What you need to know…

$\infty \longrightarrow$ # basis functions

$+\lambda\|w\|_2^2$

- Regularization
  - Penalizes complex models towards preferred, simpler models
- Ridge regression

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{n} (y_i - w^T x_i - b)^2 + \lambda \cdot \|w\|_2^2$$

  - L_2 penalized least-squares regression
  - Regularization parameter trades off model complexity with training error
  - Never regularize the offset!

Example: piecewise linear fit

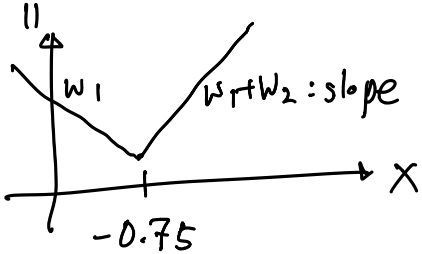
- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

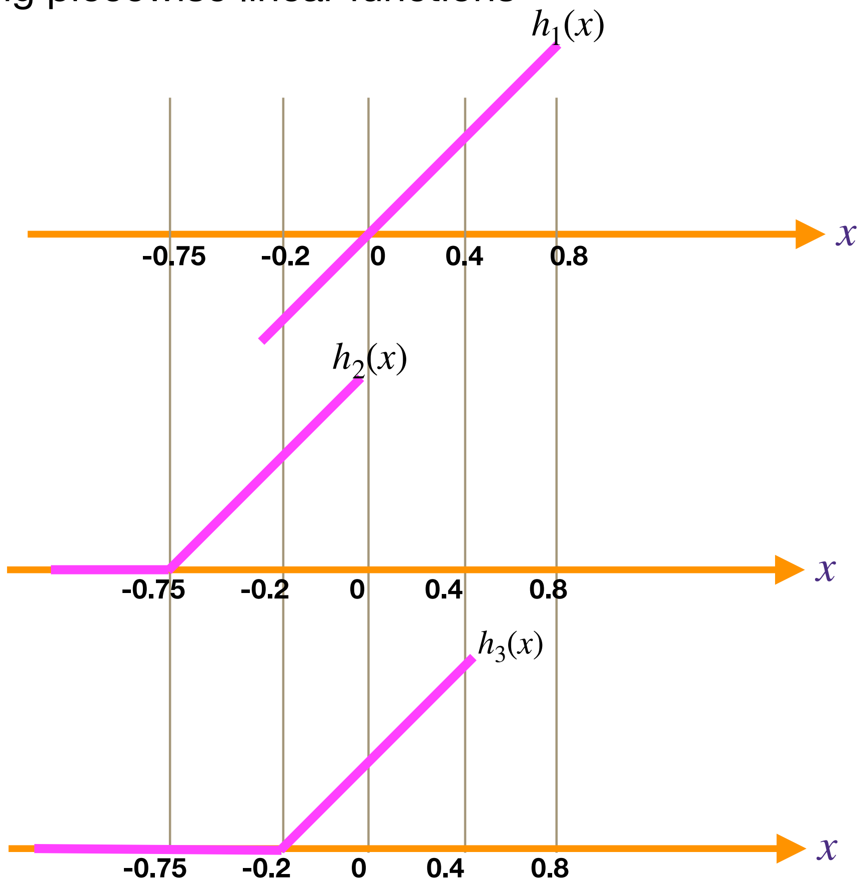
$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

$$[a]^+ \triangleq \max\{a, 0\}$$

$$w^T h(x) = w_1 x + w_2 [x+0.75]^+$$

$\parallel$





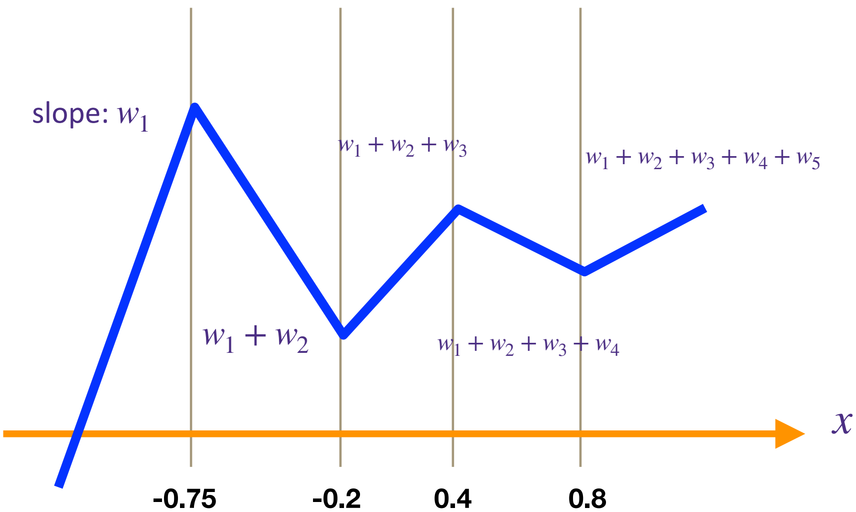
Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

