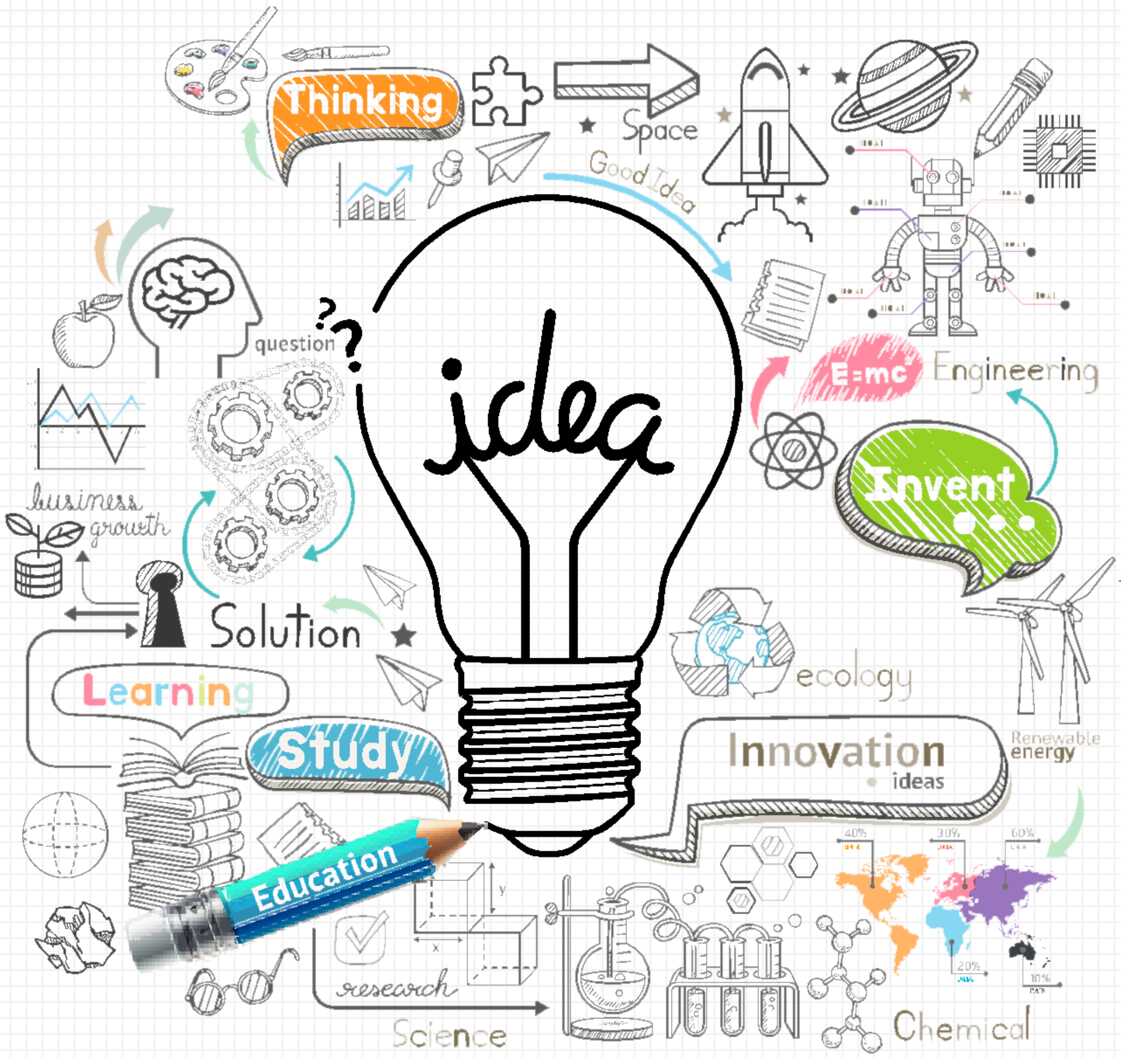
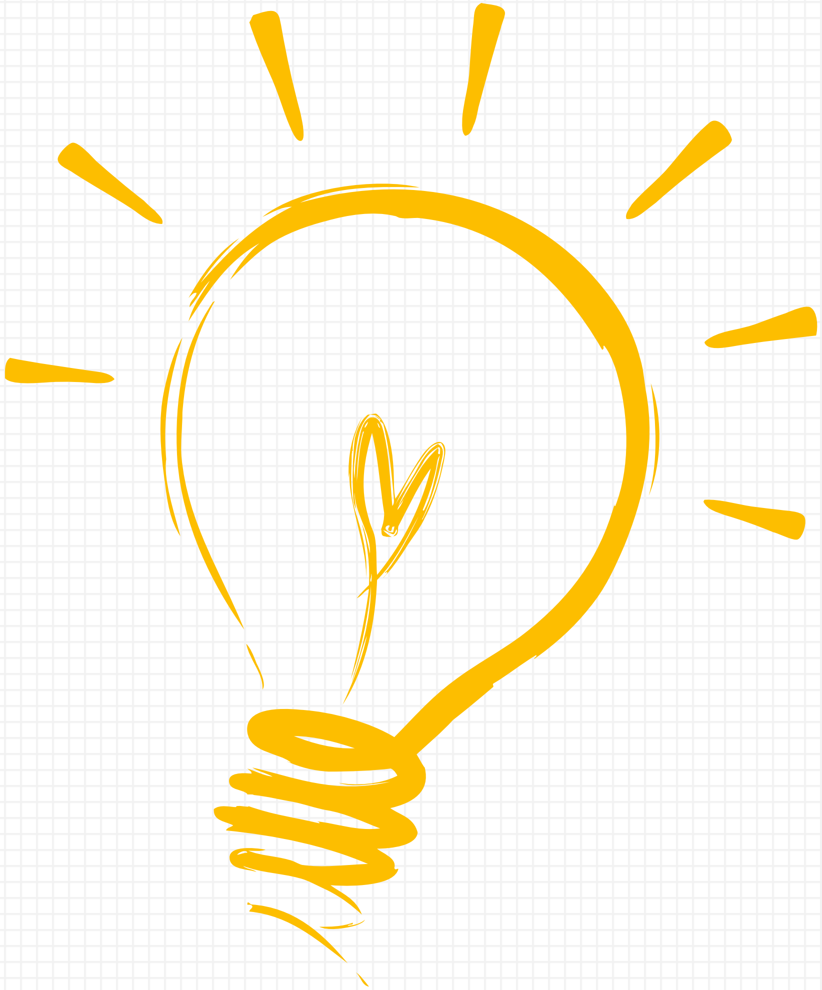
QQ：1366420642



