Logistic Regression

Algorithm

Logistic regression is a discriminative approach for classification. Can be adapted to lots of binary classification problems. Starting from a dataset D of m samples defined by n features the aim of algorithm is to find n+1 parameters called weights such that is possible to classify a new sample with the linear combination P(x=1)=w0+w1x1+...wnxn

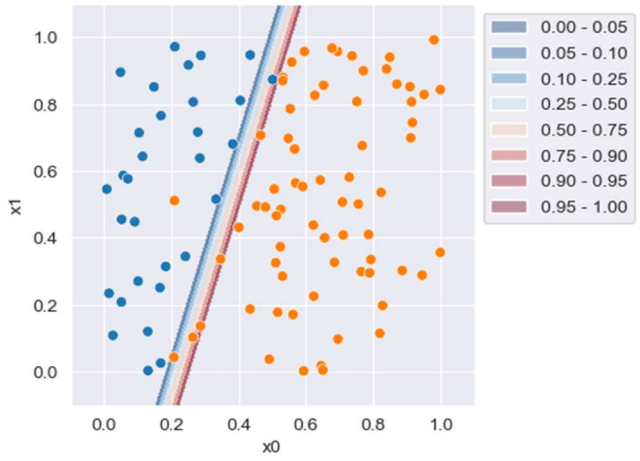
The training, that consists of maximizing the likelihood and so getting the best weights, could be done in different ways, for example with the ascent gradient method. Consist in a loop in which each round the weights are taken and updated until they reach a certain precision.

Inductive Bias

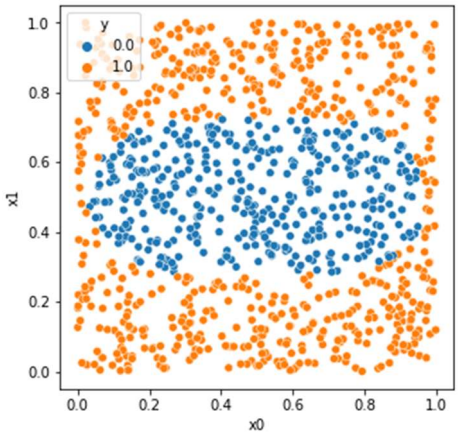
The model assumes that the separation surface between the classes is a linear surface (hyperplane), corresponding to linear decision of the form: $y=w^t x + b$

Also assume a binary outcome (unless we move to the multiclass LR), a large sample size with sufficient variability. Assume also Independent and identically distributed random variables.

Plots



This is how the trained model divides the surface for the first dataset. This plot further confirms the high quality of the model, in addition to its accuracy.



Plot of the second dataset. It's easy to determinate that the separation between the classes can't be linear. This contrast one of the inductive biases of the model.

Result/Preprocessing

```
Accuracy: 0.970
Cross Entropy: 0.385
```

dataset 1

```
Train
Accuracy: 0.898
Cross Entropy:  2.087

Test
Accuracy: 0.876
Cross Entropy:  2.381
```

dataset 2

From the plot of the second dataset, it's possible to see that the classes can't be linear separable. To overcome this problem, it's possible to map the dataset into a *higher feature space* so adding feature that are combination of the primal one, in this case a quadratic expansion is sufficient to find a space where the classes can be (approximately)) separated by a linear surface.

Finally we can say that both first and second dataset problem can be well solved by the LR method.

K-means

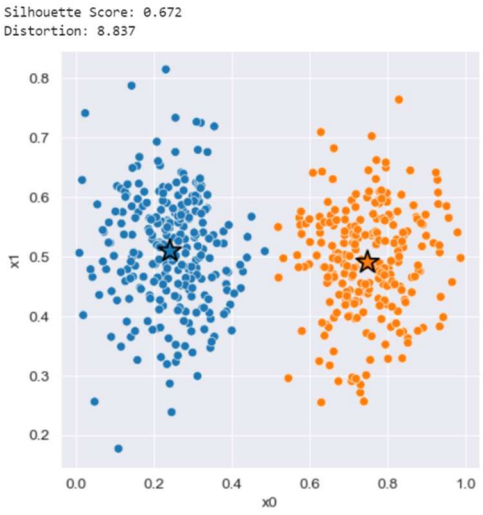
Algorithm

K-means, a widely used partitional clustering approach, relies on the fundamental concept of associating each cluster with a centroid. The number of cluster K is specified, each sample is assigned to the closest centroid and the centroid than is assigned to the mean of the cluster, these points are done in loop until the center points are stable. To determine the closeness between data points and centroids, K-means employs distance metrics like Euclidean distance. Most of the convergence happens in the first few iterations.

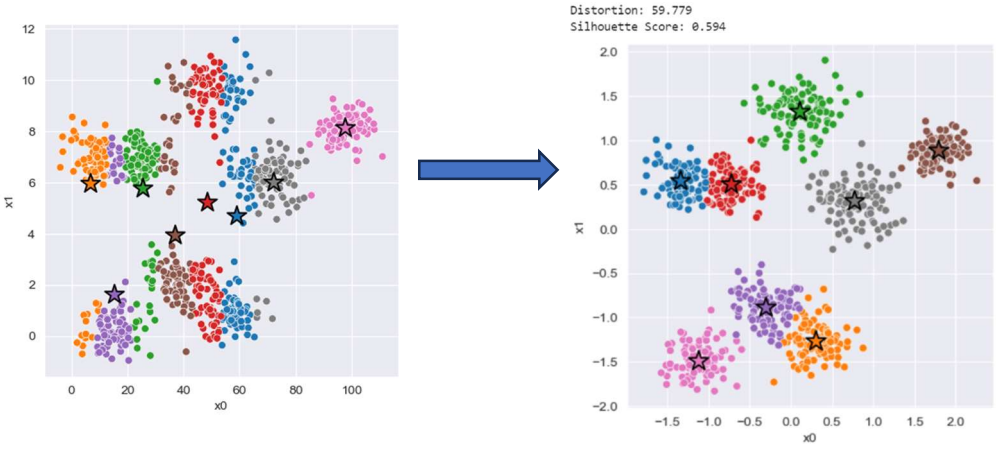
Inductive Bias

K means assume that the dataset can be partitioned into k cluster each one with a centroid and assume that are spherically and equally sized. The algorithm is also strictly dependent on the metric used to calculate the distance between points, and so assume that the chosen one is the correct metrics.

Plots



Algorithm after running on the first cluster problem. On this plot we can see clusters well defined.



On the left the result of clustering with raw features. The cluster division seems us to be more a vertical division.

On the right, the features are standardized, and the clusters founded are the correct one, each one with an expended quite circular shape.

Result/Preprocessing

The problem of the second dataset was the distance computation between points. Indeed, seeing at the final cluster it's possible to see that one dimension was favorite to compute the distance and so the division was more related on it. The solution was to apply a normalization of the features, like *z-score normalization*, to give same influence on cluster computation, always continuing using the normal Euclidian distance measure.

Finally, we obtain for both dataset a correct clusters detection and the following scores.

	Silhouette	Distortion
Dataset 1	0.672	8.837

	Silhouette	Distortion
Dataset 2	0.594	59.779