Scikit-learn for easy machine learning:

the vision, the tool, and the project

Gaël Varoquaux

Inria

scikit learn

machine learning in Python

1 Scikit-learn: the vision

1 Scikit-learn: the vision

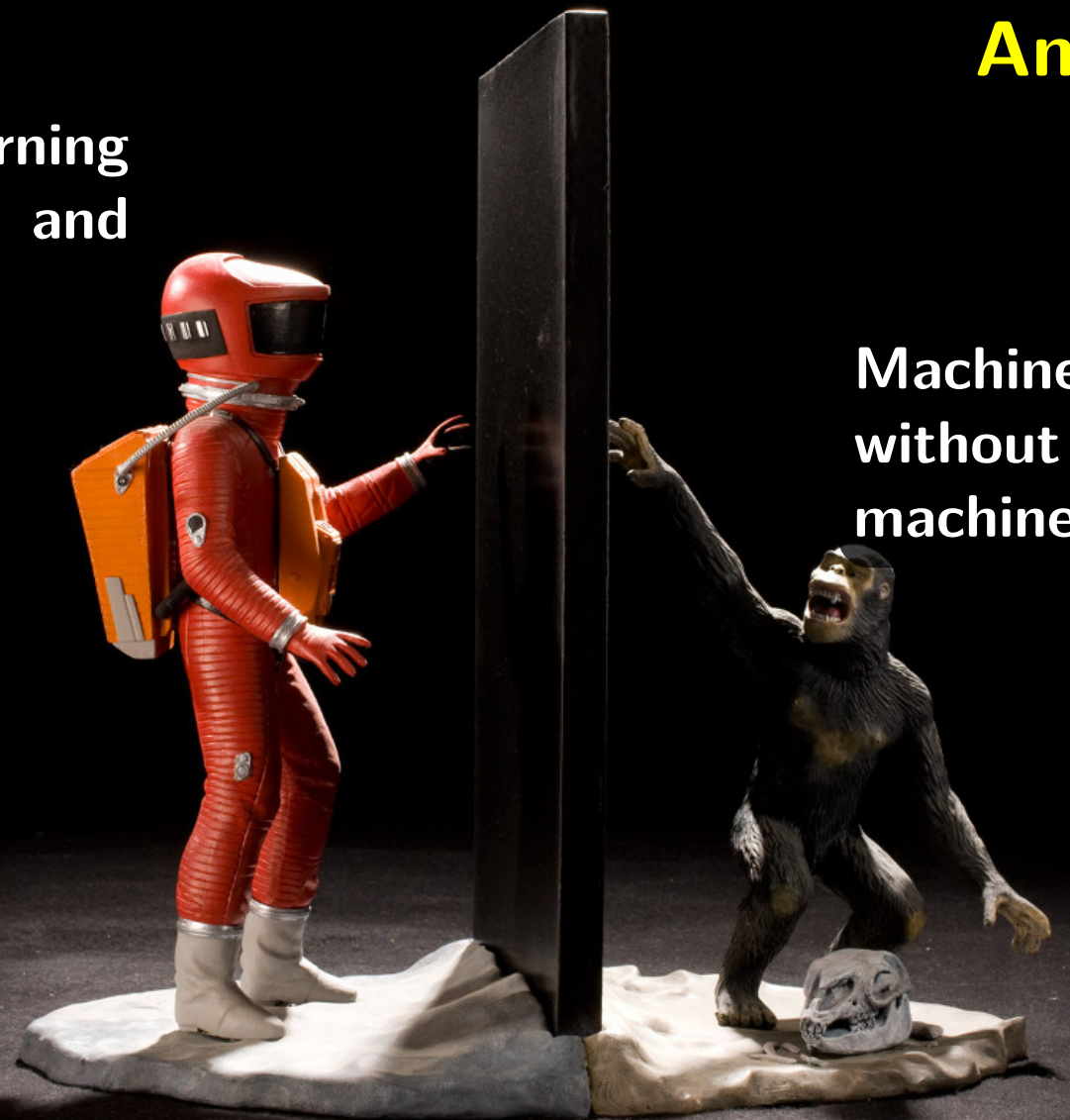
An enabler

1 Scikit-learn: the vision

An enabler

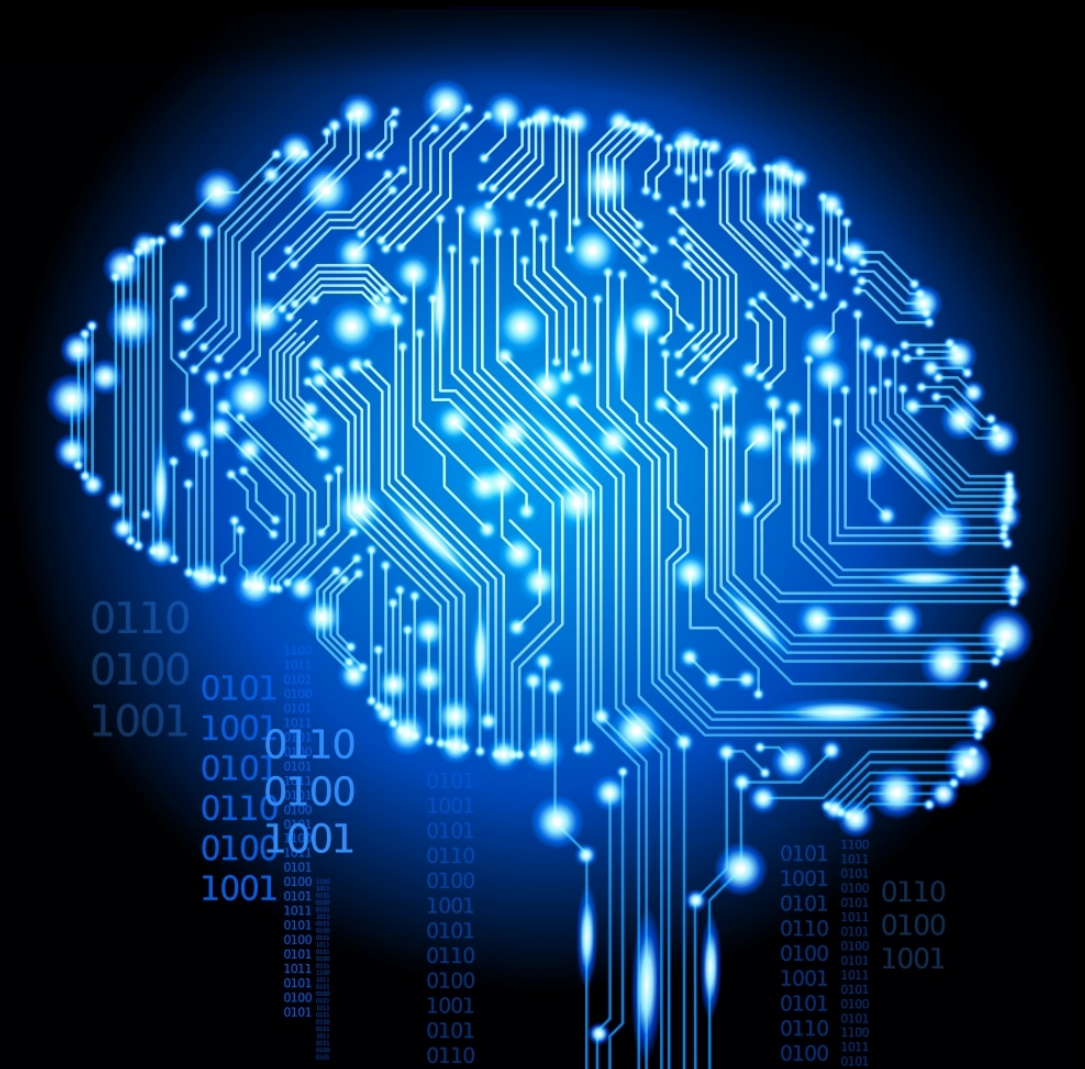
Machine learning for everybody and for everything

Machine learning without learning the machinery



Machine learning in a nutshell

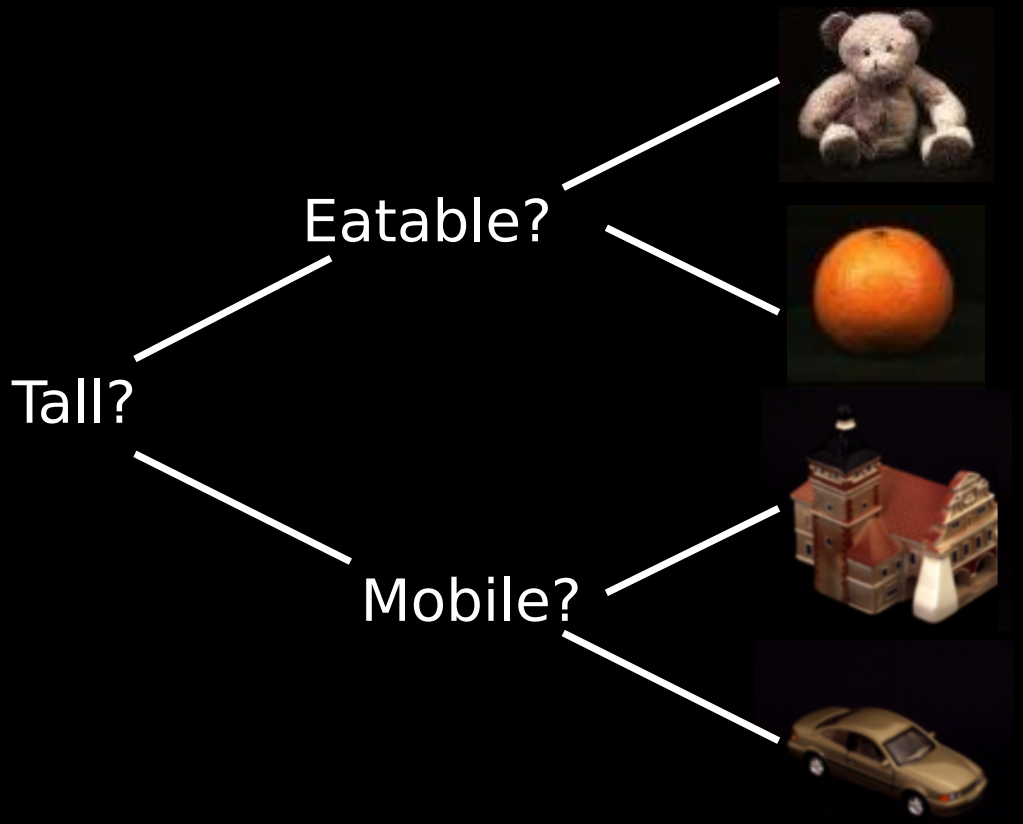
Machine learning is about making prediction from data



