Nearest Neighbour

> Pattern Recognition Homeworks
>
> Student: thu-zxs

Solutions

1.

Consider Voronoi region of a_0, w.r.t. $a_i, (i = 1, 2, \ldots, k)$. For a specific a_i, points which are closer to a_0 than a_i satisfiy:

$$\|x - a_0\| \leq \|x - a_i\|,$$

Which in Euclidean space is:

$$\begin{aligned}
\|x - a_0\|^2 \leq \|x - a_i\|^2 &\Longleftrightarrow (x - a_0)^T(x - a_0) \leq (x - a_i)^T(x - a_i) \\
&\Longleftrightarrow (x^T x - 2a_0^T x - a_0^T a_0) \leq (x^T x - 2a_i^T x - a_i^T a_i) \\
&\Longleftrightarrow 2(a_i - a_0)^T x \leq a_0^T a_0 - a_i^T a_i,
\end{aligned}$$

which represents a half-space. A half-space is a convex set by proof using convex sets's definition:

$$\begin{aligned}
S = \{x \in \mathbf{R}_n | A^T x \leq b\}&, \\
0 < \theta < 1&, \\
x_1 \in S, x_2 \in S&, \\
A^T(\theta x_1 + (1-\theta)x_2) = \theta A^T x_1 + (1-\theta)A^T x_2 \leq \theta b + (1-\theta)b = b&
\end{aligned}$$

so

$$\theta x_1 + (1-\theta)x_2 \in S$$

Also intersection of multiple convex sets is also a convex set:

$$\begin{aligned}
V &= \{x \in \mathbf{R}_n | \|x - a_0\| \leq \|x - a_i\|, i = 1, 2, \ldots, k\} \\
&= \bigcap_{i=1,2,\ldots,k} \{x | \|x - a_0\| \leq \|x - a_i\|\},
\end{aligned}$$

So Voronoi region of a_0 w.r.t. $a_i, \forall i$ is a convex set.

2.

1)

$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{\sum_j P(x|w_j)(P(w_j)}$$
$$= \begin{cases} \frac{1}{c}, 0 \le x \le \frac{cr}{c-1} \\ 1, i \le x \le i+1-\frac{cr}{c-1} \\ 0, otherwise \end{cases}$$

$$P^*(e) = \int P^*(e|x)p(x)dx = \int (1 - \max_i P^*(w_i|x))p(x)dx$$
$$= \int_0^{\frac{cr}{c-1}} \frac{c-1}{c} p(x)dx + \sum_{i=1}^{c} \int_i^{i+1-\frac{cr}{c-1}} (1-1)p(x)dx$$
$$= r$$

2)

$$P(e) = \int [1 - \sum_{i=1}^{c} P^2(w_i|x)]p(x)dx$$
$$= \int_c^{\frac{cr}{c-1}} [1 - \sum_{i=1}^{c} \frac{1}{c^2}]p(x)dx$$
$$= r$$

3.

1)

$$D_M(x,y) = (\sum_{j=1}^{d} |x_j - y_j|^s)^{\frac{1}{s}} \ge 0$$

2)

$$D_M(x,y) = (\sum_{j=1}^{d} |x_j - y_j|^s)^{\frac{1}{s}} = 0 \iff x_j = y_j$$

3)

$$D_M(x,y) = (\sum_{j=1}^{d} |x_j - y_j|^s)^{\frac{1}{s}} = D_M(y,x)$$

4) from Minkowski's inequality:

$$D_M(x,y)=(\sum_{j=1}^{d}|x_j-y_j|^s)^{\frac{1}{s}}=(\sum_{j=1}^{d}|x_j-z_j+z_j-y_j|^s)^{\frac{1}{s}}$$

$$\le(\sum_{j=1}^{d}|x_j-z_j|^s)^{\frac{1}{s}}+(\sum_{j=1}^{d}|z_j-y_j|^s)^{\frac{1}{s}}=D_M(x,z)+D_M(z,y)$$

So Minkowski distance is a metric.

Programming

编程实现中，利用`numpy`的"broadcast"特性，将遍历计算距离以及搜索前k小距离的过程写成矩阵运算的形式，使得时间复杂度近乎为$O(d+N)$, 空间复杂度为$O(MN)$,其中M为测试样本数，N为训练样本数，d为样本维度。

KNN 算法:

```
def KNN(XTrain, YTrain, XTest, YTest, disFunc, k=1, tangent=False):

    if tangent:
        dis = disFunc(XTrain, XTest, rotTangent(XTrain))
    else:
        dis = disFunc(XTrain, XTest)
    numTrain = XTrain.shape[0]
    numTest = XTest.shape[0]
    minMatch = np.argpartition(dis, kth=k-1, axis=1)[:, :k]
    matchedClasses = YTrain[minMatch.reshape(-1)].reshape(numTest, k)
    binCount = np.array(map(lambda x:np.bincount(x,minlength=10),
matchedClasses))
    matchedClass = np.argmax(binCount, axis=1)

    return (matchedClass==YTest).astype(np.float).sum()/XTest.shape[0]
```

