UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Draft

Analysis of the SVM-RFE algorithm for feature selection

Author:
Robert PLANAS

Director:
Luis A. BELANCHE

Bachelor Degree in Informatics Engineering
Specialization: Computing

April 16, 2021

Contents

Chapter 1

Introduction

This bachelor thesis of the Computer Engineering Degree, specialization in Computing, has been done in the Facultat d'Informàtica de Barcelona of the Universitat Politècnica de Catalunya and directed by Luis Antonio Belanche Muñoz, doctorate in Computer Science.

1.1   Context

In statistics, machine-learning, data-mining, and other related disciplines, it is often the case that there is redundant or irrelevant data in a dataset[1]. Indeed, before we can start working with the data, some form of data analysis and cleaning is required. Data cleaning may include removing duplicated rows or rows with missing values, removing observations that are clearly out-layers, removing irrelevant variables (e.g. name, surname, email address), etc.

With the new era of Big Data, datasets have increased in size, both in number of observations and in dimensions. Applying classical data-mining and machine-learning algorithms to this high-dimensional data rises multiple issues collectively known as "the curse of dimensionality". One such issue is the elevated, usually intractable, cost and memory requirements derived from the non-linear (on number of dimensions) complexities of the algorithms. Another issue has to do with data in a high-dimensional space becoming sparse and negatively affecting the performance of algorithms designed to work in a low-dimensional space. And finally, a third issue is that with a high number of dimensions the algorithms tend to overfit, that is, they don't generalize enough and end up producing models that perform worse with real data that their predicted performance with the training data. (Li et al., 2017)

Simple manual data cleaning is not enough to achieve satisfactory amounts of dimensionality reduction. In this case we can use automatic techniques. We can classify such techniques in two categories, feature extraction, and feature selection.

1.1.1   Feature Extraction

Feature extraction techniques transform the original high-dimensional space into a new low-dimensional space by extracting or deriving information from the original features. The premise is to compress the data in order to pack the same information at the expense of model explainability[2]. Continuing with our data compression analogy, virtually all feature extraction techniques perform lossy compression, that

---

[1]A table with rows / records / observations and columns / variables / features / dimensions / predictors / attributes.

[2]The ability to explain why certain predictions are made. Also, interpretability.

is, some data is lost which makes the process irreversible. Notice that, if the process was reversible then feature extraction would not decrement explainability.

Some well known feature extraction algorithms include Principal Component Analysis (PCA) and auto-encoders, the first being a linear transformation over the feature-space and the second a neuronal network. PCA may be extended with a kernel method in order to make non-linear transformations. Similarly, we will also use non-linear kernels to extend SVM-RFE.

1.1.2 Feature Selection

In contrast, feature selection only selects a subset of the existing features, ideally the most relevant or useful. This may imply a greater loss of information compared to feature extraction, but it doesn't reduce explainability. Some problems require feature selection explicitly. In domains such as genetic analysis and text mining, feature selection is not necessarily used to build predictors. For example in microarray analysis feature selection is used to identify genes (i.e. features) that discriminate between healthy and disease patients.

Feature selection methods may be classified by how they are constructed in three categories:

- **Filter:** A *feature ranking criteria* is used to sort the features in order of relevance, then select the *k*-most relevant.

- **Wrapper:** They use a learning machine (treated as a black box) to train and test the dataset with different subsets of variables. They rank the subsets based on the performance (score) of the model. A wide range of learning machines and search strategies can be used. Within the greedy strategies we find *forward selection* and *backward elimination*.

- **Embedded:** Like wrapper methods but more efficient. They use information from the trained model itself to make feature selection. Because they don't use the score, they can also skip testing the model.

In both wrapper and embedded methods greedy strategies can be used. **SVM-RFE** is a feature selection algorithm, of the embedded class, that uses Support Vector Machines (SVM) and a greedy strategy called Recursive Feature Elimination (RFE). This algorithm is an instance of backward elimination, it starts with a set of all features and eliminates the less relevant each iteration. Within each iteration SVM-RFE behaves like a filter method and uses a feature ranking criteria to decide which feature to eliminate. SVM-RFE takes advantage of the fact that, for linear SVM, the ranking criteria is to take the variable with the smallest weight in each iteration, where the wights are the coefficients of the hyperplane resulting from the SVM training. (Guyon and Elisseeff, 2003)

1.2 State of the art

The SVM-RFE algorithm was first proposed in a paper on the topic of cancer classification (Guyon et al., 2002). This paper uses the SVM-RFE algorithm to identify highly discriminant genes, encoded as features, that have plausible relevance to cancer diagnosis. Since then, SVM-RFE has remained a popular technique for gene selection and the original paper cited more than four thousand times.

The paper already proposes some natural extensions to the algorithm, such as eliminating multiple features each iteration based on the ranking and a *step* constant. Further research has been on improving and extending different parts of the algorithm. These include the use of a Gaussian kernel (Xue et al., 2018), using multiple SVM in the same iteration (Wang et al., 2011) or simply trying to find a better ranking criterion (Mundra and Rajapakse, 2007).

