EE445 Mod2-Lec2: Least Squares Data Fitting

References:

- [VMLS]: Chapter 13

Outline

- least squares data fitting including feature transformation
- validation and generalization $\leftarrow$
- feature engineering

Least Squares Data Fitting: What is it?

training data

observation

- In [Mod2-L1] we saw the "least squares approximate solution" to $Ax = b$
- Data fitting is one of the most important applications of least squares
- **Goal**: find a mathematical model, or an approximate model, of some relation, given some observed data say from experiments or real-world phenomena
- **Set-up**: Consider $x \in \mathbb{R}^n$ and $y \in \mathbb{R}$. Suppose we **model** x and y as being approximately related by $f : \mathbb{R}^n \to \mathbb{R}$:

$$y \approx f(x)$$

$$y \approx Ax$$

- **Components**:
  - feature vector (independent variables): x
  - outcome (response/dependent variable): y
- **Challenge**: We do not know f and need to approximate it

Data & Model

$(\vec{x}^{(i)}, y^{(i)})$

- Consider m feature vectors and corresponding scalars:

$$\underbrace{x^{(1)}, \ldots, x^{(m)}}\in \mathbb{R}^n, \quad \text{and} \quad \underbrace{y^{(1)}, \ldots, y^{(m)}} \in \mathbb{R}$$

- **Data pair** (observations or samples):

$$[p^{(i)}\ h^{(i)}\ t^{(i)}] = x^{(i)}$$

$$[t_+^{(i)}] = y^{(i)}$$

- **Model**: find a model $y \approx f(x)$

$$(y^{(i)} - \hat{y}^{(i)})^2 \Rightarrow \text{want small}$$

$$\hat{y}^{(i)} \overset{!!}{=} f(x^{(i)})$$

- **prediction**: $\hat{y} = f(x)$

$\uparrow$

Class of Linear Models

- **Linear Models**: consider p basis functions or feature mappings $f_i : \mathbb{R}^n \to \mathbb{R}$ and *model parameters* θ_i:

unknown

$$f(x) = \theta_1 f_1(x) + \theta_2 f_2(x) + \cdots + \theta_p \underset{x^{(i)}}{f_p(x)}$$

- the feature mappings f_i are chosen based on our prior knowledge of the problem
- Once they are chosen, they are a known part of the problem of data fitting.
- The remaining unknown is the model parameters $\theta = \underline{(\theta_1, \ldots, \theta_p)}$
- **Goal**:

devise a scheme to predict $\hat{y}$ given $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$

Learn / estimate θ : $\hat{\theta} \Rightarrow \hat{y} = \hat{f}(x) = \hat{\theta}_1 f_1(x) + \cdots + \hat{\theta}_p f(x)$

Prediction Error

- Consider a fixed value of θ and let $f(x) = \sum_{j=1}^{p} \theta_j f_j(x)$
- **prediction error**: For $(x^{(i)}, y^{(i)})$, the prediction error or residual

$$r^{(i)} = y^{(i)} - \hat{y}^{(i)}, \quad \text{where} \quad \underset{\leftarrow}{\hat{y}^{(i)}} = \hat{f}(x^{(i)})$$

- **Notation**: define the following vectors
  - observation:

    $$y = (y^{(1)}, \ldots, y^{(m)})$$
  - prediction:

    $$\hat{y} = (\hat{y}^{(1)}, \ldots, \hat{y}^{(m)})$$
  - residual:

    $$r = (r^{(1)}, \ldots, r^{(m)})$$

$$\Big\} \ \mathbb{R}^m \text{ vectors}$$

Prediction Error: Measures

- **Root-Mean-Square (RMS) prediction error**:

$$\mathrm{rms}(z) = \sqrt{\frac{z_1^2 + \cdots + z_m^2}{m}} = \underbrace{\frac{\|z\|_2}{\sqrt{m}}}$$

r : residual

$\mathrm{rms}(r)$

- **Relative prediction error**:

$$\frac{\mathrm{rms}(r)}{\mathrm{rms}(y)} \swarrow$$

$$z_i = (y - \mathrm{avg}(y)\underline{1})_i$$

Least Squares Model Fitting

$f_1, \ldots, f_p \rightarrow [q^{(i)}\ h^{(i)}\ t^{(i)}] = x^{(i)}$

- **Optimization Problem**: $\min_{\theta \in \mathbb{R}^p} \|r\|^2$
- **Equivalent Least Squares Problem:** with $\hat{y}^{(i)} = \hat{f}(x^{(i)})$, we write

$$\underbrace{\hat{y}^{(i)} = a_{i1}\hat{\theta}_1 + a_{i2}\hat{\theta}_2 + \cdots + a_{ip}\hat{\theta}_p} \quad \text{where} \quad a_{ij} = f_j(x^{(i)})$$

$$A = \begin{bmatrix} a_{ij} \end{bmatrix} \in \mathbb{R}^{m \times p}, \qquad \hat{y} = A\theta, \qquad \underbrace{\min_{\theta} \|A\theta - y\|^2} \qquad \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

- The j–th column of A is the j–th basis function, evaluated at each of the data points $x^{(1)}, \ldots, x^{(m)}$.
- The i–th row gives the values of the p basis functions on the i–th data point $x^{(i)}$.

