Linear Regression

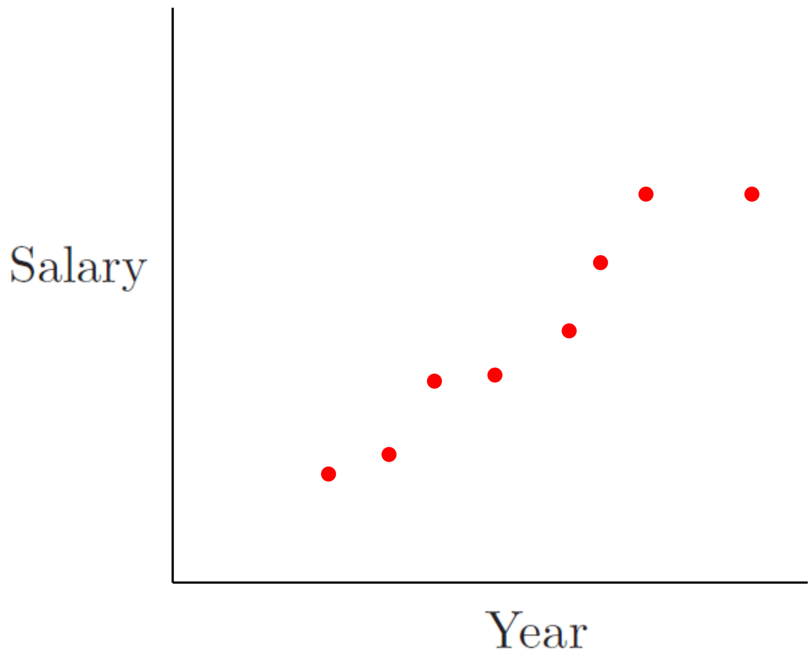
Dripta Mj
Department of Mathematics
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, India

Machine Learning
CS230

Sem 3, 2018-19

Introduction

- Variation of salary with time:



Notation

- Given a data set of N points.

- Input data comprise D features, suppose they are $x_1, x_2, ..., x_D$.

- Representation of the ith input data point:

$$\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, ..., x_D^{(i)}]^{\mathrm{T}}$$

- Set of input data points:

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ...\mathbf{x}^{(N)}\}$$

- Set of outputs for the N data points:

$$\mathbf{y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$$

Linear Regression

- Input variables can be written in vectorial form as:

$$\mathbf{x} = [x_1, x_2, ..., x_D]^T$$

- Prediction: Linear combination of input variables

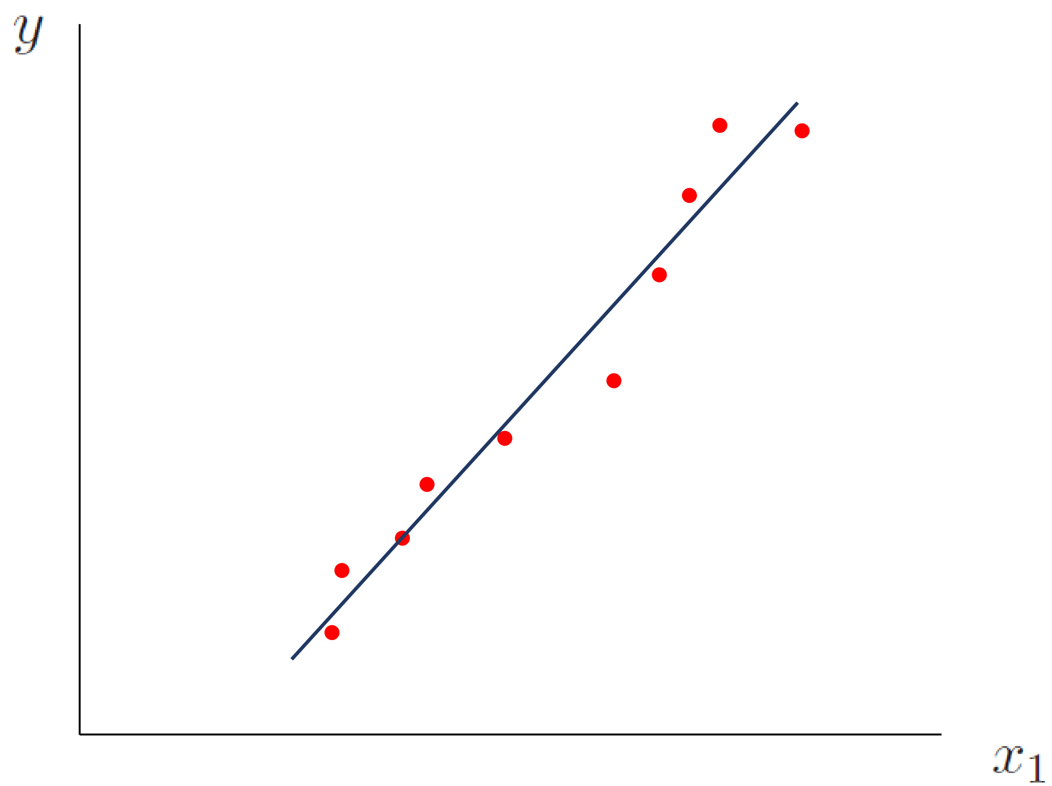
$$z(\mathbf{x}, p_0, p_1, ..., p_D) = p_0 + p_1 x_1 + p_2 x_2 + ... + p_D x_D$$

- Vectorial representation of weights:

$$\mathbf{p} = [p_0, p_1, p_2, ..., p_D]^T$$

Simplest case

- One input variable: x_1



$$\boxed{z = p_0 + p_1 x_1}$$

Using basis functions

- Output: Linear combination of functions of input variables

$$z(\mathbf{x}, p) = p_0 + \sum_{i=1}^{Q} p_i f_i(\mathbf{x})$$

where $f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_Q(\mathbf{x})$ are the basis functions.

- Taking $f_0 = 1,$ can write

$$z(\mathbf{x}, p) = \sum_{i=0}^{Q} p_i f_i(\mathbf{x})$$

- Vectorial representation:

$$z(\mathbf{x}, \mathbf{p}) = \mathbf{p}^T \mathbf{f}(\mathbf{x})$$

where $\mathbf{p} = [p_0, p_1, ..., p_Q]^{\mathrm{T}}$ and $\mathbf{f}(\mathbf{x}) = [f_0(\mathbf{x}), f_1(\mathbf{x}), ..., f_Q(\mathbf{x})]^{\mathrm{T}}$

Basis functions

- Polynomial basis functions (with single input x_1) have the form:

$$f_0(x_1) = 1, \qquad f_1(x_1) = x_1, \qquad f_2(x_1) = x_1^2, \quad \ldots. \quad f_Q(x_1) = x_1^Q,$$

  and therefore

$$\mathbf{f}(x_1) = \left[1,\ x_1,\ x_1^2, \ldots,\ x_1^Q\right]^{\mathrm{T}}.$$

- More complex functions can be generated by increasing the degree of the polynomial Q.

- Some other popular choices of basis functions:
  - Spline functions (piecewise polynomial curves)
  - Gaussian basis functions

Error

- Dataset comprise N data points.
- Outputs at the training locations: $\mathbf{y} = \left\{y^{(1)}, y^{(2)}, ..., y^{(N)}\right\}$.
- Predictions at the training locations: $\mathbf{z} = \left\{z^{(1)}, z^{(2)}, ..., z^{(N)}\right\}$.
- Sum of squared errors:

$$\begin{aligned} \mathrm{SE} &= \sum_{n=1}^{N} \left(y^{(n)} - z^{(n)}\right)^2 \\ &= \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)})\right)^2 \end{aligned}$$

- Mean squared error $\mathrm{MSE} = \dfrac{\mathrm{SE}}{N}$
- Often the error function is written in the following form that is more amenable to differentiation

$$E = \frac{1}{2} \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)})\right)^2$$

Parameter values

- Optimal weights – Setting the gradient of $E(\mathbf{p})$ to zero:

$$\nabla E(\mathbf{p}) = 0$$

which yields

$$\sum_{n=1}^{N} y^{(n)} \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} - \mathbf{p}^{\mathrm{T}} \left(\sum_{n=1}^{N} \mathbf{f}(\mathbf{x}^{(n)}) \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} \right) = 0$$

