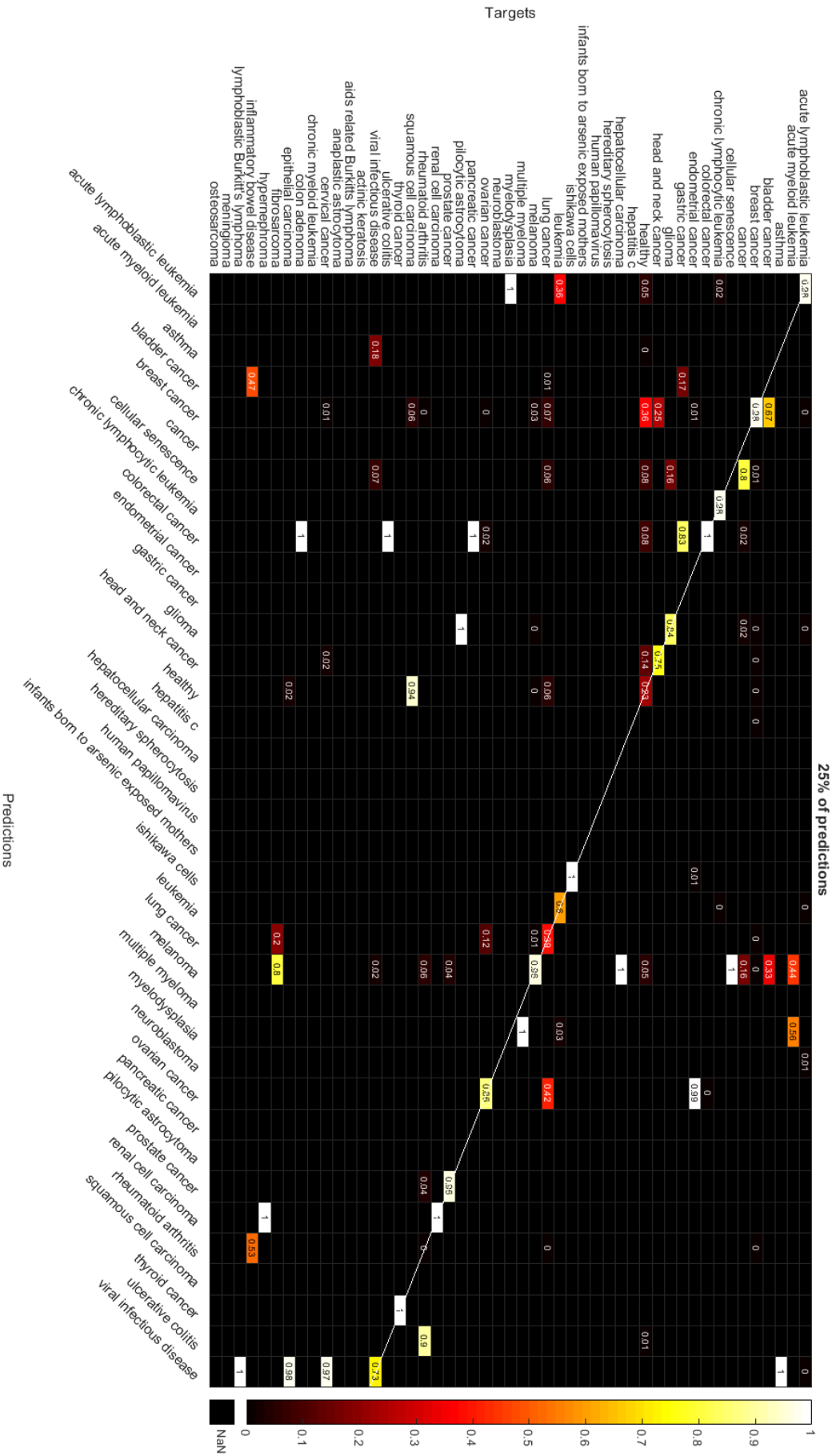

Figure 3.13: Probability of each possible outcome on 25% of the predictions (log-lik).

3.2.3 Time-space comparison

In figure 3.14 it can be observed that the log-likelihood classifier is somewhat faster than the MMD classifier in every phase of the frameworks. The precomputation phase times as well as the bootstrap phase times, depend on the number of datasets that are available (number of datasets in $Pool_i$). The bootstrap phase times also depend on the number of samples that are randomly selected for each bootstrap. The main phase times depend on both the number of datasets that are available, as well as the number of samples that are going to be classified. On average the log-likelihood classifier needs **0.81 seconds** to load a dataset and make precomputations, while, the same task takes **1.1 seconds** for the MMD classifier. Approximately **1010 seconds** are needed for the log-likelihood classifier to randomly sample 10,000 samples, 166 times (once for each different dataset in $\mathbf{Pool_i}$), compute the log-likelihood distances and estimate the distance distribution between each dataset and the random samples, while it takes near **1090 seconds** for the MMD classifier to do the same. Finally, loading all the **3735 test samples** (all the samples that ended up in S_i), the precomputations of each dataset, computing the sample-dataset distance significances and classifying each sample takes about **345 seconds** for the bootstrap log-likelihood classifier, while it takes approximately **420 seconds** for the bootstrap MMD classifier (note that, we made a memory efficient implementation, such that, no more than one dataset is loaded in memory at any given time. This has added some overhead to the experiment times reported above, due to disk reading and writing). What is more important is that the log-likelihood classifier can achieve results of the same value or even better than the MMD classifier, but through the use of PCA, only uses 10% of the original dataset size. Meaning that for each dataset the information needed on average to produce our results, is contained on the first 10% most significant principal components and eigen values.