$$[a]^+ \triangleq \max\{a, 0\}$$

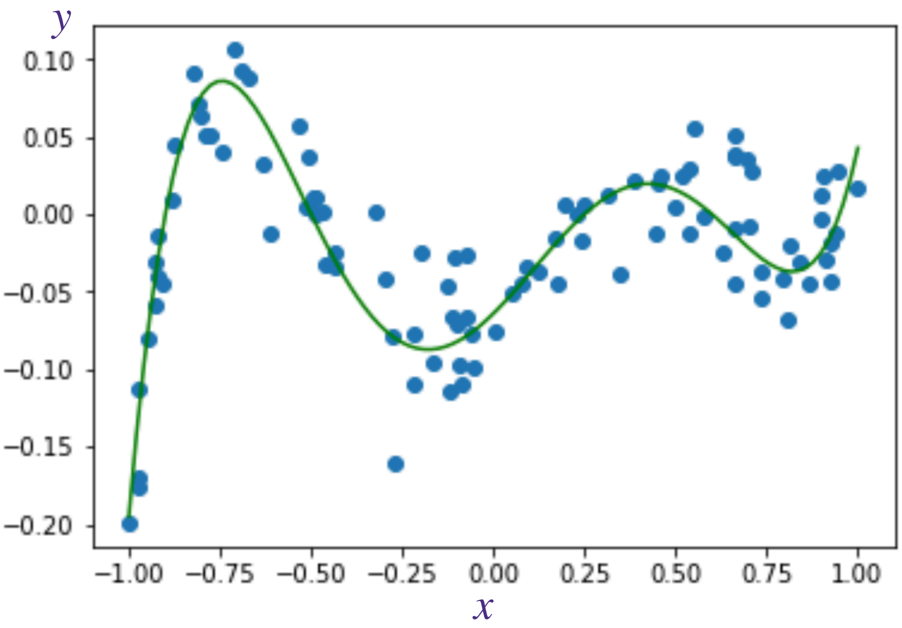
$$y = b + h(x)^T w$$



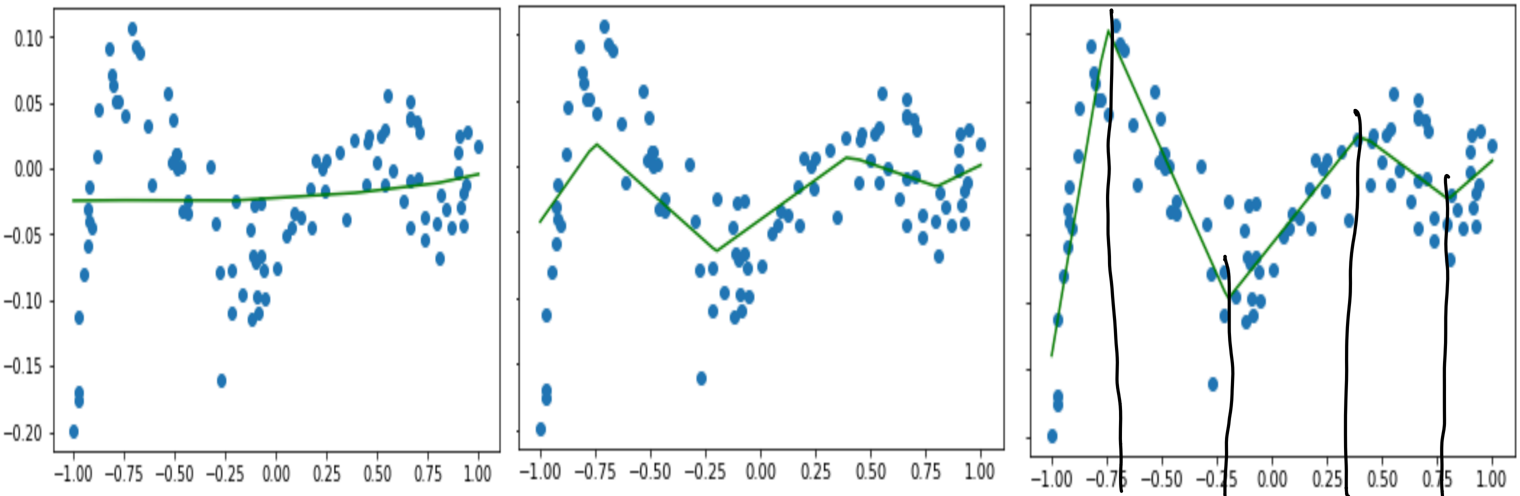
the weights capture the change in the slopes

