# 对Parzen窗/PNN算法的学习和研究报告

姓名：吴潇 学号：1333755

1. **Parzen窗方法综述、发展历史及现状**

模式识别领域的非参数估计方法大致可以分为两类。第一种类型是先估计出概率密度函数的具体形式，然后再利用这个估计出来的概率密度函数对样本进行分类。第二种类型是，不估计具体的概率密度函数，而直接根据样本进行分类。Parzen窗方法就是属于第一种类型的非参数估计方法，概率神经网络（PNN）是它的一种实现方式。Parzen窗方法的基本思想是利用一定范围内的各点密度的平均值对总体密度函数进行估计。

Parzen窗（Parzen window）又称为核密度估计（kernel density estimation），是概率论中用来估计未知概率密度函数的非参数方法之一。该方法由Emanuel Parzen于1962年在*The Annals of Mathematical Statistics* 杂志上发表的论文“*On Estimation of a Probability Density Function and Mode*” 中首次提出。Nadaraya 和 Watson最早把这一方法用于回归法中。Specht把这一方法用于解决模式分类的问题，并且在1990年发表的论文 “*Probabilistic neural networks*” 中提出了PNN网络的硬件结构。Ruppert和Cline基于数据集密度函数聚类算法提出了修订的核密度估计方法，对Parzen窗做了一些改进。

Parzen窗方法虽然是在上个世纪60年代提出来的，已经过去了45年的时间，看上去是一种很“古老”的技术，但是现在依然有很多基于Parzen窗方法的论文发表。这说明Parzen窗方法的确有很强的生命力和实用价值，虽然它也存在很多缺点。

**2、Parzen窗方法和概率神经网络**

Parzen窗方法就是基于当样本个数n非常大的时候，有公式 成立这样的一个事实而提出的。通过计算在一个区域R内的频数k/n，用这个频数来估计这一点的频率，从而得到这一点的概率。当n趋于无穷大的时候，p(**x**) 等于该点的实际概率。这种方法就是模式识别领域中的非参数估计方法。

Parzen窗方法就是通过构造一系列的区域： ，在这些区域内计算k/n。记 Vn为区域Rn的体积，kn为落在区域Rn中的样本个数， 表示对 的第n次估计，于是有：

为了保证 能够收敛到 ，必须满足以下3个条件：

1. 2） 3）

Parzen窗方法的实质就是通过对上面的区域Rn，每次按照 来构造区域序列，使区域逐渐收缩到一个给定的初始区间。它不断收缩区域，按照公式把区域不断缩小，而不关心该区域实际上包含多少个样本点。另外一种与它相对应的非参数估计方法是Kn-近邻法。

假设区间 Rn 是一个 d 维的超立方体，hn 表示超立方体的一条边的长度，那么该超立方体的体积就是 。通过定义如下的窗函数，我们能够解析地得到落在窗中的样本个数 kn 的表达式：

这样，就表示一个中心在原点的单位超立方体。如果xi落在超立方体Vn中，那么 ，否则便为0。因此，超立方体中的样本个数就是

带入公式 ， 就得到

该方程表明一种更加一般的估计概率密度函数的方法——不必规定区间必须是超立方体，可以是某种更加一般化的形式。这个公式表示我们对 p(**x**) 的估计是对一系列关于**x**和 **xi** 的函数求平均。这个 就是Parzen窗方法估计得到的概率密度函数。

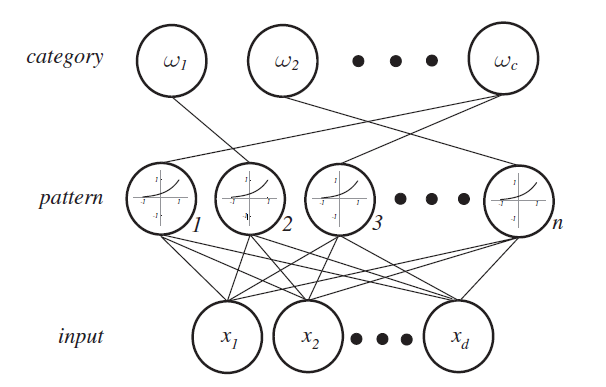
关于 是否合理的问题，也就是判断它是否保证函数值非负，而且积分的结果为1。这一点可以通过增加条件来保证：

