Stochastic versus BFGS Training in Neural Discrimination of RF-Modulation

Contents

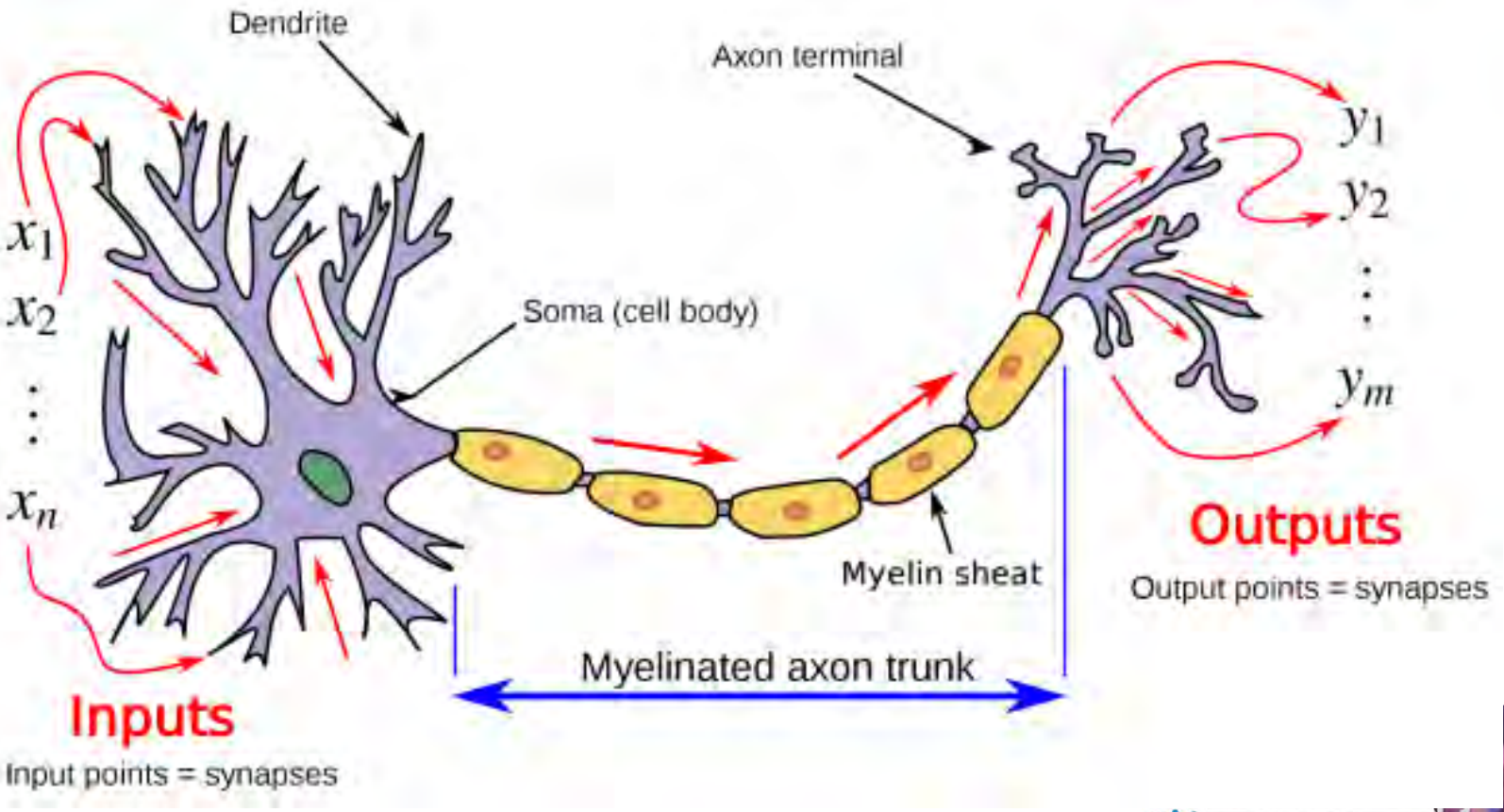
1. *Artificial intelligence*
2. *Wave reconstruction*
3. *ANN training*
4. *Results*
5. *Conclusions*

Contents

1. *Artificial intelligence*
2. *Wave reconstruction*
3. *ANN training*
4. *Results*
5. *Conclusions*

