# Hyperparameter Optimization in Machine Learning

Bin Gu

bin.gu@mbzuai.ac.ae
Machine Learning Department, MBZUAI

June 19, 2021

# Outline

# Hyperparameter Optimization (HO)

## Hyperparamters: all the parameters which are not updated during the learning

- Problem-based hyperparameters:
  - Regularization parameter
  - Architectural hyperparameters in deep neural networks: depth and width of a deep neural network
- Algorithm-based hyperparameters:
  - Learning rate
  - Momentum
  - Dropout rate

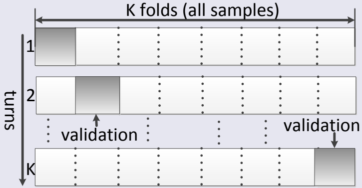
## Hyperparameter optimization

Hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.

# Hyperparameter Optimization

## Cross validation (CV)

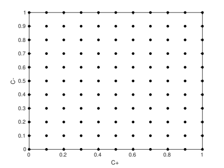
- $K$-fold




- Leave-one-out
- Repeated random sub-sampling procedures

# Hyperparameter Optimization

## Search strategy



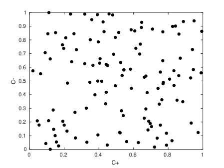




Figure: Grid (left) and random (right) search.

## Weakness of grid and random search strategies

- Only considering finite candidates due to the limited computing resources

# Hyperparameter Optimization (HO)

HO can be formulated as a bi-level optimization problem.

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^p} \quad & f(\lambda) = E(w(\lambda), \lambda), \\ s.t. \quad & w(\lambda) \in \mathrm{argmin}_{w \in \mathbb{R}^d} L(w, \lambda) \end{aligned} \tag{1}$$

- $w \in \mathbb{R}^d$ are the model parameters.
- $\lambda \in \mathbb{R}^p$ are the hyperparameters.
- $E$ represents a proxy of the generalization error w.r.t. the hyperparameters.
- $L$ represents traditional learning objective.
- $w(\lambda)$ are the optimal model parameters of $L$ for the fixed hyperparameters $\lambda$.

# Hyperparameter Optimization (HO)

HO can be formulated as a bi-level optimization problem.

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^p} \quad & f(\lambda) = E(w(\lambda), \lambda), \\ s.t. \quad & w(\lambda) \in \operatorname{argmin}_{w \in \mathbb{R}^d} L(w, \lambda) \end{aligned} \tag{1}$$

- $w \in \mathbb{R}^d$ are the model parameters.
- $\lambda \in \mathbb{R}^p$ are the hyperparameters.
- $E$ represents a proxy of the generalization error w.r.t. the hyperparameters.
- $L$ represents traditional learning objective.
- $w(\lambda)$ are the optimal model parameters of $L$ for the fixed hyperparameters $\lambda$.

The ultimate goal of HO is to find the values of hyperparameters with the minimum CV error in the whole parameter space.

# Our goals

## HO with global searching →

- Nonconvex
- Find the hyperparameters with the minimum CV error in the whole parameter space

## Practical HO

- Handle many hyperparameters
- Easy to use
- Flexible to various learning algorithms
- Efficient

# Global Optimal Hyperparameter Optimization

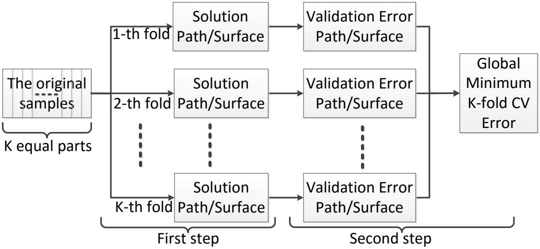



Figure: Cross validation with global searching.

**One hyperparameter**

- Solution path
- Error path

**Two hyperparameters**

- Solution surface
- Error surface

# Illustration of Solution path of Lasso

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 \quad s.t. \sum_{j=1}^{p} |\beta_j| \leq t. \tag{2}$$

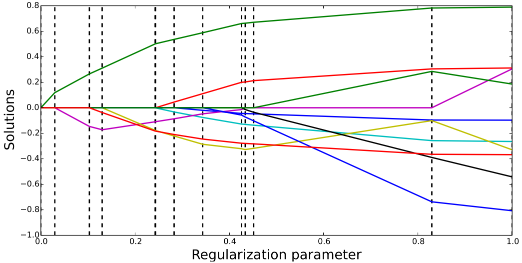
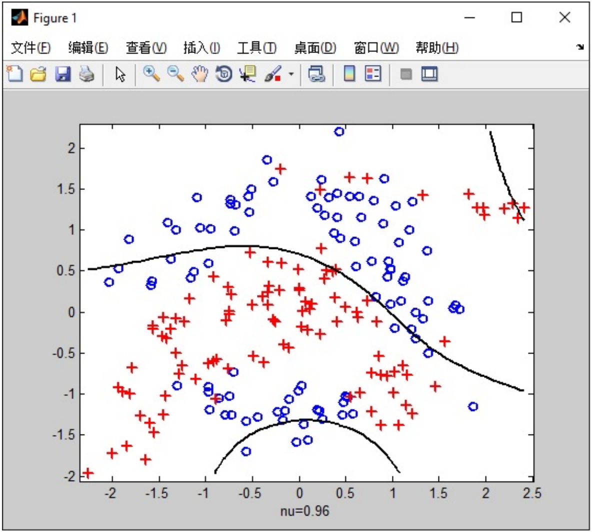



Figure: The solutions of Lasso with respect to the regularization parameter $t$.

- A finite number of representative solutions fit the entire solutions.[1]

[1] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," Ann. Statist., vol. 35, no. 3, pp. 1012-1030, 2007.

# Demo of solution path for $\nu$-SVC



# Solution path algorithms

Table: Representative solution path algorithms.

| Problem | Task | Reference | Parameter | Exact |
|---|---|---|---|---|
| $C$-SVC | BC | [Hastie et al.(2004)] | Regularization parameter $C$ | Yes |
| $2C$-SVC | BC | [Bach et al.(2006)] | Regularization parameters $C_+$, $C_-$ | Yes |
| $\varepsilon$-SVR | R | [Gunter & Zhu(2007)] | Regularization parameter | Yes |
| $\varepsilon$-SVR | R | [Wang et al.(2008)] | Regularization parameter and $\varepsilon$ | Yes |
| Lasso | R | [Rosset & Zhu(2007)] | Regularization parameter | Yes |
| KQR | R | [Takeuchi et al.(2009)] | Quantile order $\tau \in (0,1)$ | Yes |
| $C$-SVC | BC | [Ong et al.(2010)] | Regularization parameter $C$ | Yes |
| $C$-SVC | BC | [Karasuyama et al.(2011)] | Regularization parameter $C$ | No |
| $\nu$-SVC | BC | **[Gu et al.(2012,2016)]** | Regularization parameter $\nu$ | Yes |
| OSCAR | R | **[Gu et al.(2017)]** | Regularization parameters | No |
| General | BC+R | [Giesen et al.(2012)] | Regularization parameter | No |
| General | BC+R | **[Gu & Sheng.(2018)]** | Regularization parameter | Yes |

- Bin Gu, et al. Regularization Path for $\nu$-Support Vector Classification. IEEE Transactions on Neural Networks and Learning Systems, 23(5): 800-811,2012.
- Bin Gu, Victor S. Sheng. A Robust Regularization Path Algorithm for $\nu$-Support Vector Classification. IEEE Transactions on Neural Networks and Learning Systems, 2016.
- Bin Gu, Guodong Liu, Heng Huang. Groups-Keeping Solution Path Algorithm for Sparse Regression with Automatic Feature Grouping. KDD 2017.
- Bin Gu, Victor S. Sheng. A Solution Path Algorithm for General Parametric Quadratic Programming Problem. IEEE Transactions on Neural Networks and Learning Systems, 2018.

