SOURCE CODE FOR
BIOLOGY AND MEDICINE

METHODOLOGY Open Access

Knowledge Driven Variable Selection (KDVS) – a new approach to enrichment analysis of gene signatures obtained from high–throughput data

Grzegorz Zycinski[1*], Annalisa Barla[1], Margherita Squillario[1], Tiziana Sanavia[2], Barbara Di Camillo[2] and Alessandro Verri[1]

Abstract

Background: High–throughput (HT) technologies provide huge amount of gene expression data that can be used to identify biomarkers useful in the clinical practice. The most frequently used approaches first select a set of genes (i.e. gene signature) able to characterize differences between two or more phenotypical conditions, and then provide a functional assessment of the selected genes with an *a posteriori* enrichment analysis, based on biological knowledge. However, this approach comes with some drawbacks. First, gene selection procedure often requires tunable parameters that affect the outcome, typically producing many false hits. Second, *a posteriori* enrichment analysis is based on mapping between biological concepts and gene expression measurements, which is hard to compute because of constant changes in biological knowledge and genome analysis. Third, such mapping is typically used in the assessment of the coverage of gene signature by biological concepts, that is either score–based or requires tunable parameters as well, limiting its power.

Results: We present Knowledge Driven Variable Selection (KDVS), a framework that uses *a priori* biological knowledge in HT data analysis. The expression data matrix is transformed, according to prior knowledge, into smaller matrices, easier to analyze and to interpret from both computational and biological viewpoints. Therefore KDVS, unlike most approaches, does not exclude a priori any function or process potentially relevant for the biological question under investigation. Differently from the standard approach where gene selection and functional assessment are applied independently, KDVS embeds these two steps into a unified statistical framework, decreasing the variability derived from the threshold–dependent selection, the mapping to the biological concepts, and the signature coverage. We present three case studies to assess the usefulness of the method.

Conclusions: We showed that KDVS not only enables the selection of known biological functionalities with accuracy, but also identification of new ones. An efficient implementation of KDVS was devised to obtain results in a fast and robust way. Computing time is drastically reduced by the effective use of distributed resources. Finally, integrated visualization techniques immediately increase the interpretability of results. Overall, KDVS approach can be considered as a viable alternative to enrichment–based approaches.

Keywords: Prior knowledge, High–throughput data, Gene signature, Enrichment analysis, Gene Ontology, Embedded Methods, $\ell_1\ell_{2FS}$

*Correspondence: grzegorz.zycinski@unige.it
[1] DIBRIS, University of Genoa, via Dodecaneso 35, I-16146 Genova, Italy
Full list of author information is available at the end of the article

Background

One of the challenges of modern molecular biology is to reconstruct the complex network of processes that govern all the activities of living organisms. HT technologies allow to measure the expression of thousands of genes simultaneously for each single biological sample [1,2], but these data are difficult to analyze because the number of samples is always lower with respect to the number of variables (e.g. genes, proteins). Therefore, traditional statistical methods are unsuitable to face this challenge. Nevertheless, these data are currently used to identify possible biomarkers, useful in the clinical practice: 1) to stratify a group of people affected by a disease into separated groups that are predicted to progress differently, 2) to subtype a disease through the identification of subgroups of people, that are characterized by a different molecular landscape, or by a different response to a specific drug. The probability of finding biomarkers is higher within the specific list of genes, known as gene signature [3,4], whose expression levels are able to separate two distinct phenotypic conditions (e.g. disease and healthy, tumor and metastasis) in comparison of two classes of biological samples. Such list can be identified in a typical binary classification setting [5].

Microarrays are a popular example of HT technology currently used to evaluate gene expression data that potentially contain meaningful gene signatures. The typical approach to analyze HT data is to first identify meaningful gene signatures and then to perform enrichment analysis [6,7].

Several methods can be chosen to identify gene signatures. The most commonly used filtering methods (e.g. t–test, ANOVA) [8,9] compare gene expression measurements to identify differentially expressed genes (DEG). The choice of the selection method is crucial because it affects the enrichment analysis. Specifically, if some relevant genes had not been included in the signature, it may not be possible to identify processes or functions that are connected with those genes, and that are relevant to the biological question addressed [10].

In the enrichment analysis, selected genes of the signature are compared with previously determined functional gene groups (that represent the current biological knowledge). To determine such groups, first it is necessary to assign genes to the biological concepts, that can be accomplished in various ways [11,12]. Then, the significant presence of genes in an individual group is assessed with some scoring schema or statistical test [6,13,14].

Thus, in the classic enrichment analysis approach, the discovery method is not affected by the prior knowledge and it is performed independently with respect to the functional analysis. On the opposite, using prior knowledge before the selection step, the discovery method is applied according to the prior knowledge, thus giving a direct biological contextualization of the gene signature (Figure 1).

Several efforts in this direction have been recently presented in the literature. In [15], pathway information has been incorporated into logistic regression models. In [16-20] topological properties of Kyoto Encyclopedia of Genes and Genomes (KEGG) pathways have been used to constrain the learning process. The use of Gene Ontology (GO) as prior information has been explored in [21], where the authors propose a classification model based on functional groups of genes. A comparison of the performance obtained using different kind of knowledge has recently been presented in [22].

In this paper, we present KDVS (see Additional file 1), an implementation of the second case aforementioned, where the prior biological knowledge is used before the identification of the gene signature, the procedure itself is not altered, and the gene signature is defined according to the chosen source of prior knowledge. This framework can be considered as an alternative to most popular approaches of HT data analysis.

In contrast to other methods, KDVS accepts as input, besides the gene expression data, also specific annotations for HT platforms, and a representation of prior biological knowledge. As output, instead of gene signature and the results of the enrichment analysis, KDVS produces the list of significant biological concepts, as well as, for each of them, the list of significant genes that can be considered as individual signature associated to the concept.

The current implementation uses microarrays as source of gene expression data [23]. To analyze these data, among several supervised classification and variable selection methods [4,24], we choose $\ell_1\ell_{2FS}$, an embedded regularization method for variable selection, proven to identify subsets of discriminative genes [25,26]. Gene Ontology (GO) is used as prior knowledge source [27].

KDVS is implemented in Python [28]. It was designed according to the principle of individual *applications* that share common functionality through the Application Programming Interface (API). Such modularized way allowed to dynamically adjust desired functionality piece by piece across a long time frame, which is crucial in the development of successful methodologies in statistical learning community. Also, it allowed to implement necessary solutions for standard bioinformatics–related problems, such as identifier mappings, in independent way. At the end, it allowed to seamlessly introduce the usage of parallel resources for time–consuming computations. Furthermore it is general enough to be applied to any other experiment that involves a two classes setting problem and it could be extended to a regression or multi-class setting.

