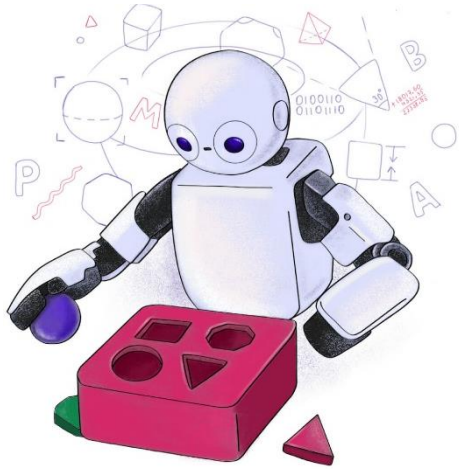
T319 - Introdução ao Aprendizado de Máquina:

Regressão Linear: Escalonamento de Atributos



Inatel

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

Recapitulando

- Vimos que a escolha do passo de aprendizagem influencia muito no processo aprendizagem do gradiente descendente.
  - Valores pequenos fazem com que o algoritmo tenha convergência muito lenta.
  - Valores grandes fazem com que o algoritmo divirja.
- Gráfico do erro em função das iterações nos ajuda a depurar o algoritmo.
- Além do ajuste manual, quando usamos GDE ou GD em mini-batches, precisamos reduzir o valor do passo de aprendizagem ao longo das iterações para garantir a convergência e estabilizaçãod do GD.
- Neste documento, veremos um tipo de ***pré-processamento*** bastante importante para algoritmos de ML que usem métricas de distância como função de erro.
  - **Pré-processamento**: Técnicas aplicadas aos dados de treinamento antes do treinamento.

Escalonamento de Atributos

- Em algumas situações, alguns atributos acabam sendo dominantes sobre os demais no sentido de que exercerem grande influência sobre o ***erro*** cometido pelo modelo.
- Isto pode ocorrer devido à grande diferença de magnitude entre os atributos.
- Essa diferença entre as magnitudes afeta o desempenho de algoritmos de ML que utilizam métricas de distância como função de erro.
  - As diferenças entre as magnitudes dos atributos faz com que as superfícies de erro tenham formato de vale, dificultando a convergência dos algoritmos.

Escalonamento de Atributos

- Dada a seguinte equação hipótese, $h(\boldsymbol{x})$

$$\hat{y}(n) = h\big(\boldsymbol{x}(n)\big) = a_1 x_1(n) + a_2 x_2(n).$$

- A função de erro é dada por

$$J_e(\boldsymbol{a}) = \frac{1}{N}\sum_{n=0}^{N-1}\Big[y_{\text{noisy}}(n) - \big(a_1 x_1(n) + a_2 x_2(n)\big)\Big]^2.$$

- Caso $x_1(n) \gg x_2(n), \forall i$, então x_1 tem uma influência maior no erro resultante, o que pode ser expresso de forma aproximada como

$$J_e(\boldsymbol{a}) \approx \frac{1}{N}\sum_{n=0}^{N-1}\Big[y_{\text{noisy}}(n) - a_1 x_1(n)\Big]^2.$$

- Portanto, o erro entre y e $h\big(\boldsymbol{x}(n)\big)$ será dominado pelo atributo x_1.

Escalonamento de Atributos

Função objetivo:

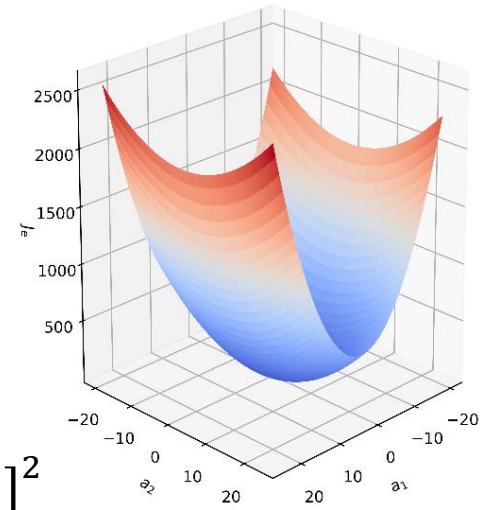
$$y(n) = a_1 x_1(n) + a_2 x_2(n),$$

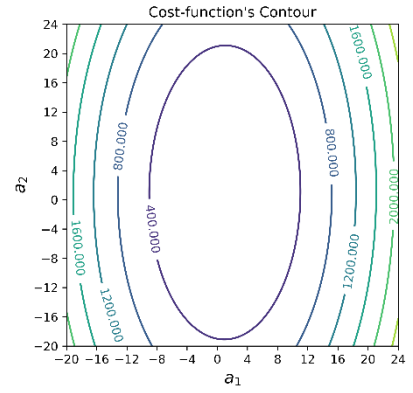
onde $a_1 = 1, a_2 = 1$.

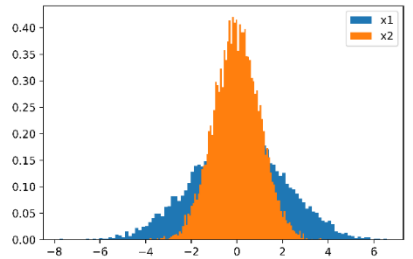
Para plotar a superfície de erro usamos:

$$J_e(\boldsymbol{a}) = \frac{1}{N}\sum_{n=0}^{N-1}\left[y_{\text{noisy}}(n) - (\widehat{a_1}x_1(n) + \widehat{a_2}x_2(n))\right]^2$$

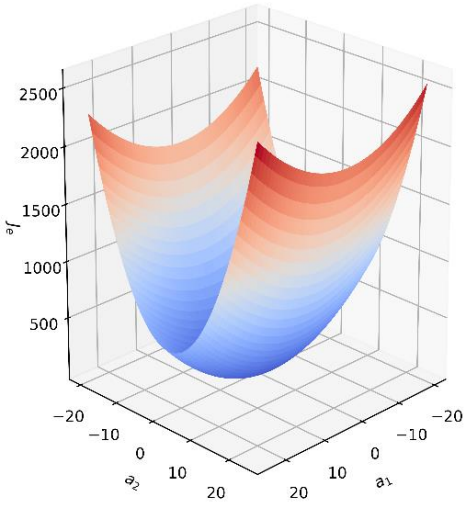
- $x_1 \gg x_2$: erro varia mais rapidamente com variações de $\widehat{a_1}$, resultando num vale.
- A mesma coisa pode ser dita para x_2 e $\widehat{a_2}$ (vale).
- Quando x_1 e x_2 têm intervalo semelhante, então, a variação tanto de $\widehat{a_1}$ quanto de $\widehat{a_2}$ tem ***pesos*** semelhante na variação do erro (tigela).

