Regressão Linear

A regressão linear é uma ferramenta usada para criar um modelo matemático que estabeleça a relação entre uma resposta escalar (variável dependente) a uma ou mais variáveis independentes. Essa relação é estabelecida estimando-se os parâmetros desconhecidos da função matemática (modelo), a partir de um conjunto de valores das varáveis dependente e independentes.

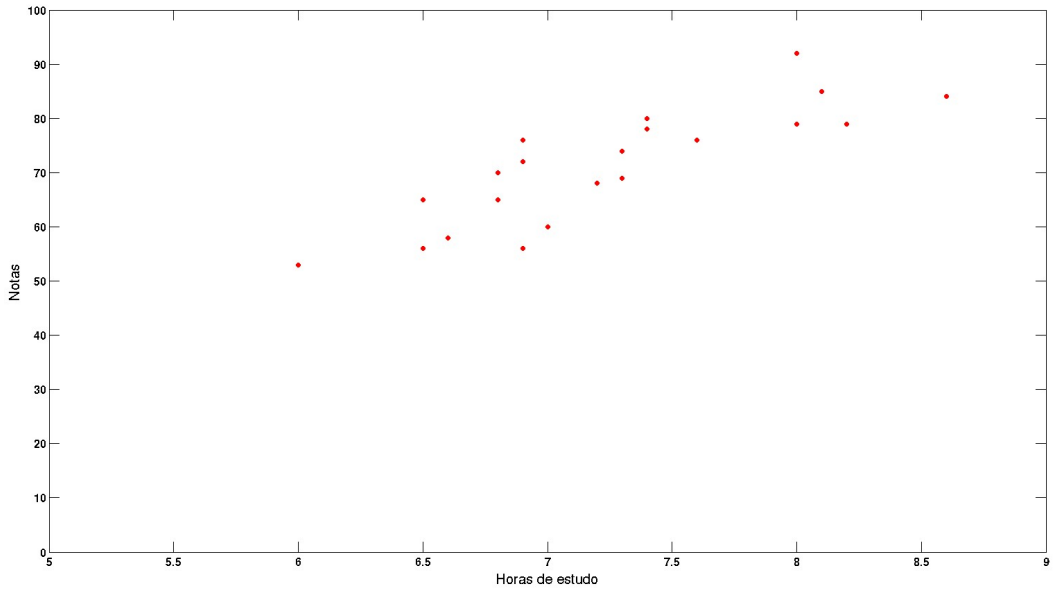
Em aprendizado de máquina, dizemos que o conjunto de dados contendo a correspondência entre as varáveis é chamado de **conjunto de treinamento** (*training set*). Por necessitar tanto dos valores das variáveis independentes, quanto da sua correspondência com a variável dependente, a regressão linear se enquadra como um **treinamento supervisionado**.

Regressão Linear de Uma Variável

É o processo de modelagem matemática relacionando somente uma variável independente a uma variável dependente. Por exemplo, considere a tabela abaixo.

6	53%
7	60%
6,5	56%
8	79%
6,6	58%
8,1	85%
6,8	70%
6,9	56%
7,3	69%
6,9	76%
8,2	79%
7,2	68%
7,3	74%
6,9	72%
8,6	84%
7,4	78%
7,6	76%
6,8	65%
8	92%
7,4	80%
6,5	65%

Nessa tabela, temos na coluna da esquerda o número de horas de estudo por semana que alunos do IFB se dedicaram à disciplina Álgebra Linear em um determinado semestre. Na coluna da direita, temos o aproveitamento (notas) desses alunos, como resultado do seu número de horas de estudo. Esses dados podem ser visualizados graficamente, colocando os valores das horas de estudo como variável independente (eixo horizontal) e seus respectivos rendimentos como variável dependente (eixo vertical)

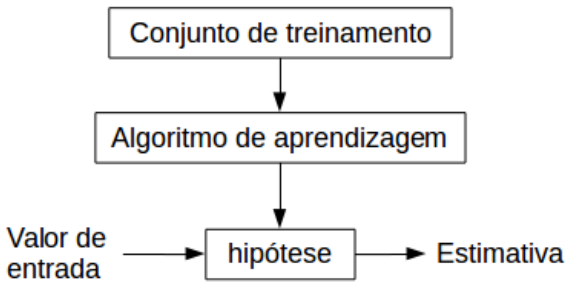


A partir desses dados, é possível estimar um modelo (função matemática) que preveja a nota de futuros alunos, considerando a quantidade de horas que eles dedicarão à disciplina.

