BIOINFORMATICS

Vol. 00 no. 00 2015
Pages 1–5

Assignment 11

Lydia Ickler

Big Data in Life Sciences

Supervisors: Tim Conrad, Cristof Schütte

Berlin, 30 September 2015

ABSTRACT

Motivation: Analyzing huge amounts of biological data is a booming field within the bioinformatics. With this grow in data volume most analyzing tools reach their limits. Be it through memory issues or just too long running time. A possible solution to handle big data and perform machine learning or similar on it is the framework Apache Flink. Within this report I will develop a sparse classifier that can distinguish between breast and lung cancer. My starting point are two different data types (methylation and mRNA) that have each more than 20.000 features. By using well established Machine Learning algorithms (SVM and MLR) I will reduce the feature quantity to 50. Along the way I will in addition analyze one data source (mRNA) with a community detection algorithm to illustrate how methods from graph theory might represent an alternative approach for medical analysis.

Results: My results are confirming the presumption that a pool of different biological data can achieve better prediction rates than a singular data source. Though MLR as well as SVM performed largely well on the mixed classifier with all features combined can easily top that with an accuracy twice as good. Another insight is that the assigned MLR-weights seem to be appropriate allocated since a set of the ”most important” features (25 of each data type) also accomplishes a reasonable precision. With regard to the community detection algorithm the findings seem to be in line with the sparse feature set since more than 50% of the set can be found in the largest cluster.

Contact: lydia.ickler@fu-berlin.de

1 INTRODUCTION

Lung cancer is the most common cause of global cancer-related mortality, leading to over a million deaths each year and adenocarcinoma is its most common histological type [1]. The identification of the underlying biological mechanisms and more precisely its causing genes is a major task of current research [2]. Even though some oncogenes are already identified it is still not fully understood and there are certainly many more undiscovered players involved. Whole genome expression studies might permit systematic approaches to understand the correlation between gene expression profiles and disease states. One of the challenges in biological data analysis, especially in cancerous gene expression profiles, is to identify a small set of genes or groups of genes that are highly expressed in tumor cells but not in normal cells and vice versa. To detect those one has many options in terms of methods. The most common and established way is to use machine learning algorithms and detect a classifier. A more recent approach is the network-based approach. For example Chuang et. al did a network-based classification of breast cancer metastasis which did lead to new insights [3]. Projecting graph theory methods onto biological data is a growing trend within bioinformatics that has not reached its peak yet and is full of possibilities [4]. In context of this assignment I want to apply two different supervised machine learning algorithms, namely Multiple Linear Regression (MLR) and Support Vector Machine (SVM), followed by a pragmatic feature selection based on the MLR weights to then finally combine two data sources (methylation and mRNA) to test wether the whole united information or the reduced set of features end in a better result. This aims to prove the concept of current researchers which are confident that omics data classifiers can provide a more precise prediction than the one from a singular data source. Besides that I want to apply the connected components method which is a network-based analysis, likewise with merging two biological data sources.

2 MATERIALS AND METHODS

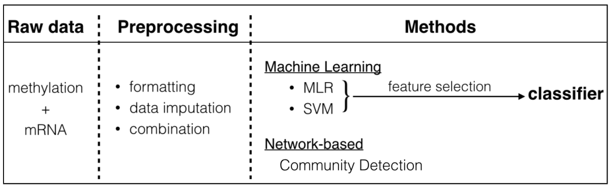


Fig. 1. Workflow-Outline

2.1 Raw Data

The data I used for my analysis was provided by The Cancer Genome Atlas (TCGA) [5]. This organization has a huge collection of various biological data types, e.g. exome, SNP, methylation, mRNA, miRNA and clinical data. Those sets they obtained are from patients that suffer from distinct kinds of cancer. I did choose two different cancer types, namely breast cancer (BRCA) and lung adenocarcinoma (LUAD). At first I wanted to only use one cancer type and create a classifier to separate healthy and diseased patients. Unfortunately, within the TCGA data are no control groups available. One advantage of using two different cancers is that you can find a more specified gene set that separates them due to the fact that they have the ”cancer marker” in common. This stands in contrast to comparing healthy and ill people where you are more likely to find rather general separation criteria. My final choice was to train a classifier using two different types of biological data from BRCA and LUAD. In detail I downloaded the methylation and mRNA data of each cancer type, always of Level 3.

