机器学习——第四次作业

Koorye

2024 年 3 月 22 日

1 试说明为什么核技巧（kernel trick）使得可以在高维特征空间运用 SVM 而不显著增加运行时间。

使用核函数的 SVM 的决策函数为：

$$f(x)=\sum_{i=1}^{n}\alpha_i y_i K(x,x_i)+b, \tag{1}$$

其中 $K(x_i,x_j)=\phi(x_i)^T\phi(x_j)$，$\phi(x)$ 是 x 的映射函数。

如果直接计算 $\phi(x_i)$ 和 $\phi(x_j)$，再计算内积，可能会耗费大量时间。例如，假设 $x=(x_1,x_2,\dots,x_n)$，$\phi(x)=(x_1x_1,x_1x_2,\dots,x_2x_1,\dots,x_nx_{n-1},x_nx_n)$。此时需要分别对 x_i,x_j 计算 $\phi(x_i),\phi(x_j)$，即

$$\phi(x_i)=(x_{i1}x_{i1},x_{i1}x_{i2},\dots,x_{in}x_{in}),\phi(x_j)=(x_{j1}x_{j1},x_{j1}x_{j2},\dots,x_{jn}x_{jn}), \tag{2}$$

再计算内积，即

$$K(x_i,x_j)=\phi(x_i)^T\phi(x_j)=x_{i1}x_{i1}x_{j1}x_{j1}+\cdots+x_{in}x_{in}x_{jn}x_{jn}, \tag{3}$$

此时需要 $3n^2$ 次计算。

然而，通过核函数可以直接计算内积，即 $K(x_i,x_j)=(x_{i1}x_{j1},x_{i2}x_{j2}+\cdots+x_{in}x_{jn})^2$。核函数只需要 n 次计算，可以大大减少计算量。

上述核函数是一个二次核函数，属于一种特例。此外，还有很多种核函数，如线性核函数、多项式核函数、高斯核函数等，可以根据具体问题选择合适的核函数。这些核函数的特点是可以直接计算内积，而不需要显式地计算映射函数 $\phi(x)$，从而减少计算量。

2 描述 K 均值算法，并说明其缺点。

K 均值算法是一种基于距离的聚类算法，其基本思想是：首先初始化 K 个聚类中心，然后将每个样本分配到与其最近的聚类中心所在的簇中，接着重新计算每个簇的中心，直到聚类中心不再发生变化或者达到最大迭代次数为止。K 均值算法可以描述为算法**??**。

K 均值算法的缺点主要有以下几点：

1. K 均值算法对初始聚类中心敏感，不同的初始聚类中心可能会导致不同的聚类结果。
2. K 均值算法对噪声和异常值敏感，可能会导致聚类结果不稳定。
3. K 均值算法需要事先确定聚类数 K，但在实际应用中，往往无法事先确定聚类数。

Algorithm 1 K 均值算法

Require: N 个样本 $x_1, x_2, \ldots, x_N, x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$，其中 d 表示维度，聚类数 K，最大迭代次数 T

Ensure: K 个聚类中心 $\mu_1, \mu_2, \ldots, \mu_K$

初始化 K 个聚类中心 $\mu_1, \mu_2, \ldots, \mu_K$
repeat
  for 每个样本 $x_i, i \in 1, \ldots, N$ **do**
    for 每个聚类中心 $\mu_j, j \in 1, \ldots, K$ **do**
      计算样本 x_i 与聚类中心 μ_j 的距离 $D(x_i, \mu_j)$
    end for
    为样本 x_i 分配距离最近的聚类中心 $C(x_i) = arg\min_j D(x_i, \mu_j)$
  end for
  for 每个聚类中心 $\mu_j, j \in 1, \ldots, K$ **do**
    重新计算位置 $\mu_j = \frac{1}{N_j} \sum_{C(x_i)=j} x_i$，其中 $N_j = \sum_{i=1}^{N} I(C(x_i) = j)$
  end for
until 聚类中心不再发生变化或达到最大迭代次数 T

4. K 均值算法只能得到凸形簇，对于非凸形簇的聚类效果不佳。

3 从 EM 算法的角度给出 KMeans 算法和 GMM 算法的 E 步和 M 步，并说明这两种算法的区别和联系。

EM 算法是一种迭代优化算法，用于求解包含隐变量的概率模型的参数估计问题。EM 算法包含两个步骤：E 步和 M 步。E 步是求期望，即在给定模型参数的情况下，计算隐变量的条件概率分布。M 步是求极大值，即在给定隐变量的条件概率分布的情况下，计算模型参数的极大似然估计。