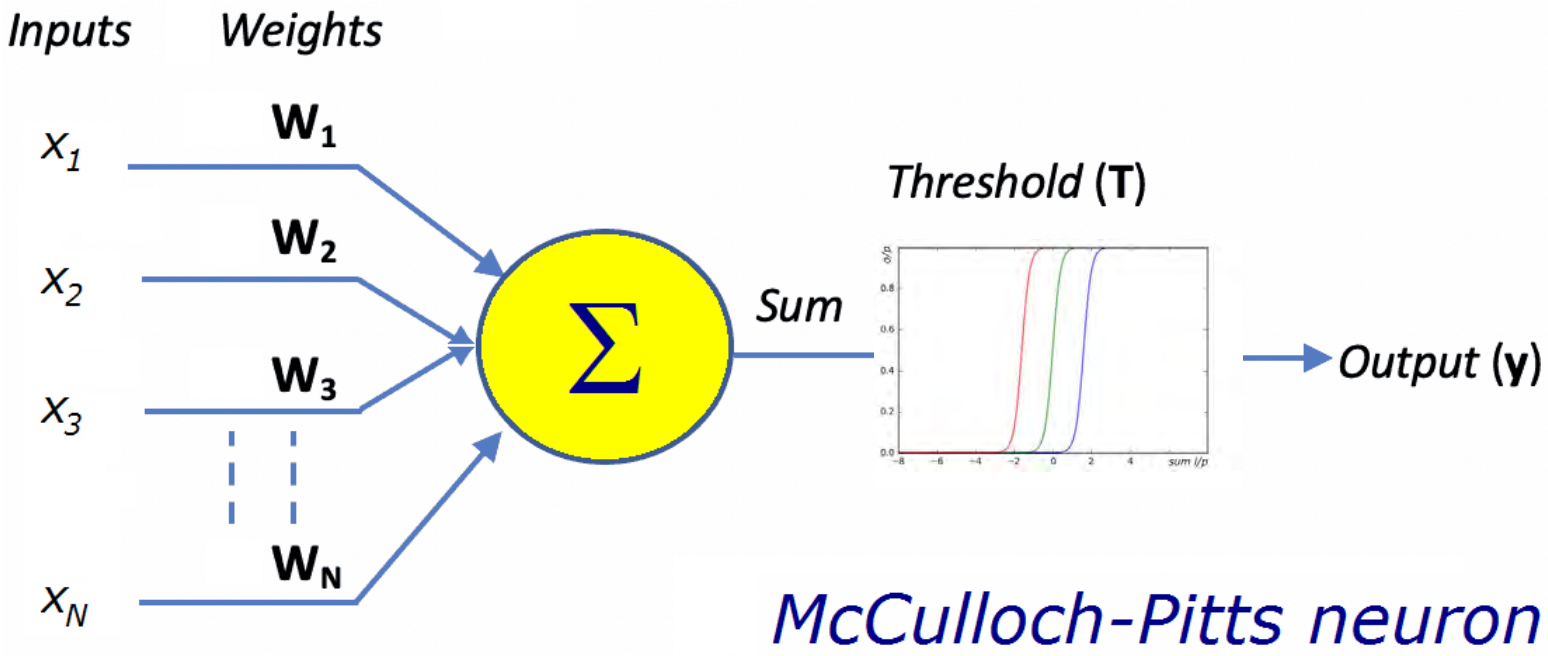
Bio-analogy

- representation of data selection with:
  - sum
  - threshold



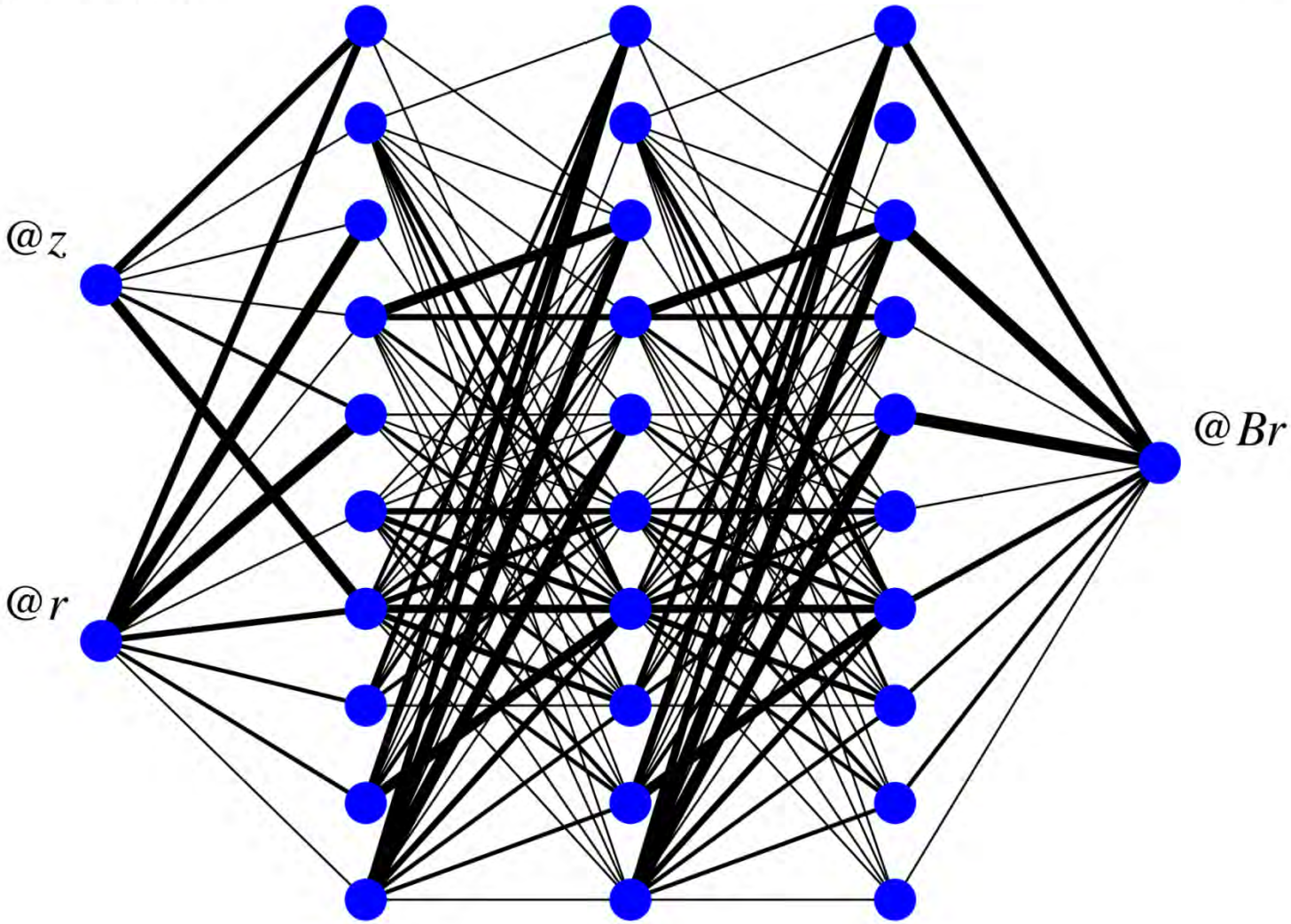
Artificial Intelligence

- representation of data selection with:
  - sum
  - threshold



Multi-Layer Perceptrons @ ROOT

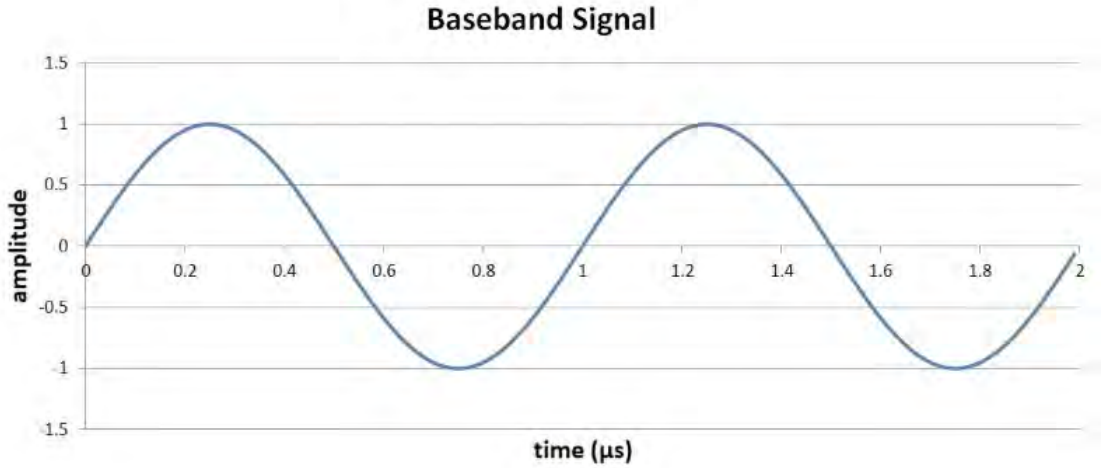
```
// show network structure
mlp->Draw()                                                    ;
```

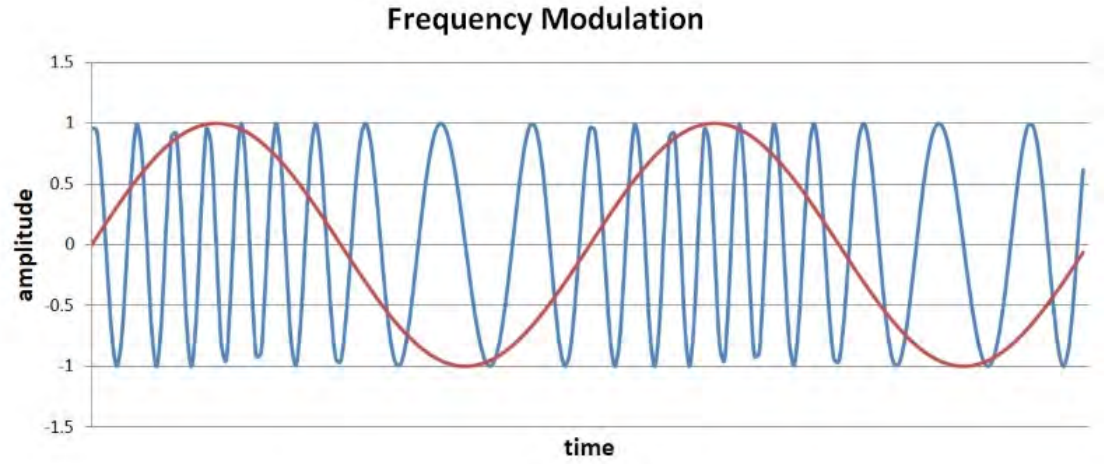


Contents

1. *Artificial intelligence*
2. *Wave reconstruction*
3. *ANN training*
4. *Results*
5. *Conclusions*

Radio frequency modulation





Shift keying:

- *ASK, amplitude*
- *FSK, frequency*
- *PSK, phase*
- *ASK-LSB*
- *ASK-USB*

Magic sample number

- RF wave $y = p + Asin(2\pi ft + \phi)$

  sampling 1 : 3.675

  f_0 12ooo Hz

  Δ 1 / 44100 s

- pedestal: find from average

$$\langle y \rangle = p + A_e sin\left(2\pi ft\frac{t_i + t_f}{2} + \phi\right) sinc\left(\frac{2\pi f \Delta t}{2}\right)$$

$$A_e = \frac{A}{sinc(\pi f \Delta)}$$

- magic N: $\Delta t = 11\Delta$ … $\delta p = 0.0023 A_e$

Amplitude

- same N = 11 : $\langle \delta^2 y \rangle = A_e \langle \delta^2 (sin) \rangle$

Frequency

- same N = 11 :

$$\langle y(y - y_{k\Delta}) \rangle = pA_e(\langle sin \rangle - \langle sin_{k\Delta} \rangle) + A_e^2(\langle sin^2 \rangle - \langle sin \cdot sin_{k\Delta} \rangle)$$

