SIMPLIFICATION AND OPTIMIZATION OF I-VECTOR EXTRACTION 这篇文章中提出的提取算法如下：

1. 根据ubm模型每个高斯分量的均值，方差，权重算出t时刻 特征矢量xt相对每个高斯分量c的状态占有率，

$$\gamma_t(c) = \frac{a_c P(x_t \mid \mu_c, \Sigma_c)}{\sum_{i=1}^{C} a_i P(x_t \mid \mu_i, \Sigma_i)}$$

   $P(x|\mu_c, \Sigma_c)$ 高斯概率分布函数

   进而求出每个说话人对应的零阶和一阶Baum – Welch统计量。

2. 参数估计环节，利用已经初始化好的T以及上一步得到的两个统计量求出后验分布的精度矩阵并得到点估计的w。

3. 模型训练环节。E step，利用估计的w，两个统计量以及精度矩阵求出两个累加器A和C。M step，利用$T(c) = CA(c)^{-1}$ 来更新T，再利用新T重复2.参数估计环节，更新精度矩阵以及w。E step与M step循环进行，收敛后得到T与w。

简化版本1：零阶统计量被替换为每个高斯分量c的权重乘以一个常数（即所有高斯分量的原零阶统计量之和），因此精度矩阵的公式得到简化，减少了计算量。

简化版本2：将简化版本1中$L_X = I + N_X W$ 的W分解为$W = Q \Lambda Q^{-1}$ ，令G=Q，通过G将精度矩阵对角化$L_X = G^{(-1)} \hat{L}_X G^{-1}$ ，进而简化了精度矩阵和w的计算。

结合sidekit.FactorAnalyser()代码：

Ivec.py中用sidekit.FactorAnalyser().total_variability()来训练T，但与论文有所不同。它的累加器有A，C，R三个，其中R用于使得散度最小。在得到新T之后并不是对w与精度矩阵进行更新，而是直接对A，C进行更新。

sidekit.FactorAnalyser().extract_ivectors()来提取ivector，与论文不同的是，T已经不发生变化，即只是纯计算过程而不涉及EM算法。