STAT 215A Fall 2019
Week 8c

Tiffany Tang

10/18/19

Plan for Today:

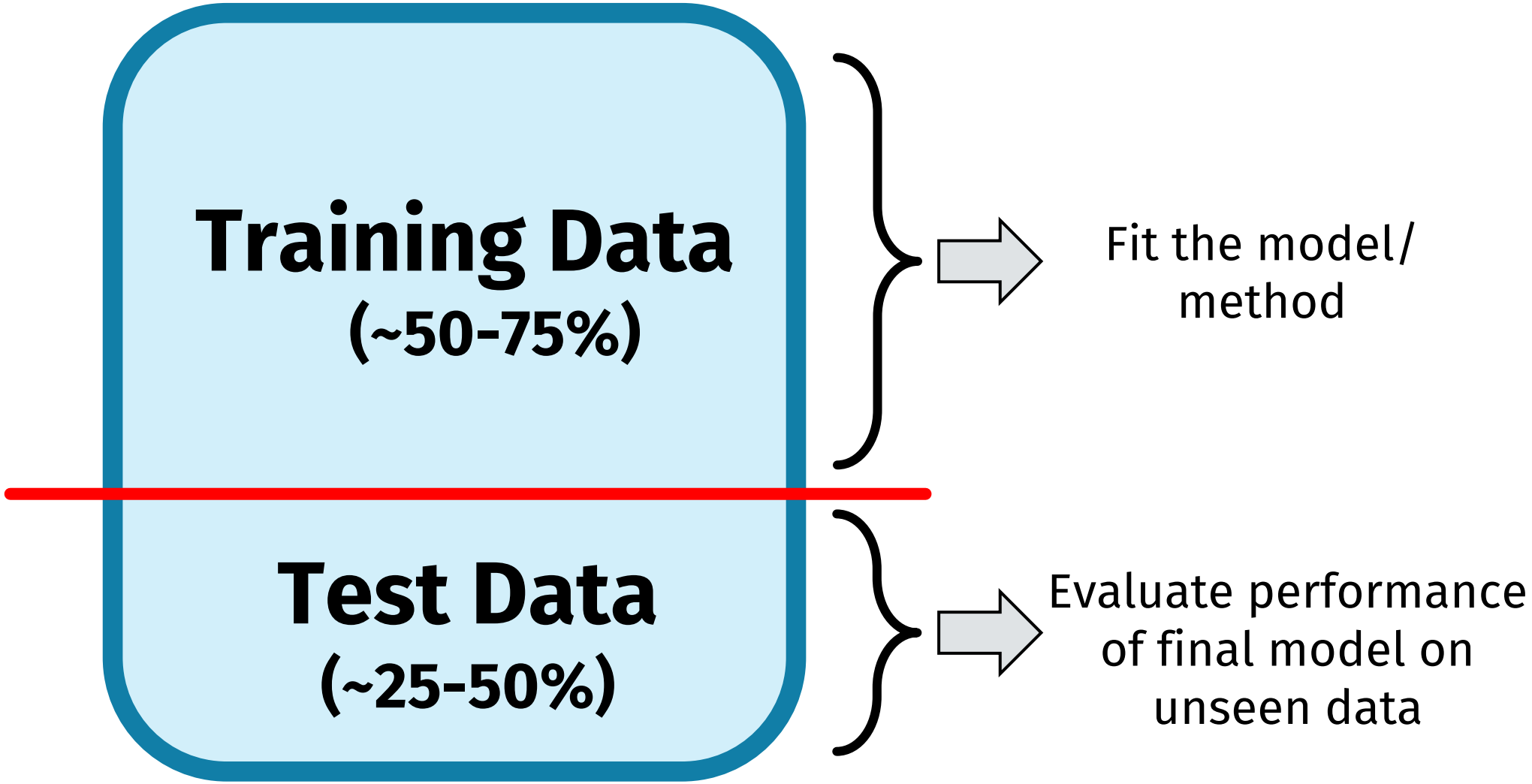
- Data splitting and the statistical learning pipeline
- Bias-variance Tradeoff
- Ordinary Least Squares
- Regularized Regression (Ridge, Lasso)

Statistical (aka Machine) Learning

- **Goal:** to "learn" insights about the current data that are generalizable to future data
- Typically, have one of the following tasks in mind:
  - **Prediction** – have an outcome of interest
  - **Data-driven discoveries** – pattern recognition, feature selection, what's important in the data
- Two main types of statistical learning:
  - **Unsupervised learning** – no outcomes/response data
    - Ex. clustering, dimension reduction, pattern recognition
  - **Supervised learning** – have outcomes/response data
    - **Classification** categorical responses
    - **Regression** – continuous responses

How to assess generalizability: data splitting

- **The simplest case:**



How to assess generalizability: data splitting

- Can use data splitting idea for both unsupervised and supervised learning methods
- **IMPORTANT:** Do NOT touch the test set until the very end when you are ready to *evaluate* the generalization performance of your final method
  - **Purpose of the test set:** to obtain an unbiased estimate of the prediction error for your statistical learning pipeline on future data
- In more realistic pipelines, we will want to compare multiple methods, tune hyper-parameters, etc. We will return to data splitting in this case later.

How to evaluate?

- For unsupervised learning methods, this is difficult and typically depends on the method itself
  - Ex. NMF, kmeans
- For supervised methods, there are several possible metrics:
  - **Mean squared error**
  - Absolute loss
  - Correlation
  - AUC
  - Classification error – 0/1 loss

Mean Squared Error

- **Notation:**
  - x_i = observed feature measurements from sample i
  - y_i = observed response for sample i
  - For simplicity, assume the underlying truth is given by $y = f(x)$
  - $\hat{y} = \hat{f}(x)$ = prediction from statistical method or algorithm
- Want prediction function to give small MSE

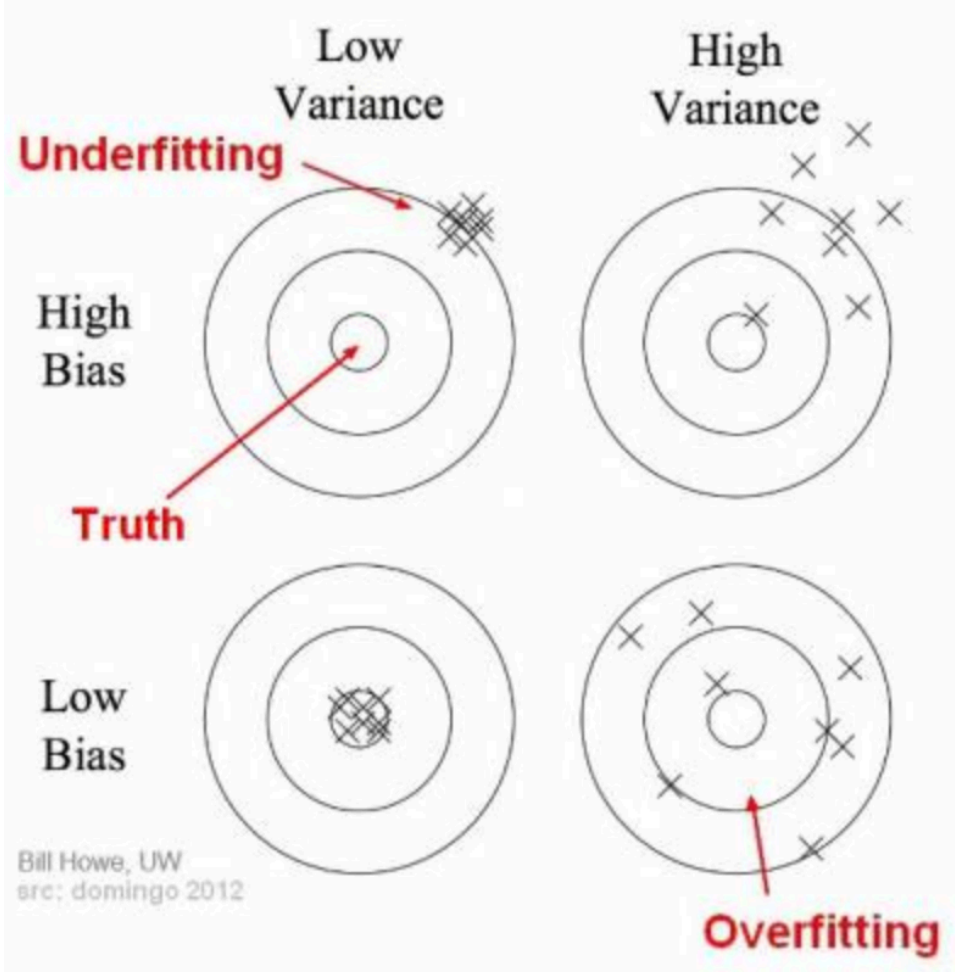
$$\text{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] \qquad \text{(Population)}$$

