機器學習大神最常用的 5 個回歸損失函數，你知道幾個？

2018/06/22

大數據文摘

【我們為什麼會挑選這篇文章】「機器學習中所有的算法都需要最大化或最小化一個函數，這個函數被稱為「目標函數」，其中，我們一般把最小化的一類函數，稱為「損失函數」。」本篇文章作者手把手教你找到損失函數達到極小值的點，解決機器學習優化中最重要的部分。（責任編輯：鄧天心）

編譯：Apricock、睡不著的 iris、JonyKai、錢天培

「損失函數」是機器學習優化中至關重要的一部分。L1、L2 損失函數相信大多數人都早已不陌生。那你了解 Huber 損失、Log-Cosh 損失、以及常用於計

算預測區間的分位數損失嗎？這些可都是機器學習大神最常用的回歸損失函數哦！

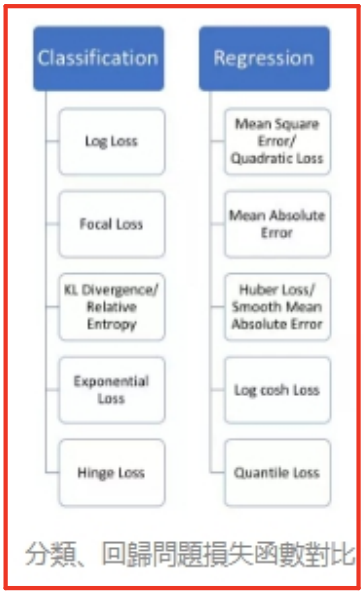
機器學習中所有的算法都需要最大化或最小化一個函數，這個函數被稱為「目標函數」。其中，我們一般把最小化的一類函數，稱為「損失函數」。它能根據預測結果，衡量出模型預測能力的好壞。

在實際應用中，選取損失函數會受到諸多因素的製約，比如是否有異常值、機器學習算法的選擇、梯度下降的時間複雜度、求導的難易程度以及預測值的置信度等等。因此，不存在一種損失函數適用於處理所有類型的數據。這篇文章就講介紹不同種類的損失函數以及它們的作用。

損失函數大致可分為兩類：分類問題的損失函數和回歸問題的損失函數。在這篇文章中，我將著重介紹回歸損失。

本文出現的代碼和圖表我們都妥妥保存在這兒了：

https://nbviewer.jupyter.org/github/groverpr/Machine-Learning/blob/master/notebooks/05_Loss_Functions.ipynb



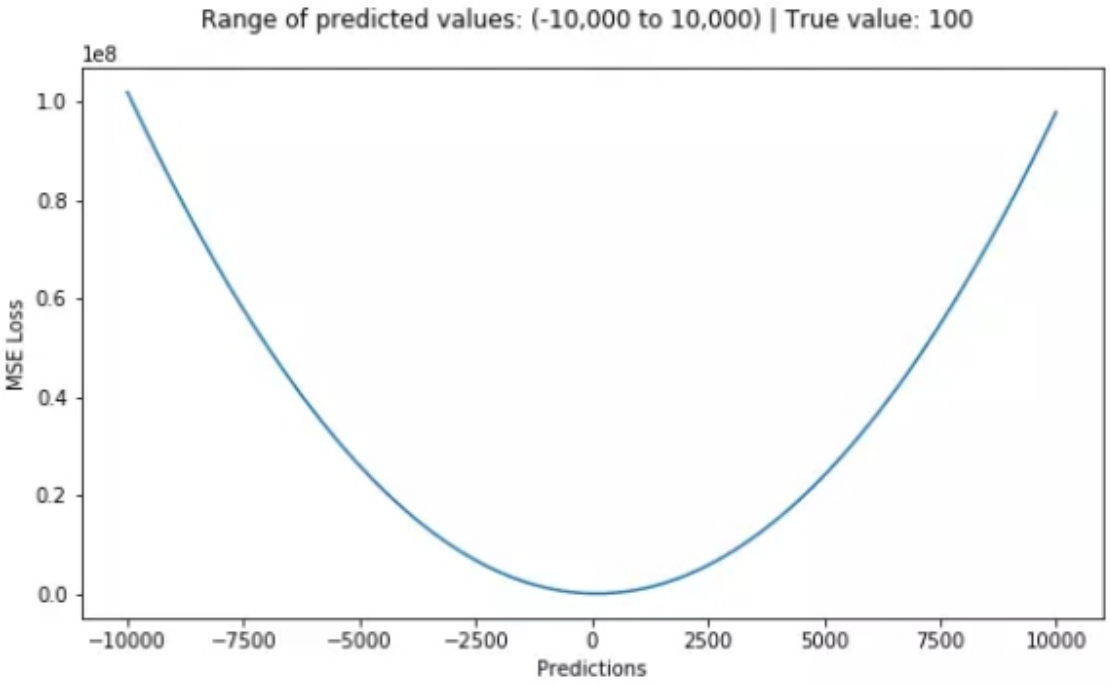
分類、回歸問題損失函數對比

均方誤差

$$MSE = \sum_{i=1}^{n} (y_i - y_i^p)^2$$

均方誤差 (MSE) 是最常用的回歸損失函數，計算方法是求預測值與真實值之間距離的平方和，公式如圖。

下圖是 MSE 函數的圖像，其中目標值是 100，預測值的範圍從 -10000 到 10000，Y 軸代表的 MSE 取值範圍是從 0 到正無窮，並且在預測值為 100 處達到最小。

