Chapter 8: Cross Validation

CMPUT 467/567: Machine Learning II

Winter 2026

Generalization Error

- Generalization error (GE) for a a function f is the expected cost
- $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)]$ where expected over RVs X, Y sampled from joint distribution $p(x, y)$
  - Or equivalently $x \sim p_x$ and $y \sim p(y \mid x)$ where $p(x, y) = p(y \mid x)p_x(x)$
- Cost depends on the problem setting

Some costs for regression

- $\text{cost}(\hat{y}, y) = (\hat{y} - y)^2$ (squared error)
- $\text{cost}(\hat{y}, y) = |\hat{y} - y|$ (absolute error)
- $\text{cost}(\hat{y}, y) = \dfrac{|\hat{y} - y|}{|y|}$ (absolute percentage error)
- Multivariate version per dimension
    - e.g., $\text{cost}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{k=1}^{m} |\hat{y}_k - y_k|$

Some costs for classification

- $\text{cost}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 1 & \text{if } \hat{y} \neq y\,. \end{cases}$ (0-1 cost)

Some costs for classification

- $\text{cost}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 1 & \text{if } \hat{y} \neq y\,. \end{cases}$ (0-1 cost)

- for $\mathscr{Y} = \{0,1\}$, $\text{cost}(\hat{y}, y) = \begin{cases} 0 & \text{if } \hat{y} = y, \\ 2 & \text{if } y = 0 \quad \text{(false positive)} \\ 1000 & \text{if } y = 1 \quad \text{(false negative)} \end{cases}$

    - e.g., $\hat{y} = 0$ do not send for (disease) test, $\hat{y} = 1$ do send for test

- What is an asymmetric cost example for 3-class classification?

Estimating GE with a Test Set

- Goal is to estimate generalization error (GE) for a learned function f
- Simplest option: split dataset $\mathscr{D}$ into training $\mathscr{D}_{\text{tr}}$ and test set $\mathscr{D}_{\text{test}}$
- Q1: For logistic regression, do we compute the cross-entropy on the test set or the 0-1 loss on the test set?

Estimating GE with a Test Set

- Goal is to estimate generalization error (GE) for a learned function f
- Simplest option: split dataset $\mathscr{D}$ into training $\mathscr{D}_{\text{tr}}$ and test set $\mathscr{D}_{\text{test}}$
- Issue 1: How much data do we use for train and test?
- Tension: want more data for $\mathscr{D}_{\text{tr}}$ to learn a good function f, but also want more data for $\mathscr{D}_{\text{test}}$ to get a good GE estimate
- Can we use all of $\mathscr{D}$ to train f, and still get an estimate of GE for it?

