Classifying extremely imbalanced data sets

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

[1]Max-Planck-Institut für Kernphysik, [2]University of Akureyri, Iceland

2010-2-23, ACAT 2010, Jaipur

Outline

Outline

Methods for imbalance

- in HEP often imbalanced problems
  e.g. much more background than signal events
- we have a method tested on Λ selection (background to signal ratio < 100)
- here try it on a D^0-selection w/o usage of particle identification (background to signal ratio ~ 3000)
- it turns out that this extreme imbalance needs special care
- I will briefly recap our basic methods in the following

Our MVA-method

- using RIPPER classifier, rule based

```
(v1 >= 1.039316) and (v2 <= 0.307358)
and (v3 <= 0.270767) and (v4 >= 0.800645)
=> class=Lambda
(v1 >= 0.637403) and (v2 <= 0.159043)
and (v3 <= 0.12081) and (v5 >= 149.2332)
and (v3 >= 0.003371)
=> class=Lambda
=> class=BG
```

Our MVA-method

- using RIPPER classifier, rule based
- introduce cost to change outcome (instead of cutting on a discriminant)

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

Our MVA-method

- using RIPPER classifier, rule based
- introduce cost to change outcome (instead of cutting on a discriminant)
- the cost is introduced by weights in training $\rightarrow$ new classifier model for each cost

Our MVA-method

- using RIPPER classifier, rule based
- introduce cost to change outcome
  (instead of cutting on a discriminant)
- the cost is introduced by weights in training
  $\rightarrow$ new classifier model for each cost
- use bagging to stabilize algorithm:
  like boosting, but without weights

orig. sample	1	2	3	4	5
1$^{\text{st}}$ iteration	2	5	1	1	4
2$^{\text{nd}}$ iteration	5	3	2	2	4
$\vdots$					
r$^{\text{th}}$ iteration	1	1	5	1	4

Our MVA-method

- using RIPPER classifier, rule based
- introduce cost to change outcome (instead of cutting on a discriminant)
- the cost is introduced by weights in training $\rightarrow$ new classifier model for each cost
- use bagging to stabilize algorithm: like boosting, but without weights
- make one or two preselections for large training sets to prevent memory overflow and to save time

Application of MVA method

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

[1] http://www.cs.waikato.ac.nz/ml/weka/

Application of MVA method

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

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Application of MVA method

- classification step using WEKA[1] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- for preselection: extra classification step:
  1. preclassification incl. bagging – high cost for loosing D^0 $\rightarrow$ keep almost all D^0s, reduce background (BG)

	pr. BG	pr. D^0
tr. BG	0	1
tr. D^0	200	0

preselection cost matrix

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Application of MVA method

- classification step using WEKA[1] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- for preselection: extra classification step:
  1. preclassification incl. bagging – high cost for loosing D^0 $\rightarrow$ keep almost all D^0s, reduce background (BG)
  2. classify including bagging with high cost for wrongly accepted BG

	pr. BG	pr. D^0
tr. BG	0	1
tr. D^0	200	0

preselection cost matrix

	pr. BG	pr. D^0
tr. BG	0	x
tr. D^0	1	0

main cost matrix

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

Application of MVA method

- classification step using WEKA[1] package:
  1. bagging
  2. set cost (instance weighting)
  3. apply RIPPER
- for preselection: extra classification step:
  1. preclassification incl. bagging – high cost for loosing D^0 $\rightarrow$ keep almost all D^0s, reduce background (BG)
  2. classify including bagging with high cost for wrongly accepted BG
  3. to produce ROC curve: scan cost x (one classifier model per point in ROC curve)

