Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Introduction

RIPPER

cost-sensitivity

Bagging

Application

Comparison

Conclusions and outlook

Application of the rule-growing algorithm RIPPER to particle physics analysis

Markward Britsch[1], Nikolai Gagunashvili[1,2], Michael Schmelling[1]

[1]Max-Planck-Institute for Nuclear Physics, [2]University of Akureyri, Iceland

2008-11-5, ACAT 2008, Erice

Outline

1. Introduction
2. RIPPER
3. Cost-sensitive classification
4. Bagging
5. Applying the MVA algorithm
6. Comparison to other methods
7. Conclusions and outlook

Outline

1. Introduction
2. RIPPER
3. Cost-sensitive classification
4. Bagging
5. Applying the MVA algorithm
6. Comparison to other methods
7. Conclusions and outlook

Multivariate analysis in HEP

- called data mining in computer science community

Multivariate analysis in HEP

- called data mining in computer science community
- classifier (neural network, decision tree . . .) learns on a training data set

Multivariate analysis in HEP

- called data mining in computer science community
- classifier (neural network, decision tree . . .) learns on a training data set
- classifier output: probability (*e.g.* for a candidate to be signal)

Multivariate analysis in HEP

- called data mining in computer science community
- classifier (neural network, decision tree . . .) learns on a training data set
- classifier output: probability (*e.g.* for a candidate to be signal)
- cut on probability

Multivariate analysis in HEP

- called data mining in computer science community
- classifier (neural network, decision tree . . .) learns on a training data set
- classifier output: probability (*e.g.* for a candidate to be signal)
- cut on probability
  - to change signal to background or significance

Multivariate analysis in HEP

- called data mining in computer science community
- classifier (neural network, decision tree . . .) learns on a training data set
- classifier output: probability (*e.g.* for a candidate to be signal)
- cut on probability
  - to change signal to background or significance
  - to account for larger abundance (in real data) of BG

Data mining: imbalanced data sets

- *i.e.* in HEP often imbalanced problems
  e.g. much more background than signal events

Data mining: imbalanced data sets

- *i.e.* in HEP often imbalanced problems
  e.g. much more background than signal events
- from data mining, possible solution:
  - appropriate classifier
  - Cost-sensitive approach
  - sampling based approach
  - bagging

Data mining: imbalanced data sets

- *i.e.* in HEP often imbalanced problems
  e.g. much more background than signal events
- from data mining, possible solution:
  - appropriate classifier
  - Cost-sensitive approach
  - sampling based approach
  - bagging
- some *partly* equivalent to cut on probability

Data mining: imbalanced data sets

- *i.e.* in HEP often imbalanced problems
  e.g. much more background than signal events
- from data mining, possible solution:
  - appropriate classifier
  - Cost-sensitive approach
  - sampling based approach
  - bagging
- some *partly* equivalent to cut on probability
- $\rightarrow$ investigate the difference between cut on probability and data mining solutions above

Data mining: imbalanced data sets

- *i.e.* in HEP often imbalanced problems
  e.g. much more background than signal events
- from data mining, possible solution:
  - appropriate classifier
  - Cost-sensitive approach
  - sampling based approach
  - bagging
- some *partly* equivalent to cut on probability
- $\rightarrow$ investigate the difference between cut on probability and data mining solutions above
- $\rightarrow$ which is better?

Outline

1 Introduction
2 RIPPER
3 Cost-sensitive classification
4 Bagging
5 Applying the MVA algorithm
6 Comparison to other methods
7 Conclusions and outlook

What are rule sets?

Technique for classifying events using a collection of "if…then…" rules. For example:

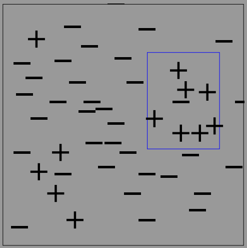
```
(IPpi >= 1.039316) and (DoCA <= 0.307358) and
(IP <= 0.270767) and (IPp >= 0.800645)
=> class=Lambda

(IPpi >= 0.637403) and (DoCA <= 0.159043) and
(IP <= 0.12081) and (ptpi >= 149.2332) and
(IP >= 0.003371) => class=Lambda

=> class=BG
```

What is RIPPER, why RIPPER??

- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets

What is RIPPER, why RIPPER??

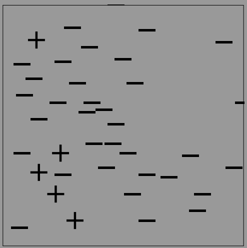
- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily



rule 1

What is RIPPER, why RIPPER??

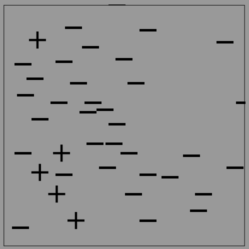
- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily



delete rule 1 instances

What is RIPPER, why RIPPER??

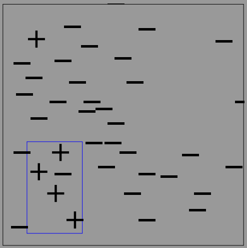
- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily
  3. prune rule



delete rule 1 instances

What is RIPPER, why RIPPER??

- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily
  3. prune rule
  4. go to 2), stopping criteria: description length, error rate



What is RIPPER, why RIPPER??

- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily
  3. prune rule
  4. go to 2), stopping criteria: description length, error rate
  5. optimization of rules

What is RIPPER, why RIPPER??

- direct rule based classifier (see Cohen (1995) [1])
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily
  3. prune rule
  4. go to 2), stopping criteria: description length, error rate
  5. optimization of rules

Advantages:

- rule set: relatively easy to interpret
- good for imbalanced problems

Outline

1 Introduction

2 RIPPER

3 Cost-sensitive classification

4 Bagging

5 Applying the MVA algorithm

6 Comparison to other methods

7 Conclusions and outlook

What is Cost-sensitive classification?

- assign a cost to wrongly (or correctly) classified instances ("events", "candidates")

What is Cost-sensitive classification?

- assign a cost to wrongly (or correctly) classified instances ("events", "candidates")
- $\rightarrow$ cost matrix, *e.g.*:

	predicted BG	predicted signal
true BG	0	100
true signal	1	0

What is Cost-sensitive classification?

- assign a cost to wrongly (or correctly) classified instances ("events", "candidates")
- $\rightarrow$ cost matrix, *e.g.*:

	predicted BG	predicted signal
true BG	0	100
true signal	1	0

- classification algorithm minimizes cost

What is Cost-sensitive classification?

- assign a cost to wrongly (or correctly) classified instances ("events", "candidates")
- $\rightarrow$ cost matrix, *e.g.*:

	predicted BG	predicted signal
true BG	0	100
true signal	1	0

- classification algorithm minimizes cost
- mainly two ways:
  - threshold adjusting
  - instance weighting

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG}, \mathrm{s})$
tr. signal	$C(\mathrm{s}, \mathrm{BG})$	0

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG}, \mathrm{s})$
tr. signal	$C(\mathrm{s}, \mathrm{BG})$	0

Minimize cost for a rule t, class $i = \mathrm{s}, \mathrm{BG}$:

$$C(i|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j,i).$$

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG}, \mathrm{s})$
tr. signal	$C(\mathrm{s}, \mathrm{BG})$	0

Minimize cost for a rule t, class $i = \mathrm{s}, \mathrm{BG}$:

$$C(i|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j,i).$$

t is assigned to the signal class if:

$$p(\mathrm{s}|t) \qquad > p(\mathrm{BG}|t)$$

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG},\mathrm{s})$
tr. signal	$C(\mathrm{s},\mathrm{BG})$	0

Minimize cost for a rule t, class $i = \mathrm{s},\mathrm{BG}$:

$$C(i|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j,i).$$

t is assigned to the signal class if:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG}, \mathrm{s})$
tr. signal	$C(\mathrm{s}, \mathrm{BG})$	0

Minimize cost for a rule t, class $i = \mathrm{s}, \mathrm{BG}$:

$$C(i|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j,i).$$

t is assigned to the signal class if:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

$$\Rightarrow p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > 1 - p(\mathrm{s}|t)C(\mathrm{BG},\mathrm{s})$$

$$\Rightarrow p(\mathrm{s}|t) > \frac{C(\mathrm{BG},\mathrm{s})}{C(\mathrm{BG},\mathrm{s}) + C(\mathrm{s},\mathrm{BG})}$$

Threshold adjusting

Let's start with a cost matrix as before:

	pred. BG	pred. signal
tr. BG	0	$C(\mathrm{BG}, \mathrm{s})$
tr. signal	$C(\mathrm{s}, \mathrm{BG})$	0

Minimize cost for a rule t, class $i = \mathrm{s}, \mathrm{BG}$:

$$C(i|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j, i).$$

t is assigned to the signal class if:

$$p(\mathrm{s}|t)C(\mathrm{s}, \mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG}, \mathrm{s})$$

$$\Rightarrow p(\mathrm{s}|t)C(\mathrm{s}, \mathrm{BG}) > 1 - p(\mathrm{s}|t)C(\mathrm{BG}, \mathrm{s})$$

$$\Rightarrow p(\mathrm{s}|t) > \frac{C(\mathrm{BG}, \mathrm{s})}{C(\mathrm{BG}, \mathrm{s}) + C(\mathrm{s}, \mathrm{BG})}$$

$\rightarrow$ This is equivalent to a cut on the probability!

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances
- mainly equivalent to applying a cost:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

$C(\mathrm{s},\mathrm{BG})$ ($C(\mathrm{BG},\mathrm{s})$) – replication factor of signal (BG)

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances
- mainly equivalent to applying a cost:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

$C(\mathrm{s},\mathrm{BG})$ ($C(\mathrm{BG},\mathrm{s})$) – replication factor of signal (BG)

- instance weighting: automated sampling/*weighting* of instances according to *cost*

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances
- mainly equivalent to applying a cost:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

$C(\mathrm{s},\mathrm{BG})$ ($C(\mathrm{BG},\mathrm{s})$) – replication factor of signal (BG)

- instance weighting: automated sampling/*weighting* of instances according to *cost*
- for some classifiers (*e.g.* neural networks) not better than threshold adjusting

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances
- mainly equivalent to applying a cost:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

$C(\mathrm{s},\mathrm{BG})$ ($C(\mathrm{BG},\mathrm{s})$) – replication factor of signal (BG)

- instance weighting: automated sampling/*weighting* of instances according to *cost*
- for some classifiers (*e.g.* neural networks) not better than threshold adjusting
- better than threshold adjusting for classifiers that change with the balance of training data

Sampling and instance weighting

- simplest forms:
  - undersampling by leaving out instances
  - oversampling by replicating instances
- mainly equivalent to applying a cost:

$$p(\mathrm{s}|t)C(\mathrm{s},\mathrm{BG}) > p(\mathrm{BG}|t)C(\mathrm{BG},\mathrm{s})$$

  $C(\mathrm{s},\mathrm{BG})$ ($C(\mathrm{BG},\mathrm{s})$) – replication factor of signal (BG)
- instance weighting: automated sampling/*weighting* of instances according to *cost*
- for some classifiers (*e.g.* neural networks) not better than threshold adjusting
- better than threshold adjusting for classifiers that change with the balance of training data
- *e.g.* decision trees, rules – typically using error rate

Outline

1 Introduction

2 RIPPER

3 Cost-sensitive classification

4 Bagging

5 Applying the MVA algorithm

6 Comparison to other methods

7 Conclusions and outlook

What is bagging, why bagging?

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Introduction

RIPPER

cost-sensitivity

Bagging

Application

Comparison

Conclusions and outlook

- draw *with replacement* at random *N* instances from your sample

orig. sample	1	2	3	4	5
1^{st} iteration	2	5	1	1	4
2^{nd} iteration	5	3	2	2	4

What is bagging, why bagging?

Introduction
RIPPER
cost-sensitivity
Bagging
Application
Comparison
Conclusions and outlook

- draw *with replacement* at random N instances from your sample
- do this r times

orig. sample	1	2	3	4	5
1^{st} iteration	2	5	1	1	4
2^{nd} iteration	5	3	2	2	4
⋮					
r^{th} iteration	1	1	5	1	4

What is bagging, why bagging?

- draw *with replacement* at random N instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these

What is bagging, why bagging?

- draw *with replacement* at random N instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities

What is bagging, why bagging?

- draw *with replacement* at random N instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities
- N is typically the number of instances in your sample

What is bagging, why bagging?

- draw *with replacement* at random N instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities
- N is typically the number of instances in your sample
- this works very well if your classifier is unstable, *i.e.* prone to change with noise (RIPPER, decision trees)

What is bagging, why bagging?

- draw *with replacement* at random N instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities
- N is typically the number of instances in your sample
- this works very well if your classifier is unstable, *i.e.* prone to change with noise (RIPPER, decision trees)
- reduces overfitting (for oversampling)

Outline

1 Introduction

2 RIPPER

3 Cost-sensitive classification

4 Bagging

5 Applying the MVA algorithm

6 Comparison to other methods

7 Conclusions and outlook

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Introduction

RIPPER

cost-sensitivity

Bagging

Application

Comparison

Conclusions and outlook

Decay and used data

- $\Lambda \rightarrow p^{+} + \pi^{-}$

Decay and used data

- $\Lambda \rightarrow p^{+} + \pi^{-}$
- LHCb Monte Carlo minimum bias

Decay and used data

- $\Lambda \to p^+ + \pi^-$
- LHCb Monte Carlo minimum bias
- candidates: pairs of differently charged *long* tracks

Decay and used data

- $\Lambda \rightarrow p^{+} + \pi^{-}$
- LHCb Monte Carlo minimum bias
- candidates: pairs of differently charged *long* tracks
- training set: 5×1000 Λ, 13000 BG

Decay and used data

- $\Lambda \to p^+ + \pi^-$
- LHCb Monte Carlo minimum bias
- candidates: pairs of differently charged *long* tracks
- training set: 5×1000 Λ, 13000 BG
- testing set: 1000 Λ, 180000 BG

Decay and used data

- $\Lambda \to p^{+} + \pi^{-}$
- LHCb Monte Carlo minimum bias
- candidates: pairs of differently charged *long* tracks
- training set: $5 \times 1000\ \Lambda$, 13000 BG
- testing set: $1000\ \Lambda$, 180000 BG
- use 10 geometric and kinematic variables

Application

- classification step using WEKA [2] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER

Application

- classification step using WEKA [2] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- make two classification steps:

Application

- classification step using WEKA [2] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- make two classification steps:
  1. preclassification using bagging (10 bags) (high cost for loosing $\Lambda \rightarrow$ keep almost all Λs, reduce BG)

	pr. BG	pr. Λ
tr. BG	0	1
tr. Λ	100	0

preselection cost matrix

Application

- classification step using WEKA [2] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- make two classification steps:
  1. preclassification using bagging (10 bags) (high cost for loosing Λ $\rightarrow$ keep almost all Λs, reduce BG)
  2. classify using bagging (25 bags) with high cost for wrongly accepted BG

	pr. BG	pr. Λ
tr. BG	0	1
tr. Λ	100	0

preselection cost matrix

	pr. BG	pr. Λ
tr. BG	0	x
tr. Λ	1	0

main cost matrix

Application

Introduction
RIPPER
cost-sensitivity
Bagging
Application
Comparison
Conclusions and outlook

- classification step using WEKA [2] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- make two classification steps:
  1. preclassification using bagging (10 bags) (high cost for loosing $\Lambda \rightarrow$ keep almost all Λs, reduce BG)
  2. classify using bagging (25 bags) with high cost for wrongly accepted BG
  3. to produce ROC curve: scan cost x

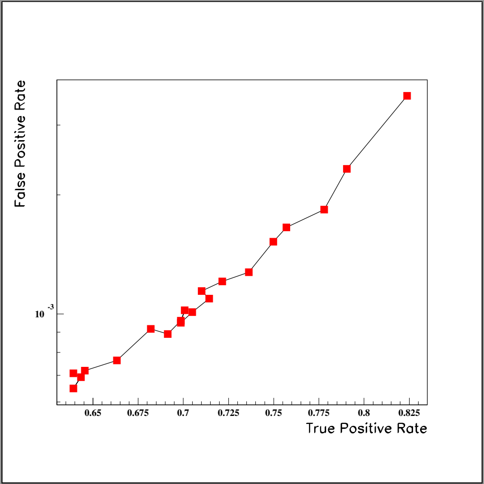
	pr. BG	pr. Λ
tr. BG	0	1
tr. Λ	100	0

preselection cost matrix

	pr. BG	pr. Λ
tr. BG	0	x
tr. Λ	1	0

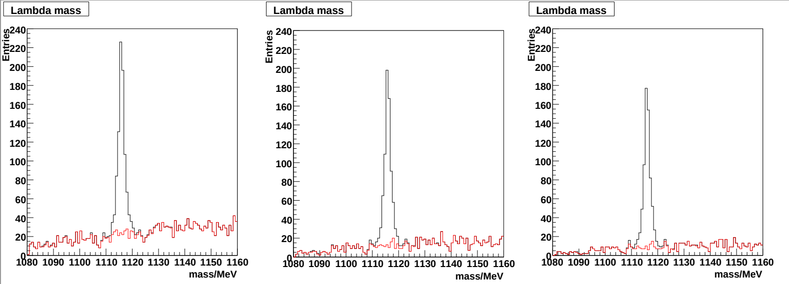
main cost matrix

ROC curve



Cost $x = 10, 20, \ldots, 200$

Mass plots



$x = 30$ $x = 100$ $x = 200$

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Introduction

RIPPER

cost-sensitivity

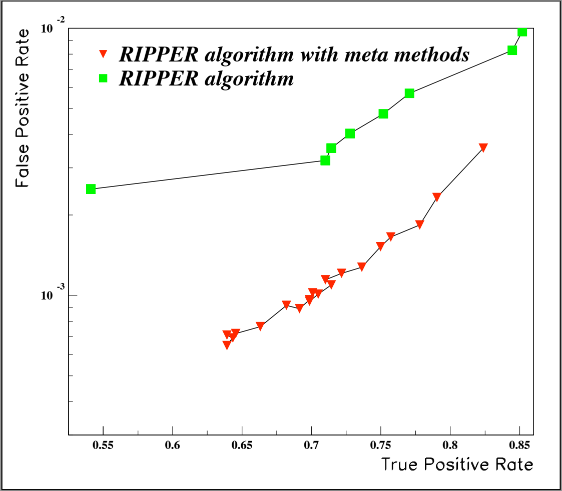
Bagging

Application

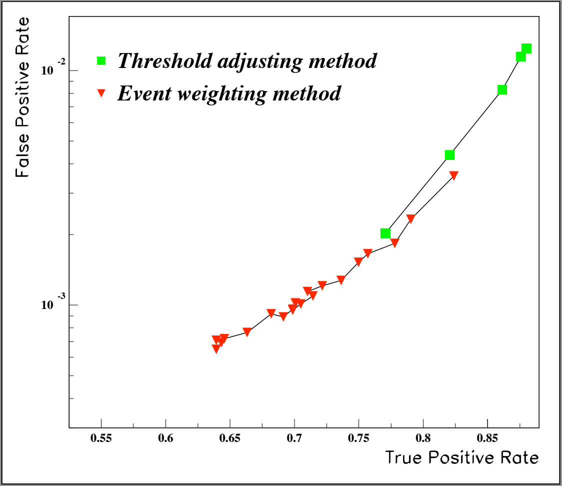
Comparison

Conclusions and outlook

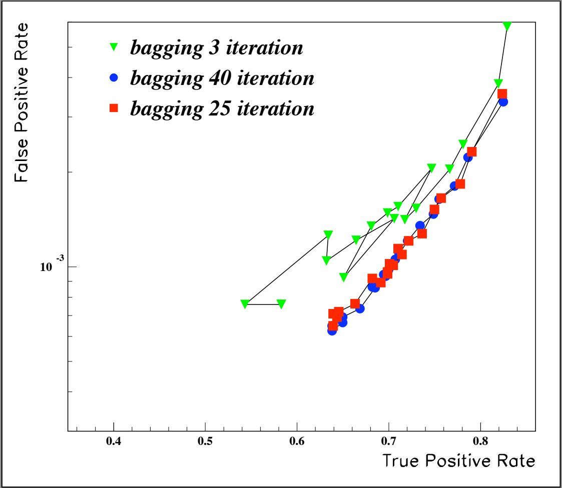
Without bagging and instance weighting



Threshold adjusting vs. instance weighting



Different bagging parameters



Outline

1 Introduction

2 RIPPER

3 Cost-sensitive classification

4 Bagging

5 Applying the MVA algorithm

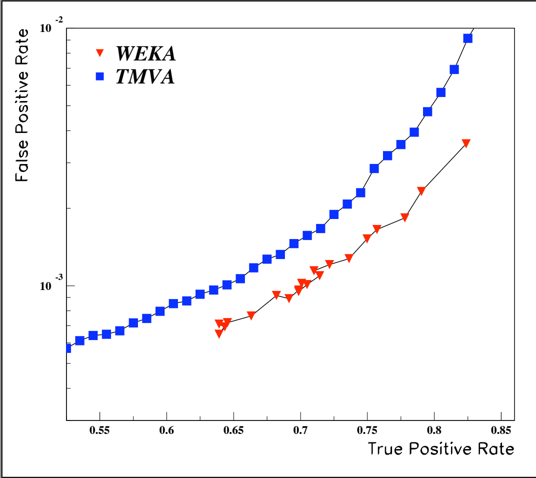
6 Comparison to other methods

7 Conclusions and outlook

Comparison with TMVA decision tree

TMVA [3] decision tree
(by Helge Voss):

- boosting
- pruning



false positive rate vs. true positive rate: TMVA tree, RIPPER

Choice of algorithm

Was RIPPER the right algorithm to choose?

Choice of algorithm

Was RIPPER the right algorithm to choose?
Compare with neural network (NN) and decision tree (DT); bagging and cost-sensitivity for *all* of the algorithms (no preclassification for *any*)

Choice of algorithm

Was RIPPER the right algorithm to choose?
Compare with neural network (NN) and decision tree (DT); bagging and cost-sensitivity for *all* of the algorithms (no preclassification for *any*)
neural network:

- multi layer perceptron
- 3 layers, 6 internal nodes
- binary output

Choice of algorithm

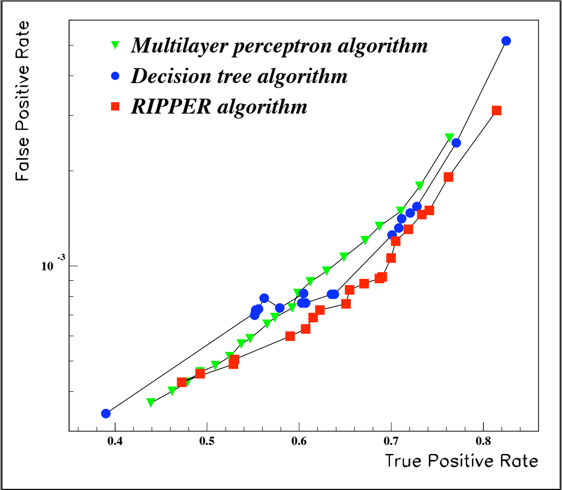
Was RIPPER the right algorithm to choose?
Compare with neural network (NN) and decision tree (DT); bagging and cost-sensitivity for *all* of the algorithms (no preclassification for *any*)
neural network:

- multi layer perceptron
- 3 layers, 6 internal nodes
- binary output

decision tree

- C 4.5
- includes pruning

Comp. with neural network and decision tree



false positive rate vs. true positive rate: NN, tree, RIPPER

Outline

1 Introduction

2 RIPPER

3 Cost-sensitive classification

4 Bagging

5 Applying the MVA algorithm

6 Comparison to other methods

7 Conclusions and outlook

Conclusions

- Cost-sensitive, sampling and cutting on probability are very similar
- instance weighting better for some classifiers
- bagging helps unstable classifiers, reduces overfitting

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Introduction
RIPPER
cost-sensitivity
Bagging
Application
Comparison
Conclusions and outlook

Conclusions

- Cost-sensitive, sampling and cutting on probability are very similar
- instance weighting better for some classifiers
- bagging helps unstable classifiers, reduces overfitting

Analysis w/ RIPPER, bagging, instance weighting:

- RIPPER fast and efficient to use
- bagging and instance weighting very important
- better than TMVA decision tree
- RIPPER better than NN or DT

Outlook

- use it for other analyzes (e.g. D^0)
- implement instance weighting in TMVA (and RIPPER?)
- dependence on MC errors
- increase weight on background in a more efficient way

Outline

8 Bibliography

9 Backup slides

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Bibliography

Backup slides

- W. W. Cohen, Fast Effective Rule Induction, In Machine Learning: Proc. of the 12th International Conference on Machine Learning, Lake Tahoe, California: Morgan Kaufman, 1995
- http://www.cs.waikato.ac.nz/ml/weka/
- TMVA Toolkit for multivariate data analysis with root, A. Höker et al., arXiv physiscs/0703039, http://tmva.sf.net.

Outline

8 Bibliography

9 Backup slides

Set of variables

- *DoCA* - distance of closest approach
- *FL* - signed flight-length
- $c \cdot t$ - flight-length in Λ frame
- *IPp* - IP proton
- *IPpi* - IP pion
- $v_2 = \frac{IPpi^2 + IPp^2}{IP^2}$
- *ptp* - pt proton
- *ptpi* - pt pion
- $\tan\vartheta = \frac{pt}{pz}$
- $\cos\xi$, ξ - angle between impact vectors

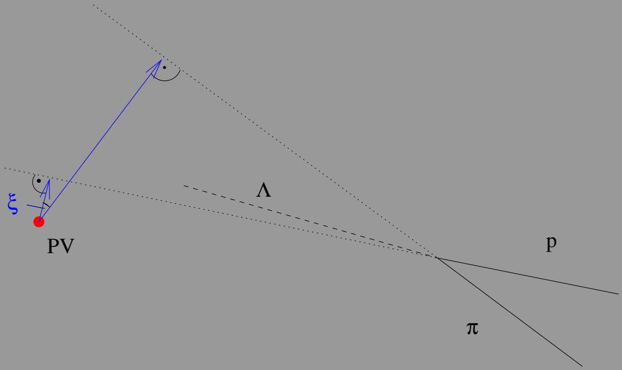
A new variable: ξ

Analysis with RIPPER

Britsch, Gagunashvili, Schmelling

Bibliography

Backup slides



Imbalanced data sets in HEP and data mining

- typical data mining application
  - credit card fraud, AIDS test
  - rare class is "fraud transaction" or "AIDS infection"
  - $\rightarrow$ high cost not detecting *rare* instance
- HEP mostly (particle selection)
  - high background
  - high cost for (*non-rare*) background classified as signal
- which translates to:
  - data mining: imbalance has to be balanced
  - HEP particle selection: imbalance has to be *enhanced*