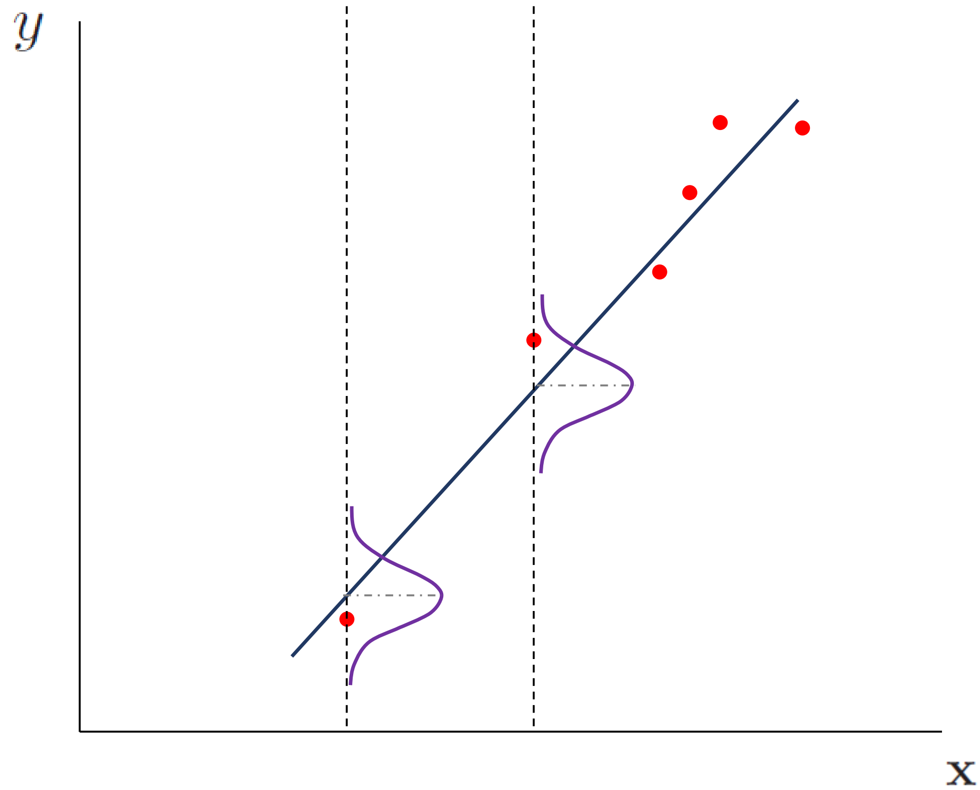
Let

$$\mathcal{F} = \begin{bmatrix} \mathbf{f}(\mathbf{x}^{(1)})^{\mathrm{T}} \\ \mathbf{f}(\mathbf{x}^{(2)})^{\mathrm{T}} \\ \cdot \\ \cdot \\ \mathbf{f}(\mathbf{x}^{(N)})^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} f_0(\mathbf{x}^{(1)}) & f_1(\mathbf{x}^{(1)}) & \cdots & f_Q(\mathbf{x}^{(1)}) \\ f_0(\mathbf{x}^{(2)}) & f_1(\mathbf{x}^{(2)}) & \cdots & f_Q(\mathbf{x}^{(2)}) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ f_0(\mathbf{x}^{(N)}) & f_1(\mathbf{x}^{(N)}) & \cdots & f_Q(\mathbf{x}^{(N)}) \end{bmatrix}$$

- Finally, the value of the weights $\mathbf{p}$ are obtained as:

$$\boxed{\mathbf{p} = (\mathcal{F}^{\mathrm{T}} \mathcal{F})^{-1} \mathcal{F}^{\mathrm{T}} \mathbf{y}}$$

Probabilistic approach



- Assumption: Gaussian noise model

$$y^{(n)} \sim \mathcal{N}(\mathbf{p}^{\mathrm{T}}\mathbf{x}^{(n)}, \sigma^2)$$

Maximum likelihood estimator

- Training data comprise N data points:

$$\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}, \qquad \mathbf{y} = \{y^{(1)}, y^{(2)}, ..., y^{(N)}\}$$

- Likelihood function

$$p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(y^{(n)}|\mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)}), \sigma^2)$$

- Taking log on both sides:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) &= \sum_{n=1}^{N} \ln \mathcal{N}(y^{(n)}|\mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)}), \sigma^2) \\ &= -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left(y^{(n)} - \mathbf{p}^T \mathbf{f}(\mathbf{x}^{(n)})\right)^2\end{aligned}$$

- For a linear model with conditional Gaussian noise distribution, maximizing likelihood with respect to weights is equivalent to minimizing the sum-of-squared error.

Maximum likelihood estimator

- Weights for maximum likelihood – setting the gradient of log likelihood to zero:

$$\nabla \ln p(\mathbf{y}|\mathbf{X}, \mathbf{p}, \sigma^2) = \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(n)})\right) \mathbf{f}(\mathbf{x}^{(n)})^{\mathrm{T}} = 0$$

finally yields

$$\boxed{\mathbf{p} = (\mathcal{F}^{\mathrm{T}} \mathcal{F})^{-1} \mathcal{F}^{\mathrm{T}} \mathbf{y}}$$

- The noise variance σ^2 can also determined by maximizing (log) likelihood with respect to σ^2, yielding:

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^{N} \left(y^{(n)} - \mathbf{p}^{\mathrm{T}} \mathbf{f}(\mathbf{x}^{(n)})\right)^2$$

- This is the residual variance of the outputs around the predictions

Least mean squares algorithm

- Useful for large datasets, real-time applications.
- Predictions can be made before the whole dataset is seen.
- Updates of $\mathbf{p}$ are made using the data points in the training set, separately.
- The objective function to be minimized:

$$
\begin{aligned}
E(\mathbf{p}) &= \sum_{n=1}^{N} \frac{1}{2} \left(y^{(n)} - \mathbf{p}^T f(\mathbf{x}^{(n)}) \right)^2 \\
&= \sum_{n=1}^{N} E_n(\mathbf{p})
\end{aligned}
$$

Stochastic gradient descent

- Initialize the learning rate ζ and $\mathbf{p}$ (say $\mathbf{p}^{(0)}$).

