Tarea 3 - Logistic Regression, classification and Metropolis Hastings

MA5204: Machine Learning

Submitted To:
Felipe Tobar

Submitted By :
Alexandre Poupeau

May 29, 2019

1 Moons Classification

First, we generate some data to play with. We divided it into two parts train and test part. Here is a graph of all the data we have (1000 points with two classes) :

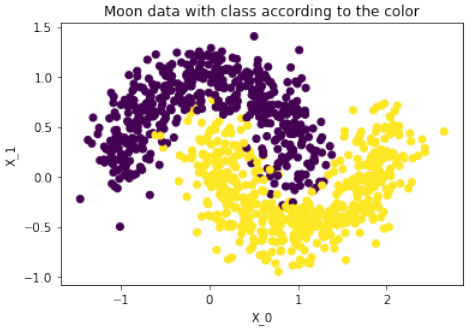

Figure 1: Graph of all the data

1.1 Logistic Regression

The model of logistic regression is $y = \sigma(\omega^T \tilde{X})$. σ is the logistic function $\sigma(x) = \dfrac{1}{1+\exp(-x)}$. $\omega \in R^3$ is the parameter we wish to find. $\tilde{X} = [X, 1]^T$ where $X \in R^2$. So when the model is trained, we can give an input X and the model returns us the probability that the point belongs to the first class. How do we train the model ?

We wish to maximize the posterior distribution : $p(\omega|X, y) \quad \alpha \quad p(y|X, \omega)p(\omega)$

We have $p(y|X, \omega) = \prod_{i=1}^{N} p(y_i|\omega, X_i) = \prod_{i=1}^{N} p(i \in C_1|\omega, X_i)^{y_i} p(i \in C_2|\omega, X_i)^{1-y_i} = \prod_{i=1}^{N} \sigma(\omega^T \tilde{X}_i)^{y_i} (1 - \sigma(\omega^T \tilde{X}_i))^{1-y_i}$

We suppose a prior as a multivariate normal distribution : $p(\omega) = \dfrac{1}{\sqrt{(2\pi)^3|\Sigma|}} \exp\left(-\dfrac{1}{2}(\omega - \mu)^T \Sigma^{-1} (\omega - \mu)\right)$

We know that at the end, the equation of the hyperplane is $X_1 = -\dfrac{\omega_0}{\omega_1} X_0 - \dfrac{\omega_2}{\omega_1}$. We choose $\mu = [0, 1, 0]$ because, at first glance, we expect a hyperplane $X_1 = \epsilon_0 X_0 + \epsilon_1$ with ϵ_0 and ϵ_1 as small positive real number. We choose $\Sigma = \beta \times I$ with a certain β value. We guess that a uniform distribution is not a good choice because there must a be a higher probability for a particular ω. Concerning the covariance matrix, it is dificult to suppose anything in particular, so we will just simply suppose that there is no correlation at all between values because we have no idea (covariance matrix diagonal and let a medium variance for the value)

We want to minimize $NLL(\omega) \equiv -\sum_{i=1}^{N} y_i \log(\sigma(\omega^T \tilde{X}_i)) + (1 - y_i)(\log(1 - \sigma(\omega^T \tilde{X}_i))) + \dfrac{1}{2} \sum_{i=0}^{2} \dfrac{(\omega_i - \mu_i)^2}{\sigma_i^2}$.

Using this model, we find a ω_{opt} and the corresponding model test accuracy is $\simeq 85\%$. The confusion matrix looks like this $[[94, 15][14, 77]]$. We can see that this model seems to predict quite well the class Here is a graph of the separation line :

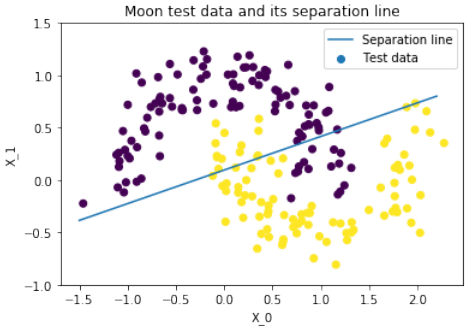

Figure 2: Graph of test data and the separation line

As the model only find a line, it can not be perfect for the data we have, however it is pretty robust though and find a good hyperplane to separate the classes.

