Logistics

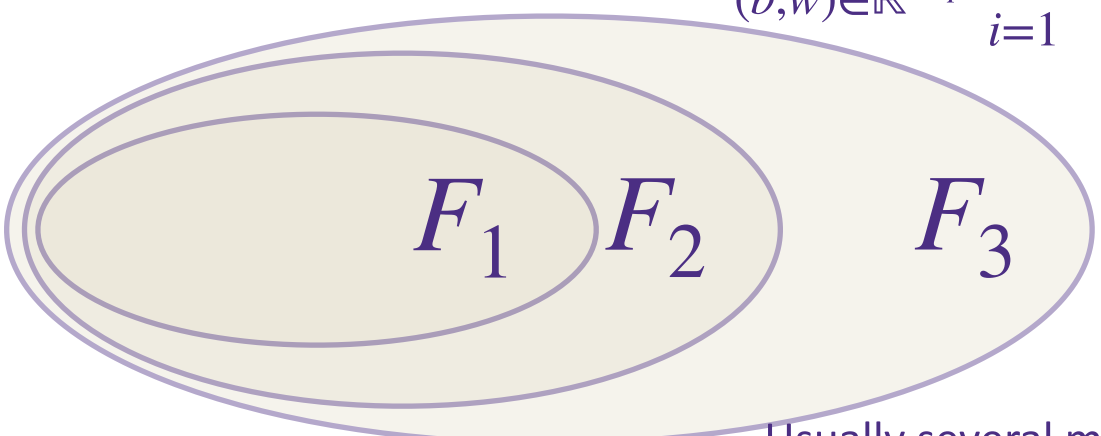
- Question from a student: can we get access to the recording from the past offerings of CSE44
  - For FERPA reasons, we are not able to release them.
- We put other public video lectures in the course website that covers a free online textbook
- Office hours are there to help you not just with homework for more generally.

Lecture 8: Model selection using Cross-validation

Parameter and hyper-parameter

- A **model class** is set of functions, each function is indexed by its **parameters** representing the function
  - e.g., a model class $F_p = \{\text{all degree-}p \text{ polynomials in } \mathbb{R}\}$ each function in that class is represented (or indexed) by parameter $(b, w) \in \mathbb{R}^{p+1}$, which can be also written more explicitly as
- $F_p = \{f_{b,w} : \mathbb{R} \to \mathbb{R} \,|\, f_{b,w}(x) = b + w_1 x + \cdots + w_p x^p, \text{ for some} (b, w) \in \mathbb{R}^{1+p}\}$
  - **Parameter** is what is optimized when training a model

    e.g., $(\widehat{b}_p, \widehat{w}_p) = \arg \min_{(b,w) \in \mathbb{R}^{1+p}} \sum_{i=1}^{n} \big(y_i - (h(x_i)^T w + b) \big)^2$



$\cdots$

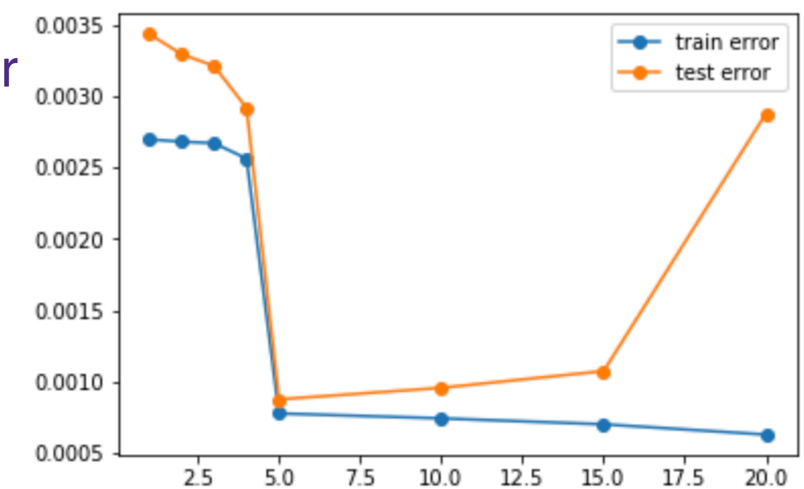
Usually several model classes form
a nested hierarchy of classes with increasing complexities

Parameter and hyper-parameter

- **A family of model classes** is a set of model classes, each class indexed by its **hyper-parameter** representing a model class
  - e.g., a family $\mathscr{F} = \{F_p \mid p \in \{1,2,\ldots\}\}$ is a set of model classes with hyper-parameter $p \in \{1,2,\ldots\}$, where F_p is a class of degree-p polynomials
  - Hyper-parameter is usually fixed during training

    e.g., $\hat{f}_p = \arg\min_{f \in F_p} \sum_{i=1}^{n} \left(y_i - f(x_i) \right)^2$
  - And we run multiple training for multiple choices of the hyper-parameter

