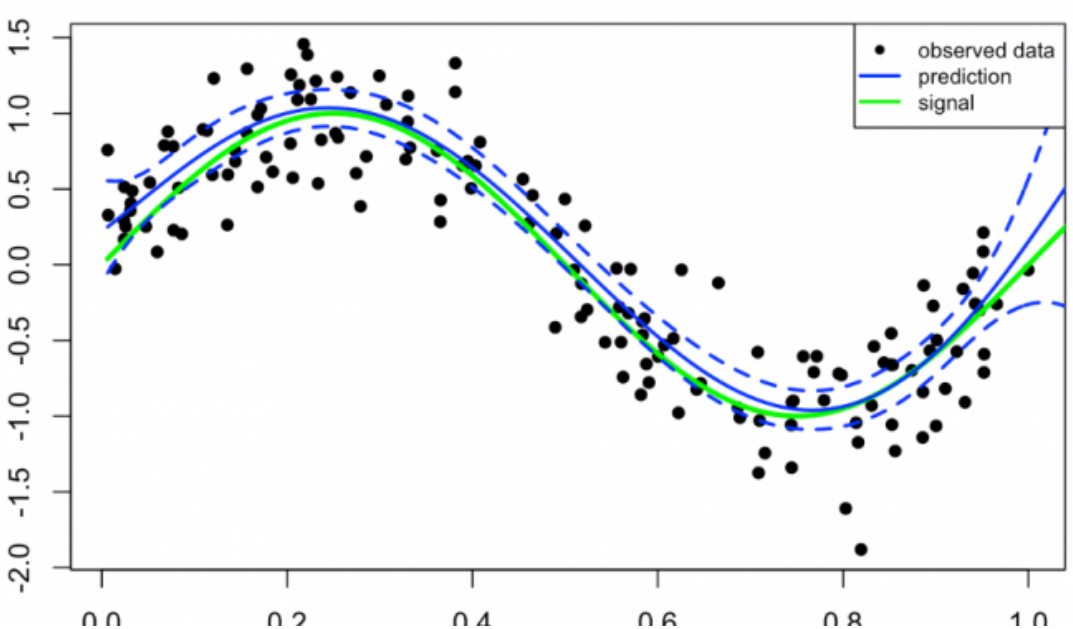
Mínimos Quadrados - 2025 - Parte 2

© Gustavo C. Buscaglia

gustavo.buscaglia@gmail.com

Parte 2: Variantes e generalizações



1 Revisão

Problema (aproximação baseada em dados): Sejam $(\mathbf{x}_n, y_n)$, $n = 1, \ldots, N$, observações independentes de variáveis $\mathbf{x}$ e y. Sabe-se que y é função de $\mathbf{x}$, mas contém um erro de medição que faz com que os pares satisfaçam

$$y_n = f(\mathbf{x}_n) + \epsilon_n,$$

onde ϵ_n é um ruído (para cada n).

Deseja-se obter, a partir de $\mathbf{X}$ e $\mathbf{y}$, a função f. Para isto, escolhe-se um conjunto paramétrico de funções V **no qual será procurada** f. **Essa é uma escolha de alto nível (hiper-parâmetro)**.

Assim, cada função $f_{\boldsymbol{\theta}}$ de V (ou **modelo**) depende de um vetor de M parâmetros $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_{M-1})$, isto é,

$$f(\mathbf{x}, \boldsymbol{\theta}) \qquad f : \text{ variáveis } \times \text{ parâmetros } \rightarrow \text{ respostas}$$

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\theta}) \ .$$

Exemplo: $f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0\phi_0(\mathbf{x}) + \ldots + \theta_{M-1}\phi_{M-1}(\mathbf{x})$, que foi o caso analisado, correspondente a problemas de regressão linear.

A pergunta é: Como determinar parâmetros $\boldsymbol{\theta}^*$ tal que a função $f^*(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\theta}^*)$ forneça uma boa estimação de $\mathbf{y}$, para todo $\mathbf{x}$? (não só para os $\mathbf{x}$s já medidos, desejamos que generalize bem).

