SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.007 Machine Learning, Fall 2021

Homework 3

Due: 3 November 2021
This homework is prepared by Prof. Berrak Sisman,
and will be graded by Ms. Zongyang Du (TA).

1 Logistic Regression

Question 1.1 [20 pts]
Please indicate whether the following statements are true (T) or false (F).
a) Logistic regression is a supervised machine learning algorithm.
b) Logistic regression is mainly used for regression, not classification.
c) It possible to design a logistic regression using a Neural Network Algorithm.
d) Logistic regression outputs a probability or confidence score, i.e., a value between 0 and 100.

Question 1.2 [20 pts]
Suppose that you have trained a logistic regression classifier, and it outputs on a new example a prediction $h_\theta(x) = 0.28$. This means (check all that apply):
1) Our estimate for $P(y = 0|x; \theta)$ is 0.28
2) Our estimate for $P(y = 0|x; \theta)$ is 0.72
3) Our estimate for $P(y = 1|x; \theta)$ is 0.28
4) Our estimate for $P(y = 1|x; \theta)$ is 0.72
Please explain your answer.

Question 1.3 [20 pts]
Suppose you train a logistic classifier $h_\theta(x) = g(\theta_0 + x_1\theta_1 + x_2\theta_2)$, and obtain $\theta = [3 \ -3 \ 0]^T$. Please formulate the decision boundary of your classifier. Note that this is a binary classification problem, which means class label y can be 0 or 1.

Question 1.4 [20 pts]
Suppose you train a logistic classifier $h_\theta(x) = g(\theta_0 + x_1\theta_1 + x_2\theta_2 + x_1^2\theta_3 + x_2^2\theta_4)$, and obtain $\theta = [-64 \ 0 \ 0 \ 1 \ 1]^T$. Please formulate the decision boundary of your classifier. Note that this is a binary classification problem, which means class label y can be 0 or 1.

Question 1.5 [20 pts]
In logistic regression, we find the parameters of a logistic (sigmoid) function that maximize the likelihood of a set of training examples. The likelihood is given as follows:

$$\prod_{i=1}^{n} P(y^{(i)}|x^{(i)}) \tag{1}$$

However, we re-express the problem of maximizing the likelihood as minimizing the following expression:

$$\frac{1}{n}\sum_{i=1}^{n} log\left(1 + exp\left(-y^{(i)}\left(\theta.x^{(i)} + \theta_0\right)\right)\right) \tag{2}$$

What is the benefit of optimizing the log-likelihood rather than the likelihood of the data? In other words, why is this expression computationally more "convenient"? *(Hint: try randomly generating, say, 1,000 probabilities in Python and multiplying them together as in Eq. 1.)*

1.1 a) True. It is a classifier and an algorithm. It makes use of labelled data.

b) False. Logistic regression is a classifier so it has to do classification. Typical function of a classifier.

c) True. Logistic regression can be modelled as a function that can take in any number of inputs and constrain the output to be between 0 and 1. The typical characteristic of a neural network is that you can take in multiple inputs and produce one output. Hence, logistic regression can be a one-layered neural network.

d) False. Logistic regression outputs the probability, that is a value between 0 and 1 but not a confidence score.

1.2 (choice 2) and 3). $h_\theta(x)$ represents the estimated probability that $y=1$ on input x to be 0.28 in this case.

$\therefore$ we can say that

$h_\theta(x) = P(y=1 \mid x; \theta) = 0.28$.

Thus, $P(y=0 \mid x; \theta) = 1 - 0.28 = 0.72$.

Hence choice 2 and 3

1.3 $h_\theta(x) = g(3 + (-3)x_1 + 0x_2)$

$y=1$ if $3 - 3x_1 \geq 0$

$\therefore y=1$ if $x_1 \leq 1$

$y=0$ if $3 - 3x_1 < 0$

$x_1 > 1 \Rightarrow y=0$.

1.4 We know that this is a non-linear classification.

$h_\theta(x) = g(-64 + 0x_1 + 0x_2 + 1x_1^2 + 1x_2^2)$

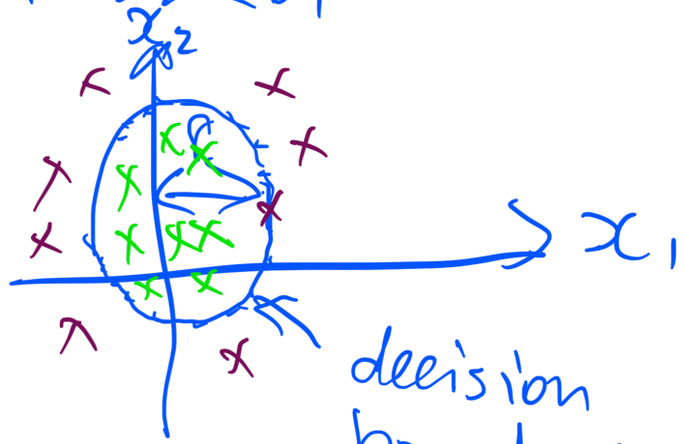
$-64 + x_1^2 + x_2^2 \geq 0$ if $y=1$

$\Rightarrow x_1^2 + x_2^2 \geq 64$ if $y=1$

Continued on the next page.

$$-64 + x_1^2 + x_2^2 < 0$$

$$x_1^2 + x_2^2 < 64$$



Maroon datapoints represent the fact that $y=1$ label

Green datapoints lie within the circle that represent $y=0$ label.

1.5 From the python script, we can multiply likelihoods and they become very small, run out of floating point precision. Hence, underflow. Using log-likelihood we can use the rules of logarithm and know that we are computing the addition which is lower in time complexity as compared to multiplication. Adding also doesn't give an underflow error as it is a summation of terms for large n.

$$\prod_{i=1}^{n} P(y^{(i)} | x^{(i)}) = \frac{P(y^{(1)} \wedge x^{(1)})}{P(x^1)} \cdot \frac{P(y^{(n)}, x^{(n)})}{P(x^{(n)})} \frac{P(y^{(2)} \wedge x^{(2)})}{P(x^2} \cdot \ldots$$

Compared to

$$\log\left(\prod_{i=1}^{n} P(y^{(i)} \mid x^{(i)})\right)$$

$$= \left(P(y^{(n)}, x^{(n)}) - P(x^{(n)})\right) + \ldots$$
$$+ \left(P(y^{(2)}, x^{(2)}) - P(x^{(2)})\right) +$$
$$P(y^{(1)}, x^{(1)}) - P(x^{(1)})$$

For the purpose of the gradient descent algorithm, (stochastic gradient descent) log-likelihood simplifies into a polynomial as compared to computing an exponential.