No exemplo acima, temos dados de 21 alunos, portanto dizemos que temos 21 valores de entrada e 21 valores de saída. Assim, dizemos que o número de exemplos de treinamento é m = 21. Em outras palavras, podemos considerar os valores de entrada como um vetor $\mathbf{x}_m$ e os valores de saída como um vetor $\mathbf{y}_m$.

Aqui, faremos uma formalização das nomenclaturas utilizadas. Para referenciar um par de exemplos específico, será usado o superscrito no vetor. Por exemplo, seguindo os dados da tabela, $x^{(1)} = 6$ e $y^{(1)} = 53\%$, $x^{(2)} = 7$ e $y^{(2)} = 60\%$, e assim por diante. Portanto, um par de exemplo do conjunto de treinamento pode ser representado como $(x^{(i)},y^{(i)})$, onde *i* é o i-ésimo par. Ao usar a representação (x,y), sem o superscrito, estamos referenciando todo o vetor **x** e **y**.

Podemos então resumir a regressão linear como:



Onde a hipótese construída é deteminada a partir do mapeamento entre os vetores **x** e **y**, e que pode ser usada futuramente para estimar valores a partir de dados não presentes em **x**. Uma possível hipótese para esse exemplo, é uma função de primeiro grau, ou primeira ordem:

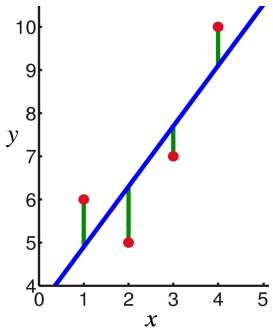
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Na equação acima, o subscrito θ na função h(x) é usado para deixar explícito que a hipótese é parametrizada pelos parâmetros θ_0 e θ_1, além de ser uma função de x. Assim, a regressão linear consiste em determinar os parâmetros do modelo (θ_0 e θ_1) que melhor se ajustam aos dados de treinamento.

Função Custo

Considerando a hipótese do exemplo anterior, formulada a partir de uma equação de primeiro grau, ou seja, graficamente é a equação de uma reta, é esperado que não será possível determinar valores para θ_0 e θ_1 que representem exatamente a correspondência da tabela anterior (conjunto de treinamento). Portanto, é preciso achar os valores dos parâmetros que **melhor** se ajuste aos dados de treinamento.

Considerando o termo "melhor ajuste", se faz necessário quantificar esse ajuste, ou seja, determinar os parâmetros da hipótese que melhor aproximem $h_\theta(x)$ de **y**, utilizando os vetores de treinamento **x** e **y**. Na figura abaixo é ilustrado esse conceito.



Na figura acima, temos os pares de treinamento representados pelos pontos vermelhos e a reta traçada a partir da equação da reta da hipótese. No exemplo acima, $x^{(1)} = 1$ e $y^{(1)} = 6$, enquanto que $h_\theta(x^{(1)}) = 5$. A diferença entre o valor estimado pela hipótese, e o valor usado no treinamento, é representado pelo traço verde. A melhor hipótese é aquela cujas as diferenças são as menores possíveis, ou seja $h_\theta(x) - y$ seja a menor possível para todas as amostras do conjunto de treinamento.

Conforme pode ser visto na figura anterior, as diferenças podem ser positivas, caso $h_\theta(x) > y$ ou negativas, caso $h_\theta(x) < y$. Contudo, ambas diferenças devem ser minimizadas.
Assim, ao invéz de minimizar a diferença entre esses valores, serão minimizadas as diferenças quadráticas, ou seja, $(h_\theta(x) - y)^2$. A partir desse princípio, podemos criar a função:

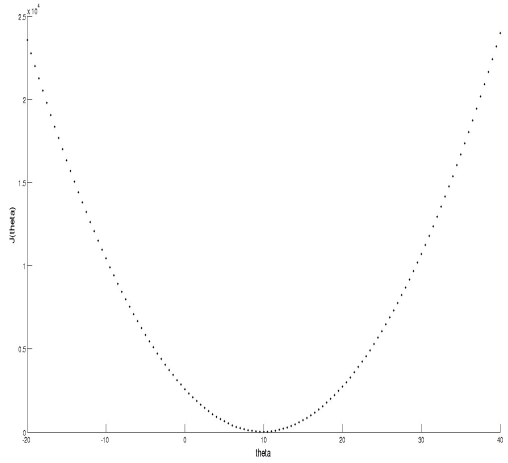
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Nessa função, temos a soma de todas as diferenças entre os valores de $h_\theta(\mathbf{x})$ e **y**. Ela é chamada de **função custo** ou **função objetivo**. Portanto, a regressão linear consiste em minimizar a função J.

