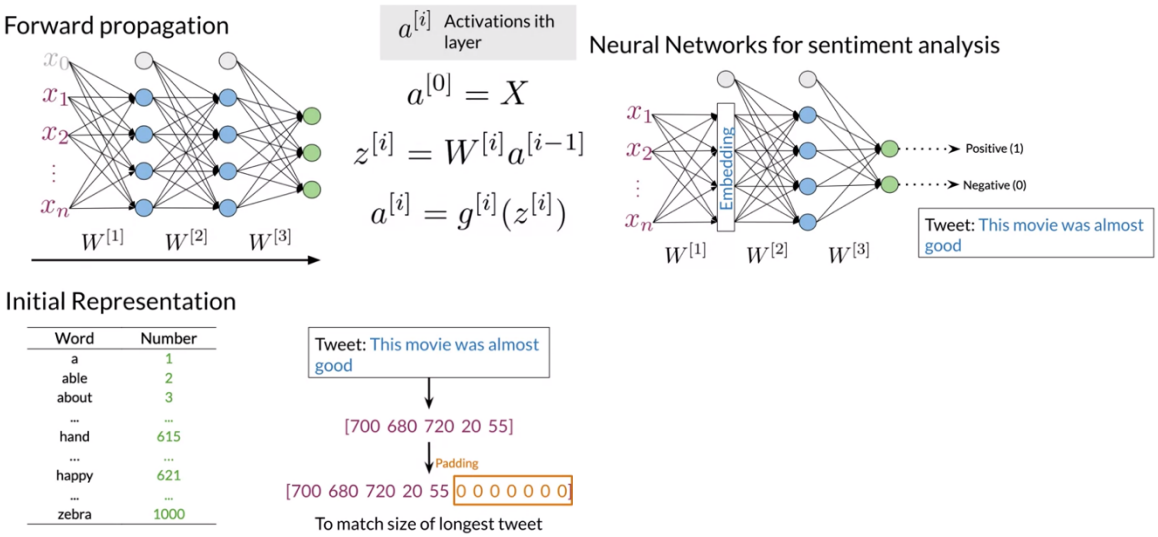
Neural Network for Sentiment Analysis

Neural Networks for Sentiment Analysis

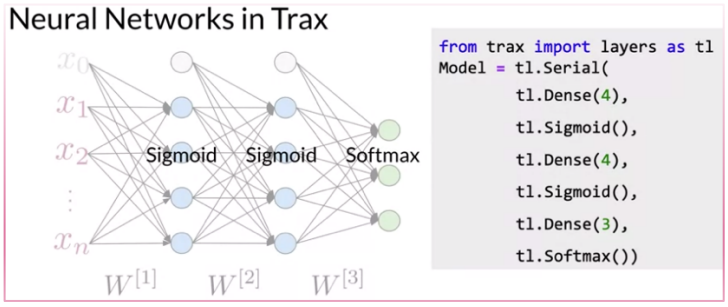


Trax: Neural Networks

Trax is built on TensorFlow

Advantages:

- Efficient computation
- Parallel computing by running gear models in multiple m/c
- Keeps a record of all algebraic operations on your NN in order of computation.
- Able to compute the gradients of the model automatically.



Why TRAX

Trax: Background

Why Trax and not TensorFlow or PyTorch?

TensorFlow and PyTorch are both extensive frameworks that can do almost anything in deep learning. They offer a lot of flexibility, but that often means verbosity of syntax and extra time to code.
Trax is much more concise. It runs on a TensorFlow backend but allows you to train models with 1 line commands. Trax also runs end to end, allowing you to get data, model and train all with a single terse statements. This means you can focus on learning, instead of spending hours on the idiosyncrasies of big framework implementation.

Why not Keras then?

Keras is now part of Tensorflow itself from 2.0 onwards. Also, trax is good for implementing new state of the art algorithms like Transformers, Reformers, BERT because it is actively maintained by Google Brain Team for

advanced deep learning tasks. It runs smoothly on CPUs,GPUs and TPUs as well with comparatively lesser modifications in code.

How to Code in Trax

Building models in Trax relies on 2 key concepts:- **layers** and **combinators**. Trax layers are simple objects that process data and perform computations. They can be chained together into composite layers using Trax combinators, allowing you to build layers and models of any complexity.

Trax, JAX, TensorFlow and Tensor2Tensor

You already know that Trax uses Tensorflow as a backend, but it also uses the JAX library to speed up computation too. You can view JAX as an enhanced and optimized version of numpy.
Watch out for assignments which import import trax.fastmath.numpy as np. If you see this line, remember that when calling np you are really calling Trax's version of numpy that is compatible with JAX.
As a result of this, where you used to encounter the type numpy.ndarray now you will find the type jax.interpreters.xla.DeviceArray.
Tensor2Tensor is another name you might have heard. It started as an end to end solution much like how Trax is designed, but it grew unwieldy and complicated. So you can view Trax as the new improved version that operates much faster and simpler.

