Lecture 3: Linear regression and polynomial features

- How to fit more complex data

- HW0 due Wednesday October 8th midnight

Linear Regression

Recap: Maximizing log-likelihood

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

Set gradient=0, solve for w

$$\widehat{w}_{MLE} = \left(\sum_{i=1}^{n} x_i x_i^\top \right)^{-1} \sum_{i=1}^{n} x_i y_i$$

The regression problem in matrix notation

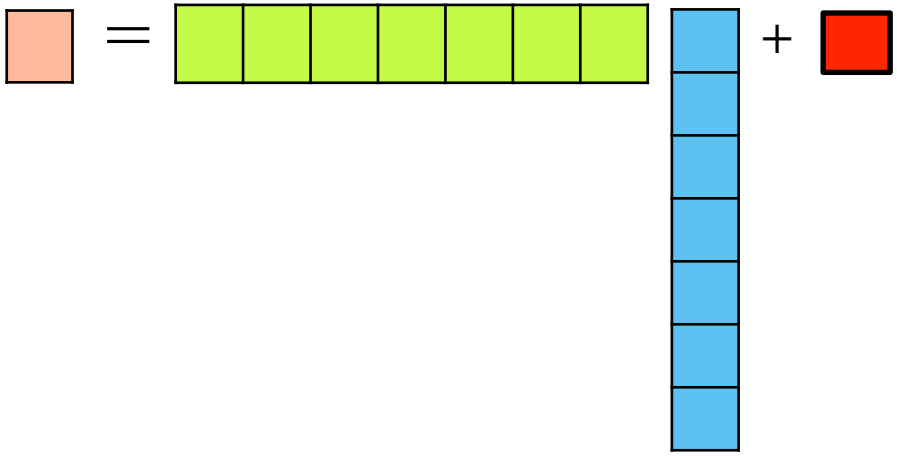
Data:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

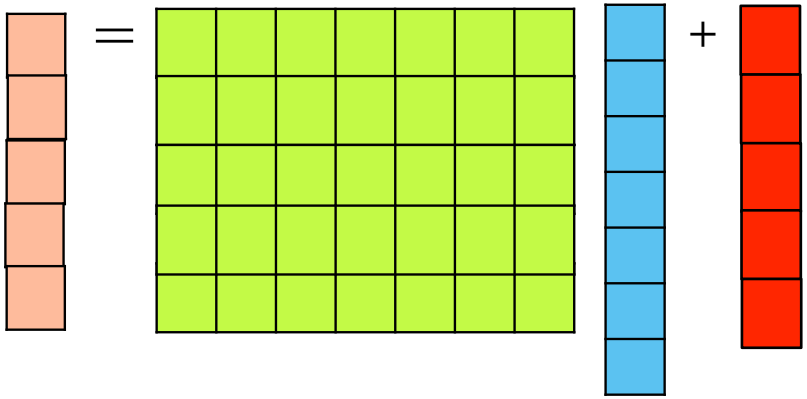
d : # of features/size of the input

n : # of examples/datapoints

$$y_1 = x_1^T w + \epsilon_1$$



$$y = Xw + \epsilon$$



The regression problem in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features
n : # of examples/datapoints

$$\hat{w}_{\text{MLE}} = \arg\min_{w} \|y - Xw\|_2^2$$
$$= \arg\min_{w} (y - Xw)^T (y - Xw)$$

ℓ_2 norm: $\|z\|_2 = \sqrt{\sum_{i=1}^{n} z_i^2} = \sqrt{z^\top z}$

[related to HW0 questions A6 and A7]

The regression problem in matrix notation

$$= \arg\min_{w} (\mathbf{y} - \mathbf{X}w)^T (\mathbf{y} - \mathbf{X}w)$$

[Friday section this week covers this in more detail, and we have amazing TAs]

The regression problem in matrix notation

$$\widehat{w}_{MLE} = \arg\min_{w} \sum_{i=1}^{n} (y_i - x_i^\top w)^2$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \mathbf{X} = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

d : # of features
n : # of examples/datapoints

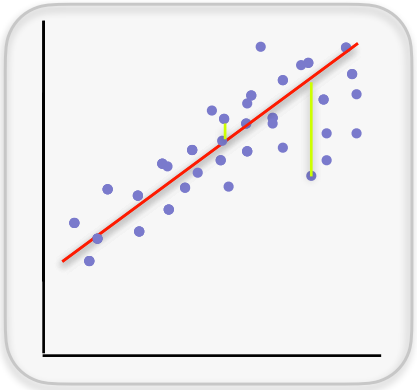
$$\hat{w}_{\text{MLE}} = \arg\min_{w} \|y - Xw\|_2^2$$
$$= \arg\min_{w} (y - Xw)^T (y - Xw)$$

$$\hat{w}_{LS} = \hat{w}_{MLE} = \left(\boldsymbol{X}^T \boldsymbol{X}\right)^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

The regression problem with offset

Recall that we start with a linear model with no offset

$$y_i = x_i^T w + \epsilon_i$$



$$\hat{w}_{\text{MLE}} = \arg\min_{w} \|y - Xw\|_2^2$$
$$= (X^T X)^{-1} X^T y$$

We can add the offset to the linear model, with a new parameter b

$$y_i = x_i^T w + b + \epsilon_i$$

$$\hat{w}_{\text{MLE}}, \hat{b}_{\text{MLE}} = \arg\min_{w,b} \sum_{i=1}^{n} (y_i - (x_i^T w + b))^2$$
$$= \arg\min_{w,b} \|y - (Xw + 1b)\|_2^2$$

Dealing with an offset

$$\hat{w}_{\text{MLE}}, \hat{b}_{\text{MLE}} = \arg\min_{w,b} \|y - (Xw + 1b)\|_2^2$$

$$X^T X \hat{w}_{\text{MLE}} + X^T \mathbf{1} \hat{b}_{\text{MLE}} = X^T y$$

$$\mathbf{1}^T X \hat{w}_{\text{MLE}} + \mathbf{1}^T \mathbf{1} \hat{b}_{\text{MLE}} = \mathbf{1}^T y$$