1）要求 满足 和

2）要求

增加这些条件就可以保证 是一个合理的概率密度函数，函数值是非负的，积分的结果为1。

Parzen窗方法可以使用神经网络的方法来实现，也就是通常所说的概率神经网络（Probabilistic neural network, PNN）。现在假设有n个d维的样本，它们都是从c个类别中选取的。在这种情况下，输入层由d个输入单元组成，每一个输入单元都与模式层中的n个模式单元相连。而每一个模式单元只与类别层中的c个类别单元中的其中之一相连。



从输入层到模式层的连线表示可修改的权系数，这些权系数可以通过训练得到。每一个类别单元都计算与之相连的各模式单元的输出结果的和。每一个模式层单元能够对它的权重向量和归一化的样本向量**x**作内积，得到  **，**然后映射为 。每一个类别单元把与它相连的模式层单元的输出结果相加。这样的结果就保证了类别单元处得到的就是使用协方差为 的圆周对称高斯窗函数的Parzen窗的估计结果（**I**表示d × d的单位矩阵）。

PNN网络是用下面的方式训练的。首先，把训练样本中的每一个样本**x**都归一化为单位长度，即 。第一个经过归一化的样本被置于输入层单元上。同时，连接输入单元和第一个模式层单元的那些连接被初始化为 。然后，从模式层的第一个单元到类别层中代表**x**1所属类别的那个单元之间就建立了一个连接。同样的过程对剩下的各个模式单元都重复进行，即 ，其中 k = 1, 2, …, n。这样就得到了一个网络：输入层单元与模式层单元之间是完全连通的，而模式层单元到类别单元之间是稀疏连接的。如果把第j个样本的第k个分量记为xjk，把这个分量到第j个模式层单元的连接权重系数记为wjk，其中j = 1,2,…,n, k = 1, 2, …, d。 得到算法描述如下：

**PNN训练算法**

1 **begin** **initialize** j ← 0,n,aji ← 0, j = 1,…,n； i=1,…,c

2 **do**

3 （归一化过程）

4 （训练）

5 **if** **then**

6 **until**

7 **end**

然后，经过训练完成的网络就可以用这样的方式实现分类：首先把一个归一化了的测试样本**x**提供给输入节点，每一个模式层单元都计算内积，得到“净激活”（net activation）：

并产生 netk的一个非线性函数 ，其中 是由用户设置的一个参数，表示有效的高斯窗的宽度。每一个类别层单元则把与它相连接的模式层单元的结果进行相加。为了实现Parzen窗算法，这里的激活函数必须是一个指数函数。对于一个中心在某一个训练样本**w**k处的未经过归一化的高斯窗函数。从期望得到的高斯窗函数倒推出模式层应采用的非线性活化函数的形式，即如果令有效宽度hn为常数，则窗函数为

其中使用了归一化条件： 。这样一个模式层单元向与它相连接的那个类别层单元贡献了一个信号，这个信号的强度等于以当前训练样本为中心的高斯函数产生这个测试样本点的概率。对这些局部估计值求和就得到判别函数 ——也就是概率密度函数的Parzen窗估计结果。通过 运算得到测试点的期望的类别：