数据挖掘算法

AdaBoost



CONTENTS

QQ：1366420642

di yi zhang jie

第一章节

算法背景

1.1 前置知识

分类器：ID3决策树、C4.5决策树、CART。

https://www.bilibili.com/video/BV1kg411u7RP

https://www.bilibili.com/video/BV18U4y1E7jQ

传统机器学习算法 (例如：决策树，人工神经网络，支持向量机，朴素贝叶斯等) 的目标都是寻找**一个**最优分类器尽可能的将训练数据分开。

集成学习 (Ensemble Learning) 算法的基本思想就是将**多个**分类器组合，从而实现一个预测效果更好的集成分类器。

集成算法可以说从一方面验证了中国的一句老话：三个臭皮匠，赛过诸葛亮。

1.2 集成算法

集成算法大致可以分为：Bagging，Boosting 和 Stacking 等类型。

Bagging(Boostrap Aggregating)：并行训练多个弱学习器，对于分类问题，采用投票的方法，得票最多子模型的分类类别为最终的类别；对于回归问题，采用简单的平均方法得到预测值。经典例子为随机森林算法。

Boosting：迭代生成弱学习器，并将其加入到当前学习分类器。加入的过程中，通常根据它们的分类准确率给予不同的权重。加和弱学习器之后，数据通常会被重新加权，来强化对之前分类错误数据点的分类。经典例子为**AdaBoost（Adaptive Boost）算法**。

Stacking：

https://www.cnblogs.com/geeksongs/p/15416820.html

https://www.cnblogs.com/huangyc/p/9975183.html

https://leovan.me/cn/2018/12/ensemble-learning/

1.3 前向分步算法

刚刚我们说过，Boosting算法通过将迭代生成弱学习器，并将产生的弱学习器与当前已有的学习器加和，最终产生一个强学习器。

假设 Dataset T = {(x_i, y_i) … …}，$i = 1$，2，3 … … N, $y_i = \{-1, +1\}$。

强学习器：

$$F_m(x) = F_{m-1}(x) + \alpha_m G_m(x)$$

$G_m(x)$是第m次迭代产生的弱学习器，比如ID3决策树等。 $G_m(x) = \{-1, +1\}$，最后将$\mathrm{sign}(F_m(x))$作为x的输出类别。

