Aula 1

1 Introdução a Machine Learning

1.1 Motivação para o objeto de estudo

Desde a invenção dos computadores, com a presença do software programável, foi possível automatizar inúmeras tarefas antes repetitivas e caras para o ser humano, desde que se pudesse descrever os processos que a compunham, provendo uma ferramenta mecânica que as executasse a velocidades imprescindíveis.

O problema surgiu com aquilo que não conseguimos descrever diretamente com algoritmos: a linguagem, o raciocínio, a generalização, a criação de imagens, etc. Para resolver esses problemas, a solução foi descrever algoritmicamente um agente que fosse capaz de solucioná-los: essa é a missão do campo de Machine Learning.

Formalmente, o campo de Machine Learning busca criar modelos que, perante um ambiente (representado pelos dados numéricos), possam se adequar de modo a satisfazer um objetivo nesse meio, via o processo de aprendizado. 'Aprender' é traduzido na prática como o ajuste automático de parâmetros internos conforme mais e mais dados são alimentados, sendo que estes vêm na forma de vetores: longas listas de números que representam aspectos desse meio.

Tudo isso é meio abstrato, então vamos concretizar um pouco essa introdução. Na vida real, todos já usamos alguma vez o ChatGPT: ele é um modelo que utiliza como dados inúmeros discursos humanos, formando internamente noções lógico-linguísticas; no campo de processamento de imagens, o modelo de Principal Component Analysis é muito útil para compressão e descompressão; a Tesla utiliza Convolutional Neural Networks a fim de reconhecer objetos nas estradas em que seus carros autônomos andam; economistas utilizam diferentes tipos de regressão para melhor prever e entender o mercado; etc.

1.2 Tipos de Modelos

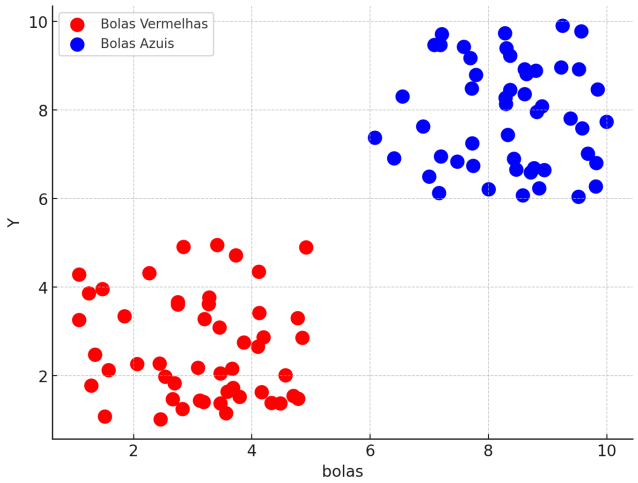
Separamos os modelos de Machine Learning existentes com base em alguns critérios: entre eles, está o tipo de aprendizado. Há, em suma, 3 tipos:

- **Supervisionado:** utilizam-se labels para tentar quantizar a qualidade de previsão do modelo. Separado em 2 tipos: Classificação (quantia discreta de possíveis previsões, na forma de classes) e Regressão (espaço de output contínuo). Ex: Regressão Linear e SVM's.
- **Não-Supervisionado:** não há labels, e o modelo deve usar algum critério interno para agrupar os dados e diferenciá-los. Ex: K-Means e PCA.
- **Semi-Supervisionado:** um campo um tanto cinza entre os dois itens anteriores, onde às vezes utilizam-se labels, e às vezes não. É aqui onde muitos dos modelos mais promissores da atualidade se encaixam. Ex: Reinforcement Learning, Algoritmos Genéticos e Modelos Generativos.

É relevante afirmar que a distinção principal entre Super e Unsupervised Learning (presença ou não de labels) não necessariamente significa que os problemas que encaramos na realidade exijam, necessariamente, uma resposta binária entre ambos caminhos. Por exemplo, se desejamos diferenciar imagens de cachorros e gatos, podemos ou extrair um labeled dataset e treinar um modelo de classificação, ou coletar imagens aleatoriamente de ambos e esperar que um modelo de clustering separe-as naturalmente. Frequentemente, o melhor tipo de modelo a ser escolhido depende da situação.

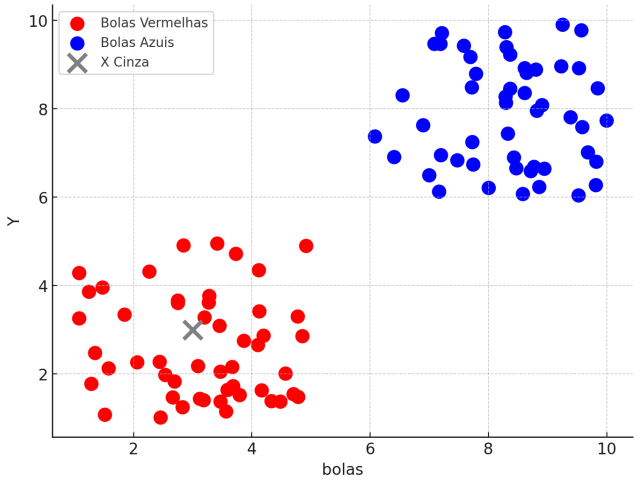
2 Classificação Linear

A classificação é um problema onde dado um exemplo você deve ser capaz de dizer a qual classe ele pertence. Para inicializar o curso começaremos discutindo a classificação em conjunto de dados lineares, isto é, conjunto de dados onde é possível fazer a classificação entre duas classes com funções lineares.

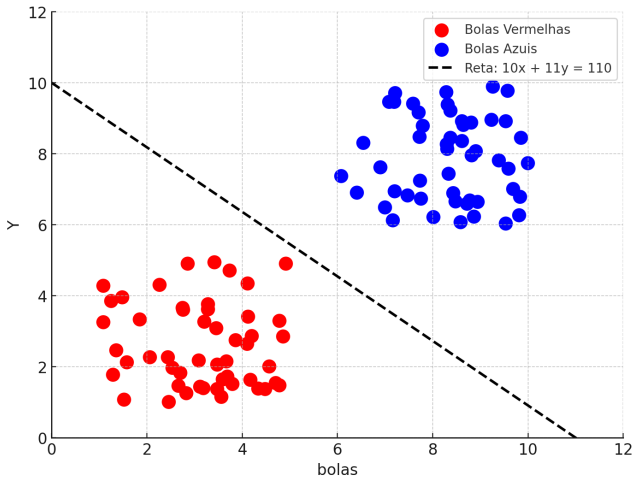


A imagem a cima mostra um conjunto de dados que contém bolas azuis e vermelhas. O objetivo de um modelo de machine learning seria aprender uma regra a qual a partir dos dados a cima sejamos capaz de classificar novos exemplos, ainda não vistos, entre bolas azuis e bolas vermelhas.

A imagem a baixo mostra um exemplo X, no qual não sabemos se ele pertence as bolas vermelhas ou as bolas azuis. Nós, humanos, ao analisar os dados podemos facilmente induzir que o mesmo está presente no grupo das bolas vermelhas, mas de alguma forma queremos que a máquina seja capaz de fazer essa classificação.



Uma simples reta é capaz de separar perfeitamente o nosso conjunto de dados entre bolas vermelhas e bolas azuis, como mostrado a baixo:



Sendo assim, tudo que está a baixo dessa reta será classificado como bola vermelha e tudo que está a cima será classificado como bola azul.

Se o conjunto de dados é linearmente separável o nosso objetivo será encontrar qual é a reta que faz a melhor separação entre as classes.

Vetor de características (Feature Vector $\vec{X}$)

Cada exemplo do nosso conjunto de dados será representado por um vetor de características (feature vector) $\vec{X}$. Tal vetor irá conter as características do nosso exemplo. No exemplo das bolas o feature vector tem dimensionalidade 2, $\vec{X} = \{x_1, x_2\}$, sendo x_1 a coordenada do ponto no eixo horizontal e x_2 a coordenada do ponto no eixo vertical.

2.1 Classificadores lineares que passam pela origem

Em um classificador binário, a decision boundary é a reta (R^2), plano (R^3) ou hiperplano ($R^n, n > 3$) que separa a parte que será positiva (bolas azuis) da parte que será negativa (bolas vermelhas).

Inicialmente podemos começar a definir o conjunto de todos os classificadores lineares que passam pela origem por questão didática antes de generalizar para todos os casos.

Para o R^2, com o feature vector sendo $\vec{X} = \{x_1, x_2\}$, o conjunto de todas as retas que passam pela origem são:

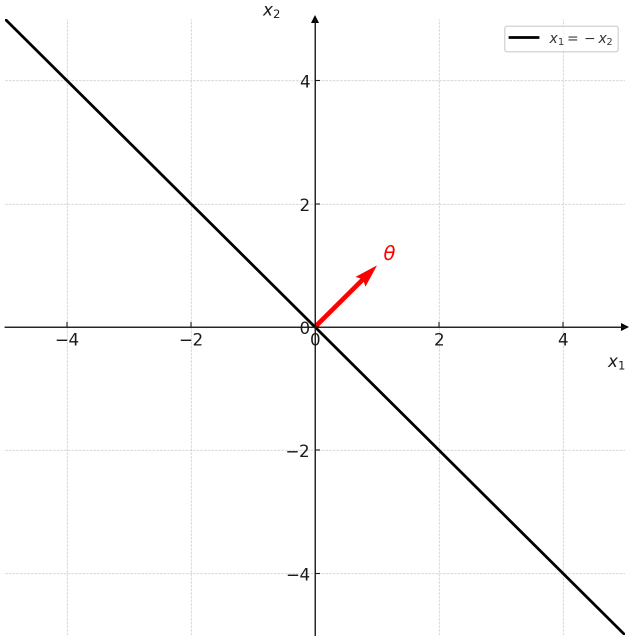
$$\theta_1 x_1 + \theta_2 x_2 = 0$$

Onde $\vec{\theta} = \{\theta_1, \theta_2\}$ é chamado de vetor de parâmetros. Esse vetor de parâmetros é o vetor que o nosso algoritmo de machine learning deve encontrar baseado nos dados para fazer as classificações.

De uma forma mais generalizada, podemos dizer que o conjunto de todos os classificadores lineares que passam pela origem é o produto escalar entre os vetores theta e X:

$$\vec{\theta}.\vec{X} = 0$$

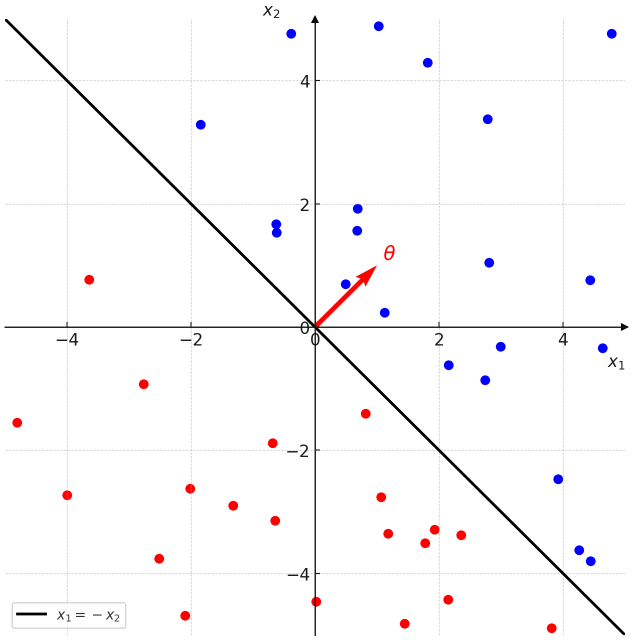
Para o valor $\vec{\theta} = (1, 1)$, a reta seria $x_1 = -x_2$, dando:



Podemos notar que o vetor theta é perpendicular a reta, isso se da pela propriedade de produto escalar, onde, caso o produto escalar entre dois vetores for zero, os mesmo serão perpendicular entre si.

A partir disso, podemos definir a noção de positivo e negativo. Para onde o vetor theta estiver apontando será definido como lado positivo e o outro lado será o negativo. No exemplo da imagem, o vetor theta está apontando para a direita da reta, fazendo com que $\theta.X > 0$ defina os pontos positivos e $\theta.X < 0$ defina os pontos negativos.

Esta reta corretamente classificaria o seguinte conjunto de dados:



Porém, mesmo que os dados sejam linearmente separáveis os classificadores lineares que passam pela origem não é capaz de separar em todos os casos, como no seguinte:

2.2 Conjunto de todos os classificadores linear

Sendo assim, o conjunto de todos os classificadores linear podem ser definidos como:

$$\theta.X + \theta_0 = 0$$

Onde o parâmetro θ_0 move a decision boundary para cima e par baixo enquanto o vetor θ rotaciona a decision boundary.

Podemos definir o classificador h como:

$$h(X, \theta, \theta_0) = sign(\theta.X + \theta_0)$$

Onde $sign(x)$ é uma função que retorna 1, caso $x > 0$, 0 caso $x = 0$ e -1 caso $x < 0$.

3 Algoritmo Perceptron

Já debatemos qual é o problema e o que estamos tentando encontrar (θ, θ_0). Sendo assim, agora iremos introduzir o primeiro algoritmo para poder fazer a classificação linear, sendo ele o algoritmo perceptron.

A primeira coisa que devemos fazer é de alguma forma definir uma forma de saber se o meu classificador está acertando ou errando.
A Loss Function é uma função que calcula o erro de um classificador sob um exemplo.

Para o algoritmo do Perceptron estaremos utilizando uma Loss Function bem simples:

$$L(x_i, y_i) = y_i(\theta.x_i) \leq 0$$

Vimos que $\theta.x_i + \theta_0$ é a classificação em si, e y_1 é a label de x_1, ou seja, y_1 pode ser ou 1, ou -1. Sendo assim, caso $\theta.x_i + \theta_0$ seja menor que zero e a label é -1, o resultado será positivo, caso

$\theta.x_i+\theta_0$ seja menor que zero e a label é 1 o resultado será negativo. Ou seja, a expressão, $y_i(\theta.x_i+\theta_0)$, caso negativa significa que a classificação foi errônea e caso positiva significa que a classificação foi correta.

Perceptron para classificadores que passam pela origem

Inicialmente iremos discutir o algoritmo para os classificadores que passam pela origem antes de generalizar.

A primeira coisa que se deve fazer é inicializar o vetor de parâmetros θ como 0, depois disso deve-se passar por todos os dados e cada vez que ocorre um erro fazer uma atualização no valor do θ:

1. $\theta = 0$
2. for i in range (n)
3. if $y_i(\theta.X_i) \leq 0$
4. $Atualizar\ \theta$

O intuito por traz dessa atualização é que, caso acabou de acontecer um erro, na próxima vez que este exemplo for visto o mesmo deve ser corretamente classificado.

Como θ é inicialmente é zero a primeira iteração será:

$$y_1.(0.x_1)$$

O que com certeza dará 0 dando erro. Para evitar que isso aconteça a próxima vez que o mesmo exemplo for encontrado podemos atualizar theta por:

$$\theta = 0 + y_1.x_1$$

Isso funciona, pois, agora, caso ele seja visto novamente, o resultado vai ser:

$$y_1(0 + y_1.x_1).x_1 = y_1^2.x_1^2$$

y_1^2 será sempre 1 e x_1^2 será positivo. Logo, o resultado vai dar positivo ocorrendo em um acerto.

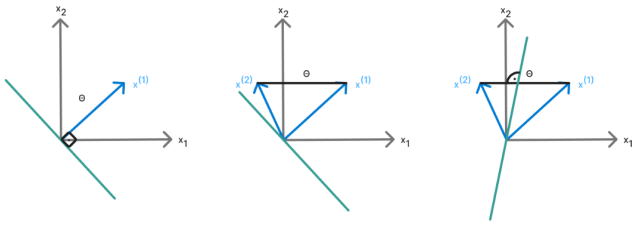
Portanto, o algoritmo será:

1. $\theta = 0$
2. for t in range(T)
2. for i in range (n)
3. if $y_i(\theta.X_i) \leq 0$
4. $\theta = \theta + y_i x_i$

Perceba que agora uma nova variável T foi introduzida. Ela diz quantas vezes deve-se passar pelo conjunto de dados. Isso deve ser feito, pois, pode ser que uma alteração feita em i_x altere a feita em i_y, sendo assim, o algoritmo é executado T vezes para garantir que caso exista um classificador linear que separe os dados ele seja devidamente encontrado.

Intuição geométrica

Imagine a situação inicial, onde $\theta = 0$, logo, para $x^{(1)}$ positivo teremos que vai dar erro, portanto: $\theta = 0 + 1.x^{(1)}$, gerando a decision boundary da primeira situação da imagem abaixo. Logo depois, para $x^{(2)}$ negativo ele vai dar erro, atualizando theta, portanto, $\theta = x^{(1)} + (-1)x^{(2)}$, logo, $\theta = x^{(1)} - x^{(2)}$ como evidenciado pela segunda situação criando assim a nova decision boundary que aparece na terceira situação.



Algoritmo geral

O algoritmo geral será dado por:

1. $\theta = 0; \theta_0 = 0$
2. for t in range(T)
2. for i in range (n)
3. if $y_i(\theta.X_i + \theta_0) \leq 0$
4. $\theta = \theta + y_i x_i$
5. $\theta_0 = \theta_0 + y_i$

A única mudança é a linha 5, onde o θ_0 será atualizado muito parecido com θ com a diferença que não vai ter $x^{(i)}$. Isso acontece, pois, a equação é: $\theta.x + \theta_0$, ou seja, é como se o $x^{(i)}$ de θ_0 fosse sempre 1, então a forma de atualizar é a mesma, porém para θ_0, $x^{(i)}$ será sempre 1.

4 O que veremos na próxima aula

Na próxima aula, nós falaremos sobre 2 tópicos importantes em Machine Learning: a definição e uso da optimização para formular algoritmos de aprendizado com a utilização de Gradiente Descendente, e a matemática por trás das 'Margin Boundaries', essenciais para a formulação matemática das Support Vector Machines.

Também exibiremos um código já pronto do algoritmo 'bag of words' na parte prática da aula, que é uma técnica incipiente de processamento de linguagem para extração de features.