Example: piecewise linear fit

- we fit a linear model:
  $$f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$$
- with a specific choice of features using piecewise linear functions



Example: piecewise linear fit (ridge regression)



$\lambda = 1$ $\quad\quad$ $\lambda = 0.005$ $\quad\quad$ $\lambda = 0.000001$

We do not observe overfitting, as d=5 and n=100

Piecewise linear with $w \in \mathbb{R}^{10}$ and n=11 samples

$h_1(x) = x$

$h_2(x) = [x + 0.8]^+$

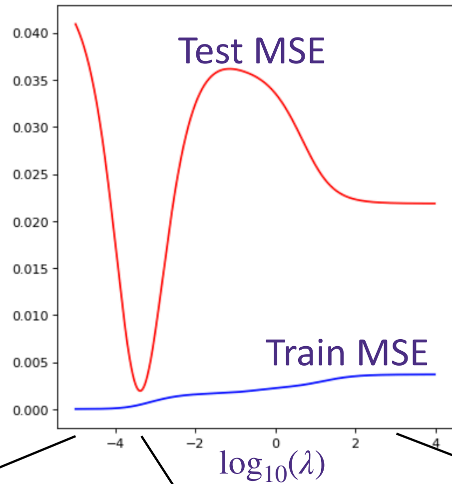