# Solution path algorithms

Table: Representative solution path algorithms.

| Problem | Task | Reference | Parameter | Exact |
|---|---|---|---|---|
| $C$-SVC | BC | [Hastie et al.(2004)] | Regularization parameter $C$ | Yes |
| $2C$-SVC | BC | [Bach et al.(2006)] | Regularization parameters $C_+$, $C_-$ | Yes |
| $\varepsilon$-SVR | R | [Gunter & Zhu(2007)] | Regularization parameter | Yes |
| $\varepsilon$-SVR | R | [Wang et al.(2008)] | Regularization parameter and $\varepsilon$ | Yes |
| Lasso | R | [Rosset & Zhu(2007)] | Regularization parameter | Yes |
| KQR | R | [Takeuchi et al.(2009)] | Quantile order $\tau \in (0,1)$ | Yes |
| $C$-SVC | BC | [Ong et al.(2010)] | Regularization parameter $C$ | Yes |
| $C$-SVC | BC | [Karasuyama et al.(2011)] | Regularization parameter $C$ | No |
| $\nu$-SVC | BC | **[Gu et al.(2012,2016)]** | Regularization parameter $\nu$ | Yes |
| OSCAR | R | **[Gu et al.(2017)]** | Regularization parameters | No |
| General | BC+R | [Giesen et al.(2012)] | Regularization parameter | No |
| General | BC+R | **[Gu & Sheng.(2018)]** | Regularization parameter | Yes |

- Bin Gu, et al. Regularization Path for $\nu$-Support Vector Classification. IEEE Transactions on Neural Networks and Learning Systems, 23(5): 800-811,2012.
- Bin Gu, Victor S. Sheng. A Robust Regularization Path Algorithm for $\nu$-Support Vector Classification. IEEE Transactions on Neural Networks and Learning Systems, 2016.
- Bin Gu, Guodong Liu, Heng Huang. Groups-Keeping Solution Path Algorithm for Sparse Regression with Automatic Feature Grouping. KDD 2017.
- Bin Gu, Victor S. Sheng. A Solution Path Algorithm for General Parametric Quadratic Programming Problem. IEEE Transactions on Neural Networks and Learning Systems, 2018.

## Octagonal shrinkage and clustering algorithm for regression (OSCAR)

- Given a training set $S = \{(x_i, y_i)\}_{i=1}^{l}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- $y_i$ is centered, *i.e.*, $\sum_{i=1}^{l} y_i = 0$
- Each feature of the training set $S$ is standardized, *i.e.*, $\sum_{i=1}^{l} x_{ij} = 0$ and $\sum_{i=1}^{l} x_{ij}^2 = 1$

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{l} \left(y_i - x_i^T \beta\right)^2 \quad (3)$$

$$+ \boxed{\lambda_1} \|\beta\|_1 + \boxed{\lambda_2} \sum_{i<j} \max\{|\beta_i|, |\beta_j|\},$$

- $\lambda_2 = 0$: Lasso.
- $\lambda_2 = \infty$: Clustering all features as a group.

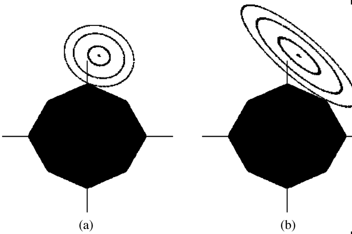



Figure: Illustration of OSCAR

## Octagonal shrinkage and clustering algorithm for regression (OSCAR)

- Given a training set $S = \{(x_i, y_i)\}_{i=1}^{l}$ with $x_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$
- $y_i$ is centered, *i.e.*, $\sum_{i=1}^{l} y_i = 0$
- Each feature of the training set $S$ is standardized, *i.e.*, $\sum_{i=1}^{l} x_{ij} = 0$ and $\sum_{i=1}^{l} x_{ij}^2 = 1$

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{l} \left(y_i - x_i^T \beta\right)^2 \quad (3)$$

$$+ \boxed{\lambda_1} \|\beta\|_1 + \boxed{\lambda_2} \sum_{i<j} \max\{|\beta_i|, |\beta_j|\},$$

- $\lambda_2 = 0$: Lasso.
- $\lambda_2 = \infty$: Clustering all features as a group.

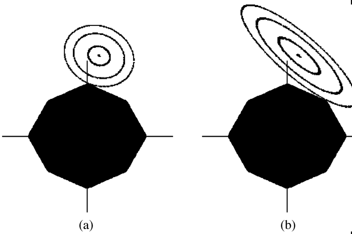



Figure: Illustration of OSCAR

Tuning the values of $\lambda_1$ and $\lambda_2$ plays an essential role for OSCAR!

# Optimality Conditions of OSCAR Model

## Definition (Feature group $\mathcal{G}_g$)

Given the orders $o(j)$ of $|\beta_j|$. The set $\mathcal{G}_g \subseteq \{1, 2, \cdots, d\}$ is called a group of features if the following conditions are satisfied.

1. $\forall j_1, j_2 \in \mathcal{G}_g$, and $j_1 \neq j_2$, we have $|\beta_{j_1}| = |\beta_{j_2}| \stackrel{\text{def}}{=} \theta_g$.
2. If $j \in \{1, 2, \cdots, d\}$ and $j \notin \mathcal{G}_g$, we have that $|\beta_j| \neq \theta_g$.

## New formulation of OSCAR free of the $\ell_1$-norm and the pair $\ell_\infty$-norm

$$\min_{\theta} \frac{1}{2} \sum_{i=1}^{l} \left(y_i - \widetilde{x}_i^T \theta\right)^2 + \sum_{g=1}^{G} w_g \theta_g \quad s.t. \qquad 0 \leq \theta_1 < \theta_2 < \cdots < \theta_G ,$$

- $\widetilde{x}_i = [\widetilde{x}_{i1} \ \widetilde{x}_{i2} \cdots \widetilde{x}_{iG}]$ and $\widetilde{x}_{ig} = \sum_{j \in \mathcal{G}_g} \text{sign}(\beta_j) x_{ij}$.
- $w_g = \sum_{j \in \mathcal{G}_g} (\lambda_1 + (o(j) - 1)\lambda_2)$

# Groups-keeping Solution Path Algorithm

**$\Delta\eta$ is a parameter to control the adjustment qualities of $\lambda_1$ and $\lambda_2$**

- Define $\Delta\lambda = \begin{bmatrix} \Delta\lambda_1 \\ \Delta\lambda_2 \end{bmatrix}$
- $\Delta\lambda = d\Delta\eta$, where $d = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$.

**Three main steps of OscarGKPath**

1. Computing the directions of $\Delta\theta$;
2. Computing the maximum adjustment of $\Delta\eta$;
3. And computing the duality gap $G(\theta, \lambda_1, \lambda_2)$.

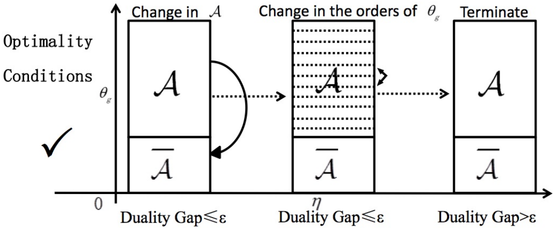



Figure: The fundamental principle of OscarGKPath.

# Global Optimal Hyperparameter Optimization

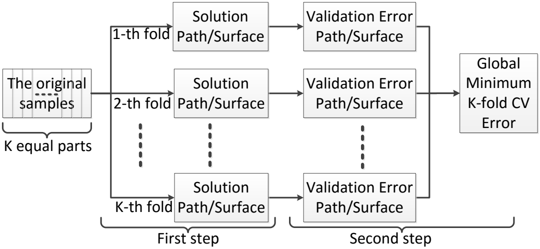



Figure: Cross validation with global searching.