Artificial Intelligence

The 80s

- Building decision rules

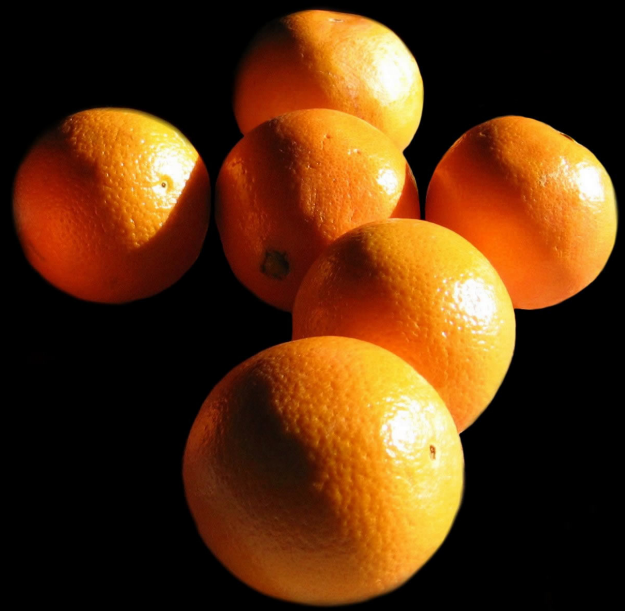


Artificial Intelligence The 80s

- Building decision rules

Machine learning The 90s

- Learn these from observations



1 Machine learning: a historical perspective

Artificial Intelligence The 80s

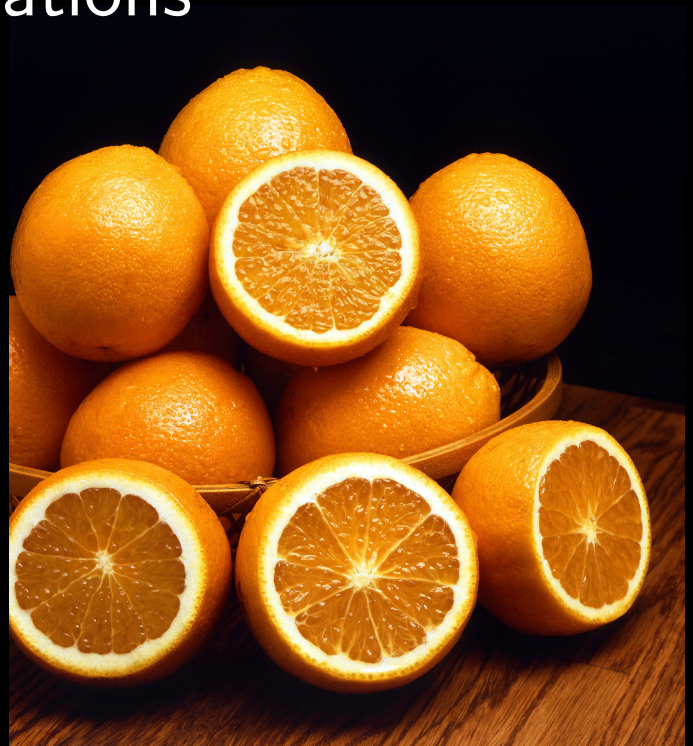
- Building decision rules

Machine learning The 90s

- Learn these from observations

Statistical learning 2000s

- Model the noise in the observations



1 Machine learning: a historical perspective

Artificial Intelligence The 80s

- Building decision rules

Machine learning The 90s

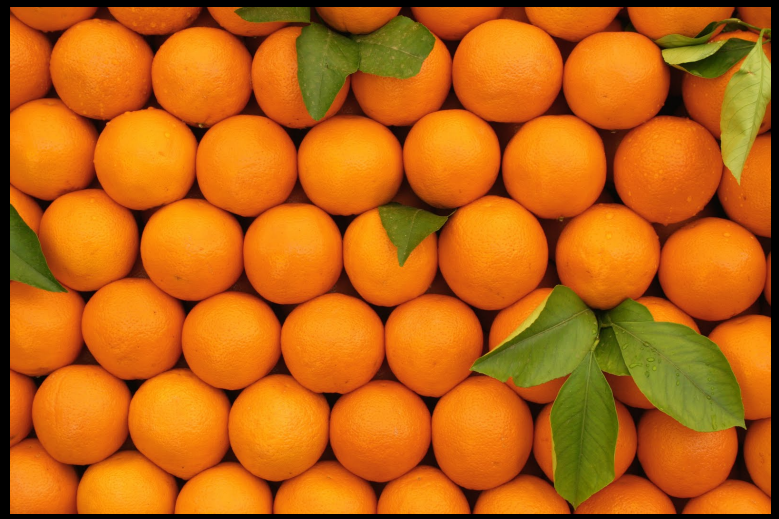
- Learn these from observations

Statistical learning 2000s

- Model the noise in the observations

Big data today

- Many observations, simple rules



1 Machine learning: a historical perspective

Artificial Intelligence The 80s

- Building decision rules

Machine learning The 90s

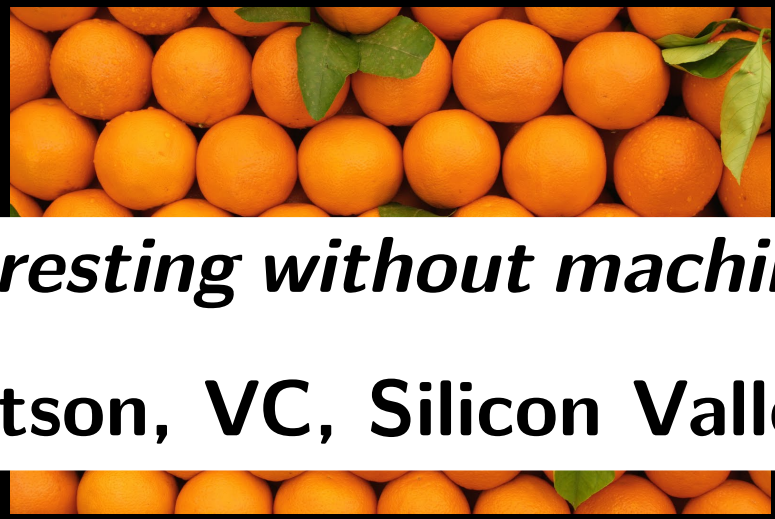
- Learn these from observations

Statistical learning 2000s

- Model the noise in the observations

Big data today

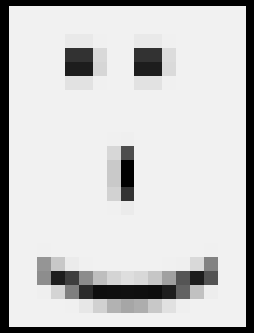
- Many observations, simple rules



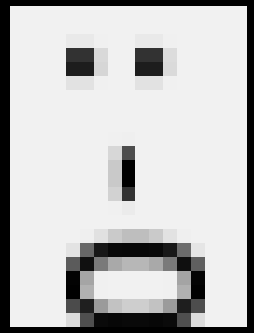
"Big data isn't actually interesting without machine learning"

Steve Jurvetson, VC, Silicon Valley

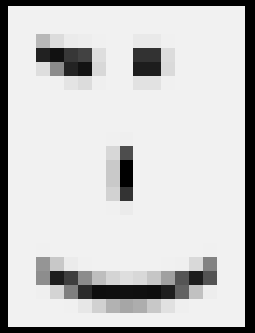
Face recognition



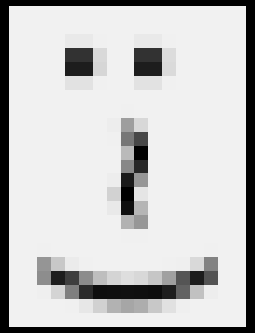
Andrew



Bill

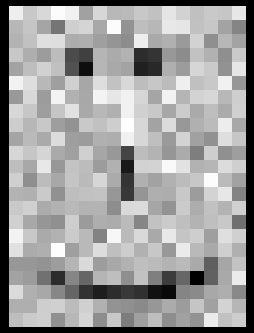


Charles

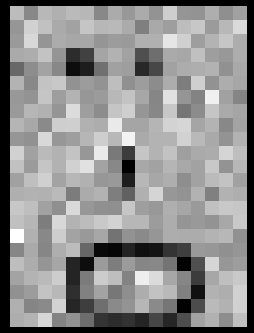


Dave

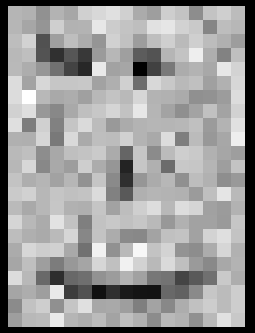
Face recognition



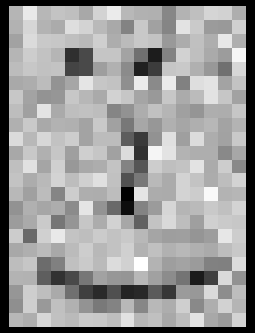
Andrew



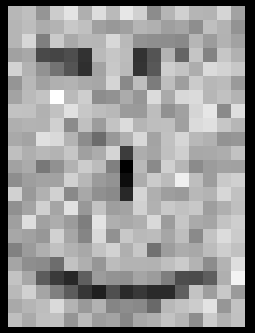
Bill



Charles



Dave



?

A simple method:

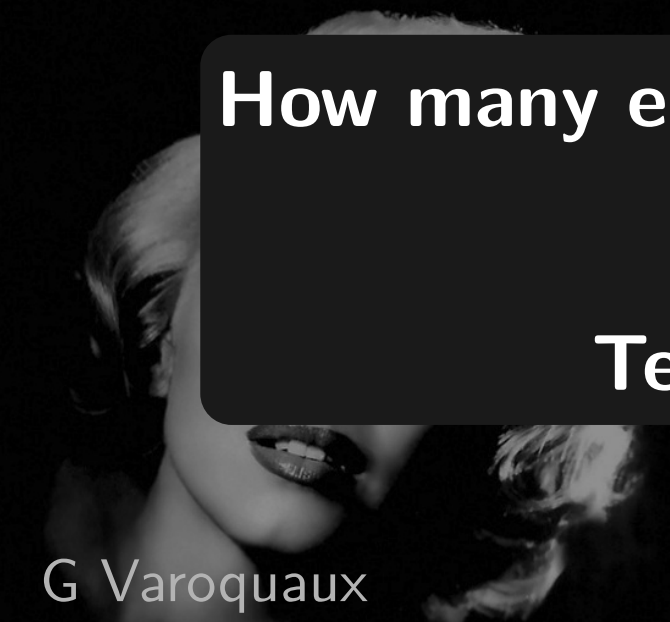
1. Store all the known (noisy) images and the names that go with them.
2. From a new (noisy) images, find the image that is most similar.

"Nearest neighbor" method

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

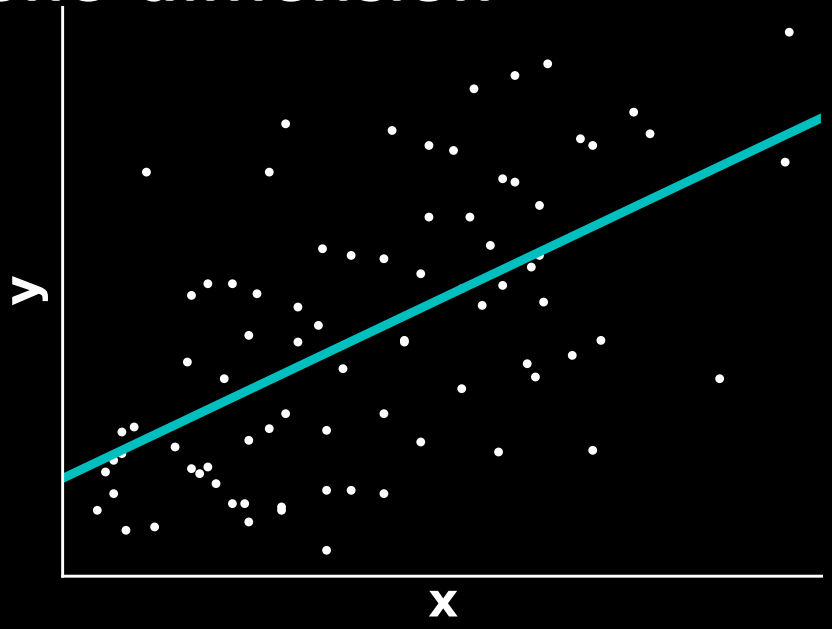


How many errors on already-known images?

