CSE 417T: Introduction to Machine Learning

Lecture 7: Linear Regression

Henry Chai

09/18/18

$E_{out}(g)$ and $E_{in}(g)$

- Previously, we proved that $E_{out}(g) \approx E_{in}(g)$ given certain conditions
- But how can we find g s.t. $E_{in}(g) \approx 0$?

3 Learning Problems

Problem	Domain
Classification	$\mathcal{Y} = \{-1, +1\}$
Predicting Probabilities	$\mathcal{Y} = [0, 1]$
Regression	$\mathcal{Y} = \mathbb{R}$

Linear Models

$$h(\vec{x}) = \text{some function of } \vec{w}^T \vec{x}$$

$$\vec{x} = [1\ x_1\ x_2\ \dots\ x_d]$$

3 Learning Solutions

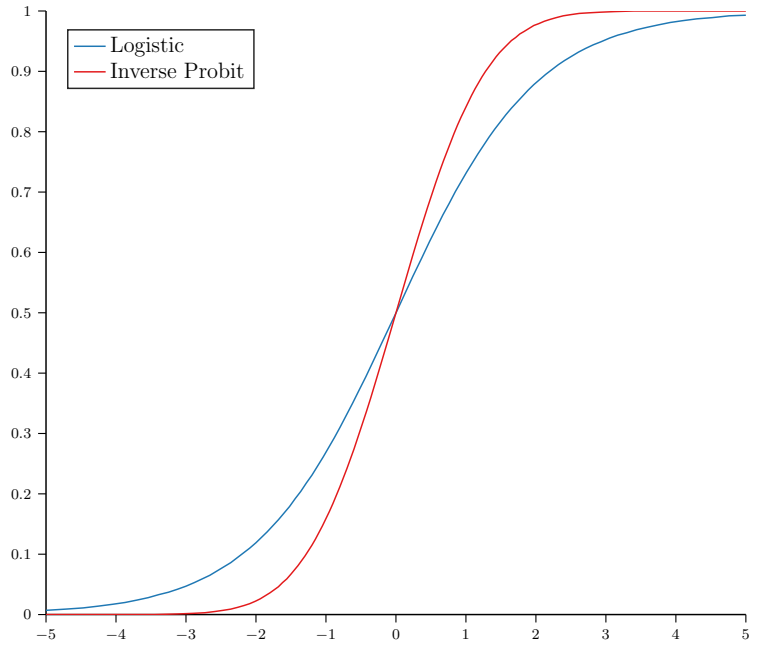
Problem	Domain
Linear Classification	$\mathcal{Y} = \{-1, +1\}$
Logistic Regression	$\mathcal{Y} = [0, 1]$
Linear Regression	$\mathcal{Y} = \mathbb{R}$

3 Learning Solutions

Problem	Model
Linear Classification	$h(\vec{x}) = \text{sign}(\vec{w}^T\vec{x})$
Logistic Regression	$h(\vec{x}) = \theta(\vec{w}^T\vec{x})$
Linear Regression	$h(\vec{x}) = \vec{w}^T\vec{x}$

What is θ?

- θ is a sigmoid function, an "S"-shaped function that maps $\mathbb{R} \to [0, 1]$
  - $\theta(x) = \frac{1}{1+e^{-x}}$ or $\theta(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{x^2}{2}}$



Linear Regression

- $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$
- $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_n}, y_n)\}$
- $X = \begin{bmatrix} 1 & \overrightarrow{x_1}^T \\ 1 & \overrightarrow{x_2}^T \\ \vdots & \vdots \\ 1 & \overrightarrow{x_n}^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{12} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- Squared Error

Squared Error

$$E_{in}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(\overrightarrow{x_i}) - y_i)^2$$

Squared Error

$$E_{in}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n}(\vec{w}^T\vec{x_i} - y_i)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(\vec{x_i}^T\vec{w} - y_i\right)^2$$

$$= \frac{1}{n}\|X\vec{w} - \vec{y}\|^2 \text{ where } \|\vec{z}\| = \sqrt{\sum_{i=1}^{d} z_i^2} = \sqrt{\vec{z}^T\vec{z}}$$

$$= \frac{1}{n}(X\vec{w} - \vec{y})^{\mathrm{T}}(X\vec{w} - \vec{y})$$

Minimizing Error

- Find the gradient
- Set it equal to zero
- Solve
- (Check that the solution is a minimum)

