Assignment 7 for STA 601

(Richard) Fangjian Guo

`NetID: fg47`

Department of Computer Science

September 24, 2013

Let us denote the test set with (y_1, y_2) and the training set (y_1^o, y_2^o).

1 *Oracle prediction*

The oracle predictor is simply

$$\hat{y}_1^{(\text{oracle})} = E[y_1 \mid y_2] = \mu_{1|2},$$

where

$$y_1 \mid y_2 \sim N(\mu_{1|2}, \sigma_{1|2}^2).$$

And the predictor is computed with true model parameters μ and Σ.

2 *MLE prediction*

The MLE predictor is again

$$\hat{y}_1^{(\text{MLE})} = E(y_1 \mid y_2) = \hat{\mu}_{1|2},$$

where $\mu_{1|2}$ is estimated with training set data (y_1^o, y_2^o), namely

$$\hat{y_1}^{(\text{MLE})} = \hat{\mu}_{1|2} = \hat{\mu}_1 + \hat{\Sigma}_{12}(\hat{\Sigma}_{22}^{-1})(y_2 - \hat{\mu}_2).$$

$\hat{\mu}$ and $\hat{\Sigma}$ are estimated from the training set as sample mean and sample covariance respectively.

3 *Bayesian prediction*

The Bayesian predictor is the mean for the posterior predictive distribution for each y_2, i.e.

$$\hat{y_1}^{(\text{Bayes})} = E(y_1 \mid y_2, \{y_1^o, y_2^o\}).$$

This predictor can be computed with the following procedure.

For each y_2, and for each iteration $s = 1, 2, \cdots, S$,

1. Sample $\mu^{(s)}$ and $\Sigma^{(s)}$ from the posterior distribution with Gibbs sampling.

2. Compute the parameters for conditional distribution with
$$\mu_{1|2}^{(s)} = \mu_1^{(s)} + \Sigma_{12}^{(s)}(\Sigma_{22}^{(s)})^{-1}(y_2 - \mu_2^{(s)}),$$
   and
$$\sigma_{1|2}^{2(s)} = \Sigma_{11}^{(s)} - (\Sigma_{12}^{(s)})^2(\Sigma_{22}^{(s)})^{-1}.$$
3. Sample $y_1^{(s)}$ from the conditional distribution $N(\mu_{1|2}^{(s)}, \sigma_{1|2}^{2(s)})$.

Then the predictor is estimated as their average, namely
$$\hat{y_1}^{(\text{Bayes})} = \frac{1}{S}\sum_{s=1}^{S} y_1^{(s)}.$$
The predictive interval is also estimated from the quantiles of these samples.

4 *Results*

The Bayesian predictor with 95% predictive interval, aside with the test set, is given by Figure 1.

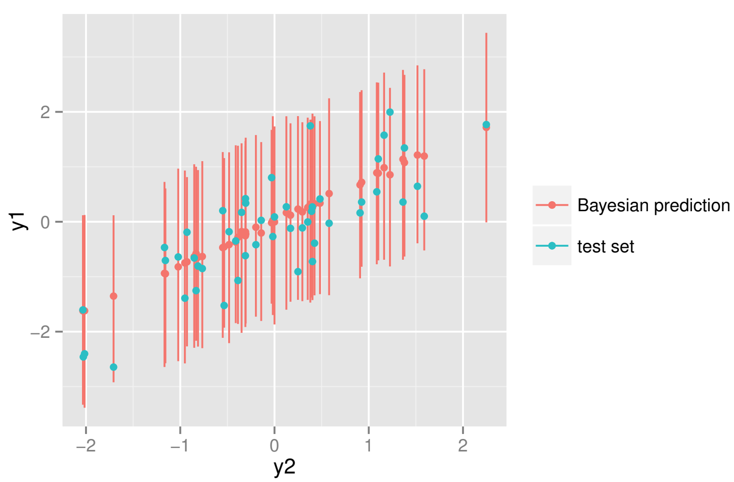

Figure 1: The Bayesian predictor with 95% predictive interval, compared with the test set.

As can be noted in the figure, the coverage rate of the predictive intervals is 100%.

The MSE (Mean Square Error) for Bayesian, MLE and oracle predictors is summarized in Table 1. Bayesian is slightly better than MLE predictor. The three predictors are compared to the test set data in Figure 2.

5 *Comparing MLE and linear regression*

The two models, despite their different definitions, actually result in the same predictor function, as shown in Figure 3. This is because

	Bayesian	MLE	Oracle
MSE	0.36949	0.37238	0.36908

Table 1: MSE

Figure 2: The predicted values compared with test set.

the close form solution to linear regression is the same as the MLE predictor.

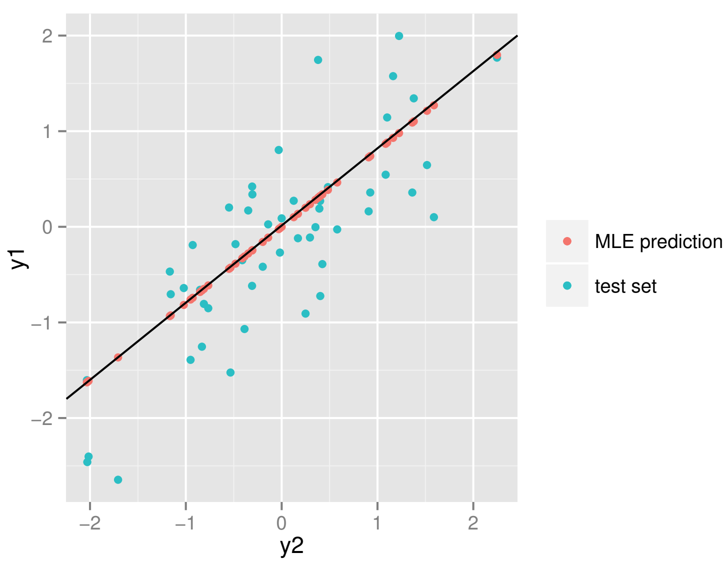

Figure 3: The MLE predictor compared with linear regression line.

6 *Appendix: R code*

```
library(ggplot2)
```

```r
library(reshape)
library(mvtnorm)
library(monomvn)

# training set
mu <- c(0,0)
sigma <- matrix(c(1,0.8,0.8,1), nrow=2, ncol=2, byrow=T)
y.sp <- rmvnorm(100, mu, sigma)

# new instances (test set)
n <- 50
y.new <- rmvnorm(n, mu, sigma)

# oracle predicitions
predict.oracle <- mu[1] + sigma[1,2]/sigma[2,2] * (y.new[,2] - mu[2])
MSE.oracle <- mean((predict.oracle-y.new[,1])^2)

# MLE estimates
mu.mle <- apply(y.sp, 2, mean)
sigma.mle <- cov(y.sp)
## MLE gaussian conditional expectation
predict.mle <- mu.mle[1] + sigma.mle[1,2]/sigma.mle[2,2] * (y.new[,2] - mu.mle[2])
MSE.mle <- mean((predict.mle-y.new[,1])^2)

# Bayesian
# hyperpara
mu.0 <- c(0,0)
Lambda.0 <- matrix(c(1,0.5,0.5,1), nrow=2, byrow=T)
S.0 <- Lambda.0
nu.0 <- 4
# consts
y.mean <- mu.mle
## Gibbs sampler
## posterior predictive for each y.new[,2]
B <- 500
S <- 500
sigma.run <- sigma.mle
y1.predict.bayes.mat <- NULL
for (t in 1:(B+S)) {
    Lambda.run <- solve(solve(Lambda.0) + n*solve(sigma.run))
    mu.run <- Lambda.run %*% (n*solve(sigma.run) %*% y.mean) # mu.0 is zero here
    theta.run <- rmvnorm(1, mu.run, Lambda.run)
    S.run <- S.0 + (t(y.sp)-c(theta.run)) %*% t((t(y.sp)-c(theta.run)))
    sigma.run <- solve(rwish(nu.0+n, solve(S.run)))
    # burn-in
    if (t<=B) {
      next
    }
    # posterior predictive
    # parameters for each new y2
    sigma1.conditional <- (sigma.run[1,1] - sigma.run[1,2]/sigma.run[2,2]*sigma.run
        [1,2]) * rep(1,n)
    mu1.conditional <- theta.run[1] + sigma.run[1,2]/sigma.run[2,2]*(y.new[,2]-theta.
        run[2])
    y1.tmp <- numeric(n)
    # sampling conditional
    for (j in 1:n) {
      y1.tmp[j] <- rnorm(1, mu1.conditional[j], sqrt(sigma1.conditional[j]))
    }
    y1.predict.bayes.mat <- rbind(y1.predict.bayes.mat, y1.tmp)
}
```

```
# prediction
predict.bayes.mean <- apply(y1.predict.bayes.mat, 2, mean)
predict.bayes.interval.left <- apply(y1.predict.bayes.mat, 2, function(x) quantile(x,
        probs=c(0.025)))
predict.bayes.interval.right <- apply(y1.predict.bayes.mat, 2, function(x) quantile(x,
        probs=c(0.975)))
MSE.bayes <- mean((predict.bayes.mean-y.new[,1])^2)
interval.hit <- sum(as.numeric(predict.bayes.interval.left <= y.new[,1] & y.new[,1]
        <= predict.bayes.interval.right))/n

# plotting
## plot
predict.df <- data.frame(y1=y.new[,1], y2=y.new[,2],
                         predict.mle, predict.oracle, predict.bayes.mean,
                         predict.bayes.interval.left, predict.bayes.interval.right)
fig1 <- ggplot(predict.df) +
  geom_point(aes(x=y2, y=predict.bayes.mean, color="Bayesian prediction")) +
  geom_point(aes(x=y2, y=y1, color="test set")) +
  geom_point(aes(x=y2, y=predict.mle, color="Oracle predictition")) +
  geom_point(aes(x=y2, y=predict.oracle, color="MLE prediction")) +
  theme(legend.title=element_blank()) + ylab("y1")
print(fig1)
```