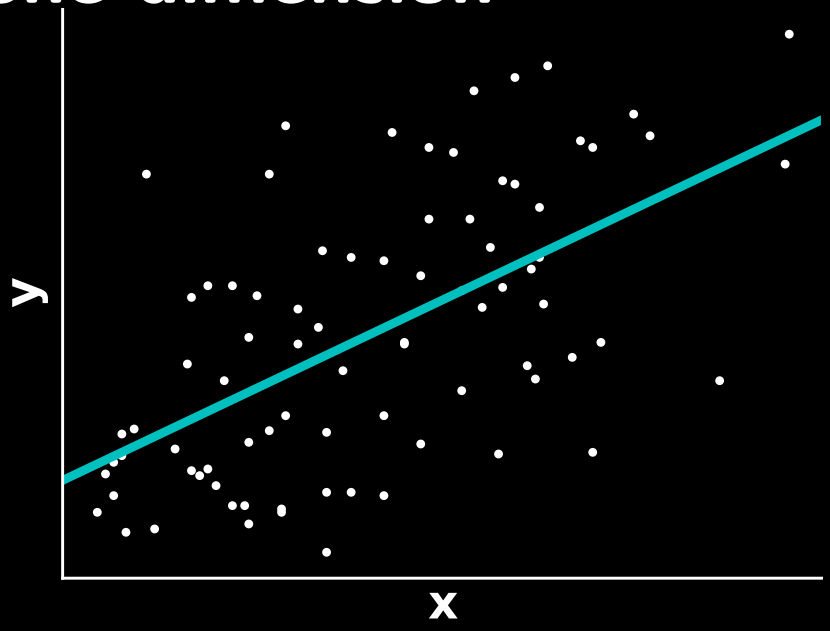
... 0: no erreurs

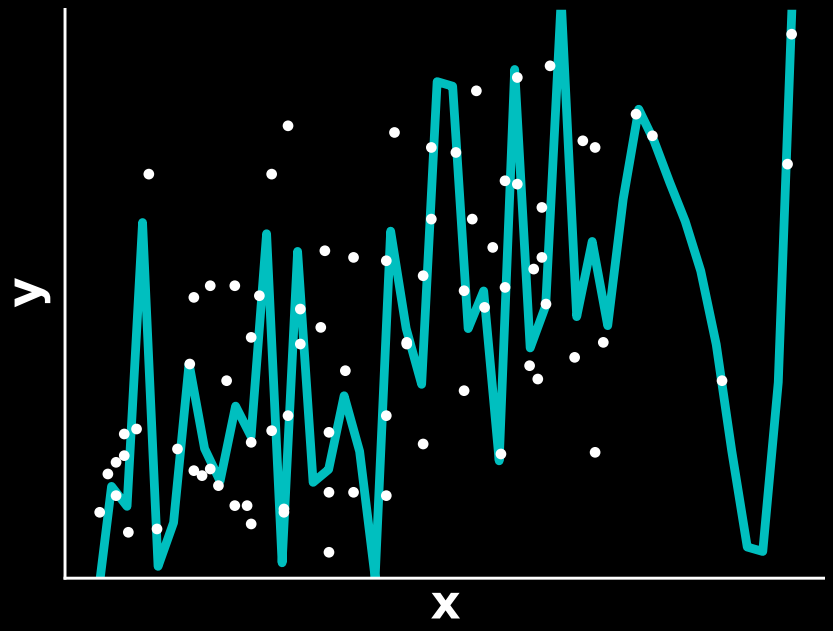
Test data $\neq$ Train data

A single descriptor: one dimension



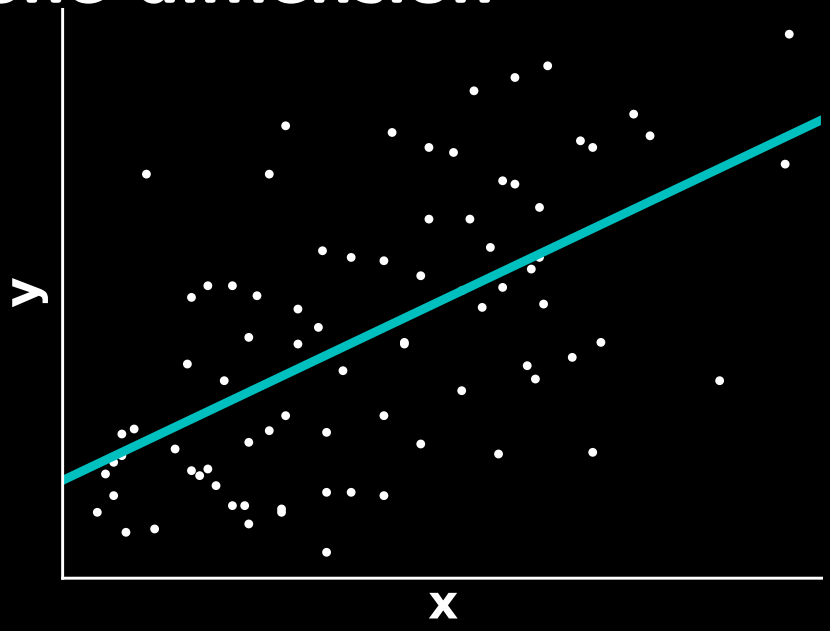
A single descriptor: one dimension

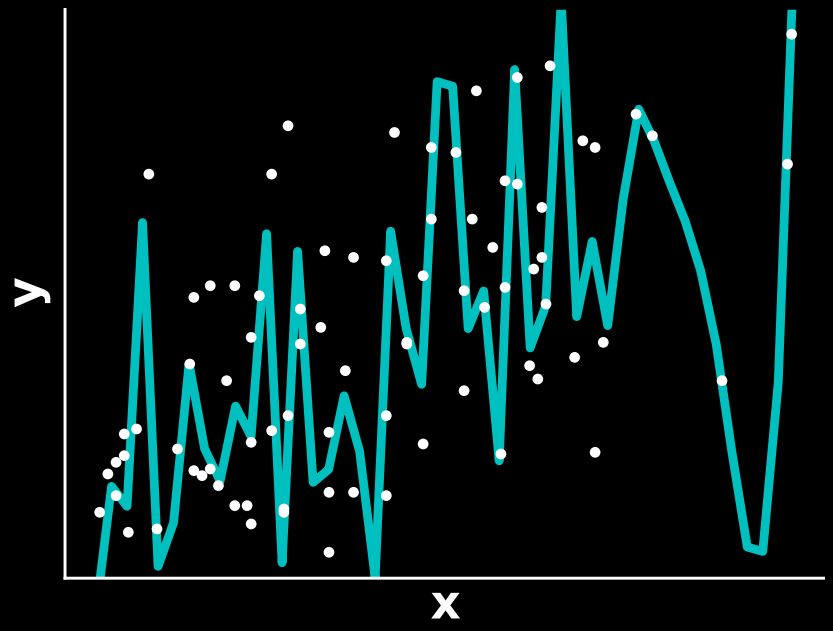




Which model to prefer?

A single descriptor: one dimension

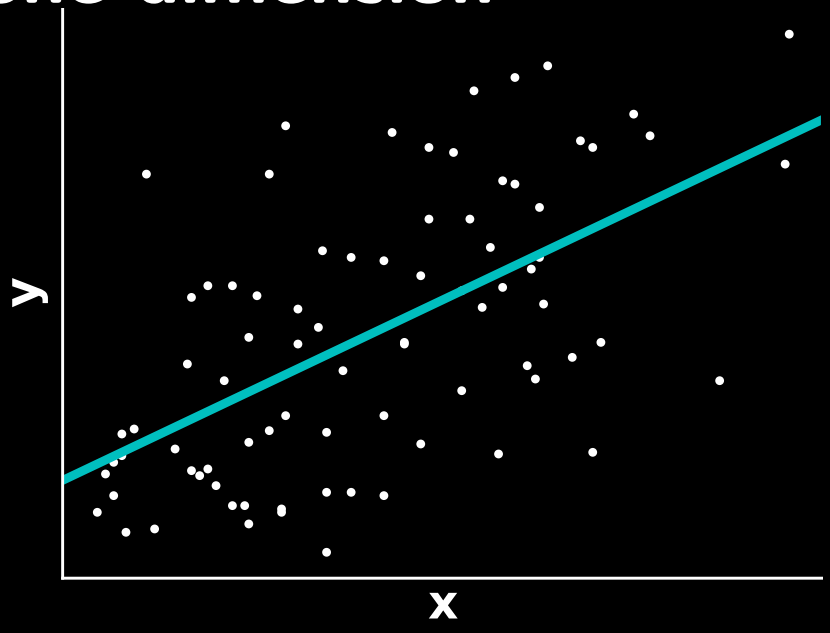


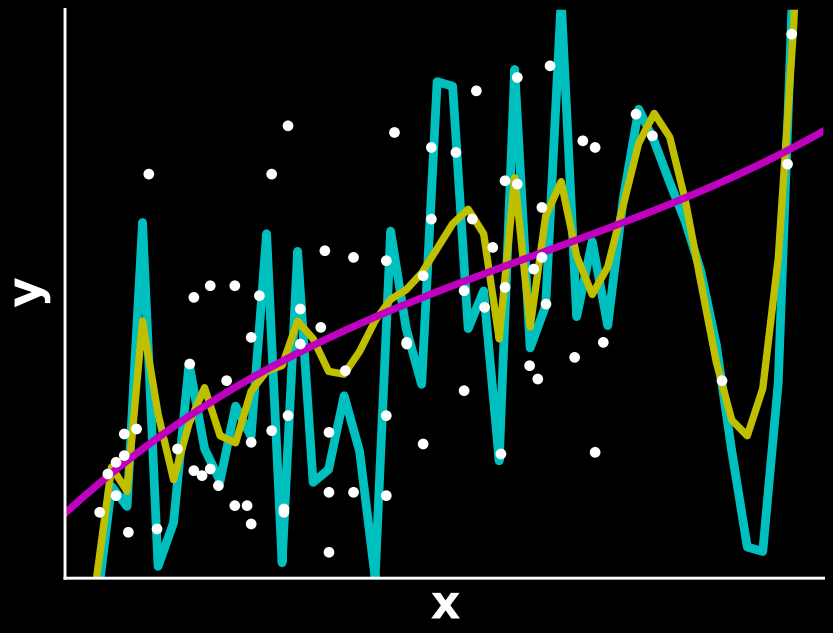


Problem of *"over-fitting"*

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

A single descriptor: one dimension





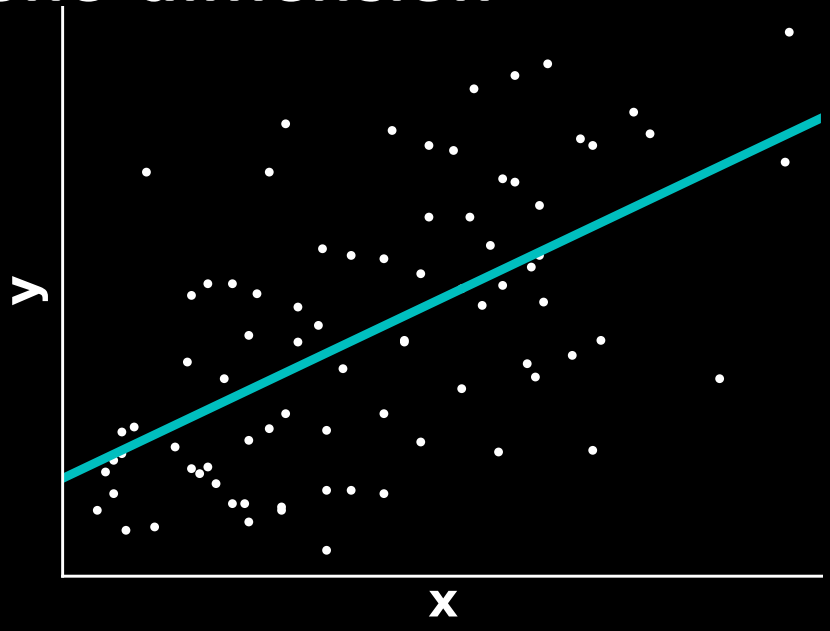
Prefer simple models

= concept of *"regularization"*

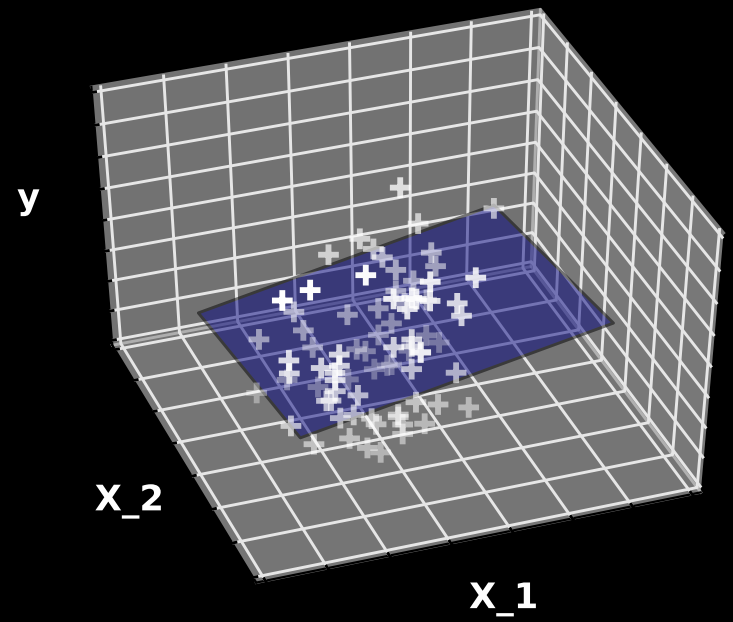
Balance the number of parameters to learn with the amount of data

