

Logistic Regression

Sidharth Baskaran

June 2021

Classification

- Is a discrete categorization of an output variable, e.g. $y \in \{0, 1\}$
 - Can also have multiple classes, so some set $S \mid \text{len}(S) > 2$
- Linear regression \rightarrow threshold classifier output
 - E.g. if $h_\theta(x) \geq 0.5$ do $y = 1$ else $y = 0$
 - However is not effective when there are > 2 clusters \rightarrow DNU
- Classification - $y = 0$ or $y = 1$
 - $0 \leq h_\theta(x) \leq 1$
 - Binary classification
- bruh
 -

Hypothesis Representation

- Need $0 \leq h_\theta(x) \leq 1$
- $h_\theta(x) = g(\theta^T x)$ where $g = \frac{1}{1+e^{-z}}$ is the sigmoid = logistical function and $z = \theta^T x$
 - $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
- Interpretation
 - $h_\theta(x)$ is estimated probability that $y = 1$ on input x
 - $h_\theta(x) = P(y = 1|x; \theta)$ is probability of $y = 1$ given x parameterized by θ
 - * Due to total sum probability $\rightarrow P(y = 0|x; \theta) = 1 - P(y = 1|x; \theta)$

Decision Boundary

- Prediction boundary - If $h_\theta(x) \geq 0.5$ do $y = 1$ else $y = 0$
 - Thus $y = 0 \implies \theta^T x < 0$ and $y = 1 \implies \theta^T x \geq 0$
 - Graph the equation $\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n \geq 0$ (higher order planes of \mathbb{R}^{n+1})
- Nonlinear decision boundaries
 - Have polynomial terms in features
 - Ex. $-1 + x_1^2 + x_2^2 \geq 0 \implies$ unit circle

Logistic cost function

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

- Setup/prime
 - Training set $S = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
 - * m examples

- A feature vector $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$
- $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$
- Cost $(h_\theta(x), y) = \frac{1}{2} (h_\theta(x) - y)^2$
 - Is non-convex - many local extrema which may hinder gradient descent
- Cost function for logistic regression

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

- Cost function behavior
 - Cost is 0 if $y = 1, h_\theta(x) = 1$
 - As $h_\theta(x) \rightarrow 0$, Cost $\rightarrow \infty$
 - If $h_\theta(x) = 0$ **but** $y = 1$, then learning algorithm penalized heavily

Simplified Cost Function

- Can rewrite cost function to be $\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$
 - Vectorized - $J(\theta) = \frac{1}{m} \cdot (-y^T \log(h) - (1 - y)^T \log(1 - h))$ where $h = g(X\theta)$
- Gradient descent algorithm
 - Simultaneous update and iterate $\theta_j := \theta_j - \alpha \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$
 - Vectorized - $\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$

Advanced optimization

- Optimization algorithm -> minimize $J(\theta)$
 - Need to compute $J(\theta)$ and $\frac{\partial}{\partial \theta_j} J(\theta)$
- Algorithms
 - Gradient descent
 - Conjugate gradient
 - BFGS
 - L-BFGS
 - Do not need to manually pick α for last 3 -> but more complex
- Use `fminunc` in Octave

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = fminunc(@costFunction, initialTheta, options);
```

- Function for cost function
 - `function [jVal, gradient] = costFunction(theta)`
 - Must return $J(\theta)$ and gradient

Multiclass Classification with Logistic Regression

- One-vs-all classification
 - Combine remaining classes to 1 class and compare with another -> multiple binary classifications
 - Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict probability that $y = i$
 - Pick class that maximizes $h \rightarrow \max_i h_\theta^{(i)}(x)$

Overfitting

- Underfitting - high **bias**
 - Straight line fit - biased to linear trend
 - Too little features - n is small
- Overfitting - high **variance**
 - Space of possible hypothesis too large, can't find a good hypothesis
 - Too many features - n too large
 - * Cannot generalize well to new examples
- Addressing overfitting
 - Reduce n - select necessary ones and model a selection algorithm
 - Regularization - keep features but reduce magnitude of values in θ
 - * Works well with lots of features

Regularization cost function

- Small values for parameters θ_j for $j \in [0, n]$
 - Simpler hypothesis
 - Less prone to overfitting
- Modify cost function to shrink parameters

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

- Convention - only regularize $\theta_1, \dots, \theta_n$ and ignore θ_0
- Regularization parameter λ controls tradeoff of small parameters and well-fitting
 - Therefore prevents overfitting
 - A large value highly reduces $\theta \rightarrow \vec{0}$ so $h_{\theta}(x) = \theta_0$ so a horizontal line is fitted
 - Important to choose

Regularized Linear Regression

- Gradient descent - separate terms in algorithm

Repeat

$$\begin{aligned} \theta_0 &:= \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_0^{(i)} \\ \theta_j &:= \theta_j - \alpha \left[\left(\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)} \right) + \frac{\lambda}{m} \theta_j \right] \end{aligned}$$

- Update rule can be expressed as $\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$
 - Term $1 - \alpha \frac{\lambda}{m} < 1$ always so it reduces θ_j by some amount each update
 - Keeps 2nd term same as previously
- Normal equation

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

- Matrix L near λ is $n + 1 \times n + 1$
 - Due to 0 as first element
- Given $\lambda > 0$

- If $m \leq n$ then $X^T X$ is noninvertible
 - Adding term λL makes $X^T X + \lambda L$ invertible

Regularized Logistic Regression

- Logistic growth prone to overfitting with many features (high n)
- Modify to use regularization
 - Add term $\frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$
- Treat θ_0 separately

$$J(\theta) = \left[-\frac{1}{m} \sum_{i=1}^m y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

- Advanced optimization method
 - Octave has 1-indexed vectors

```
function [jval, gradient] = costFunction(theta)
jval = [code to get J(theta)]
gradient(1) = [partial respect to theta_0]
.
.
.
gradient(n + 1) = [partial with respect to theta_n]
```