机器学习与深度学习核心知识点总结--写在校园招聘即将开始时

原创：AI学习与实践平台 SigAI 8月8日

导言

一年一度的校园招聘就要开始了，为了帮助同学们更好的准备面试，SIGAI 在今天的公众号文章中对机器学习、深度学习的核心知识点进行了总结。希望我们的文章能够帮助你顺利的通过技术面试，如果你对这些问题有什么疑问，可以关注我们的公众号，向公众号发消息，我们将会无偿为你解答。对于不想在近期内找工作的同学，阅读这篇文章，对加深和巩固机器学习和深度学习的知识也是非常有用的。

数学

1.列举常用的最优化方法

梯度下降法
牛顿法，
拟牛顿法
坐标下降法
梯度下降法的改进型如AdaDelta，AdaGrad，Adam，NAG等。

2.梯度下降法的关键点

梯度下降法沿着梯度的反方向进行搜索，利用了函数的一阶导数信息。梯度下降法的迭代公式为：

$$\mathrm{x}_{k+1} = \mathrm{x}_k - \gamma \nabla f\left(\mathrm{x}_k\right)$$

根据函数的一阶泰勒展开，在负梯度方向，函数值是下降的。只要学习率设置的足够小，并且没有到达梯度为0的点处，每次迭代时函数值一定会下降。需要设置学习率为一个非常小的正数的原因是要保证迭代之后的x_{k+1}位于迭代之前的值x_k的邻域内，从而可以忽略泰勒展开中的高次项，保证迭代时函数值下降。

梯度下降法只能保证找到梯度为0的点，不能保证找到极小值点。迭代终止的判定依据是梯度值充分接近于0，或者达到最大指定迭代次数。

梯度下降法在机器学习中应用广泛，尤其是在深度学习中。AdaDelta，AdaGrad，Adam，NAG等改进的梯度下降法都是用梯度构造更新项，区别在于更新项的构造方式不同。对梯度下降法更全面的介绍可以阅读SIGAI之前的公众号文章“理解梯度下降法”。

3.牛顿法的关键点

牛顿法利用了函数的一阶和二阶导数信息，直接寻找梯度为0的点。牛顿法的迭代公式为：

其中H为Hessian矩阵，g为梯度向量。牛顿法不能保证每次迭代时函数值下降，也不能保证收敛到极小值点。在实现时，也需要设置学习率，原因和梯度下降法相同，是为了能够忽略泰勒展开中的高阶项。学习率的设置通常采用直线搜索（line search）技术。

在实现时，一般不直接求Hessian矩阵的逆矩阵，而是求解下面的线性方程组：

其解d称为牛顿方向。迭代终止的判定依据是梯度值充分接近于0，或者达到最大指定迭代次数。

牛顿法比梯度下降法有更快的收敛速度，但每次迭代时需要计算Hessian矩阵，并求解一个线性方程组，运算量大。另外，如果Hessian矩阵不可逆，则这种方法失效。对牛顿法更全面的介绍可以阅读SIGAI之前的公众号文章“理解牛顿法”。

4.拉格朗日乘数法

拉格朗日乘数法是一个理论结果，用于求解带有等式约束的函数极值。对于如下问题：

$$\min f\left(\mathrm{x}\right)$$
$$h_i\left(\mathrm{x}\right) = 0, i = 1, ..., p$$

构造拉格朗日乘子函数：

$$L\left(\mathrm{x}, \lambda\right) = f\left(\mathrm{x}\right) + \sum_{i=1}^{p} \lambda_i h_i\left(\mathrm{x}\right)$$

在最优点处对x和乘子变量的导数都必须为0：

$$\nabla_{\mathrm{x}} f+\sum_{i=1}^{p} \lambda_i \nabla_{\mathrm{x}} h_i=0$$

$$h_i(\mathrm{x})=0$$

解这个方程即可得到最优解。对拉格朗日乘数法更详细的讲解可以阅读任何一本高等数学教材。机器学习中用到拉格朗日乘数法的地方有：

主成分分析
线性判别分析
流形学习中的拉普拉斯特征映射
隐马尔科夫模型

5.凸优化

数值优化算法面临两个方面的问题：局部极值，鞍点。前者是梯度为0的点，也是极值点，但不是全局极小值；后者连局部极值都不是，在鞍点处Hessian矩阵不定，即既非正定，也非负定。

凸优化通过对目标函数，优化变量的可行域进行限定，可以保证不会遇到上面两个问题。凸优化是一类特殊的优化问题，它要求：

优化变量的可行域是一个凸集
目标函数是一个凸函数

凸优化最好的一个性质是：所有局部最优解一定是全局最优解。对于凸优化更详细的讲解可以阅读SIGAI之前的公众号文章“理解凸优化”。机器学习中典型的凸优化问题有：

线性回归
岭回归
LASSO回归
Logistic回归
支持向量机
Softamx回归

6.拉格朗日对偶

对偶是最优化方法里的一种方法，它将一个最优化问题转换成另外一个问题，二者是等价的。拉格朗日对偶是其中的典型例子。对于如下带等式约束和不等式约束的优化问题：

$$\begin{aligned}&\min f(\mathrm{x})\\&g_i(\mathrm{x})\le 0 \quad \mathrm{i}=1,...,m\\&h_i(\mathrm{x})=0 \qquad \mathrm{i}=1,...,p\end{aligned}$$

与拉格朗日乘数法类似，构造广义拉格朗日函数：

$$L(\mathrm{x},\lambda,\nu)=f(\mathrm{x})+\sum_{i=1}^{m}\lambda_i g_i(\mathrm{x})+\sum_{i=1}^{p}\nu_i h_i(\mathrm{x})$$

λ_i必须满足$\lambda_i \geq 0$的约束。原问题为：

$$\min_{\mathrm{x}} \max_{\lambda,\nu,\lambda_i \geq 0} L(\mathrm{x},\lambda,\nu)$$

即先固定住x，调整拉格朗日乘子变量，让函数L取极大值；然后控制变量x，让目标函数取极小值。原问题与我们要优化的原始问题是等价的。

对偶问题为：

$$\max_{\lambda,\nu,\lambda_i \geq 0} \min_{\mathrm{x}} L(\mathrm{x},\lambda,\nu)$$

