Polishing Shogun’s linear algebra library

I’m Chris Goldsworthy (my github handle is c4goldsw) and I’m in my second year of my undergraduate education in computer science at the University of Toronto in Canada. I’m American and British and have lived in a total of six countries, previously in Denmark - I’ll mostly be in Copenhagen for the summer. I started studying computer science four and a half years ago in highschool and I’ve recently gone through a surge of interest for machine learning - Shogun is interesting to me for this reason (the first open source project to do so) and I’m in the process of making my first contributions to it. I find the project appealing, as working on it will compliment my intended academic career - I certainly expect to be contributing to the project during the rest of my undergraduate education. I’d love take this on as a major commitment for the summer - I’ve never done anything like it before.

My general skills are:

- **C / C++:** 4 / 5: I’ve taken a course in systems programming on UNIX operating systems using C, in which I received an A- . I’m in the process of learning C++ - my first language was also Java, so the object oriented aspects of C++ pose no difficulties to me. Some of my sample code can be found on SO, in which I go over a server client written in C I made for a course assignment. **My first linalg patch will be sent over the coming weekend** and will be here**.**
- **Python:** 3 / 5: I was able to skip the first year programming courses in university (when python is taught) due to courses I took in high school, so I’ve never had to use the language. I’ve dabbled with it and had no difficulty with it.
- **CMake:** 1 / 5: Shogun is my first real exposure to it.
- **SWIG:** 1 / 5: Same as above.
- **Eigen3 / LAPACK:** 1 / 5: Same as above.
- **Shogun:** 2 / 5: I came across this project a week ago and I’ve started exploring the codebase - I’m in the process of making my first contribution to it.
- **Linear Algebra:** 4 / 5: I’m in the second half of a linear algebra course, with an emphasis on proofing. We recently covered inner product spaces and we’re currently looking at Jordan canonical form.
- **Statistics / Machine learning:** 2 / 5: I haven’t done any statistics since high school - I’ll be taking two statistics courses in the next academic year (with an emphasis on proofs), as well as a course in neural networks.

My skills in relation to my selected project:

- **Advanced C / C++:** 3 / 5: I’m in the process of learning C++ and still have to brush up on generic programming with templates (I have experience with generics in Java)

- **Numerical Linear Algebra:** 2 / 5. I've done a bit before and hope to dive into it during this project.
- **Eigen3 / LAPACK:** 1 / 5: I'm new to both of these libraries.
- **Vienna/OpenCL:** 1 / 5: I'm new to both of these.
- **Shogun algorithms:** 1 / 5 : I'm new to Shogun and have yet to explore and use it's ML algorithms.

This project interests me the most due to my current coursework and is based off what is listed under the GSoC 2015 linalg project. Aside from listing off the proposed benefits in the document, the project is beneficial from a logistical point of view due to its scope and specificity - assigning an individual to it would allow that individual to specialize in carrying out work on **linalg**, whilst other developers can focus on refactoring and cleaning up other code. For example, this task may require a developer to learn Vienna/OpenCL (if they don't know about them yet). By assigning a single person to this task, another developer who would have otherwise been responsible for the work covered by this project as well as other work wouldn't have to spend time learning something they may not use much of. A proposed timeline for this project is as follows:

Present to mid-April: Starting off
- Familiarizing myself with Shogun's overall structure (by doing entrance level **linalg** related tasks) and with concepts needed for the project (e.g. GPU programming).

Mid-April to mid-May: Preliminary work
- Continue familiarizing myself with Shogun and any outstanding concepts that I need to know for the project (namely Eigen3, LAPACK and ViennaCL/OpenCL)
- Ensure that I know the complete inner workings of **linalg** (including **Core** and **Redux**)
- Start researching the specific operations listed here and going through the process of implementing some operations into **linalg,** in order to get a feel for how long it will take for each operation.
- For instance, this would involve taking library dependent classes like DirectEigenSolver (which depends on Eigen3) and replacing them with template methods which can in turn be coupled with backend specific implementations (e.g. the mean template method in Redux.h and the Eigen3 mean implementation in MeanEigen3.h).

Mid-May to the end of July: Main work
- Implement each operation I took stock of during the previous month. Ensure that each algorithm is library independent and functional (tested), ensure each operation is well documented, etc.
- This should be a fairly cyclical and repetitive process for each operation

August: Cleaning up
- Going through Shogun's current code and algorithms and ensuring that linear algebra operators come from the **linalg** interface.

- Removal of old linear algebra operations that are unnecessary