Framework Hiệu Chỉnh Siêu Tham Số AUTOMATA

Framework kết hợp chọn tập con dữ liệu thông minh với thuật toán tìm kiếm và lập lịch siêu tham số.

Giúp tăng tốc hiệu chỉnh mà vẫn giữ độ chính xác cao.

Giới thiệu AUTOMATA framework

Thuật toán tìm kiếm siêu tham số

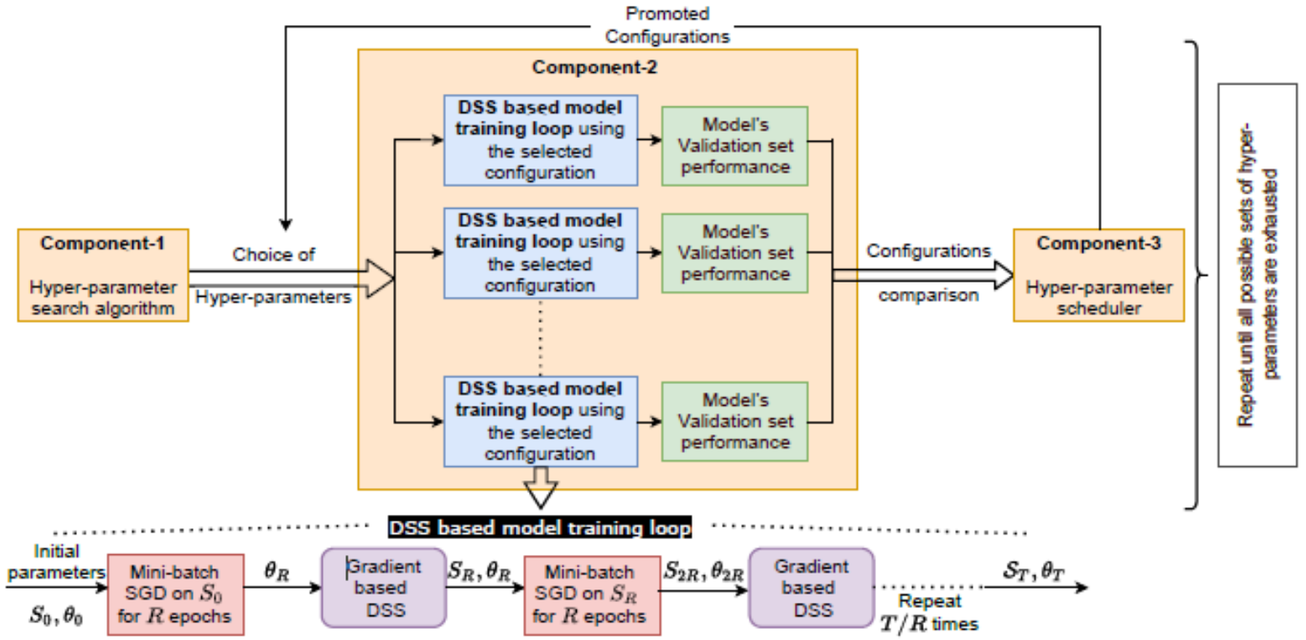
Xác định cấu hình siêu tham số cần đánh giá

Đánh giá cấu hình dựa trên tập con

Sử dụng tập con thông tin được chọn lọc dựa trên gradient để huấn luyện. Từ đó đánh giá các cấu hình tìm được.

Thuật toán lập lịch siêu tham số

Dừng sớm cấu hình kém, tăng hiệu quả



Thuật toán tìm kiếm siêu tham số

Thuật toán tìm kiếm ngẫu nhiên

Các cấu hình siêu tham số được chọn ngẫu nhiên và đánh giá để xác định cấu hình tối ưu trong số các lựa chọn đó.

TPE (Tree Parzen Structured Estimator)

TPE là một phương pháp tối ưu hóa dựa trên mô hình tuần tự (Sequential Model-Based Optimization - SMBO). Phương pháp này xây dựng một mô hình xác suất theo từng bước để xấp xỉ hiệu suất của các cấu hình siêu tham số dựa trên kết quả đánh giá lịch sử, và sử dụng mô hình đó để chọn ra các cấu hình mới.

