Table of Contents

Neural Network

```
using Flux
```

Mathematical functions

Let us consider any function, say $y = \sin(x)$.

```
x =
```

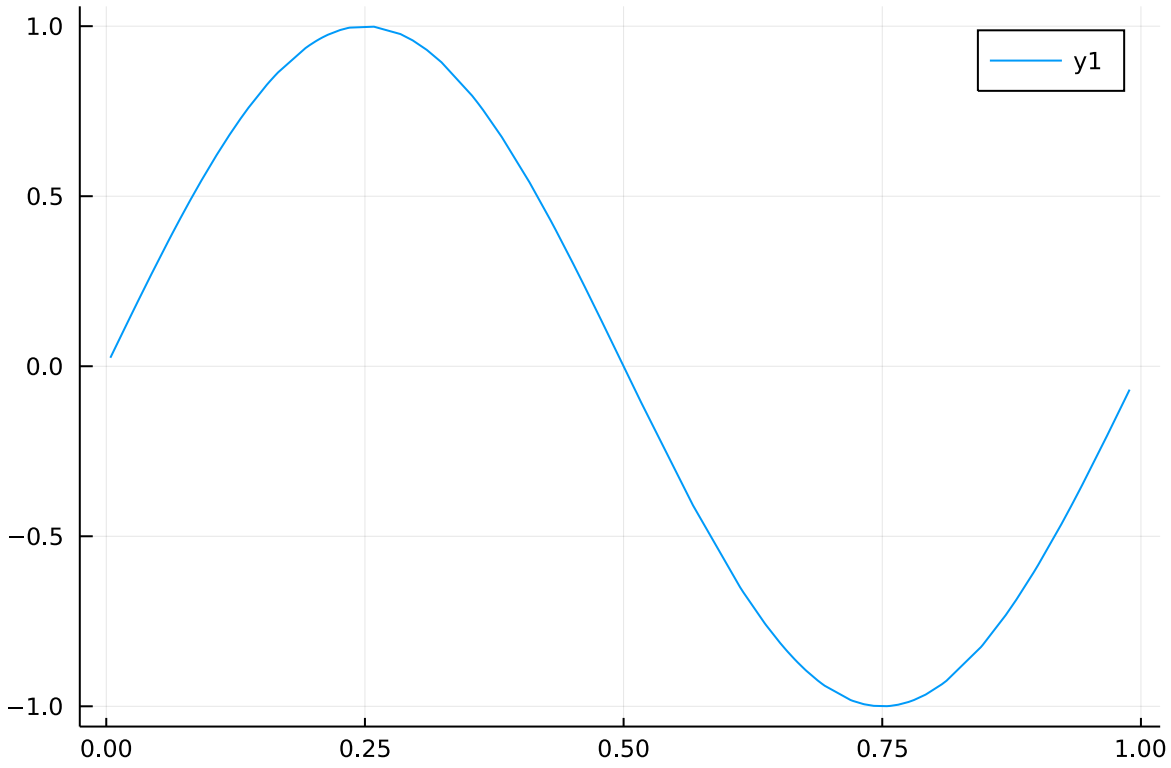
```
Float64[0.00396523, 0.0195239, 0.0254125, 0.0309179, 0.0436088, 0.0598748, 0.0640425,
```

```
x = sort(rand(100))
```

```
y =
```

```
Float64[0.0249117, 0.122365, 0.158994, 0.193043, 0.270586, 0.367393, 0.391619, 0.435
```

```
y = sin.(2π.*x)
```




```
plot(x,y)
```

Neural Network

The network can be defined s a cobination of layers. Each layer in Flux is a function. The chain method is equivalent to composition of functions.

```
NN = Chain(#1, Dense(1, 32, tanh), Dense(32, 32, tanh), Dense(32, 1), first)
```

```
NN = Chain(x -> [x], Dense(1,32,tanh), Dense(32,32,tanh),Dense(32,1),first)
```

```
pred =
  Float64[-0.00082039, -0.00403935, -0.00525761, -0.00639654, -0.00902179, -0.0123861,
```

```
pred = NN.(x)
```

```
Vector{Float64} (alias for Array{Float64, 1})
```

```
typeof(pred)
```

Neural Network as a function approximator

Let's see if the network can approximate the sin function.

```
loss (generic function with 1 method)
```