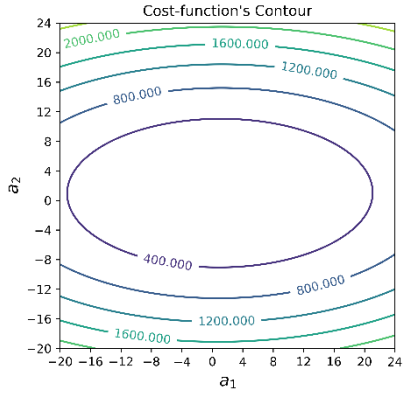


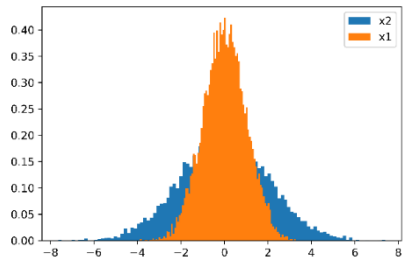




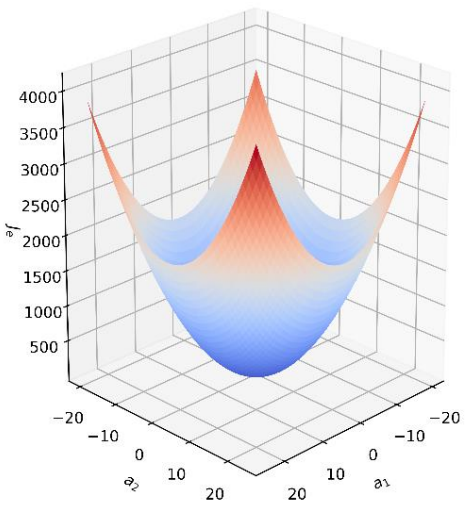
$x_1 = 2 * \text{randn}(\text{M}, 1)$
$x_2 = \text{randn}(\text{M}, 1)$

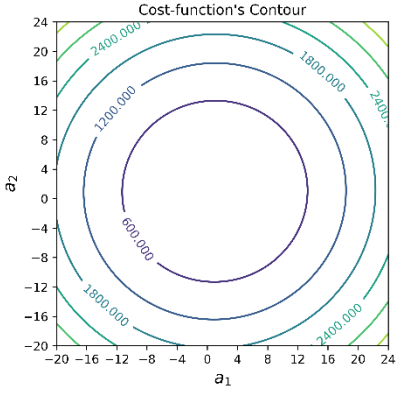


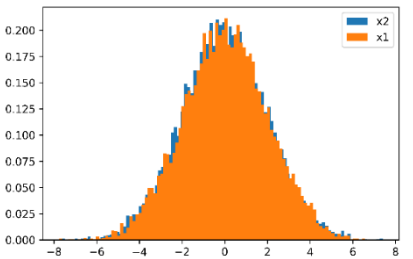




$x_1 = \text{randn}(\text{M}, 1)$
$x_2 = 2 * \text{randn}(\text{M}, 1)$







$x_1 = 2 * \text{randn}(\text{M}, 1)$
$x_2 = 2 * \text{randn}(\text{M}, 1)$

Exemplo: formatos_diferentes_da_superfície_de_erro.ipynb

Escalonamento de Atributos

- O que pode ser feito?
- Para evitar esse problema, o intervalo de variação de todos os ***atributos*** deve ser ***escalonado*** para que cada ***atributo*** contribua com o mesmo peso para o cálculo do erro.
- As duas formas mais comuns de escalonamento são:
  - **Normalização Mín-Max**

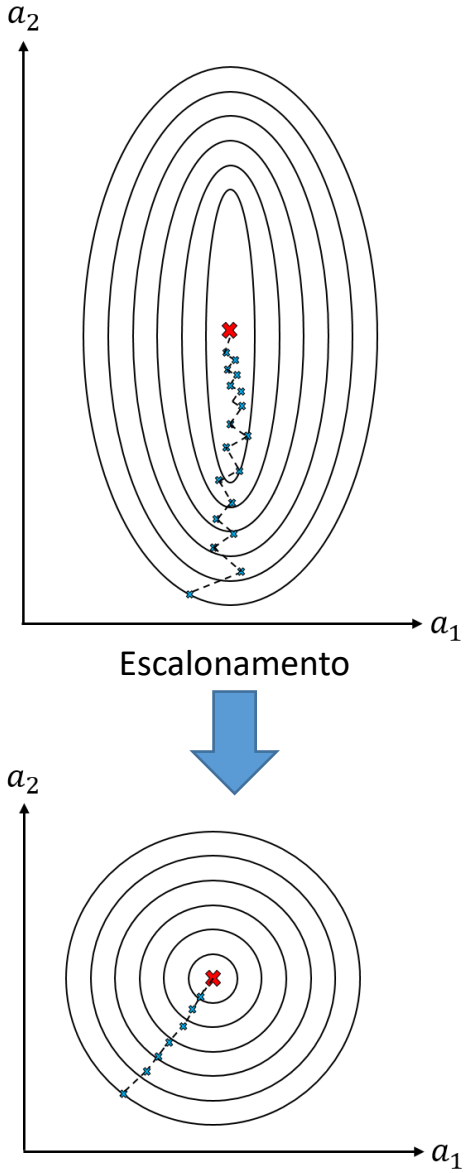
$$x'_k(i) = \frac{x_k(i) - \min(\boldsymbol{x}_k)}{\max(\boldsymbol{x}_k) - \min(\boldsymbol{x}_k)}, 0 \leq x'_k(i) \leq 1$$

  - **Padronização**

$$x'_k(i) = \frac{x_k(i) - \mu_{\boldsymbol{x}_k}}{\sigma_{\boldsymbol{x}_k}}$$

- ***Normalização mín-max*** faz com que os atributos variem entre 0 e 1.
- ***Padronização*** faz com que os atributos tenham média zero e desvio padrão unitário. Observe que, neste caso, os valores não ficam restritos a um intervalo específico.

