Brief Introduction to LEMGA

Hsuan-Tien Lin

Learning Systems Group, California Institute of Technology

Talk for CS 156b, 2006/01/12
LEMGA is HARD.......... but USEFUL.

- **Author**: Ling Li @ Learning Systems Group, 2001.
- **Learning Models**: Stump, Perceptron, Neural Network, Bagging, AdaBoost, SVM (through LIBSVM 2.81), etc.
- **Generic Optimization Algorithms**: Gradient Decent, Conjugate Gradient, etc.
- **Dataset Handling Routines**: DataSet, DataWgt, etc.
User Perspective of LEMGA

**LEMGA is HARD**
- C++ and STL are tricky
- complicated inheritance structure
- mysterious code by advanced programmer
- even need careful compiling
- evolving library through research

**.......... but USEFUL**
- in general fast
- easily extendable through OOP
- usually efficient and bugless
- many templates as a start
- always room to add your own code

**LEMGA is HARD.......... but USEFUL.**
A Tiny Example with AdaBoost

- **set up model:**
  ```cpp
  lemga::AdaBoost ada;
  lemga::Stump st;
  ada.set_base_model(st);
  ada.initialize();
  ```

- **load dataset:**
  ```cpp
  std::ifstream fd("train.dat");
  lemga::pDataSet trd =
      lemga::load_data(fd, 20, 4, 1);
  ```

- **train:**
  ```cpp
  ada.set_train_data(trd);
  ada.set_max_models(100);
  ada.train();
  ```
compute test error (first load test set in ted):

```cpp
double err = 0;
for(int i = 0; i < ted->size(); ++i)
    err += ada.c_error(ada(ted->x(i)), ted->y(i));
err /= ted->size();
```

save model:

```cpp
std::ofstream fm("ada.mdl");
fm << ada; fm.close();
```

load model:

```cpp
std::ifstream fi("ada.mdl");
lemga::AdaBoost ada2;
fi >> ada2; fi.close();
```

why does LEMGA look so easy?
Using LEMGA Learning Models

- every model inherited from LearnModel: the procedures are “almost” the same with every model
- AdaBoost ⇒ Bagging
  - just change the class name
- AdaBoost ⇒ SVM
  - no need to set base model, but need to set kernels and costs
  - SVM currently cannot be saved
- Stump ⇒ Neural Network
  - with some more lines for setting layers
- how do we know the “almost” part?

copy from the test templates; check the warnings/errors
abstract objects in iterative optimization: Objective, Direction, Step

generic information from Objective:

- gradient()
- weight(), set_weight()
- cost()
- stop_opt()

then,

iterative_optimize(
    _gradient_descent<Obj, Dir, Step>(Obj*, rate));

algorithms: GD, Line Search, CG, etc.

where are the examples?

copy from FeedForwardNN::train(), Boosting::train_gd(), etc.
Extending LEMGA Libraries

**DONTs**
- start from an empty myboost.cpp
- try to read code in detail and figure out the “best” inheritance position of myboost
- design the members and functions from scratch

**DOs**
- start from a similar file, say, adaboost.cpp
- copy adaboost.cpp to myboost.cpp (and .h)
- rename every occurrence of AdaBoost by MyBoost
- modify necessary lines

“code copying” makes LEMGA easier for you
Suggestions

from the author

- read the code before using
- be careful about assumptions
- don’t hesitate to modify any code
- look for other helpful packages/resources

from me

- more “code copying”, less “code inventing”
- read the code “when necessary”
- packages are all “hard”; LEMGA too, but “useful”
- usually “harder” to write everything on your own

Site and manual:

http://www.work.caltech.edu/ling/lemga

author: Ling Li (ling@caltech.edu)

experienced user: Hsuan-Tien Lin (htlin@caltech.edu)

bug reports very welcomed