CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Performance analysis of the Survival-SVM classifier applied to gene-expression databases

Genaro Camele[1] and Waldo Hasperué[1,2]

Instituto de Investigación en Informática (III-LIDI), Facultad de Informática
Universidad Nacional de La Plata.
Comisión de Investigaciones Científicas (CIC-PBA).
{gcamele, whasperue}@lidi.info.unlp.edu.ar

Abstract. The analysis of epigenetic information for the diagnosis and prognosis of patients has been gaining relevance in recent years due to the technological progress that entails a decrease in information extraction and processing costs. One of the tasks most commonly carried out in this area is obtaining models that allow using patient epigenetic information to make inferences about survival analysis. As a result, optimizing these models turns into a problem of great interest today. In this article, the evaluation of different metrics and execution times for the Survival Support Vector Machines model is carried out through survival analysis applied to gene expression databases. Different experiments were performed varying the number of genes used for training to measure the correlation between model performance and data growth. The results showed that linear and polynomial kernels offer a better balance between execution time and model predictive power when the number of genes to be evaluated is less than 2000, while the cosine and RBF kernels are better candidates otherwise.

Keywords: Survival analysis, Survival Support Vector Machines, Regression, Performance, Apache Spark.

1. Introduction

Functional genomics is responsible for the systematic collection of information on the function and interaction between genes and/or proteins. It allows an understanding of how the genome works as a whole, through the controlled expression of each one of its genes. In this area, gene expression profile analyses stand out; their main objective is identifying a group of genes whose expression pattern is associated with a specific phenotype, a concept known as gene expression signature, or simply signature. In medicine, signatures are particularly useful as a diagnostic, prognostic, or predictive biomarker of any given pathology. Biomarkers with prognostic value allow better stratification of patients according to their prognosis of disease progression regardless of a therapy, which opens the way to researching appropriate treatments for each defined patient category. The estimate of a biomarker's prognostic value is measured using survival analysis techniques.

Survival analysis consists of a series of statistical procedures for data analysis that study the time until an event occurs (such as patient death or tumor recurrence). The number of these procedures is large, and with the lower costs for data storage and processing and the technological advances in sequencing, there has been an exponential growth of data volumes, thus increasing the need for algorithms that meet the demand in an acceptable response time.

1.1 Motivation

The main reason for measuring the execution times of classifiers with survival data is their usefulness in Feature Selection (FS) algorithms for survival data. These algorithms are based on a gene expression dataset with prognostic, diagnostic, or predictive power in any pathology of interest. The objective of the FS is to obtain a subset of genes or molecules that have the same or better performance for the current problem. Due to the high computational cost that would require evaluating the entire solution space (i.e., all the combinations between all genes or molecules), population metaheuristics such as Binary Particle Swarm Optimization (BPSO) [1] or Binary Black Hole Algorithm (BBHA) [2] are used. These techniques intelligently traverse only a portion of the solution space until finding the subset that optimizes the fitness function, requiring much less time (knowing that an approximate solution is more likely to be found than the best solution).

The main problem is that the fitness function used by these metaheuristics involves the execution of classifiers that could require high computational power. For this reason, the parallelization and distribution of the classifier evaluation are essential to speed up the total execution time for the FS algorithm.

Multiomix [3] is a platform which allows researchers to upload datasets or use a public database (already integrated into the platform) to perform survival analysis. The development and measurement of classifiers such as Survival-SVM or Random Survival Forest [4] in distributed environments will allow the developer team to put these algorithms into production for free use by the community. Measuring the execution times for the different database models and size configurations allows defining a task distribution policy throughout the computer cluster (orchestrated by Apache Spark) to carry out the various experiments more effectively and efficiently. These optimizations are notably relevant for the community as several studies are available, including selecting the most appropriate model within a distributed environment, as in [5] where classifier models provided by Apache Spark MLib applied to FS techniques are compared. Or in [6], where models such as SurvivalSVM and Random Survival Forest are proposed to obtain a better C-Index than the Cox Proportional Hazards (CPH) offered by standard models for survival analysis in oncology studies. There are also algorithms focused on optimizing the training stage of the model without affecting the prognostic power against real data sets [7].

