课程论文精编

"Data Mining"

《数据挖掘》[1]

(2021 年 12 月)

Jingbo Xia

2021-12-25

[1]A course for graduates in HZAU

Contents

1

序

面向信息学院研究生的《数据挖掘》这门课程自 2017 年开设以来，每年固定在秋季开设，选课人数少时十来人，多时到了 30 多位，而由于研究生扩招的缘故，2020 年秋季的课堂则有 68 位同学。到了 2021 年，数量仍然保持在 68 位。除了开课后有一名研究生退选外，剩余 67 个同学尽管了解到课程的理论难度仍然坚持了下来。

这门课的假想听众是信息学院计算机类和生物信息类的研究生，实际选课的同学有 1/3 来自生命科学学院、植物科学学院等对数据挖掘有学习需求的研究生同学（包括博士研究生）。课程的主要目的是为未来进行数据挖掘研究工作的同学们打好理论基础，采用的是“从易到难、一缓/一宽、理论为主、兼顾差异”的讲授策略。关于“一缓/一宽”的讲授策略，我在课程体系中还会要谈到。

1.1 课程体系

32 个课时里，主要对数据挖掘的诸多算法进行梳理和讲解，偏重理论推导，对代码实现则讲授不多。在近几年教学中，逐渐稳定成如下一个体系：

一、前言

二、数据挖掘概貌

三、入手数据挖掘算法，朴素贝叶斯

四、最优化的语言体系——《最优化理论与算法之 1，引言》

五、典型的经典最优化方法——线性回归，LOSS 函数与范数

六、最优化的语言体系——《最优化理论与算法之 7，最优性条件》（2021 年略讲）

七、典型的经典最优化算法——SVM，超平面上的咏唱（2021 年略讲）

八、神经网络，从传统到摩登

九、高维数据下的优化算法观点，从 SVD 到 PCA 到 LSA

十、特征约简，各类方法交汇处（2021 年略讲）

十一、从矩阵分解到张量分解，知识推理的一种方法

十二、隐变量和贝叶斯推断，EM 算法

十三、若干与贝叶斯推断有关的概率统计基础（2021 年增加）

十四、隐变量和贝叶斯推断，变分推断 VI

十五、隐变量和贝叶斯推断，VI 下的 LDA 主题模型（2021 年增加了一个 IGESS 模型）

十六、漫谈流行，神经网络、强化学习、VAE 和 GAN（2021 年主要是围绕 VAE）

实际上，数据挖掘课程的内容遴选在当前流行的教材中并不统一；或重方法罗列，或重代码实现，或重经典，或重流行。而当前这个体系主要考虑能在 32 个课时内得以充分讲解的一个脉络，既能谈一些最优化的基本理论，讲一些经典方法，也能在贝叶斯推断这个部分讲些东西，故能体现一些最优化、机器学习和统计学习的脉络。

循近几年的讲授习惯，采用的是“一缓/一宽”的两套教学方案。如果头一年采用的是黑板板书缓缓推导的形式，则在次年使用 Slides 为主进行宽口径的内容讲授。黑板推导虽缓，但引导启发性强；Slides 讲授覆盖面广，多场景展示也易于实现，但对理论细节的启发不如传统板书——这样以来，“一缓/一宽”的讲法能在有效的学时内各有侧重、互能补充，甚就吸引少数学生在选课翌年再来旁听。

1.2 今年的讲授内容安排

在 2021 年秋这个奇数年份，课程采取的是宽口径的讲授方法，从基础的朴素贝叶斯、线性回归等算法，过渡到矩阵分解、张量分解、PCA 等稍稍进阶的方法，最后从十二章开始主要讲述贝叶斯框架下若干当前流行的算法。

相比往年，今天的讲授内容做了两处调整。一是 SVM 的缩减。二是贝叶斯推断的增加。

最优化和 SVM 这个较为传统的黑板演绎部分，从准备最优化理论基础（第四章和第六章）到 SVM 的过程的讲述（第七章），往往要用掉 12 个以上（甚至更多）课时。由于今年讲述内容以贝叶斯推断为重心，所以对 SVM 的内容仅仅是概貌性的介绍，而从十二章到十六章的贝叶斯推断部分的讲授时间则较为充分。在以往面向讨论班同学的算法研讨中发现低年级研究生普遍对统计学习的记号和基本概念、结论较为陌生，为便于内容的展开，2021 年增加了第十三章《若干与贝叶斯推断有关的概率统计基础》。这样，在有了基本概念的了解后，学生们则能更多理解当前各类流行算法的模型描述和求解过程。

在十分紧张的讲授计划下，今年我们在结课的时候利用 1 个学时邀请了王欣然、谢款、葛星、和王嘉琪四位同学做课程大论文的 Presentation。他们做了仔细的准备，并做了有关 SVM，CNN，Transformer，贝叶斯网络的四个 10 分钟精彩讲解。

1.3 如何从课程中获得所需

给选课同学的第一个建议是：请通过课程网站（`https://hzaubionlp.com/data-mining/`）了解课程的讲授内容和讲授计划，这不仅能帮助各位制定预习计划，甚至能帮助各位作出选课决策。

第二个建议是有关学习资料的有效利用：这门课程的全程 Course Note 是不定期更新的，并可在课程页面获取；同时，2021 年的《课程论文精编》——也就是各位拿在手头的这份资料——是从 22 篇课程论文中甄选出来的，其内容与课程讲述有较好呼应，有助于补充课程理解。

另外一个是有关笔记的分享：今年，鲁基圣同学提供了课程的电子笔记，包含课外学习心得，这个电子文档可以在课程网站获取；同时，各位可以在往年的课程页面上找到偶数年（黑板讲述年）的电子笔记，通过笔记，可以回顾偶数年份讲解时相当部分的黑板演绎。

1.4 《课程论文精编》的内容甄选

这个学期的课程论文三人成组，论文准备包括短文、初稿和终稿三个环节。短文的编写有助于团队合作形成，长文初稿有助于内容形成，长文终稿基本能达到内容完善。

在所交 22 份论文中，朴素贝叶斯 2 篇，回归算法 2 篇，SVM2 篇，矩阵分解 2 篇，PCA2 篇，神经网络 9 篇（CNN 就有 5 篇），贝叶斯推断有关 2 篇，其他算法 1 篇。我从中挑选了十篇与课程体系结合最为紧密的论文，编纂到当前这个册子里。这十篇课程论文，或算法描述准确，能呼应课程讲授；或实现细节详细，有对应的 GitHub 项目帮助研究重现 (Reproducibility)，摘录进来形成一个整体，希望能有助于后面选课的同学课外自学。

由于这本小册子的篇幅控制，我从讲述相同算法的课程论文中只能挑选一篇，内容相似的论文常各有长处，我挑选的原则是更利于后选课的同学们进行课堂外延伸自学。另有若干精彩论文，或有算法的原创研讨，或有算法的发展回顾，或有图文并茂算法实践，也十分遗憾不能放到这个册子中来了。

感谢助教欧阳思卓同学  ，她在评阅课程论文短文、修订课程论文长文初稿，和整理论文修订回馈意见时，均付出了大量的时间和精力。这样才有了手头这份材料。

同样感谢 2021 年秋选课的同学们。欢迎诸位出现在 2022 年秋的课堂，来旁听全程的黑板推演。

夏静波

2021 年 12 月 25 日

武汉狮子山

2

基础算法

在这个部分，主要收录的是介绍朴素贝叶斯和线性回归的三篇论文。

– Jingbo Xia

2.1 《朴素贝叶斯算法及其最近理论发展》——胡鑫，王志慧，赵泽旗

摘要：朴素贝叶斯分类器简单且高效，然而其基于属性间相互独立的假设限制了算法的应用，目前学者们从不同的角度提出了有针对性的改进算法，本文回溯了朴素贝叶斯分类算法的基本原理和近年来基于结构扩展、属性加权、属性选择、局部学习以及实例加权等多个方面的算法改进工作，各类改进工作均能在一定程度上满足朴素贝叶斯分类算法的属性间独立性假设，多种方法相结合的算法改进工作也取得相应进展，与单一方向改进相比效果更为明显，总的来说，各个方法均有效的提高了算法的性能，基于属性选择的方法还能大大提高算法的效率。

朴素贝叶斯算法及其最近理论发展

胡鑫[1,2], 王志慧[2]，赵泽旗[3]

[1]National Key Laboratory of Crop Genetic Improvement, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[2]College of Plant Science and Technology, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[3]College of Horticul and Forestry Science, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

朴素贝叶斯分类器简单且高效，然而其基于属性间相互独立的假设限制了算法的应用，目前学者们从不同的角度提出了有针对性的改进算法，本文回溯了朴素贝叶斯分类算法的基本原理和近年来基于结构扩展、属性加权、属性选择、局部学习以及实例加权等多个方面的算法改进工作，各类改进工作均能在一定程度上满足朴素贝叶斯分类算法的属性间独立性假设，多种方法相结合的算法改进工作也取得相应进展，与单一方向改进相比效果更为明显，总的来说，各个方法均有效的提高了算法的性能，基于属性选择的方法还能大大提高算法的效率。

关键词: 朴素贝叶斯分类器，算法改进

1 概况

1.1 选题说明（本章节作者：赵泽旗）

朴素贝叶斯算法构建的分类器是一种很简单的软输出的概率生成模型，它主要实现流程是给定一系列数据集，通过这部分数据集形成先验的概率，然后再根据输入的数据各种属性信息，来预测和判断这个数据的分类，虽然它在实现过程中存在一种朴素的假设，即各个属性数据间是相互独立的，这一个假设在现实世界中几乎是不存在的，但是，当我们实际运用朴素贝叶斯分类器到一些场景中的时候，发现它又确实起到了不错的分类效果，同时因为它的简便易懂，作为机器学习算法的小白的我们几人，从一开始上课就对这个算法产生了深深的着迷。于是我们决定在老师上课讲授的基础上，做更进一步的对朴素贝叶斯分类器这一算法的了解和拓展。

1.2 朴素贝叶斯算法的基本原理（本章节作者：赵泽旗）

朴素贝叶斯分类器是依靠贝叶斯原理形成的一种朴素贝叶斯模型，和决策树模型相比，朴素贝叶斯分类器发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。同时，贝叶斯分类器模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。理论上，朴素贝叶斯分类器模型与其他分类方法相比具有最小的误差率。

1.2.1 数据的设定

$Data : \{(X_i, Y_i)\}_{i=1}^{N}$

$X_i \in R^p, Y_i \in \{0,1\}$

由数据的提供可以看到，我们首先需要一个足够大的数据准备，这个数据包括y变量和x变量，假设y变量为某一事物的分类，它有两种类别，0类别和1类别。X变量是一个p维向量，是描述y这个事物的一系列属性的数据，这个属性数据可以是多项式分布，也可以是属于高斯分布，也可以属于伯努利分布。

1.2.2 贝叶斯假设的原理

$$\begin{aligned} P[x|y] &= P[x1|y]P[x2|y]P[x3|y]P[x4|y]\&P[xn|y] \\ &= \prod_{i=1}^{N} P\left[x_i|y\right] \end{aligned}$$

要使用朴素贝叶斯分类器，我们首先要假设关于y的一系列属性x的概率是相互独立的，即某一个x属性的变换不会影响其他属性的改变。

1.2.3 贝叶斯分类器输出原理

贝叶斯公式：$P\left(A \cap B\right) = P\left(A\right) * P\left(B|A\right) = P\left(B\right) * P\left(A|B\right)$

这是贝叶斯公式的原型，根据后两个等式：

$P\left(A\right) * P\left(B|A\right) = P\left(B\right) * P\left(A|B\right)$

推导得：

$P\left(B|A\right) = P\left(B\right) * P\left(A|B\right) / P\left(A\right)$

这就是我们贝叶斯分类器的模型，然后将B替换成Y，将A替换成X即可得到：

$P\left(Y|X\right) = P\left(Y\right) * \frac{P(X|Y)}{P(X)}$

其中，$P\left(Y|X\right)$为后验的我们想要预测的结果，$P\left(Y\right)$是来自于我们的数据集的先验的概率，而$\frac{P(X|Y)}{P(X)}$这一部分相当于一个调整的因子，来调整我们先验的概率，从而得到最终后验的概率，达到对一个新数据的分类的判别的目的。

根据我们先前输入的数据集，我们可以得到一个先验的概率，此时有一个待分类的数据$y = \{x_1, x_2, x_3, x_4 \cdots x_p\}$，我们可以通过已知的先验的概率，和我们根据数据集观测到的调整因子，从而计算调整具有此特征的y属于哪一类的概率，将每一类的概率进行比较，选择可能性最大的，此时我们的判别过程就算结束了。

因为调整因子的分母对于同一个待测样本来说是相同的，我们也可以将上述公式继续推导化为如下形式:

$$\begin{aligned} h_{baues}\left(x\right) &= argmax_{y\in\{0,1\}} P\left[Y = y|X = x\right] \\ &= argmax_{y\in\{0,1\}} P\left[Y = y\right] * P\left[X = x|Y = y\right] \\ &= argmax_{y\in\{0,1\}} P\left[Y = y\right] * \prod_{i=1}^{N} P\left[X = x_i|Y = y\right] \end{aligned}$$

即我们算出y=0，和y=1时的值，然后选取其中最大的作为我们预测的结果即可。

1.3 拉普拉斯平滑处理（本章节作者：赵泽旗）

我们要预测的数据其中某一个特征的出现次数或者概率可能会由于提供的数据集中出现的次数为零，我们知道零乘以任何数都等于零，所以我们将$P\left(x_i|y\right)$改为$P\left(x_i + 1|y + v\right)$（假设是x，y为频率），v则表示

训练数据中包含多少中属性，这样，我们在套用计算公式：$argmaxy \in \{0,1\} P[Y=y] * \prod_{i=1}^{N} P[X=x_i|Y=y]$的时候在连乘的时候就不会出现零乘任何数都等于零的尴尬局面了。

1.4 朴素贝叶斯算法的分类（本章节作者：赵泽旗）

朴素贝叶斯算法分为高斯朴素贝叶斯、多项式朴素贝叶斯和伯努利朴素贝叶斯。他们的不同之处在于数据的属性不同，即他们可以面向不同的应用场景。

1.4.1 高斯朴素贝叶斯

高斯朴素贝叶斯算法假设所有特征都具有高斯分布，意味着他可以应用于数据属性是连续的情况，例如温度、湿度等连续数据。在高斯分布中，有68%的数据都分布在均值的一个标准差范围内，有96%的数据都分布在均值的两个标准差范围内。将这些连续的数据通过高斯分布映射出具体的概率，然后带入贝叶斯公式计算即可达到判别的目的。一般来说，入股样本属性的分布是连续值，使用高斯朴素贝叶斯比较好。

1.4.2 多项式朴素贝叶斯

多项式朴素贝叶斯就是先验为多项式分布的朴素贝叶斯。它假设特征是有一个简单多项式分布生成的。多项分布可以描述各种类型样本出现次数的概率，因此多项式铺贝叶斯非常适合用于描述出现次数或者出现次数比例的特征，比如该模型就非常适合用于文本分类，特征表示的是次数，例如在分类垃圾邮件的应用场景下某个词语出现的次数等等，其公式如下:

$P(X=x_{jl}|Y=y_i) = (x_{jl}+1) / (m_i+v)$

其中，$P(X=x_{jl}|Y=y_i)$是第i个类别的第j维属性的第l个取值的条件概率，mi是数据集中输出为第k类样本的个数。V是拉普拉斯平滑中代表数据属性的个数。如果样本属性的分布大部分是多元离散值，适合使用多项式朴素贝叶斯。

1.4.3 伯努利朴素贝叶斯

伯努利朴素贝叶斯分类器是先验为伯努利分布的朴素贝叶斯。假设先验的概率为二院伯努利分布，即数据属性取值只能是0和1，引用夏老师上课所讲的例子就是判断一个男生是否能嫁，需要看这个男生的品质，比如是否上进，可以用0或者1来代替。如果样本的属性是二元离散值或者很稀疏的多院离散值，可以使用伯努利贝叶斯分布。

1.5 朴素贝叶斯算法的应用举例（本章节作者：赵泽旗）

1.5.1 结婚对象问题

仍记得夏老师的判别是否是合适的结婚对象的例子，在我脑海中留下了深刻的印象，今天就实际操作出一个例子来简单模拟伯努利朴素贝叶斯分类器下面给出一个已知的数据：

求X=(帅，性格差，躺平)的男生向女生求婚，女生会嫁给他吗？

由上数据可知先验的P(嫁)= 5/10，P(否)= 5/10

根据公式：

P(嫁|帅性格差躺平)= P(嫁)*P(帅性格差躺平+1|嫁+2)/ P(帅性格差躺平)

P(否|帅性格差躺平)= P(否)*P(帅性格差躺平+1|否+2)/ P(帅性格差躺平)

我们只需要比较这两个P值谁大即可，我们可以看到两个P值的计算公式的分母是一样的，所以我们把公式简化成如下：

颜值	性格	上进	嫁否
帅	好	上进	嫁
丑	差	躺平	否
帅	差	上进	嫁
丑	好	躺平	否
帅	好	一般	嫁
丑	好	上进	嫁
帅	差	一般	否
丑	好	一般	否
帅	好	躺平	嫁
丑	差	上进	否

表 1: 结婚对象问题选择示例（自制）

(1)P(嫁|X)正比于P(嫁)*P(X+1|嫁+2)

(2)P(否|X)正比于P(否)*P(X+1|否+2)

进一步计算式子：

(1):P(嫁)*P(帅+1|嫁+2)*P(性格差+1|嫁+2)*P(躺平+1|嫁+2)

=5/10*(4+1/5+2)*(1+1/5+2)*(1+1/5+2)

=0.0291

(2):P(否)*P(帅+1|否+2)*P(性格差+1|否+2)* P(躺平+1|否+2)

=5/10*(1+1/5+2)*(3+1/5+2)*(2+1/5+2)

=0.0350

$0.035 > 0.0291$很遗憾这位男士虽然帅，但是女生并不愿嫁给他

1.5.2 垃圾邮件过滤问题

下面用多项式贝叶斯简单模拟一个过滤垃圾邮件的分类器，给出如下已知数据

Email	dear	friend	dinner	dicount	classification
1	√	×	×	√	normal
2	√	√	√	×	normal
3	√	√	√	×	normal
4	√	√	×	×	normal
5	√	√	×	×	normal
6	√	×	√	×	normal
7	√	√	×	×	normal
8	√	×	×	×	normal
9	√	√	×	√	spam
10	√	×	×	√	spam
11	×	×	×	√	spam
12	×	×	×	√	spam

表 2: 垃圾邮件分类示例（自制）

$P(N) = 8/(8+4) = 2/3$

$P(S) = 4/(8+4) = 1/3$

假设我们收到一封邮件，它的内容是dinner discount discount discount discount，可以看到这次discount出现了四次，如果在伯努利朴素贝叶斯中，可能出现的次数就不会影响到最终的分类结果，而在多项式朴素贝叶斯中，这种特征出现的次数也是可以影响到分类结果的。而且通过我杜撰的这个例子也能体现出拉普拉斯平滑的必要性，因为如果不进行平滑处理，dinner这个词汇在垃圾邮件中的概率将会是0，最终乘积比

较大小肯定是小于它是正常邮件的概率，但是显然它更有可能是一封垃圾邮件，为了避免这种由于训练数据不充足而导致的误判，我们还是需要进行拉普拉斯平滑处理。下面进行计算来对这封邮件分类。首先进行拉普拉斯平滑，计算得到：

$P(dinner|N) = (3+1)/(4+17) = 4/21$

$P(discount|N) = (1+1)/(4+17) = 2/21$

$P(dinner|S) = (0+1)/(4+7) = 1/11$

$P(discount|S) = (4+1)/(4+7) = 5/11$

正常邮件得分：$P(N)\cdot\cdot P(dinner|N)\cdot P(discount|N)^4 = 0.00001$ 垃圾邮件得分：$P(S)\cdot\cdot P(dinner|S)\cdot P(discount|S)^4 = 0.00122$ 计算结果判别该邮件为垃圾邮件。

1.6 朴素贝叶斯算法的缺点（本章节作者：赵泽旗）

朴素贝叶斯分类器作为一个入门级的算法，通俗易懂的代价背后就意味着算法本身存在一些问题，最大的问题就如它的名字一样，在于它假设了样本属性之间的独立性，这在现实生活中往往是不存在的，其次就是需要知道一个先验的数据，它本身并不存在自我学习的能力，它有许多模型，模型的选择对先验概率会产生一定的影响，这是导致它预测结果存在一定的错误率的原因。

2 朴素贝叶斯分类器发展概况

2.1 朴素贝叶斯分类器的决策特点以及决策边界（本章节作者：赵泽旗）

关于朴素贝叶斯分类器的决策特点，他是通过先验计算后验的方式，进行分类的。通过公式$P(Y)P(X|Y) = P(X)P(Y|X)$的变换，我们可以得到$P(Y|X) = P(Y)*P(X|Y)/P(X)$，其中$P(Y|X)$的含义可以理解为具有$X(x_1, x_2, x_3 \cdots x_i)$特征的样本是Y类别的概率，我们只要把这个样本是各种类别Y的概率都计算出来然后取最大的那个就能完成对样本的分类。有趣的是通过公式我们可以看到：$P(Y)$是所有样本中是Y样本的概率，是一个已知的先验的数据，$P(X|Y)$是在Y类别中具有X系列特征的概率，这两个数据，我们都可以从一开始提供的数据集中的许多样本中统计获得，$P(X)$是样本具有X特征的样本的概率，这个计算起来相对复杂，但是在求每个类别的概率的时候，这个分母永远是一致的，所以我们可以忽略。那这样来看，我们完成贝叶斯分类，就是完成了从已知数据集获得信息，然后对未知样本的类别进行判断决策的过程，十分的简洁。下图是引自网络中一个比较热门的关于鸢尾花类别，贝叶斯分类器的决策的可视化图片。可见虽然分类不是百分百的准确，但是在数据样本特征足够分离的情境下，朴素贝叶斯分类器确实能在一定程度上胜任分类的工作。

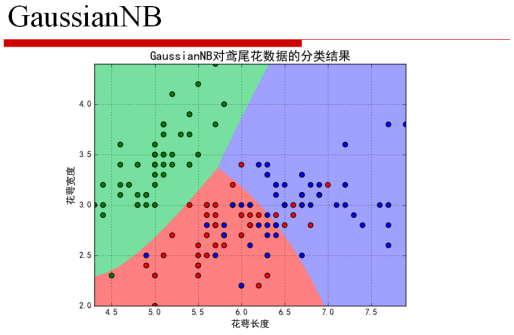


图 1: 鸢尾花分类[5]

2.2 朴素贝叶斯分类器的改进（本章节作者：胡鑫,王志慧）

朴素贝叶斯分类器是一种简单有效的分类方法，在人工智能、统计决策等领域具有广泛应用。通过对多种分类算法的对比研究发现，基于类条件独立性假设的朴素贝叶斯分类器与决策树、神经网络等分类模型等都具有很好的分类你性能，甚至在某些领域表现出更好的分类性能。但是由于类条件独立性假设在很多情况下都与现实情况不符，实际盈盈中属性间相关性和属性对分类的影响都不完全相同，假设的违背显著影响模型分类的精度。

针对朴素贝叶斯分类器条件独立性假设在很多情况下都与现实情况不符的缺点，其改进方法有很多，大体可以分为一下五类：一是结构扩展，这一类方法用有向边来表达属性间的依赖关系；二是属性选择，从原始属性中搜索出一个最佳的属性归约子集；三是属性加权，为不同的属性赋予一个不同的权值；四是局部学习，这类方法是利用局部学习原理在测试实例的邻域构建朴素贝叶斯分类器；五是实例加权，即给每个训练实例赋予不同的权值[2]。

2.2.1 基于结构扩展方法改进朴素贝叶斯分类器

结构扩展方法的基本思想就是用有限的有向边来表达属性变量间的依赖关系，进而扩展朴素贝叶斯分类器的结构。如何设计一个高效的结构扩展算法是学习扩展的朴素贝叶斯分类器的关键。如果无约束的确定所有属性节点的父节点，就需要学习有属性变量$A_1, A_2, \cdots A_m$组成的贝叶斯网络的结构。但是学习无约束的贝叶斯网络结构已经被证明是一个NP-难问题。因此广大学者提出了许多基于学习有约束的贝叶斯网络结构的模型和算法。前文所述的树扩展朴素贝叶斯分类器（TAN）就是一种典型的有约束的朴素贝叶斯网络结构。Keogh 等人提出一种超父亲的树扩展的朴素贝叶斯分类器（SP-TAN）的学习算法。与TAN 相同的是SP-TAN 假定属性节点之间的结构为树，不同的是其采用贪婪搜索的方法，通过产生超父亲节点每一步都在原有的树结构上增加一个最能提高当前分类器精度的边，直到最后添加任何一条边都不能提高分类精度为止，相比于TAN其有较高的时间复杂度和分类精度[8]。然而TAN模型的结构学习时间会随属性个数的增加而呈指数增长，这也使得对属性集庞大的数据分类时效率低下，针对这些问题Jiang等人提出了隐朴素贝叶斯分类算法(HNB)，该方法用一个隐藏的父节点表示属性间的依赖关系，有效的提高了贝叶斯方法的分类准确度[3]。

上述的三种方法都是基于学习属性节点之间的拓扑结构。同样的不需要学习结构的算法也有很多学者进行研究，例如Webb等人提出的平均一依赖性评估（AODE），其针对每一个属性节点学习一个树扩展的朴素贝叶斯分类器，并且每一个属性节点都是其他所有属性节点的父节点[17]。然后把这些树扩展的朴素贝叶斯分类器进行平均，模型中的所有树扩展的朴素贝叶斯分类器队最终的分类贡献是相同的，这一假设在实际中也并不合理，因此Jiang等人设计了一个加权平均一依赖性评估（WAODE）[16]。

近期，王峻利用x^2衡量属性间相关性，对属性进行分组，将相关性强的属性分为一组，各个属性分组之间相互条件独立。只在各个属性分组内通过添加有向边的方式表达相关属性间的相关性，将朴素贝叶斯分类器的扩展限定在每个属性分组内，从而简化扩展朴素贝叶斯分类器的结构，提高分类正确率[13]。用π_i作为变量集合X 的一个属性分组划分，在分类时假设各个属性分组之间相互条件独立，组内各属性相互依赖，通过合理选取属性分组来达到改进分类器的目的，基于属性分组的贝叶斯分类器可以用公式表示为:

$$\underset{c(\pi_1,\cdots\pi_n)}{argmax}\left\{p\left(c\right)\coprod_i=1^k p\left(\pi_i\right)|c\right\}=\underset{c(x_1,\cdots x_n)}{argmax}\left\{p\left(c\right)\prod_{i=1}^n p\left(x_{i1},\cdots,x_{in}|c\right)\right\}$$

式中π_i表示属性集X的一个子集，对原数据集合X 分组的合理性，将直接影响到分类的准确率，因此π_i的合理选取与组合是分类器改进的关键。该算法中的各个属性分组之间相互独立，组内各属性相互依赖。在每个属性分组中，在非类的父节点和子节点之间添加一条有向边来表示属性间的相关性，将朴素贝叶斯分类器的扩展限定在每个属性分组内，可以有效地简化扩展朴素贝叶斯分类器的结构，提高了分类的准确性。通过作者利用朴素贝叶斯分类器、树扩展的朴素贝叶斯分类器以及该算法对来自UCI的六个实验

数据集进行分类预测，也表明该模型的效果良好。杨航等人开发了基于R-vine Copula 的朴素贝叶斯分类器，其中R-vine Copula 模型以一种藤图形的结构来反映高维变量间的复杂相依结构，将属性之间的联合概率密度函数分解为一系列Pair Copula函数和边缘概率密度函数类型[14]。Alizadeh等人开发了多独立潜变量朴素贝叶斯分类器(MILC-NB)，该方法结构如图2所示，其主要用于在给定类变量信息的情况下放宽特征属性的独立性假设，其中属性集$\{Ai\}\, i = 1N$被划分为k个非重叠分区$\Lambda = \lambda ii = 1K$ 这时每个聚类中的属性在给定相关类属性信息时是有条件依赖性的，而与其他分区相比则具有独立性[15]。这一方法不仅保持朴素贝叶斯分类器结构简单的特点，还一定程度上解决了其属性独立性假设的缺陷，并且通过模拟数据实验表明其还适合解决罕见事件分类问题。

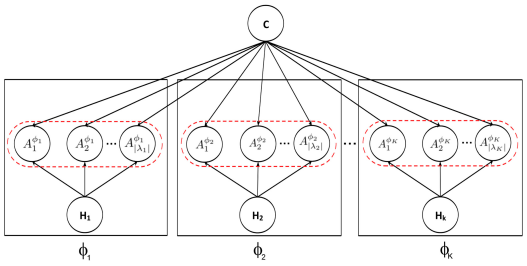


图 2: MILC-NB的网络结构图片来源

2.2.2 基于属性选择方法改进朴素贝叶斯分类器

属性选择包括属性删除和属性重构两个方向，理论上在利用朴素贝叶斯分类器进行分类的过程中，数据集包含的属性越多，其信息就越丰富，基于该数据集学习得到的模型识别能力就越强。然而事实并非如此，属性数据集可能会由于诸多原因包含无关的属性，反而会误导算法学习系统，并且在数据集属性个数较多时，容易出现“维度灾难”，高维度的数据集中属性间容易出现属性冗余，多重相关的问题，这不仅增加了模型计算的复杂度并且属性间的相关性也被模型忽略。当属性间存在冗余时，通过一定的规则剔除原数据集中的冗余属性就是属性删除方法。而属性重构则是在数据集属性间存在相互依赖关系时，把相互依赖的属性集使用一定的方法构造成一个新的属性，并用新属性替代原属性集，在一定程度上满足独立性要求，从而提升分类准确性。因此在实际应用中，我们用属性关联度表示一个属性和类属性间的相关性, 它反映这个属性对分类结果影响的程度; 用属性冗余度表示一个属性和其他属性之间相关性, 它反映这个属性和其他属性间的依赖度，对数据集进行属性选择得到一个属性子集，使得属性子集中的属性和类属性总体相关性较大，属性间的冗余度较小，这不仅可以提高算法的效率，还能提高算法的性能。

属性选择问题是一个组合最优搜索问题，解决这一类问题的方法有很多，例如爬山搜索、贪婪搜索、蚁群搜索等。Langley 和 Sage 提出一种基于分类精度和贪婪搜索的属性选择方法Selective Bayesian classifiers(SBC)。其首先假定选择的属性子集为空或为满，然后每一步增加或删除一个最能改善当前朴素贝叶斯分类器精度的属性，直到最后增加或删除其中一个任何一个属性都不能改善当前朴素贝叶斯分类器的分类精度时为止。基于粗糙集的属性归约算法以属性集的核作为属性归约的起点，每次选择最重要的属性加入到属性归约集中，在前向选择结束后，属性归约集就包含了一些重要作用的属性，且没有影响属性集和类属性之间的依赖关系。最后的反馈过程，从属性归约集中逐个去掉属性，如果在这一过程中造成属性集和类属性间的依赖程度则保留该属性，最终保留的属性归约集就是最佳属性规约集。

目前基于属性选择改善朴素贝叶斯分类器的研究仍在进行。周艳等人利用奇异值分解去除噪音数据再结合主成分分析的方法构造新属性取代旧属性，构造朴素贝叶斯分类器。作者使用该模型对来自UCI的8个数据集进行分类，与朴素贝叶斯分类器相比较，结果发现基于SVD+PCA的朴素贝叶斯分类器性能得到提升,主要是因为当数据既存在噪音时，利用SVD可以去除噪音干扰，重构一个接近原始数据集的数据集，减少主成分中的无用信息，而主成分分析考虑到属性间的相关性，处理后的数据集的新属性间互不相关，一

定程度上满足了属性间相互独立的条件[7]。张晓莉等人采用了属性选择的思想对朴素贝叶斯分类模型进行改进，他们使用支持向量机算法对属性进行筛选，通过实验结果可以看出改进后的朴素贝叶斯分类器比原始模型有了较大提升[11]。

2.2.3 基于属性加权方法改进朴素贝叶斯分类器

属性加权的基本思想是在训练阶段根据各个属性的重要程度相对应的为其赋予一个权值，然后属性加权后的训练实例集上学习一个朴素贝叶斯分类器,这种方法既释放了属性条件独立性假设，又继承了朴素贝叶斯分类器的分类精度和计算速度，其中如何设计一个高效的属性加权算法是属性加权的朴素贝叶斯分类器的关键。目前也已经有很多学者做出尝试，例如前文提到的加权平均一依赖性评估（WAODE）直接用根节点对应的属性变量和类变量之间的相互信息来对所有的树扩展的朴素贝叶斯分类器加权。Zhang等人提出了加权朴素贝叶斯分类器（WNBC），给出了多种属性权值的计算方法：爬山法、增益比法、马尔科夫链蒙特卡洛法、爬山法与增益比法相结合、马尔科夫链蒙特卡洛法与增益比法相结合，通过这些不同方法与朴素贝叶斯分类器的结合，从数据中学习加权朴素贝叶斯分类器的的各属性的权重，从而获得具有精确排序的加权朴素贝叶斯分类器，结果表明加权朴素贝叶斯分类器的分类结果明显优于朴素贝叶斯分类器[18]。Hall 等人假设一个预测属性应该与其他属性所依赖的程度成反比，通过构造未剪枝的决策树并依据每个属性在决策树中的深度大小来估计属性依赖度，提出一种基于决策树设置属性权重的朴素贝叶斯分类器算法。张明卫等人提出一种基于相关系数的加权模型，该改进模型的权重有属性之间的相关系数决定[1]。

周小燕等人开发了一种基于DC-TF-IDF（类间集中度-类内分散度-频数-逆文档频法）的加权朴素贝叶斯分类器，其中：TF代表词频，IDF代表逆文档频率，一个特征词在文本中出现的频率越小，它区别类别的能力也就越大

$TF = \frac{n_y}{N}$

$IDF = log\frac{n_{ci}}{n_y}$

CD代表类间集中度表示特征项在各个类别中分布的均匀程度，当特征项越集中分布在某些类别中，而不是均匀的分布在各个类中，此时含有的类别信息就越多月能够代表该类别，DD代表类内分散度，指的是某一个类别中特征项分布的均匀程度，在某个类的文本中特征项出现的越多说明特征越分散，能够在一定程度上代表该类，则特征对分类有更多的贡献，把CD-DD作为一个整体对TF-IDF函数做出调整得到新函数

$DC\left(c_i, y\right) = \frac{n_{ci,y}}{n_y} \cdot \frac{n_{ci,y}}{n_{ci}}$

$DC - TF - IDF\left(c_i, y\right) = \frac{n_{ci,y}^2+1}{n_y n_{ci}} \frac{n_y}{N} \cdot log\frac{n_{ci}+1}{n_y}$

最终,为每个属性引入一个权值DC-TF-IDF对朴素贝叶斯分类器模型进行改造，得到哦加权的朴素贝叶斯分类器模型：

$$\begin{aligned} DC - TF - IDF - P\left(c_i, y\right) &= DC - TF - IDF\left(c_i, y\right) p\left(x/i\right) p\left(i\right) \\ &= \prod_{i=1}^{k} \left(\frac{n_{ci,y}^2 + 1}{n_y n_{ci}} \frac{n_y}{N} \cdot log\frac{n_{ci}+1}{n_y} \right) \frac{n_{yi}+1}{n_{ci}+n} \frac{n_{ci}+1}{n+c} \end{aligned}$$

最后作者利用该模型对来自网络的电影评级进行预测，结果同样符合预期，与传统的朴素贝叶斯分类器模型相比准确度有了明显的提升[10]。

2.2.4 基于局部学习方法改进朴素贝叶斯分类器

局部学习，又称为实例选择。这种方法利用局部学习原理在测试实例的邻域构建朴素贝叶斯分类器。

局部分类比全局分类更能够解决负责的分类问题。当分类问题比较复杂时，全局分类需要再在全局上学习一个复杂的分类模型，而局部分类只需在局部区域学习一个简单的分类模型，就能达到一个很好的分类效果。

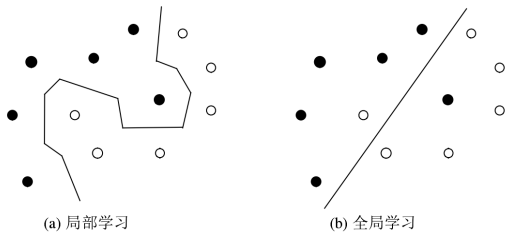


图 3: 局部学习与全局学习[6]

朴素贝叶斯模型具有简单性和有效性，但是其属性条件独立性假设在实际应用中难以成立，而属性加权是降低属性条件独立假设对分类器性能影响的主要途径。属性选择也经常作物提高分类器性能的方法，它不会改变NB模型的结构，同时可以有效提高NB的分类性能，但是实际上属性对类属性的归属的影响不同，而属性选择不能区分不同属性在分类过程中的重要程度。2014年Taheri提出用局部优化算法对于不同类属性的权重。有学者提出了基于差分进化算法求属性权重，通过优化算法求负条件对数似然或均方误差的最小值来设置属性权重，这两种方法不仅降低了违背属性条件独立性假设对分类模型的影响，还使得分类器的整体性能得到改善。但这些属性加权方法都是从整个数据集合上学习属性权重，这样的属性权重对于每个测试实例只在平均意思下十最好的，并不能反映真实的每个测试实例的属性对自身分类的重要程度。2002年Frank将局部实例加权技术应用于NB模型，为每个测试实例基于加权近邻实例建立局部实例加权朴素贝叶斯模型，但是仅仅对实例加权，忽略了不同属性对分类性能的影响。2018年张伟等提出了一种基于数据驱动的懒惰式局部属性加权方法，它在每个测试实例的近郊集合上学习属性权重，并通过最优化方法建立响应的局部属性加权朴素贝叶斯模型，其试验结果该模型具有交稿的分类准确率。

2.2.5 基于实例加权方法改进朴素贝叶斯分类器

实例加权，即给每个训练实例赋予不同的权值，然后在实例加权后的训练实例集上学习一个朴素贝叶斯分类器。在给定训练实例集构建朴素贝叶斯分类器的过程中，不同实例重要程度不同，对应在模型的贡献也不一样，因此实例加权的方法为改进朴素贝叶斯的一个重要方法。朴素贝叶斯是在原有训练实例集上估算先验概率、条件概率，而实例加权的朴素贝叶斯分类器在实例加权后的训练实例集上估算先验概率和条件概率，具体计算公式[12]如下：

$$arg\max_{c\notin C} P(c)\prod_{j=1}^{m} P(a_j|c)$$

其中$P(c)=\frac{\sum_{i=1}^{n} w_i\delta(c_i,c)+1}{n+n_c}$,$P(a_j|c)=\frac{\sum_{i=1}^{n} w_i\delta(a_{ij},a_j)\delta(c_j,c)+1}{\sum_{i=1}^{n} w_i\delta(c_j,c)+n_i}$

有学者利用Boosting技术训练实例加权，在加权的实例集上构建朴素贝叶斯分类器进行监督训练，具体为在每次迭代中，Boosting算法关注被误分的实例，给予更大的权值，经过多次迭代提升朴素贝叶斯分类性能。判别加权的朴素贝叶斯分类器，其算法假定训练实例的初始权重为1，基于训练实例集构建朴素贝叶斯分类器，根据对每个实例的类概率估测重新分类权值，经多次迭代更新实例权值，知道达到预设的循环次数停止迭代。在最后确定的加权实例集上构建出朴素贝叶斯分类器。

2.2.6 多方法结合改进朴素贝叶斯分类器

从前文所述的可以看出各种改进方法与传统的朴素贝叶斯分类器相比对模拟数据的分类能力有利明显的提升，在一定程度上提高了分类准确率，但是在少数情况下其分类准确率或者是计算效率会弱于传统的

朴素贝叶斯分类器，例如基于属性选择算法改进朴素贝叶斯分类器依然没有考虑各个属性的重要程度，而基于属性加权的改进算法虽然能很好的避免属性条件独立性假设带来的问题，但是由于增加了加权系数的计算过程，其分类效率相对较低。为了改善单一改进方法的不足，研究人员也同样开发了很多基于多种方法结合改进朴素贝叶斯的改进算法。例如王行甫等人将基于相关的属性选择算法(CFS)和加权朴素贝叶斯分类算法(WNBC)结合，提出了一种基于属性选择的改进加权朴素贝叶斯分类算法(ASWNBC)，该方法有效的结合了属性选择算法(CFS)的简化性和加权朴素贝叶斯分类算法(WNBC)的高效性，在提高分类效率的同时增加了分类准确率[4]。其中CFS算法是基于相关的属性选择算法，他能生成一个评价属性子集分类能力的评估函数：

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k\left(k-1\right)\overline{r_{ff}}}}$$

式中分子表示属性和类属性间关联性大小的度量，分母代表子集属性间冗余性大小的度量，使其可以从属性和类属性间关联的强度及属性间相互影响导致的冗余性两方面考虑问题，能有效筛选与类属性相关的属性子集，提高后续分类的准确性。WNBC算法采用加权公式如下：

$$W_{A_kC_i} = \frac{T_{A_k} + \frac{T_{(A_k=v\cap C_i)}}{T_{(A_k=v)}}}{T_{A_k}}$$

式中$W_{A_kC_i}$不仅反映了每个属性对分类的影响程度，而且量化了每个属性及其类别之间的关联，能有效提高最终模型的分类准确率。

上述方法中如果在去冗余过程中剔除了与类属性关联度较大的属性时则会导致分类准确度降低。周艳等人同样采用属性选举与属性加权结合的方法改进朴素贝叶斯算法，其在属性选择部分采用SVD和PCA算法，构造一个新属性集以近似满足属性条件独立性假设，再结合加权朴素贝叶斯模型，将弱化条件独立假设与弱化属性同等重要的假设结合构造了新的模型。通过模拟数据分类检测结果可以发现结合多方法的改进模型相较于单一方法改进的模型有了不错的提升[7]。

对朴素贝叶斯分类算法的改进中不仅有基于加权朴素贝叶斯分类算法的改良方法，也有很多学者对结构扩展的朴素贝叶斯分类算法提出了改进方案，例如：秦怀强等人提出的基于属性加权的隐朴素贝叶斯分类算法，既考虑了各个属性之间的依赖关系，又兼顾了各个属性对分类的贡献程度，从其模拟数据的测试结果看，相较于单一改良措施也获得了很大提升[9]。由于隐朴素贝叶斯在为每一个属性节点引入一个表示属性间依赖关系的隐藏父节点之后，增加了模型的结构复杂度以及算法的时间复杂性，降低了分类效率。为此瞿忠魁等人开发了自适应最大相关性属性选择的隐朴素贝叶斯分类算法，前文中提到属性选择算法存在一个缺陷就是当筛选过程中过度筛选对分类有利的属性或是对冗余属性筛除力度不足都会影响最终模型的分类准确度[3]。该算法则通过离散化属性评价标准在通过对离散化结果的判断自动选择后续分类所需属性，同时删除与分类相关性小的冗余属性。杜婷等人则将结构扩展、属性选择和属性加权三种方法结合起来开发了基于属性选择的加权隐朴素贝叶斯分类算法，该算法将筛选后的属性集划分为强属性子集（属性间有较强的依赖关系）和弱属性子集，对强属性集采用加权隐朴素贝叶斯分类算法而对弱属性子集使用朴素贝叶斯分类算法[**?**]。通过模拟数据分类实验可知利用这种划分等级的属性子集能够在保证分类准确的的同时有效的降低计算复杂度提高分类效率。

2.2.7 小结

朴素贝叶斯分类算法是贝叶斯定理的经典算法，其具有结构简单、计算高效的特点，但是基于属性条件独立性假设限制了其适用范围，同时随着技术的发展用以分类的属性集不断增大、数据类型去向多样化，导致属性集中常存在无关或者冗余的属性，往往会给分类带来负面影响，既会降低分类准确度也会减缓计

算效率。学者们也针对不同的问题提出了多样的改进方案，例如以去除冗余属性为目标的基于属性选择的朴素贝叶斯分类算法，以解释不同属性对类属性重要程度为目的的属性加权改良算法，以及综合多种改良方法综合解决多项问题的复合改进算法等等。不同的数据可能适合不同的模型，改进后的模型也并不一定对所有类型数据的分类效果均有提升，因此我们也要依据数据的类型选择不同的算法以达到最佳的分类效果。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：王志慧）

根据组内成员的专业背景，通过组内沟通我们小组选择了算法逻辑较为简单的朴素贝叶斯算法，也是夏老师在数据挖掘课程第一章中所讲的算法。本文将对朴素贝叶斯算法原理、分类及优缺点进行详细描述，介绍了朴素贝叶斯分类器的特点，梳理了五种朴素贝叶斯分类器的改进方法。

论文第一部分内容，首先对选题说明作出了介绍，然后详细介绍了朴素贝叶斯算法的基本原理及分类，同时根据自己的理解和夏老师的启发，赵泽旗杜撰了一个例子即结婚对象问题，最后介绍了朴素贝叶斯算法的优缺点。贝叶斯定理是由英国数学家托马斯·贝叶斯提出的。贝叶斯定理是用来解决两个条件概率之间的关系问题，在已知P(X—Y)时如何获得P(Y—X)的概率，假设特征P(X)在特定结果P(Y)下是独立的，其思想为基于先验概率和数据综合得到后验概率。随着计算机与互联网技术的不断发展，工作和生活的各个方面都发生了巨大的变化，涌现出大量的数据信息；尽管拥有了海量的数据信息，但是获取“有价值”信息的难度却越来越大。数据挖掘和机器学习的发展，使得我们能够对数据进行高效且智能的分析，挖掘出数据背后蕴含的不为人知的重要信息。分类是数据挖掘和机器学习的核心问题之一，其方法在生活中被广泛的应用，比如文本处理、图像识别、语音处理等等领域。分类方法主要包括贝叶斯分类法、决策树分类法、神经网络分类法、支持向量机分类法等，其中贝叶斯和决策树是应用最广的两种方法。两者相比，贝叶斯分类法是通过比较不同样本属于各个类别的概率，然后决定样本的归属，易于构造，因此易于应用到大数据环境中，但是这种方法对缺失数据的敏感度不高，在数据完全随机缺失情况下，该模型可以从观测数据中估计边缘分布来获得对整体数据统计特性的有效估计，具有较强的鲁棒性。朴素贝叶斯是最原始的贝叶斯方法，算法原理简单且对数据的适应性强。在特征条件独立假设和贝叶斯定理的基础上得到的分类方法就是朴素贝叶斯法，在现实生活中有广泛的应用，为决策提供了一定的参考依据，比如基于邮件标题和内容判别是否为垃圾邮件，基于某天的天气状况判别今天是否适合踢足球，基于病例的各项指标判别病人是否患病等等。朴素贝叶斯算法的优缺点，其优点包括：较为稳定的分类能力，由于朴素贝叶斯模型的数学基础是贝叶斯定理，因此作为概率的分类算法，该算法具有扎实的数学基础；模型结构的简单，在处理大样本数据是具有很大的优势，由于属于条件独立性的假设，因此构建的模型比其他分类算法，朴素贝叶斯模型在建模过程中的运算速度得到了很大的提高；模型逻辑结构简单，在用于解决实际问题时，运算出的结果与实际相结合可以得到很好的解释。缺点包括：数据样本量较小时，分类结果可能会有较大的偏差，比如实际发生的某种情况可能并没有出现在训练样本；属性条件独立使计算更加简单，但是分类效果降低，因为在实际生活中，事物间或多或少存在一定的关联性，假如属性间存在较强的相关性，那么分类结果与实际会存在一定的误差。

论文第二部分内容，详细讲述朴素贝叶斯分类器的发展概况，主要介绍其改进方法。赵泽琪对朴素贝叶斯分类器的决策特点和决策边界进行了说明；胡鑫介绍了朴素贝叶斯分类器以及为何要对其进行改进；胡鑫和王志慧先后介绍了5种朴素贝叶斯分类器改进的方法，其中包括：结构扩展法、属性选择法、属性加权法、局部学习和实例加权法。朴素贝叶斯分类器结果简单，但是基于简单却不现实的假设，很多情况下基于朴素贝叶斯的分类很难达到预期的效果。针对朴素贝叶斯的属性独立性假设的限制，国内外学者提出了各种不同的算法，用于改善朴素贝叶斯算法的性能。其中结构扩展方法的基本思想是用有限的有向边来

表达属性变量间的依赖关系，进而扩展朴素贝叶斯分类器的结构；属性选择方法的基本思想是在整个属性空间选择一个最佳属性子集，然后在属性规约后的训练实例集上学习一个朴素贝叶斯分类器；局部学习法是在整个训练实例集的局部构建朴素贝叶斯分类器；属性加权是给不同类别的不同属性引入一个权系数对经典的朴素贝叶斯模型进行了改进；实例加权是在训练阶段给每个实例赋一个权值，在实例加权后的训练实例集上学习一个朴素贝叶斯分类器。

论文第三部分是按照老师给定的模板和实际情况进行撰写，介绍了长文的构思和撰写的具体过程，同时对于论文前两个部分所述内容表达了一些自己的理解。

3.2 所参考主要资源（本章节作者：王志慧）

1、本文的主要参考来源已列在下方参考文献中

2、参考了周志华老师的《机器学习》

3、参考了夏静波老师的课件（https://hzaubionlp.com/data-mining/）

3.3 人员分工（本章节作者：王志慧）

人员分工基本上延续了短文的模式，正文部分有三个章节构成，根据大家的兴趣和工作量进行分工，赵泽琪负责第一部分概况和第二部分2.1的写作，胡鑫主要负责第二部分2.2.1-2.2.3，2.2.6-2.2.7的写作，王志慧负责第三部分、第二部分2.2.4和2.2.5的写作。

参考文献

[1] 张明卫，王波，张斌，朱志良. 基于相关系数的加权朴素贝叶斯分类算法. 东北大学学报, 29:952–955, 2008.

[2] 蒋孝良. 朴素贝叶斯分类器及其改进算法研究. PhD thesis, 中国地质大学, 2009.

[3] 瞿忠魁. 自适应属性选择的隐朴素贝叶斯算法研究及其应用. Master's thesis, 湖南大学, 2014.

[4] 王行甫，杜婷. 基于属性选择的改进加权朴素贝叶斯分类算法. 计算机系统应用, 24:149–154, 2015.

[5] 小象社区. https://wenda.chinahadoop.cn/question/4499, 2016.

[6] 毛承胜. 基于贝叶斯决策理论的局部分类方法研究及其应用. PhD thesis, 兰州大学, 2016.

[7] 周艳. 朴素贝叶斯分类器的研究与改进. Master's thesis, 厦门大学, 2017.

[8] 张伟，王志海，原继东，刘海洋. 一种局部属性加权朴素贝叶斯分类算法. 北京交通大学学报, 42:14–21, 2018.

[9] 秦怀强. 若干改进朴素贝叶斯分类算法的研究与应用. Master's thesis, 山东科技大学, 2018.

[10] 周小燕. 朴素贝叶斯分类的研究及应用. Master's thesis, 重庆大学, 2019.

[11] 张晓莉. 朴素贝叶斯分类器及其改进算法研究. Master's thesis, 山东科技大学, 2020.

[12] 邱晨. 贝叶斯学习算法及其应用研究. PhD thesis, 中国地质大学, 2020.

[13] 王峻. 基于属性分组的扩展朴素贝叶斯分类器. 洛阳理工学院学报, 31:85–88, 2021.

[14] 杨航，刘赫，夏美美，范元静. 基于r-vine copula 理论的改进朴素贝叶斯分类器. 甘肃科学学报, 33:12–16, 2021.

[15] Sasan H. Alizadeh, Alireza Hediehloo, and Nima Shiri Harzevili. Multi independent latent component extension of naive bayes classifier. *Knowledge-Based Systems*, 213:106646, 2021.

[16] L. Jiang and H. Zhang. Weightily averaged one-dependence estimators. 2006.

[17] Geoffrey I. Webb, Janice R. Boughton, and Zhihai Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58:5–24, 2005.

[18] Harry Zhang and Shengli Sheng. Learning weighted naive bayes with accurate ranking. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 567–570. IEEE, 2004.

