Hashing Classification for charged particle tracking

Luiza Adelina Ciucu (ATLAS)

29 May 2020

UNIVERSITÉ
DE GENÈVE

FACULTY OF SCIENCE
Physics Section

Introduction

- Compare two methods to balance the number of positive and negative hits : unbalanced (A) and balanced (B).

- Used 30 events. Each group of 10: 7 in train, 3 in test.

- Unbalanced (A): keep all buckets (with default weight of 1.0).

- Balanced (B): remove buckets so that bucket distribution is symmetric around 10 in NbPositiveHit per bucket (with default weight of 1.0).

- Used min 0, 4, 7 and 10 positive hits in the bucket; if less, consider all hits in the bucket to be negative, but use same reweighting as above.

- How a bucket is created:
  - Loop over events, and for each event build annoy index and then:
  - Loop over hits and for each hit bui2ld a bucket using annoy and 20 nearest neighbours by direction
  - Loop over hits in the bucket, find their particleID; find particleID with most hits in the bucket; denote it majority particle.
  - Loop over hits in the bucket again, if belongs to the majority particle assign output $+1$, else -1.

Bucket balancing procedure 1/4

o Remove buckets so that bucket distribution is symmetric around 10 in NbPositiveHit per bucket (with default weight of 1.0): Train and Test.

o Input: `nparray_output` (Nx20)

o Output: `nparray_outputBalanced` (Nx20)

o Step 1: count buckets for each category of nbPositiveHit
  - output: `dict_nbPositiveHit_counterBucket`
  - process: loop over buckets, for each bucket count positive hits into nbPositiveHit, increase counter in dicti at that key of nbPositiveHit.

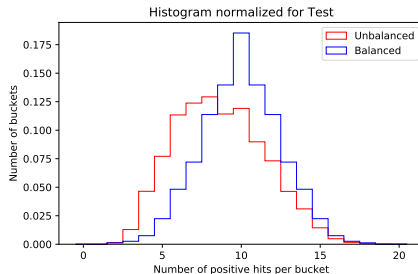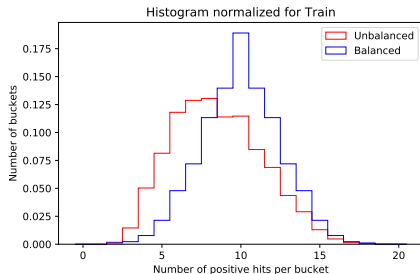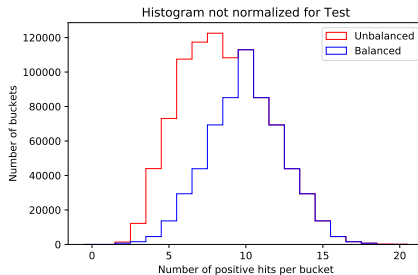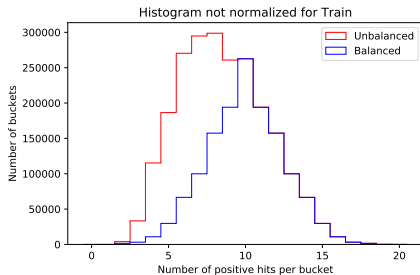o The result is shown below for Train. Note not balanced around 10.

```
Original unbalanced. Train:
nbPositiveHit=0 counterBucket=0 percentBucket=0.0
nbPositiveHit=1 counterBucket=0 percentBucket=0.0
nbPositiveHit=2 counterBucket=3973 percentBucket=0.2
nbPositiveHit=3 counterBucket=33349 percentBucket=1.5
nbPositiveHit=4 counterBucket=115266 percentBucket=5.0
nbPositiveHit=5 counterBucket=186516 percentBucket=8.1
nbPositiveHit=6 counterBucket=270380 percentBucket=11.8
nbPositiveHit=7 counterBucket=294847 percentBucket=12.9
nbPositiveHit=8 counterBucket=298765 percentBucket=13.0
nbPositiveHit=9 counterBucket=260813 percentBucket=11.4
nbPositiveHit=10 counterBucket=262604 percentBucket=11.5
nbPositiveHit=11 counterBucket=194065 percentBucket=8.5
nbPositiveHit=12 counterBucket=157420 percentBucket=6.9
nbPositiveHit=13 counterBucket=99938 percentBucket=4.4
nbPositiveHit=14 counterBucket=66595 percentBucket=2.9
nbPositiveHit=15 counterBucket=29752 percentBucket=1.3
nbPositiveHit=16 counterBucket=10828 percentBucket=0.5
nbPositiveHit=17 counterBucket=3184 percentBucket=0.1
nbPositiveHit=18 counterBucket=1560 percentBucket=0.1
nbPositiveHit=19 counterBucket=379 percentBucket=0.0
nbPositiveHit=20 counterBucket=148 percentBucket=0.0
```

Bucket balancing procedure 2/4

o Step 2: from this (unbalanced) dict calculate desired balanced dict.
  - input: `dict_nbPositiveHit_counterBucket`
  - output: `dict_nbPositiveHit_counterBucket_Balanced`
  - process: loop over i from the first half of nbPositiveHit (from 0 to 10)
  - `nbLeft=dict_nbPositiveHit_counterBucket[i]`
  - `nbRight=dict_nbPositiveHit_counterBucket[20-i]`
  - Find nbMin from nbLeft and nbRight
  - Set in the new balanced dictionary both values ot the nbMin
  - `dict_nbPositiveHit_counterBucke_Balancedt[i]=nbMin`
  - `dict_nbPositiveHit_counterBucke_Balancedt[20-i]=nbMin`

o The result is shown below for Train. Note it is balanced around 10.

o Right usually smaller, so remains the same and remove from left.

o nbPositiveHit 0 and 1 have counts of 0, → set 19 and 20 to zero.

```
Original unbalanced. Train:
nbPositiveHit=0 counterBucket=0 percentBucket=0.0
nbPositiveHit=1 counterBucket=0 percentBucket=0.0
nbPositiveHit=2 counterBucket=3973 percentBucket=0.2
nbPositiveHit=3 counterBucket=33349 percentBucket=1.5
nbPositiveHit=4 counterBucket=115266 percentBucket=5.0
nbPositiveHit=5 counterBucket=186536 percentBucket=8.1
nbPositiveHit=6 counterBucket=270380 percentBucket=11.8
nbPositiveHit=7 counterBucket=294847 percentBucket=12.9
nbPositiveHit=8 counterBucket=298765 percentBucket=13.0
nbPositiveHit=9 counterBucket=260813 percentBucket=11.4
nbPositiveHit=10 counterBucket=262604 percentBucket=11.5
nbPositiveHit=11 counterBucket=194065 percentBucket=8.5
nbPositiveHit=12 counterBucket=157420 percentBucket=6.9
nbPositiveHit=13 counterBucket=99938 percentBucket=4.4
nbPositiveHit=14 counterBucket=66595 percentBucket=2.9
nbPositiveHit=15 counterBucket=29752 percentBucket=1.3
nbPositiveHit=16 counterBucket=10828 percentBucket=0.5
nbPositiveHit=17 counterBucket=3184 percentBucket=0.1
nbPositiveHit=18 counterBucket=1560 percentBucket=0.1
nbPositiveHit=19 counterBucket=379 percentBucket=0.0
nbPositiveHit=20 counterBucket=48 percentBucket=0.0
```
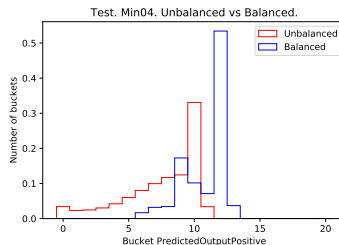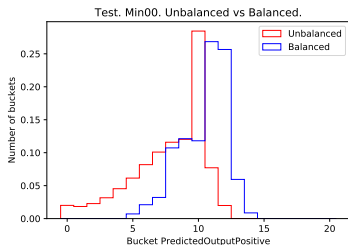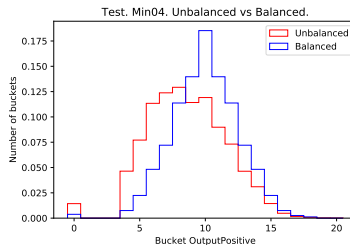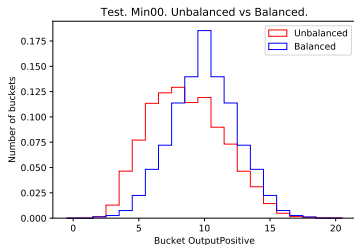
Bucket balancing procedure 3/4

o Step 3: use desired balanced dict to obtain balanced output.
- input: `dict_nbPositiveHit_counterBucket_Balanced`
- input: `nparray_output` (Nx20)
- output: `nparray_outputBalanced` (Nx20)
- process: loop over buckets from `nparray_output`:
- find nbPositiveHit for the bucket
- from nbPositiveHit find desired number of bucket in this category
- count the current number of buckets in this category
- if current counter $<$ *le* desired number, append to a list
- else (do nothing, so skip it)
- .
- after for loop convert list to `nparray_outputBalanced`
- as in step 1, calculate dictionary of counterBucket for each category
- by printing verify it is the same as the one desired (confirmed)
- next overlay histograms for nbPositiveHit in Unbalanced and Balanced
- as expected, now it is balanced, the right side is the same in both cases, and we removed buckets from the left
- But we set 19 and 20 to zero, to keep as 0 and 1. Though 0 will always have no buckets. But otherwise makes the symmetry harder.