Escalonamento de Atributos



- Ajuda a acelerar a convergência do ***gradiente descendente*** pois deixa as curvas de nível da superfície de erro mais circulares.
- Ajuda a estabilizar os algoritmos de aprendizado de máquina.
- Possibilita comparar o peso/influência de cada ***atributo*** no modelo.
- Observações:
  - Quando temos um conjunto de validação/teste do modelo, aplica-se ao conjunto de validação o escalonamento com os parâmetros (min, max, média, variância) obtidos com o conjunto de treinamento.
  - Em alguns casos, o escalonamento também é aplicado aos rótulos, i.e., aos valores de y. Mas não se esqueça de desfazer o escalonamento para realizar predições que sejam significativas.

Escalonamento de Atributos: **Exemplo**

- Função geradora:

$$y = x_1 + x_2,$$

onde $x_1 \sim N(10, 100)$, $x_2 \sim N(0, 1)$ e $a_1 = a_2 = 1$.

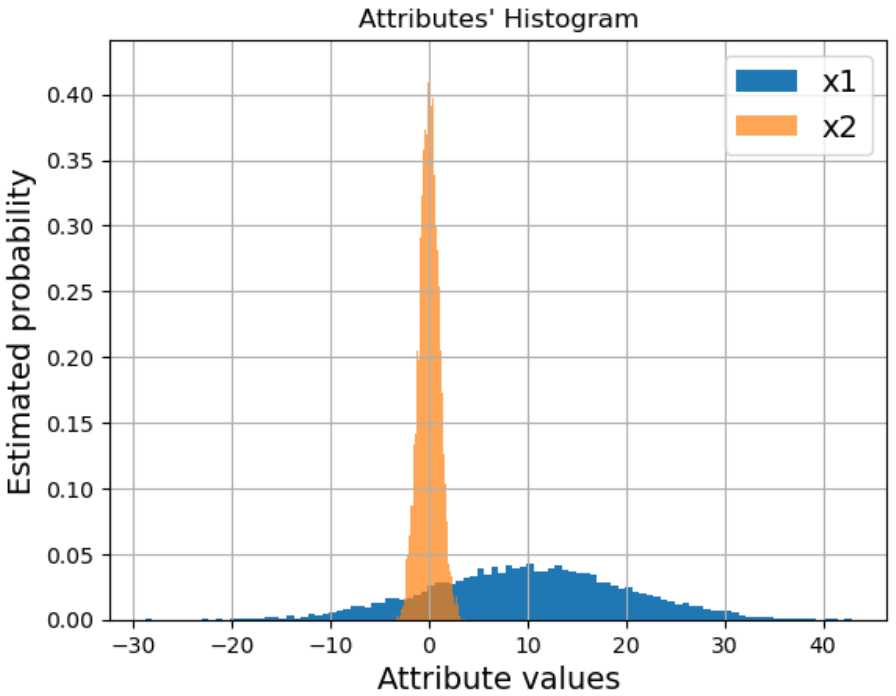
- Função observável ruidosa:

$$y_{\text{noisy}} = y + w,$$

onde $w \sim N(0, 1)$

- Função hipótese:

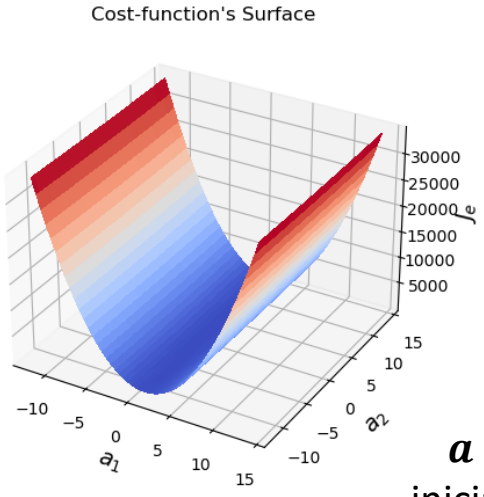
$$\hat{y} = \widehat{a_1} x_1 + \widehat{a_2} x_2.$$

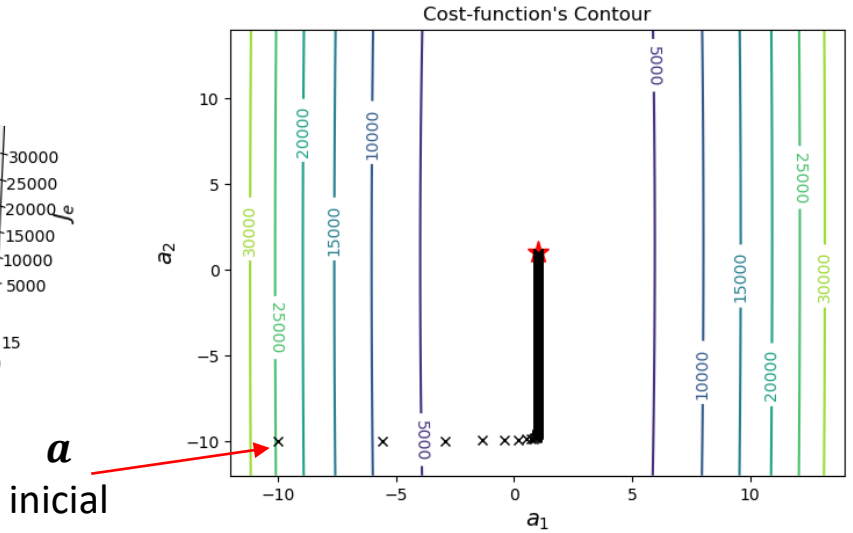


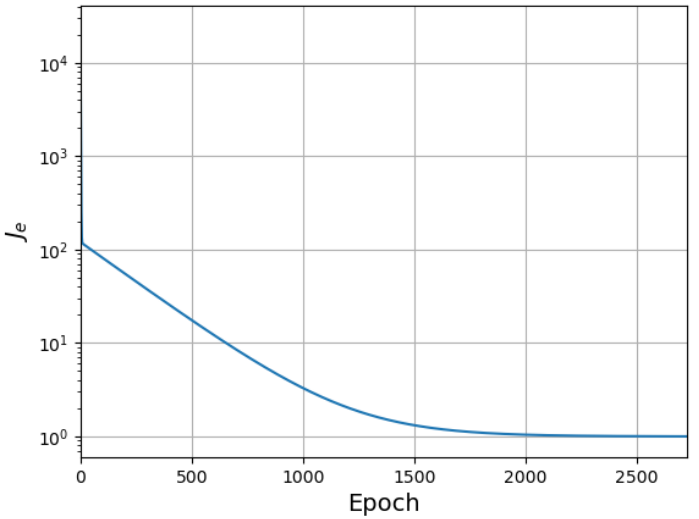
Exemplo: escalonamento_de_atributos_com_scikit_learn.ipynb

Escalonamento de Atributos: **Exemplo**

- Superfície de erro tem formato de "U", com maior taxa de variação do erro, i.e., inclinação, na direção de a_1.
- A taxa de variação do erro é praticamente constante na direção de a_2 (região com inclinação de $\approx 0°$).
- Como o gradiente na direção de a_2 é muito pequeno, o treinamento fica lento.
- Algoritmo GD em batelada converge após mais de 2000 épocas.