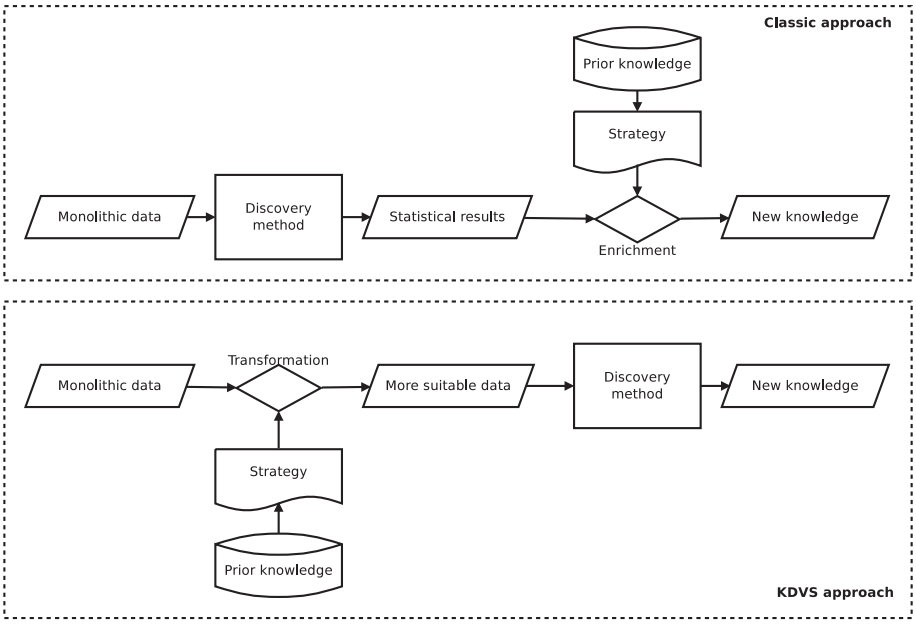

Figure 1 The difference between classic approach and the KDVS approach. Illustration of the difference between classic enrichment–based approach and the KDVS framework approach. In classic, monolithic gene expression data are mined for significant genes and prior biological knowledge is used a posteriori to verify the soundness of result. This approach is sensitive to the choice of mining technique, as well as the enrichment verification method. In KDVS, monolithic gene expression data are transformed according to prior knowledge (e.g. divided in smaller parts accordingly), and then mined for significant genes. This approach enables wider choice of mining techniques and provides biological insight before the core mining step.

The framework has been successfully applied to the analysis of microarray data in experiments regarding neurodegenerative diseases, namely Alzheimer's and Parkinson's diseases [29,30], as well as prostate cancer [31].

Methods

KDVS: Framework Overview

The schema of the overall framework activity is presented in Figure 2. It is divided into *experiment part* and *postprocess part*. During the experiment part, all initial data processing takes place, all computational tasks are performed and the results are collected. The postprocess part works further on computational results to generate additional useful information and statistics. The experiment part is implemented in *experiment.py* and the postprocess part is implemented in *postprocess.py*.

Input data

The framework follows standard procedure for preparing microarray gene expression data [32]. The data are normalized using one of the standard methods available from BioConductor [33] set of packages (e.g. RMA, GCRMA, aroma). Its output is the Gene Expression Data Matrix (GEDM), in the form of Delimiter-Separated Values (DSV) file. The matrix has P rows and $N+1$ columns, where P is the number of genes/probe sets monitored over N biological samples, with the first column reporting the genes/probe sets IDs.

Moreover, the framework uses platform–specific annotations, that describe each measurement and all the related details (e.g. concerning the biological sequence used, the gene associated with it, its relation to other biological entities). In the case of microarray data, the framework utilizes chip–related annotations provided directly by the manufacturer or the research community in the form of a DSV file. KDVS uses platform annotations available from Gene Expression Omnibus (GEO) [34] for Affymetrix microarrays.

Since the framework heavily relies on prior biological knowledge, it utilizes a tailored representation of it. For example, for GO [27], the encoded Directed Acyclic Graph (DAG) structure is obtained, along with basic data regarding individual terms, such as the term identifier, the name and the description. In the current implementation, the framework uses the representation of GO graph, encoded in the RDF-XML file.

To perform supervised classification, it is necessary to associate each biological sample from the experiment with the phenotypical outcome, symbolized by specific *labels* that are often numerical [35]. For example, in the two class setting, samples can be associated with -1 and 1 labels accordingly. The framework uses label information

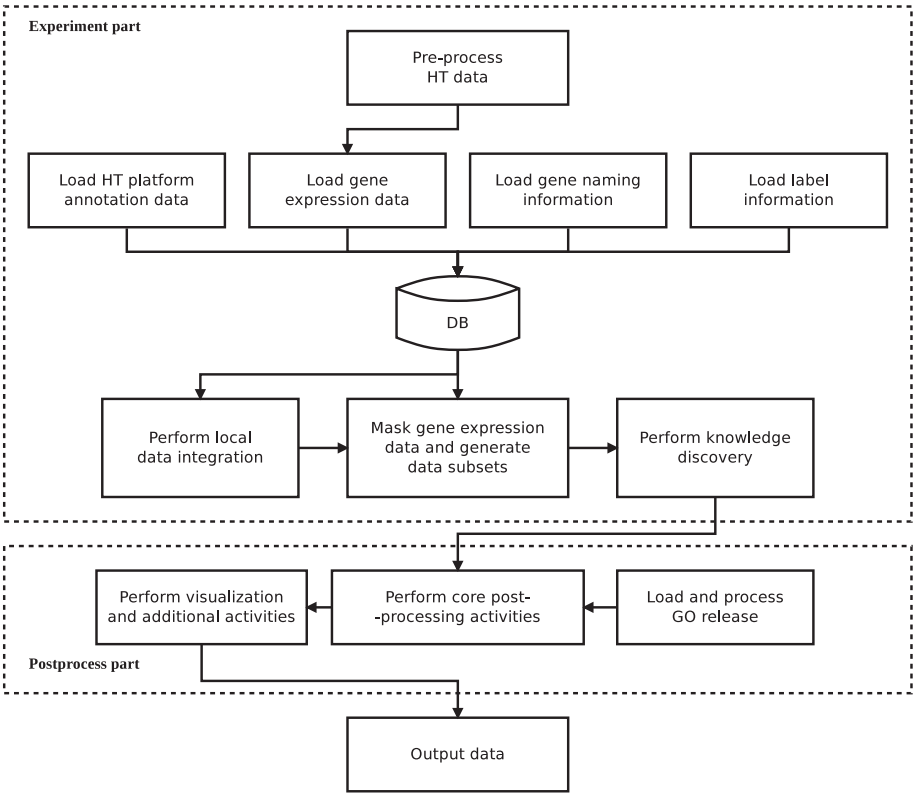

Figure 2 General activity schema of KDVS. Diagram of the general activity of KDVS.

presented in the form of DSV file, where each sample is associated with the corresponding label.

Statistical analysis

The DSV data sets are loaded into individual tables ("raw" tables) of relational database. Specifically, a DSV data set containing A columns and B rows is loaded into the corresponding relational table that has A columns and contains B rows. In current implementation, GEDM matrix is loaded into "*GEDM*" table, the platform–related annotations is loaded into "*ANNO*" table, gene naming information is loaded into "*HGNC*" table, and the label information is loaded into "*LABELS*" table.

Querying raw tables directly may not be sufficient in some cases. To fulfill the main objective in effective way, some *derived* tables may be created: they contain recombined information from raw tables, and allow fast querying of raw data. For example, the Affymetrix annotations contain mapping *probeset*→*GO terms*. It is necessary, however, to obtain reverse mapping *GO term*→*probesets*, to associate each GO term with raw measurements. Therefore, the *term2probeset* derived table is created to allow fast querying of mapping *probeset*↔*GO term* in both directions. However, platform–related annotations may not be up–to–date, regarding constant changes in annotations of genomes. This, for example, could affect gene naming (by presence of obsolete symbols), or coverage of genes by probesets (by rearrangement of gene sequences). Therefore another derived table, *probeset2gene*, is created to control the information from platform–specific annotations regarding gene naming, in accordance with the data obtained from the HUGO Gene Nomenclature Committee (HGNC) [36]. The process of construction of derived tables is also referred to as *local data integration*.

During the next step, *GEDM* and *term2probeset* tables are queried to generate specific expression data subsets that correspond to individual GO terms. The concept can be envisioned as *masking* the original monolithic GEDM data set and retrieving only the expression measurements that are associated with the specific GO term. The masking is repeated for each GO term in the *term2probeset* table that comes from the specified GO domain. Each data subset forms a matrix of N columns and P_X rows, where P_X is $||P$. By construction, the data subsets overlap given the tree–structure of GO. Indeed each GO term may include several variables also belonging to other GO terms.