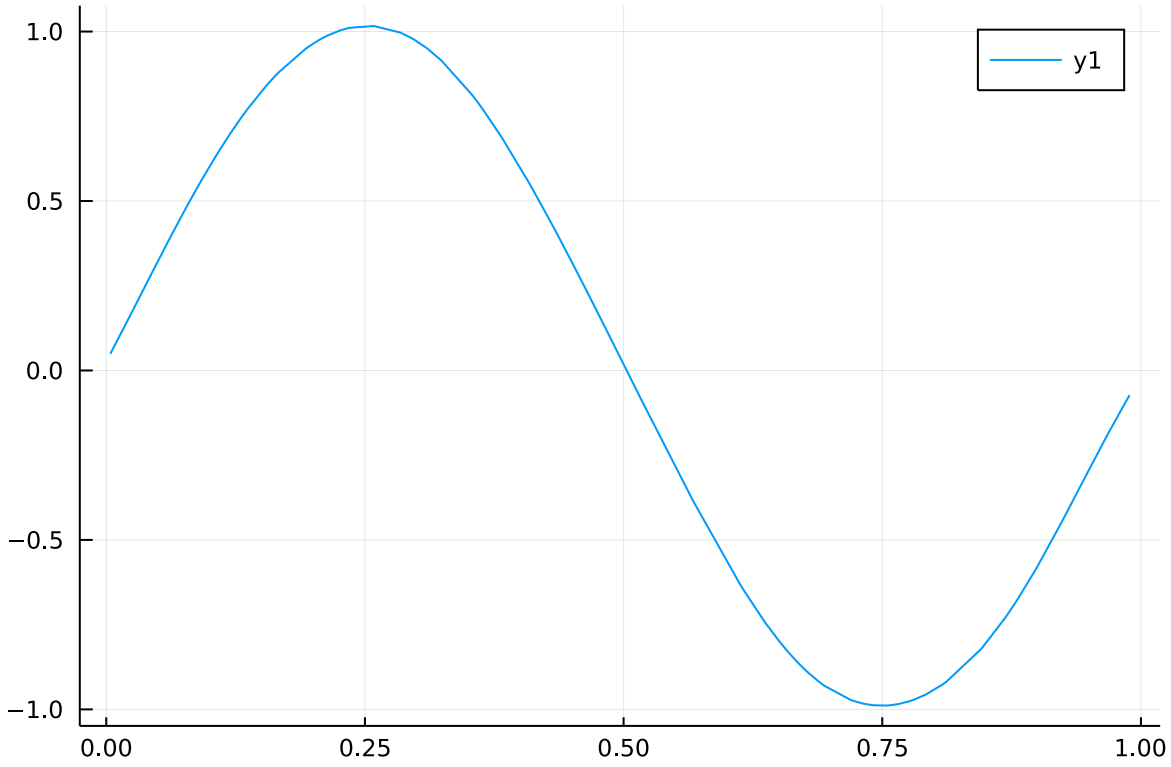
```
loss() = sum(abs2,NN.(x).-y)
```

```
#loss() = Flux.mse(NN.(x),y)
```

```
Flux.train!(loss, params(NN),Iterators.repeated((), 1000) ,ADAM(0.1))
```

```
0.018617643055051 09
```

```
loss()
```

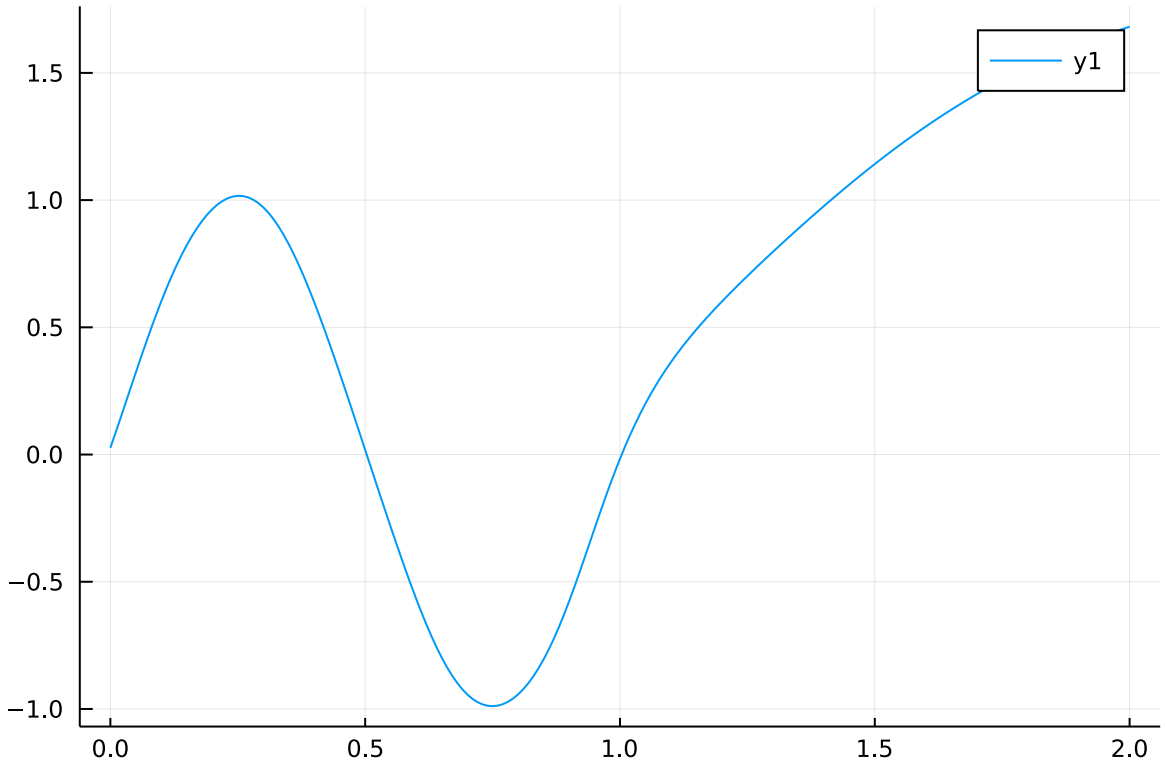



```
plot(x,NN.(x))
```

It will only learn what is taught

```
t = 0.0:0.01:2.0
```

```
t = 0:0.01:2
```




```
plot(t,NN.(t))
```