將軍

sho gun

AN INTRODUCTION TO SHOGUN MACHINE LEARNING TOOLBOX

ONE FOR ALL

Pan Deng

CppCon '17, Seattle, WA

SHOGUN MACHINE LEARNING TOOLBOX

Since 1999, SHOGUN...

(ohloh)

SHOGUN MACHINE LEARNING TOOLBOX

Since 1999, SHOGUN...

...has had 16,099 commits made by 194 contributors

representing 278,799 lines of code

(ohloh)

SHOGUN MACHINE LEARNING TOOLBOX

Since 1999, SHOGUN...

...has had 16,099 commits made by 194 contributors

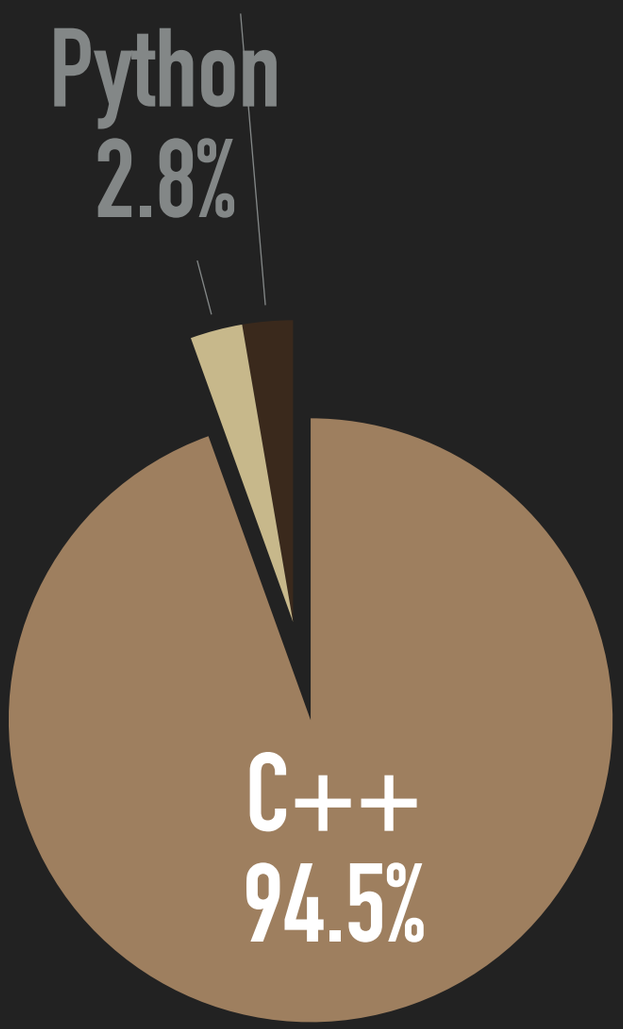
representing 278,799 lines of code

...has a well established, mature codebase

with a well-commented source code

(ohloh)

SHOGUN MACHINE LEARNING TOOLBOX



Since 1999, SHOGUN...

...has had 16,099 commits made by 194 contributors

representing 278,799 lines of code

...has a well established, mature codebase

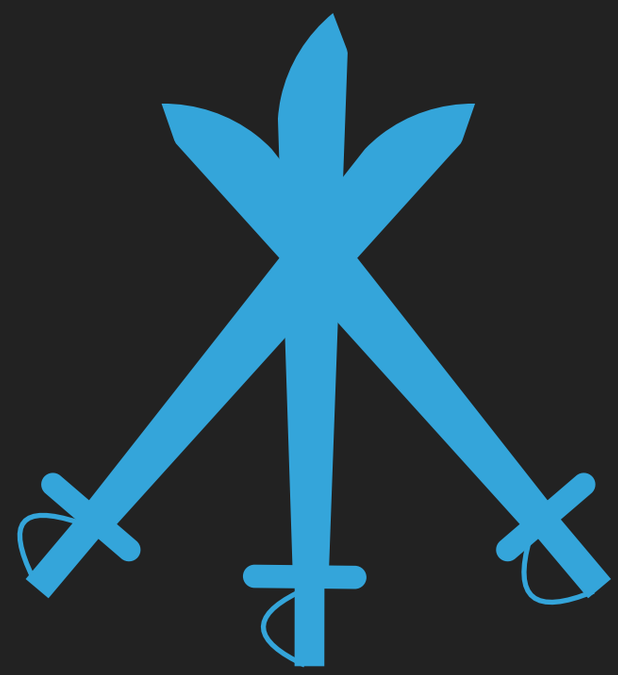
with a well-commented source code

...is mostly written in C++

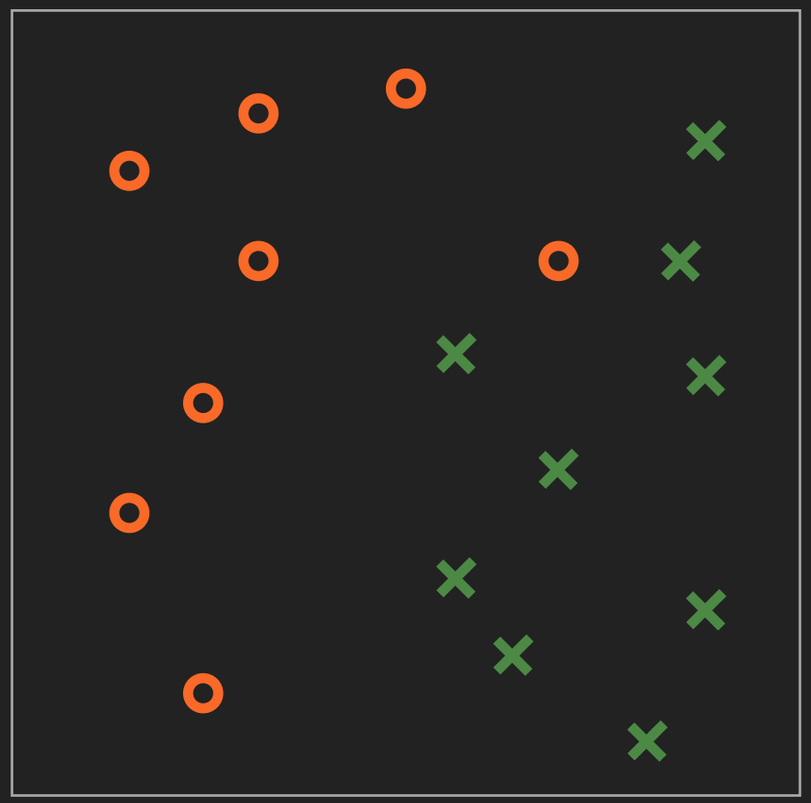
(ohloh)