**One hyperparameter**

- Solution path
- Error path

**Two hyperparameters**

- Solution surface
- Error surface

# The piecewise linearity of the solution path

## Definition

Suppose $\tilde{\alpha}(\lambda)$ is returned by a solution path. The solution $\tilde{\alpha}(\lambda)$ is called piecewise linear as a function of $\lambda$, if existing $a = a_0 < a_1 < a_2 < \cdots < a_m = b$, and the corresponding vectors $\beta^{[1]}, \beta^{[2]}, \cdots, \beta^{[m]}$, such that the solution $\hat{\alpha}(\lambda)$ is given exactly or approximately, by $\tilde{\alpha}(a_{k-1}) + \beta^{[k]}(\lambda - a_{k-1})$, $\forall \lambda \in [a_{k-1}, a_k]$.

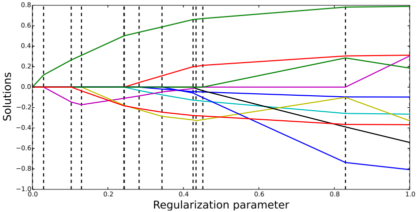



Figure: The solutions with respect to the regularization parameter.

# Generalized Error Path

- Bin Gu,Charles X. Ling. A New Generalized Error Path Algorithm for Model Selection. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15) (pp. 2549-2558)

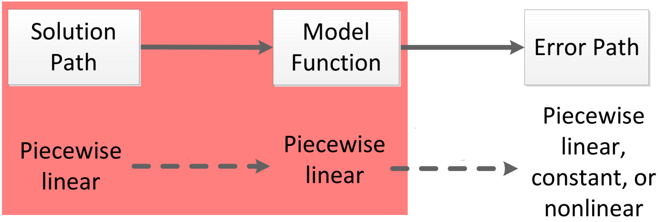



Figure: Model function builds a bridge from solution path to error path.

# Solution Path → Model Function

**A model with a linear representation $f(x) = \langle G(x), \alpha \rangle$**

1. For $C$-SVC, $2C$-SVC, $\nu$-SVC, the models are $f(x) = \sum_i^l \alpha_i y_i K(x, x_i) + \alpha_0$.
2. For $\varepsilon$-SVR, the model is $f(x) = \sum_i^{2l} \alpha_i z_i K(x, x_i) + \alpha_0$.
3. For Lasso, the model is $f(x) = x^T \alpha$.
4. For KQR, the model is $f(x) = \sum_i^l \alpha_i K(x, x_i) + \alpha_0$.

# Solution Path → Model Function

## Lemma

*Given an interval $[a_{k-1}, a_k]$ in a solution path, and the corresponding vector $\beta^{[k]}$. If the model is with linear representation, there exists a scale $\gamma^{[k]}$ such that the model function $f_\lambda(x)$ can be represented as*
$f_\lambda(x) = f_{a_{k-1}}(x) + \gamma^{[k]}(\lambda - a_{k-1})$, $\forall \lambda \in [a_{k-1}, a_k]$.

# Solution Path → Model Function

## Lemma

*Given an interval $[a_{k-1}, a_k]$ in a solution path, and the corresponding vector $\beta^{[k]}$. If the model is with linear representation, there exists a scale $\gamma^{[k]}$ such that the model function $f_\lambda(x)$ can be represented as*
$f_\lambda(x) = f_{a_{k-1}}(x) + \gamma^{[k]}(\lambda - a_{k-1}), \forall\lambda \in [a_{k-1}, a_k]$.

1. For $C$-SVC, $2C$-SVC, and $\nu$-SVC, $\gamma = \sum_i^l \beta_i y_i K(x, x_i) + \beta_0$.
2. For $\varepsilon$-SVR, $\gamma = \sum_i^{2l} \beta_i z_i K(x, x_i) + \beta_0$.
3. For Lasso, $\gamma = x^T \beta$.
4. For KQR, $\gamma = \sum_i^l \beta_i K(x, x_i) + \beta_0$.

# Generalized Error Path

- Bin Gu,Charles X. Ling. A New Generalized Error Path Algorithm for Model Selection. In Proceedings of the 32nd International Conference on Machine Learning (ICML-15) (pp. 2549-2558)

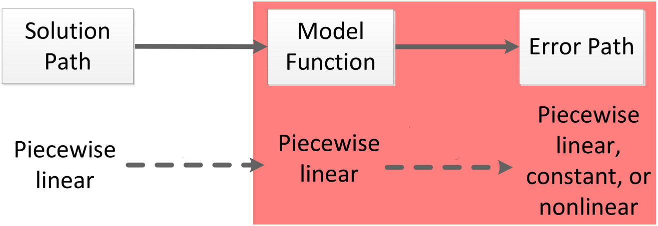



Figure: Model function builds a bridge from solution path to error path.

# Model Function → Error Path

The error path on the entire interval $[a, b]$ could be

1. piecewise quadratic,
2. piecewise linear,
3. or piecewise constant

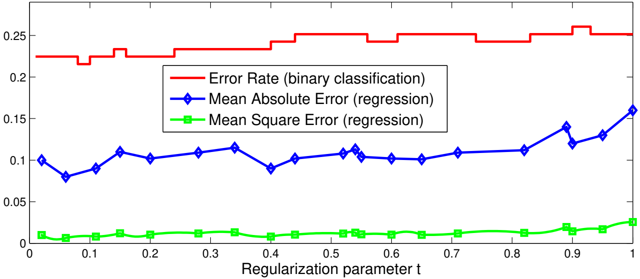



Figure: The error path w.r.t. the regularization parameter.

# Computing Generalized Error Path

To fit various error paths, we define a 2-tuple $(I, P)$

1. For piecewise quadratic error path:
   $I = [a_{k-1}, a_k]$, $P.c_2 = \frac{1}{\ell} \sum_{i=1}^{\ell} (\gamma_i^{[k]})^2$,
   $P.c_1 = -\frac{1}{\ell} \sum_{i=1}^{\ell} 2 \times \gamma_i^{[k]} (\widetilde{y}_i - f_{a_{k-1}}(\widetilde{x}_i) + \gamma_i^{[k]} a_{k-1})$,
   $P.c_0 = \frac{1}{\ell} \sum_{i=1}^{\ell} \left( \widetilde{y}_i - f_{a_{k-1}}(\widetilde{x}_i) + \gamma_i^{[k]} a_{k-1} \right)^2$.
2. For piecewise linear error path:
   $I = \mathcal{IR}(\lambda_0)$,
   $P.c_1 = \frac{1}{\ell} \left( \sum_{i \in \mathcal{I}_-(\lambda_0)} \gamma_i^{[k]} - \sum_{i \in \mathcal{I}_+(\lambda_0)} \gamma_i^{[k]} \right)$,
   $P.c_0 = \frac{1}{\ell} (\sum_{i \in \mathcal{I}_-(\lambda_0)} (\widetilde{y}_i - f_{\lambda_0}(\widetilde{x}_i) + \gamma_k \lambda_0) - \sum_{i \in \mathcal{I}_+(\lambda_0)} (\widetilde{y}_i - f_{\lambda_0}(\widetilde{x}_i) + \gamma_k \lambda_0))$.
3. For piecewise constant error path:
   $I = \widetilde{\mathcal{IR}}(\lambda_0)$, $P.c_0 = E(\hat{\alpha}(\lambda_0), \mathcal{V}, L)$.

# GEP (Generalized error path algorithm)

**Input:** A solution path w.r.t. $\lambda$ in an interval $[a,b] = \bigcup_{k=1}^{m} [a_{k-1}, a_k]$, a validation set $\mathcal{V}$.

