10-715 Advanced Introduction to Machine Learning

Homework 4 *Due Nov 21, 10.30 am*

Rules

1. Homework is due on the due date at 10.30 am. Please hand over your homework to the TAs at their office or slide it under their doors. *Please see course website for policy on late submission.*
2. The homework is due on a Friday. We will count the weekends towards your late day quota. E.g. if you have two late days left, the latest you can hand your homework in is by Sunday 10.30 am.
3. We recommend that you typeset your homework using appropriate software such as LaTeX . If you are writing please make sure your homework is cleanly written up and legible. The TAs will not invest undue effort to decrypt bad handwriting.
4. You must hand in a hard copy of the homework. The only exception is if you are out of town in which case you can email your homeworks to *both* the TAs. If this is the case, your homeworks *must* be typeset using proper software. Please do *not* email written and scanned copies. Your email must reach the TAs by 10.30 am on the due date.
5. You are allowed to collaborate on the homework, but should write up your own solution and code. Please indicate your collaborators in your submission.
6. Please hand in the solutions to Problem 1 and Problem 2 separately. Write your name, andrew id and department on both submissions.
7. Please staple your homeworks.
8. If you are confused about of any of the terms you may refer Wikipedia. We have introduced some new concepts and methods that were not discussed in class. You should be able to find all of the required definitions on Wikipedia.
9. Note that this is a short homework. You have one week, as opposed to the usual two weeks to hand it in.

---

1 Gaussian Processes (Samy)

1.1 GP Implementation

In this problem you will implement Gaussian Process Regression. While there are several packages available for GPs (e.g. GPML: gaussianprocess.org/gpml/code), here you will implement it yourself.

In this subsection, we will be using the squared exponential covariance,

$$K(x, x') = \sigma \exp\left(\frac{-(x - x')^2}{2h^2}\right) + \eta\delta(x - x')$$

where σ, h are the scale and bandwidth parameters of the Kernel and η is the noise parameter. We also need to choose the prior mean function $\mu(x)$ for the GP.

Implementing GP Regression is just a few lines of code. Algorithm 2.1 in page 19 of Rasmussen and Williams provides the pseudo code. We recommend that you follow this – especially the matrix inversions – since they are numerically stable.

Note that the pseudo code given is for a $GP(\mathbf{0}, K)$ prior where as here we will be using a $GP(\mu, K)$ prior where μ is the prior mean function. To account for this you should subtract the mean from the training ponits before computing the posterior and then add it back to your estimate. Let $\tilde{y} = y - \mu(x)$ and $\tilde{\mathbf{y}}$ denote the vector of observations with the mean subtracted. Then equations 2.18 to 2.30 follow by replacing $\mathbf{y}$ by $\tilde{\mathbf{y}}$. The only difference is that you need add the mean back to the mean of the posterior GP. Equation 2.25 (using the same notation in the book) becomes

$$\bar{f}_* = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \tilde{\mathbf{y}} + \mu(X_*)$$

We will test GP regression on a $1D$ toy dataset and the Combined Cycle Power Plant (CCPP) dataset from the UCI repository. For the latter, we will attempt to predict the net electrical energy output (EP) using the other four features–temperature(T), ambient pressure(AP), relative humidity(RH), and exhaust vacuum(V). We have given the dataset as a mat file and separated it into training and testing sets so you could use what is given.

You should write code for the following.

1. **(9 Points)** Implement a function `GPRegression`. This should take in training data, the test points and the hyper parameters for the GP. It should return the posterior mean and the posterior covariance matrix.

2. **(9 Points)** Implement a function `chooseHyperParams` to pick the hyper parameters for GP Regression. For simplicity, we will set μ to be a constant function at the mean of the labels (y values) and η to be 1% of the standard deviation of y. `chooseHyperParams` should implement these.
   To choose σ, h you can either use cross validation or maximize the GP Marginal Likelihood (see pg 19, equation 2.30 in Rasmussen & Williams).

3. **(9 Points)** Recall that a GP is distribution over functions. So the samples we draw from this distribution are functions. We can observe the values of these samples at any finite set of points.
   Implement a function `GPDrawSamples` to draw samples from a Gaussian Process. Suppose we want to observe the sample at m points in the input space. It should take in the mean ($m \times 1$) of the GP at these points and the variance ($m \times m$) and the number of samples (N). It should return a $N \times m$ matrix with each sample in one row.

Results (9 Points): For the toy problem you should report, a figure showing the true curve, the posterior mean and 100 samples drawn from the posterior GP. For the CCPP dataset you should report, the average error on the test set and the posterior mean and covariance of the first five points in the test set.

If you are using Matlab, the given starter code does this for you so you can just attach the screen shots. This was our output for the toy problem. Your solution could be different due to hyper parameter selection but it should be close.

```
>> q11
Error: 0.2233

ans =
   25.5487
   25.5181
   25.4879
   25.4587
   25.4313


ans =
    0.0753    0.0707    0.0650    0.0583    0.0507
    0.0707    0.0670    0.0621    0.0561    0.0492
    0.0650    0.0621    0.0580    0.0528    0.0467
    0.0583    0.0561    0.0528    0.0485    0.0432
    0.0507    0.0492    0.0467    0.0432    0.0388
```

Please submit all your code with the homework.

1.2 Gradients using GPs

In a typical regression problem we have labels $\mathbf{y} = y_1, \ldots, y_n$ of the function at points $\mathbf{X} = \{x_1, \ldots, x_n\}$. Here $y_i = y(x_i) \in \mathbb{R}, x_i = [x_i^{(1)}, \ldots x_i^{(d)}] \in \mathbb{R}^d$. We are interested in estimating the value of the function $y_* = y(x_*)$ at a new point x_*. Alternatively, we may also be interested in the gradient of the function at x_*. GPs provide a nice framework for obtaining an estimate for the gradient and quantifying the uncertainty.

Recall the gradient at x_* is a d-vector containing the d partial derivatives. Denote the partial derivatives of y as $g_i(x) = \partial y(x)/\partial x^{(i)}$. where $g_i : \mathbb{R}^d \to \mathbb{R}$, $i = 1, \ldots, d$. The gradient is $\mathbf{g}(x) = [g_1(x), \ldots, g_d(x)]^\top$. Write $g_{i*} = g_i(x_*)$ and $\mathbf{g}_* = \mathbf{g}(x_*)$.

We will model our observations as a Gaussian Process $y(x)$ over $\mathbb{R}^d$ with zero mean and auto covariance K. We know that if a function y is sampled from a GP, the distribution of $y_1, \ldots, y_n, y_*$ at $x_1, \ldots, x_n, x_*$ follow a Gaussian distribution. It can also be shown that the gradient $\mathbf{g}_*$ at x_* is also a Gaussian. Further, the gradient $\mathbf{g}_*$ and the function values $y_1, \ldots, y_n$ are jointly Gaussian–i.e. $[\mathbf{y}, \mathbf{g}_*] = [y_1, \ldots, y_n, g_1(x_*), \ldots, g_d(x_*)] \in \mathbb{R}^{n+d}$ is also a Gaussian.

This Gaussian has zero mean. The covariance K_i between $y(x_1)$ and $g_i(x_2)$ and the covariance K_{ij} between $g_i(x_1)$ and $g_j(x_2)$ are given respectively by,

$$K_i(x_1, x_2) = \frac{\partial K(x_1, x_2)}{\partial x_2^{(i)}}$$
$$K_{ij}(x_1, x_2) = \frac{\partial^2 K(x_1, x_2)}{\partial x_1^{(i)} \partial x_2^{(j)}}$$

1. **(4 Points)** Let $\mathbf{K} \in \mathbb{R}^{n \times n}$, $\mathbf{J} \in \mathbb{R}^{n \times d}$ and $\mathbf{B} \in \mathbb{R}^{d \times d}$ such that,

$$\mathbf{K}_{ij} = K(x_i, x_j) \qquad \mathbf{J}_{ij} = \frac{\partial K(x_i, x_*)}{\partial x_*^{(j)}} \qquad \mathbf{B}_{ij} = \frac{\partial^2 K(x_*, x_*)}{\partial x_*^{(i)} \partial x_*^{(j)}}$$

   Write the (Gaussian) prior over $[\mathbf{y}, \mathbf{g}_*]^\top$ in terms of $\mathbf{K}, \mathbf{J}, \mathbf{B}$ if we don't have any observations yet.

2. **(2 Points)** Obtain the posterior for $\mathbf{g}_*$ given that $\mathbf{y}$ was observed at $\mathbf{X}$.

3. **(2 Points)** Verify that the posterior mean of $\mathbf{g}_*$ is the same as the gradient of the posterior mean for $y(x_*)$.

4. **(6 Points)** Now we want to estimate the integral of a continuous $1D$ function $y(x)$ over the interval $[a, b]$ from labels $\mathbf{y} = y_1, \dots, y_n$ at points $X = \{x_1, \dots, x_n\}$ where $y_i = y(x_i)$ and $x_i \in \mathbb{R}$. By imposing a $\mathrm{GP}(\mathbf{0}, K)$ prior on $Y(x) = \int_0^x y(t)\mathrm{d}t$, outline a procedure to obtain an estimate and a posterior distribution for the integral of the function.

2 Dirichlet Process Mixture Model(Veeru)

In classical data clustering, the number of clusters is specified apriori or selected by cross-validation. We can use Dirichlet Process mixture(DPM) models and accommodate arbitrarily large number of clusters automatically.

