EECE 5644: Regularization

Mark Zolotas

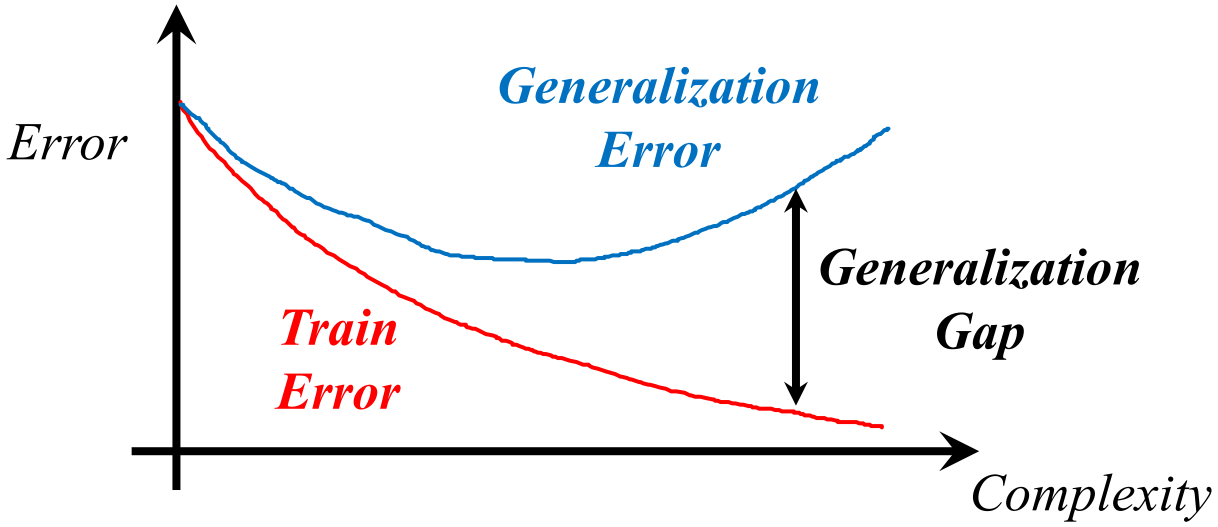
E-mail: m.zolotas@northeastern.edu
Webpage: https://coe.northeastern.edu/people/zolotas-mark/

Tentative Course Outline (Wks. 3-4)

Topics	Dates	Assignments	Additional Reading
~~Naïve Bayes Classifier &~~ ~~*Homework 0 Practice Lab*~~	~~07/18~~	~~**Homework 2** released on Canvas on 07/22~~ ~~**Due 08/01**~~	~~N/A~~
~~Model Fitting/Training: Bayesian Parameter Estimation~~	~~07/19-20~~		~~Chpts. 4.1-4.3, 8.7.2-3 Murphy 2022~~
~~Logistic Regression~~	~~07/21~~		~~Chpt. 10 Murphy 2022~~
~~Model Selection: Hyperparameter Tuning, k-fold Cross-Validation~~	~~07/26~~	**Homework 3** released on Canvas on 07/29 **Due 08/08**	~~Chpts. 4.5, 5.2, 5.4.3 Murphy 2022~~
Regularization, Ridge and Lasso Regression	07/27		Chpts. 4.5, 11.1-11.4 Murphy 2022
Neural Networks: Multilayer Perceptrons & Backpropagation	07/28		Chpts. 13.1-13.5 Murphy 2022