2.2 Preprocessing

2.2.1 Creating data sets

As mentioned before I downloaded two data sets from the TCGA platform for each cancer type. Those were compressed directories that contained the measurements for each patient in separate files. I did choose to program my pre-processing in R. The machine learning algorithms both in Flink and R do need a certain input format. In general this is a data structure that consists of a dependent variable y (cancer type) and an array of independent variables $\vec{x}$ (feature expression). So that it had at the end the following form of

<label>, <array of values>.

To obtain the desired format I iterated over all files within each folder and extracted the measurements and merged all patients into one matrix that has the required format. As a last step I united the two cancer types in one file for each data source. For this step I payed attention to have the same quantity of each cancer group. All in all I got three CSV-files: One with the methylation data, one with the mRNA data and one with the combined sets. The detailed numbers can be found in the following table.

	methylation	mRNA	methylation + mRNA
# patients	252	1.032	120
# features	24.981	20.502	45.483

Table 1. Data set details

2.2.2 Dealing with Missing Data

The set of transcriptomic data was complete and without any missing data points. In contrast to that was the methylation set missing a lot of measurements. In both the BRCA and LUAD set were $\sim 1\%$ of the cells filled with NaNs. My first step was to get rid of all those columns that are nothing but NaNs. In both cases it was 2597 columns that could be removed. I guess this has something to do with the way the measurement was performed. After that remained only widely distributed missing cells. One pragmatic solution would be to fill the gaps with the means of the belonging column. A more sophisticated way present the so called "data imputation methods" [6]. One of those methods is the matrix completion which is usually applied in the setting of recommender systems. Long story short it boils down to a minimization problem of the root-mean-square error (RMSE) between existing ratings and their predicted values plus some regularization term to avoid overfitting:

$$\min_{X,Y} \sum_{r_{ui} exists} (r_{ui} - x_u^T y_i)^2 + \lambda(\sum_u n_u||x_u||^2 + \sum_i n_i||y_i||^2)$$

$X = (x_1, \ldots, x_n)$ is the matrix of user-factor vectors and $Y = (y_1, \ldots, y_m)$ is the matrix of item-factor vectors. n_u and n_i denotes the number of existing ratings of user u and item i, respectively. According to Zhou et al., this weighted-λ-regularization gives best empirical results. To project this method onto my biological data I just had to replace users with patients and items with feature values. The Flink implementation was provided by the programmer collective data artisans [7] and is except for the special input format (patientID, featureID, value) very simple to use. At the end I got as a result an user-factor matrix as well as an item-factor matrix which I then had to multiply to finally receive the imputated matrix. The pre- and post-processing scripts were written in R and the matrix completion was performed in Flink with the script of a former homework. Unfortunately the execution was not possible on the cluster.

2.3 Methods

The custom written methods are either implemented in Flink or R. In case of the R scripts one does not have to set parameters. The applications run out of the box and just need the correct paths to the input files or directories. In addition I want to add that their purpose is mainly for transforming the data with regard to style or merging. In contrast to that do the Flink applications come with different options. The corresponding input paths are all already set and the files are stored within the HDFS. For both programs you always have to define the data type you want to analyze (methylation, mRNA, mixed or sparse). There are two different applications. You can either perform classification (MLR or SVM) on the different sets which also includes the final classifier or you can perform a community detection on the mRNA set. Following are detailed descriptions of the used algorithms and a short user manual for each implementation.

2.3.1 Mutltiple Linear Regression

Multiple linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to observed data. Every value of the independent variable x is associated with a value of the dependent variable y. Given a set of input data with its value (x, y), multiple linear regression finds a vector w such that the sum of the squared residuals is minimized. This problem has a closed form solution which is given by:

$$w^* = (X^T X)^{-1} XTy$$

However, in our case the input data set is so huge that a complete parse over the whole data set is prohibitive. A knack is to apply stochastic gradient descent (SGD) to approximate the solution [8]. SGD first calculates for a random subset of the input data set the gradients. The gradient for a given point x_i is given by:

$$\Delta_w S(w, x_i) = 2(w^T x_i - y)x_i$$

The gradients are then averaged and scaled. The scaling is defined by $\gamma = \frac{s}{\sqrt{j}}$ with s being the initial step size and j being the current iteration number. The resulting gradient is subtracted from the current weight vector giving the new weight vector for the next iteration. The multiple linear regression algorithm computes either a fixed number of SGD iterations or terminates based on a dynamic convergence criterion. The program is accessible via the jar file MachineLearning.jar <method> <data type> <output path>. It produces two outputs. One is a file that contains the weights of the linear regression and the other result is a message on the console showing the mean error rate of the 10-fold cross-validation. The produced weights are essential for the feature reduction which will follow later on.