$$\simeq A_e^2 sin^2\left(\frac{2\pi f k\Delta}{2}\right)$$

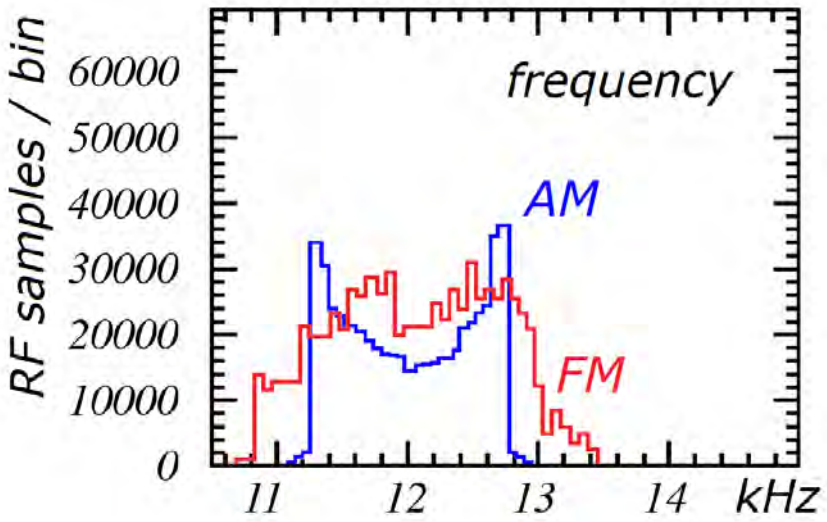
$$\simeq const + \pi k\Delta A_e^2 sin(2\pi f k\Delta) \cdot \delta f$$

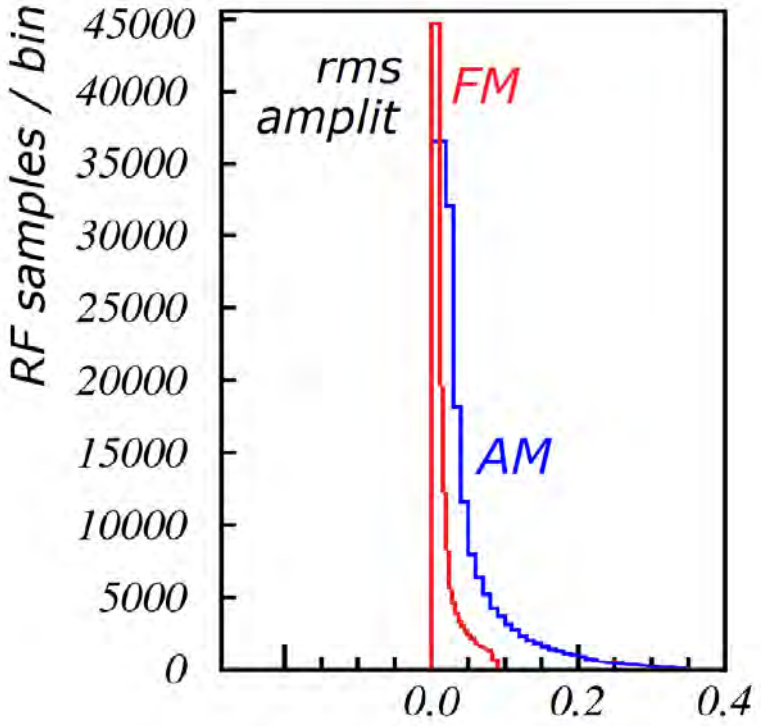
(k = 1 ; max sensitivity)

Phase

- $\delta\phi = \phi_{current} - \phi_{previous}$

$$\langle\, y \cdot cos(\pi f t)\,\rangle \simeq \frac{A_e}{2} sin\phi$$





Overcoming noise / jamming

$$u(t) = p + \text{noise} + Asin(2\pi ft + \phi)$$

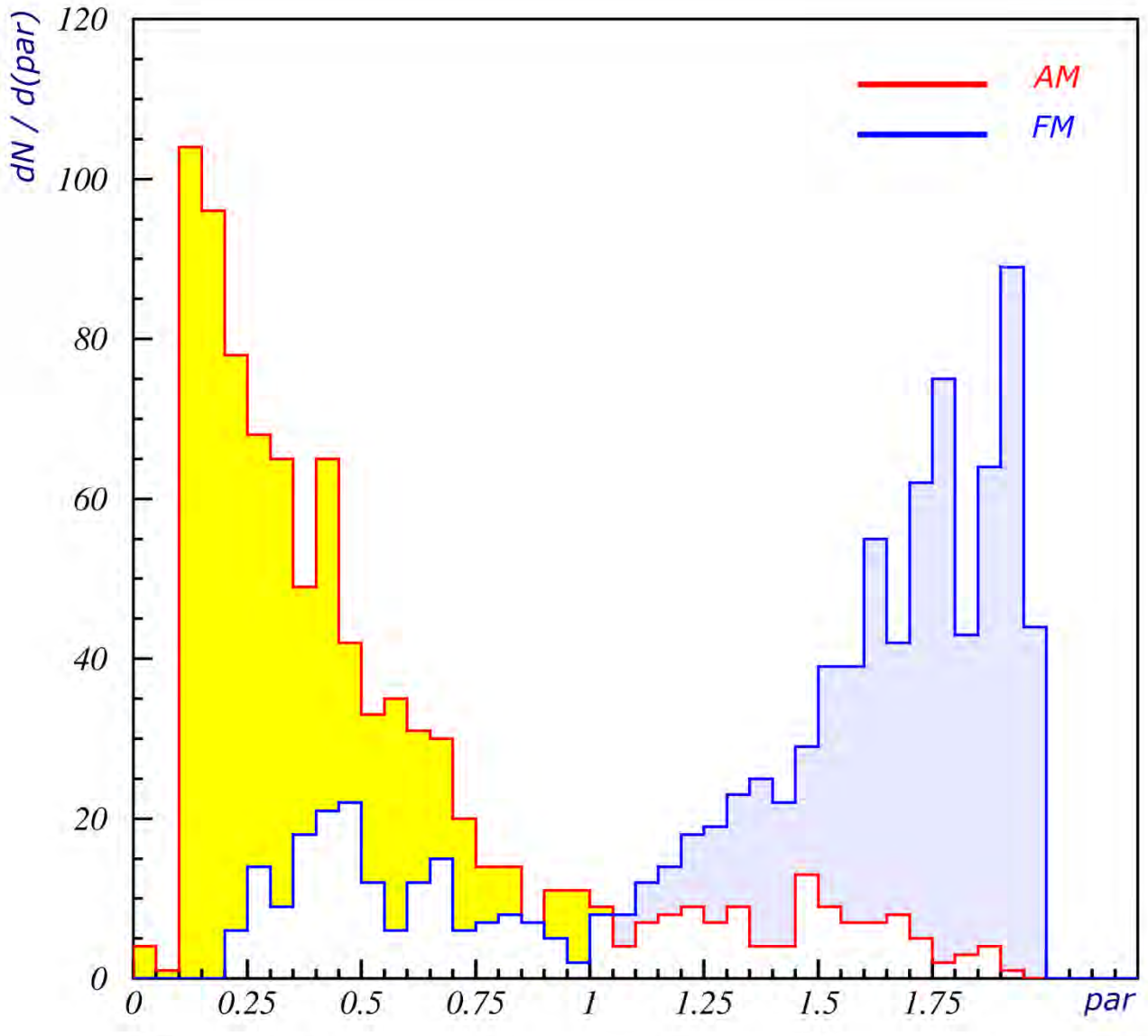
$$\langle u \rangle = p + \langle \text{noise} \rangle + A_e sin\left(2\pi f \frac{t_i + t_f}{2} + \phi\right) sinc(\pi f(t_f - t_i))$$

$$\langle \delta^2 u \rangle = \langle \delta^2 \text{noise} \rangle + A^2 \langle \delta^2 sin \rangle + 2 \underbrace{\langle \delta(\text{noise})\delta A \rangle}_{\simeq 0}$$

$$2\langle \delta^2 u \rangle_1 - 2\langle \delta^2 u \rangle_2 = A_1^2 - A_2^2 = \delta A^2$$

Parameter - 11

```
phi 1 = (L2-L1) * d_p
        L1 = where d
        L2 =
    2 = <distr.> in t
    3 = <N_phi> - <1/
    4 = <N_phi*delta^
```



Parameter - 13

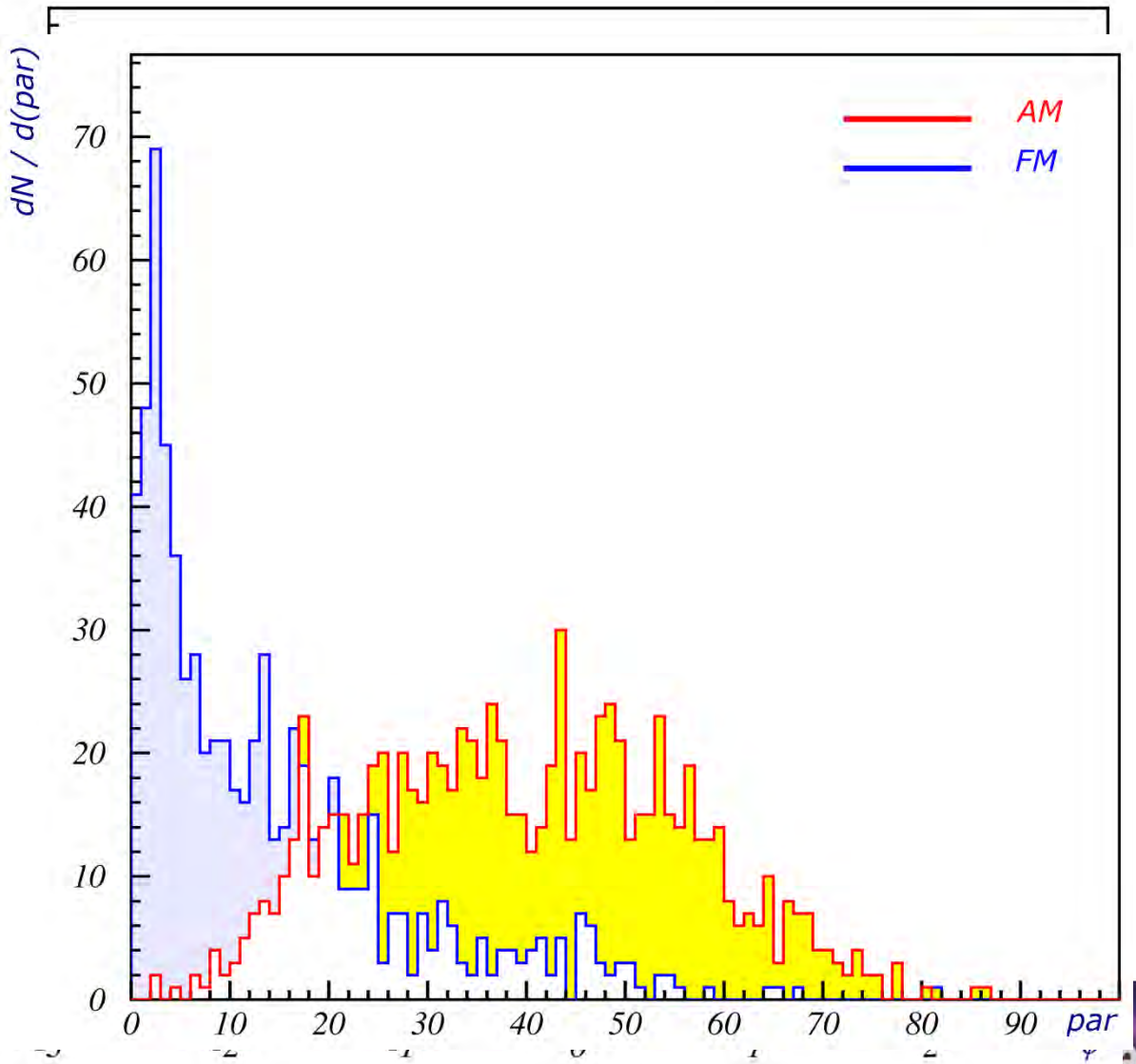
```
                                                    1/dφ
-------------------------------------------------------
phi 1 = (L2-L1) * d_phi

          L1 = where distr
          L2 =

     2 = <distr.> in the r

     3 = <N_phi> - <1/N_ph

     4 = <N_phi*delta^2_ph
-------------------------------------------------------
```



