Logisitic Regression

Marijan SORIĆ
Norwegian University of Science and Technology – Machine Learning
marijaso@stud.ntnu.no

I. What's this algorithm

A logistic regression is an algorithm of machine learning that is used to resolve binary classification problems. Given a dataset $\mathscr{D} = \{x^{(i)}, y^{(i)}\}_1^N$ where $y^{(i)} \in \{0,1\}$, the algorithm should *learn* a function h_θ such that $h_\theta(x) = \mathbb{P}_\theta\left[y=1|x\right]$ on the training dataset and beyond. Then, with a threshold of 0.5, we can define $y^{(i)} = \mathbb{1}_{\{h_\theta(x^{(i)})>0.5\}}$. We assume that $\mathbb{P}_{\theta|x} \sim \mathscr{B}(\sigma(\theta^T x))$, where σ is the sigmoid function. Thus, the algorithm will be able to generalize well. To aim that purpose, some weights θ should be found.

$$\hat{y} = h_\theta(x) \triangleq \frac{1}{1+e^{-\theta^T x}}$$

The parameters θ of the function h_θ must minimize the loss function, which measures if the target function well classifies the data. $\theta^* = \operatorname{argmin}_\theta \mathscr{L}_\theta(\hat{y}, y)$ The loss function chosen to be minimized is the negative log likelihood.

$$\mathscr{L}_\theta(\hat{y}, y) = -\sum_{(x,y)\in\mathscr{D}} y \log h_\theta(x) + \left(1-y\right)\log\left(1-h_\theta(x)\right)$$

To optimize the parameters θ, we use the gradient descent on the loss function $\mathscr{L}$. Hence, we define two hyperparameters : epoch N and the learning rate η. At each step, the parameters are updated toward the direction of the gradient :

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathscr{L}$$

II. Inductive bias

There are many possibilities to find the function $h_\theta : \mathbb{R} \rightarrow \mathbb{R}$ that satisfies the training dataset. Nevertheless, the logistic regression algorithm searches the function h in a precise subset of $\mathbb{R}^{\mathbb{R}}$ which has a finite dimension. In the case of this algorithm, the hypothesis space called $H \cong \mathbb{R}^n$ describes a restriction bias. The term $\theta^T x$ shows that the algorithm searches a boundary decision which is linear between the features $(x_1, ..., x_n)$ of the input. In fact, a hyper plan of $\mathbb{R}^n$ should exist and separate the two classes. The initials weights provided constraint the local minimum found by minimizing the loss function $\mathscr{L}$.

III. Some plots

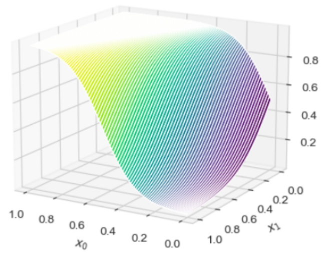

Fig. 1. Plot of function h_θ in 3D for the dataset 1. The surface describes the probability of belonging to the class 1.

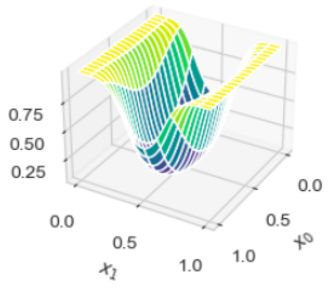

Fig. 2. Plot of function $h_\theta(x) = \mathbb{P}_\theta\left[y=1|x\right]$ in 3D for the dataset 2. At $z = 0.5$ the boundary decision is an ellipse.

IV. Preprocessing

For the second dataset, the features are well normalized but the boundary decision is no longer a line *hyperplan of* $\mathbb{R}^2$. Regarding the dataset, the boundary decision has the shape of an ellipse $\alpha(x-x_e)^2 + \beta(y-y_e)^2 = r$. To address the issue, we introduce some new features by doing what is called *feature engineering*. Furthermore, we will also introduce a *bias* in order to obtain the variable r. Thus, we have with $x_4 = 1$:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_0^2 + \theta_3 x_1^2 + \theta_4 \cdot 1$$

V. Results

	Train Dataset 1	Train Dataset 2	Test Dataset 2
Accuracy ↑	**0.900**	**0.958**	0.938
Cross Entropy ↓	**0.209**	**0.205**	0.213

To conclude, scores remain better on the training dataset rather than on the test dataset, but it seems that there is no overfitting.

