Outline

- Code design for PID
  - requirements
  - proposals
- PID with EMC
  - suitable properties
  - strategy
  - results

Requirements for the PID Code

- Goal
  - quality (probability) and significance level for each particle hypothesis

- All possible kinds of particle types should be treated with the same code
  - charged particles: e, μ, π, K, p
  - neutral particles: γ, merged π^0 (opening angle of the 2 decay photons very small), electromagnetic and hadronic split-offs, . . .

Requirements for the PID Code

- Code as flexible as possible
  - detector (subsystem) specific PID
  - global PID
  - choice of different algorithms @ runtime: e.g. simple cuts on specific properties, neuronal network, likelihood method for global PID, etc.
  - proper interface to the analysis code
- Realization with oo techniques, e.g.
  - inheritance
  - polymorphism
  - Design patter, e.g. factory pattern
  - . . .

Proposals for the PID Code

Helpful Tools: Definition of particle types and sub-systems

```
class PdtPid
public:
 enum PidType
 {
  none=-1;
  electron = 0;
  muon = 1;
  pion = 2;
  kaon = 3;
  proton = 4;
  gamma =5;
   pi0 = 6;
   K0L =7;
   neutron = 8;
   splitoff =9;
}
```

```
class PidSystem
public:
 enum System
 {
  none=-1;
  mvd = 0;
  tpc = 1;
  stt = 2;
  tof = 3;
  drc = 4;
  emc =5;
  gem = 6;
  dch =7;
  track = 8;
}
```

Proposals for the PID Code

Helpful Tools: Objects for treating (combined) probabilities

```
class Consistency
public:
...
virtual double significanceLevel() const;
virtual double likelihood() const;
...
enum ConsistencyStatus {
        OK=0, noMeasure, unPhysical}
...
```

```
class ConsistencySet
public:
...
virtual bool add(PidSystem::System,
                      const Consistency&);
virtual bool add(const ConsistencySet&);
...
const Consistency* consistency
                         (PidSystem::System);
...
```

Proposals for the PID Code

Subdetector PID

```
class AbsPidInfo
public:
...
virtual const Consistency& consistency (PdtPid::PidType) const;
virtual bool setLikelihoodAlg(const std::string&)=0;
virtual bool setSignificanceAlg(const std::string&)=0;
...
protected:
virtual AbsPidAlgorithm* pidAlgorithm(const std::string&)=0;
virtual AbsSignificanceAlgorithm* sigAlgorithm(const std::string&)=0;
...
```

pure virtual factories for the choice of the algorithm

inherited classes for sub-systems

```
class DrcPidInfo: public AbsPidInfo
...
protected:
virtual AbsPidAlgorithm* pidAlgorithm(. . .);
virtual AbsSignificanceAlgorithm* sigAlgorithm(...);
...
private:
(detector specific properties)
```

```
class DrcPidInfo: public AbsPidInfo
...
protected:
virtual AbsPidAlgorithm* pidAlgorithm(. . .);
virtual AbsSignificanceAlgorithm* sigAlgorithm(...);
e:
or specific properties)
```

concrete factories for the choice of the algorithm

Proposals for the PID Code

Collection of PID Information

```
class PidInfoSummary
public:
...
virtual const AbsPidInfo*  pidInfo (PidSystem::System sys) const;
...
```

```
class PidInfoChargedSummary
: public PidInfoSummary
...
private:
TrkObject* _trkObject
...
```

```
class PidInfoNeutralSummary
: public PidInfoSummary
...
private:
EmcObject* _emcObject;
...
```

Proposals for the PID Code

Interface to analysis and global PID

- RhoCandidate holds reference to PidInfoSummary and to an abstract (global) PID algorithm

```
class RhoCandidate
public:
...
const PidInfoSummary* pidInfoSummary();
RhoAbsPidAlgorithm* pidAlgorithm();
const Consistency* consistency(PdtPid::PidType);
...
```

