Generalized Linear Model

Fangfang Song

June 2018

1 Generalized linear model

Consider a classification or regression problem where we would like to predict the value of some random variable y as a function of x. To derive a GLM for this problem, we will make the following assumptions:

1. $y|x;\theta \sim$ Exponential Family $F(y)$

   Note: Exponential Family distributions:

   $$p(y;\eta) = b(y)exp(\eta^T T(y) - a(\eta)) \tag{1}$$

   η: natural parameter/canonical parameter; $T(y)$: sufficient statistic, often $T(y) = y$ $a(\eta)$: log partition function

2. Given x, our goal is to predict the expected value of $T(y)$ given x. In most of our examples, we will have $T(y) = y$, so this means we would like the prediction $h(x)$ output by our learned hypothesis h to satisfy $h(x) = E[y|x]$

3. Design Choice: The natural parameter η and the inputs x are related linearly: $\eta = \theta^T x$

Model

1. Model $p(y|x;\theta)$

2. Log likelihood $l(\theta) = \sum_{i=1}^{m} log(p(y^i|x^i;\theta))$

3. Learn the parameter by maximizing the likelihood.

4. Make a prediction: find the link function between distribution parameter and η, invert the link function and derive the **canonical response function** $g(\eta) = E[T(y);\eta]$, most of the time, T(y) = y, the canonical response function or the hypothesis $h(x) = E[y|x]$

Examples of GLM includes linear regression, logistic regress and softmax regression.

1.1 Linear Regression

$$y|x;\theta \sim Gaussian(\mu, \sigma^2)$$
$$p(y^i|x^i;\theta) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}) \tag{2}$$

Log likelihood:

$$\begin{aligned} l(\theta) =& log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}) \\ =& \sum_{i=1}^{m} log \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}) \\ =& mlog \frac{1}{2\pi\sigma} - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2 \end{aligned}$$

Maximize $l(\theta)$ gives the **cost function** $J(\theta)$ to minimize:

$$J(\theta) = \sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2$$

m: number of training examples
Hypothesis:

$$\begin{aligned} h_\theta(x) &= E[y|x] \\ &= \mu \\ &= \eta \\ &= \theta^T x \end{aligned}$$

θ_i : weights / parameters
x : features/ input variables
y : target / output

2 Logistic Regression

$$y|x \sim Bernoulli(\phi)$$
$$p(y|x;\theta) = \phi^y(1-\phi)^{1-y}$$

log likelihood:

$$\begin{aligned} l &= log \prod_{i=1}^{m} \phi^{y^{(i)}}(1-\phi)^{(1-y^{(i)})} \\ &= \sum_{i=1}^{m} y^{(i)} log\phi + (1-y^{(i)}) log(1-\phi) \end{aligned}$$

cost function

$$J(y,\hat{y}) = -\sum_{i=1}^{m} y^{(i)} log\hat{y} + (1-y^{(i)})log(1-\hat{y}^{(i)})$$

Hypothesis:

$$\begin{aligned} h_\theta(x) &= E[y|x] \\ &= \phi \\ &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-\theta^T x}} \end{aligned} \tag{3}$$

One vs. rest Logistic Regression: In one vs rest logistic regression a separate model is trained for **each** class predicting whether an observation is that class or not (thus making it a binary classification problem). It assumes that each classification problem (e.g. class 0 or not) is independent.

3 Softmax Regression

$$y|x \sim Multinomal(\phi)$$

$$\begin{aligned} p(y|x;\phi) &= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}...\phi_k^{1\{y=k\}} \\ &= \phi_1^{1\{y=1\}}\phi_2^{1\{y=2\}}...\phi_k^{1-\sum_{i=1}^{k-1}(1\{y=i\})} \\ &= b(y)exp(\eta^T T(y) - a(\eta)) \end{aligned}$$