ONE FOR ALL

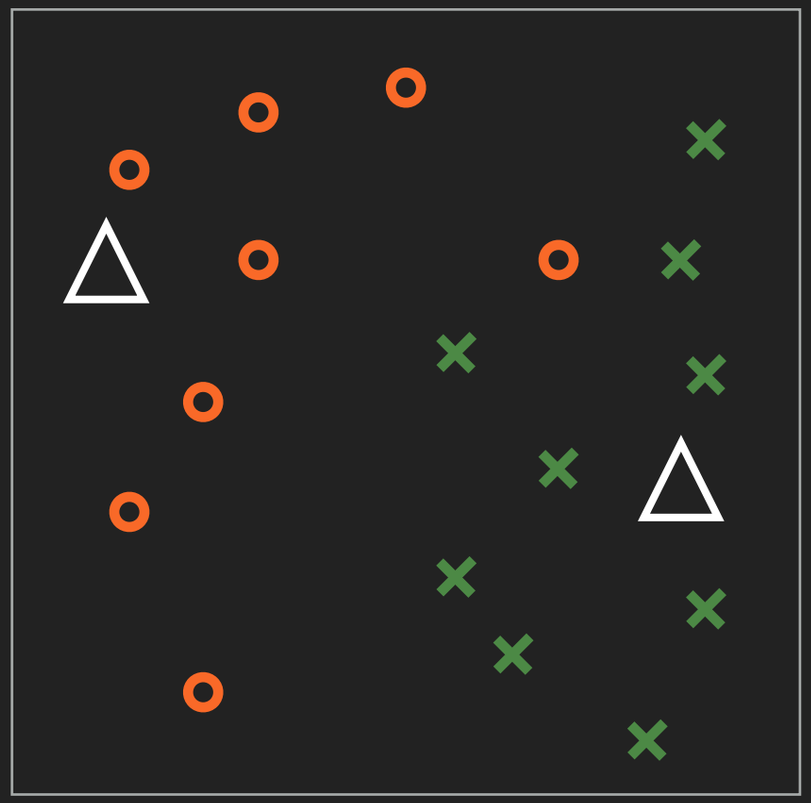
- Interfaces standard to cutting edge ML libraries
- Supports multiple high-level programming languages with unified interface



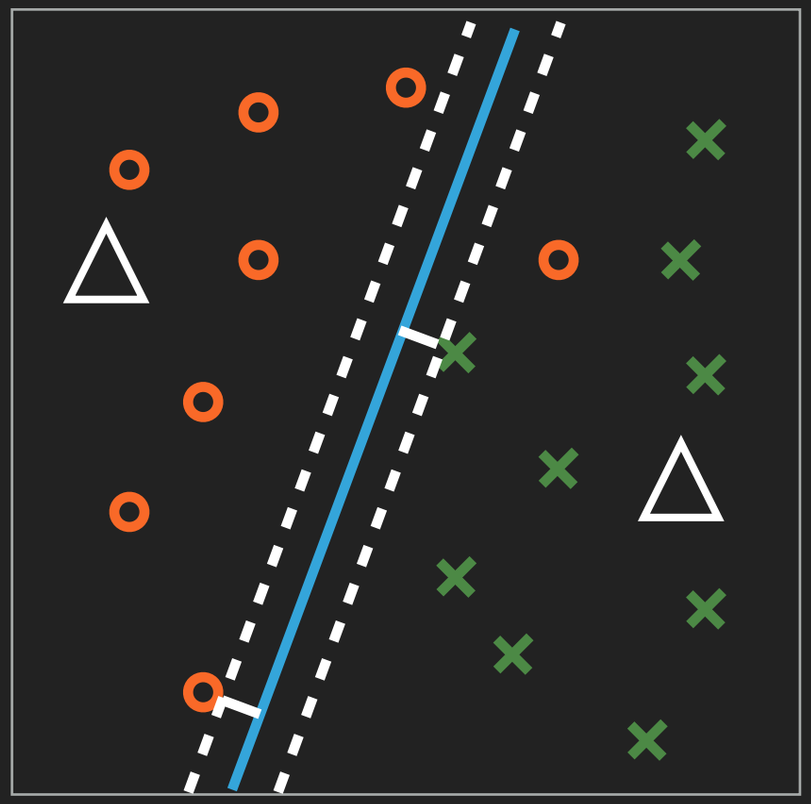
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



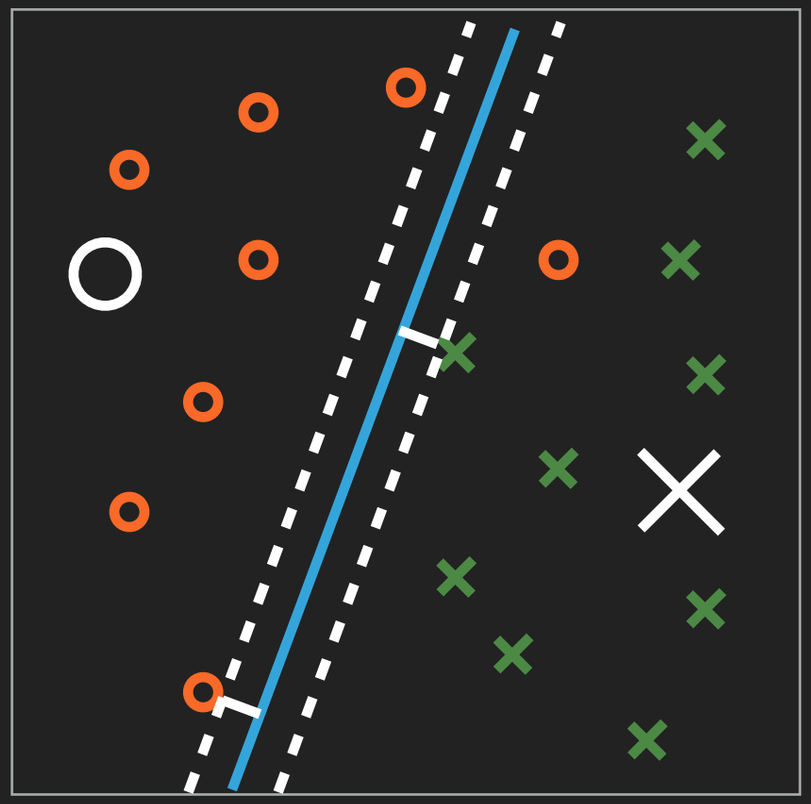
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



