Machine Learning (732A99) Examination

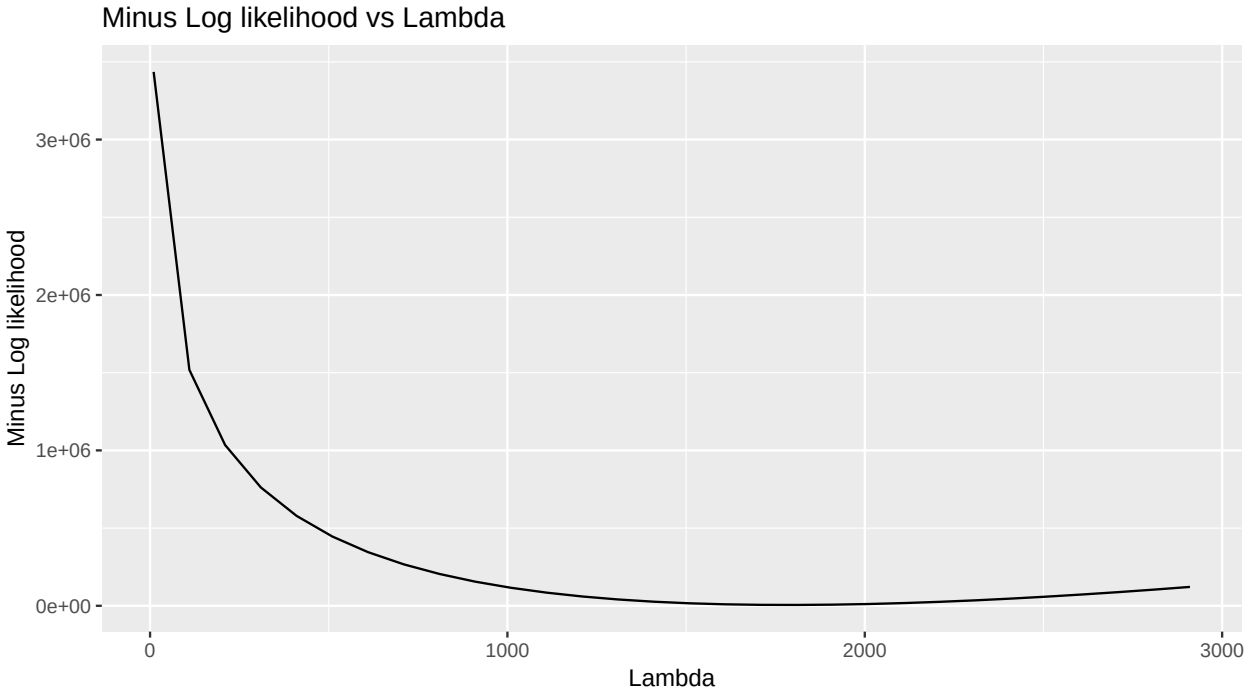
Contents

Note: All codes are available in the appendix.

Assignment 1

Task 1.1

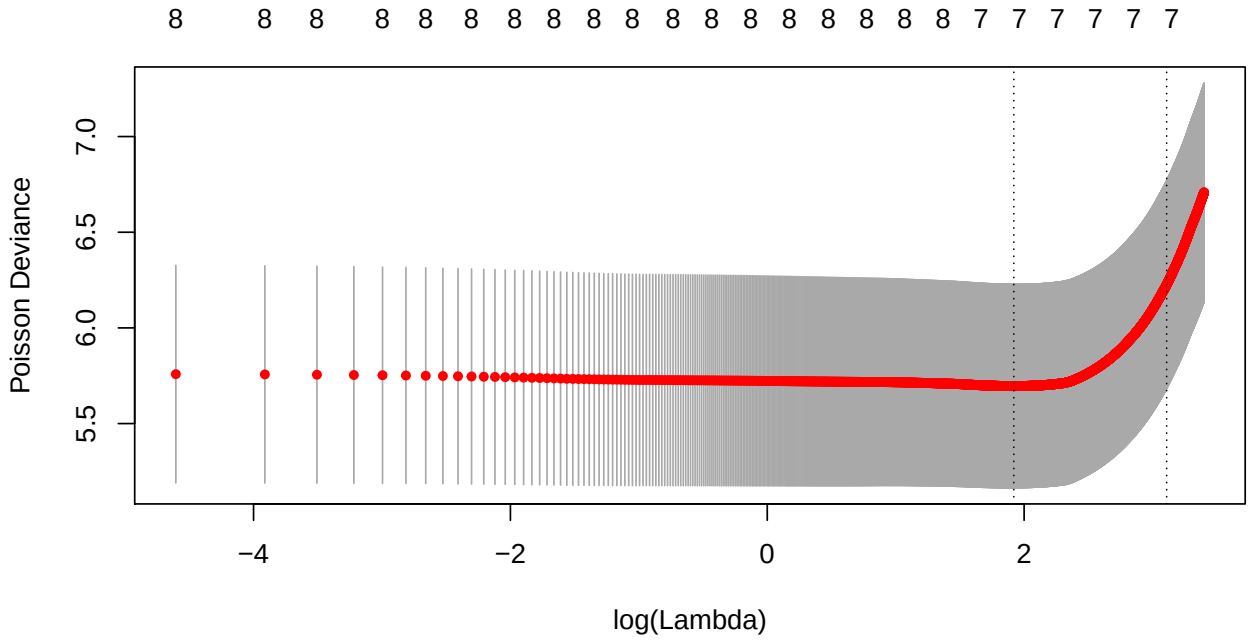
2p



The optimal value of λ is the one for which the likelihood is maximum. That is, the minus log likelihood should be minimum. By observing the above plot showing the dependence of minus log-likelihood on λ, we can observe that the minimum of minus log likelihood occurs around λ values 1500 to 2000. Hence, the optimal λ can be approximately 1800. The exact value can be determined by looking at the actual values in the data frame.

