Week 9: Linear Regression

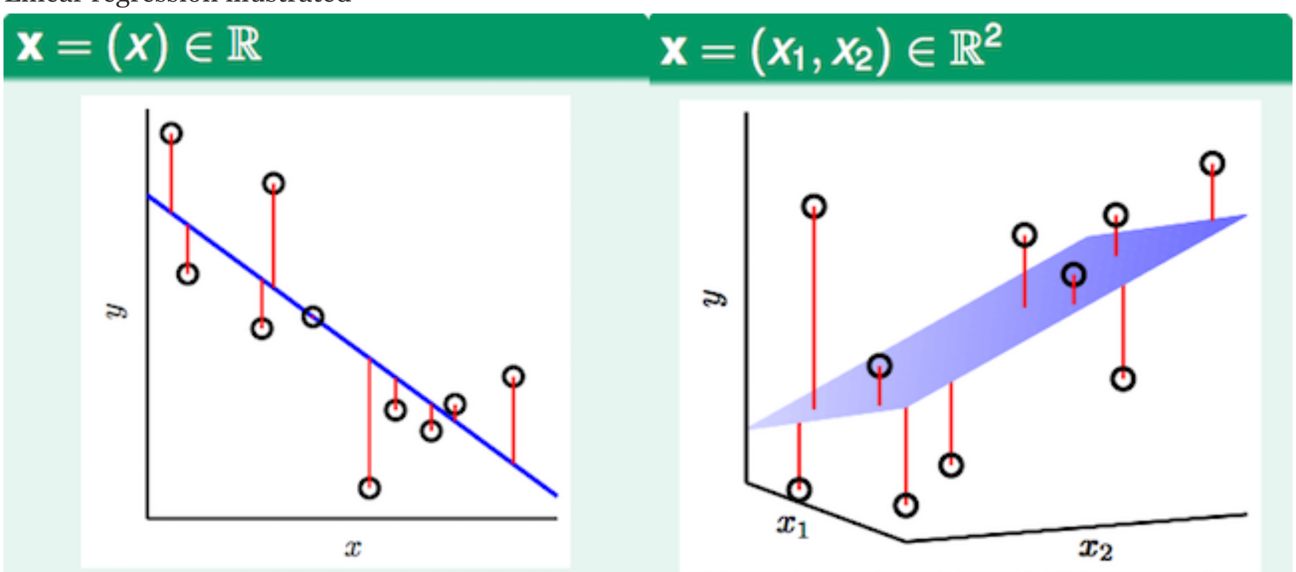
Table of Contents

Linear Regression Problem

1. For features $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_d)$, approximate the target y with a **weighted sum**

$$y \approx \sum_{i=0}^{d} w_i x_i$$

2. Linear regression hypothesis: $h(x) = \mathbf{w}^T \mathbf{x}$. Similar to perceptron, but taking the *real number* value instead of just the sign.
3. Linear regression illustrated



The goal of linear regression is to find the best-fitting *line/hyperplane* with small *residuals*

4. Linear regression error measure
    - Squared error $err(\hat{y} - y)^2$ is often used as the error measure for linear regression

in-sample	out-of-sample
$E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n)^2$	$E_{out}(\mathbf{w}) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}}(\mathbf{w}^T\mathbf{x} - y)^2$

Linear Regression Algorithm

1. Cost function of linear regression

$$E_{in}(w) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n) = \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2$$