$h_3(x) = [x + 0.6]^+$

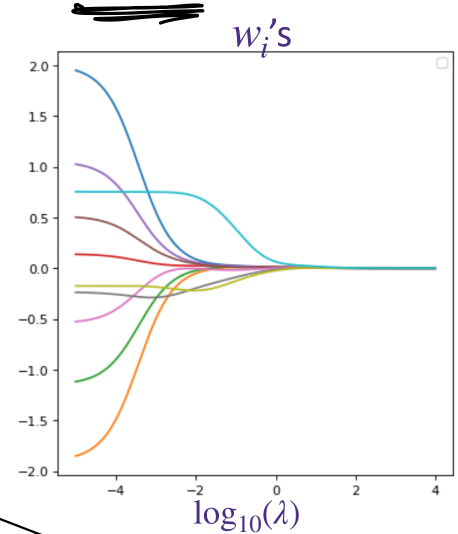
0.4

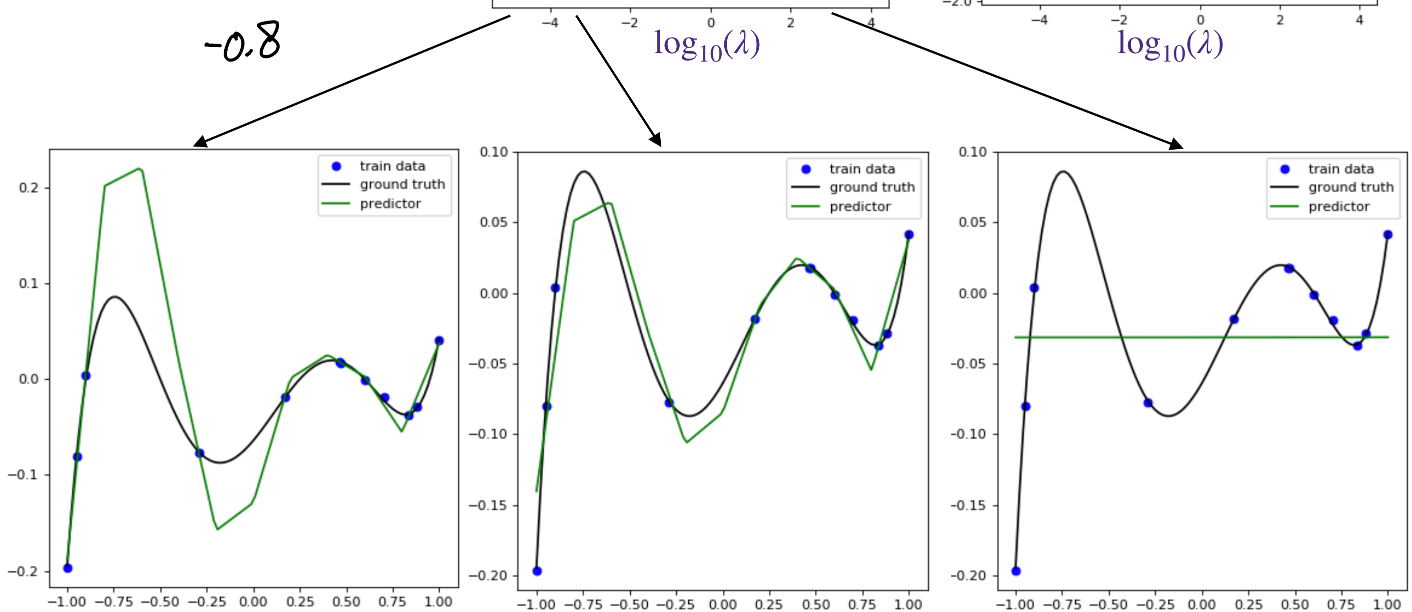
0.2

−0.2

−0.8







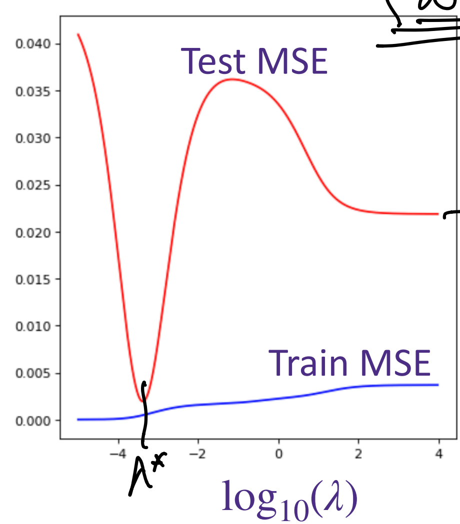
Model selection using Cross-validation

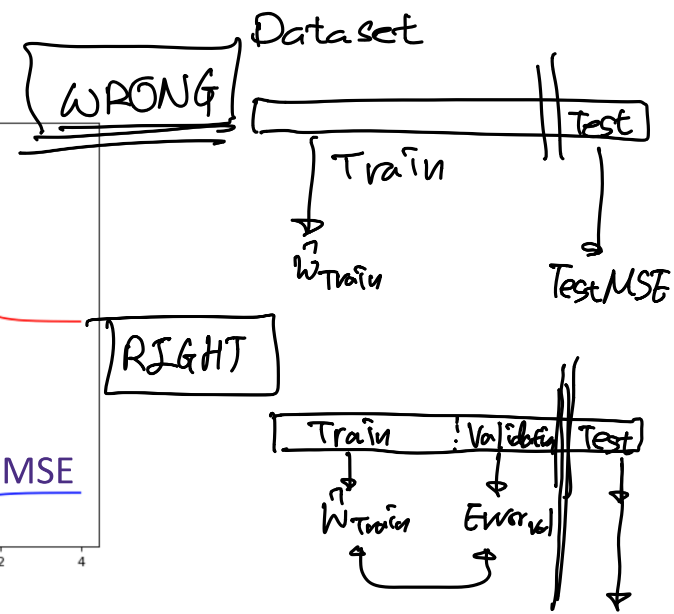
How… How… How???????

> Ridge regression:
How do we pick the regularization constant λ…

> Polynomial features:
How do we pick the number of basis functions…

> We could use the test data, but…





How… How… How???????

