2016 模式识别大作业-食物分类

邓志会 2015210926

对于报告或者程序中不清楚或者有问题可以联系：18810575596 dzh15@mails.tsinghua.edu.cn

目录

1 问题背景

PCA/Fisher 进行 10 个类别食物分类。请采用 PCA 方法或者 Fisher 线性判别准则的方法完成下面的实验

根据 train.txt 和 test.txt 列表选取训练集和测试集，采用 k 近邻分类，分析选取不同的主分量个数和近邻个数 K，对识别率和虚警率的影响。图像特征可以采用灰度像素值等。

评价该方法的性能。

每个类别$P_{i,j} = N_{i,j} / \sum_{j=1}^{c} N_{i,j}$

对角线为每个类别的正确识别率

非对角线表示每行对应类别识别为其他类别的误识率

1.1 使用 pca 方法做实物分类

最简单的想法就是对于每张训练和测试集合的图片先把彩色图片转化为灰度图片，然后从图片抽取 300 乘以 300，以 3 为间隔进行降采样之后得到 100 乘以 100 的图片，将这些图片的像素数据进行向量化，将训练集合的所有图片 7293 张合到一个集合中 7293×10000，形成一个使用 PCA 方法进行降维到 40 维，形成 7293×40 大小的矩阵。

选取测试图片约 2400 张，把彩色图片转化为灰度图片，裁剪为 300×300 的图片，以 3 为间隔进行降采样得到 100×100 的图片，使用 PCA 降维得到 2400×40 的图片。

训练集合为 7293×40 的矩阵，测试图片集合为 2400×40 的矩阵。

接下来使用 k 近邻的方法进行分类。

1.1.1 PCA 方法简单分析

该问题的解决使用了 KL 变换和主分量分析（PCA），首先给出我对于这些的理解。进行特征提取，使用了最小均方误差来进行最优正交变换，能够消除模式

特征之间的相关性，突出不同特征的差异。

以人脸分析为基础，在原来的坐标系下面向量是$X = (x1, x2, \dots, xn)^T$，新的坐标系下面的表达式是$X = \sum_{i=1}^{n} a_i \phi_i$，$\phi_i$是新的坐标系的基

忽略掉 n-m 个特征则会产生误差$\varepsilon^2 = \sum_{j=m+1}^{n} a_j^2$

平均二乘误差为$\overline{\varepsilon^2} = \sum_{j=m+1}^{n} \phi_j^T S \phi_j$

$S = \frac{1}{l} \sum_{j=1}^{l} X_j \cdot X_j^T$ 是 n*n 自相关矩阵

求 S 的特征向量和特征值，特征值从大到小排序，相应的特征向量也进行归一化。

$$\overline{\varepsilon^2} = \sum_{j=m+1}^{n} \phi_j^T S \phi_j = \sum_{j=m+1}^{n} \lambda_j$$

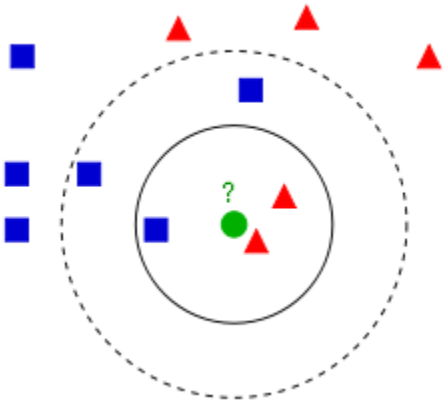
利用前 K 个特征向量$v_1\ v_2 \dots\ v_k$

投影到特征空间的过程就是$[(x - \bar{x}) \cdot v_1, (x - \bar{x}) \cdot v_2, \dots, (x - \bar{x}) \cdot v_K]$

1.1.2 K 近邻的方法简单分析

K Nearest Neighbor 算法又叫 KNN 算法，这个算法是机器学习里面一个比较经典的算法， 总体来说 KNN 算法是相对比较容易理解的算法。其中的 K 表示最接近自己的 K 个数据样本。KNN 算法和 K-Means 算法不同的是，K-Means 算法用来聚类，用来判断哪些东西是一个比较相近的类型，而 KNN 算法是用来做归类的，也就是说，有一个样本空间里的样本分成很几个类型，然后，给定一个待分类的数据，通过计算接近自己最近的 K 个样本来判断这个待分类数据属于哪个分类。你可以简单的理解为由那离自己最近的 K 个点来投票决定待分类数据归为哪一类。

Wikipedia 上的 KNN 词条中有一个比较经典的图如下：



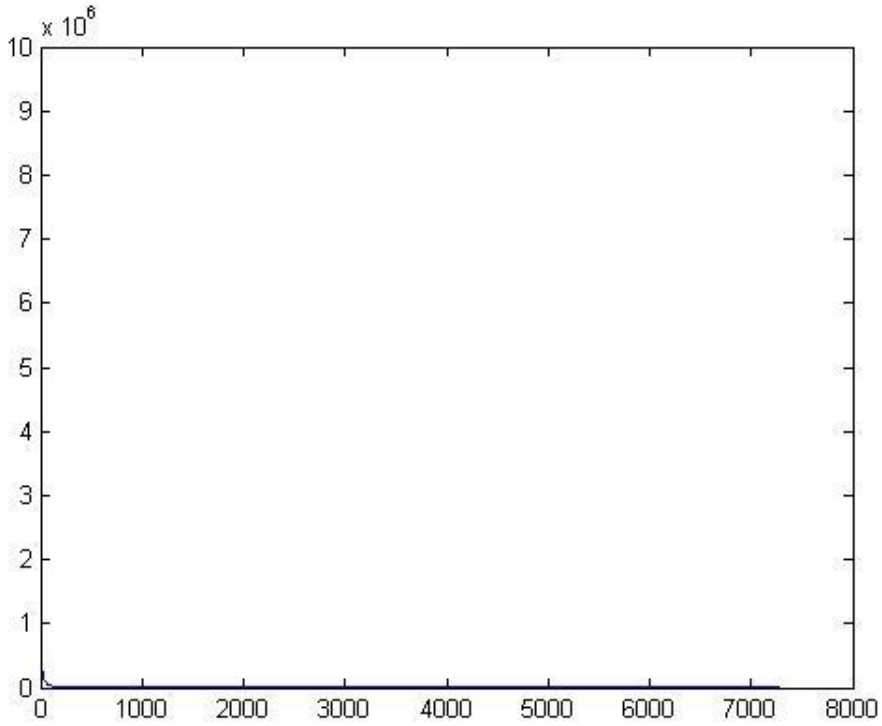
从上图中我们可以看到，图中的有两个类型的样本数据，一类是蓝色的正方形，另一类是红色的三角形。而那个绿色的圆形是我们待分类的数据。