$$\widehat{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2 = \frac{1}{n}||\,\mathbf{y} - \hat{\mathbf{y}}||_2^2 \qquad \text{(Empirical)}$$

Bias-Variance Tradeoff

$$\mathrm{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] = \mathrm{Var}(\hat{f}(X)) + \mathrm{Bias}^2(\hat{f}(X))$$

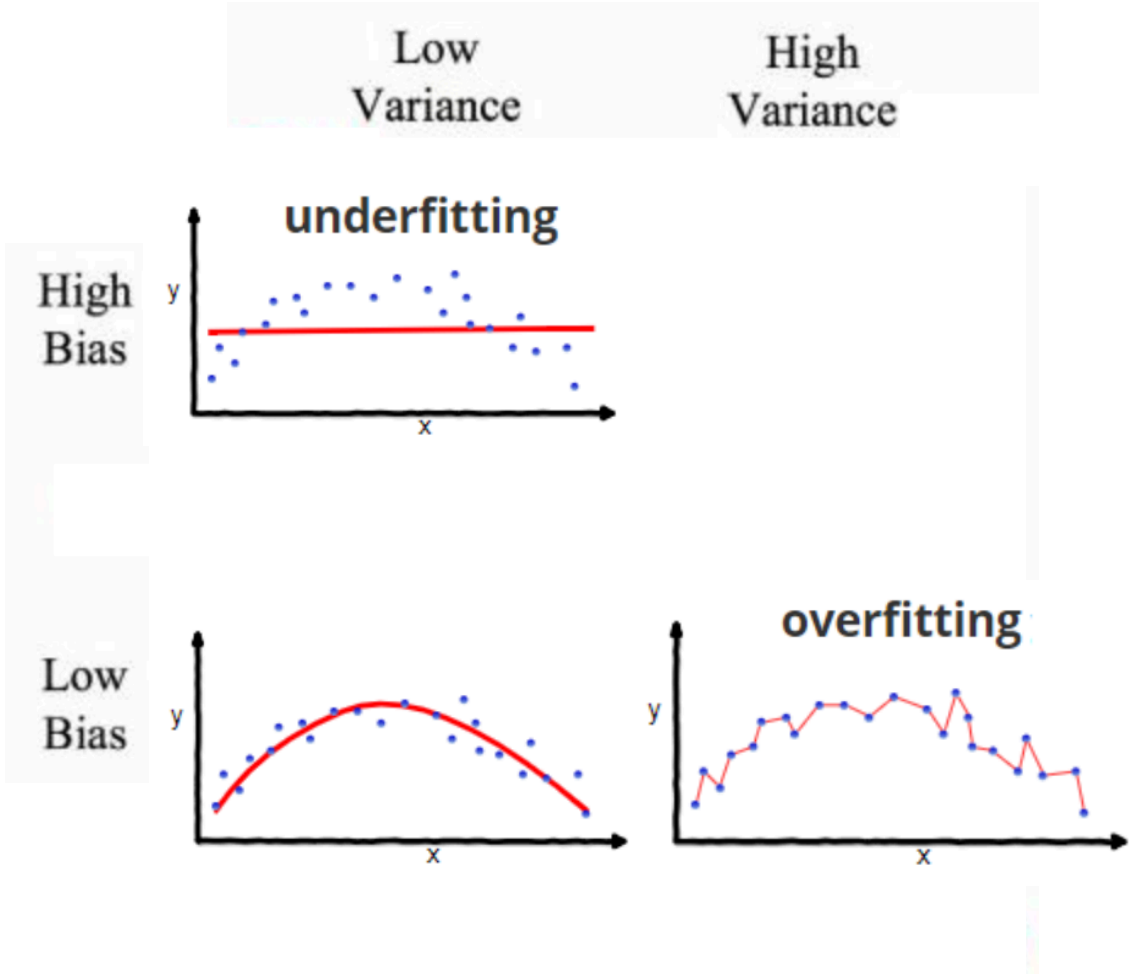
- **Bias:** on average, how wrong is your prediction from the truth
- **Variance:** if you obtain a new, but similar dataset, how much does this change your predictions



Bias-Variance Tradeoff

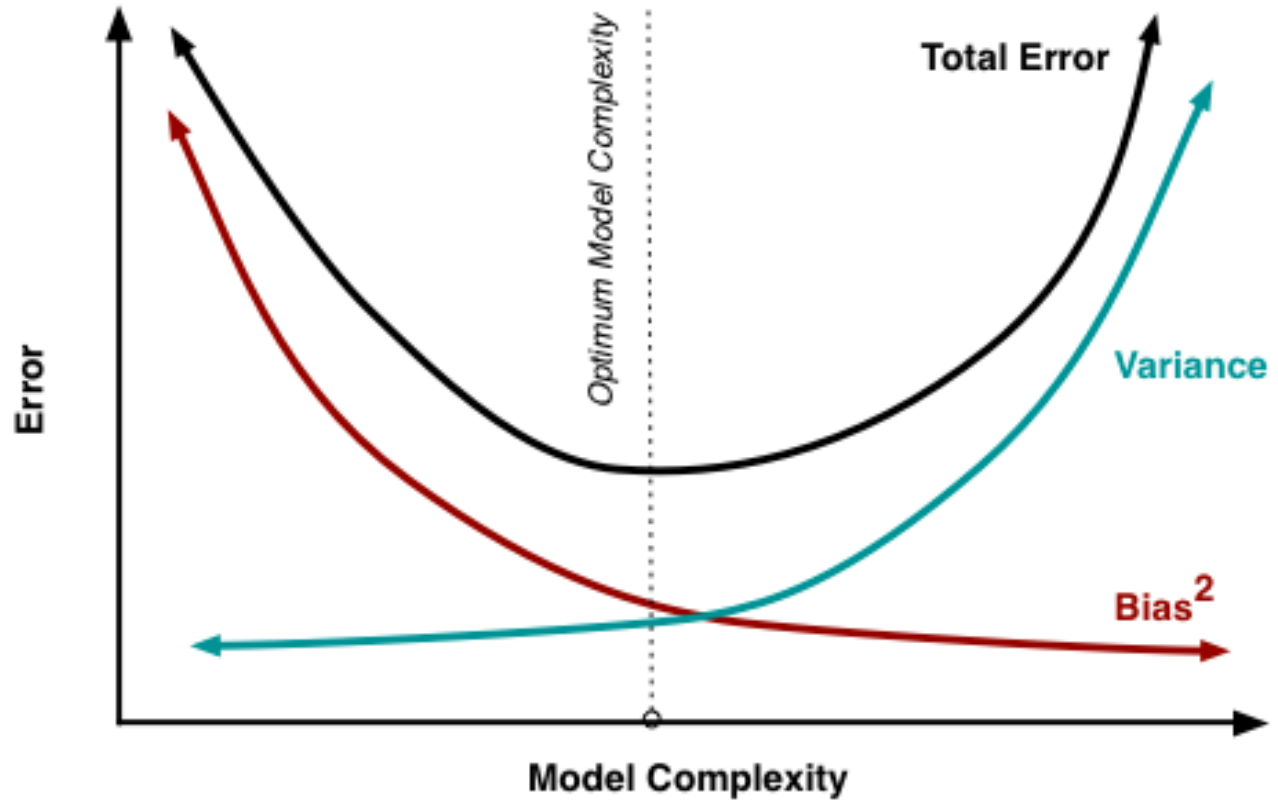
$$\text{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] = \text{Var}(\hat{f}(X)) + \text{Bias}^2(\hat{f}(X))$$

- **Bias:** on average, how wrong is your prediction from the truth
- **Variance:** if you obtain a new, but similar dataset, how much does this change your predictions



Bias-Variance Tradeoff

$$\mathrm{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] = \mathrm{Var}(\hat{f}(X)) + \mathrm{Bias}^2(\hat{f}(X))$$



Supervised Learning: A general framework

1. Split the data into training/test sets
2. On the training data, try to minimize some **loss** function, which quantifies the error between the observed values and the predicted values

$$\min_{\hat{f} \in \mathcal{F}} \ \mathcal{L}(\mathbf{y}, \hat{f}(\mathbf{X}))$$

   - Ex. MSE:

$$\min_{\hat{f} \in \mathcal{F}} \ \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

3. Report test set error as a measure of generalizability

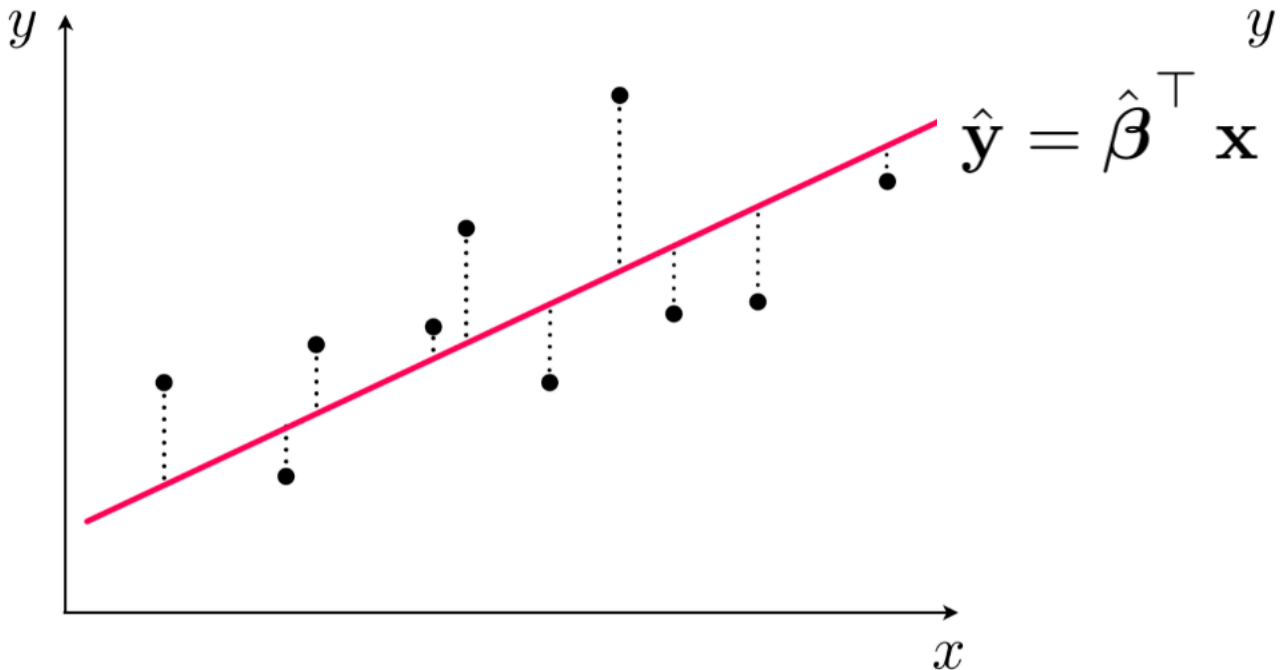
Ordinary Least Squares (OLS)

- Find best linear approximation of the data, as measured by MSE

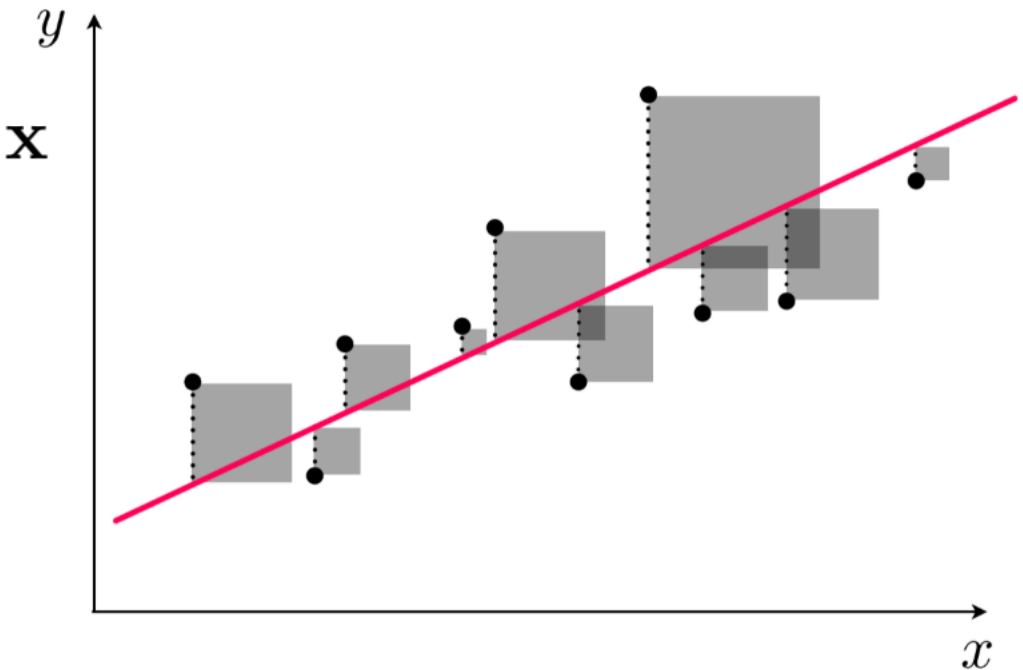
$$\hat{\boldsymbol{\beta}}_{OLS} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \underbrace{\|\mathbf{y} - \mathbf{X}\,\boldsymbol{\beta}\|_2^2}$$

Loss function

- Closed-form solution: $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1}\,\mathbf{X}^\top\,\mathbf{y}$



$$\hat{\mathbf{y}} = \hat{\boldsymbol{\beta}}^\top\,\mathbf{x}$$



Ordinary Least Squares (OLS)

Advantages

- Simple
- Closed-form solution
- Interpretable (?)
- Under some modeling assumptions, OLS has some desirable properties

Ordinary Least Squares (OLS) + Model

- Assume that

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon},$$

  where $\mathrm{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$.

