# MAC0460: Exercício-Programa 3

Regressão Logística

Gabriel Araujo - 10297689
Vinícius Moreno - 10297776

15 de Abril de 2019

## 1 Implementação

Nossa implementação envolveu uma função *sigmoid()* separada, para facilitar os cáculos, além de versões vetorizadas da função de custo (*cross-entropy*, presente no *vetor history*) e do cáculo do gradiente. Estabelecemos um vetor de pesos inicial w que, quando recebe None como argumento, é preenchido por valores aleatórios do conjunto [-1,1).

Para implementar a vetorização, optamos por primeiro construir um código válido para um $batch\_size = N$, e depois adaptá-lo para receber qualquer tamanho. Fizemos isso utilizando um ponteiro $p$, salvando a posição de início da seleção dos vetores $X$ e $y$, e atualizando-o somando *batch_size*, conforme avançamos nas iterações.

No caso onde a soma de *batch_size* e ponteiro é maior que o tamanho de colunas dos vetores $X$ e $y$, resetamos o ponteiro para 0, adicionamos as linhas das matrizes não testadas para o começo, e as deletamos do fim. Isso garante que todas as linhas sejam testadas, havendo iterações o suficiente.

Para obter o array com o histórico de updates solicitado, optamos por escrever a função de cross entropy diretamente nas linhas de código, sem criar uma função própria para isso, e usar o método append() no vetor history[] para poder retornar o que foi pedido.

## 2 Testes

Nosso método para testes foi utilizar o *script* em *python* fornecido pelo monitor e modificá-lo, afim de imprimir pontos de cor verde quando o requerimento de threshold for cumprido pelo algoritmo de regressão logística. Pontos de cor vermelha indicam casos de predição incorreta (predição abaixo do *threshold*).

Percebemos que, quanto maior o valor de *batch_size*, maior acaba sendo a constância do algoritmo, ou seja, a variância da precisão dos acertos diminui. Isso porque, como *num_iterations* não está mudando, quando reduzimos o tamanho

do lote de testes (o *batch*), reduzimos o número de exemplos sendo efetivamente utilizados para treinar o algoritmo.

Ao estabelecer o batch_size como N, ocorre um certo padrão nos resultados, que é recorrente. Isso não acontece com tamanhos de lote menores, onde, apesar de, na maioria das vezes, obterem menos precisão que batch_size = N, podem sim apresentar melhores resultados, mas isso acontece de forma esporádica e imprevisível.

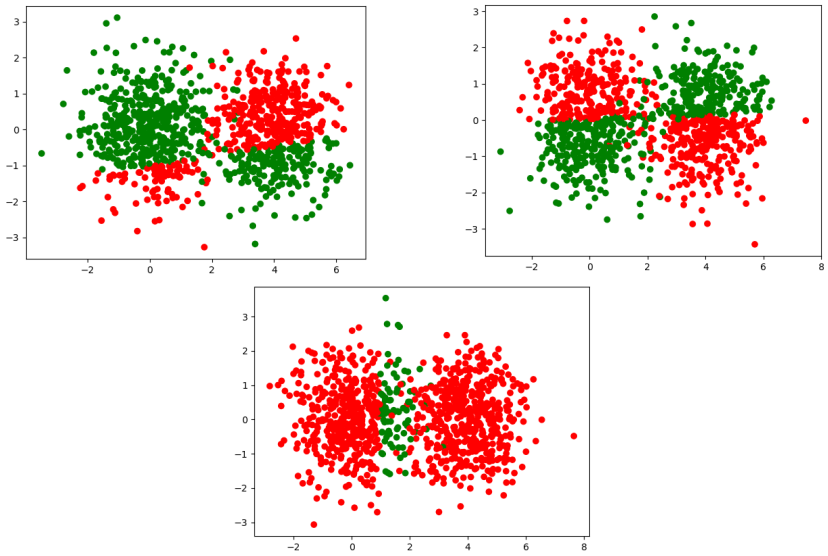
Na ultima seção desse relatório (**4-Imagens**), há as imagens dos testes feitos. Pode-se observar que, com *batch_size* = 1, 5 e 7, há uma grande imprevisibilidade do que irá acontecer com o algoritmo, enquanto que quando *batch_size* = **None**, o teste mostra o certo funcionamento.

