ORF 350: Assignment 5

David Fan

4/28/2017

Collaborator: Brandon Tan

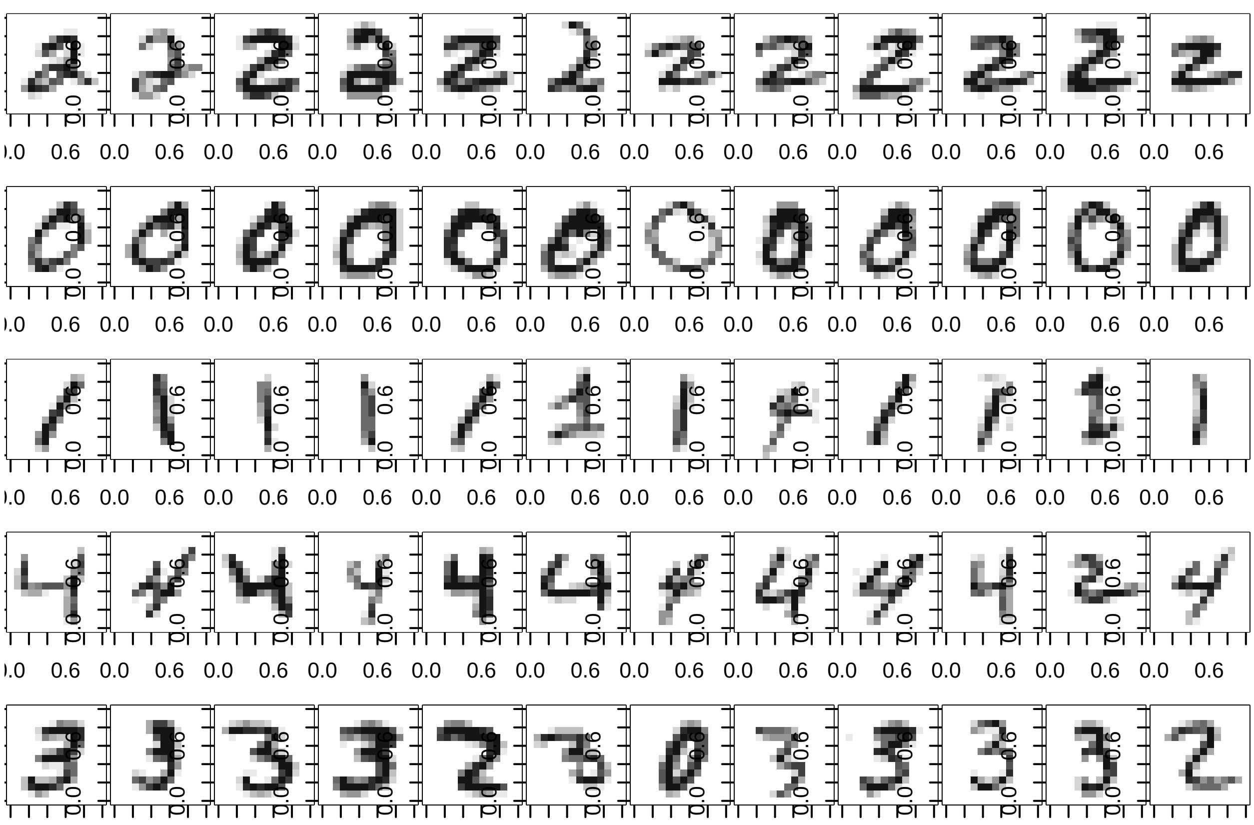
Late Days Used: 3 (handed in Monday before 5 PM)

Question 1: Clustering Algorithms (30 points)

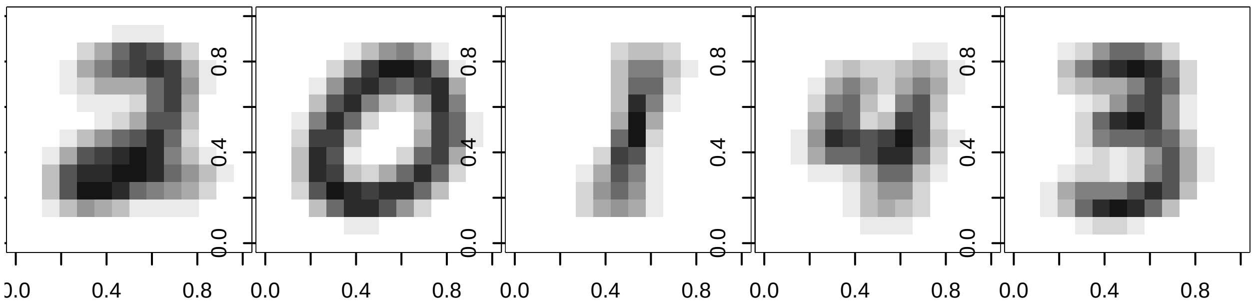
```
# Preprocessing of MNIST data done in pre-processing.R
setwd("/Users/dfan/Dropbox/School/Sophomore Year/Spring 2017/ORF 350/Assignments/HW5")
load_image_file <- function(filename) {
    ret = list()
    f = file(filename, "rb")
    readBin(f, "integer", n = 1, size = 4, endian = "big")
    ret$n = readBin(f, "integer", n = 1, size = 4, endian = "big")
    nrow = readBin(f, "integer", n = 1, size = 4, endian = "big")
    ncol = readBin(f, "integer", n = 1, size = 4, endian = "big")
    x = readBin(f, "integer", n = ret$n * nrow * ncol, size = 1,
        signed = F)
    ret$x = matrix(x, ncol = nrow * ncol, byrow = T)
    close(f)
    ret
}
load_label_file <- function(filename) {
    f = file(filename, "rb")
    readBin(f, "integer", n = 1, size = 4, endian = "big")
    n = readBin(f, "integer", n = 1, size = 4, endian = "big")
    y = readBin(f, "integer", n = n, size = 1, signed = F)
    close(f)
    y
}
train <- load_image_file("train-images-idx3-ubyte")
test <- load_image_file("t10k-images-idx3-ubyte")

train$y <- load_label_file("train-labels-idx1-ubyte")
test$y <- load_label_file("t10k-labels-idx1-ubyte")
train.images = train$x[which(train$y == 0 | train$y == 1 | train$y ==
    2 | train$y == 3 | train$y == 4), ]
train.labels = train$y[which(train$y == 0 | train$y == 1 | train$y ==
    2 | train$y == 3 | train$y == 4)]
train.num = length(which(train$y == 0 | train$y == 1 | train$y ==
    2 | train$y == 3 | train$y == 4))
# created when running preprocessing script. Loads
# compress.train.images object
load("compress.train.images.RData")

# 1.1 L2 norm squared
L2_sqr <- function(x, y) {
```

```r
    return(sum((y - x)^2))
}
assign_cluster <- function(x, centers) {
    cluster_num <- 1
    min_dist <- L2_sqr(x, centers[1, ])
    for (i in 2:nrow(centers)) {
        curr_dist <- L2_sqr(x, centers[i, ])
        if (curr_dist < min_dist) {
            cluster_num <- i
            min_dist <- curr_dist
        }
    }
    return(cluster_num)
}

# k is the number of clusters. Here we do 3 random
# initializations data is a n by m matrix where each row of m
# pixels represents an image Value of each pixel = grayscale
kmeans <- function(data, k) {
    objective_func_values <- rep(0, 3)  # pick initialization that minimizes obj. func
    best_kmeans <- NULL
    min_obj <- Inf

    for (n in 1:3) {
        # random initialization of k cluster centers
        center_idx <- floor(runif(k, min = 1, max = nrow(data) +
            1))
        centers <- data[center_idx, ]
        labels <- rep(0, nrow(data))

        # until convergence
        converge <- FALSE
        while (!converge) {
            converge <- TRUE
            # assignment to nearest cluster by min L2 norm squared
            for (i in 1:nrow(data)) {
                new_label <- assign_cluster(data[i, ], centers)  # numbered from 1 to k
                if (labels[i] != new_label) {
                  converge <- FALSE
                  labels[i] <- new_label
                }
            }
            # recalculating cluster centers
            for (i in 1:k) {
                centers[i, ] <- colMeans(data[which(labels ==
                  i), ])
            }
        }
        curr_kmeans <- list(labels = labels, centers = centers)
        # calculate objective function value for this initialization
        for (i in 1:nrow(data)) {
            objective_func_values[n] <- objective_func_values[n] +
                L2_sqr(data[i, ], centers[labels[i], ])
```

