多种机器学习方法实现KDD Cup99数据分析

December 1, 2019

目录

1 实验背景

1.1 数据集简单介绍

KDD99数据集是从一个模拟的美国空军局域网上采集来的9个星期的网络连接数据, 分成具有标识的训练数据和未加标识的测试数据。测试数据和训练数据有着不同的概率分布, 测试数据包含了一些未出现在训练数据中的攻击类型, 这使得入侵检测更具有现实性。

该数据集用于1999年举行的KDD CUP竞赛中，成为著名的KDD99数据集。虽然年代有些久远，但KDD99数据集仍然是网络入侵检测领域的事实Benckmark，为基于计算智能的网络入侵检测研究奠定基础。

KDD99数据集中每条纪录用41个特征来描述，加上最后的标记，一共有42项。其中前41 项特征分为4大类，对应的主要4大类主要说明为如表1所示。

表 1: 41项特征的4大类说明

特征类别	**相关说明**
TCP连接基本特征	包含了一些连接的基本属性，如连续时间，协议类型，传送的字节数等。
TCP连接的内容特征	该部分数据可能反映入侵行为的内容特征，如登录失败的次数等。
基于时间的网络流量统计特征	记录了与之前一段时间内的连接记录之间存在的某些联系，如过去两秒内，与当前连接具有相同的目标主机的连接数。
基于主机的网络流量统计特征	按照目标主机进行分类，使用一个具有100个连接的时间窗，统计当前连接之前100个连接记录中与当前连接具有相同目标主机的统计信息。

1.2 实验数据

本次实验训练集采用10%的数据（kddcup.data_10_percent_corrected），测试集采用相同数据量级的数据（3-corrected），进行二分类和三分类两个任务。训练集包含494021条数据，测试集包含311029条数据。对于三分类，本次实验仅关注“normal”、“smurf”和其他，在二分类任务中将“smurf”和其他类别归为“attack”。在训练集中其对应的数据分布情况如表2所示：

表 2: 41项特征的4类说明

类型	normal	smurf	others	total
数量	97278	280790	115953	494021

实验数据中，每条纪录的“标签”、“协议类型”、“网络服务类型”和“连接状态”为字符型数据。41个特征中，有3个二值数据、7个离散型数据和31个数值型数据。

2 实验内容

2.1 数据预处理

上述提到，数据中包含了字符型数值，不方便处理，需要将其进行字符到数值的转化。原始数据中，不同特征的取值范围不同，即度量范围不统一，因此便需要归一化。数据的不同特征之间存在着一定的关联，并且算法对于高维数据的处理速度更为缓慢，所以数据的降维有一定的使用价值。

2.1.1 字符数值化

因为字符型数值为标程数值，所以本次实验对数据预处理时，仅对字符数据进行了简单的映射。例如，对于三分类的标签，*normal*被映射为0，*smurf*被映射为1，其他类别映射为2。

2.1.2 独热编码

大多数算法是基于向量空间中的度量来进行计算的，为了使非偏序关系的变量取值不具有偏序性，并且到圆点是等距的。使用独热编码，将离散特征的取值扩展到了欧式空间，离散特征的某个取值对应欧式空间的某个点。将离散型特征使用独热编码，会让特征之间的距离计算更加合理。

使用独热编码后，增加了数据的维度，所以本次实验仅对标签进行了one-hot 编码。又因为本次实验的要求为二分类和三分类，所以仅在神经网络部分使用了独热编码。

2.1.3 归一化处理

数据的度量单位不同时，会影响分类问题的最终结果。当有较大数值和较小数值的特征存在时，将会导致较小数值特征起不到贡献而无用。以神经网络为例，当所有样本的输入信号都为正值时，与第一隐含层神经元相连的权值只能同时增加或减小，从而导致学习速度很慢。另外在数据中常存在奇异样本数据，奇异样本数据存在所引起的网络训练时间增加，并可能引起网络无法收敛。为了避免出现这种情况及后面数据处理的方便，加快网络学习速度，可以对输入信号进行归一化，使得所有样本的输入信号其均值接近于0 或与其均方差相比很小。

min-max归一化的手段是一种线性的归一化方法，它的特点是处理简单，不会对数据分布产生影响。对应转换关系为：

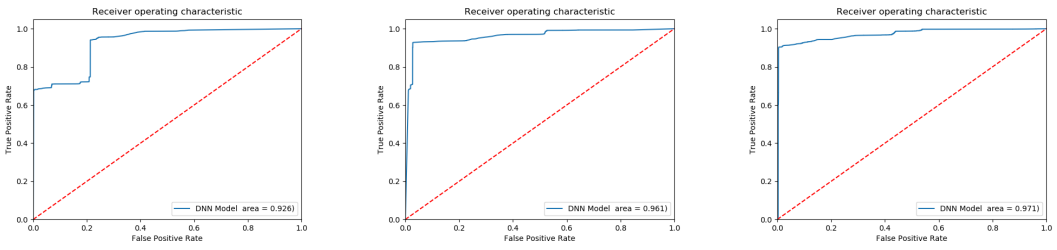
$$X = (x - X\min)/(X\max - X\min)$$

均值方差归一化可以将数据标准为正太分布，在实际应用中更有价值。其对应转换关系为：

$$X = (x - \mu)/\sigma$$

其中，μ为数据的均值，σ为数据的期望。

经过实验，均值方差归一化对数据拟合更好。在二分类任务中，两种方法数据归一化后，神经网络的实验如图1所示：



(a) 未归一化　(b) min-max归一化　(c) 均值方差归一化

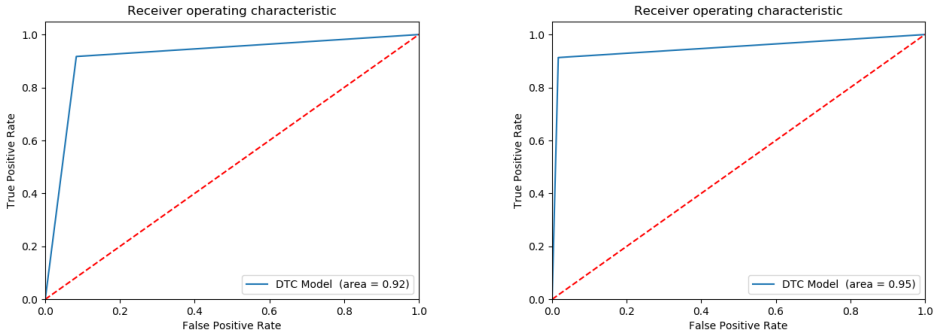
图 1: 数据归一化后DNN的预测效果

从图1中可以看到，经过数据归一化，预测的AUC值分别增长了0.035 和0.045，说明了数值归一化对分类任务的作用。因为本次实验中，有二值数据，其值已经为0或者1，对其归一化化后将以浮点数进行操作，提高了计算的复杂度，所以这里并未对二值数据进行归一化。

2.1.4 数据降维

数据的维度决定了数据的细节，但是处理较多的数据所需要的空间和时间较多，所以采用合适的算法进行降维将可以提升算法的效率。

常用的数据降维方法包括ICA、PCA和SVD等，这里仅考虑了ICA和PCA 两种方法。ICA是为了最大化独立性，使联合概率与各分量概率乘积最接近，得到的数据是原始数据集内的数据。pca为最大化方差，使得残余方差最小，或信息损失最小，得到的数据另一个映射空间的值。在第4章结果分析中，将对比分析不同维度的结果。在所选维度为15 时，使用两种降维方法的实验结果如图2所示。



(a) ICA对应ROC　　(b) PCA对应ROC

图 2: 两种降维算法的实验结果

从图2的对比中可以看到，同样的分类算法时，PCA的效果要优于ICA。PCA算法的基本流程如下：

Algorithm 1 PCA算法

Input:

训练集：m条n维数据;

Steps:

1: 将原始数据按列组成n行m列矩阵X;

2: 将X的每一行进行零均值化;

3: 求出协方差矩阵$C=\frac{1}{m}XX^T$;

4: 求协方差矩阵的特征值及对应的特征向量;

5: 将特征向量按对应特征值大小从上到下按行排列成矩阵，取前k行组成矩阵P;

6: **return** $Y = PX$即为降维到k维后的数据;

2.2 实验方法

针对分类任务，较为直观的有k近邻和朴素贝叶斯法。随着信息熵、感知机等知识的加入，分类算法又得到了发展，例如支持向量机、AdaBoost、随机森林以及神经网络等。

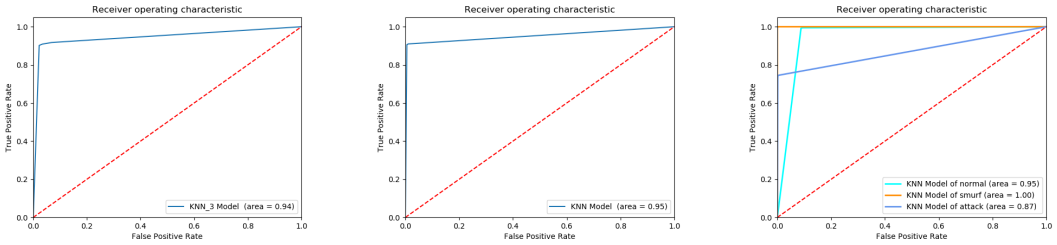
本次选用的评价指标主要包括准确率、召回率、查准率以及ROC曲线，对于细节的对比将在第3章进行总结。本次实验将对以上提到的算法简单介绍，对比分析实验结果。[1]

[1]Code will be released at https://github.com/IOEvan

2.2.1 k近邻法

k近邻法是基本且简单的分类与回归方法，其基本做法是：对给定的训练实例点和输入实例点，首先确定输入实例点的k个最近邻训练实例点，然后利用这k个训练实例点的类的多数来预测输入实例点的类。k近邻法三要素为：距离尺度、k值的选择和分类决策规则，当上述规则确定后，方法结果唯一。

本次实验通过对比分析，当K值取为5时，其实现效果最好。利用KNN进行分类的实验结果如图3 所示，分别展示了K取为3时，和K为5时的实验结果。



(a) K=3时二分类ROC曲线　(b) K=5时二分类的ROC曲线　(c) K=5时三分类的ROC曲线

图 3: k近邻算法实验结果

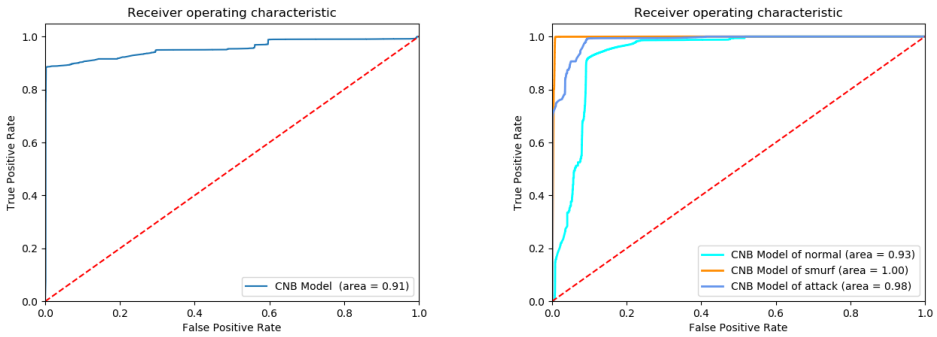
2.2.2 朴素贝叶斯法

朴素贝叶斯法是典型的生成学习方法，生成方法由训练数据学习联合概率分布$P(X,Y)$，然后求得后验概率分布$P(Y|X)$。朴素贝叶斯法的基本假设是条件独立性，

$$P\left(X^{(1)}=x^{(1)},\cdots,X^{(n)}=x^{(n)}|Y=c_k\right)=\prod_{j=1}^{n}P\left(X^{(j)}=x^{(j)}|Y=c_k\right)$$

