Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

MCS 472 Lecture 24
Industrial Math & Computation
Jan Verschelde, 9 March 2026

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

learning from experience

A popular quote from computer scientist Tom Mitchell provides as a workable definition for machine learning.

Definition (learning from experience)

A *program can be said to learn* from experience E with respect to

- some class of tasks T and
- performance measure P,

if its performance at tasks in T, as measured by P, improves with experience E.

The quote above is taken from the following book:
Gavin Hackeling: *Mastering Machine Learning with scikit-learn. Apply effective learning algorithms to real-world problems using scikit-learn.* Packt publishing, 2014.

a neural network

Definition (a neuron)

A *neuron* is a nonlinear bounded function

$$y = f(x_1, x_2, \ldots, x_n; w_1, w_2, \ldots, w_p)$$

where

- x_i are the variables, or inputs, $i = 1, 2, \ldots, n$, and
- w_j are the parameters, or weights, $j = 1, 2, \ldots, p$.

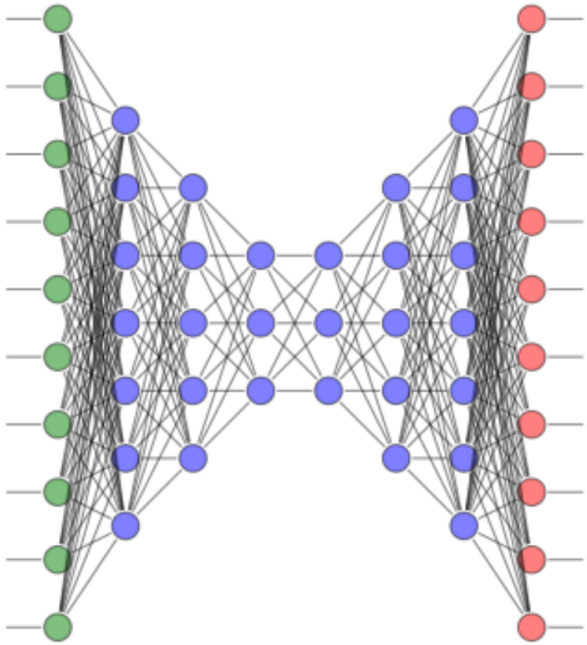
Definition (a neural network)

A *neural network* is the composition of the nonlinear functions of two or more neurons.

deep learning

Definition (a deep neural network)

A *deep neural network* is a neural network with multiple layers.



Picture taken from: *Deep Learning Notes using Julia with Flux*, by Hugh Murrel and Nando de Freitas, `https://HughMurrel.github.io/DeepLearningNotes`, 2019.

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

using `Flux.jl`

Flux is a library for machine learning geared towards high-performance production pipelines.

Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah: *Fashionable Modelling with Flux*, `https://arxiv.org/abs/1811.01457`, 2018.

Let us go through an example, step-by-step:

1. define the task
2. get training data
3. define and initialize the model
4. define the loss function
5. set the optimizer
6. train the model

step 1: the task

Consider a weight matrix W_{true} and bias vector b_{true}:

$$W_{\text{true}} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 \end{bmatrix}, \quad b_{\text{true}} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}.$$

W_{true} and b_{true} are parameters in the function

$$F_{\text{true}}(x) = W_{\text{true}} x + b_{\text{true}}$$

which

- takes on input x, a vector of five numbers, and
- return $y = F(x)$, a vector of two numbers.

Task: recover W_{true} and b_{true} from observed $(x_{\text{train}}, y_{\text{train}})$.

step 2: training data

Let N be the size of the training vectors x_{train} and y_{train}. Then,

- the i-th vector of x_{train} is x_i, and
- the i-th vector of y_{train} is y_i,

with

$$x_i = \begin{bmatrix} 5 + 5u_{1,i} \\ 5 + 5u_{2,i} \\ 5 + 5u_{3,i} \\ 5 + 5u_{4,i} \\ 5 + 5u_{5,i} \end{bmatrix}, \quad z_i = F(x_i), \quad y_i = \begin{bmatrix} z_{1,i} + 0.2v_{1,i} \\ z_{2,i} + 0.2v_{2,i} \end{bmatrix},$$

where i runs from 1 to N, and

- $u_{j,i}$ are chosen at random, uniformly from $[0, 1]$.
- $v_{j,i}$ are random, standard normally distributed numbers.