In the following step, a supervised classification is performed for each data subset. This step is also referred to

as *knowledge discovery*, being the core part of recovering the functional characterization from gene expression data. In the current implementation, a variable selection technique $\ell_1\ell_{2FS}$ is used [37-39]. The method returns the *classification error*, as well as the list of *selected variables* (in our case *probesets*), that are the most discriminant between two classes. The procedure depends on a *correlation* parameter μ that governs the amount of correlated variables selected in the final list. To evaluate an unbiased classification performance, $\ell_1\ell_{2FS}$ performs a full *model selection* by using two nested K–fold cross–validation loops [39]. This also guarantees robustness, sensitivity and specificity in feature selection [40]. Data subset is further split into smaller parts, then classification is performed on those parts and finally these partial results are integrated into the final classification call. Based on that principle, the method counts the appearance of every single variable in each selected variable list obtained for each split, and it reports in addition the resulting frequency.

Where data subset is *sufficiently small*, that is, where the number of variables is roughly the same as number of samples, there is no need to perform full model selection. Thus, for those data subsets, a classification task is performed with Regularized Least Squares (RLS), and all variables are treated as selected or not, depending on the classification outcome.

The described statistical analysis focuses on the level of *probesets* instead of *genes*. This approach shows some considerable advantages. Typically, according to microarray design principles [41], the biological sequence of the gene can be monitored by more than one probeset. Therefore, to perform the analysis on the *gene* level, some method of aggregating the values of each probeset is required, e.g. by taking the average of the expression values. However, such procedures introduce certain additional bias into subsequent statistical analysis [35,42]. Since KDVS performs the analysis at the *probeset* level, there is no need for such procedures, and the mentioned additional bias is not introduced.

Given the machine learning procedure used for supervised classification, it may be desirable to utilize *a parallel computing environment* to speed up numerical computations [43]. In the current implementation, an ad–hoc environment was built over a local network of multi–core desktop machines, where the computational tasks were distributed to individual machines and executed in the background. The environment is controlled using the Python package PPlus [44].

The output of the experiment part consists of an *ensemble* of relational database and filesystem objects. The ensemble is used to store all the information used within the pipeline for further reference. This information includes: all the input data, the results from local data integration, all the generated data subsets and the output from knowledge discovery. The output from knowledge discovery becomes the direct input for the next part.

Postprocess phase

This part is roughly composed of the *core post–processing activities* and the *visualization* tasks.

During *core activities*, the results obtained from the *knowledge discovery* step are reviewed and an *error estimate* is obtained. Here, if the classification error is *below* the chosen threshold, the GO term associated with a specific data subset is considered *significant*. Next, for each significant term, the absolute frequency of each of its selected variables is checked: if the frequency is *above* the specified threshold, the variable is considered *properly selected*. The frequency is checked across variables selected in the outer K–fold cross validation loop during the *model selection* phase (see [39] for details). Next, several useful statistics are collected, such as the standard properties of classification error (i.e. the mean, the standard deviation, the median, the histograms of selected and non–selected variables across all data subsets). Basic plots of classification errors are generated as well.

Furthermore, each variable (here defined as probeset) is *annotated* back with biologically meaningful annotations, such as the gene symbol related to the probeset, and the identifiers of the corresponding Entrez Gene [45] and Genbank [46] database records. The annotations come primarily from the *ANNO* table, and are verified against *HGNC* table with gene naming information [36], if applicable.

The output of the *core activities* of the postprocess step is the list of *significant GO terms* that pass the supervised classification step, and for each significant GO term, the list of *properly selected* probesets. The classification results allow focusing on specific biological functionality represented by the GO terms, and the frequencies allow focusing only on those genes determined as playing important role in a biological experiment. Even if the output obtained during the *core activities* in the postprocess part is meaningful, its further biological analysis may be difficult, due to its low interpretability for researchers outside machine learning and statistics communities. The introduction of the visualization step is crucial to represent the results in a more visually clear way and it is useful to perform further biologically–oriented investigations.

To this aim, significant GO terms are visualized on the minimal subgraph built over the complete DAG of specific GO domain [47,48]. The subgraph is generated with the *GOstats* BioConductor [33] package and visualized with R interface to *GraphViz* graph plotting library [49].

In order to further increase the interpretability of the results, a *semantic clustering* was introduced. Referring to the DAG, for each GO term the Information Content [50] is calculated by comparing the number of occurring

annotations in the graph subsumed by that term to the total number of annotations belonging to the GO graph subsumed by the root node. This metric is based on frequency of the term occurring in annotations by taking advantage of the structural information organized in GO; thus a rarely used term in the GO graph contains a greater amount of information. Starting from this metric, it is possible to define a semantic similarity measure in order to assess the degree of relatedness between two GO terms. There are different similarity measures based on this kind of metric [51]. Here, Resnik semantic similarity was used since it associates to a pair of terms the information content of their common ancestor, thus preserving the level of specificity of the relationships between the terms. For the implementation of both the information content and the Resnik measure the standalone *FastSemSim* Python application [52] was used. FastSemSim requires as input data the description of the DAG and the GO annotations. This application allows other different semantic similarity measures, thus the user can also decide to use a different metric. However, since the aim is to group the selected GO terms into homogeneous clusters, a normalized measure is required. In this case, Resnik measure was normalized to the maximum observed value.

Given the list of significant GO terms defined in the postprocess step, a semantic similarity matrix was build from the Resnik pairwise similarities provided by FastSemSim. The similarity matrix was then used as input to perform a hierarchical clustering. In order to select the clusters, a default semantic threshold equal to 0.8 was used in the implementation, which can be changed by the user. Finally, the resulting clusters of terms were plotted on the minimal subgraph with different colors using *GOstats* and *GraphViz*.

Output data

The main interpretable output of KDVS consists of the lists of *significant GO terms*, and for each significant term, the list of *properly selected* variables (determined during *Postprocess phase*).

Significant GO terms are obtained both with $\ell_1\ell_{2FS}$ and RLS techniques. For $\ell_1\ell_{2FS}$, a list of significant terms is generated for each used μ value [39], and for RLS a single list is generated. Since $\ell_1\ell_{2FS}$ procedure involves full model selection as well as variable selection, some meaningful details are collected here for each significant GO term as well. To obtain a consistent output, each list of GO terms obtained with $\ell_1\ell_{2FS}$ for given μ value, is merged with single list of GO terms obtained with RLS, thus making *unified term lists*.

The specific details noted for significant GO terms obtained with $\ell_1\ell_{2FS}$ include: the index of proper $\ell_1\ell_{2FS}$ μ value, mean and standard deviation of test error, mean and standard deviation of training error, median of test error, total number of variables in corresponding data subset, and number of properly selected variables. Mean test error is later re–used for unified term list as classification error estimate. $\ell_1\ell_{2FS}$ should guarantee that the classification error estimate does not change much for different values of μ. An abbreviated example output list of significant GO terms obtained with $\ell_1\ell_{2FS}$ is presented below:

```
GO term ID,Mu,Mean TS,Std TS,Mean TR,Std TR,
Med TS,Tot vars,Sel vars
GO:0005515,0,0.0595927312902,0.0237055815607,
    0.0,0.0,0.0624713039486,15035,230
GO:0046872,0,0.045928030303,0.0265278311133,
    0.00520912987994,0.00521768678784,
    0.0606060606061,6596,98
GO:0008270,0,0.0693583562902,0.033817940487,
    0.00781221192909,0.00851309206943,0.0625,
    5250,87
GO:0003677,0,0.044951467803,0.0161935917425,
    0.0104182597599,0.00708073456502,
    0.047881155303,4895,62
GO:0000166,0,0.0478739812902,0.033404962323,
    0.013073940168,0.0173600747312,
    0.049834280303,4479,65
```