1 Machine learning in a nutshell: regression

A single descriptor: one dimension



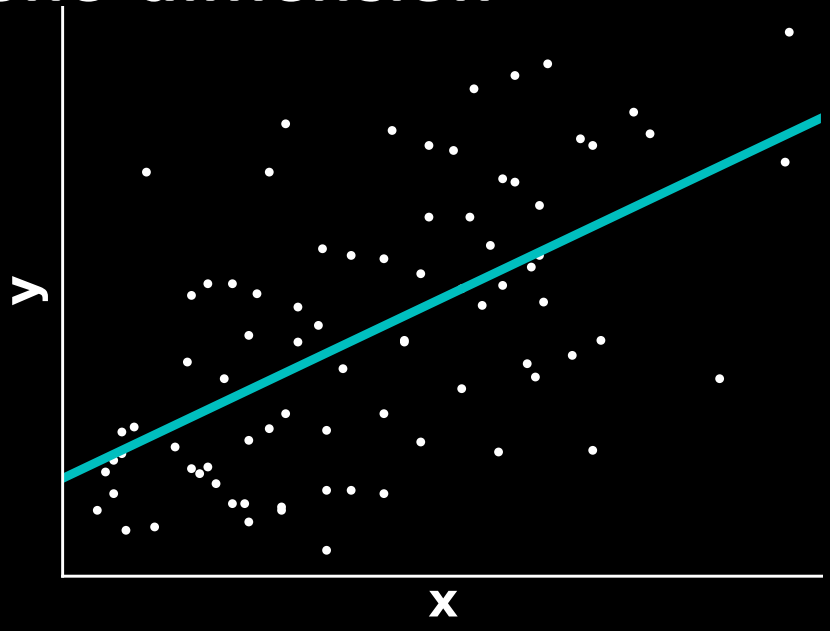
Two descriptors: 2 dimensions



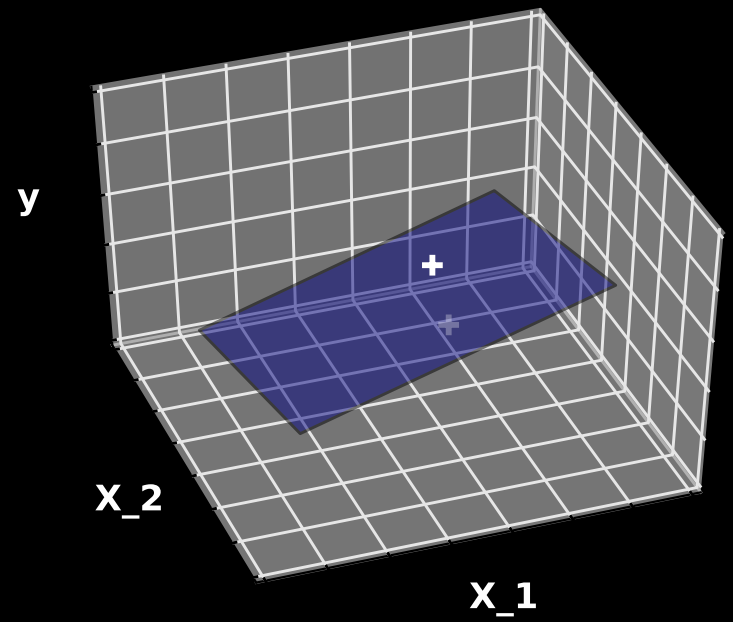
More parameters

1 Machine learning in a nutshell: regression

A single descriptor: one dimension



Two descriptors: 2 dimensions

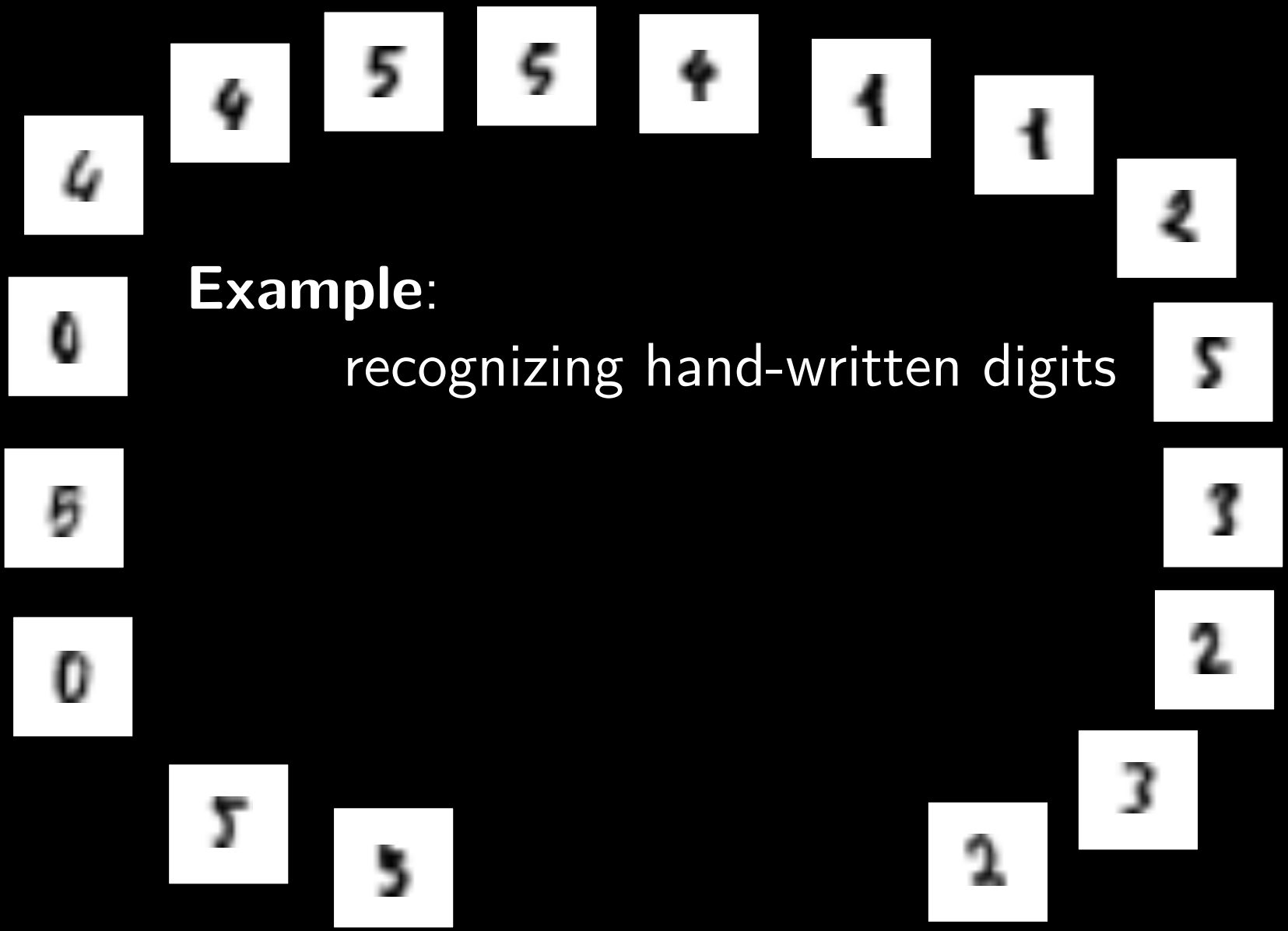


More parameters

$\Rightarrow$ need more data

"curse of dimensionality"

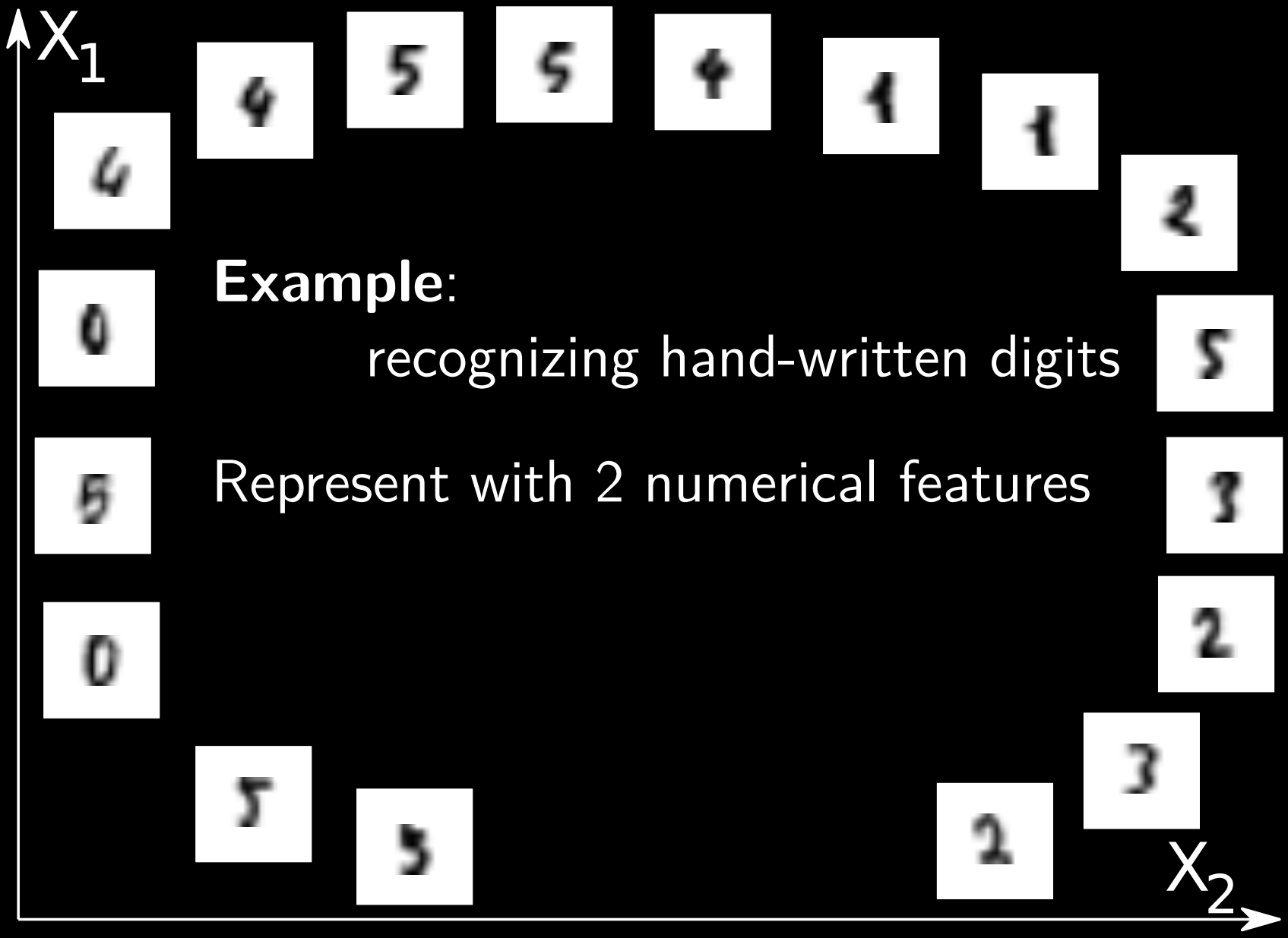
1 Machine learning in a nutshell: classification



Example:

recognizing hand-written digits

1 Machine learning in a nutshell: classification

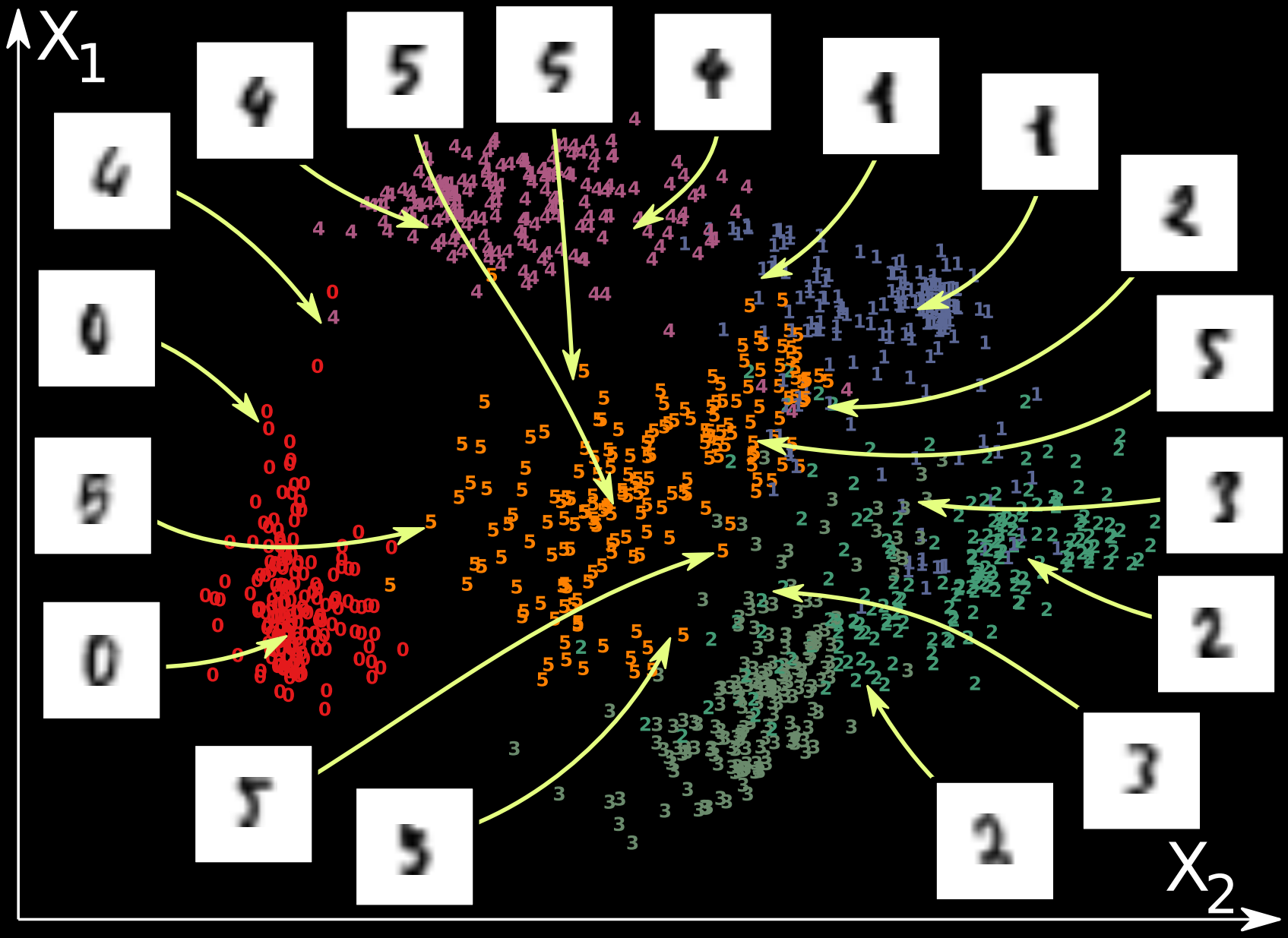


Example:

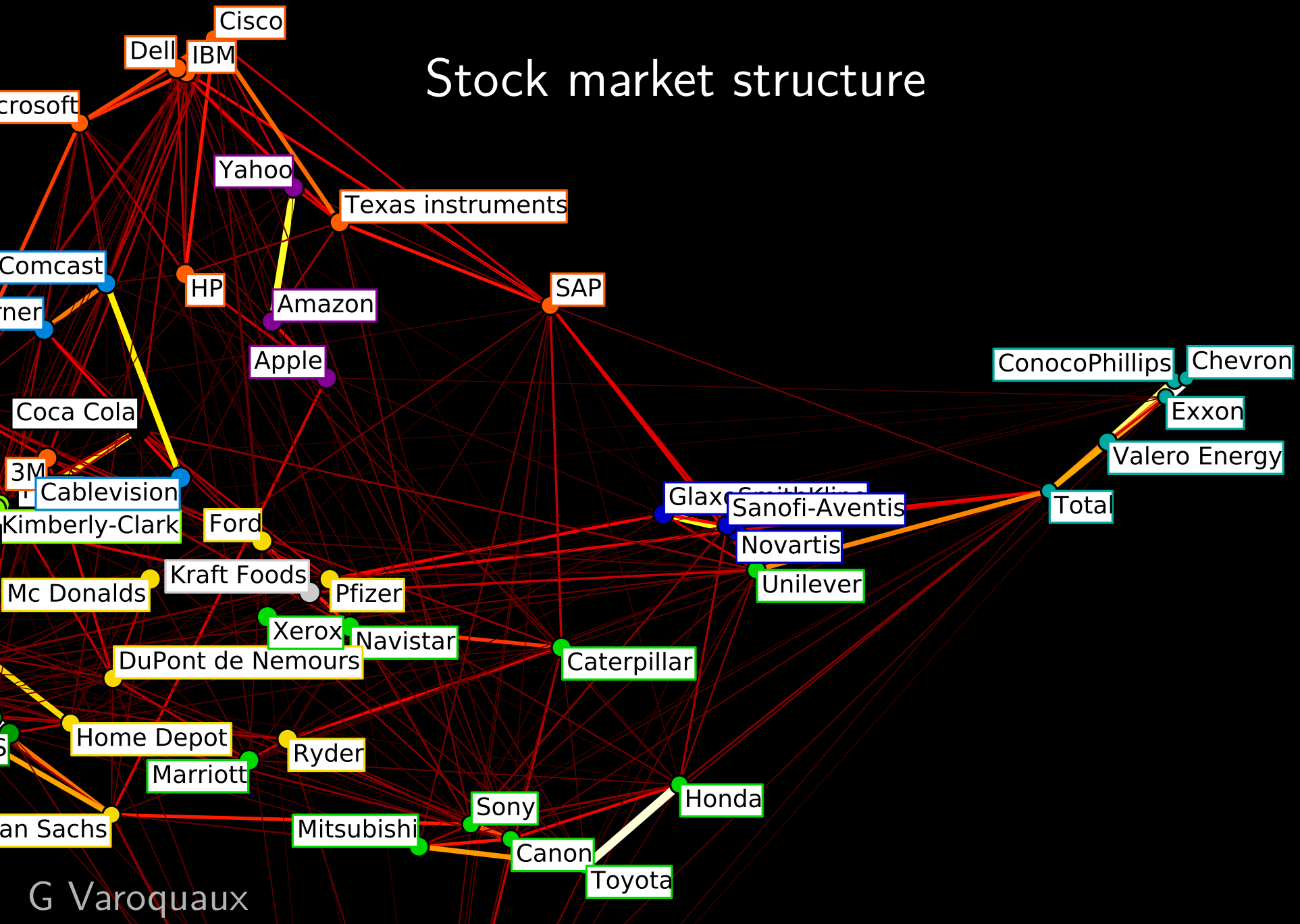
recognizing hand-written digits

Represent with 2 numerical features

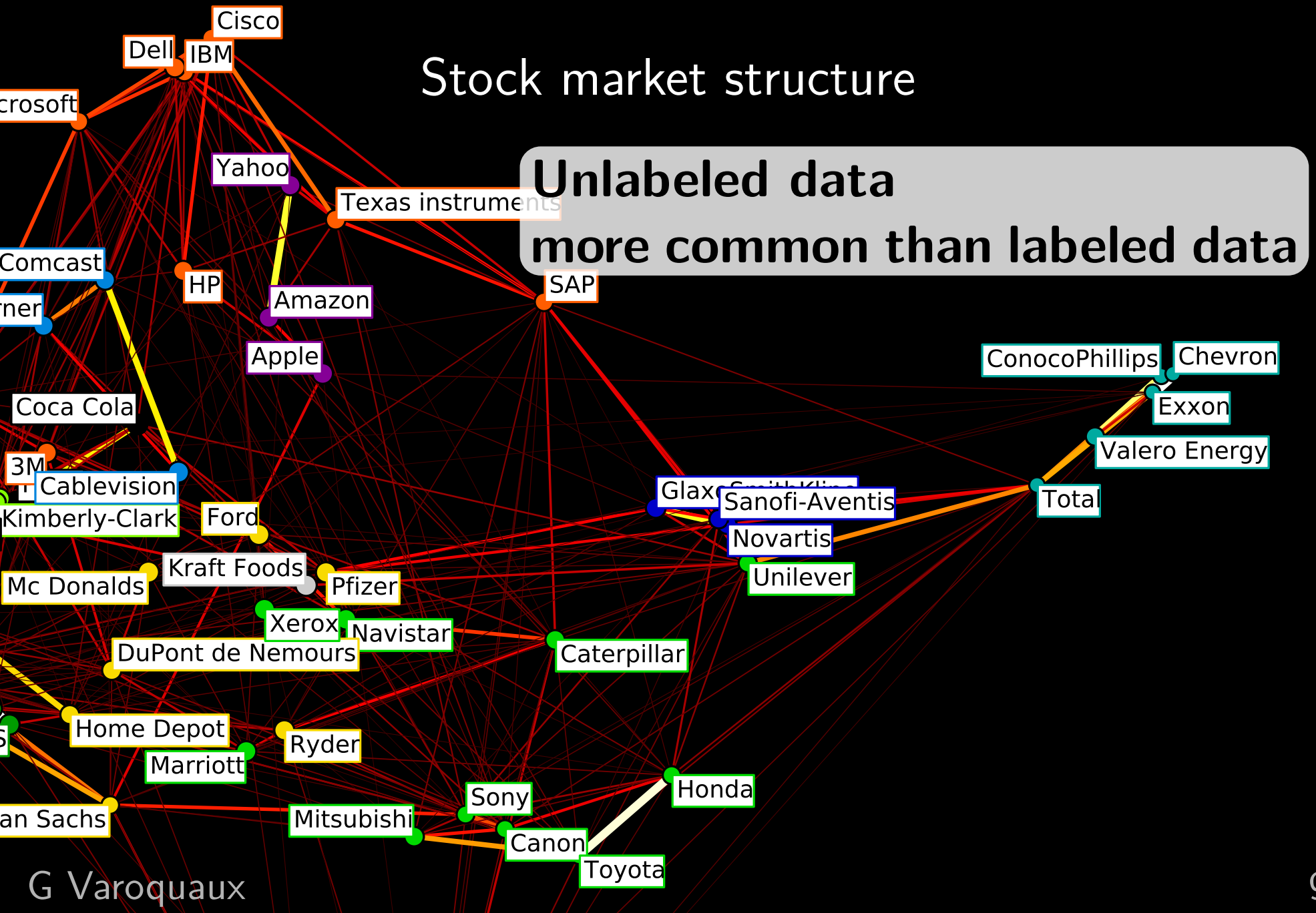
1 Machine learning in a nutshell: classification



1 Machine learning in a nutshell: unsupervised



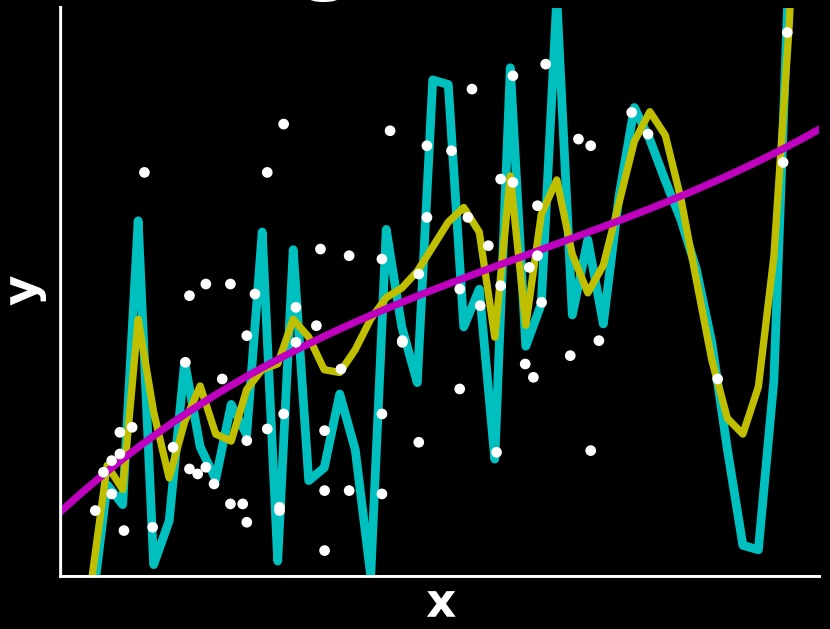
1 Machine learning in a nutshell: unsupervised



