

Extreme Learning Machine in J

Pierre-Edouard Portier

2019

1 Regression

$\mathbf{x}^{(1)} \dots \mathbf{x}^{(P)}$ are vectors of \mathbb{R}^{n-1} with associated values $y^{(1)} \dots y^{(P)}$ of \mathbb{R} . We search a function $f(\mathbf{x}) : \mathbb{R}^{n-1} \rightarrow \mathbb{R}$ to model the observed relationship between \mathbf{x} and y . f can have a fixed parameterized form. For example:

$$f(\mathbf{x}) = a_0 + a_1x_1 + a_2x_2 + \dots + a_{n-1}x_{n-1}$$

If $P = n$, parameters $a_0 \dots a_{n-1}$ are found by solving a linear system.

$$\begin{cases} y^{(1)} &= a_0 + a_1x_1^{(1)} + a_2x_2^{(1)} + \dots + a_{n-1}x_{n-1}^{(1)} \\ \dots &= \dots \\ y^{(P)} &= a_0 + a_1x_1^{(P)} + a_2x_2^{(P)} + \dots + a_{n-1}x_{n-1}^{(P)} \end{cases}$$

This system can be written in matrix form.

$$\begin{pmatrix} 1 & x_1^{(1)} & \dots & x_{n-1}^{(1)} \\ 1 & x_1^{(2)} & \dots & x_{n-1}^{(2)} \\ \dots & \dots & \dots & \dots \\ 1 & x_1^{(P)} & \dots & x_{n-1}^{(P)} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(P)} \end{pmatrix}$$

Each line of the first term matrix is a vector $\mathbf{x}^{(i)T}$ with the addition of a constant coordinate that accounts for parameter a_0 . Thus, naming this matrix \mathbf{X}^T , the linear system can also be written:

$$\mathbf{X}^T \mathbf{a} = \mathbf{y}$$

Consider the special case when x is a number and f is a polynomial of degree $n - 1$:

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

With $P = n$ examples $(x^{(k)}, y^{(k)})$, the parameters are found by solving the following linear system:

$$\begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^{n-1} \\ 1 & x^{(2)} & (x^{(2)})^2 & \dots & (x^{(2)})^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x^{(P)} & (x^{(P)})^2 & \dots & (x^{(P)})^{n-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \dots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(P)} \end{pmatrix} \quad (1)$$

Incidentally, the first term is called the Vandermonde Matrix.

1.1 Experiment with a 1-dimensional synthetic dataset

We define a non linear function f from which we generate a dataset.

2a $\langle dataset\ 2a \rangle \equiv$ (11b)

```
f=: 3 : ' (^y) * cos 2*pi * sin pi * y'
⟨noise 2b⟩
⟨gendat 2d⟩
```

In traditional mathematical form, this function is:

$$f(x) = e^x \times \cos(2\pi \sin(\pi x))$$

Function `noise` adds some random noise to the values of a vector. For example `0.5 noise v`, will add random values uniformly drawn from interval $[-0.5, 0.5]$ to the terms of vector v .

2b $\langle noise\ 2b \rangle \equiv$ (2a)

```
noise=: 4 : 'y + -&x *&(+:x) ? (#y) # 0'
```

`0.5 gendat 10` generates from f a dataset (X, Y) of 10 points with random noise in $[-0.5, 0.5]$ added to Y . It also stores in `minmaxX` the minimum and maximum values of X . It computes the pair `minmaxf`, where the first term is ten percent smaller than the minimum of f on interval $[0, 1]$, and the second term is ten percent bigger than the maximum of f on interval $[0, 1]$. `minmaxf` is later used to crop the plots so that extreme values are not visible.

A test set (XT, YT) is used to assert the capacity of the model to generalize on unseen data. Its size is fixed to 10% of the size of the training set.

2c $\langle utils\ 2c \rangle \equiv$ (11b) 4a▷

```
pushup=: ] + 0.1 * |
pushdown=: ] - 0.1 * |
```

2d $\langle gendat\ 2d \rangle \equiv$ (2a)

```
gendat=: 4 : 0
  X=: ? y $ 0
  Y=: x noise f X
  minmaxX=: (<./ , >./) X
  minmaxf=: (([: pushdown <./) , ([: pushup >./)) f steps 0 1 100
  XT=: ? (>. 0.1 * y) $ 0
  YT=: f XT
  0
)
```

plotdat 0 plots the dataset.