Dealing with an offset: standardization

$$\hat{w}_{\text{MLE}}, \hat{b}_{\text{MLE}} = \arg\min_{w,b} \|y - (Xw + 1b)\|_2^2$$

$$X^T X \hat{w}_{\text{MLE}} + X^T \mathbf{1} \hat{b}_{\text{MLE}} = X^T y$$

$$\mathbf{1}^T X \hat{w}_{\text{MLE}} + \mathbf{1}^T \mathbf{1} \hat{b}_{\text{MLE}} = \mathbf{1}^T y$$

If $X^T \mathbf{1} = 0$, i.e., if each feature is mean-zero or we pre-processed the data have zero-mean, then

$$\hat{w}_{\text{MLE}} = (X^T X)^{-1} X^T y$$

$$\hat{b}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

If you scale and shift each coordinate of x, it is known as **standardization**. Usually this pre-processing is done to ensure that after the preprocessing, $X^T \mathbf{1} = 0$ and $\frac{1}{n} \sum_{i=1}^{n} x_{i,j}^2 = 1$ for all coordinates $j \in [d]$, where $[d] = \{1, 2, \ldots, d\}$

Make Predictions with offset

$$\hat{w}_{\text{MLE}} = (X^T X)^{-1} X^T y$$

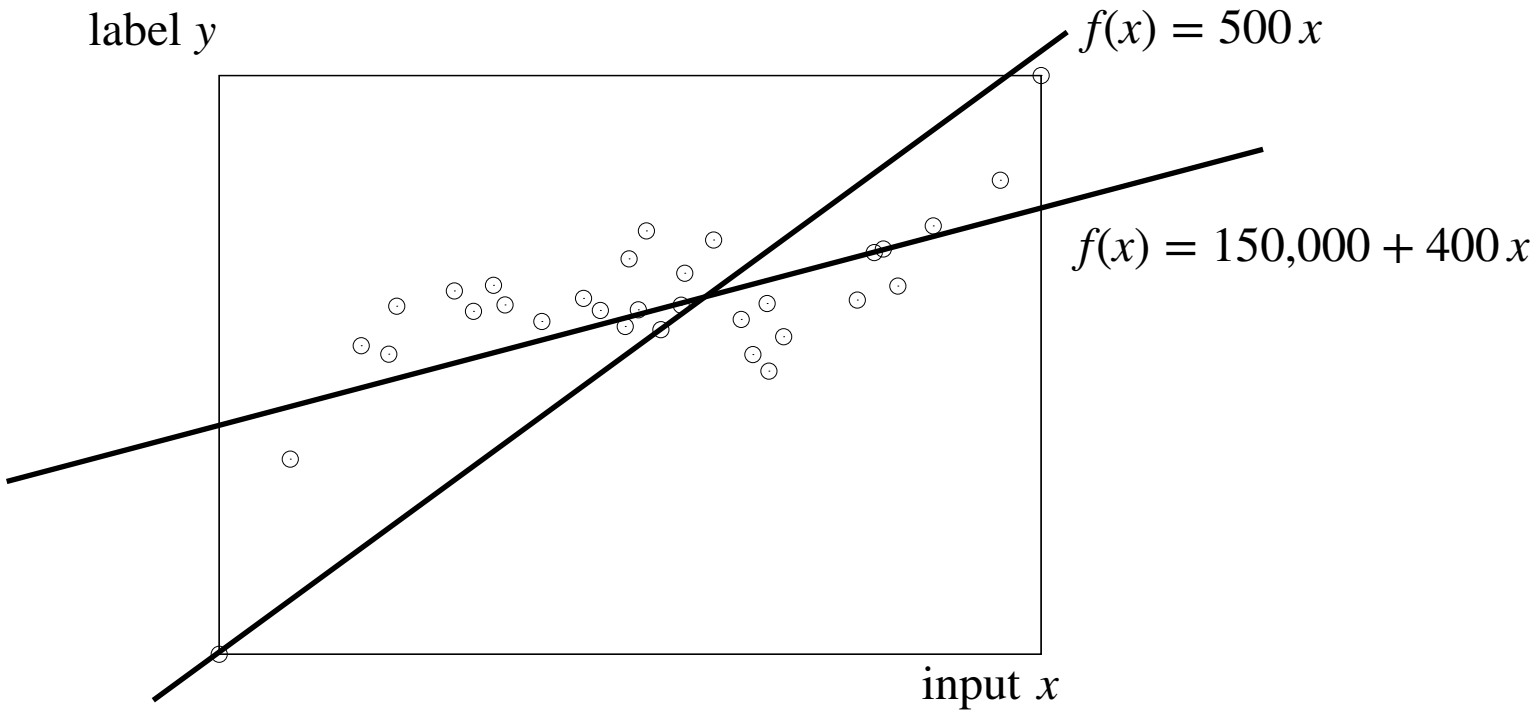
$$\hat{b}_{\text{MLE}} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

A new house is about to be listed. What should it sell for?

$$\hat{y}_{\text{new}} = x_{\text{new}}^T \hat{w}_{\text{MLE}} + \hat{b}_{\text{MLE}}$$