2.2 《线性回归及其算法实现研究》——吴航，李信，张宇豪

摘要：线性回归算法因为结构简单，可解释性好，实现简单，在各个领域得到广泛应用。线性回归算法的目的是试图学得一个线性模型以尽可能准确地预测实值输出。基于线性回归的思路，演化发展了逻辑回归 (LR)、因子分解机 (FM) 等，而且深度学习的 CNN、RNN、Attention 机制都有线性回归的影子，所以学习，了解和使用线性回归算法有其必要性。本文着眼于线性回归的算法实现研究，分析了线性回归的基本思想，运行过程与求解方法，在此基础上，基于公开的保险预测数据集，进行了线性回归的算法实现，对于该数据集作出了线性回归的分析，并针对实现过程中出现的具体问题进行评价与分析，总结了线性回归过程中出现的优缺点。

线性回归及其算法实现研究

吴航 [1], 李信 [1]，张宇豪 [1]

[1]College of Informatics, Huazhong Agricultural University , Wuhan, Hubei Province, P.R.China

摘要

线性回归算法因为结构简单，可解释性好，实现简单，在各个领域得到广泛应用。线性回归算法的目的是试图学得一个线性模型以尽可能准确地预测实值输出。基于线性回归的思路，演化发展了逻辑回归（LR）、因子分解机（FM）等，而且深度学习的 CNN、RNN、Attention 机制都有线性回归的影子，所以学习，了解和使用线性回归算法有其必要性。本文着眼于线性回归的算法实现研究，分析了线性回归的基本思想，运行过程与求解方法，在此基础上，基于公开的保险预测数据集，进行了线性回归的算法实现，对于该数据集作出了线性回归的分析，并针对实现过程中出现的具体问题进行评价与分析，总结了线性回归过程中出现的优缺点。

关键词：线性回归求解，最小二乘法，正则化，梯度下降，残差估计，R2 评估

1 概况

1.1 选题说明（本章节作者：吴航，李信）

线性模型的目的是试图学得一个通过属性的线性组合来进行预测的函数。线性模型形式简单、易于建模，但却蕴涵着机器学习中一些重要的基本思想，许多功能更为强大的非线性模型可在线性模型的基础上通过引入层级结构或高维映射而得。此外线性模型有很好的可解释性。

线性回归是线性模型的一种，是利用数理统计中回归分析，来确定两种或两种以上变量间相互依赖的定量关系的一种统计分析方法。线性回归假设特征满足线性关系，根据给定的训练数据训练一个模型，并用此模型进行预测，期望可以得到一个接近真实结果的实值输出。本文旨在研究线性回归算法的实现，我们将主要的内容放在了算法的原理分析与基于公开数据集的实现上。我们由回归思想，算法目的，以及线性回归的算法流程开始，分析了线性回归算法的原理，同时对线性回归的损失函数以及求解方法进行了推导。

为了更好的解释线性回归的流程，我们引入了一个公开的数据集，在此数据集上，我们基于线性回归算法对此数据集做出了一些数据分析，以解释我们的推导过程，同时通过一些评价标准来评估线性回归对现实问题的预测是否符合我们的预期，对于评估的结果，我们还进行了一些针对数据处理和优化的分析，以便了解如何使得线性回归达到更高的精确要求，如何使线性回归模型得到的回归结果与真实结果更接近。相信掌握了线性回归模型的推导过程以及算法原理后对我们后续学习其他更难的算法会有很大的启发。

1.2 该算法基本原理（本章节作者：张宇豪）

对于线性回归的理解，可以将其分割为两者：“线性”与“回归”。首先看“回归”，对于“回归”的官方定义为 [1]：研究一组随机变量 $(Y_1, Y_2, ..., Y_i)$ $(X_1, X_2, ..., X_k)$ 变量之间关系的统计分析方法，又称多重回归分析，公式如式 (1)。当其与线性进行结合，则可以理解为当研究的自变量与因变量为线性关系时的一种特殊线性模型。因此如果从样本数据出发构建了该模型，确定了模型中的未知变量，即可使用该模型对需要预测结果的新数据进行有效的预测，所以未知变量的值则直接与最后预测结果的好坏直接挂钩。

$$h_\theta(x) = \sum_{i=0}^{k} \theta_i * x_i \tag{1}$$

θ 中每个分量都是估计表达式函数 h(x) 中的参数。

求解 θ 的目标函数则可以使用最小二乘，对预测值与实际值进行方差求和 [2]，在其最小的情况下的 θ 值则是所求；以及递归下降法，利用其偏导函数，从最开始的初始值一步一步逼近最后的最优解。但是当样本特征很多，样本数据相对较少时，则会出现过拟合的情况。针对于这种情况，需要添加 L1 或者 L2 正则化项对损失函数进行进一步优化，从而增加其适用范围以及减少求解难度。

2 线性回归

在机器学习中的主要任务便是集中在两个问题：回归与分类。分类，如字面意思便是将目标分为不同的种类，如垃圾邮件识别等；而回归则是预测出一个具体的数值，如房价预测等。在回归中最基本的便是线性回归，即当研究的自变量与因变量为线性关系时的一种特殊线性模型。简单的线性回归只包含一个特征值如式 Y=λ_0+λ_1X，其中 λ 表示式 Y 的未知参数，但是实际的情况中，通常需要计算多个特征值对结果的影响如式 Y=λ_0+$\lambda_1 X_1$+$\lambda_2 X_2$+$\cdots$+$\lambda_n X_n$，而机器学习的任务则是通过对样本数据进行有监督的学习，从而得到所有 X 的系数 λ，具体则是构造损失函数，最小化损失函数来确定参数。在本实验中通过最小二乘法来构建损失函数，计算最优的 λ 值。最终通过判定系数 R2 等分析数据来表示实际值与回归方程预测值之间的拟合程度，其值越接近 1，表示回归模型的拟合程度越好。

2.1 最小二乘法（本章节作者：张宇豪，李信）

在机器学习中，所有的机器学习算法都或多或少的依赖于对目标函数最大化或者最小化的过程，我们常常把最小化的函数称为损失函数，它主要用于衡量机器学习模型的预测能力。回归会预测给出一个数值结果而分类则会给出一个标签。而最小二乘法就是损失函数的一种计算方法，即衡量估计值与实际之间的误差大小，将每个数据点的估计值与实际值进行作差，然后平方求和，如下式：

$$\arg\min_{\theta} \sum_{i=1}^{n} \varepsilon_i^2 = \arg\min_{\theta} \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 \tag{2}$$

其中 Y 为实际值，$\widehat{Y}$ 为预测值。最小二乘法则是用于求解此目标函数的最优值，它通过最小化误差的平方和寻找匹配项，所以又称为最小平方法，而本实验使用的就是最小二乘法，其求解方法如下：

$$\frac{\partial J(\theta)}{\partial \theta_i} = 2\sum_{i=0}^{n} (Y_i - X_i\theta) = 0 \tag{3}$$

求解得到的系数即为解。但是在最小二乘法的线性回归中如果将损失函数转换为矩阵的形式即：

$$L(w) = \sum_{i=1}^{N} ||W^T x_i - y_i||^2 \tag{4}$$

W 是一个 $p*1$ 的向量，表示 x_i 的系数

另外

$$x_i \in \mathbb{R}^p \; y_i \in \mathbb{R} \; i = 1, 2, 3, ..., N$$

$$X = (x_1\ x_2\ ...\ x_N)^T = \left\{ \begin{array}{c} x_1^T \\ x_2^T \\ . \\ . \\ . \\ x_N^T \end{array} \right\} = \left\{ \begin{array}{c} x_{11}\ x_{12}\ ...\ x_{1p} \\ x_{11}\ x_{12}\ ...\ x_{1p} \\ \cdots \\ \cdots \\ \cdots \\ x_{N1}\ x_{N2}\ ...\ x_{NP} \end{array} \right\} \tag{5}$$

$$Y = \left\{ \begin{array}{c} y_1 \\ y_2 \\ . \\ . \\ . \\ y_N \end{array} \right\}_{Nx1} \tag{6}$$

对 L(W) 进一步求解

$$L(w) = \sum_{i=1}^{N} (W^T x_i - y_i)^2 \tag{7}$$

$$= (W^T x_1 - y_1\ W^T x_2\ ...\ W^T x_N - y_N) \left\{ \begin{array}{c} (W^T x_1 - y_1)^T \\ (W^T x_2 - y_2)^T \\ . \\ . \\ . \\ (W^T x_N - y_N)^T \end{array} \right\} \tag{8}$$

$$= [W^T (x_1\ x_2\ ...\ x_N) - (y_1\ y_2\ ...\ y_N)] \left\{ \begin{array}{c} x^T W_1 - y_1^T \\ x^T W_2 - y_2^T \\ . \\ . \\ . \\ x^T W_N - y_N^T \end{array} \right\} \tag{9}$$

$$= (W^T X^T - Y^T)(XW - Y) \tag{10}$$

$$= W^T XW - W^T X^T Y - Y^T XW + Y^T Y \tag{11}$$

$$= W^T X^T XW - 2W^T X^T Y + Y^T Y \tag{12}$$

此时由于 $W^T X^T Y = (Y^T XW)^T$, 因为 $W^T X^T Y$ 是一个数，因此 $W^T X^T Y = Y^T XW$. 所以之后只需要对 W 最优化使得 L 最小即可 [3]

$$\widehat{W} = \arg\min_W L(W) \tag{13}$$

令

$$\frac{\partial L(W)}{\partial W} = 2X^T XW - WX^T Y = 0 \tag{14}$$

则

$$W = (X^T X)^{-1} X^T Y \tag{15}$$

以上是矩阵角度下对最小二乘法的表达，下面将从几何角度看最小二乘法。

从每一个数据点的误差来看，每一个点对应的 y_i 与其在 $f(w)$ 上所对应的点 $W^T x_i$ 之间的差值便是误差。因此将所有点的误差求和 $\sum_{i=1}^{N} ||W^T x_i - y_i||^2$，使得其最小，便可以求得最优的回归函数。
从投影角度来看:

$$X_{N\times p} W_{p\times 1} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NP} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix} \tag{16}$$

$$= \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + \cdots + x_{1p}w_p \\ x_{21}w_1 + x_{22}w_2 + \cdots + x_{2p}w_p \\ \vdots \\ x_{N1}w_1 + x_{N2}w_2 + \cdots + x_{Np}w_p \end{pmatrix} \tag{17}$$

$$= \left(\begin{pmatrix} x_{11} \\ x_1 \\ \vdots \\ x_{N1} \end{pmatrix} + w_2 \begin{pmatrix} x_{12} \\ x_{22} \\ \vdots \\ x_{N2} \end{pmatrix} + \cdots + w_p \begin{pmatrix} x_{1p} \\ x_{2p} \\ \vdots \\ x_{Np} \end{pmatrix} \right) \tag{18}$$

因此可以将 X 中的每一列看作一个向量，XW 便是 W 对 X 列向量的线性组合。从二维角度来看，每一个数据点并不都在同一条直线上，引入到高维空间中，便是 Y 无法由 X 的列向量线性表示，即 Y 不属于 X 的列空间，因此需要找到 Y 在 X 列空间上的投影（即 X 列向量的一个线性组合）假设 XW 是 Y 在 X 列空间上的投影，Y-XW 其垂直于 X 列空间（即与 X 的每一个列向量都垂直），因此 $X^T(Y - XW) = 0$, 所以可以求得 $W = \left(X^T X\right)^{-1} X^T Y$。在这里第一种角度把误差分散在了每一个数据点上，第二种角度把误差分散在了 X 的每一个列向量上，p 个维度上面。但不同的角度都得到了同样的结果即：$W = \left(X^T X\right)^{-1} X^T Y$

但最后得到的 $(X^T X)^{-1}$ 并不都是可逆的，需要进行正则化扩大其使用范围，增加其适用性，同时防止过拟合。

2.2 梯度下降法（本章节作者：张宇豪）

通过上述使用最小二乘法解决线性回归问题的推导可以得知,使用最小二乘法时直接求解最后的 $(X^T X)^{-1}$ 即可，但是当训练的数据集过于庞大时，其求解过程是非常耗时的。但是对于梯度下降法来说的话，其需要进行对数据进行归一化处理，将收集到的数据进行预处理，使数据值较大的书籍映射到一个比较小的范围内，而这一过程加快了求解其最优解的速度，此后则选择学习速率，然后根据多次迭代更新求得的结果，最终得到其最优解。

最小二乘法的公式为:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 \tag{19}$$

此时我们引入一个新的式子，即均方误差

$$\begin{cases} J(\theta) = \frac{1}{2n}\sum_{i=0}^{n}(Y_i - h_\theta(x_i))^2 \\ h_\theta(x_x) = \widehat{Y_i} \end{cases} \tag{20}$$

可以看到两者的区别就在于全面对整个计算结果除以了一下样本个数，就相当于求了一个误差的平均值，从而消除了样本个数对于参数求解的影响，而最后对于这个函数进行求偏导数得到：

$$\frac{\partial(J(\theta))}{\partial(\theta_j)} = \frac{-1}{n}\sum_{i=0}^{n}((h_\theta x_i) - y_i)x_{ij} \tag{21}$$

在求得了偏导函数后，现在就可以利用前面介绍的梯度下降法了，则迭代关系如下：

$$\theta_j = \theta_{j-1} - \alpha\frac{1}{n}\sum_{i=1}^{n}[((h_\theta x_i) - y_i)x_{i0}] \tag{22}$$

其中这个公式中的 n 代表了样本个数，α 代表了学习率也就是迭代过程中的步长。因为 $J(\theta)$ 为凸函数，存在极小值点，所以放 $\frac{\partial J(\theta)}{\partial\theta_j} > 0$ 时，此时的 θ 在最优值的右边，更新使 θ 值减少，而反之则使 θ 值增大。

2.3 正则化（本章节作者：李信）

正则化一般是用于防止模型出现过拟合的现象，过拟合是指学习时模型的参数过多，模型过于复杂。正则化就是在损失函数后加上一个正则化项（惩罚项），其实就是常说的结构风险最小化策略，即经验风险（损失函数）加上正则化。一般模型越复杂，正则化值越大。[4]

在回归问题中，常见的正则化有 L1 正则化和 L2 正则化，无论是 L1 还是 L2 都能够防止过拟合。Ridge 回归在线性回归的损失函数中增加了一个 L2 正则化项，而 Lasso 回归则是在线性回归的式子后增加了一个 L1 正则化项。

L1 正则化通过让原目标函数加上了所有特征系数绝对值的和来实现正则化，而 L2 正则化通过让原目标函数加上了所有特征系数的平方和来实现正则化。两者都是通过加上一个和项来限制参数大小。[5] 其中对于添加了正则化项的目标函数的求解，仍然是对添加了正则化项是损失函数进行求导。

2.3.1 线性回归的 L2 正则化-Ridge 岭回归

最小二乘法的得到的损失函数为

$$LossFunction\ L(W) = \sum_{i=1}^{N}\left\|W^T x_i - y_i\right\|^2 \tag{23}$$

可以求得最优解：

$$\hat{W} = (X^T X)^{-1}X^T Y \tag{24}$$

其中 $X_{N\times p}$ 为 N 个样本，p 个特征，$x_i \in \mathbb{R}^p$

通常 $N \gg p$ 才好，但实际问题中可能出现数据样本少，或者数据的特征过多，使得 $N \gg p$ 不满足，此时 $X^T X$ 将不可逆，导致不能求出 $\hat{W}$ 的解析解，同时这种少量样本去学习多个特征的情况也很容易导致过拟合，所以引入了正则化去解决这些问题

正则化框架为：

$$L(W) + \lambda P(W) \tag{25}$$

其中 L 为 loss Function，P 为 penalty（惩罚函数）

目标是得到 $\arg\max_W[L(W)+\lambda P(W)]$ 于是引入 L2 范数正则化（Ridge Regression，岭回归）后的代价函数为：

$$
\begin{aligned}
J(W) &= \sum_{i=1}^{N} ||W^T x_i - y_i||^2 + \lambda W^T W && (26)\\
&= (W^T X^T - y^T)(XW - Y) + \lambda W^T W && (27)\\
&= W^T X^T XW - Y^T XW - Y^T XW - W^T X^T Y + \lambda W^T W && (28)\\
&= W^T X^T XW - 2W^T X^T Y + \lambda W^T W && (29)\\
&= W^T (X^T X + \lambda I) W - 2W^T X^T Y && (30)
\end{aligned}
$$

令 $\frac{\partial J(W)}{\partial W} = 2(X^T X + \lambda I)W - 2X^T Y = 0$ 可得 $\hat{W} = (X^T X + \lambda I)^{-1} X^T Y$ 其中 $X^T X$ 为半正定，当加上 λI 后必然正定，即保证了可逆。从数学角度上看，使得其可逆；从直观角度来看，抑制了过拟合的可能性 [6]

再从贝叶斯角度看待来看岭回归，使用 MAP（最大后验估计）计算最优参数，回归结果为 $f(W) = W^T X$, 预先所给数据：

$$y = f(W) + \epsilon = W^T X + \epsilon \tag{31}$$

这里 ϵ 为噪声 ($\epsilon \sim N(0, \sigma^2)$), 并且 $y|X, W \sim N(W^T X, \sigma^2)$

接下来使用 MAP 进行计算，从 MAP 的角度来看，参数必然服从某个分布，故假设 $W \sim N(0, \sigma^2)$。因此

$$\hat{W} = \arg\max_W p(W \mid y) \tag{32}$$

首先根据条件概率可得

$$p(W \mid y) = \frac{p(y \mid W) \cdot p(W)}{p(y)} \tag{33}$$

由于已知 y|X,W 和 N 的分布，因此可得：

$$p(y \mid W) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{\left(y - W^T X\right)^2}{2\sigma^2} \right\} \tag{34}$$

$$p(W) = \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left\{ -\frac{\|W\|^2}{2\sigma_0^2} \right\} \tag{35}$$

可以求得：

$$
\begin{aligned}
\hat{W} &= \arg\max_W \frac{p(y \mid W)p(W)}{p(y)} \\
&= \arg\max_W p(y \mid W)p(W) \\
&= \arg\max_W \log\{p(y \mid W)p(W)\} \\
&= \arg\max_W \log\left\{\frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{\left(y-W^TX\right)^2}{2\sigma^2}\right\}\frac{1}{\sqrt{2\pi}\sigma_0}\exp\left\{-\frac{\|W\|^2}{2\sigma_0^2}\right\}\right\} \\
&= \arg\max_W \log\left(\frac{1}{\sqrt{2\pi}\sigma}\frac{1}{\sqrt{2\pi}\sigma_0}\right) - \frac{\left(y-W^TX\right)^2}{2\sigma^2} - \frac{\|W\|^2}{2\sigma_0^2} \\
&= \arg\max_W -\frac{\left(y-W^TX\right)^2}{2\sigma^2} - \frac{\|W\|^2}{2\sigma_0^2} \\
&= \arg\min_W \frac{\left(y-W^TX\right)^2}{2\sigma^2} + \frac{\|W\|^2}{2\sigma_0^2} \\
&= \arg\min_W \left(y-W^TX\right)^2 + \frac{\sigma^2}{\sigma_0^2}\|W\|^2 \\
&= \arg\min_W \sum_{i=1}^{N}\left(y_i-W^Tx_i\right)^2 + \frac{\sigma^2}{\sigma_0^2}\|W\|^2
\end{aligned}
$$

观察上式结果，其与加了 Ridge 正则化的 Loss Function 一致：

$$
J(W) = \sum_{i=1}^{N}\left||W^Tx_i - y_i|\right|^2 + \lambda W^TW \tag{36}
$$

其中 $\lambda = \frac{\sigma^2}{\sigma_0^2}$, 可以发现加入了正则项的最小二乘估计与包含服从高斯分布的噪声和先验的 MAP 是等价的。

2.3.2 线性回归的 L1 正则化-Lasso 回归

L2 虽然很优秀，计算出来的系数更稳定，但是多重共线性仍然存在，于是想到利用这种添加惩罚项的办法来解决掉多重共线性的问题，即把 L2 正则化中的惩罚项替换成 L0-norm(系数中非 0 的个数)，这样模型就更简单了，无关的维度系数为 0，即是去除掉该维度，但是 L0-norm 又有很多问题，如函数非连续，很难求解，所以用 L1-norm(系数的绝对值) 来近似地取代 L0-norm。[7]

L1 正则化的推导公式跟 L2 正则化的推导类似，区别主要在添加的惩罚项部分。

引入 L1 范数正则化后的代价函数为：

$$
J(W) = \sum_{i=1}^{N}\left\|W^Tx_i - y_i\right\|^2 + \lambda\left\|W\right\|_1 = J_1(W) + \lambda\left\|W\right\|_1 \tag{37}
$$

令

$$
\frac{\partial J(W)}{\partial W} = \frac{\partial J_1(W)}{\partial W} + sign(w) = 0 \tag{38}
$$

则

$$\frac{\partial J(W)}{\partial W} = \frac{\partial(||y - xW^T|| + \lambda||W||_1)}{\partial W} \tag{39}$$

$$= \frac{\partial(y - xW^T)^T(y - xW_T)}{\partial W} + \frac{\partial\lambda||W||_1}{\partial W} \tag{40}$$

$$= 0 - 2X^Ty + 2X^TXW + \lambda \tag{41}$$

$$= X^TXW - X^Ty + \frac{\lambda I}{2} \tag{42}$$

最后可得

$$X^TXW = X^Ty - \frac{\lambda I}{2} \tag{43}$$

可以从结果中看出，虽然 Lasso 没有从根本上解决多重共线性的问题，但是限制了多重共线性带来的影响。L1 正则化主导稀疏性，会将系数压缩到 0。

2.3.3 总结 LASSO 和 Ridge 回归

相同点：都是在线性回归的基础上，添加了一个惩罚项，让模型更稳定更简单，泛化能力更强，避免过拟合。

不同点：LASSO 是利用 L1-norm 来逼近 L0-norm 的，因为系数可以减小到 0，所以可以做特征选择，但是并不是每个点都是可导的，所以计算起来可能会并没有 L2-norm 方便；Ridge 是利用 L2-norm 来使系数不那么大，同时方便计算，但是不会使系数减小到 0，所以不能做特征选择，可解释性方面也没有 LASSO 高。[8]

为了总结 Ridge 回归和 Lasso 回归两者的优点，则有了第三种回归 ElasticNet 回归，其在损失函数后添加了两个正则化系数即 L1 正则化项和 L2 正则化项，而每项正则化项的前均有一个正则化系数，由此其既能够单独表示 Lasso 回归与 Ridge 回归，也能够综合两者对损失函数的影响效果得到效果更好的正则化项，诶表达式为:

$$\min_{\theta} \frac{1}{2n}[\sum_{i=0}^{n}(\theta^Tx_i) - y_i^2 + \lambda_1\sum_{j=0}^{m}||\theta|| + \lambda_2\sum_{j=0}^{m}||\theta||_2^2] \tag{44}$$

在具体的实现过程中则可以通过迭代函数寻找到最优的 λ_1 和 λ_2 值，从而提高最后参数的准确性。

2.4 评价参数（本章节作者：李信）

评价指标：

MAE(Mean Absolute Error) 平均绝对误差，其能更好地反映预测值与真实值误差的实际情况，是基础的评估方法，后面的方法一般以此为参照对比优劣。公式为

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}| \tag{45}$$

MSE(Mean Square Error) 平均平方差, 对比 MAE，MSE 可以放大预测偏差较大的值，可以比较不同预测模型的稳定性，应用场景相对多一点。公式为

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{46}$$

RMSE(Root Mean Square Error) 方均根差因为使用的是平均误差，而平均误差对异常点较敏感，如果

回归器对某个点的回归值很不合理，那么它的误差则比较大，从而会对 RMSE 的值有较大影响，即平均值是非鲁棒的。改进：使用误差的分位数来代替，如中位数来代替平均数。假设 100 个数，最大的数再怎么改变，中位数也不会变，因此其对异常点具有鲁棒性。公式为

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{47}$$

MAPE (Mean Absolute Percentage Error, 也叫 mean absolute percentage deviation (MAPD)MAPE 不仅仅考虑预测值与真实值的误差，还考虑了误差与真实值之间的比例，在统计领域是一个预测准确性的衡量指标。公式为

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{48}$$

R Squared，R2 方法是将预测值跟只使用均值的情况下相比，看结果优化了多少。其区间通常在（0,1）之间。

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2} \tag{49}$$

2.5 离散数据处理（本章节作者：张宇豪）

机器学习是从数据中自动分析获得规律（模型），并利用规律对未知数据进行预测。其中数据由离散型和连续型之分，而本实验中的数据全为离散型数据，但是其中难免分布着一些非数值型是亚数据，而将这些数据合理的转换为我们所需要的数值型数据则是关键。

常见的数据归一化方法有：min-max 标准化；z-score 标准化；改进的 zscore 标准化方法，其对原始数据进行线性变换使其结构分布在 [0,1] 区间中或者进行标准差标准化等变化使得变换后的结果满足某种概率分布，从而是的最后解决数据集难以拟合计算复杂等问题。

但在本实验中并没有大量数值很大的数据，主要任务则是将某类中的几个亚数据转化为有规律的数值型数据，即对其进行编码转换。其中比较常见的为 one-hot 编码方法，独热编码即 One-Hot 编码，又称一位有效编码。其方法是使用 N 位状态寄存器来对 n 个状态进行编码，每个状态都有它独立的寄存器位，并且在任意时候，其中只有一位有效。而本实验由于其中的亚数据分类不多，却具有唯一性，所在本实验中对 Onr-hot 编码进行简单的调整，即使用使用二进制进行表示，而非 N 位只有以为有效位，即男性为 0，女性为 1，而非男性为 10，女性为 10。

2.6 线性回归算法在具体问题中的实现（本章节作者：吴航）

本部分内容是线性回归算法在具体问题中的实现，我们通过将线性回归用于一个具体的现实问题，来分析线性回归的优缺点，并基于线性回归的结果评估来研究如何使线性回归算法得到更优的回归结果。

我们在一个公开数据集中实现线性回归算法，该数据集在 kaggle 上公开，具体的内容为通过各个用户的基本情况来预测他们将在保险上的花费。本数据集引入了多个数值属性和非数值属性，分别为

age: 主要收益人的年龄

sex: 购买保险人的性别分为 male 与 female

bmi: 身体健康指数

children: 医疗保险覆盖的子女人数/受抚养人人数

smoker: 是否抽烟（yes，no）

region: 购买人来自哪里（northeast, southeast, southwest, northwest.）

charges: 在保险上花费了多少钱

我们在本次保险预测的目标是基于 age，sex，bmi，children，smoker，region 等属性，分析最终的花费 charges。

2.6.1 数据集预处理

本数据集合中数据格式如下

age	sex	bmi	children	smoker	region	charges
37	Male	29.83	2	no	northeast	6406.4107
40	Male	29.355	1	no	northwest	6393.60345
21	female	34.87	0	no	southeast	2020.5523
64	Female	39.33	0	no	northeast	14901.5167
25	male	26.22	0	no	northeast	2721.3208

表 1: 保险费用预测数据集初始表 [a]

[a]https://www.kaggle.com/simranjain17/linear-regression-assumptions-code/notebook

由上表分析，我们在本次保险预测的目标, 即因变量为 charges 是基于 age，sex，bmi，children 等多个自变量的。

在本问题中，可以看出数值属性如 age，bmi 与最终的 charges 具有相当明显的线性趋势，具备了使用线性回归算法的先决条件——自变量与因变量之间具有线性趋势。同时，本数据集合中自变量之间基本不存在相关性，或者相关性很低，可以视为各个自变量之间相互独立，同时数据中因变量服从正态分布。基于以上假设，在模型中使用线性回归就具有了意义和可行性。

因此本问题可以归类于多元线性回归问题，同时由于属性中在非数值型属性，所以我们需要针对非数值型数据进行预处理，用数值型数据来替代。非数值型在本文中为分类问题，我们引入了哑数据来处理分类问题。

对于 N 分类问题，通常引入 N-1 个数据来减少误差，所以我们将 sex 属性更改为 1 位数据，即 0 代表 male，1 代表 female，将 smoker 同样处理为 yes=1,no=0，对于 region 的四分类问题，我们引入了 3 位数据，其中 southeast=（region1=0，region2=1，region3=0），southwest=（region1=1，region2=0，region3=0），northeast=（region1=0，region2=0，region3=0），northwest=（region1=0，region2=0，region3=1）001，进行以上处理之后，我们把非数值型参数转换为数值型参数进行处理，避免非数值参数不参与回归过程导致出现误差，虽然我们的操作有可能导致线性回归效果变差，但是基于本问题而言，我们做了预处理之后，误差在可以接受的范围之内，所以我们仍使用线性回归来进行处理。经由处理之后的新表如下

age	sex	bmi	children	smoker	region1	region2	region3	charges
37	0	29.83	2	0	0	0	0	6406.4107
40	0	29.355	1	0	0	0	1	6393.60345
21	1	34.87	0	0	0	1	0	2020.5523
64	1	39.33	0	0	0	0	0	14901.5167
25	0	26.22	0	0	0	0	0	2721.3208

表 2: 保险费用预测数据集预处理表 [a]

[a]https://www.kaggle.com/simranjain17/linear-regression-assumptions-code/notebook，已经过数据处理

2.6.2 实现过程

本文使用了 python 来实现线性回归，使用的系统为 windows10，工具为 pycharm，使用的包有 numpy 和 sklearn 包, 对于以上新的数据表 2，我们进行多元线性回归分析，训练集和测试集合划分为 7：3。在最后，为了衡量预测值与真实值的接近程度，即线性回归的效果，我们引入了 R2 作为标准。采用 R2 作为评

价标准，是由于其存在一个上下限，便于我们进行评估，一般而言，越接近于 1，表明我们的回归性能越好，越接近 0，表明性能越差。在对比过程中，我们也将引入其他参数，这点在 2.6.3 中进行额外说明。

2.6.3 结果分析

基于累次的测试，我们获得的 R2 趋近于 0.75，基本符合了我们的需求，针对最终结果的苹果，我们采用了多种处理办法，如 RMSE,MAE,MSE,R2,MAPE. 具体表如下

dateset	mse	rmse	mape	mae	R2
train	36677300.00	6056.18	0.42	4252.88	0.75
test	36255388.00	6021.24	0.45	4039.65	0.75

表 3: 评估参数对比, 数据由计算得到

MSE（Mean Squared Error），叫做均方误差, 针对本问题，我们需要在真实值-预测值后平方之后求和平均，但是在我们的实现中，mse 出乎我们意料的大，我们认为这是由于我们在针对非数值型数据处理的过程中采用的方法不够合理，或者说我们在处理的过程中，非数值型数据的线性趋势不明，导致产生此类问题，因为基于二维的图形显示，我们清晰的知道数值型数据，age 和 bmi 指数与 charge 的线性趋势非常强，但由于整体的线性回归的构建过程，我们仍然把非数值型参数视为自变量，所以整体的回归效果收到了很大的影响，同样的，基于求差值的其他的指标 RMSE，MAE 也出现了同样的问题，所以，今后在其他线性回归问题的处理中，我认为需要对每一个自变量与因变量关系进行线性趋势判断，对于非数值型参数应该做出更为明确的划分。

对于 MAPE 平均绝对百分比误差（Mean Absolute Percentage Error）指标，范围为 $[0,+\infty)$，MAPE 为 0 表示完美模型，MAPE 大于 1 则表示劣质模型。由本指标可以看出，我们在本问题采取线性回归的方法有其可行性，虽然非数值问题对回归效果有一定的影响，但是我们的回归结果仍然是具有非常明显的线性趋势，产生的线性模型在预测结果上仍然具有较好的效果。我们要获取更好的 mape 参数，数据处理部分我认为可以进行更进一步的更新，比如去除孤立点等等。

MSE,MAE,RMSE 针对不同的模型会有不同的单位，比如说预测房价那么误差单位就是万元，预测人数可能是 3，4，5，预测身高就可能是 0.1，0.6，没有什么可读性，我们缺少一个具体的概念的对比来验证我们到底产生了多少误差，如本问题中的 MSE 参数，此参数需要小于多少才能证明我们的模型效果很好? 我们需要根据模型来实际进行分析。而 MAPE 参数不具备上限，我们也很难知道我们的模型的效果。所以，我们最终仍然选择采用了 R2 来作为我们的模型评估标准。回归算法的衡量标准就是正确率，而正确率又在 0 ~ 1 之间，最高百分之百, 最低 0, 很直观，而且不同模型评估模式一致。所以，我们可以依据对于其他线性模型的评价来评估我们的模型，最终经过多次的测试，我们的模型 R2 参数为 0.75，这个标准非常接近 1，证明我们的模型线性回归效果不错，如果需求更接近 1 的 R2，可能需要进行额外的数据处理，比如不采用哑数据，而是采用更为精确的方式。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：吴航）

论文构思：本文研究线性回归这一传统方法的原理与步骤，并使用多元线性回归方法针对保险价格预测问题进行了求解，最终以 R2 来评价回归性能的优越性，事实证明，线性回归这一传统的机器学习方法可以广泛用于各种领域，他易于实现，方便快捷，可解释性强，往往可以快速构建一个问题的解决方案。基于一般的线性回归延伸除了多种的线性回归的形式，诸如 Ridge 回归，Logistic 回归，LASSO 回归等等，这些方式通常可以与其他机器学习的方法混用来比对产生的参数的准确性，使得产生的结果更为严谨，线性

回归在多个应用领域中不仅可以作为主要的学习方法，也可以作为一个非常重要的指标用于衡量其他算法的精确性。

但是同样的，我们需要了解到线性回归中存在的一系列问题，首先，由于线性回归基于一系列较为严格的线性趋势假设，所以在多维空间下，线性趋势不明显的情况下，我们常常难于感受其线性规律。比如在本文中，如果一旦引入了非数值参数，线性关系就会变得极为模糊，或者说自变量与因变量之间不存在线性关系。其次，线性回归是基于多维因变量之间相互独立的假设的，但是实际问题中，我们往往不能注意，或者说注意不到因素之间的弱相关，这极有可能影响线性回归的结果，比如地域与人的健康程度的关系，年龄与人健康程度的关系，我们在处理的过程中默认的判断这些关系不相关，因为判断这些关系的弱相关性其实非常麻烦且浪费时间。最后，有些现实问题之中的属性往往有可能并非线性关系，所以用线性回归并不能得到优秀的结果。所以在使用线性回归的场景中，我们往往需要判断基于很多的假设，才能得到更为优秀的线性回归模型。

尽管线性回归拥有一系列的问题，但是其作为一个较为传统的机器学习的算法仍然拥有非常高的学习优先级，基于线性回归的思想和基础之下，延伸出了一系列的新的机器学习算法，突破线性的要求，突破维度的需求，突破参数的数量要求，甚至将线性回归引入集成学习都可以作为一个新的延伸方向。

撰写过程：撰写本文的过程中，我们优先复习了夏静波老师上课中的 ppt 和我们之前写的小论文，基于小论文的原理过程和我们的学习，我们使用线性回归算法对数据集进行了分析，最终完成了本文。

3.2 所参考主要资源（本章节作者：吴航）

本文主要参考资源为夏静波老师上课所讲的 ppt，以及周志华《机器学习》，在完成论文期间，我们还参考了 CSDN 和知乎上部分专栏的内容，以上引用部分已经在参考文献中标出。

3.3 人员分工（本章节作者：吴航）

本次论文撰写过程中，我们将线性回归的内容分为 2 部分，第一部分为对于小论文的总结和扩展，我们研究了线性回归算法的原理和评估参数分析，本部分由李信和张宇豪同学负责，第二部分为线性回归在具体问题中的实现，和对结果的分析，由吴航同学负责。主体部分之外，论文的选题和原理描述部分由吴航和张宇豪合作，吴航同学负责最后的总结，至此，全文撰写完成。

参考文献

[1] 尚轶伦. 回归 (数学术语). https://baike.baidu.com/item/%E5%9B%9E%E5%BD%92/10412815?fr=aladdin, 2012.

[2] 土豆味玉米片. 最小二乘法. https://blog.csdn.net/qq_32864683/article/details/80368135, 2018.

[3] 杨航锋. 最小二乘法 2. https://zhuanlan.zhihu.com/p/36910496, 2021.

[4] Zero 黑羽枫. 机器学习中的正则化. https://www.jianshu.com/p/569efedf6985, 2019.

[5] 山阴少年. L1 与 l2 损失函数和正则化的区别. https://blog.csdn.net/jclian91/article/details/83239781, 2018.

[6] shuhuai008. 岭回归. https://www.bilibili.com/video/BV1hW41167iL?p=3, 2018.

[7] 十三. Lasso 回归. https://zhuanlan.zhihu.com/p/62457875, 2019.

[8] Joker_sir5. 岭回归和 lasso 回归的不同点. https://blog.csdn.net/Joker_sir5/article/details/82756089, 2018.

4 附录

4.1 代码来源（本章节作者：吴航）

```
1 代码来源：本文具体代码见github（https://github.com/xingzwh/mlhomework）
```

2.3 《Logisitic 回归在检测基因相互作用方向的应用》——段馨雅，张晓敏，张佳敏

摘要：逻辑回归主要用来解决分类问题，最原始的逻辑回归用来解决二分类问题, 在此基础. 上进行算法的修改，可以得到面向于解决多分类问题的逻辑回归方法。” Logistic 回归模型用于研究预测变量对分类结果的影响，通常结果是二元的，例如是否存在疾病 (例如，非霍奇金淋巴瘤)，在这种情况下，该模型称为二元 Logistic 模型。当有多个预测变量时 (例如，风险因素和治疗) 该模型被称为多变量或多变量逻辑回归模型，是医学期刊中最常用的统计模型之一。在本章中，我们将检查简单和多元二元逻辑回归模型并提出相关问题，包括交互作用、分类预测变量、连续预测变量和拟合优度。我们在文章里简单介绍了 Logistic 回归在检测基因相互作用方面的影响，我们挑选了几个不同的方面介绍的相应的应用实例。最后我们提出了 Logistic 回归在基因检测中存在的一些不足，以及主成分分析对 Logistic 回归的改进。

Logisitic 回归在检测基因相互作用研究方向的应用

张晓敏 [1], 张佳敏 [2], 段馨雅 [3]

[1]Hubei Key Lab of Agricultural Bioinformatics, College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

逻辑回归主要用来解决分类问题，最原始的逻辑回归用来解决二分类问题, 在此基础. 上进行算法的修改，可以得到面向于解决多分类问题的逻辑回归方法。" Logistic 回归模型用于研究预测变量对分类结果的影响，通常结果是二元的，例如是否存在疾病（例如，非霍奇金淋巴瘤），在这种情况下，该模型称为二元 Logistic 模型。当有多个预测变量时（例如，风险因素和治疗）该模型被称为多变量或多变量逻辑回归模型，是医学期刊中最常用的统计模型之一。在本章中，我们将检查简单和多元二元逻辑回归模型并提出相关问题，包括交互作用、分类预测变量、连续预测变量和拟合优度。我们在文章里简单介绍了 Logistic 回归在检测基因相互作用方面的影响，我们挑选了几个不同的方面介绍的相应的应用实例。最后我们提出了 Logistic 回归在基因检测中存在的一些不足，以及主成分分析对 Logistic 回归的改进。

关键词: Logistic 回归, 基因相互作用

1 概况

1.1 选题说明（本章节作者：张晓敏）

目的：大多数的人类疾病都是复杂的，这意味着它们通常由多种因素引起，包括多个基因的主效应、基因-基因（G×G）相互作用、基因-环境（G×E）相互作用和高阶相互作用。这种相互作用已经被证明存在于多种复杂的疾病中，例如乳腺癌和阿尔兹海默症等。逻辑回归在研究这种相互作用上应用十分广泛，因为他们可以测量相关性、预测结果和控制混杂变量效应。文章的目的在于了解逻辑回归的原理，熟悉现有的使用逻辑回归来检测基因间相互作用实例，了解基因间互作面临的局限性以及可能的改进方法。

立意：逻辑回归是一种通过量化每个自变量的独特贡献来分析一组自变量对二元结果的影响的有效方法。进行逻辑回归时的重要考虑因素包括选择自变量、确保满足相关假设以及选择适当的模型构建策略。对于自变量的选择，应以公认的理论、先前的经验调查、临床考虑和单变量统计分析等因素为指导，并确认应考虑的潜在混杂变量。我们调查了一些逻辑回归在基因相互作用中的应用，总结了逻辑回归在检测基因相互作用方面的优越性和局限性。

1.2 该算法基本原理（本章节作者：张晓敏）

根据一个人的研究目标和变量格式，有不同类型的回归，其中线性回归是最常用的一种。线性回归分析连续结果（即可以有意义地加、减、乘和除的结果，如体重），并假设结果与自变量之间的关系遵循一条直线（例如，随着消耗的卡路里增加，体重增加增加）。为了评估单个自变量对连续结果的影响（例如，消耗的卡路里对体重增加的贡献），可以进行简单的线性回归。然而，通常更希望同时确定多个因素的影响（例如，消耗的卡路里数量、每周锻炼的天数、和年龄对体重增加的影响），因为在控制其他变量的影响后，人们可以看到每个变量的独特贡献。在这种情况下，多元线性回归是正确的选择。

逻辑回归主要用来解决分类问题，最原始的逻辑回归用来解决二分类问题 [2], 在此基础. 上进行算法的修改，可以得到面向于解决多分类问题的逻辑回归方法。本文以二分类问题为例介绍逻辑回归的相关内容。逻辑回归的思想是: 二分类问题的输出 y 并不是一-定范围内连续的值，其输出只能为 0 或者 1 两种。为了将连续的输出转化为二值问题，可以通过非线性函数将连续值转换为离散的二值。逻辑回归方法主要包括以下几个因素: 假设函数、决策边界、代价函数以及参数优化方法。

Logistic 回归的主要用途: 一是寻找危险因素，例如寻找某一疾病的危险因素等，二是预测，如果已经建立了 Logistic 回归模型，则可以根据模型，预测在不同的自变量情况下，发生某病或某种情况的概率有多大。三是判别，实际上跟预测有些类似，也是根据 Logistic 模型，判断某人属于某病或属于某种情况的概率有多大，也就是看一下这个人有多大的可能性是属于某病。

Logistic 回归与多重线性回归实际上有很多相同之处，最大的区别就在于它们的因变量不同，其它的基本都差不多: 线性回归模型的一个局限性是。要求因变量是定量变量（定距变量、定比变量）而不能是定性变量（定序变量、定类变量）。但是在许多实际问题中，经常出现因变量是定性变量（分类变量）的情况，可用于处理分类因变量的统计分析方法有: 判别分析 (Discriminantanalysis)、Probit 分析、Logistic 回归分析和对数线性模型等。Logistic 回归分析根据因变量取值类别不同，又可以分为 Binary Logistic 回归分析（只能取两个值，虚拟值 0 和 1）和 Multinomial Logistic 回归分析（可以取多个值）。

1.3 逻辑回归在基因间相互作用中的应用（本章节作者：张晓敏）

回归技术在医学研究中的应用非常广泛，因为它们可以测量关联、预测结果并控制混杂的变量效应。作为这样一种技术，逻辑回归是一种通过量化每个自变量的独特贡献来分析一组自变量对二元结果的影响的有效且强大的方法。使用反映在 logit 量表中的线性回归组件，逻辑回归迭代地识别出最有可能检测到观察结果的变量的最强线性组合。进行逻辑回归时的重要考虑因素包括选择自变量、确保满足相关假设以及选择合适的模型构建策略。对于自变量选择，应以公认的理论、先前的实证研究、临床考虑和单变量统计分析等因素为指导，并承认应考虑的潜在混杂变量。

逻辑回归目前已经在多种科学问题上得到了应用。首先是在疾病分类的问题上，使用逻辑回归进行分类，应用逻辑回归对独立成分分析后的基因表达数据进行分类。将独立成分分析方法作为一种降维的有效方法引入到肿瘤分类问题中后，解决了基因芯片数据中变量个数 p 远超过样本个数 n 的问题。其次将多变量逻辑回归对人类肺癌样本进行 DNA 提取和 SNP 分型后可以研究环境变量对基因的作用，例如将肺癌患者和健康人的样本进行基因分型后执行多变量逻辑回归模型，可以用来估计环境变量 RAH 与肺癌的潜在关联。逻辑回归也被用在了检测基因上位性的模型上。GenEpi 模型利用逻辑回归可以研究基因的上位性作用，并且在 AD 的预测上有着良好的效果。

2 Logisitic 回归在基因相互作用检测研究方向的应用

2.1 Logisitic 回归核心思路（本章节作者：段馨雅）

2.1.1 Logisitic 回归假设函数

Logsitic 回归的假设函数为 Sigmoid 函数，目前在机器学习中 Sigmoid 函数已不再被广泛使用，但由于其输出区间恒 $\in [0,1]$，故仍常用于分类问题的输出单元。具体表达式如公式（1）所示。

$$g(z) = \frac{1}{1+e^{-z}} \tag{1}$$

在公式 (1) 中，g(z) 为 Sigmoid 函数，z 在逻辑回归中为参数 $W^{\top}$（可训练的权重）和 b（可训练的偏

置）和特征 $x^{(i)}$ 的线性关系表达式 $z = W^\top x + b$。故假设函数可进一步写为公式 (2)。

$$h(x) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{2}$$

对于 logisitic 回归分类问题，y 只能为 0 或 1，此时 $h(x) = P(y = 1 \mid x; w, b)$ 的输出表示 y=1 的概率，$h(x) = P(y = 0 \mid x; w, b)$ 的输出表示 y=0 的概率，且 $P(y = 0 \mid x; w, b) + P(y = 1 \mid x; w, b) = 0$。且由于对于二分类问题，y 只能为 0 或 1，故当 $h(x) \geq 0.5$ 时，将其标记为正类，即 $y = 1$；$h(x) < 0.5$ 时，将其标记为负类，即 $y = 0$。进而有：$W^{\mathrm{T}} x + b > 0$ 时, $y = 1, W^{\mathrm{T}} x + b < 0$ 时, $y = 0$。

2.1.2 Logisitic 回归代价函数

已知训练集中共 $\left\{\left(x^{(1)}, y^{(1)}, \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$ m 个样本，y 取值为 0 或 1，假设函数为 $h(x) = \frac{1}{1+e^{-(w^T x+b)}}$，线性回归中，单个样本的 loss 为公式（3）所示。

$$\mathrm{cost}(h(x), y) = \frac{1}{2}(h(x) - y)^2 \tag{3}$$

整体样本的 loss 为公式（4）。

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{cost}\left(h_{w,b}\left(x^{(i)}\right), y\right) \tag{4}$$

但在 Logisitic 回归中，$h(x) = \frac{1}{1+e^{-(w^T x+b)}}$，若代入 loss 公式，该代价函数为非凸函数，对其进行最小化时，易陷入局部最优值，因此 Logisitic 回归代价函数改写为公式 (5)。（虽然在三维空间中，局部最优的影响图形观测较为明显，但在 n 维空间中，在每一个维度都为全局最优并不现实，因此在实际问题中的 n 维空间中，局部最优的问题并不会总是产生极其巨大的影响）。

$$\mathrm{cost}\left(h_{w,b}(x), y\right) = \begin{cases} -\log\left(h_{w,b}(x)\right), & \text{if } y = 1 \\ -\log\left(1 - h_{w,b}(x)\right), & \text{if } y = 0 \end{cases} \tag{5}$$

如公式（5）所示，当 y=1, 且 h(x)=1 时，cost=0，假设与标签一致；当 h(x)=0 时，cost→∞，即标签与假设不一致时，代价函数趋近于无穷大。当 y=0，h(x)=0 时，cost=0，假设与标签一致；当取 h(x)=1 时，cost→∞，即标签与假设不一致时，代价函数趋近于无穷大。

2.1.3 Loss 最小化

使用梯度下降法对 Loss function 进行最小化，将 Loss 函数整合为一个公式，使用交叉熵函数对 Logisitic 回归计算 Loss 函数。单个样本 Loss 为公式（6）。

$$\mathrm{cost}(h(x), y) = -y \log(h(x)) - (1 - y) \log(1 - h(x)) \tag{6}$$

全体样本 Loss 为单个样本累加，如公式（7）。

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)\right] \tag{7}$$

此时仍满足 y=1 时，h(x)=1,cos=0；y=0 时，h(x)=0,cost=0。使用梯度下降法对 loss 进行最小化，将

参数初始化为 0, 对权重矩阵 w 迭代如公式（8），对偏置 b 进行迭代如公式（9），α 为学习率。

$$w_j = w_j - \alpha\partial\frac{J(w,b)}{\partial w_j} \tag{8}$$

$$b = b - \alpha\partial\frac{J(w,b)}{\partial b} \tag{9}$$

对 J(W,b) 的每个权重和偏置求偏导，迭代公式可记为公式（10），(11)。

$$w_j = w_j - \alpha\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} \tag{10}$$

$$b = b - \alpha\sum_{i=1}^{m}\left(h\left(x^{(i)}\right) - y^{(i)}\right)x_0^{(i)} \tag{11}$$

为使矩阵维度统一便于计算，引入 x_0==0 与偏置单元 b 对应。通过对参数的更新使得 Loss 函数最小化，得到每个参数的最优值。

2.2 Logisitic 回归交叉熵 loss 函数（本章节作者：段馨雅）

2.2.1 信息熵思路

信息量的定义为公式（12)。

$$f(x) := -\log_2 p(x) \tag{12}$$

某个系统的熵即为对这个系统求信息量的期望，即公式（13)，m 为该系统所有可能的事件数量。

$$\begin{aligned} H(P) :&= E\left(P_f\right) \\ &= \sum_{i=1}^{m} P_i\left(-\log_2 P_i\right) \\ &= -\sum_{i=1}^{m} P_i \log_2 P_i \end{aligned} \tag{13}$$

假设此时有两个概率系统 P 和 Q，使用 KL 散度衡量两个概率系统之间的差异程度，其中 $f_Q\left(q_i\right)$ 为系统 Q 的信息量，$f_P\left(p_i\right)$ 为系统 P 的信息量，KL 散度定义为公式（14)。

$$\begin{aligned} &D_{kL}(P\|Q) \\ &:= \textstyle\sum_{i=1}^{m} p_1\left(f_Q q_i\right) - f_p\left(P_i\right)) \\ &= \textstyle\sum_{i=1}^{m} p_i\left(\left(-\log_2 q_i\right) - \left(-\log_2 p_i\right)\right) \\ &= \textstyle\sum_{i=1}^{m} p_i\left(-\log_2 q_i\right) - \sum_{i=1}^{m} p_i\left(-\log_2 p_i\right) \end{aligned} \tag{14}$$

以系统 P 为基准，$\sum_{i=1}^{m} p_i\left(-\log_2 p_i\right)$ 即为 P 系统的熵，恒定不变，$\sum_{i=1}^{m} p_i\left(-\log_2 q_i\right)$ 即为交叉熵，交叉熵记为公式（15)。

$$H(P,Q) = \sum_{i=1}^{m} p_i\left(-\log_2 q_i\right) \tag{15}$$

根据吉布斯不等式，KL 散度恒大于等于零，交叉熵值恒大于等于 P 系统的熵值，因此希望两个系统越接近，即 KL 散度越小越好，则交叉熵值越小越好，因此可以取交叉熵函数为 loss 函数。其中 m 为样本量，p_i 为样本标签 y，y=1 时为正样本，q_i 为模型输出 $h\left(x^{(i)}\right)$；y=0 时为正样本，p_i 为 1-y，q_i 为 $1-h\left(x^{(i)}\right)$，

即为公式（7）。

2.2.2 极大似然估计思路

假设某个样本分类为 1 的概率记为 $\hat{y}$，则 $\hat{y} = P(y = 1 \mid x)$，且 $\hat{y} = P(y = 1 \mid x) = 1 - P(y = 0 \mid x)$，进一步合并则有公式（16）。

$$P(y \mid x) = \hat{y}^{y} \cdot (1 - \hat{y})^{1-y} \tag{16}$$

当 y=0，$P(y \mid x) = 1 - \hat{y}$, 当 y=1，$P(y \mid x) = \hat{y}$。

现已知训练集 $\left\{\left(x^{(1)}, y^{(1)}, \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}\right.$，此时假设样本之间独立且服从同分布，则有公式（17）。

$$P(Y \mid X) = \prod_{i=1}^{m} \hat{y}_i^{y_i} \cdot (1 - \hat{y}_i)^{1-y_i} \tag{17}$$

两边同时取 log，得到公式（18）。

$$\log P(Y \mid X) = \sum_{i=1}^{m} \left[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)\right] \tag{18}$$

此时希望输入被分配到某一类别的确定性越稳定,则希望 $P(Y \mid X)$ 越大,$\sum_{i=1}^{m} \left[y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i)\right]$ 越大越好，则 loss 函数即取为上述公式（7）。

2.3 L_2 正则化 Logisitic 回归（本章节作者：段馨雅）

2.3.1 L2 正则化 Logisitic 回归的 Loss 函数形式

该应用实例中由于基因相互作用的参数过多，为防止过拟合，使用了 L_2 正则化与逻辑回归相结合。L_2 正则化 Logisitic 回归假设函数与 Logisitic 回归相同，但在 Loss 函数最后加入权重 W 的 L_2 范数的平方作为惩罚项，以防止过拟合。原 Loss 函数为（19）所示：

$$J(w, b) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log \left(1 - h\left(x^{(i)}\right)\right)\right] \tag{19}$$

加入 W 的二范数的平方作为惩罚项，一般不对偏置项进行惩罚，Loss 函数如公式（20）所示：

$$J(w, b) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log \left(1 - h\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m} \sum_{j=1}^{n} w_j^2 \tag{20}$$

此时梯度下降迭代公式如公式（21）、（22）所示。

$$w_j = w_j - \alpha \partial \frac{J(w, b)}{\partial w_j} \tag{21}$$

$$b = b - \alpha \partial \frac{J(w, b)}{\partial b} \tag{22}$$

此时梯度下降迭代公式书写形式仍与 Logisitic 回归相同，但由于 Loss 函数已发生变化，故 Loss 对 W 的偏导数已发生变化，迭代公式可进一步书写为公式（23）、（24）。

$$\mathrm{b} = b - \alpha \sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)}\right) x_0^{(i)} \tag{23}$$

$$w_j = w_j - \alpha \left[\sum_{i=1}^{m} \left(h\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right] \tag{24}$$

由于通常不对偏置项 b 加惩罚项，故 L_2 正则 Logisitic 回归 b 的梯度下降迭代式与 Logisitic 回归相同，为公式（16），公式（17）为权重 W 的迭代公式。

2.4 Logisitic 回归优缺点（本章节作者：段馨雅）

Logisitic 回归作为一种经典的分类模型，目前已被广泛的应用，它得益于其相对简单的模型，具有如下优点：

（1）模型简单且易于理解，相对其他模型（如神经网络）来说具有较强的可解释性，因为从每个特征对应的权重值就可以直观的看到不同的特征对样本最终结果的影响的大小。

（2）训练速度较快。

（3）对样本的数量没有过多的要求，即可以不依赖于极其庞大的样本量。

（4）内存占用资源较少，只需存储 m*n 的二维矩阵，分别对应样本量和特征值。

而同时亦因为其相对简单的拟合模型，Logisitic 回归也有一些弊端：

（1）因为模型形式非常接近线性回归，很难拟合非线性的问题，对于非线性问题表现很差。

（2）准确率并不高，因为其模型形式非常简单。

但总体来说，Logisitic 回归仍然是应用非常广泛。非常经典的分类模型。作为一个优秀的分类模型，其对于噪声的容忍度也较高，同时可以避免过拟合的问题。也可以采用 L_2 正则化来避免过拟合，尤其如果样本特征矩阵中具有多重共线性，引入 L_2 正则化可以得到有效解决。

2.5 相关知识（本章节作者：张佳敏）

2.5.1 多因子降维法–MDR

尽管通过 GWAS 能够识别到一些与疾病存在关联的 SNP 位点，然而很多复杂疾病的遗传因素并不能被 GWAS 分析的结果完全解释，这一现象称之为遗传力缺失 missing heritability。造成遗传力缺失的原因有有很多种，其中有两种因素被认为是最可能和这一现象相关的:

(1) 基因和基因之间的相互作用，Gene-Gene interaction

(2) 基因和环境之间的相互作用，Gene-Environment interaction

第一种用 GxG 表示, 第二种 GxE 表示。研究相互作用有两种方式，第一种是基于回归分析的方法，在回归方程中引入自变量间的相互作用；第二种方法是机器学习，主流的方法是多因子降维法 multifactor dimensionality reduction，简称 MDR, 该方法中的因子为具有交互作用的变量，而维度指的是交互作用因子的组合，比如两个交互作用的位点对应的基因型的组合。

MDR 多因子降维法是逻辑回归的一种补充，可以有效进行基因和基因，基因和环境因素之间的相互作用分析，核心算法如下：

第一步，将数据集拆分成训练集和测试集，用于交叉验证，其中训练集占 90 第二步，提取 N 个因子，可以是 SNP 位点，也可以是环境因子，用于后续分析

第三步，对因子之间的组合进行汇总统计，上图中以 Locus3 和 Locus4 两个 SNP 位点为例，共有 9 种基因型的组合，分别统计每种基因型组合对应的 case 和 control 的样本数，左边为 case, 右边为 control

第四步，计算每种组合下 case 与 control 的比值，并根据阈值划分为 high risk 和 low risk 为例，上图中阈值为 1，大于 1 为高危，小于 1 为低危

第五步，计算因子间相互作用的错误率，错误率最小的即为本次分析识别到的最可能的相互作用的因子

第六步，对于上一步提取到的最佳的相互作用因子，用测试数据集进行验证为了降低分析结果的假阳性，会进行多次交叉验证，然后根据置检验计算对应的 p 值。

MDR 的一个强大卖点是，它可以同时检测和表征与疾病相关的多个基因位点。它搜索任何层次的互动，而不考虑主要影响的重要性。因此，它能够检测高阶相互作用，即使潜在的主要影响在统计上是无关紧要的。

2.5.2 二次优化的惩罚函数法

一般来说, 当把优化原理应用于工程问题上时, 其所形成的数学模型往往是含离散变量的混合非线性约束问题。如果我们采用常规的惩罚函数优化法, 通过对连续变量优化求解, 然后再对得到的优化结果进行圆整, 那么得到的往往是一个不够理想的可行结果, 势必对优化构件的质量产生一定的影响。所以对含离散变量的工程问题, 我们可以用离散变量惩罚函数法来解决。因为它构造了一个离散性惩罚项, 得到的优化结果是离散值, 不需要圆整便可直接应用于工程设计中。但我们知道, 离散变量与连续变量不同, 构成的目标函数不能形成一条光滑的曲线, 所以离散性惩罚项的引入使原问题变得高度非线性和非凸性。因此, 初始值的选择对优化结果显得尤为重要。这一问题, 我们可以通过对离散变量惩罚函数法加以改进来解决。

对于求 $minf(X)$

受约束于

$$g_i(X) >= 0, i = 1, 2, ...n$$

$$h_i(X) = 0, i = m + 1, ...p$$

的非线性整数规划问题, 根据改进的惩罚函数法, 可依如下步骤进行优化计算。

(1) 初优化过程

初计算过程就是求解如下的无约束问题:

$$minF(X, r_1^{(k)}) = minf(X) + r_1^{(k)} * \sum_{i=1}^{m} \frac{1}{g_i(X)} + \frac{1}{\sqrt{r_1^{(k)}}} * \sum_{j=m+1}^{p} [h_j(X)]^2$$

这一过程把优化的离散变量和连续变量都当作连续变量来处理，构造罚函数。从一个连续的初始点 $X^{(0)}$ 出发, 进行无约束极小化，得到一个连续化结果 X'^*。这个优化结果位于可行域内, 它只是一个比较合理的，可供参考的初始值。这一过程的完成使得初始值的选取不再是盲目的，而是一个更接近最优点的值，为进一步的寻优作好了准备。

(2) 再优化过程

此过程把变量分为连续变量和离散变量来考虑, 为离散变量构造惩罚项, 并重新构造惩罚函数:

$$min\phi(X, r_1^{(k)}, r_2^{(k)}) = minF(X, r_1^{(k)}) + r_2^{(k)} * E(X^D)$$

离散性惩罚项 $E(X^D)$ 是 X^D 离开可行点 (离散点) 的一种度量, 它具有如下性质:

$$E(x^D) = \begin{cases} = 0(X^D \in R^D, &) \\ > 0(X^D \notin R^D, &) \end{cases}$$

这一计算过程把 X'^* 作为参考值, 重新构造出离散初始值, 求有离散变量惩罚项的罚函数 $\phi(X, r_1^{(k)}, r_2^{(k)})$ 的无约束极小化, 得到一个离散优化结果 X''^*。

(3) 优化结果检验

我们知道, 工程问题不同于数学问题, 我们的最终目的是求得一个满足工程需要的解。在用优化方法求解时, 我们可以用变量的界限值来约定它们的范围。这一界限值一般是由工程人员根据实际经验来确定的。根拆界限值所确定的范围, 得到的优化纪果一般来讲都具有实用的价值, 易达工厂设计人员所接受。

由于包括惩罚项的罚函数的极小值可能不止一个, 在界限值约定的界限范围内, 极小值数目可能有所减少, 但却不能保证在其范围内没有极小值小于所得优化结果的情况。详见图 a。基于以上原因, 为了尽可能

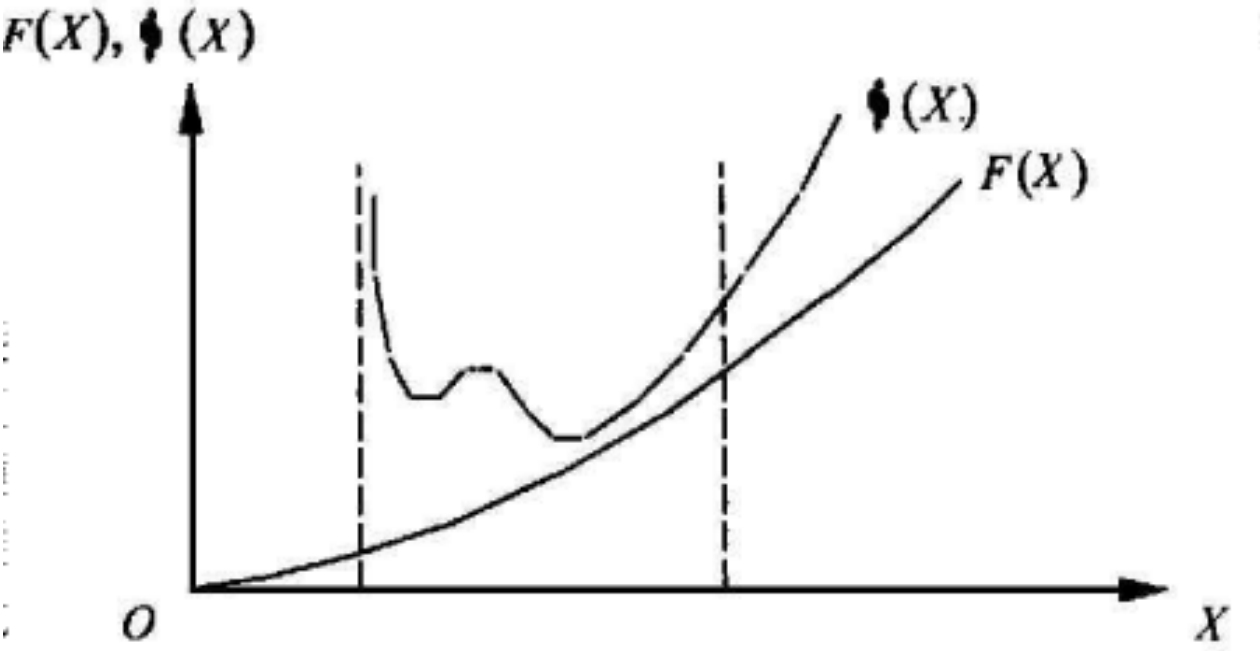


图 1: 离散性惩罚函数变化状况. 图片来源 [2]

的搜索到满足工程要求的最优解, 我们引用网格法来做最终检验。

首先以再优化过程得到的点 $X^{''*}$ 作为网格的中心点F并以扩大一定的范围作为网格的可行域 UN(X), 以适当的网格间数 N1 在 UN(X) 内打网格。最优点 X^* 应是 UN(X) 中诸网格点中惩罚函数值最小者。这就需要逐个检查网格点, 计算其惩罚函数 $\phi(X^{(k)})$ 并与惩罚函数 $\phi(X^{''*})$ 作比较, 若 $\phi(X^{(k)}) < \phi(X^{''*})$，则把 $X^{(k)}$ 作为初始点 $X^{(0)}$, 重新用惩罚函数法寻优，否则把 $X^{''*}$ 作为最优点 X^* 输出。

2.6 具体应用实例（本章节作者：张佳敏）

2.6.1 高血压数据

研究人员使用了 SAPPHIRe (Stanford asia Pacific Program for Hypertension and Insulin Resistance) 项目的数据，将逻辑回归模型与 FlexTree 和 MDR 进行了比较。

FlexTree 是一种树结构学习方法，用于识别与复杂疾病病因相关的基因及其相互作用。目的是构建一个多因素线性组合的劈叉树。从包含所有观测值的根节点开始，通过以下步骤将每个节点递归划分为 2 个子节点:

1) 使用反向剃须选择最优的预测器集来分割特定的节点。对于反向剃须，根据引导的分数形成一个递减的候选子集系列。然后，确定产生最大交叉验证杂质测度的序列中的最佳子集。

2) 进行置换测试, 看看所选预测因子子集与结果之间的线性关系是否足够强。如果是, 执行下一步。如果不是，请停止拆分节点。

3) 使用选定的预测子集计算回归系数 () 和分裂阈值 (C)，以确定二元分裂基于。使用最优评分方法估计 ，选择 C 以最大化得到的节点基尼指数。

MDR 是一种流行的技术，用于检测和表征影响复杂但常见遗传疾病的基因-基因/基因-环境相互作用。MDR 找到了最优相互作用顺序 K 和相应的 K 因素，这些因素在决定疾病状态方面是显著的。算法如下:

1) 对于每个 K，进行 10 次交叉验证，以找到 K 因子的最优集合 (如下所述)。

2) 比较不同 K 的预测误差 (左出集) 和一致性 (在 10 倍的最优因素集中选择了多少次)。

3) 选择预测误差估计最小和/或一致性最大的 K。这个 K 是模型的最终大小，所选订单的最优集形成了最佳多因素模型。

蓝宝石项目的目标是检测使个体易患高血压的基因。研究人员使用了一个类似的数据集，表明 FlexTree 方法优于许多竞争方法。该数据集包含 216 名低血压和 364 名高血压中国女性的 21 个不同位点的更年期状态和基因型。受试者的家庭信息也可用; 属于同一家族的样品被纳入所有分析的相同交叉验证折叠中预留性能。在这里应用 5 倍交叉验证来估计错误率，使用惩罚逻辑回归模型与前向逐步变量选择、FlexTree 和 MDR。为了惩罚逻辑回归模型，通过内部交叉验证为每个折叠选择一个复杂性参数。对于 MDR，我们使用内部交叉验证来选择最重要的特性集。研究人员最初对两类样本使用了不相等的损失: 对低血压样本错误分类的代价是对高血压样本错误分类的两倍。我们将惩罚后的 LR、FlexTree 和 MDR 与损失相等或不相等。对于损失不相等的 MDR，我们在对表的单元格进行标记时，使用 “1/(1+2)” 作为分类比例的阈值，而不是通常损失相等的情况下的 “1/2”。结果对比见表 1。与其他损失函数方法相比，惩罚 LR 方法的误分类代价更低。当使用相同的损失时，FlexTree 和 MDR 生成高度不平衡的预测，将大多数样本分配给更大的类。虽然惩罚 LR 的特异性也较低，但并不像其他 2 种方法那么严重。

Method (loss)	Miscost	Sensitivity	Specificity
PLR (unequal)	$141 + 2 \times 85 = 311$	$223/364 = 0.613$	$131/216 = 0.606$
FlexTree (unequal)	$129 + 2 \times 105 = 339$	$235/364 = 0.646$	$111/216 = 0.514$
MDR (unequal)	$122 + 2 \times 114 = 350$	$242/364 = 0.665$	$102/216 = 0.472$
PLR (equal)	$72 + 139 = 211$	$292/364 = 0.802$	$77/216 = 0.356$
FlexTree (equal)	$61 + 163 = 224$	$303/364 = 0.832$	$53/216 = 0.245$
MDR (equal)	$73 + 163 = 236$	$291/364 = 0.799$	$53/216 = 0.245$

图 2: Comparision of prediction performance among different methonds. 图片来源 [2]

该数据集包括 14 个基因位点的基因型和 201 名膀胱癌患者和 214 名对照组的吸烟状况。通过 5 倍交叉验证，比较了惩罚 LR 与 FlexTree 和 MDR 的预测错误率。如表 9 所示，惩罚 LR 比 FlexTree 具有更高的敏感性和特异性，比 MDR 更平衡的类预测。之后通过改变 0 到 1 之间的分类阈值来生成惩罚 LR 的 ROC 曲线 (图 5)。FlexTree 的灵敏度和特异性均低于惩罚 LR; 因此，如果将 LR 的特异性 (灵敏度) 调整为与 FlexTree 相同，惩罚 LR 的灵敏度 (特异性) 将高于 FlexTree。虽然 MDR 的灵敏度高于惩罚 LR，但正方形仍然偏离 ROC 曲线，向左下角移动。此外，惩罚 LR 的灵敏度和特异性更均匀。

2.6.2 膀胱癌数据集

该数据集包括 14 个基因位点的基因型和 201 名膀胱癌患者和 214 名对照组的吸烟状况。通过 5 倍交叉验证，比较了惩罚 LR 与 FlexTree 和 MDR 的预测错误率。如图 2 所示，惩罚 LR 比 FlexTree 具有更高的敏感性和特异性，比 MDR 更平衡的类预测。之后通过改变 0 到 1 之间的分类阈值来生成惩罚 LR 的 ROC 曲线 (图 3)。FlexTree 的灵敏度和特异性均低于惩罚 LR; 因此，如果将 LR 的特异性 (灵敏度) 调整为与 FlexTree 相同，惩罚 LR 的灵敏度 (特异性) 将高于 FlexTree。虽然 MDR 的灵敏度高于惩罚 LR，但

Method	Misclassification error	Sensitivity	Specificity
PLR	$147/415 = 0.354$	$122/201 = 0.607$	$146/214 = 0.682$
FlexTree	$176/415 = 0.424$	$107/201 = 0.532$	$132/214 = 0.617$
MDR	$161/415 = 0.388$	$137/201 = 0.682$	$117/214 = 0.547$

图 3: Comparsion of prediction performance among different methond. 图片来源 [2]

正方形仍然偏离 ROC 曲线，向左下角移动。此外，惩罚 LR 的灵敏度和特异性更均匀。

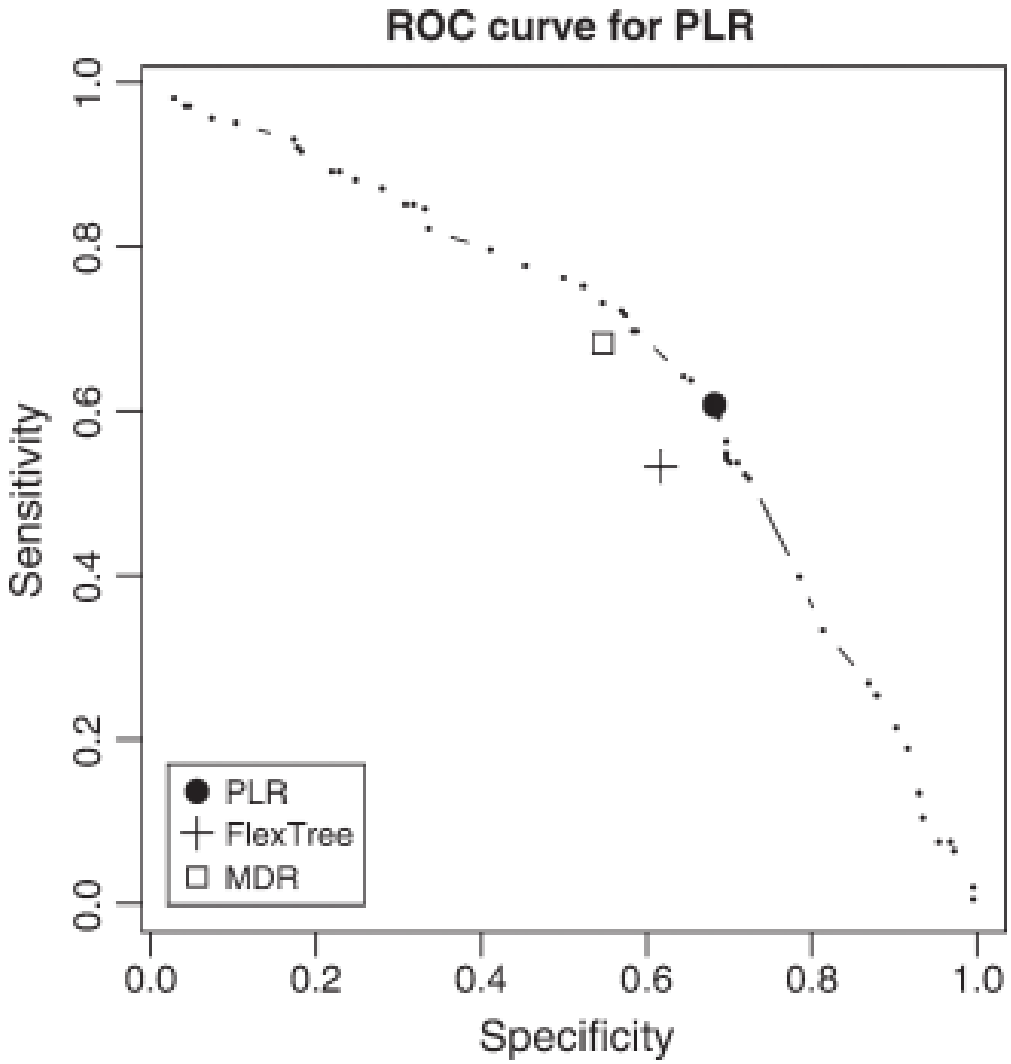


图 4: ROC curve for penalized LR.Penalized LR achieved higher sensetivity and sprcificity than FlexTree. 图片来源 [2]

图 4 是从整个数据集中选择的重要因素及其在 300 次引导运行中的频率。表 11 是指在相对高频率的 300 次正向逐步过程引导运行中选择的因素/因素的交互作用。

图 5 列出了通过引导运行经常选择的因素与因素之间的相互作用。表 10 中的因素构成了表 11 中排名

Factors selected	Frequency	Factors selected	Frequency
Smoke status	296/300	GSTM1	133/300
MPO	187/300	GSTT1	128/300

图 5: Significant factors selected from the whole data set and their frequencies in 300 bootstrap runs. 图片来源 [2]

Factor	Frequency	Interaction of factors	Frequency
Smoke status	296/300	Smoke status × MPO	38/300
MPO	187/300	GSTT1 × NAT2	34/300
GSTM1	133/300	GSTM1 × MnSOD	26/300
GSTT1	128/300	GSTT1 × XRCC1	24/300
NAT2	88/300		
MnSOD	67/300		

图 6: Factors/interactions of factors that were selected in300 bootstrap runs of forward stepwise procedure with relatively high frequencies. 图片来源 [2]

最高的因素子集，为信度提供了证据。从表 11 的后半部分可以看出，即使是计数最大的交互项，其出现频率也不如其他常见主效应项频繁。当我们应用 MDR 时，经常选择二阶相互作用项 MPO× 烟雾状态；然而，根据 bootstrap 结果，LR 方法可以用一个更简单的、可加的模型来解释它们的影响。此外，MDR 无法识别其他潜在的重要特征。研究人员还使用表 11 中各因素的共现矩阵作为不同度量，并应用层次聚类。最紧密的星团之一是 GSTM1 和 NAT2: 它们分别出现在 133 次和 88 次模型中，但只重合了 33 次，这意味着 NAT2 经常被用来取代 GSTM1。

传统分析表明，高频率的因素 (表 4 的第一列) 是明显增加膀胱癌风险的因素之一。通过这种分层建模，MPO、GSTM1 和 MnSOD 获得了较高的优势比，并提高了精度。此外，他们对基因-环境交互作用的分析表明，尽管吸烟状态本身是一个重要因素，但它与其他基因的交互作用并不显著。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：张晓敏）

由于已知许多常见疾病受某些基因型组合的影响，因此对识别有影响的基因及其相互作用结构的方法的需求不断增长。Logistic 回归是评估独立变量对二元结果的贡献的一种有效而有力的方法，但其准确性在很大程度上取决于在满足基本假设的情况下仔细选择变量，以及适当选择模型制定策略并验证结果。此外，不言而喻，构建良好的逻辑回归模型并不是高质量研究的唯一决定因素，该研究开发了与临床相关且客观可测量的假设，实施了适当的研究设计和统计分析计划，准确报告结果和结论都是重要的考虑因素。因此，在设计良好、执行良好的研究背景下密切关注逻辑回归分析参数的读者将对循证急救医学做出最有意义的贡献。LR 是一种标准工具，用于对二元响应数据的影响和交互进行建模。然而，对于这里的 SNP 数据，LR

模型有明显的缺点： 基因及其相互作用可以创建许多参数，并且在数据集相对较小的情况下，会出现过度拟合的问题。 对于许多候选位点，可以将因素关联起来，导致模型进一步退化。 通常定义交互的单元格可以是空的或几乎是空的，这需要特殊的参数化。 随着交互顺序的增加，这些问题变得更加严重。在我们搜集到的文献里面，作者采用了 L1 正则化、独立成分分析等方法来解决上述问题。于是我们决定总结一个完整流程来熟悉逻辑回归这个模型。在 GenEpi 这篇文献中，我们发现作者在解决逻辑回归无法实现高维交互这个问题上，给出了一个解决思路，就是用深度学习这种无特征方法来实现。

3.2 主成分分析对于逻辑回归方法的改进（本章节作者：张晓敏）

Logistic 回归被广泛用于分析个体风险/保护因素与结果之间的关系。但是，如果其中的变量共线，则回归方程将不稳定，其结果难以预测。主成分分析 (PCA) 是一种强大的方法，可用于探索具有多个变量的复杂数据集。PCA 使用数学算法来确定较少数量的新变量，称为主成分 (PC)，它们是原始数据集中变量的线性函数。因此，PCA 会缩小大型数据集的维度，同时尽可能多地保留统计信息。因此，当前研究使用 PCA 有助于确保回归方程的稳定性。我们找到的一篇文献使用主成分分析 (PCA)-逻辑回归模型方法可用于分析多个临床指标相互作用对同时患有甲状腺功能减退症的狼疮性肾炎 (LN) 患者的影响。使用该模型，研究者发现血清肌酐 (SCr)、血尿素氮 (BUN)、血尿酸 (UA)、总蛋白 (TP)、白蛋白 (ALB) 和抗核糖核蛋白 (RNP) 抗体对于这些患者是特别重要的因素。

logistic PCA 和 logistic SVD 方法从概率的角度考虑了二进制数据的数学特征。拟合值，即派生分数和加载矩阵的乘积中的元素，可以解释为生成二进制数据的对数赔率，并且对数赔率可以再次转换为概率。由于在 logistic PCA 中仅自由估计加载矩阵 B 以找到最佳模型，而得分矩阵 A 是固定的，给定负载，logistic PCA 模型不太容易过度拟合。

3.3 所参考主要资源 (本章节作者：张晓敏)

参考文献：

[1]GenEpi: gene-based epistasis discoveryusing machine learning

[2]Park MY, Hastie T. Penalized logistic regression for detecting gene interactions. Biostatistics. 2008 Jan;9(1):30-50. doi:10.1093/biostatistics/kxm010. Epub 2007 Apr 11. PMID: 17429103.34

代码链接：https://github.com/Chester75321/GenEpi

3

进阶算法

这个部分挑选的是 *SVM*，矩阵分解，和 *PCA* 的三篇。

– Jingbo Xia

3.1 《SVM 及其算法实现研究》——龙征武，毕思腾，杨咏琨

支持向量机是一种以监督学习方式对数据进行分类的经典模型。本文阐述了 SVM 的基本原理，推导了硬间隔 SVM 和软间隔 SVM 的基本模型，根据其原模型求解的复杂性又延伸提出了原问题的对偶问题的求解模型。为解决线性不可分问题，引入核函数，并总结了常用核函数。针对对对偶问题的求解，本文描述了 SMO 算法的过程，提出了算法的基本步骤，描述了以两个变量为起点的二次规划求解以及变量选择问题。对于 SVM 求解多分类问题，本文先介绍了常用的二分类扩展到多分类的策略，然后使用了 OvR(一对其他) 和 OvO(一对一) 策略将 SVM 扩展到多分类 SVM。最后，简单概述了本文编写代码时使用的一些优化加速策略。本文基于 python 的 numpy 库对 SVM 算法进行了实现，并和 sklearn 中的 SVM 进行了对比。

SVM 及其算法实现研究

龙征武 [1], 毕思腾 [2]，杨咏琨 [3]

[1]College of Life Science and Technology of Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[2]College of Plant Science and Technology of Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[3] Huazhong Agricultural University College of informatics, Wuhan, Hubei Province, P.R. China

摘要

支持向量机是一种以监督学习方式对数据进行分类的经典模型。本文阐述了 SVM 的基本原理，推导了硬间隔 SVM 和软间隔 SVM 的基本模型，根据其原模型求解的复杂性又延伸提出了原问题的对偶问题的求解模型。为解决线性不可分问题，引入核函数，并总结了常用核函数。针对对对偶问题的求解，本文描述了 SMO 算法的过程，提出了算法的基本步骤，描述了以两个变量为起点的二次规划求解以及变量选择问题。对于 SVM 求解多分类问题，本文先介绍了常用的二分类扩展到多分类的策略，然后使用了 OvR（一对其他）和 OvO（一对一）策略将 SVM 扩展到多分类 SVM。最后，简单概述了本文编写代码时使用的一些优化加速策略。本文基于 python 的 numpy 库对 SVM 算法进行了实现，并和 sklearn 中的 SVM 进行了对比。

关键词: SVM, 对偶问题，SMO 算法

1 概况

1.1 选题说明（本章节作者：毕思腾）

支持向量机 (Support Vector Machine, SVM) 作为一种非常经典且高效的分类模型，主要用于解决模式识别领域中数据分类问题，属于有监督学习的一种。虽然当下深度学习的发展迅猛，但学习 SVM 的基本原理，借鉴模型推导过程中的简化思想，仍然具有其现实意义。本文将通过计算机利用序列最小化（sequence minimal optimization，SMO）优化算法对 SVM 进行算法实现，并期望在此过程中加深对 SVM 基本原理的理解，掌握算法实现的基本流程，以及评估 SVM 模型，为今后选择和使用 SVM 算法奠定基础。

1.2 该算法基本原理（本章节作者：毕思腾）

用计算机求解 SVM 的问题本质是求解一个二次规划问题，而二次规划问题的通常求解方法需要很大的开销，目前主流软件包（LIVSVM，scikit-learn）的求解方法是 SMO 算法（Sequential Minimal Optimization）。基于以上，本文重点关注的是将 SVM 求解问题转换为对偶问题，然后使用 SMO 算法对求解此对偶问题。

1.3 SVM 的分类与使用场景（本章节作者：毕思腾）

1.3.1 线性可分 SVM

线性可分模型又称为硬间隔支持向量机，是 SVM 模型中最简单的一类模型，以二维空间为例，可以简单的理解为点到分类直线之间的距离，假设分类直线过原点，那么只要直线两侧点各自到直线的距离最大且总和最大时，这条之间就是最优分类直线。当然直线可以不经过原点，此处提出的是最简单的一种状态。因此，原问题就转化成为一个约束优化问题，可以被直接求解，此为线性可分的 SVM 模型。

1.3.2 线性 SVM

线性 SVM 又称为软间隔支持向量机，与线性可分 SVM 相比，其区别在于线性 SVM 虽然具有线性可分的潜质，但需要付出一些“代价”才能完成。在现实问题中，并不是所有的数据可以被完美的分类，如果过分追求两类元素的完全分类，则模型会受到个别值的影响，出现一种“过拟合”的状态，此时我们向 SVM 中引入误差的概念，也就是“松弛变量”，通过加入松弛变量，在原距离函数中需要加入新的松弛变量带来的误差，这样，最终的优化目标函数变成了两个部分组成：距离函数和松弛变量误差。这两个部分的重要程度并不是相等的，而是需要依据具体问题而定的，因此，我们加入权重参数 C，将其与目标函数中的松弛变量误差相乘，这样，就可以通过调整 C 来对二者的系数进行调和。如果我们能够容忍噪声，那就把 C 调小，让他的权重降下来，从而变得不重要；反之，我们需要很严格的噪声小的模型，则将 C 调大一点，权重提升上去，变得更加重要。通过对参数 C 的调整，可以对模型进行控制。这叫做软间隔最大化，得到的 SVM 称作线性 SVM 或软间隔支持向量机。

1.3.3 非线性 SVM

非线性的 SVM 相较于前两者 SVM 模型，特点是它根本不能在当前维度上被一条直线所分，即使允许个别分类错误的情况。对于大多数的分类问题，使 SVM 刚好在二维空间中线性可分是条件苛刻，很多问题并不能在线性，因此对于输入空间中的非线性分类问题，我们可通过非线性变换（核函数）将它转化为更高维度特征空间中的线性分类问题。在高维特征空间中学习线性支持向量机。

1.4 SVM 模型的优劣性总结（本章节作者：毕思腾）

SVM 可以用于线性、非线性的问题分类，也可用于回归分析，它是基于严格的数学推导，是可以不加修改直接的一种分类器。其泛化错误率低，不容易过拟合，对数据之外的数据点做出很好的分类决策具有良好的学习能力，并且所得到的结果的普适性好。

相较于神经网络结构模型，SVM 的优化问题具有凸优化的特点，即它所寻找到的最优解，会是全局最小值或接近值，不必担心局部最优解现象的出现。非常适用于特征值很多，而样本数很少的小样本的机器学习问题。由于 SVM 的最终决策只由少数支持向量所确定，计算的复杂度取决于支持向量的数目，而不是样本空间的维数，利用内积核函数代替向高维空间的非线性映射，可解决高维问题。

但常规的 SVM 模型并不适合多分类问题，并且 svm 模型本身对样本数量过多的问题进行学习时，速度较慢，易受到不完整向量数据的影响，对松弛变量或惩罚因子等参数和核函数的选择敏感。

2 SVM 算法的实现研究

2.1 SVM 问题的描述（本章节作者：毕思腾）

SVM 问题有样本集 $D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_m,y_m)\}$，$y_i\in\{-1,+1\}$，求解 SVM 问题需在空间 D 找到一个超平面，将不同类别的样本分开，此超平面用如下的方程描述：

$$\boldsymbol{w}^T\boldsymbol{x}+b=0 \tag{1}$$

其中 $\boldsymbol{w}=(w_1;w_2;...;w_d)$ 为法向量，b 为位移项。样本空间中的点 $\boldsymbol{x}$ 到超平面的距离可以表示为：

$$r=\frac{|\boldsymbol{w}^T\boldsymbol{x}+b|}{\|\boldsymbol{w}\|} \tag{2}$$

对于 $(\boldsymbol{x_i},y_i)\in D$, 若 $y_i=+1$，则有 $\boldsymbol{w}^T\boldsymbol{x_i}+b>0$; 若 $y_i=-1$，则有 $\boldsymbol{w}^T\boldsymbol{x_i}+b<0$。令

$$\begin{cases} \boldsymbol{w}^T\boldsymbol{x_i}+b\geq 1, y_i=+1; \\ \boldsymbol{w}^T\boldsymbol{x_i}+b\leq -1, y_i=-1. \end{cases} \tag{3}$$

总结公式(3)，可以得到：

$$f(x)=sign(\boldsymbol{w}^T\boldsymbol{x}+b) \tag{4}$$

用于预测 y，其中 sign 函数是符号函数，输入大于 0 则为 1，反正为-1。

距离超平面最近的样本点会使得式(3)中的不等式的等号成立。这样的样本点被称为“支持向量”，两个异类支持向量 $\boldsymbol{x_m},\boldsymbol{x_n}$ 到超平面的距离之和为：

$$\gamma=\frac{|\boldsymbol{w}^T\boldsymbol{x_m}+b|}{\|\boldsymbol{w}\|}+\frac{|\boldsymbol{w}^T\boldsymbol{x_n}+b|}{\|\boldsymbol{w}\|}=\frac{2}{\|\boldsymbol{w}\|} \tag{5}$$

SVM 需要找到一个最佳平面，这个平面有最大的间隔，则可以描述为 $\max\limits_{\boldsymbol{w},b}\frac{2}{\|\boldsymbol{w}\|}$，显然，这也等价于最小化 $\min\limits_{\boldsymbol{w},b}\|\boldsymbol{w}\|^2$，结合约束条件，可以总结硬间隔 SVM 问题的模型如下：

$$\begin{aligned} \min_{\boldsymbol{w},b}\quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ s.t.\quad & y_i(\boldsymbol{w}^T\boldsymbol{x}+b)\geq 1,\quad i=1,2,\ldots,m. \end{aligned} \tag{6}$$

硬间隔 SVM 试图将所有的样本完全划分开来，但是在实践中很少有样本能被完全划分，并且完全划分也容易导致过拟合。因此需要引入软间隔的方法，允许一部分的样本不满足约束条件，但是这些样本要尽量的少，常引入 hinge-loss 函数可以描述，将公式 (6) 改写如下：

$$\min_{\boldsymbol{w},b}\quad \frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}max(0,y_i(\boldsymbol{w}^T\boldsymbol{x}+b)-1) \tag{7}$$

C 是一个超参数，C>0，将 C 设置为一个足够大的值时，公式(7)退化成了(6)，将 C 设置成一个合理值，则能对优化模型起到一定的约束作用。为了方便将公式(7)转换成二次规划问题，引入松弛变量 $\xi_i\geq 0$ 改写，这些松弛变量可以理解为不满足约束条件的程度：

$$\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ s.t. \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}+b) \geq 1-\xi_i \\ & \xi_i \geq 0, \quad i=1,2,\ldots,m. \end{aligned} \tag{8}$$

2.2 SVM 模型的 KKT 条件及对偶问题（本章节作者：杨咏琨）

KKT 条件是非线性规划有最优解的必要条件。考虑标准约束优化问题：

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & g_j(x)=0, \quad j=1,2,\ldots,m. \\ & h_k(x) \leq 0, \quad k=1,2,\ldots,p. \end{aligned} \tag{9}$$

定义拉格朗日函数：

$$L(\boldsymbol{x},\{\lambda_j\},\{\mu_k\}) = f(\boldsymbol{x}) + \sum_{j=1}^{m}\lambda_j g_j(\boldsymbol{x}) + \sum_{k=1}^{p}\mu_k h_k(\boldsymbol{x}) \tag{10}$$

则 KKT 条件包括：

$$\begin{cases} \nabla_x L = 0 \\ g_j(\boldsymbol{x}) = 0, \quad j=1,2,\ldots,m. \\ h_k(\boldsymbol{x}) \leq 0 \\ \mu_k \geq 0 \\ \mu_k h_k(\boldsymbol{x}) = 0, \quad k=1,2,\ldots,p. \end{cases} \tag{11}$$

上述 5 个公式中，1,2,3 为基本条件，4 是为了要求目标函数的梯度和约束条件的梯度必须反向。

现对条件 5 进行分析。条件 5 表明，若优化问题有解则对于每一个不等式限制条件，其拉格朗日系数等于 0 或者不等式条件取等号。假设不等式限制条件组成的区域为 G，无不等式限制的 $f(\boldsymbol{x})$ 的最优解为点 m，则 m 可能落在 G 的内部，外部，边界上。若 m 落在 G 的内部，则不等式限制条件对结果无影响，拉格朗日系数可以均置为 0 满足条件 5；若 m 落在 G 的边界上，则有若干与该边界条件无关的不等式条件对结果无影响，其拉格朗日系数可以置 0，与该边界条件相关的不等式条件则需要将相应的 $h(\boldsymbol{x})$ 表达式置 0，同意满足条件 5；若 m 落在 G 的外部，为满足不等式条件组，则需要根据目标函数 $f(\boldsymbol{x})$ 的值分布在 G 中找到最佳位置，该位置一定属于上述两种情况之一。综上所述，条件 5 为最优化问题 (9) 有最优解的必要条件。

公式(8)是一个凸二次规划问题，但是求解较为复杂，为了降低计算的复杂度，可以将原问题转换为对偶问题，然后再根据 KKT 求解。

接下来讨论对偶问题，在 SVM 问题背景下，对于(9)和(10)，记(9)为原问题 p^*, 则有:

$$p^* = \min_{\boldsymbol{x}}\max_{\boldsymbol{\lambda},\boldsymbol{\mu}} L(\boldsymbol{x},\{\lambda_j\},\{\mu_k\}) \tag{12}$$

因为将 x 视为常数后，(10)的 L 的最大值取决于后两项，而根据 KKT 条件(11)中的第 2，5 项，优化问题取最优解时 $L=f(x)$，同时 L 也在拉格朗日系数的限制下取到最大值，故有公式(12)，则经过这种变换，有不等式的约束优化问题(9)就转变成没有约束优化的问题，记为原问题 p^*

记 p^* 的对偶问题为 d^*, 其形式为：

$$d^* = \max_{\boldsymbol{\lambda},\boldsymbol{\mu}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \{\lambda_j\}, \{\mu_k\}) \tag{13}$$

即先计算非拉格朗日变量约束下的最小值，再计算拉格朗日变量约束下的最大值。下面将上述两个理论运用到(8)中。首先，引入拉格朗日乘子 $\alpha_i \geq 0, \mu_i \geq 0$, 使用拉格朗日乘子法得到公式(8)的拉格朗日函数：

$$L(w, b, \alpha, \xi, \mu) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1 - \xi_i - y_i(\boldsymbol{w}^T\boldsymbol{x} + b)) - \sum_{i=1}^{m}\mu_i\xi_i \tag{14}$$

此时未知量有平面法向量的参数 $\boldsymbol{w}$ 和 b 以及松弛变量 ξ_i，和两组拉格朗日乘子。对偶问题是拉格朗日函数的极大极小问题，首先对此拉格朗日函数分别计算前三个未知量的偏导为 0：

$$\boldsymbol{w} = \sum_{i=1}^{m}\alpha_i y_i x_i \tag{15}$$

$$0 = \sum_{i=1}^{m}\alpha_i y_i \tag{16}$$

$$C = \alpha_i + \mu_i \tag{17}$$

公式(15)具体步骤如下，其余两个偏导计算相似：

$$\begin{aligned}
\frac{\partial L(w, b, \alpha)}{\partial w} &= \frac{\partial\left[\frac{1}{2}\|w\|^2 + \sum\limits_i \alpha_i(1 - y_i(w^T x_i + b))\right]}{\partial w} \\
&= \frac{\partial\left[\frac{1}{2}w^T w - \sum\limits_i \alpha_i y_i w^T x_i + \sum\limits_i \alpha_i(1 - y_i b)\right]}{\partial w} \\
&= w - \sum_i \alpha_i y_i x_i
\end{aligned}$$

然后将公式(15)-(17)带入拉格朗日函数，在对此函数求拉格朗日乘子参数求极大，即可以求解对偶问题：

$$\begin{aligned}
\max_{\alpha} \quad & \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j \\
s.t. \quad & \textstyle\sum_{i=1}^{m}\alpha_i y_i = 0 \\
& C - \alpha_i - \mu_i = 0 \\
& \alpha_i \geq 0, \mu_i \geq 0, \quad i = 1, 2, \ldots, m.
\end{aligned} \tag{18}$$

(**注意：此处 max 下标只有 α, 因为 $\alpha_i + \mu_i = C$，带入到拉格朗日函数后第二项和第四项合并，μ 项被消去了**)

对偶问题需要求解出的 α_i 需要满足 KKT 条件。结合公式(8)、(11)、(14)和(18)可以总结以下 KKT 条件：

$$\begin{cases} \alpha_i \geq 0, \mu_i \geq 0 \\ y_i f(x_i) - 1 + \xi_i \geq 0 \\ \alpha_i (y_i f(x_i) - 1 + \xi_i) = 0 \\ \xi_i \geq 0, \mu_i \xi_i = 0 \end{cases} \tag{19}$$

其中 $f(x) = \boldsymbol{w}^T \boldsymbol{x} + b$。结合上述 KKT 条件，对于任意样本 (x_i, y_i), 有以下几种情况：

- 若 $\alpha_i = 0$: 则 $y_i f(x_i) \geq 1 - \xi_i$，此时点落在支持向量的外部，对 f(x) 没有影响。
- 若 $0 < \alpha_i < C$: 则 $\mu_i > 0, \xi_i = 0, y_i f(x_i) = 1 - \xi_i$，此时松弛变量未起到作用，点落在边界上，即就是支持向量。
- 若 $\alpha_i = C$:
  - 若 $0 < \xi_i < 1$，此时点落在超平面和正类之间
  - 若 $\xi_i = 1$，此时点落在超平面上
  - 若 $\xi_i > 1$，此时点分类错误，落在超平面的另一侧

当对偶问题求得解 $\alpha_i = \{\alpha_1^*, \alpha_2^*, \ldots, \alpha_n^*\}$（求解算法见下一小节），将 α 带入公式(15)，可以计算原问题的解 w^*:

$$\boldsymbol{w^*} = \sum_i \alpha_i^* y_i * x_i \tag{20}$$

再将上式以及将 1 替换成 y_i^2，可以计算 b^*，此时需要 $\alpha_j > 0$：

$$b^* = y_j - \sum_i^m \alpha_i^* y_i (x_i . x_i) \tag{21}$$

上述 w^*, b^* 就是所求超平面方程的参数。对比原问题公式(8)和对偶问题公式(18)，可以发现对偶问题的未知数只剩下 α，这是一个向量，向量的维度为 m，此时求解问题的规模只和样本数 m 有关了。

将最优解的 $\boldsymbol{w^*}$ 和 b^* 带入式(4)，可以得到新的预测函数：

$$f(x) = sign(\sum_i \alpha_i^* y_i * x_i + b^*) \tag{22}$$

2.3 核函数（本章节作者：毕思腾）

2.3.1 线性不可分问题和核技巧

引入软间隔后，SVM 能有效的求解线性问题。但是现实生活中也有着相当多的非线性问题，非线性问题是指利用非线性模型才能很好地进行分类的问题，引入一个例子：通过观察图(1)的左上图，我们有样本集 $D_{old} = \{(3,0), (2,1), (1,2), (0,3)\}, label_{old} = \{1,2,2,1\}$ 发现无论怎么划分也不能通过一条直线将两类点划分开来，因此自然可以想到通过一条曲线划分，图(1)的右上图通过绘制曲线 $y = \frac{1}{x}$，可以将两类样本轻松划分。另外一种思路是我们将样本映射到更高的维度，在样本引入新的维度:

$$z = \phi(x, y) = xy$$

样本点可以扩展为 $D_{new} = \{(3,0,0), (2,1,2), (1,2,2), (0,3,0)\}, label_{new} = \{1,2,2,1\}$，图(1)的下图显示，发现样本点分别被划分到 $z = 0, z = 2$ 这两个平面上了，很自然就找到了超平面 $z = 1$，即正好处于前两个平面中间的平面。

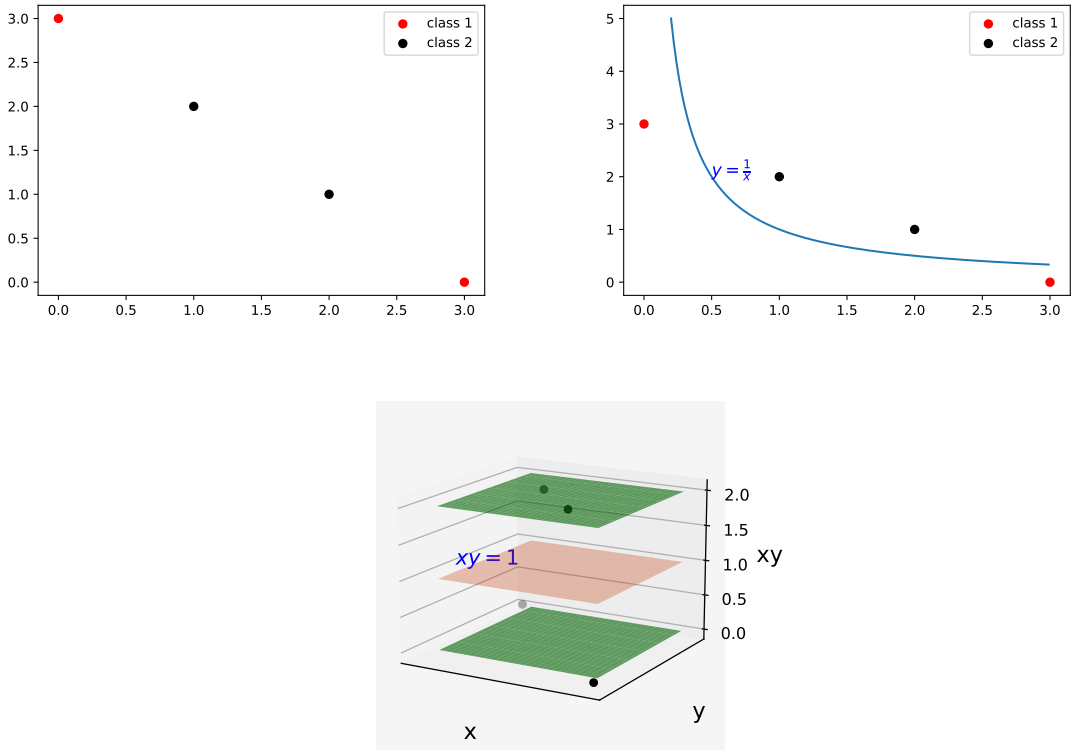


图 1: 核技巧示意图 **（思路参考了视频 [5]，绘图使用 matplotlib，代码见附录）**

这两种方法的本质是一样的，$z = xy = 1$ 实际是高维空间 $y = 1/x$ 的一种表现形式。非线性问题往往不好求解，优化求解过程中一般会将非线性问题转换为线性问题，解线性问题以求解原来的非线性问题，这也是 SVM 中经常使用的——核技巧（kernal trick）。核技巧中原输入空间记作 $\mathcal{X}$，变化后的空间称为特征空间记作 $\mathcal{H}$。如果 $\mathcal{X}$ 存在一个映射到 $\mathcal{H}$：

$$\phi(x) : \mathcal{X} \to \mathcal{H} \tag{23}$$

使得所有的 $x, z \in \mathcal{X}$，函数 $K(x, z)$ 满足条件

$$K(x, z) = \phi(x) \cdot \phi(z) \tag{24}$$

就称 $K(x, z)$ 为核函数。通常求映射空间是困难的，因为特征空间一般是高维的，一些核函数还会将输入空间映射到无穷维，例如高斯核，而直接计算核函数通常是容易的。

2.3.2 核函数在 SVM 中的应用

容易注意式(18)中的项 $\boldsymbol{x}_i^T \boldsymbol{x}_j$，通过核技巧，可以将其替换为 $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$，得到：

$$\max_{\alpha} \quad \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{25}$$

同理带入式(22)，可以得到新的预测函数：

$$f(x) = sign\left(\sum_{i=1}^{m} \alpha_i y_i K(x_i, x_j) + b^*\right) \tag{26}$$

在实际应用中，确定使用什么核函数也是一个问题，不同的核函数在不同的应用场景的效率也是不相同的，通常需要依赖邻域知识选择核函数，并在实践过程中不断优化选择构造适合的核函数。现总结常用的核函数见下表：

函数	公式
线性核	$K(x,z) = x \cdot z$
多项式核	$K(x,z) = (c_1 x \cdot z + c_2)^p$
高斯核	$K(x,z) = exp\left(-\frac{\|x-z\|^2}{2\sigma^2}\right)$
径向基核	$K(x,z) = exp\left(-\gamma\|x-z\|^2\right)$
拉普拉斯核	$K(x,z) = exp\left(-\frac{\|x-z\|}{\sigma}\right)$
sigmoid 核	$K(x,z) = tanh(c_1 x \cdot z + c_2)$

表 1: 常用核函数总结 **（参考了书 [1] 和 [2]）**

2.3.3 核函数测试

本文使用玫瑰线参数方程生成随机数据，然后对数据分别进行线性核、sigmoid 核和 Gauss 核的 SVM 预测，图(2(a))显示了 sklearn.svm 的预测结果，图(2(b))显示了 demo-SVM 的预测结果，对比线性 SVM，发现核函数能有效的解决线性不可分问题。

2.4 SMO 算法（本章节作者：龙征武）

公式(25)是一个凸二次规划问题，此时问题的求解规模核样本规模 m 有关，当 m 很大时，此时求解的效率任然很低下。SMO 算法是一种适合解决此类问题的算法，其基本思路是：先固定大部分参数，转换为只有少量未知参数的规划问题，将转换的问题解决，得到的新解将不断优于上一次的解。SMO 算法的步骤如下：

step1 初始化 $\boldsymbol{\alpha}, iter = 0,$

step2 选取两个优化的参数 $\alpha_i^{(k)}, \alpha_j^{(k)}$，其他的 α 视为定值，求新解 $\alpha_i^{(k+1)}, \alpha_j^{(k+1)}$，求解步骤见下一小结

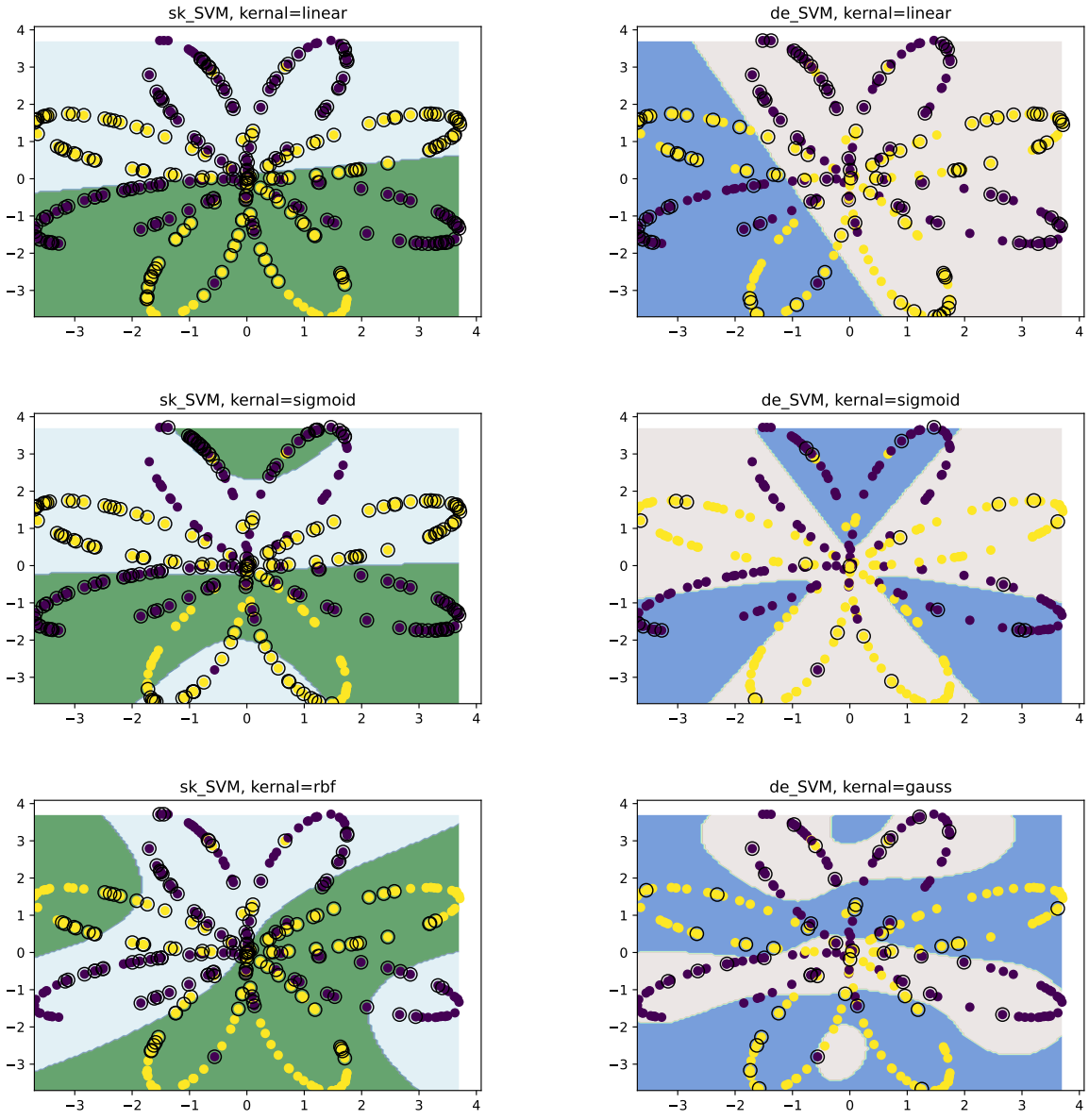
step3 更新阈值 b 和差值 E

step4 以 eps 为精度判断全部的 $\alpha^{(k+1)}$ 是否符合 KKT 条件，符合则输出解，否则转下一步

step5 $iter+ = 1$，判断 $iter < max\_iter$，符合则输出，否则转 step2

2.4.1 两个变量的二次规划求解

SMO 算法的第一步就是选择合适的参数 α。在公式(18)的约束条件约束下，需要一次确定 α_i, α_j 两个参数。不失一般性，假设选择的参数为 α_1, α_2，其余的参数是固定的，由此将公式(25)展开可得：



(a) sklearn.svm

(b) demo_svm

图 2: 不同核函数运行结果对比 （**绘图使用 matplotlib，代码见附录**）

$$\begin{aligned} \max_{\alpha_1,\alpha_2} \quad & \alpha_1+\alpha_2-\frac{1}{2}(K_{11}\alpha_1^2+K_{22}\alpha_2^2)-y_1y_2K_{12}\alpha_1\alpha_2-y_1\alpha_1C_1-y_2\alpha_2C_2+C_3 \\ s.t. \quad & \alpha_1y1+\alpha_2y_2=-\sum_{i=3}^{m}y_i\alpha_i=\varsigma \\ & 0\le\alpha_i\le C, \quad i=1,2 \end{aligned} \tag{27}$$

其中 $K_{i,j}=K(\boldsymbol{x_i},\boldsymbol{x_j})$，$C_1,C_2,C_3$ 是与求解参数无关的变量。在不考虑约束的条件下，对上式优化，将原参数记为 $\alpha_1^{old},\alpha_2^{old}$，新参数记为 $\alpha_1^{new},\alpha_2^{new}$ 可以求出不考虑约束的新 α_2 解为：

$$\alpha_2^{new,unc}=\alpha_2^{old}+\frac{y_2(E_1-E_2)}{\eta} \tag{28}$$

其中 E,η 分别为：

$$E_i=\left(\sum_{j=1}^{m}\alpha_jy_jK(x_j,x_i)+b\right)-y_i \tag{29}$$

$$\eta=K_{11}+K_{22}-2K_{12} \tag{30}$$

然后分析其约束条件，由(27)的约束条件 $\alpha_1y1+\alpha_2y_2=\varsigma$ 和 $0\le\alpha_i\le C$ 可知，α_1,α_2 被限定在一个边长为 C 的正方形盒子中，且 (α_1,α_2) 过一条斜率为-1 或 1 的直线，在此限定下，$\alpha_2^{new,unc}$ 需要满足一个上界 H 和下界 L，当 $y_i=y_j$ 时:

$$L=max(0,\alpha_2^{old}-\alpha_1^{old}),\quad H=min(C,C+\alpha_2^{old}-\alpha_1^{old})$$

而当 $y_i\ne y_j$ 时:

$$L=max(0,\alpha_2^{old}+\alpha_1^{old}-C),\quad H=min(C,\alpha_2^{old}+\alpha_1^{old})$$

然后裁剪 $\alpha_2^{new,unc}$，得到新解为：

$$\alpha_2^{new}=\begin{cases} H, & \alpha_2^{new,unc}>H \\ \alpha_2^{new,unc}, & L\le\alpha_2^{new,unc}\le H \\ L, & \alpha_2^{new,unc}<L \end{cases} \tag{31}$$

由上式和公式(27)可以计算出 α_1^{new}:

$$\alpha_1^{new,unc}=\alpha_1^{old}+y_1y_2(\alpha_2^{old}-\alpha_1^{old}) \tag{32}$$

2.4.2 变量选择

样本点选取的规则也会影响算法的收敛速度，启发式的选择变量实践中能有效的减少迭代次数。选择第一个参数时会选取违反 KKT 条件最严重的点，先遍历整个样本，先找满足 $0<\alpha_i<C$ 条件的点，即作为支持向量的点。其次再寻找其他的样本点，如果没有找到则全部都满足 KKT 条件，此时已经找到了最优解。

选择第二个参数 α_j 的标准是希望此参数变化足够大, 由公式(31)可知 α_2^{new} 依赖于 $|E_1-E_2|$，如果 E_1 为正，选择最小的 E_i 作为 E_2;E_1 为负选择 $argmax(E_i)$ 作为 E_2。上述方法无效时，这时在尝试支持向量上的点。当找不到合适的 α_2 时，重新选择新的 α_1。

在寻找参数的过程中 K_{ii}, E, b 和支持向量都会多次重复用到，一开始就计算这些值并缓存可以减少重复计算。E, b 迭代过程中都会发生变化，每次优化都需要修改部分值，修改规则如下：

$$b_1^{new} = -E_1 - y_1 K_{11}(\alpha_1^{new} - \alpha_1^{old}) - y_2 K_{21}(\alpha_2^{new} - \alpha_2^{old}) + b^{old} \tag{33}$$

$$b_2^{new} = -E_2 - y_1 K_{12}(\alpha_1^{new} - \alpha_1^{old}) - y_2 K_{22}(\alpha_2^{new} - \alpha_2^{old}) + b^{old} \tag{34}$$

$$b^{new} = \begin{cases} b_1^{new} = b_2^{new}, & 0 < \alpha_i < C \\ \frac{1}{2}(b_1^{new} + b_2^{new}), & \alpha_i = 0, C \end{cases} \tag{35}$$

$$E_i^{new} = \sum_{\boldsymbol{S}} y_j \alpha_j K(x_i, x_j) + b^{new} - y_i \tag{36}$$

其中 $\boldsymbol{S}$ 是所有支持向量点的集合，这是因为非支持向量的系数 α 为 0，只使用支持向量的系数计算可以加快计算速度。

2.5 多分类问题（本章节作者：龙征武）

通过软间隔和核技巧，SVM 能有效解决二分类问题，但现实生活中问题往往很难用两个类来描述，因此需要将 SVM 从二分类扩展到多分类。

2.5.1 OvR 和 OvO

在机器学习中，多分类问题的策略是“拆分法”：将多分类问题拆解成若干个二分类问题，每一个二分类问题使用一个分类器求解，最终集成所有分类器的结果，可以求解多分类问题。拆分“策略”是多分类问题的关键，经典的拆分算法有：“一对一”（One vs. One，简称 OvO）、“一对多”（One vs. Rest，简称 OvR）和“多对多”（Many vs. Many 简称 MvM）。

多分类问题的数据集和标签可以记为：

$$D = \{(\boldsymbol{X_1}, y_1), (\boldsymbol{X_2}, y_2), \ldots, (\boldsymbol{X_m}, y_m)\}, y_i \in \{C_1, C_2, \ldots, C_n\} \tag{37}$$

OvR 策略每次选择一个类为正类，其他的类为负类，每次需要训练全部的样本，最终将得到 n 个分类器，对其中的任一个分类器，数据集和标签可以记为：

$$D_i^{(OvR)} = \{(\boldsymbol{X_1}, y_1), (\boldsymbol{X_2}, y_2), \ldots, (\boldsymbol{X_m}, y_m)\}, y_i \in \{C_i, C_{j \neq i}\} \tag{38}$$

OvO 策略则将 m 个类别两两配对，每次训练 2 个类的样本，最终得到 $\frac{n^2-n}{2}$ 个分类器，虽然分类器数较多，由于训练的样本较少，速度可能比 OvR 更快，对其中的任一个分类器，数据集和标签可以记为：

$$D_{ij}^{(OvO)} = \{(\boldsymbol{X_{12}}, y_{12}), (\boldsymbol{X_{13}}, y_{13}), \ldots, (\boldsymbol{X_{IJ}}, y_{IJ})\}, y_{ij} \in \{C_i, C_j\}(i < j, 1 \leq i, j \leq n) \tag{39}$$

将上述二分类求解，可以得到预测结果，例如现有 C_1, C_2, C_3 三种类别，使用 OvR，预测的结果分别为 C_1, C_1C_3, C_1C_2，可以确定预测的结果为 C_1。但有时预测的结果不一，此时可以选择置信度较大的一类作为预测结果，例如预测结果为 C_1, C_1C_3, C_3，则不能确定类别为 C_1 或 C_3，此时可以计算 SVM 的偏离程度，选择偏离程度较大的一类作为预测结果。

MvM 则是上述两种策略的折中，每次选择 m 个正类和 $n-m$ 个负类，一次划分完成后，在对正类和负类的内部在进行划分。常用的策略有 ECOC（Error Correcting Output Codes）和 DAG（Directed Acyclic Graph）。

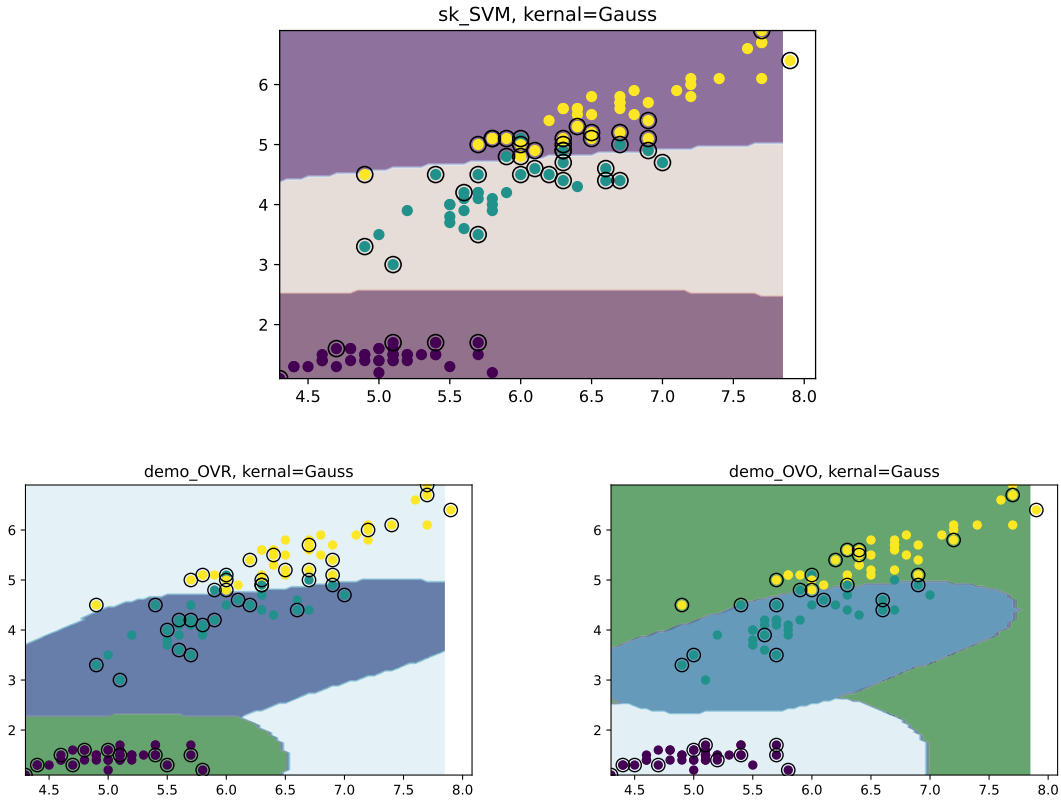


图 3: 多分类 SVM 运行结果 （**绘图使用 matplotlib，代码见附录**）

2.5.2 类别不平衡问题

上诉的算法解决了多分类问题，但是如果出现了类别不平衡问题，即分类时训练的正负样本数量差别太大，上述算法的效率将大打折扣。例如有 99 个正样本，只有 1 个负样本，此时分类器只需要输出预测结果为正就有 99% 的准确率，此时的预测就失去了意义。而在多分类问题中很容易遇到类别不平衡，假如对一个有 100 类使用一个 OvR 分类器，每次预测正样本只有一个类的样本，负样本有 99 个类的数据，负类样本将远远大于正类样本。

解决类别不平衡问题通常有三种做法：一、是负样本“欠采样”，即舍弃一些负样本使得两类样本的数目基本相同，但是这样会损失一些信息，通常可以利用集成学习机制，将负样本划分成多个集合提供给多个分类器，全局来看不会丢失重要信息；二、是正样本“过采样”，即增加一些正样本，但是实践过程中不能使用重复采样，将会导致严重的过拟合问题，可以对一些正样本进行插值得到新的正样本；三、是“阈值移动”，在 SVM 中，通过计算 $f(x) >= 0$ 判断是否是正样本，“阈值移动”则是调整合适的阈值 C，通过计算 $f(x) >= C$ 来判断。

2.5.3 多分类运行测试

通过对 iris 数据集运行 sklearn.svm、OVR、OVO 测试，得到图(3)，可以观察到 OVR 和 OVO 都能有效的解决多分类问题，但是相比 sklearn.svm，OVR 和 OVO 的边界都不够“圆润”和自然。

2.6 SVM 代码优化和测试（本章节作者：龙征武）

当上述的工作都完成后，SVM 似乎是可以在 iris 数据集上正常工作了，我们的 SVM 再测试集取得了 95% 以上的准确率；但是将数据集换成一个更大的数据集，SVM 出现了一个新的问题——运行速度太慢了。

因此 SVM 需要加入一些缓存机制、并行化机制来加速运行。考虑到运行测试的方便性，我们选取 sklearn 提供的手写数字识别数据集，数据集大小为 (1797,64)，标签有 0-9 共 10 个类别，将数据集按照 8:2 划分训练集和测试集，然后记录训练和测试的时间。使用 sklearn 中的 SVM 测试得到的时间是 0.6s。

核函数缓存优化：我们通过计数，运行一次 OvO-SVM 使用了 1.09 亿次核函数，如果我们使用核函数缓存，只需要调用 330 万次（1797×1797）核函数，效率提升了 30 倍以上，而且计算核函数缓存可以更方便使用向量化计算。又由 SVM 的核函数是正定核，即 $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = K(\boldsymbol{x_j}, \boldsymbol{x_i})$，可以使用上三角矩阵存储核函数缓存，因此又可以减少一半的存储空间。运行时间超过了 10 分钟，未测出。

核函数向量化计算：Gauss 核为例，计算 i，j 之间的核函数值可以推广到计算 i 和其他样本的值如下：

$$K(\boldsymbol{x_i}, \boldsymbol{Z}) = exp\left(-\frac{\|\boldsymbol{x_i} - \boldsymbol{Z}\|^2}{2\sigma^2}\right) \tag{40}$$

本文测试了 OvO-SVM 使用向量化之前，核函数计算花费了 160.3s（共花费 171.7s），使用向量化的核函数之后，核函数只花费了 5s（共花费 10.1s）左右。

做完上述优化后，此时最费时间的消耗在更新误差 E，花费 2.8s（共 10.1s），将公式(29)向量化：

$$\boldsymbol{E_i} = K(\boldsymbol{x_i}, \boldsymbol{X_s})(\boldsymbol{\alpha_s} * \boldsymbol{y_s}) + b - y_i \tag{41}$$

其中下标 s 表示支持向量集合，再将上述公式向量化，得到：

$$\boldsymbol{E} = K(\boldsymbol{X}, \boldsymbol{X_s})(\boldsymbol{\alpha_s} * \boldsymbol{y_s}) + b - \boldsymbol{y} \tag{42}$$

更新 E 在完成向量化后，此步花费时间 0.6s，此时共花费 5.7s，核函数计算 4.3s，smo 选择第二个参数 j 花费 0.4s，其他 0.4s。

在使用核函数的 SVM 中时根据公式(29)判别的，此时要根据预测的样本需要计算计算核函数，如果是线性核，还可以根据超平面参数公式(20)直接进行判别，线性核运行时间不超过 0.8s，未测出预测函数耗时。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：杨咏琨）

SVM 的算法实现主要包括 loss 函数和核函数两个部分，loss 函数为超平面的划分提供了度量，核函数则为 SVM 的划分提供更多”角度“，而以上两部分的内容都是基于 SVM 的凸优化问题的求解，SMO 算法为超平面的划分提供了”策略“。限于篇幅，本文的撰写更多关注了计算机求解 SVM 的一些关键性结论，而其中的推导较少。

在写本文之前，三人的状态都是只用过 SVM 的包，没有整体的学习过 SVM 的算法，因为当前深度神经网络的效果要显著优于 SVM。选择 SVM 的原因主要有二：老师的倾力推荐和 SVM 在数据挖掘和机器学习这两门课程的重要性。

课程小作业，在书《统计学习方法》[1] 有 110 个公式，在《机器学习》[2] 有 70 个公式，而且公式之间有很强的关联性，有一个公式没弄懂可能就看不下去了。因此本文的撰写过程是先参考李宏毅的 SVM 视频 [3]，课程给出了很多 SVM 的直观理解；然后再从《机器学习》和《统计学习方法》入门，跟着教材推导

公式，其中看不懂的地方参考了 B 站 UP 主大海-老师的视频 [4]；公式过了一遍之后还是有没弄懂的，此时开始撰写代码，在代码中又理解了一遍公式。

大作业是基于小作业的，此时感觉对 SVM 主要脉络比较熟悉了，先对 SVM 的对偶问题、KKT 条件和核函数进行补充，优化了核函数的向量化，极大的提升了运行速度；补充了核技巧的示意图，参考了视频 [5]，参考《机器学习》实现了 OvO 和 OvR 多分类。

3.1.1 杨咏琨构思和体会

对偶问题和 KKT 条件是 SVM 中的重要基础概念。KKT 条件提出了非线性规划有最优解的必要条件，本文对 KKT 条件进行了简要地阐述，对其中的不等式约束函数和其拉格朗日系数的关系条件进行了证明，但证明过程比较通俗，不是十分严格。对偶问题则是建立在 KKT 条件基础上对计算 SVM 问题的一种优化方法。它先通过 KKT 条件将原不等式和等式的组合问题进行了整合，等价变形为一式，随后通过改变变量的先后约束求解顺序，将问题规模从与维数有关转换到与样本数有关。这在高维问题中这样做会大大减少计算量，同时它也为核函数的使用提供了公式方法。

3.1.2 毕思腾构思和体会

SVM 是一种以监督学习方式对数据进行二分类的经典模型。可以用于线性、非线性的分类，亦可用做回归分析，是最好的、可不加修改直接的一种大间隔分类器，其泛化错误率低，学习能力强，对数据之外的数据点做出很好的分类决策，并且所得到的结果的普适性好。SVM 的问题描述是以分类问题引入，模型的基本定义是在特征空间上间隔最大的线性分类器，间隔最大使它有别于其他感知机，SVM 的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。简单说，SVM 的学习算法本质上就是求解凸二次规划的最优化算法，但其模型逐步推导过程中的优化、简化的数学思想始终贯穿其中，值得不断学习与思考。

3.1.3 龙征武构思和体会

简单就是复杂，SVM 的思想深深的给了我这种体会，SVM 试图以一个平面划分样本，直观上要比神经网络思想简单很多，但完整的推导 SVM 在我感觉要比推导 bp 神经网络难很多。SVM 的精华在于对偶问题、软间隔以及核函数，掌握了这三点就算入门了，这三点无疑对以后类似的最优化问题、核技巧都能给出不错的启示。

3.2 所参考主要资源（本章节作者：杨咏琨）

参考资源如下：

- SVM 模型和对偶问题参考 [2]
- 核技巧参考了 [5]，核函数参考了书 [1] 和 B 站 UP 主大海-老师 [4]
- 看了 SMO 算法论文原文 [6]，简易的 SMO 算法参考了博客 [7]，完整的 SMO 外层循环状态变更参考了 [8]
- 代码基于 numpy 编写，参考了 [8] 和 [9]

3.3 代码撰写的构思和体会（本章节作者：龙征武）

最好直接理解 SVM 的一种方式就是完整的实现一次 SVM，SVM 的代码编写是相当有趣的，但一步想做一个完整的 SVM 是困难的，如果把 SVM 作为一个集合，一步步的来构造组件将会简单很多。代码编写的核心我认为最重要的是理解 KKT 条件，很多时候出现了 bug 往往是 KKT 条件理解不够到位。简化代码的编写可以利用缓存机制，缓存机制能让很多公式和矩阵运算编写变得自然，但是做到这一点我认为必须得先理解 SVM 的几个核心公式。一条不错的编码路线是先使用随机确定 ij 的方法写一个简单的 SMO 算法实现 SVM，然后再将 SMO 扩展完整，核函数可以先实现一对向量的线性核核高斯核，后续在去一个较大的数据集对原来的编码进行向量化。SVM 的优化沉淀了几十年，通过测试得知代码还是不如商业化的 libsvm 和 sklearn 稳定，如果要深入研究，去完整的查看商业软件源码是必要的。原来的计划是使用 numpy 实现之后然后用 C++ 和 Eigen 库进行改写，迫于时间限制和其他课程没有实现。感谢《数据挖掘》课程和夏老师的教导，给我们带来一堂很“硬”的课程，后续的学习还没结束，下一步要扎实的去学一学变分推断。

3.4 人员分工（本章节作者：毕思腾）

作者	主要分工
毕思腾	SVM 分类和使用场景，SVM 问题的描述
杨咏琨	SVM 中的 KKT 条件和对偶问题
龙征武	SMO 算法、多分类问题、代码实现

表 2: 分工表

参考文献

[1] 李航. 统计学习方法. 清华大学出版社, 2012.