Machine learning

Mathematics and algorithms for fitting predictive models

Regression



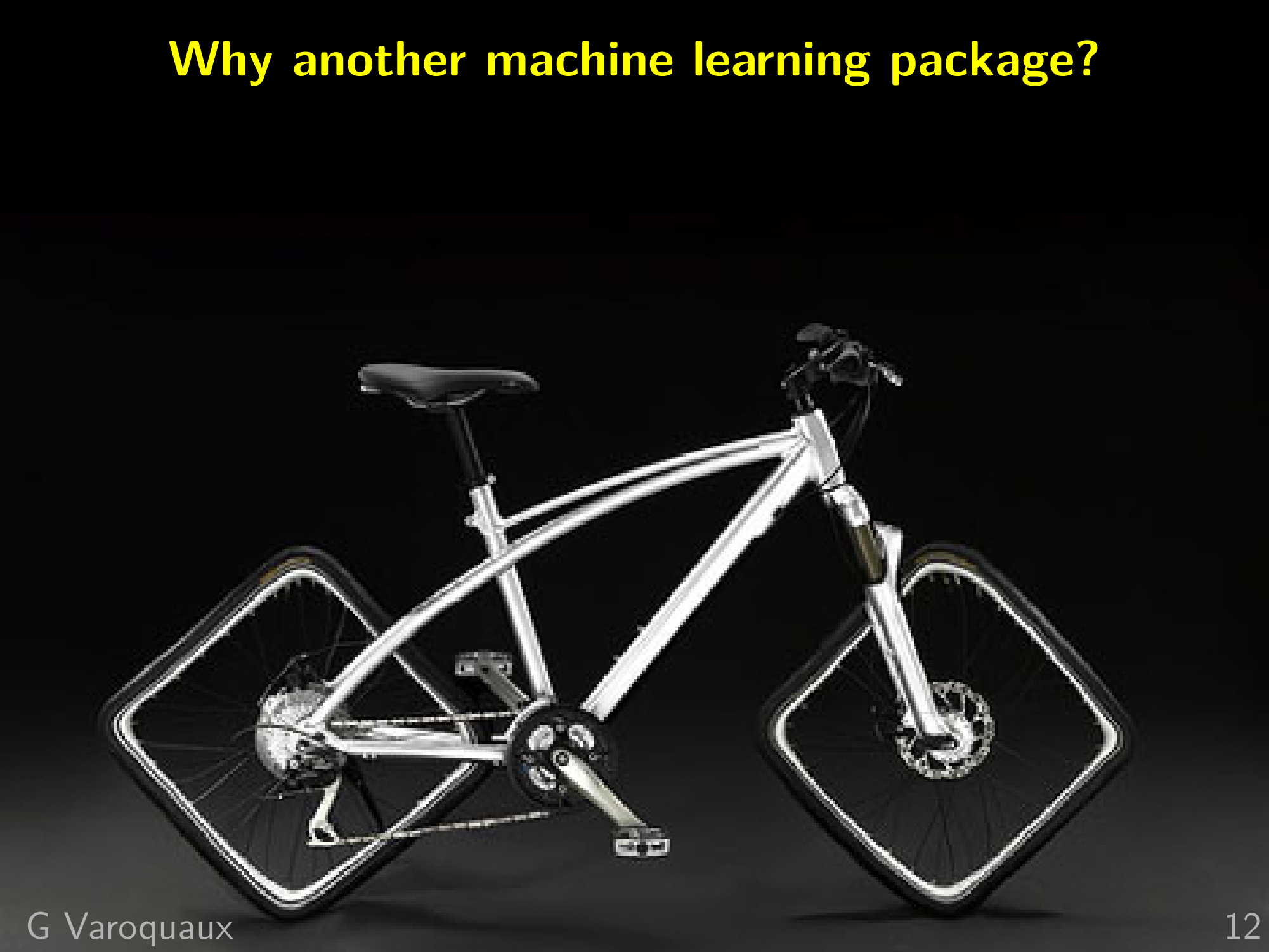
Classification

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?



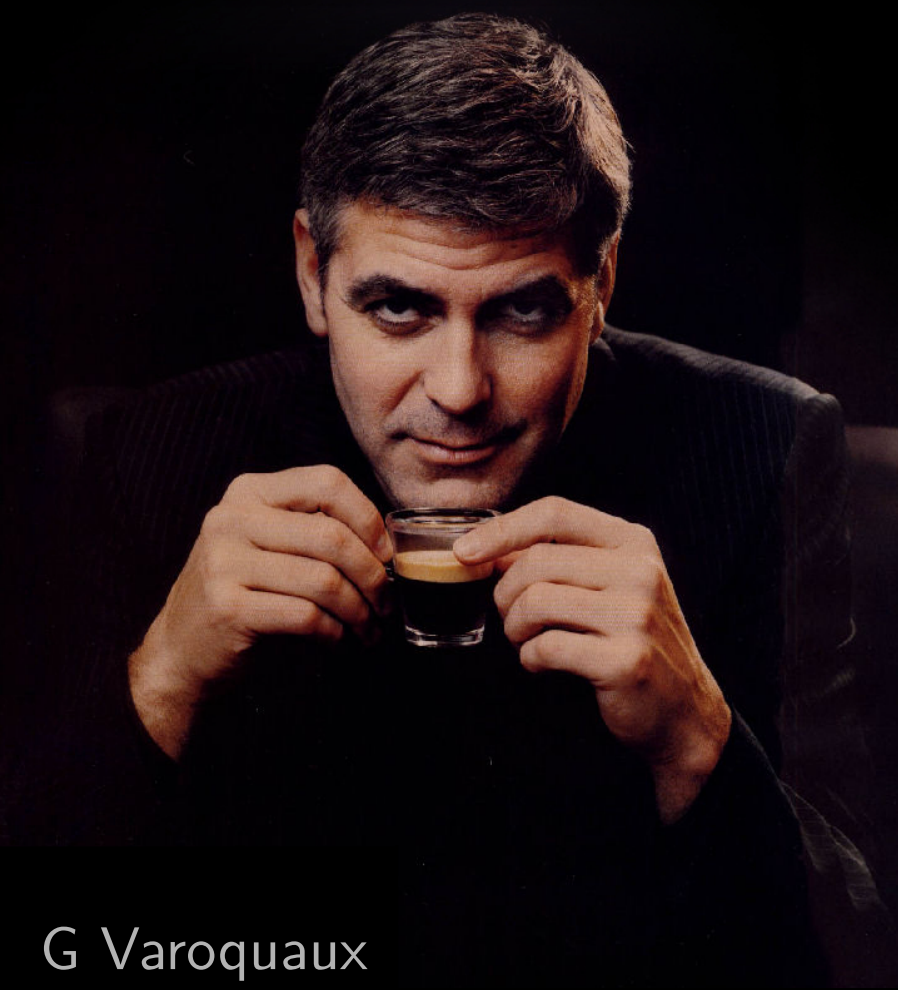
Real statisticians use R

- 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

1 scikit-learn vision

Machine learning for all

No specific application domain

No requirements in machine learning

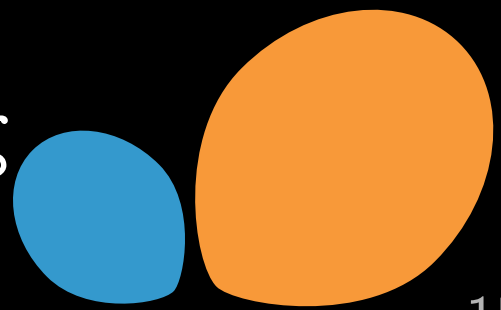
High-quality Pythonic software library

Interfaces designed for users

Community-driven development

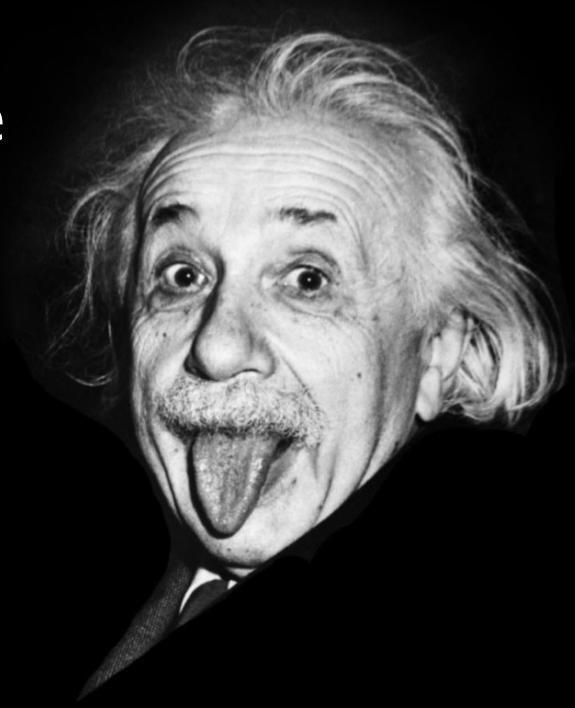
BSD licensed, very diverse contributors

`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

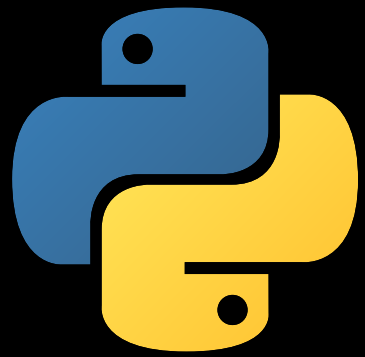
2 Scikit-learn: the tool

A Python library for machine learning

©Theodore W. Gray

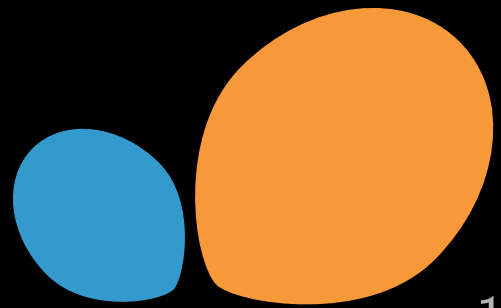
A library, not a program

- 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)
```



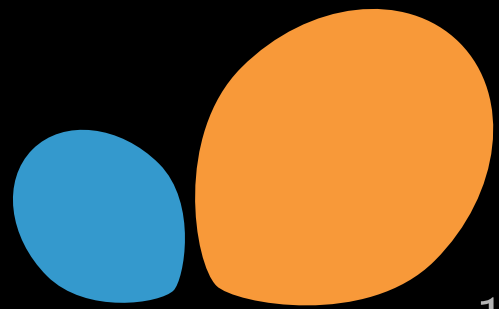
2 API: specifying a model

A central concept: **the estimator**

- Instanciated 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)
```

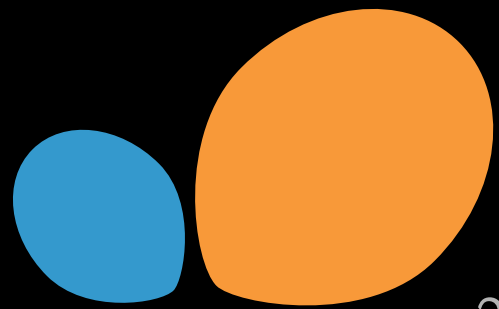
with:

- 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

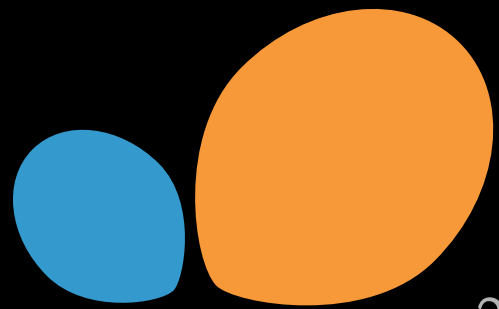
```
Y_test = estimator.predict(X_test)
```

- Transforming: dimension reduction, filter

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

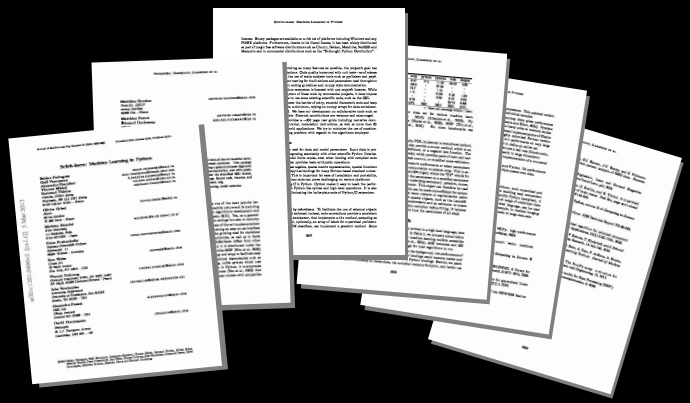
- Test score, density estimation

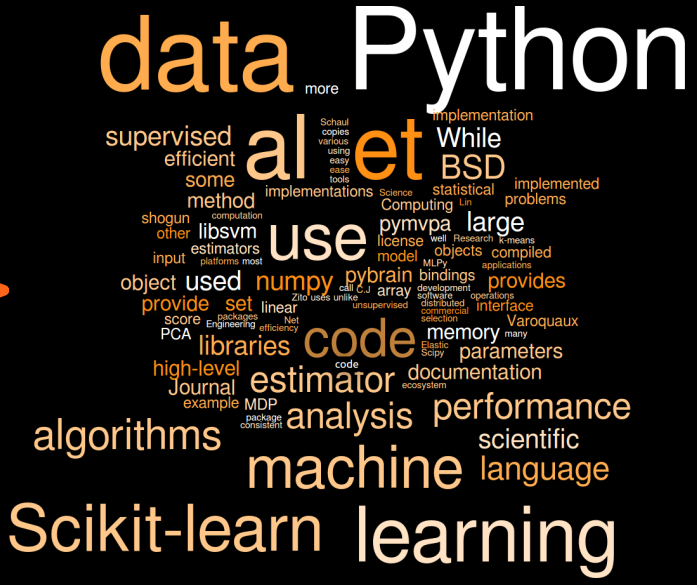
```
test_score = estimator.score(X_test)
```



From raw data to a sample matrix X

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





From raw data to a sample matrix X

- For text data: counting word occurences
  - 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

