New MVA PID code in larsoft

Martin Haigh, University of Warwick

Overview and purpose of software

- New code will provide MVA-based particle identification for recob::Track objects.
  - See talk by Nick Grant at DUNE Texas meeting here.
  - Will also be extended to do PID for recob::Shower objects.
  - Output is the variables used for MVA input, and a map containing the MVA value for each MVA run. May also calculate a likelihood value later.

- Also add an art data object anab::MVAResult to store the output of the module.
  - Rationale – existing PID object is specialised for existing code which is based on a Chi-squared method. Does not contain any of the necessary fields for our PID.

- Include .fcl file to run PID by itself for now; when stable PID will be added to normal reco or analysis processing.

- Include shell script which will steer training of MVA for a set of signal and background reco files (goes into scripts directory in larana install).

Use case

- Code based on ROOT's TMVA library. 3-step process to go from existing recon files to running real PID with MVA.
  - First need to train MVA methods. Run runPID.fcl over single-particle recon files with no MVA methods defined, to get an ntuple with MVA input variables.
  - Run a script TrainMVA.sh which generates MVA .xml weight files based on a list of signal and background files. Based on ROOT macro since it is difficult to run TMVA training from inside of art.
  - Can then run full PID over any data files, with .fcl pointing to generated weight files.

- We will provide canonical weight files for general use.
  - => normal users only need to worry about final step.
  - Users with particular analysis requirements for PID can generate weight files for their specific case of signal and background.

- Weight files are .xml files up to ~1MB in size (can be significantly compressed). Where to put these?

Details of code changes

- New module MVAPID_module.cc, and support class MVAAlg containing bulk of code, in larana/ParticleIdentification.
- Training script TrainMVA.C, and .fcl file runPID.fcl, in larana/ParticleIdentification/scripts (.fcl file is installed to job folder).
- Data struct MVAResult in lardata/AnalysisBase. Changes to classes.h, classes_def.xml to build reflex dictionaries for this.
- Small changes to CMakeLists files so that scripts get installed correctly, and calorimetry library is available to PID module.
- Will not affect existing processing until PID is added to .fcl files.

Planned future changes

- Alter MVAAlg soon to allow processing of shower objects.
- Alter MVAAlg when changes are made to PID logic.
  - New variables, change in how existing variables are calculated…
- New version of MVAResult needed when new variables are added.
- Probable changes to runPID.fcl to allow settings to be tuned in future.
- Again, Will not affect existing processing until PID is added to .fcl files.

Code detail

.fcl for PID module:

```
 producers:{
  pid: {
   module_type:      MVAPID
   CalModuleName:    pandoracalo                              #Currently run calorimetry
   CalAmpConstants: [0.9033e-3, 1.0287e-3, 0.8800e-3]         #ourselves. Need to move to
   CalAreaConstants:[5.1822e-3, 5.2682e-3, 5.3962e-3]         #using output of calo module
   CaloUseModBox:    true                                     #if possible
   MVAMethods:       [ "ANN","BDT" ]
   WeightFiles:      [ "MuEMVA_ANN.weights.xml",              #specify labels to give MVA
                       "MuEMVA_BDT.weights.xml" ]             #output and locations of weight
 }                                                            #files. Produce training ntuples
}                                                             #by making these lists empty.
```

MVAResult object:

```
struct MVAResult {     #Need to move from struct to using getters/setters
  float evalRatio, concentration, coreHaloRatio, conicalness; #Variables used as input
  float dEdxStart, dEdxEnd, dEdxPenultimate;                  #to MVA
  float nSpacePoints;
  unsigned int trackID;                     #Enable mapping to tracks without art::Assn
  std::map<std::string,double> mvaOutput;   #Map names of MVA methods to output of MVA
};
```