MO435 — Probabilistic Machine Learning
INSTITUTO DE COMPUTAÇÃO — UNICAMP
1st Semester 2020

Project 1

Prof. Adín Ramírez Rivera
`adin@ic.unicamp.br`

1 Description

Generative and discriminative models are two main paradigms to tackle classification and regression problems. The main difference, is the usage of data to build a model of it or to learn a boundary for classification. In this project, you will explore the differences between these coupled models under a probabilistic setting.

In our setup, let's assume we have a set of data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ where each pair corresponds to the i-th information vector $\mathbf{x}_i$ and its corresponding information value y_i (either a label or a value to regress).

2 Probabilistic Generative Model

Your task is, given a data $\mathbf{x}$ at testing time, to find its corresponding class c. For this, you need to compute

$$p(y = c \mid \mathbf{x}) = p(\mathbf{x} \mid y = c)\, p(y = c), \tag{1}$$

for every class c, and classify find its class $\hat{c}$ through

$$\hat{c} = \arg\max_c p(y = c \mid \mathbf{x}). \tag{2}$$

Assume that the class conditional densities $p(\mathbf{x} \mid y = c)$ are Gaussian with all the classes sharing the same covariance matrix.

Find the model to be optimized given the restrictions above. Hint: review chapter 4 in Murphy's [1] or Bishop's [2] book.

2.1 Experiments

For your experiments, you need to create a dataset for training and testing. You can use `sklearn.datasets.make_classification` or `sklearn.datasets.make_blobs` to create your data sets. You **must** create two sets of 50 data points of 2 dimensions per class for 3 classes. Split the data 50/50 per class to have a train and test partition. Setup a seed such that you can obtain the same data if the process is executed again. (This is needed to get the same data on the pipeline for execution, see § 6.)

Explain your process to fit your model to the data, and to predict the over the test partition. Plot your data and your predictions, as well as your model over the same data (you can put some contour plots per class). Report your accuracy over the test partition.

3 Discriminative Model

The logistic regression corresponds to the binary classification problem

$$p(y = c \mid \mathbf{x}, \mathbf{w}) = \frac{\exp(\sigma(\mathbf{w}_c^T x))}{\sum_{c'=1}^{C} \exp(\sigma(\mathbf{w}_{c'}^T x))}, \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function. Our likelihood is

$$p(y \mid \mathcal{D}, \mathbf{w}) = \prod_{i=1}^{N} p(y = c \mid \mathbf{x}_i, \mathbf{w}), \tag{4}$$

where $\mathscr{D}$ is our set of data for training. However, we want uncertainty estimation for our parameter $\mathbf{w}$, so we can assume a prior for it. For instance,

$$p(\mathbf{w} \mid s^2) = \mathcal{N}(\mathbf{w} \mid 0, s^2 I). \tag{5}$$

Our posterior, is then

$$p(w \mid \mathscr{D}, y, s^2) = \frac{p(y \mid \mathscr{D}, \mathbf{w})\, p(\mathbf{w} \mid s^2)}{\int p(y \mid \mathscr{D}, \mathbf{w})\, p(\mathbf{w} \mid s^2)\, \mathrm{d}\mathbf{w}} = \frac{p(y \mid \mathscr{D}, \mathbf{w})\, p(\mathbf{w} \mid s^2)}{p(y \mid \mathscr{D}, s^2)}. \tag{6}$$

So, our prediction given a sample $\hat{x}$ is

$$p(y = 1 \mid \hat{x}, \mathscr{D}, y, s^2) = \mathbb{E}_{p(\mathbf{w} \mid \mathscr{D}, y, s^2)}[p(y = 1 \mid \hat{x}, \mathbf{w})]. \tag{7}$$

However, the posterior of the expectation is not a conjugate prior, and the integral in the denominator is intractable.

To solve this issue, you must compute the MAP estimate of $\mathbf{w}$ and use a sample method to estimate its uncertainty.

To compute the MAP, you need to solve

$$\mathbf{w}_{\text{MAP}} = \arg\max_{\mathbf{w}} p(\mathbf{w} \mid \mathscr{D}, y, s^2), \tag{8}$$

$$= \arg\max_{\mathbf{w}} p(y \mid \mathscr{D}, \mathbf{w})\, p(\mathbf{w} \mid s^2). \tag{9}$$

Unlike other problems, $\mathbf{w}_{\text{MAP}}$ cannot be computed analytically. Hence, you need to use gradient descent to compute this value.

Now, you have an estimate for your parameter $\mathbf{w}$ but you don't have the uncertainty, since the posterior is intractable. To solve this issue, you will use Monte Carlo to solve the previous expectation (7). Instead of computing its closed form, we can approximate it with

$$p(y = 1 \mid \hat{x}, \mathscr{D}) \approx \frac{1}{S} \sum_{s=1}^{S} p(y = 1 \mid \hat{x}, \mathbf{w}_s), \tag{10}$$

where $\mathbf{w}_s \sim \mathcal{N}(\mathbf{w} \mid \mathbf{w}_{\text{MAP}}, s^2 I)$ are S samples.

