10702/36702 Statistical Machine Learning, Spring 2008: Homework 3 Solutions

March 24, 2008

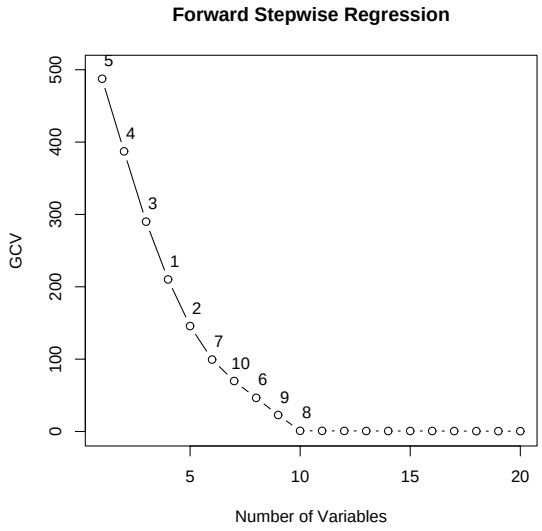
1 [25 points], (Jingrui)

(a) Generate data as follows.

$$\begin{aligned} n &= 100 \\ p &= 1000 \\ X &= matrix(rnorm(n * p), n, p) \\ beta &= c(rep(10, 5), rep(5, 5), rep(0, 990)) \\ y &= X\% * \%beta + rnorm(n) \end{aligned} \tag{1}$$

(b) Write an R function to do forward stepwise regression. Apply it to your dataset and summarize the output. The output should include the cross-validation score for the sequence of models.

★ **SOLUTION:** The first 20 selected features and their corresponding GCV scores are as follows.

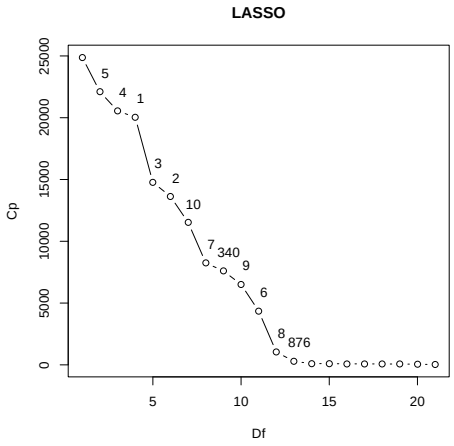


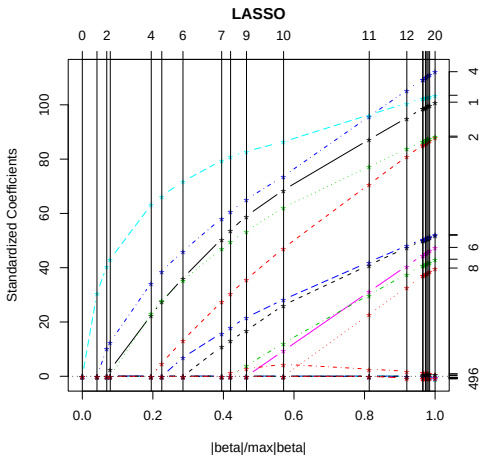
If we train a linear regression model using the 20 features, the coefficients are as follows.

Index of Variables	5	4	3	1	2	7	10	6	9	8
Coefficient	10.22	9.84	10.12	10.21	9.97	5.11	5.12	5.19	4.93	4.79
Index of Variables	403	885	320	400	152	618	292	531	376	392
Coefficient	-0.46	-0.40	-0.23	0.29	-0.29	-0.31	-0.28	-0.22	-0.27	0.22

