Departamentul de Calculatoare şi Tehnologia Informaţiei
Universitatea Tehnică din Cluj-Napoca

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

Chapter 1

Instalarea programelor (W_2)

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

În continuare se prezintă metodade instalare a tool-ului *LibSVM* şi a pachetului Oracle JDK pentru sistemul de operare Ubuntu 16.10 64, versiunea pe 64 de biţi.

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

```
tar -xvzf nume\_ fi\c sier
```

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

Nume fişier executabil	**Descriere**
`svm-scale`	Programul care scalează datele de intrare pentru a asigura acurateţe cât mai mare
`svm-train`	Programul care generaează un model pe baza exemplelor furnizate (învaţă)
`svm-predict`	Programul cu ajutorul căreia putem rula testele pe baza modelului generat prin învăţare

În continuare se prezintă instalarea a *Java Development Kit*. Acest pachet conţine atât maşina virtuală Java cât şi compilatorul Java respectiv biblioteca standard al 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 x64, cu extensia *tar.gz*.

După finalizarea descărcării fişierului, acesta trebuie dezarhivat cu ajutorul comenzii de dezarhivare folosite şi la *LibSVM*. Fişierul dezarhivat trebuie copiat în directorul `/usr/lib/jvm` cu ajutorul comenzii:

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

Următorul pas este să configurăm noul *jvm* instalat ca 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 configuram acestea cu ajutorul comenzilor

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

Trebuie selectat în fiecare caz componenta nou instalată.

Chapter 2

Rularea şi înţelegerea exemplelor (W_3)

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

O particularitate a SVM-ului este că algoritmii pot prelucra doar date numerice. Astfel în cazul în care se doreştea clasificarea unui 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 SVM, în clasificare se vor lua în considerare dor subiectele email-urilor. Aceste şiruri de caractere sunt prelucrate şi transformate de către un program Java în format numeric. Valorile obţinute trebuie scalate folosind `svm-scale` pentru a asigura o acurateţe cât mai mare a rezultatelor.

Datele obţinute după scalare sunt folosite pentru a antrena reţeaua SVM, care după această etapă generând un model este capabil să clasifice noile date de intrare. Este important ca datele de intrare să fie generale, conţinând cât mai multe exemple din fiecare clasă.

Datele de antrenare ale *LibSVM* au următorul format:

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

`label` este un număr întreg care indică numărul clasei de care aparţine 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 şi de test au formatul prezentat, şi sunt separate printre ele prin caracterul `'\n'`. Deci fiecare dată trebuie să fie conform formatului prezentat şi să fie amplasată pe un rând al fişierului de intrare pentru învăţare.

Un exemplu bun este cel în care dorim să clasificăm propoziţiile din punctul de vedere al complexităţii acestora. Avem deci propoziţii simple şi compuse. Luăm în considerare de exemplu caracteristici precum:

- numărul virgulelor
- numărul caracterelor
- numărul de apariţii ai cuvintelor de legătură (de exemplu: *şi*, *respectiv*, *aşadar*, *totodată* etc.)

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

Astfel propoziţia *Răzvan merge şi la şcoală.* este reprezentată prin linia

```
-1 1:0 2:26 3:1
```

Identic propoziţia *Anca este harnică, aşadar ea învaţă în fiecare zi.* este reprezentată prin linia

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

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

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

Apoi se creează fişierul de test (`test.txt` care are aceeaşi forma ca şi fişierul `input.txt`). Apelând predicţia pe acest fişier *LibSVM* va determina de care clasă (-1 sau 1) aparţine fiecare propoziţie şi apoi va compara rezultele obţinute cu cele date. Astfel se calculează acurateţea. Predicţia se rulează folosind comanda

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

Fişierul text generat `output.txt` conţine pe fiecare linie numărul clasei de care aparţine propoziţia codificată prin caracteristici de pe aceeaşi linie din fişierul `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 algorihms, 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

Project description (W_5)

The teaching objectives for this week are:

1. To have a clear description of what you intend to develop.
2. To point to specific resources (datasets, knowledge bases, external tools) that support the development of your idea and which minimise the risk of failure.
3. To identify related work (articles) that are relevant or similar to your approach.

My personal objectives for this class are:

1.
2.

To encourage the development of AI skills, you were required to come up with a significant semester project You have to apply ideas from the course to a problem of your own.

Which domain to choose is a decision that only you can make. The more aware of the tool capabilities, the more adequate the decision. Realistic and original scenarios are encouraged. Well known toy problems (salesmen, map colouring, logistic planning, wumpus, sudoku, queens, missionaries and canibals, etc.) do not worth much for your grade. Your scenario should be realistic and should be business oriented.

Select clearly defined problems and not generic ones (i.e I will do something in the medical domain). Note that the focus is both on programming and on modelling the reality into a formal representation.

Before specifying your project you must understand as much as possible about the application domain (medicine, bank, human resource management, etc). You must also understand functionality required by the stakeholders of your system. Let the problem drive the modelling - the more you understand the domain, the more technical solutions need to be solved by you.

Consider answering to the following questions:

1. What will your system do?
2. Which is the scope of coverage your system aims for?
3. What will be the input of your program?
4. What will be the output of your program?
5. What will be the knowledge of your system?

6. Which would be the narrative description of running scenario(s)?

7. Which are the stakeholders of your system?

8. Which are the asumptions?

Example 1 (What will system do)

Example 2 (Scope of the program)

In this problem-based model you learn what you need to know in order to solve a problem. However, note that you have total flexibility in stating your project objectives. For instance, if you consider that studying more than one computational technology (or AI algorithms) brings more benefits, you are encourage to do it. You just have to frame your task under one scenario umbrella. One example: you can investigate the problem of *fake review detection* with various machine learning algorithms: decision trees, nayve bayes, neural networks, ensemble learning. This road helps you to study and compare the above algorihtms on your own problem. A second example: you can help a *robot to escape from a maze* with various search algorithms (A*, greedy, deep first, uniform cost) or computation technologies (constraint satisfaction problems, planning, searching with observations). Hence, both the problem-based model or a more algorithmic approach to AI (if formulated under the umbrella of a single scenario) are accepted for this laboratory.

4.1 Narrative description

This should be a simple textual description of your scenario. You should explain your project objectives. Put them all together in half a page.

4.2 Facts

You start the analyze of the problem by identifying the relevant facts from the scenario. This fact-identification step helps you to represent the problem.

4.3 Specifications

List of specifications. Use your knowledge from "System Engineering" on how to write specifications and requirments (see Exercise 1).

4.4 Top level design of the scenario

Technical description of your scenario. In some cases, you may provide a figure with the architecture of the system.

4.5 Knowledge acquisition

You should be aware that computing interacts with many different domains. Solutions to many AI problems require both computing skills and domain knowledge.

First, you should ask yourself if you have the necessary background and resources to do a project in the chosen area.

How do represent knowledge? Your system relies on a knowledge base. You have to describe how do you represent this knowledge. You might choose between different logics: propositional logic, first order logic, modal logics, description logics, epistemic logics, temporal logics, and so on.

Where are you getting the required knowledge/data Point towards the knowledge bases that you plan to exploit. The existence of these sources are required to prove that your approach is realistic.

Examples of knowledge sources include:

- Data sets: i.e., https://archive.ics.uci.edu/ml/datasets.html
- Statistics: i.e., http://ec.europa.eu/eurostat
- Ontology repositories in OWL or RDF format.

If you will be using books, give their reference. If you hope to exploit people for elicitation, give their names. If you aim to use data sources or knowledge repositories list them and be sure that you have access to the needed knowledge. Indicate what you have accomplished so far in knowledge acquisition.

4.6 Related work

You have to identify articles or conference papers relevant to your scenario. Searching for adequate references can be both rewarding and frustrating.

Browse online libraries like:

- Science Direct
- IEEE Computer Society Digital Library: IEEEXplore
- SpringerLink: http://www.springerlink.com/computer-science/
- ACM Digital library: http://portal.acm.org/dl.cfm

A valuable resource is Google Scholar. Some references may be frely available on ResearchGate.

Looking at the examples remains the best ways of learning how to present a literature analysis. Eacy article does include such section.

Obtain the `.bib` file of each article that you will rely on. Cite and very briefly describe the main idea of each paper that you have read. Save the most relevant related papers in a local directory. Use your own words. Don't use automatic translation tools (i.e., Google translate) just to expand your documentation. You should already be aware that this is a form of cheating. Everything in your project that does not come with a citation is assumed to be your own work.

The following is an example of such `bib` structure:

```
@article{bench-capon:argumentation-in-ai,
author = {Bench-Capon, Trevor J. M.  and Dunne, Paul E. },
title = {{A}rgumentation in {A}rtificial {I}ntelligence},
journal = {Artificial Intelligence},
volume = {171},
number = {10-15},
year = {2007},
```

```
 issn = {0004-3702},
 pages = {619--641},
 doi = {http://dx.doi.org/10.1016/j.artint.2007.05.001},
 publisher = {Elsevier Science Publishers Ltd.},
 address = {Essex, UK}
}
```

Don't forget to include the above structure in your `.bib` file. Then, generate the `.bbl` file in order to correctly appear in the *Bibliography* section.

4.7 Exercises

1. Sum up what is the aim of your project in a Twitter-sized phrase (140 characters)
2. Recall or identify an engineering methodology to write specifications. Cite this methodology and employ it for specifying your project.
3. Which are the differences between requirments and specifications?
4. Identify similar scenarios proposed by your collegues. Think at some form of collaboration with one of your collegues having similar interests.

Solution to exercise 1

Solution to exercise 2

Solution to exercise 3

Solution to exercise 4

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, your 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 your 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 cathegory 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 previosly 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 ouputs 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/scenarious.

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 irelvant 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 collegues.

My personal objectives for this class are:

1.

2.

Learning is enhaced 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 critisised, 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 collegues or the teacher.

One of the main difficulties when designed 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". To 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 yor 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 rhave 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, satisfacatory, unsatisfacatory, 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 negatiate your grade. In the previous week you had your chanches 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 scenarious. 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