Hands-on Semester Project: Introduction, Instruction and Hints, Grading Criteria

Herbert Jaeger, April 25, 2020. Update 1: June 12, 2020. Update 2: April 23, 2021. Update 3: November 11, 2021. Update 4: Feb 21, 2022. Update 5: March 26, 2022. Update 6: October 16, 2022. Update 7: 14 Jan 2023. Update 8: 18 Feb 2023. Update 9: 12 April 2023. Update 10: Sep 20, 2023. Update 11: Sep 27, 2023

Overview

The practical project is an important part of this course. In this project you will choose, design, implement, test and document an entire pipeline of a machine learning project.

The project will be done in teams of 4. Finding or building yourself a team, learning some startup essentials of machine learning, and choosing a project topic will take 3-4 weeks at the beginning of the course – thus the "real" project work will start only thereafter.

The semester project ends with documenting your work and results in a report. This report will be carefully read, commented and graded, and the grade counts by 50% toward the final course grade.

Carry-over of project grade from last year

If you repeat this course and got a partial grade for the semester project last year, you do not have to do the project again, and you can carry over that grade to this year. If you wish to do so, inform me about this before the end of the fourth week of this course; if I don't hear from you until then, I assume that you do not want to carry over and that you want to do a semester project again in this course.

Finding a group

This is self-organized. There is a discussion forum on Brightspace (you find it under "Communication") intended to work as a bazaar for advertising group forming initiatives and group search requests. You should aim to have joined a team of exactly four members by the end of week 3.

Choosing your project topic

You can either pick one of the four ready-made project topic suggestions that you can find on the "Semester Project" Brightspace page, or define your own topic. These four suggestions are just that, *suggestions*. Whether you go for one of them, or modify one of them, or define your own theme entirely – it's up to you. In any case, make sure that at the outset you clearly define *goals* (*"objectives"*) for your project, so that you know what you want to achieve.

If you opt for a ready-made theme, not much needs to be said. If you choose your own topic, here are some pieces of advice:

- **Don't over-reach**. You only have a short time for your project. All that is expected for a very good grade is that you demonstrate that you can define, design, implement, run, evaluate and discuss a machine learning / neural network processing pipeline aiming at some interesting project objective, starting from a clear statement of the task that you want to solve. This task should be realistically doable in a few weeks by 4 group members working together and sharing their competences.
- Machine learning / neural networks is a dynamical, open-ended, diverse field and there are many intriguing "real research" questions that could be made the topic of a semester project. However, go for an "original research" project only if your group is composed of 4 partners who already know a lot about machine learning and if you have a clear project strategy right at the beginning. For the great majority of participants I would strongly advise not to think of this semester project as a "research" project, but as **a "lab exercise" project** – carrying out a mundane modeling or model comparison task of the kind that one is likely to get assigned when working in a company. The most common real-world bread-and-butter type of task for a neural network is *pattern*

classification (static or temporal patterns), a *supervised* learning task. Thus a standard real-worldish bread-and-butter project would typically be schemed like this:

- Find an application domain which your group finds interesting (Finance? Robotics? Astronomy? Social networks? Medical? Industrial process engineering? Environmental?)
- Determine a relevant classification (or prediction) task in that domain. Consider some options for reasonably doable looking tasks and search the web for existing datasets that you could use. Hint: the blog entry https://towardsdatascience.com/top-sources-for-machine-learning-datasets-bb6d0dc3378b gives a most useful overview of online ML dataset resources. The task that you specify must be matched by an available dataset, so this step may require some search.
- Solve that task by designing, implementing, testing, and documenting a full processing pipeline from data cleaning (if needed), through feature extraction or dimension reduction (if needed), data augmentation and other preprocessing (if needed or deemed useful), selecting the right sort of learning architecture (neural network or other) that makes sense for this task and dataset, choosing a good loss function and regularizations scheme, hyperparameter optimization through cross-validation, and possibly postprocessing --- in short, all the things you learnt in the Machine Learning course.

