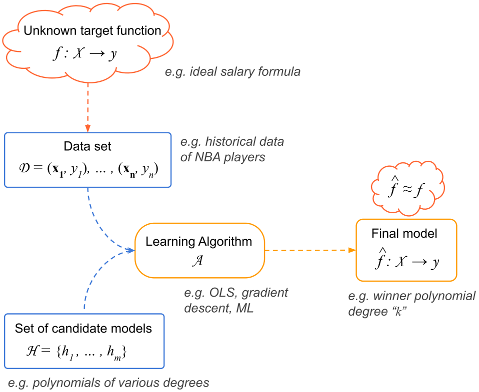
Learning Phases

Ngoc Hoang Luong

University of Information Technology (UIT), VNU-HCM

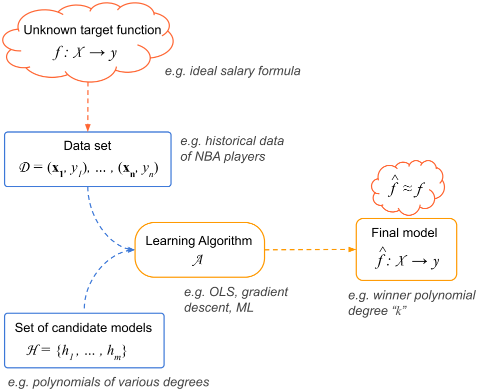
May 27, 2024

Review: Theoretical Framework for Supervised Learning



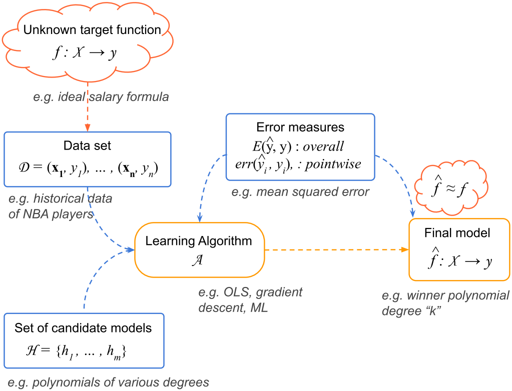
- The target function f is unknown, and we can never really "discover" f. We can only find a good enough approximation to f by estimating $\hat{f}$.
- The idea of a good approximation $\hat{f} \approx f$ is also theoretical because we don't know f.

Review: Theoretical Framework for Supervised Learning



- The dataset $\mathcal{D}$ is influenced by the unknown target function.
- The hypothesis set $\mathcal{H}$ is the type of ML models that we want to try out (e.g., linear models, polynomial models, non-parametric models, etc.)

Review: Theoretical Framework for Supervised Learning



- The learning algorithm $\mathcal{A}$ is the set of instructions to be carried out when learning from data with the individual error function $err()$.
- The overall measure of error $E()$ is used to determine which model $h()$ is the best approximation $\hat{f}()$ to the target model $f()$.

Review: Theoretical Framework for Supervised Learning

- In ML, the individual error function is known as the **loss function**:
  - Squared error: $err(\hat{f}, f) = (\hat{y}_i - y_i)^2$
  - Absolute error: $err(\hat{f}, f) = |\hat{y}_i - y_i|$
  - Misclassification error: $err(\hat{f}, f) = \mathbf{1}[\hat{y}_i \neq y_i]$
  - ...
- The overall measure of error is the **cost function** or **risk**:
  1. **In-sample Error** E_{in}: the average of individual errors from data points of the in-sample data $\mathcal{D}_{in}$:

$$E_{in}(\hat{f}, f) = \frac{1}{n} \sum_i err(\hat{f}(\mathbf{x}_i), f(\mathbf{x}_i))$$

  2. **Out-of-sample Error** E_{out}: the theoretical mean, or expected value, of the individual errors over the entire input space:

$$E_{out}(\hat{f}, f) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[err(\hat{f}(\mathbf{x}), f(\mathbf{x})) \right]$$

  The point $\mathbf{x}$ denotes a general data point in the input space $\mathcal{X}$.

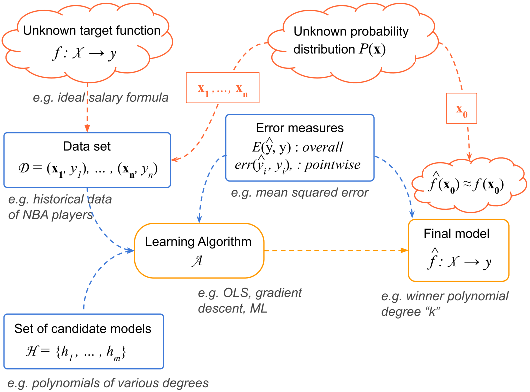
Review: Theoretical Framework for Supervised Learning

- A good approximation $\hat{f} \approx f$ means that $E_{out}(\hat{f}, f) \approx 0$.
- However, we never have access to E_{out} because out-of-sample data is theoretical: taking expectation over the entire input space $\mathcal{X}$.
- We solve this problem via:

$$E_{out}(\hat{f}) \approx 0 \Rightarrow \begin{cases} E_{in}(\hat{f}) \approx 0 & \text{practical result} \\ E_{out}(\hat{f}) \approx E_{in}(\hat{f}) & \text{theoretical result} \end{cases}$$

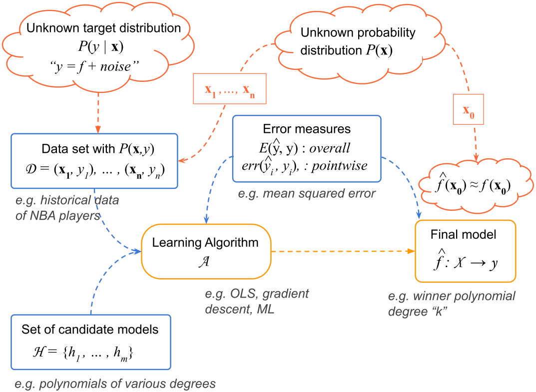
- The first goal is achievable because we have access to our training data D_{in}.
- To achieve the second goal, we need to assume a probability distribution P over the input space $\mathcal{X}$: $\mathbf{x}_0 \sim P$. We need $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are independent identically distributed (i.i.d.) samples from this distribution P.

Review: Theoretical Framework for Supervised Learning



$$E_{out}(\hat{f}) \approx 0 \Rightarrow \begin{cases} E_{in}(\hat{f}) \approx 0 & \text{practical result} \\ E_{out}(\hat{f}) \approx E_{in}(\hat{f}) & \text{theoretical result} \end{cases}$$

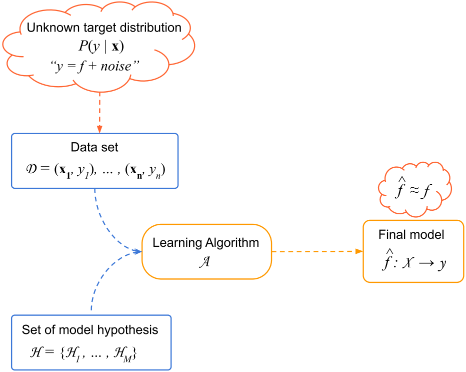
Review: Theoretical Framework for Supervised Learning



- In practice, there will be some noise: $y = f(\mathbf{x}) + \varepsilon$. We want to learn a **target conditional distribution** $P(y|\mathbf{x})$.
- Our data follow a joint probability distribution:

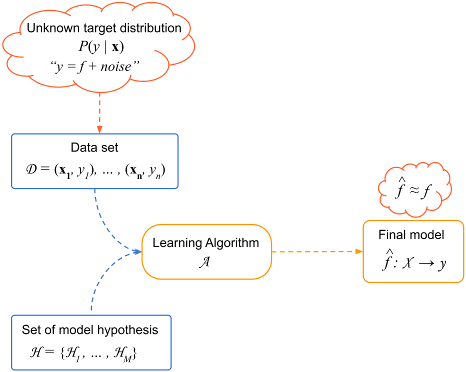
$$P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$$

Learning Phases



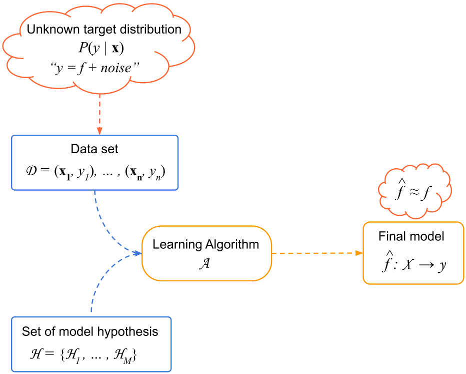
- We need to fit different models: i.e., working with different classes of hypothesis. For example: principal component regression $\mathcal{H}_1$, ridge regression $\mathcal{H}_2$, LASSO $\mathcal{H}_3$.
- Each type of models has tuning parameters that need to be determined through trial-error steps.