[2] 周志华. 机器学习. 清华大学出版社, 2016.

[3] 李宏毅. 机器学习-svm. https://www.bilibili.com/video/BV13x411v7US?p=31, 2020.

[4] 大海-老师. 核函数. https://www.bilibili.com/video/BV1xE411w7Go?spm_id_from=333.999.0.0, 2020.

[5] el mustapha ben bihi. Machine learning: Support vector machine - kernel trick. https://www.youtube.com/watch?v=vMmG_7JcfIc&t=29s, 2017.

[6] John C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[7] 邵正将. 机器学习算法实践-svm 中的 smo 算法. https://zhuanlan.zhihu.com/p/29212107, 2017.

[8] Peter Harrington. 机器学习实战. 人民邮电出版社, 2013.

[9] codestorm04. C++ 实现 svm 算法. https://blog.csdn.net/weixin_43996899/article/details/93528096, 2020.

4 附录

4.1 代码（本章节作者：龙征武）

代码地址:https://github.com/zevellong/tmp/tree/main/%E6%95%B0%E6%8D%AE%E6%8C%96%E6%8E%98%E5%A4%A7%E4%BD%9C%E4%B8%9A

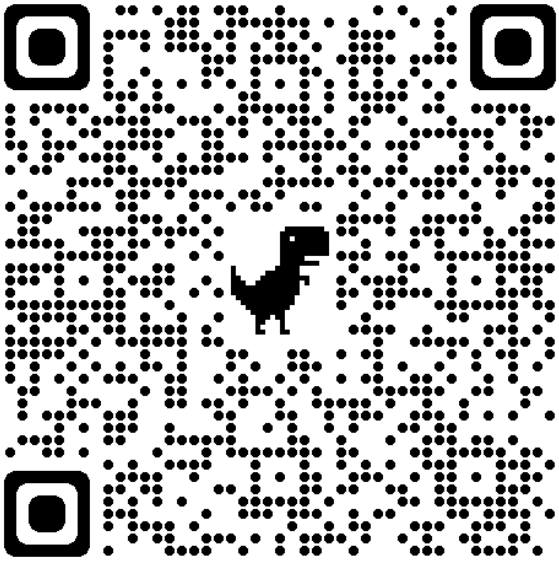


图 4: 代码地址（**二维码由 chrome 自带的二维码插件生成**）

本文的 Python 代码文件描述如下表，详情请见上述网址的 readme.md 文件，代码里面有充足的注释，如果遇到图片保存报错，请修改文件保存前缀前缀，保证该目录下有./fig 文件夹：

- demo_svm_visualization.py：主要用于多分类 SVM 运行及其可视化，数据使用 iris，绘制等高线图，图片为图(3)
- multi_svm_using_numpy.py：demo_SVM 全部实现，软间隔，支持表 1 中的核函数，默认线性核，支持 OvO 和 OvR 多分类
- svm_slove_digits.py:手写数字数据集,主要用于测试 SVM 的运行时间,测试时间下载了 pyinstrument ，pip3 命令行安装
- kernal_split.py：用于绘制核技巧示意图，生成的图片为(1)
- demo_svm_visualization_v2.py：先用玫瑰线参数方程生成散点，然后用 sklearn 和实现的 SVM 测试和以及绘制等高线图，图片为图(2(a))和(2(b))

3.2 《矩阵分解算法在推荐系统研究方向的应用》——刘洛涛，张可欣，胡天扬

摘要：本文主要讨论分析了基于矩阵分解的推荐算法，这一类型的算法通常具有较高的准确性。这里我们以矩阵分解中的奇异值分解算法为核心，展开讲解其算法的原理、核心思想、求解方法、优缺点、及其在推荐系统中的应用，并举了一个简单的例子来讲述其应用过程，进行了代码实现，对结果进行了简单分析。

矩阵分解算法在推荐系统研究方向的应用

刘洛涛 [1], 张可欣 [1]，胡天扬 [2]

[1]Hubei Key Lab of Agricultural Bioinformatics, College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[2]College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

本文主要讨论分析了基于矩阵分解的推荐算法，这一类型的算法通常具有较高的准确性。这里我们以矩阵分解中的奇异值分解算法为核心，展开讲解其算法的原理、核心思想、求解方法、优缺点、及其在推荐系统中的应用，并举了一个简单的例子来讲述其应用过程，进行了代码实现，对结果进行了简单分析。

关键词：矩阵分解，奇异值分解，推荐系统

1 概况

1.1 选题说明（本章节作者：刘洛涛）

随着互联网的迅猛发展，大数据时代的到来，网络上的各种信息越来越丰富，用户如何在繁杂的网络中找到自己需要的或者感兴趣的信息成了一个有待解决的难题。特别是现在大多数人都有一点选择困难症，更多的选择带来的不一定是愉快的体验，也可能会消磨时间，因此，推荐系统就诞生了。推荐系统利用用户的已有信息为用户推荐他们可能感兴趣的或者未来会选择的产品或信息，从而提高用户的使用体验，提高时间利用率，也为商家带来更多商机。

在传统的推荐算法中，大多都是利用相似度来实现推荐的。但是由于物品数量太多，不是所有的用户对每一个物品都有评价，这样我们得到的评分矩阵就是一个非常庞大但稀疏度很高的矩阵。因此为了解决数据稀疏这一问题，往往需要对缺失的数据进行预测 [1]。奇异值分解就是用来解决这一问题的，它可以很好的将数据进行降维，对缺失值进行预测，达到很好的推荐效果。本文主要介绍了矩阵分解算法中的奇异值分解方法，介绍了其原理，及其与特征值分解的关系；介绍了几种不同的矩阵分解算法在推荐系统中的应用，简要分析了矩阵分解算法在推荐系统中的优缺点；举了一个简单的例子来展示奇异值分解的具体过程，并进行了代码实现。

1.2 矩阵分解算法的简介（本章节作者：刘洛涛）

矩阵分解（Matrix Decomposition）是将矩阵拆散为多个矩阵的乘积。它的思路是把评分矩阵通过分解，用一个低秩的矩阵来逼近原来的评分矩阵，逼近的目标是使得预测的矩阵和原来的矩阵之间的误差平方最小 [2]。它可以分为三角分解（LU Decomposition）、满秩分解、QR 分解、Jordan 分解和奇异值（Singular Value Decomposition，SVD）分解等，其中常见的有三种：

1）三角分解 (LU 分解)：可以将一个矩阵分解为一个单位下三角矩阵和一个上三角矩阵的乘积（有时是它们和一个置换矩阵的乘积），LU 分解主要应用在数值分析中，用来解线性方程、求反矩阵或计算行列

式。本质上，LU 分解是高斯消元的一种表达方式。首先，对矩阵 A 通过初等行变换将其变为一个上三角矩阵。然后，将原始矩阵 A 变为上三角矩阵的过程，对应的变换矩阵为一个下三角矩阵。这中间的过程，就是 Doolittle algorithm(杜尔里特算法)。三角分解的前提有：a. 矩阵是方阵; b. 矩阵是可逆的，也就是该矩阵是满秩矩阵，每一行都是独立向量;c. 消元过程中没有 0 主元出现，也就是消元过程中不能出现行交换的初等变换。LU 分解的意义在于求解大型方程组，一个方程组可以简化为 Ax = b 的形式，其中 A 是 n 阶方阵，x 是未知数组成的向量，b 是 n×1 矩阵，而后进行高斯消元求解。

2）QR 分解：把矩阵分解成一个正交矩阵与一个上三角矩阵的积，实数矩阵 A 的 QR 分解是把 A 分解为 A=QR，这里的 Q 是正交矩阵，而 R 是上三角矩阵。QR 分解的实际计算有很多方法，例如 Givens 旋转、Householder 变换，以及 Gram-Schmidt 正交化等等。每一种方法都有其优点和不足。QR 分解经常用来求解线性最小二乘法问题。

3）奇异值分解（SVD）：是另一种正交矩阵分解法，SVD 是最可靠的分解法，但是它比 QR 分解法要花上近十倍的计算时间。[U,S,V]=svd(A)，其中 U 和 V 分别代表两个正交矩阵，而 S 代表一个对角矩阵。和 QR 分解法相同，原矩阵 A 不必为正方矩阵。SVD 分解法主要用于求解最小平方误差和进行数据压缩。本文主要分析奇异值分解法（SVD）及其在推荐系统中的应用。

1.3 奇异值分解算法的描述（本章节作者：刘洛涛）

假设 M 是一个 m×n 阶矩阵，其中的元素全部属于实数域或复数域，那么存在一个分解，使得

$$M = U\Sigma V^*$$

其中 U 是 m×m 阶酉矩阵（酉矩阵是一个 n×n 方阵，它满足如下性质：$U^*U = UU^* = I_n$，其中 U^* 是 U 的共轭转置，I_n 是 n×n 阶单位矩阵）Σ 是 m×n 阶非负实数对角矩阵；而 V^*，即为 V 的共轭转置，是 n×n 阶酉矩阵。这样的分解就称作 M 的奇异值分解。Σ 对角线上的元素即为 M 的奇异值。常见的做法是将奇异值由大到小排列，如此 Σ 便能由 M 唯一确定了（当然 U 和 V 仍然不能确定）。其表示如下（后文进一步讲解其含义）：

$$A_{m\times n} = U_{m\times m}\begin{pmatrix} \triangle_{r\times r} & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{pmatrix}_{m\times n} V^T_{n\times n} \tag{1}$$

其中 $\Delta_{r\times r} = \begin{pmatrix} \delta_1 & & \\ & \delta_2 & \\ & & \ddots \quad \delta_r \end{pmatrix}$，对角线上的元素为 $\delta_i = \sqrt{\lambda_i}$，U 和 V 满足 $U^\top U = E_m$，$V^\top V = E_n$，中间的值除了 $\Delta_{r\times r}$ 外，其余由零补齐。（注：本文中的“*”和“$^\top$”均表示转置）。进一步来说，V 的列向量组成一套对 M 的基向量，这些向量是 M^*M 的特征向量，U 的列向量也组成一套对 M 的基向量，这些向量是 MM^* 的特征向量。Σ 对角线上的元素是奇异值，这些是 MM^* 及 M^*M 的特征值的非负平方根，并与 U 和 V 的行向量对应。

由所学知识可知，若 $A \in \mathbb{R}^{n\cdot m}$，则有 r(A)=r($A^\top A$)=r($AA^\top$)。现将其证明（该定理在文中有体现，故证之）：若 Ax=0, 则等式两边左乘 $A^\top$, 有 $A^\top$Ax=0, 故 Ax=0 的解也是 $A^\top$Ax=0 的解；反之，若 $A^\top$Ax=0，则等式两边左乘 $x^\top$，有 $(Ax)^\top$Ax=0，一个向量转置乘以其本身为 0，则这个向量为零向量，故 Ax=0，由上可证 Ax=0 与 $A^\top$Ax=0 方程同解，故 r(A)=r($A^\top A$)。我们都知道，r(A)=r($A^\top$), 故将上述证明过程的 A 全部换为 $A^\top$，容易得到 r(A)=r($AA^\top$，综上有，若 $A \in \mathbb{R}^{n\cdot m}$，则有 r(A)=r($A^\top A$)=r(A$A^\top$)。

1.4 奇异值和奇异向量与奇异值分解的关系（本章节作者：刘洛涛）

一个非负实数 δ 是 M 的一个奇异值，存在 K^m 的单位向量 u 和 K^n 的单位向量 v 与 δ 的关系如下：

$$Mv = \delta_u$$

$$M^*u = \delta_v$$

其中向量 u 和 v 分别为 δ 的左奇异向量和右奇异向量。对于任意的奇异值分解，$M = U\Sigma V^*$，矩阵 Σ 的对角线上的元素等于 M 的奇异值 U 和 V 的列分别是奇异值中的左、右奇异向量。由上可知，一个 m×n 的矩阵至多有 p = min(m,n) 个不同的奇异值；总能在 K^m 中找到由 M 的左奇异向量组成的一组正交基 U，总能在 K^n 中找到由 M 的右奇异向量组成的一组正交基 V。如果对于一个奇异值，可以找到两组线性无关的左（右）奇异向量，则该奇异值称为简并的（或退化的）。

根据定义，退化的奇异值具有不唯一的奇异向量。因为，如果 u1 和 u2 为奇异值 δ 的两个左奇异向量，则它们的任意归一化线性组合也是奇异值 δ 一个左奇异向量，右奇异向量也具有类似的性质。因此，如果 M 具有退化的奇异值，则它的奇异值分解是不唯一的。非退化的奇异值具有唯一的左、右奇异向量。因此，如果 M 的所有奇异值都是非退化且非零，则除去一个可以同时乘在 U、V 上的任意的相位因子外，M 的奇异值分解唯一。

1.5 奇异值分解与特征分解的关系（本章节作者：刘洛涛）

1.5.1 特征值分解（本章节作者：刘洛涛）

如果一个矩阵 A 有特征值（eigenvalue）和特征向量（eigenvector），则有：

$$A\mathbf{u} = \lambda\mathbf{u}$$

矩阵 A 是一个变换，然后这个变换的特殊之处是当它作用在特征向量 u 上的时候，u 只发生了缩放变换，它的方向并没有改变，并没有旋转。特征向量是经过了变换，这个向量可能会 scale，但是依旧保持其原有的方向，与特定的特征值对应。所以特征向量某种意义上展示了这个变换的‘特征’。

特征分解（Eigendecomposition），又称谱分解（Spectral decomposition）是将矩阵分解为由其特征值和特征向量表示的矩阵之积的方法。需要注意只有对可对角化矩阵才可以施以特征分解。对于矩阵 A，有一组特征向量 v，将这组向量进行正交化单位化，就能得到一组正交单位向量。特征值分解，就是将矩阵 A 分解为如下式：

$$A = Q\Sigma Q^{-1}$$

其中，Q 是矩阵 A 的特征向量组成的矩阵，是正交矩阵，正交矩阵是可逆的，其转置等于其逆矩阵，Σ 则是一个对角矩阵，对角线上的元素就是特征值。首先，我们要明确的是，一个矩阵其实就是一个线性变换，一个矩阵乘以一个向量后得到的还是一个向量，其实就相当于将这个向量进行了线性变换。当矩阵是高维的情况下，那么这个矩阵就是高维空间下的一个线性变换，这个线性变化可能没法通过图片来表示，但是可以想象，这个变换也同样有很多的变换方向，我们通过特征值分解得到的前 N 个特征向量，那么就对应了这个矩阵最主要的 N 个变化方向。我们利用这前 N 个变化方向，就可以近似这个矩阵（变换）。也就是之前说的：提取这个矩阵最重要的特征。

总结一下，特征值分解可以得到特征值与特征向量，特征值表示的是这个特征到底有多重要，而特征向量表示这个特征是什么，可以将每一个特征向量理解为一个线性的子空间，我们可以利用这些线性的子空间干很多的事情。不过，特征值分解也有很多的局限，比如说变换的矩阵必须是方阵。

1.5.2 奇异值分解及与特征值分解的关系（本章节作者：刘洛涛）

特征值分解是一个提取矩阵特征很不错的方法，但是它只适用于方阵。而在现实的世界中，我们看到的大部分矩阵都不是方阵，比如说有 M 个客户，每个客户对应有 N 种物品选择，这样形成的一个 M*N 的矩阵就可能不是方阵，我们怎样才能像描述特征值一样描述这样一般矩阵呢的重要特征呢?

奇异值分解就是用来干这个事的，奇异值分解是一个能适用于任意的矩阵的一种分解的方法。假设 A 是一个 M * N 的矩阵，那么得到的 U 是一个 M * M 的方阵, 里面的向量是正交的，（U 里面的向量为左奇异向量），Σ 是一个 M * N 的实数对角矩阵（对角线以外的元素都是 0，对角线上的元素称为奇异值），V^{top}(V 的转置) 是一个 N*N 的矩阵，里面的向量也是正交的，（V 里面的向量称为右奇异向量）如 1.3 中的公式（1）所示。那么奇异值和特征值是怎么对应起来的呢?

首先，我们将一个矩阵 A 的转置 $A^\top$*A，将会得到 $A^\top$A 是一个方阵，我们用这个方阵求特征值可以得到：

$$\left(A^T A\right) v_i = \lambda_i v_i$$

这里得到的 v，就是我们上面的右奇异向量。然后我们根据 A*$A^\top$，同样可以得到一个方阵，那我们同样可以根据这个方阵求其特征值和特征向量：

$$\left(AA^T\right) u_i = \lambda_i u_i$$

（这里简单证明一下矩阵 A$A^\top$ 和 $A^\top$A 的特征值相等：设 A$A^\top$ 的特征值为 λ，特征向量为 x，所以有

$$(AA^\top - \lambda I)x = 0$$

这里 I 为单位矩阵，等式两边的左端乘以 $A^\top$，可以得到

$$(A^\top AA^\top - \lambda A^\top)x = 0$$

从而因式分解有：

$$(A^\top A - \lambda I)A^\top x = 0$$

从而可以看出来，$A^\top$x 是 $A^\top$ 的一个特征向量，其对应的特征值为 λ，综上所述可知，矩阵 A$A^\top$ 和 $A^\top$A 有相同的特征值，但其特征向量不一定相同）。这样就可以得到左奇异向量 u，两个矩阵 $\left(A^T A\right)$ 和 $\left(AA^T\right)$ 的特征值 λ 就是我们所要求的奇异值。奇异值 λ 与特征值类似，矩阵 Σ 中一般也是从大到小排列，而且 λ 的减少特别的快，在很多情况下，前 10%甚至 1%的奇异值的和就占了全部的奇异值之和的 99%以上了。也就是说，我们也可以用 topR 大的奇异值来近似描述矩阵，这也是 1.3 中的公式（1）中右下矩阵对角线也全部为 0 的原因。

进一步分析，给定一个 M 的奇异值分解，M 的转置为 M^*，两者的关系如下：

$$M^*M = V\Sigma^*U^*U\Sigma V^* = V\left(\Sigma^*\Sigma\right)V^*$$

$$MM^* = U\Sigma V^*V\Sigma^*U^* = U\left(\Sigma\Sigma^*\right)U^*$$

关系式的右边描述了关系式左边的特征值分解。于是：V 的列向量（右奇异向量）是 M^*M 的特征向量，U 的列向量（左奇异向量）是 MM^* 的特征向量。Σ 的非零对角元（非零奇异值）是 M^*M 或者 MM^* 的非零特征值的平方根。

特殊情况下，当 M 是一个正规矩阵，根据谱定理（谱定理是关于线性算子或者矩阵的一些结果，它给

出了算子或者矩阵可以对角化的条件），M 可以被一组特征向量酉对角化，所以它可以表示为：

$$M = UDU^*$$

其中 U 为一个酉矩阵，D 为一个对角阵。如果 M 是半正定（半正定：当且仅当对所有不为零的 $z \in \mathbb{R}^n$ (或 $z \in \mathbb{C}^n$)，都有 $z^*Mz \geq 0$），则 $M = UDU^*$ 的分解也是一个奇异值分解。

然而，一般矩阵的特征分解跟奇异值分解不同。特征分解如下：

$$M = UDU^{-1}$$

其中 U 是不需要酉的，D 也不需要是半正定的。而奇异值分解如下：

$$M = U\Sigma V^*$$

其中 Σ 是对角半正定矩阵，U 和 V 是酉矩阵，两者除了都是通过矩阵 M 没有必然联系。

综上所述可知，奇异值分解的步骤如下：a. 求 $AA^\top$ 的特征值和特征向量，用单位化的特征向量构成 U；b. 求 $A^\top A$ 的特征值和特征向量，用单位化的特征向量构成 V；c. 将 $AA^\top$ 或者 $A^\top A$ 的特征值求平方根，然后构成 Σ。

2 矩阵分解算法在推荐系统中的应用

2.1 推荐系统策略（本章节作者: 张可欣）

2.1.1 推荐系统策略（本章节作者: 张可欣）

随着大数据时代到来，网络信息越来越复杂，用户们经常迷失在琳琅满目的购物清单中，这种浏览大量无关信息的过程大大减低了用户的购物体验。为了解决这一问题，推荐系统应运而生。1997 年，Rensnick 和 Varian 等人首次提出推荐系统 [3] 这一概念：“利用电子商务网站向客户提供商品信息和建议，帮助用户决定购买什么产品，模拟销售人员帮助客户完成购买过程。”这样的个性化推荐系统能够根据用户的兴趣爱好和购买记录为用户推荐合适的产品。可以说，推荐系统本质上是对人的某种行为的模拟，其通过推荐策略对特定的数据信息处理与建模，从而将最可能的结果推荐给有相关需求的用户。

2.1.2 协同过滤推荐（本章节作者: 张可欣）

常见的推荐策略分为：1、基于内容的推荐（Content-based Recommendation)2、协同过滤推荐（Collaborative Filtering-based Recommendation, CFR)3、混合推荐 (Hybrid Recommendation)。协同过滤推荐与基于内容的推荐相比, 其优点在于: 首先, 可以帮助用户发现新的兴趣, 协同过滤可以发现与用户已知兴趣不同的潜在兴趣偏好; 其次, 通过共享他人经验, 可避免内容分析的不完全和不准确, 并且能够基于复杂概念 (如个人品味) 进行过滤; 最后, 能够通过使用其他相似用户的反馈信息加快个性化学习速度。因此, 协同过滤成为目前应用最广泛最成功的推荐技术之一。

协同过滤推荐来源于一个简单猜想：如果用户对这个产品感兴趣，那和这个产品相似性较高的其他产品同样能吸引住用户。此外，对一类产品具有相似兴趣的用户来说，他们可能也会同时喜欢其他相似的产品。协同过滤技术可以分为基于内存的协同过滤（Memory-based CF）和基于模型的协同过滤（Model-based CF）。其中基于模型的推荐算法使用机器学习与数据挖掘技术, 从训练数据中确定模型并将模型用于预测未知商品的评分。近十几年，由于矩阵分解在预测模型上的良好效果，包括主成分分析（Principle Component

Analysis）算法、奇异值矩阵分解（Singular Value Decomposition）算法和非负矩阵分解（Non-negative Matrix Factorization）算法等在推荐系统领域得到了大量应用和拓展。

2.2 矩阵分解用于推荐算法要解决的问题（本章节作者：胡天扬）

首先，我们想一下推荐系统为什么需要矩阵分解？因为大型网站的用户太多了，而且推荐的商品也多。你可以想象一下 3 亿的用户对上 300 万商品，这是个什么概念，这个维度得多大，所以，直接求解这个大矩阵不现实，所以通过求解两个小矩阵还是比较方便存储。另一个是我们想挖掘用户和物品的隐藏关联，但这个关联是未知的，隐藏的关联，比如：用户看电影，用户与电影类型的关系，电影与电影类型的关系，那么电影类型就是一个隐藏因子，通过隐藏因子将用户和物品关联起来。而在推荐系统中，我们常常遇到的问题是这样的，我们有很多用户和物品，也有少部分用户对少部分物品的评分，我们希望预测目标用户对其他未评分物品的评分，进而将评分高的物品推荐给目标用户。对于每个用户，我们希望较准确的预测出用户对未评分物品的评分。对于这个问题我们有很多解决方法，本文我们关注于用矩阵分解的方法来做。如果将 m 个用户和 n 个物品对应的评分看做一个矩阵 MM，我们希望通过矩阵分解来解决这个问题。

2.3 几种应用于推荐系统的矩阵分解算法（本章节作者：张可欣）

2.3.1 SVD（奇异值分解算法）（本章节作者：张可欣）

SVD 算法通过将用户、产品的评分矩阵 $M_{m\times n}$ 通过降维、分解，计算成三个低阶矩阵相乘，对这三个低阶矩阵进行训练最后还原回初始的矩阵。在下面的公式中左右两个矩阵分别表示用户特征矩阵 U 和产品特征矩阵 V，中间矩阵为由 k 个奇异值组成的对角矩阵。这样如果我们要预测第 i 个用户对第 j 个产品的评分 m_{ij}, 则只需要计算 $u_i \sum v_j^T$ 即可。

$$M_{m\times n} = U_{m\times k} \sum_{m\times k} V_{k\times n}^T$$

通过这种方法，我们可以将评分矩阵中没有评分的产品通过模型得到预测评分并将几个评分最高的物品推荐给用户。

SVD 算法虽然思路简单，但它的缺陷是必须要求评分矩阵里的元素非空，因此在使用前需要先使用均值或者其他统计学方法来进行矩阵补全。在实际应用过程中，缺失值往往可以达到 95% 的高额比例，这也导致了传统 SVD 算法复杂度过高从而引发的过拟合和缺乏精确度等问题。

2.3.2 FunkSVD 算法（本章节作者：张可欣）

2006 年，Simon Funk 在优化 SVD 计算效率问题上提出了 FunkSVD 模型，它不再分解为三个矩阵，而是首先把评分矩阵 $M_{m\times n}$ 分解成用户特征矩阵 P 和产品特征矩阵 Q。

$$M_{m\times n} = P_{m\times k}^T Q_{k\times n}$$

这样第 i 个用户对第 j 件产品的评分预测可以表示为 $q_j^T p_i$。因此可以通过计算产品预测评分与真实评分之间的均方误差得到损失函数，其目标优化函数为

$$\arg\min_{p_i,q_j} \sum_{i,j} \left(m_{ij} - q_j^T p_i\right)^2 + \lambda \left(\|p_i\|_2^2 + \|q_j\|_2^2\right)$$

最小化上式的过程可以用我们熟悉的梯度下降或最小交替二乘法来实现。此外为防止模型过拟合，FunkSVD 引入了大量数据挖掘算法同样出现过的正则化项（例如逻辑回归）。这里的正则化其实就是在目标函数中加上用户因子向量和产品因子向量的一范数或二范数。至于使用 $l1$ 正则化还是 $l2$ 正则化要看具体情况，与逻辑回归正则项的意义类似，这里 $l1$ 正则化同样会使很多因子为 0，从而减小模型大小，而 $l2$ 正则化只能使因子接近于 0，而不能使其为 0。将上式分别对 p_i, q_j 求导我们得到

$$\frac{\partial J}{\partial p_i} = -2\left(m_{ij} - q_j^T p_i\right) q_j + 2\lambda p_i$$

$$\frac{\partial J}{\partial q_j} = -2\left(m_{ij} - q_j^T p_i\right) p_i + 2\lambda q_j$$

进而得出 p_i, q_j 的迭代公式

$$p_i = p_i + \alpha\left(\left(m_{ij} - q_j^T p_i\right) q_j - \lambda p_i\right)$$

$$q_j = q_j + \alpha\left(\left(m_{ij} - q_j^T p_i\right) p_i - \lambda q_j\right)$$

通过迭代，最终可以得到矩阵 P 和 Q 并用于产品推荐。

FunkSVD 算法可以有效处理海量数据并能达到比较理想的推荐精度，但它的缺点是没有考虑用户评分偏置，预测精度有待提高。此外当数据集非常稀疏时，该算法性能明显下降 [4]。

2.3.3 BiasSVD 算法（本章节作者：张可欣）

FunkSVD 算法出现后，人们经过观测发现，评分数据大部分都是和用户或物品无关的因素产生的效果，即有很大一部分因素和用户对物品的喜好无关而只取决于用户或物品本身特性。出于这样的考虑，BiasSVD 算法引入了用户的评分偏置和产品的被评分偏置项，这些独立于用户或独立于物品的因素称为偏置 (Bias) 部分。假设评分系统平均分为 μ，第 i 个用户的用户偏置项为 b_i，而第 j 个物品的物品偏置项为 b_j，则加入了偏置项以后的优化目标函数 $J(p,q)$ 可以表示为：

$$\arg\min_{p_i,q_j} \sum_{i,j} \left(m_{ij} - \mu - b_i - b_j - q_j^T p_i\right)^2 + \lambda\left(\|p_i\|_2^2 + \|q_j\|_2^2 + \|b_i\|_2^2 + \|b_j\|_2^2\right)$$

并同样可以采用梯度下降法优化目标函数。和 FunkSVD 算法不同，因为考虑了两个评分偏执项 b_i,b_j，因此 b_i,b_j 也要参与迭代过程，其迭代公式可以写为：

$$b_i = b_i + \alpha\left(m_{ij} - \mu - b_i - b_j - q_j^T p_i - \lambda b_i\right)$$

$$b_j = b_j + \alpha\left(m_{ij} - \mu - b_i - b_j - q_j^T p_i - \lambda b_j\right)$$

通过迭代最终可以得到 P 和 Q，进而用于产品评分及推荐。

BiasSVD 因为增加了一些额外因素的考虑，因此在某些场景会比 FunkSVD 表现好。

2.3.4 SVD++ 算法（本章节作者：张可欣）

首先，我们知道产品推荐系统的运作是依赖于用户提供的反馈信息的。反馈信息一般分为两种：隐式反馈信息 (如用户的购买记录、浏览记录、鼠标点击情况等) 和显式反馈信息 (如用户对产品的评分情况)。显示反馈信息虽然容易分析但是含有大量隐藏信息的隐式反馈也同样重要。SVD++ 算法在 BiasSVD 算法基础上针对此问题做了增强，对于某一个用户 i，它提供了隐式反馈的物品集合定义为 N(i)，这个用户对某个物品 j 对应的隐式反馈修正的评分值 c_{ij}，那么该用户所有的评分修正值为 $\sum_{s\in N(i)} c_{sj}$。则加入了隐式反

馈项以后的优化目标函数 J(p,q) 可以写为:

$$\arg\min_{p_i,q_j}\sum_{i,j}\left(m_{ij}-\mu-b_i-b_j-q_j^Tp_i-q_j^T|N(i)|^{-1/2}\sum_{s\in N(i)}y_s\right)^2$$
$$+\lambda\left(\|p_i\|_2^2+\|q_j\|_2^2+\|b_i\|_2^2+\|b_j\|_2^2+\sum_{s\in N(i)}\|y_s\|_2^2\right)$$

其中 $|N(i)|^{-1/2}$ 是为了消除不同 |N(i)| 个数引起的差异。

SVD++ 算法在预测评分时，引入隐式反馈增加了预测准确度，尤其适用于显式反馈缺失的情况。但是它同样伴随着缺点：隐式反馈数据的采集并没有统一的衡量标准，算法的性能受使用场景的限制 [4]。

2.3.5 timeSVD++ 算法（本章节作者：张可欣）

上述的几种模型都是考虑的静态情况，但在现实过程中，物品的流行度和用户的兴趣都会随着时间而动态改变。因此 timeSVD++ 在此前提下考虑了用户和产品偏置及用户的隐含因子。该模型的核心思想是：为每个时间段学习一个参数，某个时间段参数使用该时间段数据进行学习。通过建模海量数据，自动发现潜在的时间模式。timeSVD++ 模型可以表示为：

$$\hat{r}_{ui}=\mu+b_u(t)+b_i(t)+q_i^Tp_u(t)$$

其中，$b_u(t)$、$b_i(t)$ 分别是用户和物品偏置随着时间变化的函数，$p_u(t)$ 是用户隐因子随时间变化的函数。

timeSVD++ 模型提高了用户近期隐式反馈行为的权重，而对用户的早期的反馈信息的权重进行了衰减，近似的实现了动态的推荐目的 [5]。

2.3.6 PMF（概率矩阵分解算法）（本章节作者：张可欣）

PMF 算法从概率的角度预测用户对项目的评分，是 funkSVD 算法的概率化形式。它认为评分矩阵中的元素 $R_{i,j}$ 由用户潜在偏好向量 U_i^T 和物品潜在属性向量 V_j 的内积决定，并且服从均值为 $U_i^TV_j$，方差为 σ^2 的高斯分布：

$$\mathbf{R}_{i,j}\sim\mathbf{N}\left(U_i^TV_j,\sigma^2\right)$$

因为本次实验我们主要分析 SVD 算法，因此仅仅对 PMF 的算法原理简单提及。

PMF 算法的优点是能够自动学习模型中的参数，达到更好的模型稳定性和更高的推荐准确率。但是该算法的前提假设是用户和项目的特征矩阵均满足高斯分布，因此当评价数据极度稀疏的情况下，很难保证这一假设 [4]。

2.3.7 矩阵分解算法小结（本章节作者：张可欣）

FunkSVD 将矩阵分解用于推荐系统推到了新的高度，在实际应用中也使用非常广泛。当然矩阵分解方法也在不停的进步，目前张量分解和分解机方法是矩阵分解应用推荐系统今后的一个趋势。对于矩阵分解用于推荐系统本身来说，它容易编程实现，实现复杂度低，预测效果也好，同时还能保持扩展性，这些都是它宝贵的优点。当然，矩阵分解方法有时候解释性还是没有基于概率的逻辑回归之类的推荐算法好，不过这也不影响它的流行程度。小型推荐系统用矩阵分解应该是一个不错的选择，大型的话，深度学习模型的推荐能够克服传统的推荐系统出现的问题以及改进不同领域推荐模型的方向。

2.4 SVD 应用于推荐系统（本章节作者：胡天扬）

在上一节中，我们介绍几种经典且热门的矩阵分解算法，而在本文中我们使用的是 SVD，也就是传统的奇异值分解。我们会使用实际类似 word2vec 的思想，一开始初始化这两个矩阵 P、Q，把它们作为需要训练的参数，然后通过样本来训练模型，构建损失函数，通过梯度下降来更新参数，这就是一个完整的落地流程。数据集中行代表用户 user，列代表物品 item，其中的值代表用户对物品的打分。基于 SVD 的优势在于：用户的评分数据是稀疏矩阵，可以用 SVD 将原始数据映射到低维空间中，然后计算物品 item 之间的相似度，可以节省计算资源。整体思路：先找到用户没有评分的物品，然后再经过 SVD “压缩” 后的低维空间中，计算未评分物品与其他物品的相似性，得到一个预测打分，再对这些物品的评分从高到低进行排序，返回前 N 个物品推荐给用户。

2.4.1 SVD 降维（本章节作者：胡天扬）

机器学习的一个最根本也是最有趣的特性是数据压缩概念的相关性。如果我们能够从数据中抽取某些有意义的感念，则我们能用更少的比特位来表述这个数据。从信息论的角度则是数据之间存在相关性，则有可压缩性。SVD 就是用来将一个大的矩阵以降低维数的方式进行有损地压缩。下面我们将用一个具体的例子展示 SVD 的具体过程。首先我们假设 A 矩阵的值是 $A=\begin{bmatrix}5&5&0&5\\5&0&3&4\\3&4&0&3\\0&0&5&3\\5&4&4&5\\5&4&5&5\end{bmatrix}$ 使用 matlab 调用 svd 函数：

[U,S,Vtranspose]=svd(A) 能得出

$$U=\begin{bmatrix}-0.4472&-0.5373&-0.0064&-0.5037&-0.3857&-0.3298\\-0.3586&0.2461&0.8622&-0.1458&0.0780&0.2002\\-0.2925&-0.4033&-0.2275&-0.1038&0.4360&0.7065\\-0.2078&0.6700&-0.3951&-0.5888&0.0260&0.0667\\-0.5099&0.0597&-0.1097&0.2869&0.5946&-0.5371\\-0.5316&0.1887&-0.1914&0.5341&-0.5485&0.2429\end{bmatrix}$$

$$S=\begin{bmatrix}17.7139&0&0&0\\0&6.3917&0&0\\0&0&3.0980&0\\0&0&0&1.3290\\0&0&0&0\\0&0&0&0\end{bmatrix}$$

$$Vtranspose=\begin{bmatrix}-0.5710&-0.2228&0.6749&0.4109\\-0.4275&-0.5172&-0.6929&0.2637\\-0.3846&0.8246&-0.2532&0.3286\\-0.5859&0.0532&0.0140&-0.8085\end{bmatrix}$$

分解矩阵之后我们首先需要明白 S 的意义。可以看到 S 很特别，是个对角线矩阵。每个元素非负，而且依次减小，具体要讲明白元素值的意思大概和线性代数的特征向量，特征值有关。但是可以大致理解如下，在线性空间里，每个向量代表一个方向。所以特征值是代表该矩阵向着该特征值对应的特征向量的方向的变化权重。

所以可以取 S 对角线上前 k 个元素，当 k=2 时候即将 S（6*4）降维成 S（2*2），同时 U(6*6),Vtranspose(4*4) 相应地变为 U(6*2),Vtranspose(4*2)。此时我们用降维后的 U，S，V 来相乘得到 A2。A2 和 A 很接近，这就是之前说的降维可以看成一种数据的有损压缩。

2.4.2 寻找相似用户（本章节作者：胡天扬）

我们假设现在有个名字叫 Bob 的新用户，并且已知这个用户对 season n 的评分向量为：[5 5 0 0 0 5]。我们的任务是要对他做出个性化的推荐，我们的思路首先是利用新用户的评分向量找出该用户的相似用户。通过算法我们会得到一个 Bob 的二维向量，即知道 Bob 的坐标，根据这个坐标我们将找出和 Bob 最相似的用户。但要注意的是，最相似并不是距离最近的用户，这里的相似用余弦相似度计算。

2.4.3 推荐系统隐语义模型中应用（本章节作者：胡天扬）

假定有 U 个用户，V 个 item，R 为打分矩阵假定有 K 个隐含变量，我们需要找到矩阵 P(U*K) 和 Q(K*V):

$$R \approx P \times Q^T = \widehat{R}$$

$$\widehat{r}_{ij} = q_j^T p_i = \sum_{k=1}^{k} q_{ik} p_{kj}$$

那么我们应该如何找到最佳的 P 和 Q 呢，这是我们就可以用到梯度下降，由于用户和物品的特征向量维度比较低，因而可以通过梯度下降的方法高效地求解。我们先定义损失函数 (加正则化项)：

$$e_{ij}^2 = \left(r_{ij} - \sum_{k=1}^{k} q_{ik} p_{kj} \right)^2 + \frac{\beta}{2} \left(\|P\|^2 + \|Q\|^2 \right)$$

然后求梯度/偏导，更新迭代公式：

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha \left(2 e_{ij} q_{kj} - \beta p_{ik} \right)$$

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha \left(2 e_{ij} p_{ik} - \beta q_{kj} \right)$$

最后再还原回矩阵乘积，即可补充未打分项。

2.4.4 代码实现（本章节作者：胡天扬）

具体代码见 https://github.com/hty1551/SDV 而我们的代码主要分为 5 个部分。第 1 部分：加载测试数据集；第 2 部分：定义三种计算相似度的方法，分别是欧式距离、皮尔逊相关系数和余弦相似度；第 3 部分：通过计算奇异值平方和的百分比来确定将数据降到多少维才合适，按照前 k 个奇异值的平方和占总奇异值的平方和的百分比 percentage 来确定 k 的值，后续计算 SVD 时需要将原始矩阵转换到 k 维空间，返回需要降到的维度；第 4 部分：在已经降维的数据中，基于 SVD 对用户未打分的物品进行评分预测，返回未打分物品的预测评分值，这里用到了 svdEst() 函数，数据矩阵的行对应用户，列对应物品，函数的作用是基于 item 的相似性对用户未评过分的物品进行预测评分；第 5 部分：产生前 N 个评分值高的物品，返回物品编号以及预测评分值，函数 recommend() 会产生预测评分最高的 N 个推荐结果，默认返回 5 个。

2.5 实验结果与分析（本章节作者：胡天扬）

2.5.1 实验结果（本章节作者：胡天扬）

方法	结果
余玄相似度	(2,2.5)，(1,2.0)
欧氏距离	(2,3.0)，(1,2.8)
皮尔逊相关度	(2,2.5)，(1,2.0)
更换数据集且物品评价函数	(4,3.344)，(7,3.329)，(9，3.328)

表 1: 实验结果

表 1 前三行是使用三种计算距离的函数之中的一种进行的测试，而表一最后一行的结果是用 svdEst 函数来替换物品评价函数 standEst，并且使用了更复杂的数据集。上面的之所以使用 4 这个数字，是因为通过预先计算得到能满足九成的奇异值能量的前 N 个奇异值，在函数 svdEst 中使用 SVD 方法，将数据集映射到低纬度的空间中，再做运算。其中的 xformedItems = dataMat.T*U[:,:4]*Sig4.I 可能不是很好理解，它就是 SVD 的降维步骤，通过 U 矩阵和 Sig4 逆矩阵将商品转换到低维空间。

2.5.2 矩阵分解在推荐中的优缺点（本章节作者：胡天扬）

SDV 的优势在于：用户的评分数据是稀疏矩阵，可以用 SVD 将数据映射到低维空间，然后计算低维空间中的 item 之间的相似度，对用户未评分的 item 进行评分预测，最后将预测评分高的 item 推荐给用户。而且比较容易编程实现，随机梯度下降方法依次迭代即可训练出模型；比较低的时间和空间复杂度，高维矩阵映射为两个低维矩阵节省了存储空间，训练过程比较费时，但是可以离线完成；评分预测一般在线计算，直接使用离线训练得到的参数，可以实时推荐；预测的精度比较高，预测准确率要高于基于领域的协同过滤以及内容过滤等方法；非常好的扩展性，很方便在用户特征向量和物品特征向量中添加其它因素，例如添加隐性反馈因素的 SVD++，添加时间动态 time SVD++，此方法将偏置部分和用户兴趣都表示成一个关于时间的函数，可以很好的捕捉到用户的兴趣漂移。

但矩阵分解当然还有不足之处，第一就是在训练模型的时候时间开销比较大。其次就是推荐结果不具有很好的可解释性，分解出来的用户和物品矩阵的每个维度无法和现实生活中的概念来解释，无法用现实概念给每个维度命名，只能理解为潜在语义空间。其实我们不必在每次评分是都做 SVD 分解，大规模数据上可能降低效率，可以在程序调用时运行一次，在大型系统中每天运行一次或频率不高，还要离线运行。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：刘洛涛）

相比于第一次的小论文写作，我们小组这次大论文的构思和撰写相对来说要顺畅很多，在之前出现问题的环节这次都有所避免，但又难免出现了一些新的问题，下面对我们整个构思和撰写过程做一个简单回顾。

首先在选题上，我们是先确定了两个可选项，矩阵分解算法和 EM 算法，因为这两个算法老师讲解的相对更为详细，我们理解更透彻，也更有钻研的兴趣，进一步讨论决定，选择矩阵分解算法。通过查阅资料了解到，矩阵分解算法在推荐系统方面的应用已经较为成熟，并且取得了不错的效果，我们便决定做这方面的研究工作。

分工方面，我们这次没有太多交叉划分，避免在写作上有牵绊，直接每个人独立负责一大块，比如一个人负责原理分析，一个人负责推荐系统上的应用，一个人主要负责实例研究，即便如此，我们在写作过程中

还是出现了因沟通不当而出现的内容重复事件，即个体在写作过程中对群消息关注度不够导致重复某部分内容的撰写，但是进一步协商后问题也得到了解决。

通过这两次大小论文的写作，我们一致认为自己的选题能力、写作能力、协商能力、处理问题的能力都提升了不少，期待老师的指点，我们会一直在前进的路上，永不停息！

3.2 所参考主要资源（本章节作者：胡天扬）

参考代码以及算法

https://blog.csdn.net/wuyanyi/article/details/7964883

https://blog.csdn.net/qq$_3$6523839/*article*/*details*/82347332

https : //*zhuanlan.zhihu.com*/*p*/29846048

https : //*blog.csdn.net*/*jirongzi*$_c$*s*2011/*article*/*details*/9499633

3.3 代码撰写的构思和体会（本章节作者：胡天扬）

毕竟我们也是第一次接触到这个课题，而且在学习过程中觉得 SVD 有一定的难度，但主要步骤还是能勉强理解的，在学习 SVD 的同时补足了许多基础知识，当然代码大部分也是借鉴的网上开源项目，自己做了一些调整。首先在做推荐系统的时候，我们通过 SVD 对数据的处理，可以使用小得多的数据集来表示原始数据集，这样做实际上是去除了噪声和冗余信息，以此达到了优化数据、提高结果的目的。在撰写代码的时候用到了基于协同过滤的搜索引擎，大致有两种方法来实现他分别是基于用户的协作型过滤和基于物品的协作型过滤，两种方法大致相同，但是在不同的环境下，使用最佳的方法能最大化的提升算法的效果。基于物品相似度的计算时间会随着物品数量的增加而增加。基于用户相似度则取决于用户数量，打一个比方，就像一个最大的商店拥有大概 100000 种商品，而它的用户可能有 500000 人，这时选择基于物品相似度可能效果好很多。其实代码的构思在上一章中也大致介绍完毕了。

在本次课题代码撰写中我们学习到了 SVD 的相关理论与应用，了解了多种矩阵分解算法，也学习到了基于协同过滤的搜索引擎的算法编写。总而言之见识到也学习到了许多。

3.4 人员分工（本章节作者：刘洛涛）

刘洛涛：论文构思、矩阵分解算法的原理分析部分。

张可欣：矩阵分解算法在推荐系统中的应用。

胡天杨：矩阵分解算法的优缺点、实例应用、代码实现。

参考文献

[1] Zhao Yajuan. Review of recommendation algorithms based on singular value decomposition. 2015.

[2] Xiao Huihui; Liu Huiting; Chen Yan. Matrix decomposition recommendation algorithm based on user preference. 2015.

[3] Resnick; Paul and Varian; Hal R. Recommender systems. 40(3), 1997.

[4] 翁小兰; 王志坚. 协同过滤推荐算法研究进展. 计算机工程与应用, 54:25–31, 2018.

[5] 于蒙; 何文涛; 周绪川; 崔梦天; 吴克奇; 周文杰. 推荐系统综述. 计算机应用, 20:1001–9081, 2021.

4 附录

4.1 源码（本章节作者：胡天扬）

```
参考代码: https://blog.csdn.net/qq_36523839/article/details/82347332
代码: https://github.com/hty1551/SDV
from numpy import *
from numpy import linalg as la

 # 为0表示该用户未评价此商品, 即可以作为推荐商品
def loadExData():
    return [[0, 0, 0, 2, 2],
            [0, 0, 0, 3, 3],
            [0, 0, 0, 1, 1],
            [1, 1, 1, 0, 0],
            [2, 2, 2, 0, 0],
            [5, 0, 5, 0, 0],
            [1, 1, 1, 0, 0]]

# 欧几里德距离 这里返回结果已处理 0, 1   0最大相似, 1最小相似   欧氏距离转换为2范数计算
def ecludSim(inA,inB):
    return 1.0 / (1.0 + la.norm(inA-inB))

# 皮尔逊相关系数 numpy的corrcoef函数计算
def pearsSim(inA,inB):
    if(len(inA) < 3):
        return 1.0
    return 0.5 + 0.5*corrcoef(inA,inB,rowvar=0)[0][1]   # 使用0.5+0.5*x 将-1, 1 转为 0, 1

# 余玄相似度 根据公式带入即可, 其中分母为2范数计算, linalg的norm可计算范数
def cosSim(inA,inB):
    num = float(inA.T * inB)
    denom = la.norm(inA) * la.norm(inB)
    return 0.5 + 0.5*(num/denom)     # 同样操作转换 0, 1


# 对物品评分
def standEst(dataMat, user, simMeas, item):
    n = shape(dataMat)[1]   # 获得特征列数
    simTotal = 0.0; ratSimTotal = 0.0   # 两个计算估计评分值变量初始化
    for j in range(n):
        userRating = dataMat[user,j]    #获得此人对该物品的评分
        if userRating == 0: # 若此人未评价过该商品则不做下面处理
            continue
        overLap = nonzero(logical_and(dataMat[:,item].A>0,dataMat[:,j].A>0))[0]   # 获得相比较的两列同时都不为0的数据行号
        if len(overLap) == 0:
            similarity = 0
        else:
            # 求两列的相似度
            similarity = simMeas(dataMat[overLap,item],dataMat[overLap,j]) # 利用上面求得的两列同时不为0的行的列向量 计算距离
        # print('%d 和 %d 的相似度是: %f' % (item, j, similarity))
        simTotal += similarity  # 计算总的相似度
        ratSimTotal += similarity * userRating  # 不仅仅使用相似度, 而是将评分当权值*相似度 = 贡献度
    if simTotal == 0:   # 若该推荐物品与所有列都未比较则评分为0
        return 0
    else:
        return ratSimTotal/simTotal # 归一化评分 使其处于0-5（评级）之间
```

```
54
55 # 给出推荐商品评分
56 def recommend(dataMat, user, N=3, simMeas=cosSim, estMethod=standEst):
57     unratedItems = nonzero(dataMat[user,:].A==0)[1]  # 找到该行所有为0的位置
58     if len(unratedItems) == 0:
59         return '所有物品都已评价...'
60     itemScores = []
61     for item in unratedItems:   # 循环所有没有评价的商品列下标
62         estimatedScore = estMethod(dataMat, user, simMeas, item)    # 计算当前产品的评分
63         itemScores.append((item, estimatedScore))
64     return sorted(itemScores, key=lambda jj: jj[1], reverse=True)[:N]   # 将推荐商品排序
65
66 # 结果测试如下:
67 myMat = mat(loadExData())
68 myMat[0,1] = myMat[0,0] = myMat[1,0] = myMat[2,0] = 4   # 将数据某些值替换, 增加效果
69 myMat[3,3] = 2
70 result1 = recommend(myMat,2)          # 余玄相似度
71 print(result1)
72 result2 = recommend(myMat,2,simMeas=ecludSim)   # 欧氏距离
73 print(result2)
74 result3 = recommend(myMat,2,simMeas=pearsSim)   # 皮尔逊相关度
75 print(result3)
76
77
78
79
80  # 为0表示该用户未评价此商品, 即可以作为推荐商品
81 def loadExData2():
82     return [[0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 5],
83             [0, 0, 0, 3, 0, 4, 0, 0, 0, 0, 3],
84             [0, 0, 0, 0, 4, 0, 0, 1, 0, 4, 0],
85             [3, 3, 4, 0, 0, 0, 0, 2, 2, 0, 0],
86             [5, 4, 5, 0, 0, 0, 0, 5, 5, 0, 0],
87             [0, 0, 0, 0, 5, 0, 1, 0, 0, 5, 0],
88             [4, 3, 4, 0, 0, 0, 0, 5, 5, 0, 1],
89             [0, 0, 0, 4, 0, 4, 0, 0, 0, 0, 4],
90             [0, 0, 0, 2, 0, 2, 5, 0, 0, 1, 2],
91             [0, 0, 0, 0, 5, 0, 0, 0, 0, 4, 0],
92             [1, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0]]
93
94 # 替代上面的standEst, 该函数用SVD降维后的矩阵来计算评分
95 def svdEst(dataMat,user,simMeas,item,percentage):
96     n=shape(dataMat)[1]
97     simTotal=0.0;ratSimTotal=0.0
98     u,sigma,vt=la.svd(dataMat)
99     k=sigmaPct(sigma,percentage) #确定了k的值
100    sigmaK=mat(eye(k)*sigma[:k])  #构建对角矩阵
101    xformedItems=dataMat.T*u[:,:k]*sigmaK.I  #根据k的值将原始数据转换到k维空间(低维),xformedItems表示物品(
       item)在k维空间转换后的值
102    for j in range(n):
103        userRating=dataMat[user,j]
104        if userRating==0 or j==item:continue
105        similarity=simMeas(xformedItems[item,:].T,xformedItems[j,:].T) #计算物品item与物品j之间的相似度
106        simTotal+=similarity #对所有相似度求和
107        ratSimTotal+=similarity*userRating
108    if
109        simTotal==0:return 0
110    else
111        return ratSimTotal/simTotal
```

3.3 《PCA 算法和推广算法及其算法实现研究》——文雯，陈龙龙，刘有恒

摘要：主成分分析 (Principal Component Analysis 简称 PCA)，是多元统计方法，它通过一系列的线性变换对数据进行重构，在低维空间中重新表示数据。它通过这种方法往往能够从复杂的数据信息中获取最重要的元素和结构，去除其中包含的噪音和冗余。PCA 的降维方式相较于特征筛选保留了更多的原始信息，是一个高普适用的方法。在本文中，我们详细介绍了主成分分析的思想、不同角度的理论推导、PCA 求解过程的分析，更进一步地，我们同时研究了 PCA 算法自提出以来的一系列改进与推广，并结合自身的研究方向，最终决定以鲁棒主成分以及低秩表示等 PCA 的推广算法为例，介绍这些算法在机器学习，数据挖掘其他领域的作用。实验部分，我们以 PCA 算法与 LRR 算法相结合，在 Hopkins 155 运动分割数据集上对 156 个聚类任务进行实验，使用 PCA 算法对原始数据集进行降维，通过对降维前后聚类精度，算法执行时间等指标进行比较，从实验上验证 PCA 算法的有效性。本文代码链接:https://github.com/chenlonglong97/datamining.git

PCA 算法和推广算法及其算法实现研究

文雯 [1], 陈龙龙 [1]，刘有恒 [1]

[1]Team of Artificial Intelligence and Statistical Learning (AISLE), College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

主成分分析 (Principal Component Analysis 简称 PCA)，是多元统计方法，它通过一系列的线性变换对数据进行重构，在低维空间中重新表示数据。它通过这种方法往往能够从复杂的数据信息中获取最重要的元素和结构，去除其中包含的噪音和冗余。PCA 的降维方式相较于特征筛选保留了更多的原始信息，是一个高普适用的方法。

在本文中，我们详细介绍了主成分分析的思想、不同角度的理论推导、PCA 求解过程的分析，更进一步地，我们同时研究了 PCA 算法自提出以来的一系列改进与推广，并结合自身的研究方向，最终决定以鲁棒主成分以及低秩表示等 PCA 的推广算法为例，介绍这些算法在机器学习，数据挖掘其他领域的作用。实验部分，我们以 PCA 算法与 LRR 算法相结合，在 Hopkins 155 运动分割数据集上对 156 个聚类任务进行实验，使用 PCA 算法对原始数据集进行降维，通过对降维前后聚类精度，算法执行时间等指标进行比较，从实验上验证 PCA 算法的有效性。本文代码链接：https://github.com/chenlonglong97/datamining.git

关键词：PCA，鲁棒 PCA，数据降维

1 概况

1.1 选题说明（本章节作者：文雯）

当今社会是一个飞速发展的社会，科技发达、信息流通使人与人之间的交流越来越密切，大数据则是信息时代最大的产物。随着移动终端设备技术不断迭代更新，移动互联网应用的持续发展，以智能手机以及平板电脑为代表的智能移动终端日益普及，全球互联网用户数量持续提升。中商产业研究预计 2021 年全球互联网用户数量将达到 40.47 亿人，相应产生的用户网络行为数据规模是海量的，甚至无法用数字量化。