和原问题相反，这里是先控制变量x，让函数L取极小值；然后控制拉格朗日乘子变量，让函数取极大值。

一般情况下，原问题的最优解大于等于对偶问题的最优解，这称为弱对偶。在某些情况下，原问题的最优解和对偶问题的最优解相等，这称为强对偶。

强对偶成立的一种条件是Slater条件：一个凸优化问题如果存在一个候选x使得所有不等式约束都是严格满足的，即对于所有的i都有$g_i(\mathrm{x}) < 0$，不等式不取等号，则强对偶成立，原问题与对偶问题等价。注意，Slater条件是强对偶成立的充分条件而非必要条件。

拉格朗日对偶在机器学习中的典型应用是支持向量机。

7.KKT条件

KKT条件是拉格朗日乘数法的推广，用于求解既带有等式约束，又带有不等式约束的函数极值。对于如下优化问题：

$$\begin{aligned} &\min f(\mathrm{x}) \\ &g_i(\mathrm{x}) \leq 0 \quad \mathrm{i}=1,\ldots,q \\ &h_i(\mathrm{x}) = 0 \quad \mathrm{i}=1,\ldots,p \end{aligned}$$

和拉格朗日对偶的做法类似，KKT条件构如下乘子函数：

$$L(\mathrm{x},\lambda,\mu) = f(\mathrm{x}) + \sum_{j=1}^{p} \lambda_j h_j(\mathrm{x}) + \sum_{k=1}^{q} \mu_k g_k(\mathrm{x})$$

λ和μ称为KKT乘子。在最优解处x^*应该满足如下条件：

$$\begin{aligned} &\nabla_{\mathrm{x}} L(\mathrm{x}^*) = 0 \\ &\mu_k \geq 0 \\ &\mu_k g_k(\mathrm{x}^*) = 0 \\ &h_j(\mathrm{x}^*) = 0 \\ &g_k(\mathrm{x}^*) \leq 0 \end{aligned}$$

等式约束

$$h_j(\mathrm{x}^*) = 0$$

和不等式约束

$$g_k(\mathrm{x}^*) \leq 0$$

是本身应该满足的约束，$\nabla_x L(x^*)=0$和之前的拉格朗日乘数法一样。唯一多了关于$g_i(x)$的条件：

$$\mu_k g_k(x^*)=0$$

KKT条件只是取得极值的必要条件而不是充分条件。

8.特征值与特征向量

对于一个n阶矩阵A，如果存在一个数λ和一个非0向量X，满足：

$$Ax=\lambda x$$

则称λ为矩阵A的特征值，X为该特征值对应的特征向量。根据上面的定义有下面线性方程组成立：

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

根据线性方程组的理论，要让齐次方程有非0解，系数矩阵的行列式必须为0，即：

$$|A-\lambda I|=0$$

上式左边的多项式称为矩阵的特征多项式。矩阵的迹定义为主对角线元素之和：

$$tr(A)=\sum_{i=1}^{n} a_{ii}$$

根据韦达定理，矩阵所有特征值的和为矩阵的迹：

$$\sum_{i=1}^{n}\lambda_i = tr(A)$$

同样可以证明，矩阵所有特征值的积为矩阵的行列式：

$$\prod_{i=1}^{n}\lambda_i = |A|$$

利用特征值和特征向量，可以将矩阵对角化，即用正交变换将矩阵化为对角阵。实对称矩阵一定可以对角化，半正定矩阵的特征值都大于等于0，在机器学习中，很多矩阵都满足这些条件。特征值和特征向量在机器学习中的应用包括：正态贝叶斯分类器、主成分分析，流形学习，线性判别分析，谱聚类等。

9.奇异值分解

矩阵对角化只适用于方阵，如果不是方阵也可以进行类似的分解，这就是奇异值分解，简称SVD。假设A是一个*m x n*的矩阵，则存在如下分解：

$$A=U\Sigma V^T$$

其中U为*m x m*的正交矩阵，其列称为矩阵A的左奇异向量；Σ为*m x n*的对角矩阵，除了主对角线σ_{ii}以外，其他元素都是0；V为*n x n*的正交矩阵，其行称为矩阵A的右奇异向量。U的列为AA^T的特征向量，V的列为A^T A的特征向量。

10.最大似然估计

有些应用中已知样本服从的概率分布，但是要估计分布函数的参数θ，确定这些参数常用的一种方法是最大似然估计。

最大似然估计构造一个似然函数，通过让似然函数最大化，求解出θ。最大似然估计的直观解释是，寻求一组参数，使得给定的样本集出现的概率最大。

假设样本服从的概率密度函数为$p(\mathrm{x};\theta)$，其中X为随机变量，θ为要估计的参数。给定一组样本$\mathrm{x}_i, i=1,\ldots,l$，它们都服从这种分布，并且相互独立。最大似然估计构造如下似然函数：

$$L(\theta)=\prod_{i=1}^{l} p(\mathrm{x}_i;\theta)$$

其中x_i是已知量，这是一个关于θ的函数，我们要让该函数的值最大化，这样做的依据是这组样本发生了，因此应该最大化它们发生的概率，即似然函数。这就是求解如下最优化问题：

$$\max \prod_{i=1}^{l} p(\mathrm{x}_i;\theta)$$

乘积求导不易处理，因此我们对该函数取对数，得到对数似然函数：

$$\ln L(\theta)=\ln \prod_{i=1}^{l} p(\mathrm{x}_i;\theta)=\sum_{i=1}^{l} \ln p(\mathrm{x}_i;\theta)$$

最后要求解的问题为：

$$\max \sum_{i=1}^{l} \ln p(\mathrm{x}_i;\theta)$$

最大似然估计在机器学习中的典型应用包括logistic回归，贝叶斯分类器，隐马尔科夫模型等。

基本概念

1.有监督学习与无监督学习

根据样本数据是否带有标签值，可以将机器学习算法分成有监督学习和无监督学习两类。有监督学习的样本数据带有标签值，它从训练样本中学习得到一个模型，然后用这个模型对新的样本进行预测推断。有监督学习的典型代表是分类问题和回归问题。