**PNN分类算法**

1 **begin** **initialize** k ← 0， x ← 测试点

2 **do** k ← k + 1

3 netk ←

4 **if** **then**

5 **until** k = n

6 **return** class ←

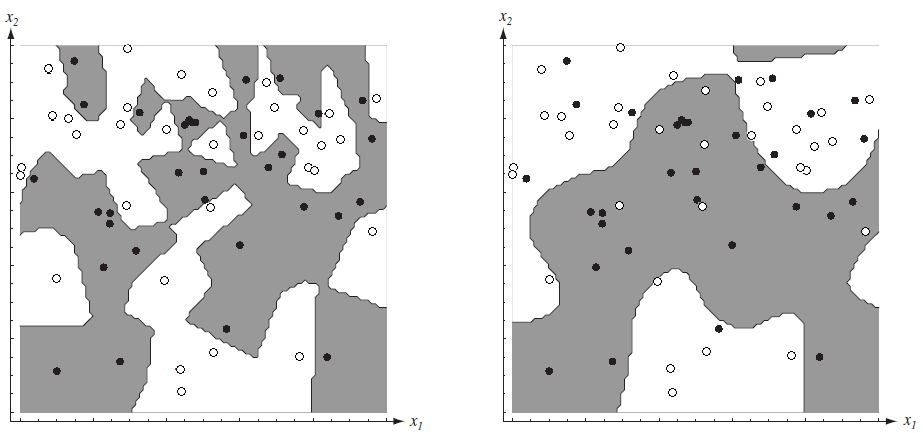
7 **end**

**3、Parzen窗方法的优点和缺点**

Parzen窗方法的好处是不需事先知道概率密度函数的参数形式，比较通用，可以应对不同的概率密度分布形式。在处理有监督学习过程的时候，现实世界的情况往往是我们不知道概率密度函数形式。就算能够给出概率密度函数的形式，这些经典的函数也很少与实际情况符合。所有经典的概率密度函数的形式都是单模的（只有单个局部极大值），而实际情况往往是多模的。非参数方法正好能够解决这个问题，所以从这个意义上来讲，Parzen窗方法能够更好地对应现实世界的概率密度函数，而不必实现假设概率密度函数的形式是已知的。Parzen窗方法能处理任意的概率分布，不必假设概率密度函数的形式已知，这是非参数化方法的基本优点。

Parzen窗方法的一个缺点是它需要大量的样本。在样本有限的情况下，很难预测它的收敛性效果如何。为了得到较精确的结果，实际需要的训练样本的个数是非常惊人的。这时要求的训练样本的个数比在知道分布的参数形式下进行估计所需要的训练样本的个数要多得多。而且，直到今天人们还没有找到能够有效的降低训练样本个数的方法。这也导致了Parzen窗方法对时间和存储空间的消耗特别大。更糟糕的是，它对训练样本个数的需求，相对特征空间的维数呈指数增长。这种现象被称为“维数灾难（curse of dimensionality）”，严重制约了这种方法的实际应用。Parzen窗方法的另外一个缺点是，它在估计边界区域的时候会出现边界效应。

Parzen窗方法的一个问题是，窗宽度的选择难以把握。下图是一个二维Parzen窗的两类分类器的判决边界。其中窗宽度h不相同。左边的图中的窗宽度h较小，右边的图中的窗宽度h较大。所以左侧的Parzen窗分类器的分类界面比右边复杂。这里给出的训练样本的特点是，上半部分适合用较小的窗宽度h，而下半部分适合用较大的窗宽度h。所以，这个例子说明没有一个理想的固定的h值能够适应全部区域的情况。这算是Parzen窗方法的一个不足之处。



PNN是Parzen窗方法的实现。PNN的好处之一是学习速度很快，因为学习规则非常简单（），并且每一个样本点只需要提供一遍。这个算法的空间复杂度也很容易计算：只要查看PNN图中的连接个数即可。而空间复杂度是 O((n+1)d)。 可见，PNN算法的缺点就是在硬件实现的时候，对存储空间的要求比较高，特别是当n 和 d 都比较大的时候。PNN算法的时间复杂度是 O(1)，因为公式中的内积都可以用并行的方式来完成计算。 所以，PNN算法适合于对计算速度要求高而存储器资源比较容易满足的场合。