**Output:** An error path $\{(I_1, P_1), (I_2, P_2), (I_3, P_3), \cdots\}$.

1: Initialize $\lambda = a$, $k = 0$, $j = 1$.
2: **while** $\lambda < b$ **do**
3:     Update $k = k + 1$, read the $k$-th sub-interval $[a_{k-1}, a_k]$ from the solution path.
4:     **while** $\lambda < a_k$ **do**
5:         Compute $(I_j, P_j)$ for the leftmost in $[\lambda, a_k]$.
6:         Update $\lambda = R(I_j)$, $j = j + 1$.
7:     **end while**
8: **end while**

Table: The results of CV error obtained from GS, MS, RS, and our GEP.

| Dataset | ER | | | | Dataset | MAE | | | | MSE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | MS | RS | GEP | | GS | MS | RS | GEP | GS | MS | RS | GEP |
| Ionosphere | 0.062 | 0.068 | 0.068 | **0.060** | Friedman | 2.08 | 2.1 | 2.07 | **1.98** | 6.81 | 6.82 | 6.82 | **6.60** |
| Diabetes | 0.655 | 0.655 | 0.565 | **0.345** | Housing | 2.12 | 2.13 | 2.13 | **2.09** | 8.11 | 8.14 | 8.14 | **8.05** |
| Hill-Valley | 0.465 | 0.470 | 0.468 | **0.460** | Forest Fires | 17.9 | 18.3 | 18.2 | **17.2** | 3251 | 3258 | 3263 | **3238** |
| Breast Cancer | 0.528 | 0.528 | 0.474 | **0.342** | Auto MPG | 2.39 | 2.41 | 2.39 | **2.36** | 9.62 | 9.63 | 9.62 | **9.58** |
| Spine | 0.060 | 0.058 | 0.059 | **0.055** | Triazines | 2.39 | 2.40 | 2.40 | **2.37** | 9.66 | 9.67 | 9.66 | **9.63** |

Table: The results of CV error obtained from GS, MS, RS, and our GEP.

| Dataset | ER | | | | Dataset | MAE | | | | MSE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | MS | RS | GEP | | GS | MS | RS | GEP | GS | MS | RS | GEP |
| Ionosphere | 0.062 | 0.068 | 0.068 | **0.060** | Friedman | 2.08 | 2.1 | 2.07 | **1.98** | 6.81 | 6.82 | 6.82 | **6.60** |
| Diabetes | 0.655 | 0.655 | 0.565 | **0.345** | Housing | 2.12 | 2.13 | 2.13 | **2.09** | 8.11 | 8.14 | 8.14 | **8.05** |
| Hill-Valley | 0.465 | 0.470 | 0.468 | **0.460** | Forest Fires | 17.9 | 18.3 | 18.2 | **17.2** | 3251 | 3258 | 3263 | **3238** |
| Breast Cancer | 0.528 | 0.528 | 0.474 | **0.342** | Auto MPG | 2.39 | 2.41 | 2.39 | **2.36** | 9.62 | 9.63 | 9.62 | **9.58** |
| Spine | 0.060 | 0.058 | 0.059 | **0.055** | Triazines | 2.39 | 2.40 | 2.40 | **2.37** | 9.66 | 9.67 | 9.66 | **9.63** |

Table: The average errors on the test data, over 10 trails, obtained from GS, MS, RS, and our GEP.

| Dataset | ER | | | | Dataset | MAE | | | | MSE | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | GS | MS | RS | GEP | | GS | MS | RS | GEP | GS | MS | RS | GEP |
| Ionosphere | 0.066 | 0.066 | 0.065 | **0.063** | Friedman | 2.12 | 2.16 | 2.14 | **2.08** | 6.84 | 6.83 | 6.84 | **6.75** |
| Diabetes | 0.356 | 0.356 | 0.358 | **0.347** | Housing | 2.05 | 2.06 | 2.06 | **2.05** | 8.56 | 8.63 | 8.66 | **8.44** |
| Hill-Valley | 0.514 | 0.519 | 0.514 | **0.512** | Forest Fires | 18.4 | 19.1 | 18.9 | **17.9** | 5468 | 5475 | 5471 | **5446** |
| Breast Cancer | 0.528 | 0.528 | 0.474 | **0.342** | Auto MPG | 2.74 | 2.75 | 2.76 | **2.62** | 14.6 | 14.6 | 14.7 | **14.0** |
| Spine | 0.060 | 0.061 | 0.061 | **0.058** | Triazines | 2.76 | 2.77 | 2.77 | **2.54** | 14.7 | 14.8 | 14.6 | **14.2** |

# Global Optimal Hyperparameter Optimization

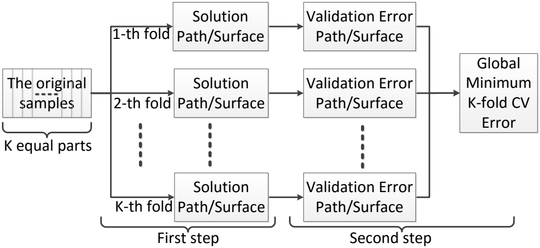



Figure: Cross validation with global searching.

**One hyperparameter**

- Solution path
- Error path

**Two hyperparameters**

- Solution surface
- Error surface

# Two Hyperparameters: Cost-Sensitive SVM (CS-SVM/$2C$-SVM)

## Cost Sensitive Learning

- $\mathcal{S}^+ = \{(x_i, y_i) : y_i = +1\}$, $\mathcal{S}^- = \{(x_i, y_i) : y_i = -1\}$
- $C(-,+) \neq C(+,-)$

## CS-SVM/2C-SVM

$$\min_{w,b,\xi} \quad \frac{1}{2}\langle w, w\rangle + C_+ \sum_{i\in\mathcal{S}^+} \xi_i + C_- \sum_{i\in\mathcal{S}^-} \xi_i \tag{4}$$

$$s.t. \quad y_i\left(\langle w, \phi(x_i)\rangle + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, l$$

- $C_+$ denotes the cost of false negative
- $C_-$ denotes the cost of false positive

# Two Hyperparameters: Cost-Sensitive SVM (CS-SVM/$2C$-SVM)

## Cost Sensitive Learning

- $\mathcal{S}^+ = \{(x_i, y_i) : y_i = +1\}$, $\mathcal{S}^- = \{(x_i, y_i) : y_i = -1\}$
- $C(-,+) \neq C(+,-)$

## CS-SVM/2C-SVM

$$\min_{w,b,\xi} \quad \frac{1}{2}\langle w, w\rangle + C_+ \sum_{i\in\mathcal{S}^+} \xi_i + C_- \sum_{i\in\mathcal{S}^-} \xi_i \tag{4}$$

$$s.t. \quad y_i\left(\langle w, \phi(x_i)\rangle + b\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \cdots, l$$

- $C_+$ denotes the cost of false negative
- $C_-$ denotes the cost of false positive

Tuning $(C_+, C_-)$ is a central problem of CS-SVM

# Motivation

## Weakness of Existing Methods

- Only considering one parameter
  - Solution path
  - Error path
- However, CS-SVM has two regularization parameters $C_+$ and $C_-$

# Motivation

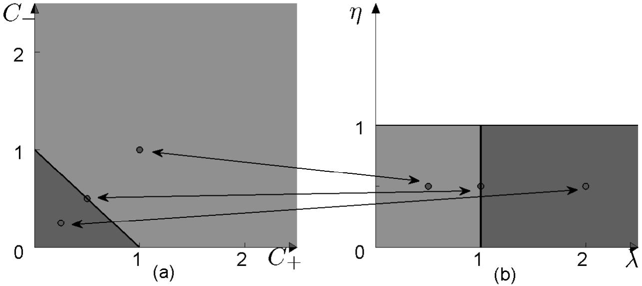
## Weakness of Existing Methods

- Only considering one parameter
    - Solution path
    - Error path
- However, CS-SVM has two regularization parameters $C_+$ and $C_-$

## Our work

- Bin Gu, Victor S. Sheng, Keng Yeow Tay, Walter Romano, and Shuo Li. Cross Validation Through Two-dimensional Solution Surface for Cost-Sensitive SVM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016.(online, long version)
- Bin Gu, Victor S. Sheng, Shuo Li. Bi-parameter Space Partition for Cost-Sensitive SVM. IJCAI 2015 (short version).

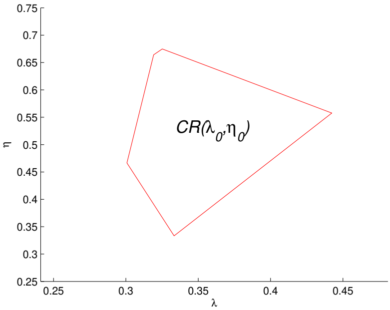
# $2C$-SVM $\leftrightarrow$ $(\lambda, \eta)$-SVM




- $\lambda = \frac{1}{C_+ + C_-}$
- $\eta = \frac{C_+}{C_+ + C_-}$

Exploring in 1.5 square units

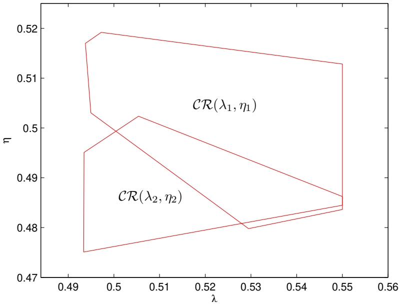
# Critical Region




## Theorem

*The set $\mathcal{CR}(\lambda_0, \eta_0)$ is a convex set and its closure is a convex polygon region.*

# Overlapped Critical Regions



# Partitioning the Parameter Space

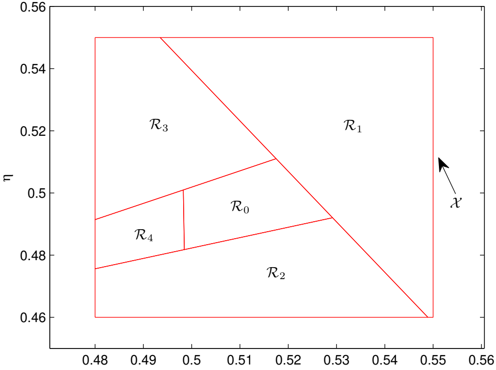
## Theorem

*Let $\mathcal{X} \subseteq \mathbb{R}^2$ be a convex polygon region, and $\mathcal{R}_0 = \{(\rho, \varrho) \in \mathcal{X} : A\,[\rho \;\; \varrho]^T \leq b\}$ be a convex polygon subregion of $\mathcal{X}$, where $A \in \mathbb{R}^{m\times 2}$, $b \in \mathbb{R}^{m\times 1}$, $\mathcal{R}_0 \neq \emptyset$. Also let*

$$\mathcal{R}_i \;=\; \left\{ (\rho, \varrho) \in \mathcal{X} \;\middle|\; \begin{array}{l} A_i\,[\rho \;\; \varrho]^T > b_i \\ A_j\,[\rho \;\; \varrho]^T \leq b_j, \quad \forall j < i \end{array} \right\}, \quad \forall i = 1, \cdots, m$$

*then $\{\mathcal{R}_0, \mathcal{R}_1, \cdots, \mathcal{R}_m\}$ is a partition of $\mathcal{X}$, i.e., $\bigcup_{i=0}^{m} \mathcal{R}_i = \mathcal{X}$, and $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, $\forall i \neq j$, $i, j \in \{0, 1, \cdots, m\}$.*

# Partitioning the Parameter Space based on $\mathcal{CR}$s



# Partitioning the Parameter Space

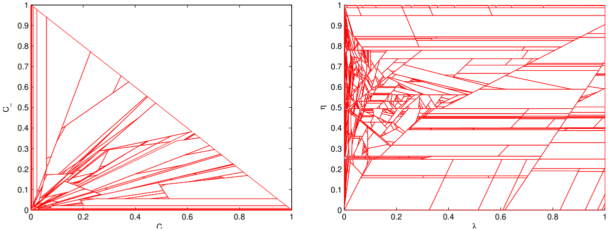



Figure: **Left**: Partitioning the lower triangle region of $[0,1] \times [0,1]$ for $(C_+, C_-)$ through $\mathcal{CR}$s. **Right**: Partitioning $(0,1] \times [0,1]$ for $(\lambda,\ \eta)$ through $\mathcal{CR}$s.

# Invariant Region

Given a validation set $\mathcal{V} = \{(\widetilde{x}_1, \widetilde{y}_1), \cdots, (\widetilde{x}_n, \widetilde{y}_n)\}$

$$\begin{aligned} \widetilde{\pi}(\rho, \varrho) &= \{\{i \in \mathcal{V} : f(\rho, \varrho)(\widetilde{x}_i)) \geq 0\}, \{i \in \mathcal{V} : f(\rho, \varrho)(\widetilde{x}_i)) < 0\}\} \\ &\stackrel{\text{def}}{=} \{\mathcal{I}_+(\rho, \varrho), \mathcal{I}_-(\rho, \varrho)\} \end{aligned} \tag{5}$$

$$\mathcal{IR}(\rho_0, \varrho_0) = \{(\rho, \varrho) \in \mathcal{CR}(\rho_0, \varrho_0) : \widetilde{\pi}(\rho, \varrho) = \widetilde{\pi}(\rho_0, \varrho_0)\}$$

# Invariant Region

Given a validation set $\mathcal{V} = \{(\widetilde{x}_1, \widetilde{y}_1), \cdots, (\widetilde{x}_n, \widetilde{y}_n)\}$

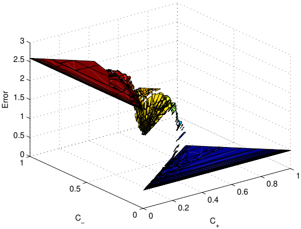
$$\begin{aligned} \widetilde{\pi}(\rho, \varrho) &= \{\{i \in \mathcal{V} : f(\rho, \varrho)(\widetilde{x}_i)) \geq 0\}, \{i \in \mathcal{V} : f(\rho, \varrho)(\widetilde{x}_i)) < 0\}\} \\ &\stackrel{\text{def}}{=} \{\mathcal{I}_+(\rho, \varrho), \mathcal{I}_-(\rho, \varrho)\} \end{aligned} \tag{5}$$

$$\mathcal{IR}(\rho_0, \varrho_0) = \{(\rho, \varrho) \in \mathcal{CR}(\rho_0, \varrho_0) : \widetilde{\pi}(\rho, \varrho) = \widetilde{\pi}(\rho_0, \varrho_0)\}$$

**Theorem**

*The sets $\mathcal{IR}(\rho_0, \varrho_0)$ is a convex set and its closure is a convex polygon region.*

# Partitioning the Parameter Space using $\mathcal{IR}$s



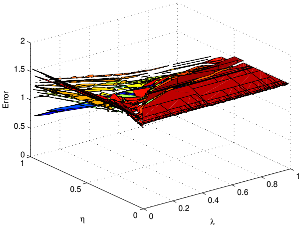




Figure: Validation error. **Left**: All parameter pairs of $(C_+, C_-)$ in the lower triangle region of $[0,1] \times [0,1]$. **Right**: All parameter pairs of $(\lambda, \eta)$ in $(0,1] \times [0,1]$.

# The results of $5$-fold CV.

| $C(+,-)$ | Dataset | GS | | SP$_\eta$+GS$_\lambda$ | | SP$_\lambda$+GS$_\eta$ | | BPSP | |
|---|---|---|---|---|---|---|---|---|---|
| | | CV error | time | CV error | time | CV error | time | CV error | time |
| 2 | Son | 0.4667 | 43 | 0.282 | 7.7 | 0.271 | 7.3 | **0.2564** | **4.4** |
| | Ion | 0.3623 | 73 | 0.0725 | 12.7 | 0.0857 | 13.3 | **0.0435** | **9.7** |
| | Dia | 0.6275 | 294 | 0.5948 | 9.5 | 0.606 | 10.2 | **0.5752** | **5.5** |
| | BC | 0.6593 | 229 | 0.6 | 9 | 0.611 | 9.82 | **0.5642** | **7.1** |
| | Hea | 0.52 | 59 | 0.464 | 9.2 | 0.478 | 9.1 | **0.444** | **5.9** |
| | HV | 0.463 | 176 | 0.45 | 5.3 | 0.462 | 5.8 | **0.4417** | **4.9** |
| | SI | 0.5017 | 86 | 0.2754 | 7.4 | 0.278 | 8.2 | **0.2650** | **4.9** |
| 5 | Son | 0.4872 | 51 | 0.3167 | 7 | 0.322 | 7.3 | **0.3167** | **4.6** |
| | Ion | 0.3768 | 65 | 0.1159 | 14 | 0.1324 | 16.3 | **0.1159** | **10.3** |
| | Dia | 0.6536 | 302 | 0.632 | 9.6 | 0.638 | 10.1 | **0.632** | **5.9** |
| | BC | 0.6741 | 227 | 0.6074 | 8 | 0.6222 | 8.3 | **0.6074** | **6.7** |
| | Hea | 0.537 | 57 | 0.463 | 6.8 | 0.485 | 8.5 | **0.463** | **5.5** |
| | HV | 0.6 | 164 | 0.55 | 5.6 | 0.493 | 7.2 | **0.4417** | **5.2** |
| | SI | 0.524 | 77 | 0.383 | 8.1 | 0.3795 | 8.5 | **0.3562** | **5.6** |
| 10 | Son | 0.564 | 46 | 0.4615 | 6.4 | 0.473 | 6.8 | **0.4359** | **4.9** |
| | Ion | 0.3823 | 77 | 0.2319 | 15.3 | 0.2425 | 16.1 | **0.2319** | **9.9** |
| | Dia | 0.6863 | 312 | 0.6601 | 9.2 | 0.672 | 9.8 | **0.6601** | **5.6** |
| | BC | 0.6815 | 219 | 0.6741 | 7.3 | 0.6741 | 7.2 | **0.6626** | **6.9** |
| | Hea | 0.556 | 69 | 0.556 | 5.6 | 0.562 | 5.9 | **0.556** | **5.4** |
| | HV | 0.5 | 169 | 0.5 | 4.9 | 0.5 | 6.3 | **0.458** | **4.7** |
| | SI | 0.536 | 81 | 0.4783 | 7.8 | 0.464 | 8.3 | **0.4493** | **5.1** |
| ratio | Ecoli1 | 0.1722 | 65 | 0.117 | 12.2 | 0.124 | 13.1 | **0.0833** | **8.8** |
| | Ecoli3 | 0.1905 | 76 | 0.0909 | 11.6 | 0.1102 | 12.3 | **0.0595** | **9.3** |
| | Vowel0 | 0.1586 | 195 | 0.101 | 103 | 0.095 | 89 | **0.0449** | **21** |
| | Vehicle0 | 0.472 | 262 | 0.1834 | 16d5 | 0.2092 | 134 | **0.1024** | **26** |

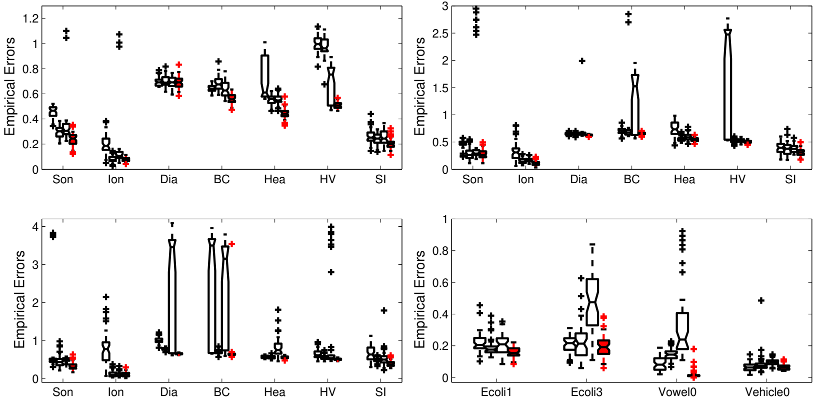



Figure: The results of cost sensitive errors on the test sets. (a): $C(-,+) = 2$. (b): $C(-,+) = 5$. (c): $C(-,+) = 10$. (d): $C(-,+) = \text{ratio}$, for imbalanced learning.

# 5 Criteria to estimate HO methods

- "effective": good generalization performance.
- "efficient": running fast.
- "scalable": scalable in terms of the sizes of hyperparameters and model parameters.
- "simple": it can be implemented easily.
- "flexible": flexible to various learning algorithms.

# Bayesian optimization algorithms

BO is an strategy to transform the problem

$$x_M = \arg\min_{x \in \mathcal{X}} \quad f(x) \tag{6}$$

into a series of problems:

$$x_{n+1} = \arg\min_{x \in \mathcal{X}} \quad \alpha(x; \mathcal{D}_n, \mathcal{M}_n) \tag{7}$$

where $\alpha(x; \mathcal{D}_n, \mathcal{M}_n)$ is inexpensive to evaluate.

- Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. NIPS
- Falkner, Stefan, Aaron Klein, and Frank Hutter. "BOHB: Robust and efficient hyperparameter optimization at scale." ICML, 2018.
- K. Kandasamy, K. Vysyaraju, W. Neiswanger, B. Paria, C. Collins, J. Schneider, B. Poczos, and E. P. Xing. Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly., JMLR, 21 (81), 1-27, 2020

Commercial or open source systems: Microsoft Azure, Google Cloud AutoML solution, Dragonfly, auto-sklearn, ...

# Gradient-based algorithms: Back-propagation

Replace the inner problem with a dynamical system:

$$\begin{aligned} \min_{\lambda} \quad & E(s_T) \\ s.t. \quad & s_t = \Phi_t(s_{t-1}, \lambda), \quad t \in \{1, \ldots, T\}. \end{aligned} \tag{8}$$

According to chain rule, and similarly to Back-propagation, we have

$$\frac{\partial E(s_T)}{\partial \lambda} = \nabla E(s_T) \sum_{t=1}^{T} (A_{t+1} \ldots A_T) B_t \tag{9}$$

$$A_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial s_{t-1}}, \quad B_t = \frac{\partial \Phi_t(s_{t-1}, \lambda)}{\partial \lambda}$$

- Maclaurin, Dougal, David Duvenaud, and Ryan Adams. "Gradient-based hyperparameter optimization through reversible learning." ICML, 2015.
- Franceschi, Luca, et al. "Bilevel programming for hyperparameter optimization and meta-learning." ICML, 2018.

# Gradient-based algorithms: Penalty method

- Replace the inner problem with its first order necessary condition.

$$\min_{u,v} f(u,v), \ \ s.t. \ c(u,v) = \nabla_v g(u,v) = 0 \tag{10}$$

- The augmented Lagrangian function
$\mathcal{L}(u,v,z;\mu_k) = f(u,v) + \frac{1}{d}\sum_{j=1}^{d}\left(z_j c_j(u,v) + \frac{\mu_k}{2}c_j^2(u,v)\right).$
- Randomly sample a constraint and the validation data to calculate stochastic gradient of the augmented Lagrangian function.

$$\nabla_v \mathcal{L} = \left[\mu_k c_j(u,v^t) + z_j^t\right] \nabla_v c_j(u,v^t) + \tilde{\nabla}_v f(u,v^t) \tag{11}$$

$$\nabla_u \mathcal{L} = \left[\mu_k c_j(u^k,v) + z_j^k\right] \nabla_u c_j(u^k,v) + \tilde{\nabla}_u f(u^k,v)$$

- Wanli Shi, Bin Gu, Heng Huang. Improved Penalty Method via Doubly Stochastic Gradients for Bilevel Hyperparameter Optimization. AAAI 2021.

# Representative HO algorithms

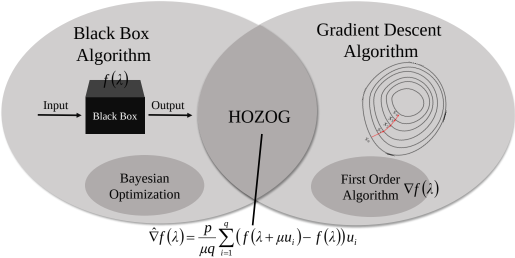
Table: Representative black-box optimization and gradient based hyperparameter optimization algorithms.

| Algorithm | Type | Reference | Properties | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Effective | Efficient | Scalable-H | Simple | Flexible | Scalable-P |
| GPBO | BB | Snoek et al. (2012) | ♣ | ♣ | ✗ | ✓ | ✓ | ✓ |
| BOHB | BB | Falkner et al. (2018) | ♣ | ♣ | ✗ | ✓ | ✓ | ✓ |
| HOAG | G | Pedregosa (2016) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| RMD | G | Maclaurin et al. (2015) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| RFHO | G | Franceschi et al. (2018) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| HOZOG | BB+G | Our | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

# Representative HO algorithms

Table: Representative black-box optimization and gradient based hyperparameter optimization algorithms.

| Algorithm | Type | Reference | Properties | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Effective | Efficient | Scalable-H | Simple | Flexible | Scalable-P |
| GPBO | BB | Snoek et al. (2012) | ♣ | ♣ | ✗ | ✓ | ✓ | ✓ |
| BOHB | BB | Falkner et al. (2018) | ♣ | ♣ | ✗ | ✓ | ✓ | ✓ |
| HOAG | G | Pedregosa (2016) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| RMD | G | Maclaurin et al. (2015) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| RFHO | G | Franceschi et al. (2018) | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| HOZOG | BB+G | Our | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |



# Our method: HOZOG

**Enlightenment:** We hope that the hyperparameter optimization method bases on zeroth-order hyper-gradients can inherit all benefits.

$$\hat{\nabla} f(\lambda) = \frac{p}{\mu q} \sum_{i=1}^{q} \left( f(\lambda + \mu u_i) - f(\lambda) \right) u_i \tag{12}$$

- $\mu > 0$ is the smoothing paramete.
- $\{u_i\}$ are the random query directions drawn from the standard Gaussian distribution.
- $q$ is the number of sampled query directions.

# Our method: HOZOG

**Enlightenment:** We hope that the hyperparameter optimization method bases on zeroth-order hyper-gradients can inherit all benefits.

$$\hat{\nabla} f(\lambda) = \frac{p}{\mu q} \sum_{i=1}^{q} \left( f(\lambda + \mu u_i) - f(\lambda) \right) u_i \tag{12}$$

- $\mu > 0$ is the smoothing paramete.
- $\{u_i\}$ are the random query directions drawn from the standard Gaussian distribution.
- $q$ is the number of sampled query directions.
- If $\mu \to 0$, $\boxed{\frac{1}{\mu}\left(f(\lambda + \mu u) - f(\lambda)\right)} \to \boxed{f'(\lambda, u)}$, a directional derivative of function $f(\lambda)$ along $u$.

# Our method: HOZOG

**Enlightenment:** We hope that the hyperparameter optimization method bases on zeroth-order hyper-gradients can inherit all benefits.

$$\hat{\nabla} f(\lambda) = \frac{p}{\mu q} \sum_{i=1}^{q} \left( f(\lambda + \mu u_i) - f(\lambda) \right) u_i \tag{12}$$

- $\mu > 0$ is the smoothing paramete.
- $\{u_i\}$ are the random query directions drawn from the standard Gaussian distribution.
- $q$ is the number of sampled query directions.
- If $\mu \to 0$, $\boxed{\frac{1}{\mu}\left(f(\lambda + \mu u) - f(\lambda)\right)} \to \boxed{f'(\lambda, u)}$, a directional derivative of function $f(\lambda)$ along $u$.
- If $f(\lambda)$ is smooth, $\nabla f(\lambda) = E_u(f'(\lambda, u)u)$

# Hyperparameter optimization method with zeroth-order hyper-gradients (HOZOG)

**Input:** Learning rate $\gamma$, approximate parameter $\mu$, size of directions $q$ and black-box inner solver $\mathcal{A}$.

1: Initialize $\lambda_0 \in \mathbb{R}^p$.
2: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
3: Generate $u = [u_1, \ldots, u_q]$, where $u_i \sim N(0; I_p)$.
4: Compute the average zeroth-order hyper-gradient $\hat{\nabla} f(\lambda_t) = \frac{p}{\mu q} \sum_{i=1}^{q} \left( f(\lambda_t + \mu u_i) - f(\lambda_t) \right) u_i$, where $f(\lambda_t)$ is estimated based on the solution returned by the black-box inner solver $\mathcal{A}$.
5: Update $\lambda_{t+1} \leftarrow \lambda_t - \gamma \hat{\nabla} f(\lambda_t)$.
6: **end for**

**Output:** $\lambda_T$.

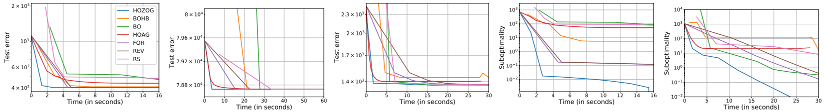
# $l_2$-Regularized Logistic Regression: 1 Hyperparamter

$$\arg\min_{\lambda\in[-10,10]} \sum_{i\in\mathcal{D}_{val}} l(y_i x_i w(\lambda)) \tag{13}$$

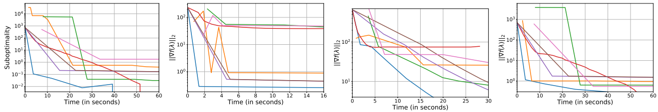
$$s.t. \quad w(\lambda) \in \arg\min_{w\in\mathbb{R}^d} \sum_{i\in\mathcal{D}_{tr}} l(y_i x_i w(\lambda)) + e^{\lambda}\|w\|^2$$

- $\mathcal{D}_{tr}$: training samples
- $\mathcal{D}_{val}$: validation samples
- $l(t) = \log(1+e^{-t})$: logistic loss

# $l_2$-Regularized Logistic Regression: 1 Hyperparamter



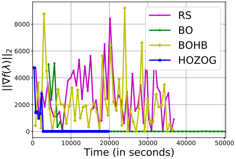

(a) News20 (b) Covtype (c) Real-sim (d) News20 (e) Real-sim



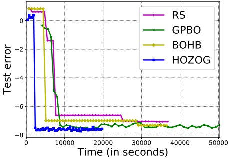

(f) Covtype (g) News20 (h) Real-sim (i) Covtype

Figure: Comparison of different hyperparameter optimization algorithms for $l_2$-regularized logistic regression sharing the same legend. (a)-(c): Test error. (d)-(f): Suboptimality. (g)-(i): $\|\nabla f(\lambda)\|_2$.

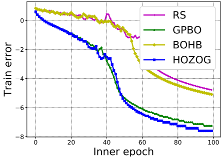
# Convolutional Neural Networks: 100 learning rates



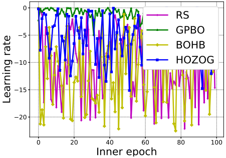

(a) $\|\nabla f(\lambda)\|_2$




(b) Test error




(c) Retrain




(d) Learning rate

Figure: Comparison of different hyperparameter optimization algorithms for 2-layer CNN sharing the same legend. (a) $\|\nabla f(\lambda)\|_2$. (b) Test error. (c) The training curve using corresponding hyperparameters. (d) Optimized learning rate by different methods.