1.3 Objective

The main objective of this project is to research extensions of the SVM-RFE algorithm and try to optimize it. Optimizations may be in the form of improved performance, a reduction in time utilization or less memory required. We've classified the possible extensions as follows.

Main extensions

These focus on improving the performance of the selection.

- **Non-linear kernel:** New ranking criteria needs to be found for the kernel used.

- **Avoid using cross-validation:** The C parameter may be used in a better way.

- **Use information form previous iterations:** E.g. make a better ranking criterion by combining the weights of the previous iteration with the current one.

- **Multi-class criteria:** Find a ranking criterion that can handle multiple weight vectors in a useful way.

Secondary extensions

Focus on reducing the computational cost.

- **Sampling:** Use a subset of the observations. Ideally, use a different subset on each iteration, see if results converge.

- **Dynamic Step:** Instead of using a constant value in each iteration, calculate it dynamically.

- **Stop condition:** Determine the amount of features that are relevant (SVM-RFE only provides a ranking) in an effective and inexpensive manner.

1.3.1 Objective break down

To accomplish this objective, the project has been subdivided in two parts, each having specific tasks for each extension:

Theoretical Part

- Do research in SVM-RFE and in the extensions that will be tackled in the project.

- For each extension:

  - Design the algorithm and write its formalization in pseudocode.

– Define the expected advantages or disadvantages of this extension over the base SVM-RFE.

– Compute the space and time complexities.

Practical Part

- Program the base SVM-RFE algorithm and the extensions.

- For each extension:

  – Analyze its behavior for artificial data sets.

  – Analyze its behavior for real-world data sets.

- Compare the results obtained with the ones expected.

- Combine multiple extensions to further improve the algorithm.

- Draw conclusions about all the results obtained in the project.

1.3.2 Stakeholders

This project is intended to be of use for many involved parties. The most directly involved group, is the tutor and the researcher. Luis Antonio Belanche Muñoz is the tutor of this project. Robert Planas Jimenez would be the researcher. Feature selection algorithms is one of the areas of research of the tutor, and he has wanted to explore extensions to the SVM-RFE algorithm. Thus, he will lead and guide the researcher for the correct development of the project. The researcher is responsible for planning, developing and documenting the project, as well as experimenting, analyzing and drawing conclusions.

The other group of interested parties would be stakeholders that do not interact with the project directly but still benefit from it. In the first place we have researchers on the fields of bioinformatics and data mining, that use machine learning methods (specifically, SVM-RFE) for micro-array analysis, text analysis, or other of its popular applications. Indirectly companies that make use of their findings will also benefit, and finally the general population may also benefit from better diagnostics and more effective drugs.

1.3.3 Potential obstacles and risks

Some obstacles and risks identified that could potentially prevent the correct execution of the project are:

- **Deadline of the project:** There is a deadline for the delivery of the project. This being a research project however, is considerably hard to estimate how much time tasks will take, or even decide whether a task has been finished or not.

- **Bugs on some libraries:** This is considered of low risk, but is still a possibility that errors on the software package used extend to code, making it work incorrectly.

- **Insufficient computational power:** Machine learning algorithms in general can be very resource intensive. It could be the case that our hardware can not handle some datasets.

- **Hardware related issues:** A hard drive failure could occur that would end in lost data, or a failure in a router could disconnect us from the internet.

- **Health related issues:** In addition to health issues that can occur at any time without prior notice, we're in the middle of a pandemic.

1.4 Methodology

1.4.1 Framework

The methodology that I will use for the project is a combination of waterfall and Kanban methodologies. Waterfall will be used to define the general phases of the project, and Kanban for tracking the individual tasks. In waterfall tasks can not start until the previous task has been completed, and thus following strict deadlines is important. Phases will be managed like this, one phase will not start until the previous phase ends. Each phase will be composed of multiple tasks, which will then be managed by a different methodology. That will be Kanban.

Kanban is much more flexible than waterfall, its principal objective is to manage tasks in a general way, by assigning different status to them. Kanban stands out by its simplicity, and we continue to endorse that simplicity by managing the visual representation of the cards in a simple plain text formatting. Each card will be in a row, with the first column defining its name and the other its status. The statuses I've considered are:

- **To do:** A basic idea of the task is present.

- **Definition:** The task is in the theoretical part.

- **Implementation:** The task is in the practical part.

- **Completed:** The task is finished.

An uppercase letter `"X"` will mark the current state of any given card. If more granular information is required, other marks may be used instead. For example, to indicate that the task is paused a `"P"` would be used. To indicate the progress within some stage a percentage would be used. This table will be kept within the `readme.md` file of the `GitHub` repository of the project, for easy monitoring.

