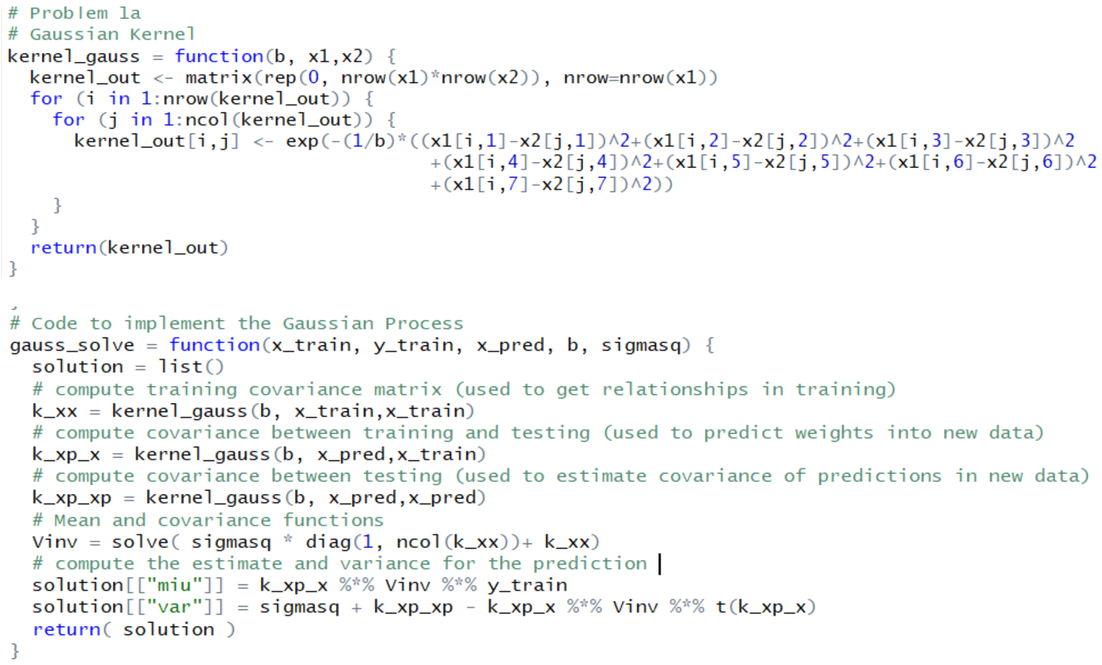
ELEN E4903: MACHINE LEARNING HOMEWORK 3

Problem 1:

a)



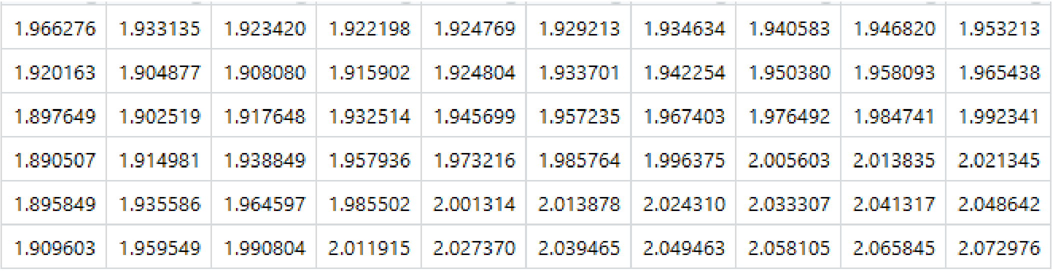
```
# Problem 1a
# Gaussian Kernel
kernel_gauss = function(b, x1,x2) {
  kernel_out <- matrix(rep(0, nrow(x1)*nrow(x2)), nrow=nrow(x1))
  for (i in 1:nrow(kernel_out)) {
    for (j in 1:ncol(kernel_out)) {
      kernel_out[i,j] <- exp(-(1/b)*((x1[i,1]-x2[j,1])^2+(x1[i,2]-x2[j,2])^2+(x1[i,3]-x2[j,3])^2
                                +(x1[i,4]-x2[j,4])^2+(x1[i,5]-x2[j,5])^2+(x1[i,6]-x2[j,6])^2
                                +(x1[i,7]-x2[j,7])^2))
    }
  }
  return(kernel_out)
}

# Code to implement the Gaussian Process
gauss_solve = function(x_train, y_train, x_pred, b, sigmasq) {
  solution = list()
  # compute training covariance matrix (used to get relationships in training)
  k_xx = kernel_gauss(b, x_train,x_train)
  # compute covariance between training and testing (used to predict weights into new data)
  k_xp_x = kernel_gauss(b, x_pred,x_train)
  # compute covariance between testing (used to estimate covariance of predictions in new data)
  k_xp_xp = kernel_gauss(b, x_pred,x_pred)
  # Mean and covariance functions
  Vinv = solve( sigmasq * diag(1, ncol(k_xx))+ k_xx)
  # compute the estimate and variance for the prediction
  solution[["miu"]] = k_xp_x %*% Vinv %*% y_train
  solution[["var"]] = sigmasq + k_xp_xp - k_xp_x %*% Vinv %*% t(k_xp_x)
  return( solution )
}
```

