The SHOGUN Machine Learning Toolbox

(let's get started)

Sören Sonnenburg

(for the SHOGUN Team)

将軍

Who Am I?

About Me

- 1997-2002 studied Computer Science
- 2002-2009 doing ML & Bioinformatics research at Fraunhofer Institute FIRST and Max Planck Society
- 2008 PhD: Machine Learning for Genomic Sequence Analysis
- 2009-2011 Researcher at Berlin Institute of Technology
- 2011- TomTom R&D

Open Source Involvement

- Debian Developer `http://www.debian.org`
- Machine Learning OSS `http://mloss.org`
- Machine Learning Data `http://mldata.org`
- **Initiator of SHOGUN - this talk**

More about me `http://sonnenburgs.de/soeren`

将軍

What is Machine Learning and what can it do for you?

What is ML?

AIM: Learning from empirical data!

Applications

- speech and handwriting recognition
- search engines, natural language processing
- medical diagnosis, bioinformatics, chemoinformatics
- detecting credit card fraud
- computer vision, object recognition
- stock market analysis
- network security, intrusion detection
- brain-machine interfaces

...

What is Machine Learning and what can it do for you?

What is ML?

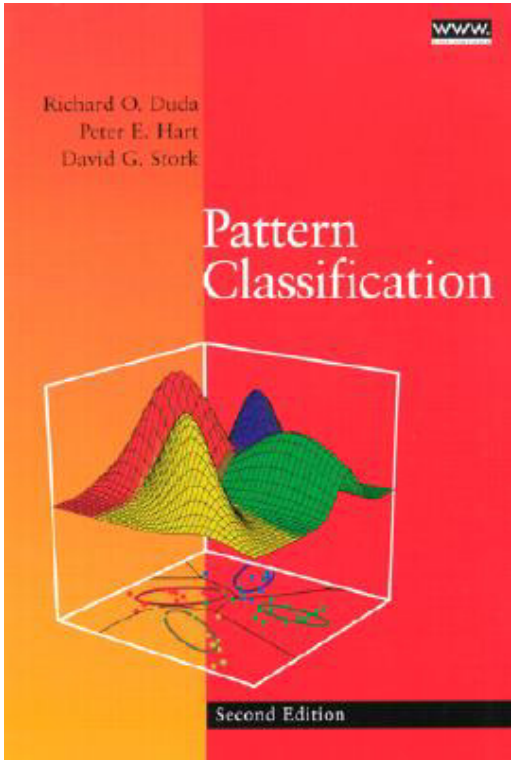
AIM: Learning from empirical data!