Para simplificar o entendimento, consideraremos a hipótese como uma função de um único parâmetro, fazendo $\theta_0 = 0$, isto é, $h_\theta(x)$ é uma equação da reta que passa pela origem. Assim, temos:

$$h_\theta(x) = \theta_1 x \qquad \text{e portanto} \qquad J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

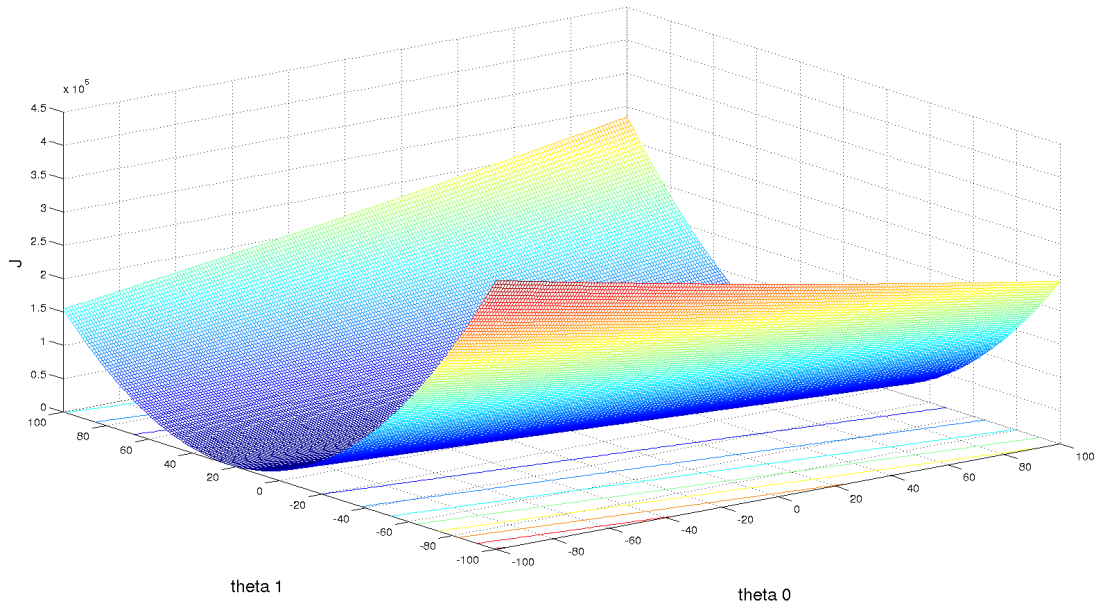
Para ilustrar o problema, podemos traçar o gráfico de $J(\theta_1)$ para diferentes valores de θ_1.



Percebe-se que o valor de θ_1 que resulta no menor valor da função custo é $\theta_1 \approx 10$. Entretanto, calcular a função custo exige um certo processamento computacional, e determinar o valor dos parâmetros dessa forma, se torna muitas vezes inviável, especialmente se tivermos muitos parâmetros.

Por esse motivo, o algoritmo para cálculo dos parâmetros do modelo (algoritmo de aprendizagem) começará com um valor inicial de θ_1, e tentará numericamente minimizar o valor de J.

Voltando agora para a hipótese inicial com dois parâmetros, a função custo será uma função de duas variáveis. Portanto, o gráfico dessa função será tridimensional.



Para funções objetivo com mais de 2 parâmetros, não é possível visualizar o gráfico da função, mas o significado matemático dela continua inalterado.

Algoritmo do Gradiente Descendente

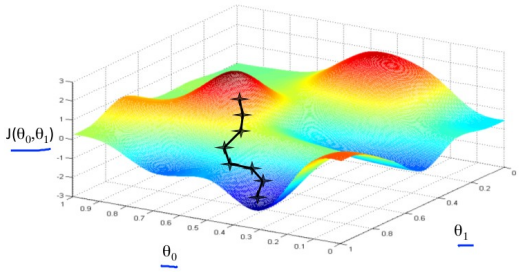
Uma vez calculada a função custo de um modelo, busca-se minimizar tal função, ajustando (calibrando) os parâmetros da hipótese (modelo) para o melhor ajuste aos dados de treinamento. Um dos algoritmos usados para esse processo, é chamado de **gradiente descendente**, ou **gradiente negativo**.

