A practical introduction to Deep Learning

BIOF509 – 04/12/18

Nicolas Fiorini

Machine learning

Note: please ask questions!

What is machine learning?

Essentially, learning a model

What is a model?

A set of parameters such that when you provide an input, it processes it and gives a prediction

Deep learning

So what is new in deep learning?

Not much. You will see approaches similar to those used in machine learning.

The main difference is in the number of trained parameters;

And most of them are **not manually designed.** For deep learning, these parameters are represented by neurons.

An intuitive example

Identify people's last names in text.

Machine learning – for each word in the text, have a gold standard and the features:

- Is the previous word "Mr", "Ms", "Mr.", … E.g. Mr. Doe
- Does it start with a capital letter? E.g. [...] Doe
- Is the previous word capitalized too? E.g. John Doe

Deep Learning – for each word in the text, have a gold standard and pass it to the network. With an appropriate network, it will design the features itself to fit the gold standard.

A vast world

There are tons of learning methods/tasks and this course is not about covering all of them.

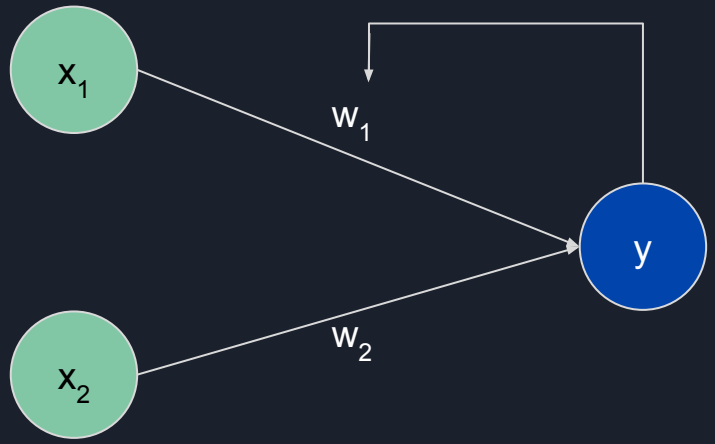
Instead, I will try to give an intuitive introduction to how it works.

Particularly, we will focus on

- Feedforward Neural Networks
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)

(Almost) single layer perceptron

Inputs		Output
0	0	0
1	1	1
1	0	1
0	1	0



Weights set randomly

$y = 1$ if $\sum x_i w_i > 1$;
$y = 0$ otherwise

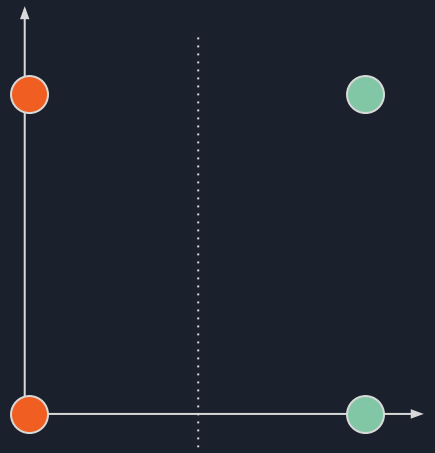
Input layer

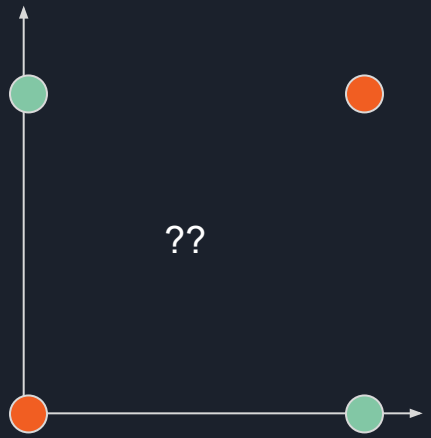
Output layer

Ideas?

