Chapter 7: Evaluating Generalization Performance

CMPUT 467: Machine Learning II

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

- cost$(\hat{y}, y) = (\hat{y} - y)^2$
  - Exercise: write GE$(f) = \mathbb{E}[\text{cost}(f(X), Y)]$ explicitly using $(x, y) \sim p$ or $x \sim p_x$ and $y \sim p(y \mid x)$ where $p(x, y) = p(y \mid x)p_x(x)$

Exercise: GE for squared error

- Write $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)] = \mathbb{E}[(f(X) - Y)^2]$ explicitly using $(x, y) \sim p$ or $x \sim p_x$ and $y \sim p(y|x)$ where $p(x, y) = p(y|x)p_x(x)$
- $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)] = \int p(x, y)\text{cost}(f(x), y)dxdy$
- $= \int p(x, y)(f(x) - y)^2 dxdy = \int p_x(x) \int p(y|x)(f(x) - y)^2 dydx$

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)
    - Exercise: write $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)]$ explicitly using $(x, y) \sim p$ or $x \sim p_x$ and $y \sim p(y\,|\,x)$ where $p(x, y) = p(y\,|\,x)p_x(x)$

Exercise: GE for 0-1 cost

- Write $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)]$ explicitly using $(x, y) \sim p$ or $x \sim p_x$ and $y \sim p(y|x)$ where $p(x, y) = p(y|x)p_x(x)$
- $\text{GE}(f) = \mathbb{E}[\text{cost}(f(X), Y)] = \int \sum_{y \in \{1,2,\ldots,m\}} p(x, y)\text{cost}(f(x), y)dx$
- $= \int p_x(x) \sum_{y \in \{1,2,\ldots,m\}} p(y|x)\text{cost}(f(x), y)dx = \int p_x(x) \sum_{y \in \{1,2,\ldots,m\}, y \neq f(x)} p(y|x)dx$
- $= \int p_x(x)(1 - p(f(x)|x))dx$

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 another asymmetric cost example for 3-class classification?

Some costs for generative model

- What cost could we use for a mixture model? How do you know you did a good job of fitting the data?
- Notice that the function f we evaluate is the distribution p_θ where $\theta = (w_1, w_2, \ldots, w_m, \beta_1, \beta_2, \ldots, \beta_m)$ are the parameter for the mixture
  - $$p_\theta = \sum_{k=1}^{m} w_k p(y \mid \beta_k)$$
- What is β_k if the mixture components are Gaussian? Or Poisson?

Some costs for generative model

- What cost could we use for a mixture model? How do you know you did a good job of fitting the data?
- Notice that the function f we evaluate is the distribution p_θ
- Log-likelihood of the data is a common choice
- GE$(p_\theta) = \mathbb{E}[-\ln p_\theta(y)]$

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?