Do cálculo diferencial, temos que a derivada de uma função fornece a taxa de variação da função em um determinado ponto. Ao aplicarmos esse conceito em funções com mais de uma variável, temos o conceito de *gradiente*.

Se calcularmos o gradiente da função custo, o "sentido" negativo dessa função, apontará para um menor valor da função, que é o objetivo da regressão linear.

Assim, podemos resumir o algoritmo como:

- Calcular o valor da função custo para um conjunto de parâmetros iniciais;
- Atualizar (simultaneamente) os valores dos parâmetros visando minimizar a função;
- Repetir o passo acima até atingir o mínimo da função.



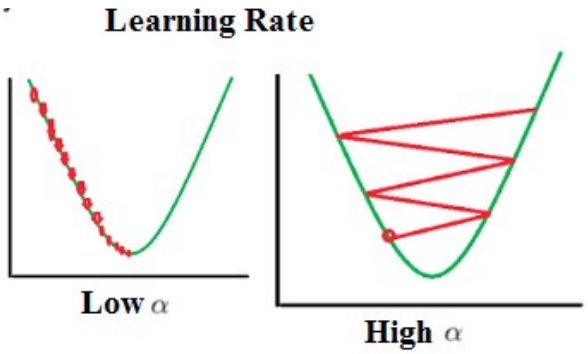
O cálculo realizado para a atualização dos parâmetros é dado por:

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

onde *j = 0 e 1*, α é a taxa de aprendizagem, e a derivada parcial da função define o gradiente da função para a variável θ_j. O fato de atualizarmos o parâmetro subtraindo seu valor do gradiente, determina a direção de minimização da função.

Um detalhe importante da implementação desse cálculo, é a atualização simultânea dos parâmetros. Isso significa que o valor de J usado para o cálculo da derivada, deve ser o mesmo para os dois parâmetros. Por isso, é aconselhável armazenar os valores atualizados em variáveis temporárias até todas as derivadas serem calculadas, depois disso se faz a atualização dos parâmetros.

A **taxa de aprendizagem**, que multiplica a derivada, determina o "tamanho" do passo de atualização para cada iteração do algoritmo. Como podemos ver na figura de exemplo acima, a direção que o algoritmo percorre do valor inicial ao final, nem sempre é a mesma. Um valor muito pequeno da taxa de aprendizagem fará com que o algoritmo faça passos pequenos, exigindo muitas iterações até a convergência. Passos muito grande de α fará com que o algoritmo ultrapasse a região de menor gradiente, o chamado efeito costura, fazendo com que o algoritmo não ache a convergência.



Em geral, usar uma taxa de aprendizado fixa, fará com que o tamanho do passo diminua a medida que a função custo caminha para uma região de mínimo.

Ao aplicarmos o algoritmo descrito para a regressão linear, devemos considerar as equações principais:

$$h_\theta(x)=\theta_0+\theta_1 x \qquad J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})^2$$

A partir dessas equações, devemos fazer a atualização iterativa dos parâmetros na forma:

$$\theta_j=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$$

Devemos, portanto, calcular a derivada parcial da função objetivo com relação a cada um dos parâmetros. Unindo os conhecimentos de cálculo diferencial e álgebra linear, temos:

$$\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)}) \qquad \frac{\partial}{\partial\theta_1}J(\theta_0,\theta_1)=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x^{(i)}$$

Dessa forma, temos as equações completas necessárias para a atualização de cada um dos parâmetros.

$$\theta_0=\theta_0-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})$$

$$\theta_1=\theta_1-\alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)})-y^{(i)})x^{(i)}$$

Regressão Linear de Múltiplas Variáveis

Suponha agora que mais de um fator pode influenciar as notas dos alunos na disciplina de Aprendizagem de Máquina. Por exemplo, além das horas de estudo, podemos considerar, a nota obtida pelo aluno em Álgebra Linear, o número de disciplinas cursando concomitantemente, etc.

Nesse caso, teríamos mais de uma variável (dados de entrada) que resultaria em diferentes notas (dados de saída). A hipótese (modelo), portanto, deverá contemplar todas essas variáveis. Assim:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Assim, a tabela de dados teria uma coluna para cada variável independente, além da coluna correspondente para a variável dependente. As linhas dessa tabela, portanto, não representam apenas um par, mas um conjunto de dados de entrada com um respectivo valor de saída, formando um exemplo de treinamento.