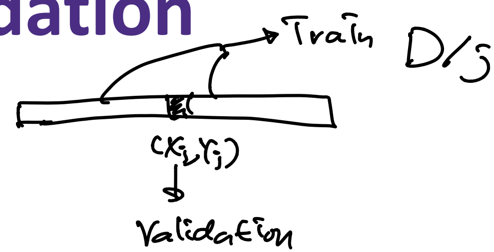
- Ridge regression:
  How do we pick the regularization constant λ…
- Polynomial features:
  How do we pick the number of basis functions…

- We could use the test data, but…
  - Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever **train on the test data**

    train $\hat{w}$
    choosing $\hat{\lambda}$

  - Use test data only for reporting the test error (once in the end)

(LOO) Leave-one-out cross validation



- Consider a validation set with 1 example:
  - $\mathcal{D}$: training data
  - $\boxed{\mathcal{D}\setminus j}$: training data with j-th data point (x_j, y_j) moved to validation set
- Learn model $f_{\mathcal{D}\setminus j}$ with $\mathcal{D}\setminus j$ dataset
- The squared error on predicting y_j:

$$\overbrace{(y_j - \underbrace{f_{\mathcal{D}\setminus j}(x_j)}_{\text{Trained on } D\setminus j})^2}^{\text{error on } (x_j, y_j)} = \text{Error}_{\text{val-}j}$$

is an unbiased estimate of the **true error**

$$\text{error}_{\text{true}}(f_{\mathcal{D}\setminus j}) \triangleq \mathbb{E}_{(x,y)\sim P_{x,y}}[(y - f_{\mathcal{D}\setminus j}(x))^2]$$

but, variance of $(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$ is too large

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - $\mathscr{D}$: training data
  - $\mathscr{D}\backslash j$: training data 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
- The squared error on predicting y_j: $(y_j - f_{\mathscr{D}\backslash j}(x_j))^2 \triangleq \text{Error}_j$

  is an unbiased estimate of the **true error**

$$\text{error}_{\text{true}}(f_{\mathscr{D}\backslash j}) = \mathbb{E}_{(x,y)\sim P_{x,y}}[(y - f_{\mathscr{D}\backslash j}(x))^2] = \mathbb{E}[\text{Error}_j]$$

  but variance of $(y_j - f_{\mathscr{D}\backslash j}(x_j))^2$ is too large, so instead

  $\text{Var}[\text{Error}_j] \uparrow$ Huge

- **LOO cross validation**: Average over all data points j:
  - Train n times:
    for each data point you leave out, learn a new classifier $f_{\mathscr{D}\backslash j}$
  - **Estimate the true error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{D\backslash j}(x_j))^2 \rightarrow \text{Var}\downarrow$$

LOO cross validation is (almost) unbiased estimate!

$$\text{Error}_{LOO} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \boxed{f_{D\setminus i}(X_i)}\right)^2$$

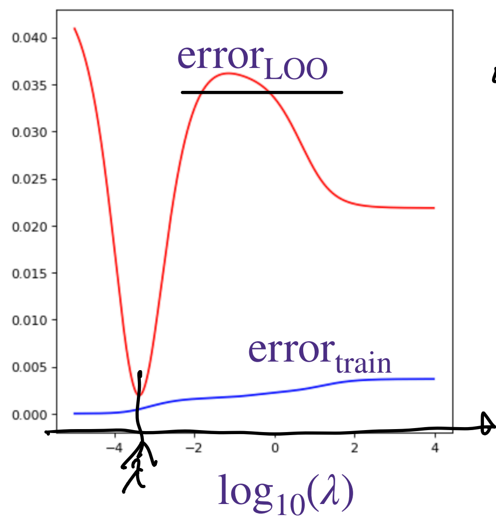
- When computing LOOCV error, we only use $n-1$ data points to train
  - So it's not estimate of true error of learning with n data points
  - Usually pessimistic – learning with less data typically gives worse answer. (Leads to an over estimation of the error)

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

$$\text{Error}_{True,n} = \mathbb{E}_{(X,Y)}\left[\left(f_D(X) - Y\right)^2\right]$$

$$\wedge$$

$$\text{Error}_{LOOCV} \approx \mathbb{E}_{(X,Y)}\left[\left(f_{D\setminus j}(X) - Y\right)^2\right]$$



→ $\boxed{\text{Retrain}}$

model selection

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!!