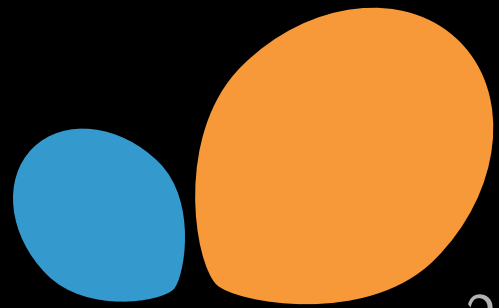
- Decision trees (Random-Forest, Boosted Tree)
- Linear models
- SVM

Unsupervised Learning

- Clustering
- Dictionary learning
- Outlier detection

Model selection

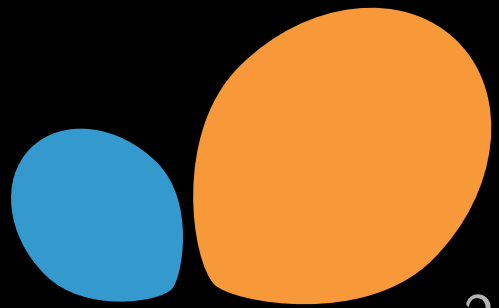
- Built in cross-validation
- Parameter optimization



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

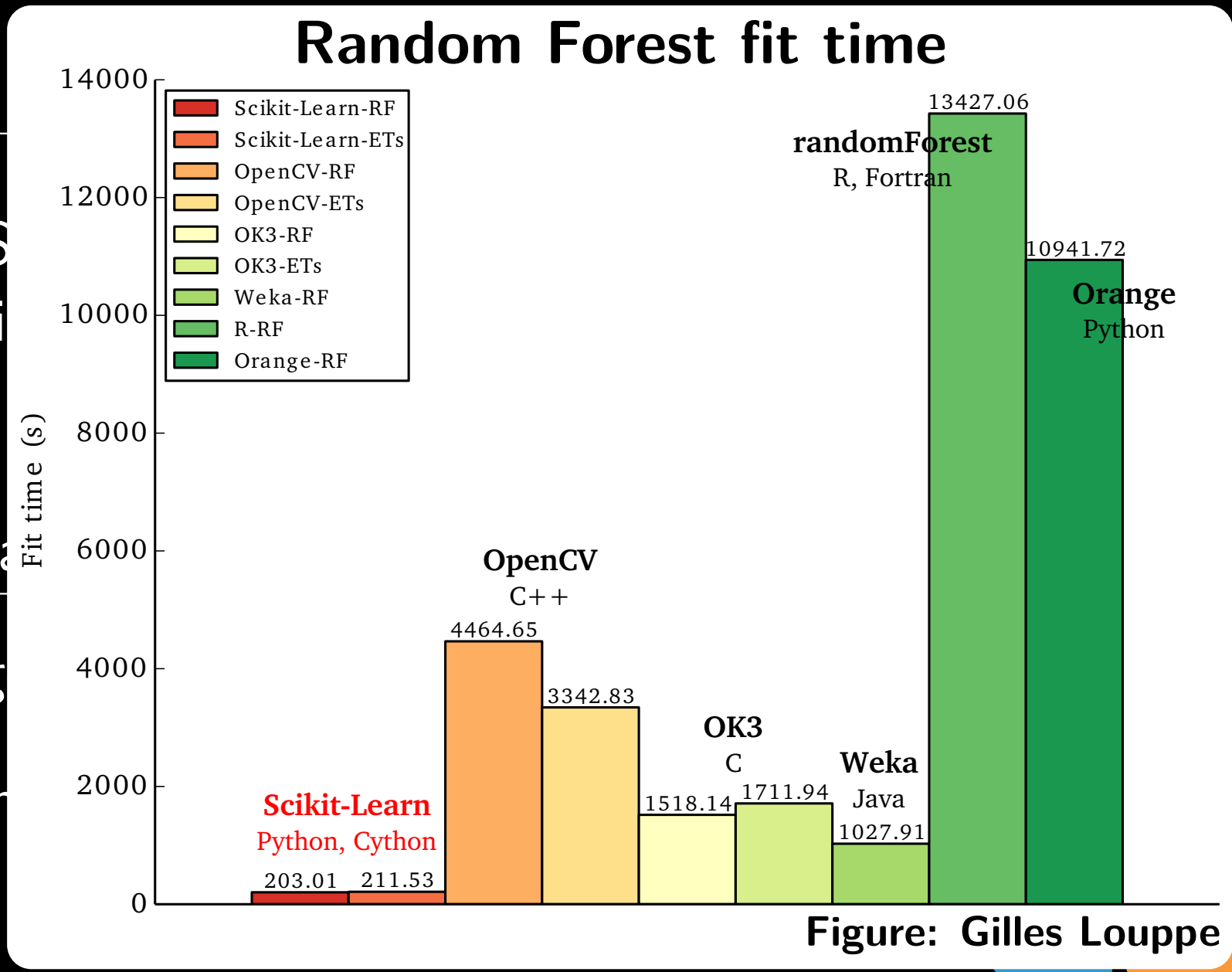


Figure: Gilles Louppe

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":

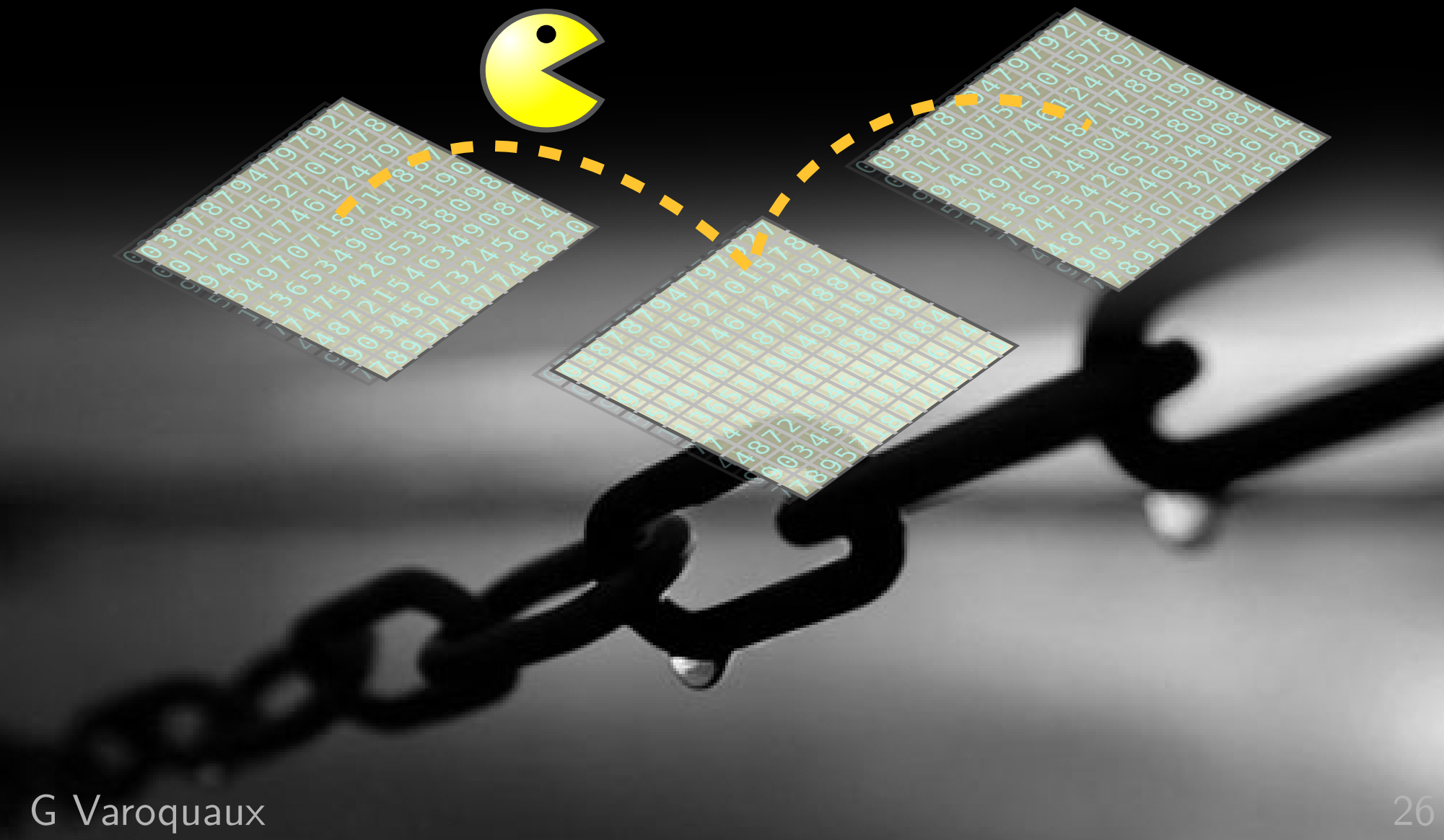
- Petabytes...
- Distributed storage
- Computing cluster

