

# Regressão Linear

---

Advanced Institute for Artificial Intelligence – AI2

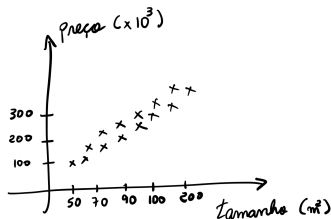
<https://advancedinstitute.ai>

## Regressão Linear Univariada

Em aprendizado de máquina, existem diversos problemas para os quais conhecemos um conjunto de valores de entrada e desejamos **estimar** o valor de saída. Um exemplo bastante comum é a precificação de imóveis, cujo valor de entrada corresponde ao tamanho de uma casa e a saída desejada é o seu preço.

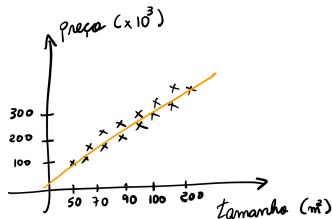
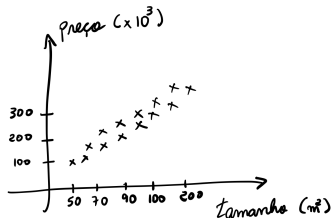
## Regressão Linear Univariada

Em aprendizado de máquina, existem diversos problemas para os quais conhecemos um conjunto de valores de entrada e desejamos **estimar** o valor de saída. Um exemplo bastante comum é a precificação de imóveis, cujo valor de entrada corresponde ao tamanho de uma casa e a saída desejada é o seu preço.

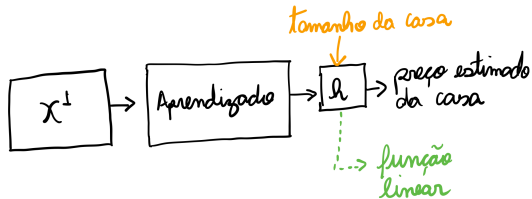


## Regressão Linear Univariada

Em aprendizado de máquina, existem diversos problemas para os quais conhecemos um conjunto de valores de entrada e desejamos **estimar** o valor de saída. Um exemplo bastante comum é a precificação de imóveis, cujo valor de entrada corresponde ao tamanho de uma casa e a saída desejada é o seu preço.



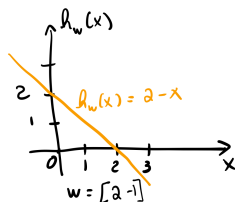
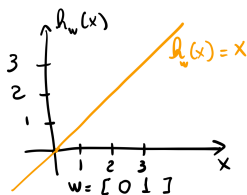
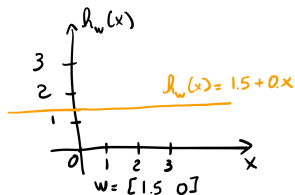
Definição do problema: seja um conjunto de dados  $\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \dots, (x_z, y_z)\}$  tal que  $x_i \in \mathbb{R}$  corresponde ao dado de entrada e  $y_i \in \mathbb{R}$  denota o seu respectivo valor de saída. Temos, ainda, que  $\mathcal{X}$  pode ser **particionado** da seguinte forma:  $\mathcal{X} = \mathcal{X}^1 \cup \mathcal{X}^2$ , em que  $\mathcal{X}^1$  e  $\mathcal{X}^2$  denotam os conjuntos de dados de **treinamento** e **teste**, respectivamente. Nosso objetivo é, dado o conjunto de treinamento, aprender uma função  $h : \mathbb{R} \rightarrow \mathbb{R}$  que consiga estimar o valor de uma casa dado o seu tamanho.



A técnica chama-se **Regressão Linear Univariada** porque aprendemos uma **função linear** utilizando apenas **uma** variável de entrada. Desta forma, temos a seguinte formulação para essa função:

$$h_{\mathbf{w}}(x) = w_0 + w_1x, \quad (1)$$

em que  $w_0$  e  $w_1$  denotam os parâmetros do modelo (equação da reta), podendo ser representados como  $\mathbf{w} = [w_0 \ w_1]$ .



Desta forma, como podemos observar, temos diferentes comportamentos para  $h_{\mathbf{w}}(x)$ , a qual é também chamada de **função hipótese**. A ideia principal da regressão linear é escolher o vetor de parâmetros  $\mathbf{w}$  tal que  $h_{\mathbf{w}}(x)$  seja a mais próxima possível das saídas das amostras de nosso conjunto de treinamento. Matematicamente falando, temos o seguinte problema de **minimização**:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i)^2 \right\}, \quad (2)$$

em que  $m$  denota o tamanho do conjunto de treinamento. Essa formulação corresponde ao conhecido **erro médio quadrático**, do inglês *mean squared error* (MSE).