3a $\langle plotdat\ 3a \rangle \equiv$ (11b)

```

plotdatnoshow=: 3 : 0
   $\langle initplot\ 3b \rangle$ 
  pd X;Y
   $\langle plotf\ 3c \rangle$ 
)
plotdat=: 3 : 0
  plotdatnoshow 0
  pd 'show'
)
```

3b $\langle initplot\ 3b \rangle \equiv$ (3a 4e)

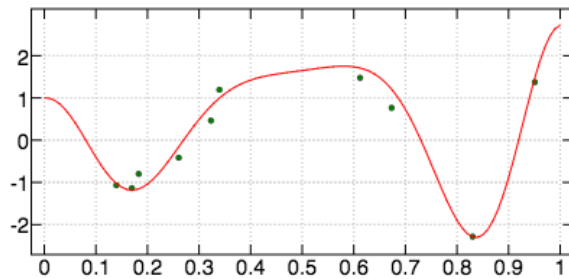
```

pd 'reset'
pd 'color green'
pd 'type marker'
pd 'markersize 1'
pd 'markers circle'
```

3c $\langle plotf\ 3c \rangle \equiv$ (3a 4e)

```

pd 'color red'
pd 'type line'
pd 'pensize 1'
pd (;f) steps 0 1 100
```



0.5 gendat 10
plotdat 0

polyreg 0 solves the linear system (1), stores the coefficients of the polynomial in variable c and computes YThat, the predictions on the test dataset.

3d $\langle polyreg\ 3d \rangle \equiv$ (11b)

```

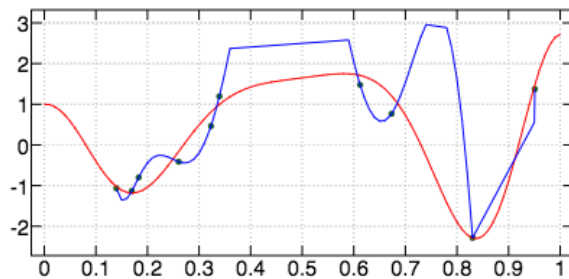
polyreg=: 3 : 0
  c=: Y ([ %. ] ^/ i.@#0]) X
  YThat=: c&p. XT
  plotpoly 0
)
```

4a `<utils 2c>+≡` (11b) `<2c 4c>`

NB. locate the elements with values between `{.x` and `{:x`
`sel=: (] >: {.@[] *. (] <: {:@[]`

4b `<plotpoly 4b>≡` (11b)

```
plotpoly=: 3 : 0
  plotdatnoshow 0
  pd 'color blue'
  xs=: (] #~ minmaxX"_ sel ]) /:~ X,steps 0 1 100
  pval=: c&p. xs
  crop=: minmaxf sel pval
  pd (crop # xs);(crop # pval)
  pd 'show'
)
```



polyreg 0

`test 0` returns the root mean square error (RMSE) on the test set, and a plot of the predictions.

4c `<utils 2c>+≡` (11b) `<4a 6b>`

```
mean=: +/ % #
rmse=: [: %: [: mean ([: *: -)
```

4d `<test 4d>≡` (11b)

```
test=: 3 : 0
  plottest 0
  YT rmse YThat
)
```

4e `<plottest 4e>≡` (11b)

```
plottest=: 3 : 0
  <initplot 3b>
  pd XT;YT
  pd 'color magenta'
  pd XT;YThat
  <plotf 3c>
  pd 'show'
)
```

1.2 Generalization to a function space

Given a basis for a function space, we can try to express \mathbf{f} as a combination of basis functions.

$$f(\mathbf{x}) = a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \dots + a_n f_n(\mathbf{x})$$

Given a dataset of n pairs $(\mathbf{x}^{(k)}, \mathbf{y}^{(k)})$, the coefficients a_i are found by solving a linear system.

$$\begin{pmatrix} f_1(\mathbf{x}^{(1)}) & f_2(\mathbf{x}^{(1)}) & \dots & f_n(\mathbf{x}^{(1)}) \\ f_1(\mathbf{x}^{(2)}) & f_2(\mathbf{x}^{(2)}) & \dots & f_n(\mathbf{x}^{(2)}) \\ \dots & \dots & \dots & \dots \\ f_1(\mathbf{x}^{(n)}) & f_2(\mathbf{x}^{(n)}) & \dots & f_n(\mathbf{x}^{(n)}) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \dots \\ y^{(n)} \end{pmatrix}$$

Let us denote this linear system by $\mathbf{Ax} = \mathbf{b}$.

1.3 Least squares

The linear system $\mathbf{Ax} = \mathbf{b}$ (with $\mathbf{A} \in \mathbb{R}^{m \times n}$) doesn't necessarily have a solution when there are more examples than the number of basis functions (i.e. $m > n$). Thus, we want to find an approximate solution $\mathbf{Ax} \approx \mathbf{b}$ that minimizes the squares of the errors: $\|\mathbf{Ax} - \mathbf{b}\|_2^2$.

$$\begin{aligned} & \|\mathbf{Ax} - \mathbf{b}\|_2^2 \\ &= \{ \|\mathbf{x}\|_2 = \sqrt{\mathbf{x} \cdot \mathbf{x}} \} \\ & \quad (\mathbf{Ax} - \mathbf{b}) \cdot (\mathbf{Ax} - \mathbf{b}) \\ &= \{\text{euclidean scalar product}\} \\ & \quad (\mathbf{Ax} - \mathbf{b})^T (\mathbf{Ax} - \mathbf{b}) \\ &= \{\text{property of transposition}\} \\ & \quad (\mathbf{x}^T \mathbf{A}^T - \mathbf{b}^T) (\mathbf{Ax} - \mathbf{b}) \\ &= \{\text{multiplication}\} \\ & \quad \mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - \mathbf{x}^T \mathbf{A}^T \mathbf{b} - \mathbf{b}^T \mathbf{Ax} + \mathbf{b}^T \mathbf{b} \\ &= \{\text{Since each element of the sum is a scalar, } \mathbf{b}^T \mathbf{Ax} = (\mathbf{b}^T \mathbf{Ax})^T = \mathbf{x}^T \mathbf{A}^T \mathbf{b}\} \\ & \quad \mathbf{x}^T \mathbf{A}^T \mathbf{Ax} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b} \end{aligned}$$

To this quadratic expression corresponds a convex surface. Its minimum is found by setting its derivative to zero.

$$\begin{aligned} \mathbf{0} &= 2\mathbf{A}^T \mathbf{Ax} - 2\mathbf{A}^T \mathbf{b} \\ &= \\ \mathbf{A}^T \mathbf{Ax} &= \mathbf{A}^T \mathbf{b} \end{aligned}$$

Thus, when $m > n$, we solve $\mathbf{Ax} \approx \mathbf{b}$ by solving $\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$. $\mathbf{A}^T \mathbf{A}$ is called the Gram matrix.

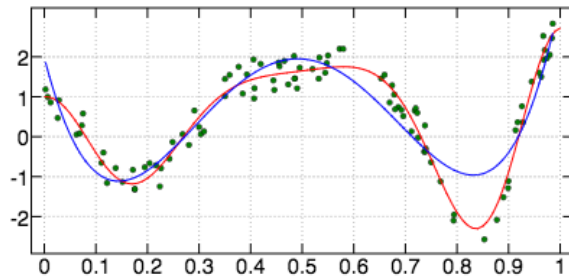
`gram y` computes the Gram matrix **S** for a polynomial basis of degree `y-1`.

6a $\langle \textit{gram } 6a \rangle \equiv$ (11b) 6c
 `gram=: 3 : 0`
 `A=: X ^/ i.y`
 `S=: (mp~ |:) A`
 `)`

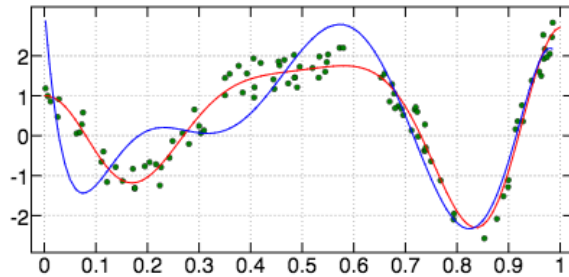
6b $\langle \textit{utils } 2c \rangle + \equiv$ (11b) <4c 8b
 `mp=: +/ . * NB. matrix product`

`leastsq y` solves the overdetermined linear system by computing the Gram matrix for a polynomial basis of degree `y-1`.

6c $\langle \textit{gram } 6a \rangle + \equiv$ (11b) <6a
 `leastsq=: 3 : 0`
 `gram y`
 `c=: ((|:A) mp Y) %. S`
 `YThat=: c&p. XT`
 `plotpoly 0`
 `)`



0.5 gendat 100
leastsq 5



leastsq 8

1.4 Tikhonov regularization

With less examples than the number of basis functions (i.e. $m < n$, underdetermined system), $\mathbf{Ax} = \mathbf{b}$ doesn't have a unique solution. Even with $m \geq n$, the linear system can have approximate solutions more desirable than the optimal one. In particular, this is the case when several examples are very similar. For example, the solution to...

$$\begin{pmatrix} 1 & 1 \\ 1 & 1.00001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.99 \end{pmatrix}$$

...is $\mathbf{x}^T = (1001, -1000)$. However, the approximate solution $\mathbf{x}^T = (0.5, 0.5)$ is more suitable. Indeed, the optimal solution is not likely to adapt well to new inputs (e.g., input $(1, 2)$ would be projected onto $-999...$).

Thus, when several solutions are feasible, we want to favor smaller norms $\|\mathbf{x}\|_2$ by solving a new minimization problem:

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2^2 + \alpha \|\mathbf{x}\|_2^2$$

with $0 < \alpha < 1$

The minimum of this expression is found by setting its derivative to zero.

$$\begin{aligned} \mathbf{0} &= 2\mathbf{A}^T \mathbf{Ax} - 2\mathbf{A}^T \mathbf{b} + 2\alpha \mathbf{x} \\ &= \\ &= (\mathbf{A}^T \mathbf{A} + \alpha \mathbf{I}_{n \times n}) \mathbf{x} = \mathbf{A}^T \mathbf{b} \end{aligned}$$

It comes down to adding a small positive value to the diagonal of the Gram matrix. This approach has been given several names: Tikhonov regularization, ridge regression...

1E_3 ridge 5 will solve the ridge regression for a polynomial basis of degree 5 and a regularization coefficient equal to 10^{-3} .

8a $\langle \text{ridge } 8a \rangle \equiv$ (11b)

```

    ridge=: 4 : 0
      gram y
      c=: ((|:A) mp Y) %. x addDiag S
      YThat=: c&p. XT
      plotpoly 0
    )

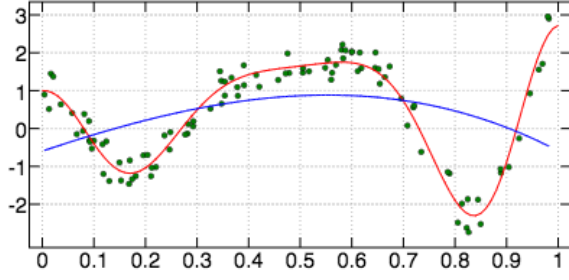
```

8b $\langle \text{utils } 2c \rangle + \equiv$ (11b) <6b

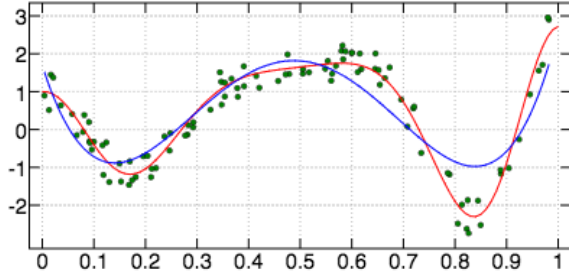
```

    diag=: (<0 1)&|: : (([:(>:*i.)[:#]))
    addDiag=: ([+diag@]) diag ] NB. add x to the diagonal of y

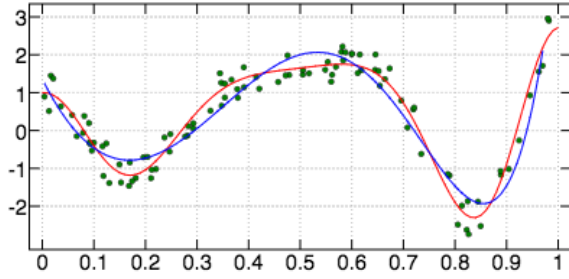
```

1E_4 ridge 4



1E_4 ridge 5



1E_4 ridge 8

1.5 Extreme Learning Machine

The following parameterized form of f corresponds to a single hidden layer neural network.

$$f(\mathbf{x}) = c_1 g(\mathbf{w}_1 \cdot \mathbf{x} + b_1) + c_2 g(\mathbf{w}_2 \cdot \mathbf{x} + b_2) + \dots + c_M g(\mathbf{w}_M \cdot \mathbf{x} + b_M)$$

g is a non-linear activation function. We use the rectified linear unit (ReLU): $g(y) = \max(0, y)$.