Use *k*-fold cross validation

- Randomly divide training data into *k* equal parts
  - $D_1, \ldots, D_k$

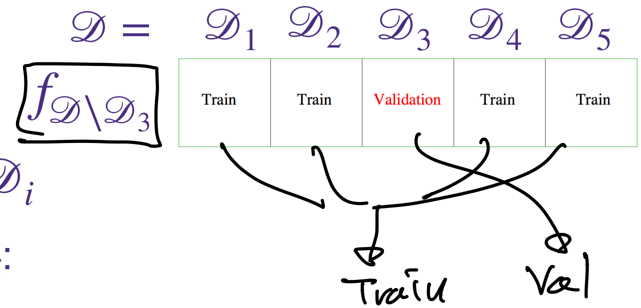
$\mathscr{D} = \quad \mathscr{D}_1 \quad \mathscr{D}_2 \quad \mathscr{D}_3 \quad \mathscr{D}_4 \quad \mathscr{D}_5$

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

- 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 $\mathscr{D}_i$:

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

Use *k*-fold cross validation



> Randomly divide training 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 $\mathscr{D}_i$:

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

> 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:

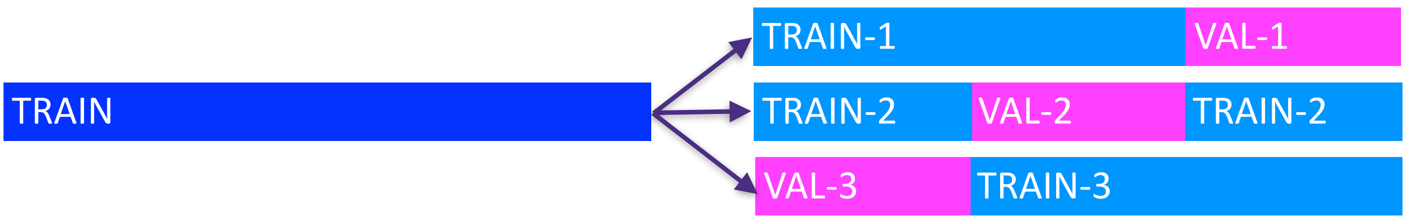
loo : $n-1$

  - Much faster to compute than LOO as $k \ll n$
  - More (pessimistically) biased – using much less data, only $n - \frac{n}{k}$: k-fold
  - 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 ever ever ever ever train or choose parameters based on the test data

Model selection using cross validation

- For $\lambda \in \{0.001, 0.01, 0.1, 1, 10\}$
  - For $j \in \{1, \ldots, k\}$
    - $$\hat{w}_{\lambda, \text{Train}-j} \leftarrow \arg\min_{w} \sum_{i \in \text{Train}-j} (y_i - w^T x_i)^2 + \lambda \|w\|_2^2$$
- $$\hat{\lambda} \leftarrow \arg\min_{\lambda} \frac{1}{k} \sum_{j=1}^{k} \sum_{i \in \text{Val}-j} (y_i - \hat{w}_{\lambda, \text{Train}-j}^T x_i)^2$$

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?

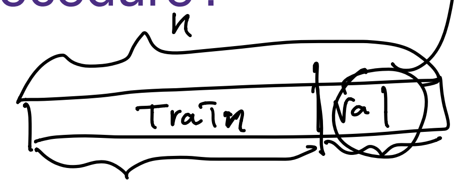
Example 2

> 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:

$$\text{50 indices j that have largest} \quad \left.\frac{\left|\sum_{i=1}^{n} x_{i,j} y_i\right|}{\sqrt{\sum_{i=1}^{n} x_{i,j}^2}}\right\} \text{feature selection.}$$

> 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?



Recap

Train × 9 × # hyper parameters

Single Train Time × # hyperParm × K

> Learning is…

- Collect some data
  - > E.g., housing info and sale price
- Randomly split dataset into TRAIN, VAL, and TEST
  - > E.g., 80%, 10%, and 10%, respectively

    9-fold cross
- Choose a hypothesis class or model
  - > E.g., linear with non-linear 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?

$$\mathbb{E}_{train}\left[\hat{\varepsilon}(f_{train})\right]$$

$$\mathbb{E}_{test}\left[\hat{\varepsilon}(f_{train})\right]$$

$$\parallel$$

$$\mathbb{E}_{\substack{\{(x_i, y_i)\}_{i \in Test} \\ \sim P_{XY}}}\left[\boxed{\frac{1}{N_{test}} \sum_{i \in Test} \left(y_i - f_{train}(x_i)\right)^2}\right]$$