Overfitting

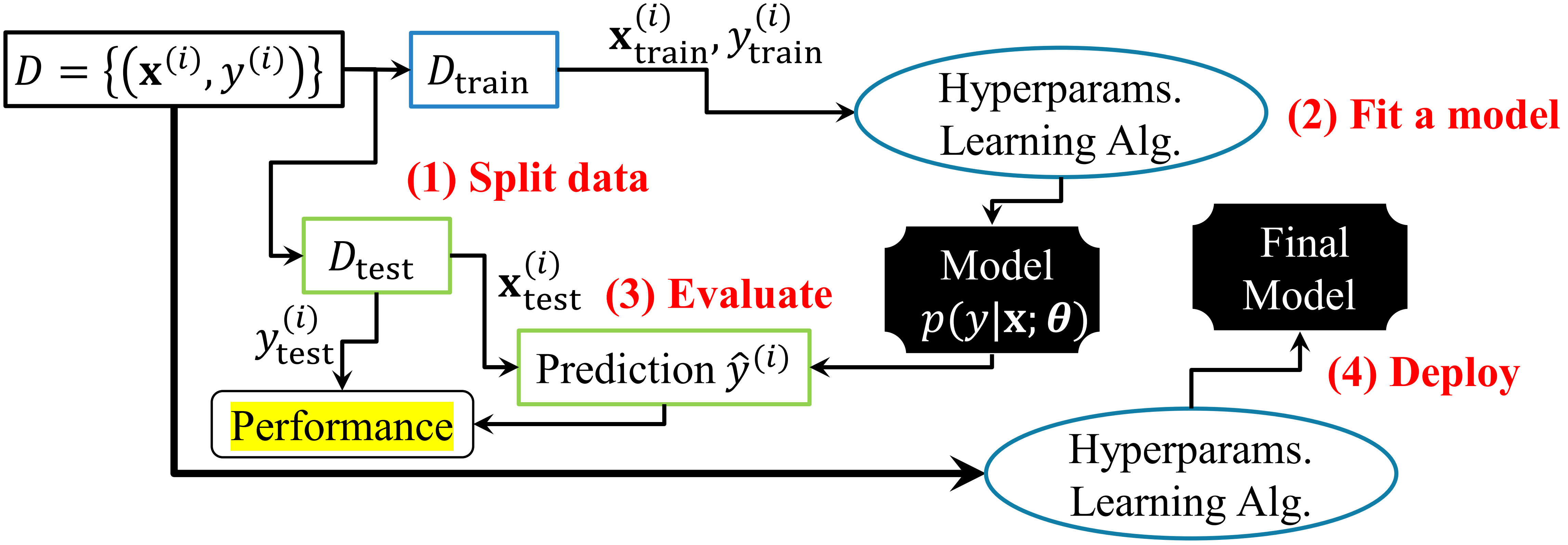
- Training models using MLE, or $L(\boldsymbol{\theta})$ in general, risks perfectly fitting the training data D_{train} and **not generalizing** well to unseen, future data…



- **Generalization gap:** $L(\boldsymbol{\theta}; p^*) - L(\boldsymbol{\theta}; D_{\text{train}})$
- Large generalization gap (low empirical, high theoretical loss) = **Overfitting**

Performance Estimation – Test Set

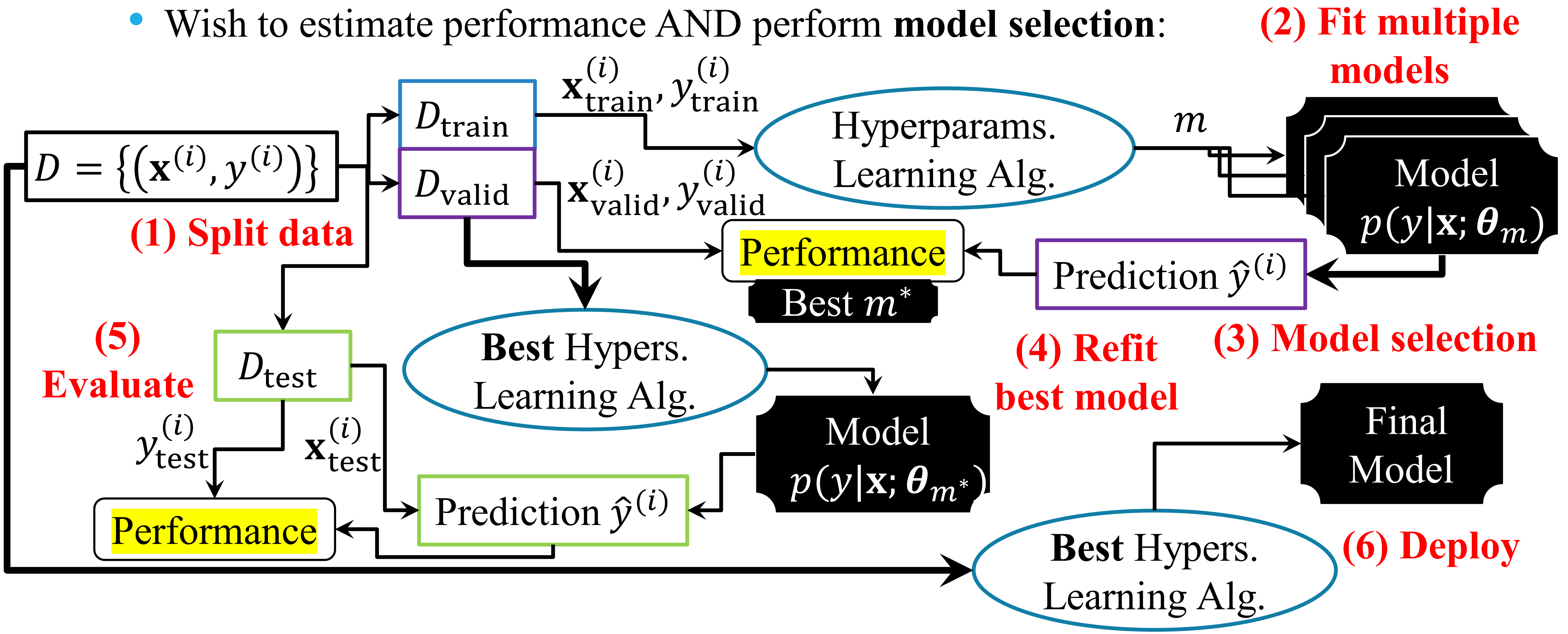
- If we are only interested in evaluating **generalization performance**:



- If **large** N → train/test split (80/20); **small** N → K-fold CV on D_{train}

Model Selection – Validation Set

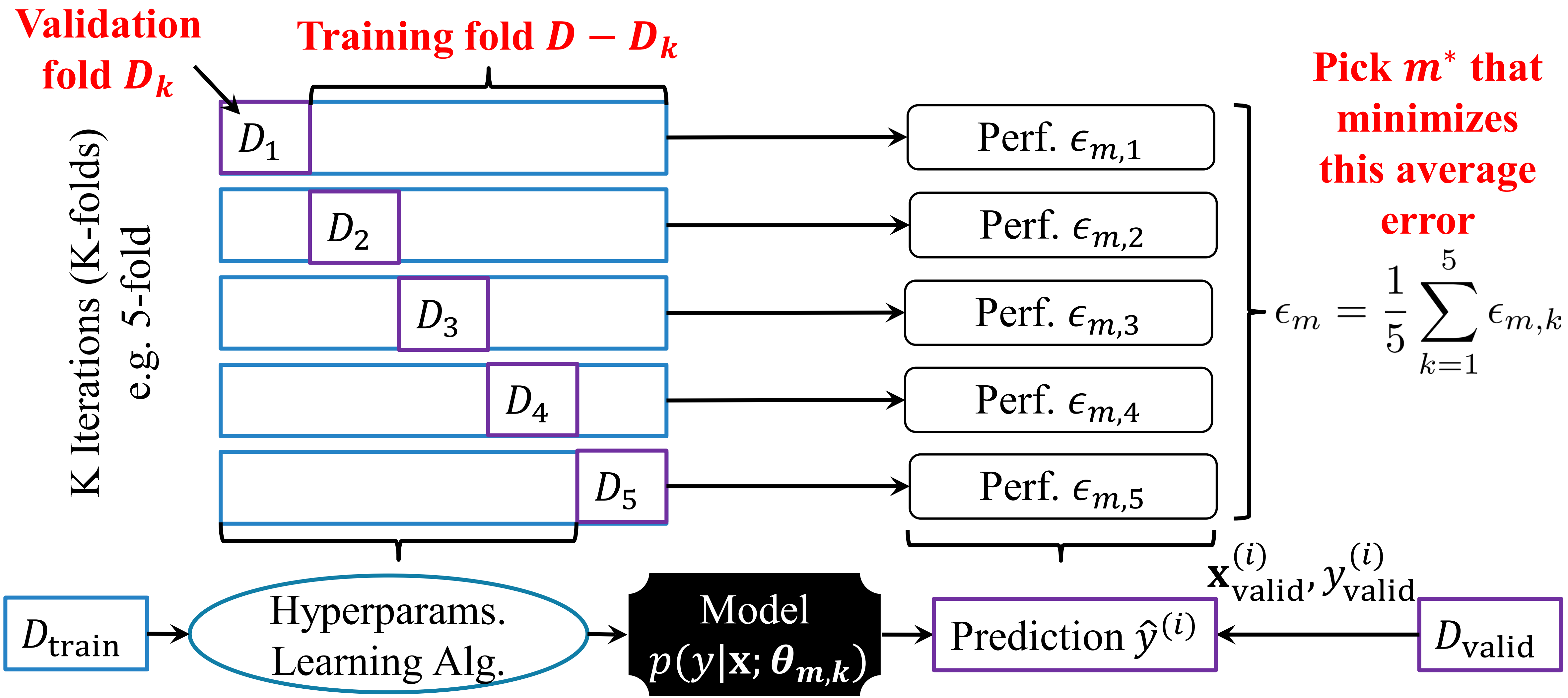
- Wish to estimate performance AND perform **model selection**:



- If **large** N → train/valid/test (60/20/20); **small** N → K-fold CV on D_{train}!

Model Selection – K-fold CV

- **Inner loop** of K-fold CV for one tested model m:



K-fold CV Algorithm

Algorithm 1 K-fold Cross-validation to estimate (hyper)parameters

Partition D into $D_1, D_2, \ldots, D_K$ with equal partition sizes (or close)

for $m \in \{1, \ldots, M\}$ **do**

  for $k \in \{1, \ldots, K\}$ **do**

    $D_{\text{valid}} = D_k \,;\; D_{\text{train}} = D - D_k$

    $\boldsymbol{\theta}^*_{m,k} = \arg\min_{\boldsymbol{\theta}_m} L_{\text{train}}(\boldsymbol{\theta}_m; D_{\text{train}})$

    $\epsilon_{m,k} = L_{\text{valid}}(\boldsymbol{\theta}^*_{m,k}; D_{\text{valid}})$

Evaluate model m on each k fold of D and record error $\varepsilon_{m,k}$

  end for

  $\epsilon_m = \frac{1}{K}\sum_{k=1}^{K} \epsilon_{m,k}$ ← Avg. CV error for model m

end for

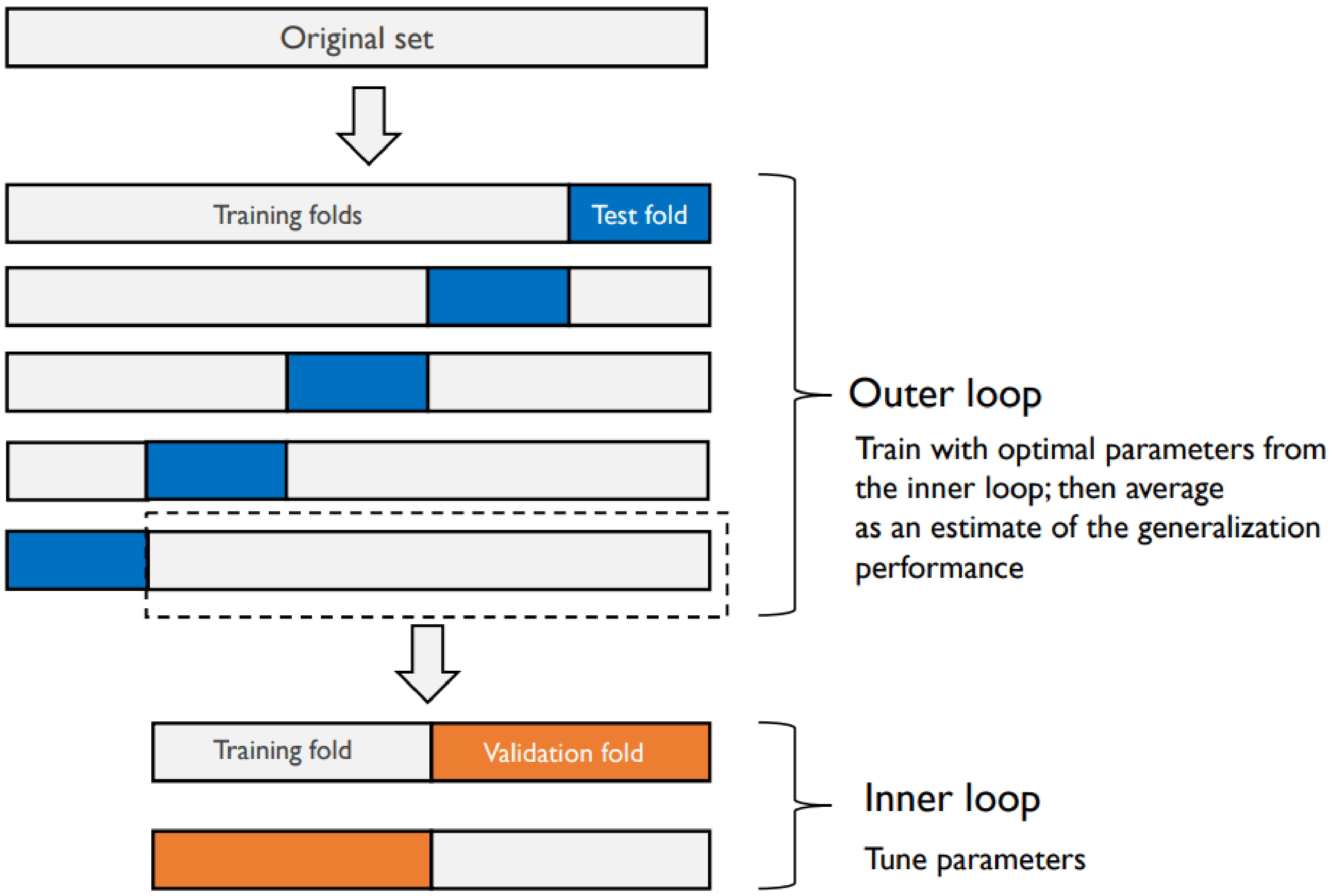
"Best" model has smallest $\boxed{\text{average}}$ error: $m^* = \arg\min_{m \in \{1,\ldots,M\}} \epsilon_m$

Given m^*, train on entire dataset: $\boldsymbol{\theta}^*_{m^*} = \arg\min_{\boldsymbol{\theta}_{m^*}} L_{\text{train}}(\boldsymbol{\theta}_{m^*}; D)$

Final trained model

Nested CV

- **Inner loop** for model selection
- **Outer loop** to estimate generalization accuracy
- Provides an **unbiased** estimate of true error
- Can be **slow**, e.g., 5*2*M iterations
- Useful for **algorithm comparison**



Source: https://androidkt.com/pytorch-k-fold-cross-validation-using-dataloader-and-sklearn/

Techniques to Handle Overfitting

- **Hold out data** → Split into data subsets D_{train} and D_{test} (80/20)
- **CV** → K-folds of D_{valid} or D_{test}; all data used for training eventually

Explicit
- **Regularization** → Automatically controls the complexity of the model

Tightly coupled with MAP estimation of θ

- And others…

Implicit
  - **Feature selection** → Eliminate unwanted features
  - **Data augmentation** → Transform inputs for more data (e.g. rotate, flip, etc.)
  - **Early stopping** → Monitor validation performance and stop training early
  - **Feature scaling** → Expect all inputs to be similar in magnitude

Regularization

- **Key Idea** – Add a term to loss function that penalizes complexity:

$$L(\boldsymbol{\theta}; \lambda) = \frac{1}{N} \sum_{i=1}^{N} l\left(y^{(i)}, f(\mathbf{x}^{(i)}; \boldsymbol{\theta})\right) + \lambda C(\boldsymbol{\theta})$$