Learning Phases



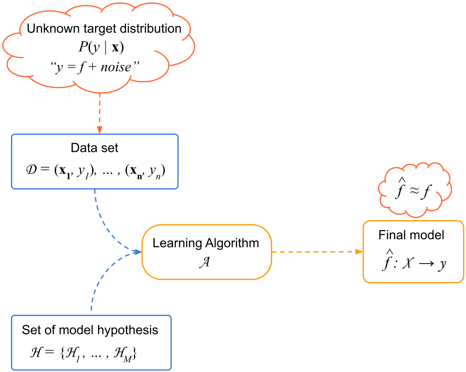
- For each hypothesis class, we need to find the optimal tuning parameter, which involves choosing a finalist model: the best principal component regression, the best polynomial regression model, etc.
- Among these finalist models, we need to select the best model $\hat{f}$.

Learning Phases



- Finally, we need to measure the predicting performance of the final model $\hat{f}$: measuring how the model behaves with out-of-sample data points.

Learning Phases



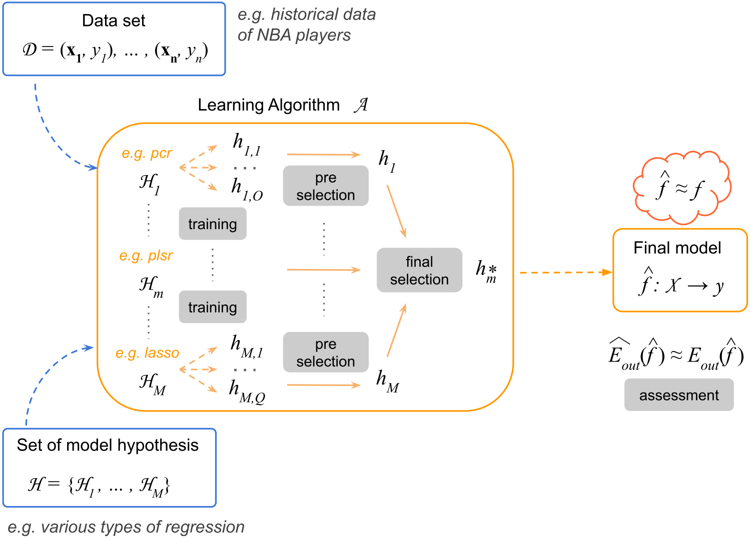
Three phases in supervised learning:

1. Model training
2. Model selection
3. Model assessment

Learning Phases

1. Training: We need to consider different classes of hypothesis $\mathcal{H}_1, \ldots, \mathcal{H}_m, \ldots, \mathcal{H}_M$. Given some data, for each hypothesis class $\mathcal{H}_m$, we fit a model h_m.
2. Selection: We choose a model from a set of candidate models.
   - Pre-Selection: Choose a finalist model from a set of models belonging to a certain class of hypothesis.
   - Final-Selection: Choose a final model among a set of finalist models.
3. Assessment: Given a final model $\hat{f}$, how can we estimate the out-of-sample error $E_{out}(\hat{f})$?

Learning Phases



What data should we use to perform each task?

Model Assessment

- Given a final model h, how to measure its prediction quality?
- In-sample error E_{in} measures **the resubstitution power** of the model h (i.e., how well the model fits the learning data).

$$E_{in}(h) = \frac{1}{|\mathcal{D}_{in}|} \sum_{\mathbf{x}_i \in \mathcal{D}_{in}} err(h(\mathbf{x}_i) = \hat{y}_i, y_i)$$

- Out-of-sample error E_{out} measures **the generalization power** of the model h:

$$E_{out}(h) = \mathbb{E}[err(h(\mathbf{x}_0) = \hat{y}_0, y_0)]$$

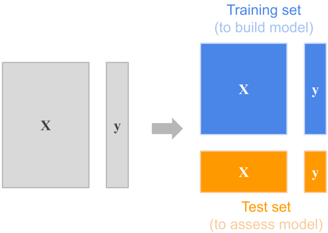
- Because we don't have access to out-of-sample dataset $\mathcal{D}_{out}$, we need to find a proxy set $\mathcal{D}_{proxy} \subset \mathcal{D}_{out}$ to compute E_{proxy} and use it to approximate E_{out}:

$$\underbrace{\hat{E}_{out}(h)}_{\mathcal{D}_{proxy}} \approx \underbrace{E_{out}(h)}_{\mathcal{D}_{out}}$$

Model Assessment - Holdout Test Set

- How can we estimate E_{out}?
- Our dataset $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. If we use all n data points to train/fit a model, then we have a good measure for E_{in}, but we don't have any out-of-sample data points to get an **honest** approximation $\hat{E}_{out}$ of E_{out}.
- Why not use E_{in} as an estimate $\hat{E}_{out}$ of E_{out}?
- We may have a model that produces very small in-sample error $E_{in} \approx 0$ but does not perform well on out-of-sample data points.
- Given that the only available dataset is $\mathcal{D}$, the Holdout Method splits $\mathcal{D}$ into two subsets:
    1. **The training set** $\mathcal{D}_{train}$ for training/fitting the model.
    2. **The test set** $\mathcal{D}_{test}$ to be used a proxy for $\mathcal{D}_{out}$ for the assessment purpose.

Model Assessment - Holdout Test Set

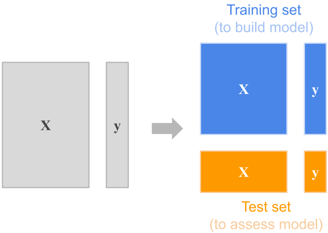


Taking random sample of size a without replacement from $\mathcal{D}$:

$$\mathcal{D} \rightarrow \begin{cases} \mathcal{D}_{train} & \text{size } n - a \\ \mathcal{D}_{test} & \text{size } a \end{cases}$$

We fit a model using $\mathcal{D}_{train}$ to obtain a model $h^{-}(\mathbf{x})$

Model Assessment - Holdout Test Set



Taking random sample of size a without replacement from $\mathcal{D}$:

$$\mathcal{D} \rightarrow \begin{cases} \mathcal{D}_{train} & \text{size } n - a \\ \mathcal{D}_{test} & \text{size } a \end{cases}$$

With the remainder points in $\mathcal{D}_{test}$, we can measure the performance of $h^{-}(\mathbf{x})$ as:

$$E_{test}(h^{-}) = \frac{1}{a} \sum_{l=1}^{a} err(h^{-}(\mathbf{x}_l), y_l); \quad (\mathbf{x}_l, y_l) \in \mathcal{D}_{test}$$

Model Assessment - Holdout Test Set

- If $\mathcal{D}_{test}$ is a representative sample of $\mathcal{D}_{out}$, E_{test} gives an unbiased estimate of E_{out}. Why?
- Consider an out-of-sample point $(\mathbf{x}_0, y_0) \in \mathcal{D}_{test}$, given an error function, we can measure:

$$err(h(\mathbf{x}_0), y_0)$$

- Let's determine its expectation over the input space $\mathcal{X}$:

$$\mathbb{E}_{\mathcal{X}}[err(h(\mathbf{x}_0), y_0)]$$

- This is the definition of out-of-sample error E_{out}. Therefore $err(h(\mathbf{x}_0), y_0)$ is an **unbiased estimate** of the out-of-sample error.

Model Assessment - Holdout Test Set - Expectation

- Let's consider a set $\mathcal{D}_{test} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_a, y_a)\}$ containing $a > 1$ points. We can get E_{test} by

$$E_{test}(h) = \frac{1}{a} \sum_{l=1}^{a} err(h(\mathbf{x}_l), y_l)$$

- Is $E_{test}(h)$ an unbiased estimate of $E_{out}(h)$?

$$\begin{aligned}
\mathbb{E}_{\mathcal{X}}[E_{test}(h)] &= \mathbb{E}_{\mathcal{X}}\left[\frac{1}{a} \sum_{l=1}^{a} err(h(\mathbf{x}_l), y_l)\right] \\
&= \frac{1}{a} \sum_{l=1}^{a} \mathbb{E}_{\mathcal{X}}[err(h(\mathbf{x}_l), y_l)] \\
&= \frac{1}{a} \sum_{l=1}^{a} E_{out}(h) = E_{out}(h)
\end{aligned}$$

- $E_{test}(h)$ is an **unbiased estimate** of $E_{out}(h)$. What about its **variance**?

Model Assessment - Holdout Test Set - Variance

- The variance[1] of $E_{test}(h)$ is given by:

$$\begin{aligned} Var[E_{test}(h)] &= Var\left[\frac{1}{a}\sum_{l=1}^{a} err(h(\mathbf{x}_l), y_l)\right] \\ &= \frac{1}{a^2}\sum_{l=1}^{a} Var[err(h(\mathbf{x}_l), y_l)] = \frac{s^2}{a} \end{aligned}$$

  Assume that the variance $Var[err(h(\mathbf{x}_0), y_0)] = s^2$ is constant.
- As we increase the number a of test data points, the variance of the estimator $E_{test}(h)$ decreases. The more test points we use, the more reliably $E_{test}(h)$ estimates $E_{out}(h)$.
- But the larger a is, the smaller the training dataset will be.

[1] https://eli.thegreenplace.net/2009/01/07/variance-of-the-sum-of-independent-variables

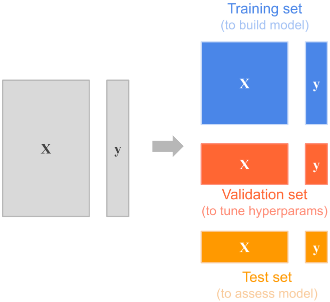
Model Assessment - Holdout Algorithm

1. Compile the available data into $\mathcal{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$.
2. Choose $a \in \mathbb{N}$ points from $\mathcal{D}$ to get a test set $\mathcal{D}_{test}$, and place the remaining $n - a$ points into the training set $\mathcal{D}_{train}$.
3. Use $\mathcal{D}_{train}$ to fit a particular model $h^{-}(\mathbf{x})$.
4. Measure the performance of h^{-} using $\mathcal{D}_{test}$:

$$E_{test}(h^{-}) = \frac{1}{a} \sum_{l=1}^{a} err(h^{-}(\mathbf{x}_l), y_l)$$

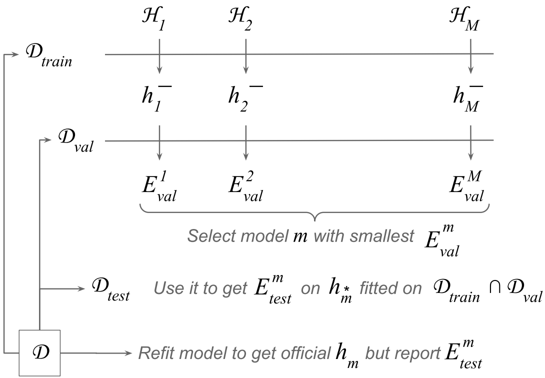
   where $(\mathbf{x}_l, y_l) \in \mathcal{D}_{test}$.
5. Generate the final model $\hat{h}$ by refitting h^{-} to the entire dataset $\mathcal{D}$.

Model Selection - Three-Way Holdout Method



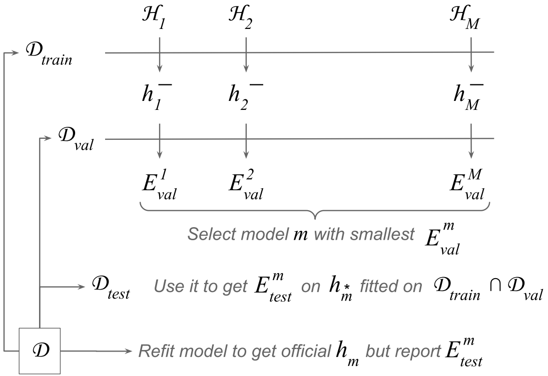
- The test set $\mathcal{D}_{test}$ can only be used at the end of the learning process to quantify the generalization error of the final model. We don't use this set to make any **learning decisions**.
- Thus, we create a validation set $\mathcal{D}_{val}$ for selecting the final model from a set of finalist models.

Model Selection



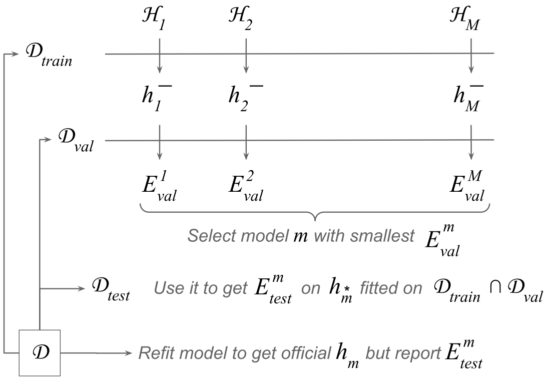
- $\mathcal{H}_m$ represents the m-th hypothesis class. h_m^- is the finalist model from class $\mathcal{H}_m$.
- After a finalist model h_m^- has been pre-selected for each hypothesis class, we use $\mathcal{D}_{val}$ to compute validation error E_{val}^m.