Each unified term list consists of significant GO terms obtained with $\ell_1\ell_{2FS}$ for single μ value, and significant GO terms obtained with RLS. The following details are collected for each term: its full name, total number of variables in corresponding data subset, number of properly selected variables, classification error estimate used for significance call, numbers of: true positives, true negatives, false positives, and false negatives, collected across outer K–fold cross–validation loop, as well as the corresponding Matthews Correlation Coefficient (MCC). An abbreviated example of unified term list is presented below:

```
GO term ID GO term name Tot vars Sel vars
Error estimate #TP #TN #FP #FN MCC
GO:0003677 DNA binding 2699 1295
    0.262235054572 262 618 432 100
    0.272783666823
GO:0000166 nucleotide binding 2641 1570
    0.257119002375 261 635 415 101
    0.284727627745
GO:0008270 zinc ion binding 2532 1489
    0.252428297493 262 645 405 100
    0.295650992308
GO:0004872 receptor activity 2025 1535
    0.271277433846 250 628 422 112
    0.25240601164
```

For significant GO terms, obtained either with $\ell_1\ell_{2FS}$ or RLS, a list of *properly selected* variables is generated. The following details are noted for each properly selected variable: its corresponding probeset symbol, gene symbol (verified on the *ANNO* and *HGNC* tables), its corresponding Entrez Gene ID and Genbank record ID, as well as the frequency checked across outer cross–validation loop (if obtained with $\ell_1\ell_{2FS}$). For $\ell_1\ell_{2FS}$, the variables list is generated for the GO term only if this term is considered significant for that particular $\ell_1\ell_{2FS}$ μ value. For RLS, the details are similar, only frequency is omitted, since variable selection is not performed. Note that some corresponding external IDs may not be present. An abbreviated example output list of properly selected variables is presented below:

```
223055_s_at XPO5 57510 AF271159 100
201475_x_at MARS 4141 NM_004990 100
215208_x_at RPL35A 6165 AK021571 100
200079_s_at KARS 3735 AF285758 100
212160_at XPOT 11260 AI984005 100
204283_at FARS2 10667 NM_006567 75
223076_s_at NSUN2 54888 BC001041 75
223015_at EIF2A 83939 AF212241 50
201139_s_at SSB 6741 NM_003142 50
202541_at BF589679 25
238760_at YARS 8565 AW452122 25
201000_at AARS 16 NM_001605 25
```

Regarding variables, two histograms are also constructed. In the first case, for each variable present in original data set, it is counted how many times that variable was selected in data subsets correspoding to significant GO terms. In the second case, also for each variable in original data set, it is counted how many times that variable was *not* selected in data subsets correspoding to significant GO terms. In both cases, the following details are recorded for each variable: its corresponding probeset ID, gene symbol, and the respective count for that variable. Also noted are: the total number of entries in the histogram, the number of significant GO terms, and the number of GO terms in the GO domain considered. For $\ell_1\ell_{2FS}$–related data subsets, we produce one histogram for each μ value. For RLS–related data subsets, we produce single histogram for selected variables, since for corresponding significant GO terms *all* variables are properly selected. An abbreviated sample histogram of variables selected at least once is presented below:

```
Selected at least once: 4334 Nodes passing
TS error: 154 Nodes in BP:18029
1767_s_at TGFB3 14
1852_at TNF 12
2067_f_at BAX 10
1998_i_at BAX 10
448_s_at MEN1 9
33440_at ZEB1 9
```

Besides interpretable output, KDVS also collects technical output, such as binary files, additional diagnostic files and logs.

KDVS: Framework Architecture

The architecture of the framework is presented in Figure 3. All the involved applications work on a single *ensemble* of information, that consists of a relational database and filesystem objects. The ensemble is first obtained by *experiment.py*, and then modified by subsequent applications.

The default relational database engine, available for Python *SQLite* [53], is used due to its simplicity of usage, low memory footprint and low maintenance needs. Each database can contain many tables and is stored as single file. Currently, a single database is created for each experiment.

Applications in KDVS

Each application in KDVS is composed as a sequence of *actions* that are executed within specified *execution environment*. Actions are Python procedures. In this case, actions can use *shared variables*, maintained by the environment, to transfer their state to other actions. The most basic execution environment provides uniform logging and error handling; more complicated environments can provide access to distributed resources, interface to external applications etc.

Applications use KDVS *API* to perform their tasks: some functionalities shared by all applications, as well as some specific functionality considered stable are placed into separate Python package `kdvs`.

experiment.py

The *experiment.py* application generates the ensemble, performs local data integration and knowledge discovery. The general activity schema is presented in Figure 4.

We used `L1L2Py` [54], a Python implementation of $\ell_1\ell_{2FS}$, that contains also an implementation of RLS (note that we use RLS with regularization parameter equal to 0). Since $\ell_1\ell_{2FS}$ is quite time–consuming because it performs a full model selection, *experiment.py* manages a simple environment for distributed computations. The environment consists of a group of multi–core machines, connected over a local network, that accept and execute computational tasks. Each *split*, performed by $\ell_1\ell_{2FS}$, is a single task that is distributed and executed. The environment utilizes *shared storage*, a dedicated machine that exposes storage device available over the local network.

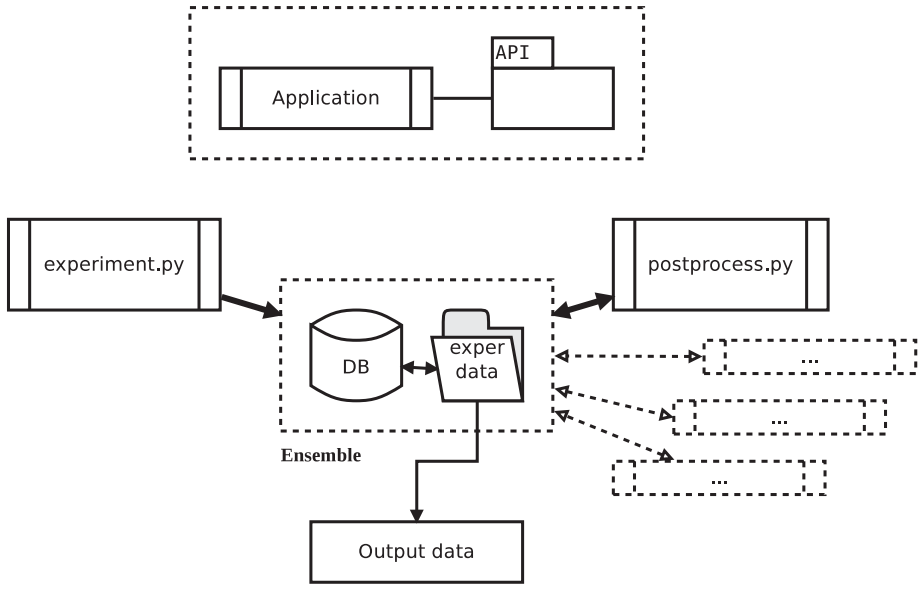

Figure 3 Schema of the architecture of KDVS. KDVS consists of applications that work on common ensemble of data. Each application uses the KDVS API component.

The environment, as a whole, is managed by the PPlus Python library [44], that encapsulates the access to shared storage as simple file operations, and controls distribution and execution of computational tasks. PPlus was built on top of Parallel Python (PP) [55], that provides a simple solution for managing local group of machines that execute Python code in tasks.

To speed up the process of building the ensemble, only the data that are crucial for execution of computational tasks are put on shared storage and accessed from there. The relational database is managed uniquely on the machine that started *experiment.py* application. Later, the data from the shared storage are copied back to the same location as database file and the ensemble is considered complete.

postprocess.py

The *postprocess.py* application works on the completed ensemble to identify the functional output of the framework and to generate useful miscellaneous data for the experiment. The general activity schema is presented in Figure 5.

Since this application does not have high time requirements, it uses very simple execution environment.