Polynomial features

Recap: Linear Regression

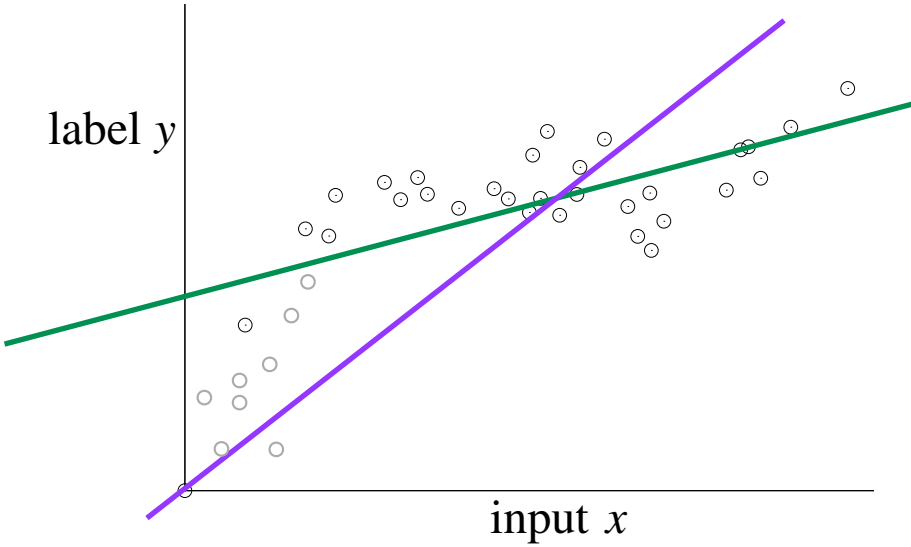


- We fit a linear model with **one parameter** w: $y_i = wx_i + \epsilon_i$
  and make prediction as $y = \hat{w}_{\text{MLE}}\, x$
- To get more expressive (or ***complex***) model, we can fit a linear model with offset, having **two parameters** (b, w): $y_i = wx_i + b + \epsilon_i$
  and make prediction as $y = \hat{w}_{\text{MLE}}\, x + \hat{b}_{\text{MLE}}$

Quadratic regression in 1-dimension

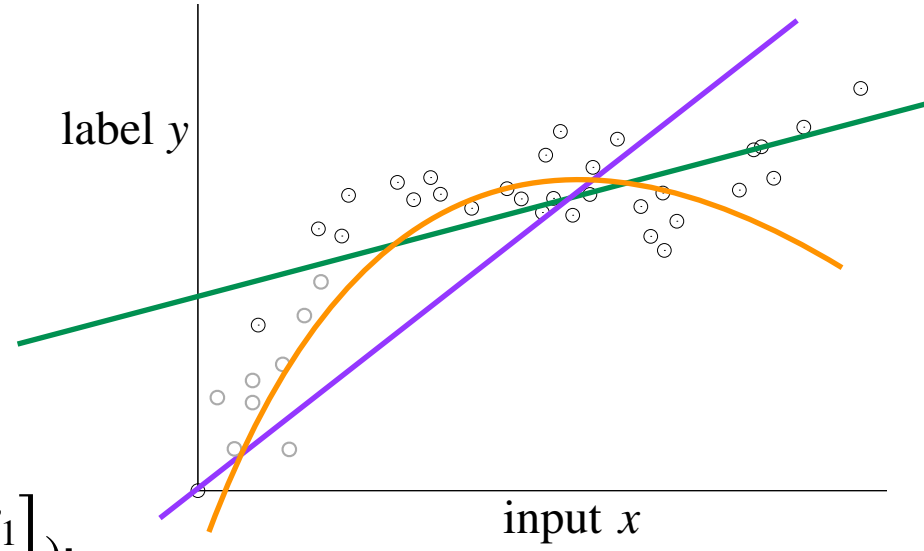
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$



Quadratic regression in 1-dimension

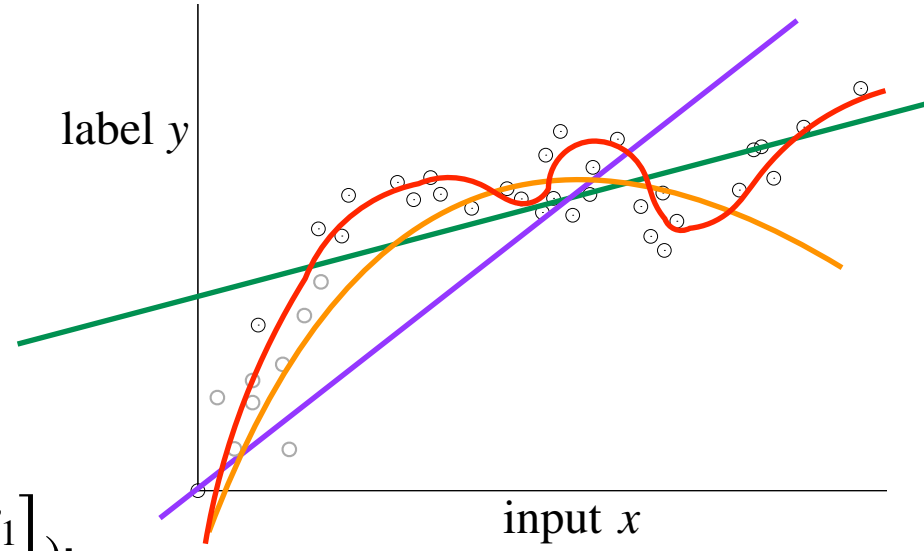
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$

Polynomial regression in 1-dimension

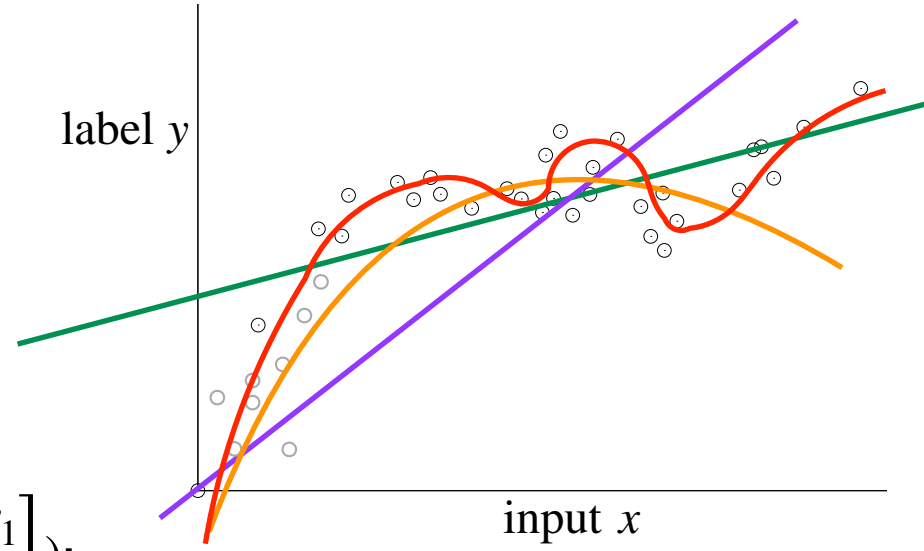
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$
- **Degree-p polynomial model with parameter** $(b, w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p$

