Computational Biology HW2: Genome

Yifeng Tao

January 16, 2016

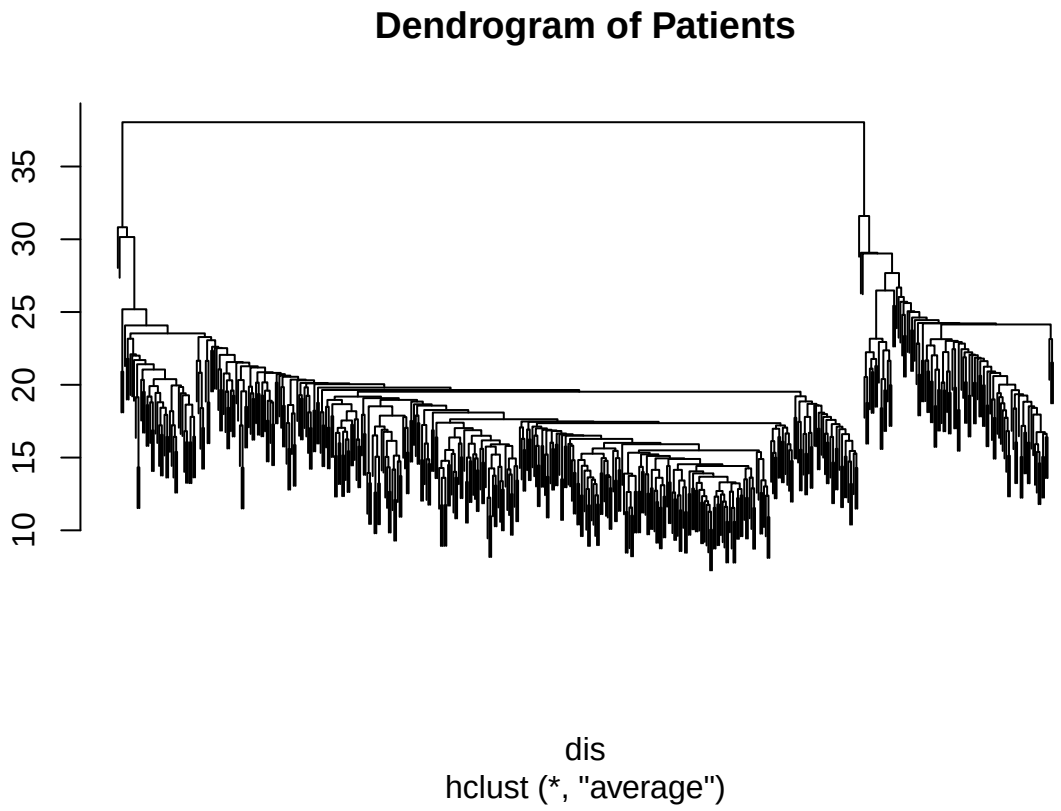
1 TCGA Data Analysis

1.1 Hierarchical Clustering

First of all, we read into the data and apply hierarchical clustering to genes:

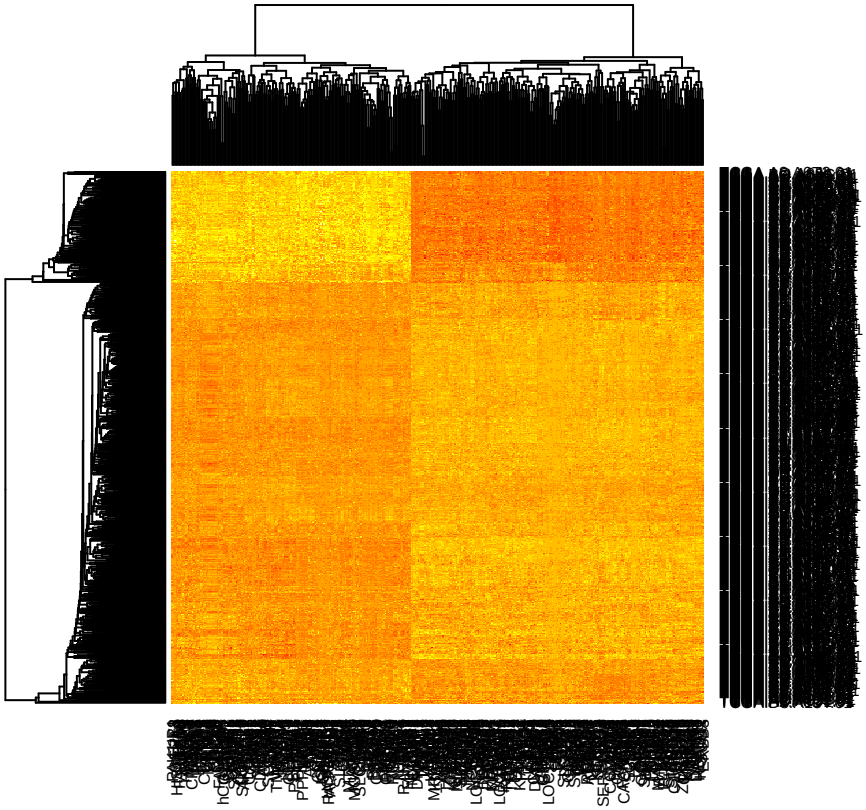
```
# Read in data
gene = read.table("GeneMatrix.txt",header = TRUE, sep = "\t", row.names = 1)
# Row: patient, Col: gene
gene = t(gene)

# Hierarchical clustering
dis = dist(gene)
tree = hclust(dis, method = "average") # distance using average
plot(tree, main = "Dendrogram of Patients", xlab = NULL, ylab = NULL, labels = FALSE, sub = NULL)
```



Then, plot the heatmap of genes and patients:

```r
# Plot heatmap
dendcompletem = as.dendrogram(tree)
# Note: Genes have already been clustered.
heatmap(gene, Rowv=dendcompletem, Colv=NULL, scale="none", xlab = "Gene", ylab = "Patient")
```



Note: From the heatmap, we can see that the patients are generally classified into two categories. They may be related to the ER status. We do not cluster the genes because it seems that they are well classified already.

We read into the clinical data and compute the accuracy.

```r
# clinical data
clinical = read.table("clinical_data", header = TRUE, sep = "\t", row.names = 1)
# change - into .
clinicalID = gsub("-",".",rownames(clinical))
clinicalER = clinical$ER_Status_nature2012

# gene2 is the classified patients.
gene2 = cutree(tree, k = 2)
truenum = 0
total = 0
for (i in 1:length(gene2)) {
  j = which(clinicalID == names(gene2)[i])
  # Prevent the geneID is not in the clinical data
  if (length(j) == 1) {
    total = total + 1
    # If the gene data classify patient in the right way.
    if ((gene2[i] == 1 && clinicalER[j] == "Positive")
```

```
        || (gene2[i] == 2 && clinicalER[j] == "Negative")) {
      truenum = truenum + 1
    }
  }
}

# Accuracy
acc = truenum/total
```

One thing we need to note is that the gene data and clinical data share part of the patients.

