Statistics and Machine Learning

Week 15: Family of functions in Machine Learning
Neural network and Gaussian Process

Week 15 04/26 — 04/30

Contents of week 15

- Grading policy (again), no final exam for this class.
- Deadline for homework and quizzes: May 14 8AM.
- Only 2 students choose project track, the rest of you just do best finishing hw and quizzes.
- Final homework. Essay on one of the three major concepts (bias-variance tradeoff, bootstrap, generalization)
- Family of general functions: neural network, Gaussian Process
- Final remarks

Grading policy

Track A: weekly test+midterm + hw's | Track B: weekly test+midterm+hw's+project

- Track A (the usual track): weekly test 30% + midterm 30% + hw’s (about 10 assignments) 40%
- Track B (the project track): weekly test 30% + midterm 30% + hw’s before midterm 20% + project 20%
- Each project team can have two students, and both students must show strong evidence that they can do independent project (i.e. good midterm grade, strong project proposal, passion about applications, etc)

Student instructional rating survey

https://sirs.ctaar.rutgers.edu/blue

5 building blocks for ML tasks

Data set

Model

Loss function

Training error

Test error

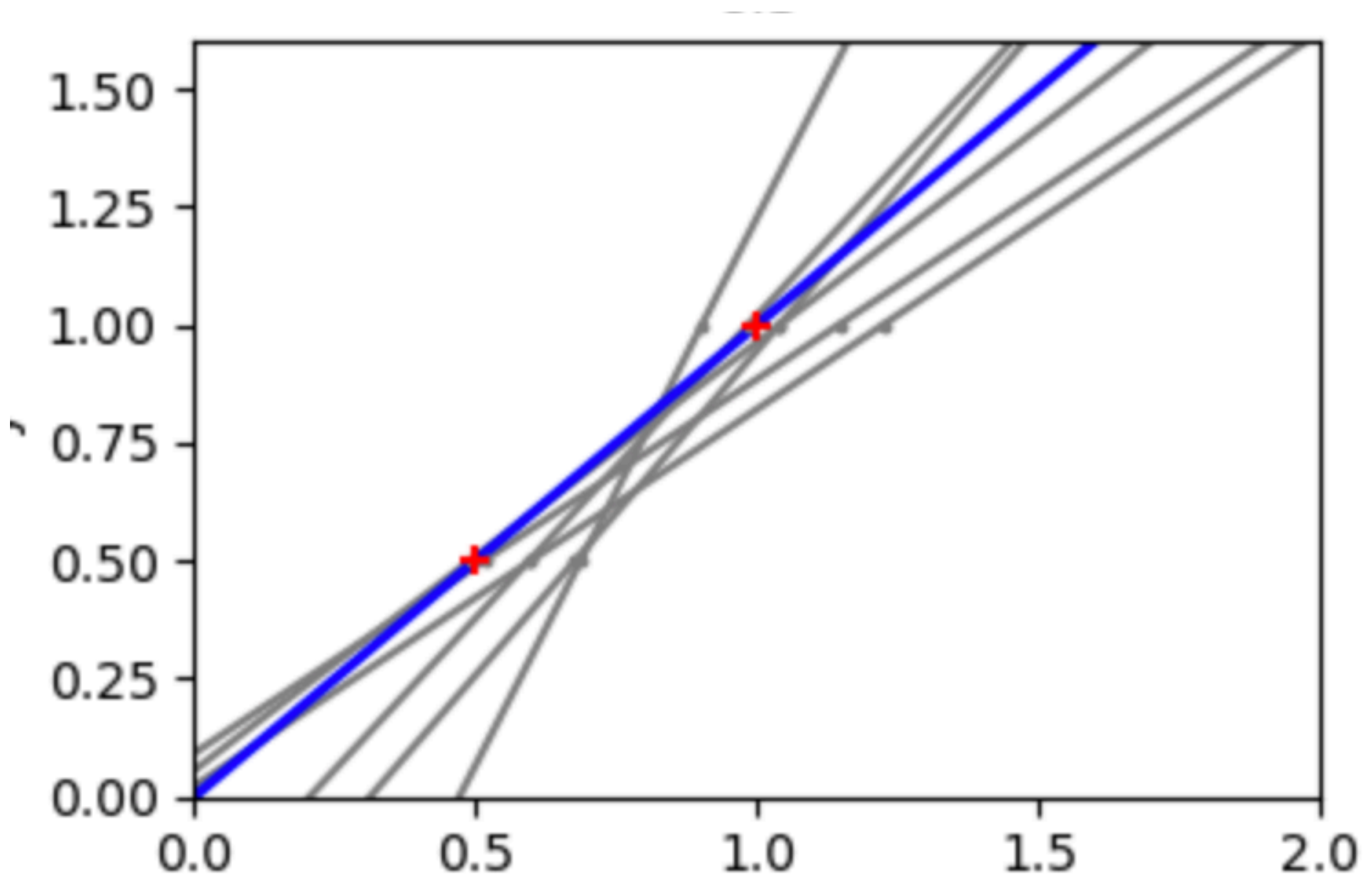
Two ways of describing a function

Two ways of describing a function

$\boxed{\text{Parametric: say your parameters}}$

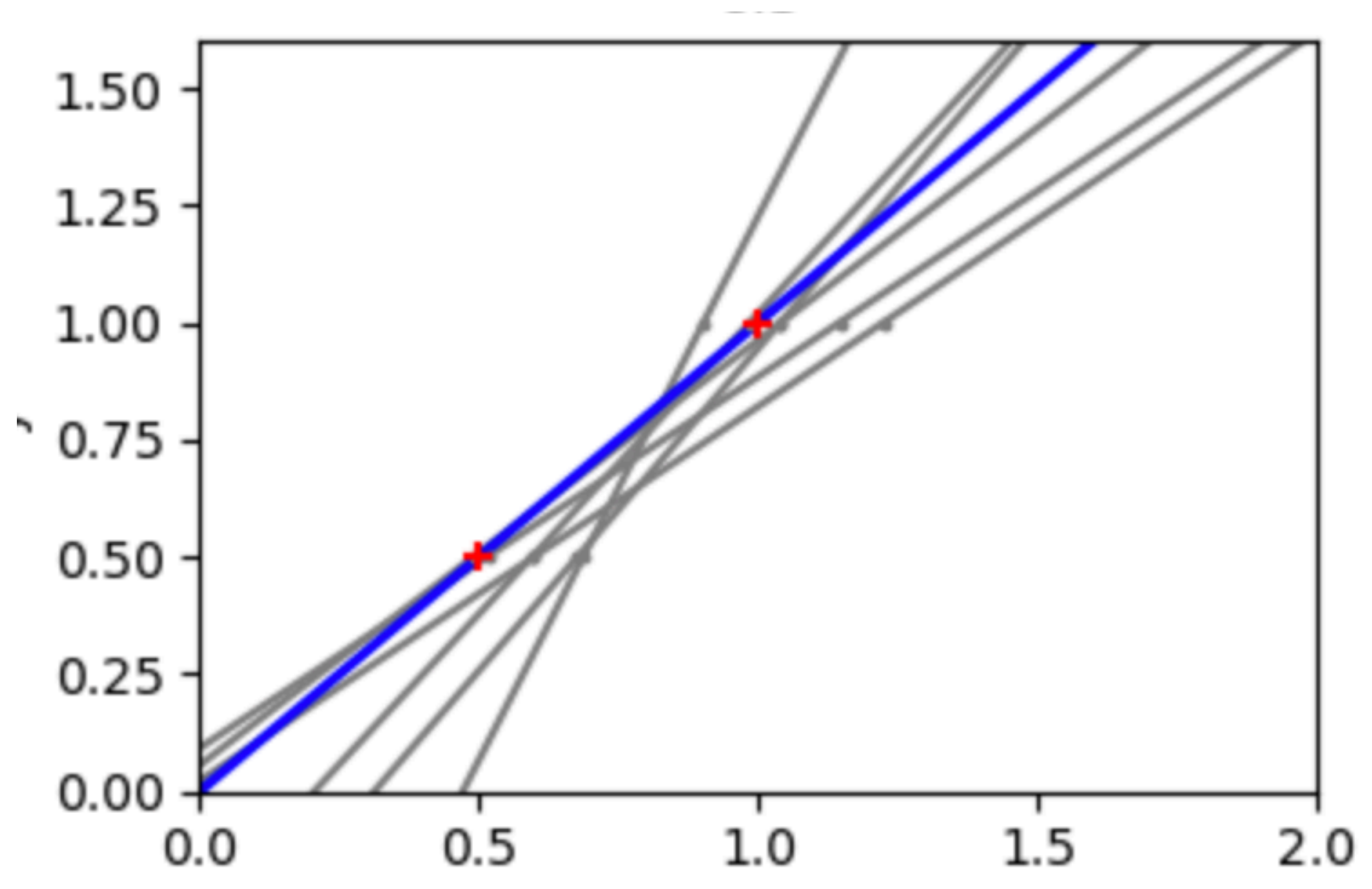
Two ways of describing a function

Parametric: say your parameters



Two ways of describing a function

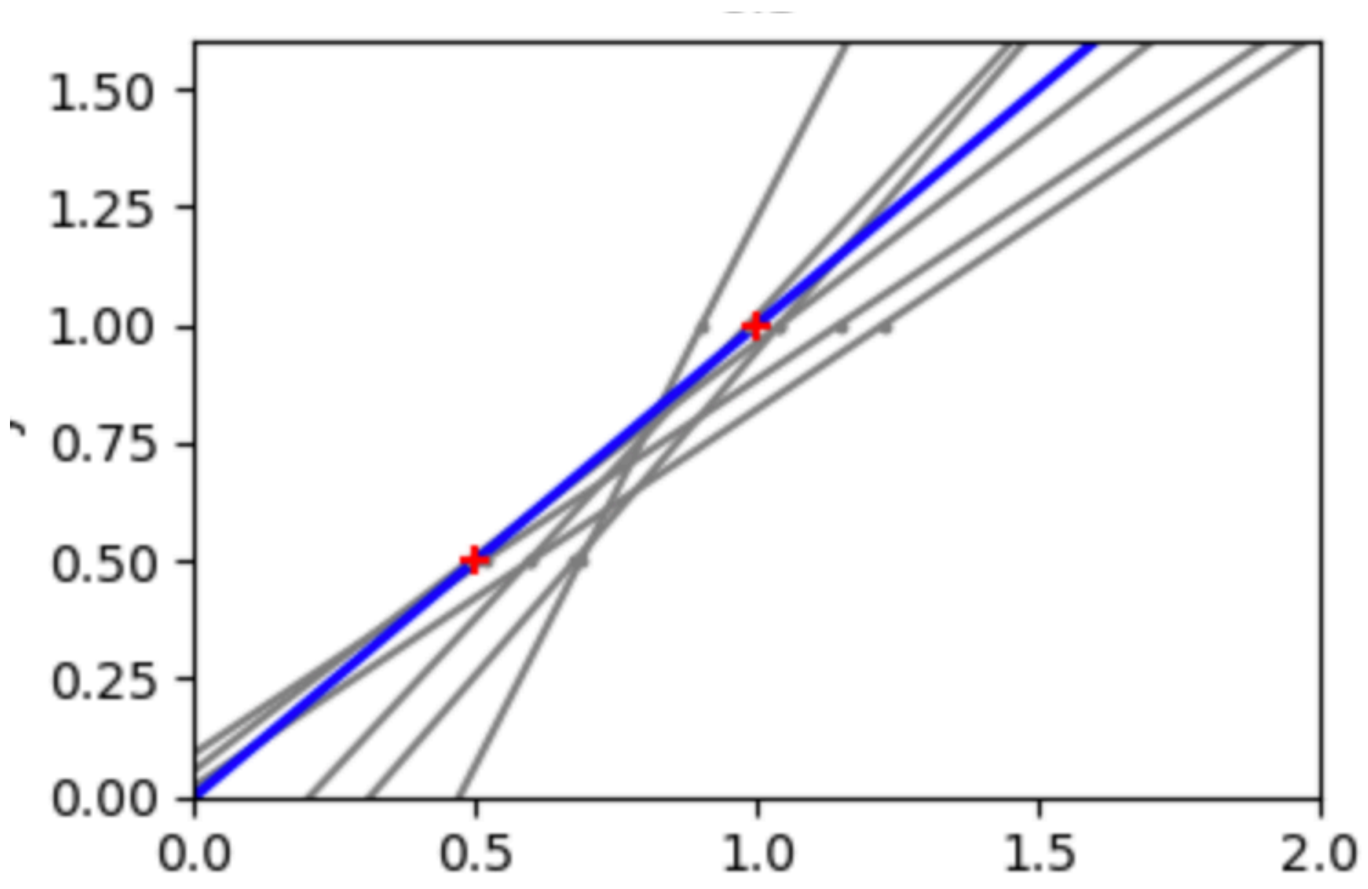
Parametric: say your parameters



$$f(x) = wx + b$$

Two ways of describing a function

Parametric: say your parameters



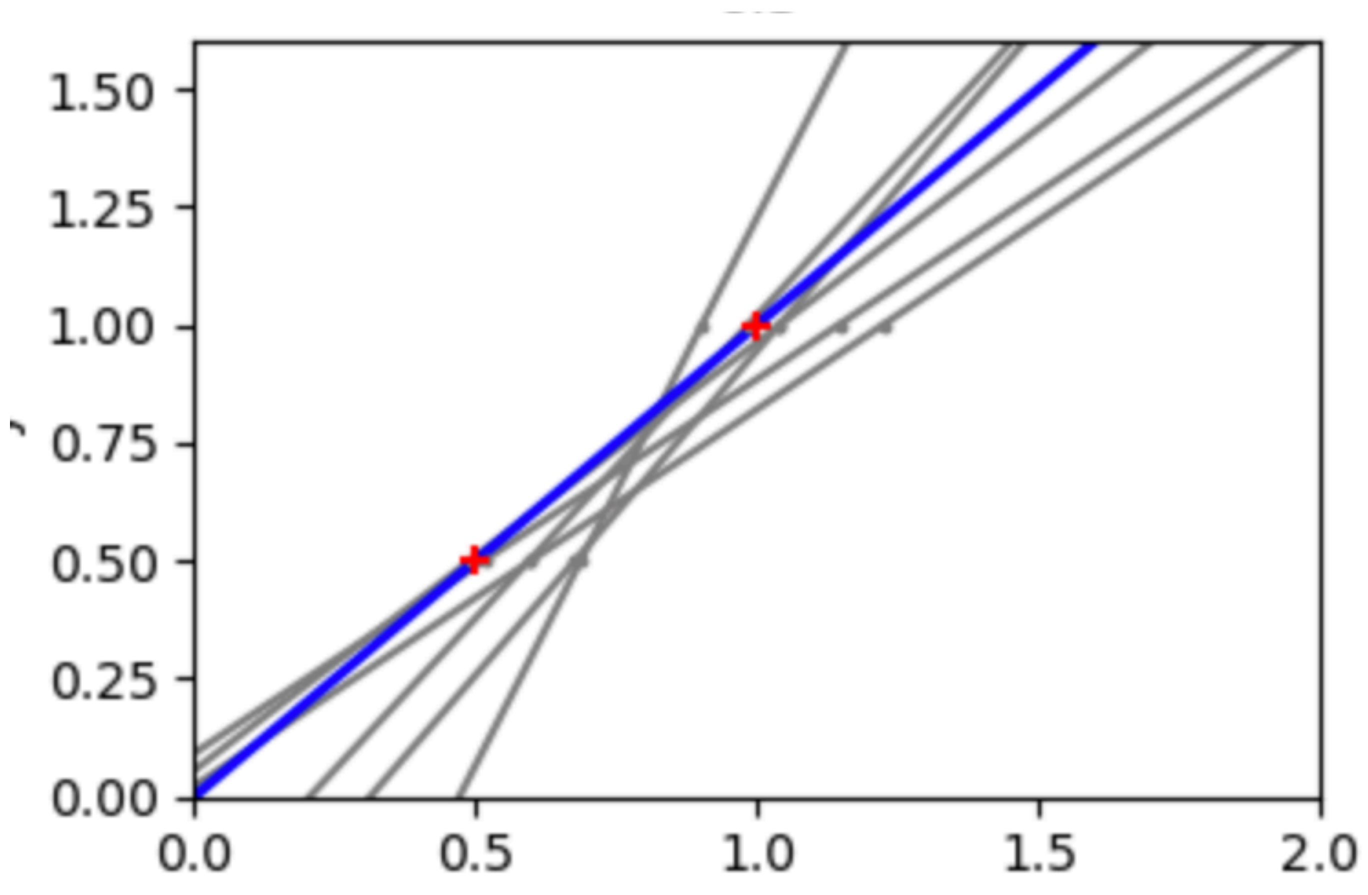
$$f(x) = wx + b$$

W and b are parameters of the linear function, they determine the behavior of the function!