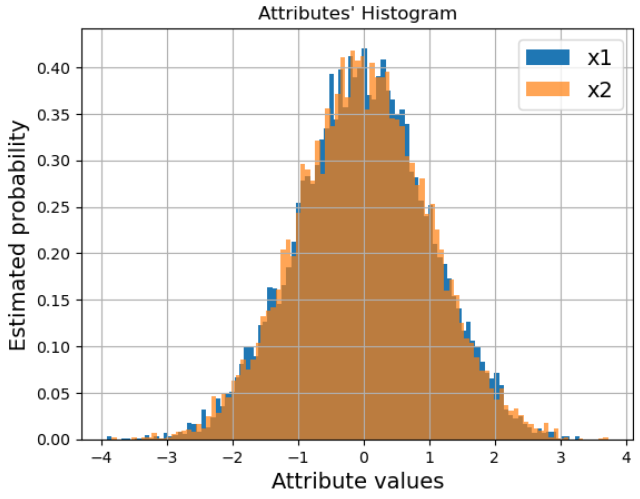


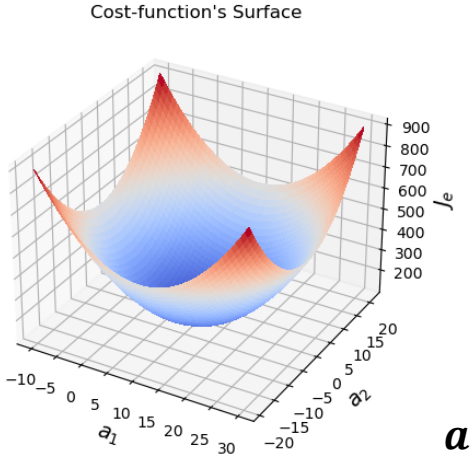


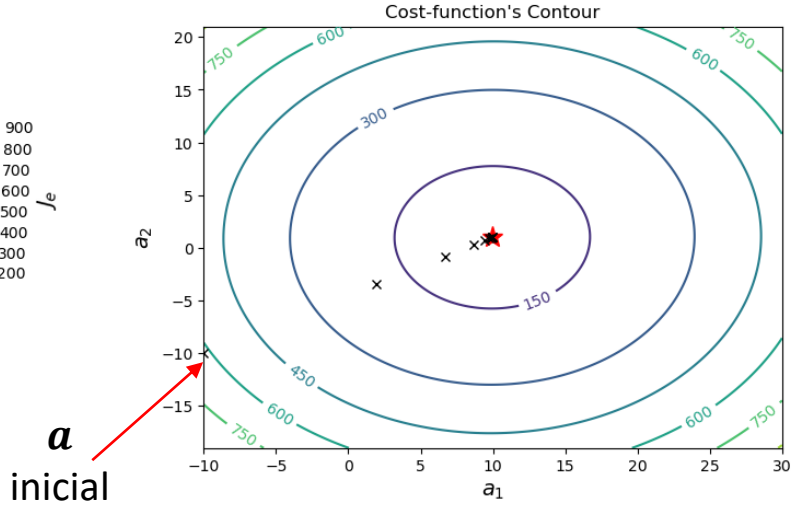


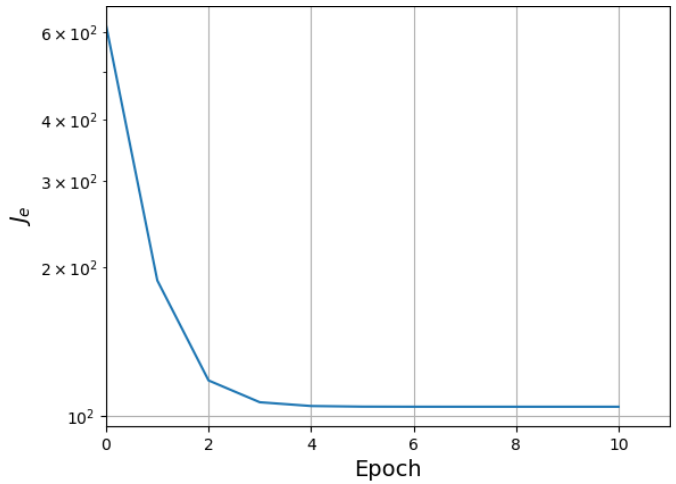
Escalonamento de Atributos: **Exemplo**

- Agora aplicamos ***padronização*** aos atributos.
- Após a padronização, a superfície passa a ter o formato de uma "*tigela*".
- As linhas de contorno se tornam mais "circulares", denotando que a superfície tem inclinação similar em todas as direções.
- Nesse exemplo, o algoritmo converge após 4 épocas.
- O treinamento se torna mais rápido, pois a inclinação da superfície se torna mais íngreme em todas as direções.





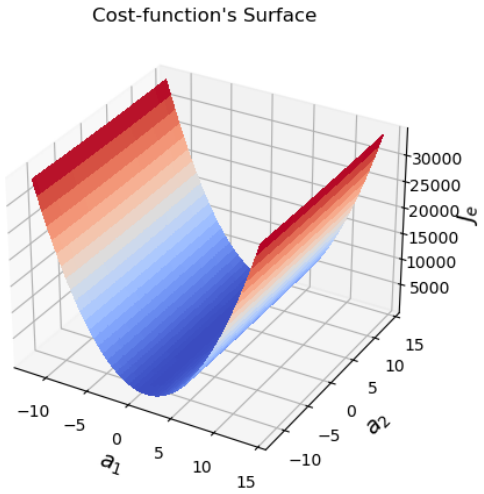


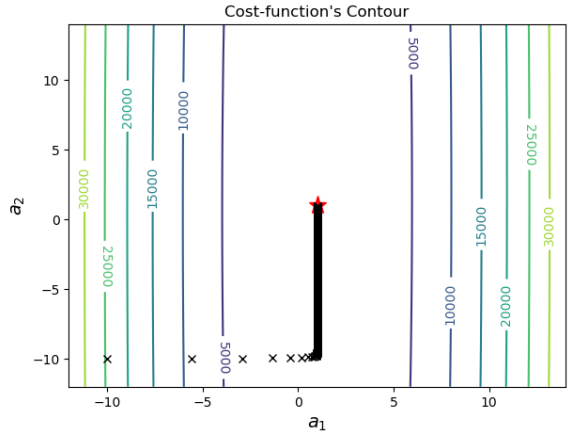


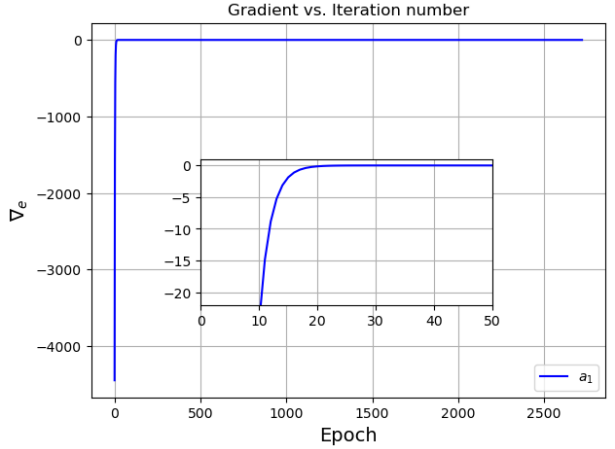
Escalonamento de Atributos: **Exemplo**

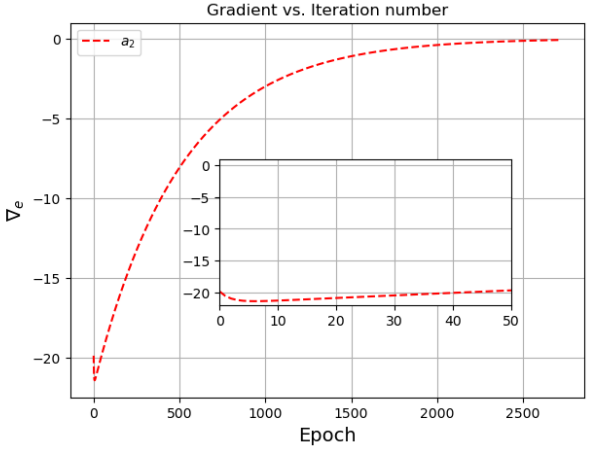
variação do vetor gradiente

Sem escalonamento



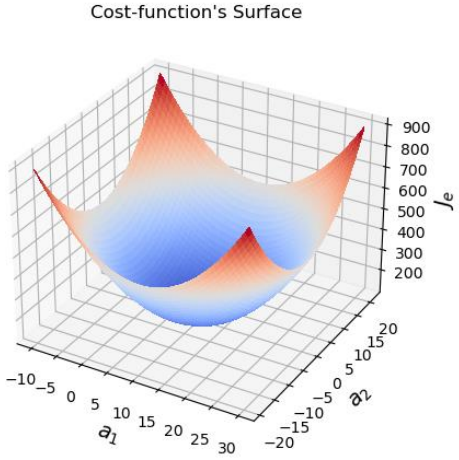


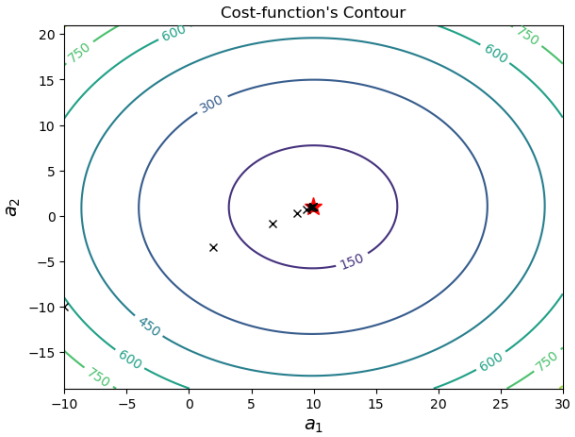


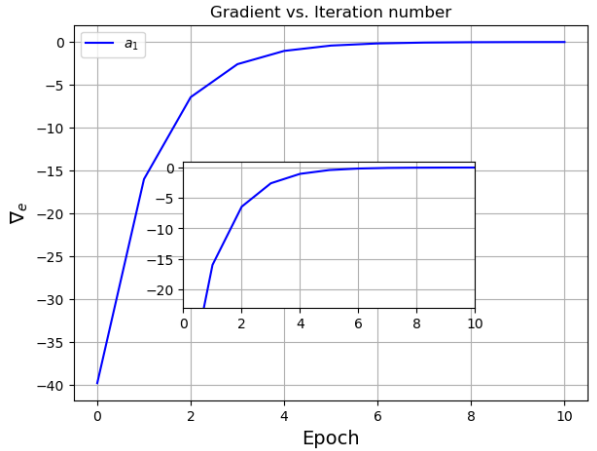


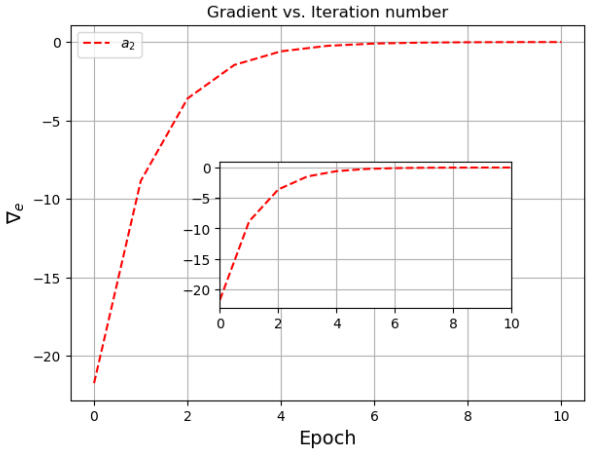
Pesos de atributos com variação muito grande são atualizados mais rapidamente do que pesos de atributos com variação pequena. x_1 contribui muito mais no valor final do erro, fazendo com que a_1 seja rapidamente atualizado, tendendo a seu valor correto mais rapidamente.

