COMP309 in Week 10, 2024

tensors and autograd

Marcus Frean `marcus.frean@vuw.ac.nz`

Week 10

- Lecture 1
  - Why Deep learning
  - The perceptron
    - activation function
  - Multi-layer Perceptrons (a.k.a. Neural networks)

- Lecture 2
  - tensors
  - automatic differentiation (autograd)
  - PyTorch for example

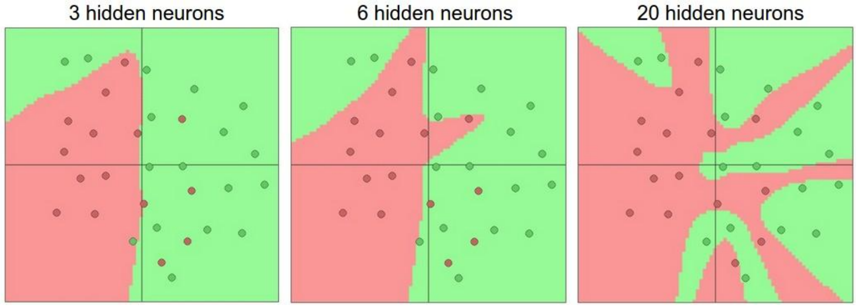
- Tutorial:
  - An end-to-end example of using PyTorch to solve a non-linear regression problem.

Neural networks: linear maps plus simple non-linearities

I. non-linearities provide the bare-minimum in expressive power. Then…

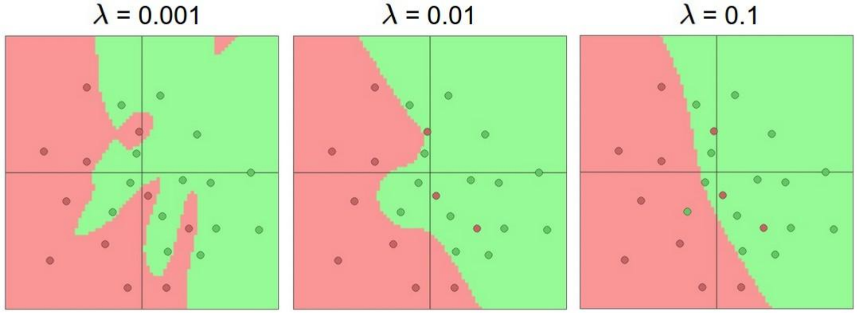
II. more layers □ richer mappings possible

III. more neurons per layer □ more flexibility

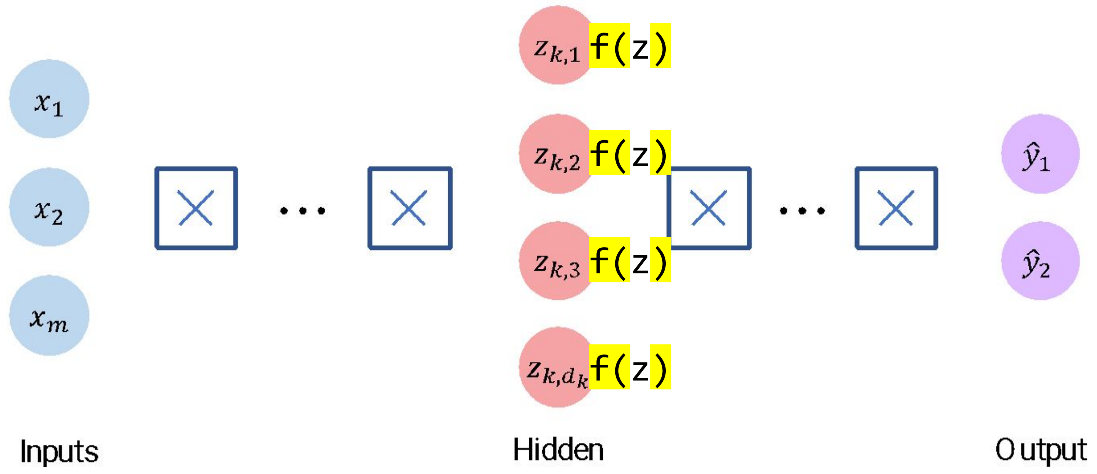


demo:
https://playground.tensorflow.org

HOWEVER too much freedom can □ overfitting!
We attempt to "regularise" with penalties and other constraints on model power ("smoothing")



