

目录 | Contents

- 1、精确率
- 2、召回率
- 3、F1-Score

评价指标

任何的AI系统需要有一个合理的评价指标。有一个评价指标我相信大家再熟悉不过了，就是准确率(accuracy)。虽然准确率用处很广泛，但其实也有它不太适合的地方。

准确率

假设我们有一个工业上检测瑕疵的装置，合格品的比率是99%，也就是说生产100件产品有99件是合格的，只有一件是不合格的。

准确率 ACC	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$	分类模型所有判断正确的结果占总观测值的比重
------------	---	-----------------------

准确率

如果这个装置检测出来是100件都合格，那么对于不合格品来说实际是不合格但是被误判断为合格了，我们按照准确率（ACC）这个公式来衡量的话我们的准确率就达到了99%，但是如果按照精确率（precision）来衡量的话我们的精确率是0（在实践中，我们优先将关注度高的作为正类），在这个案例中，显而易见的是不合格品被监测出来是我们更为关注的，因为一旦不合格品流向了市场，那对一家企业的口碑来说是非常受影响的。

真实值是positive, 模型认为是positive的数量 (True Positive=TP)

真实值是positive, 模型认为是negative的数量 (False Negative=FN)

真实值是negative, 模型认为是positive的数量 (False Positive=FP)

真实值是negative, 模型认为是negative的数量 (True Negative=TN)

令TP、TN、FP、FN分别表示其对应的样本数, 则 $TP + TN + FP + FN = \text{样本总数}$, 分类的“混淆矩阵”:

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

准确率 ACC	$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$	分类模型所有判断正确的结果占总观测值的比重
精确率 PPV	$\text{Precision} = \frac{TP}{TP + FP}$	在模型预测是Positive的所有结果中，模型预测对的比重
灵敏度 TPR	$\text{Sensitivity} = \text{Recall} = \frac{TP}{TP + FN}$	在真实值是Positive的所有结果中，模型预测对的比重



F1值是用来衡量二分类模型精确度的一种指标。它同时兼顾了分类模型的精确率和召回率。F1分数可以看作是模型精确率和召回率的一种加权平均，它的最大值是1，最小值是0。越接近1代表了模型的精确度越高，反之越差。

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \leftarrow$$

案例

这个图片到底是不是猫：🐱	真实值 (1是, 0不是)	预测结果	预测对了吗?		实际：正	实际：负
不是猫	0	1	✗	预测：正	3	1
不是猫	0	0	✓	预测：负	2	4
不是猫	0	0	✓			
不是猫	0	0	✓			
不是猫	0	0	✓			
是猫	1	1	✓			
是猫	1	1	✓			
是猫	1	1	✓			
是猫	1	0	✗			
是猫	1	0	✗			

案例

T : True、
F : False、
P : Positive、正、1、是猫
N : Negative、负、0、不是猫

		正	负
正	真实：是猫 预测：是猫 	真实：不是猫 预测：是猫 	
负	真实：是猫 预测：不是猫 	真实：不是猫 预测：不是猫 	

TP (True Positive) points to the top-left cell (Actual: Cat, Predicted: Cat).
FP (False Positive) points to the top-right cell (Actual: Not Cat, Predicted: Cat).
FN (False Negative) points to the bottom-left cell (Actual: Cat, Predicted: Not Cat).
TN (True Negative) points to the bottom-right cell (Actual: Not Cat, Predicted: Not Cat).

案例

这个图片到底是不是猫：🐱	真实值 (1是, 0不是)	预测结果	预测对了吗?
不是猫	0	1	✗
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	0	✗
是猫	1	0	✗

准确率

(accuracy)

就是总样本中预测对了多少

预测对了7个
总样本有10个

案例

这个图片到底是不是猫：🐱	真实值 (1是, 0不是)	预测结果	预测对了吗？
不是猫	0	1	✗
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	0	✗
是猫	1	0	✗

精确率

(查准率、precision)

预测为正的样本中实际为正的有多少

其中真的为正的3个

预测为正的4个

精确率： $3/4 = 75\%$

案例

这个图片到底是不是猫：🐱	真实值 (1是, 0不是)	预测结果	预测对了吗？
不是猫	0	1	✗
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
不是猫	0	0	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	1	✓
是猫	1	0	✗
是猫	1	0	✗

召回率

(查全率、recall)

实际为正的样本中有多少被预测为正了

其中预测为正的3个

实际为正的5个

召回率：3/5 = 60%

Sklearn对应库

sklearn当中提供了大量的类来帮助我们了解和使用混淆矩阵。

类	含义
sklearn.metrics.confusion_matrix	混淆矩阵
sklearn.metrics.accuracy_score	准确率accuracy
sklearn.metrics.precision_score	精确度precision
sklearn.metrics.recall_score	召回率recall
sklearn.metrics.precision_recall_curve	精确度-召回率平衡曲线，可以展示不同阈值下的精确度和召回率如何变化。
sklearn.metrics.f1_score	F1 measure

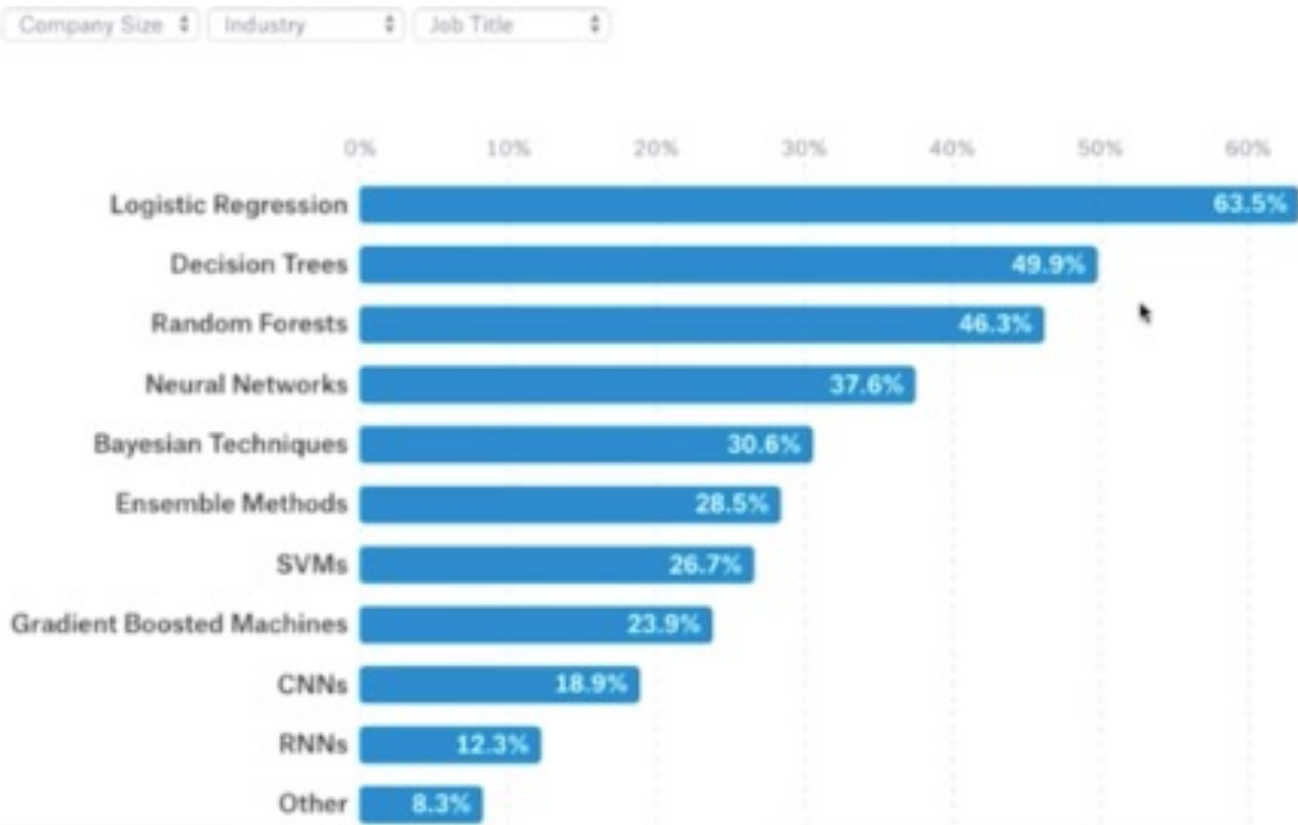
目录 | Contents

- 1、机器学习概述
- 2、分类算法-逻辑回归

逻辑回归的地位

What data science methods are used at work?

Logistic regression is the most commonly reported data science method used at work for all industries *except* **Military and Security** where Neural Networks are used slightly more frequently.



从线性到逻辑回归

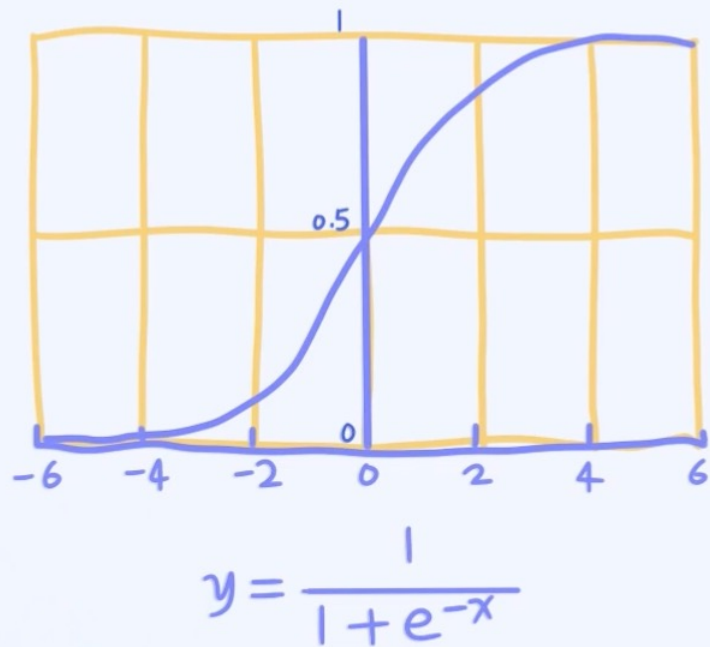
我们使用的 $w^T x + b$ 方式来描述，则它的值域是负无穷到正无穷区间，显然这不符合概率的定义。



可不可以把线性回归 $w^T x + b$
做些改造使得值域映射到 $(0, 1)$ 区间

把正无穷到负无穷的值域映射到 $(0, 1)$ 之间呢？
那这样一来，就可以得到合理的概率值了！
答案就是使用逻辑函数。

逻辑函数 (sigmoid函数)



定义域(x): $(x): (-\infty, +\infty)$

值域(y): $(y): y \in (0, 1)$

对于特征向量 x 和二分类标签 y , 我们可以定义如下的条件概率:

$$p(y = 1|x, w, b) = \frac{1}{1 + e^{-(w^T x + b)}}$$

$$p(y = 0|x, w, b) = \frac{e^{-(w^T x + b)}}{1 + e^{-(w^T x + b)}}$$

两个式子可以合并成:

$$p(y|x) = p(y = 1|x, w, b)^y [1 - p(y = 1|x, w, b)]^{1-y}$$

合并之后的式子不难理解。当 $y=1$ 时, 后面那一项不起任何作用也就变成了第一个条件概率; 当 $y=0$ 时, 前面那一项不起作用, 也就等同于第二个条件概率。

逻辑回归

逻辑回归 Logistic Regression

逻辑回归：解决分类问题

回归问题怎么解决分类问题？

将样本的特征和样本发生的概率联系起来，概率是一个数

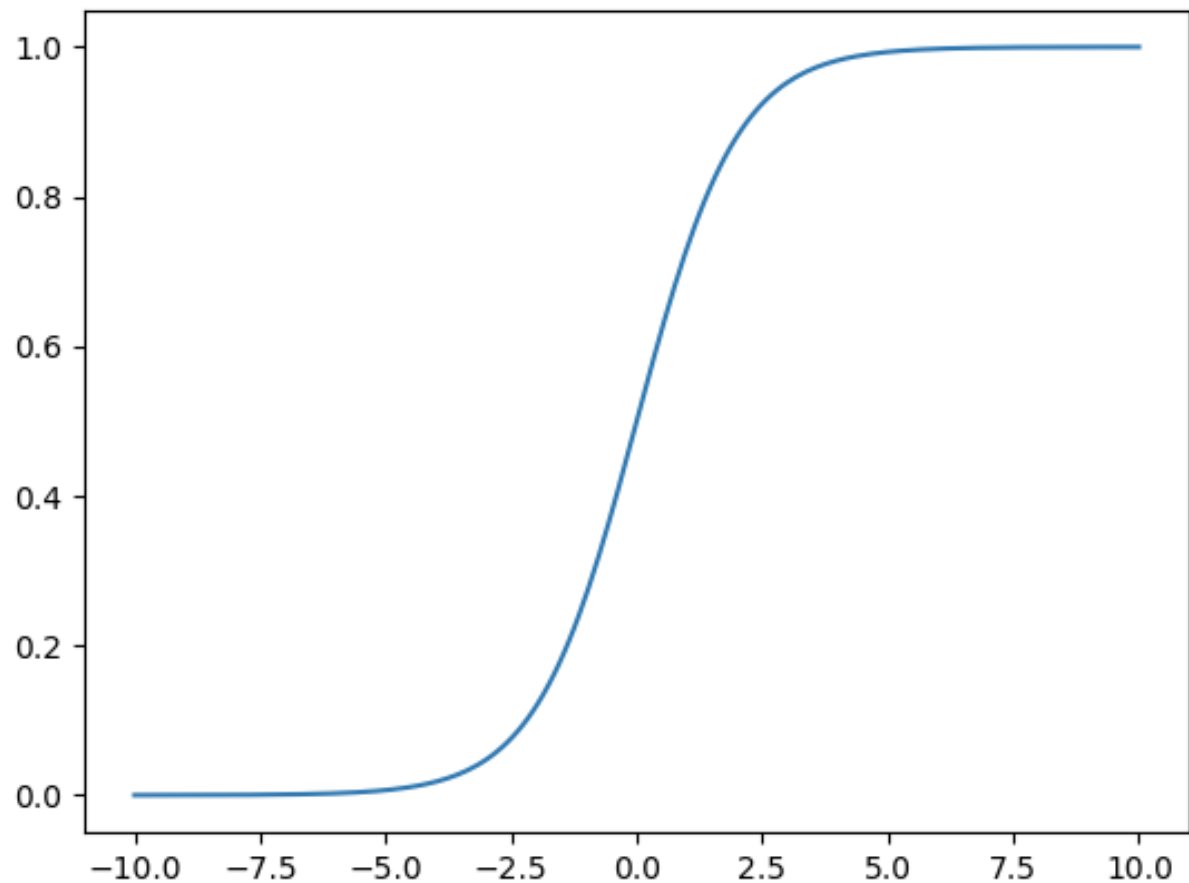
Sigmoid函数

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def sigmoid(t):
    return 1 / (1+np.exp(-t))
```

```
x = np.linspace(-10,10,200)
y=sigmoid(x)
plt.plot(x,y)
plt.show()
```

Sigmoid函数



定义域x在（-无穷，+无穷）之间
值域在（0，1）之间

t > 0 时, p > 0.5

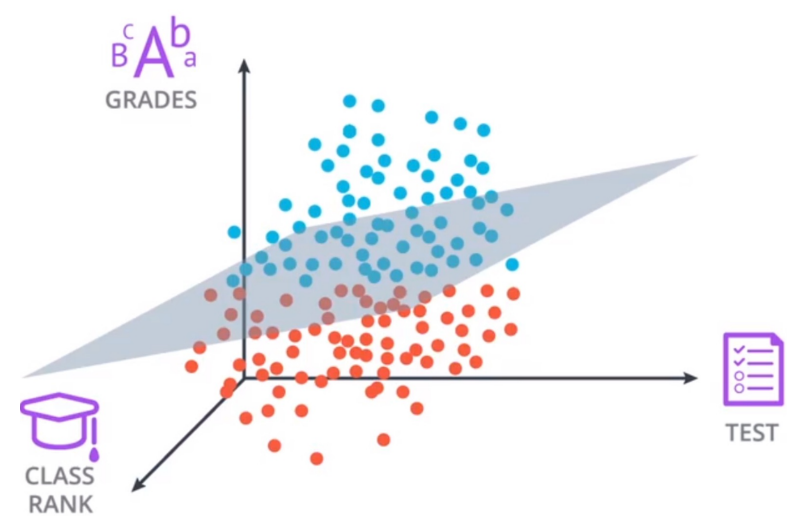
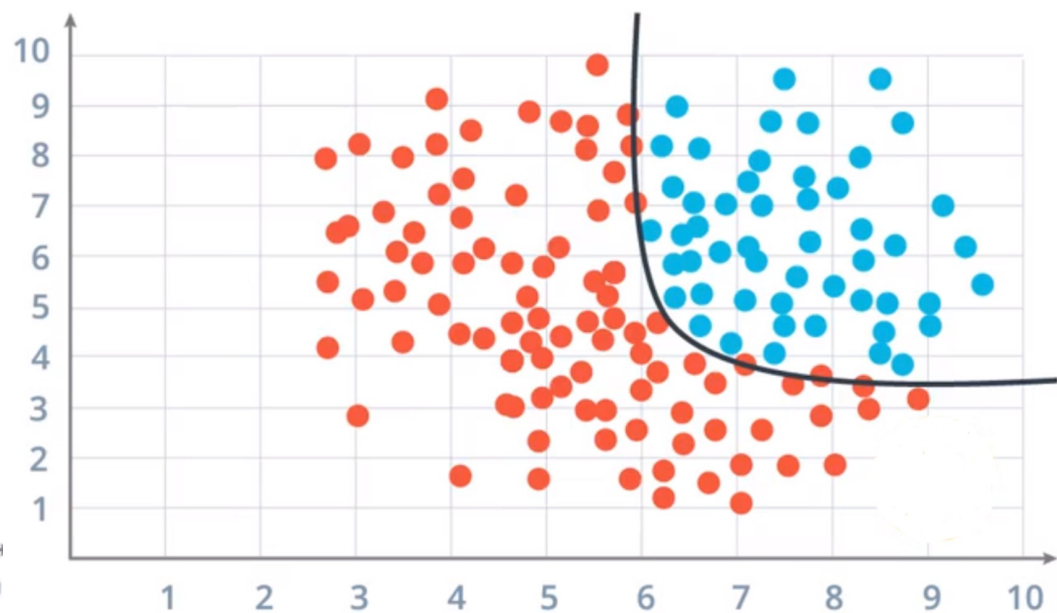
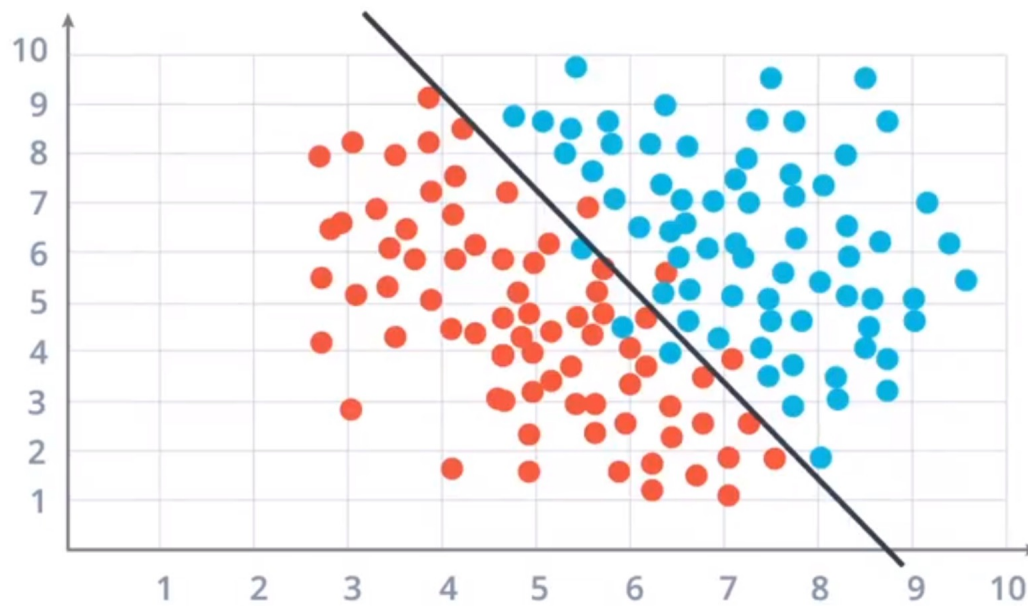
t < 0 时, p < 0.5

最后我们就看结果：

$$\hat{p} = \sigma(\theta^T \cdot x_b) = \frac{1}{1 + e^{-\theta^T \cdot x_b}}$$

$$\hat{y} = \begin{cases} 1, & \hat{p} \geq 0.5 \\ 0, & \hat{p} \leq 0.5 \end{cases}$$

线性分类vs非线性



逻辑回归是否是线性分类器

决策边界上的样本被划分为正负样本的概率相等

给定任意点 x , 我们有

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

$$\Leftrightarrow \quad 1 = e^{-(w^T x + b)} \quad \left. \begin{array}{l} \log \\ \downarrow \end{array} \right\}$$

$$\Leftrightarrow \quad 0 = -(w^T x + b)$$

$$\Leftrightarrow \quad w^T x + b = 0 \quad \Leftarrow \text{线性}$$

梯度下降

参看notebook梯度下降

柏拉图有一天问老师苏格拉底什么是爱情？苏格拉底叫他到麦田走一次，摘一颗最大的麦穗回来，不许回头，只可摘一次。柏拉图空着手出来了，他的理由是，看见不错的，却不知道是不是最好的，一次次侥幸，走到尽头时，才发现还不如前面的，于是放弃。苏格拉底告诉他：“这就是爱情。”这故事让我们明白了一个道理，因为生命的一些不确定性，所以全局最优解是很难寻找到的，或者说根本就不存在，我们应该设置一些限定条件，然后在这个范围内寻找最优解，也就是局部最优解——有所斩获总比空手而归强，哪怕这种斩获只是一次有趣的经历。



机器学习中存在凸函数，但是很大一部分的机器学习还是存在右边（非凸函数）这样的情况（比如深度学习）

梯度下降法的缺点

随机初始化 w^1, b^1

for $t = 1, 2, \dots$ 直到收敛

$$w^{t+1} = w^t - \eta_t \sum_{i=1}^n (\sigma(w^{tT} x_i + b^t) - y_i) x_i$$

$$b^{t+1} = b^t - \eta_t \sum_{i=1}^n (\sigma(w^{tT} x_i + b^t) - y_i)$$

当样本很多的时候，每一次迭代所花费的时间成本是很高的。举个例子，当数据集里有一百万个样本的时候，每一次的参数更新就需要循环所有的样本并把它们的梯度做累加。

随机梯度下降

随机梯度下降法

随机梯度下降法(SGD)可以看作是梯度下降法的极端的情况。在梯度下降法里，每次的参数更新依赖于所有的样本。然而，在随机梯度下降法里，每一次的迭代不再依赖于所有样本的梯度之和，而是仅仅依赖于其中一个样本的梯度。所以这种方法的优势很明显，通过很“便宜”的方式获得梯度，并频繁的对参数做迭代更新，这有助于在更短的时间内得到收敛结果。

随机梯度下降法

随机初始化 w^1, b^1

for $t = 1, \dots$ // 每一个外层循环扫描所有的样本

shuffle $((x_i, y_i))_{i=1}^n$ // 把样本的顺序随机化

for $i = 1, \dots, n$ // 循环每一个样本

// 基于单个样本的梯度来更新参数

$$w^{t+1} = w^t - \eta_t (\sigma(w^T x_i + b) - y_i) x_i$$

$$b^{t+1} = b^t - \eta_t (\sigma(w^T x_i + b) - y_i)$$



虽然SGD用很低的成本可以更新到模型的参数,但也有自身的缺点。你觉得SGD的缺点是什么?

- ☒ A. 由于梯度的计算仅依赖于一个样本,计算出的结果包含较大的噪声
- ☐ B. SGD的收敛效率低
- ☐ C. SGD的准确率下降
- ☐ D. 所有都符合

SGD的一大缺点就是计算出的梯度包含很多噪音。除此之外,其他两项都不属于SGD的缺点。实际上,SGD的收敛效率通常是更高的,而且有些时候SGD的最后找出来的解更优质。

由于在随机梯度下降法中，我们用一个样本的梯度来代替所有样本的梯度之和，计算出来的结果包含大量噪声，并且不太稳定。为了部分解决这个问题，我们通常会把学习率设置为较小的值，这样可以有效削弱梯度计算中带来的不稳定性。

相比梯度下降法，当我们使用随机梯度下降法的时候可以看到每一次迭代之后的目标函数或者损失函数会有一些波动性。有时候的更新会带来目标值的提升，其他时候的更新可能反而让目标值变得更差。但只要实现细节合理，大的趋势是沿着好的方向而发展的。

比较不同的算法

对于各类梯度下降法做一个总结:

-

在工业界中，最常用的方法为小批量梯度下降法

- 小批量梯度下降法也有助于更好地利用GPU的计算能力

- 小批量梯度下降法折中了梯度下降法和随机梯度下降法各自的优缺点，更好地解决梯度噪声的问题，更新更加稳定。

- 随机梯度下降法或者小批量梯度下降法有助于解决鞍点(saddle point)的问题。

Baseline

逻辑回归是很靠谱的基准模型（baseline），搭建任何分类模型时都可以考虑使用逻辑回归，之后再逐步尝试更复杂的模型。

基准(baseline)在建模过程中非常重要。简单来讲，在设计模型的阶段，首先试图通过简单的方法来快速把系统搭起来，之后逐步把模块细化，从而不断得到更好的解决方案。对于分类任务，逻辑回归模型可以称得上是最好的基准，也是比较靠谱的基准。在实际工作中，切忌一上来就使用复杂的模型，实际上，刚步入AI行业的人士经常会犯这个问题。一上来就想使用深度学习来做训练，比如提出要使用BERT，或者深度增强学习。。。这些都不是正确的做法。