Two ways of describing a function

Parametric: say your parameters

non-parametric: say your y-values

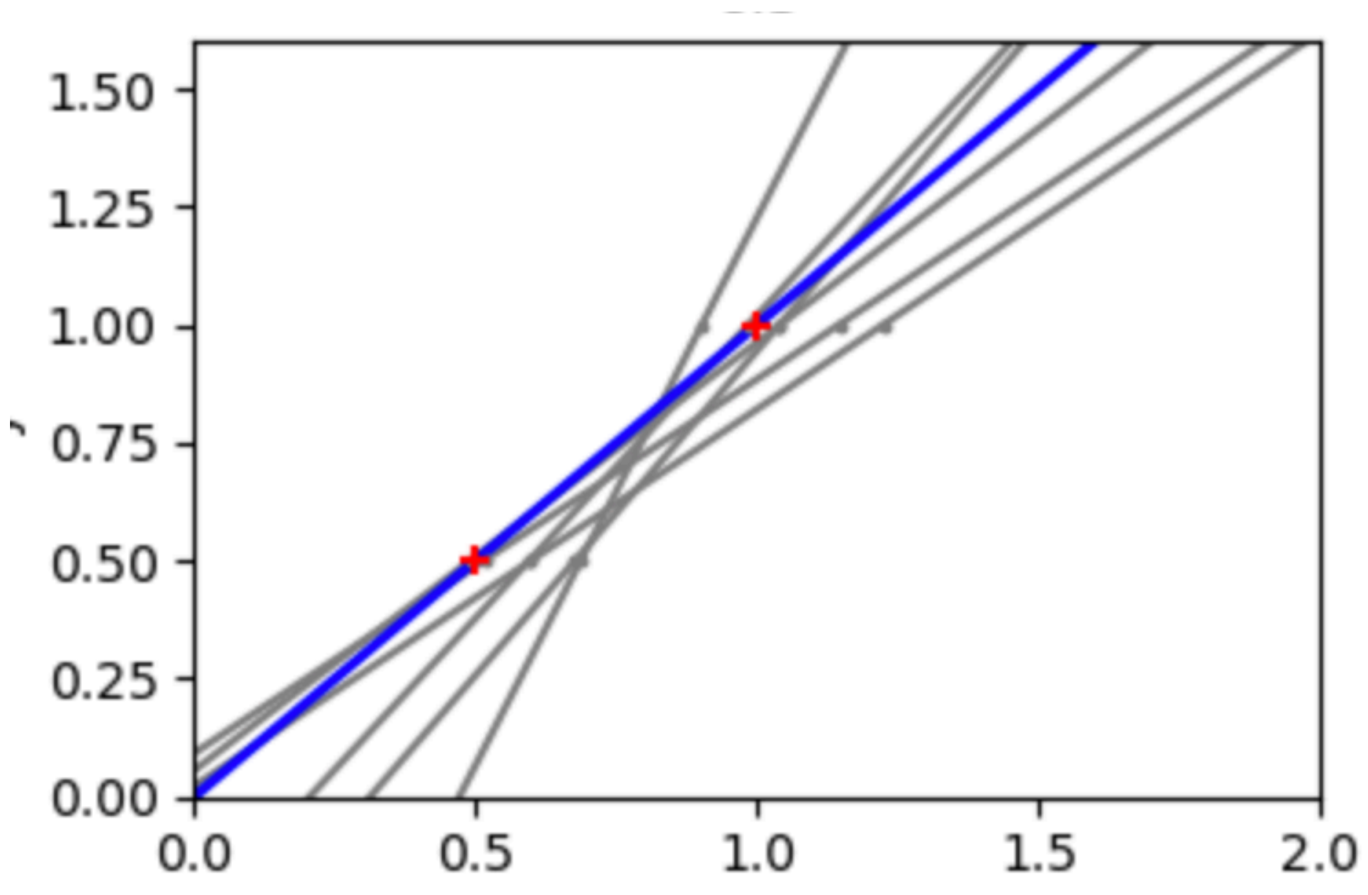


$$f(x) = wx + b$$

W and b are parameters of the linear function, they determine the behavior of the function!

Two ways of describing a function

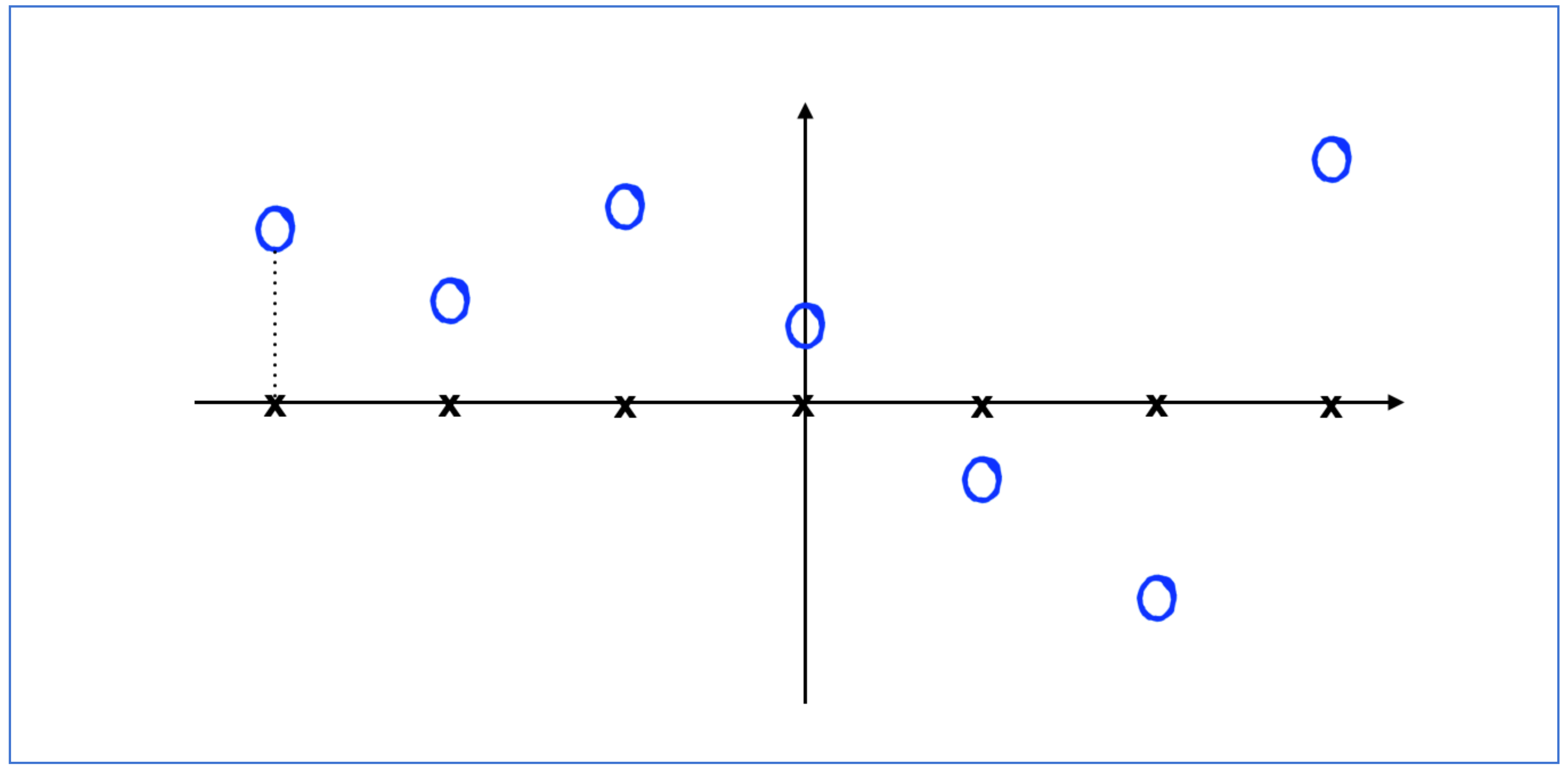
Parametric: say your parameters



$$f(x) = wx + b$$

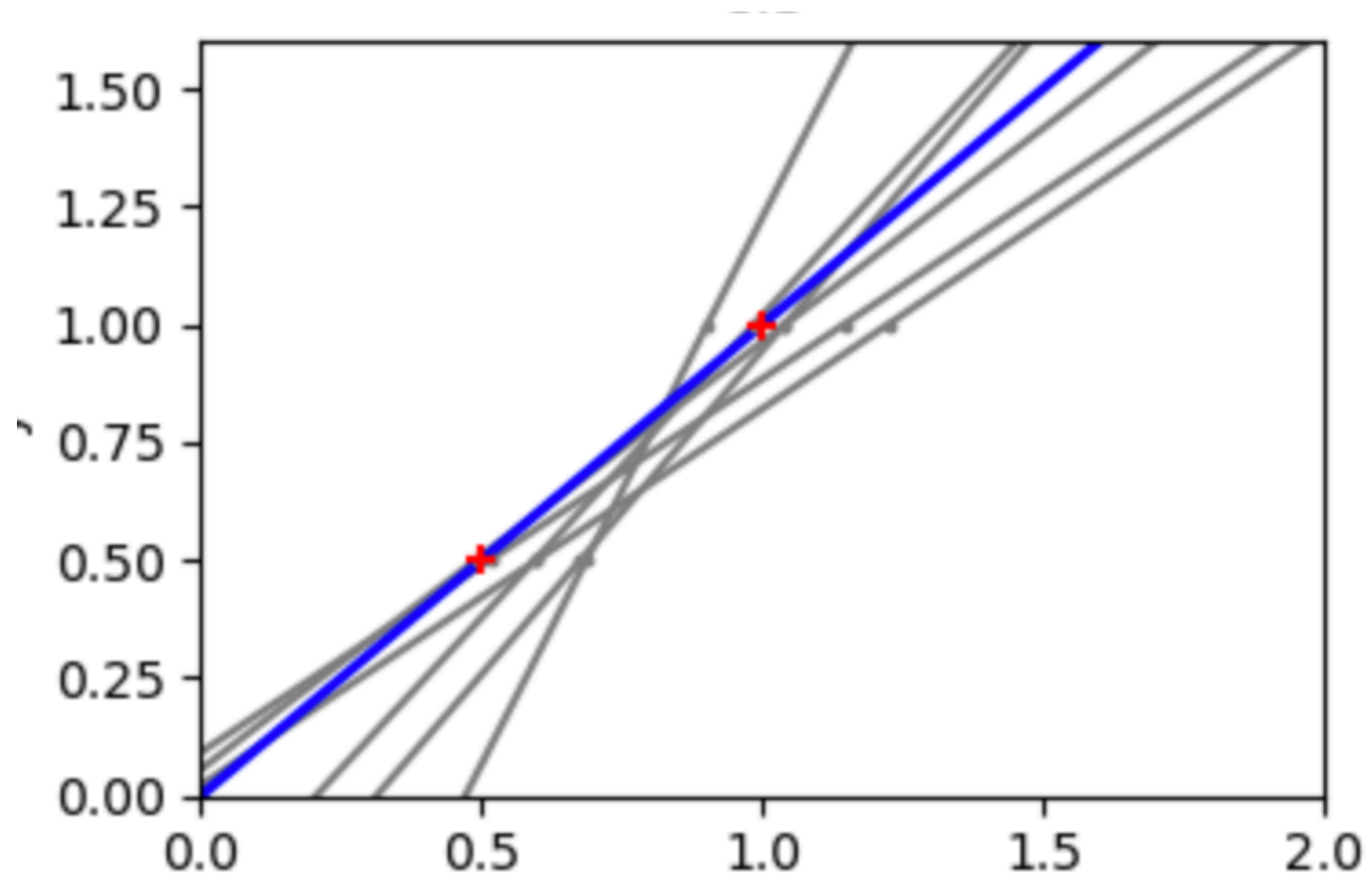
W and b are parameters of the linear function, they determine the behavior of the function!

non-parametric: say your y-values



Two ways of describing a function

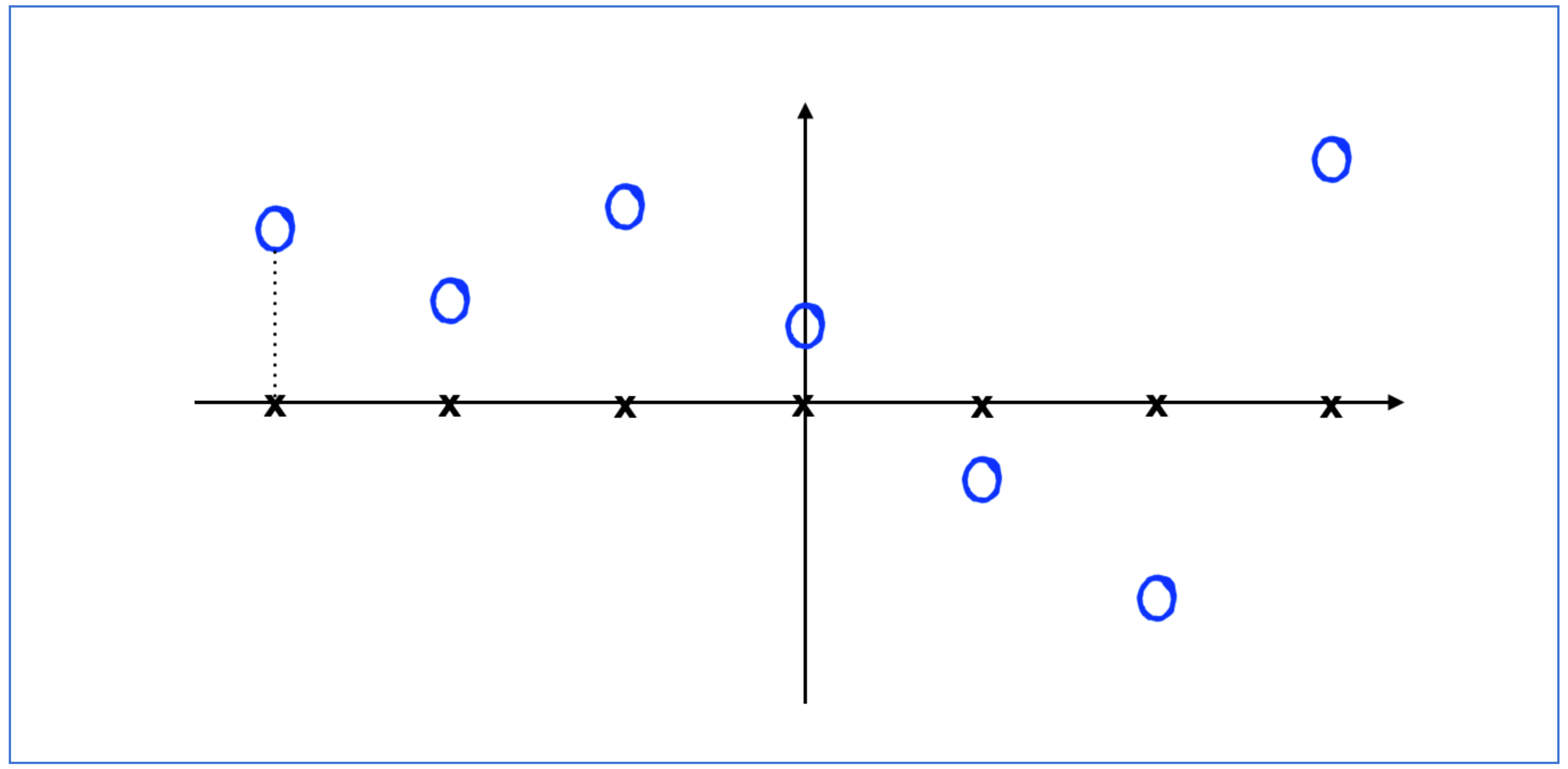
Parametric: say your parameters



$$f(x) = wx + b$$

W and b are parameters of the linear function, they determine the behavior of the function!

