Python for brain mining: (neuro)science with state of the art machine learning and data visualization

Gaël Varoquaux

1. Data-driven science “Brain mining”
2. Data mining in Python Mayavi, scikit-learn, joblib
1 Brain mining
Learning models of brain function
1 Imaging neuroscience

Brain images → Models of function

Cognitive tasks
1 Imaging neuroscience

Brain images

Models of function

Data-driven science

\[ i \hbar \frac{\partial}{\partial t} \psi = H \psi \]

Gaël Varoquaux
Brain functional data

Rich data
50,000 voxels per frame
Complex underlying dynamics

Few observations
\[ \sim 100 \]

Drawing scientific conclusions?
Ill-posed statistical problem
1 Brain functional data

Rich data
50,000 voxels per frame
Complex underlying dynamics
Few observations
\sim 100

Modern complex system studies:
from strong hypotheses to rich data

Drawing scientific conclusions?
Ill-posed statistical problem

Gaël Varoquaux
Statistics: the curse of dimensionality

$y$ function of $x_1$

More fit parameters? $\Rightarrow$ need exponentially more data

y function of 50,000 voxels?

Expert knowledge (pick the right ones)

Machine learning

Gaël Varoquaux
Statistics: the curse of dimensionality

More fit parameters?
⇒ need exponentially more data
Statistics: the curse of dimensionality

More fit parameters?  ⇒ need exponentially more data

y function of $x_1$ and $x_2$

y function of 50,000 voxels?

Expert knowledge (pick the right ones)

Machine learning
Brain reading

Predict from brain images the object viewed

Correlation analysis

Gaël Varoquaux
Brain reading

Predict from brain images the object viewed

**Inverse problem**

Observations → Spatial code

**Correlation analysis**

Inject prior: regularize

Sparse regression = compressive sensing
Brain reading

Predict from brain images the object viewed

Inverse problem

Observations → Spatial code

Inject prior: regularize

Extract brain regions

Total variation regression

Correlation analysis

[Michel, Trans Med Imag 2011]
Brain reading

Predict from brain images the object viewed

Inverse problem

Inject prior: regularize

Observations

Spatial code

Correlation analysis

Inject prior: regularize

Cast the problem in a prediction task: supervised learning.

Prediction is a model-selection metric

[Michel, Trans Med Imag 2011]

Gaël Varoquaux
On-going/spontaneous activity

95% of the activity is unrelated to task
Learning regions from spontaneous activity

Multi-subject dictionary learning

Sparsity + spatial continuity + spatial variability

⇒ Individual maps + functional regions atlas

[Varoquaux, Inf Proc Med Imag 2011]
Graphical models: interactions between regions

Estimate covariance structure

Many parameters to learn

Regularize: conditional independence

= sparsity on inverse covariance

[Varoquaux NIPS 2010]
Graphical models: interactions between regions

Estimate covariance structure

Many parameters to learn

Regularize: conditional independence = sparsity on inverse covariance

Find structure via a density estimation: unsupervised learning.

Model selection: likelihood of new data

[Varoquaux NIPS 2010]
My data-science software stack

Mayavi, scikit-learn, joblib
Mayavi: 3D data visualization

Requirements
- large 3D data
- interactive visualization
- easy scripting

Solution
- VTK: C++ data visualization
- UI (traits)
  + pylab-inspired API

Black-box solutions don’t yield new intuitions

Limitations
- hard to install
- clunky & complex
  C++ leaking through
- 3D visualization doesn’t pay in academia

Tragedy of the commons or niche product?
Vision

- Address non-machine-learning experts
- Simplify but don’t dumb down
- Performance: be state of the art
- Ease of installation
Technical choices

- Prefer Python or Cython, focus on readability
- Documentation and examples are paramount
- Little object-oriented design. Opt for simplicity
- Prefer algorithms to framework
- Code quality: consistency and testing
2 scikit-learn: statistical learning

API

- Inputs are numpy arrays
- Learn a model from the data:
  \[
  \text{estimator.fit}(X_{\text{train}}, Y_{\text{train}})
  \]
- Predict using learned model
  \[
  \text{estimator.predict}(X_{\text{test}})
  \]
- Test goodness of fit
  \[
  \text{estimator.score}(X_{\text{test}}, y_{\text{test}})
  \]
- Apply change of representation
  \[
  \text{estimator.transform}(X, y)
  \]
### Computational performance

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

- Algorithms rather than low-level optimization
  - convex optimization + machine learning
- Avoid memory copies

Gaël Varoquaux
Community

- 35 contributors since 2008, 103 github forks
- 25 contributors in latest release (3 months span)

Why this success?

- Trendy topic?
- Low barrier of entry
- Friendly and very skilled mailing list
- Credit to people
We keep recomputing the same things
Nested loops with overlapping sub-problems
Varying parameters

Standard solution: pipelines

Challenges
Dependencies modeling
Parameter tracking
**joblib: Python functions on steroids**

---

**Philosophy**

- Simple: don’t change your code
- Minimal: no dependencies
- Performant: big data
- Robust: never fail

---

**joblib’s solution = lazy recomputation:**

- Take an MD5 hash of function arguments,
- Store outputs to disk
Lazy recomputing

```python
>>> from joblib import Memory
>>> mem = Memory(cachedir='./tmp/joblib')
>>> import numpy as np
>>> a = np.vander(np.arange(3))
>>> square = mem.cache(np.square)
>>> b = square(a)

[Memory] Calling square...
square(array([[0, 0, 1],
               [1, 1, 1],
               [4, 2, 1]]))

>>> c = square(a)
>>> # No recomputation
```

Gaël Varoquaux
Conclusion

- Data-driven science will need machine learning because of the curse of dimensionality

- Scikit-learn and joblib: focus on large-data performance and easy of use

Cannot develop software and science separately