$y = 1$ if $\sum x_i w_i > 1$;
$y = 0$ otherwise

We need a **nonlinear activation function**



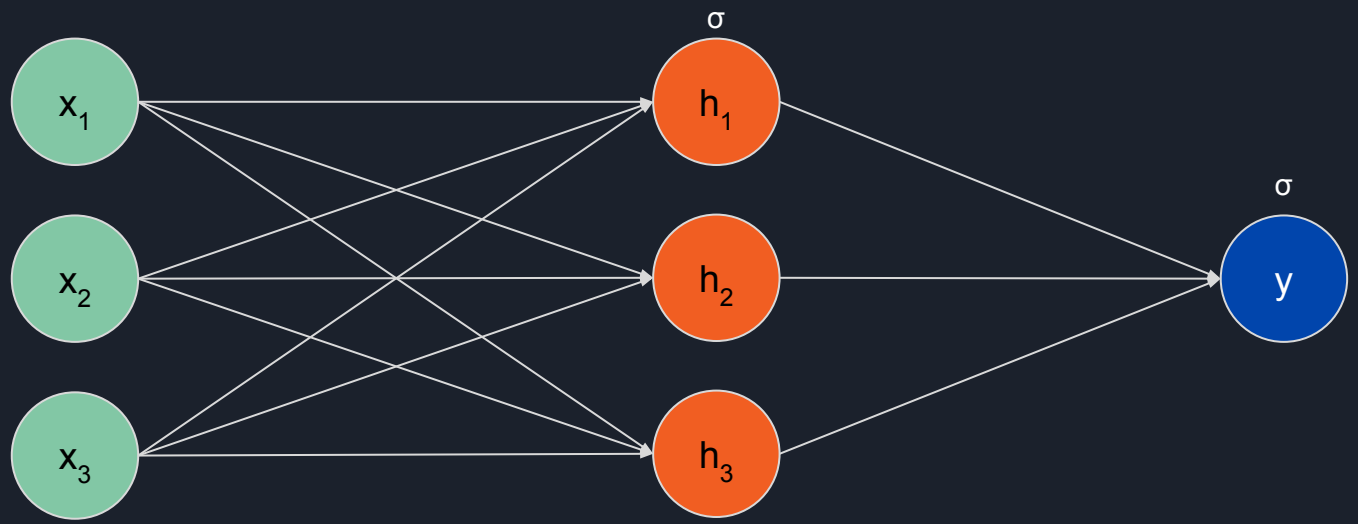


Inputs		Output
0	0	0
1	1	1
1	0	1
0	1	0

Inputs		Output
0	0	0
0	1	1
1	0	1
1	1	0

Multiple layer perceptron

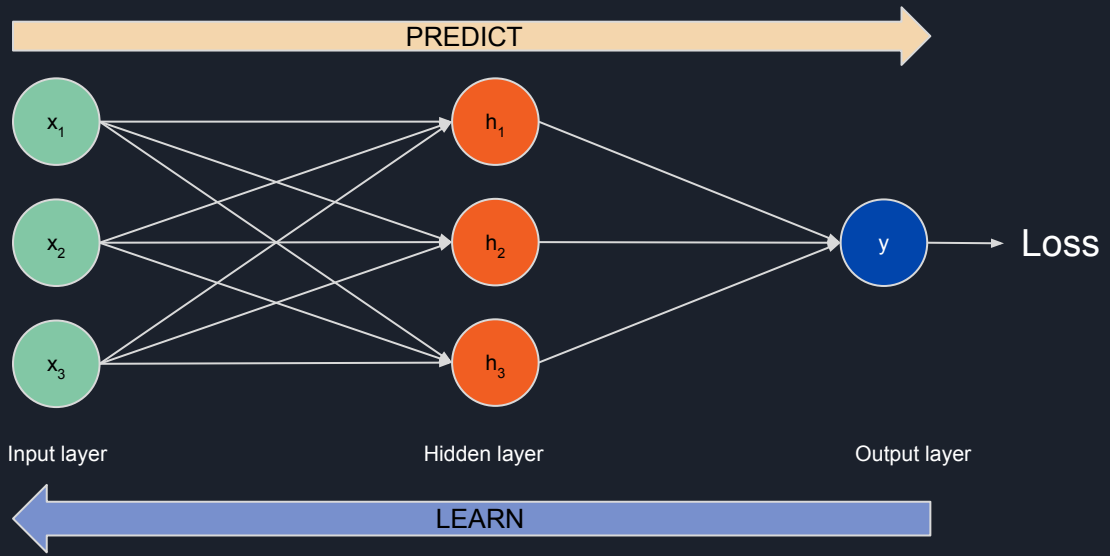
More parameters = more modelling freedom
Non-linear activation function (e.g. sigmoid) gives sense to multilayer (deep) networks



