ST340 Lab 9: Gaussian Processes

2019–20

Load in the required libraries for data manipulation and multivariate normal distribution:

```
require(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.5.2
```

```
require(plyr)
require(reshape2)
require(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.2
```

Set a seed for repeatable plots:

```
set.seed(12345)
```

Continuous observations

The squared exponential (SE, or RBF) covariance function with length scale ℓ can be calculated as follows:

```
calcK <- function(X1,X2,l=1) {
  K <- matrix(0, nrow=length(X1), ncol=length(X2))
  for (i in 1:nrow(K)) {
    for (j in 1:ncol(K)) {
      K[i,j] <- exp(-0.5*(abs(X1[i]-X2[j])/l)^2)
    }
  }
  return(K)
}
```

(a) Run the following code to plot some sample functions drawn from a GP:

```
x.star <- seq(-5,5,len=50)
Kmat <- calcK(x.star,x.star)
n.samples <- 3
values <- matrix(rep(0,length(x.star)*n.samples), ncol=n.samples)
for (i in 1:n.samples) {
  # Each column represents a sample from a multivariate normal distribution
  # with zero mean and covariance Kmat
  values[,i] <- mvrnorm(1, rep(0, length(x.star)), Kmat)
}
values <- cbind(x=x.star,as.data.frame(values))
values <- melt(values,id="x")
# Plot the result
ggplot(values,aes(x=x,y=value)) +
  geom_rect(xmin=-Inf, xmax=Inf, ymin=-2, ymax=2, fill="grey80") +
  geom_line(aes(group=variable)) +
  theme_bw() +
  scale_y_continuous(lim=c(-2.5,2.5), name="output, f(x)") +
  xlab("input, x")
```

Now let's assume that we have some known data points;

```
f <- data.frame(x=c(-4,-3,-1,0,2),
                y=c(-2,0,1,2,-1))
```

(b) Calculate the covariance matrices using the same `x.star` values as above:

```
x <- f$x
k.xx <- calcK(x,x)
k.xxs <- calcK(x,x.star)
k.xsx <- calcK(x.star,x)
k.xsxs <- calcK(x.star,x.star)
f.star.bar <- k.xsx%*%solve(k.xx)%*%f$y
cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx)%*%k.xxs
```

(c) Draw 50 samples from the multivariate Normal distribution $\mathbf{f}^* \sim \mathcal{N}(\bar{\mathbf{f}}^*, \boldsymbol{\Sigma}^*)$ and plot them as above.

(d) Try some different **known** data points `f` with the previous code, to see how that changes the resulting plot.

(e) Now assume that each of the observed data points has some normally-distributed noise:

```
sigma.n <- 0.1
f.bar.star <- k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))%*%f$y
cov.f.star <- k.xsxs - k.xsx%*%solve(k.xx + sigma.n^2*diag(1, ncol(k.xx)))%*%k.xxs

# YOUR CODE HERE

ggplot(values, aes(x=x,y=value)) +
  geom_line(aes(group=variable), colour="grey80") +
  geom_errorbar(data=f,aes(x=x,y=NULL,ymin=y-2*sigma.n, ymax=y+2*sigma.n), width=0.2) +
  geom_point(data=f,aes(x=x,y=y)) +
  theme_bw() +
  scale_y_continuous(lim=c(-3,3), name="output, f(x)") +
  xlab("input, x")
```

(f) Try some different values for the noise standard deviation `sigma.n` with the previous code, to see how that changes the resulting plot.

GP classifier

Install the variational Bayes code from BioConductor:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
   install.packages("BiocManager")
 BiocManager::install("vbmp", version = "3.8")

# For older versions of R, use:
# source("https://bioconductor.org/biocLite.R")
# biocLite("vbmp")
```

(a) Generate some synthetic data:

```
th=runif(100,-pi,pi)
y= ifelse(th < 0, 1, 2)
x1=sin(th)-y/3+rnorm(100,sd=0.1)
x2=cos(th)+y/2+rnorm(100,sd=0.1)
x1=c(x1,rnorm(50,-2,0.2))
x2=c(x2,rnorm(50,1.5,0.2))
y=c(y,rep(3,50))
plot(x1,x2,asp=1,col=y,pch=y)

idx <- sample.int(length(y),length(y)/2)
train.X <- cbind(x1[idx],x2[idx])
train.Y <- y[idx]
test.X <- cbind(x1[-idx],x2[-idx])
test.Y <- y[-idx]
```

(b) Use a multinomial probit GP with ARD covariance to classify the points:

```
library(vbmp)
theta <- runif(ncol(train.X))
resSym <- vbmp(train.X, train.Y, test.X, test.Y, theta,
                 list(sKernelType="gauss", bThetaEstimate=TRUE, maxIts=50,
                      InfoLevel=1, bMonitor=TRUE))
covParams(resSym)
predError(resSym)
```

(c) Try classifying the MNIST images as '1', '2', or '3':

```
load("mnist.tiny.rdata")
train.idx123 <- which(train.labels %in% c(1,2,3))
test.idx123 <- which(test.labels %in% c(1,2,3))
newXtrain <- train.X[train.idx123,]/255
newYtrain <- train.labels[train.idx123]
newXtest <- test.X[test.idx123,]/255
newYtest <- test.labels[test.idx123]
theta <- rep(1., ncol(train.X))

# YOUR CODE HERE
```

(d) Fit a multiclass SVM and compare the results.