BLG 527E Machine Learning

FALL 2021-2022

Assoc. Prof. Yusuf Yaslan & Assist. Prof. Ayse Tosun

Linear Regression

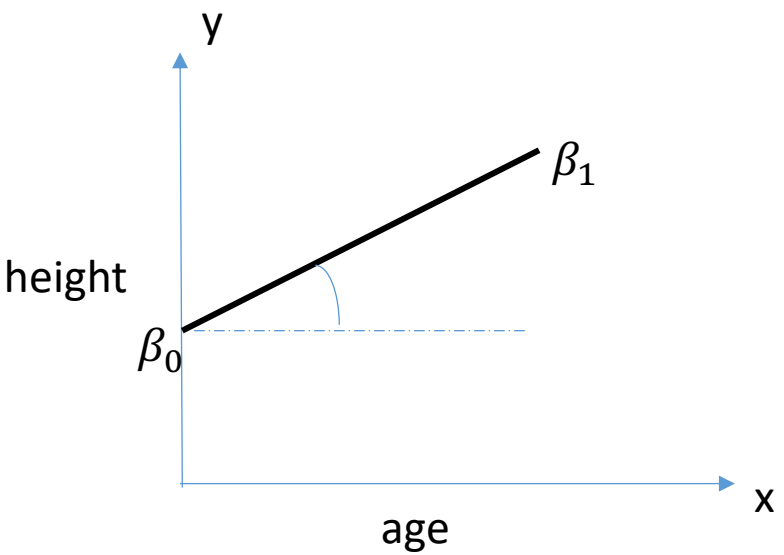
Simple Linear Regression

- $\hat{y} = f(x) = \beta_0 + \beta_1 x$
- Training $\{(x_i, y_i)\}_{i=1}^{N}$
- Choose/Find β_0, β_1 that fits the model
- How to formulate the problem?
  - Loss / Cost function
- $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$

square deviation

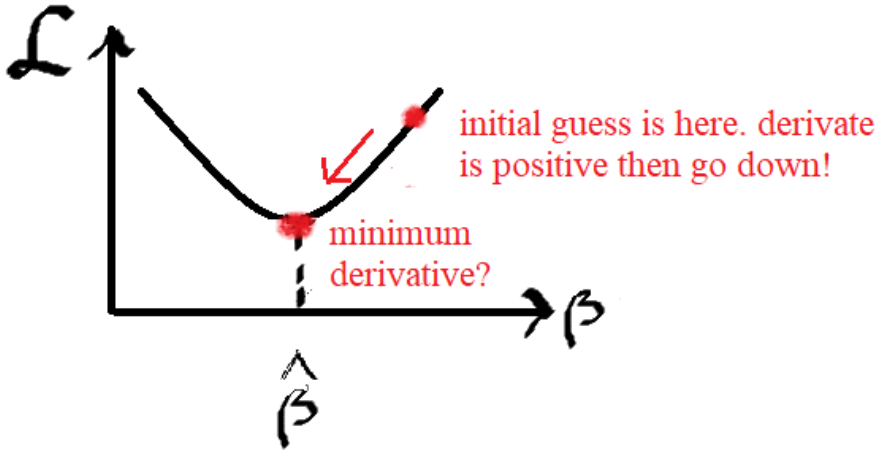
Mean Squared Error (MSE)

- Question: Why to use MSE for linear regression?



‘Baby’ linear regression

- $\hat{y} = f(x) = \beta x$
- $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta x_i)^2$
- ‘baby’ gradient descent
- $\beta \leftarrow \beta - \eta \frac{d\mathcal{L}}{d\beta}$ where η is the learning rate

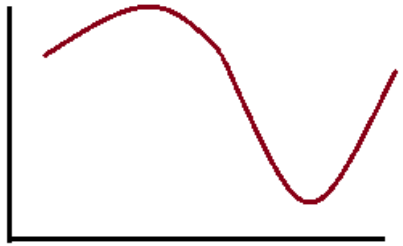


- If learning rate is too large, you keep jumping from one side to the other

  Else (too small), slow convergence

Depending on where you start,

you may stuck in local minimum. For non-convex problem function, gradient descent does**not** work.



When to stop

- Stopping criterion for lowering the loss function
  - If the change in loss is too small
  - If the change is smaller than a value
  - # iterations

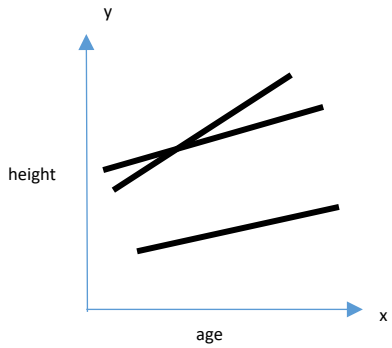
Solution for baby linear regression

- $\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta x_i)^2$

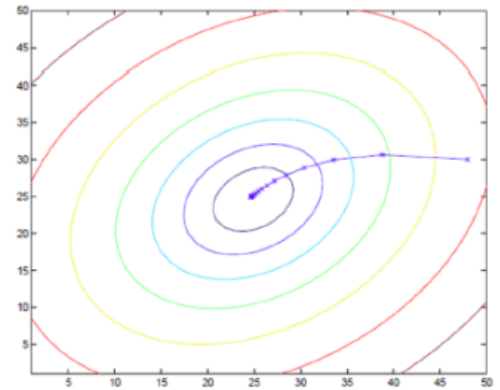
  $= (y_i - \beta x_i)(y_i - \beta x_i)$