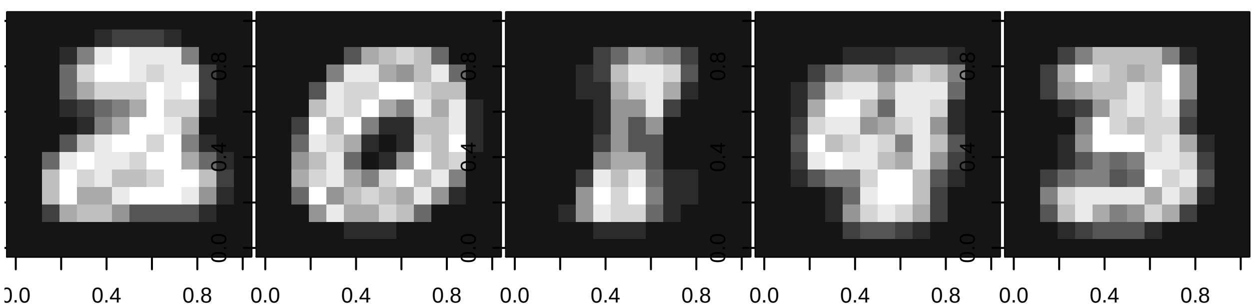
```
        }
        if (objective_func_values[n] < min_obj) {
            best_kmeans <- curr_kmeans
            min_obj <- objective_func_values[n]
        }
    }
    return(best_kmeans)
}
clusters <- kmeans(compress.train.images, 5)

# Plot 1
plotTable <- function(numCol, vec.labels, mat.images) {
    vec.uniq = sort(unique(vec.labels))
    par(mfrow = c(length(vec.uniq), numCol), pty = "s", mar = c(0.1,
        0.1, 0.1, 0.1))
    for (i in 1:length(vec.uniq)) {
        tmpidx = which(vec.labels == i)
        for (j in 1:numCol) {
            show_digitsmall(mat.images[tmpidx[j], ], asp = TRUE)
        }
    }
}
show_digitsmall <- function(arr196, col = gray(12:1/12), ...) {
    image(matrix(arr196, nrow = 14)[, 14:1], col = col, ...)
}
plotTable(12, clusters$labels, compress.train.images)
```



```r
# Plot 2
par(mfrow = c(1, 5))
for (i in 1:5) {
    show_digitsmall(clusters$centers[i, ])
}
```




```r
# Plot 3
for (i in 1:5) {
    curr_cluster <- compress.train.images[which(clusters$labels ==
        i), ]
    var_mat <- apply(curr_cluster, 2, var)
    var_mat <- rescale(var_mat, c(0, 255))
    show_digitsmall(255 - var_mat)
}
```




```r
# Table 1
table <- matrix(0, 5, 5)
for (i in 1:5) {
    ithcluster <- which(clusters$labels == i)
    for (j in 0:4) {
        count <- 0
        for (k in 1:length(ithcluster)) {
            if (train.labels[ithcluster[k]] == j)
                count = count + 1
        }
        table[i, j + 1] = count
    }
}
error <- rep(0, 5)
for (i in 1:5) {
    error[i] <- 1 - max(table[i, ])/sum(table[i, ])
}
print(table)
```

```
##      [,1] [,2] [,3] [,4] [,5]
```

```
## [1,]  220   82 4402  427   36
## [2,] 5225    0   70   37    9
## [3,]   23 6608  667  392  235
## [4,]  109   14  327  146 5559
## [5,]  346   38  492 5129    3
```

```r
paste("Cluster 1 represents digit", which.max(table[1, ]) - 1)
```

```
## [1] "Cluster 1 represents digit 2"
```

```r
paste("Cluster 2 represents digit", which.max(table[2, ]) - 1)
```

```
## [1] "Cluster 2 represents digit 0"
```

```r
paste("Cluster 3 represents digit", which.max(table[3, ]) - 1)
```

```
## [1] "Cluster 3 represents digit 1"
```

```r
paste("Cluster 4 represents digit", which.max(table[4, ]) - 1)
```

```
## [1] "Cluster 4 represents digit 4"
```

```r
paste("Cluster 5 represents digit", which.max(table[5, ]) - 1)
```

```
## [1] "Cluster 5 represents digit 3"
```

```r
indices <- c(which.max(table[1, ]), which.max(table[2, ]), which.max(table[3,
    ]), which.max(table[4, ]), which.max(table[5, ]))
print(error)
```

```
## [1] 0.14805496 0.02171878 0.16618297 0.09683184 0.14630493
```

```r
paste("Digit 1 error rate:", error[which(indices == 1)])
```

```
## [1] "Digit 1 error rate: 0.0217187792548212"
```

```r
paste("Digit 2 error rate:", error[which(indices == 2)])
```

```
## [1] "Digit 2 error rate: 0.166182965299685"
```

```r
paste("Digit 3 error rate:", error[which(indices == 3)])
```

```
## [1] "Digit 3 error rate: 0.148054964195858"
```

```r
paste("Digit 4 error rate:", error[which(indices == 4)])
```

```
## [1] "Digit 4 error rate: 0.146304926764314"
```

```r
paste("Digit 5 error rate:", error[which(indices == 5)])
```

```
## [1] "Digit 5 error rate: 0.0968318440292445"
```

Question 1.2: Expectation-Maximization Algorithm

a) $P(Z_i = j) = \eta_j$

b)

$$\begin{aligned}
P(Z_i = j|x_i) &= \frac{P(x_i|Z_i = j)P(Z_i = j)}{P(x_i)} \\
&= \frac{P(x_i|Z_i = j)P(Z_i = j)}{\sum\limits_{l=1}^{k} P(x_i|Z_i = l)P(Z_i = l)} \\
&= \frac{\frac{\eta_j}{(2\pi)^{d/2}|\sum_j|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i - \mu_j)^T \sum_j^{-1}(x_i - \mu_j))}{\sum\limits_{l=1}^{k} \frac{\eta_l}{(2\pi)^{d/2}|\sum_l|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i - \mu_l)^T \sum_l^{-1}(x_i - \mu_l))} \\
&= \frac{\frac{\eta_j}{(|\sum_j|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i - \mu_j)^T \sum_j^{-1}(x_i - \mu_j))}{\sum\limits_{l=1}^{k} \frac{\eta_l}{|\sum_l|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i - \mu_l)^T \sum_l^{-1}(x_i - \mu_l))}
\end{aligned}$$

c) We want to show that $l(\theta) = \sum\limits_{i=1}^{n} log[\sum\limits_{j=1}^{k} \gamma_{ij} \frac{p_\theta(x_i, Z_i=j)}{\gamma_{ij}}] \geq \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{k} \gamma_{ij} log[\frac{p_\theta(x_i, Z_i=j)}{\gamma_{ij}}]$

The summation $\sum\limits_{j=1}^{k}$ is only over the possible values of Z_i, so it's basically taking an expectation over the values of Z_i where γ_{ij} is the conditional density. Let $\gamma_{ij} = P(Z_i = j|x_i)$. We can then write the log-likelihood as: $l(\theta) = \sum\limits_{i=1}^{n} log E_z[f(Z_i)|x_i]$ where $f(Z_i) = \frac{p_\theta(x_i, Z_i=j)}{\gamma_{ij}}$.

Applying Jensen's inequality, we are done:

$\geq \sum\limits_{i=1}^{n} E_z[log f(Z_i)|x_i] = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{k} \gamma_{ij} log(\frac{p_\theta(x_i, Z_i=j)}{\gamma_{ij}})$

d) Prove $l(\theta^{old}) = F(\gamma, \theta^{old})$ when $\gamma_{ij} = p_{\theta^{old}}(Z_i = j|x_i)$

$$\begin{aligned}
F(\gamma, \theta^{old}) &= \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} log[\frac{p_{\theta^{old}}(x_i, Z_i = j)}{\gamma_{ij}}] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\theta^{old}}(Z_i = j|x_i) log[\frac{p_{\theta^{old}}(x_i, Z_i = j)}{p_{\theta^{old}}(Z_i = j|x_i)}] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\theta^{old}}(Z_i = j|x_i) log[\sum_{l=1}^{k} p_{\theta^{old}}(x_i, Z_i = l)] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\theta^{old}}(Z_i = j|x_i) log[p_{\theta^{old}}(x_i)] \\
&= \sum_{i=1}^{n} log[p_{\theta^{old}}(x_i)] \sum_{j=1}^{k} p_{\theta^{old}}(Z_i = j|x_i) \\
&= \sum_{i=1}^{n} log[p_{\theta^{old}}(x_i)]
\end{aligned}$$

Notice that:

$$
\begin{aligned}
l(\theta^{old}) &= \sum_{i=1}^{n} log[\sum_{j=1}^{k} \gamma_{ij} \frac{p_{\theta^{old}}(Z_i = j, x_i)}{\gamma_{ij}}] \\
&= \sum_{i=1}^{n} log[\sum_{j=1}^{k} p_{\theta^{old}}(x_i, Z_i = j)] \\
&= \sum_{i=1}^{n} log[p_{\theta^{old}}(x_i)]
\end{aligned}
$$

And thus $l(\theta^{old}) = F(\gamma, \theta^{old})$.

e)

$$
\begin{aligned}
F_{\theta^{old}}(\theta) &= \sum_{i=1}^{n} \sum_{j=1}^{k} p_{\theta^{old}}(Z_i = j|x_i) log[\frac{p_\theta(x_i, Z_i = j)}{p_{\theta^{old}}(Z_i = j|x_i)}] \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} log[p_\theta(x_i|Z_i = j) p_\theta(Z_i = j)] - \gamma_{ij} log[p_{\theta^{old}}(Z_i = j|x_i)]
\end{aligned}
$$

Deriving $\hat{\mu_j}$: update rule for μ_j :
We can ignore the second term since it only involves old terms.

$$
\begin{aligned}
&= \frac{d}{d\mu_j} [\sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} log(p_\theta(x_i|Z_i = j) p_\theta(Z_i = j))] \\
&= \frac{d}{d\mu_j} [\sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} log[\frac{\eta_j}{(2\pi)^{d/2} |\sum|^{1/2}} exp(-\frac{1}{2}(x_i - \mu_j)^T \sum^{-1} (x_i - \mu_j))]] \\
&= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \frac{d}{d\mu_j} [log[\frac{\eta_j}{(2\pi)^{d/2} |\sum|^{1/2}}] + log[exp((x_i - \mu_j)^T \sum^{-1} (x_i - \mu_j))]] \\
&= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} \frac{d}{d\mu_j} (\frac{1}{\sigma_j^2} (x_i - \mu_j)^T (x_i - \mu_j)) \\
&= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{\gamma_{ij} * (-2x_i)}{\sigma_j^2} - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{\gamma_{ij} * (2\mu_j)}{\sigma_j^2} = 0 \\
&= \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{\gamma_{ij} x_i}{\sigma_j^2} - \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{\gamma_{ij} \mu_j}{\sigma_j^2} = 0
\end{aligned}
$$

$$
\boxed{\hat{\mu_j} = \frac{\sum\limits_{i=1}^{n} \gamma_{ij} x_i}{\sum\limits_{i=1}^{n} \gamma_{ij}}}
$$

Deriving $\hat{\eta_j}$: update rule for η_j :
Only one term depends on η_j.

