Report of Deep Learning for Natural Langauge Processing

高志磊
gaozhilei@buaa.edu.cn

Introduction

极大似然算法可以估计模型参数，但是如果有多个分布，比如已知男生、女生身高分别服从正态分布，但是其身高数据混杂在一起，如何确定男生、女生的分布情况？针对这一问题，需要引入隐变量，隐变量表示男生、女生的比例，随后进行参数估计，这就是EM 算法的核心思想。

Methodology

M1：高斯混合模型

高斯混合模型（Gaussian Mixed Model，GMM）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分布的情况（或者是同一类分布但参数不一样，或者是不同类型的分布，比如正态分布和伯努利分布）。GMM 的概率密度函数如下：

$$p(x)=\sum_{k=1}^{K}p(k)p(x|k)=\sum_{k=1}^{K}\pi_k N(x|\mu_k,\sigma_k)$$

其中，$p(x|k)=N(x|\mu_k,\sigma_k)$ 是第 k 个高斯模型的概率密度函数，可以看成选定第 k 个模型后，该模型产生 x 的概率。$p(k)=\pi_k$ 是第 k 个高斯模型的权重，称作选择第 k 个高斯模型的先验概率，且满足 $\sum_{k=1}^{K}\pi_k=1$。

M2: EM 算法

EM 是一种用于含有隐变量的模型参数的最大似然估计，是一种迭代算法，通过 E 步和 M 步交替迭代直至收敛估计出模型参数。

E 步：求第 j 个数据属于第 k 个模型的后验概率 γ_{jk}：

$$\gamma_{jk} = \frac{\theta_k p(x_j \mid \mu_k, \sigma_k)}{\sum_{k=1}^{K} \theta_k p(x_j \mid \mu_k, \sigma_k)}$$

其中，θ_k 是每个高斯模型的权重，$p(x_j \mid \mu_k, \sigma_k)$ 是第 k 个高斯模型的概率密度函数。

M 步：利用所有数据的后验概率重新估计高斯混合模型的参数：

$$\mu_k^{new} = \frac{\sum_{j=1}^{N} \gamma_{jk} x_j}{\sum_{j=1}^{N} \gamma_{jk}}$$

$$\sigma_k^{new} = \frac{\sum_{j=1}^{N} \gamma_{jk} (x_j - \mu_k)^2}{\sum_{j=1}^{N} \gamma_{jk}}$$

$$\theta_k^{new} = \frac{\sum_{j=1}^{N} \gamma_{jk}}{N}$$

重复计算 E 步和 M 步直至收敛。

M3：程序实现

（一）参数初始化，如果指定初始参数，则使用给定的初始参数，如果没有，则假设模型概率相等，所有模型均值为随机抽取一定样本的均值，所有模型方差为 1。

```python
def __init__(self, data, n, theta=None, miu=None, sigma=None):
    self.data = data
    self.data_len = len(data)
    self.n = n
    if theta is not None:
        self.theta = theta
    else:
        self.theta = [1 / self.n for i in range(self.n)]

    if miu is not None:
        self.miu = miu
    else:
        self.miu = []
        for i in range(self.n):
            sample_count = int(len(self.data) * self.theta[i])
            sample = random.sample(self.data, sample_count)
            self.miu.append(sum(sample) / sample_count)

    if sigma is not None:
        self.sigma = sigma
    else:
        self.sigma = [1 for i in range(self.n)]
```

（二）高斯密度函数：

```python
def gaussian(self, x, miu, sigma):
    result = (1 / (math.sqrt(2 * math.pi) * sigma)) * math.exp(-0.5 * ((x - miu) / sigma) ** 2)
    return result
```

（三）E 步：

```python
# e步
gamma = np.zeros((self.data_len, self.n))
for i in range(self.data_len):
    for n in range(self.n):
        gamma[i][n] = self.theta[n] * self.gaussian(self.data[i], self.miu[n], self.sigma[n])
    gamma[i] = gamma[i] / sum(gamma[i])
```

（四）M 步：

```
miu_new = [0, 0]
sigma_new = [0, 0]
theta_new = [0, 0]
for n in range(self.n):
    miu_new[n] = np.dot(self.data, gamma[:, n]) / np.sum(gamma[:, n])
    sigma_new[n] = math.sqrt(np.dot((np.array(self.data) - self.miu[n]) ** 2, gamma[:, n]) / np.sum(
        gamma[:, n]))
    theta_new[n] = np.mean(gamma[:, n])
if np.max(np.array(miu_new) - np.array(self.miu)) < eps and np.max(
        np.array(sigma_new) - np.array(self.sigma)) < eps and np.max(
    np.array(theta_new) - np.array(self.theta)) < eps:
    self.miu = miu_new
    self.sigma = sigma_new
    self.theta = theta_new
    break
else:
    self.miu = miu_new
    self.sigma = sigma_new
    self.theta = theta_new
```

Experimental Studies

给定 2000 个身高数据，已知男生身高和女生身高都服从正态分布，使用 EM 算法估计男女生身高分布的参数。实验结果如表 1 所示：

Table 1：实验结果

		θ_1	θ_2	μ_1	μ_2	σ_1	σ_2
第一组	真实值	0.75	0.25	176	164	5	3
	EM 算法初始值	0.5	0.5	170	160	1	1
	EM 算法估计值	0.7674	0.2326	175.8614	163.5913	5.2711	2.9442
第二组	真实值	0.75	0.25	176	164	5	3
	EM 算法初始值	0.5	0.5	170	170	1	1
	EM 算法估计值	0.5	0.5	173.0078	173.0078	7.0859	7.0859

Conclusions

从上述结果可以看出，EM 算法并不能保证收敛到全局最优值，算法执行结果与初始值的选取有关。