如果 K=3，那么离绿色点最近的有 2 个红色三角形和 1 个蓝色的正方形，这 3 个点投票，于是绿色的这个待分类点属于红色的三角形。

如果 K=5，那么离绿色点最近的有 2 个红色三角形和 3 个蓝色的正方形，这 5 个点投票，于是绿色的这个待分类点属于蓝色的正方形。

1.1.3 pca 加上 k 近邻方法实验结果及相关分析

对于 pca 降维选取了前面 40 维

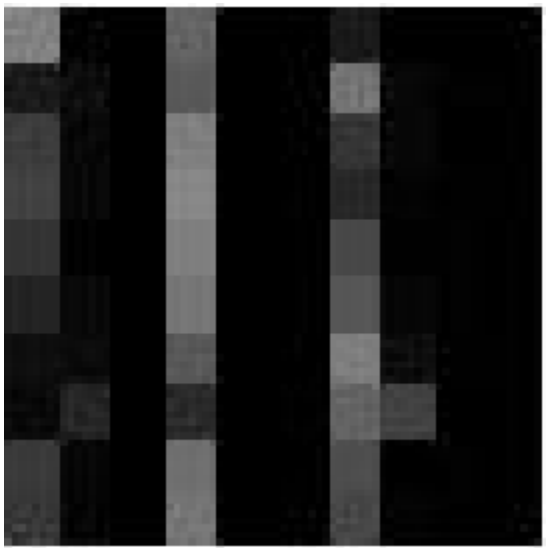


在取前面 200 维，k 取 800 时候的结果是

识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

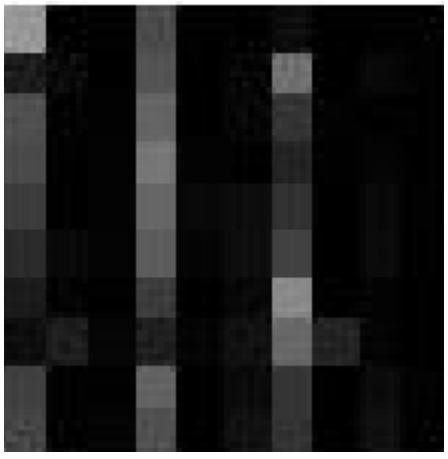
125	2	0	96	0	0	25	0	0	0
26	13	0	87	0	0	112	7	2	1
59	7	0	123	0	0	51	7	0	0
63	8	0	129	0	0	37	3	1	1
51	1	0	123	0	0	70	1	1	0
33	9	0	113	0	0	82	6	1	0
14	11	0	84	0	0	120	13	1	0
9	35	0	40	0	1	94	63	0	0
55	6	0	105	0	0	74	0	2	0
52	5	0	110	0	0	67	3	1	1

第 1 类食物图片的识别召回率为 0.5040，虚警率为 0.1643
第 2 类食物图片的识别召回率为 0.0524，虚警率为 0.0381
第 3 类食物图片的识别召回率为 0，虚警率为 0
第 4 类食物图片的识别召回率为 0.5331，虚警率为 0.3999
第 5 类食物图片的识别召回率为 0，虚警率为 0
第 6 类食物图片的识别召回率为 0，虚警率为 0.0004
第 7 类食物图片的识别召回率为 0.4938，虚警率为 0.2778
第 8 类食物图片的识别召回率为 0.2603，虚警率为 0.0182
第 9 类食物图片的识别召回率为 0.0083，虚警率为 0.0032
第 10 类食物图片的识别召回率为 0.0042，虚警率为 0.0009



在取前面 200 维，k 取 50 时候的结果是

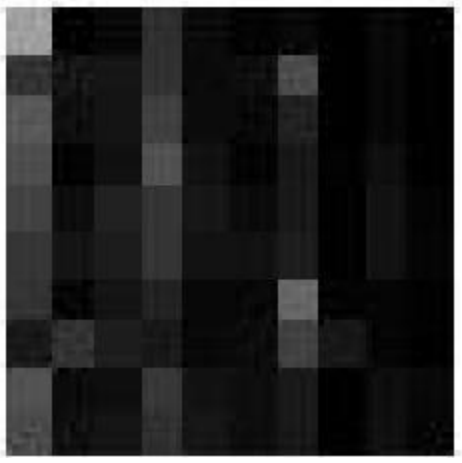
第 1 类食物图片的识别召回率为 0.6129，虚警率为 0.207839562443026
第 2 类食物图片的识别召回率为 0.0403，虚警率为 0.0232452142206016
第 3 类食物图片的识别召回率为 0.0081，虚警率为 0.0113895216400911
第 4 类食物图片的识别召回率为 0.4876，虚警率为 0.326818181818182
第 5 类食物图片的识别召回率为 0.0323886639676113，虚警率为 0.00728929384965832
第 6 类食物图片的识别召回率为 0.0573770491803279，虚警率为 0.0369
第 7 类食物图片的识别召回率为 0.563786008230453，虚警率为 0.2424
第 8 类食物图片的识别召回率为 0.140495867768595，虚警率为 0.0018
第 9 类食物图片的识别召回率为 0.0454545454545455，虚警率为 0.0264
第 10 类食物图片的识别召回率为 0.00418410041841004，虚警率为 0.0054



在取前面 200 维，k 取 4 时候的结果是

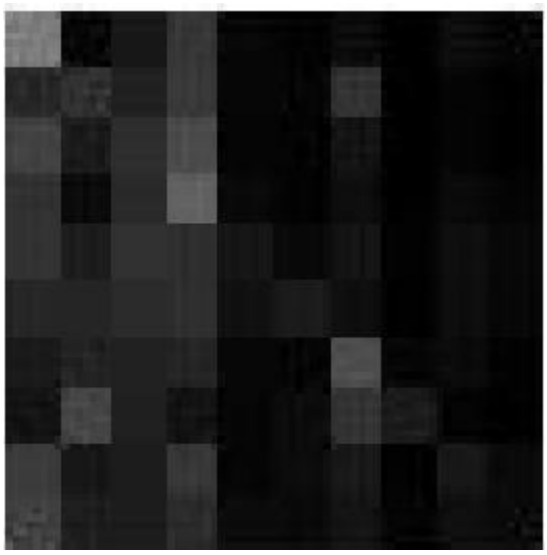
识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