Task 1.2

All variables except mortality are scaled. The data is divided into train and test in the ratio 50/50. A LASSO regression is fit with Mortality as a Poisson distributed target and all other variables as features. The following plot shows the variation of poisson deviance with $log(\lambda)$ obtained using cross validation.




```
## [1] "Optimal lambda:  6.82"
```

The optimal LASSO penalization parameter, lambda is 6.82 which has the minimum value of poisson deviance. This model has the following MSE values for train and test.

```
## [1] "Train MSE:  9239.75507647494"
```

```
## [1] "Test MSE:  11734.2336428137"
```

Is the MSE actually the best way of measuring the error rate in this case?

MSE is not the best way to measure the error rate in this case. For a poisson target, the distribution is not symmetric. MSE is optimal for a normally distributed target. The cv.glmnet function actually uses poisson deviance to calculate the CV error by default for a Poisson distributed target.

Optimal LASSO coefficients

3p

The coefficients in the optimal model are as follows.

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                    s0
## (Intercept)           7.481344759
## Year                  .
## Week                 -0.007079788
## Influenza             0.015557250
## Temperature.deficit   0.017858702
## Influenza_lag1        0.005085960
## Temp_lag1             0.007629540
## Influenza_lag2        0.022457409
## Temp_lag2             0.007323202
```

We see that the Year variable is rejected by the LASSO regression (coef = 0). All other variables contribute to mortality. The variable Influenza_lag2 has the highest impact on mortality while the variables Temperature.deficit and Influenza also have much higher impact on mortality compared to the remaining variables.