2.3.2 *Support Vector Machine*

A support vector machine constructs a hyperplane or a set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. The Flink-method implements an SVM with soft-margin using the communication-efficient distributed dual coordinate ascent algorithm with hinge-loss function [**?**]. The algorithm solves the following minimization problem:

$$\min_{w \in \mathbb{R}^d} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} l_i(w^T x_i)$$

with w being the weight vector, λ being the regularization constant, $x_i \in \mathbb{R}^d$ being the data points and l_i being the convex loss functions, which can also depend on the labels $y_i \in \mathbb{R}$. In this implementation the regularizer is the ℓ_2-norm and the loss functions are the hinge-loss functions:$l_i = \max(0, 1 - y_i w^T x_i)$ The minimization problem is solved by applying stochastic dual coordinate ascent (SDCA). In order to make the algorithm efficient in a distributed setting, the CoCoA algorithm calculates several iterations of SDCA locally on a data block before merging the local updates into a valid global state. This state is redistributed to the different data partitions where the next round of local SDCA iterations is then executed. The number of outer iterations and local SDCA iterations control the overall network costs, because there is only network communication required for each outer iteration. The local SDCA iterations are embarrassingly parallel once the individual data partitions have been distributed across the cluster. The implementation of this algorithm is based on the work of Jaggi et al. Unfortunately, the SVM implementation does not produce an insight to the weights of each feature. Therefore, I did not take the SVM-feature weights into account for the following feature selection. Another issue with the SVM is that I did not get it to work on the cluster due to some exceptions.

2.3.3 *Creation of a Custom Classifier*

To reduce the number of features I saved the feature weights of the MLR-classification into a separate file. This file I again read in via an R-script. I ordered the weights according to their absolute values and then took the 25 most important (highest) features of each the methylation set and the mRNA set. Collectively I then had my combined and sparse feature set of 50 different genes (see Table 2).

SFTPA1	CEACAM5	S100A6	NAPSA	SERPINA3
LTF	CD24	AZGP1	EEF1A1	SLC39A6
AHNAK	C4A	COL3A1	CD74	CPB1
CEACAM6	XBP1	SLC34A2	COL1A2	COL1A1
SFTPA2	MGP	FTL	ADAM6	SFTPB
C9orf167	TNFRSF10D	ARFIP2	ICAM4	CORIN
SLC30A4	BANK1	PI15	TNFRSF10D	SLC16A3
PKIA	MYT1	FAM107B	CLIP4	NOX3
PDGFRB1	TAS2R13	TNS1	SPP2	TAL2
STON2	SCOC	TBX19	DNM3	IL20

Table 2. Sparse Feature Set

Compared to the combined classifier with 45.483 features this is an immense sparse set (submixed_Flink.csv). To see how well my final classifier performs I again applied both the MLR and SVM with an additional 10-fold cross-validation.

2.3.4 *Network-based Approach: Community Detection*

In the study of graph theory, a network is said to have community structure if the nodes of the network can be easily grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally. In the particular case of non-overlapping community finding, this implies that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups [10]. To create such a network for my biological data I did calculate a correlation matrix. The main idea is to build a network based on expression values and then find clusters of similar nodes. The correlation matrix I determined with the help of Pearson's correlation coefficients. Unfortunately, the methylation measurements are within a range of 0 to 1. This brings the disadvantage that relations are assumed to soon. Therefore, I applied the community detection algorithm only to the mRNA data. Every time a correlation coefficient between two features exceeded 0.8 I did draw an edge. Finally I created a adjacency file with featureID1 , featureID2 and an 1.0 that is indicating the connection. This mRNA_edges.txt file was then the input for my final program. The output of the community detection algorithm I transformed again via an R-script to finally analysis it visually.

3 RESULTS

3.1 Performance of Prediction Methods

3.1.1 *Multiple Linear Regression*

The MLR implementation in Flink is heavily dependent on the initial step size parameter. This variable was more or less arbitrary chosen and finally set to $1.0 * 10^{-11}$ for all data types. The iteration number I adjusted to 10. The performance of the MLR-classification in terms of error rates were surprisingly good. All data sets obtained an error rate below 10%. As one can see in Table 3 did the data sets with only a singular data source (methylation, mRNA) perform the worst. The combined set with more than 40.000 features has the best accuracy with 98.12%. But if you compare that with the result of the sparse feature set it only worsens by $\sim$ 1%. This is a really good prediction rate if you think about the enormously small feature set and in addition about the way it was randomly chosen. Obviously there is much room for improvement when it comes to the feature selection.

	methylation	mRNA	mixed	sparse
error rate	0.0573	0.0442	0.0188	0.0299

Table 3. Error rates of the different classifiers using MLR.

