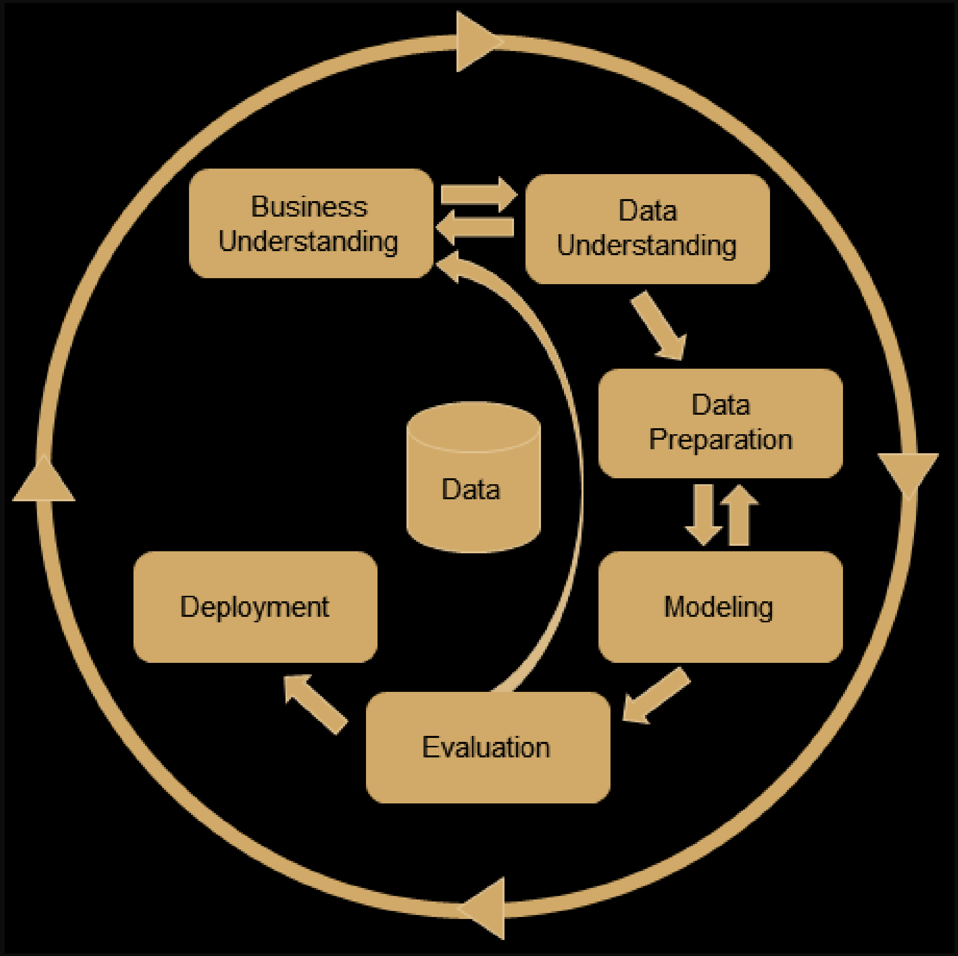
mlpack: or, How I Learned To Stop Worrying and Love C++

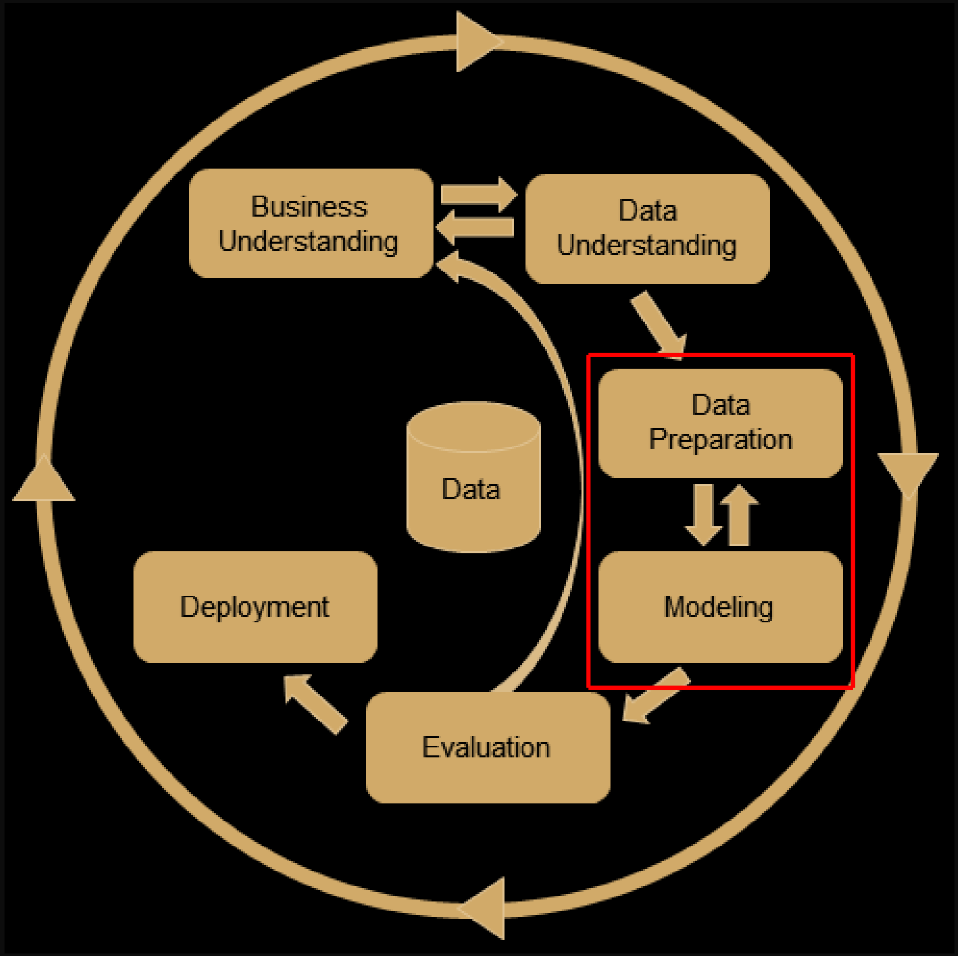
Ryan R. Curtin
ryan.curtin@relational.ai

May 30, 2019

Introduction: the data science cycle



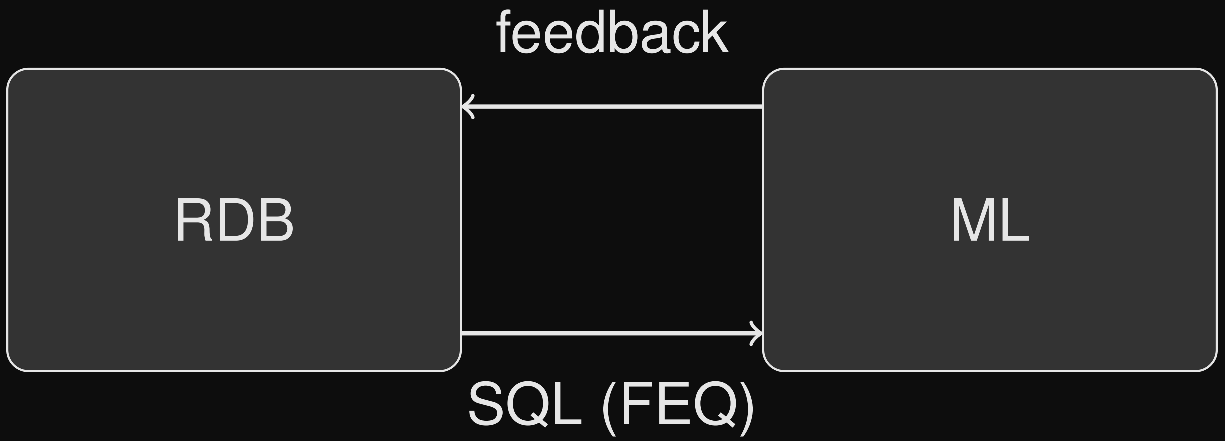
Introduction: the data science cycle



How long does this take your organization?

Feature Extraction Queries

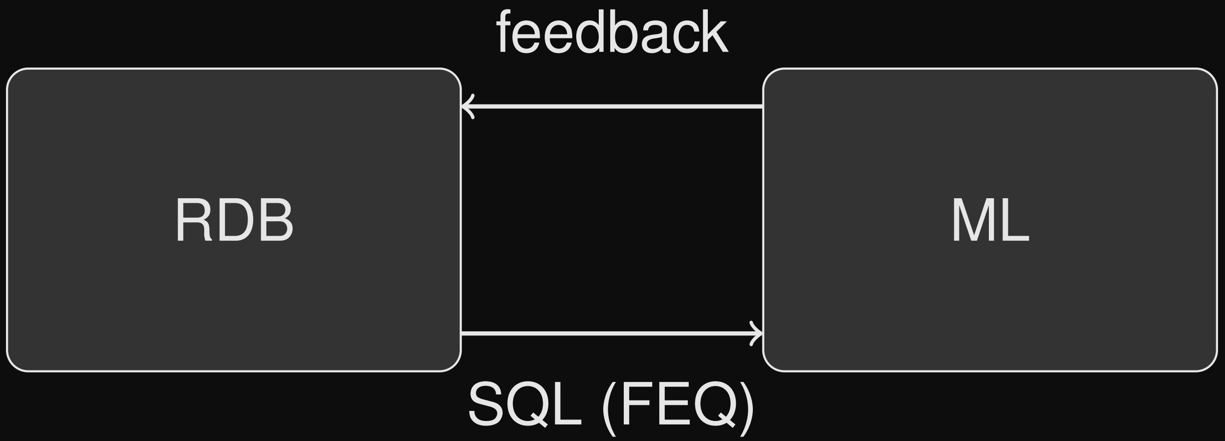
Typically the data scientist extracts data with a feature extraction query (FEQ) and then builds an ML model, then iterates.



How long can this take?

Feature Extraction Queries

Typically the data scientist extracts data with a feature extraction query (FEQ) and then builds an ML model, then iterates.



How long can this take? Case study: at Symantec, to train neural networks to detect malicious domains, the FEQ took 8–16 hours and the ML training took 24 hours.

Feature Extraction Queries

Typically the da ction query

(FEQ) and ther

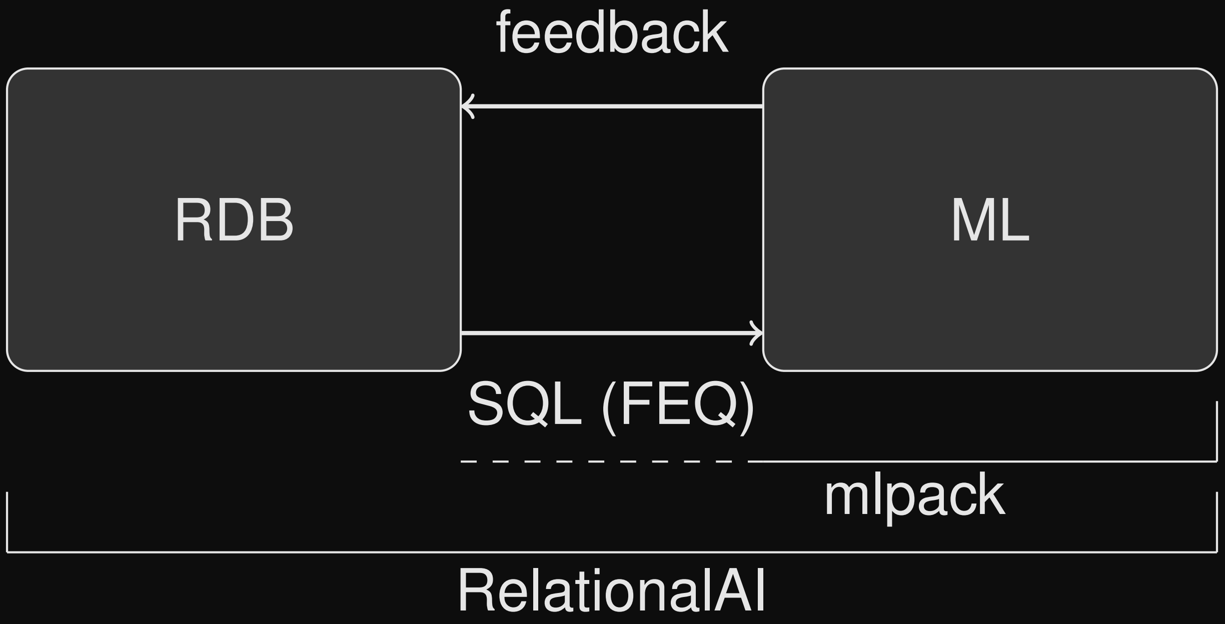
How long can n neural

networks to det ours and the

ML training too

Feature Extraction Queries

Typically the data scientist extracts data with a feature extraction query (FEQ) and then builds an ML model, then iterates.

