This notebook is an exercise in the Intro to Deep Learning course. You can reference the tutorial at this link.

Introduction

In this exercise you'll train a neural network on the *Fuel Economy* dataset and then explore the effect of the learning rate and batch size on SGD.

When you're ready, run this next cell to set everything up!

In [ ]:

```
# Setup plotting
import matplotlib.pyplot as plt
from learntools.deep_learning_intro.dltools import animate_sgd
plt.style.use('seaborn-whitegrid')
# Set Matplotlib defaults
plt.rc('figure', autolayout=True)
plt.rc('axes', labelweight='bold', labelsize='large',
       titleweight='bold', titlesize=18, titlepad=10)
plt.rc('animation', html='html5')

# Setup feedback system
from learntools.core import binder
binder.bind(globals())
from learntools.deep_learning_intro.ex3 import *
```

In the *Fuel Economy* dataset your task is to predict the fuel economy of an automobile given features like its type of engine or the year it was made.

First load the dataset by running the cell below.

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import make_column_transformer, make_column_selector
from sklearn.model_selection import train_test_split

fuel = pd.read_csv('../input/dl-course-data/fuel.csv')

X = fuel.copy()
# Remove target
y = X.pop('FE')

preprocessor = make_column_transformer(
    (StandardScaler(),
     make_column_selector(dtype_include=np.number)),
    (OneHotEncoder(sparse=False),
     make_column_selector(dtype_include=object)),
)

X = preprocessor.fit_transform(X)
y = np.log(y) # log transform target instead of standardizing

input_shape = [X.shape[1]]
print("Input shape: {}".format(input_shape))
```

Take a look at the data if you like. Our target in this case is the `'FE'` column and the remaining columns are the features.

In [ ]:

```
# Uncomment to see original data
#fuel.head()
# Uncomment to see processed features

pd.DataFrame(X[:10,:]).head()
```

Run the next cell to define the network we'll use for this task.

In [ ]:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=input_shape),
    layers.Dense(128, activation='relu'),
    layers.Dense(64, activation='relu'),
    layers.Dense(1),
])
```

1) Add Loss and Optimizer

Before training the network we need to define the loss and optimizer we'll use. Using the model's `compile` method, add the Adam optimizer and MAE loss.

In [ ]:

```
# YOUR CODE HERE
model.compile("optimizer")

# Check your answer
q_1.check()
```

In [ ]:

```
# Lines below will give you a hint or solution code
#q_1.hint()
#q_1.solution()
```

2) Train Model

Once you've defined the model and compiled it with a loss and optimizer you're ready for training. Train the network for 200 epochs with a batch size of 128. The input data is `X` with target `y`.

In [ ]:

```
# YOUR CODE HERE
history = ____

# Check your answer
q_2.check()
```

In [ ]:

```
# Lines below will give you a hint or solution code
#q_2.hint()
#q_2.solution()
```

The last step is to look at the loss curves and evaluate the training. Run the cell below to get a plot of the training loss.

In [ ]:

```
import pandas as pd
```

```
history_df = pd.DataFrame(history.history)
# Start the plot at epoch 5. You can change this to get a different view.
history_df.loc[5:, ['loss']].plot();
```

3) Evaluate Training

If you trained the model longer, would you expect the loss to decrease further?

In [ ]:

```
# View the solution (Run this cell to receive credit!)
q_3.check()
```

With the learning rate and the batch size, you have some control over:

- **How long it takes to train a model**
- **How noisy the learning curves are**
- **How small the loss becomes**

To get a better understanding of these two parameters, we'll look at the linear model, our ppsimplest neural network. Having only a single weight and a bias, it's easier to see what effect a change of parameter has.

The next cell will generate an animation like the one in the tutorial. Change the values for `learning_rate`, `batch_size`, and `num_examples` (how many data points) and then run the cell. (It may take a moment or two.) Try the following combinations, or try some of your own:

learning_rate	batch_size	num_examples
0.05	32	256
0.05	2	256
0.05	128	256
0.02	32	256
0.2	32	256
1.0	32	256
0.9	4096	8192
0.99	4096	8192

In [ ]:

```
# YOUR CODE HERE: Experiment with different values for the learning rate, batch size, and
number of examples
learning_rate = 0.05
batch_size = 32
num_examples = 256

animate_sgd(
    learning_rate=learning_rate,
    batch_size=batch_size,
    num_examples=num_examples,
    # You can also change these, if you like
    steps=50, # total training steps (batches seen)
    true_w=3.0, # the slope of the data
    true_b=2.0, # the bias of the data
)
```

4) Learning Rate and Batch Size

What effect did changing these parameters have? After you've thought about it, run the cell below for some discussion.

In [ ]:

```
# View the solution (Run this cell to receive credit!)
q_4.check()
```

Keep Going

Learn how to improve your model's performance by tuning capacity or adding an early stopping callback.

Have questions or comments? Visit the Learn Discussion forum to chat with other Learners.