Facebook 最新提出元宇宙的概念其核心还是数据依赖。然而，数据的海量往往伴随着价值密度低、数据特征维度高、大量无标记数据、类型繁多等特点。给定一数据分析任务时，首先要对数据进行预处理以便后续分析处理，即去除冗余特征及干扰因素、提取主要信息、高维数据转换成低维数据等等。常见的做法是去均值、归一化、特征压缩、特征选择。

由于高维数据往往具有稀疏表示且存在冗余特征，增加了模型学习的难度以及计算复杂度。因此我们从特征压缩的角度来谈论数据处理，代表性算法为主成分分析算法，其主要思想是数据重构和特征约减，去掉携带信息较少的维度并保留主要的特征信息来对数据进行降维处理。

1.2 该算法基本原理（本章节作者：陈龙龙）

具体实践中，如果对像图片等高维数据直接进行处理，显然是非常困难的。这些困难包括分析上的困难，可视化几乎不可能做到，且从经济的角度来说，存储这些向量的代价是十分昂贵的。然而，高维数据中存在一些特有的性质。例如，高维数据经常是过载的，其中有许多数据是冗余的，它们可以通过其它维度相结合得到或解释。更进一步，在高维数据中，数据之间经常是相关的，也就是说在高维数据中存在比较低维

的本质结构 [1]。通过一些技术和变换从而得到能够表示高维数据的低维结构的方法被称作为降维技术。降维技术通过利用数据间的结构性和相关性，使得我们可以在理想条件下不损失数据信息的同时，可以获得高维数据一个更加紧凑的表示形式。具体的，我们可以把降维技术看作是一种压缩技术，类似于图像的的 jpeg 和音乐的 mp3 格式，它们就是常见的对图像和音乐进行压缩的算法。

在本篇文章中，我们所谈论研究的主成分分析算法，就是一种降维算法。主成分分析算法从由 Pearson 和 Hotelling 提出到现在已经有 100 多年的历史，但该算法仍然是数据压缩和数据可视化中广泛使用的技术。主成分分析算法也被用于识别高维数据中的简单模式、潜在因素和结构。在信号处理领域，主成分分析算法，以 Karhunen-Loeve transform 而被广泛应用。

主成分分析顾名思义，就是找出数据的主要方面，然后用数据的主要方面来代替原始的高维数据。这样在使用高维数据进行计算时，就可以直接利用高维数据的主成分进行计算，而不必采用原始数据，从而做到在保证原始数据性质的同时，降低了计算复杂度。主成分分析算法也可以看作一种非监督学习的方法，因为它的原始数据涉及一系列特征 $X_1, X_2, ..., X_p$，然而它所对应的响应数据确是不定的。简单的例子，就二维平面来说，如果数据集中的数据大多分布在某一坐标轴，X 轴或者 Y 轴，以都分布在 X 轴附近为例，这样我们就可以直接舍弃 Y 轴坐标，而几乎不损失数据的信息。当然如果数据分布在某一线性子空间，我们则可以通过旋转坐标轴的方式，线性变换，从而达到和分布在 X 轴或 Y 轴附近同样的效果。更一般的，推广到高维空间中，则是通过衍射等方式寻找一个超平面，来进行向二维平面上类似的操作。简而言之，主成分分析算法，就是在高维空间中找到一种合理的方法，在减少需求分析的指标同时，尽量减少原指标包含信息的损失，以达到对所收集数据进行全面分析的目的。

1.3 PCA 的推广（本章节作者：刘有恒）

由于理论研究以及实际问题的不断复杂化，机器学习领域所需要处理的样本数据越来越多，同时样本数据的维度也越来越大，这不可避免地导致了“维数灾难”。奇异值分解、主成分分析等降维算法通过去除噪声和不重要的特征，在一定程度上提升了高维数据存储以及运算的速度。在这些降维算法得到广泛应用的同时，关于其在高维数据降维和去噪方面的改进也层出不穷，本文以 PCA 算法为例，介绍一些应用广泛的推广算法。

1. 鲁棒主成分分析（Robust Principal Component Analysis,RPCA）
   PCA 算法用于数据降维和去噪，其基本假设是噪声服从高斯分布，即数据的噪声比较小，因此 PCA 算法对于噪声的鲁棒性不强，而 RPCA 基于 PCA 的思想，但其假设数据噪声是稀疏的，并且可能是很强的噪声。因此 RPCA 模型对噪声具有较强的鲁棒性，能被广泛应用于数据降维、恢复、去噪等。其基本假设是观测到的数据矩阵能被分解成一个低秩矩阵与一个稀疏矩阵之和，前者表示原始的，干净数据，后者代表噪声。

2. 低秩表示（Low Rank Representation,LRR）
   LRR 可以被认为是 RPCA 在子空间上的推广，RPCA 认为观测数据可以被一个低秩空间及稀疏矩阵之和，前者表示原始数据，后者表示噪声，而 LRR 认为观测数据内部有聚类的性质，即观测数据可以被多个独立的低秩子空间及稀疏矩阵之和表示，在这种假设下，LRR 不仅可以用来实现降维、去噪、还能用来实现聚类及图像分割等。

3. 张量鲁棒主成分分析（Tensor Robust Principal Component Analysis,TRPCA）
   TRPCA 旨在将 RPCA 算法推广到更高维的张量 Tensor 形式，其目的是从带噪声的观测数据中还原出真实的低秩张量，该方法往往从数据降维去噪以及算法运行时间上比 RPCA 有更好的效果。

2 PCA 基本思想及原理

2.1 PCA 思想（本章节作者：陈龙龙）

主成分分析方法，是一种被广泛使用的一种数据降维算法。其在数据压缩、消除冗余、数据噪音消除和数据可视化等领域都有广泛的应用。此部分为对主成分分析方法的基本思想进行介绍，后续将进行理论的分析，相应的扩展以及相应的应用进行具体介绍和应用。

2.1.1 PCA 算法的背景

在许多领域的研究与应用中，通常需要对反应事物本质的多个变量进行大量的观察，以从大量的数据中寻找潜在的规律，但与此同时将会产生大量高维数据，对分析和处理带来了巨大的困难。多变量大样本无疑会研究和应用提供丰富的信息，但也一定程度上增加了数据采集的工作量，更重要的是在绝大多数情况下，许多变量之间存在着一定的结构性与相关性，从而对分析带来了不便，增加了问题的复杂度 [1]。但如果仅仅对每一个指标进行单独分析，分析则是独立的，而不是综合的，这显然不符合实际情况。再者盲目的减少或删除指标会损失很多的有用信息，这样势必会产生不符实际或压根错误的结论。

因此，我们需要找到一个合理的方法，在减少需要分析的指标的同时，尽量相应的减少原指标包含信息的损失，以达到对所收集的高维大量数据进行有效全面分析的目的。由于各变量之间或维度之间存在一定的结构和相关关系，因此，我们有可能用较少的综合指标分别综合存在于各个变量中的各类信息。主成分分析方法就是用于解决此类问题的有效降维的方法。

2.1.2 PCA 的基本理念

主成分分析算法的主要思想是将 N 维的特征映射到 K 维上，这 K 维是全新的正交特征，它们即所谓的主成分，是在原有 N 维特征的基础上重新构造出来的 K 维特征。主成分分析方法其实就是从原始的高维空间中顺序的找出一组相互正交的坐标轴，显然新的坐标轴的选择与数据本身是密切相关的。这也就是为什么在进行应用主成分分析算法是都是采用奇异值分解的原因之一，即在奇异值分解中，其左右奇异值矩阵是严格单位正交的，且奇异值是按从大到小的顺序排列的。在坐标之间，第一个新坐标选择是原始数据中方差最大的方向，第二个新坐标轴选取是从与第一个坐标轴正交的平面中选择使得方差最大的，第三个新坐标轴则选择是与第一个新坐标轴与第二个新坐标轴正交平面中方差最大的那一个。按这种方式，依此类推，就可以得到 N 个这样的新坐标轴。如果，我们通过这种方式获得所需的新的坐标轴，我们不难发现，数据的大部分方差都包含在了新坐标轴的前面的 K 个新坐标中了，而后面的 $N-K$ 个新的坐标轴中所包含的方差几乎为 0。自然而然地，我们就可以删除或者忽略后面的 $N-K$ 个新的坐标轴，仅仅保留前面 K 个包含几乎绝大部分方差的坐标轴。其实不难想到，这样的做法就相当于只保留包含绝大部分方差的维度特征，而忽略了包含方差几乎为零的特征维度。即在几乎不损失原有信息的同时，找到了表示原始数据的低维结构，这就实现了对高维原始数据特征降维的操作，也就是主成分分析方法的设计思路和原理。

2.1.3 最大差异性主成分方向的获取思路

实际上，通过计算原始数据的协方差矩阵，然后计算协方差矩阵的特征值和对应的特征向量，然后对特征值进行从大大小的排列，依次选择特征值最大（也是方差最大的）的 K 个特征所对应的特征向量所组成对应矩阵。这样其实就可以将数据矩阵转换到新的空间中，实现了数据特征降维的目的。

相应的，由于得到协方差矩阵特征值与特征向量有两种不同的方法，特征值分解协方差矩阵，奇异值分解协方差矩阵，所以主成分分析方法算法也有两种实现方法：基于特征值分解协方差矩阵实现 PCA 算法、基于 SVD 分解协方差矩阵实现 PCA 算法。

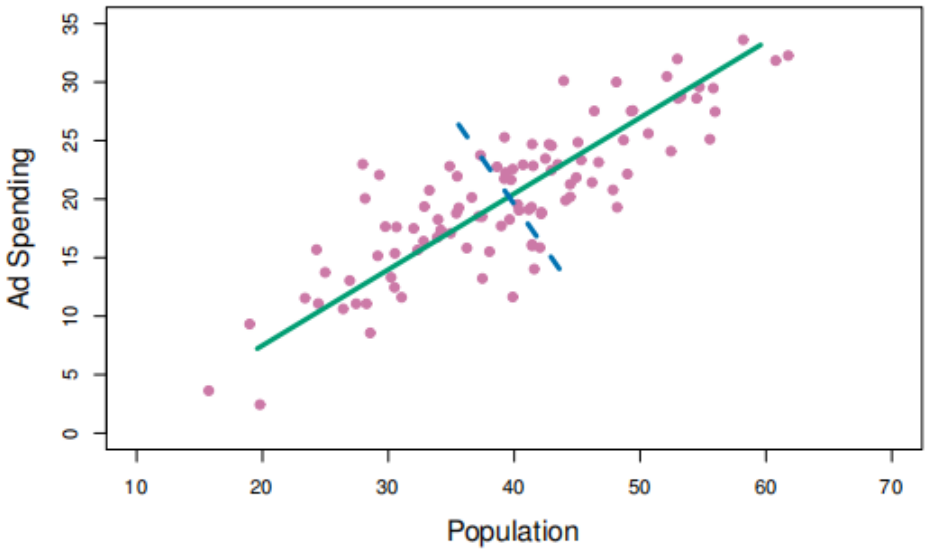


图 1: 100 个不同城市的人口规模和广告支出 [2]

2.1.4 PCA 的直观解释

主成分分析方法是一种对 n x p 数据矩阵进行降维的技术。数据的第一主分量方向是观测结果变化最大的方向。例如，如图 1，其展示了 100 个不同的城市的人口规模与广告支出的花费比例。其中，绿色实线表示数据的第一个主分量的方向。我们可以很清楚地看出在绿色实线方向上，数据有最大的方差。也就是说，我们把这 100 个观察点都投射在这条线上，投射结果将会产生最大的方差。如果我们把最大结果投射到另一条直线，也投射结果的方差将会很小。将一个点投射到一条直线上的方法只需要在直线上找到最接近这个点的位置即可。这样我们就可以得到结论，该数据集的主要信息可以用第一分量的信息近似表示。

主成分分析方法的另一个解释：第一主分量向量定义了尽可能接近数据的线。例如，在图 1 中，第一条主分量线最小化了每个点与该线之间垂直距离的平方和。也就是说，主成分分析方法就是选择一条直线，这条直线可以使观测数据投影后的结果尽可能的还原或接近原始数据。简言之，在二维平面中，主成分分析方法可以看作是把数据点进行固定，然后对坐标轴进行旋转操作，从而使得数据集都大致分布在某一条坐标轴上，X 轴或 Y 轴。以 X 轴为例，这样该数据集的全部信息就可以看作都是位于某一坐标轴上，而另一坐标轴上的信息就会出现要么为零要么很小的情况，这样我们就可以把 Y 轴坐标的信息全部丢弃，同时还不会产生太大的数据丢失的情况。

更一般的，上升到高维情况，坐标轴的旋转过程，即两个相互垂直的 X 轴与 Y 轴旋转，则可以看作相互垂直的平面变换的过程。这种表换的目的是使要处理的高维数据尽可能的落在某一超平面上，具体如图 2 左图所示。我们对图 2 左图的三维数据进行进行 PCA 操作使其保留两个维度的主成分信息，具体结果如图 2 右图所示。我们可以清清楚楚地看到，二维数据成功的捕获了三维数据的主要信息。具体的，在三维空间中彼此接近的橙色、绿色和青色的数据点，在进行 PCA 操作后的二维结构中，仍然保持了相对位置的不变。

2.1.5 白话 PCA

就像北宋著名诗人苏轼的诗《题西林壁》中，“横看成岭侧成峰，远近高低各不同。不识庐山真面目，只缘身在此山中。”。降维就像拍照，旨在寻找一个最好角度尽可能的捕捉原始镜像。照相和“街头仿真画”，就可以看作是把三维的数据降低到二维空间中，只要采用的角度完美，照出和画出的图像，我们仍然会有一种身临其境的感觉。但又如我们小时候听到的一个关于诚实的故事，一个男孩上绘画课，因坐在角落里，

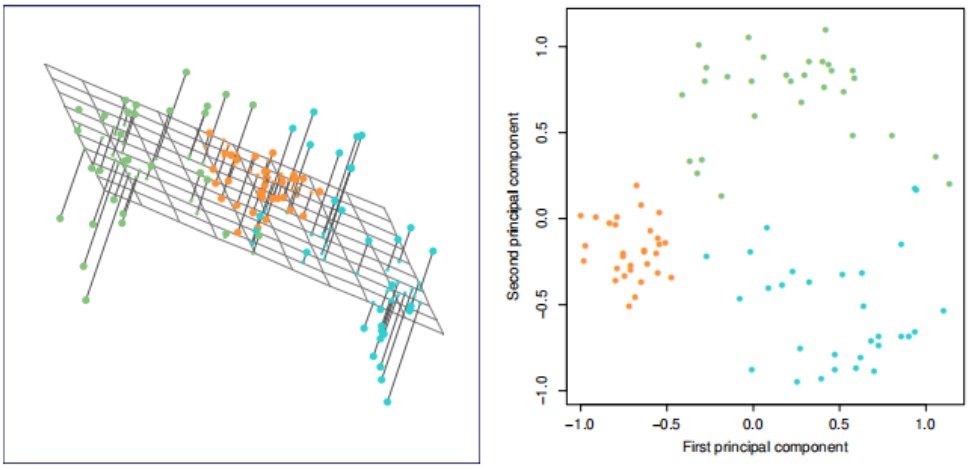


图 2: 90 个 3 维数据点的观测与可视化 [2]

而观察到要画的核桃，已经发生“变异”，从而导致画的结果极差。这就说明在我们有限的视野范围内观察一个事物，找对角度很关键。也不难发现，我们在一个很差的角度观察事物，会导致效果极差。即好的观察视角会保留原始事物的绝大多数信息，坏的角度会导致丢失很多信息，即我们无法照出或画出人眼可以分辨的照片或画作。

同理，主成分分析方法就可看作如何找到在一个低维空间的最好角度去去观察高维空间的观察方法，以使得观察的效果最好，即保留绝大多数有用信息。例如，对整数 150 进行两种形式的分解：

$$\begin{aligned} \mathbf{150} &= \mathbf{50} + \mathbf{50} + \mathbf{50} \\ \mathbf{150} &= \mathbf{120} + \mathbf{25} + \mathbf{5}. \end{aligned} \tag{1}$$

显然，第一种分解方式比较平均，去掉任意一个数，和分解之前的数值 150 差别都比较大；而第二中分解方式，不仅给出了分解，而且从大到小进行了排序，去掉 5 剩下 145，去掉 25 和 5 剩下 120，不难看出越是高位与原始数据就越接近。主成分分析方法则可以简单的理解为，寻找类似第二种分解方式的方法，这样使得其丢弃一些信息后仍对原始信息影响不大。更进一步，依据矩阵分解相关理论，更换坐标系，仍会从大到小的包含原始矩阵的信息，这样在降维的时候，我们就可以根据需求，保留前面的坐标轴，去掉尾巴的坐标轴，得到数据的降维表示。即数据还是那个数据，维数减少了，但信息却没有丢失多少，这就是主成分分析想要做的事。

2.2 PCA 理论分析（本章节作者：文雯）

现实世界中普遍存在着高维数据，而数据维度越高，则样本在空间中的分布越稀疏，容易引起维度灾难。缓解维数灾难的重要途经就是降维，PCA 算法是最常用的一种降维方法。它希望找到一个合适的超平面，能够对样本进行恰当的表达，使其具有最大可分性和最近重构性，即样本点到超平面的距离足够近以及在超平面上的投影尽可能分得开 [3]。理想情况下，降维接近原始数据的本征维。

2.2.1 PCA 算法的理论推导

下面我们分别阐述最大可分性和最近重构性背后的数学含义。

最大可分性希望在新坐标空间中投影后的样本点尽可能分开，保留原始样本之间的差异性。为了刻画“分的开”，这里引入了距离度量。若考虑任意两个样本点间的距离，其计算复杂度为 $\mathcal{O}(n^2)$；若考虑每个样

本点到样本中心的距离，其计算复杂度为 $\mathcal{O}(n)$。对比之下，我们采用后者距离度量方式。

假设样本点 $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathbb{R}^{d\times n}$ 均已进行中心化，有 $\bar{\boldsymbol{Z}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i = 0$，以及经投影变换后得到的标准正交基向量 $\{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{d'}\} \in \mathbb{R}^{d\times d'}$，即 $\|\boldsymbol{w}_i\|_2 = 1$，$\boldsymbol{w}_i^T\boldsymbol{w}_j = 0\,(i \neq j)$. 变换后的维度 $d' < d$, 相应地，原始样本在新的低维空间中表达为

$$\boldsymbol{Z} = \boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{X}, \tag{2}$$

其中 $\boldsymbol{z}_i = \boldsymbol{W}^T\boldsymbol{x}_i$. 与特征选择不同，降维后的样本属性是原空间中属性的线性组合，仍保留了原始属性的特征信息。考虑重构后的样本点 $\boldsymbol{z}_i$ 与样本中心的距离

$$\begin{aligned}\sum_{i=1}^{n}\|\boldsymbol{z}_i - \bar{\boldsymbol{Z}}\|_2^2 &= \sum_{i=1}^{n}\|\boldsymbol{z}_i\|_2^2 = \sum_{i=1}^{n}\left(\boldsymbol{x}_i^T\boldsymbol{W}\right)^T\boldsymbol{x}_i^T\boldsymbol{W}\\ &= \sum_{i=1}^{n}\boldsymbol{W}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{W}\\ &= \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}\end{aligned} \tag{3}$$

根据最大可分性，式 (3) 应该被最大化，其中 $\boldsymbol{X}\boldsymbol{X}^T$ 为协方差矩阵，有 PCA 优化的目标函数

$$\begin{aligned}&\max_{\mathbf{W}}\ \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}\\ &\text{s.t.}\quad \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}.\end{aligned} \tag{4}$$

最大可分性是从空间分布的角度出发，使投影变换后损失的信息量达到最小；而最近重构性则是从样本自身的角度出发，希望原样本点与重构后的样本点之间的距离尽可能地相近。虽然看问题的角度不同，但二者优化的目标函数是等价的。

下面我们介绍最近重构性的理论推导,首先与最大可分性的假设相同。样本点 x_i 在低维空间标准正交基下的投影为 $\boldsymbol{z}_i = (z_{i1}; z_{i2}; \ldots; z_{id'})$, 其中 $z_{ij} = \boldsymbol{w}_j^T\boldsymbol{x}_i$ 是 $\boldsymbol{x}_i$ 在低维坐标系下第 j 维的坐标. 由于 $\|\boldsymbol{w}_i\|_2 = 1$, 因此样本点向基向量投影的投影长度即为投影坐标。基于 $\boldsymbol{z}_i$ 来重构 $\boldsymbol{x}_i$, 得到样本点 $\hat{\boldsymbol{x}}_i = \sum_{j=1}^{d'} z_{ij}\boldsymbol{w}_j = \boldsymbol{W}\boldsymbol{z}_i$。相应地，样本点由高维空间向低维空间投影产生的信息损失为 $\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2$。为了最大程度保留原始样本携带的特征信息，需最小化信息损失。即优化的目标函数为

$$\min\sum_{i=1}^{n}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 \tag{5}$$

进一步分析目标函数有

$$\sum_{i=1}^{n}\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 = \sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{z}_i\|_2^2 = \|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{Z}\|_2^2 \tag{6}$$

其中 $\boldsymbol{z}_i = \boldsymbol{W}^T\boldsymbol{x}_i$，$\boldsymbol{Z} = \boldsymbol{W}^T\boldsymbol{X}$. 将 $\boldsymbol{Z}$ 代入等式 (7) 右项中得到

$$\begin{aligned}\|\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}\|_2^2 &= tr((\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})^T(\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}))\\ &= tr((\boldsymbol{X}^T - \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T)(\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}))\\ &= tr(\boldsymbol{X}^T\boldsymbol{X} - 2\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\\ &= tr(\boldsymbol{X}^T\boldsymbol{X} - 2\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X} + \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\\ &= tr(\boldsymbol{X}^T\boldsymbol{X} - \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\\ &= tr(\boldsymbol{X}^T\boldsymbol{X}) - tr(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\end{aligned} \tag{7}$$

上面第 4 个等式变换利用了 $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$. 注意 $\boldsymbol{W}$ 不是方阵，故 $\boldsymbol{W}\boldsymbol{W}^T \neq \boldsymbol{I}$ 无法消去。又由于

$tr(\boldsymbol{X}^T\boldsymbol{X})$ 是常数，因此优化的目标函数方程可以不考虑此项。另外

$$tr(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}) = tr((\boldsymbol{W}^T\boldsymbol{X})^T(\boldsymbol{W}^T\boldsymbol{X})) = tr((\boldsymbol{W}^T\boldsymbol{X})(\boldsymbol{W}^T\boldsymbol{X})^T) = tr(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}) \tag{8}$$

结合式 (7) 和 (8), 得到最后优化的目标函数

$$\begin{aligned} &\max_{W} tr(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W}) \\ &\text{s.t.} \quad \boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I} \end{aligned} \tag{9}$$

其中对式 (7) 最小化等价于最大化式 (8)。对比最大可分性和最近重构性的优化目标 (4) 和 (9)，易知二者优化的是同一目标函数方程，利用拉格朗格乘子法求解目标函数可得

$$\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{W} = \boldsymbol{\lambda}\boldsymbol{W} \tag{10}$$

对 $\boldsymbol{X}\boldsymbol{X}^T$ 进行特征值分解，将求得的特征值从小到大排序：$\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_d \geqslant 0$。前 d' 个特征值对应的特征向量即为 $\boldsymbol{W}^*$ 的各个列向量，利用 $\boldsymbol{W}^{*T}\boldsymbol{X}$ 进一步得到降维后的样本矩阵 $\boldsymbol{Z}$。

2.2.2 主成分

本节我们从降维后的样本矩阵出发，进一步讨论主成分的含义，并且给出了主成分分析应用的理论分析。

在上节中通过拉格朗日乘子法求得了 PCA 目标函数的特征值以及特征向量。相应地有，$\lambda_1 \geqslant \lambda_2 \geqslant \ldots \geqslant \lambda_d \geqslant 0$，以及第 i 个特征值对应的单位特征向量 $\boldsymbol{w}_i^T = (w_{i1}, w_{i2}, \ldots, w_{id}), i = 1, \ldots, d$ 相互正交。

下面，用最大特征值对应的特征向量对样本 $\boldsymbol{x}_i$ 进行线性组合，得到重构后第一维的坐标 $\boldsymbol{z}_{i1} = \boldsymbol{w}_1^T\boldsymbol{x}_i$，即第一主成分；若对样本点进行中心化，则是第一主成分得分

$$z_{i1} = w_{11}x_{i1} + w_{21}x_{i2} + \ldots + w_{d1}x_{id} \tag{11}$$

因为第一主成分 z_{i1} 是由最大特征值的特征向量对样本点的线性组合而来,因此它解释原始变量 $(x_{i1}, x_{i2}, \ldots, x_{id})$ 的能力最强, 而 $z_{i2}, z_{i3}, \ldots, z_{id}$ 解释能力依次递减。相应地，z_{i1} 包含原始数据信息的比例为

$$\lambda_i / \sum_{i=1}^{d} \lambda_i \tag{12}$$

式 (12) 也称主成分 z_{i1} 的贡献率。主成分分析的本质就是降维，因此需要选取合适的前 $d'(d' < d)$ 个主成分使降维后产生的的信息损失达到最小，即计算前 d' 个主成分累计贡献之和的总体占比

$$\sum_{i=1}^{d'} \lambda_i \,/\, \sum_{i=1}^{d} \lambda_i \tag{13}$$

式 (13) 也称为主成分 $z_{i1}, z_{i2}, \ldots, z_{id'}$ 的累计贡献率，它表明 $z_{i1}, z_{i2}, \ldots, z_{id'}$ 解释 $x_{i1}, x_{i2}, \ldots, x_{id}$ 的能力。通常取（相对于 d）较小的 d', 使得累计贡献率达到较高的百分比。而低维空间的维数 d' 有时是由用户事先指定，或者利用交叉验证来选取合适的值。

此外，降维必然会舍弃一部分样本信息，但我们应该辩证的看待这个事实。一方面，舍弃部分样本信息往往会增大样本的采样密度；另一方面，当数据受到噪声影响时，最小特征值对应的特征向量往往与噪声相关，因此，舍弃部分信息在一定程度上可以起到去噪的效果 [3]。

2.2.3 从应用的角度看待主成分分析

在实际应用中，需要注意以下几点：首先要保证提取的前几个主成分的累计贡献率达到较高的水平，如果原始数据特征之间具有较高的相关性，则选取少数几个主成分其累计贡献率就能达到较高的水平。其次对被提取的主成分必须能给出符合实际背景意义的解释，困难之处也在于此。

然而，主成分解释重构样本点各属性之间的关系、本身的含义存在着模糊性，不如原始变量的意义那么清晰、明确，这也是降维过程中不可避免的代价。因此，为了降维的有效性，重构样本的维数应明显小于原始变量的。

下面参考统计分析的一简单实例 [4]。在制定服装标准的过程，对 128 名成年男子的身材进行了测量，每人测得的指标中含有六项：身高 (d_1)、坐高 (d_2)、胸围 (d_3)、臂长 (d_4)、肋围 (d_5) 和腰围 (d_5)。经主成分分析降维后有表1

特征向量	z_1	z_2	z_3
d_1：身高	0.469	-0.365	0.092
d_2：坐高	0.404	-0.397	0.613
d_3：胸围	0.394	0.397	-0.279
d_4：臂长	0.408	-0.365	-0.705
d_5：肋围	0.337	0.569	0.164
d_5：腰围	0.427	0.308	0.119
特征值	3.287	1.406	0.459
贡献率	0.548	0.234	0.077
累计贡献率	0.548	0.782	0.859

表 1: 前三个特征值、特征向量及贡献率 [4]

根据表1，自然的得出重构后前三个主成分

$$
\begin{aligned}
z_1 = &\ 0.469\,d_1 + 0.404\,d_2 + 0.394\,d_3 + 0.408\,d_4 + 0.337\,d_5 + 0.427\,d_6 \\
z_2 = &\ -0.365\,d_1 - 0.397\,d_2 + 0.397\,d_3 - 0.365\,d_4 + 0.569\,d_5 + 0.308\,d_6 \\
z_3 = &\ 0.092\,d_1 + 0.613\,d_2 - 0.279\,d_3 - 0.705\,d_4 + 0.164\,d_5 + 0.119\,d_6
\end{aligned}
$$

从表达式中可以看出 z_1 对所有属性变量都有近似相等的正载荷系数，故第一主成分表示的是身材大小成分。第二主成分 z_2 在 x_3, x_5, x_6 上面有中等程度的正载荷，具有同样的变化方向，而在 d_1, d_2, d_4 上有中等程度的负载荷，与其属性变化方向相反，总体体现人的胖瘦因素，我们称其叫形状成分。第三主成分 z_3 在 d_2 上有较大的正载荷，在 d_4 上有较大的负载荷且在其余特征变量上的载荷都较小，可以称第三主成分为臂长成分。

此外，由表1看到，前三个主成分的立即贡献率已经达到 85.9%，但第三个主成分贡献率不如前两个且实际意义不重要，因此一般考虑前两个主成分足以。

另外注意，根据数值在解释主成分时，带有人为主观观念，但只要所作的解释符合产生的现象且逻辑通顺即可，因此主成分的解释并非固定不变。

2.3 PCA 算法拓展（本章节作者：刘有恒）

2.3.1 鲁棒主成分分析

上述介绍了使用主成分分析算法作为高维数据降维方法的基本原理和应用，然而 PCA 作为数据降维的假设是观测到的数据是无噪，或者噪声服从高斯分布，即数据噪声比较小。在实际应用中数据的噪声往往是稀疏（相对于原始数据而言），但可能是比较强的噪声，以图 3 为例，图 3（a）是在某机场拍摄的包含 200 帧的视频序列中的某一帧，（b）为该原始帧中在整个视频中静止的部分，该部分在整个视频帧中可以被认为是低秩的。(c) 为该原始帧在整个视频中运动的部分，该部分可以被认为是噪声，而且因为该运动部分占整个图像的比重很小，故这部分是稀疏的。

(a) Original frames (b) Low-rank $\hat{L}$ (c) Sparse $\hat{S}$

图 3: 视频背景模型 [5]

在实际应用中，很多数据矩阵往往是低秩的（如人脸图像左右对称具有低秩性），在理想状态下，数据矩阵列与列（假定一列表示一个样本）之间应当具有极强的相似性，整个矩阵应当是低秩的，如图 3（b）。但由于数据噪声的存在，如图 3（c），破坏了这种低秩性。因此为了恢复出数据矩阵的低秩结构，可以将观测到的数据矩阵 X 分解成一个低秩矩阵 D 与稀疏矩阵 E 之和，分别对应于图 2 中的（b）与（c）。故 RPCA 可以描述为以下的最优化问题

$$\begin{aligned} &\min_{\boldsymbol{D},\boldsymbol{E}} rank(\boldsymbol{D}) + ||\boldsymbol{E}||_0 \\ &\text{s.t.} \boldsymbol{X} = \boldsymbol{D} + \boldsymbol{E} \end{aligned} \tag{14}$$

其中 $rank(\boldsymbol{D})$ 即为矩阵 $\boldsymbol{D}$ 的秩，该项用来约束矩阵 $\boldsymbol{D}$ 低秩，$||\boldsymbol{E}||_0$ 为矩阵 $\boldsymbol{E}$ 的 0 范数，该项用来约束矩阵 $\boldsymbol{E}$ 稀疏。然而矩阵秩和矩阵 0 范数都是非连续的，关于其的求解问题都是非凸的，难以求解，在 RPCA 的实际求解中，我们考虑以下的优化问题

$$\begin{aligned} &\min_{\boldsymbol{D},\boldsymbol{E}} ||\boldsymbol{D}||_* + ||\boldsymbol{E}||_1 \\ &\text{s.t.} \boldsymbol{X} = \boldsymbol{D} + \boldsymbol{E} \end{aligned} \tag{15}$$

上式中 $||\boldsymbol{D}||_*$ 为矩阵 $\boldsymbol{D}$ 的核范数，即矩阵 $\boldsymbol{D}$ 的奇异值之和，该项可以作为 $rank(\boldsymbol{D})$ 的最紧凸松弛，同理上式中 $||\boldsymbol{E}||_1$, 为矩阵 $\boldsymbol{E}$ 的 1 范数，该项可以作为 $||\boldsymbol{E}||_0$ 的最紧凸松弛。

对于 RPCA 模型，即式(14)的求解，可以使用经典的增广拉格朗日乘子法 ALM(Augmented Lagrangian method) 进行求解

2.3.2 低秩表示

前面部分的内容介绍了 PCA 算法与 RPCA 算法，前者是高维数据降维的经典算法，而后者作为 PCA 算法的推广，对于噪声有着更强的鲁棒性。而本小节介绍的低秩表示算法可以认为是 RPCA 在子空间上的

推广，RPCA 背后其实是假设高维数据样本可以表达为一个低维子空间与稀疏噪声的和，数学形式就是将高维数据代表的矩阵分解为一个低秩矩阵与一个稀疏矩阵的和。而 LRR 低秩分解在 RPCA 假设的基础上认为高维数据样本可以表达为多个线性低维子空间与噪声的和，我们以 A 表示空间中的字典集，LRR 模型可以表达如下

$$\begin{aligned} &\min_{\boldsymbol{Z},\boldsymbol{E}} rank(\boldsymbol{Z}) + ||\boldsymbol{E}||_l \\ &\text{s.t.}\boldsymbol{X} = \boldsymbol{A}\boldsymbol{Z} + \boldsymbol{E} \end{aligned} \tag{16}$$

其中 X 观测到的带有噪声的高维数据矩阵，其中每一列表示一个样本，每一行表示一个维度。A 表示空间中的字典集，Z 高维数据在该字典集下表示矩阵，E 为噪声矩阵，这里为了使 LRR 模型对于不同的噪声类型都具有鲁棒性，这里模型中对于噪声 E 的约束范数可以根据问题的先验信息来确定，部分结论如下

1. 若先验信息表明数据为小的高斯噪声
   在模型中噪声 E 使用 $||\boldsymbol{E}||_F^2$ 进行表征，其中 $||\boldsymbol{E}||_F$ 为矩阵 F-范数

2. 若先验信息表明数据为随机的损坏
   在模型中噪声 E 使用 $||\boldsymbol{E}||_1$ 进行表征，其中 $||\boldsymbol{E}||_1$ 为矩阵 1-范数

3. 若先验信息表明数据为异常值或者是特定样本的直接损坏（列缺失）
   在模型中噪声 E 使用 $||\boldsymbol{E}||_{2,1}$ 进行表征，其中 $||\boldsymbol{E}||_{2,1}$ 为矩阵 2,1 范数，即先对矩阵每一列求向量-2 范数，再对所得的向量求向量的 1-范数

观察 LRR 模型式（16）与 RPCA 模型式（14），不难发现，当 LRR 模型中的字典集 A 取为单位矩阵 I，对于噪声 E 的表征取 1 范数时，LRR 模型退化为 RPCA 模型，这也解释了为什么 LRR 是 RPCA 的一个推广。同时，LRR 是基于字典集学习的任务，即根据字典集合 A 选取的不同，可以使用 LRR 模型去解决机器学习领域中的不同的任务，例如降维、去噪、鲁棒图像恢复、图像分割、聚类等。本文即后续实验主要使用 LRR 模型去解决机器学习中的聚类问题，故选取观测到的数据集 X 本身作为 LRR 模型中的字典集。同时，后续的实验中我们选取矩阵 2,1 范数作为对误差 E 的表征，故整个 LRR 的求解模型如下

$$\begin{aligned} &\min_{\boldsymbol{Z},\boldsymbol{E}} ||\boldsymbol{Z}||_* + ||\boldsymbol{E}||_{2,1} \\ &\text{s.t.}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{Z} + \boldsymbol{E} \end{aligned} \tag{17}$$

其中 $||\boldsymbol{Z}||_*$ 为 $rank(Z)$ 的最紧凸松弛，即保证模型是凸问题，后续 2.33 小节的实验中 LRR 的部分都基于此模型进行，同时对于该模型的优化求解也可以使用增广拉格朗日乘子法（ALM）算法进行优化，这里不再赘述。

2.3.3 PCA 及 LRR 实验

本文基于 PCA 算法与 LRR 算法相结合进行实验，其中 PCA 算法主要对数据进行预处理，通过主成分分析降低高维数据维度，以达到在后续求解过程中加快算法收敛速度的效果，而 LRR 算法作为一种子空间聚类算法 [6]，旨在通过对数据集降维前后使用 LRR 进行聚类的效果进行比较，从而分析 PCA 算法的效果，并对理论进行验证。

该实验采用的数据集为 Hopkins 155 运动分割数据集，采用该数据集的目的是该数据集噪声很小，能够更好地比较由 PCA 算法降维前后 LRR 算法聚类的果，该运动分割数据集一共包含 156 个聚类任务，每个任务采样数据样本数为 39-556（均值 295.7），这些采样样本分别来自 2-3（均值 2.3）个不同的聚类，整个数据集中的数据维度为 31-201（均值 59.7）。

使用 PCA 进行高维数据降维的一个很重要的问题是如何确定降维后的数据维度，通过前面章节的理论分析可知，将数据维度降到多少的维度是我们人为决定的，若指定的据降维后的维度比较大，则经过 PCA

算法进行数据预处理之后对后续算法的收敛速度并不明显，若指定的数据降维之后的维度比较小，那么使用 PCA 进行降维后对数据的信息丢失比较严重，对后续算法的求解精度会造成影响。所以这里引入本征维度的概念，本征维度可以理解为数据集包含我们所求解问题所有必要信息的维度，因此使用 PCA 算法将高维数据降低到其本征维度时，能在不影响后续求解精度的情况下提升后续的求解效率。

本文使用 PCA 与 LRR 算法结合分解将 Hopkins 155 数据集降低到 4n(本征维度，n 为该任务中聚类的大小) 与 2 的维度，对 PCA 算法的效果进行对比，主要评价指标为聚类精度和算法运行时间，本文自行编写代码实验运行结果如表 2

	未使用 PCA 降维	使用 PCA 降到 4n 维	使用 PCA 降到 2 维
聚类精度最小值 (%)	52.78	52.78	42.24
聚类精度最大值 (%)	100	100	100
聚类精度平均值 (%)	96.07	95.09	68.74
聚类精度标准差	8.49	8.49	14.4
运行时间最大值 (s)	0.5130	0.1902	0.0655
运行时间最小值 (s)	0.0344	0.0140	0.0063
运行时间平均值 (s)	0.2491	0.0908	0.0262

表 2: 使用 PCA 与 LRR 算法对 Hopkins 155 数据集进行实验结果

由上述实验不难得出以下结论：

1. 使用 PCA 预处理方法对数据进行合适的预处理能在一定程度上实现高维数据的可视化，例如将高维数据降低到 2 至 3 维时可以通过平面坐标系或空间坐标系观察降维后的数据分布，降维后的数据可能仍能保持比较好的分类或者聚类等机器学习性质，同时 PCA 算法降维可以加快后续算法的收敛速度。

2.PCA 降维过程中会造成数据信息的丢失，因此使用 PCA 对高维数据进行降维的时候要根据数据分布的先验知识，选择合适的维度进行降维，在求解速度于求解精度之间做一个平衡。

2.3.4 张量鲁棒主成分分析

张量鲁棒主成分分析是基于 RPCA 的高阶推广 [5]，近几年才被提出并得到发展。本文只作简单介绍。TRPCA 将 RPCA 的二维矩阵形式推广到高阶张量的形式，TPCA 同样在降维、数据去噪等领域拥有广泛应用，其基本思想还是基于矩阵形式的 RPCA，即将观测到的高阶数据张量 $\mathcal{X}$ 分解成一个低秩张量 $\mathcal{L}$ 和一个稀疏张量 ξ 之和，换句话说，即恢复出张量的低秩性质。那么基于 RPCA 的思想，TRPCA 模型可以表示如下

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\xi}} rank(\boldsymbol{\mathcal{L}}) + ||\boldsymbol{\xi}||_0 \\ &\text{s.t.}\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\xi} \end{aligned} \tag{18}$$

同样，上述问题是非凸的，我们同样使用张量核范数作为秩的凸松弛，使用张量的 1-范数作为 0-范数的松弛，得到的模型如下

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\xi}} ||\boldsymbol{\mathcal{L}}||_* + ||\boldsymbol{\xi}||_1 \\ &\text{s.t.}\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\xi} \end{aligned} \tag{19}$$

不难发现，TRPCA 模型与 RPCA 模型从形式上基本一致，即 TRPCA 只是作为 RPCA 向高维的拓展，但是 TRPCA 相较于 RPCA 一个很大的问题就是，RPCA 考虑二维的数据矩阵，在模型中是对于矩阵的秩以及矩阵的核范数进行约束，而对于矩阵的秩和核范数已经有了十分完备的理论推导与数学分析，矩

阵的秩即为最大线性无关组的个数，也等价于对矩阵做 SVD 分解，其非零奇异值个数，而矩阵的核范数为奇异值绝对值之和。同时通过这个性质可以证明矩阵核范数在一定情况下是矩阵秩的最小闭包，这里由于篇幅关系不再赘述。

总之，对于矩阵而言，其秩和核范数都有着严格的理论定义，而对于张量而言，关于张量秩 (Tensor Rank)，张量核范数 (Tensor Nuclear Norm) 以及张量 SVD 分解 (Tensor SVD) 都没有严格的定义。近几年对于张量秩的研究很多，针对不同问题提出的张量秩的形式层出不穷，例如张量的 CP rank、张量的塔克秩 (Tucker rank) 以及张量的管秩（Tubal Rank）等等。基于这些定义就能针对实际问题对 TRPCA 模型进行优化并能应用到具体问题，更详细的有关 TRPCA 问题的信息可以参考本文 3.2 小节提到的文章。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：文雯）

在神经网络受着算力、硬件等约束时，PCA 是最经典的降维方法。现今，数据量激增的背景下，处理高维数据是任何一种机器学习算法首先要思考的问题。通过降维对数据特征进行了重构，使重构数据的每一维度特征都包含着丰富的信息，保存的内容更完整。这样的低维数据也使得模型利用机器学习算法学习起来更容易，更能发掘到数据内部的深刻含义以及附加信息。神经网络有强大的能力处理复杂任务，然而并不知道它内部基于何种原则进行降维的、模型在数据中学到了什么。相比之下，主成分分析降维方法的理论体系更加完备，数据重构后的每一维度特征都能被很好的解释。

基于上述的考量，我们详尽的阐述了 PCA 的背景、具体的理论推导、给出了一个实例来分析主成分的含义以及根据此实例说明了贡献率及主成分的选取并通过实验直观的感受 PCA 的降维过程，之后我们进一步介绍 PCA 的拓展，鲁棒 PCA 以及鲁棒 PCA 的实际应用。每一部分我们都做了详细的分工，文章内容都加入了自己对于 PCA 的理解。

3.2 所参考主要资源（本章节作者：刘有恒）

本文关于 PCA 主成分分析的基本思想以及 PCA 模型的数学推导来自周志华的机器学习 [3] 以及李航的统计学习方法 [7]。而 PCA 的推广方面，鲁棒主成分分析 RPCA 主要参考了一些 CSDN 博客进行总结，而低秩表示 LRR 主要参考了 G.Liu 等人于 2013 年发表在计算机视觉顶级期刊 IEEE Transactions on Pattern Analysis and Machine Intelligence（简称 PAMI）的论文，名为 Robust Recovery of Subspace Structures by Low-Rank Representation, 而张量鲁棒主成分分析是近几年来该方向比较前沿的研究, 这里我们参考了一篇由 C. Lu 等人于 2018 年发表在该顶级期刊的论文, 名为 Tensor Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Tensors via Convex Optimization。最后，本文有关实验部分，PCA 与 LRR 算法的代码由小组成员自己编写。代码见附录。

3.3 代码撰写的构思和体会（本章节作者：陈龙龙）

“纸上得来终觉浅，觉知此事要躬行”，为了进一步的加深对主成分分析算法的理解，并提升代码应用和实际问题解决能力，本文结合我们当前的研究方向，具体设计并进行了主程序分析算法及其拓展和低秩表示聚类算法相互结合的实验，一步一步调试和测试了算法的实现过程。我们采用具有很小噪声的 Hopkins 运动分割数据集，以至于我们能更好的对比使用主成分分析算法前后在聚类效果上的差异。我们从两方面来对实验结果进行评估。首先，分别求出未使用 PCA 算法进行降维、使用 PCA 算法降到 4n 维和使用 PCA 算法降到 2 维时聚类精度的最小值、聚类精度的最大值、聚类精度的平均值以及聚类精度的标准差进

行比较分析。然后，从算法的运行时间上展现 PCA 算法的优越性，具体指标分为运行时间的最大值、最小值以及平均值，具体实验结果与描述详见表 2。

通过结合具体的算法进行对主成分分析算法实验，使我们对主主成分分析算法的原理有了更加深刻的理解，对其具体的推导过程有了更直观的洞悉，并且对如何应用 PCA 算法解决实际问题有了一定的了解，对我们的后续研究学习给予了很大的帮助。

3.4 人员分工（本章节作者：陈龙龙）

为了尽量让每个小组成员都对主成分分析算法从头到尾有一个全面的把握，对整篇文章的各个部分都熟悉。在进行分工时，尽量让小组的每个成员都参与到文章的各部分撰写工作中。具体分工为，文雯负责选题说明、理论分析以及对论文构思的总结；陈龙龙负责算法基本原理说明、算法基本思想的描述以及代码构思；刘有恒负责 PCA 算法推广以及 PCA 算法的扩展与实验具体过程的阐述。

参考文献

[1] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambrige University Press, 2020.

[2] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Application in R*. Springer, 2014.

[3] 周志华. 机器学习 *[M]*. 清华大学出版社, 北京, 2016.

[4] 王学民. 应用多元分析. 应用多元分析, 2004.

[5] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5249–5257, 2016.

[6] Guangcan Liu, Zhouchen Lin, Shuicheng Yan, Ju Sun, Yong Yu, and Yi Ma. Robust recovery of subspace structures by low-rank representation. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):171–184, 2012.

[7] 李航. 统计学习方法. 清华大学出版社, 北京, 2012.

4 附录

4.1 使用 PCA 和 LRR 对 Hopkins 155 数据集测试代码（本章节作者：刘有恒）

```
代码来源：小组成员自行编写
PCA部分代码
%使用Hopkins数据集测试，使用PCA降维到4F或2前后子空间聚类效果
%------------------------导入数据集
%close all;clear;clc;
function [Err3]=PCAtext2(p)
load Hopkins155
fea=data(p).X;
gnd=data(p).ids;
```

```
N=length(gnd);%样本个数
n=max(gnd);%子空间类别数
r=4*n;%降维后的维数
lambda=1;
%-------------------------初始子空间聚类
 %[Err]=Text(fea,gnd,lambda);
%-------------------------主成分分析
fea=fea';
[coeff,score,latent]=pca(fea);
%-------------------------PCA降维到4n后子空间聚类
% fea4r=fea*coeff(:,1:r);
% fea4r=fea4r';
% [Err2]=Text(fea4r,gnd,lambda);
%-------------------------PCA降维到2后子空间聚类
fea2=fea*coeff(:,1:2);
fea2=fea2';
[Err3]=Text(fea2,gnd,lambda);
%-----------------------------------------------
%使用PCA对Hopkins 155测试降维前后运行时间
close all;clear;clc;
load HopKins155
lambda=4;%子空间聚类算法参数
Time=zeros(length(data),1);
Time2=zeros(length(data),1);
Time4n=zeros(length(data),1);
%-------------------------降维到2维运行时间
for i=1:156
    %初始化数据
    fea=data(i).X;%样本矩阵D*N
    gnd=data(i).ids;%真实标签N*1
    N=length(gnd);%样本数目
    n=max(gnd);%子空间类别
    r=2;%降维后的维数
    %PCA降维
    fea=fea';
    [coeff,score,latent]=pca(fea);
    fea=fea*coeff(:,1:r);
    fea=fea';
    tic;%tic
    [Err3]=Text(fea,gnd,lambda);
    Time2(i)=toc;
    disp(['降维到2第',num2str(i),'次循环运行时间: ',num2str(Time2(i))]);

end
%-------------------------未降维运行时间
for i=1:156
    %初始化数据
    fea=data(i).X;%样本矩阵D*N
    gnd=data(i).ids;%真实标签N*1
    N=length(gnd);%样本数目
    n=max(gnd);%子空间类别
    r=4*n;%降维后的维数
    %PCA降维
    fea=fea';
    [coeff,score,latent]=pca(fea);
    fea=fea*coeff(:,1:r);
    fea=fea';
    tic;%tic
    [Err3]=Text(fea,gnd,lambda);
```

```
69      Time(i)=toc;
70      disp(['未降维第',num2str(i),'次循环运行时间: ',num2str(Time(i))]);
71  end
72  %-----------------------------降维到4n运行时间
73  for i=1:156
74      %初始化数据
75      fea=data(i).X;%样本矩阵D*N
76      gnd=data(i).ids;%真实标签N*1
77      N=length(gnd);%样本数目
78      n=max(gnd);%子空间类别
79      r=4*n;%降维后的维数
80      %PCA降维
81      fea=fea';
82      [coeff,score,latent]=pca(fea);
83      fea=fea*coeff(:,1:r);
84      fea=fea';
85      tic;%tic
86      [Err3]=Text(fea,gnd,lambda);
87      Time4n(i)=toc;
88      disp(['降维到4n第',num2str(i),'次循环运行时间: ',num2str(Time4n(i))]);
89  end
90  LRR部分代码
91  function [missrate,C] = LRR(X,r,lambda,s)
92  n = max(s);
93  Xp = DataProjection(X,r);
94  C= solve_lrr(Xp,Xp,lambda);
95  %post processing
96      [U,S,V] = svd(C,'econ');
97      S = diag(S);
98      r = sum(S>1e-4*S(1));
99      U = U(:,1:r);S = S(1:r);
100     U = U*diag(sqrt(S));
101     U = normr(U);
102     L = (U*U').^4;
103 % spectral clustering
104 D = diag(1./sqrt(sum(L,2)));
105 L = D*L*D;
106 [U,S,V] = svd(L);
107 V = U(:,1:n);
108 V = D*V;
109 idx = kmeans(V,n,'emptyaction','singleton','replicates',20,'display','off');
110 % idx = SpectralClustering(Z,n);
111 missrate = Misclassification(idx,s);
112 %missrate = compacc_ce(idx,s);
113 %missrate=compacc_ce(idx,s);
114 %-----------------------------------------------
115 #使用ALM算法求解LRR问题
116 function [Z,E] = inexact_alm_lrr_l21(X,A,lambda,display)
117 if nargin<4
118     display = false;
119 end
120
121 tol = 1e-8;
122 maxIter = 1e6;
123 [d n] = size(X);
124 m = size(A,2);
125 rho = 1.1;
126 max_mu = 1e10;
127 %max_mu=1e6;
```

```
mu = 1e-6;
atx = A'*X;
inv_a = inv(A'*A+eye(m));
%% Initializing optimization variables
% intialize
J = zeros(m,n);
Z = zeros(m,n);
E = sparse(d,n);

Y1 = zeros(d,n);
Y2 = zeros(m,n);
%% Start main loop
iter = 0;
if display
    disp(['initial,rank=' num2str(rank(Z))]);
end
while iter<maxIter
    iter = iter + 1;
    %update J
    temp = Z + Y2/mu;
    [U,sigma,V] = svd(temp,'econ');
    sigma = diag(sigma);
    svp = length(find(sigma>1/mu));
    if svp>=1
        sigma = sigma(1:svp)-1/mu;
    else
        svp = 1;
        sigma = 0;
    end
    J = U(:,1:svp)*diag(sigma)*V(:,1:svp)';
    %udpate Z
    Z = inv_a*(atx-A'*E+J+(A'*Y1-Y2)/mu);
    %update E
    xmaz = X-A*Z;
    temp = xmaz+Y1/mu;
    E = solve_l1l2(temp,lambda/mu);

    leq1 = xmaz-E;
    leq2 = Z-J;
    stopC = max(max(max(abs(leq1))),max(max(abs(leq2))));
    if display && (iter==1 || mod(iter,50)==0 || stopC<tol)clc
        disp(['iter ' num2str(iter) ',mu=' num2str(mu,'%2.1e') ...
            ',rank=' num2str(rank(Z,1e-3*norm(Z,2))) ',stopALM=' num2str(stopC,'%2.3e')]);
    end
    if stopC<tol
        break;
    else
        Y1 = Y1 + mu*leq1;
        Y2 = Y2 + mu*leq2;
        mu = min(max_mu,mu*rho);
    end
end
```

4

神经网络专题

在 22 篇文章中，有 9 篇是关于神经网络的。如果考虑 GNN 也是以神经网络来实现，则有 10 篇之多，占了一半内容。这和当前 ANN 的大流行趋势是保持一致的。尽管神经网络在我的课程体系中只有 3 个学时的内容，考虑到受欢迎程度，还是挑选了 BP，CNN 和 Attention 的三篇。

– Jingbo Xia

4.1 《BP 神经网络及其算法实现研究》——王鑫源，招欣乐，张俊瑛

摘要：反向传播 (Backpropagation，缩写为 BP) 是“误差反向传播”的简称，是一种与最优化方法 (如梯度下降法) 结合使用的，用来训练人工神经网络的常见方法。本文主要介绍了 BP 神经网络的有关内容，首先对误差逆传播算法的产生和发展做了简要介绍，之后推演了 BP 神经网络的原理与算法流程。BP 神经网络在训练有三个重要的过程: 正向传播过程、反向传播过程以及权重更新过程，本文结合两个实例实现了 BP 神经网络的搭建，并对神经网络进行训练和测试，实战部分包括手写字体的识别与简单二分类问题。通过本次学习，我们更加深刻体会到深度学习框架在算法学习中的角色，但框架只是工具，更重要的是我们本身对算法的理解。

BP 神经网络及其算法实现研究

王鑫源 [1], 招欣乐 [2]，张俊瑛 [3]

[1] College of Life Science And Technology, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[2,3]College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

反向传播（Backpropagation，缩写为 BP）是“误差反向传播”的简称，是一种与最优化方法（如梯度下降法）结合使用的，用来训练人工神经网络的常见方法。本文主要介绍了 BP 神经网络的有关内容，首先对误差逆传播算法的产生和发展做了简要介绍，之后推演了 BP 神经网络的原理与算法流程。BP 神经网络在训练有三个重要的过程：正向传播过程、反向传播过程以及权重更新过程，本文结合两个实例实现了 BP 神经网络的搭建，并对神经网络进行训练和测试，实战部分包括手写字体的识别与简单二分类问题。通过本次学习，我们更加深刻体会到深度学习框架在算法学习中的角色，但框架只是工具，更重要的是我们本身对算法的理解。

关键词：BP 神经网络，前馈神经网络，深度学习

1 概况

1.1 选题说明（本章节作者：张俊瑛）

神经网络在学习复杂的非线性假设上是一种很好的算法，尤其是特征空间很大时，也能轻松应对。神经网络诞生于 1943 年沃伦 · 麦卡洛克和沃尔特 · 皮茨发表的《神经活动中内在思想的逻辑演算》一文 [1]，当兴奋超过神经元的阈值时，它就会触发脉冲。作者从上述情况中抽象出一个简单的数学模型，在论文中首次提出了人工神经网络的概念以及沿用至今的 “M-P 神经元模型”。在模型中，当前神经元接收其他 n 个神经元的输入，并连接相应的不同权重值。神经元将接收到的总输入值与阈值进行比较后，通过激活函数得到新的输出。之后 “感知器” 的提出，掀起了神经网络第一次研究热潮，感知器由两层神经元组成，输入层接收外部信息，输出层是 M-P 神经元，可以实现逻辑和、或和非操作。但由于其只能解决简单的线性问题，学习能力有限，研究陷于低潮 [2]。

要解决非线性的问题就要使用多层网络，即输入层与输出层之间存在隐层或称为隐含层，隐层和输出层都是拥有激活函数的功能单元。每层神经元与下一层神经元全互联，神经元之间不存在同层连接，也不存在跨层连接，这种结构的神经网络即为 “多层前馈神经网络”。学习过程就是根据训练数据，调整神经元之间的 “连接权” 以及每个功能神经元的阈值。训练多层前馈神经网络常使用误差逆传播算法（error Back-Propagation），算法工作流程大致为：先根据示例值与初始化参数算出输出值，然后计算输出层的误差，再将误差逆向传播至隐层神经元，根据隐层神经元的误差对连接权和阈值进行调整，迭代过程循环进行，直到达到某个条件。BP 算法具有较强的表示能力，但也存在容易拟合的缺陷，训练误差在减少，但是预测准确度在下降。数据一般分为训练集和测试集，训练集计算梯度并更新连接权限和阈值，测试集用于验证错误，当测试集错误增加时停止训练。

本文介绍推演了前馈神经网络和 BP 神经网络的原理公式，并通过实现手写字体的识别与解决二分类问题，在实践中深刻理解人工神经网络的反向传播算法。

1.2 前馈神经网络基本原理（本章节作者：招欣乐）

神经网络是指在机器学习领域，一种模仿生物神经网络的结构和功能的数学模型或计算模型，其中最基本的成分是神经元。前馈神经网络是神经网络的一种，也是最简单、最有效的神经网络，其本质是一种数学函数，它能将一组输入数据映射成输出数据，这样复杂的函数是由许多较简单的函数复合而成 [3]。它的基本原理是信号从输入层向输出层单向传播，在这种神经网络中，各神经元分层排列，神经元之间不存在同层连接以及跨层连接，每个神经元可以接收上一层 n 个神经元传递的输入信号，这些输入信号可以连带着权重进行传递，然后将神经元接收到的信号与神经元的阈值比较进行函数处理，输出给下一层神经元。但是，输入层神经元只接收外界输入，不进行函数处理，而隐层和输出层神经元可以对信号进行加工，并且输出层神经元需要输出结果。因此，整个神经网络的信息都是从输入层向输出层传递，不存在反向的信息传递。

对于前馈神经网络，一般可以分为单层前馈神经网络和多层前馈神经网络。其中单层前馈神经网络只包含一个输出层，输出层上的节点的值一般由输入值乘以权重得到，其输入与输出的变换关系为：

$$s_j = \sum_{i=1}^{n} w_{ji} x_i - \theta_j \tag{1}$$

$$y_j = f(s_j) = \begin{cases} 1 & s_j \geq 0 \\ 0 & s_j < 0 \end{cases} \tag{2}$$

而多层前馈神经网络包括一个输入层和一个输出层，中间有多个隐层。在这里每一层相当于一个单层前馈神经网络，其输入与输出的变换关系为：

$$s_i^{(q)} = \sum_{i=1}^{n_{q-1}} w_{ij}^{(q)} x_i^{(q-1)}, (x_0^{(q-1)} = \theta_j, w_{i0}^{(q-1)} = -1) \tag{3}$$

$$x_i^{(q)} = f(s_i^{(q)}) = \begin{cases} 1 & s_i^{(q)} \geq 0 \\ -1 & s_i^{(q)} < 0 \end{cases} \tag{4}$$

接着，我们介绍一下前馈神经网络中比较重要的一个环节-激活函数。从之前的感知机模型中，我们知道神经元对输入的数值是利用加权求和的方式来进行运算的，这样以来，每一层的输入数据则为上一层数据的线性叠加，到最后整个网络的输出结果也是线性的。但是我们遇到的实际问题往往是非线性的，这个时候我们就需要一个处理来解决这些非线性的问题，因此我们在输出结果之前加了一个费心行的函数即为激活函数，将线性的结果映射到非线性的空间中，然后将达到要求的数据输出，而未达到要求的数据则会被抑制输出。另外，我们常用的激活函数有 Sigmoid 函数、tanh 函数、Relu 函数，而使用者需要根据自己的需求和具体情况来选择不同的激活函数。

虽然前馈神经网络的结构简单，但是其应用十分广泛，我们常用的前馈神经网络有感知器网络、BP 网络、RBF 网络，其中感知器网络主要用于模式分类，是最简单的前馈神经网络；RBF 神经网络是指隐含层由 RBF 神经元组成的神经网络；BP 神经网络可以实现从输入到输出的任意的非线性映射。近年来，前馈神经网络在各个研究领域都有了较多的应用与研究，极大影响了如自然语言理解等领域，推动了机器学习的发展。

1.3 BP 神经网络（本章节作者：王鑫源）

1.3.1 BP 神经网络概述

BP 神经网络是指使用了反向传播算法（即 BP 算法）的多层前馈神经网络，它是 1986 年由 Rumelhart 和 McCelland 为首的科学家小组提出的，当前比较流行的卷积神经网络（CNN）、循环神经网络（RNN）等训练算法都是在 BPNN 算法基础上经过优化和改进发展而来的。BP 神经网络在训练时有三个重要的过程：正向传播过程、反向传播过程以及权重更新过程 [4]。

在神经网络信号的正向传播中，神经元接收到来自其他神经元的输入信号，这些信号乘以所在边的权重累加到下一层神经元接收的总输入值上，然后经过激活函数的处理，产生神经元的输出。我们常采用 sigmoid 函数作为激活函数。

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{5}$$

BP 算法的核心思想是：使信号正向传播和误差反向传播这两个过程交替进行，每当信号正向传播产生输出层的结果之后，都要计算输出层的误差，然后将误差反向传播到神经网络的各层，依次调节隐含层到输出层的权重和偏置，输入层到隐含层的权重和偏置，此循环过程迭代进行 [5]。从理论上来说，在经过多次循环过程之后，神经网络的误差会收敛在一个稳定的范围之内，这时候就可以认为各层参数矩阵的值达到了理想值，即模型达到了最优状态，当达到某些停止条件时，该循环过程就会停止。

事实上，我们使用 BP 神经网络的目的是要找到一个最优解，也就是累计误差最小的解。为了找到这个最优解，最常在机器学习中使用的算法是梯度下降法。在理想情况下，我们希望有一个最优解，但在实际使用的过程中，局部最优解（即对应梯度为 0）往往有多个，全局最优解只有一个，这种情况下很有可能会掉入局部最优解而找不到全局最优解，因此我们只能尽可能找到最优解，并不能保证每次都能找到全局最优解。

1.3.2 具体理论推导

作为有监督的 BP 模型，我们用于训练的训练集往往包括了网络的输入 X 和它所期望的输出 Y。因此，对于当前已有的网络模型，可以得到网络输出相对于期望输出的误差。在训练时，首先进行信号正向传播，即经过输入层-各隐藏层-输出层，得出输出层的实际输出与期望输出的误差，然后将误差反传，使用反传的误差来修正各连接层神经元的权值，直到网络的误差收敛或者已经完成指定的学习次数。

一个神经网络包括输入层、隐含层和输出层，以一个简单的 BP 神经网络为例进行推导，定义变量如下 [6]：

假设神经网络的输入层有 d 个神经元，隐含层有 l 个神经元，输出层有 q 个神经元;

输出层第 j 个神经元的阈值为 θ_j；

隐含层第 h 个神经元的阈值为 γ_h；

输入层第 i 个神经元与隐含层第 h 个神经元之间的连接权为 V_{ih}；

隐含层第 h 个神经元与输出层第 j 个神经元之间的连接权为 W_{hj}；

记隐含层第 h 个神经元接收到来自输出层的输入为 α_h，输出层第 j 个神经元接收到来自隐含层的输入为 β_j：

$$\alpha_h = \sum_{i=1}^{d} V_{ih} X_i \tag{6}$$

$$\beta_j = \sum_{h=1}^{q} V_{hj} b_h \tag{7}$$

在神经网络中，神经元接收到来自其他神经元的输入信号，这些信号乘以权重累加到神经元接受的总输入值上，然后与当前神经元的阈值进行比较，再通过激活函数的处理，产生神经元的输出。假设神经网络的隐含层和输出层的激活函数都使用 $Sigmoid$ 函数，用 $f(.)$ 表示激活函数。

对于训练样例 (x_k, y_k)，假设神经网络的输出为 Y_k，则

$$Y_j^K = f(\beta_j - \theta_j) \tag{8}$$

(x_k, y_k) 的均方误差为：

$$E_k = \frac{1}{2}\sum_{j=1}^{l}[Y_j(k) - y_j(k)]^2 \tag{9}$$

那么，在此神经网络中，对于隐含层的第 h 个神经元，它要接收来自输入层的 d 个权重参数，向输出层传送 l 个权重参数，再加上自身的一个阈值，一个隐含层神经元有 $(d+l+1)$ 个待定参数。输出层的每个神经元还有一个阈值，输出层共有 l 个参数。最后网络中共有 $(d+l+1)*q+l$ 个待定参数。根据 BP 算法的迭代对参数进行更新估计，任意权重参数 w 的更新公式为：

$$w \longleftarrow w + \triangle w \tag{10}$$

BP 算法基于梯度下降法来求解最优解，以目标的负梯度方向对参数进行调整。对于误差 E_k，给定学习率 η，有

$$\triangle w_{hj} = -\eta\frac{\partial E_k}{\partial w_{hj}} \tag{11}$$

要计算 $\frac{\partial E_k}{\partial w_{hj}}$ 根据链式法则，有

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial Y_j^k}\frac{\partial Y_j^k}{\partial \beta_j}\frac{\partial \beta_j}{\partial w_{hj}} \tag{12}$$

根据输出层神经元的输入值 β_j 的定义，可得

$$\frac{\partial \beta_j}{\partial w_{hj}} = b_h \tag{13}$$

对于激活函数（$sigmoid$ 函数），求导可得如下性质，并有

$$f'(x) = f(x)[1 - f(x)] \tag{14}$$

利用这个性质进行如下推导，令

$$g_j = -\frac{\partial E_k}{\partial Y_j^k}\frac{\partial Y_j^k}{\partial w_{hj}} \tag{15}$$

结合公式 (8) 和公式 (9)，可以得到

$$\begin{aligned} g_j &= -(Y_j^k - y_j^k)f'(\beta_j - \theta_j) \\ &= (y_j^k - Y_j^k)f(\beta_j - \theta_j)[1 - f(\beta_j - \theta_j)] \\ &= (y_j^k - Y_j^k)Y_j^k(1 - Y_j^k) \end{aligned} \tag{16}$$

此结果结合公式 (12) 和公式 (13) 可得

$$\begin{aligned}\frac{\partial E_k}{\partial w_{hj}} &= -g_j b_h \\ &= -Y_j^k(1-Y_j^k)(y_j^k - Y_j^k)b_h\end{aligned} \tag{17}$$

将结果代入公式 (11)，就可以得到梯度更新公式

$$\begin{aligned}\triangle w_{hj} &= \eta g_j b_h \\ &= \eta Y_j^k(1-Y_j^k)(y_j^k - Y_j^k)b_h\end{aligned} \tag{18}$$

在此公式中，η 为学习率（即步长），步长太大可能会导致振荡，步长太小收敛速度会太慢；Y_j^k 为神经网络输出层第 j 个神经元的输出值；y_j^k 为训练样例 (x_k, y_k) 的标志，即训练集给出的正确输出；b_h 为隐含层第 h 个神经元的输出。

类似可得

$$\triangle \theta_j = -\eta g_j \tag{19}$$

$$\triangle v_{ih} = \eta e_h x_i \tag{20}$$

$$\triangle \gamma_h = -\eta e_h \tag{21}$$

用前文的推导方法类似可推得

$$\begin{aligned}e_h &= -\frac{\partial E_k}{\partial b_h}\frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^{l}\frac{\partial E_k}{\partial \beta_j}\frac{\partial \beta_j}{\partial b_h}f(\alpha_h - \gamma_h) \\ &= b_h(1-b_h)\sum_{j=1}^{l} w_{hj}g_j\end{aligned} \tag{22}$$

1.3.3 BP 神经网络的超参数

BP 算法为一阶梯度方法，因此 BP 神经网络收敛起来比较慢，特别是在多层网络的时候，代价函数曲面一般是非二次、非凸和高维的，因此有非常多的局部最小值的区域。所以 BP 算法无法保证收敛的快速性和所得解的最优性。在使用 BP 神经网络进行训练时，在训练数据质量较好的情况下，BP 神经网络可以解决一些简单的线性回归或逻辑回归问题，但是当训练数据比较庞大冗余时，BP 神经网络容易陷入局部最优解，同时由于神经网络内部使用全连接，会带来大规模数据的计算问题。因此我们考虑尽可能的让 BP 神经网络收敛速度加快并得到相对好的解。

一般而言，参数初始化对训练过程有着重大的影响，我们对参数初始化的原则是：参数应该随机初始化在能让 sigmoid 函数在线性区域激活的值。如果参数的初始值设置较大，那么 sigmoid 函数在运行之初就会饱和，使得参数的更新变慢，收敛速度变慢；如果参数的初始值设置较小，那么模型的梯度值也会很小，同样导致收敛速度慢。因此，参数应该取能让 sigmoid 函数在线性区域激活的值，这样可以使得梯度足够大，从而使得参数学习正常进行。

学习率的选择也很重要，学习率即参数更新的步长。学习率如果设置的太大，就会导致模型在最优解附近震荡，学习率太小会导致收敛的太慢。一般来说，在模型训练的过程中，起初可以设置稍大的学习率，

当接近最优值即将震荡时，使学习率减小，寻找最优值。

2 BP 神经网络的实现

2.1 BP 神经网络实例之手写数字识别

上文中，我们简单介绍了神经网络，阐述了前馈神经网络的基本原理，并详细介绍了 BP 神经网络及其理论推导。本章我们主要通过 BP 神经网络在手写数字图片识别方面的应用来介绍 BP 神经网络的实现步骤 [7]。

2.2 数据集准备及处理（本章节作者：招欣乐）

我们使用 Handwritten Digits 数据集，该数据集分为 optdigits.tra 和 optdigits.tes 两个数据集，分别是训练数据集和测试数据集，数据集中有训练数据样本 3823 条，测试数据 1797 条, 数据如表1、2。首先利用 Numpy 中 genfromtxt 函数读取数据集；接着对训练集的数据进行拟合处理，并对训练集和测试集的 x 轴数据进行标准化处理；最后对数据集的 y 轴数据进行“1 of n”编码处理。

类别	数字 0	数字 1	数字 2	数字 3	数字 4	数字 5	数字 6	数字 7	数字 8	数字 9
样本数	376	389	380	389	387	376	377	387	380	382

表 1: 训练集样本分布

类别	数字 0	数字 1	数字 2	数字 3	数字 4	数字 5	数字 6	数字 7	数字 8	数字 9
样本数	178	182	177	183	181	182	181	179	174	180

表 2: 测试集样本分布

2.3 模型训练（本章节作者：王鑫源）

在此前数据准备的基础上，我们创建一个 3 层的简单神经网络模型，该神经网络的隐藏层只有一层。训练数据有 64 个特征，分类问题有 10 个类别（分别为数字 0-9），因此，输入层设置 64 个节点，输入层由样本输入向量构成，神经网络设置一个包含 100 个节点的隐藏层，输出层设置 10 个节点，表示共有 10 个类别，使用反向传播算法的训练模型的过程如下 [8]：

（1）前馈计算每一层的净输入和激活值，直到最后一层；

（2）反向传播计算每一层的误差；

（3）计算每一层参数的偏导数，并更新参数。

算法1为使用反向传播算法训练模型的过程 [6]。

算法 1 使用反向传播算法的模型训练过程

输入: 训练集 $D = (x_k, y_k)_{k=1}^{m}$，学习率 η

输出: 确定连接权与阈值的神经网络

随机初始化网络中所有的连接权和阈值
repeat
  for all $(x_k, y_k) \in D$ **do**
    前馈计算每一层的净输入和激活值，直到最后一层；
    反向传播每一层的误差；
    更新连接权与阈值；
  end for
until 达到某些停止条件

我们将此训练过程迭代了 500 次，并将训练过程中的误差打印出来，如图1, 我们观察到在模型训练的前期误差下降的速度比较快，前 20 次迭代误差就收敛到小于 0.01，而在误差下降的后期则变得非常缓慢，趋近于收敛。

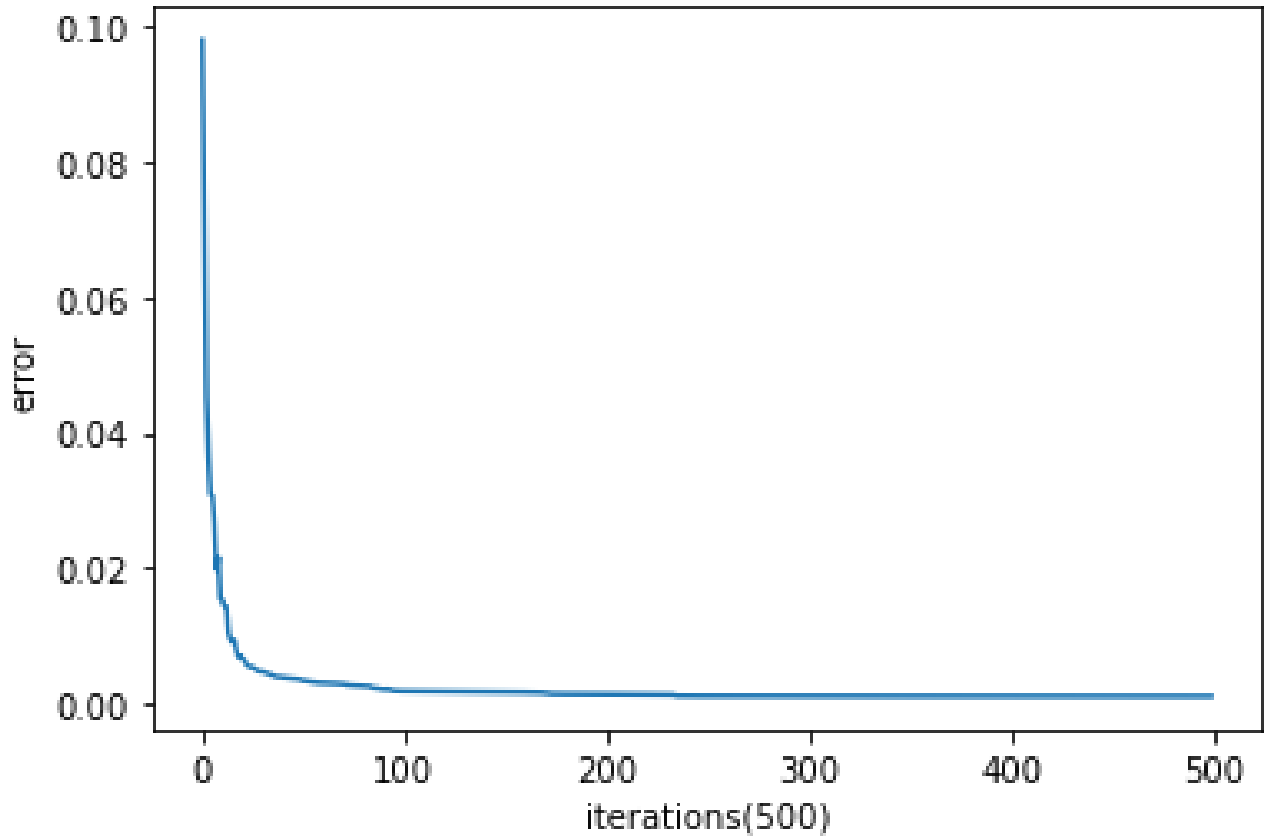


图 1: 模型训练误差图 (图片来源：实验自制图)

2.4 模型预测与测试（本章节作者：张俊瑛）

使用上一步训练后的模型对测试集进行预测，将类别的”1 of n”编码格式的测试结果解码，使预测值结果与测试集类标记类型一致。最后利用 sklearn 中的 accuracy _score 函数计算预测准确率，在隐藏层结

构为一层一百个节点、随机梯度下降算法学习率为 0.3、迭代次数为 500、判断收敛误差阈值为 0.00001 参数设置情况下，得到的准确率为 96.55%。

2.5 调参优化（本章节作者：张俊瑛）

保持隐藏层、学习率和迭代次数不变，更改判断收敛误差阈值为 0.000001 时，再次重复训练模型并使用模型进行预测，得到对手写数字的识别准确率为 96.61%。保持判断收敛误差阈值、学习率和迭代次数不变，隐藏层更改为 150 时，再次重复训练模型并使用模型进行预测，得到对手写数字的识别准确率为 96.61%。保持判断收敛误差阈值、隐藏层和迭代次数不变，学习率更改为 0.25 时，再次重复训练模型并使用模型进行预测，得到对手写数字的识别准确率为 96.88%。保持判断收敛误差阈值、隐藏层和学习率不变，迭代次数更改为 700 时，再次重复训练模型并使用模型进行预测，得到对手写数字的识别准确率为 96.88%。综上可知，在增加隐藏层、降低学习率、提高迭代次数或缩小判断收敛误差阈值的情况下，都能提高模型预测的准确率。

	1	2	3	4	5
隐藏层节点数	100	100	150	100	100
学习率	0.3	0.25	0.3	0.3	0.3
迭代次数	500	500	500	700	500
收敛误差阈值	0.00001	0.00001	0.00001	0.00001	0.000001
准确率	96.55%	96.88%	96.61%	96.88%	96.61%

表 3: 各参数与准确率关系表

2.6 BP 神经网络实例之二分类问题-网络结构与数据集（本章节作者：王鑫源）

2.6.1 网络结构

作为 BP 神经网络原理的实现和应用，我们将实现一个 5 层的 BP 神经网络来完成二分类任务 [9]，神经网络由输入层、三个隐藏层、输出层构成。在实例中，神经网络的设置如下：

（1）网络有一个输入层、三个隐藏层、一个输出层；

（2）输入层节点数为 2，三个隐藏层的节点数分别设置为：25，50 和 25，输出层的节点数为 2，分别代表属于两个类别的概率。

为了更直接的实现和利用梯度传播算法，本实例实现的重点是针对全连接层，并使用 sigmoid 激活函数，损失函数为均方误差，直接利用均方误差计算与 one-hot 编码的真实标签之间的误差。

2.6.2 数据集

为了获得理想的数据集，我们使用函数库中的便捷工具生成一个线性不可分的 2 分类数据集，该数据集中样本个数为 2000，数据的特征长度为 2，数据分布如图2所示，两种颜色的点分别表示两类样本，根据图2可以看出，两个类别的数据分布呈月牙形，且是线性不可分的，这就意味着我们无法通过线性模型获得较好的效果。对于数据集的划分，我们按照 7：3 的比例划分训练集和测试集，即有 1400 个样本用于训练，600 个样本用于测试。

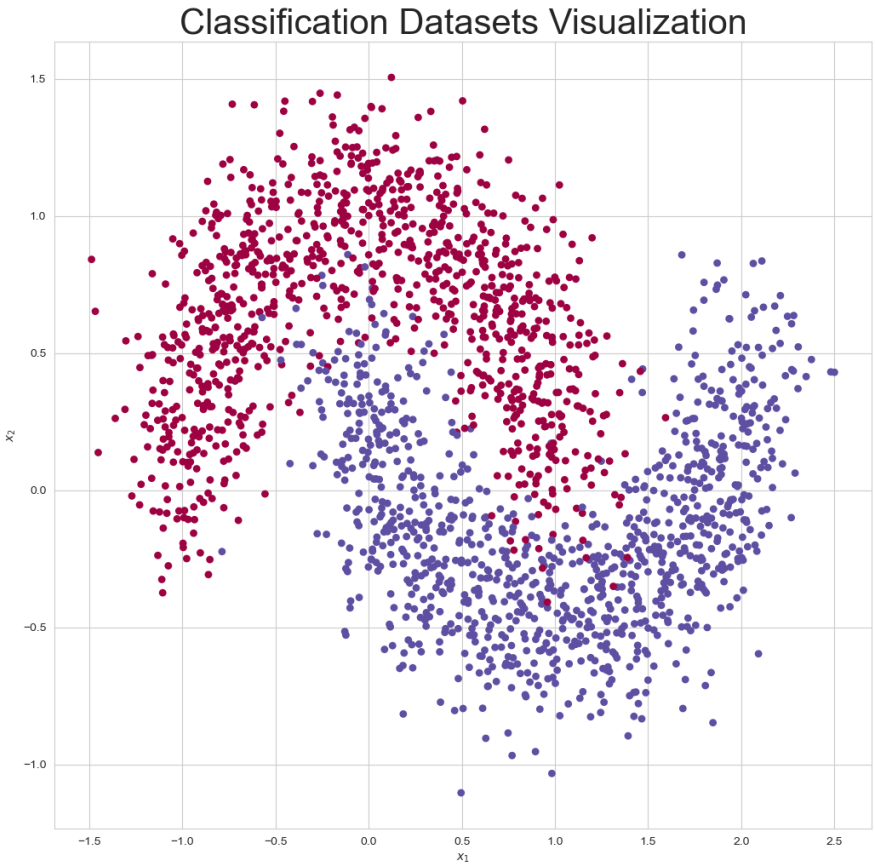


图 2: 数据集分布（图片来源：实验自制图）

2.7 网络层和网络模型（本章节作者：招欣乐）

2.7.1 网络层

BP 神经网络是由输入层、中间层、输出层三部分组成，其中中间层可以是一层，也可以是多层，其相邻的神经元之间进行全连接，而每层神经元之间无连接。因此，建立合适的全连接网络层对于 BP 神经网络的构建来说至关重要。

首先我们通过新建一个类 Layer 实现一个网络层的建立，输入网络层需要的输入节点数、输出节点数、激活函数类型等参数，其中输入节点数和输出节点数这两个参数必须设定为期望的节点数量，然后各层神经元之间的联系即权重和偏置张量可以在初始化的过程中根据输入、输出节点数自动生成并通过正态分布初始化，其中初始化对于整个神经网络的搭建非常重要，如果没有初始化或者采用不合适的初始化的话，可能会导致神经网络出现不收敛的情况。

接着，我们定义激活函数类型，例如：Sigmoid 函数，设置激活函数的输出值，利用 self.error 参数计算位于当前层的 delta 变量的中间变量和使用 self.delta 函数记录位于当前层的 delta 变量便于计算梯度，然后定义网络层的前向传播函数 activate，通过激活函数得到全连接层的输出，计算当前层的输出值并保存其输出值，其中 last_activation 函数可以用于保存当前层的输出值。这里我们只采用了 Sigmoid 激活函数一种，如果无激活函数则会直接返回，但 self_apply_activation 函数可以实现不同类型的激活函数的前向计算过程，因此，针对不同类型的激活函数，我们可以计算其导数，例如：Sigmoid 函数的导数实现为 r(1-r)，

其中 r 即为 σ（z)。

2.7.2 网络模型

上文中我们建立了单层网络类 Layer，在此基础上我们新建了一个神经网络大类 NeuralNetwork 类，其内部可以用来维护各层的网络类 Layer 的对象，如果想要增加网络层，可以使用 add_layer 函数以达到创建不同结构的网络层的目的，因此，我们利用 NeuralNetwork 类创造的神经网络和 add_layer 函数共添加了 4 层全连接层。

同时，网络的前向传播比较简单，只需要循环调用各个网络层对象的前向计算函数，自动生成权重，计算输出值并输出相应的输出值传递给下一网络层对象作为输入值，这样依次通过各个网络层，最后输出最终的值。

而网络模型的反向传播的实现就较为复杂，需要从最后一层开始，计算每层的 delta 变量，然后根据推导的梯度公式，将计算出来的 delta 变量保存在 Layer 类的 delta 变量中。在创建的这个网络模型中，我们是先利用前向传播得到的输出值反向循环，得到当前层的对象。如果当前层为输出层，则需要计算 2 分类任务的均方差的导数，然后计算最后一层的 delta，并参考输出层的梯度公式；如果是隐藏层，则需要得到下一层的对象，并计算隐藏层的 delta，参考隐藏层的梯度公式，这一步是网络模型反向传播能够实现的关键步骤。接着，在计算完每层的 delta 变量后，按照公式

$$\frac{\partial L}{\partial w_{ij}} = o_i {\delta_j}^J \tag{23}$$

计算每层参数的梯度值，最后按照梯度下降算法进行每一次网络参数的更新。由于撰写的代码中 delta 计算的其实是其负值，因此在更新时使用了加号进行纠正。

2.8 网络训练和网络性能优化（本章节作者：张俊瑛）

2.8.1 网络训练

网络训练中，因为输出节点有两个，所以需要对真实标签 0 或 1 进行 One-bot 编码。One-Hot 编码又称为一位有效编码，主要是采用 N 位状态寄存器来对 N 个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效。在机器学习中，当特征值不是连续值而是离散值时，通常将分类变量转换为整数值，再表示为二进制向量，转换后的特征向量可以提高模型的运行效率。之后通过调用反向传播算法更新网络模型参数，设定循环迭代训练集次数为 1000 次，每次循环训练集时，样本一次一个单独输入。反向传播算法实现过程中，首先获得当前层的输出值，如果是输出层，计算 One-hot 编码后的真实标签与前向传播得到的的输出值的均方误差导数，参考输出层的梯度公式计算最后一层的梯度项; 如果是隐藏层，根据下一层的梯度项和权重计算误差与当前层的梯度项；最后根据梯度更新公式循环更新权值，1000 次后停止即得到训练好的网络模型。

2.8.2 网络性能测试

将训练集输入到网络模型进行预测，学习率为 0.01，最大迭代次数为 1000 次。根据预测值与 One-hot 编码后的实际值计算预测数据和原始数据对应点的误差，即真实值与预测值差的平方和的均值；同时计算并记录预测准确率。在训练完 1000 次后，最终在测试集 600 个样本上得到的均方误差为 0.024415，准确率为 97.67%。

将每次循环迭代得到的训练损失绘制成曲线3，可以看到，通过手动计算梯度公式并手动更新网络参数的方式，我们在简单的二分类任务中也能获得较低的错误率。

在每一次循环中，我们也进行了准确度测试并绘制了曲线3。可以从图中看到，随着迭代次数的增加，模型的准确率稳步提升，开始阶段提升加快，后续收敛提升较为平缓。

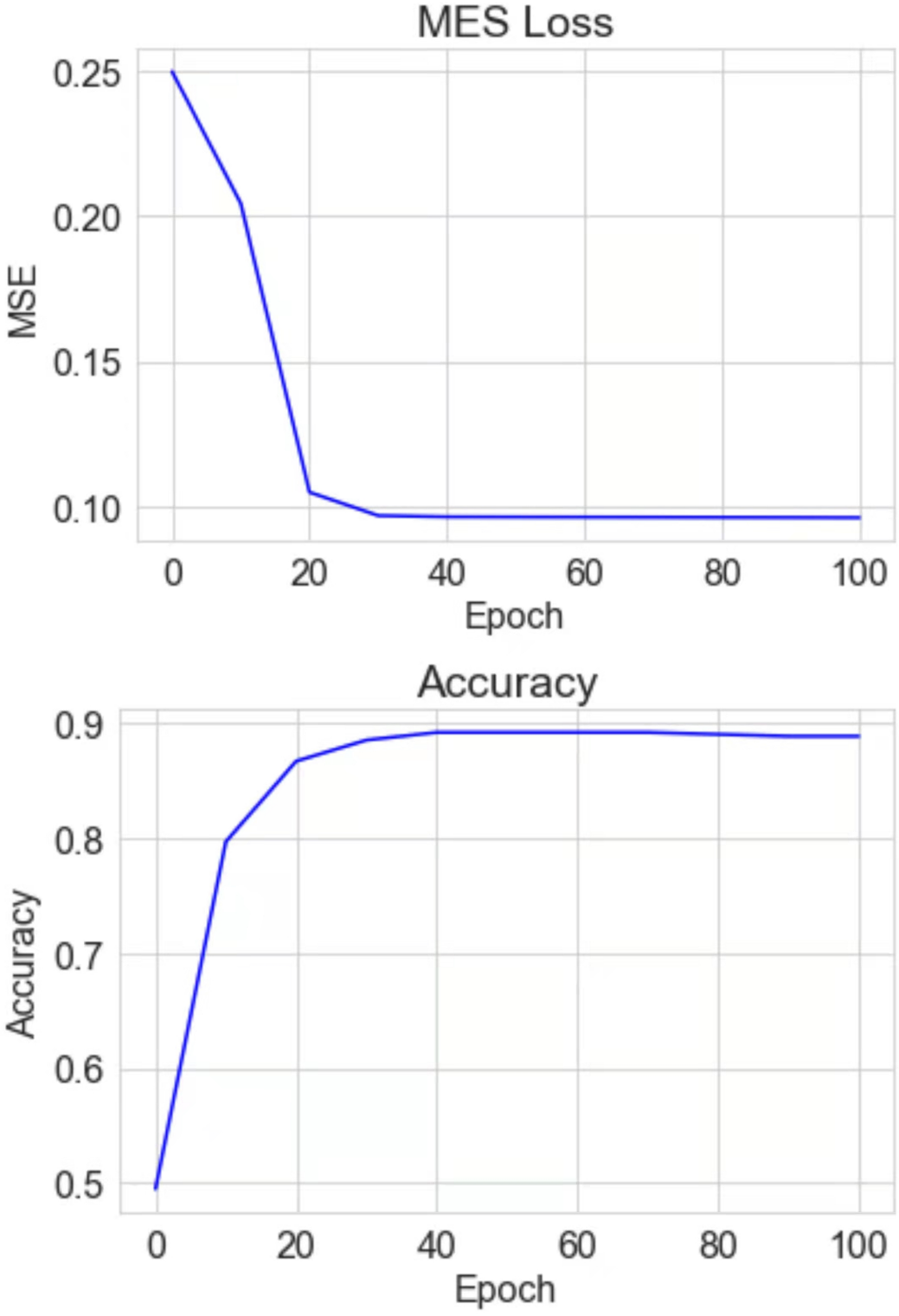


图 3: 网络训练误差曲线与网络测试准确率曲线图（图片来源：实验自制图）

3 后记

3.1 课程论文构思和撰写过程（本章节作者：张俊瑛）

在给定范围内选择课题方向，经小组讨论后，大家一致决定选择 BP 神经网络，且开会敲定了课程论文思路大纲与每个人的分工明细。我们讨论确定论文一共分为三个部分，第一部分先介绍 BP 神经网络的起源，对它的发展过程做个简单的介绍，更细分了总览、前馈神经网络和误差逆传播算法；第二部分详细介绍了 BP 神经网络的原理，并一步步推演算法原理，罗列公式；第三部分就是算法的实现，借助实例搭建了 BP 神经网络，在实战中加强对神经网络的理解。全部内容完成后，小组一起整理全文，完善参考文献及格式等细节问题。

3.2 所参考主要资源（本章节作者：张俊瑛）

算法的主要思想内容是阅读了《机器学习》（周志华著）、看了吴恩达机器学习课程以及课程《数据挖掘》的教授内容，还参阅了邱锡鹏的《神经网络与深度学习》，同时也查阅了一些网上资料，加深对算法的了解。参考代码来源于《Python 机器学习算法：原理、实现与案例》（刘硕著）和龙龙老师的《TensorFlow 深度学习》。

3.3 人员分工（本章节作者：王鑫源）

张俊瑛：撰写选题说明及第二章中模型预测与测试部分，同时还撰写了第三章中网络模型训练与性能预测部份，还包括了第四章中课程论文构思及参考资源部分。

王鑫源：撰写 BP 神经网络概述及理论推导部分，以及第一个例子中中模型训练部分和第二个例子中网络结构介绍与数据集准备部分。

招欣乐：撰写前馈神经网络基本原理，及第一个例子中数据集准备及处理部分和第二个例子中网络层与网络模型的介绍部分。

3.4 代码撰写的构思和体会（本章节作者：王鑫源）

我们撰写代码的基本思路是要应用 BP 神经网络的基本原理和算法，实现一些简单的功能，但整体的重点要放在 BP 神经网络的实现上。首先，基于对前文介绍的 BP 神经网络基本原理和对反向传播算法的推导，我们构建了一个三层的神经网络模型对加州大学 Irvine 机器学习库的手写字体数据集进行识别，然后改变神经网络的参数，得到各参数对结果准确率的影响。第三章中基于前文简单的算法介绍神经网络的搭建，我们构建了一个稍加复杂的五层神经网络模型，包括四层全连接层，我们使用函数库中的便捷工具生成一个理想的线性不可分的二分类数据集，样本标签使用 one-hot 编码，应用构建的神经网络对这个二分类的数据集进行分类。

在撰写代码的过程中，我们也遇到了各种各样的问题，但经过查阅资料，还是比较顺利的完成了代码实现。写代码这个过程更偏重工程性，要了解整个项目的结构写出的代码才更有逻辑性，同时写代码的过程中也要注意实现的细节，不然很容易就会出错。我们发现现在所使用的库函数都隐藏了有关函数的底层实现，这的确更方便了我们调用函数进行自己想要的功能实现，但也增加了我们和底层算法的距离，需要我们在日常的学习中更加注重算法的原理。在写代码的过程中，只有清楚了解网络和算法的实现过程，才能更有逻辑性、更有效率的进行代码实现。

参考文献

[1] Pitts W. McCulloch WS. A logical calculus of the ideas immanent in nervous activity. 1990.

[2] 尹雪婷张亮明 白艳明王F, 贾文雅. 人工神经网络的发展及展望. 智能城市,*2021,7(08)*, (12-13), 2021.

[3] 黄丽婷. 基于前馈神经网络的预测分类器. 2021.

[4] 郑柳刚袁冰清, 功 程. Bp 神经网络基本原理. 数字通信世界,*Digital communication World,2018(8)*, 1672-7274(28-29), 2018.

[5] 韩普, 周汉辰, 周北望. Bp 神经网络原理研究与实现. 广播电视信息,*Radio Television Information,2018(10)*, 1007-1997(121-125), 2018.

[6] 周志华. 机器学习. 清华大学出版社, 2016.

[7] 刘硕著. *Python* 机器学习算法：原理、实现与案例. 清华大学出版社, 2019.

[8] 邱锡鹏. 神经网络与深度学习. 机械工业出版社, 2020.

[9] 龙龙老师. *TensorFlow* 深度学习. 2019.

4 附录

4.1 文中所使用的训练数据集和测试数据集及完整代码

```
数据来源: E. Alpaydin, C. Kaynak Department of Computer Engineering Bogazici University, 80815 Istanbul
    Turkey alpaydin@boun.edu.tr July 1998
代码地址: https://github.com/handsomezjy/-
图1代码:
for _ in range(self.max_iter):
    np.random.shuffle(idx)
    X, y = X[idx], y[idx]
    for x, t in zip(X, y):
        out = x
        for i in range(layer_n):
            in_ = np.ones(out.size + 1)
            in_[1:] = out
            z = self._z(in_, W_list[i])
            out = self._sigmoid(z)
            in_list[i], z_list[i], out_list[i] = in_, z, out
        delta_list[-1] = out * (1. - out) * (t - out)
        for i in range(layer_n - 2, -1, -1):
            out_i, W_j, delta_j = out_list[i], W_list[i+1], delta_list[i+1]
            delta_list[i] = out_i * (1. - out_i) * np.matmul(W_j[1:], delta_j[:, None]).T[0]
        for i in range(layer_n):
            in_i, delta_i = in_list[i], delta_list[i]
            W_list[i] += in_i[:, None] * delta_i * self.eta
    y_pred = self._predict(X, W_list)
    err = self._error(y, y_pred)
    error.append(err)
    if err < self.tol:
        break
    print('%4s. err: %s' % (_+1, err))

plt.plot(np.squeeze(error))
```

```
plt.ylabel("error")
plt.xlabel("iterations(500)")
plt.show()
图2代码:
def make_plot(X,y,plot_name,file_name=None,XX=None,YY=None,preds=None,dark=False):
    if (dark):
        plt.style.use('dark_backgroud')
    else:
        sns.set_style('whitegrid')
    plt.figure(figsize=(16,12))
    axes = plt.gca()
    axes.set(xlabel="$x_1$",ylabel="$x_2$")
    plt.title(plot_name,fontsize=30)
    plt.subplots_adjust(left=0.20)
    plt.subplots_adjust(right=0.80)
    if (XX is not None and YY is not None and preds is not None):
        plt.contourf(XX,YY,preds.reshape(XX.shape),25,alpha=1,cmap=cm.Spectral)
        plt.contour(XX,YY,preds.reshape(XX.shape),levels=[.5],cmap="Greys",vmin=0,vmax=.6)
    plt.scatter(X[:,0], X[:,1],c=y.ravel(),s=40,cmap=plt.cm.Spectral,edgecolors='none')
make_plot(X,y,"Classification␣Datasets␣Visualization")
plt.show()
图3代码:
nn = NeuralNetwork()
nn.add_layer(Layer(2, 25, 'sigmoid'))
nn.add_layer(Layer(25, 50, 'sigmoid'))
nn.add_layer(Layer(50, 25, 'sigmoid'))
nn.add_layer(Layer(25, 2, 'sigmoid'))
mses, accuracys = nn.train(X_train, X_test, y_train, y_test, 0.01, 1000)

x = [i for i in range(0, 101, 10)]
plt.title("MES␣Loss")
plt.plot(x, mses[:11], color='blue')
plt.xlabel('Epoch')
plt.ylabel('MSE')
plt.show()

plt.title("Accuracy")
plt.plot(x, accuracys[:11], color='blue')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.show()
```

4.2 《卷积神经网络原理及其理论发展》——李克勤，何峰，李振国

摘要：卷积神经网络 (Convolutional Neural Networks, 简称 CNN) 可以说是深度神经网络模型中的“明星” 网络架构，在计算机视觉方面贡献颇丰。一个标准的卷积神经网络架构主要由卷积层、池化层和全连接层等核心层次构成。随着研究者对卷积神经网络的研究，一些卷积神经网络的框架逐渐问世。本文主要是对卷积神经网络的理论介绍和实现以及对一些经典的卷积神经网络架构的理解。

卷积神经网络原理及其最近理论发展

李振国, 李克勤，何锋

College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

卷积神经网络（Convolutional Neural Networks, 简称 CNN）可以说是深度神经网络模型中的“明星”网络架构，在计算机视觉方面贡献颇丰。一个标准的卷积神经网络架构主要由卷积层、池化层和全连接层等核心层次构成。随着研究者对卷积神经网络的研究，一些卷积神经网络的框架逐渐问世。本文主要是对卷积神经网络的理论介绍和实现以及对一些经典的卷积神经网络架构的理解。

关键词：卷积神经网络，CNN，网络架构

1 概况

1.1 选题说明（本章节作者：李振国）

近年来，人工智能一词逐渐进入了大众的视野。从越来越智能的语音助手，再到智能驾驶等等，其中最为火热的莫过于 2016 年 AlphaGo 以 4：1 大胜人类顶尖棋手李世石。但是实际上，神经网络并不是一样新新事物。早在上世纪初，神经网络一词就已经为人所知。由 Rosenblatt 提出并实现的感知机首次将人工神经网络的研究从理论复现到了现实。然而在第三次科技革命时期，数字计算机技术飞速发展，加之当时的硬件和数据量不足以支撑神经网络所需要的庞大的数据运算，感知机注定只能是昙花一现。但它同时也为 Hinton 等人打了一剂强心针。在大多数人都不看好神经网络的当时，Hinton 等人坚持神经网络能够解决常规算法不能解决的问题，最终在 1986 年提出了著名的 BP 算法，重新点燃了神经网络领域研究人员的热情，这种误差反向传播算法直到现在还在网络模型当中发挥着作用。

本文主要按照时间顺序叙述卷积神经网络在近年来的几个经典神经网络模型：一是加深一下对神经网络模型的了解与发展历程；再是为了追寻该领域前辈们的奇思妙想，致敬前辈们几十年来兢兢业业的不懈坚持。

1.2 卷积神经网络基本原理（本章节作者：何锋）

20 世纪 90 年代，LeCun 等人发表论文，确立了 CNN 的现代结构，后来又对其进行完善。他们设计了一种多层的人工神经网络，取名叫做 LeNet-5，可以对手写数字做分类。和其他神经网络一样，LeNet-5 也能使用反向传播算法（backpropagation）训练。CNN 能够得出原始图像的有效表征，这使得 CNN 能够直接从原始像素中，经过极少的预处理，识别视觉上面的规律。然而，由于当时缺乏大规模训练数据，计算机的计算能力也跟不上，LeNet-5 对于复杂问题的处理结果并不理想。之后，人们设计了很多方法，想要克服难以训练深度 CNN 的困难。近年来卷积神经网络在多个方向持续发力，在语音识别、人脸识别、通用物体识别、运动分析、自然语言处理甚至脑电波分析方面均有突破。

卷积神经网络与普通神经网络的区别在于，卷积神经网络包含了一个由卷积层和子采样层构成的特征抽取器。在卷积神经网络的卷积层中，一个神经元只与部分邻层神经元连接。在 CNN 的一个卷积层中，通常包含若干个特征平面 (featureMap)，每个特征平面由一些矩形排列的的神经元组成，同一特征平面的神

经元共享权值，这里共享的权值就是卷积核。卷积核一般以随机小数矩阵的形式初始化，在网络的训练过程中卷积核将学习得到合理的权值。共享权值（卷积核）带来的直接好处是减少网络各层之间的连接，同时又降低了过拟合的风险。子采样也叫做池化（pooling），通常有均值子采样（mean pooling）和最大值子采样（max pooling）两种形式。子采样可以看作一种特殊的卷积过程。卷积和子采样大大简化了模型复杂度，减少了模型的参数。

1.3 卷积神经网络应用范围（本章节作者：李克勤）

卷积神经网络为图像而生，但应用不限于图像。在图像处理任务上，卷积神经网络可以用来识别位移、缩放及物体形态扭曲的二维图形。一方面，由于其网络模型中的特征是通过训练数据集进行图像特征学习，从而避免了显式地特征抽取；另一方面，由于图像上同一特征映射面上的神经元权值相同，所以卷积神经网络模型可以并行训练，极大地提高神经网络的训练时长。此外，与神经元彼此相连的神经网络 (如传统的人工神经网络) 相比，卷积神经网络模型的组织方式特殊，其结构模型更易于理解和分析。卷积神经网络的应用场景和案例并不一定能够真正应用在实际工程领域，但也是足够精彩，因为它不仅代表业界最先进的视觉技术 (state-of-the-art)，甚至还可能超出我们的想象范围。它主要应用于图像分类与识别、自然语言处理、图像着色、视频帧预测、视频内容分类和视频标注等。

2 神经网络基础

2.1 卷积神经网络结构（本章节作者：何锋）

卷积神经网络 (Convolutional Neural CNN) 的基本结构由输入层、卷积层（convolutional layer）、池化层（pooling layer）、全连接层及输出层构成。卷积层和池化层一般会取若干个，采用卷积层和池化层交替连接，即一个卷积层连接一个池化层，池化层后再连接一个卷积层，依此类推。由于卷积层中输出特征面的每个神经元与其输入进行局部连接并通过对应的连接权值与局部输入进行加权求和再加上偏置值，得到该神经元输入值，该过程类似于卷积过程，CNN 也由此而得名。

2.1.1 输入层

比如说 LeNet-5 解决的手写数字分类问题的输入为一张 32x32 像素的灰度图像 (Gray Scale)。在日常生活中计算机常用的图像的表示方式为 RGB，即将一张图片分为红色通道 (Red Channel)，绿色通道 (GreenChannel) 和蓝色通道 (BlueChannel)，其中每个通道的每个像素点的数值范围为 [0,255]。灰度图像表示该图片仅包含一个通道，也就是不具备彩色信息，每个像素点的数值范围同 RGB 图像的取值范围相同。

因此，一张图片在计算机的眼里就是一个如下图所示的数字矩阵。在将图像输入到 CNN 网络之前，通常我们会对其进行预处理，因为每个像素点的最大取值为 255，因此将每个像素点的值除以 255 则可以将其归一化到 [0,1] 的范围。

2.1.2 卷积层

在了解卷积层之前，让我们先来了解一下什么是卷积? 设 f(x),g(x) 是 R 上的两个可积函数，则卷积定义为:

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(\tau)g(x-\tau)d\tau$$

离散形式定义为：

$$(f*g)(x)=\sum_{\tau=-\infty}^{\infty}f(\tau)g(x-\tau)d\tau$$

我们用一个示例来形象的理解一下卷积的含义，以离散的形式为例，假设我们有两个骰子，f(x),g(x) 分别表示投两个骰子，x 面朝上的概率。

$$f(x)=g(x)=\begin{cases}1/6 & \text{x=1,2,3,4,5,6}\\ 0 & \text{otherwise}\end{cases}$$

卷积 (f * g)(x) 表示投两个骰子，朝上数字之和为 x 的概率。则和为 4 的概率为：

$$\begin{aligned}(f*g)(4)&=\sum_{\tau=1}^{6}f(\tau)g(4-\tau)\\&=f(1)g(4-1)+f(2)g(4-2)+f(3)g(4-3)\\&=1/6\times1/6+1/6\times1/6+1/6\times1/6\\&=1/12\end{aligned}$$

这是一维的情况，我们处理的图像为一个二维的矩阵，因此类似的有：

$$(f*g)(z,y)=\sum_{v=-\infty}^{\infty}\sum_{h=+\infty}^{\infty}g(z-h,y-v)$$

这次我们用一个抽象的例子解释二维情况下卷积的计算，设 f,g 对应的概率矩阵如下：

$$f(x)=\begin{bmatrix}a_{0,0} & a_{0,1} & a_{0,2}\\ a_{1,0} & a_{1,1} & a_{1,2}\\ a_{2,0} & a_{2,1} & a_{2,2}\end{bmatrix}$$

则 (f * g)(1,1) 计算方式如下：

$$(f*g)(1,1)=\sum_{v=0}^{2}\sum_{h=0}^{2}g(1-h,1-v)$$

从这个计算公式中就不难看出为什么上面的 f,g 两个概率矩阵的角标会写成上述形式，即两个矩阵相同位置的角标之和均为 1。(f * g)(1,1) 即为 f,g 两个矩阵中对应位置的元素乘积之和。

在上例中，f,g 两个概率矩阵的大小相同，而在 CNN 中，f 为输入的图像，g 一般是一个相对较小的矩阵，我们称之为卷积核。这种情况下，卷积的计算方式是类似的，只是会将 g 矩阵旋转 180 度使得相乘的元素的位置也相同，同时需要 g 在 f 上进行滑动并计算对应位置的卷积值。

除此之外，卷积层还有有两个很重要的性质：

第一个性质是，局部连接。在卷积层（假设是第 L 层）中的每一个神经元都只和下一层（第 L−1 层）中某个局部窗口内的神经元相连，构成一个局部连接网络。作为参数的卷积核对于第 L 层的所有的神经元都是相同的。

第二个性质是，权值共享。权值共享可以理解为一个卷积核只捕捉输入数据中的一种特定的局部特征，所以如果要提取多种特征就需要使用多个不同的卷积核。

简单来说，卷积层就是负责提取图像中的局部特征。

2.1.3 池化层

池化层是一个利用池化函数 (pooling function) 对网络输出进行进一步调整的网络层。池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出。常用的池化函数包括最大池化 (max pooling) 函数 (即给出邻域内的最大值) 和平均池化 (average pooling) 函数 (即给出邻域内的平均值) 等。但无论选择何种池化函数，当对输入做出少量平移时，池化对输入的表示都近似不变 (invariant)。局部平移不变性是一个很重要的性质，尤其是当我们关心某个特征是否出现而不关心它出现的位置时。

池化层同卷积层类似，具有三个比较重要的参数：pool-size，strides 和 padding，分别表示池化窗口的大小，步长以及是否对图像的外侧进行补零操，池化层同时也能够提高网络的计算效率。

2.1.4 全连接层

全链接层（Fully-connected or Dense Layer 的目的就是将我们最后一个池化层的输出连接到最终的输出节点上。例如，最后一个池化层的输出大小为，也就是有个节点，对于手写数字识别的问题，我们的输出为 0 至 9 共 10 个数字，采用 one-hot 编码的话，输出层共 10 个节点。例如在 LeNet 中有 2 个全连接层，每层的节点数分别为 120 和 84，在实际应用中，通常全连接层的节点数会逐层递减。需要注意的是，在进行编码的时候，第一个全连接层并不是直接与最后一个池化层相连，而是先对池化层进行 flatten 操作，使其变成一个一维向量后再与全连接层相连。

2.1.5 输出层

输出层根据具体问题的不同会略有不同，例如对于手写数字识别问题，采用 one-hot 编码的话，输出层则包含 10 个节点。对于回归或二分类问题，输出层则仅包含 1 个节点。当然对于二分类问题，我们也可以像多分类问题一样将其利用 one-hot 进行编码，例如表示类型 0，表示类型 1。

2.2 AlexNet（本章节作者：李振国）

2.2.1 背景

在神经网络领域有一个为人所熟知的数据集叫做 ImageNet，该数据集包含 120 万张分辨率不同的图片，分为 1000 各类别作为神经网络的训练。其中验证集、测试集和训练集的比为 1：3：20。由该数据集所在网站举办的 ImageNet LSVRC 比赛作为一个判断神经网络模型优劣与否的依据。

在 2010 年的第一届比赛中，由 Hinton 团队设计的 ALexNet 获得了冠军，也正是该模型引出了之后其他更优秀和高效的网络模型。但尽管是抛砖引玉之作，AlexNet 的分类能力也已经相当的出色。在该数据集下该网络的 top1 和 top5 的错误率为分别为 37.5%和 17.0%。

2.2.2 网络结构

AlexNet 网络模型 AlexNet 共包含了 8 个学习层，即含权重矩阵的层——5 个卷积层和 3 个完全连接层。最后一个全连接层的输出被送到一个 1000 路的 softmax，他将产生一个超过 1000 各类标签的分布。

AlexNet 最大化了多项 logistic 回归目标，相当于最大化了在预测分布下正确标签的对数概率的训练案例的平均值。在卷积层中，在第二、四、五层卷积层的核至于前一层的核映射连接，且该核映射位于同一 GPU 上。第三层卷积层的核与第二层的所有核映射连接。完全连接层的神经元与前一层的所有神经元相连。在每一层的输出都是用 ReLu 激活函数进行映射。

2.2.3 模型介绍

AlexNet 模型使用了一些较新的技术，包括 ReLu 激活函数、Dropout、LRN、平均池化、GPU 并行计算等。

- 使用 ReLu 激活函数替代了 Sigmoid，并验证其效果在较深层网络中优于 Sigmoid。解决了 Sigmoid 在网络中的梯度弥散问题。
- 在避免过拟合上，首次将 Drpout 的思想应用在神经网络中，使之更加具有生物的遗忘特性，提高了网络的准确率。
- 一改以往神经网络的平均池化为最大池化，避免平均池化对重点区域的干扰，使得网络能够更加高效的提取到输入的特征。
- 由于 ReLu 函数不像 tanh 和 sigmoid 一样有有限的值域区间，所以需要对被激活的神经元进行抑制，借用侧抑制的生物概念提出了局部响应归一化（LRN）。局部响应归一化就是借鉴侧抑制的思想来实现局部抑制，使得其中响应大的值变得相对更大），并抑制其他较小反馈的神经元，增强了泛化能力。
- 由于 GPU 的简单运算、并行和高效的计算特性，AlexNet 选择使用 GPU 训练网络模型。解决了 CPU 在处理神经网络训练时大量简单矩阵运算的低效问题。
- 数据增强。将原始 256*256 的图像截取为 224*224 的区域，并进行水平翻转以增大输入的数据量。保证网络模型既有足够的样本进行训练，又减少了过拟合的可能性，大大提高了模型的泛化能力。

2.3 VGG（本章节作者：李克勤）

2.3.1 背景

模型的名称—— “VGG” 代表了牛津大学的 Oxford Visual Geometry Group。由于它的简洁性和实用性，当这个模型被提出时，马上成为了当时最流行的卷积神经网络模型。它在图像分类和目标检测任务中都表现出非常好的结果。VGG 是基于 Alexnet 网络的，VGG 在 Alexnet 基础上对深度神经网络在深度和宽度上做了更多深入的研究，为了更好的探究深度对网络的影响，必须要解决参数量的问题，更深的网络意味着更多的参数，训练更困难，使用大卷积核时尤其明显。业界普遍认为，更深的网络具有比浅网络更强的表达能力，更能刻画现实，完成更复杂的任务。

2.3.2 VGG 特点

VGG 网络的特点

- 结构简单，作者将卷积核全部替换为 3×3（极少用了 1×1）；相比于 AlexNet 的池化核，VGG 全部使用 2×2 的池化核。
- 参数量大，而且大部分的参数集中在全连接层中。网络名称中有 16 表示它有 16 层 conv/fc 层。
- 合适的网络初始化和使用批量归一 (batch normalization) 层对训练深层网络很重要。

- VGG-19 结构类似于 VGG-16，有略好于 VGG-16 的性能，但 VGG-19 需要消耗更大的资源，因此实际中 VGG-16 使用得更多。由于 VGG-16 网络结构十分简单，并且很适合迁移学习，因此至今 VGG-16 仍在广泛使用。

2.3.3 VGG 结构

VGG 结构模型图 结构图介绍：

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图 1: VGGNet。图片来源 [1]

- conv3-64 是指第三层卷积后维度变成 64，同样地，conv3-128 指的是第三层卷积后维度变成 128；
- input（224x224 RGB image）指的是输入图片大小为 224x244 的彩色图像，通道为 3，即 224x224x3；
- maxpool 是指最大池化，在 vgg16 中，pooling 采用的是 2x2 的最大池化方法；
- FC-4096 指的是全连接层中有 4096 个节点，同样地，FC-1000 为该层全连接层有 1000 个节点；
- padding 指的是对矩阵在外边填充 n 圈，padding=1 即外边缘填充 1 圈，5x5 大小的矩阵，填充一圈后变成 7x7 大小；在进行卷积操作的过程中，处于中间位置的数值容易被进行多次的提取，但是边界数值的特征提取次数相对较少，为了能更好的把边界数值也利用上，所以给原始数据矩阵的四周都补上一层 0，这就是 padding 操作。(6)：vgg16 每层卷积的滑动步长 stride=1，padding=1。

VGG 是一个良好的特征提取器，其与训练好的模型也经常被用来做其他事情，比如计算 perceptual loss(风格迁移和超分辨率任务中)，尽管现在 resnet 和 inception 网络等等具有很高的精度和更加简便的网络结构，但是在特征提取上，VGG 一直是一个很好的网络。

2.4 ResNet（本章节作者：何锋）

2.4.1 背景

在 VGGNet 和 Inception 出现之后，研究者们为追求更加优越的性能而不断增加网络层数，但是随着网络层数的加深，网络却越发难以训练，一方面会产生梯度消失现象；另一方面越深的网络返回的梯度相关性会越来越差，接近于白噪声，导致梯度更新也接近于随机扰动。[2]

由微软研究院的 Kaiming He 等四名华人提出的 ResNet(Residual Neural Network) 较好的解决了这个问题，通过使用 ResNet Unit 成功训练出 152 层的神经网络，并获得了 2015 年 ImageNet 分类任务的第一名。此后的分类、检测、分割等任务也大规模使用 ResNet 作为网络骨架。

2.4.2 结构思想

ResNet 的思想在于引入了一个深度残差框架来解决梯度消失问题，即让卷积网络去学习残差映射，而不是期望每一个堆叠层的网络都完整地拟合潜在的映射（拟合函数）。如下图所示，对于神经网络，如果我们期望的网络最终映射为 H(x)，左侧的网络需要直接拟合输出 H(x)，而右侧由 ResNet 提出的子模块，通过引入一个 shortcut（捷径）分支，将需要拟合的映射变为残差 F(x):H(x)-x。ResNet 给出的假设是：相较于直接优化潜在映射 H(x)，优化残差映射 F(x) 是更为容易的。

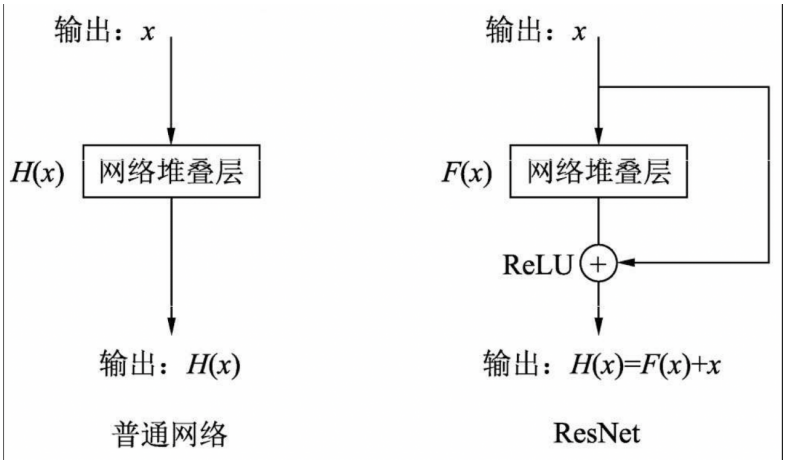


图 2: ResNet 图片来源 [2]

这种思想，即 Highway Network 的思想。简单来说就是在网络中添加了直连通道，在网络结构进行一个非线性变化的同时也保留了之前网络层的一定比例的输出。

2.4.3 网络优化点

深度学习对于网络深度遇到的主要问题是梯度下降和梯度爆炸，传统对应的解决方案则是数据的初始化 (normlized initializatition) 和正则化 (batch normlization)，但是这样虽然解决了梯度的问题，梯度加深了，但却带来了另外的问题，就是网络性能退化问题，深度加深了，错误率却上升了，而残差用来设计解决退化问题，其同时也解决了梯度问题，更使得网络的性能也提升了。

相比传统的网络来说，ResNet 也很容易训练，而且前向和反向传播都可以保证，线性使得网络加深，可以达到深层网络的精度提升并且可以被移植到其他问题中。残差元的主要设计有两个，快捷连接和恒等映射，快捷连接使得残差变得可能，而恒等映射使得网络变深，而恒等映射主要有两个：快捷连接为恒等映射和相加后的激活函数。残差映射也更容易优化。

2.5 卷积神经网络工作原理（本章节作者：李克勤）

卷积神经网络是人工神经网络的一种，卷积神经网络是机器深度学习中的一种“前馈神经网络”，前馈是信号量的输入获得，到输出过程是往前方传导的，简言之信号是往前方传递的，所以称之为前馈。前馈用以神经网络的计算输出，不对神经网络调整，每一层中每一个神经元算出该层的输出并向下一层传递到输出层，进而计算出网络的输出结果。Back Propagation 神经网络，即 BP 神经网络。反向传播训练神经网络权值和阈值的调整。网络前向传递计算输出结果时与正确结果存在误差，因此需要 Back Propagation 调整神经网络的前向计算过程。卷积神经网络本质上是一种输入到输出的映射网络，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对卷积神经网络加以训练，神经网络就具有输入输出之间的映射能力。卷积神经网络主要包括四个操作。