欧式距离:

```
def Euclidean(XTrain, XTest):

    testNorm = np.sum(XTest*XTest, axis=1)[:, np.newaxis]
    trainNorm = np.sum(XTrain*XTrain, axis=1)[np.newaxis, :]
    cross = XTest.dot(XTrain.T)
    return np.sqrt(testNorm + trainNorm - 2*cross)
```

l_4距离:

```python
def L4(XTrain, XTest):

    testQuad = np.sum(XTest**4, axis=1)[:, np.newaxis]
    trainQuad = np.sum(XTrain**4, axis=1)[np.newaxis, :]

    cross = -4*(XTest**3).dot(XTrain.T)
    cross += -4*(XTest).dot((XTrain**3).T)
    cross += 6*(XTest**2).dot((XTrain**2).T)

    return (testQuad + trainQuad + cross)**(1.0/4)
```

1) 固定$k=1$，采用欧式距离

训练样本数N	准确率	空间复杂度$O(MN)$	时间复杂度
1000	0.8622	$M=10000, N=1000$	$O(2d+N)$
10000	0.9463	$M=10000, N=10000$	$O(2d+N)$
60000	0.9705	$M=10000, N=60000$	$O(2d+N)$

2) 固定$N=10000$，采用欧式距离

k	准确率	空间复杂度$O(MN)$	时间复杂度
1	0.9463	$M=10000, N=10000$	$O(2d+N)$
3	0.9463	$M=10000, N=10000$	$O(2d+N)$
10	0.9411	$M=10000, N=10000$	$O(2d+N)$
50	0.9122	$M=10000, N=10000$	$O(2d+N)$

3) 固定$k=1, N=10000$

度量	准确率	空间复杂度$O(MN)$	时间复杂度
欧式距离	0.9463	$M=10000, N=10000$	$O(2d+N)$
l_4距离	0.9532	$M=10000, N=10000$	$O(2d+N)$

4）存在

使用方差为0.1，大小为9×9的高斯窗函数对图像做预处理：

```python
def make_gaussian_window(n, sigma=1):
    nn = int((n-1)/2)
    a = np.asarray([[x**2 + y**2 for x in range(-nn,nn+1)] for y in range(-
nn,nn+1)])
    return np.exp(-a/(2*sigma**2))
```

```python
win_size = 9
down_size = 28 - win_size + 1
window = make_gaussian_window(win_size, 0.1)
train, test = [], []
for im in XTrain:
    train.append(conv(im.reshape(28, 28), window).reshape(-1).copy())
XTrain = np.array(train)
for im in XTest:
    test.append(conv(im.reshape(28, 28), window).reshape(-1).copy())
XTest = np.array(test)
```

在欧式距离下，使用$N = 60000$个训练样本，k =1，准确率提升至0.9719，如下

是否采用高斯滑窗	准确率	空间复杂度$O(MN)$	时间复杂度
N	0.9705	$M = 10000, N = 60000$	$O(2d + N)$
Y	0.9719	$M = 10000, N = 60000$	$O(2d + N)$

5) 切线距离设计采用**旋转变换**产生的切平面：

i) 计算梯度

$$\nabla = y\frac{\partial}{\partial x} - x\frac{\partial}{\partial y}$$

```python
def rotTangent(XTrain, h=28, w=28):

    Xtemp = XTrain.reshape((-1, h, w))
    partial_x = np.zeros(Xtemp.shape)
    partial_y = np.zeros(Xtemp.shape)
    partial_x[:, 0:h-1, :] = Xtemp[:, 1:h, :] - Xtemp[:, 0:h-1, :]
    partial_y[:, :, 0:w-1] = Xtemp[:, :, 1:w] - Xtemp[:, :, 0:w-1]

    factor = 1.0/(w*0.5)
    offset = 0.5 - 1.0/factor
    mesh_y, mesh_x = np.meshgrid(range(h), range(w))
    mesh_x, mesh_y = offset+mesh_x, offset+mesh_y
    mesh_x, mesh_y = mesh_x[np.newaxis, :], mesh_y[np.newaxis, :]

    tangent = (mesh_y*partial_x - mesh_x*partial_y)*factor
    return tangent.reshape((-1, h*w)).copy()
```

ii) 切线距离

$$\min_{\alpha} \|x - (x' + \alpha \nabla x')\|_2 \Longleftrightarrow \min_{\alpha} (x - (x' + \alpha \nabla x'))^T (x - (x' + \alpha \nabla x'))$$

为凸函数，通过求梯度可解得：

$$\alpha = \frac{x^T T - x'^T T}{T^T T}$$

其中T为切向量，将α回代可求得切距离：

```
def tangentDistance(XTrain, XTest, Tangent):

    cross1 = XTest.dot(Tangent.T)
    cross2 = np.sum(XTrain*Tangent, axis=1)[np.newaxis, :]
    cross3 = np.sum(Tangent*Tangent, axis=1)[np.newaxis, :]
    Alpha = (cross1 - cross2)/cross3

    item = Euclidean(XTrain, XTest)
    item += -2*Alpha*(cross1-cross2)
    item += (Alpha**2)*cross3

    return item
```

结果如下：

度量	准确率	空间复杂度$O(MN)$	时间复杂度
欧式距离	0.9463	$M = 10000, N = 10000$	$O(2d + N)$
切线距离	0.9511	$M = 10000, N = 10000$	$O(2d + 2N)$