$F_{\theta^{old}}(\theta) = \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k}\gamma_{ij}log[p_\theta(x_i|Z_i=j)] + \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k}\gamma_{ij}log[\eta_j] - \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k}\gamma_{ij}log[p_{\theta^{old}}(Z_i=j|x_i)]$

We want to maximize $\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k}\gamma_{ij}log[\eta_j]$ subject to the constraint that $\sum\limits_{j=1}^{k}\eta_j = 1$.

$$L = \sum_{i=1}^{n}\sum_{j=1}^{k} log\eta_j - \lambda\sum_{j=1}^{k}\eta_j$$
$$\frac{\partial L}{\partial \eta_j} = \frac{1}{\eta_j}\sum_{i=1}^{n}\gamma_{ij} - \lambda = 0$$
$$\eta_j = \frac{\sum\limits_{i=1}^{n}\gamma_{ij}}{\lambda}$$
$$\sum_{j=1}^{k}\eta_j = \frac{\sum\limits_{j=1}^{k}\sum\limits_{i=1}^{n}\gamma_{ij}}{\lambda} = \frac{\sum\limits_{j=1}^{k}1}{\lambda} = 1$$
$$\frac{\eta}{\lambda} = 1$$
$$\lambda = \eta$$

$$\boxed{\eta_j = \frac{\sum\limits_{i=1}^{n}\gamma_{ij}}{\eta}}$$

Deriving $\hat{\sum}_j$: update rule for $\sum_j$:
Note that $|\sum_j| = \sigma_j^{2d}$, $\sum_j^{-1} = \frac{1}{\sigma_d^2}I_d$ since $\sum_j = \sigma_j^2 I_d$.
Need to maximize: $\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{k} -\frac{1}{2}\gamma_{ij}log|\sum_j| - \frac{1}{2}\gamma_{ij}(x_i-\mu_j)^T{\sum_j}^{-1}(x_i-\mu_j)$

$$= \sum_{i=1}^{n}\sum_{j=1}^{k}\gamma_{ij}[-\frac{1}{2}log(\sigma^{2d}) - \frac{1}{2\sigma_j^2}(x_i-\mu_j)^T(x_i-\mu_j)]$$
$$\frac{\partial(...)}{\partial\sigma_j} = \sum_{i=1}^{n}\gamma_{ij}[-\frac{d}{\sigma_j} + \frac{2}{2\sigma_j^2}(x_i-\mu_j)^T(x_i-\mu_j)] = 0$$
$$\hat{\sigma_j^2} = \frac{\sum\limits_{i=1}^{n}\gamma_{ij}(x_i-\mu_j)^T(x_i-\mu_j)}{\sum\limits_{i=1}^{n}\gamma_{ij}d}$$

$$\boxed{\hat{\sum}_j = \hat{\sigma_j^2}I_d}$$

f) For diagonal Gaussians, the derivation of η_j and μ_j don't depend on $\sum_j$. We have as before $\hat{\eta_j} = \frac{\sum_{i=1}^{n}\gamma_{ij}}{\eta}$ and $\hat{\mu_j} = \frac{\sum_{i=1}^{n}\gamma_{ij}x_i}{\sum_{i=1}^{n}\gamma_{ij}}$.

Deriving $\hat{\sum_j}$: update rule for $\sum_j$:

Now, $\sum_j = diag(\sigma_{j1}^2, \sigma_{j2}^2, ..., \sigma_{jd}^2)$, $|\sum_j| = \prod_{l=1}^{d} \sigma_{jl}^2$.

Need to maximize $\sum_{i=1}^{n} \sum_{j=1}^{k} -\frac{1}{2}\gamma_{ij} log|\sum_j| - \frac{1}{2}\gamma_{ij}(x_i - \mu_j)^T {\sum_j}^{-1}(x_i - \mu_j)$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} -\frac{1}{2}\gamma_{ij} log(\prod_{l=1}^{d} \sigma_{jl}^2) - \frac{1}{2}\gamma_{ij}(x_i - \mu_j)^T * diag(\hat{\sigma_{j1}^2}, \hat{\sigma_{j2}^2}, ..., \hat{\sigma_{jd}^2})(x_i - \mu_j)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} -\frac{1}{2}\gamma_{ij} (\sum_{l=1}^{d} log(\sigma_{jl}^2)) - \frac{1}{2}\gamma_{ij}$$

$$\frac{\partial}{\partial \sigma_{jl}} = \sum_{i=1}^{n} -\frac{\gamma_{ij}}{\sigma_{jl}} + \frac{\gamma_{ij}(x_{il} - \mu_{jl})^2}{\sigma_{jl}^3} = 0$$

$$\sum_{i=1}^{n} \gamma_{ij}\sigma_{jl}^2 = \sum_{i=1}^{n} \gamma_{ij}(x_{il} - \mu_{jl})^2$$

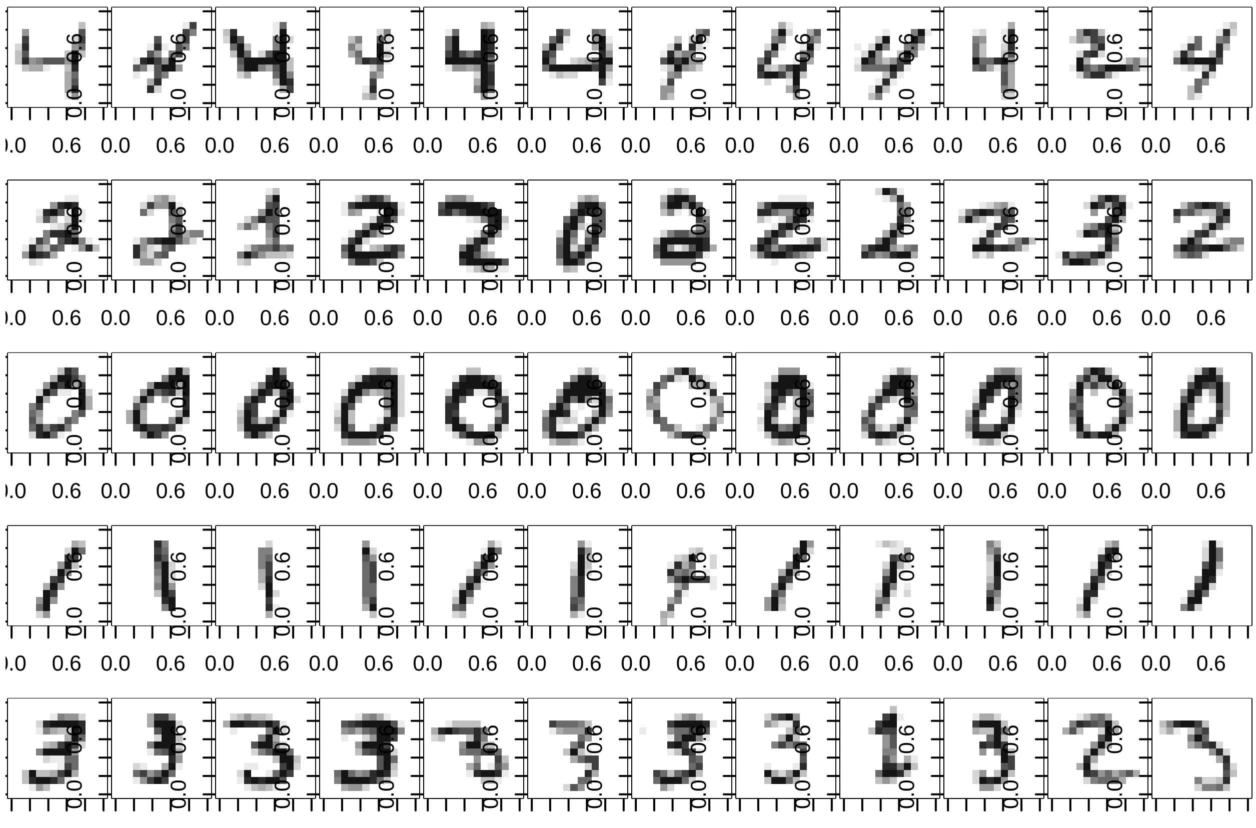
$$\hat{\sigma_{jl}^2} = \frac{\sum_{i=1}^{n} \gamma_{ij}(x_{il} - \mu_{jl})^2}{\sum_{i=1}^{n} \gamma_{ij}}$$

$$\boxed{\hat{\sum_j} = diag(\hat{\sigma_{j1}^2}, \hat{\sigma_{j2}^2}, ..., \hat{\sigma_{jd}^2})}$$