General regression in 1-dimension

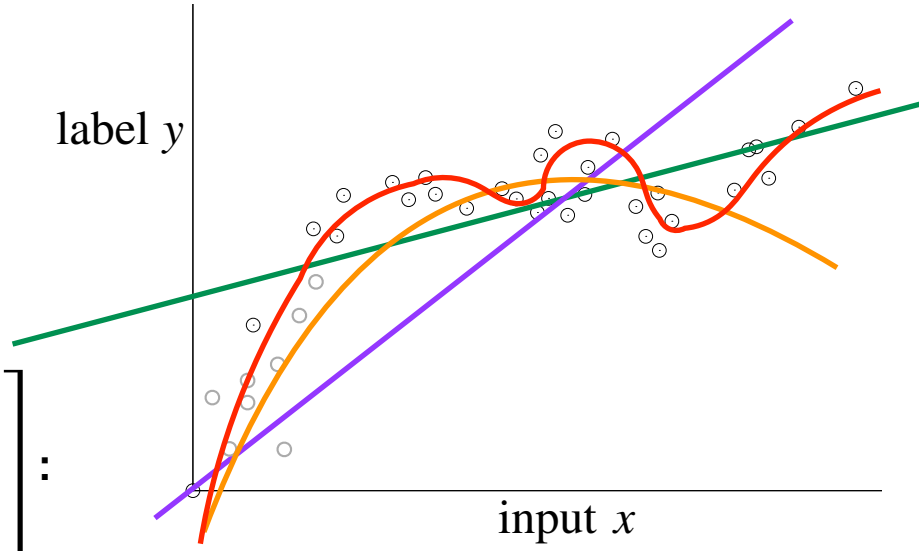
- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $\hat{y}_i = b + w_1 x_i$
- **Quadratic model with parameter** $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2$
- **Degree-p polynomial model with parameter** $(b, w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix})$**:**
  - $\hat{y}_i = b + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p$
- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$**:**
  - $\hat{y}_i = w^T h(x_i)$ where $h : \mathbb{R} \to \mathbb{R}^p$

General regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

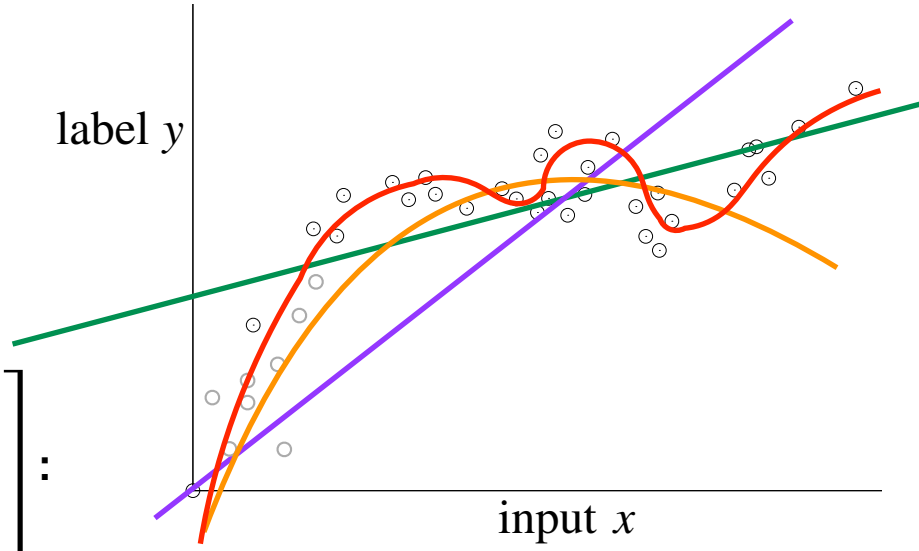
Note: h can be arbitrary non-linear functions!

$$h(x) = \begin{bmatrix} \log(x) \\ x^2 \\ \sin(x) \\ \sqrt{x} \end{bmatrix}$$

How do we learn w?

General regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **General p-features with parameter** $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

How do we learn w?

$$\mathbf{H} = \begin{bmatrix} - - h(x_1)^\top - - \\ \vdots \\ - - h(x_n)^\top - - \end{bmatrix} \in \mathbb{R}^{n \times p}$$

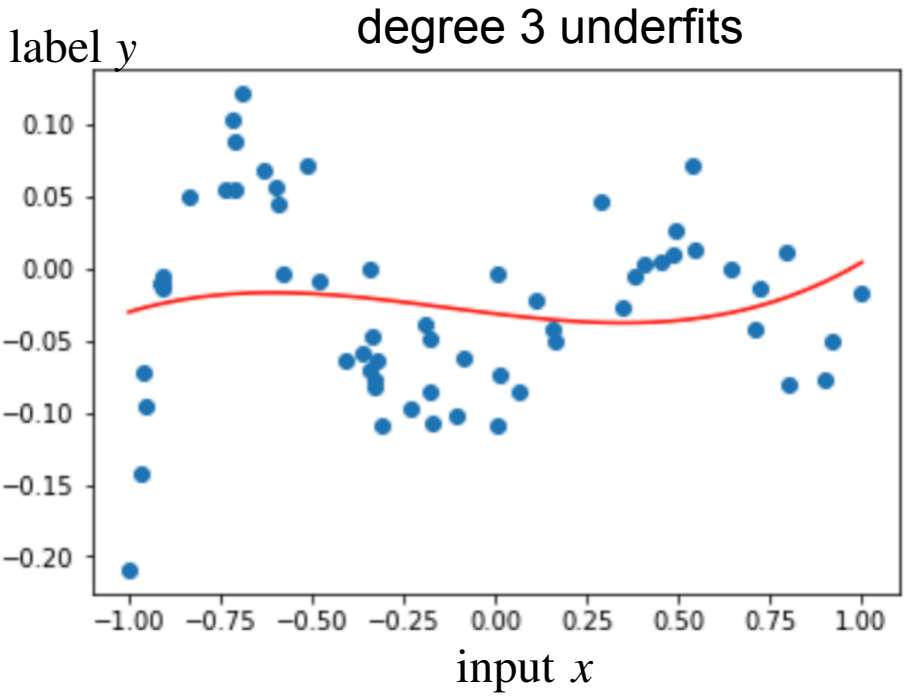
$$\widehat{w} = \arg\min_w \|\mathbf{H}w - \mathbf{y}\|_2^2$$