Error



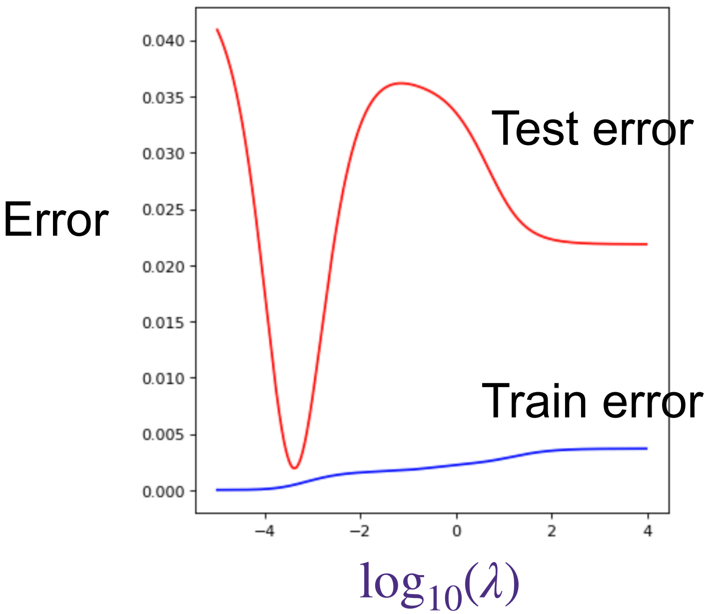
Hyper-parameter p

Hyper-parameter for ridge regression

- Hyper parameter does not have to represent a model class
- It can represent the algorithm being used also
- **hyper-parameter** λ for ridge regression
  - e.g., a linear model class $F_1 = \{f(x) = b + x^T w \,|\, (b, w) \in \mathbb{R}^{d+1}\}$ trained by minimizing a regularized loss

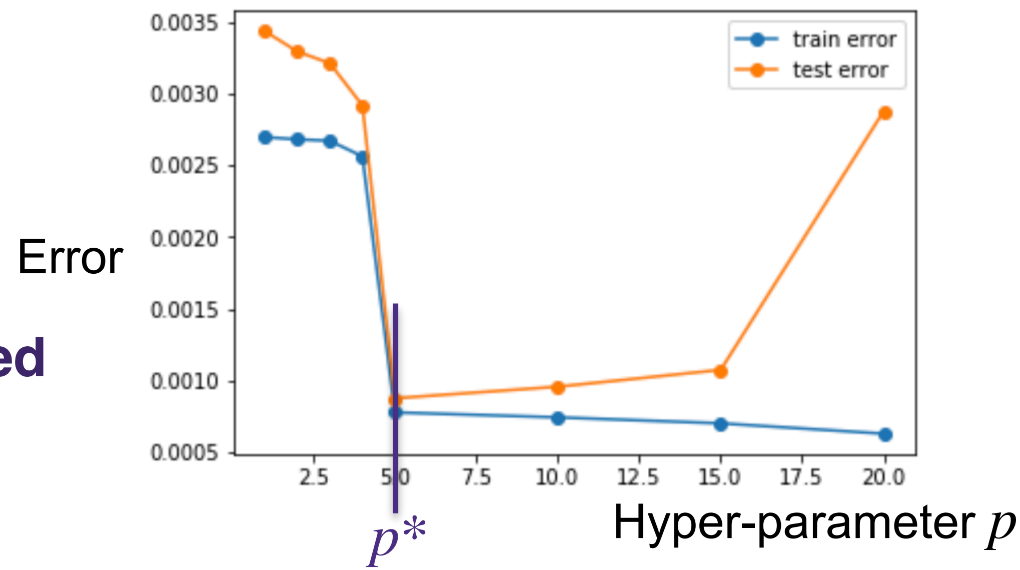
$$(\widehat{b}_\lambda, \widehat{w}_\lambda) = \arg \min_{(b,w)\in\mathbb{R}^{d+1}} \sum_{i=1}^{n} (y_i - (b + x_i^T w))^2 \; + \; \lambda \|w\|_2^2$$

  - λ is a hyper-parameter, because it is fixed during training
  - And we run multiple training for multiple choices of the hyper-parameter



Model selection

- **Model selection** asks the following question: among all the models we got for different hyper-parameters, how do we choose the "best" one to deploy?
- Wrong approach 1:
  - Randomly split the dataset into Train Set and Test Set with 80/20 split
  - Train models for various hyper-parameters and report the Train Error and Test Error
  - Deploy the model $\widehat{w}_{p*}$ achieving minimum Test Error
  - Report its Test Error as an approximation of the True Error
- Issue:
  - **Test Error is underestimated**
  - **Relying on one data split has large variance**



