Boosted Decision Trees

A modern method of data analysis

Christian Autermann
University of Hamburg

- Decision Trees
- "Boosting"
- ν_e – ID at MinibooNE
- Evidence for single-top Production at DØ

Boosted Decision Trees References

- Talk from B. Roe, U. Michigan (MiniBooNE)
- Studies of Boosted Decision Trees for MiniBooNE Particle Identification, H.-J. Yang, B. Roe, J. Zhu, arXiv:physics/0508045
- Wine&Cheese Talk, D. O'Neil, Evidence for Single Top at DØ

→ used for this talk

R.E. Schapire ``The strength of weak learnability.'' *Machine Learning* **5** (2), 197-227 (1990). First suggested the boosting approach for 3 trees taking a majority vote

Y. Freund, ``Boosting a weak learning algorithm by majority'', *Information and Computation* **121** (2), 256-285 (1995) Introduced using many trees

Y. Freund and R.E. Schapire, ``Experiments with an new boosting algorithm, *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kauffman, SanFrancisco, pp.148-156 (1996). Introduced AdaBoost

J. Friedman, T. Hastie, and R. Tibshirani, ``Additive logistic regression: a statistical view of boosting'', *Annals of Statistics* **28** (2), 337-407 (2000). Showed that AdaBoost could be looked at as successive approximations to a maximum likelihood solution.

T. Hastie, R. Tibshirani, and J. Friedman, ``The Elements of Statistical Learning'' Springer (2001). Good reference for decision trees and boosting.

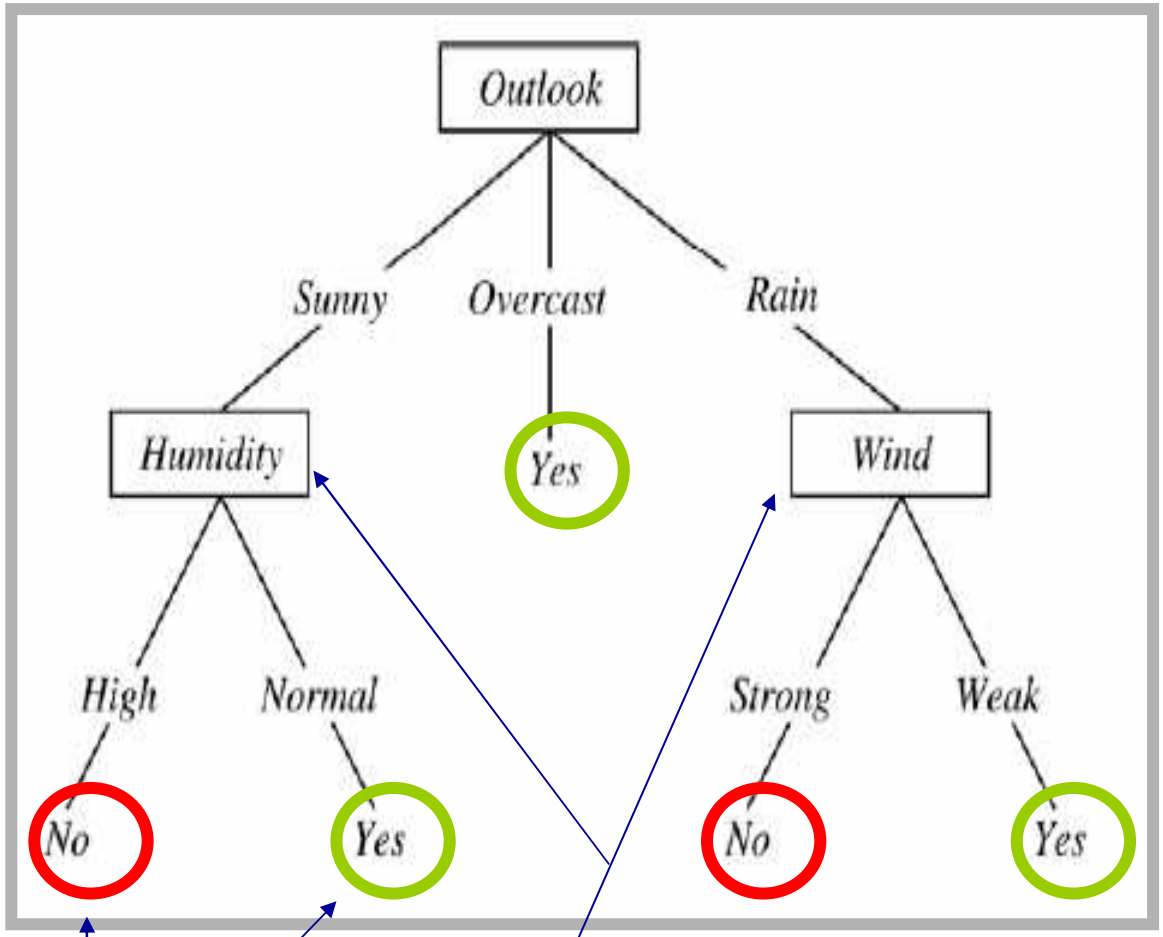
Play Tennis?

The manager of a tennis court wants to optimize the personnel costs:

Number of players ↔ weather conditions; but how ?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree: “Play Tennis ?”



- Many possibilities to build a tree
- No real „rules“
- Criteria like *Outlook*, *Wind, etc.* can be used more than once in a single tree

background like leaf

signal like leaf

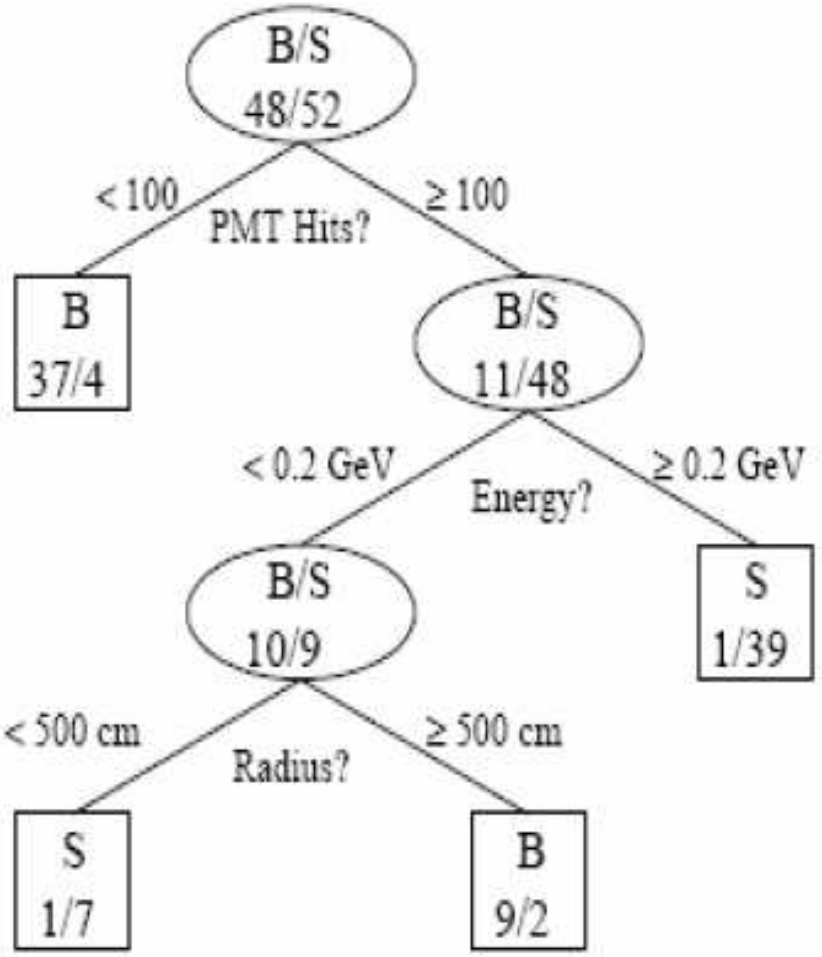
“leaf”

“node”

Some Physics Example

- Go through all variables and find best variable and value to split events.
- For each of the two subsets repeat the process
- Proceeding in this way a tree is built.
- Ending nodes are called leaves.

Background/Signal



(~ ν_e – ID at MiniBooNE)

Optimizing the Decision Tree (1)

Very important to optimize the DT consistently and automatically

Equivalent to optimizing "cuts"

- Assume an equal weight of signal and background training events.
- If more than ½ of the weight of a leaf corresponds to signal, it is a signal leaf; otherwise it is a background leaf.
- Signal events on a background leaf or background events on a signal leaf are misclassified.

Optimizing the Decision Tree (2)

Criterion for "Best" Split

- Signal Purity, P, is the fraction of the weight of a leaf due to signal events.
- Gini: Note that Gini is 0 for all signal or all background. W_i is the weight of event "i".

$$G_{ini} = \left(\sum_{i=1}^{n} W_i \right) P(1-P)$$

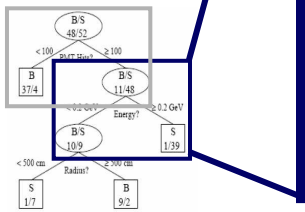
- The criterion is to minimize gini_left + gini_right of the two children from a parent node

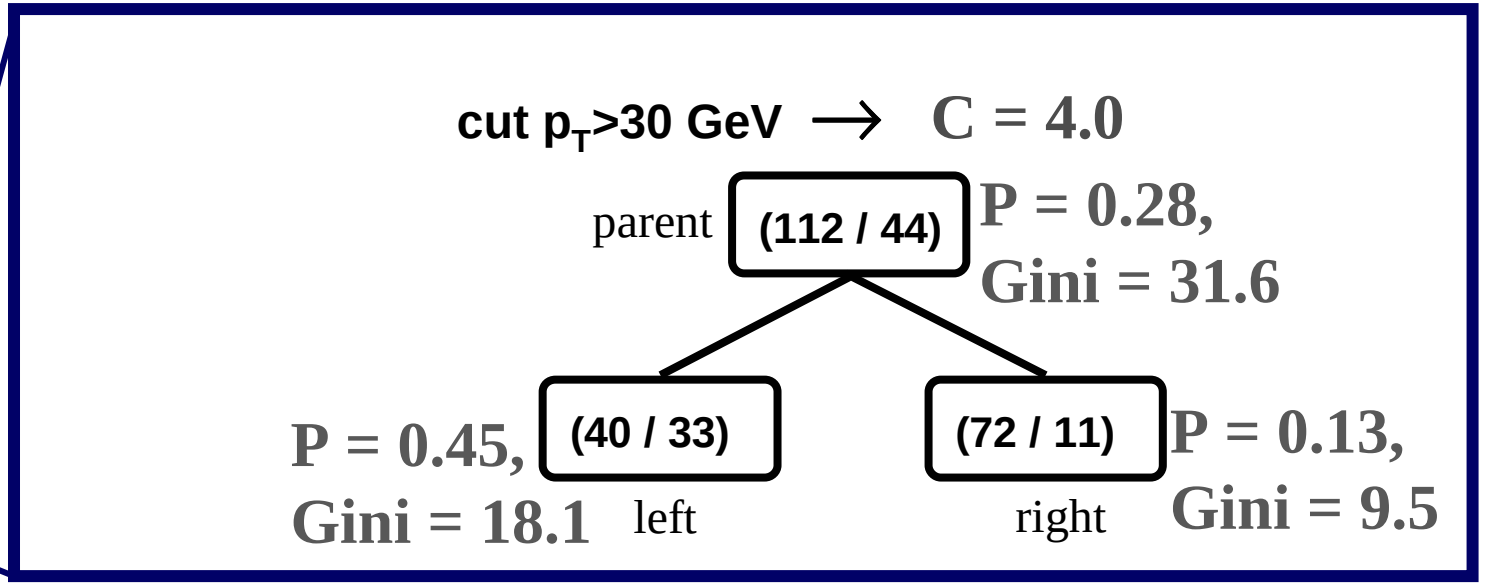
Optimizing the Decision Tree (3)