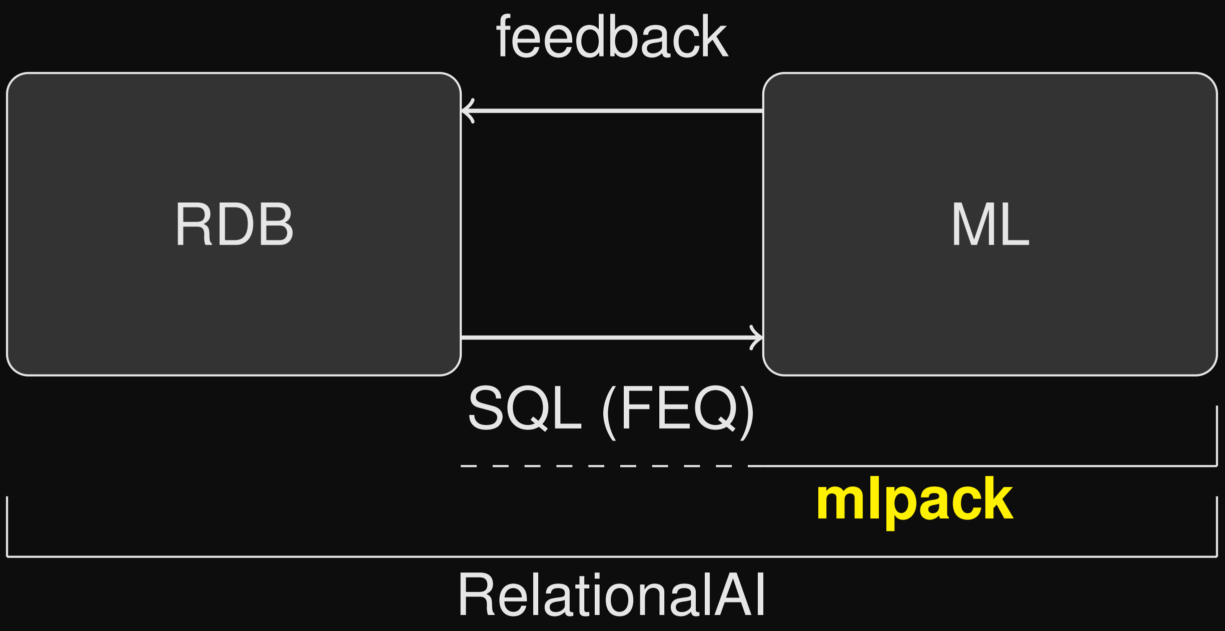


We can do better: we can combine both of these operations and get massive speedups in some cases!

M.A. Khamis, H.Q. Ngo, A. Rudra. FAQ: Questions Asked Frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 13–28, 2016.

Feature Extraction Queries

Typically the data scientist extracts data with a feature extraction query (FEQ) and then builds an ML model, then iterates.

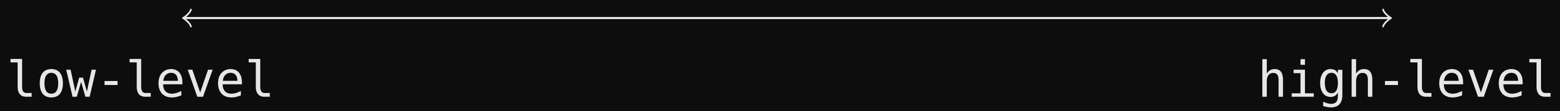


We can do better: we can combine both of these operations and get massive speedups in some cases!

M.A. Khamis, H.Q. Ngo, A. Rudra. FAQ: Questions Asked Frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 13–28, 2016.