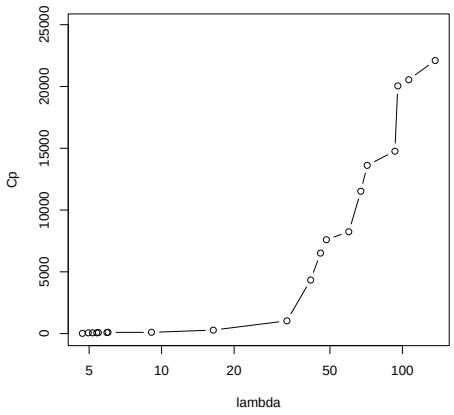
With forward stepwise regression, the first 10 covariates are selected in the first 10 steps, which roughly recovers the true underlying model. After the first 10 steps, the GCV score keeps decreasing, which is due to the large variance of the noise. However, the decrease in the score is not as significant as in the first 10 steps.

(c) Run the lasso on these data and compare to the results from stepwise.



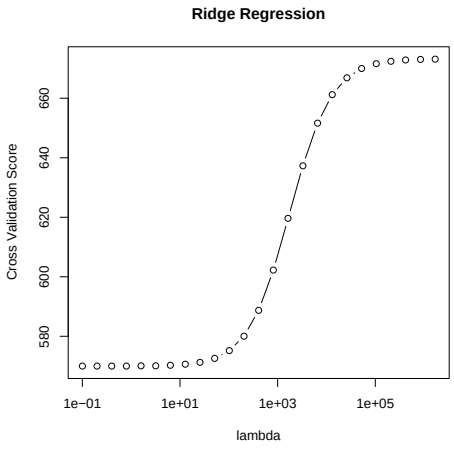


★ **SOLUTION:**



Based on the above figures, the first 10 covariates are among the first few features selected by Lasso, and the Cp scores stops decreasing significantly after the 10 covariates are included in the model. Compared with forward stepwise regression, with Lasso, two noise features are also included in the model.

(d) Write a function to do ridge regression. Run your code on these data and summarize the results.



★ **SOLUTION:** Based on the above figure, we train the ridge regression model with $\lambda = 0.1$, and the coefficients of the first 10 covariates are as follows.

Index of Variables	1	2	3	4	5	6	7	8	9	10
Coefficient	0.94	0.62	0.88	1.09	1.37	0.39	0.84	0.32	0.51	0.70
Index of Variables	11	12	13	14	15	16	17	18	19	20
Coefficient	-0.21	0.09	-0.01	0.03	0.53	0.09	-0.17	0.10	-0.12	-0.03

Therefore, with ridge regression, the coefficients of the noise features may be very close to 0, but not exactly 0. So ridge regression can not perform feature selection.

2 [25 points], (Robin)

In this problem you will fit a logistic regression model to the UCI Pima Indians diabetes database. The data, and a description of the data, can be downloaded from

`http://www.ics.uci.edu/~mlearn/databases/pima-indians-diabetes.`

It is a binary classification problem with 768 instances having eight features each.

(a) Fit a maximum likelihood logistic regression model using Newton's method (iteratively reweighted least squares). Summarize your findings.

★ SOLUTION:

```
# load data
data = read.table("pima-indians-diabetes.data",sep=",");

# split data into input and output
x = as.matrix(data[,1:8]);
y = as.matrix(data[,9]);

# run logistic regression
out = glm(y ~ x, family = "binomial")
summary.glm(out);

############################
output of summary.glm(out)
############################
Call:
glm(formula = y ~ x, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5566  -0.7274  -0.4159   0.7267   2.9297

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -8.4046964  0.7166359 -11.728  < 2e-16 ***
xV1          0.1231823  0.0320776   3.840 0.000123 ***
xV2          0.0351637  0.0037087   9.481  < 2e-16 ***
xV3         -0.0132955  0.0052336  -2.540 0.011072 *
xV4          0.0006190  0.0068994   0.090 0.928515
xV5         -0.0011917  0.0009012  -1.322 0.186065
xV6          0.0897010  0.0150876   5.945 2.76e-09 ***
xV7          0.9451797  0.2991475   3.160 0.001580 **
xV8          0.0148690  0.0093348   1.593 0.111192
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 723.45  on 759  degrees of freedom
AIC: 741.45

Number of Fisher Scoring iterations: 5
```

The output shows tht only a few features: x_1, x_2, x_6, x_7 are important for classification.