Resources

- Trax source code can be found on Github: Trax
- JAX library: JAX
- Official maintained documentation - trax-ml not to be confused with this TraX

```
#!pip install trax==1.3.1
```

```
import numpy as np  # regular ol' numpy
```

```
from trax import layers as tl  # core building block
```

```
from trax import shapes  # data signatures: dimensionality and type
```

```
from trax import fastmath  # uses jax, offers numpy on steroids
```

```
!pip list | grep trax
```

```
'/opt/conda/bin/python -m pip install --upgrade pip'
```

Trax: Layers Classes and Subclasses

Subclasses

```
class MyClass(Object):

    def __init__(self,y):
        self.y = y

    def my_method(self,x):
        return x + self.y

    def __call__(self,x):
        return self.my_method(x)
```

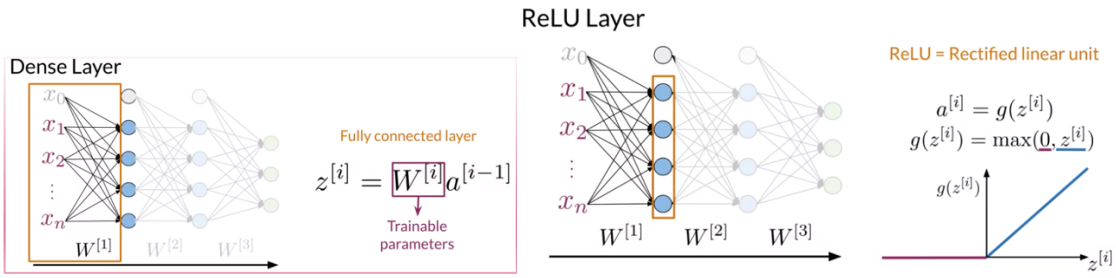
```
class SubClass(MyClass):

    def my_method(self,x):
        return x + self.y**2

f = SubClass(7)
print(f(3))
```

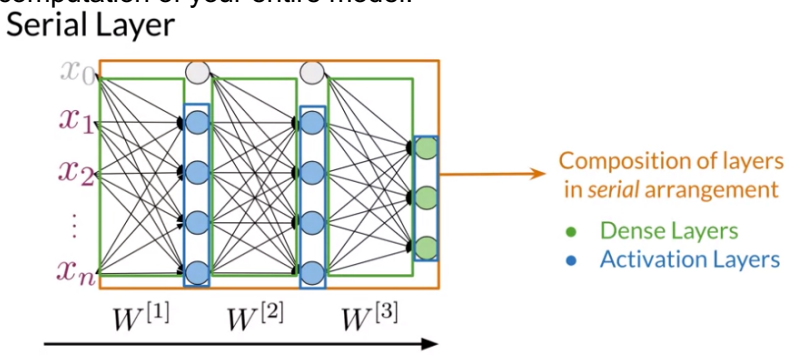
```
52
```

Dense and ReLU Layers



Serial Layer

A serial layer is a composition of sublayers that operates sequentially to perform the forward computation of your entire model.

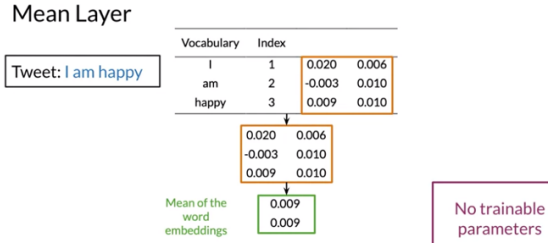


Other Layers

Embedding Layer – Embedding is trainable using an embedding layer. An embedding layer takes an index assigned to each word from your vocabulary and maps it to a representation of that word with a determined dimension. In this example, embedding of size equal to two.

Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010
because	4	-0.011	-0.018
learning	5	-0.040	-0.047
NLP	6	-0.009	0.050
sad	7	-0.044	0.001
not	8	0.011	-0.022

Mean Layer – Mean layer gives a vector representation.



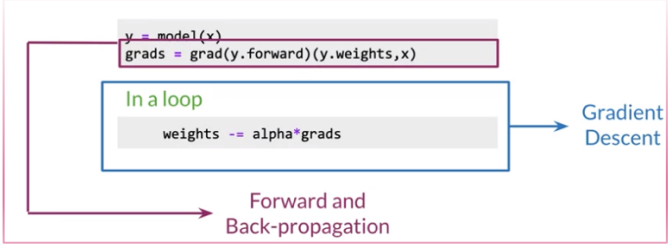
Vocabulary	Index		
I	1	0.020	0.006
am	2	-0.003	0.010
happy	3	0.009	0.010

0.020 0.006
-0.003 0.010
0.009 0.010

0.009
0.009

Training

- Computing gradients in trax
- Grad()
    - F = 3*x^2 + x; gradf = trax.math.grad(f)
- Y=model(x); grads = grad(y.forward)(y.weights, x); weights -=alpha*grads



Trax: Reading

Official Trax documentation maintained by the Google Brain team:

https://trax-ml.readthedocs.io/en/latest/

Trax source code on GitHub:

https://github.com/google/trax

JAX library:

https://jax.readthedocs.io/en/latest/index.html