1.2 Survival Support Vector Machines

The Survival Support Vector Machines model [8][9] is an extension of the standard support vector machine for time and event data, with support for right-censored data, i.e., samples that do not present the event during a survival study. Its main advantage is that it considers complex and non-linear relationships between traits and survival. The kernel function of this model implicitly maps input features into highdimensional feature spaces that describe survival data with a hyperplane. That makes SVMs very versatile and applicable to a wide range of data.

Survival analysis in the context of support vector machines can be described in two different ways:

- As a ranking problem: the model learns to assign samples with shorter survival times a lower rank by considering all possible pairs of samples in the training data.
- As a regression problem: the model learns to directly predict log survival time.

The training data used to obtain such a model consists of a triplet (x_i, y_i, δ_i) where x_i is a d-dimensional feature vector, $y_i > 0$ is the survival or censoring time, and $\delta_i \in \{0, 1\}$ is the occurrence indicator for the event (0 = the event did not occur as is said to be censored, 1 = the event occurred). The model's objective will be to minimize the function presented in [8].

In this article, the experiments are limited to regression tasks only.

2. Experimentation

2.1 Hardware configuration and experiments

All experiments were carried out in an Apache Spark cluster made up of a single master node and three worker nodes. All four nodes had Ubuntu 20.04 LTS, an Intel(R) Core(TM) i3-4160 CPU @ 3.60GHz, and 8GB of RAM. As regards software, Spark 3.1.1 was used.

To avoid any bias in the results, a Cross Validation (CV) of 10 folds was performed. To obtain these folds, stratified sampling was used [10]. Additionally, the entire process of randomly selecting attributes, dividing into folds, training, and evaluating the models was performed 30 times with each type of kernel and optimizer available in the Scikit-Survival Python library. The kernels available for the SSVM algorithm are Linear, Cosine, Polynomial, and Radial Basis Function (RBF). As regards the optimizers, the library allows choosing between AVLTree and RBTree. The metric to be optimized is Harrell's concordance index (C-Index) [11][12], which is a suitable metric for survival analysis because it evaluates the predictive ability of models in terms of the relative order of survival times, considering both observed and censored events. By encompassing categorical and continuous data, and having a range of interpretable values between 0.5 and 1.0, the C-index allows comparing different models and selecting the most suitable one to predict events of interest over time, which makes it a versatile and widely accepted tool in medical and scientific research in general.

2.2 Algorithm and dataset used

The algorithm used to carry out the experiments is detailed below:

```
program SurvivalSVMExecution
parameters
    ds, {dataset to evaluate}
    n_individuals, {number of individuals}
begin
    total_n_f= get_dataset_n_features(ds)
    current_n_f = 10
    step = 100
```

```
    for iter in 1 .. 30 do
        while current_n_f <= total_n_f do
            for ind_idx in n_individuals do
                rand_features = get_rand_feat(ds,current_n_f)
                indivs[ind_idx] = rand_features
                if current_n_f > 100
                    current_n_f += step
                else
                    current_n_f = current_n_f + 10
            end
            results = compute_in_spark(indivs)
            store_results_in_csv(results)
        end
    end
end.
```

The dataset used for the experiments is the Breast Invasive Carcinoma (TCGA, PanCancer Atlas), which contains data from 1082 patients (931 samples without event occurrence and 151 samples with event definition) and 20014 genes [13].

As previously mentioned, 30 iterations are performed to achieve statistical significance. In each of them, *N* individuals (*indivs* variable) of the metaheuristic under study are computed (particles in the case of the BPSO algorithm, or stars in BBHA). Each of these individuals represents a random subset of features and computes prognostic power and other metrics from the assigned subset by the 10-folds CV.

