



AN INTRODUCTION TO **SHOGUN** MACHINE LEARNING TOOLBOX

ONE FOR ALL

Pan Deng

CppCon '17, Seattle, WA

SHOGUN MACHINE LEARNING TOOLBOX

Since 1999, SHOGUN...

SHOGUN MACHINE LEARNING TOOLBOX

Since 1999, SHOGUN...

...has had 16,099 commits made by 194 contributors
representing 278,799 lines of code

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

SHOGUN MACHINE LEARNING TOOLBOX



Since 1999, SHOGUN...

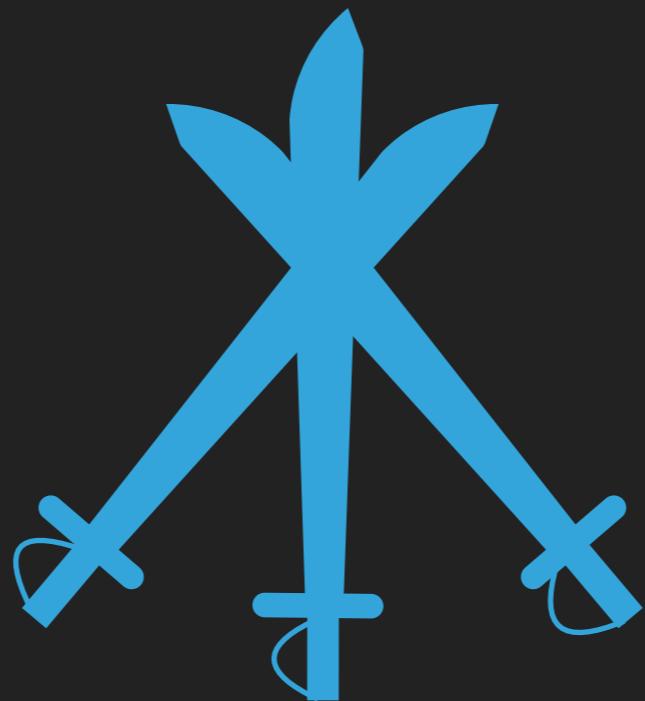
...has had 16,099 commits made by 194 contributors
representing 278,799 lines of code

...has a well established, mature codebase
with a well-commented source code

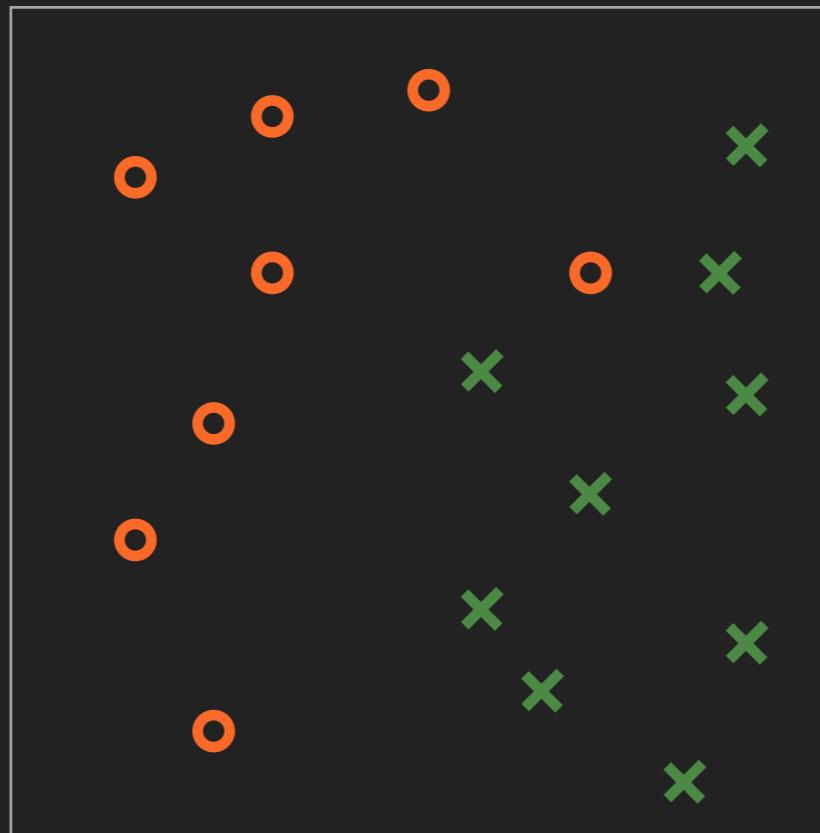
...is mostly written in C++

ONE FOR ALL

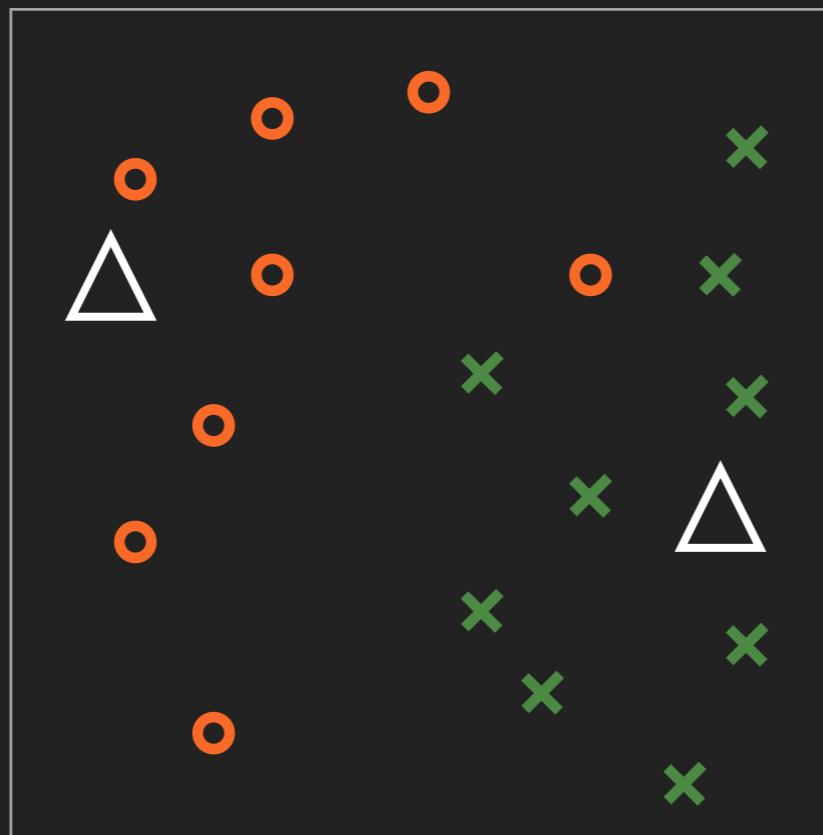
- ▶ Interfaces standard to cutting edge ML libraries
- ▶ Supports multiple high-level programming languages with unified interface