```
## [1] "alpha =  7.48134475939446"
```

```
## [1] "exp(alpha) =  1774.62561652386"
```

The value of α is 7.48134475939446 and the value of $exp(\alpha)$ is 1774.62561652386. The optimal value of λ from step 1 was approximately 1800. These values appear to be similar.

Should these quantities be similar?

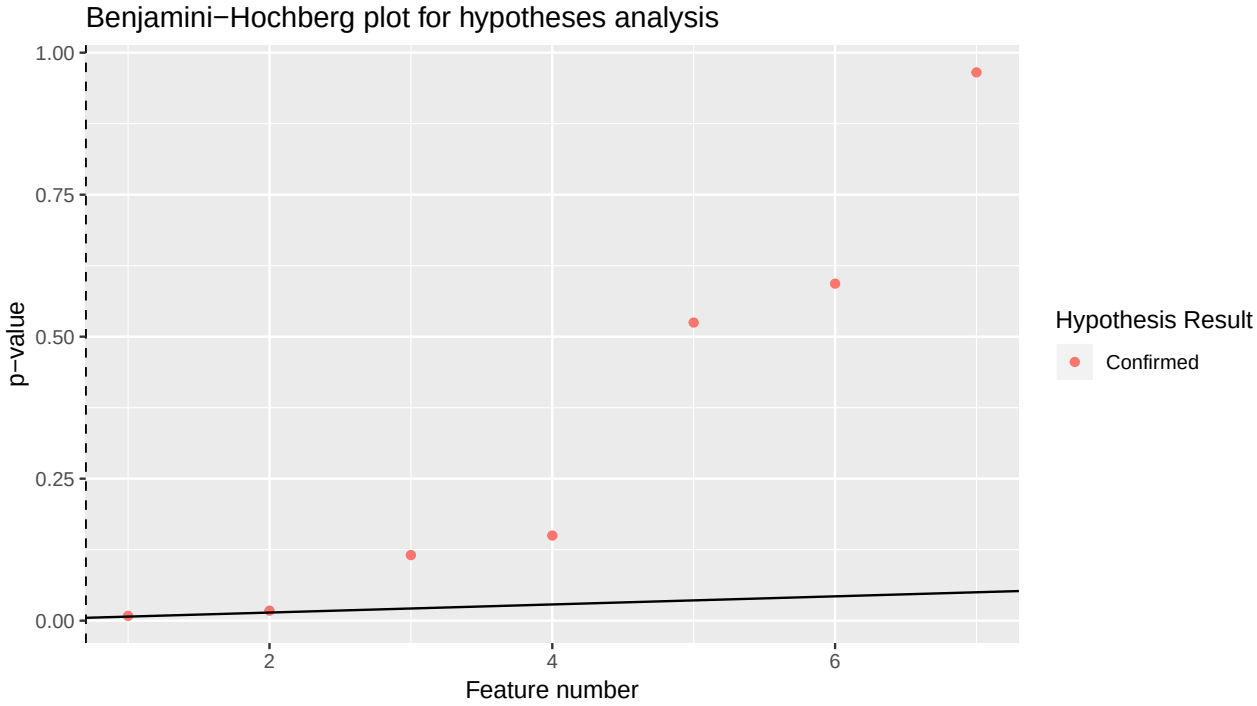
For a Poisson distribution, $\mu = \lambda$.

$$log(\lambda) = \mathbf{w}^T\mathbf{x}$$

The exp(alpha) and the lambda value need not be similar as lambda also depends on the other coefficients and terms in the model. In this particular case, the intercept of the model is very high compared to the contributions from the other terms. Hence, thse values are similar for this case but in general, they need not be similar.

Task 1.3

The Benjamini Hochberg method is carried out with target as Year using data from years 1995, 1996 alone and using paired t-tests. The alpha = 0.05.



2p

We observe that none of the p-values are small enough to fall below the rejection line and hence none of the hypotheses are rejected. The variable Week was not considered as the paired t-test between Week and Year produced a NaN p-value - Week is independent of Year which makes sense.

From the Benjamini Hochberg method, none of the hypothese were rejected. This means that none of the variables are significantly related to the target variable Year.

Why is it in fact not so important to use BH method in this case?

The number of variables are very less (only 8) in this case. The BH method is more useful when the number of variables are very high. For such a small number of variables, the correction to the alpha is not very essential. So, it is not so important to use the BH method in this case.

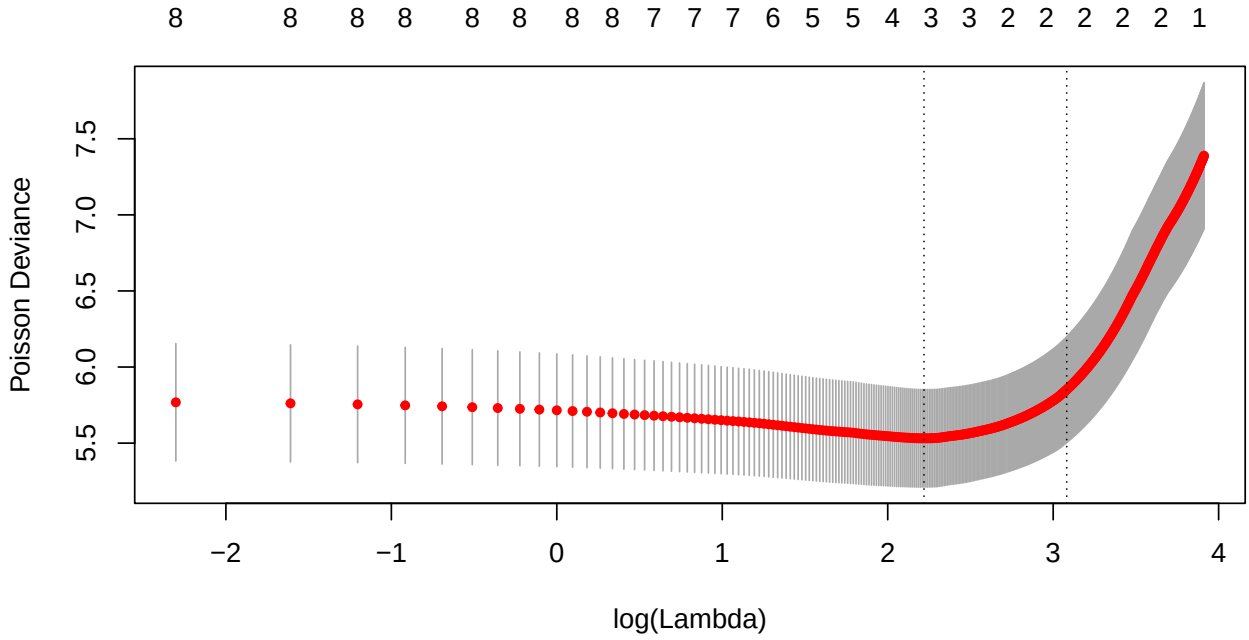
Task 1.4

PCA is carried out with all variables except mortality in the training data. The following table shows the proportion of variance and the cumulative proportion of variance explained by each of the PC components.

var_prop	cum_var_prop	pc_num
0.4302747	0.4302747	1
0.2088848	0.6391595	2
0.1058903	0.7450498	3
0.0947520	0.8398019	4
0.0781603	0.9179621	5
0.0451469	0.9631090	6
0.0323151	0.9954241	7
0.0045759	1.0000000	8

The above table shows the cumulative proportion of variance explained by the variables. We see that 5 variables are sufficient to explain more than 90% of the variance.

Next, we create a LASSO regression model for mortality as a poisson distributed target using all the PC features. The following is the CV plot for the same showing the dependence of CV error (poisson deviance) on $log(\lambda)$.



Does complexity of the model increase when lambda increases?

No, the complexity of the model actually decreases when lambda increases. Higher lambda penalizes the coefficients more and hence leads to the model coefficients becoming more sparse. So, higher lambda leads to lesser selected features and lower complexity.

```
## [1] "Optimal lambda:  9.2"
```

How many features are selected by the LASSO regression?

Coefficients

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept)  7.484957015
## PC1         -0.028907033
## PC2         -0.011559431
## PC3          .
## PC4          0.003196155
## PC5          .
## PC6          .
## PC7          .
## PC8          .
```

3p

From the above coefficients, we see that only 3 features are selected by the LASSO regression. The selected features are PC1, PC2 and PC4.