无监督学习对没有标签的样本进行分析，发现样本集的结构或者分布规律。无监督学习的典型代表是聚类，表示学习，和数据降维，它们处理的样本都不带有标签值。

2.分类问题与回归问题

在有监督学习中，如果样本的标签是整数，则预测函数是一个向量到整数的映射，这称为分类问题。如果标签值是连续实数，则称为回归问题，此时预测函数是向量到实数的映射。

3.生成模型与判别模型

分类算法可以分成判别模型和生成模型。给定特征向量x与标签值y，生成模型对联合概率$p(x,y)$建模，判别模型对条件概率p(y|x)进行建模。另外，不使用概率模型的分类器也被归类为判别模型，它直接得到预测函数而不关心样本的概率分布：

$$y=f(\mathrm{x})$$

判别模型直接得到预测函数f(x)，或者直接计算概率值p(y|x)，比如SVM和logistic回归，softmax回归，判别模型只关心决策面，而不管样本的概率分布的密度。

生成模型计算p(x, y)或者p(x|y) ，通俗来说，生成模型假设每个类的样本服从某种概率分布，对这个概率分布进行建模。

机器学习中常见的生成模型有贝叶斯分类器，高斯混合模型，隐马尔可夫模型，受限玻尔兹曼机，生成对抗网络等。典型的判别模型有决策树，kNN算法，人工神经网络，支持向量机，logistic回归，AdaBoost算法等。

4.交叉验证

交叉验证（cross validation）是一种统计准确率的技术。*k*折交叉验证将样本随机、均匀的分成*k*份，轮流用其中的*k-1*份训练模型，1份用于测试模型的准确率，用*k*个准确率的均值作为最终的准确率。

5.过拟合与欠拟合

欠拟合也称为欠学习，直观表现是训练得到的模型在训练集上表现差，没有学到数据的规律。引起欠拟合的原因有模型本身过于简单，例如数据本身是非线性的但使用了线性模型；特征数太少无法正确的建立映射关系。

过拟合也称为过学习，直观表现是在训练集上表现好，但在测试集上表现不好，推广泛化性能差。过拟合产生的根本原因是训练数据包含抽样误差，在训练时模型将抽样误差也进行了拟合。所谓抽样误差，是指抽样得到的样本集和整体数据集之间的偏差。引起过拟合的可能原因有：

模型本身过于复杂，拟合了训练样本集中的噪声。此时需要选用更简单的模型，或者对模型进行裁剪。训练样本太少或者缺乏代表性。此时需要增加样本数，或者增加样本的多样性。训练样本噪声的干扰，导致模型拟合了这些噪声，这时需要剔除噪声数据或者改用对噪声不敏感的模型。

6.偏差与方差分解

模型的泛化误差可以分解成偏差和方差。偏差是模型本身导致的误差，即错误的模型假设所导致的误差，它是模型的预测值的数学期望和真实值之间的差距。

方差是由于对训练样本集的小波动敏感而导致的误差。它可以理解为模型预测值的变化范围，即模型预测值的波动程度。

模型的总体误差可以分解为偏差的平方与方差之和：

$$\mathrm{E}\left(\left(y-f\left(\mathrm{x}\right)\right)^2\right)=\mathrm{Bias}^2\left(f\left(\mathrm{x}\right)\right)+\mathrm{Var}\left(f(\mathrm{x})\right)+\sigma^2$$

如果模型过于简单，一般会有大的偏差和小的方差；反之如果模型复杂则会有大的方差但偏差很小。

7.正则化

为了防止过拟合，可以为损失函数加上一个惩罚项，对复杂的模型进行惩罚，强制让模型的参数值尽可能小以使得模型更简单，加入惩罚项之后损失函数为：

$$L(\theta)=\frac{1}{2l}\sum_{i=1}^{l}\left(f_{\theta}(\mathrm{x}_i)-y_i\right)^2+\frac{\lambda}{2}r(\theta)$$

正则化被广泛应用于各种机器学习算法，如岭回归，LASSO回归，logistic回归，神经网络等。除了直接加上正则化项之外，还有其他强制让模型变简单的方法，如决策树的剪枝算法，神经网络训练中的dropout技术，提前终止技术等。

8.维数灾难

为了提高算法的精度，会使用越来越多的特征。当特征向量维数不高时，增加特征确实可以带来精度上的提升；但是当特征向量的维数增加到一定值之后，继续增加特征反而会导致精度的下降，这一问题称为维数灾难。

贝叶斯分类器

贝叶斯分类器将样本判定为后验概率最大的类，它直接用贝叶斯公式解决分类问题。假设样本的特征向量为x，类别标签为y，根据贝叶斯公式，样本属于每个类的条件概率（后验概率）为：

$$p(y|\mathrm{x})=\frac{p(\mathrm{x}|y)p(y)}{p(\mathrm{x})}$$

分母p(x)对所有类都是相同的，分类的规则是将样本归到后验概率最大的那个类，不需要计算准确的概率值，只需要知道属于哪个类的概率最大即可，这样可以忽略掉分母。分类器的判别函数为：

$$\arg\max_y p(\mathrm{x}|y)p(y)$$

在实现贝叶斯分类器时，需要知道每个类的条件概率分布p(x|y)即先验概率。一般假设样本服从正态分布。训练时确定先验概率分布的参数，一般用最大似然估计，即最大化对数似然函数。

如果假设特征向量的各个分量之间相互独立，则称为朴素贝叶斯分类器，此时的分类判别函数为：

$$\arg\max_y p(y=c_i)\prod_{j=1}^{n}p(x_j|y=c_i)$$

实现时可以分为特征分量是离散变量和连续变量两种情况。贝叶斯分分类器是一种生成模型，可以处理多分类问题，是一种非线性模型。

决策树

决策树是一种基于规则的方法，它用一组嵌套的规则进行预测。在树的每个决策节点处，根据判断结果进入一个分支，反复执行这种操作直到到达叶子节点，得到预测结果。这些规则通过训练得到，而不是人工制定的。

