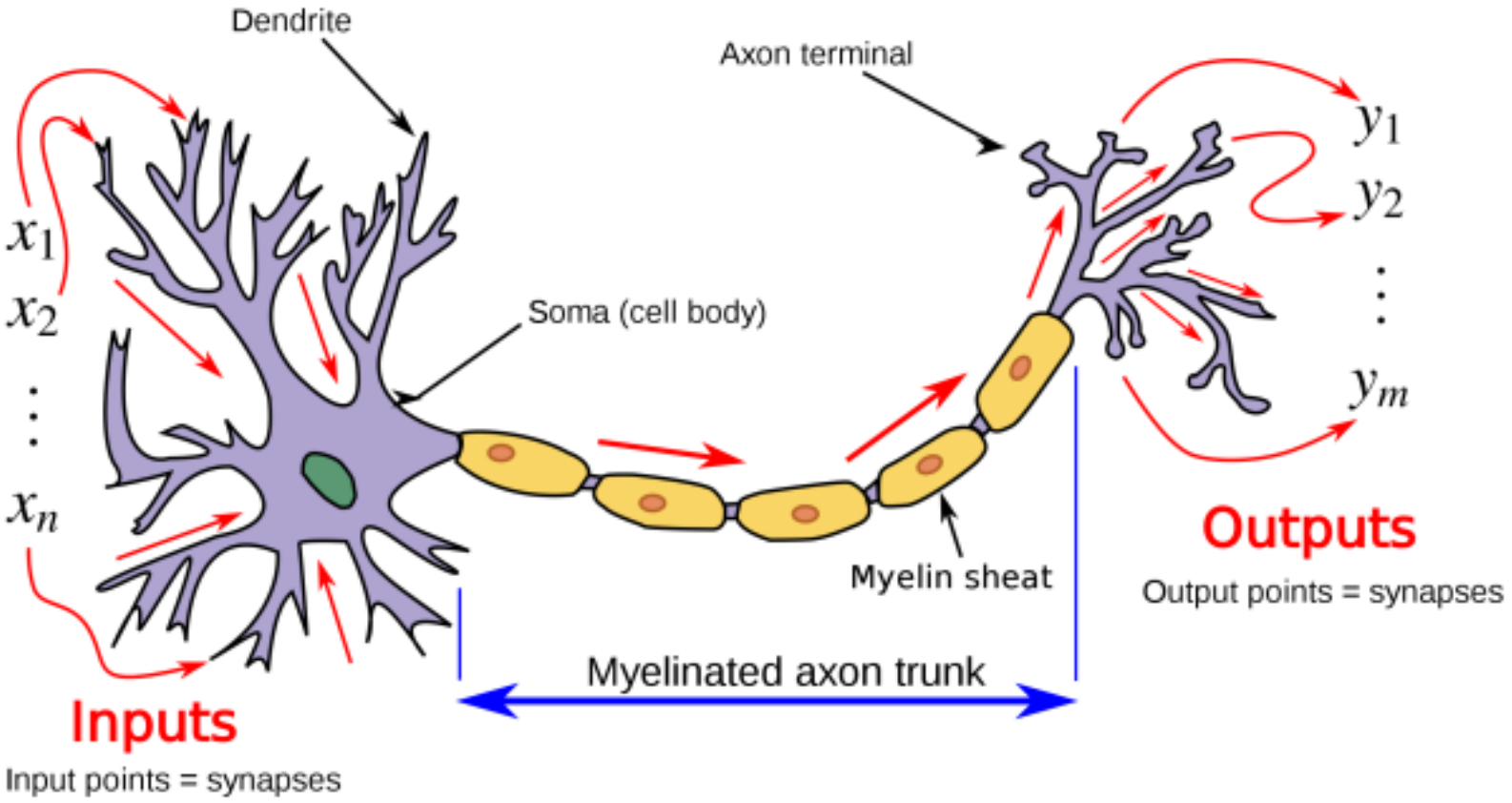
Deep learning for automatic RF-modulation classification