	pr. BG	pr. D^0
tr. BG	0	1
tr. D^0	200	0

preselection cost matrix

	pr. BG	pr. D^0
tr. BG	0	x
tr. D^0	1	0

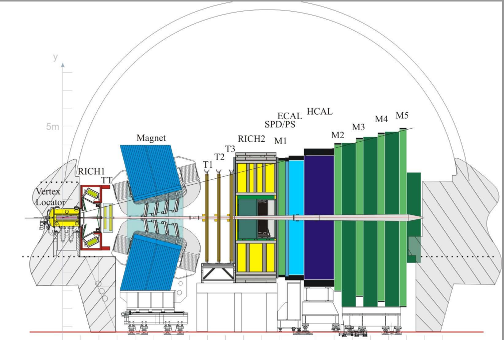
main cost matrix

[1] http://www.cs.waikato.ac.nz/ml/weka/

Outline

The LHCb experiment

- one of the four large experiments at *pp*-collider LHC
- made for precision measurements of CP violation & rare decays
- forward spectrometer
- Only tracking information used for these studies, no RICH



Decay and used data

- $D^0 \to \pi^+ + K^-$
- LHCb minimum bias Monte Carlo, $3.6 \cdot 10^7$ events from 2006, $\sqrt{s} = 14$ TeV
- candidates: pairs of differently charged tracks passing through full spectrometer
- distance of closest approach < 10 mm
- use 14 geometric and kinematic variables

Extremely imbalanced data sets

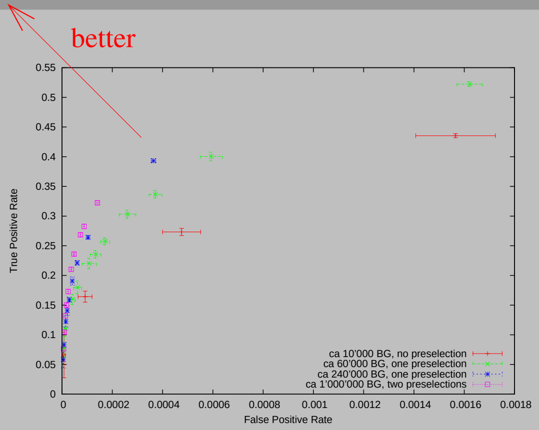
Britsch, Gagunashvili, Schmelling

Decay and used data

- $D^0 \to \pi^+ + K^-$
- LHCb minimum bias Monte Carlo, $3.6 \cdot 10^7$ events from 2006, $\sqrt{s} = 14$ TeV
- candidates: pairs of differently charged tracks passing through full spectrometer
- distance of closest approach < 10 mm
- use 14 geometric and kinematic variables
- training data sets: same number of signal increasing number of background

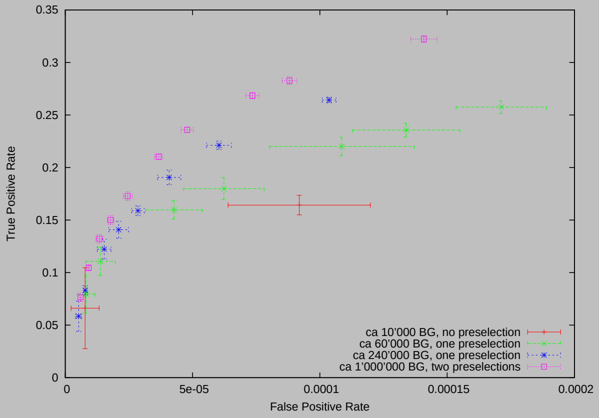
data set	# BG	# sig.	# presel.
test	$6.5 \cdot 10^6$	1827	–
training small	ca 10'000	1851	0
training mid	ca 60'000	1851	1
training larger	ca 240'000	1851	1
training largest	ca 1'000'000	1851	2

ROC curve, different # BG in training

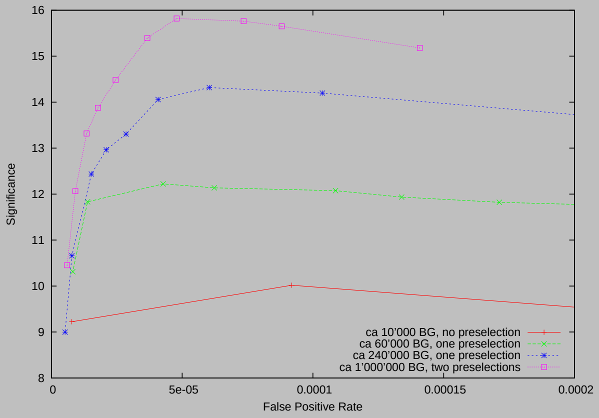


ROC curve: true positive rate (TPR = signal efficiency) versus false positive rate (FPR = background efficiency)