1.2 Metropolis-Hastings and posterior estimation

We sample 1000 elements $\omega \in R^3$ from the posterior distribution. We do not use the classic acceptance probability directly $r = \frac{p(w_t|X,y) * q(w_c|w_t)}{p(w_c|X,y) * q(w_t|w_c)}$ as it is commonly used in the Metropolis-Hastings algorithm. Indeed, this leads to crash the code as probability are really small (because of the product of sigmoid (or 1-sigmoid)). In order to overcome this problem, we first calculated $t = \log(p(w_t|X,y)) + \log(q(w_c|w_t)) - \log(p(w_c|X,y) - \log(q(w_t|w_c))$ and then $r = \exp(t)$. As $t = \log(r)$ this gives the exact same r. We used a multivariate normal distribution as a proposal with the mean equal to the previous value w_c and the covariance the identity matrix $\in M_{3,3}(R)$. Five different ω_{MH} at different random indexes in the samples provide the following hyperplanes :

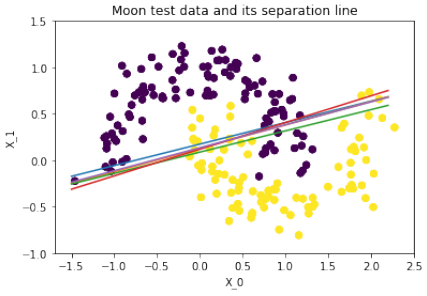

Figure 3: Graph of test data and some separation lines from MH samples

As we can see, the Metropolis-Hastings algorithm allows us to find very good solutions too and to generate plenty of them. All the samples are parameters close to the optimal solution found in the logistic regression part.

1.3 Histograms of parameters

Now that we sampled 1000 elements we can get a higher of every individual distribution for each parameter ω_i that composed $\omega = [\omega_0, \omega_1, \omega_2]$. Here are the three histograms :

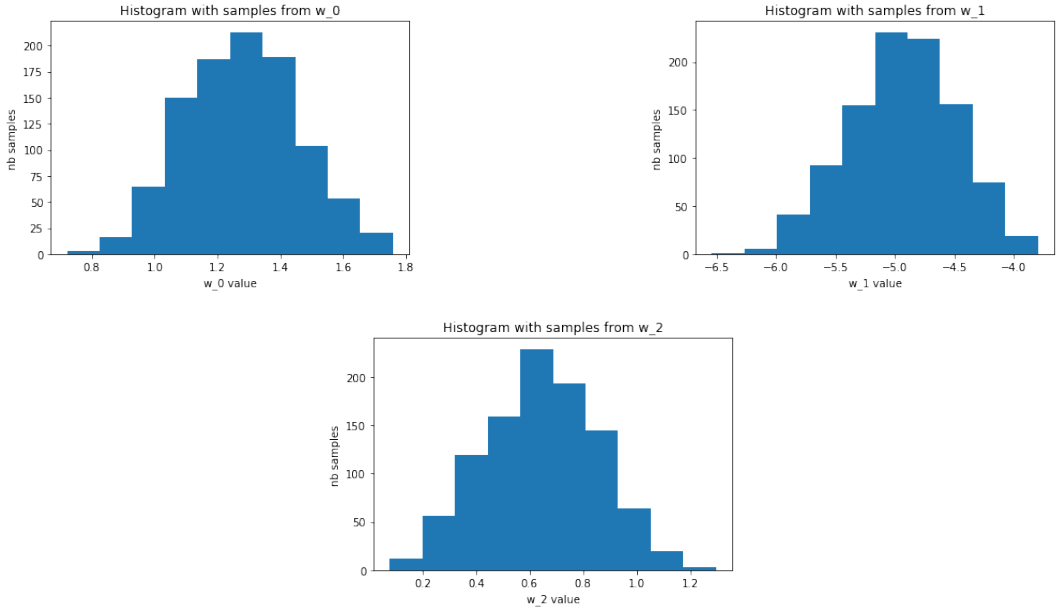

Figure 4: Histogram of the parameters sampled by Metropolis-Hastings

I have to admit that I was not expecting those results : the histograms are really "perfect". The histograms really look like real normal distributions. Therefore our initial choice of prior was not stupid. It is really easy to see the optimal value for each parameter using the sampled distribution (abscissa on maximum value).