(b) Now use sparse logistic regression. Use corrdinate-wise descent. Plot the estimates as a function of λ. Also, plot the AIC score as a function of λ.

★ SOLUTION:

```
# sparse logistic regression
# assumed that the first column of x is all 1's
sparse <- function(x,y,lambda){
  # Initialization
  n <- length(y);
  numFeats <- dim(x)[2];
  beta <- 1/numFeats/colMeans(x);
  epsilon <- 1;

  while ( epsilon > 1e-6 ){
    betaOld <- beta;
    for ( j in 1:numFeats ){
      betaTmp <- beta;
      betaTmp[j] <- 0;
      pTilda <- as.vector(1 - 1/(1+exp(x%*%betaTmp)));

if ( abs( t(y-pTilda)%*%x[,j] ) < lambda ){
        beta[j] = 0;
      }
      else {
        delta <- 1;
        while ( delta > 1e-3 ){
    p <- as.vector(1 - 1/(1+exp(x%*%beta)));
          delta <- (t(y-p)%*%x[,j]-lambda*sign(beta[j]))/(t(p*(1-p))%*%(x[,j]^2));
          beta[j] <- beta[j]+delta;
        }
      }
    } # j loop terminates

    epsilon <- sum(abs(beta-betaOld));
  } # while (epsilon < 1e-6) loop end

  ## compute log-likelihood
  ll <- y %*% (x%*%beta) - sum(log(1+exp(x%*%beta)));
  S <- sum(beta!=0);

  AIC1 <- ll - S;
  AIC2 <- -2*ll + 2*S;

  return (list(beta=beta,AIC1=AIC1,AIC2=AIC2));
} # sparse function end

# main program
# load data
data = read.table("pima-indians-diabetes.data",sep=",");
```

```
# split data into input and output
x = as.matrix(data[,1:8]);
y = as.vector(data[,9]);

# Add column for intercept beta_0
x = cbind(rep(1,length(y)),x);

size <- 500;
maxLambda <- 250;
lambda <- seq(0,maxLambda,length=size);
beta <- matrix(rep(0,size*dim(x)[2]),nrow=dim(x)[2]);
aic <- matrix(rep(0,size*2),nrow=2);

for ( i in 1:size){
  out <- sparse(x,y,lambda[i]);
  beta[,i] <- out$beta;
  aic[1,i] <- out$AIC1;
  aic[2,i] <- out$AIC2;
  cat(i,"\n");
}

# plotting results
par(mfrow=c(1,3));

# plotting the estimates
plot(lambda,beta[1,],type='l',col=10,ylab="Beta")
for ( i in 2:9 )
  lines(lambda,beta[i,],type='l',col=10*i);

#plotting AIC
plot(lambda,aic[1,],type='l',ylab="AIC=ll-|S|");
plot(lambda,aic[2,],type='l',ylab="AIC=-2*ll+2*|S|");
```

The figure shows that β values reduce to zeroes for higher values of λ. You can either maximize $AIC = ll - |S|$ or minimize $AIC = -2(ll + |S|)$. Either case the figure shows the optimal value of λ is 0.