- Then $\widehat{\boldsymbol{\beta}_{OLS}}$ is an unbiased estimator for $\boldsymbol{\beta}$.

- If, in addition, $Var(\varepsilon_i) = \sigma^2$ for all i and $Cov(\varepsilon_i, \varepsilon_j) = 0$ for all $i \neq j$, then OLS is BLUE (the "best" linear unbiased estimator)

  - Out of all linear unbiased estimators, $\widehat{\boldsymbol{\beta}_{OLS}}$ has the minimum variance (Gauss-Markov Theorem)

$$\mathrm{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] = \mathrm{Var}(\hat{f}(X)) + \mathrm{Bias}^2(\hat{f}(X))$$

Ordinary Least Squares

Advantages

- Simple
- Closed-form solution
- Interpretable (?)
- Under some modeling assumptions, OLS has some desirable properties

Disadvantages

- Too simple
- When p > n
  - Non-unique solution; high variance
  - Massive overfitting since training error = 0

One possible solution

$$\mathrm{MSE} = \mathbb{E}\left[(Y - \hat{f}(X))^2\right] = \mathrm{Var}(\hat{f}(X)) + \mathrm{Bias}^2(\hat{f}(X))$$

- Let's sacrifice some bias for a larger reduction in variance
- One way to reduce the variance of your predictions is to restrict the class of functions you search over
- In the linear setting, this motivates regularized linear regression methods such as ridge, Lasso, and elastic net

Ridge Regression

$$\hat{\boldsymbol{\beta}}^r = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; \underbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2}_{\text{Loss function}} \quad \text{subject to} \quad \underbrace{||\boldsymbol{\beta}||_2^2 \leq \tau}_{\text{Constraint}}$$

or equivalently,

$$\hat{\boldsymbol{\beta}}^r = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; \overbrace{||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2}^{\text{Loss function}} + \overbrace{\lambda||\boldsymbol{\beta}||_2^2}^{\text{Penalty}}$$

- $\lambda \geq 0$ is called a tuning or penalty parameter and regulates how much *shrinkage* is introduced
  - $\lambda \to 0 \implies \hat{\boldsymbol{\beta}}^r \to$ OLS (no bias, high variance)
  - $\lambda \to \infty \implies \hat{\boldsymbol{\beta}}^r \to \mathbf{0}$ (high bias, but no variance)
  - λ somewhere in between $\implies$ some shrinkage
  - Choose λ via CV

But still, why is ridge regression a good idea?

$$\hat{\boldsymbol{\beta}}^r = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\,\mathbf{y} - \mathbf{X}\,\boldsymbol{\beta}\,||_2^2 + \lambda||\,\boldsymbol{\beta}\,||_2^2$$

1. The solution exists and is unique even when p > n (unlike OLS)

$$\hat{\boldsymbol{\beta}}^r = (\mathbf{X}^\top\,\mathbf{X} + \lambda\,\mathbf{I})^{-1}\,\mathbf{X}^\top\,\mathbf{y}$$

But still, why is ridge regression a good idea?

1. The solution exists and is unique even when p > n (unlike OLS)

$$\hat{\boldsymbol{\beta}}^r = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

2. Under the model from before,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \text{where} \quad \mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0}, \;\; \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^2 \mathbf{I}$$

there **always** exists a λ such that the Ridge(λ) MSE is less than the OLS MSE:

$$\mathrm{MSE}\left(\mathbf{X}\hat{\boldsymbol{\beta}}^r(\lambda)\right) < \mathrm{MSE}\left(\mathbf{X}\hat{\boldsymbol{\beta}}^{OLS}\right)$$

But still, why is ridge regression a good idea?

1. The solution exists and is unique even when p > n (unlike OLS)

$$\hat{\boldsymbol{\beta}}^{r} = (\mathbf{X}^{\top}\mathbf{X} + \lambda\,\mathbf{I})^{-1}\,\mathbf{X}^{\top}\,\mathbf{y}$$

2. Under the model from before,

$$\mathbf{y} = \mathbf{X}\,\boldsymbol{\beta} + \boldsymbol{\epsilon}, \qquad \text{where} \quad \mathbb{E}[\boldsymbol{\epsilon}] = \mathbf{0},\ \ \mathrm{Var}(\boldsymbol{\epsilon}) = \sigma^{2}\,\mathbf{I}$$

   there **always** exists a λ such that the Ridge(λ) MSE is less than the OLS MSE:

$$\mathrm{MSE}\left(\mathbf{X}\,\hat{\boldsymbol{\beta}}^{r}(\lambda)\right) < \mathrm{MSE}\left(\mathbf{X}\,\hat{\boldsymbol{\beta}}^{OLS}\right)$$

3. Handles correlated features very well – Features that are highly correlated tend to be shrunken together, i.e. they are given equal contribution to the linear model

Ridge Regression in Practice

- Don’t want to penalize the intercept term
  - Before applying any regularized regression such as ridge regression, center columns of **X** and center **y**
- Most of the time, should also scale columns of **X** so that we don’t penalize coefficients more than others simply because of differing scales
- These practical guidelines apply to ALL regularization methods

Feature Selection

- Want to select the best subset of s variables for prediction (typically s << p)
- Looking for data-driven discoveries or interpretable insights
- Ridge regression does not perform *automatic* feature selection
  - Can use ridge regression to order the importance of features using the magnitude of their coefficients (which is often done in practice)
  - But typically there are no exact 0's in the coefficients vector
- Methods that more directly solve the feature selection problem:
  - Best subsets (L_0) regression
  - Lasso

Best Subsets (L_0) Regression

$$\hat{\boldsymbol{\beta}}^{best} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\, \mathbf{y} - \mathbf{X}\boldsymbol{\beta} \,||_2^2 \qquad \text{subject to} \quad ||\, \boldsymbol{\beta} \,||_0 \leq s$$

- Want to find the subset of s features that give the smallest MSE
- **Major problem:**
  - Computationally infeasible for moderate-sized problems
  - Need to do an exhaustive search over $\binom{p}{s}$ regressions

Lasso (L_1) Regression

$$\hat{\boldsymbol{\beta}}^{l} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\,\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\,||_2^2 \qquad \text{subject to} \quad ||\,\boldsymbol{\beta}\,||_1 \leq \tau$$

or equivalently,

$$\hat{\boldsymbol{\beta}}^{l} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\,\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\,||_2^2 + \lambda||\,\boldsymbol{\beta}\,||_1$$

- "Best convex relaxation" of the best subsets (L_0) problem (i.e., the next best thing that we can actually compute in a reasonable amount of time)
- This L_1 penalty results in **sparsity** (lots of non-zero entries in $\hat{\boldsymbol{\beta}}^{l}$)
- More interpretable than ridge but can be worse in prediction if the true underlying process is not exactly sparse

Lasso (L_1) Regression

$$\hat{\boldsymbol{\beta}}^{l} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\,\mathbf{y} - \mathbf{X}\,\boldsymbol{\beta}\,||_2^2 \qquad \text{subject to} \quad ||\,\boldsymbol{\beta}\,||_1 \leq \tau$$