# 3 Fontes

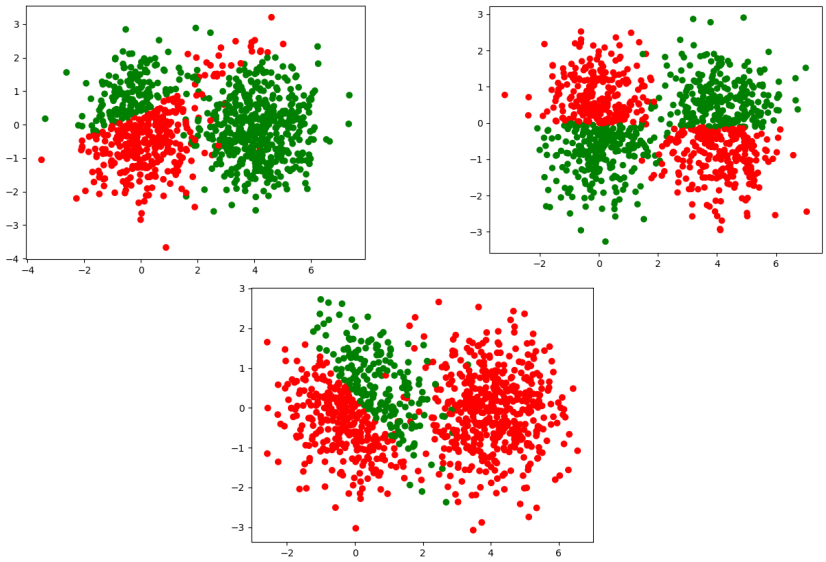
Utilizamos os slides da aula, tanto da Nina quanto do Yaser (Caltech), e o código de teste disponibilizado pelo monitor da matéria no fórum de discussões.

# 4 Imagens

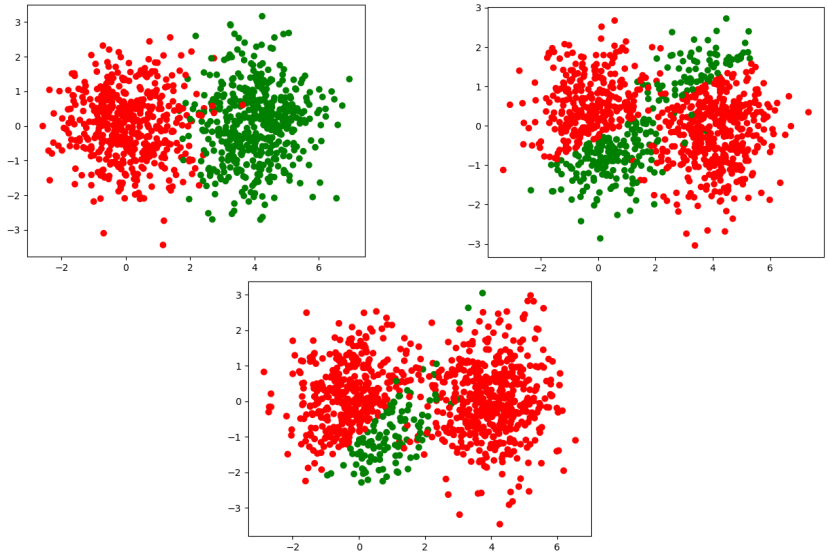
***batch_size* = 1, *threshold* = 0.5**



$batch\_size = 5$, $threshold = 0.5$



$batch\_size = 7$, $threshold = 0.5$



***batch_size* = None, *threshold* = 0.5**