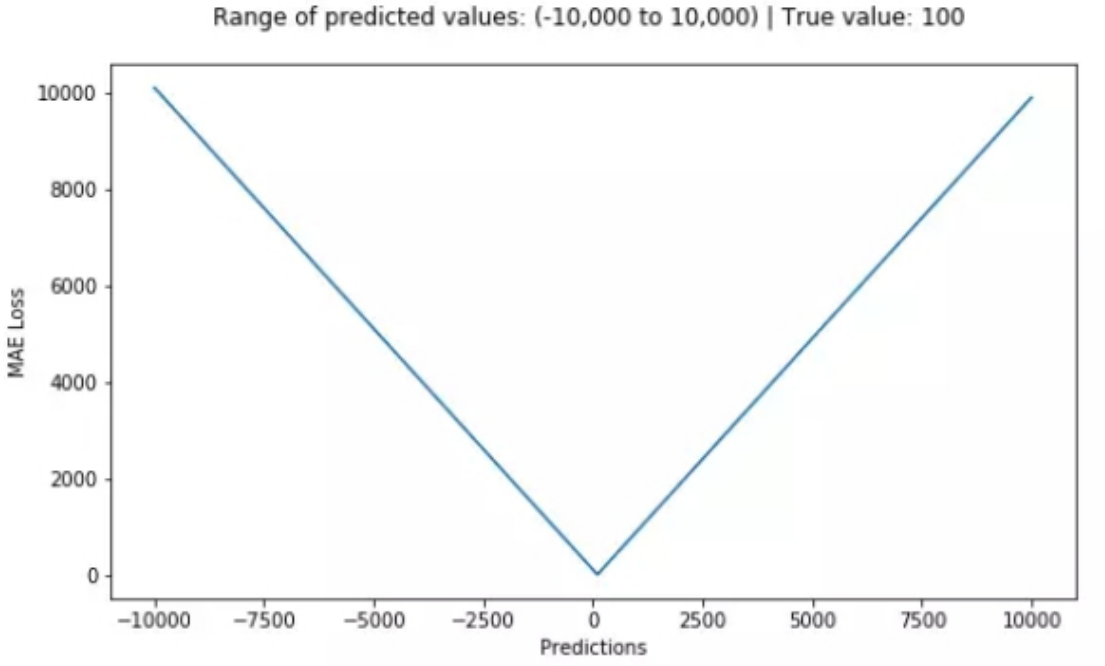


MSE損失（Y軸）-預測值（X軸）

平均絕對值誤差（也稱 L1 損失）

$$MAE = \sum_{i=1}^{n} |y_i - y_i^p|$$

平均絕對誤差（MAE）是另一種用於回歸模型的損失函數。MAE 是目標值和預測值之差的絕對值之和。其只衡量了預測值誤差的平均模長，而不考慮方向，取值範圍也是從 0 到正無窮（如果考慮方向，則是殘差/誤差的總和——平均偏差（MBE））。