2.1 Understanding the generative process

Recall that the generative process of a DPM can be described as follows. We want to generate points $x_1, x_2, \cdots, x_n$ (say in $\mathbb{R}^2$) forming some clusters. For that we first generate cluster centers μ_i as in (1) for $i \in [n]$ and then sample x_i from a distribution parametrized by μ_i as in (2). In the Chinese Restaurant metaphor, μ_is correspond to tables and x_i's correspond to customers.(1) can be understood as follows: After drawing the cluster centers $\mu_1, \mu_2, \cdots, \mu_{i-1}$, the new draw μ_i is either set to one of the previous cluster centers μ_j with probability $\frac{1}{i-1+\alpha}$ each or to a new cluster center drawn from the base distribution F_0. Here $\alpha > 0$ is called scaling/concentration parameter and F_0 is called base distribution. Note that, we expect repetitions in $\mu_1, \mu_2, \cdots, \mu_{i-1}$ by construction. Also note that μ_i is set to a previous cluster μ_j with a probability proportional to the number of times μ_j appears in $\mu_1, \mu_2, \cdots, \mu_{i-1}$. On a different note, observe that we can obtain the generative process for Dirichlet process by drawing x_i from $\delta(\mu_i)$.

$$\mu_i|\mu_1, \cdots, \mu_{i-1} \sim \frac{1}{i-1+\alpha}\sum_{j=1}^{n-1}\delta(\mu_j) + \frac{\alpha}{i-1+\alpha}F_0 \tag{1}$$

$$x_i \sim f(.|\mu_i) \text{ such as } N(\mu_i, \sigma^2) \tag{2}$$

In this problem, assume that $f(.|\mu) = N(\mu, \sigma^2 = I)$. If you are using Matlab, set the random seed to 0 by `rng(0);` before generating the samples each time.

1. **(5 points)** Generate $n = 200$ points in $\mathbb{R}^2$ using the above process, with $\alpha = 1$, $F_0 = N(\mu = 0, 25I)$ and plot them. How many clusters have been generated and how many points are generated from each cluster? Call this case I.

2. **(5 points)** Keep $\alpha = 1$ but generate more points this time. Try $n = 2000$ and $n = 2e5$. Do you observe a smaller or larger number of clusters compared to case I? Convince yourself that this behaviour is consistent by making a few random runs.

3. **(5 points)** Repeat case I with $\alpha = 2n = 400$. Do you observe a smaller or larger number of clusters compared to case I? Justify.

2.2 Inference with Gibbs sampling

Suppose we are given data $x = (x_1, \cdots, x_n)$ and we want to find the posterior distribution $p(x_*|x)$. Suppose z_i is a positive integer indicating the cluster assignment for x_i. You can use Gibbs sampling to draw samples from the posterior $p(z, \mu|x)$ and then compute $p(x_*|x)$ from these samples. For Gibbs sampling you need to compute the conditionals $p(z_i = k|\mu, x, z_{-i})$ and $p(\mu_k = u|x, z, \mu_{-k})$. The latter one is easy to compute and you may refer to the previous homework to see that it is:

$$p(\mu_k = u|x, z, \mu_{-k}) \propto p(\mu_k = u) \prod_{\{i:z_i=k\}} p(x_i|z_i = k, \mu_k = u) \tag{3}$$

Now let us see how to compute $p(z_i|\mu, x, z_{-i})$. Assume that there are K clusters in the observed data of this conditional, that is, the set $\{z_j|j \neq i\}$ has size K. Note that z_i can be one of the cluster assignments in $\{z_j|j \neq i\}$ or a new cluster number K_{new}. The exact value of K_{new} is immaterial. For concreteness, set it to $1 + \max\{z_j|j \neq i\}$.

2.2.1 Conditional distributions(10 points)

For $k \in \{z_j|j \neq i\}$, show that, for some constant C,

$$p(z_i = k|\mu, x, z_{-i}) = Cp(x_i|z_i = k, \mu_k)p(z_i = k|z_{-i}) = C\frac{n_{-i,k}}{n-1+\alpha}p(x_i|z_i = k, \mu_k) \tag{4}$$

where $n_{-i,k}$ is the number of points assigned to cluster k, excluding point i from the count.

For the new cluster assignment, show that,

$$\begin{aligned} p(z_i = K_{new}|\mu, x, z_{-i}) &= Cp(z_i = K_{new}|z_{-i}) \int p(x_i|z_i = K_{new}, \mu)\pi(\mu)d\mu \\ &= C\frac{\alpha}{n-1+\alpha} \int p(x_i|z_i = K_{new}, \mu)\pi(\mu)d\mu \end{aligned} \tag{5}$$

where C is the constant in (4) and $\pi(\mu)$ is the pdf of F_0.

2.2.2 Implementation(20 points)

Plugin the known distributions and simplify the update equations (4), (5), (3). Implement Gibbs sampling on the 2D data `q2.mat` for x given with the homework. Use $\alpha = 1, F_0 = N(0, 25I)$. Update z's before μ's. Use a burn-in period of $B = 1000$ and draw $T = 100$ samples. Plot x_i's and clearly color code them with cluster assignments for the last sample from the Gibbs sampling procedure.

You may refer to Algorithm 2 in Radford Neal's paper in the reading list on the course website for better understanding of the update steps.

2.2.3 Prediction(5 points)

Given a new point x_*, and T samples from the posterior $p(z, \mu|x_1, x_2, \cdots, x_n)$, how do you compute the posterior density(that is, density after observing $x_1, x_2, \cdots, x_n$) at x_*? Hint: First draw $\mu_{n+1}^{(t)}$ for samples $t = 1, 2, \cdots, T$ based on $\mu^{(t)}$. For further hints, you may again refer to Radford Neal's paper.