

FASTlib

*A library of Fundamental Algorithmic and
Statistical Tools*

FASTlab

College of Computing
Georgia Institute of Technology

Why our own library?

Fundamentally, FASTlab wants to take control of its code, and you can help! We aim to:

- Establish extensible framework for rapid algorithm development
- Be specifically geared towards statistical machine learning
- Do things right with regards to documentation and usability

Design mission statement

To help both you and ourselves, we've designed FASTlib with the following in mind:

- Support a low learning curve
- Allow/ecourage distributed development
- Be fast, but avoid the worst problems of low-level development

Our goals

In the long run:

- Offer fast code for state-of-the-art algorithms in many fields
- Make it easy for others to incorporate/understand/contribute to our work

But for now:

- Broaden our selection of methods for comparison purposes
- Hammer out the problems in our nascent library

To quell one question...

We're aware of Frank's OCaml library, but we have reasons to head in our own direction:

- Audience - astrophysicists, etc., would rather use FORTRAN
- Speed - empirically, OCaml often beats C by small margins, but sometimes loses by unacceptable margins
- HPC - C and C++ have wide support for parallel computing

Aside: if anyone know's OCaml forwards and backwards, we should talk.

Organization

There are two main parts:

- Core - What this presentation describes. This part can be built by the community but will have much stricter standards.
- Arsenal - Collaborateively-built standalone algorithms. Most anything goes.

Everyone will have read/write access to a Subversion repository with both, although the core is maintained separately.

Coding for FASTlib

C/C++ is our language of choice:

- Portable, well-known even among non-CS
- Low-level enough to sidestep most overhead
- High-level enough to allow significant code reuse

Python plays a role in our library as well, but mostly for scripting. You won't have to worry much about Makefiles.

Some design decisions

We've had to make some tough choices:

- initializers and default constructors - allows broader templating, sidesteps several fundamental issues
- class hierarchies - we favor shallow class hierarchies and templating, with no virtual functions
- ease/power duality - easy, quick functions for casual programmer, but "power user" features also available without too much extra effort
- C/C++ biased towards C - when in doubt, the code may look like C++ but it "feels" like C

We will eventually hash out the standard on the wiki.

Main Packages

- Debug/Base:** Tools to help figure out what's going on
- FASTexec:** Parameters, timers, results, and automation
- Datasets:** Reading/writing/accessing mixed data
- Collections:** Dynamic arrays and other storage classes
- Math:** Kernels, metrics, etc.; eventually more
- Linear Alg:** An interface for LAPACK
-
- Spatial Trees:** *(future)* Formation and utilities

Features

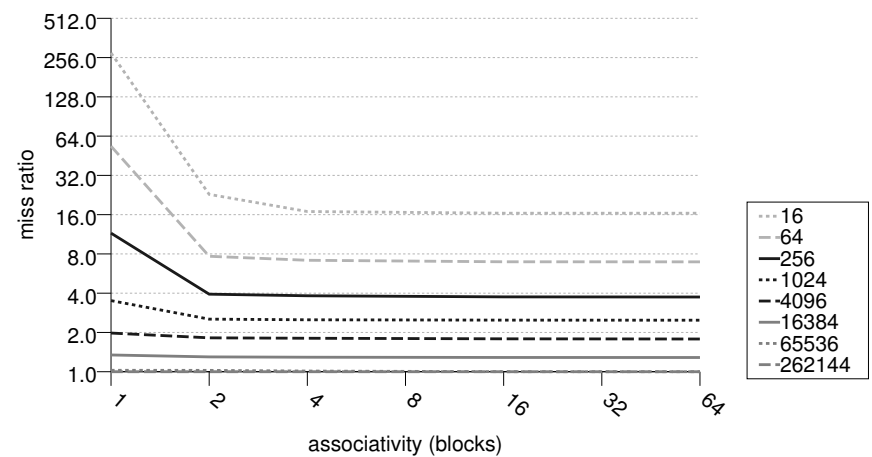
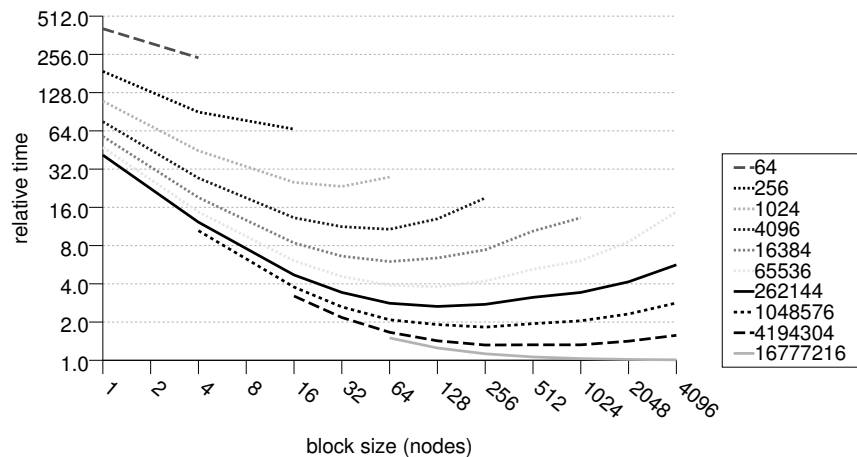
Debug/Base:

- Low-overhead bounds-checking and assertions, in debug mode
- Messages with adjustable verbosity levels
- Warnings and error messages that can be compiled out
- Compiler directives used to help branch prediction (`likely` macro)

Features

FASTexec:

- Registry-like storage of parameters, results, and timers
- Parse parameters from the command line
- Automatically collects various system information
- Create registry submodules suitable for modular code
- Python scripts scrape these into CSV and charts



Features

Datasets:

- Data stored as a matrix of doubles
- But, features may be continuous, integral, or nominal
- Column major storage for LAPACK
- I/O for .arff and .cvs file formats
- Aliasing for copy-less access to column vectors
- Cross validation support

Features

Collections:

- Dynamic array template - has knowledge of library standards
- Priority queue
- More to come later, C++ has a wide variety of STL classes

Features

Math:

- Matrices, vectors, linear systems, decompositions, eigenvalues
- Gaussian and Epanechnikov kernels; eventually more
- Tools for discrete math, geometry, etc.
- Great opportunity for extension, hint hint

Features

Linear Algebra:

- Easier-to-use calls to LAPACK functions
- Supports custom built ATLAS, or builds reference implementation automatically

Spatial Trees: (future)

- Various building options
- Tempated statistics computed for nodes

Code Examples

Main for K Nearest Neighbors:

```
#include "knn.h"
#include "fastlib/fastlib.h"
int main(int argc, char *argv[]) {
    fx_init(argc, argv);
    const char *fname =
        fx_param_str(fx_root, "fname", NULL);
    Dataset dataset;
    dataset.InitFromFile(data);
    SimpleCrossValidator<KnnClassifier> xvd;
    xvd.Init(&dataset, 0, 10, fx_root, "knn");
    xvd.Run();
    fx_done();
}
```

Code Example

The last example:

- Accepts parameters for number of cross validation folds and number of nearest neighbors:

```
./main --kfold/k=10 --knn/k=5 --fname=q.arff
```

- **Builds:** `fl-build main --mode=fast`

- **Yields results (and timers for everything):**

```
/kfold/results/p_correct 0.805
```

```
/kfold/timers/total/wall/sec 0.025244
```

Code Example - 1 Nearest Neighbor

```
int KNNClassifier::Classify(const Vector& test)
{
    double closest = DBL_MAX; int label = -1;
    for (index_t i = 0; i < matrix_.n_cols(); i++)
        Vector train;
        double dist_squared;
        matrix_.MakeColumnSubvector(i, 0,
            matrix_.n_rows() - 1, &train);
        dist_squared = la::DistanceSqEuclidean(
            test, train);
        if (unlikely(dist_squared < closest)) {
            closest = dist_squared;
            label = int(
                matrix_.get(i, matrix_.n_rows()-1));
        }
    }
return label; }
```

How do I start coding?

When starting out, keep in mind:

- A tutorial should be on wiki very soon, with installation instructions and some basic pointers.
- For common tasks, we'll collaboratively build a "cookbook" on the wiki.
- Almost every non-power-user feature is documented with Doxygen (like Javadoc).