Model Selection



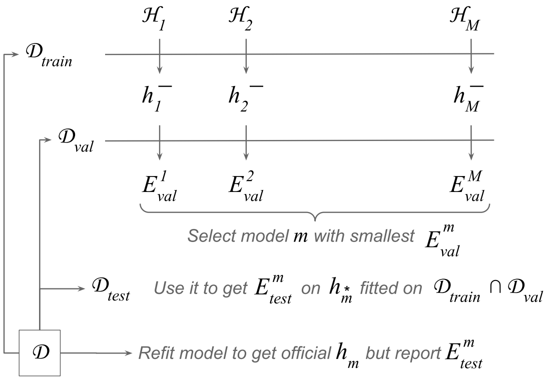
- After selecting the best model h_m^- with the smallest E_{val}^m, a model h_m^* is fitted using $\mathcal{D}_{train} \cap \mathcal{D}_{val}$

Model Selection



- The performance of h_m^* is assessed by using $\mathcal{D}_{test}$ to obtain $E_{test}^m = E_{test}(h_m^*)$.
- Finally, the "official" model is the model fitted on the entire dataset $\mathcal{D}$.

Model Selection



- Choosing the best model (smallest E_{val}^m) is a **learning decision**.
- Thus, E_{val}^m is a **biased estimate** of E_{out}.
- The larger the test and validation sets are, the more reliable the estimates of the out-of-sample performance. But the smaller the training set will be.

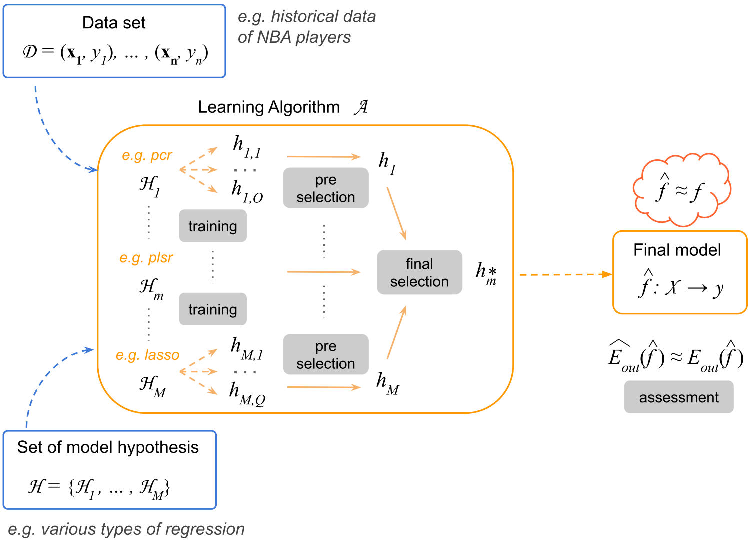
Model Training

- For each hypothesis class, we need to train all candidate models and pre-select the finalist models using the training set $\mathcal{D}_{train}$.
- For example, $\mathcal{H}_m$ is principal component regression (PCR), we need to train several models

$$h_{1,m}, h_{2,m}, \ldots, h_{q,m}$$

  with $1, 2, \ldots, q$ are the number of principal components (i.e., **the tuning parameter** - hyperparameter).
- Using $\mathcal{D}_{train}$ to fit all possible PCR models, and also to choose the one with the smallest error E_{train} can cause overfitting.
- We could use $\mathcal{D}_{val}$ to pre-select the finalist model for $\mathcal{H}_m$, but then we don't have **fresh** data points for the final-selection phase.
- $\longrightarrow$ Resampling methods.

Learning Phases



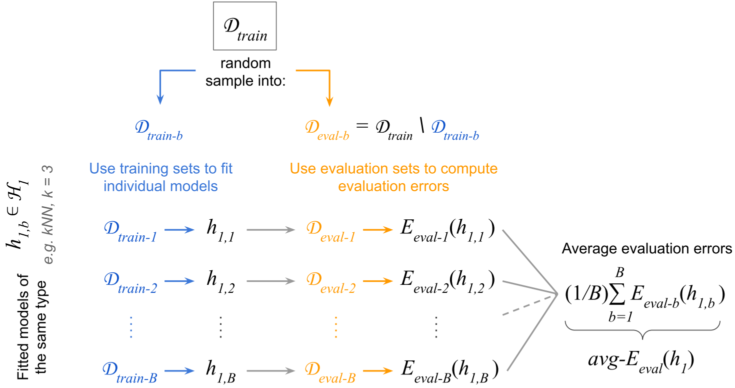
Resampling Approaches - General Sampling Blueprint