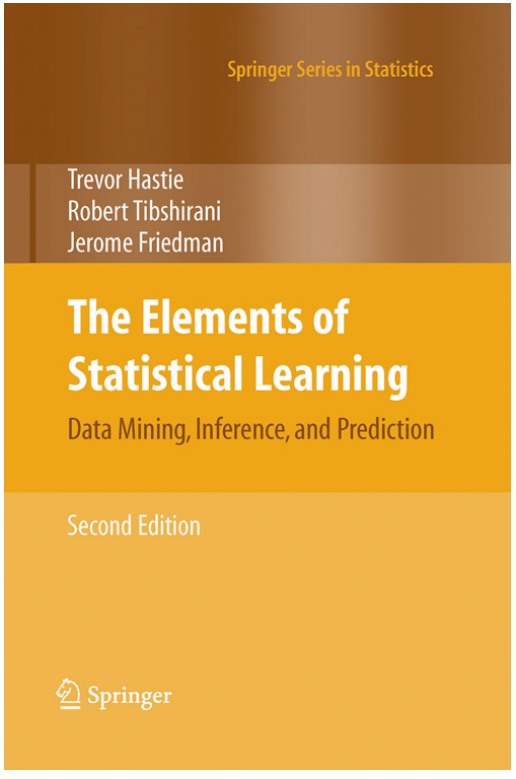
Applications

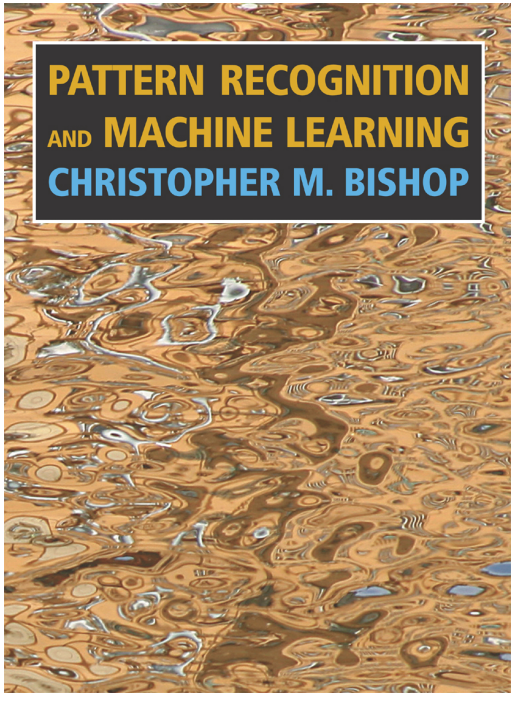
- speech and handwriting recognition
- search engines, natural language processing
- medical diagnosis, bioinformatics, chemoinformatics
- detecting credit card fraud
- computer vision, object recognition
- stock market analysis
- network security, intrusion detection
- brain-machine interfaces

...

Books on Machine Learning







. . . **and many more** . . .

SHOGUN Machine Learning Toolbox History

History

1999	Roots in my student research project *Hidden Markov Model for Genome Analysis*
2001	Framework for Kernels, and SVMs (with G. Rätsch) architectures linux, solaris, ospf1 alpha cluster first interface (cmdline)
2002-2005	Developed at Fraunhofer FIRST under the name **gf**
2006	First public release (June) under (SHOGUN) based on the initators first names S. Sonnenburg and G. Rätsch
2008	used in 3rd party code (PyVMPA)
2011-2013	Part of Google Summer of Code

Current Core Developers

Core Developers

- Fernando Iglesias (since GSoC 2012)
- Heiko Strathmann (since GSoC 2011)
- Sergey Lisitsyn (since GSoC 2011)
- Sören Sonnenburg
- Viktor Gal (since GSoC 2012)

By now

- $> 20,000$ commits
- $> 240,000$ lines of code
- over 1100 examples
- from 96 contributors

SHOGUN Main Features and Focus

Machine Learning Methods Overview

- Regression
- Distributions
- Dimension Reduction
- Performance Measures
- Clustering
- Classification
- Structured Output Learning
- Transfer Learning
- Model Selection

⋮

Focus

Unified (large-scale) learning for various feature types and settings

将軍

SHOGUN Machine Learning Toolbox

Implementation and Interfaces

- Implemented in C++
- **Interfaces:** libshogun, python, octave, R, matlab, cmdline, lua, ruby, java, csharp
- **Over 1100 examples**
- Doxygen documentation
- Inline python documentation (e.g. `help(GaussianKernel)`)
- **Testsuite** ensuring that obvious bugs do not slip through

Continuity

Due to being GPLv3'd it survived policy changes at the authors institute and job changes.

SHOGUN Machine Learning Toolbox

Implementation and Interfaces

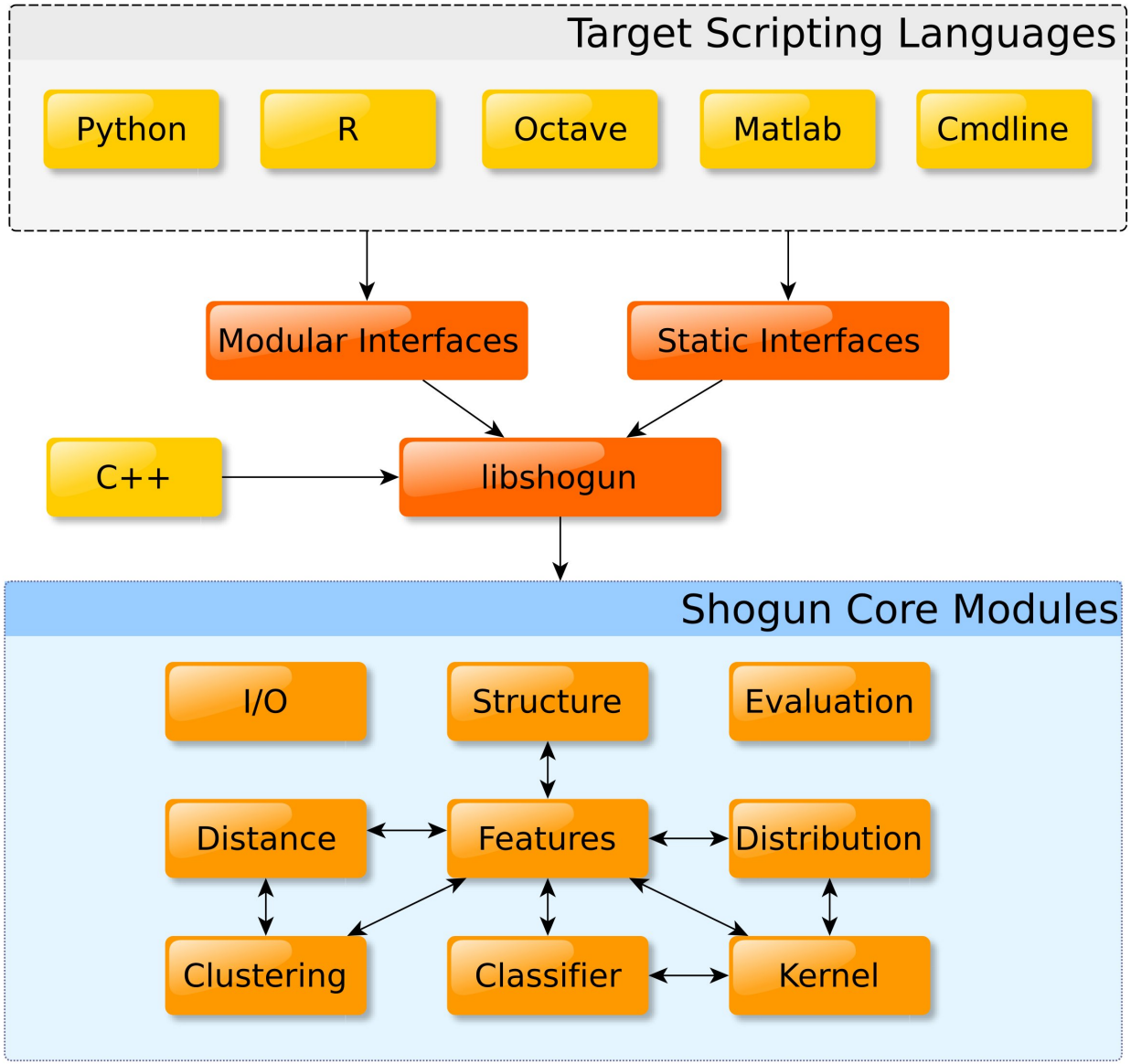
- Implemented in C++
- **Interfaces:** libshogun, python, octave, R, matlab, cmdline, lua, ruby, java, csharp
- **Over 1100 examples**
- Doxygen documentation
- Inline python documentation (e.g. `help(GaussianKernel)`)
- **Testsuite** ensuring that obvious bugs do not slip through

Continuity

Due to being GPLv3'd it survived policy changes at the authors institute and job changes.

