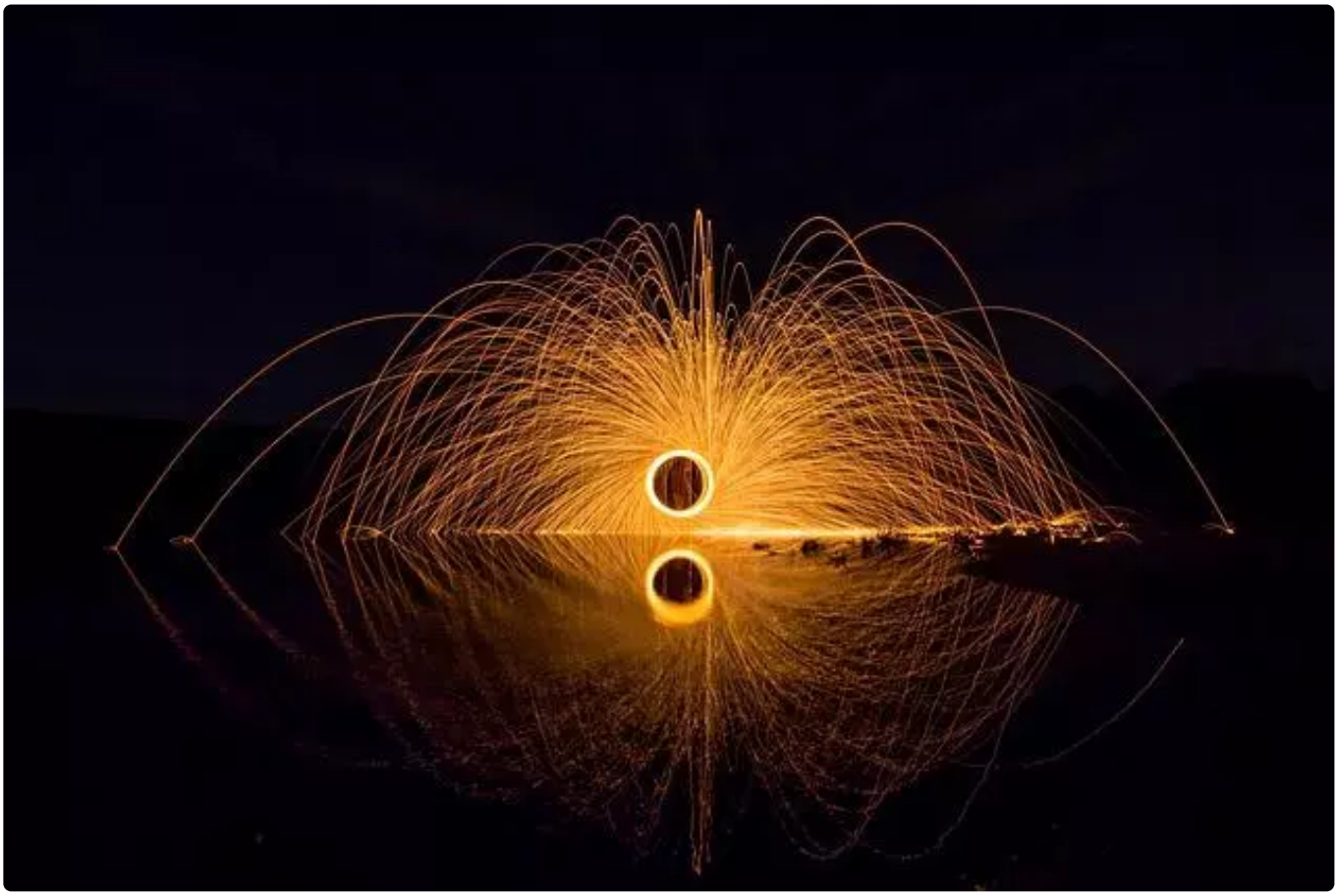
降维：PCA真的能改善分类结果吗?

