Linear Regression with Basis Functions

Read: Murphy 4.2, Murphy 11 ÷ 11.2 Code: demo_polynomial.ipynb

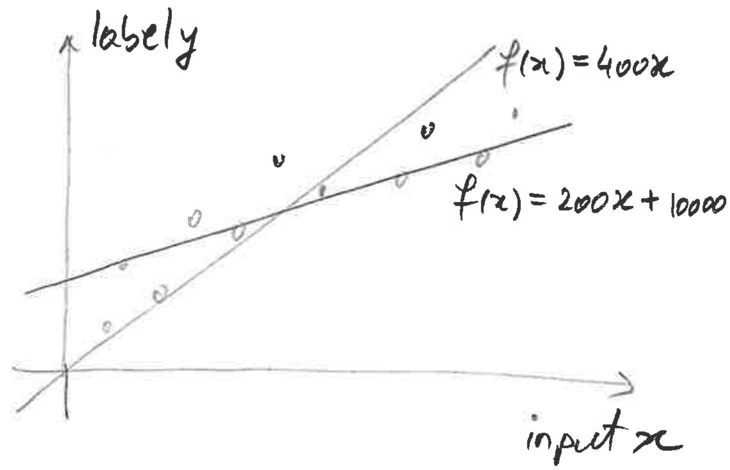
> Recap of Linear Regression

We fit a $\boxed{\text{linear model}}$ with intercept:

$\boxed{y_i \simeq w^T \cdot x_i + b}$ or equivalently $\boxed{y_i = w^T x_i + b + \varepsilon_i}$

with model parameters $(w \in \mathbb{R}^d, b \in \mathbb{R})$ that $\boxed{\text{minimizes}}$ ℓ_2-loss:

$$\mathcal{L}(w,b) = \sum_{i=1}^{n} \underbrace{(y_i - (w^T x_i + b))}_{\text{error } \varepsilon_i}{}^2,$$

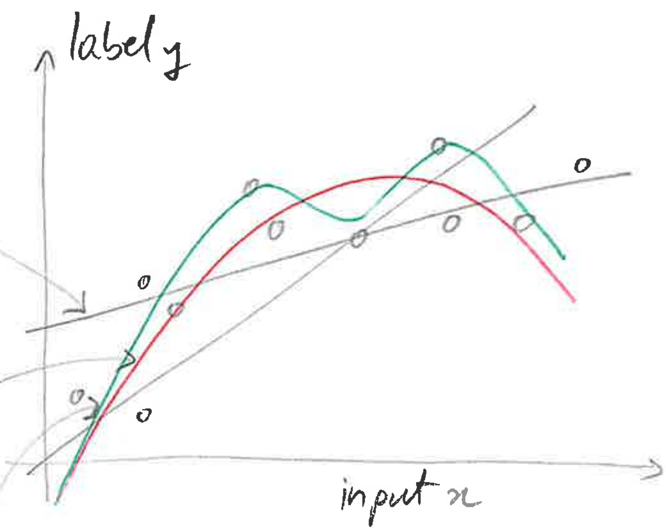


Quadratic Regression in 1-dimension

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

→ **Linear** model with parameter (b, w_1):

$$\boxed{\hat{y}_i = b + w_1 x_i}$$



→ **Quadratic** model with parameter $\left(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}\right)$:

$$\boxed{\hat{y}_i = b + w_1 x_i + w_2 x_i^2}$$

$$h(x_i) = \begin{bmatrix} 1 \\ x_i \\ x_i^2 \end{bmatrix}, \quad \hat{y} = h(x_i)^T \cdot \begin{bmatrix} b \\ w_1 \\ w_2 \end{bmatrix}$$

Stil linear regression → linear combination of non-linear functions of the input variables.

→ **Degree-p** polynomial model with parameter $\left(b, w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}\right)$:

$$\boxed{\hat{y}_i = b + w_1 x_i + w_2 x_i^2 + \dots + w_p x_i^p}$$

→ **General p-features** with parameter $w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}$

$$\boxed{\hat{y}_i = \langle w, h(x_i) \rangle}$$ where $h : \mathbb{R} \to \mathbb{R}^p$

Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x),\ x^2,\ \sin(x),\ \sqrt{x} \right]^T$$

How do we learn w?