Padronização



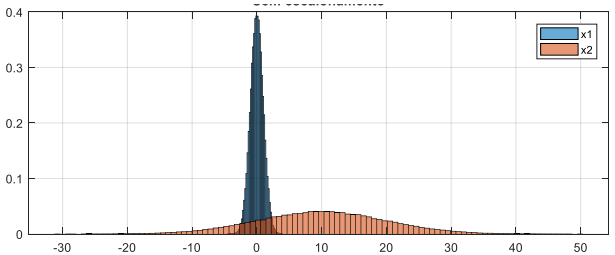


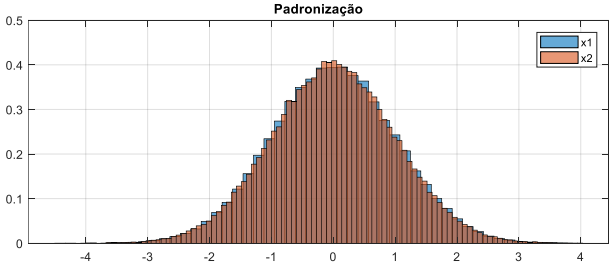




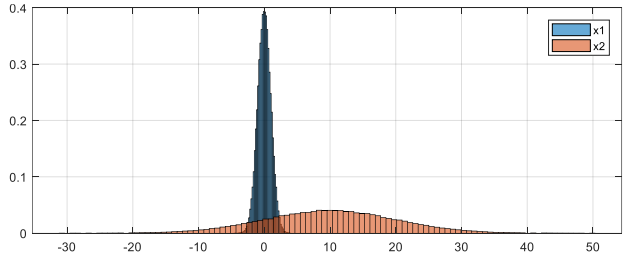
Escalonamento de atributos com a biblioteca SciKit-Learn

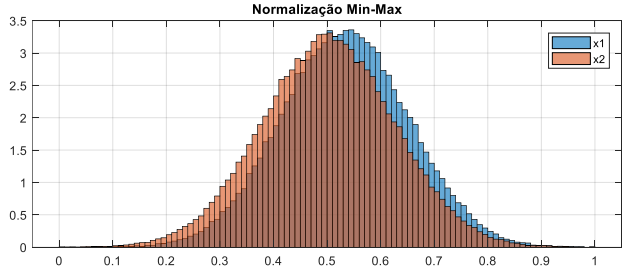
```
# Import Class StandardScaler from module Preprocessing of library sklearn
responsible for standardizing the data.
from sklearn.preprocessing import StandardScaler

# Instantiate a Standard scaler.
stdScaler = StandardScaler()

# Concatenate both column vectors.
X = np.c_[x1, x2]

# Standardize the features.
scaled_X = stdScaler.fit_transform(X)
```



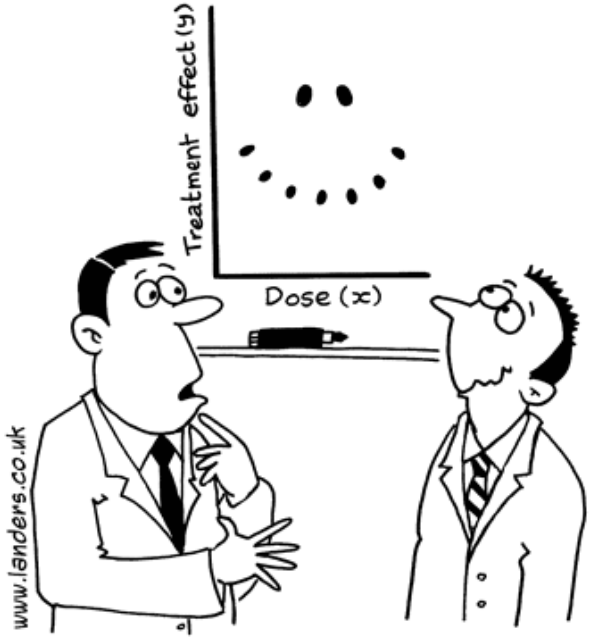



```
# Import Class MinMaxScaler from module Preprocessing of library sklearn
responsible for normalizing the data.
from sklearn.preprocessing import MinMaxScaler

# Instantiate a MinMax scaler.
minMaxScaler = MinMaxScaler()

# Concatenate both column vectors.
X = np.c_[x1, x2]

# Standardize the features.
scaled_X = minMaxScaler.fit_transform(X)
```

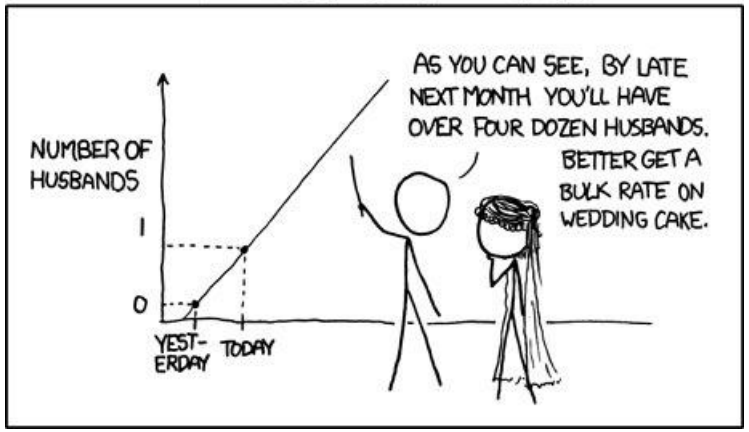


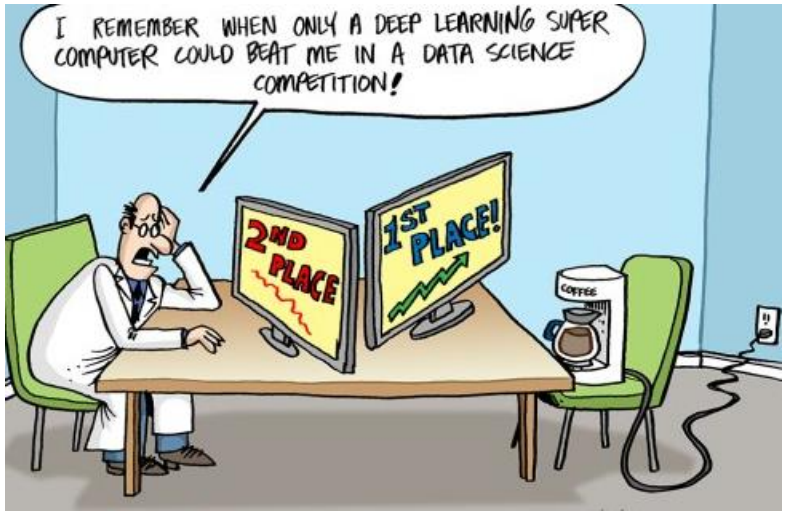


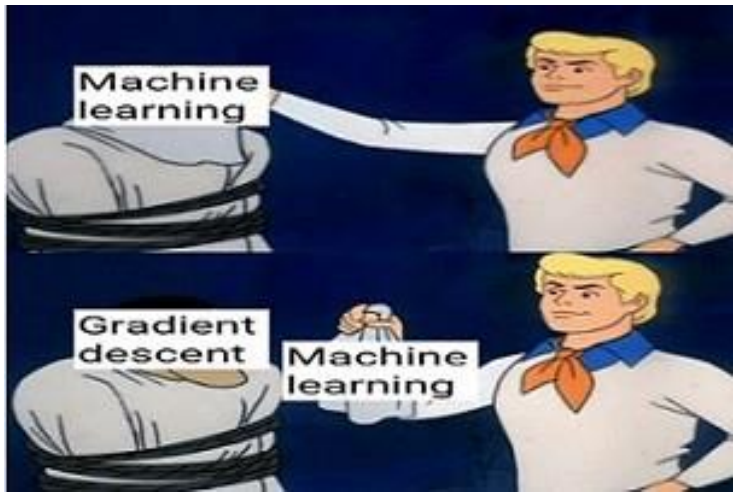
Obrigado!



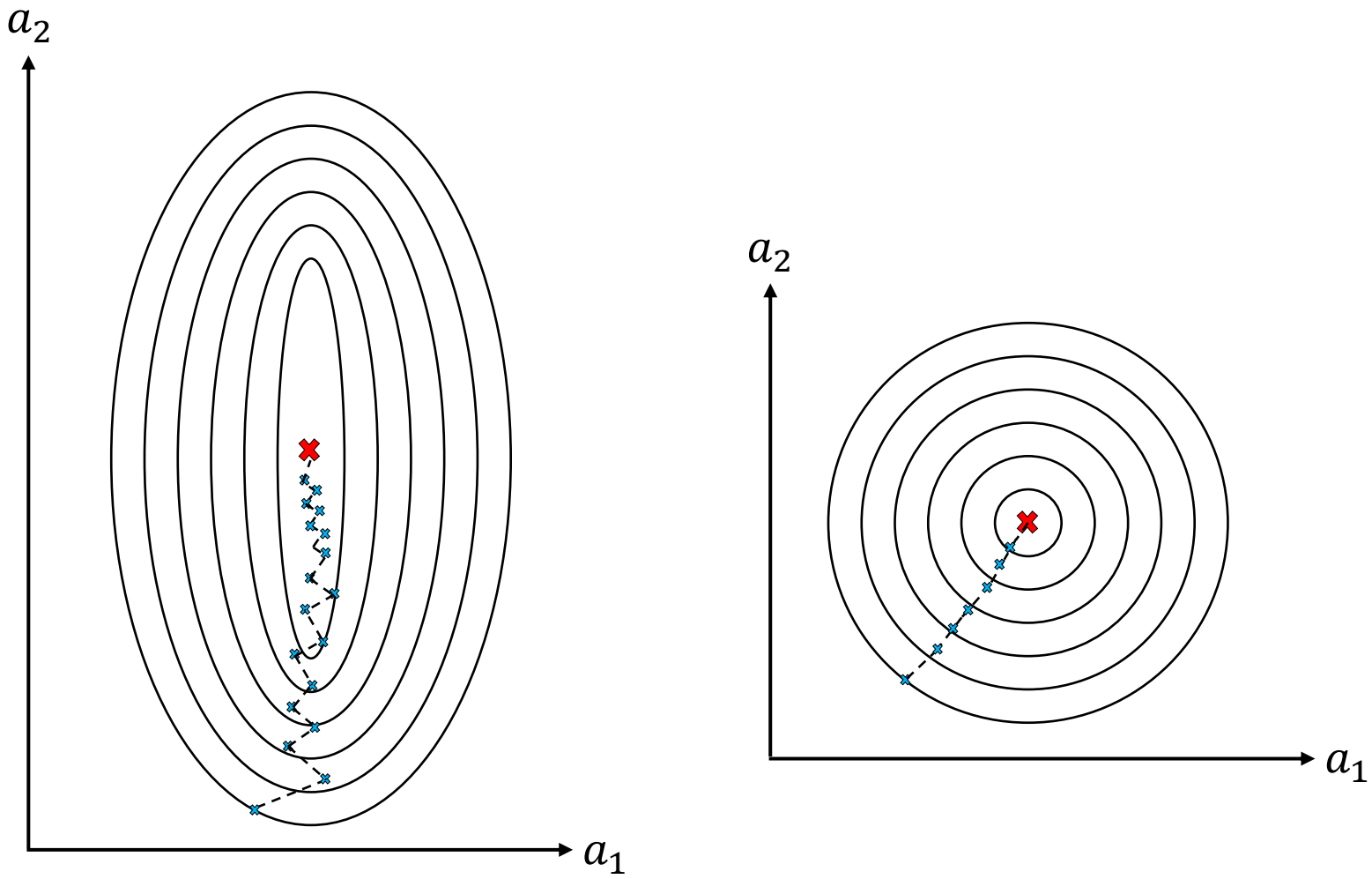
"It's a non-linear pattern with outliers.....but for some reason I'm very happy with the data."