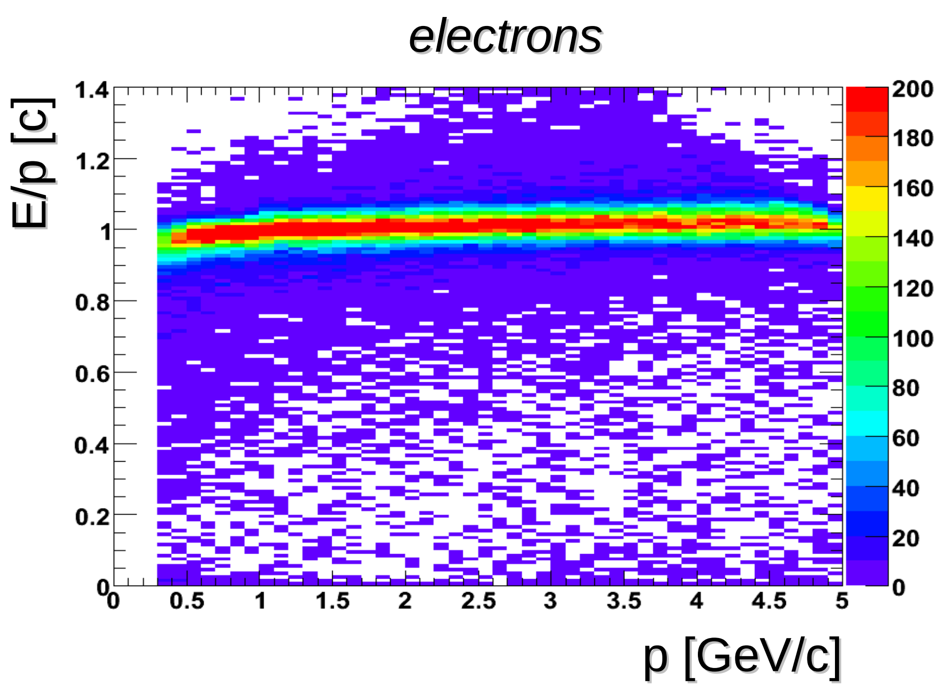
calculation based on the chosen PID algorithm

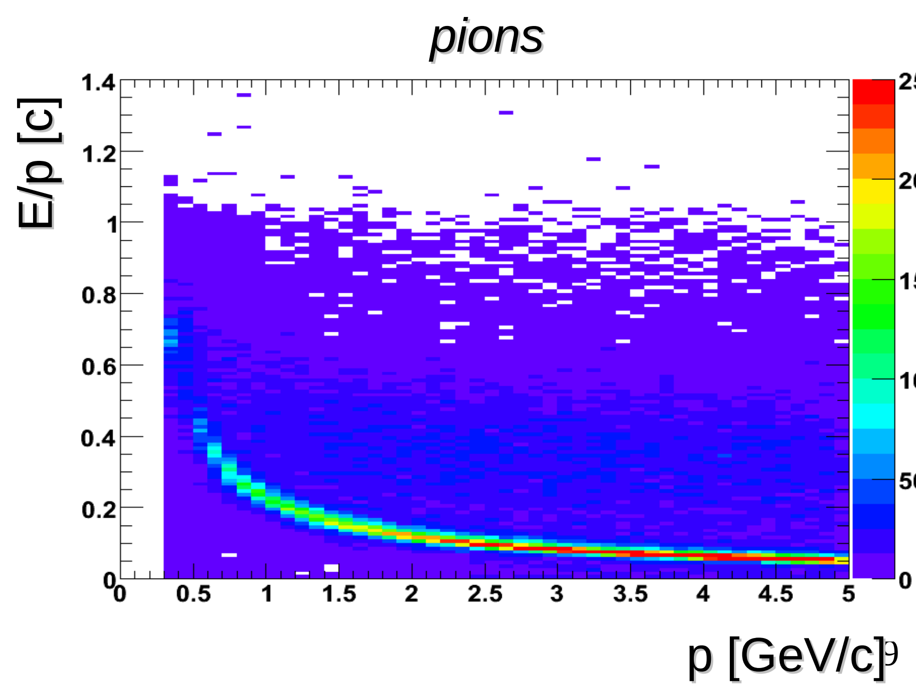
Open Questions

- What to persist on which data level (AOD, ESD, ...) ?

EMC PID: Suitable Properties

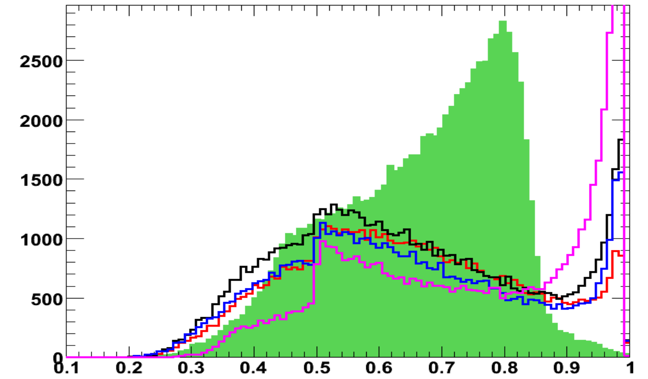
- Most important property: $E_{cluster} / p$
  - p: reconstructed momentum
  - $E_{cluster}$: energy deposit of the charge particle
  - electrons deposit the full kinetic energy -> $E_{cluster} / p \sim 1$
  - hadrons/muons deposit in only a fraction of their energy: $E_{cluster} / p < 1$