The function *compute_in_spark* distributes the *N* individuals among the Workers of the Apache Spark cluster, where a Worker can receive several individuals with different subsets of genes to compute. Even so, the CV task is carried out on a single individual at a time to avoid variations in the time measurements and avoid bias when obtaining the real execution times with the parameters set.

The result of all computed individuals is expressed in the algorithm as results. This variable, where *N* is the number of individuals to be evaluated, consists of an *N*-tuple of the following data: the average fitness value (C-Index) for the every of the folds of the CV during the model training and validation phases, the time it took the Spark worker to compute a given agent (i.e., the entire CV process), the number of features evaluated by that worker, the time that took each Survival-SVM iteration, average testing time, number of iterations it took the SVM to converge, and other trace data within the cluster such as partition number within the Spark cluster and the hostname of the computer where the Spark Worker is operating.

3. Result and discussion

The results obtained after running the different gene subsets for 30 independent runs are discussed below.

Regarding the optimizers, the experiments were carried out with the AVLTree and RBTree structures. Both structures were found to impact required time for each iteration only during the training stage, thus not interfering with model fitness function. However, RBTree results were discarded because they solely led to a degradation in

training performance; for that reason, only AVLTree results were presented from now on.

As expected, the relationship between testing time and the number of features follows a linear trend (Fig. 1) since there is no error function to optimize in the process, and the computation of the inference only consists of the execution of the Survival-SVM kernel function.

The direct relationship between run time and number of features presents an abrupt change after 100 genes as very high run times are observed for the CV process evaluating a reduced set of genes. After that number the relationship starts to be more linear in nature with the Linear, Polynomial kernels, while in the case of the RBF and Cosine kernels, the difference is significant. The Cosine kernel presents an erratic runtime with high variance, stabilizing after 2500 genes, while RBF flattens out after 100 genes and varies with increasing features. All these details can be seen in Fig. 2.

Based on the total CV time, individual iteration times also vary for the same kernels that are inefficient with few genes (Fig. 3). However, for the Linear and Polynomial kernels, the curve is linear and it is consistent with the number of genes evaluated. Therefore, it is concluded that the model requires more iterations to converge when it has less information, explaining the high peaks in total execution time at the beginning of graphs in Fig. 2.

Regarding the C-Index values obtained during the training phase, all kernels suffer a degradation in performance as the number of features increases (Fig. 4). This may be because the dataset used is highly censored and, therefore, unbalanced: it consists of 86% of the samples without event occurrence (931 samples) and only 14% with event definition (151 samples). Furthermore, the fact that the data is not linearly separable explains the poor performance of the linear kernel.

Expanding the experiments by adjusting the C parameter of the Survival-SVM or changing the distribution function to norm-1, which is preferred with highdimensional data, could yield better results. Different techniques could also be applied to address the imbalance problem, as proposed in [14].

