S21CSE471: Statistical Methods in AI **Spring 2021**

Logistic Regression

Prepared by: Yochan, Shubham, Srivathsa

Logistic regression is a classification technique. Classification problem is just like regression problem except that the target values y we now want to predict take on only a small number of discrete values.we will focus on the binary classification problem for now, in which y can take on only two values, 0 and 1. For example, if we are trying to build a spam classifier for email, then let $x^{(i)}$ may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. 0 is also called the **negative class**, and 1 the **positive** class, and they are sometimes also denoted by the symbols "-" and "+". Given $x^{(i)}$, the corresponding $y^{(i)}$ is also called the **label** for the training example.

1 Logistic Regression approach

We could approach the classification problem ignoring the fact that y is discrete-valued, and use our old linear regression algorithm to try to predict y given x. However, it is easy to construct examples where this method performs very poorly. y. Intuitively, it also doesn't make sense for $y^{(i)}$ to take values larger than 1 or smaller than 0 when we know that $y\epsilon\{0,1\}$. To fix this we use a function called **sigmoid function** or the **logistic function**.

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

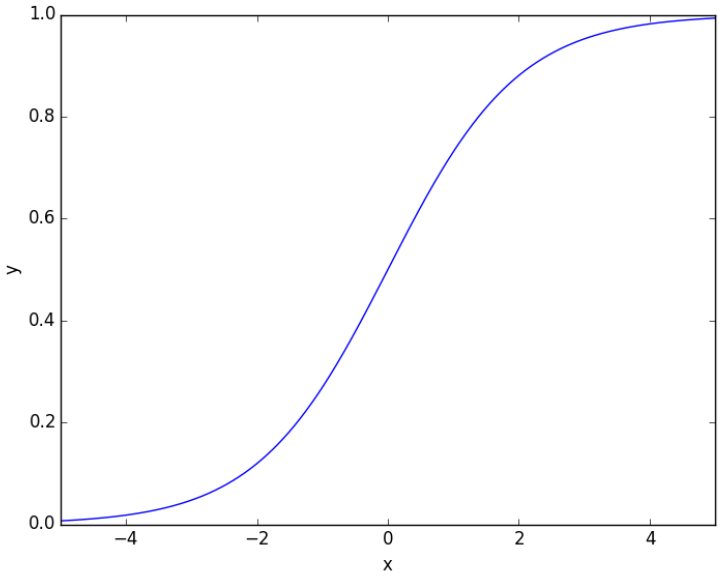

Figure 1: Sigmoid graph

Notice that $g(z)$ tends towards 1 as $z \to \infty$, and $g(z)$ tends towards 0 as $z \to -\infty$. This property of sigmoid i.e., mapping a real value to a output value(probability estimate) between 0 and 1 makes it

suitable for classification task. The derivative of sigmoid function we write as $g'(z)$.

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}(e^{-z}) \\ &= \frac{1}{(1+e^{-z})}\left(1-\frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)) \end{aligned} \tag{2}$$

Now $g(w^T x_i) = g(w^T x_i)\ (1 - g(w^T x_i)$. Instead of fitting a line like in linear regression $y = w^T x$ we now want fit $y = \frac{1}{1+e^{(-w^T x_i)}}$ in logistic regression. So from linear regression we have

$$J(w) = \sum_{i=1}^{n}(y_i - w^T x_i)^2 \tag{3}$$

The equivalent function is

$$J(w) = \sum_{i=1}^{n}\left(y_i - \frac{1}{1+e^{(-w^T x_i)}}\right)^2 \tag{4}$$

From here we go on and solve using differentiation like we did in linear regression, but the function is difficult because it not a convex function. We use maximum likelihood estimation to resolve the issue.

Let us assume that

$$\begin{aligned} P(y=1 \mid x_i; w) &= g(w^T x_i) \\ P(y=0 \mid x_i; w) &= 1 - g(w^T x_i) \end{aligned} \tag{5}$$

This can be written more compactly as

$$P(y_i \mid x_i; w) = \big(g(w^T x_i)\big)^{(y_i)}\big(1 - g(w^T x_i)\big)^{(1-y_i)} \tag{6}$$

We want to maximise likelihood $P(y \mid x_i; w)$ of each training sample i.e., whether the sample belongs to class 1 or class 0.

We want if y_i=0 then $g(w^T x_i)$ is close to 0 and if y_i=1 then $g(w^T x_i)$ is close to 1. This behaviour can be obtained we maximise equation 5, because when y_i=0 the first term will 1 and to maximise second term we need $g(w^T x_i)$ to be very small. Similarly when y_i=1 second term will become 1 and to maximise the equation we need $g(w^T x_i)$ to be close to 1.