Why using test error for model selection gives under estimation of the true error

- Consider a simple experiment where we have two coins
  - $x_i \sim \text{Bern}(p)$ and $y_i \sim \text{Bern}(q)$ such that

$$x_i = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases} \qquad y_i = \begin{cases} 1 & \text{with probability } q \\ 0 & \text{with probability } 1-q \end{cases}$$

- I want to find out $\min\{p, q\}$ given n samples from each
- Using test set to both select the model and report the test error is same as
  - Computing the empirical averages $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $\hat{q} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and reporting the smaller one, i.e., $\min\{\hat{p}, \hat{q}\}$
  - We can show that this reported value is strictly smaller than what we wanted in expectation: $\mathbb{E}[\,\min\{\hat{p}, \hat{q}\}\,] < \min\{p, q\}$
  - For example, if $n = 1$, then $\mathbb{E}[\min\{\hat{p}, \hat{q}\}] = \mathbb{E}[\min\{x_1, y_1\}] =$

To avoid underestimating test error

- Never use the **test set** for
  - training any model, or
  - tuning hyper-parameter = model selection
- **Test set** should only be used once to report test error (as an approximation of the true error) in the end
- Idea:
  - use part of training data (called **Validation Set**) to estimated the error and for model selection
  - For example:

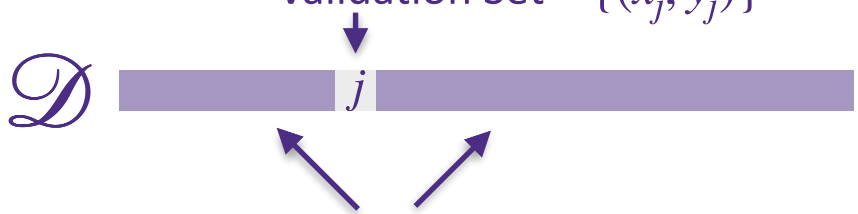
    - **Train set**: train (multiple) models for different hyper-parameters
    - **Validation set**: compute validation error, to be used in model selection
    - **Test set**: use it once in the end to report test error for the selected model

(LOO) Leave-one-out cross validation

Notation:
$A \backslash B = A \cap B^C$ denotes setminus

- Consider a validation set with 1 example:
  - $\mathscr{D}$: dataset
  - $\mathscr{D} \backslash \{j\}$: train set with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathscr{D} \backslash \{j\}}$ with $\mathscr{D} \backslash \{j\}$ dataset: $f_{\mathscr{D} \backslash \{j\}} = \arg\min_f \sum_{i \in \mathscr{D} \backslash \{j\}} (y_i - f(x_i))^2$

Validation Set = $\{(x_j, y_j)\}$



Train Set = $\mathscr{D} \backslash \{j\}$

- **Validation error**: $\text{error}_j \triangleq (y_j - f_{\mathscr{D} \backslash \{j\}}(x_j))^2$
  - It is an unbiased estimate of the error
$$\text{error}_{\text{true}}(f_{\mathscr{D} \backslash \{j\}}) \triangleq \mathbb{E}_{(x,y) \sim P_{x,y}}[(y - f_{\mathscr{D} \backslash \{j\}}(x))^2]$$
  - but, variance of error_j is too large. Why? **Validation set is small**
  - Any ideas?

(LOO) Leave-one-out cross validation

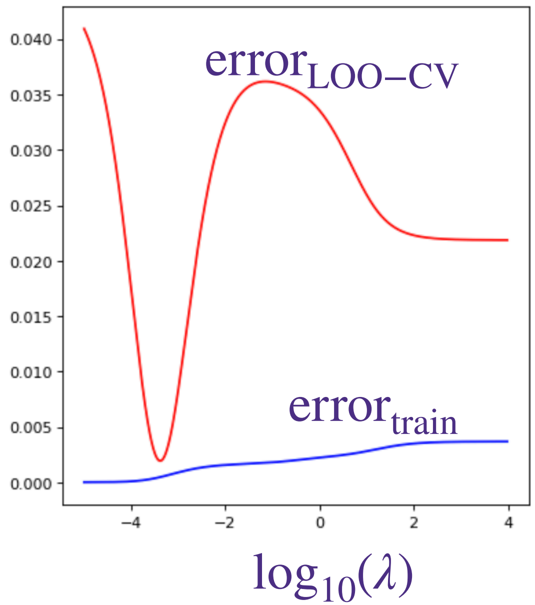
- To reduce the variance of the validation error, use instead
- **LOO cross validation**: Average over all possible single sample validation set $\{j\}$ for $j \in \{1,\ldots,n\}$:
  - Train n times:
    for each data point you leave out, learn a new classifier $f_{\mathcal{D}\setminus\{j\}}$
  - **Validation error** is now averaged over all different splits:

$$\text{error}_{\text{LOO-CV}} = \frac{1}{n}\sum_{j=1}^{n} \text{error}_j = \frac{1}{n}\sum_{j=1}^{n} (y_j - f_{\mathcal{D}\setminus\{j\}}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate!