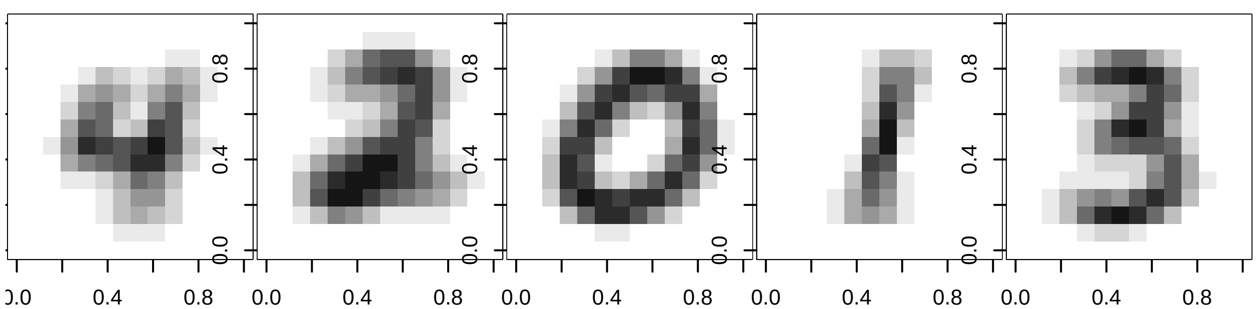
```
### Spherical Gaussians ### (code modified from blog post)
load("compress.train.images.RData")

log.sum <- function(v) {
    log.sum.pair <- function(x, y) {
        if ((y == -Inf) && (x == -Inf)) {
            return(-Inf)
        }
        if (y < x)
            return(x + log1p(exp(y - x))) else return(y + log1p(exp(x - y)))
    }

    r <- v[1]
    for (i in 2:length(v)) r <- log.sum.pair(r, v[i])
    return(r)
}

calc.gamma <- function(mat.images) {
    tmpvalues = rep(NA, k)

    for (j in 1:k) {
        tmpsigsquare = as.numeric(sigma[[j]][1, 1])
        tmpvalues[j] = log(eta[j]) - (14 * 14)/2 * log(2 * pi) -
            0.5 * 14 * 14 * log(tmpsigsquare)
    }

    calc.gamma_indiv <- function(i) {
```

```
        tmpvec = rep(NA, k)
        for (j in 1:k) {
            tmpsigsquare = as.numeric(sigma[[j]][1, 1])
            tmpvecdiff = mat.images[i, ] - mu[j, ]
            tmpvec[j] = tmpvalues[j] - 0.5 * (1/tmpsigsquare) *
                tmpvecdiff %*% tmpvecdiff
        }

        return(tmpvec)
    }

    tmpmat.log = t(sapply(1:nrow(mat.images), calc.gamma_indiv))
    tmpvec.logsum = apply(tmpmat.log, 1, log.sum)

    for (j in 1:k) {
        tmpmat.log[, j] = tmpmat.log[, j] - tmpvec.logsum
    }
    tmpmat.log = exp(tmpmat.log)
    return(list(gamma = tmpmat.log, obj = sum(tmpvec.logsum)))
}

calc.Sigma <- function(j, mat.images) {
    # add pertubation to diagonal
    sigma.squared <- 0.05
    for (i in 1:nrow(mat.images)) {
        sigma.squared <- sigma.squared + gamma[i, j] * t(mat.images[i,
            ] - mu[j, ]) %*% (mat.images[i, ] - mu[j, ])
    }

    sigma.squared = sigma.squared/(14 * 14 * sum(gamma[, j]))
    tmpmat = as.numeric(sigma.squared) * diag(14 * 14)
    return(tmpmat)
}

getLabel <- function(gamma) {
    label = rep(0, nrow(gamma))
    for (i in 1:nrow(gamma)) {
        maxVal = gamma[i, 1]
        maxIndex = 1
        for (j in 1:ncol(gamma)) {
            if (gamma[i, j] > maxVal) {
                maxVal = gamma[i, j]
                maxIndex = j
            }
        }
        label[i] = maxIndex
    }
    return(label)
}

k <- 5
best_mu <- NULL
best_gamma <- NULL
```

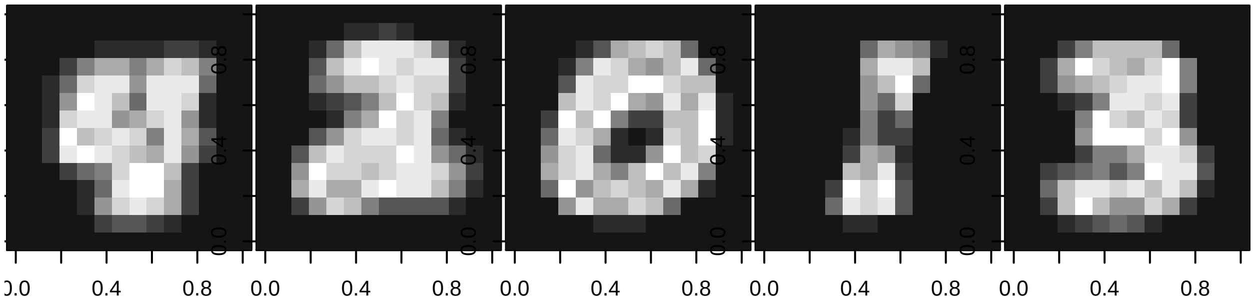
```
max_obj <- -Inf
for (i in 1:3) {
    # initialize parameters
    mu <- compress.train.images[sample(0:nrow(compress.train.images),
        5), ]
    eta <- rep(1/k, k)
    sigma <- list(0)
    for (j in 1:k) {
        sigma[[j]] = 2 * diag(14 * 14)
    }
    gamma <- matrix(0, ncol = k, nrow = nrow(compress.train.images))

    num.EPS = 1e-04
    num.iter = 1
    num.MAXITER = 100
    vec.obj = rep(NA, num.MAXITER)
    while (TRUE) {
        # E-step: calculate gammas
        eres = calc.gamma(compress.train.images)
        vec.obj[num.iter] = eres$obj
        gamma = eres$gamma
        if (num.iter > 1 && abs((vec.obj[num.iter] - vec.obj[num.iter -
            1])/vec.obj[num.iter - 1]) < num.EPS)
            (break)()
        # M-step: updating the parameters
        for (j in 1:k) {
            eta[j] = 1/nrow(compress.train.images) * sum(gamma[,
                j])
            mu[j, ] = t(gamma[, j] %*% compress.train.images)/sum(gamma[,
                j])
            sigma[[j]] = calc.Sigma(j, compress.train.images)
        }

        num.iter = num.iter + 1
        if (num.iter > num.MAXITER)
            (break)()
    }
    if (vec.obj[num.iter] > max_obj) {
        best_mu = mu
        best_gamma = getLabel(gamma)
        max_obj <- vec.obj[num.iter]
    }
}

plotTable(12, best_gamma, compress.train.images)
```



```r
# Plot 2
par(mfrow = c(1, 5))
for (i in 1:5) {
    show_digitsmall(best_mu[i, ])
}
```



```r
# Plot 3
for (i in 1:5) {
    curr_cluster <- compress.train.images[which(best_gamma ==
        i), ]
    var_mat <- apply(curr_cluster, 2, var)
    var_mat <- rescale(var_mat, c(0, 255))
    show_digitsmall(255 - var_mat)
}
```




```r
# Table 1
table <- matrix(0, 5, 5)
for (i in 1:5) {
    ithcluster <- which(best_gamma == i)
    for (j in 0:4) {
        count <- 0
        for (k in 1:length(ithcluster)) {
            if (train.labels[ithcluster[k]] == j)
                count = count + 1
        }
        table[i, j + 1] = count
    }
}
error <- rep(0, 5)
for (i in 1:5) {
    error[i] <- 1 - max(table[i, ])/sum(table[i, ])
}
print(table)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   48   27  268  130 5598
## [2,]  324  717 4966  634  116
## [3,] 5253    0   74   57   19
## [4,]    2 5806  124  118   94
## [5,]  296  192  526 5192   15
```

```r
paste("Cluster 1 represents digit", which.max(table[1, ]) - 1)
```

```
## [1] "Cluster 1 represents digit 4"
```

```r
paste("Cluster 2 represents digit", which.max(table[2, ]) - 1)
```

```
## [1] "Cluster 2 represents digit 2"
```

```r
paste("Cluster 3 represents digit", which.max(table[3, ]) - 1)
```

```
## [1] "Cluster 3 represents digit 0"
```

```r
paste("Cluster 4 represents digit", which.max(table[4, ]) - 1)
```

```
## [1] "Cluster 4 represents digit 1"
```

```r
paste("Cluster 5 represents digit", which.max(table[5, ]) - 1)
```

```
## [1] "Cluster 5 represents digit 3"
```

```r
indices <- c(which.max(table[1, ]), which.max(table[2, ]), which.max(table[3,
    ]), which.max(table[4, ]), which.max(table[5, ]))
```

```r
print(error)
```

```
## [1] 0.07791138 0.26505846 0.02776235 0.05501302 0.16540749
```

```r
paste("Digit 1 error rate:", error[which(indices == 1)])
```

```
## [1] "Digit 1 error rate: 0.0277623542476402"
```

```r
paste("Digit 2 error rate:", error[which(indices == 2)])
```

```
## [1] "Digit 2 error rate: 0.0550130208333334"
```

```r
paste("Digit 3 error rate:", error[which(indices == 3)])
```

```
## [1] "Digit 3 error rate: 0.265058457895516"
```

```r
paste("Digit 4 error rate:", error[which(indices == 4)])
```

```
## [1] "Digit 4 error rate: 0.165407490757113"
```