3.3 Visualizing datasets and samples in reduced space

With the help of t-SNE we can further visually inspect and interpret how a sample is classified and how the distances between samples and datasets are used in assigning a sample to a disease. To achieve accurate visualization, we constructed a dataset-to-dataset distance matrix, containing the distances of all the pairs of datasets from P1, using the cSKL algorithm (see section 2.3). Since final distances of our framework are expressed in the form of p-values, we designed a bootstrap step for the cSKL as well, so that we can express the cSKL distances in a similar manner. Additionally, we added to this matrix sample-to-dataset distances for a specific sample. This matrix was then passed through Matlab's t-SNE implementation, with the following settings: Algorithm: exact (the default: 'barneshut' would provide a faster but approximate solution), Distance: minkowski (visually returned better 2d embeddings), NumDimensions: 2 (returns values for 2d representation), Exaggeration: 2 (returned smaller loss than the default which

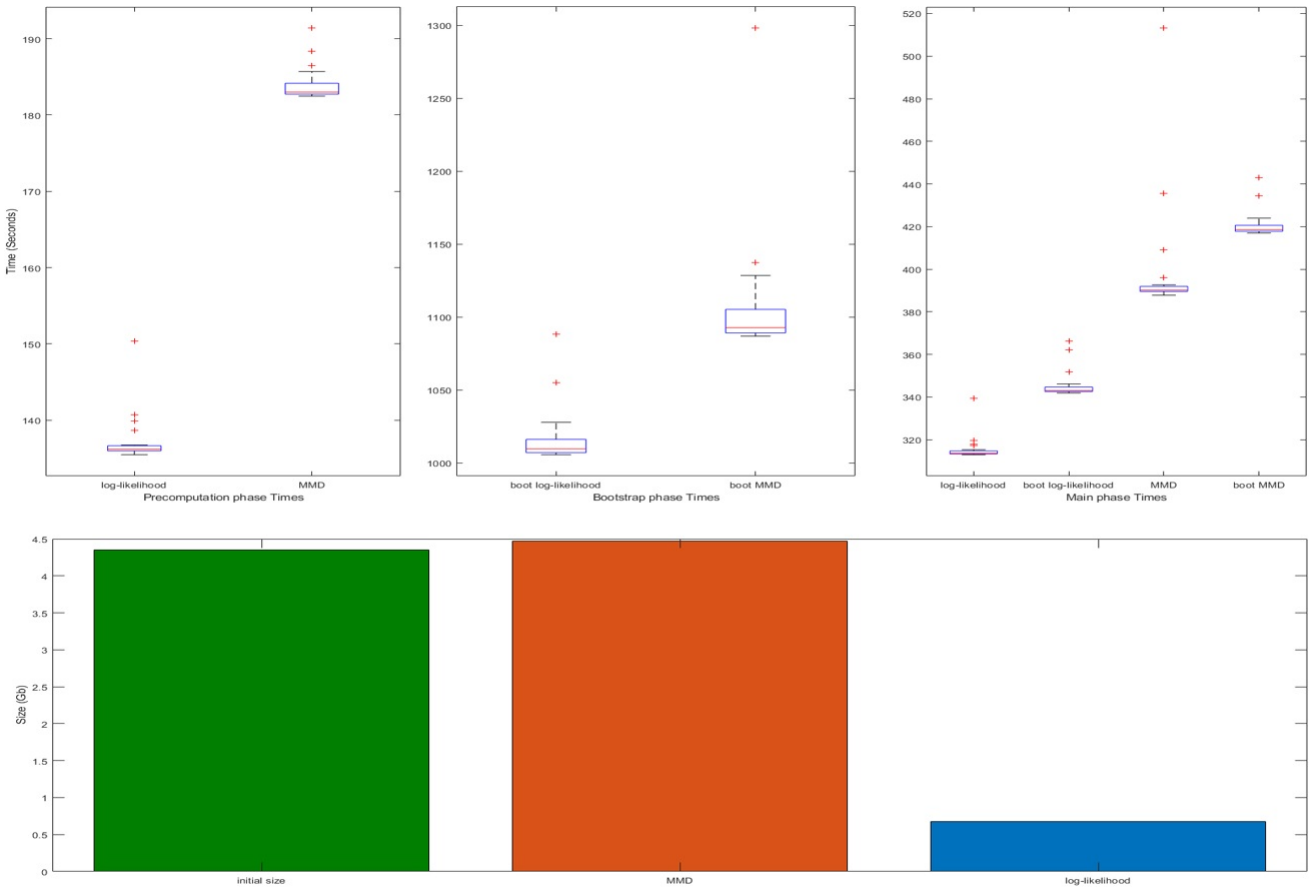

Figure 3.14: Computational time and storage capacity needed comparison of the two classifiers

is 4), LearnRate: 200 (this learning rate helped tSNE to converge to smaller loss). Loss of the representation is defined by the Kullback-Leibler divergence between modeled input and output distributions, returned as a nonnegative scalar (the smaller the better). Furthermore, since the t-SNE algorithm, starts with a random initialization and iteratively converges to a local minima of the loss, for each produced figure, we run the algorithm with the above settings 20 times, and kept the embeddings of the iteration that produced the smallest loss (i.e. the iteration with the most accurate visualization). Some interesting cases of the resulting scatter plots can be found in figures 3.15, 3.16, 3.17, 3.18. In these figures, each dot represents a dataset with a specific disease, while the black star represents a specific sample.

