Efficient Hyperparameter Optimization of Deep Learning Algorithms Using Deterministic RBF Surrogates

Yakovlev Konstantin

BMM, March 2023

Goals of the research

Problem

Automatically searching for optimal hyperparameter configurations

Challenge

Probabilistic surrogates require accurate estimates of sufficient statistics. This makes them inefficient for optimizing hyperparameters of deep learning algorithms.

Solution

Consider radial basis functions as error surrogates. The proposed algorithm (HORD) requires fewer function evaluations.

Problem statement

Geven a set of hyperparameters $\mathbb{R}^D$, train and validation datasets: $\mathfrak{D}_{\text{train/val}}$, model parameters $\boldsymbol{\theta}$, and black-box error function f.

$$\begin{aligned}
\min_{\mathbf{x}\in\mathbb{R}^D} \quad & f(\mathbf{x}, \boldsymbol{\theta}, \mathfrak{D}_{\text{val}}), \\
\text{s.t.} \quad & \boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} f(\mathbf{x}, \boldsymbol{\theta}, \mathfrak{D}_{\text{train}})
\end{aligned}$$

The surrogate model

Given a hyperparameter configuration $\mathbf{x}_{1:n}$ of size n and its corresponding validation errors $f_{1:n}$. Define the RBF interpolation model as:

$$S_n(\mathbf{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|_2) + p(\mathbf{x}),$$

where $\phi(r) = r^3$ - cubic spline RBF, $p(\mathbf{x}) = \mathbf{b}^\top \mathbf{x} + a$. The parameters of the spline are determined by solving the following system:

$$\begin{cases} \mathbf{\Phi}\boldsymbol{\lambda} + \mathbf{P}\mathbf{c} = \mathbf{F}, \quad \mathbf{\Phi} = \{\phi(\|\mathbf{x}_i - \mathbf{x}_j\|_2)\}_{i,j=1}^{n}, \ \mathbf{P} = [\{\mathbf{x}_i^\top, 1\}_{i=1}^{n}]^\top \\ \mathbf{P}^\top \boldsymbol{\lambda} = \mathbf{0}, \quad \mathbf{F} = [\{f(\mathbf{x}_i)\}_{i=1}^{n}], \ \mathbf{c} = [\mathbf{b}, a]. \end{cases}$$

Spline theory

Show that $\mathbf{P}^\top \boldsymbol{\lambda} = \mathbf{0}$ is necessary for minimizing energy. For simpicity, consider $D = 1$.

$$J(S) = \int_{\mathbb{R}^1} [S''(\mathbf{x})]^2 d\mathbf{x} = 12 \sum_{i=1}^{n} \lambda_i S(\mathbf{x}_i) =$$

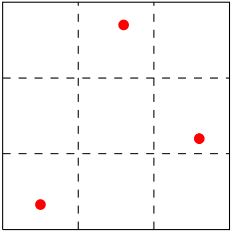
$$12(\boldsymbol{\lambda}^\top \boldsymbol{\Phi} \boldsymbol{\lambda} + \boldsymbol{\lambda}^\top \mathbf{P}\mathbf{c}) \to \min_{\mathbf{c}} \Rightarrow \mathbf{P}^\top \boldsymbol{\lambda} = \mathbf{0}.$$

Theorem

A matrix $\begin{pmatrix} \boldsymbol{\Phi} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{0} \end{pmatrix}$ is non singular $\Leftrightarrow$ columns of $\mathbf{P}$ are linearly dependent. [a]

[a] See (2.11) of Gutmann, 2001

Latin hypercube sampling[1]



The task is to generate n_0 samples from $\mathbf{U}[0,1]^D$. Let U_{ij} be i.i.d. samples from $U[0,1]$, and $\boldsymbol{\pi}$ is a uniform permutation of $\overline{0, n_0 - 1}$. Then

$$X_{ij} := \frac{\pi_j(i-1) + U_{ij}}{n_0}, \quad i = \overline{1, n_0},\ j = \overline{1, D}.$$

Theorem

Let X_{ij} be a Litin hypercube samples. Then $\mathbf{X}_i \sim \mathbf{U}[0,1]^D$ for each $i = \overline{1, n_0}$.

[1] https://artowen.su.domains/mc/Ch-var-adv.pdf

General algorithm

1. Start by drawing $n_0 = 2(D+1)$ samples, using Latin hypercube sampling.
2. While $n < N_{\max}$, maximum evaluation number, generate $m = 100D$ candidates $\mathbf{t}_{n,1:m}$. The probability of perturbing a coordinate:

$$\varphi_n = \min(20/D, 1)\left[1 - \frac{\log(n - n_0 + 1)}{\log(N_{\max} - n_0)}\right].$$

The additive perturbation $\delta_i \sim \mathcal{N}(0, \sigma_n^2)$, where $\sigma_{n_0}^2 = 0.2$, after each $\max(5, D)$ iteration with no improvements $\sigma_{n+1}^2 = \min(\sigma_n^2/2, 0.005)$. After 3 consecutive iterations with improvement $\sigma_{n+1}^2 = \min(0.2, 2\sigma_n^2)$.

