Logistic Regression Notes – Aron Culotta – CMPS470

From Regression to Classification

Recall our model of regression with multiple variables:

$$h(x_i) = \sum_{j=1}^{k} x_{ij}\theta_j = \theta \cdot x_i$$

where x_{ij} is the value of feature j for instance i (e.g., "Julie has blonde hair") and θ are our model parameters. (The term $\theta \cdot x_i$ is the dot product, $\sum_j \theta_j * x_{ij}$.)

The RSS is:

$$RSS(h, D) = \frac{1}{2}\sum_{i=1}^{|D|}(y_i - \theta \cdot x_i)^2$$

The gradient we have to compute is with respect to one of the θ variables:

$$\frac{\partial RSS(h, D)}{\partial \theta_j} = \sum_{i=1}^{|D|}(y_i - \theta \cdot x_i)(-x_{ij})$$

The above assumes that the output variable y_i is a real number. Thus, this is a model of **regression**. When y is disrete, the problem is one of **classification**. One could use the linear model above to do classification. Assume the binary case where y_i can either be -1 or 1. We can convert the regression model to a classifier by assuming that if the model outputs a number greater than 0, then predict 1; otherwise, predict -1. However, this can cause weird things to happen in our update rule:

$$\theta_j^{t+1} = \theta_j^t + \sum_{i=1}^{|D|}(y_i - \theta \cdot x_i)x_{ij}$$

Suppose the true value of y_1 is 1, and the model (dot product) returns 2. The instance is technically classified correctly, but the update still counts this as an error of size 1 (since $y_i - \theta \cdot x_i$ is 1). In fact, the update considers this error equivalent to a dot product of 0, which would in fact result in a classification error.

This is clearly not what we want – in fact, our gradient descent algorithm may never converge, since we can always make our correct classifications "more correct" by cranking up the value of θ.

The way around this is to change our model. Rather than regression, we need classification. We can do this by passing the dot product $x_i \cdot \theta$ through a "squashing function" (the logistic function) that ensures its value is always between 0 and 1:

$$h(x_i) = \frac{1}{1 + e^{-x_i \cdot \theta}}$$

Because $h(x_i)$ will always be between 0 and 1, we have the right to call this a **probability** $p(y_i|x_i)$. This is the conditional probability of a label y_i given input x_i. For the binary case, assume y_i can be either -1 or 1. Then, we can write

$$p(y_i = 1|x) = \frac{1}{1 + e^{-x_i \cdot \theta}}$$

for the positive case and

$$p(y_i = -1|x) = 1 - p(y_i = 1|x) = \text{ by algebra } = \frac{1}{1 + e^{x_i \cdot \theta}}$$

for the negative case (note the negative sign is missing before x_i in the negative case).

If we assume $p(y_i|x_i)$ is the probability of **the true label** for instance x_i, we can write this as:

$$p(y_i|x_i) = \frac{1}{1 + e^{-y_i x_i \cdot \theta}}$$

This is the probability of x_i being labeled *correctly* by the classifier. Now, we can rephrase our learning objective as maximizing the *joint probability of the true labels for all training instances.* Since we assume each instance is drawn independently, we can write this joint probability as a product of individual probabilities:

$$p(y_1 \dots y_n|x_1 \dots x_n) = p(y_1|x_n) * p(y_2|x_2) * \dots * p(y_n|x_n) = \prod_{i=1}^{n} p(y_i|x_i)$$

Because we're used to minimizing functions using gradient descent, rather than maximizing the probability, we can instead minimize the negative probability. This is our new error function:

$$E(D, h) = -\prod_{i=1}^{n} p(y_i|x_i)$$

Note that this is very similar to RSS, but by using probabilities, we ensure that the output for each instance is always between 0 and 1.

Following our learning recipe, our next step is to minimize $E(D, h)$ using gradient descent. Computing the gradient of $E(D, h)$ in its current form is rather hard. So, we can simply transform it to something that's easier to take the gradient of:

$$E(D, h) = -\ln \prod_{i=1}^{n} p(y_i|x_i)$$

where ln is the natural log. This equation is called the **negative log likelihood**. It turns out that minimizing $f(x)$ or $\ln f(x)$ results in the same answer, so we can make this transformation without affecting our final solution.