决策树既可以用于分类问题，也可以用于回归问题。分类树的映射函数是多维空间的分段线性划分，用平行于各坐标轴的超平面对空间进行切分；回归树的映射函数是分段常数函数。决策树是分段线性函数而不是线性函数。只要划分的足够细，分段常数函数可以逼近闭区间上任意函数到任意指定精度，因此决策树在理论上可以对任意复杂度的数据进行拟合。对于分类问题，如果决策树深度够大，它可以将训练样本集的所有样本正确分类。

决策树的训练算法是一个递归的过程，首先创建根节点，然后递归的建立左子树和右子树。如果练样本集为D，训练算法的流程为：

1.用样本集D建立根节点，找到一个判定规则，将样本集分裂成D1和D2两部分，同时为根节点设置判定规则。
2.用样本集D1递归建立左子树。
3.用样本集D2递归建立右子树。
4.如果不能再进行分裂，则把节点标记为叶子节点，同时为它赋值。

对于分类树，如果采用Gini系数作为度量准则，决策树在训练时寻找最佳分裂的依据为让Gini不纯度最小化，这等价于让下面的值最大化：

$$G = \frac{\sum_i N_{L,i}^2}{N_L} + \frac{\sum_i N_{R,i}^2}{N_R}$$

寻找最佳分裂时需要计算用每个阈值对样本集进行分裂后的纯度值，寻找该值最大时对应的分裂，它就是最佳分裂。如果是数值型特征，对于每个特征将l个训练样本按照该特征的值从小到大排序，假设排序后的值为：

$$x_1, \ldots, x_l$$

接下来从x_1开始，依次用每个x_i作为阈值，将样本分成左右两部分，计算上面的纯度值，该值最大的那个分裂阈值就是此特征的最佳分裂阈值。在计算出每个特征的最佳分裂阈值和上面的纯度值后，比较所有这些分裂的纯度值大小，该值最大的分裂为所有特征的最佳分裂。

决策树可以处理属性缺失问题，采用的方法是使用替代分裂规则。为了防止过拟合，可以对树进行剪枝，让模型变得更简单。如果想要更详细的了解决策树的原理，请阅读SIGAI之前的公众号文章“理解决策树”，在SIGAI云端实验室有决策树训练算法的原理实验，此功能免费，网址为：

www.sigai.cn

决策树是一种判别模型，既支持分类问题，也支持回归问题，是一种非线性模型，它支持多分类问题。

随机森林

随机森林是一种集成学习算法，是Bagging算法的具体实现。集成学习是机器学习中的一种思想，而不是某一具体算法，它通过多个模型的组合形成一个精度更高的模型，参与组合的模型

称为弱学习器。在预测时使用这些弱学习器模型联合进行预测，训练时需要依次训练出这些弱学习器。

随机森林用有放回抽样（Bootstrap抽样）构成出的样本集训练多棵决策树，训练决策树的每个节点时只使用了随机抽样的部分特征。预测时，对于分类问题，一个测试样本会送到每一棵决策树中进行预测，然后投票，得票最多的类为最终分类结果。对于回归问题，随机森林的预测输出是所有决策树输出的均值。

假设有n个训练样本。训练每一棵树时，从样本集中有放回的抽取n个样本，每个样本可能会被抽中多次，也可能一次都没抽中。如果样本量很大，在整个抽样过程中每个样本有0.368的概率不被抽中。由于样本集中各个样本是相互独立的，在整个抽样中所有样本大约有36.8%没有被抽中。这部分样本称为包外（Out Of Bag，简称OOB）数据。

用这个抽样的样本集训练一棵决策树，训练时，每次寻找最佳分裂时，还要对特征向量的分量采样，即只考虑部分特征分量。由于使用了随机抽样，随机森林泛化性能一般比较好，可以有效的降低模型的方差。

如果想更详细的了解随机森林的原理，请阅读SIGAI之前的公众号文章“随机森林概述”。随机森林是一种判别模型，既支持分类问题，也支持回归问题，并且支持多分类问题，这是一种非线性模型。

AdaBoost算法

AdaBoost算法也是一种集成学习算法，用于二分类问题，是Boosting算法的一种实现。它用多个弱分类器的线性组合来预测，训练时重点关注错分的样本，准确率高的弱分类器权重大。AdaBoost算法的全称是自适应，它用弱分类器的线性组合来构造强分类器。弱分类器的性能不用太好，仅比随机猜测强，依靠它们可以构造出一个非常准确的强分类器。强分类器的计算公式为：

$$F(\mathrm{x})=\sum_{t=1}^{T}\alpha_t f_t(\mathrm{x})$$

其中x是输入向量，F(x)是强分类器，ft(x)是弱分类器，at是弱分类器的权重，T为弱分类器的数量，弱分类器的输出值为+1或-1，分别对应正样本和负样本。分类时的判定规则为：

$$\mathrm{sgn}\left(F(\mathrm{x})\right)$$

强分类器的输出值也为+1或-1，同样对应于正样本和负样本。

训练时，依次训练每一个若分类器，并得到它们的权重值。训练样本带有权重值，初始时所有样本的权重相等，在训练过程中，被前面的弱分类器错分的样本会加大权重，反之会减小权重，这样接下来的弱分类器会更加关注这些难分的样本。弱分类器的权重值根据它的准确率构造，精度越高的弱分类器权重越大。

给定l个训练样本(x_i, y_i)，其中x_i是特征向量，y_i为类别标签，其值为+1或-1。训练算法的流程为：

初始化样本权重值，所有样本的初始权重相等：

$$w_i^0 = 1/l, i = 1,...,l$$

循环，对$t = 1,...,T$依次训练每个弱分类器：

训练一个弱分类器$f_t(\mathrm{x})$，并计算它对训练样本集的错误率e_t

计算弱分类器的权重：

$$\alpha_t = \frac{1}{2}\log\left((1-e_t)/e_t\right)$$

更新所有样本的权重：

$$w_i^t = w_i^{t-1}\exp\left(-y_i\alpha_t f_t(\mathrm{x}_i)\right)/Z_t$$

其中Z_t为归一化因子，它是所有样本的权重之和：

$$Z_t = \sum_{i=1}^{n} w_i^{t-1}\exp\left(-y_i\alpha_t f_t(\mathrm{x}_i)\right)$$

结束循环

最后得到强分类器：

$$\mathrm{sgn}\left(F(\mathrm{x})\right) = \mathrm{sgn}\left(\sum_{t=1}^{T}\alpha_t f_t(\mathrm{x})\right)$$

