1 Logistic Regression

1.1 What is logistic regression?

Logistic regression estimates the probability of an event occurring based on a given data set of independent variables. The data set is composed of the input variables x^i (features) and the output variables y^i. This type of model is often used for classification and predictive analytic where output variables take one of two values, or are associated to one out of two groups. For example, logistic regression can be used to predict whether an email is a spam or not or if someone has voted or not.

Since the outcome is a probability, the output variable is bounded between 0 and 1. The goal of the logistic regression algorithm is, given a training set, to learn a hypothesis function so that $h(x)$ is a good predictor for the corresponding value of y. In our case, the form of the hypothesis function is $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$.

To learn the parameter θ, we try to make $h(x)$ close to y for the training examples we have. To do this, we define a cost-function $J(\theta)$ that measures how close the $h(x^i)$'s are to the corresponding y^i's. The cost-function is defined by $J(\theta) = \frac{1}{2}\sum(h_\theta(x^i) - y^i)^2$.

For the implementation of this problem, we used a gradient descent algorithm to determine the parameter θ of the model. We first initialize θ randomly. Then, we use an update rule to minimise the cost-function. The update rule is : $\forall i \in (0, n), \theta_j = \theta_j + \alpha(y^i - h_\theta(x^i)x^i_j$, where we manually set α to optimize the algorithm accuracy. In the algorithm we have to add a column of $x_0 = 1$ so that θ and x^i have the same dimensions.

1.2 Inductive bias

For any given data points (x, y), we assume that the variable y is linearly dependent on the features variables x. Therefore, the resulting model linearly fits the training data. This assumption can limit the model's capacity to learn non-linear functions. We also assume that there's a hyperplane that separates the two classes from each other. This simplifies the problem, but one can imagine that if the assumption is not valid, we won't have a good model.

1.3 Difficulty of second data set and solutions

The first data set can easily be split into two groups using a straight line. That is not the case with the second data set. When the decision function is not linear, we can map the original feature plane to an higher-dimensional feature space where the training set is separable. In our algorithm, we added a feature representing the absolute value of the first and second feature to create a new feature. The algorithm was then able to find a linear function to split the data in two groups.

2 K-means

2.1 What is k-means?

K-Means is a clustering algorithm where the number of clusters is defined in advance. K-means clustering is generally used when there is not a specific outcome variable that we are trying to predict. Instead, it is used when we have a set of numeric features we want to use to find collections of observations that share similar characteristics. It can be used when we have large data set because it's fast compared to other clustering algorithms: it's performance scales linearly with the number of data points in a data set.

K-Means begins with k randomly placed centroids. Centroids are the center points of the clusters. Then we assign each existing training example to its nearest centroid. The nearest centroid is defined by the euclidean distance between a centroid and a data point.

After the assignment, we move the centroids to the average location of points assigned to it so that the centroids are still the center points of the clusters. To do that, we sum the number of data points assigned to a centroid and their distance. If no data points is assigned to a centroid, it is randomly re-initialized.

The current iteration concludes each time we're done relocating the centroids. We repeat these iterations until the assignment between multiple consecutive iterations stops changing.

2.2 Inductive bias

In the k-means algorithm, we assume that entities belonging to a particular cluster should appear near each other, and those that are part of different groups should be distant. In other words, we assume that similar data points are clustered near each other and away from the dissimilar ones.

Moreover, we assume that clusters have a spherical-like shape and that they have a similar size or density because k-means does a poor job of clustering when the clusters have a complicated geometric shape.

Lastly, we have looked at the 2D-plot of the data to fix k.

2.3 Difficulty of second data set and solutions

The first data set can easily be split into two clusters. Both of the cluster have a similar radius and density. However, in the second data set, the clusters have different radius size and some clusters are more dense than others. Moreover, the features of the second data set are bound between -1 and 1 and -10 and 10 (approximately). This means that when we calculate the euclidean distance between a cluster and a data point, one value has more weight than the other.

To have a better clustering, first we have to scale the values so that both features are bound between the same values, by using a multiplicative factor. This means that we will also have to divide our centroids by that factor.

Then, we optimize the initialization of the algorithm because otherwise, the algorithm will probably be stuck in a local optimum and won't converge to global optimum. To do that, we use the K-means++ initialization method to spread out the initial cluster centers. The first cluster center is chosen at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points by computing the distance with closest existing cluster center and selecting the higher one.