Contents

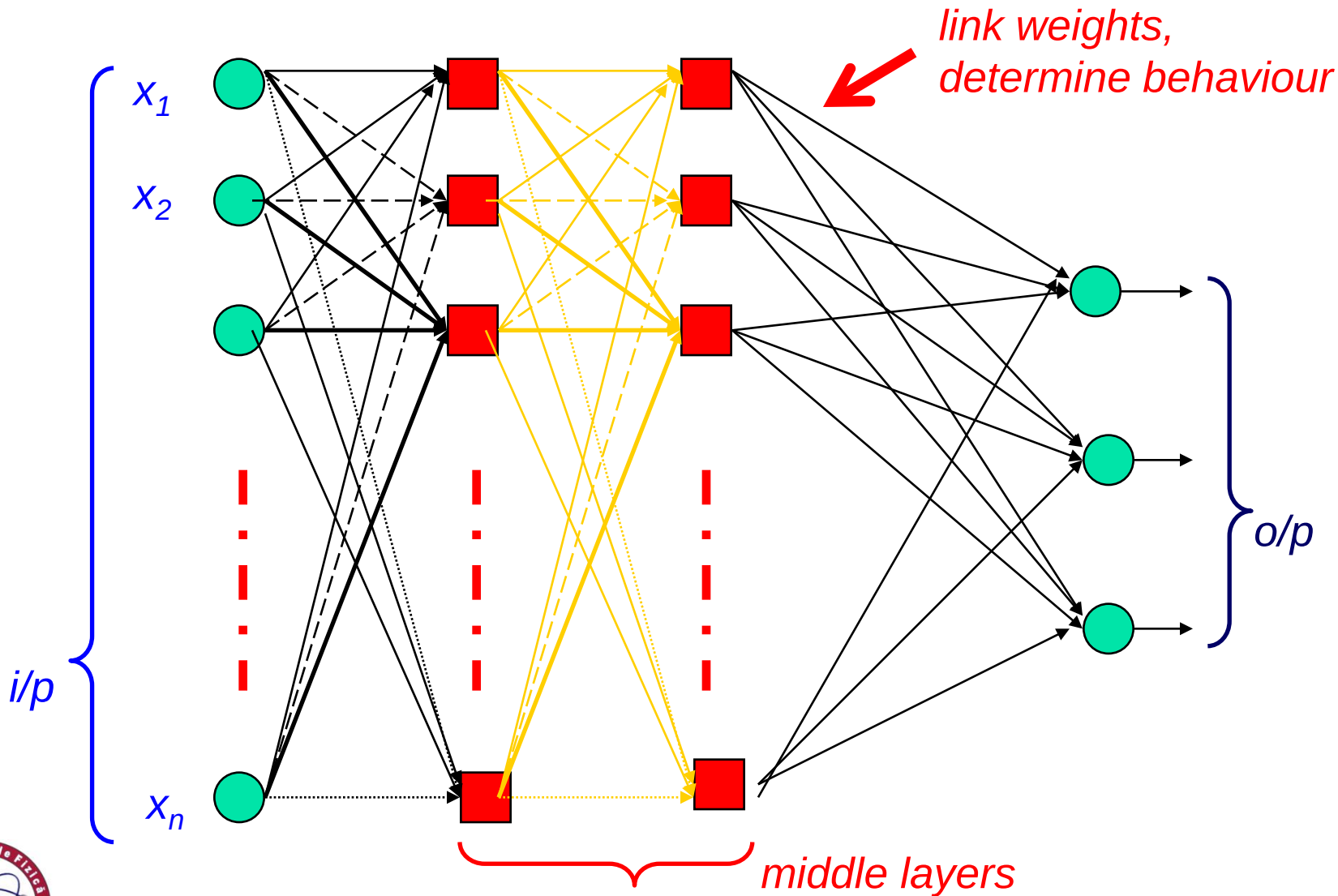
Contents

Bio-analogy

- representation of data selection with:
  - sum
  - threshold

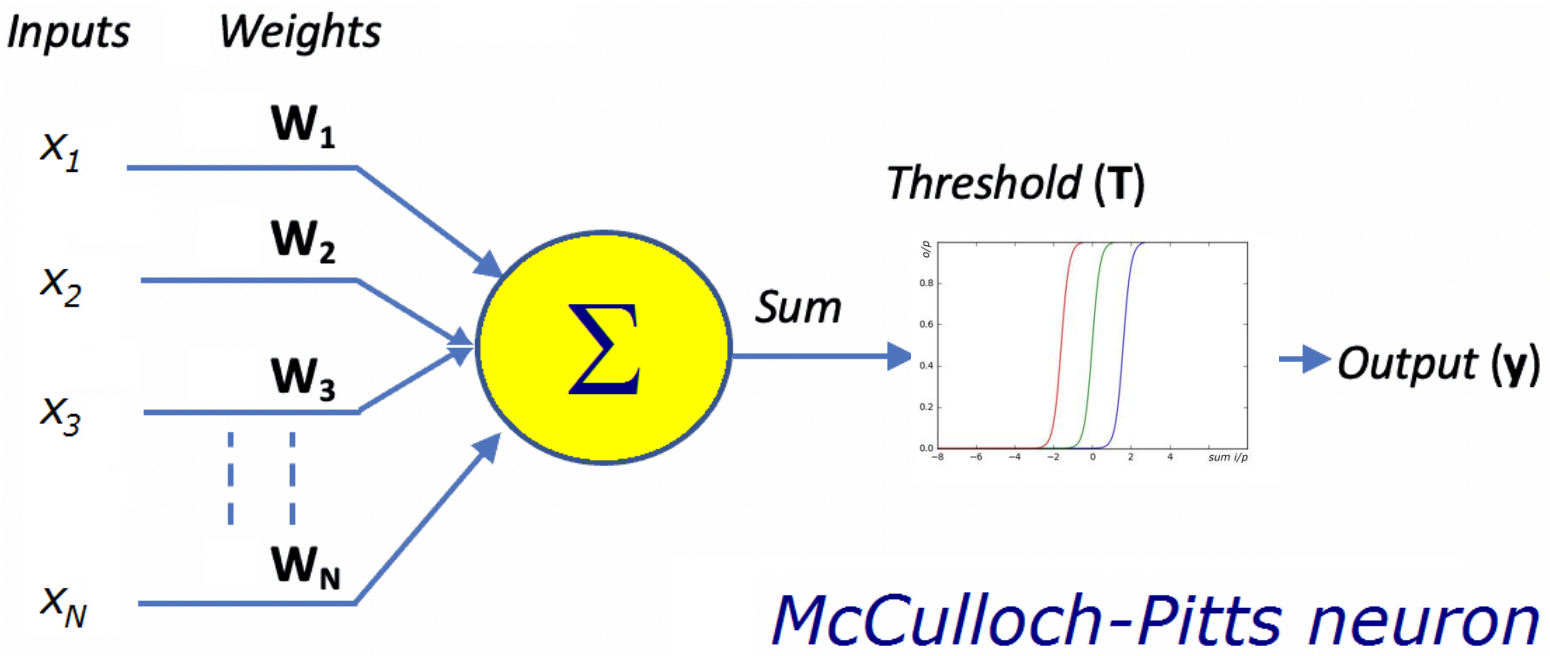


Multi-layer perceptron



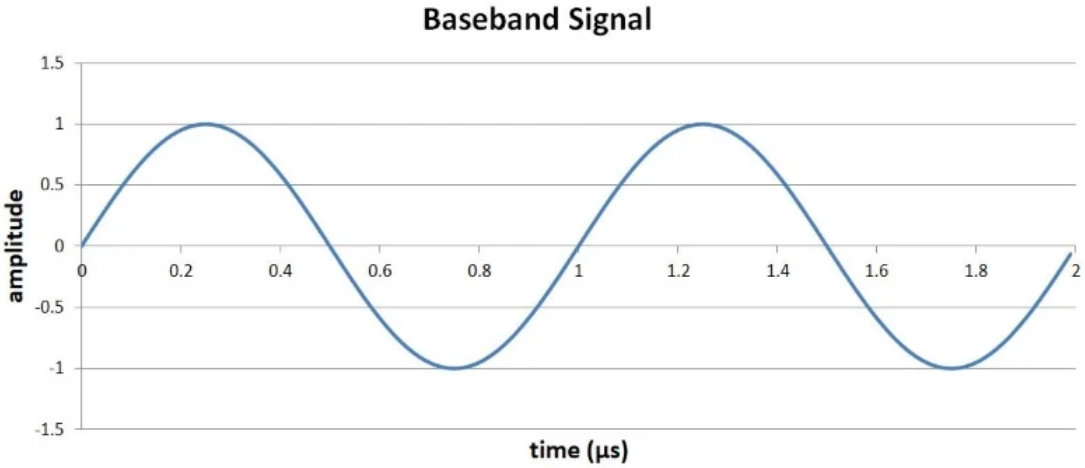
Artificial Intelligence

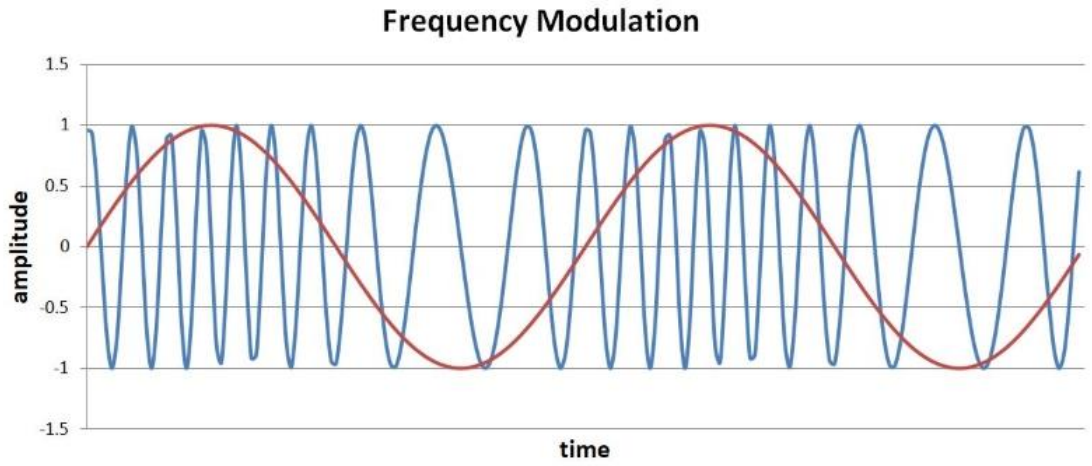
- representation of data selection with:
  - sum
  - threshold



Contents

