# LOGISTIC REGRESSION and KERNEL LOGISTIC REGRESSION

A comparative study of logistic regression and kernel logistic regression

Ezukwoke K.I[1], Samaneh Z.[2]

[1, 2] Department of Computer Science
Machine Learning and Data Mining
{ifeanyi.ezukwoke, samaneh.zareian.jahromi}@etu.univ-st-etienne.fr
University Jean Monnet, Saint-Etienne, France

**Abstract**

Logistic regression is a binary classification algorithm frequently used for classification problems.

**Keywords**
Classification, logistic regression, kernel logistic regression, multi-kernel learning.

## 1 INTRODUCTION

Linear regression is a statistical method used for univariate and multivariate analysis. Given a set of observations $\{x_i, y_i\}^n$, where $\{x_i\}^N$ is the independent variables ($featurespace$) and $y_i$ is the dependent (*response variable*- usually continuous is discrete), Linear regression models an estimate parameter $\beta$ that best maps the predictors to the response variable $y_i$.

$$y = X_1\beta_1 + X_2\beta_2 + \cdots + X_N\beta_N \tag{1}$$

Using Ordinary Least Squares (**OLS**) we can estimate the unknown parameters in the linear regression problem [1]. It does this by minimizing the sum of square differences between the predictors and the response variable.

$$\min_{\beta} \|X\beta - y\|_2 \tag{2}$$

We minimize this objective function using maximum likelihood estimation and derive the following closed form solution.

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{3}$$

This results in a model that produces as straight line that maps the predictors to the response [1].

However, linear regression is only sufficient for explaining the relationship in observations with continuous variable. For observations with categorical variables it becomes impossible to adopt this model, hence **logistic regression**.

Logistic regression solves the limitation of linear regression for categorical variable using maximum likelihood estimation of probability log function. This idea is further explained in the next sections. Our focus however is on its kernel version and how we explore the inner product of the independent variable to classify non-seperable data.

## 2 CLASSIFICATION

Classification is a supervised machine learning approach to categorizing data into **distinct number of classes** where we can assign label to each class. Given a set of data $\{x^{(i)}, y^{(i)}\}$ where x is the feature space in $m \times (n+1)$ dimension, $y$ is the classfication output such $y \in \{0, 1\}$ for binary output or $\{1, 2, ...n\}$ for multiclass output. Classification algorithms are most used for Spam detection, Voice and image recognition, sentiment analysis, fraud detection and many more.

Logistic regression is a linear binary classification algorithm that maps a set of predictors to their corresponding categorical response variables. The algorithm is capable of classifying linearly separable dataset. The linear version of logistic regression is however not able to accurately classify non-linear data, hence its kernel version.

Kernel logistic regression is similar to support vector machines in its operational output [4]. Existing papers already implement kernel logistic regression using Newton-Ralphson method [3], Sequential Minimal Optimization (**SMO**) [4] and Truncated Newton-method [5].

In this paper however we solve logistic regression and kernel-logistic regression using using gradient descent (**GD**) and stochastic gradient descent (**SGD**) optimization techniques.

## 2.1 LOGISTIC REGRESSION

Logistic regression is a discriminative model since it focuses only on the posterior probability of each class $Pr(Y|x;\beta)$. It is also a generalized linear model, mapping output of linear multiple regression to posterior probability of each class $Pr(Y|x;\beta) \in \{0,1\}$ [2]. probabilty a data-sample belongs to class 1 is given by

$$Pr(Y=1|X=x;\beta) = \sigma(z), where\ z = \beta^T x \tag{4}$$

$$P(Y=1|X=x;\beta) = \sigma(\beta^T x) \tag{5}$$

where

$$Pr(Y=1|X=x;\beta) + Pr(Y=0|X=x;\beta) = 1 \tag{6}$$

$$Pr(Y=0|X=x;\beta) = 1 - Pr(Y=1|X=x;\beta) \tag{7}$$

Hence, probability that a data-sample belongs to class 0 is given by:

$$Pr(Y=0|X=x;\beta) = 1 - \sigma(z) \tag{8}$$

$\sigma(z)$ is called the **logistic sigmoid function** and is give by

$$\sigma(z) = \frac{1}{1+\exp^{-z}} \tag{9}$$

The uniqueness of this function is that it maps all real numbers $\mathbb{R}$ to range $\{\ 0,\ 1\}$.

