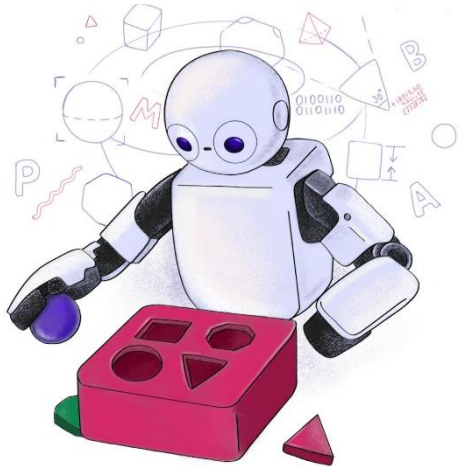
TP555 - Inteligência Artificial e Machine Learning:

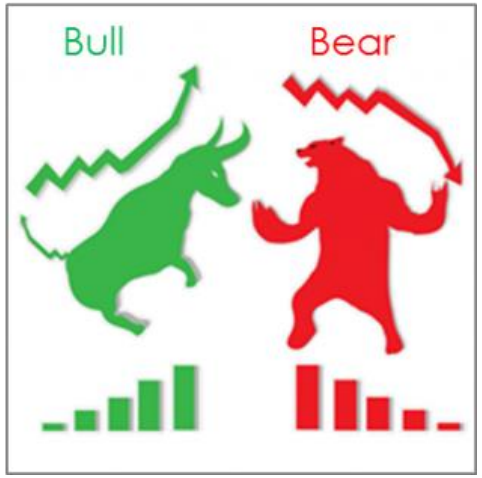
k-Vizinhos mais Próximos

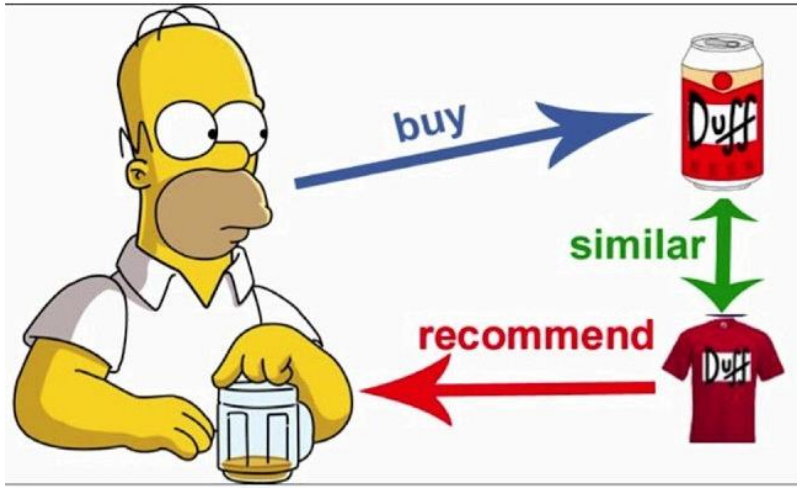


Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Motivação





Além dos exemplos de classificação que nós vimos nas outras aulas, nós podemos também utilizar classificadores para:

- Predizer se o mercado de ações irá subir ou cair.
- Análise de crédito para diferenciar entre clientes de baixo e alto risco.
- Sistemas de recomendação (e.g., de produtos como filmes, bebidas, etc.).

k-vizinhos mais próximos (k-NN)

- O algoritmo k-NN (do inglês, k-Nearest Neighbours) é uma das estratégias mais simples para se solucionar problemas tanto de ***classificação*** quanto de ***regressão***.
- Ele é um algoritmo do tipo ***não-paramétrico***, pois diferentemente dos outros algoritmos que vimos até o momento, não há um ***modelo*** a ser treinado, tampouco se faz qualquer suposição a respeito dos dados.
- A única suposição é que uma medida de distância entre dois exemplos pode ser calculada.
- **Funcionamento**:
  - o algoritmo k-NN necessita que todos os exemplos de treinamento, $\boldsymbol{x}(i) = [x_1(i) \quad \cdots \quad x_K(i)] \in \mathbb{R}^{K\times 1}$, e seus respectivos rótulos $y(i), i = 0, \dots, N-1$ (i.e., as respostas desejadas) sejam armazenados em memória.
  - Em seguida, dado um exemplo de entrada $\boldsymbol{x}'$, a saída para este exemplo dependerá dos rótulos associados aos k exemplos de treinamento ***mais próximos*** do exemplo de entrada $\boldsymbol{x}'$ no espaço de atributos.