2.5.1 卷积工作原理

卷积的目的：为了从输入图像中提取特征。卷积可以通过从输入的一小块数据中学到图像的特征，并可以保留像素间的空间关系。滤波器在原始输入图像上的作用是特征检测器。通过在图像上滑动滤波器并计算点乘得到矩阵叫做“卷积特征（Convolved Feature）”或者“激活图（Activation Map）”或者“特征图（Feature Map）”。通过在卷积操作前修改滤波矩阵的数值，我们可以进行诸如边缘检测、锐化和模糊等操作——这表明不同的滤波器可以从图中检测到不同的特征，比如边缘、曲线等。在实践中，CNN 会在训练过程中学习到这些滤波器的值（尽管我们依然需要在训练前指定诸如滤波器的个数、滤波器的大小、网络架构等参数）。我们使用的滤波器越多，提取到的图像特征就越多，网络所能在未知图像上识别的模式也就越好。

2.5.2 非线性处理（ReLU）工作原理

在每次的卷积操作后都使用了一个叫做 ReLU 的操作。ReLU 表示修正线性单元（Rectified Linear Unit），是一个非线性操作。ReLU 是一个元素级别的操作（应用到各个像素），并将特征图中的所有小于 0 的像素值设置为零。ReLU 的目的是在 ConvNet 中引入非线性，因为在大部分的我们希望 ConvNet 学习的实际数据是非线性的（卷积是一个线性操作——元素级别的矩阵相乘和相加，所以我们需要通过使用非线性函数 ReLU 来引入非线性。其他非线性函数，比如 tanh 或者 sigmoid 也可以用来替代 ReLU，但 ReLU 在大部分情况下表现是更好的。

2.5.3 池化或亚采样工作原理

空间池化（Spatial Pooling）（也叫做亚采用或者下采样）降低了各个特征图的维度，但可以保持大部分重要的信息。空间池化有下面几种方式：最大化、平均化、加和等等。对于最大池化（Max Pooling），我们定义一个空间邻域（比如，2x2 的窗口），并从窗口内的修正特征图中取出最大的元素。除了取最大元素，我们也可以取平均（Average Pooling）或者对窗口内的元素求和。在实际中，最大池化被证明效果更好一些。池化函数可以逐渐降低输入表示的空间尺度。特别地，池化：

- 使输入表示（特征维度）变得更小，并且网络中的参数和计算的数量更加可控的减小，因此，可以控制过拟合。
- 使网络对于输入图像中更小的变化、冗余和变换变得不变性（输入的微小冗余将不会改变池化的输出——因为我们在局部邻域中使用了最大化/平均值的操作。
- 帮助我们获取图像最大程度上的尺度不变性。它非常的强大，因为我们可以检测图像中的物体，无论它们位置在哪里。

2.5.4 分类（全连接层）工作原理

全连接层是传统的多层感知器，在输出层使用的是 softmax 激活函数（也可以使用其他像 SVM 的分类器，但在本文中只使用 softmax）。“全连接（Fully Connected）”这个词表明前面层的所有神经元都与下一层的所有神经元连接。卷积和池化层的输出表示了输入图像的高级特征。全连接层的目的是为了使用这些特征把输入图像基于训练数据集进行分类。除了分类，添加一个全连接层也（一般）是学习这些特征的非线性组合的简单方法。从卷积和池化层得到的大多数特征可能对分类任务有效，但这些特征的组合可能会更好。从全连接层得到的输出概率和为 1。这个可以在输出层使用 softmax 作为激活函数进行保证。softmax 函数输入一个任意大于 0 值的矢量，并把它们转换为 0 1 之间的数值矢量，其和为 1。把它们组合起来——使用反向传播进行训练。卷积 + 池化层的作用是从输入图像中提取特征，而全连接层的作用是分类器。

2.5.5 总结

卷积网络在本质上是一种输入到输出的映射，它能够学习大量的输入与输出之间的映射关系，而不需要任何输入和输出之间的精确的数学表达式，只要用已知的模式对卷积网络加以训练，网络就具有输入输出对之间的映射能力。CNN 一个非常重要的特点就是头重脚轻（越往输入权值越小，越往输出权值越多），呈现出一个倒三角的形态，这就很好地避免了 BP 神经网络中反向传播的时候梯度损失得太快。卷积神经网络 CNN 主要用来识别位移、缩放及其他形式扭曲不变性的二维图形。由于 CNN 的特征检测层通过训练数据进行学习，所以在使用 CNN 时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。

2.6 卷积神经网络的应用（本章节作者：李振国）

自从卷积神经网络问世以来，这项技术逐渐被应用到各个领域。迄今为止，卷积神经网络已经成为我们身边随处可见甚至必不可少的一项技术。从谷歌识图到人工智能小度、siri，从语音识别到智能编辑，从气象预测到生物序列预测，无一不是通过各种模型对先验知识进行学习后才能完成这些以往甚至不可完成的任务。接下来本文将展示卷积神经网络在图片识别和序列预测方向的应用。

2.6.1 图像识别

卷积神经网络方法融合了计算机视觉领域的特征提取以及机器学习领域的分类器设计。通过以上两种方式卷积神经网络就能够根据输入进行图像识别的功能。而我们在图像识别方向的网络模型一般都会在以上两种方式上进行改进从高提高准确率。一个优秀的网络模型能够从不同的输入中提取到有效的特征信息，即 hindon 所说的：旋转不变性、镜面不变性、平移不变性等。除此之外还需要有良好的泛化能力的分类器，常见的分类器有 softmax、SVM 等。

卷积神经网络在图像识别上可以说是较早的应用，随着模型的发展和数据集的丰富，图像识别逐渐变得精确且便捷。传统的多层神经网络由输入层、多层隐藏层以及输出层组成。在图像处理问题中，输入层的每个神经元可以表示单个像素灰度值，但这存在一定的问题：首先相邻两层的神经元是全连接的，这样导致了训练参数的指数级增长，进而影响到整个模型项目的效率；其次，图像作为一个平面输入，肯定会损失一定的立体特征，这样会使得图像识别的困难。

图像识别作为卷积网络早期实现的应用在当下已经十分完善了，本文将使用 AlexNet 对图片进行分类并使用 CAM 进行可视化。

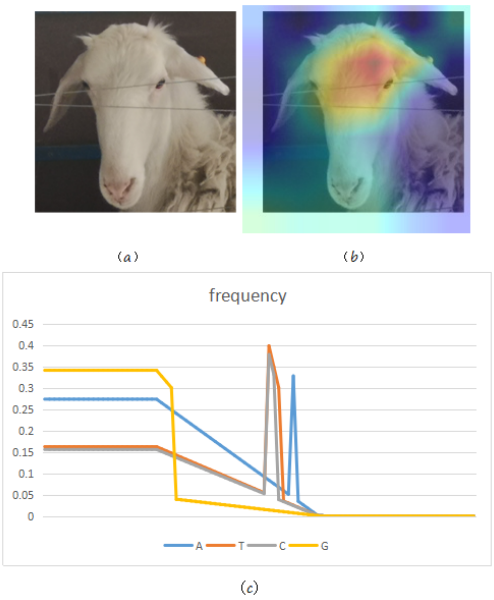


图 3: result visualizations

2.6.2 序列预测

在卷积神经网络问世前，有许多问题是无法使用计算机进行计算解决的。一方面在于需要计算的数据已经膨胀到一个惊人的地步，以至于常规人工的处理方式根本不能够完成这样的任务；另一方面在于硬件的发展已经到了一个瓶颈期，很难再有创世纪性的新型数据处理的硬件支持的提升。是故研究员们将目光转向了卷积神经这个黑盒子。卷积神经网络很好的解决了监督学习需要大量人力物力的难题，将人脑的记忆机制和机器的大数据处理的优点很好的结合起来。基于此，生物学家们开始着眼使用卷积神经来预测基因表达或交互等人类难以做到的事情，并在实践中取得了一个又一个的突破。

本文将用卷积神经网络模型进行训练，并给出一个 3000bp 的序列进行预测序列上每个位点的重要程度。

2.7 前景展望 (本章节作者：李振国)

随着时代的进步与发展，CNN 系列模型逐渐也不能够满足日新月异的技术需求，所以 RNN、GNN 等一些更加新颖、更有针对性的模型被提出。比如适合做自然语言处理的 RNN 模型，以及适合处理非欧几里得数据的 GNN 模型。

2.7.1 RNN

在自然语言处理中会有这样一种情况：句子的前后之间有着某种关联。而有着这种关联的句子如果在适当的模型中进行训练就能够实现预测下一个词出现的可能性。但典型的 CNN 网络并不能通过训练获取这种前后关联的时序关系，它不能保持之前所习得的知识。而 RNN 就解决了这个问题，RNN(Recurrent Neural Network) 时包含循环的网络，它允许了信息的持久化。

RNN 的发展自 LSTM 模型开始，其特点在于时序性较强，非常适用于时序数据的建模。从自然语言处理的角度，典型的神经网络往往采用加和的方法对句子进行判断情感极性，比如”I do not love him.” 其中在”not” 上被判作负值，”love” 会被判作正值，但句子的情感极性是明显偏贬义的，而直接加和往往会失掉前后关联，比如 “do not” 是对于 “love” 的否定。但 LSTM 模型能够更好的捕获较长距离的依赖关系，因为 LSTM 可以通过训练过程学到记忆或是遗忘某些信息。但是，尽管 LSTM 对于语义的前后联系已经有着一

定的作用，但很明显其输入和训练都是基于时序的。也就是说，它并不能编码从后到前的信息，即如果我们对某个词的形容出现在后半部分，那么 LSTM 并不能够准确地判断该句子。所以 BiLSTM(Bi-directional Long Short-Term Memory) 也就应运而生了，它是由前向 LSTM 和后向 LSTM 组合而成，能够更好地捕捉到双向的语义依赖。

虽然 BiLSTM 的效果优于 LSTM，但其训练效率较低，耗费时间较长，仍有优化空间。RNN 较为人所熟知的就是 Transformer 模型，该模型为 RNN 的集大成者（至少到目前为止）。自从 Attention 机制提出后，序列模型的准确度都得到了一定的提升，而 Transformer 模型区别于 LSTM 作为一个全 Attention 的结构，在翻译上取得了更好的效果。在 2020 年，双向 Transformer 模型——即 BERT——也被提出，相信会在自然语言处理和序列预测方向取得较好的成果。

2.7.2 GNN

预测与决策，作为机器学习的核心使命，一直都是神经网络模型所一直努力的方向。而图神经网络在卷积网络、循环网络的基础上，提出了新的研究方向：用于处理图数据的神经网络结构。

传统的神经网络学习方法在提取欧氏空间数据的特征上取得了巨大成功，但在实际中有许多的场景数据产生于非欧氏空间的，在这部分数据上，传统神经网络方法并不能够表现得很好。因为图的复杂性和不规则形使得卷积操作无法直接进行计算，所以 GNN 就应运而生。GNN 的主要优势在于：

- 强大的图数据拟合能力。能够在图数据中拟合所需的数据从而能够做出预测。
- 强大的推理能力。相较于传统的关系三元组的建模方式，GNN 能够对表征语义关系的网络进行整体性的建模，学习到更复杂丰富的语义信息，从而提升推理能力。
- 与知识图谱相结合。可以将先验知识以端到端的形式高效的嵌入到模型中。

基于这些优势，GNN 模型已经广泛应用到各行各业的生产活动中，比如 3D 视觉、社交网络推荐系统、视觉推理等。

3 后记

3.1 课程论文构思和撰写过程（本章节作者：李振国）

提及神经网络，ImageNet LSVRC 比赛是肯定绕不开的。在历届比赛中出现了许多优秀的模型，直到 2017 年由于准确率已经足够高（acc of top1 $>$ 80%）且模型已经开始过拟合而停止举办，它为神经网络领域提供了极大的助益。本文主要以时间顺序对历届出现的冠军模型进行解析，深入理解了领域前辈们对于优化模型的思路和实现。文章首先分模型进行撰写前两部分，再完成代码实现部分，交换彼此心得后撰写摘要部分。在撰写论文过程中可谓受益匪浅。

3.2 所参考主要资源（本章节作者：何锋）

本文主要是以周志华所著的《机器学习》、董洪义所著的《深度学习之 PyTorch 物体检测实战》以及提出网络模型架构的论文原文为参考，在此基础上对于卷积神经网络的框架知识借鉴了博客上一些博主的思想观点，并结合自己对于框架知识的理解编辑而成，并且在自己的研究方向上进行了相关的实验验证。本文中的代码为个人编写，源码`https://gitcode.net/abc_ambition/cam`

3.3 代码撰写的构思与体会（本章节作者：李振国）

代码部分主要包括模型的建立、数据预测以及可视化，其中模型部分使用了 torch 框架，可视化输入基于给出的输入数据的预测结果。

3.3.1 cam 可视化原理

CAM 可视化的原理在于取输出类别的最后一个卷积层特征图输出乘以模型输出相对于最后一层卷积层激活输出的梯度，将梯度做全局平均池化，即

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{y^c}{A_{ij}^c}$$

其中，α_k^c 代表 c 类别下模型输出相对于最末一层卷积层第 k 个特征图的梯度。$\frac{y^c}{A_{ij}^k}$ 代表未经过 softmax 的输出对第 k 个特征图的偏导。

需要注意的是，这样产生的热力图的二维得分矩阵中会有负值的出现，但我们只需要考虑对得分矩阵做 ReLu 函数处理。即：

$$L_{Grad_CAM}^c = ReLu(\sum_k \alpha_k^c A^k)$$

3.3.2 应用实现

图片分类使用了已训练的模型给出合适参数到本文模型，输入一张 224X224 的图片到模型中，对其进行分类。模型的总共有 28 种类别，即 softmax 将在 20 个类别中进行判别输出。根据 CAM（Gradient-weighted Class Activation Map）梯度加权类激活映射的方式，与输出类别相关的二维特征分数网络，网格的每个位置表示在该类别下的重要程度。这一方式能够显著地观察到原始对象的那一部分对 CNN 模型最的分类决策起到了决定性作用。

序列的训练和预测我们使用了 FormatNet，通过 torch 框架进行模型架构，并用序列文件对模型进行训练，最后给出一串 3000bp 的序列进行预测。根据图像的权重可视化，我们将其移植到序列上面。将序列输入到模型中进行预测，判断序列中哪一部分对该基因所表达出来的特征影响最大。从热力图的例子来看，我们最后的得分是一个二维矩阵形式的得分矩阵。那么我们可以尝试将模型训练返回的该二维矩阵的得分在纵向上进行平均得到一个针对每个位点的不同得分。类似地，我们对最后的得分进行 relu 处理，将负值消除。对最终的得分矩阵做归一化。本文中由于训练样本较少，在结果上可能并不具有生物学意义。

3.4 人员分工（本章节作者：李克勤）

本文共分为三章。第一章概况部分由李振国构写本文的选题说明；何锋构写卷积神经网络的基本原理；李克勤构写卷积神经网络的应用范围。第二章第一节卷积神经网络结构由何锋撰写，主要包括卷积层、池化层、全连接层、输出层的讲解。第二章第 2、3、4 小节神经网络的发展进程由李振国讲述 CNN 模型中 AlexNet 的发展背景，结构以及模型介绍；李克勤讲述 CNN 模型中 VGG 的发展背景，特点以及结构；何锋讲述 CNN 模型中 ResNet 的发展背景及主要思想。第二章第 5 小节卷积神经网络的工作原理由李克勤撰写，主要包括卷积工作原理，非线性处理（ReLU）工作原理，池化工作原理和全连接层工作原理。第二章第 6、7 小节卷积神经网络的应用由李振国撰写，主要包括图像识别、序列预测等应用的讲解。第三部分后记主要由李振国总结课程论文的构思和撰写过程以及代码撰写的构思与体会；何锋总结论文所参考的主要资源；李克勤总结论文撰写过程中的人员分工情况。

参考文献

[1] Andrew Zisserman+ Karen Simonyan∗. *VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION*. Published as a conference paper at ICLR, 2015.

[2] 董洪义. 深度学习之 *PyTorch* 物体检测实战. 机械工业出版社, 2020.

[3] 周志华. 机器学习. In 神经网络, pages 97–120, 2016.

[4] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks. 2012.

4 附录

4.1 关键代码（本章节作者：李振国）

```
代码来源: https://gitcode.net/abc_ambition/cam
代码属原创, 参考代码来源https://github.com/jacobgil/keras-grad-cam
#### 图片识别模块
img = cv2.imread(path_img, 1)
img_input = img_preprocess(img)

#model = AlexNet(num_classes=56)
net = AlexNet(num_classes=28)
pthfile = 'AlexNet.pth'
#net = nn.DataParallel(net)
net.load_state_dict(torch.load(pthfile, map_location='cpu'))
net.eval()
print(net)

net.features[-1].register_forward_hook(farward_hook)
net.features[-1].register_backward_hook(backward_hook)

#forward
#img_input = img_input.reshape((224, 224))
output = net(img_input)
idx = np.argmax(output.cpu().data.numpy())
print("predict: {}".format(classes[idx]))

# backward
net.zero_grad()
# 初始化梯度为零
class_loss = output[0,idx]
class_loss.backward()

# cam
grads_val = grad_block[0].cpu().data.numpy().squeeze()
fmap = fmap_block[0].cpu().data.numpy().squeeze()

cam_show_img(img, fmap, grads_val, output_dir)

#### 序列预测模块
net = FormatNet()
pklfile = open('raw_data.pkl', 'rb')
pickle.load(pklfile)
net.eval()
```

```
41
42 fmap_block = list()
43 grad_block = list()
44
45 net.first_block[-1].register_forward_hook(farward_hook)
46 net.first_block[-1].register_backward_hook(backward_hook)
47
48 # forward
49 inp = onehot_data
50 inp = np.mean(inp, axis=0)
51 inp = np.swapaxes(inp, axis1=0, axis2=1)
52 print(inp.shape)
53 output = net(torch.tensor(inp, dtype=int), torch.tensor(inp, dtype=int))
54 idx = np.argmax(output.cpu().data.numpy())
55 print(output.shape)
56 print(output)
57
58 # backward
59 '''
60 net.zero_grad()
61 class_loss = output[0, idx]
62 class_loss.backward()
63 '''
64 net.zero_grad()
65 loss = torch.abs(1 - output.mean())
66 print(loss)
67 loss.backward()
68
69 #### 可视化模块
70 def cam_compute(fmap, grads):
71     grads = np.mean(grads, axis=0)
72     fmap = np.mean(fmap, axis=2)
73     print(grads.shape, fmap.shape)
74     #cam = np.zeros(fmap.shape[1:], dtype=np.float32)
75     #print(cam.shape)
76     out = np.dot(grads, fmap)
77     print(out.shape)
78     out = cv2.resize(out, (4,3000))
79     print(out.shape)
80     #cam = np.mean(out, axis=1)
81     cam = np.maximum(out, 0)
82     cam = cam/cam.max()
83     return cam
```

4.3 《注意力机制在机器翻译研究方向上的应用》——倪坤宇，杨永康，王博文

摘要：随着近年来数据爆炸式的增长与计算能力的提升，用户对如何从海量数据中快速提取有价值的信息有了更高的要求。机器翻译作为自然语言处理领域中的一个重要方向，最早期是从词典匹配词典结合语言学家知识的规则翻译，再到基于语料库的统计机器翻译。而随着人工智能与深度学习的逐渐发展，神经网络与注意力机制被引入到机器翻译任务中，并达到了不错的效果。本文意在研究注意力机制这一前沿手段在机器翻译方面的应用情况，我们结合使用 Tensoflowdataset 中的语料集并且使用 Tensorflow 框架实现了一个 Transformer 模型并将其应用到葡萄牙语到英语的翻译中，并且将代码上传到 Github 仓库中`https://github.com/awantedboy/data_work`。

注意力机制在机器翻译研究方向的应用

倪坤宇 [1], 杨永康 [1]，王博文 [1]

[1]College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

随着近年来数据爆炸式的增长与计算能力的提升，用户对如何从海量数据中快速提取有价值的信息有了更高的要求。机器翻译作为自然语言处理领域中的一个重要方向，最早期是从词典匹配词典结合语言学家知识的规则翻译，再到基于语料库的统计机器翻译。而随着人工智能与深度学习的逐渐发展，神经网络与注意力机制被引入到机器翻译任务中，并达到了不错的效果。本文意在研究注意力机制这一前沿手段在机器翻译方面的应用情况，我们结合使用 Tensoflowdataset 中的语料集并且使用 Tensorflow 框架实现了一个 Transformer 模型并将其应用到葡萄牙语到英语的翻译中，并且将代码上传到 Github 仓库中 [1]。

关键词：机器翻译，注意力机制，Transformer

1 概况

1.1 选题说明（本章节作者：杨永康）

随着人工智能技术的不断进步，现有机器模型已经基本达到了感知智能，正朝着认知智能前进。自然语言处理是智能认知的基础，是学界和工业界的研究热点。为了满足社会对各种语言的需求以及世界各国日益频繁交流更加的便捷，价格低廉的机器翻译研究正在逐渐兴盛。随着深度学习技术的持续提高，机器翻译也逐渐融合了这些方法和策略，并且在多个任务中斩获不错的成绩。团队成员来自三个不同的专业，但研究方向均与自然语言处理相关，且我们在短文中已探究了 RNN 算法的基本原理与应用场景，因此希望在长文中更近一步探究注意力机制这一算法，以便日后能解决在研究工作中的实际问题。

相信通过本次作业，能为我们以后的科研工作打下坚实的基础。

1.2 该算法基本原理（本章节作者：杨永康）

注意力机制 (Attention Mechanism) 是人们在机器学习模型中嵌入的一种特殊结构，用来自动学习和计算输入数据对输出数据的贡献大小，其将数据源（Source）中的元素想像成是一系列的 <Key, value> 数据对组成，此时给定目标中某个元素的 Query，通过计算 Query 和各个 Key 的相似性或者相关性，得到每个 Key 对应 Value 的权重系数，然后对 Value 进行加权求和，即得到了最终的注意力数值。所以本质上注意力机制是对数据源（Source）中元素的 Value 值进行加权求和，而 Query 和 Key 用来计算对应 Value 的权重系数。注意力机制原本是在计算机图像视觉里使用的一种策略，之后在自然语言处理的领域被逐步推广使用。其具有速度更快、模型更少以及效果更好的三大优点。注意力机制的本质可以看作是一个寻址（addressing）的过程，主要可以分为三个步骤：

（1）首先用 $X = [x_1, \cdots, x_N]$ 表示为输入序列;

[1]https://github.com/awantedboy/data_work

（2）其次对 Query 和 Key 计算相似度，并归一化计算其值，用公式（1）进行计算。

$$\alpha_i = \text{softmax}\left(s\left(key_i, q\right)\right) \tag{1}$$

（3）将（2）中的权值与 value 值相乘并求和，如公式 (2) 所示。

$$\text{att}(q, X) = \sum_{i=1}^{N} \alpha_i X_i \tag{2}$$

其中 α_i 表示注意力分布 (概率分布)，$s(key_i, q)$ 表示注意力计算机制，有如下几种具体方式（公式 (3)）：

$$s(\text{Query}, \text{Key}) = \begin{cases} \text{Query} \cdot \text{Key} & \text{dot} \\ \text{Query} \cdot \text{W} \cdot \text{Key} & \text{general} \\ \tanh(\text{W}[\text{Query}; \text{Key}]) & \text{concat} \end{cases} \tag{3}$$

其中注意力分布 α_i，可以理解为在 key 值中查询 q 时，第 i 个值受关注的程度，通过“软性”的信息挑选策略对输入序列 $X(value)$ 进行处理。

1.3 机器翻译的基本任务（本章节作者：王博文）

机器翻译是自然语言处理中不可或缺的内容，有两个核心的任务需要解决，即自然语言理解以及生成。只有机器先理解了自然语言，并且成功地生成另一门语言，这样的机器翻译才是完整的。随着社会和科学水平的迅速进步，世界各国之间文化交流越发普遍。并且互联网时代的到来，导致不同语种之间的交流也更频繁，对语言翻译的需求逐渐变多，国内外各大公司都在不断地发展和完善自己的机器翻译系统，如网易有道、百度和谷歌等。由于相较于人工翻译价格昂贵、效率低以及一些其他的原因，机器翻译因其高效且低成本的特点，成为目前各大行业必不可少的组成部分。目前机器翻译总体上由统计机器翻译（Statistical Machine Translation, SMT ）以及神经机器翻译（Neural Machine Translation，NMT）这两个方向组成。

统计机器翻译，是一种基于统计学习知识生成目标语言的翻译策略，可以理解为信息传播的一个过程，对于相同的源语言可以对应多个目标语言的语句，只是选择概率最大的句子当做最终的结果。相对于规则的方法，统计机器翻译克服了翻译知识方面的限制，从最初以词为基础，逐步演化为以短语为基础的统计翻译，然后又结合句法结构信息，一步又一步的提高统计机器翻译的准确性。一般主要运用基础概率知识、贝叶斯公式等知识去构建基本的翻译模型，一般来说假定源语句 f ，对应的目标语句 e，通过如下公式可以得出最优翻译目标语句 e。

$$\arg\max_{e} P(e \mid f) \tag{4}$$

其中，$P(e \mid f) = \frac{p(f|e)p(e)}{p(f)}$，又因为在翻译中源句子 f 是已知的，所以 f 是固定值不会影响其概率大小。

在翻译模型之前需要通过单语的语料，进行语言模型的训练，为翻译模型提供准确的先验概率 $P(e)$，公式如下：

$$p(e) = p\left(w_1\right) p\left(w_2 \mid w_1\right) p\left(w_3 \mid w_1, w_2\right) \ldots p\left(w_n \mid w_1, w_2, \ldots, w_{n-1}\right) \tag{5}$$

随着句子长度的增加，上式计算变得十分耗费时间和出现数据的稀疏问题，因而，一般 n 不超过 4。句子 e 是由 $(e_1, e_2, \cdots, e_n)$ 按一定顺序组成，而 f 是由 $(f_1, f_2, \cdots, f_n)$ 按序组成，则那么条件概率 $P(e)$ 这个问题可以看成 $(e_1, e_2, \cdots, e_n)$ 到 $(f_1, f_2, \cdots, f_n)$ 的概率问题。理论上当 n 值越大，相关依赖词就越多，模

型就可以获得更多信息量和获得更好的翻译效果，对未来词的预测也就越准确，但是会出现数据稀疏和模型参数规模过大的问题；当 n 的值很小时，由于受到 n 的限制，模型很难捕获远距离词之间存在的依存关系。虽然统计学习模型给出了不错的翻译模型构建思路，但是针对数据稀疏的问题，难以捕获句子中长距离依存关系等难点都没有很好的解决方案，正因为如此统计机器翻译开始逐渐被以神经网络为基础的翻译模型所取代

神经机器翻译伴随深度学习的崛起不停地完善，其中一类相比统计机器翻译模型，其结构更加简洁但是仍然保留了统计机器翻译的大致框架，相对应只是用深度学习的框架替换了中间的部分模块。而另外一类模型，几乎抛弃了之前的统计机器翻译的基本结构，代替的是端到端（End to End）算法结构的一种变形框架–序列到序列（Sequence to Sequence，Seq2Seq）。Seq2Seq 模型用于神经机器翻译，主要是通过使用编码-解码（Encoder-Decoder）机制来完成的，其中，序列一般是一句话（文章片段或独立的一句话），通过 Seq2Seq 框架去生成目标语言的句子。相比较过去的各类翻译策略，神经机器翻译优势更加明显，生成具有更高的质量目标语句。

1.4 神经网络在机器翻译方面的应用（本章节作者：王博文）

人工神经网络（artificial neural network，ANN），简称神经网络（neural network，NN），是一种模仿生物神经网络的结构和功能的数学模型或计算模型。神经网络以人造神经元为基础，各神经元相互连接，完成信号传输、接收和处理。在 ANN 实现中，人造神经元间传输信号为实数，随着学习深度的加深，该参数发生一定变化，每个神经元均对应一个阈值，当总信号高于阈值时借助激励函数完成信号计算。一般情况下，人造神经元为多层结构，每层可能具备不同的转换处理功能。信号从输入层进入再到输出层输出，期间经历多次穿层活动，最终输出的结果受拓扑结构影响，节点间的权重反映节点上的映射关系。该模型拟合复杂非线性函数的功能极强，依靠带有标记的数据即可完成判断、识别活动。例如，在识别图像中，人工神经网络在深度学习过程中，记忆对象特征，进而判断图像是否为目标图像。目前基于人工神经网络翻译的方式大致分为三种，基于循环神经网络的机器翻译，基于卷积神经网络的机器翻译和基于注意力机制的机器翻译。

基于循环神经网络的机器翻译是 2013 年，Kalchbrenner 等人提出了一类连续句子级别的基于神经网络的翻译模型。该模型的短语和句子的表示，不依赖于对齐方式或者短语翻译单元。该模型利用卷积神经网络（Convolution Neural Network，CNN）把提供的源语言编码为连续的一个向量，再用循环神经网络对该向量进行解码转化为目标语言。其解决了统计机器翻译中长距离重排序问题，为后续的纯神经网络进行机器翻译打下了基础. 由于反向传播（Back-Propagation）算法是循环网络训练的基础，通过相对应的时间序列进行反向传播误差信号，并且通过计算各个时刻的梯度去更新网络中参数的权值。但是在实际的训练过程中，依然会出现“梯度消失”和“梯度爆炸”的情况，使得 Kalchbrenner 等人的提出的模型并未取得理想的性能。在这基础之上，2014 年 Sutskever 等人在长短时记忆（Long-Short Term Memory, LSTM）结构的循环网络基础上，发明了 Seq2Seq 架构的神经翻译模型。由于长短时记忆结构采用了特殊的门控机制（Gate Mechanism），使得整个网络中的“梯度消失和爆炸”的问题得到了一定程度的解决，而且更好的解决了句子中存在长距离依存的问题。同年，Google 的 Cho 的等人提出了一种更加简洁的网络结构，其由一个状态单元和二个门控单元组成的门控循环单元（Gate Recurrent Unit, GRU）。并且在许多神经机器翻译的项目中，其收敛速度和性能取得了更加优秀的表现。循环神经网络模仿人类的翻译过程，在一定的程度上，具有良好的时间序列的建模能力。但是不论是原始 RNN 还是后来的 LSTM、GRU 等都没有脱离整体网络时序性的束缚，不可以进行并行训练，导致训练的效率十分的低。而且在长句子中不能学习到有效地全局信息，对长距离依赖的难点解决的不够全面。

基于卷积神经网络的机器翻译可以定位到 1962 年，Hubel 与 Wiesel 等人对生物猫咪脑内视觉神经的实验研讨。基于此研讨 1980 年日本研究人员福岛邦彦，发明了 Neocognitron 模型，其包含了现在卷积网

络的卷积层以及池化层等网络架构。1989 年，LeNet-5 模型被 Lecun 等人提出，通过 BP 算法结合这个网络结构的训练，在图像分类的任务中获得优异的成果。即便如此，这时的卷积神经网络的整体效果在现实中的一些问题比不上 SVM、Boosting 等统计学习算法，而且训练起来比较的有难度。卷积神经网络真正的得到重视和发展于 2012 年，Hinton 带领的小组在 ImageNet 图像识别大赛中提出的 Alexnet 模型，取得了巨大成功一时间颠覆了图像识别领域。随后卷积神经网络经过了许多的学者改进，在多个领域都大放异彩，Gehring 等人在 2017 年提出了完全基于卷积神经网络的一种编码解码的神经机器翻译模型。目前，卷积神经网络在自然语言处理范畴的文本分类、机器翻译以及语音识别等多个方向都得到了普遍的推广。

1.5 注意力机制在机器翻译方面的应用（本章节作者：杨永康）

初期的基于 Seq2Seq 框架的机器翻译，都是编码器生成一个大小不变的上下文 (context) 向量 C ，解码器通过使用该固定长度的向量作为解码器输入然后输出最终的答案序列。通常编码器和解码器都是通过最大化每对输入空间序列和输出空间序列的最大似然估计值进行训练。有一个较为明显的缺陷，当编码器的输出固定长度的上下文向量维度过低，而翻译输入的语句过长的情形下，很难进行表达全部的输入信息。因而，Bahdanau 等人发明了可变的上下文向量计算方法，并且，他们将注意力机制带入模型之中，使得模型可以选择相关度更高的输入，提升了模型的区分辨别能力和翻译的整体性能。在 Bahdanua 等人的基础上，斯坦福大学的 Luong 等人提出了关于注意力机制的各种变形，如计算方面只用到了隐藏层的向量，引入了即的对其位置策略，使得模型具有局部的注意力的能力。

Facebook 的 Gehring 等人，介绍了一个以卷积网络为基础的神经翻译的简单编码器框架。这种方法比循环网络更加并行化，并且提供了更短的路径来捕获源中的长期依赖关系。他们使用了源位置嵌入以及大小不同核的 CNN 进行注意力得分计算和输入聚合。他们的实验表明，相对于双向 LSTM 编码器，卷积编码器的性能与基准相当或更好。随后 Gehring 等人结合 Dauphin 等人提出的门控卷积网络 (Gated Convolution Network) 和多步注意力 (Multi-step Attention) 机制，发明了编码与解码器都以卷积神经网络为基础的神经翻译模型，在大型的基准数据集上其效果更好，并且速度快一个数量级。相较于递归网络，由于表示通过分层构造，因此他们的卷积策略能够更轻松地找到序列中的组成结构。

2017 年，Google 的 Vaswani 等人发明了一种全新简易的网络框架，模型仅以注意力机制和前馈网络为基础，没有使用一点 RNN 和 CNN 的网络结构。实验结果显示，Transformer 的翻译效果更有竞争力，同时具有更高的可并行性，除此之外，显著地减少了模型训练所需要的时间。Transformer 采用了多头自注意力 (Multi-head Self-attention) 结构以及前馈神经网络 (Feed Forward Network) 的组合，构建的编码和解码的神经机器翻译模型，其在机器翻译乃至深度学习领域引起了不小的响应。

但是，Transformer 模型其自身框架存在一些不完善，其一是自注意力机制每轮都需对输入序列计算所有序列间的相互注意力得分，其计算时间复杂度都是输入序列的平方; 其二是模型的输入的上下文是先前预设的固定长度，在微调阶段，模型不可以获得超过最大长度的上下文依赖，由于 Transformer 对长文本的截断和拆分的片段编码，导致存在上下文碎片化的难点 (Contextual Fragmentation)。2020 年，Kitaev 等人提出了 Reformer 模型，针对大规模的 Transformer 进行改进，主要提出了两个改进策略。首先，他们结合局部敏感哈希的方式进行注意力的计算，将其计算的复杂度从 $O(L^2)$ 下降到 $O(LlogL)$(其中 L 是输入的句子长度); 此外，还提出了可逆残差层去替代先前的残差，只需要激活一次内存在训练过程中的任意阶段，而不是堆叠的层数次。2019 年，Dai 等人发明了 Transformer-XL 模型，其可以学习到超出最大长度的上下文依赖，并且不会损坏序列在时间上的持续效果。其包含了两个部分: 分段级循环机制 (Segment-Level Recurrence Mechanism) 以及一种相对位置编码策略 (Positional Encoding Scheme)。该神经网络结构除了能够捕获长期依赖 (Long-Term Dependency)，而且可以解决上下文语境中碎片化的难点。

2 Transformer 模型原理与构建

2.1 背景 (本章节作业：倪坤宇)

现行的主流神经网络翻译模型都是采用” 编码器-解码器” 的大框架进行模型的搭建，每一个时间步产生的输出都将作为下一时间步的输入，但几乎所有的网络结构都是采用的循环神经网络 (RNN) 及其变体的神经网络，其本质上都是循环神经网络的结构。然后 Google 在 2017 年发明了全新的神经网络翻译模型 Transformer 模型与之前所有的主流模型相比有了很大的改变，Transformer 模型同样采用了” 编码器-解码器” 的大框架进行了模型搭建。但是与之前的翻译模型的区别在于完全摒弃了在当时主流的循环神经网络及其变种网络。该模型只单纯的采用了多头注意力机制和前馈神经网络，整个网络完全由注意力机制组成，准确地说，是一个完全由自注意力机制 (Self-Attention) 和前馈神经网络组成的解码器-编码器模型。得益于自注意力机制，Transformer 模型具有两个明显的优势:

1). 并行处理，大大提升训练速度。

2). 自注意力机制能更好地捕获全局信息，词与词之间的距离为 1，更好地表征长距离依赖。

2.2 相关技术 (本章节作者：倪坤宇)

2.2.1 编码器-解码器架构 (Encoder-Decoder)

我们使用 “编码器-解码器” 的结构来进行翻译转换工作，这标志着神经机器翻译建模思想的出现，当输入和输出均是不定长序列时，可以使用编码器-解码器来构建模型框架，其中编码器对应输入序列，解码器对应输出序列，编码阶段将整个源序列编码成一个向量，解码截断通过最大化预测序列概率来解码出整个目标序列。

该框架结构的实际含义在于，对于源语言序列 $X = (x_1, x_2, ..., x_n)$，目标语言序列 $Y = (y_1, y_2, ..., y_m)$，可以构建生成目标语言的概率如公式所示。

$$P(y_1, y_2, ..., y_m | x_1, x_2, ..., x_n) = \prod_{t=1}^{t=m} P(y_t | c, y_1, ..j., y_{t-1}) \tag{6}$$

其中，c 表示编码器阶段生成的固定维度的背景向量，其中包含了源语言序列的相关信息。公式后半部分针对每一个目标语言词汇的生成概率通过 $softmax$ 的方式计算得到，如下列公式所得

$$\begin{aligned} P(y_t | x, y < t; \theta) &= \frac{exp(\varphi(y_t, x, y < t, \theta))}{\sum_{y \in Y} exp(\varphi(y_t, x, y < t, \theta))} \\ &= \frac{exp(\varphi(v_{yt}, c_s, c_t, \theta))}{\sum_{y \in Y} exp(\varphi(v_y, c_s, c_t, \theta))} \end{aligned} \tag{7}$$

其中,v_y 表示目标语言词向量，y 表示目标语言词汇，c_s 表示源语言上下文向量，c_t 表示目标语言上下文向量，该式子的意义即通过已知源语言句子和已经生成的目标语言句子来预测当前的目标词。由于源语言句子和已经生成的目标语言句子非常稀疏，生成的句子会有多种结果，很难用传统的离散表示来建立这样的一种概率分布，所以神经机器翻译用连续表示对这个条件概率进行建模。式子中的 φ 函数定义了通过源语言以及生成的目标译文来生成当前目标语言词汇 y_n 的可能性，引入 softmax 函数的目的是为了保证函数值满足概率分布。

整个编码器-解码器结构由编码器和解码器两部分组成: 编码器部分可以将源语言序列进行编码，并输出一个固定长度的向量表示; 编码器的作用是将一个不定长的输入序列转化为定长的背景向量 c，其中包含了输入序列的相关信息。

编码器部分主要分为三步进行:

1). 首先将输入源语言序列进行 one-hot 编码表示，将源语言句子 $X = x_1, x_2, ..., x_n$ 中的每个 x_i 表示成一个列向量 w_i，该向量的维度等于词汇表大小 $|V|$，且只有在词汇表对应位置的索引值所代表的维度上值为 1，其余为 0。

2). 将上一步得到的向量映射到低微语义空间构成词向量，由于仅仅使用 One-hot 编码得到的向量过于稀疏，对语义相似性无法描述，需要采用分布式表示的词向量模型将其映射到的低维语义空间，生成一个固定维度的词向量。映射矩阵表示为 $C \in R^{K*|V|}$, 第 i 个词的词向量表示为 s_i，且 $s_i = CW_i$，其中 K 表示词向量维度。

3). 采用神经网络编码生成源语言序列，计算公式如下所示:

$$h_i = \varphi(h_{i=1}, s_i) \tag{8}$$

其中 h_0 为全零向量，φ 表示非线性的激活函数，式子得到的 $h = (h_1, h_2, ..., h_n)$ 即为输入的 n 个源语言词汇的状态编码序列，也就是背景向量 c。

编码器的任务是获取下一个目标语言词的最大化概率，其处理流程如下

1). 对于序列中的某一个时刻 i, 根据由源语言序列生成的背景向量 C, 目标语言序列的第 i 个词 u_i，当前时刻的隐含层 z_i，来计算下一因此层的状态 z_{i+1}。通过下式进行进行计算:

$$z_{i+1} = \varphi(c, u_i, z_i) \tag{9}$$

其中 z_0 为全零向量，φ 表示非线性的激活函数。

2). 采用 softmax 函数对 $z_i + 1$ 进行归一化处理，得到目标语言序列第 i+1 个词的概率分布。

3). 根据得到的概率计算代价函数，重复上述步骤知道目标语言学列中所有词处理完毕。

下图同时演示了使用神经及其翻译模型将句子 “X Y Z” 翻译到句子 “A B” 的过程。首先编码器将源语言句子编码成固定长度的向量的表示 C，然后 C 做为解码器的输入。解码器会逐词地进行解码，初始状态地输入为 C 和起始符 <bos>，此时对应地解码器输出单词为 “A”，下一状态解码器的输入为 C 和上一时刻的输出 “A”，此时此刻对应的解码单词为 “B”，当解码器解码出句子结束符 <eos> 时停止解码。

同时也可以解释神经机器翻译的训练过程。若 “X Y Z ” 和 “A B” 为已有的一对平行语料，首先 “X Y Z” 输入编码器并获得一个固定长度的向量表示 C 作为解码器的输入。当 “X Y Z” 全部输入编码器后，解码器同时会输入 <bos> 作为起始状态，并解码输出第一个词。此时无论是否正确解码出了目标词 “A”，都不会作为下一时刻解码的输入。当解码下一时刻的词时，解码器的输入为向量表示 C 以及上一时刻的真实词 “A”。这样的在生成目标序列第 t 时刻的单词时，利用真实目标序列第 t-1 时刻的单词的方法，称为 Teacher-Forcing 方法。该方法有效解决了循环神经网络训练初期，模型预测的单词可信度较低，无法进行后续训练的问题。

2.2.2 文本特征表示

机器翻译作为自然语言处理任务的一种，首先需要将原始的语言输入转换为计算机可以处理的数字向量，这一过程称为 Embedding 词嵌入。

在自然语言处理任务种，Embedding 词嵌入即文本特征表示。文本特征表示是自然语言理解过程种最核心和基本的部分，准确地文本特征表示是机器翻译任务顺利解码并生成准确译文的关键

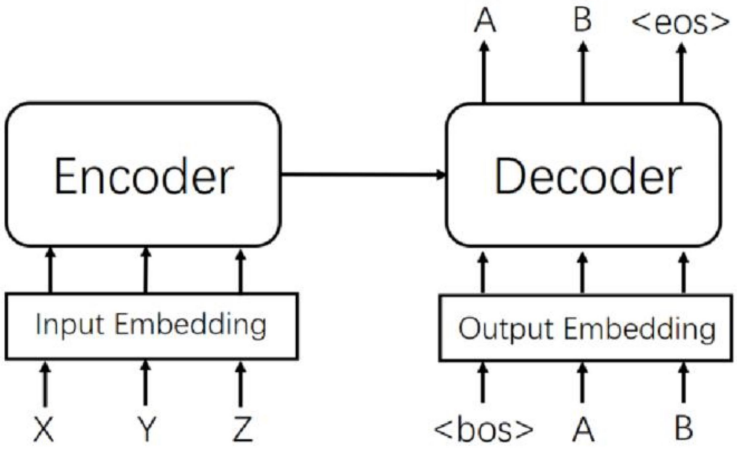


图 1: 编码器-解码器结构图 [2]

传统的文本特征表示使用独热码 (one-hot) 作为最早的词向量。但是独热向量仅使用 “0” 和 “1” 进行编码，其简单的编码方式会造成 “语义鸿沟”，同时，其稀疏性会带来 “维度灾难”。

因此有研究者基于分布式假设，即处于相似上下文中的词语具有相似的含义，提出了基于分布式的词向量模式，分布式的思想最早由 Hinton 提出，为之后各种词向量表示方法建立了基础。相比 One-Hot 方法，分布式方法将词语表示成一个定长的连续的稠密向量。向量空间中的距离则表示了词之间的相似关系，词向量也能包含更多信息。Word2vec 是 Google 公司在 2013 年所提出的开源项目，可以将意思相近的词语转化处理成稠密向量便于计算机所理解，从原始语料集中学习字词空间向量，是一种高效的预测模型。向量空间模型能将字词转化为连续值，并且可以将意思相近的词语映射到向量空间中相似的位置，实际使用 Word2vec 来训练语料也会发现，在向量空间中意思相近的词的位置更接近。Word2Vec 主要分为 Skip-gram 和 CBOW 两种模式。前者从目标字词出发推测原始语句，后者则以原始语句来推测目标字词。在实际的训练过程中还有两种降低复杂度的高效训练方法: 负采样以及 Hierarchical softmax 的方法。

2.2.3 分词处理

在采用词向量模型对语料数据进行处理生成词向量之前，首先需要对数据进行分词处理。在面向葡英文的机器翻译中，首先以对齐可信度为条件从双语字对齐语料库抽取对齐集合，再根据得到的对齐集合对双语语料的英文部分进行分词，最后将分词结果和单语分词工具得出的分词结果相结合，得到完整的分词结果。

2.2.4 自注意力机制

目前，自注意力机制已经成为机器翻译任务模型中的重要组成部分，这种机制忽略输入与输出之间的距离和顺序关系，显示地捕捉到了当前单词与所有单词之间的语义关系，因此可以使用并行化的计算方式，能够显著地减少训练时间，且在建模地依赖方面具有非常高的灵活性，自注意力机制能够捕捉到非常全面的语义信息，却无法把握句子的结构信息。因此在一定的特殊任务中，会将这种注意力机制与循环神经网络或卷积神经网络结合使用以便在计算效率和获取结构信息方面达到一定的平衡。

自注意力机制由其并行计算能力和建模的灵活性引起了广泛的研究兴趣，而自注意力机制中的多头注意力机制 (Multi Head Attention) 机制使模型能够从不同子空间关注的相应的信息。由于自注意力机制忽

[2]https://blog.csdn.net/program_developer/article/details/78752680

略句子中单词的位置因素，它可以显示地捕捉当前单词和句子中所有单词之间的语义关系，而 MHA 机制则是把输入序列映射到不同的子空间中，这些子空间分别采用自注意力机制，进一步增强了机器翻译模型的性能。自注意力机制具有以下优点:

1). 参数更少：相比于传统的 LSTM 模型，自注意机制复杂度更小，参数也更少，因此对计算能力的要求也就更低。

2). 速度更快：自注意力机制的每一步计算结果并不依赖于上一步的计算结果，解决了 RNN 无法并行训练的问题。

3). 效果更好：自注意力机制能够捕获全局单词之间的语义关系，有效解决了传统 RNN 网络中长距离信息被弱化的问题。

当使用自注意力机制处理每个单词 (即输入序列中每个元素) 时，比如对第 i 个单词进行计算时，自注意机制能够使其与序列中的所有单词进行关联，并计算出它们之间的语义相似度，这样的好处在于能够帮助挖掘序列中所有单词之间的语义关系，从而更准确地对单词进行编码。每一个注意力头对一组 n 元组地输入序列 $x = (x_1, x_2, ..., x_n)$ 进行操作，然后通过计算得到一组 n 元组地输出序列 $z = (z_1, z_2, ..., z_n)$。

对于输出序列中地元素 z_i，是由输入元素 x_i, x_j 经过线性变化后并计算其加权和得到:

$$z_i = \sum_{j=1}^{n} softmax(\frac{Q_i K_j^T}{\sqrt{d_k}})V_j \tag{10}$$

softmax 函数中，对输入元素进行线性变换增强了表达能力。softmax 分数决定了每个单词在当前位置所表达的注意力分数的大小。这里用值向量 V_j 乘以 softmax 分数是要宝石当前被关注的单词的值的完整性并淹没无关的单词。然后对这些加权的值向量进行求和，得到自注意力输出，这个输出会被发送到前馈神经网络层进行进一步计算。softmax 函数的计算方式如下:

$$softmax(a_{ij}) = \frac{exp(a_i j)}{\sum_{k=1}^{n} exp(a_{ik})} \tag{11}$$

Q,K,V 分别 query,key,value，它们分别是对计算注意力分数有用的抽象表示，d_k 是 key 的维度，除以 $\sqrt{d_k}$ 是缩放点积，这样可以使渐变更加稳定，Q,K,V 计算方式分别如下:

$$Q_i = x_i W^Q \tag{12}$$

$$K_j = x_j W^K \tag{13}$$

$$V_j = x_j W^V \tag{14}$$

如下图所示,W^Q,W^Q,W^V 是在训练过程中学习得到的矩阵，它们分别是 Q,K,V 的权重矩阵。每一个注意力头由属于自己特有的权重矩阵。值得一提的是，与传统的注意力机制有所区别，在自注意力机制中，同一个注意力头中，$W^Q = W^K = W^V$, 所以对于同一个输入元素 x_i，$Q_i = K_i = V_i$。

自注意机制使用 l 个注意力头，所有注意力头的输出 z_h 都被合并，然后进行线性变换得到每个子层的输出。多头注意力机制扩展了模型专注于不同位置的能力，例如，要翻译 “este é o primeiro livro que eu já li” 这句葡萄牙语时，需要知道 “este” 指代什么，多头注意力机制对于这样的情况能够提供非常大的帮助。多头注意力机制为注意力层提供了多个表示子空间，多头注意力机制提供了多个表示子空间，多头注意力机制提供了多组 Query，Key，Value 集合，这些集合都是随机初始化生成的，在训练之后，每个集合都将用于输入的嵌入，然后被投入到不同的表示子空间中。多头注意力机制的输出结果计算公式如下:

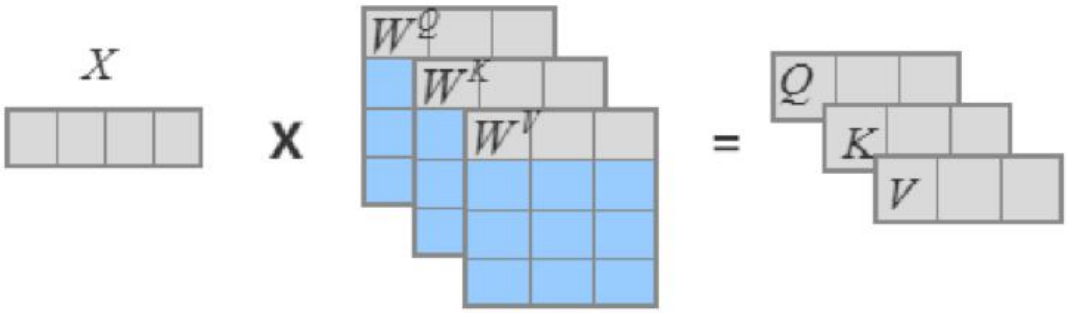


图 2: Q,K,V 计算过程 [3]

$$MultiHead(Z) = Concat(z_{head_1}, ..., z_{head_l})W^O \tag{15}$$

z_{head_i} 表示第 i 个注意力头的输出向量，Concat() 的功能时将所有注意力头的输出向量合并，W^O 是模型训练过程中产生的权重矩阵。多头注意力机制合并各个注意力头的输出，然后进行线性变化得出最终输出。

2.3 实验环境与模型设计

2.3.1 Python3(本章节作者：王博文)

Python 是一种解释型，面向对象，动态数据类型的高级程序设计语言，语法相对简洁，清晰，具有丰富和强大的类库，它常被称为胶水语言，能够很轻松的将其他语言制作的模块结合在一起。结合自己的感受而谈，Python 语言简单易学，同时具有强大的功能，比如在：Web 开发，Python 爬虫，数据分析，人工智能，系统运维等方面都具有很好的表现。同时 Python 具有轻巧的特点，运行环境安装简单，方便。

2.3.2 Tensorflow(本章节作者：王博文)

TensorFlow 是一个基于数据流编程的符号数学系统，被广泛应用于各类机器学习算法的编程实现。它的流行让深度学习门槛变得越来越低，只要你有 Python 和机器学习基础，入门和使用神经网络模型变得非常简单。TensorFlow 支持 Python 和 C++ 两种编程语言，再复杂的多层神经网络模型都可以用 Python 来实现，如果业务使用其他编程也不用担心，使用跨语言的 gRPC 或者 HTTP 服务也可以访问使用 TensorFlow 训练好的智能模型。

Tensorflow 拥有多层级结构，可部署于各类服务器、PC 终端和网页并支持 GPU 和 TPU 高性能数值计算，被广泛应用于谷歌内部的产品开发和各领域的科学研究。

Tensorflow 的优势：

1). 可用性：TensorFlow 工作流程相对容易，API 稳定，兼容性好，并且 TensorFlow 与 Numpy 完美结合，这使大多数精通 Python 数据科学家很容易上手。与其他一些库不同，TensorFlow 不需要任何编译时间，这允许你可以更快地迭代想法。在 TensorFlow 之上已经建立了多个高级 API，例如 Keras 和 SkFlow，这给用户使用 TensorFlow 带来了极大的好处

2). 灵活性：TensorFlow 能够在各种类型的机器上运行，从超级计算机到嵌入式系统。它的分布式架构使

[3]https://blog.csdn.net/ningyanggege/article/details/89812558

大量数据集的模型训练不需要太多的时间。TensorFlow 可以同时在多个 CPU，GPU 或者两者混合运行。

3). 效率：自 TensorFlow 第一次发布以来，开发团队花费了大量的时间和努力来改进 TensorFlow 的大部分的实现代码。随着越来越多的开发人员努力，TensorFlow 的效率不断提高。

4). 支持：TensorFlow 由谷歌提供支持，谷歌投入了大量精力开发 TensorFlow，它希望 TensorFlow 成为机器学习研究人员和开发人员的通用语言。此外，谷歌在自己的日常工作中也使用 TensorFlow，并且持续对其提供支持，在 TensorFlow 周围形成了一个强大的社区。谷歌已经在 TensorFlow 上发布了多个预先训练好的机器学习模型，他们可以自由使用。

2.3.3 TensorflowDataset(本章节作者：王博文)

Dataset 可以看作是相同类型“元素”的有序列表, 作用是为了更加高效便捷的读取数据。在实际使用时，单个“元素”可以是向量，也可以是字符串、图片，甚至是 tuple 或者 dict。Dataset 是 google 点名建议的读取数据的方式，所以在 tf 使用中有很重要的地位。上面这个定义其实已经描述了 dataset，他是一个可迭代对象，而且是有序的，而且每个元素的类型相同。在工作中为了方便理解，我一般把它看作 java 中的 List<T>，即声明了泛型的有序集合 List。从 python 的角度看的话，就是规定了所有元素类型必须相同的 list。

tf.data 包提供的 API 就是用来帮助用户快速的构建输入的管道 pipeline 的。以文本的输入为例，tf.data 提供的功能包括：从原始文本中抽取符号；将文本符号转化成查找表（embeddings）；把长度不同的输入字符串转化成规范的 batch 数据。总的来说，这个接口能够帮助用户轻松的应对大数据量的处理，以及不同格式的归一化处理。

2.3.4 模型构建与设计（本章节作者：倪坤宇）

Transformer 模型的整体结构包含了”解码器-编码器”结构与多头注意力层 (Multi-head attention layer)，一个是位置前馈层，应用了残差连接和层标准化。解码器与编码器的结构类似，多列一层的多头注意力子层和位置前馈层，还有一层多头注意力子层用于接收编码器的输出。实际的 Transformer 模型由 6 层编码器和 6 层解码器堆叠而成。具体模型结构设计如下图所示。

多层的神经网络结构能够对句子层级化信息进行建模，Encode 分两个子网络部分，第一个是 Self-Attention，第二个部分是 Feed Forward，大家比较熟悉，就是传统的前馈神经网络，我们摒弃了传统的并行化比较低，而且比较简单且高度并行化的前馈神经网络。Self-Attention 是自注意力机制层，表征句子当中不同位置词之间的关系，是我们前面提到的 it 和 street 或 dog 之间的依赖关系。Decoder 层比 Encoder 层多一个子网络，就是 Encoder-Decoder Attention，它是源端到目标端的注意力机制，对源端词到目标端的助理机制，不是源端到目标端词的依赖关系，用到翻译里是说这个源端待翻译的词和源端生成翻译词之间的依赖关系。我们输入代翻译词是“Thinking Machines ”，先会去查找这两个词分别表示成词向量的形式，再经过 Self-Attention 层得到 Attention 的输出，再经过 Feed Forward 自动生成 Encoder1 的输出，Encoder1 的输出作为 Encoder2 的输入，这样以此类推，6 层一直拼到上面去。

我们设计并实现了 Transfomer 模型从葡萄牙语到英语的翻译模型，并实际测试了翻译效果，并将代码和运行结果上传到 Github 这个开源的代码管理仓库当中 [4]。

[4]https://github.com/awantedboy/data_work

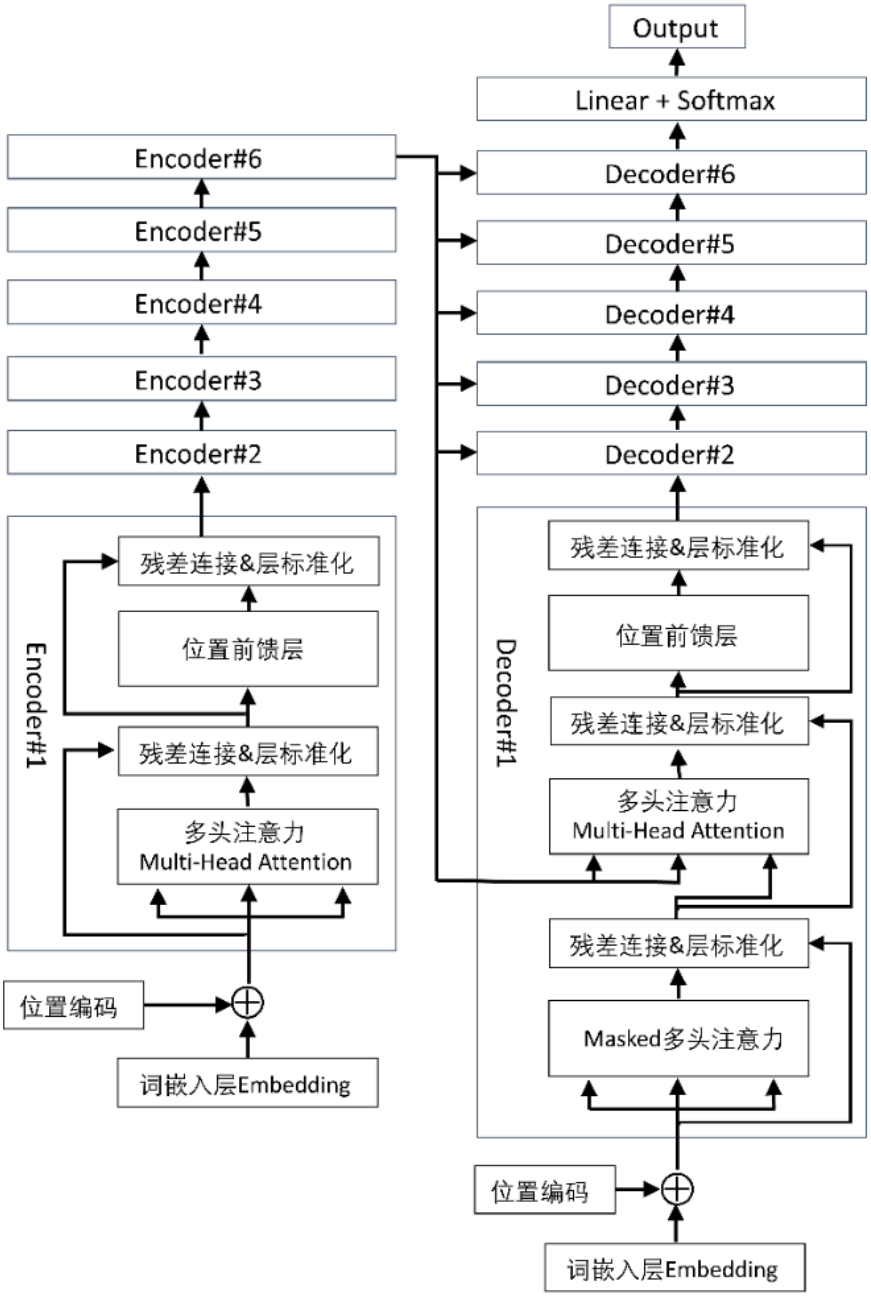


图 3: Transformer 模型结构图 [5]

3 后记

3.1 课程论文构思和撰写过程（本章节作者：杨永康）

近年来，神经网络与注意力机制被广泛应用于自然语言处理领域中，并取得了非常好的效果。本小组在短文中已探究了 RNN 及其两个变种，LSTM 和 GRU 的特性及其应用场景，因此想要在长文中更近一步探究机器翻译方面的研究情况。而注意力机制及 Transformer 模型相比普通的神经网络有着更好的处理效果，因此希望通过探究注意力机制这一手段在机器翻译方面的效果来更好的理解这一算法。

Attention 机制因为其早在上世纪九十年代时，就被应用到了图像处理领域。但直到 2014 年 Google

[5] https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

Mind 团队才在 RNN 模型上使用了 Attention 机制来进行图像分类后 [1]，这一机制才真正火起来。随后，Bahanau 等人的工作首次将 Attention 机制应用到 NLP 领域中 [2]，也就是本文中提到的 Attention 机制应用到机器翻译任务上将翻译和对齐同时进行。此后 Attention 机制被广泛应用于 RNN/CNN 等神经网络模型的各种 NLP 任务中。2017 年来自 Google 的机器翻译团队发表的 Attention is All You Need 中大量使用了自注意力（self-attention）机制来学习文本表示 [3]。自注意力机制也成为了大家近期的研究热点，并在各种 NLP 任务上进行探索。

注意力机制分为软注意力机制（Soft attention）和强注意力（Hard attention），以及被用来做文本处理的 NLP 领域的自注意力机制（Self attention)。本文所讨论的注意力机制即为自注意力机制，其是注意力机制的一种变体，减少了对外部信息的依赖，更擅长捕捉数据或特征的内部相关性。自注意力机制在文本中的应用，主要是通过计算单词间的互相影响，来解决长距离依赖问题。

基于以上一些原因，小组成员经过讨论题后，一致对注意力机制如何应用到机器翻译这一任务产生了较大兴趣，并迅速开展资料查阅，论文撰写等工作。结合分工，团队成员分别了解了机器翻译基本任务，神经网络，注意力机制的基本原理与应用到机器翻译上的现状，并结合部分网络资源进行了模型的构建与代码编写，最终将代码上传到小组成员倪坤宇同学的 github 上。最终撰写过程中，小组成员各司其职，于 12 月 14 日正式成文。

3.2 所参考主要资源（本章节作者：杨永康）

由于课堂上没有注意力机制相关的内容，本小组成员主要参考了来自知乎、CSDN、github 等网络资源，同时通过知网、dblp 等学术论文数据库查阅了相关领域的论文进行阅读。

1). 关于机器翻译的基本任务与研究进展，主要参考了知乎上的一篇博客 [4] 与知网上的相关论文 [5]；

2). 关于神经网络在机器翻译上的应用部分，本文主要参考了一篇 CSDN 博客 [6] 与部分论文 [2][7][8]；

3). 关于注意力机制与 Transformer 模型构建，主要参考了部分论文 [3][9][10] 和来自 tensorflow 官网的一些例子 [11]

3.3 代码撰写的构思与体会（本章节作者：杨永康）

本研究主要代码由倪坤宇同学撰写。由于我们为初次进行 transformer 模型的构建与应用，在撰写代码时我们选择了通用的 jupyter notebook 平台以及基于 Tensorflow 的深度学习框架，便于模型构建与参数调整的可视化。事实上在模型应用的过程中最重要的一步在我们看来是数据的预处理部分，本研究对数据集中要翻译的每一句话分别进行了 padding 填充到最大长度与 mask 处理，防止后续 token 对现在 token 的影响；其次，模型构建后的训练过程中，应注意输入到模型中数据的形状；在模型的参数调整与训练的过程中，我们使用了 Adam 算法，加快了调参过程。同时，在撰写代码时，及时参考 API 官网的说明文档，以及网上的一些例子是非常有必要的。

3.4 人员分工（本章节作者：杨永康）

本文主要分工如下：倪坤宇同学主要负责 Transformer 模型的原理与设计部分，并撰写代码完成了相关实验，即本文第三节；杨永康同学主要负责摘要、第一节概况第一，二，五小节，以及后记与参考文献的整理工作；王博文同学主要负责第一节中机器翻译基本任务，神经网络在机器翻译方面研究现状的撰写，以及第二节实验环境的撰写。

3.5 总结（本章节作者：杨永康）

本文主要讨论了注意力机制的发展现状，并自主设计了一个 transformer 模型应用到机器翻译任务中。在长达半个多月的论文撰写过程中，我们对注意力机制有了初步的了解与认识，并能适当撰写部分代码，又一次开拓了我们的视野并增长了一些能力。虽然三位成员对研究领域并不会直接用到这一算法，但我们仍然认为大论文的撰写过程令我们受益匪浅。

参考文献

[1] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in neural information processing systems*, pages 2204–2212, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[4] Xiecheng. About machine translation, it's enough to read this one. Website, 2019. `https://zhuanlan.zhihu.com/p/66584038`.

