CS6487 2019B - Assignment 1

Bayesian Supervised Learning - Gaussian Processes

Antoni Chan
Department of Computer Science
City University of Hong Kong

In this assignment you will derive a new Gaussian Process model for regression, implement, and test it.

Regression for Check-in Time

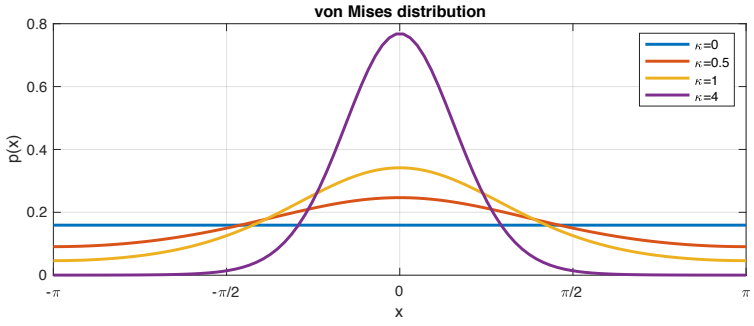
We are interested in modeling social-media check-ins. In particular, the goal is to predict the time-of-day using the check-in location. That is the input of the regression is the check-in coordinates ($x \in \mathbb{R}^2$), and the output is the time-of-day (t). Note that the values of time-of-day are circular, i.e., they wrap around from 23:59:59 to 00:00:00. Hence, we need to select a likelihood function that works with this type of data. Suppose that we have normalized the time of day t between $[0, 2\pi]$, i.e., it is an angle in radians. The Von Mises distribution (also called a circular normal distribution) is a probability distribution over a circle, given by:

$$p(t|\mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(t-\mu)}, \tag{1}$$

where μ is the mean, κ is the precision parameter (larger values yield peaks around μ), and $I_\nu(z)$ is the "modified Bessel function of the first kind" with order ν,

$$I_\nu(z) = \frac{1}{\pi} \int_0^\pi e^{z \cos y} \cos(\nu y) dy. \tag{2}$$

Here is a plot of a few example pdfs of the Von-Mises distribution with $\mu = 0$:



Regression model

To perform regression using the Von-Mises distribution as the observation likelihood, we first rewrite the distribution according to parameter vector $\theta = \begin{bmatrix} \kappa \cos \mu \\ \kappa \sin \mu \end{bmatrix}$ with vector $y = \begin{bmatrix} \cos t \\ \sin t \end{bmatrix}$,

$$p(y|\theta) = \frac{1}{2\pi I_0(\|\theta\|)} e^{y^T \theta}. \tag{3}$$

The equivalence can be shown using trigonometry identities. The parameter θ now encompasses both the mean and the precision. In particular, the length of θ determines the precision κ, and the angle of θ gives the mean μ. Finally, note that y can only take on coordinate values along the unit circle, i.e, $||y|| = 1$.

We can now specify the regression model by setting θ as the outputs of two independent latent functions. The latent function vector with GP prior is

$$f = \begin{bmatrix} f^{(1)} \\ f^{(2)} \end{bmatrix}, \tag{4}$$

$$f^{(1)} \sim \mathcal{GP}(0, k(x, x')), \tag{5}$$

$$f^{(2)} \sim \mathcal{GP}(0, k(x, x')), \tag{6}$$

and the likelihood model is

$$p(y|f) = \frac{1}{2\pi I_0(\|f\|)} e^{y^T f}. \tag{7}$$

Again note that the latent function vector f controls both the mean and the precision of the Von-Mises distribution. When the latent function vector is $f = 0$, then $p(y|f)$ becomes a uniform distribution, which fits nicely with the zero-mean GP prior. When f is large magnitude, then $p(y|f)$ is peaked around the mean – since the GP prior regularizes the length of f, then this prevents the regression model from becoming "too confident".

Inference

Let $\{(y_i, x_i)\}_{i=1}^N$ be the training dataset, where $y_i \in \mathbb{R}^2$ and $x_i \in \mathbb{R}^D$. Let $Y = [y_1, \cdots, y_N]$ and $X = [x_1, \cdots, x_N]$. For input point x_i, define the corresponding latent function vector $f_i = [f_i^{(1)}, f_i^{(2)}]^T$. The collection of all latent function vectors is $\mathbf{f} = [f_1^{(1)}, \cdots, f_N^{(1)}, f_1^{(2)}, \cdots, f_N^{(2)}]^T$, and the prior is given as,

$$p(\mathbf{f}|X) = \mathcal{N}(\mathbf{f}|0, \begin{bmatrix} K & 0 \\ 0 & K \end{bmatrix}). \tag{8}$$

The data likelihood is

$$p(Y|\mathbf{f}) = \prod_{i=1}^N p(y_i|f_i) = \prod_{i=1}^N \frac{1}{2\pi I_0(\|f_i\|)} e^{y_i^T f_i}. \tag{9}$$

The posterior of the function values is computed using Bayes' rule,

$$p(\mathbf{f}|X, Y) = \frac{p(Y|\mathbf{f})p(\mathbf{f}|X)}{p(Y|X)}, \tag{10}$$

where the marginal likelihood is $p(Y|X) = \int p(Y|\mathbf{f})p(\mathbf{f}|X)d\mathbf{f}$.

Given a novel point x_*, we compute the predictive distribution of vector $f_* = \begin{bmatrix} f_*^{(1)} \\ f_*^{(2)} \end{bmatrix}$,

$$p(f_*|X, Y, x_*) = \int p(f_*|\mathbf{f}, X, x_*)p(\mathbf{f}|X, Y)d\mathbf{f}, \tag{11}$$

and finally the predictive distribution of y_*,

$$p(y_*|X, Y, x_*) = \int p(y_*|f_*)p(f_*|\mathbf{f}, X, x_*)df_*. \tag{12}$$

Part 1 Approximate Inference

In the first problem, you will derive an approximate inference algorithm for the above regression model.

(a) Use the Laplace approximation to derive an algorithm to approximate the posterior $p(\mathbf{f}|X,Y)$ as a multivariate Gaussian distribution $q(\mathbf{f}|X,Y)$.

(b) Using the approximation $q(\mathbf{f}|X,Y)$, derive an approximation to the the predictive latent distribution $p(f_*|X,Y,x_*)$.

(c) Design an approximation to the predictive output distribution $p(y_*|X,Y,x_*)$ using Gauss-Hermite quadrature.

.....

Note: the following identities might be useful:

$$\frac{\partial}{\partial\kappa}I_0(\kappa) = I_1(\kappa), \tag{13}$$

$$\frac{\partial}{\partial\kappa}I_1(\kappa) = \frac{1}{2}(I_0(\kappa)+I_2(\kappa)) = \frac{1}{\kappa}I_1(\kappa)+I_2(\kappa) = I_0(\kappa)-\frac{1}{\kappa}I_1(\kappa) \tag{14}$$

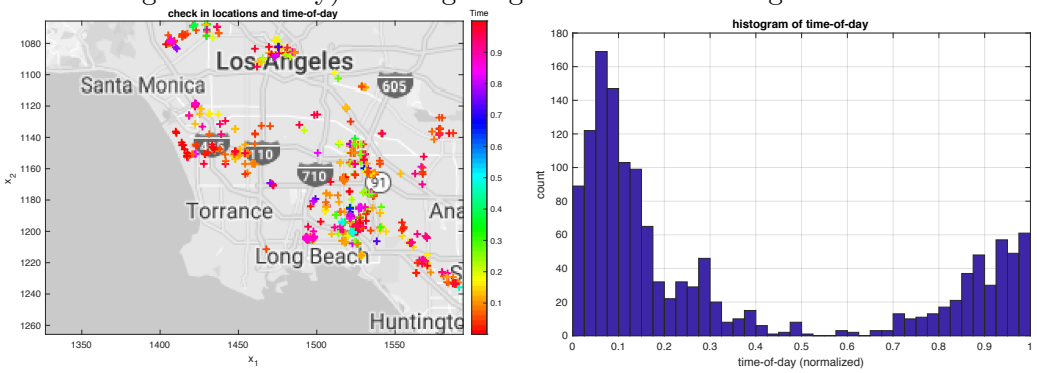
More information about Bessel functions can be found here.

Part 2 Implementation and Experiments

The check-in data can be found in the provided file `social-data.zip`. Each user has one csv file, containing the following variables (each row is a sample check-in):

- column 1 – the x-coordinates on the SoCal map (`map.png`) of the check-in.
- column 2 – the y-coordinates on the SoCal map (`map.png`) of the check-in.
- column 3 – normalized time-of-day (between 0 and 1). You will need to multiply by 2π to get the radian angle representation.

The map `map.png` is provided for visualizing the check-in data. Here is an example plot for User 9. The left figure shows the check-in location and normalized time (as the marker color) on the map (zoomed into LA region for clarity). The right figure shows the histogram of check-in times.



In the second problem, you will implement the above model and inference algorithm and test it on real data.

(a) Implement the algorithm for approximating the posterior $p(\mathbf{f}|X,Y)$ and the predictive latent distribution $p(f_*|X,Y,x_*)$. Using the predictive distribution, plot the latent function on the map. Try different kernel functions and compare the results.

(b) Implement the the algorithm for approximating the predictive output distribution $p(y_*|X, Y, x_*)$. Plot the mean and variance of the output distribution on the map. How well can the regression method interpolate values? How is the variance/uncertainty characterized by the location of the data points?

Notes: the bessel function is implemented in MATLAB as `besseli` and in SciPy as `scipy.special.iv`. One way to visualize the latent function vectors on the image is use the color to indicate direction (using H in the HSV colorspace), and the brightness as the magnitude (using V or S in HSV). This is similar to how optical flow is visualized.

.....

- **What to hand in** – You need to turn in the following things:

  1. Answers, plots, analysis, discussion, etc. for the above problems.
  2. Source code files.

  You must submit your Assignment using the Canvas website.

- **Plagiarism** – You should implement the regression algorithm using your own code. Do not use someone else's implementation of the regression algorithms. It is okay to use common library components and functions, e.g. standard matrix operations (e.g., `inv`, `eig`), general purpose optimization toolboxes (e.g., `quadprog`, `linprog`), and special functions (e.g., quadrature approximation, and bessel functions). If you use any special toolboxes or libraries, make sure to note them in your report.

- **Grading** – The marks for this assignment will be distributed as follows:

  – 45% – Derivation of the approximate inference algorithm (1a, 1b, 1c).
  – 25% – Implementation of the algorithms (2a).
  – 20% – Experiment results demonstrating the regression algorithm (2b).
  – 10% – Quality of the written report. More points given to insightful observations and analysis.

  Note: if you cannot implement the algorithms correctly, it is okay to use 3rd party software. You will not get marks for implementation, but you can still get marks for presenting the results.