If vectors $\mathbf{w}_1 \dots \mathbf{w}_M$ and scalars $b_1 \dots b_M$ are initialized randomly and never modified (i.e., if they are not parameters), we can solve a linear system $\mathbf{H}\mathbf{c} = \mathbf{y}$ of unknown \mathbf{c} .

$$\mathbf{H} : \begin{pmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_M \cdot \mathbf{x}_1 + b_M) \\ \dots & \dots & \dots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_M \cdot \mathbf{x}_N + b_M) \end{pmatrix}$$

$$\mathbf{c}^T : (c_1 \dots c_M)$$

$$\mathbf{y}^T : (y_1 \dots y_N)$$

This approach is named *Extreme Learning Machine*¹.

¹<https://scholar.google.fr/scholar?q=extreme+learning+machine>

`initelm` 100 initializes randomly matrix H with 100 neurons on the hidden layer (i.e., $M = 100$) and computes its Gram form S .

10a $\langle elm\ 10a \rangle \equiv$ (11b) 10b \triangleright

```

initelm=: 3 : 0
  W=: _1 + 2 * ? (y,1) $ 0 NB. input weights
  B=: ? y $ 0 NB. bias
  H=: mkH ,. X
  0 [ S=: (mp~ |: ) H
)
mkH=: 3 : '0&>. B +"1 y mp"1/ W'

```

`elm` 1E.4 solves the extreme learning machine linear system with a Tikhonov regularization coefficient of 10^{-4} .

10b $\langle elm\ 10a \rangle + \equiv$ (11b) \triangleleft 10a

```

elm=: 3 : 0
  c=: ((|:H) mp Y) %.. y addDiag S
  YThat=: (mkH ,. XT) mp c
  plotelm 0
)

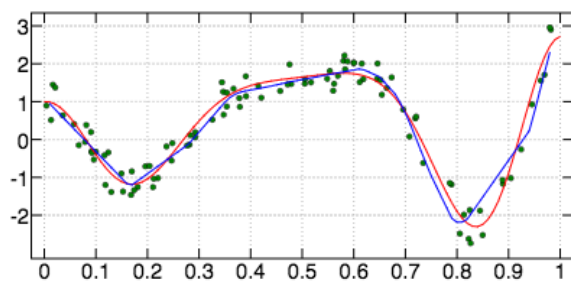
```

10c $\langle plotelm\ 10c \rangle \equiv$ (11b)

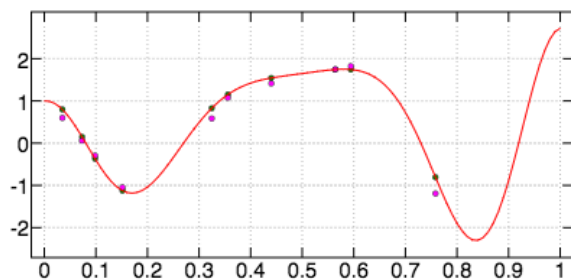
```

plotelm=: 3 : 0
  plotdatnoshow 0
  pd 'type line'
  pd 'color blue'
  xs=: (] #~ minmaxX"_ sel ]) steps (<.<./X),(>.>./X),100
  pd xs;(mkH ,. xs) mp c
  pd 'show'
)

```



```
initelm 100
0
elm 1E_3
```



```
test 0
```

11a $\langle \text{require 11a} \rangle \equiv$
`require 'trig'`
`require 'plot'`
`require 'numeric'`

(11b)

11b $\langle \text{jelm.ijs 11b} \rangle \equiv$
 $\langle \text{require 11a} \rangle$
 $\langle \text{utils 2c} \rangle$
 $\langle \text{dataset 2a} \rangle$
 $\langle \text{plotdat 3a} \rangle$
 $\langle \text{plotpoly 4b} \rangle$
 $\langle \text{polyreg 3d} \rangle$
 $\langle \text{gram 6a} \rangle$
 $\langle \text{ridge 8a} \rangle$
 $\langle \text{plotelm 10c} \rangle$
 $\langle \text{elm 10a} \rangle$
 $\langle \text{plottest 4e} \rangle$
 $\langle \text{test 4d} \rangle$