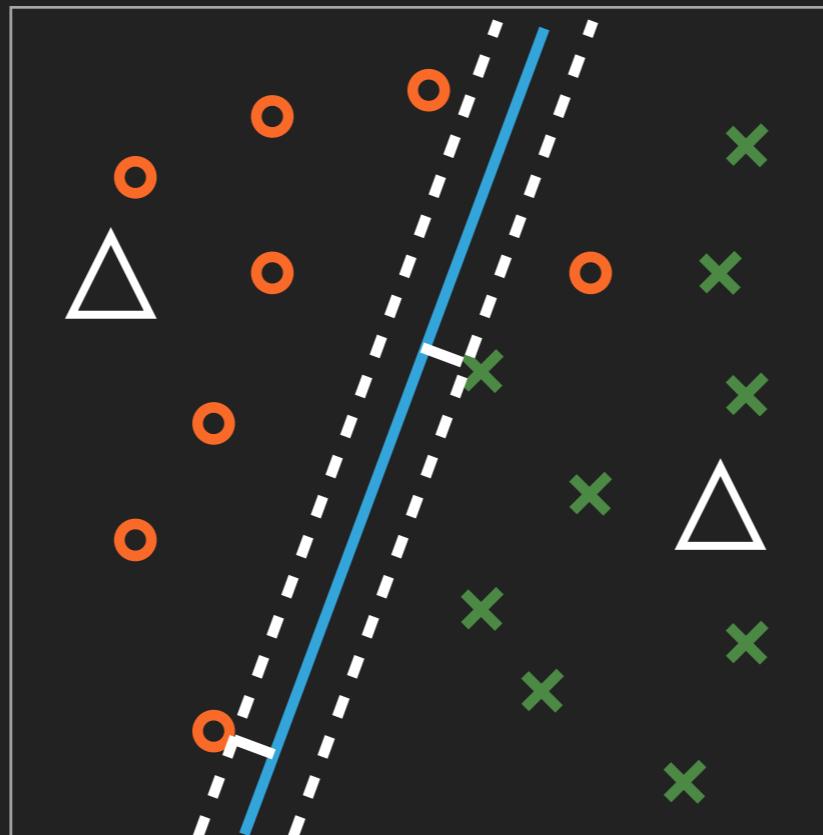
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



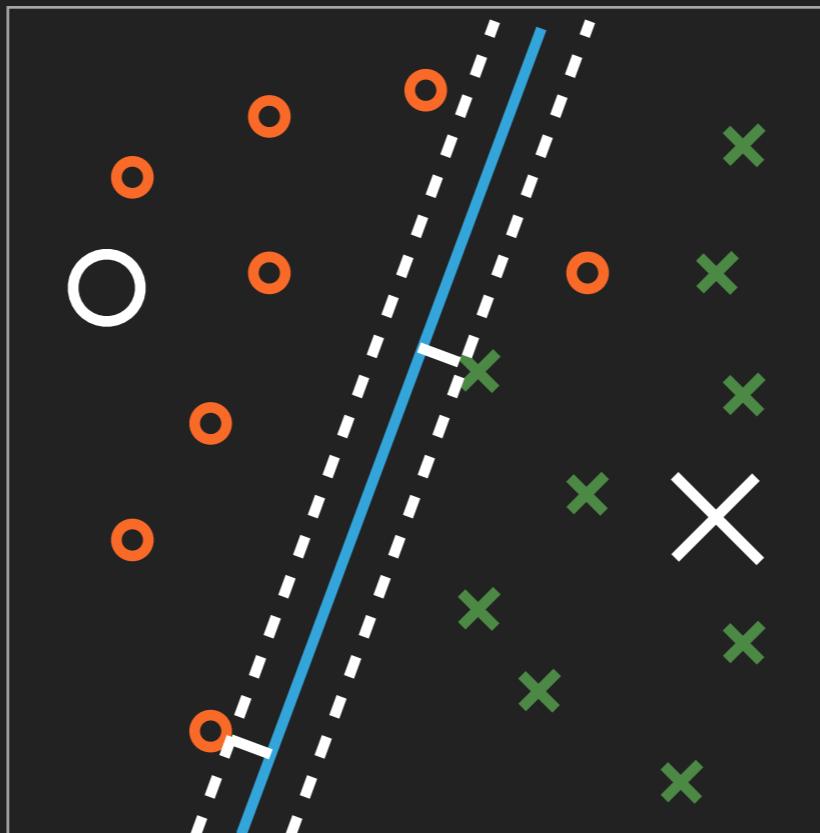
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



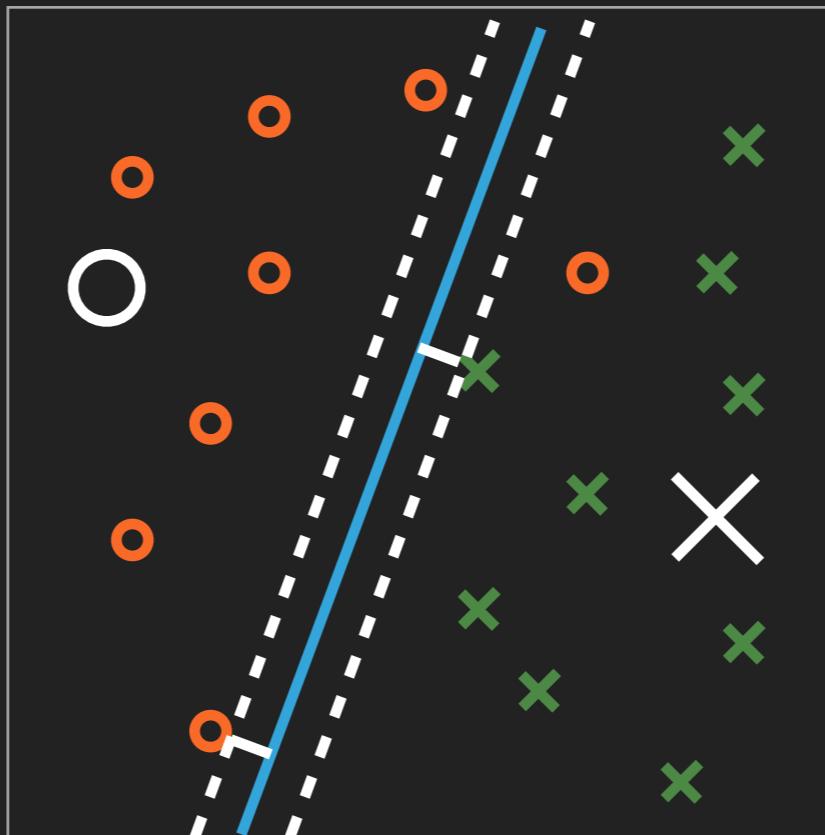
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