Least Squares Model Fitting

- Data fitting as least squares:

$$\min_{\theta} \|A\theta - \boldsymbol{y}\|_2^2 \quad \text{[Least Squares Problem]}$$

- **Solution**: (cf. [Mod2-L1])

$$\hat{\theta} = \arg\min_{\theta} \|A\theta - y\|^2 = \underbrace{(A^T A)^{-1} A^T}_{A^\dagger} y = A^\dagger y$$

- We say that the model parameter values $\hat{\theta}$ are obtained by least squares fitting on the data set.
- **~~Minimum~~ Mean square error (MSE)**: $\|\boldsymbol{y} - A\hat{\theta}\|_2^2$
- **Minimum mean square error (MMSE)**: $\frac{1}{m}\|\boldsymbol{y} - A\hat{\theta}\|_2^2$

Intuition Road Map

$\underbrace{a}\, x + b$ ↗

↙ one feature

- [Univariate] Least squares fit with a constant $\to$ $\hat{y}$ is average of data y
- [Univariate] Straight line fit (linear/affine model) $\to$ $\hat{y}$ is linear combination of average of y and linear scaling of $(x - \text{avg}(x))$
- More complex feature maps such as polynomials, piecewise linear functions etc.
- [Multivariate] Regression

↑ many features

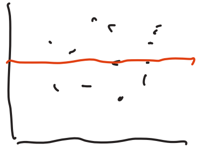
[Univariate]: Least squares fit with a constant

$A^{m \times p}$

$f(x) = \theta_1 \cdot f_1(x) = \theta_1 \cdot 1 = \theta_1$

- **set-up**: Let $p = 1$ and $f_1(x) = 1$ for all x
- **model**: $f(x) = \theta_1$ and $A = \mathbf{1} \in \mathbb{R}^{m \times 1}$—i.e., least squares fitting in this case is the same as choosing the best constant value θ_1 to approximate the data $y^{(1)}, \ldots, y^{(m)}$
- **Solution**: $A^T A = \mathbf{1}^T \mathbf{1} = m$

$$\hat{\theta}_1 = \underbrace{(A^T A)}^{-1} A^T y = \frac{1}{m} \mathbf{1}^T y = \frac{\sum_{i=1}^{m} y_i}{m} = \underline{\mathrm{avg}}(y)$$



- The RMS fit to the data (i.e., the RMS value of the optimal residual) is

$$\mathrm{rms}(y - \mathrm{avg}(y)\mathbf{1}) = \underline{\mathrm{std}}(y)$$

- **Interpretation**: the average value and the standard deviation of the outcomes, as the best constant fit and the associated RMS error, respectively.

[Univariate]: Straight Line Fit

- Suppose that $n = 1$—i.e., the feature vector $x \in \mathbb{R}$ is a scalar and so is $y \in \mathbb{R}$
- **Visualization**: Data points $\{(x^{(i)}, y^{(i)})\}_{i=1}^m$ can thus be plot in the plane
- **Straight Line fit**: Let feature mappings be $\underline{f_1(x) = 1}$ and $\underline{f_2(x) = x}$ so that our model is

$$f(x) = \theta_1 + \theta_2 x \quad \text{[Straight-Line]} \qquad = \theta_1 \cdot f_1(x) + \theta_2 f_2(x)$$

- **Note**: this is often called a "linear model" however its really *affine*
- The data matrix $A \in \underline{\mathbb{R}^{m\times 2}}$ is given by

$$A = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix} = \begin{bmatrix} 1 & x \end{bmatrix}$$

[Univariate]: Straight Line Fit (P1)

- **Solution**: as before we have $\theta = \underline{(A^\top A)^{-1}}A^\top \boldsymbol{y}$
- Explicit solution:

$$A^\top A = [1\ x]^\top [1\ x] = \underbrace{\begin{bmatrix} m & 1^\top x \\ 1^\top x & x^\top x \end{bmatrix}}\in \mathbb{R}^{2\times 2}, \quad \underline{A^\top y} = \begin{bmatrix} 1^\top y \\ x^\top y \end{bmatrix} \in \mathbb{R}^2$$

$$\hat{\theta} = \frac{1}{m x^\top x - (1^\top x)^2} \underline{\begin{bmatrix} x^\top x & -1^\top x \\ -1^\top x & m \end{bmatrix}} \begin{bmatrix} 1^\top y \\ x^\top y \end{bmatrix} \qquad \frac{\underline{m^2}}{m^2}$$

$$= \frac{1}{\mathrm{rms}(x)^2 - \mathrm{avg}(x)^2} \underbrace{\begin{bmatrix} \mathrm{rms}(x)^2 & -\mathrm{avg}(x) \\ -\mathrm{avg}(x) & 1 \end{bmatrix}} \begin{bmatrix} \mathrm{avg}(x) \\ \frac{1}{m} x^\top y \end{bmatrix}$$

[Univariate]: Straight Line Fit (P2)

$f(x) = \theta_1 + \theta_2 x$

- Explicit solution:

$$\begin{bmatrix}\hat{\theta}_1 \\ \hat{\theta}_2\end{bmatrix} = \hat{\theta} = \frac{1}{m\boldsymbol{x}^\top\boldsymbol{x} - (\mathbf{1}^\top\boldsymbol{x})^2}\begin{bmatrix}\boldsymbol{x}^\top\boldsymbol{x} & -\mathbf{1}^\top\boldsymbol{x} \\ \underbrace{-\mathbf{1}^\top\boldsymbol{x} \quad m}\end{bmatrix}\begin{bmatrix}\mathbf{1}^\top\boldsymbol{y} \\ \boldsymbol{x}^\top\boldsymbol{y}\end{bmatrix}$$

slope

- Multiply by m^2/m^2 (put the m^2 in the numerator on the scalar term and the $1/m$ terms on the vector and matrix, resp.) to get

$$\hat{\theta}_2 = \underbrace{\frac{\text{std}(y)}{\text{std}(x)}}\rho \quad , \quad \rho = \frac{(x - \text{avg}(x)\mathbb{1})^\top(y - \text{avg}(y)\mathbb{1})}{m\ \text{std}(x)\ \text{std}(y)}$$

