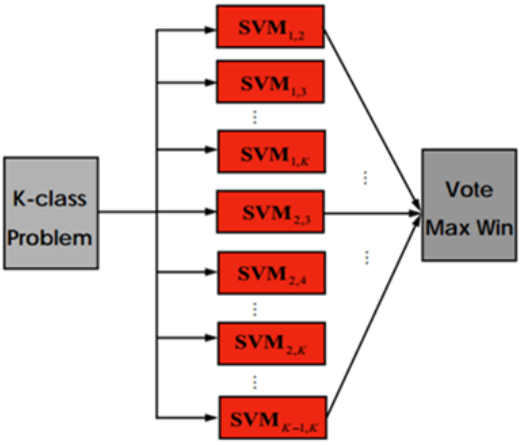
Neural Network Theory and Applications Homework Assignment 1

林煜

2017 年 4 月 5 日

1. 本次实验采用C++编写，程序见nnhw3.cpp。以下对one-versus-one和one-versus-rest实现进行简要的介绍。

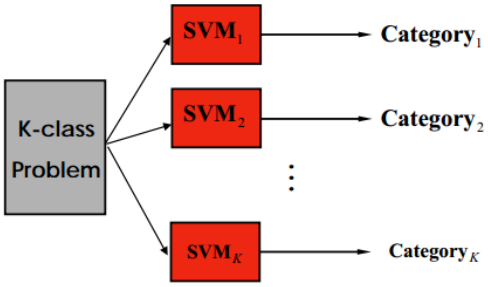
   (a) one-versus-one
   由于在实验过程中可以知道该问题是一个12分类问题。故针对one-versus-one，需要训练$12 * (12-1)/2 = 66$ 个二分类器。
   对于每一个二分类器$M^{i,j}$，均采用同一组参数，训练数据为类别i 及类别j 数据的并集。构造好参数后，之后调用接口svm-train进行训练，获得各个分类器的训练模型。
   最后读取测试文件进行测试，在one-versus-one中，使用svm-predict接口，对于每一个分类器都将得到一个分类结果，汇总这66 个结果，我们选取出现最多的类别作为最终的结果。
   以下是总体结构图：



   这里，选择的svm类型为C_SVC, kernel类型为RBF，惩罚因子C为500，参数gamma为0.5。其他参数均随意设置(其实惩罚因子也基本没有影响)。对于测试集，得到结果：正确率为64.7486%。

   (b) one-versus-rest
   对于one-versus-rest，实验过程中，将训练12个二分类器$M^{i,-i}$。
   其中每个分类器使用同一组参数（也使用ovo的参数），训练数据为类别i 及除了i以外的其他类别的总和作为另一类。注意此时i类标签为1，其他类之和标签为-1。之后调用接口svm-train进行训练，获得各个分类器的训练模型。
   最后读取测试文件进行测试，由于采用one-versus-rest，故需使用svm-predict-values接口获得每一个二分类器返回的值及预测的类别。选取最大的值对应的类别作为分类结果。（如果最大的值为负数，则分类出错，没有分类结果）

以下是总体结构图：



对于测试集，得到结果：正确率为36.521%。相比较ovo，正确率明显下降，由于ovr采用剩下所有类的数据作为一个类别，此时将会出现很严重的二分类数据不均衡，测试会更偏向于数据多的类别，故正确率相比ovo显著下降。

2. 对于part-versus-part，是基于one-versus-one。由于实验过程中我们可以知道每一个类别的训练数据个数，发现类别间相差很大，所以对于一个二分类问题，如果两个分类类别数据相差较大的话，分类器将更偏向于数据量大的类别。这种数据不均衡性可以通过划分数据集，采用$Min-Max$网络进行转换。另外划分数据还能够将大的训练任务分解成小的训练任务，从而提高并行性，缩短训练时间。所以对one-versus-one中的每一个分类器，设计$Min-Max$网络进行转换。即可完成分类。
结构图如下：