Now we're ready to calculate the gradient with respect to one parameter θ_j (close your eyes if necessary):

$$\begin{aligned}
\frac{\partial E(D,h)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} - \ln \prod_i \frac{1}{1 + e^{-y_i x_i \cdot \theta}} \\
&= \frac{\partial}{\partial \theta_j} - \sum_i \ln \frac{1}{1 + e^{-y_i x_i \cdot \theta}} \quad \text{(by definition of log of products)} \\
&= -\sum_i 1 + e^{-y_i x_i \cdot \theta} \frac{\partial}{\partial \theta_j} \frac{1}{1 + e^{-y_i x_i \cdot \theta}} \quad \left(\text{by } \frac{d}{dx} \ln(f(x)) = \frac{1}{f(x)} \frac{d}{dx} f(x)\right) \\
&= -\sum_i (1 + e^{-y_i x_i \cdot \theta}) \left(\frac{-y_i x_{ij} e^{-y_i x_i \cdot \theta}}{(1 + e^{-y_i x_i \cdot \theta})^2} \right) \quad \text{(by quotient and chain rules)} \\
&= -\sum_i \frac{-y_i x_{ij} e^{-y_i x_i \cdot \theta}}{1 + e^{-y_i x_i \cdot \theta}} \quad \text{(by algebra)} \\
&= \sum_i y_i x_{ij} (1 - p(y_i|x_i)) \quad \left(\text{by } \frac{e^{-y_i x_i \cdot \theta}}{1 + e^{-y_i x_i \cdot \theta}} = 1 - p(y_i|x_i)\right)
\end{aligned}$$

Thus, the final logistic regression update is:

$$\theta_j^{t+1} \leftarrow \theta_j^t + \eta \sum_i y_i x_{ij} (1 - p(y_i|x_i))$$

Compare this with the update on the previous page, and you can see that we have solved the problem of the unbounded update.

Multiclass Logistic Regression

Here, we'll extend the above for the multi-class case. Rather than a single parameter vector θ, we will have m vectors, one per class: $\theta = \{\theta^{(1)} \dots \theta^{(m)}\}$. The probability of a particular class then becomes:

$$p(y_i = r|x_i) = \frac{e^{x_i \cdot \theta^{(r)}}}{\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}}}$$

To be more clear, we'll let y_i^* be the true label for instance x_i. The error function is then:

$$E(D, h) = -\prod_{i=1}^{n} p(y_i = y^*|x_i)$$

$$\begin{aligned}
\frac{\partial E(D,h)}{\partial \theta_j^{(r)}} &= \frac{\partial}{\partial \theta_j^{(r)}} - \ln \prod_i \frac{e^{x_i \cdot \theta^{(y^*)}}}{\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}}} \\
&= \frac{\partial}{\partial \theta_j^{(r)}} - \sum_i \ln \frac{e^{x_i \cdot \theta^{(y^*)}}}{\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}}}
\end{aligned}$$

The numerator in the previous equation is a constant if $y^* \neq r$. So, we'll compute two derivatives. First, if $y^* = r$:

$$\begin{aligned}
\frac{\partial E(D,h)}{\partial \theta_j^{(r)}} &= \sum_i \frac{1}{p(y_i = y^*|x_i)} \frac{\partial}{\partial \theta_j^{(r)}} p(y_i = y^*|x_i) \\
&= \sum_i \frac{1}{p(y_i = y^*|x_i)} \frac{x_{ij} e^{x_i \theta^{(y^*)}} \sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}} - x_{ij} e^{x_i \cdot \theta^{(y^*)}} e^{x_i \cdot \theta^{(r)}}}{(\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}})^2} \quad \text{(by quotient and chain rules)} \\
&= \sum_i \frac{p(y_i = y^*|x_i) x_{ij} (\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}} - e^{x_i \cdot \theta^{(r)}})}{p(y_i = y^*|x_i) \sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}}} \quad \text{(by algebra)} \\
&= \sum_i x_{ij} (1 - p(y_i = r|x_i)) \\
&= \sum_i x_{ij} - x_{ij} p(y_i = r|x_i)
\end{aligned}$$

And we do similar steps when $y^* \neq r$:

$$\begin{aligned}
\frac{\partial E(D,h)}{\partial \theta_j^{(r)}} &= \sum_i \frac{1}{p(y_i = y^*|x_i)} \frac{\partial}{\partial \theta_j^{(r)}} p(y_i = y^*|x_i) \\
&= \sum_i \frac{1}{p(y_i = y^*|x_i)} \frac{-x_{ij} e^{x_i \cdot \theta^{(r)}} e^{x_i \cdot \theta^{y^*}}}{(\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}})^2} \quad \text{(by reciprocal and chain rules)} \\
&= \sum_i \frac{p(y_i = y^*|x_i)}{p(y_i = y^*|x_i)} \frac{-x_{ij} e^{x_i \cdot \theta^{(r)}}}{\sum_{k=1}^{m} e^{x_i \cdot \theta^{(k)}}} \quad \text{(by algebra)} \\
&= \sum_i -x_{ij} p(y_i = r|x_i)
\end{aligned}$$