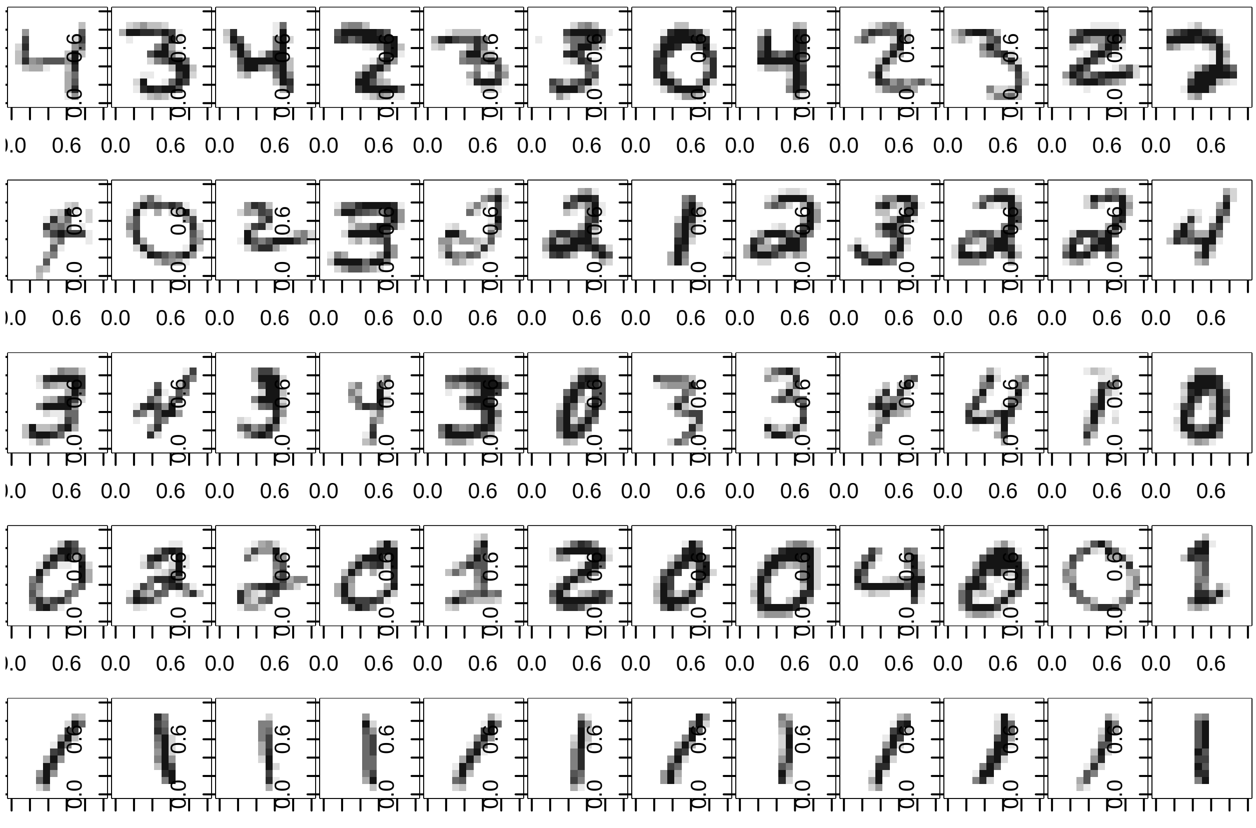
```r
paste("Digit 5 error rate:", error[which(indices == 5)])
```

```
## [1] "Digit 5 error rate: 0.0779113819799044"
```

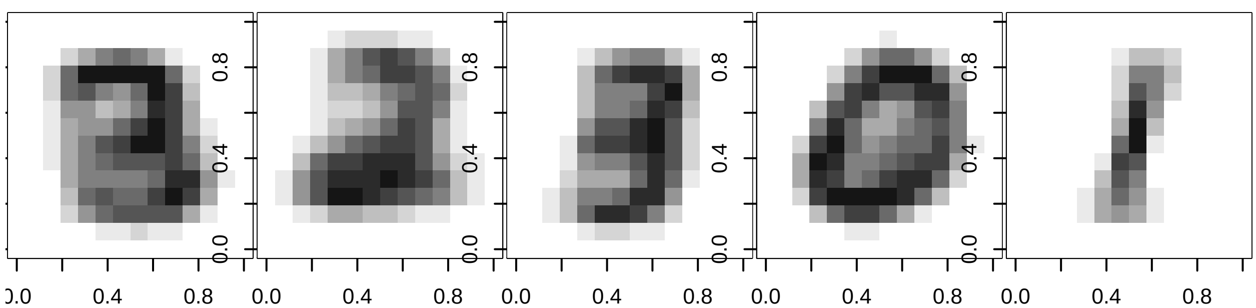
```r
### Diagonal Gaussians ### (code modified from blog post)
load("compress.train.images.RData")
log.sum <- function(v) {
    log.sum.pair <- function(x, y) {
        if ((y == -Inf) && (x == -Inf)) {
            return(-Inf)
        }
        if (y < x)
            return(x + log1p(exp(y - x))) else return(y + log1p(exp(x - y)))
    }

    r <- v[1]
    for (i in 2:length(v)) r <- log.sum.pair(r, v[i])
    return(r)
}

calc.gamma <- function(mat.images) {
    tmpsigsquare = matrix(0, nrow = 5, ncol = 14 * 14)
    tmpinvsigsquare = matrix(0, nrow = 5, ncol = 14 * 14)
    for (j in 1:5) {
        for (i in 1:(14 * 14)) {
            tmpsigsquare[j, i] = as.numeric(sigma[[j]][i, i])
            tmpinvsigsquare[j, i] = 1/tmpsigsquare[j, i]
        }
    }
    tmpvalues = rep(NA, k)
    tmp.identity = matrix(1, 14 * 14, 1)
    # 5 by 1 matrix, each entry is log determinant of Sigma
    tmp.logDeterminant = log(tmpsigsquare) %*% tmp.identity

    for (j in 1:k) {
        tmpvalues[j] = log(eta[j]) - (14 * 14)/2 * log(2 * pi) -
            0.5 * tmp.logDeterminant[j]
    }
```

```
    calc.gamma_indiv <- function(i) {
        tmpvec = rep(NA, k)
        for (j in 1:k) {
            tmpvecdiff = mat.images[i, ] - mu[j, ]
            tmpvec[j] = tmpvalues[j] - 0.5 * (tmpvecdiff * tmpvecdiff) %*%
                t(tmpinvsigsquare[j, , drop = FALSE])
        }

        return(tmpvec)
    }
    tmpmat.log = t(sapply(1:dim(mat.images)[1], calc.gamma_indiv))
    tmpvec.logsum = apply(tmpmat.log, 1, log.sum)
    for (j in 1:k) {
        tmpmat.log[, j] = tmpmat.log[, j] - tmpvec.logsum
    }
    tmpmat.log = exp(tmpmat.log)
    return(list(gamma = tmpmat.log, obj = sum(tmpvec.logsum)))
}

calc.Sigma <- function(j, mat.images) {
    tmpmat = matrix(0, 14 * 14, 14 * 14)
    sigma.tmp = matrix(0, 1, 14 * 14)
    for (i in 1:dim(mat.images)[1]) {
        sigma.tmp = sigma.tmp + gamma[i, j] * (mat.images[i,
            ] - mu[j, ]) * (mat.images[i, ] - mu[j, ])
    }
    for (p in 1:(14 * 14)) {
        tmpmat[p, p] = sigma.tmp[1, p]/sum(gamma[, j])
    }
    tmpmat = tmpmat + diag(14 * 14)
    return(tmpmat)
}

k <- 5
best_mu <- NULL
best_gamma <- NULL
max_obj <- -Inf
for (i in 1:3) {
    # initialize parameters
    mu <- compress.train.images[sample(0:nrow(compress.train.images),
        5), ]
    eta <- rep(1/k, k)
    sigma <- list(0)
    for (j in 1:k) {
        sigma[[j]] = 2 * diag(14 * 14)
    }
    gamma <- matrix(0.1, ncol = k, nrow = nrow(compress.train.images))

    num.EPS = 1e-04
    num.iter = 1
    num.MAXITER = 100
    vec.obj = rep(NA, num.MAXITER)
```