Figure 3.15, is an example of a breast cancer sample which got placed near breast cancer datasets. The classification label was derived from the label of the nearest dataset to the sample which was a breast cancer label. Figure 3.16, depicts

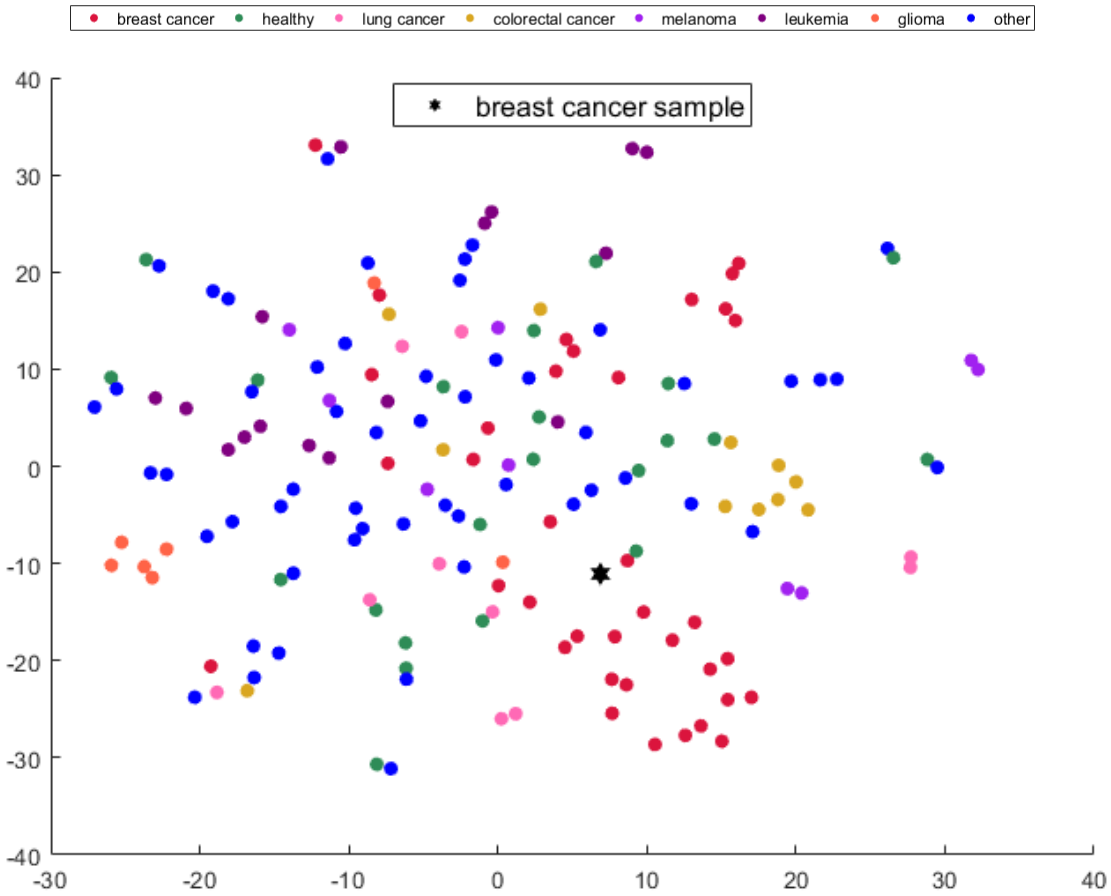

Figure 3.15: Placing a breast cancer sample in the same space as the datasets that were used.

the placement of a colorectal cancer sample in the same two-dimensional space as the datasets that where used. In this example the colorectal cancer sample got placed near colorectal cancer datatasets and as a result, it was correctly labeled with a colorectal cancer label. Figure 3.17, is a case with a glioma sample which got placed closer to glioma datasets, than other kind of datasets.

Finally, figure 3.18 is a case were a missclassification happened. In this case a lung cancer sample was placed near lung cancer datasets. However, it also got placed near a breast cancer dataset. The classification label was derived from the label of the nearest dataset to the sample which in this case was the breast cancer dataset, and as a result the sample received incorrect labeling. This figure shows that using the nearest-K datasets for the classification of each sample, instead of just using the nearest dataset, would probably improve the results. The difficulty