step 3: defining and initializing the model

The model is

$$M(x) = Wx + b, \quad W \in \mathbb{R}^{2\times 5}, \quad b \in \mathbb{R}^2.$$

Initialize $M(x)$ with

1. a random 2-by-5 matrix for W, and
2. a random 2-vector for b.

steps 4, 5, and 6

step 4: the loss function

The performance is measured by a loss function:

$$\text{loss}(x, y) = \sum_{i=1}^{2} \left(y_i - \widehat{y}_i \right)^2, \quad y = F(x), \quad \widehat{y} = M(x).$$

step 5: set an optimizer

We choose the classic gradient descent as the optimizer with learning rate $\eta = 0.01$.

step 6: train the model

```
Flux.train!(loss, params(W, b), train_data, opt)
```

train the model — the longer way

```
opt = Descent(0.01)
train_data = zip(x_train, y_train)
ps = Flux.params(W, b)

for (x, y) in train_data
    gs = gradient(ps) do
        loss(x,y)
    end
    Flux.Optimise.update!(opt, ps, gs)
end
```

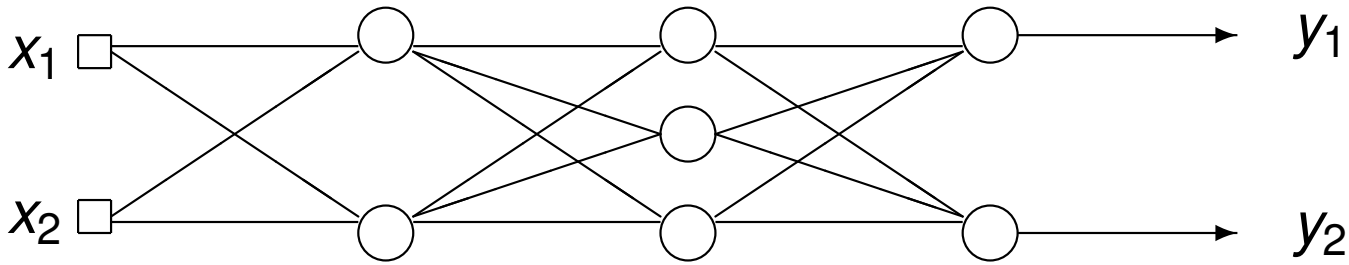
The main benefit of this lower way to train the model is that we can monitor the progress of the loss function.

Exercise 1: Make a plot of the loss function for all N steps. Does N really have to be 10,000 for sufficient accuracy?

