Speaker: Nicolas Usunier
- Francis Maes,
University Pierre et Marie Curie
Computer Science Laboratory of Paris 6 (LIP6)
Statistical Machine Learning Team
The toolbox
  Overview
  Energy Based Models
  Architecture of Nieme

Data formats
  Input File Formats
  Ranking
  Composite Vectors

Conclusion
  Additional Information
  Who should use Nieme?
  Questions
**Overview**

- **Statistical Machine Learning Toolbox:**
  - **Supervised Learning (SL):** Classification, Regression and Ranking
  - **Decision Processes (DP):** Supervised learning of policies, Reinforcement learning of policies

SL is **mature**. DP is still **work-in-progress**, not detailed here.

- Unified view of learning machines: **Energy Based Models**
- Both **batch** and **online** learning.
- Emphasis on **large-scale learning**, especially with **large number of features**.
Energy Based Models

Learning Machine =

Architecture  How to compute outputs given inputs ?
+ Loss  How to penalize parameters given an example ?
+ Regularizers  How to enforce simple models ?
+ Learner  How to learn the parameters ?

Speaker: Nicolas Usunier - Francis Maes, University Pierre et Marie Curie Computer Science Laboratory of Paris 6 (LIP6) Statistical Machine Learning Team

Nieme 0.8 Classification, regression and ranking
Energy Based Models

- **Architecture:** Linear, Multi-class linear, Transfer, Compose
- **Loss:** Hinge, Perceptron, Log-binomial, Exponential, Squared error, Absolute error
- **Regularizers:** L1-norm, L2-norm
- **Learners:** Stochastic Descent, Mini-batchs, Pegasos SVM, LBFGS, OWLQN, RProp

<table>
<thead>
<tr>
<th>Model</th>
<th>Architecture</th>
<th>Loss</th>
<th>Regularizers</th>
<th>Learner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron</td>
<td>linear</td>
<td>perceptron</td>
<td>none</td>
<td>stochastic descent</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>linear</td>
<td>log-binomial</td>
<td>none</td>
<td>batch quasi-newton</td>
</tr>
<tr>
<td>Pegasos linear SVM</td>
<td>linear</td>
<td>hinge loss</td>
<td>L2</td>
<td>pegasos learner</td>
</tr>
<tr>
<td>Multilayer perceptron</td>
<td>linear  (\circ) transfer  (\circ) linear</td>
<td>perceptron</td>
<td>none</td>
<td>stochastic descent</td>
</tr>
<tr>
<td>L1-maxent classifier</td>
<td>multi-class linear</td>
<td>log-binomial</td>
<td>L1</td>
<td>batch quasi-newton</td>
</tr>
<tr>
<td>Pegasos multi-class SVM</td>
<td>multi-class linear</td>
<td>hinge loss</td>
<td>L2</td>
<td>pegasos learner</td>
</tr>
<tr>
<td>Least-square regression</td>
<td>linear</td>
<td>squared loss</td>
<td>none</td>
<td>batch quasi-newton</td>
</tr>
<tr>
<td>Custom</td>
<td>linear  (\circ) transfer</td>
<td>absolute loss</td>
<td>L1 + L2</td>
<td>batch rprop</td>
</tr>
<tr>
<td>Many others</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Speaker: Nicolas Usunier - Francis Maes, University Pierre et Marie Curie

Nieme 0.8 Classification, regression and ranking
Architectural Overview

**Core**
16,500 lines of C++ code
+ 300 classes
Object-Oriented Design
(Design Patterns...)

**Plugins**
C++
Dynamic Libraries

**Scripting Interfaces**
18 classes
+/- 200 methods

**User Interface**
"Nieme Explorer"
Results visualizer

Compiles under Windows, Linux and MacOS X

---

**Speaker:** Nicolas Usunier - Francis Maes, University Pierre et Marie Curie

**Program:** Nieme 0.8 - Classification, regression and ranking
Interface

- **SWIG**: connects programs written in C and C++ with a variety of high-level programming languages.
- Currently: wrappers for Python, Java and C++

```python
train = InstanceSet.loadClassificationData("example.data")
machine = EnergyBasedMachine.createMaxentClassifier()
machine.train(train)
machine.save("example.model")
```

```java
InstanceSet train = InstanceSet.loadClassificationData("example.data");
LearningMachine machine = EnergyBasedMachine.createMaxentClassifier();
machine.train(train);
machine.save("example.model");
```

```cpp
InstanceSet train = InstanceSet::loadClassificationData("example.data");
LearningMachine machine = EnergyBasedMachine::createMaxentClassifier();
machine.train(train);
machine.save("example.model");
```