这是一个较强的假设，由于该假设，所以称之为朴素。由于这一假设，模型包含的条件概率的数量大为减少。因而朴素贝叶斯法高效，且易于实现，缺点是分类的性能不一定很高。

利用朴素贝叶斯算法进行分类的实验结果如图4所示：



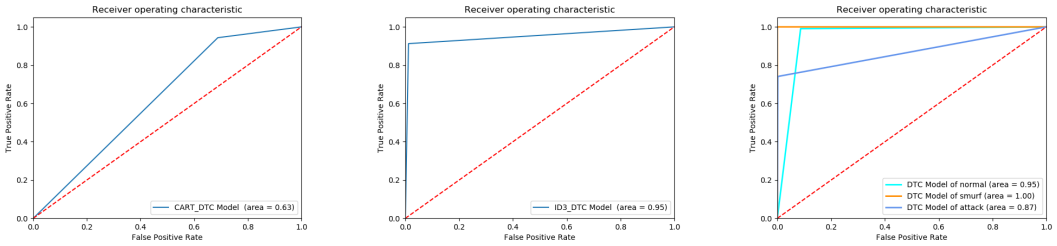
(a) 二分类的ROC曲线　(b) 三分类的ROC曲线

图 4: 朴素贝叶斯算法实验结果

2.2.3 决策树

分类决策树模型是表示基于特征对实例进行分类的树形结构，其可以转换成一个*if-then*规则的集合，也可以看作是定义在特征空间划分上的类的条件概率分布。决策树算法包括3部分：特征选择、树的生成和树的剪枝，常用的算法有ID3、C4.5 和CART。

本次实验仅对CART树和ID3进行了对比，两者实验结果如图5所示。从图5 中可以看到，对于本实验的分类任务，ID3算法的分类效果优于CART，所以本次实验决策树相关算法均采用ID3算法。



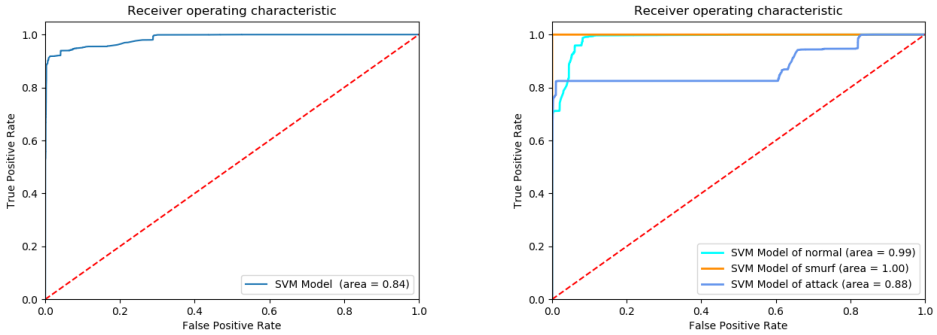
(a) CART算法对应的ROC曲线　(b) ID3算法对应的ROC曲线　(c) ID3算法三分类的ROC曲线

图 5: 不同决策树算法实验结果

2.2.4 支持向量机

支持向量机最简单的情况是线性可分支持向量机，或硬间隔支持向量机，构建它的条件是训练数据线性可分。现实中训练数据是线性可分的情形较少，训练数据往往是近似线性可分的，这时使用线性支持向量机，或软间隔向量机。

本次实验因数据较多，本次仅对训练数据集的前100000条数据进行训练。利用支持向量机算法进行分类的实验结果如图6 所示：



(a) 二分类的ROC曲线　(b) 三分类的ROC曲线

图 6: 支持向量机算法实验结果

2.2.5 集成学习

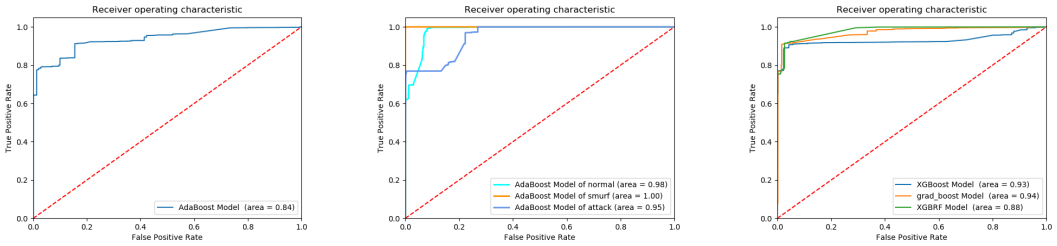
根据个体学习器的生成方式，目前的集成学习方法大致分为两大类：个体间存在依赖关系、必须串行生成的序列化方法，以及个体学习器间不存在强依赖关系、可同时生成的并行化方法。前者的代表是Boosting，后者的代表是Bagging和随机森林。