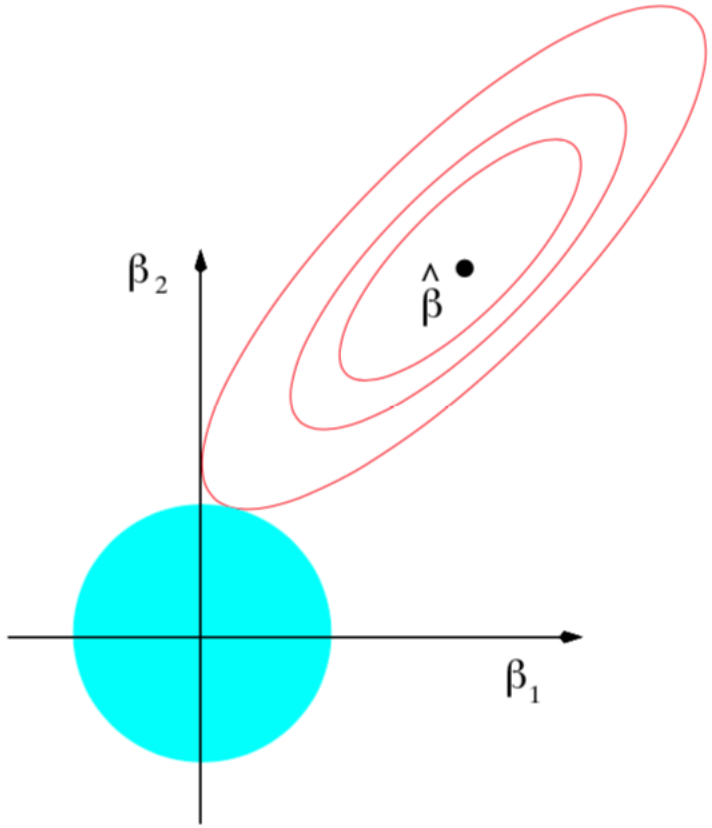
or equivalently,

$$\hat{\boldsymbol{\beta}}^{l} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\,\mathbf{y} - \mathbf{X}\,\boldsymbol{\beta}\,||_2^2 + \lambda||\,\boldsymbol{\beta}\,||_1$$

- $\lambda \geq 0$ is called a tuning or penalty parameter and regulates how the *sparsity* level
  - $\lambda \to 0 \implies \hat{\boldsymbol{\beta}}^{l} \to$ OLS (no bias, high variance)
  - $\lambda \to \infty \implies \hat{\boldsymbol{\beta}}^{l} \to \mathbf{0}$ (high bias, but no variance)
  - λ somewhere in between $\implies$ some sparsity
  - Choose λ via CV

Why does the Lasso induce sparsity?

$$\hat{\boldsymbol{\beta}}^r = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; || \mathbf{y} - \mathbf{X}\boldsymbol{\beta} ||_2^2$$
$$\text{subject to} \quad || \boldsymbol{\beta} ||_2^2 \leq \tau$$



$$\hat{\boldsymbol{\beta}}^l = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; || \mathbf{y} - \mathbf{X}\boldsymbol{\beta} ||_2^2$$
$$\text{subject to} \quad || \boldsymbol{\beta} ||_1 \leq \tau$$

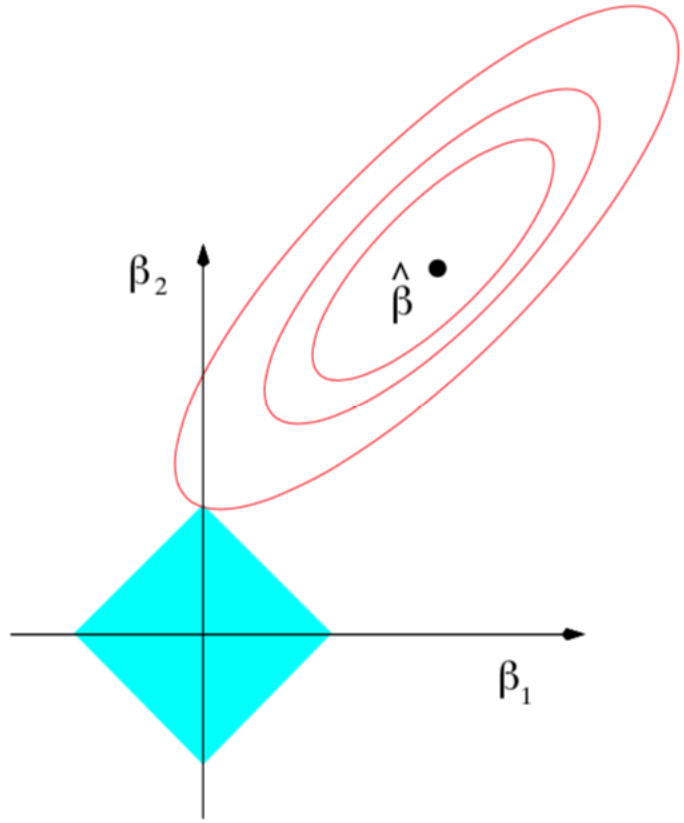


Image source: Elements of Statistical Learning (p. 71)

Lasso in Practice

- Great for feature selection and interpretability
- **Warning:**
  - Non-zero Lasso coefficient ≠ statistically significant
  - Cannot fit OLS after doing Lasso and interpret the p-values in the same way; OLS inference is no longer valid after doing model selection
- Tends to select one representative from a group of correlated features
  - Good if goal is a parsimonious model
  - Bad for data-driven discoveries
- Like ridge regression, don't forget to center and scale data
- For large λ, the coefficients of selected variables are heavily biased

Other Alternatives/Modifications to the Lasso

- **Weighted Lasso:** use weighted penalty = $\lambda \sum_{j=1}^{p} w_j |\beta_j|$
  - If true β_j is large, don't want to penalize it, so want w_j small
  - If true β_j is small, want to penalize it more, so want w_j large
- **Adaptive Lasso** (Zou (2006)): $w_j = \left(\frac{1}{|\hat{\boldsymbol{\beta}}_j^{OLS/ridge}|} \right)^{\gamma} \quad (\gamma \in \{0.5, 1, 2\})$
- **Non-convex Penalties:**
  - **SCAD** (Fan (2001))
  - **MCP** (Zhang (2009)) - variables that are non-zero in this model are essentially unbiased
- **Randomized Lasso** (Meinshausen (2010)): combines stability via bootstrapping and randomized weights to handle correlated variables

Elastic Net

$$\hat{\boldsymbol{\beta}}^{el} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; || \mathbf{y} - \mathbf{X}\boldsymbol{\beta} ||_2^2 + \lambda \left(\alpha || \boldsymbol{\beta} ||_1 + (1-\alpha) || \boldsymbol{\beta} ||_2^2 \right)$$