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

- See also Bishop pp142
- Murphy 13.1

$$\hat{\omega} = \arg\min_{\omega} \| H\omega - y \|_2^2$$

$$\boxed{\hat{\omega} = \left(H^T \cdot H \right)^{-1} \cdot H^T \cdot y}$$

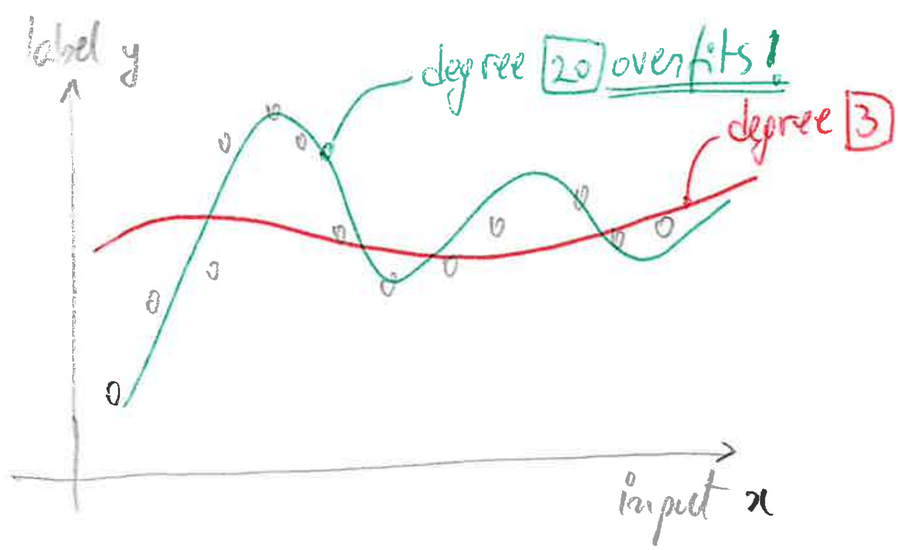
(see previous lecture notes for a derivation)

For a new test point x, predict:

$$\hat{y} = \langle \bar{\omega}, h(x) \rangle$$

Which $\boxed{p}$ should we choose?

See: Code Time!
demo_polynomial.ipynb
(next page)



Generalization

> We say a predictor generalizes if it performs well on unseen data as on training data

> Data used to train a predictor is training data. (in-sample data)

> We want a predictor to work well on out-of-sample data!

> A predictor fails to generalize if it performs well on training data, but, it does not perform well on out-of-sample data!
(test data)

Example

> train a cubic predictor on 32 train data : MSE = 174

> predict label y for 30 test data : MSE = 192

This predictor generalizes effectively? Yes because MSE's are similar.

Split Data

into training and testing (e.g., 90/10)

> A way to mimic how the predictor performs on unseen data

> Given single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$

- Training set – used to train model

$$\text{minimize } \mathcal{L}_{train}(w) = \frac{1}{|S_{train}|} \sum_{i \in S_{train}} (y_i - x_i^T w)^2$$

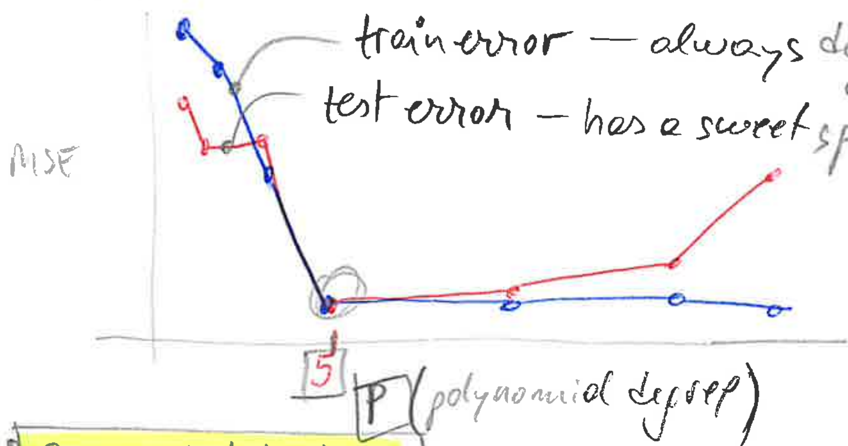
- Test set – evaluate the model

$$\mathcal{L}_{test}(w) = \frac{1}{|S_{test}|} \sum_{i \in S_{test}} (y_i - x_i^T w)^2$$

> NOTE: This assumes test set is similar to unseen data.