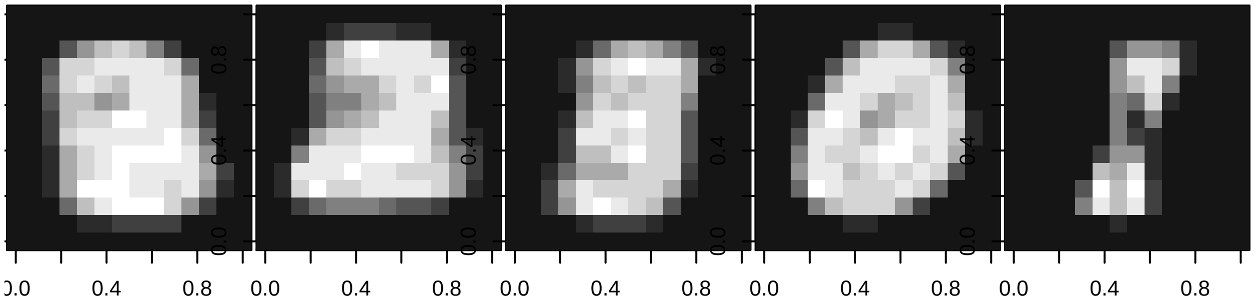
```
    while (TRUE) {
        # E-step: calculate gammas
        eres = calc.gamma(compress.train.images)
        vec.obj[num.iter] = eres$obj
        gamma = eres$gamma

        should.break = FALSE
        for (j in 1:k) {
            if (sum(gamma[, j]) == 0) {
                should.break = TRUE
            }
        }
        if (should.break)
            (break)()

        if (num.iter > 1 && abs((vec.obj[num.iter] - vec.obj[num.iter -
            1])/vec.obj[num.iter - 1]) < num.EPS)
            (break)()

        # M-step: updating the parameters
        for (j in 1:k) {
            eta[j] = 1/nrow(compress.train.images) * sum(gamma[,
                j])
            mu[j, ] = t(gamma[, j] %*% compress.train.images)/sum(gamma[,
                j])
            sigma[[j]] = calc.Sigma(j, compress.train.images)
        }

        num.iter = num.iter + 1
        if (num.iter > num.MAXITER)
            (break)()
    }
    if (vec.obj[num.iter] > max_obj) {
        best_mu = mu
        best_gamma = getLabel(gamma)
        max_obj <- vec.obj[num.iter]
    }
}
plotTable(12, best_gamma, compress.train.images)
```



```
# Plot 2
par(mfrow = c(1, 5))
for (i in 1:5) {
    show_digitsmall(best_mu[i, ])
}
```



```
# Plot 3
for (i in 1:5) {
    curr_cluster <- compress.train.images[which(best_gamma ==
        i), ]
    var_mat <- apply(curr_cluster, 2, var)
    var_mat <- rescale(var_mat, c(0, 255))
    show_digitsmall(255 - var_mat)
}
```



```r
# Table 1
table <- matrix(0, 5, 5)
for (i in 1:5) {
    ithcluster <- which(best_gamma == i)
    for (j in 0:4) {
        count <- 0
        for (k in 1:length(ithcluster)) {
            if (train.labels[ithcluster[k]] == j)
                count = count + 1
        }
        table[i, j + 1] = count
    }
}
error <- rep(0, 5)
for (i in 1:5) {
    error[i] <- 1 - max(table[i, ])/sum(table[i, ])
}
print(table)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  332   37 1617 1411  931
## [2,]  199   53 1501  386  241
## [3,] 1318  832  913 3813 2931
## [4,] 4071  203 1914  438 1729
## [5,]    3 5617   13   83   10
```

```r
paste("Cluster 1 represents digit", which.max(table[1, ]) - 1)
```

```
## [1] "Cluster 1 represents digit 2"
```

```r
paste("Cluster 2 represents digit", which.max(table[2, ]) - 1)
```

```
## [1] "Cluster 2 represents digit 2"
```

```r
paste("Cluster 3 represents digit", which.max(table[3, ]) - 1)
```

```
## [1] "Cluster 3 represents digit 3"
```

```r
paste("Cluster 4 represents digit", which.max(table[4, ]) - 1)
```

```
## [1] "Cluster 4 represents digit 0"
```

```r
paste("Cluster 5 represents digit", which.max(table[5, ]) - 1)
```

```
## [1] "Cluster 5 represents digit 1"
```

```r
indices <- c(which.max(table[1, ]), which.max(table[2, ]), which.max(table[3,
    ]), which.max(table[4, ]), which.max(table[5, ]))
```

```r
print(error)
```

```
## [1] 0.62638632 0.36932773 0.61119608 0.51274686 0.01903598
```

```r
paste("Digit 1 error rate:", error[which(indices == 1)])
```

```
## [1] "Digit 1 error rate: 0.512746858168761"
```

```r
paste("Digit 2 error rate:", error[which(indices == 2)])
```

```
## [1] "Digit 2 error rate: 0.0190359762486901"
```

```r
paste("Digit 3 error rate:", error[which(indices == 3)])
```

```
## [1] "Digit 3 error rate: 0.626386321626617"
## [2] "Digit 3 error rate: 0.369327731092437"
```

```r
paste("Digit 4 error rate:", error[which(indices == 4)])
```

```
## [1] "Digit 4 error rate: 0.611196084429489"
```

```r
paste("Digit 5 error rate:", error[which(indices == 5)])
```

```
## [1] "Digit 5 error rate: "
```

Question 1.3: Relationship between K-Means and EM

Set $\sigma_1^2 = \sigma_2^2 = ... = \sigma_j^2$ so that $\sum_i = \sum_j$ for all $i \neq j$. Then,

$$\begin{aligned}
\gamma_{ij} &= \frac{p_\theta(x_i|z_i = j)p(z_i = j)}{\sum\limits_{l=1}^{k} p_\theta(x_i|z_i = l)p(z_i = l)} \\
&= \frac{\eta_j exp(-\frac{1}{2\sigma^2}||x_i - \mu_j||_2^2)}{\sum\limits_{l=1}^{k} \eta_l exp(-\frac{1}{2\sigma^2}||x_i - \mu_l||_2^2)}
\end{aligned}$$

When $\sigma^2 \to 0$, we have the following hard assignment the k-means algorithm.

$\gamma_{ij} = 1$ if $||x_i - \mu_j||_2^2 < ||x_i - \mu_l||_2^2$ for all $i \neq j$, $\gamma_{ij} = 0$ otherwise.

Question 1.4: Comparison of K-Means and EM

1) The k-means and spherical Gaussians models performed similarly with diagonal Gaussians performing the worst by far. The outputs of k-means and spherical Gaussians were both pretty satisfactory; the clusters clearly identified a unique digit and the error rates were comparatively low. In contrast, the diagonal Gaussians model did not distinguish between digits clearly as Plot 2 and 3 show (the output is amorphous). The images in each cluster vary by digit.

The EM algorithm assumes all data is generated from degenerate multivariate Gaussians and thus provides a great deal of flexibility by assigning probabilities that a data point belongs to a given cluster rather than outright assigning a cluster as in k-means. Since Gaussian distributions are the limiting distributions of everything, the EM algorithm is good at modeling things that can be represented with Gaussians. Intuitively, people's handwritings look like they should come from a Gaussian distribution since most people preserve certain common characteristics in their writing (thus allowing us to read). Some people have atrociously bad handwriting and some have amazing handwriting, but most people have average handwriting that is decipherable. As the k-means algorithm is the limiting algorithm of the k-spherical gaussians EM algorithm, it makes sense that k-means should also model the data well.

K-means and EM algorithm tend to emphasize features that occur in many images of the same class, so they will ignore traits that do not show up often between digits of the same class, such as the way you cross your 4s or whether you add a stem to your 1 vs. drawing a vertical line.

2) In order of increasing runtime: k-means, spherical, diagonal. The diagonal model took by far the longest amount of time and spherical took only slightly longer than k-means. Since k-means and spherical both performed the same, these two models provided relatively good outputs in reasonable time.

3) Mixture models seemed to be good for modeling the data as long as the model didn't overfit. The diagonal model seemed to overfit by assuming each pixel has an individual variance, thus not capturing enough of the large-scale information.

4) I just randomly sampled five rows of the image data to start as clusters. I this this strategy was successful since in each model I took the best of three random initializations, and k-means/spherical both clustered pretty well.

Question 2: EM Algorithms for Arbitrary Distributions (20 points)

2.1

$P(Z=1|X) = \frac{P(X|Z=1)P(Z=1)}{P(X|Z=1)P(Z=1)+P(X|Z=0)P(Z=0)} = \frac{\eta p_1(x)}{\eta p_1(x)+(1-\eta)p_2(x)}$

2.2

$$
\begin{aligned}
log(P(X_1, Z_1, X_2, Z_2...X_n, Z_n)) &= \sum_{i=1}^{n} logp(X_i, Z_i) \\
&= \sum_{i=1}^{n} log(p(X_i|Z_i)P(Z_i)) \\
&= \sum_{i=1}^{n} log(p_1(x_i)^{z_i}) + \sum_{i=1}^{n} log(p_1(x_i)^{1-z_i}) + \sum_{i=1}^{n} log(\eta^{z_i}) + \sum_{i=1}^{n} (1-\eta)^{1-z_i} \\
&= \sum_{i=1}^{n} z_i(log(p_1(x_i)) + log(\eta)) + (1-z_i)(logp_2(x_i) + log(1-\eta))
\end{aligned}
$$

2.3

E-Step: We first need to derive a lower-bound function such that $F_\psi \leq l(\psi) = \sum_{i=1}^{n} logP_\psi(x_i)$ and $F_\psi(\psi_{old}) = l(\psi_{old})$.

Let $\gamma_{ij} = P_{\psi^{old}}(Z_i = j|X_i)$

$$
\begin{aligned}
L(\psi) &= \sum_{i=1}^{n} log p_\psi(x_i) \\
&= \sum_{i=1}^{n} log(\sum_{j=0}^{1} p_\psi(x_i, Z_i = j)) \\
&= \sum_{i=1}^{n} log[\sum_{j=0}^{1} \gamma_{ij} \frac{p_\psi(x_i, Z_i = j)}{\gamma_{ij}}] \\
&= \sum_{i=1}^{n} log E[\frac{p_\psi(x_i, Z_i = j)}{\gamma_{ij}}] \\
&\geq \sum_{i=1}^{n} E[log(\frac{p_\psi(x_i, Z_i = j)}{\gamma_{ij}})], \ By\ Jensen's\ Inequality \\
Let\ F_\psi &= \sum_{i=1}^{n} \sum_{j=0}^{1} \gamma_{ij} log(\frac{p_\psi(x_i, Z_i = j)}{\gamma_{ij}}) \\
F_\psi(\psi_{old}) &= \sum_{i=1}^{n} \sum_{j=0}^{1} \gamma_{ij} log[\frac{p_{\psi^{old}}(x_i, Z_i = j)}{p_{\psi^{old}}(x_i|Z_i = j)}] \\
&= \sum_{i=1}^{n} \sum_{j=0}^{1} p_{\psi^{old}}(x_i|Z_i = z_i) log[p_{\psi^{old}}(x_i)] \\
&= \sum_{i=1}^{n} log[p_{\psi^{old}}(x_i)] \sum_{j=0}^{1} p_{\psi^{old}}(x_i|Z_i = z_i) \\
&= \sum_{i=1}^{n} log p_{\psi^{old}}(x_i) \\
&= l(\psi_{old})
\end{aligned}
$$