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

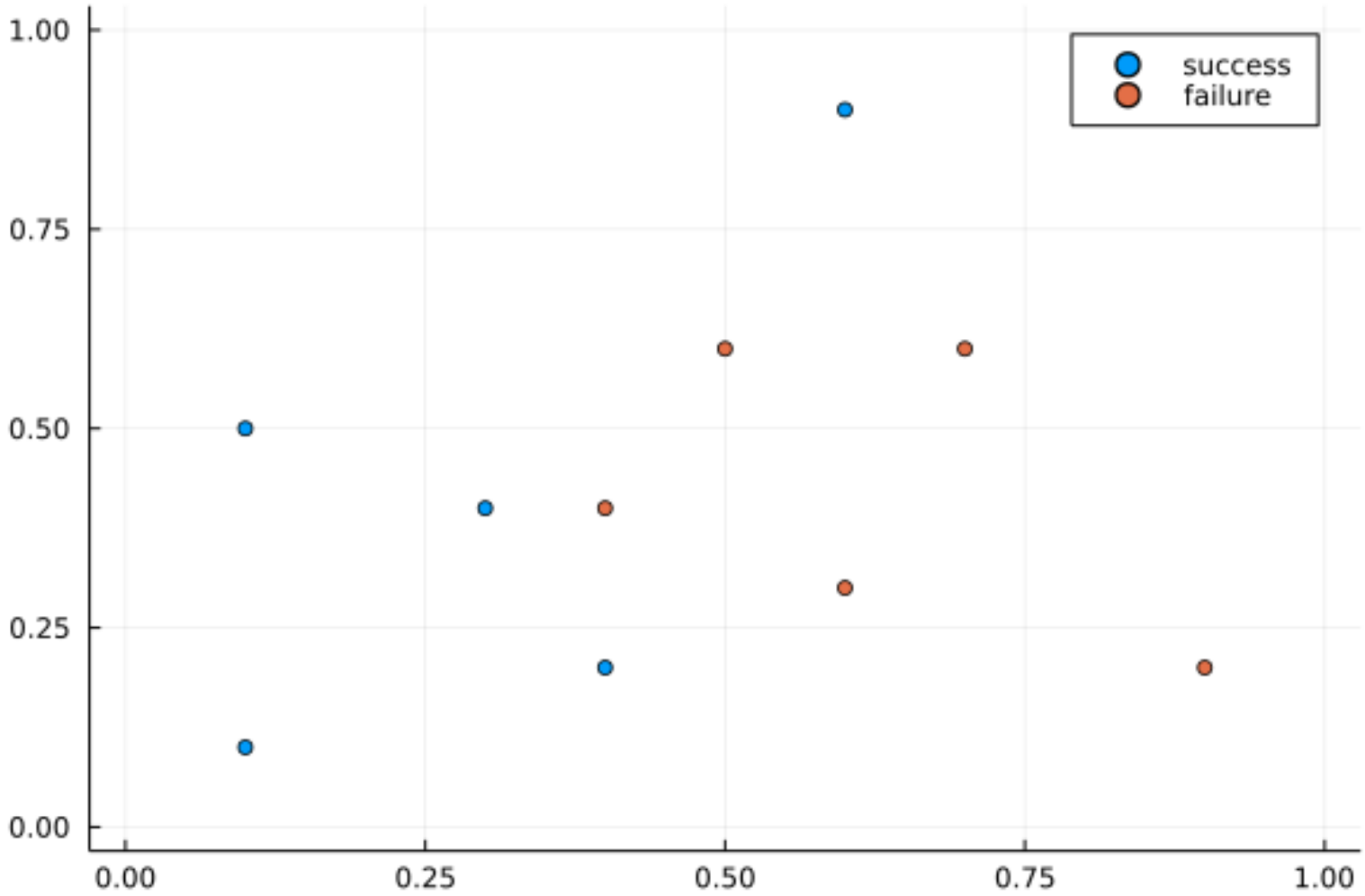
an artificial neural network



Let us redo the example of the previous lecture, taken from

- Catherine F. Higham and Desmond J. Higham:
  Deep Learning: An Introduction for Applied Mathematicians.
  SIAM Review, Vol. 61, No. 4, pages 860–891, 2019.

a collection of labeled points



the points and the labels

The coordinates of the points:

```
x1 = [0.1,0.3,0.1,0.6,0.4,0.6,0.5,0.9,0.4,0.7]
x2 = [0.1,0.4,0.5,0.9,0.2,0.3,0.6,0.2,0.4,0.6]
```

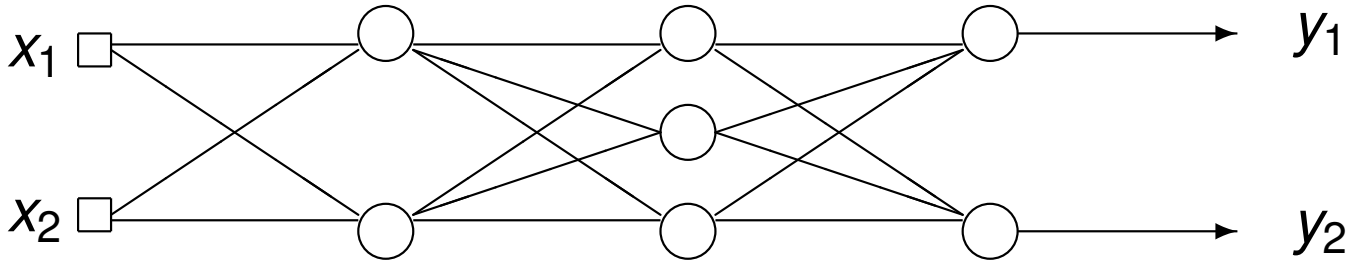
and their labels

```
ylabels = [ones(1,5) zeros(1,5); zeros(1,5) ones(1,5)]

2×10 Matrix{Float64}:
  1.0  1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0
  0.0  0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  1.0
```