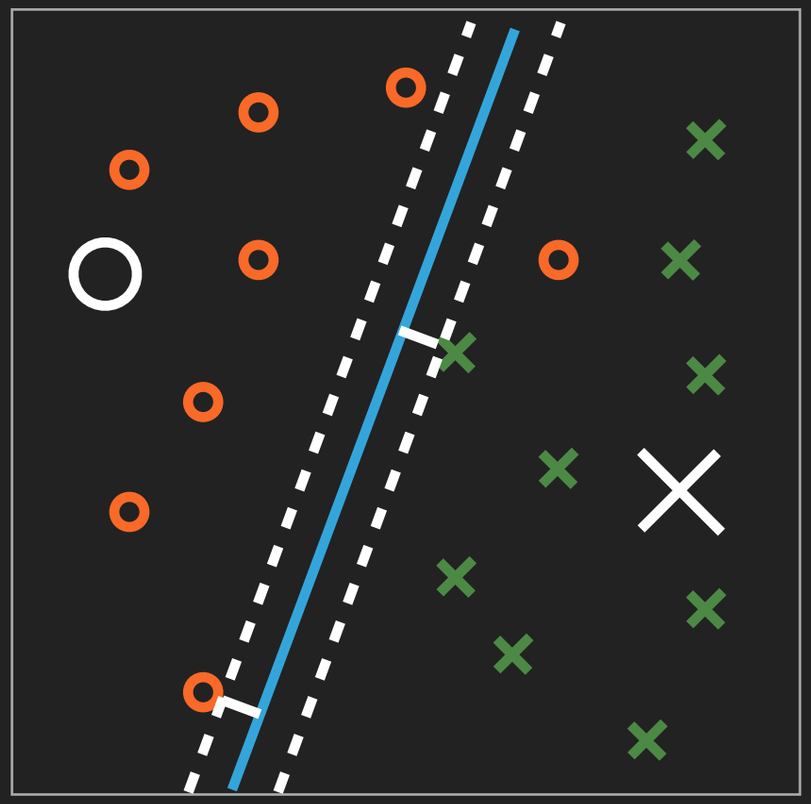
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



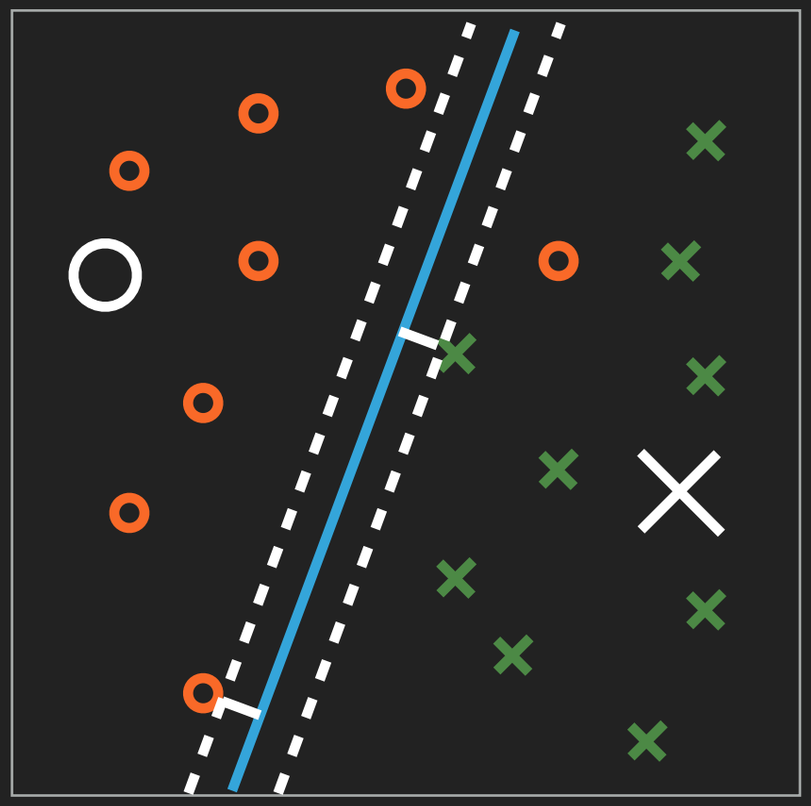
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)