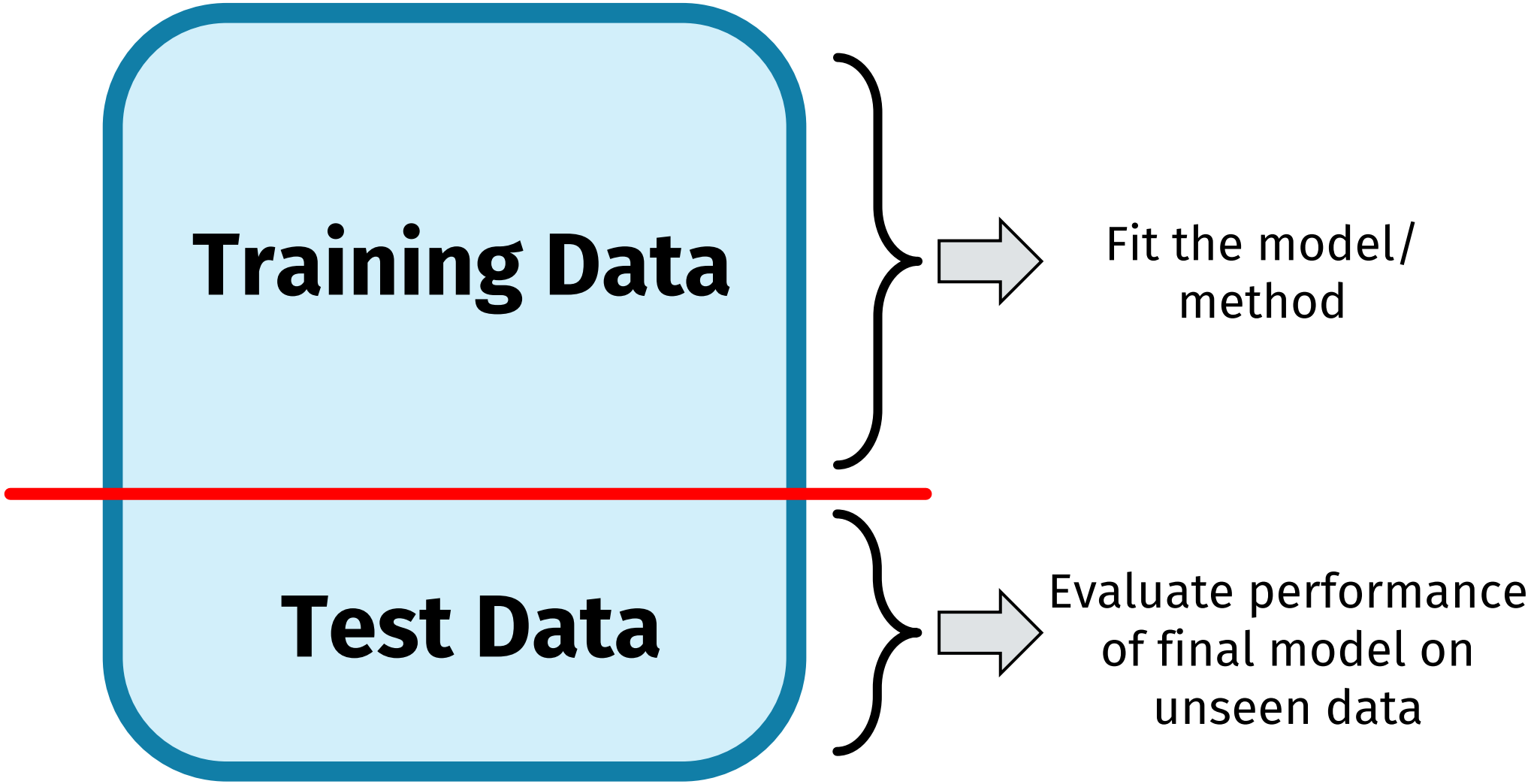
- Compromise between ridge and Lasso depending on choice of $\alpha \in [0, 1]$
  - Ridge: $\alpha = 0$
  - Lasso: $\alpha = 1$
- Can handle correlated features better than the Lasso and enforces more sparsity than ridge
- Difficult to tune both α and λ in practice

Review: OLS + Regularized Regression

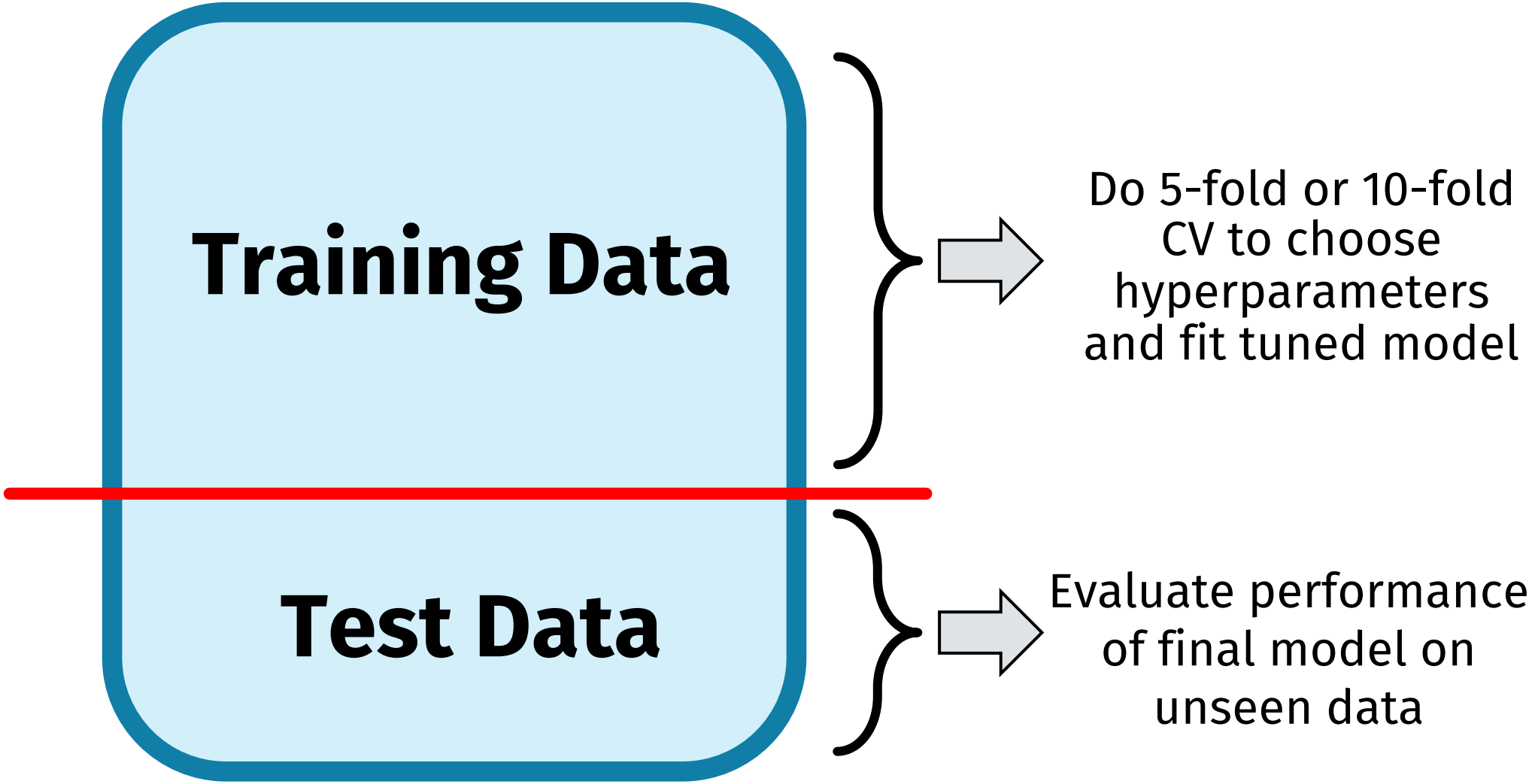
	OLS	Ridge	Lasso	Elastic Net
Advantages	• Simple • Can do inference (with all the caveats)	• Good with correlated features • MSE Existence Theorem (good for prediction)	• Feature selection and sparsity • Interpretability	• Compromise between ridge and Lasso
Disadvantages	• Major problems when p>n	• Dense model is not interpretable • No automatic feature selection	• Tends to choose 1 feature from correlated group • Can give worse prediction results if truth is not sparse	• Difficult to tune two parameters in practice

