

## 5.1 Minimizing the maximum intracluster distance

### 5.1.1 Introduction

There are many variations of the clustering problem. One of the main differences among these problems is in their objective function. Most of the common clustering algorithms are heuristic algorithms that very little has been proven about their merits. In this section we will discuss the clustering problem of minimizing the maximum intracluster distance and present two algorithms for it. The advantage of this objective is that it has a provable polynomial approximation algorithm, when the input satisfies the triangle inequality.

Formally the problem of minimizing the maximum intracluster distance is defined as follows:

Given a weighted undirected graph  $G = (V, E, W)$  with edge weight function  $W : E \rightarrow \mathbb{R}^+$ , and an integer  $k$ , partition  $V$  into  $k$  sets,  $(B_1, \dots, B_K)$  so that  $\max_k \{w(i, j) | i, j \in B_k\}$  is minimum.

### 5.1.2 FPF

FPF (Furthest Point First) is a polynomial algorithm for the above clustering objective introduced by Gonzalez [6]. When the weight function satisfies the triangle inequality, it guarantees a 2-approximation. The FPF works in iterations. In each iteration it keeps a subset of nodes called heads and partitions  $V$  into clusters according to their distance from the heads. In each iteration the algorithm will choose the furthest point from the current set of heads as the new head.

#### FPF Algorithm:

- Initialization:
  1. Pick an arbitrary point, mark it as  $head_1$  - the head of cluster 1.
  2. Assign all points to cluster 1.
- Iteration  $i$  ( $i = 2, \dots, k$ )

1. Pick the furthest point from the current set of heads.
2. Designate it  $head_i$ .
3. Move to cluster  $i$  every point closer to  $head_i$  than to its current head.

See Figure 5.1 a-c for illustration.

**Complexity:** At the beginning of each iteration we have the distance of every point from its head. We search for the maximum distance in  $O(n)$  to find the new head. Then we compare for each point's the distance from its current head to the distance from the new head and update the point head if necessary. Therefore there is  $O(n)$  work in each iteration. Since there are  $k$  iterations the time complexity of FPF is  $O(kn)$ .

**Theorem 5.1** *FPF guarantees a 2-approximation if the weight function satisfies the triangle inequality.*

**Proof:** Let  $h^i(x)$  be the head of the cluster containing  $x$  in iteration  $i$ . Let  $x^*$  denotes the furthest point from its head when the algorithm terminates, and let  $\Delta \equiv w(x^*, h^k(x^*))$ .

Observe that after each iteration the distance between any point and its head can only become smaller:  $w(x, h^i(x)) \geq w(x, h^{i+1}(x))$ . In particular it is true for  $x^*$ . Therefore for every  $i$ ,  $w(x^*, h^i(x^*)) \geq \Delta$ . Also observe that since  $x^*$  was never chosen as head, the distance of all heads from each other is at least  $\Delta$ . From the two observations we get that when the algorithm ends there are at least  $k + 1$  points that are more than  $\Delta$  apart (the  $k$  final heads and  $x^*$ ). Therefore in any way we will partition  $V$  into  $k$  sets we will have at least one cluster that contains two points that are more than  $\Delta$  apart. This implies that  $OPT(G) \geq \Delta$ .

On the other hand, the maximum distance between a point and its head after the algorithm terminates is  $\Delta$ . Therefore according to the triangle inequality the distance between any two points in any cluster is at most  $2\Delta$ . Therefore  $FPF(G) \leq 2\Delta$ , hence the 2-approximation ratio (see Figure 5.1 d-f). ■

Gonzalez actually shows that this approximation ratio is tight:

**Theorem 5.2** *(Gonzalez, 1985 [6])  $2 - \varepsilon$  approximation is NP-hard for all  $\varepsilon > 0$  when the weights are distances between points in a three dimensional space.*

### 5.1.3 K-Boost

K-Boost is a clustering algorithm that uses the FPF idea combined with stability based method for determining a plausible number of clusters [4]. K-Boost runs FPF with several values of  $k$  and estimates the predictive strength of each partition.