Resposta ao problema: Mínimos quadrados "ordinários"

a) Defina seu conjunto de funções linearmente nos seus parâmetros, através de funções base (ou atributos)

$$f(\mathbf{x}, \boldsymbol{\theta}) = \theta_0 \phi_0(\mathbf{x}) + ... + \theta_{M-1} \phi_{M-1}(\mathbf{x})$$

b) Defina a função de perda de mínimos quadrados

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{n=0}^{n<N} |f(\mathbf{x}_n, \boldsymbol{\theta}) - y_n|^2 \ ,$$

c) Calcule $\boldsymbol{\theta}^*$ minimizando $\mathcal{L}$ sobre o espaço de parâmetros possíveis.

$$\boldsymbol{\theta}^* = \boldsymbol{\theta}_{\mathrm{MQ}} = \ \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y})$$

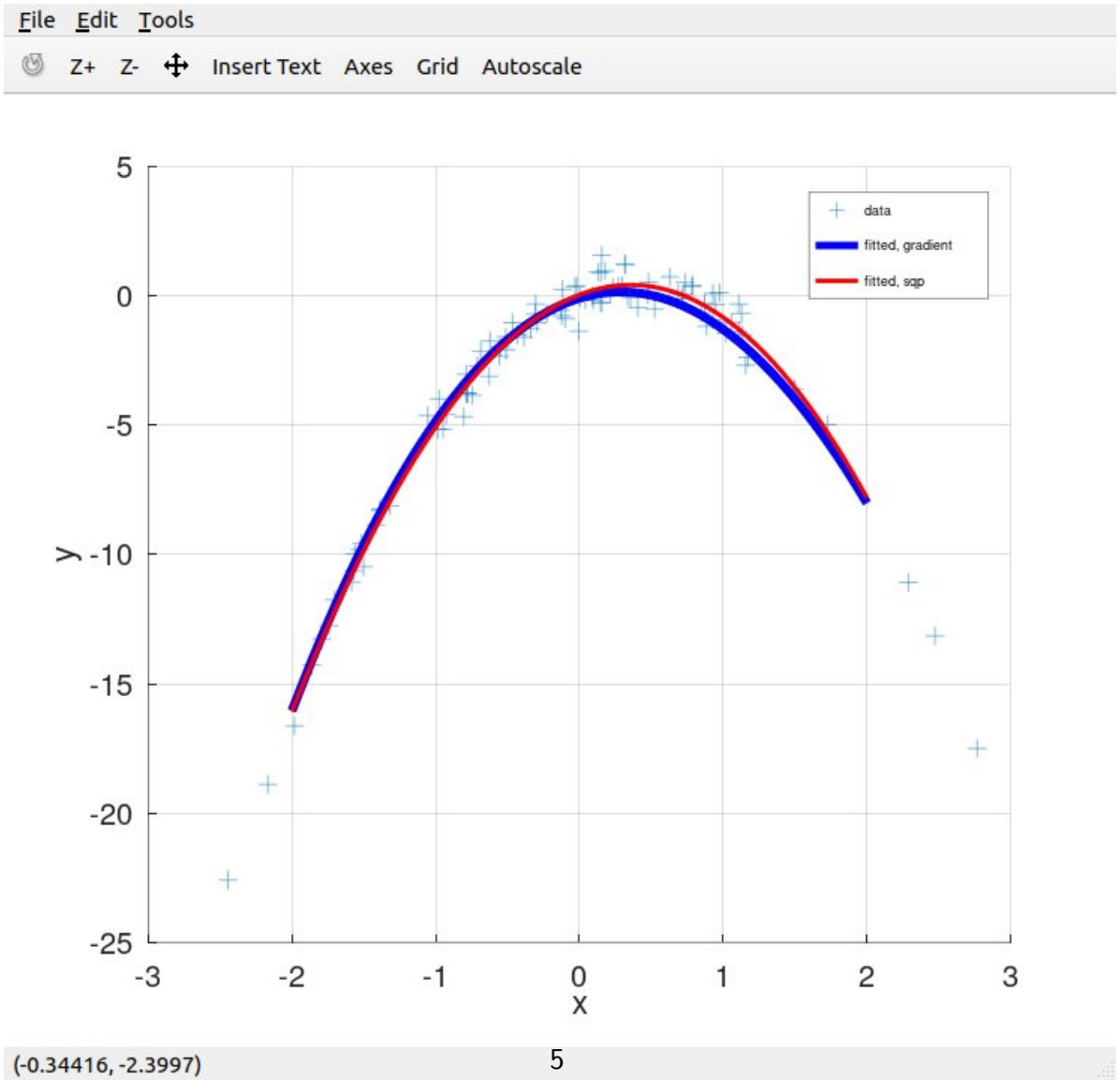
Quando $\mathbf{X}$ é de posto completo, det $\left(\mathbf{X}^T \mathbf{X}\right)$ é distinto de zero, e

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

2 Implementação flexível

Ver notebook MinimosQuadradosEVariantes no E-Disciplinas.

```
import numpy as np
from scipy.optimize import minimize

def fmodel(XX,theta):        ##MODELO: caso linear
    m = len(theta)
    res = theta[0] * phi(0, XX)
    for k in range(1, m):
        res = res + theta[k] * phi(k, XX)
    return res

def phi(k, x):
    return x**k

def perda(theta,XX,yy):      ##FUNCAO DE PERDA: minimos quadrados
    nx = len(XX)
    fx = fmodel(XX,theta)
    ll = np.dot(fx - yy, fx - yy) / nx
    return ll

result = minimize(           ##ALGORITMO DE OTIMIZACAO
    lambda theta: perda(theta,XX,yy),
    theta_initial, method='BFGS', callback=print_iteration)

theta_optimized = result.x
loss_final = result.fun
```


File Edit Tools
Z+ Z- Insert Text Axes Grid Autoscale
data
fitted, gradient
fitted, sqp
5
0
-5
-10
-15
-20
-25
y
-3
-2
-1
0
1
2
3
X
(-0.34416, -2.3997)

3 Generalizações

Até aqui, nossa aproximação dos dados $(\mathbf{X}, \mathbf{y})$ pode ser resumida e generalizada nos passos seguintes:

- **Passo 1:** Definir uma função *modelo*

$$f(\mathbf{x}, \boldsymbol{\theta}) \qquad f: \text{ parâmetros } \times \text{ variáveis } \rightarrow \text{ respostas}$$

  - No caso visto (MQ ordinários) foi $f(\mathbf{x}, \boldsymbol{\theta}) = \theta_1\phi_1(\mathbf{x}) + ... + \theta_M\phi_M(\mathbf{x})$, em particular $f(x, \theta) = \theta_1 + \theta_2 x + \theta_3 x^2 + ...]$

  - Pode ser generalizado primeiramente definindo features não lineares arbitrários.

  - As aproximações com redes neurais não são senão famílias $f(\mathbf{x}, \boldsymbol{\theta})$ não lineares em $\boldsymbol{\theta}$ (inspiradas em redes de neurônios). São chamadas de "interpolantes não lineares" ou "interpolantes flexíveis").

- **Passo 2:** Definir uma função *de perda*

  - No caso visto (MQ ordinários) foi

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} |f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i|^2 \ .$$

  - Generalização frequente:

$$\mathcal{L}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}) = \sum_{i=1}^{n} |f(\mathbf{x}_i, \boldsymbol{\theta}) - y_i|^{\alpha} + \epsilon \|\boldsymbol{\theta}\|_{\beta}^{\beta},$$

    onde

$$\|\boldsymbol{\theta}\|_{\beta} = \left(\sum_{k=1}^{M} |\theta_k|^{\beta} \right)^{\frac{1}{\beta}}$$

  - O termo adicionado é um termo de **regularização**, sendo que quando $\beta = 2$ é chamada de RIDGE e quando $\beta = 1$ é chamada de LASSO.