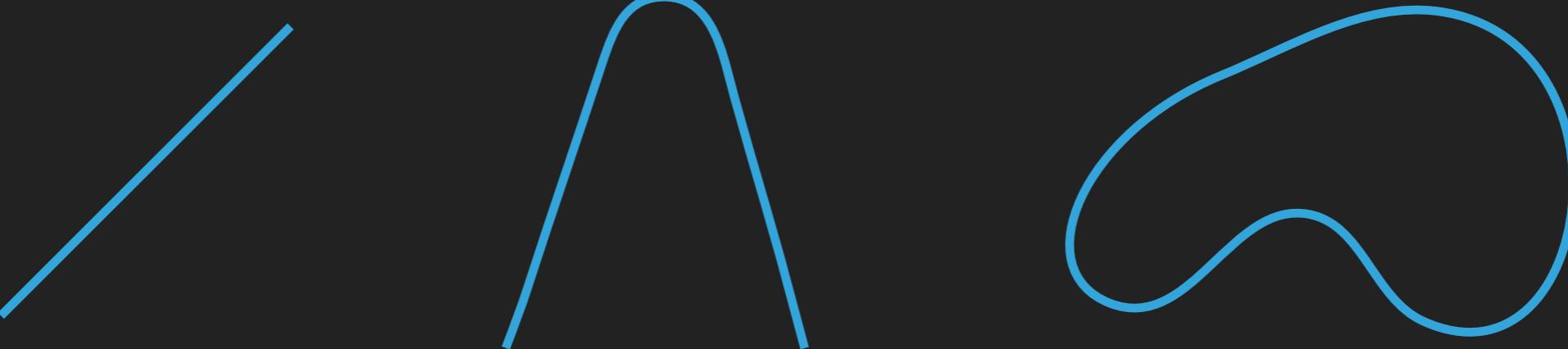
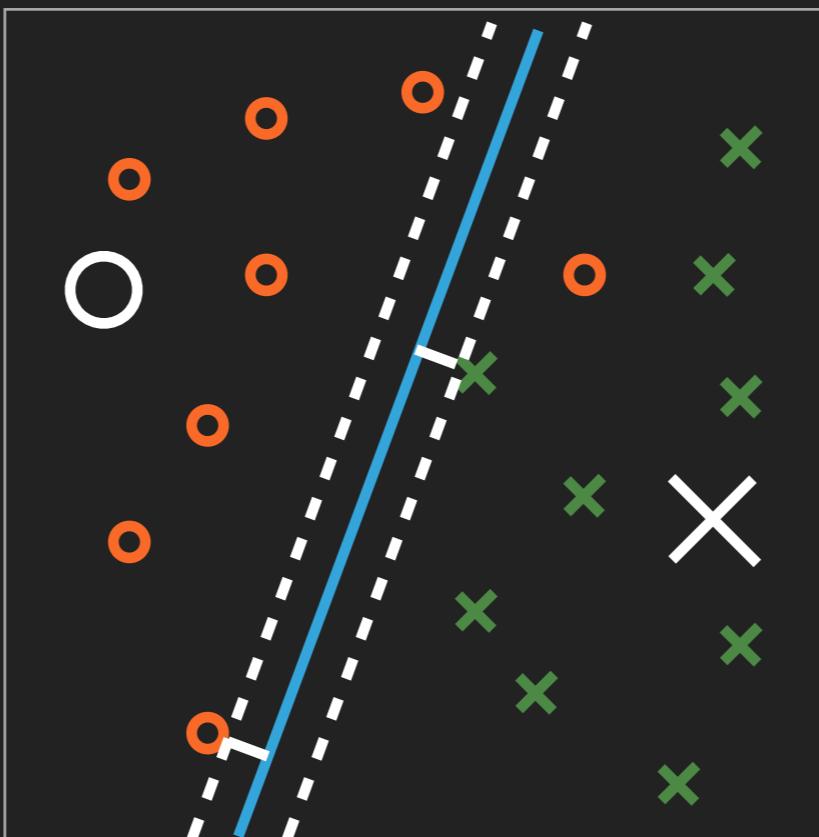
EXAMPLE: LINEAR SUPPORT VECTOR MACHINE (SVM)



EXAMPLE: SUPPORT VECTOR MACHINE



EXAMPLE: SUPPORT VECTOR MACHINE



EXAMPLE: SUPPORT VECTOR MACHINE

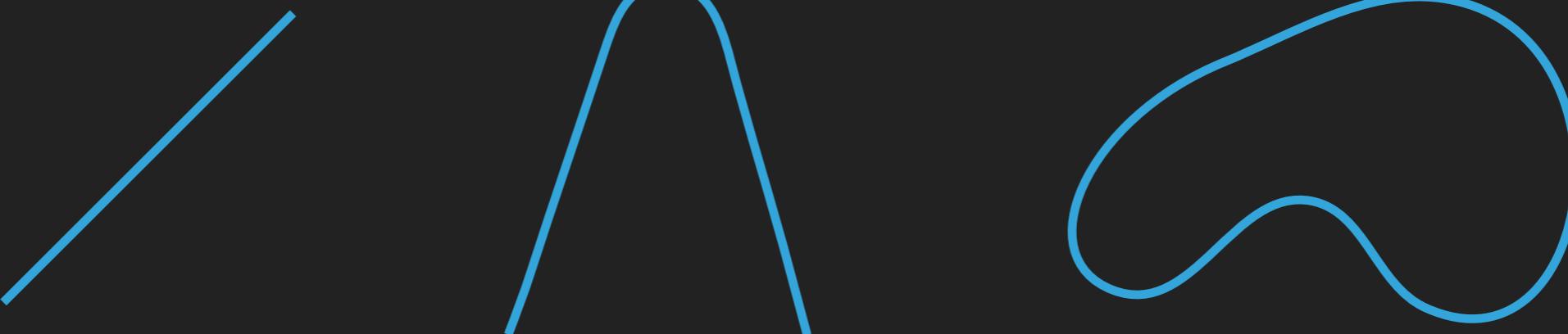
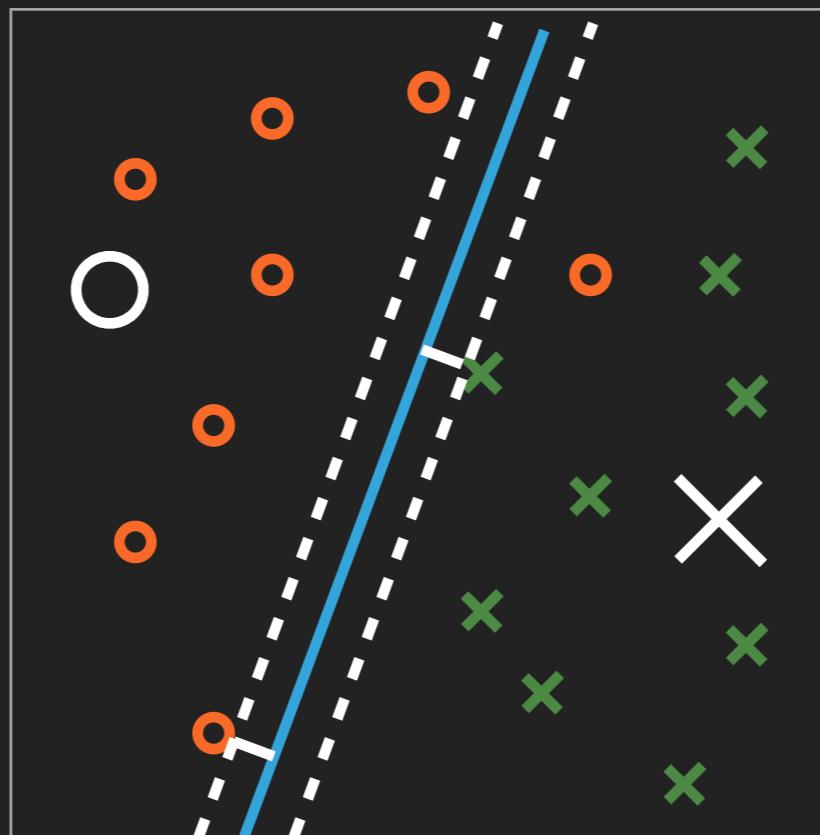
SVMLight