Code time: See demo_polynomial.ipynb

Summary of .ipynb example:



train error — always decreases as we increase complexity of model

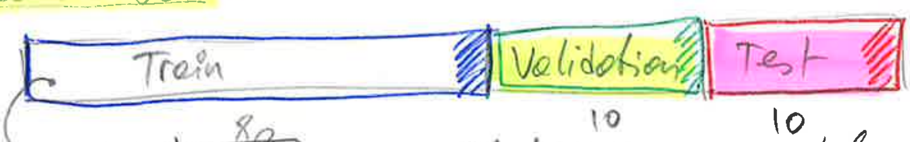
test error — has a sweet spot — correspond to true complexity of model.

p=5 achieves minimum test error

Cross-Validation

→ How to pick the number of basis functions ?

→ **Validation set**



- used to train a model for each model complexity
- Add another portition called validation set.
  Used as a hold-out to evaluate each model.
  Generally, choose model complexity w/ lowest validation error.
- Test - once model is chosen, use to get an estimate of future error. Only do this once.
  This would be what you report.

LOO | Leave-one-Out Cross Validation

- A validation set with 1 example

- D training data. (n samples)
- $D \backslash j$ training data w/ j-th data point (x_j, y_j) moved to validation set.

- Learn model $f_{D\backslash j}$ with $D\backslash j$ dataset. ($n-1$ samples)
- Estimate true error as squared error on predicting y_j

Unbiased estimate of $\boxed{error_{true}(f_{D\backslash j})}$

- Computationally expensive!!!

- Average over all data points j.
  - For each data point we leave out, learn new model $f_{D\backslash j}$
  - Estimate error as:

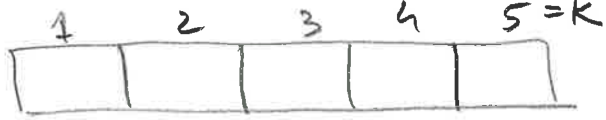
$$\boxed{error_{LOO} = \frac{1}{n} \sum_{j=1}^{n} (y_j - f_{D\backslash j}(x_j))^2}$$

- LOO is almost unbiased!
  - uses $\boxed{N-1}$ data points => not estimate of true error of learning w/ $\boxed{N}$ datapoints.

- For example $n = 100\,000$ datapoints
  - Learning algorithm takes 1 second.
  - Computing LOO will take $\sim$ 1 day!

K-fold Cross Validation

→ Randomly divide training data into $\boxed{k}$ equal parts $D_1, D_2, \dots, D_k$



→ For each i

- Learn model $f_{D \setminus D_i}$ using data points not in D_i
- Estimate error of $f_{D \setminus D_i}$ on validation set D_i as:

$$error_{D_i} = \frac{1}{|D_i|} \sum_{(x_j, y_j) \in D_i} \left(y_j - f_{D \setminus D_i}(x_j)\right)^2$$

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

$$\boxed{error_{k\text{-fold}} = \frac{1}{k} \sum_{i=1}^{k} error_{D_i}}$$

→ Properties:

- much faster to compute than LOO
- more (pessimistically) biased – using much less data for training; only $\frac{n(k-1)}{k}$
- usually $\boxed{k = 10}$

- use Test portion of dataset to asses accuracy of the mode you output.
- Never ever train or choose parameters based on the test data!

Recap

Learning is:

> Collect some data
  eg: housing info and sale price
> Randomly split dataset into $\boxed{\text{Train}}$, $\boxed{\text{Val}}$ and $\boxed{\text{Test}}$
  eg: 80% 10% 10%
> Choose a hypothesis class or model
  eg: Linear with non-linear feature transformations
> Choose a loss function
  eg: Least Squares on $\boxed{\text{Train}}$ dataset
> Choose an optimization procedure
  eg: Set derivative to zero to obtain estimator
  estimate generalization error
  cross-validation on $\boxed{\text{Val}}$ dataset to pick number of features
  pick hyper-parameters
> Justify accuracy of the estimate
  eg. report $\boxed{\text{Test}}$ dataset error.