Radio frequency modulation





Shift keying:

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

Determination of: (ped, A, f, φ)

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

Determination of: (ped, A, f, φ)

Amplitude

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

Frequency

- same N = 11 : $\langle y(y - y_{k\Delta}) \rangle = pA_e(\langle sin \rangle - \langle sin_{k\Delta} \rangle)$

$$+$$

$$A_e^2(\langle sin^2 \rangle - \langle sin \cdot sin_{k\Delta} \rangle)$$

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

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

(k = 1 ; max sensitivity)

Determination of: (ped, A, f, φ)

Phase

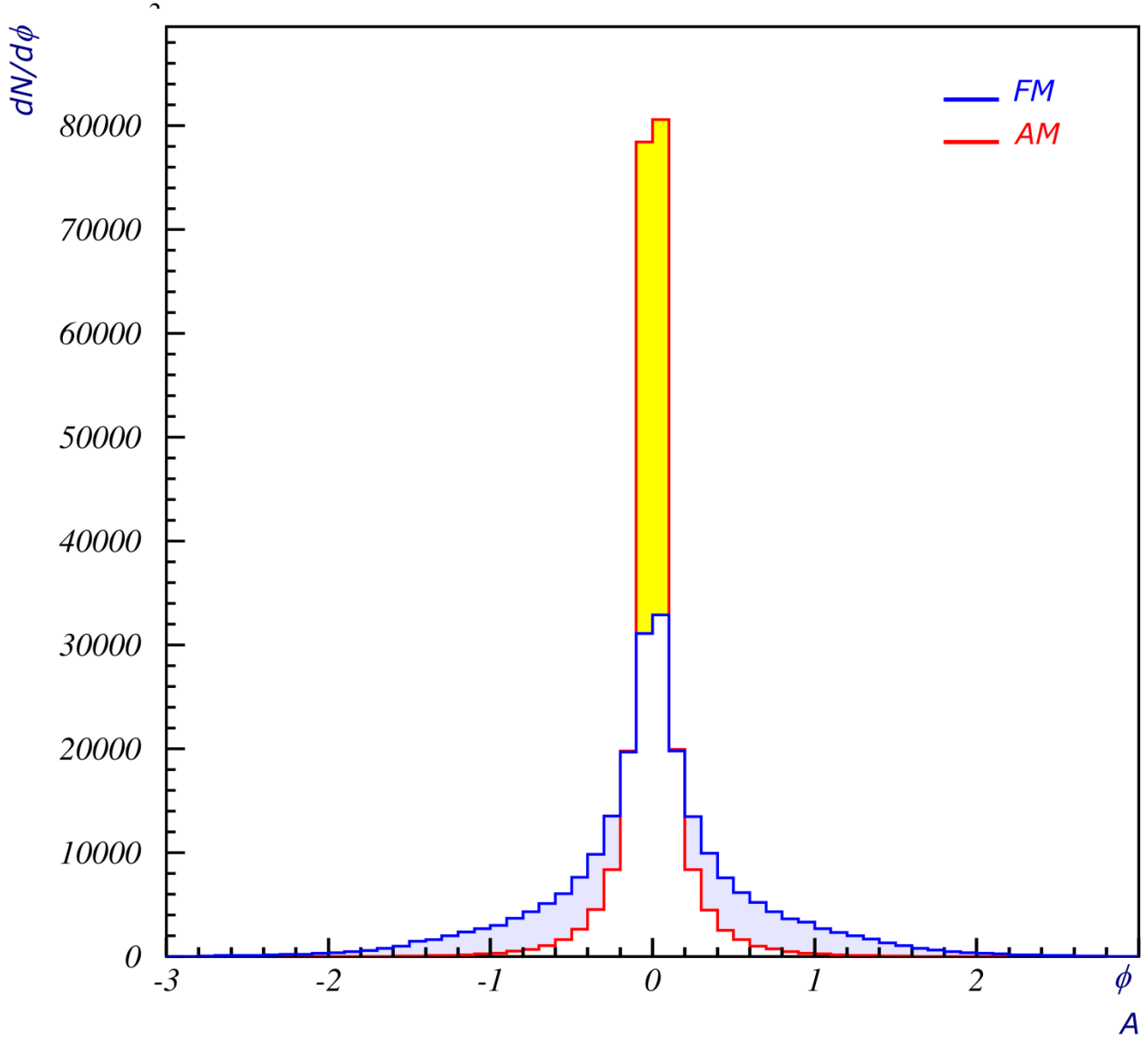
- $\delta\phi = \phi_{current} - \phi_{previous}$