原创 AI前线小组 译 AI前线 2018-07-31



作者 | Meigarom Diego Fernandes
译者 | 姚佳灵
编辑 | Natalie

AI 前线导读： 我已经接触过很多降维技术。这个话题绝对是最有趣的话题之一，很高兴能有这样的想法：算法能够通过选择仍能代表整个数据集的最重要的特征，从而减少特征的数量。作者们指出的优点之一是，这些算法能够改善分类任务的结果。

我在本文中将利用主成分分析（Principal Component Analysis，简称 PCA）来验证这个结论，以尝试改善神经网络在数据集上的分类性能。PCA 真的能改善分类结果吗？我们一起来看看。

更多优质内容请关注微信公众号“AI 前线”（ID：ai-front)

降维算法

在直接讲代码之前，我们来讨论一下降维算法。降维有两个主要算法：线性判别分析（Linear Discriminant Analysis，简称 LDA）和主成分分析。这两者之间最根本的区别在于，LDA 利用类的信息去找新特征，以最大化类的可分离性，而 PCA 利用每个特征的方差来做同样的事情。在这种情况下，LDA 可以被看做是监督算法，而 PCA 是非监督算法。

谈谈 PCA

PCA 背后的想法就是去找一组概括数据的低维轴。那么，我们为什么需要概括数据呢?

我们来看个例子：我们有个数据集，里面包含一组汽车属性。这些属性通过尺寸、颜色、形状、紧凑度、半径、座位数量、门的数量、后备箱尺寸等等来描述每一辆车。但是，很多这类特征要测量相关属性，因此可能是多余的。我们应该把这些多余的特征除去，用更少的属性来描述每一辆汽车。这就是 PCA 的目标所在。例如，把轮子的数量看作是轿车和公共汽车的特征。而来自这两类的每个例子几乎都有 4 个轮子，因此，我们可以说这个特征的方差很小（一些罕见的公共汽车有 6 个轮子），也即这个特征让汽车和轿车看上去是一样的，但是，事实上两者是很不相同的。现在，把高度看作特征，轿车和公共汽车的高度不同，从最矮的轿车到最高的公共汽车之间的方差有一个很大的范围。很显然，车辆的高度是区分车辆的好属性。回想起 PCA 不考虑分类信息，它只看每个特征的方差，因为假设呈现高方差的特征更有可能具有良好的类的分割性是有道理的。

通常，人们错误地认为 PCA 从数据集选择某些特征而丢弃其他特征。事实上，算法在旧属性的组合基础上构造了一组新属性。从数学上讲，PCA 执行了一个线性转换，从原始特征集合转到由主成分组成的新空间。这些新特征对于我们来说没有任何实际意义，只是代数问题，因此，不要认为 PCA 会找到那些你从未想到会存在的新特征。很多人仍然认为机器学习算法非常神奇，他们把数以千计的输入直接提供给算法，就希望能找到所有的见解和解决他们问题的方案。不要被骗了！对于业务的见解，那是数据科学家的工作，这要通过把机器学习算法当做一套工具而不是魔术棒、并有效开展的探索性数据分析来完成。请牢牢记住这一点。

主成分空间看起来是怎样的?

在这个新特征空间中，我们在寻找不同类之间存在很大差异的某些属性。正如我在前一个例子中展示的，某些呈现低方差的属性是没有用的，它会让样本看上去几乎都一样。另一方面，PCA 寻找那些尽可能展示类中最大方差的属性，以构建主成分空间。该算法使用方差矩阵、协方差矩阵、特征向量和特征值对的概念，以执行 PCA，得到一组特征向量及其相应的特征值。PCA 的具体表现将是下一篇文章的内容。

那么，我们该用特征值和特征向量来做什么呢？很简单，特征向量代表一组主成分空间的新坐标轴，特征值携带每个特征向量的方差数量的信息。因此，为了降低数据集的维度，我们准备选择那些具有较大方差的特征向量，丢弃那些具有较小方差的特征向量。随着我们对下面的例子的讲解，就会越来越清楚它的工作原理。

我们终于要来看看代码了

现在，我们来到了本文让人感到有趣又快乐的部分。我们来看看，PCA 是否真的改善了分类任务的结果。

为了验证它，我的策略是在数据集上应用神经网络，看看它最初的结果。然后，我准备在分类前运行 PCA，并在新的数据集上应用同样的神经网络，最后将两个结果比较一下。

这个数据集最初来自 UCI 叫做“Statlog （车辆轮廓）数据集”的机器学习库。该数据集存储了 4 种车辆轮廓的度量值用于分类。它由 946 个样本和 18 个度量值（属性）的所有数值组成，你可以通过以下链接获取更多细节：https://archive.ics.uci.edu/ml/datasets/Statlog+(Vehicle+Silhouettes)。神经网络将是多层感知器（Multilayer Perceptron），具有 4 个隐藏节点和 1 个输出节点，所有都具有 sigmoid 函数作为激活函数，PCA 函数将来自于一个 R 包。