MAE損失（Y軸）-預測值（X軸）

MSE（L2 損失）與 MAE（L1 損失）的比較

簡單來說，MSE 計算簡便，但 MAE 對異常點有更好的魯棒性。下面就來介紹導致二者差異的原因。

訓練一個機器學習模型時，我們的目標就是找到損失函數達到極小值的點。當預測值等於真實值時，這兩種函數都能達到最小。

下面是這兩種損失函數的 python 代碼。你可以自己編寫函數，也可以使用 sklearn 內置的函數。

```
# true: Array of true target variable
# pred: Array of predictions
def mse ( true , pred):
return np. sum (( true − pred)**2)
def mae ( true , pred):
return np. sum (np . abs ( true − pred))

# also available in sklearn
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

下面讓我們觀察 MAE 和 RMSE（即 MSE 的平方根，同 MAE 在同一量級中）在兩個例子中的計算結果。第一個例子中，預測值和真實值很接近，而且誤差的方差也較小。第二個例子中，因為存在一個異常點，而導致誤差非常大。

MAE vs. RMSE for cases with slight variance in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	-2	2	4
4	-0.5	0.5	0.25
5	1.5	1.5	2.25

MAE: 1　　RMSE: 1.22

MAE vs. RMSE for cases with outliers in data

ID	Error	|Error|	Error²
1	0	0	0
2	1	1	1
3	1	1	1
4	-2	2	4
5	15	15	225

outlier

MAE: 3.8　　RMSE: 6.79

左圖：誤差比較接近右圖：有一個誤差遠大於其他誤差

從圖中可以知道什麼？應當如何選擇損失函數？

MSE 對誤差取了平方（令 e=真實值-預測值），因此若 e>1，則 MSE 會進一步增大誤差。如果數據中存在異常點，那麼 e 值就會很大，而 e^2則會遠大於|e|。

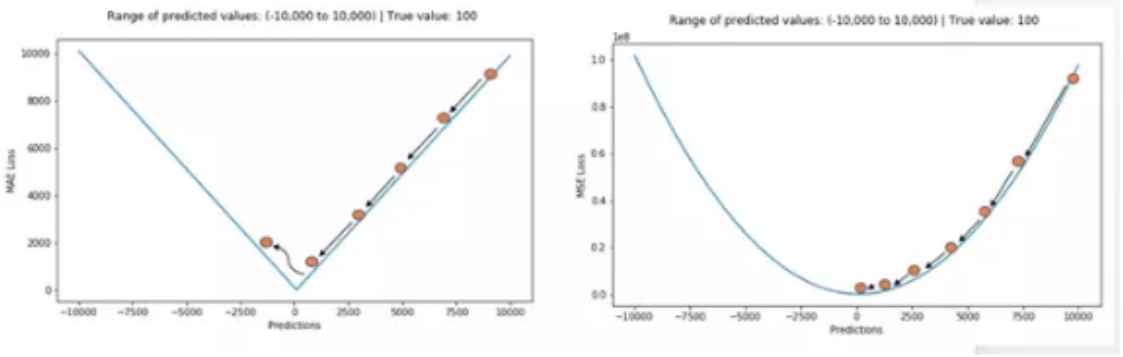
因此，相對於使用 MAE 計算損失，使用 MSE 的模型會賦予異常點更大的權重。在第二個例子中，用 RMSE 計算損失的模型會以犧牲了其他樣本的誤差為代價，朝著減小異常點誤差的方向更新。然而這就會降低模型的整體性能。

