Logistic Regression

CS 584: Data Mining

Prof. Sanmay Das (Adapted from notes of Prof. Malik Magdon-Ismail)

Recap + Setting

- Linear regression: Find a weight vector **w** that minimizes $E_{\text{in}}(\mathbf{w}) = \dfrac{1}{n}\sum_{i=1}^{n}(\mathbf{w}\,.\,\mathbf{x}_i - f(\mathbf{x}_i))^2$

- $h(\mathbf{x}) = \mathbf{w}\,.\,\mathbf{x}$

- Classification problem: Will someone have a heart attack over the next year?

age	62 years
gender	male
blood sugar	120 mg/dL40,000
HDL	50
LDL	120
Mass	190 lbs
Height	5′ 10″
. . .	. . .

The Supervised Learning Problem

- Unknown target function $f : \mathcal{X} \to \mathcal{Y}$
  - Classification is when $\mathcal{Y}$ is categorical (e.g. binary)
- Training data $\mathcal{D} : (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)$ where $y_i = f(\mathbf{x}_i)$ (possibly noisy).
- Want to learn h "close to" f.
- Two central questions:
  - How do we learn h?
    - ★ Key algorithmic question!
  - What can we say about how close h is to f?
    - ★ Why is this hard?

- Logistic regression: Predict probability of a heart attack. $y \in [0,1]$

- $h(\mathbf{x}) = \sigma(\mathbf{w}\,.\,\mathbf{x})$ where $\sigma(z) = \dfrac{1}{1+e^{-z}}$

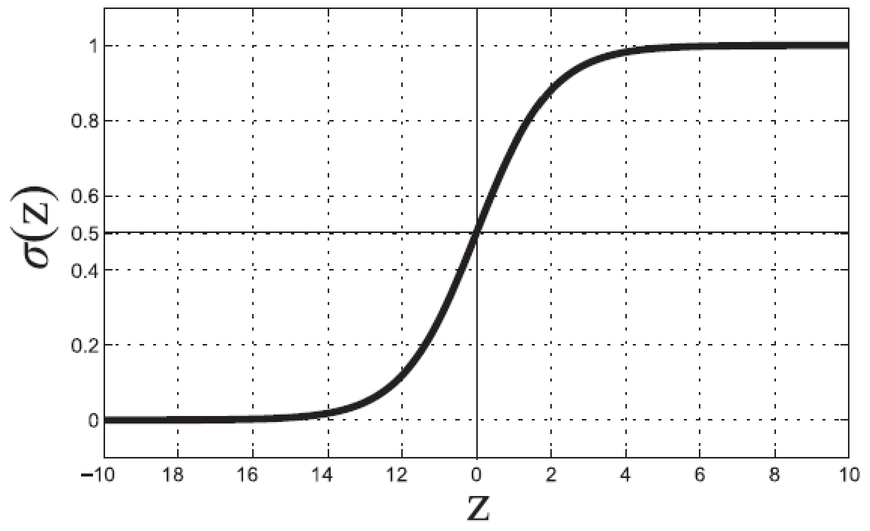

Figure 4.19. Plot of sigmoid (logistic) function, $\sigma(z)$.

Understanding the Setting

- The data is still binary. $y \in \{-1,1\}$
- We never see the *probability* $f(\mathbf{x}) = \Pr[y = 1 \mid \mathbf{x}]$ of someone having a heart attack
    - Just whether they *did* or not
    - Ways to think about this:
        - Each individual has a risk state or subjective probability, and whether the outcome happens is based on a biased coin flip
        - Each individual has a true state $\in \{-1, +1\}$ and then the data is generated from a *noisy* target function $\Pr[y = 1 \mid \mathbf{x}] = f(\mathbf{x})$

Finding a Good Hypothesis

- Our hypothesis is basically just a vector **w** — the logistic transformation is pre-specified.
- Two notes:
  - I will use the convention of just using **w** rather than *b* and **w** separately. The simplest way to think about this is just of augmenting the **x** vector with a first element that always has the value 1 for any *i*.
  - We are using labels -1 and +1. The book uses 0, 1 for logistic regression. This leads to minor differences in formulas.
- What does it mean to find a “good” **w** in this setting?
- Two ways to see it:
  - Minimizing an error measure (conventional ML view)
  - Maximizing the probability of seeing the actual *y* values we saw

Probabilistic Interpretation

- $\mathscr{L}(\mathbf{w}) = \Pi_{i=1}^{n} \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$
- We want to pick the **w** that maximizes this. Let's walk through this:
    - Same as maximizing: $\ln \mathscr{L}(\mathbf{w}) = \sum_{i=1}^{n} \ln \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$
    - Or, minimizing: $-\ln \mathscr{L}(\mathbf{w})$.
    - Which is the same as minimizing: $-\frac{1}{n} \sum_{i=1}^{n} \ln \Pr(y_i \mid \mathbf{x}_i, \mathbf{w})$.
    - I can write this minimization as: $\min \frac{1}{n} \sum_{i=1}^{n} \ln \frac{1}{\Pr(y_i \mid \mathbf{x}_i, \mathbf{w})}$

Minimizing the log likelihood

- When $y = +1, \Pr(y \,|\, \mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w} \,.\, \mathbf{x}) = \sigma(y_i \mathbf{w} \,.\, \mathbf{x})$
- When $y = -1, \Pr(y \,|\, \mathbf{x}, \mathbf{w}) = 1 - \sigma(\mathbf{w} \,.\, \mathbf{x}) = \sigma(-\mathbf{w} \,.\, \mathbf{x}) = \sigma(y_i \mathbf{w} \,.\, \mathbf{x})$
- So we want to minimize $\displaystyle \frac{1}{n} \sum_{i=1}^{n} \ln \frac{1}{\sigma(y_i \mathbf{w} \,.\, \mathbf{x}_i)} = \frac{1}{n} \sum_{i=1}^{n} \ln(1 + e^{-y_i \mathbf{w} \,.\, \mathbf{x}_i})$
- This is also known as the **cross entropy error**
  - Exercises:
    - Error if you “predict” 0.9 on a +1 example? -1 example?
    - What if you “predict” 0.7 on a +1 example? -1 example?

Low when y_i has the same sign as w.x$_i$ and is large

$\approx 0.105, 2.303$

$\approx 0.357, 1.204$

Minimizing Cross-Entropy Error

- Turns out to be a convex function, therefore has a unique minimum.
- Can use conventional optimization methods (e.g. Newton-Raphson).
- Later today we'll talk about how to do this using *gradient descent*, a key technique in modern ML and DM.