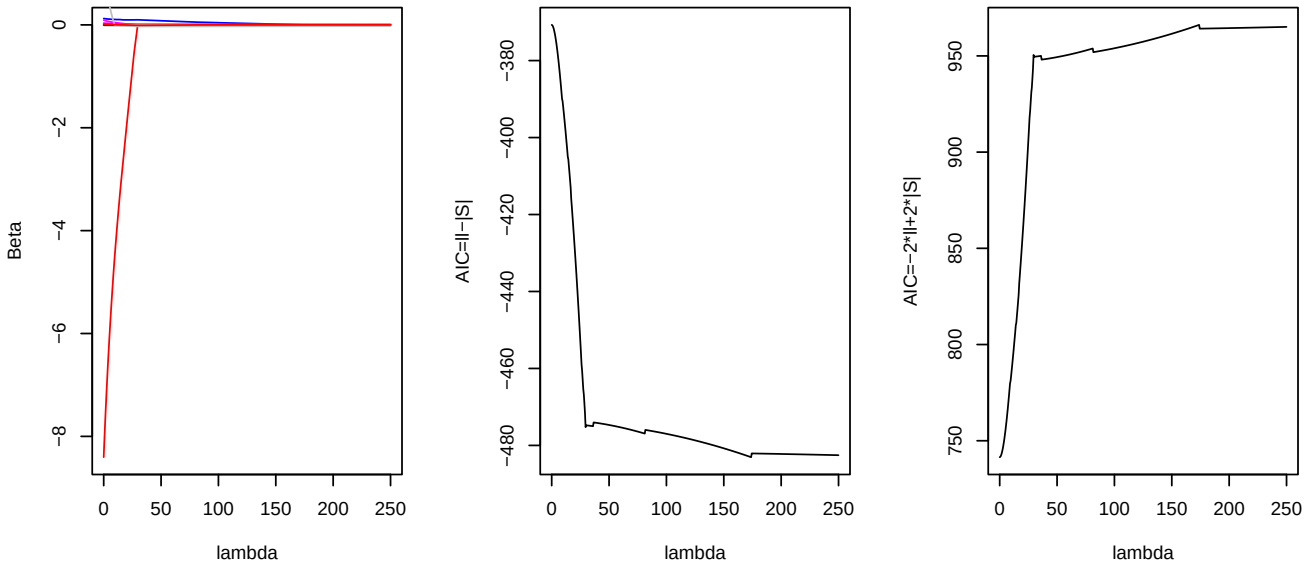
3 [25 points], (Jingrui)

Let $x = (x_1, \ldots, x_d)^T \in \mathbb{R}^d$ and let

$$g(x) = \|x\|_p = (\sum_{j=1}^{d} |x_j|^p)^{1/p}$$

where $0 < p < 1$. Find the subgradient of g.

★ **SOLUTION:** Subgradient/subderivative is only defined for convex functions. As we saw in the last homework, when $0 < p < 1$, $g(x)$ is not convex, so it does not have subgradient. When $p = 1$, $g(x) = \sum_{j=1}^{d} |x_j|$ is a convex function. Its subdifferential at $x_j = 0$ is $[-1, 1]$; its subdifferential at $x_j < 0$ is -1; and its subdifferential at $x_j > 0$ is 1.



4 [25 points], (Robin)

(a) Write an R program to fit a mixture of multivariate Normals using the EM algorithm (with the number of components k known).