如果訓練數據被異常點所污染，那麼 MAE 損失就更好用（比如，在訓練數據中存在大量錯誤的反例和正例標記，但是在測試集中沒有這個問題）。

直觀上可以這樣理解：如果我們最小化 MSE 來對所有的樣本點只給出一個預測值，那麼這個值一定是所有目標值的平均值。但如果是最小化 MAE，那麼這個值，則會是所有樣本點目標值的中位數。眾所周知，對異常值而言，中位數比均值更加魯棒，因此 MAE 對於異常值也比 MSE 更穩定。

然而 MAE 存在一個嚴重的問題（特別是對於神經網絡）：更新的梯度始終相同，也就是說，即使對於很小的損失值，梯度也很大。這樣不利於模型的學習。為了解決這個缺陷，我們可以使用變化的學習率，在損失接近最小值時降低學習率。

而 MSE 在這種情況下的表現就很好，即便使用固定的學習率也可以有效收斂。MSE 損失的梯度隨損失增大而增大，而損失趨於 0 時則會減小。這使得在訓練結束時，使用 MSE 模型的結果會更精確。



根據不同情況選擇損失函數

如果異常點代表在商業中很重要的異常情況，並且需要被檢測出來，則應選用 MSE 損失函數。相反，如果只把異常值當作受損數據，則應選用 MAE 損失函數。

推薦大家讀一下這篇文章，文中比較了分別使用 L1、L2 損失的回歸模型在有無異常值時的表現。

文章網址：

http://rishy.github.io/ml/2015/07/28/l1-vs-l2-loss/

這裡 L1 損失和 L2 損失只是 MAE 和 MSE 的別稱。

總而言之，處理異常點時，L1 損失函數更穩定，但它的導數不連續，因此求解效率較低。L2 損失函數對異常點更敏感，但通過令其導數為 0，可以得到更穩定的封閉解。

二者兼有的問題是：在某些情況下，上述兩種損失函數都不能滿足需求。例如，若數據中 90% 的樣本對應的目標值為 150，剩下 10% 在 0 到 30 之間。那麼使用 MAE 作為損失函數的模型可能會忽視 10% 的異常點，而對所有樣本的預

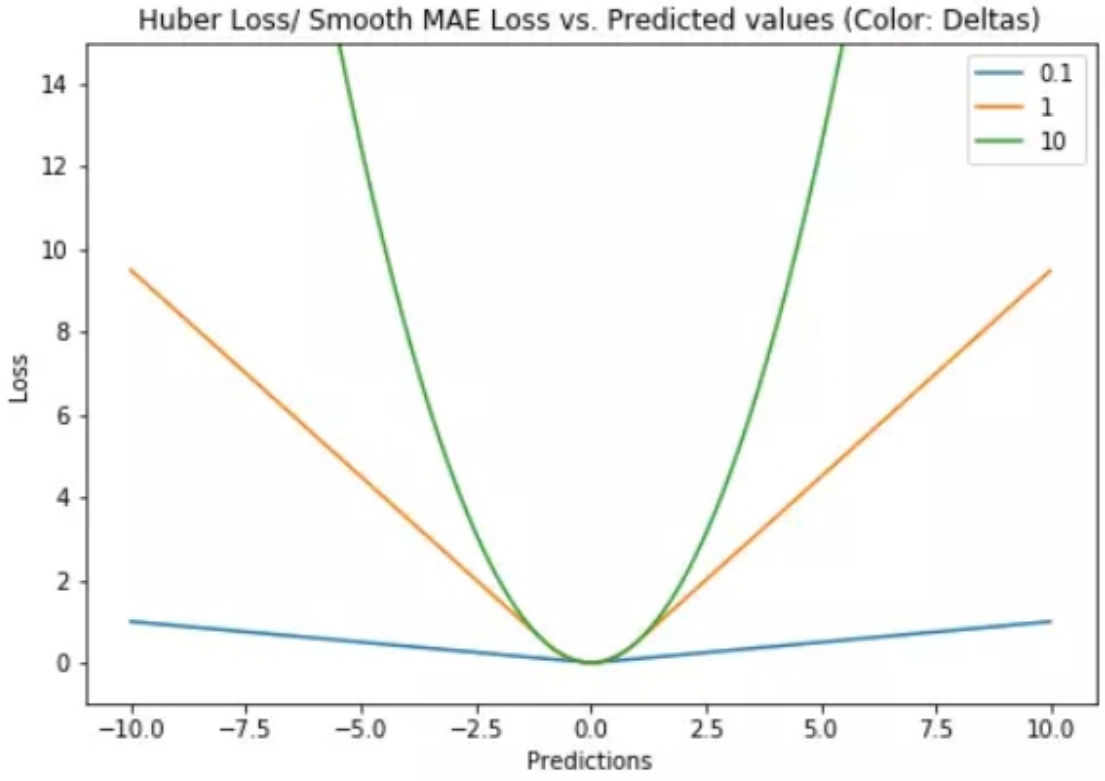
測值都為 150。

這是因為模型會按中位數來預測。而使用 MSE 的模型則會給出很多介於 0 到 30 的預測值，因為模型會向異常點偏移。上述兩種結果在許多商業場景中都是不可取的。

這些情況下應該怎麼辦呢？最簡單的辦法是對目標變量進行變換。而另一種辦法則是換一個損失函數，這就引出了下面要講的第三種損失函數，即 Huber 損失函數。

Huber 損失，平滑的平均絕對誤差