ROC curve (zoom), different # BG in training

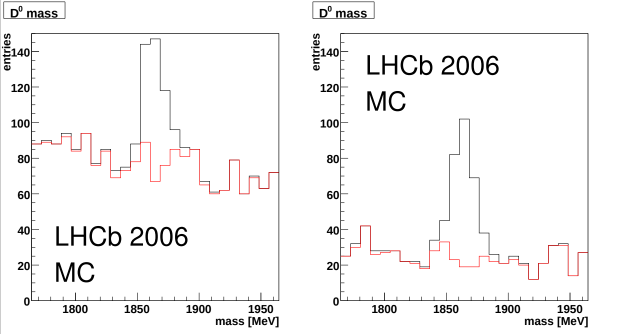


Significance – FPR, different # BG in training



$$\text{significance} = \frac{\#\text{signal}}{\sqrt{\#\text{signal} + \#\text{BG}}} \text{ after selection}$$

Mass plot comparison to cuts based analysis



cuts based,
same variables

multivariate analysis
(for same signal yield)

No RICH PID information used

Britsch, XVII International Workshop on Deep-Inelastic Scattering and Related Subjects, 2009, Madrid

Outline

The forest cover type data set

- we want to see if this behavior is special to our data set
- use some known data mining data set repository: `http://archive.ics.uci.edu/ml/`
- we choose the one called forest cover type: predicting forest cover type from cartographic variables
- observation (30 $\times$ 30 meter cell) determined from US Forest Service (USFS) in the Roosevelt National Forest of northern Colorado
- use the 10 integer variables (leaving out 44 binary ones)
- use class 4 (of 0 to 7) as “signal”, rest “background” to get unbalanced data set

Cover type training samples

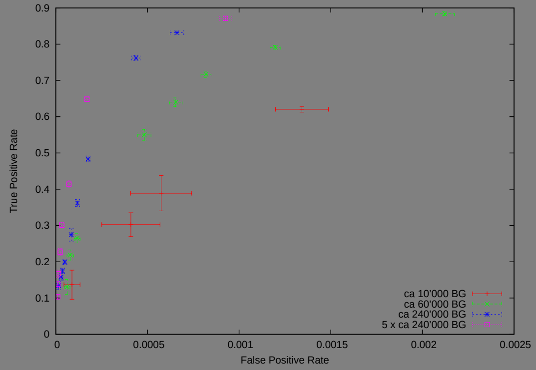
Again: use different training sets with same number of signal but increasing number of background:

data set	# BG	# sig.	# presel.
test	ca 290'000	1365	–
training small	ca 10'000	1382	0
training mid	ca 60'000	1382	1
training large	ca 240'000	1382	1
training artificial	5 $\times$ ca 240'000	1382	2

additional artificial BG data by 4 $\times$ randomization of existing BG instances using SMOTE algorithm[1]

[1] Chawla, Bowyer, Hall, Kegelmeyer, Journal of Artificial Intelligence Research 16 (2002) 341

Cover type ROC curves



We see the same effect here as in the D^0 data.
And the artificial data improves the result!

