

Scikit-learn for easy machine learning: the vision, the tool, and the project

Gaël Varoquaux

Inria



1 Scikit-learn: the vision



1 Scikit-learn: the vision

An enabler

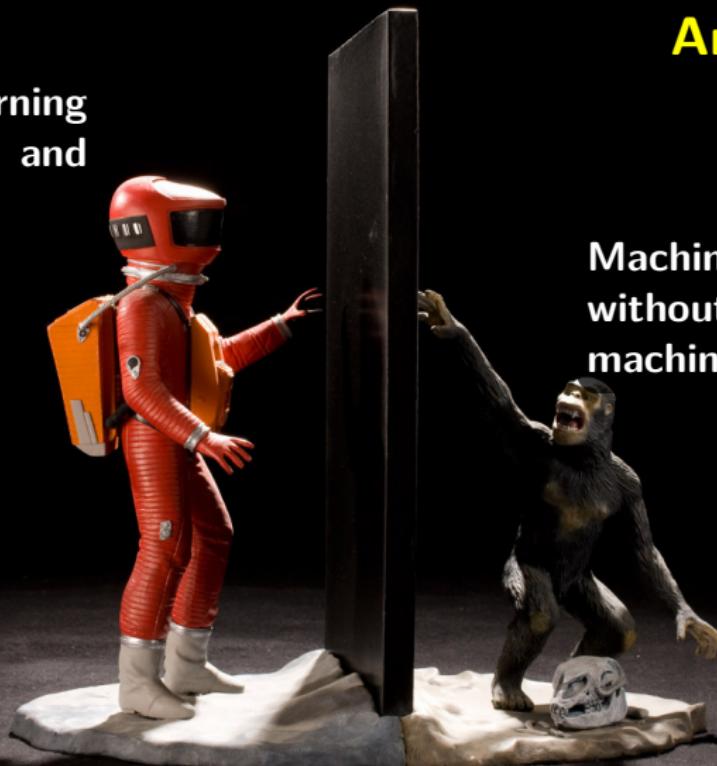


1 Scikit-learn: the vision

Machine learning
for everybody and
for everything

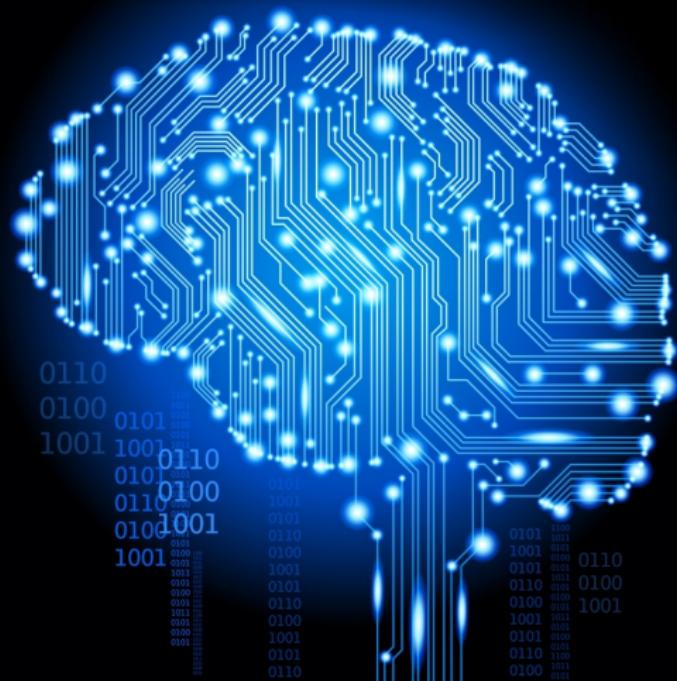
An enabler

Machine learning
without learning the
machinery



Machine learning in a nutshell

Machine learning is about making prediction from data

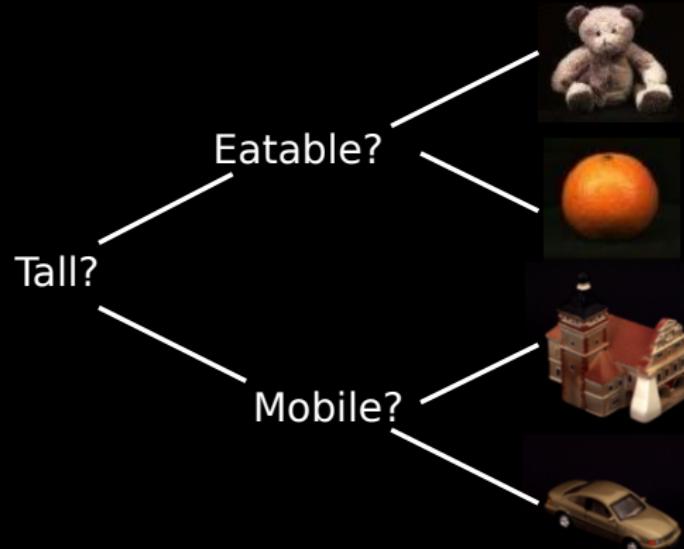


1 Machine learning: a historical perspective

Artificial Intelligence

- Building decision rules

The 80s



1 Machine learning: a historical perspective

Artificial Intelligence

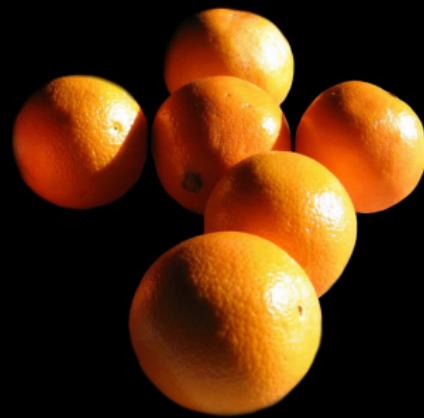
- Building decision rules

The 80s

Machine learning

- Learn these from observations

The 90s



1 Machine learning: a historical perspective

Artificial Intelligence

- Building decision rules

The 80s

Machine learning

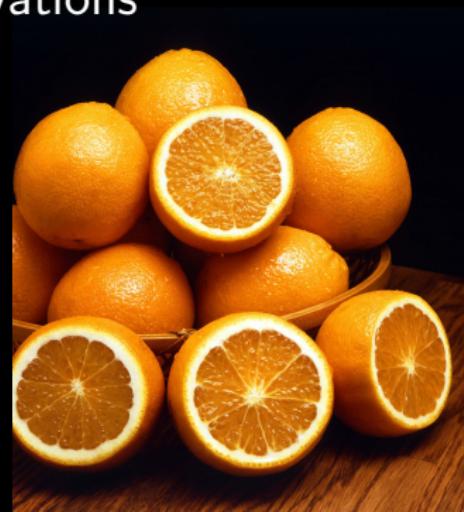
- Learn these from observations

The 90s

Statistical learning

- Model the noise in the observations

2000s



1 Machine learning: a historical perspective

Artificial Intelligence

- Building decision rules

The 80s

Machine learning

- Learn these from observations

The 90s

Statistical learning

- Model the noise in the observations

2000s

Big data

- Many observations,
simple rules

today



1 Machine learning: a historical perspective

Artificial Intelligence

- Building decision rules

The 80s

Machine learning

- Learn these from observations

The 90s

Statistical learning

- Model the noise in the observations

2000s

Big data

- Many observations,
simple rules

today



“Big data isn’t actually interesting without machine learning”

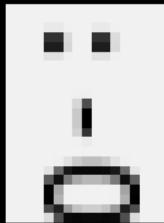
Steve Jurvetson, VC, Silicon Valley

1 Machine learning in a nutshell: an example

Face recognition



Andrew



Bill



Charles

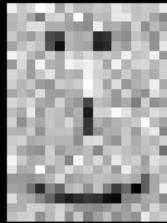


Dave

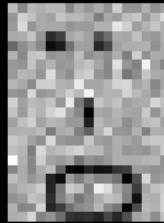


1 Machine learning in a nutshell: an example

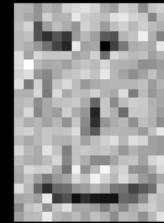
Face recognition



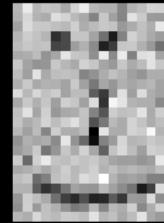
Andrew



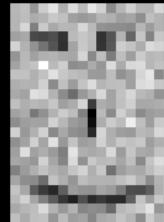
Bill



Charles



Dave



?

1 Machine learning in a nutshell

A simple method:

- 1 Store all the known (noisy) images and the names that go with them.
- 2 From a new (noisy) images, find the image that is most similar.