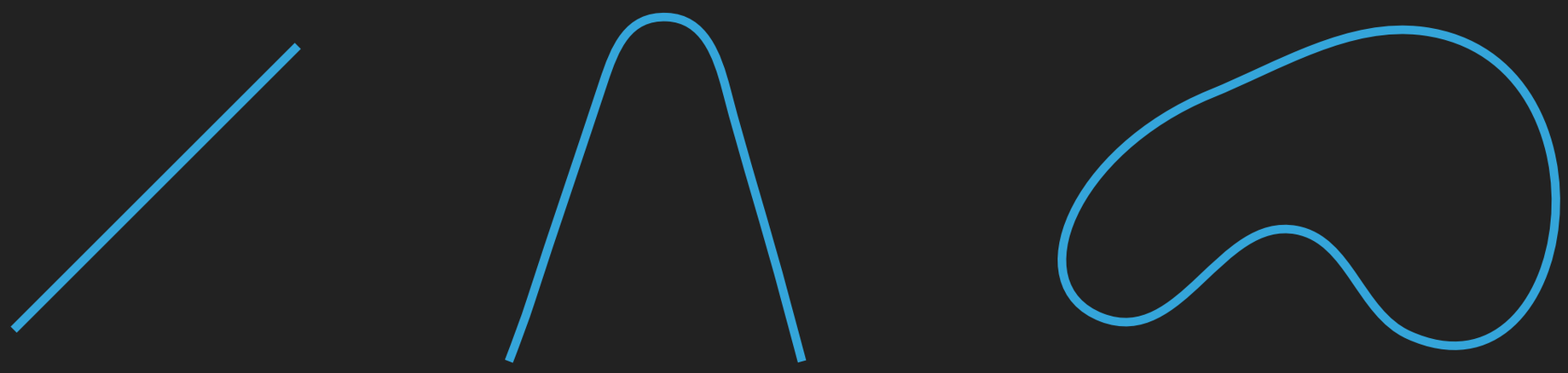


EXAMPLE: SUPPORT VECTOR MACHINE



EXAMPLE: SUPPORT VECTOR MACHINE

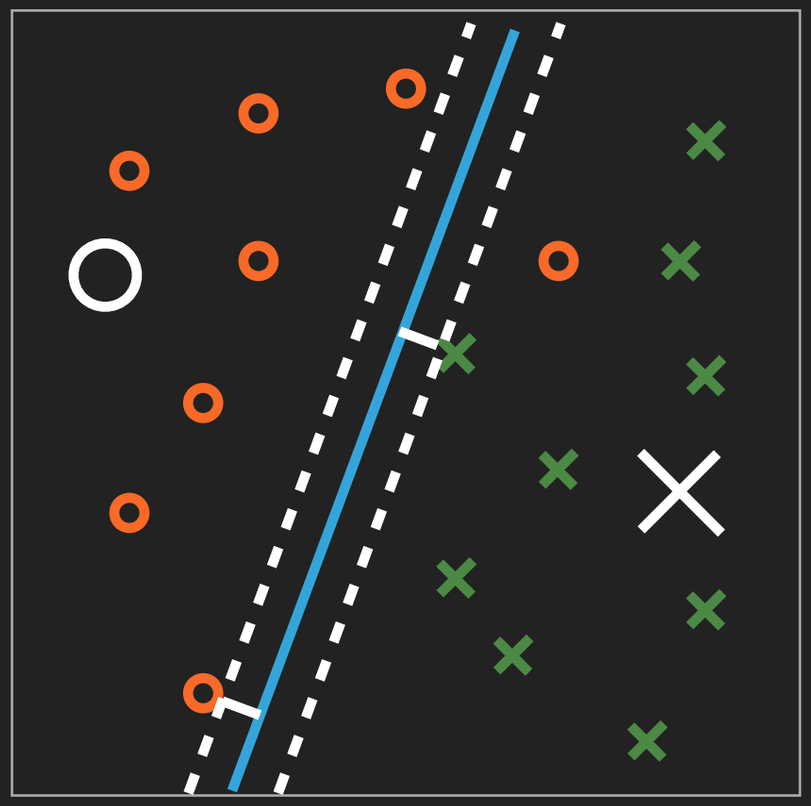




EXAMPLE: SUPPORT VECTOR MACHINE

SVMLight

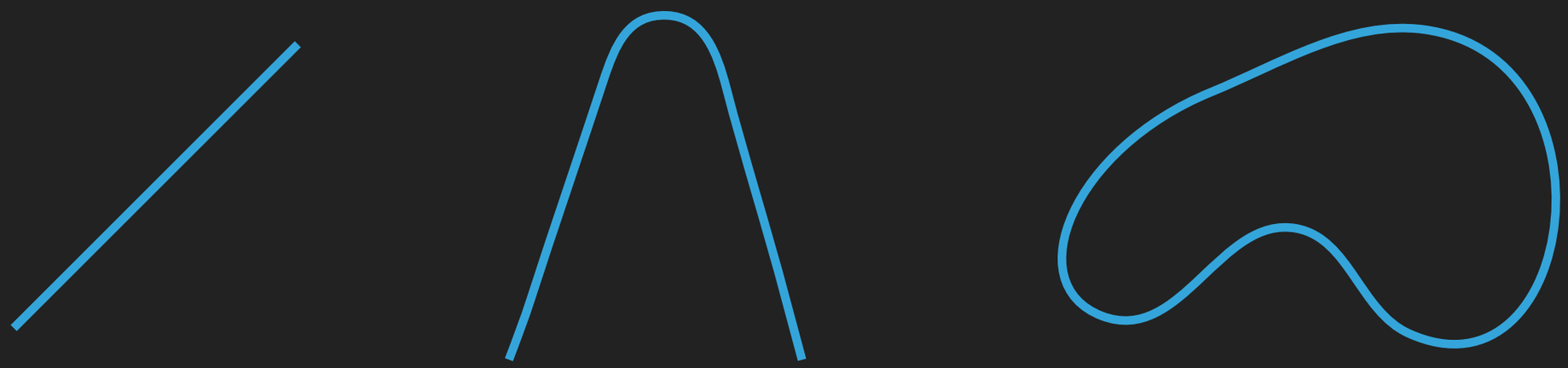
LibSVM



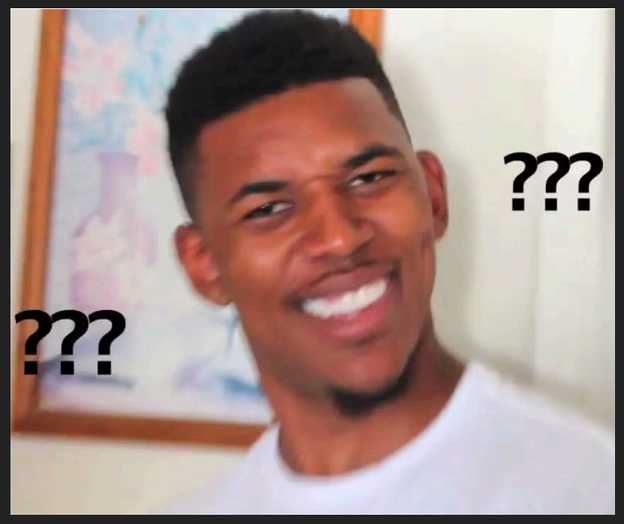
MPDSVM

GNPPSVM

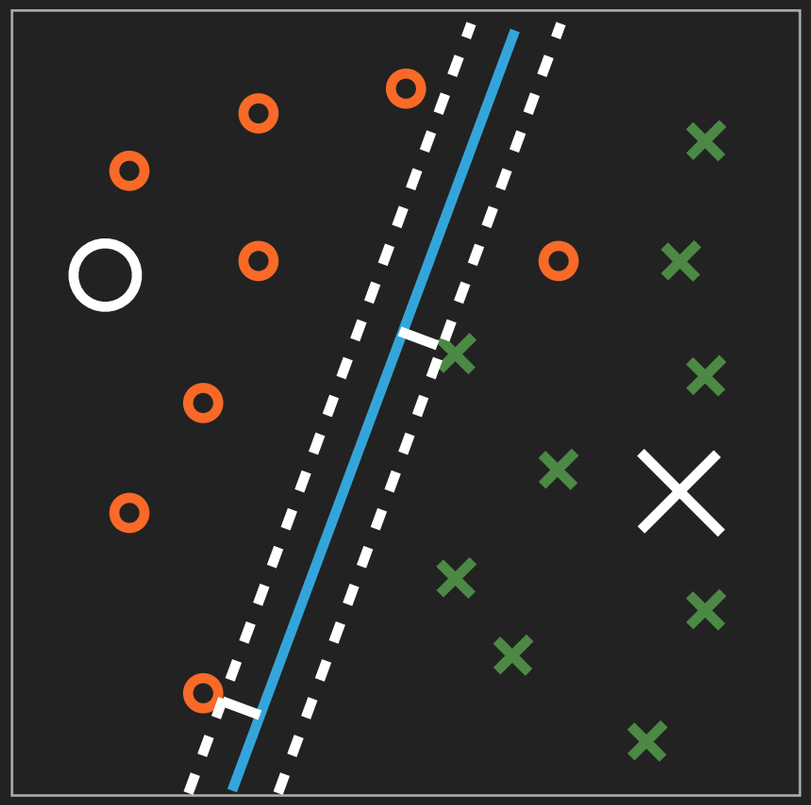
NewtonSVM



EXAMPLE: SUPPORT VECTOR MACHINE



SVMLight

