DEEP LEARNING

AAKASH GHOSH

Contents

Part I

Theory of Deep Learning

Chapter 1

Deep Learning

1.1 Traditional Machine Learning

Traditionally, solving a problem by machine

Part II

Natural Language Processing for Deep Learning

Chapter 2

Why is language a pain in the ass?

Written as a person fluent in English, Bengali and Hindi

Chapter 3

Natural Language Processing

3.1 Introduction

Language arose about $100,000 - 1,000,000$ years ago. As it turned out, the biggest power was not fangs and poison but communication. And it was language and writing that took us from Bronze Age to modern science. As far as AI is concerned we have made large progress. Machine translation are readable (even if all the nuances are not conveyed). The single biggest development was of course GPT3. One of the key feature was that it was a universal model: we no longer need different classifier/neural network to o different task, one universal model was enough.

3.2 Laguage modelling

A word is the verbal/written representation of idea. Traditionally, a computer understands a word by using a dictionary and looking up synonyms(word with similar meaning), hypernyms(expression of belonging to a more general category), hyponyms(expression of belonging to more specific subclass). A minimal example implemented in NLTK is shown below. Some arrays are truncated for brevity.

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synonyms('car')
[['auto', 'automobile', 'machine', 'motorcar'], ['railcar', 'railroad_car',
    'railway_car'], ['gondola'], ['elevator_car'], ['cable_car']]
>>> wn.synsets('car')[:2]
[Synset('car.n.01'), Synset('car.n.02')]
>>> wn.synset('car.n.01').definition()
'a motor vehicle with four wheels; usually propelled by an internal combustion
    engine'
>>> wn.synset('car.n.01').examples()
['he needs a car to get to work']
>>> wn.synset('car.n.01').hypernyms()
[Synset('motor_vehicle.n.01')]
>>> wn.synset('dog.n.01').hypernyms()
[Synset('canine.n.02'), Synset('domestic_animal.n.01')]
>>> wn.synset('car.n.01').hyponyms()[:5]
```

```
[Synset('ambulance.n.01'), Synset('beach_wagon.n.01'), Synset('bus.n.04'),
    Synset('cab.n.03'), Synset('compact.n.03')]
```

But this is not a very robust solution: it should be trivial to note that meaning of words are often dependent on context and nuances are not conveyed(Ex: "crimson" is a synonym of "red" but "the color of an apple is crimson" is a weird sounding sentence). Moreover, language is ever-changing and keeping up to date is hard. In traditional bag of words approach, words are given a one hot vector representation. For example:

$$\text{"hotel"} = [0, 0, 0, 0, 0, 1, 0, 0, 0]$$
$$\text{"motel"} = [0, 0, 0, 0, 0, 0, 0, 1, 0]$$

But this approach has its own limitations:

1. Vector size increase with increase in vocabulary
2. We understand that the words above are similar, but that similarity is not reflected in such one-hot encoding.

Part III

Codes

Chapter 4

Tensorflow ANN Cookbook

4.1 Introduction

4.2 Include Script

We will be using TensorFlow for our works here. The `include` script looks like the following:

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib
```

4.3 Loading Data

For the examples we mentioned above we can simply use already available datasets:

```
fashion_mnist = tf.keras.datasets.fashion_mnist.load_data()
```

We split in train-validation-test segments:

```
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist
X_train, y_train = X_train_full[:-5000], y_train_full[:-5000]
X_valid, y_valid = X_train_full[-5000:], y_train_full[-5000:]
```

Now define the class labels:

```
class_names = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat","Sandal",
    "Shirt", "Sneaker", "Bag", "Ankle boot"]
```

And we are ready to go. Things to note:

1. Inputs must be `numpy` arrays.