- The weight vector $\mathbf{p}^{(\tau+1)}$ at the τ-th iteration can be updated using the stochastic gradient descent algorithm:

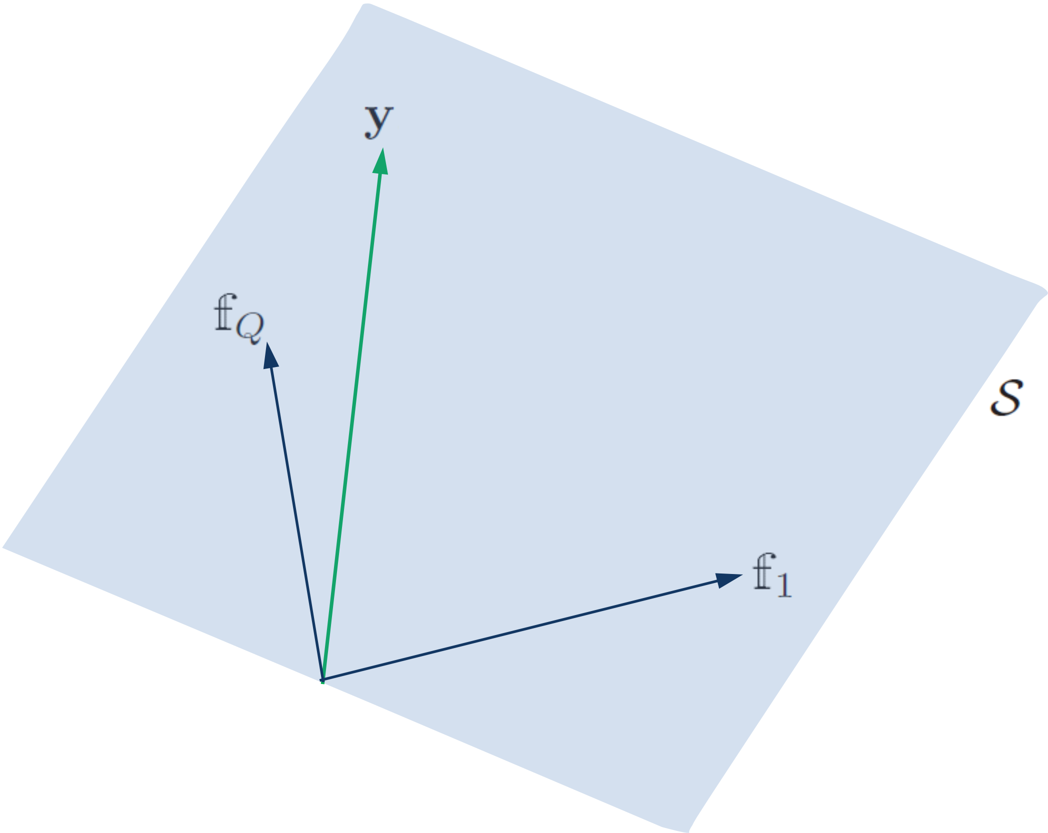
$$\mathbf{p}^{(\tau+1)} = \mathbf{p}^{(\tau)} - \zeta \nabla E_n(\mathbf{p})$$

  – On substitution of $\nabla E_n(\mathbf{p})$ we have

$$\mathbf{p}^{(\tau+1)} = \mathbf{p}^{(\tau)} + \zeta\Big(y^{(n)} - (\mathbf{p}^{(\tau)})^{\mathrm{T}} f(\mathbf{x}^{(n)})\Big) f(\mathbf{x}^{(n)})^{\mathrm{T}}$$

Geometrical interpretation

- Consider an N-dimensional space, whose axes indicate the N output values of the training dataset.
- Let $\mathbb{f}_j$ be the j-th column of $\mathcal{F}$, i.e. $\mathbb{f}_j = [f_j(\mathbf{x}^{(1)}), f_j(\mathbf{x}^{(2)}), .., f_j(\mathbf{x}^{(N)})]^{\mathrm{T}}$.
- In the N-dimensional space consider a subspace $\mathcal{S}$ spanned by Q basis vectors $\{\mathbb{f}_1, \mathbb{f}_2, .., \mathbb{f}_Q\}$.



Geometrical interpretation

- Let $\mathbf{z}$ be an arbitrary linear combination of the basis vectors.
- The vector $\mathbf{z}$ can lie anywhere in the subspace $\mathcal{S}$.
- The least squares algorithm yields solution $\mathbf{z}$ lying on the subspace $\mathcal{S}$ which is closest to the vector $\mathbf{y}$.