CS224 - Spring 2024 - HW1 - Joy-o-Meter

Submit **PDF** of completed IPython notebook on Canvas

Due: February 1, 2024 @ 11:59pm PDT

Maximum points: 15 (each HW is %15 of total grade)

Enter your information below:

(full) Name: Joseph Mangapit **Student ID Number**: 862175676

By submitting this notebook, I assert that the work below is my own work, completed for this course. Except where explicitly cited, none of the portions of this notebook are duplicated from anyone else's work or my own previous work.

Overview

In this assignment you will implement a simple linear neural network that reads in text and uses pretrained embeddings to predict the **happiness intensity** of the text. You'll fit the network weights using the analytic expression we discussed in class.

For this assignment we will use the functionality of PyTorch, HuggingFace "transformers" library for getting pretrained models, "pandas" for data loading, matplotlib for visualization. Before you start, make sure you have installed all those packages in your local Jupyter instance. Or use Google Colab (which has everything you need pre-installed).

Read **all** cells carefully and answer **all** parts (both text and missing code). You will complete all the code marked `TODO` and print desired results.

In [ ]:
```
%pip install torch
```

```
Requirement already satisfied: torch in c:\python311\lib\site-packages (2.1.2)
Requirement already satisfied: filelock in c:\python311\lib\site-packages (from torc
h) (3.13.1)
Requirement already satisfied: typing-extensions in c:\python311\lib\site-packages
(from torch) (4.9.0)
Requirement already satisfied: sympy in c:\python311\lib\site-packages (from torch)
(1.12)
Requirement already satisfied: networkx in c:\python311\lib\site-packages (from torc
h) (3.2.1)
Requirement already satisfied: jinja2 in c:\python311\lib\site-packages (from torch)
(3.1.3)
Requirement already satisfied: fsspec in c:\python311\lib\site-packages (from torch)
(2023.12.2)
Requirement already satisfied: MarkupSafe>=2.0 in c:\python311\lib\site-packages (fr
om jinja2->torch) (2.1.4)
Requirement already satisfied: mpmath>=0.19 in c:\python311\lib\site-packages (from
sympy->torch) (1.3.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
%pip install pandas
%pip install matplotlib
```

```
Requirement already satisfied: pandas in c:\python311\lib\site-packages (2.2.0)
Requirement already satisfied: numpy<2,>=1.23.2 in c:\python311\lib\site-packages (f
rom pandas) (1.26.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\python311\lib\site-packa
ges (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\python311\lib\site-packages (from
pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.7 in c:\python311\lib\site-packages (fro
m pandas) (2023.4)
Requirement already satisfied: six>=1.5 in c:\python311\lib\site-packages (from pyth
on-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: matplotlib in c:\python311\lib\site-packages (3.8.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\python311\lib\site-packages (f
rom matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\python311\lib\site-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\python311\lib\site-packages
(from matplotlib) (4.47.2)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\python311\lib\site-packages
(from matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\python311\lib\site-packages (fro
m matplotlib) (1.26.3)
Requirement already satisfied: packaging>=20.0 in c:\python311\lib\site-packages (fr
om matplotlib) (23.2)
Requirement already satisfied: pillow>=8 in c:\python311\lib\site-packages (from mat
plotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\python311\lib\site-packages (f
rom matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\python311\lib\site-package
s (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\python311\lib\site-packages (from pyth
on-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
%pip install transformers --user
```

```
Requirement already satisfied: transformers in c:\python311\lib\site-packages (4.37.
1)
Requirement already satisfied: filelock in c:\python311\lib\site-packages (from tran
sformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in c:\python311\lib\site
-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in c:\python311\lib\site-packages (from t
ransformers) (1.26.3)
Requirement already satisfied: packaging>=20.0 in c:\python311\lib\site-packages (fr
om transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in c:\python311\lib\site-packages (from t
ransformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in c:\python311\lib\site-packages
(from transformers) (2023.12.25)
Requirement already satisfied: requests in c:\python311\lib\site-packages (from tran
sformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in c:\python311\lib\site-packa
ges (from transformers) (0.15.1)
Requirement already satisfied: safetensors>=0.3.1 in c:\python311\lib\site-packages
(from transformers) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in c:\python311\lib\site-packages (from tr
ansformers) (4.66.1)
Requirement already satisfied: fsspec>=2023.5.0 in c:\python311\lib\site-packages (f
rom huggingface-hub<1.0,>=0.19.3->transformers) (2023.12.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\python311\lib\site-p
ackages (from huggingface-hub<1.0,>=0.19.3->transformers) (4.9.0)
Requirement already satisfied: colorama in c:\users\josep\appdata\roaming\python\pyt
hon311\site-packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\python311\lib\site-pac
kages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\python311\lib\site-packages (from
requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\python311\lib\site-packages
(from requests->transformers) (2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\python311\lib\site-packages
(from requests->transformers) (2023.11.17)
Note: you may need to restart the kernel to use updated packages.
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:

```
import torch
from transformers import AutoModel, AutoTokenizer
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```
c:\Python311\Lib\site-packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Pl
ease update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/
user_install.html
  from .autonotebook import tqdm as notebook_tqdm
C:\Users\josep\AppData\Local\Temp\ipykernel_9904\3359917248.py:3: DeprecationWarnin
g:
Pyarrow will become a required dependency of pandas in the next major release of pan
das (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better inte
roperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
```

In [ ]:

Getting and processing data [2 points]

You can download the two data files here: https://elearn.ucr.edu/files/11510987/download?download_frd=1 https://elearn.ucr.edu/files/11510996/download?download_frd=1 You'll have to make them available locally or upload them to your colab instance.

In [ ]:
```
# Load dataset and visualize
train_file = 'EI-reg-En-joy-train.txt'
val_file = '2018-EI-reg-En-joy-dev.txt'
df_train = pd.read_csv(train_file, sep='\t')
df_val = pd.read_csv(val_file, sep='\t')

tweets_train = df_train['Tweet'].tolist()  # Create a list of tweets
tweets_val = df_val['Tweet'].tolist()

# Create a list of intensity scores
y_train = torch.tensor(df_train['Intensity Score'], dtype=torch.float32)  # match t
y_val = torch.tensor(df_val['Intensity Score'], dtype=torch.float32)

print('Score - Tweet')
for i in range(5):
    print('{:0.2f} - {}'.format(y_train[i], tweets_train[i]))
```

```
Score - Tweet
0.14 - @david_garrett Quite saddened.....no US dates, no joyous anticipation of atte
nding a DG concert (since 2014). Happy you are keeping busy.
0.79 - 2 days until #GoPackGo and 23 days until #GoGipeGo..... I'm so excited!
0.27 - Positive #psychology research shows salespeople who score in the top 10% for
#optimism have 88% &gt; sales than those in top 10% for pessimism.
0.48 - As the birds chirp and the cows moo we need to listen to the sound of nature
to ensure that all is well.
0.94 - Howling with laughter at “WELL DONE BEZZA!” #bakeoff #GBBO
```

In [ ]:
```
# TODO [1 point]: Load a pretrained model and write a function that embeds sentence
# Use the approach shown in Jan. 19 class (or improve on it)
```

```
from transformers import AutoTokenizer, AutoModel
model_name="bert-base-uncased"  # Many possibilities on huggingface.com
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)
#sentence = "Hello, this right here is an example sentence!"
def embed_sentence(model, tokenizer, sentence):
    """Function to embed a sentence as a vector using a pre-trained model."""
    inputs = tokenizer(sentence, return_tensors="pt")

    with torch.no_grad():
        outputs = model(**inputs)

    return outputs.last_hidden_state[0].mean(dim=0)
```

In [ ]:

```
#TODO [1 point]: Use embed_sentence to turn text into a matrix of embeddings.
# Create a pytorch matrix where each row corresponds to a tweet,
# and the number of columns/features is the size of the embedding
# (Obviously one for train and one for validation)
# For me, on the CPU of my laptop, it took about a minute or two to do the processi
#sentence = "wowowowo w213owow 1231 3123 123 1 "
#embedding = embed_sentence(model, tokenizer, sentence)
#print(embedding.shape)
X_train = torch.tensor(df_train['Intensity Score'], dtype=torch.float32)  # match t
X_val = torch.tensor(df_val['Intensity Score'], dtype=torch.float32)
X_train_embeddings = []
X_val_embeddings = []
#print(X_train_embeddings)
#X_train_embeddings.append(embedding.size(dim=0))
#X_train_embeddings = torch.cat((X_train_embeddings, embedding.size(dim=0)), dim=0)
#print(X_train_embeddings)
#embed_tensor = torch.tensor(X_train_embeddings)


for tweet in tweets_train:
    tweet_embedding = embed_sentence(model, tokenizer, tweet)
#    print(tweet_embedding.size(dim=-1))
    X_train_embeddings.append(tweet_embedding)
X_train = torch.stack(X_train_embeddings)

for tweets in tweets_val:
    tweets_embedding = embed_sentence(model, tokenizer, tweets)
    X_val_embeddings.append(tweets_embedding)
X_val = torch.stack(X_val_embeddings)
#print(sentence)
#print(embedding.shape)
#X_train = torch.randn((len(y_train), 768))  # Delete and replace placeholder
#X_val = torch.randn((len(y_val), 768))  # Delete and replace placeholder


#print(tweets_train)



#print(X_train.shape)
```

```
print(X_train.shape, X_val.shape)
#print(X_val)
```

```
torch.Size([1616, 768]) torch.Size([290, 768])
```

Define the model [5 points]

In [ ]:
```
class MyLinearNet(torch.nn.Module):
    def __init__(self, embedding_size):
        super().__init__()  # init superclass - enables many pytorch model capabili
        self.d = embedding_size  # Often convenient to store this (not a "Parameter
        # TODO [1 point]: define weights and bias parameters
        self.weights = torch.nn.Parameter(torch.randn(embedding_size, 1))
        self.bias = torch.nn.Parameter(torch.tensor(0.))
        #self.weights = torch.normal(0, requires_grad=True)
        #self.bias = torch.zeros(1, requires_grad=True)

    def forward(self, x):
        """Implement a linear model"""
        # TODO [1 point]: implement linear model, in terms of weights and biases
        # It should work on a single x, or a batch
        #y_hat = torch.ones(len(x))  # This gives a constant prediction - delete wh
        y_hat = torch.matmul(x, self.weights) + self.bias
        return y_hat

    def fit(self, X, y):
        """Given a data matrix, X, and a vector of labels for each row, y,
        analytically fit the parameters of the linear model."""
        # TODO [3 points]: Use linear regression formula to set weight and bias par

        # (a) First, construct the augmented data matrix as discussed in class

        # (b) Next, use matrix multiplication and torch.linalg.inv to implement the

        #w = torch.randn(self.d + 1)  # delete and replace
        X_t = X.t()
        X_t_X_inv = torch.linalg.inv(torch.matmul(X_t, X))
        self.weights.data = torch.nn.Parameter(torch.matmul(torch.matmul(X_t_X_inv,

        #weights = (X_t * X)^-1 * X_t * Y
        # (c) Put the solution (which includes weights and biases) into parameter
        # Use "data" to update parameters without affecting computation graph
        # (Kind of a subtle point - no need to modify my code below)
        self.weights.data = self.weights[:self.d]
        self.bias.data = self.weights[-1]
```

Results [8 points]

In [ ]:
```
def loss(model, X, y):
    predictions = model(X)
    loss = torch.nn.functional.mse_loss(predictions, y)
    return loss
```

```
    #loss = ((X-y) ** 2).mean()
    #return loss.mean()
    # TODO [1 point]: implement the mean square error loss

d = X_train.shape[1]  # embedding dimension
model = MyLinearNet(d)

loss_train = loss(model, X_train, y_train)
loss_val = loss(model, X_val, y_val)
print("\nLoss on train and validation BEFORE fitting.\nTrain: {:0.3f}, Val: {:0.3f}

model.fit(X_train, y_train)

loss_train = loss(model, X_train, y_train)
loss_val = loss(model, X_val, y_val)

# TODO [5 points]: Show that Train loss is reduced below 0.02
# and Validation loss is reduced below 0.05, at least

print("\nLoss on train and validation AFTER fitting.\nTrain: {:0.3f}, Val: {:0.3f}"
```

```
Loss on train and validation BEFORE fitting.
Train: 36.703, Val: 32.168

Loss on train and validation AFTER fitting.
Train: 0.016, Val: 0.045
```

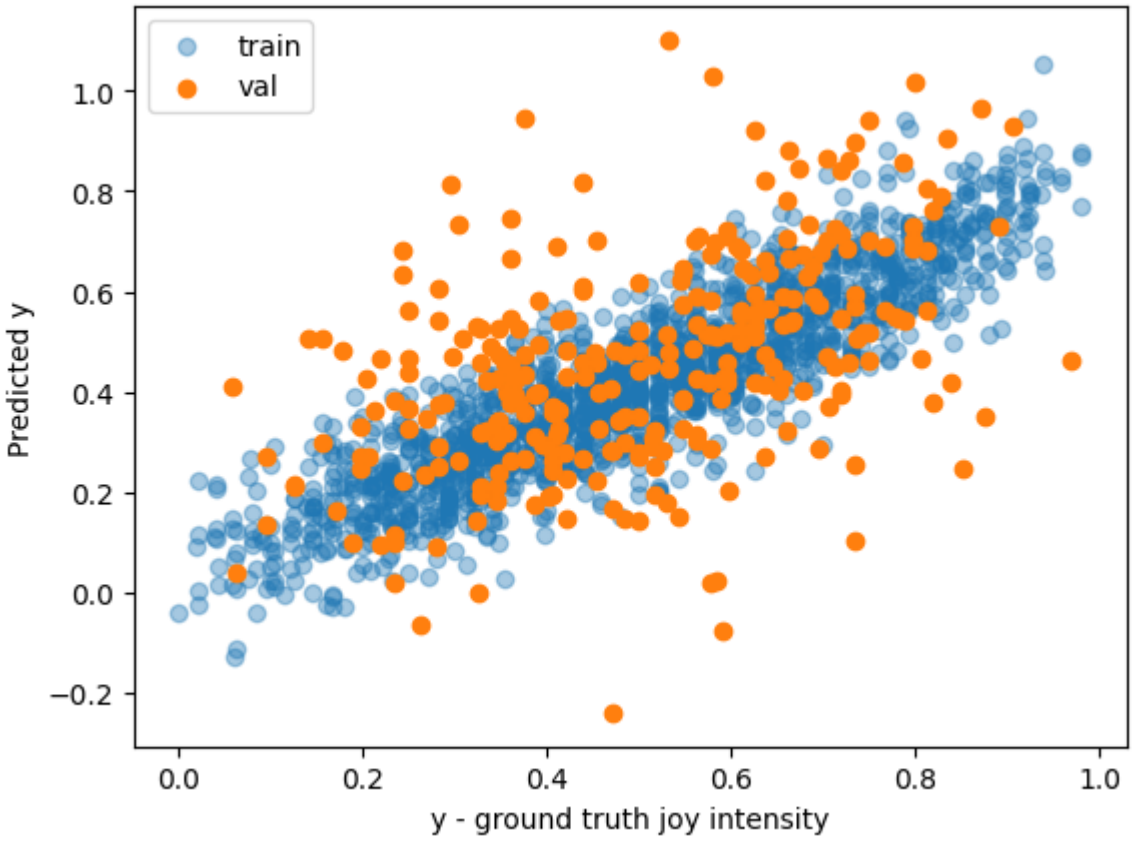
```
C:\Users\josep\AppData\Local\Temp\ipykernel_9904\917262457.py:3: UserWarning: Using
a target size (torch.Size([1616])) that is different to the input size (torch.Size
([1616, 1])). This will likely lead to incorrect results due to broadcasting. Please
ensure they have the same size.
  loss = torch.nn.functional.mse_loss(predictions, y)
C:\Users\josep\AppData\Local\Temp\ipykernel_9904\917262457.py:3: UserWarning: Using
a target size (torch.Size([290])) that is different to the input size (torch.Size([2
90, 1])). This will likely lead to incorrect results due to broadcasting. Please ens
ure they have the same size.
  loss = torch.nn.functional.mse_loss(predictions, y)
```

In [ ]:

```
# Create a scatter plot of the actual vs. predicted values of `y` using this functi
def plot(y_train, y_hat_train, y_val, y_hat_val):
    fig, ax = plt.subplots(1)
    ax.scatter(y_train, y_hat_train, alpha=0.4, label='train')
    ax.scatter(y_val, y_hat_val, label='val')
    ax.set_xlabel('y - ground truth joy intensity')
    ax.set_ylabel('Predicted y')
    ax.legend()


# TODO [1 point] show y_hat versus y on train and val data
# Should be no need to modify code, you get 1 point for getting
# something that looks correct.
with torch.no_grad():  # remember to turn off auto gradient tracking
    y_hat_train = model(X_train)
    y_hat_val = model(X_val)

plot(y_train, y_hat_train, y_val, y_hat_val)
```



In [ ]:
```
# TODO [1 point]  Put in a sample sentence of your own construction and output the
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)
happy = "I am so happy that I've finished all my work for the next week XD"
sad = "I am super tilted right now, it's ridicoulous."
embed_happy = embed_sentence(model, tokenizer, happy)
embed_sad = embed_sentence(model, tokenizer, sad)
#X_val_embeddings.append(embed_happy)
#X_val_embeddings.append(embed_sad)
#X_val = torch.stack(X_val_embeddings)

d = X_val.shape[1]  # embedding dimension
model = MyLinearNet(d)

model.fit(X_train, y_train)
#y_hat_happy = 0.7  # Delete and replace
#y_hat_sad = 0.3  # Delete and replace
with torch.no_grad():
    y_hat_happy = model(embed_happy)
    y_hat_sad = model(embed_sad)
print('{:0.2f} - {}'.format(y_hat_happy, happy))
print('{:0.2f} - {}'.format(y_hat_sad, sad))
```

```
0.75 - I am so happy that I've finished all my work for the next week XD
-0.04 - I am super tilted right now, it's ridicoulous.
```

Extra credit

There are some nice opportunities for extra credit, though I will be fairly stingy with the points, so you should only try it if you're interested in learning more. Some examples of things you could try for 1 extra point. princeton-nlp/unsup-simcse-bert-base-uncased or sentence-transformers/all-mpnet-base-v2 and a CLIP variant (trained on text and images). I hypothesize the contrastive methods will be better than CLIP for this task (which is not a visual task). You could also try a GPT model embedding, but while they are great at generation the embeddings are typically not useful for other tasks.

- Compare multiple embedding methods. For instance, I'd look at a contrastive method like

- Work ahead and try putting in a multi-layer MLP and training with SGD. How much can you improve the validation loss?
- Compare different strategies for extracting BERT embeddings: instead of using the mean embedding like I showed in class, compare to the embedding from the first token (to make this work better, people sometimes prepend the sentence with a special [cls] token before tokenizing).