将軍

Architecture



Installing SHOGUN

Steps to Install SHOGUN from Source

- `$ git clone \`
  `https://github.com/shogun-toolbox/shogun.git`
- `($ cd shogun && git submodule update --init)`
- `$ cd shogun/src`
- `$ ./configure --interface=` (for libshogun)
- `$ ./configure --interface=python_modular` (for python)
- `$ make && sudo make install`

Don't

- Don't use a static interface if you can avoid it.
- Don't compile for all interfaces (it takes loooong).

Installing SHOGUN

Steps to Install SHOGUN from Source

- `$ git clone \`
  `https://github.com/shogun-toolbox/shogun.git`
- `($ cd shogun && git submodule update --init)`
- `$ cd shogun/src`
- `$ ./configure --interface=` (for libshogun)
- `$ ./configure --interface=python_modular` (for python)
- `$ make && sudo make install`

Don't

- Don't use a static interface if you can avoid it.
- Don't compile for all interfaces (it takes loooong).

Installing SHOGUN

Steps to Install SHOGUN from Source

- `$ git clone \`
  `https://github.com/shogun-toolbox/shogun.git`
- `($ cd shogun && git submodule update --init)`
- `$ cd shogun/src`
- `$ ./configure --interface=` (for libshogun)
- `$ ./configure --interface=python_modular` (for python)
- `$ make && sudo make install`

Don't

- Don't use a static interface if you can avoid it.
- Don't compile for all interfaces (it takes loooong).

First Steps

A) Generate Toy Data

```
from numpy import concatenate as con
from numpy import ones,mean,sign
from numpy.random import randn

num=1000; dist=1; width=2.1; C=1.0

traindata=con((randn(2,num)-dist,
                    randn(2,num)+dist), axis=1)
testdata=con((randn(2,num)-dist,
                   randn(2,num)+dist), axis=1)
trainlab=con((-ones(num), ones(num)))
testlab=con((-ones(num), ones(num)))
```

Simple code example: SVM Training

B) Train and Apply SVM with SHOGUN

```
from shogun.Features import BinaryLabels,RealFeatures
from shogun.Kernel import GaussianKernel
from shogun.Classifier import LibSVM

feats_train=RealFeatures(traindata)
kernel=GaussianKernel(feats_train, feats_train, width)
labels=BinaryLabels(trainlab)
svm=LibSVM(C, kernel, labels)
svm.train()

out=svm.apply(RealFeatures(testdata)).get_labels()
testerr=mean(out!=testlab)
print testerr
```