non-parametric: say your y-values

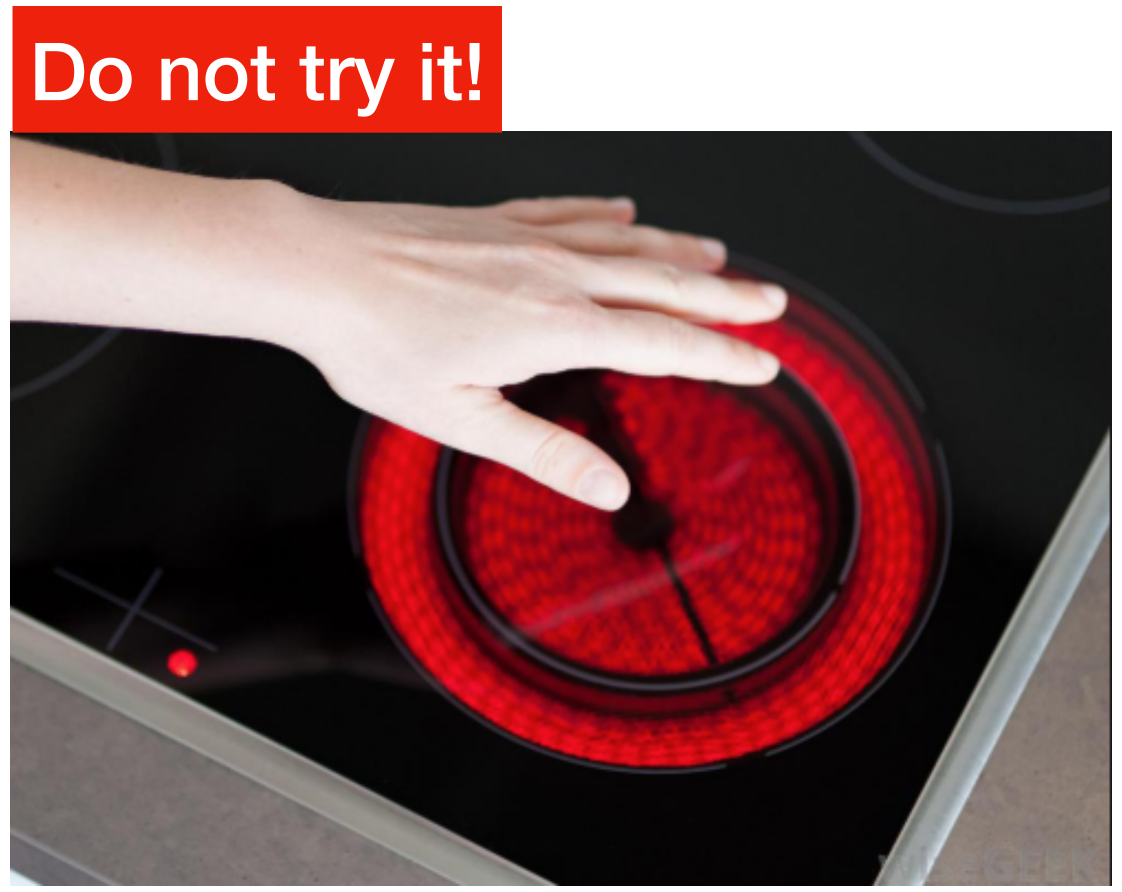


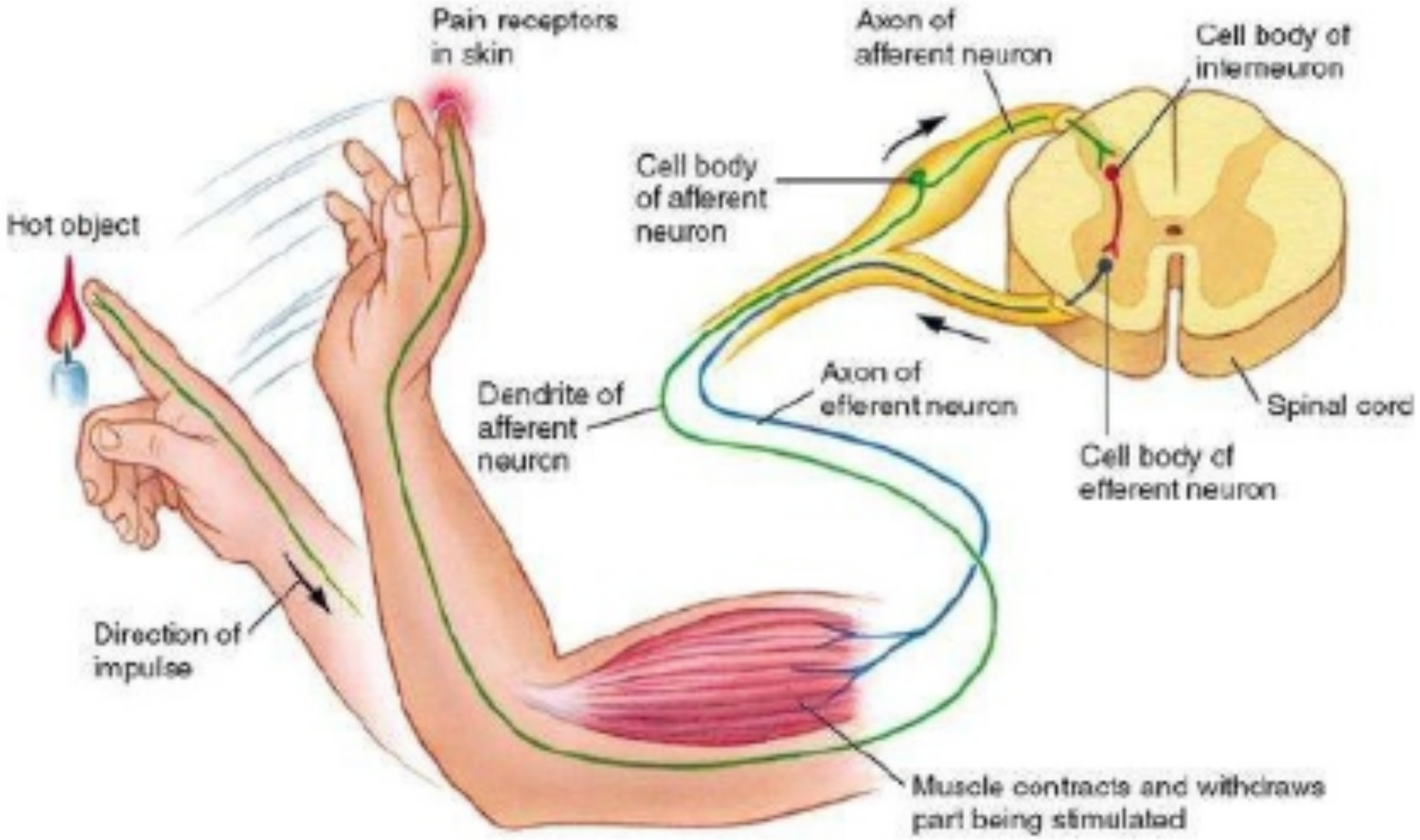
How to describe your function? Simple!.
f(x1)=y1, f(x2)=y2, ...,

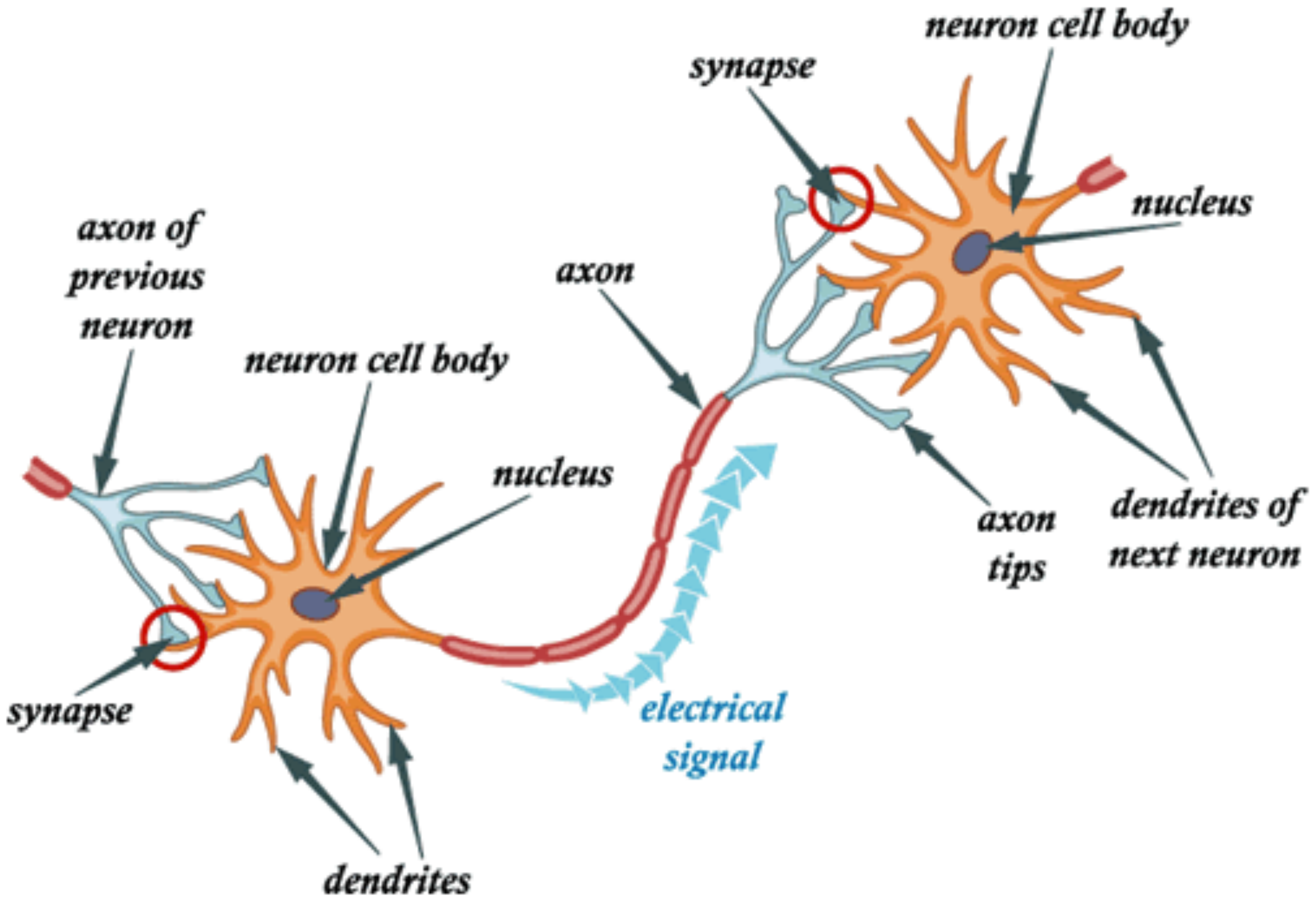
How neural networks work?

Do not try it!

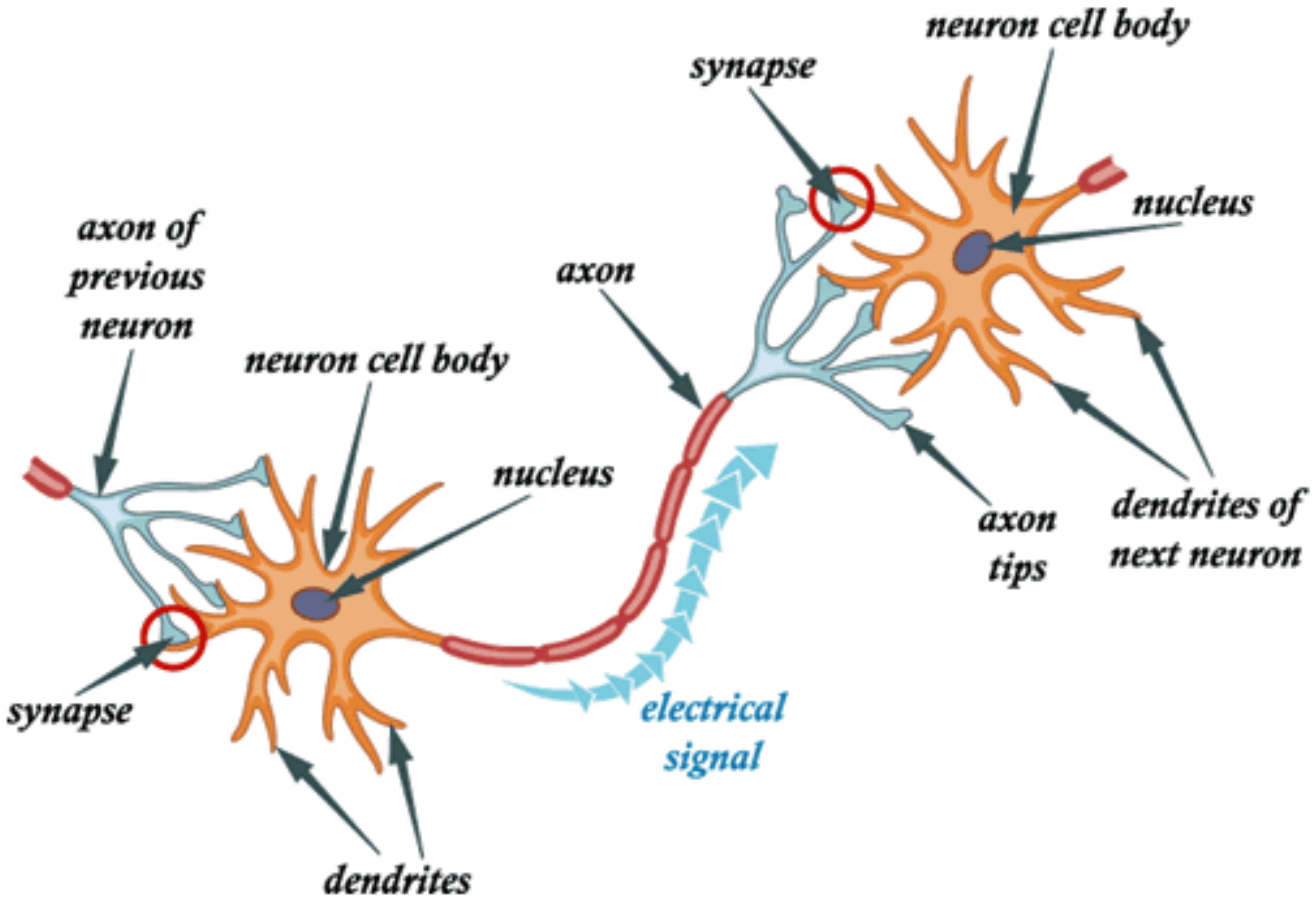
How neural networks work?