根据计算公式，错误率低的弱分类器权重大，它是准确率的增函数。AdaBoost算法在训练样本集上的错误率会随着弱分类器数量的增加而指数级降低。它能有效的降低模型的偏差。

AdaBoost算法从广义加法模型导出，训练时求解的是指数损失函数的极小值：

求解时采用了分阶段优化，先得到弱分类器，然后确定弱分类器的权重值，这就是弱分类器，弱分类器权重的来历。除了离散型AdaBoost之外，从广义加法模型还可以导出其他几种AdaBoost算法，分别是实数型AdaBoost，Gentle型AdaBoost，Logit型AdaBoost，它们使用了不同的损失函数和最优化算法。

标准的AdaBoost算法是一种判别模型，只能支持二分类问题。它的改进型可以处理多分类问题。

主成分分析

主成分分析是一种数据降维和去除相关性的方法，它通过线性变换将向量投影到低维空间。对向量进行投影就是对向量左乘一个矩阵，得到结果向量：

$$y = Wx$$

结果向量的维数小于原始向量的维数。降维要确保的是在低维空间中的投影能很好的近似表达原始向量，即重构误差最小化：

$$L = \sum_{i=1}^{n} \left\| \mathrm{m} + \sum_{j=1}^{d'} a_{ij} \mathrm{e}_j - \mathrm{x}_i \right\|^2$$

其中e为投影后空间的基向量，是标准正交基；a为重构系数，也是投影到低维空间后的坐标。如果定义如下的散布矩阵：

$$\mathrm{S} = \sum_{i=1}^{n} (\mathrm{x}_i - \mu)(\mathrm{x}_i - \mu)^{\mathrm{T}}$$

其中m和μ为所有样本的均值向量。则上面的重构误差最小化等价于求解如下问题：

$$\min_{\mathrm{W}} -\mathrm{tr}\left(\mathrm{W}^{\mathrm{T}}\mathrm{S}\mathrm{W}\right)$$

$$\mathrm{W}^{\mathrm{T}}\mathrm{W} = \mathrm{I}$$

通过拉格朗日乘数法可以证明，使得该函数取最小值的e_j为散度矩阵最大的d'个特征值对应的单位长度特征向量。矩阵W的列e_j是我们要求解的基向量，由它们构成投影矩阵。计算时，先计算散布矩阵（或者协方差矩阵），再对该进行进行特征值分解，找到最大的一部分特征值和对应的特征向量，构成投影矩阵。可以证明，协方差矩阵或散布矩阵是实对称半正定矩阵，因此所有特征值非负。进行降维时，先将输入向量减掉均值向量，然后左乘投影矩阵，即可得到投影后的向量。

主成分分析一种无监督学习算法，也是一种线性方法。

线性判别分析

线性判别分析向最大化类间差异、最小化类内差异的方向线性投影。其基本思想是通过线性投影来最小化同类样本间的差异，最大化不同类样本间的差异。具体做法是寻找一个向低维空间的投影矩阵W，样本的特征向量x经过投影之后得到的新向量：

$$y = Wx$$

同一类样投影后的结果向量差异尽可能小，不同类的样本差异尽可能大。简单的说，就是经过这个投影之后同一类的样本进来聚集在一起，不同类的样本尽可能离得远。这种最大化类间差异，最小化类内差异的做法，在机器学习的很多地方都有使用。

类内散布矩阵定义为：

$$\mathrm{S}_i = \sum_{\mathrm{x} \in D_i} (\mathrm{x} - \mathrm{m}_i)(\mathrm{x} - \mathrm{m}_i)^{\mathrm{T}}$$

$$\mathrm{S}_W = \sum_{i=1}^{c} \mathrm{S}_i$$

它衡量的内类样本的发散程度。其中m_j为每个类的均值向量，m为所有样本的均值向量。类间散布矩阵定义为：

$$\mathrm{S}_B = \sum_{i=1}^{c} n_i \left(\mathrm{m}_i - \mathrm{m}\right)\left(\mathrm{m}_i - \mathrm{m}\right)^{\mathrm{T}}$$

它衡量的了各类样本之间的差异。训练时的优化目标是类间差异与类内差异的比值：

$$\max L\left(\mathrm{W}\right) = \frac{\mathrm{tr}\left(\mathrm{W}^{\mathrm{T}}\mathrm{S}_B\mathrm{W}\right)}{\mathrm{tr}\left(\mathrm{W}^{\mathrm{T}}\mathrm{S}_W\mathrm{W}\right)}$$

上面的问题带有冗余，如果w是最优解，将其乘以一个不为0的系数之后还是最优解。为了消掉冗余，加上如下约束：

$$\mathrm{W}^{\mathrm{T}}\mathrm{S}_W\mathrm{W} = \mathrm{I}$$

然后使用拉格朗日乘数法，最后归结于求解矩阵的特征值与特征向量：

$$\mathrm{S}_B\mathrm{w}_i = \lambda_i \mathrm{S}_W\mathrm{w}_i$$

LDA是有监督的学习算法，在计算过程中利用了样本标签值，是线性模型。LDA也不能直接用于分类和回归问题，要对降维后的向量进行分类还需要借助其他算法。

kNN算法

kNN算法将样本分到离它最相似的样本所属的类。算法本质上使用了模板匹配的思想。要确定一个样本的类别，可以计算它与所有训练样本的距离，然后找出和该样本最接近的k个样本，统计这些样本的类别进行投票，票数最多的那个类就是分类结果。

由于需要计算样本间的距离，因此需要依赖距离定义，常用的有欧氏距离，Mahalanobis距离，Bhattacharyya距离。另外，还可以通过学习得到距离函数，这就是距离度量学习。

kNN算法是一种判别模型，即支持分类问题，也支持回归问题，是一种非线性模型。它天然的支持多分类问题。kNN算法没有训练过程，是一种基于实例的算法。

人工神经网络

人工神经网络是一种仿生的方法，参考了动物的神经元结构。从本质上看，它是一个多层复合函数。对于多层前馈型神经网络，即权连接网络，每一层实现的变换为：

$$\mathrm{u}^{(l)} = \mathrm{W}^{(l)}\mathrm{x}^{(l-1)} + \mathrm{b}^{(l)}$$

$$\mathrm{x}^{(l)} = f\left(\mathrm{u}^{(l)}\right)$$

其中W为权重矩阵，b为偏置向量，f为激活函数。正向传播时反复用上上对每一层的输出值进行计算，得到最终的输出。使用激活函数是为了保证非线性，对于激活函数更深入全面的介绍请参考SIGAI之前的公众号文章“理解神经网络的激活函数”，“神经网络的激活函数总结”。万能逼近定理保证了神经网络可以比较闭区间上任意一个连续函数。

权重和偏置通过训练得到，采用的是反向传播算法。反向传播算法从复合函数求导的链式法则导出，用于计算损失函数对权重，偏置的梯度值。算法从最外层的导数值算起，依次递推的计算更内层的导数值，这对应于从神经网络的输出层算起，反向计算每个隐含层参数的导数值。其核心是误差项的定义，定义误差项为损失函数对临时变量u的梯度：

$$\delta^{(l)} = \nabla_{\mathrm{u}^{(l)}} L = \begin{cases} \left(\mathrm{x}^{(l)} - \mathrm{y}\right) \odot f'\left(\mathrm{u}^{(l)}\right) & l = n_l \\ \left(\mathrm{W}^{(l+1)}\right)^{\mathrm{T}} \left(\delta^{(l+1)}\right) \odot f'\left(\mathrm{u}^{(l)}\right) & l \neq n_l \end{cases}$$

其中，n_l是神经网络的层数。最后一个层的误差项可以直接求出，其他层的误差项根据上面的递推公式进行计算。根据误差项，可以计算出损失函数对每一层权重矩阵的梯度值：

$$\nabla_{\mathrm{w}^{(l)}} L = \delta^{(l)} \left(\mathrm{x}^{(l-1)}\right)^{\mathrm{T}}$$

以及对偏置向量的梯度值：

$$\nabla_{\mathrm{b}^{(l)}} L = \delta^{(l)}$$

然后用梯度下降法对它们的值进行更新。参数初始化一般采用随机数，而不是简单的初始化为0。为了加快收敛速度，还可以使用动量项，它积累了之前的梯度信息。

神经网络训练时的损失函数不是凸函数，因此有陷入局部极值，鞍点的风险。另外，随着层数的增加，会导致梯度消失问题，这是因为每次计算误差项时都需要乘以激活函数的导数值，如果其绝对值小于1，多次连乘之后导致误差项趋向于0，从而使得计算出来的参数梯度值接近于0，参数无法有效的更新。

如果对反传播算法的推导细节感兴趣，可以阅读SIGAI之前的公众号文章“反向传播算法推导-全连接神经网络”。神经网络的训练算法可以总结为：

复合函数求导 + 梯度下降法

标准的神经网络是一种有监督的学习算法，它是一种非线性模型，它既可以用于分类问题，也可以用于回归问题，并且支持多分类问题。

支持向量机

支持向量机的核心思想是最大化分类间隔。简单的支持向量机就是让分类间隔最大化的线性分类器，找到多维空间中的一个超平面。它在训练是求解的问题为：

$$\begin{aligned} &\min \ \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} + C\sum_{i=1}^{l}\xi_i \\ &y_i(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0, i = 1, \ldots, l \end{aligned}$$

这从点到超平面的距离方程导出，通过增加一个约束条件消掉了优化变量的冗余。可以证明，这个问题是凸优化问题，并且满足Slater条件。这个问题带有太多的不等式约束，不易求解，因此通过拉格朗日对偶转换为对偶问题求解：

$$\begin{aligned} &\min_{\alpha} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j - \sum_{k=1}^{l}\alpha_k \\ &0 \leq \alpha_i \leq C \\ &\sum_{j=1}^{l}\alpha_j y_j = 0 \end{aligned}$$

同样的，这个问题也是凸优化问题。此时支持向量机并不能解决非线性分类问题，通过使用核函数，将向量变换到高维空间，使它们更可能是线性可分的。而对向量先进行映射再做内积，等价于先做内积再做映射，因此核函数并不用显式的对向量进行映射，而是对两个向量的内积

进行映射，这是核函数的精髓。要理解核函数，可以阅读SIGAI之前的公众号文章“【实验】理解SVM的核函数和参数”。

加入核函数K之后的对偶问题变为：

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K\left(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}_j\right)-\sum_{k=1}^{l}\alpha_k$$

$$0 \le \alpha_i \le C$$

$$\sum_{j=1}^{l}\alpha_j y_j = 0$$

预测函数为：

$$\mathrm{sgn}\left(\sum_{i=1}^{l}\alpha_i y_i K\left(\mathrm{x}_i^{\mathrm{T}}\mathrm{x}\right)+b\right)$$

其中b通过KKT条件求出。如果使用正定核，这个问题也是凸优化问题。求解采用了SMO算法，这是一种分治法，每次挑选出两个变量进行优化，其他变量保持不动。选择优化变量的依据是KKT条件，对这两个变量的优化是一个带等式和不等式约束的二次函数极值问题，可以直接得到公式解。另外，这个子问题同样是一个凸优化问题。

标准的支持向量机只能解决二分类问题。对于多分类问题，可以用这种二分类器的组合来解决，有以下几种方案：

1对剩余方案。对于有*k*个类的分类问题，训练*k*个二分类器。训练时第*i*个分类器的正样本是第*i*类样本，负样本是除第*i*类之外其他类型的样本，这个分类器的作用是判断样本是否属于第*i*类。在进行分类时，对于待预测样本，用每个分类器计算输出值，取输出值最大那个作为预测结果。

1对1方案。如果有*k*个类，训练C_k^2个二分类器，即这些类两两组合。训练时将第*i*类作为正样本，其他各个类依次作为负样本，总共有$k(k-1)/2$种组合。每个分类器的作用是判断样本是属于第*i*类还是第*j*类。对样本进行分类时采用投票的方法，依次用每个二分类器进行预测，如果判定为第m类，则m类的投票数加1，得票最多的那个类作为最终的判定结果。

