UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Followup

Analysis of the SVM-RFE algorithm for feature selection

Author:
Robert PLANAS

Director:
Luis A. BELANCHE

Bachelor Degree in Informatics Engineering
Specialization: Computing

May 28, 2021

Contents

Chapter 1

Follow Up

This document describes the changes and deviations that have occurred from the initial work plan. Points not specified in this document remain unchanged, this includes cost, sustainability, and completed extensions (see below).

Completed Extensions

- Dynamic Step
- Dynamic Sampling
- Dynamic Stop Condition
- Non-linear Kernels

1.1 Changes to the schedule

1.1.1 Theory first

Although the writing of the theory section of this thesis was planned for the final phase, it has been switched to the first phase together with research. This has was needed because, in order to have a clear understanding of the ongoing research, it was useful to write it down. Having a written version of the research helps clarify ideas and have an immediate source of truth for the problems encountered while designing the experiments.

This was specially important for the theory on kernels of SVM and the theory of ranking criteria of SVM-RFE. Without a clear understanding of it, the non-linear kernel extension (a cornerstone experiment) would not been possible to implement.

At this moment, only small corrections and discussed are likely to be necessary. Some points in this direction are: regression problems, multi-class theory, discussion about naming Hessian the matrix H and discussion about the Taylor series approximation method for calculating the ranking criteria.

1.1.2 Various delays

Some delays have occurred due to unforeseen circumstances. These forced some meetups to be pushed to a later date. The specific circumstances are:

- 2 week delay caused by medical problems.
- 1 week delay caused by ransomware infection on a managed (family business) server.

1.2 Problems and solutions

1.2.1 Cancelled extensions

Some experiments have been cancelled due various circumstances.

Avoid CV

Our definition consisted in using other classifier algorithms, such as LDA (Linear Discriminant Analysis) or Logistic Regression. These however fall too much outside the scope of this project. If they were to be introduced, the theory for both would also need to be written. The theory about the specific characteristics that may make them more suitable, or not, for the RFE procedure would need to be researched as well.

Not only a lot of work would have to be put on researching classifiers that do not really have a direct connection with the SVM-RFE algorithm itself, but this work would also be incompatible with non-linear kernels (due not direct application of the general ranking criteria formula), thus creating two disjoint research.

Historic of weight

This was dropped soon after realizing how the ranking criteria of SVM-RFE works. It is stated in the theory that weights of previous iterations will necessarily be worse estimators of feature importance compared to the current iteration. Therefore, doing a mean, or a similar average, will inevitably lead to worse performance.

Two alternatives have been proposed but also dropped:

1. Utilize the variance accumulated in all previous iterations in some sensible manner.

2. Instead of using the weight, use the alpha values of the last iteration to initialize the SVM optimization problem. This is expected to reduce the computational cost required to reach a solution.

The first alternative lacks any kind of theoretical background, and is likely to not perform any better. Is thus, simply not attractive enough of an experiment to pursue (high chance of failing to produce any improvement at all).

The second alternative is more interesting but has one major flaw. In our current framework we're not implementing the quadratic solver program required to solve SVM, instead we're using the C++ implementation LIBSVM/LIBLINEAR. We're not using C++, instead we're using Python (for the numerous tools in its ecosystem catered to data analysis and machine learning). This means that in order to use this implementation we're limited to a Python API the performs the binding. This API does not expose the alpha values on initialization, only as a final result. In order to make these aviable we would need to make changes to the C++ implementation, recompile, bind, and then modify the Python API. This is way too much outside the scope of this project and will take much more time than that allocated for correcting bugs in the initial plan.

1.2.2 New extensions

Combining the successful extensions (Non-linear Kernel + Sampling + Dynamic Step) into one has been proposed and added to the planing.

1.2.3 Model Selection

It was well known beforehand that SVM are very sensitive to hyper-paremters. This makes model selection a very important step in the process. For model selection to work, we needed a way to compare how good two different feature rankings are.

We used an approx based on multi-variate optimization using linear scalarization. This allows us to convert the ranking into a scalar (representing cost) that we can easily compare.

For numerical stability we use cross-validation in every experiment. We've parallelized the cross-validation procedure in order to gain some speed. We're using 6-fold cross-validation on specially slow experiments because we have a computer with 8 cores, and we want to make sure that the experiments are done in a single round.

For the actual model selection we're using a simple grid search strategy.

1.2.4 Madelon

Soon after starting the first experiments we found that the Madelon dataset was not linearly separable, and thus a poor choice for all experiments except for non-linear kernels. To overcome this problem we choose to use artificial problems generated with the sklearn function `make_classification`. This allowed us to showcase the features of each extension with a dataset that suited it the most.

Once the non-linear kernel extension is implemented, future planing is to combine it with the other extensions and hopefully see similar results that with the linear version.

1.2.5 Non-linear Kernel

This has clearly become the bulk of the project as it required the most effort for both the research and the implementation. Various problems where found during the implementation phase:

LIBLINEAR vs LIBSVM

We had to switch to the LIBSVM implementation, which made the experiments run considerably slower. No real alternative has been found, and likely no alternative that is readily available exists.