Again, we know

$$\begin{aligned} &log(odds(Pr(Y=1|X=x;\beta))) \\ &= \frac{Pr(Y=1|X=x;\beta)}{Pr(Y=0|X=x;\beta)} \\ &= \frac{Pr(Y=1|X=x;\beta)}{1-Pr(Y=1|X=x;\beta)} \end{aligned}$$

Assuming $P(Y=1|X=x;\beta) = p(x)$, the next most obvious idea is to let $logp(x)$ be a linear function of x, so that changing an input variable multiplies the probability by a fixed amount. This is done by taking a $log$ transformation of $p(x)$.

Formally, $logit(p(x)) = \beta_0 + \beta^T x$ making

$$logit(p(x)) = log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta^T x \tag{10}$$

Simplifying for $p(x)$ and $1-p(x)$ we have

$$\frac{p(x)}{1-p(x)} = \exp(\beta_0 + \beta^T x) \tag{11}$$

$$p(x) = (1-p(x))\exp(\beta_0 + \beta^T x) \tag{12}$$

$$p(x) = \exp(\beta_0 + \beta^T x) - p(x)\cdot\exp(\beta_0 + \beta^T x) \tag{13}$$

$$p(x) + p(x)\cdot\exp(\beta_0 + \beta^T x) = \exp(\beta_0 + \beta^T x) \tag{14}$$

$$p(x)(1+\exp(\beta_0 + \beta^T x)) = \exp(\beta_0 + \beta^T x) \tag{15}$$

$$\begin{aligned} p(x) &= \frac{\exp(\beta_0 + \beta^T x)}{1+\exp(\beta_0 + \beta^T x)} \\ &= \frac{\frac{1}{exp(\beta_0+\beta^T x)}\cdot\exp(\beta_0 + \beta^T x)}{\frac{1}{exp(\beta_0+\beta^T x)} + \frac{exp(\beta_0+\beta^T x)}{exp(\beta_0+\beta^T x)}} \\ &= \frac{1}{1+\exp-(\beta_0 + \beta^T x)} \end{aligned}$$

$$1 - p(x) = \frac{1}{1+\exp(\beta_0 + \beta^T x)} \tag{16}$$

## 2.2 Learning Logistic regression

Assume that $P(Y = 1|X = x; \beta) = P(x; \beta)$, for some function $p$ parameterized by $\beta$. Further assume that observations are independent of each other. The conditional likelihood function is given by Bernoulli sequence:

$$\Pi_{i=1}^{n} Pr(Y = y_i|X = x_i; \beta) = \Pi_{i=1}^{n} p(x_i; \beta)^{y} (1 - p(x_i; \beta)^{(1-y_i)})$$

The probability of a class is $p$, if $y_i = 1$, or $1 - p$, if $y_i = 0$. The likelihood is then

$$L(\beta_0, \beta) = \Pi_{i=1}^{n} p(x_i)^{y} (1 - p(x_i)^{(1-y_i)}) \quad (17)$$

taking the log of this likelihood we have

$$l(\beta_0, \beta) = \sum_{i=1}^{n} y_i log p(x_i) + (1 - y_i) log(1 - p(x_i))$$

$$= \sum_{i=1}^{n} y_i log p(x_i) - y_i log(1 - p(x_i)) + log(1 - p(x_i))$$

$$= \sum_{i=1}^{n} log(1 - p(x_i)) + y_i log \left( \frac{p(x_i)}{1 - p(x_i)} \right) \quad (18)$$

replace $log \left( \frac{p(x)}{1-p(x)} \right)$ with $\beta_0 + x \cdot \beta$ as seen in equation (8) and $(1 - p(x))$ with $\frac{1}{1+\exp(\beta_0 + x \cdot \beta)}$. Hence,

$$l(\beta_0, \beta) = \sum_{i=i}^{n} log \left( \frac{1}{1 + \exp(\beta_0 + x \cdot \beta)} \right) + y(\beta_0 + x \cdot \beta)$$

$$= \sum_{i=i}^{n} -log(1 + \exp(\beta_0 + x \cdot \beta)) + y(\beta_0 + x \cdot \beta) \quad (19)$$

$$\nabla l(\beta_0, \beta) = -\sum_{i=1}^{n} \frac{1}{1 + \exp(\beta_0 + x \cdot \beta)} x_{ij} \exp(\beta_0 + x \cdot \beta) + \sum_{i=1}^{n} y_i x_i$$

$$\nabla l_\beta = \sum_{i=1}^{n} (y_i - p(x_i; \beta_0, \beta)) x_{ij} \quad (20)$$

