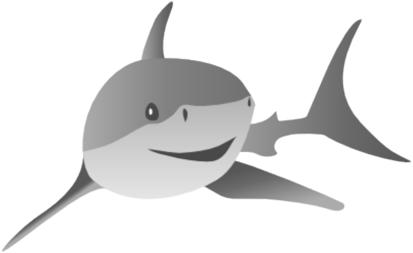
Restricted Boltzmann Machines in Shark

Practical Session

Oswin Krause
Department of Computer Science



Shark 3.0

- General machine learning library
- Written in C++
  - 80k source LOC
  - 10 Years of development
  - LGPL
- Targeted towards research
- Trade-offs
  - ⊕ performance
  - ⊕ genericity
  - ⊖ complexity
  - ⊖ knowledge about C++
- C. Igel, V. Heidrich-Meisner, and T. Glasmachers. Shark. JMLR 9, pp. 993-996, 2008.

What does it offer?

- Does not offer ready made solutions
- Shark is a toolbox. . .
  - Classification
  - Regression
  - Optimization
- . . . and a toolbox to make tools
  - Algorithms are modular
  - Library includes many bits and pices. . .
  - . . . and code to connect them
  - Easy to add new pieces

Examples of Implemented Algorithms

- SVMs: linear, nonlinear, multi-class
- Neural Networks: classification, regression, autoencoder
- Classification and Regression Trees, Random Forests
- Nearest-Neighbor algorithms: linear and kernel
- Restricted Boltzman Machines
- Clustering: (kernel) k-means
- Multi-Objective Optimization (MO-CMA-ES)
- CMA-ES
- General optimisation Methods: CG, BFGS, ...
- ...

Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMS)

- $p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}e^{-\frac{1}{T}E(\boldsymbol{v},\boldsymbol{h})}$ with $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-\frac{1}{T}E(\boldsymbol{v},\boldsymbol{h})}$
- Generalized formula for $E(\boldsymbol{v},\boldsymbol{h})$

$$E(\boldsymbol{v},\boldsymbol{h}) = f_h(\boldsymbol{h}) + f_v(\boldsymbol{v}) + \phi_h(\boldsymbol{h})^T W \phi_v(\boldsymbol{v})$$

- Allows for a wide set of RBMs used in practice(binary, exponential-binary, Gaussian-binary, semi-RBMs,. . .)
- f_h and ϕ_h depend on the choice of hidden distributions (same for visible)
- $f_h(\boldsymbol{h}) = \boldsymbol{b}^T\boldsymbol{h}, f_v(\boldsymbol{v}) = \boldsymbol{c}^T\boldsymbol{v}$ and $\phi_h(\boldsymbol{h}) = \boldsymbol{h}, \phi_v(\boldsymbol{v}) = \boldsymbol{v}$ leads to

$$E(\boldsymbol{v},\boldsymbol{h}) = \boldsymbol{b}^T\boldsymbol{h} + \boldsymbol{c}^T\boldsymbol{v} + \boldsymbol{h}^T W \boldsymbol{v}$$

The binary-binary RBM for $\boldsymbol{v} \in \{0,1\}^n$ and $\boldsymbol{h} \in \{0,1\}^m$.

Restricted Boltzmann Machines in Shark

- Shark implements the generalized energy formula.
- Supports binary, Gaussian, bipolar and truncated exponential distributions.
- Variety of gradient approximators:
  - Analytic (exponential time!)
  - Contrastive Divergence
  - Persistent Contrastive Divergence
  - Parallel Tempering
- Estimators for Z
  - Analytic (exponential time!)
  - AIS
  - Bridge-Sampling

Take a look at it

Code-Time!

Some Assignments

Play around with the code:

- a) BinaryRBM.cpp is about training an RBM
  - Try to get a high likelihood/ prevent divergence
  - Change hyper parameters(k, learning rate, momentum, regularization)
  - What happens with low/high values?
  - Try Parallel Tempering instead of CD

```
size_t chains = 10;
BinaryParallelTempering estimator(&rbm);
estimator.chain().setUniformTemperatureSpacing(chains)
```

  - Momentum is easier to use when multiplying the learning rate by $(1-\nu)$, where $\nu \in [0,1]$ is the strength of the momentum

Some Assignments

- b) BinaryRBMFeatureClassification.cpp is a semi-supervised task
  - First an RBM is trained and then an LDA is trained on the learned features
  - Goal: Reduce classification error
  - Does RBM model quality affect performance?
  - Do not forget to regularize LDA
- c) BinaryRBMClassification.cpp is also semi-supervised
  - First an RBM is trained and then transformed into an FFNet which is then fine tuned
  - Goal: Reduce classification error
  - Does RBM model quality affect performance?
  - (Try the other tasks first!)