



Sisteme Inteligente

Activitate de laborator 2016-2017

Titlul proiectului: Clasificarea email-urilor
Tool: LibSVM

Nume: Bánhidi Zoltán
Grupa: 30234
Email: banhidizoli@gmail.com

Prof. dr. ing. Adrian Groza
Adrian.Groza@cs.utcluj.ro



Contents

1	Instalarea programelor (W_2)	4
2	Rularea și înțelegerea exemplelor (W_3)	6
3	Understanding conceptual instrumentation (W_4)	8
3.1	Exercises	8
4	Descrierea proiectului (W_5)	9
4.1	Narrative description	11
4.2	Facts	11
4.3	Specifications	12
4.4	Top level design of the scenario	12
4.5	Knowledge acquisition	12
4.6	Related work	12
4.7	Exercises	13
5	Preliminary results (W_7)	14
5.1	Exercises	14
6	Implementation details (W_9)	15
6.1	Relevant code	15
6.2	Common bad practice in AI undergraduate projects	15
6.3	Exercises	16
7	Tool expressivity (W_{10})	17
8	Graphs and experiments (W_{11})	18
8.1	Evaluation metrics	18
9	Related work and documentation (W_{12})	19
9.1	Related approaches	19
9.2	Advantages and limitations of your solution	19
9.3	Possible extensions of the current work	19
10	Project demo and documentation (W_{13})	20
10.1	Exercises	20
11	Results dissemination and feedback (W_{14})	21
11.0.1	Public presentation	21
11.0.2	Self-assessment	22
11.0.3	Formative feedback	22
11.0.4	Problem-based learning	22

A Your original code	23
B Quick technical guide for running your project	24
C Check list	25

Chapter 1

Instalarea programelor (W_2)

În cadrul acestui proiect este folosit programul *LibSVM* pentru clasificarea email-urilor. Pentru rularea proiectului este necesară instalarea mașinii virtuale Java, deoarece prelucrarea și transformarea datelor în formatul numeric înțeles de *LibSVM* este realizată de un program scris în limbajul Java.

În continuare se prezintă metoda de instalare a tool-ului *LibSVM* și a pachetului Oracle JDK (*Java Development Kit*) pentru sistemul de operare Ubuntu 16.10 64, versiunea pe 64 de biți (Oracle JDK este preferat față de versiunile Open JDK pentru performanță și compatibilitate maximă).

Programul *LibSVM* se poate descărca de la adresa <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>. După finalizarea descărcării, fișierul descărcat poate fi dezarhivat folosind comanda

```
tar -xvzf nume_fisier
```

După dezarhivare, trebuie să navigăm în interiorul fișierului dezarhivat cu comanda `linux cd nume_fisier_dezarhivat`. Pentru a compila programul, trebuie doar să introducem comanda `make`. După executarea compilării vor fi generate următoarele executabile:

Nume fișier executabil	Descriere
<code>svm-scale</code>	Programul care scalează datele de intrare pentru a asigura acuratețe cât mai mare
<code>svm-train</code>	Programul care generează un model pe baza exemplelor furnizate (învață)
<code>svm-predict</code>	Programul cu ajutorul căreia putem rula testele pe baza modelului generat prin învățare

În continuare se prezintă instalarea pachetului *Java Development Kit*. Acest pachet conține atât mașina virtuală Java cât și compilatorul Java respectiv biblioteca standard a limbajului. JDK poate fi descărcat de pe pagina web

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Trebuie selectată versiunea destinată pentru sistemul de operare Linux pe 64 de biți (*x64*) cu extensia *tar.gz*. După finalizarea descărcării fișierului, acesta trebuie dezarhivat cu ajutorul comenzii de dezarhivare folosite și la *LibSVM*. Fișierul dezarhivat trebuie copiat în directorul `/usr/lib/jvm` cu ajutorul comenzii:

```
mv nume_fisier /usr/lib/jvm
```