Graph #1

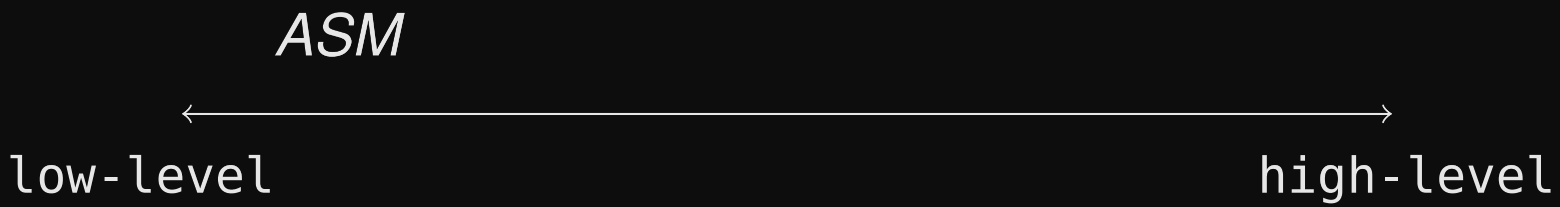
Graph #1



low-level

high-level

Graph #1

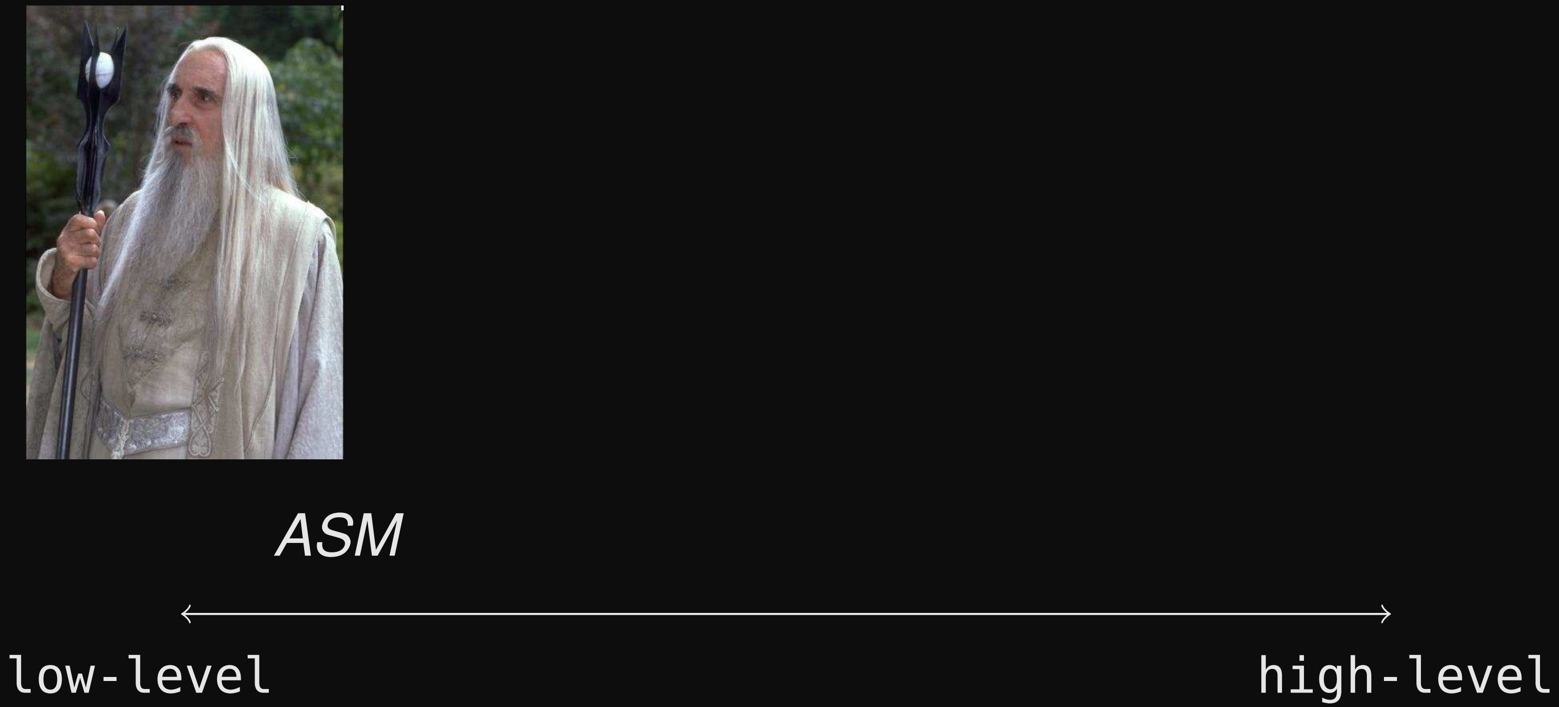


ASM

low-level

high-level

Graph #1

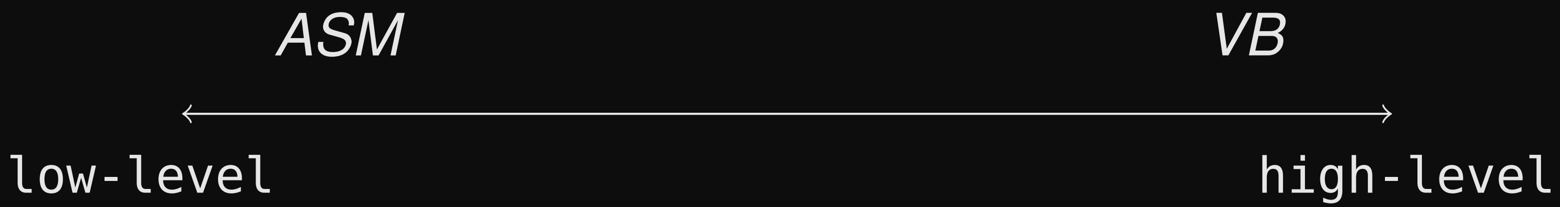


ASM

low-level

high-level

Graph #1



ASM

VB

low-level

high-level

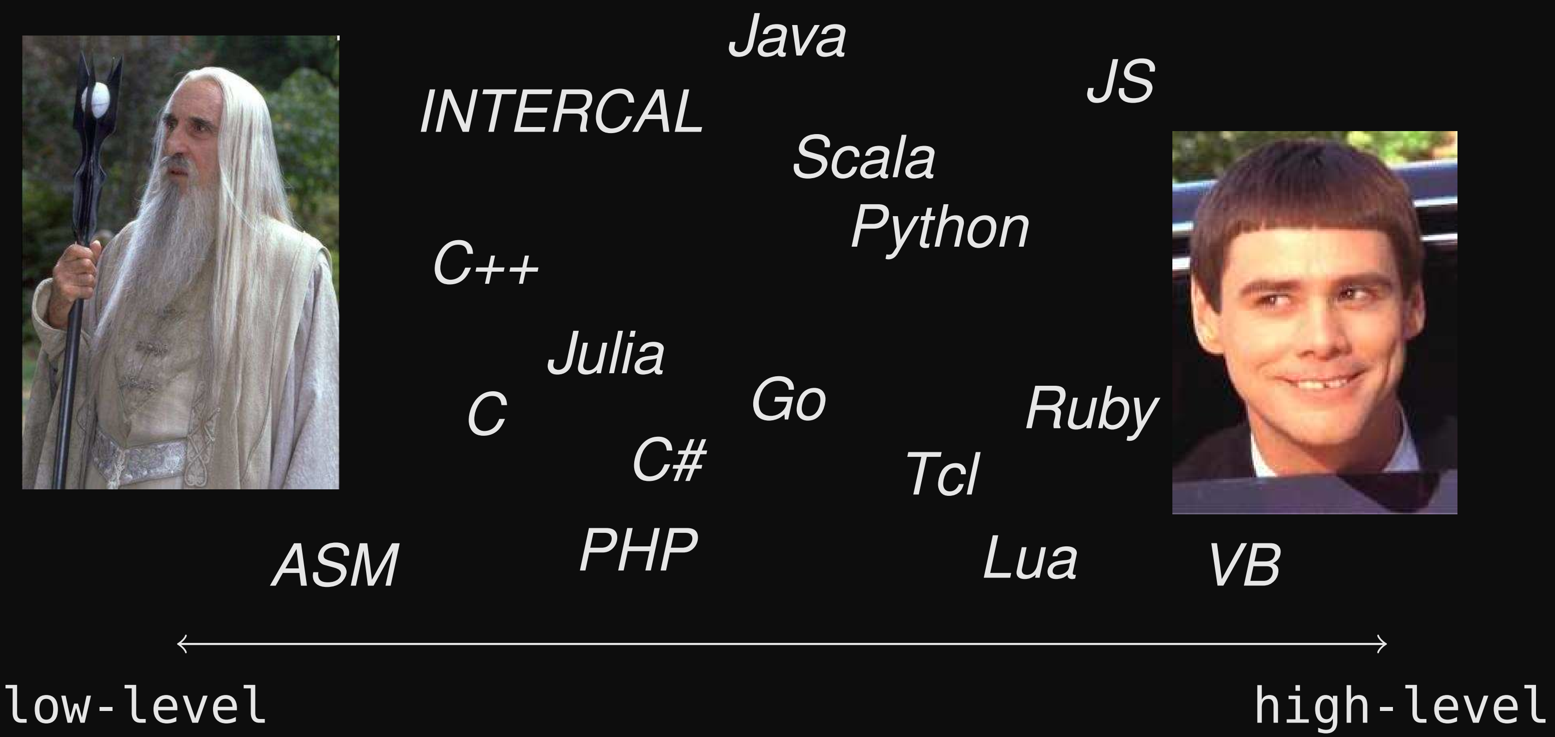
Graph #1

ASM

VB

low-level ↔ high-level

Graph #1



Java

JS

INTERCAL

Scala

Python

C++

Julia

Go

C

Ruby

C#

Tcl

PHP

Lua

ASM

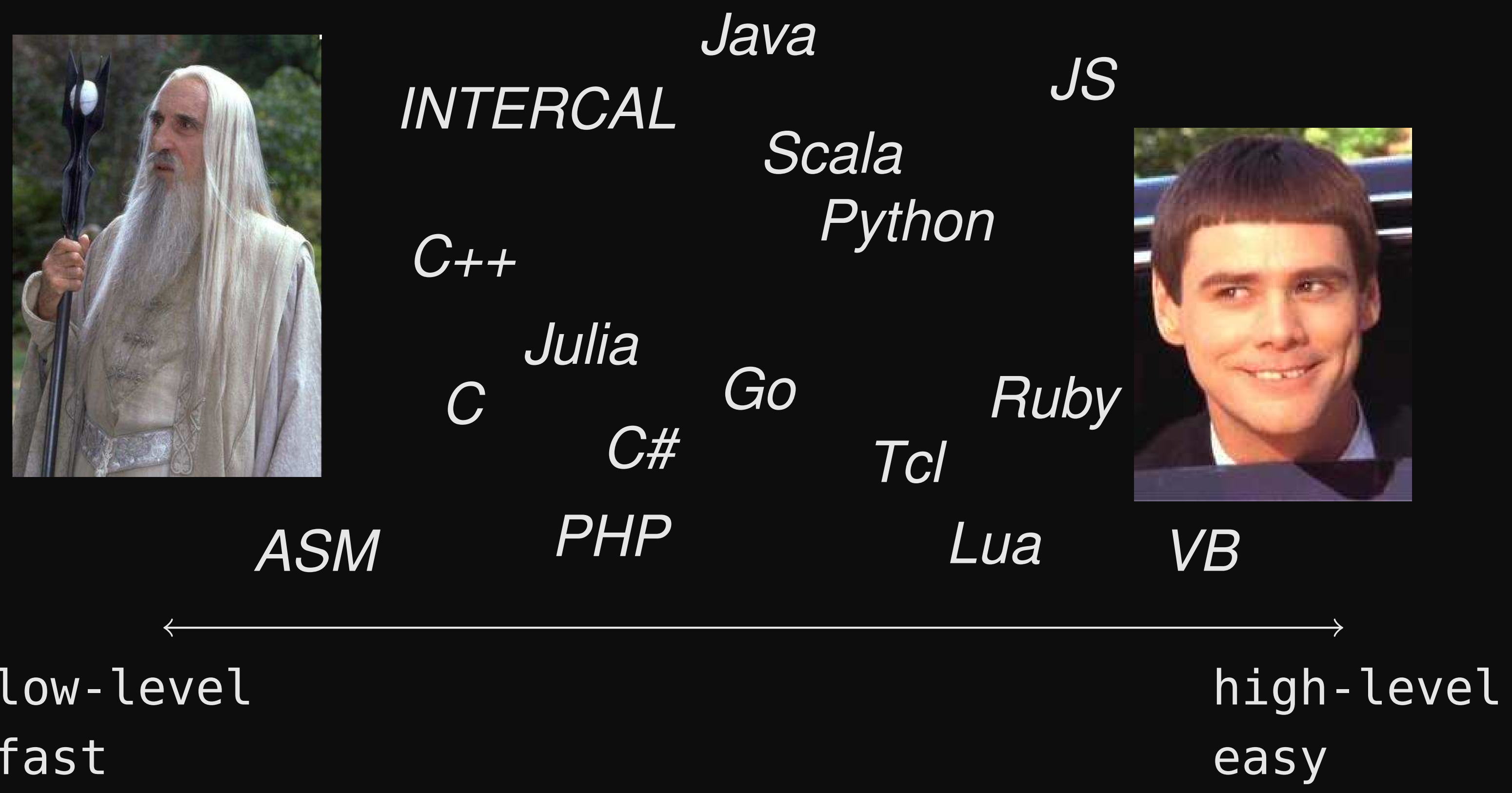
VB

low-level

high-level

Note: this is not a scientific or particularly accurate representation.

Graph #1



Java

JS

INTERCAL

Scala

Python

C++

Julia

Go

Ruby

C

C#

Tcl

PHP

Lua

ASM

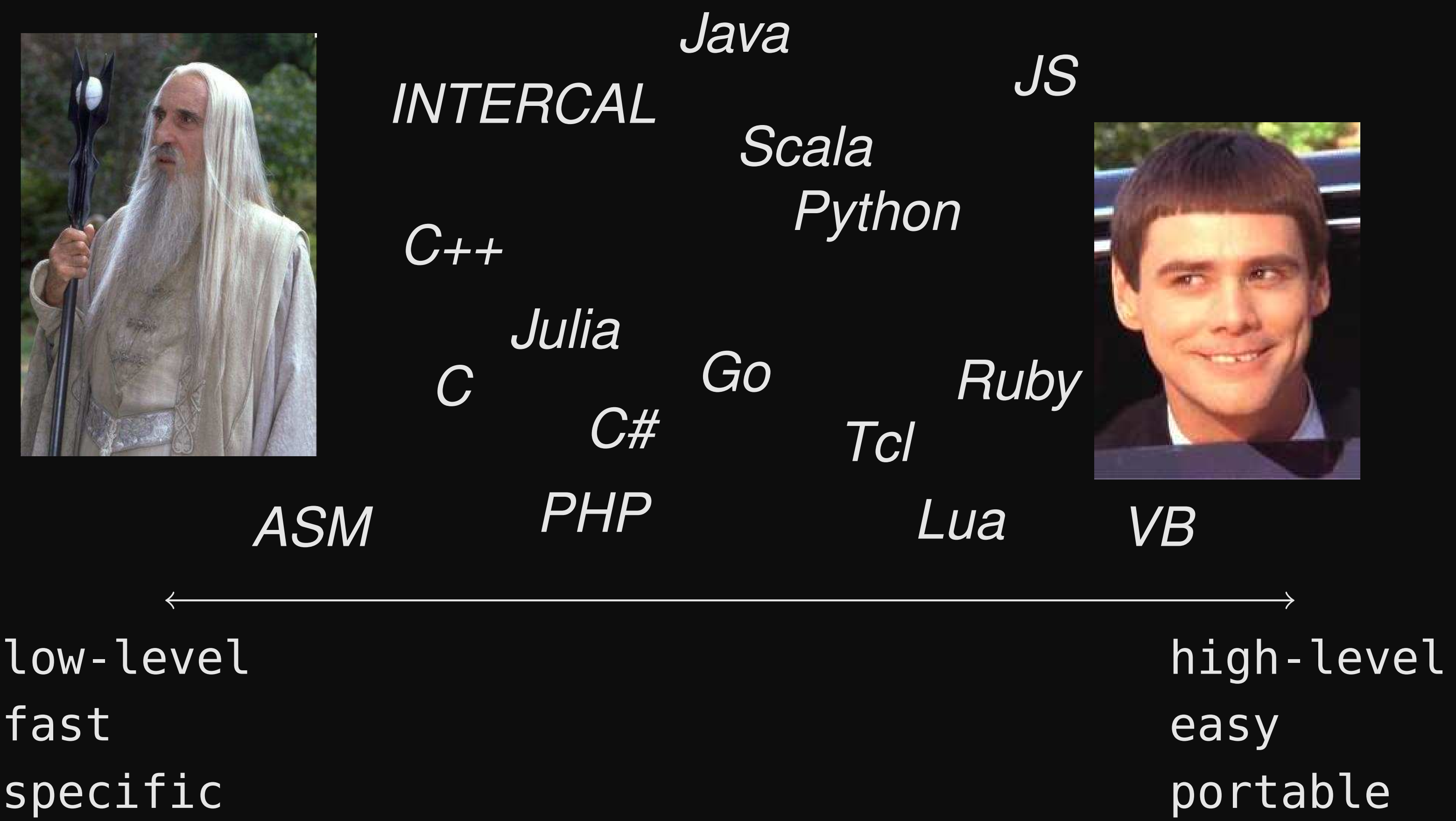
VB

low-level
fast

high-level
easy

Note: this is not a scientific or particularly accurate representation.

Graph #1



Java

JS

INTERCAL

Scala

Python

C++

Julia

Go

Ruby

C

C#

Tcl

PHP

Lua

ASM

VB

low-level
fast
specific

high-level
easy
portable

Note: this is not a scientific or particularly accurate representation.

The Big Tradeoff

speed vs. **portability and readability**

The Big Tradeoff

speed vs. **portability and readability**

The Big Tradeoff

speed vs. **portability and readability**

If we’re careful, we can get speed, portability, *and* readability by using C++.

So, mlpack.

What is it?

So, mlpack.

What is it?

- a fast general-purpose C++ machine learning library
- contains flexible implementations of common and cutting-edge machine learning algorithms
- for fast or big runs on single workstations
- bindings are available for R, Python, and the command line, and are coming for other languages (Go, Julia, etc.)

- 140+ developers from around the world
- regular participation in the Google Summer of Code program

So, mlpack.

What is it?

- a fast general-purpose C++ machine learning library
- contains flexible implementations of common and cutting-edge machine learning algorithms
- for fast or big runs on single workstations
- bindings are available for R, Python, and the command line, and are coming for other languages (Go, Julia, etc.)