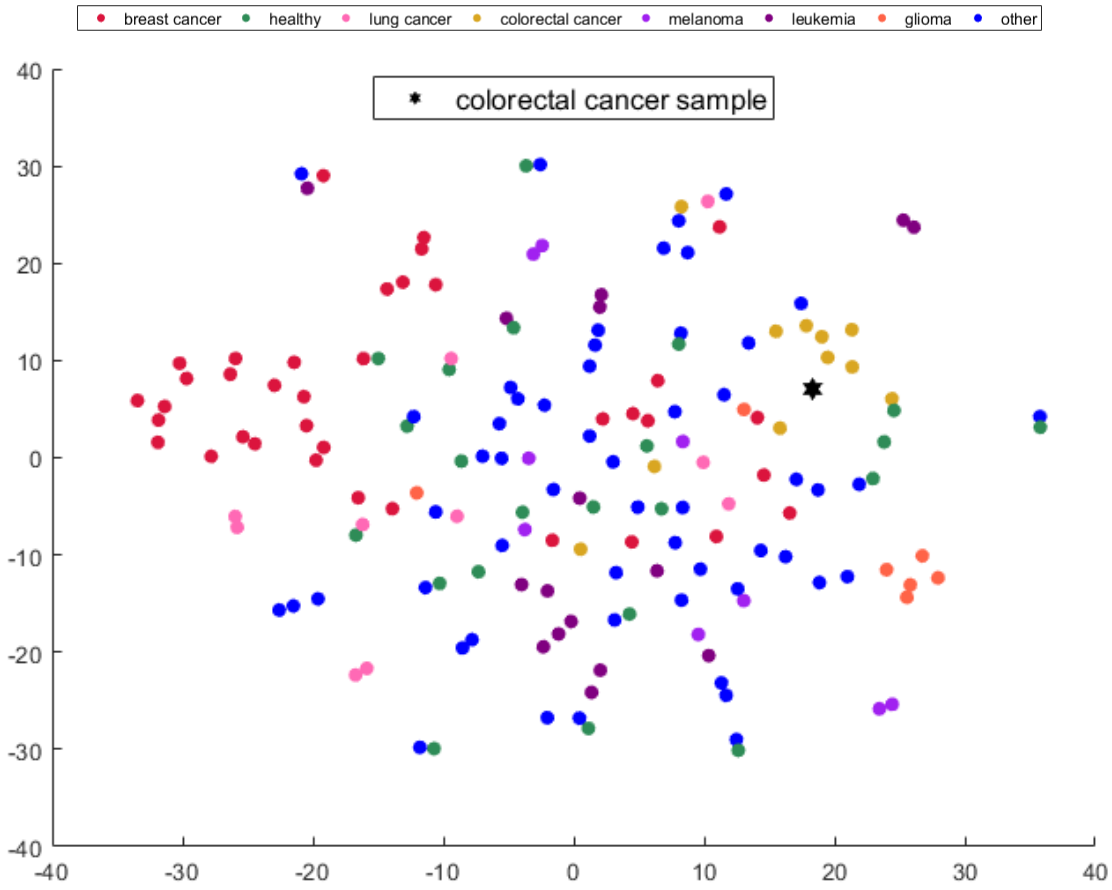

Figure 3.16: Placing a colorectal cancer sample in the same space as the datasets that were used.

with using this method is that the number of datasets per disease is very unbalanced. For example there are 46 "healthy" datasets, while there are just 2 multiple myeloma datasets.
Another striking observation from the tSNE figures, is that datasets of the same disease seem to become clustered together. This corroborates to some extent, the results of [Lakiotaki et al., 2019].

3.4 Same tissue experiments

Since the samples in some of the datasets that share the same disease, also share same types of tissues, one could argue that our algorithms are managing to identify correct diseases because of their ability to identify the tissue that was sampled, instead of truly identifying the characteristics of each disease. In order

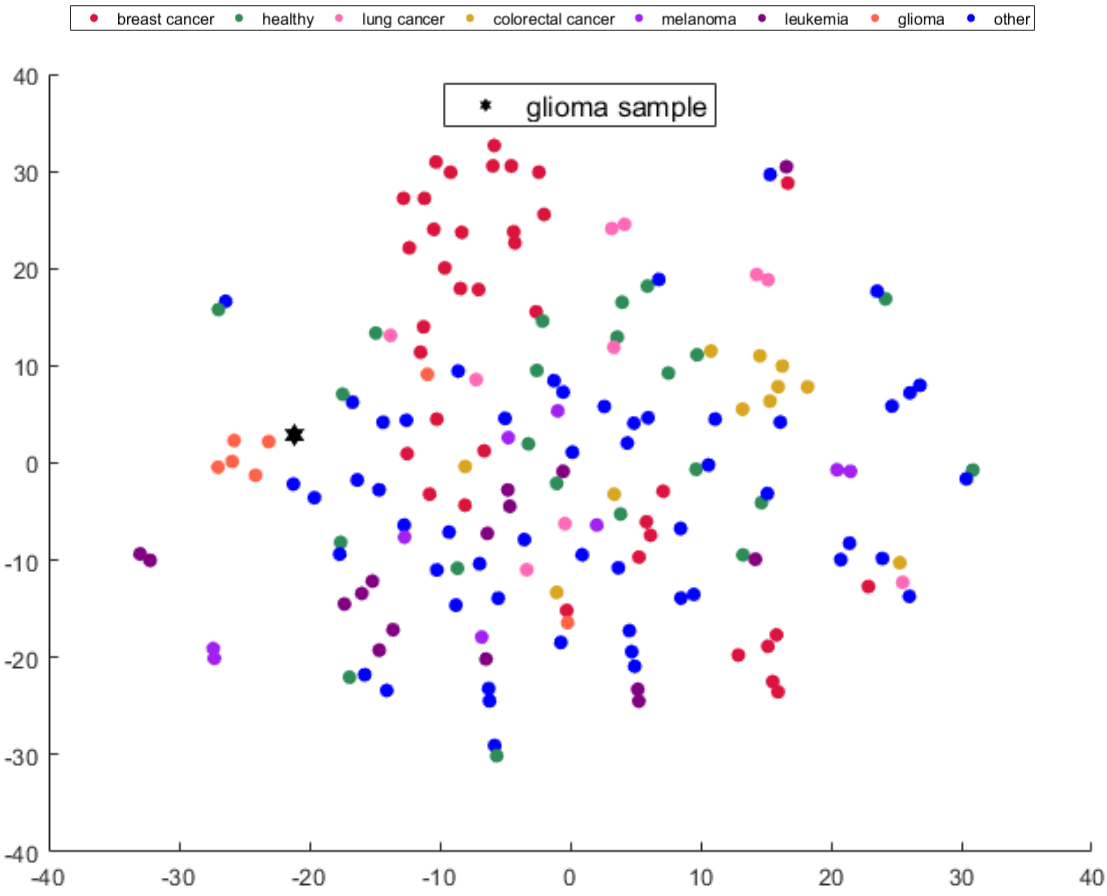

Figure 3.17: Placing a glioma sample in the same space as the datasets that were used.

to test this assumption, we specifically selected some datasets from Biodataome which contained samples sharing same tissues and different pathologies. We then employed **leave-one-out cross-validation** on each dataset separately, to examine how well our algorithms are managing to differentiate samples of different pathologies on identical tissues (figure 3.19). For the purpose of this experiment two datasets were selected. The first dataset (**GSE15061**) contains **870** bone marrow samples, spanning three different outcomes; Myelodysplastic syndrome, Acute myeloid leukemia, "None-of-the-targets" containing **404**, **328** and **138** samples respectively. The second dataset (**GSE68848**) contains **471** brain tissue samples spanning four different outcomes; Astrocytoma, Oligodendroglioma, Glioblastoma and Non-tumor containing **148**, **67**, **228** and **28** samples respectively. From what is observed, our algorithms manage to differentiate the possible outcomes in both cases. While, in the second case, the starting accuracy is not optimal, when we