A Equação 2 é usualmente chamada de **função de custo** ou **função de perda**, também conhecida por **erro médio quadrático**. Podemos simplificar a notação escrevendo a Equação 2 da seguinte forma:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \{J(\mathbf{w})\}, \quad (3)$$

em que

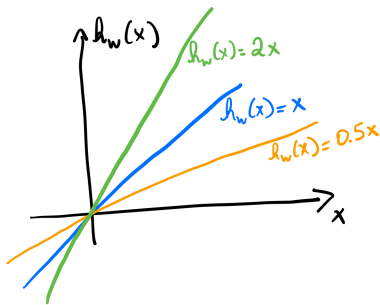
$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i)^2. \quad (4)$$

No entanto, vamos simplificar um pouco mais o problema e assumir que  $\mathbf{w} = [0 \ w_1]$ , ou seja, vamos assumir que  $w_0 = 0$  (a reta passa na origem do sistema de coordenadas).



Assim sendo, a Equação 1 pode ser escrita da seguinte forma:

$$h_w(x) = w_0 + w_1x = w_1x. \quad (5)$$



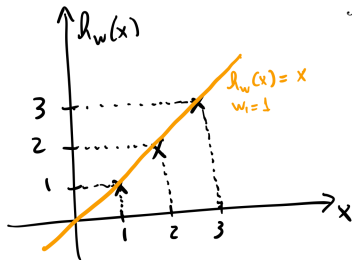
Assumindo, então, que  $w_0 = 0$ , podemos escrever a Equação 3 da seguinte forma:

$$\mathbf{w}^* = \arg \min_{w_1} \{J(w_1)\}. \quad (6)$$

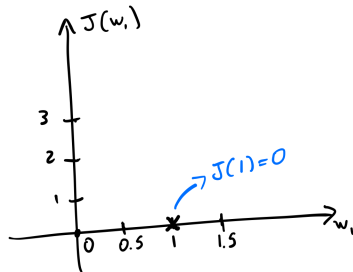
O nosso problema passar a ser, agora, encontrar um valor apropriado para  $w_1$  de tal forma que o valor de  $J(w_1)$  seja o menor possível. Resumindo, o que temos definido até então para o problema:

- Função hipótese:  $h_{\mathbf{w}}(x) \approx w_1 x$
- Função de custo:  $J(\mathbf{w}) \approx J(w_1)$

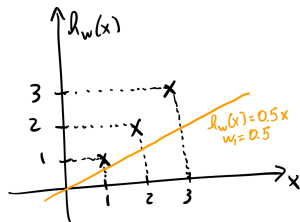
Vamos ilustrar algumas situações para entendermos o comportamento de ambas funções (hipótese e custo). Suponha o conjunto de treinamento  $\mathcal{X}^1 = \{(1, 1), (2, 2), (3, 3)\}$ . Neste caso, temos que  $m = 3$ . Ademais, suponha que  $h_{\mathbf{w}}(x) = x$ , ou seja,  $w_1 = 1$ .



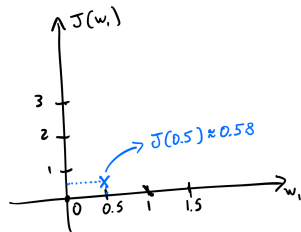
$$\begin{aligned}
 J(w_1) &= \frac{1}{2m} \sum_{i=1}^3 (h_{\mathbf{w}}(x_i) - y_i)^2 \\
 &= \frac{1}{2m} \sum_{i=1}^3 (w_1 x_i - y_i)^2 \\
 &= \frac{1}{2m} \sum_{i=1}^3 (x_i - y_i)^2 \\
 &= \frac{1}{2 \cdot 3} [(1 - 1)^2 + (2 - 2)^2 + (3 - 3)^2] \\
 &= 0
 \end{aligned}$$



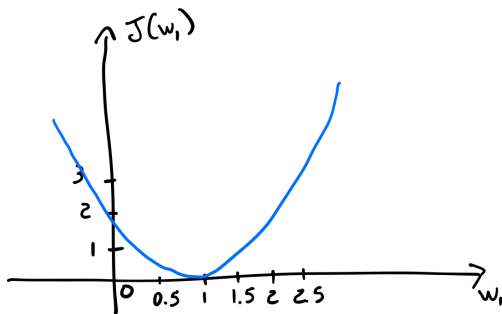
Agora, suponha que a nossa função hipótese é dada por  $h_w(x) = 0.5x$ , ou seja,  $w_1 = 0.5$ .



$$\begin{aligned} J(w_1) &= \frac{1}{2.3} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] \\ &= \frac{1}{2.3} (0.25 + 1 + 2.25) \\ &= \frac{1}{6} (3.5) \\ &\approx 0.58 \end{aligned}$$



Qual o comportamento da função de custo caso tenhamos diferentes valores para  $w_1$ ?



Desta forma, temos que  $w^* = w_1 = 1$  é o valor **ótimo**, ou seja, aquele que minimiza o valor de  $J(w_1)$ .

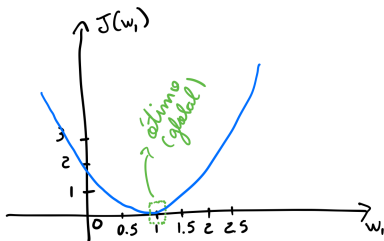
Uma alternativa ao uso do MSE é o chamado **erro médio absoluto**, do inglês *mean absolute error* (MAE). Mas qual o intuito de utilizarmos MSE? Notem que o formato da função de custo nos remete à uma função quadrática!

A pergunta agora é: como podemos escolher bons valores para  $\mathbf{w} = [w_0 \ w_1]$ ? Podemos usar o método da **força bruta** testando um número suficientemente grande valores de aleatórios para  $\mathbf{w}$ . É uma boa abordagem?

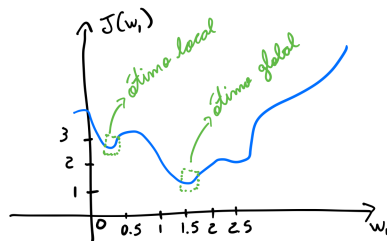
Uma solução é o uso da técnica conhecida por **gradiente descendente**, que faz a busca pelos valores de  $w$  baseando-se na **derivada** da função de custo. Como ele funciona?

- ➊ Atribua valores aleatórios para  $w = [w_0 \ w_1]$ .
- ➋ Avalie a função de custo  $J(w)$ .
- ➌ Caso o **critério de parada tenha sido atingido**, vá para o passo 5.
- ➍ Atualize o vetor  $w$  e retorne ao Passo 2.
- ➎ Fim do algoritmo.

Qual o problema? As funções de custo geralmente não são **convexas**.



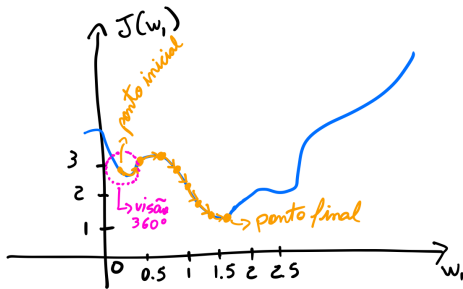
Função convexa



Função não convexa



Gradiente descendente busca a “melhor” solução a partir de um ponto inicial. Com base em informações dadas pela **derivada** da função, ele tende a encontrar um ótimo local ou global. Caso a função seja convexa, ele encontrará o ótimo global. Quais as informações que a derivada nos dá?

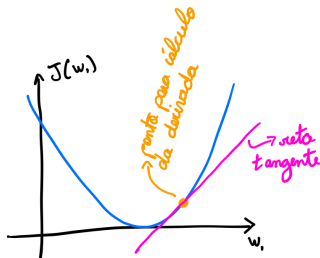


Matematicamente falando, podemos modelar a ideia por meio da **regra de atualização** dos parâmetros de  $\mathbf{w}$ :

$$w_j^{(t+1)} = w_j^{(t)} - \alpha \frac{\partial J(\mathbf{w})}{\partial w_j}, \quad (7)$$

em que  $j \in \{0, 1\}$  e  $\alpha$  corresponde à **taxa de aprendizado**. Já o termo  $\frac{\partial J(\mathbf{w})}{\partial w_j}$  denota a **derivada parcial**, que deverá ser calculada para cada parâmetro de nossa equação da reta.

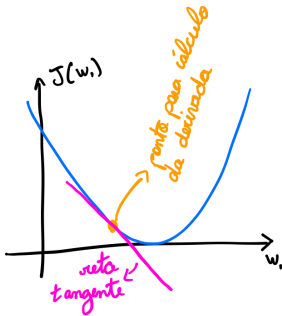
A questão principal que temos agora é: como calcular o termo derivativo, ou seja, as derivadas parciais? Para fins de simplicidade, suponha, novamente que  $w_0 = 0$ . Assim, nosso objetivo passa a ser o de minimizar  $J(w_1)$  para algum valor  $w_1 \in \mathbb{R}$ .



A derivada em um ponto nos retorna duas principais informações: **direção e magnitude**. A primeira é dada pela inclinação (sinal) da reta tangente no ponto que desejamos calcular a derivada.

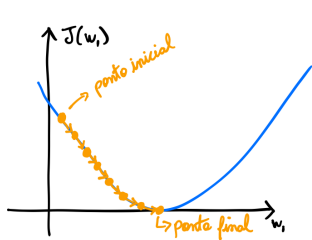
Neste exemplo, a inclinação é positiva, ou seja,  $\frac{\partial J(w)}{\partial w_j} > 0$ . Desta forma, estamos diminuindo o valor de  $w_1$ .

Já no exemplo abaixo, temos que a inclinação da reta é negativa, ou seja,  $\frac{\partial J(\mathbf{w})}{\partial w_j} < 0$ . Assim, estamos aumentando o valor de  $w_1$ .

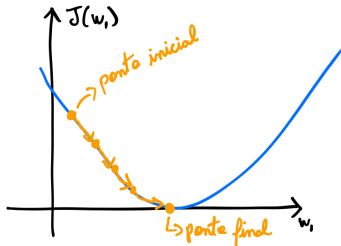


A informação de magnitude é o próprio valor retornado pelo termo derivativo, desconsiderando o sinal.

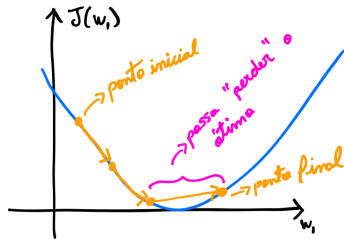
Qual a importância da taxa de aprendizado na Equação 7? Basicamente, baixos valores de  $\alpha$  nos levam à uma convergência mais lenta, ao passo que valores maiores podem acelerar esse processo. No entanto, precisamos tomar cuidado com essa escolha.



(a)  $\alpha$  pequeno

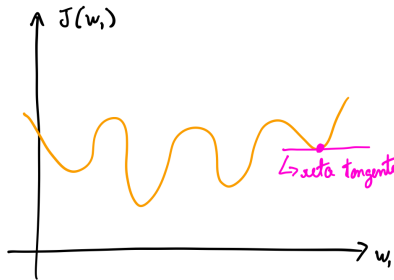
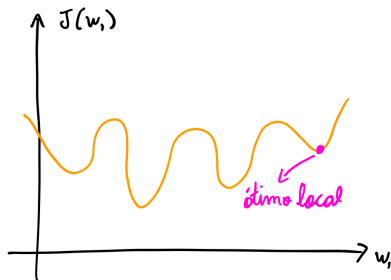


(b)  $\alpha$  adequado



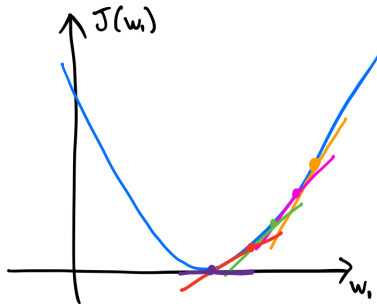
(c)  $\alpha$  grande

No entanto, o que acontece com o gradiente descendente se a inicialização ocorrer em um ótimo local? A inclinação é igual à 0!



$$\begin{aligned}w_1^{(t+1)} &= w_1^{(t)} - \alpha \frac{\partial J(\mathbf{w})}{\partial w_j} \\&= w_1^{(t)} - \alpha \cdot 0 \\&= w_1^{(t)}\end{aligned}$$

À medida que aproximamos do ponto ótimo, o próprio gradiente descendente vai tomando passos menores, pois o termo derivativo  $\frac{\partial J(\mathbf{w})}{\partial w_j} \rightarrow 0$ .



Precisamos, agora, calcular as derivadas parciais para  $\frac{\partial J(\mathbf{w})}{\partial w_j}$ . Temos que:

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i), \quad (8)$$

e

$$\frac{\partial J(\mathbf{w})}{\partial w_1} = \frac{1}{m} \sum_{i=1}^m [(h_{\mathbf{w}}(x_i) - y_i)x_i]. \quad (9)$$

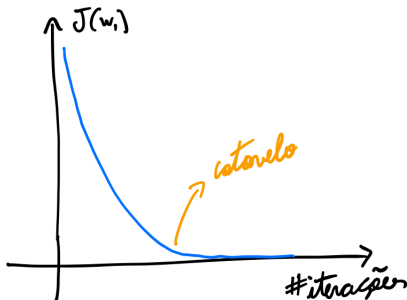


Desta forma, o algoritmo do gradiente descendente pode ser sumarizado da seguinte forma:

- ❶ Atribua valores aleatórios para  $\mathbf{w} = [w_0 \ w_1]$ .
- ❷ Avalie a função de custo  $J(\mathbf{w})$ .
- ❸ Caso o **critério de parada tenha sido atingido**, vá para o passo 7.
- ❹  $w_0^{(t+1)} = w_0^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i)$ .
- ❺  $w_1^{(t+1)} = w_1^{(t)} - \alpha \frac{1}{m} \sum_{i=1}^m [(h_{\mathbf{w}}(x_i) - y_i)x_i]$ .
- ❻ Retorne ao Passo 2.
- ❼ Fim do algoritmo.

Note que  $w_0$  e  $w_1$  precisam ser atualizados simultaneamente!

Quais são bons critérios de parada? Podemos utilizar um número fixo de iterações ou verificar quando o valor função de custo não é mais atualizado (erro não diminui).



Suponha, agora, que tenhamos mais variáveis para representar o problema, visto que utilizar apenas o tamanho da casa não é suficiente para estimarmos o seu valor (número de quartos, capacidade da garagem, piscina, etc).

Neste caso, assuma que  $n$  corresponde ao número dessas variáveis, ou seja, temos agora que  $\mathbf{x}_i \in \mathbb{R}^n$ . Assim sendo, nossa função hipótese possui a seguinte formulação:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x^1 + w_2x^2 + \dots w_nx^n \quad (10)$$

$$= w_0 + \sum_{j=1}^n w_jx^j. \quad (11)$$

Para fins de simplicidade, vamos assumir que  $x^0 = 1$ , ou seja,  $\mathbf{x} = [x^0 \ x^1 \ x^2 \ \dots \ x^n] \in \mathbb{R}^{n+1}$ . Assim sendo, a Equação 11 pode ser escrita da seguinte forma:

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 x^0 + \sum_{j=1}^n w_j x^j \quad (12)$$

$$= \mathbf{w}^T \mathbf{x}. \quad (13)$$

A equação acima é, na verdade, a equação do hiperplano, ou seja, uma reta multimensional. Desta forma, a regressão linear multivariada generaliza a regressão linear univariada para um número maior de dimensões.

Nosso problema agora passa a ter os seguintes itens:

- Função hipótese:  $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ .
- Função de custo:  $J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ .
- Derivada parcial:  $\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{m} \sum_{i=1}^m [(h_{\mathbf{w}}(\mathbf{x}_i) - y_i) x_i^j]$ .

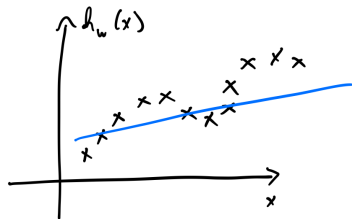
O algoritmo do gradiente descendente é o mesmo do apresentado para a regressão linear univariada.

Como podemos melhorar a convergência durante o aprendizado? Algumas opções:

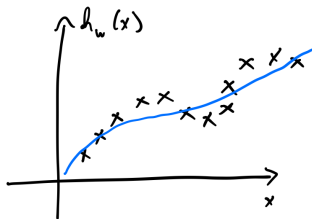
- Gradiente descendente em lotes (*mini-batches*) ou *online*.
- Gradiente descendente estocástico:
  - Bom para grandes conjuntos de dados.
  - Converge mais rápido do que o gradiente descendente (pode não avaliar todo o espaço de busca).
  - Parecido com o gradiente descendente *online*, porém é aplicado sobre amostragens aleatórias do conjunto de treinamento.