  $= y_i y_i - \beta x_i\, y_i - y_i \beta x_i + (\beta x_i)^2$
- $\frac{d\mathcal{L}(\beta)}{d\beta} = \frac{-2}{N}\sum_{i=1}^{N} x_i(y_i - \beta x_i)$
- Find β that minimizes the loss function
- $\hat{\beta} = \frac{\sum_i x_i y_i}{\sum_i {x_i}^2}$

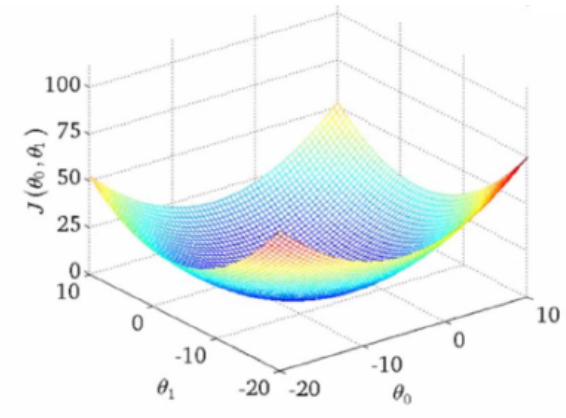
Linear regression



- $\mathcal{L}(\beta_0, \beta_1) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \beta_0 - \beta_1 x_i)^2$
- Partial derivatives
  - Update rules for each parameter
    - $\beta_0 \leftarrow \beta_0 - \eta \frac{d\mathcal{L}}{d\beta_0}$
    - $\beta_1 \leftarrow \beta_1 - \eta \frac{d\mathcal{L}}{d\beta_1}$
  - In vector form
    - $\vec{\beta} \leftarrow \vec{\beta} - \eta \nabla \mathcal{L}$
- $\frac{d\mathcal{L}(\beta_0)}{d\beta_0} = -\frac{2}{N}\sum(y_i - \beta_0 - \beta_1 x_i)$
- $\frac{d\mathcal{L}(\beta_1)}{d\beta_1} = -\frac{2}{N}\sum(y_i - \beta_0 - \beta_1 x_i)x_i$
- Compute $\frac{d\mathcal{L}(\beta_j)}{d\beta_j}$=0



contours of the quadratic function



Figures from Lecture: Gradient Descent and Logistic Regression, Aalto University, School of Electrical Engineering

Multiple Linear Regression

- $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ \ldots + \beta_p x_p$
  - Total number of predictors:p
  - It is convenient to use $x_0 = 1$, then $f(x) = \vec{\beta}\vec{x}$ (scalar product)
- $f(x) = \beta^T X = (\beta_0 \ldots . \beta_p)\begin{pmatrix} x_0 \\ \cdots \\ x_p \end{pmatrix}$
- $\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \beta^T x^{(i)})^2$ where i indicates ith training sample
- Partial derivative wrt $\beta_k = -\frac{2}{N}\sum(y^{(i)} - \beta^T x^{(i)})x_k^{(i)}$

Solve it with matrix notation

- X = $\begin{pmatrix} x^1 \\ x^2 \\ \dots \\ x^n \end{pmatrix} = \begin{pmatrix} x_0^1 & \dots & x_p^1 \\ \dots & \dots & \dots \\ x_0^n & \dots & x_p^n \end{pmatrix}$ FEATURE VECTOR
- y = $\begin{pmatrix} y^1 \\ y^2 \\ \dots \\ y^n \end{pmatrix}$ RESPONSE VECTOR
- Given X and β, how to compute $\hat{y}$?
  - $\hat{y} = \mathrm{X}\,\beta$
  - (nxp) times (px1) = (nx1)

Solution for multilinear regression

- $\mathcal{L}(\beta) = \frac{1}{N}\sum_{i=1}^{N}(y - X\beta)^2 \Rightarrow \frac{1}{N}(y - X\beta)^T(y - X\beta)$
  - Dimension of this equation?
- Gradient $\nabla \mathcal{L}$ = - $\frac{2}{N} X^T(y - X\beta)$
- - $\frac{2}{N} X^T(y - X\beta)$ = 0 → $X^T y = X^T X\beta$
- $\hat{\beta}$= $(X^T X)^{-1} X^T y$

Another solution for multilinear regression

- $\frac{\partial \mathcal{L}(\beta)}{\partial \beta} = \frac{\partial (y - X\beta)^T (y - X\beta)}{\partial \beta} = \frac{\partial (y^T y + \beta^T X^T X\beta - 2y^T X\beta)}{\partial \beta}$
- $\frac{\partial (y^T y + \beta^T X^T X\beta - 2y^T X\beta)}{\partial \beta} = 2X^T X\beta - 2X^T y$
- Find $\beta = (X^T X)^{-1} X^T y$

If $(X^T X)$ is identity matrix

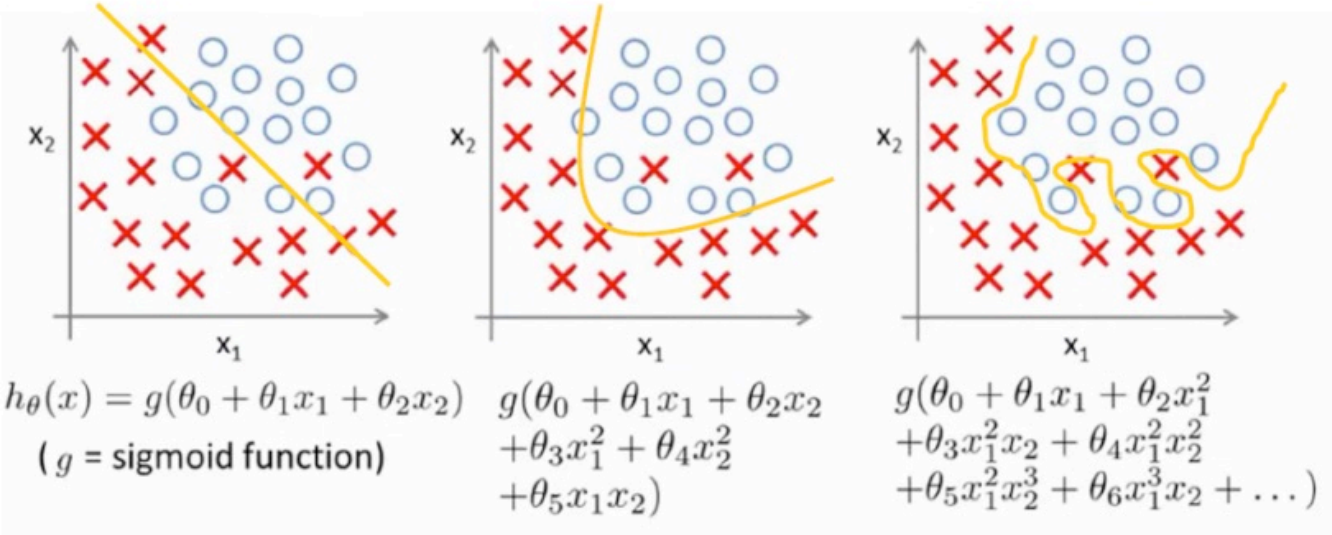
- Simpler solution. How?
- Feature vectors are orthonormal (orthogonal and norm=1)
- If all features have mean 0 and variance of 1, then $\hat{\beta}_i = x_i^T y \ \ (i \neq 0)$
  - Uncorrelated features

Compute y *hat*

- $\hat{\beta} = (X^T X)^{-1} X^T y$
- $\hat{y} = \mathrm{X}\,\beta = X(X^T X)^{-1} X^T y$
-

Overfitting versus Underfitting

- Seems having high order is better fit.
- But
  - Generalization (How well model performs on a new data) error increases
  - Complexity of the model increases
- Reduce the number of features, or apply regularization



Figures from Lecture: Gradient Descent and Logistic Regression, Aalto University, School of Electrical Engineering

Cross Validation

- Tradeoff between complexity of the model, training data size and generalization error.
- To estimate generalization error, data split
  - Training (e.g. 50%)
  - Validation (e.g. 25%)
  - Test (e.g. 25%)

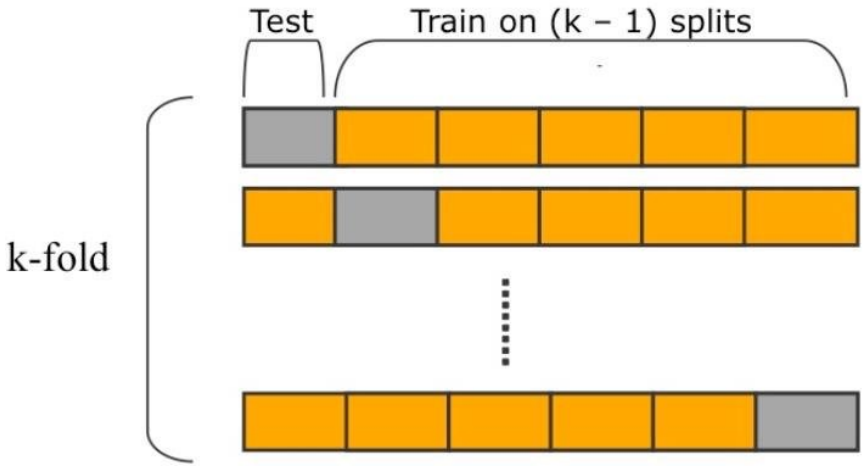


Figure from this paper

- **K-fold cross validation**
  - randomly partition the data in *k* parts or 'folds', set one fold aside for testing,
  - train a model on the remaining *k*-1 folds and evaluate it on the test fold.
  - This process is repeated *k* times until each fold has been used for testing once.