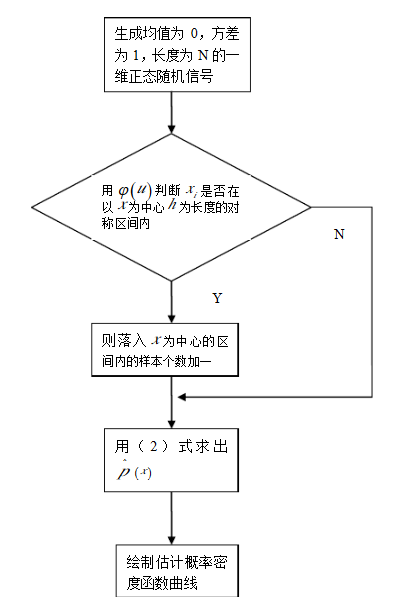
PNN算法的另外一个优点是新的训练样本很容易被加入以前训练好的分类器中，这一特性对于“在线”应用特别有意义。

**4、Parzen窗的仿真实验**

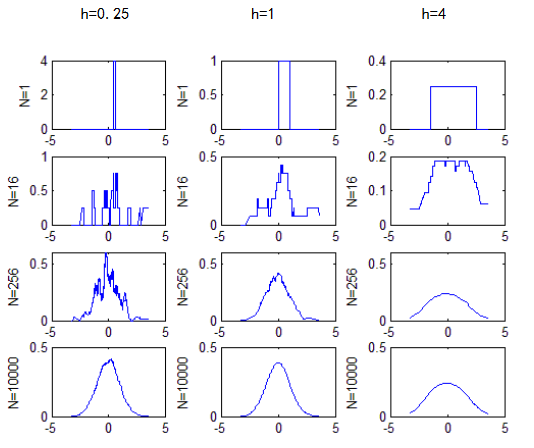
为了编写 Parzen窗，首先要确定一个窗函数。一般可以选择的窗函数有方窗、正态窗等。接下来将分别用方窗、正态窗实现Parzen窗法。