Estimating GE via Cross Validation

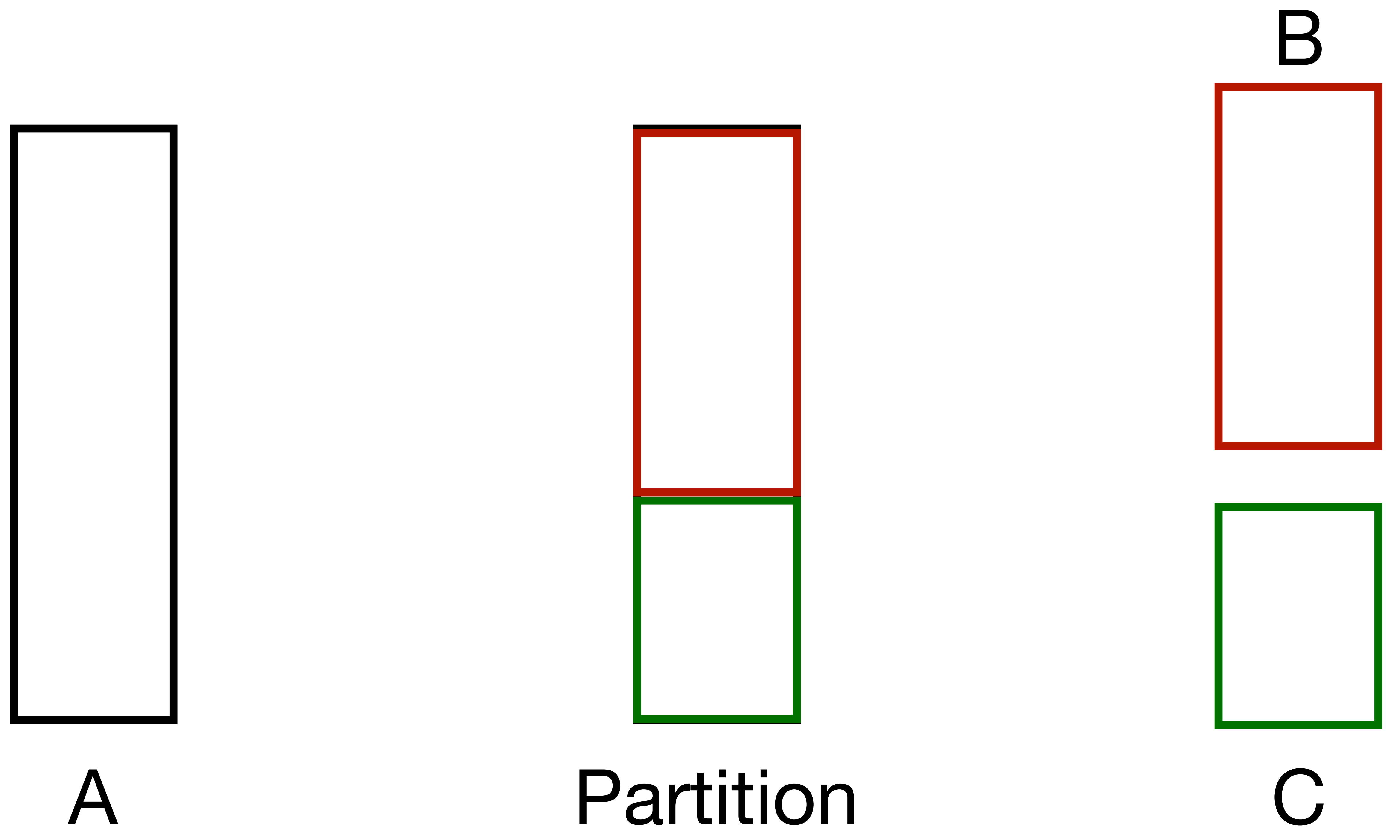
- Cross-validation let’s us use the training data for training and evaluation
  - But, what?!?
- Unlike having a separate test set, we get a biased estimator, but still a good one
- **The idea**: we use unbiased evaluations of **different** functions

Which functions?

- Step 1: Get k partitions of the dataset, $\mathscr{D}^{(i)}_{\text{tr}}, \mathscr{D}^{(i)}_{\text{test}}$

What is a partition?

- A partition of a set A is a split into two disjoint sets B, C
- $A = B \cup C$ where $B \cap C = \varnothing$ (i.e., they are disjoint, they don't share any elements)