- 140+ developers from around the world
- regular participation in the Google Summer of Code program

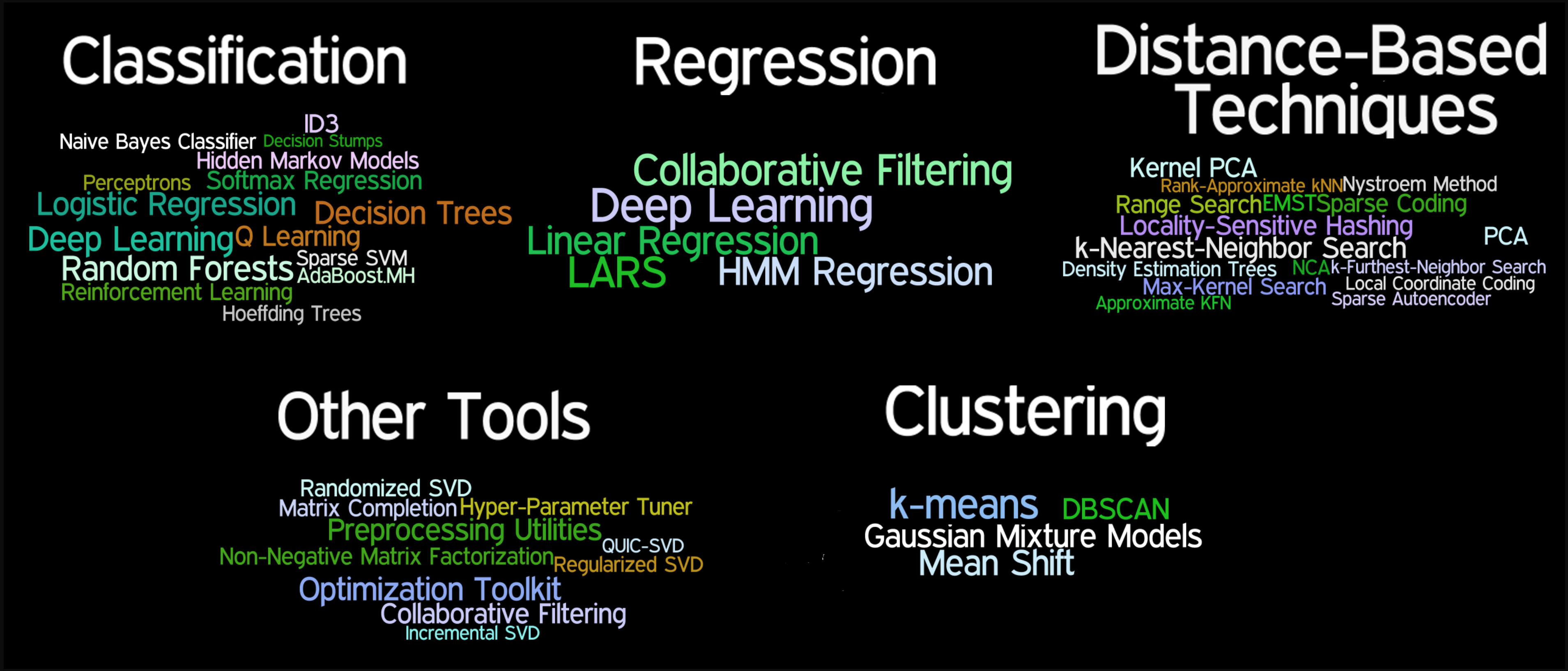
`http://www.mlpack.org/`

`https://github.com/mlpack/mlpack/`

R.R. Curtin, J.R. Cline, N.P. Slagle, W.B. March, P. Ram, N.A. Mehta, A.G. Gray, "**mlpack**: a scalable C++ machine learning library", in *The Journal of Machine Learning Research*, vol. 14, p. 801–805, 2013.

What does mlpack implement?

mlpack implements a lot of standard machine learning techniques and also new, cutting-edge techniques.



How do we get mlpack?

Linux (Debian/Ubuntu):	`$ sudo apt-get install libmlpack-dev`
Linux (Red Hat/Fedora):	`$ sudo dnf install mlpack-devel`

OS X (Homebrew):	`$ brew tap brewsci/science &&`
	`brew install mlpack`

Windows (nuget):	`> nuget add mlpack-windows`

Or install from source:

```
$ git clone https://github.com/mlpack/mlpack
$ mkdir mlpack/build && cd mlpack/build
$ cmake ../
$ make -j8 # Probably good to use many cores.
$ sudo make install
```

https://www.mlpack.org/docs/mlpack-3.0.4/doxygen/build.html
https://keon.io/mlpack/mlpack-on-windows/

Installing from Python

Use pip:

```
$ pip install mlpack3
```

Or use conda:

```
$ conda install -c mlpack mlpack
```

Command-line programs

You don’t need to be a C++ expert.

```
# Train AdaBoost model.
$ mlpack_adaboost -t training_file.h5 -l training_labels.h5 \
> -M trained_model.bin
# Predict with AdaBoost model.
$ mlpack_adaboost -m trained_model.bin -T test_set.csv \
> -o test_set_predictions.csv
```

Command-line programs

You don’t need to be a C++ expert.

```
# Train AdaBoost model.
$ mlpack_adaboost -t training_file.h5 -l training_labels.h5 \
> -M trained_model.bin
# Predict with AdaBoost model.
$ mlpack_adaboost -m trained_model.bin -T test_set.csv \
> -o test_set_predictions.csv
```

```
# Find the 5 nearest neighbors of the data in dataset.txt, storing the
# indices of the neighbors in ’neighbors.csv’.
$ mlpack_knn -r dataset.txt -k 5 -n neighbors.csv
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
[INFO ] Performing PCA on dataset...
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
[INFO ] Performing PCA on dataset...
[INFO ] 99.9491% of variance retained (5 dimensions).
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
[INFO ] Performing PCA on dataset...
[INFO ] 99.9491% of variance retained (5 dimensions).
>>>
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
[INFO ] Performing PCA on dataset...
[INFO ] 99.9491% of variance retained (5 dimensions).
>>> result[’output’].shape
```

Python bindings

Can be dropped directly into a Python workflow.

```
>>> import numpy as np
>>> from mlpack import pca
>>> x = np.genfromtxt(’my_data.csv’, delimiter=’,’)
>>> x.shape
(2048, 10)
>>> result = pca(input=x, new_dimensionality=5, verbose=True)
[INFO ] Performing PCA on dataset...
[INFO ] 99.9491% of variance retained (5 dimensions).
>>> result[’output’].shape
(2048, 5)
>>>
```

Python bindings

Documentation is straightforward and extensive.

```
>>>
```

Python bindings

Documentation is straightforward and extensive.

```
>>> from mlpack import cf
```

Python bindings

Documentation is straightforward and extensive.

```
>>> from mlpack import cf
>>>
```

Python bindings

Documentation is straightforward and extensive.

```
>>> from mlpack import cf
>>> help(cf)
```

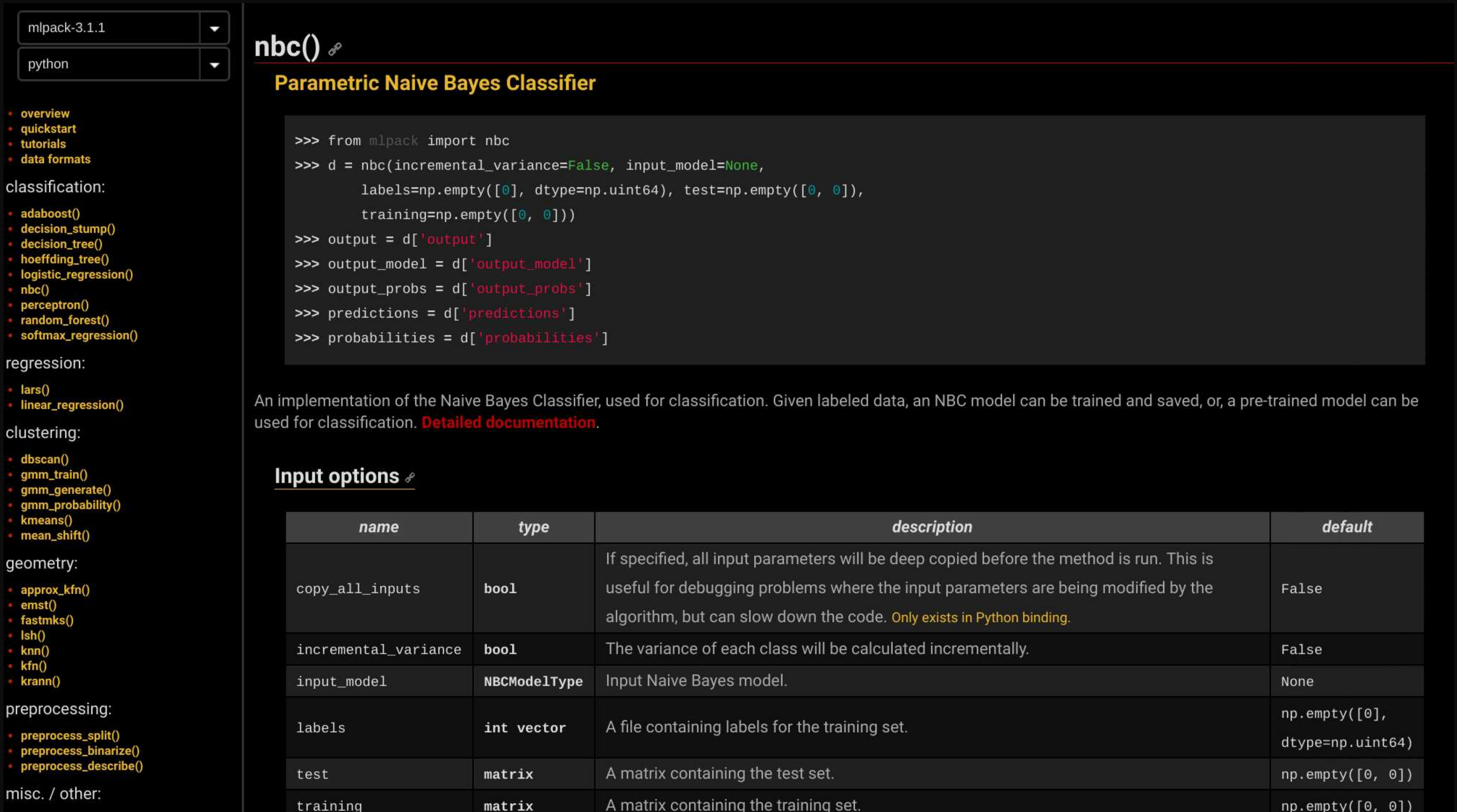
```
Help on built-in function cf in module mlpack.cf:

cf(...)
      Collaborative Filtering

      This program performs collaborative filtering (CF) on the given dataset. Given
      a list of user, item and preferences (the ’training’ parameter), the program
      will perform a matrix decomposition and then can perform a series of actions
      related to collaborative filtering.  Alternately, the program can load an
      existing saved CF model with the ’input_model’ parameter and then use that
      model to provide recommendations or predict values.

      The input matrix should be a 3-dimensional matrix of ratings, where the first
      dimension is the user, the second dimension is the item, and the third
      dimension is that user’s rating of that item.  Both the users and items should
      be numeric indices, not names. The indices are assumed to start from 0.

      A set of query users for which recommendations can be generated may be
      specified with the ’query’ parameter; alternately, recommendations may be
      generated for every user in the dataset by specifying the
      ’all_user_recommendations’ parameter.  In addition, the number of
      recommendations per user to generate can be specified with the
      ’recommendations’ parameter, and the number of similar users (the size of the
      neighborhood)  to be considered when generating recommendations can be
      specified with the ’neighborhood’ parameter.

      For performing the matrix decomposition, the following optimization algorithms
      can be specified via the ’algorithm’ parameter:
      ’RegSVD’ -- Regularized SVD using a SGD optimizer
```

```
'NMF' -- Non-negative matrix factorization with alternating least squares
update rules
'BatchSVD' -- SVD batch learning
'SVDIncompleteIncremental' -- SVD incomplete incremental learning
'SVDCompleteIncremental' -- SVD complete incremental learning
A trained model may be saved to with the 'output_model' output parameter.

To train a CF model on a dataset 'training_set' using NMF for decomposition
and saving the trained model to 'model', one could call:

>>> cf(training=training_set, algorithm='NMF')
>>> model = output['output_model']

Then, to use this model to generate recommendations for the list of users in
the query set 'users', storing 5 recommendations in 'recommendations', one
could call

>>> cf(input_model=model, query=users, recommendations=5)
>>> recommendations = output['output']


Input parameters:

 - algorithm (string): Algorithm used for matrix factorization.  Default
      value 'NMF'.
 - all_user_recommendations (bool): Generate recommendations for all
      users.
 - copy_all_inputs (bool): If specified, all input parameters will be
      deep copied before the method is run.  This is useful for debugging
      problems where the input parameters are being modified by the algorithm,
      but can slow down the code.
 - input_model (CFType): Trained CF model to load.
```

Documentation

The documentation is also readily available online.