准备数据集

首先，我为二分法准备数据集。

我打算只从两个类中选取样本，以构成二分法。样本将来自“bus”和“saab”两类。“saab”这类将由类别 0 代替，“bus”类将由类别 1 代替。下一步是把数据集分成训练集和测试集，分别占总分类样本的 60% 和 40%。

在数据集准备好后，我们一次性使用所有特征建模神经网络，然后应用测试数据集。

```
# Load library
library( dplyr )

# Load dataset
data = read.csv( "../dataset/vehicle.csv", stringsAsFactor = FALSE )

# Transform dataset
dataset = data %>%
                filter( class == "bus" | class == "saab" ) %>%
                transform( class = ifelse( class == "saab", 0, 1 ) )
dataset = as.data.frame( sapply( dataset, as.numeric ) )

# Spliting training and testing dataset
index = sample( 1:nrow( dataset ), nrow( dataset ) * 0.6, replace = FALSE )

trainset = dataset[ index, ]
test = dataset[ -index, ]
testset = test %>% select( -class )

# Building a neural network (NN)
library( neuralnet )
n = names( trainset )
f = as.formula( paste( "class ~", paste( n[!n %in% "class"], collapse = "+" ) ) )
nn = neuralnet( f, trainset, hidden = 4, linear.output = FALSE, threshold = 0.01 )

plot( nn, rep = "best" )
```

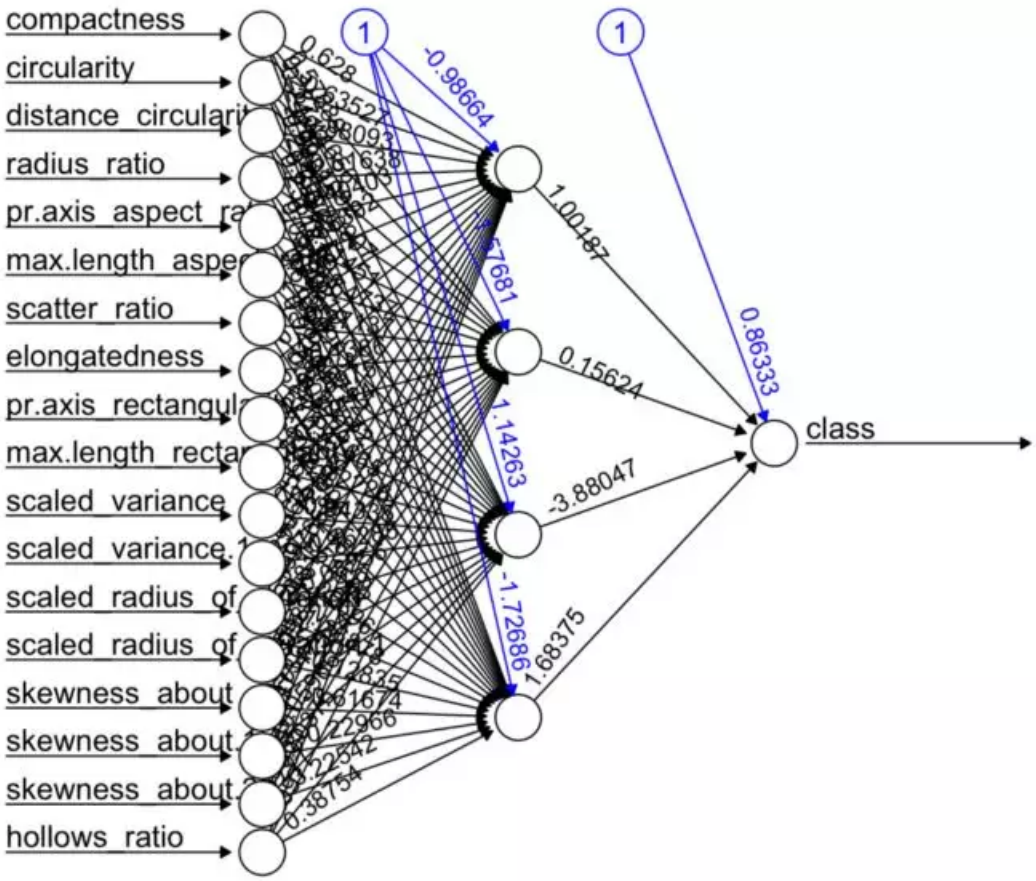


图 1. 神经网络 多层感知器

```
# Testing the result output
nn.results = compute( nn, testset )

results = data.frame( actual = test$class, prediction = round( nn.results$net.result ) )

# Confusion matrix
library( caret )
t = table( results )
print( confusionMatrix( t ) )
## Confusion Matrix and Statistics
##
##          prediction
## actual   0   1
##         0 79   0
##         1 79 16
##
##                          Accuracy : 0.545977
##                            95% CI : (0.4688867, 0.6214742)
##       No Information Rate : 0.908046
##       P-Value [Acc > NIR] : 1
##
##                             Kappa : 0.1553398
##   Mcnemar's Test P-Value : < 0.0000000000000002
##
```

```
##                  Sensitivity : 0.5000000
##                  Specificity : 1.0000000
##               Pos Pred Value : 1.0000000
##               Neg Pred Value : 0.1684211
##                    Prevalence : 0.9080460
##               Detection Rate : 0.4540230
##      Detection Prevalence : 0.4540230
##         Balanced Accuracy : 0.7500000
##
##           'Positive' Class : 0
##
```

没有用 PCA 的结果

看起来我们得到了一些结果。首先，来看看这个混淆矩阵。基本上，混淆矩阵表示有多少样本被分类。主对角线表示正确分类的样本，次对角线表示错误的分类。在第 1 个结果中，分类器显示自己也是混乱的，因为它正确分类了几乎所有来自“saab”类的样本，但是，它也把大多数“bus”类样本分为“saab”类。对应这个结果，我们能看到精确度大约为 50%，对于分类任务来说，这真是个糟糕的结果。分类器基本上有 50% 的概率把一个新样本分为“轿车”类，分为“公共汽车”类的概率也是 50%。这个神经网络就像在随机抛硬币那样为每个新样本选择其应该归属的类。

我们来看看 PCA 是否能帮助我们

现在，我们对数据集执行主成分分析，得到特征值和特征向量。实际上，你将看到来自 R 包的 PCA 函数提供一组已经按降序排序的特征值，这意味着第 1 个成分具有最大方差，第 2 个成分是具有第 2 大方差的特征向量，以此类推。下面的代码展示了如何选择特征向量查看特征值。

```
# PCA
pca_trainset = trainset %>% select( -class )
pca_testset = testset
pca = prcomp( pca_trainset, scale = T )

# variance
pr_var = ( pca$sdev )^2

# % of variance
prop_varex = pr_var / sum( pr_var )
```

```
# Plot
plot( prop_varex, xlab = "Principal Component",
                          ylab = "Proportion of Variance Explained", type = "b" )
```

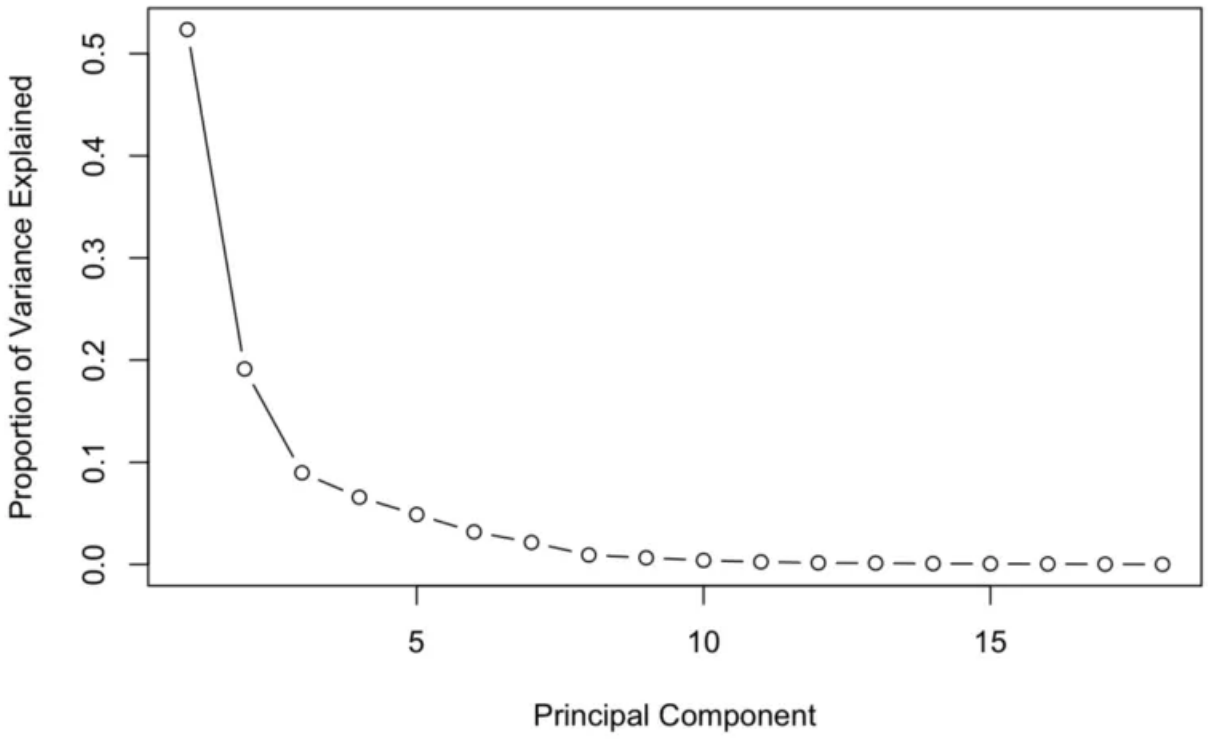


图 2. 每个主成分的方差百分比

```
# Scree Plot
plot( cumsum( prop_varex ), xlab = "Principal Component",
ylab = "Cumulative Proportion of Variance Explained", type = "b" )
```