由于要划分数据集，这里我设计了一个划分准则。
对于两个训练样本集，数据量分别为a和b，需要将a和b分别划分为x份和y份，由于我们需要尽量让数据集数量均衡，即不同类别的数据量差值最小，由于划分后每个数据集个数为a/x和b/y，一共有$x*y$个分类器，得

$$f(x,y) = xy|a/x - b/y| \tag{1}$$
$$= |ay - bx| \tag{2}$$

又由于这样的话将得到$x = a, y = b$，所以我们需要在最小化差值与划分个数间做tradeoff，所以加入划分个数的项，这里令其为$\theta(x+y)$。于是得：

$$f(x,y) = xy|a/x - b/y| + \theta(x+y) \tag{3}$$
$$= |ay - bx| + \theta(x+y) \tag{4}$$

所以问题就转换为下面这样一个整数规划问题：

$$min \qquad f(x,y) = |ay - bx| + \theta(x+y) \tag{5}$$
$$s.t. \qquad 1 \le x \le a \tag{6}$$
$$1 \le y \le b \tag{7}$$

x,y均为整数。由于a和b在此题中均比较小，我并没有采用什么高级的算法（分支定界法等）求解，直接遍历x，y的值，得到划分准则。
划分完数据后训练xy个分类器。以这些分类器作为Min-Max网络组成部分求解二分类问题。
测试部分跟one-versus-one过程差不多，唯一不同的是原来一个二分类器直接得到分类结果，现在需要组织Min-Max求解得到结果。之后跟ovo一样采用voting的方式决定最终分类。
测试结果为：在θ设置成1的情况下，一共划分成了2417个分类器，正确率为31.4427%。效果很差，所以我又使用了不同的θ值，如下表：

θ \ 运行结果	分类器个数	测试正确率
1	2417	31.4427%(1907/6065)
2	2066	34.3611%(1907/6065)
5	1197	40.4617%(2454/6065)
10	829	41.418%(2512/6065)
1000	66	64.7486%(3927/6065)

从上表可知，随着θ值的变大，分类器个数变小（原因在于式5），测试正确率在逐渐提升。当将θ设置成1000时，pvp完全退化成ovo，且得到了完全一样的结果，说明程序实现没有问题。正确率不高的原因可能是参数需要调整。

3. 分别采用RBF和LINEAR得到的结果如下面表格：

$$One - versus - One$$

核函数 \ 运行结果	测试正确率
LINER	62.3743%(3783/6065)
RBF	64.7486%(3927/6065)

$$One - versus - Rest$$

核函数 \ 运行结果	测试正确率
LINER	34.0313%(2064/6065)
RBF	36.521%(2215/6065)

$$Part - versus - Part(\theta = 1)$$

核函数 \ 运行结果	测试正确率
LINER	30.7007%(1862/6065)
RBF	31.4427%(1907/6065)

从上述结果来看采用这两种不同的核函数对结果没有太大影响（虽然LINER正确率均会低一点点）。RBF效果优于LINER是可预见的，毕竟RBF将低维向量映射成高维向量做分类，当RBF并不做这种映射时将退化成LINER。

4. (a) One-versus-One
   优点：思想简单，且效果出众。
   缺点：由于要训练$classnum(classnum-1)/2$个分类器，当类别相当大的时候，训练代价很高。

   (b) One-versus-Rest
   优点：思想简单，训练分类器个数很少，与类别个数一样。
   缺点：由于训练的数据是i分类与其他所有分类之和，故大部分情况下会存在明显的数据不均衡，分类效果较差。

(c) Part-versus-Part
优点：能够很好的解决数据不均衡问题，且通过划分数据使得训练能够并行是的效率大大提高。
缺点：训练的分类器的数量将非常的大，虽然由于每一个分类器数据量很小，对于训练效率没什么影响。但是在测试时候，如果没有采用并行的方式，测试的成本过高。