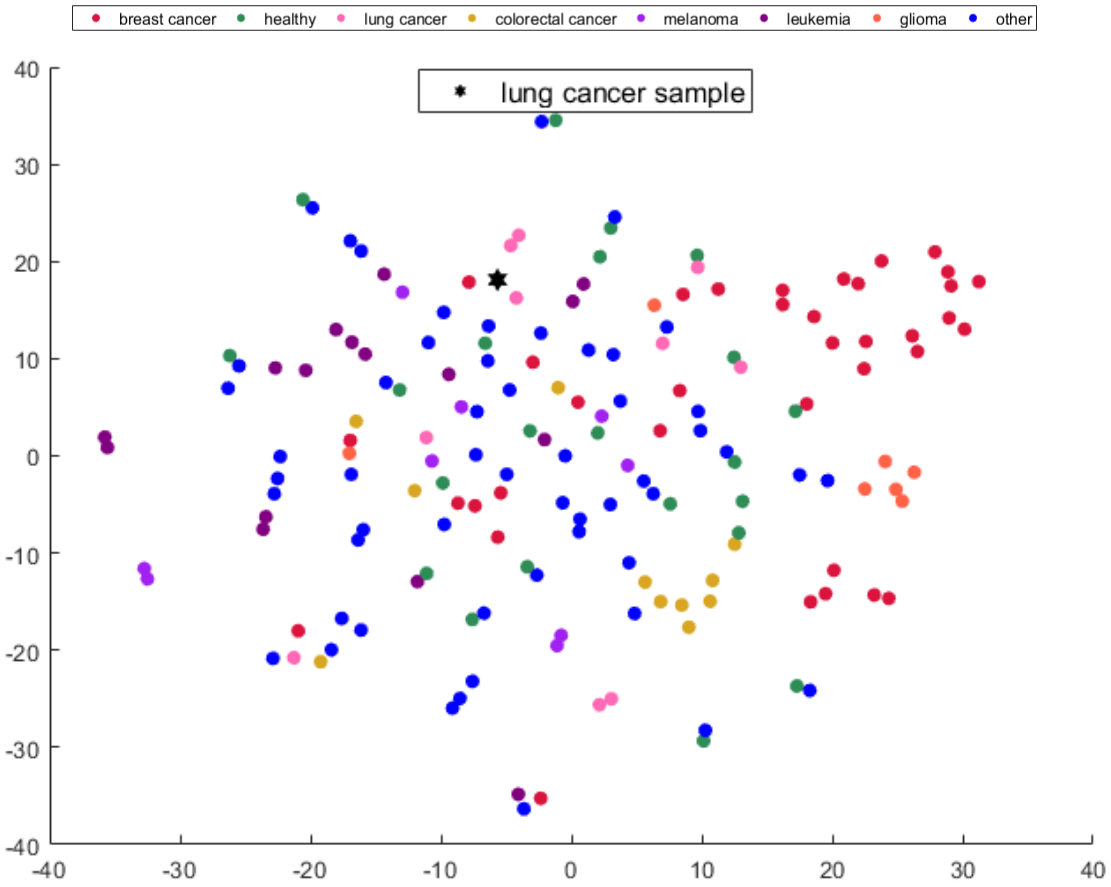

Figure 3.18: Placing a lung cancer sample in the same space as the datasets that were used.

allow for no-prediction cases to occur, the accuracy for both algorithms improves. These experiments suggest that even if there exists bias originating from the tissue of each sample, our algorithms still manage to identify disease characteristics and differentiate between similar diseases of the same tissues.

3.5 Ensembling

An interesting observation occurred in the results of the two frameworks. Both frameworks manage to find the correct disease mostly on the same samples, and mostly disagree in the results in cases where misclassifications happen. To test if the above observation was truly occurring, we employed a classification rule, to only make a prediction for a sample when both frameworks agree on the label.

In order to create a significance value for the results of the ensemble classifier,

we normalized the MMD based p-values and the log-likelihood based p-values so that the smallest p-value is 0 and the largest is 1. This results in a p-value pair for each separate result, so for each such result we take the maximum p-value as the p-value of that computation. As observed in figure 3.20 the results seem to be on par with what was previously observed from the log-likelihood classifier, no improvement seemed to occur. Figure 3.21 depicts the average probability of finding a dataset with the target disease in the **first-K** results of each ranked list, for different percentages of predictions. When $\sim 75\%$ of the predictions are made, we observe that the log-likelihood classifier performs slightly better, with the MMD classifier and the ensembling classifier being seemingly on par and below the curve of the log-likelihood classifier. When $\sim 75\%$ of the predictions are made, there is approximately a 85% probability that the target disease will appear in the **top-10** most simillar datasets, with the log-likelihood classifier, while for $\sim 25\%$ of the predictions there is $\sim 92.5\%$ probability that the target disease will appear in the **top-10** datasets for that classifier. For $\sim 50\%$ and $\sim 25\%$ of the predictions, the MMD and the log-likelihood classifiers seem to be on par, whereas the ensembling classifier being always below the curves of the aforementioned classifiers.

