Ex. 3 NN and Deep Learning

Federico Bottaro

February 13, 2020

Abstract

In this document we are going to explore the field of the Recurrent Neural Network (RNN). In particular we are going to train our network using a text (in our case we choose *Siddharta*) and then, giving to the network an initial seed, we get a text as output. In this document we apply a seq2seq approach so the algorithm produces the text letter by letter.

1 Project development

Before start with the core of our project we have to make some pre-processing on the data. Some class are created for this scope. In practice the goal is to simplify the text itself. Different steps have been taken to reach the final result:

- We split the dataset into paragraphs.
- We extract the alphabet present in the book.
- We can transform the dataset of letter into a dataset of number (with unique match between them).
- We can encode the information also in a one hot matrix.
- We can perform a random crop on the paragraphs meaning that given a text as input we get as output a set of consecutive character (of fixed length) starting from a random point.
- At last we create a function for transfer the various information to a tensor (to use in the training with pythorch).

We can now start to build and to train various model. Due to the complexity of the RNN's we do not perform a systematic search on the parameters or on the structure of the network but we just use a random approach.

The first two implementation that we try reguard a research on the structure of the network. We train two RNN's with respectively:

- 2 hidden layer with 128 neurons.
- 4 hidden layers with 236 neurons.

We fix the other parameter of the training:

- number of epochs: 50000
- loss: CrossEntropyLoss

- optimizer: Adam with a learning rate of 0.001.

We can now move to the prediction of the networks giving as input the sequence: *siddhartha had been*. The idea behind this first prediction script was to give as output the character that has the greater probability in the output layer.

The results that we obtain with this setting are:

- 128 / 2 layer:

```
    siddhartha had beenihhe ta h hbgggww,l, rrrriuuuudd h ha hucc,mmpppoootth h
h bbbbb,,eeeerrrrrrtwww hh hyy..csshhhh....hh fffffwttwwhhbb,,hh hhhruummmdddhh
ia uurrriimmmddd
    aauu tooommmmdddh aa eeooommmhh  yaa sttoooimmdddd aa
uurrtttmmmmmmddrrrigwwwd h iii h iggggwwooppppddddhnnibhh hhbbbwwshh hhh
bbceeehhhh hh bbhhh aah  frrrfttmmdddddniiiiur rrugtwwddd  aa  nn ttgggww,,
```

- 236 / 4 layer:

```
    siddhartha had beenadddddnpppppttttttttbbbb..
yyccaeeeeeesddddddddpppvavvttttttccccvv hoo eee  ddddddddpppppvvttttttbbbc..
yccaaeee  thhaaiiiee  idd  lpppptttttttbbbbb.  cccaaeee  dddavvvvv  eeeaa
ddddnpppppttttttttbbbb.. ycccaeeeeeetddddddddpppvvvahhhe
eeaudddunnnnnssssssooooooovhh ,,,,usssnnssssooooovvoeeeeeeoudddaannanv va  ee
```

As we can see the output are sensless.

Trying to explain this bad behavior we can find that using only a fixed sequence of character for each paragraph for training the network we are passing in fact a dataset that is too small. To bypass the problem we try to implement two different methods. For the training part we decide to keep five sequences for each paragraph insted that one to make a more robust training set. In addition we decrease the crop len to 50. In this way we also exclude a fewer number of paragraphs (that one with a number of characters lower that the selected crop len). Reguarding the prediction script we implement a selection of the next character applying a softmax function to the output layer and we make a sampling from this softmax distribution.

In addition to facilitate the training we reduce the vocabulary excluding from the text all the special characters that are more rare (";!?") and we put all the text in lower case.

Using a batchsize of 100 we have that the training is much slower than the previous tests so we have to reduce the number of epoch for get the result in a reasonable time.

A first run to test the new generator was performed with 2500 epochs. Figure 1 shows the loss function during the training. For reducing the risk of overfitting we use an Adam optimizer with a weight decay of 5×10^{-4}. Doing so we are introducing an L2 penalty. We can now present a comparison of the output obtained using the two different generators:

- using maximum:

```
siddhartha had been the samanas and siddhartha was not a serface and siddhartha
was not a serface and siddhartha was not a serface and siddhartha was not a
serface and siddhartha was not a serface and siddhartha was not a serface and
siddhartha was not a serface and siddhartha was ...
```

- sampling from a softmax:

```
siddhartha had beent tell,a silgeines,that,her telenge abling
all a self has a singe tourst,hers aloust tougist teath had sampertand,wat has
now
to
hass the
saraity
this hast of had,she havollowed had,stalkind,timarind ouge,the
roway,whenersell,will
thim,tis on here welt,this
sittly.es,anceeded,also...