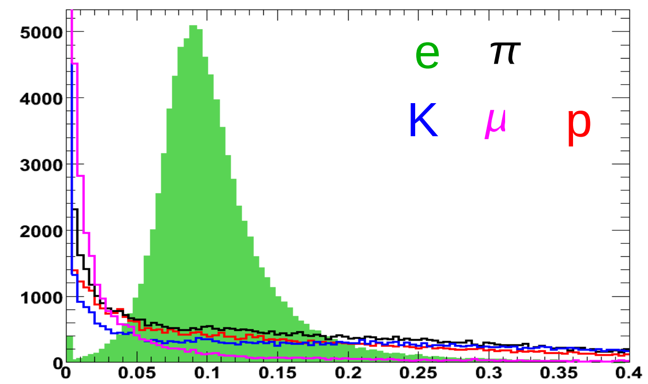


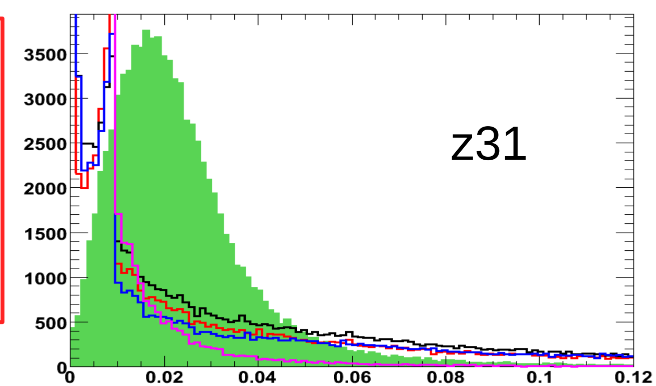


EMC PID: Suitable Properties

- Other suitable properties: energy distribution within the cluster -> shower shape, examples:
  - E_1/E_9
  - lateral distribution
  - Zernike moments







e: largest fraction contained only in few crystals
hadrons: energy distribution less concentrated for the same energy
-> differences reflected in the shower shape

Zernike moments:
polynomial expansion of the energy distribution within the cluster towards radial- and angular-dependent parts.
Analogy: spherical harmonic functions