3.1 Experiments

Using the same parameters as the previous section, create a dataset with 2 classes. Implement the gradient descent method, and experiment with different learning rates to find the MAP estimate $\mathbf{w}_{\text{MAP}}$. Once you have it, extract S samples from the normal defined by it, and find the prediction through Monte Carlo (sampling).

Explain what is the equation to be optimized for the gradient descent (9). Show your results for the data points in the test set, and plot the sample classifiers. You should discuss the results of learning with different learning rates, and using different amounts of samples.

4 Evaluation

Your grade will be defined by the following aspects:

1. Explanation of the derivation of the generative model to solve 20%
2. Implementation and experiments of the generative model 20%
3. Gradient descent implementation 20%
4. Implementation of Monte Carlo 20%
5. Discussion of the effects of learning rates 10%
6. Discussion of the effects of amount of sampling 10%

Each item corresponds to the questions and requirements defined in the previous sections. The overall report, will be evaluated on the aspects regarding writing and requested plots or figures in the text. **Your language usage won't be graded**, but your ability to present your results, ideas, and how they are supported will be. Each point will be evaluated according to the completeness and correctness of the requested items.

Moreover, **the evaluation includes the contribution of each team member to the solution through the commits history and level of engagement of each team member through the repository**. Hence, do not commit all the work using one single student account. Each member should commit (using the email from the university that contains their student ID[1]) and handle their work on the repository.

5 Submission

Your submission must be through your assigned group on Gitlab. You must first declare a group in this form `https://forms.gle/Dv8SQ595xAqThLBR9`, and a group will be assign to you. You must create a repository named `project-1` and commit all the code and report there.

Your submission must have the following subfolders:

- `input`: a directory containing the input assets (images, videos or other data) needed to execute your experiments. Your generated data should be placed here.
- `output`: a directory where your application should produce all the generated files (otherwise stated in the problem). 🚀 This folder and all its contents must be added to the `artifacts` path on the `.gitlab-ci.yml` setup.

  Name your outputs according to a convention that reflects the experimental setup.
- `src`: a directory containing all your source code. You only need to submit files that are not derived from other files or through compilation. In case some processing is needed, prefer to submit a script that does that instead of submitting the files.
- `Makefile`: a makefile (or equivalent) that executes your code through the docker image. You must provide the image for the project. It is strongly recommended that you use a vanilla version of your language of choice, for example for python use `python:latest`. The code **must** be executed through a standard call to `make` (or equivalent). 🚀 Moreover, note that your code **must** run on the Gitlab executor (pipeline) on the server itself. Hence, you need to use the `.gitlab-ci.yml` configuration. You can base your work on the demo that already provides an example of such functionality.
- `report`: a directory containing the source files that produce your report. This directory must be only on its branch called `report`.

  🚀 Your report **must** be written using LaTeX (and friends) and compiled within the pipeline of the repository in a stage named `report`. Consequently, **the PDF must not be committed**. You can use the images from `adnrv/texlive` to build your PDFs. **Only the PDF of the report** must be added to the `artifacts` path on the `.gitlab-ci.yml` setup, and not other intermediary files of your report (e.g., images, log files, auxiliary files).

  Your report must show all your work for the given project, including images (labeled appropriately, that is, following the convention given) and other outputs needed to explain and convey your work. When needed include explanations to the questions given in the project. **Your report should be constrained up to 4 pages in content (excluding images, tables, and references).**

🚀 The last commit that triggered the build must be before the deadline of the submission. Do not worry about the time executing the pipeline. However, **you must ensure that your code works as no attempt will be made to patch or run broken code**.

[1] Note that the repository information is extracted through scripts that use your student ID as an identifier for the committer. If you do not have the repository set correctly it won't find you. No effort will be made to find the missing information if you do not have your git set up correctly.

6 Notes

- All commits **must** come from the student's email (the one that includes the student ID). That is, you **must** configure your local git credentials in **every** machine that you will code with

```
    git config --global user.email 123@unicamp.br
```

- **You are not allowed to use libraries that implement the inference, the gradient descent or the models.** You may use, however, a library to do the sampling from the Gaussian. But even that can be achieved by re-parameterizing the Gaussian and then generating random numbers.
- All the submissions must be self-contained and must be executable in a Linux environment. Specifically, your code must execute in the docker image within the Gitlab pipeline.
- Your report must be written in English.
- While there are several images available for LaTeX, I recommend you to use one of the docker images `adnrv/texlive`, available at docker hub (`https://hub.docker.com/r/adnrv/texlive/`). In particular the `basic` and `full` images are standard versions of `texlive`. However, their sizes vary considerably (consequently, the execution times on the pipelines). If you need particular packages it may be easier to take a `basic` image and just install the packages you need using `tlmgr`.
- It is your responsibility to make sure your code compiles and executes correctly. No effort will be made to run your code besides executing the pipeline within the Gitlab repository.
- The proposal for the execution is to use a `Makefile`—§ 5. However, you may use another scripting language for running and executing the code from the stages of the pipeline. No support will be given in this case regarding your decisions on other tools.
- To test your code locally, I recommend a two-fold approach. Execute the code on the docker image by binding your working directory and the docker using volumes. Once your code is running, test the pipelines with a local Gitlab runner. That way, you won't be waiting for the runners to compile, run, and then report errors back to you.

References

[1] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[2] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.