OTB Users Days 2016

Machine Learning in OTB: s/opencv/shark?

OTB Team

07/06/2016

Available algorithms in OTB

- ITK : KMeans
- libSVM : SVM
- OTB : SOM (several approaches), SEM, MRF
  - plus some strange things in `LearningBase` : decision trees, gaussian models

Available algorithms in OTB

- ITK : KMeans
- libSVM : SVM
- OTB : SOM (several approaches), SEM, MRF
  - plus some strange things in `LearningBase` : decision trees, gaussian models
- From OpenCV : DT, RF, ANN, KNN, GBT, Boost, normal naive Bayes.

otb : :MachineLearningModel

- Developed for the integration of the OpenCV ML module
- Also used now for libSVM
- Simple interface for integration of ML algorithms
  - `Train()` and `Save()`
  - `Load()` and `Predict()`
  - Recently extended for regression

Some issues

- `otb::MachineLearningModel` only used for supervised learning
  - KMeans, SEM, SOM are not implemented using `otb::MachineLearningModel`
  - Unsupervised algorithms have an *estimation* step which is equivalent to the *training* for supervised ones.

Some issues

- `otb::MachineLearningModel` only used for supervised learning
  - KMeans, SEM, SOM are not implemented using `otb::MachineLearningModel`
  - Unsupervised algorithms have an *estimation* step which is equivalent to the *training* for supervised ones.
- `itk::ListSample` is no good
  - too much boiler plate
  - labels and features are disjoint
  - we don't really use its services (frequency)
  - we spend cycles copying data from `itk::ListSample` to `cv::Mat`

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case
  - no hooks available for accessing useful information : confidence in RF

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case
  - no hooks available for accessing useful information : confidence in RF
  - learning is not parallel

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case
  - no hooks available for accessing useful information : confidence in RF
  - learning is not parallel
  - OpenCV is a big dependency for getting only ML

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case
  - no hooks available for accessing useful information : confidence in RF
  - learning is not parallel
  - OpenCV is a big dependency for getting only ML
  - OpenCV's code is not really C++, but rather C with classes

What's wrong with OpenCV's ML ?

- Not much : thanks to OpenCV OTB has a very rich set of ML algorithms
- But there are some annoying things
  - reliability of some algorithms : the linear SVM case
  - no hooks available for accessing useful information : confidence in RF
  - learning is not parallel
  - OpenCV is a big dependency for getting only ML
  - OpenCV's code is not really C++, but rather C with classes
- So we started looking for a replacement

About Shark

Shark is a fast, modular, feature-rich open-source C++ machine learning library.
It provides methods for linear and nonlinear optimization, kernel-based learning algorithms, neural networks, and various other machine learning techniques.
It serves as a powerful toolbox for real world applications as well as research.
Shark depends on Boost and CMake. It is compatible with Windows, Solaris, MacOS X, and Linux.
Shark is licensed under the permissive GNU Lesser General Public License.

- `http://image.diku.dk/shark/index.html`

Why Shark ?

- Speed and flexibility : "Shark provides an excellent trade-off between flexibility and ease-of-use on the one hand, and computational efficiency on the other."
  - Parallel learning (OpenMP), use of data batches for cache coherency

Why Shark ?

- Speed and flexibility : "Shark provides an excellent trade-off between flexibility and ease-of-use on the one hand, and computational efficiency on the other."
  - Parallel learning (OpenMP), use of data batches for cache coherency
- One for all : "Shark offers numerous algorithms from various machine learning and computational intelligence domains in a way that they can be easily combined and extended."
  - At Cesbio we have started combining supervised and unsupervised learning, adding some small declinations of algorithms using Shark's framework

Why Shark ?

- Speed and flexibility : "Shark provides an excellent trade-off between flexibility and ease-of-use on the one hand, and computational efficiency on the other."
  - Parallel learning (OpenMP), use of data batches for cache coherency
- One for all : "Shark offers numerous algorithms from various machine learning and computational intelligence domains in a way that they can be easily combined and extended."
  - At Cesbio we have started combining supervised and unsupervised learning, adding some small declinations of algorithms using Shark's framework
- Unique features : "Shark comes with a lot of powerful algorithms that are to our best knowledge not implemented in any other library, for example in the domains of model selection and training of binary and multi-class SVMs, or evolutionary single- and multi-objective optimization."
  - Also auto-encoders for deep learning

Why Shark ?

- Speed and flexibility : "Shark provides an excellent trade-off between flexibility and ease-of-use on the one hand, and computational efficiency on the other."
  - Parallel learning (OpenMP), use of data batches for cache coherency
- One for all : "Shark offers numerous algorithms from various machine learning and computational intelligence domains in a way that they can be easily combined and extended."
  - At Cesbio we have started combining supervised and unsupervised learning, adding some small declinations of algorithms using Shark's framework
- Unique features : "Shark comes with a lot of powerful algorithms that are to our best knowledge not implemented in any other library, for example in the domains of model selection and training of binary and multi-class SVMs, or evolutionary single- and multi-objective optimization."
  - Also auto-encoders for deep learning
- Very good documentation and tutorials
- Active project : 3.0 released Oct.2015, 3.1 released Mar.2016.

Random Forests in Shark

```
std::vector<RealVector> features;
std::vector<unsigned int> class_labels;
// load data
ClassificationDataset data =  createLabeledDataFromRange(features,
                                                          class_labels);
data.shuffle();
double initTrainSamplesRatio{0.15};
double initTestSamplesRatio{0.5};
double validationSamplesRatio{1-initTrainSamplesRatio-initTestSamplesRatio};

auto totalNbSamples = data.numberOfElements();
auto nbClasses = numberOfClasses(data);

//Split the dataset into a training, test and validation datasets: views!
ClassificationDataset dataTest = splitAtElement(data,
                                                initTrainSamplesRatio*
                                                totalNbSamples);

ClassificationDataset dataValidation = splitAtElement(dataTest,
                                                      initTestSamplesRatio*
                                                      totalNbSamples);


//Really split the data
data.makeIndependent();
dataTest.makeIndependent();
dataValidation.makeIndependent();
```

Random Forests in Shark

```
RFTrainer trainer;
RFClassifier model;
trainer.train(model, data);

// evaluate Random Forest classifier
ZeroOneLoss<unsigned int, RealVector> loss;
Data<RealVector> prediction = model(dataTest.inputs());
cout << "Random Forest on test set accuracy:     "
     << 1. - loss.eval(dataTest.labels(), prediction) << endl;

auto pred_elements = prediction.elements();

ArgMaxConverter<RFClassifier> amc;
amc.decisionFunction() = model;
auto amc_prediction = amc(dataTest.inputs());

prediction = model(dataValidation.inputs());
cout << "Random Forest on validation set accuracy:     "
<< 1. - loss.eval(dataValidation.labels(), prediction) << endl;
```

Random Forests in OTB using Shark

```
namespace Shark
{
template <class TInputValue, class TTargetValue>
class ITK_EXPORT RandomForestsMachineLearningModel
  : public MachineLearningModel <TInputValue, TTargetValue>
{
//..
private:
  shark::RFClassifier m_RFModel;
  shark::RFTrainer m_RFTrainer;
};
}//namespace Shark
```

Train

```
template <class TInputValue, class TOutputValue>
void
RandomForestsMachineLearningModel<TInputValue,TOutputValue>
::Train()
{
  std::vector<shark::RealVector> features;
  std::vector<unsigned int> class_labels;

  ListSampleToSharkVector(this->GetInputListSample(), features);
  ListSampleToSharkVector(this->GetTargetListSample(), class_labels);
  shark::ClassificationDataset TrainSamples =
    shark::createLabeledDataFromRange(features,class_labels);
  //Set parameters
  // ...

  m_RFTrainer.train(m_RFModel, TrainSamples);

}
```

Predict

```
template <class TInputValue, class TOutputValue>
typename RandomForestsMachineLearningModel<TInputValue,TOutputValue>
::TargetSampleType
RandomForestsMachineLearningModel<TInputValue,TOutputValue>
::Predict(const InputSampleType & value, ConfidenceValueType *quality) const
{
  //Inefficient: we don't use batches!
  shark::RealVector samples;
  for(size_t i = 0; i < value.Size();i++)
     samples.push_back(value[i]);
  std::vector<shark::RealVector> sampleVector;
  sampleVector.push_back(samples);
  auto InputsSamples = shark::createUnlabeledDataFromRange(sampleVector);
  //Shark's RF predicts probability!
  shark::ArgMaxConverter<shark::RFClassifier> amc;
  amc.decisionFunction() = (m_RFModel);
  auto amc_prediction = amc(InputsSamples.inputs());
  auto amc_labels = amc_prediction.elements();
  TargetSampleType target;
  target[0] = static_cast<TOutputValue>(amc_labels[0]);
  return target[0];
}
```

Issues to solve

OpenMP and ITK's multi-threading

- Shark uses OpenMP for parallel learning : great, since `otb::MachineLearningModel::Train()` is not parallel
- Shark uses OpenMP for parallel prediction : not so great, since `otb::ImageClassificationFilter` spawns its own threads
  - use `ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS=1` is a pain, and I am still not sure to understand what's happening

Issues to solve

OpenMP and ITK's multi-threading

- Shark uses OpenMP for parallel learning : great, since `otb::MachineLearningModel::Train()` is not parallel
- Shark uses OpenMP for parallel prediction : not so great, since `otb::ImageClassificationFilter` spawns its own threads
  - use `ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS=1` is a pain, and I am still not sure to understand what's happening

Using data batches

- Not a problem for the learning step since we have all the samples
- `otb::MachineLearningModel::Predict()` takes one sample at a time
  - should the `otb::ImageClassificationFilter` call a method using batches in `ThreadedGenerateData()` ?
  - but how to keep the same interface for OpenCV : `otb::MachineLearningModel::PredictBatch()` could loop over samples and call `Predict` for OpenCV ?

Issues to solve

OpenMP and ITK's multi-threading

- Shark uses OpenMP for parallel learning : great, since `otb::MachineLearningModel::Train()` is not parallel
- Shark uses OpenMP for parallel prediction : not so great, since `otb::ImageClassificationFilter` spawns its own threads
  - use `ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS=1` is a pain, and I am still not sure to understand what's happening

Using data batches

- Not a problem for the learning step since we have all the samples
- `otb::MachineLearningModel::Predict()` takes one sample at a time
  - should the `otb::ImageClassificationFilter` call a method using batches in `ThreadedGenerateData()` ?
  - but how to keep the same interface for OpenCV : `otb::MachineLearningModel::PredictBatch()` could loop over samples and call `Predict` for OpenCV ?

s/OpenCV/Shark ?

- Well, not really, let's keep them side by side

s/OpenCV/Shark ?

- Well, not really, let's keep them side by side
- Roadmap for integration
  - For 5.6
    - Start with RF to understand issues
    - Add Shark to Superbuild
  - Later (and not Shark specific)
    - Add unsupervised algorithms with the same iterface