MAP

- $\lambda \geq 0$ is the **regularization** parameter (also a hyperparameter)

parameter estimation

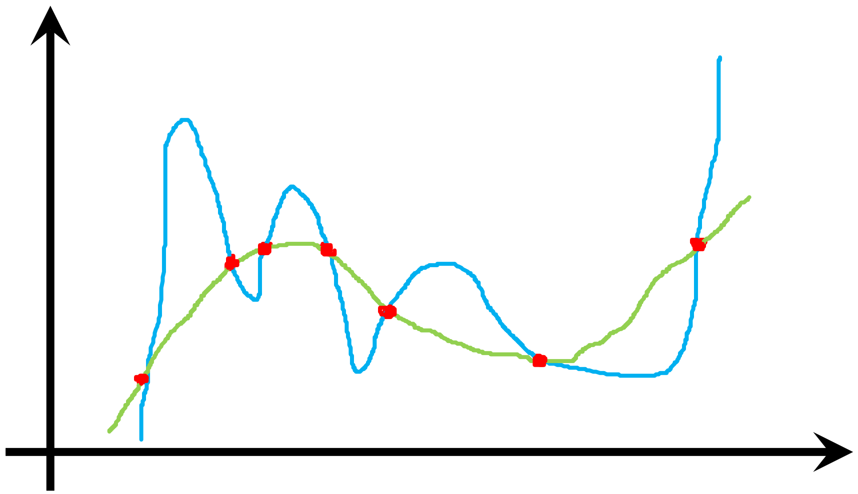
- $C(\boldsymbol{\theta})$ is the **complexity penalty**, e.g., negative log prior $C(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta})$

log loss

$$L(\boldsymbol{\theta}; \lambda) = -\left[\log p(D \mid \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right] = -\frac{1}{N} \sum_{i=1}^{N} \log p(y^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) \underbrace{- \log p(\boldsymbol{\theta})}_{C(\theta) \text{ with } \lambda = 1}$$

