Lecture 8: 1 February, 2024

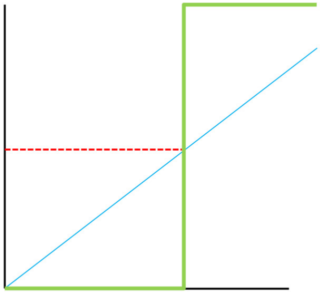
Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Data Mining and Machine Learning
January–April 2024

Regression for classification

- Regression line
- Set a threshold
- Classifier
  - Output below threshold : 0 (No)
  - Output above threshold : 1 (Yes)
- Classifier output is a step function



Smoothen the step

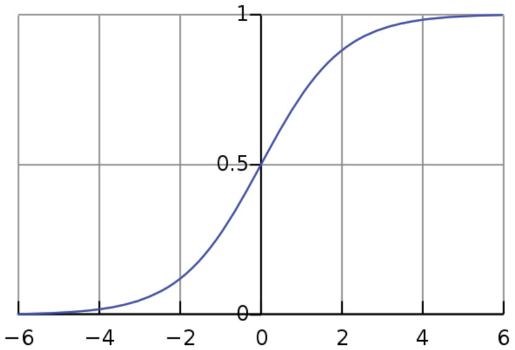
- Sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Input z is output of our regression

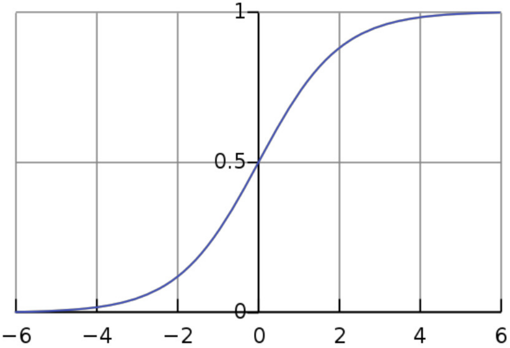
$$\sigma(z) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \cdots + \theta_k x_k)}}$$

- Adjust parameters to fix horizontal position and steepness of step



Logistic regression

- Compute the coefficients?
- Solve by gradient descent
- Need derivatives to exist
  - Hence smooth sigmoid, not step function
  - $\sigma'(z) = \sigma(z)(1 - \sigma(z))$
- Need a cost function to minimize



Loss function for logistic regression

- Goal is to maximize log likelihood
- Let $h_\theta(x_i) = \sigma(z_i)$. So, $P(y_i = 1 \mid x_i; \theta) = h_\theta(x_i)$,
  $P(y_i = 0 \mid x_i; \theta) = 1 - h_\theta(x_i)$
- Combine as $P(y_i \mid x_i; \theta) = h_\theta(x_i)^{y_i} \cdot (1 - h_\theta(x_i))^{1-y_i}$
- Likelihood: $\mathcal{L}(\theta) = \prod_{i=1}^{n} h_\theta(x_i)^{y_i} \cdot (1 - h_\theta(x_i))^{1-y_i}$
- Log-likelihood: $\ell(\theta) = \sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i))$
- Minimize cross entropy: $-\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1 - y_i) \log(1 - h_\theta(x_i))$

MSE for logistic regression and gradient descent

- Suppose we take mean sum-squared error as the loss function.
- Consider two inputs $x = (x_1, x_2)$

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \sigma(z_i))^2, \text{ where } z_i = \theta_0 + \theta_1 x_{i_1} + \theta_2 x_{i_2}$$

- For gradient descent, we compute $\dfrac{\partial C}{\partial \theta_1}, \dfrac{\partial C}{\partial \theta_2}, \dfrac{\partial C}{\partial \theta_0}$
  - For $j = 1, 2$,

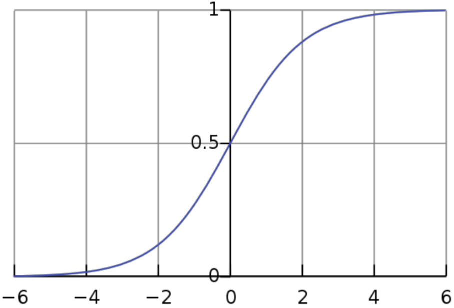
$$\frac{\partial C}{\partial \theta_j} = \frac{2}{n}\sum_{i=1}^{n}(y_i - \sigma(z_i)) \cdot -\frac{\partial \sigma(z_i)}{\partial \theta_j} = \frac{2}{n}\sum_{i=1}^{n} (\sigma(z_i) - y_i) \frac{\partial \sigma(z_i)}{\partial z_i}\frac{\partial z_i}{\partial \theta_j}$$

$$= \frac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)x_{i_j}$$

  - $\dfrac{\partial C}{\partial \theta_0} = \dfrac{2}{n}\displaystyle\sum_{i=1}^{n}(\sigma(z_i) - y_i)\frac{\partial \sigma(z_i)}{\partial z_i}\frac{\partial z_i}{\partial \theta_0} = \frac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)$

MSE for logistic regression and gradient descent ...

- For $j = 1, 2$, $\dfrac{\partial C}{\partial \theta_j} = \dfrac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)x_j^i$, and $\dfrac{\partial C}{\partial \theta_0} = \dfrac{2}{n}\sum_{i=1}^{n}(\sigma(z_i) - y_i)\sigma'(z_i)$
- Each term in $\dfrac{\partial C}{\partial \theta_1}, \dfrac{\partial C}{\partial \theta_2}, \dfrac{\partial C}{\partial \theta_0}$ is proportional to $\sigma'(z_i)$
- Ideally, gradient descent should take large steps when $\sigma(z) - y$ is large
- $\sigma(z)$ is flat at both extremes
- If $\sigma(z)$ is completely wrong, $\sigma(z) \approx (1 - y)$, we still have $\sigma'(z) \approx 0$
- Learning is slow even when current model is far from optimal



Cross entropy and gradient descent

- $C = -[y \ln(\sigma(z)) + (1-y)\ln(1-\sigma(z))]$

- $$\begin{aligned}
\frac{\partial C}{\partial \theta_j} = \frac{\partial C}{\partial \sigma}\frac{\partial \sigma}{\partial \theta_j} &= -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\frac{\partial \sigma}{\partial \theta_j} \\
&= -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\frac{\partial \sigma}{\partial z}\frac{\partial z}{\partial \theta_j} \\
&= -\left[\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right]\sigma'(z)x_j \\
&= -\left[\frac{y(1-\sigma(z)) - (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}\right]\sigma'(z)x_j
\end{aligned}$$

Cross entropy and gradient descent . . .

- $\frac{\partial C}{\partial \theta_j} = -\left[\frac{y(1-\sigma(z)) - (1-y)\sigma(z)}{\sigma(z)(1-\sigma(z))}\right]\sigma'(z)x_j$

- Recall that $\sigma'(z) = \sigma(z)(1-\sigma(z))$

- Therefore, $\begin{aligned}\frac{\partial C}{\partial \theta_j} &= -[y(1-\sigma(z)) - (1-y)\sigma(z)]x_j \\ &= -[y - y\sigma(z) - \sigma(z) + y\sigma(z)]x_j \\ &= (\sigma(z) - y)x_j\end{aligned}$

- Similarly, $\frac{\partial C}{\partial \theta_0} = (\sigma(z) - y)$

- Thus, as we wanted, the gradient is proportional to $\sigma(z) - y$

- The greater the error, the faster the learning rate