When do we learn?

Two critical concepts

- **Loss function**: a way to evaluate how the network currently performs
- **Backpropagation**: updating the weights based on the current loss value



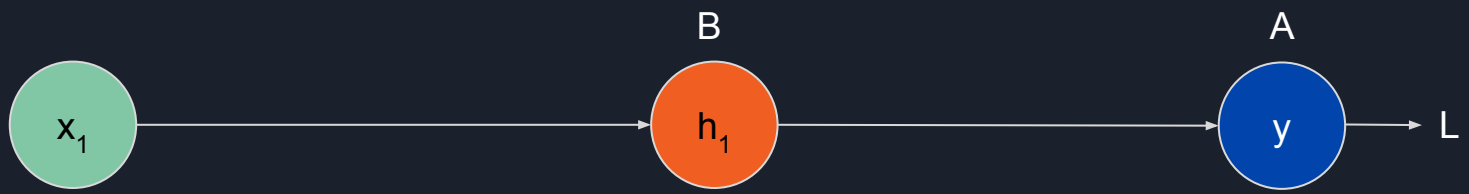
Loss and backpropagation

- Many loss functions - you can design your own to fit a particular problem
- Backpropagation mainly relies on the chain rule:

Loss = L(A(B((WX)))

A and B are activation functions. W is the weight matrix, X is the input matrix.

We can recursively derive the cost for each cell and update their weights (e.g. with gradient descent)



Let’s code it!

Open the notebook.

Convolutional Neural Networks

- Huge domain
- Inspired from human vision
  - Neuron response to horizontal edges, or vertical edges – **features**
- Gained a lot of popularity for its efficiency on image processing

Applications:

- Handwriting recognition
- Face recognition
- Tumor detection
- Even sequencing and alignments based on colored pictures of DNA sequences