K-Means

Marijan SORIĆ
Norwegian University of Science and Technology – Machine Learning
Norway, Trondheim
marijaso@stud.ntnu.no

I. What's this algorithm

The K-Means algorithm is an unsupervised algorithm which creates K clusters from a dataset $\mathscr{D} = \{x^{(i)}\}_1^N$. The centroids μ of each cluster c are located in order to minimize within-cluster variances (metric : squared Euclidean distances) over the dataset. Here it is the *distortion*, which is basically the Euclidean distance between each data $x^{(i)}$ and their centroids μ. Cluster centroids μ are randomly initialized by choosing K data points. *K-Means++ initializes cluster centroids with data points that are far from each other.* The algorithm is a loop until it converges (defined with a fixed number of epochs).

$$\begin{cases} \forall i \in [\![1, N]\!] & c^{(i)} := \text{argmin}_j \|x^{(i)} - \mu_j\|^2 \\ \forall j \in [\![1, K]\!] & \mu_j := \frac{1}{\text{card}(\{i, c^{(i)} = j\})} \sum_{i, c^{(i)} = j} x^{(i)} \end{cases}$$

The convergence of the algorithm and the local minimum found depends on the initialization. That is why we compute several initializations and keep the one with the lowest distortion $d = \sum_{i, c^{(i)} = j} \|x^{(i)} - \mu_j\|^2$. Hopefully, the local minimum found turns out to be the global minimum.

The main hyperparameter of K-Means is K. Plotting $d = f(K)$ shows that $\frac{\partial d}{\partial K} < 0$ which implies $\lim\limits_{K \to N} d(K) = 0$. Another way to find the right K is to locate the *elbow* of the function d : where $\frac{\partial^2 d}{\partial K^2} = 0$. Finally, rather than using distortion, silhouette $s \in [-1, 1]$ can be used as metric : $K^* = \text{argmax}_K s(K)$.

II. Inductive bias

The K-means algorithm assumes that data points of a cluster must be near of the centroid of the cluster. In the dataset 3 for example, a circle should form a cluster but the centroid will be in the center of the circle which doesn't belong to the circle. This describes exactly the notion of convex set C $(\forall x, y \in C, 0 < t < 1, (1-t)x + ty \in C)$. The K-means assumes that all clusters are convex sets, which might be not the case. The norm used also plays a prior role (in distortion) because centroids should be close to each data points.

III. Some plots

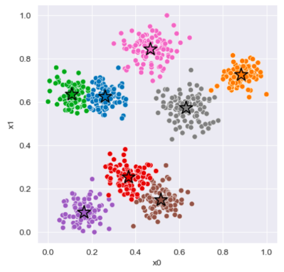

Fig. 1. Clusters for the dataset 2, $K = 8$

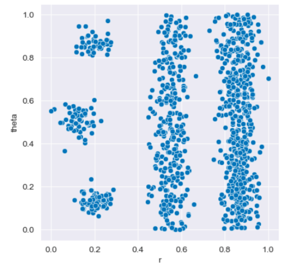

Fig. 2. Dataset 3 in polar coordinate (r, θ) after applying ϕ and normalization.

IV. Preprocessing

For the second dataset, one can notice that the range of values taken by each feature x_0, x_1 are different. To avoid a predominance of a feature over the others, we decide to normalize the dataset. The easier way is the following $[0,1] \ni x \leftarrow \frac{x - \min x}{\max x - \min x}$. Consequently, both features can have the same weight on the clustering algorithm.

The third dataset is more complex: data points seems to be symmetrical, and there are two circles. Without preprocessing, the algorithm fails to find the right clusters because the euclidean distance is no longer appropriate (exemple : circles). Intuitively, the polar coordinate are more appropriate to represent the dataset. We define the bijection $\phi : \mathbb{R}^2 \setminus \{0\} \to \mathbb{R}_+^* \times [0, 2\pi[$ that converts cartesian coordinates to polar ones. Then, we normalize the dataset, to obtain the Fig. 2.

V. Results

	Distortion ↓	**Silhouette Score ↑**
Dataset 1	8.837	0.672
Dataset 2	3.922	0.592