We also tried to use different covariance matrixes to change the proposal and see the impact on the distributions found. We found out that it does not change the results. Moreover, we also tried to use a totally different proposal distribution : we tried with a multivariate uniform distribution. Nonetheless, there is no strong difference, the algorithm still converge to the same histograms and gives the same parameters which is quite an expected result. We know the Metropolis-Hastings algorithm does not provide different distributions based on the proposal used.

1.4 Predictions using Monte Carlo Integral

We have that $p(y_*|x_*, X, Y) = \int p(y_*|x_*, \omega)p(\omega|X, Y)d\omega = E_{\omega|X,Y}[p(y_*|x_*, \omega)] \simeq \frac{1}{n}\sum_{i=1}^{n} p(y_*|x_*, \omega) = \frac{1}{n}\sum_{i=1}^{n} \sigma(\omega_i^T \tilde{x_*})$ where the elements ω_i are generated from the posterior distribution $p(\omega|X, Y)$.

We did not used any burnin because it was not necessary as we started the Metropolis-Hastings algorithm with the optimal solution from the logistic regression. Using all the samples and using this method to get predictions, we obtain at the end an accuracy $\simeq 85\%$ which is exactly like the one obtained before. Confusion matrix : $[[94, 15][14, 77]]$. Then we tried to use different thinning step and we manage to obtain 86% accuracy using a bigger thinning step (50). Confusion matrix : $[[94, 15][13, 78]]$. We gain around 1% which is not a huge difference : seeing at the confusion matrix we can actually see that it just classified one better. Concerning the Area Under the Curve from the ROC curve, it is always around 85%.

2 Project information

Our goal is to propose music recommendations based on what a listener listens to. We came up with two ideas to make this. We will use a subset of the Million Song Dataset (MSD, 10000 songs, a feature vector for each song, 1.9Gb) and the Taste Profile dataset linked to the MSD. This dataset contains, for a certain number of persons, the number of times he listened to a track (given a track id in MSD) in a fixed period of time (48,373,586 user - song - play count triplets).

2.1 Genre recommendation method

The idea here is to make a genre-oriented music recommendations. We first need to train a neural network (simple as MLP) where the input is an audio feature vector (key of the song, rhythm, minor or major, ... etc). The output is the genre probability. We can choose different label datasets linked to MSD for music genres. We will use one with around $n = 25$ genres. So the output will be a vector $y \in R^n$.Then for a given person we can choose the most listened musics and get the genres out of it. We then can make recommendations based on what genres a person listen to.

2.2 Implicit recommendation method

The second method is to detect "implicit genres". The idea is not to use supervised learning to classify genres. Here we are going to do clustering on the most listened music (features vectors) for a given person and extract k different types of musics. Then, for each type, we will do similarity search with all the other musics on the MSD dataset. The first m musics with the lowest distance are the recommendations for the given type of music.

This approach is interesting because it is not based on genres but more on the overall implicit kinds of music the listener listen to. The hard part is that as it is unsupervised learning here, it is difficult to choose the "good" number of clusters. It will be something to study and examine in depth. Moreover, this approach looks simpler in the way that we use only two datasets (not the genre one) and that similarity search is computationally effective (only looking for distance between vectors).

2.3 More information

The difficulty of the problem is that is it hard to evaluate how good a music recommendation is. One of the goal we have is to try to find a good way to evaluate it using metrics.

Before, we first wanted to use song datasets but there are to big. Thus it will take too much time to train or preprocess the audio. The smaller audio dataset we found contains 8000 songs of 30 seconds and weights 8Gb. Thus we only use directly dataset of audio feature vectors. We take for granted the feature vectors from the audios. We do not make any preprocessing to turn any audios into feature vectors.