★ **SOLUTION:**

```
# mixture of multi-variate norms using EM algo.
mvg <- function(X,mu,Sigma,i,j)
{
  d <- dim(X)[2];
  p <- 1/((2*pi)^(d/2))/(det(Sigma)^(1/2))*exp(-0.5*t(X-mu)%*%solve(Sigma)%*%(X-mu));
  return(p);
}

EMmmn <- function(X,k){
  n <- dim(X)[1];
  d <- dim(X)[2];

  m <- rep(0,d);
  for ( j in 1:d )
    m[j] <- mean(X[,j]);

  # initialize parameters
  w <- matrix(rep(0,n*k),nrow = n);
  mu <- matrix(runif(d*k),nrow=k);
  p <- matrix(rep(1/k,k),nrow=k);

  Sinit <- 1/n*t(X-m)%*%(X-m);
```

```
S <- matrix(rep(0,d*d*k),nrow=k);

for ( j in 1:k ) S[j,] <- as.vector(Sinit);

tol <- 1;
# repeat until convergence
while ( tol > 1e-03*k*d )
{
  ll <- 0;
  oldmu <- mu;
  # compute weights
  for ( j in 1:k )
  {
    Sj <- matrix(S[j,],nrow=d);
    for ( i in 1:n )
    {
      prob <- mvg(as.matrix(X[i,]),as.matrix(mu[j,]),Sj);
      w[i,j] <- p[j]*prob;
      ll <- ll + log(prob);
    }
  }

  # normalize weights
  for ( i in 1:n )
  {
    sumWi <- sum(w[i,]);
    w[i,] <- w[i,]/sumWi;
  }

  # update parameters
  for ( j in 1:k )
  {
    sumWj <- sum(w[,j]);
    p[j] <- 1/n*sumWj;

    for ( i in 1:d )
      mu[j,i] <- sum(X[,i]*w[,j])/sumWj;

    S[j,] <- as.vector( (t(X-mu[j,])%*%((X-mu[j,])*w[,j])/sumWj) );
  }

  tol <- sum(abs(mu-oldmu));
  % cat("tol: ",tol,"\n");
}

# compute log-likelihood
ll <- 0;
for ( j in 1:k )
{
  Sj <- matrix(S[j,],nrow=d);
  for ( i in 1:n )
    ll <- ll + w[i,j]*log(mvg(as.matrix(X[i,]),as.matrix(mu[j,]),Sj));
}
```

```
  s <- k+d*k+k*d*d;
  aic <- ll - s;
  bic <- ll - s/2*log(n);
  return(list(prob=p,mu=mu,var=S,aic=aic,bic=bic));
}
```

(b) Simulate 1000 observations from the model

$$\frac{1}{5}N(0, I) + \frac{4}{5}N(\mu, I)$$

where $\mu = (3, 3, 3, 3, 3)$ and I is the 5 by 5 identity matrix. Use your program to find the MLE.

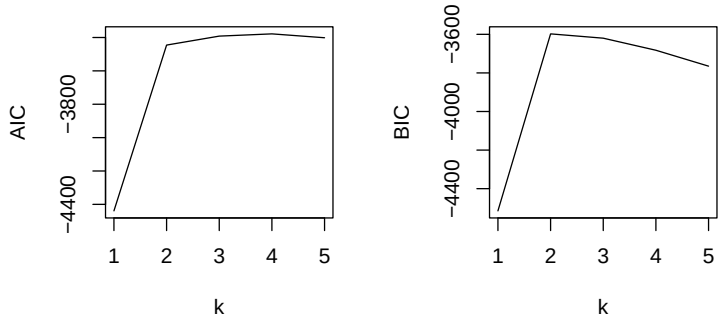
★ SOLUTION:

```
# load library
library(MASS);

d <- 5; # number of dimensions
n <- 1000; # number of points
S <- diag(d); # identity matrix

# generate data
X <- matrix(rep(0,n*d),nrow=n);

for ( i in 1:n ) {
  if ( runif(1,0,1) < 0.2 ) {
    X[i,] <- mvrnorm(1,mu=rep(0,d),Sigma=S);
  }
  else { X[i,] <- mvrnorm(1,mu=rep(3,d),Sigma=S); }
}

res <- EMmmn(X,2);

cat("cluster probs\n");
res$prob
cat("cluster means\n");
res$mu[1,]
res$mu[2,]
cat("cluster vars\n");
matrix(res$var[1,],nrow=d)
matrix(res$var[2,],nrow=d)

##############
output of code
##############
cluster probs
> res$prob
          [,1]
[1,] 0.8061191
[2,] 0.1938809
> cat("cluster means\n");
cluster means
> res$mu[1,]
```

```
[1] 2.970932 3.049995 3.012737 2.900860 2.986491
> res$mu[2,]
[1] 0.10066126 0.02896716 0.04202755 0.04309818 0.06456465
> cat("cluster vars\n");
cluster vars
> matrix(res$var[1,],nrow=d)
            [,1]         [,2]         [,3]         [,4]         [,5]
[1,]  0.94865362 -0.010098981  0.048339147 -0.036357260  0.093404195
[2,] -0.01009898  1.026635129 -0.005396046 -0.054604952 -0.015011003
[3,]  0.04833915 -0.005396046  0.978004876  0.002056654  0.007193553
[4,] -0.03635726 -0.054604952  0.002056654  1.034921146 -0.035570629
[5,]  0.09340420 -0.015011003  0.007193553 -0.035570629  1.007609017
> matrix(res$var[2,],nrow=d)
            [,1]        [,2]         [,3]         [,4]        [,5]
[1,]  0.85527994  0.05907938  0.022014719 -0.040955944 -0.01855187
[2,]  0.05907938  1.08175250 -0.062179874  0.101668376  0.01957809
[3,]  0.02201472 -0.06217987  1.061560303 -0.004966484  0.04057524
[4,] -0.04095594  0.10166838 -0.004966484  0.799421239  0.05533264
[5,] -0.01855187  0.01957809  0.040575241  0.055332636  1.01671623
```