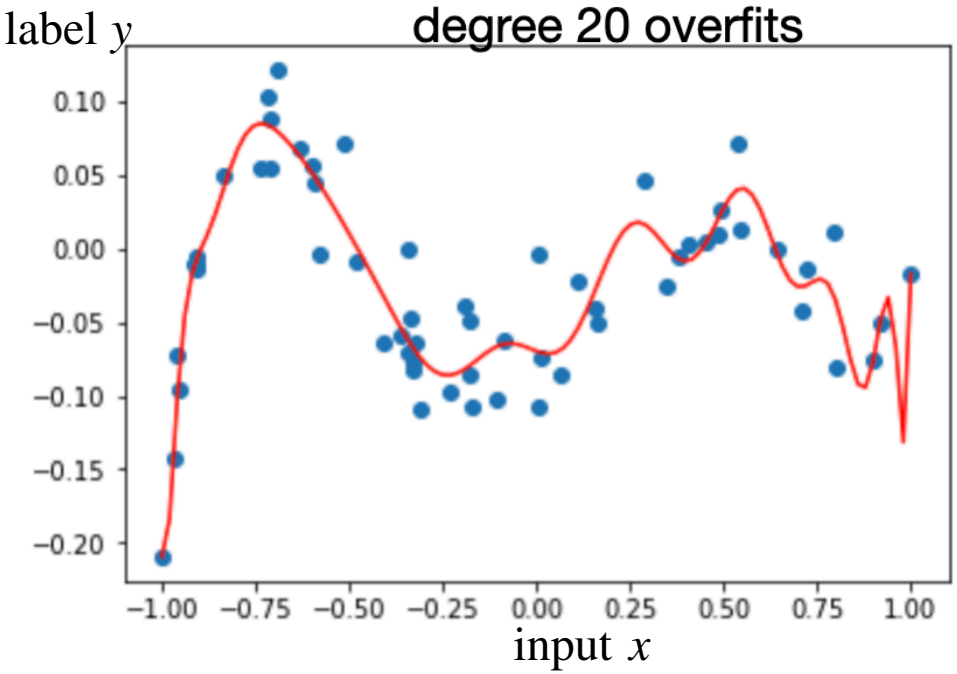
For a new test point x, predict

$\hat{y} = \widehat{w}^T h(x)$

Which p should we choose?

- First instance of class of models with different **representation power** = **model complexity**





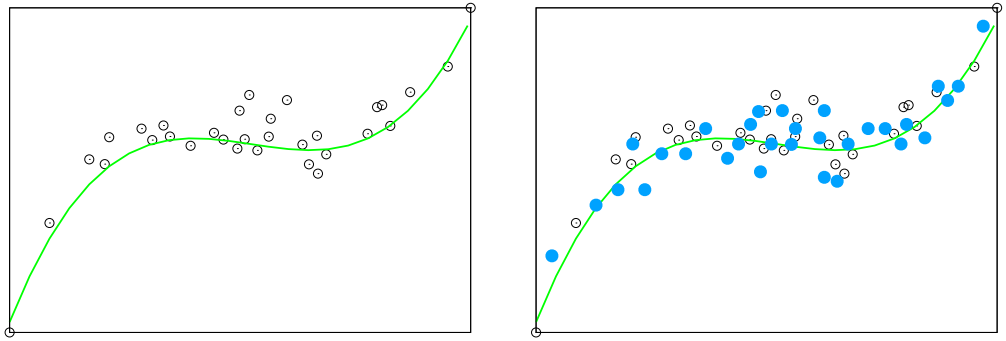
- How do we determine which is better model?

Generalization

- we say a predictor **generalizes** if it performs as well on unseen data as on training data (we will formalize this in the next lecture: bias-variance trade off)
- the data used to train a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it performs well on in-sample data but does not perform well on out-of-sample data

Generalization

- we say a predictor **generalizes** if it performs as well on unseen data as on training data (we will formalize this in the next lecture: bias-variance trade off)
- the data used to train a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it performs well on in-sample data but does not perform well on out-of-sample data



- **train** a cubic predictor on 32 (**in-sample**) white circles: Mean Squared Error (MSE) 174
- **predict** label y for 30 (**out-of-sample**) blue circles: MSE 192
- In this case, we can conclude that this predictor/model generalizes, as in-sample MSE $\simeq$ out-of-sample MSE

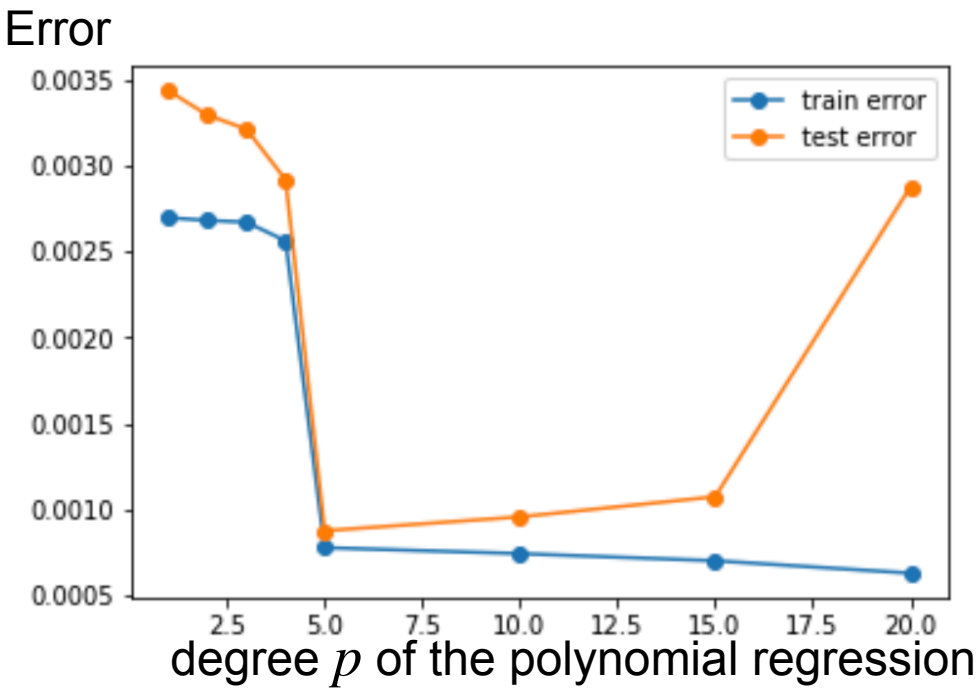
Split the data into training and testing