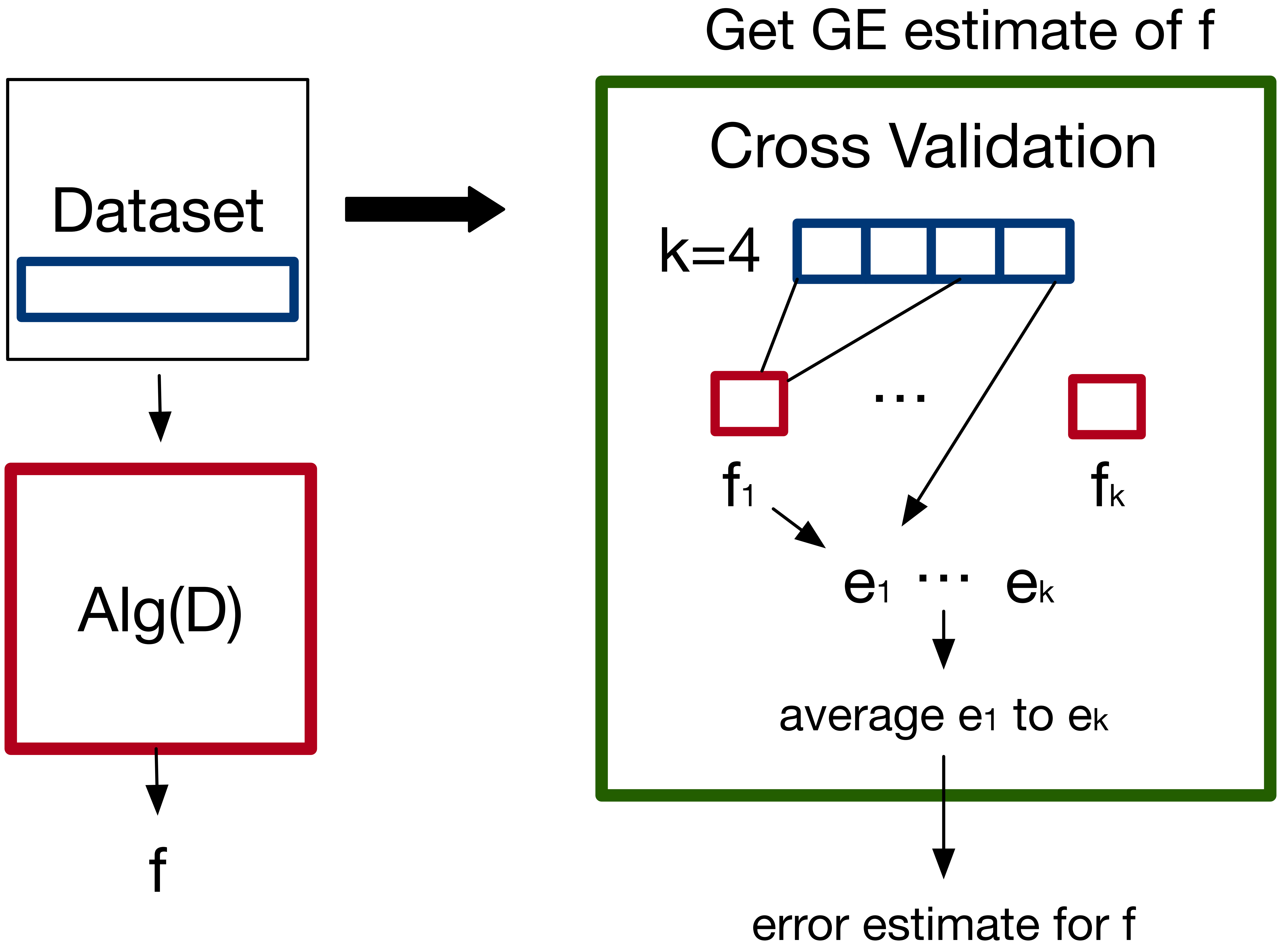


Which functions?

- Step 1: Get k partitions of the dataset, $\mathcal{D}^{(i)}_{\text{tr}}, \mathcal{D}^{(i)}_{\text{test}}$
- Train a function f_i on training set $\mathcal{D}^{(i)}_{\text{tr}}$ and evaluate on test $\mathcal{D}^{(i)}_{\text{test}}$ to get error e_i
- We now have functions $f_1, f_2, \ldots, f_k$ with corresponding errors $e_1, e_2, \ldots, e_k$
- We actually throw away these functions and only use the errors to get our GE estimate for the function f learned on the entire dataset $\mathcal{D}$,

$$\hat{\text{GE}}(f) = \frac{1}{k}\sum_{i=1}^{k} e_i$$

Cross validation



Step 1: Learn f on the entire dataset

Step 2: Do CV to estimate the GE for f

Step 2 consists of
1. Get k partitions of the dataset, to get k training and test splits

2. For every i = 1 to k,
train $f_i = \text{Alg}(\mathscr{D}_{tr}^{(i)})$ and
compute error e_i on $\mathscr{D}_{test}^{(i)}$

3. Get average error $\frac{1}{k}\sum_i e_i$

Why is this a biased estimate of GE?