[Univariate]: Straight Line Fit

- **Solution**:

$$\hat{\theta} = \frac{1}{\mathtt{rms}(\boldsymbol{x})^2 - \mathtt{avg}(\boldsymbol{x})^2} \begin{bmatrix} \mathtt{rms}(\boldsymbol{x})^2 & -\mathtt{avg}(\boldsymbol{x}) \\ -\mathtt{avg}(\boldsymbol{x}) & 1 \end{bmatrix} \begin{bmatrix} \mathtt{avg}(\boldsymbol{y}) \\ \frac{1}{m}\boldsymbol{x}^\top \boldsymbol{y} \end{bmatrix}$$

- $\hat{\theta}_2$ is the slope and can be expressed as follows

$$\hat{\theta}_2 = \frac{m\boldsymbol{x}^\top \boldsymbol{y} - (\mathbf{1}^\top \boldsymbol{x})(\mathbf{1}^\top \boldsymbol{y})}{m\boldsymbol{x}^\top \boldsymbol{x} - (\mathbf{1}^\top \boldsymbol{x})^2} = \frac{(\boldsymbol{x} - \mathtt{avg}(\boldsymbol{x})\mathbf{1})^\top (\boldsymbol{y} - \mathtt{avg}(\boldsymbol{y})\mathbf{1})}{\|\boldsymbol{x} - \mathtt{avg}(\boldsymbol{x})\mathbf{1}\|_2^2} = \frac{\mathtt{std}(\boldsymbol{y})}{\mathtt{std}(\boldsymbol{x})}\rho$$

where ρ is the correlation coefficient [VMLS, Ch. 3] defined by

$$\rho = \frac{(\boldsymbol{x} - \mathtt{avg}(\boldsymbol{x})\mathbf{1})^\top (\boldsymbol{y} - \mathtt{avg}(\boldsymbol{y})\mathbf{1})}{m\,\mathtt{std}(\boldsymbol{x})\mathtt{std}(\boldsymbol{y})} \quad \text{and} \quad \mathtt{std}(\boldsymbol{x}) = \frac{\|\boldsymbol{x} - \mathtt{avg}(\boldsymbol{x})\mathbf{1}\|_2}{\sqrt{m}}$$

[Univariate]: Straight Line Fit

$f_1(x) = 1$

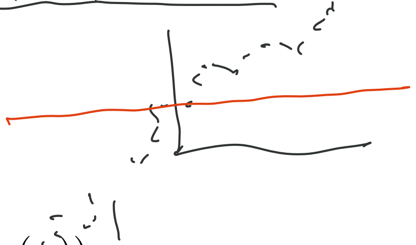
$f_2(x) = x$

- $\hat{\theta}_1$ is the intercept and it also has a simple expression. $\implies$

$$\text{[Normal Equations]: } \underline{(A^\top A)\theta = A^\top y} \implies \underline{m\theta_1 + (\mathbf{1}^\top \boldsymbol{x})\theta_2 = \mathbf{1}^\top \boldsymbol{y}}$$

so that

$$\hat{\theta}_1 = \underline{\texttt{avg}(\boldsymbol{y})} - \underline{\hat{\theta}_2 \texttt{avg}(\boldsymbol{x})}$$



- Hence,

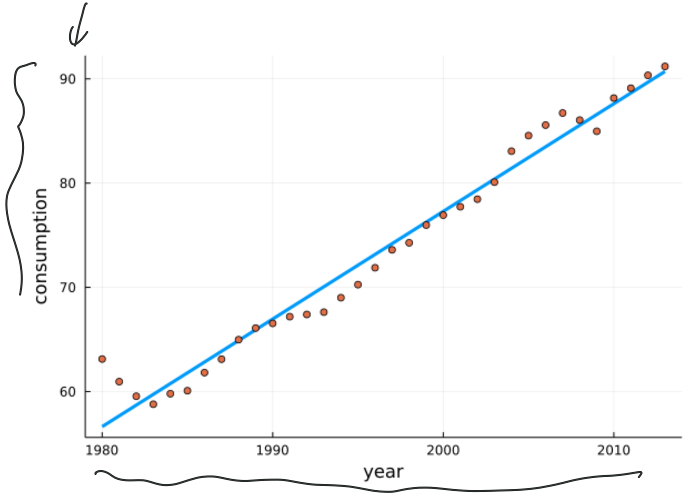
$$\hat{y} = f(x) = \underline{\texttt{avg}(\boldsymbol{y})} + \rho \frac{\texttt{std}(\boldsymbol{y})}{\underline{\texttt{std}(\boldsymbol{x})}} \underline{(x - \texttt{avg}(\boldsymbol{x}))}$$