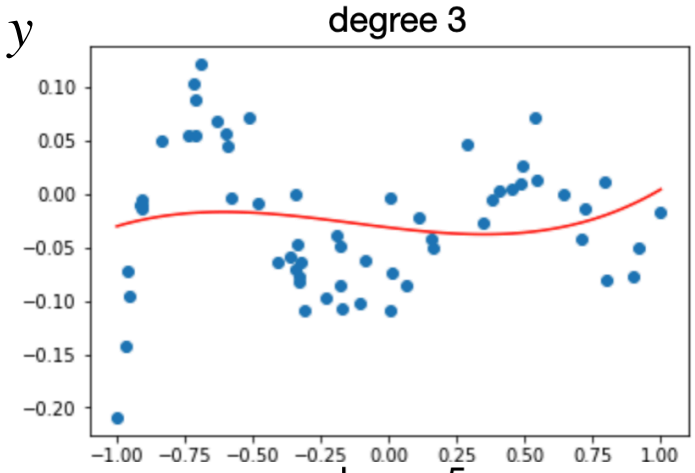
- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set
- selection of data train/test should be done randomly (80/20 or 90/10 are common)
- **training set** used to train the model
  - $$\text{minimize} \quad \mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$$
- **test set** used to evaluate the model
  - $$\mathscr{L}_{\text{test}}(\hat{w}_{\text{MLE}}) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T \hat{w}_{\text{MLE}})^2$$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or choosing some models class**

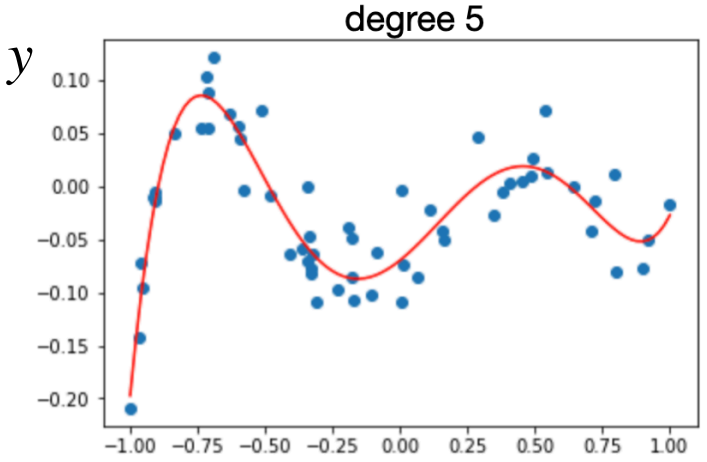
Demo on polynomial feature linear regression

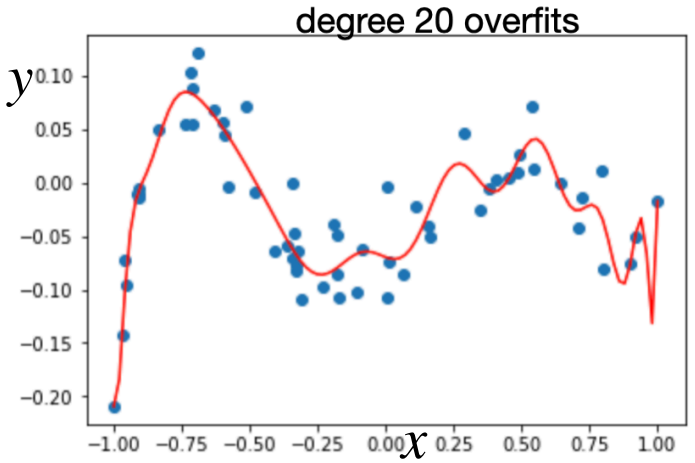
https://jupyter.rttl.uw.edu/2025-autumn-cse-446-a/hub/user-redirect/lab/tree/L3_polynomial.ipynb

Train/test error vs. complexity







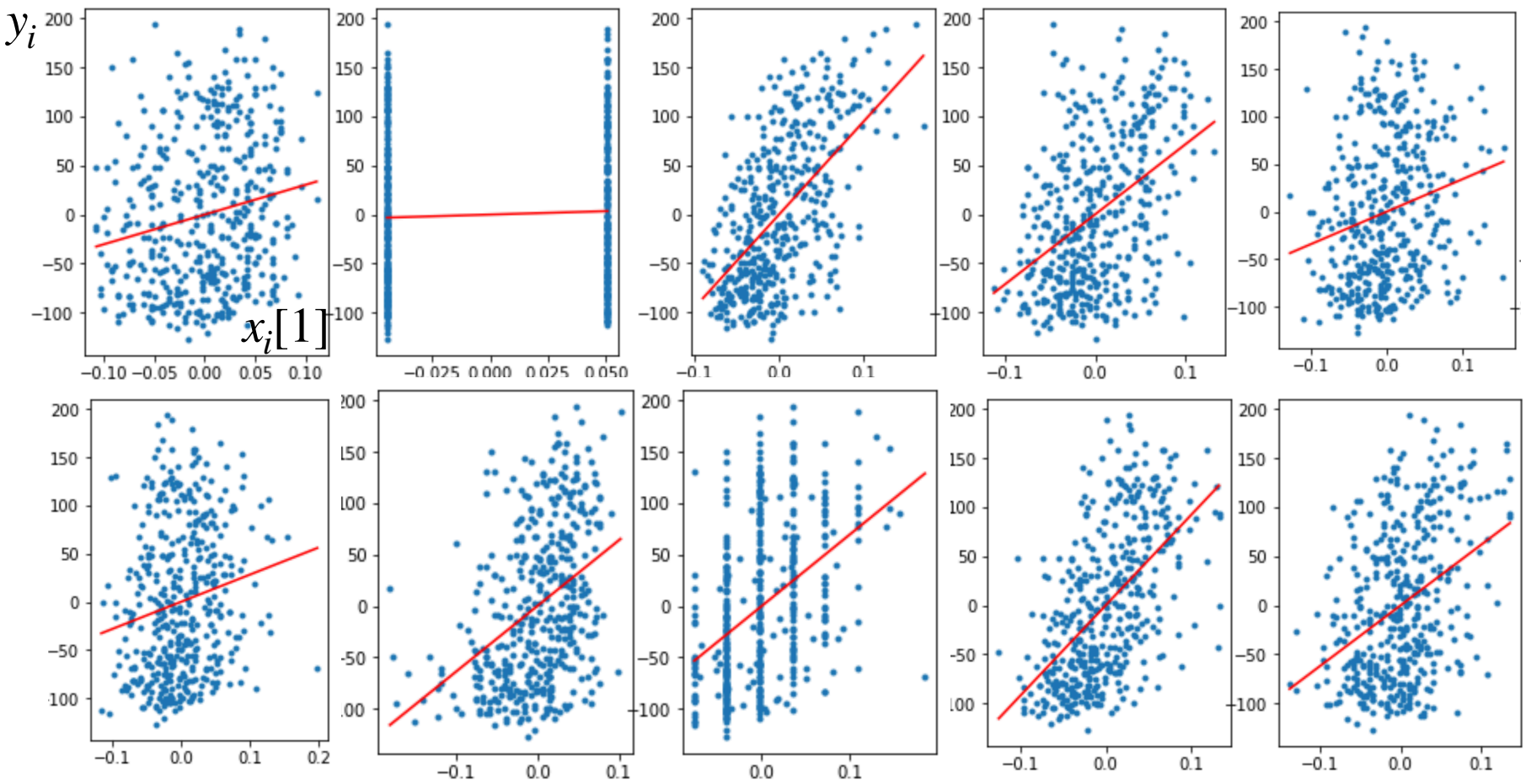


- Degree $p = 5$, since it achieves **minimum test error**
- **Train error** monotonically decreases with model complexity
- **Test error** has a U shape

test set should never be used in training or picking degree

Another example: Diabetes

- Example: Diabetes
  - 10 explanatory variables
  - from 442 patients
  - we use half for train and half for validation



Demo on linear regression

Features	Train MSE	Test MSE
All	2640	3224
S5 and BMI	3004	3453
S5	3869	4227
BMI	3540	4277
S4 and S3	4251	5302
S4	4278	5409
S3	4607	5419
None	5524	6352

- **test MSE is the primary criteria for model selection**
- Using only 2 features (S5 and BMI), one can get very close to the prediction performance of using all features
- Combining S3 and S4 does not give any performance gain

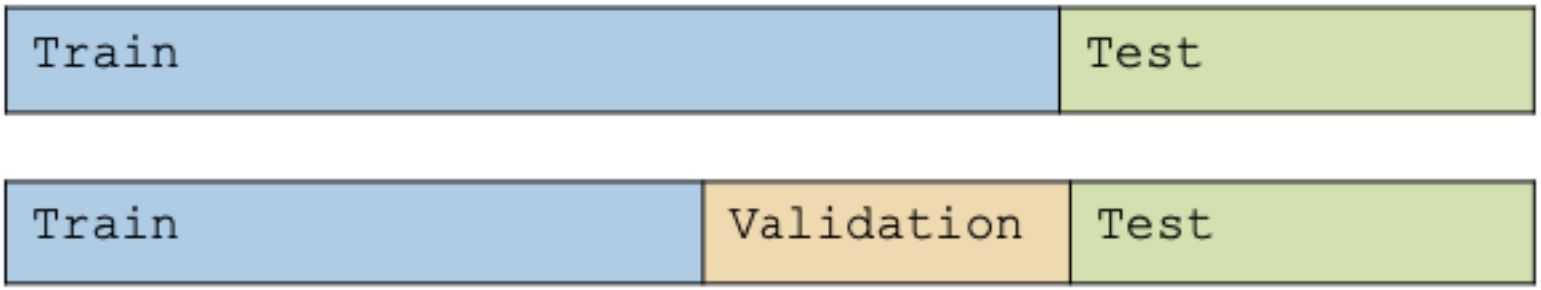
Cross-validation

Choosing model class

- How do we pick hyper-parameters like number of features to use or degree of polynomial to use?
- Never use **test data** in training a model or on any choices made in training.
- **Test data** is only to be used in reporting the test accuracy or loss, in the end.

Validation Set

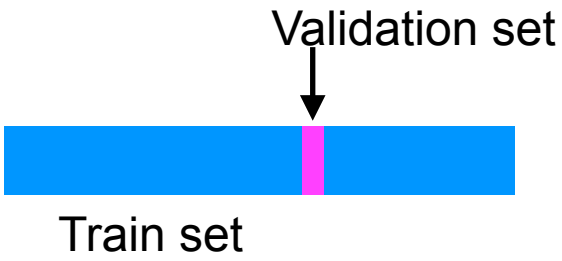
- Simplest approach is to add another dataset partition called the **validation set**



- Each set has its own role
  - **Train Set**: Used to train a model of each model complexity
  - **Validation Set**: Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  - **Test Set**: Once the model is chosen, can get an estimate of future error by testing on this set
- Leave-one-out cross validation and k-fold cross validation

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - D ← original data to be split into train/validation
  - D\j ← training data with j-th data point (x_j ,y_j) removed
  - {(x_j ,y_j)} ←a set with a single data point (x_j ,y_j)
- Learn model $f_{D\setminus j}$ with D\j dataset
- Estimate true error as squared error on predicting y_j: $\text{error}_j(f_{D\setminus j}) = (y_j - f_{D\setminus j}(x_j))^2$
  - This **validation error** gives an unbiased estimate of **true error**, $\text{error}_{true}(f_{D\setminus j})$ achieved by the model.
    where $\text{error}_{true}(f_{D\setminus j})$:=



(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - D ← original data to be split into train/validation
  - D\j ← training data with j-th data point (x_j ,y_j) removed to validation set
  - {(x_j ,y_j)} ←a set with a single data point (x_j ,y_j)
- Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset
- Estimate true error as squared error on predicting y_j: $\text{error}_j(f_{D\setminus j}) = (y_j - f_{D\setminus j}(x_j))^2$
  - This give an unbiased estimate of $\text{error}_{true}(f_{D\setminus j})$ achieved by the model

- LOO cross validation:
  1. For each data point you leave out, learn a new classifier $f_{D\setminus j}( )$.
  2. Estimate error as:

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

(LOO) Leave-one-out cross validation

- Consider a validation set with 1 example:
  - D ← original data to be split into train/validation
  - D\j ← training data with j-th data point (x_j ,y_j) removed to validation set
  - {(x_j ,y_j)} ←a set with a single data point (x_j ,y_j)
- Learn model $f_{D\setminus j}$ with $D\setminus j$ dataset
- Estimate true error as squared error on predicting y_j: $\text{error}_j(f_{D\setminus j}) = (y_j - f_{D\setminus j}(x_j))^2$
  - This give an unbiased estimate of $\text{error}_{true}(f_{D\setminus j})$ achieved by the model.

- Repeat LOO cross validation with various choices of **model class**
  - LOO cross validation:
    1. For each data point you leave out, learn a new classifier $f_{D\setminus j}(\ )$.
    2. Estimate error as: $$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\setminus j}(x_j))^2$$
  - Choose the hyper-parameter that achieves smallest error_{LOO}
  - Train a new model on the entire D with the newly chosen **model class**

LOO cross validation is (almost) unbiased estimate!

- When computing **LOOCV error, we only use $n-1$ data points**
  - So it's not estimate of true error of learning with n data points
  - learning with less data typically gives worse answer ⇒ Usually validation error is pessimistic
  - but that bias is small, since $n-1$ is very close to n

- LOO is almost unbiased, and it is common to use LOO error for **model class** selection
  - E.g., picking degree is a model class selection since, for example, a set of all degree-4 polynomial functions is a **model class**

- But, LOOCV requires a lot of computational time
  - Suppose you have 100,000 data points
  - You implemented a great version of your learning algorithm that 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**
  - $D_1,\ldots,D_k$
- For each *i*
  - Learn model $f_{D\setminus Di}$ using data point not in D_i
  - Estimate error of $f_{D\setminus Di}$ on validation set D_i:

1	2	3	4	5
Train	Train	Validation	Train	Train

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

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

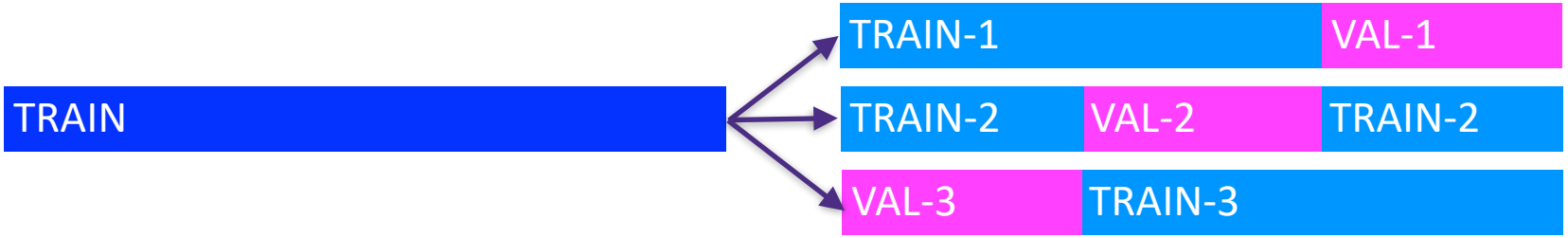
$$\text{error}_{k-fold} = \frac{1}{k} \sum_{i=1}^{k} \text{error}_{D_i}(f_{D\setminus D_i})$$

- k-fold cross validation properties:
  - Much faster to compute than LOO
  - More (pessimistically) biased – using much less data, only $\frac{k-1}{k}n$
  - 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 degree



- Model assessment: Use **TEST** to assess the accuracy of the model you output
  - **Never train or choose hyper-parameters based on the test data**

Example 1

- **You wish to predict the stock price of zoom.us given historical stock price data**
- **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 entire dataset:**

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 break data into train and test dataset.**
- **We train linear regression on these selected features on the training set. We compute the test error and report it**
- **What's wrong with this procedure?**

Recap

- 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
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., mean squared error on TRAIN
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features
  - Justifying the accuracy of the estimate
    - E.g., report TEST error

Questions?

- Good questions on Ed Discussion
  - Will we be tested in “bias”?
    - Bias shows up in many places, and you will have to know the concept. Anything that is taught in lectures can show up in the exams.
  - Why do we use x to denote a column vector and not a row vector?
  - θ^* is the same as θ_*, it is just my writing that is not always consistent
  - What is θ^*?
    - The reason it is unnatural to think about θ^* is that it is something that does not exist in reality. Only time it exists is when you generate simulated data yourself (like in lecture notes and homework).
    - The right interpretation is that we hypothesize that nature has chosen to generate the data from a distribution, which can be written as $P(\cdot\,; \theta^*)$.
    - Whether this assumption is correct or not, we are deciding to go ahead with our MLE process.
    - That gives us some MLE estimate and corresponding distribution $P(\cdot\,; \theta^*_{\text{MLE}})$. What we do with it, and what we believe about it is up to us. (Hence you need to check your accuracy on a holdout set, which we will learn later)