Course reports/master's thesis: advice and common errors

by Thomas Moerland
LIACS, Leiden University
2021

Contents

This document will shortly summarize some common errors in scientific reports for courses. Before we start, you need to decide on the software to write your thesis or report in. I strongly recommend (to learn to) **use Latex**. It makes it much easier to write mathematical equations, and it makes your reports look more neat, with a clean lay-out by definition. You can find many tutorials online, and your learning curve will be very steep.

1 Cheat Sheet: Most Common Errors

A short list of very common errors in report writing (see remainder of document as well):

- Explain your full method. Do not stick with textual statements only, provide equations and/or algorithm boxes.
- Keep symbols/notation of equations consistent throughout the document, i.e., always use the same symbol for the same quantity. Explain every new symbol. Do not copy in equations from other sources, write them yourself (for consistency and lay-out).
- For hyperparameter tuning: always tune the most important parameters first. In learning experiments, always start with extensive tuning of the learning rate. In RL experiments, also extensively tune the exploration hyperparameter. When these are off, your methods will simply not work, no matter how much you tune the other parameters.
- Do not just describe your results, also interpret them. State what you learn from a graph, and provide possible explanations for your observations.
- Average results over repetitions. Smooth curves. Include confidence areas around curves.
- Make graph/table captions self contained. They should be understandable without the main text.
- Be precise in your phrasing (or it will seem as if you did not really understand).

2 Hyperparameter optimization

Hyperparameter optimization is a large part of learning experiments. Unfortunately, it is more art than science, and there it can be really struggling in the beginning. We will discuss two things: 1) intuition about which parameter are most important, and 2) ways in which you may set up your hyperparameter experiments.

Most important hyperparameters The crucial rule with hyperparameter optimization is: tune the most important parameter first!

- In learning experiments, the learning rate is practically always the most important parameter. **Always start with extensive tuning of the learning rate**.
- In reinforcement learning experiments, the second most crucial parameter is the amount of exploration. This may be the ϵ in ϵ-greedy exploration, the temperature in a Boltzmann policy, the standard deviation of the Gaussian noise added to a policy network, etc.
- Crucially, when the key hyperparameters are off, your method will simply not work. You can experiment forever with the number of layers and types of nonlinearities in your network, but when the learning rate is completely wrong, they all won't do anything.

Identify which hyperparameters you will vary

- You usually can't afford to optimize over all your hyperparameters. Therefore, first identify which parameters are crucial for performance (see above), and which parameters are of your specific interest.
- For the other hyperparameters, you wish to start from 'reasonable values'. This can be hard if you are new to the field. See the section on general intuition below.

Optimize your hyperparameters

- A main approach to hyperparameter optimization is *grid search*. Partition each hyperparameter of interest into a grid of values, for example learning rates $\{0.01, 0.001, 0.0001, 0.00001\}$ and exploration ϵ $\{0.2, 0.1, 0.05\}$. Then, run each combination of these settings, for example for 5 repetitions. This gives in total $4 \cdot 3 \cdot 5 = 60$ runs. Visualize the learning performance for each setting to find the optimum.
- When a single run is expensive, the above procedure may take very long, especially when you have more than 2 important hyperparameters (the number of runs scales exponentially). As an alternative, you may also try to optimize one hyperparameter at a time (keeping the others fixed). This will give suboptimal results (the parameters likely interact), but can be more manageable.
- Finally, there is much research into *automatic hyperparameter optimization*. You do not need to do a full study into these methods, but they typically try to come up with a smart sampling scheme in hyperparameter space based on previous runs, and/or cut training runs early

when they are not promising[1]. However, there are many public libraries that do automatic hyperparameter optimization for you, which you can try. Be sure to log performance of runs to show in you report (or appendix).

General intuition about some common hyperparameters

- We list some common hyperparameters and their considerations in the table below. One other good solution is to check other online codebases of similar experiments, and look what they use for particular values. For example, the discount parameter in reinforcement learning can often be set to something like 0.99, 0.999, or even 1.0, and you do not need to change it much.

Table 1: Intuition on hyperparameters. Note: this list is by no means exclusive, and gives no guarantees. Performance is always problem dependent.