Criterion for Best Split

- Pick the branch to maximize the change in gini.
- Criterion C = $Gini_{parent}$ – $Gini_{right\text{-}child}$ –$Gini_{left\text{-}child}$

Example:



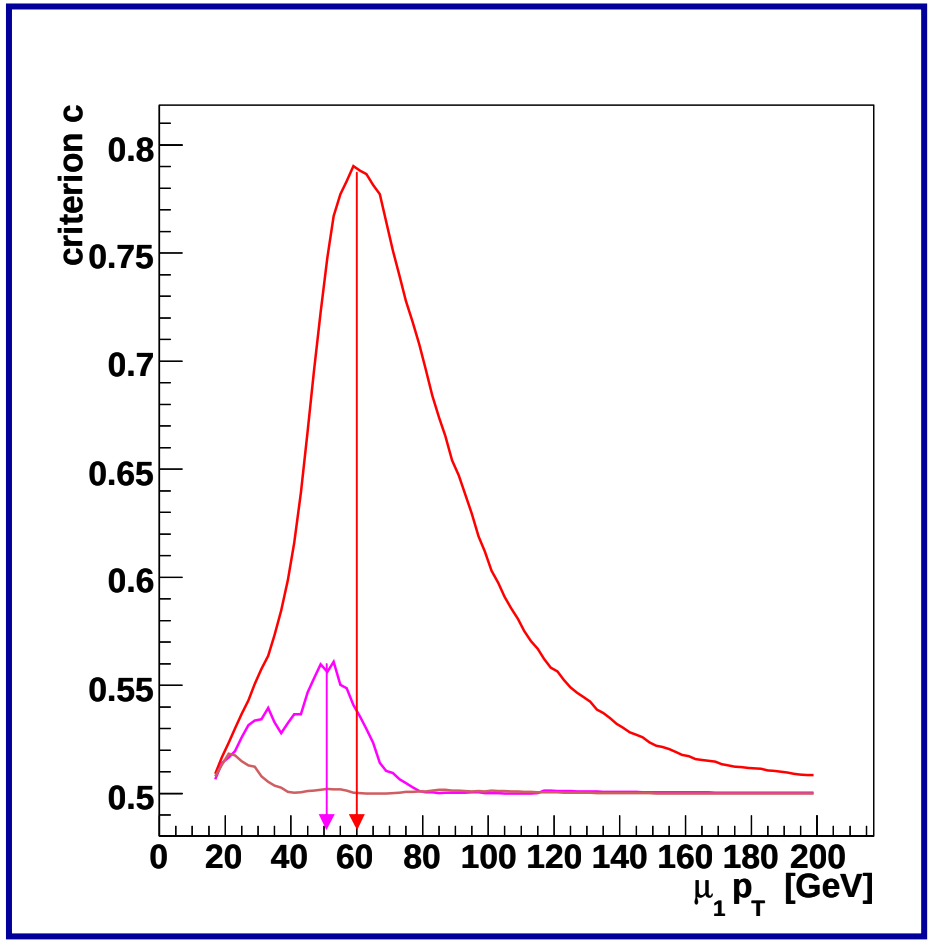


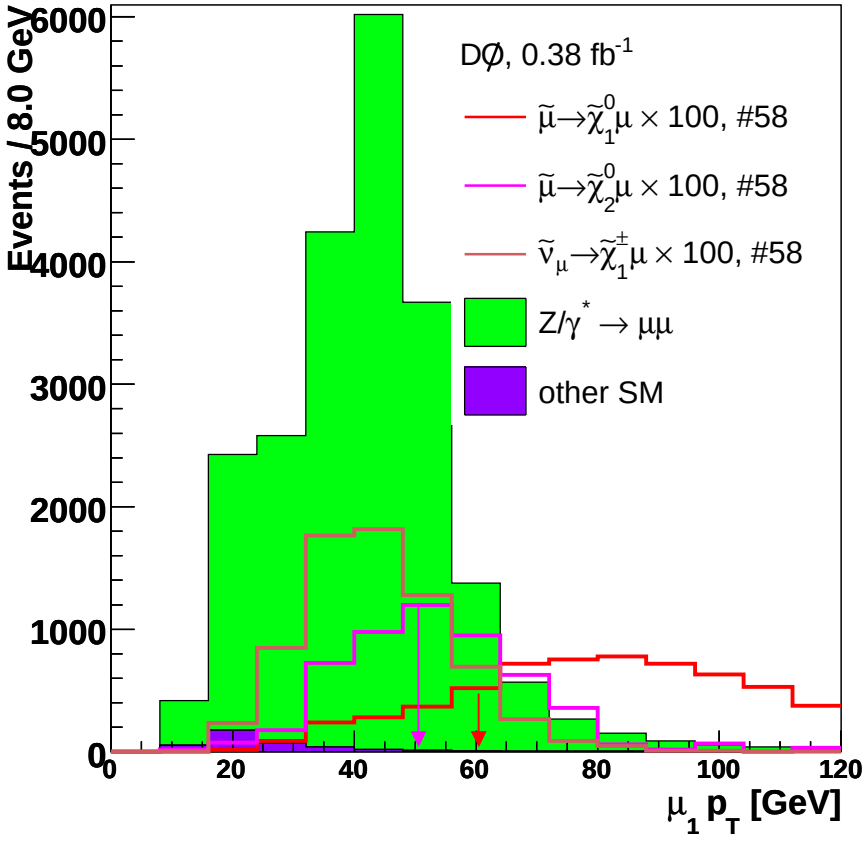
- Optimize each node (e.g. p_T>30 GeV) by maximizing “C”.

Optimizing the Decision Tree (4)

Example: (Resonant Sleptons)

Variable: pT(1. muon)





Decision Trees: Advantages

- Already known for quite some time
- Easy to understand
- We understand (at each node) how a decision is derived, in contrast to Neural Nets

- As simple as a conventional cut approach
- No event is removed completely, no information is lost

Decision Trees: Disadvantages

- Decision trees are not stable,
  a small change/fluctuation in the data
  can make a large difference!
  This is also known from Neural Nets.

- Solution: Boosting!
  $\rightarrow$ Boosted Decision Trees

Overview

- Decision Trees
- “Boosting”
- ν_e – ID at MinibooNE
- Evidence for single-top Production at DØ

Boosting

- Idea: Built many trees O(1000), so that the weighted average over all trees is insensitive to fluctuations
- Impossible to build & weight 1000 trees by hand. How to do it automatically?

→ Reweight misclassified events
(i.e. signal events on a background leaf and background events on a signal leaf)
→ Build & optimize new tree with reweighted events
→ Score each tree
→ Average over all trees, using the tree-scores as weights

Boosting

Several Boosting Algorithms for changing weights of misclassified events:

1. AdaBoost (Adaptive Boosting)
2. Epsilon boost ("shrinkage")
3. Epsilon-Logit Boost
4. Epsilon-Hinge Boost
5. Logit Boost
6. Gentle AdaBoost
7. Real AdaBoost
8. ...

} covered here

Definitions

W_i = weight of event *i*

Y_i = +1 if event *i* is signal,
–1 if background

$T_m(x_i)$ = +1 if event *i* lands on a signal leaf of tree *m*
–1 if the event lands on a background leaf.

Goal: $Y_i = T_m(x_i)$

misclassified event: $Y_i \neq T_m(x_i)$

AdaBoost

Define err_m = weight wrong/total weight for tree *m*

$$err_m = \sum_{Y_i \neq T_m(x_i)} W_i$$

α_m is the score (weight) of tree *m*

$$\alpha_m = \beta \cdot \ln \frac{1 - err_m}{err_m} \qquad (ß: \text{constant})$$

Increase weight for misidentified events:

$$W_i \rightarrow W_i \cdot e^{\alpha_m}$$

Renormalize all events: $w_i \rightarrow w_i / \sum_{i=1}^{N} w_i$

Scoring events with AdaBoost

m trees have been optimized and scored (α_m) with Monte Carlo training events. (Knowledge of Y_i was necessary.)

Now, score test-sample and data by summing over trees

$$T(x_i) = \sum_{m=1}^{N_{tree}} \alpha_m T_m(x_i)$$

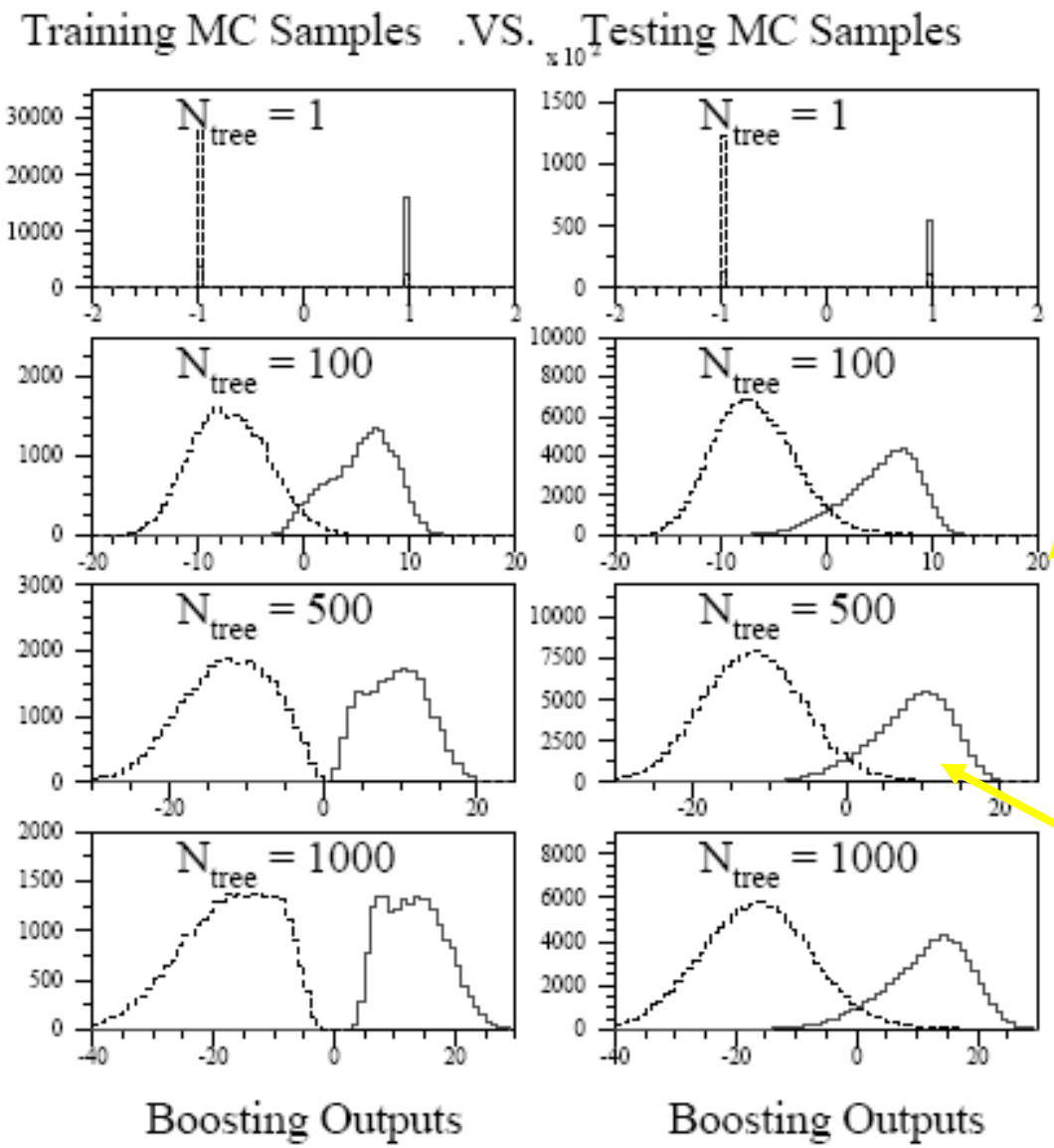
α_m score/weight for each tree *m*

$T_m(x_i)$ = +1 event lands on a signal leaf
– 1event lands on a backgd leaf

$T(x_i)$ final BDT output variable

equivalent to NN output

MiniBooNE Training / Test Samples



Scores increase with Ntree since

$$T(x) = \sum_{m=1}^{N_{tree}} \alpha_m T_m(x)$$

High Purity

Epsilon Boost (shrinkage)

- After tree *m*, change weight of misclassified events, typical $\varepsilon \sim O(0.01)$:

$$w_i \rightarrow w_i \exp(2\varepsilon)$$

- Renormalize weights

$$w_i \rightarrow w_i / \sum_{i=1}^{N} w_i$$

Training Sample

Test Sample & Data

- Score by summing over trees

$$T(x) = \sum_{m=1}^{N_{tree}} \varepsilon T_m(x)$$

- Here: all trees have same score/weight

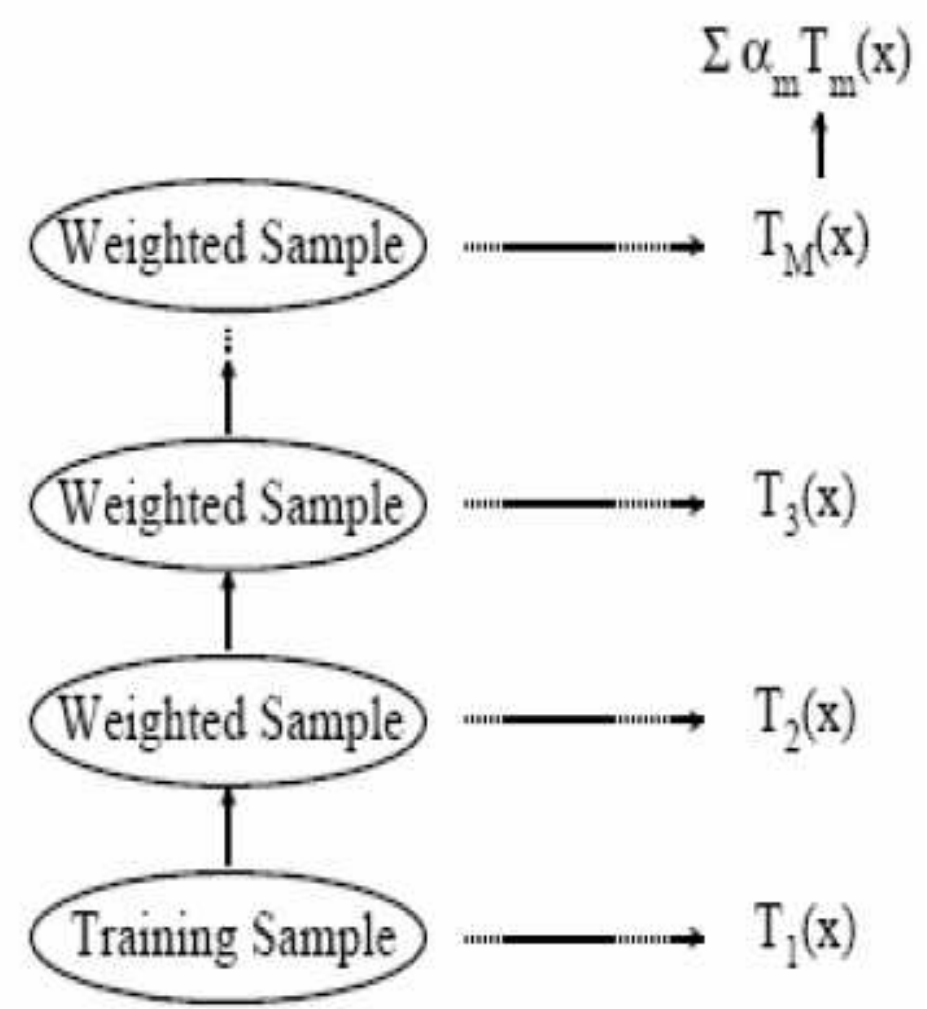
Example

- AdaBoost: Suppose the weighted error rate is 40%, i.e., err_w= 0.4 and ß = 0.5
- Then α = (1/2)ln((1-.4)/.4)= .203
- Weight of a misclassified event is multiplied by exp(0.203) = 1.225

- Epsilon boost: The weight of wrong events is increased by exp(2*.01) = 1.02
- always the same factor

Boosting Summary

- Give the training events misclassified under this procedure a higher weight.
- Continuing build perhaps 1000 trees and do a weighted average of the results (1 if signal leaf, -1 if background leaf).



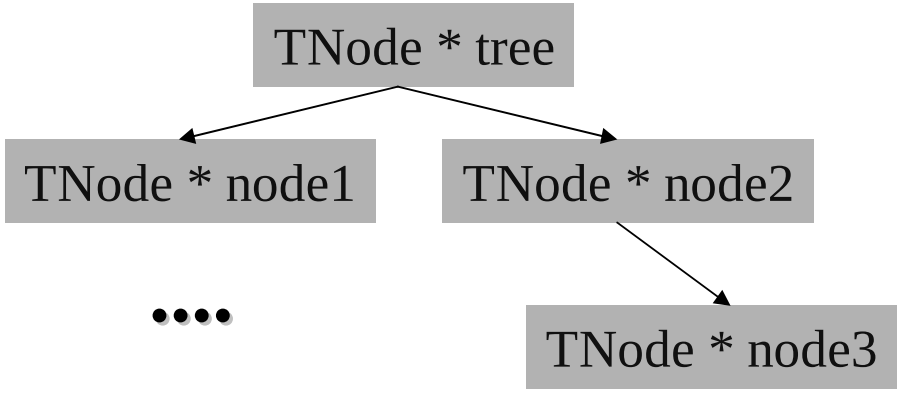
Coding* 1/2

* Not necessarily the best way to do it

```
class TNode {
  TNode * left_child
  TNode * right_child
  int          cut
  double       value
  bool *       passed()
}
```

```
class TEvent {
  vector<double>
      variable_value
  double weight
  int       type
}
```

Build tree:



Fill all events (signal, background, data) in:

vector<TEvent *> AllEvents

Split Samples:

vector<TEvent *> Training

vector<TEvent *> Test

Coding 2/2

The following functions are needed:

TNode* OptimizeNode (TNode*, vector<TEvent*>)
double ScoreTree (TNode*, vector<TEvent*>)
void ReweightMisclassified (TNode*, vector<TEvent*>&, "AdaBoost")

double ScoreEvent (vector<TNode*>, TEvent*)

Now it's easy to put a BDT program together:

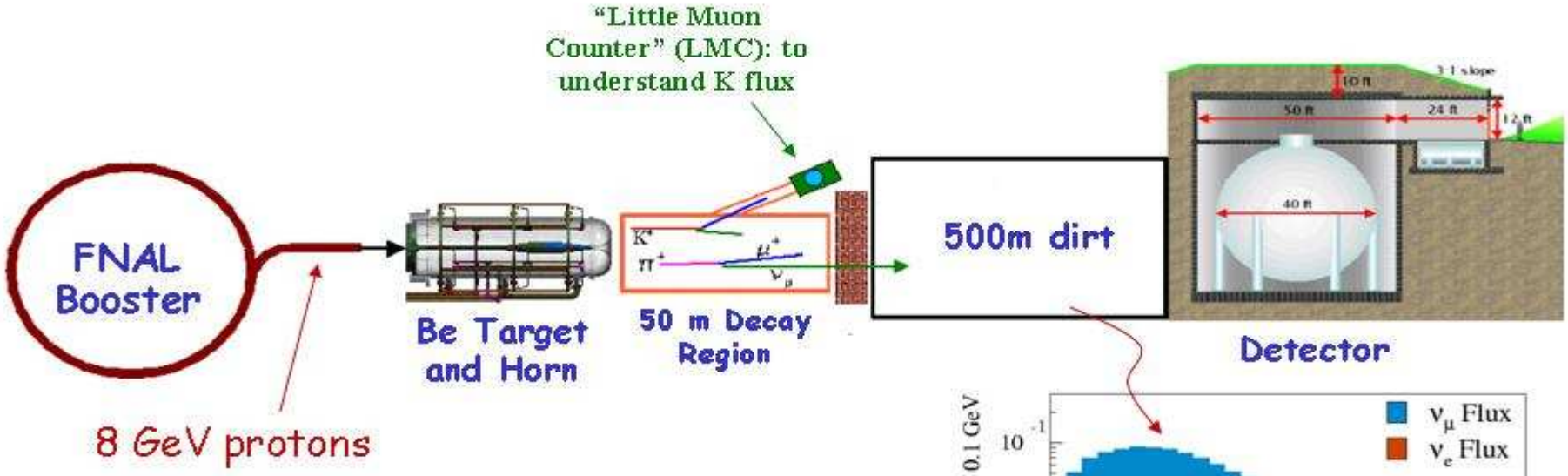
```
// Training
loop over 1000 Decision trees: {
   NewTree = OptimizeNode( OldTree, TrainingSample) //Optimization off all cuts in one tree
   TreeScores->push_back( ScoreTree( NewTree, TrainingSample)  ) //Calculate α_m
   Trees->push_back( NewTree )  //vector<TNode*> containing all 1000 optimized Decision Trees
   ReweightMisclassified( NewTree, TrainingSample, "AdaBoost" ) //Boosting
   OldTree = NewTree
}
// TestSample and Data:
for (vector<TEvent*>::iterator it = TestSample->begin(); it!=TestSample->end(); it++)
    hist_bdt_output->Fill( ScoreEvent(Trees, it) )
```

Overview

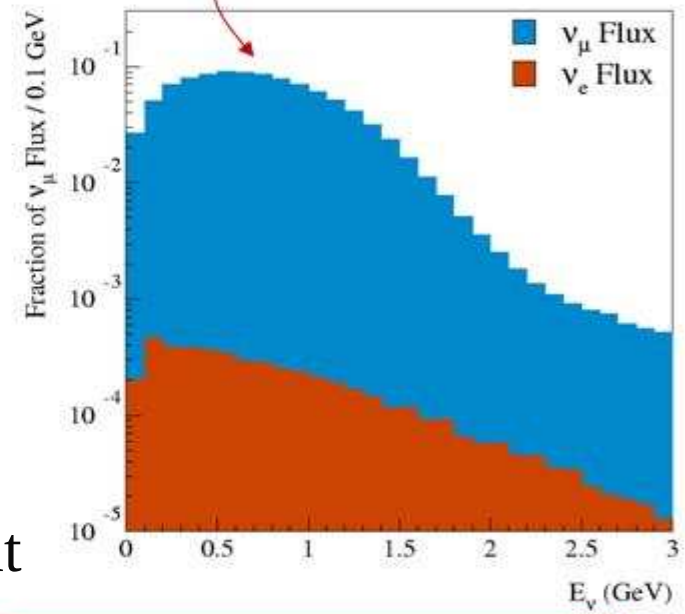
- Decision Trees
- “Boosting”
- ν_e – ID at MinibooNE
- Evidence for single-top Production at DØ

MiniBooNE (Fermilab)

Neutrino Beam

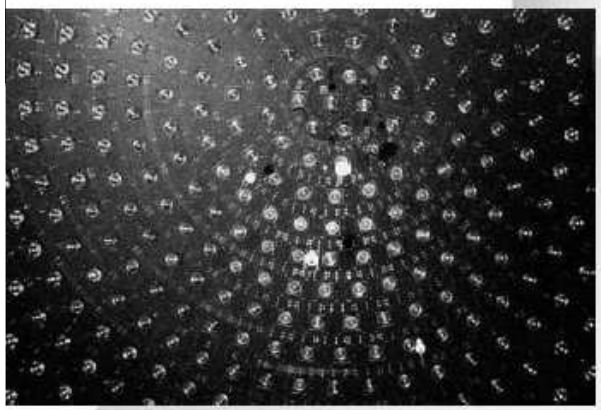


- Proton flux ~ 6E16 p/hr (goal 9E16 p/hr)
  - ~ 1 detected neutrino/minute
  - L/E ~ 1



BooNE: Booster Neutrino Experiment

The MiniBooNE Detector

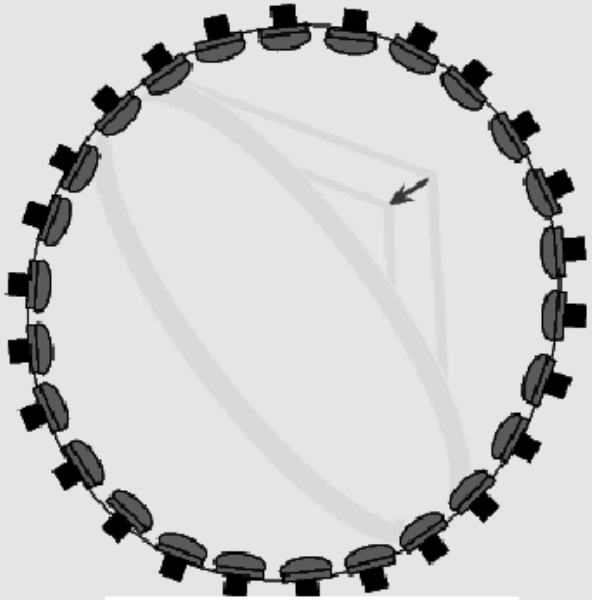


- 12 meter diameter sphere
- Filled with 950,000 liters (900 tons) of very pure mineral oil
- Light tight inner region with 1280 photomultiplier tubes
- Outer veto region with 241 PMTs.
- Oscillation Search Method:

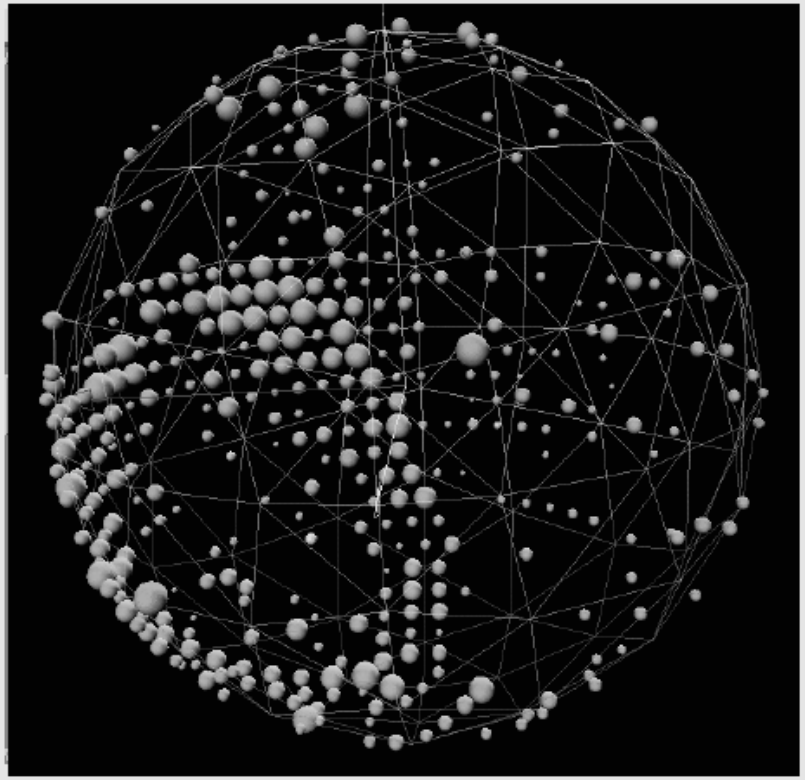
  Look for ν_e events in a pure ν_μ beam

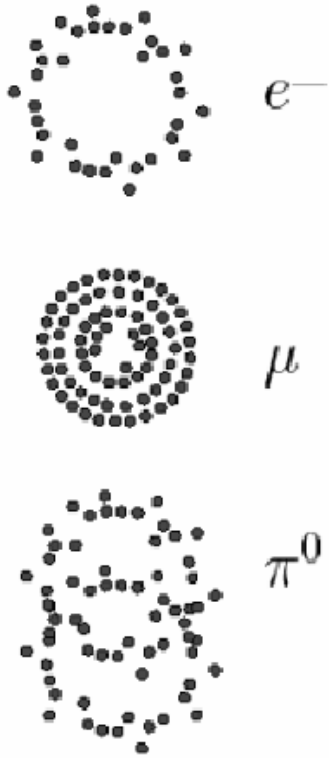
Particle Identification

- Separation of ν_μ from ν_e events
  - Exiting ν_μ events fire the veto
  - Stopping ν_μ events have a Michel electron after a few μsec
  - Also, scintillation light with longer time constant $\Rightarrow$ enhanced for slow pions and protons
  - Čerenkov rings from outgoing particles
    - Shows up as a ring of hits in the phototubes mounted inside the MiniBooNE sphere
    - Pattern of phototube hits tells the particle type

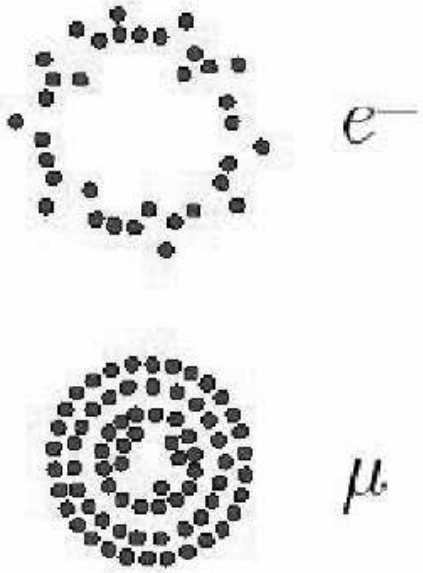


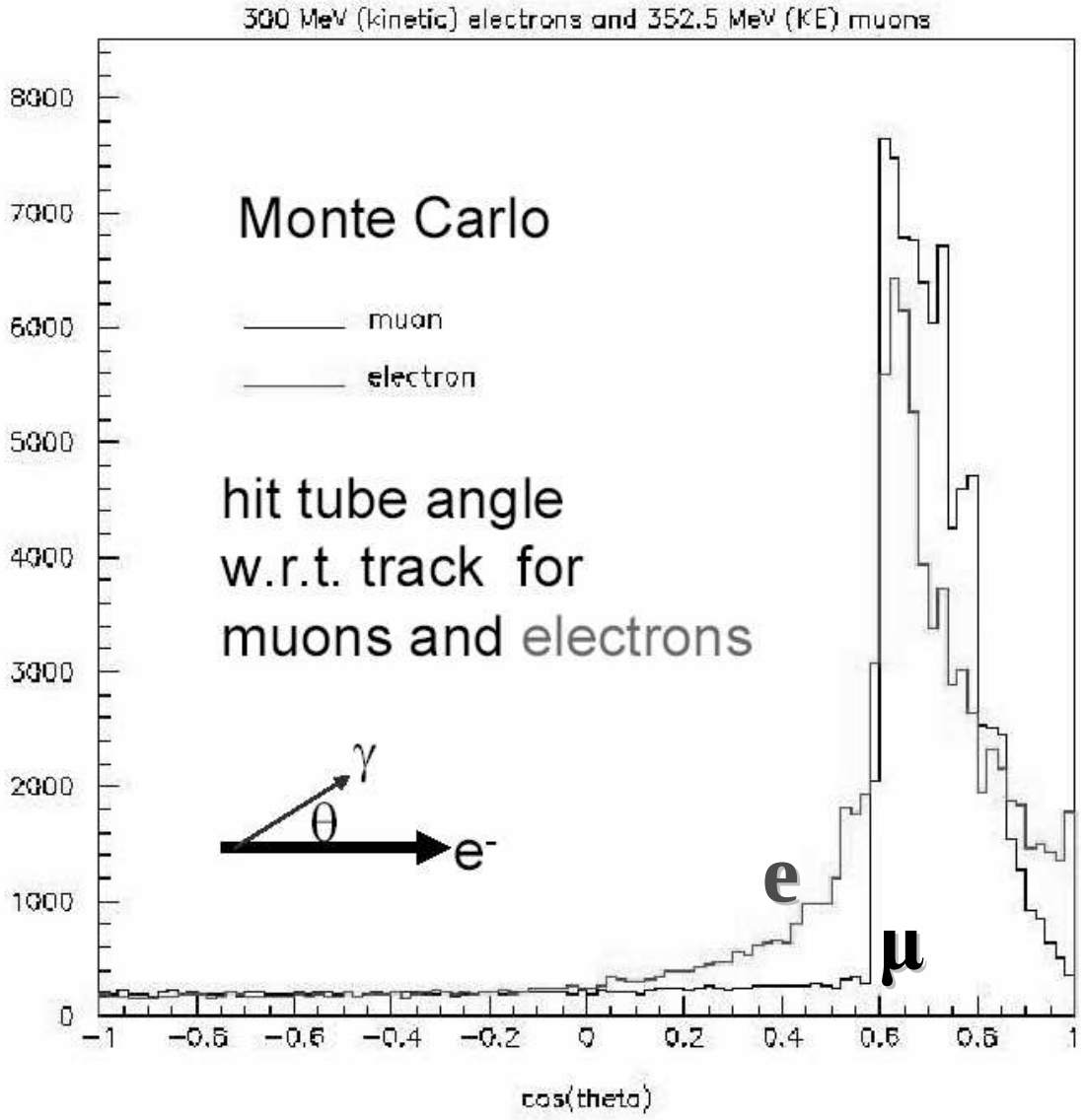
Stopping muon event





electrons are fuzzy
muons are sharp



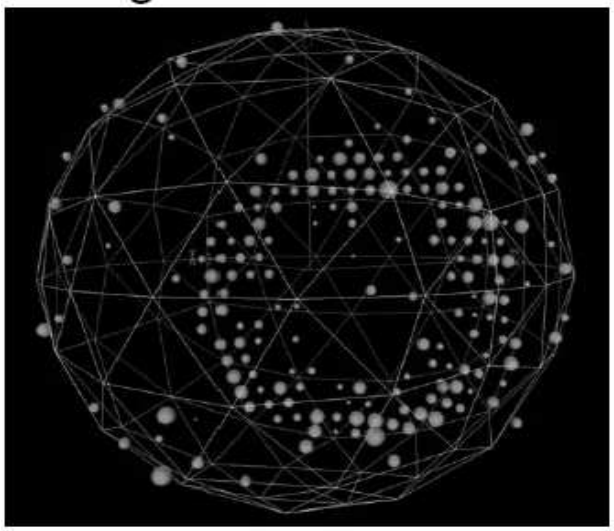


Examples of data events

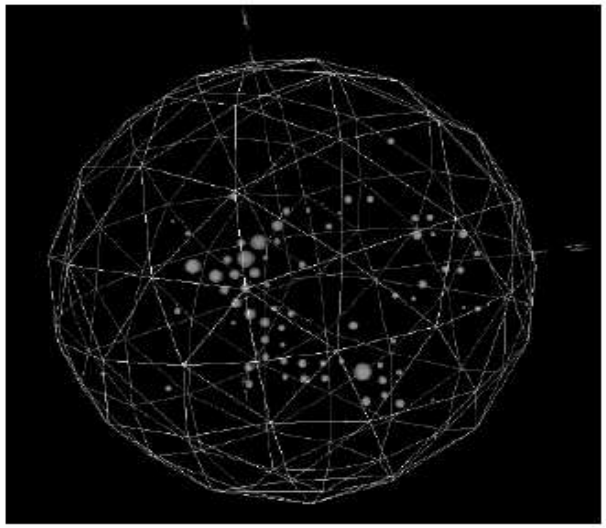
Pattern of hit tubes (with charge and time information)
allows reconstruction of track location and direction
and separation of different event types.

e.g. candidate events:

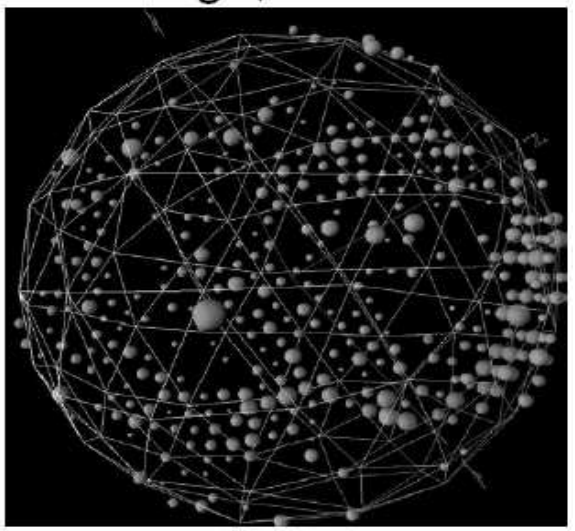
size = charge, color = time



muon
from ν_μ interaction



Michel electron
from stopped μ decay
after ν_μ interaction



$\pi^0 \rightarrow$ two photons
from ν_μ interaction

Numerical Results

- There are 2 reconstruction-particle id packages used in MiniBooNE
- The best results for ANN (artificial neural nets) and Boosting used different numbers of variables, 21 or 22 being best for ANN and 50-52 for boosting
- Only relative results are shown
- Results quoted are ratios of background kept by ANN to background kept for boosting, for a given fraction of signal events kept

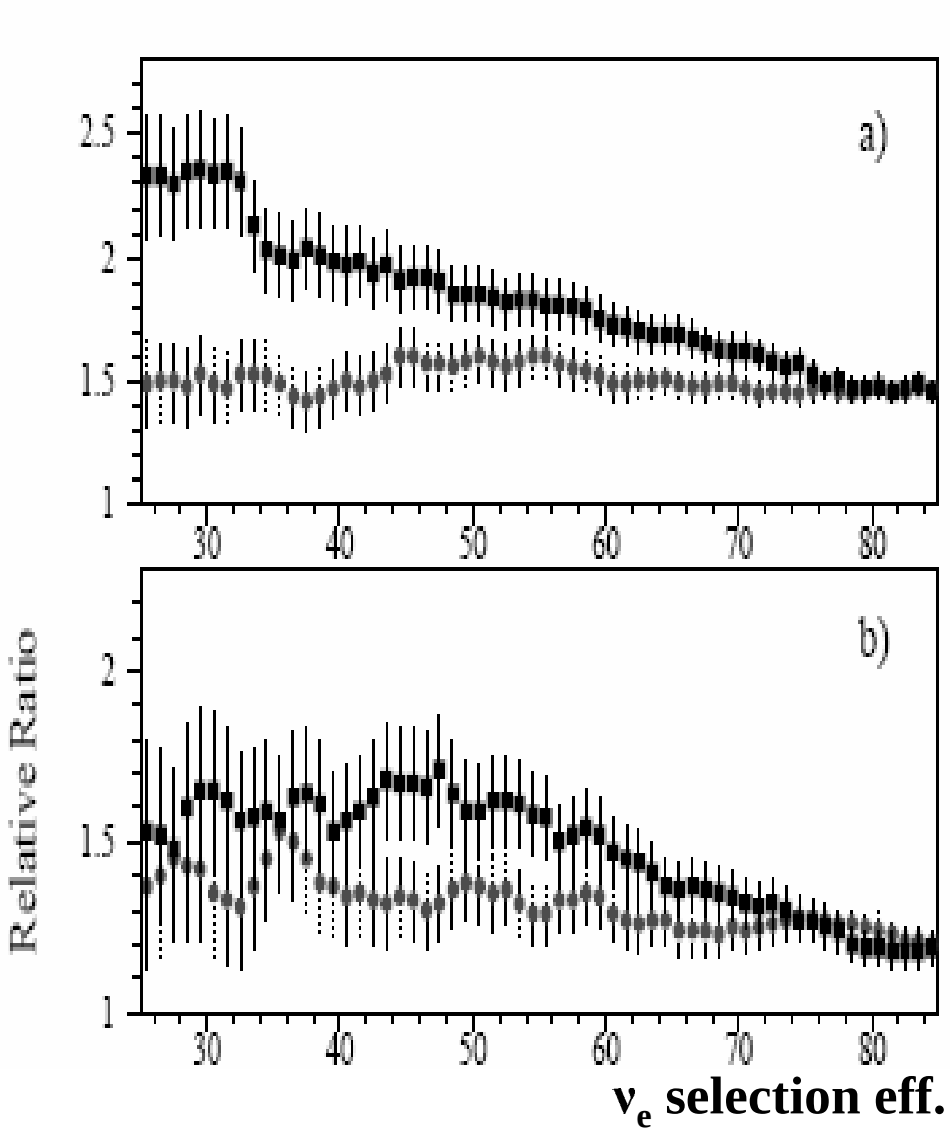
$\varepsilon<1$: ANN better than BDT

$\varepsilon>1$: BDT better

Comparison of Boosting and ANN

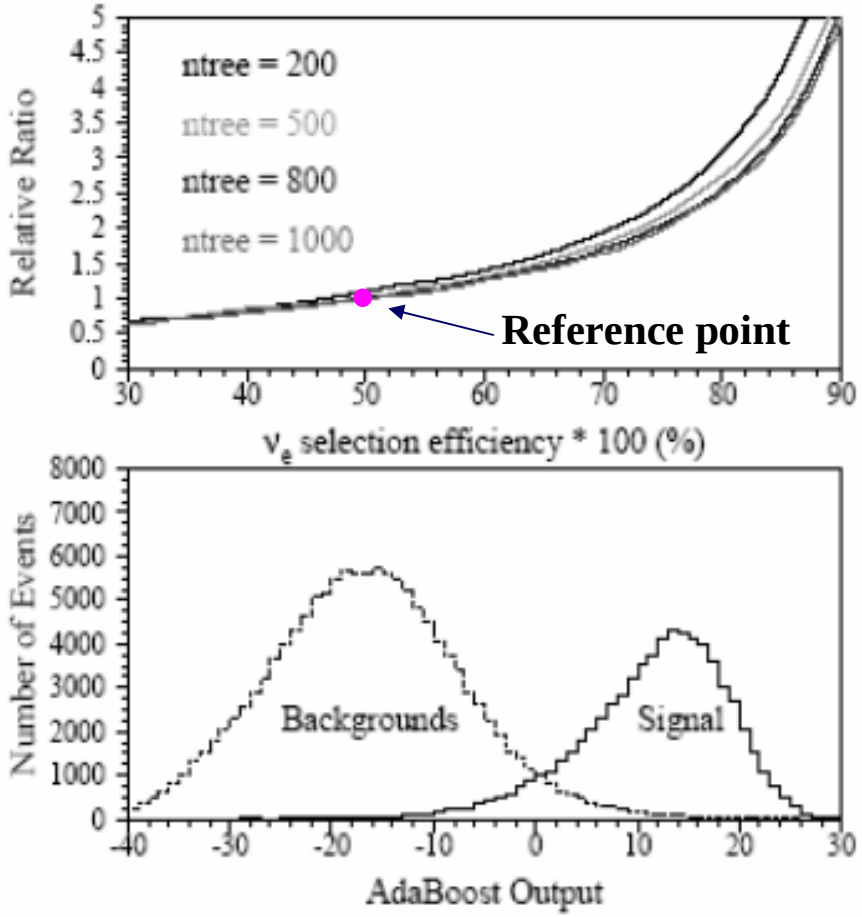
Relative ratio is
ANN bkrd kept /
Boosting bkrd kept

- a. Bkrd are cocktail events.
  Red is 21 and black is 52 training var.
- b. Bkrd are pi0 events. Red is 22 and black is 52 training var.



Boosting Results versus Ntree

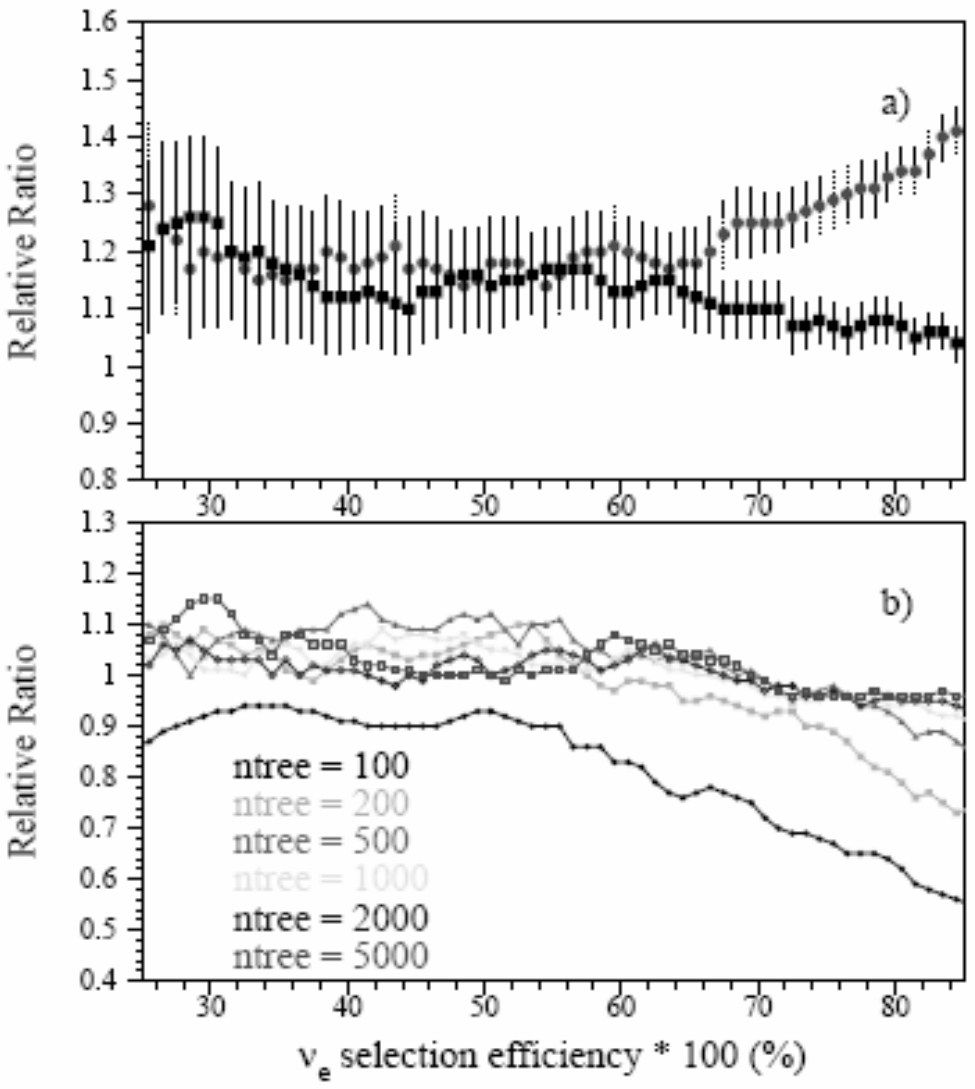
- Top: N_bkrd divided by N_bkrd for 1000 trees and 50% nue selection efficiency vs nue efficiency for CCQE events.
- Bottom: AdaBoost output for background and signal events for 1000 trees



AdaBoost vs. Epsilon Boost

- 8 leaves/ 45 leaves.
  Red is AdaBoost,
  Black is Epsilon Boost

- AdaBoost/ Epsilon Boost (Nleaves = 45).



For Neural Nets (ANN)

- **For ANN one needs to set temperature, hidden layer size, learning rate… There are lots of parameters to tune.**
- For ANN if one
    a. Multiplies a variable by a constant, var(17)→ 2.var(17)
    b. Switches two variables var(17)←→var(18)
    c. Puts a variable in twice
- The result is very likely to change.

For Boosting

- Boosting can handle more variables than ANN; it will use what it needs.
- Duplication or switching of variables will not affect boosting results.
- Suppose we make a change of variables y=f(x), such that if x_2>x_1, then y_2>y_1. The boosting results are unchanged. They depend only on the ordering of the events
- There is considerably less tuning for boosting than for ANN.

Robustness

- For either boosting or ANN, it is important to know how robust the method is, i.e. will small changes in the model produce large changes in output.

- In mini-BooNE this is handled by generating many sets of events with parameters varied by about 1 sigma and checking on the differences. This is not complete, but, so far, the selections look quite robust.

Summary MiniBooNE

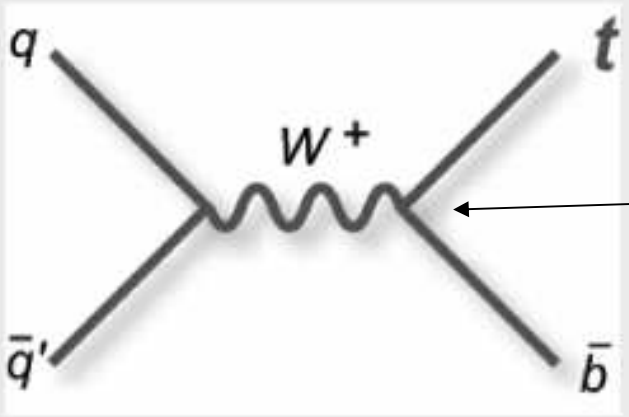
- **For MiniBooNE boosting is better than ANN by a factor of 1.2—1.8**
- **AdaBoost and Epsilon Boost give comparable results within the region of interest (40%--60% nue kept)**
- **Use of a larger number of leaves (45) gives 10--20% better performance than use of a small number (8).**
- **Preprint Physics/0408124; N.I.M., in press**
- **C++ and FORTRAN versions of the boost program (including a manual) are available on homepage:**
- **http://www.gallatin.physics.lsa.umich.edu/~roe/**

Overview

- Decision Trees
- “Boosting”
- ν_e – ID at MinibooNE
- Evidence for single-top Production at DØ

Single Top Production

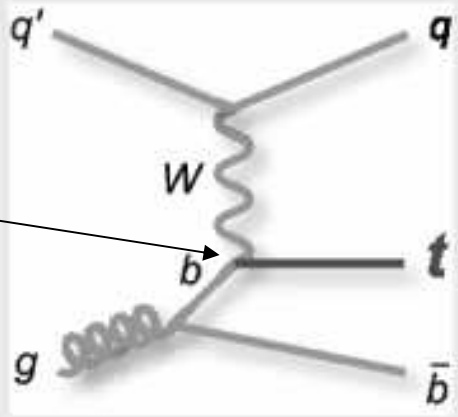
s-channel (tb)



- $\sigma_{NLO} = 0.88 \pm 0.11$ pb(*)
- current limits (95% C.L.):
  Run II DØ: < 5.0 pb (370pb^{-1})
  Run II CDF: < 3.1 pb (700pb^{-1})

$\sim|V_{tb}|$

t-channel (tqb)

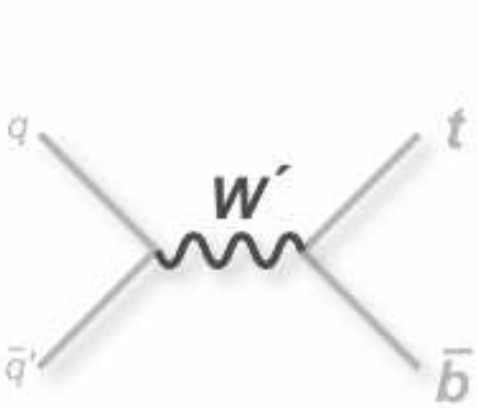


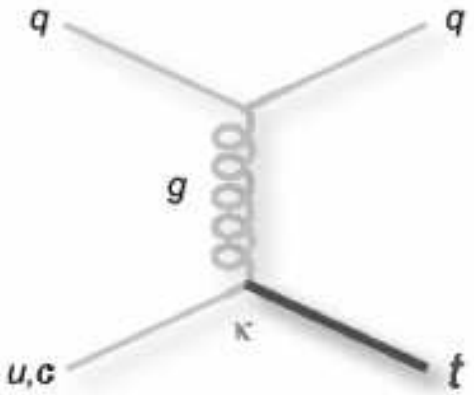
- $\sigma_{NLO} = 1.98 \pm 0.25$ pb (*)
- current limits (95% C.L.):
  Run II DØ: < 4.4 pb (370pb^{-1})
  Run II CDF: < 3.2 pb (700pb^{-1})

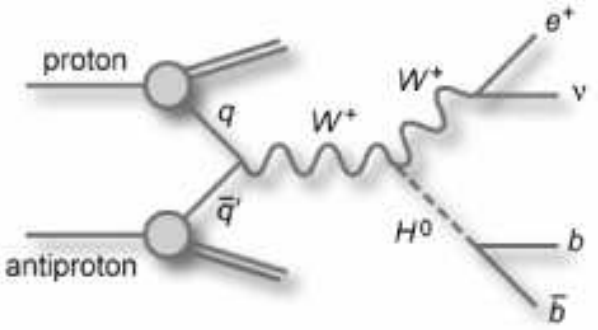
(*) Phys.Rev. D70 (2004) 114012

Motivation

- Directly measure $|V_{tb}|$ for the first time
- Cross section sensitivity to beyond the SM processes
- Source of polarized top quarks. Spin correlations measurable in decay products.
- Important background to Higgs search
- Test of techniques to extract a small signal out of a large background







Decision Tree Application to Single Top

DT Choices

- 1/3 of MC for training
- Adaboost $\beta = 0.2$
- Boosting cycles $= 20$
- Signal leaf if purity > 0.5
- Minimum leaf size $= 100$ events
- Same total weight to signal and background to start
- Goodness of split - Gini factor

Analysis Strategy

- Train 36 separate trees: $(s,t,s+t)$ x (e,μ) x (2,3,4 jets) x (1,2 tags)
- For each signal train against the sum of backgrounds

The 49 input variables

Object Kinematics

- p_T(jet1)
- p_T(jet2)
- p_T(jet3)
- p_T(jet4)
- p_T(best1)
- p_T(notbest1)
- p_T(notbest2)
- p_T(tag1)
- p_T(untag1)
- p_T(untag2)

Angular Correlations

- ΔR(jet1,jet2)
- cos(best1,lepton)$_{\mathrm{besttop}}$
- cos(best1,notbest1)$_{\mathrm{besttop}}$
- cos(tag1,alljets)$_{\mathrm{alljets}}$
- cos(tag1,lepton)$_{\mathrm{btaggedtop}}$
- cos(jet1,alljets)$_{\mathrm{alljets}}$
- cos(jet1,lepton)$_{\mathrm{btaggedtop}}$
- cos(jet2,alljets)$_{\mathrm{alljets}}$
- cos(jet2,lepton)$_{\mathrm{btaggedtop}}$
- cos(lepton, Q(lepton)$\times z$)$_{\mathrm{besttop}}$
- cos(lepton,besttopframe)$_{\mathrm{besttopCMframe}}$
- cos(lepton,btaggedtopframe)$_{\mathrm{btaggedtopCMframe}}$
- cos(notbest,alljets)$_{\mathrm{alljets}}$
- cos(notbest,lepton)$_{\mathrm{besttop}}$
- cos(untag1,alljets)$_{\mathrm{alljets}}$
- cos(untag1,lepton)$_{\mathrm{btaggedtop}}$

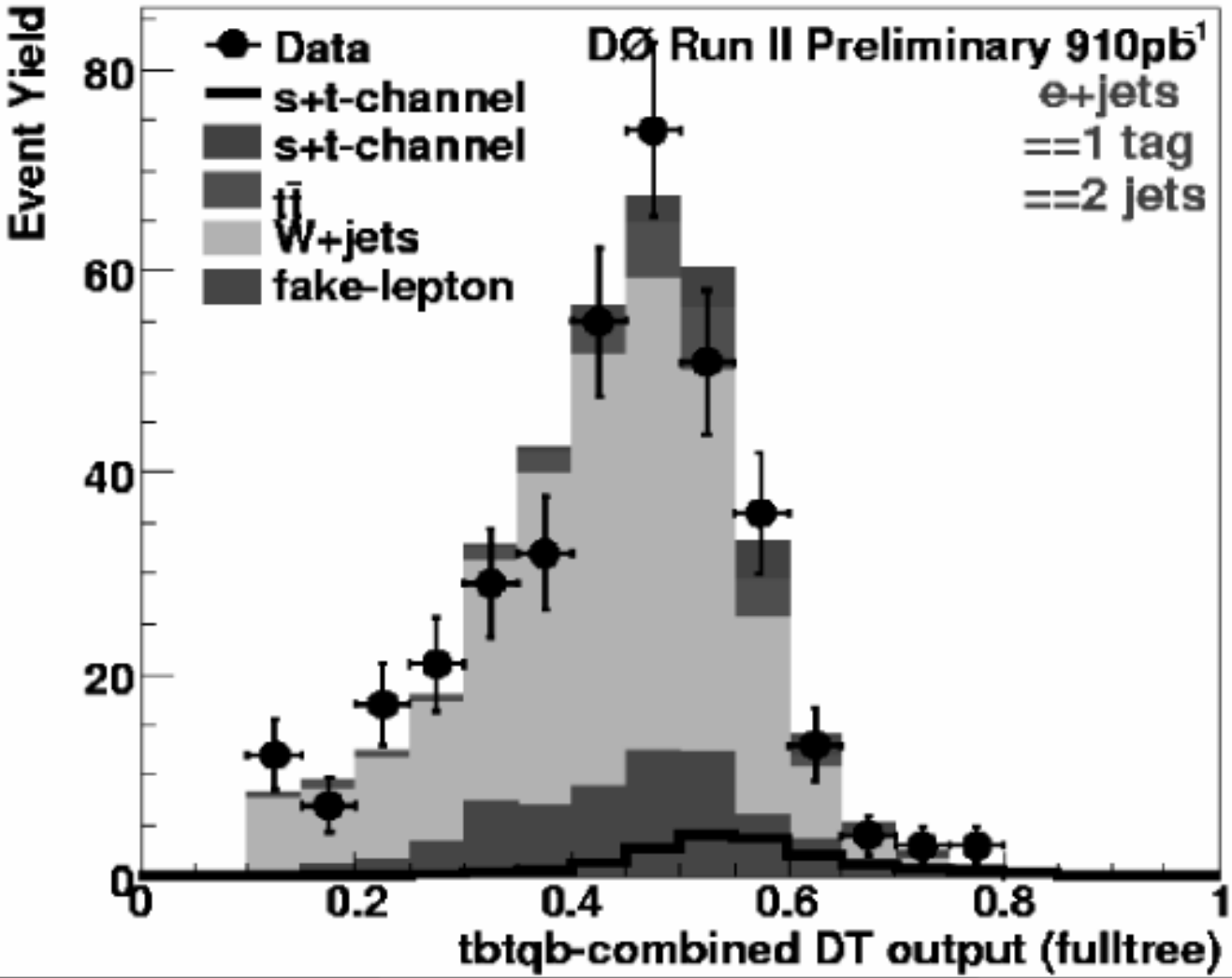
Event Kinematics

- Aplanarity(alljets, W)
- M(W,best1) ("best" top mass)
- M(W,tag1) ("b-tagged" top mass)
- H_T(alljets)
- H_T(alljets$-$best1)
- H_T(alljets$-$tag1)
- H_T(alljets, W)
- H_T(jet1,jet2)
- H_T(jet1,jet2, W)
- M(alljets)
- M(alljets$-$best1)
- M(alljets$-$tag1)
- M(jet1,jet2)
- M(jet1,jet2, W)
- M_T(jet1,jet2)
- M_T(W)
- Missing E_T
- p_T(alljets$-$best1)
- p_T(alljets$-$tag1)
- p_T(jet1,jet2)
- Q(lepton)$\times\eta$(untag1)
- $\sqrt{\hat{s}}$
- Sphericity(alljets, W)

- Adding variables does not degrade performance
- Tested shorter lists, lose some sensitivity
- Same list used for all channels

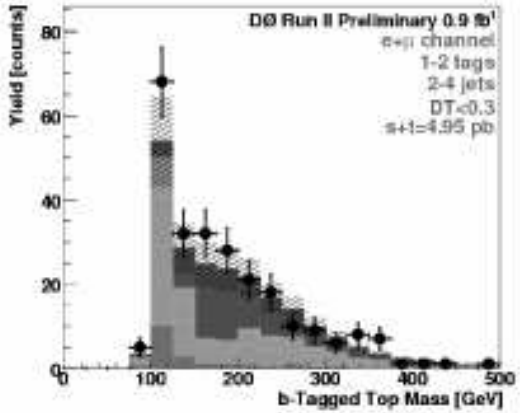
Decision Tree Output: 2 jet, 1 tag

Of course, we have 36 different Decision Trees, let's look at electron, 2 jet, 1 tag:

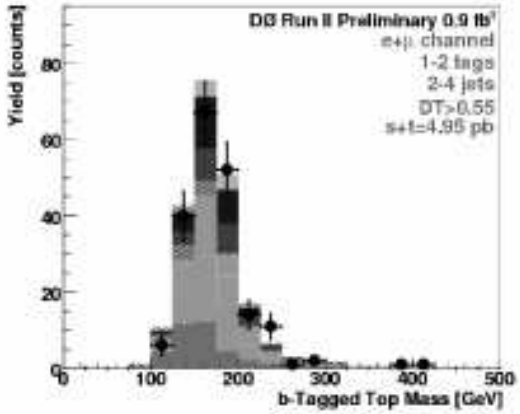


Invariant W – b mass: M(W, b)

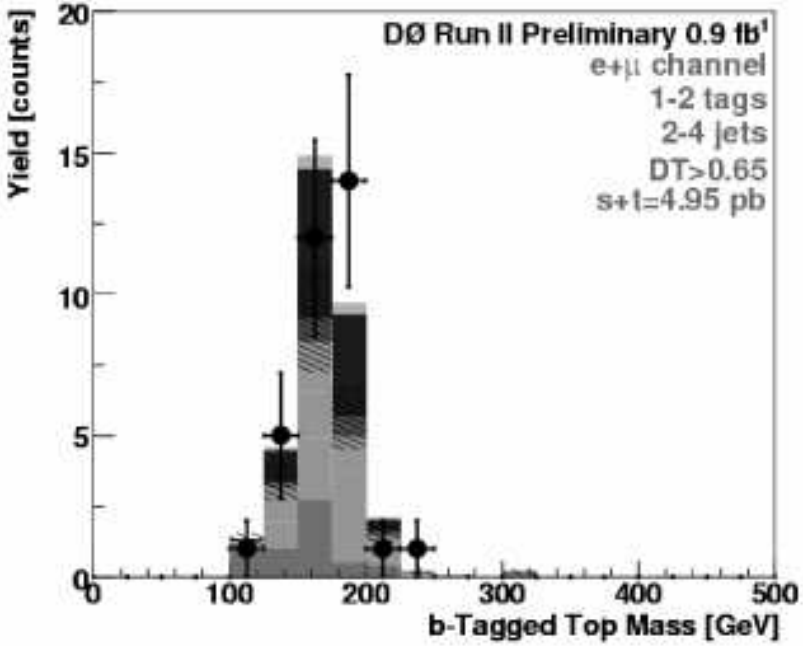
$DT < 0.3$



$DT > 0.55$



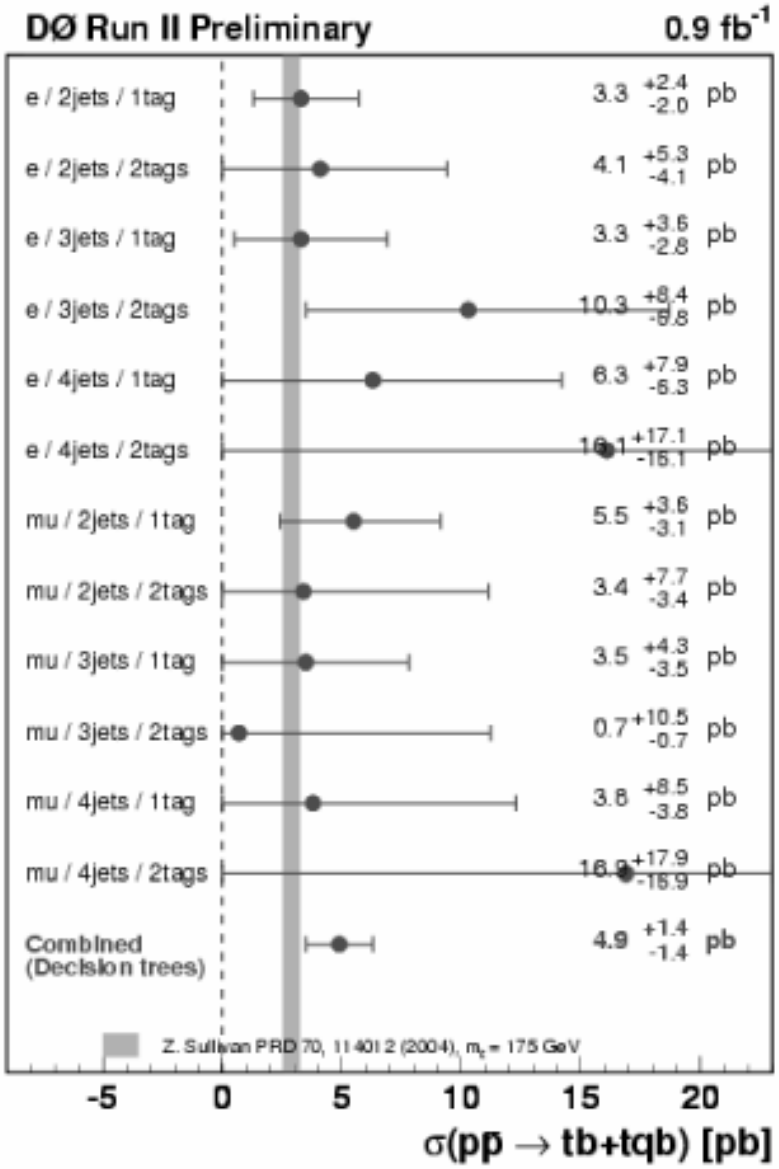
$DT > 0.65$



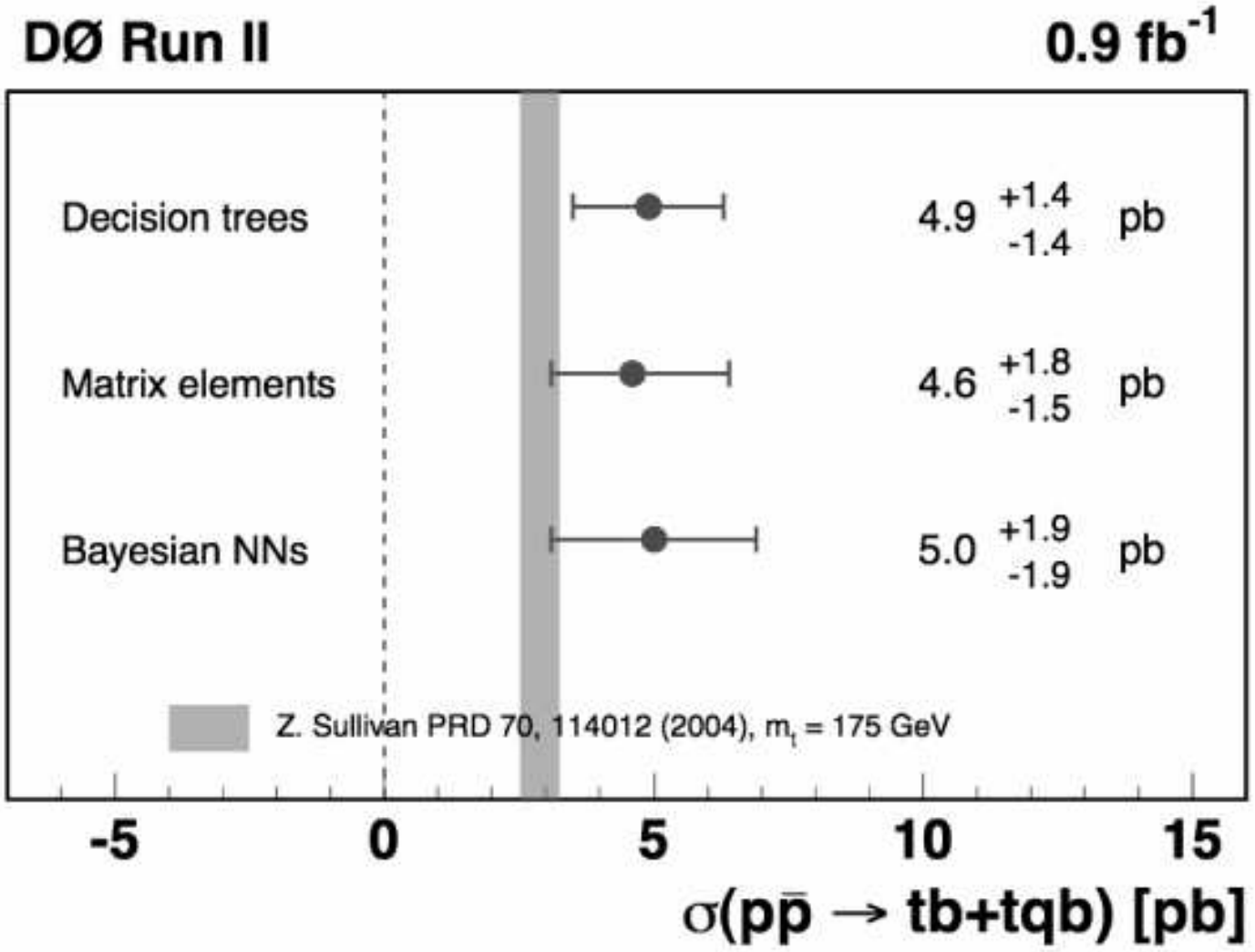
- Excess in high DT output region.

DT: Decision Tree output

DT Single Top Results



Single Top – All methods

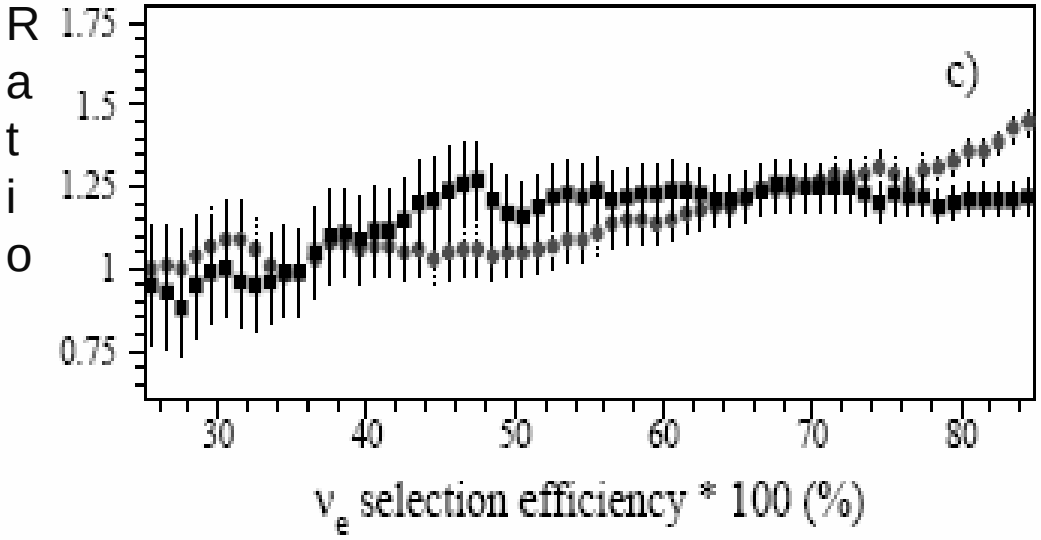


Conclusion

- **The Boosted Decision Tree is a modern (5-10 years old) concept in data analyses**
- **Very successfully established at MiniBooNE and at DØ**
- **Might become a standard method in future**

Boosting with 22 and with 52 variables

Vertical axis is the ratio of bkrd kept for 21(22) var./that kept for 52 var., both for boosting



- Red is if training sample is cocktail and black is if training sample is pi0
- Error bars are MC statistical errors only