Contents

1 *Artificial intelligence*

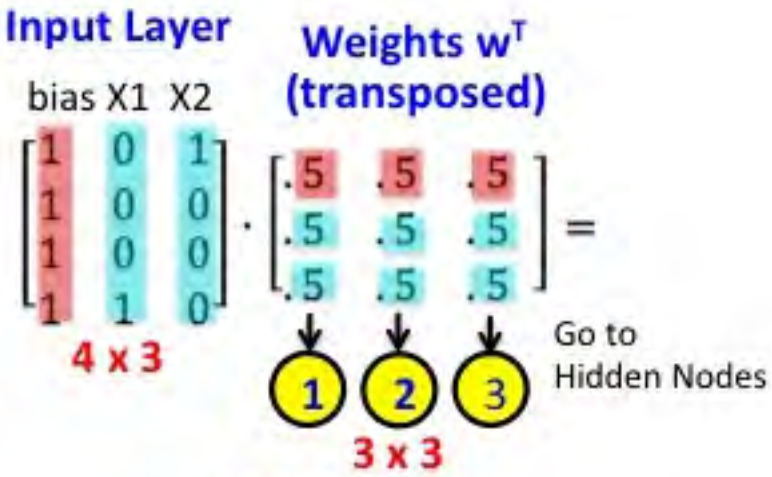
2 *Wave reconstruction*

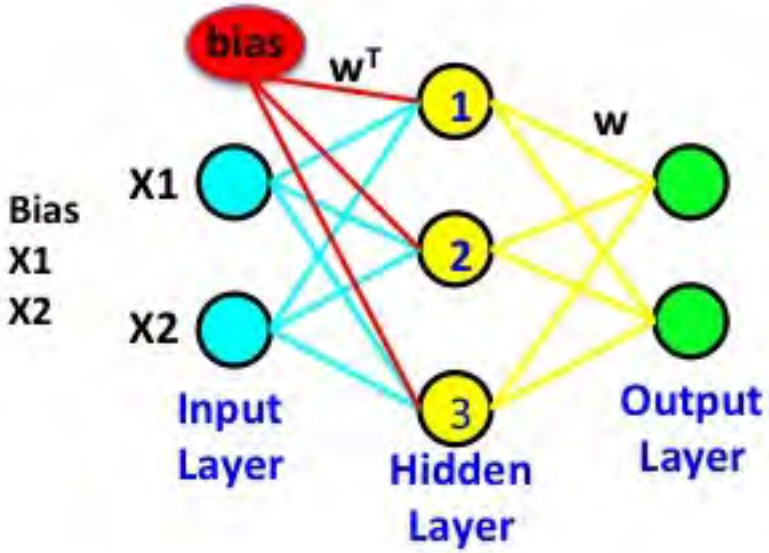
3 *ANN training*

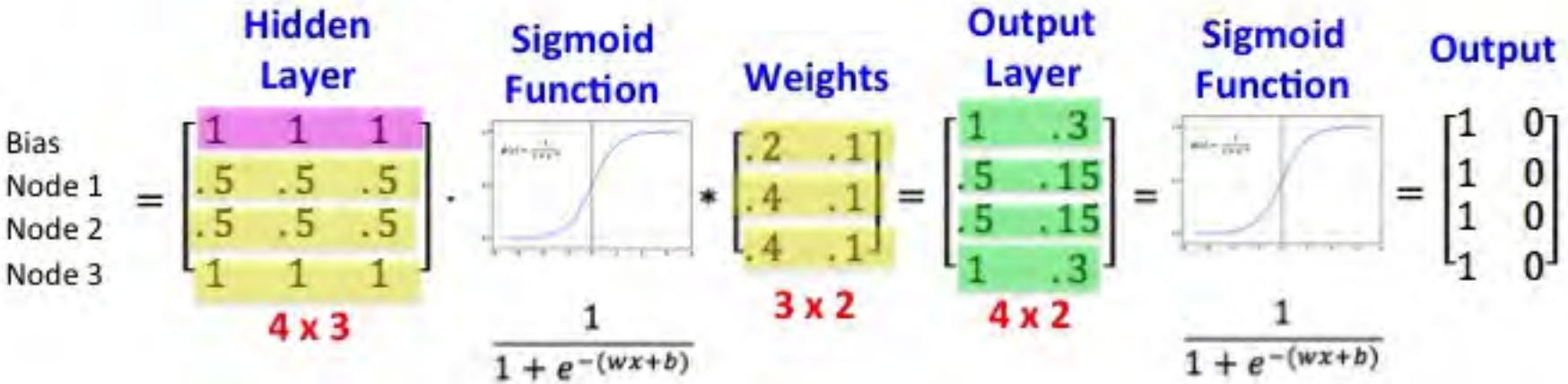
4 *Results*

5 *Conclusions*

MLP run-through







ROOT MLP

ROOT
Reference Guide

Version master

https://root.cern.ch/doc/master/classTMultiLayerPerceptron.html

Public Types

enum EDataSet { kTraining, kTest }

enum ELearningMethod {
kStochastic, kBatch, kSteepestDescent, kRibierePolak,
kFletcherReeves, kBFGS
}

▸ Public Types inherited from TObject

Public Member Functions

TMultiLayerPerceptron ()
Default constructor. More...