- We have a limited dataset.
- We want to generate multiple training sets, multiple evaluation sets to be used for both training phase and the pre-selection phase.
- The training set is the one we usually apply resampling.
- For example, we want to fit a k-Nearest Neighbor (kNN) model with $k = 3$ neighbors.
- We randomly split the training set into two subsets:
  1. $\mathcal{D}_{train-b}$ to train a model $h_{1,b}$.
  2. $\mathcal{D}_{eval-b} = \mathcal{D}_{train} \setminus \mathcal{D}_{train-b}$ to evaluate $h_{1,b}$.
- The procedure is repeated B times. We average all the evaluation errors to get a single measure indicating the performance of the particular type of trained models:

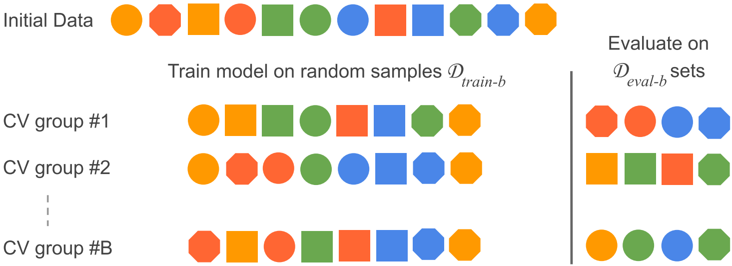
$$E_{eval} = \frac{1}{B} \sum_{b=1}^{B} E_{eval-b}$$

- The average measure E_{eval} would be the typical performance of 3-NN models.

Resampling Approaches



Monte Carlo Cross Validation

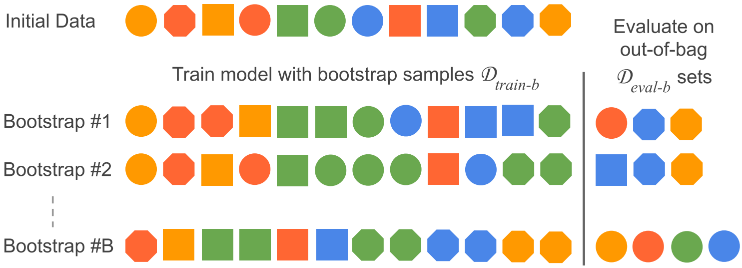


Monte Carlo Cross Validation - Repeated Holdout

1. Compile training dataset $\mathcal{D}_{train} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
2. Repeat the Holdout method B times (B is very large, e.g., 500).
   - Generate $\mathcal{D}_{train-b}$ by sampling $n - r$ elements without replacement from $\mathcal{D}_{train}$.
   - Generate h_b^- by fitting a model to $\mathcal{D}_{train-b}$.
   - Compute $E_{eval-b} = \frac{1}{r} \sum_i err(h_b^-(\mathbf{x}_i), y_i)$; where $(\mathbf{x}_i, y_i) \in \mathcal{D}_{eval-b}$.
3. Obtain an overall value for E_{eval} by averaging the E_{eval-b} values:

$$E_{eval} = \frac{1}{B} \sum_{b=1}^{B} E_{eval-b}$$

Bootstrap Method



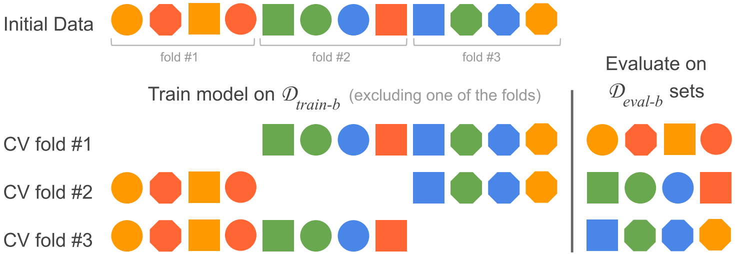
- A bootstrap sample is a random sample of the data taken **with replacement**.
- The bootstrap sample has the same size as $\mathcal{D}_{train}$.
- Some individuals are reprensented multiple times in the bootstrap while others are not selected at all. The unselected individuals are out-of-bag elements.