(c) Now repeat but take k as unknown. Use AIC and BIC to choose k.

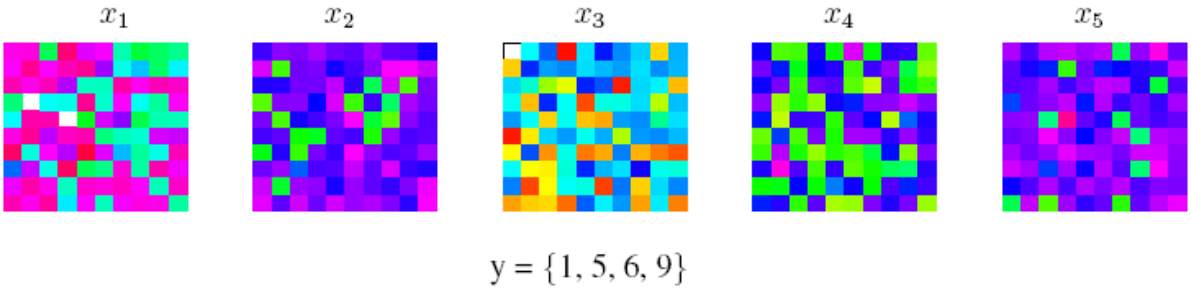
```
AIC = rep(0,5);
BIC = rep(0,5);

# Run EM for multiple k
for ( k in 1:5 )
{
  res <- EMmmn(X,k);
  AIC[k] <- res$aic;
  BIC[k] <- res$bic;
  cat(k,"\n");
}

#plotting results
par(mfrow=c(1,2));
plot(AIC,xlab='k',ylab='AIC',type='l');
plot(BIC,xlab='k',ylab='BIC',type='l');
```

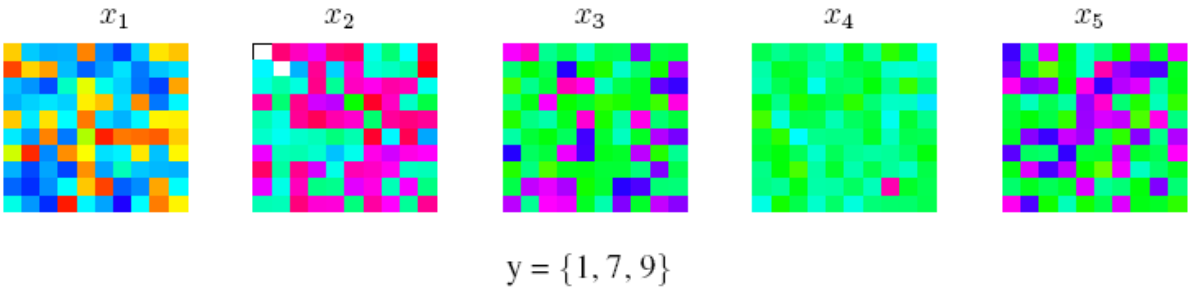


Using the graphs of AIC abd BIC we notice that optimal k using AIC and BIC are 4 and 2 respectively. Hence BIC is a better criteria to estimate the number of mixtures.