Task: Find a function F, so $y = F(z_1, z_2)$, for a point (z_1, z_2).

defining and initializing the model



The weights and the bias vectors are

$$\texttt{W2} \in \mathbb{R}^{2\times 2}, \texttt{W3} \in \mathbb{R}^{3\times 2}, \texttt{W4} \in \mathbb{R}^{2\times 3}, \texttt{b2} \in \mathbb{R}^{2}, \texttt{b3} \in \mathbb{R}^{3}, \texttt{b4} \in \mathbb{R}^{2}.$$

With `Flux.jl`, we define the model as

```
L2 = Dense(W2, b2, sigmoid)
L3 = Dense(W3, b3, sigmoid)
L4 = Dense(W4, b4, sigmoid)
M = Chain(L2, L3, L4)
```

the model defined and initialized

```
L2 = Dense(W2, b2, sigmoid)
```

makes one layer with weights `W2`, bias `b2`, and the sigmoid function.

To collect multiple layers into one network:

```
M = Chain(L2, L3, L4)
```

The output is

```
Chain(
  Dense(2 => 2, \sigma),    # 6 parameters
  Dense(2 => 3, \sigma),    # 9 parameters
  Dense(3 => 2, \sigma),    # 8 parameters
)     # Total: 6 arrays, 23 parameters, 568 bytes.
```

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

defining the loss function

The loss function is

$$\text{loss}\left(\texttt{W2,W3,W4,b2,b3,b4}\right) = \frac{1}{10}\sum_{i=1}^{10}\frac{1}{2}\left\| y\left(x_1^{(i)}, x_2^{(i)}\right) - M\left(x_1^{(i)}, x_2^{(i)}\right)\right\|_2^2,$$

where M is the model.

```
function loss(x, y)
    result = 0.0
    for i=1:10
        point = [x1[i]; x2[i]]
        yy = M(point)
        result = result + 0.5*((yy[1] - ylabels[1,i])^2
                               + (yy[2] - ylabels[2,i])^2)
    end
    return result/10.0
end
```

defining the training data and the optimizer

Take one million data points:

```
N = 1_000_000
xtrain = [randn(2) for _ in 1:N];
ytrain = [randn(2) for _ in 1:N];
```