1.4.2 Validation

We will use a GitHub repository as a tool for version control, which will allows us to share code easily and the recover from data lose. The repository will contain both the code for the different experiments, each in one subfolder, and code for the documentation. In order to verify the implemented code, it will be tested with multiple standard data sets. In the practical part, hyper-parameters will be selected using some model validation technique, such as the cross-validation. Each experiment will be done 3 times and the average of the results will be the final result. Lastly, face-to-face meetings will be scheduled once every two weeks with the tutor of the project. In these meetings it will be discussed the project status and the tasks to do during the following two weeks, before the next meeting. In case of problems in the project, extraordinary meetings can be arranged.

Chapter 2

Project Planning

This thesis is worth 15 ECTS credits, each of which with an estimate cost of 25 to 30 hours. Therefore, the total time allocated for this project, as indicated by the faculty, is of 375 to 450 hours. This time is to be distributed in 100 days, from 03/08/21 to 06/15/21, with an estimated work of 4 to 5 hours a day. The date of the oral defense is planned for the first week of July, this sets the deadline to be on the 06/18/21.

An extra 3 ECTS credits are to be used for project managment, this is roughly 80 hours, which makes the total time estimate for the whole project (Theisis + Project Managment) to be at best 450 hours and at worse 540 hours. In order to make a proper plannification, we have defined the estimated cost to be at 500 hours.

2.1 Task definition

In this section it is presented all the tasks that will be carried out along the project. For each, a description, duration and a list of dependencies with other tasks are given.

Project management is a mandatory group of such tasks, albeit not very useful, considering that: One, this project is done by a single individual, with assistance from a project director; And two, this is a research project, which makes planning of specific tasks difficult, since it is the results from the research that drive the next steps to be done.

- **Context and scope:** We have to indicate the general objective(s) of the project, contextualize it and justify the reason for selecting this subject area.

- **Project planning:** This will help us not lose focus while we're working on the project.

- **Budget and sustainability:** For this specific project, this is irrelevant. The budget required is negligible and already defrayed; and the impact, beyond trivial matters, is likely zero and otherwise unknown.

- **Final project definition:** Review the work done in the project management tasks.

- **Meetings:** Online meetings are scheduled once every two weeks with the tutor of the project. Discussion of the status and next tasks to do will be done.

This project is research focused. Therefore, before starting the practical tasks research on the various topics needs to be done. This will involve collecting and analyzing previous studies that tried new methods and extensions to the SVM-RFE algorithm. We will also have to document ourselves in the SVM and feature selection

areas, as well as the algorithms and the statistical theory used in the studies. Some basic understanding of bioinformatics may also be required considering the use case of most if these studies.

- **Research** previous work on the literature on extensions of the algorithm and create a short **report** with the findings.

- Write the algorithm formalization in pseudocode.

- Define the expected advantages or disadvantages of this method over the base SVM-RFE.

- Compute the space and time complexities.

Once the initial research is done the algorithm must be codified and tested. This group is composed of the following tasks:

- **Program the base SVM-RFE algorithm.** For this we will use a library for the SVM. For the RFE part, since we need to be able to extend the algorithm, we will program it from scratch.

- **Program the extensions of the SVM-RFE algorithm** based on the research done and the pseudocode.

- **Test the new extensions with artificial data.** This requires creating models and testing their performance with artificial data sets.

- **Obtain data sets with real data.** Multiple papers refer to publically available data-sets, we could use those and compare our results.

- **Test the new extensions with real data.** This requires creating models and testing their performance with real data sets.

- **Analyze the results** obtained in the experiments and draw conclusions. A report will be made for reference during the final documentation.

Related to testing and analyzing the results, if we want to draw fair conclusions, comparing our results with the state of the art is important. However, the start of the art is bast and fluid, and making a fair comparison with all available research papers is a daunting task. Instead, we will use a common ground, in this case the NIPS 2003 feature selection challenge. The results of this challenge, as well as a general analysis of the algorithms used, will serve as a reference point. Thankfully an analysis of these results has already been done (Guyon et al., 2004). This simplifies our problem, now we only need to compare our algorithms performance with those already presented in the challenge.

Once finished, the final documenting phase will begin. Firstly, we will collect all the information obtained in the experimental and analysis part, which will be available in the form of the reports that we've done along the tasks. Afterwards, we can start writing the documentation of the project. This will include a fairly extensive review on concepts related to SVM, statistics, feature selection and RFE. Finally, we will have to prepare for the oral defense of the project.