  - A regularização RIDGE tem um sistema linear que permite calcular o mínimo de maneira direta:

$$\boldsymbol{\theta}^* = \left(\frac{1}{N} \mathbf{X}^T \mathbf{X} + \epsilon \, \mathbf{I} \right)^{-1} \left(\frac{1}{N} \mathbf{X}^T \, \mathbf{y} \right)$$

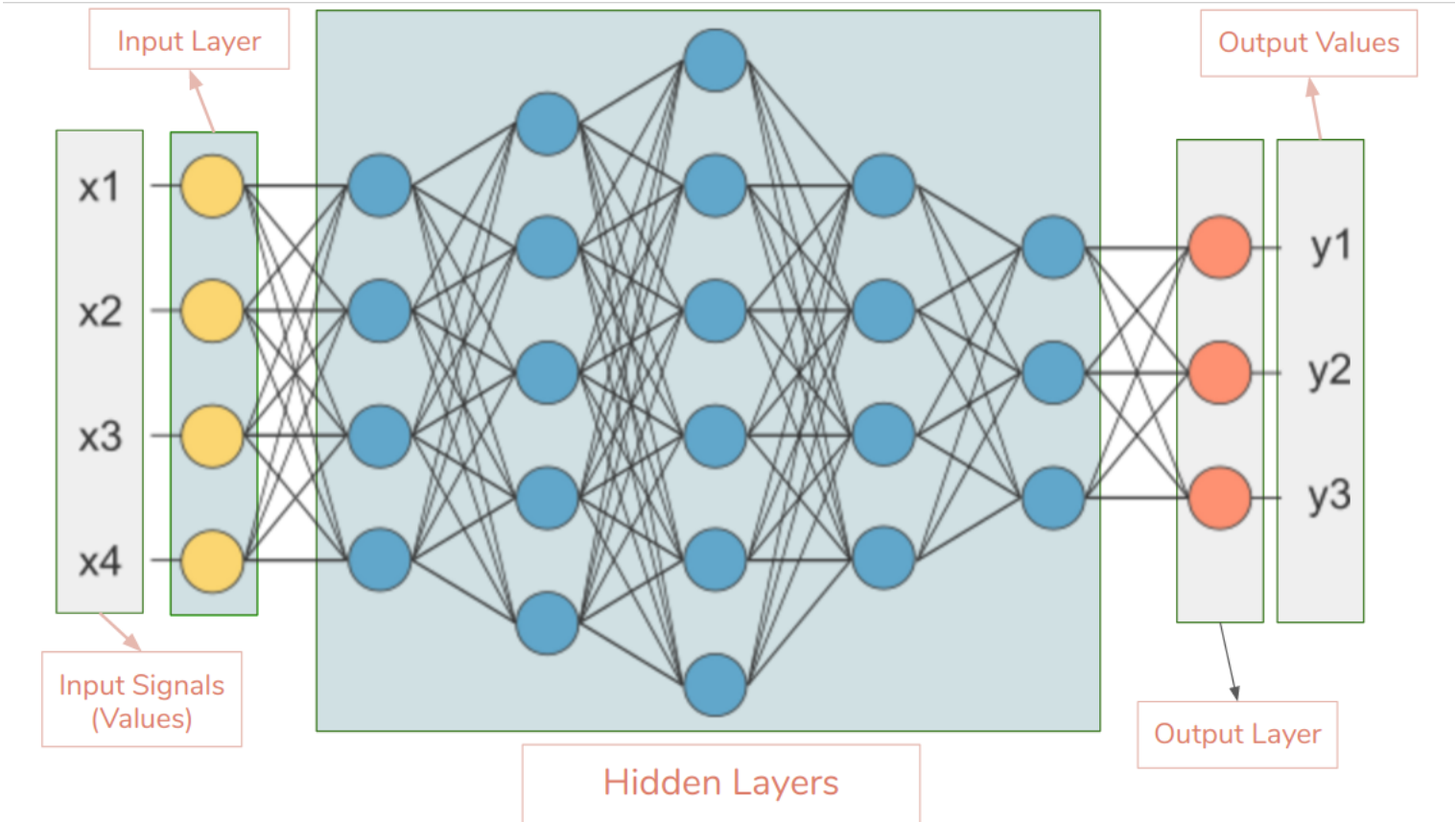
- A regularização LASSO tende a fornecer aproximações mais **esparsas** (com menos variáveis envolvidas, e assim mais interpretáveis).