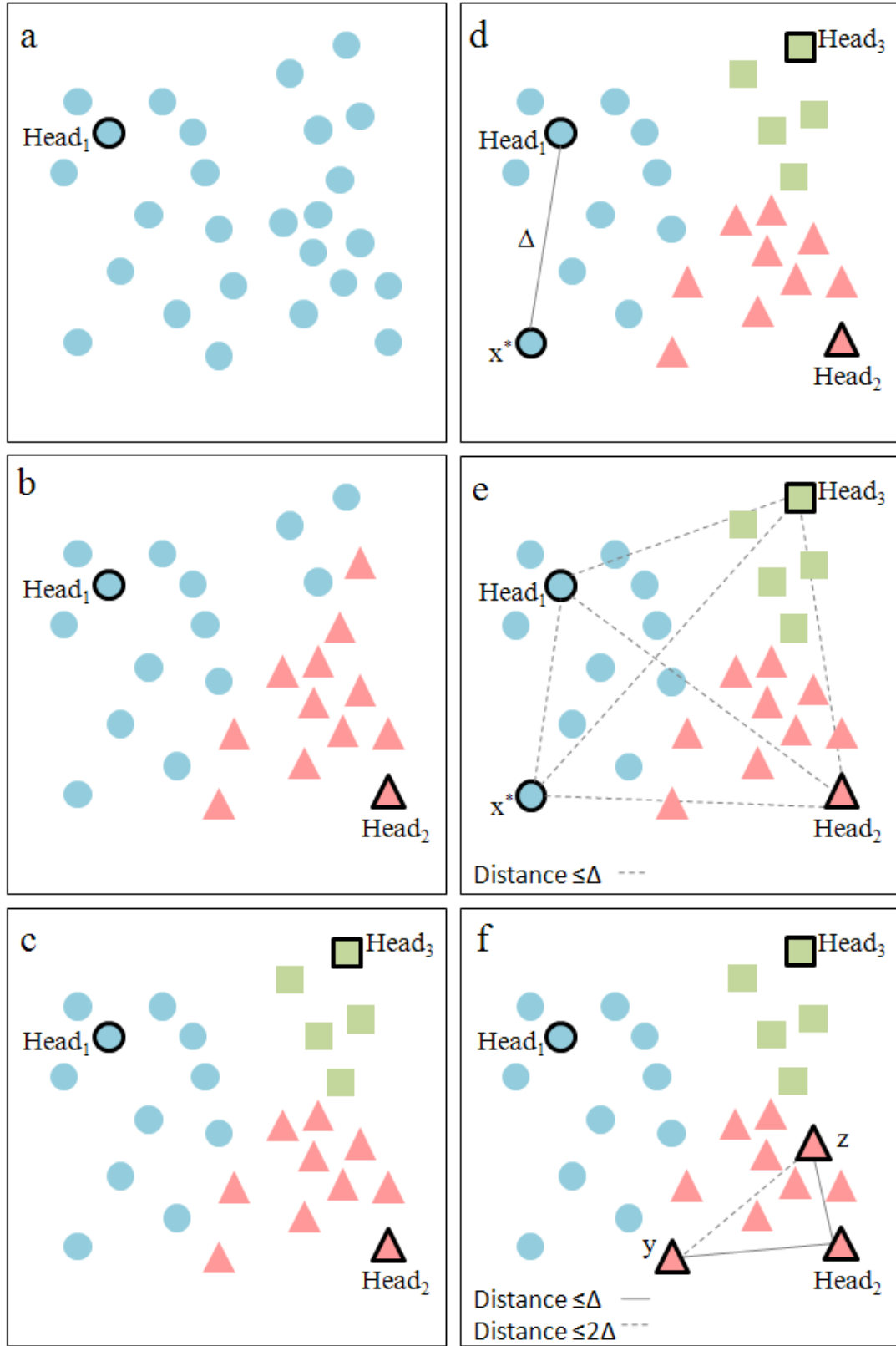


Figure 5.1: Illustration of the FPF algorithm with  $k = 3$ , and its proof.

(a-c) Three iterations in the FPF algorithm. (d)  $x^*$  is the furthest point from its head when the algorithm terminates. Its distance from its head is  $\Delta$ . (e)  $head_1$ ,  $head_2$ ,  $head_3$  and  $x^*$  are more than  $\Delta$  apart from each other. Therefore  $OPT(G) \geq \Delta$ . (f) For any two points  $y$ ,  $z$  that have the same head,  $head_2$ , the distance between  $y$  and  $z$  to  $head_2$  is at most  $\Delta$ . Therefore the distance between  $y$  and  $z$  is smaller than  $2\Delta$ . Hence  $FPF(G) \leq 2\Delta$ .

### Weight Function

The Pearson Coefficient is a very popular measure of similarity in the context of gene expression microarray data clustering. Pearson Coefficient,  $P(p_i, p_j)$ , given by

$$P(p_i, p_j) = \frac{\sum_{t=1}^m (p_{i,t} - \mu_i)(p_{j,t} - \mu_j)}{\sqrt{(\sum_{t=1}^m (p_{i,t} - \mu_i)^2)(\sum_{t=1}^m (p_{j,t} - \mu_j)^2)}}$$

where  $\mu_i$  and  $\mu_j$  are the means of  $p_i$  and  $p_j$ , respectively. However, since Pearson correlation is not a distance function, it can not be used within algorithms that assume the triangle inequality like the FPF algorithm. To overcome this problem, K-Boost uses  $d(p_i, p_j) \equiv \sqrt{1 - P(p_i, p_j)}$  as distance measure. We will show that  $d(p_i, p_j)$  is proportional to the Euclidean distance between  $p_i$  and  $p_j$ . Let's define  $q_i \equiv (p_i - \mu_i)/\sigma_i$ . Notice that  $\|q_i\| = 1$  and  $P(p_i, p_j) = q_i * q_j$ . Hence:

$$\|q_i - q_j\|^2 = q_i * q_i + q_j * q_j - 2q_i * q_j = 2(1 - P(p_i, p_j))$$

Therefore after normalizing  $d(p_i, p_j)$  is  $\frac{1}{\sqrt{2}}$  times the Euclidean distance between  $p_i$  and  $p_j$ .

### Speeding-up The FPF Algorithm

K-Boost calls the FPF algorithm several times. To improve running time unnecessary comparisons should be avoided. When  $head_i$  is added to the group of heads, the FPF algorithm checks for every point if it is closer to  $head_i$  than to its current head. To avoid unnecessary comparisons all the points associated with a specific head are ordered according to their distance from it. When checking points in cluster  $j$  we will scan them in decreasing order from  $head_j$ . We will stop checking when we reach a point  $p$  that satisfies  $d(p, head_j) \leq \frac{1}{2}d(head_j, head_i)$ . The triangle inequality guarantees that  $d(p, head_j) \leq d(p, head_i)$ . This improvement does not change the worst case time complexity but can save time in practice.

