Turku PET Centre Modelling report TPCMOD0029 2004-08-19
Appendix A
Marketta Hiissa

Pseudo-code for the clustering algorithm based on a mixture model

The idea of the algorithm

The aim of the algorithm is to segment the dynamic PET data into k clusters according to its probability of belonging to each of the clusters. The algorithm is based on a mixture model and Expectation Maximization (EM) algorithm and it proceeds as follows.

1. Compute the sum image (i.e. for each pixel compute the sum of values over time frames) and define initial belonging probabilities (i.e. probabilities that a chosen TAC belongs to a chosen cluster) such that the k clusters are of the equal size. The initial belonging probabilities are set to be either zero or one.
2. Compute mean TACs for each cluster and variances for each time frame. It is assumed that the variance matrix is common to all clusters but variances for time frames could also be computed for each cluster.
3. Compute probability densities for each TAC given a cluster and compute new belonging probabilities on the basis of these probability densities.
4. Calculate the value of log likelihood based on the probability densities. If the log likelihood has increased significantly since the previous iteration, repeat the process from step 2.
5. Create the clusters by assigning each TAC to a cluster such that the belonging probabilities are maximized.

To reduce the number of needed for-loops in the following pseudo-code, we use only a two-dimensional input matrix instead of the original four-dimensional data matrix.

Pseudo-code

/* Input is a matrix A with m rows (voxels) and n columns (time frames). Element in place (i, j) is denoted by a_{ij} and k is the number of clusters. */

/* Define the initial belonging probabilities. The result is a $m \times t$ matrix P and element p_{it} in place (i,t) is the probability that voxel i belongs to cluster t. First all elements of the matrix P are set to be zero and then on each row one element is set to be one to create k clusters of the equal size. */

```
for i  = 1 to m
    for t  = 1 to k
        p_it  = 0
    endfor
endfor

b  = 1
t  = 1
while  t ≤ k
        e  = t * m / k     // the result must be an integer
        for i  = b to e
            p_it  = 1
```

```
        endfor
        b  = e + 1
        t  = t +1
    endwhile
```

/* Compute the integral image by summing elements in each row. The result is a vector constituting of *m* elements b_i. */

```
    for i  = 1 to m
        bi  = 0
            while  j ≤ n
                bi  = bi + aij
                j = j + 1
            endwhile
    endfor


old_llh = 0
changed  = true

while  changed = = true
```

/* Compute a weighted mean for expression a_{ij} / b_i for each time frame in each cluster. The weighting factor is $p_{it} * b_i^2$. The result is an $n \times t$ matrix X. */

```
        for j = 1 to n
            for t  = 1 to k
                num  = 0   //  the numerator of the expression for the mean
                denom  = 0  // the denominator of the expression for the mean
                i  = 1
                while  i ≤ m
                        num  = num + aij * pit * bi
                        denom  = denom + pit * bi * bi
                        i  = i + 1
                endwhile
                xjt  = num / denom
            endfor
        endfor
```

/* Compute a variance for each time frame. Variance in a time frame is common to all clusters. */

```
            for j  = 1 to n
                num  = 0
                t : = 1
                  while  t ≤ k
                        i : = 1
                        while  i  ≤ m
                                num  = num + (aij –xjt*bi)^2 * pit
                                i  = i +1
```

```
            endwhile
            t : = t +1
        endwhile
        f_j : = num / m
    endfor
```

/* Compute the sum of belonging probabilities for each cluster. This represents the number of pixels associated to each cluster. */

```
    for t  = 1 to k
        g_t  = 0
        i  =1
        while  i ≤ m
                g_t  = g_t + p_it
                i  = i + 1
        endwhile
    endfor
```

/* Compute the probability densities for each pixel and each cluster. Probabilities are based on the normal distribution and the result is a $m \times t$ matrix Q. */

```
    /* Define a common denominator for all distributions. */
    c_denom  = 1
    j = 1
    while  j ≤ n
            c_denom =  sqrt( c_denom * 2π f_j )
            j  = j +1
    endwhile

    / * Compute the values of probability densities for each voxel and each cluster. */

    for i  = 1 to m
        for t  = 1 to k
            expon  = 0
            j  = 1
            while  j ≤ n
                    expon  = expon + (a_ij −x_jt * b_i)^2  / f_j
                    j  = j + 1
            endwhile
            q_it : = ((g_t/m) / (denom) ) *  exp (-0.5 * expon )
            endfor
    endfor
```

/* Compute new belonging probabilities. */

```
    for i = 1 to m
        sum = 0
        for t = 1 to k
            r = 1
            while r ≤ k
```

```
                sum = sum + q_ir
                r = r + 1
              endwhile
              p_it  = q_it / sum
          endfor
      endfor

    /* Compute the log likelihood */

      llh : = 0
      i  := 1
      while  i ≤ m
              llh_a = 0
              t = 1
              while  t ≤ k
                      llh_a  = llh_a + q_it
                      t  = t + 1
              endwhile
              llh  = llh + log ( llh_a )
              i  = i + 1
      endwhile

  /* Make the comparisons to define whether to continue iterating or not. */

      if llh − old_llh > 0.001
         then   changed  = true
                old_llh = llh
         else changed  = false
      endif
endwhile
```

/* Create the clusters by assigning each pixel to clusters such that the belonging probability p_{it} is maximised for each pixel. */

```
for i = 1 to m
    cluster_i  = 0
    max  = 0
    t  = 1
    while  t ≤ k
            if p_it > max
              then cluster_i : = t
                   max : = p_it
            endif
            t : = t + 1
    endwhile
endfor
```

Reference

J. Ashburner et al.: A cluster analysis approach for the characterization of dynamic PET data. In Myers, R., Cunningham, V., Bailey, D., and Jones, T., editors, *Quantification of Brain Function Using PET*, pages 301–306, San Diego, 1996, Academic Press.