160 0 8 47 11 3 9 0 9 1
50 22 22 49 9 14 75 0 7 0
88 15 20 64 11 9 33 0 7 0
78 3 17 81 17 4 24 2 15 1
66 14 33 53 22 11 25 1 18 4
54 21 28 52 18 18 29 1 18 5
51 6 20 35 8 8 103 4 6 2
25 50 29 24 8 12 65 23 5 1
82 12 14 55 17 14 25 0 14 9
74 16 20 47 22 13 24 0 16 7
第 1 类食物图片的识别召回率为 0.6452，虚警率为 0.2589
第 2 类食物图片的识别召回率为 0.0887，虚警率为 0.0624
第 3 类食物图片的识别召回率为 00810，虚警率为 0.0870
第 4 类食物图片的识别召回率为 0.3347，虚警率为 0.1936
第 5 类食物图片的识别召回率为 0.0891，虚警率为 0.06=551
第 6 类食物图片的识别召回率为 0.0738，虚警率为 0.04
第 7 类食物图片的识别召回率为 0.4239，虚警率为 0.1405
第 8 类食物图片的识别召回率为 0.0950，虚警率为 0.0036
第 9 类食物图片的识别召回率为 0.0579，虚警率为 0.0459
第 10 类食物图片的识别召回率为 0.0293，虚警率为 0.0104



在取前面 40 维，k 取 4 时候的结果是
识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：
125 7 23 54 7 3 11 0 12 6
40 49 33 57 6 5 51 0 5 2
66 27 46 75 5 4 19 0 4 1
50 11 42 98 9 3 13 1 10 5
50 28 49 46 22 9 19 2 14 8
38 37 43 47 17 26 15 1 13 7
26 36 33 37 7 6 78 6 11 3
19 70 32 19 6 12 51 27 5 1

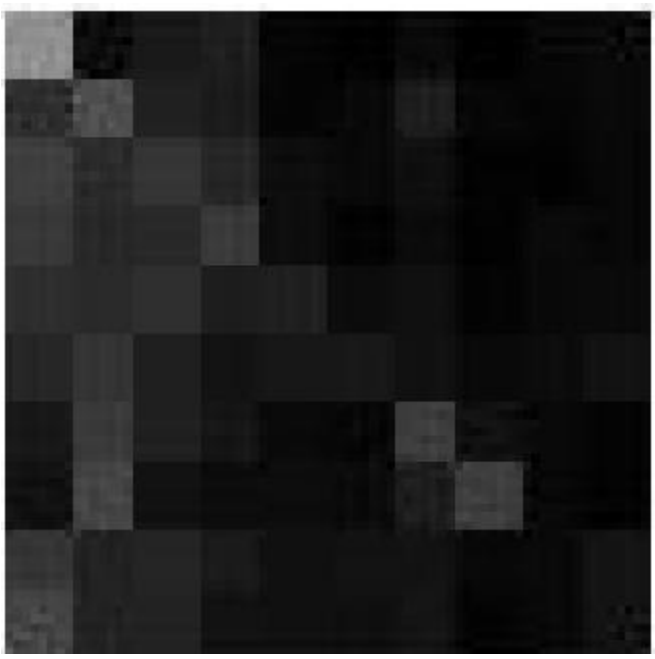
59 23 28 55 5 12 25 1 24 10
65 29 29 43 11 10 21 5 15 11
第 1 类食物图片的识别召回率为 0.5040，虚警率为 0.1882
第 2 类食物图片的识别召回率为 0.1976，虚警率为 0.1222
第 3 类食物图片的识别召回率为 0.1862，虚警率为 0.1421
第 4 类食物图片的识别召回率为 0.4050，虚警率为 0.1968
第 5 类食物图片的识别召回率为 0.0891，虚警率为 0.0333
第 6 类食物图片的识别召回率为 0.1066，虚警率为 0.0291
第 7 类食物图片的识别召回率为 0.3210，虚警率为 0.1023
第 8 类食物图片的识别召回率为 0.1116，虚警率为 0.0073
第 9 类食物图片的识别召回率为 0.0992，虚警率为 0.0405
第 10 类食物图片的识别召回率为 0.0460，虚警率为 0.0195



在取前面 10 维，k 取 4 时候的结果是
识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：
144 13 25 30 6 2 10 1 7 10
46 78 33 27 5 9 29 7 5 9
63 43 53 38 14 9 15 3 2 7
56 40 41 58 13 3 11 3 10 7
46 42 47 29 31 13 16 4 9 10
38 52 31 19 23 24 17 10 12 18
24 58 32 22 10 7 61 16 9 4
17 69 18 12 13 11 29 62 8 3
54 34 37 26 16 19 18 8 10 20
69 34 34 18 17 15 20 4 9 19

第 1 类食物图片的识别召回率为 0.5806，虚警率为 0.1882
第 2 类食物图片的识别召回率为 0.3145，虚警率为 0.1755
第 3 类食物图片的识别召回率为 0.2146，虚警率为 0.1358

第 4 类食物图片的识别召回率为 0.2397，虚警率为 0.1005
第 5 类食物图片的识别召回率为 0.1255，虚警率为 0.0533
第 6 类食物图片的识别召回率为 0.0984，虚警率为 0.0400
第 7 类食物图片的识别召回率为 0.2510，虚警率为 0.0750
第 8 类食物图片的识别召回率为 0.2562，虚警率为 0.0255
第 9 类食物图片的识别召回率为 0.0413，虚警率为 0.0323
第 10 类食物图片的识别召回率为 0. 0795，虚警率为 0.0399

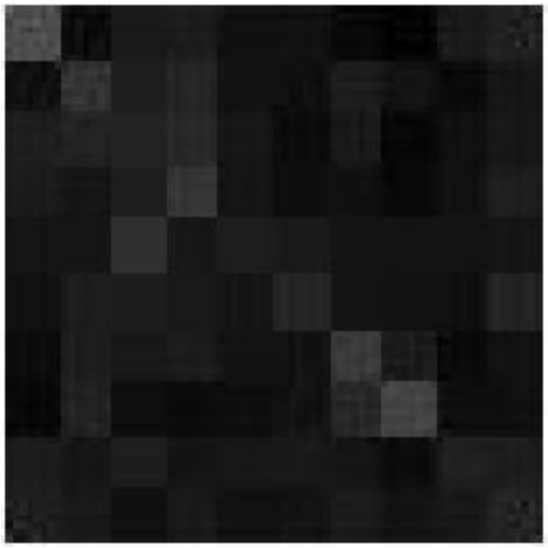


在取前面 10 维，k 取 1 时候的结果是
识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

77	13	26	28	17	17	9	3	25	33
17	52	30	37	17	15	29	19	10	22
30	35	35	40	23	12	25	7	17	23
27	21	30	54	21	15	16	9	18	31
21	21	48	23	29	26	20	19	19	21
17	28	29	24	21	37	20	17	16	35
12	30	24	28	18	18	58	27	10	18
8	34	15	12	15	19	40	68	15	16
28	22	30	20	23	25	21	12	32	29
31	25	17	25	17	19	27	19	24	35

第 1 类食物图片的识别召回率为 0.3105，虚警率为 0.0871
第 2 类食物图片的识别召回率为 0.2097，虚警率为 0.1044
第 3 类食物图片的识别召回率为 0.1417，虚警率为 0.1134