Bootstrap Algorithm

1. Compile training dataset $\mathcal{D}_{train} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
2. Repeat the following steps for $b = 1, 2, \ldots, B$ where B is very large (e.g., 500):
   - Generate $\mathcal{D}_{train-b}$ by sampling n times **with replacement** from $\mathcal{D}_{train}$.
   - Generate $\mathcal{D}_{eval-b}$ by compiling out-of-bag elements ($\mathcal{D}_{eval-b}$ should not contain repeated elements).
   - Compute $E_{eval-b} = \frac{1}{r_b} \sum_i err(h_b^-(\mathbf{x}_i), y_i)$; where $(\mathbf{x}_i, y_i) \in \mathcal{D}_{eval-b}$ and r_b is the size of $\mathcal{D}_{eval-b}$.
3. Obtain an overall value for bootstrap E_{eval} by averaging the E_{eval-b} values:

$$E_{eval}^{bootstrap} = \frac{1}{B} \sum_{b=1}^{B} E_{eval-b}$$

K-Fold Cross Validation



- Split $\mathcal{D}_{train}$ into K sets of equal size. Each subset is called a **fold**.
- At each iteration b, a fold is held out for the evaluation purpose $\mathcal{D}_{eval-b}$, and the remaining folds are merged into $\mathcal{D}_{train-b}$ for the training purpose.
- At the end $b = K$, the cross-validation error E_{cv} is defined as the average E_{eval}, which is computed from all evaluation errors E_{eval-b}.

Leave-One-Out Cross Validation (LOOCV)

A special case of K-Fold cross validation is when each fold contains one single data point, i.e., $K = n$.

1. Compile training dataset $\mathcal{D}_{train} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$
2. For $i = 1, 2, \ldots, n$:
    - Generate the i-th training dataset by removing the i-th element from $\mathcal{D}_{train}$: $\mathcal{D}_{train-i} = \mathcal{D}_{train} \setminus \{(\mathbf{x}_i, y_i)\}$
    - Fit the model to $\mathcal{D}_{train-i}$ to obtain h_i^-.
    - Use the point $(\mathbf{x}_i, y_i)$ to compute $E_{eval-i} = err(h_i^-(\mathbf{x}_i), y_i)$
3. Obtain cross-validation error E_{cv} by averaging the individual errors:

$$E_{cv} = \frac{1}{n} \sum_{i}^{n} E_{eval-i}$$

Example - Tuning the parameter λ of Ridge Regression

The ridge regression coefficients:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$$

λ is a tuning parameter (i.e., a hyperparameter). How to find the finalist model of the hypothesis class of ridge regression models?

Example - Tuning the parameter λ of Ridge Regression

1. Split $\mathcal{D}_{train} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ into K folds:
$$\mathcal{D}_{train} = \mathcal{D}_{fold-1} \cup \mathcal{D}_{fold-2} \cup \ldots \cup \mathcal{D}_{fold-K}$$
2. Create K training sets:
$$\mathcal{D}_{train-1} = \mathcal{D}_{train} \setminus \mathcal{D}_{fold-1}; \mathcal{D}_{eval-1} = \mathcal{D}_{fold-1}$$
$$\mathcal{D}_{train-2} = \mathcal{D}_{train} \setminus \mathcal{D}_{fold-2}; \mathcal{D}_{eval-2} = \mathcal{D}_{fold-2}$$
$$\ldots\ldots$$
$$\mathcal{D}_{train-K} = \mathcal{D}_{train} \setminus \mathcal{D}_{fold-K}; \mathcal{D}_{eval-K} = \mathcal{D}_{fold-K}$$
3. For $\lambda_l = 0.001, 0.002, \ldots, \lambda_L$:
   1. For $b = 1, 2, \ldots, K$:
      - Fit ridge regression model $h_{l,b}$ with λ_l on $\mathcal{D}_{train-b}$
      - Compute $E_{eval-b}(h_{l,b})$ on $\mathcal{D}_{eval-b}$
   2. Compute cross-validation error $E_{cv-l} = \frac{1}{K}\sum_{b=1}^{K} E_{eval_b}(h_{l,b})$
4. Compare all L cross-validation errors $E_{cv-1}, E_{cv-2}, \ldots, E_{cv-L}$ and choose the λ^* that yields the smallest E^*_{cv}.
5. Use λ^* to fit the finalist model $\mathbf{w} = (\mathbf{X}^\top\mathbf{X} + \lambda^*\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}$.