- Se desejamos penalizar os erros **relativos** equilibradamente, podemos tomar o logaritmo de $|f(x,\theta) - y|$ em vez de uma potência dele.

- **Passo 3:** Definir a *aproximante* $f^*(x)$, ou *interpolante generalizada*, como o mínimo de $\mathcal{L}$ no espaço de parâmetros; i.e.,

$$f^*(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\theta}^*), \quad \mathcal{L}(\boldsymbol{\theta}^*, \mathbf{X}, \mathbf{y}) \leq \mathcal{L}(\boldsymbol{\theta}, \mathbf{X}, \mathbf{y}), \ \forall \boldsymbol{\theta}$$

4 Redes neurais artificiais



Exemplo concreto: Rede com uma entrada, uma saida, e uma hidden layer de $N_{\mathbf{neu}} = L$ neurônios

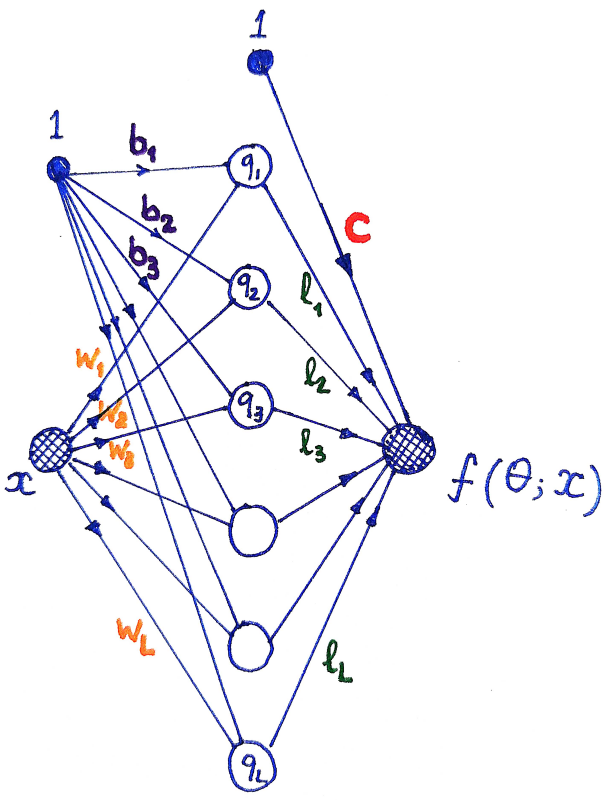
Seja

$$\theta = (b_1, b_2, \ldots, b_L, w_1, w_2, \ldots, w_L, \ell_1, \ell_2, \ldots, \ell_L, c)^T,$$

onde b_k é o bias de cada neurônio, w_k seu peso de entrada, ℓ_k o peso de saida e c o bias da saida. O número total de parâmetros es $3N_{\text{neu}} + 1$. A resposta surge da seguinte fórmula (ver esquema)

$$\begin{aligned} q_k &= \text{ReLU}\,(b_k + w_k\, x) \qquad k = 1, \ldots, L \\ f(\theta; x) &= c + \sum_{k=1}^{L} \ell_k\, q_k \end{aligned}$$

onde $\text{ReLU}(t) = \max(0, t)$.



Programação em Octave

$$\begin{aligned} q_k &= \text{ReLU}\,(b_k + w_k\, x) \qquad k = 1, \ldots, L \\ f(\theta; x) &= c + \sum_{k=1}^{L} \ell_k\, q_k \end{aligned}$$

onde $\text{ReLU}(t) = \max(0, t)$.

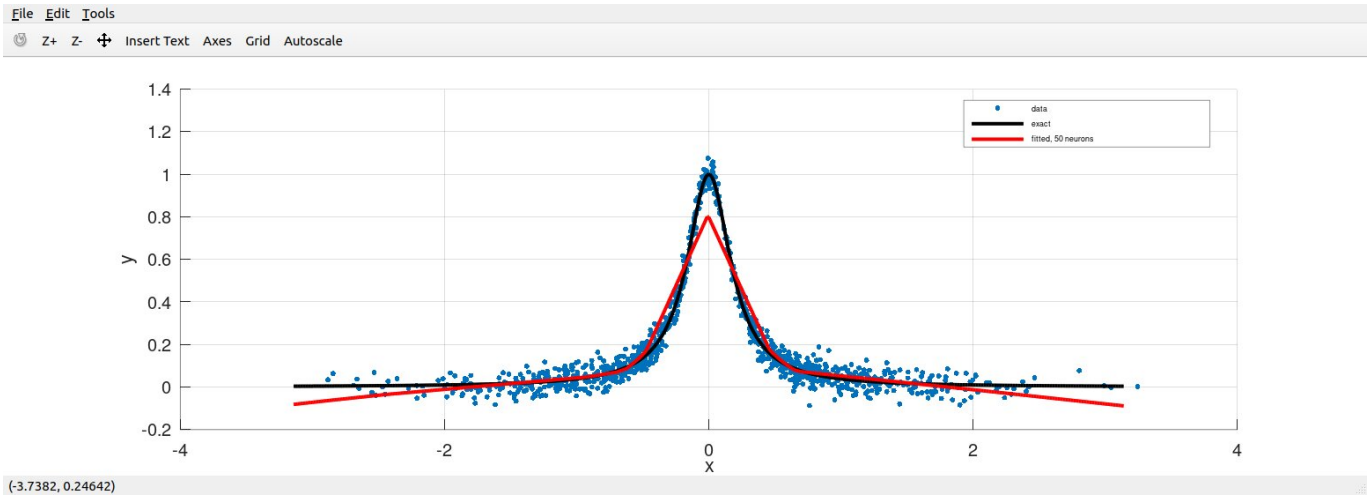
```
function r=relu(s)
 r=max(s,0);
end

function fx=fmodelrn(theta,x)
 nx=length(x); nneu=(length(theta)-1)/3;
 b(1:nneu,1)=theta(1:nneu);
 w(1:nneu,1)=theta(nneu+1:2*nneu);
 ell(1:nneu,1)=theta(2*nneu+1:3*nneu);
 c=theta(3*nneu+1);
 q=relu(b*ones(1,nx)+w*x);
 fx=c+ell'*q;
end
```

```
function ll=perda(theta)
 global xx
 global yy
 x=xx; y=yy; nx=length(x); eps=1e-3;
 fx=fmodelrn(theta,x);
 residuo=y-fx;
 ll=1./nx*norm(residuo,2)^2+eps*norm(theta,2)^2;
end
```

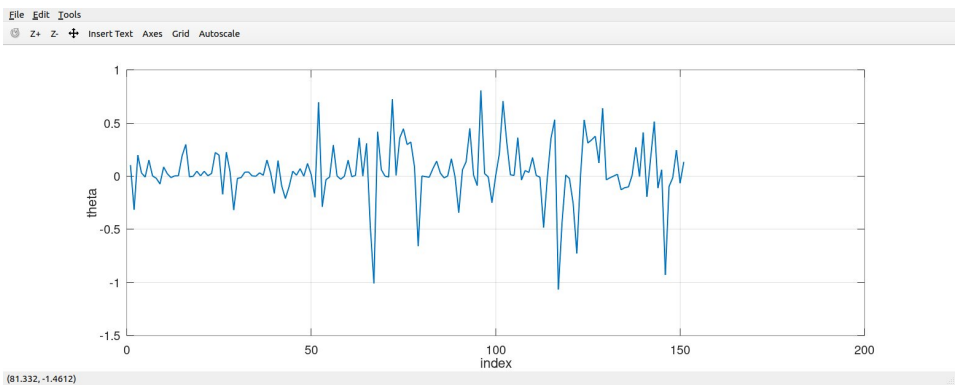
Com isto programado, só resta carregar os vetores `xx` e `yy`, escolher o número de neurônios e executar

```
theta_opt=sqp(randn(3*nneu+1,1),@perda);
```

Resultado



Aproximação com 50 neurônios



Valores dos parâmetros θ. Notar a dificuldade de interpretar eles.