Probabilistic model for optimal LASSO model

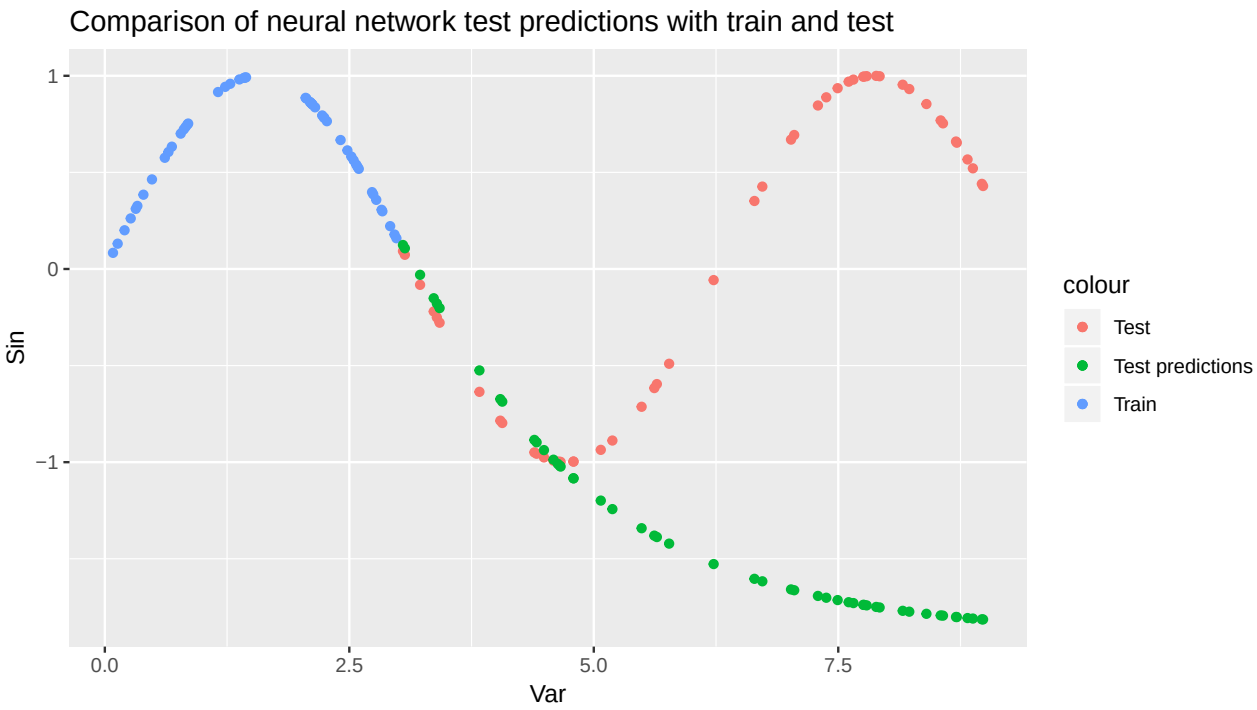
$$Y = Poisson(e^{\mathbf{w}^T\mathbf{x}})$$

$$log(\mu) = \mathbf{w}^T\mathbf{x}$$

$$log(\mu) = 7.484941006 - 0.029007258 * PC1 - 0.011708612 * PC2 + 0.003418258 * PC4$$

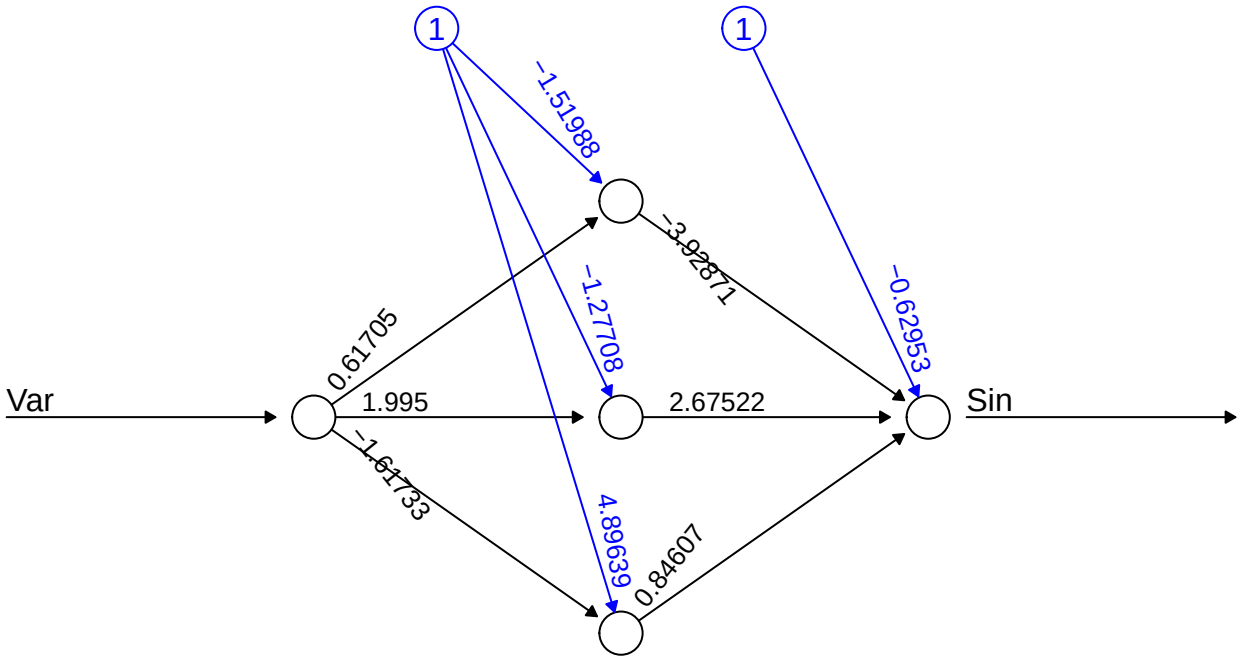
Assignment 2

Neural Network



From the above plot, we can confirm that the result matches the plot provided in the examination.