Back to Data Splitting

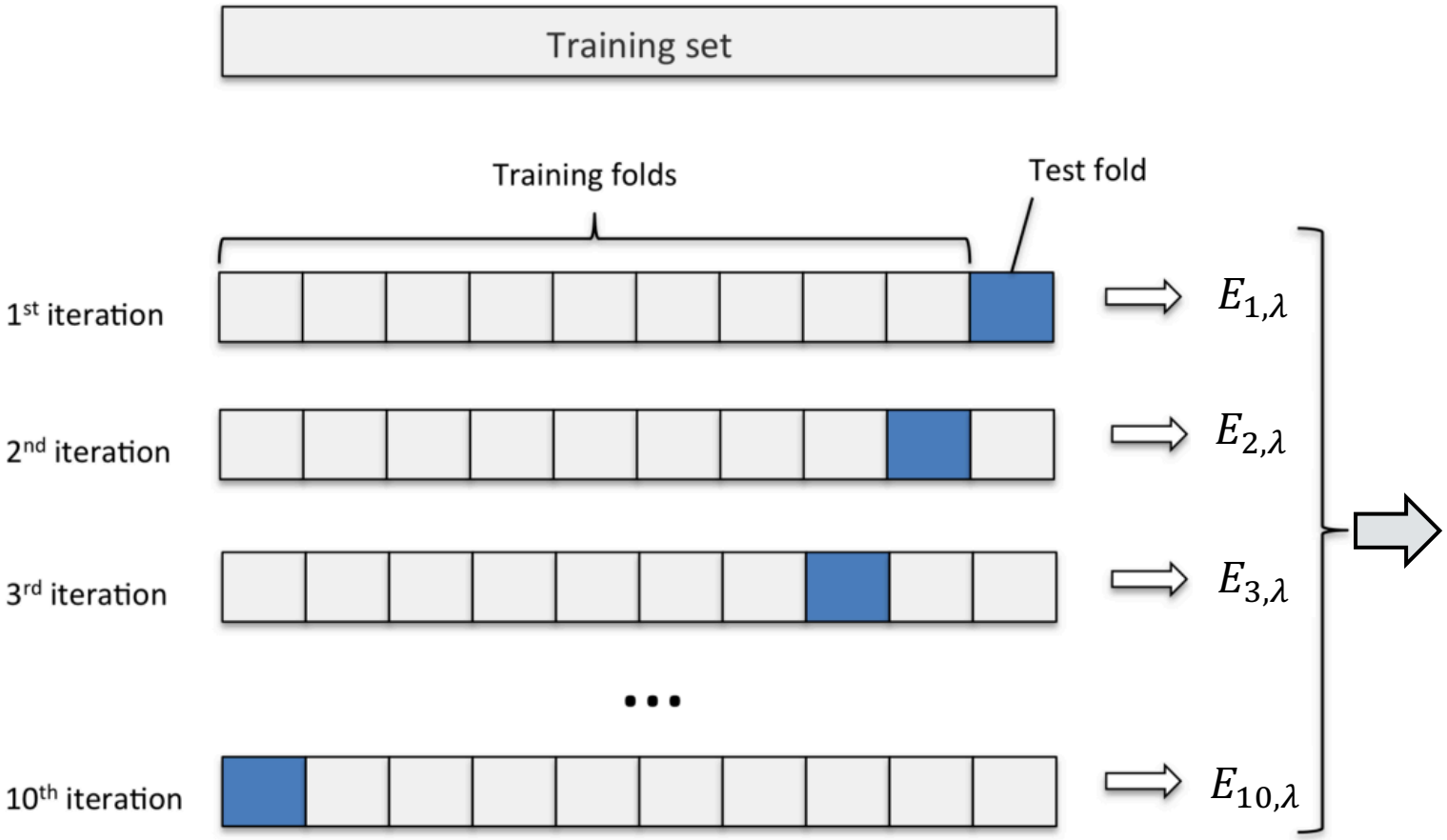
- **The simplest case:**

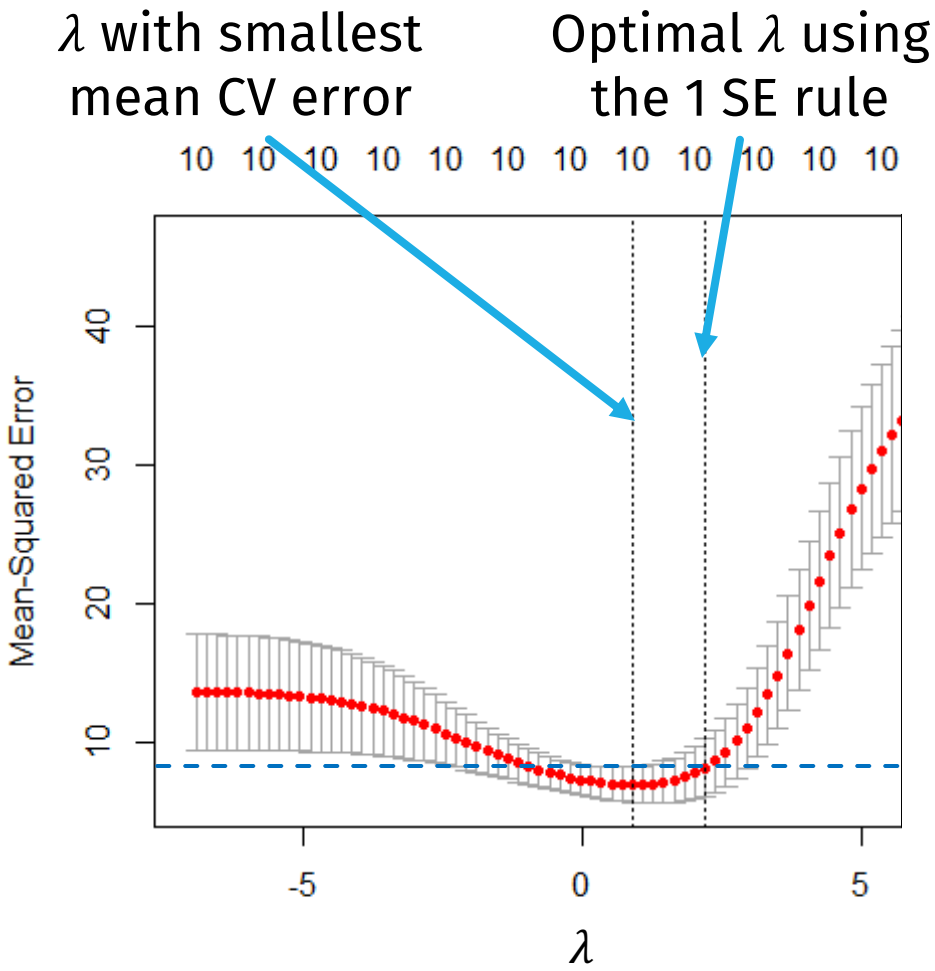


Data Splitting + Tuning Hyperparameters

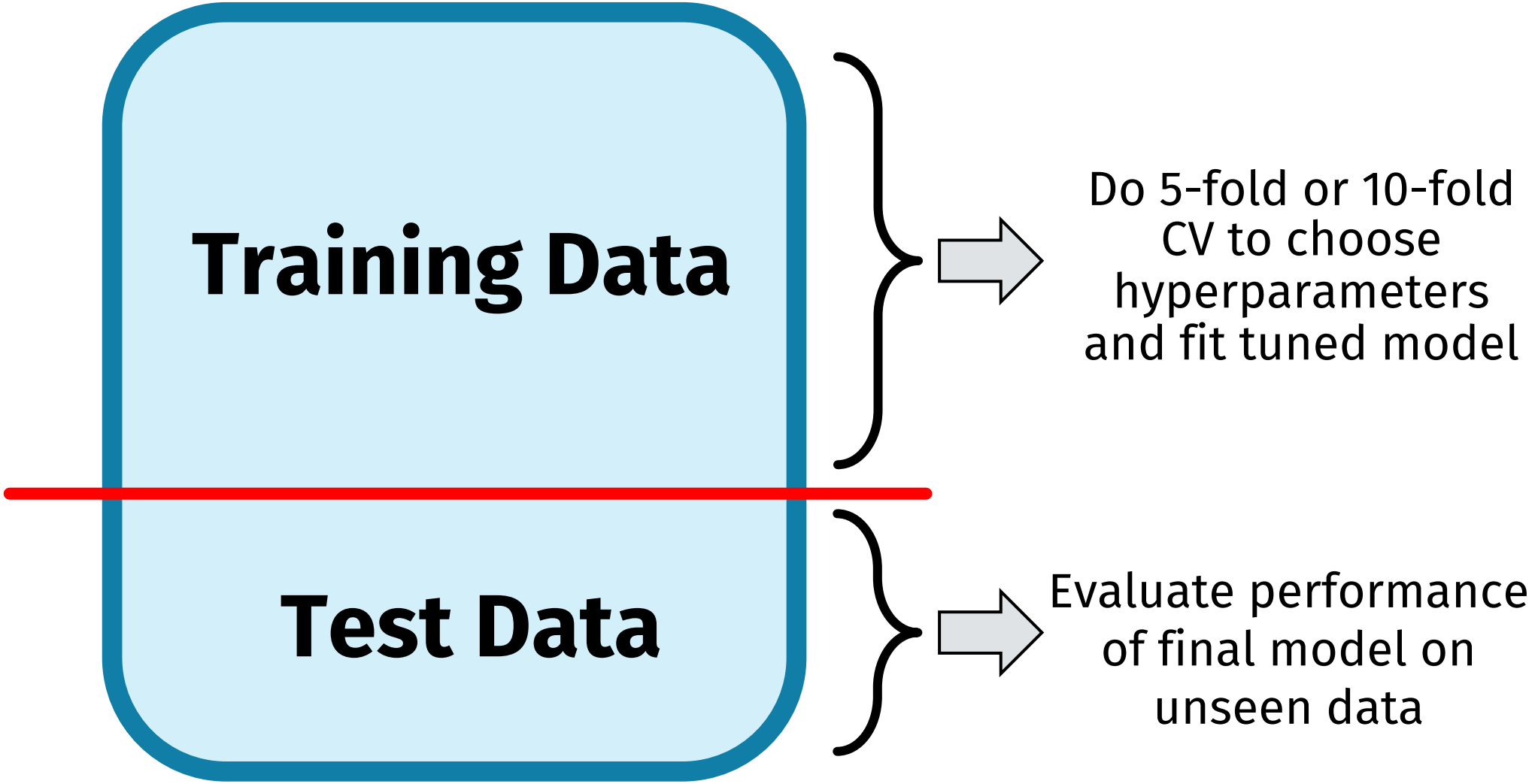


K-fold Cross Validation for choosing hyperparameters

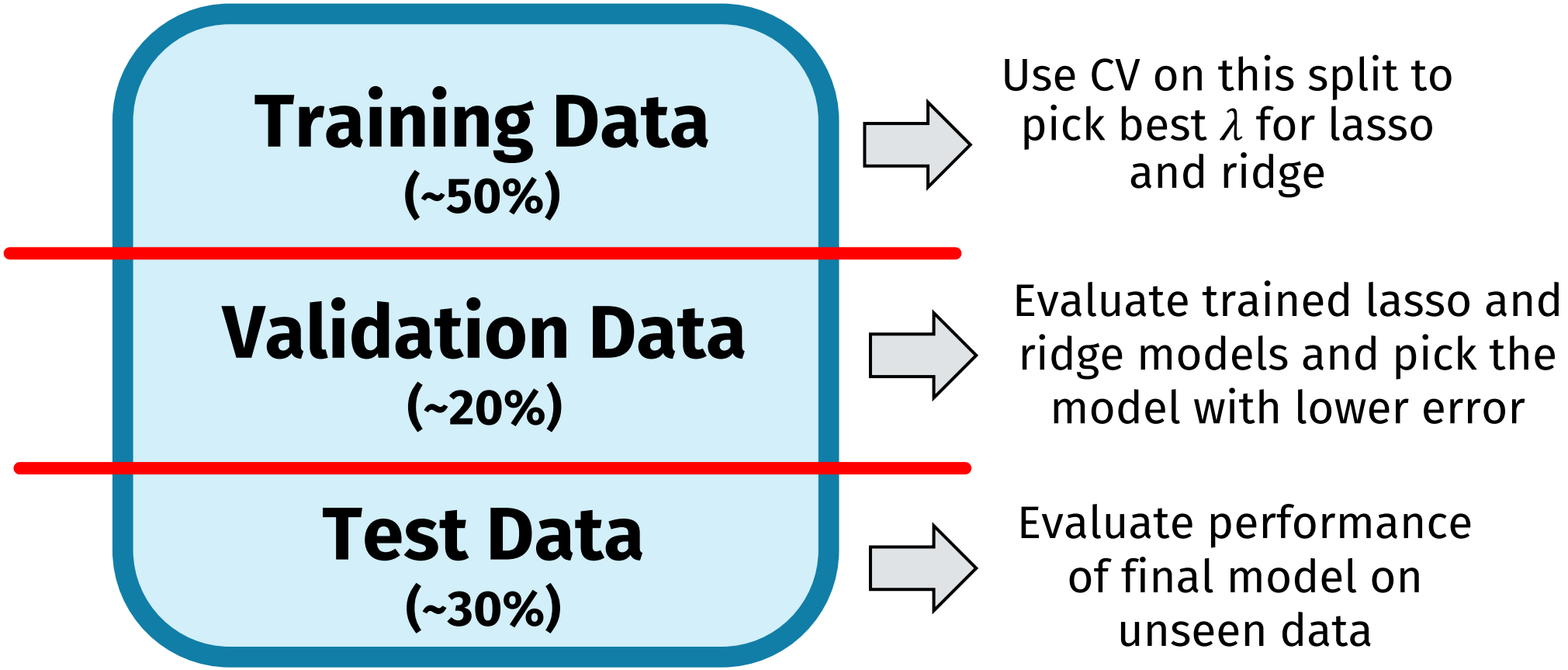




Data Splitting + Tuning Hyperparameters



Data Splitting + Tuning Hyperparameters + Multiple Methods



- *Can* repeat this data splitting B times to get a variance estimate of the test error
- This gives you an unbiased estimate of the prediction error for the **statistical learning process**, NOT a specific model

Let’s implement this pipeline in R using the ozone data

1. Pick two (or more) regression methods and an error metric
   - Ex: ridge, lasso, elastic net, SCAD, MCP, etc.
2. Split the data into training/validation/test splits
3. On the training set, use CV to tune hyperparameters for each method under consideration
4. Evaluate the tuned methods on the validation set
5. Report the test error of your “best” method
6. Time permitting, repeat steps 2-5 B times to get a variance estimate
7. Time permitting, perform EDA to “diagnose” the errors – are there any observations that are particularly difficult to predict? Any common mistakes?