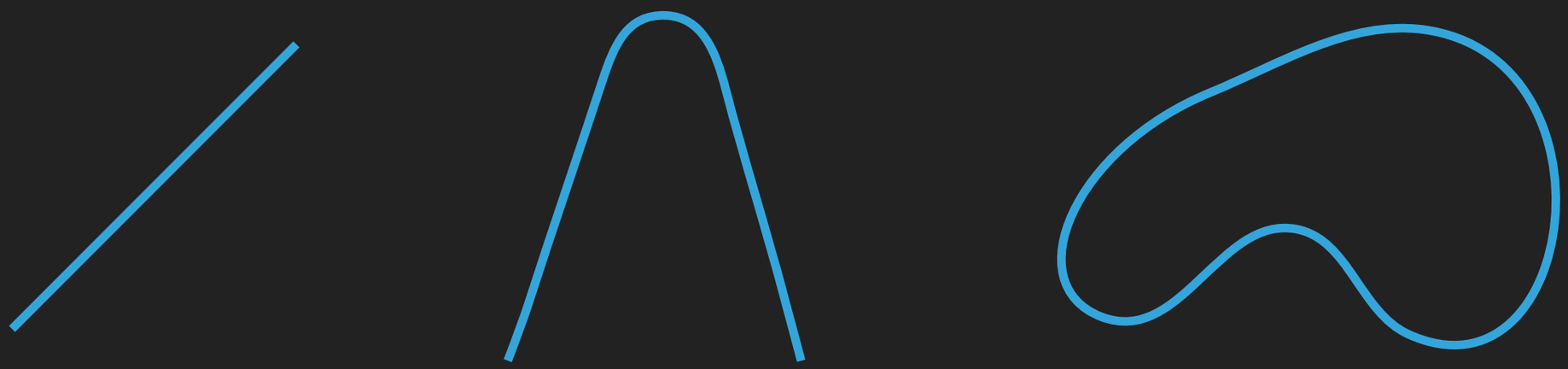


MPDSVM

GNPPSVM

LibSVM

NewtonSVM



INTERFACES STANDARD TO CUTTING EDGE ML LIBRARIES

```
                    ...

                <CLibSVM>

auto svm = some<CLibLinear>(1.0, features_train, labels_train);

svm -> set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE::L2R_L2LOSS_SVC_DUAL);
svm -> set_epsilon(0.001);


svm -> train();

auto labels_predict = svm -> apply_binary(features_test);
```