“Nearest neighbor” method



1 Machine learning in a nutshell

A simple method:

- 1 Store all the known (noisy) images and the names that go with them.
- 2 From a new (noisy) images, find the image that is most similar.

“Nearest neighbor” method

How many errors on already-known images?

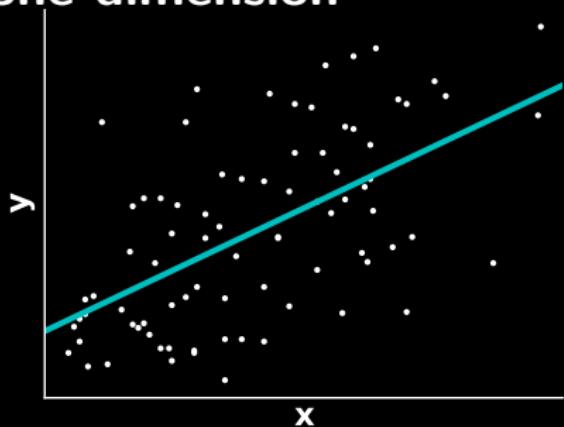
...

0: no erreurs

Test data \neq Train data

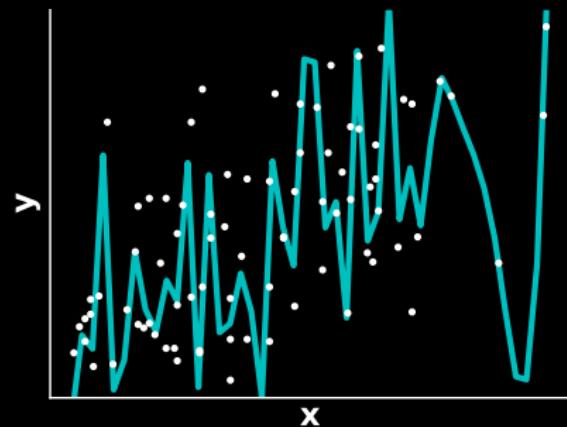
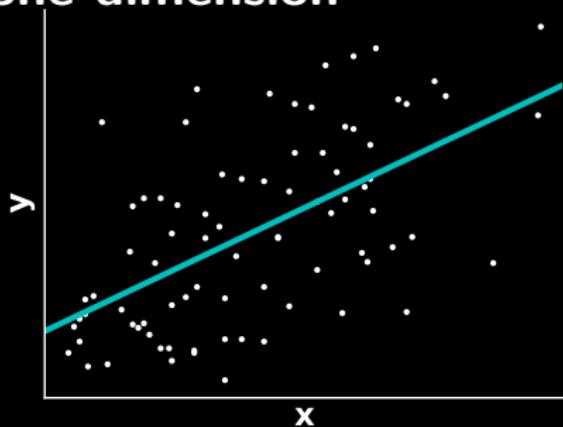
1 Machine learning in a nutshell: regression

A single descriptor:
one dimension



1 Machine learning in a nutshell: regression

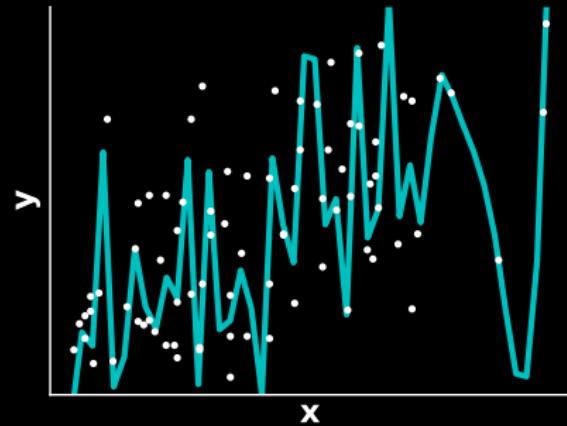
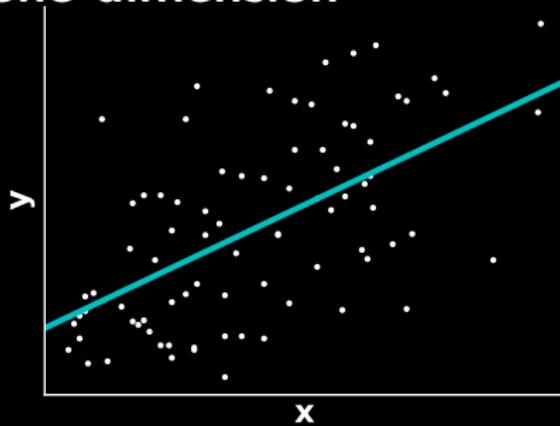
A single descriptor:
one dimension



Which model to prefer?

1 Machine learning in a nutshell: regression

A single descriptor:
one dimension

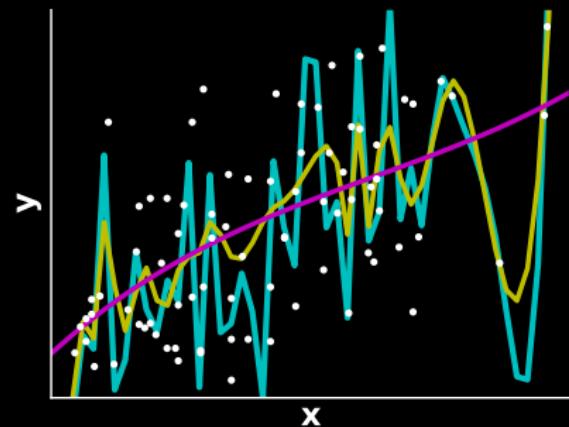
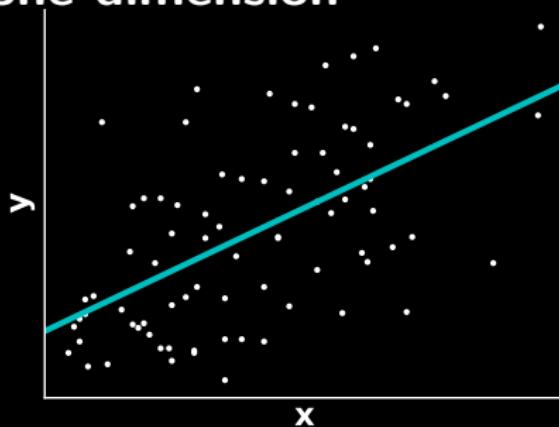


Problem of “*over-fitting*”

- Minimizing error is not always the best strategy
(learning noise)
- Test data \neq train data

1 Machine learning in a nutshell: regression

A single descriptor:
one dimension



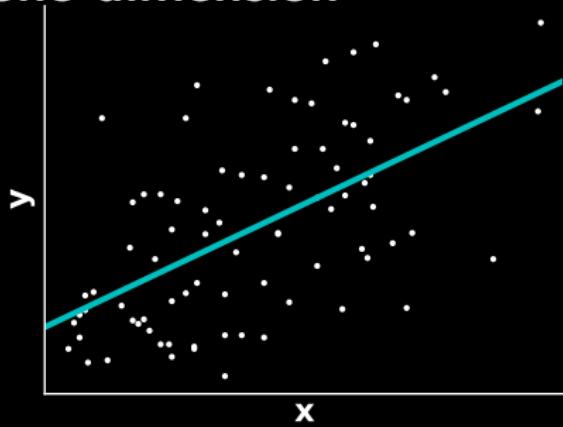
Prefer simple models

= concept of “*regularization*”

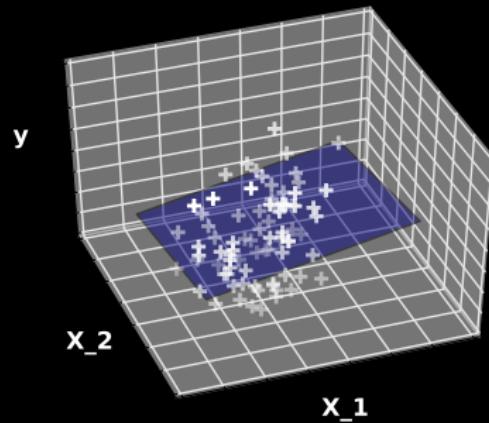
Balance the number of parameters to learn
with the amount of data