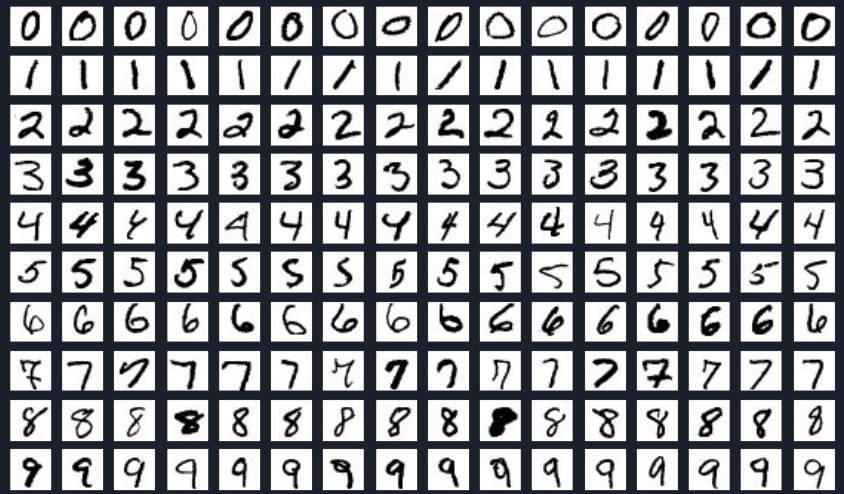
Handwritten digit recognition

MNIST: standard dataset – all novelties are compared on this dataset

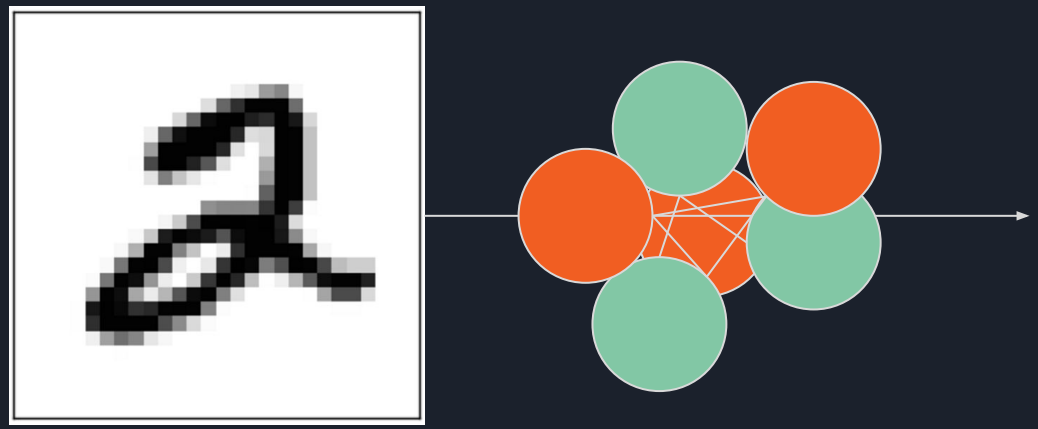
10 classes

Best accuracy in literature: 99.79%

Perfect example where CNNs work well



Desired pipeline



Probability distribution:

```
[
  [ 0 ],
  [ 0 ],
  [ 1 ],
  [ 0 ],
  [ 0 ],
  [ 0 ],
  [ 0 ],
  [ 0 ],
  [ 0 ],
  [ 0 ]
]
```

What is an image?