Chọn Tập Con Dựa Trên Gradient (GSS)

Ý tưởng chính

Chọn tập con và trọng số sao cho gradient gần với toàn bộ dữ liệu

$$(w_i^t, S_i^t) = \arg\min_{w_i^t, S_i^t : |S_i^t| \le k, w_i^t \ge 0} \left\| \sum_{l \in S_i^t} w_{il}^t \nabla_\theta L_T^l(\theta_i^t) - \nabla_\theta L_T(\theta_i^t) \right\| + \lambda \| w_i^t \|^2$$

Thuật toán Orthogonal Matching Pursuit (OMP)

Thuật toán tham lam chọn tập con hiệu quả

Algorithm 3: OMP

Input: Training loss L_T, current parameters: θ, regularization coefficient: λ, subset size: k, tolerance: ϵ
Initialize $\mathcal{S} = \emptyset$
$r \leftarrow \nabla_w(\| \sum_{l \in \mathcal{S}} \boldsymbol{w} \nabla_\theta L_T^l(\theta) - \nabla_\theta L_T(\theta) \| + \lambda \|\boldsymbol{w}\|^2 |)_{\boldsymbol{w}=0}$
repeat
  $e = \text{argmax}_j |r_j|$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{e\}$
  $\boldsymbol{w} \leftarrow \text{argmin}_{\boldsymbol{w}}(\| \sum_{l \in \mathcal{S}} \boldsymbol{w} \nabla_\theta L_T^l(\theta) - \nabla_\theta L_T(\theta) \| + \lambda \|\boldsymbol{w}\|^2)$
  $r \leftarrow \nabla_{\boldsymbol{w}}(\| \sum_{l \in \mathcal{S}} \boldsymbol{w} \nabla_\theta L_T^l(\theta) - \nabla_\theta L_T(\theta) \| + \lambda \|\boldsymbol{w}\|^2 |)$
until *until* $|\mathcal{S}| \le k$ *and* $\| \sum_{l \in \mathcal{S}} \boldsymbol{w} \nabla_\theta L_T^l(\theta) - \nabla_\theta L_T(\theta) \| + \lambda \|\boldsymbol{w}\|^2 \ge \epsilon$
return $\mathcal{S}, \boldsymbol{w}$

Tăng tốc độ cho GSS

Warm-starting

Huấn luyện toàn bộ dữ liệu vài epoch để thu gradient thông tin. Cụ thể, toàn bộ dữ liệu được huấn luyện trong :

$$T_w = \frac{\kappa T k}{N} \text{ epochs}$$

Trong đó T: tổng số epoch,

k: kích thước coreset,

N: kích thước tập dữ liệu,

κ: tham số khởi động

Chọn tập con theo batch

Thay vì chọn trực tiếp các điểm dữ liệu, ta chọn các **mini-batch** bằng cách so khớp tổng có trọng số của gradient huấn luyện trên các mini-batch với gradient hàm mất mát trên toàn bộ tập huấn luyện. Kết quả là ta có một tập con các mini-batch được chọn cùng với trọng số tương ứng. Mô hình được huấn luyện trên các mini-batch này bằng **gradient descent theo mini-batch với hàm mất mát có trọng số**.

Tác động: số vòng lặp tham lam trong thuật toán OMP được giảm xuống, từ đó mang lại tốc độ cải thiện. Tuy nhiên, có một sự đánh đổi quan trọng: dùng batch lớn sẽ cải thiện tốc độ nhưng làm giảm chất lượng chọn tập con

Algorithm 2: subset-config-evaluation

Input: Training dataset: $\mathcal{D}$, Validation dataset: $\mathcal{V}$, Initial model parameters: θ_0, Total no of epochs: T, Epoch interval for subset selection: R, Size of the coreset: k, Reg. Coefficient: λ, Learning rates: $\{\alpha_t\}_{t=0}^{t=T-1}$, Tolerance: ϵ

Set $t = 0$; Randomly initialize coreset $\mathcal{S}_0 \subseteq \mathcal{D} : |\mathcal{S}_0| = k$;

repeat

  if $(t\%R == 0) \wedge (t > 0)$ **then**

    $\mathcal{S}_t$ = OMP($\mathcal{D}, \theta_t, \lambda, \alpha_t, k, \epsilon$)

  else

    $\mathcal{S}_t = \mathcal{S}_{t-1}$

  Compute batches $\mathcal{D}_b = ((x_b, y_b); b \in (1 \cdots B))$ from $\mathcal{D}$

  Compute batches $\mathcal{S}_{tb} = ((x_b); b \in (1 \cdots B))$ from $\mathcal{S}$

  *** Mini-batch SGD ***

  Set $\theta_{t0} = \theta_t$

  for $b = 1$ *to* B **do**

    Compute mask $\boldsymbol{m}_t$ on $\mathcal{S}_{tb}$ from current model parameters $\theta_{t(b-1)}$

    $\theta_{tb} = \theta_{t(b-1)} - \alpha_t \nabla_\theta L_S(\mathcal{D}_b, \theta_t) - \alpha_t \lambda_t \sum_{j \in \mathcal{S}_{tb}} \boldsymbol{m}_{jt} \nabla_\theta l_u(x_j, \theta_t(b-1))$