LibSVM

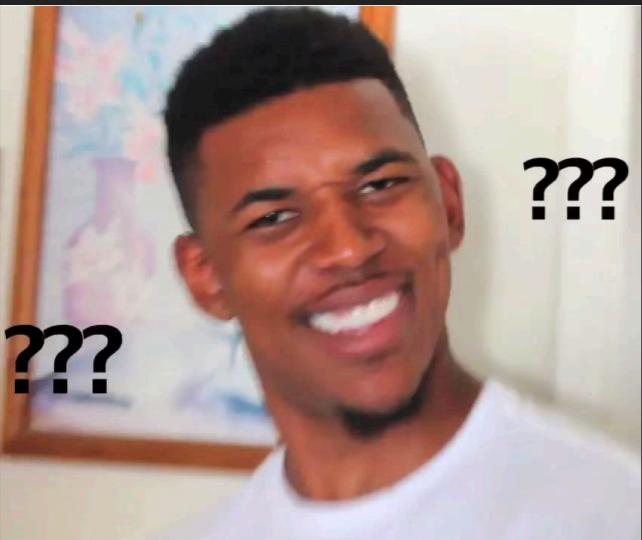
MPDSVM

GNPPSVM

NewtonSVM

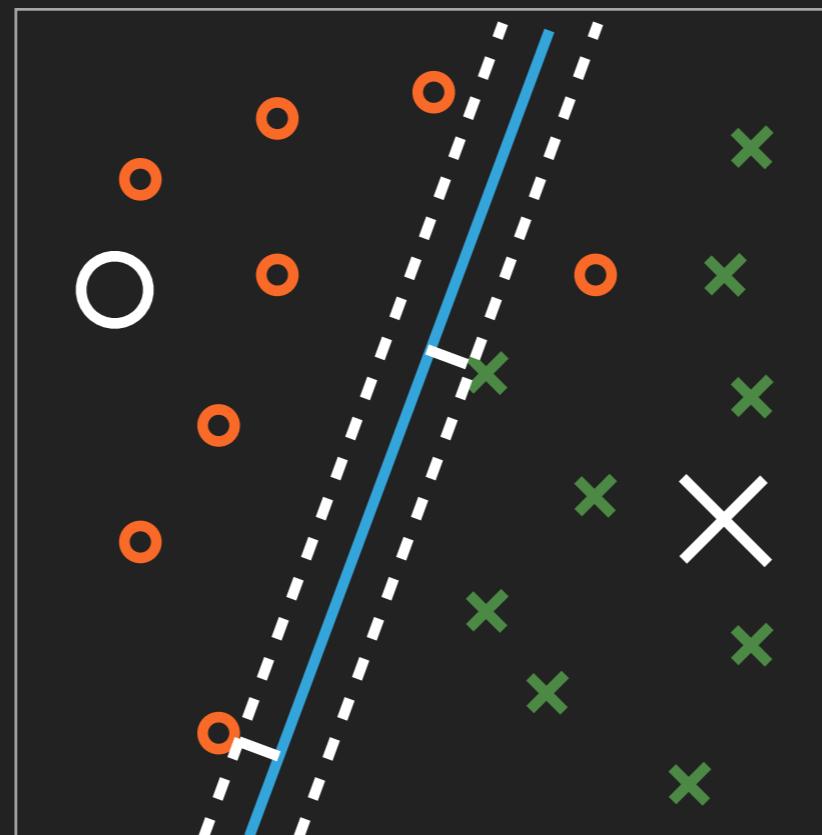


EXAMPLE: SUPPORT VECTOR MACHINE



SVMLight

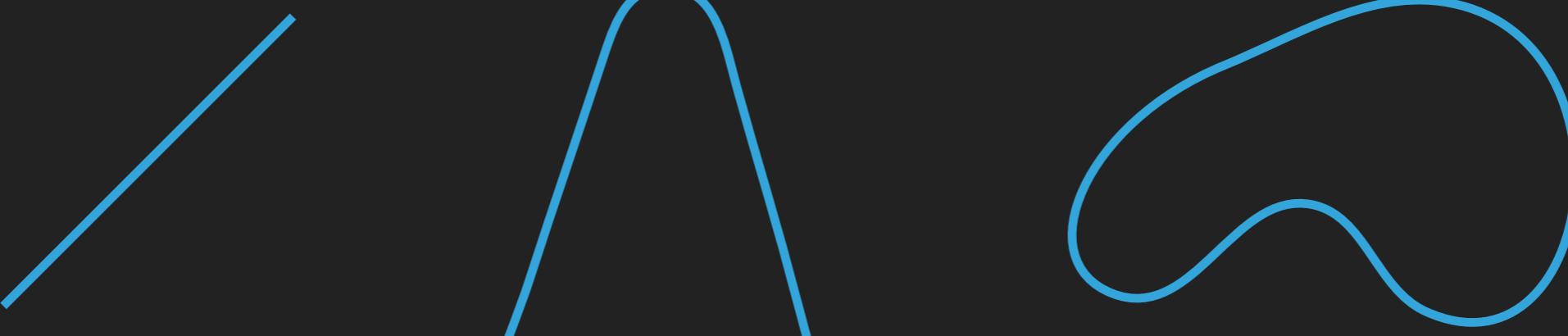
LibSVM



MPDSVM

GNPPSVM

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

C++



SWIG

PYTHON

R

JAVA

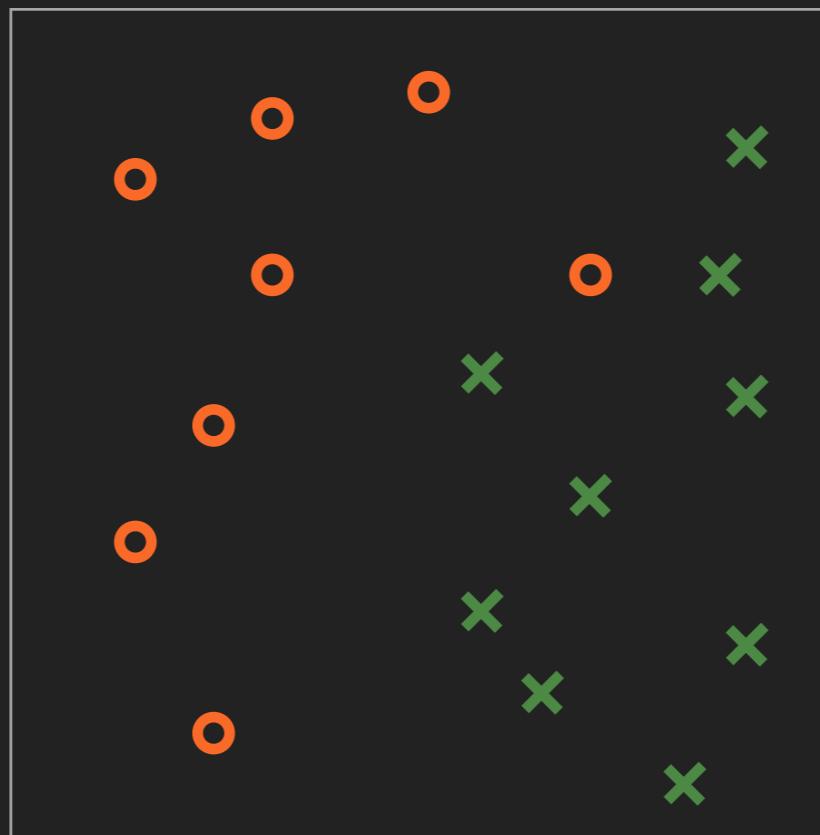
OCTAVE

RUBY

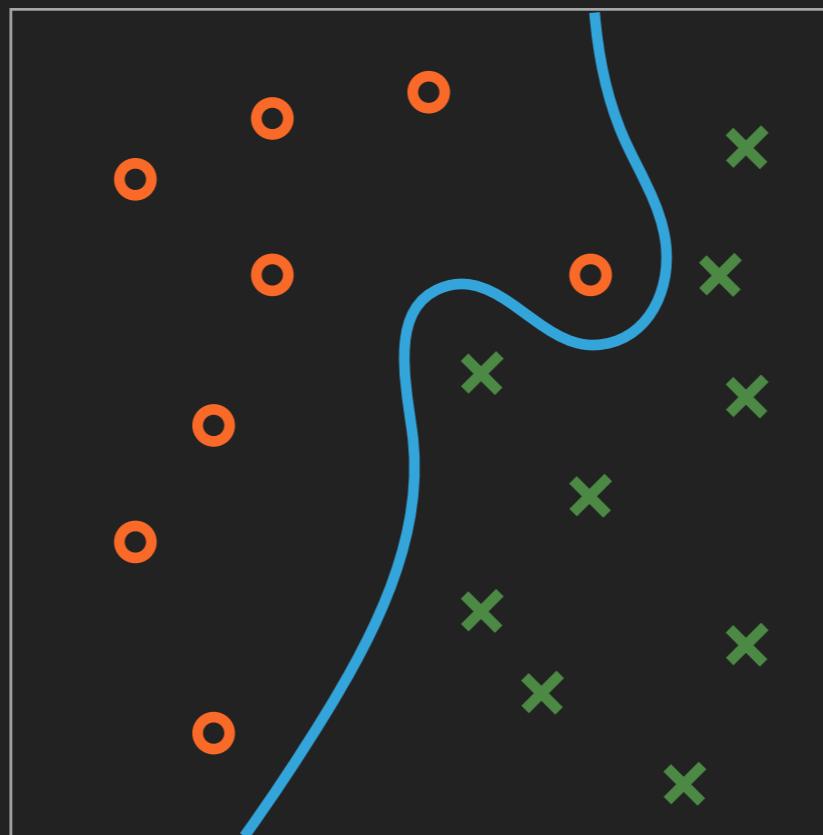
LUA

C#

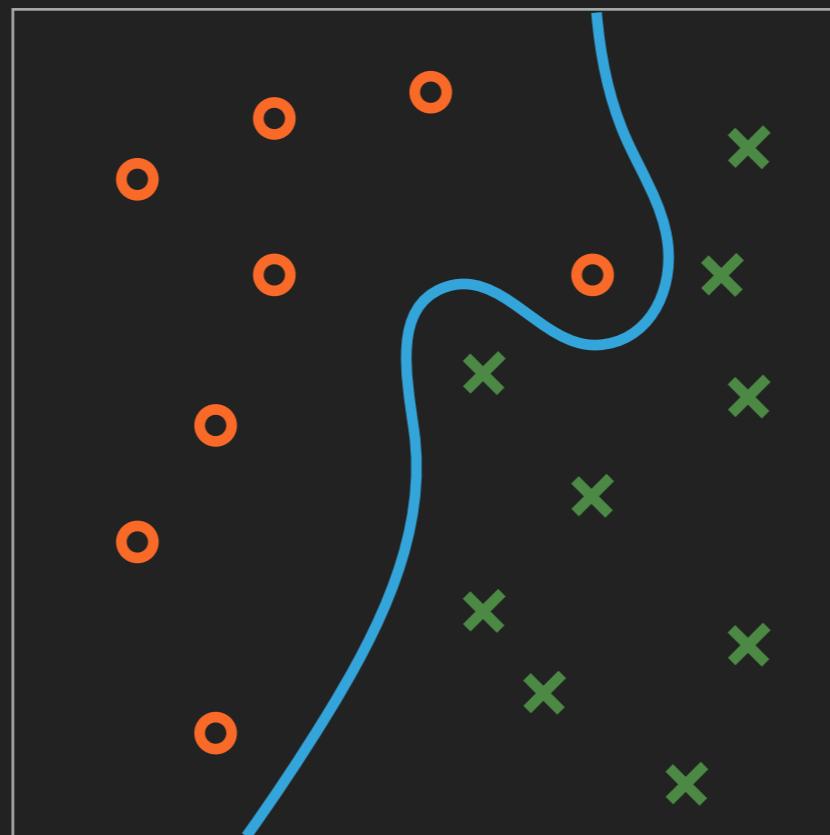
SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

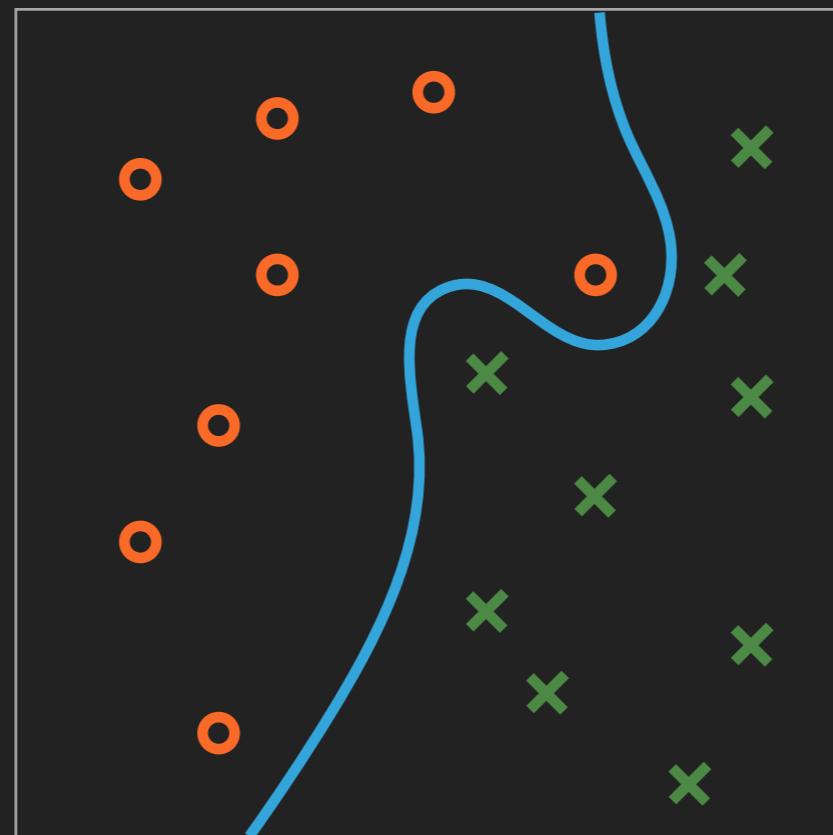


I AM THE BEST!



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

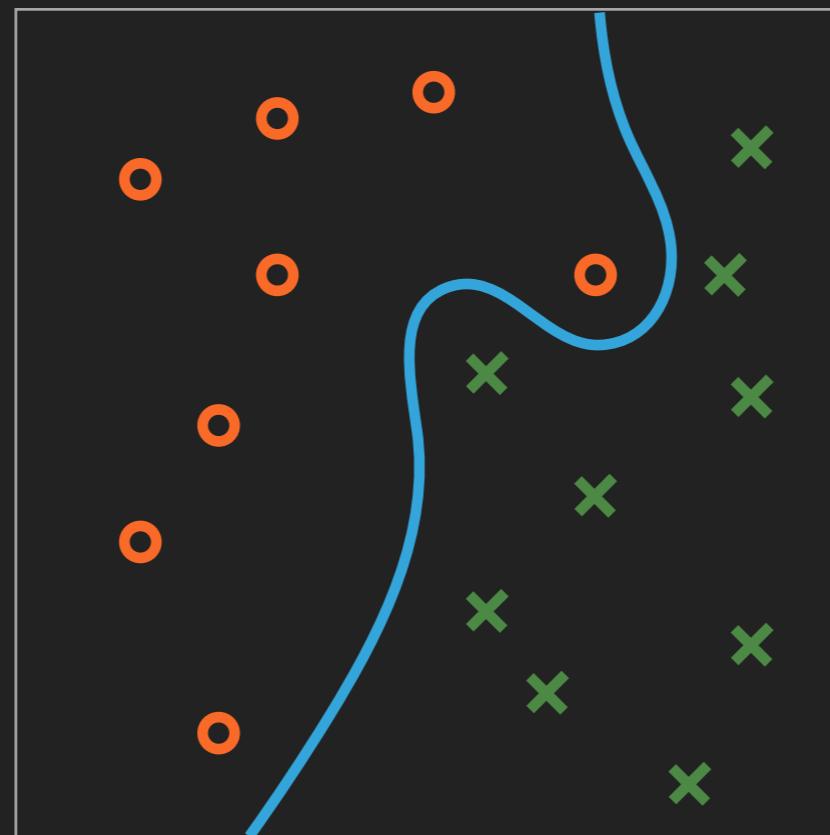


I AM THE BEST!



SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++



PYTHON

R

JAVA

OCTAVE

RUBY

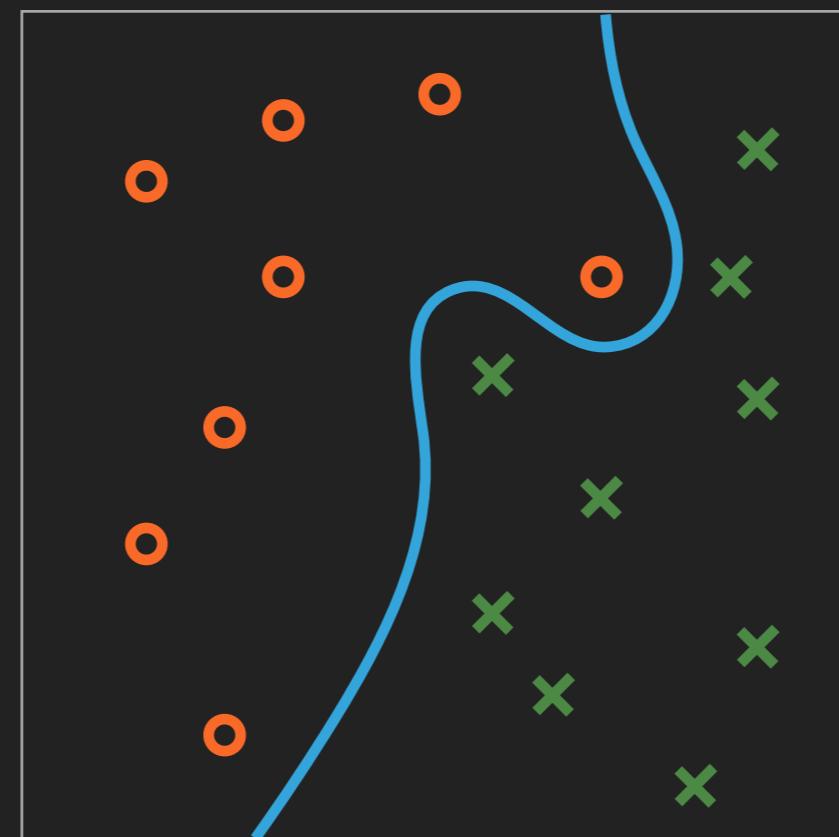
LUA

C#

SUPPORTS MULTIPLE HIGH-LEVEL PROGRAMMING LANGUAGES WITH UNIFIED INTERFACE

C++

← Shogun ↓



PYTHON

R

JAVA

OCTAVE

RUBY

LUA

C#

EXAMPLE: LINEAR SVM WITH C++

```
1 #include <shogun/base/init.h>
2 #include <shogun/base/some.h>
3 #include <shogun/classifier/svm/LibLinear.h>
4 #include <shogun/classifier/svm/LibLinear.h>
5 #include <shogun/evaluation/ContingencyTableEvaluation.h>
6 #include <shogun/features/DenseFeatures.h>
7 #include <shogun/io/CSVFile.h>
8 #include <shogun/io/SerializableAsciiFile.h>
9 #include <shogun/labels/BinaryLabels.h>
10 #include <shogun/lib/SGVector.h>
11 #include <shogun/lib/WrappedObjectArray.h>
12
13 using namespace shogun;
14
15 int main(int, char[])
16 {
17     init_shogun_with_defaults();
18
19     auto f_feats_train = some<CCSVFile>("../data/classifier_binary_2d_linear_features_train.dat");
20     auto f_feats_test = some<CCSVFile>("../data/classifier_binary_2d_linear_features_test.dat");
21     auto f_labels_train = some<CCSVFile>("../data/classifier_binary_2d_linear_labels_train.dat");
22     auto f_labels_test = some<CCSVFile>("../data/classifier_binary_2d_linear_labels_test.dat");
23
24     //![create_features]
25     auto features_train = some<CDenseFeatures<float64_t>>(f_feats_train);
26     auto features_test = some<CDenseFeatures<float64_t>>(f_feats_test);
27     auto labels_train = some<CBinaryLabels>(f_labels_train);
28     auto labels_test = some<CBinaryLabels>(f_labels_test);
29     //![create_features]
30
31     //![set_parameters]
32     auto C = 1.0;
33     auto epsilon = 0.001;
34     //![set_parameters]
35
36     //![create_instance]
37     auto svm = some<CLibLinear>(C, features_train, labels_train);
38     svm->set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE::L2R_L2LOSS_SVC);
39     svm->set_epsilon(epsilon);
40     //![create_instance]
41
42     //![train_and_apply]
43     svm->train();
44     auto labels_predict = svm->apply_binary(features_test);
45     //![train_and_apply]
46
47     //![extract_weights_bias]
48     auto w = svm->get_w();
49     auto b = svm->get_bias();
50     //![extract_weights_bias]
51
52     //![evaluate_accuracy]
53     auto eval = some<CAccuracyMeasure>();
54     auto accuracy = eval->evaluate(labels_predict, labels_test);
55     //![evaluate_accuracy]
56
57     // additional integration testing variables
58     auto output = labels_predict->get_labels();
59
60     exit_shogun();
61     return 0;
62 }
```

EXAMPLE: LINEAR SVM WITH PYTHON

```
1 import numpy as np
2 from shogun import AccuracyMeasure
3 from shogun import BinaryLabels
4 from shogun import CSVFile
5 from shogun import L2R_L2LOSS_SVC
6 from shogun import LibLinear
7 from shogun import RealFeatures
8 from shogun import SerializableAsciiFile
9 from shogun import WrappedObjectArray
10
11 f_feats_train = CSVFile("../data/classifier_binary_2d_linear_features_train.dat")
12 f_feats_test = CSVFile("../data/classifier_binary_2d_linear_features_test.dat")
13 f_labels_train = CSVFile("../data/classifier_binary_2d_linear_labels_train.dat")
14 f_labels_test = CSVFile("../data/classifier_binary_2d_linear_labels_test.dat")
15
16 #![create_features]
17 features_train = RealFeatures(f_feats_train)
18 features_test = RealFeatures(f_feats_test)
19 labels_train = BinaryLabels(f_labels_train)
20 labels_test = BinaryLabels(f_labels_test)
21 #![create_features]
22
23 #![set_parameters]
24 C = 1.0
25 epsilon = 0.001
26 #![set_parameters]
27
28 #![create_instance]
29 svm = LibLinear(C, features_train, labels_train)
30 svm.set_liblinear_solver_type(L2R_L2LOSS_SVC)
31 svm.set_epsilon(epsilon)
32 #![create_instance]
33
34 #![train_and_apply]
35 svm.train()
36 labels_predict = svm.apply_binary(features_test)
37 #![train_and_apply]
38
39 #![extract_weights_bias]
40 w = svm.get_w()
41 b = svm.get_bias()
42 #![extract_weights_bias]
43
44 #![evaluate_accuracy]
45 eval = AccuracyMeasure()
46 accuracy = eval.evaluate(labels_predict, labels_test)
47 #![evaluate_accuracy]
48
49 # additional integration testing variables
50 output = labels_predict.get_labels()
51
52 # Serialize output for integration testing (automatically generated)
53 __sg_storage = WrappedObjectArray()
54 __sg_storage_file = SerializableAsciiFile("linear_svm.dat", 'w')
55 __sg_storage.append_wrapped_real(C, "C")
56 __sg_storage.append_wrapped_real(epsilon, "epsilon")
57 __sg_storage.append_wrapped_real_vector(w, "w")
58 __sg_storage.append_wrapped_real(b, "b")
59 __sg_storage.append_wrapped_real(accuracy, "accuracy")
60 __sg_storage.append_wrapped_real_vector(output, "output")
61 __sg_storage.save_serializable(__sg_storage_file)
```

