Predictive Modelling

Multiple Linear Regression

Jonathan Mwaura

Khoury College of Computer Science

July 16, 2024

The Roux Institute
at Northeastern University

Introduction

Textbook

Reading: Chapter 3 of: Gareth James et al (2021) . An Introduction to Statistical Learning (2nd Edition) .
`https://www.statlearning.com/`

Acknowledgements

These slides have been adapted from the following Professors:
1) Andrew Ng - Stanford
2) Eric Eaton - UPenn
3) David Sontag - MIT
4) Alina Oprea - Northeastern

Outline

- Multiple linear regression
  - Optimal closed-form solution
- Lab linear regression
- Gradient descent
  - Batch algorithm
  - Algorithm for linear regression
  - Line search

Supervised Learning: Regression

Training

Data → Pre-processing → Feature extraction → Learning model

Data: Labeled, $x_i, y_i \in R$

Pre-processing: Normalization

Feature extraction: Feature Selection

Learning model: Regression, $f(x)$

Testing

New data → Learning model → Predictions $y' = f(x') \in R$

New data: Unlabeled, x'

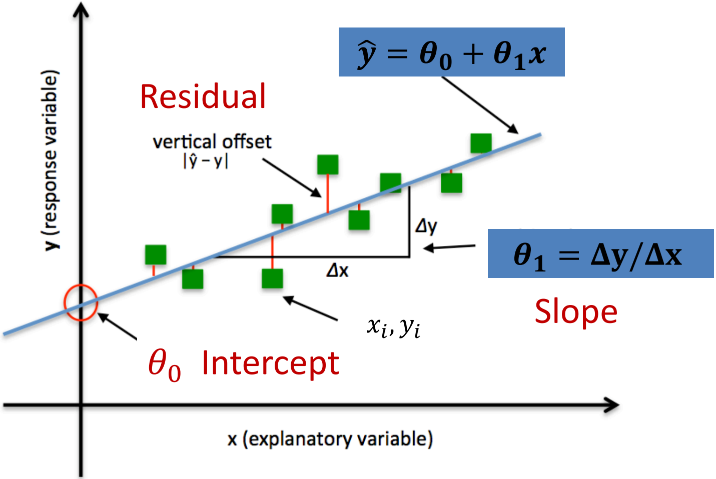
Learning model: $f(x)$

Predictions:

Response variable

Regression

Interpretation



$$h_\theta(x) = \theta_0 + \theta_1 x$$

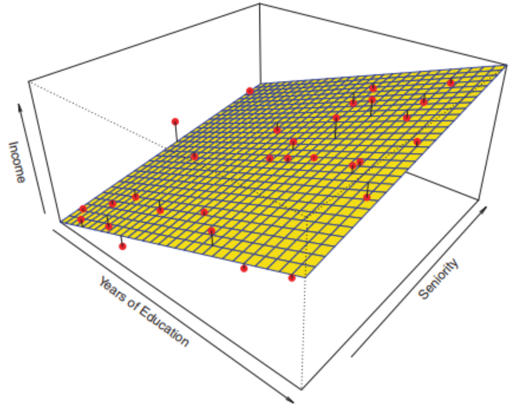
MSE$= \frac{1}{N}\sum_{i=1}^{N}[h_\theta(x_i) - y_i]^2$

Review linear regression

- Simple linear regression: one dimension
- Multiple linear regression: multiple dimensions
- Minimize MSE is equivalent to MLE estimator
  - MSE: average of squared residuals
- Can derive closed-form solution for simple LR
  - $\theta_0 = \bar{y} - \theta_1 \bar{x}$
  - $\theta_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$

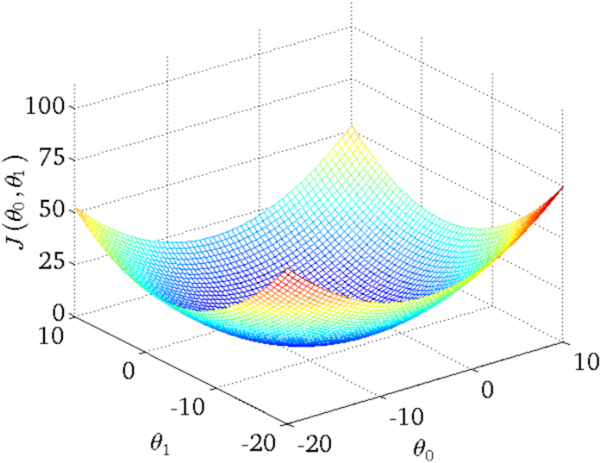
The Roux Institute
at Northeastern University

Multiple Linear Regression



- Linear Regression with 2 predictors
- Dataset: $x_i \in R^d, y_i \in R$

MSE function



Convex function implies unique minimum

Vector Norms

Vector norms: A norm of a vector ||x|| is informally a measure of the "length" of the vector.

$$||x||_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

- Common norms: L_1, L_2 (Euclidean)

$$||x||_1 = \sum_{i=1}^{n} |x_i| \qquad ||x||_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

- L_∞

$$||x||_\infty = \max_i |x_i|$$

Vector products

We will use lower case letters for vectors
The elements are referred by x_i.