Huber 損失對數據中的異常點沒有平方誤差損失那麼敏感。它在 0 也可微分。本質上，Huber 損失是絕對誤差，只是在誤差很小時，就變為平方誤差。誤差降到多小時變為二次誤差由超參數 δ（delta）來控制。當 Huber 損失在 [0-δ,0+δ] 之間時，等價為 MSE，而在 [-∞,δ] 和 [δ,+∞] 時為 MAE。

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



Huber損失（Y軸）與預測值（X軸）圖示。真值取0

這裡超參數 delta 的選擇非常重要，因為這決定了你對與異常點的定義。當殘差大於 delta，應當採用 L1（對較大的異常值不那麼敏感）來最小化，而殘差小於超參數，則用 L2 來最小化。

為何要使用 Huber 損失？

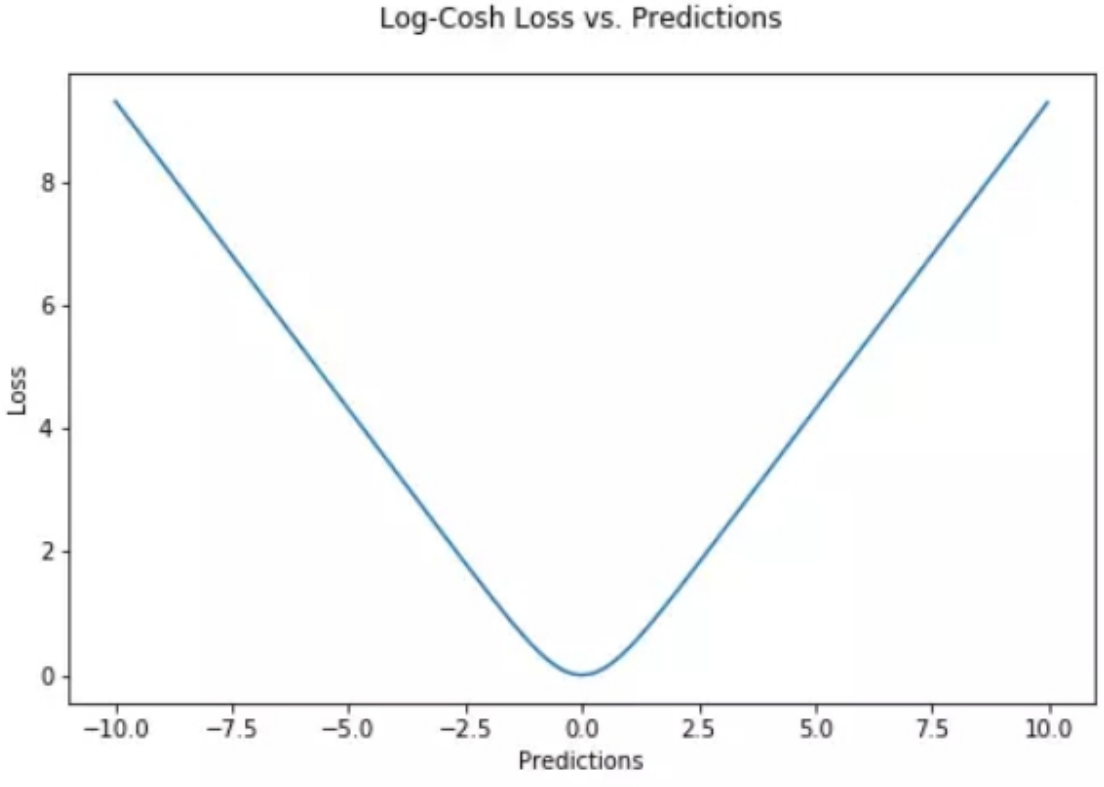
使用 MAE 訓練神經網絡最大的一個問題就是不變的大梯度，這可能導致在使用梯度下降快要結束時，錯過了最小點。而對於 MSE，梯度會隨著損失的減小而減小，使結果更加精確。

在這種情況下，Huber 損失就非常有用。它會由於梯度的減小而落在最小值附近。比起 MSE，它對異常點更加魯棒。因此，Huber 損失結合了 MSE 和 MAE 的優點。但是，Huber 損失的問題是我們可能需要不斷調整超參數 delta。

Log-Cosh 損失

Log-cosh 是另一種應用於回歸問題中的，且比 L2 更平滑的的損失函數。它的計算方式是預測誤差的雙曲餘弦的對數。

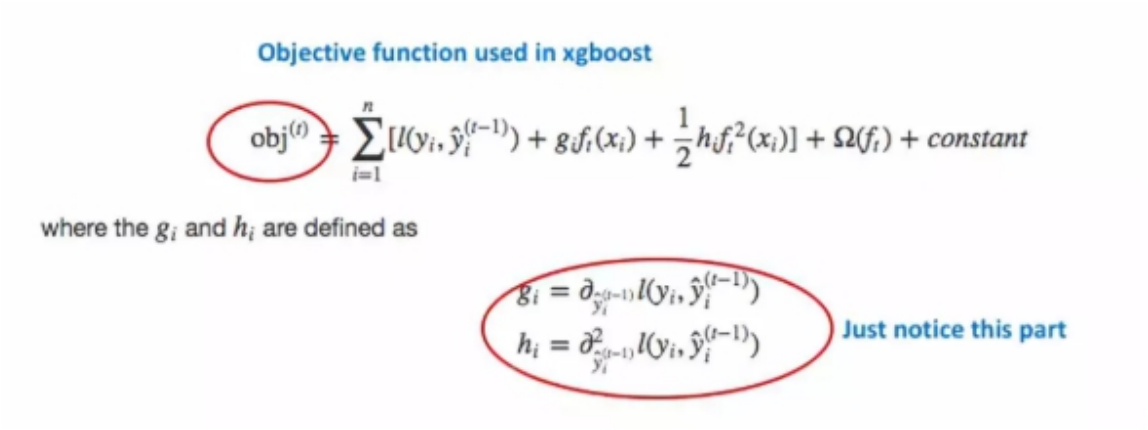
$$L(y, y^p) = \sum_{i=1}^{n} \log(\cosh(y_i^p - y_i))$$



Log-cosh損失（Y軸）與預測值（X軸）圖示。真值取0

優點：對於較小的 x，log(cosh(x)) 近似等於 (x^2)/2，對於較大的 x，近似等於 abs(x)-log(2)。這意味著'logcosh' 基本類似於均方誤差，但不易受到異常點的影響。它具有 Huber 損失所有的優點，但不同於 Huber 損失的是，Log-cosh 二階處處可微。

為什麼需要二階導數？許多機器學習模型如 XGBoost，就是採用牛頓法來尋找最優點。而牛頓法就需要求解二階導數（Hessian）。因此對於諸如 XGBoost 這類機器學習框架，損失函數的二階可微是很有必要的。



XgBoost中使用的目標函數。注意對一階和二階導數的依賴性

但 Log-cosh 損失也並非完美，其仍存在某些問題。比如誤差很大的話，一階梯度和 Hessian 會變成定值，這就導致 XGBoost 出現缺少分裂點的情況。

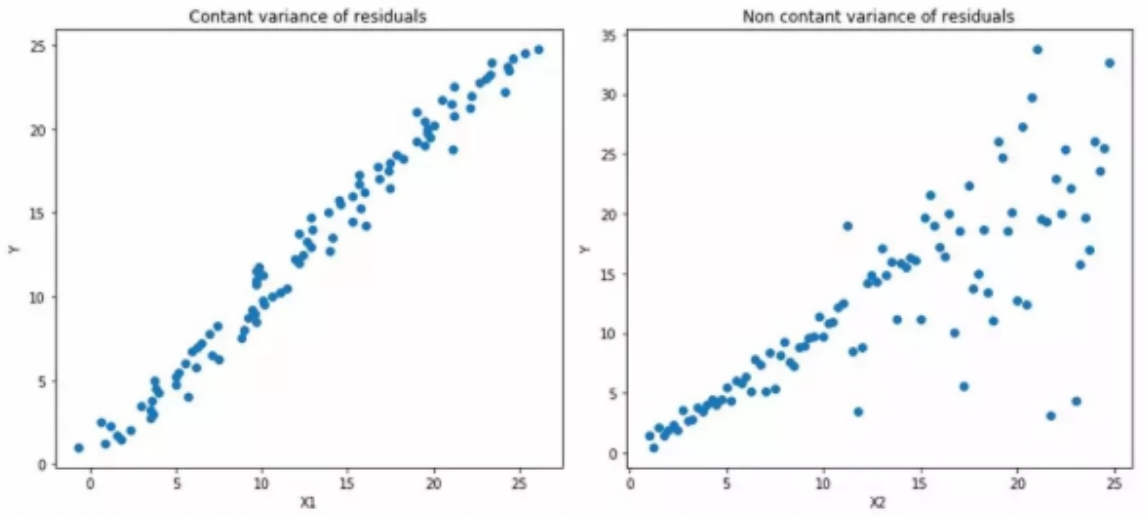