- When $\texttt{std}(\boldsymbol{y}) \neq 0$ we have

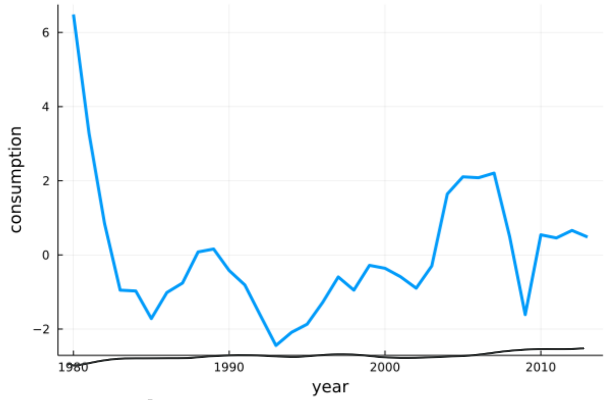
$$\boxed{\frac{\hat{y} - \texttt{avg}(\boldsymbol{y})}{\underline{\texttt{std}(\boldsymbol{y})}} = \rho \frac{x - \texttt{avg}(\boldsymbol{x})}{\underline{\texttt{std}(\boldsymbol{x})}}}$$

Example: Petroleum Consumption

- Data represents a series of samples of world petroleum consumption y at time $x^{(i)} = i$.
- straight-line fit: $\hat{y}^{(i)} = \theta_1 + \theta_2 i$, $i = 1, \dots, m$ (trend line)



- Slope: θ_2 is the *trend*
- de-trended time series: $\boldsymbol{y} - \hat{\boldsymbol{y}}$
- positive $\rightarrow$ time series above straight-line fit



[Univariate]: Polynomial Fit

$$\begin{array}{cc} 1 & x \\ \hline x^0 & x^1 \end{array}$$

- A simple extension beyond the straight-line fit is a polynomial fit—i.e., where

$$f_i(x) = x^{i-1}, \quad \underbrace{i = 1, \ldots, p}$$

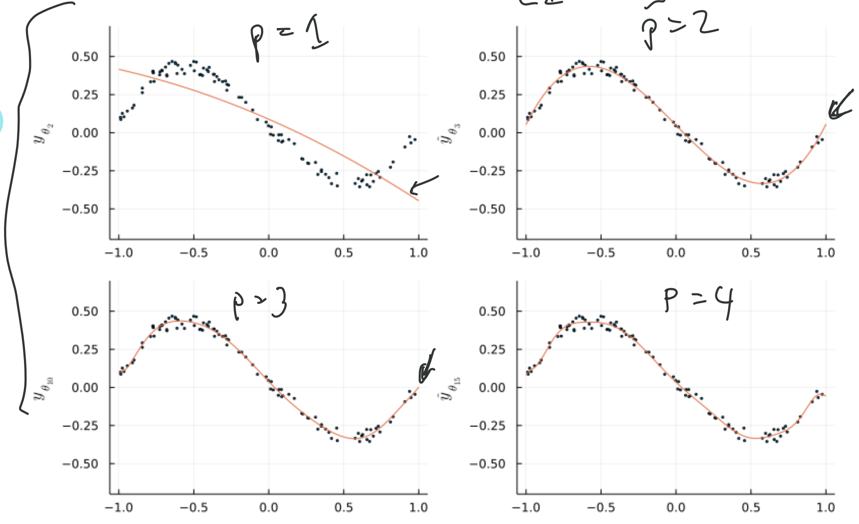
$$\begin{bmatrix} 1 & x^{(1)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix}$$

- f is a polynomial of degree $p-1$:

$$f(x) = \theta_1 + \theta_2 x + \cdots + \theta_p x^{p-1}$$

- Data matrix:

$$A = \underbrace{\begin{bmatrix} 1 & x^{(1)} & \cdots & (x^{(1)})^{p-1} \\ 1 & x^{(2)} & \cdots & (x^{(2)})^{p-1} \\ \vdots & \vdots & & \vdots \\ 1 & x^{(m)} & \cdots & (x^{(m)})^{p-1} \end{bmatrix}}$$

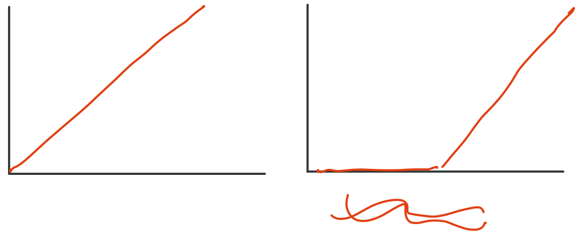


[Univariate]: Piecewise Linear Fit

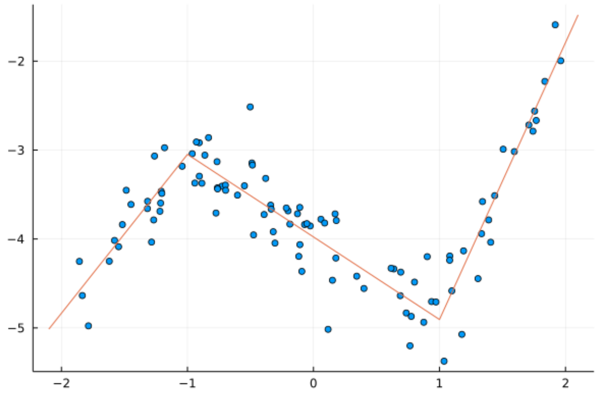
log transform

RBF: radial basis fns.