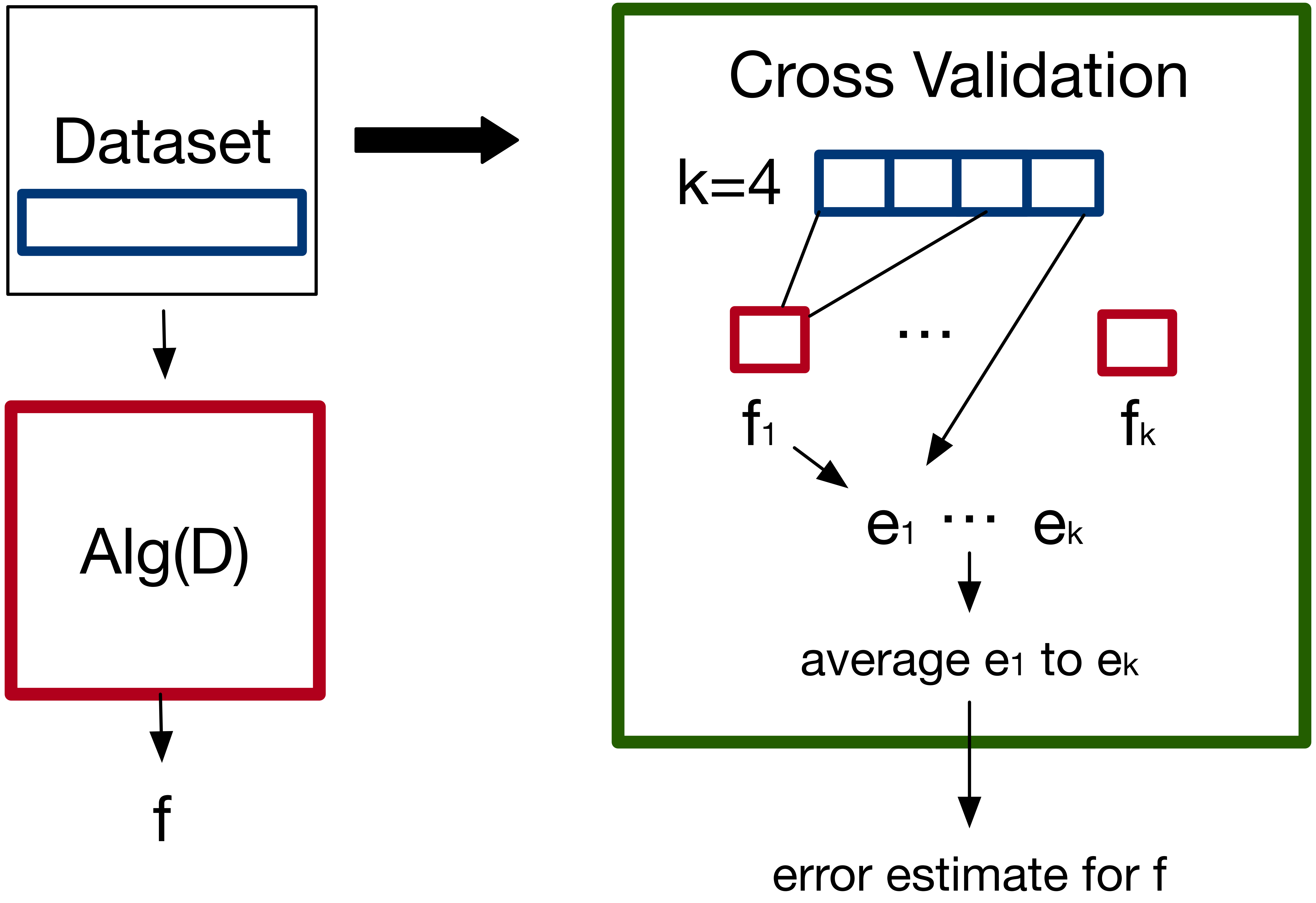
Estimating GE via Cross Validation

- 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, $\mathscr{D}_{\text{tr}}^{(i)}, \mathscr{D}_{\text{test}}^{(i)}$
- Train a function f_i on training set $\mathscr{D}_{\text{tr}}^{(i)}$ and evaluate on test $\mathscr{D}_{\text{test}}^{(i)}$ 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 $\mathscr{D}$, $\hat{\text{GE}}(f) = \frac{1}{k}\sum_i e_i$

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?

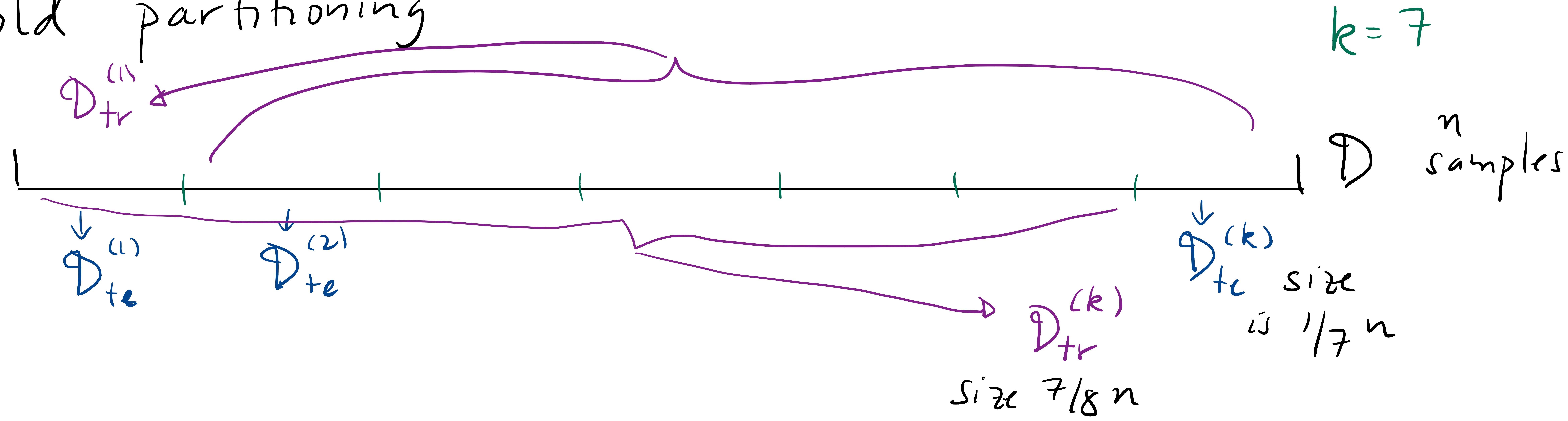
How do we get the k partitions?

- 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 vs RSS

- 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
- Repeated random subsampling (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

k-fold partitioning



RRS with percentage p for test

$k = 7$

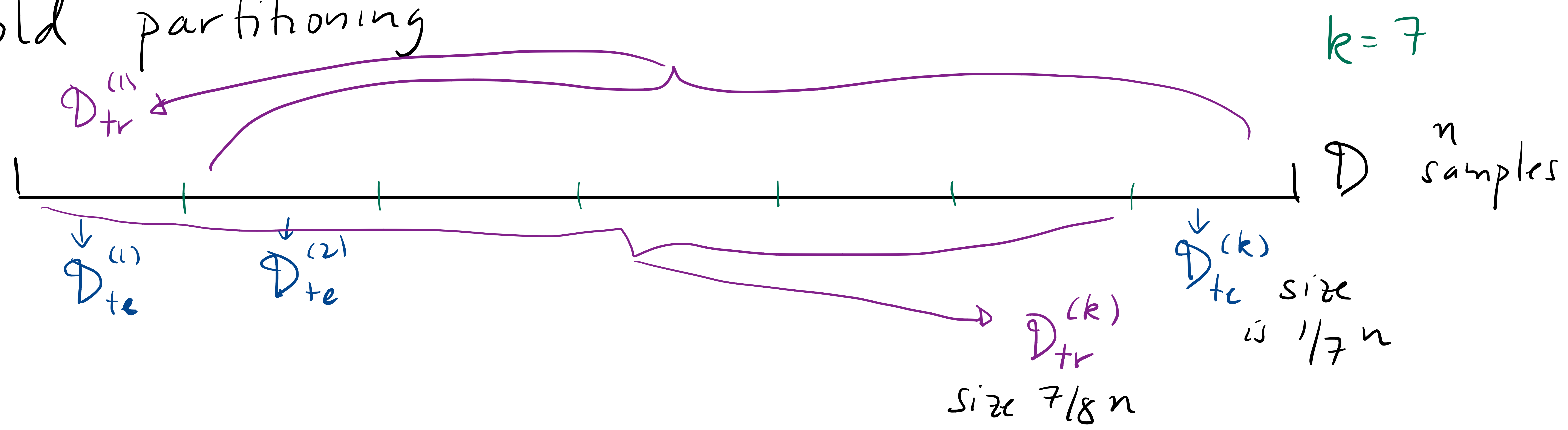
$\mathcal{D}$

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

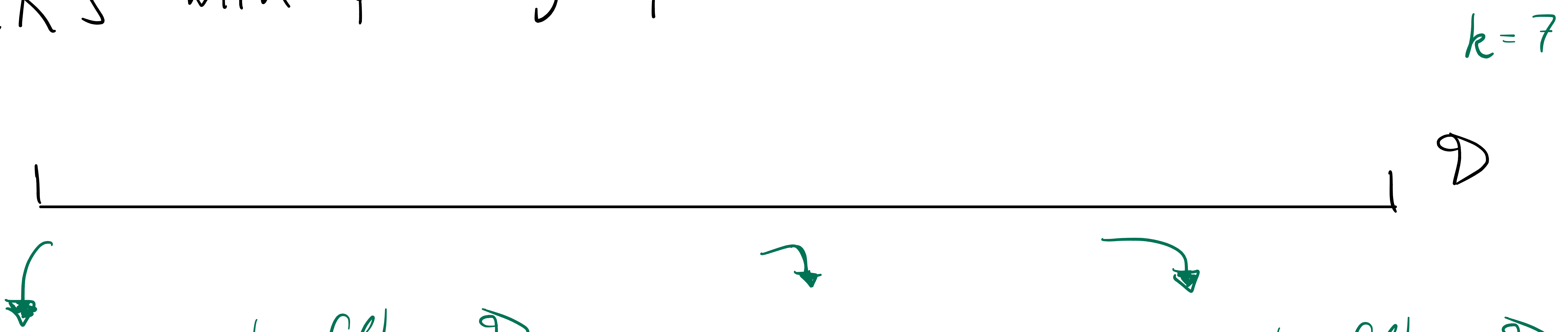
. . .

