LE49 Probabilistic Machine Learning

Investigative projects 2021/2022

You will write a one-page project proposal (5% of the final module grade) and a project report (65%). The following pages give very brief project suggestions, though you are also free to choose your own project.

The project proposal is to make sure that you think through your project well before the deadline. (In the course of preparing the proposal you might decide that your project isn't a good one after all. You may if you wish switch to a different project, for the report.)

Your proposal should be no more than one page, including references. You should have searched the literature, starting with any papers mentioned in the project brief. What's the big idea that you will explore? What will be your first step to exploring it? What probability models will it be based on? What style of report will you write – a literature review? a data science investigation? methodological research?

- *Analytical literature review.* You will explore the literature yourself, using the project descriptions as a starting point. In your writeup you should bring together common threads across a range of papers, and you should analyse the strengths and weaknesses of the papers rather than simply reporting what they say. You should illustrate your writeup with the output of code that you have run yourself, for example applying the tools from several papers to a common dataset or task.

- *Data science investigation.* You will seek to model a dataset, using the project description as a starting point for the type of model to use. The goal of a data science investigation is to gain insight into a dataset — but all research involves an element of risk, and there is no guarantee that you will find what the project description suggests you might find. Spend your time on the leads that are most interesting and productive. You can still write a perfectly good report about a negative result! In your writeup you should judge a model's strengths and weaknesses, rather than simply fitting it and reporting its output, and you should consider directions for improving the model.

- *Methodological research.* Again, all research involves some element of risk, and the project description is just a starting point. These projects can be seen as a 'taster' for a full MPhil dissertation — they should describe an approach and give some initial ideas, but they need not be comprehensive. For these projects you might well illustrate your approach using synthetic data, and you should construct good examples to illustrate the methodological points that you want to make. For this course, the emphasis is probabilistic modelling as a mathematical activity, not on crunching huge datasets, so illustrate your work using small tractable datasets.

Your project report should be written up in the style of a NIPS or ICML conference paper, 8 pages plus one for references. You should aim to spend around 40 hours on this project. You will be graded on your report, not on your code. You must submit your code, but you needn't spend time tidying it.

The general marking guidelines for coursework are at `https://www.cl.cam.ac.uk/teaching/exams/acs_assessment.html`. For a mark of 70% or higher, you will be expected to use appropriate evaluation criteria for your work. For a mark of 80% or higher, you will be expected to show significant insight or creativity. There are different types of evaluation, and different ways to show insight and creativity, depending on the style of project you choose. Some writeups might involve a mix of styles. Evaluation might involve, for example, inventing metrics and stating them at the beginning of your report; running appropriate cross validation; or running tests to see if your findings are just noise. Insight might mean, for example, using machine learning to discover something interesting and non-obvious in a dataset.

Project 1 (Your project). You may propose your own project, or your own dataset for one of the projects listed here — but you must get approval from Dr Wischik, to confirm your proposal is suitable for this course.

Project 2 (GANs for text, using particle filters). It is a widespread belief that GANs cannot be used to train RNNs for text. See for example `https://arxiv.org/abs/1905.01976` which asserts the problem is "the argmax operator which is not differentiable and blocks the gradient flow from the discriminator to the generator." See also `https://becominghuman.ai/generative-adversarial-networks-for-text-generation-part-1-2b886c8cab10`.

However, if we interpret RNNs as probabilistic models, then it's clear that the *likelihood* of an output is a differentiable function of the neural network weights, and so GAN training should work. In this project, you will train a GAN to generate text from a RNN. This requires Monte Carlo probability estimation, and for this you will use a particle filter.

For background on particle filters, see Dr Wischik's IB Data Science course, lecture notes section 10.4 and example sheet 4.

Project 3 (Function autoencoders). Bayesian regression is based on random functions, e.g. $f(x) = A+Bx+Cx^2+N(0,\sigma^2)$ where A, B, C are random variables. We can equivalently write a random function as $f_\theta(x, Z)$ where f_θ is a deterministic neural network and Z is a latent variable. If we have a dataset of readings from many instances of the function,

$$\mathcal{D} = (\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_n, \boldsymbol{y}_n)$$

where $\boldsymbol{x}_i = [x_{i1}, \ldots, x_{im_i}]$, and $\boldsymbol{y}_i = [y_{i1}, \ldots, y_{im_i}]$ consists of evaluations of instance i of the random function, then we should be able to train the network, autoencoder-style. We'll need an encoder of the form $g_\phi(\boldsymbol{x}_i, \boldsymbol{y}_i)$ to estimate the latent variables, and a decoder of the form $f_\theta(x_{ij}, Z)$. Investigate such autoencoders.