`https://www.mlpack.org/docs.html`



- overview
- quickstart
- tutorials
- data formats

classification:

- adaboost()
- decision_stump()
- decision_tree()
- hoeffding_tree()
- logistic_regression()
- nbc()
- perceptron()
- random_forest()
- softmax_regression()

regression:

- lars()
- linear_regression()

clustering:

- dbscan()
- gmm_train()
- gmm_generate()
- gmm_probability()
- kmeans()
- mean_shift()

geometry:

- approx_kfn()
- emst()
- fastmks()
- lsh()
- knn()
- kfn()
- krann()

preprocessing:

- preprocess_split()
- preprocess_binarize()
- preprocess_describe()

misc. / other:

nbc()

Parametric Naive Bayes Classifier

```
>>> from mlpack import nbc
>>> d = nbc(incremental_variance=False, input_model=None,
        labels=np.empty([0], dtype=np.uint64), test=np.empty([0, 0]),
        training=np.empty([0, 0]))
>>> output = d['output']
>>> output_model = d['output_model']
>>> output_probs = d['output_probs']
>>> predictions = d['predictions']
>>> probabilities = d['probabilities']
```

An implementation of the Naive Bayes Classifier, used for classification. Given labeled data, an NBC model can be trained and saved, or, a pre-trained model can be used for classification. Detailed documentation.

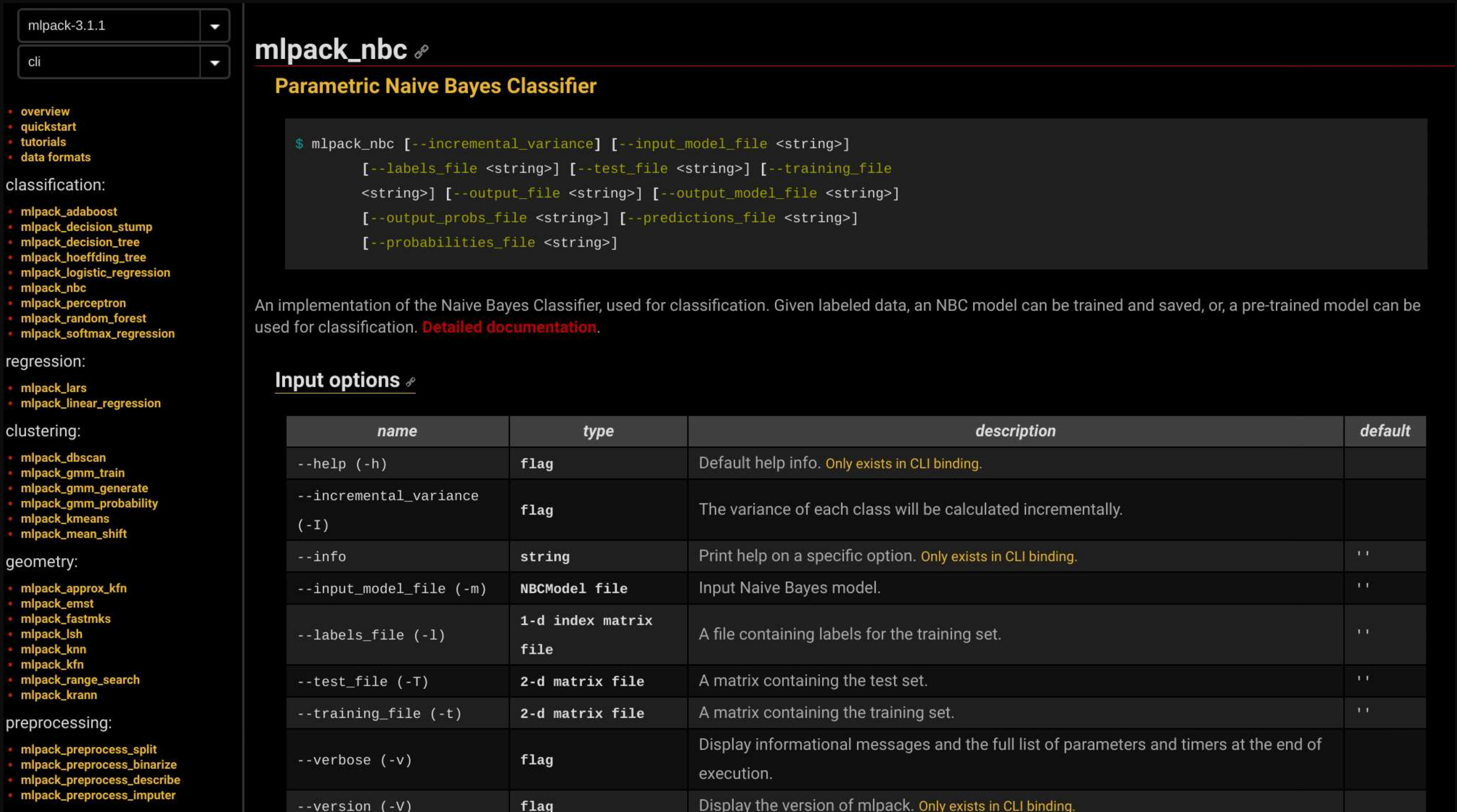
Input options

name	*type*	*description*	*default*
`copy_all_inputs`	**`bool`**	If specified, all input parameters will be deep copied before the method is run. This is useful for debugging problems where the input parameters are being modified by the algorithm, but can slow down the code. Only exists in Python binding.	`False`
`incremental_variance`	**`bool`**	The variance of each class will be calculated incrementally.	`False`
`input_model`	**`NBCModelType`**	Input Naive Bayes model.	`None`
`labels`	**`int vector`**	A file containing labels for the training set.	`np.empty([0], dtype=np.uint64)`
`test`	**`matrix`**	A matrix containing the test set.	`np.empty([0, 0])`
`training`	**`matrix`**	A matrix containing the training set.	`np.empty([0, 0])`

Documentation

The documentation is also readily available online.

`https://www.mlpack.org/docs.html`



mlpack-3.1.1

cli

- overview
- quickstart
- tutorials
- data formats

classification:

- mlpack_adaboost
- mlpack_decision_stump
- mlpack_decision_tree
- mlpack_hoeffding_tree
- mlpack_logistic_regression
- mlpack_nbc
- mlpack_perceptron
- mlpack_random_forest
- mlpack_softmax_regression

regression:

- mlpack_lars
- mlpack_linear_regression

clustering:

- mlpack_dbscan
- mlpack_gmm_train
- mlpack_gmm_generate
- mlpack_gmm_probability
- mlpack_kmeans
- mlpack_mean_shift

geometry:

- mlpack_approx_kfn
- mlpack_emst
- mlpack_fastmks
- mlpack_lsh
- mlpack_knn
- mlpack_kfn
- mlpack_range_search
- mlpack_krann

preprocessing:

- mlpack_preprocess_split
- mlpack_preprocess_binarize
- mlpack_preprocess_describe
- mlpack_preprocess_imputer

mlpack_nbc

Parametric Naive Bayes Classifier

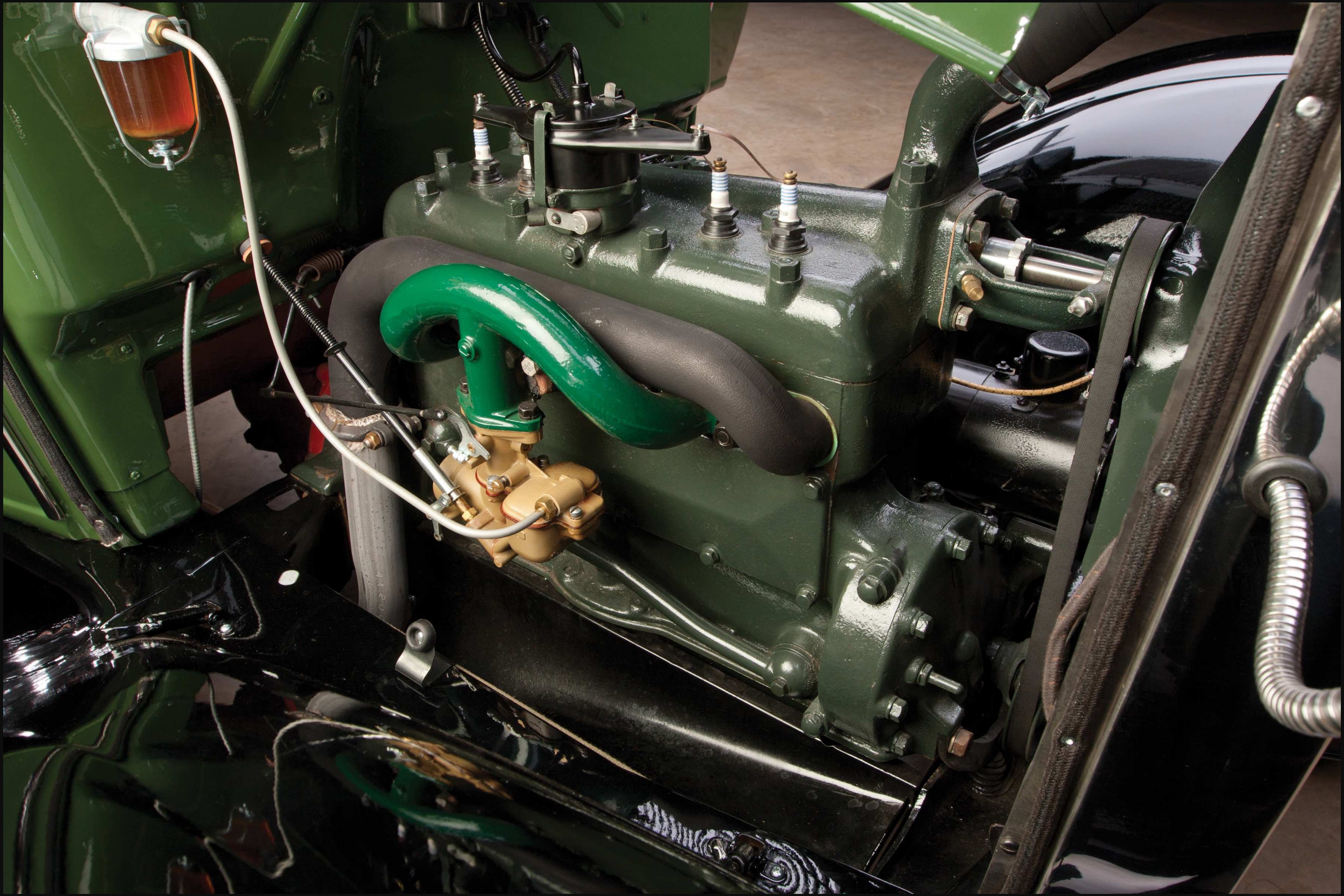
```
$ mlpack_nbc [--incremental_variance] [--input_model_file <string>]
        [--labels_file <string>] [--test_file <string>] [--training_file
        <string>] [--output_file <string>] [--output_model_file <string>]
        [--output_probs_file <string>] [--predictions_file <string>]
        [--probabilities_file <string>]
```

An implementation of the Naive Bayes Classifier, used for classification. Given labeled data, an NBC model can be trained and saved, or, a pre-trained model can be used for classification. Detailed documentation.

Input options

name	*type*	*description*	*default*
`--help (-h)`	**`flag`**	Default help info. Only exists in CLI binding.	
`--incremental_variance (-I)`	**`flag`**	The variance of each class will be calculated incrementally.	
`--info`	**`string`**	Print help on a specific option. Only exists in CLI binding.	`''`
`--input_model_file (-m)`	**`NBCModel file`**	Input Naive Bayes model.	`''`
`--labels_file (-l)`	**`1-d index matrix file`**	A file containing labels for the training set.	`''`
`--test_file (-T)`	**`2-d matrix file`**	A matrix containing the test set.	`''`
`--training_file (-t)`	**`2-d matrix file`**	A matrix containing the training set.	`''`
`--verbose (-v)`	**`flag`**	Display informational messages and the full list of parameters and timers at the end of execution.	
`--version (-V)`	**`flag`**	Display the version of mlpack. Only exists in CLI binding.	

Under the hood



Pros of C++

C++ is great!

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.
- Low-level memory management.

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.
- Low-level memory management.
- Little to no runtime overhead.

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.
- Low-level memory management.
- Little to no runtime overhead.
- Well-known!

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.
- Low-level memory management.
- Little to no runtime overhead.
- Well-known!
- The Armadillo library gives us good linear algebra primitives.

Pros of C++

C++ is great!

- Generic programming *at compile time* via templates.
- Low-level memory management.
- Little to no runtime overhead.
- Well-known!
- The Armadillo library gives us good linear algebra primitives.

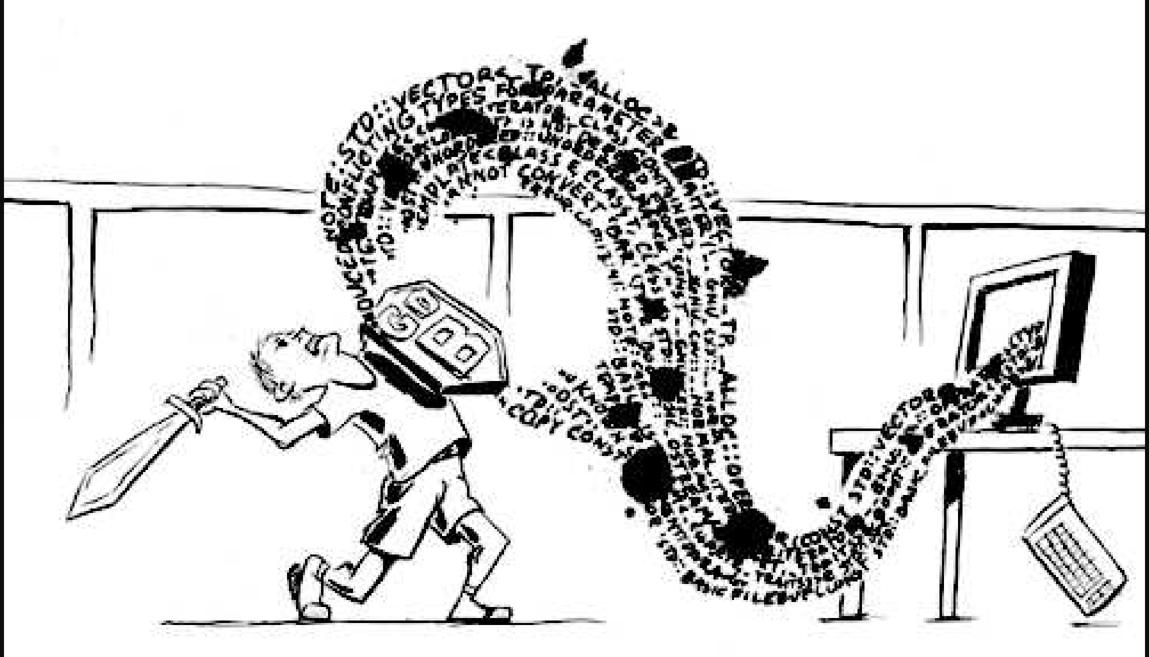
```
using namespace arma;
extern mat x, y;
mat z = (x + y) * chol(x) + 3 * chol(y.t());
```

Cons of C++

C++ is not great!

Cons of C++

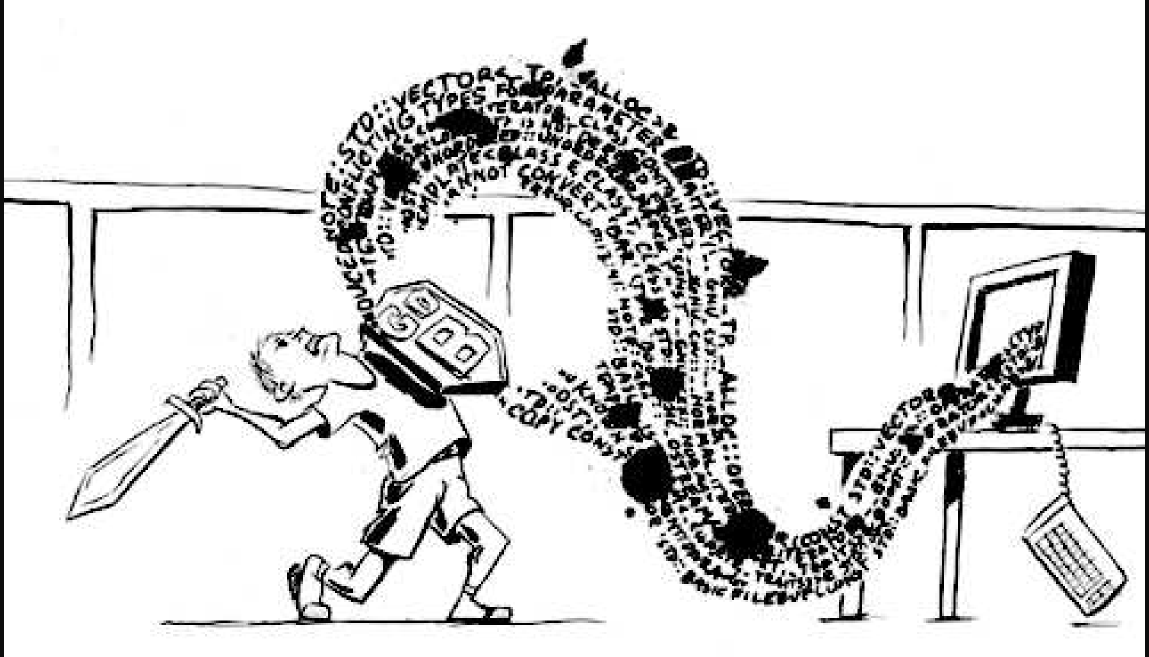
C++ is not great!


NOTE: STD::VECTOR<_TP, ALLOC>
CONFLICTING TYPES FOR PARAMETER
CANNOT CONVERT
COPY CONST
BASIC FILEBUF
GDB

- Templates can be hard to debug because of error messages.

Cons of C++

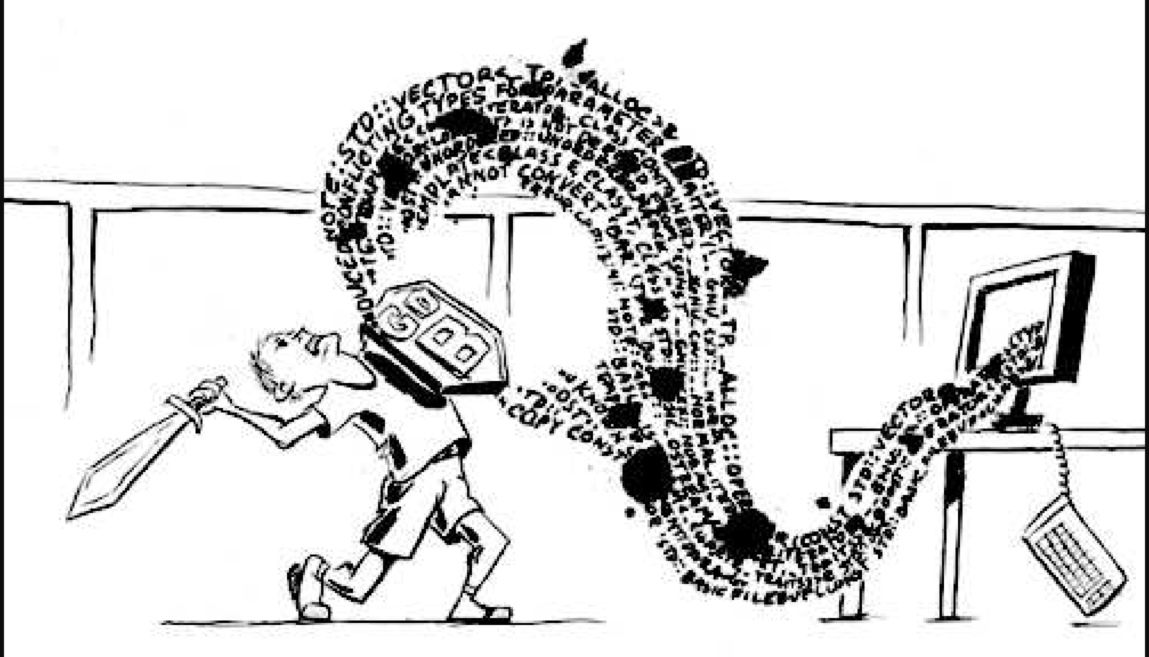
C++ is not great!



- Templates can be hard to debug because of error messages.
- Memory bugs are easy to introduce.

Cons of C++

C++ is not great!



- Templates can be hard to debug because of error messages.
- Memory bugs are easy to introduce.
- The new language revisions are not making the language any simpler...

Genericity

Why write an algorithm for one specific situation?

Genericity

Why write an algorithm for one specific situation?

```
NearestNeighborSearch n(dataset);
n.Search(query_set, 3, neighbors, distances);
```

What if I don’t want the Euclidean distance?

Genericity

Why write an algorithm for one specific situation?

```
// The numeric parameter is the value of p for the p-norm to
// use.  1 = Manhattan distance, 2 = Euclidean distance, etc.
NearestNeighborSearch n(dataset, 1);
n.Search(query_set, 3, neighbors, distances);
```

Ok, this is a little better!

Genericity

Why write an algorithm for one specific situation?

```
// ManhattanDistance is a class with a method Evaluate().
NearestNeighborSearch<ManhattanDistance> n(dataset);
n.Search(query_set, 3, neighbors, distances);
```

This is much better! The user can specify whatever distance metric they want, including one they write themselves.

Genericity

Why write an algorithm for one specific situation?

```cpp
// This will _definitely_ get me best paper at ICML!  I can
// feel it!
class MyStupidDistance
{
  static double Evaluate(const arma::vec& a,
                         const arma::vec& b)
  {
    return 15.0 * std::abs(a[0] - b[0]);
  }
};

// Now we can use it!
NearestNeighborSearch<MyStupidDistance> n(dataset);
n.Search(query_set, 3, neighbors, distances);
```

Genericity

Why write an algorithm for one specific situation?

```
// We can also use sparse matrices instead!
NearestNeighborSearch<MyStupidDistance, arma::sp_mat>
    n(sparse_dataset);
n.Search(sparse_query_set, 3, neighbors, distances);
```

Genericity

Why write an algorithm for one specific situation?

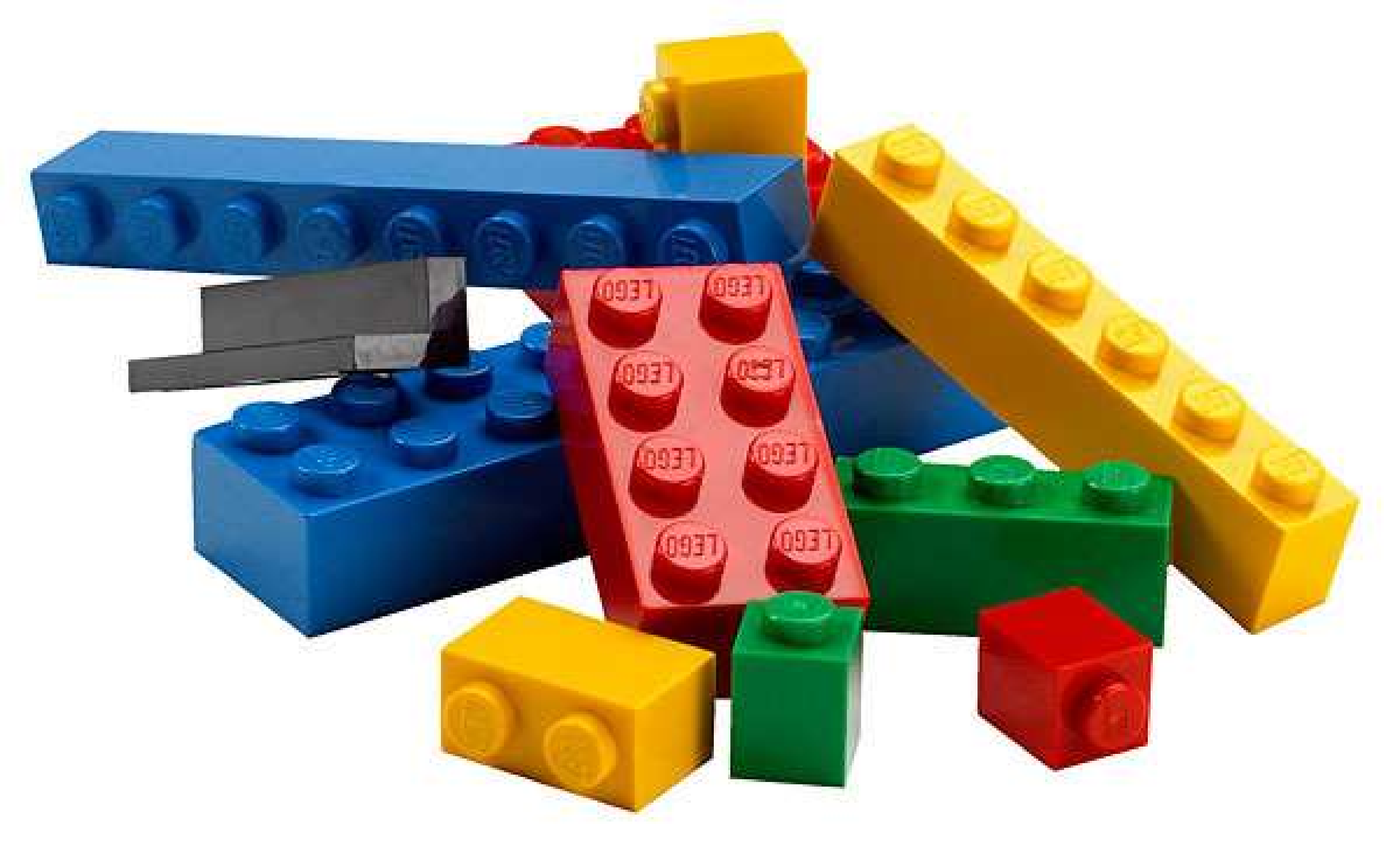
```
// Nearest neighbor search with arbitrary types of trees!
NearestNeighborSearch<EuclideanDistance, arma::mat, KDTree> kn;
NearestNeighborSearch<EuclideanDistance, arma::sp_mat, CoverTree> cn;
NearestNeighborSearch<ManhattanDistance, arma::mat, Octree> on;
NearestNeighborSearch<ChebyshevDistance, arma::sp_mat, RPlusTree> rn;
NearestNeighborSearch<MahalanobisDistance, arma::mat, RPTree> rpn;
NearestNeighborSearch<EuclideanDistance, arma::mat, XTree> xn;
```