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

```
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

$\Rightarrow$ 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

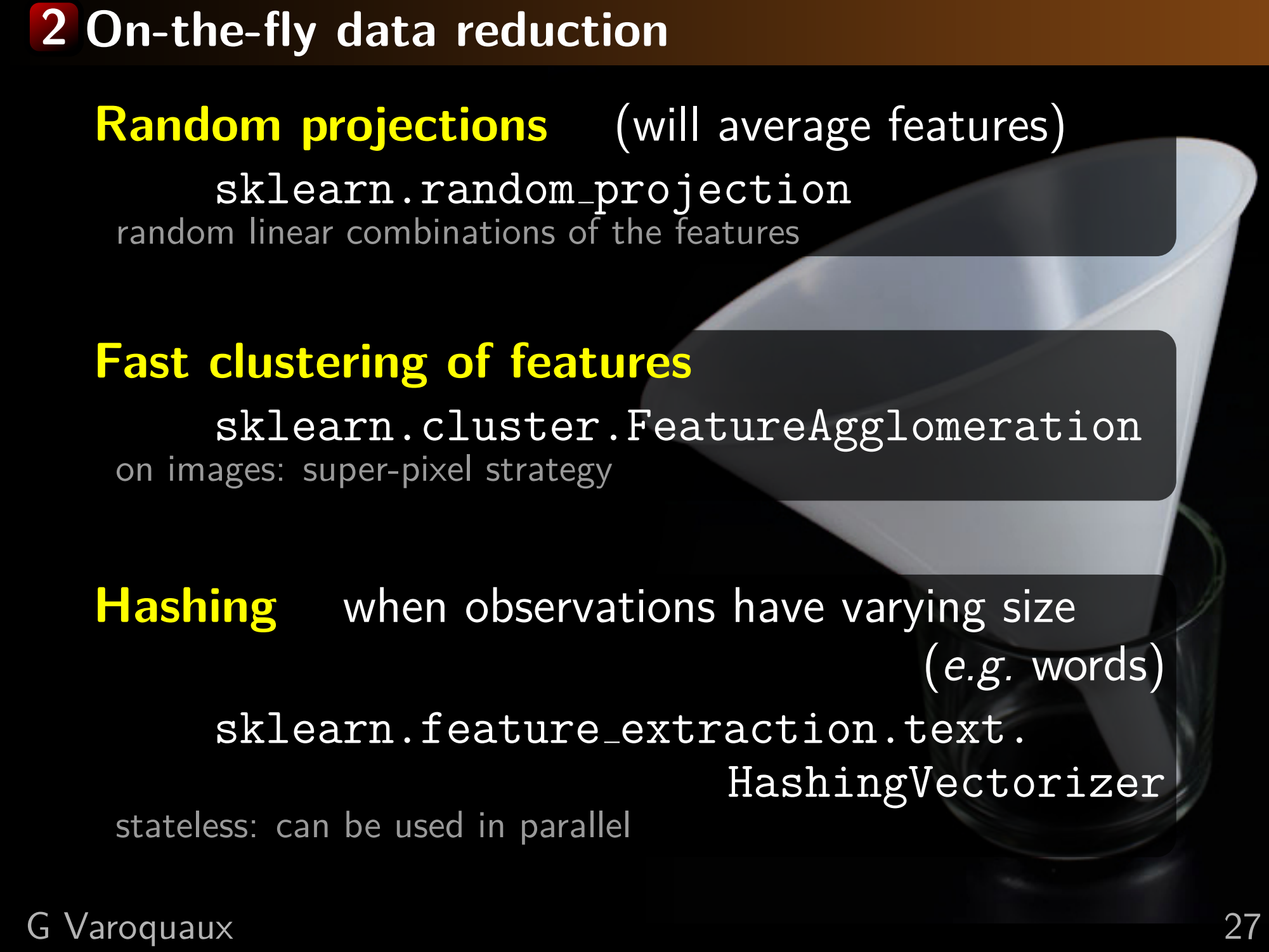
`sklearn.cluster.FeatureAgglomeration`

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

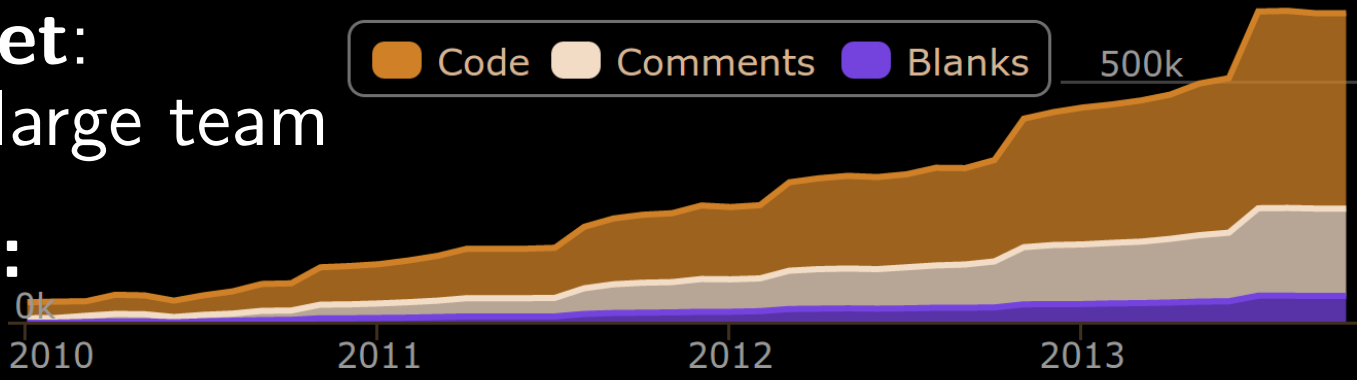


3 Scikit-learn: the project

3 Community-based development in scikit-learn

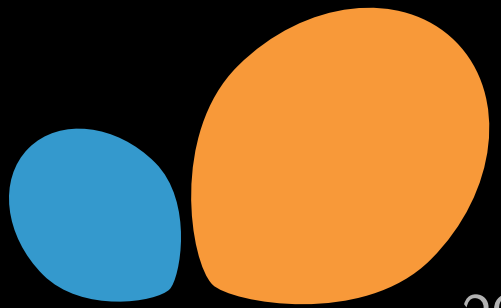
Huge feature set:
benefits of a large team

Project growth:



- More than 200 contributors
- $\sim$ 12 core contributors
- 1 full-time INRIA programmer from the start

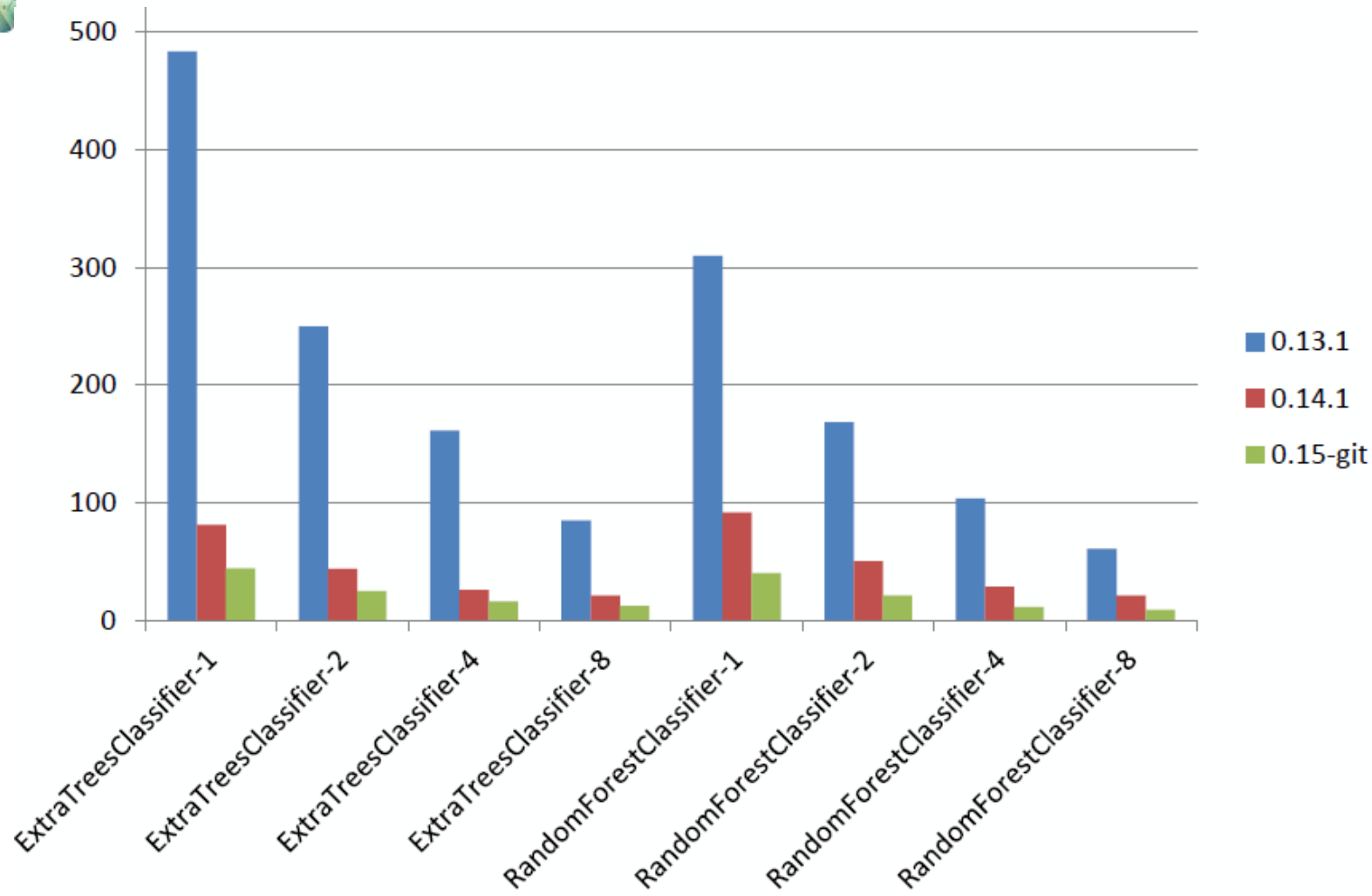
Estimated cost of development: **$ 6 millions**
COCOMO model,
`http://www.ohloh.net/p/scikit-learn`



3 Many eyes makes code fast

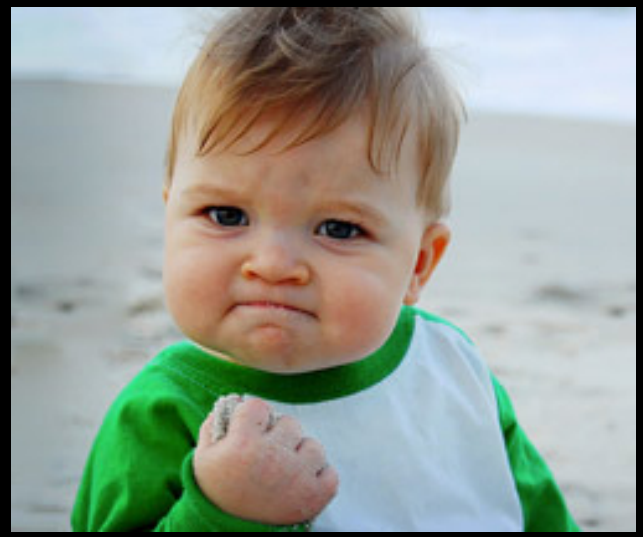
Gilles Louppe @glouppe · Feb 18

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



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

3 6 steps to a community-driven project

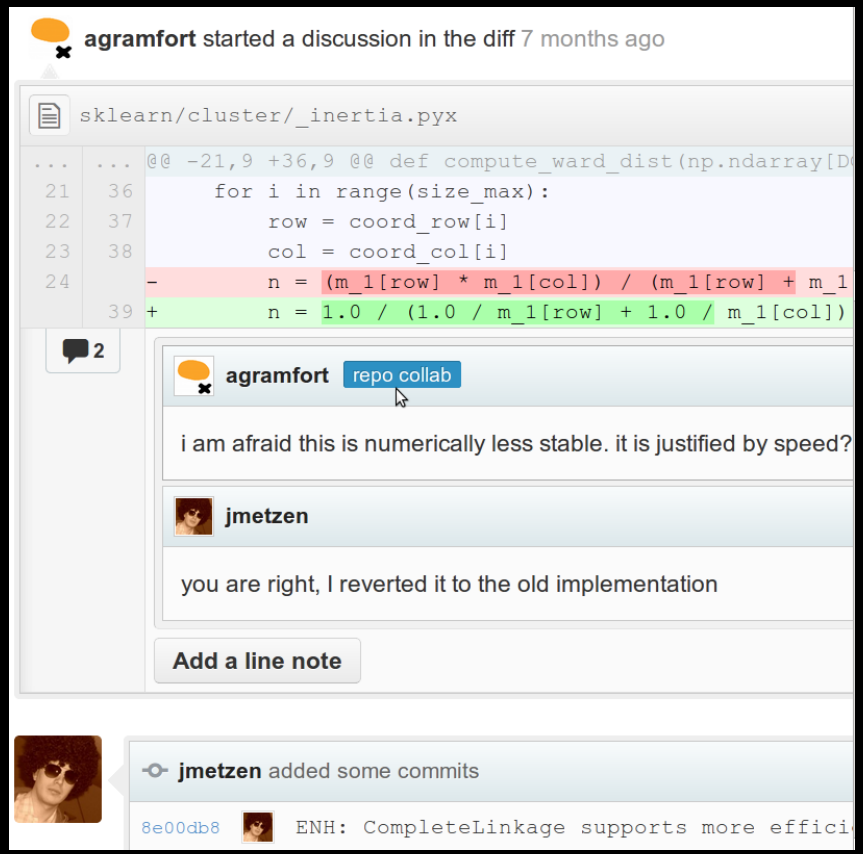


1. Focus on **quality**
2. Build great **docs and examples**
3. Use **github**
4. Limit the technicality of your codebase
5. Releasing and packaging matter
6. Focus on your contributors, give them credit, decision power

`http://www.slideshare.net/GaelVaroquaux/scikit-learn-dveloppement-communautaire`

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 meaningfull?
  - Are the numerics stable?
  - Could it be faster?



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

$\Rightarrow$ Hard to fund, less excitement

They need citation, in papers & on corporate web pages

3 The tragedy of the commons

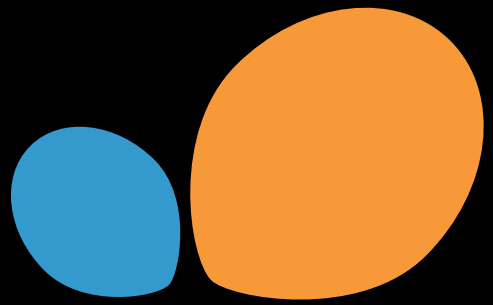
Make it work, make it right, make it boring

Core projects (boring) taken for granted

$\Rightarrow$ Hard to fund, less excitement

They need citation, in papers & on corporate web pages

Scikit-learn



The vision

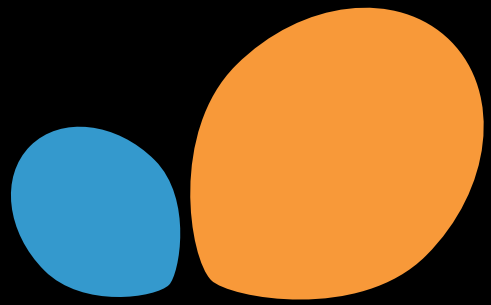
Machine learning as a means not an end

Versatile library: the “right” level of abstraction

Close to research, but seeking different tradeoffs

@GaelVaroquaux

Scikit-learn



The vision

Machine learning as a means not an end

The tool

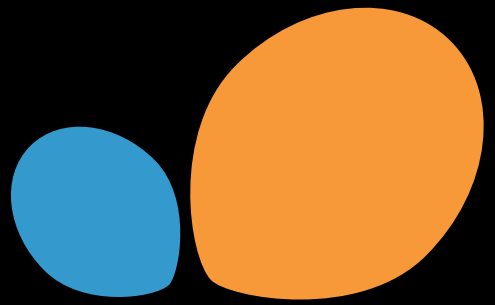
Simple API uniform across learners

Numpy matrices as data containers

Reasonnably fast

@GaelVaroquaux

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!

@GaelVaroquaux