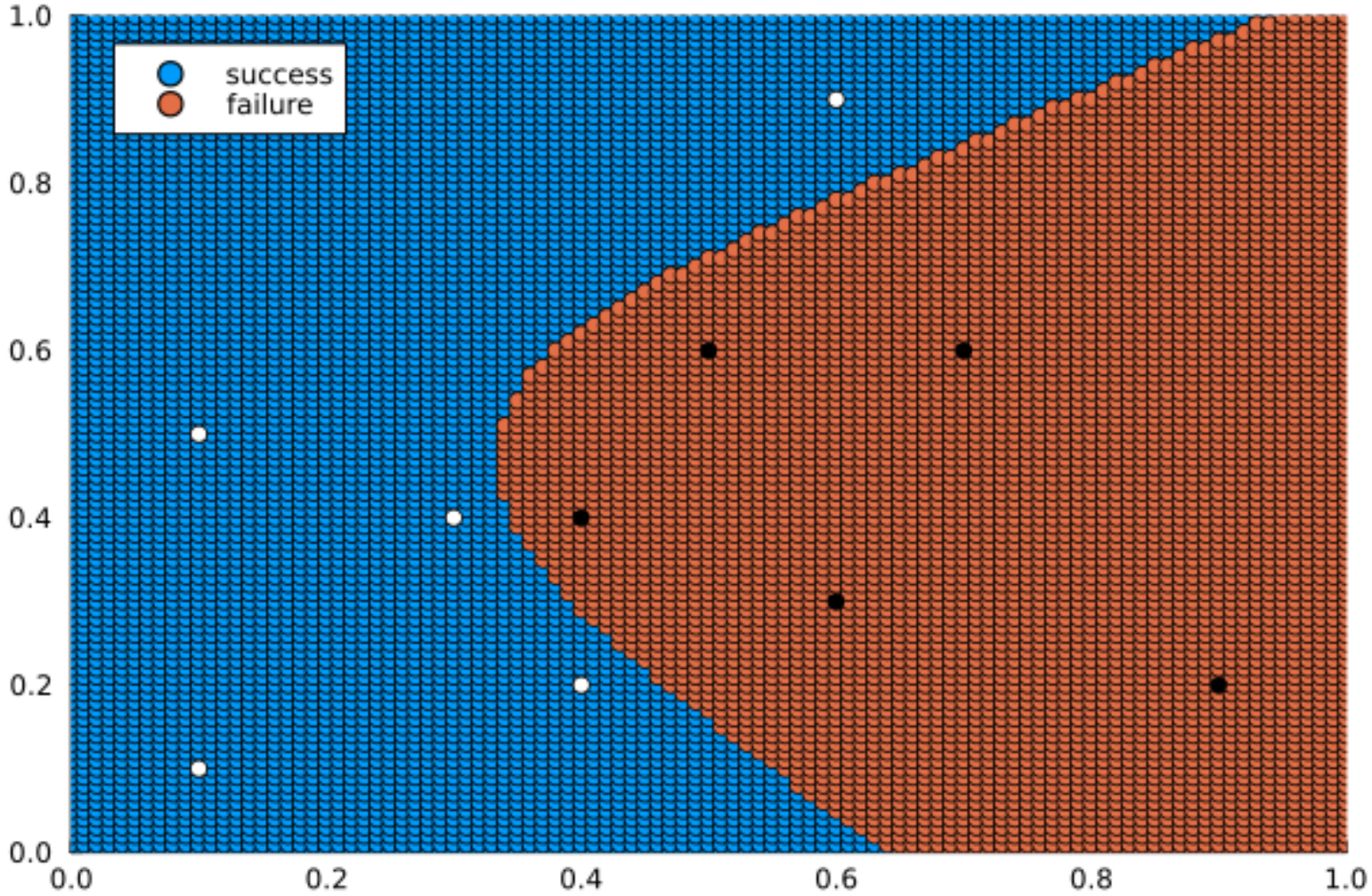
Define the optimizer and parameters:

```
train_data = zip(xtrain, ytrain)
opt = Descent(0.01)