The accuracy of classification is

```
acc
```

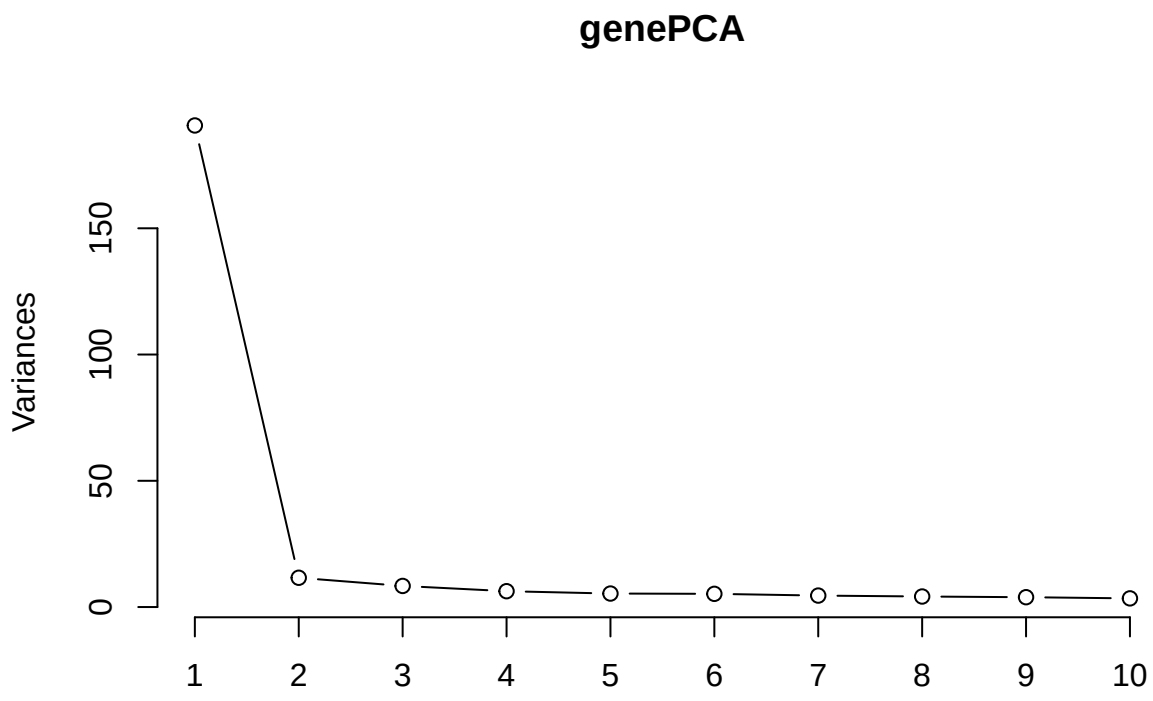
```
## [1] 0.9409091
```

which means our classification is pretty good and make sense.

1.2 PCA

We then apply PCA to the gene data and plot the top PCs:

```
genePCA = prcomp(gene, center = TRUE, scale = TRUE)
plot(genePCA, type = "l")
```

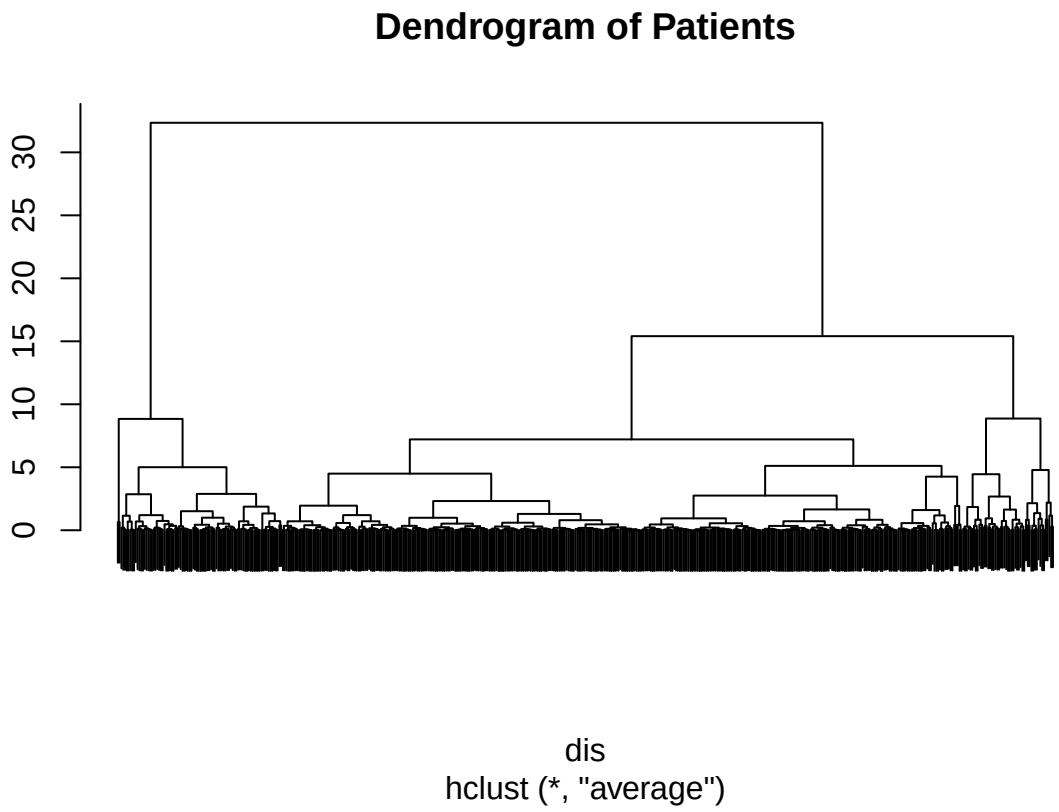


The 1st PC has such large variance that we can omit all the other PCs.