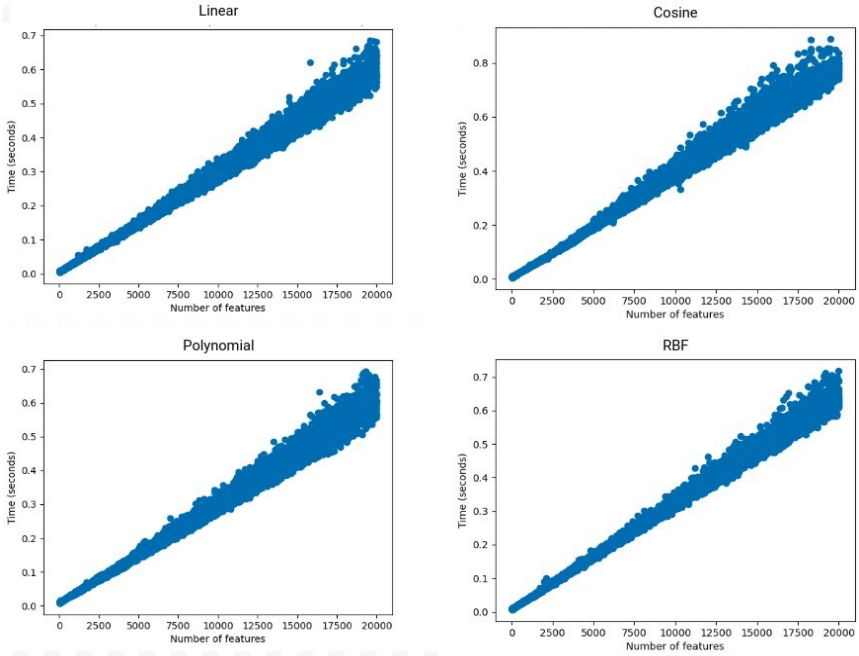

Fig. 1. Average test times during cross-validation for the Linear, Cosine, Polynomial, and RBF kernels.

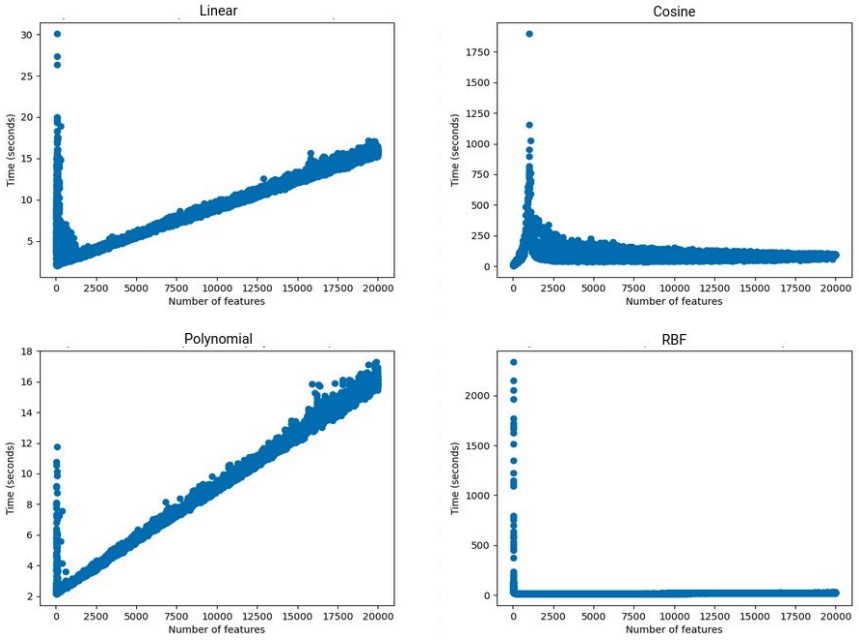

Fig. 2. Total execution time of the 10-fold CV process for Linear, Cosine, Polynomial, and RBF kernels.