Above is the screenshot of the code used to implement the Gaussian process and make predictions on test data. The central idea behind the code is to obtain $\mu(x) = K(x, D_n)(\sigma^2 I + K_n)^{-1} y$ and $\sum(x) = \sigma^2 + K(x,x) - K(x, D_n)(\sigma^2 I + K_n)^{-1} K(x, D_n)^T$ given measured data $D_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where the distribution of $y(x)$ can be calculated at any new x to make predictions and $K(x, D_n) = [K(x, x_1), \ldots, K(x, x_n)]$ with K_n being the n x n kernel matrix restricted points in D_n.

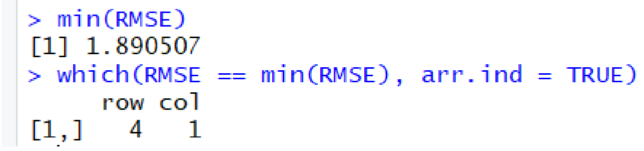
b) Table 1 below shows the RMSE on the 42 test points with the mean of the Gaussian process at the test point as the prediction

Table 1: RMSE for the 42 test points



						σ^2					
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
	5	1.966276	1.933135	1.923420	1.922198	1.924769	1.929213	1.934634	1.940583	1.946820	1.953213
	7	1.920163	1.904877	1.908080	1.915902	1.924804	1.933701	1.942254	1.950380	1.958093	1.965438
b	9	1.897649	1.902519	1.917648	1.932514	1.945699	1.957235	1.967403	1.976492	1.984741	1.992341
	11	1.890507	1.914981	1.938849	1.957936	1.973216	1.985764	1.996375	2.005603	2.013835	2.021345
	13	1.895849	1.935586	1.964597	1.985502	2.001314	2.013878	2.024310	2.033307	2.041317	2.048642
	15	1.909603	1.959549	1.990804	2.011915	2.027370	2.039465	2.049463	2.058105	2.065845	2.072976

c) The best value is **1.890507 which occurs when b = 11 and σ^2= 0.1** as shown by the output below.



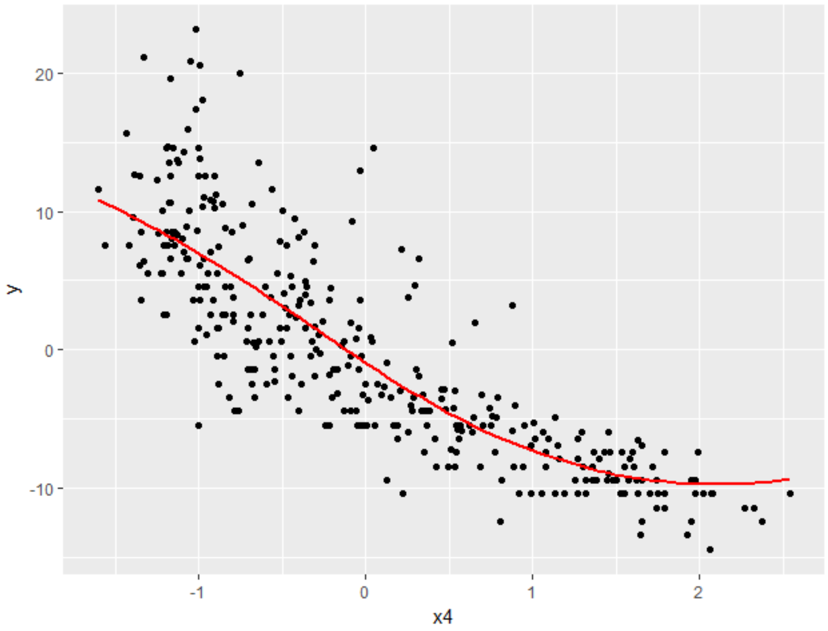
```
> min(RMSE)
[1] 1.890507
> which(RMSE == min(RMSE), arr.ind = TRUE)
     row col
[1,]   4   1
```

(continued next page)

From homework 1, the lowest RMSE in part c was 2.633644 at λ = 0 (which corresponded to least squares as a preferred model to ridge as the optimal regularization parameter is 0) while in part d the lowest RMSE was 2.192574 at p = 2 and λ = 23. Both of the lowest RMSEs in homework 1 are higher than the lowest RMSE of 1.890507 of this homework. Therefore, it can be seen from the best (lowest) value for the RMSE of the prediction from the Gaussian process with the Gaussian kernel that **the Gaussian process with the Gaussian kernel does better in predicting y than the linear regression model in homework 1.** The better (lower) RMSE of the Gaussian process in this homework can be attributed to the fact the Gaussian kernel took into account higher ordered dimensions than what was considered in homework 1, thereby leading to better predictions in the testing data.

The **drawback** of the approach in this homework compared to homework 1 is **computation time**. The computation time for a Gaussian process regression in this homework scales exponentially with the number of points because the kernel matrices which take into account the relationship between the predictors, x, of every point in the training and testing data to form infinitely many dimensions (as shown by the Gaussian kernel in code in part 1a) is needed to form predictions. This is in contrast to simply solving an optimization problem to obtain the parameters needed to obtain the prediction in the ridge regression of homework 1.

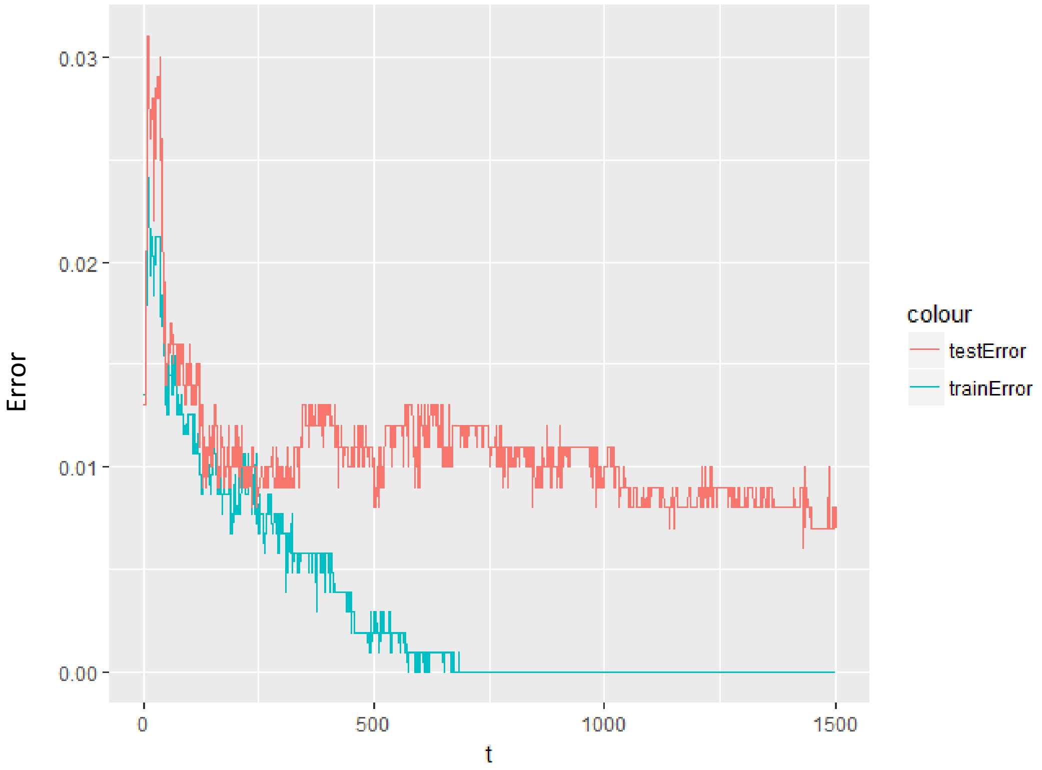
d) **Figure 1: Scatterplot of x[4] against y for b = 5 and σ^2= 2 with solid red line being the predictive mean of the Gaussian process at each point in the training set.**



ELEN E4903: MACHINE LEARNING HOMEWORK 3

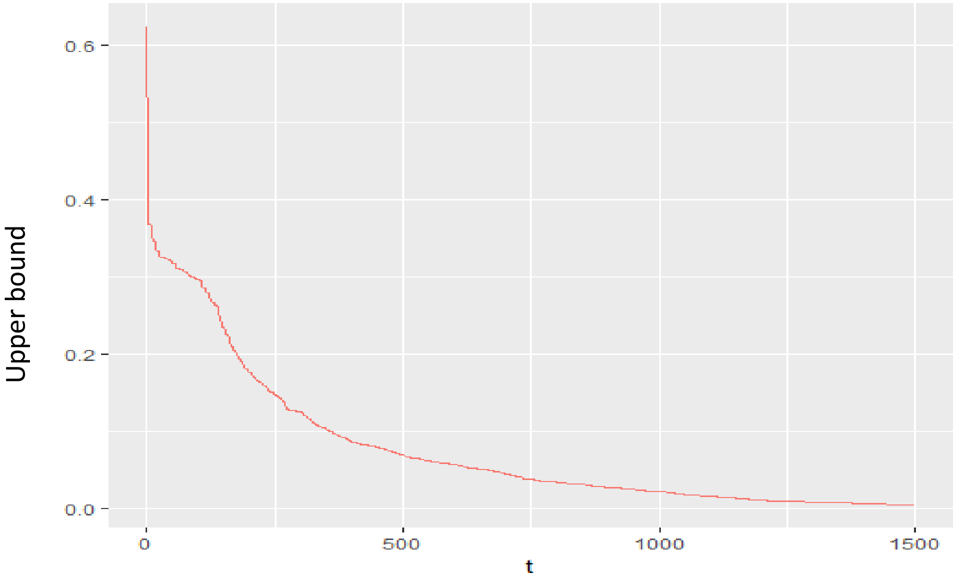
Problem 2:

a) **Figure 1: Plot of training and testing error of $f_{boost}^{(t)}(.)$ for t = 1, …, T and T = 1500 rounds.**



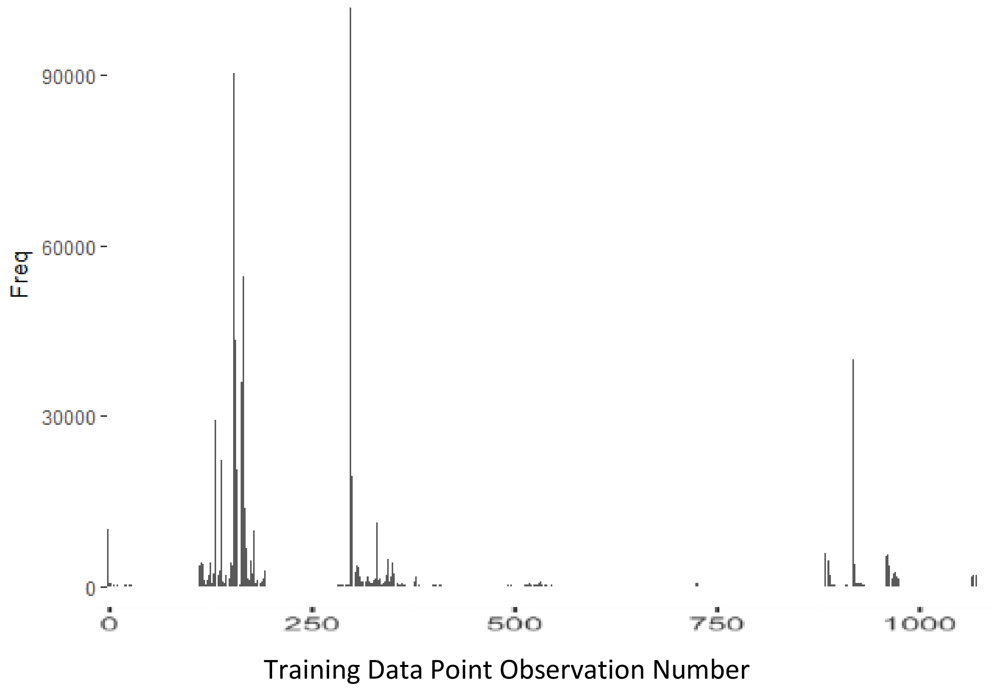
What figure 1 shows is that boosting drives the training error to 0 without causing overfitting as can be seen by how the testing error continues to decrease as t increases.

b) **Figure 2: Plot of the upper bound on the training error** $\exp\left(-2\sum_{t=1}^{T}(\frac{1}{2}-\epsilon_t)^2\right)$ **as a function of t**



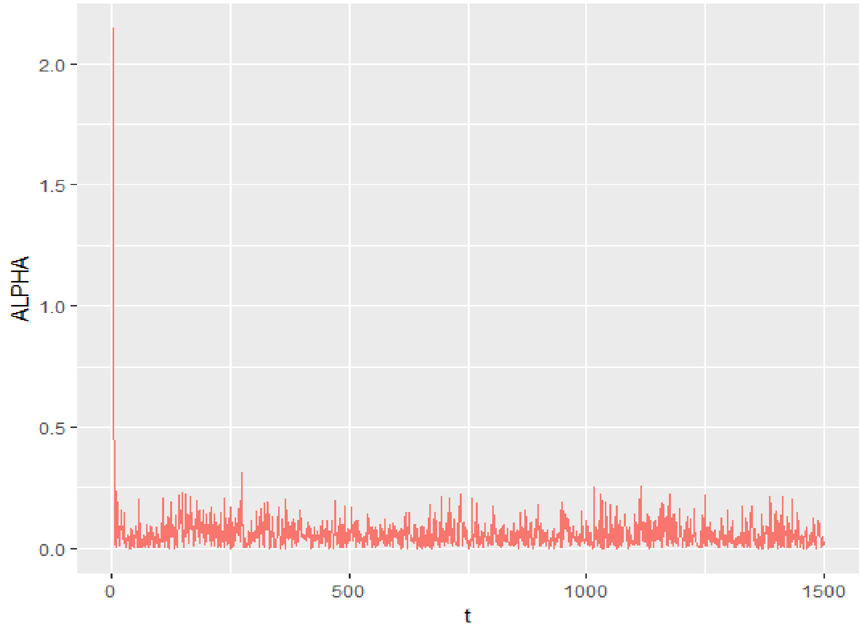
It can be seen from figure 2 above that the upperbound of the training error decreases as t increases.

c) Figure 3: Histogram of the total number of times each training data point was selected by bootstrap method across all rounds

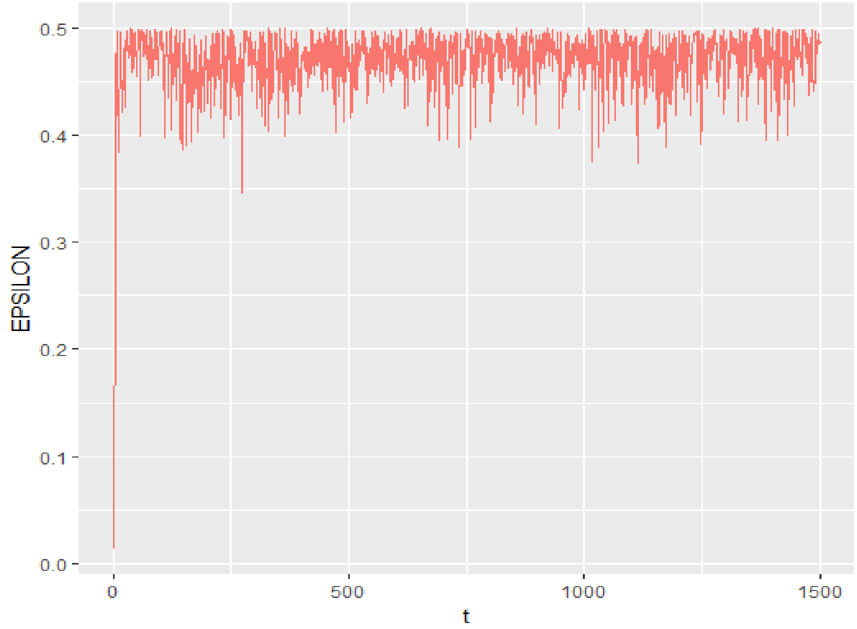


It can be seen from the above histogram that some observations between 125 and 250 as well as between 250 and 500 were selected more than others. This is due to them being misclassified more often at the initial stages so that higher weight were given to them in the bootstrapping stage of the ADABoost algorithm so that they would end up being selected more often to get the classification correct.

d) **Figure 4: Plot of α_t as a function of t**



e) **Figure 5: Plot of ϵ_t as a function of t**



It can be seem from figure 4 and 5 that α_t falls quite quickly to a value between 0 to 0.25 and stays around that value while ϵ_t rises very quickly to a value between 0.4 to 0.5 and stays there as t increases.