第 4 类食物图片的识别召回率为 0.2231，虚警率为 0.1077
第 5 类食物图片的识别召回率为 0.1174，虚警率为 0.0784
第 6 类食物图片的识别召回率为 0.1516，虚警率为 0.0755
第 7 类食物图片的识别召回率为 0.2387，虚警率为 0.0941
第 8 类食物图片的识别召回率为 0.2810，虚警率为 0.06
第 9 类食物图片的识别召回率为 0.1322，虚警率为 0.07
第 10 类食物图片的识别召回率为 0. 1464，虚警率为 0.1035



1.1.4 实验相关代码说明

code/pca
main2.m 程序入口
loadData.m 加载数据
pc_evectors.m　在这个函数中先求 Imgs 矩阵的协方差矩阵，然后求取该协方差矩阵的特征向量和相应的特征值，将特征值进行从大到小排序，并且相应的特征向量也跟着排序并且进行归一化。返回的 Vecs 是相应特征向量矩阵，Vals 是特征值
sortVV　将相应的特征值和特征向量排序

sortresult　按照 k 近邻方法将计算的距离排序 并且返回相应的结果

1.2 使用 Fisher 对于两类和多类问题降维简单分析

因为使用 PCA 降维再使用 knn 分类效果并不好，而我们知道使用 PCA 降维不考虑各类之间的差异，而 Fisher 降维的时候考虑了类之间的差异。所以我决定在使用 PCA 降维之后再使用 Fisher 进行降维。最后训练集合为 7293×9 的矩阵，

测试图片集合为 2400×9 的矩阵。

1.2.1 两类情况

首先我们分析一下对于两类的情况下的Fisher准则函数,设样本d维特征空间中描述，则两类别问题中线性判别函数的一般形式可表示成 $g(X) = W^T X + w_0$ ，其中 WT 表示垂直于超平面的法向量，在二维的情况下，便是判别直线的法向量，W0 称为阈权值，它只决定超平面在空间上的上下或者左右平移的位置。在使用线性分类器时，样本的分类由其判别函数值决定，而每个样本的判别函数值是其各分量的线性加权和再加上一阈值 w0。如果我们只考虑各分量的线性加权和，则它是各样本向量与向量 W 的向量点积。如果向量 W 的幅度为单位长度，则线性加权和又可看作各样本向量在向量 W 上的投影。显然样本集中向量投影的分布情况与所选择的 W 向量有关。如下图：

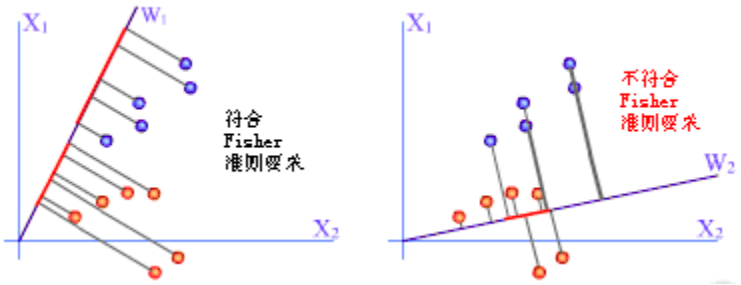


图 1

红色跟蓝色分别为两类样本，显然，从分类的角度来看，W1 要比 W2 要好，因此，Fisher 准则函数的基本思路是向量 W 的方向选择应能使两类样本投影的均值之差尽可能大些，而使类内样本的离散程度尽可能小。

为了给出 Fisher 准则函数的数学定义,我们必须定义一些基本参量,如下：

(1) 各类样本均值向量 mi

$$m_i = \frac{1}{N_i}\sum_{x \in X_i} x \qquad i=1,2$$

(2) 样本类内离散度矩阵 Si 与总类内离散度矩阵 Sw

$$S_i = \sum_{X \in \omega_i} (X - m_i)(X - m_i)^T, \qquad i=1,2$$

$$S_w = S_1 + S_2$$

注释：类内离散矩阵 Si 在形式上与协方差矩阵很相似，但协方差矩阵是一种期望值，而类内离散矩阵只是表示有限个样本在空间分布的离散程度

2 在一维 Y 空间

(1) 各类样本均值$\tilde{m}_i$

$$\tilde{m}_i = \frac{1}{N_i}\sum_{y\in Y_i} y, \qquad i=1,2$$

(2) 样本类内离散度$\tilde{S}_i^2$ 和总类内离散度$\tilde{S}_w$

$$\tilde{S}_i^2 = \sum (y - \tilde{m}_i)^2, \qquad i=1,2$$

$$\tilde{S}_w = \tilde{S}_1^2 + \tilde{S}_2^2$$

在定义了上述一系列描述量后，可以用这些量给出 Fisher 准则的函数形式。根据 Fisher 选择投影方向 W 的原则，即使原样本向量在该方向上的投影能兼顾类间分布尽可能分开，类内样本投影尽可能密集的要求，用以评价投影方向 W 的函数为：

$$J_F(W) = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{S}_1^2 + \tilde{S}_2^2}$$

显然，准则函数的函数值跟总类内离散度成反比，跟样本差值的均方成正比，也就说，两类样本的均值相差越大，函数值越大，反之，则越小，类内离散度越小，函数值越大，反之则越小。同一类的样本，离散度应该要小。

$$J_F(W) = \frac{W^T S_B W}{W^T S_W W}$$

使用拉格朗日方法求解$J_F(W)$极值

定义拉格朗日函数$L(W,\lambda) = W^T S_B W - \lambda(W^T S_W W - c)$

对其进行求导$S_W^{-1} S_B W^* = \lambda W^*$

$S_B W^* = (M_1 - M_2)R$其中 $R = (M_1 - M_2)^T W^*$是一标量

$$W^* = \frac{R}{\lambda} S_W^{-1}(M_1 - M_2)$$

归一化处理

$$W_0 = \frac{W^*}{\|W^*\|} = \frac{S_W^{-1}(M_1 - M_2)}{\|S_W^{-1}(M_1 - M_2)\|}$$

Fisher 其实就是将各个类的中心按照一定方向做了变换。

1.2.2 在多类的情况下面比较特殊

在有两个类的情况下，在 Fisher 判别推导过程中使用的分析，可以扩展到找到一个子空间，这似乎包含所有的不同的类。假设 C 个类别中的每个类有一个均值u_i和相同的协方差$\sum$。然后，类间不同的分散性可以由类的样本协方差定义

$$\sum_b = \frac{1}{C}\sum_{i=1}^{C}(\mu_i - \mu)(\mu_i - \mu)^T$$

其中μ 是所有类的均值的均值，类按照方向$\vec{w}$ 分隔，给出相应的公式为

$$S = \frac{\vec{w}^T \sum_b \vec{w}}{\vec{w}^T \sum \vec{w}}$$

这意味着$\vec{w}$是 $\sum^{-1}\sum_b$特征向量，分隔等于相应的特征值，如果$\sum^{-1}\sum_b$可以对角化，则取最大的 C-1（C 为类数）个特征值对应的特征向量。