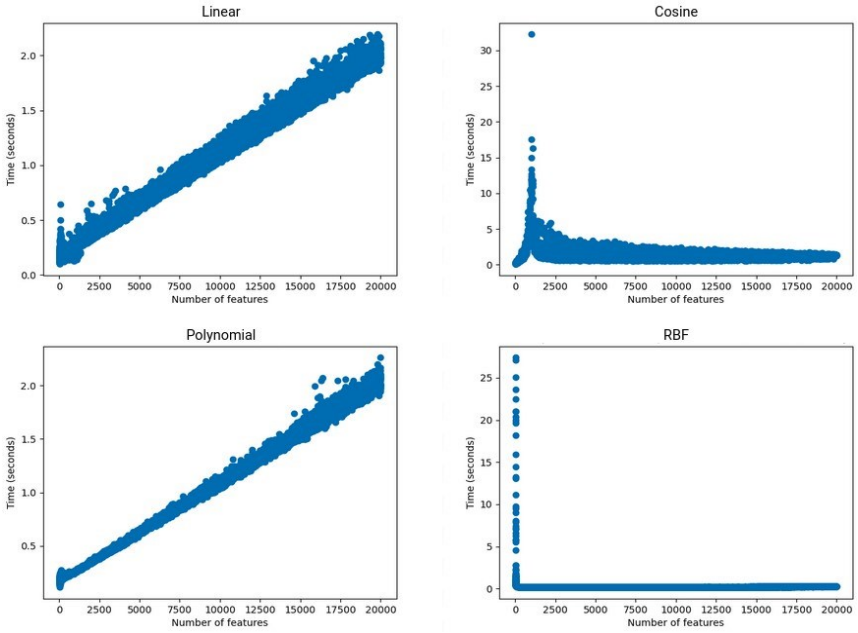

Fig. 3. Times per iteration for Linear, Cosine, Polynomial, and RBF kernels.

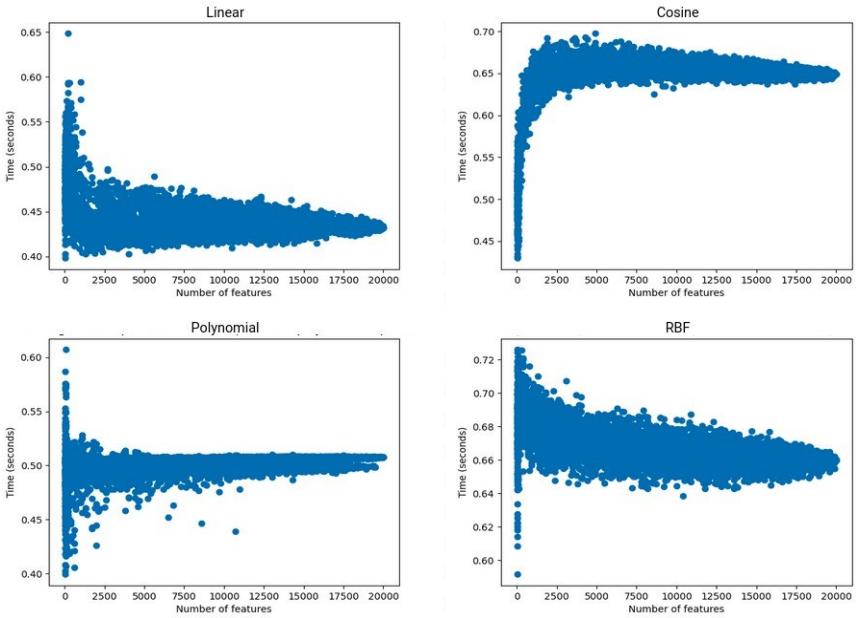

Fig. 4. C-Index obtained during the training phase of the Survival-SVM for Linear, Cosine, Polynomial, and RBF kernels.

4. Conclusion

The execution times and the C-Index obtained by the Survival-SVM model for different numbers of features in a gene dataset have been evaluated. The experiments were carried out with different kernels and optimizers, allowing a direct comparison between different configurations available.

Concluding, of the two available optimizers, RBTree generated a significant degradation in execution times during the training phase of the model in all cases. Regarding kernel configuration, worse time metrics were observed when the dataset had less than 100 features. Even so, iteration times remained directly correlated to dataset size for the Linear and Cosine kernels, making it clear that the high time required with few features is due to the number of iterations the model needs to reach convergence. It was not the case with the Polynomial and RBF kernels, whose times per iteration were also erratic for measurements with few training features.

The behavior of the model during the training stage was analyzed, and it was observed that the unbalanced nature of the data (a large amount of censored data) and the impossibility of separating them linearly resulted in worse metrics as the size of the dataset increased.

The results obtained show that, for subsets with few genes, the linear and polynomial kernels have a shorter execution time and an acceptable C-index. On the other hand, if the number of genes in the subset exceeds 2000, then the Cosine and RBF kernels obtain a better balance between execution time and prognostic power. Both tests show that it will be trivial to obtain models that predict the execution time of a model with N features, which will allow to improve task distribution in a distributed environment. Although the results presented in this work are indeed obtained from the analysis of a specific database, in the future it may include replicating the experiment in more databases to determine if there is variability.