Boosting方法是将弱学习算法提升为强学习算法的统计学习方法。代表性的方法是AdaBoost算法，其主要原理是弱分类器的线性组合。

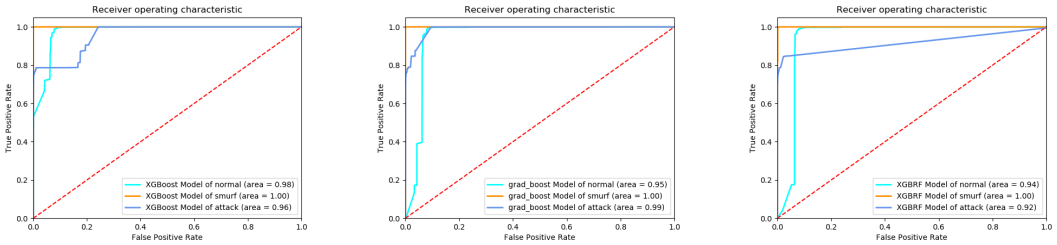
AdaBoost算法的特点是通过迭代每次学习一个基本分类器，每次迭代提高那些被前一轮分类器错误分类的权值，而降低那些被正确分类的数据的权值。最后AdaBoost 将基本分类器的线性组合作为强分类器，其中给分类误差率小的基本分类器以大的权值，给分类误差率大的基本分类器小的权值。AdaBoost的相关流程见算法2。

与AdaBoost类似，还有其他方法的增强型分类器，例如XGBoost、Gradient Boosting和XGBoost结合随机森林等方法。对应的实验效果如图7所示，可以看到XGBoost 和Gradient Boosting 的效果要明显优于另外两种方法。

Bagging是并行式集成学习方法最著名的代表，是一个很高效的集成学习算法。给定包含m个样本的数据集，先随机取出一个样本放入采样集中，再把该样本放回初始数据集，使得下次采样时该样本仍有可



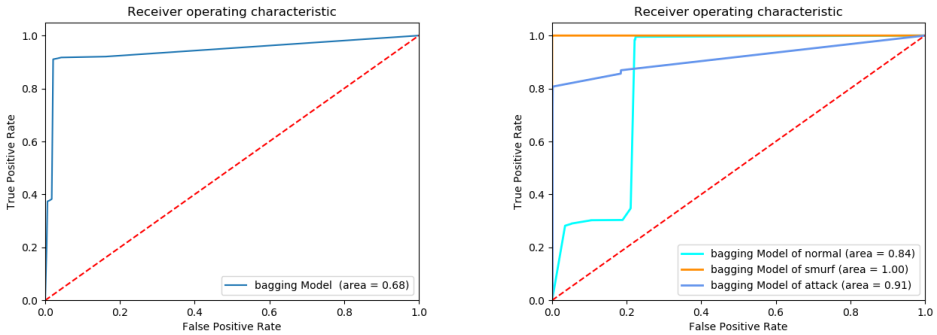
(a) AdaBoost二分类ROC曲线　(b) AdaBoost三分类ROC曲线　(c) 其他三种Boost二分类ROC曲线



(d) XGBoost三分类的ROC曲线　(e) GradientBoost三分类ROC曲线　(f) XGBoost_RF三分类ROC曲线

图 7: Boost类算法实验结果

能被选中。经过m次随机采样操作，便可以得到含m个样本的采样。将上述操作重复T次，可以采样出T个含m个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合，这就是Bagging算法。其实验结果如图8所示：