A notação adicional para representar essa situação é:

- n é o número de variáveis
- $x^{(i)}$ é o *i-ésimo* exemplo de treinamento
- $x_j^{(i)}$ é o valor da variável *j* do *i-ésimo* exemplo de treinamento

A partir da equação anterior, podemos perceber que o número de parâmetros excede em uma unidade o número de variáveis. Por conveniência, com o prósito de igualar as dimensões e possibilitar o uso da álgebra linear, podemos acrescentar a variável $x_0 = 1$ na equação:

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

Consequentemente, podemos representar a equação acima na forma vetorial:

$$h_\theta(x) = \theta^T x$$

Onde **x** é o vetor de variáveis independentes e **θ** é o vetor formado pelos parâmetros, ambos com dimensão *n+1*. Na multiplicação acima, usamos a transposta do vetor **θ**. Repare que ao usarmos os vetores, não há o subscripto de cada um deles.

A definição da função custo para múltiplas variáveis não sofre alteração. O único cuidado a ser tomado é considerar que, não havendo o subscrito, estamos considerando todo o vetor.

$$J(\theta)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)^2$$

No algoritmo do gradiente descendente, as equações também permanecem as mesmas, assim como a atualização dos parâmetros durante o processo iterativo.

$$\theta_j=\theta_j-\alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

$$\theta_j=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)})-y^{(i)}\right)x_j^{(i)}$$

atualizado simultaneamente para j = 0, 1, 2, … , n.

Escalonamento de variáveis

Um detalhe importante na regressão linear de múltiplas variáveis, é a variação de valores das variáveis. Por se tratarem de informações diferentes, elas podem ter uma faixa de valores também bastante diferentes.

Por exemplo, considere a modelagem de um sistema de precificação de apartamentos. A variável dependente (valor alvo) é o valor do imóvel. As variáveis de entrada podem ser área do imóvel e número de quartos. Nesse caso, os possíveis valores de área podem ir de 30 a 500 (m^2), enquanto que o número de quartos provavelmente iria de 1 a 4 (talvez 5).

Se durante a regressão, forem usados os valores brutos, o processo será tendencioso, procurando diminuir prioritariamente o valor da função custo com relação àquela variável com maior variação (a área no exemplo acima).

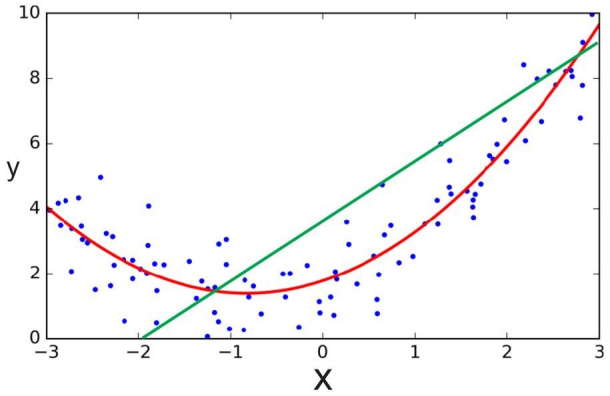
Por isso, é importante fazer com que todas as variáveis tenham uma faixa comparável de possíveis valores. Uma das formas mais simples de fazer isso, é a normalização pela média. Nesse caso, cada amostra dos dados de entrada serão re-calculados da forma:

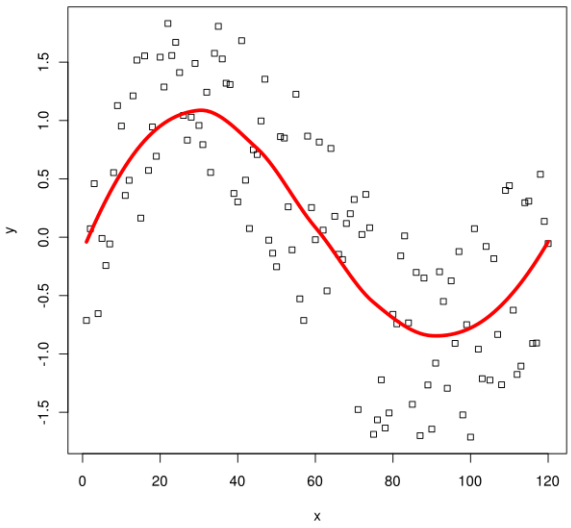
$$x_i = \frac{x_i - \mu_i}{S_i}$$

onde μ_i é a média do conjunto de amostras para a variável i, e S_i é a faixa de valores da mesma variável, ou seja, seu valor máximo subtraído do seu valor mínimo.

