

《语音识别:从入门到精通》 第三章作业讲解





EM算法



最大期望算法 (Expectation-Maximization algorithm, EM) 用于对包含隐变量 (latent variable) 或缺失数据 (incomplete-data) 的概率模型进行参数估计

最大似然估计



求解步骤:

1、数据集中各样本的联合概率

$$L(heta) = L(x_1, x_2, \ldots, x_n; heta) = \prod_{i=1}^n p(x_i; heta), heta \in \Theta$$

2、对似然函数取对数

$$l(heta) = lnL(heta) = ln\prod_{i=1}^n p(x_i; heta) = \sum_{i=1}^n lnp(x_i; heta)$$

- 3、令导数为0求导,得到似然方程
- 4、解方程,得到参数值

Jensen不等式

若f(x) 是区间[a,b]上的下凸函数,则对任意的 $x_1, x_2, x_3, \ldots, x_n \in [a,b]$,有不等式:

$$\frac{\sum_{i=1}^{n} f(x_i)}{n} \ge f\left(\frac{\sum_{i=1}^{n} x_i}{n}\right)$$

EM算法



EM算法流程

1、对于n个观察数据和模型参数,求极大化模型分布的对数似然函数

$$\hat{ heta} = argmax \sum_{i=1}^{n} logp(x_i; heta)$$

2、加入未观察到的隐变量z,帮助求解

$$\hat{ heta} = argmax \sum_{i=1}^n logp(x_i; heta) = argmax \sum_{i=1}^n log \sum_{z_i} p(x_i, z_i; heta)$$

$$Jensen 不等式$$

$$\sum_{i=1}^n log \sum_{z_i} p(x_i, z_i; heta) = \sum_{i=1}^n log \sum_{z_i} Q_i(z_i) \frac{p(x_i, z_i; heta)}{Q_i(z_i)} (1)$$

$$\geq \sum_{i=1}^n \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; heta)}{Q_i(z_i)} (2)$$

3、未知分布Q的选择(E步)

$$Q_i(z_i) = rac{p(x_i, z_i; heta)}{\sum_z p(x_i, z_i; heta)} = rac{p(x_i, z_i; heta)}{p(x_i; heta)} = p(z_i | x_i; heta)$$

4、极大化对数似然函数的下界(M步)

$$argmax \sum_{i=1}^{n} \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}$$

EM算法在GMM中的应用



高斯混合模型(Gaussian Mixed Model)指的是多个高斯分布函数的线性组合,理论上GMM可以拟合出任意类型的分布,通常用于解决同一集合下的数据包含多个不同的分布的情况

3. M 步

未知分布Q的选择(E步)

$$Q_i(z_i) = rac{p(x_i,z_i; heta)}{\sum_{z}p(x_i,z_i; heta)} = rac{p(x_i,z_i; heta)}{p(x_i; heta)} = p(z_i|x_i; heta)$$

使用当前参数计算后验概率

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

极大化对数似然函数的下界(M步)

$$argmax \sum_{i=1}^{n} \sum_{z_i} Q_i(z_i) log \frac{p(x_i, z_i; \theta)}{Q_i(z_i)}$$

使用后验重新估计参数

$$\begin{split} \mu_k^{new} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n \\ \Sigma_k^{new} &= \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k^{new}) (x_n - \mu_k^{new})^T \\ \pi_k^{new} &= \frac{N_k}{N}, \qquad N_k = \sum_{n=1}^N \gamma(z_{nk}) \end{split}$$

作业代码 (整体介绍)



GMM类

```
def __init__(self, D, K=5):
    assert(D>0)
    self.dim = D
    self.K = K
    #Kmeans Initial
    self.mu_, self.sigma_, self.pi = self.kmeans_initial()
```

定义均指向量、协方差矩阵和混合系数,由 kmeans初始化而得

训练

```
def train(gmms, num_iterations = num_iterations):
    dict_utt2feat, dict_target2utt = read_feats_and_targets('train/feats.scp', 'train/text')

for target in targets:
    feats = get_feats(target, dict_utt2feat, dict_target2utt) #
    for i in range(num_iterations):
        log_llh = gmms[target].em_estimator(feats)
    return gmms
```

对每一个GMM使用相应语料进行五次迭代训练

测试

```
ef test(gmms):
  correction num = 0
  error_num = 0
  dict_utt2feat, dict_target2utt = read_feats_and_targets('test/feats.scp', 'test/text')
  dict_utt2target = {}
  for target in targets:
      utts = dict_target2utt[target]
      for utt in utts:
          dict_utt2target[utt] = target
  for utt in dict_utt2feat.keys():
      feats = kaldi_io.read_mat(dict_utt2feat[utt])
     scores = []
      for target in targets:
          scores.append(gmms[target].calc_log_likelihood(feats))
      predict_target = targets[scores.index(max(scores))]
      if predict_target == dict_utt2target[utt]:
          correction_num += 1
          error_num += 1
  acc = correction_num * 1.0 / (correction_num + error_num)
  return acc
```

每一条测试语料求每一个GMM模型下的似然, 求argmax得到似然最大的模型作为输出结果

作业代码 (对数似然函数)



```
def calc_log_likelihood(self_, X):
   n_s = X.shape[0]
   log_llh = []
   for n in range(n_s):
       log_llh_k = []
           log_llh_k.append(self.pi[k] * self.gaussian(X[n], self.mu[k], self.sigma[k]))
       log_llh.append(np.log(np.sum(log_llh_k)))
   log_llh = np.sum(log_llh)
   return log_llh
```



GMM模型

· GMM的对数似然函数

$$\left\{ \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$



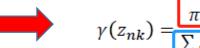
其中,
$$\mathbf{X} = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_N^T \end{bmatrix}$$
 同时给出潜变量矩阵定义 $\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_N^T \end{bmatrix}$



作业代码 (最大期望算法-E步)



```
def em_estimator(self_, X):
   n_s = X.shape[0]
   for i in range(n_s):
       sample.append(np.zeros(self.K))
   for n in range(n_s):
       numerator = []
           numerator.append(self.pi[k] * self.qaussian(X[n], self.mu[k], self.sigma[k]))
       denominator = np.sum(np.array(numerator))
       sample[n] = numerator / denominator
```

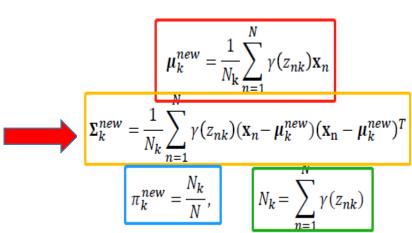


$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

作业代码 (最大期望算法-M步)



```
for k in range(self.K):
   nk = []
    for n in range(n_s):
        nk.append(sample[n][k])
   nk = np.sum(nk)
    if nk != 0:
       mut = []
        for n in range(n_s):
            mut.append(sample[n][k] * X[n])
       self.mu[k] = np.sum(mut, axis=0) / nk
       siqt = []
        for n in range(n_s):
            sigt.append(sample[n][k] * (X - self.mu[k])[n].reshape(self.dim, 1) * (X - self.mu[k])[n])
        self.sigma[k] = np.sum(sigt, axis=0) / nk
        self.pi[k] = nk / n_s
log_llh = self.calc_log_likelihood(X)
return log_llh
```



参考资料



- 1、第三章课件 "GMM以及EM算法"
- 2、EM算法详解, https://zhuanlan.zhihu.com/p/40991784



感谢各位聆听 Thanks for Listening