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

- next: form features

Determination of: (ped, A, f, φ)

Distributions

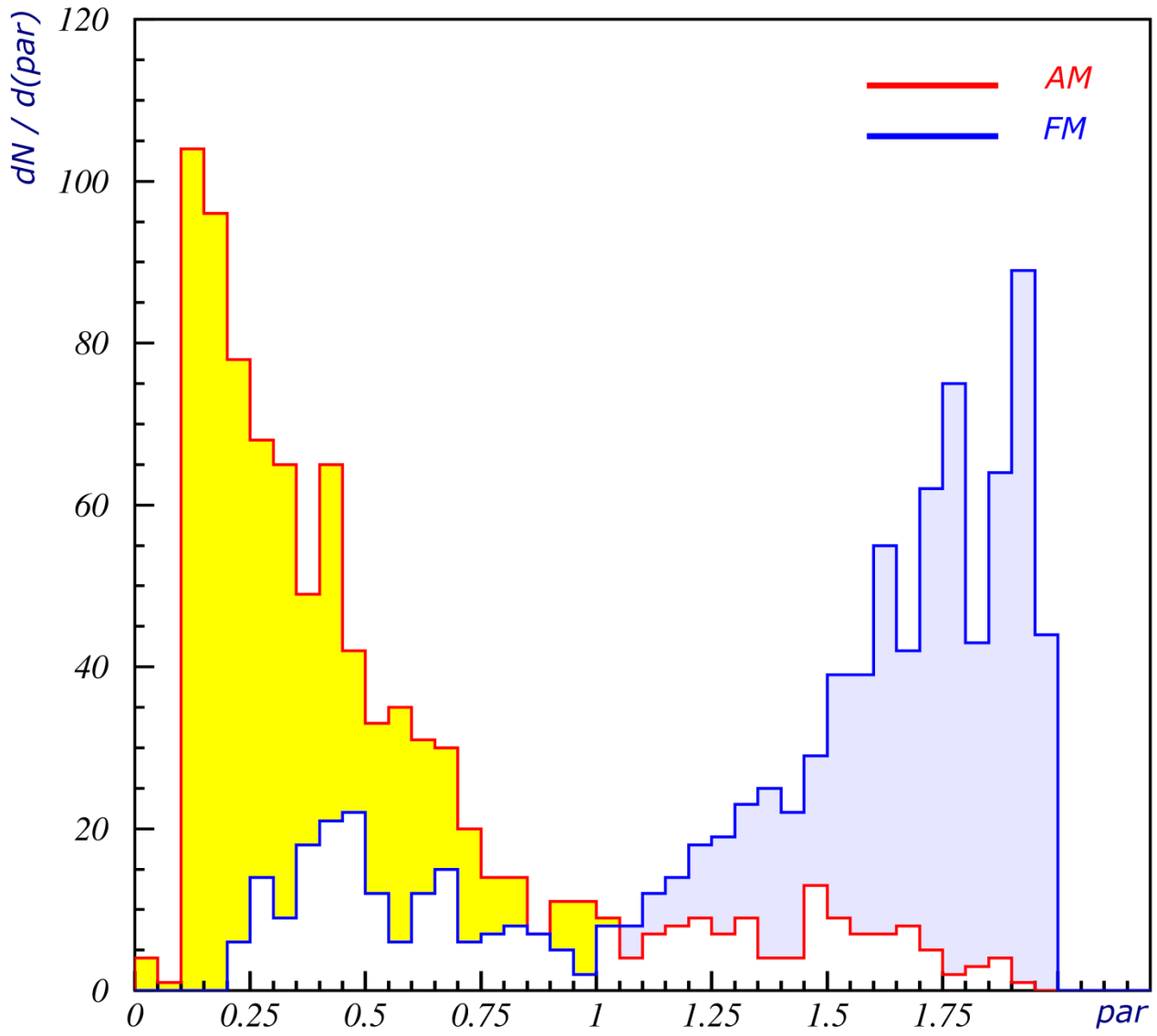


Contents

Feature formation

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

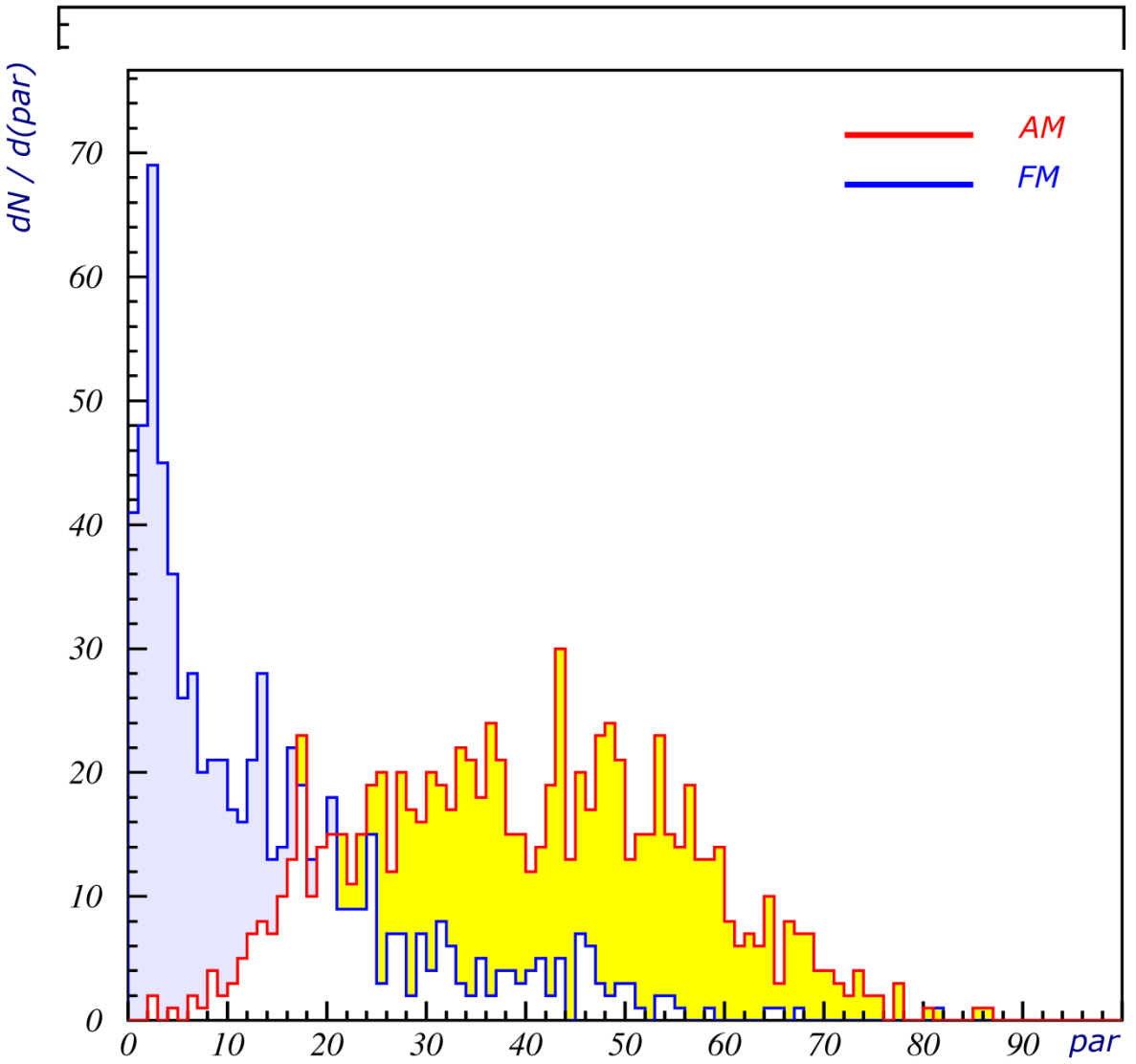
N/dϕ

```
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
```