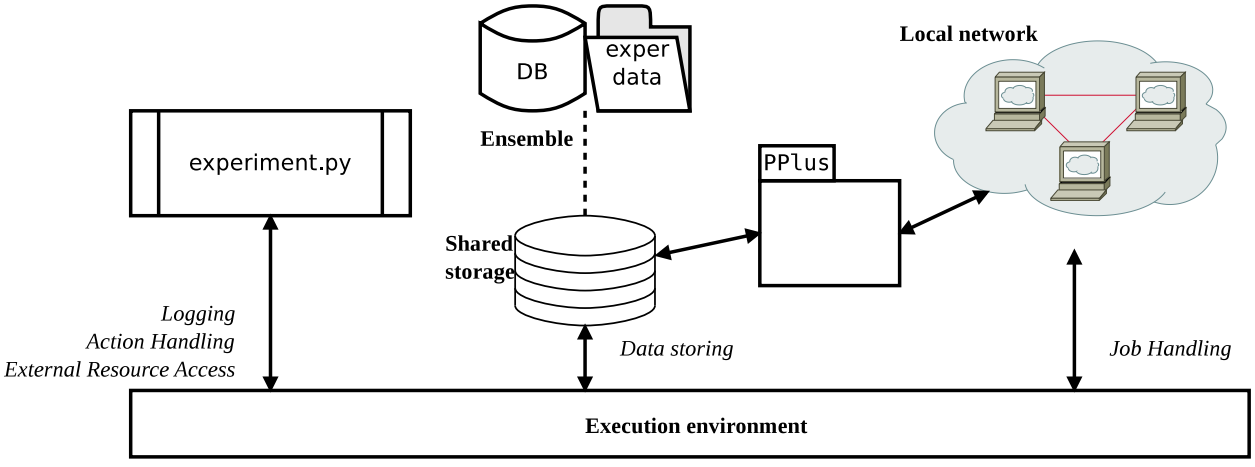

Figure 4 Activity schema of experiment.py application. The experiment.py application creates ensemble of data, performs data integration and transformation, manages distributed computational environment and performs knowledge discovery procedure.

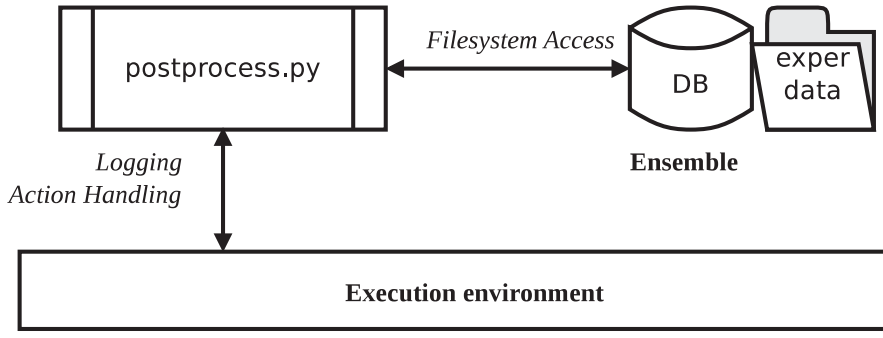

Figure 5 Activity schema of postprocess.py application. The postprocess.py application performs supplementary activities on ensemble of data (e.g. assembling final results, collecting useful statistics).

Additional scripts

Some independent R and Python scripts are used with the KDVS framework, to perform functionalities that are either not easily accomplished with Python alone or are considered the standard in its domain.

To perform normalization of Affymetrix microarray data, standalone R script is used (see Additional file 2 for an example). It uses standard BioConductor [33] normalization methods (i.e. RMA, GCRMA, aroma) and is tailored separately for each biological experiment.

A standalone R script is used (see Additional file 3 for an example) to perform visualization of the minimal subgraph of GO graph. This script utilizes `GOstat` and `RGraphViz` packages and is tailored separately for the specific list of GO terms obtained in the experiment.

To perform semantic clustering, the standalone `fastsemsim` Python package script (see Additional file 4 for an example) is used. It is tailored separately for the specific list of GO terms obtained in the experiment.

Enrichment analysis

For comparison purposes, KDVS results were compared to classic enrichment–based approaches [25,26]. To this aim, we used WebGestalt, an online gene set analysis toolkit [56] taking as input a list of relevant genes/probesets and performing an enrichment analysis in KEGG and GO, identifying the most relevant pathways and ontologies in the signatures.

Benchmark list

In order to evaluate the biological soundness of the results, we identified two benchmark lists, namely genes and corresponding GO terms, that refer to the current state of knowledge for the disease of interest. Together, they provide a reference way to evaluate the biological consistency of our results.

The benchmark gene list is the union of three lists available from public repositories, namely KEGG PATHWAY [57], KEGG DISEASE [57], and the Gene Prospector tool [58]. The first two sources refer to experimentally validated results, while the last one contains also the results of up–to–date literature mining of genetic association studies. We chose those prior knowledge sources because they are explicitly associated with the investigated diseases.

The benchmark list of GO terms was obtained by identifying associated GO terms, according to Gene Ontology Annotation (GOA) data [27], for each gene contained in the benchmark gene list. We chose this method because, to our knowledge, there are no established sources of GO terms that correspond to specific diseases.

The results produced by enrichment–based approach and KDVS are based on different methodologies, even if the same discovery method – $\ell_1\ell_{2FS}$ – is used. Therefore, instead of comparing them directly, we chose to make pairwise comparisons between the list of genes and list of discriminant GO terms, produced by each approach, and the respective benchmark lists.

In case of gene lists, the benchmark coverage is calculated for all benchmark genes. In case of GO term lists, however, it is necessary to acknowledge the fact that KDVS produces the results only for single GO domain at the time. Therefore, during the calculation of the benchmark coverage of GO terms, we recognize two cases. For enrichment–based approach, we consider all benchmark GO terms, while for KDVS we consider only the benchmark terms from that specific GO domain used in the experimental setting.

All benchmark lists used in this work are available as Additional file 5. The detailed description of the workflow followed to make the benchmark data is provided as Additional file 6.

Results and discussion

Prostate cancer study

We applied KDVS to analyze the GSE6919 [59,60] microarray dataset available in GEO. The dataset contains

gene expression data from metastatic, primary tumor and control prostate tissue samples processed on a platform measuring the expression of about 10.000 genes. After the normalization and the quality control steps, performed using the *aroma* package and the *arrayQualityMetrics* R packages [33], we analyzed 25 metastatic, 63 primary tumor and 80 normal tissue samples. We addressed two classification problems: one concerning the characterization of metastasis and primary tumor and one discriminating between primary tumor and normal tissue.

In preliminary results [31], only the first classification problem was addressed using Molecular Function (MF) as source of GO domain knowledge.

Successively each classification problem was addressed twice, considering as prior knowledge MF and Biological Process (BP) GO domains. Those GO terms whose classification error was below 30% were considered discriminant for both GO domains. The genes associated with these terms were merged and redundant genes were discarded. For each classification problem, the framework identified two lists of GO terms (one for MF and one for BP) and two gene lists, respectively. In addition, we addressed the same classification problems using classic enrichment–based approach (see Table 1). In this case, in the binary classification problem comparing primary tumor and metastasis, the 4–fold cross–validation error was 1%. In the other problem, the 5–fold cross–validation error was 27%.

In order to evaluate the biological soundness of the results, we identified two benchmark lists, namely genes and the corresponding GO terms, that refer to the current state of knowledge for prostate cancer (see Additional files 5 and 6). The benchmark gene list consists of 851 elements. The benchmark GO term list consists of 2437 BP terms and 824 MF terms, 3593 terms in total.

The comparison of our results with the benchmark lists shows the better performance of KDVS, as shown in Table 1. High percentage of genes and GO terms identified by KDVS were already known to be associated with the disease. Therefore, these results increase the likelihood that the remaining identified knowledge could be related with the biological issue under investigation.

Parkinson's disease study I

In another case study, first presented in [30], we analyzed the microarray dataset GSE20295 [61,62], also available from GEO. It is composed of Parkinson's disease (PD) samples in advanced stage of development and control sample tissues, 40 and 53 cases respectively, measured on a platform that includes about 33.000 genes. The MF domain of GO was chosen as source of prior knowledge to decompose GEDM into data subsets, associated with their relative GO terms. The terms whose classification error was below 30%, were considered as discriminant between the two classes. As reported in [30], these terms were found to be correlated with the disease under study, and their representation as subgraph provided a way to better visualize the results. In addition, we addressed the same classification problem using the classic enrichment–based approach (see Table 2). In this case, the 8–fold cross–validation error was 21%.