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

PYTHON

R

JAVA

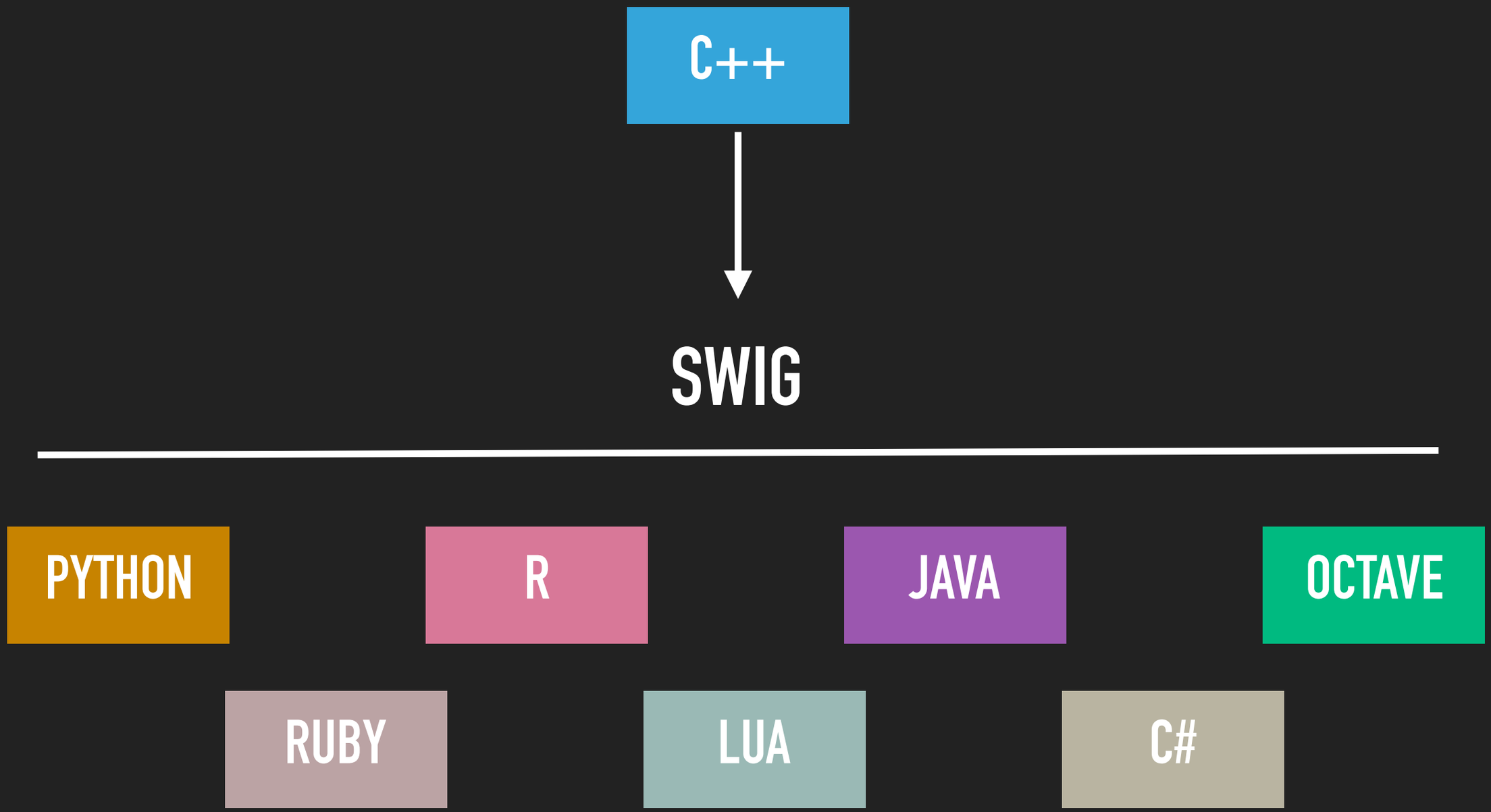
OCTAVE

RUBY

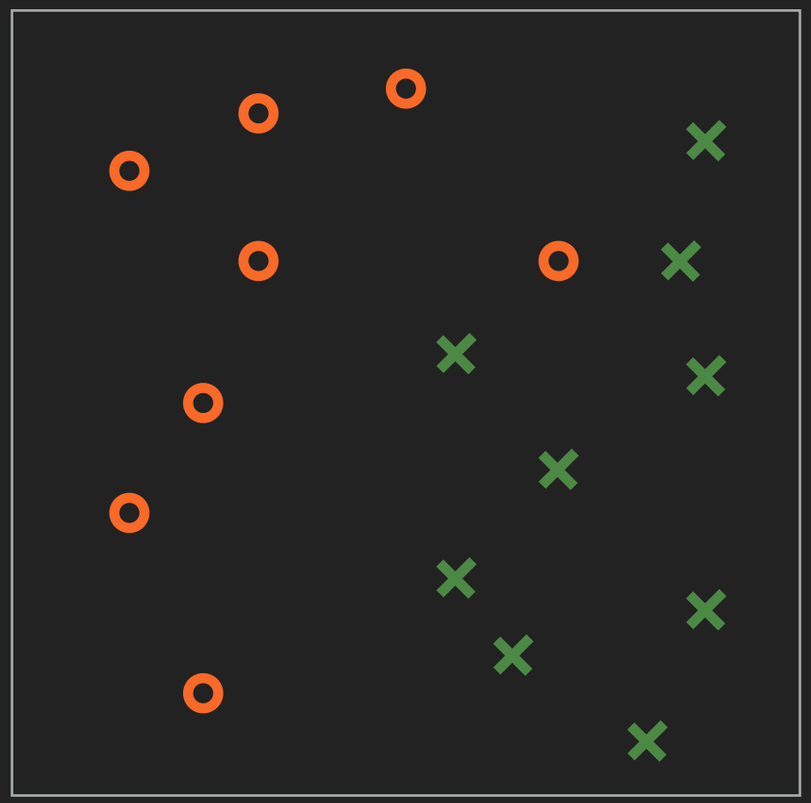
LUA

C#

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

PYTHON

R

JAVA

OCTAVE

RUBY

LUA

C#

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

Shogun

PYTHON

R

JAVA

OCTAVE

RUBY

LUA

C#

EXAMPLE: LINEAR SVM WITH C++

```cpp
#include <shogun/base/init.h>
#include <shogun/base/some.h>
#include <shogun/classifier/svm/LibLinear.h>
#include <shogun/classifier/svm/LibLinear.h>
#include <shogun/evaluation/ContingencyTableEvaluation.h>
#include <shogun/features/DenseFeatures.h>
#include <shogun/io/CSVFile.h>
#include <shogun/io/SerializableAsciiFile.h>
#include <shogun/labels/BinaryLabels.h>
#include <shogun/lib/SGVector.h>
#include <shogun/lib/WrappedObjectArray.h>

using namespace shogun;

int main(int, char*[])
{
init_shogun_with_defaults();

auto f_feats_train = some<CCSVFile>("../../data/classifier_binary_2d_linear_features_train.dat");
auto f_feats_test = some<CCSVFile>("../../data/classifier_binary_2d_linear_features_test.dat");
auto f_labels_train = some<CCSVFile>("../../data/classifier_binary_2d_linear_labels_train.dat");
auto f_labels_test = some<CCSVFile>("../../data/classifier_binary_2d_linear_labels_test.dat");

//![create_features]
auto features_train = some<CDenseFeatures<float64_t>>(f_feats_train);
auto features_test = some<CDenseFeatures<float64_t>>(f_feats_test);
auto labels_train = some<CBinaryLabels>(f_labels_train);
auto labels_test = some<CBinaryLabels>(f_labels_test);
//![create_features]

//![set_parameters]
auto C = 1.0;
auto epsilon = 0.001;
//![set_parameters]

//![create_instance]
auto svm = some<CLibLinear>(C, features_train, labels_train);
svm->set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE::L2R_L2LOSS_SVC);
svm->set_epsilon(epsilon);
//![create_instance]

//![train_and_apply]
svm->train();
auto labels_predict = svm->apply_binary(features_test);
//![train_and_apply]

//![extract_weights_bias]
auto w = svm->get_w();
auto b = svm->get_bias();
//![extract_weights_bias]

//![evaluate_accuracy]
auto eval = some<CAccuracyMeasure>();
auto accuracy = eval->evaluate(labels_predict, labels_test);
//![evaluate_accuracy]

// additional integration testing variables
auto output = labels_predict->get_labels();

exit_shogun();
return 0;
}
```