How (not) to get a good grade: important recommendations

In past courses where similar projects have been done I have witnessed an increasing trend to do these projects in a way that (i) is encouraged by the apparently easy use of high-caliber toolboxes, (ii) means a lot of un-interesting work, and (iii) is bad for good grades. This unfortunate project strategy goes like this:

- Pick some interesting dataset and some modeling task (prediction, classification…)
- Try to solve this task with several different ML methods (from linear regression through SVNs, decision trees to MLPs and CNNs), using high-end toolboxes for each of these – I often see 3 or 4 different high-end methods thrown at the data
- "sell" the project as a comparison study of the relative merits of these different methods.

This often (even almost always) leads to studies where each of the methods is applied without much optimization (using default toolbox settings for hyperparameters, with none or only one sort of data preprocessing that is not separately tuned toward the different ML methods), and where each method is described only superficially in a way that reveals an incomplete or misinformed understanding of the respective method. This is not what I want to see and it typically does not give good grades.

What instead I would want to see, and can give good grades:

- Choose only one ML/NN method that is appropriate for the task (plus, recommended, a very elementary baseline method like linear regression or kNN)
- Carefully optimize data preprocessing and possibly postprocessing in a way that is specially suited for the chosen modeling method (which requires insightful intuitions about what makes the chosen dataset difficult or easy)
- Carefully optimize the hyperparameters of the chosen architecture (or, if that is not possible because hyperparameters search would be too expensive, give plausible arguments why you chose the hyperparameters that you used)
- Describe the chosen method in enough detail and correct use of formalism that I get the impression that you really understood it.

Programming language and tools

You are entirely free here. Today's standard programming language for machine learning / neural networks is Python, and there are well known toolboxes and interfaces available, which you can use as you wish. However, I must speak out a little warning here: if you use ready-made toolboxes, like for example TensorFlow, I expect that you understand what the algorithms that you use are actually doing. Not in every detail (these professionally implemented machine learning algorithms are super complex), but in principle. The essential idea or mechanism behind your tools must be described in the report. If

from your write-up I get the impression that you just ran a black box miracle machine without actually having insight why or how it worked, this will lead to a grade reduction.

Matlab is a good alternative to Phython if you do not want to use ready-made toolboxes but program your code from scratch – Matlab is a commercial product and the user interface, especially the graphics functionalities, is more convenient to use than in Python.

Steven Abreu will offer an intro tutorial on using Python-based toolboxes in week 3 or 4.

Deliverables and grading

Please write a project report and submit this together with your reasonably commented code (or a link to your Git repository if you use that). If it is illustrative of your work, also submit supplementary materials (for instance, if you generated music in your project, some mpeg4 demo files). I will give detailed feedback on your report. I will base my grading only on the report by default, and will inspect the code only in cases of doubt.

A reasonable size for the report is 5-8 pages, 10 pt. font, not counting graphics, title page and references. Together with title page, graphics (use them a lot!) and references you will end up in a typical range of 10-15 pages. Note that it is easier and faster to write a somewhat longer report than it is to condense all the relevant material into a small number of pages in a still clearly readable way. Only very good technical writers can write very good very short reports.

The main grading criterion is the technical quality of your experimental work and the clarity of your technical writing, *not* the performance level of your final results. A report which demonstrates a clear experimental strategy with a transparent formal specification of your data, processing steps, optimization routine and quality assessment method can get a perfect grade even if the used ML model was very simple and the final results are disappointing. Conversely, super results that come out of work that is poorly documented or with methods that are apparently hardly understood by the authors will earn a poor grade.

Not using Latex gives a grade reduction of 0.5 pts (on the RUG scale from 1-10).

A perfectly insightfully done and transparently documented project gets a 9.0 grade even when it is "only" a totally standard routine job and the results aren't sparkling. Only for a 10.0 grade I expect some significant "extra", like an insightfully analyzed comparison of several methods, an original research question, or nontrivial (statistical or conceptual) analyses of results.

A compilation of essential advice for good technical writing is in Appendix A.

The Excel sheet that I use for grading is appended at the end of this document (Appendix B).

Use of ChatGPT and other such large language models in report writing

After ChatGPT has come into the world, the rules of the games of report writing and report grading have changed. I think that a conscientious and well-informed use of such tools will become standard practice in technical writing, and students should exercise to apply these tools in a diligent way, and consider this a professional skill. Here is how I will deal with this new situation.