Deep Neural Networks, Feed forward



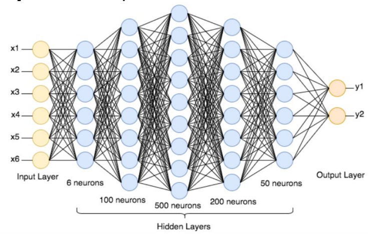
a neural net is really an alternating sequence of matrix multiplies ☒ and element-wise non-linearities

problems with gradients in really deep nets

Training a big neural net is computationally expensive (many epochs).

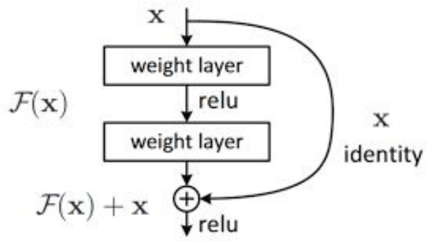
There are many “architectures” one might consider…

- number of layers
- number of nodes per layer



Depth was a major problem:

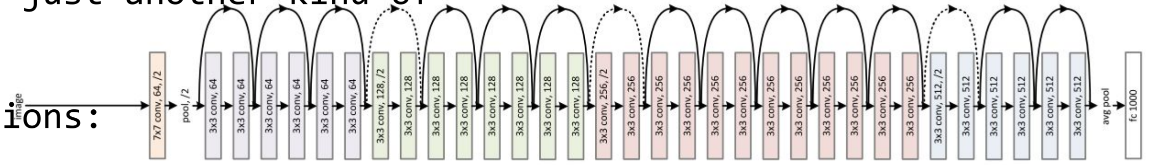
- gradients tend to (a) <u>vanish</u> and (b) get “<u>shattered</u>” in regular, deep, nets


x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Two particular techniques that seem to help:

BatchNorm (“batch normalisation”) - a scheme for “normalising” gradients all the way along. Implemented in frameworks as just another kind of layer.

ResNets, with “skip” connections:



TensorFlow *versus* PyTorch

TensorFlow - `developed at GoogleBrain (released 2015). Google used it for research and production. Intuitive high-level APIs such as Keras.`

Static graph : user first defines the computation graph of model and then runs the ML model (compile ☐ execute)

- more features, perhaps better for mobile and embedded deployments
- need "`Sessions`" and "`Placeholders`"...
- tensorboard – for visualising real-time accuracy graphs while training
- smooth production-ready deployment

PyTorch – `developed at Facebook (released 2016). No API as it smoothly integrates with the python data science stack and is similar to NumPy.`

Dynamic graph allows defining / manipulating the graph on the go. Imperative, interpreted, on-the-go.

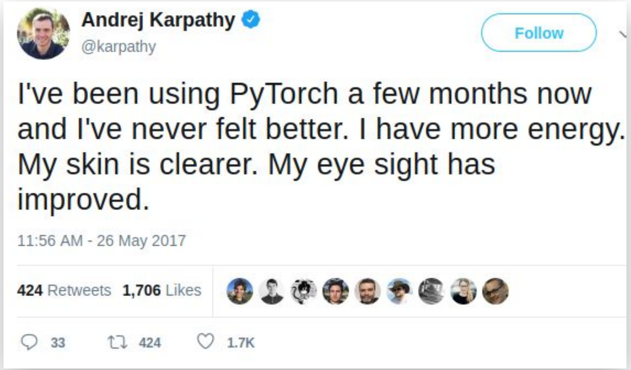
- attracting Python developers
- no need to create `session` or `placeholder` objects
- scripting, just like using NumPy, easier to debug

Both essentially provide fast (GPU-enabled) **tensors** and **graphs**, including **autograd.** In 2023 it seems:

- **TensorFlow** with Keras : industry - make things faster and build ML products at scale
- **PyTorch**: research-oriented developers and python programmers - more customisable

We'd prefer you use PyTorch for the final project

PyTorch



Andrej Karpathy
@karpathy

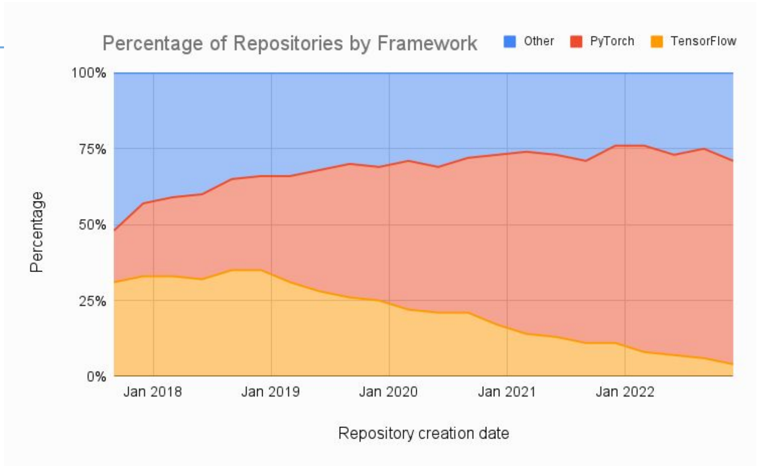
Follow

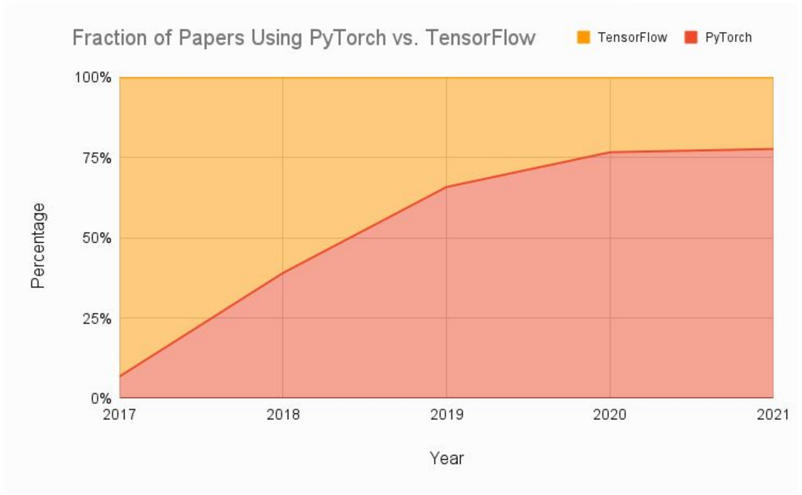
I've been using PyTorch a few months now and I've never felt better. I have more energy. My skin is clearer. My eye sight has improved.