Both the list of GO terms and the corresponding list of genes were successfully compared with the benchmark lists for PD, that contain 2121 GO terms and 444 genes respectively. The benchmark lists were obtained following the same procedure as described in Prostate cancer study (see Additional files 5 and 6).

Table 1 Results of the KDVS for Prostate cancer study

Kind	Enrichment		KDVS			
Lists	**PTvsM**	**PTvsN**	**PTvsM MF**	**PTvsM BP**	**PTvsN MF**	**PTvsN BP**
Discr. GO Terms	60	120	1115	2242	320	689
Discr. Genes	59	389	3619	4457	3118	3271
Comm. GO Terms	27	61	375	1000	158	378
Comm. Genes	7	51	418	504	334	371
Bench. Cov. GO Terms	1%	2%	46%	41%	19%	16%
Bench. Cov. Genes	1%	6%	49%	59%	39%	44%

In the first two rows, the table shows the discriminant gene and GO term lists identified by enrichment–based approach and KDVS for each classification problem solved: Primary Tumor (PT) versus Metastasis (M) and Primary Tumor vs Normal (N). For KDVS, in addition, each classification task was addressed two times, based on the GO domain utilized, Molecular Function (MF) or Biological Process (BP). Discr. stands for discriminant.
In the next two rows, the table shows the gene and GO term lists obtained as an intersection between respective discriminant lists and benchmark lists, built for prostate cancer. Comm. means common.
In the last two rows, the table shows the coverage of benchmark gene and GO term lists with respective discriminant lists. For KDVS, the coverage was calculated only for benchmark MF and BP terms, respectively, according to GO domain utilized in classification task. Benchmark gene list consists of 851 elements. Benchmark GO term list consists of 2437 BP terms and 824 MF terms, 3593 terms in total. Bench. cov. means benchmark coverage.

Table 2 Results of the KDVS for Parkinson disease study I & II

Kind	Enrichment		KDVS	
Lists	**PDvsN I**	**PDvsN II**	**PDvsN I**	**PDvsN II**
Discr. GO terms	77	65	364	150
Discr. Genes	77	66	5705	4286
Comm. GO Terms	31	31	113	54
Comm. Genes	9	3	274	196
Benchmark Cov. GO Terms	2%	1%	25%	12%
Benchmark Cov. Genes	2%	1%	62%	44%

In the first two rows, the table shows the discriminant gene and GO term lists identified by enrichment–based approach and KDVS for Parkinson study I and II, respectively. Both studies were performed on two different microarray datasets of Parkinson (PD) and Normal (N) tissue samples. Discr. means discriminant.
In the next two rows, the table shows the gene and GO term lists obtained as an intersection between respective discriminant lists and benchmark lists, built for Parkinson's disease. Comm. means common.
In the last two rows, the table shows the coverage of benchmark gene and GO term lists with respective discriminant lists. For KDVS, the coverage was calculated only for benchmark MF terms, according to the nature of classification task. Benchmark gene list consists of 444 elements. Benchmark GO term list consists of 2121 terms in total, including 446 MF terms. Bench. cov. means benchmark coverage.

As shown in Table 2, significant number of genes and GO terms identified by KDVS were already known to be associated with PD. In addition, the KDVS results show greater coverage of benchmark data with respect to the enrichment–based approach.

Parkinson's disease study II

The last case study regards the analysis of GSE6613, a microarray dataset [63], available from GEO. The dataset is composed of 33 early stage Parkinson samples and 22 controls, measured on a platform that includes about 33.000 genes. The MF domain of GO was chosen as source of prior knowledge to decompose GEDM into data subsets, associated with their relative GO terms. Also in this study, the terms whose classification error was below 30%, were considered as discriminant between the two classes. In addition, we addressed the same classification problem using the classic enrichment–based approach (see Table 2). In this case, the 8–fold cross–validation error was 36%.

For comparison, we used the same benchmark lists as for Parkinson's disease study I.

Table 2 shows that both approaches were able to extract sound biological knowledge, nonetheless the results obtained from KDVS underline a higher coverage of the benchmark knowledge, with respect to those obtained by the classic enrichment–based approach.

KDVS conceptual framework

KDVS pipeline presented here is one instance of a conceptual framework that integrates prior knowledge with standard machine-learning-based selection procedures.

In [64] a similar method was proposed, evaluating the predictive performance of functional categories based on Leave-One-Out Regularized Least Squares (LOO-RLS). The authors also aimed at assessing the statistical significance of each predictor exploiting the closed form solution for LOO-RLS and a multiple random validation strategy. Despite presenting a very similar concept, KDVS adds the feature selection phase combined with the prediction assessment. This is particularly useful especially when the number of variables is very high with respect to the number of samples, which is a very common situation for a large number of GO terms.

In particular, we considered $\ell_1\ell_{2FS}$ as selection method. Nevertheless, KDVS is, by design, flexible to incorporate the selection method of choice: $\ell_1\ell_{2FS}$ could be replaced by any other *embedded or wrapper* variable selection method based on classification [65,66] or *filter* methods combined with a prediction step [67].

Finally, it is worth remarking that KDVS aims at using the existing knowledge as additional information to structure the data matrix before the analysis step, while enrichment methods such as GSEA, Random sets, GLAPA and others [13,68] incorporate the domain knowledge *a posteriori*.

Conclusions

The KDVS framework has been developed to demonstrate the advantages of using prior biological knowledge before identifying the list of GO terms and genes statistically discriminant between two classes in the analysis of high-throughput data. Therefore it is presented as an alternative method to the most frequently used approaches that use prior knowledge a posteriori, once the gene signature has been identified. Its modular structure allows quick adaptation to different HT data types, prior knowledge sources, and different classification and variable selection approaches. The usage of distributed resources overcomes demanding computational requirements that are common

in HT data analysis (microarray, NGS, etc.). Furthermore the conceptual integration of the visualization approach in the framework activities determines greater interpretability of the results, especially for life science researchers.

In the future, we plan to make tighter integration with existing semantic clustering approach, to introduce different sources of prior biological knowledge, to accept different HT data, and to introduce wide range of classification and variable selection techniques.

Additional files

Additional file 1: Source code of KDVS. Format: ZIP. It contains the Python source code, the documentation, and the internal data files.

Additional file 2: Example script performing microarray data normalization and quality check. Format: R. The proper input directory and chip type must be provided. The script uses aroma folder structure to identify input data.

Additional file 3: Example script producing visualization artifacts. Format: R. The proper input directory and GO domain must be specified. The script scans KDVS ensemble for postprocessing data and visualizes minimal subgraphs for the list of selected GO terms.

Additional file 4: Example script performing semantic clustering. Format: Python. The script accepts single DSV output files generated by KDVS, containing list of significant GO terms, and performs semantic clustering with *fastsemsim* Python library. Proper data files used by *fastsemsim*, as well as clustering details, must be provided. The clustering is performed with default settings, using *scipy.cluster.hierarchy.fcluster*. For more details see *fastsemsim* documentation.

Additional file 5: Benchmark lists for Prostate Cancer and PD. Format: XLS. The spreadsheet contains the benchmark lists of genes and GO terms, considered trustable enough to use in comparisons between enrichment–based results and KDVS results. See Additional file 6 for more information.

Additional file 6: Detailed description of the construction of benchmark data.