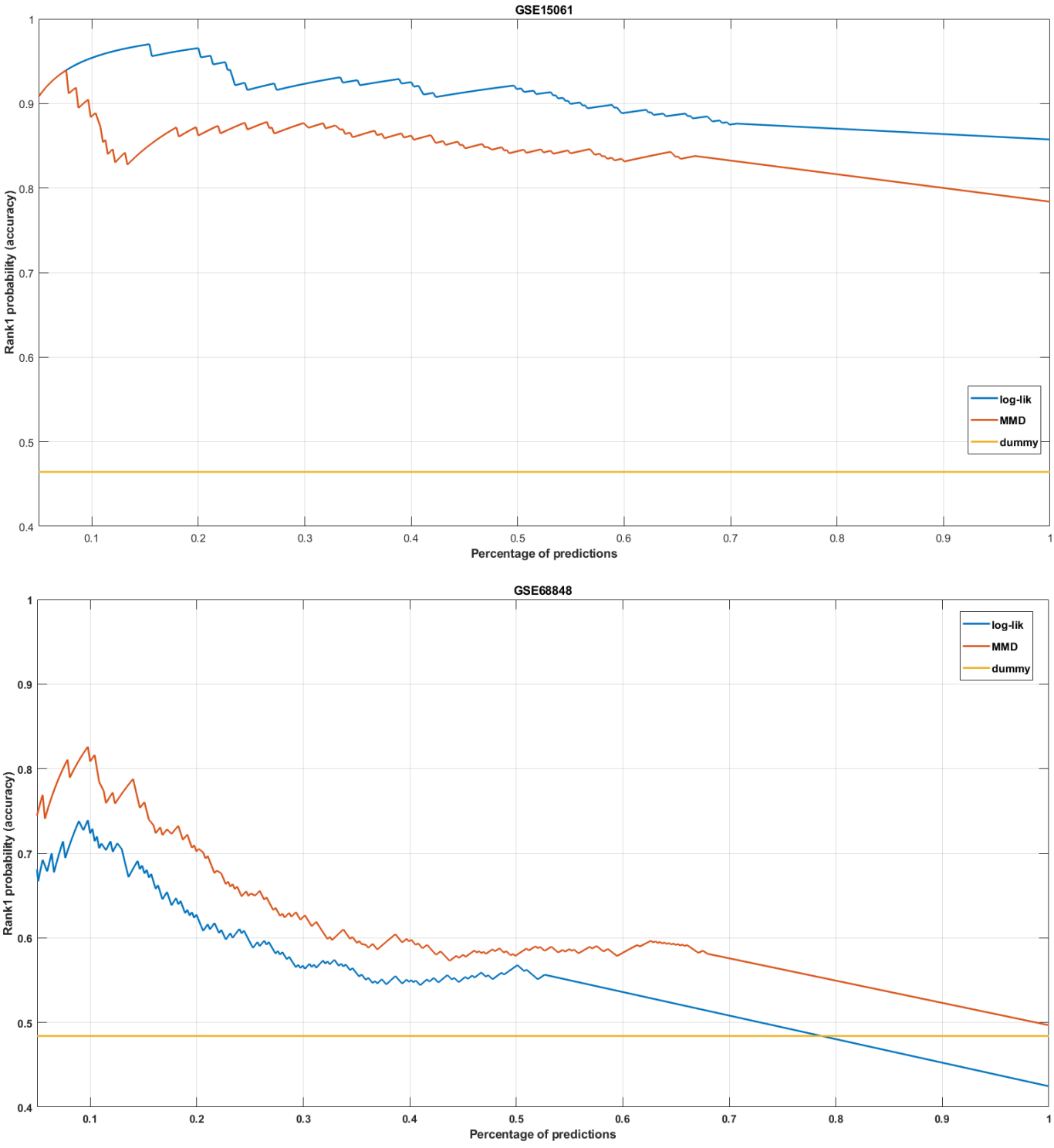

Figure 3.19: Same tissue classification

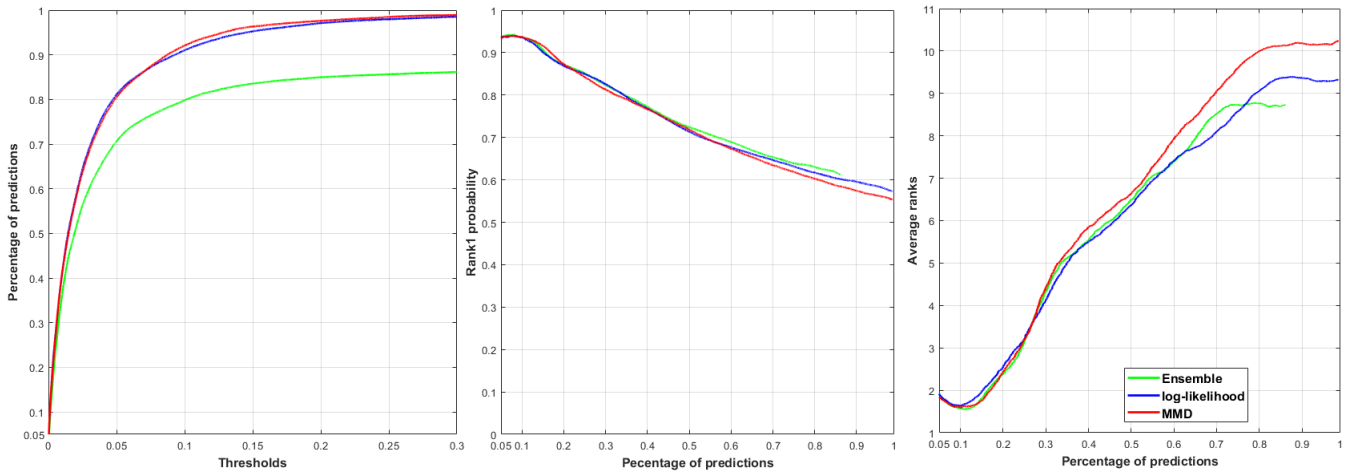

Figure 3.20: Making a prediction only when the two frameworks agree on the result.

