Less painful machine learning for big data using sklearn!!

(actually mainly with pyspark 😅)

Peng Yu

@ Shopify

Logistic regression using sklearn

- What we want to do:

```
from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(loss='log', random_state=42)

clf.fit(train_X, train_y)
```

- Real life:

```
[{
    "referrer": "http://logmatic.io/",
    "request": "/wp-content/uploads/2015/10/logo.png",
    "agent": "Mozilla/5.0 (Linux; Android 5.1.1; A0001 Build/LMY48
    "response": 304,
    "clientip": "178.150.207.###",
    "verb": "GET",
    "httpversion": "1.1",
    "timestamp": "2016-01-10T22:53:44.000Z"
  },...]
```

Feature extraction

The machine learning task is where the most time will be spent.

- What we want to do:

```
from sklearn.feature_extraction import DictVectorizer
vec = DictVectorizer()
data = vec.fit_transform(list_of_raw_data_in_dict)
X = data[:, feature_dimension]
y = data[:, label_dimension]
...
```

- The big data case 👊:

```
vec = DictVectorizer()
data_iter = vec.ifit_transform(iterator_of_data_in_dict)
```

Model training

- What we want to do:

```
clf.fit(X, y)
```

- The big data case:

```
for batch_data in dict_vec.fit_transform(iterator_of_data_in_dict)
    batch_X = batch_data[:, feature_dimension]
    batch_y = batch_data[:, label_dimension]
    clf.partial_fit(batch_X, batch_y)
```

We can't actually do that ⬆️ because we don't know the exact `feature_dimension` in advance... 😢

How to get from `iterator_of_data_in_dict` to `partial_fit` ?

Enter the `on_the_fly` package.

- How about chaining iterables?

```
from on_the_fly import FlyVectorizer, FlyClassifier
vec = FlyVectorizer()
clf = FlyClassifier()
features = ['name', 'age', 'something']
label = ['gender']
for batch_data_in_dict in iterator_of_data_in_dict:
    batch_data = vec.partial_fit_transform(batch_data_in_dict)
    feature_dimension = vec.subset_features(features)
    label_dimension =  vec.subset_features(label)
    batch_X = batch_data[:, feature_dimension]
    batch_y = batch_data[:, label_dimension]
    clf.partial_fit(batch_X, batch_y)
```

- Find it on PyPI or Github. 🎉

Or wait... maybe pyspark is better for big data jobs?

- What we want to do:

```
from pyspark.ml.classification import LogisticRegression
training = spark.read.format("libsvm").load("data/mllib/sample_lib

lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=

lrModel = lr.fit(training)
```

- But wait... `sample_libsvm_data.txt` ??
- Maybe I should try `StreamingLinearRegressionWithSGD` from `mllib`.

Streaming

```
import sys

from pyspark.mllib.linalg import Vectors
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.regression import StreamingLinearRegressionWith

def parse(lp):
    label = float(lp[lp.find('(') + 1: lp.find(',')])
    vec = Vectors.dense(lp[lp.find('[') + 1: lp.find(']')].split('
    return LabeledPoint(label, vec)

trainingData = ssc.textFileStream(sys.argv[1]).map(parse).cache()
testData = ssc.textFileStream(sys.argv[2]).map(parse)

numFeatures = 3
model = StreamingLinearRegressionWithSGD()
model.setInitialWeights([0.0, 0.0, 0.0])

model.trainOn(trainingData)
```

- Need to figure out `LabeledPoint` before you can do it!

Ok, so let's use `on_the_fly`

- Training on pyspark should looks like this!

```
from on_the_fly import FlyClassifier, RddVectorizer, RddClassifier

vec = RddVectorizer(features=['name', 'age', 'stuff'], label='gend
base_clf = FlyClassifier(loss='log')
clf = RddClassifier(base_clf)

training_design_matrix = vec.fit_transform(trainning_rdd_of_dicts)

clf.fit(training_design_matrix)

testing_design_matrix = vec.transform(testing_rdd_of_dicts)

clf.score(testing_design_matrix)
```

How about `cross_validation`?

- Maybe next time...
- Or you can check it for yourself here: https://github.com/yupbank/on_the_fly/blob/master/model_selection.py

Comparison

	sklearn	pyspark	on_the_fly
vectorization	see all data	don't care	streaming
classification	feature dimension has to be fixed	data in streaming Vector object	streaming
chain iterable	✗	✗	✅

Thanks!

...and we are hiring! 😸