5 [25 points], (Jingrui)

In this problem you will use mixture models for a classification task. The training data consist of 100 examples x_i, y_i, where $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$ and each $x_{ij} \in \mathbb{R}^{100}$ is a 100-dimensional vector, and $y_i = \{c_{ij}\}$ is an unordered set of up to five labels, with each $c_{ij} \in \{1, 2, \ldots, 10\}$. Two examples are shown below (check them out in color in the electronic version):





In the first example, there are four classes $y = \{1, 5, 6, 9\}$. Each of the five x_i was generated from exactly one of these four classes; thus, one of the four classes must have generated two of the x_i. Considering the alignment of each of the five images with one of the classes as a latent variable $z = (z_1, z_2, z_3, z_4, z_5)$, a possible alignment is $z = (9, 5, 9, 1, 6)$ where class 9 generates x_1, class 5 generates x_2, class 9 generates x_3, and so on. The test data consist of 100 unlabeled examples x_i. The task is to label each of the test examples with an *ordered* set of labels $y_i = (c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5})$ where c_{ij} is the predicted class of x_{ij}.

You may use any method and model you choose, but it should make use of mixture models in some way. Give a detailed explanation of your approach and methodology. The training data and test data are on the course website.

Describe your method clearly and report a summary of how well you do on the test data.

★ **SOLUTION:** Let the hidden variables be z_{ij}^k. $z_{ij}^k = 1$ iff the j^{th} image of the i^{th} example is from the k^{th} component, and $z_{ij}^k = 0$ otherwise. The parameters in the mixture model include: the priors of the components $p_1, \ldots, p_{10}$, the mean vector of the components $\mu_1, \ldots, \mu_{10}$, and the covariance matrices of the components $\Sigma_1, \ldots, \Sigma_{10}$. For the sake of simplicity, assume that the features are independent given the component. In this case, the covariance matrices are diagonal matrices, and we only need to estimate the diagonal element σ_{kd}^2, $k \in \{1, \ldots, 10\}, d \in \{1, \ldots, 100\}$, which is the variance of j^{th} feature in the i^{th} component.

The EM algorithm works as follows.

1. E-step: if $k > 0$ and $k \in y_i$, then $w_{ij}^k = \mathbb{P}(z_{ij}^k = 1 | x_1, \ldots, x_n) = \frac{p_k \phi(x_{ij}^k; \mu_k, \Sigma_k)}{\sum_{l \in y_i, l > 0} p_l \phi(x_{ij}^l; \mu_l, \Sigma_l)}$; otherwise, $w_{ij}^k = 0$.

2. M-step:

$$p_k = \frac{1}{5n} \sum_{i=1}^{n} \sum_{j=1}^{5} w_{ij}^k$$

$$\mu_k = \frac{\sum_{i=1}^{n} \sum_{j=1}^{5} x_{ij} w_{ij}^k}{\sum_{i=1}^{n} \sum_{j=1}^{5} w_{ij}^k}$$

$$\sigma_{kd}^2 = \frac{\sum_{i=1}^{n} \sum_{j=1}^{5} (x_{ij}^d - \mu_k^d)^2 w_{ij}^k}{\sum_{i=1}^{n} \sum_{j=1}^{5} w_{ij}^k}$$

where μ_k^d is the d^{th} component of μ_k, and x_{ij}^d is the d^{th} component of x_{ij}.

The test error of the above model is 35.6%.