The experiments were carried out on an Apache Spark cluster to reduce the time required to complete them, in addition to establishing a base configuration that will allow carrying these experiments to other models and continue advancing in the development of algorithms that will be used in production on the Multiomix platform for the application of FS techniques with survival data. Measuring these times will allow carrying out a comprehensive analysis, and establish a load balance algorithm within the computer cluster to obtain an optimal algorithm distribution and reduce execution times.

References

1. Khanesar, M.A., Teshnehlab, M., Shoorehdeli, M.A. A novel binary particle swarm optimization. 2007 Mediterranean Conference on Control & Automation, pp. 1-6, doi: 10.1109/MED.2007.4433821 (2007)
2. Pashaei, E., Aydin, N. Binary black hole algorithm for feature selection and classification on biological data. Applied Soft Computing, 56, 94-106 (2017)
3. Camele, G., Menazzi, S., Chanfreau, H., Marraco, A., Hasperué, W., Butti, M. D., Abba, M.C. Multiomix: a cloud-based platform to infer cancer genomic and epigenomic events associated with gene expression modulation. Bioinformatics, 38(3), 866-868 (2022)

4. Ishwaran, H., Kogalur, U. B., Blackstone, E. H., Lauer, M. S. Random survival forests. The annals of applied statistics, 2(3), 841-860 (2008)
5. Camele, G., Hasperué, W., Ronchetti, F., Quiroga, F. M. Statistical analysis of the performance of four Apache Spark ML algorithms. Journal of Computer Science and Technology, 22(2), e14-e14 (2022)
6. Moncada-Torres, A., van Maaren, M. C., Hendriks, M. P., Siesling, S., Geleijnse, G. Explainable machine learning can outperform Cox regression predictions and provide insights in breast cancer survival. Scientific Reports, 11(1), 1-13 (2021)
7. Pölsterl, S., Navab, N., Katouzian, A. An efficient training algorithm for kernel survival support vector machines. arXiv preprint arXiv:1611.07054 (2016)
8. Pölsterl, S., Navab, N., Katouzian, A., Fast Training of Support Vector Machines for Survival Analysis, Machine Learning and Knowledge Discovery in Databases: European
9. Conference, ECML PKDD 2015, Porto, Portugal, Lecture Notes in Computer Science, vol. 9285, pp. 243-259 (2015)
10. Pölsterl, S., Navab, N., Katouzian, A., An Efficient Training Algorithm for Kernel Survival Support Vector Machines 4th Workshop on Machine Learning in Life Sciences, 23 September 2016, Riva del Garda, Italy (2016)
11. Botev, Z., Ridder, A. Variance reduction. Wiley statsRef: Statistics reference online, 1-6 (2017)
12. Harrell, F. E., Lee, K. L., Mark, D. B. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. Statistics in Medicine 15, 361–387 (1996)
13. Harrell, F. E., Califf, R. M., Pryor, D. B., Lee, K. L., Rosati, R. A. Evaluating the yield of medical tests. The Journal of the American Medical Association 247, 2543–2546. (1982)
14. Liu, J., Lichtenberg, T., Hoadley, K. A., Poisson, L. M., Lazar, A. J., Cherniack, A. D., Kovatich, A. J., Benz, C. C., Levine, D. A., Lee, A. V., Omberg, L., Wolf, D. M., Shriver, C. D., Thorsson, V., Cancer Genome Atlas Research Network, Hu, H. An Integrated TCGA Pan-Cancer Clinical Data Resource to Drive High-Quality Survival Outcome Analytics. Cell, 173(2), 400–416.e11. https://doi.org/10.1016/j.cell.2018.02.052 (2018)
15. Batuwita, R., Palade, V. Class imbalance learning methods for support vector machines. Imbalanced learning: Foundations, algorithms, and applications, 83-99 (2013)