k-vizinhos mais próximos (k-NN)

- Por exemplo, para regressão, nós podemos tomar a ***média aritmética*** dos rótulos/saídas dos k vizinhos mais próximos:

$$\hat{y}(\boldsymbol{x}') = \frac{1}{k}\sum_{\boldsymbol{x}(i) \in N_k(\boldsymbol{x}')} y(i),$$

onde $N_k(\boldsymbol{x}')$ é a vizinhança de $\boldsymbol{x}'$, formada pelos exemplos de treinamento $\boldsymbol{x}(i)$ que correspondem aos k vizinhos mais próximos de $\boldsymbol{x}'$.

- Para classificação, por exemplo, dentre k vizinhos mais próximos, escolhemos a classe com maior número de exemplos.
- **OBS**.: Não confundam o número de atributos K com o número de vizinhos mais próximos, k.

k-Vizinhos mais Próximos (k-NN)

- Portanto, o uso do k-NN envolve a definição de:
  - Uma ***métrica de distância*** que deve ser calculada no ***espaço de atributos*** a fim de identificar os vizinhos mais próximos.
  - Um valor para o ***hiperparâmetro*** k, ou seja, a escolha do número de vizinhos que devem ser levados em consideração para a geração da saída correspondente ao exemplo de entrada, $\boldsymbol{x}'$.
- Como k é um ***hiperparâmetro*** do algoritmo k-NN, pode-se utilizar, por exemplo, a abordagem da ***validação cruzada q-fold*** para encontrar o melhor valor de k (para não haver confusão com o parâmetro ***k*** do k-NN, utilizei ***q*** aos invés de k para especificar o número de pastas/dobras do ***k-fold***). Podemos utilizar também ***GridSearch*** ou ***RandomSearch***.
- Devido a estas características, o k-NN é visto como um ***algoritmo de aprendizado competitivo***, uma vez que os elementos do modelo (que são os próprios exemplos de treinamento) competem entre si pelo direito de influenciar a saída do algoritmo quando a ***medida de similaridade*** (***distância***) é calculada para cada novo dado de entrada.

k-Vizinhos mais Próximos (k-NN)

- Além disso, o k-NN explora a ideia conhecida como ***lazy learning***, uma vez que o algoritmo não "*constrói*" um ***modelo*** até o instante em que uma predição é solicitada.
  - Ou seja, a "*construção*" do modelo é atrasada até que uma consulta seja feita.
- O k-NN segue o paradigma de ***aprendizado-baseado em exemplos***, onde ao invés de obter/treinar um modelo que generalize a partir do conjunto de treinamento, ele compara novos exemplos de entrada com os exemplos do conjunto de treinamento armazenados em memória.
- O k-NN tem como desvantagem o fato de que todos os dados de treinamento precisam ser ***armazenados*** e ***consultados*** para se identificar os ***k*** vizinhos mais próximos.
  - Portanto, a ***predição*** poderá ser demorada dependendo do tamanho do conjunto de treinamento, pois deve-se calcular a ***distância*** entre o exemplo de entrada e todos os exemplos do conjunto.
  - Além disto, como vimos, o conjunto de treinamento deve ser armazenado em memória, e caso esse conjunto seja muito grande, pode não haver memória o suficiente para armazená-lo.