### Using Prediction Strength To Determine $k$

To obtain estimation of  $k$  a stability-based method suggested by Tibshirani at el. [11] is used. The method uses a random subset  $S$  of  $V$ . For each  $k$  the subsets  $S$  and  $V - S$  are clustered separately and the resulting partitions are compared. Assuming that when the correct value of  $k$  is used the two partitions will be similar.

### Prediction Strength Procedure:

- Initialization:
  1. Sample  $S \subset V$ , define  $T \equiv V - S$ .
- Iteration  $i$  ( $i = 1, \dots, n$ )

1. Partition  $S$  into  $i$  sets  $B = (B_1, \dots, B_i)$ .
2. Partition  $T$  into  $i$  sets  $C = (C_1, \dots, C_i)$ .
3. Use the heads of the clusters  $(C_1, \dots, C_i)$  to partition  $T$  again into  $i$  sets  $C' = (C'_1, \dots, C'_i)$ .
4. Compute for each cluster in  $C$  the fraction of its genes that remain mates in partition  $C'$ .  $PS(i)$  is the minimum fraction among all clusters. More formally:  $PS(i) = \min_{1 \leq j \leq i} PS^j(i)$  where

$$PS^j(i) = \frac{1}{\binom{|C_j|}{2}} |\{(x, y) | (x, y) \in C_j \text{ and } \exists z \text{ s.t. } (x, y) \in C'_z\}|$$

- $k = \arg \max_{i > 1} \{PS(i)\}$

See Figure 5.2 for an illustration of the prediction strength procedure.

K-Boost uses the prediction strength algorithm with two differences:

- Instead of choosing  $k$  as the global maximum of  $PS(i)$ ,  $k$  is the first local maximum of  $PS(i)$  different from  $k = 1$ . This heuristic is supported by experimental results (see Figure 5.3).
- To improve the robustness of the solution two fold validation is used to estimate  $k$ .

#### **K-Boost Algorithm First Phase (finding $k$ ):**

- Initialization:
  1. Partition  $V$  to three disjoint groups  $S_L$ ,  $S_R$  and  $T = V - (S_L \cup S_R)$  so that  $|S_L| = |S_R| = \eta = \sqrt{n}$ .
- Iteration  $i$  ( $i = 1, \dots, \eta$ )
  1. Partition  $S_L$  into  $i$  sets  $B = (B_1, \dots, B_i)$  using FPF.
  2. Partition  $T$  into  $i$  sets  $C = (C_1, \dots, C_i)$  using FPF.
  3. Use the heads of  $C$  to partition  $T$  again into  $i$  sets  $C' = (C'_1, \dots, C'_i)$
  4. Compute for each cluster in  $C$  the fraction of its genes that remain mates in partition  $C'$ .  $PS(i)$  is the minimum fraction among all clusters.
  5. Stop when  $PS(i - 2) < PS(i - 1) > PS(i)$  and assign  $k_L = i - 1$ .
- Perform the iterations on  $S_R$  and calculate  $k_R$ .
- $k = \lceil \text{average}(k_L, k_R) \rceil$ .

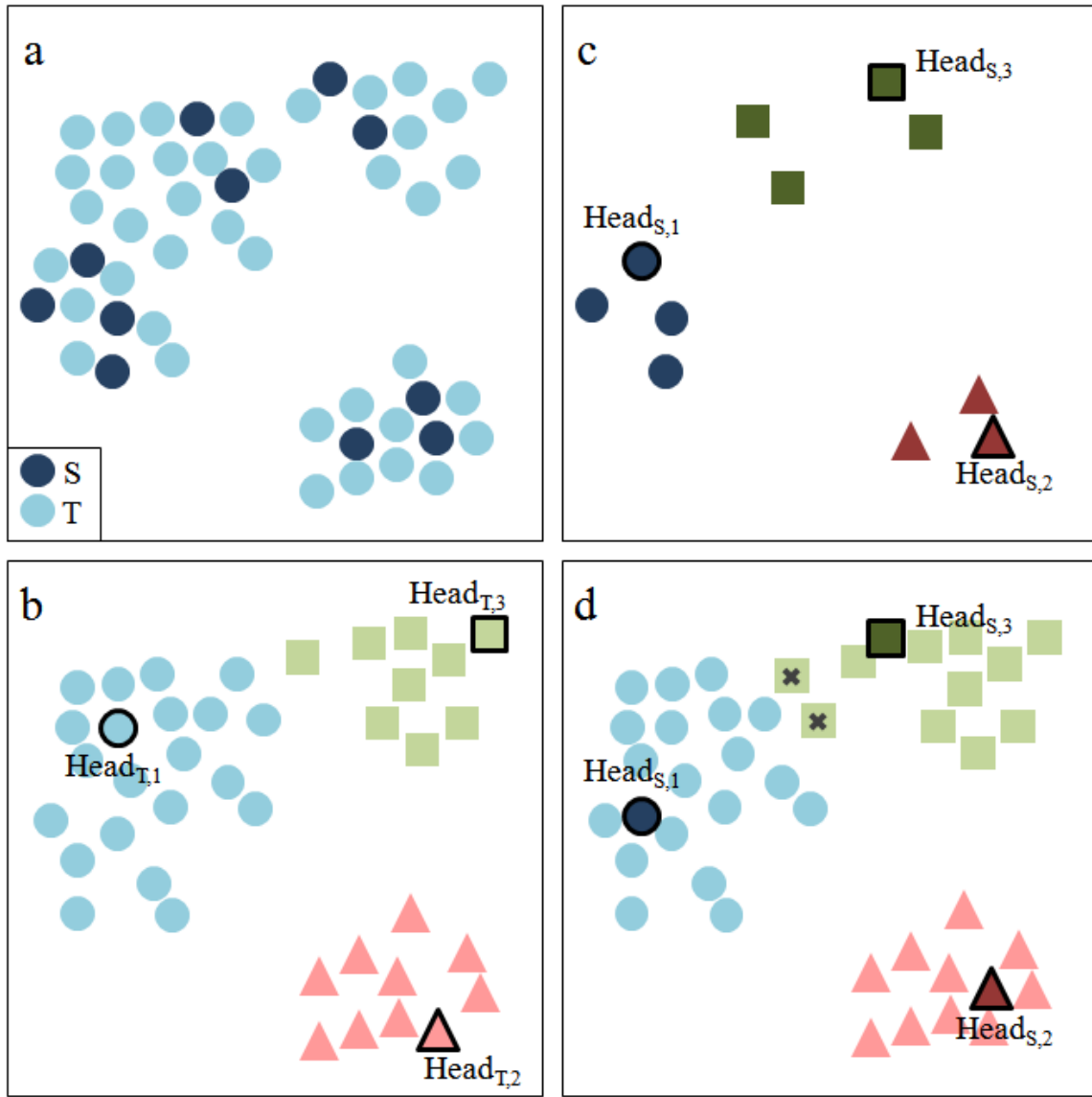


Figure 5.2: Illustration of the prediction strength procedure on the FPF algorithm. (a) Randomly partition  $V$  to  $S$  and  $T$ . (b) Partition  $T$  to three groups using FPF. (c) Partition  $S$  to three groups using FPF. (d) Use the heads obtained in the partition of  $S$  to divide  $T$  again. For each of  $T$ 's original clusters the fraction of genes that remain mates in the second partition of  $T$  is computed. The measure of predictive strength is the minimum fraction among all clusters. Notice that both partitions of  $T$  are the same except for 2 genes (marked with a x). Therefore the predictive strength is determined according to the cluster 1 and equals to  $PS(3) = \frac{\binom{19}{2} + 1}{\binom{21}{2}}$ .

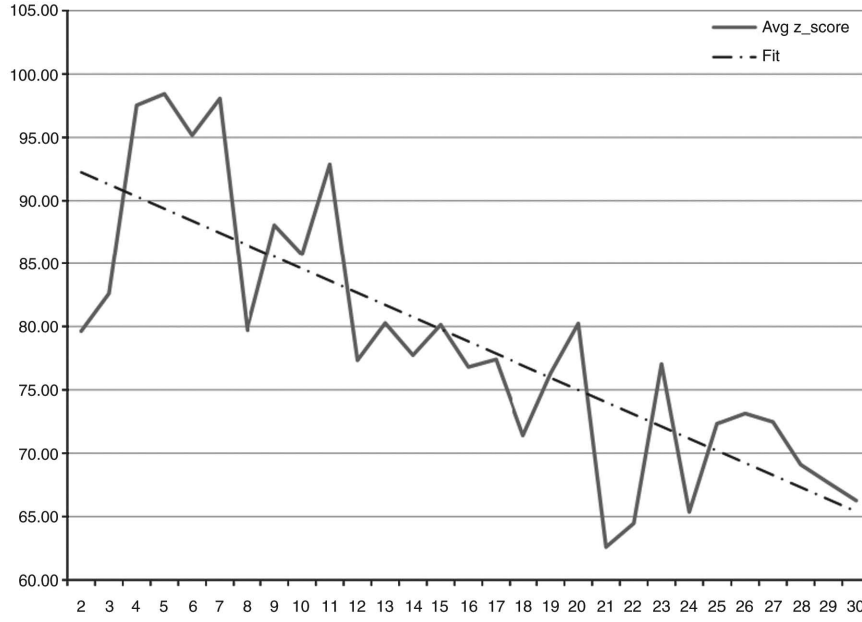


Figure 5.3: Plotting z-score as a function of  $k$  for the Cho et al. data set [1]. One can notice a decreasing trend after the first peak. Notice that here z-score and not  $PS$  is used as the evaluation criterion. Source: [4].

#### K-Boost Algorithm Second Phase (clustering):

- Start with the partition  $C = (C_1, \dots, C_k)$  and corresponding heads  $head_1, \dots, head_k$ .
- For each  $C_i$ , rank its genes in increasing distance from  $head_i$ . Order all genes by interleaving the ranks.
- Add the rest of the genes to the partition according to that order, assigning each gene to the cluster with the closest centroid. When adding a gene to a cluster update its centroid.

**Complexity:** It can be shown that the time complexity of K-Boost is  $O(knm)$ , where  $n$  is the number of genes and  $m$  is the number of conditions.

#### Algorithm Performance Comparisons:

K-Boost results were compared with CLICK and FPF-SB, an earlier version of the K-boost [3]. All three algorithms are able to determine a plausible number of clusters. Furthermore, since FPF-SB, CLICK, and K-Boost usually suggest a different number of clusters, the robustness of their proposed number of clusters was evaluated by feeding them to three other clustering algorithms: FPF, HAC [2] and k-means. (They were chosen as popular clustering

algorithms for microarray gene expression data that require  $k$  as an input parameter.)

The results were evaluated by both internal and external measures of quality. Homogeneity and separation were used as internal measures, while  $z_{score}$  [5], which measures the homogeneity of the gene annotations within each cluster, served as an external measure. Higher values of homogeneity and  $z_{score}$  and lower values of separation indicate higher quality of the clustering.

The algorithms were tested on three well-studied yeast data sets. The first is the yeast cell cycle data set described in Cho et al. [1]. The second data set, described in Spellman et al. [10], is a catalog of yeast expression profiles under various cell cycle conditions. The third data set, described in Eisen et al. [2] consists of an aggregation of expression profiles from experiments on the budding yeast *Saccharomyces cerevisiae* (including time courses of the mitotic cell division cycle, sporulation, and the diauxic shift).

The results of comparing K-Boost with CLICK and FPF-SB are presented in Table 5.1. K-Boost achieves a significantly better  $z_{score}$  on all the three yeast data sets using less computation time. The results of using the  $k$  computed by FPF-SB, CLICK and K-Boost as an input for FPF, HAC, and k-means is presented in Tables 5.2, 5.4 and 5.3. FPF, HAC, and k-means always attain significantly better performance in terms of  $z_{score}$  and separation when fed with K-Boosts estimate of  $k$ ; on the other hand, Clicks estimates leads to better homogeneity figures.

Clustering is still an open and active research area, and there is no universally recommended method of choice. Since there is no algorithm that gets better result in all quality measures, one might prefer methods that are stronger in separation or those that are stronger in homogeneity and  $z_{score}$ . Notice that all the quality measures used to estimate the clustering results are global measures. However, focused biological analysis is usually done over a small fraction of the resulting clusters. A quality measure that evaluates clustering method according to the few top scoring clusters might estimate the algorithms abilities better.

## 5.2 Principal Components Analysis

### 5.2.1 Data Reduction

High dimensional data could be difficult to analyze. Data reduction is a procedure that reduces a matrix  $X_{p \times n}$  to a new matrix  $Y_{k \times n}$  that is a more compact representation of  $X$ .

Method	Cho et al.				Eisen et al.				Spellman et al.			
	$k$	$Sg$	$T$	$z_{score}$	$k$	$Sg$	$T$	$z_{score}$	$k$	$Sg$	$T$	$z_{score}$
CLICK	30	136	540	62.47	8	0	165	42.26	27	17	3000	73.93
FBF-SB	14	0	16	61.93	12	0	19	57.90	16	0	59	58.20
K-Boost	10	0	21	<b>79.00</b>	8	0	23	<b>69.03</b>	19	0	134	<b>78.60</b>

Table 5.1: Comparison of three clustering algorithms on three yeast data sets. For each algorithm and data set the table presents:  $k$  - the number of clusters,  $Sg$  - the number of singleton data points (unclustered elements),  $T$  - the running time in seconds, and the  $z_{score}$  computed by ClusterJudge. The results shown for  $z_{score}$  are the average of three independent runs. Source: [4].

Method	FBF-SB estimate					CLICK estimate					K-Boost estimate				
	$k$	$T$	$z_{score}$	$Hom$	$Sep$	$k$	$T$	$z_{score}$	$Hom$	$Sep$	$k$	$T$	$z_{score}$	$Hom$	$Sep$
FPF	14	5	56.5	0.572	0.011	30	10	52.43	<b>0.645</b>	0.016	10	4	<b>60.63</b>	0.548	<b>-0.025</b>
HAC	14	103	53.1	0.517	-0.143	30	103	56.57	<b>0.617</b>	-0.059	10	102	<b>61.00</b>	0.511	<b>-0.151</b>
k-means	14	18	74.6	0.655	-0.035	30	38	67.80	<b>0.703</b>	-0.001	10	10	<b>95.33</b>	0.631	<b>-0.056</b>

Table 5.2: Experimental results on Cho et al. data set. Comparing algorithms that take  $k$  as input with the values computed by FPF-SB, CLICK, and K-Boost. Source: [4].

Method	FBF-SB estimate					CLICK and K-Boost estimate				
	$k$	$T$	$z_{score}$	$Hom$	$Sep$	$k$	$T$	$z_{score}$	$Hom$	$Sep$
FPF	12	6	53.4	0.483	0.079	8	4	<b>56.87</b>	<b>0.524</b>	<b>-0.076</b>
HAC	12	7	34.3	0.440	0.042	8	7	<b>37.10</b>	0.439	<b>-0.292</b>
k-means	12	14	62.3	0.528	0.102	8	10	<b>64.86</b>	<b>0.572</b>	<b>-0.021</b>

Table 5.3: Experimental results on Eisen et al. data set. Comparing algorithms that take  $k$  as input with the values computed by FPF-SB, CLICK, and K-Boost. Source: [4].