1）方窗函数：

方窗法的实验流程图：

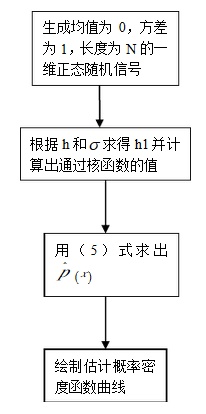


根据上述实验流程图可以得到matlab代码，此处省略。获得的实验结果如下图所示：

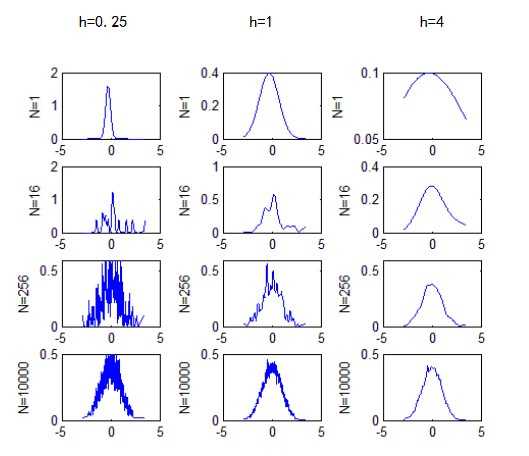


2）正态窗函数

使用该窗函数时，得到的实验流程图如下（matlab代码已省略）：



该方法得到的实验运行结果如下：

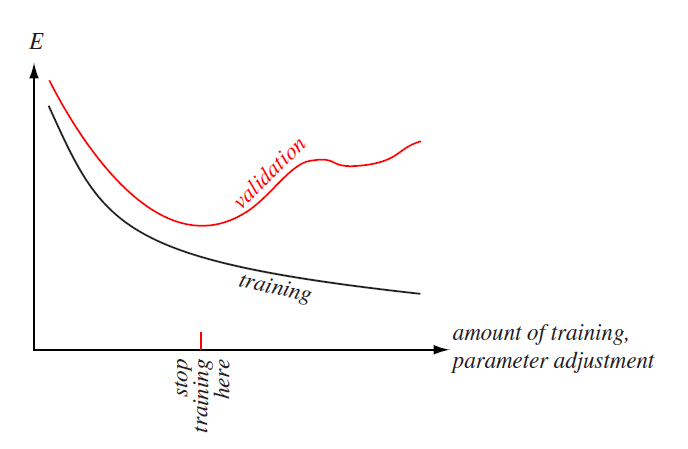


**5、对Parzen窗法的改进**

Parzen窗/PNN算法中的一个关键问题就是如何选取体积序列 V1, V2, … , Vn。例如，当我们选取 ，那么对于有限的n， 估计结果将对初始体积 V1非常敏感。如果V1非常小，那么大多数的体积内都是空的，估计得到的将包含很大的误差。但是如果V1非常大的话，那么平滑效应就会非常剧烈，以至于概率密度的空间变化被掩盖了。而且很有可能对于某一个区域适合的体积对于另一个区域就非常不适合。所以，可以考虑更加一般化的方法，例如使用交叉验证方法来辅助Parzen窗方法，提高算法的性能。

简单地说，“交叉验证方法”使用数据集中的一小部分来形成一个“验证集”，而窗的宽度就是通过使验证集上的误差率最小来调节得到的。我们可以通过使用“m-重交叉验证”（*m*-fold cross validation）来估计Parzen窗方法得到的分类器的推广性能。首先讲训练样本集D，样本点的总数为n。然后把训练样本随机地划分为m个集合，每个集合包含 n/m 个样本点，这m个训练样本子集互不相交。用这m个训练样本集去训练Parzen窗分类器，训练完成之后，再选择一个与训练样本集不同的样本集作为“验证集”（validation set），在验证的时候可以计算出这一次的推广误差（generalization error），把这个值作为Parzen分类器推广能力的度量。上述过程重复m次，将获得的推广误差取平均值，就可以估计Parzen分类器的推广能力，从而评价它的性能如何。

利用交叉验证技术，我们可以对Parzen窗技术中的高斯窗函数的宽度进行调整。通过不断调整窗宽度h的值，使得交叉验证方法得到的推广误差最小。从而达到优化Parzen窗方法的目的。下面的例子中，将数据集D分为2部分，第一部分占样本点总数的90%，用作训练集；第二部分占样本总数的10%，用作验证集以测试推广性能。对Parzen窗分类器以及大多数的问题而言，训练误差会随着训练的进行而单调下降。而在验证集上误差典型的情况是首先单调下降，然后上升。这是因为后面出现了对训练集“过拟合”（overfitting）的现象。所以，在调整窗宽度h的时候，在验证集误差达到第一个局部极小值时就停止。如图所示：



前面提到了Parzen窗方法存在着“维数灾难”的问题，这严重限制了Parzen窗方法的实际应用。产生“维数灾难”的最核心的问题是，高维函数事实上远比低维函数复杂，人们对其复杂度几乎无法进行有效的分析和掌握。现在对付“维数灾难”的惟一有效的方法就是尽可能多的在处理问题时嵌入关于模式数据本身的可靠的先验知识。

**6、参考文献**

[1] Richard O. Duda and Peter E. Hart and David G. Stork, *Pattern Classification 2nd edition*, 北京：机械工业出版社，2003.9

[2] 边肇祺、张学工等编著：《模式识别（第二版）》，北京：清华大学出版社，2001

[3] 维基百科.核密度估计. http://zh.wikipedia.org/zh-cn/核密度估计.

[4] Emanuel Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33(3):1065-1076, 1962.

[5] Donald F. Specht. Generation of polynomial discriminant functions for pattern recognition. *IEEE Transactions on Electronic Computers*, EC-16(3):308-319, 1967.