  Set $\theta_{t+1} = \theta_{tB}$

  $t = t + 1$

until *until* $t \geq T$

*** Evaluate trained model on validation set ***

$eval = \text{evaluate}\ (\theta_T, \mathcal{V})$

return $eval, \theta_T$

Thuật toán lập lịch siêu tham số

Sequential Halving (SHA)

Thuật toán SHA bắt đầu với nnn cấu hình ban đầu, mỗi cấu hình được cấp lượng tài nguyên tối thiểu là rrr. SHA sử dụng một hệ số giảm η để giảm số lượng cấu hình qua mỗi vòng lặp bằng cách giữ lại $\frac{1}{\eta}$ số cấu hình tốt nhất, đồng thời tăng lượng tài nguyên cho các cấu hình này lên η lần sau mỗi vòng

Hyperband

Một biến thể của SHA. Khắc phục vấn đề hiệu suất của thuật toán phụ thuộc nhiều vào số lượng cấu hình ban đầu n

ASHA

ASHA là một biến thể phi tuần tự (asynchronous) của SHA. Thuật toán khắc phục tính chất tuần tự của thuật toán SHA: nó phải chờ tất cả các tiến trình (được cấp tài nguyên bằng nhau) trong một nhóm hoàn thành trước khi chọn ra các cấu hình cho vòng tiếp theo. Do đó, trong các thiết lập huấn luyện phân tán, tài nguyên GPU/CPU có thể không được sử dụng hiệu quả, dẫn đến mất nhiều thời gian hơn để tinh chỉnh siêu tham số.

Algorithm 1: AUTOMATA Algorithm

Input: Hyper-parameter scheduler Algorithm: scheduler , Hyper-parameter search Algorithm: search , No. of configuration evaluations: n, Hyper-parameter search space: $\mathcal{H}$, Training dataset: $\mathcal{D}$, Validation dataset: $\mathcal{V}$, Total no of epochs: T, Epoch interval for subset selection: R, Size of the coreset: k, Reg. Coefficient: λ, Learning rates: $\{\alpha_t\}_{t=0}^{t=T-1}$, Tolerance: ϵ

Generate n configurations by calling the search algorithm $H = \{h_1, h_2, \cdots, h_n\} = \text{search } (\mathcal{H}, n)$

Randomly initialize each configuration model parameters $h_1.\theta = h_2.\theta = \cdots = h_n.\theta = \theta$

Set $h_1.t = h_2.t = \cdots = h_n.t = 0$;

Set $h_1.eval = h_2.eval = \cdots = h_n.eval = 0$;

Assign the initial resources(i.e., in our case training epochs) using the scheduler for all initialized configurations $\{h_i.r\}_{i=1}^{i=n} = \text{scheduler } (H, T)$

repeat

  Evaluate all remaining configurations

  for *each configuration numbered i in H* **do**

    Train configuration h_i using informative data subsets for $h_i.r$ epochs and evaluate on validation set

    $h_i.eval, h_i.theta = \text{subset-config-evaluation } (\mathcal{D}, \mathcal{V}, h_i.theta, h_i.r, R, k, \lambda, \{\alpha_t\}_{t=0}^{t=h_i.r}, \epsilon)$

    $h_i.t = h_i.t + h_i.r$

  Assign resources again based on evaluation performance

  $\{h_i.r\}_{i=1}^{i=n} = \text{scheduler } (H, T)$

until *until* $h_1.r == 0 \ \ \& \ \ h_2.r == 0 \ \ \& \ \ \cdots h_n.r == 0$

Get the best performing hyper-parameters based on final configuration evaluations $finalconfig = \underset{h_i.config}{\text{argmax}} \ [h_i.eval]_{i=1}^{n}$ ***Perform final training using the best hyper-parameter configurations*** $\theta_{final} = \text{finaltrain } (\theta, \mathcal{D}, finalconfig, T)$ **return** θ_{final}