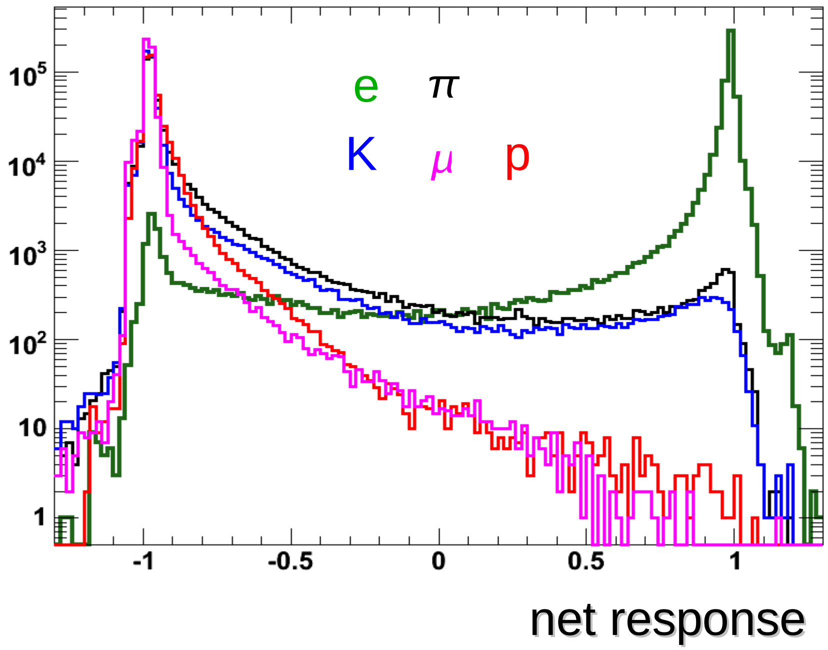
EMC PID: Strategy

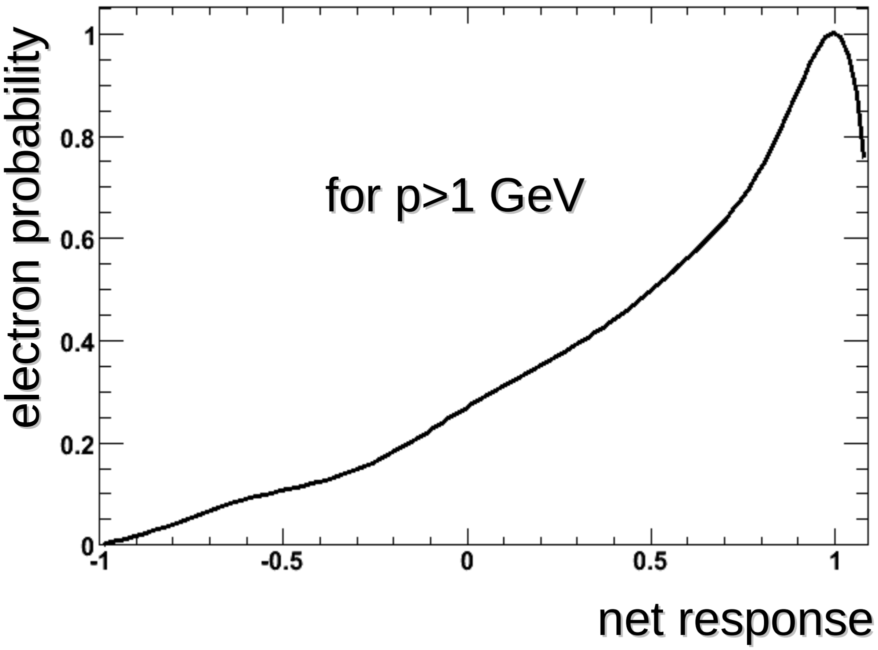
- Problem
  - lots of properties suitable for PID with EMC
  - question: how to find optimal cut parameters in a multi-dimension space?
  - possible solution: neuronal network

- PB software: training of a multilayer perceptron (MLP)
  - training files: 10^7 tracks for e, π, μ, K and p each (p: 0.2-15 GeV/c, homogeneous in $\cos(\Theta)$ and ϕ)
  - 10 input parameters: E/p, E_1/E_9, E_9/E_{25} , lateral distribution,
    6 different Zernike moments
  - net response: " 1" for good tracks (electrons)
    "-1" for bad tracks (π, μ, K, p)

EMC PID: Results

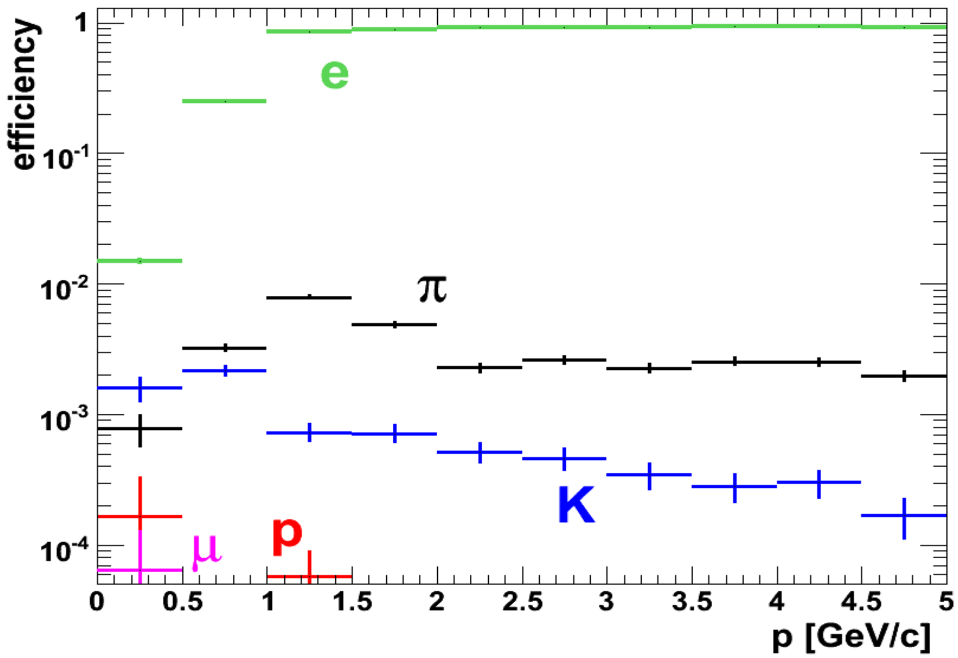
- Test sample
  - 10^6 tracks for e, π, μ, K and p each
- Good distinction between e and other particle species with MLP
- Global PID: correlation net response <-> probability for particle hypothesis





EMC PID: Results

EMC: electron LH>95%



Global PID: electron LH>99.8%

MVD, STT, Drc, EMC, Muon