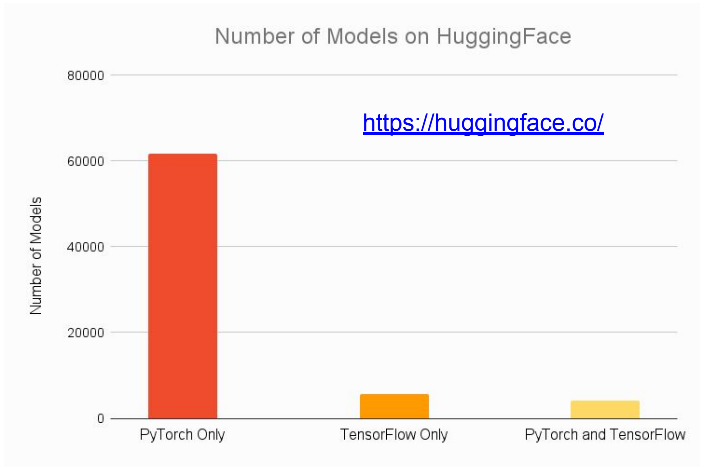
11:56 AM - 26 May 2017

424 Retweets 1,706 Likes

33 424 1.7K

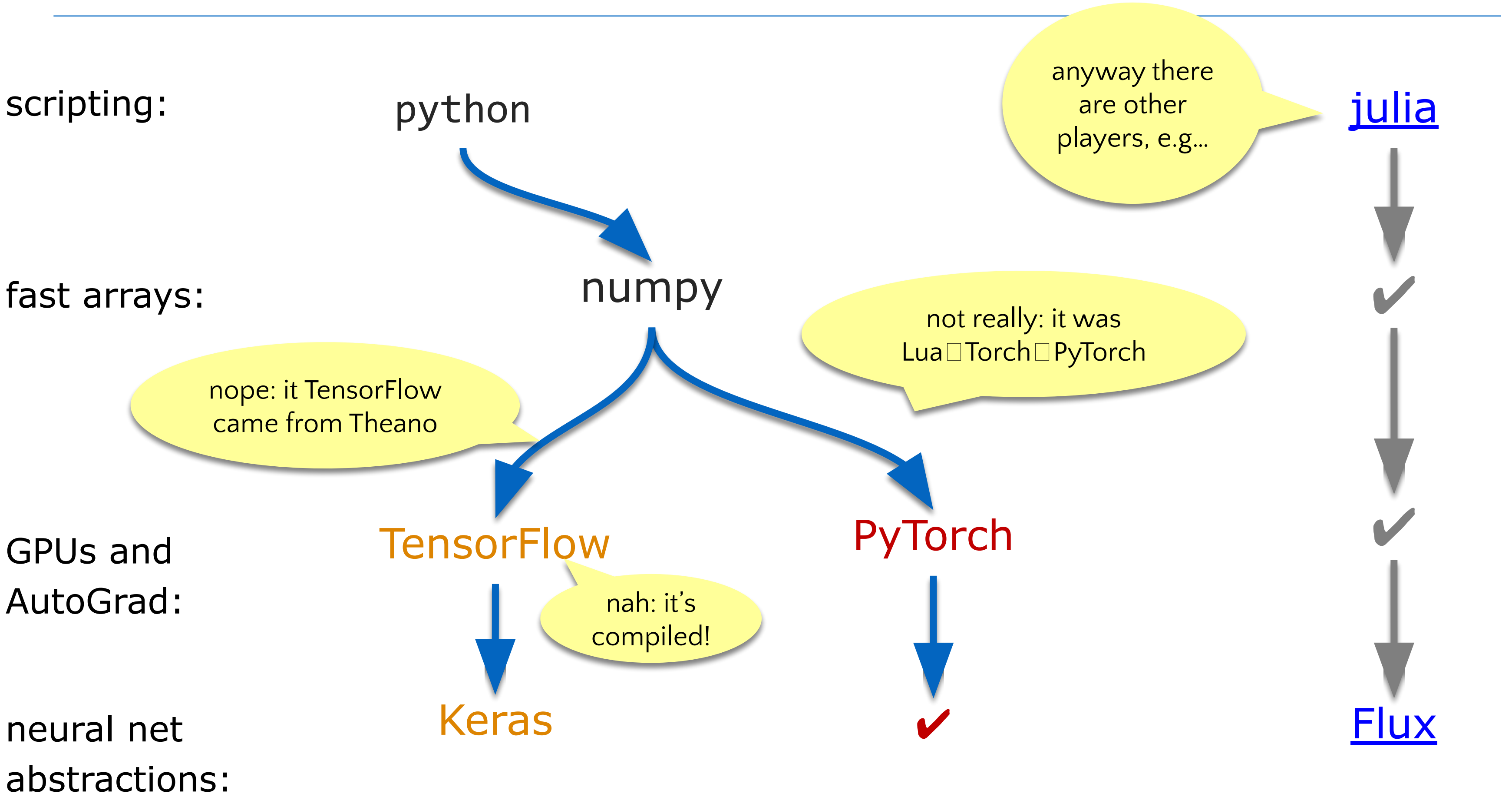






https://huggingface.co/

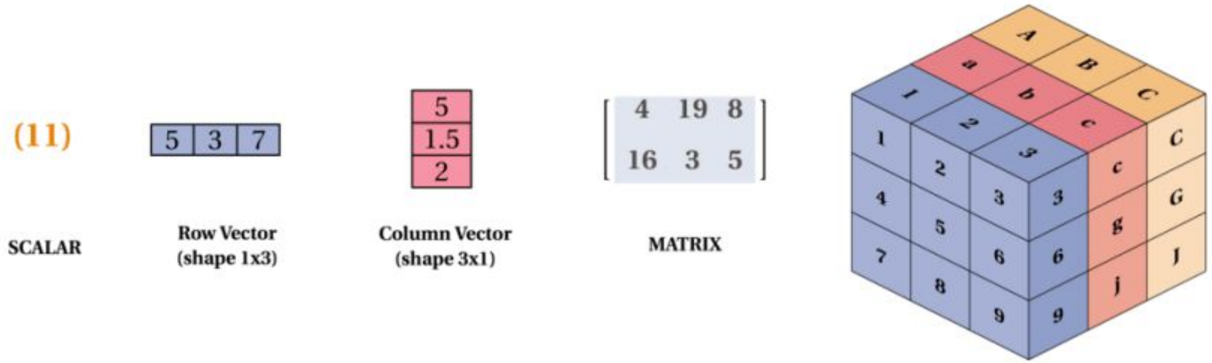
loose and bad history

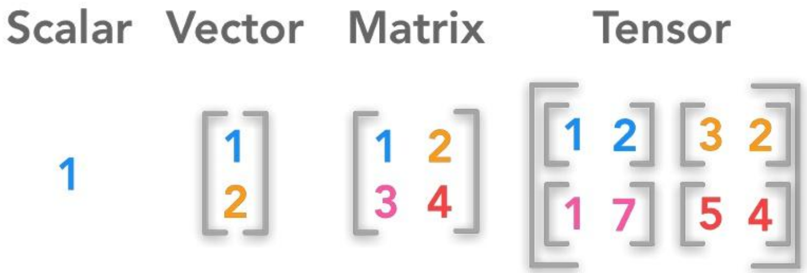


PyTorch

Basically, PyTorch is like numpy but + GPUs and autograd

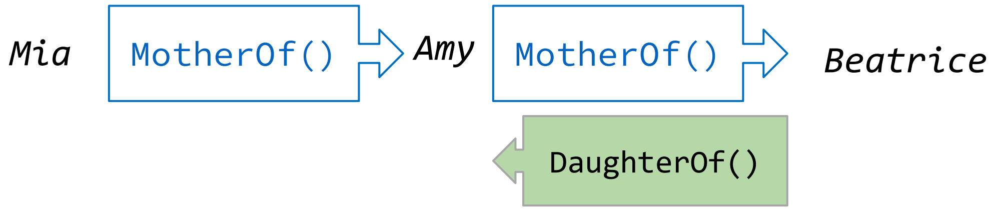
Essential data structure is the tensor





what we're *not* going to do, with functions

Function "compose": the grandmother of *Mia* is *Beatrice*



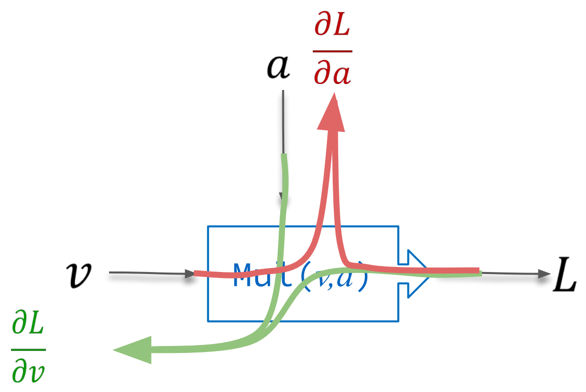
And we can often go backwards via the inverse of the forwards function…

However inverses are *NOT* what we're talking about next!

gradients of a function

$$\frac{\partial x}{\partial x} = 1$$

$$L = v \times a$$



a
$\frac{\partial L}{\partial a}$
v
Mult(v,a)
L
$\frac{\partial L}{\partial v}$

we can think of both the forward and the gradient computations as processing steps on a computational graph.

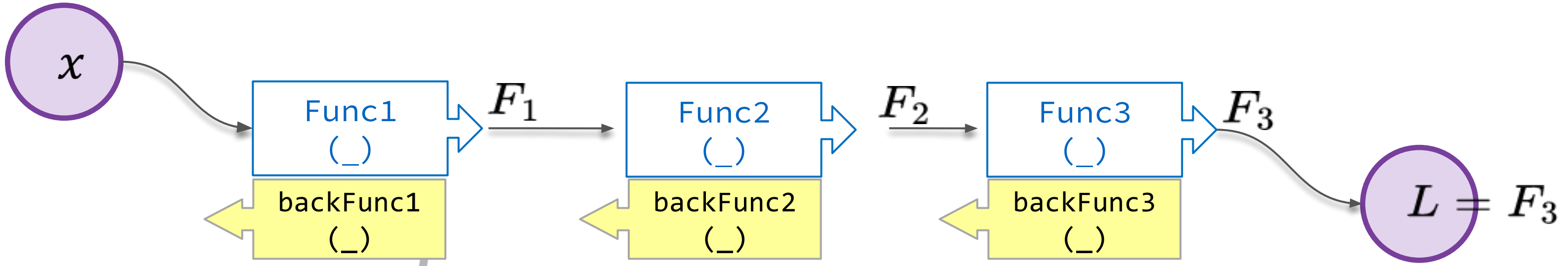
It makes sense to store those gradient functions at nodes on the same graph as the forward functions.

the chain rule

Say I have a function of a function of a function…

for example, L = Func3(Func2(Func1(x)))

To calculate L we do a forward pass through this graph. Each node remembers its forward-travelling values.



Q: how does L change if we wiggle x?

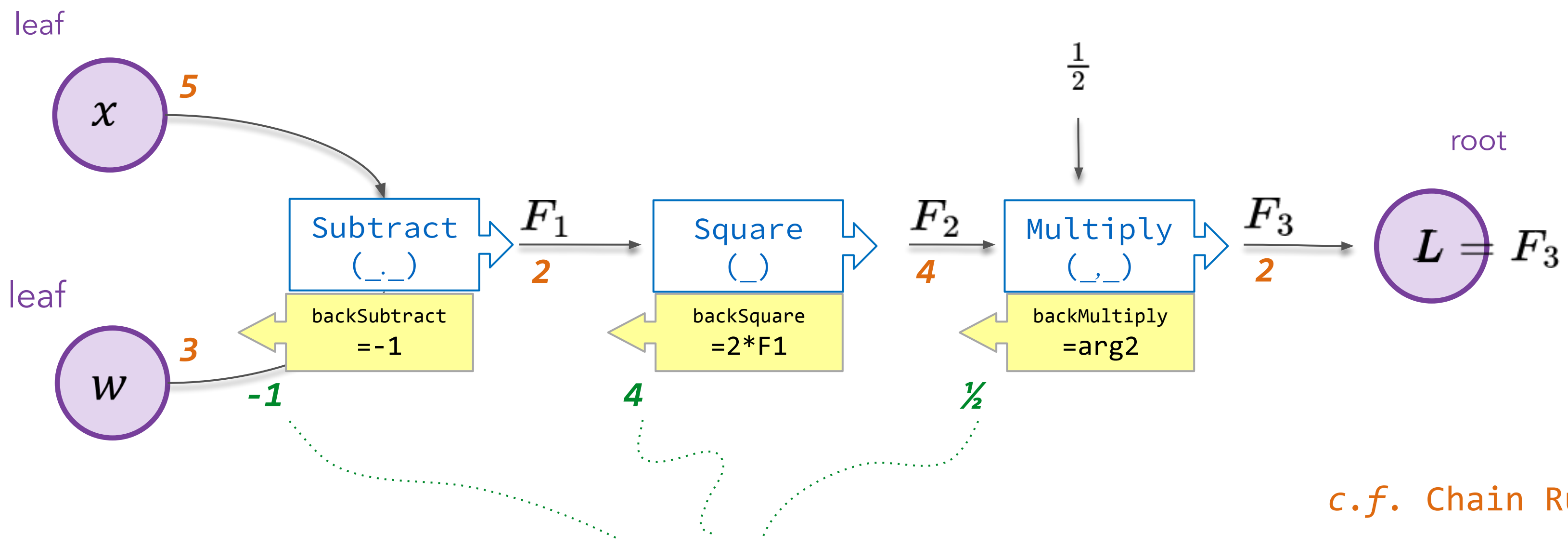
Ans: Chain Rule of calculus

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial F_3}\frac{\partial F_3}{\partial F_2}\frac{\partial F_2}{\partial F_1}\frac{\partial F_1}{\partial x}$$

autograd: automatic differentiation

Consider finding the gradient of this function: $L = \frac{1}{2}(x - w)^2$

when the inputs were 5 and 3



Gradient for w at that point is $-1 * 4 * ½$ = -2

c.*f*. Chain Rule:

$$\frac{\partial L}{\partial w} = 2 * \frac{1}{2}(x - w) * (-1)$$
$$= -2$$

“autograd”

- Implemented in TensorFlow and PyTorch (and elsewhere)

They “already know” Back…() for a set of mathematical operations Forw(), for example…

- ✔ plus, sum
- ✔ times (“Mul”)
- ✔ matrix multiplication (“MatMul”)
- ✔ exp, log,… and so on and on…
- ✔ batchNorm…
- ✔ linear…

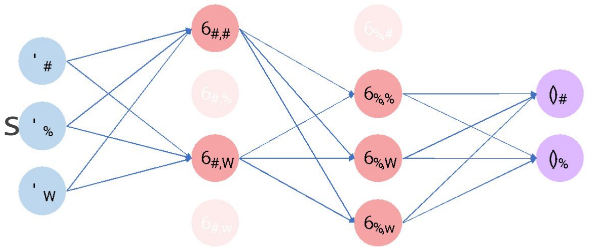
Note for the younger user: When I was a kid in grad school, we had to write our own, and it hurt, and we called it “backpropagation”

Autograd makes things *so so so much more pleasant*.

Regularisation – done better next week I think!...

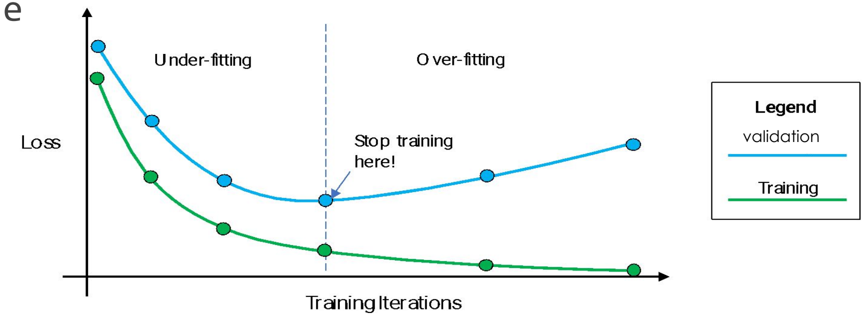
discourage *overly complex* models, to improve generalization of our model on unseen data

- Regularization 1: Penalties on weights
  - During training, pull a bit towards smaller values of the weights
  - L1 vs L2 (minimize abs(w) versus w^2)
- Regularization 2: Dropout
  - During training, randomly set some activations to 0
  - Typically ‘drop’ 50% of activations in layer
  - Forces network to not rely on any one node



- Regularization 3: Early Stopping
  - Stop training before we have a chance to overfit

Q: How to set regularisation parameters?...


Under-fitting
Over-fitting
Loss
Stop training here!
Training Iterations
Legend
validation
Training