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Let’s use filters

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	1	0
0	1	0
0	1	0

Detects vertical lines

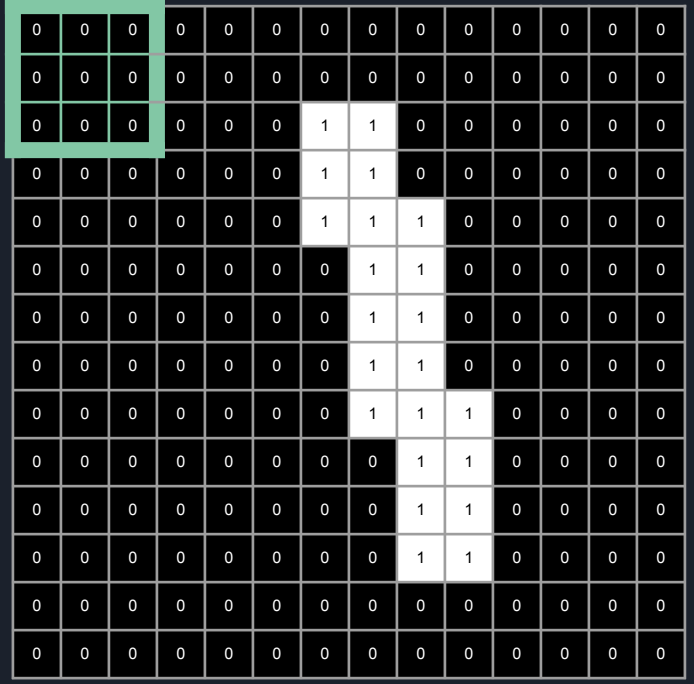
0	0	0
1	1	1
0	0	0

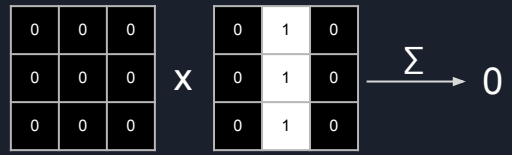
Detects horizontal lines

1	0	0
0	1	0
0	0	1

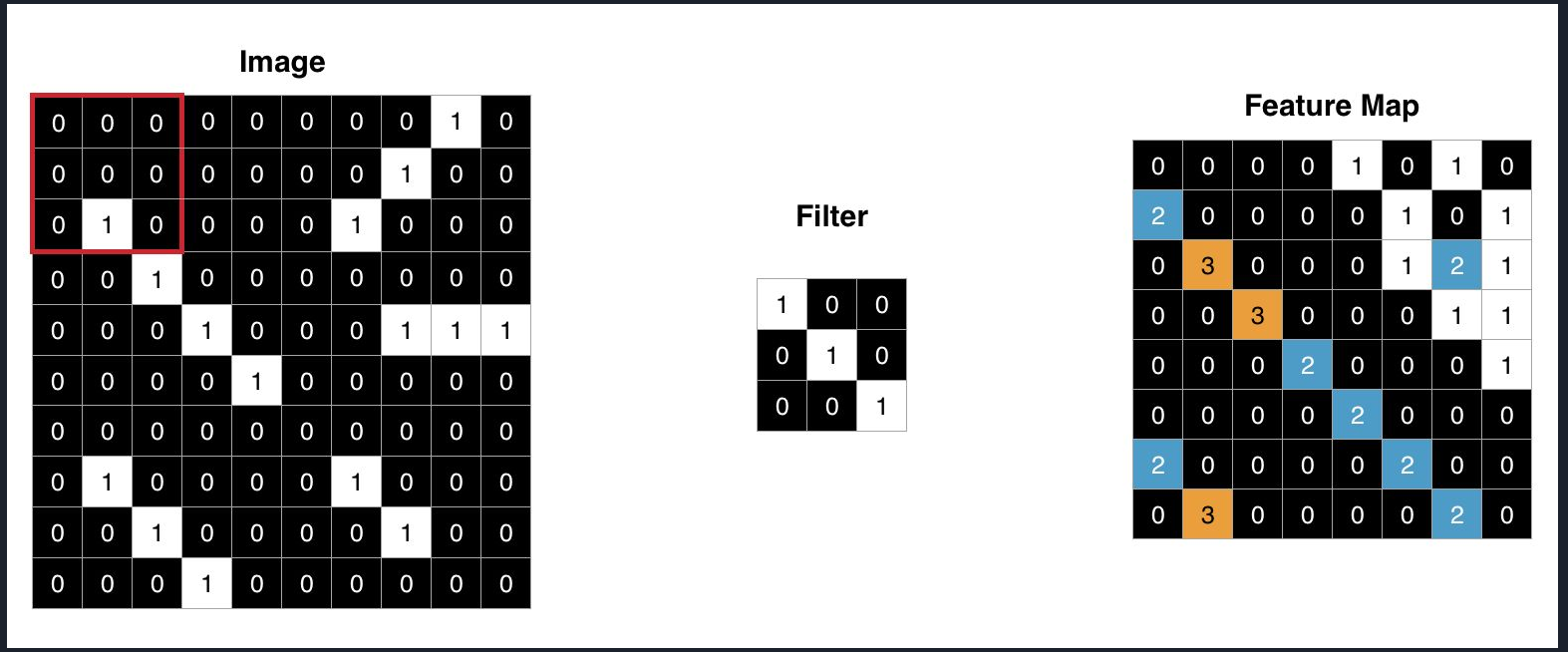
Detects diagonals

And convolve





And convolve

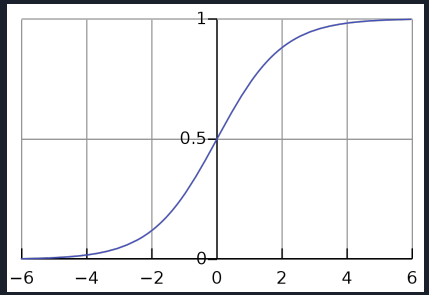


And stack layers – but hold on

Note:

- filters in reality are set randomly, as usual
- They will converge individually (hopefully) to more complex shapes
- Real values are used
- Color images have 3 dimensions: R, G and B

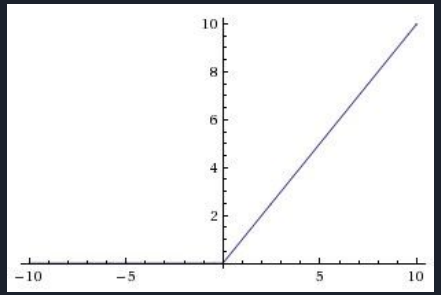
Gradient vanishing



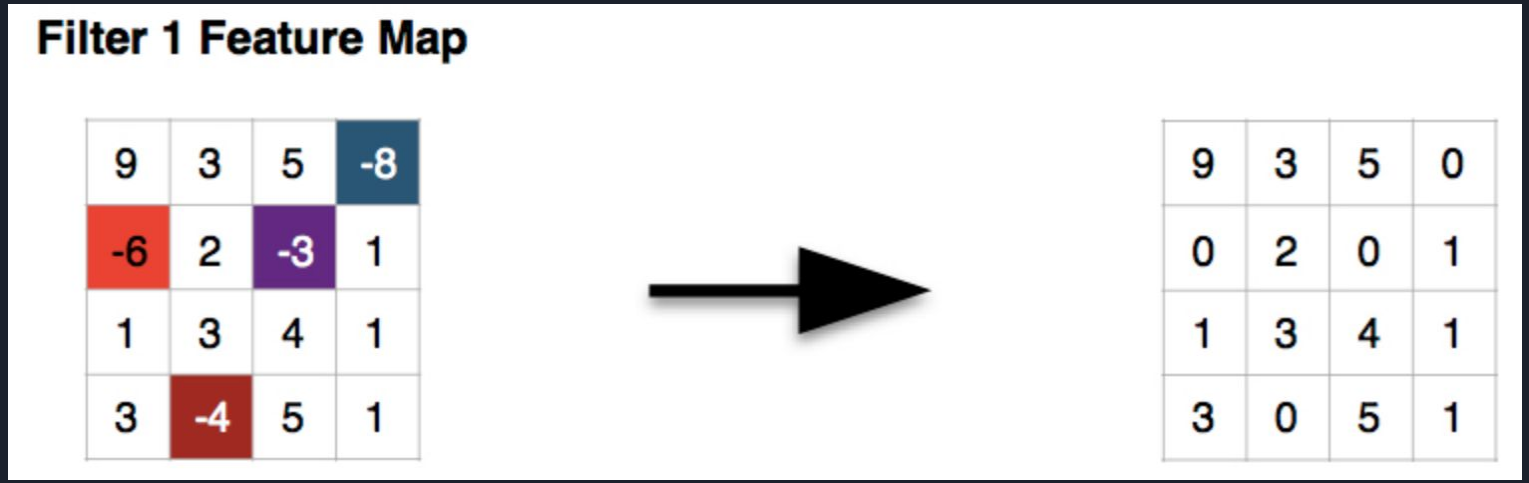
The sigmoid activation works well in general