Métricas de distância

- **Definição**: Uma métrica de distância fornece a distância entre os elementos de um conjunto. Se a distância é igual a zero, os elementos são equivalentes, caso contrário, os elementos são diferentes uns dos outros.
- Existem várias ***métricas de distância***, mas vamos discutir apenas as mais utilizadas através de uma métrica de distância generalizada.
- **Distância de Minkowski**: é uma métrica definda no ***espaço vetorial normado*** (ou seja, um ***espaço vetorial*** no qual uma ***norma***, $p(.)$, é definida) que satisfaz algumas propriedades.
- A ***norma vetorial***, $p(.)$, é uma função que mapeia $\mathbb{R}^{K\times 1} \rightarrow \mathbb{R}$ e que exibe algumas propriedades que veremos à seguir.
- Sejam 2 vetores, $\boldsymbol{v}$ e $\boldsymbol{u} \in \mathbb{R}^{K\times 1}$, a norma $p(.)$ dos vetores é uma função com valores não-negativos com as seguintes propriedades:
  - $p(\boldsymbol{v}+\boldsymbol{u}) \leq p(\boldsymbol{v}) + p(\boldsymbol{u})$ (ou seja, a norma satisfaz a ***desigualdade do triângulo***).
  - $p(a\boldsymbol{v}) = |a|p(\boldsymbol{v})$, para todo $a \in \mathbb{R}$ (ou seja, a norma é ***absolutamente escalável***).
  - Se $p(\boldsymbol{v}) = 0$, então $\boldsymbol{v} = \mathbf{0}$, ou seja, o ***vetor nulo*** (ou seja, a norma é ***positiva definida***).

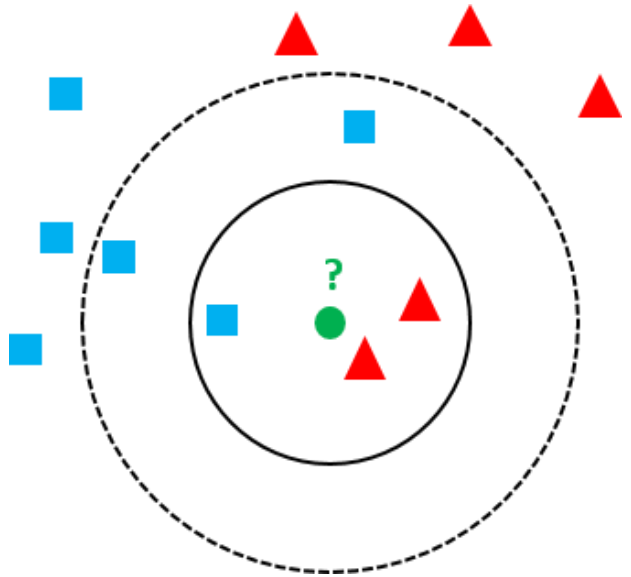
Distância de Minkowski

- A ***distância de Minkowski*** de ordem p é calculada usando-se a equação abaixo

$$p(\boldsymbol{x}; \boldsymbol{y}) = \left(\sum_{i=1}^{K} |x_i - y_i|^p\right)^{1/p}.$$

- A ***distância de Minkowski*** é uma métrica de ***distância generalizada***, ou seja, podemos manipular a equação acima, através do parâmetro p, para calcular a distância entre dois pontos de dados de formas diferentes.
- **Casos particulares**:
  - Para $p = 1$, temos a ***distância de Manhattan***: $p(\boldsymbol{x}; \boldsymbol{y}) = \sum_{i=1}^{K} |x_i - y_i|$.
  - Para $p = 2$, temos a ***distância Euclidiana***: $p(\boldsymbol{x}; \boldsymbol{y}) = \sqrt{\sum_{i=1}^{K} |x_i - y_i|^2}$.

k-NN para classificação



- Com relação ao problema da classifcação, a saída da equação

$$\hat{y}(\boldsymbol{x}') = \arg\max_{q \in Q} \sum_{\boldsymbol{x}(i) \in N_k(\boldsymbol{x}')} \delta(q, y(i)),$$

Voto majoritário

gerada pelo k-NN equivale a tomar o voto majoritário dos k vizinhos mais próximos de $\boldsymbol{x}'$, onde q é uma das classes do conjunto de classes Q e $\delta(i,j) = 1$ se $i == j$ e 0 caso contrário.

- Ou seja, um novo exemplo de entrada, $\boldsymbol{x}'$, é classificado como sendo pertencente à classe que contiver o maior número de vizinhos de $\boldsymbol{x}'$.
- **Exemplo de classificação com k-NN**: na figura ao lado, o exemplo de teste (ponto verde) pode ser classificado como pertencente à classe ***quadrados azuis*** ou à classe ***triângulos vermelhos***. Se k = 3 (círculo com linha sólida), ele é atribuído à classe de triângulos vermelhos pois existem 2 triângulos e apenas 1 quadrado dentro do círculo interno. Se k = 5 (círculo tracejado), ele é atribuído à classe de quadrados azuis (3 quadrados vs. 2 triângulos dentro do círculo externo).
- **Observação**: em algumas circunstâncias, uma técnica bastante utilizada para se classificar os exemplos de entrada é atribuir pesos diferentes à contribuição de cada vizinho à decisão final de tal forma que vizinhos mais próximos contribuem mais do que vizinhos mais distantes. Portanto, uma forma usual é definir os pesos como sendo inversamente proporcionais às distâncias dos vizinhos ao exemplo de entrada $\boldsymbol{x}'$.