   Goal: Find **w** that minimizes cost funciton / in-sample error

2. Matrix form of linear regression in-sample error

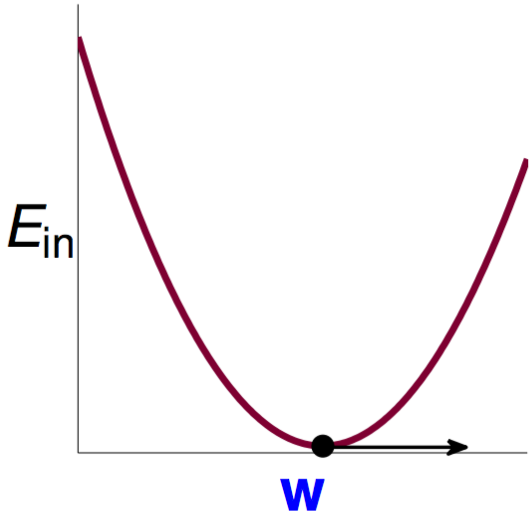
$$\begin{aligned}
E_{in}(w) &= \frac{1}{N}\sum_{n=1}^{N}(w^T x_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n^T w - y_n)^2 \\
&= \frac{1}{N}\left\| \begin{matrix} x_1^T w - y_1 \\ x_2^T w - y_2 \\ \cdots \\ x_N^T w - y_N \end{matrix} \right\|^2 \\
&= \frac{1}{N}\left\| \begin{bmatrix} --x_1^T-- \\ --x_2^T-- \\ \cdots \\ --x_N^T-- \end{bmatrix} w - \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_3 \end{bmatrix} \right\|^2 \\
&= \frac{1}{N}\| \underbrace{X}_{N\times d+1} \underbrace{w}_{d+1\times 1} - \underbrace{y}_{N\times 1} \|^2
\end{aligned}$$

Optimizing In-Sample Error

1. $E_{in}(w)$ is a convex function

$$\min_{w} E_{in}(w) = \frac{1}{N}\|Xw - y\|^2$$

   - X and y come from the training dample $\mathcal{D}$, therefore E_{in} is only a function of w
   - $E_{in}(w)$ is **continuous, differentiable, and convex**, which are the necessary conditions for minimizing E_{in} w.r.t. w



2. To minimize E_{in}, find w that gives gradien of 0.

$$E_{in}(w) \equiv \begin{bmatrix} \frac{\partial E_{in}}{\partial w_0}(w) \\ \frac{\partial E_{in}}{\partial w_1}(w) \\ \dots \\ \frac{\partial E_{in}}{\partial w_d}(w) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

3. The gradient $\nabla E_{in}(w)$

$$\begin{aligned} E_{in}(w) &= \frac{1}{N}\|Xw - y\|^2 \\ &= \frac{1}{N}(w^T \underbrace{X^T X}_{A} w \; - \; 2w^T \underbrace{X^T y}_{b} \; + \; \underbrace{y^T y}_{c}) \quad \text{polynomial expansion} \end{aligned}$$

Substitute and take partial derivative:

Single w		Vector w
a, b, c are constants	Generalizes to $\rightarrow$	A, $\mathbf{b}$ are vectors, $c$is a constant
$E_{in}(w) = \frac{1}{N}(aw^2 - 2bw + c)$		$E_{in}(w) = \frac{1}{N}(w^T A w - 2w^T \mathbf{b} + c)$
$\nabla E_{in}(w) = \frac{1}{N}(2aw - 2b)$		$\nabla E_{in}(w) = \frac{1}{N}(2Aw - 2\mathbf{b})$

Substitute again and simplify. The following applies to both 1D and multi-dimentional cases:

$$\begin{aligned} \nabla E_{in}(w) &= \nabla \frac{1}{N}(w^T X^T X w - 2w^T X^T y + y^T y) \\ &= \frac{2}{N}(X^T X w - X^T y) \end{aligned}$$

4. Optimal linear regression weights
    - Task: Find w_{LIN} such that $\frac{2}{N}(X^TXw - X^Ty) = \nabla E_{in}(w) = 0$
    - When X^TX is **invertible**
        - **Unique** solution
        - X^TX is often invertible
            - X has dimension $N \times (d+1) \to X^TX$ has dimension $(d+1) \times (d+1)$
                - N being number of training samples, d being the number of variables in the model, or degrees of freedom, or VC dimension
            - $N >> d$ most of the time, so there's likely enough 0s in the resulting matrix for X^TX to be invertible

$$w_{LIN} = \underbrace{(X^TX)^{-1}X^T}_{\text{pseudo-inverse } X^\dagger}\mathbf{y}$$