Neural network plot



If we look at the values of the weights learned by the neural network, we can see that when the input value is

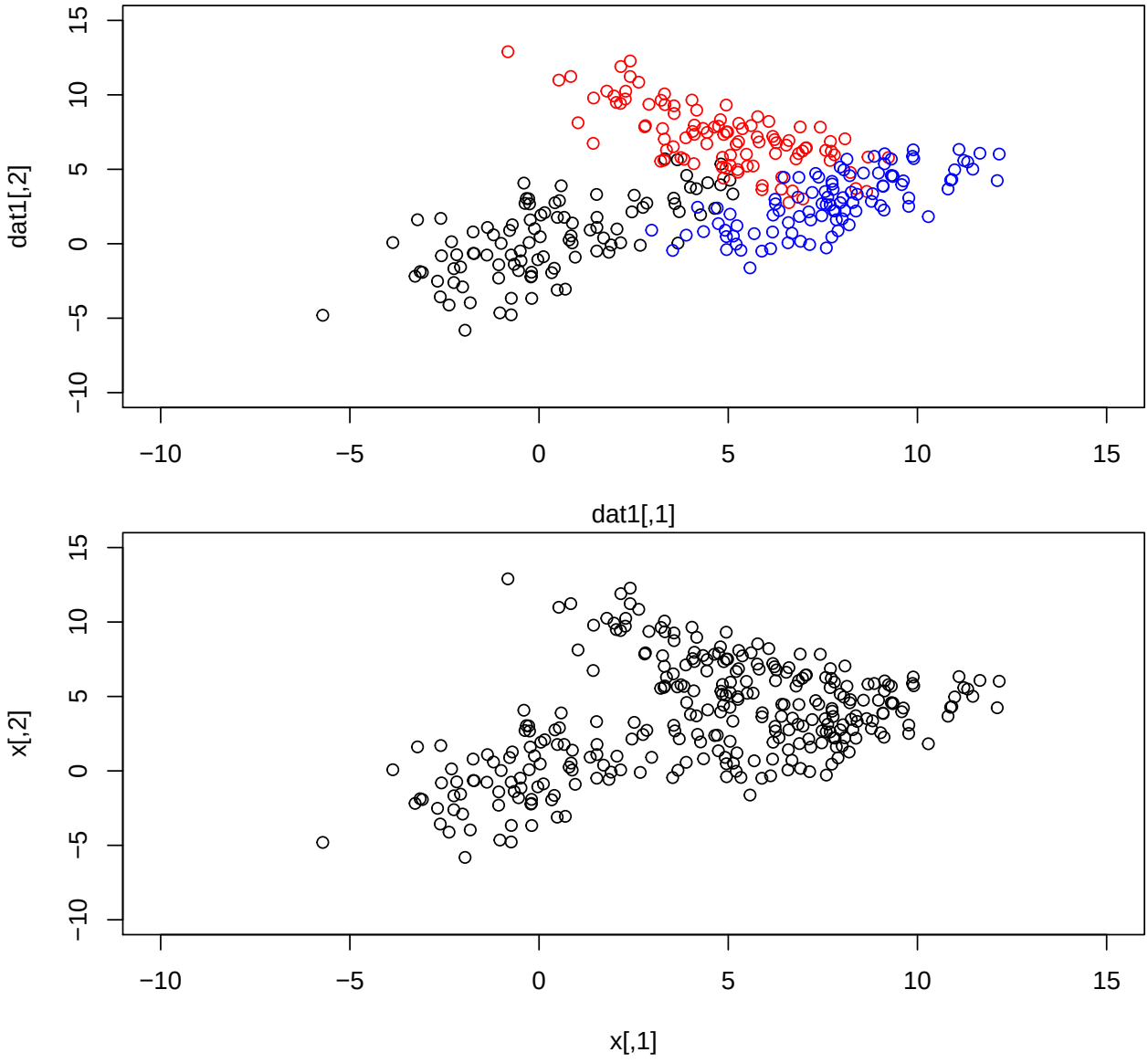
high, the activation function (tanh) result converges to 1 or -1. So, the final output converges to a linear combination of the bias and the weights for the output similar to (b + w1 + w2 - w3). This value is close to 2 and that is why the output converges when the input value increases to high values.