But as the network grows, gradients become smaller and smaller => poor training

Solution: ReLU

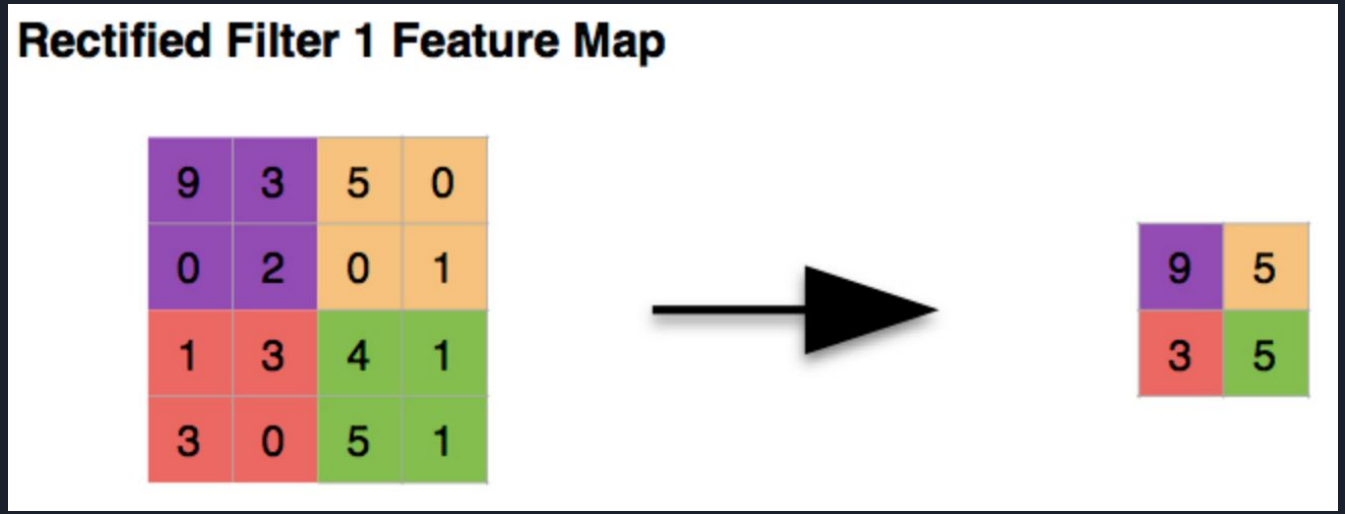


ReLU activation



(Max) pooling

Aggregates features = makes the network more robust to small changes



Fully connected layer

All neurons are connected as in the multilayer perceptron we saw

Acts like a vote:

- Input: high-level features from rectified filters of convolutions
- Each feature can “vote” for a class
- But some features can have more than one voice (e.g. one that identifies the cross in an “x”)

This layer is also part of training!

Let’s code it!

I know this is a lot of concepts, but they’re important.

Let’s see how it works in practice.

Open the notebook at the CNN section.

Recurrent Neural Networks

Another large class of networks

Intuition: like in hidden Markov models, better understand **sequences**

Particularly useful for text (e.g. sentiment analysis), or anything where each input can depend on the previous one(s).

Intuition of RNNs

Predict the blank word:

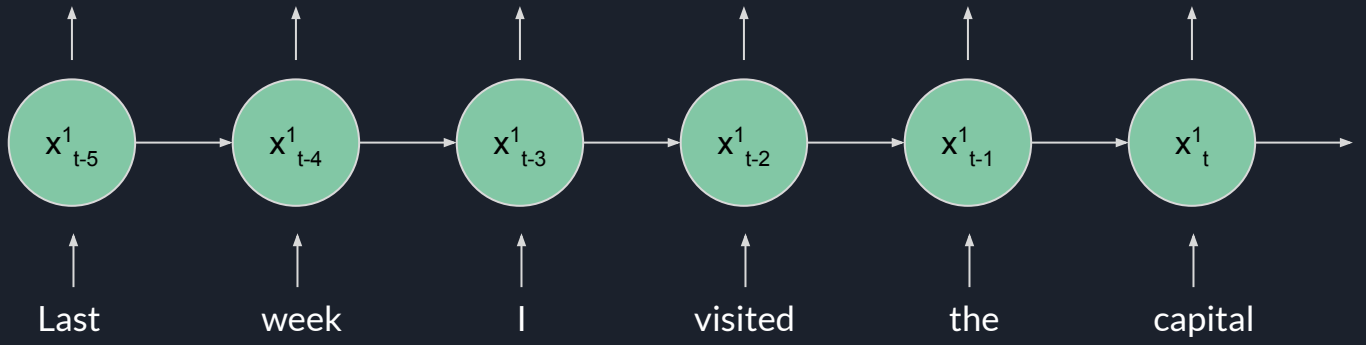
Last week I visited the capital of France. _____ really is a nice city!

