1RT705/1RT003: Project assignment
Advanced Probabilistic Machine Learning 2023

Abstract

This document contains the instructions for the project assignment for the course Advanced Probabilistic Machine Learning, 1RT705/1RT003. In this project, you will implement a Bayesian method, based on the TrueSkill[1] ranking system, to estimate the skill of players involved in a competition. You will evaluate the proposed method on real-world data and compare to alternative approaches. You will document your project by writing a report, which will be reviewed anonymously by your peers.

Deadline: See information on Studium for deadline.

Requirements: The project is to be done in groups of 3-4 students. All group members need to take part in the project.

Instructions for how to write your report: The report should be coherent including: an introduction, motivated solutions and discussion. *All* questions in these instructions should be addressed in the report and the requested plots included. You can assume that whoever will read your report has read these instructions, nonetheless your text should be coherent and well-connected and not only a plain list of result. You should also include an *Introduction* section motivating the problem and a *Discussion* section discussing the obtained results. You should deliver *one* PDF file of a maximum of *eight* (8) pages. The file should use the *NeurIPS 2020* template[2] (`\usepackage{neurips_2020}`); you should not change margins, font sizes, or formatting. Do not change the package options! All figures have to be legible, complete with labels, caption, and legend (where applicable). Bibliography, simulation code, and additional material can be added as an appendix beyond the eighth page, but should not be required in the grading: *the first eight pages should contain all important derivations and results, as well as design choices made in the implementation.* Use plots only when necessary to convey a point across: do not plot something just because you can. Do not repeat theory from the book when it is not strictly needed: use references and citations for that.

Coding: You are free to use any programming language; however, `python 3` is the officially supported language in the course. All simulation code should be submitted in a separate ZIP file, including *one* (1) runnable script or Jupyter notebook (called `runme`) to generate all the figures and results in the submission. All datasets required should be submitted together with the code. Also add a file `requirements.txt` stating which packages (with version number) required to run the code and a `readme.txt` with instructions about how to run the code. This is to ensure that the code is reproducible.

First submission: Submit one PDF with the report and one zip with the code as *one* submission in Studium. Do *not* put the PDF inside the zip, and do *not* submit the two files as two sequential submissions.

Peer review: Each student will receive the report of another group, which you have to review. This means that the peer review is done individually and each group will receive multiple reviews.

As a peer reviewer you are mainly expected to comment on the technical quality of the report, if relevant figures have been included, and if the language of the report is satisfactory. The review process

[1] Trademark Microsoft research, used with permission for non-commercial projects `https://www.microsoft.com/en-us/research/project/trueskill-ranking-system` (accessed: 10 Aug 2023)

[2] `https://nips.cc/Conferences/2020/PaperInformation/StyleFiles` (accessed: 10 Aug 2023)

is "double-blind", meaning that both the authors of the report and the reviewers are anonymous. The review is done by filling out scores in the rubric of the mini-project in Studium and by adding text comments in that rubric. Please follow the instructions in Studium for how to fill in and submit your review. Once the review deadline has passed, each group will get the reviews on their report from the other students.

Every student has to pass the peer review individually. In order to pass, you have to fill out the rubic and provide constructive comments. You are not limited by the number of comments you can give. A guideline is to write 1/4 to 1/2 page of comments in total in order to pass. Note: Even if you think that a point is well done, then comment on it and explain why you think so.

Second submission: After the peer review deadline has passed, all groups will be requested to update their reports based on the received peer review comments or other insights gathered since the first submission. The second submission shall *not* be anonymous. The second submission shall, equivalent to the first submission, consist of two files – one pdf with the report and on zip file with the code. The second submission will be reviewed and graded (pass/revise/fail) by the teaching assistants. In addition, you should also submit a simple text file (.txt) which clearly states the contributions of each group member: who contributed to each part and to which degree. The contribution statement shall also be submitted in Studium, but via different submission page. Follow the instructions in Studium.

Grading: Both peer reviewers and teachers will evaluate the report based on the rubric (sv bedömningsmatris) available in Studium. The teachers will also, based on this evaluation, give a grade (pass/revise/fail) of the report. If the second submission is graded revise, a third submission can be submitted, which will be graded (pass/fail). A report with grade fail can only be resubmitted and graded in conjunction with a reexam.

Introduction

In this project, you will work through the whole process of solving a real-world problem using probabilistic machine learning. You are tasked to estimate the skill of players involved in a competition based on the results of matches between pairs of players. You will first define a probabilistic model, where all the quantities are represented as random variables, based on the *Trueskill* Bayesian ranking system developed by Microsoft research for raking on-line matches. The model assigns a *skill* to each player, in the form of a Gaussian random variable. When two players meet in a match, the *outcome* of that match is a Gaussian random variable with a mean equal to the difference between the two players' skills. The *result* of the match is 1 (indicating the victory of Player 1) if the outcome is greater than zero, -1 if it is less than zero (indicating the victory of Player 2).

You will use Bayesian inference to find the posterior distribution of the players' skills given observations of the results of matches. Because the posterior distribution is intractable, you will use two different approximation methods based on graphical models.

On the course website, you will find the `SerieA.csv` dataset containing the results of the Italian 2018/2019 *Serie A* elite football (soccer) division; however, the Trueskill model can be used to solve a variety of skill and ranking problems. Later in the project, you will be asked to use a dataset of games/competitions of your choosing. We suggest that you start thinking about it early on!

Assignments

The following questions need to be addressed in the report. We suggest that you solve the tasks sequentially, as later questions build upon results of previous questions.

Q.1 Modeling (Lecture 1)

Formulate the *Trueskill* Bayesian model for one match between two players. The model consists of four random variables, two Gaussian random variables s_1 and s_2 for the skills of the players, one Gaussian random variable t, with mean $s_1 - s_2$, for the outcome of the game, and one discrete random variable $y = \text{sign}(t)$ for the result of the game (there is no possibility of a *draw* between the players). Note that there are 5 *hyperparameters* in the model whose values you have to set. You

can read about TrueSkill on the Microsoft research website[3], on the original publication[4], or on Jeff Moser's website[5]. Hint: remember that a "model" in Bayesian parlance is a joint distribution of all the random variables.

Q.2 Bayesian Network (Lecture 4)

Draw the *Bayesian network* of the model from Q.1. Using the network, verify that both s_1 and s_2 are conditionally independent of t given y, i.e., that

$$s_1 \perp y|t, \qquad \text{and} \qquad s_2 \perp y|t.$$

Q.3 Computing with the model (Lecture 2 and 4)

Using results on Gaussian random variables, compute

- $p(s_1, s_2|t, y)$: the full conditional distribution of the skills. *Hint:* use the conditional independence statements in Q.2 to eliminate y, and the results on Gaussian random variables from Lecture 2[6] to get to the solution.
- $p(t|s_1, s_2, y)$: the full conditional distribution of the outcome. *Hint:* using Bayes' theorem, $p(t|y, s_1, s_2) \propto p(y|t)p(t|s_1, s_2)$; the second factor is Gaussian while the first one is nonzero only when y and t have the same sign; the result should be a *Truncated Gaussian*.
- $p(y = 1)$ the marginal probability that Player 1 wins the game. *Hint:* $p(y = 1) = p(t > 0)$ where t is the Gaussian random variable obtained by marginalizing out s_1 and s_2 from $p(t, s_1, s_2)$.

Hint: There is a quiz available in Studium which you can use to verify your distributions.

Q.4 A first Gibbs sampler (Lecture 5)

In this question, you will implement a method based on Gibbs sampling to compute the posterior distribution of the skills s_1 and s_2 given the result of one match y.

Using the results of Q.3, implement a Gibbs sampler that targets the posterior distribution $p(s_1, s_2|y)$ of the skills given the result of one game between two players. Consider the same prior distributions for the two players ($p(s_1)$ and $p(s_2)$ have the same hyperparameters). The values for these hyperparameter is a design you choose. *Hint:* the function `scipy.stats.truncnorm` may be useful.

- Only the *stationary distribution* of the Gibbs sampler represents the posterior distribution of the skills. Initial samples will depend on the initial condition of the chain and may be far away from the stationary distribution and should be discarded (the so called *burn-in*).

  Plot the samples of the posterior distributions generated by the Gibbs sampler when $y = 1$ (Player 1 wins). What seems to be a reasonable value for the burn-in? Comment on your choice of burn-in. Then, re-run the experiment. Was your burn-in a good choice also for the second run? Comment.
- To recover the *Trueskill* representation of skills as Gaussian random variables, we need to transform the samples drawn from the Gibbs sampler into Gaussian distributions. Implement a function that uses the mean and covariance of the samples drawn by the Gibbs sampler to find a Gaussian approximation of the posterior distribution of the skills.
- When deciding how many samples to use (after burn-in), there is a tradeoff between accuracy of the estimate and computational time. Plot the histogram of the samples generated (after burn-in) together with the fitted Gaussian posterior for at least four (4) different numbers of

[3] `https://www.microsoft.com/en-us/research/project/trueskill-ranking-system` (accessed: 10 Aug 2023)

[4] Herbrich, Minka, Graepel, "TrueSkill(TM): A Bayesian Skill Rating System," *Advances in Neural Information Processing Systems*, January 2007, MIT Press

[5] `http://www.moserware.com/2010/03/computing-your-skill.html` (accessed 10 Aug 2023)

[6] These results are also summarized in `https://uppsala.instructure.com/courses/85923/files/5392916?wrap=1` (accessed 10 Aug 2023)

samples, and report the time required to draw the samples. What is a reasonable number of samples? Comment on your choice.

- Compare the prior $p(s_1)$ with the Gaussian approximation of the posterior $p(s_1|y = 1)$; similarly, compare $p(s_2)$ with $p(s_2|y = 1)$. What has happened? Comment.

Q.5 Assumed Density Filtering (Lecture 5)

The Gibbs sampler from Q.4 processes the result of one match to give a posterior distribution of the skills given the match. We can use this posterior distribution as a prior for the next match in what is commonly known as *assumed density filtering* (ADF). In this way, we can process a stream of different matches between the players, each time using the posterior distribution of the previous match as the prior for the current match.

- Use ADF with Gibbs sampling to process the matches in the `SerieA` dataset and estimate the skill of all the teams in the dataset (each team is one Player with an associated skill s_i). Note that there are draws in the dataset! For now, skip these matches and suppose that they leave the skill unchanged for both players. For now, also skip the information of goals scored. Only consider how won or lost the game.
  What is the final ranking? Present the results in a suitable way. How can you interpret the variance of the final skills?
- Change the order of the matches in the `SerieA` dataset at random and re-run ADF. Does the result change? Why?

Q.6 Using the model for predictions (Lecture 5)

Being a Bayesian model, Trueskill can be used to answer probabilistic questions such as *"what is the probability that Player 1 wins against Player 2?"*. In particular, we can make predictions based on these probabilities. Propose a prediction function that returns $+1$ if the Player 1 will win and -1 if the Player 2 will win (note that the result should be *deterministic* and that you are only allowed to use the information of previous matches in this prediction!).

Using the `SerieA` dataset, compute the *one-step-ahead* predictions of the results based on the model and compare with the actual results for all matches; for each match

1. compute the prediction and compare to the result in the dataset,
2. update the model using the new match,
3. iterate over the whole dataset.

Report the *prediction rate*

$$r = \frac{\text{number of correct guesses}}{\text{number of total guesses}}$$

of your method over the whole dataset. Is it better than random guessing?

Q.7 Factor graph (Lecture 6-7)

Draw the *Factor graph* of the model and identify the messages needed to compute $p(t|y)$. Write explicit forms for these messages.

Hints:

- the final graph should have 4 variable nodes and 4 factor nodes.
- $p(t|y)$ is a truncated Gaussian
- Exercise 7.1a may be useful!

Q.8 A message-passing algorithm (Lecture 6-7)

Use *moment-matching* to approximate $p(t|y)$ with a Gaussian. Implement the message-passing algorithm to compute the posterior distribution for each of the two skills given the result of one game between two players.

(You may want to check Exercise 7.2 for inspiration). Plot the posteriors computed with message passing. In the same plot, show also the histogram and the Gaussian approximation from Gibbs sampling. *Hint:* The posteriors using moment-matching and Gibbs sampling should look very similar.

Q.9 Your own data

Test the Trueskill methods you have developed on a dataset of your own choosing. Some suggestions are other team sports (e.g., hockey, basketball, rugby), two-player sports (e.g. tennis), tabletop games, or computer games. Include details about the source of the data and any pre-processing you have done. Be creative!

Q.10 Open-ended project extension

In implementing Trueskill, you may have noticed that both the model and the inference algorithms have limitations. In this part of the project, you are free to define your own extensions of the model and of the method in some way you find interesting. For example, the model may be tuned more to the specific competition considered:

- Model draws?
- Does one of the players have an advantage *a-priori* (e.g., white in chess)?
- Should skill change over time?

Alternatively, the algorithm may be improved upon in various ways:

- Process more matches together to improve performance?
- Tune the hyperparameters differently/better?
- Use the scores of the matches to improve the estimates?
- Use data from other datasets to improve the predictions?
- Implement a "smarter" prediction function?

Implement at least one extension on the model and test it on the datasets. Do the results change? Can you get better prediction rates?

The answer is open-ended and only one, well motivated, extension is strictly required; you are free to implement and extend as much as you please!

Good Luck