TMultiLayerPerceptron (const char *layout, const char *weight, TTree *data, TEventList *training, TEventList *test, TNeuron::ENeuronType type=TNeuron::kSigmoid, const char *extF="", const char *extD="")
The network is described by a simple string: The input/output layers are defined by giving the branch names separated by comas. More...

Broyden-Fletcher-Goldfarb-Shanno algorithm - is very popular version of the gradient descent method. It overcomes some of the limitations of plain gradient descent by seeking with the hessian a stationary point of the cost function. The gradient needs to be zero in said point, however neither Newton's method nor BFGS are guaranteed to converge unless the function posesses a quadratic expansion in the vecinity of the optimum.

Stochastic Networks

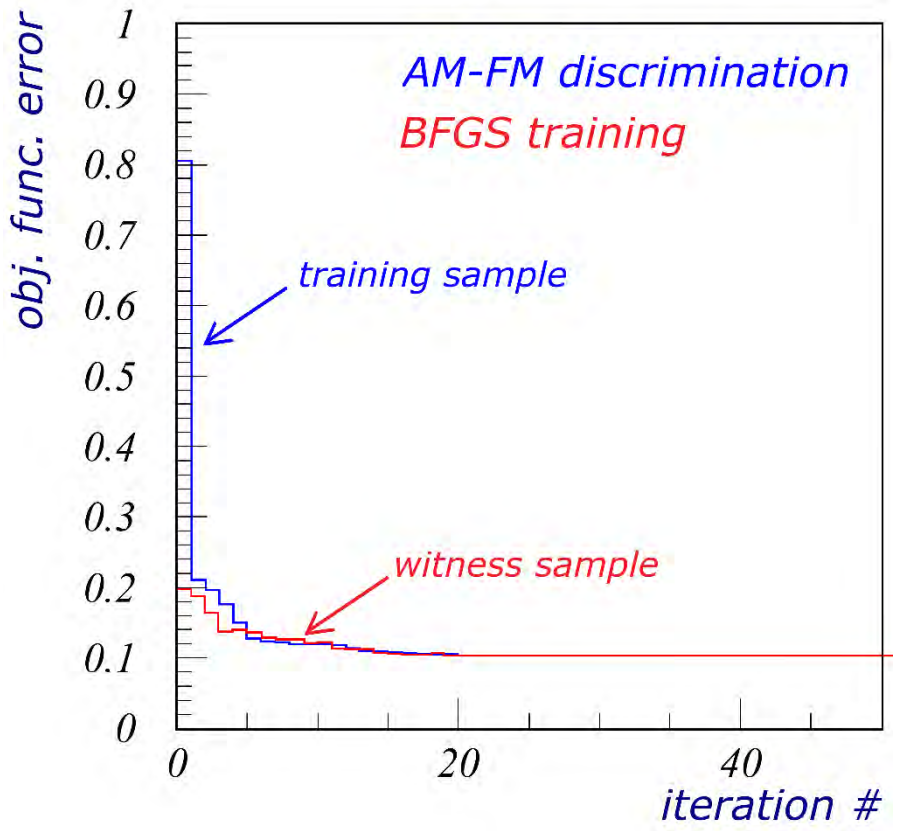
Simulated annealing has been used for many years in the field of numerical optimization. The technique is a special case of the Monte Carlo method.

The learning algorithm was the first proposed for stochastic networks and allowed dealing with hidden units in networks of the Hopfield type.

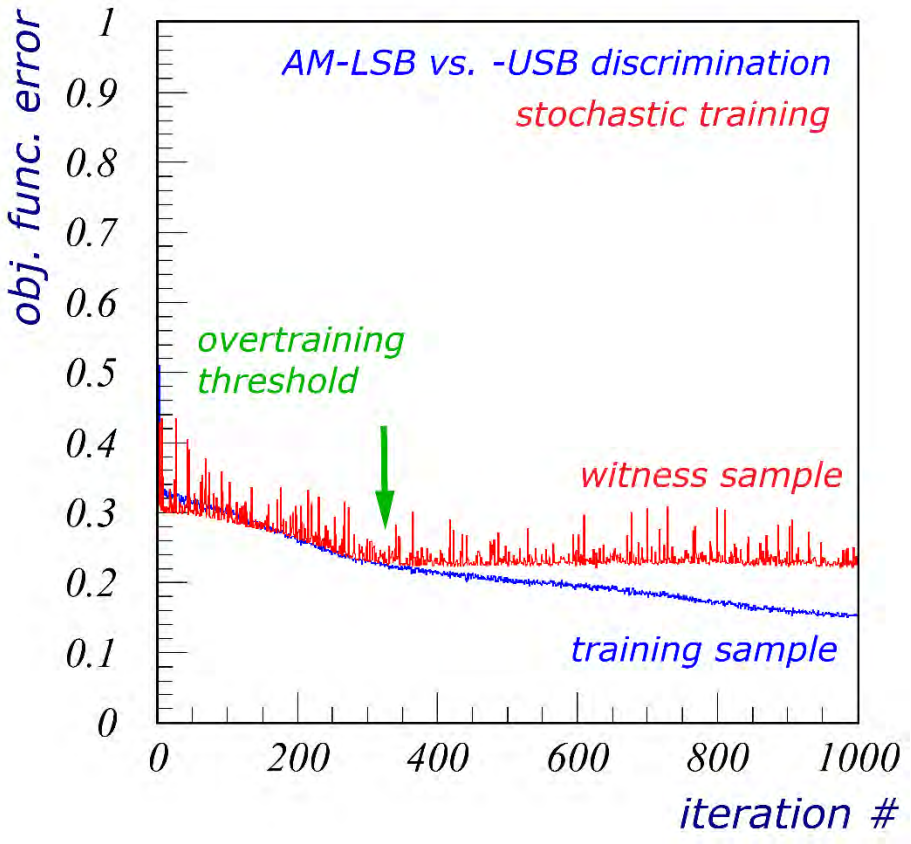
The role of noise in Hopfield networks has been studied by many physicists It can be shown, for example, that the number of false local minima that arise in Hopfield networks can be compensated with a stochastic dynamic, so that the statistical capacity of the network is improved.