Method	FBF-SB estimate					CLICK estimate					K-Boost estimate				
	$k$	$T$	$z_{score}$	$Hom$	$Sep$	$k$	$T$	$z_{score}$	$Hom$	$Sep$	$k$	$T$	$z_{score}$	$Hom$	$Sep$
FPF	16	19	62.2	0.456	0.188	27	32	46.16	<b>0.489</b>	0.066	19	22	<b>62.47</b>	0.481	<b>0.054</b>
HAC	16	92	55.8	0.420	0.176	27	92	56.63	<b>0.463</b>	-0.018	19	92	<b>57.00</b>	0.448	<b>-0.017</b>
k-means	16	83	80.10	0.507	0.149	27	130	79.66	<b>0.559</b>	0.047	19	75	<b>81.07</b>	0.538	<b>0.035</b>

Table 5.4: Experimental results on Spellman et al. data set. Comparing algorithms that take  $k$  as input with the values computed by FPF-SB, CLICK, and K-Boost. Source: [4].

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix}_{p \times n} \rightarrow \begin{pmatrix} Y_1 \\ \vdots \\ Y_k \end{pmatrix}_{k \times n}$$

Data reduction should balance two objectives:

- Reduce the number of dimensions so the resulting data will be easier to understand and general trends will be easier to distinguish.
- Minimize the loss of important data.

### 5.2.2 PCA

PCA (Principal Components Analysis) is a popular data reduction method invented by Pesrson [9] and Hotelling [7]. To understand the objective of the PCA one should be familiar with the concept of variance and covariance.

Given  $n$  samples each consisting of  $p$  variables, the variance of each variable is estimated as the squared deviation of the sample values from the variable mean. When the variable mean is unknown it is estimated by the variable average. Therefore the the variance of variable  $X_i$  is estimated as:

$$V_i = \frac{1}{n-1} \sum_{m=1}^n (X_{im} - \bar{X}_i)^2$$

where  $\bar{X}_i$  is the average of variable  $X_i$ . Since the mean of  $X_i$  is unknown and estimated by its average the sum is divided by  $n-1$  and not by  $n$ , to get unbiased estimator of the variance. The covariance of two variables  $i$  and  $j$  is the degree which those variables are linearly correlated. Formally the covariance of variable  $i$  and  $j$  is estimated as:

$$C_{ij} = \frac{1}{n-1} \sum_{m=1}^n (X_{im} - \bar{X}_i)(X_{jm} - \bar{X}_j)$$

When the covariance of two variable is zero we say that these variables are uncorrelated. (Again,  $n$  is replaced by  $n-1$  to avoid biased estimation).

PCA is a mathematical procedure that given  $n$  samples with  $p$  correlated variables uses a linear transformation to generate uncorrelated variables called principal components. The

principal components are ordered by their variability. So by taking the first  $k$  principal components we will obtain the maximal variation possible by  $k$  components.

### PCA geometrical meaning:

Given  $n$  points in a  $p$ -dimensional space PCA will perform two affine transformations:

1. Translation of the coordinate system so the average of each dimension is 0.
2. Rigid rotation of the axes so their new positions (principal axes) will have the following properties:
  - Principal axis 1 has the highest variance, axis 2 has the next highest variance, .... and axis  $p$  has the lowest variance
  - Covariance among each pair of the principal axes is zero (The principal axes are uncorrelated).

PCA uses Euclidean distance calculated from the  $p$  variables as the measure of dissimilarity among the  $n$  objects. The first principal component is the direction of the maximum variance of the  $n$  objects in the  $p$ -dimensional space. The second principal component is in the direction of the next highest variance subject to the constraint that it has zero covariance with the first principal component. The  $p$  principal component has the lowest variance and it has zero covariance with all the previous  $p - 1$  principal components. When taking the first  $k$  principal component they define  $k$ -dimensional hyperplane that represents the maximal possible variance that could be achieved in  $k$  dimensions. A 2-dimensional illustration of the PCA procedure is presented in Figure 5.4.

### 5.2.3 PCA Algorithm

The principal components are determined by eigenvectors of the covariance matrix. The covariance matrix is a positive semi-definite matrix whose  $(i, j)$  entry is:

$$\Sigma_{ij} = \begin{cases} V_i & \text{if } (i = j) \\ C_{ij} & \text{otherwise} \end{cases}$$

Notice that  $\Sigma_{p \times p} = \frac{1}{n-1} \hat{X} \hat{X}^T$  where

$$\hat{X} = \begin{pmatrix} X_1 - E(X_1) \\ X_2 - E(X_2) \\ \vdots \\ X_p - E(X_p) \end{pmatrix}_{p \times n}$$