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

k-fold partitioning



RRS with percentage p for test



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

How do we pick k? (for bias)

- How is bias impacted by the choice of k in 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 in 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

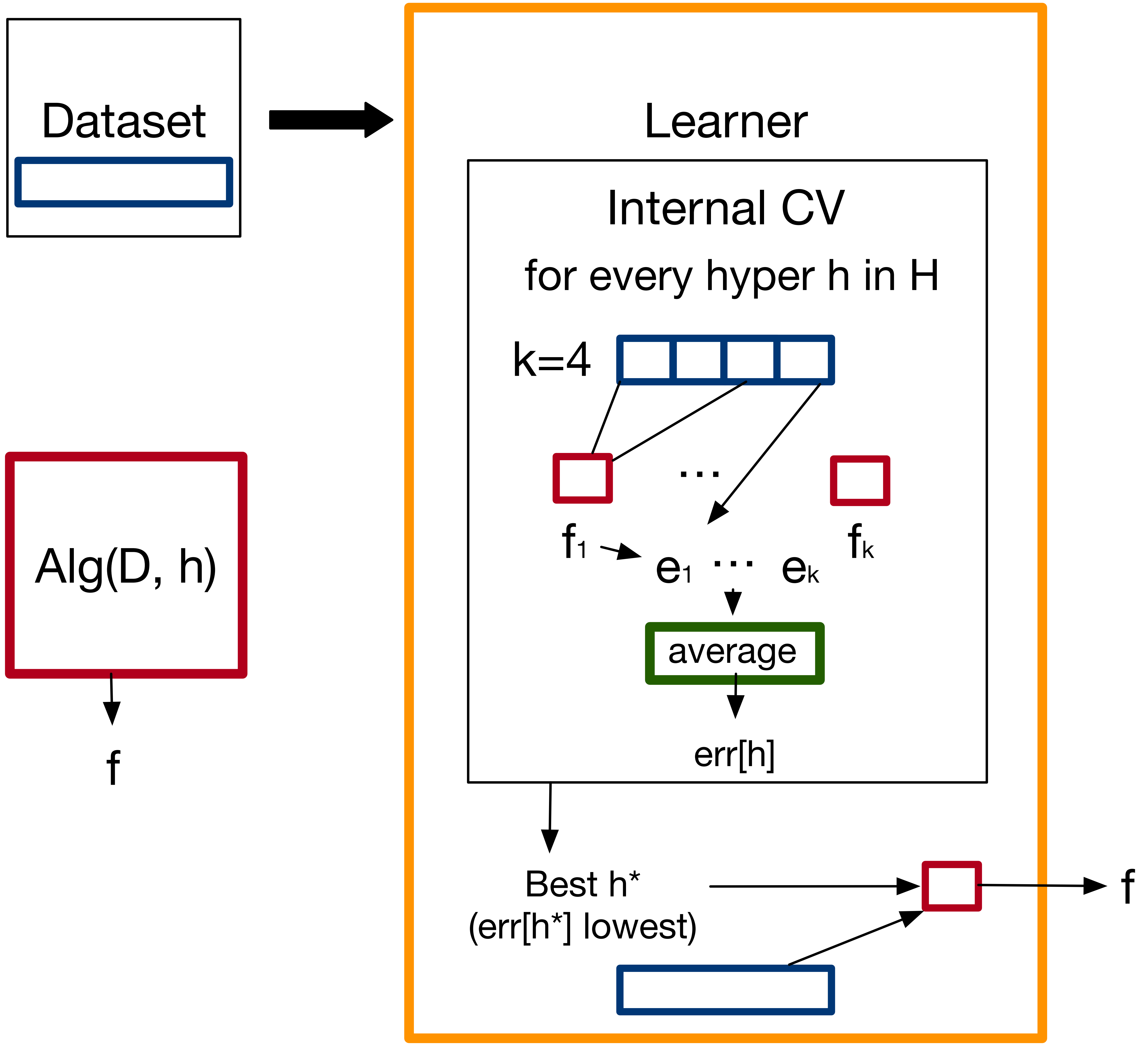
Couple of exercises

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

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 produce 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 $\mathscr{D}_{\text{test}}$

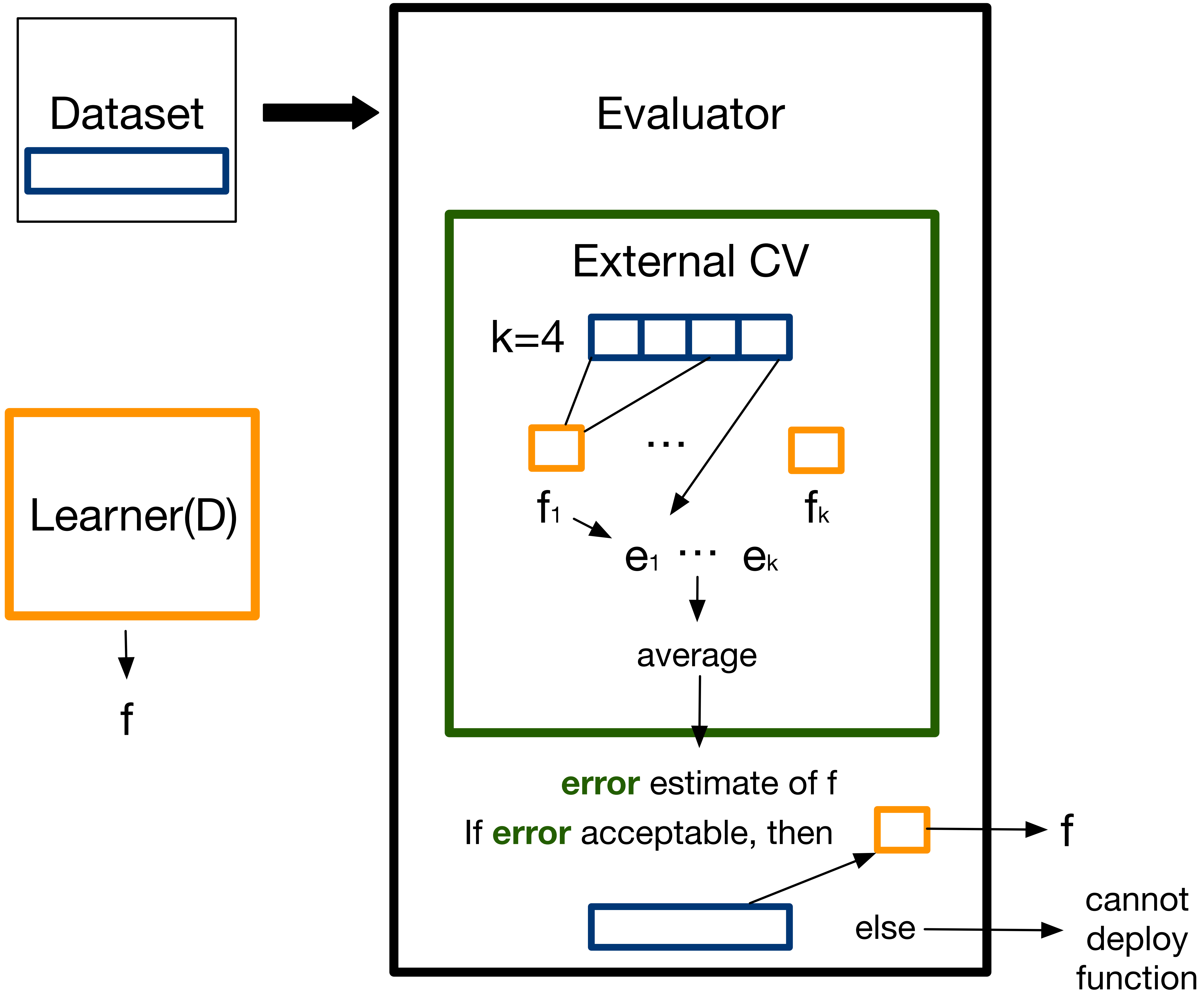
Evaluating the Learner

- Our estimate of (GE) is a good criteria to pick hyperparameters
- We still need to evaluate the model produce 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 produce 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 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 $\frac{1}{k}\sum_i e_i$