Exemplo: Classificação k-NN com SciKit-Learn

```
# Import all necessary libraries.
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_blobs

# Number of examples.
N = 100
centers = [[-1, 0], [0, 0]]
# Create a 2-class dataset for classification.
x, y = make_blobs(n_samples=N, centers=centers, random_state=42)

# Create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(k, weights='distance')
# Train the classifier.
clf.fit(x, y)
# Predict.
clf.predict(x)
```

Importa classe KNeighborsClassifier .

Cria duas classes de dados que se sobrepõem.

Número de vizinhos mais próximos a serem considerados.

Peso de cada vizinho é o inverso da distância para o exemplo de entrada.

Armazena conjunto de dados na memória.

Realiza predição.

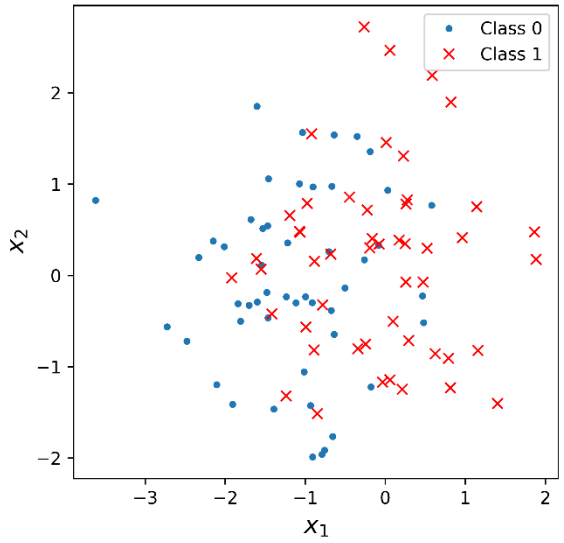


Figura com a distribuição dos exemplos de treinamento.

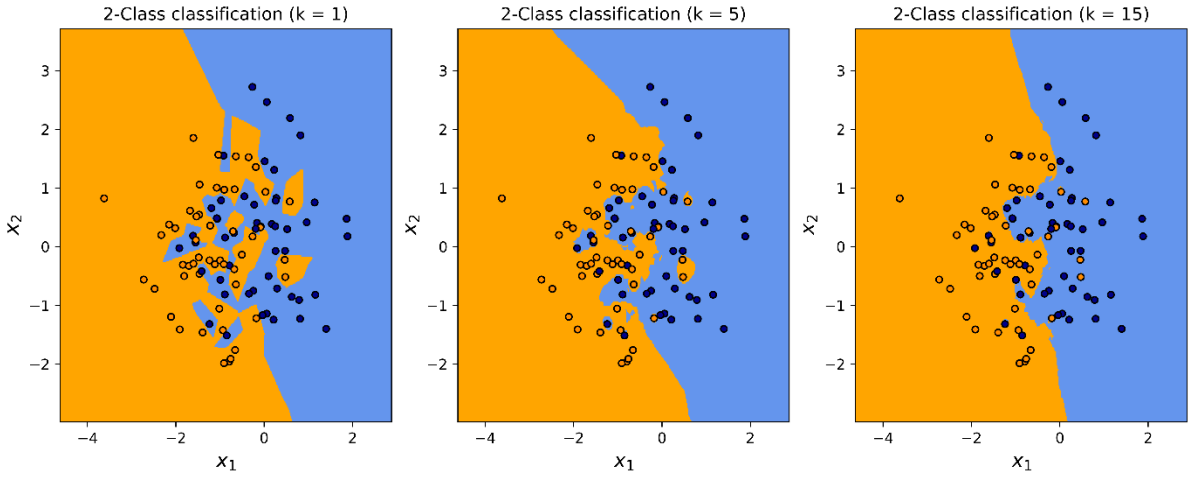


Figura mostra a fronteira de decisão criada pelo k-NN para diferentes valores de k. Como podemos ver, à medida que k aumenta, a fronteira tende a ficar mais suave e menos regiões isoladas são criadas para cada classe.

