nieme

classification - ranking - regression
reinforcement learning

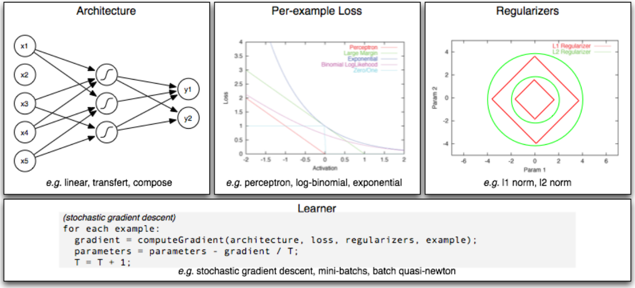
Speaker: Nicolas Usunier
-
Francis Maes,
University Pierre et Marie Curie
Computer Science Laboratory of Paris 6 (LIP6)
Statistical Machine Learning Team

Outline

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 =

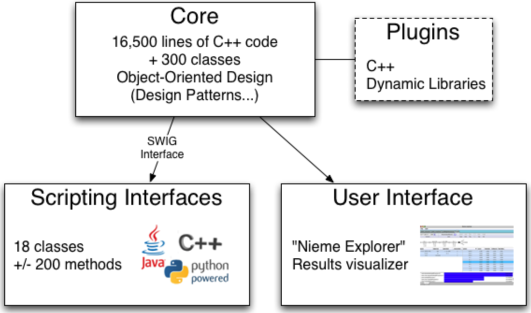
Architecture	How to compute outputs given inputs ?
+ Loss	How to penalize parameters given an example ?
+ Regularizers	How to enforce simple models ?
+ Learner	How to learn the parameters ?

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

Model	Architecture	Loss	Regularizers	Learner
Perceptron	linear	perceptron	none	stochastic descent
Logistic regression	linear	log-binomial	none	batch quasi-newton
Pegasos linear SVM	linear	hinge loss	l2	pegasos learner
Multilayer perceptron	linear $\circ$ transfer $\circ$ linear	perceptron	none	stochastic descent
L1-maxent classifier	multi-class linear	log-binomial	l1	batch quasi-newton
Pegasos multi-class SVM	multi-class linear	hinge loss	l2	pegasos learner
Least-square regression	linear	squared loss	none	batch quasi-newton
Custom	linear $\circ$ transfer	absolute loss	l1 + l2	batch rprop
Many others	...	...	...	...

Architecture



Compiles under Windows, Linux and MacOS X

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")
```

python powered

Java

```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");
```

C++

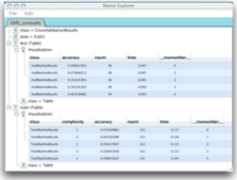
Maximum Entropy = Multiclass Linear architecture, Log-binomial loss, L2 regularizer, L-BFGS learner

Explorer

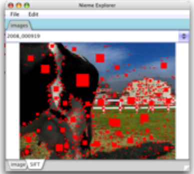
Visualization tool: vectors, tables, models and more.

Parameter Vectors

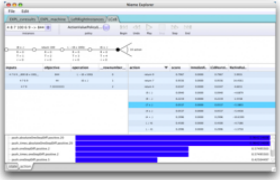
Experimental Results



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:-0.286957 3:-0.271523 4:-0.298701 5:-0.876712 6:-1
2 1:0.105263 2:-0.46087 3:-0.615894 4:-0.714286 5:-0.664384 6:-1
```

- Sparse Vectors, binary features, 3 classes:

```
yes 6:1 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:
no 2:1 3:1 20:1 22:1 23:1 33:1 35:1 36:1 47:1 50:1 51:1 58:1 67:1 76:1 79:1
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 7
```

- Equivalently, short syntax:

```
yes 6 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 87
```

- 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:

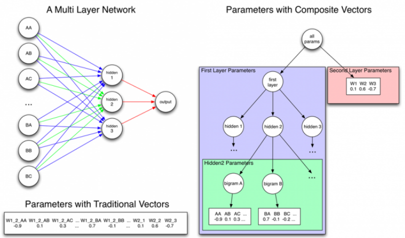
```
# First example (bipartite)
1 f4 f5 f6
0 f1:0.7 f2:0.3 f3:1.0
1 f6 f7:0.2 f8

# 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

- Website: `http://nieme.lip6.fr/`
- 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.

Questions

Nieme - `http://nieme.lip6.fr/`