ID	Description	Hours	Dependencies
T1	Project Managment	80	
T1.1	Context and Scope	20	T2.1
T1.2	Project planning	10	
T1.3	Budget and sustainability	10	T1.2
T1.4	Final project definition	20	T1.1, T1.2, T1.3
T1.5	Meetings	20	
T2	Theoretical Part	160	
T2.1	Research	90	
T2.2	Formalize	20	T2.1
T2.3	Analyze	50	T2.2
T3	Practical Part	160	
T3.1	Program the base SVM-RFE algorithm	10	T2.1
T3.2	Program the extensions	50	T2.1, T3.1
T3.3	Test with artificial data	20	T3.2
T3.4	Test with real data	30	T3.2
T3.5	Analyze the results	50	T3.3, T3.4
T4	Documentation	100	
T4.1	Writte the documentation	80	T2, T3
T4.2	Prepare the thesis defense	20	T4.1

TABLE 2.1: Summary and time estimates of the tasks.
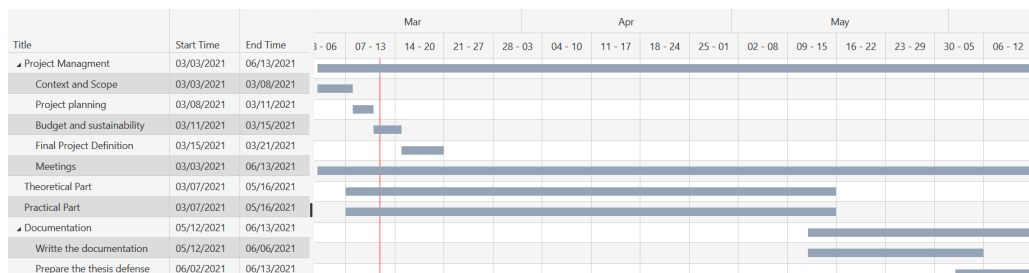
FIGURE 2.1: A summary of the tasks represented with a gantt chart.
Notice that all the theoretical and practical tasks are done in parallel.

2.2 Resources

Our project needs resources to carry out its correct development. These resources have been divided in 4 different groups: human, hardware, software and material resources.

2.2.1 Human resources

There are three human resources that are directly involved in this project.

- **The researcher:** He is responsible for the development of the project, that is, he will have to plan, analyze, program, experiment and document the project.

- **The director/tutor:** He is responsible for leading and guiding the researcher for the correct development of the project.

- **The GEP tutor:** He is in charge of reviewing the project management tasks done in the initial stage of the project.

2.2.2 Hardware Resources

The most essential resource needed is a computer connected to the internet. In this project a personal computer will be used. Its specializations are 16 GB of RAM and a CPU *AMD Ryzen 7 4800HS*, with a base speed of 2.9 GHz and 8 cores. Hardware required for a connection to the internet (a router, an access point, etc) is also taken into account.

2.2.3 Software Resources

For project management tasks `Google Calendar` will be used. A number of other Google products such as `Google Mail`, `Google Drive` or `Google Meet` will also be used as tools required for communication with the director or storing of information.

The documentation will be written in LaTeX, a document preparation system often used in academia. It has the advantage to integrate well with version control systems. A LaTeX template named "`Masters/Doctoral Thesis`" will be used to facilitate the typesetting and styling of the document. This template was made by Vel and Johannes Böttcher and licensed under `LPPL 1.3`, based on previous work from Steve R. Gun and Sunil Patel and used with minor modifications. The original is available at http://www.latextemplates.com/. The document will be written with the `Microsft Visual Studio Code` editor, using the `LaTeX Workshop` extension, and it will be compiled with the `MiKTex` distribuition installed on a `Linux` machine running virtualized within a `WSL` container on top of the actual operating system, a `Microsoft Windows 10`. For browsing the internet the latest available version of the `Mozilla Firefox` web browser will be used, and references to papers will be kept with the `Zotero` reference manager.

For the practical part, the programming language of choice will be `Python3`. This is currently one of the programming languages with better popularity in machine learning and related applications. Various libraries and software packages will be used for different tasks. For parsing datasets the `pandas` library will be used. For data visualization the `mathplotlib` library will be used. Also, a general tool-set designed for machine learning, the library `sklearn`, will be used. Finally, code will be documented in-place during its development with the `Jupyter Notebook` software.

Other libraries and software packages could also be used if the need arises.

2.2.4 Material Resources

In a research focused project such as this, access to scientific journals, books and similar research material is needed. Some articles related to this research are freely available on the internet, but some require a subscription or single time payment. Fortunately, the UPC, the university this project is being developed at, has a subscription agreement with most of these journals and provides free access to their memebers, including the students.

2.3   Risk Management

The potential risks and obstacles have already been introduced in section 1.3.2. In this section we will focus on a contingency plan to mitigate the risks.

- **Deadline of the project:** The flexibility of the Kanban methodology should help us modify our schedule and working hours if required. If it becomes apparent that the deadline will not be meet, that is, more than 50 hours a week of work are required to finish the project in time, an extension of the deadline can be requested.

- **Bugs on some libraries:** Alternative libraries can be used. If no alternative is found, because most of the used libraries are open source, a *bugfix* could be implemented.

- **Insufficient computational power:** This can be mitigated by using a small sample of the data-set. Working with fewer data, although faster, can induce a small performance reduction and make the results not comparable with each other. Therefore, this will solution is not ideal.

- **Hardware related issues:** To avoid data loss, all code and documentation will be routinely uploaded to the cloud in a `GitHub` repository and `Google Drive` account.