This justifies the subsequent E-Step of the algorithm. $F_{\psi^{(t)}}(\psi) = \sum_{i=1}^{n} \sum_{j=0}^{1} \gamma_{ij}^{(t+1)} log[\frac{p_\psi(x_i|Z_i=j)\eta_j}{\gamma_{ij}^{(t+1)}}]$ where $\eta_1 = \eta$ and $\eta_0 = 1 - \eta$. Assuming we are on the t-th iteration, we now compute $\gamma_{ij}^{(t+1)}$:

$$
\begin{aligned}
\gamma_{i1}^{(t+1)} &= p_{\psi^{(t)}}(Z_i = 1|X_i) \\
&= \frac{\eta^{(t)} p_1(x_i)}{\eta^{(t)} p_1(x_i) + (1 - \eta^{(t)}) p_2(x_i)} \\
\gamma_{i0}^{(t+1)} &= p_{\psi^{(t)}}(Z_i = 0|X_i) \\
&= \frac{(1 - \eta^{(t)}) p_1(x_i)}{\eta^{(t)} p_1(x_i) + (1 - \eta^{(t)}) p_2(x_i)}
\end{aligned}
$$

Now we have $F_{\psi^{(t)}}(\psi)$.

M-Step: We want to maximize $F_{\psi^{(t)}}(\psi)$ with respect to η. That is, we need to maximize $\sum_{i=1}^{n} \sum_{j=0}^{1} \gamma_{i0}^{(t+1)} log\eta$ such that $\eta_0 = 1 - \eta$ and $\eta_1 = \eta$. Or: $\sum_{i=1}^{n} \gamma_{i0}^{(t+1)} log(1 - \eta) + \gamma_{i1}^{(t+1)} log(\eta)$

$$
\begin{aligned}
\frac{\partial(...)}{\partial\eta} &= \sum_{i=1}^{n} -(\frac{\gamma_{i0}^{(t+1)}}{1-\eta}) + \frac{\gamma_{i1}^{(t+1)}}{\eta} = 0 \\
\eta\sum_{i=1}^{n}\gamma_{i0}^{(t+1)} &= (1-\eta)\sum_{i=1}^{n}\gamma_{i1}^{(t+1)} \\
\eta(\sum_{i=1}^{n}\gamma_{i0}^{(t+1)} + \gamma_{i1}^{(t+1)}) &= \sum_{i=1}^{n}\gamma_{i1}^{(t+1)} \\
\hat{\eta} &= \frac{\sum_{i=1}^{n}\gamma_{i1}^{(t+1)}}{\sum_{i=1}^{n}\gamma_{i0}^{(t+1)} + \gamma_{i1}^{(t+1)}} \\
&= \frac{\sum_{i=1}^{n}\gamma_{i1}^{(t+1)}}{\sum_{i=1}^{n} 1} \\
&= \frac{\sum_{i=1}^{n}\frac{\eta^{(t)}p_1(x_i)}{\eta^{(t)}p_1(x_i)+(1-\eta^{(t)})p_2(x_i)}}{\eta}
\end{aligned}
$$

2.4

The EM algorithm is guaranteed to converge because our lower bound $F_\psi(\psi)$ is tight such that $F_\psi(\psi_{old}) = l(\psi_{old})$. Because we are improving our lower bound each step the objective function must also improve. And since the objective function doesn't go to infinity, the algorithm has to stop somewhere (at a local maximum).

Question 3: K-Means Wikipedia Documents Clustering (30 points)

```
load("Wikipedia.RData")  # loads dat object
# 3.1
print(head(dat))
```

```
##                   name
## 1         Michel Che
## 2 Hossein Modarressi
## 3      Xiao-Gang Wen
## 4  Nicholas Doumanis
## 5         Don Cahoon
## 6        Lance Davis
##
## 1
## 2
## 3
## 4
## 5 don toot cahoon is a retired american ice hockey coach he was the head coach of the princeton tige
## 6
```

```
paste("First three individuals are:")
```

```
## [1] "First three individuals are:"
```

```
dat[, 1][1:3]
```

```
## [1] Michel Che         Hossein Modarressi Xiao-Gang Wen
## 59070 Levels:  Renate Lorenz \\'Ilima Lei Tohi ... Zygfryd Szo%C5%82tysik
```

```
paste("These first three individuals are all academics")
```

```
## [1] "These first three individuals are all academics"
```

```
# Run script1_HW5.R
text = dat[, "text"]
text = iconv(text, to = "utf-8")  #some conversion on SMILE needed
corpus = Corpus(VectorSource(text))
dtm.control.raw <- list(tolower = TRUE, removePunctuation = TRUE,
    removeNumbers = TRUE, removestopWords = TRUE, stemming = TRUE,
    wordLengths = c(3, 15), bounds = list(global = c(2, Inf)))

dtm.raw = DocumentTermMatrix(corpus, control = dtm.control.raw)
dtm.mat.raw = as.matrix(dtm.raw)  # our term-document matrix

paste("Number of individuals:", nrow(dtm.mat.raw))
```

```
## [1] "Number of individuals: 812"
```

```
paste("Number of words:", length(unique(colnames(dtm.mat.raw))))
```

```
## [1] "Number of words: 6889"
```

```
paste("10 most common words:")
```

```
## [1] "10 most common words:"
```

```
sort(colSums(dtm.mat.raw), decreasing = TRUE)[1:10]
```

```
##      the      and univers      for      was      his     from      has     with 
##    17317    10729     3169     3165     2817     2353     2126     1708     1674 
##      new 
##     1174
```

```r
paste("0%, 25%, 50%, 75%, 100% quantiles")
```

```
## [1] "0%, 25%, 50%, 75%, 100% quantiles"
```

```r
quantile(colSums(dtm.mat.raw))
```

```
##    0%   25%   50%   75%  100% 
##     2     3     5    14 17317
```

```r
# 3.2 Run script2_HW5.R
dtm.control <- list(tolower = TRUE, removePunctuation = TRUE,
    removeNumbers = TRUE, removestopWords = TRUE, stemming = TRUE,
    wordLengths = c(3, 15), bounds = list(global = c(2, Inf)),
    weighting = function(x) {
        weightTfIdf(x, normalize = FALSE)
    })

dtm = DocumentTermMatrix(corpus, control = dtm.control)
dtm.mat = as.matrix(dtm)

paste("10 words with highest weight:")
```

```
## [1] "10 words with highest weight:"
```

```r
sort(colSums(dtm.mat), decreasing = TRUE)[1:10]
```

```
##       she       her     music    econom       law    scienc  mathemat 
## 2414.6500 1611.8679 1179.0219 1160.7760 1132.1644  989.8247  975.4508 
##       new   histori  research 
##  908.1503  905.1748  903.4017
```

```r
# 3.3 Run script3_HW5.R
dtm.mat.indicator = dtm.mat.raw
dtm.mat.indicator[dtm.mat.indicator != 0] = 1

word.presence = apply(dtm.mat.indicator, 2, sum)
idx = which(word.presence >= quantile(word.presence, prob = 0.99))
dtm.mat.raw = dtm.mat.raw[, -idx]

common.words = read.csv("google-10000-english.txt", header = F)
idx = which(colnames(dtm.mat.raw) %in% common.words[1:300, 1])
dtm.mat.raw = dtm.mat.raw[, -idx]

paste("Words remaining in analysis:", ncol(dtm.mat.raw))
```

```
## [1] "Words remaining in analysis: 6665"
```

```r
paste("Top 10 words in remaining matrix:")
```

```
## [1] "Top 10 words in remaining matrix:"
```

```r
sort(colSums(dtm.mat.raw), decreasing = TRUE)[1:10]
```

```
##  histori   econom    polit      law mathemat   theori   physic     play
```

```
##      418      397      390      389      329      323      310      295
##  academi    board
##      290      287
```

```r
# 3.4
dat[which(dat$name == "Ben Bernanke"), 2]
```

```
## [1] ben shalom bernanke brnki brnangkee born december 13 1953 is an american economist at the brookin
## 59071 Levels: 108 born 1978 is an italian artist in the field of street art and contemporary art fron
```

```r
sort(dtm.mat[which(dat$name == "Ben Bernanke"), ], decreasing = TRUE)[1:10]
```

```
##   bernank    reserv  chairman     feder      term      bush   succeed
## 40.401867 27.087042 20.619103 20.107399 15.431924  9.929792  9.929792
##    econom     janet   volatil
##  8.771607  8.665336  8.665336
```

```r
sort(dtm.mat.raw[which(dat$name == "Ben Bernanke"), ], decreasing = TRUE)[1:10]
```

```
## chairman  bernank    feder   reserv     term   econom     bush februari
##        6        5        5        5        4        3        2        2
##   second  succeed
##        2        2
```

```r
# 3.5 normalize each row to have sum 1
dtm.mat.norm <- t(apply(dtm.mat, 1, function(x) {
    return(x/sum(x))
}))
set.seed(10)
result <- norm.sim.ksc(dtm.mat.norm, 8)
paste("Cluster sizes:")
```

```
## [1] "Cluster sizes:"
```

```r
result$size
```

```
## [1]  26 144 150 189  67  90  97  49
```

```r
paste("Top 25 words in each cluster")
```

```
## [1] "Top 25 words in each cluster"
```

```r
cluster1 <- colSums(dtm.mat.raw[which(result$cluster == 1), ])
cluster2 <- colSums(dtm.mat.raw[which(result$cluster == 2), ])
cluster3 <- colSums(dtm.mat.raw[which(result$cluster == 3), ])
cluster4 <- colSums(dtm.mat.raw[which(result$cluster == 4), ])
cluster5 <- colSums(dtm.mat.raw[which(result$cluster == 5), ])
cluster6 <- colSums(dtm.mat.raw[which(result$cluster == 6), ])
cluster7 <- colSums(dtm.mat.raw[which(result$cluster == 7), ])
cluster8 <- colSums(dtm.mat.raw[which(result$cluster == 8), ])

head(sort(cluster1, decreasing = TRUE), 25)
```

```
##     geolog      medal       armi     church       team    academi
##         24         21         18         17         15         14
##   canadian      field       gold california      divis     museum
##         14         14         14         13         13         13
##      known      offic       salt       texa      young    histori
##         12         12         12         12         12         11
```

```
##        law        lds        led        oak       play     servic
##         11         11         11         11         11         11
##      sever
##         11
```

```r
quantile(cluster1[which(cluster1 != 0)])
```

```
##   0%  25%  50%  75% 100%
##    1    1    1    3   24
```

```r
head(sort(cluster2, decreasing = TRUE), 25)
```

```
##      mathemat        physic        theori        comput         engin
##           267           266           193           190           111
##       academi         field         prize       theoret      technolog
##            90            90            85            84            75
##       quantum        advanc mathematician       develop        mechan
##            71            68            67            65            61
##     contribut         known         paper    california     physicist
##            56            52            52            51            51
##         visit       faculti    laboratori     scientist         model
##            51            49            49            48            46
```

```r
quantile(cluster2[which(cluster2 != 0)])
```

```
##   0%  25%  50%  75% 100%
##    1    1    2    4  267
```

```r
head(sort(cluster3, decreasing = TRUE), 25)
```

```
##         team         play        coach       season        leagu
##          199          146          137          112          103
##       career         game      histori         danc       assist
##           88           80           80           75           71
##       player         citi         head       jersey      perform
##           70           69           67           67           66
##          won championship       record        write        organ
##           64           60           60           57           56
##        three         mani        board         live      foundat
##           56           55           54           54           53
```

```r
quantile(cluster3[which(cluster3 != 0)])
```

```
##   0%  25%  50%  75% 100%
##    1    1    2    5  199
```

```r
head(sort(cluster4, decreasing = TRUE), 25)
```

```
##     econom        law     polici      board      offic      polit
##        328        265        131        128        121        107
##     former     affair    develop      court       educ washington
##        104         96         89         87         86         86
##     servic    journal   chairman      senat    appoint      elect
##         82         79         78         75         74         72
##       firm      manag       dure       busi     senior     assist
##         72         72         71         70         70         69
##    compani
##         69
```

```r
quantile(cluster4[which(cluster4 != 0)])
```

```
##   0%  25%  50%  75% 100% 
##    1    1    2    6  328 
```

```r
head(sort(cluster5, decreasing = TRUE), 25)
```

```
##      play    record    festiv orchestra   perform    compos      film 
##        85        82        77        75        73        70        70 
##      band   premier  symphoni    theatr    artist     album   ensembl 
##        64        52        51        50        48        46        39 
##    featur     opera      hall  composit      jazz    london    releas 
##        39        39        37        36        34        33        33 
##      mani    produc    academi      citi 
##        32        30        29        29 
```

```r
quantile(cluster5[which(cluster5 != 0)])
```

```
##   0%  25%  50%  75% 100% 
##    1    1    2    3   85 
```

```r
head(sort(cluster6, decreasing = TRUE), 25)
```

```
##      polit    histori     social      press    scholar     modern 
##        205        179         61         60         58         56 
##      human      visit     oxford      teach     econom     advanc 
##         53         53         50         49         45         43 
##     cultur philosophi       educ     jewish     recent    foundat 
##         43         43         42         42         42         41 
##    journal      relat        war    academi     theori  historian 
##         41         40         40         39         39         38 
##    america 
##         36 
```

```r
quantile(cluster6[which(cluster6 != 0)])
```

```
##   0%  25%  50%  75% 100% 
##    1    1    2    4  205 
```

```r
head(sort(cluster7, decreasing = TRUE), 25)
```

```
##    theolog     church    histori  literatur philosophi    languag 
##        149        104         87         79         72         62 
##   seminari      teach     cultur    english    scholar     taught 
##         60         58         54         52         50         50 
##    journal   religion      human       citi  christian    episcop 
##         48         47         46         42         40         40 
##     poetri      visit       mani      press     articl     modern 
##         40         40         38         38         37         37 
##     editor 
##         35 
```

```r
quantile(cluster7[which(cluster7 != 0)])
```

```
##   0%  25%  50%  75% 100% 
##    1    1    2    4  149 
```

```r
head(sort(cluster8, decreasing = TRUE), 25)
```

```
##       medic     biolog      human      genom    medicin  molecular
##          52         36         35         33         32         31
##     develop      board       gene   investig laboratori     cancer
##          29         26         26         24         23         22
## washington     attend      genet      prize       busi    faculti
##          22         21         20         19         18         18
##       later      polit       cell     career    histori       move
##          18         18         17         16         16         16
##      pulitz
##          16
```

```
quantile(cluster8[which(cluster8 != 0)])
```

```
##   0%  25%  50%  75% 100%
##    1    1    2    3   52
```