- Yet another generalization is the piecewise linear fit: a piecewise linear function with knot points $c_1 < c_2 < \cdots < c_k$ is a continuous function that is affine between the knot points.
- **basis functions**: $f_1(x) = 1$, $f_2(x) = x$, $f_{i+2} = (x - c_i)_+ = \max\{x - c_i, 0\}$, $i = 1, \ldots, k$



$$f(x) = \theta_1 \cdot 1 + \sum_{i=0}^{p-2} f_{i+2}(x) \cdot \theta_{i+2}$$



$\theta_1 + \theta_2 x$

[Multivariate]: Regression (Multivariate)

- Regression: $\hat{y} = x^\top \beta + v$, where β is the weight vector and v the offset [VMLS, Ch. 2]
- Let $f_1(x) = 1$ and $f_i(x) = x_{i-1}$, $i = 2, \ldots, n+1$ so that $p = n+1$
- Transforming to data fitting model: $\hat{y} = x^\top \bar{\theta} + \theta_1$, $\bar{\theta} = (\theta_j)_{j=2}^{n+1} = (\theta_2, \ldots, \theta_{n+1})$

$-\beta = \bar{\theta} = (\theta_2, \ldots, \theta_{n+1})$

$-v = \theta_1$

$\min_{b} \|A\theta - y\|^2$

$x^{(i)} = (p, t_-, h)$ $y^{(i)} = t_+$

- Data matrix: $A = [\mathbb{1} \;\; X^\top] \in \mathbb{R}^{m \times (n+1)}$, X feature matrix w/ cols.

$y = (y^{(1)}, \ldots, y^{(m)})$

$\begin{bmatrix} | \\ x^{(1)} \\ | \end{bmatrix}, \ldots, x^{(m)} \in \mathbb{R}^n$ $X = \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix}$

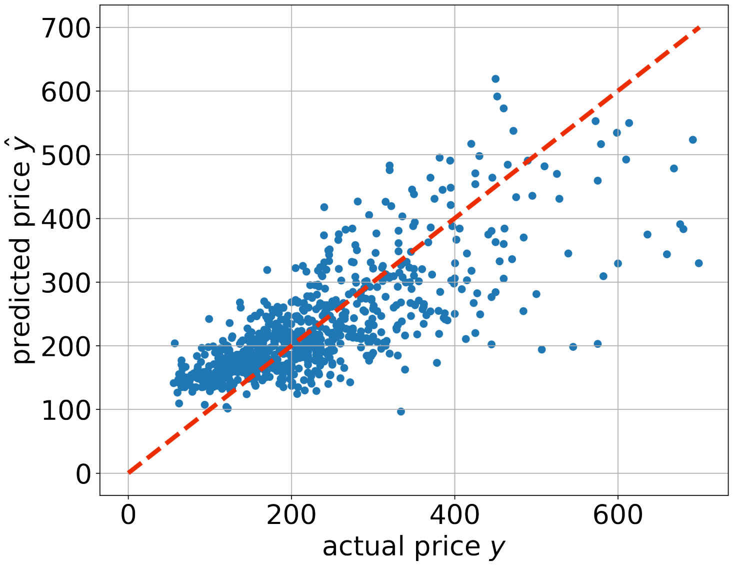
- General data fitting as regression: $\tilde{x} = (f_2(x), \ldots, f_p(x))$ so that $\hat{y} = \tilde{x}^\top \beta + v$
- Comparing to linear parametric model:

$$\hat{y} = \theta_1 f_1(x) + \cdots + \theta_p f_p(x) \quad \text{where} \quad v = \theta_1,\ \beta = (\theta_2, \ldots, \theta_p)$$

Example: House Price Prediction

$$\hat{\theta} = (A^T A)^{-1} A^T y$$
$$\hat{y} = A\hat{\theta} = Py$$

- 774 house sales in Sacramento over 5 day period
- `RMS` fitting error is ≈\$74.8k
- Compare to standard deviation of the data which is \$112.8k
- i.e., basic regression model predicts the prices substantially better than a constant model



y : price

$x^{(i)} = [\text{area}^{(i)}, \#\text{beds}^{(i)}]$, $y^{(i)}$: actual price

$$A\theta = \begin{bmatrix} 1 & \text{area}^{(1)} & \text{bed}^{(1)} \\ \vdots & \vdots & \\ 1 & \text{area}^{(m)} & \text{bed}^{(m)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} \approx \hat{y}$$

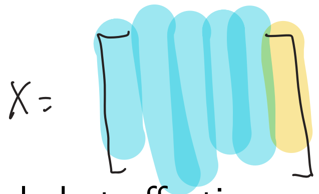
Validation & Generalization

[VMLS, 13.2]

$$y^{(i)} = \theta_1 + (x^{(i)})^T \tilde{\theta} \qquad \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \approx \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ \vdots & & & & \\ 1 & x_1^{(m)} & \cdots & \cdots & x_n^{(m)} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_{n+1} \end{bmatrix}$$

Generalization

- **Goal of model fitting**: not to just achieve a good fit on the given data set, but rather to achieve a good fit on new data that we have not yet seen.
- *How well can we expect a model to predict y for future or other unknown values of x?*
- **Generalization ability**: a model that makes reasonable predictions on new, unseen data has generalization ability, or generalizes
- **Over-fitting**: a model that makes poor predictions on new, unseen data is said to suffer from over-fit



Validation

- A simple but effective method for assessing the generalization ability of a model is called *out-of-sample validation*.
- **Method**:
  - ▶ **step 1**: divide data into training set and test (validation) set: e.g., randomly assign 80% of data pairs to training and 20% to testing
  - ▶ **step 2**: fit model only on training set
  - ▶ **step 3**: evaluate (e.g., via RMS) on test set
- **Assessment**: If the RMS prediction error on the test set is much larger than the RMS prediction error on the training set, we conclude that our model has poor generalization ability.
- **Over-fitting**: When the RMS prediction error on the training set is much smaller than the RMS prediction error on the test set, we say that the model is over-fit.