- When computing LOO-CV error, we only use $n-1$data points to train
  - So it's not an estimate of true error of learning with n data points
    true error = $\mathbb{E}_{X,Y}[\,(Y - f_{\mathcal{D}}(X))^2\,]$
  - Usually (slightly) pessimistic – learning with less data typically gives worse answer.
  - Leads to a (slight) over estimation of the error compared to true error

- LOO-CV is almost unbiased! Use LOO-CV error for model selection!!!
  - **E.g., picking λ**



Computational cost of LOO

- Suppose you have 100,000 data points
- say, you implemented a fast version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day
- In general, LOO takes n times longer than training one model
- Any ideas?

Use *k*-fold cross validation

- Randomly divide data into *k* equal parts
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{\mathscr{D}\backslash\mathscr{D}_i}$ using data point not in $\mathscr{D}_i$
  - Estimate error of $f_{\mathscr{D}\backslash\mathscr{D}_i}$ on validation set $\mathscr{D}_i$:

$$\mathscr{D} = \quad \mathscr{D}_1 \quad \mathscr{D}_2 \quad \mathscr{D}_3 \quad \mathscr{D}_4 \quad \mathscr{D}_5$$

$f_{\mathscr{D}\backslash\mathscr{D}_3}$	Train	Train	Validation	Train	Train

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\backslash\mathcal{D}_i}(x_j))^2$$

- k-fold cross validation error is average over data splits:

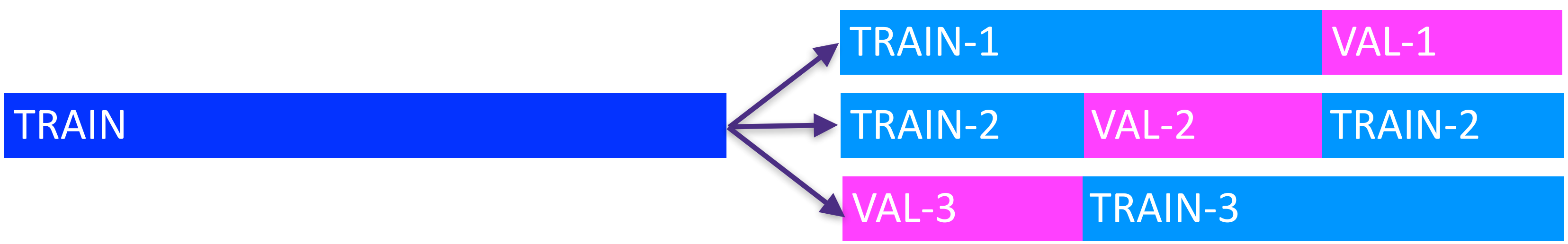
$$\text{error}_{k-\text{fold}} = \frac{1}{k}\sum_{i=1}^{k} \text{error}_{\mathscr{D}_i}$$

- k-fold cross validation properties:
  - Much faster to compute than LOO-CV as $k \ll n$
  - More (pessimistically) biased – using much less data, only $n - \frac{n}{k}$
  - Usually, k = 10

Recap

> Given a dataset, begin by splitting into

> Model selection: Use k-fold cross-validation on TRAIN to train predictor and choose hyper-parameters such as λ



> Model assessment: Use TEST to assess the accuracy of the model you output

- Never train or choose parameters based on the test data

Example 1

- You wish to predict the stock price of zoom.us given historical stock price data y_i's (for each i-th day) and the historical news articles x_i's
- You use all daily stock price up to Jan 1, 2020 as TRAIN and Jan 2, 2020 - April 13, 2020 as TEST
- What's wrong with this procedure?

- **train and test data are from different distributions and has sampling bias (of time)**

Example 2

- Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the training set:

50 indices j that have largest $\frac{|\sum_{i=1}^{n} x_{i,j} y_i|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}$

- After picking our 50 features, we then use CV with the training set to train ridge regression with regularization λ
- What's wrong with this procedure?

- **We are underestimating the error as the features have been chosen with the score that depends on the validation set**

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly select TEST set and split the remaining dataset into TRAIN, and VAL (multiple splits are needed if doing cross validation)
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear features (also called transformations)
  - Choose a loss function
    - E.g., least squares with ridge regression penalty on TRAIN
  - **Choose an optimization procedure**
    - **E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features and amount of regularization**
  - Justifying the accuracy of the estimate
    - E.g., report TEST error

Questions?

Questions?

Questions?