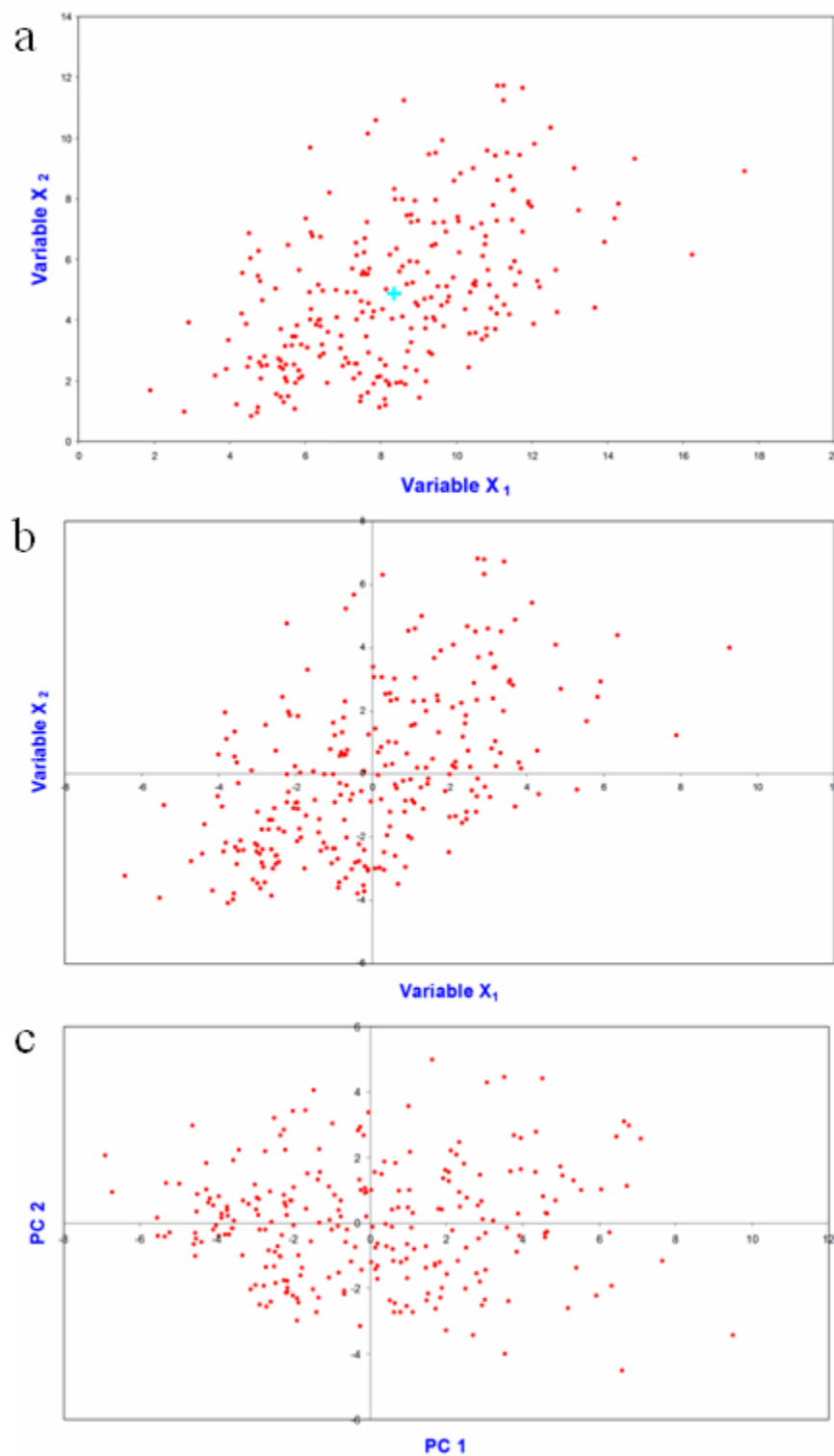


Figure 5.4: Illustration of the PCA procedure on 2-dimensional data.

(a) Variables  $X_1$  and  $X_2$  have the following properties:  $\bar{X}_1 = 8.35$ ,  $\bar{X}_2 = 4.91$ ,  $V_1 = 6.67$ ,  $V_2 = 6.24$  and  $C_{1,2} = 3.42$ . (b) Each variable is adjusted to a mean of zero (by subtracting the mean from each value). (c) Rigid rotation of the axes creates the principal components. The highest variance is along the direction of the first principal component (9.88). The variance along the second principal component is lower (3.03) and the covariance of the two principal components is zero. The first principal component is also the direction of the least-squares regression line (the squared distance of the points from the first principal component is minimal). Source: <http://www.plantbiology.siu.edu/PLB444/PCA.ppt>.

Since  $\Sigma$  is a symmetric matrix it could diagonalized as follows:

$$\Sigma = UDU^T$$

Where  $D = \text{Diagonal}(\lambda_1, \dots, \lambda_p)$  is a diagonal matrix of the eigenvalues, and  $U$  is orthonormal matrix ( $U$ 's columns are an orthonormal basis). Furthermore  $U$ 's columns are the eigenvectors of  $\Sigma$ . We will order the columns of  $U$  according to their respective eigenvalues and denote them as  $U_1, \dots, U_p$ , so the eigenvalue of  $U_1$  is the largest.

The first principal component is  $U_1$ , the second principal component is  $U_2$ , and so on.  $U_1 \dots U_p$  satisfy the following requirements:

- All of these vectors are normalized ( $U_i^T U_i = 1$  since the columns of  $U$  are orthonormal basis).
- $\hat{X}$  in the direction of  $U_i$  is orthogonal to  $\hat{X}$  in the direction of  $U_j$  for each  $i \neq j$ . Since the columns of  $U$  are orthonormal basis  $U_i^T U_j = 0$ . Therefore:

$$(U_i^T \hat{X})(U_j^T \hat{X})^T = U_i^T \hat{X} \hat{X}^T U_j = U_i^T \Sigma U_j = U_i^T \lambda_j U_j = 0$$

- The variance of  $\hat{X}$  in the direction of  $U_i$  is  $\sqrt{\lambda_i}$  since:

$$\| U_i^T \hat{X} \|^2 = (U_i^T \hat{X})(U_i^T \hat{X})^T = U_i^T \hat{X} \hat{X}^T U_i = U_i^T \Sigma U_i = U_i^T \lambda_i U_i = \lambda_i$$

Therefore variance of  $\hat{X}$  along the direction of  $U_1 \dots U_p$  is decreasing.