Outline

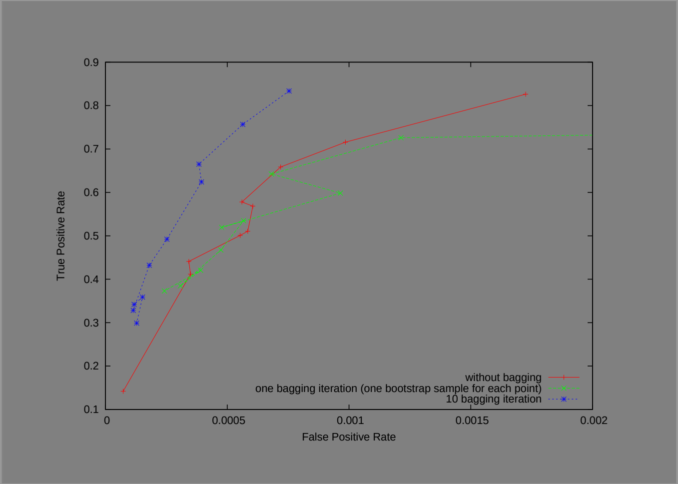
How do we make the error bars?

- we have a different classifier model for each point in ROC space
- these classifier models depend on
  1. random choices in bagging and RIPPER
  2. training sample choice
- (1) $\Rightarrow$ pure ROC curves look noisy

So we need:

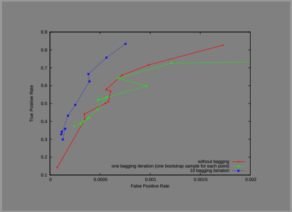
1. a way to smooth the curve (average many)
2. a measure for the scatter (error bars)

ROC curves w/ and w/o bootstrapping



red curve uses the **same** sample for training for all points, for the green training set re-sampled for each point.

What does that mean?



- the less noisy curve (red) hides its scatter (*i.e.*, its dependence on the training set)
- **the same is true for ordinary ROC curves** (cutting on a discriminant)
- the more noisy curve (green) tells us something about this scatter
- similar to using different (cross-validation) samples
- bagging reduces this scatter by using many bagging iterations (blue)

The way we do the errors

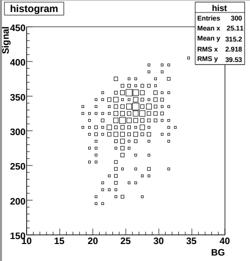
There are different methods discussed in literature, but **none** (that we could find) takes the scatter due to the training set into account.

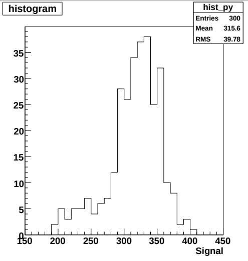
This is our (ad hoc) method:

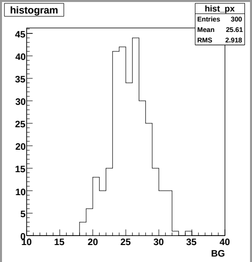
- do each main selection 10 times with different random seeds
- take the mean FPR and TPF as the point in ROC space
- similar to using 10 cross-validation samples
- take the standard deviations (SD) as errors in x and y
- the result is what you have seen in the plots

What is the distribution like? $\rightarrow$ next slide for 300 samples for one cost

Distributions for 300 samples, one cost







- using 300 samples, one cost
- different random seeds, no averaging
- distributions are asymmetric and have tails
- $\rightarrow$ SD has no interpretation as confidence level

Outline

Conclusions

For extremely imbalanced data sets:

- more BG in training is better for the LHCb-D^0 as well as the cover type data set – in an important region of FPR
- one or two preselections w/ less BG helps reducing data to handle large training sets
- even using extra artificial BG instances helps

For ROC curve errors:

- smooth ROC curves by doing 10 points w/ different random seed per point in ROC space
- get mean and standard deviation as position and error
- this seem reasonable and practical
- but it can not be interpreted as a confidence level

Additional, ongoing and future work

- more sophisticated ways to reduce data size w/o loosing classification quality
- better ways to average ROC curves and to produce error bars
- try different classifiers (e.g., decision trees) to see that behavior is general
- trying these methods on rare decays