ps = Flux.params(M)
```

Then the training of the model:

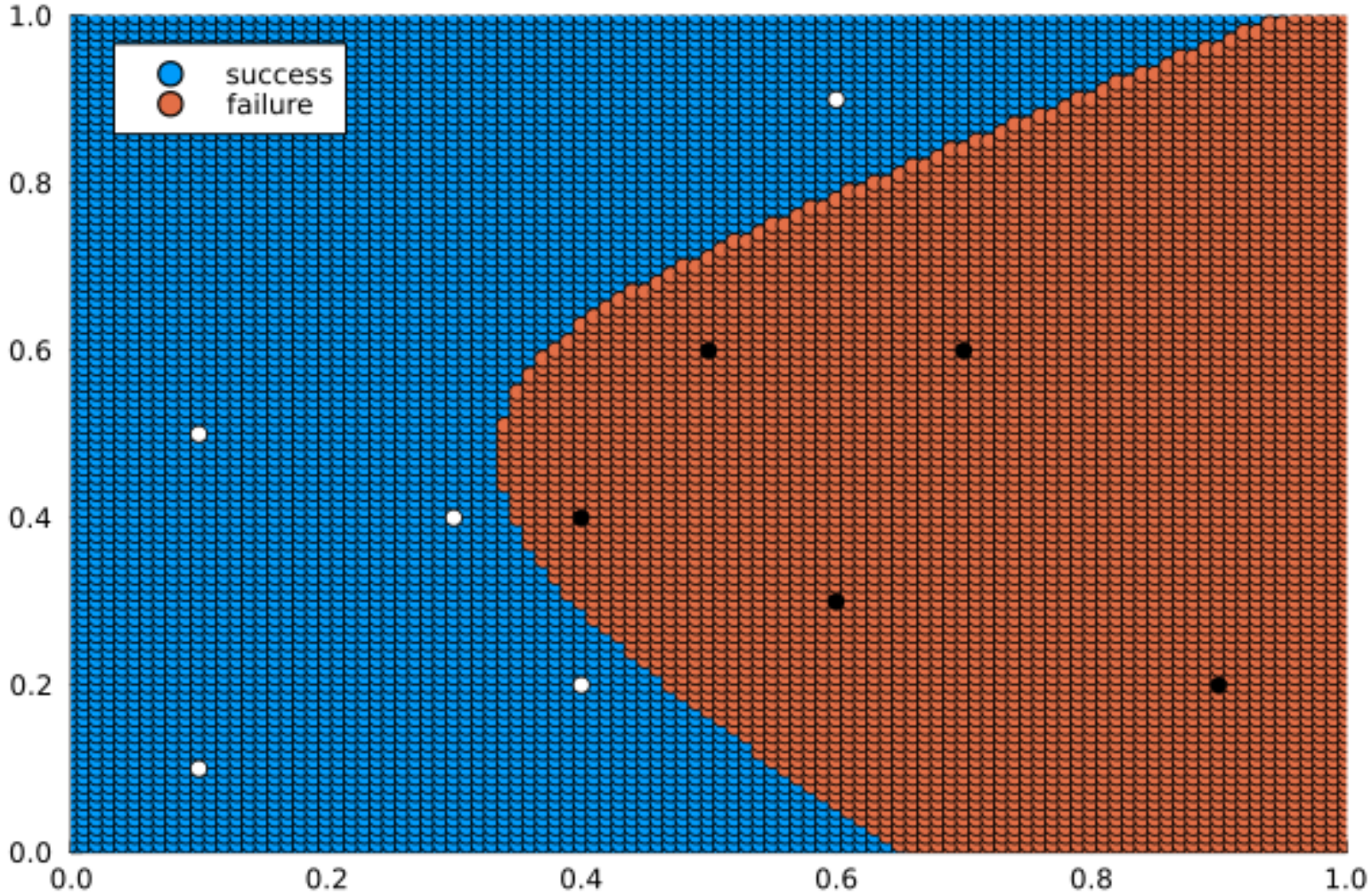
```
Flux.train!(loss, ps, train_data, opt)
```