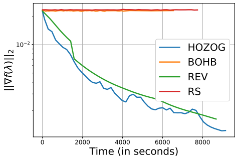
# Data Hyper-cleaning: Super Multiple Hyperparameters

$$\begin{aligned} \arg\min_{\lambda \in \mathbb{R}^{N_h}} \quad & E_{val}(W(\lambda), b(\lambda)), \qquad (14) \\ s.t. \quad & [W(\lambda), b(\lambda)] \approx \arg\min_{W,b} L(W, b) \end{aligned}$$

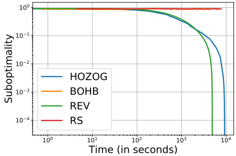
$$L(W, b) = \frac{1}{N_{tr}} \sum_{g \in \mathcal{G}} \sum_{i \in g} \mathrm{sigmoid}(\lambda_g) l(W, b, (x_i, y_i)) \qquad (15)$$

- $\mathcal{D}_{tr}$: $N_{tr}$ training samples
- $\mathcal{D}_{val}$: $N_{val}$ validation samples
- $\mathcal{G}$ contain $N_h$ groups random select from $\mathcal{D}_{tr}$

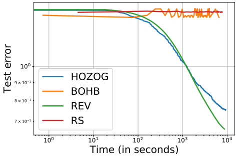
# Data Hyper-cleaning: Super Multiple Hyperparameters



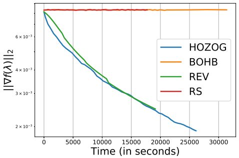

(a) 500 hyperparamters



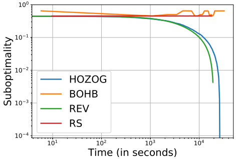

(b) 500 hyperparamters



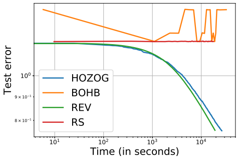

(c) 500 hyperparamters




(d) 1250 hyperparamters




(e) 1250 hyperparamters




(f) 1250 hyperparamters

Figure: Comparison of different hyperparameter optimization algorithms for data hyper-cleaning. (a&d): $\|\nabla f(\lambda)\|_2$. (b&e): Suboptimality. (c&f): Test error.

# Challenges of HOZOG

- Gu, Bin, et al. "Optimizing Large-Scale Hyperparameters via Automated Learning Algorithm." arXiv preprint arXiv:2102.09026 (2021).
- Our code is available at `https://github.com/jsgubin/HOZOG`.

# Challenges of HOZOG

- Gu, Bin, et al. "Optimizing Large-Scale Hyperparameters via Automated Learning Algorithm." arXiv preprint arXiv:2102.09026 (2021).
- Our code is available at `https://github.com/jsgubin/HOZOG`.

- How to reduce the sampling complexity to obtain a ZO estimator →
    - In HOZOG, we use q queries to obtain a good approximation of gradient
- Theoretical guarantee of HOZOG for the randomness of the inner algorithm $\mathcal{A}(\lambda)$.
    - We assume that the iterative procedure of $\mathcal{A}(\lambda)$ is deterministic.
- How to handle discontinuous hyperparameters, like the width/depth of NN
    - In HOZOG, we assume hyperparameters are continuous.

# Existing works to reduce the sampling complexity

## Learn Sampling Distribution

1. Evolutionary Strategies: Maheswaranathan et al. let the covariance matrix $\sum_k = \frac{\alpha}{n} I + \frac{1-\alpha}{k} UU^T$ be related with the recent history of ZO gradients during optimization.
2. Neural Networks: Ruan et al. dynamically predicted the covariance matrix $\Sigma_k = \mathrm{QueryRNN}([\hat{\nabla} f(x_k), \Delta x_{k-1}])$ with recurrent neural networks.


- Y. Ruan, Y. Xiong, S. Reddi, S. Kumar, and C.-J. Hsieh, "Learning to learn by zeroth-order oracle," arXiv preprint arXiv:1910.09464, 2019.
- N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: Augmenting random search with surrogate gradients," ICML, 2019, pp. 4264C4273.

# Ongoing project

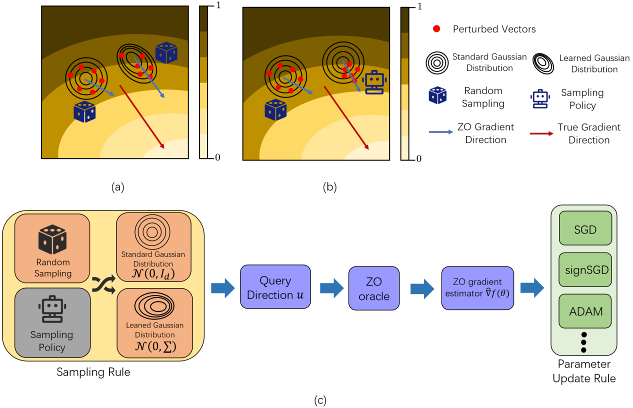
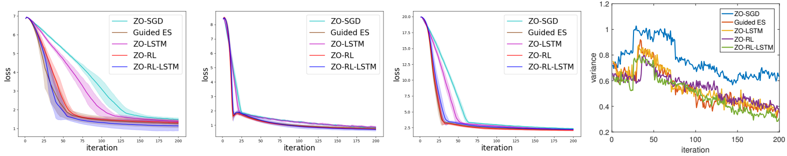
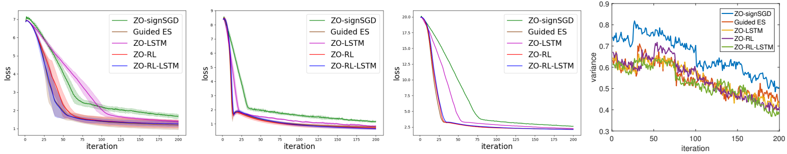



Figure: (a) Comparison of the ZO gradient directions obtained by sampling perturbed vectors from the standard Gaussian distribution and a learned Gaussian distribution. (b) Comparison of the ZO gradient directions obtained by a random sampling policy and a learned sampling policy. (c) The architecture of our ZO optimizer.




(a) mnist test 1 (b) mnist test 2 (c) mnist test 3 (d) mnist test 3

Figure: Adversarial attack to black-box models in the SGD setting.




(a) mnist test 1 (b) mnist test 2 (c) mnist test 3 (d) mnist test 3

Figure: Adversarial attack to black-box models in the signSGD setting.

# Conclusions

## Conclusions

- Global optimal hyperparameter optimization based on solution and error paths/surfaces
  - Finding global optimum of CV error is valid for one/two hyperparamters.
- New hyperparameter optimization paradigm with zeroth-order hyper-gradients
  - HOZOG is a good HO method in terms of *effectiveness, efficiency, scalability, simplicity and flexibility*.

# References


- G. Cauwenberghs and T. Poggio, Incremental and decremental support vector machine learning, in Advances in Neural Information Processing Systems 13. Cambridge, MA, USA: MIT Press, 2001, pp. 409-415.
- M. Martin, On-line support vector machine regression, in Proc. 13th Eur. Conf. Mach. Learn. (ECML), London, U.K., 2002, pp. 282-294.
- P. Laskov, C. Gehl, S. Krger, and K. R. Mller, Incremental support vector learning: Analysis, implementation and applications, J. Mach. Learn. Res., vol. 7, pp. 1909-1936, Jan. 2006.
- B. Gu and V.S. Sheng. Feasibility and finite convergence analysis for accurate on-line $\nu$- support vector machine. Neural Networks and Learning Systems, IEEE Transactions on, 24(8):1304-1315, 2013.
- Bin Gu, Victor S. Sheng, Keng Yeow Tay, Walter Romano, and Shuo Li. Cross Validation Through Two-dimensional Solution Surface for Cost-Sensitive SVM. IEEE Transactions on Pattern Analysis and Machine Intelligence.(accepted)
- Bin Gu, Victor S. Sheng, Zhijie Wang, Derek Ho, Said Osman, and Shuo Li.Incremental Learning for $\nu$-Support Vector Regression. Neural Networks, 67 (2015): 140-150.
- Bin Gu, Jian-Dong Wang, Yue-Cheng Yu, Guan-Sheng Zheng, Yu-Fan Huang, and Tao Xu. Accurate on-line $\nu$-support vector learning. Neural Networks, 27(0):51-59, 2012.
- Zaslavskiy M, Bach F, Vert J P. A path following algorithm for the graph matching problem[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009, 31(12): 2227-2242.

Thank You!
Q&A