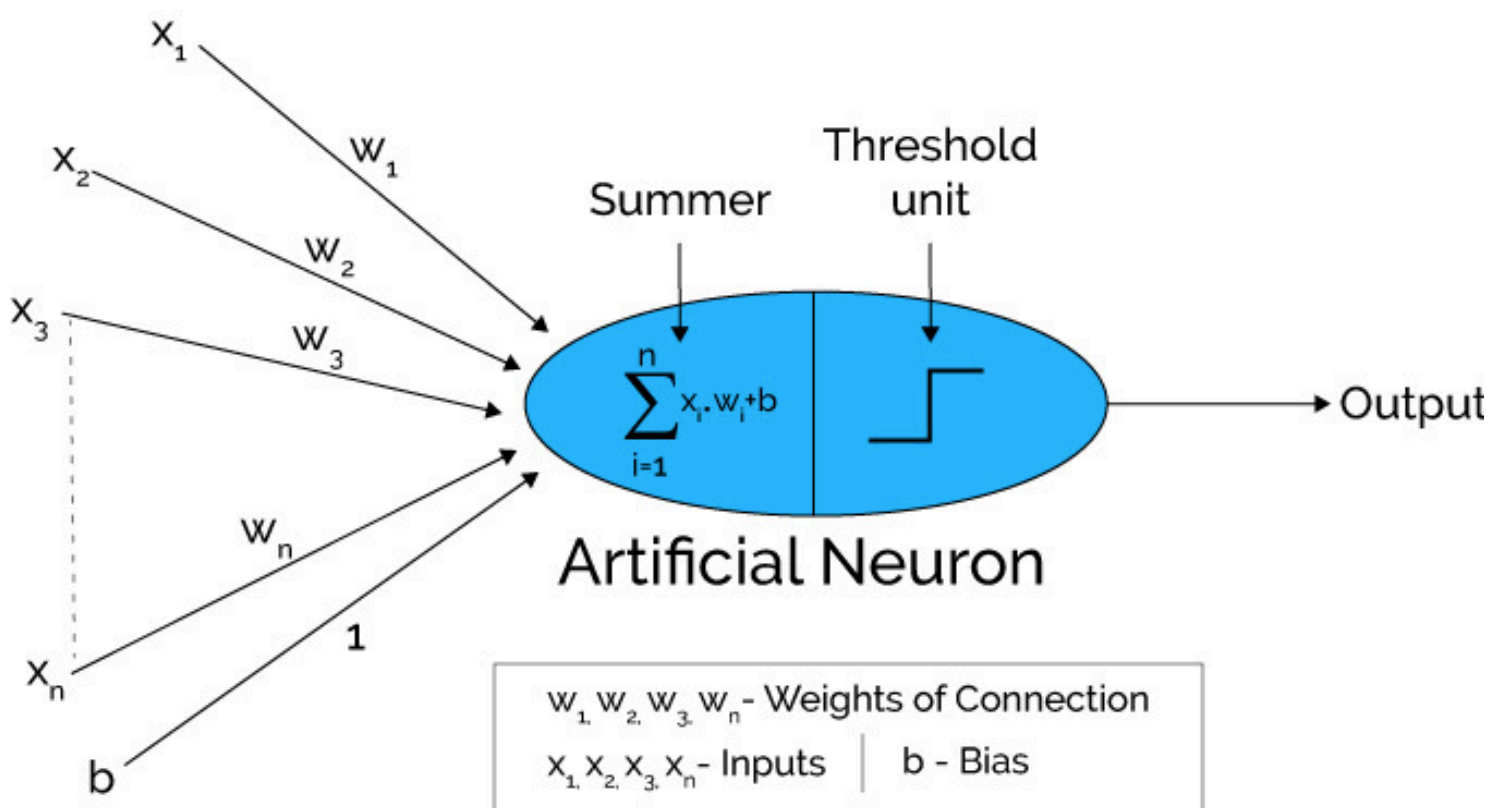




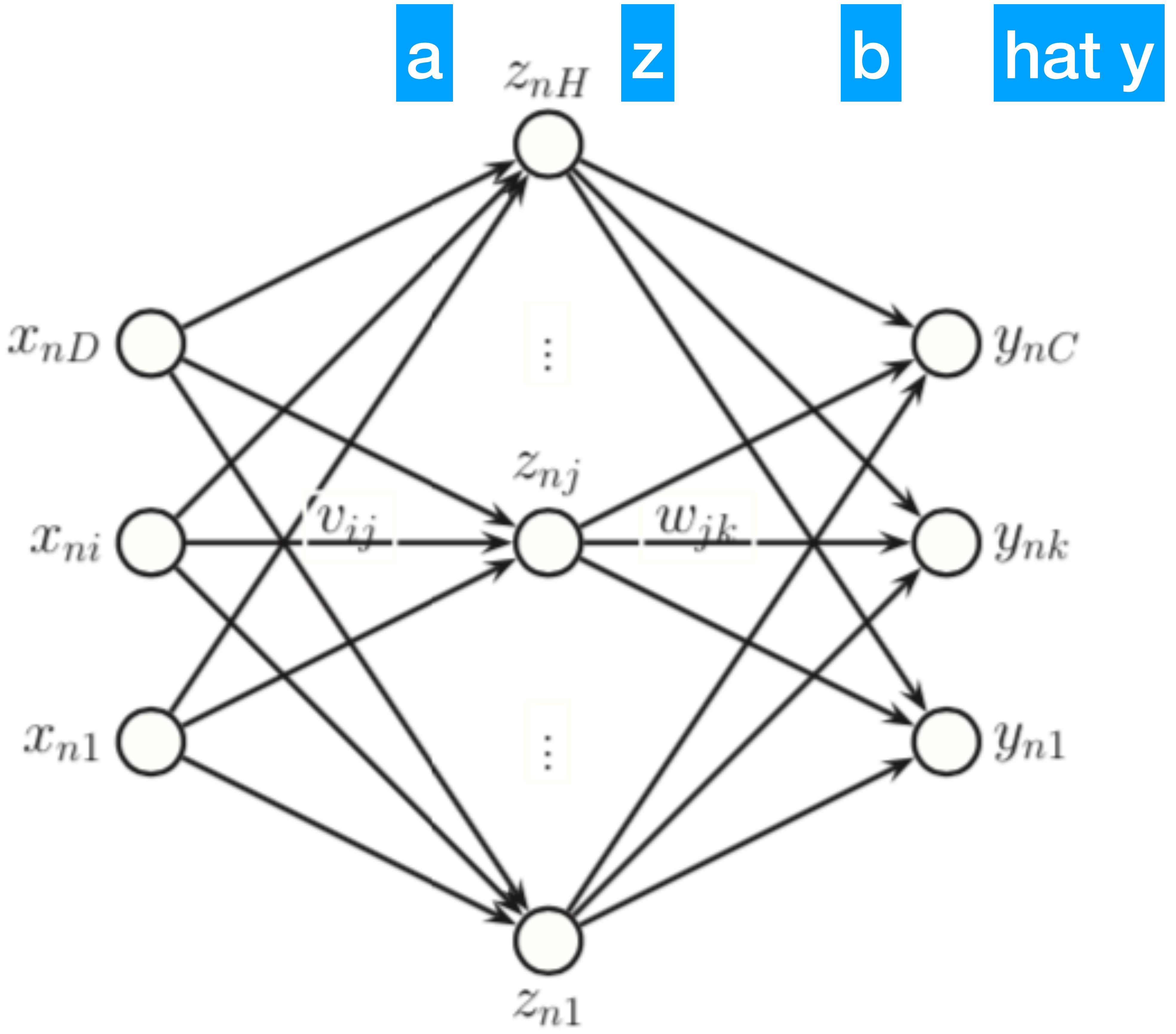


axon of previous neuron
neuron cell body
nucleus
synapse
dendrites
axon
electrical signal
synapse
neuron cell body
nucleus
axon tips
dendrites of next neuron


neuron cell body
synapse
nucleus
axon of previous neuron
axon
neuron cell body
nucleus
axon tips
dendrites of next neuron
synapse
electrical signal
dendrites


x_1
x_2
x_3
x_n
b
w_1
w_2
w_3
w_n
1
Summer
Threshold unit
$\sum_{i=1}^{n} x_i.w_i+b$
Output
Artificial Neuron
w_1, w_2, w_3, w_n- Weights of Connection
x_1, x_2, x_3, x_n- Inputs
b - Bias

N.N. as universal function



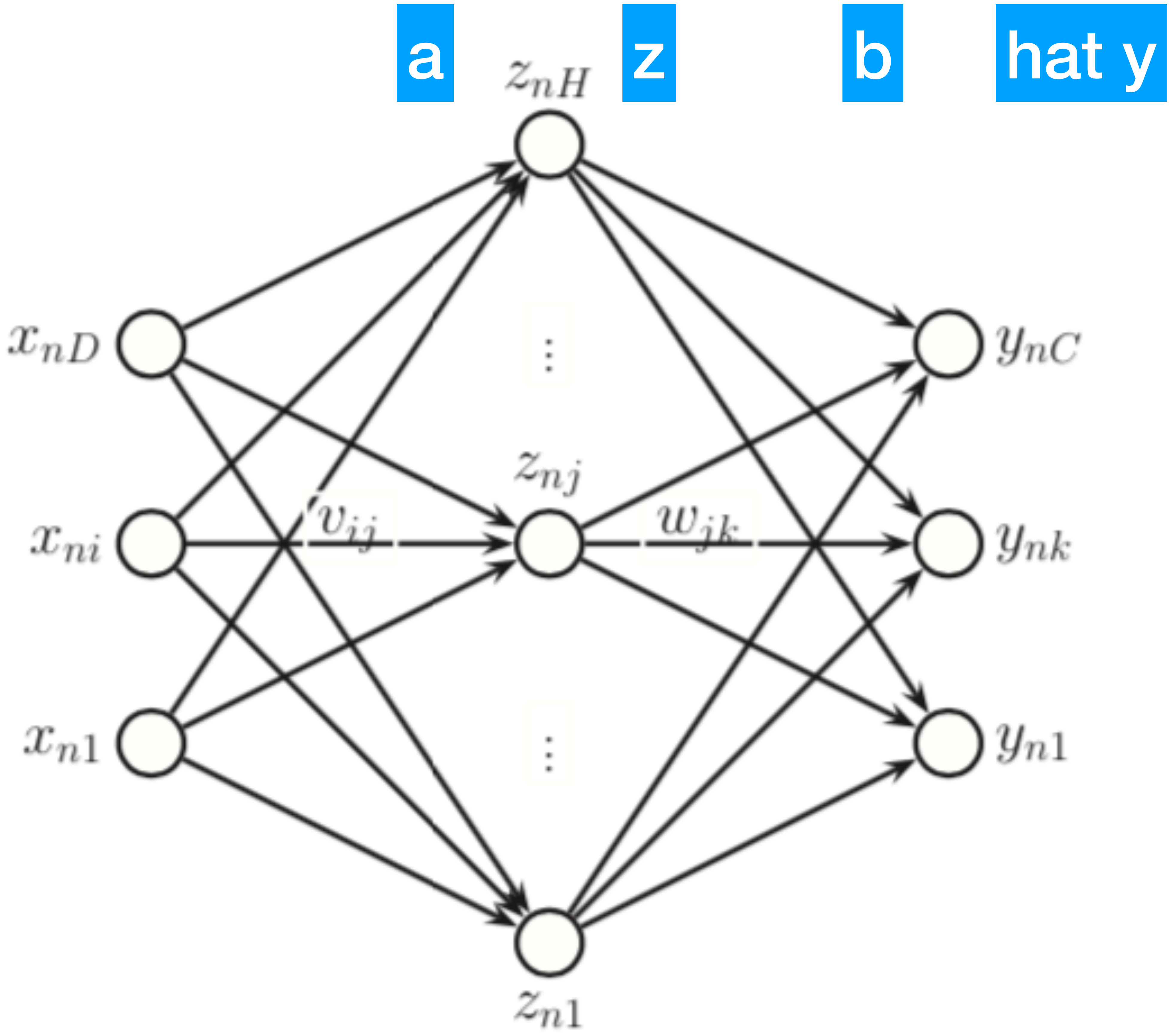
$$\mathbf{a} = \mathbf{W}\mathbf{x}$$

$$\mathbf{z} = g(\mathbf{a})$$

$$\mathbf{b} = \mathbf{V}\mathbf{z}$$

$$\boxed{\hat{\mathbf{y}} = \mathbf{b}} \qquad \boxed{\hat{\mathbf{y}} = h(\mathbf{b})}$$

N.N. as universal function



$$\mathbf{a} = \mathbf{W}\mathbf{x}$$

$$\mathbf{z} = g(\mathbf{a})$$

Activation

$$\mathbf{b} = \mathbf{V}\mathbf{z}$$

$$\boxed{\hat{\mathbf{y}} = \mathbf{b}} \qquad \boxed{\hat{\mathbf{y}} = h(\mathbf{b})}$$

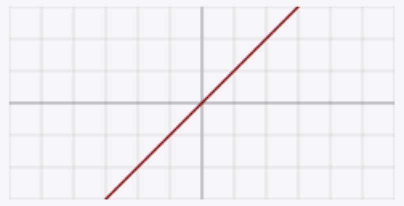
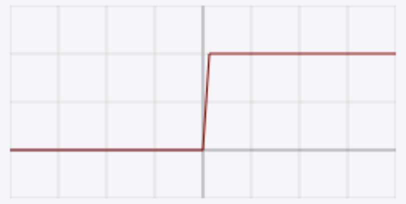
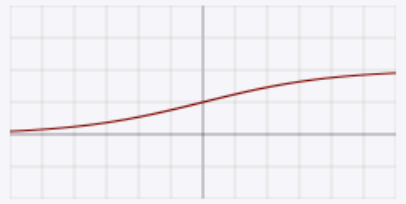
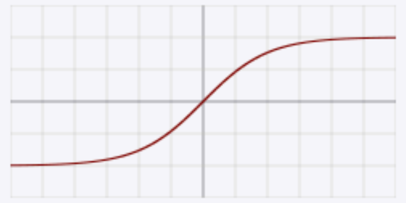
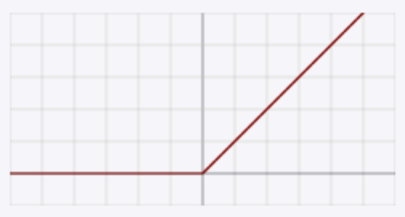
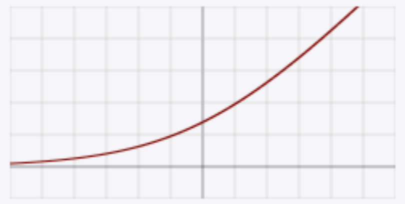
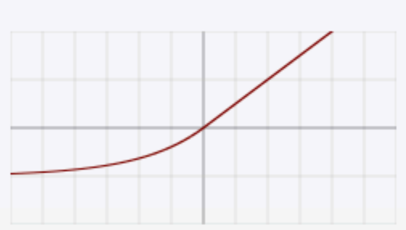
Many types of activation functions