1. I do not forbid to use ChatGPT or similar tools in report-writing – I would even invite using them. Furthermore, it is just not possible to check (as a reviewer/grader) whether ChatGPT has been used.
2. If you use such a tool, the way how you used it must be documented. This does not lead to a grading penalty.

Timelines…

for assembling group members, deciding on a project theme, and submitting the final report are given on Brightspace (Semester project -> Timelines).

Don't stress yourselves out

I emphasize again that I do not expect or require scientific innovation or state-of-the-art performance from your project. This semester project is meant as a hands-on **exercise** where you go through the entire standard machine learning routine of on some interesting task with real-world data. I want you to experience all the steps from data preprocessing to deciding for a machine learning method and architecture to implementing to optimizing your system by cross-validation to documenting everything in a report. If all of that is done correctly and insightfully, super. The final results need not be sparkling at all – all I want to see is a systematic walk-through through the routines.

Thus, it is good enough for a perfect 9.0 grade if you choose a simple dataset and a simple neural network architecture, and do a clean and complete job with those.

A general **strategy**: in a first stage, use the simplest possible version of training data that you can procure, possibly even only mock data that you produce synthetically (no real data). And use a small model (training is fast and numerically robust). If you work with neural networks: All tasks that I can imagine can be done with simple MLPs. While LSTMs or other RNNs might be more elegant and powerful in some projects, MLPs are generally easier to work with. Get this simplest-possible set-up to work, including cross-validation. Then you already have something that would be good enough for the project report if carefully documented! If then ambition and time allow, you can expand the scope of your work (richer data, wider exploration of model types and architectures, more effort in finetuning).

It's also a good idea to parallelize work within the group. Specifically, in many cases it is possible that one subgroup takes care of data preprocessing, while another subgroup implements the learning machinery in a basic version and tests it for technical correctness with mock data. These two lines of work can initially very well be done separately.

Brightspace-based group management

On Brightspace under "Tools" you find "Groups", where in turn you find a collection of ready-made groups called "Semester project workgroups". Each of the groups has been created empty, ready for self-enrollment. Find yourself together in **groups of exactly four members, not more, not fewer**. Make sure that within your group there is consensus on which of the groups you enroll before you start the self-enrollment (avoid at all costs that members of one group start enrolling themselves into different groups). These groups are my main semester project management interface to you – in particular, this is where I see what groups actually have formed and what project topic they have chosen. You will also upload your final results in the "assignment" section of your group.

Assistance and coaching

We cannot and do not expect that do this project all on your own. We offer three main ways of assisting and consulting:

- Weekly walk-in gatherings in the times and places that you find as "Project support sessions" on Rooster where you and your team can ask the course-supporting PhD students and the TA for advice
- Individual consulting meetings of your group with one of us, by email arrangement for time and place (request such consultations by email to me (h.jaeger@rug.nl)
- Feel invited to always send us emails when you have specific questions that can be answered in an email. Please explain your problem with words and graphics, not by sending code that "somehow doesn't work". We will not do code inspection unless you are really really badly stuck. Email these questions to the entire support crew for this course (h.jaeger@rug.nl, s.abreu@rug.nl, g.a.pourcel@rug.nl, a.theuwissen@student.rug.nl, d.macrae@student.rug.nl, c.hadjichristodoulou@student.rug.nl) and one of us will quickly jump forward to help.

Cheers,

Appendix A: hints for writing your report in good technical writing style

1. The type of reports for semester projects.

Consider this report more of a *lab report* than a *scientific paper*. Specifically, you do not have to do a thorough literature research, which would be mandatory for a true scientific report. Your report should give a clear account of the goals that you set for yourself in your project, the processing pipeline that you design and implement (this includes a description of raw and preprocessed data, machine learning method and architecture, optimization criterion (loss function) and optimization algorithm, hyperparameter optimization through cross-validation), framed by a brief introduction and a not-so-brief discussion at the end where you summarize results, point out highlights and shortcomings of your work as well as options for future work (there will be surely many things you would have liked to do more or better in retrospect). Use mathematical formalism wherever precision in the documentation is needed. While a solid literature review on the scientific context of your project is not needed (though it wouldn't harm), do cite all technical references that you looked up for the machine learning methodologies that you implemented.

2. What I want to see (and not to see) in the reports.

I repeat myself, but that can't hurt: What I want to see is that you thought out, implemented and tested a ML pipeline *in a 100% insightful way*. This means that the report should document all your processing steps in enough detail to allow an informed reader to reproduce your experiments; and that all steps should be accompanied by some reasoning *why* you chose that specific design option. In this spirit, if you implement a single learning architecture – even a very basic one like just plain linear regression – but you do that with a full deliberation of data cleaning (if needed), possibly data augmentation, feature extraction, dimension reduction, choice of loss function and learning algorithm, optimizing data preprocessing options and hyperparameters and regularization with cross-validation (if applicable – not feasible for very large MLPs), discussion results (and statistical testing of significance if you compare different learning variants, or at the very least documenting standard deviations / error bars) – then that's perfect. What I do *not* want to see is that you throw some out-of-the-box procedures from a high-power toolbox at your data, use the default hyperparameter settings offered by the toolbox, get some sort of results (even good-looking ones), print them into your report and call it a good. Specifically, if you use some ML method that was not explained in the lecture, I expect that you explain the basic functioning principle of that method in a way that it becomes clear to the reader that you understood it. **Elementary methods treated in the ML lecture** (PCA, linear regression, cross-validation, k-means clustering, MLPs) **need not be re-explained as such in the report** because these can be considered generally shared knowledge.

3. Overall structure

A recommendable structuring goes like this:

1. A title page, including a list of team members who contributed (note that a grade will be given only to the members listed as authors)
2. A separate page with an abstract (10 lines) and a table of contents
3. **Introduction**: a statement of given or self-chosen task, including a description of the dataset used. You do *not* need to write a scientific-paper style introduction with an outline of scientific background, related work or research motivation (unless you were bold enough in your group to tackle a "real research" problem for the semester project). The introduction should also specify the objectives you had for your project, i.e. what you wanted to achieve and how to measure success.
4. **Data**: Describe the dataset that you are using. The reader should get a good intuitive grasp on them – best achieved by graphical representations. Data are the blood, flesh and bone of machine

learning – it is always good to describe the dataset nicely such that the reader of the report gets a good intuition about it.

5. **Methods and experiments:** This is the main technical part of the report. You describe all steps in the processing pipeline in formal detail (mathematical specification, or pseudocode, or at the very least: clean and precise technical English). The description should be detailed enough to allow an informed reader to *replicate* your experiments. In particular this means a precise documentation of your learning architecture, loss function, and the model optimization and training procedures (including cross-validation). An imprecise or incomplete description of methods is the most common reason for downgrading reports.
6. **Results:** Document the results of your final optimized architecture (also, if that is of interest, results of other architecture versions that you tried).
7. **Discussion:** what you learnt, why you think some things didn't work, what potential you see for improvement, … whatever flows from your heart after suffering through this project.
8. **References** list.

You can fine role models for how a very well-written report looks like on the Brighspace page *Semester Project*.

4. A quick guide for technical writing

Here is a list of Do's and Don'ts that you may find helpful as a checklist.

Using material from other papers or online sources

This is the most important point of all. Every import of wisdom must be credited by giving the source. This holds for word-by-word quotes (which MUST be put into quotes), for rephrased arguments, for graphics, for data. Even when you follow a source paper/book but re-word it entirely, you MUST acknowledge the source, for instance by beginning your import of material with something like „In the account given below, we follow the presentation given in [XY]“. Failure to cite sources is considered as violation of the code of academic conduct and will *automatically and irrevocably* lead to a grade reduction, up to zeroing the grade if the infringement is serious.

Intellectual honesty

Only make claims and write down equations that you understand. I frequently find equations taken out the hardly understood context of some paper, with not all symbols explained, and not well connected to the rest of the report. Bad for the grade. If you want to use a method that is difficult but for which ready-made program packages exist which you want to use – you are free to use them, but you should honestly explain the extent to which you understood them. Engineers working "out there" often use methods they don't understand. No-one blames them if the results are good: but if they are not good, it's likely that they didn't use the methods properly. If authors are aware of their gaps in understanding, and overtly describe their limitations, and document the details of how exactly they did employ the not-so-perfectly-understood methods, no reason for blame.

Never state something without a justification

EVERY statement in scientific writing must be backed up by a justification. This holds, in particular, for the introduction. The justification can take the form of a pointer to a reference (which is why well-written introduction sections often cite dozens of references); or your own reasoning; or a mathematical or experimental reasoning from your own working; or (dangerous) a phrase like „… it is well-known that …<claim follows>“.

English grammar and style

- If you are not firmly commanding of English grammar, pass your manuscript through the hands of a proficient speaker/writer friend. Poor English leaves the reader with an impression of lack of professionality and in real life will harm the impact of your work, or worse, make the reader quit reading. When I grade reports, bad English or more than a few typos/grammar errors is penalized by some grade reduction.
- Do not use colloquial short forms like „we’re“, „isn’t“ etc. Write them out: we are, is not, etc.

- Avoid colloquialisms and superlatives like „huge“, „enormous“, "sort of", "tremendously"etc. If you want to emphasize impact/size/importance of something, express this in dry terms, best backed up by numbers and references, like „… this method is suitable for very large datasets (up to 10 GB, for example in [XY])…“ instead of „… this method is suitable for gigantic datasets …“
- Avoid global judgements of the kind "the most famous...", "the most widely used..." – because you don't actually know this! Statements of this sort (expressed in proper terms) are only admissible for authors that have a long experience and a good standing in a field. In a student's report, they sound presumptuous.

Spaces before braces

I really don’t know why, but an estimated half of technical writing beginners do not insert spaces before brackets. I often see something like „…as shown in earlier work[2]…“ or „… seems a good argument(but …)“ . Why, oh why, do so many students not insert spaces correctly: „…as shown in earlier work [2]…“ or „… seems a good argument (but …“

Equations

Equations (regardless whether they are inline and non-numbered or set apart and numbered) are part of the text flow and must be surrounded by the appropriate interpunctuation. For instance, write

"… as can be seen in by the inequality

$$A < B, \qquad (12)$$

which demonstrates that A is always smaller than B."

Note the “,” after the B!

Every variable or constant in an equation must be explained in words somewhere in the text before, or right after the equation. A typical phrasing goes like this:

"bla ...

$$f(\mathbf{x}) = 3, \qquad (13)$$

where f is any sigmoid-shaped function and $\mathbf{x}$ is the normalized network state. "

Acronyms

The full wording of any acronym that you use must be given at the first appearance of the acronym.

Math symbols

Mathematical symbols appearing in the main text must be rendered in the same mathematical font as when they appear in set-apart formulas. In Latex, write N in your latex code if you want to refer to the number N in the main text! When you want to use bold font math symbols (for instance, for matrices), write $\mathbf{X}$ etc.

References list

- Use of bibtex saves you from a lot of references formatting errors.
- Spend much care on the reference list. It is the quality fingerprint of a technical article and is one of the first things to be inspected by professional readers and reviewers. When it is not perfectly formatted, or when it contains low-quality or irrelevant references, the reviewer will immediately have doubts about your qualification.
- Use the appropriate bibitem category (article, proceedings, techreport, unpublished…)
- Do not simply use the bibtex records that Google Scholar offers. They are often incomplete, incongruously spelled, and frequently contain errors. Grab the bibliographic information from the original version of the paper, and/or open the website of the journal / conference / institution where it actually appeared, to fix the details. This is some work, but is a service to scientific hygiene.

- Use a homogeneous formatting for all your references (e.g. use full names for all items; or use initial + family name for all items, don't mix the two)
- Capitalize names of Journals and Conferences (e.g. „Neural Computation", not „Neural computation"
- When the article title contains uppercase elements - for instance acronyms - make sure that they appear in uppercase; bibtex turns everything in titles to lowercase which can be prevented by putting those parts into {}
- Study my publications webpage or the reference list in a paper published in a good journal to see how a properly formatted reference list looks and feels.