Using the 1st PC, we can get the hierarchical dendrogram:

```
# Predict PCs
pre = predict(genePCA, newdata=gene)
pre = pre[,1]

# Hierarchical clustering
dis = dist(pre)
tree = hclust(dis, method = "average") # distance using average
plot(tree, main = "Dendrogram of Patients", xlab = NULL, ylab = NULL, labels = FALSE, sub = NULL)
```



Still, the patients are classified into two categories.

```
# clinical data
clinical = read.table("clinical_data", header = TRUE, sep = "\t", row.names = 1)
# change - into .
clinicalID = gsub("-",".",rownames(clinical))
clinicalER = clinical$ER_Status_nature2012

# gene2 is the classified patients.
gene2 = cutree(tree, k = 2)
truenum = 0
total = 0
for (i in 1:length(gene2)) {
  j = which(clinicalID == names(gene2)[i])
  # Prevent the geneID is not in the clinical data
  if (length(j) == 1) {
```

```
    total = total + 1
    # If the gene data classify patient in the right way.
    if ((gene2[i] == 1 && clinicalER[j] == "Positive")
        || (gene2[i] == 2 && clinicalER[j] == "Negative")) {
      truenum = truenum + 1
    }
  }
}

# Accuracy
acc = truenum/total
```

The accuracy of classification is

```
acc
```

```
## [1] 0.9136364
```

It only decreases a little compared with the effect of using more than all the 130 genes.

2 PCA for stratificaiton in GWAS

Read into data from the paper and perform PCA:

```
mdata = matrix(c(1, 0, 2, 0, 2, 0, 2,
                 1, 1, 1, 0, 1, 0, 2,
                 1, 2, 1, 1, 1, 1, 1,
                 0, 1, 0, 2, 0, 1, 1,
                 0, 2, 1, 2, 0, 1, 0), nrow = 7, ncol = 5)
ev = eigen(cov(mdata))
```

Output the first principal component:

```
ev$vectors[,1]
```

```
## [1] -0.6383335 -0.3494259  0.1082564  0.4002980  0.5463277
```

The first PC is proportional to the ones displayed in the paper:

$$[0.7, 0.4, -0.1, -0.4, -0.5]^T$$

3 Deduction of PCA

There are two ways to deduce PCA formulation. Namely, the maximum variance or minimum error way.

I refered to Bishop's book and found it has serious **logic errors** (the proof can't be finished using induction), in addition, to achieve the minimum error goal, we do **NOT** have to choose the top eigenvectors of feature matrix (The vectors we choose only need to satisfy: They form the same subspace of top eigenvectors).

Assume we choose M direction: $u_1, ..., u_M$ in the D-dimension space, where $u_i, i \in [M]$ are orthogonal. That is: $u_i^T u_j = \delta_{ij}$.

Assume $U = [u_1, ..., u_M]$, and

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$$

.

3.1 Maximum Variance Formulation

We want to maximize:

$$J = \frac{1}{N}\sum_{n=1}^{N}||U^T x_n - U^T \bar{x}||^2$$

$$J = \frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M}||u_m^T x_n - u_m^T \bar{x}||^2 = \sum_{m=1}^{M}\frac{1}{N}\sum_{n=1}^{N}||u_m^T x_n - u_m^T \bar{x}||^2 = \sum_{m=1}^{M} u_m^T S u_m$$

Assume $r-1$ (Of course r can be equal to 1) of u_i are the top $r-1$ eigenvectors of S, and there is no u_i such that it is the top r^{th} eigenvector of S (Note: for simplicition, when I say top eigenvectors, I mean the eigenvectors with top eigenvalues of S).