    - When X^TX is **singular**
        - **many** optimal solution, one of them being

$$w_{LIN} = X^\dagger \mathbf{y} \quad \text{with different definition for } X^\dagger$$

    - Practical suggestion
        - Use **well-implemented, existing** routine to obtain $X^\dagger$ directly, instead of calculating $(X^TX)^-1X^T$ on a case-by-case basis
        - Helps with cases where X^TX is *almost* singular, as such edge cases are already taken care of by built-in routine

5. The linear regression algorithm, in d-dimensions
    - From $\mathcal{D}$, construct input matrix X and output vector $\mathbf{y}$ as:

$$X = \underbrace{\begin{bmatrix} --x_1^T-- \\ --x_2^T-- \\ \dots \\ --x_N^T-- \end{bmatrix}}_{N\times(d+1)} \qquad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_3 \end{bmatrix}}_{N\times 1}$$

    - Calculate *pseudo-inverse* $\underbrace{X^\dagger}_{(d+1)\times N}$
    - Return $\underbrace{\mathbf{w}_{LIN}}_{(d+1)\times 1} = X^\dagger \mathbf{y}$

Generalization of Lienar Regression

1. Guarantee of linear regression analytic solution: **Average** in-sample error $\overline{E_{in}}$ (across all training samples) is **smaller** than the noise level contained in training data, and decreses as sample size N grows:

$$\overline{E_{in}} = \mathcal{E}_{D\sim P_N}\{E_{in}(w_{LIN} \text{ w.r.t } \mathcal{D})\} = \text{ noise level} \cdot (1 - \frac{d+1}{N})$$

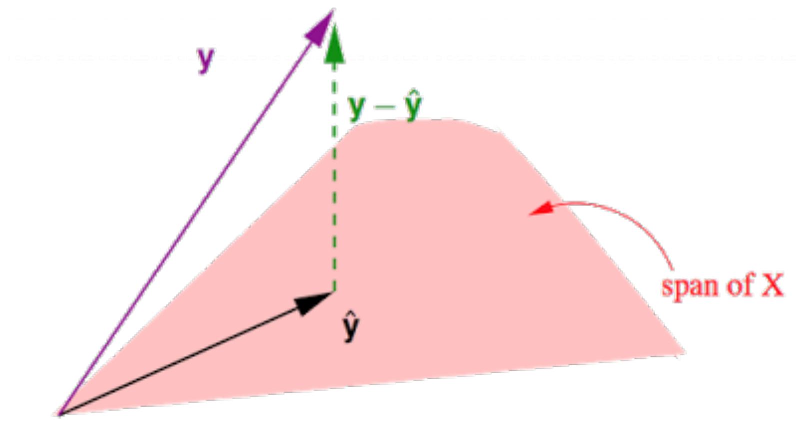
The Hat Matrix

1. Summary: Given optimal w_{LIN} , in-sample error of linear regression can be represented as:

$$\begin{aligned} E_{in}(w_{LIN}) &= \frac{1}{N}\|y - \hat{y}\|^2 \\ &= \frac{1}{N}\|y - X\underbrace{X^{\dagger}y}_{w_{LIN}}\|^2 \\ &= \frac{1}{N}\|(\underbrace{I}_{identity} - XX^{\dagger})y\|^2 \end{aligned}$$

   - $XX^{\dagger}$ is known as **hat matrix** $H = X(X^TX)^{-1}X^T$

2. Geometric view of hat matrix



In n-dimensional $\mathbb{R}^N$:

   - X matrix can be viewed as a hyperplane (red area)
   - Geometrically, the smallest possible residual $y - \hat{y}$ should be **perpendicular** to the X hyperplane
   - H creates $\hat{y}$, the projection of y onto X hyperplane
     - For smallest residual, let

$$\begin{aligned} Hy &= \hat{y} \\ y - Hy &= y - \hat{y} \\ (I - H)y &= y - \hat{y} \end{aligned}$$