Exemplo: knn_classification_2_classes.ipynb

k-NN para regressão

- Seja $N_k(\boldsymbol{x}')$ o conjunto formado pelos k exemplos de treinamento $\boldsymbol{x} \in \mathbb{R}^{K\times 1}$ mais próximos ao exemplo de entrada $\boldsymbol{x}'$. As saídas associadas a estes exemplos de treinamento são denotadas por $y_j\big(\boldsymbol{x} \in N_k(\boldsymbol{x}')\big), j = 1, \dots, k$.
- Desta forma, quando utilizado para ***regressão***, a saída do algoritmo k-NN para um novo exemplo de entrada $\boldsymbol{x}'$ pode ser escrita de forma geral como

$$\hat{y}(\boldsymbol{x}') = \frac{\sum_{j=1}^{k} w_j y_j\big(\boldsymbol{x}\in N_k(\boldsymbol{x}')\big)}{\sum w_j},$$

onde $w_j, j = 1, \dots, k$ representa o peso associado ao *j*-ésimo vizinho de $\boldsymbol{x}'$.

- Os pesos associados aos vizinhos podem ser ***uniformes*** ou ***inversamente proporcionais à distância.***

Exemplo: Regressão k-NN com SciKit-Learn

```
# Import all necessary libraries.
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsRegressor

# Generate sample data.
N = 40
np.random.seed(42)
X = np.sort((6*np.random.rand(N, 1) - 3), axis=0)
T = np.linspace(-3, 3, 100)[:, np.newaxis]
y = (0.5*X**2 + X + 2).ravel()
y_orig = (0.5*X**2 + X + 2).ravel()

y += np.random.randn(N)

# Fit regression model
n_neighbors = 5

plt.figure(figsize=(15,5))
for i, weights in enumerate(['uniform', 'distance']):
    knn = KNeighborsRegressor(n_neighbors, weights=weights)
    y_ = knn.fit(X, y).predict(T)

    plt.subplot(1, 2, i + 1)
    plt.scatter(X, y, color='darkorange', label='noisy data')
    plt.plot(X, y_orig, color='red', label='original data')
    plt.plot(T, y_, color='navy', label='prediction')
    plt.axis('tight')
    plt.legend()
    plt.title("KNeighborsRegressor (k = %i, weights = '%s')" % (n_neighbors, weights))
.
plt.show()
```

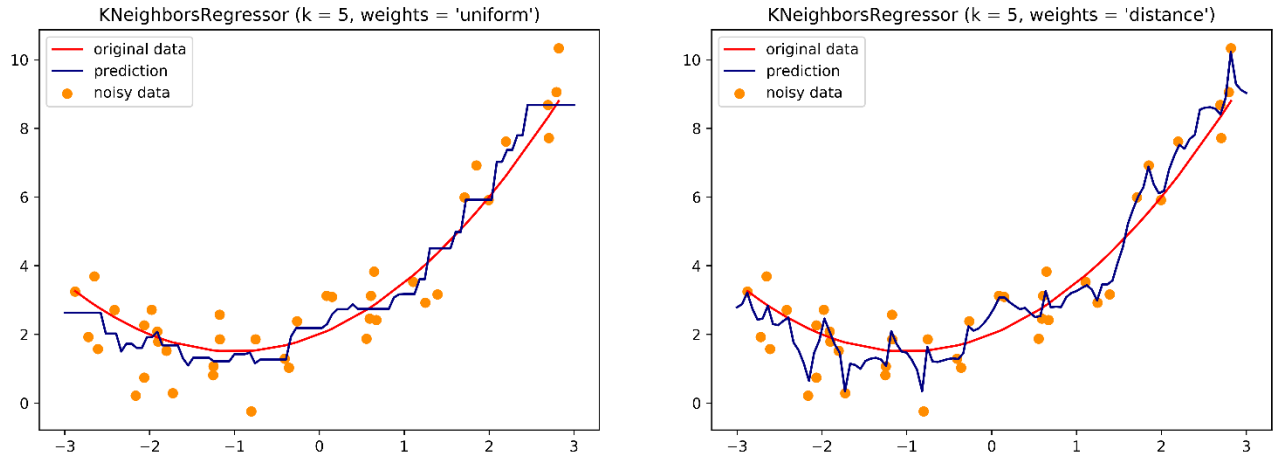
Importa classe KNeighborsRegressor.

Cria dados para regressão.

Peso de cada vizinho é uniforme ou o inverso da distância para o exemplo de entrada.

Armazena conjunto de dados na memória e realiza predição.

- A figura abaixo compara a regressão feita com o algoritmo k-NN quando os pesos associados aos vizinhos são ***uniformes*** (figura da esquerda) e ***inversamente proporcionais à distância*** (figura da direita).
- Pesos uniformes resultam em uma aproximação mais suave, pois o valor de saída será a média dos k valores, porém, com pesos inversamente proporcionais à distância, amostras próximas ao exemplo de entrada terão grande influência no valor de saída, fazendo com que ele seja bem próximo desse valor.



Exemplo: knn_regression.ipynb

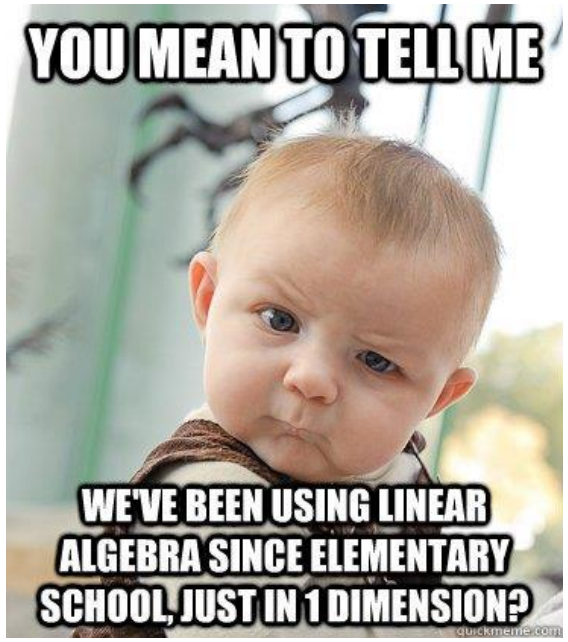
Avisos

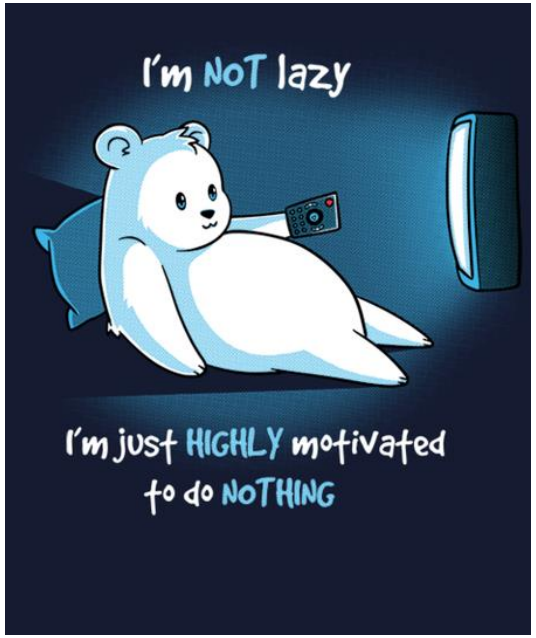
- Lista de exercícios #6 e exemplos já estão disponíveis.
- Façam os exercícios:
  - Exercício 1
  - Exercício 5

Obrigado!


KNN BE LIKE
"Show me your friends, and I'll tell you who you are."


*le teacher
today, we're going to learn about matrix..
EXPECTATION...
REALITY...
Share your fun on 9GAG.COM


YOU MEAN TO TELL ME
WE'VE BEEN USING LINEAR ALGEBRA SINCE ELEMENTARY SCHOOL, JUST IN 1 DIMENSION?
quickmeme.com


I'm NOT lazy
I'm just HIGHLY motivated to do NOTHING


THESE HYPERPARAMETERS
THEY ARE KIND OF A BIG DEAL

Figuras