除了通过二分类器的组合来构造多类分类器之外，还可以通过直接优化多类分类的目标函数得到多分类器。

SVM是一种判别模型。它既可以用于分类问题，也可以用于回归问题。标准的SVM只能支持二分类问题，使用多个分类器的组合，可以解决多分类问题。如果不使用核函数，SVM是一个线性模型，如果使用非线性核，则是非线性模型，这可以从上面的预测函数看出。如果想更详细的了解支持向量机，可以阅读SIGAI之前的公众号文章“用一张图理解SVM的脉络”。

logistic回归

logistic回归是一种二分类算法，直接为样本估计出它属于正负样本的概率。先将向量进行线性加权，然后计算logistic函数，可以得到[0,1]之间的概率值，它表示样本x属于正样本的概

率：

$$h(\mathrm{x})=\frac{1}{1+\exp\left(-\mathrm{w}^{\mathrm{T}}\mathrm{x}\right)}$$

正样本标签值为1，负样本为0。使用logistic函数的原因是它单调增，并且值域在(0, 1)之间，刚好符合概率的要求。训练时采用最大似然估计，求解对数似然函数的极值：

$$\max\sum_{i=1}^{l}\left(y_i\log h\left(\mathrm{x}_i\right)+\left(1-y_i\right)\log\left(1-h\left(\mathrm{x}_i\right)\right)\right)$$

可以证明这是一个凸优化问题，求解时可以用梯度下降法，也可以用牛顿法。如果正负样本的标签为+1和-1，则可以采用另外一种写法：

$$p\left(y=\pm1\left|\mathrm{x},\mathrm{w}\right.\right)=\frac{1}{1+\exp\left(-y\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}+b\right)\right)}$$

训练时的目标同样是最大化对数似然函数：

$$\min_{\mathrm{w},b}\sum_{i=1}^{l}\log\left(1+\exp\left(-y_i\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}_i+b\right)\right)\right)$$

同样的，这也是一个凸优化问题。预测时并不需要计算logistic函数，而是直接计算：

$$\mathrm{sgn}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x}+b\right)$$

Logistic回归是一种二分类算法，虽然使用了概率，但它是一种判别模型！另外要注意的是，logistic回归是一种线性模型，这从它的预测函数就可以看出。它本身不能支持多分类问题，它的扩展版本softmax回归可以解决多分类问题。

K均值算法

K均值算法是一种聚类算法，把样本分配到离它最近的类中心所属的类，类中心由属于这个类的所有样本确定。

k均值算法是一种无监督的聚类算法。算法将每个样本分配到离它最近的那个类中心所代表的类，而类中心的确定又依赖于样本的分配方案。

在实现时，先随机初始化每个类的类中心，然后计算样本与每个类的中心的距离，将其分配到最近的那个类，然后根据这种分配方案重新计算每个类的中心。这也是一种分阶段优化的策略。

与k近邻算法一样，这里也依赖于样本之间的距离，因此需要定义距离的计算方式，最常用的是欧氏距离，也可以采用其他距离定义。算法在实现时要考虑下面几个问题：

1.类中心向量的初始化。一般采用随机初始化。最简单的是Forgy算法，它从样本集中随机选择k个样本作为初始类中心。第二种方案是随机划分，它将所有样本随机的分配给k个类中的一个，然后按照这种分配方案计算各个类的类中心向量。

2.参数k的设定。可以根据先验知识人工指定一个值，或者由算法自己确定。

3.迭代终止的判定规则。一般做法是计算本次迭代后的类中心和上一次迭代时的类中心之间的距离，如果小于指定阈值，则算法终止。

卷积神经网络

卷积神经网络是对全连接神经网络的发展，它使用卷积层，池化层自动学习各个尺度上的特征。卷积运算为：

$$x_{ij}^{(l)} = f\left(u_{ij}^{(l)}\right) = f\left(\sum_{p=1}^{s}\sum_{q=1}^{s} x_{i+p-1,j+q-1}^{(l-1)} \times k_{pq}^{(l)} + b^{(l)}\right)$$

在这里需要注意多通道卷积的实现，它的输入图像，卷积核都有多个通道，分别用各个通道的卷积核对输入图像的各个通道进行卷积，然后再累加。这里也使用了激活函数，原因和全连接神经网络相同。池化运算最常见的有均值池化，max池化，分别用均值和最大值代替图像的一块矩形区域。使用池化的原因是为了降维，减小图像的尺寸，另外，它还带来了一定程度的平移和旋转的不变性。Max池化是非线性操作，现在用的更多。

对于经典的网络结构，包括LeNet-5网络，AlexNet，VGG网络，GoogLeNet，残差网络等经典的网络结构，创新点，要熟记于心。

自Alex网络出现之后，各种改进的卷积网络不断被提出。这些改进主要在以下几个方面进行：卷积层，池化层，激活函数，损失函数，网络结构。对于这些典型的改进，也要深刻理解。

由于引入了卷积层和池化层，因此反向传播算法需要为这两种层进行考虑。卷积层误差项的反向传播的公式为

$$\delta^{(l-1)} = \delta^{(l)} * \mathrm{rot180}(\mathrm{K}) \odot f'\left(\mathrm{u}^{(l-1)}\right)$$

根据误差项计算卷积核梯度值的公式为：

$$\nabla_{\mathrm{K}^{(l)}} L = \mathrm{conv}\left(\mathrm{X}^{(l-1)}, \delta^{(l)}\right)$$

如果采用均值池化，池化层的误差项反向传播计算公式为：

$$\begin{bmatrix} \frac{\delta}{s \times s} & \cdots & \frac{\delta}{s \times s} \\ \cdots & \cdots & \cdots \\ \frac{\delta}{s \times s} & \cdots & \frac{\delta}{s \times s} \end{bmatrix}$$

如果使用max池化，则为：

$$\begin{bmatrix} 0 & \cdots & 0 \\ 0 & \cdots & \cdots \\ \delta & \cdots & 0 \end{bmatrix}$$

注意，池化层没有需要训练得到的参数。如果对卷积神经网络反向传播算法的推导感兴趣，可以阅读SIGAI之前的公众号文章“反向传播算法推导-卷积神经网络”。

卷积神经网络具有迁移学习的能力，我们可以把这个网络的参数作为训练的初始值，在新的任务上继续训练，这种做法称为fine-tune，即网络微调。大量的实验结果和应用结果证明，这