对于 KMeans 算法来说，E 步是将每个样本分配到与其最近的聚类中心所在的簇中，M 步是重新计算每个簇的中心。E 步可以表示为：

$$\gamma_{ij} = \begin{cases} 1, & arg\min_j D(x_i, \mu_j) \\ 0, & \text{Otherwise,} \end{cases} \tag{4}$$

其中 x_i, μ_j 分别表示第 i 个样本和第 j 个聚类中心。而 M 步可以表示为：

$$\mu_j = \frac{\sum_{i=1}^{N} \gamma_{ij} x_i}{\sum_{i=1}^{N} \gamma_{ij}}. \tag{5}$$

对于 GMM 算法来说，E 步是计算每个样本属于每个高斯分布的概率，M 步是重新计算每个高斯分布的均值和方差。E 步可以表示为：

$$\gamma_{ij} = \frac{\pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)}, \tag{6}$$

其中 π_i, μ_i, Σ_i 分别表示第 i 个高斯分量的系数、均值和方差。而 M 步可以表示为：

$$
\begin{aligned}
\mu_j &= \frac{\sum_{i=1}^N \gamma_{ij} x_i}{\sum_{i=1}^N \gamma_{ij}}, \\
\Sigma_j &= \frac{\sum_{i=1}^N \gamma_{ij} (x_i - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^N \gamma_{ij}}, \\
\pi_j &= \frac{1}{N} \sum_{i=1}^N \gamma_{ij}.
\end{aligned} \tag{7}
$$

KMeans 算法和 GMM 算法的区别主要在于：

1. KMeans 算法是一种硬聚类算法，每个样本只能属于一个簇；而 GMM 算法是一种软聚类算法，每个样本可以属于多个高斯分布。
2. KMeans 算法对初始聚类中心敏感，而 GMM 算法对初始参数不敏感。
3. KMeans 算法对噪声和异常值敏感；而 GMM 算法对噪声和异常值不敏感。
4. KMeans 算法假设每个簇是一个凸形簇，而 GMM 算法假设每个高斯分布是一个椭球形簇。

两者的联系主要在于：

1. KMeans 算法和 GMM 算法都是基于 EM 算法的聚类算法，都是通过迭代求最大似然的过程。
2. KMeans 算法和 GMM 算法求得的都是局部最优解。
3. KMeans 算法和 GMM 算法都是无监督学习方法，不需要标签。
4. KMeans 算法和 GMM 算法都需要事先确定聚类数，这难以确定，且会直接影响聚类质量。

4 写出 EM 算法的一般形式，试说明其收敛性。

EM 算法是一种迭代优化算法，用于求解包含隐变量的概率模型的参数估计问题。EM 算法包含两个步骤：E 步和 M 步。E 步是求期望，即在给定模型参数的情况下，计算隐变量的条件概率分布。M 步是求极大值，即在给定隐变量的条件概率分布的情况下，计算模型参数的极大似然估计。具体来说，EM 算法的一般形式可以表示为算法**??**。

Algorithm 2 EM 算法

Require: 观测数据 $X = \{x_1, x_2, \dots, x_N\}$，模型参数 θ，最大迭代次数 T

Ensure: 模型参数 θ

1: 初始化模型参数 θ_0
2: **repeat**
3: E 步：计算隐变量的条件概率分布 $P(Z|X, \theta_t)$
4: M 步：计算极大似然 $\theta_{t+1} = arg\max_\theta Q(\theta, \theta_t)$，其中 $Q(\theta, \theta_t) = \sum_Z P(Z|X, \theta_t) \ln P(X, Z|\theta)$
5: **until** 收敛或达到最大迭代次数 T

EM 算法之所以能够收敛，是因为 EM 算法的目标函数是单调递增的。具体来说，通过引入隐变量 Z 的概率分布 $q(Z)$，EM 算法的目标函数即极大似然可以表示为：

$$\ln P(X|\theta) = \mathcal{L}(q,\theta) + KL(q||p), \tag{8}$$

其中

$$\mathcal{L}(q,\theta) = \sum_Z q(Z) \ln \frac{P(X,Z|\theta)}{q(Z)}, KL(q||p) = -\sum_Z q(Z) \ln \frac{P(Z|X,\theta)}{q(Z)}. \tag{9}$$

由于 $KL(q||p) \geq 0$，则 $\mathcal{L}(q,\theta)$ 是极大似然的下界。

在 E 步中，固定 θ，最大化下界 $\mathcal{L}(q,\theta)$，即通过调整 q，使得 q 与当前 θ 下的 $P(Z|X,\theta)$ 尽可能接近。

在 M 步中，固定 q，最大化下界 $\mathcal{L}(q,\theta)$，即通过调整 θ，使得极大似然随下界的增大而增大。

因此 EM 算法的目标函数是单调递增的，即每次迭代都能使目标函数增大，从而 EM 算法能够收敛。