R.R. Curtin, “Improving dual-tree algorithms”. *PhD thesis, Georgia Institute of Technology*, Atlanta, GA, 8/2015.

Genericity

Why write an algorithm for one specific situation?

```
// Near
Nearest
Nearest
Nearest
Nearest
Nearest
Nearest
```



R.R. Curtin, “Improving dual-tree algorithms”. *PhD thesis, Georgia Institute of Technology*, Atlanta, GA, 8/2015.

Genericity

Why wri

```
// Near
Nearest
Nearest
Nearest
Nearest
Nearest
Nearest
```

R.R. Curtin, “Improving dual-tree algorithms”. *PhD thesis, Georgia Institute of Technology*, Atlanta, GA, 8/2015.

Why templates?

What about virtual inheritance?

Why templates?

What about virtual inheritance?

```
class MyStupidDistance : public Distance
{
  virtual double Evaluate(const arma::vec& a,
                          const arma::vec& b)
  {
    return 15.0 * std::abs(a[0] - b[0]);
  }
};

NearestNeighborSearch n(dataset, new MyStupidDistance());
n.Search(3, neighbors, distances);
```

Why templates?

What about virtual inheritance?

```
class MyStupidDistance : public Distance
{
  virtual double Evaluate(const arma::vec& a,
                          const arma::vec& b)
  {
    return 15.0 * std::abs(a[0] - b[0]);
  }
};

NearestNeighborSearch n(dataset, new MyStupidDistance());
n.Search(3, neighbors, distances);
```

vtable lookup penalty!

Why templates?

Using inheritance to call a function costs us instructions:

```
Distance* d =
    new MyStupidDistance();
d->Evaluate(a, b);
```

```
MyStupidDistance::Evaluate(a, b);
```

Why templates?

Using inheritance to call a function costs us instructions:

Distance* d =
    new MyStupidDistance();
d->Evaluate(a, b);
	MyStupidDistance::Evaluate(a, b);

; push stack pointer movq %rsp, %rdi ; get location of function movq $_ZTV1A+16, (%rsp) ; call Evaluate() call _ZN1A1aEd	; just call Evaluate()! call _ZN1B1aEd.isra.0.constprop.1

Why templates?

Using inheritance to call a function costs us instructions:

`Distance* d =`
`    new MyStupidDistance();`
`d->Evaluate(a, b);`
	`MyStupidDistance::Evaluate(a, b);`

`; push stack pointer`
`movq  %rsp, %rdi`
`; get location of function`
`movq  $_ZTV1A+16, (%rsp)`
`; call Evaluate()`
`call  _ZN1A1aEd`
	`; just call Evaluate()!`
	`call _ZN1B1aEd.isra.0.constprop.1`

Up to 10%+ performance penalty in some situations!

Compile-time expressions

What about math? (Armadillo)

Compile-time expressions

What about math? (Armadillo)

In C:

```
extern double** a, b, c, d, e;
extern int rows, cols;

// We want to do e = a + b + c + d.
mat_copy(e, a, rows, cols);
mat_add(e, b, rows, cols);
mat_add(e, c, rows, cols);
mat_add(e, d, rows, cols);
```

Compile-time expressions

What about math? (Armadillo)

In C with a custom function:

```
extern double** a, b, c, d, e;
extern int rows, cols;

// We want to do e = a + b + c + d.
```

Compile-time expressions

What about math? (Armadillo)

In C with a custom function:

```
extern double** a, b, c, d, e;
extern int rows, cols;

// We want to do e = a + b + c + d.
mat_add4_into(e, a, b, c, d, rows, cols);
```

Fastest! (one pass)

Compile-time expressions

What about math? (Armadillo)

In C with a custom function:

```
extern double** a, b, c, d, e;
extern int rows, cols;

// We want to do e = a + b + c + d.
mat_add4_into(e, a, b, c, d, rows, cols);
```

Fastest! (one pass)

```
void mat_add4_into(double** e, double** a, double** b,
                   double** c, double** d, int rows, int cols)
{
  for (int r = 0; r < rows; ++r)
    for (int c = 0; c < cols; ++c)
      e[r][c] = a[r][c] + b[r][c] + c[r][c] + d[r][c];
}
```

Compile-time expressions

What about math? (Armadillo)

In MATLAB:

```
e = a + b + c + d
```

Compile-time expressions

What about math? (Armadillo)

In MATLAB:

```
e = a + b + c + d
```

Beautiful!

Compile-time expressions

What about math? (Armadillo)

Compile-time expressions

What about math? (Armadillo)

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

No temporaries, only one pass! Just as fast as the fastest C implementation.

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`

  → `op<mat, mat, add>`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`
  → `op<mat, mat, add>`
- `mat + mat + mat`
  → `op<mat, mat, add> + mat`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`
  → `op<mat, mat, add>`
- `mat + mat + mat`
  → `op<op<mat, mat, add>, mat, add>`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`
  → `op<mat, mat, add>`
- `mat + mat + mat`
  → `op<op<mat, mat, add>, mat, add>`
- `mat + mat + mat + mat`
  → `op<mat, mat, add> + mat + mat`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`
  → `op<mat, mat, add>`
- `mat + mat + mat`
  → `op<op<mat, mat, add>, mat, add>`
- `mat + mat + mat + mat`
  → `op<op<mat, mat, add>, mat, add> + mat`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

- `mat + mat`
  → `op<mat, mat, add>`
- `mat + mat + mat`
  → `op<op<mat, mat, add>, mat, add>`
- `mat + mat + mat + mat`
  → `op<op<op<mat, mat, add>, mat, add>, mat, add>`

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

The expression yields type `op<op<op<mat, mat, add>, mat, add>, mat, add>`.

```
// This can accept an op<...> type.
template<typename T1, typename T2>
mat::operator=(const op<T1, T2, add>& op);
```

Compile-time expressions

What about math? (Armadillo)

In C++ (with Armadillo):

```
using namespace arma;
extern mat a, b, c, d;

mat e = a + b + c + d;
```

C++ allows us templated operator overloading:

```
template<typename T1, typename T2>
const op<T1, T2, add> operator+(const T1& x, const T2& y);
```

The expression yields type `op<op<op<mat, mat, add>, mat, add>, mat, add>`.

```
// This can accept an op<...> type.
template<typename T1, typename T2>
mat::operator=(const op<T1, T2, add>& op);
```

The assignment operator "unwraps" the operation and generates optimal code.

Take-home

- Templates give us generic code.
- Templates allow us to generate fast code.

Deep Neural Networks with mlpack

With `ensmallen`, we can do deep learning.

Deep Neural Networks with mlpack

With `ensmallen`, we can do deep learning.

```
using namespace mlpack::ann;
extern arma::mat data, responses, testData;

// Create a 3-layer sigmoid neural network with 10 outputs.
FFN<NegativeLogLikelihood<>, RandomInitialization> model;
model.Add<Linear<>>(data.n_rows, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 10);
model.Add<LogSoftMax<>>();
```

Deep Neural Networks with mlpack

With `ensmallen`, we can do deep learning.

```
using namespace mlpack::ann;
extern arma::mat data, responses, testData;

// Create a 3-layer sigmoid neural network with 10 outputs.
FFN<NegativeLogLikelihood<>, RandomInitialization> model;
model.Add<Linear<>>(data.n_rows, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 10);
model.Add<LogSoftMax<>>();

// Train the model.
SGD<> optimizer(0.001 /* step size */, 1024 /* batch size */,
                100000 /* max iterations */);
model.Train(data, responses, optimizer);
```

Deep Neural Networks with mlpack

With `ensmallen`, we can do deep learning.

```
using namespace mlpack::ann;
extern arma::mat data, responses, testData;

// Create a 3-layer sigmoid neural network with 10 outputs.
FFN<NegativeLogLikelihood<>, RandomInitialization> model;
model.Add<Linear<>>(data.n_rows, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 100);
model.Add<SigmoidLayer<>>();
model.Add<Linear<>>(100, 10);
model.Add<LogSoftMax<>>();

// Train the model.
SGD<> optimizer(0.001 /* step size */, 1024 /* batch size */,
                100000 /* max iterations */);
model.Train(data, responses, optimizer);

// Predict on test points.
arma::mat predictions;
model.Predict(testData, predictions);
```

Benchmarks

Did C++ get us what we wanted?

Benchmarks

Task 1: $z = 2(x' + y) + 2(x + y')$.

```
extern int n;
mat x(n, n, fill::randu);
mat y(n, n, fill::randu);
mat z = 2 * (x.t() + y) + 2 * (x + y.t()); // only time this line
```

n	**arma**	**numpy**	**octave**	**R**	**Julia**
1000	0.029s	0.040s	0.036s	0.052s	**0.027s**
3000	0.047s	0.432s	0.376s	0.344s	**0.041s**
10000	**0.968s**	5.948s	3.989s	4.952s	3.683s
30000	**19.167s**	62.748s	41.356s	*fail*	36.730s

Benchmarks

Task 3: $z = abcd$ for decreasing-size matrices.

```
extern int n;
mat a(n, 0.8 * n, fill::randu);
mat b(0.8 * n, 0.6 * n, fill::randu);
mat c(0.6 * n, 0.4 * n, fill::randu);
mat d(0.4 * n, 0.2 * n, fill::randu);
mat z = a * b * c * d; // only time this line
```

n	**arma**	**numpy**	**octave**	**R**	**Julia**
1000	0.042s	0.051s	**0.033s**	0.056s	0.037s
3000	**0.642s**	0.812s	0.796s	0.846s	0.844s
10000	**16.320s**	26.815s	26.478s	26.957s	26.576s
30000	**329.87s**	708.16s	706.10s	707.12s	704.032s

Armadillo can automatically select the correct ordering for multiplication.

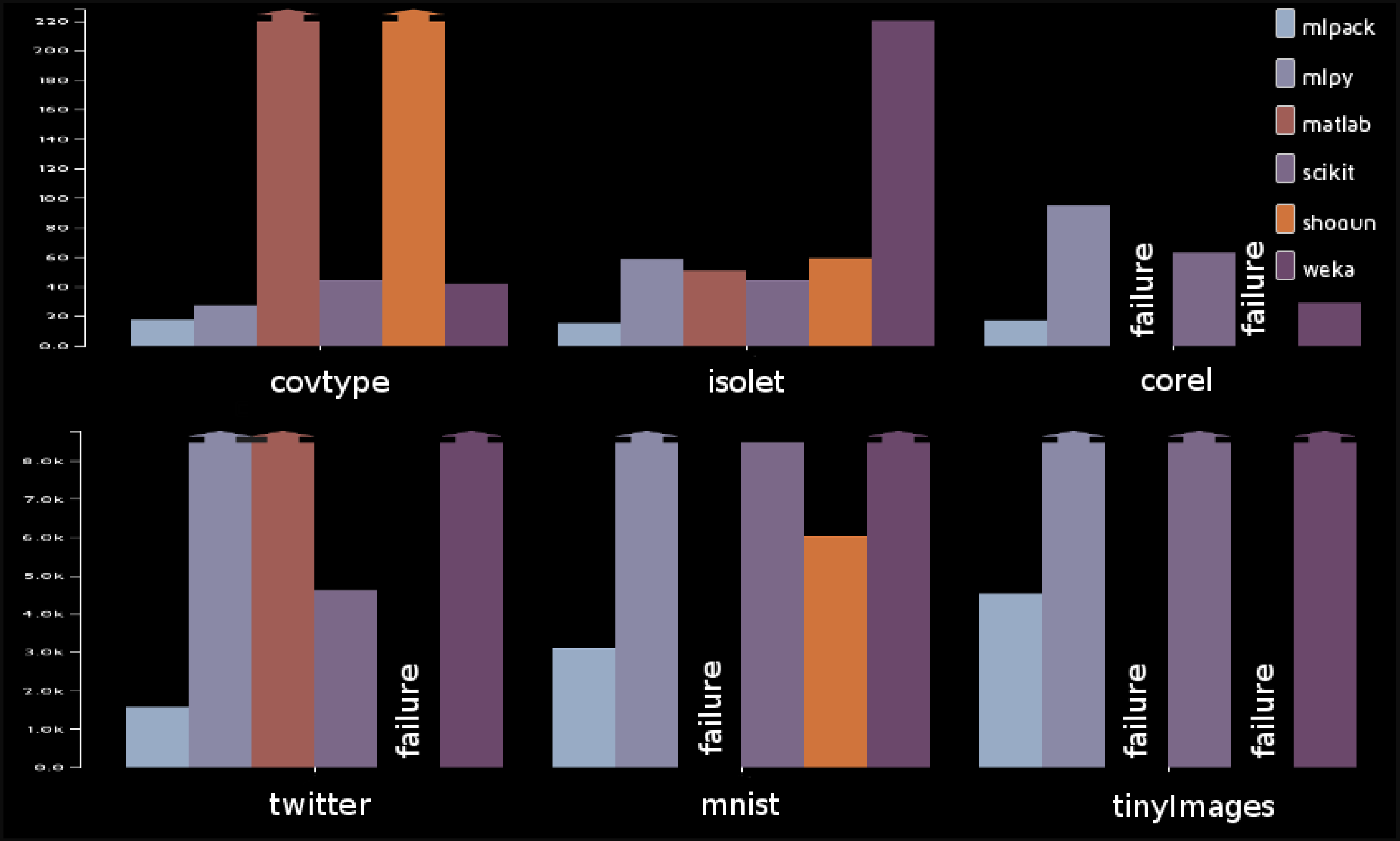
Benchmarks

Task 4: $z = a'(\mathrm{diag}(b)^{-1})c$.

```
extern int n;
vec a(n, fill::randu);
vec b(n, fill::randu);
vec c(n, fill::randu);
double z = as_scalar(a.t() * inv(diagmat(b)) * c); // only time this line
```

n	**arma**	**numpy**	**octave**	**R**	**Julia**
1k	**8e-6s**	0.100s	2e-4s	0.014s	0.057s
10k	**8e-5s**	49.399s	4e-4s	0.208s	18.189s
100k	**8e-4s**	*fail*	0.002s	*fail*	*fail*
1M	0.009s	*fail*	0.024s	*fail*	*fail*
10M	0.088s	*fail*	0.205s	*fail*	*fail*
100M	0.793s	*fail*	1.972s	*fail*	*fail*
1B	8.054s	*fail*	19.520s	*fail*	*fail*

kNN benchmarks



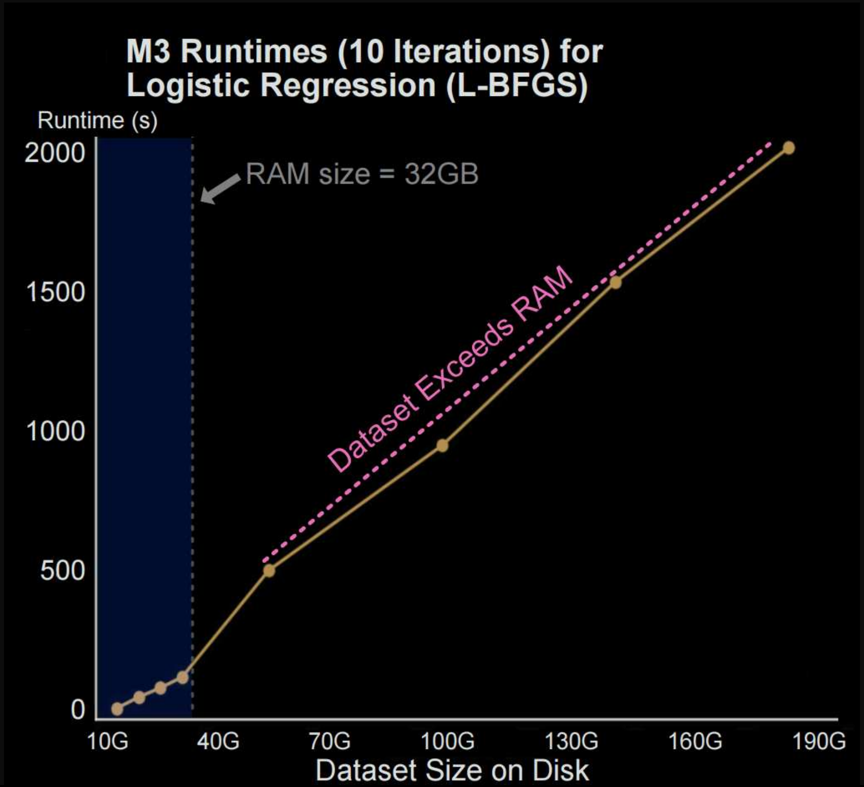
dataset	d	N	mlpack	mlpy	matlab	scikit	shogun	Weka
isolet	617	8k	**15.65s**	59.09s	50.88s	44.59s	59.56s	220.38s
corel	32	68k	**17.70s**	95.26s	*fail*	63.32s	*fail*	29.38s
covertype	54	581k	**18.04s**	27.68s	*>9000s*	44.55s	*>9000s*	42.34s
twitter	78	583k	**1573.92s**	*>9000s*	*>9000s*	4637.81s	*fail*	*>9000s*
mnist	784	70k	**3129.46s**	*>9000s*	*fail*	8494.24s	6040.16s	*>9000s*
tinyImages	384	100k	**4535.38s**	*9000s*	*fail*	*>9000s*	*fail*	*>9000s*

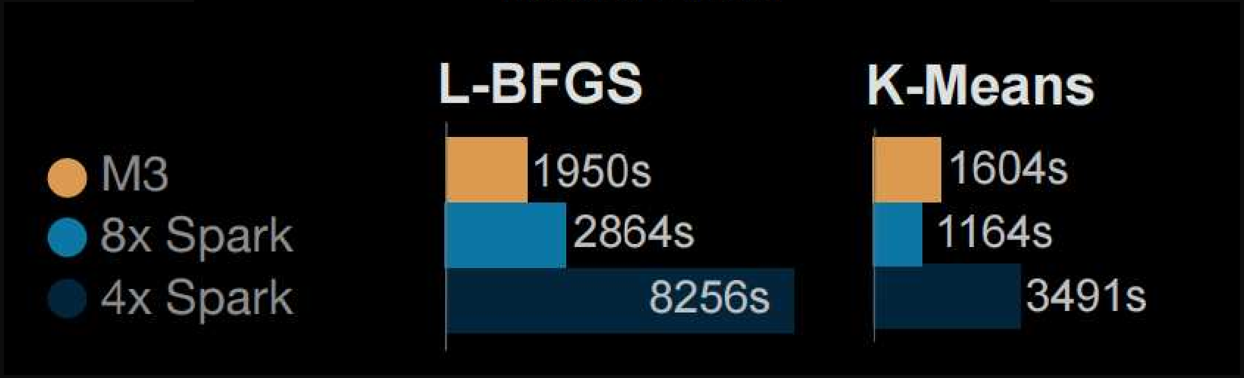
vs. Spark

We can use `mmap()` for out-of-core learning since our algorithms are generic!

vs. Spark

We can use `mmap()` for out-of-core learning since our algorithms are generic!





D. Fang, P. Chau. M3: scaling up machine learning via memory mapping, *SIGMOD/PODS 2016*.

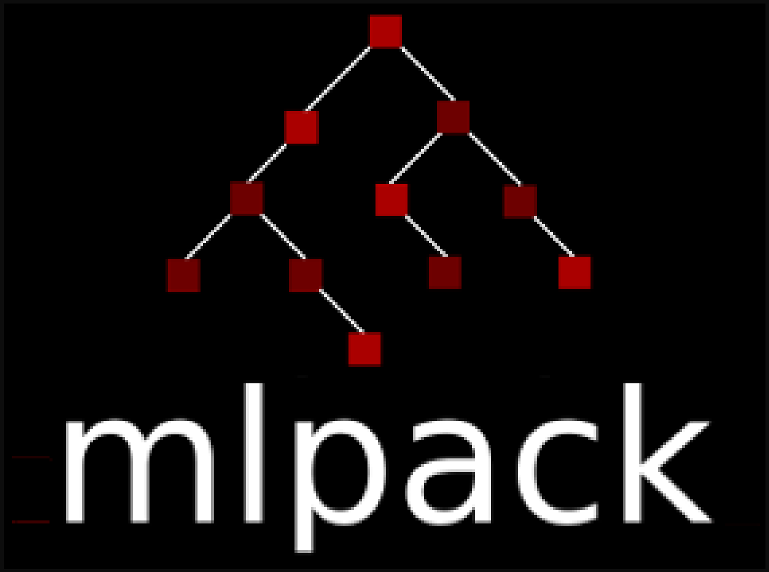
What didn’t I talk about in depth?

- optimization toolkit (`ensmallen`)
- hyper-parameter tuner
- tree infrastructure for problems like nearest neighbor search
- reinforcement learning code
- matrix decomposition infrastructure
- benchmarking system
- automatic binding generator
- preprocessing utilities
- ...and surely more I am not thinking of...

What’s coming?

mlpack 3.1.1 was just released and ready for production use!

`http://mlpack.org/blog/mlpack-3-released.html`

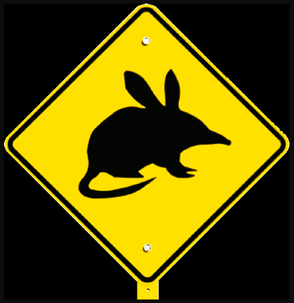


`http://www.mlpack.org/`

`https://github.com/mlpack/mlpack/`

Further out

Armadillo-like library for GPU matrix operations: **Bandicoot**



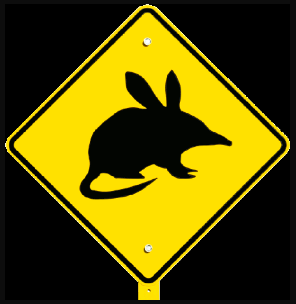
`http://coot.sourceforge.io/`

Two separate use case options:

- Bandicoot can be used as a drop-in accelerator to Armadillo, offloading intensive computations to the GPU when possible.
- Bandicoot can be used as its own library for GPU matrix programming.

Further out

Armadillo-like library for GPU matrix operations: **Bandicoot**



http://coot.sourceforge.io/

Two separate use case options:

- Bandicoot can be used as a drop-in accelerator to Armadillo, offloading intensive computations to the GPU when possible.
- Bandicoot can be used as its own library for GPU matrix programming.

```
using namespace coot;
mat x(n, n, fill::randu); // matrix allocated on GPU
mat y(n, n, fill::randu);
mat z = x * y; // computation done on GPU
```

Questions and comments?

`http://www.mlpack.org/`

`https://github.com/mlpack/mlpack/`