Assuming that the n training examples were generated independently, we can then write down the likelihood of the parameters as

$$\begin{aligned} L(w) &= \prod_{i=1}^{n} P(y_i \mid x_i; w) \\ &= \prod_{i=1}^{n}\big(g(w^T x_i)\big)^{(y_i)}\big(1 - g(w^T x_i)\big)^{(1-y_i)} \end{aligned} \tag{7}$$

We want to maximise above equation with respect to w. It will be easier to maximize the log likelihood:

$$\begin{aligned} G(w) = \log L(w) &= \log \prod_{i=1}^{n}\big(g(w^T x_i)\big)^{(y_i)}\big(1 - g(w^T x_i)\big)^{(1-y_i)} \\ &= \sum_{i=1}^{n} y_i \log\big(g(w^T x_i)\big) + (1-y_i)\log\big(1 - g(w^T x_i)\big) \end{aligned} \tag{8}$$

Now we need to maximise the likelihood. Like in linear regression once we compute gradient, we use gradient descent. Similarly here we can use gradient ascent.) Let's start by working with just one training example (x_i, y_i), and take derivatives to derive the stochastic gradient ascent rule:

$$\begin{aligned} G'(w) &= \sum_{i=1}^{n} \left(y_i \frac{1}{g(w^T x_i)} - (1-y_i) \frac{1}{1-g(w^T x_i)} \right) g'(w^T x_i) x_i \\ &= \sum_{i=1}^{n} \left(y_i \frac{1}{g(w^T x_i)} - (1-y_i) \frac{1}{1-g(w^T x_i)} \right) g(w^T x_i)(1-g(w^T x_i)) x_i \\ &= \sum_{i=1}^{n} \Big(y_i(1-g(w^T x_i)) - (1-y_i)(g(w^T x_i)) \Big) x_i \\ &= \sum_{i=1}^{n} \big(y_i - g(w^T x_i) \big) x_i \end{aligned} \tag{9}$$

Above, we used the fact that $g'(z) = g(z)(1-g(z))$. This therefore gives us the stochastic gradient ascent rule

$$w_{t+1} = w_t + \lambda G'(w) \tag{10}$$

From here we can compute weights and solve equation 8. Note that we have used $z = w^T x_i$. When $z \to \infty$ then log(g(z)) becomes 0 so the first term from equation 8 becomes zero and second term might tend to infinity which means that our model will generalise only one class and though may perform better on train, will fail to perform on test data. This scenario is called overfitting. So like in linear regression we need to use regularisation to handle overfitting. The idea is to impose a penalty on the magnitude of the parameter values. This penalty should be minimized, in a trade-off with maximizing likelihood. Mathematically, the optimization problem to be solved is

$$w = \underset{w}{\operatorname{argmax}}\, L(w) - \mu \|w\|_2^2 \tag{11}$$

where $\|w\|_2^2 = \sum_{j=1}^{d} w_j^2$ is the squared L2 norm of the parameter vector w of length d. The constant μ quantifies the trade-off between maximizing likelihood and making parameter values be close to zero.

Stochastic gradient following is easily extended to include regularization. We simply include the penalty term when calculating the gradient for each example. Consider

$$\frac{\partial}{\partial w_j} \Big[\log L(w) - \mu \sum_{j=1}^{d} w_j^2 \Big] = \frac{\partial}{\partial w_j} [\log L(w)] - \mu 2 w_j \tag{12}$$

Remember that for logistic regression the partial derivative of the log conditional likelihood for one example is from equation 9 where $p = g(w^T x_i)$

$$\frac{\partial}{\partial w_j} [\log L(w)] = (y-p) x_j \tag{13}$$

so the update rule with regularization is

$$w_j = w_j + \lambda [(y-p) x_j - \mu 2 w_j] \tag{14}$$

where λ is the learning rate. Update rules like the one above are often called "weight decay" rules, since the weight w_j is made smaller at each update unless $y - p$ has the same sign as x_j. Straightforward stochastic gradient ascent for training a regularized logistic regression model loses the desirable sparsity

property described above, because the value of every parameter w_j must be decayed for every training example. Writing the regularized optimization problem for logistic regression as a minimization gives

$$w = \underset{w}{\operatorname{argmin}} - \sum_{i=1}^{n} \log L(w) + \sum_{j=1}^{d} w_j^2 \tag{15}$$

The term $-\log L(W)$ is called the "loss" for training example i.If the predicted probability, using w, of the true label y_i is close to 1, then the loss is small. But if the predicted probability of y_i is close to 0, then the loss is large. Losses are always non-negative; we want to minimize them.We also want to minimize the numerical magnitude of the trained parameters, because as the parameters increase complexity increases.

References

[1] Stanford cs229:https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

[2] Stanford: https://web.stanford.edu/ jurafsky/slp3/5.pdf

[3] USC :http://cseweb.ucsd.edu/ elkan/250B/logreg.pdf

[4] CMU:http://www.stat.cmu.edu/ ryantibs/advmethods/notes/logreg.pdf