Since this is a transcendental equation with no closed-form solution, we apply the gradient ascent optimization algorithm.

**Algorithm 1:** Logistic regression via Gradient Descent **GD**

```
  Input  : $x \in \mathcal{X}$ where $y \in \mathcal{Y}$
  Output : $\beta$
1 begin
2     $\beta_j \leftarrow \beta^0$;
3     while not converged do
4         $\beta_{j+1} = \beta_j - \alpha \nabla_\beta l$;
5         where $\nabla_\beta l =$
            $\sum_{i=1}^{n} (y_i - p(x_i; \beta_0, \beta)) x_{ij}$
6     end
7     return $\beta$
8 end
```

We also solve it using stochastic approach

**Algorithm 2:** Logistic regression via Stochastic Gradient Descent **SGD**

```
   Input  : $x \in \mathcal{X}$ where $y \in \mathcal{Y}$
   Output : $\beta$
 1 begin
 2     $\beta_j \leftarrow \beta^0$;
 3     while not converged do
 4         for $i \in$ randshuffle($\{1, \ldots, N\}$)
             do
 5             for $k \in \{1, \ldots, K\}$ do
 6                 $\beta_k = \beta_k - \alpha \nabla_{beta} l$;
 7                 where $\nabla_\beta l = \sum_{i=1}^{n} (y_i -$
                     $p(x_i; \beta_0, \beta)) x_{ij}^{k}$
 8             end
 9         end
10     end
11     return $\beta$
12 end
```

## 2.3 KERNEL LOGISTIC REGRESSION

However, classical logistic regression will fail to classify accurately non-linearly separable data, hence its kernel version. It also has a direct probabilistic interpretation that makes it suited for Bayesian design [4].

The vector space can be expressed as a linear combination of the input vectors such that

$$\beta = \sum_{i=1}^{N} \alpha_i \phi(\mathbf{x_i}) \tag{21}$$

where $\alpha \in \mathbb{R}^{nx1}$ is the dual variable. The function $\phi(x_i)$ maps the data points from lower dimension to higher dimension.

$$\phi : \mathbf{x} \in \mathbb{R}^{\mathbb{D}} \rightarrow \phi(x) \in \mathbb{F} \subset \mathbb{R}^{\mathbb{D}'} \tag{22}$$

Let $\kappa(x_i, x)$ be a kernel function resulting from the inner product of $\phi(x_i)$ and $\phi(\mathbf{x_j})$, such that

$$\kappa(x_i, x) = \langle \phi(x_i)\phi(x_j) \rangle \tag{23}$$

From **representer theorem** we know that

$$\begin{aligned} F = \beta^T \phi(x) &= \alpha \langle \phi(\mathbf{x_i})\phi(\mathbf{x_j}) \rangle \\ &= \alpha\kappa(x_i, x_j) \end{aligned}$$

We can now express $p(x;\beta)$ is subspace of input vectors only such that

$$p(\phi;\alpha) = \frac{1}{1 + e^{-\alpha_i \kappa(x_i, x_j)}} \tag{24}$$

and

$$1 - p(\phi;\alpha) = \frac{1}{1 + e^{\alpha_i \kappa(x_i, x_j)}} \tag{25}$$

The logit function is mapped into the kernel space as

$$logit(\frac{p(\phi;\alpha)}{1 - p(\phi;\alpha)}) = \alpha\kappa(x_i, x) \tag{26}$$

Deriving the equation of kernel logistic regression requires the regularized logistic regression, precisely the regularized $l2 - norm$ of the log-likelihood. This is in comparison to the SVM objective function used in [3].

$$\begin{aligned} L_\alpha = \sum_{i=1}^{n} y_i logp(x_i) + (1 - y_i) log(1 - p(x_i)) \\ -\frac{\lambda}{2}\alpha^{\mathbf{T}} \kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

## 2.4 Learning kernel logistic regression

As mentioned earlier, some of the methods for finding the maximum likelihood estimate include gradient descent (**GD**), iterative reweighted least sqaures (**IRLS**) method. Here we employ the use of IRLS which is based on the Newton-Ralphson algorithm.