Abbreviations

HT: high-throughput; KDVS: Knowledge Driven Variable Selection; DEG: differentially expressed genes; GO: Gene Ontology; API: Application Programming Interface; GEDM: Gene Expression Data Matrix; DSV: Delimiter Separated Values; GEO: Gene Expression Omnibus; DAG: Directed Acyclic Graph; RDF: Resource Description Framework; XML, Extensible Markup Language; HGNC: HUGO Gene Nomenclature Committee; RLS: Regularized Least Squares; MCC: Matthews Correlation Coefficient; FS: Feature Selection; PP: Parallel Python; GOA: Gene Ontology Annotations; MF: Molecular Function; BP: Biological Process; PD: Parkinson's disease; LOO-RLS: Leave-One-Out-Regularized Least Squares; NGS: next–generation sequencing.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

GZ conceived, designed, and implemented the KDVS framework. AB designed the study, developed statistical methodology, and supervised KDVS development. MS provided biological insight. TS and BDC contributed to the ideas of prior knowledge usage and of semantic clustering. AV developed statistical methodology and supervised the study. All authors have contributed to the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We thank Salvatore Masecchia for the development of L1L2Py and PPlus.

Author details

[1]DIBRIS, University of Genoa, via Dodecaneso 35, I-16146 Genova, Italy. [2]Information Engineering Department, University of Padova, via Gradenigo 6A I-35131 Padova, Italy.

Received: 28 September 2012 Accepted: 13 December 2012
Published: 9 January 2013

References

1. Brown P, Botstein D: **Exploring the new world of the genome with DNA microarrays.** *Nat Genet* 1999, **21:**33–37.
2. Shendure J, Ji H: **Next–generation DNA sequencing.** *Nat Biotech* 2008, **26**(10):1135–1145.
3. Golub T, Slonim D, Tamayo P, Huard C, Gaasenbeek M, Mesirov J, Coller H, Loh M, Downing J, Caligiuri M, Bloomfield C, Lander E: **Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.** *Science* 1999, **286**(5439):531–537.
4. Guyon I, Weston J, Barnhill S, Vapnik V: **Gene selection for cancer classification using support vector machines.** *Mach Learn* 2002, **46:**389–422.
5. Bailey R: *Design of Comparative Experiments. Cambridge Series in Statistical and Probabilistic Mathematics*. New York: Cambridge University Press; 2008.
6. Irizarry R, Wang C, Zhou Y, Speed T: **Gene set enrichment analysis made simple.** *Stat Methods Med Res* 2009, **18**(6):565–575.
7. Subramanian A, Tamayo P, Mootha V, Mukherjee S, Ebert B, Gillette M, Paulovich A, Pomeroy S, Golub T, Lander E, Mesirov J: **Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles.** *Proc Natl Acad Sci USA* 2005, **102**(43):15545–15550.
8. Nadon R, Shoemaker J: **Statistical issues with microarrays: processing and analysis.** *Trends Genet* 2002, **18**(5):265–271.
9. Cui X, Churchill G: **Statistical tests for differential expression in cDNA microarray experiments.** *Genome Biol* 2003, **4**(4):210.
10. Mootha V, Lindgren C, Eriksson K, Subramanian A, Sihag S, Lehar J, Puigserver P, Carlsson E, Ridderstrale M, Laurila E, Houstis N, Daly M, Patterson N, Mesirov J, Golub T, Tamayo P, Spiegelman B, Lander E, Hirschhorn J, Altshuler D, Groop L: **PGC-1α-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes.** *Nat Genet* 2003, **34**(3):267–273.
11. Huang D, Sherman B, Stephens R, Baseler M, Lane C, Lempicki R: **DAVID gene ID conversion tool.** *Bioinformation* 2008, **2**(10):428–430.
12. van Iersel M, Pico A, Kelder T, Gao J, Ho I, Hanspers K, Conklin B, Evelo C: **The BridgeDb framework: standardized access to gene, protein and metabolite identifier mapping services.** *BMC Bioinformatics* 2010, **11:**5.
13. Huang D, Sherman B, Lempicki R: **Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists.** *Nucleic Acids Res* 2009, **37:**1–13.
14. Hung J, Yang T, Hu Z, Weng Z, DeLisi C: **Gene set enrichment analysis: performance evaluation and usage guidelines.** *Brief Bioinforma* 2011, **13:**281–291.
15. Chuang H, Lee E, Liu Y, Lee D, Ideker T: **Network–based classification of breast cancer metastasis.** *Mol Syst Biol* 2007, **3**(140).
16. Rapaport F, Zinovyev A, Dutreix M, Barillot E, Vert J: **Classification of microarray data using gene networks.** *BMC Bioinform* 2007, **8:**35.
17. Li C, Li H: **Network–constrained regularization and variable selection for analysis of genomic data.** *Bioinformatics* 2008, **24**(9): 1175–1182.
18. Yousef M, Ketany M, Manevitz L, Showe L, Showe M: **Classification and biomarker identification using gene network modules and support vector machines.** *BMC Bioinform* 2009, **10:**337.
19. Tai F, Pan W: **Incorporating prior knowledge of predictors into penalized classifiers with multiple penalty terms.** *Bioinformatics* 2007, **23**(14):1775–1782.
20. Binder H, Schumacher M: **Incorporating pathway information into boosting estimation of high–dimensional risk prediction models.** *BMC Bioinform* 2009, **10:**18.
21. Chen X, Wang L: **Integrating biological knowledge with gene expression profiles for survival prediction of cancer.** *J Comput Biol* 2009, **16**(2):265–278.