- Learning rate	Try a good grid, like $\{0.01, 0.001, 0.0001, 0.00001\}$. Always vary.
- Exploration constant	Depends on the exploration method. Ensure enough exploration initially during training. Potentially decay, but usually leave some finite exploration towards the end. Always vary.
- Discount	No need to vary much. Put at 0.99 or 1.0 (we often care about long term reward). Only make smaller to increase stability of training.
- Neural architecture	Neural network structure: The required size of neural network often depends on the size of the input. For a flat input vector of length 4, you will only need one or two hidden layers. Do not make them too small, e.g., 32, 64, 128 or 256 units. In general, depth should go with some width and vice versa, i.e., you do not want a 10 hidden layer neural network with only 4 units per layer, or a single hidden layer neural network with 10.000 hidden nodes. Convolutions are only useful when the input has spatial structure, e.g., a 2D or 3D structure for visual input. For visual input you want to increase the number of layers, first a few convolutional layers, and afterwards a few fully connected layers, which will give your network much more depth. In general: do not make the network too small (undercapacity is often a bigger problem than overfitting).
- Batch size	Not too small (unstable gradients) nor too large (slow learning). For example fix at 32 or 64. Not the first parameter to vary.
- Depth of value target computation (1-step, n-step, Monte Carlo target)	One-step (high bias, low variance) versus Monte Carlo roll-out (low bias, high variance). Trade-off, can try intermediate value n-step target) or reweighted average (eligibility trace/generalized advantage estimation). Often fixed per algorithm, but can have big influence on performance.
- Algorithmic components	Sometimes, entire algorithmic components maybe switched on or off (i.e., a 0/1 hyperparameter). For example, with/without target network or with/without experience replay.

[1]The early cutting approach is more suited to supervised learning, where learning curve are more smooth. In RL, performance increases more jumpy, and is harder to predict

3 Statistical practice on results

1. Whenever your runs have a stochastic component: average your results over repetitions! We want to see the average performance of an algorithm, not just a single lucky or unlucky run. Never optimize the seed, but run multiple experiments. This is **really** important.

2. Average your curves over the repetitions. Then also include the standard deviation around the curve, to show the variation in the experiment. For the confidence interval of the mean (to compare lines), you can divide the SD by $\sqrt{n}$ (when you have n independent repetitions).

3. Smooth your learning curves! Especially in reinforcement learning experiments, there is often a lot of noise in your run. Do not plot a wobbly line that goes up and down. Averaging over repetitions will already help, but you may after that also use a sliding window to smooth the curve.

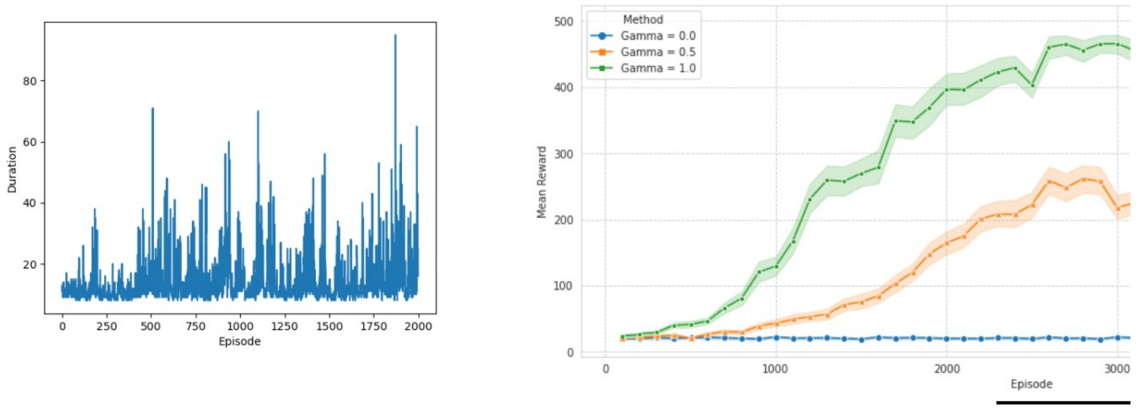

Figure 1: Comparison of good and bad result graph. Left: BAD, single parameter setting, noisy/wobbly line, single repetition, no smoothing. Right: GOOD, compares settings, smooth lines, averaged over repetitions, confidence intervals included.

4 Equations

Equations can be very challenging/intimidating for some students. There are 5 main rules you need to follow:

1. *Keep notation consistent throughout the text*! Do not use r for reward in the first equation, and then suddenly R for reward two pages later.

2. *Use symbols in equations, not full text*:

   - BAD:
     $$\text{parameter} \leftarrow \text{parameter} - \text{lr x } \nabla J(\theta)$$
   - GOOD:
     $$\theta \leftarrow \theta - \alpha \cdot \nabla J(\theta).$$

3. *Always introduce/define every new symbol you use.* (Often forgotten, and the reader will not know what you mean with a symbol).