### 2.4.1 Optimization problem

$$\begin{aligned} L_\alpha = \sum_{i=1}^{n} y_i logp(x_i) + (1 - y_i) log(1 - \\ p(x_i)) - \frac{\lambda}{2}\alpha^{\mathbf{T}} \kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

We can expand the objective function as follows

$$\begin{aligned} L_\alpha &= ylog\left(\frac{p}{1-p}\right) + log(1 - p(x_i)) \\ &\quad -\frac{\lambda}{2}\alpha^{\mathbf{T}} \kappa(\mathbf{x_i}, \mathbf{x})\alpha \\ &= ylog\left(\frac{p}{1-p}\right) + log\left(\frac{1}{1 + e^{\alpha\kappa(x_i, x)}}\right) \\ &\quad -\frac{\lambda}{2}\alpha^{\mathbf{T}} \kappa(\mathbf{x_i}, \mathbf{x})\alpha \\ &= y\alpha\kappa(x_i, x) - log(1 + e^{\alpha\kappa(x_i, x)}) \\ &\quad -\frac{\lambda}{2}\alpha^{\mathbf{T}} \kappa(\mathbf{x_i}, \mathbf{x})\alpha \end{aligned}$$

First order derivative of the log-likelihood

$$\begin{aligned} \nabla_\alpha L &= y\kappa(x_i, x) - \frac{\kappa(x_i, x) e^{\alpha\kappa(x_i, x)}}{1 + e^{\alpha\kappa(x_i, x)}} \\ &\quad -\lambda\alpha\kappa(x_i, x) \\ &= y\kappa(x_i, x) - p\kappa(x_i, x) - \lambda\alpha\kappa(x_i, x) \\ \nabla_\alpha L &= \kappa(x_i, x)(y - p) - \lambda\alpha\kappa(x_i, x) \end{aligned}$$

### 2.4.2 Prediction

Still using the representer theorem, we compute the posterior probability of a new data point such that

$$y = sign\left(\frac{1}{1 + \exp^{-\alpha\kappa(x_i, x)}}\right) \tag{27}$$

Here, the prediction is dependent only on $\alpha$ and the inner product of the training and test data.

**Algorithm 3:** KLR using **Gradient descent**

```
   Input  : $\kappa, y, \alpha_j$
   Output : $\alpha$
1  begin
2      $\alpha_j \leftarrow \alpha^{0}$';
3      while not converged do
4          $\alpha_{j+1} = \alpha_j - lr\nabla_{\alpha}L$;
5          where $\nabla_{\alpha}L = \kappa(x_i, x_j)(y - p) - \lambda\alpha\kappa(x_i, x_j)$
6      end
7      return $\alpha$
8  end
```

In algorithm 4 of Algorithm 3, $lr$ is the learning rate.

**Algorithm 4:** KLR using **Stochastic Gradient descent**

```
   Input  : $\kappa, y, \alpha_j$
   Output : $\alpha$
1  begin
2      $\alpha_j \leftarrow \alpha^{0}$';
3      while not converged do
4          for $i \in$ randshuffle($\{1, \dots, N\}$) do
5              for $k \in \{1, \dots, K\}$ do
6                  $\alpha_{j+1} = \alpha_j - lr\nabla_{\alpha}L$;
7                  where $\nabla_{\alpha}L = \kappa(x_i, x_j)(y - p) - \lambda\alpha\kappa(x_i, x_j)$;
8              end
9          end
10     end
11     return $\alpha$
12 end
```

## 2.5 Muli-kernels

The reason behind the use of multiple kernels is similar to the notion of multiclassification, where cross-validation is used to select the best performing classifier [6]. By using multiple kernels, we hope to learn a different similarity uncovered using single kernels.

We can prove from **Mercer's Theorem** that a kernel is **Positive Definite (PD)** if $\kappa(x_i, x_j) \geq 0$. Hence by performing arithmetic or any mathematical operation on two or more kernel matrix, we obtain a new kernel capable of exploiting different property or similarities of training data.

Given a kernel $\kappa$, we prove that $\kappa$ is PD if

$$\langle u, \kappa u \rangle \geq 0 \tag{28}$$

**Proposition**: *A symmetric function* $\kappa$*:* $\times\ \chi \rightarrow \mathbb{R}$ *is positive definite* $\iff \kappa$
*Proof*:
Suppose that $\kappa$ is a kernel which is the inner product of the mapping functions $\langle \phi(x_i)\phi(x_j) \rangle$. $\kappa$ is a kernel if its inner product are positive and the solution of $\kappa u = \lambda u$ gives non-negative eigenvalues.

So that,

$$\langle u, \kappa u \rangle = \sum_{i=1}^{N} u_i(\kappa u)_i \tag{29}$$

$$= \sum_{i=1}^{N} u_i \sum_{j=1}^{N} \langle \phi(x_i)\phi(x_j)_{\mathcal{H}} \rangle\, u_j \tag{30}$$

Where $\mathcal{H}$ represent the Hilbert space we project the kernel [?].

$$= \left\langle \sum_{i=1}^{N} u_i\phi(x_i), u_j\phi(x_j) \right\rangle_{\mathcal{H}} \tag{31}$$

$$\langle u, \kappa u \rangle = \left\| \sum_{i=1}^{N} u_i\phi(x_i) \right\|_{\mathcal{H}}^{2} \geq 0 \tag{32}$$

Therefor $\kappa$ is positive definite.

Using this property of the kernel $\kappa$ we are able to create serveral other kernels including

- **LinearRBF**
  Here we combine two kernels, precisely **Linear and RBF kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{linrbf}} = \kappa(x_i, x_j)^T\kappa(x_i, x_l) \tag{33}$$

- **RBFPoly**
  Here we combine **RBF and Polynomial kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{rbfpoly}} = \kappa(x_i, x_j)^T\kappa(x_i, x_l) \tag{34}$$

- **RBFCosine**
  **RBF and Cosine kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{rbfcosine}} = \kappa(x_i, x_j)^T\kappa(x_i, x_l) \tag{35}$$

- **EtaKernel**
  The **EtaKernel** is a composite combination of **LinearRBF, RBFPoly and RBFCosine** and it is given by
  $$\hat{\mathbf{K}}_{\mathbf{etarbf}} = \hat{\mathbf{K}}^T_{\mathbf{linrbf}}\hat{\mathbf{K}}_{\mathbf{rbfpoly}} + \hat{\mathbf{K}}^T_{\mathbf{rbfpoly}}\hat{\mathbf{K}}_{\mathbf{rbfcosine}}$$

## 3 EXPERIMENT

### 3.1 Dataset

We perform a comparison between PCA and its kernel version using different datasets including circle, moon, classification, swiss roll and iris datasets. Given $N$, the total number of samples. Circle ($N = 1000$), Moon ($N = 1000$), Classification ($N = 1000$), Swiss roll ($N = 1000$), Iris ($N = 150$).

### 3.2 Performance analysis

## 4 CONCLUSION

## References

[1] Goldberger, Authur S. Classical Linear Regression. Econometric Theory. John Wiley & Sons. pp. 158. ISBN- 04471-31101-4, 1964.

[2] Scott Menard. Applied Logistic Regression Analysis. SAGE PUBLICATIONS. ISBN-0-7619-2208-3, 2001.

[3] Zhu J, Hastie T. Kernel logistic regression and import vector machine. *J Comput Graphic Stat*, 14:185–205, 2005.

[4] Keerthi, S., Duan, K., Shevade, S., and Poo, A. A Fast Dual Algorithm for Kernel Logistic Regression. *International Conference on Machine Learning, 19*, 2002.

[5] Maher Maalouf, Theodore B. Trafalis, Indra Adrianto., Kernel logistic regression using truncated Newton method. *Computer Management Science*, 8:415-428, 2009.

[6] Mehmet Gönen, Ethem Alpaydin. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research*, 12:2211-2268, 2011.