在大多數現實世界預測問題中，我們通常希望了解預測中的不確定性。清楚預測的範圍而非僅是估計點，對許多商業問題的決策很有幫助。

當我們更關注區間預測而不僅是點預測時，分位數損失函數就很有用。使用最小二乘回歸進行區間預測，基於的假設是殘差（y-y_hat）是獨立變量，且方差保持不變。

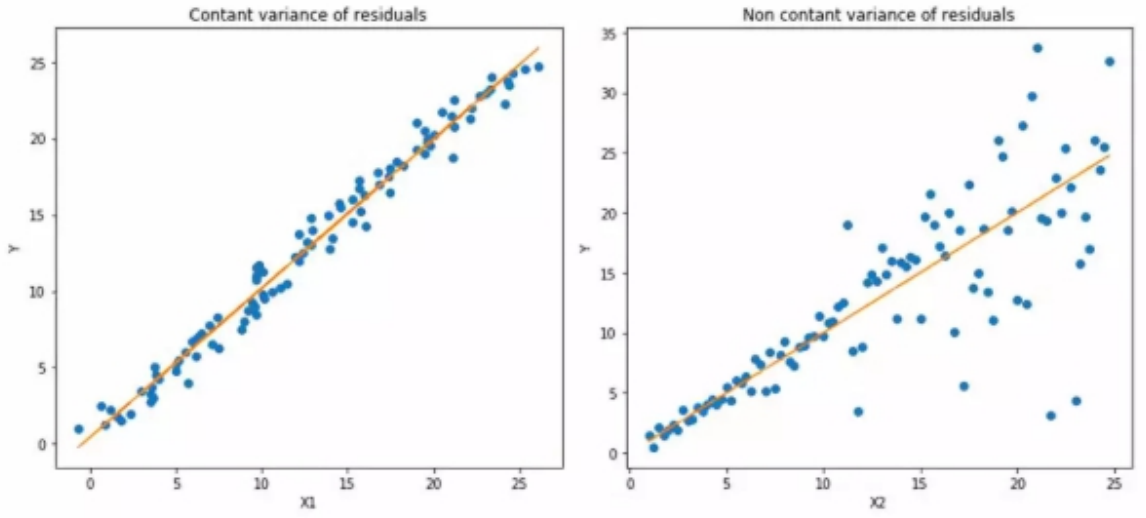
一旦違背了這條假設，那麼線性回歸模型就不成立。但是我們也不能因此就認為使用非線性函數或基於樹的模型更好，而放棄將線性回歸模型作為基線方法。這時，分位數損失和分位數回歸就派上用場了，因為即便對於具有變化方差或非正態分佈的殘差，基於分位數損失的回歸也能給出合理的預測區間。

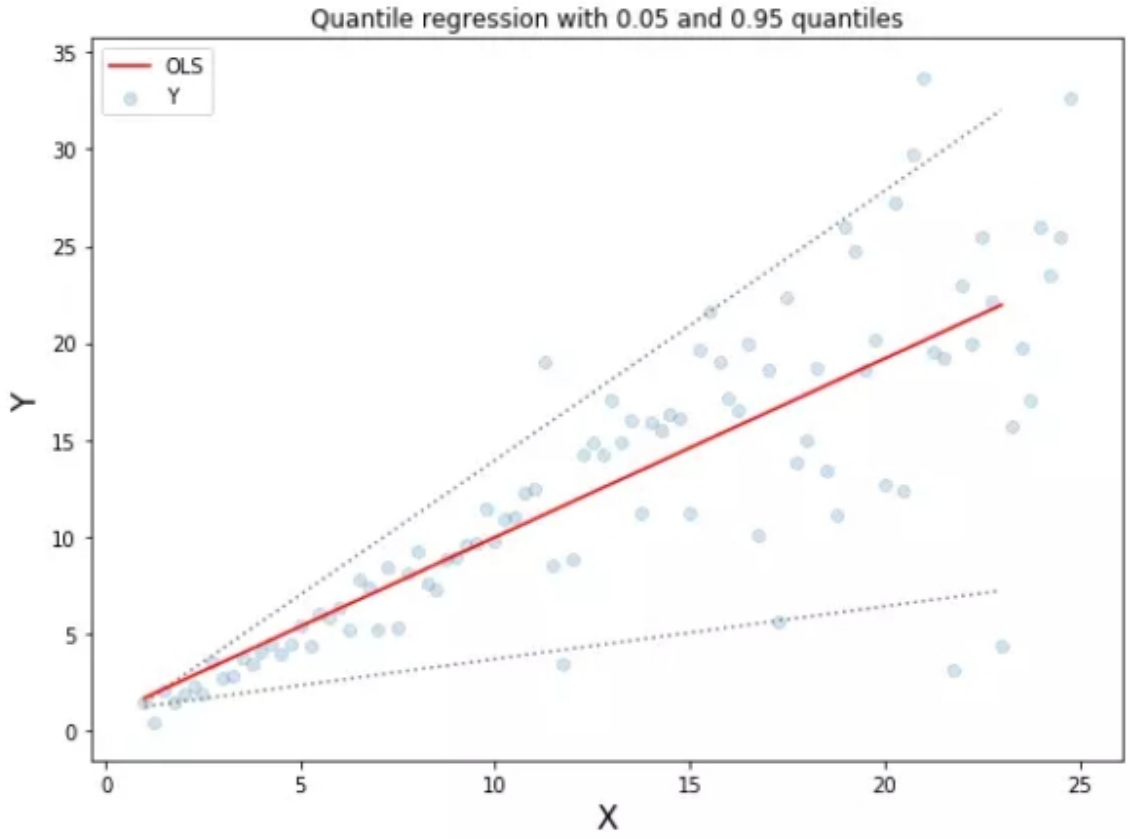
下面讓我們看一個實際的例子，以便更好地理解基於分位數損失的回歸是如何

對異方差數據起作用的。



左：b/wX1和Y為線性關係。具有恆定的殘差方差。右：b/wX2和Y為線性關係，但Y的方差隨著X2增加。（異方差）





附上圖中所示分位數回歸的代碼：

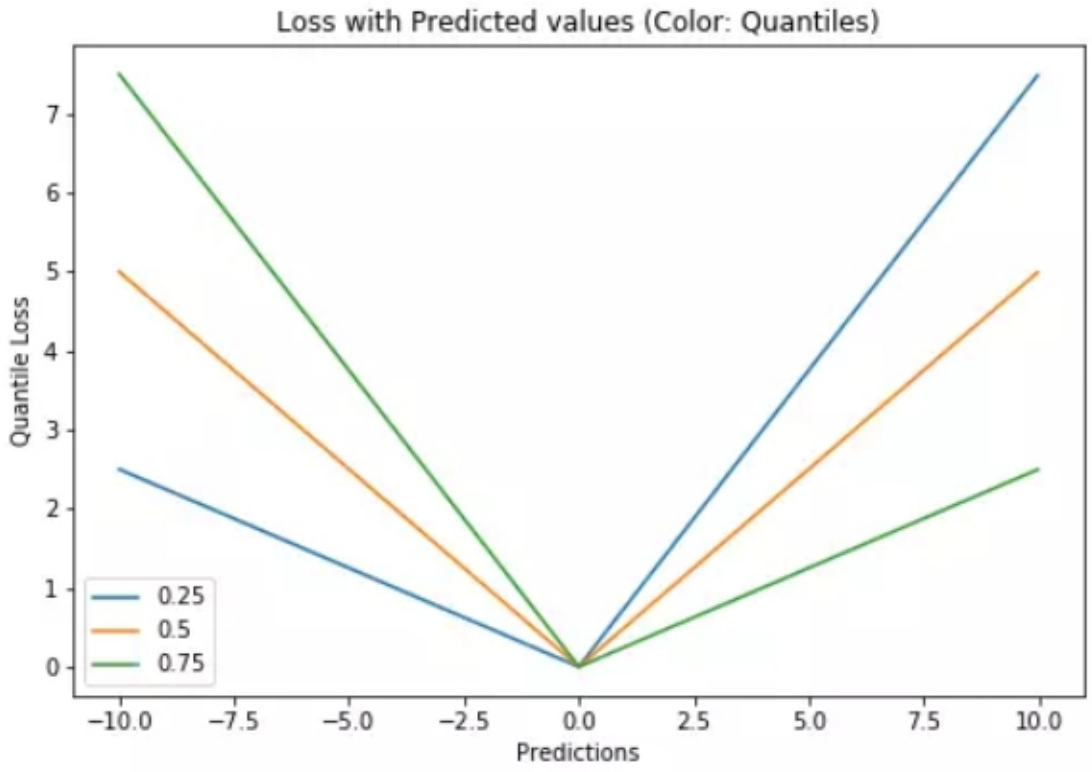
https://github.com/groverpr/Machine-Learning/blob/master/notebooks/09_Quantile_Regression.ipynb

理解分位數損失函數

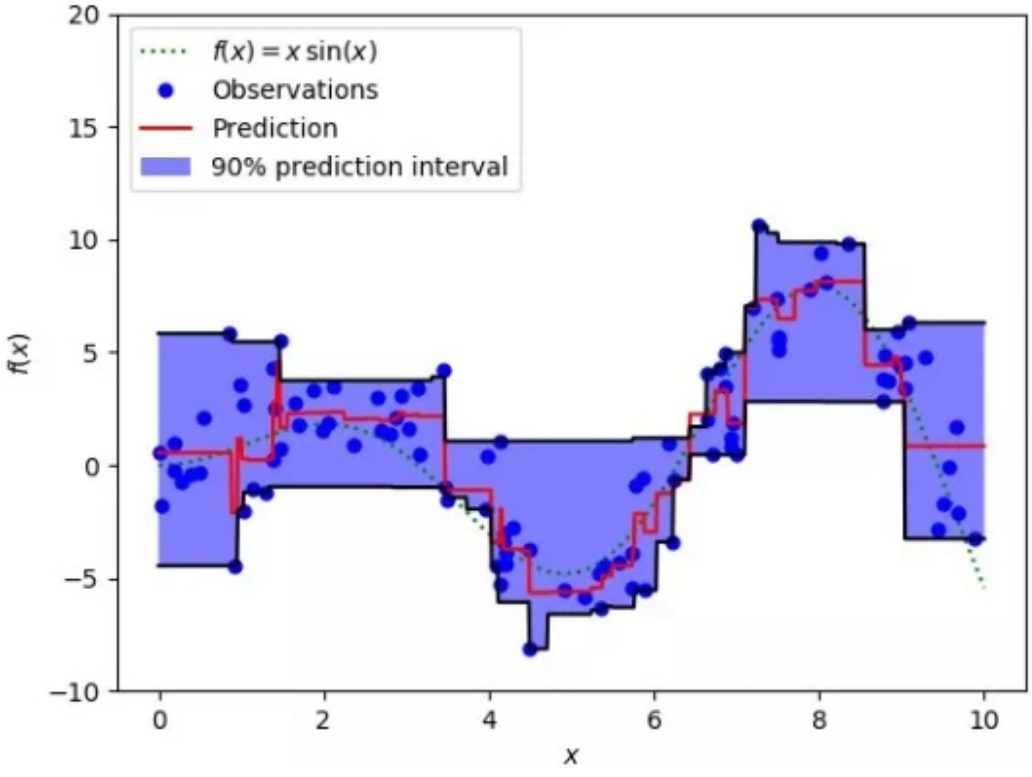
如何選取合適的分位值取決於我們對正誤差和反誤差的重視程度。損失函數通過分位值（γ）對高估和低估給予不同的懲罰。例如，當分位數損失函數 γ=0.25 時，對高估的懲罰更大，使得預測值略低於中值。

$$L_\gamma(y, y^p) = \sum_{i:y_i<y_i^p} (1-\gamma)|y_i - y_i^p| + \sum_{i:y_i \geq y_i^p} \gamma|y_i - y_i^p|$$

γ 是所需的分位數，其值介於 0 和 1 之間。



這個損失函數也可以在神經網絡或基於樹的模型中計算預測區間。以下是用 Sklearn 實現梯度提升樹回歸模型的示例。



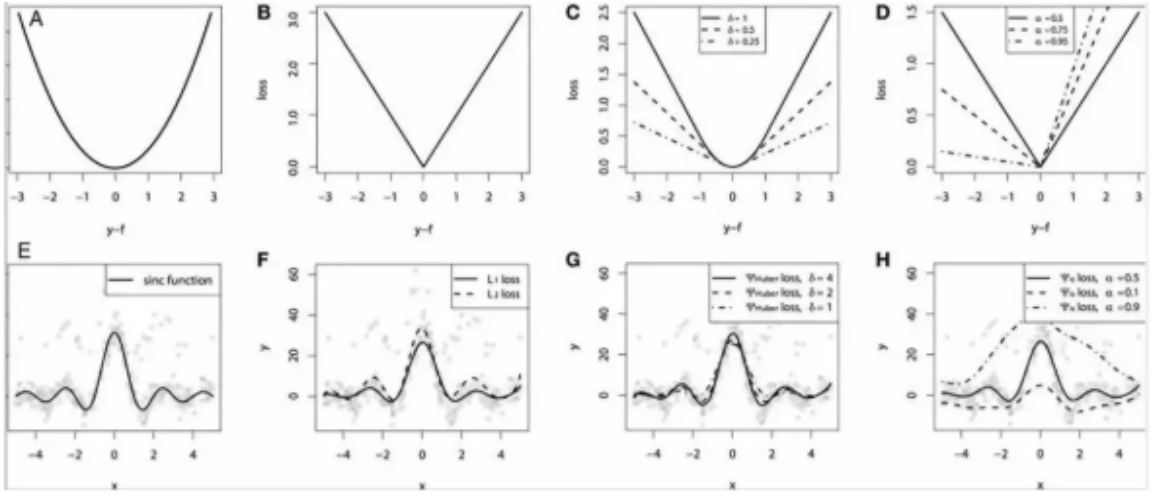
上圖表明：在 sklearn 庫的梯度提升回歸中使用分位數損失可以得到 90% 的

預測區間。其中上限為 γ=0.95，下限為 γ=0.05。

對比研究

為了證明上述所有損失函數的特點，讓我們來一起看一個對比研究。首先，我們建立了一個從 sinc（x）函數中採樣得到的數據集，並引入了兩項人為噪聲：高斯噪聲分量 ε∼N（0，σ2）和脈衝噪聲分量 ξ∼Bern（p）。