In other words, $I - H$ creates a *perpendicular* projection of y onto X hyperplane

3. Properties of H

   - Symmetric

$$\begin{aligned} H^T &= (X(X^TX)^{-1}X^T)^T \\ &= X((X^TX)^{-1})^TX^T \\ &= X(X^TX)^{-1}X^T \\ &= H \end{aligned}$$

   - Idempotent

$$
\begin{aligned}
H^2 &= (X(X^TX)^{-1}X^T)(X(X^TX)^{-1}X^T) \\
&= X\underbrace{(X^TX)^{-1}(X^TX)}_{I}(X^TX)^{-1}X^T \\
&= X(X^TX)^{-1}X^T \\
&= H
\end{aligned}
$$

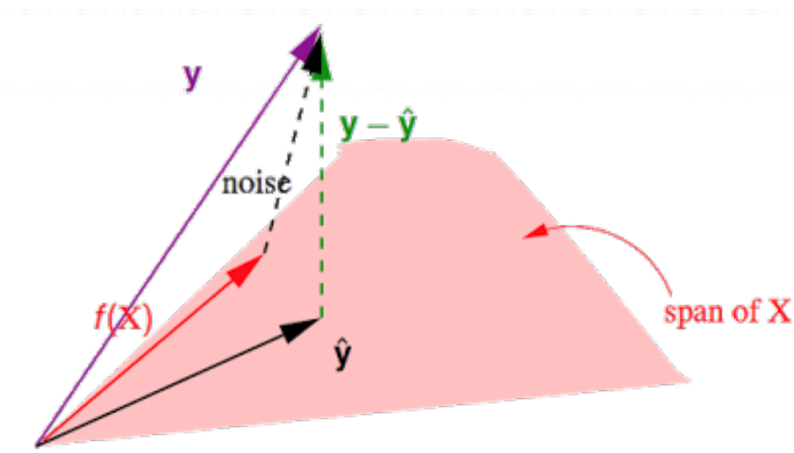
- Positive semi-definite
  - All eigenvalues are non-negative
  - $\lambda = eigenvalues, b = eigenvectors$

$$
\begin{aligned}
Hb &= \lambda b \\
H^2b &= \lambda Hb = \lambda(\lambda b) \\
\because H &= H^2 \\
H^2b &= Hb = \lambda b \\
\therefore \lambda^2 b &= \lambda b \\
\lambda(\lambda - 1)b &= 0 \\
\lambda = 0 &\text{ or } \lambda = 1
\end{aligned}
$$

- Trace of H
  - *Trace* of a matrix is the sum of all its eigenvalues

$$trace(I - H) = N - (d + 1)$$

4. Hat matrix when y contains noise



- Assume training input y comes from some ideal target function $f(X) \in span + noise$

$$
\begin{aligned}
y &= f(X) + noise \\
(I - H)noise &= y - \hat{y}
\end{aligned}
$$

- Substituting into definition of E_{in}

$$
\begin{aligned}
E_{in}(w_{LIN}) &= \frac{1}{N}\|y - \hat{y}\|^2 \\
&= \frac{1}{N}\|(I - H)noise\|^2 \\
&= \frac{1}{N} trace(I - H)\|noise\|^2 \\
&= \frac{1}{N}(N - (d+1))\|noise\|^2
\end{aligned}
$$

- Averaging across all possible training samples of size N from the population results in the analytical guarantee of linear regression

$$
\begin{aligned}
\overline{E_{in}} &= \text{noise level} \cdot \left(1 - \frac{d+1}{N}\right) \\
\overline{E_{out}} &= \text{noise level} \cdot \left(1 + \frac{d+1}{N}\right)
\end{aligned}
$$

5. The learning curve

- Both in-sample and out-of-sample errors converge to noise leve σ^2
- Generalization error $E_{out} - E_{in}$ is bounded
  - With respect to the same ideal target function + noise
  - The bounded difference can be expressed as function of VC dimension d and sample size N, $\frac{2(d+1)}{N}$