Maximum Entropy = **Multiclass Linear** architecture, **Log-binomial** loss, **L2** regularizer, **L-BFGS** learner

**Speaker**: Nicolas Usunier - Francis Maes, University Pierre et Marie Curie Computer Science Laboratory of Paris 6 (LIP6) Statistical Machine Learning Team
Visualizer tool: vectors, tables, models and more.

Parameter Vectors

Experimental Results

Plugins: Images, Decision Processes, ...

(work in progress)
Input File Formats

- Generalization of LibSVM’s format.
- Similar formats for Classification, Regression and Ranking.
- The whole feature set is never specified.

- Dense Vectors, 6 continuous features, 2 classes:

```
1 1:0.0526316 2:0.2 3:-0.456954 4:-0.428571 5:-0.821918 6:-1
2 1:0.0526316 2:-0.286997 3:-0.271523 4:-0.298701 5:-0.876712 6:-1
2 1:0.105263 2:-0.046087 3:-0.015389 4:-0.714286 5:-0.664384 6:-1
```

- Sparse Vectors, binary features, 3 classes:

```
yes 5:1 8:1 15:1 21:1 25:1 33:1 34:1 37:1 42:1 50:1 53:1 57:1 67:1 76:1 78:1
maybe 2:1 8:1 19:1 21:1 27:1 33:1 34:1 36:1 44:1 50:1 53:1 57:1 67:1 76:1 78:1
```

- Equivalently, short syntax:

```
yes 5 8 15 21 25 33 34 37 42 50 53 57 67 76 78 81 84 86
no 2 3 20 22 23 33 35 36 47 50 51 58 67 76 79 81 82 86
maybe 2 8 19 21 27 33 34 36 44 50 53 57 67 76 78 83 97
```

- Features do not need to be sorted and can use any alphanumeric identifier:

```
class1 afeature anotherfeature
class2 feat150 feat12 feat315
class3 501636 23543 2353262
class4 aaa AAA bbb BBB
```
A format for Ranking data (Work-in-progress)

- Ranking example = list of alternatives
- Alternative = a vector and a “cost-to-predict” value
- Encompasses instance-ranking and label-ranking, bipartite-ranking and generalized-ranking.
- Examples:

  # First example (bipartite)
  1 f4 f5 f6
  0 f1:0.7 f2:0.3 f3:1.0
  1 f6 f7:0.2 f8

  # Second example
  0 f10
  10 f4 f5
  1 f9
  0 f11

- Pro: Very simple; Cons: Vectors may be duplicated several times.
Composite Vectors

- An original data-structure for vectors.
- A vector is either flat (standard vectors) or composite.

→ Generic Architecture composition.
→ Easier visualization.
→ Sub-vector sharing.
→ Sub-linear dot-products.

Feature names separated by dots:

`params.firstLayer.hidden2.feature51`
Additional Information

- **Quick-start guide** for C++, Java, Python under Linux, Windows and MacOS X
- **Tutorials**
  1. Basic operations on learning machines.
  2. Synthetic data, Vectors and Cross-Validation.
  3. Tuning L1 regularization with cross-validation, using Tables.
- **Documentation** Full reference of the interface.
- **Unit Tests** 362 Unit-tests with Python unittest.
- Released under the **GPL license**.
Who should use Nieme?

Use Nieme’s interfaces and explorer if:

- You have many features.
- You have many examples.
- You want to compare the behavior of various architecture/losses/regularizer combinations.
- For large-scale sparse linear learning.
- You have a structure in your features (see Composite Vectors).

Extends Nieme’s core if:

- You want to experiment a new component: Architecture, Loss, Regularizer or Learner.
- You have a good knowledge of C++.

Do not use Nieme if:

- You want to use kernels.

Speaker: Nicolas Usunier - Francis Maes, University Pierre et Marie Curie Computer Science Laboratory of Paris 6 (LIP6) Statistical Machine Learning Team
Questions