AUTOMATA: LỰA CHỌN TẬP CON DỮ LIỆU DỰA TRÊN GRADIENT ĐỂ ĐIỀU CHỈNH SIÊU THAM SỐ HIỆU QUẢ TÍNH TOÁN

(Dưa trên bài báo "AUTOMATA : GRADIENT BASED DATA SUBSET SELECTION FOR COMPUTE-EFFICIENT HYPER-PARAMETER TUNING" của Krishnateja, 2022)

Người trình bày: Vũ Hoài Thư

Nội dung

1. Đặt vấn đề
2. Kiến trúc hệ thống AUTOMATA
   - Component-1
   - Component-2
   - Component-3
3. Thực nghiệm và kết quả

Đặt vấn đề

Đặt vấn đề

- Việc tối ưu siêu tham số là thiết yếu cho mô hình học sâu.
- Tuy nhiên, quá trình này tốn kém tài nguyên, yêu cầu huấn luyện nhiều lần trên toàn bộ dữ liệu.
- Dẫn đến chi phí tính toán lớn, phát thải CO_2 cao và tiêu tốn nhiều thời gian.

$=>$ Cần có giải pháp giảm chi phí mà vẫn đảm bảo hiệu quả của các mô hình.

Ý tưởng chính của bài báo

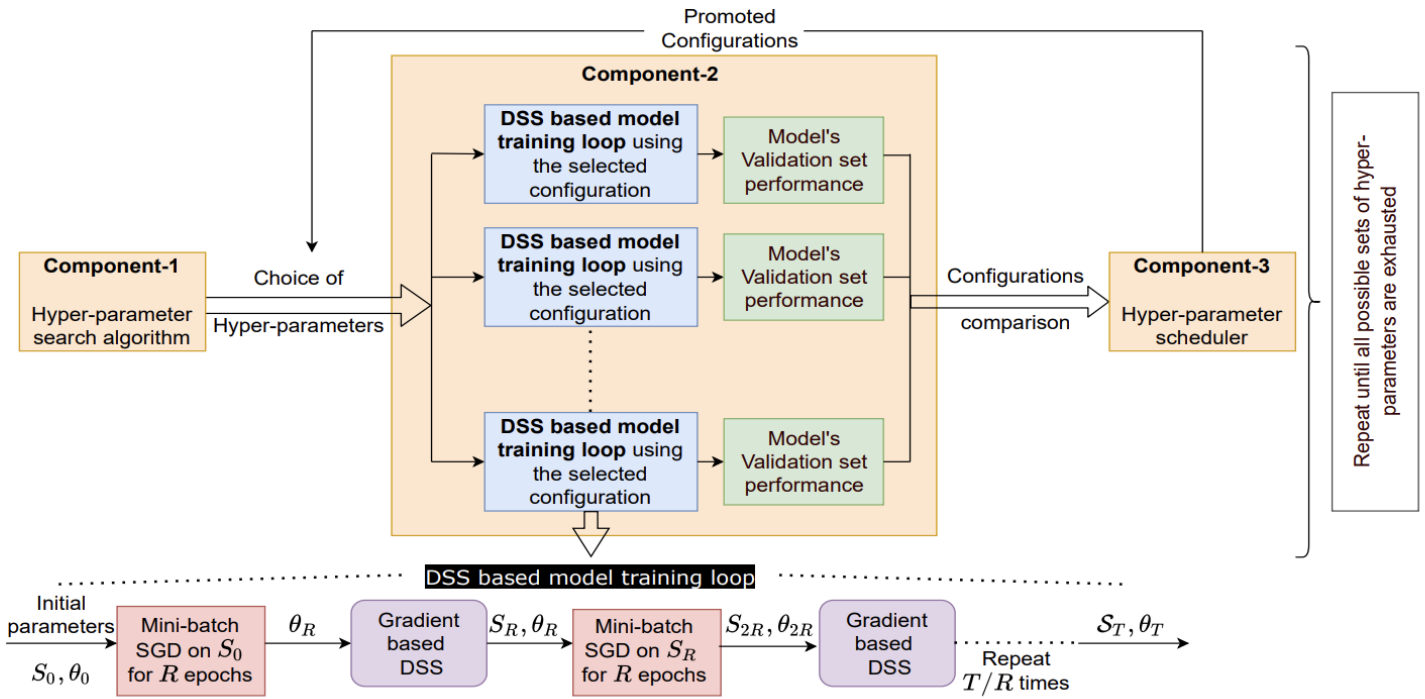
- Tận dụng tập dữ liệu con mang tính thông tin cao thay vì toàn bộ dữ liệu.
- Đề xuất **AUTOMATA** – một framework chọn dữ liệu con dựa trên gradient để đánh giá cấu hình siêu tham số.
- Mục tiêu: giảm thời gian và chi phí mà vẫn duy trì hiệu quả mô hình học sâu.

