

HW10

612415013 蕭宥羽

1. List all parameters that should be set before running Adaboost. Explain the meanings of those parameters.

1. Base learners： Adaboost 作為一種強大的集成學習方法，其核心在於巧妙地組合多個簡單的學習模型，這些模型被稱為基學習器或弱學習器。在 Adaboost 的框架下，這些基學習器可以是多種形式，靈活性很高。常見的選擇包括簡單的決策樹、線性分類器，甚至是淺層的神經網絡。Adaboost 的魅力在於它能夠將這些相對簡單、可能單獨效果並不出色的學習器，通過特定的策略組合成一個性能優越的整體模型。這種方法不僅提高了模型的預測能力，還增強了其對不同類型數據的適應性。選擇合適的基學習器類型往往需要根據具體問題和數據特性來決定，這為算法的應用帶來了很大的靈活性。

2. Number of iterations： Adaboost 算法的精髓在於其迭代學習的過程，這一過程體現了算法不斷自我完善的能力。在每一次迭代中，Adaboost 都會仔細分析前一個基學習器的表現，特別關注其犯錯的地方。這些錯誤成為了寶貴的學習資源，指導算法如何調整和優化下一個基學習器。為了控制這個學習過程，算法引入了一個關鍵參數：迭代次數 T 。這個參數決定了 Adaboost 將訓練多少個基學習器，直接影響了最終集成模型的複雜度和性能。選擇適當的 T 值是一門藝術，需要在模型複雜度和學習能力之間取得平衡，以避免過擬合或欠擬合的風險。通過這種循序漸進、不斷改進的方式，Adaboost 能夠將多個相對簡單的學習器融合成一個強大的預測模型。

3. Learning rate： 在 Adaboost 算法的精妙設計中，學習率扮演著舉足輕重的角色，它如同一個微調旋鈕，精確控制著每個基學習器對最終模型的影響程度。這個參數通常表示為 α ，其取值範圍被慎重地限定在 0 到 1 之間，即 $0 < \alpha \leq 1$ 。學習率的巧妙之處在於它能夠在模型的學習過程中起到調節作用，防止算法過於激進地追隨單個基學習器的預測，從而降低過度擬合的風險。較小的學習率會使模型更加謹慎，逐步累積各個基學習器的貢獻，而較大的學習率則允許模型更快地適應訓練數據。


4. Initial data weight： Adaboost 算法的一大創新在於其巧妙運用數據權重機制，這一機制在算法的整個學習過程中起著關鍵作用。在訓練開始時，每個樣本都被賦予一個初始權重，這些權重可以視為算法對各個數據點的關注程度。有趣的是，這些初始權重的設置方式並非一成不變，而是可以根據具體問題的

需求進行靈活調整。在許多情況下，研究者們會選擇給所有樣本賦予相等的起始權重，體現出對每個數據點的公平對待。然而，在面對某些特殊問題時，比如處理不平衡數據集或需要特別關注某些樣本時，採用不均等的初始權重分配可能會更為合適。這種靈活的權重設置為 Adaboost 提供了處理各種複雜情況的能力，使得算法能夠更好地捕捉數據中的重要模式，並在迭代過程中不斷優化每個基學習器對最終模型的貢獻。

5. Base estimator parameters : 在 Adaboost 算法的實現過程中，基學習器的具體參數設置扮演著舉足輕重的角色，這些參數如同精密儀器的調節旋鈕，能夠微妙地影響整個集成模型的性能。這些參數的選擇並非一刀切，而是需要根據所選用的基學習器類型進行靈活調整。舉例來說，若選用決策樹作為基學習器，研究者可能需要仔細考慮樹的最大深度，這直接關係到模型的複雜度和泛化能力。相對地，如果採用神經網絡作為基礎，隱藏層的大小和結構就成為了關鍵考量因素，這些設置將決定網絡的學習容量和特徵提取能力。通過精心調節這些參數，研究者能夠在保持基學習器簡單性的同時，最大化其在集成過程中的效用，從而使 Adaboost 算法在面對各種複雜數據時都能展現出色的適應性和預測能力。

6. Data set : 在 Adaboost 算法的實施過程中，數據集扮演著重要的角色。這個算法需要一個精心準備的數據集來進行模型的訓練和測試。這個數據集可以是單一的，用於直接的訓練和評估；也可以是經過交叉驗證劃分的多個子集，以提高模型評估的可靠性。典型的 Adaboost 數據集包含 m 個樣本，每個樣本都有 n 個特徵，形成一個豐富的特徵空間。為了適應二元分類問題，每個樣本都會被賦予一個標籤，通常用 1 和 -1 來表示兩個不同的類別。這種結構化的數據集為 Adaboost 提供了必要的信息基礎，使其能夠逐步改進弱學習器，最終構建出一個強大的集成模型。

2. How each weak learner is decided and trained in each iteration? What is the learning algorithm A? Does it use bootstrapped dataset? If not, how D_t is obtained for each iteration?