To translate the original data to the  $k$ -dimension space spanned by the first  $k$  principal components use the following multiplication:

$$\begin{pmatrix} U_1^T \\ \vdots \\ U_k^T \end{pmatrix}_{k \times p} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix}_{p \times n} \rightarrow \begin{pmatrix} Y_1 \\ \vdots \\ Y_k \end{pmatrix}_{k \times n}$$

### Preprocessing Data:

Using covariances among variables only makes sense if they are measured in the same units. Even then, variables with high variances will dominate the principal components. These problems are generally avoided by standardizing each variable to unit variance and zero mean before performing the PCA procedure.

$$X'_{im} = \frac{X_{im} - \bar{X}_i}{\sqrt{V_i}}$$

### **5.2.4 Examples**

PCA is a useful method to visually validate clustering results (as illustrated in Figure 5.5). PCA could also help to visualize the difference between samples (as illustrated in Figure 5.6).

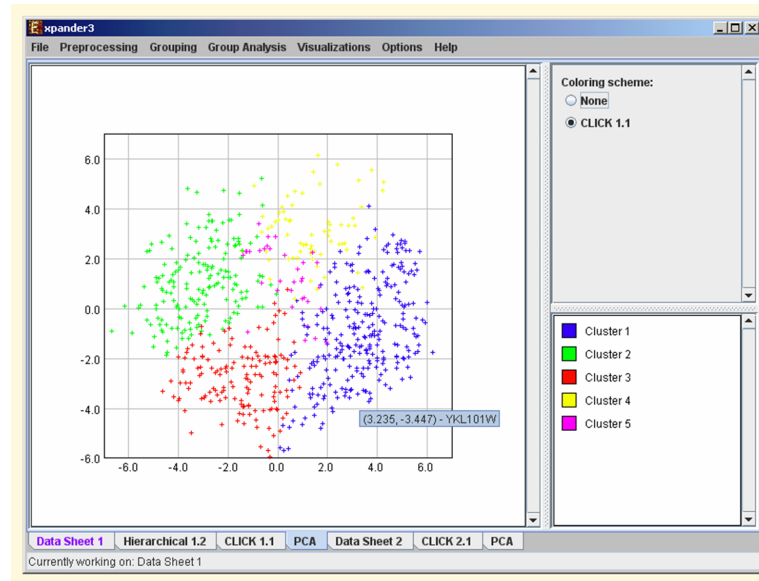


Figure 5.5: PCA results support clustering results done by CLICK. The different clusters (shown by different colors of points) are located on different areas with very little overlaps. Source: <http://acgt.cs.tau.ac.il/expander/screenshots.html>.

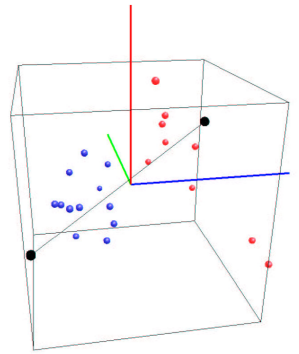


Figure 5.6: Gene expression of 21 samples using 7,913 genes was reduced to 3-dimensions using PCA. Red points: good prognosis (upper right), Blue points: bad prognosis (lower left). It's highly noticeable that the good prognosis samples were separated from the bad prognosis samples. This result could in principle be used in predicting the outcome of other samples. Source: [8].



# Bibliography

- [1] R.J. Cho et al. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol Cell*, 2:65–73, 1998.
- [2] MB. Eisen, PT. Spellman, PO. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science USA*, 95:14863–14868, 1998.
- [3] F. Geraci, M. Leoncini, M. Montangero, M. Pellegrini, and M. E. Renda. Fpf-sb: a scalable algorithm for microarray gene expression data clustering. *Lect. Notes Comput. Sci.*, 4561:606–615, 2007.
- [4] F. Geraci, M. Leoncini, M. Montangero, M. Pellegrini, and M. E. Renda. K-boost: A scalable algorithm for high-quality clustering of microarray gene expression data. *Journal of Computational Biology*, 16:859–873, 2009.
- [5] F. Gibbons and F. Roth. Judging the quality of gene expression-based clustering methods using gene annotation. *Genome Res*, 12:1574–1581, 2000.
- [6] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Comput. Sci.*, 38:293–306, 1985.
- [7] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.
- [8] K. Nishimura, K. Abe, S. Ishikawa, S. Tsutsumi, K. Hirota, H. Aburatani, and M. Hirose. Analysis of a complex of statistical variables into principal components. *Genome Informatics*, 14:346–347, 2003.
- [9] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 12:559–572, 1901.
- [10] PT. Spellman et al. Comprehensive identification of cell cycle regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. *Mol Biol Cell*, 9:3273–3297, 1998.

- [11] R. Tibshirani, G. Walther, and D. Botstein et al. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics*, 14:511–528, 2005.