PSEUDO-CODE FUZZY K MEANS

Juan Manuel Torres Moreno

```
fonct(*a, *b, n)
{
  // calcul de distance euclidienne entre a et b (dimension n)
  return distance
}

// Il reçoit une matrice X de données m x n avec m patterns
// means est une matrice k x n avec les centroïdes=prototypes
// k nb de clusters
// p>1 coefficient fuzzy
// fonct() fonction qui calcule la distance entre 2 points
// e tolérance
// w une matrice m x k avec le degré d'appartenance

fuzzykmeans(X(m x n ), m, n, means( k x n ), k, p,
            fonct(*a, *b, n), e, w(m x k))
{
  it ← 0;           // Itérations
  //normaliser w par ligne
  Pour i ← 0...m
  {
    sum ← 0
    Pour j ← 0...k : sum ← sum + w( i , j)
    Pour j ← 0...k : w( i , j) ← w( i , j) / sum
  }

  running ← 1     // ITERATIONS
  while (running)
  {
    //normaliser w par colonne (par cluster)
    Pour j ← 0...k
    {
      sum ← 0
      Pour i ← 0...m) : sum ← sum + w( i , j)
      Pour i ← 0...m) : w( i , j) ← w( i , j) / sum
    }

    //calculer les prototypes means[l]
    Pour l ← 0...k {
      Pour j ← 0...n {
        sum ← 0
        Pour i ← 0...m : sum ← sum + X[i][j] * w(i, l)
        oldmeans(l, j) ← means(l, j)
        means(l, j) ← sum
      }
    }
  }
}
```

```

// calculer les poids w
Pour i ← 0...m
{
  sum ← 0;
  Pour l ← 0...k
  {
    dtemp ← fonct (X[i], means[l], n)
    w(i, l) ← 1 / dtemp(p-1)
    sum ← sum + w(i, l)
  }
  Pour l ← 0...k  w(i, l) ← w(i, l) / sum
}

it←it+1    // Incrémenter les itérations

// vérifier si qq chose a change
running ← 0
Pour i ← 0...k
{
  Si( fonct(oldmeans[i], means[i], n) ≥ e )
  alors running ← 1    // Continuer une autre itération
  else break;
} // Si
} // Pour i
} // Itérations

} // fuzzy k means

```