- **Health related issues:** Not much can be done if a health related problem occurs, but various preventive actions, such as a good diet, exercise and resting habits can be promoted.

Chapter 3

Budget and Sustainability

In this section a budget estimation will be made. It will include personnel costs per task, generic costs and other costs. Moreover, some questions regarding the sustainability aspect of the project will be answered.

3.1 Budget

3.1.1 Costs by role and activity

In this section we will add the personnel costs to the tasks defined in section 2.1. For each task a cost will be calculated based on the hourly pay wage per role and the time spend each. Four roles have been defined with their corresponding hourly wage, these are:

- The **project manager** (T1) who is responsible for leading the project direction, planning and correct development.

- The **researcher** (T2), who must perform research in the topic and related ideas, experiment, analyze the results and draw conclusions from them.

- The **programmer** (T3), who must set up the developer environment, code the algorithms in the specified programming language and test them for correctness.

- The **technical writer** (T4), who is responsible for writing this project documentation. This includes the reports for each task that requires it and the final thesis.

These roles have a clear mapping to the task groups defined in table 2.1. All roles will be played by the researcher (see section 2.2.1) except the project manager role, which will be played by the researcher, the director and the GEP tutor. For a detailed description of each task cost see table 3.1.

Notice that, although we aligned the project roles with the task groups so that we could simplify our calculations, it is not required to do so. A more complicated scheme where multiple tasks are assigned to a given role is also possible. In such cases we would also have to think about weather the work distribution is uniform or not, and assign percentages if it isn't.

Role	Year (€)	Year +SS (€)	Hour +SS (€)	Task	Task Cost (€)
Project Manager	39 000	50 700	28.7	T1	2 296
Researcher	32 000	41 600	23.5	T2	3 760
Programmer	26 000	33 800	19.1	T3	3 065
Technical writer	22 000	28 600	16.2	T4	1 260
Total					**10 381**

TABLE 3.1: Annual estimated salary for the different project roles (*Salarios, ingresos, cohesión social* 2017). The amount of working hours in a year is defined to be 1764. *+SS indicates "Social Security included".*

3.1.2 Generic costs

Amortization

In this section the amortization costs for the resources purchased in a single payment are calculated. Notice that since all the software used is free and open source, it doesn't contribute to the cost, thus it is not displayed here. In fact, the only resource that is valid for an amortization analysis is the computer specified in section 2.2.2.

The equation we use to compute the amortization cost for each resource is the following:

$$\text{Amortization (€)} = \text{Cost (€)} \times \frac{1}{4 \text{ years}} \times \frac{1}{100 \text{ days}} \times \frac{1}{5 \text{ hours}} \times \text{Hours Used} \quad (3.1)$$

If we apply the amortization equation to the computer, which we purchased for €999.95, and assuming 500 hours of usage, its estimated amortized cost is €249.98.

Electric cost

In this section we only calculate a rough estimate. Calculating accurately the cost of electricity involves many variables, and it's outside the scope of this thesis. The average cost of electricity in Spain, in therms of kWh, is €0.12. We only count the cost when the hardware is turned on. The following table (3.2) shows the individual and total cost per item.

Item	Power (W)	Time used (h)	Consumption (kWh)	Cost (€)
Computer	180	500	90	10.8
Router	10	500	5	0.6
Total				**11.4**

TABLE 3.2: Electric cost estimate.

Internet cost

The internet cost in my current location is €29.00 per month, this is roughly about €0.95 per day (variance is introduced because a month length is not constant). The

amount of working hours per day is assumed to be 5, thus the total cost the whole project is:

$$€0.95 \text{ /day} \times 100 \text{ days} \times 5/24 \text{ hours} = €19.79$$

Work space

This project will be developed at my parents home at Figueres, with a rent of €400. Since the amount of people living there is 2, the actual cost is €200.

Total generic costs

Table 3.3 summarizes the total generic costs of this project.

Group	Cost (€)
Amortization	249.98
Electricity	11.4
Internet	19.79
Rent	200
Total	**481.17**

TABLE 3.3: Total generic cost estimate.

3.1.3 Other costs

Contingencies

Unexpected problems that were not foreseen may appear during the development of the project, which would take part of our budget. For this reason it as always a good idea to prepare a contingency budget calculated from the budgets we've calculated up to now. We will apply a 10% contingency margin, that is €108.62.

Incidental cost

Unexpected problems that were foreseen and can be mitigated are described in section 2.3. Some of these mitigations may incur some extra cost. To be able to handle that cost in this section we will calculate a budget based on the predicable problems, their chances of actually occurring, and the expected cost associated for solving them.

- **Deadline of the project:** If an extension of the deadline is requested, that would require extra hours expend by the researcher. Assuming the extra time required to be 50 hours that would give us a cost of €1,175.

- **Bugs on some libraries:** Implementing a *bugfix* we assume would cost around 20 hours, a task done by the programmer. The estimated cost is €382. The risk is small.