Mixture Models

Plot of original data

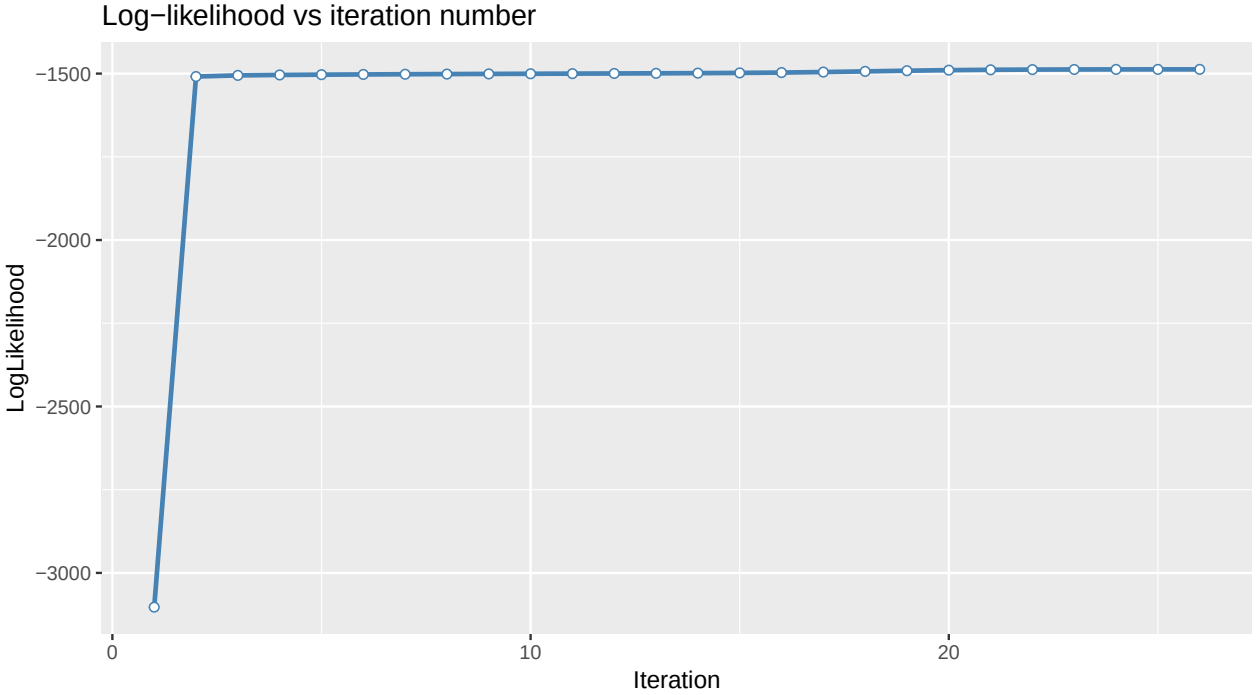
The formula given in the examination were used to compute the ML estimates for the mu, sigma and pi. The log likelihood was computed using:

$$ln(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\pi}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} ln \left\{ \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k) \right\}$$



EM iterations

```
## iteration:  1 log likelihood:  -3102.839199
## iteration:  2 log likelihood:  -1508.843076
## iteration:  3 log likelihood:  -1505.372994
## iteration:  4 log likelihood:  -1504.040734
## iteration:  5 log likelihood:  -1503.19948
## iteration:  6 log likelihood:  -1502.532973
## iteration:  7 log likelihood:  -1501.954378
## iteration:  8 log likelihood:  -1501.434248
## iteration:  9 log likelihood:  -1500.957702
## iteration:  10 log likelihood:  -1500.512032
## iteration:  11 log likelihood:  -1500.082448
## iteration:  12 log likelihood:  -1499.649056
## iteration:  13 log likelihood:  -1499.182016
## iteration:  14 log likelihood:  -1498.632104
## iteration:  15 log likelihood:  -1497.913758
## iteration:  16 log likelihood:  -1496.886605
## iteration:  17 log likelihood:  -1495.387049
## iteration:  18 log likelihood:  -1493.416445
## iteration:  19 log likelihood:  -1491.349148
## iteration:  20 log likelihood:  -1489.687007
## iteration:  21 log likelihood:  -1488.61156
## iteration:  22 log likelihood:  -1487.993819
## iteration:  23 log likelihood:  -1487.65056
## iteration:  24 log likelihood:  -1487.458247
## iteration:  25 log likelihood:  -1487.348396
## iteration:  26 log likelihood:  -1487.284279
## [1] "Change in log likelihood is less than min_change =  0.1 . Ending iterations."
```



From the above plot and the printed values, we can see that the log-likelihood value increases with the number of iterations.

Comparison of mu and sigma with true values

```
## [1] "Original mu values"
```

```
##     [,1] [,2]
## mu1    0    0
## mu2    5    7
## mu3    8    3
```

```
## [1] "Final mu value"
```

```
##               [,1]           [,2]
## [1,] -0.1096711342 -0.02341510203
## [2,]  4.9736527626  6.89649646076
## [3,]  7.5481551652  2.87162986843
```

```
## [1] "\n\nOriginal sigma values"
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    5    3
## [2,]    3    5
##
## , , 2
##
##      [,1] [,2]
## [1,]    5   -3
## [2,]   -3    5
##
## , , 3
##
##      [,1] [,2]
## [1,]    3    2
## [2,]    2    3
```

```
## [1] "\n\nFinal sigma values"
```

```
## , , 1
##
##             [,1]        [,2]
## [1,] 3.940797379 2.706444705
## [2,] 2.706444705 5.796523323
##
## , , 2
##
##              [,1]         [,2]
## [1,]  4.513016518 -3.317680382
## [2,] -3.317680382  5.154162760
##
## , , 3
##
##             [,1]        [,2]
## [1,] 4.638159570 2.828175975
## [2,] 2.828175975 3.861885534
```

From the above values of the original and final mu and sigma values, we can see that the final mu values are very close to the actual mu values. The final sigma values are also close to the original sigma values but there

are some differences - the diagonal elements are equal in the original but they differ slightly in the final sigma values.

