# Kernel Logistic Regression

**Boluwatife E. Awoyemi**
Department of Mathematics and Statistics, Texas Tech University, 2500 Broadway, Lubbock, TX 79409, US

## Description of Problem

In this project, we will be using the gradient descent method and ultimately the Kernel method on the logistic regression model. The application of Kernel method makes this model a Kernel Logistic Regression model.

The KLR models to be evaluated is

$$P(y_i = 1) = P(x_i) = \frac{1}{1 + e^{-w^T \phi(x_i)}} \quad \text{Binary Classification (BC)}$$

$$P(y_i = j) = \frac{e^{w_j^T \phi(x_i)}}{\sum_{l=1}^{k} e^{w_l^T \phi(x_i)}}, \quad j = 1, \cdots, k \quad \text{Multi-class Classification (McC)}$$

We aim to use these methods to find the optimal parameters - weights ($\mathbf{w}$) and intercept ($b$) of the logistic regression models by minimizing the cross entropy function for random specific data sets.

We also hope to compare both methods to see if we get the same results or different results after computation.

# Method

## Solutions and Algorithms (BC)

### Cross Entropy Function

$$
\begin{aligned}
L &= -\sum_{i=1}^{n} \left(y_i \log p(x_i) + (1-y_i) log(1-p(x_i))\right) \\
L(w) &= -\sum_{i=1}^{n} \left[ y_i \log \left( \frac{1}{1+e^{-w^T\phi(x_i)}} \right) + (1-y_i) \log \left( 1 - \frac{1}{1+e^{-w^T\phi(x_i)}} \right) \right] \\
&= -\sum_{i=1}^{n} \left[ y_i \log \left( \frac{1}{1+e^{-w^T\phi(x_i)}} \right) + (1-y_i) \log \left( \frac{e^{-w^T\phi(x_i)}}{1+e^{-w^T\phi(x_i)}} \right) \right] \\
&= -\sum_{i=1}^{n} y_i \left(0 - \log\left(1+e^{-w^T\phi(x_i)}\right)\right) + (1-y_i)\left[-w^T\phi(x_i) - \log\left(1+e^{-w^T\phi(x_i)}\right)\right] \qquad \text{by laws of log} \\
&= -\sum_{i=1}^{n} \left(-w^T\phi(x_i) - \log\left(1+e^{-w^T\phi(x_i)}\right) + y_i w^T\phi(x_i)\right) \\
&= \sum_{i=1}^{n} \left(w^T\phi(x_i) + \log\left(1+e^{-w^T\phi(x_i)}\right) - y_i w^T\phi(x_i)\right) \\
&= \sum_{i=1}^{n} \left((1-y_i)w^T\phi(x_i) + \log\left(1+e^{-w^T\phi(x_i)}\right)\right)
\end{aligned}
$$

### Gradient of the Cross Entropy Function

$$
\begin{aligned}
\nabla_w L(w) &= \sum_{i=1}^{n} \frac{\partial}{\partial w}\left[(1-y_i)w^T\phi(x_i)\right] + \frac{\partial}{\partial w}\left[\log\left(1+e^{-w^T\phi(x_i)}\right)\right] \\
&= \sum_{i=1}^{n} \left( (1-y_i)\phi(x_i) - \frac{\phi(x_i)e^{-w^T\phi(x_i)}}{1+e^{-w^T\phi(x_i)}} \right) \\
&= \sum_{i=1}^{n} \left( (1-y_i)\phi(x_i) - \phi(x_i)p(x_i)\cdot\frac{1-p(x_i)}{p(x_i)} \right) \qquad \text{where } e^{-w^T\phi(x_i)} = \frac{1-p(x_i)}{p(x_i)} \\
&= \sum_{i=1}^{n} \left( (1-y_i)\phi(x_i) - \phi(x_i)(1-p(x_i)) \right) \\
&= \sum_{i=1}^{n} (1-y_i-1-p(x_i))\phi(x_i) \\
&= \sum_{i=1}^{n} (p(x_i)-y_i)\phi(x_i).
\end{aligned}
$$

□

### Derivation of the iterative algorithm for the KLR model (BC)

$$
\begin{aligned}
\nabla_w L(w) &= \sum_{i=1}^{n}(p(x_i) - y_i)\phi(x_i) \\
w^{(k+1)} &= w^{(k)} - \gamma \nabla_w L(w) \\
&= w^{(k)} + \gamma \sum_{i=1}^{n} \left( y_i - \frac{1}{1 + e^{-w^T \phi(x_i)}} \right) \phi(x_i) \\
w^{(k)} &= \sum_{i=1}^{n} \alpha_i^{(k)} \phi(x_i) \\
\implies w^{(k+1)} &= \sum_{i=1}^{n} \alpha_i^{(k)} \phi(x_i) + \gamma \sum_{i=1}^{n} \left( y_i - \frac{1}{1 + e^{-\left(\sum_{j=1}^{n} \alpha_j^{(k)} \phi(x_i)\right)^T \phi(x_i)}} \right) \phi(x_i) \\
&\text{Recall that } K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \quad \text{Kernel function} \\
\implies w^{(k+1)} &= \sum_{i=1}^{n} \alpha_i^{(k)} \phi(x_i) + \gamma \sum_{i=1}^{n} \left( y_i - \frac{1}{1 + e^{-\sum_{j=1}^{n} \alpha_j^{(k)} K}} \right) \phi(x_i) \\
&= \sum_{i=1}^{n} \left[ \alpha_i^{(k)} + \gamma \left( y_i - \frac{1}{1 + e^{-\sum_{j=1}^{n} \alpha_j^{(k)} K}} \right) \right] \phi(x_i) \\
\implies \alpha_i^{(k+1)} &= \alpha_i^{(k)} + \gamma \left( y_i - \frac{1}{1 + e^{-\sum_{j=1}^{n} \alpha_j^{(k)} K}} \right).
\end{aligned}
$$

□

## Solutions and Algorithms (McC)

### Cross Entropy Function

$$
\begin{aligned}
L(w_1, \cdots, w_k) &= -\sum_{i=1}^{n}\sum_{j=1}^{n} \chi(y_i = j) \cdot \log p(y_i = j) \\
&\text{where } \chi(y_i = j) = \begin{cases} 1, & if\, y_i = j \\ 0, & \text{otherwise} \end{cases} \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{n} \chi(y_i = j) \cdot \log \left( \frac{e^{w_j^T \phi(x_i)}}{\sum_{l=1}^{k} e^{w_l^T \phi(x_i)}} \right) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{n} \chi(y_i = j) \cdot \left[ \log \left( e^{w_j^T \phi(x_i)} \right) - \log \left( \sum_{l=1}^{k} e^{w_l^T \phi(x_i)} \right) \right] \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{n} \chi(y_i = j) \cdot w_j^T \phi(x_i) + \sum_{i=1}^{n}\sum_{j=1}^{n} \chi(y_i = j) \cdot \log \left( \sum_{l=1}^{k} e^{w_l^T \phi(x_i)} \right)
\end{aligned}
$$

## Gradient of the Cross Entropy Function

$$
\begin{aligned}
\nabla_w L(w_m) &= -\sum_{i=1}^{n}\sum_{j=1}^{n}\chi(y_i=j)\frac{\partial}{\partial w_m}\left(w_j^T\phi(x_i)\right) + \sum_{i=1}^{n}\sum_{j=1}^{n}\chi(y_i=j)\frac{\partial}{\partial w_m}\left(\log\left(\sum_{l=1}^{k}e^{w_l^T\phi(x_i)}\right)\right) \\
&= -\sum_{i=1}^{n}\chi(y_i=m)\phi(x_i) + \sum_{i=1}^{n}\sum_{j=1}^{n}\chi(y_i=j)\sum_{j=1}^{n}\chi(y_i=j)\left(\phi(x_i)\frac{e^{w_m^T\phi(x_i)}}{\sum_{l=1}^{k}e^{w_l^T\phi(x_i)}}\right) \\
&\text{where } \frac{\partial}{\partial w_m}\left(w_j^T\phi(x_i)\right) = \begin{cases}\phi(x_i), & if m=j\\ 0, & \text{otherwise}\end{cases} \\
&= -\sum_{i=1}^{n}\chi(y_i=m)\phi(x_i) + \sum_{i=1}^{n}\sum_{j=1}^{n}\chi(y_i=j)\phi(x_i)p(y_i=m) \\
&= -\sum_{i=1}^{n}\chi(y_i=m)\phi(x_i) + \sum_{i=1}^{n}\phi(x_i)p(y_i=m) \\
&\text{where } \sum_{i=1}^{n}\chi(y_i=j)=1 \\
&= -\sum_{i=1}^{n}\left[\chi(y_i=m)-p(y_i=m)\right]\phi(x_i).
\end{aligned}
$$

□

### Derivation of the iterative algorithm for the KLR model

$$
\begin{aligned}
\nabla w_m L(w_m) &= -\sum_{i=1}^{n} \left[\chi(y_i = m) - p(y_i = m)\right] \phi(x_i) \\
&= -\sum_{i=1}^{n} \left[\chi(y_i = m) - \frac{e^{w_j^T \phi(x_i)}}{\sum_{l=1}^{k} e^{w_l^T \phi(x_i)}}\right] \phi(x_i) \\
w_m^{(k+1)} &= w_m^{(k)} - \gamma \nabla w_m L(w_m^{(}k)) \\
&= w^{(k)} + \gamma \sum_{i=1}^{n} \left[\chi(y_i = m) - \frac{e^{w_j^T \phi(x_i)}}{\sum_{l=1}^{k} e^{w_l^T \phi(x_i)}}\right] \phi(x_i) \\
w_m^{(k)} &= \sum_{i=1}^{n} \alpha_{mi}^{(k)} \phi(x_i) \\
\implies w_m^{(k+1)} &= \sum_{i=1}^{n} \alpha_{mi}^{(k)} \phi(x_i) + \gamma \sum_{i=1}^{n} \left[\chi(y_i = m) - \frac{e^{\left(\sum_{j=1}^{n} \alpha_{mj}^{(k)} \phi(x_j)\right)^T \phi(x_i)}}{\sum_{l=1}^{k} e^{\left(\sum_{j=1}^{n} \alpha_{lj}^{(k)} \phi(x_j)\right)^T \phi(x_i)}}\right] \phi(x_i)
\end{aligned}
$$

Recall that $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ Kernel function

$$
\begin{aligned}
\implies w_m^{(k+1)} &= \sum_{i=1}^{n} \alpha_{mi}^{(k)} \phi(x_i) + \gamma \sum_{i=1}^{n} \left[\chi(y_i = m) - \frac{e^{\sum_{j=1}^{n} \alpha_{mj}^{(k)} K(x_j, x_i)}}{\sum_{l=1}^{k} e^{\sum_{j=1}^{n} \alpha_{lj}^{(k)} K(x_j, x_i)}}\right] \phi(x_i) \\
&= \sum_{i=1}^{n} \left[\alpha_{mi}^{(k)} + \left(\gamma \chi(y_i = m) - \frac{e^{\sum_{j=1}^{n} \alpha_{mj}^{(k)} K(x_j, x_i)}}{\sum_{l=1}^{k} e^{\sum_{j=1}^{n} \alpha_{lj}^{(k)} K(x_j, x_i)}}\right)\right] \phi(x_i) \\
\implies \alpha_{mi}^{(k+1)} &= \alpha_{mi}^{(k)} + \gamma \left(\chi(y_i = m) - \frac{e^{\sum_{j=1}^{n} \alpha_{mj}^{(k)} K}}{\sum_{l=1}^{k} e^{\sum_{j=1}^{n} \alpha_{lj}^{(k)} K}}\right)
\end{aligned}
$$

Now, let $\alpha = \alpha_m = \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{Bmatrix}$ where $m = 1, 2, \cdots, k$

Then,

$$
\alpha_i^{(k+1)} = \alpha_i^{(k)} + \gamma \left(\chi(y_i = m) - \frac{e^{\sum_{j=1}^{n} \alpha_j^{(k)} K}}{\sum_{l=1}^{k} e^{\sum_{j=1}^{n} \alpha_j^{(k)} K}}\right).
$$

□

# Results & Discussion

For all the data sets I was able to explore, I found out that the algorithms gave the same results in terms of accuracy. The weights of the two methods only differed by values in the last 3 to 4 decimal places, therefore the accuracy giving the same results is very reasonable.
I was not able to get accuracy of 90% and above in the two data sets I was able to explore. I believe this was because of my choice of step size and number of iterations, that is, a smaller step coupled with higher number of iterations could have gotten me to better accuracy.

**Table 1: Accuracy results of each method explored**

| **Data sets** | **Gradient Descent Method (Kernels)** | **Kernel Method (Kernels)** | **Step size** ($\gamma$) | Iterations |
|---|---|---|---|---|
| Random (BC) | 0.966 (2) | 0.966 (2) | 0.01 | 100000 |
| Make moons (BC) [1] | 0.887(3)—0.886(4)—0.886(5) | 0.887(3)—0.886(4)—0.886(5) | 0.01 | 100000 |
| Make circles (BC) [1] | 0.725(3)—0.734(4)—0.729(5) | 0.725(3)—0.734(4)—0.729(5) | 0.01 | 100000 |
| Iris (McC) [1] | Nil | Nil | 0.01 | 100000 |

Lastly, in terms of the number of kernels, 3 kernels was shown to be the best choice for the make moons data set while 4 kernels was the best choice for the make circles data set based on the kernels I used in this work. There is a chance that there are better kernel number options that I can not see due to the fact that I used only 3 options.
Finally, I was not able to get results for the Multi class classification model because I ran into a lot of problems during coding.

# Bibliography

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.