evaluating $F(x_1, x_2)$ over $[0, 1] \times [0, 1]$, previous lecture



The white and black dots are the ten given points.

evaluating $M(x_1, x_2)$ over $[0, 1] \times [0, 1]$, with Flux



The white and black dots are the ten given points.

two questions

First, look at the posted Jupyter notebook.

Exercise 2: Do we really need one million data points?
Plot the evolution of the loss function for all steps.

Exercise 3: What if we use `rand` instead of `randn`
in the definition of the `xtrain` and `ytrain` vectors?

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

the MNIST database of handwritten digits

MNIST is a classic image classification dataset with 70,000 images.

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner:
  Gradient-based learning applied to document recognition.
  Proceedings of the IEEE, 86(11):2278-2324, November 1998

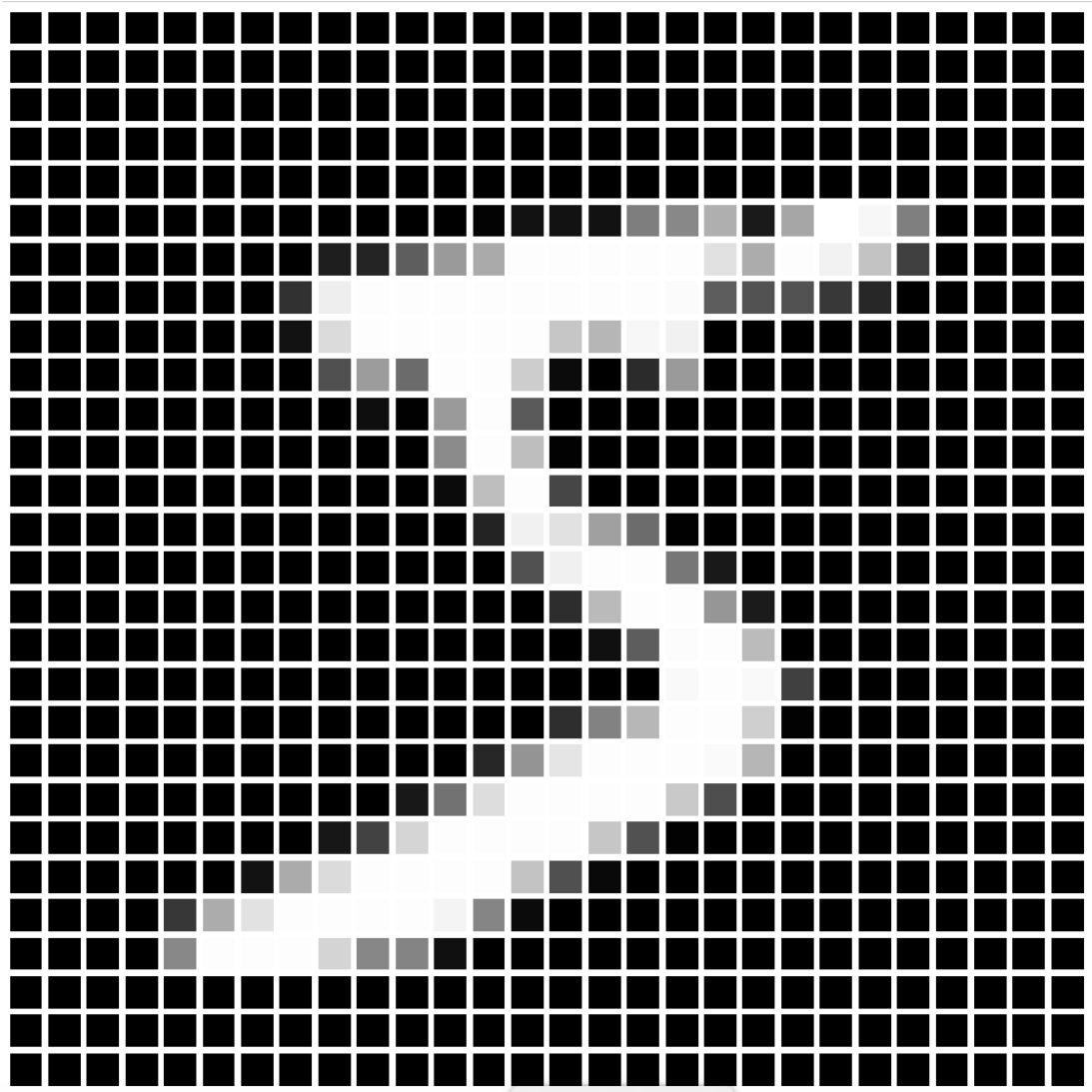
The website: `http://yann.lecun.com/exdb/mnist`.

```
julia> using MLDatasets: MNIST

julia> MNIST.download()
```

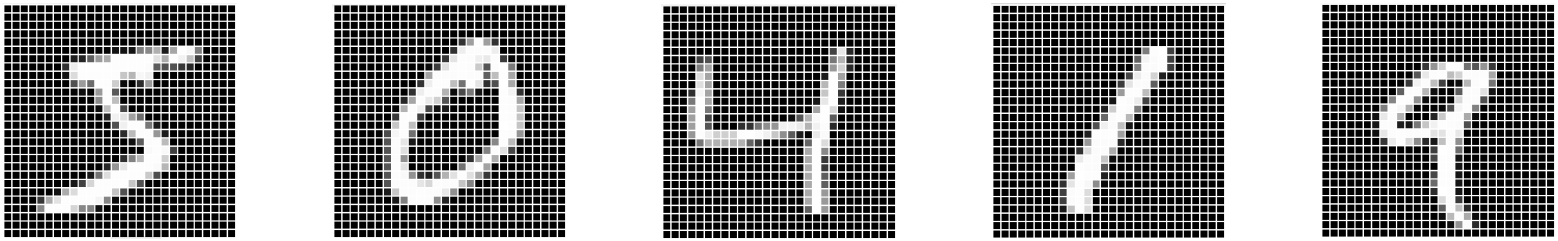
The preparation of this lecture benefited from the post at
`https://machinelearninggeek.com/mnist-with-julia`.

the first image in the MNIST data set



the first five images

The first five images



are labeled as 5, 0, 4, 1, and 9.