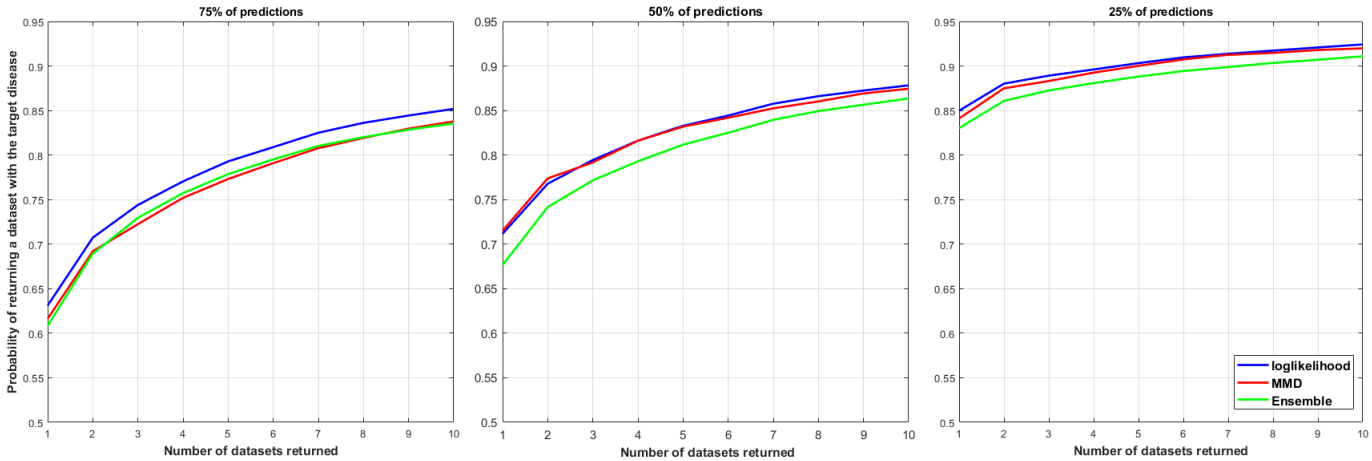

Figure 3.21: Comparison of all frameworks on number of returned datasets. Average probability of finding the target disease in the first-K returned datasets.

Chapter 4

Discussion

The current work is the first, in the whole scientific community, in which gene-expression data are modeled in such a way, that **any** disease **could** be identified in a very close to real world scenario, without having to make assumptions about the number of possible outcomes. Not having to make such assumptions results in easily salable models on new datasets and diseases. New data and diseases can easily be included; One must have stored the necessary precomputations and apply the bootstrap step only on the new data. In contrast, previous works have to make an assumption on the number of possible outcomes and hence they can not be employed in real world scenaria, where their assumption does not hold. Moreover, previous classifiers modeling such problems would have to be retrained from scratch on all the data, in order for new datasets and diseases to be added. We observe, based on our results, that while the problem of identifying **any** given sample's disease from gene expression data seems to be very difficult to solve given our available datasets (very unbalanced, drug treated, very high-dimensional, etc), our models manage to adequately tackle it when cases with don't-know-predictions are allowed, reaching near 72% overall accuracy when about 50% of the predictions are made, and an expected rank value of 5 when 35% of the predictions are made. Furthermore, our scrutinous evaluating procedures (we removed from the possible results of each sample one possible correct outcome (bias removal sec 2.1.3)), result in having negative bias in our performance estimation (i.e. underestimation), meaning that the true performance of our algorithms is probably exceeding what is reported in this work. While the results of both the log-likelihood and the MMD classifiers are similar in terms of performance, the log-likelihood classifier is computationally more efficient than the MMD classifier. The log-likelihood classifier requires approximately 1/9 of the memory space required by MMD to achieve results of the same value. There is a range of **possible uses** for the current work. For example, by returning a list of the top-k most similar diseases for a given sample, this work could be used clinically as a recommendation system, for finding a given patient's most probable diseases in mere milliseconds. Another clinical use of this work would be as a quality control tool; The models from this work could

be used in order to identify whether, a sample of a certain pathology resembles to other samples of the same pathology, or if the sample seems more like an outlier. Since our models are in essence **dataset classifiers**, the framework proposed in this work could also be employed as a fast and lightweight dataset retrieval system (sec. 3.1). The query that it could take as an input would be a single sample (i.e. a vector), and it would return the most relevant datasets with regards to that sample. Furthermore, though our visualization techniques, the output for each possible use of our classifiers can be visually interpretable. As previously described, one limitation of our work is that the datasets we used were not disease specific. In order to set up our experiments, we had to split our datasets of mixed diseases, into smaller disease specific datasets. To achieve this, more than 19,000 samples had to be manually evaluated and labeled to a specific disease, which prevented us from using as many datasets as we had originally planned to use in our final experiments. Moreover, due to the fact that the datasets we used where created for specific studies, many of the samples that we had at our disposal where treated with drugs, which probably affects how these samples will ultimately be classified. Finally, as it became obvious from the t-SNE figures, having more datasets per disease, would further improve our results, as it would become relevant to use the nearest-K datasets in order to classify a sample to a disease, instead of just using the nearest-1 dataset.

Bibliography

Hanaa Salem, Gamal Attiya, and Nawal El-Fishawy. Classification of human cancer diseases by gene expression profiles. *Applied Soft Computing*, 50:124 – 134, 2017. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2016.11.026. URL http://www.sciencedirect.com/science/article/pii/S1568494616305956.

M. Bittner, P. Meltzer, Y. Chen, Y. Jiang, E. Seftor, M. Hendrix, M. Radmacher, R. Simon, Z. Yakhini, A. Ben-Dor, N. Sampas, E. Dougherty, E. Wang, F. Marincola, C. Gooden, J. Lueders, A. Glatfelter, P. Pollock, J. Carpten, E. Gillanders, D. Leja, K. Dietrich, C. Beaudry, M. Berens, David S Alberts, V. Sondak, N. Hayward, and J. Trent. Molecular classification of cutaneous malignant melanoma by gene expression profiling. *Nature*, 406(6795):536–540, 8 2000. ISSN 0028-0836. doi: 10.1038/35020115.

Sandrine Dudoit, Jane Fridlyand, and Terence P Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97(457):77–87, 2002. doi: 10.1198/016214502753479248. URL https://doi.org/10.1198/016214502753479248.

Amir Ben-Dor, Laurakay Bruhn, Nir Friedman, Iftach Nachman, Michèl Schummer, and Zohar Yakhini. Tissue classification with gene expression profiles. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, RECOMB '00, pages 54–64, New York, NY, USA, 2000. ACM. ISBN 1-58113-186-0. doi: 10.1145/332306.332328. URL http://doi.acm.org/10.1145/332306.332328.

Md Shahjaman, Nishith Kumar, Md Ahmed, AnjumanAra Begum, S. M. Shahinul Islam, and Md Nurul Mollah. Robust feature selection approach for patient classification using gene expression data. *Bioinformation*, 13:327–332, 10 2017. doi: 10.6026/97320630013327.

Eunjung Lee, Han-Yu Chuang, Jong-Won Kim, Trey Ideker, and Doheon Lee. Inferring pathway activity toward precise disease classification. *PLOS Computational Biology*, 4(11):1–9, 11 2008. doi: 10.1371/journal.pcbi.1000217. URL https://doi.org/10.1371/journal.pcbi.1000217.

Javed Khan, Jun S Wei, Markus Ringnér, Lao H. Saal, Marc Ladanyi, Frank Westermann, Frank Berthold, Ed. Manfred Schwab, Cristina R Antonescu, Carsten

Peterson, and Paul S. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7:673–679, 2001.

Chuanze Kang, Yanhao Huo, Lihui Xin, Baoguang Tian, and Bin Yu. Feature selection and tumor classification for microarray data using relaxed lasso and generalized multi-class support vector machine. *Journal of Theoretical Biology*, 463:77 – 91, 2019. ISSN 0022-5193. doi: https://doi.org/10.1016/j.jtbi.2018.12.010. URL http://www.sciencedirect.com/science/article/pii/S0022519318306003.

Young-suk Lee, Arjun Krishnan, Rose Oughtred, Jennifer Rust, Christie S. Chang, Joseph Ryu, Vessela N. Kristensen, Kara Dolinski, Chandra Theesfeld, and Olga G. Troyanskaya. A computational framework for genome-wide characterization of the human disease landscape. *Cell Systems*, 8, 02 2019. doi: 10.1016/j.cels.2018.12.010.

Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13 (4):18–28, July 1998. ISSN 1541-1672. doi: 10.1109/5254.708428. URL http://dx.doi.org/10.1109/5254.708428.

Carolyn Lipscomb. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88:265–6, 08 2000.

Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, pages 133–136. AAAI Press, 1982. URL http://dl.acm.org/citation.cfm?id=2876686.2876719.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi: 10.1080/14786440109462720. URL https://doi.org/10.1080/14786440109462720.

Kleanthi Lakiotaki, Nikolaos Vorniotakis, Michail Tsagris, Georgios Georgakopoulos, and Ioannis Tsamardinos. BioDataome: a collection of uniformly preprocessed and automatically annotated datasets for data-driven biology. *Database*, 2018, 03 2018. ISSN 1758-0463. doi: 10.1093/database/bay011. URL https://doi.org/10.1093/database/bay011.

Robert Fortet and Edith Mourier. Convergence de la répartition empirique vers la répartition théorique. *Annales scientifiques de l'École Normale Supérieure*, 3e série, 70(3):267–285, 1953. doi: 10.24033/asens.1013. URL http://www.numdam.org/item/ASENS_1953_3_70_3_267_0.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(1):723–773, March 2012. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2503308.2188410.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, July 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl242. URL https://doi.org/10.1093/bioinformatics/btl242.

Y. L. Tong. *Fundamental Properties and Sampling Distributions of the Multivariate Normal Distribution*, pages 23–61. Springer New York, New York, NY, 1990. ISBN 978-1-4613-9655-0. doi: 10.1007/978-1-4613-9655-0_3. URL https://doi.org/10.1007/978-1-4613-9655-0_3.

Alain Berlinet and Christine Thomas-Agnan. *Theory*, pages 1–54. Springer US, Boston, MA, 2004. ISBN 978-1-4419-9096-9. doi: 10.1007/978-1-4419-9096-9_1. URL https://doi.org/10.1007/978-1-4419-9096-9_1.

Caglar Aytekin, Francesco Cricri, Lixin Fan, and Emre Aksu. A theoretical investigation of graph degree as an unsupervised normality measure, 2018.

Dougal J. Sutherland, Hsiao-Yu Fish Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alexander J. Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. *ArXiv*, abs/1611.04488, 2016.

Stuart Coles. *An introduction to statistical modeling of extreme values*. Springer Series in Statistics. Springer-Verlag, London, 2001. ISBN 1-85233-459-2.

Kleanthi Lakiotaki, George Georgakopoulos, Elias Castanas, and Dimitri Røe Oluf. A data driven approach revealing disease similarity at the molecular level. *npj Systems Biology and Applications*, pages 1–24, 2019.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.*, 30(7):1145–1159, July 1997. ISSN 0031-3203. doi: 10.1016/S0031-3203(96)00142-2. URL http://dx.doi.org/10.1016/S0031-3203(96)00142-2.