3.1.2 *Support Vector Machine*

Exactly as the MLR is the SVM implementation in Flink strongly dependent on the initial step size parameter. Also this time the variable was more or less arbitrary chosen and finally set to 0.01

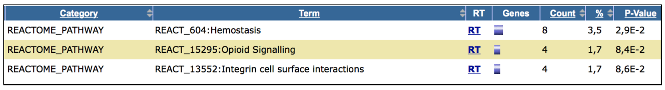
for all data sources. The iteration number I adjusted again to 10. The performance of the SVM-classification in terms of error rates did present a rather mixed picture. The obtained error rate did differ from pure chance to very accurate. As listed in Table 4 did the data set with methylation as its only data source perform by far the worst. The prediction on the basis of the transcriptomic data on the other hand had an error rate just below 10%. The combined set with more than 40.000 features has again the best accuracy with 2.5%. But if you compare that with the result of the sparse feature set it only worsens by $\sim 1\%$. This is a really good prediction rate if you think about the enormously small feature set and in addition about the way it was randomly chosen. Obviously there is much room for improvement when it comes to the feature selection.

	methylation	mRNA	mixed	sparse
error rate	0.2527	0.0890	0.0250	0.0370

Table 4. Error rates of the different classifiers using SVM.

3.2 Performance of Community Detection

The netword created by the correlation matrix had in total 3015 edges. By the usage of the community detection algorithm 1003 communities could be discovered. This number indicates that the majority of nodes are a module by themselves. In fact about 75% features were clustered alone. The largest community had 233 members. Finally I wanted to see how big the overlap of my sparse feature set is with the biggest community. Surprisingly 36 features of the classifier were present within the largest cluster. To gain more insights into the potential biological meaning I parsed my gene list into the web client of DAVID Bioinformatics Resources [11] which is an online Functional Annotation Tool. There I checked the association of my gene list within the pathway database Reactome [12] (see Figure 2). As the first hit with a p-value of 0.029 is "Hemostasis" listed. Hemostasis is a process which causes bleeding to stop, meaning to keep blood within a damaged blood vessel. In general it is the first stage of wound healing. After some research I discovered that alterations in this specific pathway are well known among cancer patients. This result might be interesting to investigate further.

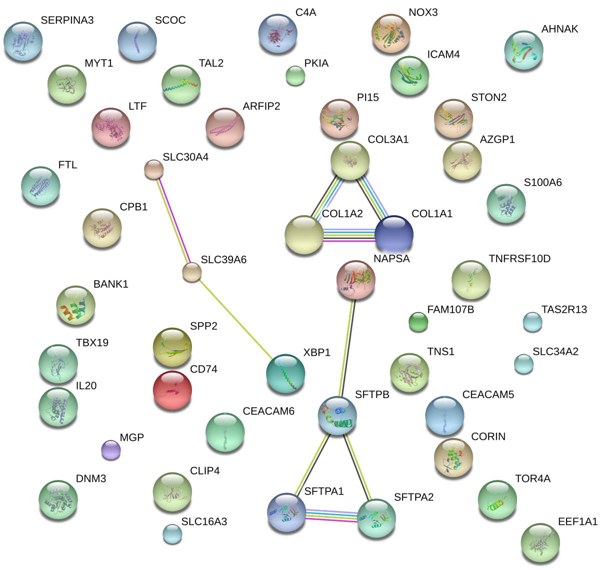


Category	Term	RT	Genes	Count	%	P-Value
REACTOME_PATHWAY	REACT_604:Hemostasis	RT	▬	8	3,5	2,9E-2
REACTOME_PATHWAY	REACT_15295:Opioid Signalling	RT	▬	4	1,7	8,4E-2
REACTOME_PATHWAY	REACT_13552:Integrin cell surface interactions	RT	▬	4	1,7	8,6E-2

Fig. 2. Results of pathways analysis with web service Reactome

3.3 Visualisation of the sparse feature set

In conclusion of my result section I will now present the sparse feature list (see Table 2) as a graph with possible connections. The web service of String-DB [13] offers a great way to visualize protein interactions based on known and predicted Protein-Protein Interactions. I only had to parse my gene list into the web client and obtained then a graphical interpretation. The result can be seen in Figure 3.3. Out of the 50 features only 2 were not recognized by the platform. Those were ADAM6 and CD24. Both genes are positively associated with the state of cancer. Nevertheless, the majority of nodes could be displayed. Though most vertices have no edges at all one can see three clusters consisting of 3 to 4 proteins. Especially the connections between COL1A1, COL1A2 and COL3A1 seems to be very significant. But since I am no doctor or biologist I do not want to speculate on the meaning of said and leave it at a monitoring level.