- **Insufficient computational power:** If using smaller datasets where not an option, we would abandon the project, since purchasing a new, more powerful, computer or renting a supercomputer is too expensive. The risk is very small.

- **Hardware related issues:** New hardware would be purchased, if possible only the faulty component would be replaced. This would have an estimated cost of about €100. The risk is small.

The following table summarized the expected cost due to foreseen incidents.

Risk	Expected (€)	Risk (%)	Cost (€)
Deadline of the project	1 175	30	352.5
Bugs on some libraries	382	10	38.2
Insufficient computational power		5	
Hardware related issues	100	10	10
Total			**400.7**

TABLE 3.4: Total incidental cost estimate.

3.1.4 Total cost

The total cost for the project is summarized in table 3.5. The cost has been computed as the sum of the justified costs explained in the other sections of the budget plan.

Section	Cost (€)
Costs by role and activity	10 381.00
Generic costs	481.17
Other costs	509.32
Total	**11 371.49**

TABLE 3.5: Total cost estimate.

3.2 Sustainability

This project does not have a major environmental impact, since it's a research project. Still, some impact, a very small amount, is expected from the electricity consumption and hardware used. This can hardly be reduced, as is required for the development of the project. If the project succeeds at finding improvements to the SVM-RFE algorithm, it will reduce computing power requirements for future researches that use it compared to the state-of-the-art solutions. Because this algorithm is used in medical research, a substantial improvement could create a chain reaction that benefits the health of the population in general. This, however, is a very optimistic expectation.

3.2.1 Environmental dimension

Have you estimated the environmental impact of undertaking the project?

This project will not have a major environmental impact. Still, some impact, a very small amount, is present in the form of electricity consumption and the method used by the provider to generate such electricity. Also, the hardware used will eventually contribute to the technological waste problem, not to mention the environmental cost for their production.

Have you considered how to minimize the impact, for example by reusing resources?

The impact is directly derived from the project requirements and can thus not be reduced easily. A low-cost computer doesn't necessarily imply a lower impact on the environment or lower electricity consumption. Recycling is of course always an option, but the decision to do so will be taken way after the project has been completed.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution environmentally improve existing solutions?

This project will use the same resources as those used in state-of-the-art alternatives. Therefore, this solution does not environmentally improve existing solutions. If the project succeeds at finding an improvement to the SVM-RFE algorithm however, it could imply a reduction on computational power required to solve some problems, and thus also a reduction in power consumption.

3.2.2 Economic dimension

Have you estimated the cost of undertaking the project (human and material resources)?

Yes, see section 3.1.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution economically improve existing solutions?

If improvements to the SVM-RFE are found in this project, this will make any research that makes use of it less expensive and produce better results.

3.2.3 Social dimension

What do you think undertaking the project has contributed to you personally?

This is the last step required to finish my degree in computer science. Finalizing this degree will allow me to enter the work force and become self-sufficient. Beyond that, it has introduced me to the fields of bioinformatics and data mining, as well as shown the importance of feature analysis and how it can be not just a cleaning prerequisite for a more important problem, but the main dish.

How is the problem that you wish to address resolved currently (state of the art)? In what ways will your solution socially improve (quality of life) existing. Is there a real need for the project?

Improvements to the SVM-RFE algorithm may or may not be found. Even if found it is still unknown if they will be substantial enough. Moreover, an analysis of the variants of the algorithm will be useful for those who want to use SVM-RFE in the most optimal way.

Chapter 4

Background

In this section we're going to review the concepts required to understand the SVM-RFE algorithm. Section 1.1 already introduced some concepts around the SVM-RFE algorithm, placed it in context, and enumerated some of its applications. In this section we'll focus on the inner workings of the algorithm and how these parts add together.

4.1 Machine learning

Machine learning is a subfield in the broader discipline that is artificial intelligence, with a particular take in statistics. These algorithms, also called learning machines or just machines, use data to learn patterns and make predictions. The data is collected in a separated unrelated process, and structured in the form of a *dataset* (a collection of data). Once the dataset is further cleaned and prepared, the learning machine finally consumes it in a process called *training*. After this process the machine produces a *model*, which is a function that can be used to make predictions on new data.

Two canonical problems in machine learning are regression and classification problems.

4.1.1 The dataset

A dataset is simply a collection of data. In the context of machine learning this dataset will be used to make predictions, take decisions, or find patterns. For a machine learning algorithm to be able to consume a dataset the first step is to represent it in tabular form.

Most datasets come already in tabular form. Some of the most notable exceptions are datasets involving images. In this case computer vision methods are often used to extract numeric data representing characteristics of the image. Sometimes a more direct transformation can also be made, for example making each feature be the intensity level of a single pixel in the image. This of course produces a high number of features, most of which are redundant or irrelevant (e.g. features representing pixels in the background).

In a dataset represented as a table, columns describe different *features, properties or attributes* of some group of objects and rows represent *instances* of that group. For example, if objects were vehicles then features could include the brand, power, weight, maximum speed, and other such characteristics of various vehicles, each of which would be in a row. Different names are used in different contexts. One of the most typical naming conventions comes from the statistics domain which refers to columns as *variables* and rows as *observations*. Sometimes different nomenclature is used to differentiate features in different stages of the cleaning and preparation process, but in this thesis we use them all instinctively.

Src	Dst	NAT-Src	NAT-Dst	Action	Sent (B)	Rcvd. (B)	Packets	Elapsed (sec)
57222	53	54587	53	allow	94	83	2	30
56258	3389	56258	3389	allow	1600	3168	19	17
6881	50321	43265	50321	allow	118	120	2	1199
43537	2323	0	0	deny	60	0	1	0
50002	443	45848	443	allow	6778	18580	31	16

TABLE 4.1: Example extracted from the Internet Firewall Data dataset (Ertam, 2019). Only five observations and main variables shown.

4.1.2 Classification

For the classification problem we want to predict in witch group or *class* to assign some new observation. Table 4.1 provides an example of what could be a good classification problem. Imagine we are building a firewall and want it to predict if some new packet in the network should be allowed to pass. We could train a classification learning machine with the dataset represented in the table (the full version) and produce a model capable of doing such predictions. Although it is trivial to identify a pattern in the example, it may not be so for real world examples. Automatic *pattern recognition* is key in solving many real world problems, and it is one of the features of these algorithms.

Mathematically, the set of observations is defined as $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n\}$ and the set of the corresponding labels or target classes is $y = \{y_1, y_2, \ldots, y_n\}$ with every element of this set being some class $y_i = C_k$ of a discrete set of classes with size K. For our example, this set would be $C = \{\text{allow}, \text{deny}\}$. Thus, we can describe the goal of a classification problem as assigning some class C_k to an input vector $\vec{x}$.

Although we've used strings to represent classes, part of the preparation process of the dataset involves turning all values into numerical scalars, so our classes would actually be some natural numbers. Also notice that every $\vec{x}_i$ is a vector containing a single numerical value for each feature excluding the label.

We make a distinction between two-class problems and multi-class problems. Two-class problems typically use $C = \{0, 1\}$ as classes and thus can be modeled with a boolean function or, if a probability is desired, a function that returns values between 0 and 1. This simplifies the model substantially, in fact some learning machines such as the SVM can only work with two-class problems, and use different methods to extend to the multi-class version.

4.1.3 Visualization

A visualization of the problem in some euclidean space is required to understand how most classification algorithms work, and in particular SVM. Typically, classification models divide the input vector space[1] into *decision regions*. The boundaries of such regions are called *decision boundaries* or *decision surfaces*. If the decision boundary is in the form of some hyperplane of dimension $(D-1)$, with D being the dimension of the space, then we say that it's a linear model. A dataset whose classes can be completely separated by such linear decision boundary is said to be *linearly separable*.

---

[1]The euclidean space of minimum dimension containing all possible input vectors $\vec{x}$.

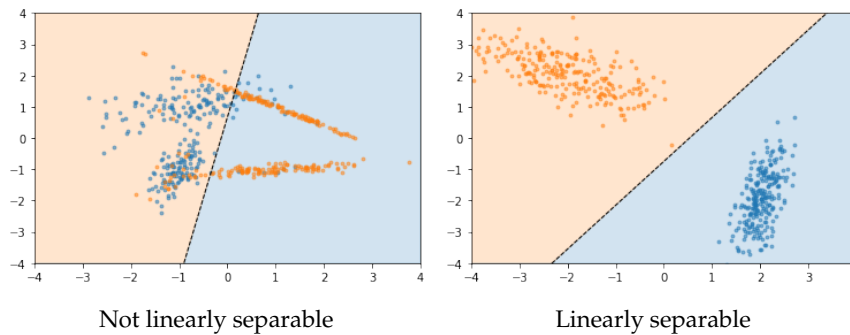|	
Not linearly separable	Linearly separable

FIGURE 4.1: Decision regions and 1-D hyperplane boundary of some linear model for two datasets. Points are observations, colors indicate the class.

4.1.4 Performance

4.1.5 Microarray Analysis

EXTRA

Talk about tagging data, unsupervised (clusetring) vs suppervised problems, regression vs classification, dimensions and representation in space, and visualizationa and microarray analysis

On Regression and Classification, talk about parameters and hyperparameters, statistical models, mathematical expression and test and training sets, etc

———————————–

- What is machine learning.

- Regression.

- Classification.

- The dataset, cleaning, embeddings, etc.

4.2 Support Vector Machines

- 2D classification example.

- Mathematical formulation as an optimization problem.

- Dual

- SVM with regularization.

- Dual

- Optimization algorithms.

4.3 Kernel Methods

- What is a kernel.

- The kernel trick.

4.4 SVM-RFE

- What is RFE.

- The ranking criteria for the linal case.

Algorithm 1: SVM-RFE

Output: $\vec{r}$

Data: $X_0, \vec{y}$

1   $\vec{s} = [1, 2, \ldots, n]$ `// subset of surviving features`

2   $\vec{r} = []$ `// feature ranked list`

3   **while** $|\vec{s}| > 0$ **do**

      `/* Restrict training examples to good feature indices   */`

4      $X = X_0(:, \vec{s})$

      `/* Train the classifier                                  */`

5      $\vec{\alpha} = \text{SVM-train}(X, y)$

      `/* Compute the weight vector of dimension length $|\vec{s}|$   */`

6      $\vec{w} = \sum_k \vec{\alpha}_k \vec{y}_k \vec{x}_k$

      `/* Compute the ranking criteria                          */`

7      $\vec{c} = [(w_i)^2 \text{ for all } i]$

      `/* Find the feature with the smallest ranking criterion  */`

8      $f = \text{argmin}(\vec{c})$

      `/* Update the feature ranking list                       */`

9      $\vec{r} = [\vec{s}(f), ... \vec{r}]$

      `/* Eliminate the feature with smallest ranking criterion */`

10     $\vec{s} = [...\vec{s}(1 : f - 1), ...\vec{s}(f + 1 : |\vec{s}|)]$

11   **end**

Algorithm 2: SVM-RFE with Step

    Input: t `// t = step`

    Output: $\vec{r}$

    Data: $X_0, \vec{y}$

1 $\vec{s} = [1, 2, \dots, n]$ `// subset of surviving features`

2 $\vec{r} = []$ `// feature ranked list`

3 while $|\vec{s}| > 0$ **do**

      /* Restrict training examples to good feature indices   */

4 $X = X_0(:, \vec{s})$

      /* Train the classifier   */

5 $\vec{\alpha} = \texttt{SVM-train}(X, y)$

      /* Compute the weight vector of dimension length $|\vec{s}|$ */

6 $\vec{w} = \sum_k \vec{\alpha}_k \vec{y}_k \vec{x}_k$

      /* Compute the ranking criteria   */

7 $\vec{c} = [(w_i)^2 \text{ for all } i]$

      /* Find the t features with the smallest ranking criterion   */

8 $\vec{f} = \texttt{argsort}(\vec{c})(:t)$

      /* Update the feature ranking list   */

9 $\vec{r} = [\vec{s}(\vec{f}), \dots \vec{r}]$

      /* Eliminate the features with the t smallest ranking criterion   */

10 $\vec{s} = [[\dots \vec{s}(1 : f_i - 1), \dots \vec{s}(f_i + 1 : |\vec{s}|)] \text{ for all } i]$

11 end

Bibliography

Ertam, Fatih (Feb. 2019). *Internet Firewall Data Data Set*. URL: https://archive.ics.
uci.edu/ml/datasets/Internet+Firewall+Data (visited on 03/25/2021).

Guyon, Isabelle and André Elisseeff (2003). "An Introduction to Variable and Feature
Selection". In: *Journal of Machine Learning Research* 3.Mar, pp. 1157–1182. ISSN:
1533-7928.

Guyon, Isabelle et al. (Jan. 2002). "Gene Selection for Cancer Classification using
Support Vector Machines". en. In: *Machine Learning* 46.1, pp. 389–422. ISSN: 1573-
0565.

Guyon, Isabelle et al. (Jan. 2004). "Result Analysis of the NIPS 2003 Feature Selection
Challenge". In: vol. 17.

Li, Jundong et al. (Dec. 2017). "Feature Selection: A Data Perspective". In: *ACM
Comput. Surv.* 50.6, 94:1–94:45. ISSN: 0360-0300. URL: https://doi.org/10.
1145/3136625 (visited on 03/01/2021).

Mundra, Piyushkumar A. and Jagath C. Rajapakse (2007). "SVM-RFE with Relevancy
and Redundancy Criteria for Gene Selection". en. In: *Pattern Recognition in Bioinformatics*.
Ed. by Jagath C. Rajapakse, Bertil Schmidt, and Gwenn Volkert. Lecture Notes in
Computer Science. Berlin, Heidelberg: Springer, pp. 242–252. ISBN: 978-3-540-
75286-8.

Salarios, ingresos, cohesión social (2017). Tech. rep. Encuestas de Estructura Salarial.
INE. URL: https://www.ine.es/jaxiT3/Tabla.htm?t=10911.

Wang, Jingjing et al. (Nov. 2011). "Classification of lip color based on multiple SVM-
RFE". In: pp. 769–772.

Xue, Yangtao et al. (Oct. 2018). "Nonlinear feature selection using Gaussian kernel
SVM-RFE for fault diagnosis". en. In: *Appl Intell* 48.10, pp. 3306–3331. ISSN: 1573-
7497. URL: https://doi.org/10.1007/s10489-018-1140-3 (visited on
03/18/2021).