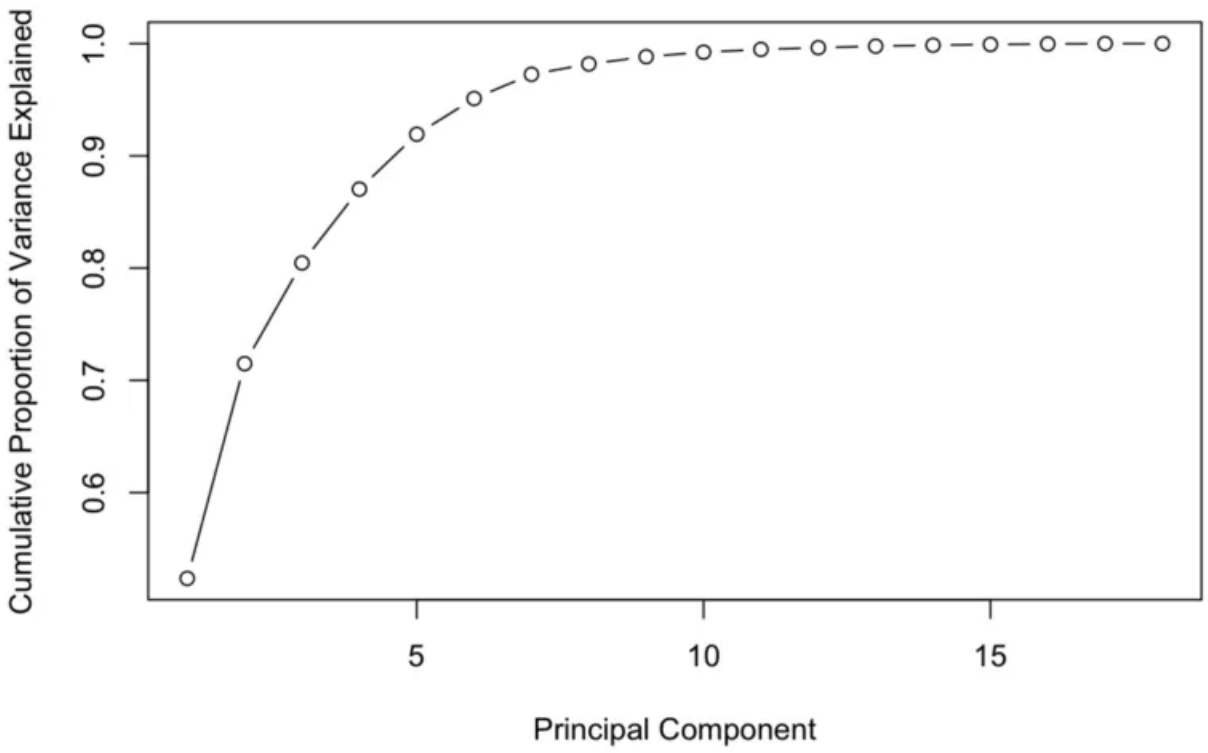


图 3. 方差的累积和

来自 stats 默认包的本地 R 函数“prcomp”执行 PCA，它返回所有所需的特征值和特征向量。第 1 张图显示每个特征的方差百分比。你能看到，第 1 个成分具有最大的方差，大约为 50%，而第 8 个成分大约只有占 0% 的方差。因此，我们应该选择最前面的 8 个成分。第 2 个图显示了方差的另一个视角，尽管累积和超过了所有的方差，你能看到前 8 个特征值对应大约所有方差的 98%。这的确是一个相当不错的数字，它意味着只有 2% 的信息丢失。最大的好处是我们从一个具有 18 个特征的空间转到了一个只有 8 个特征的空间，而且只丢失了 2% 的信息。这绝对是降维的力量!

现在，我们知道了将要构成新空间的特征的数量，我们来创建一个新数据集，然后再次建模神经网络，并查看我们是否获得新的更好的输出结果。

```
# Creating a new dataset
train = data.frame( class = trainset$class, pca$x )
t = as.data.frame( predict( pca, newdata = pca_testset ) )

new_trainset = train[, 1:9]
new_testset =   t[, 1:8]

# Build the neural network (NN)
library( neuralnet )
n = names( new_trainset )
```

```
f = as.formula( paste( "class ~", paste( n[!n %in% "class" ], collapse = "+" ) ) )
nn = neuralnet( f, new_trainset, hidden = 4, linear.output = FALSE, threshold=0.01 )

# Plot the NN
plot( nn, rep = "best" )
```