Appendix

```
# Global options
knitr::opts_chunk$set(echo = FALSE, error = FALSE, warning = FALSE,
                      message = FALSE, fig.width=8)

# Required libraries
library(readxl) # reading excel
library(dplyr)
library(stringr) # string manipulation

library(kknn) # K nearest neighbors
library(MASS) #
library(glmnet) # GLM
library(tree) # Decision trees
library(e1071) # Naive bayes
library(boot) # Bootstrap
# library(fastICA) # ICA
library(kernlab) # SVM
library(neuralnet) # Neural networks
library(mvtnorm)

library(ggplot2)
library(gridExtra)

library(knitr)
options(kableExtra.latex.load_packages = FALSE)
library(kableExtra)


#-------------------------------------------------------------
# Assignment 1
#-------------------------------------------------------------

inf = read.csv("given_files/Influenza.csv")

lambdas = seq(10, 2910, 100)

log_poisson = function(lambda, y){
  lp = -lambda + y*log(lambda) - sum(log(1:y))
}

calc_llik = function(lambda){
  m_llik = 0

  for(i in 1:nrow(inf)){
    m_llik = m_llik - log_poisson(lambda, inf$Mortality[i])
  }
```

```
  return(m_llik)
}

llik_df = data.frame()

for(lamb in lambdas){
  llik_lb = calc_llik(lamb)
  llik_df = rbind(llik_df, data.frame(lambda = lamb, minus_log_likelihood = llik_lb))
}

# Plot that shows the dependence of minus log-likelihood on lambda
ggplot(llik_df) + geom_line(aes(x=lambda, y=minus_log_likelihood)) +
ylab("Minus Log likelihood") + xlab("Lambda") + ggtitle("Minus Log likelihood vs Lambda")



inf2 = as.data.frame(scale(inf[ , -3]))
inf2$Mortality = inf$Mortality

n = dim(inf2)[1]
set.seed(12345)
id = sample(1:n, floor(n*0.5))
inf2_train = inf2[id,]
inf2_test = inf2[-id,]

# Perform k-fold CV to select lambda for a LASSO model
inf2_cv_model = cv.glmnet(as.matrix(inf2_train[ , -9]), matrix(inf2_train$Mortality),
                          alpha = 1, family = "poisson", lambda = seq(0, 30, 0.01))

plot(inf2_cv_model)

# Find optimal lambda and model
optimal_lambda = inf2_cv_model$lambda.min

print(paste("Optimal lambda: ", optimal_lambda))

optimal_lasso_model = glmnet(as.matrix(inf2_train[ , -9]), matrix(inf2_train$Mortality),
                             alpha = 1, family = "poisson", lambda = optimal_lambda)

pred_trn = predict(optimal_lasso_model, newx = as.matrix(inf2_train[ , -9]), type = "response")
pred_tst = predict(optimal_lasso_model, newx = as.matrix(inf2_test[ , -9]), type = "response")

train_mse = sum((pred_trn - inf2_train$Mortality)^2)/nrow(inf2_train)
test_mse = sum((pred_tst - inf2_test$Mortality)^2)/nrow(inf2_test)

print(paste("Train MSE: ", train_mse))
print(paste("Test MSE: ", test_mse))

print(coef(optimal_lasso_model))

print(paste("alpha = ", coef(optimal_lasso_model)[1]))
print(paste("exp(alpha) = ", exp(coef(optimal_lasso_model)[1])))
```

```r
inf_9596 = inf[inf$Year == 1995 | inf$Year == 1996, ]

# Benjamini Hochberg method

# Function to compute p-value for given column with Conference
get_p_value = function(col){
  form = as.formula(paste(col, "~", "Year"))
  res = t.test(form, data = inf_9596, alternative = "two.sided", paired = TRUE)
  res$p.value
}

# Compute p-values
p_values = sapply(colnames(inf_9596[ , -1]), get_p_value)
p_value_df = data.frame(feature = colnames(inf_9596[, -1]),
                        p_value = p_values)
p_value_df = p_value_df[order(p_value_df$p_value), ]
p_value_df$feature_num = 1:nrow(p_value_df)

# Removing Week because the p.value is NaN
p_value_df = p_value_df[p_value_df$feature!="Week", ]

alpha = 0.05
M = nrow(p_value_df)
L = max(which(p_value_df$p_value < (alpha * p_value_df$feature_num / M)))
p_L = p_value_df[L, "p_value"]

# Set hypotheses results
p_value_df$hypo_res = ifelse(p_value_df$p_value <= p_L, "Rejected", "Confirmed")

ggplot(p_value_df) +
  geom_point(aes(x = feature_num, y = p_value, color = "Confirmed")) +
  geom_abline(slope = alpha/M, intercept = 0) +
  labs(color = "Hypothesis Result") +
  geom_vline(xintercept = L, linetype = "dashed") +
  xlab("Feature number") + ylab("p-value") +
  ggtitle("Benjamini-Hochberg plot for hypotheses analysis")

rejected_features = p_value_df$feature[p_value_df$hypo_res == "Rejected"]


pca_inf = prcomp(inf2_train[,-9])
lambda = pca_inf$sdev^2
var_prop = lambda / sum(lambda)
cum_var_prop = cumsum(var_prop)

var_prop_df = data.frame(var_prop = var_prop, cum_var_prop = cum_var_prop,
                         pc_num = 1:length(var_prop))

kable(var_prop_df)
inf_pc_x = pca_inf$x

infpc_cv_model = cv.glmnet(inf_pc_x, matrix(inf2_train$Mortality),
                           alpha = 1, family = "poisson", lambda = seq(0, 50, 0.1))
```

```r
plot(infpc_cv_model)