No API for retrieving the Kernel Matrix

SkLearn does not provide an API that allows retrieving the kernel matrix. Usually it computes the kernel internally based on its name and parameters and hides it from the user. Fortunately sklearn does provide a precomputed mode that allows computing the kernel matrix yourself and pass it to the solver. This mode is poorly documented and it was necessary to read to source code to find handle it.

Using this mode also slightly increased the computational cost of SVM-RFE, even though SkLearn own functions where being used to generate the kernel matrix.

Slow optimizations

Experimentally we found that computing the ranking criteria took about 90% of the time on each run of SVM-RFE with the naive implementation. This is grounds for applying the known (but not formalized) optimizations mentioned on some research papers. This consists on caching results from previous computations and restricting the computation to support vectors.

To apply these optimizations we had to drop the `sklearn` kernel function implementation and made our own. This proved immediately problematic because we could not use `numpy` for most computations and had to rely on explicitly declaring a double loop. `Numpy` speeds up computations by relying on a low level compiled implementation of its functions. By not using it, our equivalent implementation in pure Python performed much slower (in the order of 70 times slower).

We mitigated this problem by using a library called `numba`. This library compiles selected Python functions so that we can get similar speeds to what we would get with an implementation made in a compiled language. Still, our implementation in `numba` was about 3 times slower than with `sklearn`.

Applying the optimizations we managed to get a speed increase by a factor of 3. Although the code is slower, the complexity is one degree faster, form $O(dn^2)$ for the `sklearn` version to $O(n^2)$ for ours (where d is the number of features and n the number of observations).

1.2.6 Plotting inconsistencies

Generating the plots has been one of the most time-consuming and involved tasks. We've managed to create a representation that handles all the important information in a single place. The plots display the train and test accuracy as well as the Peano optimal selected and the projection line of all possible optimal solutions. It also showcases the *step* parameter used by plotting the train accuracy calculated at each iteration of the SVM-RFE algorithm (this is done with a scatter plot).

Interestingly, for some plots the test accuracy calculated after the SVM-RFE phase and the one during the SVM-RFE are substantially different. When this happens it is either due to a bug or an intentional difference on the data used to provide the score (like when sampling is used internally). In this second case, it is a feature, not a bug.

Also note that some $step > 1$ is often used in both phases. If this step is not the same, further inconsistencies may appear. Those are hardly noticeable when the difference is small, and provide some insight when the difference is big. That is, we can plot the accuracy at multiple feature subset sizes even when only one feature subset was returned from SVM-RFE (as in the case where the *step* = nºfeatures).

1.2.7 Dynamic stop-condition fails to improve performance substantially

Although the experiment itself was a success, that is, the implementation worked as expected, the resulting gain in performance is minimal due to the fact that the stop point is often in the last iterations and these are precisely the ones that are less computationally expensive.

We believe that for some specific datasets (where the stop condition is at half the amount of features), we could get a better impact (of about 25% performance improvement). This remains to be tested experimentally.

1.3 Remaining work

The theory documentation part of the project has been completed. The remaining work is on experimentation, analysis and conclusions. Of the experimentation 2 of

ID	Description	Total (h)	Remaining (h)
T1	Project Managment	-	-
T1.1	Context and Scope	-	-
T1.2	Project planning	-	-
T1.3	Budget and sustainability	-	-
T1.4	Final project definition	-	-
T1.5	Meetings	-	-
T2	Theoretical Part	160	20
T2.1	Research	90	5
T2.2	Formalize	20	5
T2.3	Analyze	50	10
T3	Practical Part	160	70
T3.1	Program the base SVM-RFE algorithm	10	0
T3.2	Program the extensions	50	15
T3.3	Test with artificial data	20	5
T3.4	Test with Madelon	30	20
T3.5	Analyze the results	50	30
T4	Documentation	80	50
T4.1	Write the documentation	80	50
T4.2	Prepare the thesis defense	-	-
	Total	400	140

TABLE 1.1: Summary and time estimates of the completed and remaining tasks.

6 extensions remains to be implemented, 1 of 4 requires changes in the implementation, and 4 of 6 require further analysis.

Specifically the remaining tasks are:

- Extension (T3 — 5h + 10h | 15h): Combine Non-linear kernels with Sampling and Dynamic Step.
- Extension (T2 — 20h, T3 — 5h + 5h + 10h | 40h): Program the multi-class extension.
- Implementation change (5h): Optimize RBF kernel with Numba.
- Analysis (10h): Multi-class.
- Analysis (10h): Non-linear kernels with Sampling and Dynamic Step.
- Analysis (5h): Document a case where dynamic stop condition is useful.
- Analysis (5h): Non-linear kernel complexity analysis, with and without extensions.
- (50h) Write the final document (experiments + conclusions).

The expected time needed to fully complete the remaining tasks are 140h, which represents 35% of the project and should be attainable with a work schedule of 7h/day and completed by 17/06/2021.