Small Problem 1: Bayesian Linear Regression

Query 1: Posterior distribution of the estimated weight vector.

Metrics

Metric 1
Let $P(\widehat{w}|D)$ be the posterior distribution of the estimated weight vector. Metric 1 is the expected squared error $\mathbb{E}[\|\widehat{w} - w\|^2]$under this distribution.

Metric 2
We have provided samples from the true posterior distribution $P_{true}(\widehat{w}|D)$. Estimate the total variation distance between the provided samples and the samples from your estimate $P(\widehat{w}|D)$:

$$\int_w |P(\widehat{w}|D) - P_{true}(\widehat{w}|D)|\, dw$$

Ground Truth:

For metric 1, the ground truth is the weight vector given in the problem details document. For ease of reference, this weight vector is also given here:

$$w = (-1.731855, 2.986017, 2.698284, -3.591651, -3.714157)$$

For metric 2, samples from the true posterior distribution are in the file `problem-1-solution-samples.xlsx` (without headers in the .csv version) and the columns represent the values of w1, w2, w3, w4, w5.

TODO:

Compute metric 1 using the ground truth weight vector given above.

Compute metric 2, the total variation between ground truth samples and the samples generated by your solution.

For the metric 2 computation, we have provided a Java program that that computes the total variation between ground truth samples and the samples generated by your solution. The Java program is provided as both a jar file and as source code.

The jar file is "`totalvar.jar`"
The source code is located in folder `problem-1-total-variation-java`.

The file containing your samples should be a comma separated value (CSV) file without column headers where the columns represent the dimension of the weight vector (in this case there should be 5 columns corresponding to w1, w2, w3, w4, and w5), and the rows represent samples.

To evaluate your samples using the jar file, use the command line given below:
`java –jar totalvar.jar <ground-truth-file-path> <input-file-path>`

Evaluation output will be written to `stdout`.

To evaluate your samples using the provided source code, follow these steps:
Find the variable "`static String dataFileA`" in the "`Main.java`" file and set its value to be the complete path of the CSV file that contains the ground truth samples, e.g. the complete path to the `problem-1-solution-samples.csv` file referenced above.

Find the variable "`static String dataFileB`" in the "`Main.java`" file and set its value to be the complete path of the CSV file that contains your samples.

Run the program using the main method in this same "`Main.java`" file and the program will produce a total variation score. The method "`computeScore`" is also provided if you wish to programmatically evaluate your samples.

Submit the metric and your code as described in the main CP4 problem description document, e.g. PPAML Challenge Problem 4-v7.pdf.

Note that we have included a detailed description of the "total variation" computation procedure in "`problem-1-total-variation.pdf`".

Ground Truth Details:

This problem can be solved using the Stan probabilistic programming language (Stan Development Team, 2014). Here is the Stan/R program that estimates the posterior distribution of the estimated weight vector.

Step 1: Save the following code as a Stan file (attached as "`cp4_1.stan`")

```
data {
  int<lower=0> N;
  int<lower=1> K;
  matrix[N, K] x;
  vector[N] y;
}

parameters {
  vector[K] w_mean;
  cov_matrix[K] w_prec;
  vector[K] w;
  real<lower=0> noise_sd;
}

model {
  // Just the identity matrix
  matrix[K,K] idK;
  idK <- diag_matrix(rep_vector(1.0,K));

  w_mean ~ normal(0,2);
  w_prec ~ wishart(K, idK);
  //w ~ multi_normal_prec(w_mean, w_prec);
  w ~ normal(w_mean,1);
```

```
  // The second parameter is the scale 0.5,
  // corresponding to a rate parameter of 2.
  noise_sd ~ inv_gamma(0.5, 0.5);

  y ~ normal(x * w, noise_sd);
}
```

Step 2: Save the following code as a R file (attached as "cp4_1.R")

```
setwd("~/galois/ppaml/challenge-problems/cp4/")

xy <- read.csv("problem-1-data.csv")
x <- xy[,2:6]
y <- xy[,7]

theData <- list(
    x=x
  , y=y
  , K=ncol(x)
  , N=nrow(x)
)

library(rstan)
sm <- stan_model("cp4_1.stan")

# For MAP estimate
map <- optimizing(sm, data=theData)

# To sample from posterior
fit <- sampling(sm, data=theData)

samp <- extract(fit)
write.table(samp$w,"cp4_1_w_posterior.csv",row.names=F, col.names=F, sep=',')
```