1.2.3 实验结果及相关分析

在取前面 40 维，k 取 4 时候的结果是

识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

132	21	22	31	6	4	9	0	11	12
36	75	35	21	10	16	32	13	4	6
66	41	53	32	10	8	13	7	9	8
74	30	42	52	8	3	9	2	12	10
55	36	39	26	28	15	19	7	11	11
52	43	33	21	17	15	22	10	14	17
32	51	32	23	14	9	50	18	7	7
16	72	20	8	9	14	30	63	8	2
54	45	40	31	13	11	14	4	17	13
77	40	20	19	19	15	10	8	11	20

第 1 类食物图片的识别召回率为 0.5323，虚警率为 0.2106
第 2 类食物图片的识别召回率为 0.3024，虚警率为 0.1727
第 3 类食物图片的识别召回率为 0.2149，虚警率为 0.1289

第 4 类食物图片的识别召回率为 0.2149，虚警率为 0.0964
第 5 类食物图片的识别召回率为 0.1134，虚警率为 0.0483
第 6 类食物图片的识别召回率为 0.0645，虚警率为 0.0432
第 7 类食物图片的识别召回率为 0.2058，虚警率为 0.0719
第 8 类食物图片的识别召回率为 0.2603，虚警率为 0.0314
第 9 类食物图片的识别召回率为 0.0702，虚警率为 0.0395
第 10 类食物图片的识别召回率为 0.0837，虚警率为 0.0390

1.2.4 实验相关代码说明

code/pca+fisher
pcaPlusFisher.m 程序入口
loadData.m 加载数据
pc_evectors.m　在这个函数中先求 Imgs 矩阵的协方差矩阵，然后求取该协方差矩阵的特征向量和相应的特征值，将特征值进行从大到小排序，并且相应的特征向量也跟着排序并且进行归一化。返回的 Vecs 是相应特征向量矩阵，Vals 是特征值
FDA.m 求解 Fisher 方法的转换矩阵
sortVV　将相应的特征值和特征向量排序

sortresult　按照 k 近邻方法将计算的距离排序 并且返回相应的结果

2 问题背景

使用其他特征和方法进行 10 个类别食物识别，比如 HOG 特征+SVM 分类方法，CNN 分类等。只需实现一种算法即可，特征和分类器均可以使用开源代码（选做：实现多种算法）同 PCA/Fisher 一样，评价该方法的性能。

2.1 使用 HOG 特征+SVM 分类方法

2.1.1 hog 特征相关原理介绍

Dalal 提出的 Hog 特征提取的过程：把样本图像分割为若干个像素的单元（cell），把梯度方向平均划分为 9 个区间（bin），在每个单元里面对所有像素的梯度方向在各个方向区间进行直方图统计，得到一个 9 维的特征向量，每相邻的 4 个单元构成一个块（block），把一个块内的特征向量联起来得到 36 维的特征向量，用块对样本图像进行扫描，扫描步长为一个单元。最后将所有块的特征串联起来，就得到了人体的特征。例如，对于 64*128 的图像而言，每 16*16 的像素组成一个 cell，每 2*2 个 cell 组成一个块，因为每个 cell 有 9 个特征，所以每个块内有 4*9=36 个特征，以 8 个像素为步长，那么，水平方向将有 7 个扫描窗口，垂直方向将有 15 个扫描窗口。也就是说，64*128 的图片，总共有 36*7*15=3780 个特征。为改善准确率，局部直方图可以通过计算图像中一个较大区域(称为block)的光强作为measure被对比标准化，然后用这个值(measure)归一化这个block中的所有cells.这个归一化过程完成了更好的照射/阴影不变性。与其他描述子相比，HOG 得到的描述子保持了几何和光学转化不变性（除非物体方向改变）。HOG 特征提取方法就是将一个 image：灰度化（将图像看做一个 x,y,z（灰度）的三维图像），划分成小 cells（2*2），计算每个 cell 中每个 pixel 的 gradient（即 orientation），统计每个 cell 的梯度直方图（不同梯度的个数），即可形成每个 cell 的 descriptor。实现时使用了 vlfeat 库来提取 hog 特征

2.1.2 svm 相关原理介绍

在 svm 算法的变种中，为了允许数据点在一定程度上偏离一下超平面，加上了松弛变量。原来的约束条件为：$y_i(w^T x_i + b) \geq 1, i = 1,2,\dots,n$

考虑会偏离的问题，约束条件变成了：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1,2,\dots,n$$

其中ξ_i是松弛变量，对应数据点x_i允许偏离的 functional margin 的量。当然，如果我们运行ξ_i任意大的话，那任意的超平面都是符合条件了。所以，我们在原来的目标函数后面加上一项，使得这些ξ_i的总和最小：

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

其中 C 是一个常数，用于控制目标函数中两项（寻找 margin 最大的超平面和保证数据点偏差最小）之间的权重。注意，其中 ξ 是需要优化的变量之一，而 C 是一个之前确定好的变量

使用 svm]分类方法

$$\min \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

s. t. $y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1,2,\dots,n$

$$\xi_i \geq 0, i = 1,\dots,n$$

相应的对偶公式为

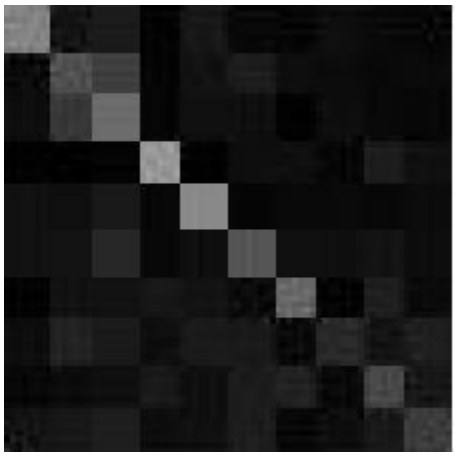
$$\text{Minimize } \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j y_i y_j \overrightarrow{x_i}\,\overrightarrow{x_j} \text{ ,subject to } 0 \leq \alpha_i \leq C \text{ and } \sum_{i=1}^{N}\alpha_i y_i = 0$$

2.1.3 实验结果及相关分析

识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

131	20	27	3	23	8	6	14	9	7
11	80	63	5	18	28	14	11	8	10
14	56	108	6	16	15	2	14	9	7
2	1	4	152	2	14	15	7	28	17
17	15	25	8	137	6	9	10	8	12
17	19	39	8	14	85	15	15	19	13

4	18	18	23	19	12	104	4	31	10
14	34	27	14	26	25	11	40	20	31
8	12	15	25	15	28	33	10	72	24
20	24	30	13	16	29	8	14	25	60