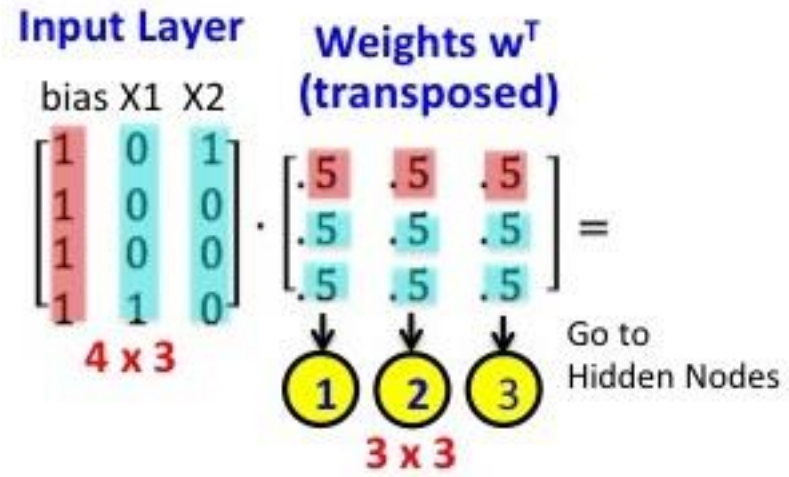
300

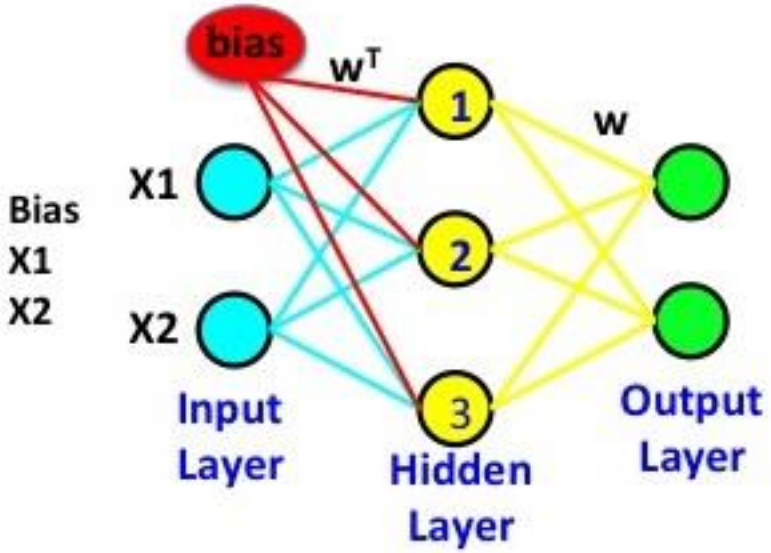
200

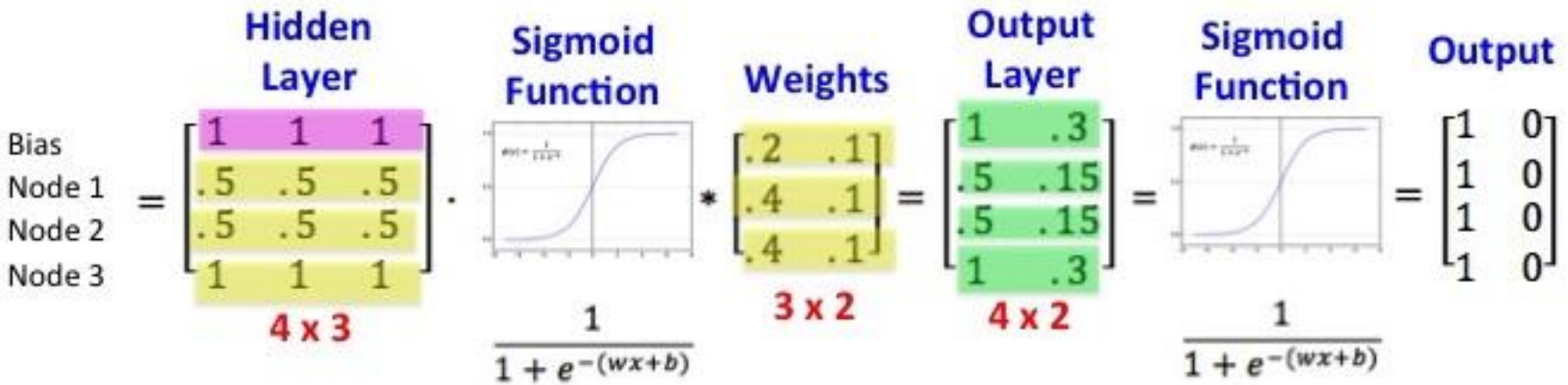
100



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

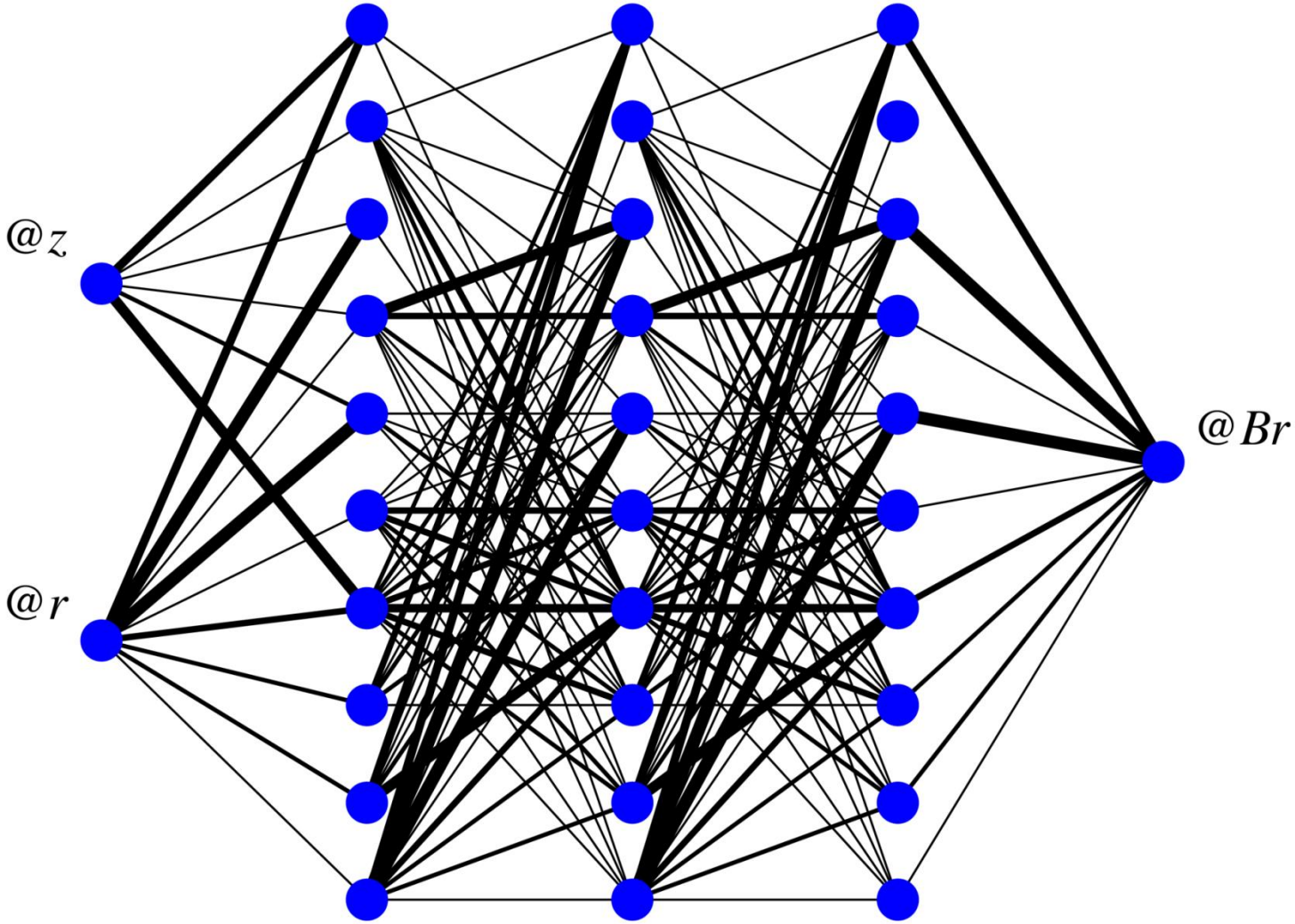
Public Member Functions

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...

ROOT MLP

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



Define ANN parameters

- learning method

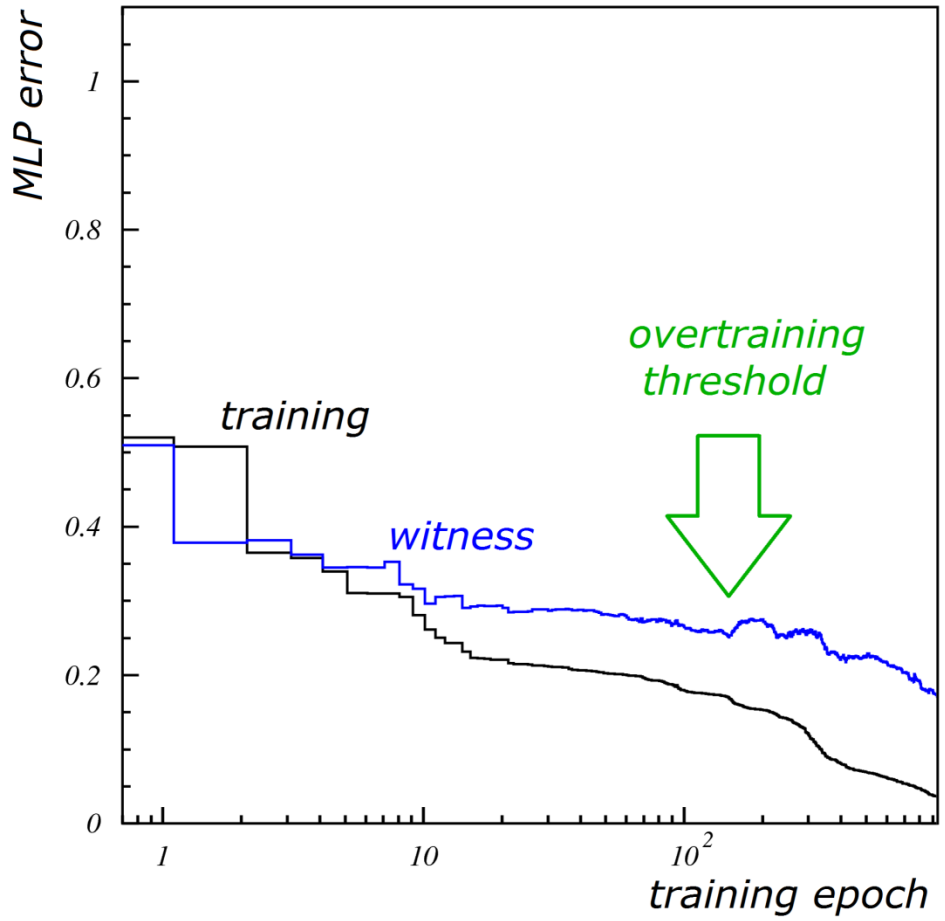
```
// set learn method ____________________________________________

mlp->SetLearningMethod(TMultiLayerPerceptron::kBFGS )      ;

    // kStochastic = default
    // kBatch
    // kSteepestDescent
    // kRibierePolak
    // kFletcherReeves
    // kBFGS
```

- # training epochs and error plot updates

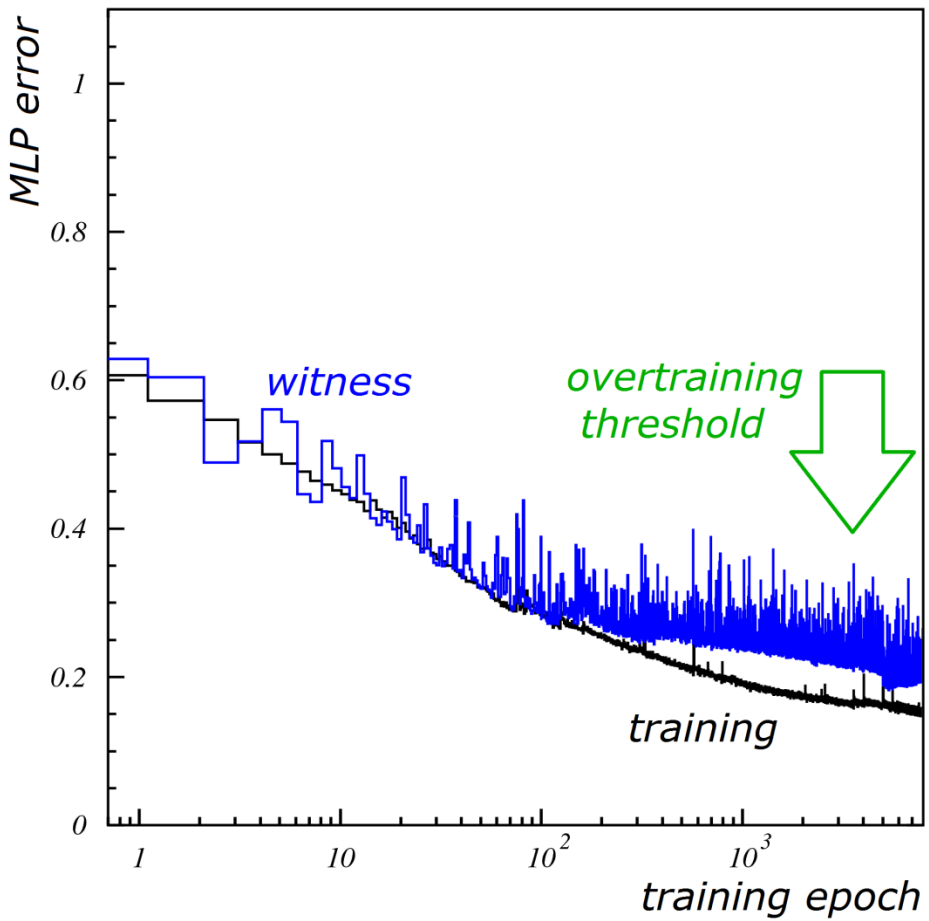
```
        // training ____________________________________________________

        mlp->Train( 1000
                    "text,update=100" )                                ;

            //  1000 events
            //  write text to console
            //  updates every 100 epochs
```

ANN training

AM vs. FM: BFGS



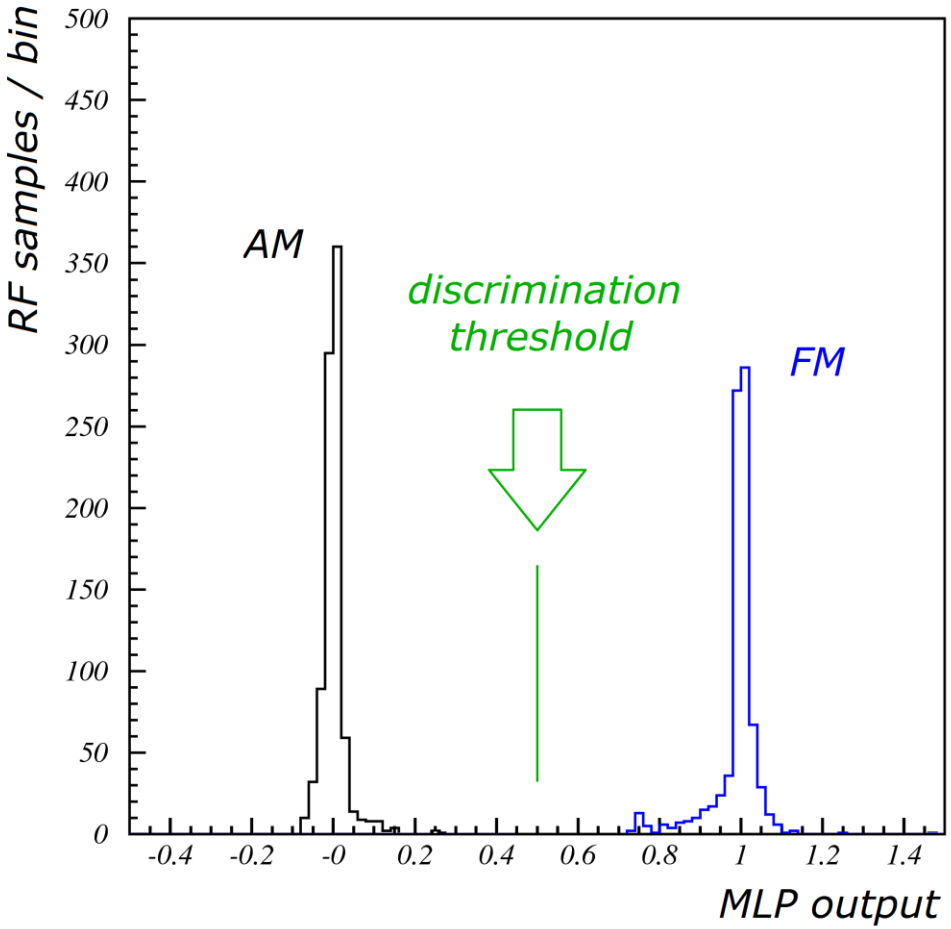
AM-LSB vs. AM-USB: stochastic



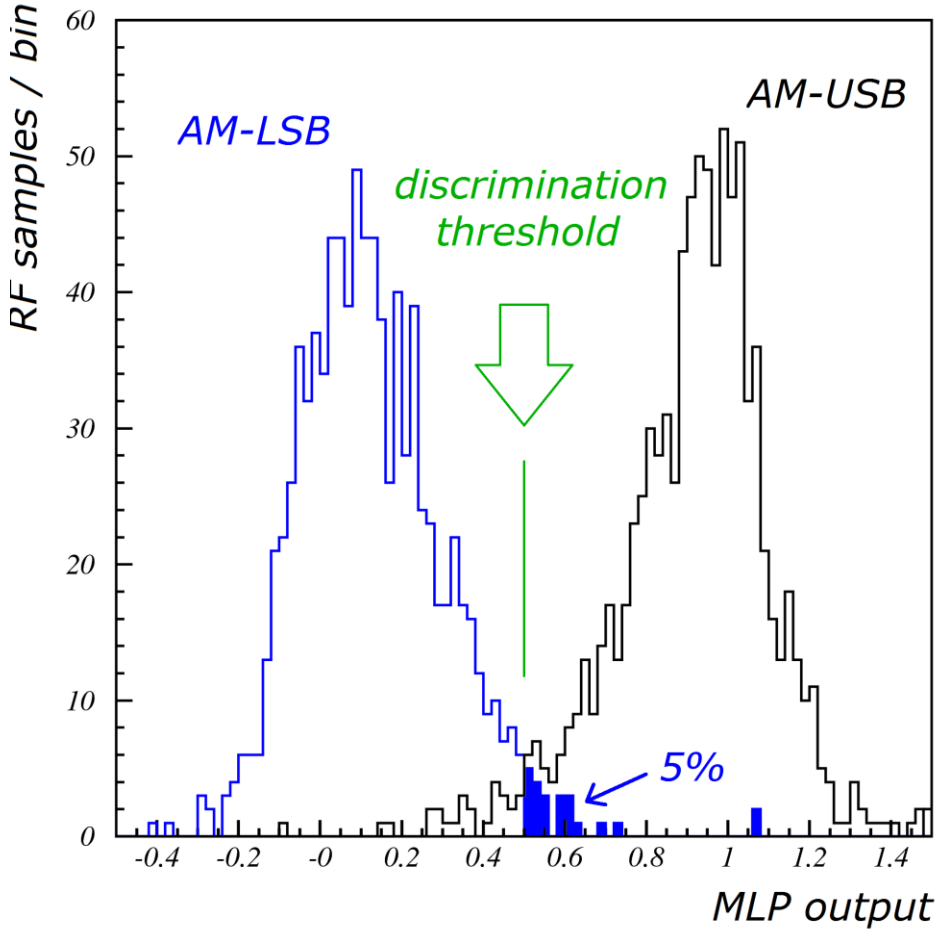
Contents

Results

AM vs. FM: BFGS



AM-LSB vs. AM-USB: stochastic



Contents

Conclusions

Neuromorphic software

- advantageous for classification of signals
  - excellent performance on AM vs. FM
  - very good even on the difficult AM-LSB vs. AM-USB case
- additive
  - can join sub-task networks to form master-network
- ROOT
  - easy to use implementation of Multi-Layer Perceptrons
  - helpful user-analysis features