Minimizing Error

$$E_{in}(\vec{w}) = \frac{1}{n}(X\vec{w} - \vec{y})^{\mathrm{T}}(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}\left((X\vec{w})^{T} - \vec{y}^{\mathrm{T}}\right)(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}(\vec{w}^{T}X^{T}X\vec{w} - 2\vec{w}^{T}X^{T}\vec{y} + \vec{y}^{T}\vec{y})$$

$$\rightarrow \nabla_{\vec{w}}E_{in}(\vec{w}) = \frac{1}{n}(2X^{T}X\vec{w} - 2X^{T}\vec{y})$$

Minimizing Error

$$E_{in}(\vec{w}) = \frac{1}{n}(X\vec{w} - \vec{y})^{\mathrm{T}}(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}\left((X\vec{w})^{T} - \vec{y}^{\mathrm{T}}\right)(X\vec{w} - \vec{y})$$

$$= \frac{1}{n}(\vec{w}^{T}X^{T}X\vec{w} - 2\vec{w}^{T}X\vec{y} + \vec{y}^{T}\vec{y})$$

$$\rightarrow \nabla_{\vec{w}}E_{in}(\vec{w}^{*}) = \frac{1}{n}(2X^{T}X\vec{w}^{*} - 2X^{T}\vec{y}) = 0$$

$$\rightarrow 2X^{T}X\vec{w}^{*} - 2X^{T}\vec{y} = 0$$

$$\rightarrow X^{T}X\vec{w}^{*} = X^{T}\vec{y}$$

$$\rightarrow \vec{w}^{*} = (X^{T}X)^{-1}X^{T}\vec{y}$$

Checking

$$\nabla_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X\vec{w} - 2X^T\vec{y})$$

$$H_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X)$$

$\rightarrow H_{\vec{w}} E_{in}(\vec{w})$ is positive semidefinite

Checking

$$\nabla_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X\vec{w} - 2X^T\vec{y})$$

$$H_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{n}(2X^T X)$$

$\rightarrow H_{\vec{w}} E_{in}(\vec{w})$ is (almost always) positive definite

$\rightarrow \vec{w}^* = (X^T X)^{-1} X^T \vec{y}$ is a unique global minimum

Linear Regression "Algorithm"

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \ldots, (\overrightarrow{x_n}, y_n)\}$

1. Construct X and $\vec{y}$
2. Compute the pseudo-inverse of $X = X^{\dagger} = (X^T X)^{-1} X^T$
3. Compute $\vec{w}^* = \mathrm{X}^{\dagger}\vec{y}$