For simplicition, let's say u_1 is not among the top $r-1$ eigenvectors. Using Lagrangian function:

$$L = \sum_{m=1}^{M} u_m^T S u_m - \lambda(u_1^T u_1 - 1)$$

$$\frac{dL}{du_1} = 0$$

$$\sum_{m=1}^{M} S u_m = \lambda u_1$$

Multiply u_1^T on the left, we get:

$$S u_1 = \lambda u_1$$

and

$$u_1^T S u_1 = \lambda$$

As we mentioned, u_1 is not the top r eigenvectors. However, if we set it to be the top r^{th} eigenvectors, the result will satisfy all constraints and still get a lager J. Thus, if we have: $r-1$ of u_i are the top $r-1$ eigenvectors of S. To maiximize J, there is u_i such that it is the top r^{th} eigenvector of S. In this way, using induction, all the $u_i, i \in [M]$ are top M eigenvectors of feature matrix S.

3.2 Minimum Error Formulation

We can have a complete orthogonal set of D-dimension basis vectors $\{u_i\}, i \in [D]$.

Each point can be represented as:

$$x_n = \sum_{i=1}^{D} \alpha_{ni} u_i$$

And multiply u_i^T on the left, we have:

$$x_n = \sum_{i=1}^{n} (x_n^T u_i) u_i$$

Now, we only select M of these u_i to represent all the points. We approximate them into:

$$\tilde{x_n} = \sum_{i=1}^{M} z_{ni} u_i + \sum_{i=M+1}^{D} b_i u_i$$

We want to minimize

$$J = \frac{1}{N} \sum_{n=1}^{N} ||x_n - \tilde{x_n}||^2$$

First minimize with respect to z_{ni}, use Lagrangian function, we get:

$$z_{nj} = x_n^T u_j$$

We can also minimize with respect to b_i, use Lagrangian function, we get:

$$b_j = \bar{x}^T u_j$$

Thus,

$$x_n - \tilde{x_n} = \sum_{i=M+1}^{D} ((x_n - \bar{x})^T u_i) u_i$$

So, our opimization problem become minimizing

$$J = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=M+1}^{D} (x_n^T u_i - \bar{x}^T u_i)^2 = \sum_{i=M+1}^{D} u_i^T S u_i$$

The explanation is similar to former(in section 3.1), we finally get $u_i, i = M+1, ..., D$ should be the least eigenvectors of S. Thus, if we choose top M eigenvectors of S, it will satisfy all.

However, we have to note that, $u_i, i \in [M]$ don't have to be the top M eigenvectors, they only have to satisfy that: They form the same subspace of top eigenvectors

4 EM Algorithm

EM algorithm is an iterative method for finding maximum likelihood estimates of parameters in statistical models. The iteration contains 2 steps:

- E step:

$$Q(\Theta, \Theta^{(i-1)}) = E[\log p(X, Y|\Theta)|X, \Theta^{(i-1)}]$$

- M step:

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)})$$

4.1 EM Algorithm of GMM

Follow the steps of general EM algorithm, we can get the EM algorithm for GMM.

First, initialize means μ_l, covariance $\sum_l$ and mixing coefficients α_l(They are denoted as Θ), and evaluate the initial value of log likelihood.

Then, iterate:

- E step:

$$Q(\Theta, \Theta^{(i-1)}) = \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^{(i-1)}) + \sum_{l=1}^{M}\sum_{i=1}^{N} \log(p_l(x_i|\theta_l)) p(l|x_i, \Theta^{(i-1)})$$

- M step:

$$\alpha_l^i = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i, \Theta^{i-1})$$

$$\mu_l^i = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^{i-1})}{\sum_{i=1}^{N} p(l|x_i, \Theta^{i-1})}$$

$$\Sigma_l^{(i)} = \frac{\sum_{i=1}^{N} p(l|x_i, \Theta^{i-1})(x_i - \mu_l^{(i)})(x_i - \mu_l^{(i)})^T}{\sum_{i=1}^{N} p(l|x_i, \Theta^{i-1})}$$

4.2 EM of GMM vs. K-means

The two algorithm are quite similar to each other in that: They both iterate, and at each iteration, EM of GMM performs a soft assignment of data points to clusters, while K-means performs a hard one.

In addition, K-means is a particular limit of EM of GMM:

- The covariance of mixture components are ϵI.
- We make $\epsilon \to 0$.

References

[1] https://www.biostars.org/p/14156/

[2] http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/

[3] Bishop et al. Pattern Recognition and Machine Learning. Springer Science+Business. 2006

[4] Jeff Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.1998