 弱學習器的選擇和訓練

1. 初始化權重分佈：所有訓練樣本的初始權重 $D_1(i)$ 設定為相等，即每

個樣本的初始權重是 $\frac{1}{N}$ ，其中 N 是訓練樣本的數量。

2. 標籤轉換：設定一個閾值（5），將標籤轉換為二進制形式，大於等於 5 為真（1），小於 5 為假（0）。

3. 迭代訓練：AdaBoost 進行多個迴圈（由 cycles 參數控制）：

- ✓ 調用 weakLearner 函數選擇最佳的弱學習器。這個學習器是基於當前的權重分佈對訓練數據進行最佳擬合的特徵和閾值。
- ✓ 計算錯誤率：錯誤率是加權不正確分類的樣本權重的總和。錯誤 $error(j)$ 是當前弱學習器對加權樣本分類的錯誤率。
- ✓ 更新beta： $\beta(j) = \frac{error(j)}{1-error(j)}$ ，用於後續的權重更新
- ✓ 更新權重分佈：對於被正確分類的樣本，權重減少；對於被錯誤分類的樣本，權重增加。更新公式為：

$$D_{t+1}(i) = \frac{D_t(i) * e^{\log(\beta(j))(1-abs(label-(train(:,i))\geq t))}}{\Sigma(D_t)}$$

這裡使用了正規化步驟來確保權重分佈總和為 1。

 A

在這個實現中，學習算法 A 指的是 weakLearner 函數，它根據加權數據來選擇最佳的特徵和閾值作為分類器。具體的學習算法細節依賴於 weakLearner 的實現，通常包括檢查所有可能的特徵和閾值來找到最小化加權錯誤的分類器。

 是否使用 Bootstrap 資料集

AdaBoost 不使用 bootstrap 抽樣生成數據集。相反，它透過調整訓練樣本的權重來專注於那些在之前迭代中分類錯誤的樣本。

3. List the first three weak learners when the learning iteration stops.
Explain these decision stumps by their three parameters i , θ and s .

在 Adaboost 算法中，經過設定的 T 次迭代後，我們將獲得一個強分類器。

這一過程中，每一次迭代都會生成一個新的弱分類器，最終這些弱分類器會被綜合起來，形成一個性能更強的綜合分類器。

在使用弱分類器來判斷數據點的類別時，我們會設置一個閾值 θ 來劃定分界線，並確定分類的方向 s 。當數據點的某個特徵值超過這個閾值時，根據符號函數 sgn 的結果，這個數據點就會被分類為正類；如果未超過，則被分類為負類。這樣的設定讓弱分類器能夠有效地將輸入數據劃分為正類和負類。

4. Following 3), list the blending weights of these three decision stumps. Explain how their blending weights are decided and what are their actual values in the program?

$$w1 = 0.423$$

$$w2 = 0.290$$

$$w3 = 0.287$$

在每次迭代中，決策樁的混合權重是根據其在訓練集上的加權錯誤率 ϵ_t 和學習率 α_t 來確定的。這裡的分類錯誤率反映了決策樁在分類過程中的表現，而學習率則影響著每個決策樁對最終預測結果的貢獻度。計算決策樁混合權重的公式如下：

$$wt = \alpha_t * \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

在該程式中，每個樣本最初的權重被設定為 $1/m$ ，其中 m 代表樣本總數。隨著每次迭代，錯誤分類的樣本權重會增加，而正確分類的樣本權重則會減少。這些調整後的權重會被歸一化，使其總和為 1，為訓練下一個決策樁提供基礎。

在問題設定中，第一個決策樁的學習率設定為 0.5，而第二和第三個決策樁的學習率則均為 0.2。這三個決策樁在訓練過程中展示的分類錯誤率依次為 **0.283**、**0.328** 和 **0.346**。通過前面的公式，計算出三個決策樁的混合權重，如下所示：

$$w1 = 0.5 * \ln((1 - 0.283)/0.283) = \mathbf{0.423}$$

$$w2 = 0.2 * \ln((1 - 0.328)/0.328) = \mathbf{0.290}$$

$$w3 = 0.2 * \ln((1 - 0.346)/0.346) = \mathbf{0.287}$$

程式碼執行結果

