Lab 18

Today, we will spend a bit of time on Decision Trees and Emsemble methods, before transitioning to examining the efficacy of an algorithm. Today, we will:

1. Define standard metrics for efficacy
2. Work on timing implementations of algorithms

It will be tempting to read quickly through the preamble of this lab, but take your time here. These terms are critical to discussing machine learning algorithms. It is recommended that you either work individually or in a small group.

Machine Learning at Scale

So far in the course, we have not paid particular attention to the efficacy of our algorithms. Efficiency becomes increasingly important as we scale our algorithms to work on larger and larger datasets. Instead, we have just coded for heuristic understanding of the underlying principles of machine learning. Today, we turn our attention to the standard measures for the efficacy of an implementation.

Parts of this lab may feel similar to when we worked with Gradient Descent, we did consider the raw number of computations that would need to happen for each round of gradient descent.

Epoch

Often we see comparisons of an implementation's training error compared to the number of *epochs*. An epoch is the point where the algorithm has seen the whole set of training data exactly one time.

We have seen examples where an algorithm has to work iteratively over the whole training dataset. With your group, list at least 3 such examples:

1.
2.
3.

Batch Size

There are other comparisons for an implementation's training error. Another common one is the *batch size* or the number of training examples that the algorithm "sees" in each pass.

What are two examples of where we see batch size as part of the design?

1.
2.

Iterations

When an algorithm relies on batches of the training data, we often consider the number of *iterations* of the algorithm needed to complete one epoch (or to "see" all the training data). This number can be directly computed from the total number of data points and the batch size.

Epochs, Batch Sizes, and Iterations, oh my!

Let's take a moment to look closer at a few of these algorithms via these metrics. For each row, fill in the relationship between epochs and iterations. We'll assume that there are n data points. If there is a batch size that is neither 1 nor n, we denote that b. I've filled in row for you:

	Epochs	Batch Size	Iterations
k-means	...	...	...
kNN	...	...	...
Gradient Descent	1	n	1
Mini-Batch Gradient Descent	...	...	...
Stochastic Gradient Descent	...	...	...
Decision Trees	...	...	...

Time?

While we can talk about the number of passes of the data through an algorithm, it would be more helpful, if we could *time* our algorithms. Today, we will do just that!

In this example, we will use the random forest from last time.

Imports for today

In [ ]:
```
## Import block
%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

import numpy as np
import pandas as pd

from sklearn.ensemble import RandomForestClassifier
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier

import time
```

In [ ]:
```
# For function testing
```

In [ ]:

```
## Import Data
dog_pd = pd.read_csv("lab16data.csv", sep = ",", index_col = "Breed Name")

#Change all the booleans to numbers
dog_pd.iloc[:,7:11] = dog_pd.iloc[:,7:11].astype("int")

# Export to numpy
dog_full_np = dogs_pd.to_numpy()
```

In [ ]:
```
# Split into the input variables and the target classes
in_dog_data = dog_full_np[:,:-1]
out_class = dog_full_np[:,-1]

# Get the variable names
var_names = list(dog_pd.columns)[:-1]
```

Timing our implementations with `time`

Using the `time` module, we can time how long it takes us to get from the starting to the end of training a random forest on our weather data:

In [ ]:
```
# Start the timer:
start_time = time.time()

# Specify and fit the model
grove = RandomForestClassifier(n_estimators=10, max_features = 3, max_depth=2, r
grove.fit(in_dog_data, out_class)

# Stop the clock and determine the length of time
stop_time = time.time()

print("This took %s seconds to run" % (stop_time-start_time))
```

Making comparisons

In this next section, we will compare your own implementations of k-means and kNN to the off-the-shelf ones. You will need:

1. Your k-means implementation from HW2
2. Your kNN implementation from Lab 7
3. The associated data for both labs (you may want to copy this data into this directory)

In [ ]:
```
## The data for your labs
```

In [ ]:
```
## Add your k-means implementation here
```

In [ ]:
```
## Your kNN implementation from Lab 7
```

Use `time` to time k-means

Using the `time` module, we can time how long it takes us to get from the starting to the end of your k-means implementation. Then time how long `sklearn` takes.

In [ ]:
```
## Run a time analysis for your k-means
```

In [ ]:
```
## Run a time analysis for sklearn's k-means
```

Comparing implementations

Which was faster? By how much?

What do you think would happen if you had 100 million data points?

Use `time` to time k-means

Using the `time` module, we can time how long it takes us to get from the starting to the end of your kNN implementation. Then time how long `sklearn` takes.

In [ ]:
```
## Run a time analysis for your kNN
```

In [ ]:
```
## Run a time analysis for sklearn's kNN
```

Comparing implementations

Which was faster? By how much?

What do you think would happen if you had 100 million data points?

Final Thoughts

To finish up this lab, create a slack post on **#lab18_submission** channel with 1) what surprised you most in this lab about the notion of *time* in this lab, and 2) answer the question: **Why are both epoch and run time important analysis for machine learning?**

If your have questions from this lab, post them to #lab_questions with the preamble **Lab18.** If you have the same question, please use one of the emoji's to upvote the question. If you would like to answer someone's question, please use the thread function. This will tie your answer to their question.

Next Time

We will look at benchmarking and bottlenecks.

Resources consulted

1. Epoch vs Batch Size vs Iterations
2. Epoch, Iterations & Batch Size
3. How do you calculate program run time in python?
4. How do I get time of a Python program's execution?
5. 10 tricks for converting Data to a Numeric Type in Pandas