- $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} e_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[e_i\right]$
- It is not likely that $\mathbb{E}\left[e_i\right] = \text{GE}(f_i)$ equals $\text{GE}(f)$, because the functions f_i and f are not the same. But, their generalization error should be pretty similar
- Q: We contrasted to using a training test split, where we train f on the training set and the get the GE estimate on the test. Is this unbiased?

Why is this a biased estimate of GE?

- $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} e_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[e_i\right]$
- It is not likely that $\mathbb{E}\left[e_i\right] = \text{GE}(f_i)$ equals $\text{GE}(f)$, because the functions f_i and f are not the same. But, their generalization error should be pretty similar
- Q: We contrasted to using a training test split, where we train f on the training set and the get the GE estimate on the test. Is this unbiased?
  - Yes

Why is this a biased estimate of GE?

- $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} e_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[e_i\right]$
- It is not likely that $\mathbb{E}\left[e_i\right] = \text{GE}(f_i)$ equals $\text{GE}(f)$, because the functions f_i and f are not the same. But, their generalization error should be pretty similar
- Q: We contrasted to using a training test split, where we train f on the training set and the get the GE estimate on the test. Is this unbiased?
- Q: What if we split up the data into train and validation, trained $\tilde{f}$ on the train and got error e on validation. Then we train f on the full data set (train + validation). Is the error estimate e an unbiased estimate of the GE of f? What about of $\tilde{f}$?

Why is this a biased estimate of GE?

- $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} e_i\right] = \frac{1}{k}\sum_{i=1}^{k}\mathbb{E}\left[e_i\right]$
- It is not likely that $\mathbb{E}\left[e_i\right] = \text{GE}(f_i)$ equals $\text{GE}(f)$, because the functions f_i and f are not the same. But, their generalization error should be pretty similar
- Q: We contrasted to using a training test split, where we train f on the training set and the get the GE estimate on the test. Is this unbiased?
- Q: What if we split up the data into train and validation, trained $\tilde{f}$ on the train and got error e on validation. Then we train f on the full data set (train + validation). Is the error estimate e an unbiased estimate of the GE of f? What about of $\tilde{f}$?
  - No for f, Yes for $\tilde{f}$

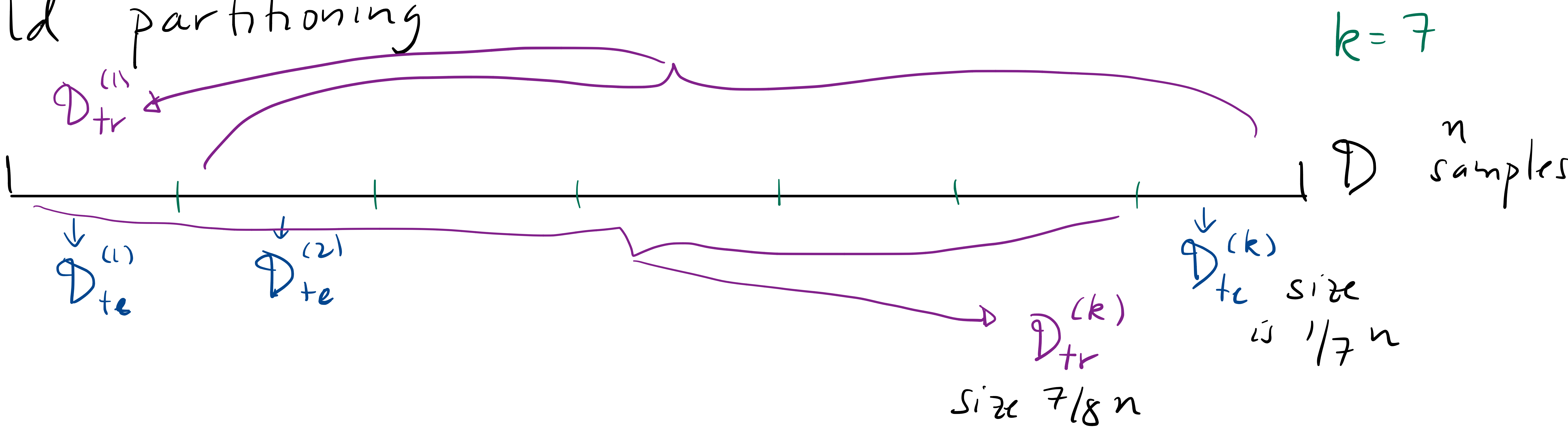
How do we get the k partitions?

- Partition means disjoint subsets that cover the data
- There are many ways we can get multiple train and test splits
- k-fold and repeated random subsampling (RSS) are two common ones

k-fold is one way to get partitioning

- Partition data into k folds/chunks
- Each fold is set to a test dataset, the training is union of the remaining folds

k-fold partitioning



RSS is another way to get a partitioning

- Randomly sample points for test dataset (without replacement), and set the rest to the training set
- Have to specify percentage for test **p** and number repeats **k**

RRS with percentage p for test

$k = 7$

$\mathcal{D}$

Randomly shuffle $\mathcal{D}$

Set first $(1-p)n$ as $\mathcal{D}_{tr}^{(1)}$

Set last pn as $\mathcal{D}_{te}^{(1)}$

. . .

Randomly shuffle $\mathcal{D}$

Set first $(1-p)n$ as $\mathcal{D}_{tr}^{(k)}$

Set last pn as $\mathcal{D}_{te}^{(k)}$

k-fold vs RSS

- k-fold
    - Partition data into k folds/chunks
    - Each fold is set to a test dataset, the training is union of the remaining folds
- Repeated random subsampling
    - Randomly sample points for test dataset (without replacement), and set the rest to the training set
    - Have to specify percentage for test p and number repeats k

k-fold partitioning

$k = 7$

$D^{(1)}_{tr}$

D n samples

$D^{(1)}_{te}$ $D^{(2)}_{te}$ $D^{(k)}_{te}$ size is $1/7\, n$

$D^{(k)}_{tr}$ size $7/8\, n$

RRS with percentage p for test

$k = 7$

D

Randomly shuffle D
Set first $(1-p)n$ as $D^{(1)}_{tr}$
Set last pn as $D^{(1)}_{te}$

. . .

Randomly shuffle D
Set first $(1-p)n$ as $D^{(k)}_{tr}$
Set last pn as $D^{(k)}_{te}$

How do we pick k?

- How is bias impacted by the choice of k for k-fold CV?
- How is bias impacted by the choice of k or p for RRS CV?

How do we pick k? (for bias)

- How is bias impacted by the choice of k for k-fold CV?
  - Bigger k means training set size (k-1)/k n closer to full dataset size n
  - Each f_i more similar to f learned on all the data
  - Extreme: leave-one-out CV, where train n functions!

How do we pick k? (for bias)

- How is bias impacted by the choice of k for k-fold CV?
    - Bigger k means less bias
- How is bias impacted by the choice of k or p for RRS CV?
    - Smaller p means training set size (1-p) n closer to full dataset size n
    - Each f_i more similar to f learned on all the data
    - Can get same behavior as leave-one-out k-fold CV, but do not need to learn n functions, k is independently chosen from p

How do we pick k?

- For **lower bias** pick **k large** for k-fold and **p smaller** for RRS
- **But** variance can increase with large k for k-fold or smaller p for RRS, as variance of errors larger (error is computed with smaller # of testing samples)
- And large k or smaller p means there is likely more covariance between errors

$$\mathrm{Var}\left[\bar{G}\right] = \frac{1}{k^2}\left(\sum_{j=1}^{k} \mathrm{Var}\left[\mathsf{err}^{(j)}\right] + \sum_{i,j} \mathrm{Cov}[\mathsf{err}^{(i)}, \mathsf{err}^{(j)}]\right)$$

How do we pick k?

- For **lower bias** pick **k large** for k-fold and **p smaller** for RRS
- **But** variance can increase with large k for k-fold or smaller p for RRS, as variance of errors larger (error is computed with smaller # of testing samples)
- And large k or smaller p means there is likely more covariance between errors
- Finally, large k is computationally expensive, so rarely set very big
- No clear answers, just some rules of thumb, usually pick interim k

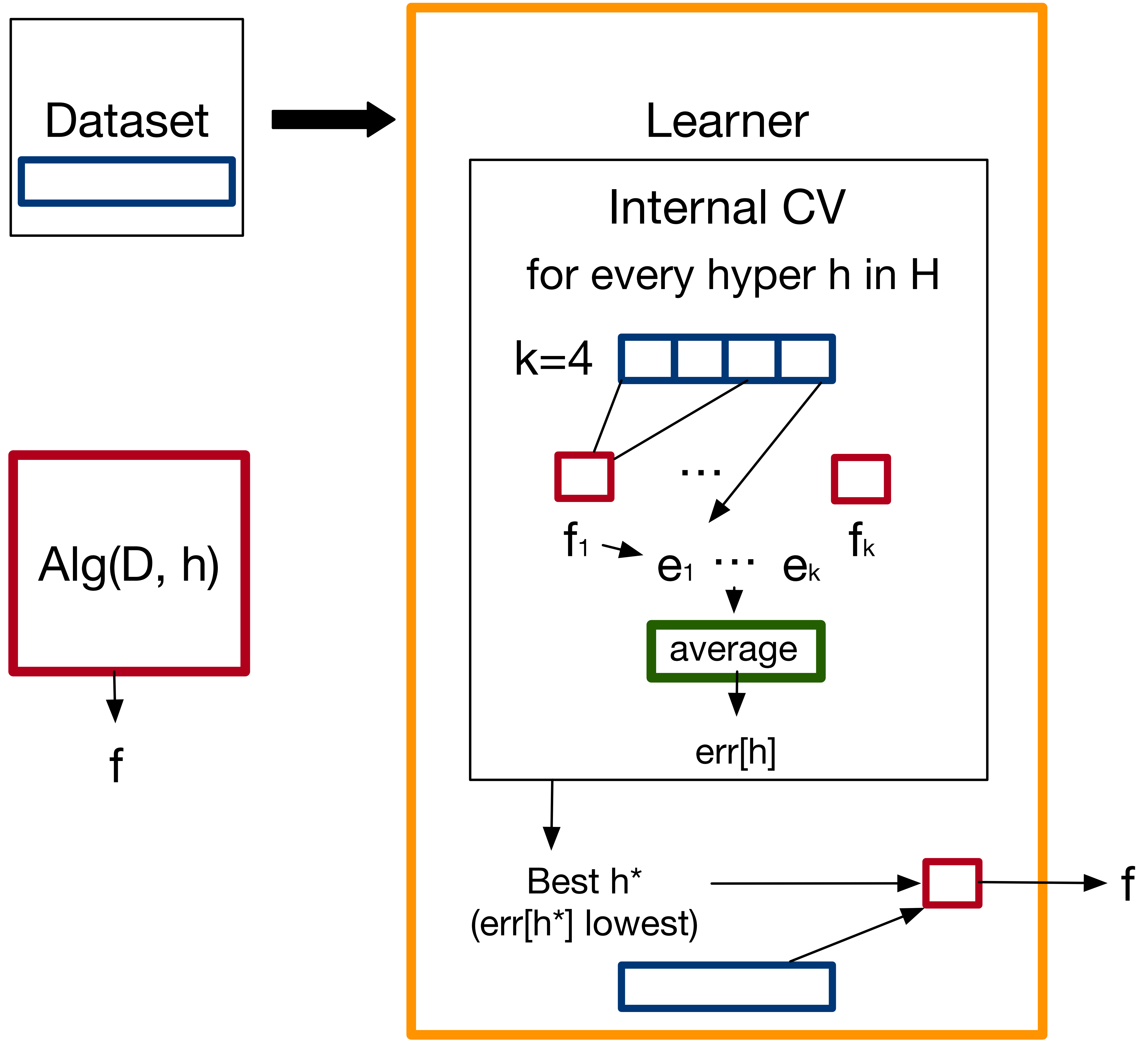
Couple of exercises

- Can we pick k = 2 for k-fold? Any issues?
- What if we pick k =2 and p = 0.01 for RSS?

CV for hyperparameter selection

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We can use it as an algorithm to pick hypeparameters
- Let us define a fully-specified algorithm, Learner(D), that uses CV to pick hyperparameters for Alg(D, h)
  - Essentially, Learner is also an algorithm, but one that does not have hyperparameters

CV for hyperparameter selection



Evaluating the Learner

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produced by Learner
- Can use training / validation set to evaluate it
    - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
    - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
    - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$