4 DISCUSSION

Overall the created workflow is working locally and produces solid results. Of course the building process did contain some difficulties as well as a lot of possibilities for improvements. One issue I want to point out is the fact that running a program on a computing cluster does afford much more work than the local execution. For once when you build the maven project from source you would expect to download all required libraries as well. This is unfortunately only partly true. In case of the machine learning section everything runs smoothly when you execute your program locally. But during the creation of an executable jar file not all libraries are included. You get an error messages of "java.lang.NoSuchMethodError". To get rid of those I had to search in total several times for the necessary jar in the worldwide web. This research was not only annoying but also time consuming. It were the libraries of breeze, blas and core. After some further searching I observed that these are "excluded" during the building of the maven project. I did not find out why this is the case. In case of the matrix completion as well as the SVM implementation I did not find a solution.

Another issue was the missing data points within the methylation data. Missing data is a common phenomenon when it comes to biological data. It can occur due to wrong measurement, technical failures or mostly because of mistakes by the person executing the experiment. In this case I tried to restore the data with the help of a matrix completion that was pre-implemented within the Flink-framework. The use case is normally a completion for recommendation matrices and not for biological data. Therefore, I am not sure if the choice of algorithm is neither reliable nor appropriate for this type of data source. Nevertheless I applied the method to get a complete input file so that my following analysis steps would work. Of course further thinking should be invested if one had more time.

Finally I want to speak about the feature selection that was only touched on the surface and which was handled very practical. The main reason it was dealt that way is because there is no gold standard implementation available in Flink or at all. In the bioinformatics community do exist various ways to solve that issue, e.g. PCA or correlation analysis - to name a few. Unfortunately I had not the time to read into it. Of course there are a ton of much more sophisticated ways to setup a classification workflow but overall I am satisfied with what I have come up. In addition I learned along the way how to plan and execute a project as well as new programming skills.

REFERENCES

[1]Siegel, R. L., Miller, K. D. & Jemal, A. (2015). Cancer Statistics. *Cancer Journal for Clinicians*, 65(1), 5–29.

[2]Zhang, L., Farrell, J. J., Zhou, H., Elashoff, D., Akin, D., Park, N.-H.,. . . Wong, D. T. (2010). Salivary Transcriptomic Biomarkers for Detection of Resectable Pancreatic Cancer. *Gastroenterology*, 138(3), 949–957.

[3]Chuang, H.-Y., Lee, E., Liu, Y.-T., Lee, D., & Ideker, T. (2007). Network-based classification of breast cancer metastasis. *Molecular Systems Biology*, 3, 140.

[4]Berger, B., Peng, J., & Singh, M. (2013). Computational solutions for omics data. *Nature Reviews. Genetics*, 14(5), 333?346.

[5]The Cancer Genome Atlas, http://cancergenome.nih.gov.

[6]Williams, D., Xuejun L., Ya X., Carin, L., Krishnapuram, B. (2007). On Classification with Incomplete Data. *Pattern Analysis and Machine Intelligence, IEEE Transactions*, 29(3), 427–436.

[7]Rohrmann, T., How to factorize a 700 GB matrix with Apache Flink. *http://data-artisans.com/how-to-factorize-a-700-gb-matrix-with-apache-flink/*, posted on March 30, 2015.

[8]Apache Flink, FlinkML - Multiple linear regression. *https://ci.apache.org/projects/flink/flink-docs-master/libs/ml/multiple_linear_regression.html*.

[9]Apache Flink, FlinkML - SVM using CoCoA. *https://ci.apache.org/projects/flink/flink-docs-master/libs/ml/svm.html*.

[10]Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3–5), 75?174.

[11]Huang da, W., Sherman, B. T., Lempicki, R. A. (2009). Systematic and integrative analysis of large gene lists using DAVID bioinformatics resources. *Nature Protocols*, 4(1), 44–57.

[12]Milacic ,M., Haw, R., Rothfels, K., Wu, G., Croft, D., Hermjakob, H., D'Eustachio, P., Stein L. (2012) Annotating cancer variants and anti-cancer therapeutics in reactome. *Cancers (Basel)*, 4(4), 1180–211.

[13]Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., . . . , Jensen, L. J. (2013). STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Research*, 41(Database issue), D808?D815.