加入脈衝噪聲是為了說明模型的魯棒效果。以下是使用不同損失函數擬合 GBM 回歸器的結果。



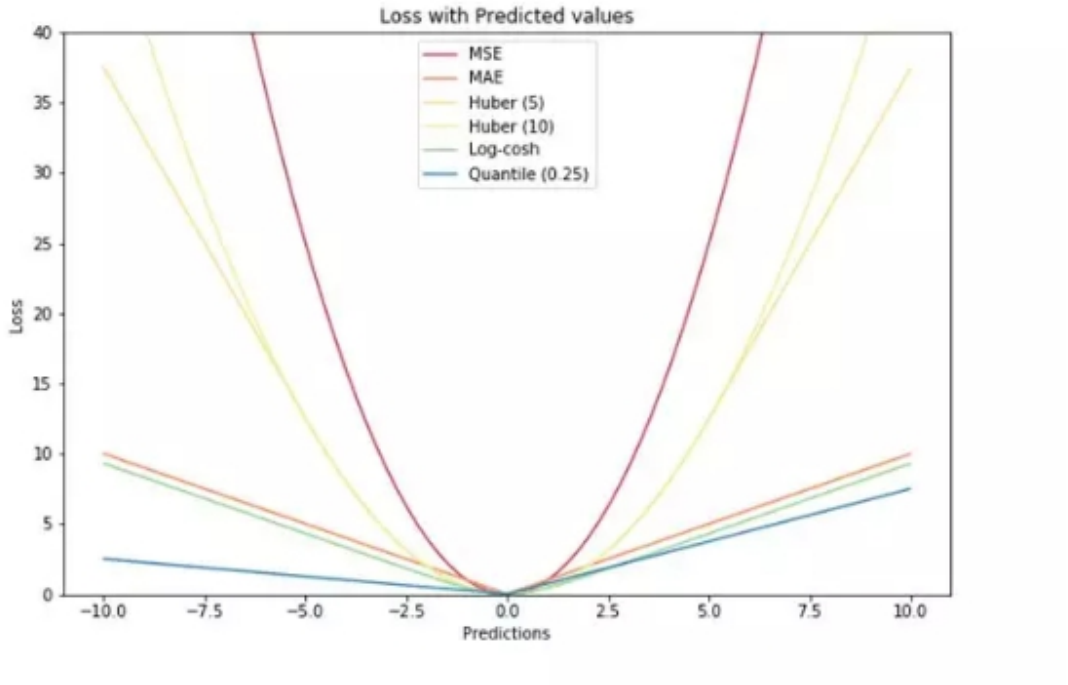
連續損失函數：（A）MSE 損失函數；（B）MAE 損失函數；（C）Huber 損失函數；（D）分位數損失函數。將一個平滑的 GBM 擬合成有噪聲的 sinc（x）數據的示例：（E）原始 sinc（x）函數；（F）具有 MSE 和 MAE 損失的平滑 GBM；（G）具有 Huber 損失的平滑 GBM ，且δ={4,2,1}；（H）具有分位數損失的平滑的 GBM，且α={0.5,0.1,0.9}。

仿真對比的一些觀察結果：

- MAE 損失模型的預測結果受脈衝噪聲的影響較小，而 MSE 損失函數的預測結果受此影響略有偏移。
- Huber 損失模型預測結果對所選超參數不敏感。
- 分位數損失模型在合適的置信水平下能給出很好的估計。

最後，讓我們將所有損失函數都放進一張圖，我們就得到了下面這張漂亮的圖

片！它們的區別是不是一目了然了呢~



——

延伸閱讀：

10 萬名工程師的真心話：2018 最符合世界趨勢的程式語言是？

工程師幹過最缺德的事：叫初學程式的朋友去學 C++！

【附 Github 代碼】工程師必備的「裝忙」祕技，讓電腦自己動起來的神奇小程式

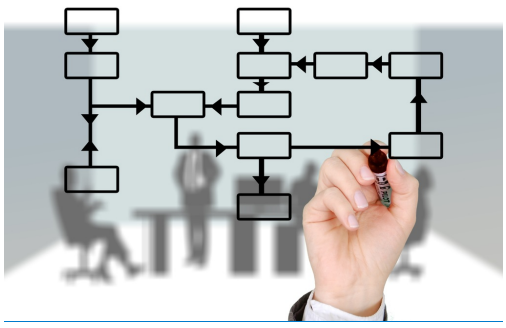
為什麼寫程式這麼難？

上萬名工程師都在知乎上面吵架：寫程式到底需不需要資工畢業？

（本文經 大數據文摘 授權轉載，並同意 TechOrange 編寫導讀與修訂標題，原文標題為 〈機器學習大牛最常用的 5 個回歸損失函數，你知道幾個？〉。）

點關鍵字看更多相關文章：

工程師 損失函數 機器學習優化 程式



專案管理必備 3 個基本功：不會寫 code 沒關係，至少要會畫 wireframe 吧！

怎樣才能算是一個卓越的 JavaScript 開發者？看你怎麼對待屎一樣的 code

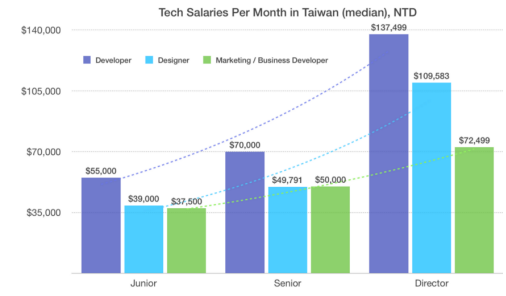
134 分享

衝上 GitHub 熱門第四名！Python 機器學習最強教學資源，新手工程師快存起來

1.4 K 分享

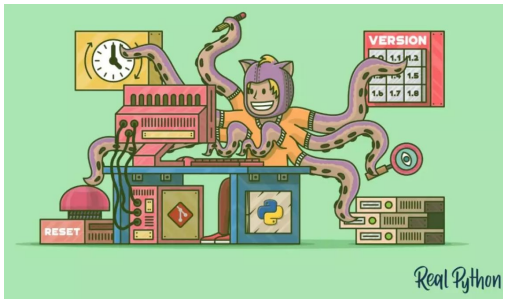
蘋果執行長庫克：要成為一位優秀的工程師，不一定要大學畢業

2.0 K 分享



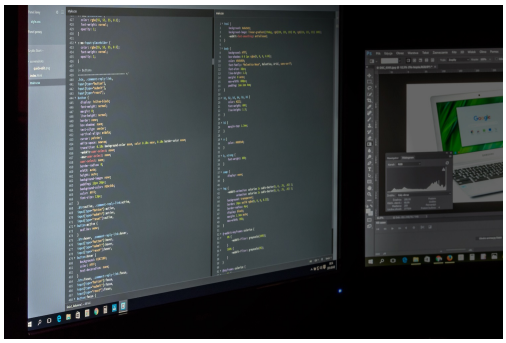
台灣科技業薪水大公開！你的薪資是否在同業中少的可憐？

3.0 K 分享