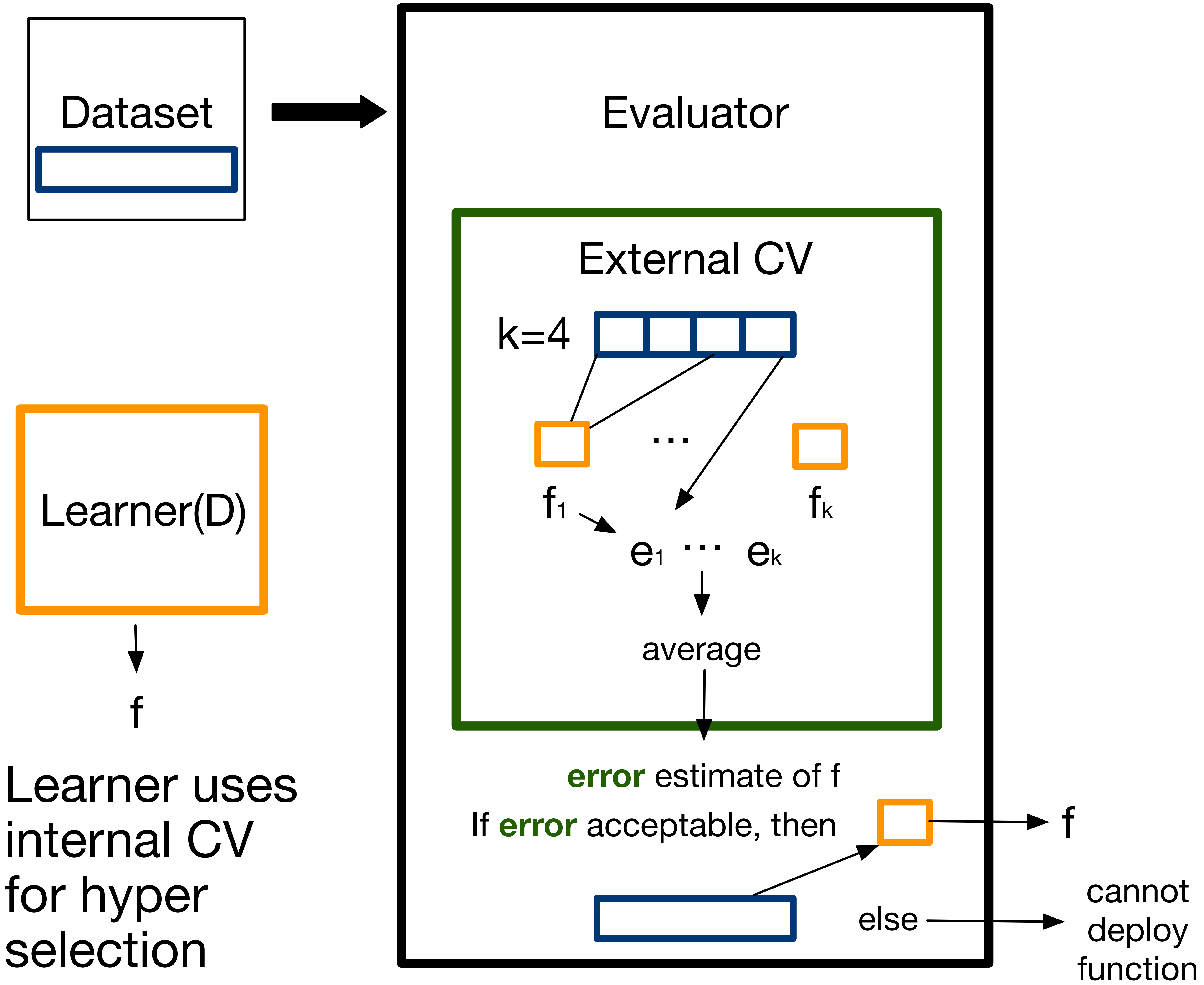
Evaluating the Learner

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produced by Learner
- Can use training / validation set to evaluate it
  - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
  - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
  - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$
- What is the issue with this approach?

Evaluating the Learner

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produced by Learner
- Can use training / validation set to evaluate it
  - Step 0: Split data into training $\mathscr{D}_{\text{tr}}$ and validation set $\mathscr{D}_{\text{test}}$
  - Step 1: Call Learner on dataset $\mathscr{D}_{\text{tr}}$, to get function f
  - Step 2: Evaluate f on $\mathscr{D}_{\text{test}}$
- What is the issue with this approach? Data inefficient, let's use CV!

Nested Cross-Validation



Step 1: Learn f on the entire dataset

Step 2: Do (external) CV to estimate the GE for f

Step 2 consists of
1. Get k partitions of the dataset, to get k training and test splits

2. For every i = 1 to k,
train f_i = Learner($\mathscr{D}_{tr}^{(i)}$) and
compute error e_i on $\mathscr{D}_{test}^{(i)}$

3. Get average error $\frac{1}{k}\sum_i e_i$

Exercise

- The simplest choice is to split the dataset into training, validation and test.
  - Hypers chosen using validation set (corresponds to the internal CV step)
  - The final model is trained on training+validation and evaluated on test (corresponds to external CV)
- RRS: Randomly sample pn points for test dataset (without replacement), and set the rest to the training set; do this k times
- Q: Is there a way to set p and k in RSS for internal CV to get back the simpler strategy of having one split to get a training and validation set?
- Q: Is there a way to set p and k in RSS for external CV to get back the simpler strategy of having a training and test split?

Do we do cross-validation in practice?

- Depends on the 1) cost of learning, 2) dataset size and 3) access to compute
- For medium to smaller datasets (< 1 million), cross-validation is used
  - There are still many problems that fit in this category
- For very large datasets, it is more ok to just use a train and test set
  - Or a single train and validation split for hyperparameter selection

Do we do cross-validation in practice?

- Depends on the 1) cost of learning, 2) dataset size and 3) access to compute
- For compute expense, can also
  - split off a test set for (external) evaluation and use (internal) CV for hyper selection
  - do external CV and internally use a single validation set for hyper selection
- Tempting but not really a good idea to do a two stage approach: CV on the entire data to pick hypers, followed by CV on the entire dataset with those best hypers