Thực nghiệm đa dạng tập dữ liệu

Dữ liệu văn bản

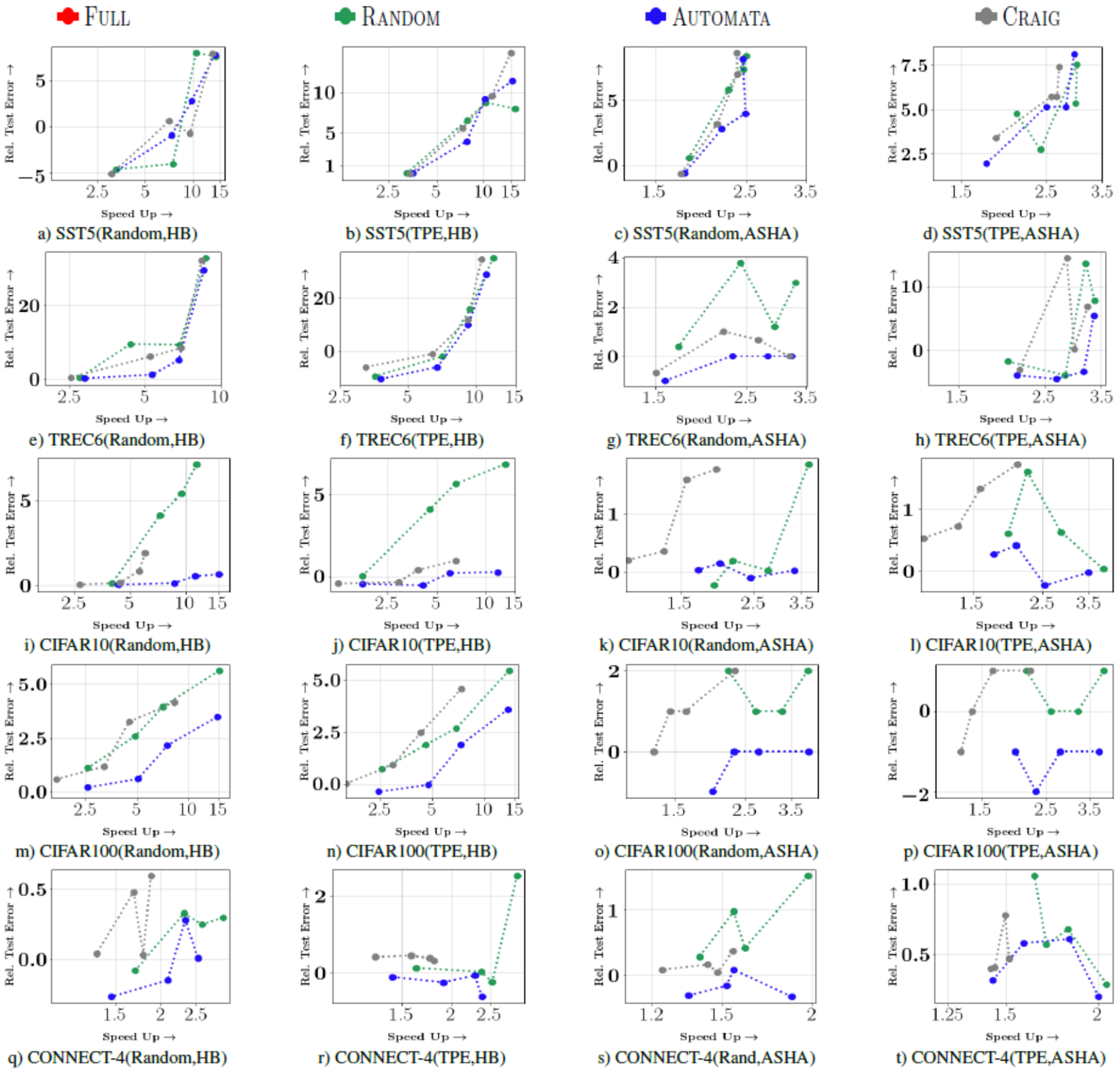
SST2, SST5, GLUE-SST2, TREC6 - Mô hình LSTM với embedding GloVe

Dữ liệu hình ảnh

CIFAR10, CIFAR100, SVHN - Mô hình ResNet18, ResNet50

Dữ liệu bảng

DNA, SATIMAGE, LETTER, CONNECT-4 - Mô hình MLP 2 tầng ẩn


FULL
RANDOM
AUTOMATA
CRAIG
Rel. Test Error →
Speed Up →
a) SST5(Random,HB)
b) SST5(TPE,HB)
c) SST5(Random,ASHA)
d) SST5(TPE,ASHA)
e) TREC6(Random,HB)
f) TREC6(TPE,HB)
g) TREC6(Random,ASHA)
h) TREC6(TPE,ASHA)
i) CIFAR10(Random,HB)
j) CIFAR10(TPE,HB)
k) CIFAR10(Random,ASHA)
l) CIFAR10(TPE,ASHA)
m) CIFAR100(Random,HB)
n) CIFAR100(TPE,HB)
o) CIFAR100(Random,ASHA)
p) CIFAR100(TPE,ASHA)
q) CONNECT-4(Random,HB)
r) CONNECT-4(TPE,HB)
s) CONNECT-4(Rand,ASHA)
t) CONNECT-4(TPE,ASHA)

Kết Quả Tinh Chỉnh Siêu Tham Số

Văn bản

Tốc độ nhanh hơn 3-10 lần, độ chính xác giữ hoặc tăng

Hình ảnh

Tốc độ tăng 2-15 lần, mất độ chính xác rất nhỏ

Bảng

Vượt trội so với các phương pháp chọn tập con khác