(a) 二分类的ROC曲线　(b) 三分类的ROC曲线

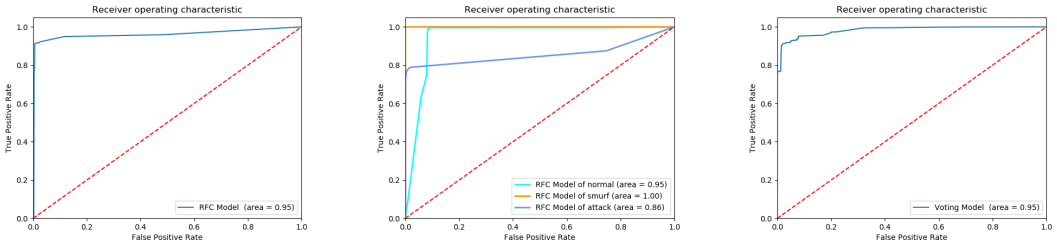
图 8: Bagging实验结果

随机森林是Bagging的一个拓展变体，进一步在决策树的训练过程中引入了随机属性选择。具体来说，在随机森林中，对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含k个属性的子集，然后再从这个子集中选择一个最优属性用于划分。 可以看到随机森林的实现效果要优于Bagging，并且训练速度要更快。

集成学习除了利用上述强分类器，还可以将上述分类器进行投票或者加权平均，得到一个更强的分类器如9所示。

2.2.6　神经网络

在进行数据降维时，其本至就是矩阵变换。矩阵变换也是人工神经网络的基本操作，并且通过设置非线性激活函数，增加了模型的拟合能力。因为人工神经网络的全连接层的后一层决定了该层特征的个数，所以



(a) 二分类的ROC曲线　　(b) 三分类的ROC曲线　　(c) Voting的ROC曲线

图 9: 随机森林实验结果

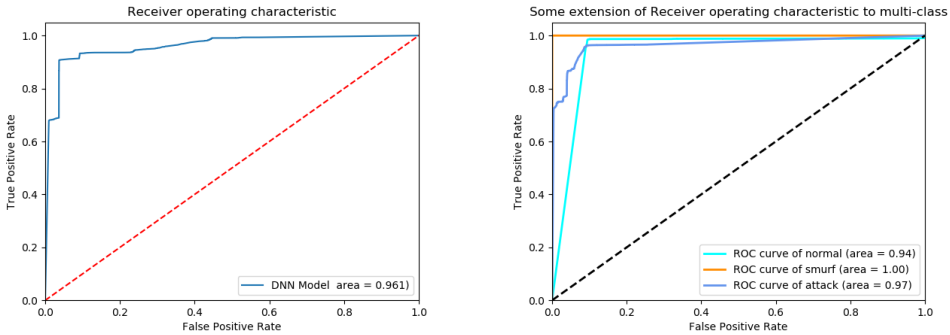
本次实验在使用人工神经网络时，并未使用降维，而是将第二层的神经元个数设置为16，便可以完成数据的拟合。

本次实验三分类采用的神经网络具体结构如表3，二分类任务仅将输出改为2即可。

表 3: 人工神经网络各层设置

Layers	**Input**	**fc1**	**fc2**	**fc3**	**fc4**	**fc5**	**output**
nodes	41	16	128	64	32	16	3

本次实验中，batch_size设置为100，训练集与验证集的比例为。除最后一层采用Softmax外，所有的激活函数均采用ReLU。仅训练2个epoch后，便可以达到10 的效果，可以看到神经网络的拟合性较强。



(a) 二分类的ROC曲线　　(b) 三分类的ROC曲线

图 10: 神经网络实验结果

3 实验分析

3.1 数据降维

本次实验使用PCA进行数据的降维，但是维度的大小直接决定了算法训练的复杂度和精确度。因为数据量相对较小，对所使用算法的复杂度要求不高。这里仅从精确率方面考虑，来实现数据降维。因为本次实验中，存在二值数据和标程型数据，所以在进行降维时，将标程型数据和数值型数据分别处理。在使用决策树算法时，不同降维方式对比如表4。