Shogun Development Process

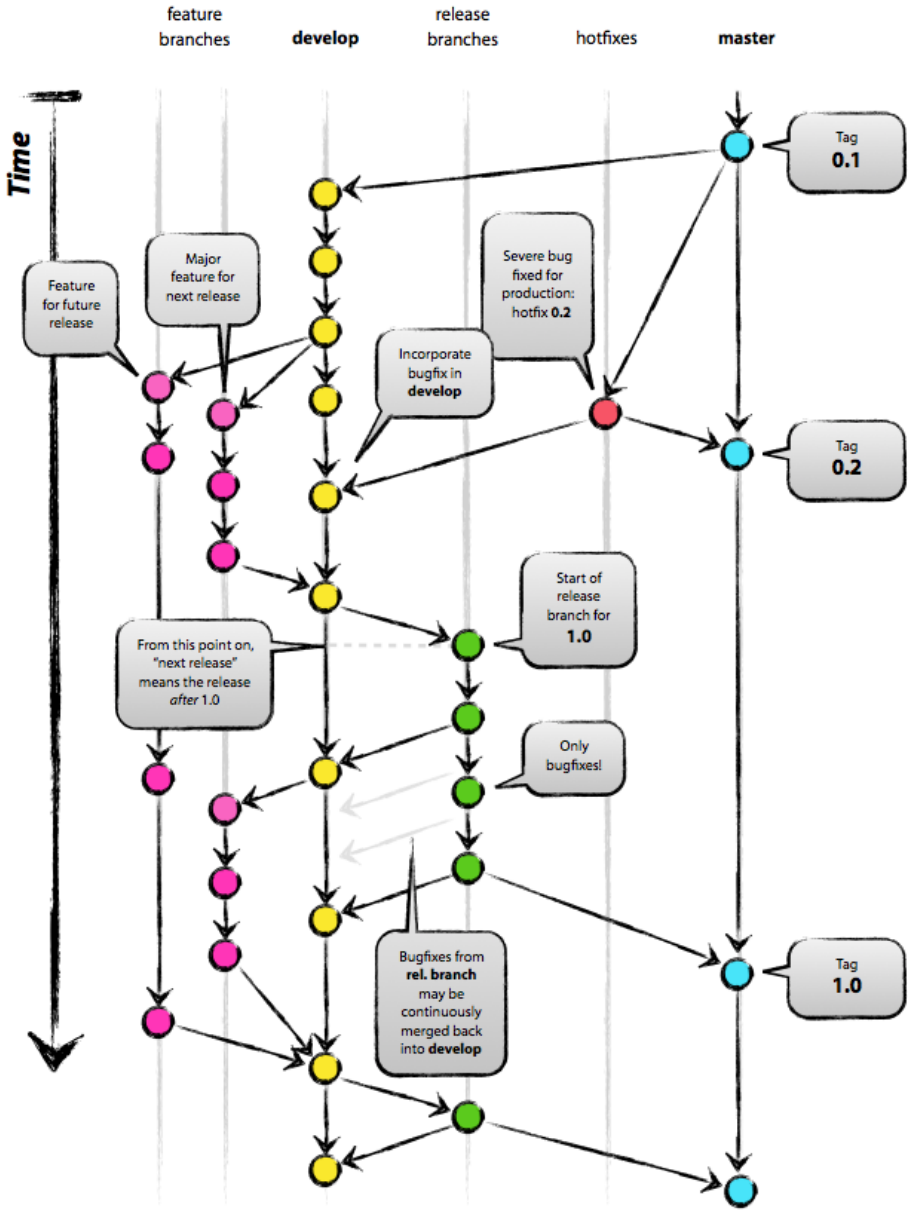
We use git & github

- shogun at github `https://github.com/shogun-toolbox`
- workflow is
  1. fork the project
  2. create a development branch with your changes
  3. do a pull request against the development branch
  4. we will do a code review
  5. eventually merged
- `https://help.github.com/articles/using-pull-requests`

Benefits

- Collaborative code review
- Unit and integration tests help to keep things stable.

git flow model



In a nutshell

- development happens on feature branches
- when a feature is ready it is merged to the development branch
- when we do a release we merge into master and tag the release

`http://nvie.com/posts/a-successful-git-branching-model/`

Developing an Algorithm

Flow to get an algorithm into shogun

- Follow `http://www.shogun-toolbox.org/doc/en/current/developer_tutorial.html` outside of shogun
- write a unit test (shogun uses gtest/gmock), write an example
- move file under the `src/shogun` hierarchy and get interface support

Efficient Native Data Representations

Input Features

- Dense Vectors/Matrices (DenseFeatures)
  - `uint8_t`
  - ⋮
  - float64_t
- Sparse Vectors/Matrices (SparseFeatures)
  - `uint8_t`
  - ⋮
  - float64_t
- Variable Length Vectors/Matrices (String-,MatrixFeatures)
  - `uint8_t`
  - ⋮
  - float64_t

Interfaces (legacy static interface)

Interface Types

- (legacy) Static Interfaces (single object of each type only)
- Modular Interfaces (really object oriented, SWIG based)

Support for all Feature Types

- Dense, Sparse, Strings
- Possible by defining generic get/set functions, e.g.

```
void set_int(int32_t scalar);
void set_real(float64_t scalar);
void set_bool(bool scalar);
void set_vector(float64_t* vector, int32_t len);
void set_matrix(float64_t* m, int32_t rws, int32_t cls);
...
```

$\Rightarrow$ set/get functions for access from python, R, octave, matlab

The Eierlegendewollmilchsau™ Interface

Embed Interface A from Interface B

- possible to run python code from octave
- possible to run octave code from python
- possible to run r code from python
- ...