Graphics and figures

- When creating graphics, make sure you get high-quality displays in the report. Never use low-resolution jpg or low-resolution pixel graphics. Thin lines should come out crystal-clear. The best thing to do is to use a graphics tool that outputs eps or pdf formats.
- Figures must be referenced from within the main text. For every figure there must be a sniplet in the text like „… see Figure 3 …" or „… this leads to significant improvement (Figure 3)".
- *All* symbols in a figure must be explained, either directly in the figure, or in the caption, or in the main text.
- Avoid pointing to figures by „as shown in the following figure" etc. Instead, always use numbering: "Figure 2.1 shows ...". Besides tradition, a good reason for this strategy is that journal editors are prone to shift figures around on a page or across pages, destroying references by relative location.
- Axes annotations must be legible-sized fonts (not microscopic!)

Numbered items

like figures, sections, tables, should be written in Uppercase when referred to. Example: „… as can be seen in Table 3, …" (not: …in table 3).

Sectioning

When you use subsections (latex: \section{}, \subsection{}, \subsubsection{}), it is standard to insert a brief overview text or motivation after \section{}, explaining what is going to happen in the section; then start the technical contents with \subsection.

Program code

NEVER reprint original program code in scientific writing. Instead, provide *pseudocode* descriptions of the algorithmic bone of your procedures (there are a number of latex packages for nice pseudocode environments), or specify what your program does in abstract mathematical formalism. -- In professional academic writing it has become good practice to supply easily runnable code online on the author's homepage, so that readers can re-run the procedures from the article. Some journals also offer "supplementary online materials" associated with a published paper, where one can also deposit runnable code.

Using high-end toolboxes (TensorFlow and friends)

You can use such toolboxes – would be madness to implement a complex neural network architecture and its learning algorithms from scratch. But: you need to understand – to a reasonable degree – what the functions and architectures that you use actually do. It is not enough to just mention the tool-specific names of the modules and algorithms that you use – I want to see from your write-up that you had insight in what those powerful things do, and what control hyperparameters you set how and why (and not just drop the tool-specific names of those parameters but also explain what they mean and do).

Appendix B: Grading template for final report

Evaluation form for semester project report, Neural Networks (AI)

Name of students:
Name of grader:
Date:

Fill in violet fields with grades on scale 0-10. If not applicable, put weight to zero. For bonus, fill in green field (as For Latex use, enter a 1 in the red field when Latex was used, else 0.

	grade	weight %	total	criteria
Presentation (40%)				
Figures, tables, pseudocode		5	0	good quality figs? Captions? Symbols explained in caption? Figs etc. referenced in text? Sources of imported graphics given? Legible in B/W printout? Figs informative (not redundant)? Pseudocode transparent?
Technical writing		35	0	clear formulations? Concise formulation? Mastership of technical formulations / math formulations? Good text flow? Appropriate language (not sloppy in technical sections, not too dry in motivation sections?) Clear flow of argumentation? Contents well structured? Correct English, typos? All used external sources (literature, code) referenced? Reference list uniformly and correctly formatted? are experiments/implementations clearly, completely, succintly described? Can experiments be checked/reproduced? results clearly, completely, succintly documented? Technical descriptions done in clear English and/or formulas and/or pseudocode, without relying on toolbox terminology?
Latex	1.00		0	was Latex used? If yes leave the 1.00, else write 0 in the red field
Technical quality (60%)				
Penetration of subject		20	0	Challenges of data / task well recognized? Clear perception of goals for the prooject? Good task statement? Challenges inherent in dataset perceived and adequately addressed? Appropriate choice of model type? choice of preprocessing, feature extraction, learning algorithm motivated? Motivational examples?
Technical work		30	0	Results connected back to starting question? Statistical basics done properly where applicable (error bars!)? Maths correct? Reasonable choice of model type? Appropriate pre-processing of raw data? Implementations / experiments reasonably thought out? Cross-validation / early stopping appropriately used? Planned professionally (modular, efficient, transparent, documented)? Realistic assessment of qualities / deficiencies of results?
Exhaustiveness		10	0	Is the amount of work done adequate for an extended "lab exercise"? (if ok, full score here - for extra effort give some bonus below)
Bonus		100	0	extraordinary achievements, e.g. much extra work, very independent work, very difficult topic, interdisciplinary connections, original thinking. Max 1.5 bonus grade points possible; typically fractions of 1.0
Report Grade			***0.00***	