EXAMPLE: LINEAR SVM WITH PYTHON

```
import numpy as np
from shogun import AccuracyMeasure
from shogun import BinaryLabels
from shogun import CSVFile
from shogun import L2R_L2LOSS_SVC
from shogun import LibLinear
from shogun import RealFeatures
from shogun import SerializableAsciiFile
from shogun import WrappedObjectArray

f_feats_train = CSVFile("../../data/classifier_binary_2d_linear_features_train.dat")
f_feats_test = CSVFile("../../data/classifier_binary_2d_linear_features_test.dat")
f_labels_train = CSVFile("../../data/classifier_binary_2d_linear_labels_train.dat")
f_labels_test = CSVFile("../../data/classifier_binary_2d_linear_labels_test.dat")

#![create_features]
features_train = RealFeatures(f_feats_train)
features_test = RealFeatures(f_feats_test)
labels_train = BinaryLabels(f_labels_train)
labels_test = BinaryLabels(f_labels_test)
#![create_features]

#![set_parameters]
C = 1.0
epsilon = 0.001
#![set_parameters]

#![create_instance]
svm = LibLinear(C, features_train, labels_train)
svm.set_liblinear_solver_type(L2R_L2LOSS_SVC)
svm.set_epsilon(epsilon)
#![create_instance]

#![train_and_apply]
svm.train()
labels_predict = svm.apply_binary(features_test)
#![train_and_apply]

#![extract_weights_bias]
w = svm.get_w()
b = svm.get_bias()
#![extract_weights_bias]

#![evaluate_accuracy]
eval = AccuracyMeasure()
accuracy = eval.evaluate(labels_predict, labels_test)
#![evaluate_accuracy]

# additional integration testing variables
output = labels_predict.get_labels()

# Serialize output for integration testing (automatically generated)
__sg_storage = WrappedObjectArray()
__sg_storage_file = SerializableAsciiFile("linear_svm.dat", 'w')
__sg_storage.append_wrapped_real(C, "C")
__sg_storage.append_wrapped_real(epsilon, "epsilon")
__sg_storage.append_wrapped_real_vector(w, "w")
__sg_storage.append_wrapped_real(b, "b")
__sg_storage.append_wrapped_real(accuracy, "accuracy")
__sg_storage.append_wrapped_real_vector(output, "output")
__sg_storage.save_serializable(__sg_storage_file)
```

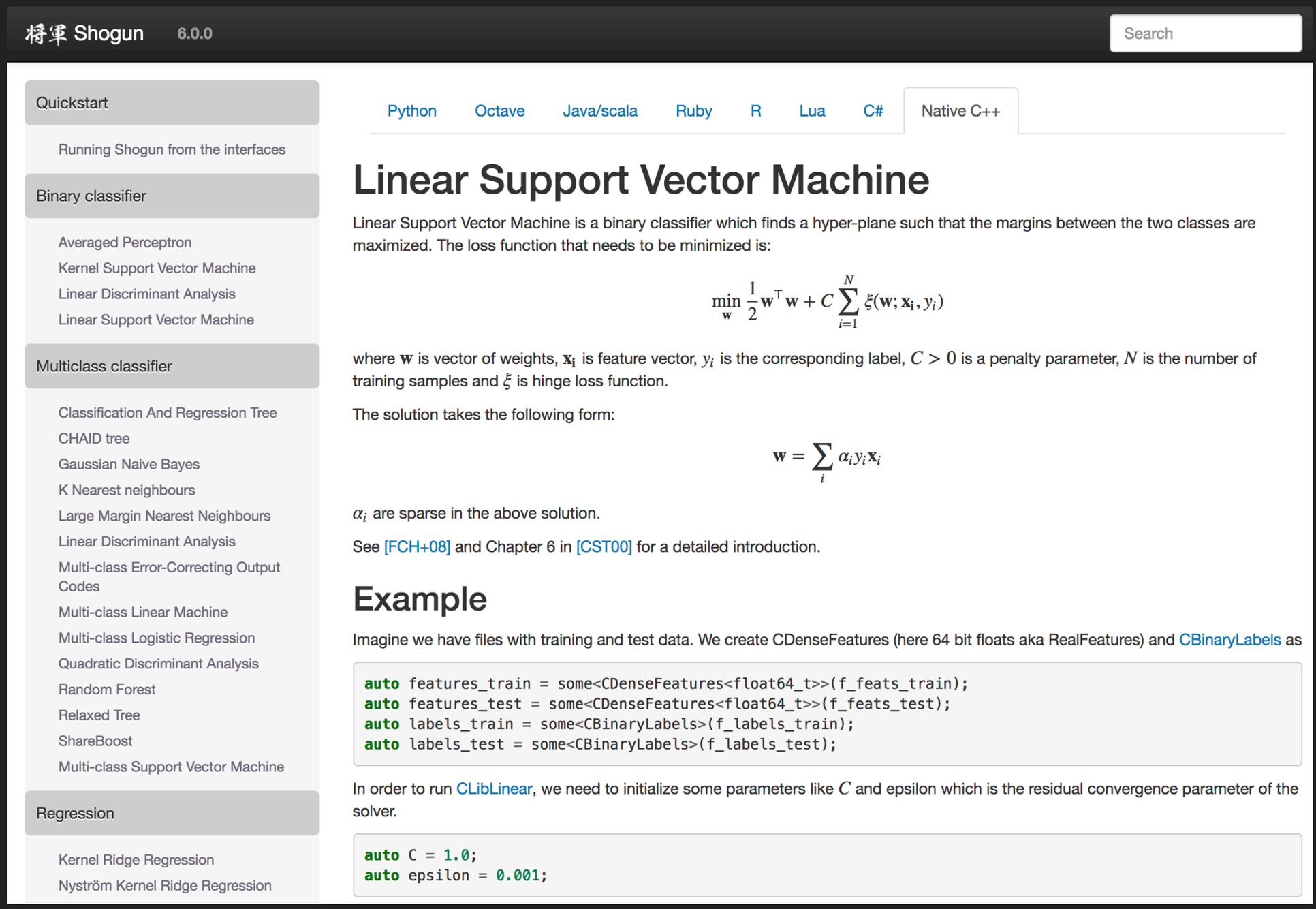
EXAMPLE: LINEAR SVM WITH C#

```
using System;

public class Application {
public static void Main() {
shogun.init_shogun_with_defaults();

CSVFile f_feats_train = new CSVFile("../../data/classifier_binary_2d_linear_features_train.dat");
CSVFile f_feats_test = new CSVFile("../../data/classifier_binary_2d_linear_features_test.dat");
CSVFile f_labels_train = new CSVFile("../../data/classifier_binary_2d_linear_labels_train.dat");
CSVFile f_labels_test = new CSVFile("../../data/classifier_binary_2d_linear_labels_test.dat");

//![create_features]
RealFeatures features_train = new RealFeatures(f_feats_train);
RealFeatures features_test = new RealFeatures(f_feats_test);
BinaryLabels labels_train = new BinaryLabels(f_labels_train);
BinaryLabels labels_test = new BinaryLabels(f_labels_test);
//![create_features]

//![set_parameters]
double C = 1.0;
double epsilon = 0.001;
//![set_parameters]

//![create_instance]
LibLinear svm = new LibLinear(C, features_train, labels_train);
svm.set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE.L2R_L2LOSS_SVC);
svm.set_epsilon(epsilon);
//![create_instance]

//![train_and_apply]
svm.train();
BinaryLabels labels_predict = svm.apply_binary(features_test);
//![train_and_apply]

//![extract_weights_bias]
double[] w = svm.get_w();
double b = svm.get_bias();
//![extract_weights_bias]

//![evaluate_accuracy]
AccuracyMeasure eval = new AccuracyMeasure();
double accuracy = eval.evaluate(labels_predict, labels_test);
//![evaluate_accuracy]

// additional integration testing variables
double[] output = labels_predict.get_labels();

// Serialize output for integration testing (automatically generated)
WrappedObjectArray __sg_storage = new WrappedObjectArray();
SerializableAsciiFile __sg_storage_file = new SerializableAsciiFile("linear_svm.dat", 'w');
__sg_storage.append_wrapped_real(C, "C");
__sg_storage.append_wrapped_real(epsilon, "epsilon");
__sg_storage.append_wrapped_real_vector(w, "w");
__sg_storage.append_wrapped_real(b, "b");
__sg_storage.append_wrapped_real(accuracy, "accuracy");
__sg_storage.append_wrapped_real_vector(output, "output");
__sg_storage.save_serializable(__sg_storage_file);

}
}
```