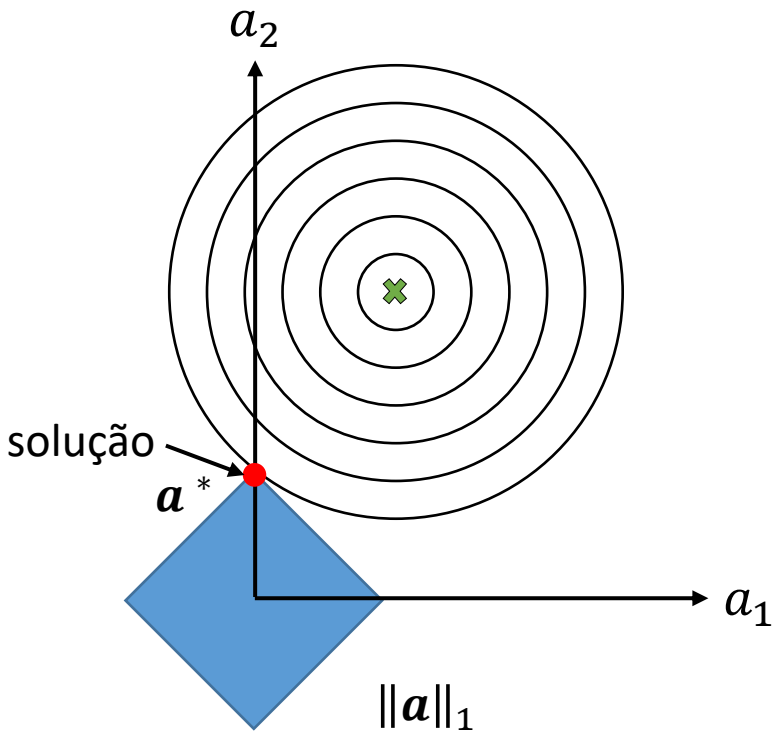


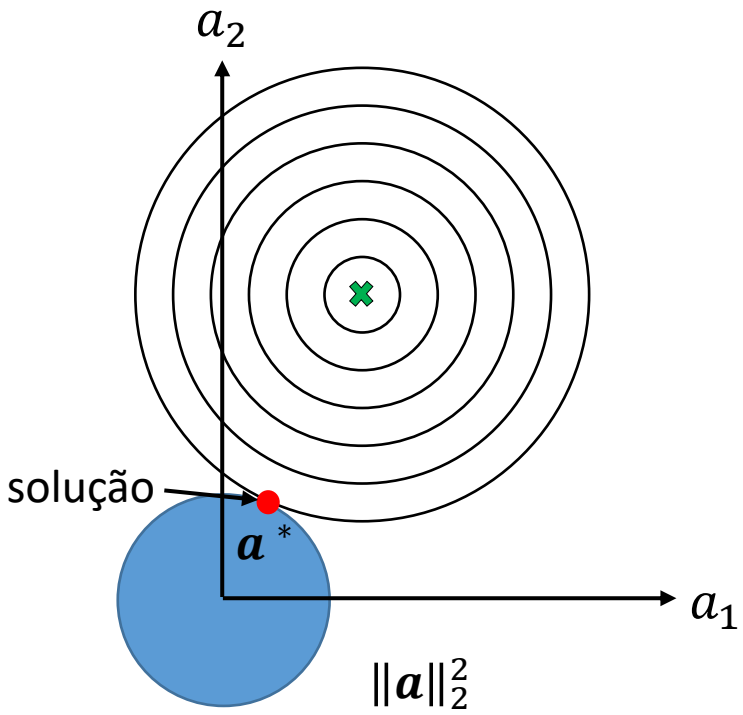


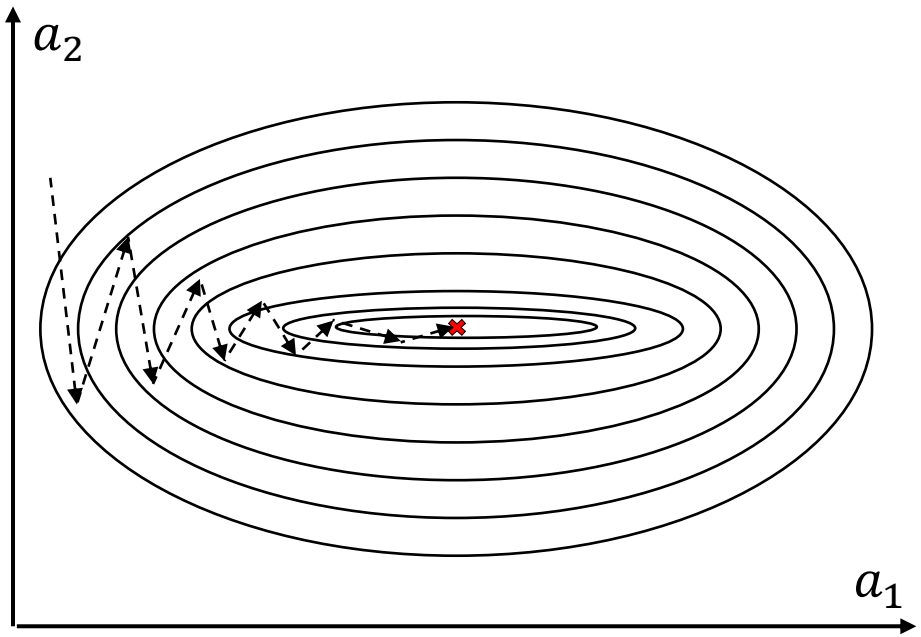


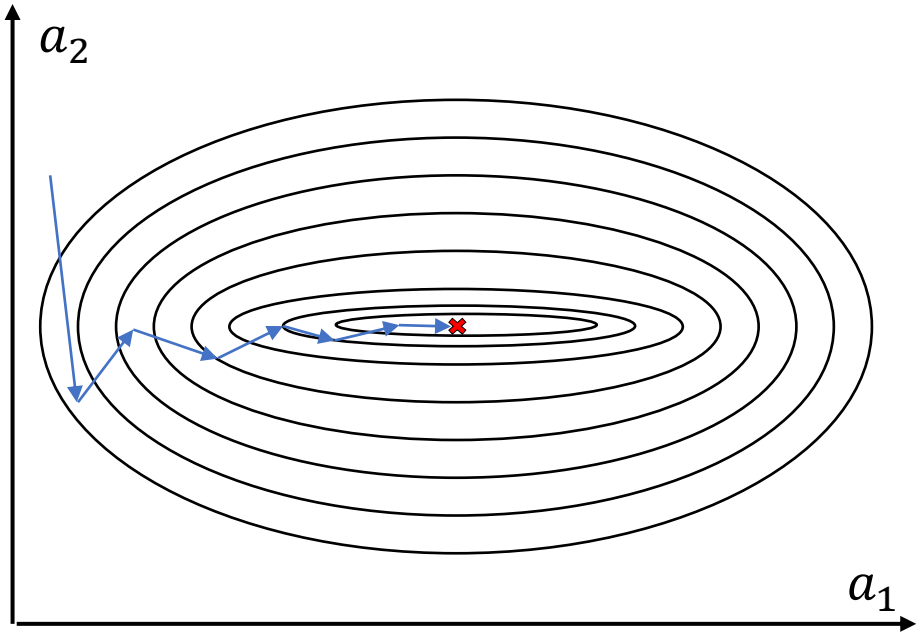
FIGURAS


a_2
a_1
a_2
a_1


a_2
solução
$\boldsymbol{a}^*$
a_1
$\|\boldsymbol{a}\|_1$


a_2
solução
$\boldsymbol{a}^*$
a_1
$\|\boldsymbol{a}\|_2^2$


a_2
a_1


a_2
a_1