# Find optimal lambda and model
optimal_pc_lambda = infpc_cv_model$lambda.min

print(paste("Optimal lambda: ", optimal_pc_lambda))

optimal_pc_lasso_model = glmnet(inf_pc_x, matrix(inf2_train$Mortality),
                                alpha = 1, family = "poisson", lambda = optimal_pc_lambda)

print(coef(optimal_pc_lasso_model))

#-------------------------------------------------------------
# Assignment 2
#-------------------------------------------------------------

#-------------------------------------------------------------
# Neural Network
#-------------------------------------------------------------

set.seed(1234567890)
Var <- runif(50, 0, 3)
tr <- data.frame(Var, Sin=sin(Var))
Var <- runif(50, 3, 9)
te <- data.frame(Var, Sin=sin(Var))

# Random initialization of the weights in the interval [-1, 1]
w_init = runif(10, -1, 1)

# Training neural network
nn = neuralnet(Sin ~ Var, data = tr, hidden = 3, startweights = w_init)

# Predicting values
te_res = neuralnet::compute(nn, te$Var)$net.result

ggplot(tr) + geom_point(aes(x=Var, y=Sin, color="Train")) +
  geom_point(data=te, aes(x=Var, y=Sin, color="Test")) +
  geom_point(data=te, aes(x=Var, y=te_res, color="Test predictions")) +
  ggtitle("Comparison of neural network test predictions with train and test")

plot(x = nn, rep = "best", information = F)

#-------------------------------------------------------------
# Mixture Models
#-------------------------------------------------------------

set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=300 # number of training points
D=2 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
```

```r
# Sampling the training data
mu1<-c(0,0) # component 1
Sigma1 <- matrix(c(5,3,3,5),D,D)
dat1<-rmvnorm(n = 100, mu1, Sigma1)
mu2<-c(5,7) # component 2
Sigma2 <- matrix(c(5,-3,-3,5),D,D)
dat2<-rmvnorm(n = 100, mu2, Sigma2)
mu3<-c(8,3) # component 3
Sigma3 <- matrix(c(3,2,2,3),D,D)
dat3<-rmvnorm(n = 100, mu3, Sigma3)
plot(dat1,xlim=c(-10,15),ylim=c(-10,15))
points(dat2,col="red")
points(dat3,col="blue")
x[1:100,]<-dat1
x[101:200,]<-dat2
x[201:300,]<-dat3
plot(x,xlim=c(-10,15),ylim=c(-10,15))

K = 3 # number of guessed components

z <- matrix(nrow=N, ncol=K) # fractional component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional means
Sigma <- array(dim=c(D,D,K)) # conditional covariances
llik <- vector(length = max_it) # log likelihood of the EM iterations

# Random initialization of the paramters
pi <- runif(K,0,1)
pi <- pi / sum(pi)
for(k in 1:K) {
  mu[k,] <- runif(D,0,5)
  Sigma[ , ,k] <- c(1,0,0,1)
}

# z <- matrix(nrow = N, ncol = K) # fractional component assignments
# pi <- vector(length = K) # mixing coefficients
# mu <- matrix(nrow = K, ncol = D) #conditional distributions
# llik <- vector(length = max_it) # log likelihood of the EM iterations

# # Random initialization of the paramters
# pi <- runif(K, 0.49, 0.51)
# pi <- pi / sum(pi)
# for(k in 1:K) {
#   mu[k,] <- runif(D, 0.49, 0.51)
# }

for(it in 1:max_it) {

  # Sys.sleep(0.5)

  # E-step: Computation of the fractional component assignments
  p_xn = matrix(nrow = nrow(x), ncol = K)
  z = matrix(nrow = nrow(x), ncol = K)
```

```
  for(n in 1:nrow(x)){
    for(k in 1:K){
      p_xn[n, k] = pi[k] * dmvnorm(x[n, ], mu[k, ], Sigma[,,k])
    }
  }
  z = p_xn
  z = z / rowSums(z)

  #Log likelihood computation
  llik[it] = sum(log(rowSums(p_xn)))
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()

  # Stop if the lok likelihood has not changed significantly
  if((it>1) && (abs(llik[it] - llik[it-1]) < min_change)){
    print(paste("Change in log likelihood is less than min_change = ", min_change, ". Ending iterations
    break
  }

  #M-step: ML parameter estimation from the data and fractional component assignments
  pi = colSums(z) / nrow(x)
  mu = (t(z) %*% x) / colSums(z)

  for(k in 1:K){
    nr = matrix(rep(0,4), nrow=2)
    dr = 0
    for(n in 1:nrow(x)){
      dr = dr + z[n,k]
      nr = nr + (x[n,] - mu[k,]) %*% t(x[n,] - mu[k,]) * z[n,k]
    }
    Sigma[,,k] = nr/dr
  }
}

loglik_df = data.frame(Iteration = 1:it, LogLikelihood = llik[1:it])

ggplot(loglik_df) +
  geom_line(aes(x=Iteration, y=LogLikelihood), color = "steelblue", size=1) +
  geom_point(aes(x=Iteration, y=LogLikelihood),
             color = "steelblue", fill="white", shape=21, size=1.7) +
  ggtitle("Log-likelihood vs iteration number")

print("Original mu values")
print(rbind(mu1,mu2,mu3))

print("Final mu value")
print(mu)

act_sigma = array(dim=c(D,D,K))
act_sigma[,,1] = Sigma1
act_sigma[,,2] = Sigma2
act_sigma[,,3] = Sigma3
```

```
print("\n\nOriginal sigma values")
print(act_sigma)

print("\n\nFinal sigma values")
print(Sigma)
```