Demo: Use matplotlib to plot functions from octave.

Modular Python SWIG based Interface

⇒ typemaps for numpy,scipy.sparse,files,lists of strings defined

Wrapping a C++ Object with swig (Kernel)

```
%{
#include <shogun/kernel/GaussianKernel.h>
%}
%rename(GaussianKernel) CGaussianKernel;
%include <shogun/kernel/GaussianKernel.h>
```

Wrapping a C++ Object with swig (Classifier)

```
%{
#include <shogun/classifier/svm/LibSVM.h>
%}
%rename(LibSVM) CLibSVM;
%include <shogun/classifier/svm/LibSVM.h>
```

Unique Features of SHOGUN I

Input Features

- efficient feature representation
- possible to stack together features of arbitrary types (sparse, dense, string) via CombinedFeatures and DotFeatures
- chains of "preprocessors" (e.g. substracting the mean) can be attached to each feature object (on-the-fly pre-processing)

Kernels

- working with custom pre-computed kernels.
- possible to stack together kernels via CombinedKernel (weighted linear combination of a number of sub-kernels, not necessarily working on the same domain)
- kernel weighting can be learned using MKL
- Methods (e.g., SVMs) can be trained using unified interface

Unique Features of SHOGUN II

Large Scale

- multiprocessor parallelization (training with up to 10 million examples and kernels)
- implements COFFIN framework (dynamic feature / example generation; training on 200,000,000 dimensions and 50,000,000 examples)
- streaming features for online methods

Unified Framework

- Interfaces to many languages available at **no coding cost.**
- Serialization - loading from other language possible
- Documentation available, many many examples
- There is a Debian Package, MacOSX
- IRC, Mailing-List, git repository & buildbot infrastructure

将軍

Application

Genomic Signals

- Transcription Start (Sonnenburg et al., 2006)
- Acceptor Splice Site (Sonnenburg et al., 2007)
- Donor Splice Site (Sonnenburg et al., 2007)
- Alternative Splicing (Rätsch et al., 2005)
- Transsplicing (Schweikert et al., 2009)
- Translation Initiation (Sonnenburg et al., 2008)

Genefinding

- Splice form recognition - mSplicer (Rätsch et al. 2008)
- Genefinding - mGene (Schweikert et al., 2009)

Demo I

Support Vector Classification

- Task: separate 2 clouds of gaussian distributed points in 2D

Simple code example: SVM Training

```
lab = Labels(labels)
train = RealFeatures(features)
gk = GaussianKernel(train, train, width)
svm = LibSVM(10.0, gk, lab)
svm.train()
```

Demo II

Support Vector Regression

- Task: learn a sine function

Simple code example: Support Vector Regression

```
lab = Labels(labels)
train = RealFeatures(features)
gk = GaussianKernel(train, train, width)
svm = LibSVR(10.0, gk, lab)
svm.train()
```

Demo III

Clustering

- Task: find clustering of 3 clouds of gaussian distributed points in 2D

Many more available

- dimension reduction
- EM density estimation

⋮

Check them out

- see `examples/<interface>/*`
- see `examples/<interface>/graphical/*`

Summary

SHOGUN Machine Learning Toolbox

- Unified framework, for various interfaces
- Applicable to huge datasets
- Various Algorithms

Documentation, Examples, Source Code

- Implementation `http://www.shogun-toolbox.org`
- Documentation `http://www.shogun-toolbox.org/doc`
- More machine learning software `http://mloss.org`
- Machine Learning Data `http://mldata.org`

We need you!

- Documentation, Examples, Testing, Extensions

将軍