- Works well with **high** n-dimensional inputs

Regularization for Linear Regression

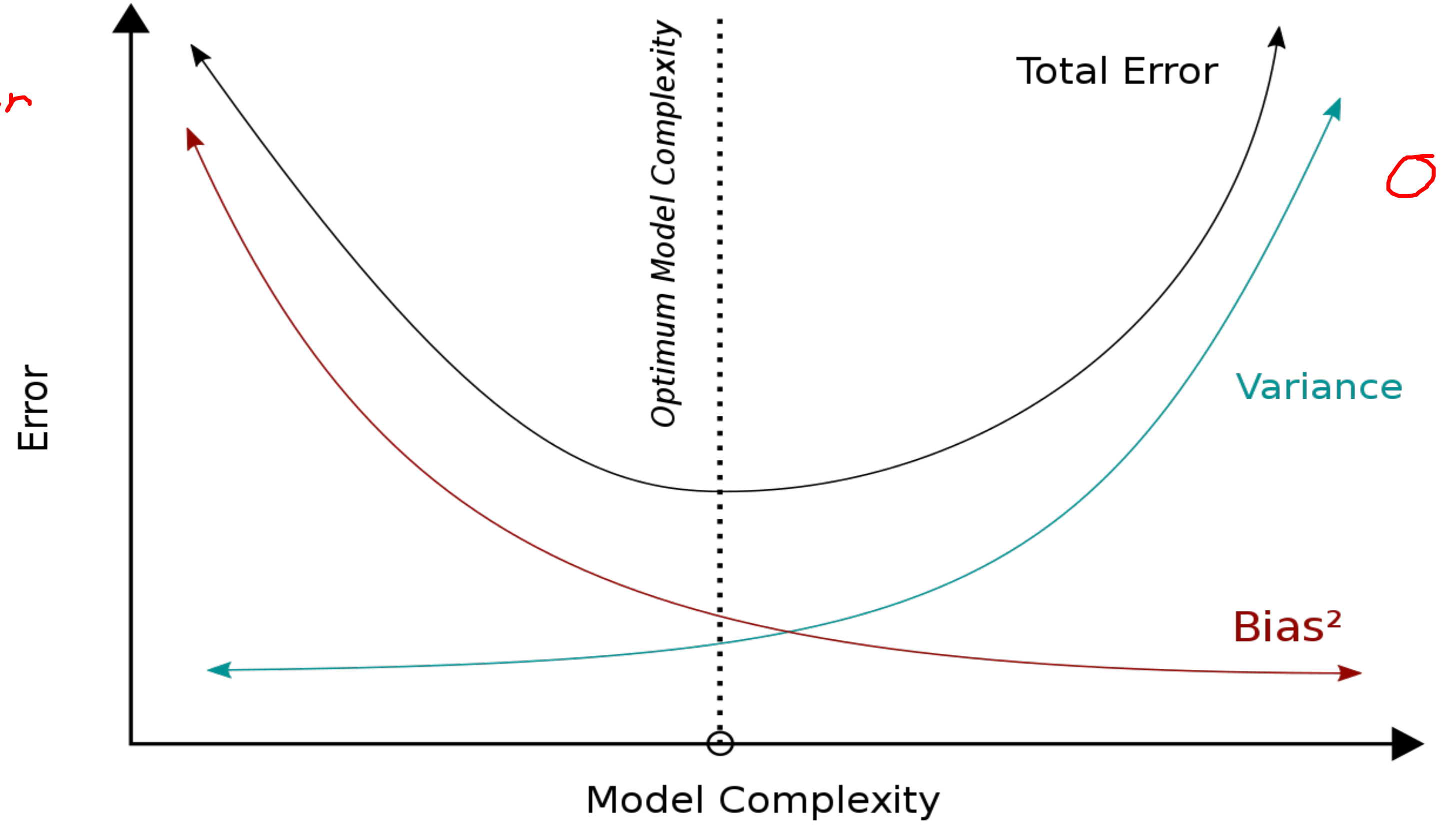


Red: Training set
Green: True target function
Blue: What we have learned (overfit)

Bias-Variance Tradeoff

Under

Reduce model **complexity**; Reduce model **variance**



Over

Bias = Difference between estimated and true models
Variance = Model sensitivity/fluctuations to different training sets

Mean Squared Error (MSE) proportional to Variance and Bias

Least Squares Linear Regression

- Recall linear regression model as a conditional Gaussian:

$$p(y \,|\, \mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{w}^\mathsf{T}\mathbf{x}, \sigma^2\right)$$

- Optimize using MLE, i.e., minimize NLL:

$$\mathrm{NLL}(\boldsymbol{\theta}) = \underbrace{\frac{1}{2\sigma^2}}_{\text{const}} \overbrace{\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^\mathsf{T}\mathbf{x}^{(i)})^2}^{\text{RSS}} + \underbrace{\frac{N}{2}\log(2\pi\sigma^2)}_{\text{const}}$$

- Assume fixed variance, e.g. $\sigma = 1$, and focus on $\mu = \mathbf{w}^\mathsf{T}\mathbf{x}$:

$$\mathrm{NLL}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^\mathsf{T}\mathbf{x}^{(i)})^2 = \frac{1}{2}||\mathbf{X}\mathbf{w} - \mathbf{y}||_2^2 = \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Ridge Regression – Formulation

- Penalize **large weight magnitudes** of NLL($\boldsymbol{\theta}$) to avoid overfitting

Also called $\boldsymbol{l_2}$ regularization or **weight decay**

$$\mathrm{PNLL}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^\intercal\tilde{\mathbf{x}}^{(i)})^2 + \lambda\sum_{j=1}^{n} w_j^2$$

Not bias w_0

$$= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^\intercal(\mathbf{X}\mathbf{w} - \mathbf{y}) + \underbrace{\lambda\mathbf{w}^\intercal\mathbf{w}}_{\lambda\|w\|_2^2}$$

L2-norm of w

- $\lambda \geq 0$ is the regularization parameter

- As if computing the $\widehat{\boldsymbol{\theta}}_{\mathrm{MAP}}$ estimate with a **zero-mean Gaussian prior** on the parameters or in our cases, weights: $p(\mathbf{w}) = N(\mathbf{w} \mid \mathbf{0}, \lambda^{-1}\mathbf{I})$

$\theta = [w_1 \cdots w_n]^T$

Lasso Regression – Formulation

- Uses a **Laplace prior** instead, with a resulting form:

$$\mathrm{PNLL}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - \mathbf{w}^{\intercal}\tilde{\mathbf{x}}^{(i)})^2 + \lambda\sum_{j=1}^{n}|w_j|$$

$$= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{y})^{\intercal}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda||\mathbf{w}||_1$$