General algorithm

3. Select most promising point $\mathbf{x}_{n+1}$ from $\mathbf{t}_n$. For each $1 \leq j \leq m$ define $\Delta(\mathbf{t}_{n,j}) = \min_{1 \leq i \leq n} \|\mathbf{t}_{n,j} - \mathbf{x}_i\|_2$. Let also $\Delta_{\max} = \max_j \Delta(\mathbf{t}_{n,j})$, $\Delta_{\min} = \min_j \Delta(\mathbf{t}_{n,j})$. We compute estimate value score:

$$V_{\mathrm{ev}}(\mathbf{t}_{n,j}) = \frac{S(\mathbf{t}_{n,j}) - S_{\min}}{S_{\max} - S_{\min}}.$$

And distance metric:

$$V_{\mathrm{dm}}(\mathbf{t}_{n,j}) = \frac{\Delta_{\max} - \Delta(\mathbf{t}_{n,j})}{\Delta_{\max} - \Delta_{\min}}.$$

The final score is $W(\mathbf{t}_{n,j}) = wV_{\mathrm{ev}}(\mathbf{t}_{n,j}) + (1-w)V_{\mathrm{dm}}(\mathbf{t}_{nj})$. Select the next point:

$$\mathbf{x}_{n+1} = \arg\min_{\mathbf{t} \in \mathbf{t}_{n,1:m}} W(\mathbf{t}).$$

Pseudocode

Algorithm 1 **H**yperparameter **O**ptimization using **R**BF-based surrogate and **D**YCORS (HORD)

input $n_0 = 2(D+1)$, $m = 100D$ and $N_{\max}$.
output optimal hyperparameters $\mathbf{x}_{\text{best}}$.

Use Latin hypercube sampling to sample n_0 points and set $\mathcal{I} = \{\mathbf{x}_i\}_{i=1}^{n_0}$.
Evaluating $f(x)$ for points in $\mathcal{I}$ gives $\mathcal{A}_{n_0} = \{(\mathbf{x}_i, f(\mathbf{x}_i))\}_{i=1}^{n_0}$.
while $n < N_{\max}$ **do**
  Use $\mathcal{A}_n$ to fit or update the surrogate model $S_n(\mathbf{x})$ (Eq. 2).
  Set $\mathbf{x}_{\text{best}} = \arg\min\{f(\mathbf{x}_i) : i = 1, \ldots, n\}$.
  Compute φ_n (Eq. 5), i.e, the probability of perturbing a coordinate.
  Populate Ω_n with m candidate points, $\mathbf{t}_{n,1:m}$, where for each candidate $\mathbf{y}_j \in \mathbf{t}_{n,1:m}$, (a) Set $\mathbf{y}_j = \mathbf{x}_{\text{best}}$, (b) Select the coordinates of $\mathbf{y}_j$ to be perturbed with probability φ_n and (c) Add δ_i sampled from $\mathcal{N}(0, \sigma_n^2)$ to the coordinates of $\mathbf{y}_j$ selected in (b) and round to nearest integer if required.
  Calculate $V_n^{ev}(\mathbf{t}_{n,1:m})$ (Eq. 6), $V_n^{dm}(\mathbf{t}_{n,1:m})$ (Eq. 7), and the final weighted score $W_n(\mathbf{t}_{n,1:m})$ (Eq. 8).
  Set $\mathbf{x}^* = \arg\min\{W_n(\mathbf{t}_{n,1:m})\}$.
  Evaluate $f(\mathbf{x}^*)$.
  Adjust the variance σ_n^2 (see text).
  Update $\mathcal{A}_{n+1} = \{\mathcal{A}_n \cup (\mathbf{x}^*, f(\mathbf{x}^*))\}$.
end while
Return $\mathbf{x}_{\text{best}}$.

Experimental setup

Experiments

1. **6-MLP**: 4 continuous and 2 integer hyperparameters.
2. **8-CNN**: 4 continuous and 4 integer hyperparameters.
3. **15-CNN**: 10 continuous and 5 integer hyperparameters.
4. **19-CNN**: 14 continuous and 5 integer hyperparameters.

Baselines

1. **GP-EI**: Gaussian processes with expected improvemen.
2. **GP-PES**: Gaussian processes with predictive entropy search
3. **TPE**: Tree Parzen Estimator
4. **SMAC**: Sequential Model-based Algorithm Configuration

Experimental Results

Data Set Problem	MNIST 6-MLP	MNIST 8-CNN	MNIST 15-CNN	CIFAR-10 19-CNN
GP-EI	78%(155)	73%(145)	18%(36)	17%(33)
GP-PES	16%(32)	50%(100)	10%(20)	—
TPE	38%(75)	50%(100)	29%(58)	25%(49)
SMAC	20%(39)	28%(55)	20%(40)	27%(54)

(a) Speed comparison.

Data Set Problem	MNIST 6-MLP	MNIST 8-CNN	MNIST 15-CNN	CIFAR-10 19-CNN
GP-EI	1.94(.11)	**0.77**(**.07**)	0.99(.11)	37.19(4.1)
GP-PES	1.94(.07)	0.87(.04)	1.06(.07)	—
TPE	2.00(.079)	0.96(.07)	0.97(.03)	27.13(3.2)
SMAC	2.13(.11)	0.85(.07)	1.10(.07)	29.74(2.1)
HORD	**1.87**(**.06**)	0.84(.04)	0.94(.07)	23.23(1.9)
HORD-ISP	—	—	**0.82**(**.05**)	**20.54**(**1.2**)

(b) Test error comparison.

We see that HORD (HORD-ISP) outperforms the baselines in terms of accuracy and speed.

References

- Gutmann H. M. (2001) A radial basis function method for global optimization, Journal of global optimization.
- https://artowen.su.domains/mc/Ch-var-adv.pdf
- Ilievski I. et al. (2017) Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates, Proceedings of the AAAI conference on artificial intelligence.