表 4: 不同降维方法时的精准度对比

维度	41	25	25^1	20	20^1	18	15	15^1	13	10	8
准确率	92.2%	92.6%	92.3%	92.5%	**93.4%**	92.5%	92.3%	93%	88.9%	92.4%	92.3%
查准率	**86.3%**	73%	72.5%	73.1%	72.3%	72.7%	73.8%	74.6%	65.9%	72.8%	72.6%
查全率	71.3%	**98.7%**	98.4%	97.5%	96.9%	98.5%	93.9%	96.9%	89.1%	97.5%	97%
AUC	0.84	**0.95**	0.94	0.94	**0.95**	**0.95**	0.93	0.94	0.89	0.94	0.94

表4中，25^1表示数值型数据15维，其他数据维度不变；20^1表示数值型数据13维，离散数据5维，二值数据2维；15^1 表示数值型数据10维，离散数据3维，二值数据2维；可以看到，当数据降维到20^1时，实现的精确相对较好。所以本文所用的维度将是20^1维。

3.2 结果对比

经过上述实验，现在对二分类任务进行总结如表5。

表 5: 二分类任务实验结果对比[2]

方法	贝叶斯	SVM	KNN	DCT	RF	Ada	Bagging	XGB	GDB	XGB RF	DNN
准确率	90.1%	74.8%	92.5%	91%	92.4%	92.6%	49.7%	92%	92.5%	81.5%	**99.82%**
查准率	67.9%	43.6%	72.5%	71.2%	72.3%	74.1%	27.7%	71.6%	72.8%	51.3%	**87.6%**
查全率	93.6%	**99.9%**	99.4%	90.4%	99%	95%	98.5%	97.9%	97.9%	98.3%	99.8%
AUC	0.91	0.84	0.95	0.91	0.95	0.94	0.68	0.94	0.95	0.88	**0.96**

[2]DCT:决策树；RF:随机森林；GDB:Gradient Boosting；XGB RF: XGBoost + 随机森林

从表5中可以看到，神经网络的模拟效果最好。从10中可以看到神经网络针对三分类任务也能达到较好的实验结果，三分类精度可达99.82%。

从训练速度来看，朴素贝叶斯方法训练速度最快，仅需1s左右便可以得到较好的结果。SVM对于大数据量的速度较慢，使用全部数据需要2个小时达到收敛。其次速度较慢的为KNN，这也取决于K 值和维数的选取，当K取值为5时，需要3min左右完成模型的训练和预测。其余算法均在半分钟左右完成训练和预测。

3.3 实验环境

本实验基于的环境如下：

- 硬件设备：2080ti；
- 软件环境：python=3.6.8、keras=2.0.8、numpy=1.16.3、pandas=0.24.2、scikit-learn=0.20.3、tensorflow-gpu=1.9.0；

4 实验总结

本次实验通过多种机器学习方法，对入侵检测数据KDD99进行了充分的分析。针对二分类和三分类任务，总结如下：

- 利用多种机器学习方法，对实验进行详细的对比分析，发现简单的人工神经网络即可获得较好的分类效果；
- 通过对数据的预处理，简化了计算复杂度并提高了分类的效果；
- 通过降维方法的对比，选择了较为合适的维度和降维方法；
- SVM核函数以及数据处理不完全合理，所以分类效果相对表现不佳。

附录

Algorithm 2 AdaBoost算法

Input:

训练集$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$

基学习算法ζ;

训练轮数T;

Steps:

$D_1(x)=1/m.$;

for $i=1$ to n **do**

  $h(t)=\zeta(D,D_t)$

  $\varepsilon_t=P_{x\tilde{}D_t}(h_t(x)\neq f(x))$;

  if $\varepsilon_t>0.5$ **then**

    break

  end if

  $\alpha_t=\frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

  $D_{t+1}(x)=\frac{D_t(x)}{Z_t}\times\begin{cases}\exp(-\alpha_t), & h_t(x)=f(x)\\ \exp(\alpha_t), & h_t(x)\neq f(x)\end{cases}=\frac{D_t(x)\exp(-\alpha_t f(x)h_t(x))}{Z_t}$

end for

return $H(x)=\operatorname{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right)$;