How did you guess?

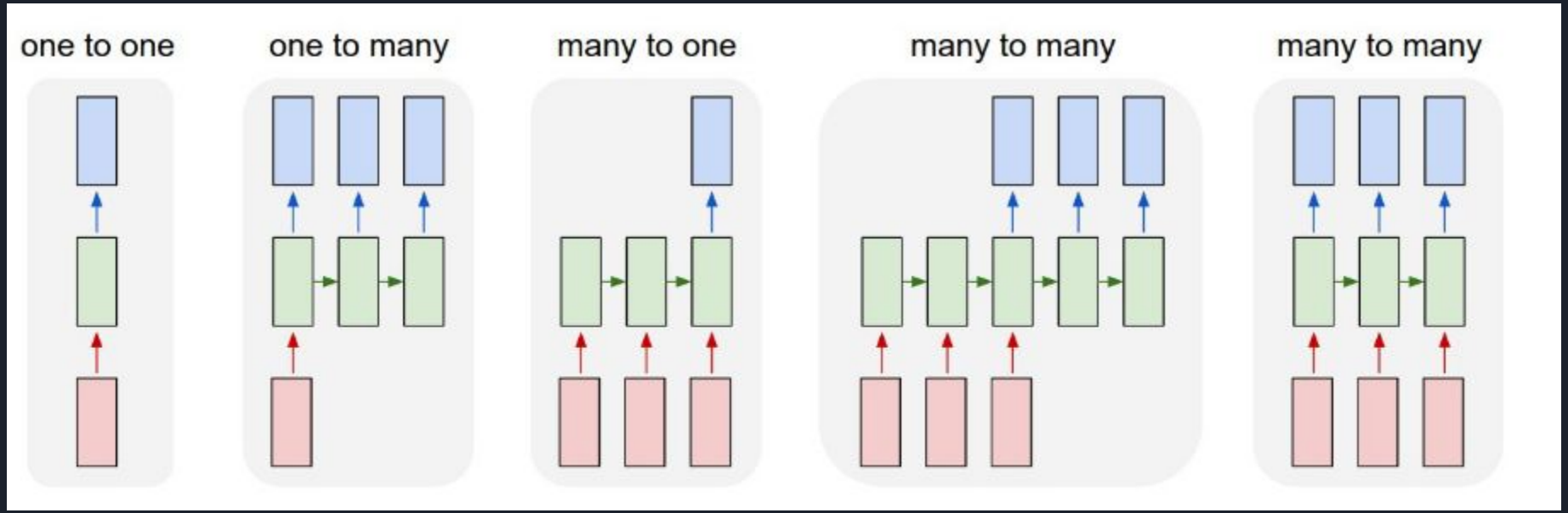
Recurrences

RNNs are less deep in the number of layers (usually 2-3 is enough)

They are deep in the number of **states** (memory)



Many applications



Different cells

LSTM: Long Short Term Memory

GRU: Gated Recurrent Unit

Bi-directional RNN

Etc.

But overall, despite their (theoretical) better modelling power, RNNs are slower than CNN.

Yet some applications seem **magical**.

Let’s code it!

Open the notebook to the RNN section.

What we haven’t seen today

Mathematical details

Optimization algorithms

Loss function details

Scalability (& GPU processing)

MANY other networks

But hopefully you’ll have enjoyed practicing with NNs and it got you enthusiastic about applying them to other tasks.

Resources

Used for this course:

- http://iamtrask.github.io/2015/07/12/basic-python-network/
- https://hashrocket.com/blog/posts/a-friendly-introduction-to-convolutional-neural-networks

Further reading:

- http://neuralnetworksanddeeplearning.com/
- https://www.deeplearning.ai/
- Wikipedia has a ton of resources too