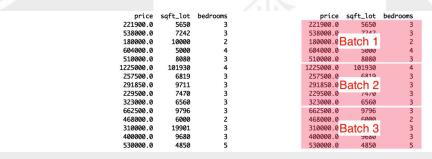
Batch training

Puteaux, Fall/Winter 2020-2021

- §2 Introduction to TensorFlow in Python
- §2.2 Linear models

1 Batch training

1.1 What is batch training?



1.2 What is the chunksize parameter?

- pandas.read_csv() allows to load data in batches:
 - avoid loading entire dataset
 - chunksize parameter provides batch size

1.3 Code of the chunksize parameter:

```
[1]: # Import pandas and numpy
import pandas as pd
import numpy as np

# Load data in batches
for batch in pd.read_csv('ref1. King county house sales.csv', chunksize=100):
```

```
# Extract price column
price = np.array(batch['price'], np.float32)

# Extract size column
size = np.array(batch['sqft_living'], np.float32)
```

1.4 Code of training a linear model in batches:

```
[2]: # Import tensorflow, pandas, and numpy
     import tensorflow as tf
     import pandas as pd
     import numpy as np
[3]: # Define trainable variables
     intercept = tf.Variable(0.1, tf.float32)
     slope = tf.Variable(0.1, tf.float32)
[4]: # Define the model
     def linear_regression(intercept, slope, features):
         return intercept + features * slope
[5]: # Compute predicted values and return loss function
     def loss function(intercept, slope, targets, features):
         predictions = linear_regression(intercept, slope, features)
         return tf.keras.losses.mse(targets, predictions)
[6]: # Define optimization operation
     opt = tf.keras.optimizers.Adam()
[7]: # Load the data in batches from pandas
     for batch in pd.read_csv('ref1. King county house sales.csv', chunksize=100):
         # Extract the target and feature columns
         price_batch = np.array(batch['price'], np.float32)
         size_batch = np.array(batch['sqft_lot'], np.float32)
         # Minimize the loss function
         opt.minimize(
             lambda: loss_function(intercept, slope, price_batch, size_batch),
            var_list=[intercept, slope])
```

```
[8]: # Print parameter values print(intercept.numpy(), slope.numpy())
```

0.31781912 0.29831016

1.5 Compare full sample versus batch training, what are the differences?

- Full Sample
 - 1. One update per epoch
 - 2. Accepts dataset without modification
 - 3. Limited by memory
- Batch Training
 - 1. Multiple updates per epoch
 - 2. Requires the division of dataset
 - 3. No limit on dataset size

1.6 Practice exercises for batch training:

► Package pre-loading:

```
[9]: from tensorflow import Variable, keras, float32
```

▶ Batch train preparing practice:

```
[10]: # Define the intercept and slope
intercept = Variable(10.0, float32)
slope = Variable(0.5, float32)

# Define the model
def linear_regression(intercept, slope, features):
        # Define the predicted values
        return intercept + features * slope

# Define the loss function
def loss_function(intercept, slope, targets, features):
        # Define the predicted values
        predictions = linear_regression(intercept, slope, features)

# Define the MSE loss
        return keras.losses.mse(targets, predictions)
```

► Package re-pre-loading:

```
[11]: import pandas as pd import numpy as np
```

▶ Linear model batches training practice:

```
[12]: # Initialize adam optimizer
    opt = keras.optimizers.Adam()

# Load data in batches
    for batch in pd.read_csv('ref1. King county house sales.csv', chunksize=100):
        size_batch = np.array(batch['sqft_lot'], np.float32)

# Extract the price values for the current batch
        price_batch = np.array(batch['price'], np.float32)

# Complete the loss, fill in the variable list, and minimize
        opt.minimize(
            lambda: loss_function(intercept, slope, price_batch, size_batch),
            var_list=[intercept, slope])

# Print trained parameters
    print(intercept.numpy(), slope.numpy())
10.217888 0.7016
```