Model prediction quality & generalization ability: Not the same!

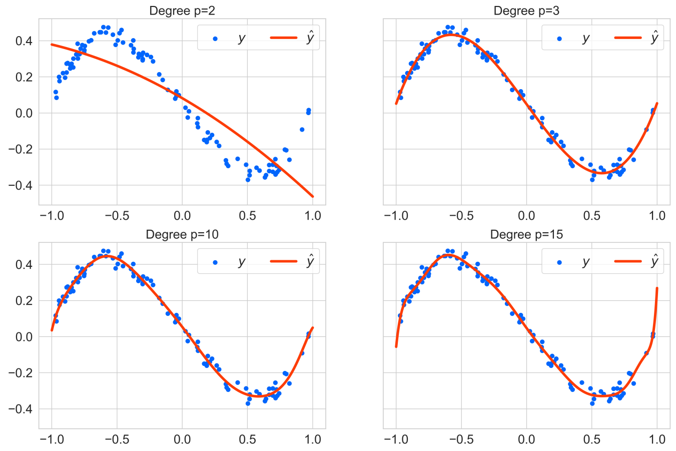
- Caution!: A model can perform poorly and yet have good generalization ability.
- e.g., let $\hat{y} = 0$ always.
- This will perform poorly on both training and test data, however the `RMS` will be similar meaning it has good generalization ability
- **Goal: Performance+generalization**: we seek a model that makes *good predictions on the training data set* and also makes *good predictions on the test data set*.

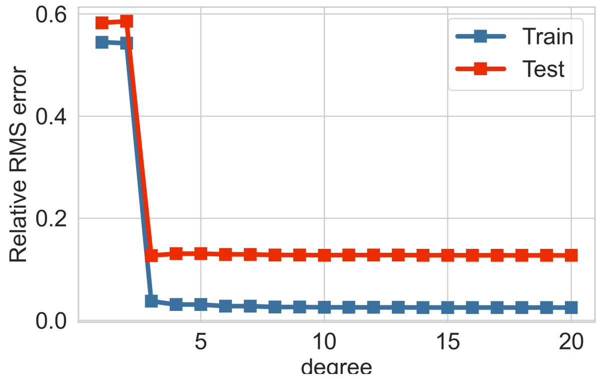
Generalization & Model Choice

$$\to \begin{bmatrix} 1 & \text{area} \\ \vdots & \vdots \\ 1 & \text{area} \end{bmatrix} \text{ vs. } \begin{bmatrix} 1 & \overset{\text{location}}{\overline{\text{area} \quad \text{beds}}} \\ \vdots & \\ 1 & \end{bmatrix}$$

- **Key Question**: *which is the best model?* e.g., straight-line fit, polynomial of degree 2,3,4 etc.
- **Rule of thumb 1 (least error)**: we should choose a model that has test set RMS error that is near the minimum over the candidates
- **Rule of thumb 2 (simplest among them)**: If multiple candidates achieve test set performance near the minimum, we should choose the ‘simplest’ one among these candidates.

Generalization & Model Choice: Example





Cross-validation

- Cross-validation is an extension of out-of-sample validation that can be used to get even more confidence in the generalization ability of a model.
- **Method**: e.g., 10-fold cross-validation
  - Divide the data set into 10 sets (called folds)
  - Fit the model using folds 1–9 as training data and fold 10 as test
  - Fit the model using folds 1–$8, 10$ as training data and fold 9 as test
  - $\vdots$
  - End up with ten models and ten assessments of these models
  - RMS cross-validation error:

$$\sqrt{\frac{\varepsilon_1^2 + \cdots \varepsilon_5^2}{5}}$$

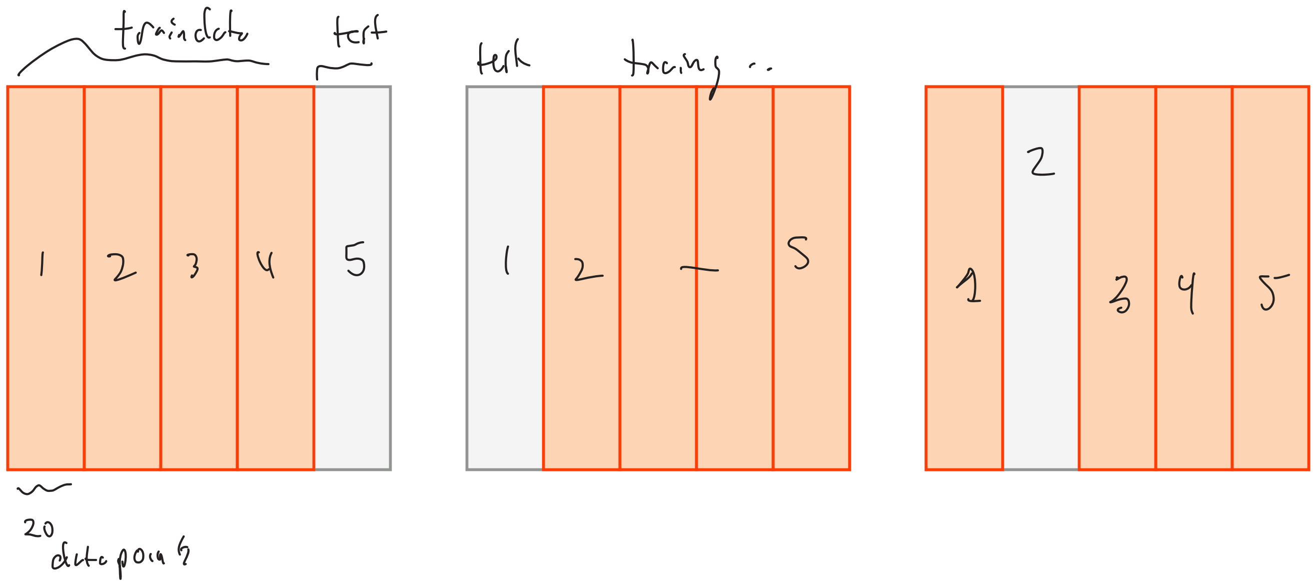
$$\texttt{rms} = \sqrt{\frac{\epsilon_1^2 + \cdots + \epsilon_{10}^2}{10}}, \quad \epsilon_i := \texttt{rms} \text{ error of model } i$$

- Cross validation is used for checking of the selection of basis functions.
- Which model to use?: The models should be not too different, so the choice really should not matter much.

Cross-validation

5-folds, 100 = m

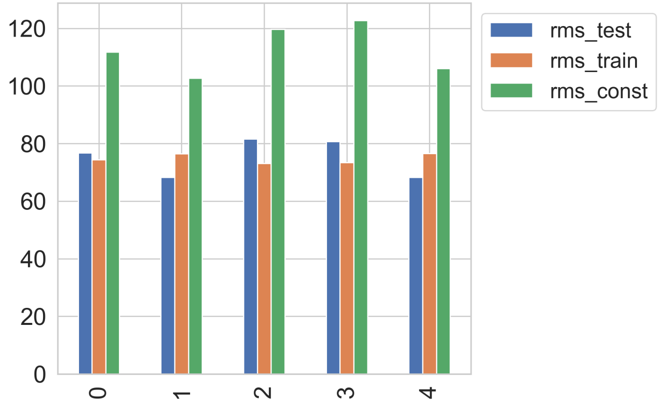
np.random.permutation(n-folds)



Example: House Price Prediction

- Cross-validation to assess the generalization ability of the simple regression model of the house sales data [VMLS, §2.3]
- Current `RMS` error \$74.8k; cross-validation will help us answer the question of how the model might do on different, unseen houses.

- randomly partition the data set of 774 sales records into five folds, four of size 155 and one of size 154.



Limitations

Performative Prediction

- **Caution**: the basic assumption that the test data and future data are similar can (and does) fail in some applications
- e.g.: A model that predicts consumer demand, trained and validated on this year's data, can make much poorer predictions next year, simply because consumer tastes shift.
- e.g.: In finance, patterns of asset returns periodically shift, so models that predict well on test data from this year need not predict well next year.
- **Caution**: the data set is small $\Longrightarrow$ harder to interpret out-of-sample and cross-validation results.
- e.g.: out-of-sample test $\texttt{RMS}$ error might be small due to good luck, or large due to bad luck, in the selection of the test set.

Feature Engineering

Feature Engineering

- (cf. [VMLS, §13.1.2]) Fitting a linear parametric model reduces to regression with new features which are the original features x mapped through the basis (or feature mapping) functions $f_1, \ldots, f_p$.
- **Feature engineering/selection**: the process of choosing or selecting the feature maps
- To choose among candidate basis functions, we use out-of-sample validation or cross-validation.
- Heuristics:
  - Include the constant basis function $f_1(x) = 1$ (equiv. offset in basic regression model)
  - Include the original features $f_i(x) = x_{i-1}$, $i = 2, \ldots, n+1$ (equiv. starting with the basic regression model)
- **Adding more features vs. reducing them**: we can both add more features (i.e., $p > n$) to get a richer model (and select amongst them via cross-validation) or combine or reduce the number of features to reduce the dimension (i.e., $p < n$).

Transforming Features: Standardizing Features

- The first thing to consider is simple transformations of features.
- **Standardization**: scale and offset

$$f_i(x) = \frac{(x_i - b)}{a_i}, \quad i = 2, \ldots, n+1$$

  - $b_i = \texttt{avg}(x)$
  - $a_i = \texttt{std}(x)$
- ensures that the average of $f_i(x)$ is zero and the standard deviation is near one.
- This is called *standardizing* or *z-scoring*

Transforming Features: Windsorizing Features

- When the data include some very large values that are thought to be errors (say, in collecting the data), it is common to **clip** or **winsorize** (for Charles P. Windsor) the data
- e.g.,

$$f_i(x) = \begin{cases} x, & |x| \leq 3 \\ 3, & x > 3 \\ -3, & x < -3 \end{cases}$$

Transforming Features: Log Transform

- When feature values are positive and vary over a wide range, it is common to replace them with their logarithms.
- If the feature value also includes the value zero, then a common trick is to replace the log transform with $f_i(x) = \log(x+1)$
- **Effect**: compresses the range of values that we encounter.

Creating New Features: Expanding Categoricals

- Some features take on only a few values, such as -1 and 1 or 0 and 1, which might represent some value like presence or absence of some symptom (e.g., Boolean features).
  - Likert scale ([VMLS, page 71]) takes small number of values $-2, -1, 0, 1, 2$
  - Days of the week $0, 1, \ldots, 6$
- **Expanding Categoricals**: replace a feature with ℓ values with a set of $\ell - 1$ new features each of which is Boolean. This simply records whether or not the original feature has the associated value.
- If all values are zero across the $\ell - 1$ new features, then the original feature had the default value
- **Example:** suppose $x_1 \in \{-1, 0, 1\}$. Using the feature value 0 as the default feature, we replace x_1 with two mapped features

$$f_1(x) = \begin{cases} 1, & x_1 = -1 \\ 0, & \text{otherwise} \end{cases} \qquad f_2(x) = \begin{cases} 1, & x_1 = 1 \\ 0, & \text{otherwise} \end{cases}$$

Creating New Features: Generalized Additive model

- New features that are nonlinear functions of the original features—e.g.,

$$\min\{x_i + a, 0\} \quad \text{and} \quad \max\{x_i - b, 0\} \quad \text{where } a, b \text{ are parameters}$$

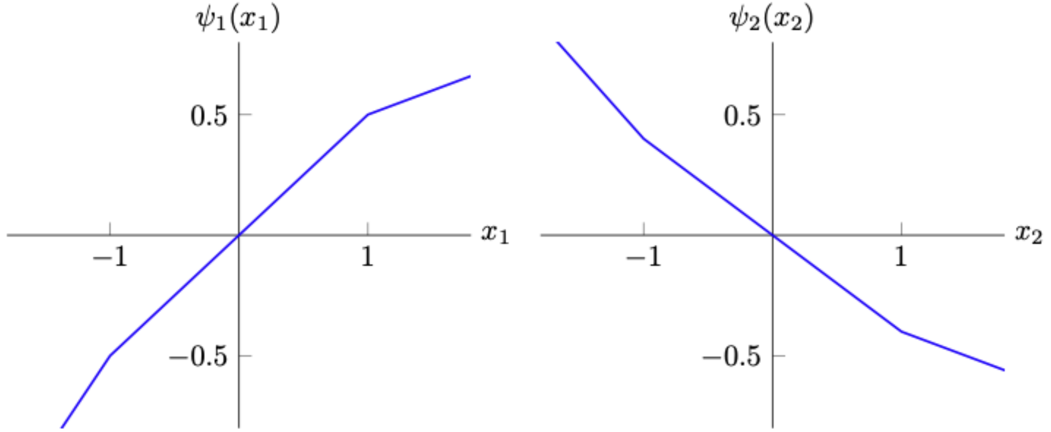
- $\min\{x_i + a, 0\}$: amount by which feature x_i is below $-a$, and
- $\max\{x_i - b, 0\}$: amount by which feature x_i is above b
- **Generalized additive model**: if the data is standardized set $a = b = 1$ so that

$$\hat{y} = \psi_1(x_1) + \cdots + \psi_n(x_n)$$

where

$$\psi_i(x_i) = \theta_{n+i} \min\{x_i + a, 0\} + \theta_i x_i + \theta_{2n+i} \max\{x_i - b, 0\}$$

Creating New Features: Generalized Additive model



- Example: $n = 2$ original features. Prediction $\hat{y}$ is a sum of two piecewise-linear functions, each depending on one of the original features.
- $n = 2$, $a = b = 1$, and $\theta_1 = 0.5$, $\theta_2 = -0.4$, $\theta_3 = 0.3$, $\theta_4 = -0.2$, $\theta_5 = -0.3$, $\theta_6 = 0.2$.

Creating New Features: Products and Interactions

- New features can be developed from pairs of original features, for example, their product.
- e.g., $x_i x_j$ for $i, j = 1, \ldots, n$ and $i \leq j$
- Product features are easily interpretable when the original features are Boolean (take the values 0 or 1)—i.e, $x_i = 1$ means that feature i is present and the new product has the value one exactly when both features i and j have occurred.

Creating New Features: Stratefied Models

- **Stratefied Model**: several different sub-models, and choose the one to use depending on the values of the regressors.
- e.g., instead of treating gender as a regressor in a single model of some medical outcome, we build two different sub-models, one for male patients and one for female patients.
- More generally, we can carry out clustering of the original feature vectors, and fit a separate model within each cluster.
- To evaluate $\hat{y}$ for a new x, we first determine which cluster x is in, and then use the associated model.
- Whether or not a stratified model is a good idea is checked using out-of-sample validation.

Advanced Feature Generation

- Custom mappings: driven by application
- Predictions from other models: if there are existing models for the data we can exploit those
- Distance to cluster representatives $z_1, \ldots, z_k$: $f_i(x) = e^{-\|x-z_i\|^2/\sigma^2}$
- Random features: random linear combination of original features
- Neural network features: computes transformed features using compositions of linear transformations interspersed with nonlinear mappings such as the absolute value.

Summary of Heuristics/Rule-of-Thumb

- **Try simple models first.** Start with a constant, then a simple regression model, and so on. You can compare more sophisticated models against these.
- **Compare competing candidate models using validation.** Adding new features will always reduce the RMS error on the training data, but the important question is whether or not it substantially reduces the RMS error on the test or validation data sets.
- **Adding new features can easily lead to over-fit.** The most straightforward way to avoid over-fit is to keep the model simple. We mention here that another approach to avoiding over-fit, called *regularization* [VMLS, ch. 15]

Example: House Price Prediction

see `Mod2-Lec2.ipynb`