Question 4: SVM Theory (20 points)

Question 4.1: Geometric Interpretation

We have two hyperplanes $\beta^T x - b = 1$ and $\beta^T x - b = -1$. Let $\vec{x_0}$ be a vector on $\beta^T x - b = -1$. Let $\vec{x_1} = \vec{x_0} + \alpha\vec{\beta}$. Let's find α such that $\vec{x_1}$ is on $\beta^T x - b = 1$.

$$
\begin{aligned}
\beta^T \vec{x_1} - b &= 1 \\
\beta^T(\vec{x_0} + \alpha\vec{\beta}) - b &= 1 \\
\beta^T \vec{x_0} + \alpha\beta^T\vec{\beta} - b &= 1 \\
b - 1 + \alpha||\beta||^2 - b &= 1 \\
\alpha &= \frac{2}{||\beta||^2} \\
\vec{x_1} &= \vec{x_0} + \frac{2}{||\beta||^2}\vec{\beta} \\
||\vec{x_1} - \vec{x_0}|| &= \sqrt{||\frac{2}{||\beta||^2}\vec{\beta}||^2} \\
&= \sqrt{\frac{4||\vec{\beta}||^2}{||\vec{\beta}||^4}} \\
&= \frac{2}{||\vec{\beta}||}
\end{aligned}
$$

Which is our distance between the two hyperplanes.

Question 4.2: Simple Reformulation of SVM

(a) 1) Minimizing $\frac{1}{2}||\beta||_2^2$ is equivalent to maximizing the distance between the two hyperplanes, since the reciprocal of this objective function is $\frac{2}{||\beta||_2^2}$, and when $\frac{2}{||\beta||_2^2}$ is maximized, the distance $\frac{2}{||\beta||}$ is also maximized.

2) If $y_i = 1$, then we have the constraint $(\beta^T x_i - b) \geq 1$. This means data points (x_i, y_i) whose $y_i = 1$ must lie above the plane defined by $\beta^T x - b = 1$.
If $y_i = -1$, then we have the constraint $(\beta^T x_i - b) \leq 1$. This means data points (x_i, y_i) whose $y_i = -1$ must lie belows the plane defined by $\beta^T x - b = -1$.
Then, we have no data points between the hyperplanes and the data is perfectly separated.

(b) Consider the following data points A, B and C defined such that $\vec{x_A} = (-1, 0)$ and $y_A = -1$, $\vec{x_B} = (0, 0)$ and $y_B = 1$, $\vec{x_C} = (1, 0)$ and $y_C = -1$. These are not linearly separable:

Suppose we find a $\vec{\beta} = (\beta_1, \beta_2)$. Then we must have $y_B(\beta^T x_B - b) \geq 1 \rightarrow 1(0 - b) \geq 1$ so $b \leq -1$. But we also have $y_A(\beta^T x_A - b) \geq 1 \rightarrow -1(-\beta_1 - b) \geq 1$ so $\beta_1 \geq 1 - b$. We also have $y_C(\beta^T x_C - b) \geq 1 \rightarrow -1(\beta_1 - b) \geq 1$ so $\beta_1 \leq b - 1$. But since $b \leq 1$, we have $\beta_1 \geq 2$ and $\beta_1 \leq -2$ which is a contradiction.

Question 4.3: General Reformulation of SVM

We know that $\hat{\beta}, \hat{b}$ optimizes $\underset{\beta,b}{min}\frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\beta^T x_i - b)]_+ + \lambda||\beta||_2^2$ which is the same as $\underset{\beta,b}{min}\frac{1}{2n\lambda}\sum_{i=1}^{n}[1 - y_i(\beta^T x_i - b)]_+ + \frac{||\beta||_2^2}{2}$.

Let $C = \frac{1}{2n\lambda}$ and $\zeta_i = [1 - y_i(\beta^T x_i - b)]_+$. Then the objective function becomes $\underset{\beta,b}{min} C \sum_{i=1}^{n} \zeta_i + \frac{1}{2}||\beta||_2^2$. We know that $\zeta_i = max(0, 1 - y_i(\beta^T x_i - b))$, so $\zeta_i \geq 0$ and $\zeta_i \geq 1 - y_i(\beta^T x_i - b)$.

Therefore, $\hat{\beta}$, $\hat{b}$ optimizes 0.13.