Următorul pas este să configurăm mașina virtuală Java instalată în acest mod ca și cel de bază al sistemului. Prima dată trebuie să comunicăm sistemului de operare instalarea cu ajutorul comenzilor

```
sudo update-alternatives --install  
/usr/bin/java java /usr/lib/jvm/nume_fisier/jre/bin/java 2000
```

```
sudo update-alternatives --install  
/usr/bin/javac javac /usr/lib/jvm/nume_fisier/bin/javac 2000
```

Apoi trebuie să configurăm acestea. Trebuie selectat în fiecare caz componenta nou instalată în listele afișate după rularea comenzilor

```
sudo update-alternatives --configure java  
sudo update-alternatives --configure javac
```

Chapter 2

Rularea și înțelegerea exemplelor (W_3)

Support Vector Machines (SVM) este o tehnică des folosită în domeniul inteligenței artificiale pentru clasificarea datelor. SVM este de fapt o colecție de algoritmi de învățare (machine learning) care sunt folosite pentru clasificarea datelor.

O particularitate a SVM-ului este că algoritmi pot prelucra doar date numerice. Astfel în cazul în care se dorește clasificarea unei date de alt tip (de exemplu text), acesta trebuie transformat într-o reprezentare numerică.

În cadrul acestui proiect va fi folosit programul *LibSVM* pentru clasificarea email-urilor. Acesta este o implementare răspândită a SVM. Deoarece proiectul se concentrează pe clasificarea texturilor folosind algoritmi SVM, în clasificare se vor lua în considerare doar subiectele email-urilor. Aceste șiruri de caractere sunt prelucrate și transformate de către un program Java în format numeric ținând cont de caracteristicile semnificative ale șirurilor de caractere. Valorile obținute trebuie scalate folosind `svm-scale` pentru a asigura o acuratețe cât mai mare a rezultatelor. Scalarea constă din aplicarea unei funcții de transformare liniară a numerelor din intervalul $[a, b]$ în intervalul $[0, 1]$ sau $[-1, 1]$ obținute prin convertirea șirurilor de caractere.

Datele obținute după scalare sunt folosite pentru a antrena rețeaua SVM (învățare), care după această etapă generând un așa numit *model* este capabil să clasifice noile date de intrare. Este important ca datele de intrare să fie generale, conținând cât mai multe exemple din fiecare clasă (clasele sunt acele tipuri în care SVM trebuie să clasifice șirurile de caractere prelucrate).

Datele de antrenare ale *LibSVM* au următorul format:

```
label index1:value1 index2:value2 ...
```

`label` este un număr întreg care indică numărul clasei de care aparține data respectivă. `indexi` sunt valori întregi în ordine crescătoare care reprezintă numărul caracteristicii, iar `valuei` este numărul real care indică caracteristica respectivă a datei curente. Altfel spus data curentă are valoarea `valuei` la caracteristica `indexi`.

Datele de intrare și de test au formatul prezentat, și sunt separate printre ele prin caracterul `'\n'`. Deci fiecare dată trebuie să fie conform formatului prezentat și să fie amplasată pe un rând al fișierului de intrare pentru învățare. Nu este permisă amplasarea mai multor date pe același rând.

Un exemplu ușor de înțeles este cel în care dorim să clasificăm propozițiile din punctul de vedere al complexității acestora. Avem deci propoziții simple și compuse. Luăm în considerare de exemplu caracteristici precum:

- numărul virgulelor din propoziție
- numărul total al caracterelor

- numărul de apariții al cuvintelor de legătură (de exemplu: *și, respectiv, așadar, totodată* etc.)

Numărul virgulelor va apare la indexul 1, numărul caracterelor la indexul 2 și numărul de cuvinte de legătură la indexul 3. Considerăm că propoziția simplă are eticheta clasei -1 și cea compusă eticheta clasei 1.

Astfel propoziția *Răzvan merge la școală.* este reprezentată prin linia

```
-1 1:0 2:23 3:0
```

Identic propoziția *Anca este harnică, așadar ea învață în fiecare zi.* este reprezentată prin linia

```
1 1:1 2:50 3:1
```

După crearea fișierului de antrenare (denumit `input.txt` care conține date precum cele două enumerate mai sus), se antrenează SVM-ul care va rezulta un fișier denumit `model` folosind comanda

```
./svm-train input.txt model
```

Apoi se creează fișierul de test (`test.txt` care are aceeași forma ca și fișierul `input.txt`). Apelând predicția pe acest fișier *LibSVM* va determina de care clasă (-1 sau 1) aparține fiecare propoziție și apoi va compara rezultele obținute cu cele date (autoevaluare). Astfel se calculează acuratețea. Predicția se rulează folosind comanda

```
./svm-predict test.txt model output.txt
```

Fișierul text generat `output.txt` conține pe fiecare linie numărul clasei de care aparține propoziția codificată prin caracteristici de pe aceeași linie din fișierul `test.txt` conform SVM-ului.

Chapter 3

Understanding conceptual instrumentation (W_4)

The teaching objectives for this week are:

1. To understand the algorithm(s) on which your tool relies.
2. To get used with writing algorithms in Latex

Latex provides various environments for writing algorithms, including: `algorithm`, `algorithmic`, `algorithmicx`, `algpseudocode`, `algorithm2e`.

The following example algorithm has been taken from [?].

3.1 Exercises

1. Big O complexity
2. Which are the latex options to write algorithms? Describe in one paragraph the main features of one such package for algorithms.

Solution to exercise 1

Solution to exercise 2

Chapter 4

Descrierea proiectului (W_5)

Proiectul de față are titlul *Clasificarea email-urilor*. Așa cum sugerează și titlul scopul aplicației dezvoltate este filtrarea și clasificarea email-urilor. Se consideră următoarele două clase de email-uri:

- email-uri cu caracter de reclamă (advertising email)
- alte email-uri

Email-urile trebuie prelucrate automat de către program. Pentru aceasta se dezvoltă o aplicație Java care folosește biblioteca *JavaMail*. Acesta se conectează la contul de email al utilizatorului folosind conexiunea internet și descarcă subiectele email-urilor necitite.

Programul Java pe lângă descărcarea email-urilor care trebuie analizate mai are o componentă. Aceasta transformă șirurile de caractere (subiectele fiecărui email) într-o reprezentare numerică înțeleasă de *LibSVM*. În această transformare trebuie să se țină cont de caracteristicile relevante ale șirului de caractere în privința clasificării în cele două categorii. De exemplu cuvintele precum *preț*, *reducere* sau *ofertă* mai mult ca sigur aparțin unei reclame.

Rețeaua SVM este antrenată cu ajutorul email-urilor personale recepționate de-a lungul timpului. Setul de date trebuie să cuprindă cca. jumătate de email-uri reclamă și cealaltă jumătate alte email-uri. Acestea vor fi convertite automat folosind programul scris în Java dar necesită și intervenție umană pentru a le clasifica.

Pe baza cunoștințelor asimilate prin învățare SVM va putea decide dacă un email nou este sau nu o reclamă. Pentru a îmbunătăți acuratețea programului (rata de clasificare corectă), înainte de etapa de învățare, caracteristicile trebuie scalate (din intervalul de care aparțin acestea se vor transforma printr-o funcție liniară în intervalul $[0, 1]$).

Scopul aplicației este deci clasificarea email-urilor în cele două categorii amintite. Această operație poate fi efectuată cu ajutorul antrenamentului prin email-urile personale primite de-a lungul timpului de la diferite market-uri online (de exemplu *emag.ro*, *cel.ro*, *flanco.ro*) și alte email-uri care aparțin clasei secunde.

Datele de intrare ale aplicației sunt:

- adresă de email
- parola contului

Datele de ieșire sunt două liste de email-uri: o listă care conține emailurile necitite reclamă și cealaltă listă populată cu email-urile care nu sunt reclame.

Diagrama bloc al aplicației dezvoltate este reprezentată în figura 4.1.

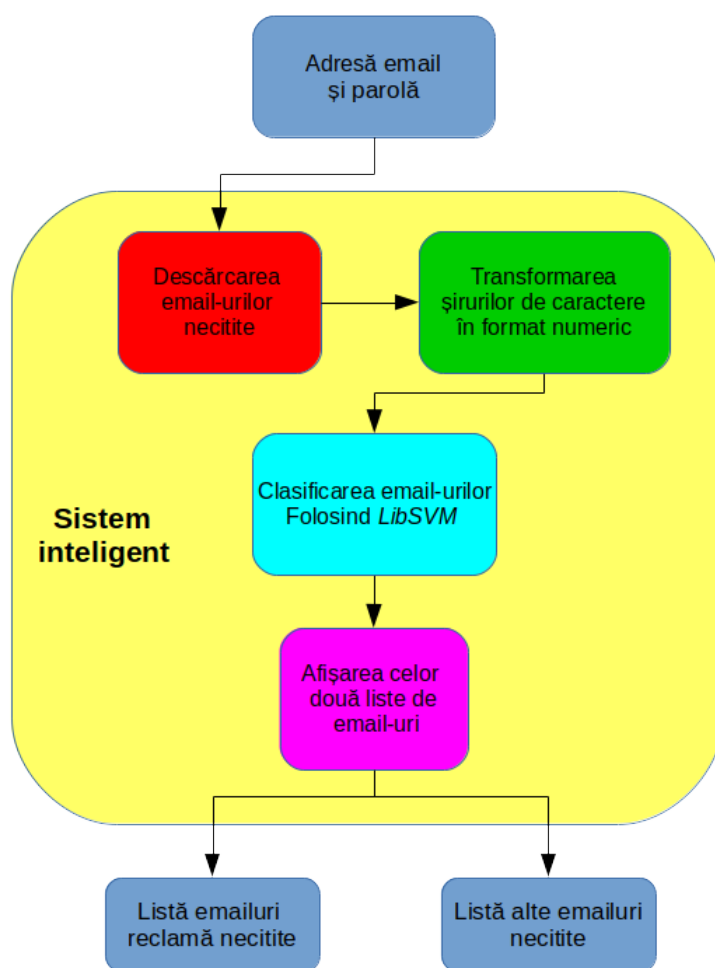


Figure 4.1: Diagrama bloc a aplicației

Chapter 5

Preliminary results (W_7)

This section corresponds to the midway report in week 7. The teaching objectives for this week are:

1. To prove that you have managed to write few lines of code of your own.
2. To prove that the knowledge or data required are already obtained.

These objectives decreases the risk to fail. You should be aware that failing to meet the above objectives in week 7 indicates high risks in obtaining relevant results at the end of the semester. Take urgent measures to overcome these difficulties.

5.1 Exercises

1. Write the preliminary results explaining any realizations or insights found during the research of the subject.
2. Discuss new information and questions found during the domain investigation or during coding.

Chapter 6

Implementation details (W_9)

The teaching objectives for this week are:

1. Illustrate each aspect of the reality that you have modelled in your solution.
2. To explain the relevant code from your scenario.

My personal objectives for this class are:

- 1.
- 2.

Projects in artificial intelligence consist of developing new solutions.

6.1 Relevant code

Provide the relevant code (see an example in Fig. 6.1). You can use "verbatim" package or "listing" package. Complement the code with its corresponding textual description.

The eager student may use concepts from *literate programming*.

6.2 Common bad practice in AI undergraduate projects

The over-estimated AI programmer.

Bad practice: Excepting few genial students, you tend to overestimate your AI-programming abilities. That is, you start to write a large amount of code. (Here large might be 20 lines). When testing it, nothing run. You start to debug a line or to remove it. Your program will not run this time too. You remove or comment another line. And so on, until you have a single line of code. If you are lucky, that could run. But you lose a lot of time in this enterprise.

Solution: In the early stage of writing code, write a line of code and test it. If it works, write another line and test it. And so on. That is, you are exploiting the interactive environments

```
(full-reset)
(instance a Argument)
(related a b attacks)
(concept-instances Argument)
```

Figure 6.1: Modelling arguments in Racer.

provided by AI tools or languages like LISP and PROLOG. You should hold your horses and have the most possible skeptical attitude towards your code. As you get experience, you will be noticing that writing AI-declarative code is more effective than procedural one.

The eyewash bug. *Bad practice:* You spend most of your programming time to develop a GUI for your AI-system. Don't bother. I am sympathetic with Sania Twain's view on GUIs: "You don't impress me much". Such things are indeed important in computer science, but not relevant in this AI class.

The not-organised student. You are not organised, if something like this will happen to you:

- You do not find your project and yield "Someone removed my project!". Most of the time you are logged with a different user as usual. Check this with `who am i`. This is not a rhetorical question, but a Linux command.
- You are working in a different directory. Type `pwd` and `ls` to check that your executables are indeed in the current working directory. If you have been lazy to set your PATH variable, you might just forgot to type `./` for executing the command in the current directory.

The omniscient student. You are in this category if you fail to add references. Reading relevant references is mandatory to deliver a decent project.

6.3 Exercises

1. What latex packages can be used to format code?
- 2.

Solution to exercise 1

Solution to exercise 2

Chapter 7

Tool expressivity (W_{10})

The teaching objectives for this week are:

1. Describe each technical instrumentation provided by the tool that was enacted in your implementation.

My personal objectives for this class are:

- 1.
- 2.

Chapter 8

Graphs and experiments (W_{11})

The objectives for this week are:

1. To describe and interpret each experiment that you have performed

My personal objectives for this class are:

- 1.
- 2.

An experiment investigates how some variables are related. Usually, experiments verify a previously formulated hypothesis. Such hypothesis may investigate how your software degrades its performance with larger inputs. You will need to run simulations to see how your implementation is affected by different inputs.

Note that running experiments mean more than testing your solution. It helps to describe and prove how did you test your implementation. Moreover, during this lab, you will often need to: 1) generate random data for your algorithms, 2) measure their performance (number of operations, execution time), 3) draw charts, 4) interpret the obtained results.

The eager student might want to take a look at literature on how to design computer experiments, such as [?]. Section 5.6 from [?] might be of particular interest for some of you. If your experiments include a stochastic parameter, you need to include a test for statistical significance. This is important to prove that your outputs are not a random effect.

You should develop a test suite that can be used to show your code works correctly under a various conditions/problems/scenarios.

8.1 Evaluation metrics

Graphs always impress teachers...

Figure 8.1: Increasing the accuracy with the number of samples.

Chapter 9

Related work and documentation (W_{12})

The teaching objectives for this week are:

1. To compare your results to related work.
2. To discuss the advantages and limitations of your solution.
3. To deliver a professional documentation of your work.

My personal objectives for this class are:

- 1.
- 2.

This chapter convinces me that you know how your work fits into the larger domain area.

9.1 Related approaches

You need to support your opinions with trustworthy evidence and references. You need also to decide how your problem fits into a wider context. The quality of your reference is a strong indicator that you managed to scrutinise different perspectives on the topic. Proving understanding of the references is the foundation of a good grade. By start coding without reading relevant references, you will most probable write something irrelevant for the application domain.

Identify and describe other solutions for solving the same (or similar) scenario like yours.

9.2 Advantages and limitations of your solution

This part of the conclusions chapter should be an evaluation of your work.