第 1 类食物图片的识别召回率为 0.5382，虚警率为 0.0488
第 2 类食物图片的识别召回率为 0.3226，虚警率为 0.0907
第 3 类食物图片的识别召回率为 0.4372，虚警率为 0.1130
第 4 类食物图片的识别召回率为 0.6281，虚警率为 0.0477
第 5 类食物图片的识别召回率为 0.5547，虚警率为 0.0679
第 6 类食物图片的识别召回率为 0.3484，虚警率为 0.0751
第 7 类食物图片的识别召回率为 0.4280，虚警率为 0.0514
第 8 类食物图片的识别召回率为 0.1653，虚警率为 0.0450
第 9 类食物图片的识别召回率为 0.2975，虚警率为 0.0713
第 10 类食物图片的识别召回率为 0.2510，虚警率为 0.0595

2.1.4 实验相关代码说明

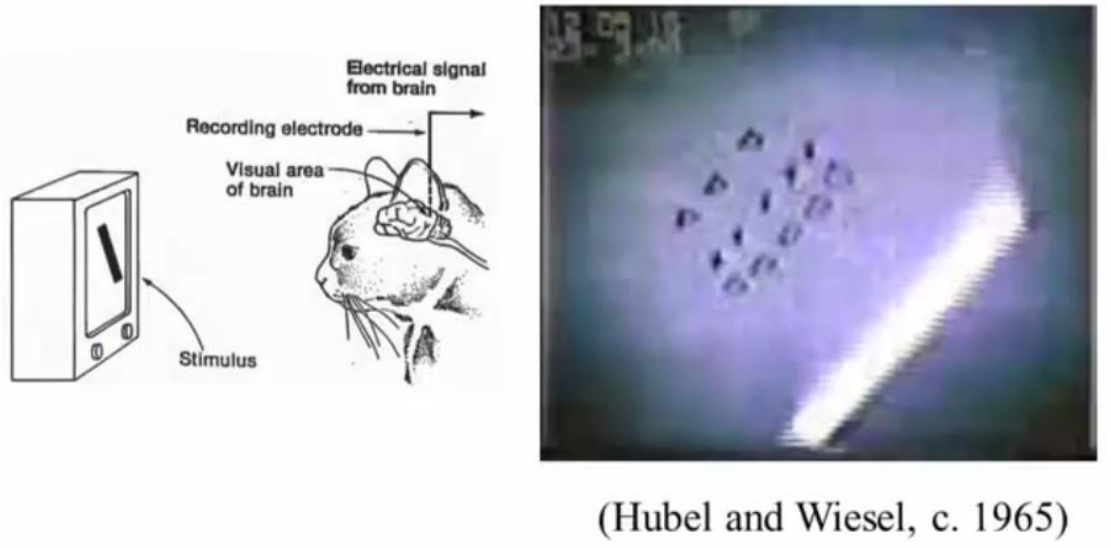
code/hog+svm
pro2.m 程序入口，在程序实现的时候训练了 10 个模型，每个模型把使用一个类别的实物图片提取特征作为正样本，把其他类型食物图片提取的特征作为负样本。提取了特征之后使用 pca 进行降维，使用 libsvm 训练这十个模型，测试的时候对于 10 个模型均进行测试，看它从属于哪一类的概率最高就判别为该类图片。

loadTestData2.m 加载测试数据的 hog 特征
pc_evectors.m　在这个函数中先求 Imgs 矩阵的协方差矩阵，然后求取该协方差矩阵的特征向量和相应的特征值，将特征值进行从大到小排序，并且相应的特征向量也跟着排序并且进行归一化。返回的 Vecs 是相应特征向量矩阵，Vals 是特征值

sortVV　将相应的特征值和特征向量排序

2.2 使用 CNN 分类方法

卷积神经网络是一个多层感知器的变种，这种方法由生物学启发，在动物大脑中有一些视觉皮质，在这些视觉皮质存在复杂的排列，它们对于输入空间的子区域敏感，被称为接受域，并且以这种方式来覆盖整个视觉区域。这些滤波器在输入空间是局部的并且能够更好的开发原始图片中的空间局部相关性。