Regressão polinomial

Para determinados dados, a modelagem usando uma equação da reta acarreta em uma aproximação não muito fiel. Nesse caso, hipóteses mais complexas podem ser usadas.





Essas hipóteses podem ser formuladas a partir de equações de grau maior que um:

$$h_\theta(x) = \theta_0 + \theta_1 x^2 + \theta_2 x^3 + \cdots \qquad \text{ou} \qquad h_\theta(x) = \theta_0 + \theta_1 x^2 + \theta_2 \sqrt{x} + \cdots$$

O ajuste da função reduzirá o erro (função custo), sendo portanto mais fiel ao conjunto de dados. Um compromisso deve ser feito, entretanto, entre a complexidade da função e a validade da predição. Funções extremamente complexas terão menor erro, mas podem não representar com fidelidade uma situação real.

Equação Normal

Nos exemplos acima, o vetor de parâmetros θ foi estimado através de um método numérico iterativo, o gradiente descendente. No entanto, a partir dos conhecimentos em álgebra linear e cálculo diferencia, é possível determinar o vetor analiticamente.

Sabe-se que o mínimo de uma função é um ponto cuja a derivada de primeira ordem é igual a zero, e a derivada de segunda ordem é positiva. Dessa forma, podemos calcular analiticamente a derivada da função custo e igualá-la a zero.

Para um problema de ***n*** variáveis e ***m*** conjuntos de treinamento, temos:

entradas					saída
$\boldsymbol{x_0}$	$\boldsymbol{x_1}$	$\boldsymbol{x_2}$	$\cdots$	$\boldsymbol{x_n}$	$\boldsymbol{y}$
$x_0^{(1)}$	$x_1^{(1)}$	$x_2^{(1)}$	$\cdots$	$x_n^{(1)}$	$y^{(1)}$
$x_0^{(2)}$	$x_1^{(2)}$	$x_2^{(2)}$	$\cdots$	$x_n^{(2)}$	$y^{(2)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_0^{(m)}$	$x_1^{(m)}$	$x_2^{(m)}$	$\cdots$	$x_n^{(m)}$	$y^{(m)}$

Os dados de entrada podem ser escritos na forma vetorial:

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \qquad \text{portanto} \qquad X = \begin{bmatrix} \left(x^{(1)}\right)^T \\ \left(x^{(2)}\right)^T \\ \vdots \\ \left(x^{(m)}\right)^T \end{bmatrix}$$

Onde a matriz X é chamada de **matriz de projeto** (*design matrix*). A partir da matriz ***X*** e do vetor ***y***, podemos calcular a derivada da função custo ***J*** e achar o vetor θ que a iguale a zero.

$$\theta = \left(X^T X\right)^{-1} X^T y$$

A equação acima fornece, automaticamente, nos valores de θ que resultam numa função custo igual a zero e, portanto, na solução do problema. O maior problema em tal abordagem é o cálculo de $(X^TX)^{-1}$, pois envolve a inversão de uma matriz. Conforme dito anteriormente, nem todas matrizes são inversíveis (p.e. número de variáveis maior que número de conjunto de treinamento, $n > m$), e esse cálculo pode ser inviável.

Esse problema pode ser resolvido usando SVD (*single value decomposition*), que calcula a matriz pseudo-inversa, que se aproxima do valor de uma matriz inversa real, e pode ser aplicado a matrizes não-inversíveis. (ver função ***pinv*** do MatLab/Octave).

Adicionalmente, o esforço computacional para o cálculo de uma matriz inversa pode ser excessivo, caso o número de variáveis for muito alto (p.e. 800, 10.000 ou mais). Nesses casos, o método iterativo do gradiente negativo torna-se uma opção mais viável.

Resumindo, as principais características dos dois métodos são:

Gradiente negativo	**Equação normal**
Requer uma taxa de aprendizado α	Não requer taxa de aprendizado α
Requer várias iterações	Não requer iterações
	Requer o cálculo de uma matriz inversa
Funciona bem para muitas variáveis (***n*** alto)	Muito lento para muitas variáveis (***n*** alto)