损失函数：

$$L(x, y) = L(F_{m-1}(x) + \alpha_m G_m(x), y)$$

$\forall i, j$，$G_i(x)$、 $G_j(x)$是独立的弱学习器. 因此

$$\min L(F_{m-1}(x) + \alpha_m G_m(x), y) \Leftrightarrow \min L(G_m(x), y)$$

QQ：1366420642

di yi zhang jie

第二章节

算法原理

2.1 算法推导

Dataset T = $\{(x_i, y_i)\ldots\ldots\}$ ， $i = 1, 2, 3\ldots\ldots N$, $y_i = \{-1, +1\}$ ， $\boldsymbol{w_i = \frac{1}{N}}$ 。

输出$F_m(x) = F_{m-1}(x) + \alpha_m G_m(x)$。

1.为什么有$\boldsymbol{w_i}$?

目的是体现出样本的重要性。举个例子，假如我们在第K次得到的学习器，在某个样本上分类结果出现错误，那么我们自然的想法是，第K+1次产生的弱学习器应该更加关注这个样本，使其尽可能的划分正确。

2.为什么有α_m?

一个直观的想法，比较菜的学习器，权重相应的小一点，好的学习器，权重大一些。

2.3 算法推导

可以知道学习器权重、样本的权重均和学习器的错误率有关，而错误率具体表现在损失函数值上，因此**我们可以从损失函数推导出样本权重W和学习器权重α 的推导公式。**

损失函数取

$$Loss = \sum_{i}^{N} \exp(-y_i F_m(x_i))$$

当$y_i = +1$时，$F_m(x_i)$为一个很大的正数，$\exp(-y_i F_m(x_i))$则会趋于0。那么大多数$G_m(x)$的输出结果应该是+1，这表明大多数弱学习器认为 x_i 属于+1类。反之，$\exp(-y_i F_m(x_i))$ 会非常大，这表明此时预测结果不佳。$y_i = -1$时，同理可得相似结果。

2.3 算法推导

$$F_m(x) = F_{m-1}(x) + \alpha_m G_m(x)$$

$$Loss = \sum_{i}^{N} \exp\{-y_i[F_{m-1}(x_i) + \alpha_m G_m(x_i)]\}$$

$$Loss = \sum_{i}^{N} \{\exp[-y_i F_{m-1}(x_i)] * \exp[-y_i \alpha_m G_m(x_i)]\}$$

$$Loss = \sum_{i}^{N} \{W_{mi} * \exp[-y_i \alpha_m G_m(x_i)]\}$$

$$W_{mi} = \exp[-y_i F_{m-1}(x_i)]$$

2.3 算法推导

又$y_i = \pm 1$、 $G_m(x_i) = \pm 1$，所以：

$$Loss = \sum_{i}^{N} \{W_{mi} * \exp[-y_i \alpha_m G_m(x_i)]\}$$

$$Loss = \sum_{y_i = G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\} + \sum_{y_i \neq G_m(x_i)}^{N} \{W_{mi} * \exp(\alpha_m)\}$$

$$Loss = \sum_{y_i = G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\} + \sum_{y_i \neq G_m(x_i)}^{N} \{W_{mi} * \exp(\alpha_m)\} + \sum_{y_i \neq G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\} - \sum_{y_i \neq G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\}$$

QQ：1366420642

2.3 算法推导

$$Loss = \sum_{y_i=G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\} + \sum_{y_i\neq G_m(x_i)}^{N} \{W_{mi} * \exp(\alpha_m)\} + \sum_{y_i\neq G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\} - \sum_{y_i\neq G_m(x_i)}^{N} \{W_{mi} * \exp(-\alpha_m)\}$$

$$Loss = \sum_{i}^{N} \{W_{mi} * \exp(-\alpha_m)\} + (e^{\alpha_m} - e^{-\alpha_m}) \sum_{y_i\neq G_m(x_i)}^{N} W_{mi}$$

令$\partial Loss / \partial \alpha_m = 0$

$$\partial Loss / \partial \alpha_m = -e^{-\alpha_m} \sum_{i}^{N} W_{mi} + (e^{\alpha_m} + e^{-\alpha_m}) \sum_{y_i\neq G_m(x_i)}^{N} W_{mi}$$

$$\frac{e^{-\alpha_m}}{e^{\alpha_m} + e^{-\alpha_m}} = \frac{\sum_{y_i\neq G_m(x_i)}^{N} W_{mi}}{\sum_{i}^{N} W_{mi}} = \frac{\sum_{i}^{N} W_{mi} * I(y_i \neq G_m(x_i))}{\sum_{i}^{N} W_{mi}} = e_m$$

2.3 算法推导

$$\frac{e^{-\alpha_m}}{e^{\alpha_m}+e^{-\alpha_m}}=\frac{\sum_{y_i\neq G_m(x_i)}^{N} W_{mi}}{\sum_i^N W_{mi}}=\frac{\sum_i^N W_{mi}*I(y_i\neq G_m(x_i))}{\sum_i^N W_{mi}}=e_m$$

$$\alpha_m=\frac{1}{2}ln\frac{1-e_m}{e_m}$$

而e_m就可以看做弱学习器G_m在数据集上的**加权**错误率。自然的，W_{mi}就是我们前边提到的，样本的权重。对于W_{mi}的求解：

$$Loss=\sum_i^N \exp\{-y_i[F_{m-1}(x_i)+\alpha_m G_m(x_i)]\}=\sum_i^N\{\exp[-y_iF_{m-1}(x)] * \exp[-y_i\alpha_m G_m(x_i)]\}$$

$$Loss=\sum_i^N\{W_{mi}*\exp[-y_i\alpha_m G_m(x_i)]\},W_{mi}=\exp[-y_iF_{m-1}(x)]$$

2.3 算法推导

$$Loss = \sum_{i}^{N}\{W_{mi} * \exp[-y_i\alpha_m G_m(x_i)]\}, W_{mi} = \exp[-y_i F_{m-1}(x)]$$

$$F_{m-1}(x) = F_{m-2}(x) + \alpha_{m-1}G_{m-1}(x)$$

$$W_{mi} = \exp[-y_i F_{m-1}(x)] = \exp[-y_i(F_{m-2}(x) + \alpha_{m-1}G_{m-1}(x))] = W_{m-1,i} * \exp[-y_i\alpha_{m-1}G_{m-1}(x_i)]$$

得到$\boldsymbol{W_{mi}}$的递推公式:

$$\boldsymbol{W_{mi}} = W_{m-1,i} * \exp[-y_i\alpha_{m-1}G_{m-1}(x_i)]$$

$$\boldsymbol{\alpha_m = \frac{1}{2} ln\frac{1-e_m}{e_m}, e_m = \frac{\sum_i^N W_{mi} * I(y_i \neq G_m(x_i))}{\sum_i^N W_{mi}}}$$

2.3 算法流程

Input : Dataset $T=\{(x_i, y_i)\ldots\ldots\}$, $i=1, 2, 3\ldots\ldots N, y_i=\{-1,+1\}$

Initialize $\boldsymbol{w}_{\mathbf{0},\boldsymbol{i}}=\frac{\mathbf{1}}{\boldsymbol{N}}$, $F_0(x)=0$

for $m=1:M$

根据数据集 得到$G_m(x)$

根据$\boldsymbol{w}_{m-1,\boldsymbol{i}}$得到 $\boldsymbol{w}_{m,\boldsymbol{i}}$，既而得到$\boldsymbol{e}_{\boldsymbol{m}}$

根据$\boldsymbol{e}_{\boldsymbol{m}}$得到$\alpha_m$

得到当前强学习器$F_m(x)=F_{m-1}(x)+\alpha_m G_m(x)$

end

Output : 强学习器$F_m(x)$，各弱学习器$G_m(x)$，弱学习器权重α_m，样本权重$\boldsymbol{w}_{m,\boldsymbol{i}}$

QQ：1366420642

di san zhang jie

第三章节

算法分析

3.1 算法分析

Adaboost的主要优点有:

- Adaboost作为分类器时，分类精度很高
- 在Adaboost的框架下，可以使用各种回归分类模型来构建弱学习器，非常灵活。
- 作为简单的二元分类器时，构造简单，结果可理解。
- 不容易发生过拟合

Adaboost的主要缺点有:

- 对异常样本敏感，异常样本在迭代中可能会获得较高的权重，影响最终的强学习器的预测准确性。
- 不能并行。

QQ：1366420642

di si zhang jie

第四章节

算法拓展

4.1 算法拓展

多分类、代码实现时的性能改进、正则化等。

Zhu, H. Zou, S. Rosset, T. Hastie, “Multi-class AdaBoost”, 2009.

QQ：1366420642

di wu zhang jie

第五章节

案例实操

5.1 案例实操

需要做的准备是：

首先要有Python环境，最好可以安装上Pycharm。

然后安装一下必要的库（以后也需要的）：如numpy，pandas，xlrd，scikit-learn库等

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

课程代码、PPT：

https://github.com/CHENHUI-X/My-lecture-slides-and-code

THANKS

谢谢观看