- Vector dot (inner) product:

$$x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ x_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i .$$

- Vector outer product:

$$xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

Hypothesis Multiple LR

- Linear Model

- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$

Vector inner product

Training data

$$\boldsymbol{X} = \begin{array}{cc} \begin{array}{cccc} & \text{Feature 1} & & \text{Feature d} \end{array} & \\ \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{id} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix} & \begin{array}{l} \\ \\ \text{Training example i} \\ \\ \\ \end{array} \end{array}$$

- Total number of training example: *N*
- Dimension of training data point (number of features): *d*
- Dimension of matrix: *Nx(d+1)*

Use Vectorization

- Consider our model for n instances:

$$h(x_i) = \sum_{j=0}^{d} \theta_j x_{ij} = \theta^T x_i$$

- Let

Model parameter

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \cdots & x_{id} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}$$

Training data

$\mathbb{R}^{(d+1)\times 1}$ $\qquad$ $\mathbb{R}^{n\times(d+1)}$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

Model prediction vector $\hat{y}$

Loss function MSE

- For the linear regression cost function:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N}[h_\theta(x_i) - y_i]^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}[\hat{y}_i - y_i]^2$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_N \end{bmatrix}$$

$$= \frac{1}{N}||\hat{y} - y||^2$$

$$= \frac{1}{N}||X\theta - y||^2$$

$$\hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ . \\ . \\ . \\ \hat{y}_N \end{bmatrix}$$

Matrix and vector gradients

If $y = f(x), y \in R$ scalar , $x \in R^n$ vector

$$\frac{\partial y}{\partial x} = \left[\frac{\partial y}{\partial x_1} \quad \frac{\partial y}{\partial x_2} \quad \cdots \quad \frac{\partial y}{\partial x_n} \right]$$

Vector gradient (row vector)

If $y = f(x), y \in R^m, x \in R^n$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

Jacobian matrix (Matrix gradient)

Properties

- If w, x are$(d \times 1)$ vectors, $\frac{\partial w^T x}{\partial x} = w^T$
- If A: $(n \times d)$ x: $(d \times 1)$, $\frac{\partial Ax}{\partial x} = A$

- If A: $(d \times d)$ x: $(d \times 1)$, $\frac{\partial x^T Ax}{\partial x} = (A + A^T)x$
- If A symmetric: $\frac{\partial x^T Ax}{\partial x} = 2Ax$
- If x: $(d \times 1)$, $\frac{\partial ||x||^2}{\partial x} = 2x^T$

Min loss function

– Notice that the solution is when $\frac{\partial}{\partial \boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$

$$J(\theta) = \frac{1}{N} ||X\theta - y||^2$$

Using chain rule

$$f(\theta) = h\big(g(\theta)\big), \frac{\partial f(\theta)}{\partial \theta} = \frac{\partial h(g(\theta))}{\partial \theta} \frac{\partial g(\theta)}{\partial \theta}$$

$$h(x) = ||x||^2, g(\theta) = X\theta - y$$

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{2}{N}[(X\,\theta - y)^T X] = 0 \;\Rightarrow X^T\;(X\theta - y) = 0$$

$$(\boldsymbol{X}^\intercal \boldsymbol{X})\boldsymbol{\theta} = \boldsymbol{X}^\intercal \boldsymbol{y}$$

Closed Form Solution: $\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$

Vectorization

- Two options for operations on training data
  - Matrix operations
  - For loops to update individual entries
- Most software packages are highly optimized for matrix operations
  - Python numpy
  - Preferred method!

- Matrix operations are much faster than loops!

Closed-form solution

- Can obtain $\boldsymbol{\theta}$ by simply plugging $\boldsymbol{X}$ and $\boldsymbol{y}$ into

$$\boldsymbol{\theta} = (\boldsymbol{X}^\intercal \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{y}$$

$$X = \begin{bmatrix} 1 & x_{11} & \dots & x_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & \dots & x_{id} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & \dots & x_{Nd} \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_N \end{bmatrix}$$

- If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $d \leq n$

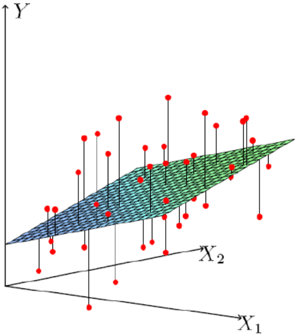
$$AGA = A$$

The Roux Institute at Northeastern University

Multiple Linear Regression

- Dataset: $x_i \in R^d, y_i \in R$
- Hypothesis $h_\theta(x) = \theta^T x$
- MSE $= \frac{1}{N}\sum (\theta^T x_i - y_i)^2$ Loss / cost

$$\theta = (X^\top X)^{-1} X^\top y$$



The Roux Institute at Northeastern University

Practical issues: Feature Standardization

- Rescales features to have zero mean and unit variance
  - Let μ_j be the mean of feature j: $\mu_j = \frac{1}{n}\sum_{i=1}^{n} x_j^{(i)}$
  - Replace each value with:

$$x_j^{(i)} \leftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

for j = 1...d
(not x_0!)

    - s_j is the standard deviation of feature j

- Must apply the same transformation to instances for both training and prediction
- Mean 0 and Standard Deviation 1

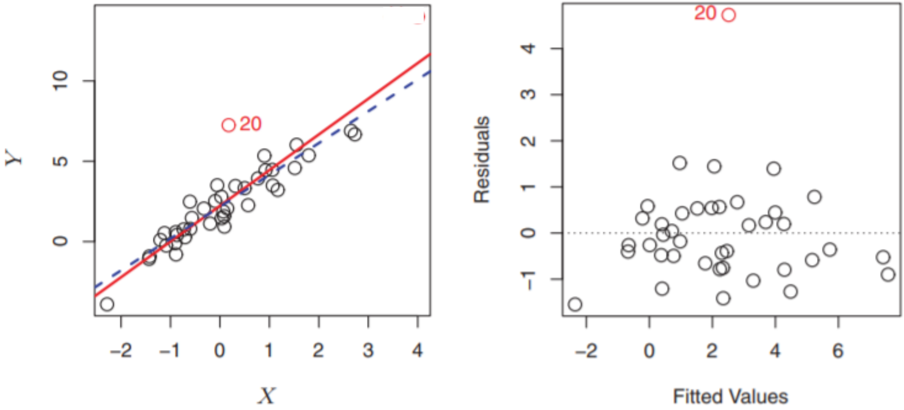
Other feature normalization

- Min-Max rescaling
  - $x_{ij} \leftarrow \frac{x_{ij} - min_j}{max_j - min_j} \in [0,1]$
  - min_j and max_j: min and max value of feature j
- Mean normalization
  - $x_{ij} \leftarrow \frac{x_{ij} - \mu_j}{max_j - min_j}$
  - Mean 0

Feature standardization/normalization

- Goal is to have individual features on the same scale
- Is a pre-processing step in most learning algorithms
- Necessary for linear models and Gradient Descent
- Different options:
  - Feature standardization
  - Feature min-max rescaling
  - Mean normalization

Practical issues: Outliers



- Dashed model is without outlier point
- Linear regression is not resilient to outliers!
- Outliers can be eliminated based on residual value
  - Use different loss functions (Huber loss)

Categorical variables

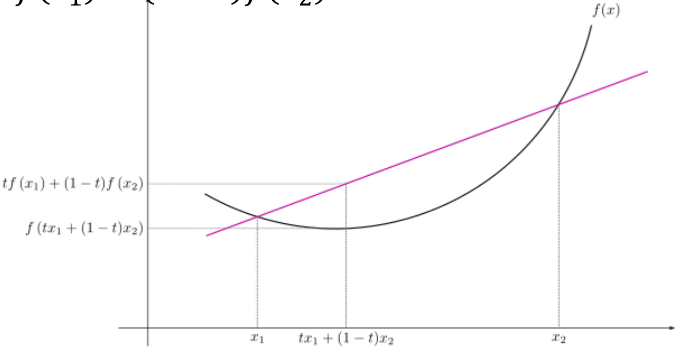
- Predict credit card balance
  - Age
  - Income
  - Number of cards
  - Credit limit
  - Credit rating
- Categorical variables
  - Student (Yes/No)
  - State (50 different levels)

Indicator Variables

- One-hot encoding
- Binary (two-level) variable
  - Add new feature $x_j = 1$ if student and 0 otherwise
- Multi-level variable
  - State: 50 values
  - $x_{MA} = 1 \text{ if State } = \text{MA and } 0, \text{otherwise}$
  - $x_{NY} = 1 \text{ if State } = \text{NY and } 0, \text{otherwise}$
  - …
  - How many indicator variables are needed?
- Disadvantages: data becomes too sparse for large number of levels
  - Will discuss feature selection later in class

How to optimize $J(\theta)$?

- (Strictly) Convex functions
  - $\forall x_1, x_2 \in X, t \in [0,1], f(tx_1 + (1-t)x_2) < tf(x_1) + (1-t)f(x_2)$



  - Have single global minimum
  - Condition for differentiable functions: $f''(x) > 0$