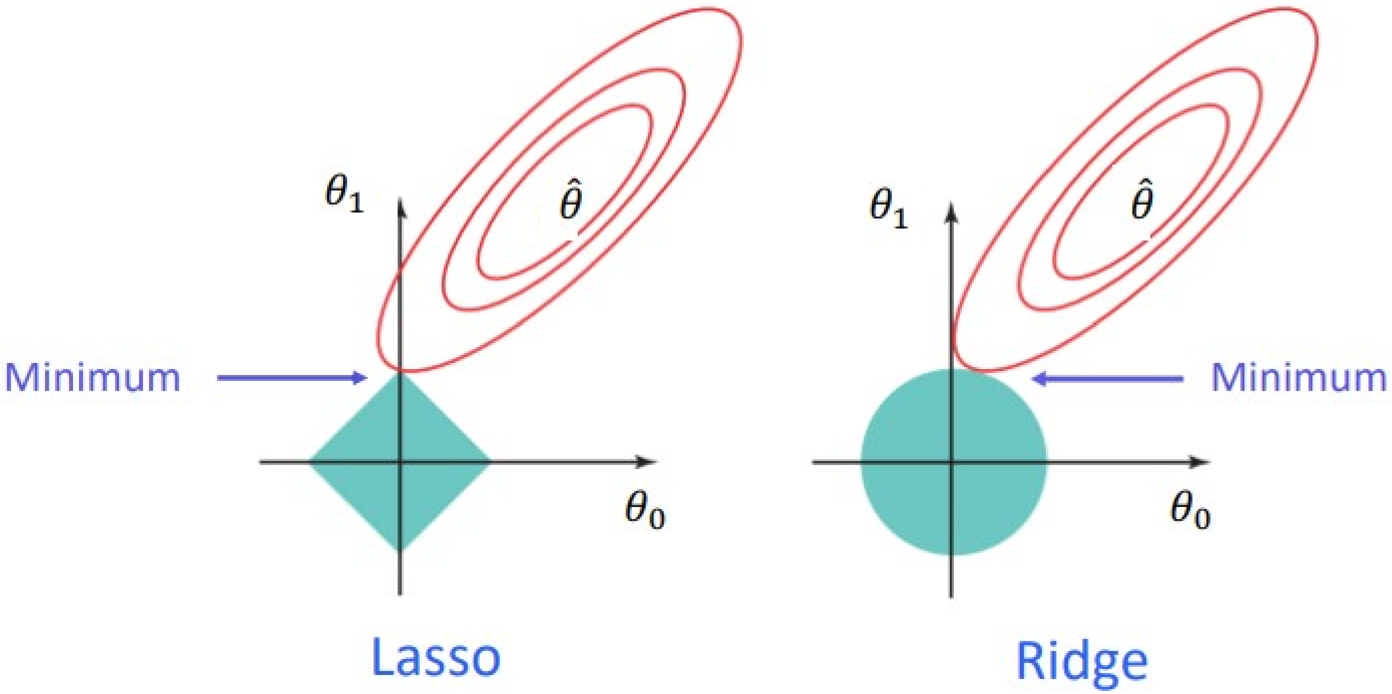
L1-norm of w

Also called $\boldsymbol{l_1}$ regularization

- **Lasso:** "least absolute shrinkage and selection operator"
- **Warning:** Gradients cannot be computed around zero (absolute value)

Geometric Interpretation

- **Ridge** regression **shrinks** all coefficients (parameters)
- **Lasso** regression sets coefficients to **zero** (sparse solution); **feature selection**



Regularization for Logistic Regression

Same idea as **Ridge** regression:

$$\mathrm{PNLL}(\cancelto{w}{\boldsymbol{\theta}}) = \mathrm{NLL}(\cancelto{w}{\boldsymbol{\theta}}) + \lambda \mathbf{w}^{\intercal}\mathbf{w}$$

$$\nabla_{\cancelto{w}{\boldsymbol{\theta}}}\mathrm{PNLL}(\cancelto{w}{\boldsymbol{\theta}}) = \nabla_{\boldsymbol{\theta}}\mathrm{NLL}(\cancelto{w}{\boldsymbol{\theta}}) + 2\lambda\mathbf{w}$$

Concluding Remarks

- **Explicit** regularization adds a term to optimization problem to penalize complexity, e.g., **prior** term for **MAP** parameter estimation
- **Implicit** regularization refers to other techniques like early stopping
- Some early CV-related slides adapted from:

  https://arxiv.org/pdf/1811.12808.pdf

- Questions?