Kiến trúc hệ thống AUTOMATA

Kiến trúc hệ thống

Kiến trúc hệ thống AUTOMATA gồm 3 thành phần:

- Thuật toán tìm kiếm siêu tham số (Random, TPE)
- Thuật toán chọn tập con dựa trên gradient (GSS)
- Bộ lập lịch siêu tham số (Hyperband, ASHA)



Component-1: Thuật toán tìm kiếm siêu tham số

- Cách đơn giản để tìm siêu tham số là sử dụng Grid-Search, tuy nhiên cách này mất nhiều thời gian và cần đánh giá trên toàn bộ tập dữ liệu.
- Bài báo đề xuất 2 thuật toán tìm kiếm siêu tham số là Random Search và Tree-structured Parzen Estimator (TPE).
- Random Search: Chọn ngẫu nhiên trong không gian siêu tham số để tìm ra bộ siêu tham số tối ưu, hiệu quả hơn GridSearch về thời gian, giảm tình trạng overfitting.
- TPE: Lựa chọn cấu hình tiềm năng dựa trên phân phối xác suất học được.

Thuật toán TPE (Tree-structured Parzen Estimator) (1/3)

- TPE không lựa chọn ngẫu nhiên (random) hay thử hết (grid).
- TPE xây dựng mô hình xác suất để dự đoán các cấu hình có khả năng tốt.
- TPE xây dựng một mô hình xác suất để xấp xỉ hiệu suất của các cấu hình siêu tham số dựa trên kết quả đánh giá trong quá khứ, từ đó chọn ra các cấu hình mới tiềm năng hơn.

Thuật toán TPE (Tree-structured Parzen Estimator) (2/3)

Các bước thực hiện:

- **Bước 1**: Thử ngẫu nhiên một vài cấu hình ban đầu và đánh giá hiệu suất
- **Bước 2**: Dựa vào lịch sử các cấu hình đã thử => Chia nhóm các cấu hình: Dựa trên giá trị đánh giá (ví dụ: độ chính xác trên tập validation), TPE sắp xếp các quan sát thành 2 nhóm theo một ngưỡng phân vị (quantile):
  - Nhóm tốt x_1: các cấu hình cho kết quả tốt hơn ngưỡng y^*
  - Nhóm còn lại x_2: các cấu hình kém hơn

Thuật toán TPE (Tree-structured Parzen Estimator) (3/3)

- **Bước 3:** Xây dựng hai phân phối xác suất bằng kỹ thuật KDE (kernel density estimation):
  - $l(x) = p(x|y < y^*)$ - mật độ của cấu hình tốt
  - $g(x) = p(x|y \geq y^*)$ - mật độ của cấu hình kém
- **Bước 4:** Chọn mô hình mới để đánh giá: Thay vì chọn ngẫu nhiên, TPE lấy mẫu từ phân phối tối đa hóa giá trị kỳ vọng:

$$\max \mathrm{E}[\frac{l(x)}{g(x)}]$$

Nghĩa là chọn các cấu hình mà theo mô hình, có tiềm năng cải thiện lớn nhất.

Component-2: Thuật toán chọn tập dữ liệu con (Data Subset Selection - DSS) (1/2)

- Chọn tập con thông minh từ tập huấn luyện bằng cách xấp xỉ gradient.
- Mỗi cấu hình siêu tham số được huấn luyện trên một tập con riêng biệt phù hợp với nó.
- Mục tiêu: Tìm tập con $S \in D$ và trọng số w sao cho gradient loss trên S gần nhất với gradient trên toàn bộ tập dữ liệu huấn luyện D:

$$(w_i^t, S_i^t) = \arg \min_{w \geq 0, |S| \leq k} || \sum_{l \in S} w_l \nabla_\theta L_l(\theta) - \nabla_\theta L(\theta)|| + \lambda ||w||^2$$

Component-2: Thuật toán chọn tập dữ liệu con (Data Subset Selection - DSS) (2/2)

Bài báo đề xuất các thuật toán chọn tập dữ liệu con:

- Thuật toán Orthogonal Matching Pursuit (OMP)
  - Chọn mẫu có gradient gần vuông góc nhất với residual hiện tại
  - Cập nhật tập con và trọng số tuần tự theo cách tham lam (greedy)
  - Bảo đảm hiệu quả gần tối ưu với chi phí tính toán thấp
- Per-batch Subset Selection (Chọn theo mini-batch)
  - Chọn mini-batch thay vì từng mẫu $\rightarrow$ giảm thời gian chọn tập con
  - Áp dụng gradient trung bình trên mini-batch thay cho từng điểm
  - Tăng tốc lựa chọn tập con gấp B lần (với batch size B).