```

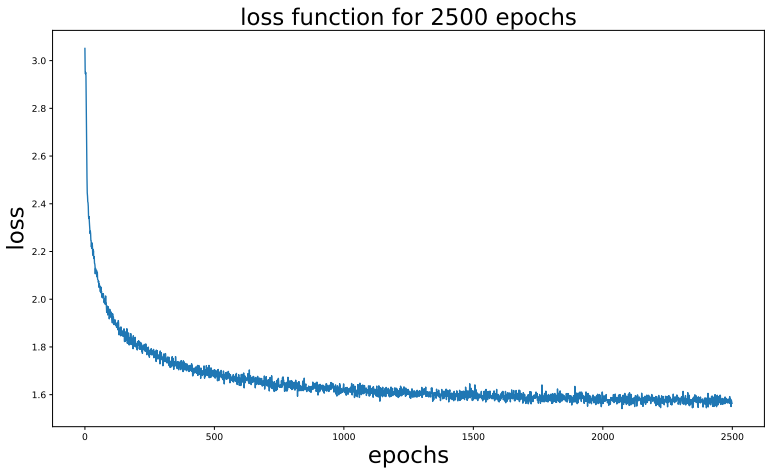

Figure 1: *Loss function for 2500 epochs*

As we can see we have that extracting the characters using the maximum, it seems to falls in a local minimum. On the other hand we have that sampling from a softmax distribution, the text has no sense but the grammatical part seems to be almost correct. In any case the outputs are a lot better with respect to the onces that we get from the firsts tests.

Another test that we can perform is to change the optimizer. The results that we get using a ASGD optimizer with learning rate of 0.05 and a weight decay of 5×10^{-4} is:

```
siddhartha had been a cerfeled,ank teare head alouge tomaci,has
aleed
alontered,a seace,it
indees that him,sam wathed
thine waten to bout,that,i wishot alow tiel,oud,teron hid,a himess of
alyes thick,welr,wist has a cilerced
ase to see all that
in shise asted
a sander
has,where
healte thine whene,thanded
then his...
```

Also in this case the grammar rule seems to be fallowed and we can distinguish some real word in the output. Figure 2 show the loss function. As we can see the loss function decrease also in this case so in fact this model can be a good model.

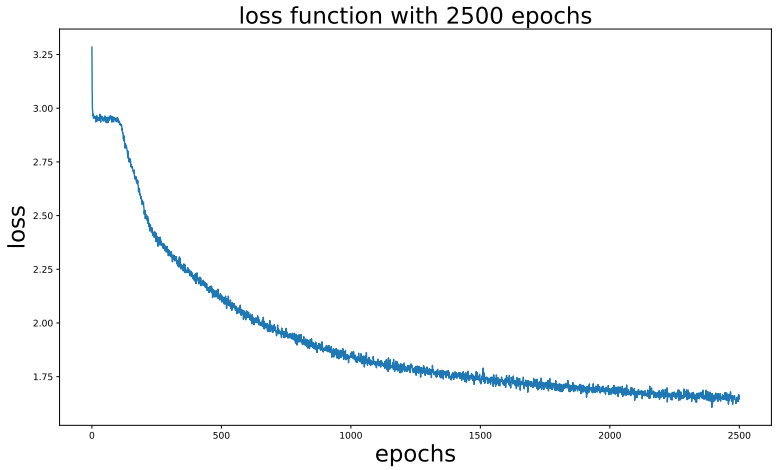

Figure 2: *Loss function for 2500 epochs using an ASGD optimizer*

Both the loss function that we plot are evaluated using the sampling methode to predict the next character.

We can now present the results that we obtain with our final model. The training is performed using the fallowing parameter:

- number of hidden layer: 2
- number of neurons: 128
- optimizer: Adam with lr=0.0005, weight decay= 5×10^{-4}
- loss= CrossEntropyLoss
- number of epochs: 10000

The output is:

```
siddhartha had been a sate
thinkist,his heand
any dead him they,while,whee seed
thangh and
thinks whenself he who
have longinne his
thing,her withon tee to
tell,the
sawart,a dead,to
head tout he wast they
antand that,in
sind that
shoughts
and
searded,an to
love had
thit
wan and,hadded his formand towas
allowint and,hast the
bedhe anythink,and he his else to
```

Similarly to that obtained for 2500 epochs we have that the model produce real words and also the grammatical structure seems to be respected. The highest number of epochs

produce an output that is more clear respect to the one obtained with 2500 epochs. The plot of the loss function is shown if figure 3.

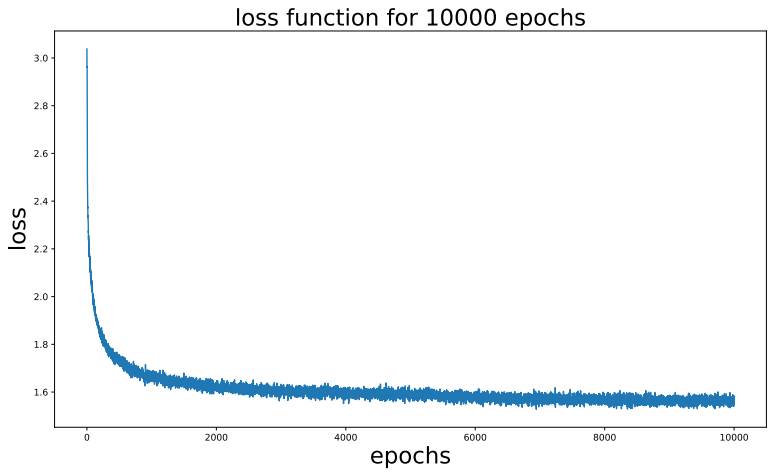

Figure 3: *Loss function for 10000 epochs*

2 Discussion

In this document we explore the field of the RNN and we find out a model that predict a text that seems to be good. Due to the computational complexity of the RNN a better model can't be trained in my personal pc (i use Google colaboratory but after a certain time i was kicked). Having found that with 10000 epochs we get a better results, probably if we set even more epochs we have a text with sense.