【GitHub 上破萬顆星】Python 新手 100 天學習計劃，這次學不會算我輸！

2.8 K 分享



GitHub 遭駭客攻擊！勒索交出比特幣贖金，不然就公開你的私有程式碼

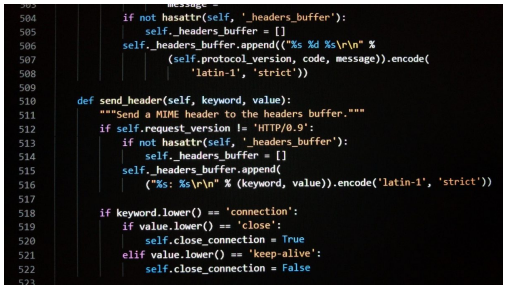
2.0 K 分享

【日本工程師の夢魘】改元「令和」工程師集體崩潰，後頭還有更可怕的「昭和 100 年」要解決呢

16.0 K 分享

【我只想靜靜 coding】菜鳥工程師是怎麼一步步失去理想與熱情的？問題其實出在主管身上

1.0 K 分享



```
            if not hasattr(self, '_headers_buffer'):
                self._headers_buffer = []
            self._headers_buffer.append(("%s %d %s\r\n" %
                    (self.protocol_version, code, message)).encode(
                        'latin-1', 'strict'))

    def send_header(self, keyword, value):
        """Send a MIME header to the headers buffer."""
        if self.request_version != 'HTTP/0.9':
            if not hasattr(self, '_headers_buffer'):
                self._headers_buffer = []
            self._headers_buffer.append(
                ("%s: %s\r\n" % (keyword, value)).encode('latin-1', 'strict'))

        if keyword.lower() == 'connection':
            if value.lower() == 'close':
                self.close_connection = True
            elif value.lower() == 'keep-alive':
                self.close_connection = False
```

【內附程式碼】工程師技能大全：如何用 Python 寫出所有的演算法？

9.3 K 分享