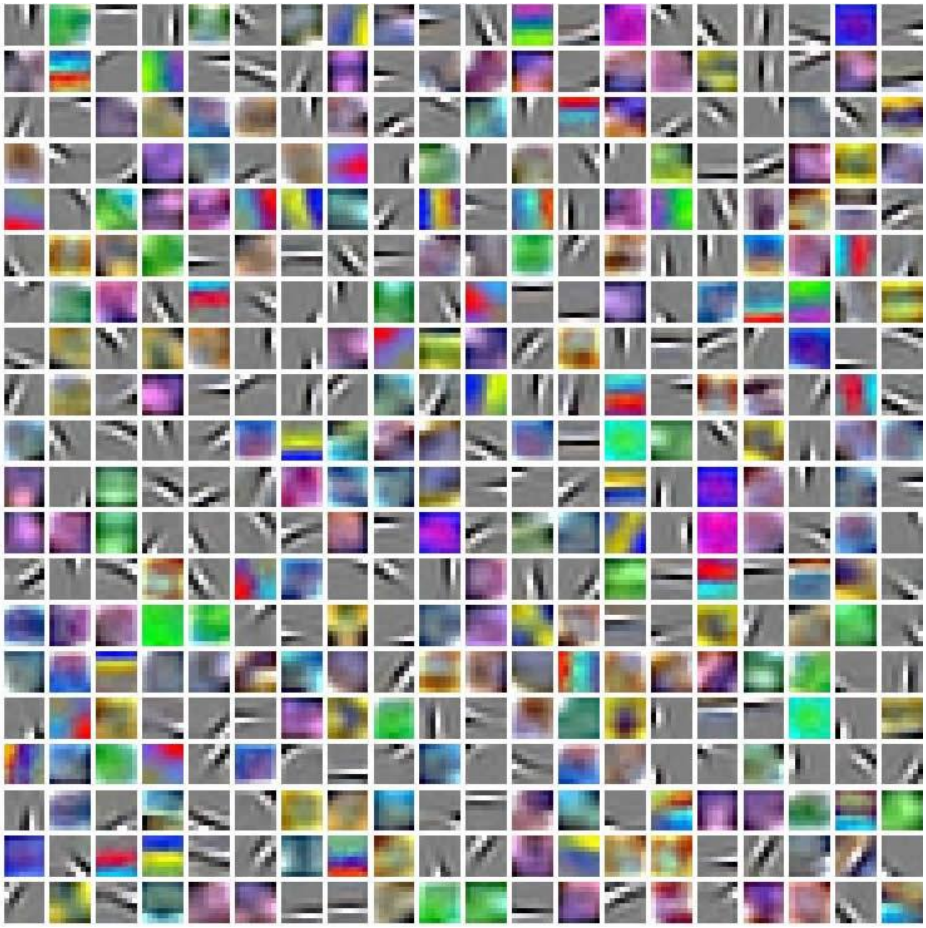
2.2.1 稀疏自编码

另外传统计算开销非常大，而使用 cnn 可以减少开销，使用稀疏自编码技术（sparse auto encoder），全连接所有的影藏层单元到输入单元开销巨大，简单方法就是限制影藏层和输入层的链接，使得每个影藏层单元只链接输入层的一个小的子集。特别是每个影藏层单元将只用连接输入中一个小的连续的区域。更确切地说，已经学习小的（8*8）patches，从更大的图像随机采样，我们可以把这个学会的 8x8 的特征检测图像的任何位置。具体来说，我们可以把学到的 8x8 的特征与较大的图像进行卷积，从而在每个图像中的位置获得不同的特征激活值。使用稀疏自编码和输出层的解码来学习了 8*8patches 特征。稀疏性可以被简单地解释如下。如果当神经元的输出接近于 1 的时候我们认为它被激活，而输出接近于 0 的时候认为它被抑制，那么使得神经元大部分的时间都是被抑制的限制则被称作稀疏性限制。这里我们假设的神经元的激活函数是 sigmoid 函数。如果你使用 tanh 作为激活函数的话，当神经元输出为-1 的时候，我们认为神经元是被

抑制的。注意到 $a_j^{(2)}$ 表示隐藏神经元 j 的激活度，但是这一表示方法中并未明确指出哪一个输入 x 带来了这一激活度。所以我们将使用 $a_j^{(2)}(x)$ 来表示在给定输入为 x 情况下，自编码神经网络隐藏神经元 j 的激活度。 进一步，让

$$\hat{\rho}_j = \frac{1}{m}\sum_{i=1}^{m}\left[a_j^{(2)}(x^{(i)})\right]$$

表示隐藏神经元 j 的平均活跃度（在训练集上取平均）。



然后把这些特征，400 个过滤器每个和整个图像进行卷积

Figure (a): Image 3 convoluted with 2nd feature

Figure (b): Image 3 convoluted with 3rd feature

2.2.2 卷积操作

输入的图像维数为 64x64 像素，被 400 个 8x8 特征过滤器进行卷积，得到了 (64-8+1) x (64-8+1) x 400 维度的矩阵，这些 400 57x57 矩阵图像就是特征地图。

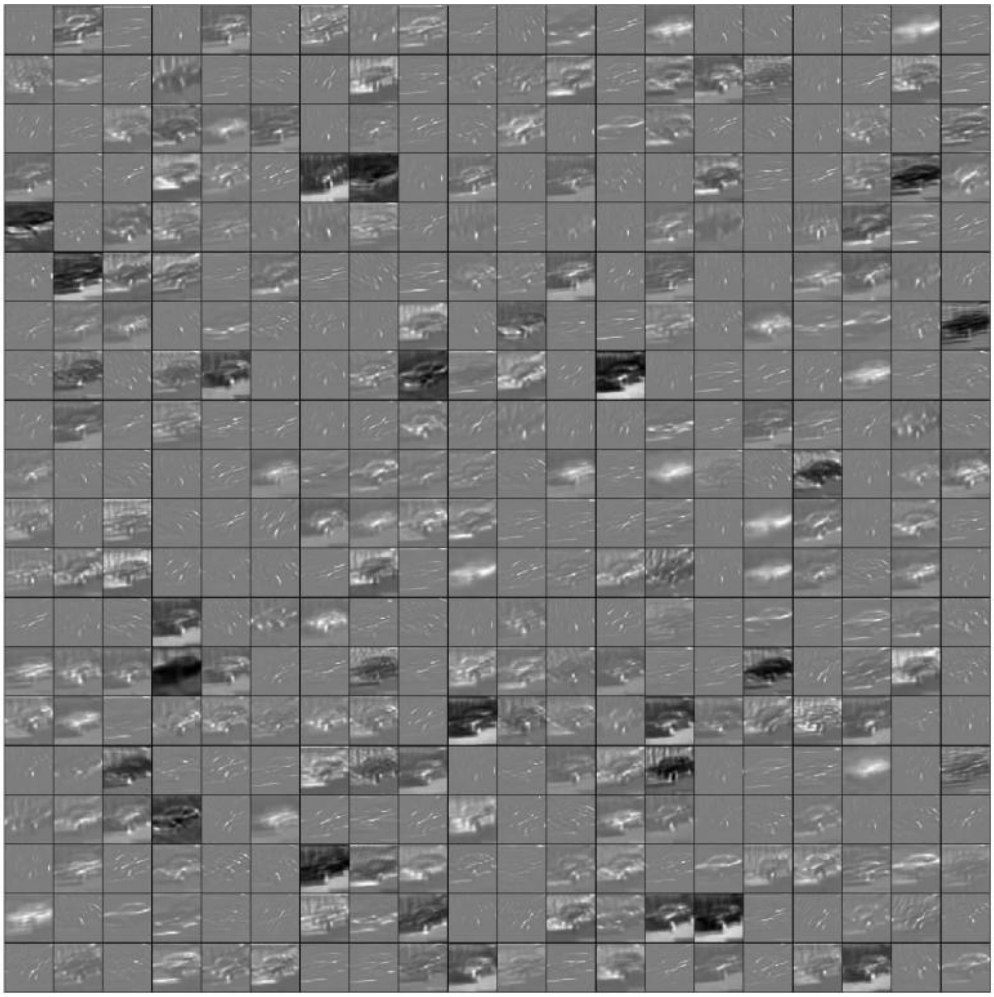


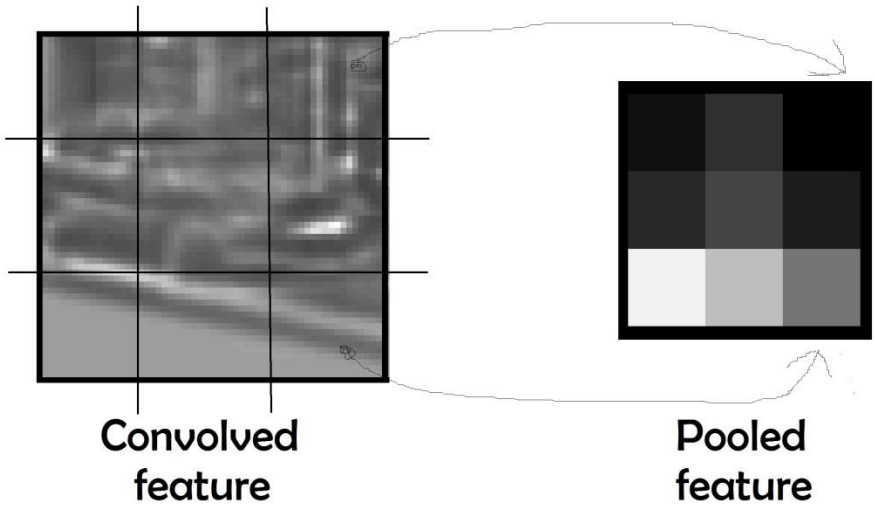
Figure 7: Convoluted Features of Car (Image 2).

使用卷积获取特征之后，我们接下来使用它们进行分类，理论上可以使用分类器比如 softmax 对于提取出来的特征进行分类。但是这对于计算太有挑战性了，考虑图像大小的实例，64x64 像素和 400 特征，每个图品例子有 1,299,600 个特征。有一百万特征输入学习分类器会是过拟合的。

2.2.3 池化 pooling

为了解决这个问题通常做法是对不同区域的这些特征统计。可以使用一个特定特征的平均和最大值，这种方法称为 pooling。使用卷积后的特征是因为图像具有一种“静态性”的属性，这也就意味着在一个图像区域有用的特征极有可能在另一个区域同样适用。因此，为了描述大的图像，一个很自然的想法就是对不同位置的特征进行聚合统计，例如，人们可以计算图像一个区域上的某个特定特征的平均值（或最大值）。这些概要统计特征不仅具有低得多的维度（相比使用

所有提取得到的特征)，同时还会改善结果(不容易过拟合)。这种聚合的操作就叫做池化 (pooling)，有时也称为平均池化或者最大池化 (取决于计算池化的方法)。



2.2.4 softmax 分类

经过这个之后使用 softmax 来对特征进行学习分类。在 softmax 回归中，我们解决的是多分类问题（相对于 logistic 回归解决的二分类问题），类标 y 可以取 k 个不同的值（而不是 2 个）。因此，对于训练集 $\{(x^{(1)},y^{(1)}),\ldots,(x^{(m)},y^{(m)})\}$，我们有 $y^{(i)}\in\{1,2,\ldots,k\}$。(注意此处的类别下标从 1 开始，而不是 0)。例如，在这次的食物识别任务中，我们有 $k=10$ 个不同的类别。对于给定的测试输入 x，我们想用假设函数针对每一个类别 j 估算出概率值 $p(y=j|x)$。也就是说，我们想估计 x 的每一种分类结果出现的概率。因此，我们的假设函数将要输出一个 k 维的向量（向量元素的和为 1）来表示这 k 个估计的概率值。 具体地说，我们的假设函数 $h_\theta(x)$ 形式如下：

$$h_\theta(x^{(i)})=\begin{bmatrix}p(y^{(i)}=1|x^{(i)};\theta)\\p(y^{(i)}=2|x^{(i)};\theta)\\\vdots\\p(y^{(i)}=k|x^{(i)};\theta)\end{bmatrix}=\frac{1}{\sum_{j=1}^{k}e^{\theta_j^T x^{(i)}}}\begin{bmatrix}e^{\theta_1^T x^{(i)}}\\e^{\theta_2^T x^{(i)}}\\\vdots\\e^{\theta_k^T x^{(i)}}\end{bmatrix}$$

其中 $\theta_1, \theta_2, \ldots, \theta_k \in \Re^{n+1}$ 是模型的参数。请注意 $\frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}}$ 这一项对概率分布进行归一化，使得所有概率之和为 1 。

我们通过添加一个权重衰减项 $\frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$ 来修改代价函数，这个衰减项会惩罚过大的参数值，现在我们的代价函数变为：

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}\theta_{ij}^2$$

有了这个权重衰减项以后 $(\lambda > 0)$，代价函数就变成了严格的凸函数，这样就可以保证得到唯一的解了。 此时的 Hessian 矩阵变为可逆矩阵，并且因为 $J(\theta)$是凸函数，梯度下降法和 L-BFGS 等算法可以保证收敛到全局最优解。

为了使用优化算法，我们需要求得这个新函数 $J(\theta)$ 的导数，如下：

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}(1\{y^{(i)} = j\} - p(y^{(i)} = j|x^{(i)};\theta))\right] + \lambda\theta_j$$

通过最小化 $J(\theta)$，我们就能实现一个可用的 softmax 回归模型。

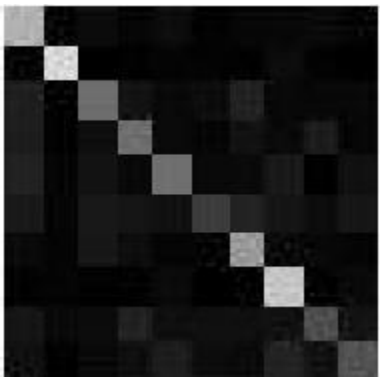
2.2.5 实验结果及相关分析

识别矩阵（对角线为每个类别的正确识别数量，非对角线为每行对应类别识别为每列对应类别的数量）：

177	6	11	5	12	2	7	7	10	5
5	216	0	1	5	4	2	11	3	0
31	2	110	19	9	13	34	8	7	8
28	3	14	111	10	6	18	10	28	6
33	6	18	9	111	9	10	25	1	17
24	6	24	20	17	60	34	21	12	22
10	5	23	13	5	6	153	9	8	5
2	5	2	3	11	0	3	197	3	11
21	9	12	34	15	16	16	18	78	20
16	1	10	15	29	13	10	35	18	82

第 1 类食物图片的识别召回率为 0.7314，虚警率为 0.0793
第 2 类食物图片的识别召回率为 0.8745，虚警率为 0.0201
第 3 类食物图片的识别召回率为 0.4564，虚警率为 0.0532
第 4 类食物图片的识别召回率为 0.4744，虚警率为 0.0553

第 5 类食物图片的识别召回率为 0.4644，虚警率为 0.0527
第 6 类食物图片的识别召回率为 0.25，虚警率为 0.0322
第 7 类食物图片的识别召回率为 0.6456，虚警率为 0.0624
第 8 类食物图片的识别召回率为 0.8312，虚警率为 0.00670
第 9 类食物图片的识别召回率为 0.3264，虚警率为 0.00419
第 10 类食物图片的识别召回率为 0.3581，虚警率为 0.0436



2.2.6 实验相关代码说明

这个程序实现借鉴了 stanford 的深度学习相应资料
http://deeplearning.stanford.edu/tutorial/
https://github.com/amaas/stanford_dl_ex
code/cnn/cnn
loadTrainDataForCnn.m 从图片中提取相应的特征并且保存为 stlTestSubsetMy，stlTrainSubsetMy
cnnExercise.m 是进行 cnn 训练测试的程序入口
cnnConvolve.m 进行卷积操作的函数
cnnPool.m 进行池化操作的函数
resultMatrix.m 计算相应的召回率和虚警率
code/cnn/linear_decoder_exercise
linearDecoderExercise.m 存储了生成稀疏自便码学习特征的程序入口
displayColorNetwork.m 显示学习的特征结果

code/cnn/common 调用的库函数