1.(a) $P(w) = P(w_1, \ldots, w_N) = P(w_1)\, P(w_2|w_1)\, P(w_3|w_1, w_2) \cdots P(w_N|w_1, \ldots, w_{N-1})$

$= \prod_{i=1}^{N} P(w_i | w_{i-n+1}, \ldots, w_{i-1})$ For n-gram

$P(w) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2) \cdots P(w_N|w_{N-1})$

$= \prod_{i=1}^{N} P(w_i | w_{i-1})$ For bigram.

where $P(w_0) = P(*)$

(b)

i	made	cheese	at	home	like	is	tasty	that	salty
3	3	4	2	3	2	2	1	1	1

	i	made	cheese	at	home	like	is	tasty	that	salty	EOS
i		1				2					
made			2	1							
cheese		1		1					1		1
at					2						
home		1					1				1
like			1		1						
is								1		1	
tasty											1
that							1				
salty											1

$$P(\text{[BOS] I like cheese made at home [EOS]})$$

$$= P(\text{[BOS]}) \cdot P(\text{I} \mid \text{[BOS]}) \cdot P(\text{like} \mid \text{I}) \cdot P(\text{cheese} \mid \text{like}) \cdot P(\text{made} \mid \text{cheese})$$

$$\cdot P(\text{at} \mid \text{made}) \cdot P(\text{home} \mid \text{at}) \cdot P(\text{[EOS]} \mid \text{home})$$

$$= \frac{3}{4} \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{3} \times \frac{2}{2} \times \frac{1}{3} = \boxed{\frac{1}{144} = 6.944 \times 10^{-3}}$$

(c) perplexity is measurement of how well a probablity distribution or probability model predicts a model. Lower is better, indicating the probability distribution is good at predicting the sample.

$$\text{perplexity} = 2^{-\frac{1}{M}\sum_{i=1}^{\text{sent}} \log_2 P(S_i)}$$

$$= 2^{-\frac{1}{8}\log_2\left(\frac{1}{144}\right)}$$

$$= 1.861.$$

(d) $P(\text{[BOS] I like cheese made at home [EOS]})$

$= P(\text{[BOS]}) \cdot P(\text{I} \mid \text{[BOS]}) \cdot P(\text{like} \mid \text{I}) \cdot P(\text{cheese} \mid \text{like}) \cdot P(\text{made} \mid \text{cheese})$

$\cdot P(\text{at} \mid \text{made}) \cdot P(\text{home} \mid \text{at}) \cdot P(\text{[EOS]} \mid \text{home})$

$= \frac{3+0.1}{4+0.1\times 12} \times \frac{2+0.1}{3+0.1\times 12} \times \frac{1+0.1}{2+0.1\times 12} \times \frac{1+0.1}{4+0.1\times 12} \times \frac{1+0.1}{3+0.1\times 12} \times \frac{2+0.1}{2+0.1\times 12} \times \frac{1+0.1}{3+0.1\times 12}$

$= \frac{31}{52} \times 0.5 \times 0.3435 \times \frac{11}{52} \times \frac{11}{42} \times \frac{21}{32} \times \frac{11}{42}$

$= \boxed{9.757 \times 10^{-4}}$

(e)

i	made	cheese	at	home	like	is	UNK
3	3	4	2	3	2	2	3

	i	made	cheese	at	home	like	is	UNK	EOS
i	1					2			
made		2	1						
cheese	1		1					1	1
at				2					
home	1					1			1
like		1			1				
is								2	
tasty									
that						1			2
scuffy									

UNK

$$P([BOS]\ I\ like\ pepperjack\ cheese\ [EOS])$$

$$= P([BOS]) \times P(I \mid [BOS]) \times P(like \mid I) \times P(UNK \mid like) \times P(cheese \mid UNK) \times P([EOS] \mid cheese)$$

$$= 1 \times \frac{3+0.1}{4+10\times0.1} \times \frac{2+0.1}{3+10\times0.1} \times \frac{0.1}{2+10\times0.1} \times \frac{0.1}{3+10\times0.1} \times \frac{1+0.1}{4+10\times0.1}$$

$$= 5.9675 \times 10^{-5}$$

2 (a)

$$P(\text{store} \mid \text{computer}) = \frac{C(\text{computer, store})}{C(\text{computer})} \quad \begin{matrix}\text{Bigram} \\ \text{Unigram}\end{matrix}$$

$$= \boxed{\frac{8}{22} = 0.364}$$

$$P(\text{monitor} \mid \text{computer}) = \frac{C(\text{computer, monitor})}{P(\text{computer})} = \boxed{\frac{8}{22} = 0.364}$$

Based on raw bigram probability, we can NOT compare which one is more likely, because the bigram probability are the same, 0.364.

2(b) $d = 0.5$

$$P(\text{store} \mid \text{computer}) = \frac{\max(C(\text{computer, store}) - d, 0)}{C(\text{computer})} + \frac{d}{\sum_v C(\text{computer}, v)} \times |w : C(\text{computer}, w) > 0| \times \frac{|v : C(v, \text{store}) > 0|}{\sum_{w'} |v \in V : C(v, w') > 0|}$$

$$= \frac{\max(8 - 0.5, 0)}{22} + \frac{0.5}{6+8+8} \times 3 \times \frac{3}{9}$$

$$= 0.364$$

$$P(\text{monitor} \mid \text{computer}) = \frac{\max(8-0.5,0)}{22} + \frac{0.5}{6+8+8} \times 3 \times \frac{1}{9}$$

$$= 0.348$$

$\because P(\text{store} \mid \text{computer}) = 0.364 > P(\text{monitor} \mid \text{computer}) = 0.348$

So "store" is more likely to appear in this sentence

I think the reason is that "store" is more likely to be seen everywhere, but "monitor" only appears after "computer" and also the frequency of bigram is the same, but "store" will also appear after other words, so it appears in more diverse context, hence "store" is slightly more likely to be seen.

2(c)

$$P(\text{store} \mid \text{computer}) = \frac{\max(C(\text{computer}, \text{store}) - d, 0)}{C(\text{computer})} + \frac{d}{\sum_v C(\text{computer}, v)} \times |w : C(\text{computer}, w) > 0| \times \frac{|v : C(v, \text{store}) > 0|}{\sum_{w'} |v \in V : C(v, w') > 0|}$$

$$= \frac{\max(8 - 0.1, 0)}{22} + \frac{0.1}{6+8+8} \times 3 \times \frac{3}{9}$$

$$= 0.364$$

$$P(\text{monitor} \mid \text{computer}) = \frac{\max(8 - 0.1, 0)}{22} + \frac{0.1}{6+8+8} \times 3 \times \frac{1}{9}$$

$$= 0.361$$

The result does NOT change. "store" is more likely to be seen than "monitor"

3(a)

$y = f(x)$ $\quad a = W^{(1)}x + b^{(1)}$

$z = \text{sigmoid}(a)$

$y = W^{(2)}z + b^{(2)}$

$\Rightarrow S(a) = \frac{1}{1+e^{-a}}$

Now change σ from sigmoid to tanh

$y = f'(x)$ $\quad a' = W^{(1)'}x + b^{(1)'}$

$z' = \tanh(a')$

$y = W^{(2)'}z' + b^{(2)'}$

$\Rightarrow \tanh(a') = \frac{e^{a'} - e^{-a'}}{e^{a'} + e^{-a'}}$

It is obvious that $\quad 1 - 2S(x) = -\tanh(\frac{x}{2})$

Hence if we want the output is the same,

$$\begin{cases} S(a) = z \\ \tanh(\frac{a}{2}) = 2S(a) - 1 \end{cases}$$

($\tanh(\frac{a}{2}) \rightarrow z'$, $2S(a)-1 \rightarrow z$)

① We make $W^{(1)'} = \frac{1}{2}W^{(1)}$

$b^{(1)'} = \frac{1}{2}b^{(1)}$

we can get $a' = \frac{a}{2}$

After this, we can make sure $1 - 2S(a) = -\tanh(\frac{a}{2})$

$\Rightarrow 1 - 2z = -z'$

② In order to make the output is the same

$W^{(2)}z + b^{(2)} = W^{(2)'}z' + b^{(2)'}$

$W^{(2)}(\frac{z'+1}{2}) + b^{(2)} = W^{(2)'}z' + b^{(2)'}$

$\Downarrow$

We need $W^{(2)'} = \frac{1}{2}W^{(2)}$

$b^{(2)'} = \frac{1}{2}W^{(2)} + b^{(2)}$

Heme:

$W^{(1)} \rightarrow \frac{1}{2}W^{(1)}$

$b^{(1)} \rightarrow \frac{1}{2}b^{(1)}$

$W^{(2)} \rightarrow \frac{1}{2}W^{(2)}$

$b^{(2)} \rightarrow \frac{1}{2}W^{(2)} + b^{(2)}$

3(b) Use Expectation to solve this problem.

Consider the probability of drop is P_{drop}.

then $E_{P_{drop}}[h_{drop}]_i = h_i \quad i \in \{1, \ldots, DK\}$

definitely choose

Before drop outs. $E[h] = h \cdot \times 1$

After dropouts. $E_p[h_{drop}] = (1 - P_{drop})\, h$

for each h_i, now expectation is $(1 - P_{drop})\, h$

Heme we should multiple $\gamma = \frac{1}{1 - P_{drop}}$ to balance that.

3 cc) I think it is RNN.

Because for language modeling, we need to consider the previous context when predicting next token.

RNN use "gru", which takes into x_t and hidden state from previous context as input to predict next token.

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

Also, like in LSTM, we has forget gate mechanism, helping to forget unimportant context over time.

4 (a) Results are shown in autoencoder

4(b). Here we did not consider smoothing, when we run

self.n ↓

m.prob('~~~', 'F')

we can find the probility is 1.

This is because the training file is started with 'F'

In training process. we never see any other letter after pad char,

so the probability of {'F' is the first letter} = 1 without smoothing.

For example k=2.

$$P\{'F' \mid '~~'\} = \frac{count('~~', 'F')}{count('~~')} \uparrow 0 = 0$$

Same with word-level, in train set. there is only 'In' after padding char. hence without smoothing, for example, self.n = 2

$$P\{'In' \mid '~ ~'\} = \frac{count('~ ~', 'In')}{count('~ ~')} \uparrow 0 = 0$$

4 (c) ① Which one is better ?

Word level seems better than Charcuter level

Reason:

- Word level can generate 100% correct words. But char level often give me incorrect word.
- Word level can give us a story with more correct grammar because it considers grammar when doing ngram in word level. For example the words after "she" is often "loves," "likes". But in char level, we need a much longer "n" to consider these.

② Perplexity.

Perplexity of same corpus in word-level n gram model is MUCH LARGER than that in char-level.

Because some words in test set may never appear in train set, but in char-level, it may appeared.

"Apple tree" never appear in train set, but "let" appears.

What's more, vocab of char-level model is very small when we use k-smoothing, even though we never see this context, there is still $\frac{1}{|V|}$ probability. But in word-level, vocabulary is much larger,

① CHAR-MODEL

4(d) We use perplexity as the metrics here.

We consider $n \in \{1, 2, 3, 4, 5\}$, $k \in \{0.01, 0.05, 0.1, 0.5\}$ and many different λ.

We find for

Best result for fixed n

n	k	λ	P
1	0.5	[1, 0]	12.28
2	0.5	[1, 0, 0]	7.91
3	0.1	[1, 0, 0, 0]	5.88
4	0.01	[0.6, 0.2, 0.1, 0.1, 0]	5.35
5	0.01	[0.4, 0.3, 0.2, 0.05, 0.05, 0]	5.09
6	0.01	[0.3, 0.2, 0.15, 0.15, 0.1, 0.1]	5.25

Hence, I tune more hyper-parameter based on n=5

$k \in \{0.001, 0.005, 0.01\}$

The best hyper-parameter I got is

$n = 5 \quad k = 0.01$

$\lambda = [0.4, 0.3, 0.2, 0.05, 0.05, 0]$

perplexity = 5.09.

② For word-level.

(2.1) shakespear : n= 1 λ= [0.5, 0.5]

k= 0.005

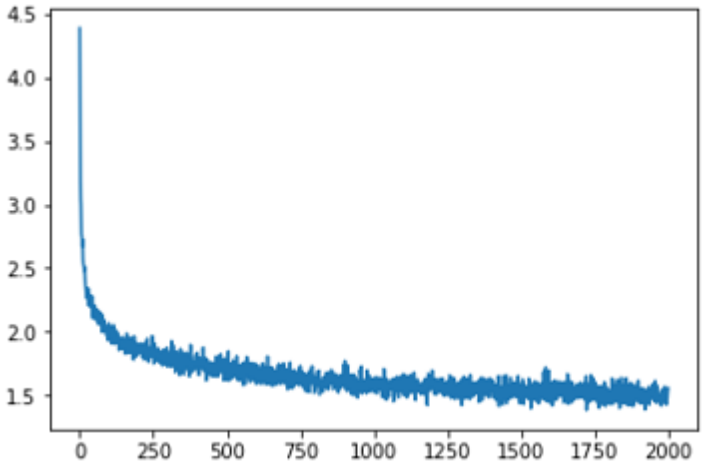
perplexity = 3037.37

(2.2) train.txt : n= 2 λ= [0.2, 0.5, 0.3]

k= 0.001

perplexity = 794.45

4 (e)



n_epoch = 10,000

hidden size = 256

n_layers = 2

lr = 0.0005

Perplexity = 5.41

The RNN model can produce much more readable and coherent texts than char-level n-gram.

Because RNN consider the previous context

What, have serve your love,
He hath the storseting blast, any exarch
In the own men. Go, did there

← Generated TEXT.

Generating Shakespeare with a Character-Level RNN

In this part, we'll turn from traditional n-gram based language models to a more advanced form of language modeling using a Recurrent Neural Network. Specifically, we'll be setting up a character-level recurrent neural network (char-rnn) for short.

Andrej Karpathy, a researcher at OpenAI, has written an excellent blog post about using RNNs for language models, which you should read before beginning this assignment. The title of his blog post is The Unreasonable Effectiveness of Recurrent Neural Networks (http://karpathy.github.io/2015/05/21/rnn-effectiveness/).

Karpathy shows how char-rnns can be used to generate texts for several fun domains:

- Shakespeare plays
- Essays about economics
- LaTeX documents
- Linux source code
- Baby names

Here are the materials that you should download for this assignment: * [training data and devlopment data for generation](https://www.cc.gatech.edu/classes/AY2020/cs7650_spring/hw3/lm/data.zip).

Recommended Reading

You should install PyTorch, know Python, and understand Tensors:

- http://pytorch.org/ (http://pytorch.org/) For installation instructions
- jcjohnson's PyTorch examples (https://github.com/jcjohnson/pytorch-examples) for an in depth overview

It would also be useful to know about RNNs and how they work:

- The Unreasonable Effectiveness of Recurrent Neural Networks (http://karpathy.github.io/2015/05/21/rnn-effectiveness/) shows a bunch of real life examples
- Understanding LSTM Networks (http://colah.github.io/posts/2015-08-Understanding-LSTMs/) is about LSTMs specifically but also informative about RNNs in general

Also see these related tutorials from the series:

- Classifying Names with a Character-Level RNN (https://github.com/spro/practical-pytorch/blob/master/char-rnn-classification/char-rnn-classification.ipynb) uses an RNN for classification
- Generating Names with a Conditional Character-Level RNN (https://github.com/spro/practical-pytorch/blob/master/conditional-char-rnn/conditional-char-rnn.ipynb) builds on this model to add a category as input

You can also set up Pytorch in Google Colab

Pytorch is one of the most popular deep learning frameworks in both industry and academia, and learning its use will be invaluable should you choose a career in deep learning.

Setup

Using Google Colab (recommended)

1. Upload this notebook on Colab (https://colab.research.google.com/notebooks/welcome.ipynb).
2. Set hardware accelerator to `GPU` under `notebook settings` in the `Edit` menu.
3. Run the first cell to set up the environment.

Note

Please look at the FAQ section before you start working.

Prepare data

The file we are using is a plain text file. We turn any potential unicode characters into plain ASCII by using the `unidecode` package (which you can install via `pip` or `conda`).

In [1]:
```
import unidecode
import string
import random
import re

all_characters = string.printable
n_characters = len(all_characters)

file = unidecode.unidecode(open('data/shakespeare_input.txt').read())
file_len = len(file)
print('file_len =', file_len)
```

```
file_len = 4573338
```

To make inputs out of this big string of data, we will be splitting it into chunks.

In [2]:
```
chunk_len = 200

def random_chunk():
    start_index = random.randint(0, file_len - chunk_len)
    end_index = start_index + chunk_len + 1
    return file[start_index:end_index]

print(random_chunk())
```

```
s was sequent
Thou know'st already.

HORATIO:
So Guildenstern and Rosencrantz go to't.

HAMLET:
Why, man, they did make love to this employment;
They are not near my conscience; their defeat
Does by th
```

Build the Model

This model will take as input the character for step t_{-1} and is expected to output the next character t. There are three layers - one linear layer that encodes the input character into an internal state, one GRU layer (which may itself have multiple layers) that operates on that internal state and a hidden state, and a decoder layer that outputs the probability distribution. You need to finish the forward method. (Refer to Pytorch GRU Documentation (https://pytorch.org/docs/stable/nn.html#gru))

In [40]:
```
import torch
import torch.nn as nn

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, n_layers=1
):
        super(RNN, self).__init__()
        self.input_size = input_size    # 100
        self.hidden_size = hidden_size    # 100
        self.output_size = output_size    # 100
        self.n_layers = n_layers    # 1

        self.encoder = nn.Embedding(input_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size, n_layers)
        print(self.gru.weight_ih_l0.shape)
        self.decoder = nn.Linear(hidden_size, output_size)

    def forward(self, input, hidden):
        #Input input: torch Tensor of shape (1,)
        #hidden: torch Tensor of shape (self.n_layers, 1, self.hidden_si
ze)
        #Return output: torch Tensor of shape (1, self.output_size)
        #and hidden: torch Tensor of shape (self.n_layers, 1, self.hidde
n_size)

        # embed input
        encode_layer = self.encoder(input).reshape(1,1,-1)
        y, hidden = self.gru(encode_layer, hidden)
        y = self.decoder(y.reshape(1,-1))
        return y, hidden

    def init_hidden(self):
        return torch.zeros(self.n_layers, 1, self.hidden_size).to(device
)
```

Inputs and Targets

Each chunk will be turned into a tensor, specifically a `LongTensor` (used for integer values), by looping through the characters of the string and looking up the index of each character in `all_characters`.

In [27]:
```
# Turn string into list of longs
def char_tensor(string):
    tensor = torch.zeros(len(string)).long().to(device)
    for c in range(len(string)):
        tensor[c] = all_characters.index(string[c])
    return tensor

print(char_tensor('abcDEF'))
```

```
tensor([10, 11, 12, 39, 40, 41])
```

Finally we can assemble a pair of input and target tensors for training, from a random chunk. The input will be all characters *up to the last*, and the target will be all characters *from the first*. So if our chunk is "abc" the input will correspond to "ab" while the target is "bc".

In [28]:
```
def random_training_set():
    chunk = random_chunk()
    inp = char_tensor(chunk[:-1])
    target = char_tensor(chunk[1:])
    return inp, target
```

Evaluating

To evaluate the network we will feed one character at a time, use the outputs of the network as a probability distribution for the next character, and repeat. To start generation we pass a priming string to start building up the hidden state, from which we then generate one character at a time.

In [29]:
```
def evaluate(prime_str='A', predict_len=100, temperature=0.8):
    hidden = decoder.init_hidden()
    prime_input = char_tensor(prime_str)
    predicted = prime_str

    # Use priming string to "build up" hidden state
    for p in range(len(prime_str) - 1):
        _, hidden = decoder(prime_input[p], hidden)
    inp = prime_input[-1]

    for p in range(predict_len):
        output, hidden = decoder(inp, hidden)

        # Sample from the network as a multinomial distribution
        output_dist = output.data.view(-1).div(temperature).exp()
        top_i = torch.multinomial(output_dist, 1)[0]

        # Add predicted character to string and use as next input
        predicted_char = all_characters[top_i]
        predicted += predicted_char
        inp = char_tensor(predicted_char)

    return predicted
```

Training

A helper to print the amount of time passed:

In [30]:
```
import time, math

def time_since(since):
    s = time.time() - since
    m = math.floor(s / 60)
    s -= m * 60
    return '%dm %ds' % (m, s)
```

The main training function

In [31]:
```
def train(inp, target):
#     print('inp', inp)
    hidden = decoder.init_hidden()
#     print('hidden', hidden.shape)
    decoder.zero_grad()
    loss = 0
#     print('chunk_len', chunk_len)
    for c in range(chunk_len):
#         print('inp[c], hidden', inp[c], hidden.shape)
        output, hidden = decoder(inp[c], hidden)
#         print('target.unsqueeze(1)[c]', target.unsqueeze(1)[c].shape)
        loss += criterion(output, target.unsqueeze(1)[c])

    loss.backward()
    decoder_optimizer.step()

    return loss.item() / chunk_len
```

Then we define the training parameters, instantiate the model, and start training:

In [41]:
```
n_epochs = 10000
print_every = 100
plot_every = 10
hidden_size = 256
n_layers = 2
lr = 0.0005
print('parameters => ', n_characters, hidden_size, n_characters, n_layer
s)
decoder = RNN(n_characters, hidden_size, n_characters, n_layers).to(devi
ce)
decoder_optimizer = torch.optim.Adam(decoder.parameters(), lr=lr)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
loss_avg = 0

for epoch in range(1, n_epochs + 1):
    loss = train(*random_training_set())
    loss_avg += loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) %.4f]' % (time_since(start), epoch, epoch /
n_epochs * 100, loss))
        print(evaluate('Wh', 100), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        loss_avg = 0
```

```
parameters =>  100 256 100 2
torch.Size([768, 256])
[0m 39s (100 1%) 2.5673]
Wh%asbf is wapgane le hand hal gese the hand wethre moy I eoused ve ly
minn nanveore the myan hand inu

[1m 26s (200 2%) 2.4654]
WhQatily so an.

L-ROKEEI:
And of in and thou leld, wold dFer inith il.

NIT
No whs sto to thigy rall

[2m 6s (300 3%) 2.2788]
Whe deare'd the aftrose, his hat ther, the to deme mat by wits coo<n ca
t's wentuy not matseos hat to t

[2m 45s (400 4%) 2.0963]
Whall and our beaversen, this this mut of ou, Searn sall ind bearmon; a
nd there paves
Andern!

CORCE:


[3m 34s (500 5%) 1.9813]
Whia to he ofour manston hin the, I ake with it upetiess to withs the m
owef do bet outher kir?

BENMIN

[4m 15s (600 6%) 1.8821]
What heard hom, onges as him
Intin:
The he ounder him the sourone,
I, a with and to have of eartlbine,

[4m 55s (700 7%) 2.2878]
Wher shere be thence the dotsen,
And the neving was a and dows:
What beat the ringue enave like this t

[5m 34s (800 8%) 1.7517]
What to munt the!

ONAMDADNOW:
My say o' Caron,
And beat be his of the cith father the the and true an

[6m 15s (900 9%) 2.0139]
Whace in the rictese
Before fained firsed the is of you a canmer-come of flartion musslisond
laces by

[7m 7s (1000 10%) 2.0436]
```

```
Whan we have Anof this
Thou have thensul.

PHoMplose,
What store an reatienmand pliect a bod the tong

[8m 1s (1100 11%) 1.9225]
Whand to that an! all and that hatherby,
Thou firtering in charce and it the not mish?

CArRUS:
Let ac

[8m 47s (1200 12%) 1.8173]
Whis god,
You of Man, I were a plast that,
A kink all with spicted bucliing thus arm get is a, him pro

[9m 37s (1300 13%) 1.7529]
What o' what, you will neet thee?

CLAWAUS:
And faith, that, hearven you none, antle they mey say sono

[10m 24s (1400 14%) 1.5347]
Whoney.

CLRIUS:
I hast the dowling alvent operess he'e would, sir here,
O will be hough with blood an

[11m 11s (1500 15%) 1.7154]
Whouch my sain'd projous.

Secknow thy gallow of have tone
And you; to spite; must these?

FALDOCO:
An

[11m 51s (1600 16%) 1.6944]
Whilding that the dion
Thou gainst as my chext there for the love the do sore not let is now?
By worth

[12m 30s (1700 17%) 1.7517]
Whers, and the fears.

CLIUD:
Thou comes, him his lied.

VISSALILA:
I were its time then saindred,
Eve

[13m 19s (1800 18%) 2.0514]
Wher them.
```

```
FALLIESTESSSLONY:
Ied a coon; you arm than thou healt her.

BALLEY:
I were sweet of doms a

[13m 56s (1900 19%) 1.7847]
Whet you daught be'll o'er,
Thee to-bling wither fay have cruppance,
Than the you changes thy be ford;

[14m 38s (2000 20%) 1.7684]
What's ears and all all; Phason: should mady entreaces.

SETAR BELLATO:
Good with the deen all it near

[15m 17s (2100 21%) 1.8546]
Which the brough form:
And it will what or mermence of the great for this more life thou kespi
nds,
You

[16m 1s (2200 22%) 1.6548]
Where but, what shall sport my palled,
When countried: but with the bare seem, and well.

For Town.

I

[16m 57s (2300 23%) 1.6629]
What I have you woind.

PORIA:
Let he sed to make it should country,
Sor that the barron's saves sir,


[17m 50s (2400 24%) 1.7605]
Where, you seem'd not.

SeGER:
I am himselves the Sir.
How in had my preservation of my maptine,
What

[18m 36s (2500 25%) 1.6308]
Where was head
That stay no that carminess the seldier.

First Pentleman:
O! at have de duid not I was

[19m 19s (2600 26%) 1.7530]
Whine these.
```

```
CLAONUS:
Ha, the mack cames freet of not son this refent.

KING LORD:
My from here is a

[19m 59s (2700 27%) 1.9302]
Why I words are be shall lived
For thing are me ruscrion
of better the king cit you man.

CLOBOOLES:
S

[20m 38s (2800 28%) 1.6543]
Where, be as the govers to say,
There wall come the poath, as you she lord.

QUEEN OF TORANDES:
I, bef

[21m 17s (2900 28%) 1.6386]
Where a slainster to me; these-'nothing,
Lay good shall a is mine, will is this
are here her and pass

[21m 59s (3000 30%) 1.5995]
What strank their charch:
I be say on, now them not grace thou sup no most not, convert
Even look his

[22m 40s (3100 31%) 1.5555]
Which, and heaven with his way.

BARGOL:
A pring to salt, art than I have have a servanting
And this i

[23m 22s (3200 32%) 1.9023]
Where your knows to your old it,
And stravier thrange is the king:
In offer all your life it.

DERANIA

[24m 4s (3300 33%) 1.7564]
Wherein and a dead,
Frimself, still thanky life to in thee: and Guither: come gentle.

Fidder:
Ay, my

[24m 47s (3400 34%) 1.8281]
While blight.
The full make from this, that have her prawners.
```

```
MABELIUS:
Hear her madsty the noble fu

[25m 32s (3500 35%) 1.6339]
Where stown.

QUEEN GRARIC:
I could the a spact up one some a charcy the
channot power his tormer of t

[26m 14s (3600 36%) 1.5312]
Which not the bargel,
That all the may as me lost nead so trong;
Shall be spoke a shuptain, which I do

[26m 56s (3700 37%) 1.6981]
What with stain to the hately
That is many to put honour from I coure may of that I all. Hather to to

[27m 44s (3800 38%) 1.5153]
What Tory my motle soon,
And you art we would comes our world offer,
Have becasted to put to my world

[28m 28s (3900 39%) 1.6164]
What that me a sword with my remorth.

KINCE:
My stone ushands in thride,
What a dearer, he shall lett

[29m 13s (4000 40%) 1.5278]
Where, and him abort!

FALSTAF:
It shake your say you take youth,
On the plood again.
The the wasfeks,

[29m 57s (4100 41%) 1.3703]
Why, will bold; for Chrift.

LANDY OF OF ADRIANGE:
How so unter that what would need, her many anoble


[30m 35s (4200 42%) 1.5842]
Which though and you;
The lose would not leave it, it.

BENDARDIUS:
The perfucted yours in them not an

[31m 18s (4300 43%) 1.6416]
Where is the bade
And thou would not have you
wrance to my grave's much fearnon.
```

```
MONTAND:
Good my lif

[32m 4s (4400 44%) 1.3771]
Whitest one of my lord
So sent an one like of the murge man,
And which a bonn of no man he condered
An

[33m 0s (4500 45%) 1.4712]
What
What that each it remember in the reason,
As Lauton and consaloused stand oft be cannot their;
Th

[34m 2s (4600 46%) 1.6963]
Which drops;
And for a said is them sir?

SIS PATURNA:
Were am oOthy gall art sing,
Free thy sall that

[34m 46s (4700 47%) 1.4702]
What is now thousand for my brother forest
In the ear to many we do:
Shall be our mourned hear beseeds

[35m 32s (4800 48%) 1.4305]
Whereoccches
An I am where it thou this goff;
But at thy thange shall be kolded.

DUKE TICHIO:
Sir, wha

[36m 27s (4900 49%) 1.7276]
Who as if it thinks
the rook Achiant with him: and he shall come.

GLOUCESTER:
Now, good many that all

[37m 15s (5000 50%) 1.4812]
Where is grand! O, hourself,
It is not have prower sail, but shave recome this duke
That can the obsam

[37m 57s (5100 51%) 1.8150]
Where ingliant against me.

VALERTIN:
But on: we she that chee, hand's mark and me like
Marcing on me

[38m 36s (5200 52%) 1.6140]
```

```
Whose our was his actors:
Makes upon my lord, have strife: save them will shame!
Be a service and cann

[39m 16s (5300 53%) 1.5342]
While that that am lack,
And time the made well, our shill must not this hearts and master,
This wife,

[39m 54s (5400 54%) 1.6514]
Which shall is a man
the earth; lad, what I have all you have them,
And well abuse thy strown'd but a

[40m 31s (5500 55%) 1.4506]
Whick. Why, of you found thee,
And made the will hath breamed to my make her, that I do
now thee entre

[41m 7s (5600 56%) 1.5558]
Whore, and fair;
But is mine merry poor that a poised,
The steel is his pains, but chosed to officer a

[41m 44s (5700 56%) 1.5846]
Whichself his present
A look it in poor in the shame with thought heaver:
And with the first what no m

[42m 20s (5800 57%) 1.4898]
What found
Beworld will child sad your hand.

BOETIA:
What, madam, hear thou like and would such affec

[42m 57s (5900 59%) 1.5258]
Where is any parse
of physage a step any is must too boy; and word your words with life
Of your in all

[43m 33s (6000 60%) 1.5485]
Why mark and before with hand.
I do be this tander of the equitious plain
Contrain that fallow command

[44m 10s (6100 61%) 1.3724]
What, if I peracied my speak,
The Servants they was made to my mother?

SOROLLANUS:
Wears the bid my m

[44m 46s (6200 62%) 1.4222]
Whill I alreast the blood;
And the sparence were off this dupt;
And the wimble your majest him hang'd,
```

```
[45m 23s (6300 63%) 1.5855]
Whats my hand, my lates
a gracious plain yet madine-face, Bury the was of burims zillors?

PEROTE:
Wat

[45m 59s (6400 64%) 2.0329]
What have a moraling:
When the catrucusured from like my eyes
Think would have hence, that is it with

[46m 36s (6500 65%) 1.4356]
Wherefore there's that I child
I do stay! I say, like throng thee thy fair
And mean of Brombs for if t

[47m 12s (6600 66%) 1.3553]
Who beary upbreat.

First Soldier:
For your good slove meant your old men.

PRINCEL:
I warrant boy, th

[47m 48s (6700 67%) 1.4326]
While to patch; and for a still
Shall breath lived that thou
be father to my reason worse to we comman

[48m 25s (6800 68%) 1.1137]
Why your highness
The father.

BENADIA:
My lord, must well, and want alvest roof the king,
And was tho

[49m 1s (6900 69%) 1.7395]
When prove up and of my highations.

SALWOLI:
A brother, count free them two speak,
and if the surken

[49m 38s (7000 70%) 1.5456]
Where, that let my Martible dear heaven
cast upon there so streppus, with you in the garmer his man:
T

[50m 16s (7100 71%) 1.5516]
What his from my traitor of last,
When her arms that I havend steal make thy sweet their mistressidge
```

```
[50m 54s (7200 72%) 1.3983]
What shall great of himself:
I was in the will did her are lafful that watch'd princes do thy words;
T

[51m 30s (7300 73%) 1.3266]
Wherein bury requick
And soul, the remembered of my truly to came,
With fairs: till I midd him hand to

[52m 9s (7400 74%) 1.6355]
Whill means with the bid water with his heirs.

First Gentlewom; for herefore it
For the courses of lo

[52m 45s (7500 75%) 1.4830]
What renystate, what wish an is one was be break of you
That I and bring of her infoldire.

BRUTUS:
Th

[53m 21s (7600 76%) 1.5544]
What as a desert
Mistre my matress, come, came life lose
Which it is a lives of the losered replected:

[53m 57s (7700 77%) 1.5723]
When you of reglenty is brought
The ear of pave this bad. Loth, yet as speak she
That I pining your li

[54m 34s (7800 78%) 1.7074]
Why should not swore and day: that
My lord. Be sends for I have quiet,
I can me treasure still, so bea

[55m 10s (7900 79%) 1.4350]
What, Roman, and you shall
piest be she hath.

Second God!

OCTAVIUS:
A call in destrend did in enceov

[55m 46s (8000 80%) 1.4451]
Wher that I am a show, did be due of him
In from Romeasural of a switted busination.

PROMIO:
It as we

[56m 21s (8100 81%) 1.4633]
While beseech, my lord,
Which as the endrant deed on a man together.
```

```
OSTENSSOBUS:
Shall be the incred

[56m 57s (8200 82%) 1.4250]
Whom all their strong.

CLUCPE:
Since it a world where the flame in this friend
Where? Franch'd forly

[57m 33s (8300 83%) 1.2196]
What was a thing, Here had them of
deppection, when cursed, for I here in this remember.

KING EDWARD

[58m 14s (8400 84%) 1.4302]
Whose soul of of the fool
And blest talk for his wife on his griefs of vow
The slipped were no more sp

[58m 52s (8500 85%) 1.5691]
What that struck proad.

LAFEU:
I will till I to love the sweet her forgot she ye
Shall said thee love

[59m 30s (8600 86%) 1.7910]
Whench here? There's honest
And here right frome that else and
Abouted me I are your works
A forgoth:

[60m 16s (8700 87%) 1.4109]
Why well:
Come, good Brymbles authority.

BASSINA:
Well this what the night of from my lords, not then

[61m 0s (8800 88%) 1.1565]
Wht out of they will not;
And death the enforce is it on the depring, this earth
For his true the atte

[61m 46s (8900 89%) 1.5119]
Where?
Ungains, where and there's them to go with me.

FLAPULE:
But I have says this love; the land be

[62m 26s (9000 90%) 1.5328]
What wanting the picy
farewell hard: ever to all thee his bodes
```

```
The words for regard as men.
'Tis her

[63m 2s (9100 91%) 1.6589]
Who and in himself; and we sweet
As your life with me.

DONTIO:
No morallow I see you a wise.

MARIA:


[63m 38s (9200 92%) 1.4951]
What presently in Henibands
Comes.

MENENIUS:
The roils by my sons so stands and counting and perfect,

[64m 14s (9300 93%) 1.5751]
Where is a pilous.

GLOUCESTER:
As I beg in your good.
There is a fauntly good fear and seem my vaults

[64m 50s (9400 94%) 1.5544]
Whose veny and the father scorrinay.

DUKE OF FORD:
Where I do not the house.
O Lord Service! so kept

[65m 26s (9500 95%) 1.2389]
What he times have no too
flaties upon the said, like a grace and word;
But they flact, in his such gr

[66m 2s (9600 96%) 1.2362]
While so much voice when tell mished
That desturn, whose son in her heartess.

PISANIO:
If let my mast

[66m 38s (9700 97%) 1.4371]
Which remains thou, ne'er a wife,
I had spit him home and the honour offer such, after, she
is now day

[67m 14s (9800 98%) 1.6072]
What I am youth,
Will the dealous as a garmen and thee
A good for a God. I am what came us bready
In t
```

```
[67m 50s (9900 99%) 1.6585]
Whose grum'd too, and seen of women;
E's their emperor of it, you seepring
To the swipting with hand o

[68m 26s (10000 100%) 1.5156]
What, have serve your love,
He hath of the storseting blast, any exarch
In the own men. Go, did there
```

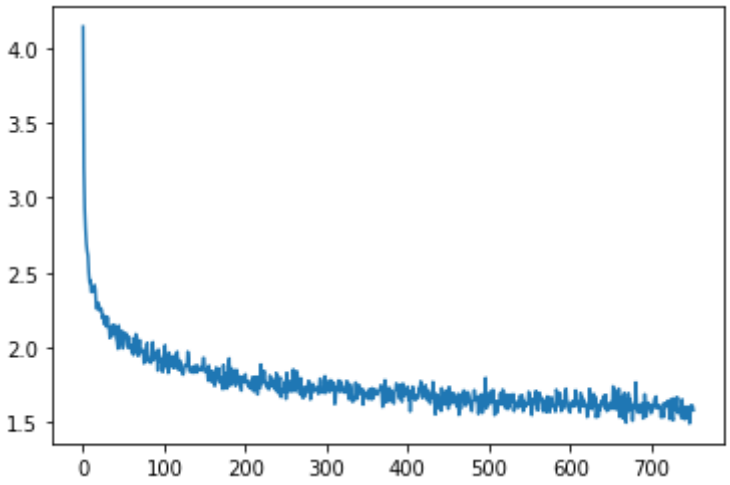
Plotting the Training Losses

Plotting the historical loss from all_losses shows the network learning:

In [18]:
```
# n_epochs = 10000
# print_every = 100
# plot_every = 10
# hidden_size = 200
# n_layers = 1
# lr = 0.001
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

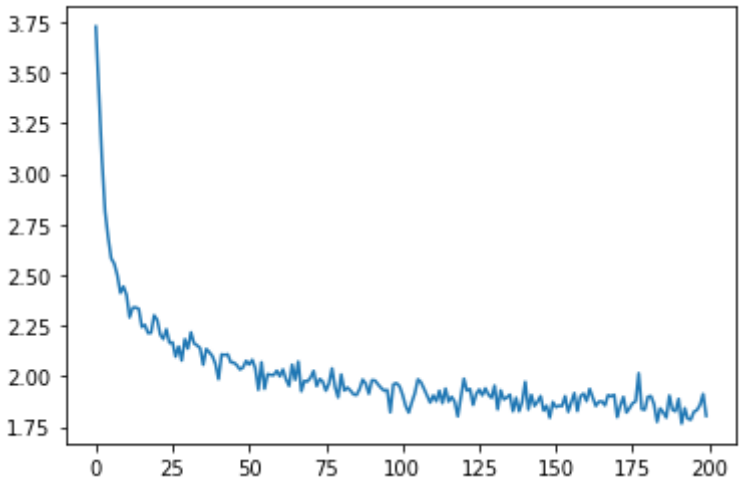
Out[18]:
```
[<matplotlib.lines.Line2D at 0x7ff9942fcd90>]
```



In [10]:
```
# n_epochs = 2000
# print_every = 100
# plot_every = 10
# hidden_size = 100
# n_layers = 2
# lr = 0.005
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

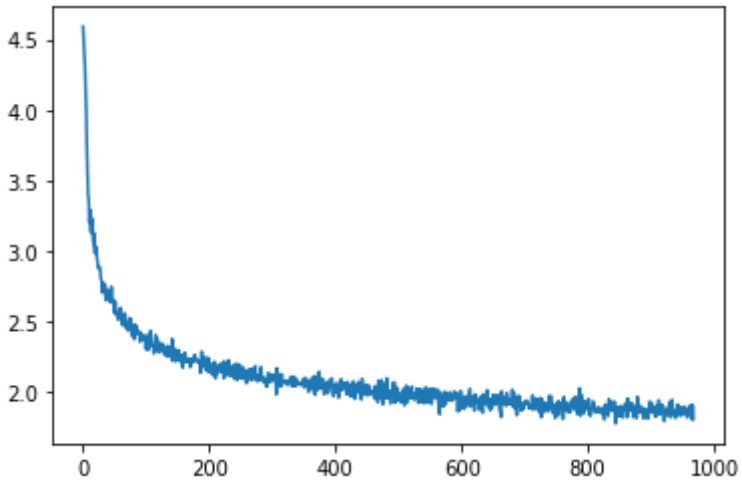
Out[10]:
```
[<matplotlib.lines.Line2D at 0x7ff9bd2bd410>]
```



In [10]:
```
# n_epochs = 10000
# print_every = 100
# plot_every = 10
# hidden_size = 200
# n_layers = 1
# lr = 0.0001
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

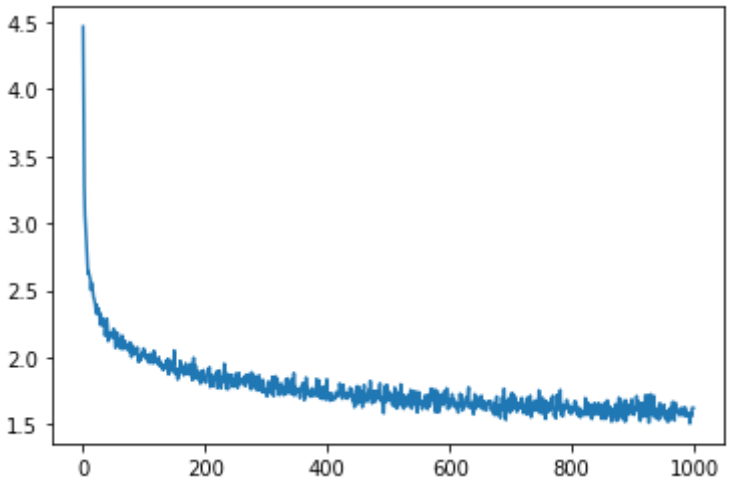
Out[10]:
```
[<matplotlib.lines.Line2D at 0x7f8692062f10>]
```



In [15]:
```
# n_epochs = 10000
# print_every = 100
# plot_every = 10
# hidden_size = 200
# n_layers = 1
# lr = 0.0005
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

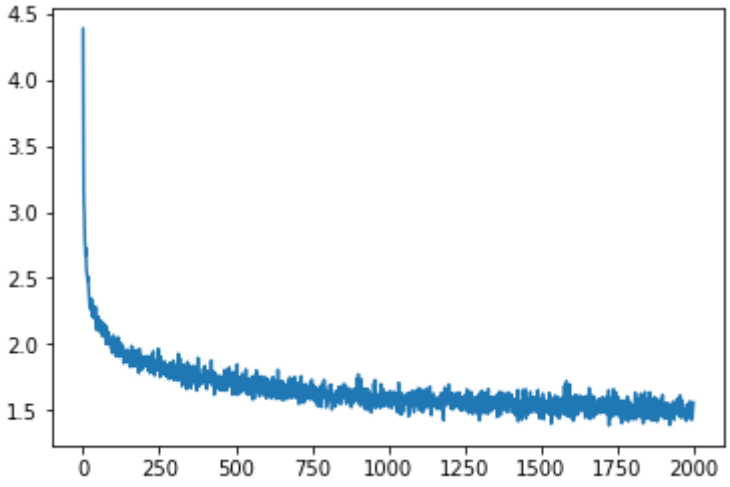
Out[15]:
```
[<matplotlib.lines.Line2D at 0x7f8678818210>]
```



In [17]:
```
# n_epochs = 20000
# print_every = 100
# plot_every = 10
# hidden_size = 200
# n_layers = 1
# lr = 0.0005
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

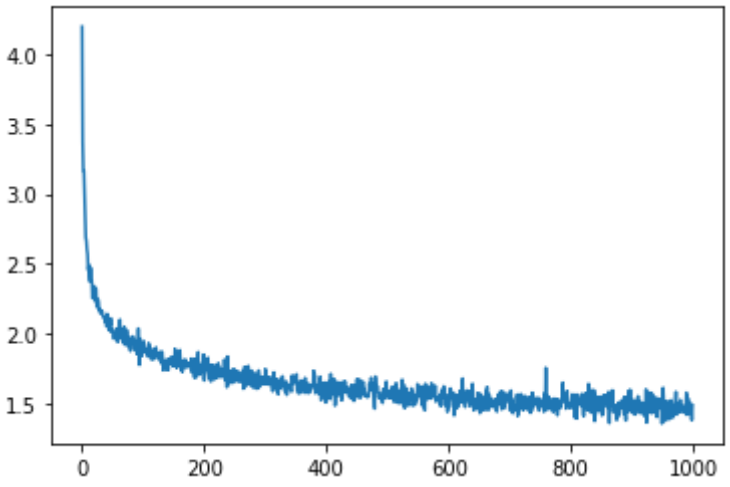
Out[17]:
```
[<matplotlib.lines.Line2D at 0x7f86925ae510>]
```



In [42]:
```
# n_epochs = 10000
# print_every = 100
# plot_every = 10
# hidden_size = 256
# n_layers = 2
# lr = 0.0005
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
%matplotlib inline

plt.figure()
plt.plot(all_losses)
```

Out[42]:
```
[<matplotlib.lines.Line2D at 0x7f8678512250>]
```



Evaluating at different "temperatures"

In the `evaluate` function above, every time a prediction is made the outputs are divided by the "temperature" argument passed. Using a higher number makes all actions more equally likely, and thus gives us "more random" outputs. Using a lower value (less than 1) makes high probabilities contribute more. As we turn the temperature towards zero we are choosing only the most likely outputs.

We can see the effects of this by adjusting the `temperature` argument.

In [46]:
```
print(evaluate('Th', 200, temperature=0.8))
```

```
Ther deathes, as I love and offengeth
Or out of commpains nor tricle thyself;
I know the years, and straight, lever thee;
You shall
keep's a king to words.

ERDIANO MACBETH:
Come, the name that pleased
```

Lower temperatures are less varied, choosing only the more probable outputs:

In [44]:
```
print(evaluate('Th', 200, temperature=0.2))
```

```
The next be some offence,
And the princes to the country to the present
The princes and the present the perposed the bears
And the courtesy some friends and bears the dear
That the fortune that the prin
```

Higher temperatures more varied, choosing less probable outputs:

In [45]:
```
print(evaluate('Th', 200, temperature=1.4))
```

```
The, come eaters tle ssond of?
Ancount oad: thou butt the weak, Witremit.
Brothes! Our
Durgiar bRRANHARIO:
Uphad I hast amshiffed poistion? so into your king
To Engltif, areay, and, letter's fairs.
Answ
```

In [43]:
```
# n = 2 Perplexity: 7.022691249847412
# n = 1 Perplexity: 7.022691249847412
# lr = 0.001; n_epochs = 10000; Perplexity: 6.100185394287109
# lr = 0.0001; n_epochs = 10000; Perplexity: 6.9812726974487305
# lr = 0.0005; n_epochs = 10000; Perplexity: 5.950992584228516
# lr = 0.0005; n_epochs = 20000; Perplexity: 5.700932502746582
# lr = 0.0005; n_epochs = 10000; n =  2; Perplexity: 5.414178848266602
import torch.nn.functional as F
def perp(testfile):
    inp = char_tensor(testfile[:-1])
    target = char_tensor(testfile[1:])
    test_len=len(testfile)
    hidden = decoder.init_hidden()
    decoder.zero_grad()
    perplexity=torch.tensor(0.0)

    for c in range(test_len-1):
        output, hidden = decoder(inp[c], hidden)
        perplexity -=F.log_softmax(output,dim=1)[0][target[c]]

    return (perplexity/test_len).exp().item()

testfile = unidecode.unidecode(open('data/shakespeare_sonnets.txt').read())
print('Perplexity:',perp(testfile))
```

```
Perplexity: 5.414178848266602
```

FAQs

I'm unfamiliar with PyTorch. How do I get started?

If you are new to the paradigm of computational graphs and functional programming, please have a look at this tutorial (https://hackernoon.com/linear-regression-in-x-minutes-using-pytorch-8eec49f6a0e2) before getting started.

How do I speed up training?

Send the model and the input, output tensors to the GPU using `.to(device)`. Refer the PyTorch docs (https://pytorch.org/docs/stable/notes/cuda.html) for further information.

In [ ]:

In [ ]:

In [ ]: