Imperial College London

COURSEWORK

PROBABILISTIC INFERENCE (CO-493)

Logistic Regression and MCMC

Logistic Regression

The goal of this coursework is to familiarise yourself with Logistic Regression. We will also see two different approaches to solve the regression task.

For simplicity we will only consider the binary classification case, in which target variables are $y \in \{0, 1\}$.

In logistic regression, the probability of a data point $\boldsymbol{x}$ being of class 1 is given by

$$p(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = \sigma(\boldsymbol{x}^\top \boldsymbol{\theta}) \,,$$

where, $\sigma(z) = 1/(1 + \exp(-z))$ is the *sigmoid* function.
Combining this with a Bernoulli likelihood and summing over all data points $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ we end up with a negative log-likelihood function that looks like this:

$$-\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\sum_i \Big(y_i \log \sigma(\boldsymbol{x}_i^\top \boldsymbol{\theta}) + (1 - y_i) \log(1 - \sigma(\boldsymbol{x}_i^\top \boldsymbol{\theta})) \Big)$$

You will see this expression in many other classification problems, especially in deep learning, where it is known as the *cross-entropy loss*.
Your goal in this tutorial is to learn how to perform inference over the parameters $\boldsymbol{\theta}$ in logistic regression, including point estimates $\boldsymbol{\theta}_{\text{ML}}$ and $\boldsymbol{\theta}_{\text{MAP}}$ and approximations to the posterior $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$.

Provided to you are **logistic_regression_assignment.py**, a skeleton file in which you will provide solutions, and **logistic_regression_extra.ipynb**, a jupyter notebook that can be used for visualisation purposes. The notebook provides plots of the regression and might help the understanding.

Submit your final version to CATe via the LabTS system.

Task 1: 10 marks

Complete the definition of the function **predict**. This function takes data points $\boldsymbol{X}$ and a parameter vector $\boldsymbol{\theta}$ and returns $p(y = 1|\boldsymbol{x}, \boldsymbol{\theta})$ for $x \in \boldsymbol{X}$.

Task 2: 10 marks

Complete the definition of the function **log_likelihood**. This function takes data points $\boldsymbol{X}$, labels $\boldsymbol{y}$ and a parameter vector $\boldsymbol{\theta}$ and returns $\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$.

Task 3: 10 marks

The likelihood function does not have a closed form solution and similar to GP coursework we will use numerical optimisation to fit parameters.

In the previous coursework you saw how to derive the gradients and optimise likelihood with them. For this coursework you can do it that way, or any other way you want. The optimisation is convex, so numerical difference [default in scipy-optimizers] should also work and give you same answer. With correct gradients you should converge in 1 or 2 iterations.
Complete the function **max_lik_estimate** to optimise the log-likelihood function you've written above and obtain θ_{ML}.

Bayesian logistic regression

To move on to Bayesian inference on the parameters θ, we put a prior on the parameters.
More specifically, we use a Gaussian prior parametrised by a mean $\boldsymbol{m}$ and a covariance S:

$$\theta \sim \mathcal{N}(\boldsymbol{m}, S)$$

Similarly to θ_{ML}, θ_{MAP} does not have an analytical solution and we are required to use numerical optimisation methods.

Maximum A Posteriori (MAP)

First, we build a MAP estimate of $\boldsymbol{m}$ and S

Task 4: 10 marks

Complete the definition of the function **neg_log_posterior**. This function takes data points X, labels y, prior mean $\boldsymbol{m}$ and covariance S as well as a parameter vector θ and returns the negative log posterior.

Task 5: 10 marks

Complete the function **map_estimate** to optimise the negative log posterior you've written above and obtain θ_{MAP}.

The Laplace approximation

As we have hinted above, in logistic regression the posterior distribution over θ doesn't have an analytical solution. This is the an example of *approximate Bayesian inference*: The exact posterior is analytically intractable so that we have to approximate it using one of various techniques. The one we'll use in this part of the coursework is called the "Laplace approximation".
In brief, the Laplace approximation is a Gaussian centred at the peak of the pdf of interest with the same curvature. Let us make this a bit more rigorous below.

Let us say we have a probability distribution $p(\boldsymbol{z})$ we want to approximate. The distribution $p(\boldsymbol{z})$ is of the form

$$p(\boldsymbol{z}) = \frac{1}{Z}\tilde{p}(\boldsymbol{z}) \,,$$

where $\tilde{p}(\boldsymbol{z})$ is an unnormalised distribution that we can evaluate easily, but Z is unknown. Formally, the Laplace approximation results from a second-order Taylor expansion of $\log \tilde{p}(\boldsymbol{z})$ around $\boldsymbol{z}_0$:

$$\log \tilde{p}(\boldsymbol{z}) \approx \log \tilde{p}(\boldsymbol{z}_0) + \frac{d}{d\boldsymbol{z}} \log \tilde{p}(\boldsymbol{z})\Big|_{z=z_0} (\boldsymbol{z} - \boldsymbol{z}_0) + \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_0)^\top \frac{d^2}{d\boldsymbol{z}^2} \log \tilde{p}(\boldsymbol{z})\Big|_{z=z_0} (\boldsymbol{z} - \boldsymbol{z}_0)$$

Now let us evaluate this expression at the mode of $p(\boldsymbol{z})$, which is the same as the mode of $\tilde{p}(\boldsymbol{z})$. We define the mode $\boldsymbol{z}^*$ such that

$$\frac{d}{d\boldsymbol{z}} \tilde{p}(\boldsymbol{z})\Big|_{z=z^*} = \mathbf{0} \,.$$

At this point, the $\mathcal{O}(\boldsymbol{z})$ term of the expansion vanishes and we are left with

$$\log \tilde{p}(\boldsymbol{z}) \approx \log \tilde{p}(\boldsymbol{z}^*) - \frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}^*)^\top \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{z}^*)$$

Or, equivalently,

$$\tilde{p}(\boldsymbol{z}) \approx \tilde{p}(\boldsymbol{z}^*) \exp\Big(-\tfrac{1}{2}(\boldsymbol{z} - \boldsymbol{z}^*)^\top \boldsymbol{A} (\boldsymbol{z} - \boldsymbol{z}^*)\Big) \,,$$

where

$$\boldsymbol{A} = -\frac{d^2}{d\boldsymbol{z}^2} \log \tilde{p}(\boldsymbol{z})\Big|_{z=z^*} \,.$$

Because this distribution is Gaussian, we can easily calculate the normalisation constant. By inspection, we can identify the mean and the covariance, and write down the Laplace approximation of $p(\boldsymbol{z})$ as

$$q(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z} | \boldsymbol{z}^*, \boldsymbol{A}^{-1})$$

Task 6: 10 marks

As an example, let us use the unnormalized distribution $\tilde{p}(z) = xe^{-x/2}$. When normalized properly, this is in fact the χ^2 distribution with $k = 4$ degrees of freedom.

Complete the function **laplace_q** that takes an input vector z and returns the value of the Laplace approximation $q(z)$ of $\tilde{p}(z) = xe^{-x/2}$.

Task 7: 20 marks

So far, we have obtained the mode (peak) of the posterior through the MAP estimate above. Here we are extending this to a posterior distribution over $\boldsymbol{\theta}$. However, as we mentioned above the posterior doesn't have an analytical form, so we will use the **Laplace transform**.

Complete the function **get_posterior**, based on your previous code, that will calculate the Laplace approximation $q(\boldsymbol{\theta})$ of the true posterior $p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})$ and return the mean and variance of q.

The jupyter notebook offers code to visualise the behaviour and the diversity of q by drawing a number $j = 1,...,J$ of samples $\boldsymbol{\theta}_j \sim q(\boldsymbol{\theta})$. For each sample it plots its predicted class probabilities $\sigma(x\boldsymbol{\theta}_j)$.

Hint: The extension of the Laplace approximation to multivariate distributions is straightforward, and in this case the variance of the Gaussian is the inverse Hessian of the negative log posterior $\boldsymbol{A} = -\nabla_\theta \nabla_\theta \log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y})$.

MCMC Sampling

The Laplace approximation is part of a family of methods known as *deterministic approximate inference*. In addition, there's another set of methods known as *stochastic approximate inference* which, includes most of the sampling techniques mentioned in the course.
Here we will implement the Metropolis algorithm that uses a proposal distribution $q(\boldsymbol{x}'|\boldsymbol{x}^{(t)})$ that depends on the last sample $\boldsymbol{x}^{(t)}$. We then perform the Metropolis algorithm:

1. Generate proposal $\boldsymbol{x}' \sim q(\boldsymbol{x}'|\boldsymbol{x}^{(t)})$
2. If
$$\frac{q(\boldsymbol{x}^{(t)}|\boldsymbol{x}')\tilde{p}(\boldsymbol{x}')}{q(\boldsymbol{x}'|\boldsymbol{x}^{(t)})\tilde{p}(\boldsymbol{x}^{(t)})} \geq u\,, \qquad u \sim U[0,1]$$
accept the sample $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}'$. Otherwise set $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$.

Task 8: 20 marks

Complete the function **metropolis_hastings_sample** that returns nb_iter samples of the true posterior $p(\boldsymbol{\theta} \mid \boldsymbol{X},\boldsymbol{y})$ with prior $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{m},\boldsymbol{S})$ using the Metropolis algorithm.