total_variability()

变量解释：

distrib_nb 相当于高斯分量个数 C，feature_size 相当于 F，

sv_size 相当于 CF，upper_triangle_indices 是 T 矩阵上三角各个值的索引，

tv_rank 相当于 M，ivec.py 里面赋值为 400。Statserver 中 stat0 是 X*C 矩阵（X 是语段数目），stat1 是 X*CF 矩阵。

主要步骤：

T 初始化为一个 CF*M 的矩阵，各个值在 0-1 之间，累加器_A 初始化为 C*K 的零矩阵，（设 K=M*（M+1）/2），累加器_C 初始化为 M*CF 的零矩阵，累加器_R 初始化为维度 K 的向量。

E-step 累加每个 *StatServer* 的统计量：

令 nb_sessions=说话人数目，分 batch 进行计算(多线程)

1. 对第一个 batch：累加器 e_h 初始化为 X*M 的零矩阵，累加器 e_hh（用于计算 iv_sigma，但是最终不需要 iv_sigma）初始化为 X*K 的零矩阵，利用

$f(c) \leftarrow f(c) - N(c)m(c)$　　$f(c) \leftarrow \Sigma(c)^{-1/2} f(c)$将数据*预*处理。之后对

X 的每个 idx 进行循环：先用 inv_lambda = scipy.linalg.inv(numpy.eye(tv_rank) + (F.T * stat0[idx, index_map]).dot(F))求 inv_lambda（即精度矩阵 L 的逆），再用 aux = F.T.dot(stat1[idx, :])求 aux（即 $T' f_X$），这两个相乘得到 e_h[idx]，

得到 e_hh[idx]方法是 e_hh[idx]=(inv_lambda+numpy.outer(e_h[idx],e_h[idx]))[upper_triangle_indices]。将每个 batch 的 e_h 累加得到 w。

2.一个语段的 statserver 完成，进入下一个语段的 statserver 后利用_A += stat0.T.dot(e_hh)，_C += e_h.T.dot(stat1)，_R += numpy.sum(e_hh, axis=0)更新三个累加器。

3.遍历所有 statserver 进行第一步与第二步

```

```

```

```

M-step

```
            _A_tmp = numpy.zeros((tv_rank, tv_rank), dtype=STAT_TYPE)

for c in range(distrib_nb):
                distrib_idx = range(c * feature_size, (c + 1) * feature_size)
                _A_tmp[upper_triangle_indices] = _A_tmp.T[upper_triangle_indices] = _A[c, :]
                self.F[distrib_idx, :] = scipy.linalg.solve(_A_tmp, _C[:, distrib_idx]).T
```

```
extract_ivectors ()
```

把 iv_server.stat0 的 stat0 初始化为 X*1 的矩阵， watcher = pool.apply_async(iv_collect, ((iv_server.stat1, iv_sigma), q))这一步有疑问，因为 q = manager.Queue()，应当是空的。

分 batch 进行多线程计算：

```
1.对第一个 batch，进行 total_variability（() E-step 的第一步得到 E_h, E_hh ，用 q.put((batch_indices,) + (E_h, E_hh[:, numpy.array([i * tv_rank-((i*(i-1))//2) for i in range(tv_rank)])]))
```