22. Sanavia T, Aiolli F, Da San Martino G, Bisognin A, Di Camillo, B: **Improving biomarker list stability by integration of biological knowledge in the learning process.** *BMC Bioinform* 2012, **13**(Suppl 4):S22.
23. Hoheisel J: **Microarray technology: beyond transcript profiling and genotype analysis.** *Nat Rev Genet* 2006, **7**(3):200–210.
24. Ma S, Huang J: **Penalized feature selection and classification in bioinformatics.** *Brief Bioinform* 2008, **9**(5):392–403.
25. Fardin P, Barla A, Mosci S, Rosasco L, Verri A, Varesio L: **The l1-l2 regularization framework unmasks the hypoxia signature hidden in the transcriptome of a set of heterogeneous neuroblastoma cell lines.** *BMC Genomics* 2009, **10:**474.
26. Squillario M, Barla A: **A computational procedure for functional characterization of potential marker genes from molecular data: Alzheimer's as a case study.** *BMC Med Genomics* 2011, **4:**55.
27. Ashburner M, Ball C, Blake J, Botstein D, Butler H, Cherry J, Davis A, Dolinski K, Dwight S, Eppig J, Harris M, Hill D, Issel-Tarver L, Kasarskis A, Lewis S, Matese J, Richardson J, Ringwald M, Rubin G, Sherlock G: **Gene Ontology: tool for the unification of biology. The Gene Ontology Consortium. .** *Nat Genet* 2000, **25:**25–29.
28. **Python Programming Language – Official Website.** [http://www.python.org]
29. Squillario M, Masecchia S, Zycinski G, Barla A: **Uncovering Candidate Biomarkers for Alzheimer's and Parkison's Diseases with Regularization Methods and Prior Knowledge.** *Neuro-Degenerative Diseases - Proc AD/PD 2011, Barcelona, Spain* 2011, **8**(Supp 1).
30. Zycinski G, Barla A, Verri A: **SVS: Data and knowledge integration in computational biology.** In *Engineering in Medicine and Biology Society, EMBC , 2011 Annual International Conference of the IEEE*. Boston, MA: IEEE; 2011:6474–6478 .
31. Zycinski G, Squillario M, Barla A, Sanavia T, Verri A, Di Camillo B: **Discriminant functional gene groups identification with machine learning and prior knowledge.** In *ESANN 2012*. Edited by Verleysen M. Louvain–la–Neuve, Belgium: Ciaco; 2012:221–226.
32. Draghici S: *Statistics and Data Analysis for Microarrays Using R and Bioconductor*. New York: Chapman & Hall/CRC Mathematical & Computational Biology, Chapman and Hall/CRC; 2011.
33. Gentleman R, Carey V, Bates D, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini A, Sawitzki G, Smith C, Smyth G, Tierney L, Yang J, Zhang J: **Bioconductor: Open software development for computational biology and bioinformatics.** *Genome Biol* 2004, **5:**R80.
34. Edgar R, Domrachev M, Lash A: **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository.** *Nucleic Acids Res* 2002, **30:**207–10.
35. Hastie T, Tibshirani R, Friedman J: *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer-Verlag; 2009.
36. Seal R, Gordon S, Lush M, Wright M, Bruford E: **genenames.org: the HGNC resources in 2011.** *Nucleic Acids Res* 2011, **39**(suppl 1):D514—D519.
37. De Mol C, De Vito E, Rosasco L: **Elastic–net regularization in learning theory.** *J Complex* 2009, **25:**201–230.
38. De Mol C, Mosci S, Traskine M, Verri A: **A regularized method for selecting nested groups of relevant genes from microarray data.** *J Comput Biol* 2009, **16:**1–15.
39. Barla A, Mosci S, Rosasco L, Verri A: **A method for robust variable selection with significance assessment.** In *Proceedings of ESANN 2008* . Edited by Verleysen M. Brussels, Belgium: D–side; 2008:83–88.
40. Di Camillo B, Sanavia T, Martini M, Jurman G, Sambo F, Barla A, Squillario M, Furlanello C, Toffolo G, Cobelli C: **Effect of size and heterogeneity of samples on biomarker discovery: synthetic and real data assessment.** *PLoS ONE* 2012, **7**(3):e32200.
41. Barrett J, Kawasaki E: **Microarrays: the use of oligonucleotides and cDNA for the analysis of gene expression.** *Drug Discov Today* 2003, **8**(3):134–141.
42. Jaksik R, Polanska J, Herok R, Rzeszowska-Wolny J: **Calculation of reliable transcript levels of annotated genes on the basis of multiple probe-sets in Affymetrix microarrays.** *Acta Biochimica Polonica* 2009, **56**(2):271–277.
43. Zomaya A: *Parallel Computing for Bioinformatics and Computational Biology: Models, Enabling Technologies, and Case Studies*. Hoboken, NJ, Wiley–Interscience: Wiley Series on Parallel and Distributed Computing; 2006.
44. **PPlus Home Page.** [http://slipguru.disi.unige.it/Software/PPlus/]
45. Maglott D, Ostell J, Pruitt K, Tatusova T: **Entrez Gene: gene-centered information at NCBI.** *Nucleic Acids Res* 2005, **33**(suppl 1):D54—D58.
46. Benson D, Karsch-Mizrachi I, Lipman D, Ostell J, Wheeler D: **GenBank.** *Nucleic Acids Res* 2005, **33**(suppl 1):D34—D38.
47. Herman I, Melancon G, Marshall M: **Graph visualization and navigation in information visualization: a survey.** *Vis Comput Graphics, IEEE Trans* 2000, **6:**24–43.
48. Katifori A, Halatsis C, Lepouras G, Vassilakis C, Giannopoulou E: **Ontology visualization methods–a survey.** *ACM Comput Surv* 2007, **39**(4). http://dl.acm.org/citation.cfm?id=1287621.
49. Ellson J, Gansner E, Koutsofios L, North S, Woodhull G: **Graphviz–open source graph drawing tools.** In *Lecture Notes in Computer Science*. Berlin Heidelberg: Springer-Verlag ; 2001:483–484.
50. Lord P, Stevens R, Brass A, Goble C: **Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation.** *Bioinformatics* 2003, **19**(10): 1275–1283.
51. Guzzi P, Mina M, Guerra C, Cannataro M: **Semantic similarity analysis of protein data: assessment with biological features and issues.** *Brief Bioinforma* 2012, **13**(5):569–585.
52. **FastSemSim Home Page.** [http://sourceforge.net/projects/fastsemsim/]
53. **SQLite Home Page.** [http://www.sqlite.org/]
54. **L1L2Py Home Page.** [http://slipguru.disi.unige.it/Software/L1L2Py/]
55. **Parallel Python Home Page.** [http://www.parallelpython.com/]
56. Zhang B, Kirov S, Snoddy J: **WebGestalt: an integrated system for exploring gene sets in various biological contexts.** *Nucleic Acids Res* 2005, **33**(suppl 2):W741—W748.
57. Kanehisa M, Goto S, Sato Y, Furumichi M, Tanabe M: **KEGG for integration and interpretation of large-scale molecular data sets.** *Nucleic Acids Res* 2012, **40**(D1):D109—D114.
58. Yu W, Wulf A, Liu T, Khoury M, Gwinn M: **Gene Prospector: An evidence gateway for evaluating potential susceptibility genes and interacting risk factors for human diseases.** *BMC Bioinform* 2008, **9:**528.
59. Yu Y, Landsittel D, Jing L, Nelson J, Ren B, Liu L, McDonald C, Thomas R, Dhir R, Finkelstein S, Michalopoulos G, Becich M, Luo J: **Gene expression alterations in prostate cancer predicting tumor aggression and preceding development of malignancy.** *J Clin Oncol* 2004, **22**(14):2790–2799.
60. Chandran U, Ma C, Dhir R, Bisceglia M, Lyons-Weiler M, Liang W, Michalopoulos G, Becich M, Monzon F: **Gene expression profiles of prostate cancer reveal involvement of multiple molecular pathways in the metastatic process.** *BMC Cancer* 2007, **7:**64.
61. Zhang Y, James M, Middleton F, Davis R: **Transcriptional analysis of multiple brain regions in Parkinson's disease supports the involvement of specific protein processing, energy metabolism, and signaling pathways, and suggests novel disease mechanisms.** *Am J Med Genet Part B: Neuropsychiatric Genet* 2005, **137B:**5–16.
62. Zheng B, Liao Z, Locascio J, Lesniak K, Roderick S, Watt M, Eklund A, Zhang-James Y, Kim P, Hauser M, Grünblatt E, Moran L, Mandel S, Riederer P, Miller R, Federoff H, Wüllner U, Papapetropoulos S, Youdim M, Cantuti-Castelvetri I, Young A, Vance J, Davis R, Hedreen J, Adler C, Beach T, Graeber M, Middleton F, Rochet J, Scherzer C: **the Global PD Gene Expression (GPEX) Consortium: PGC-1α, A potential therapeutic target for early intervention in Parkinson's disease.** *Sci Transl Med* 2010, **2**(52):52–73.
63. Scherzer C, Eklund A, Morse L, Liao Z, Locascio J, Fefer D, Schwarzschild M, Schlossmacher M, Hauser M, Vance J, Sudarsky L, Standaert D, Growdon J, Jensen R, Gullans S: **Molecular markers of early Parkinson's disease based on gene expression in blood.** *Proc Natl Acad Sci* 2007, **104**(3):955–960.
64. Maglietta R, Piepoli A, Catalano D, Licciulli F, Carella M, Liuni S, Pesole G, Perri F, Ancona N: **Statistical assessment of functional categories of genes deregulated in pathological conditions by using microarray data.** *Bioinformatics* 2007, **23**(16):2063–2072.
65. Guyon I, Elisseeff A: **An introduction to variable and feature selection.** *J Mach Learn Res* 2003, **3:**1157–1182.

66. Furlanello C, Serafini M, Merler S, Jurman G: **Entropy-based gene ranking without selection bias for the predictive classification of microarray data.** *BMC Bioinform* 2003, **4**(1):54.
67. Kohavi R, John GH: **Wrappers for feature subset selection.** *Artif Intell* 1997, **97:**273–324.
68. Abatangelo L, Maglietta R, Distaso A, D'Addabbo A, Creanza T, Mukherjee S, Ancona N: **Comparative study of gene set enrichment methods.** *BMC Bioinform* 2009, **10:**275.

doi:10.1186/1751-0473-8-2
Cite this article as: Zycinski *et al.*: Knowledge Driven Variable Selection (KDVS) – a new approach to enrichment analysis of gene signatures obtained from high–throughput data. *Source Code for Biology and Medicine* 2013 **8**:2.