Back up slides

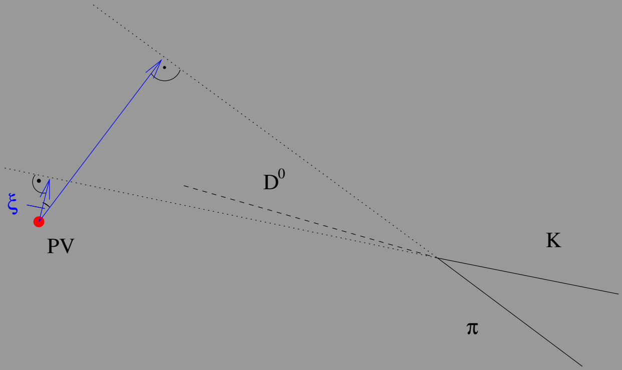
Outline

$D^0 \to K^-\pi^+$-Cuts

- long tracks only
- pion/kaon track #LHCbIDs > 27
- $pt > 700$ MeV
- $pt_{\mathrm{daughters}} > 500$ MeV
- $\cos\xi < -0.7$
- $FL > 1.5$ mm
- $DoCA < 0.07$ mm
- $\log \frac{DoCA}{FL} < -4.0$
- $IP < 0.08$ mm
- $IP_{\mathrm{daughters}} > 0.05$ mm
- $\log\left(\frac{IP_K^2 + IP_\pi^2}{IP^2}\right) > 3.0$
- for MVA: $FL \cdot \frac{M}{p} \approx ct$

ξ: angle between impact vectors

A new variable: ξ



Outline

What are rule sets?