Hypothesis

$$\begin{aligned} h_\theta(x) = P(y=i|x;\theta) = \phi_i \\ = \frac{e^{\eta_j}}{\sum_{j=1}^{k} e^{\eta_j}} \end{aligned}$$

log likelihood

$$\begin{aligned} l(\theta) &= \sum_{i=1}^{m} log(p(y^{(i)}|x^i;\theta)) \\ &= \sum_{i=1}^{m} log\Pi_{l=1}^{k} \frac{e^{\theta_l^T x^i}}{\sum_{j=1}^{k} e^{\theta_j^T x^i}})^{1\{y_i=1\}} \end{aligned}$$

cross entropy loss

$$\begin{aligned} J(y,\hat{y}) &= -l(\theta) \\ &= -\sum_{i=1}^{m} y^i log(\hat{y}^i) \end{aligned}$$

4 Optimization algorithms

4.1 Gradient Descent algorithm

$$\begin{aligned}\theta_j &:= \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ &= \theta_j + \sum_{i=1}^{m} \alpha (y^{(i)} - h_\theta(x^{(i)})) x_j^{(i)}\end{aligned} \tag{4}$$

α: learning rate

4.1.1 Batch Gradient Descent Algorithm

Repeat until convergence{

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} (y^i - h_\theta(x^i)) x_j^i \qquad \text{(For every j)} \tag{5}$$

}

4.1.2 Incremental/stochastic gradient descent

In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only. Whereas batch gradient descent has to scan through the entire training set before taking a single step, which is a costly operation if m is large, stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at. So, stochastic gradient descent gets θ close to the minimum much faster than batch gradient descent.

```
loop{
for i = 1 to m, {
        $\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}$        (6)
    }
}
```

4.2 Newton's method

$$\theta := \theta - \frac{f('\theta)}{f''(\theta)} \tag{7}$$

Generalize Newton's method to vector setting:

$$\theta := \theta - H^{-1} \nabla_\theta l(\theta) \tag{8}$$

$$\text{Hessian} \qquad H_{ij} = \frac{\partial^2 l(\theta)}{\partial \theta_i \partial \theta_j} \tag{9}$$

4.3 Normal Equation for linear regression

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - \vec{y}) \\
&= \frac{1}{2}\nabla_\theta(\theta^T X^T X\theta - \theta^T X^T X\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}) \\
&= \frac{1}{2}\nabla_\theta tr(\theta^T X^T X\theta - \theta^T X^T X\vec{y} - \vec{y}^T X\theta + \vec{y}^T\vec{y}) \\
&= \frac{1}{2}_\theta(tr(\theta^T X^T X\theta) - 2tr(\vec{y}^T X\theta)) \\
&= \frac{1}{2}\nabla_\theta(tr(\theta^T X^T X\theta) - 2(\vec{y}^T X\theta)) \\
&= \frac{1}{2}(X^T X\theta + X^T X\theta - 2X^T\vec{y}) \\
&= X^T X\theta - X^T\vec{y}
\end{aligned} \tag{10}$$

Set $\nabla_\theta J(\theta) = 0$, we get

$$X^T X\theta = X^t\vec{y} \tag{11}$$

$$\theta = (X^T X)^{-1}X^T\vec{y} \tag{12}$$

if $m \leq n$, $X^T X$ is noninvertable/singular.

4.4 Conjugate gradient

4.5 BFGS

4.6 L-BFGS

5 Underfitting and Overfitting

Address overfitting Options:

1. Reduce number of features: use model selection algorithm to select which features to keep
2. Regulation: keep all features, but reduce magnitude/values of the parameters; this method works well when we have a lot of features, each of which contribute a bit to predicting y. Small value of parameters lead to "simpler hypothesis" and less prone to overfitting

Mathematically, we add an regulation to the cost function:

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

By convention, θ_0 is not penalized.By practice, it make very little different whether include θ_0 or not.

Withe the regulation, Newton method's for linear regression becomes:

$$\theta = (X^T X - \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix})^{-1} X^T \vec{y})$$

Regulation also take cares of noninvertability issues as well: as long as > 1, it can be proved that the new matrix is always invertable.