# Regularização

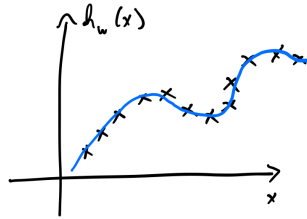
Com o intuito de evitar com que nossos modelos tornem-se muito "especializados", podemos fazer uso de técnicas de **regularização**. Vamos visualizar algumas as três situações que podem ocorrer quando estamos trabalhando com regressão de dados.



Subtreinamento

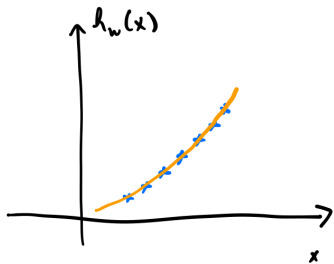


Bom treinamento

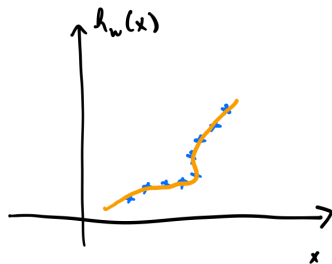


Sobretreinamento

Vamos observar dois exemplos. A tendência é que polinômios com graus maiores sejam mais propensos ao sobretreinamento, dado que conseguem aprender funções mais complexas.



$$h_w(x) = w_0 + w_1x + w_2x^2$$



$$h_w(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$$



Como podemos minimizar o efeito dos pesos associados aos graus maiores? Uma solução para penalizá-los seria multiplicá-los por valores altos. Vamos modificar a Equação 2, que corresponde ao erro médio quadrático:

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \left\{ \frac{1}{2m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i)^2 + \overbrace{500w_3}^{w_3 \approx 0} + \underbrace{500w_4}_{w_4 \approx 0} \right\}. \quad (14)$$

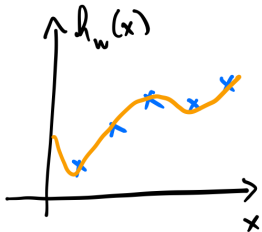
Como o objetivo é minimizar a função de custo, a tendência é que o gradiente descendente atribua valores muito pequenos para  $w_3$  e  $w_4$  no intuito de diminuir a sua influência no valor final. Assim, a regularização visa, de maneira geral, atribuir valores pequenos para os elementos em  $\mathbf{w}$  de tal forma a termos funções “mais simples”.

Desta forma, temos uma nova função de custo por erro médio quadrático, que agora adiciona um termo de regularização à Equação 4:

$$J(\mathbf{w}) = \frac{1}{2m} \left[ \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i)^2 + \lambda \sum_{j=1}^n w_j^2 \right], \quad (15)$$

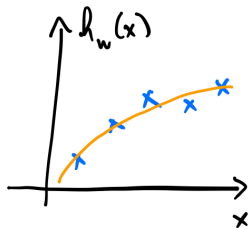
em que  $\lambda$  corresponde ao parâmetro de regularização, o qual controla a relação custo-benefício entre não regularização e sobretreinamento. **Lembrando que a regularização não se aplica à  $w_0$ .**

Vejamos qual é o significado prático da taxa de regularização.



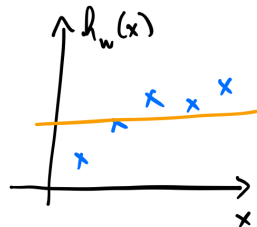
$$\lambda = 0$$

Sem regularização



$$0 < \lambda < \infty$$

Algum valor apropriado



$$\lambda \rightarrow \infty$$

$$h_w(x) = w_0$$

As derivadas parciais para  $\frac{\partial J(\mathbf{w})}{\partial w_j}$  dadas pelas Equações 8 e 9 agora podem ser formuladas da seguinte maneira:

$$\frac{\partial J(\mathbf{w})}{\partial w_0} = \frac{1}{m} \sum_{i=1}^m (h_{\mathbf{w}}(x_i) - y_i), \quad (16)$$

e

$$\frac{\partial J(\mathbf{w})}{\partial w_j} = \frac{1}{m} \sum_{i=1}^m [(h_{\mathbf{w}}(x_i) - y_i)x_i^j] + \frac{\lambda}{m} w_j. \quad (17)$$

Note que a derivada parcial com relação à  $w_0$  não sofre alterações. O algoritmo do gradiente descendente é o mesmo do que a versão da regressão linear sem a regularização.