ANN training

BFGS training



STOCHASTIC training



Contents

1 *Artificial intelligence*

2 *Wave reconstruction*

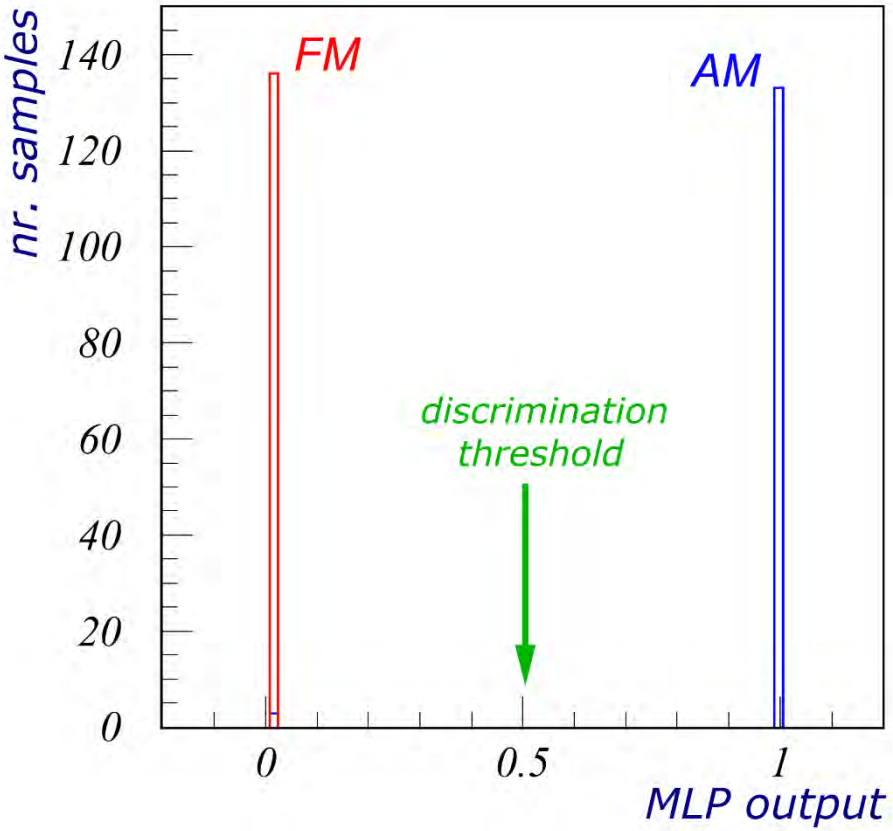
3 *ANN training*

4 *Results*

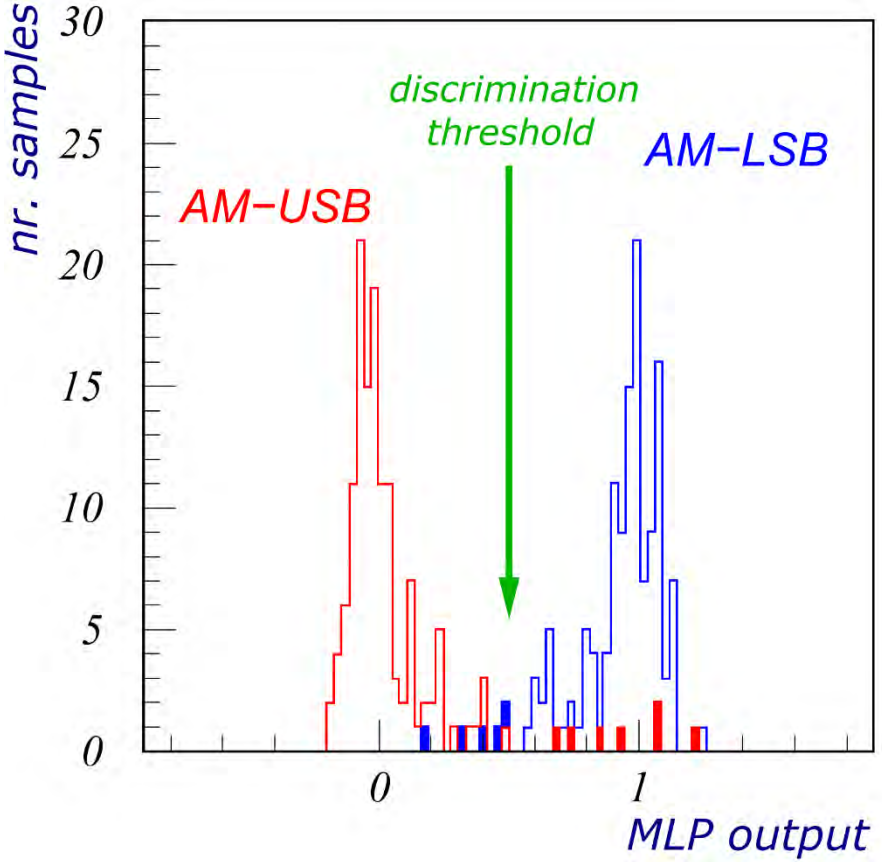
5 *Conclusions*

Results

BFGS training



STOCHASTIC training



Contents

1. *Artificial intelligence*
2. *Wave reconstruction*
3. *ANN training*
4. *Results*
5. *Conclusions*

Conclusions

Training method

- BFGS
  - adequate for “strong” distinctive features
  - can get stuck on local-minima
- Stochastic
  - adequate for categories very similar
  - slow, many iterations

Спасибо !