- The *i*-th image is available as
  `MNIST(split=:train).features[:,:,i].`
- Its corresponding label is in
  `MNIST(split:=train).targets[i].`

Introduction to Deep Learning

1. Getting Started with Flux
   - what is deep learning?
   - a first example
2. An Artificial Neural Network
   - defining the model
   - training the network
3. Recognizing Handwritten Numbers
   - the MNIST data
   - training the network

step 1: representing the labels as vectors

```
using Flux: onehotbatch

labels = onehotbatch(MNIST(split=:train).targets, 0:9)
```

With `onehotbatch`, each trainlabel is converted to a vector,

As `MNIST(split=:train).targets[i]` returns an integer in the range from 0 to 9, the vector that corresponds to the label will have length 10.

The vector will be zero, except for a one at the entry the corresponds to the value of the label.

For example, the number 5 corresponds to the vector

$$[0,0,0,0,0,1,0,0,0,0]^T.$$

step 2: representing the images as vectors

The images are 28-by-28 matrices of `UInt8` types.

The code below takes the first 100 images,
reshapes the matrix into a vector and converts the numbers.

```
NBR = 100
images = Matrix{Float32}(zeros(28*28, NBR))
for i=1:size(images,2)
    image = MNIST(split=:train).features[:,:,i]
    imagereshaped = reshape(image,:)
    imagenumbers = [Float32(x) for x in imagereshaped]
    for j=1:size(images,1)
        images[j, i] = imagenumbers[j]
    end
end
```

step 3: defining the model

```
using Flux

model = Chain(Dense(28*28, 40, relu), Dense(40, 10),
              softmax)
```

where

- `relu` stands for rectified linear output, and
- `softmax` formulates a logistic regression.

The output is

```
Chain(
  Dense(784 => 40, relu), # 31_400 parameters
  Dense(40 => 10),        # 410 parameters
  NNlib.softmax,
)  # Total: 4 arrays, 31_810 parameters, 124.508 KiB.
```

Test the evaluation at the 10-th image: `model(images[:,10])`.

steps 4 and 5: the loss function and optimizer

4. In classifications with multiple classes,
   where the labels are given in a one-hot format, we use:

```
using Flux: crossentropy

loss(X, y) = crossentropy(model(X), y)
```

5. The optimizer is set to one of the gradient descent methods:

```
opt = Adam()
```

`Adam` is a variation of Adaptive Gradient Descent
where the components of the gradient are weighted.

step 6: training the model

To monitor the progress during the training,
we define a callback function:

```
progress = () -> @show(loss(images, labels[:,1:NBR]))
```

Then the training happens as follows:

```
using Flux: throttle
for epoch in 1:100
    Flux.train!(loss, Flux.params(model),
        [(images,labels[:,1:NBR])], opt,
        cb = throttle(progress, 10))
```

The loss is reported every 10 seconds.

One `epoch` loops over the data only once.
In the training, we limited the number of epochs to 100.

verification

To verify the model, we evaluate an image in the model, and then check for the index of the largest value:

```
M1 = model(images[:,1])
println("output of the model :\n", M1)
println("the number is ", argmax(M1)-1)
```

The first five image are classified correctly.

To check a random image from the training data:

```
idx = rand((1:NBR),1)
Midx = model(images[:,idx[1]])
println("output of the model :\n", Midx)
println("the number is ", argmax(Midx)-1)
println("label : ", MNIST(split=:train).targets[idx])
```

Exercise 4: Verify a random image not in the training data.

See the posted Jupyter notebook.

summary and references

We trained a network to recognize handwritten numbers.

We did this with `Flux.jl`, described in

- Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah: *Fashionable Modelling with Flux*, `https://arxiv.org/abs/1811.01457`, 2018.

and applied it to an artificial neural network used in

- Catherine F. Higham and Desmond J. Higham: Deep Learning: An Introduction for Applied Mathematicians. *SIAM Review*, Vol. 61, No. 4, pages 860–891, 2019.

A more detailed description of sigmoids starts in Chapter 7 of

- Hugh Murrel and Nando de Freitas: *Deep Learning Notes using Julia with Flux*, `https://HughMurrel.github.io/DeepLearningNotes`, 2019.

1. machine learning with Flux

proposal for a topic of a project

Michael Innes, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral Shah: *Fashionable Modelling with Flux*, `https://arxiv.org/abs/1811.01457`, 2018.

One software that learns from experience is Flux.

1. Read the software documentation.
2. Describe how it fits in the computational ecosystem of Julia.
3. Illustrate the capabilities by a good use case.
   Do not use MNIST, but a similar good data set.

What is scientific machine learning?