[5] 李亚超, 熊德意, and 张民. 神经机器翻译综述. 计算机学报, 41(12):2734–2755, 2018.

[6] littlelyll. Neural Network Machine Translation Summary. Website, 2018. `https://blog.csdn.net/littlely_ll/article/details/79026870`.

[7] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *proceedings of the 52nd annual meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, 2014.

[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[9] Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? a targeted evaluation of neural machine translation architectures. *arXiv preprint arXiv:1808.08946*, 2018.

[10] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[11] Tensorflow Group. Transformer model for language understanding. Website, 2021. `https://www.tensorflow.org/text/tutorials/transformer`.

5

贝叶斯专题

贝叶斯推断是今年着重讲授的专题部分，许是课程讲述靠后的原因，挑选这个部分题目的同学较少。幸有张仪棣，许瑞清，李就良三位同学的一篇与 *GAN* 有关的工作。

– Jingbo Xia

5.1 《GAN 在图像风格迁移研究方向的应用》——张仪棣，许瑞清，李就良

摘要：传统的图像风格迁移受限于画作风格的迁移，随着生成式对抗网络 (GAN) 的兴起与发展，风格迁移不再局限于画风的迁移，而是泛化到任意图像集之间的跨域迁移。凭借强大的数据生成能力，生成对抗网络现如今已被成功应用于图像处理与计算机视觉、自然语言处理、语音识别等领域中。本文对生成对抗网络及其在跨域图像风格转换中的应用展开了深入研究，首先针对图像风格迁移和生成对抗思想进行了概述，简单阐述了 CGAN 的基本原理，紧接着围绕 Pix2Pix、CycleGAN 这两种受 CGAN 模型启发的图像风格迁移模型进行了详细的理论介绍。最后，我们将理论付诸实践，对上述三种模型进行代码复现及改进，并基于 mnist、facades、apple2orange 三种数据集开展实验，成功实现了图像风格迁移任务，根据实验结果验证了生成对抗模式在风格迁移领域中的有效性。代码和预训练模型通过以下链接可进行访问:`https://github.com/xiegangyouth/CodeOfDataMining.git`.

GAN 在图像风格迁移研究方向的应用

张仪棣 [1], 许瑞清 [1], 李就良 [2]

[1]Computer Application Technology, College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

[2]Agricultural Information Engineering, College of Informatics, Huazhong Agricultural University, Wuhan, Hubei Province, P.R. China

摘要

传统的图像风格迁移受限于画作风格的迁移，随着生成式对抗网络 (GAN) 的兴起与发展，风格迁移不再局限于画风的迁移，而是泛化到任意图像集之间的跨域迁移。凭借强大的数据生成能力，生成对抗网络现如今已被成功应用于图像处理与计算机视觉、自然语言处理、语音识别等领域中。本文对生成对抗网络及其在跨域图像风格转换中的应用展开了深入研究，首先针对图像风格迁移和生成对抗思想进行了概述，简单阐述了 CGAN 的基本原理，紧接着围绕 Pix2Pix、CycleGAN 这两种受 CGAN 模型启发的图像风格迁移模型进行了详细的理论介绍。最后，我们将理论付诸实践，对上述三种模型进行代码复现及改进，并基于 mnist、facades、apple2orange 三种数据集开展实验，成功实现了图像风格迁移任务，根据实验结果验证了生成对抗模式在风格迁移领域中的有效性。代码和预训练模型通过以下链接可进行访问：https://github.com/xiegangyouth/CodeOfDataMining.git.

关键词: 图像风格迁移，GAN，CGAN，Pix2Pix，CycleGAN

1 概况

1.1 选题说明（本章节作者：张仪棣）

在信息爆炸增长的大数据时代，我们需要功能强大和通用的工具，以便从数据中发现有价值的信息，进而转化成有组织的知识，应用于不同场景的任务解决，这种需求促使了数据挖掘的诞生。数据挖掘是一个跨学科的计算机科学分支。它是用人工智能、机器学习、统计学和数据库的交叉方法在相对较大型的数据集中发现模式的计算过程。数据挖掘过程的总体目标是从一个数据集中提取信息，并将其转换成可理解的结构，以进一步使用。除了原始分析步骤，它还涉及到数据库和数据管理方面、数据预处理、模型与推断方面考量、兴趣度度量、复杂度的考虑，以及发现结构、可视化及在线更新处。数据挖掘是“数据库知识发现”（Knowledge-Discovery in Database,KDD）的分析步骤，本质上属于机器学习的范畴 [1]

生成对抗网络是非监督学习的一种方法，主要实现原理是让两个不同的神经网络相互博弈，生成网络从潜在的空间中随机抽样作为输入的来源，此时的生成结果是尽可能的接近真实值的，判别网络的输入则一般是真实数据或者生成网络的输出，主要目的是尽可能对结果进行判别真伪。本论文主要从现在最流行的三种生成对抗网络的改进进行描述以及实验，从改良之后的条件生成对抗网络 (CGAN) 到 Pix2Pix 一种基于生成对抗网络的改进, 是基于 CGAN 上的一个图像迁移的鼻祖，再发展到 CycleGAN。在传统的 GAN 中加入一个条件作为生成网络和判别网络的合成条件即构成了 CGAN，在 CGAN 中，生成网络的输入端仅仅只有一个条件而不需要噪声的影响，如果把一张图像作为条件，那么生成的假图像就和我们原始输入的条件图像有一种对应关系，此时就是 Pix2Pix 的原理。基于这个基础之上，去掉成对的训练数据，只需要提供不同域的图像即可完成训练图像之间的映射就是 CycleGAN，具体的说明会在后文进行展开。

1.2 该算法基本原理

1.2.1 图像风格迁移介绍（本章节作者：许瑞清）

图像风格迁移作为计算机视觉领域的热点研究方向，近年来取得了一系列进步，其在图像处理、风格设计及视频摄影处理等方面得到了广泛应用。图像风格迁移的主要任务是将给定风格图像的特定风格迁移到一张内容图像上，即在目标图像上合成风格图像纹理，使得新生成的图像既能保留原始内容图像的具体内容，又能以新的风格形式呈现。

当前实现图像风格迁移的方法主要分为三类：第一类是基于迭代优化的方法，主要解决了固定输入内容图像和目标风格图像的一对一图像风格迁移问题，其主要思想是将输入图像的内容信息和目标图像的艺术风格进行量化表示，从而将风格迁移问题转换为一个迭代优化问题；第二类是基于转换网络的方法，主要解决了将输入的任意图像渲染成指定风格的多对一图像风格迁移问题，其主要思想是训练一个端到端的卷积神经网络来学习、保存内容图像和风格图像的特征信息，使用此转换网络完成对输入图像的风格转换；第三类是基于生成对抗网络的方法，主要解决了多对多的图像跨域风格迁移问题，其主要思想是在成功训练生成对抗网络（Generative Adversarial Network，GAN）的基础上，利用其生成器强大的生成能力来完成对图像的风格转换。

在图像风格迁移技术的发展历程中，Gatys 等人于 2016 年首次提出了基于迭代优化思想的 Nueral Style 深度学习方法，此方法模拟人类视觉的处理方式，训练多层卷积神经网络以从不同层输出的特征图中统计特征信息，实现对图像内容及风格特征的提取和量化，并通过降低内容与风格损失函数完成目标任务 [2]。使用此方法虽然能够实现风格迁移，但会出现内容扭曲的现象，迁移效果不佳且生成速度较慢。Johnson 等人为提高风格迁移速度，提出了基于转换网络思想的 Fast Nueral Style 方法，其使用预先训练好的风格模型，能够在极短的时间内对输入的任意内容图像完成风格的转换 [3]。尽管该方法提高了转换速度，但其在图像生成质量方面没有显著的提高。大量的研究工作表明，基于迭代优化和转换网络思想的风格迁移方法无论如何改进，其生成效果只能停留在简单纹理转变层面，无法凸显出图像特有的风格特点。而基于卷积生成对抗网络的学习方法为图像风格迁移的研究提供了新思路，其对分属两个不同图片集的输入内容图像和目标风格图像之间的复杂关系进行建模，以完成跨域风格迁移任务 [4]。此方法有着极强的风格纹理抓取能力，能够输出与原图像样本风格极为相似的结果，并且在抽象画系风格迁移任务中，生成对抗网络也能通过其独有的对抗训练能力，高效地抓取纹理特点，输出效果理想的图片。在本文中，我们将着重介绍如何应用基于映射函数思想，并受条件式生成对抗网络（Conditional Generative Adversarial Nets，CGAN）启发的 Pix2Pix、CycleGAN 实现对图像的风格迁移。

1.2.2 生成对抗思想（本章节作者：李就良）

生成式模型和判别式模型是目前机器学习领域最为常见的两种模型类型，二者各有优点，且适用于不同的应用场景。生成对抗网络（Generative Adversarial Network）自 2014 年提出以来，作为一种将生成模式和判别模式成功结合的新型学习模型，在深度学习领域一直备受关注。该模型不依赖于任何先验假设来学习高维复杂的数据分布，这一特性也使其在诸多应用领域取得了显著的研究成果。

GAN 的核心思想来源于博弈论，该模型由两个神经网络构成，分别被称为生成器 (Generator，G) 和判别器 (Discriminator，D)。其中生成器用于捕捉真实样本的数据分布，并以随机的噪声作为输入试图生成近似真实数据分布的虚假样本。判别器以真实数据或者生成数据作为输入，对所输入的样本进行评估，并输出该样本是真实样本的概率。GAN 的网络结构不需要构造复杂的损失函数，通过生成器G和判别器D的博弈过程达到全局最优 [5]。生成器的训练目标是为了使得判别器错判的概率最大化，判别器的训练目标是为了提高自身对真假样本的辨别能力。二者在训练过程中相互竞争、相互激励，促使着生成器和判别器双方不断优化各自方法和逼近各自的训练目标，最终使得生成数据无限逼近真实数据。GAN 的优化过程实质

上是一个极小极大博弈 (Minimax game) 问题，其具体目标函数如公式 (1) 所示 [6]：

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z))] \tag{1}$$

其中，x 表示输入判别器 D 的真实样本，z 表示输入生成器 G 的随机噪声，G(z) 表示由生成器输出的生成数据，$p_{data}(x)$ 为真实数据分布，$p_z(z)$ 为生成数据分布。当达到全局最优解时，判别器的输出结果始终保持为 1/2，表明生成数据分布 $p_z(z)$ 已无限逼近真实数据分布 $p_{data}(x)$，证明过程如下所示。

固定 G，求解 $D_G^* = arg\max_D V(D,G)$:

$$\begin{aligned}
\frac{\partial\left(\max_D V(D,G)\right)}{\partial(D)} &= \frac{\partial\left(\mathrm{E}_{x\sim p_{\text{data}}\ (x)}[\log D(x)] + \mathrm{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]\right)}{\partial(D)} \\
&= \frac{\partial\left(\mathrm{E}_{x\sim p_{\text{data}}\ (x)}[\log D(x)] + \mathrm{E}_{x\sim p_g(x)}[\log(1-D(x))]\right)}{\partial(D)} \\
&= \frac{\partial\left(\int\left[p_{\text{data}}\ \times \log(D)\right]dx + \int\left[p_g \times \log(1-D)\right]dx\right)}{\partial(D)} \\
&= \frac{\partial\left(\int\left\{\left[p_{\text{data}}\ \times \log(D)\right] + \left[p_g \times \log(1-D)\right]\right\}dx\right)}{\partial(D)} \\
&= \frac{\partial\left(\int\left\{\left[p_{\text{data}}\ \times \log(D)\right] + \left[p_g \times \log(1-D)\right]\right\}dx\right)}{\partial(D)} \\
&= \int \frac{\partial\left[p_{\text{data}}\ \times \log(D) + p_g \times \log(1-D)\right]}{\partial D}dx \\
&= \int\left[p_{\text{data}}\ \times \frac{1}{D} + p_g \times \frac{-1}{1-D}\right]dx = 0 \Rightarrow D_G^* = \frac{p_{\text{data}}}{p_{\text{data}}\ + p_g}
\end{aligned} \tag{2}$$

代入 D_G^*，求解 $\min_G V(D_G^*, G)$，其中 D_{KL} 和 D_{JS} 分别指 KL 散度及 JS 散度：

$$\begin{aligned}
\min_G \max_D V(D,G) &= \mathrm{E}_{x\sim p_{\text{data}}\ (x)}[\log D(x)] + \mathrm{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \\
&= \min_G E_{x\sim p_{\text{data}}\ (x)} \log\frac{p_{\text{data}}}{p_{\text{data}}\ + p_g} + E_{x\sim p_g(x)} \log\frac{p_g}{p_{\text{data}}\ + p_g} \\
&= \min_G E_{x\sim p_{\text{data}}\ (x)} \log\frac{p_{\text{data}}\ /2}{\left(p_{\text{data}}\ + p_g\right)/2} + E_{x\sim p_g(x)} \log\frac{p_g/2}{\left(p_{\text{data}}\ + p_g\right)/2} \\
&= \min_G \int p_{\text{data}}\ \cdot \log\frac{p_{\text{data}}\ /2}{\left(p_{\text{data}}\ + p_g\right)/2}dx + \int p_q \cdot \log\frac{p_q/2}{\left(p_{\text{data}}\ + p_g\right)/2}dx \\
&= \min_G D_{KL}\left(p_{\text{data}}\ \|\frac{p_{\text{data}}\ + p_q}{2}\right) + D_{KL}\left(p_q\|\frac{p_{\text{data}}\ + p_q}{2}\right) - \log 4 \\
&= \min_G \left\{2 \times D_{JS}\left(p_{\text{data}}\ \|p_q\right) - \log 4\right\} \geq -\log 4 \Rightarrow p_g^* = p_{\text{data}}
\end{aligned} \tag{3}$$

生成对抗网络 (GAN) 作为目前最为流行的生成式模型之一，由于生成对抗的训练模式具备学习视觉语义特征的优势，在超分辨率重构、图像分割、图像修复等应用领域都取得了巨大的成功 [7]。基于 GAN 的生成对抗思想，近几年涌现了许多优秀的变种模型，如 CGAN、Pix2pix、CycleGAN 等。实验结果表明，这些基于 GAN 的变种模型可以在风格迁移类型的迁移学习任务中生成无限逼近真实数据的图像 [8]，本文后续章节将针对这些变种模型在图像风格迁移任务中的应用展开具体介绍与探讨。

2 基于生成对抗网络的图像风格迁移

2.1 基于 CGAN 的图像生成（本章节作者：许瑞清）

2.1.1 CGAN

在图像处理领域，可以通过设计传统的神经网络模型，完成在给定标签的情况下，生成相应图像的任务。然而，这种方法的实际效果往往并不理想，问题在于所给标签数据集可能存在一对多的情况。以给定文本来生成相应图像的任务为例，依据文本对应的图像可能会有多个，其标签虽然相同，但内容却相差甚远，也因此训练得到的神经网络模型会让该标签的输出结果和每一个训练结果都尽量接近，这就造成生成图像非常模糊甚至无法分辨的情况发生。

生成对抗网络是目前深度学习领域中发展最为快速的一个分支，它可用于诸如图像生成、编辑和着色、风格转换、物体变形、照片增强等多个领域。与其他图像生成模型相比，GAN 无需指定分布，其通过对抗的方式学习到符合真实数据特征的分布，完成模型的训练，以确保生成更为真实的图像。然而，传统 GAN 的训练非常自由，其不具备在给定标签或条件下生成相应真实图片的功能。例如，我们输入 “0” 标签，希望得到一张显示数字 0 的图片，但传统 GAN 只着重于分辨图片是生成器生成的还是真实原图，不能确保输入标签 “0” 就会生成数字为 0 的图片。具体来讲，如果输入标签 “0”，却生成了一张形态显示真实的数字 “1” 图片，这样的生成器也会被认为是一个训练良好的生成器，但事实上这并不是我们想要的效果。

为解决带标签图像的生成问题，研究者们在 GAN 的基础上提出了条件式生成对抗网络（Conditional Generative Adversarial Nets，CGAN）这一概念。在 CGAN 中，生成器并不是单单依靠一个未知的噪声分布学习，在其学习训练过程中还辅以特定条件或某些特征信息（例如一张图片的标签或其他更细节的特征），CGAN 的目标函数如公式 (4) 所示。

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x|y)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y))] \tag{4}$$

与传统 GAN 的目标函数相比，其整体没有什么变化，只是在生成器和判别器的输入阶段增加了额外信息 y（y 可以是任意信息，如模态数据或类别信息等）作为条件。公式 (4) 中的生成器和判别器的概率均以条件概率形式定义，由此我们可以实现通过改变条件 y 来控制生成效果的目标。CGAN 的网络结构如图 1(a) 所示。

CGAN 的优化目标为：给定条件 y，在判别器达到最大化真实数据与生成数据差异判别标准（即判别器能够很好地判别数据来源）的情况下，通过最小化真实数据与生成数据之间的差异来训练模型。生成器的输入是随机噪声 z 与条件 y 的拼接组合，判别器的输入是数据 x 和条件 y 的拼接组合。在本实验中，条件 y 是 MNIST 手写数字数据集的数字标签。除此之外，还需要注意的是在训练过程中条件 y 不但要在输入时与噪声 z 融合在一起，其在生成器和判别器的每一层输入里都要与特征图相融合，这样才能让模型 “学好条件 y”，让标签更好地发挥作用。

2.1.2 实验数据集

此部分实验旨在训练 CGAN，以实现在给定标签的情况下生成相应图片的目标。通过实践，充分理解 CGAN 的网络架构，为受 CGAN 启发而设计得到的应用于图像风格迁移的 Pix2Pix 和 CycleGAN 模型提供知识支撑。在对 CGAN 进行训练时，我们选择带有 0-9 数字标签的 MNIST 手写数字数据集。此数据集是机器学习领域中非常经典的一个数据集，由 60000 个训练样本和 10000 个测试样本组成，每个样本都是一张 28×28 像素的灰度手写数字图片。图片中每个像素点值的范围在 0-255 之间，其表示像素点的明暗度。若数值越小，代表像素点越暗，反之越亮。

2.1.3 实验设置

此部分实验在 Intel i5 8250U 1.6GHz 4 核 8 线程处理器上进行，使用 Windows 操作系统，以 Pycharm 作为开发环境，深度学习框架采用基于 CPU 版本的 Tensorflow 2.4.1。CGAN 中生成器网络及判别器网络均使用 Keras API 中的 Sequential 模型进行定义。其中，生成器网络的输入是一个带标签（0-9）的随机数，具体操作方式是生成一个 N 维的正态分布随机数，再利用 Embedding 层将条件信息（即标签信息：0-9）转换为 N 维的稠密向量，之后将其与 N 维的正态分布随机数相乘以完成输入拼接，损失函数的定义需考虑两方面：其一为生成图片是否被判别器判别为真，其二是生成图片是否被分类到正确的标签下。判别器网络的输入为真实/生成图片及其标签经过 Embedding 操作后的拼接结果，此网络不仅要判别输入图片的真假，还要判别输入图片与真实标签是否相符合，因此损失函数的定义也要考虑两方面：其一为输入图片的真伪判别结果与真实情况的对比，其二为图片所属标签的判断结果。

2.2 基于 Pix2pix 的图像风格迁移（本章节作者：张仪棣）

2.2.1 Pix2pix

在 CGAN 被提出之后，Phillip Isola 等人在对条件生成网络提出了一种在图像方面的应用，图像处理、计算机图形学和计算机视觉中的许多问题都可以看作是将输入图像“转换”成相应的输出图像。正如概念可以用英语或法语表达一样，场景可以呈现为 RGB 图像、梯度场、图形映射、语义标签映射等 [9]。

Pix2pix 的算法示意图如图 1(b) 所示，首先我们将输入的图像（例如此时为本人画的一只简笔猫）来表示，猫的边缘是我们将输入的部分，暂时记作 X，在此过程中我们暂时不对噪声进行描述，因为在 Pix2pix 中噪声是可以忽略不计的，然后通过生成器 G 进行合成之后得到生成的图像 G（x）作为判别器 D 的输入，在判别器端通过概率的计算来确定是否是一对真实的图像，如果得到的概率越大那么表示越接近真实的图片，因此生成器 G 的目标就是在 G（x）和 X 作为输入源时能够使得 D 的判别越准确，这样即成功的表示我们欺骗了生成器 D。

Pix2pix 在 CGAN 篇的公式 2 去掉 x 即为了测试鉴别器对条件调整的重要性，与鉴别器未观测到的无条件变量进行比，并且在先前的研究当中将 GAN 目标与更传统的损耗 (如 L2 距) 混合是有益的。鉴别器的工作保持不变，但发生器的任务不仅是欺骗鉴别器，而且还要接近 L2 意义上的真实输出。我们还将探讨此选项，使用 1 范数而不是 2 范数，因为 1 范数可以减少模糊程度，作者在论文中做出两种 loss 的明显对比，但是由于篇幅限制原因暂不做展示，如下公式所述：

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1 \tag{5}$$

本方法的核心是求出以下公式

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G) \tag{6}$$

GAN 的优化过程中，优化器判别的部分是使公式 4 越大越好，生成器的目的则是让误差越小越好，即公式中取对数的部分越小越好，但是在 GAN 本身的文章当中提到训练的时候容易出现判别器过于强大但是生成器过于弱小导致生成器训练不起来的情况，因此需要对 D 进行最大化处理，即不用 log(1-D(x,G(x,z))) 转而使用最大化的 log(D(x,G(x,z)))，即作者在这里采用的是原始 GAN 的训练网络方法，先对 D 做一次梯度下降，在对 G 做一次梯度下降，在生成器中采用的是 U-Net 设计，这是在图像分割领域内应用非常广泛的一个网络结构，也是在原本的 GAN 采用的 encoder-deconder 上对解码器上进行编码器的越级连接。判别器采用的是 PatchGAN，它对每一个输入的图像部分都可以给出一个预测值，可以判别输入的每个区域是真还是假。对于低频的参数 1 范数损失表现还是不错的，但是对于高频信息 PatchGAN 它只判断一个

N*N 的 patch 是真是假，所以能更好的判别高频信息。即使 N 远小于原图的大小，patchGAN 依然可以产生较好的结果。并且，它具有更少的参数，运行更快，并且可以运用于任意大的图像。

2.2.2 实验数据集

本论文可以使用原始作者已经提供一些数据集例如 Facades、Citycapes、maps、UT 的鞋类数据等，此外也可以使用自己定义的数据集来进行生成。评价生成图形的质量是一个很麻烦的问题，传统的评价方法是对于每一个像素点进行 MS error，但是在本文中主要是选取两个方面来进行判别：首先是对于图像上色和照片生成在生成的过程中展示给观众 1s 的时间来让被试者判断是否合理，另外就是用成型的系统识别来进行目标的识别。

2.2.3 实验设置

本实验是在 Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz 处理器上进行，使用 centos 操作系统，以 Anconda 作为开发环境，深度学习框架采用基于 CPU 版本的 Tensorflow 版本为 1.4.1。首先将 Facades 中的模型图作为输入部分，然后使用 Tensorflow 来进行处理。由于时间限制，于是最大 epochs 我们暂时设定为 10，步长为 50，方向是从 B 到 A，即从模型当中学习到仿真图片。

2.3 基于 CycleGAN 的图像风格迁移（本章节作者：李就良）

2.3.1 CycleGAN

基于原始 GAN 搭建的网络模型虽然可以实现图像源域到目标域的映射，但是受限于其迁移方向的单一性，导致该映射高度不受约束，即单独的对抗训练无法有效保证不同的输入样本 x 被映射为不同的生成样本 y。而针对 Pix2pix，虽然也能成功实现风格迁移任务，但是 Pix2pix 对数据集要求过于苛刻——要求训练数据集一一配对，而这种一一配对的数据在现实场景下是极难获得的。

针对上述图像风格化存在问题，循环一致性对抗网络（Cycle-Consistent Adversarial Networks，CycleGAN）的提出成功解决了上述缺陷。CycleGAN 通过将两个单向 GAN 进行镜像连接构成环形 GAN，不仅有效避免了 X 中所有图像映射为 Y [9–11] 中同一图像的可能性，还解决了在缺乏配对训练数据集的情况下依旧可以实现不同数据集的跨域迁移的瓶颈 [9,12]。由于其采用了双向循环生成的结构，CycleGAN 也因此得名。

CycleGAN 将两个单向 GAN 进行镜像连接，实现了两种图像集的双向映射，其网络结构如图 1 所示。其中 G、F 均为生成器，D_X、D_Y 均为判别器，X、Y 则对应不同的图像集。

首先，针对单向 GAN，设 X→Y 表示图像样本 x∈X 通过生成器 G 被映射为近似于图像集 Y 的生成样本 G(x)。随后将生成样本输入判别器数学公式: D_Y 进行真假数据的鉴别。多轮对抗训练将会使得图片从源域 X 无限逼近于目标域 Y。生成器 G 和判别器 D_Y 的损失函数如下：

$$\begin{aligned} L_{GAN}\left(G, D_Y, X, Y\right) &= \min_{G} \max_{D_Y} V\left(D_Y, G\right) \\ &= E_{y \sim p_{\text{data}}(y)}\left[\log D_Y(y)\right] + E_{x \sim p_{\text{data}}(x)}\left[\log\left(1 - D_Y(G(x))\right)\right] \end{aligned} \tag{7}$$

CycleGAN 还引入了映射 Y→X，训练原理与 X→Y 类似，可以得出生成器 F 和判别器 D_X 的损失函数如下：

$$\begin{aligned} L_{GAN}\left(F, D_X, Y, X\right) &= \min_{F} \max_{D_X} V\left(D_X, F\right) \\ &= E_{x \sim p_{\text{data}}(x)}\left[\log D_X(x)\right] + E_{y \sim p_{\text{data}}(y)}\left[\log\left(1 - D_X(F(y))\right)\right] \end{aligned} \tag{8}$$

为了避免 X 中所有图像映射为 Y 中同一图像的可能性，CycleGAN 引入了循环一致性损失的概念，即让模型同时学习 X→Y 和 Y→X 两种映射。在图像样本 x∈X 经 G 映射生成的近似样本 G(x)∈Y 后，利用生成器 F 将其重构再度转换回原始数据分布，即 x → G(x) → F(G(x)) ≈ x，反之亦然。将基于原始图片产生的生成图片进行再度反转而生成的样本称为重构样本，通过计算重构样本和原始图像样本之间的距离，构建起循环一致性损失的数学表达式如公式 (9) 所示：

$$L_{cyc}(G,F) = E_{x\sim p_{data}(x)}[||F(G(x)) - x||_1] + E_{y\sim p_{data}(y)}[||G(F(y)) - y||_1] \tag{9}$$

综合公式 (7) 、公式 (8) 及公式 (9) ，得出 CycleGAN 的目标函数如公式 (10) 所示，其中 λ 起着权衡对抗损失和循环一致性损失相对重要性的作用。

$$L\left(G,F,D_X,D_Y\right) = L_{GAN}\left(G,D_Y,X,Y\right) + L_{GAN}\left(F,D_X,Y,X\right) + \lambda L_{cyc}(G,F) \tag{10}$$

综上所述，图像风格迁移任务成功被转换为对以下函数的数学求解：

$$G_*, F_* = arg\min_{G,F}\max_{D_X,D_Y} L(G,F,D_X,D_Y) \tag{11}$$

2.3.2 实验数据集

传统的图像风格迁移受限于配对训练数据集，而这种一对一配对的数据在现实场景下不易获得，而 CycleGAN 突破了这一限制，在没有成对训练数据集的前提下成功实现了源域到目标域的迁移任务。本文使用的数据集来自 berkeley 提供的 apple2orange 数据集，图像尺寸均为 256x256，其中包含 1261 个苹果的照片和 1267 个橙子的照片。

2.3.3 实验设置

卷积神经网络 (CNN) 在深度学习领域受到了广泛应用，它在图像检测、图像分割中均表现良好。本文实验先后使用 ResNet（Residual Neural Network）[13] 和 DenseNet（Dense Convolutional Network）[14] 分别作为生成器的骨干网络，在 Linux 操作系统下，采用 PyTorch 开源深度框架作为实验环境。传统 CycleGAN 网络的生成器采用 ResNet，该生成器是全卷积连接类型，由编码器、转换器和解码器组成。而 DenseNet 则是基于 ResNet 的优化模型，在 ResNet 的基础上网络浅层和深层之间联系得更加稠密，提炼特征得能力也强于 ResNet。在训练过程中，采用均方误差 (MSE) 和 1 范式 (1NF) 分别作为衡量对抗损失和循环一致性损失的评价指标。

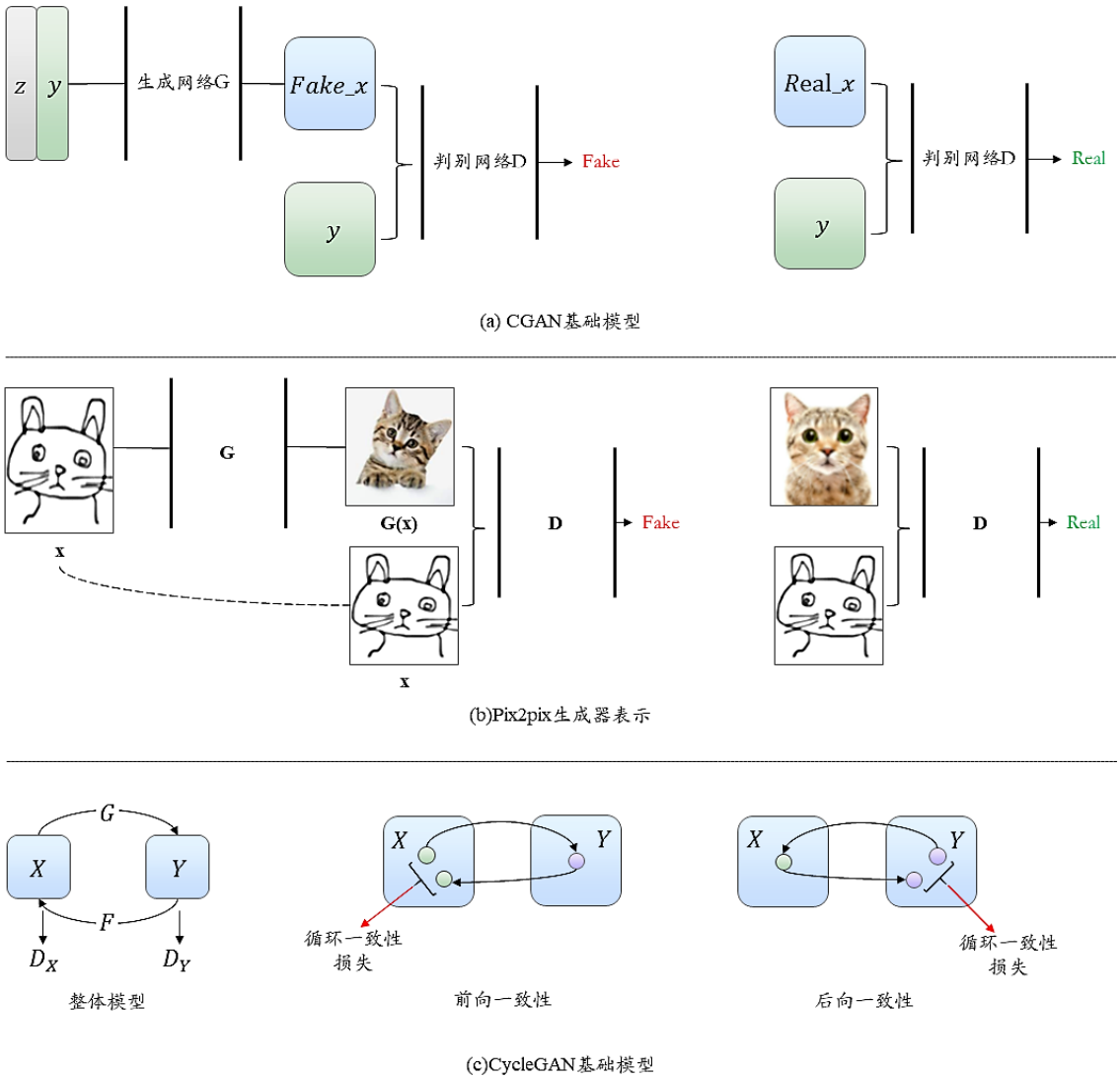


图 1: CGAN、Pix2pix、CycleGAN 的基本模型图。图片来源 [自主绘制]

2.4 实验结果与分析

2.4.1 基于 CGAN 的图像生成（本章节作者：许瑞清）

此部分实验的网络设计均使用全连接神经网络层及二进制交叉熵损失函数，我们使用 LeakyReLU 激活函数及 Dropout 函数（防止模型过拟合，提高模型泛化能力）优先训练判别器，当判别器训练到能够较好区分真假图片时固定其参数，再应用 LeakyReLU 激活函数及 BatchNormalization 操作（批标准化操作，提高模型训练的稳定性）训练生成器。我们设置训练次数为 20000 次，每次迭代输入的 batch size 大小为 32，每 200 轮保存一次图像生成结果，实验结果如图 2(a) 所示。

从图中可以观察到在训练 800 轮后，模型已经开始生成显示数字的图片，但图片信息非常模糊；在 6800 轮后，已经能够生成比较清晰的数字，但是标签条件对数字生成的控制还不好；13400 轮后，模型已经能够按照标签信息生成对应的正确图片；20000 轮后，模型不仅能够生成指定标签下正确的图像，其效果呈现也变得更好。实验结果充分证明了 CGAN 作为一种有监督的机器学习方法在图像生成方面有着优越性表现。

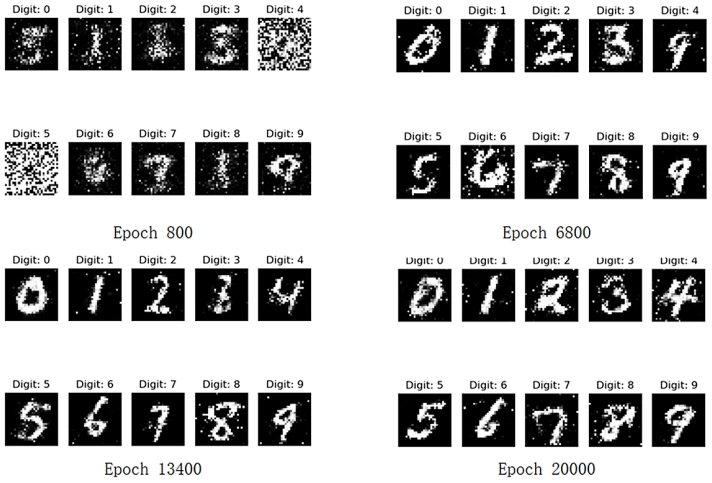
2.4.2 基于 Pix2pix 的图像风格迁移 (本章节作者：张仪棣)

PatchGAN，它只判断一个 N*N 的 patch 是真是假，所以能更好的判别高频信息。即使 N 远小于原图的大小，patchGAN 依然可以产生较好的结果。并且，它具有更少的参数，运行更快，还可以运用于任意大的图像。当 patchsize 为 1×1 时，则为 pixelGAN, 为 70×70 是 PatchGAN，286×286 是 ImageGAN，在本次实验中将最大步长调整为 50 和 200，做了 10 个 epochs，并且对 100 张图像进行测试之后的结果，如图 2(b) 所示。

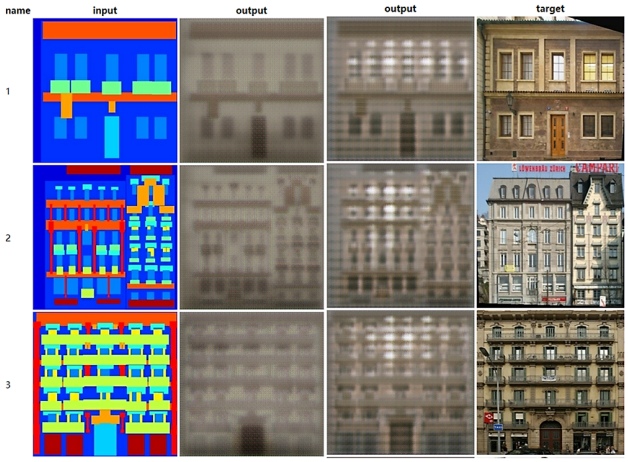
可以看到将模型抽象成具体建筑的时候，虽然训练的次数不多但是已经可以大概抽样成目标了，并且运行时间也还是比较快速的，并且在增加迭代次数之后越来越接近真实图像，这也证明了 Pix2pix 在图像迁移领域上的初步应用是具有一定的成效的。

2.4.3 基于 CycleGAN 的图像风格迁移 (本章节作者：李就良)

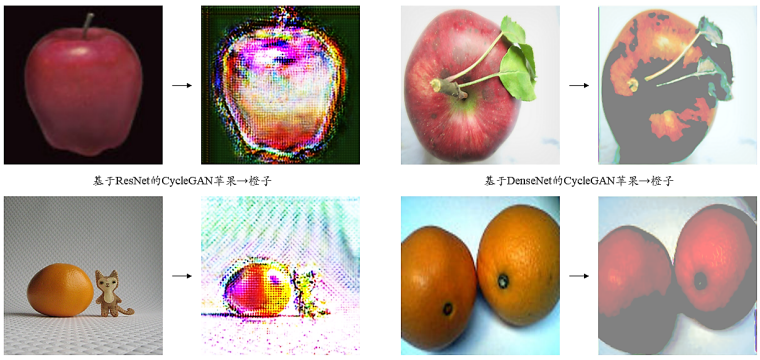
受限于显卡配置和训练轮次较少，采用 ResNet 版原始 CycleGAN(生成器以 ResNet 作为骨干网) 得到的训练结果表现不是十分理想。据此，本文将生成器的骨干网络由 ResNet 替换成 DenseNet 在训练相同轮次前提下进行再次尝试，由于 DenseNet 为 ResNet 的优化模型，其稠密连接结构使得训练效果优于 ResNet，证实了 CycleGAN 在图像风格迁移任务中的有效性以及 DenseNet 的优越性。受限于论文插图数量，论文仅展示部分基于 ResNet 和 DenseNet 的 CycleNet 得到的迁移效果图，如图 2(c) 所示。可以发现，在训练设备稍逊及训练轮次较低的环境之下，CycleGAN 已经初步具备风格迁移的能力（效果图中已经实现了两种水果的颜色迁移），证明了 CycleGAN 对于图像风格迁移任务具有不俗的表现效果。



(a)CGAN实验效果图



(b)Pix2pix实验效果图



(c)CycleGAN实验效果图

图 2: 实验效果图。图片来源 [自主绘制]

3 后记

3.1 课程论文构思和撰写过程（本章节作者：许瑞清）

初期,我们通过查阅文献及其他相关资料了解到了生成对抗网络(GAN)的提出背景。它是由 Goodfellow 等人于 2014 年，在计算复杂且生成效果有限的传统生成模型的基础上，提出的一种基于博弈论的新型生成

模型。该模型是一种用于半监督和无监督学习的新兴技术，一经提出便引起了研究者的广泛关注，它不仅能为深度学习中的数据增强、数据预处理方法生成样本，而且还在计算机视觉领域中有着广泛的应用场景。

近年来，图像风格迁移的相关研究得到了飞速发展，大量的研究工作表明传统风格迁移方法的生成效果不尽如人意。生成对抗网络在图像风格迁移任务中的应用打破了画作风格迁移的局限性，其将图像风格的转换泛化到任意图像集之间的跨域风格迁移，使得生成图片在较好保留原图片内容信息的同时能更加真实地以目标风格进行呈现。

本文首先对图像风格迁移和生成对抗思想进行了较为通俗详细的介绍，为帮助我们设计完成基于 GAN 的风格迁移模型提供相关的知识背景。如何准确地对两个图片集之间的复杂关系进行建模是基于 GAN 的跨域图像风格迁移任务的关键，主要有两种思路：其一是利用隐变量编码构建图片集之间的映射关系；其二是学习能够完成图片集之间相互转换的映射函数 [4]。

通过阅读相关文献，我们了解到 Isola 等人于 2016 年提出了一种基于 CGAN 的有监督图像风格转换的生成对抗网络 Pix2Pix [9]，该模型中的生成器是一个端到端的图像转换网络。具体来讲，Pix2Pix 将真实图像作为额外信息输入到生成器中，为了使生成器产生更逼真的结果，在原有对抗损失函数的基础上，利用 L_1 范数惩罚生成器所生成样本与真实图像之间的差异。为了解决 Pix2Pix 算法需要给生成器引入额外信息的问题，Zhu 等人于 2017 年提出了一种无监督图像风格转换的生成对抗网络 CycleGAN [15]，其给出了循环一致性假设，即源域中的一幅图像经过源域到目标域的映射以及目标域到源域的映射后得到原始图像的重建图像，选取 L_1 范数作为循环损失一致函数，用于衡量重建图像与原始图像的差异程度，并作为生成器目标函数的一部分。

Pix2Pix 及 CycleGAN 这两种图像风格迁移模型均基于映射函数思想，受 CGAN 启发设计得到的。在文中，我们首先对 CGAN 的基本原理进行了详细介绍，并通过设计训练 CGAN 网络来进一步学习如何利用额外信息来达到理想的图像生成效果，为 Pix2Pix 模型的设计提供知识支撑。本文着眼于实现基于 Pix2Pix 和 CycleGAN 模型的图像风格迁移任务，在对这两种模型进行详细的理论介绍后，我们进行了代码复现和改进，并基于 facades、apple2orange 这两类数据集开展实验。最后，我们对实验结果进行了简单的分析，验证了生成对抗网络模型在图像风格迁移任务中的有效性。

本文的意图在于记录我们了解学习及实现生成对抗网络模型在图像风格迁移任务中的应用，为帮助我们在之后相关科研任务中应用 GAN 模型打下良好的理论知识与实践基础。

3.2 所参考主要资源（本章节作者：张仪棣）

算法的参考来源于 github 上一些开源项目，具体会在参考文献中指出，主要的参考资源还有来源于 CVPR 上一些文章，以及从计算机学报、一些从 Google Scholar 上检索到的综述类文章、从知网上查询到的一些博士论文等资源 [16–19]。

3.3 人员分工（本章节作者：李就良）

在小组集体构思之后，我们决定基于 CGAN、Pix2pix、CycleGAN 三种 GAN 的变种模型对 GAN 在图像风格迁移任务中的应用进行介绍与探讨。我们初步设想从模型的理论介绍、实验的开展进行两部分进行操作。为实现工作量的均衡分配，小组内成员各自负责了一种模型的实验实操和内容撰写工作，具体分工如下图所示（不同颜色代表每位成员的负责内容，其中标红章节为三人合力书写）。在内容初步撰写完毕之后，我们各自对其他成员的撰文进行了检阅，数易文稿，多次修正，最终完成了本课程论文的撰写。

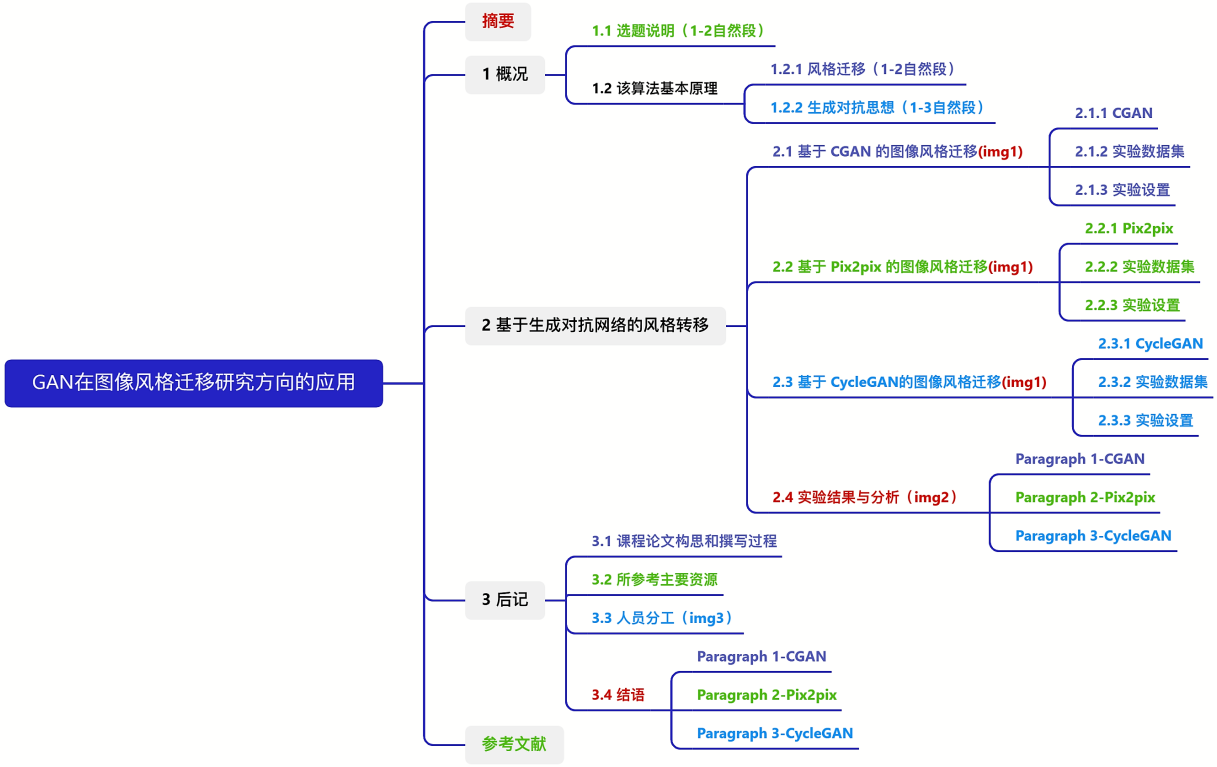


图 3: 成员撰写任务分工图。图片来源 [自主绘制]

3.4 结语（本章节作者：许瑞清、张仪棣、李就良）

在图像处理方面，传统的生成对抗网络（GAN）模型只能拟合原数据集的像素概率分布，其生成的样本并没有提供新的信息以优化模型的分类边界。而条件式生成对抗网络（CGAN）模型的提出能够在给定条件的情况下完成对应的图像生成，简单来讲，CGAN 实际上就是在传统 GAN 的基础上添加了一个“按钮”，让生成对抗网络模型按照我们的需求给出我们真正想要的结果。在 CGAN 的生成器及判别器的构造中，都需要将在传统 GAN 中的输入与给定的条件结合起来，具体方法为利用 Embedding 层进行输入拼接。这一改进为某些特定工程及应用提供了较强的方法性，比如它启发了一系列用于图像风格迁移的 GAN 模型——Pix2Pix、CycleGAN 等。(by 许瑞清)

Pix2pix 是我接触到的第一个有关迁移学习的项目，在本次结课作业期间，通过上课中学习到的知识，比如在一范数和二范数的选取会带来不同的模糊，以及具体公式中的一些推导上能够比以前更加快速，具体到本项目而言，首先尝试过 pytorch、tensorflow、keras 等不同的平台去运行该项目，也尝试自己生成数据集，虽然最终的效果并不是十分理想，于是还是选取论文本身的数据集 facde 来进行处理，本次作业给我的收获在于理论上的公式推导以及具体的如何运行一个项目上。(by 张仪棣)

通过对 CycleGAN 原理以及其在图像风格迁移领域中的应用进行了介绍与探讨，并针对 apple2orange 数据集进行了原始网络代码复现 (基于 ResNet) 和改进 (基于 DenseNet)。同时对两种版本 CycleGAN 的迁移效果进行了对比。结果证明 CycleGAN 相较于传统的图像风格迁移算法，具有更好的表现效果。总体而言，在人工智能的浪潮下，基于生成对抗思想的图像风格迁移是一个富有挑战性的新兴领域，不论实在学术指导还是工业实践中都具有非常重要的研究意义和应用前景。于本人而言，通过本次结课作业也令我对生成对抗网络思想的掌握程度更加深刻（尤其在公式推导方面），也更加明白团队协作能力的重要性。(by 李就良)

参考文献

[1] Website. https://zh.wikipedia.org/wiki/Data_mining.

[2] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[3] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[4] 董伟. 生成对抗网络研究及其在图像风格转换的应用. Master's thesis, 宁波大学, 2018.

[5] 林懿伦, 戴星原, 李力, 王晓, and 王飞跃. 人工智能研究的新前线: 生成式对抗网络. *自动化学报*, 44(5):775–792, 2018.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] 生成对抗网络理论模型和应用综述. 金华职业技术学院学报, 17(3), 2017.

[8] 许哲豪 and 陈玮. 基于生成对抗网络的图片风格迁移. *软件导刊*, 17(6):207–209, 2018.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[10] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[15] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[16] Website. `https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix.git`.

[17] Website. `https://github.com/liuzhuang13/DenseNet.git`.

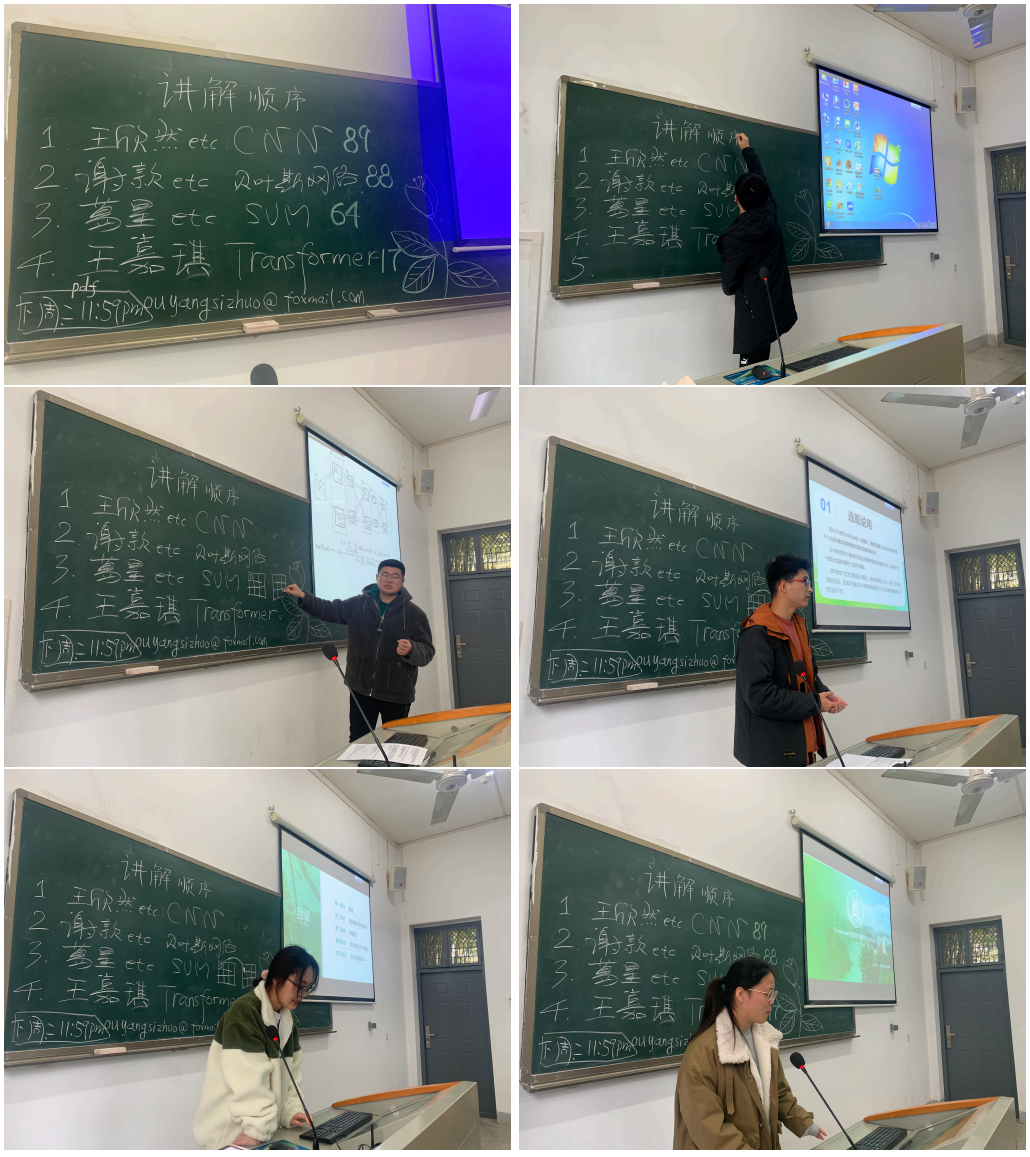
[18] Website. `https://github.com/phillipi/pix2pix`.

[19] Website. `https://blog.csdn.net/weixin_44791964/article/details/103744620?spm=1001.2014.3001.5501`.

6

Acknowledgement

6.1 课程掠影



6.2 ——完——

Thank everyone who contributes to the formation of this material.
谢谢所有帮助这个教学材料成稿的人。