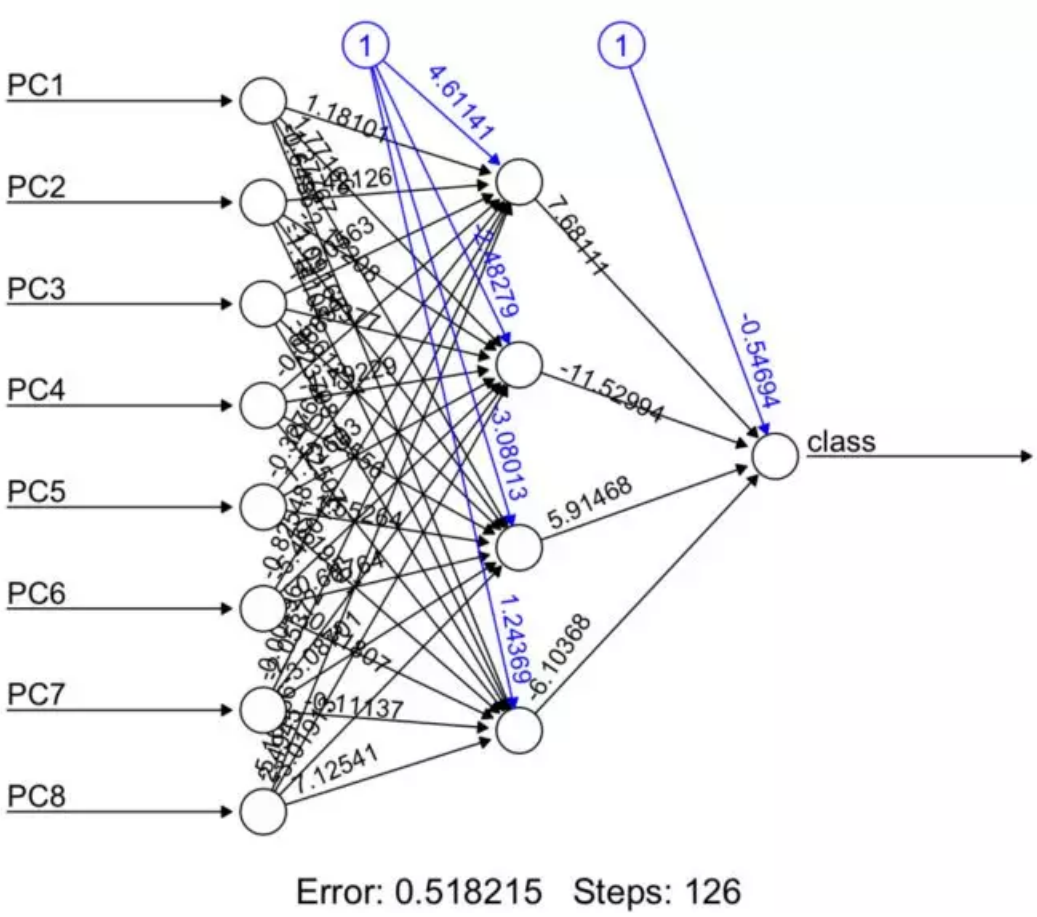


图 4. 包含新数据集的神经网络

```
# Test the resulting output
nn.results = compute( nn, new_testset )

# Results
results = data.frame( actual = test$class,
                      prediction = round( nn.results$net.result ) )

# Confusion Matrix
library( caret )
t = table( results )
print( confusionMatrix( t ) )
## Confusion Matrix and Statistics
##
##          prediction
## actual   0   1
##        0 76   3
```

```
##          1   1 94
##
##                    Accuracy : 0.9770115
##                      95% CI : (0.9421888, 0.9937017)
##       No Information Rate : 0.5574713
##       P-Value [Acc > NIR] : < 0.00000000000000022
##
##                     Kappa : 0.9535318
##   Mcnemar's Test P-Value : 0.6170751
##
##                 Sensitivity : 0.9870130
##                 Specificity : 0.9690722
##              Pos Pred Value : 0.9620253
##              Neg Pred Value : 0.9894737
##                   Prevalence : 0.4425287
##              Detection Rate : 0.4367816
##      Detection Prevalence : 0.4540230
##          Balanced Accuracy : 0.9780426
##
##           'Positive' Class : 0
##
```

我猜我们现在得到了更好的结果。我们来仔细地检查一下。

混淆矩阵这次确实显示了很好的结果，神经网络在两个类中分错类的情况大大减少，这可以通过主对角线的值看出来，而且精度值约为 95%。这意味着分类器对于一个新的未曾见过的样本，正确对其分类的概率为 95%。就分类问题来说，这是个不错的结果。

结论

降维在机器学习中起了非常重要的作用，特别是在处理数以千计的特征时。主成分分析是顶尖降维算法之一，它容易理解并易于应用在实际项目中。这个技术除了让特征处理工作变得更容易外，还有助于改善分类器的输出结果，正如我们在本文中所见的那样。

最后，对于本文最初提出的问题，我们的的答案是肯定的，主成分分析确实有助于改善分类器的输出结果。

接下来是什么?

正如我之前提到的，还有另外的降维技术，比如线性判别分析、因子分析、等距映射（Isomap）及其变种。这系列文章的目的是探索每种技术的优缺点，并分别和结合起来查看它们的输出结果。LDA 和 PCA 结合起来会改善分类器的输出吗？我们将在下一篇文章中探索。

文中提到的代码可以从我的 GitHub 存储库中获取完整代码，还可以获取数据集。链接如下：

https://github.com/Meigarom/machine_learning

作者简介

Meigarom Diego Fernandes 是一位热爱数据的工程师。他热衷于机器学习算法和数据科学工作。

阅读英文原文：

https://www.kdnuggets.com/2018/07/dimensionality-reduction-pca-improve-classification-results.html

如果你喜欢这篇文章，或希望看到更多类似优质报道，记得给我留言和点赞哦！