1 Machine learning in a nutshell: regression

A single descriptor:
one dimension



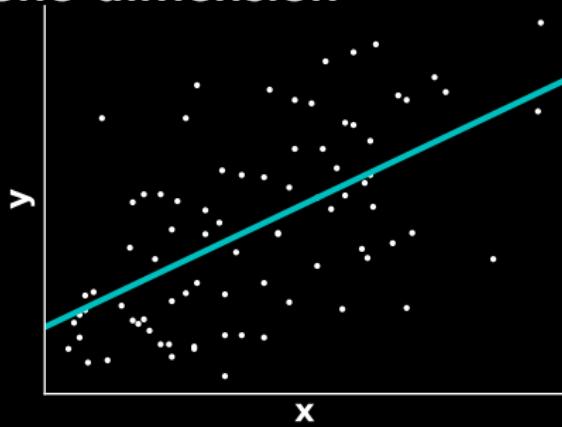
Two descriptors:
2 dimensions



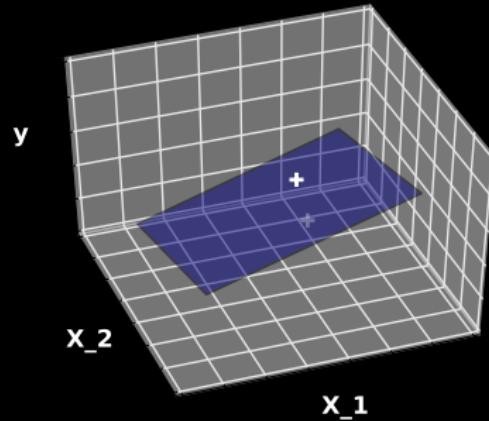
More parameters

1 Machine learning in a nutshell: regression

A single descriptor:
one dimension



Two descriptors:
2 dimensions

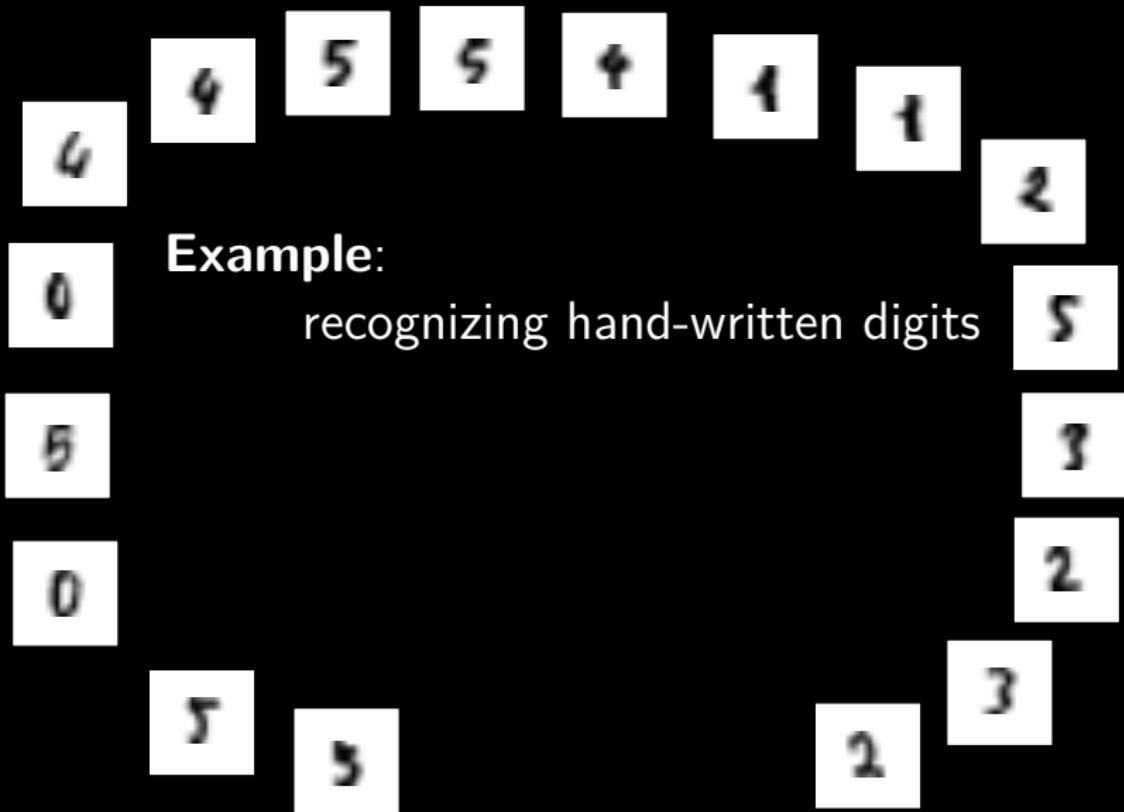


More parameters

⇒ need more data

“curse of dimensionality”

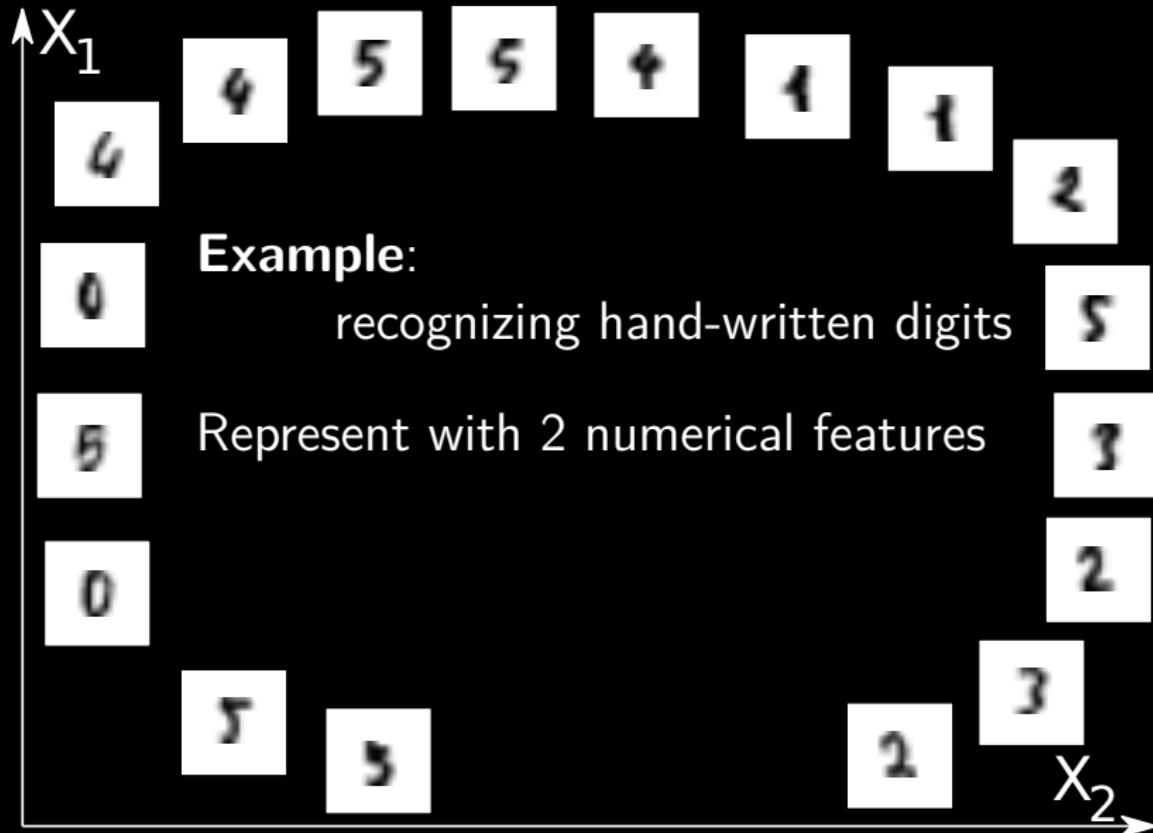
1 Machine learning in a nutshell: classification



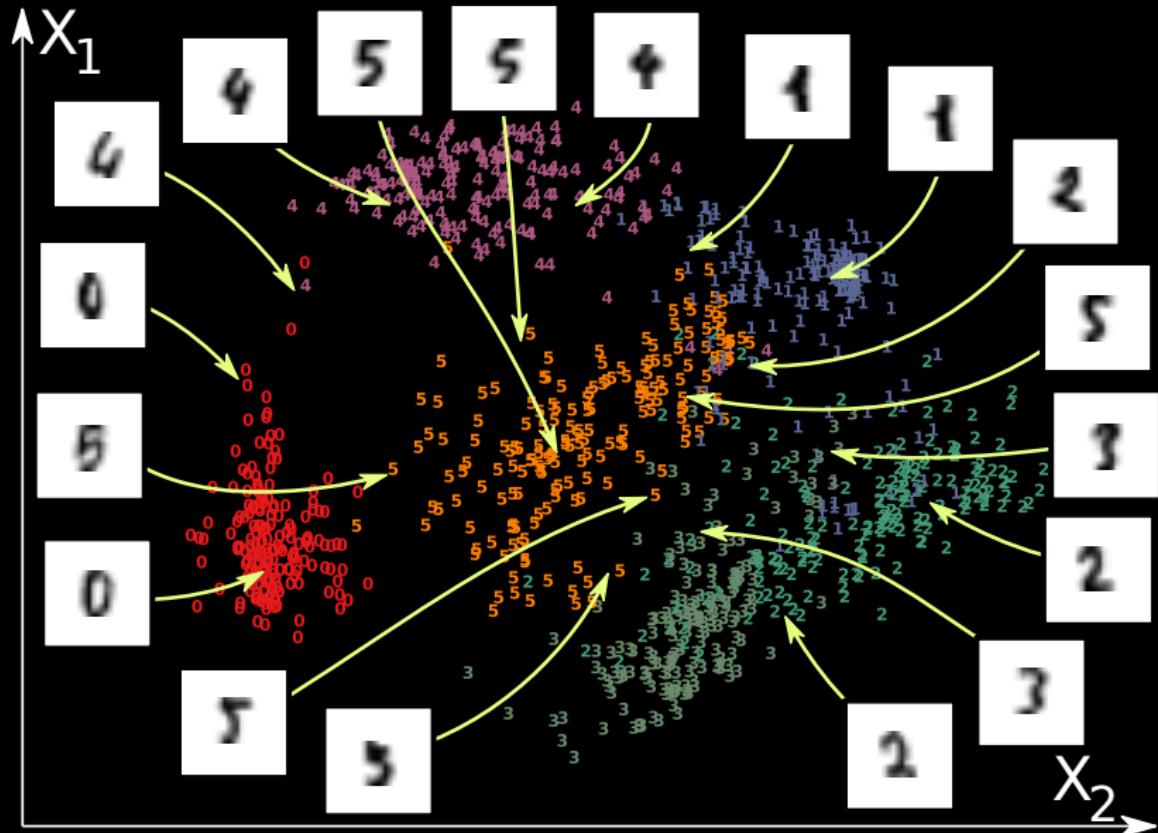
Example:

recognizing hand-written digits

1 Machine learning in a nutshell: classification

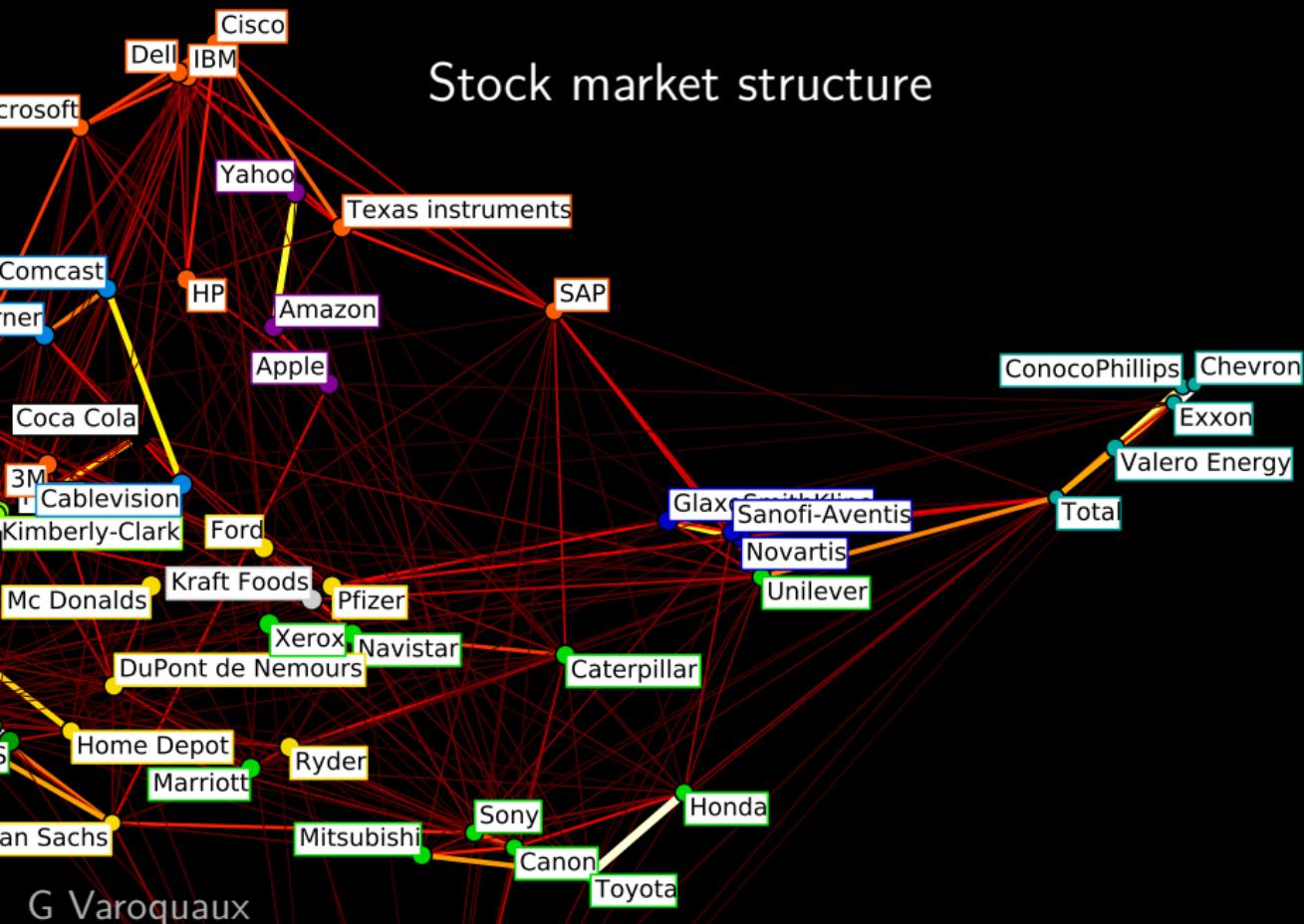


1 Machine learning in a nutshell: classification

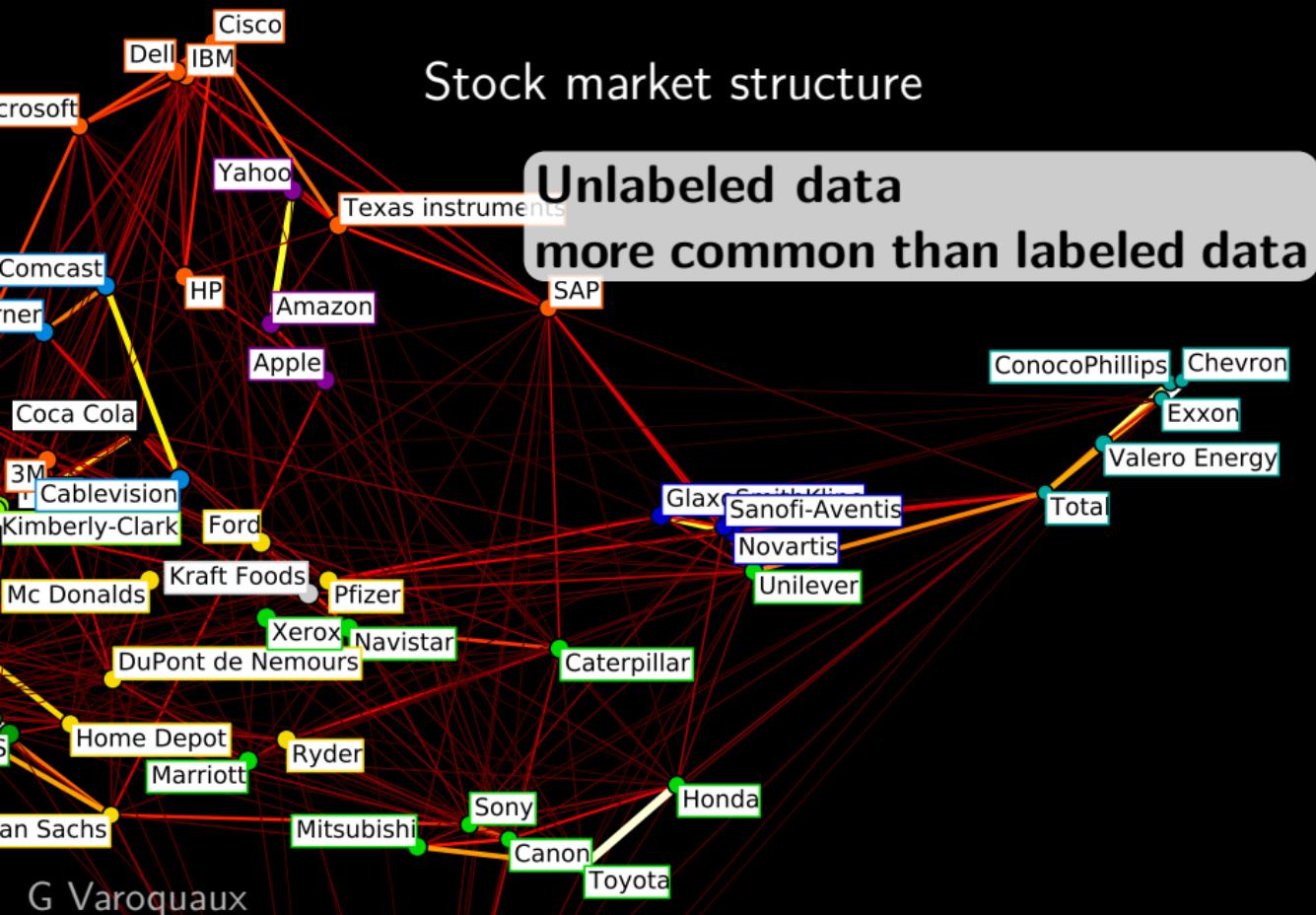


1 Machine learning in a nutshell: unsupervised

Stock market structure

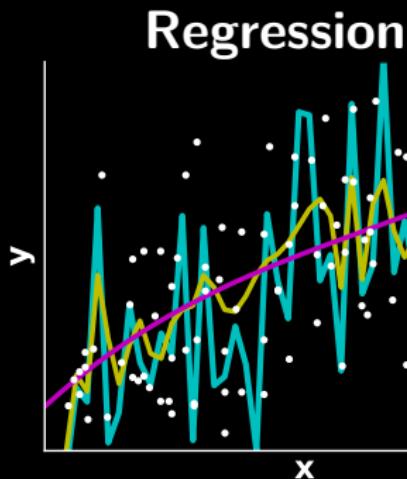


1 Machine learning in a nutshell: unsupervised



Machine learning

Mathematics and algorithms for fitting predictive models



Classification



Notions of overfit and test error

Machine learning is everywhere

- Image recognition
- Marketing (click-through rate)
- Movie / music recommendation
- Medical data
- Logistic chains (eg supermarkets)
- Language translation
- Detecting industrial failures



Why another machine learning package?



Real statisticians use R

- And real astronomers use IRAF
- Real economists use Gauss
- Real coders use C assembler
- Real experiments are controlled in Labview
- Real Bayesians use BUGS stan
- Real text processing is done in Perl
- Real Deep learner is best done with torch (Lua)
- And medical doctors only trust SPSS



Python, what else?

- General purpose
- Interactive language
- Easy to read / write



The scientific Python stack

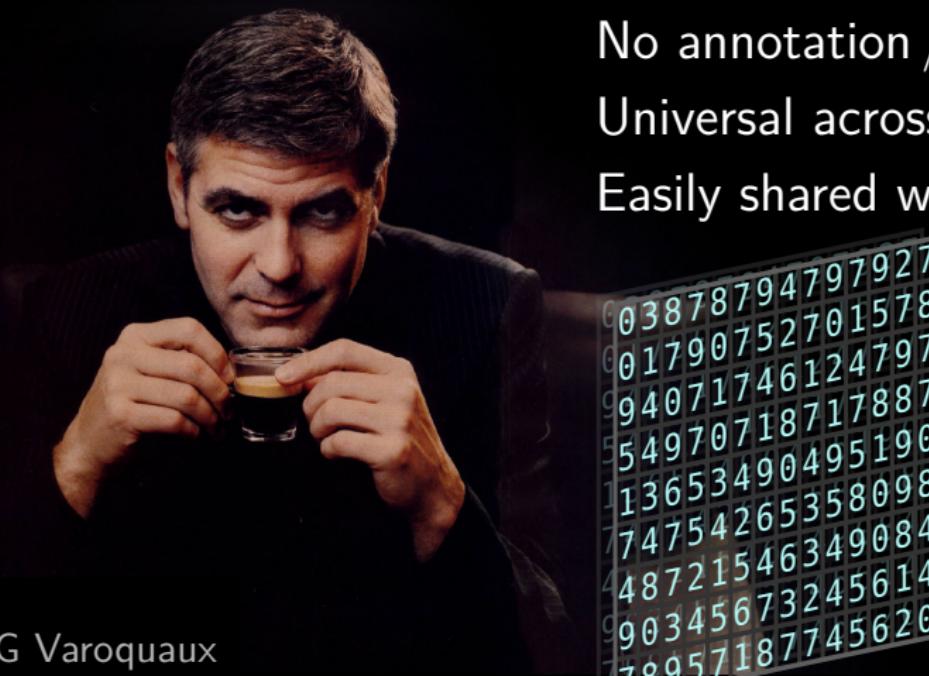
numpy arrays

Mostly a float**

No annotation / structure 😞

Universal across applications 😞

Easily shared with C / fortran



The scientific Python stack

numpy arrays

Connecting to

- scipy
- scikit-image
- pandas

...

It's about plugin things
together



The scientific Python stack

numpy arrays

Connecting to

- scipy
- scikit-image
- pandas

...

**Being Pythonic and
SciPythonic**

Machine learning for all

No specific application domain

No requirements in machine learning

High-quality Pythonic software library

Interfaces designed for users

Community-driven development

BSD licensed, very diverse contributors

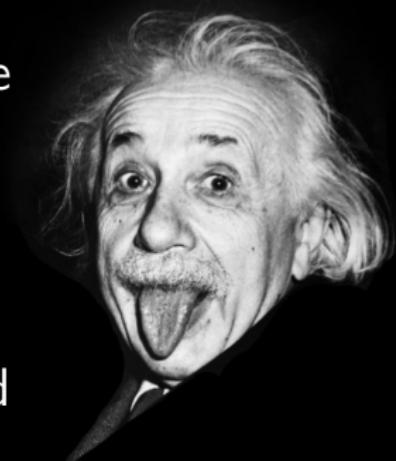
<http://scikit-learn.org>

Machine learning research

- Conceptual complexity is not an issue
- New and bleeding edge is better
- Simple problems are old science

In the field

- Tried and tested (aka boring) is good
- Little sophistication from the user
- API is more important than maths



Solving simple problems matters

Solving them really well matters a lot

2 Scikit-learn: the tool

A Python library for machine learning

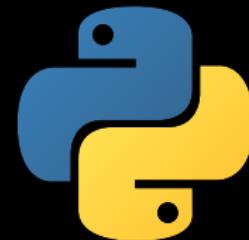


©Theodore W. Gray

2 A Python library

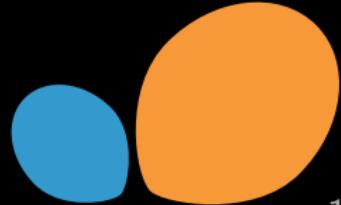
A library, not a program

- More expressive and flexible
- Easy to include in an ecosystem



As easy as py

```
from sklearn import svm  
classifier = svm.SVC()  
classifier.fit(X_train, Y_train)  
Y_test = classifier.predict(X_test)
```

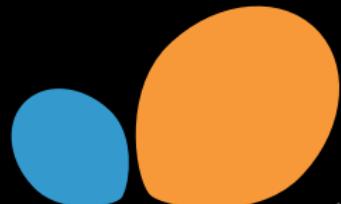


2 API: specifying a model

A central concept: **the estimator**

- Instantiated without data
- But specifying the parameters

```
from sklearn.neighbors import  
KNearestNeighbors  
  
estimator = KNearestNeighbors(  
    n_neighbors=2)
```



2 API: training a model

Training from data

```
estimator.fit(X_train, Y_train)
```

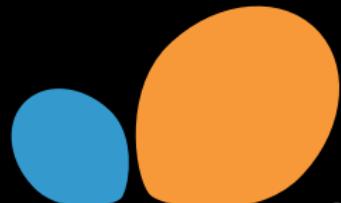
with:

- X a numpy array with shape

$$n_{\text{samples}} \times n_{\text{features}}$$

- y a numpy 1D array, of ints or float, with shape

$$n_{\text{samples}}$$



2 API: using a model

- Prediction: classification, regression

```
Y_test = estimator.predict(X_test)
```

- Transforming: dimension reduction, filter

```
X_new = estimator.transform(X_test)
```

- Test score, density estimation

```
test_score = estimator.score(X_test)
```



From raw data to a sample matrix X

- For text data: counting word occurrences
 - Input data: list of documents (string)
 - Output data: numerical matrix



data Python
more implementation
supervised efficient While
efficient some BSD statistical problems
some method implementations source Computing
method shogun pymvpa large
shogun other libsvm input objects compiled
other estimators platforms need use license
estimators input numpy & array learning provides
input object used scikit-learn interface
used provide set linear PCA libraries code memory
provide score engineering estimator high-level parameters
engineering estimator Journal MDP documentation
engineering estimator algorithms analysis performance
estimator machine scientific language
Scikit-learn learning

From raw data to a sample matrix \mathbf{X}

- For text data: counting word occurrences
 - Input data: list of documents (string)
 - Output data: numerical matrix

```
from sklearn.feature_extraction.text  
      import HashingVectorizer  
hasher = HashingVectorizer()  
  
X = hasher.fit_transform(documents)
```

2 Scikit-learn: very rich feature set

Supervised learning

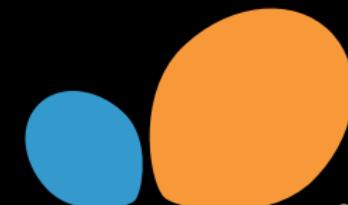
- Decision trees (Random-Forest, Boosted Tree)
- Linear models
- SVM

Unsupervised Learning

- Clustering
- Dictionary learning
- Outlier detection

Model selection

- Built in cross-validation
- Parameter optimization

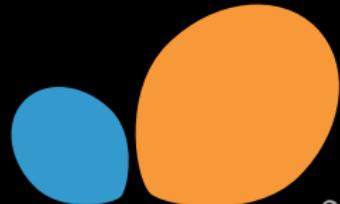


2 Computational performance

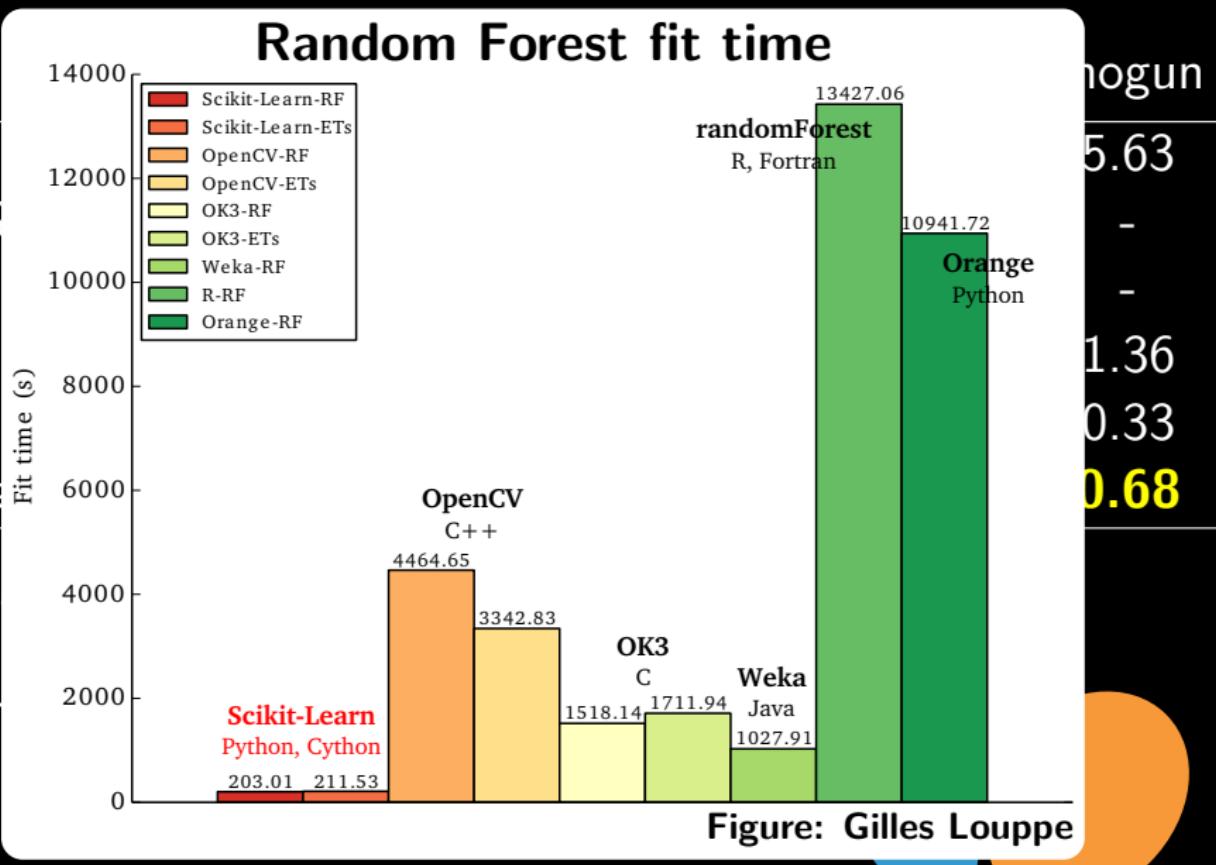
	scikit-learn	mlpy	pybrain	pymvpa	mdp	shogun
SVM	5.2	9.47	17.5	11.52	40.48	5.63
LARS	1.17	105.3	-	37.35	-	-
Elastic Net	0.52	73.7	-	1.44	-	-
kNN	0.57	1.41	-	0.56	0.58	1.36
PCA	0.18	-	-	8.93	0.47	0.33
k-Means	1.34	0.79	∞	-	35.75	0.68

■ Algorithmic optimizations

■ Minimizing data copies



2 Computational performance



What if the data does not fit in memory?



“Big data”:

- Petabytes...
- Distributed storage
- Computing cluster

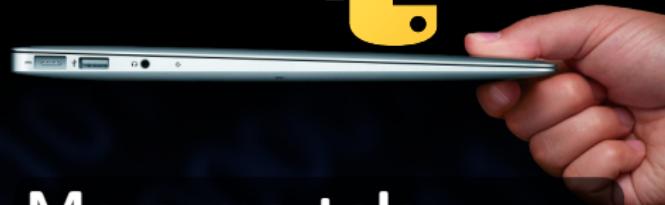
What if the data does not fit in memory?



“Big data”:

- Petabytes...
- Distributed storage
- Computing cluster

WE ARE THE
99%



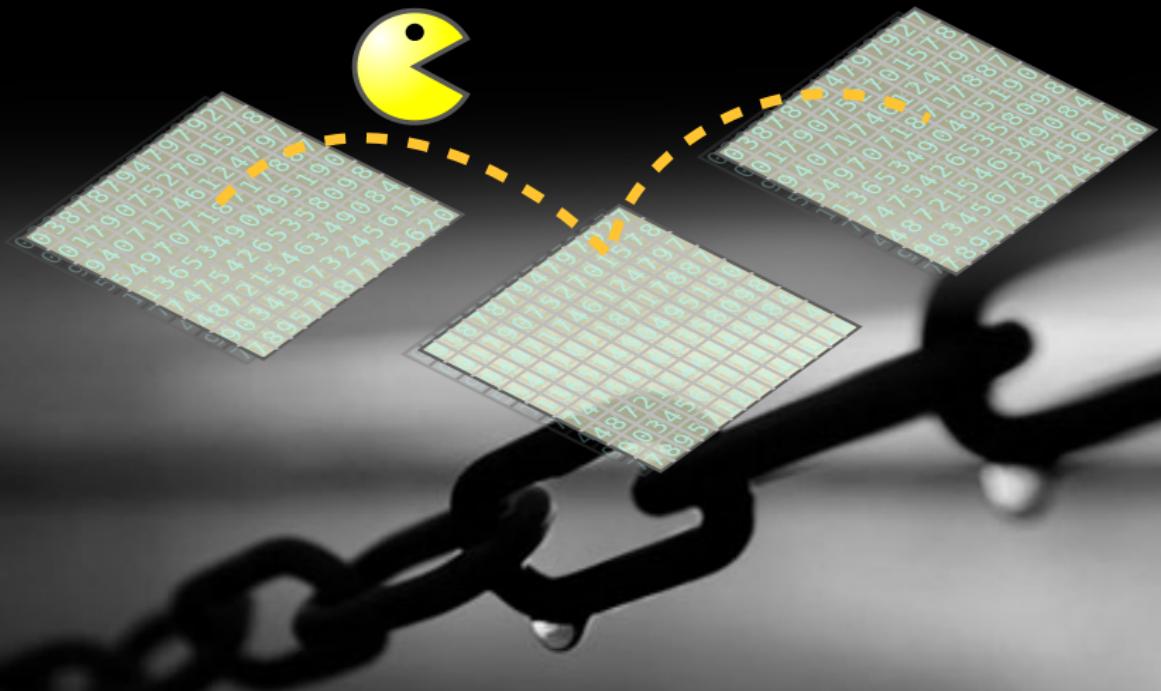
Mere mortals:

- Gigabytes...
- Python programming
- Off-the-self computers

See also: <http://www.slideshare.net/GaelVaroquaux/processing-biggish-data-on-commodity-hardware-simple-python-patterns>

2 On-line algorithms

```
estimator.partial_fit(X_train, Y_train)
```



2 On-line algorithms

```
estimator.partial_fit(X_train, Y_train)
```

Linear models

```
sklearn.linear_model.SGDRegressor
```

```
sklearn.linear_model.SGDClassifier
```

Clustering

```
sklearn.cluster.MiniBatchKMeans
```

```
sklearn.cluster.Birch      (new in 0.16)
```

PCA (new in 0.16)

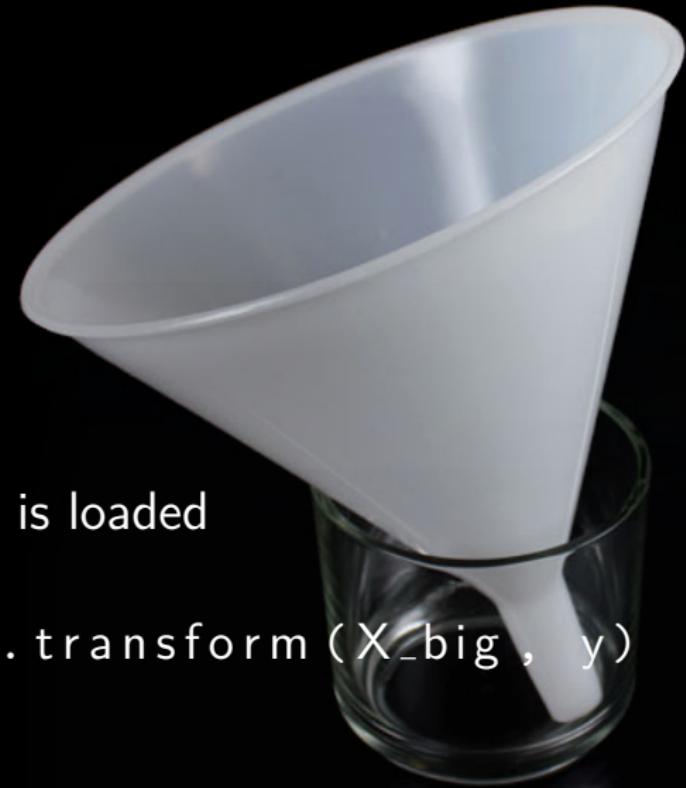
```
sklearn.decompositions.IncrementalPCA
```

2 On-the-fly data reduction

Many features

⇒ Reduce the data as it is loaded

```
X_small = estimator.transform(X_big, y)
```



2 On-the-fly data reduction

Random projections (will average features)

`sklearn.random_projection`
random linear combinations of the features

Fast clustering of features

`sklearn.cluster.FeatureAgglomeration`
on images: super-pixel strategy

Hashing when observations have varying size

(e.g. words)

`sklearn.feature_extraction.text.`

`HashingVectorizer`

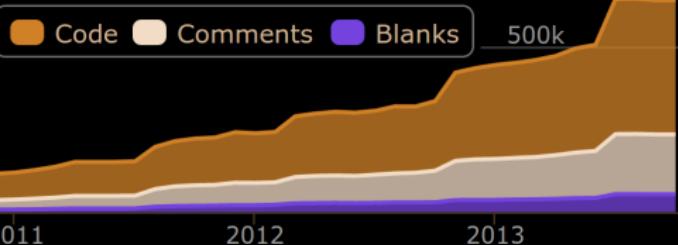
stateless: can be used in parallel

3 Scikit-learn: the project

Heribred Christian Ossendorfer Yann N. Dauphin Thomas Unterthiner Aaron Schumacher
ugurthmaster Andrei Wirlman Jan Hendrik Metzen Michael Eickerberg Dario Duckwirth
ellermann mr.Shi Michael Bommarito Alexander Fabisch James Bergstra Amt Akles Andrew Tulach
Jeffrey Blackburne Koenig Kernal Eren Nicolas Pinto Cosy Lynch Martin Lüssi
Mathieu Blondel Yannick Schwartz Edouard DUCHESNAY Kyle Beauchamp
Denis Engemann Noel Dawe Clay Woolam Hamzeh Alsaifi Baptiste Lagarde
Salvatore Missocchia David Marek unknown Yaroslav Halchenko James Bergstra Amt Akles Andrew Tulach
Vincent Schulz Jean Kosalla Noel Dawe Clay Woolam Hamzeh Alsaifi Baptiste Lagarde
unknown Bertrand Thirion Brian Holt
Luís Pedro Coelho Sérgio Gómez Nelle Varoquaux
Sebastien Seznec Doug Coleman
Amra Waterfall Robert Sylvw
Nikolay Mayorov Loïc Estève
Brooke Osborn
Yu-Chin Richard T. Guy Vlad Niculae Arnaud Joly Lars Buitinck Andreas Mueller
A. Flaxman Manoj Kumar David Wande-Farley Brian Caije
Richard T. Guy Vlad Niculae Arnaud Joly Lars Buitinck Andreas Mueller
David Wande-Farley Brian Caije
Kyle Kastner Daniel Noun CJ Carey
CJ Carey Conrad Lee Ben Root akshayank sgrishke
Tadej Jereb Jake VanderPlas Federico Harieu
Imran Haque Immanuel Bayer Federico Vaggi
Hamed Ramin Vincent Michel Sathya Rajagopalan
Nick Williams Ron Weiss Akashdeep Agrawal
Raul Garnelo Raghav R Vandy
Chen Liu Hervé Chapel Ignacio Rossi
Hector Ballesteros Tiago Nunes Andrei Mignot Xintan Meng
Sergio Guadalupe Arne Mikalsen Vinayak Mehta
Giovanni Susto Alessio Losi Rob Zinkov Mario Burak
Alessio Losi Alessio Losi Shiqiao Du Adrien Gadon
Keny Goto
Gael Varoquaux Olivier Grisel Gilles Louppe Fabian Pedregosa
Manoj Kumar David Wande-Farley Brian Caije
Robert Layton Conrad Lee Ben Root akshayank sgrishke
Joel Nothman Tadej Jereb Jake VanderPlas Federico Harieu
Jake VanderPlas Federico Harieu
David Wande-Farley Brian Caije
Kyle Kastner Daniel Noun CJ Carey
CJ Carey Conrad Lee Ben Root akshayank sgrishke
Tadej Jereb Jake VanderPlas Federico Harieu
Immanuel Bayer Federico Vaggi
Vincent Michel Sathya Rajagopalan
Ron Weiss Akashdeep Agrawal
Raghav R Vandy Ignacio Rossi
Daniele Vilani Andrei Mignot Xintan Meng
Andrea Mignot Vinayak Mehta
Paolo Losi Rob Zinkov Mario Burak
Paolo Losi Alessio Losi Shiqiao Du Adrien Gadon
Alessio Losi Alessio Losi Shiqiao Du Adrien Gadon
Rajat Khurana Satrajit Ghosh Jaques Grobler Gilles Louppe Fabian Pedregosa
soentaraq Nicolas Tresquin Philippe Gervais Michael Korobov Balu Subrahmanyam Varanasi
queqiqihi
Alexis Metzressi Vincent Dubourg Matthieu Perrot Peter Prettenhofer
Hervé Jégouau Pet-Antoine Forte Thibaut Lecoutre
Thouis Jones Robert Marchman Peter Prettenhofer
Ronald Phlypo
Emmanuelle Baudot Sandeep Mohapatra Wei Li Clemens Brunner Martin Billinger
Eustache Diemert Virgil Fritsch Pietro Berker Alexander Amini
Tim Sheehan-Chase Ben Holzman Florian Wilhelm Robert McGibbon
Ilanbarathi Karanth Yannick Schwartz Alexandre Passos Christian Jauvin
Yannick Schwartz Edouard DUCHESNAY maheshakya Daniel Ober
Baptiste Lagarde
Yannick Schwartz Edouard DUCHESNAY Oscar Navaea ApproximateIdentity

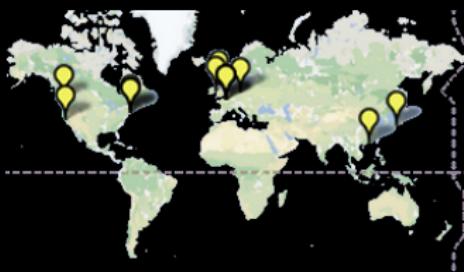
3 Community-based development in scikit-learn

Huge feature set:
benefits of a large team



Project growth:

- More than 200 contributors
- ~ 12 core contributors
- 1 full-time INRIA programmer from the start



Estimated cost of development: \$ 6 millions

COCOMO model,

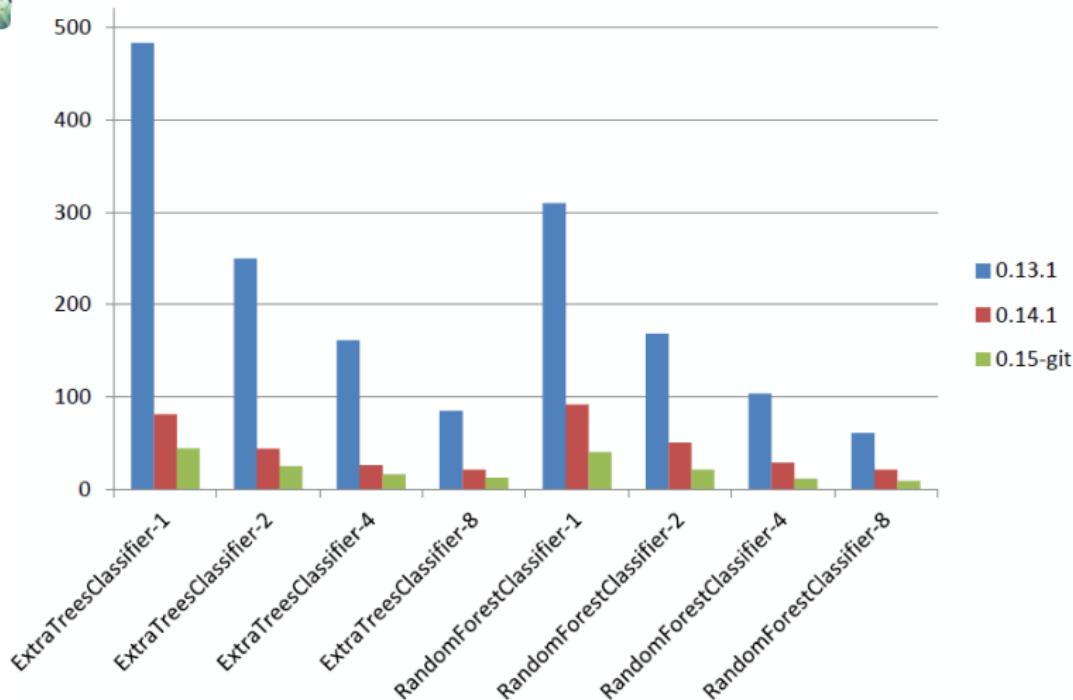
<http://www.ohloh.net/p/scikit-learn>

3 Many eyes makes code fast



Gilles Louuppe @glouppe · Feb 18

Speed improvement from 0.13 to 0.15-git of Random Forests in Scikit-Learn:



L. Buitinck, O. Grisel, A. Joly, G. Louuppe, J. Nothman, P. Prettenhofer

3 6 steps to a community-driven project

- 1 Focus on **quality**
- 2 Build great **docs and examples**
- 3 Use **github**
- 4 Limit the technicality of your codebase
- 5 Releasing and packaging matter
- 6 Focus on your contributors,
give them credit, decision power



[http://www.slideshare.net/GaelVaroquaux/
scikit-learn-dvelopement-communautaire](http://www.slideshare.net/GaelVaroquaux/scikit-learn-dvelopement-communautaire)

3 Quality assurance

Code review: pull requests

- Can include newcomers
- We read each others code
- Everything is discussed:
 - Should the algorithm go in?
 - Are there good defaults?
 - Are names meaningful?
 - Are the numerics stable?
 - Could it be faster?

The screenshot shows a GitHub pull request interface. At the top, a message from agramfort states: "agramfort started a discussion in the diff 7 months ago". Below this is a code diff for the file `sklearn/cluster/_inertia.pyx`. Lines 21 through 39 are shown, with line 24 highlighted in red and line 39 highlighted in green. A comment from agramfort follows, asking if the new implementation is numerically less stable but justified by speed. jmetzen replies that it is right and reverts it to the old implementation. At the bottom, a commit from jmetzen is shown: "ENH: CompleteLinkage supports more efficient".

```
...  ... @@ -21,9 +36,9 @@ def compute_ward_dist(np.ndarray[D
21  36           for i in range(size_max):
22  37             row = coord_row[i]
23  38             col = coord_col[i]
24 - n = (m_1[row] * m_1[col]) / (m_1[row] + m_1
25 + n = 1.0 / (1.0 / m_1[row] + 1.0 / m_1[col])
26
27
28
29
30
31
32
33
34
35
36
37
38
39 +
```

agramfort started a discussion in the diff 7 months ago

sklearn/cluster/_inertia.pyx

... ... @@ -21,9 +36,9 @@ def compute_ward_dist(np.ndarray[D

21 36 for i in range(size_max):
22 37 row = coord_row[i]
23 38 col = coord_col[i]
24 - n = (m_1[row] * m_1[col]) / (m_1[row] + m_1
25 + n = 1.0 / (1.0 / m_1[row] + 1.0 / m_1[col])
26
27
28
29
30
31
32
33
34
35
36
37
38
39 +

2

agramfort repo collab

i am afraid this is numerically less stable. it is justified by speed?

jmetzen

you are right, I reverted it to the old implementation

Add a line note

jmetzen added some commits

8e00db8 ENH: CompleteLinkage supports more efficient

Unit testing

- Everything is tested
- Great for numerics
- Overall tests enforce on all estimators
 - consistency with the API
 - basic invariances
 - good handling of various inputs



Make it work, make it right, make it boring

3 The tragedy of the commons

Individuals, acting independently and rationally according to each one's self-interest, behave contrary to the whole group's long-term best interests by depleting some common resource.

Wikipedia



Make it work, make it right, make it boring

Core projects (boring) taken for granted

⇒ Hard to fund, less excitement

They need citation, in papers & on corporate web pages

3 The tragedy of the commons

Individuals, acting independently and rationally according to each one's self-interest, behave contrary to the whole group's long-term best interests by depleting some common resource.

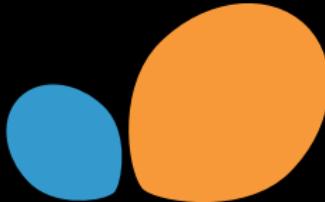
Wikipedia

- + It's so hard to scale
- User support
- Growing codebase

Make it work, make it right, make it boring

Core projects (boring) taken for granted
⇒ Hard to fund, less excitement

They need citation, in papers & on corporate web pages

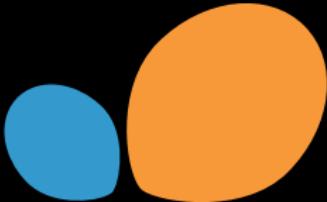


The vision

Machine learning as a means not an end
Versatile library: the “right” level of abstraction
Close to research, but seeking different tradeoffs



Scikit-learn



The vision

Machine learning as a means not an end

The tool

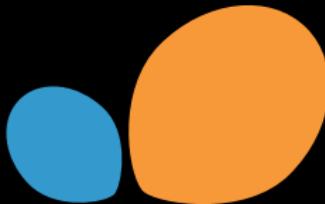
Simple API uniform across learners

Numpy matrices as data containers

Reasonnably fast



Scikit-learn



The vision

Machine learning as a means not an end

The tool

Simple API uniform across learners

The project

Many people working together
Tests and discussions for quality



We're hiring!