9.3 Possible extensions of the current work

Chapter 10

Project demo and documentation (W_{13})

The teaching objectives for this week are:

1. Deliver the technical documentation of your project
2. Demonstrate your running scenario to the instructor

My personal objectives for this class are:

- 1.
- 2.

Demonstrate in 4-5 minutes your running scenario to the instructor. The demo should take place on a Linux distribution

From your final report, remove the text/examples/algorithms/rules/bibliographic references/etc - keep only your notes. If the documentation does not meet minimum standard for lisability and scientific discourse, it will be classified by the furious teaching assistant as unacceptable and therefore rejected.

10.1 Exercises

1. How you would advocate your project to a possible client?
2. Write five highlights of your results (maximum 85 characters including spaces).
- 3.

Solution to exercise 1

Solution to exercise 2

Chapter 11

Results dissemination and feedback (W_{14})

The teaching objectives for this week are:

1. To practice public presentation.
2. To get used with the beamer template for making scientific presentations
3. To get feedback from your colleagues.

My personal objectives for this class are:

- 1.
- 2.

Learning is enhanced by constructive feedback on the strong/weak points of your performance during AI laboratory. This feedback focuses on the scientific relevance of your results and it aims to complement the feedback encapsulated in the grade. Do not take criticism personally. It is the project that is being criticised, and not your competence or intelligence.

11.0.1 Public presentation

10 slides in beamer format for a timeslot of 5 minutes presentation plus 5 minutes questions. Questions may be posed by your colleagues or the teacher.

One of the main difficulties when designing slides is how to find the right amount of technical details to be included. No technical details rise the question of "bla bla story telling". Too much technical details may bore the audience and also you may fail to fit within the time assigned.

Two introductory tutorials on beamer are [?] and [?]. The beamer manual is:

Presentation template

During presentation, it is a mistake to focus on the tool that you have been used. During 5 minutes, you have to focus only on your results and to market your work. Don't forget to include technical details and graphs.

A method for disseminating your research consists of writing 3-5 highlights. Highlights consist of a set of bullet points that convey the core findings of your work. For examples, see <http://www.elsevier.com/highlights>.

You are now playing the role of your project advocator. Always keep in mind that you have to market your project only, and not the tool that you have relied on.

Aspect	Self-assessment
How did you manage to master the tool?	
How realistic was your scenario?	
Relevance of the running experiments	
Knowledge and skills achieved	
Capacity to market your effort and results through documentation and presentation	

Table 11.1: Self-assessment. Assess each aspect, with: enthusiastic, satisfactory, unsatisfactory, bad

11.0.2 Self-assessment

‘What did you do well? Give examples’

‘Where do you think the assignment is weak?’

11.0.3 Formative feedback

The last week is an opportunity for interested students to obtain a formative feedback. This is not an opportunity to negotiate your grade. In the previous week you had your chances to advocate your work.

11.0.4 Problem-based learning

One scope was to engage you in a kind of self-directed learning. The rationale is that you are more heterogeneous and you have different learning experiences and maturity levels. The focus was not on mastering AI algorithms but to apply them in practice. By practice I mean realistic scenarios. Ideally, you should have developed more awareness of the social, environmental, economic aspects of a real problem.

Appendix A

Your original code

This section should contain only code developed by you, without any line re-used from other sources. This section helps me to correctly evaluate your amount of work and results obtained. Including in this section any line of code taken from someone else leads to failure of IS class this year. Failing or forgetting to add your code in this appendix leads to grade 1. Don't remove the above lines.

Appendix B

Quick technical guide for running your project

Requirments

Step by step technical manual

Appendix C

Check list

1. Your original code is included in the Appendix .
2. Your original code and figures are readable.
3. All the references are added in the Bibliography section.
4. All your figures are referred in text (with command `ref`), described in the text, and they have relevant caption.
5. The final documentation describes only your project. Don't forget to remove all tutorial lines in the template (like these one).
6. The main algorithm of your tool is formalised in latex in chapter 3.

Bibliography

Intelligent Systems Group