Thuật toán OMP (1/2)

Mục tiêu của thuật toán OMP trong framework AUTOMATA: Tìm tập con $S \subset D$ và trọng số w sao cho:

$$\sum_{l \in S} w_l \nabla_\theta L_T^l(\theta) \approx \nabla_\theta L_T(\theta)$$

Trong đó:

- L_T: Training loss, L_T^l: loss tại điểm dữ liệu l
- θ: Tham số mô hình
- w: vector trọng số gán cho các mẫu được chọn
- λ: hệ số regularization (để hạn chế overfitting lên mẫu ít)
- k: số lượng điểm tối đa được chọn vào tập con S
- ϵ: ngưỡng dừng thuật toán (tolerance)

Thuật toán OMP (2/2)

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* $|\mathcal{S}| \leq k$ *and* $\| \sum_{l \in \mathcal{S}} \boldsymbol{w} \nabla_\theta L_T^l(\theta) - \nabla_\theta L_T(\theta) \| + \lambda \|\boldsymbol{w}\|^2 \geq \epsilon$
return $\mathcal{S}, \boldsymbol{w}$

Hình: Thuật toán OMP

Thuật toán Per-Batch Subset Selection (1/2)

Các ký hiệu chính:

- Ý tưởng chính: Thay vì chọn các mẫu đơn lẻ như OMP, thuật toán chọn ra một tập con các mini-batch sao cho tổng gradient có trọng số của các mini-batch gần nhất với gradient của toàn bộ tập huấn luyện.
- Tại mỗi thời điểm t, bài toán chọn tập con mini-batch:

$$(w^t_{B_i}, S^t_{B_i}) = \arg \min_{w \geq 0, |S| \leq b_k} || \sum_{l \in S} w_l \nabla_\theta L_T^{B_l}(\theta^t_i) - \nabla_\theta L_T^B(\theta^l_i)|| + \lambda ||w||^2$$

- Mục tiêu: chọn tập con S các mini-batch và trọng số w, sao cho tổng gradient có trọng số xấp xỉ gradient toàn bộ.

Thuật toán Per-Batch Subset Selection (1/2)

Trong đó:

- B: kích thước của mỗi mini-batch
- N: tổng số mẫu dữ liệu
- $b_N = \frac{N}{B}$: số lượng mini-batch
- k: số lượng điểm cần chọn (subset size)
- $b_k = \frac{k}{B}$: số lượng mini-batch cần chọn
- D_B: tập dữ liệu huấn luyện chi theo mini-batch
- $\nabla_\theta L_T^{B_i}(\theta)$: gradient loss của mini-batch thứ i
- w_{B_i}: trọng số gán cho mỗi mini-batch thứ i

=> Vấn đề warm-start: Trước khi chọn tập con, mô hình được huấn luyện sơ bộ vài epoch trên toàn bộ dữ liệu => Giúp thu được gradient ổn định hơn, tăng chất lượng của tập con được chọn.

Thuật toán 2: subset-config-evaluation

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 ***

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

return $eval, \theta_T$

Component-3: Bộ lập lịch siêu tham số

- Phân bổ tài nguyên huấn luyện theo hiệu suất ban đầu.
- Hai thuật toán sử dụng:
  - Hyperband: dừng sớm cấu hình kém sau mỗi vòng.
  - ASHA (Asynchronous Successive Halving Algorithm): phiên bản không đồng bộ, sử dụng tài nguyên tốt hơn.

Thực nghiệm và kết quả

Thực nghiệm

- Dữ liệu sử dụng:
  - Văn bản: SST2, SST5, glue-SST2, TREC6
  - Ảnh: CIFAR10, CIFAR100, SVHN
  - Dữ liệu dạng bảng: CONNECT-4, DNA, LETTER
- So sánh với các trường hợp sử dụng đầy đủ dữ liệu, chọn ngẫu nhiên dữ liệu và CRAIG

Kết quả

- Tăng tốc 3×–30× so với tuning trên toàn bộ dữ liệu.
- Mức giảm hiệu năng nhỏ (thường $< 2\%$), đôi khi còn tăng nhẹ độ chính xác.
- Giảm đáng kể lượng CO_2 phát thải.
- AUTOMATA cho kết quả tốt nhất về cân bằng tốc độ và độ chính xác so với các phương pháp khác.

Đề xuất một số hướng phát triển tiếp theo

- Tự động điều chỉnh kích thước coreset: Giá trị k đang lựa chọn thủ công $=>$ đề xuất: xây dựng cơ chế tự điều chỉnh k theo độ hội tụ của gradient.
- Thay vì chọn tập con bằng thuật toán như OMP, có thể huấn luyện một mô hình học sâu để dự đoán xác suất chọn mẫu (Sử dụng GNN hoặc Reinforcement Learning).