种微调是有效的。这说明卷积神经网络在一定程度上具有迁移学习的能力，卷积层学习到的特征具有通用性。VGG网络在ImageNet数据集上的训练结果在进行微调之后，被广泛应用于目标检测、图像分割等任务。

和全连接神经网络一样，卷积神经网络是一个判别模型，它既可以用于分类问题，也可以用用于回归问题，并且支持多分类问题。

循环神经网络

循环神经网络是一种具有记忆功能的神经网络，每次计算时，利用了上一个时刻的记忆值，特别适合序列数据分析。网络接受的是一个序列数据，即一组向量，依次把它们输入网络，计算每个时刻的输出值。记忆功能通过循环神层实现：

$$\mathrm{h}_t = f\left(\mathrm{W}_{xh}\mathrm{x}_t + \mathrm{W}_{hh}\mathrm{h}_{t-1} + \mathrm{b}\right)$$

它同时利用了本时刻的输入值和上一个时刻的记忆值。输出层的变换为：

$$\mathrm{y}_t = g\left(\mathrm{W}_o\mathrm{h}_t + \mathrm{b}_o\right)$$

这和普通神经网络没什么区别。由于引入了循环层，因此反向传播算法有所不同，称为BPTT，即时间轴上的反向传播算法。算法从最后一个时刻算起，沿着时间轴往前推。误差项的递推公式为：

$$\delta_t = \left(\left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\left(\nabla_{\mathrm{y}_t^*}L_t\right)\odot g^{'}\left(\mathrm{v}_t\right)\right)\right)\odot f^{'}\left(\mathrm{u}_t\right) + \left(\mathrm{W}_{hh}\right)^{\mathrm{T}}\delta_{t+1}\odot f^{'}\left(\mathrm{u}_t\right)$$

递推的终点为最后一个时刻。

$$\delta_T = \left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\nabla_{\mathrm{v}_T}L\right)\odot f^{'}\left(\mathrm{u}_T\right) = \left(\mathrm{W}_o\right)^{\mathrm{T}}\left(\left(\nabla_{\mathrm{y}_T^*}L\right)\odot g^{'}\left(\mathrm{v}_T\right)\right)\odot f^{'}\left(\mathrm{u}_T\right)$$

根据误差项计算对权重和偏置的梯度值的公式为：

$$\nabla_{\mathrm{W}_{hh}}L = \sum_{t=1}^{T}\left(\nabla_{\mathrm{u}_t}L\right)\mathrm{h}_{t-1}^{\mathrm{T}} = \sum_{t=1}^{T}\delta_t\mathrm{h}_{t-1}^{\mathrm{T}}$$

$$\nabla_{\mathrm{W}_{xh}}L = \sum_{t=1}^{T}\left(\nabla_{\mathrm{u}_t}L\right)\mathrm{x}_t^{\mathrm{T}} = \sum_{t=1}^{T}\delta_t\mathrm{x}_t^{\mathrm{T}}$$

$$\nabla_{\mathrm{b}_h}L = \sum_{t=1}^{T}\nabla_{\mathrm{u}_t}L = \sum_{t=1}^{T}\delta_t$$

循环神经网络同样存在梯度消失问题，因此出现了LSTM，GRU等结构。

以循环神经网络为基础，构造出了两类通用的框架，分别是连接主义时序分类（CTC），以及序列到序列学习（seq2seq）。用于解决语音识别，自然语言处理中的问题。其中，seq2seq采用了编码器-解码器结构，用两个循环神经网络组合起来完成计算，一个充当编码器，一个充当解码器。

和其他类型的神经网络一样，循环神经网络是一个判别模型，既支持分类问题，也支持回归问题，并且支持多分类问题。

高斯混合模型

高斯混合模型通过多个正态分布的加权和来描述一个随机变量的概率分布，概率密度函数定义为：

$$p(\mathrm{x}) = \sum_{i=1}^{k} w_i N_i(\mathrm{x}; \mu_i, \Sigma_i)$$

其中x为随机向量，k为高斯分布的个数，w_i为权重，μ为高斯分布的均值向量，Σ为协方差矩阵。所有权重之和为1，即：

$$\sum_{i=1}^{k} w_i = 1$$

任意一个样本可以看作是先从k个高斯分布中选择出一个，选择第i个高斯分布的概率为w_i，再由第i个高斯分布$N(\mathrm{x}, \mu_i, \Sigma_i)$产生出这个样本数据x。高斯混合模型可以逼近任何一个连续的概率分布，因此它可以看做是连续性概率分布的万能逼近器。之所有要保证权重的和为1，是因为概率密度函数必须满足在$(-\infty, +\infty)$内的积分值为1。

指定高斯分布的个数，给定一组训练样本，可以通过期望最大化EM算法确定高斯混合模型的参数。每次迭代时，在E步计算期望值，在M步最大化期望值，如此循环交替。

EM算法

EM算法是一种迭代法，其目标是求解似然函数或后验概率的极值，而样本中具有无法观测的隐含变量。因为隐变量的存在，我们无法直接通过最大化似然函数来确定参数的值。可以采用一种策略，构造出对数似然函数的一个下界函数，这个函数不含有隐变量，然后优化这个下界。不断的提高这个下界，使原问题达到最优解，这就是EM算法所采用的思路。算法的构造依赖于Jensen不等式。

算法在实现时首先随机初始化参数 的值，接下来循环迭代，每次迭代时分为两步：

E步，基于当前的参数估计值 ，计算在给定x时对z的条件概率的数学期望：

M步，求解如下极值问题，更新 的值：

$$\theta = \arg\max_{\theta} \sum_{i} \sum_{z_i} Q_i(z_i) \log \frac{p(\mathrm{x}_i, z_i; \theta)}{Q_i(z_i)}$$

实现Q_i时可以按照下面的公式计算：

$$Q_i(z_i) = \frac{p(x_i, z_i; \theta)}{\sum_{z} p(x_i, z; \theta)}$$

迭代终止的判定规则是相邻两次函数值之差小于指定阈值。需要注意的是，EM算法只能保证收敛到局部极小值。