   - BAD: Finally, the update rule is given by
     $$\theta \leftarrow \theta - \alpha \cdot \nabla J(\theta).$$
   - GOOD: Finally, the update rule is given by
     $$\theta \leftarrow \theta - \alpha \cdot \nabla J(\theta),$$
     where $\alpha \in \mathbb{R}^+$ denotes the learning rate (assuming θ and $J(\theta)$ were already introduced before).

4. *Write equations yourself (preferably in Latex).* Don't start taking screenshots from other formulas, since: a) the lay-out will become a mess, and b) your notation will almost by definition become inconsistent when you start copying from different sources.

5. *Probably most important: don't be afraid of formulas*! You need them to properly explain what is happening, **you can never correctly/fully explain operations on numbers with words only**. The more you practice, the better you get, and vice versa (the more you avoid it, the harder it becomes).

5 Figures and tables

Captioning

- **Figure/table captions should be self-contained** (readable without the main text). Describe what the figure shows, on which domain. Describe what you can interpret/learn from the graph. Indicate how the results were produced. Explain abbreviations from the graph/table.
  - WRONG: *Figure 1.*
  - WRONG: *Figure 1: learning curve on CartPole.*
  - CORRECT: *Figure 1: Effect of discount factor gamma on learning performance in CartPole. Graphs show learning curves for three values of gamma. Clearly, a discount of 0.0 does not let the agent learn at all (likely since it completely ignores long-term rewards). Optimal performance is achieved for the highest discount rate of 1.0. Results averaged over 10 repetitions, shaded areas indicate 1 standard deviation confidence intervals of the mean. MSE = mean squared error.*

Lay-out

- Provide axis labels, e.g., 'episodes' or 'steps' on the x-axis of a learning curve. Don't leave people guessing what you actually plotted.
- With multiple curves, include a legend. Clearly show what each line represents.
- Make the text of legend and labels large enough/readable (standard Python plotting plots them really small).
- Export your figures in high enough quality and size.

6 Software frameworks

Developing a machine learning application has a common lifecycle. Many parts of this cycle have been automated in software packages, and these can strongly reduce the time you have to spend on this yourself.

- Parallelization: Ray (`ray.io`) makes parallelization in Python very easy.
- Logging experiments & visualization: Weights and Biases (`wandb.ai`) is a useful package to set-up your experiments, make hyperparameter sweeps, and log all your output to an online tool for visualization.
- Hyperparameter optimization: An alternative for hyperparameter optimization is Ray Tune (`docs.ray.io/en/master/tune/index.html`).
- Environments: There are third-party packages that integrate with OpenAI Gym. Examples can be found at `https://github.com/openai/gym/blob/master/docs/environments.md#third-party-environments`.

7 Thesis/report structure

Scientific documents have a general structure, as shown in Table 2. We will discuss each section and its requirements separately. You have the freedom to vary within this structure when applicable, but most elements should be present somewhere. After reading this document, check your favorite scientific paper and see if your recognize the structure (or alterations from it).

7.1 Abstract

Your abstract should be short (e.g. 10-15 sentences), and summarize the entire report. Include the follow topics (spend 1-3 sentences on each of them):

- the overall topic/setting
- the problem you address
- your proposed solution
- the results you find
- what this adds/why this is relevant to the reader

7.2 Introduction

The introduction should explain to the reader whether or not this is a paper/report/thesis to further read. Discuss the following items:

- Topic/setting: what type of problem are you working on, and why is this a relevant problem?
- Problem identification: what is the state-of-the-art approach for this problem (cite the most relevant related literature), and what is still lacking in the state of the art?
- Contribution: what do you propose to solve this deficit?
- Results: What results do you find? What does your paper add to the literature, or teach the reader?
- Structure of document (optional): describe the overall structure of your document

For a paper: spend less than a page on all of them, e.g., one paragraph for each item (although you may also take items 1-2 or 3-4 together in one paragraph. For a thesis: be more extensive, maybe 5 pages in total, but these are not strict guidelines.

7.3 Background/preliminaries

This section is optional. It should establish common ground for the rest of your report to build upon, and thereby keep the report self-contained (notation wise). You may describe:

- The problem setting/definition. You may want introduce the Markov Decision Process formulation, or the principles of Bayesian learning, or deep learning, etc.

Table 2: General structure of thesis or research paper/report. Left: structure with single methodology and report sections. Right: structure for larger reports and/or clearly separate research questions, where the methodology and results section are broken up per topic. Choose the structure which you believe gives the highest clarity to the reader (note: methods and results should not get too far apart, in those cases it might be useful to choose the right structure.

• Abstract • Introduction • Background/preliminaries • Related Work (or before/in the Discussion) • Methodology Algorithmic description Experimental design • Results • Discussion • Conclusion • References • Appendix (possibly)	• Abstract • Introduction • Background/preliminaries • Related Work (or before/in the Discussion) • Research topic/question 1: Methodology Algorithmic description Experimental design Results/experiments • Research topic/question 2: Methodology Algorithmic description Experimental design Results • Discussion • Conclusion • References • Appendix (possibly)

- Common notation. Very importantly, along with the above you should also introduce your basic notations and symbols you will use throughout. Never use a symbol which you have not first defined!
- Existing algorithms which you will use/extend in this work.

7.4 Related work

Related work is an important aspect of a scientific report. In a thesis, it should be a separate section. In smaller reports, you may also push it into the discussion. Related work can also be a separate section before the discussion (you may move this one around).

Finding related work

- Search Google Scholar with relevant key words, and identify other papers that have taken a similar approach to yours.
- When you find a relevant paper, the Related Work section of that specific paper can often help you identify much more previous related work. This is the most effective way to quickly identify literature on a topic!
- You can also go in the forward time direction. Start from a relevant related paper, and look in Google Scholar which recent papers have cited this work. These may also be relevant for your work.

Writing related work

- Do not fully read each potential related paper. Scan the abstract and introduction, glance at the graphs, check the discussion and conclusion. Try to get a feeling for what happens in the paper, whether it is relevant. Sometimes, you may need/want to read it line by line, sometimes this is not necessary.
- Write the related work section. Make it structured. Often, your research has different connections, e.g., there are different groups of approaches to your problem, or your research combines ideas from multiple research fields. For each group, write a separate paragraph in which you mention related work, and how it compares to your work.

7.5 Methods

You do not need to call this section 'Methodology', but may also give it the name of your proposed solution. You need to describe two things: 1) your algorithm/proposed method, and 2) your experimental set-up. Sometimes, the latter can also be a separate section.

Algorithmic description

- Clearly describe what you propose. Use equations and/or algorithm boxes. For additional advice on equations see Sec. **??**.

- Visual illustration can really help here. If possible, make a picture that shows your algorithm lay-out.
- Make sure a reader could full explain your approach from the text and illustrations.

Experimental setup

- Describe the performance measures you choose (unless it is very obvious/common). Given your research goal/question, what are relevant things to vary, and which thing will you measure.
- Mention all hyperparameters, ideally in a table (e.g. in the appendix). Once you start writing them out, you realize that you have many more than you thought. Describe how you optimized them.
- Attach your source code, or provide a link to a public repository!
- Again, make sure a reader could reproduce your experiments from the text and illustrations.

7.6 Results

- Show your results in graphs and tables. Carefully think about ways to best display your results. What visualizations answer your research question. What visualization gives you good insight into what happens in the algorithm, or which approach works best?
- In the main text: refer to the figures/tables, describe what you see. Keep the numbering ordered.
- Very important: not only describe, but also *interpret*. Indicate what you learn (and what you think the reader could learn) from this figure/table. Come up with possible explanations for your observations.
- Show as much results as possible. Move less relevant results to the appendix, to keep the main text readable and focused on the main story.

For specific advice on figures, tables and statistically correct experiments, see the later sections of this document.

7.7 Discussion

The discussion should reflect on your work. It may contain the following parts:

- A high-level summary of your main findings: what main conclusions do you draw. You have already done this in smaller form in your results section, but you may zoom out now, and draw the bigger picture.
- The main weaknesses of your approach. To what extend are these limiting? Why did you make these design choices. What could have been done differently?
- Related work (when you have no separate section for it).
- Future work: what are the most promising directions for extending your ideas. Maybe connect to some existing related work in literature.

7.8 Conclusion

- Can also be the last paragraph of your discussion. Shortly state what you proposed, found, and why this is relevant.

7.9 References

- References and appendix are not numbered!
- Do not make references a bullet point list. Instead, give them a number, and cite in the text with [4], or sort them on last name, and cite with (author, year) in the text.
- Use a reference manager, like bibtex, to give you references a good lay-out. This will automatically generate your references and reference style, and save you an extreme amount of hassle.

7.10 Appendix

You can always include as many appendices as you want. You use these for elements of your text that take much space, and would thereby intervene in the flow of the report, but are nevertheless interesting for readers that want to dig deeper. The appendix mostly includes:

- Full experimental details: like a full table of all hyperparameters.
- Additional results: for example extra visualizations for hyperparameter runs etc. This quickly takes up much space.
- Mathematical details: proofs or expansions that would take too much space in the main text.