EXAMPLE: LINEAR SVM WITH C#

```
1  using System;
2
3  public class Application {
4  public static void Main() {
5    shogun.init_shogun_with_defaults();
6
7    CSVFile f_feats_train = new CSVFile("../data/classifier_binary_2d_linear_features_train.dat");
8    CSVFile f_feats_test = new CSVFile("../data/classifier_binary_2d_linear_features_test.dat");
9    CSVFile f_labels_train = new CSVFile("../data/classifier_binary_2d_linear_labels_train.dat");
10   CSVFile f_labels_test = new CSVFile("../data/classifier_binary_2d_linear_labels_test.dat");
11
12  //![create_features]
13  RealFeatures features_train = new RealFeatures(f_feats_train);
14  RealFeatures features_test = new RealFeatures(f_feats_test);
15  BinaryLabels labels_train = new BinaryLabels(f_labels_train);
16  BinaryLabels labels_test = new BinaryLabels(f_labels_test);
17  //![create_features]
18
19  //![set_parameters]
20  double C = 1.0;
21  double epsilon = 0.001;
22  //![set_parameters]
23
24  //![create_instance]
25  LibLinear svm = new LibLinear(C, features_train, labels_train);
26  svm.set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE.L2R_L2LOSS_SVC);
27  svm.set_epsilon(epsilon);
28  //![create_instance]
29
30  //![train_and_apply]
31  svm.train();
32  BinaryLabels labels_predict = svm.apply_binary(features_test);
33  //![train_and_apply]
34
35  //![extract_weights_bias]
36  double[] w = svm.get_w();
37  double b = svm.get_bias();
38  //![extract_weights_bias]
39
40  //![evaluate_accuracy]
41  AccuracyMeasure eval = new AccuracyMeasure();
42  double accuracy = eval.evaluate(labels_predict, labels_test);
43  //![evaluate_accuracy]
44
45  // additional integration testing variables
46  double[] output = labels_predict.get_labels();
47
48  // Serialize output for integration testing (automatically generated)
49  WrappedObjectArray __sg_storage = new WrappedObjectArray();
50  SerializableAsciiFile __sg_storage_file = new SerializableAsciiFile("linear_svm.dat", 'w');
51  __sg_storage.append_wrapped_real(C, "C");
52  __sg_storage.append_wrapped_real(epsilon, "epsilon");
53  __sg_storage.append_wrapped_real_vector(w, "w");
54  __sg_storage.append_wrapped_real(b, "b");
55  __sg_storage.append_wrapped_real(accuracy, "accuracy");
56  __sg_storage.append_wrapped_real_vector(output, "output");
57  __sg_storage.save_serializable(__sg_storage_file);
58
59 }
60 }
```

EXAMPLE: LINEAR SVM WITH JAVA

```
1 import org.jblas.DoubleMatrix;
2 import org.jblas.FloatMatrix;
3
4 import org.shogun.shogun;
5 import org.shogun.AccuracyMeasure;
6 import org.shogun.BinaryLabels;
7 import org.shogun.CSVFile;
8 import org.shogun.LibLinear;
9 import org.shogun.RealFeatures;
10 import org.shogun.RealVector;
11 import org.shogun.SerializableAsciiFile;
12 import org.shogun.WrappedObjectArray;
13
14
15 public class linear_svm {
16     static {
17         System.loadLibrary("shogun");
18     }
19
20     public static void main(String argv[]) {
21         shogun.init_shogun_with_defaults();
22
23         CSVFile f_feats_train = new CSVFile("../data/classifier_binary_2d_linear_features_train.dat");
24         CSVFile f_feats_test = new CSVFile("../data/classifier_binary_2d_linear_features_test.dat");
25         CSVFile f_labels_train = new CSVFile("../data/classifier_binary_2d_linear_labels_train.dat");
26         CSVFile f_labels_test = new CSVFile("../data/classifier_binary_2d_linear_labels_test.dat");
27
28         //![create_features]
29         RealFeatures features_train = new RealFeatures(f_feats_train);
30         RealFeatures features_test = new RealFeatures(f_feats_test);
31         BinaryLabels labels_train = new BinaryLabels(f_labels_train);
32         BinaryLabels labels_test = new BinaryLabels(f_labels_test);
33         //![create_features]
34
35         //![set_parameters]
36         double C = 1.0;
37         double epsilon = 0.001;
38         //![set_parameters]
39
40         //![create_instance]
41         LibLinear svm = new LibLinear(C, features_train, labels_train);
42         svm.set_liblinear_solver_type(LIBLINEAR_SOLVER_TYPE.L2R_L2LOSS_SVC);
43         svm.set_epsilon(epsilon);
44         //![create_instance]
45
46         //![train_and_apply]
47         svm.train();
48         BinaryLabels labels_predict = svm.apply_binary(features_test);
49         //![train_and_apply]
50
51         //![extract_weights_bias]
52         DoubleMatrix w = svm.get_w();
53         double b = svm.get_bias();
54         //![extract_weights_bias]
55
56         //![evaluate_accuracy]
57         AccuracyMeasure eval = new AccuracyMeasure();
58         double accuracy = eval.evaluate(labels_predict, labels_test);
59         //![evaluate_accuracy]
60
61         // additional integration testing variables
62         DoubleMatrix output = labels_predict.get_labels();
63
64         // Serialize output for integration testing (automatically generated)
65         WrappedObjectArray __sg_storage = new WrappedObjectArray();
66         SerializableAsciiFile __sg_storage_file = new SerializableAsciiFile("linear_svm.dat", 'w');
67         __sg_storage.append_wrapped_real(C, "C");
68         __sg_storage.append_wrapped_real(epsilon, "epsilon");
69         __sg_storage.append_wrapped_real_vector(w, "w");
70         __sg_storage.append_wrapped_real(b, "b");
71         __sg_storage.append_wrapped_real(accuracy, "accuracy");
72         __sg_storage.append_wrapped_real_vector(output, "output");
73         __sg_storage.save_serializable(__sg_storage_file);
74
75     }
76 }
```

將軍 Shogun 6.0.0

Quickstart
Running Shogun from the interfaces

Binary classifier
Averaged Perceptron
Kernel Support Vector Machine
Linear Discriminant Analysis
Linear Support Vector Machine

Multiclass classifier
Classification And Regression Tree
CHAID tree
Gaussian Naive Bayes
K Nearest neighbours
Large Margin Nearest Neighbours
Linear Discriminant Analysis
Multi-class Error-Correcting Output Codes
Multi-class Linear Machine
Multi-class Logistic Regression
Quadratic Discriminant Analysis
Random Forest
Relaxed Tree
ShareBoost
Multi-class Support Vector Machine

Regression
Kernel Ridge Regression
Nyström Kernel Ridge Regression

Python Octave Java/scala Ruby R Lua C# Native C++

Linear Support Vector Machine

Linear Support Vector Machine is a binary classifier which finds a hyper-plane such that the margins between the two classes are maximized. The loss function that needs to be minimized is:

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi(\mathbf{w}; \mathbf{x}_i, y_i)$$

where \mathbf{w} is vector of weights, \mathbf{x}_i is feature vector, y_i is the corresponding label, $C > 0$ is a penalty parameter, N is the number of training samples and ξ is hinge loss function.

The solution takes the following form:

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

α_i are sparse in the above solution.

See [FCH+08] and Chapter 6 in [CST00] for a detailed introduction.

Example

Imagine we have files with training and test data. We create CDenseFeatures (here 64 bit floats aka RealFeatures) and CBinaryLabels as

```
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);
```

In order to run CLibLinear, we need to initialize some parameters like C and epsilon which is the residual convergence parameter of the solver.

```
auto C = 1.0;
auto epsilon = 0.001;
```

FUTURE PLANS

- ▶ Interface Tensorflow, Stan and etc.
- ▶ Interface Matlab and Javascript with SWIG
- ▶ Migrate to C++ 17
- ▶ ...
- ▶ Get more developers involved!

THANKS FOR YOUR ATTENTION

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