EXAMPLE: LINEAR SVM WITH JAVA

```java
import org.jblas.DoubleMatrix;
import org.jblas.FloatMatrix;

import org.shogun.shogun;
import org.shogun.AccuracyMeasure;
import org.shogun.BinaryLabels;
import org.shogun.CSVFile;
import org.shogun.LIBLINEAR_SOLVER_TYPE;
import org.shogun.LibLinear;
import org.shogun.RealFeatures;
import org.shogun.RealVector;
import org.shogun.SerializableAsciiFile;
import org.shogun.WrappedObjectArray;

public class linear_svm {
static {
System.loadLibrary("shogun");
}

public static void main(String argv[]) {
shogun.init_shogun_with_defaults();

CSVFile f_feats_train = new CSVFile("../../data/classifier_binary_2d_linear_features_train.dat");
CSVFile f_feats_test = new CSVFile("../../data/classifier_binary_2d_linear_features_test.dat");
CSVFile f_labels_train = new CSVFile("../../data/classifier_binary_2d_linear_labels_train.dat");
CSVFile f_labels_test = new CSVFile("../../data/classifier_binary_2d_linear_labels_test.dat");

//![create_features]
RealFeatures features_train = new RealFeatures(f_feats_train);
RealFeatures features_test = new RealFeatures(f_feats_test);
BinaryLabels labels_train = new BinaryLabels(f_labels_train);
BinaryLabels labels_test = new BinaryLabels(f_labels_test);
//![create_features]

//![set_parameters]
double C = 1.0;
double epsilon = 0.001;
//![set_parameters]

//![create_instance]
LibLinear svm = new LibLinear(C, features_train, labels_train);
svm.set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE.L2R_L2LOSS_SVC);
svm.set_epsilon(epsilon);
//![create_instance]

//![train_and_apply]
svm.train();
BinaryLabels labels_predict = svm.apply_binary(features_test);
//![train_and_apply]

//![extract_weights_bias]
DoubleMatrix w = svm.get_w();
double b = svm.get_bias();
//![extract_weights_bias]

//![evaluate_accuracy]
AccuracyMeasure eval = new AccuracyMeasure();
double accuracy = eval.evaluate(labels_predict, labels_test);
//![evaluate_accuracy]

// additional integration testing variables
DoubleMatrix output = labels_predict.get_labels();

// Serialize output for integration testing (automatically generated)
WrappedObjectArray __sg_storage = new WrappedObjectArray();
SerializableAsciiFile __sg_storage_file = new SerializableAsciiFile("linear_svm.dat", 'w');
__sg_storage.append_wrapped_real(C, "C");
__sg_storage.append_wrapped_real(epsilon, "epsilon");
__sg_storage.append_wrapped_real_vector(w, "w");
__sg_storage.append_wrapped_real(b, "b");
__sg_storage.append_wrapped_real(accuracy, "accuracy");
__sg_storage.append_wrapped_real_vector(output, "output");
__sg_storage.save_serializable(__sg_storage_file);

}
}
```

http://shogun.ml/examples/latest/index.html



- http://shogun.ml/showroom

FUTURE PLANS

- Interface Tensorflow, Stan and etc.
- Interface Matlab and Javascript with SWIG
- Migrate to C++ 17
- ...

- Get more developers involved!

Part of NUMFOCUS

THANKS FOR YOUR ATTENTION

- http://shogun.ml/
- https://github.com/shogun-toolbox/shogun
- IRC: #shogun