LogisticRegression

Tuesday, August 11, 2020 9:08 AM

(A) Derivation of logistic Regression

input(x) $\longrightarrow$ hypothesis function f $\longrightarrow$ $\hat{y}$

"learn f"

True label $\rightarrow y$

$Div(\hat{y}, y)$ is loss.

$y | x ; \theta \sim \text{Bernoulli}(\phi)$

or generally $\sim$ Exponential family (η)

<u>PMF</u> $p(y ; \eta) = b(y) \exp(\eta^T T(y) - a(\eta))$

$h(x) = E[y|x]$

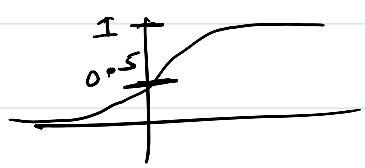
or $h_\theta(x) = E[y|x ; \theta] = \phi = \dfrac{1}{1+e^{-\eta}}$

$= \dfrac{1}{1+e^{-\theta^T x}}$

$g(\eta) = E[T(y) ; \eta]$ is cannonical response function

In our case Sigmoid.

$h_\theta(x) = g(\theta^T x) = \dfrac{1}{1+e^{-\theta^T x}}$ when g is sigmoid



<u>PMF</u>

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

$$= (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$$

Likelihood (function of θ)

$$L(\theta) = P(y \mid x; \theta)$$

$$= \prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)}; \theta)$$

$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$

assumption, $\epsilon^{(i)}$ are IID distributed

$$= \prod_{i=1}^{n} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}$$

Since log is strictly increasing function, we could find θ that maximize $\log(L(\theta))$

so,

$$\ell(\theta) = \log L(\theta) = \sum_{i=1}^{n} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

$$\frac{\partial}{\partial \theta_j} \ell(\theta) = (y - h_\theta(x)) x_j$$

Note $[\, g'(x) = g(x)(1 - g(x))$

so, $\theta := \theta + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x$

(B) Problem set 1(A) and 1(B)

(a) [10 points] In lecture we saw the average empirical loss for logistic regression:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)})),$$

where $y^{(i)} \in \{0,1\}$, $h_\theta(x) = g(\theta^T x)$ and $g(z) = 1/(1+e^{-z})$.
Find the Hessian H of this function, and show that for any vector z, it holds true that

$$z^T H z \geq 0.$$

Hint: You may want to start by showing that $\sum_i \sum_j z_i x_i x_j z_j = (x^T z)^2 \geq 0$. Recall also that $g'(z) = g(z)(1-g(z))$.

Remark: This is one of the standard ways of showing that the matrix H is positive semi-definite, written "$H \succeq 0$." This implies that J is convex, and has no local minima other than the global one. If you have some other way of showing $H \succeq 0$, you're also welcome to use your method instead of the one above.

Answer:

$$\nabla_\theta J(\theta) = \begin{bmatrix} | \\ \frac{\partial}{\partial\theta_i} J(\theta) \\ | \end{bmatrix}$$

For simplicity assume we have only one training example on the set so we have

$$\frac{\partial}{\partial\theta_i} J(\theta) = \frac{\partial}{\partial\theta_i}\left(y\log(h_\theta(x)) + (1-y)\log(1-h_\theta(x))\right)$$

$$= \frac{y}{h_\theta(x)}\frac{\partial}{\partial\theta_i}h_\theta(x) + \frac{(1-y)}{1-h_\theta(x)}\frac{\partial}{\partial\theta_i}(1-h_\theta(x))$$

$$= g'(\theta^T x)x_i\left(\frac{y}{h_\theta x} + \frac{y-1}{1-h_\theta x}\right)$$

$$\left(\because \; h_\theta(x) = g(\theta^T x), \quad g'(\theta^T x) = g(\theta^T x)(1-g(\theta^T x))\right)$$

$$= g'(\theta^T x)x_i\left(\frac{y-h_\theta(x)}{h_\theta(x)(1-h_\theta(x))}\right)$$

$$= \cancel{g(\theta^T x)}x_i\frac{(y-h_\theta x)}{\cancel{g(\theta^T x)}}$$

$$= (y - h_\theta x)x_i$$

$i \leftrightarrow j$ s.t. $\frac{\partial}{\partial\theta_j}J(\theta) = (y - h_\theta(x))x_j$

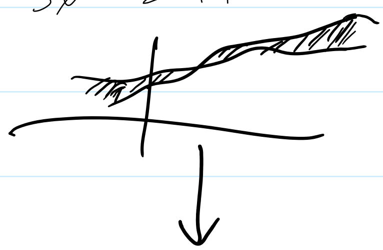
Now

$$H_{ij} = x_j\, h_\theta(x)(h_\theta(x)-1)\,x_i$$

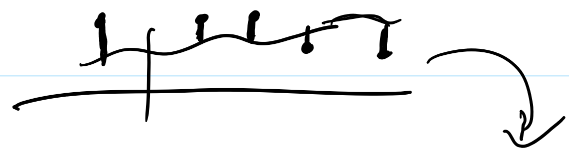
$$z^T H z = \sum_i\sum_j H_{ij} z_i z_j = \sum_i\sum_j h_\theta(x)(h_\theta(x)-1)\, z_i x_i z_j x_j$$

$$= h_\theta(x)(h_\theta(x)-1)\sum_i\sum_j z_i x_i z_j x_j$$
$$= h_\theta(x)(h_\theta(x)-1)(x^T z)^2$$
$$\geq 0$$

So $z^T H z \geq 0$. Hence, $H \succeq 0$ i.e. H is PSD. So, H is convex

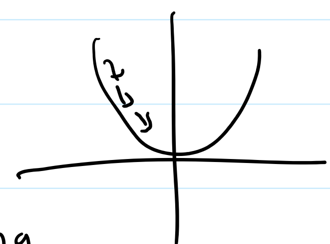


Use input sample to estimate error.



$$E[\mathrm{div}(f(x;\theta), g(x))] \approx \frac{1}{n}\sum_{i=1}^{n} \mathrm{div}(f(x_i;\theta), \hat{y}_i)$$

Implementation Notes (C)



Instead of iteratively going down hill by 'α' learning rate (Gradient descent), we used quadratic approximation using Newton's method. For vector input, ...

$$\theta := \theta - H^{-1}\nabla_\theta \ell(\theta)$$ (Newton - Raphson method)

"numerical analysis"

H is Hessian, $H_{ij} = \frac{\partial^2}{\partial\theta_i\theta_j}\ell(\theta)$

Algorithm

Training

1. Initialize the parameter θ

① Define $h_\theta(x) = g(\theta^T x)$ where g is Sigmoid
$y \in \{0, 1\}$ (in PS '1')

② Compute $\nabla_\theta J(\theta)$
emperical loss, $\frac{1}{n} (y - h_\theta(x))x$

③ Compute Hessian H,
average, $\frac{1}{n} h_\theta(x)(1-h_\theta x)\, x^T x$

④ Use Newton-Raphson, Reapeat until till Convergence

Testing

⑤ To predict, feed the input x to hypothesis, $h_\theta(x)$ to produce prediction.

(b) [5 points] **Coding problem.** Follow the instructions in `src/p01b_logreg.py` to train a logistic regression classifier using Newton's Method. Starting with $\theta = \vec{0}$, run Newton's Method until the updates to θ are small: Specifically, train until the first iteration k such that $\|\theta_k - \theta_{k-1}\|_1 < \epsilon$, where $\epsilon = 1 \times 10^{-5}$. Make sure to write your model's predictions to the file specified in the code.

Answer: