Assignment2

December 10, 2022

Alexander Fok 308669944

Avi Dvir 204423735

Gal Cohen 204675805

1 Assignment 2: Word Prediction

Deadline: Sunday, December 11th, by 8pm.

Submission: Submit a PDF export of the completed notebook as well as the ipynb file.

In this assignment, we will make a neural network that can predict the next word in a sentence given the previous three.
In doing this prediction task, our neural networks will learn about *words* and about how to represent words. We'll explore the *vector representations* of words that our model produces, and analyze these representations.

You may modify the starter code as you see fit, including changing the signatures of functions and adding/removing helper functions. However, please make sure that you properly explain what you are doing and why.

[1]:
```
import pandas
import numpy as np
import matplotlib.pyplot as plt
import collections

import torch
import torch.nn as nn
import torch.optim as optim
```

1.1 Question 1. Data (18%)

With any machine learning problem, the first thing that we would want to do is to get an intuitive understanding of what our data looks like. Download the file `raw_sentences.txt` from the course page on Moodle and upload it to Google Drive. Then, mount Google Drive from your Google Colab notebook:

[2]:
```
from google.colab import drive
drive.mount('/content/gdrive',force_remount=True)
```

```
Mounted at /content/gdrive
```

Find the path to `raw_sentences.txt`:

[3]:
```
file_path = '/content/gdrive/My Drive/IntroDeepLearning2022Data/raw_sentences.
↪txt'
```

The following code reads the sentences in our file, split each sentence into its individual words, and stores the sentences (list of words) in the variable `sentences`.

[4]:
```
sentences = []
for line in open(file_path):
    words = line.split()
    sentence = [word.lower() for word in words]
    sentences.append(sentence)
```

There are 97,162 sentences in total, and these sentences are composed of 250 distinct words.

[5]:
```
vocab = set([w for s in sentences for w in s])
print(len(sentences)) # 97162
print(len(vocab)) # 250
```

```
97162
250
```

We'll separate our data into training, validation, and test. We'll use '10,000 sentences for test, 10,000 for validation, and the rest for training.

[6]:
```
test, valid, train = sentences[:10000], sentences[10000:20000], sentences[20000:
↪]
```

1.1.1 Part (a) -- 3%

Display 10 sentences in the training set. **Explain** how punctuations are treated in our word representation, and how words with apostrophes are represented.

[58]:
```
import random
random.sample(train,10)
```

[58]:
```
[['no', 'one', 'to', 'see', '.'],
 ['those', 'days', 'may', 'be', 'over', '.'],
 ['i', 'did', 'what', 'she', 'said', '.'],
 ['but', 'we', 'are', 'a', 'good', 'school', '.'],
 ['what', 'good', 'is', 'it', 'going', 'to', 'be', '?'],
 ['and', 'for', 'a', 'few', 'days', 'i', 'was', '.'],
```

```
 ['but', 'this', 'is', 'the', 'game', '.'],
 ['i',
  'do',
  'nt',
  'know',
  'if',
  'people',
  'will',
  'like',
  'it',
  ',',
  'but',
  'we',
  'may',
  'have',
  'to',
  'do',
  'it',
  '.'],
 ['what', 'are', 'we', '?'],
 ['i', 'like', 'people', ',', 'she', 'said', '.']]
```

Write your answers here: Punctuation characters are treated as word delimiters in our dictionary. So words with apostrophes are splited and represented as separate words. For example: "do'nt": "do", "nt".

1.1.2 Part (b) -- 4%

Print the 10 most common words in the vocabulary and how often does each of these words appear in the training sentences. Express the second quantity as a percentage (i.e. number of occurences of the word / total number of words in the training set).

These are useful quantities to compute, because one of the first things a machine learning model will learn is to predict the **most common** class. Getting a sense of the distribution of our data will help you understand our model's behaviour.

You can use Python's `collections.Counter` class if you would like to.

[59]:
```
voca = [word for sentence in train for word in sentence]
size_voca = len(voca)
voca_count = dict(collections.Counter(voca))
voca_prec = dict((i, 100*(j/size_voca)) for (i,j) in voca_count.items())
sorted_voca = sorted(voca_prec.items(), key = lambda item: -item[1])
for k in sorted_voca[: 10] :
  print(f"{k[0]} >>> " + "{:.2f}".format(k[1]) + " %")
```

```
. >>> 10.70 %
it >>> 3.85 %
```

```
, >>> 3.25 %
i >>> 2.94 %
do >>> 2.69 %
to >>> 2.58 %
nt >>> 2.16 %
? >>> 2.14 %
the >>> 2.09 %
's >>> 2.09 %
```

1.1.3 Part (c) -- 11%

Our neural network will take as input three words and predict the next one. Therefore, we need our data set to be comprised of seuqnces of four consecutive words in a sentence, referred to as *4grams*.

Complete the helper functions `convert_words_to_indices` and `generate_4grams`, so that the function `process_data` will take a list of sentences (i.e. list of list of words), and generate an $N \times 4$ numpy matrix containing indices of 4 words that appear next to each other, where N is the number of 4grams (sequences of 4 words appearing one after the other) that can be found in the complete list of sentences. Examples of how these functions should operate are detailed in the code below.

You can use the defined `vocab`, `vocab_itos`, and `vocab_stoi` in your code.

[9]:
```
# A list of all the words in the data set. We will assign a unique
# identifier for each of these words.
vocab = sorted(list(set([w for s in train for w in s])))
# A mapping of index => word (string)
vocab_itos = dict(enumerate(vocab))
# A mapping of word => its index
vocab_stoi = {word:index for index, word in vocab_itos.items()}

def convert_words_to_indices(sents):
    """
    This function takes a list of sentences (list of list of words)
    and returns a new list with the same structure, but where each word
    is replaced by its index in `vocab_stoi`.

    Example:
    >>> convert_words_to_indices([['one', 'in', 'five', 'are', 'over', 'here'],␣
 ↪['other', 'one', 'since', 'yesterday'], ['you']])
    [[148, 98, 70, 23, 154, 89], [151, 148, 181, 246], [248]]
    """

    #Write your code here
    return [[vocab_stoi[word] for word in sent] for sent in sents]

def generate_4grams(seqs):
    """
```

```
    This function takes a list of sentences (list of lists) and returns
    a new list containing the 4-grams (four consequentively occuring words)
    that appear in the sentences. Note that a unique 4-gram can appear multiple
    times, one per each time that the 4-gram appears in the data parameter␣
↪`seqs`.

    Example:

    >>> generate_4grams([[148, 98, 70, 23, 154, 89], [151, 148, 181, 246],␣
↪[248]])
    [[148, 98, 70, 23], [98, 70, 23, 154], [70, 23, 154, 89], [151, 148, 181,␣
↪246]]
    >>> generate_4grams([[1, 1, 1, 1, 1]])
    [[1, 1, 1, 1], [1, 1, 1, 1]]
    """
    # Write your code here
    indices = []
    for sent in seqs:
      sent_idx = 0
      while sent_idx + 4 <= len(sent):
        t_4gram = sent[sent_idx:sent_idx + 4]
        indices.append(t_4gram)
        sent_idx += 1
    return indices;

def process_data(sents):
    """
    This function takes a list of sentences (list of lists), and generates an
    numpy matrix with shape [N, 4] containing indices of words in 4-grams.
    """
    indices = convert_words_to_indices(sents)
    fourgrams = generate_4grams(indices)
    return np.array(fourgrams)

# We can now generate our data which will be used to train and test the network
train4grams = process_data(train)
valid4grams = process_data(valid)
test4grams = process_data(test)
```

1.2 Question 2. A Multi-Layer Perceptron (44%)

In this section, we will build a two-layer multi-layer perceptron. Our model will look like this:

Since the sentences in the data are comprised of 250 distinct words, our task boils down to claissfication where the label space $\mathcal{S}$ is of cardinality $|\mathcal{S}| = 250$ while our input, which is comprised of a combination of three words, is treated as a vector of size 750×1 (i.e., the concatanation of three

one-hot 250×1 vectors).

The following function `get_batch` will take as input the whole dataset and output a single batch for the training. The output size of the batch is explained below.

Implement yourself a function `make_onehot` which takes the data in index notation and output it in a onehot notation.

Start by reviewing the helper function, which is given to you:

[10]:
```
def make_onehot(data):
    """
    Convert one batch of data in the index notation into its corresponding␣
 ↪onehot
    notation. Remember, the function should work for both xt and st.

    input - vector with shape D (1D or 2D)
    output - vector with shape (D,250)
    """

    # Write your code here
    n_class = 250
    return np.eye(n_class)[data]
def get_batch(data, range_min, range_max, onehot=True):
    """
    Convert one batch of data in the form of 4-grams into input and output
    data and return the training data (xt, st) where:
     - `xt` is an numpy array of one-hot vectors of shape [batch_size, 3, 250]
     - `st` is either
            - a numpy array of shape [batch_size, 250] if onehot is True,
            - a numpy array of shape [batch_size] containing indicies otherwise

    Preconditions:
     - `data` is a numpy array of shape [N, 4] produced by a call
        to `process_data`
     - range_max > range_min
    """
    xt = data[range_min:range_max, :3]
    xt = make_onehot(xt)
    xt = xt.reshape(-1, 3 , 250)

    st = data[range_min:range_max, 3]
    if onehot:
        st = make_onehot(st).reshape(-1, 250)
    return xt, st
```

[11]:
```
# Test get_batch
test1 = sentences[:3]
#print(test1)
```

```
data = process_data(test1)
print(data)
range_min = 0
range_max = 4
xt, st = get_batch(data, range_min, range_max, onehot=True)
#xt, st = get_batch(data, range_min, range_max, onehot=False)
# test make_onehot
print(f'xt.shape {xt.shape}')
xt_flat = xt.reshape(-1, 250)
print(f'xt_flat.shape {xt_flat.shape}')
for idx, val in enumerate(xt_flat):
    for idx2, val_1 in enumerate(val):
      if val_1 > 0:
        print(f'idx: {idx2}, ')
print(f'st.shape {st.shape}')
for idx, val in enumerate(st):
    for idx2, val_1 in enumerate(val):
      if val_1 > 0:
        print(f'idx: {idx2}, ')
```

```
[[138   3  87 171]
 [  3  87 171 140]
 [ 87 171 140   6]
 [ 20 230  55  87]
 [230  55  87  57]
 [ 55  87  57   9]
 [193 125   1 197]
 [125   1 197   6]]
xt.shape (4, 3, 250)
xt_flat.shape (12, 250)
idx: 138,
idx: 3,
idx: 87,
idx: 3,
idx: 87,
idx: 171,
idx: 87,
idx: 171,
idx: 140,
idx: 20,
idx: 230,
idx: 55,
st.shape (4, 250)
idx: 171,
idx: 140,
idx: 6,
idx: 87,
```

1.2.1 Part (a) -- 8%

We build the model in PyTorch. Since PyTorch uses automatic differentiation, we only need to write the *forward pass* of our model.

Complete the `forward` function below:

[12]:
```
class PyTorchMLP(nn.Module):
    def __init__(self, num_hidden=400):
        super(PyTorchMLP, self).__init__()
        self.layer1 = nn.Linear(750, num_hidden)
        self.layer2 = nn.Linear(num_hidden, 250)
        self.num_hidden = num_hidden

    def forward(self, inp):
        inp = inp.reshape([-1, 750])
        # TODO: complete this function
        # Note that we will be using the nn.CrossEntropyLoss(), which computes␣
↪the softmax operation internally, as loss criterion
        x = self.layer1(inp)
        x = self.layer2(x)
        return x
```

1.2.2 Part (b) -- 10%

We next train the PyTorch model using the Adam optimizer and the cross entropy loss.

Complete the function `run_pytorch_gradient_descent`, and use it to train your PyTorch MLP model.

Obtain a training accuracy of at least 35% while changing only the hyperparameters of the train function.

Plot the learning curve using the `plot_learning_curve` function provided to you, and include your plot in your PDF submission.

[13]:
```
def estimate_accuracy_torch(model, data, batch_size=5000, max_N=100000):
    """
    Estimate the accuracy of the model on the data. To reduce
    computation time, use at most `max_N` elements of `data` to
    produce the estimate.
    """
    correct = 0
    N = 0
    for i in range(0, data.shape[0], batch_size):
        # get a batch of data
        xt, st = get_batch(data, i, i + batch_size, onehot=False)

        # forward pass prediction
```

```
        y = model(torch.Tensor(xt))
        y = y.detach().numpy() # convert the PyTorch tensor => numpy array
        pred = np.argmax(y, axis=1)
        correct += np.sum(pred == st)
        N += st.shape[0]

        if N > max_N:
            break
    return correct / N

def run_pytorch_gradient_descent(model,
                                 train_data=train4grams,
                                 validation_data=valid4grams,
                                 batch_size=100,
                                 learning_rate=0.001,
                                 weight_decay=0,
                                 max_iters=1000,
                                 checkpoint_path=None):
    """
    Train the PyTorch model on the dataset `train_data`, reporting
    the validation accuracy on `validation_data`, for `max_iters`
    iteration.

    If you want to **checkpoint** your model weights (i.e. save the
    model weights to Google Drive), then the parameter
    `checkpoint_path` should be a string path with `{}` to be replaced
    by the iteration count:

    For example, calling

    >>> run_pytorch_gradient_descent(model, ...,
            checkpoint_path = '/content/gdrive/My Drive/Intro_to_Deep_Learning/
↪mlp/ckpt-{}.pk')

    will save the model parameters in Google Drive every 500 iterations.
    You will have to make sure that the path exists (i.e. you'll need to create
    the folder Intro_to_Deep_Learning, mlp, etc...). Your Google Drive will be␣
↪populated with files:

    - /content/gdrive/My Drive/Intro_to_Deep_Learning/mlp/ckpt-500.pk
    - /content/gdrive/My Drive/Intro_to_Deep_Learning/mlp/ckpt-1000.pk
    - ...

    To load the weights at a later time, you can run:

    >>> model.load_state_dict(torch.load('/content/gdrive/My Drive/
↪Intro_to_Deep_Learning/mlp/ckpt-500.pk'))
```

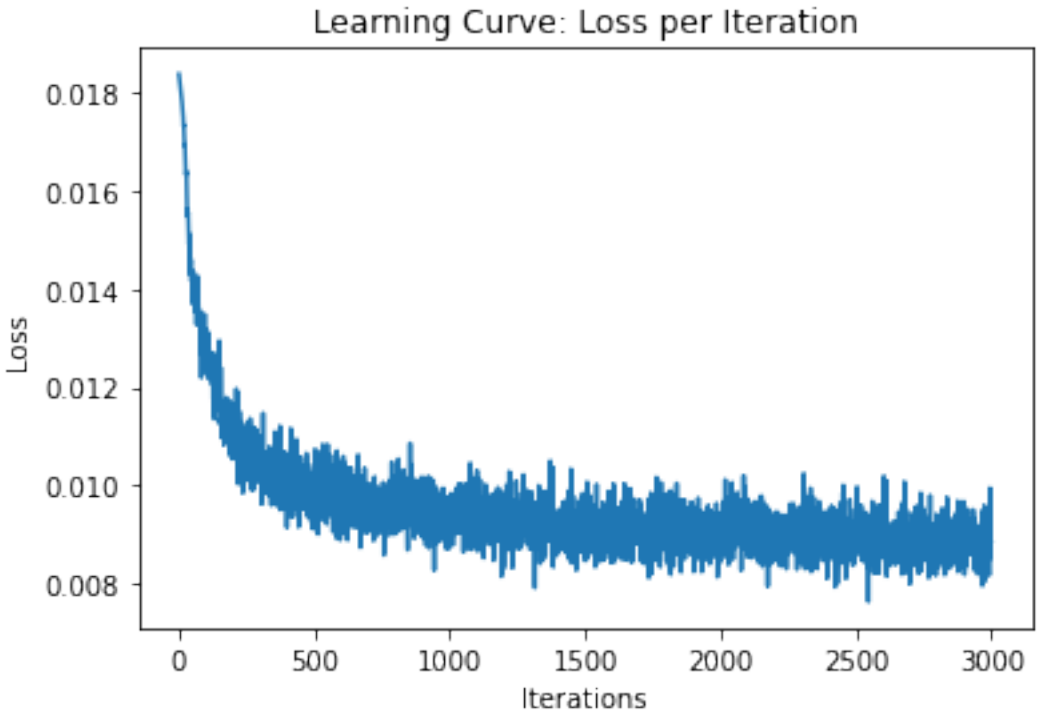
```
    This function returns the training loss, and the training/validation␣
↪accuracy,
    which we can use to plot the learning curve.
    """
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(),
                           lr=learning_rate,
                           weight_decay=weight_decay)

    iters, losses = [], []
    iters_sub, train_accs, val_accs  = [], [] ,[]

    n = 0 # the number of iterations
    while True:
        for i in range(0, train_data.shape[0], batch_size):
            if (i + batch_size) > train_data.shape[0]:
                break

            # get the input and targets of a minibatch
            xt, st = get_batch(train_data, i, i + batch_size, onehot=False)

            # convert from numpy arrays to PyTorch tensors
            xt = torch.Tensor(xt)
            st = torch.Tensor(st).long()

            # zs = ...                  # compute prediction logit
            output = model(xt)
            # loss =                    # compute the total loss
            loss = criterion(output, st)
            # Backward pass to compute the gradient
            # of loss w.r.t our learnable params.
            # ...                       # compute updates for each parameter
            loss.backward()

            # ...                       # make the updates for each parameter
            # Update params
            optimizer.step()
            # ...                       # a clean up step for PyTorch
            # zero the gradients before running
            # the backward pass.
            optimizer.zero_grad()

            # save the current training information
            iters.append(n)
            losses.append(float(loss)/batch_size)  # compute *average* loss
```

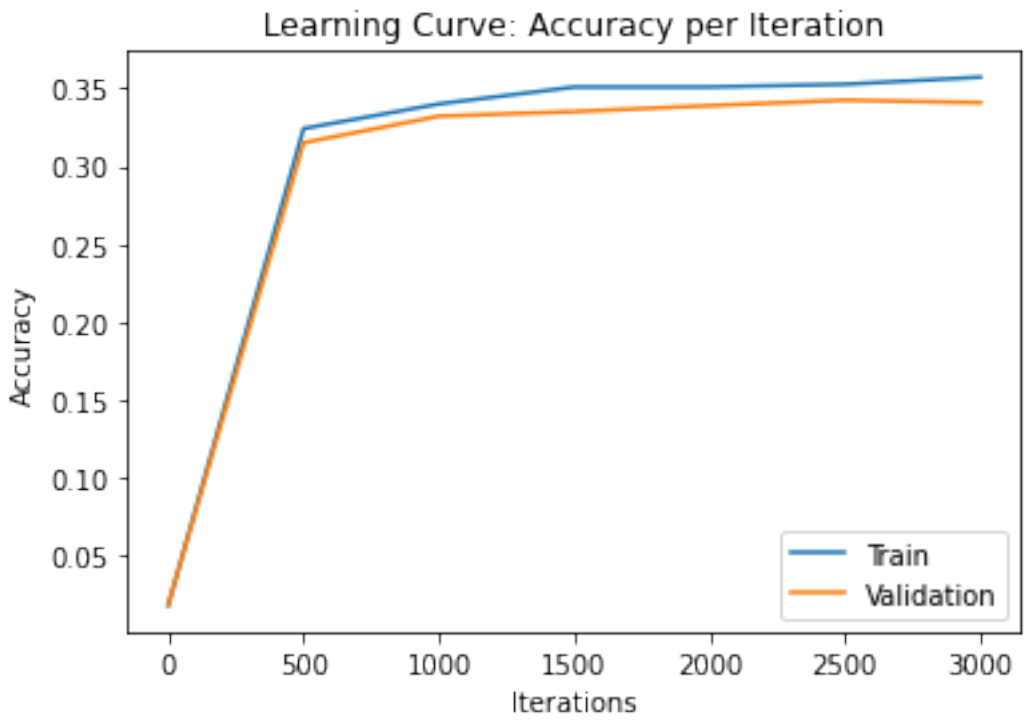
```
            if n % 500 == 0:
                iters_sub.append(n)
                train_cost = float(loss.detach().numpy())
                print(f'estimate_accuracy_torch')
                train_acc = estimate_accuracy_torch(model, train_data)
                train_accs.append(train_acc)
                val_acc = estimate_accuracy_torch(model, validation_data)
                val_accs.append(val_acc)
                print("Iter %d. [Val Acc %.0f%%] [Train Acc %.0f%%, Loss %f]" %␣
 ↪(
                      n, val_acc * 100, train_acc * 100, train_cost))

                if (checkpoint_path is not None) and n > 0:
                    torch.save(model.state_dict(), checkpoint_path.format(n))

            # increment the iteration number
            n += 1

            if n > max_iters:
                return iters, losses, iters_sub, train_accs, val_accs


def plot_learning_curve(iters, losses, iters_sub, train_accs, val_accs):
    """
    Plot the learning curve.
    """
    plt.title("Learning Curve: Loss per Iteration")
    plt.plot(iters, losses, label="Train")
    plt.xlabel("Iterations")
    plt.ylabel("Loss")
    plt.show()

    plt.title("Learning Curve: Accuracy per Iteration")
    plt.plot(iters_sub, train_accs, label="Train")
    plt.plot(iters_sub, val_accs, label="Validation")
    plt.xlabel("Iterations")
    plt.ylabel("Accuracy")
    plt.legend(loc='best')
    plt.show()
```

[14]:
```
pytorch_mlp = PyTorchMLP()
checkpoint_path = '/content/gdrive/My Drive/IntroDeepLearning2022Data/mlp/
 ↪ckpt-{}.pk'
learning_curve_info = run_pytorch_gradient_descent(model=pytorch_mlp,
                                   train_data=train4grams,
                                   validation_data=valid4grams,
                                   batch_size=300,
```

```
                                    learning_rate=0.001,
                                    weight_decay=0,
                                    max_iters=3000,
                                    checkpoint_path=checkpoint_path )
plot_learning_curve(*learning_curve_info)
```

```
estimate_accuracy_torch
Iter 0. [Val Acc 2%] [Train Acc 2%, Loss 5.515302]
estimate_accuracy_torch
Iter 500. [Val Acc 31%] [Train Acc 32%, Loss 2.973987]
estimate_accuracy_torch
Iter 1000. [Val Acc 33%] [Train Acc 34%, Loss 2.775578]
estimate_accuracy_torch
Iter 1500. [Val Acc 33%] [Train Acc 35%, Loss 2.829674]
estimate_accuracy_torch
Iter 2000. [Val Acc 34%] [Train Acc 35%, Loss 2.670304]
estimate_accuracy_torch
Iter 2500. [Val Acc 34%] [Train Acc 35%, Loss 2.596553]
estimate_accuracy_torch
Iter 3000. [Val Acc 34%] [Train Acc 36%, Loss 2.655112]
```





1.2.3 Part (c) -- 10%

Write a function `make_prediction` that takes as parameters a PyTorchMLP model and sentence (a list of words), and produces a prediction for the next word in the sentence.

[15]:
```
def make_prediction_torch(model, sentence):
    """
    Use the model to make a prediction for the next word in the
    sentence using the last 3 words (sentence[:-3]). You may assume
    that len(sentence) >= 3 and that `model` is an instance of
    PYTorchMLP.

    This function should return the next word, represented as a string.

    Example call:
    >>> make_prediction_torch(pytorch_mlp, ['you', 'are', 'a'])
    """
    global vocab_stoi, vocab_itos
    #  Write your code here
    # Prepare data
    sent_i = [vocab_stoi[word] for word in sentence]
    xt_oh = make_onehot(sent_i)
```

```
    xt = torch.Tensor(xt_oh)
    y_hat = model(xt)
    # retrieve numpy array from torch
    pred = y_hat.detach().numpy()
    pred_idx = np.argmax(pred, axis=1)
    word_idx = pred_idx[0]
    p_word = vocab_itos[word_idx]
    print(f'word_idx {word_idx}, p_word {p_word}')

make_prediction_torch(pytorch_mlp, np.array(['you', 'are', 'a']))
```

```
word_idx 81, p_word good
```

1.2.4 Part (d) -- 10%

Use your code to predict what the next word should be in each of the following sentences:

- ”You are a”
- ”few companies show”
- ”There are no”
- ”yesterday i was”
- ”the game had”
- ”yesterday the federal”

Do your predictions make sense?

In many cases where you overfit the model can either output the same results for all inputs or just memorize the dataset.

Print the output for all of these sentences and **Write** below if you encounter these effects or something else which indicates overfitting, if you do train again with better hyperparameters.

[16]:
```
# Write your code here
sentences_set = ["You are a",
                 "few companies show",
                 "There are no",
                 "yesterday i was",
                 "the game had",
                 "yesterday the federal"]
for sent in sentences_set:
    words = sent.split()
    sentence = [word.lower() for word in words]
    make_prediction_torch(pytorch_mlp, np.array(sentence))
```

```
word_idx 81, p_word good
word_idx 6, p_word .
word_idx 151, p_word other
word_idx 141, p_word nt
```

```
word_idx 208, p_word to
word_idx 82, p_word government
```

Write your answers here: When we had errors in primitive functions (one_hot_encoder), the training process didn't converge and stack on ~17% Accuracy for both - training and validation set. It means that the model just memorized the training samples - was in overfitted. Once we fixed the code issues, the model converged to 36% accuracy in 3,000 iterations.

1.2.5 Part (e) -- 6%

Report the test accuracy of your model

[17]:
```
# Write your code here
print(f"model's test accuracy: {estimate_accuracy_torch(pytorch_mlp,␣
 ↪test4grams)*100:.3f}%")
```

```
model's test accuracy: 34.285%
```

1.3 Question 3. Learning Word Embeddings (24 %)

In this section, we will build a slightly different model with a different architecture. In particular, we will first compute a lower-dimensional *representation* of the three words, before using a multi-layer perceptron.

Our model will look like this:

This model has 3 layers instead of 2, but the first layer of the network is **not** fully-connected. Instead, we compute the representations of each of the three words **separately**. In addition, the first layer of the network will not use any biases. The reason for this will be clear in question 4.

1.3.1 Part (a) -- 10%

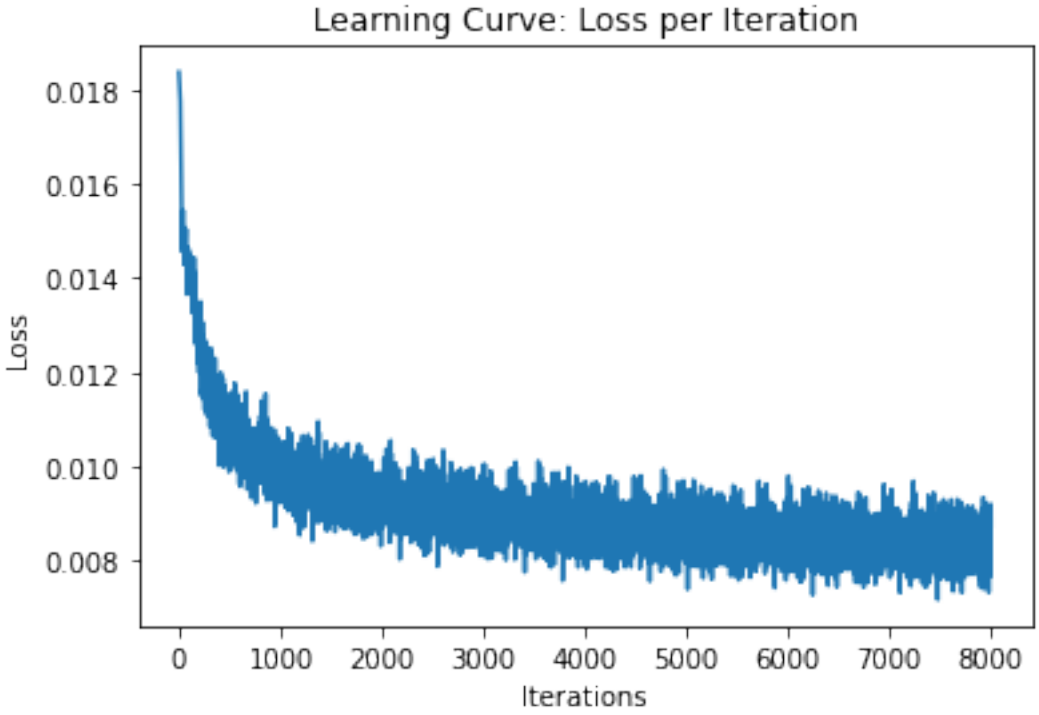
The PyTorch model is implemented for you. Use `run_pytorch_gradient_descent` to train your PyTorch MLP model to obtain a training accuracy of at least 38%. Plot the learning curve using the `plot_learning_curve` function provided to you, and include your plot in your PDF submission.

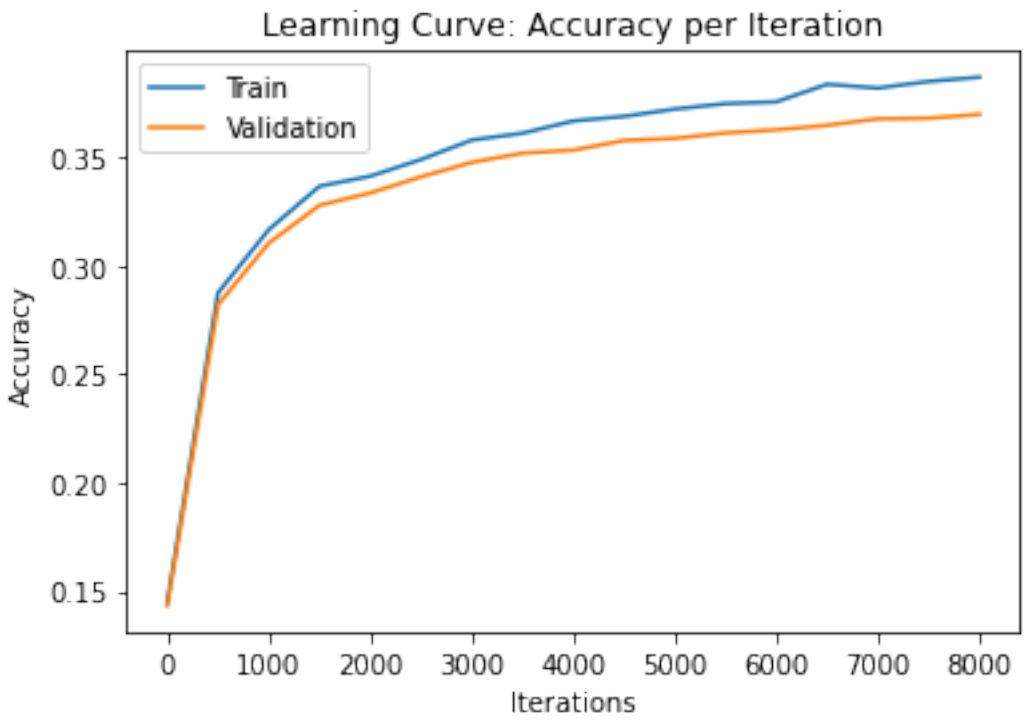
[18]:
```
class PyTorchWordEmb(nn.Module):
    def __init__(self, emb_size=100, num_hidden=300, vocab_size=250):
        super(PyTorchWordEmb, self).__init__()
        self.word_emb_layer = nn.Linear(vocab_size, emb_size, bias=False)
        self.fc_layer1 = nn.Linear(emb_size * 3, num_hidden)
        self.fc_layer2 = nn.Linear(num_hidden, 250)
        self.num_hidden = num_hidden
        self.emb_size = emb_size
    def forward(self, inp):
        embeddings = torch.relu(self.word_emb_layer(inp))
        embeddings = embeddings.reshape([-1, self.emb_size * 3])
```

```
        hidden = torch.relu(self.fc_layer1(embeddings))
        return self.fc_layer2(hidden)

pytorch_wordemb= PyTorchWordEmb()
checkpoint_path = '/content/gdrive/My Drive/IntroDeepLearning2022Data/mlp/
 ↪ckpt-{}.pk'
result = run_pytorch_gradient_descent(pytorch_wordemb,
                                      train_data=train4grams,
                                      validation_data=valid4grams,
                                      batch_size=300,
                                      learning_rate=0.001,
                                      weight_decay=0,
                                      max_iters=8000,
                                      checkpoint_path=checkpoint_path)

plot_learning_curve(*result)
```

```
estimate_accuracy_torch
Iter 0. [Val Acc 14%] [Train Acc 14%, Loss 5.512062]
estimate_accuracy_torch
Iter 500. [Val Acc 28%] [Train Acc 29%, Loss 3.225503]
estimate_accuracy_torch
Iter 1000. [Val Acc 31%] [Train Acc 32%, Loss 2.958367]
estimate_accuracy_torch
Iter 1500. [Val Acc 33%] [Train Acc 34%, Loss 2.966555]
estimate_accuracy_torch
Iter 2000. [Val Acc 33%] [Train Acc 34%, Loss 2.790647]
estimate_accuracy_torch
Iter 2500. [Val Acc 34%] [Train Acc 35%, Loss 2.667546]
estimate_accuracy_torch
Iter 3000. [Val Acc 35%] [Train Acc 36%, Loss 2.663187]
estimate_accuracy_torch
Iter 3500. [Val Acc 35%] [Train Acc 36%, Loss 2.632345]
estimate_accuracy_torch
Iter 4000. [Val Acc 35%] [Train Acc 37%, Loss 2.646979]
estimate_accuracy_torch
Iter 4500. [Val Acc 36%] [Train Acc 37%, Loss 2.731651]
estimate_accuracy_torch
Iter 5000. [Val Acc 36%] [Train Acc 37%, Loss 2.605778]
estimate_accuracy_torch
Iter 5500. [Val Acc 36%] [Train Acc 37%, Loss 2.520568]
estimate_accuracy_torch
Iter 6000. [Val Acc 36%] [Train Acc 38%, Loss 2.540332]
estimate_accuracy_torch
Iter 6500. [Val Acc 36%] [Train Acc 38%, Loss 2.569995]
estimate_accuracy_torch
Iter 7000. [Val Acc 37%] [Train Acc 38%, Loss 2.462113]
```

```
estimate_accuracy_torch
Iter 7500. [Val Acc 37%] [Train Acc 39%, Loss 2.566820]
estimate_accuracy_torch
Iter 8000. [Val Acc 37%] [Train Acc 39%, Loss 2.627192]
```





1.3.2 Part (b) -- 10%

Use the function `make_prediction` that you wrote earlier to predict what the next word should be in each of the following sentences:

- "You are a"
- "few companies show"
- "There are no"
- "yesterday i was"
- "the game had"
- "yesterday the federal"

How do these predictions compared to the previous model?

Print the output for all of these sentences using the new network and **Write** below how the new results compare to the previous ones.

Just like before, if you encounter overfitting, train your model for more iterations, or change the hyperparameters in your model. You may need to do this even if your training accuracy is >=38%.

[19]:
```
# Your code goes here
make_prediction_torch(pytorch_wordemb, np.array(['you', 'are', 'a']))

sentences_set = ["You are a",
                 "few companies show",
```

```
                  "There are no",
                  "yesterday i was",
                  "the game had",
                  "yesterday the federal"]

for sent in sentences_set:
    words = sent.split()
    sentence = [word.lower() for word in words]
    make_prediction_torch(pytorch_wordemb, np.array(sentence))
```

```
word_idx 81, p_word good
word_idx 81, p_word good
word_idx 6, p_word .
word_idx 151, p_word other
word_idx 6, p_word .
word_idx 30, p_word been
word_idx 82, p_word government
```

Write your explanation here: This time we didn't encounter model overfitting. Due to the lower number of parameters in word_repr and hidden layers, we were able to perform further training iteractions (8,000) resulting in higher accuracy ~39%.

1.3.3 Part (c) -- 4%

Report the test accuracy of your model

[20]:
```
# Write your code here
print(f"model's test accuracy: {estimate_accuracy_torch(pytorch_wordemb,␣
 ↪test4grams)*100:.3f}%")
```

```
model's test accuracy: 37.318%
```

1.4 Question 4. Visualizing Word Embeddings (14%)

While training the `PyTorchMLP`, we trained the `word_emb_layer`, which takes a one-hot representation of a word in our vocabulary, and returns a low-dimensional vector representation of that word. In this question, we will explore these word embeddings, which are a key concept in natural language processing.

1.4.1 Part (a) -- 4%

The code below extracts the **weights** of the word embedding layer, and converts the PyTorch tensor into an numpy array. Explain why each *row* of `word_emb` contains the vector representing of a word. For example `word_emb[vocab_stoi["any"],:]` contains the vector representation of the word "any".

[21]:
```
word_emb_weights = list(pytorch_wordemb.word_emb_layer.parameters())[0]
word_emb = word_emb_weights.detach().numpy().T
```

Write your explanation here: The word_emb_layer takes a one-hot representation of a word in our vocabulary [1, 250], and returns a low dimensional vector [1, 100] (of size emb_size). Thus, by projecting the vector into low dimension space, we've actually done feature extraction to be able to represent the same vector with less entries, preserving (most) of the information. These features are the word's embeddings. Therefore, each row of word_emb contains the features vector of the word, with values that are related to its meaning. So with the word embedding, we've performed features reduction - each word from vocabulary size (250 in our case) to 100.

1.4.2 Part (b) -- 5%

One interesting thing about these word embeddings is that distances in these vector representations of words make some sense! To show this, we have provided code below that computes the *cosine similarity* of every pair of words in our vocabulary. This measure of similarity between vector $\mathbf{v}$ and $\mathbf{w}$ is defined as

$$d_{\cos}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v}^T \mathbf{w}}{||\mathbf{v}||||\mathbf{w}||}.$$

We also pre-scale the vectors to have a unit norm, using Numpy's `norm` method.

[22]:
```
norms = np.linalg.norm(word_emb, axis=1)
word_emb_norm = (word_emb.T / norms).T
similarities = np.matmul(word_emb_norm, word_emb_norm.T)

# Some example distances. The first one should be larger than the second
print(similarities[vocab_stoi['any'], vocab_stoi['many']])
print(similarities[vocab_stoi['any'], vocab_stoi['government']])
```

```
0.26410422
0.0054571438
```

Compute the 5 closest words to the following words:

- "four"
- "go"
- "what"
- "should"
- "school"
- "your"
- "yesterday"
- "not"

[ ]:
```

```

[57]:
```
# Write your code here
words = ["four", "go", "what", "should", "school", "your", "yesterday", "not"]
for w in words:
```

```
# calc cosine similarity to all words (except the candidate word w)
# sorted in ascending order (higher value = closer)
# Note that d(v,w) = 1 <=> v=w, therefore we exlude it from our search in order␣
 ↪to include
# five other closest words

  cosine_sim_w = dict( (vocab_itos[j], similarities[vocab_stoi[w], j]) for j in␣
 ↪range(250) if vocab_stoi[w] != j)
  sorted_voca_sim_w = sorted(cosine_sim_w.items(), key = lambda item: -item[1])
  print(f"{w} -> {sorted_voca_sim_w[:5]}")
```

```
four -> [('several', 0.5556028), ('five', 0.49490786), ('two', 0.4887824),
('many', 0.45272106), ('university', 0.3436991)]
go -> [('come', 0.4190465), ('going', 0.41681942), ('war', 0.38345993), ('days',
0.3550124), ('down', 0.3524234)]
what -> [('ms.', 0.42305347), ('how', 0.40496728), ('where', 0.4009357),
('when', 0.34530815), ('while', 0.3349619)]
should -> [('could', 0.49474075), ('might', 0.47298324), ('would', 0.4709409),
('can', 0.45531), ('has', 0.424331)]
school -> [('office', 0.39080352), ('well', 0.38559234), ('best', 0.37287626),
('here', 0.35278255), ('now', 0.34925443)]
your -> [('my', 0.4481828), ('our', 0.4434374), ('the', 0.38736218), ('without',
0.33359846), ('president', 0.33344522)]
yesterday -> [('year', 0.46137556), ('office', 0.4009933), ('john', 0.37942997),
('said', 0.37225956), ('season', 0.36452544)]
not -> [('nt', 0.5700117), ('never', 0.4029301), ('to', 0.30867824), ('street',
0.3078495), ('still', 0.27839538)]
```

1.4.3 Part (c) -- 5%

We can visualize the word embeddings by reducing the dimensionality of the word vectors to 2D. There are many dimensionality reduction techniques that we could use, and we will use an algorithm called t-SNE. (You don't need to know what this is for the assignment; we will cover it later in the course.) Nearby points in this 2-D space are meant to correspond to nearby points in the original, high-dimensional space.

The following code runs the t-SNE algorithm and plots the result.

Look at the plot and find at least two clusters of related words.

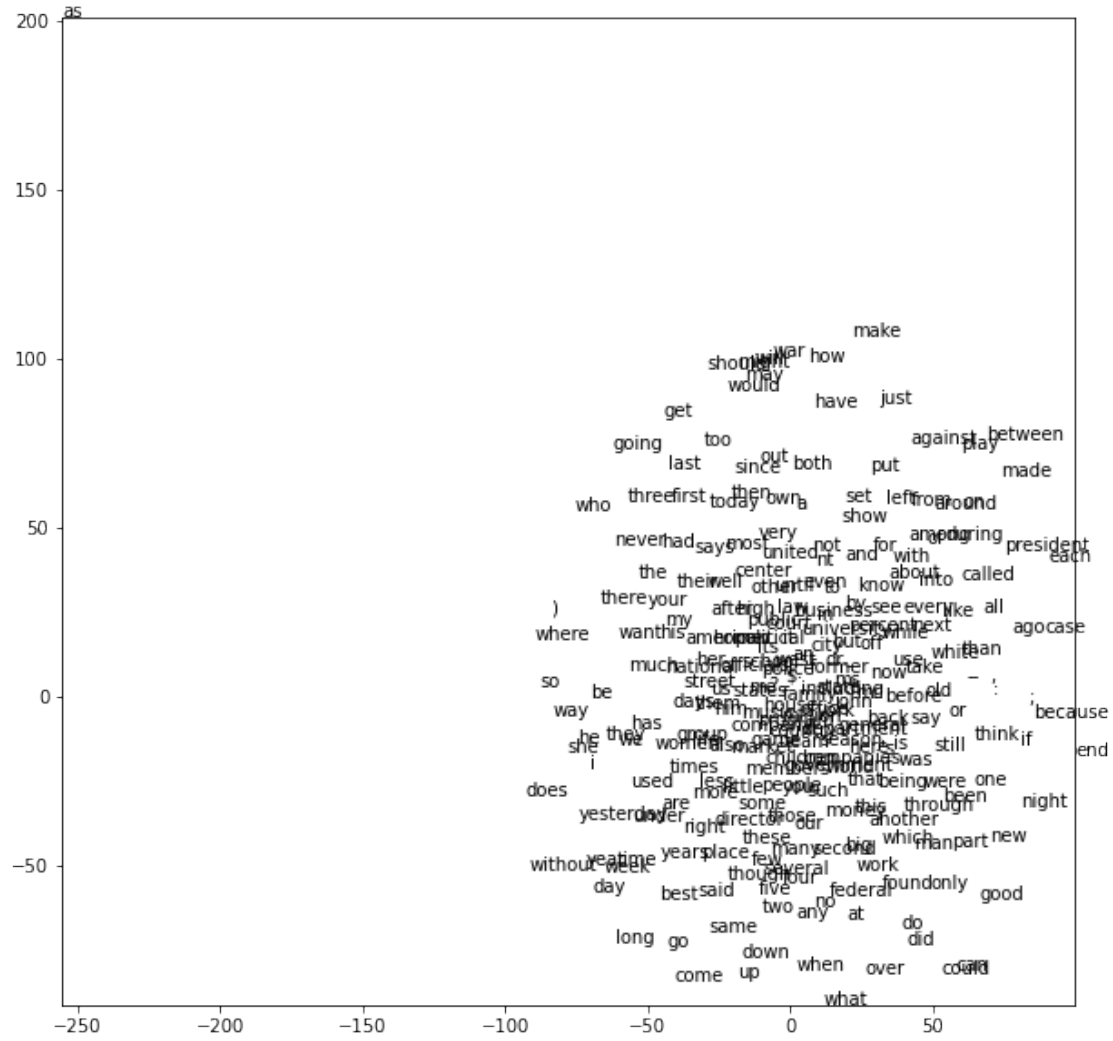
Write below for each cluster what is the commonality (if there is any) and if they make sense.

Note that there is randomness in the initialization of the t-SNE algorithm. If you re-run this code, you may get a different image. Please make sure to submit your image in the PDF file.

[44]:
```
import sklearn.manifold
tsne = sklearn.manifold.TSNE()
Y = tsne.fit_transform(word_emb)
```

```
plt.figure(figsize=(10, 10))
plt.xlim(Y[:,0].min(), Y[:, 0].max())
plt.ylim(Y[:,1].min(), Y[:, 1].max())
for i, w in enumerate(vocab):
    plt.text(Y[i, 0], Y[i, 1], w)
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/manifold/_t_sne.py:780:
FutureWarning: The default initialization in TSNE will change from 'random' to
'pca' in 1.2.
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/sklearn/manifold/_t_sne.py:790:
FutureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
  warnings.warn(
```



Explain and discuss your results here: in the above below we see many different clusters. We will analyze two of them as instructed. Under the t-SNE projection, word's embedding proximity is mapped to euclidian proximity. Thus, we are able to visualize the clustering on 2 dimensial plainsurfa, using euclidian distance. Therefore we look for cluster of words that are located close to each other in the above graph.

[three, several, four, five, two] → cluster of counting words.

[years, year, weak, day, night] → cluster of time measurements.

Both clusters make sense, as well as the other three that were marked in green.

2 Automatic PDF Generation and store in GDrive

[45]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc

from google.colab import drive
drive.mount('/content/drive')
!cp 'drive/My Drive/Colab Notebooks/Assignment2.ipynb' ./

!jupyter nbconvert --to PDF "Assignment2.ipynb"
!ls -la
!cp './Assignment2.pdf' 'drive/My Drive/Colab Notebooks'
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
pandoc is already the newest version (1.19.2.4~dfsg-1build4).
pandoc set to manually installed.
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-recommended texlive-pictures texlive-plain-generic tipa
Suggested packages:
  fonts-noto apache2 | lighttpd | httpd poppler-utils ghostscript
```

```
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf-reader
  | pdf-viewer texlive-fonts-recommended-doc texlive-latex-base-doc
  python-pygments icc-profiles libfile-which-perl
  libspreadsheet-parseexcel-perl texlive-latex-extra-doc
  texlive-latex-recommended-doc texlive-pstricks dot2tex prerex ruby-tcltk
  | libtcltk-ruby texlive-pictures-doc vprerex
The following NEW packages will be installed:
  fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono fonts-texgyre
  javascript-common libcupsfilters1 libcupsimage2 libgs9 libgs9-common
  libijs-0.35 libjbig2dec0 libjs-jquery libkpathsea6 libpotrace0 libptexenc1
  libruby2.5 libsynctex1 libtexlua52 libtexluajit2 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-did-you-mean ruby-minitest
  ruby-net-telnet ruby-power-assert ruby-test-unit ruby2.5
  rubygems-integration t1utils tex-common tex-gyre texlive texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa
0 upgraded, 47 newly installed, 0 to remove and 20 not upgraded.
Need to get 146 MB of archives.
After this operation, 460 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-droid-fallback
all 1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic/main amd64 poppler-data all
0.4.8-2 [1,479 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic/main amd64 tex-common all 6.09
[33.0 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-lmodern all
2.004.5-3 [4,551 kB]
Get:6 http://archive.ubuntu.com/ubuntu bionic/main amd64 fonts-noto-mono all
20171026-2 [75.5 kB]
Get:7 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-texgyre all
20160520-1 [8,761 kB]
Get:8 http://archive.ubuntu.com/ubuntu bionic/main amd64 javascript-common all
11 [6,066 B]
Get:9 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsfilters1
amd64 1.20.2-0ubuntu3.1 [108 kB]
Get:10 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcupsimage2
amd64 2.2.7-1ubuntu2.9 [18.6 kB]
Get:11 http://archive.ubuntu.com/ubuntu bionic/main amd64 libijs-0.35 amd64
0.35-13 [15.5 kB]
Get:12 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjbig2dec0 amd64
0.13-6 [55.9 kB]
Get:13 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9-common
all 9.26~dfsg+0-0ubuntu0.18.04.17 [5,092 kB]
```

```
Get:14 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgs9 amd64
9.26~dfsg+0-0ubuntu0.18.04.17 [2,267 kB]
Get:15 http://archive.ubuntu.com/ubuntu bionic/main amd64 libjs-jquery all
3.2.1-1 [152 kB]
Get:16 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libkpathsea6
amd64 2017.20170613.44572-8ubuntu0.1 [54.9 kB]
Get:17 http://archive.ubuntu.com/ubuntu bionic/main amd64 libpotrace0 amd64
1.14-2 [17.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libptexenc1
amd64 2017.20170613.44572-8ubuntu0.1 [34.5 kB]
Get:19 http://archive.ubuntu.com/ubuntu bionic/main amd64 rubygems-integration
all 1.11 [4,994 B]
Get:20 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 ruby2.5 amd64
2.5.1-1ubuntu1.12 [48.6 kB]
Get:21 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby amd64 1:2.5.1
[5,712 B]
Get:22 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 rake all
12.3.1-1ubuntu0.1 [44.9 kB]
Get:23 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-did-you-mean all
1.2.0-2 [9,700 B]
Get:24 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-minitest all
5.10.3-1 [38.6 kB]
Get:25 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:26 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-power-assert all
0.3.0-1 [7,952 B]
Get:27 http://archive.ubuntu.com/ubuntu bionic/main amd64 ruby-test-unit all
3.2.5-1 [61.1 kB]
Get:28 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libruby2.5
amd64 2.5.1-1ubuntu1.12 [3,073 kB]
Get:29 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libsynctex1
amd64 2017.20170613.44572-8ubuntu0.1 [41.4 kB]
Get:30 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexlua52
amd64 2017.20170613.44572-8ubuntu0.1 [91.2 kB]
Get:31 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtexluajit2
amd64 2017.20170613.44572-8ubuntu0.1 [230 kB]
Get:32 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 libzzip-0-13
amd64 0.13.62-3.1ubuntu0.18.04.1 [26.0 kB]
Get:33 http://archive.ubuntu.com/ubuntu bionic/main amd64 lmodern all 2.004.5-3
[9,631 kB]
Get:34 http://archive.ubuntu.com/ubuntu bionic/main amd64 preview-latex-style
all 11.91-1ubuntu1 [185 kB]
Get:35 http://archive.ubuntu.com/ubuntu bionic/main amd64 t1utils amd64 1.41-2
[56.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tex-gyre all
20160520-1 [4,998 kB]
Get:37 http://archive.ubuntu.com/ubuntu bionic-updates/main amd64 texlive-
binaries amd64 2017.20170613.44572-8ubuntu0.1 [8,179 kB]
```

```
Get:38 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-base all
2017.20180305-1 [18.7 MB]
Get:39 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-fonts-
recommended all 2017.20180305-1 [5,262 kB]
Get:40 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-base all
2017.20180305-1 [951 kB]
Get:41 http://archive.ubuntu.com/ubuntu bionic/main amd64 texlive-latex-
recommended all 2017.20180305-1 [14.9 MB]
Get:42 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive all
2017.20180305-1 [14.4 kB]
Get:43 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-pictures
all 2017.20180305-1 [4,026 kB]
Get:44 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-latex-
extra all 2017.20180305-2 [10.6 MB]
Get:45 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-plain-
generic all 2017.20180305-2 [23.6 MB]
Get:46 http://archive.ubuntu.com/ubuntu bionic/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:47 http://archive.ubuntu.com/ubuntu bionic/universe amd64 texlive-xetex all
2017.20180305-1 [10.7 MB]
Fetched 146 MB in 3s (55.2 MB/s)
Extracting templates from packages: 100%
Preconfiguring packages …
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 124013 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2_all.deb …
Unpacking fonts-lato (2.0-2) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.8-2_all.deb …
Unpacking poppler-data (0.4.8-2) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.09_all.deb …
Unpacking tex-common (6.09) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/04-fonts-lmodern_2.004.5-3_all.deb …
Unpacking fonts-lmodern (2.004.5-3) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/05-fonts-noto-mono_20171026-2_all.deb …
Unpacking fonts-noto-mono (20171026-2) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/06-fonts-texgyre_20160520-1_all.deb …
Unpacking fonts-texgyre (20160520-1) …
Selecting previously unselected package javascript-common.
Preparing to unpack …/07-javascript-common_11_all.deb …
Unpacking javascript-common (11) …
```

```
Selecting previously unselected package libcupsfilters1:amd64.
Preparing to unpack …/08-libcupsfilters1_1.20.2-0ubuntu3.1_amd64.deb …
Unpacking libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Selecting previously unselected package libcupsimage2:amd64.
Preparing to unpack …/09-libcupsimage2_2.2.7-1ubuntu2.9_amd64.deb …
Unpacking libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/10-libijs-0.35_0.35-13_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-13) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/11-libjbig2dec0_0.13-6_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.13-6) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/12-libgs9-common_9.26~dfsg+0-0ubuntu0.18.04.17_all.deb
…
Unpacking libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/13-libgs9_9.26~dfsg+0-0ubuntu0.18.04.17_amd64.deb …
Unpacking libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Selecting previously unselected package libjs-jquery.
Preparing to unpack …/14-libjs-jquery_3.2.1-1_all.deb …
Unpacking libjs-jquery (3.2.1-1) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/15-libkpathsea6_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libpotrace0.
Preparing to unpack …/16-libpotrace0_1.14-2_amd64.deb …
Unpacking libpotrace0 (1.14-2) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/17-libptexenc1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/18-rubygems-integration_1.11_all.deb …
Unpacking rubygems-integration (1.11) …
Selecting previously unselected package ruby2.5.
Preparing to unpack …/19-ruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking ruby2.5 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package ruby.
Preparing to unpack …/20-ruby_1%3a2.5.1_amd64.deb …
Unpacking ruby (1:2.5.1) …
Selecting previously unselected package rake.
Preparing to unpack …/21-rake_12.3.1-1ubuntu0.1_all.deb …
Unpacking rake (12.3.1-1ubuntu0.1) …
Selecting previously unselected package ruby-did-you-mean.
Preparing to unpack …/22-ruby-did-you-mean_1.2.0-2_all.deb …
Unpacking ruby-did-you-mean (1.2.0-2) …
```

```
Selecting previously unselected package ruby-minitest.
Preparing to unpack …/23-ruby-minitest_5.10.3-1_all.deb …
Unpacking ruby-minitest (5.10.3-1) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/24-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-power-assert.
Preparing to unpack …/25-ruby-power-assert_0.3.0-1_all.deb …
Unpacking ruby-power-assert (0.3.0-1) …
Selecting previously unselected package ruby-test-unit.
Preparing to unpack …/26-ruby-test-unit_3.2.5-1_all.deb …
Unpacking ruby-test-unit (3.2.5-1) …
Selecting previously unselected package libruby2.5:amd64.
Preparing to unpack …/27-libruby2.5_2.5.1-1ubuntu1.12_amd64.deb …
Unpacking libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Selecting previously unselected package libsynctex1:amd64.
Preparing to unpack …/28-libsynctex1_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexlua52:amd64.
Preparing to unpack …/29-libtexlua52_2017.20170613.44572-8ubuntu0.1_amd64.deb
…
Unpacking libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/30-libtexluajit2_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/31-libzzip-0-13_0.13.62-3.1ubuntu0.18.04.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Selecting previously unselected package lmodern.
Preparing to unpack …/32-lmodern_2.004.5-3_all.deb …
Unpacking lmodern (2.004.5-3) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/33-preview-latex-style_11.91-1ubuntu1_all.deb …
Unpacking preview-latex-style (11.91-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/34-t1utils_1.41-2_amd64.deb …
Unpacking t1utils (1.41-2) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/35-tex-gyre_20160520-1_all.deb …
Unpacking tex-gyre (20160520-1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/36-texlive-
binaries_2017.20170613.44572-8ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/37-texlive-base_2017.20180305-1_all.deb …
```

```
Unpacking texlive-base (2017.20180305-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/38-texlive-fonts-recommended_2017.20180305-1_all.deb …
Unpacking texlive-fonts-recommended (2017.20180305-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/39-texlive-latex-base_2017.20180305-1_all.deb …
Unpacking texlive-latex-base (2017.20180305-1) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/40-texlive-latex-recommended_2017.20180305-1_all.deb …
Unpacking texlive-latex-recommended (2017.20180305-1) …
Selecting previously unselected package texlive.
Preparing to unpack …/41-texlive_2017.20180305-1_all.deb …
Unpacking texlive (2017.20180305-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/42-texlive-pictures_2017.20180305-1_all.deb …
Unpacking texlive-pictures (2017.20180305-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/43-texlive-latex-extra_2017.20180305-2_all.deb …
Unpacking texlive-latex-extra (2017.20180305-2) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/44-texlive-plain-generic_2017.20180305-2_all.deb …
Unpacking texlive-plain-generic (2017.20180305-2) …
Selecting previously unselected package tipa.
Preparing to unpack …/45-tipa_2%3a1.3-20_all.deb …
Unpacking tipa (2:1.3-20) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/46-texlive-xetex_2017.20180305-1_all.deb …
Unpacking texlive-xetex (2017.20180305-1) …
Setting up libgs9-common (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libkpathsea6:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libjs-jquery (3.2.1-1) …
Setting up libtexlua52:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1) …
Setting up libsynctex1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up libptexenc1:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up tex-common (6.09) …
update-language: texlive-base not installed and configured, doing nothing!
Setting up poppler-data (0.4.8-2) …
Setting up tex-gyre (20160520-1) …
Setting up preview-latex-style (11.91-1ubuntu1) …
Setting up fonts-texgyre (20160520-1) …
Setting up fonts-noto-mono (20171026-2) …
Setting up fonts-lato (2.0-2) …
Setting up libcupsfilters1:amd64 (1.20.2-0ubuntu3.1) …
Setting up libcupsimage2:amd64 (2.2.7-1ubuntu2.9) …
Setting up libjbig2dec0:amd64 (0.13-6) …
Setting up ruby-did-you-mean (1.2.0-2) …
Setting up t1utils (1.41-2) …
```

```
Setting up ruby-net-telnet (0.1.1-2) …
Setting up libijs-0.35:amd64 (0.35-13) …
Setting up rubygems-integration (1.11) …
Setting up libpotrace0 (1.14-2) …
Setting up javascript-common (11) …
Setting up ruby-minitest (5.10.3-1) …
Setting up libzzip-0-13:amd64 (0.13.62-3.1ubuntu0.18.04.1) …
Setting up libgs9:amd64 (9.26~dfsg+0-0ubuntu0.18.04.17) …
Setting up libtexluajit2:amd64 (2017.20170613.44572-8ubuntu0.1) …
Setting up fonts-lmodern (2.004.5-3) …
Setting up ruby-power-assert (0.3.0-1) …
Setting up texlive-binaries (2017.20170613.44572-8ubuntu0.1) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
(bibtex) in auto mode
Setting up texlive-base (2017.20180305-1) …
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4:
/var/lib/texmf/tex/generic/config/pdftexconfig.tex
Setting up texlive-fonts-recommended (2017.20180305-1) …
Setting up texlive-plain-generic (2017.20180305-2) …
Setting up texlive-latex-base (2017.20180305-1) …
Setting up lmodern (2.004.5-3) …
Setting up texlive-latex-recommended (2017.20180305-1) …
Setting up texlive-pictures (2017.20180305-1) …
Setting up tipa (2:1.3-20) …
Regenerating '/var/lib/texmf/fmtutil.cnf-DEBIAN'… done.
Regenerating '/var/lib/texmf/fmtutil.cnf-TEXLIVEDIST'… done.
update-fmtutil has updated the following file(s):
        /var/lib/texmf/fmtutil.cnf-DEBIAN
        /var/lib/texmf/fmtutil.cnf-TEXLIVEDIST
If you want to activate the changes in the above file(s),
you should run fmtutil-sys or fmtutil.
Setting up texlive (2017.20180305-1) …
Setting up texlive-latex-extra (2017.20180305-2) …
Setting up texlive-xetex (2017.20180305-1) …
Setting up ruby2.5 (2.5.1-1ubuntu1.12) …
Setting up ruby (1:2.5.1) …
Setting up ruby-test-unit (3.2.5-1) …
```

```
Setting up rake (12.3.1-1ubuntu0.1) …
Setting up libruby2.5:amd64 (2.5.1-1ubuntu1.12) …
Processing triggers for mime-support (3.60ubuntu1) …
Processing triggers for libc-bin (2.27-3ubuntu1.6) …
Processing triggers for man-db (2.8.3-2ubuntu0.1) …
Processing triggers for fontconfig (2.12.6-0ubuntu2) …
Processing triggers for tex-common (6.09) …
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting pypandoc
  Downloading pypandoc-1.10-py3-none-any.whl (20 kB)
Installing collected packages: pypandoc
Successfully installed pypandoc-1.10
Mounted at /content/drive
[NbConvertApp] Converting notebook Assignment2.ipynb to PDF
[NbConvertApp] Support files will be in Assignment2_files/
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Making directory ./Assignment2_files
[NbConvertApp] Writing 118317 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 242845 bytes to Assignment2.pdf
ls: cannot access 'gdrive': Transport endpoint is not connected
total 520
drwxr-xr-x 1 root root   4096 Dec 10 16:59 .
drwxr-xr-x 1 root root   4096 Dec 10 15:52 ..
-rw------- 1 root root 264442 Dec 10 16:59 Assignment2.ipynb
-rw-r--r-- 1 root root 242845 Dec 10 16:59 Assignment2.pdf
drwxr-xr-x 4 root root   4096 Dec  8 14:35 .config
drwx------ 6 root root   4096 Dec 10 16:59 drive
d????????? ? ?    ?         ?            ? gdrive
drwxr-xr-x 1 root root   4096 Dec  8 14:36 sample_data
```