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

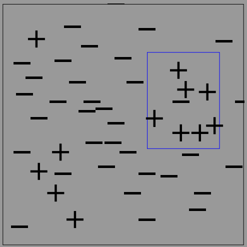
```
(IPpi >= 1.039316) and (DoCA <= 0.307358) and
(IP <= 0.270767) and (IPp >= 0.800645)
=> class=Lambda

(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 (Cohen 1995)
  1. divide training set into growing and pruning sets

What is RIPPER, why RIPPER?

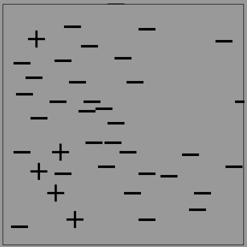
- direct rule based classifier (Cohen 1995)
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily



rule 1

What is RIPPER, why RIPPER?

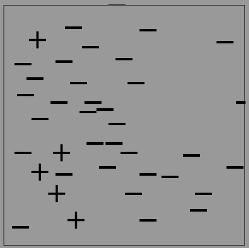
- direct rule based classifier (Cohen 1995)
  1. divide training set into growing and pruning sets
  2. grow a rule adding conditions greedily



delete rule 1 instances

What is RIPPER, why RIPPER?

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



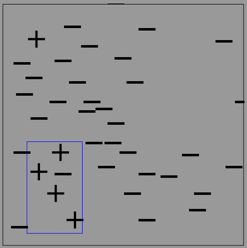
delete rule 1 instances

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

What is RIPPER, why RIPPER?

- direct rule based classifier (Cohen 1995)
  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



rule 2

What is RIPPER, why RIPPER?

- direct rule based classifier (Cohen 1995)
  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 (Cohen 1995)
  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

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

Outline

What is Cost-sensitive classification?

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

- 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

Compare costs for a rule t, class s, BG:

$$C(\mathrm{BG}|t) \quad >^{?} \quad C(\mathrm{s}|t)$$

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

Compare costs for a rule t, class s, BG:

$$C(\mathrm{BG}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j, \mathrm{BG}) >^{?} C(\mathrm{s}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j, \mathrm{s})$$

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

Compare costs for a rule t, class s, BG:

$$C(\mathrm{BG}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j, \mathrm{BG}) >^{?} C(\mathrm{s}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t)C(j, \mathrm{s})$$

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

Compare costs for a rule t, class s, BG:

$$C(\mathrm{BG}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j, \mathrm{BG}) >^{?} C(\mathrm{s}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j, \mathrm{s})$$

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

Compare costs for a rule t, class s, BG:

$$C(\mathrm{BG}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j, \mathrm{BG}) >^? C(\mathrm{s}|t) = \sum_{j=\mathrm{s},\mathrm{BG}} p(j|t) C(j, \mathrm{s})$$

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

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

- Variables
- RIPPER
- cost-sensitivity
- Bagging
- Cover type confidence
- Cover type – the data
- SMOTE

- 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

What is bagging, why bagging?

- similar to boosting, but no weights

What is bagging, why bagging?

- similar to boosting, but no weights
- draw *with replacement* at random 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?

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

- similar to boosting, but no weights
- draw *with replacement* at random 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?

- similar to boosting, but no weights
- draw *with replacement* at random instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these

What is bagging, why bagging?

- similar to boosting, but no weights
- draw *with replacement* at random instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

What is bagging, why bagging?

- similar to boosting, but no weights
- draw *with replacement* at random instances from your sample
- do this *r* times
- learn *r* classifiers (here *r* rule sets) on these
- let them vote or average their probabilities
- this works very well if your classifier is unstable, *i.e.* prone to change with noise (RIPPER, decision trees)

What is bagging, why bagging?

- similar to boosting, but no weights
- draw *with replacement* at random instances from your sample
- do this r times
- learn r classifiers (here r rule sets) on these
- let them vote or average their probabilities
- this works very well if your classifier is unstable, *i.e.* prone to change with noise (RIPPER, decision trees)
- reduces overfitting

Outline

Confidence intervals

From the distributions we can compute confidence intervals:

CL	kind	interval	interval center
90 %	signal	[229, 368]	299
90 %	BG	[20, 30]	25
68 %	signal	[282, 351]	317
68 %	BG	[23, 28]	26
SD	signal	[276, 354]	315
SD	BG	[22.2, 28.0]	25.1

Agreement between 68 % CL and SD, 90 % interval asymmetric for the signal.

Time limitations → not practical to produce 300 classifiers (× number of bagging iterations) per point in ROC space. So we have to live with the standard deviations as errors.

Outline

The data set – variables

- 54 variables, of which 44 are binary, the rest integer
- integer variables, *e.g.*,
  - Elevation: Elevation in meters
  - Slope: Slope in degrees
  - Vertical_Distance_To_Hydrology: vert dist to nearest surface water features in meters
- binary variables are: wilderness types and soil types
- classes 1-7 (# instances):
  1. Spruce/Fir (211840)
  2. Lodgepole Pine (283301)
  3. Ponderosa Pine (35754)
  4. Cottonwood/Willow (2747)
  5. Aspen (9493)
  6. Douglas-fir (17367)
  7. Krummholz (20510)
- total # instances: 581012

Data preparation

- first lesson: draw training & test sample *randomly*
- ignore the 40 soil type binary variables
- use class 4 (Cottonwood/Willow) as “signal”
- use all other classes as “background”
- $\Rightarrow$ 2747 signal and 578265 BG
- use half as test sample

Outline

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*
- SMOTE:
  - find *n* nearest neighbors (NN) for each instance (candidate)

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*
- SMOTE:
  - find *n* nearest neighbors (NN) for each instance (candidate)
  - do *k* loops

Randomization

Extremely imbalanced data sets

Britsch, Gagunashvili, Schmelling

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*
- SMOTE:
  - find *n* nearest neighbors (NN) for each instance (candidate)
  - do *k* loops
    - choose one of the NN randomly for each instance

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*
- SMOTE:
  - find n nearest neighbors (NN) for each instance (candidate)
  - do k loops
    - choose one of the NN randomly for each instance
    - choose all variables randomly in between the value of this variable of the instance and that of its neighbor

Randomization

The SMOTE algorithm

- multiply # of instances in a cunning way (instead of just replication)
- usually you want to balance the signal to BG ratio in the training set, *but not here!*
- SMOTE:
  - find *n* nearest neighbors (NN) for each instance (candidate)
  - do *k* loops
    - choose one of the NN randomly for each instance
    - choose all variables randomly in between the value of this variable of the instance and that of its neighbor
    - these variable choices make up a new instance