This is the neural network alternative to Bayesian regression and to Gaussian processes.

Project 4 (PCA-like autoencoding). PCA produces an ordered list of orthogonal latent variables. Compare approaches for achieving the same with autoencoders. Relevant papers include *A PCA-like autoencoder* (`https://arxiv.org/abs/1904.01277`) and *Learning Ordered Representations with Nested Dropout* (`https://arxiv.org/abs/1402.0915`).

Project 5 (Autoencoders with hierarchical noise). Consider a generative model in which noise is injected at multiple layers of a neural network, rather than entirely at the first layer. How should the autoencoder be set up? Can such a network achieve crisper output than the standard VAE? This will be a literature review project, with these starting points: *Importance weighted autoencoders* (`https://arxiv.org/abs/1509.00519`) and *Variational Inference with Normalizing Flows* (`https://arxiv.org/abs/1505.05770`).

In lectures, we learnt about autoencoders as type of importance sampling. There is another view, the so-called variational inference view, where the problem is framed as "what is the Gaussian distribution that most closely approximates the posterior $\Pr_Z(z \mid X = x)$*?" You should start by reading the original paper on probabilistic encoders, referred to in lecture notes, to familiarize yourself with the variational inference approach, since that approach is more common in the literature.*

Project 6 (Learning pen strokes from images). Train an autoencoder to encode an MNIST image, and decode it to pen strokes. Starting point: use a RNN to produce the pen strokes, as in *Grammar VAE* (`https://arxiv.org/abs/1703.01925`).

Project 7 (Process mining). There is an area of modelling called *process mining* that consists of making sense of event logs for business processes, for example the sequence of processing steps that happen to an order in a business (place order, send invoice, receive payment, prepare delivery, make delivery). The goal of process mining is to understand commonalities between the processing of different orders, and to discover discrepancies.

Train a probabilistic neural-network sequence generator to generate event logs. Can your probabilistic model be used to answer 'commonalities and discrepancies' type questions?

Example dataset: `https://icpmconference.org/2020/conformance-checking-challenge-ccc-2020/`. Introduction to the field: `http://www.padsweb.rwth-aachen.de/wvdaalst/publications/p1093.pdf`. Startup using process mining to help businesses process their event logs: `https://www.celonis.com/`.

Project 8 (GPT as a probability model). OpenAI's GPT-3 is a generative model for text, which can produce Guardian-editorial level text on demand (`https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3`). Explain how GPT-3 works *as a probabilistic model for random sequences*, paying particular attention to how the output is prompted. Discuss 'few-shot learning' in this context.

This is a literature review project. It will require reading between the lines, since the literature does not give a clear statement of the probability model behind GPT. You should train your own much smaller model on toy data, to illustrate how it works.

Project 9 (Wasserstein GANs are really autoencoders). The Wasserstein GAN (`https://arxiv.org/abs/1701.07875`) is premised on the idea that the white noise term used in the autoencoder's generative model is silly, and that a better way to measure the distance between the model distribution $\Pr_X$ and the dataset distribution $\Pr_{\hat{X}}$ is via the so-called *Earth-Mover* or *Wasserstein* distance. This distance metric involves a maximum over all possible joint distributions of $(\hat{X}, X)$. It is then claimed that it is impractical to directly compute the maximum over joint distributions, and the Wasserstein GAN is an alternative way to compute the same result.

Explain the sense in which an autoencoder *does* find a maximum over all joint distributions of $(\hat{X}, X)$. Does this lead you to any better autoencoder models for e.g. generating MNIST images?

Project 10 (Graphical model for restaurant ranking). Yelp has made available a dataset of restaurants, patrons, and reviews. Star ratings are notoriously unreliable — but it may be that the ordering implied by each user's rankings is more helpful. Build a ranking model for restaurants, where the match outcomes are of the form "User u thinks that restaurant r is better than r'". Experiment with other ways to obtain these outcomes, for example using only pairs of restaurants with one star difference in rating, or restaurants of the same type; also consider how you will treat draws. Evaluate the predictive performance of your model. Also, compare the ordering you obtain to the average star ratings.

Data is at `https://www.yelp.co.uk/dataset/`

Project 11 (Cambridge bumps). Build a ranking model for the Cambridge bumps. You should consider how to model the distinctive features of the bumps, such as overbumps and gradual change in a boat's performance from year to year.

Data is at `http://www.cucbc.org/mays/results`

Project 12 (Multidimensional ranking with graphical models). Using the Yelp review dataset, treat the entirety of a single user's reviews as a document, and encode it into a single variable $\pi_u \in [0, 1]$ for user u. (See note below.) Fit a ranking model similar to TrueSkill, but in which each restaurant r has two dimensions (w_r^1, w_r^2), and a user ranks restaurants based on $\pi_u w_r^1 + (1-\pi_u) w_r^2$. This lets us score restaurants on two separate scales. Interpret these scales. What happens with more dimensions?

Data is at `https://www.yelp.co.uk/dataset/`. This project involves finding a 1d representation of a user's reviews. A simple way to do this is with a mixture model, studied in the LDA section of the course: simply fit a mixture model with $K = 2$ topics and obtain a weighting $(\pi_u, 1 - \pi_u)$ of the two topics for user u's document.

Project 13 (Mixture models for image classification). Yelp has made available a dataset of photos taken by restaurant patrons. Feed these photos into a pre-trained CNN image classifier, and for each photo and for each top-level convolution feature, extract the activation level of that feature pooled across the entire image. Treat each top-level feature as a 'word', and each image as a 'document', and use a categorical mixture model to identify clusters of image types. Experiment with activation thresholds, and with the layer of features you use. Display your groupings appropriately, and relate them to the picture labels.

Data is at `https://www.yelp.co.uk/dataset`

Project 14 (Gaussian process classifiers and CNN uncertainty). Convolutional neural networks (CNNs) achieve state-of-the-art performance on image classification tasks, but provide no measure of confidence in their predictions. Train a CNN image classifier on MNIST. After training the model, use the outputs from the top level feature layer before the softmax classifier to train a Gaussian process classifier. Without re-training, use your model to classify the n-MNIST dataset (`https://www.cl.cam.ac.uk/teaching/2122/DataSci/data/nmnist/`), and also try adversarial perturbations using the CleverHans toolbox. Does

the Gaussian process classifier report lower confidence for images outside its experience? How does the choice of kernel affect this? How should you evaluate a classifier that can report "I don't know"?

This project requires that you have learnt about Gaussian processes, either from L48 or from the Engineering course on Probabilistic Machine Learning. You will also need to learn about Gaussian process classification, sections 3.1–3.4 of Rasmussen's book. Make sure the classifier library you are using returns an appropriate measure of confidence, as per equation (3.24) in section 3.4.

Project 15 (Holdout validation for Bayesians). Bayesian models typically have hyper-parameters, e.g. the parameters of the prior distribution, and Bayesians typically fit them by maximizing marginal likelihood. "Bayesian modelling automatically avoids overfitting, because the prior acts as a regularizer." How true is this? Test this claim, using datasets from the coursework.

There is a large literature about validation and Bayesian inference. In this project you will be expected to run your own experiments, and interpret your findings in the light of a literature survey. Some papers: Bayesian leave-on-out cross-validation approximations for Gaussian latent variable models *by Behtari et al., 2016;* On the marginal likelihood and cross-validation *by Fong and Holmes, 2019.*

Project 16 (The failure of likelihood). It is often reported that generative models trained by maximum likelihood estimation do not perform well when generating new data. Some people drawn the inference is "we need better models, e.g. LSTM rather than fully-connected units for recurrent neural networks", and others draw the inference "likelihood is a poor metric". See for example *A note on the evaluation of generative models*, Theis et al., 2016; *An empirical study on evaluation metrics of generative adversarial networks*, Xu et al., 2018; *How (not) to train your generative model*, Huszár, 2015. The problem is acute with RNNs, where there can be an 'accumulation of error', resulting in generated sequences that seem very unrealistic.

Consider an RNN trained on samples of a Brownian bridge, which is a Brownian motion conditioned to hit a target value at a specified end-time. The Brownian bridge is a convenient model since it is Markovian and there is a simple formula for the likelihood of a sequence; and because there is a clear metric for realism of a generated sequence, namely how close it is to the target value at the end-time. Using this RNN as an example, critically evaluate the claim that 'likelihood is a poor metric for generating realistic sequences'.