WIKIPEDIA
The Free Encyclopedia

Article Talk

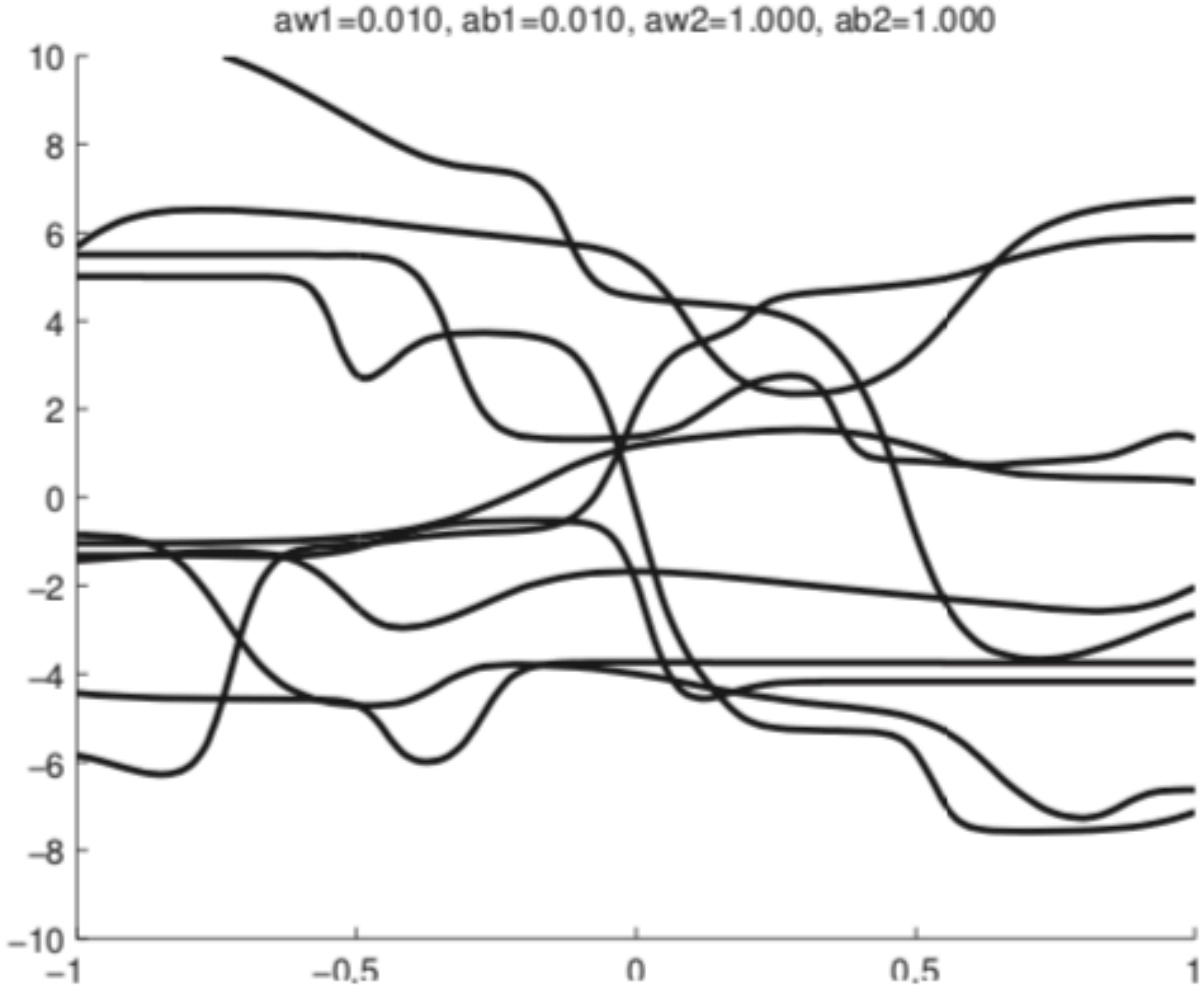
Activation function

From Wikipedia, the free encyclopedia

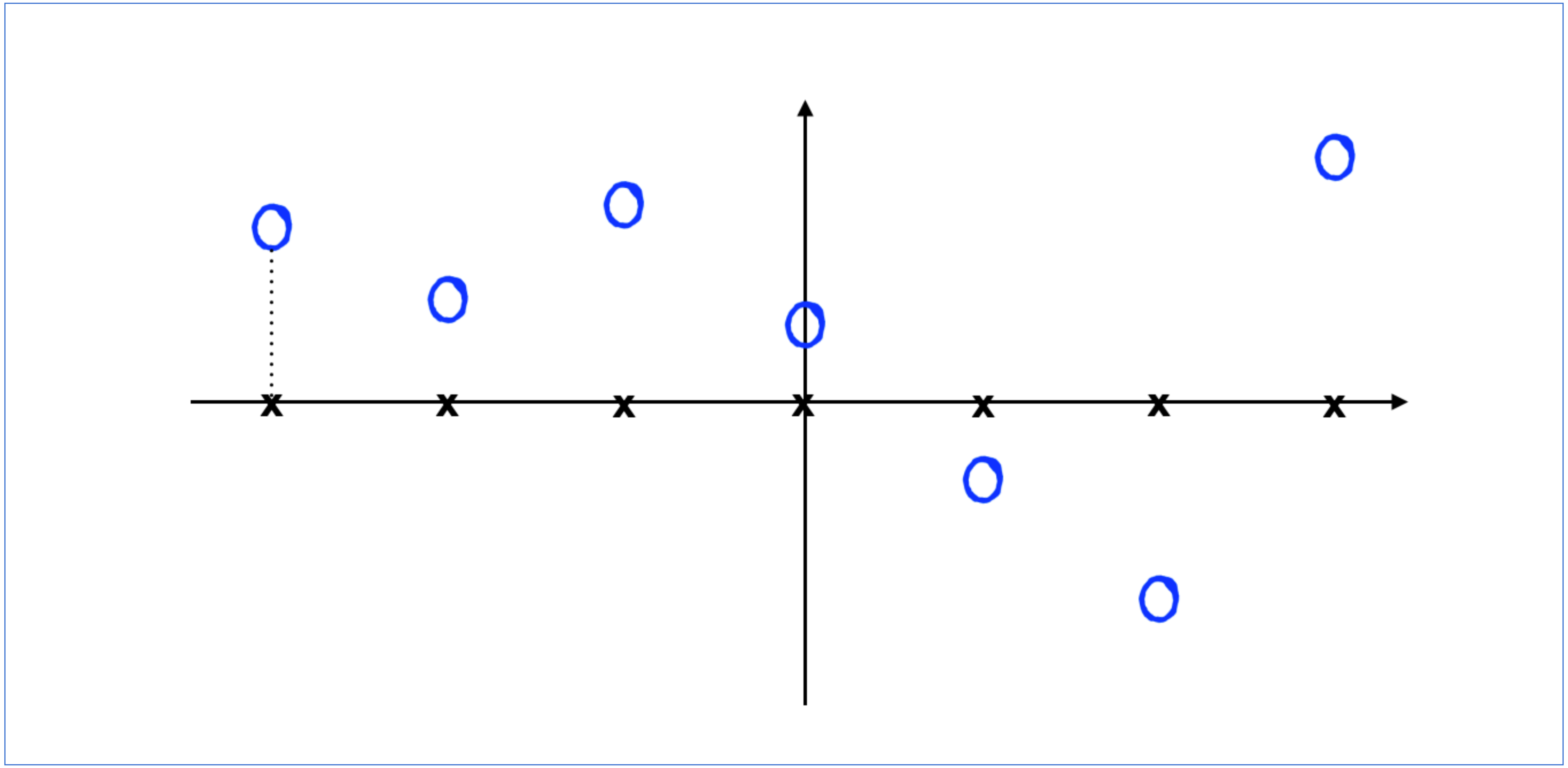
Identity		x	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1+e^{-x}}$ [1]	$f(x)(1-f(x))$	$(0, 1)$
tanh		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[7]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$
Gaussian Error Linear Unit (GELU)[4]		$\frac{1}{2}x\left(1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$ $= x\Phi(x)$	$\Phi(x) + x\phi(x)$	$(-0.17\ldots, \infty)$
Softplus[8]		$\ln(1+e^x)$	$\frac{1}{1+e^{-x}}$	$(0, \infty)$
Exponential linear unit (ELU)[9]		$\begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ with parameter α	$\begin{cases} \alpha e^x & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ 1 & \text{if } x = 0 \text{ and } \alpha = 1 \end{cases}$	$(-\alpha, \infty)$

Random functions from random parameters

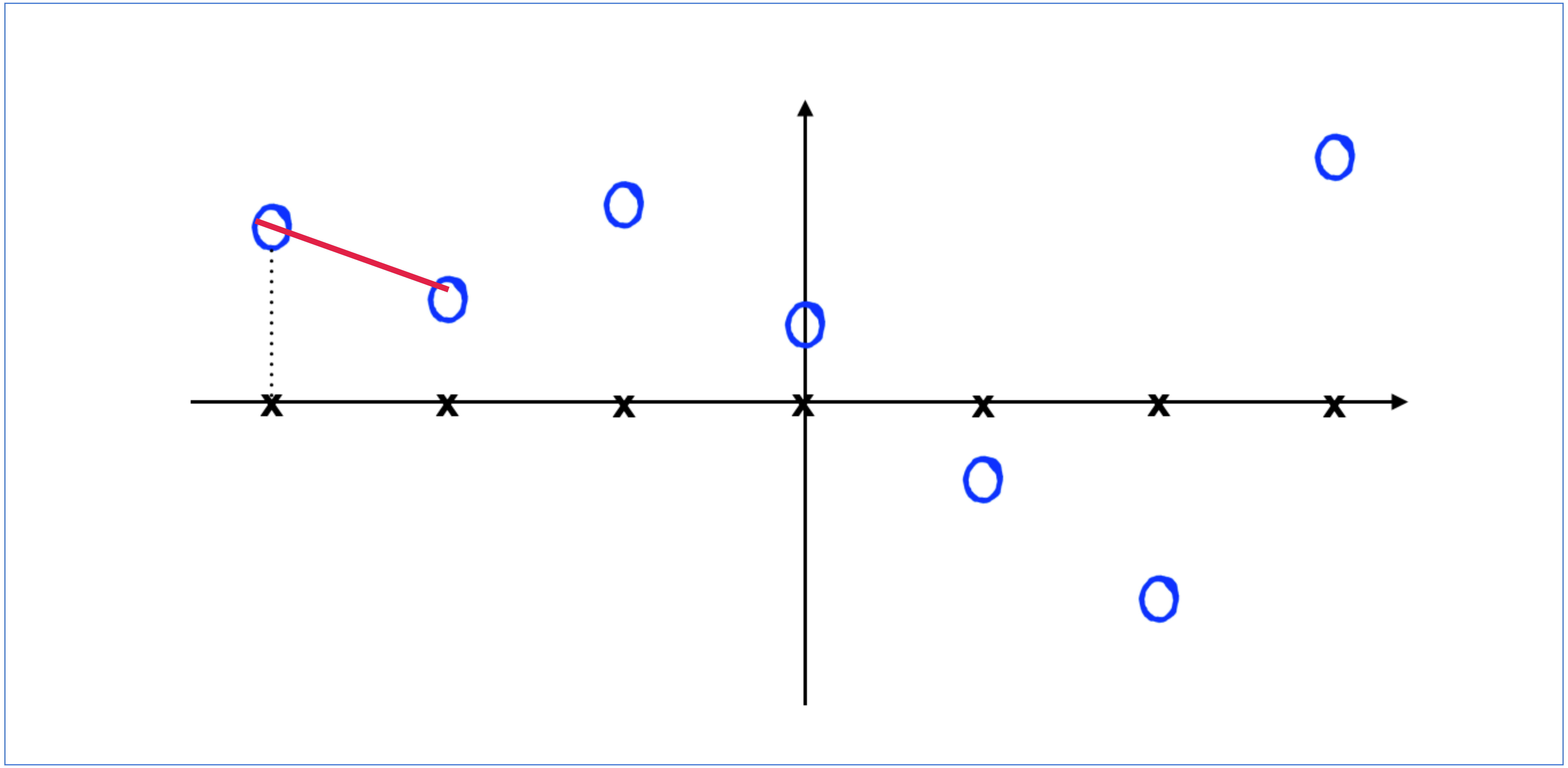
Random functions from random parameters



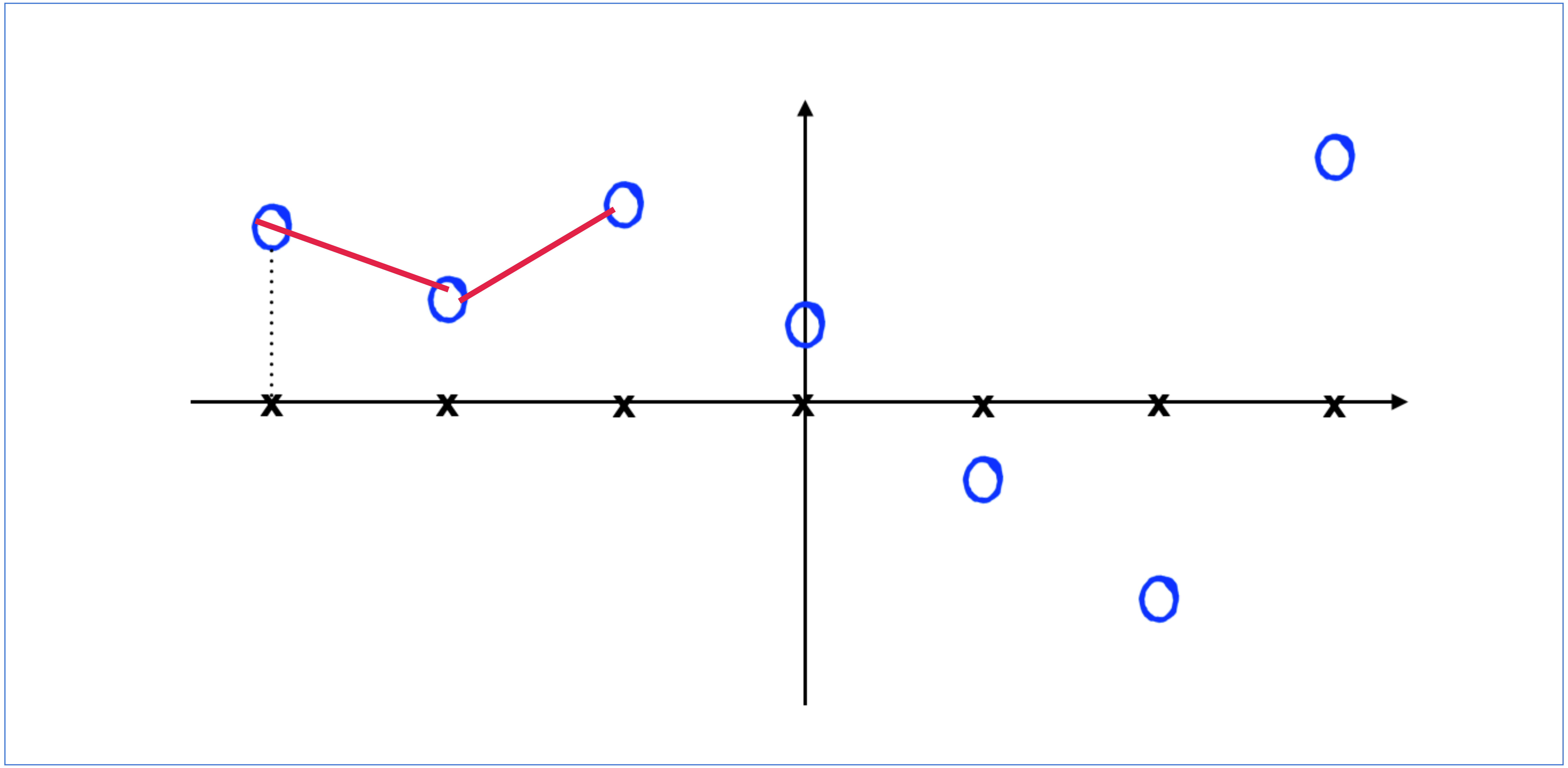
Non-parametric description of a function



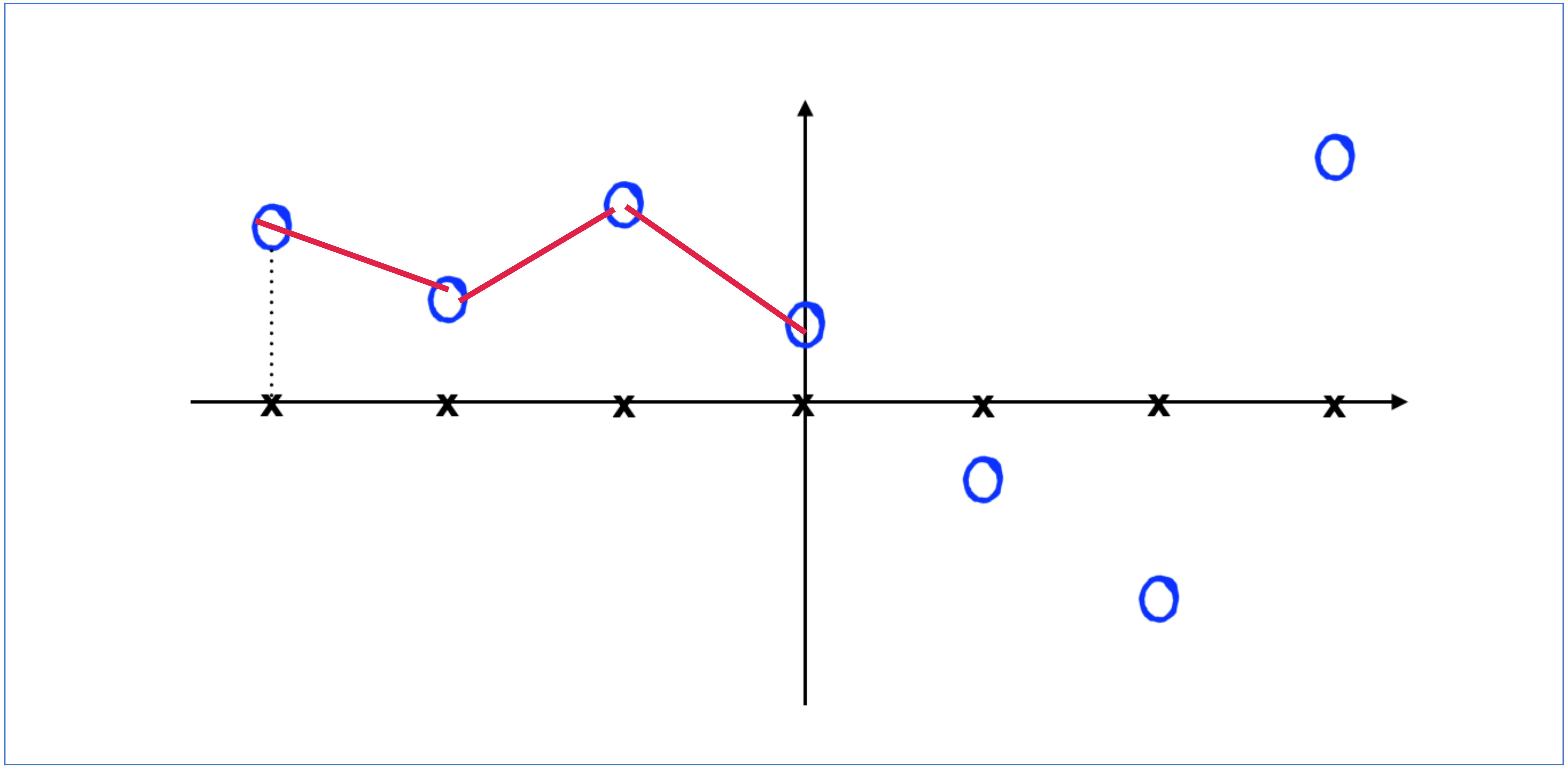
Non-parametric description of a function



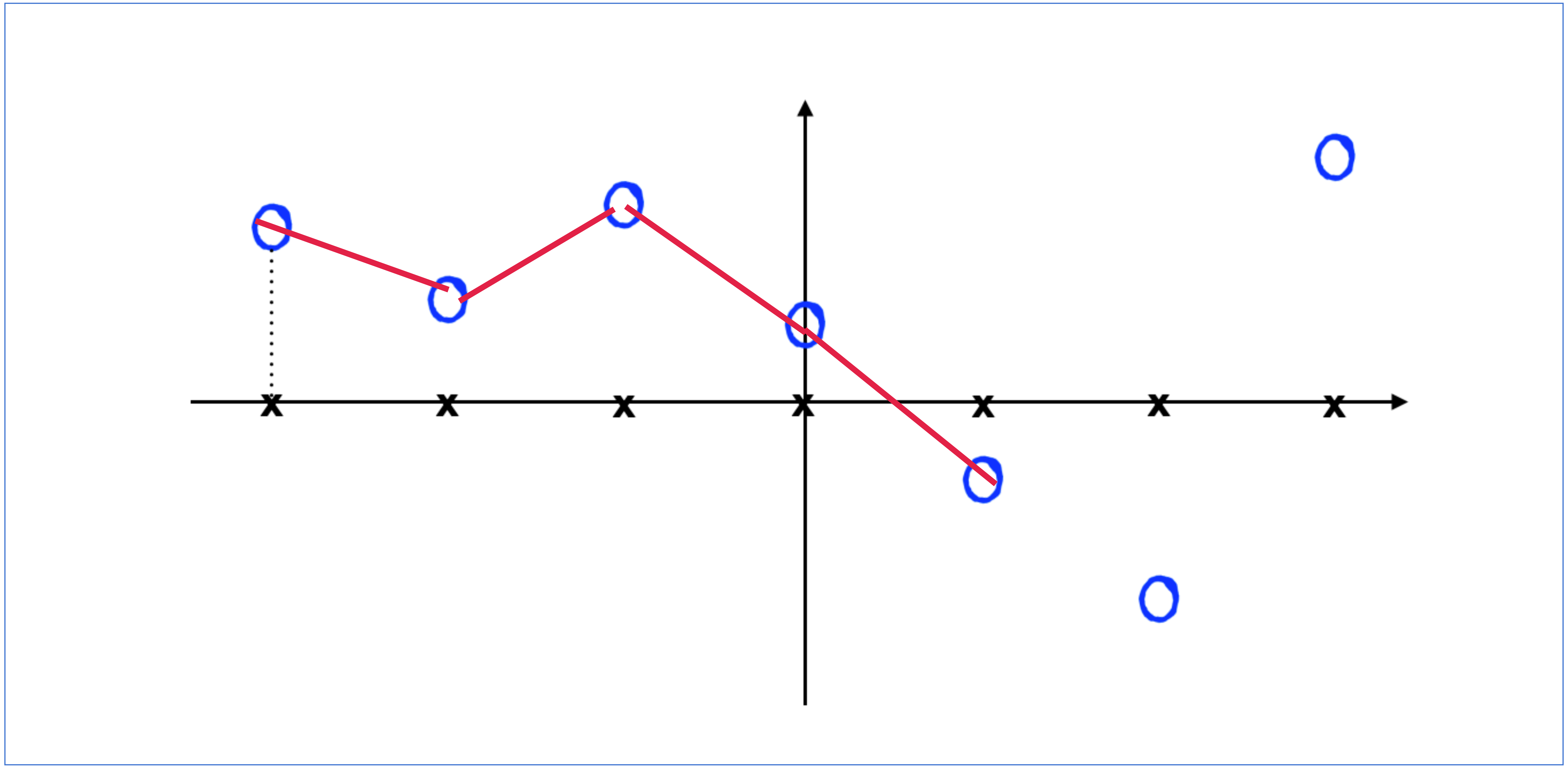
Non-parametric description of a function



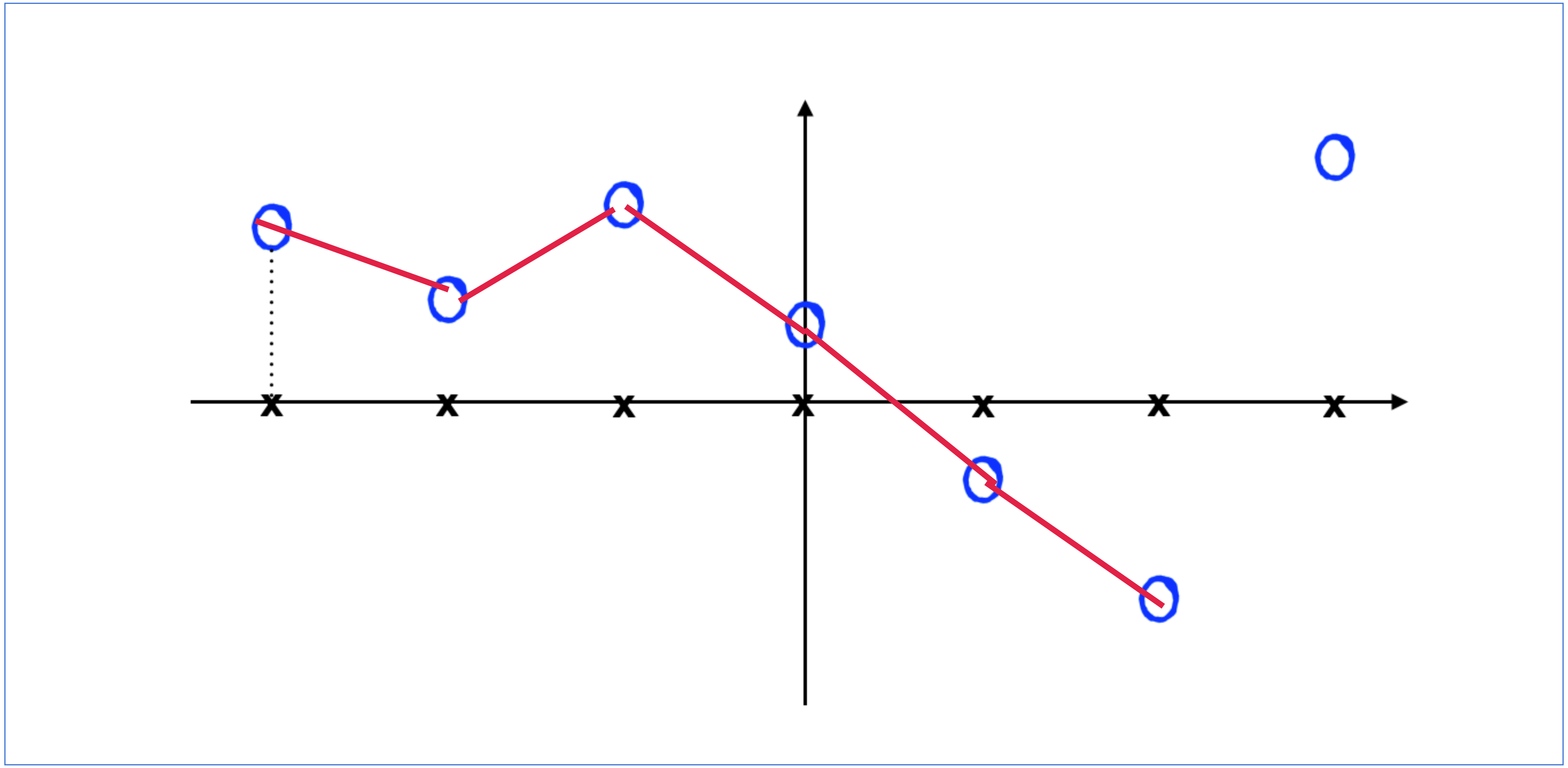
Non-parametric description of a function



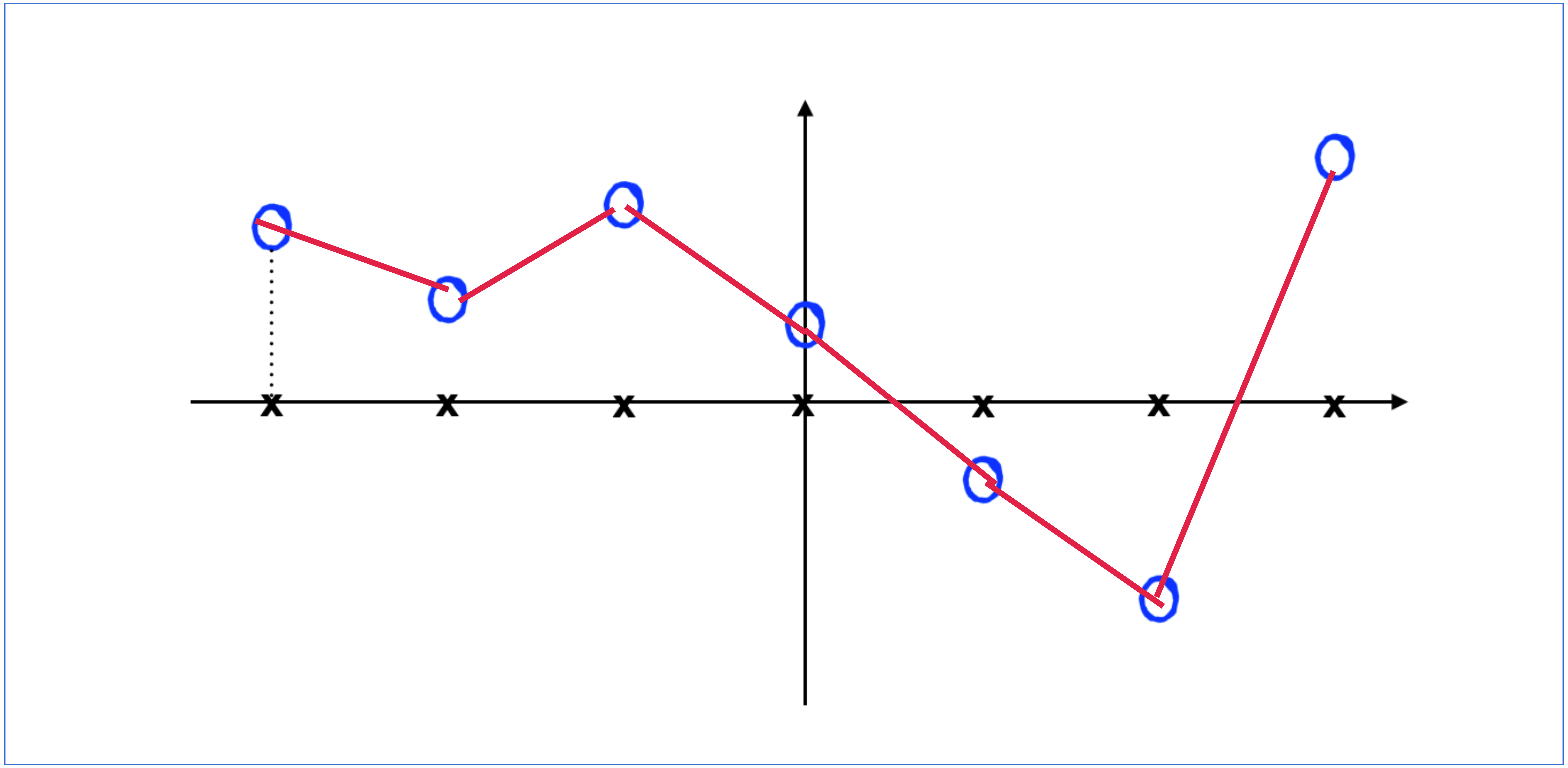
Non-parametric description of a function



Non-parametric description of a function



Non-parametric description of a function



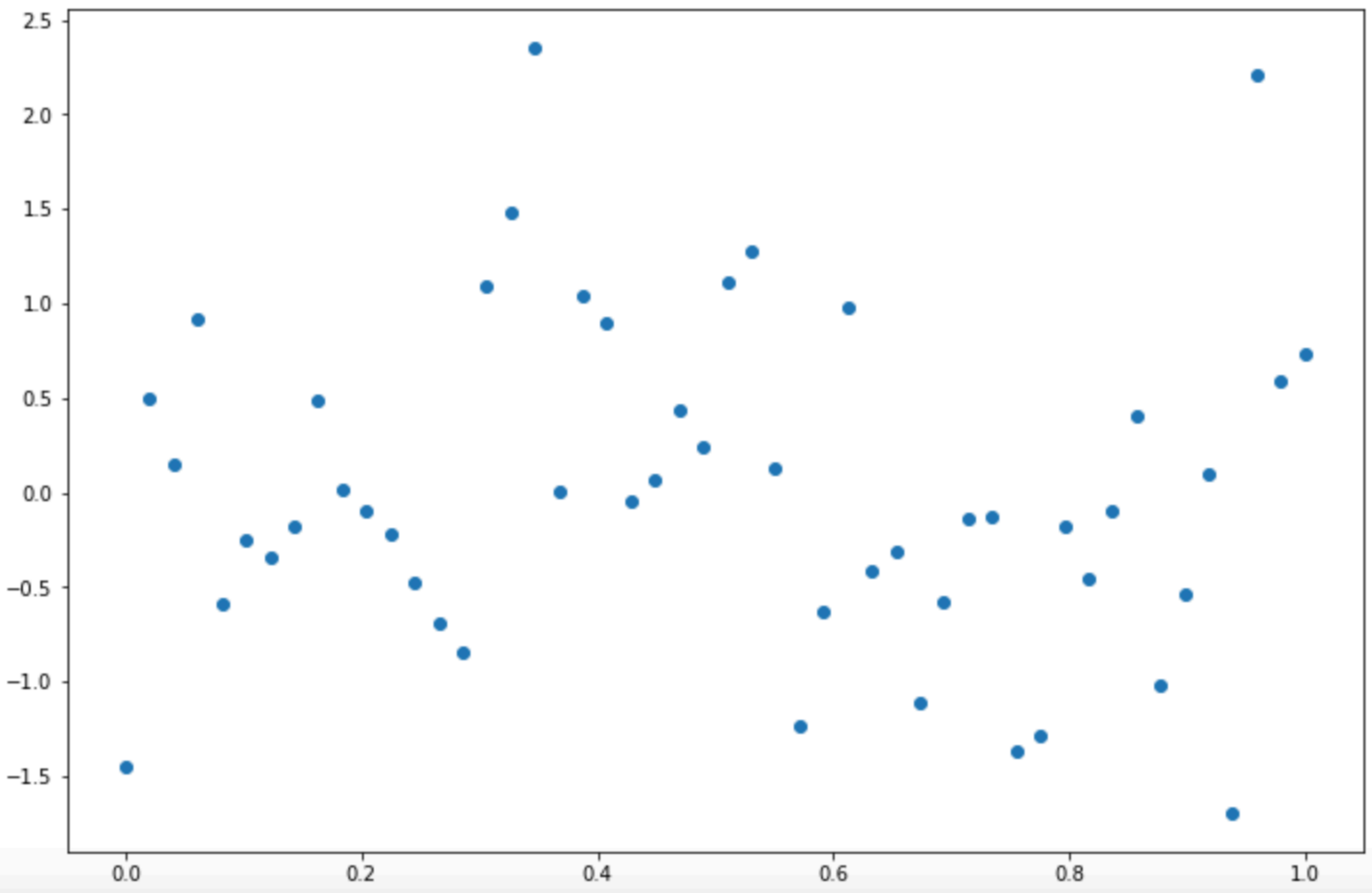
Sampling a function from prior GP

Now let me generate 50 x's in [0,1], and also generate 50 random y's.

In [35]:
```
z = np.random.normal(0,1,size=50)
```

In [36]:
```
plt.scatter(x_input, z)
```

Out[36]: <matplotlib.collections.PathCollection at 0x7fdc7184e1d0>



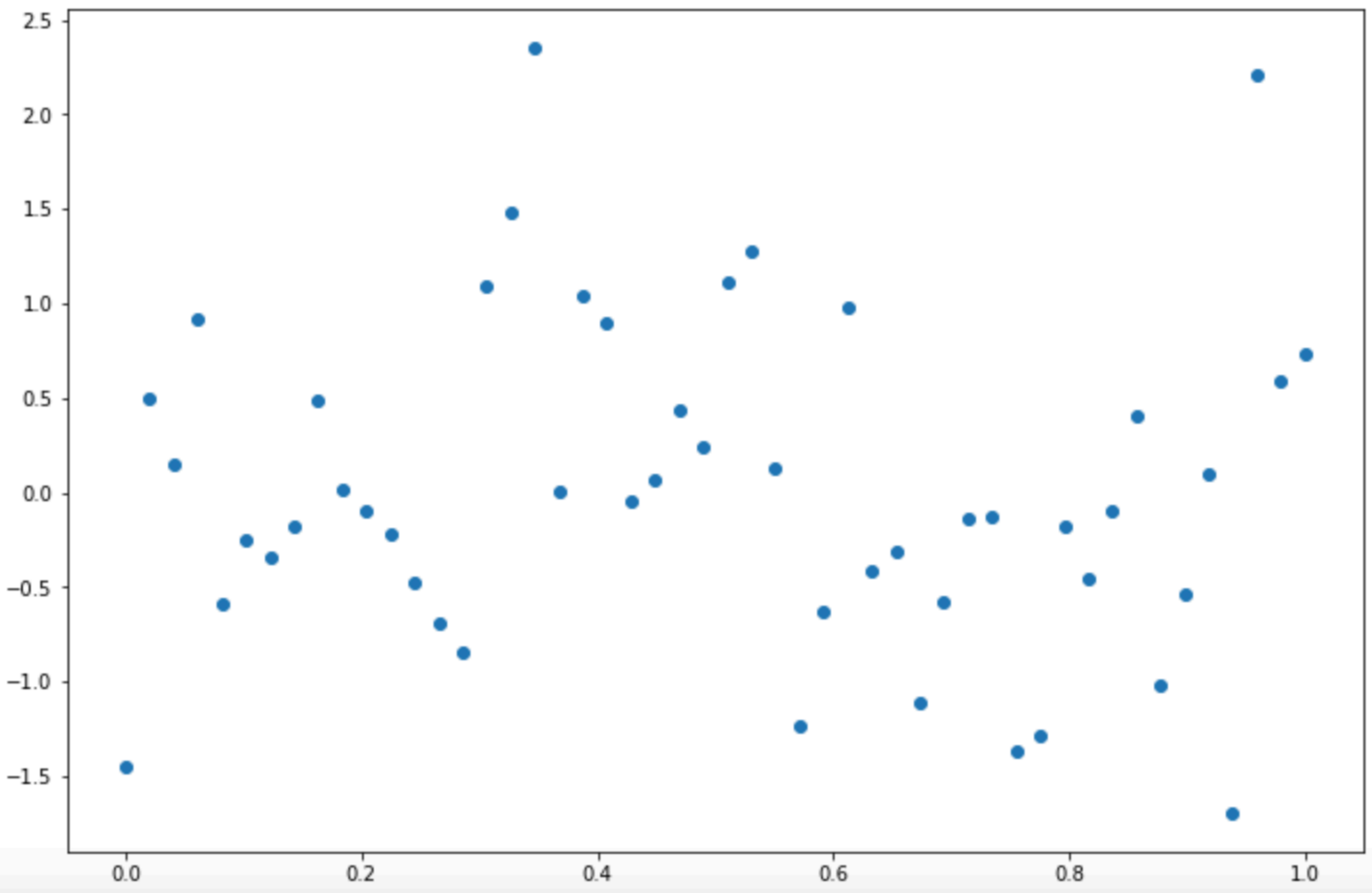
Sampling a function from prior GP

Now let me generate 50 x's in [0,1], and also generate 50 random y's.

```
In [35]: z = np.random.normal(0,1,size=50)

In [36]: plt.scatter(x_input, z)

Out[36]: <matplotlib.collections.PathCollection at 0x7fdc7184e1d0>
```



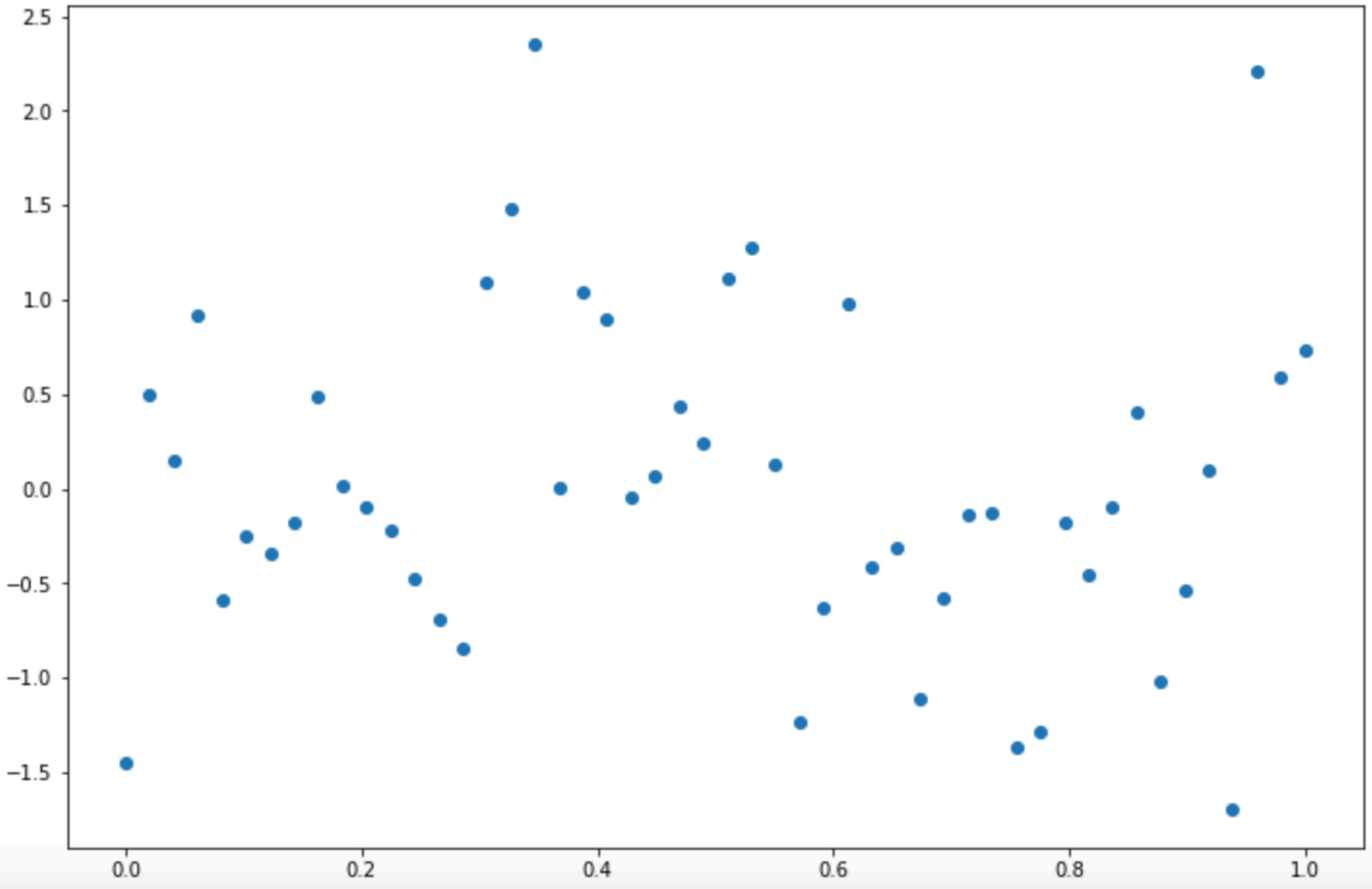
Next, let us connect the dots and make it look like a function

Sampling a function from prior GP

Now let me generate 50 x's in [0,1], and also generate 50 random y's.

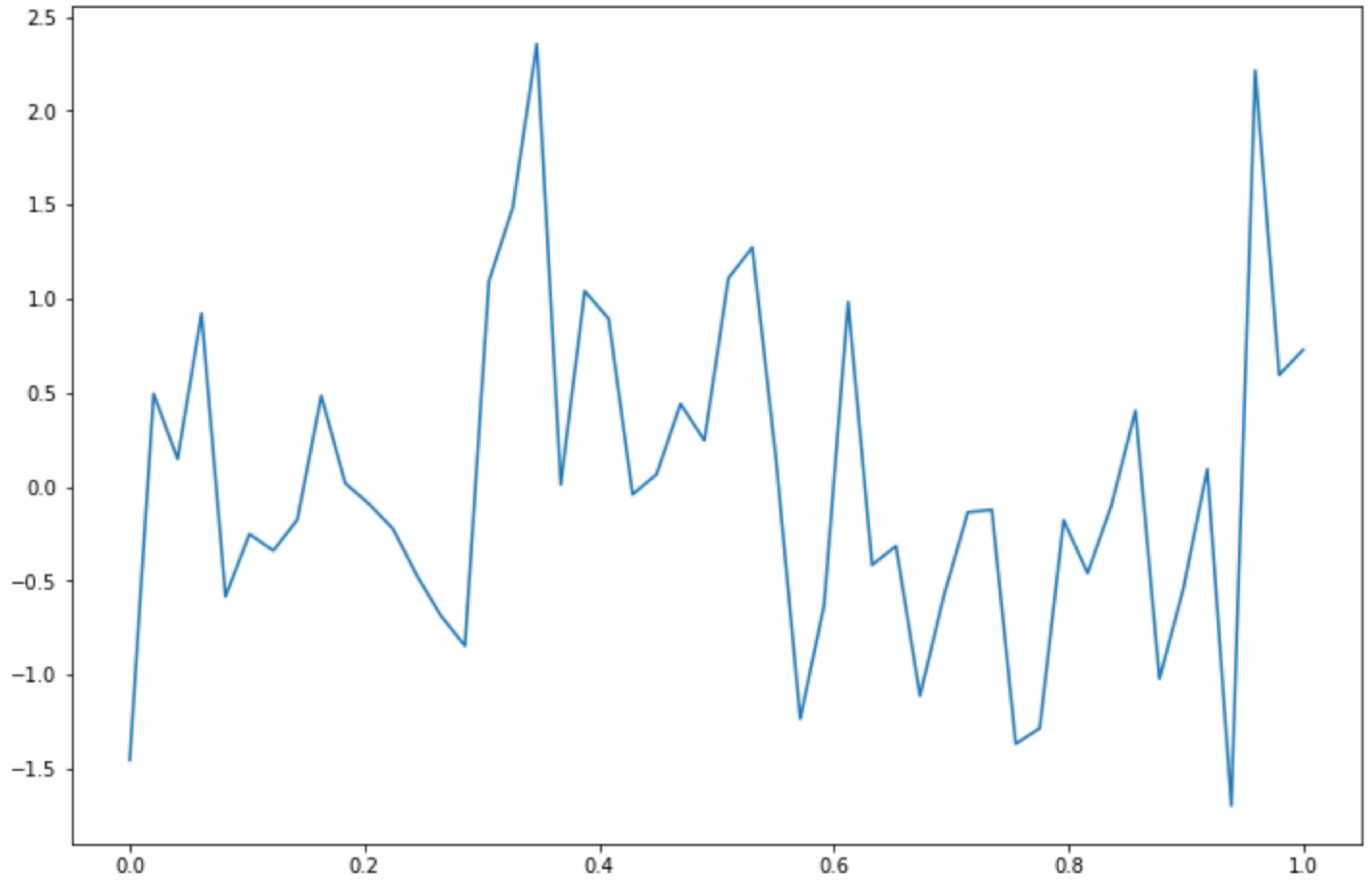
```
In [35]: z = np.random.normal(0,1,size=50)

In [36]: plt.scatter(x_input, z)
Out[36]: <matplotlib.collections.PathCollection at 0x7fdc7184e1d0>
```



Next, let us connect the dots and make it look like a function

```
In [49]: plt.plot(x_input, z)
Out[49]: [<matplotlib.lines.Line2D at 0x7fdc71bf35f8>]
```



Before: x versus z

What do we have:
x[0:50]: evenly separated points along [0,1]
z[0:50]: random numbers from np.normal(0,1,size=50)

After: x versus z1

Before: x versus z

What do we have:
x[0:50]: evenly separated points along [0,1]
z[0:50]: random numbers from np.normal(0,1,size=50)

Now there is a magic matrix K, size = 50x50,
and we are going to take $\sqrt{K}$

After: x versus z1

Before: x versus z

What do we have:
x[0:50]: evenly separated points along [0,1]
z[0:50]: random numbers from np.normal(0,1,size=50)

↓

Now there is a magic matrix K, size = 50x50,
and we are going to take $\sqrt{K}$

↓

Generate a new vector $z_1 = \sqrt{K}z$

After: x versus z1

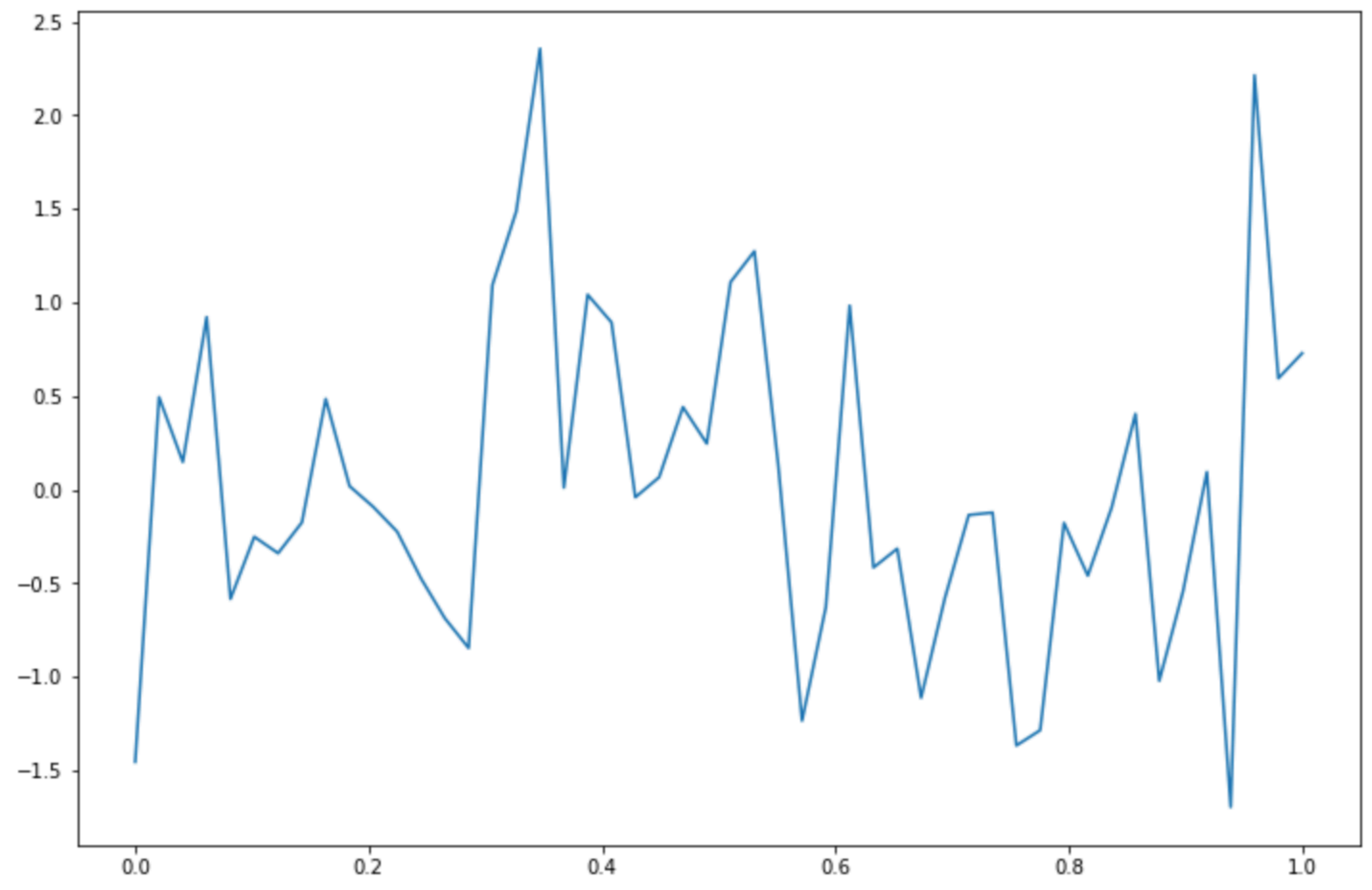
What do we have:
x[0:50]: evenly separated points along [0,1]
z[0:50]: random numbers from np.normal(0,1,size=50)

Now there is a magic matrix K, size = 50x50, and we are going to take $\sqrt{K}$

Generate a new vector $z_1 = \sqrt{K}z$

Before: x versus z

```
In [49]: plt.plot(x_input, z)
Out[49]: [<matplotlib.lines.Line2D at 0x7fdc71bf35f8>]
```



After: x versus z1

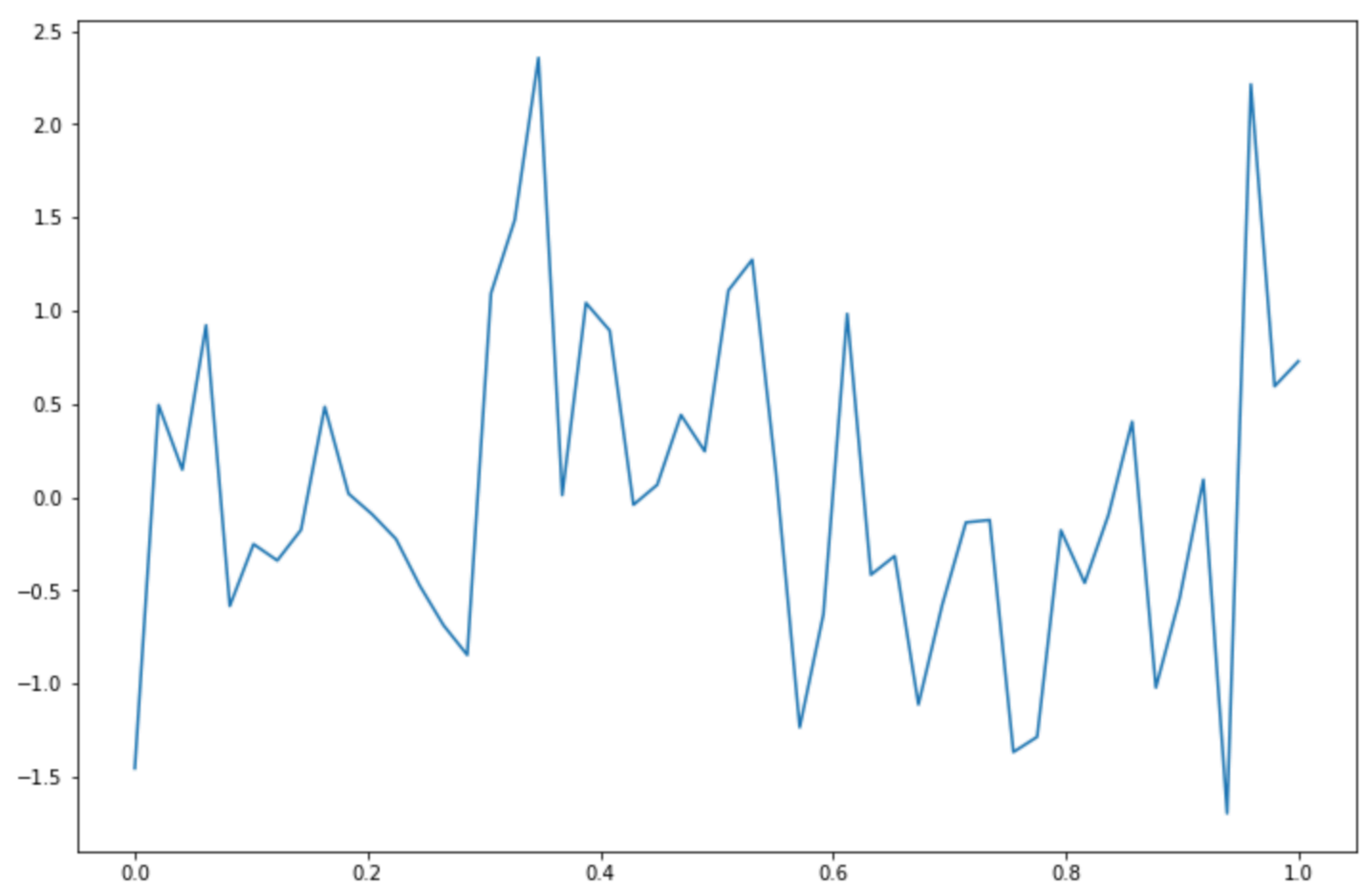
What do we have:
x[0:50]: evenly separated points along [0,1]
z[0:50]: random numbers from np.normal(0,1,size=50)

Now there is a magic matrix K, size = 50x50, and we are going to take $\sqrt{K}$

Generate a new vector $z_1 = \sqrt{K}z$

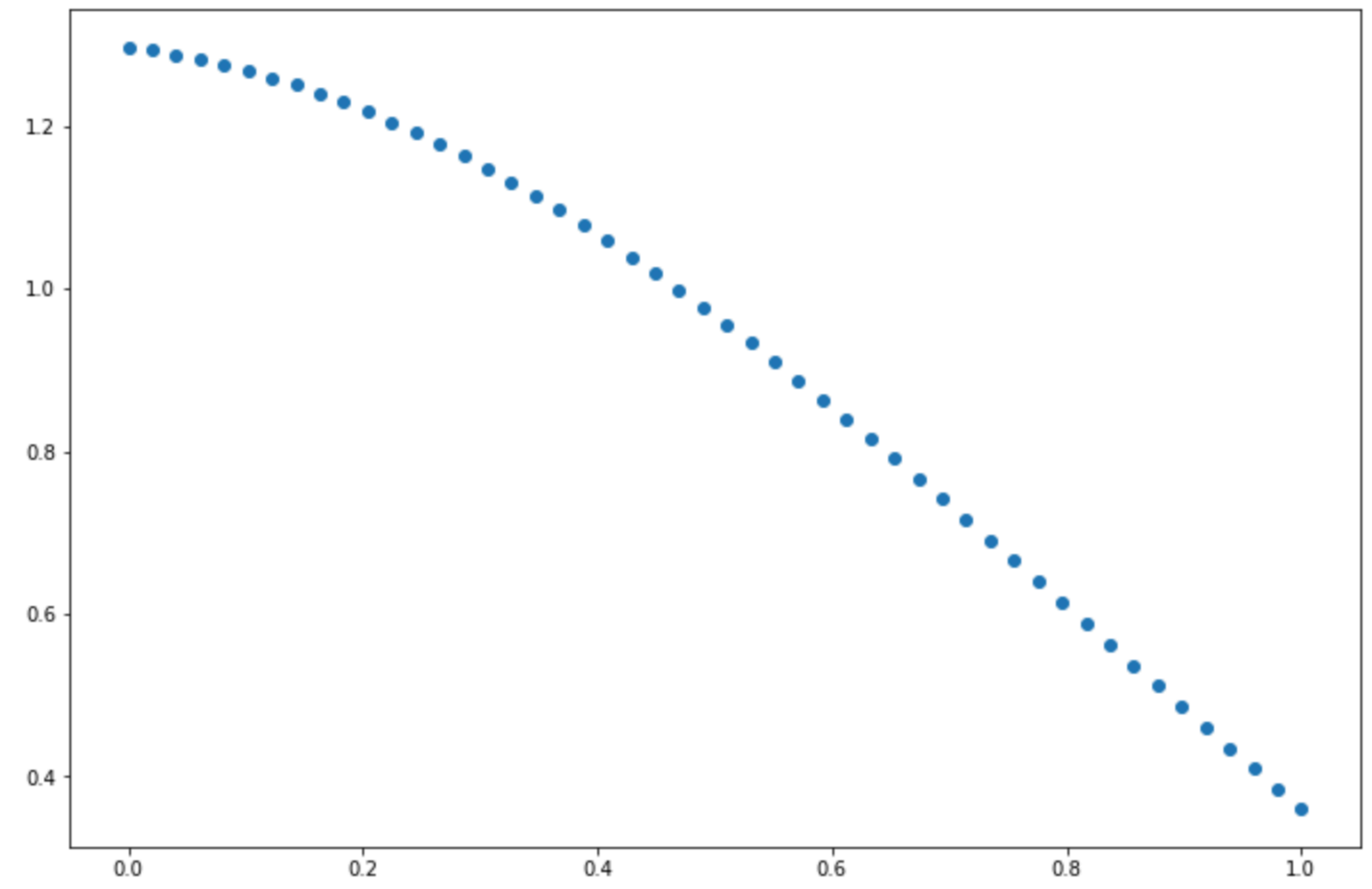
Before: x versus z

```
In [49]: plt.plot(x_input, z)
Out[49]: [<matplotlib.lines.Line2D at 0x7fdc71bf35f8>]
```

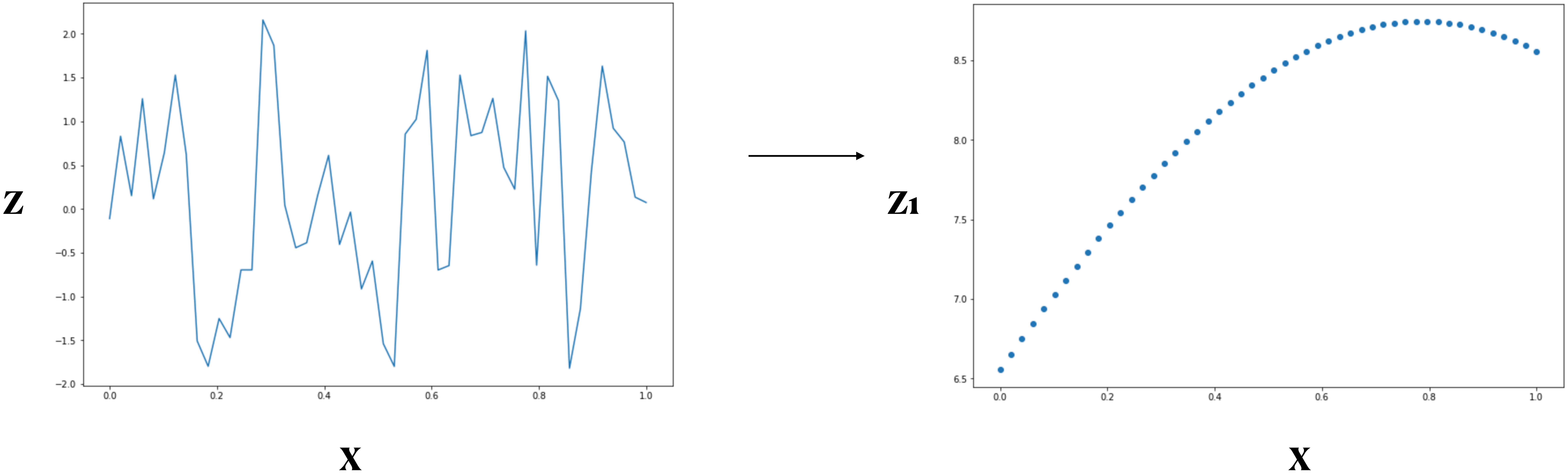


After: x versus z1

```
In [48]: plt.scatter(x_input, z1)
Out[48]: <matplotlib.collections.PathCollection at 0x7fdc71b9c1d0>
```



Do it again!



Z
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.2
0.4
0.6
0.8
1.0
X
Z1
8.5
8.0
7.5
7.0
6.5
0.0
0.2
0.4
0.6
0.8
1.0
X

Final remarks

What you have learnt and what not.

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Many more types: reinforcement learning, semi-supervised learning, etc…

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Many more types: reinforcement learning, semi-supervised learning, etc...

How can one apply machine learning: regression, classification

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Many more types: reinforcement learning, semi-supervised learning, etc…

How can one apply machine learning: regression, classification

More: you can teach computer to play start-craft, trading, generating fake pictures, fake articles, …

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Many more types: reinforcement learning, semi-supervised learning, etc…

How can one apply machine learning: regression, classification

More: you can teach computer to play start-craft, trading, generating fake pictures, fake articles, …

The key question to academia and industry: we can train complex models with large amount of data, but why models learn? (How the computers get the ability of generalization?)

Final remarks

What you have learnt and what not.

Two types of machine learning: supervised (classification, regression) and unsupervised (k-means and PCA)

Many more types: reinforcement learning, semi-supervised learning, etc…

How can one apply machine learning: regression, classification

More: you can teach computer to play start-craft, trading, generating fake pictures, fake articles, …

The key question to academia and industry: we can train complex models with large amount of data, but why models learn? (How the computers get the ability of generalization?)

Regularization: Occam's razor, simpler models preferred.