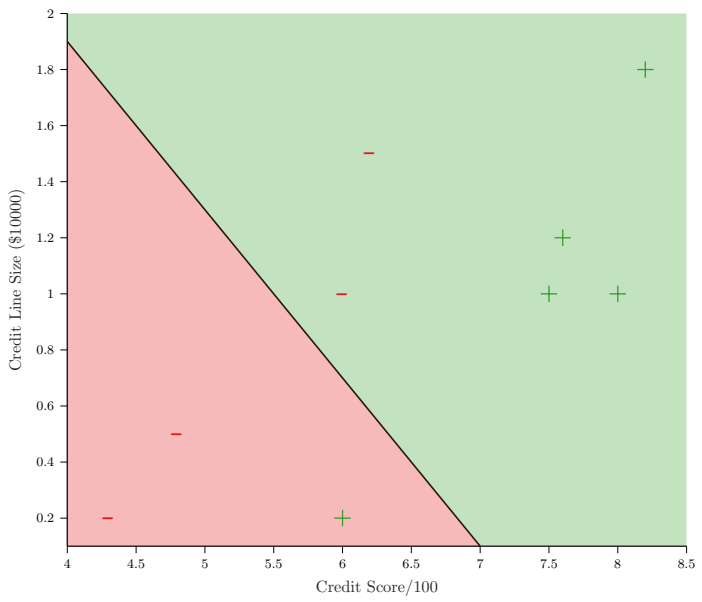
- Output: $\vec{w}^*$

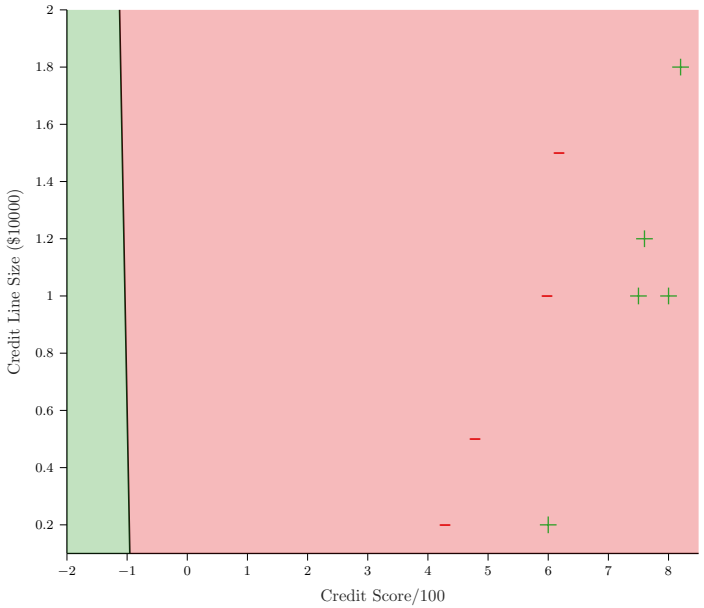
Linear Classification

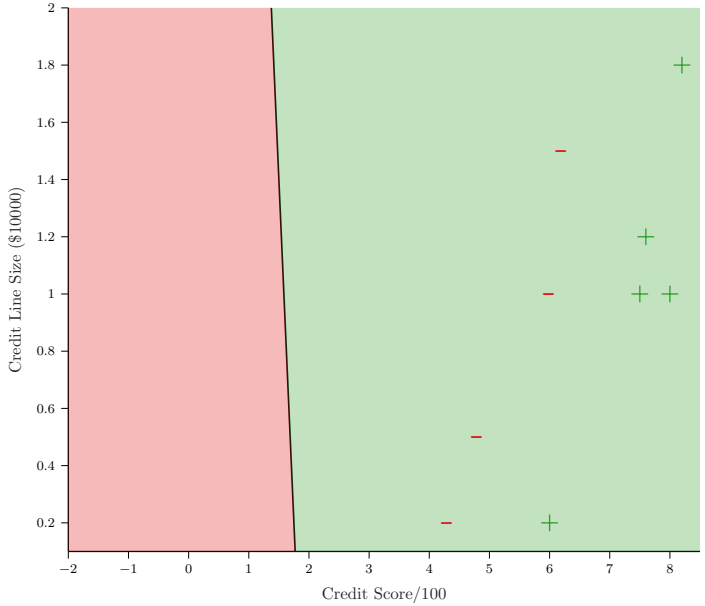
- $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{-1, +1\}$
- $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_n}, y_n)\}$
- $X = \begin{bmatrix} 1 & \overrightarrow{x_1}^T \\ 1 & \overrightarrow{x_2}^T \\ \vdots & \vdots \\ 1 & \overrightarrow{x_n}^T \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ 1 & x_{12} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nd} \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- Binary Error

Recall: Perceptron Learning Algorithm

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \dots, (\overrightarrow{x_n}, y_n)\}$
- Initialize $\vec{w}$ to all zeros
- While (∃ some misclassified training example)
  - Randomly pick a misclassified training example, $(\vec{x}, y)$
  - Update $\vec{w}$: $\vec{w} = \vec{w} + y\vec{x}$
- Output: $\vec{w}$
- PLA finds a linear separator in finite time, **if the data is linearly separable**







Recall: Perceptron Learning Algorithm

Binary Error

$$E_{in}(h) = \frac{1}{n}\sum_{i=1}^{n} [\![h(\vec{x_i}) \neq y_i]\!]$$

Binary Error

$$E_{in}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} [\![\text{sign}(\vec{w}^T\vec{x_i}) \neq y_i]\!]$$

awkward silence

Linear Regression for Classification

- Key observation: $\{-1, +1\} \subset \mathbb{R}$
- Use linear regression to find $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$
- $\vec{w}^*$ minimizes $E_{in}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (\vec{w}^T \vec{x_i} - y_i)^2$
- In general, $\text{sign}\left(\vec{w}^{*T} \vec{x_i}\right) \approx y_i \; \forall \; i$
- Use $\vec{w}^*$ for linear classification: $g(\vec{x}) = \text{sign}\left(\vec{w}^{*T} \vec{x}\right)$

The Pocket Algorithm

- Input: $\mathcal{D} = \{(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_n}, y_n)\},\ T$
- Initialize $\vec{w}$ to all zeros and $E_{best} = \infty$
- For $t = 1, 2, \ldots, T$
  - Randomly pick a misclassified training example, $(\vec{x}, y)$
  - Update $\vec{w}$: $\vec{w} = \vec{w} + y\vec{x}$
  - If $E_{in}(\vec{w}) < E_{best}$
    - $E_{best} = E_{in}(\vec{w})$
    - $\vec{w}^* = \vec{w}$
- Output: $\vec{w}^*$