Histogram NbBuckets vs NbPositiveHit in a bucket

o Unbalanced (Balanced) are not (are) symmetric around 10.

Count buckets in each NbPositiveHit category.

- Balanced around 10 NbPositiveHit via bucket removal.
- 0 and 1 no buckets → 19 and 20 are balanced to no buckets (ignored).
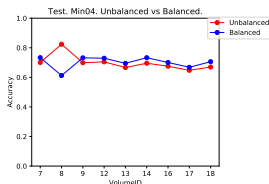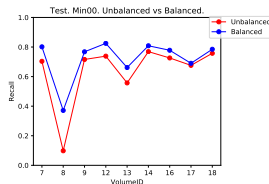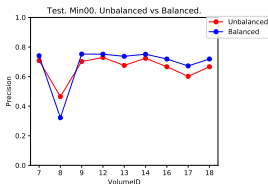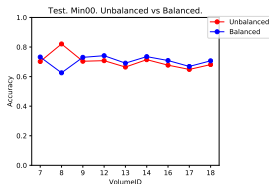
○ Min00 (left) and Min04 (right)

o Min07 (left) and Min10 (right).

o Min10 and method Unbalanced predicts all hits to be negative.
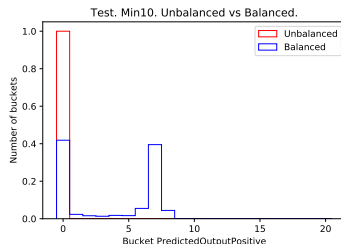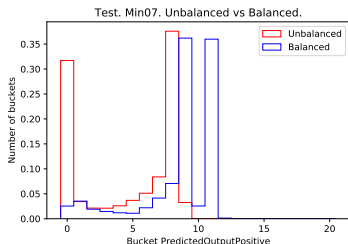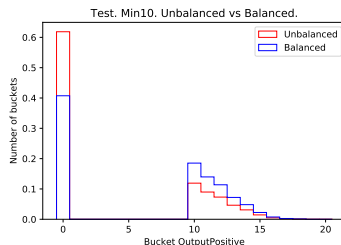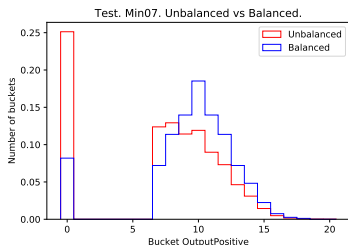
Metrics for each VolumeID overlay two methods 1/2.
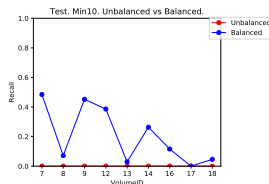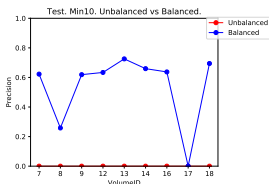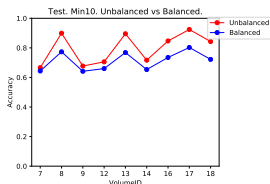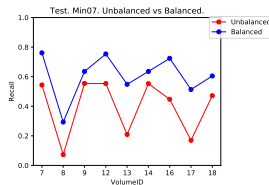
o Min00 and Min04 very similar.

Accuracy	Precision	Recall
$\dfrac{\text{TP+TN}}{\text{TP+FP+FN+TN}}$	$\dfrac{\text{TP}}{\text{TP+FP}}$	$\dfrac{\text{TP}}{\text{TP+FN}}$

o

o Min10 learns all hits to be negative, so precision and recall at zero.

Accuracy	Precision	Recall
$\dfrac{TP+TN}{TP+FP+FN+TN}$	$\dfrac{TP}{TP+FP}$	$\dfrac{TP}{TP+FN}$
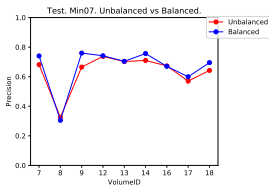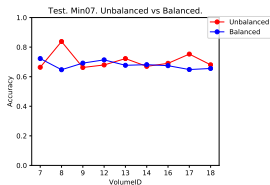
Metrics for each VolumeID with min value used.

- Min00 and Min04 very similar.
- Min10 learns all hits to be negative, so precision and recall at zero.

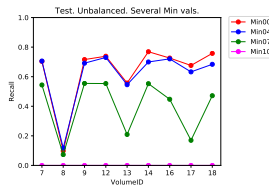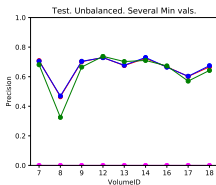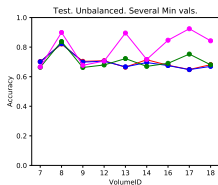| Accuracy	Precision	Recall
$\dfrac{TP+TN}{TP+FP+FN+TN}$	$\dfrac{TP}{TP+FP}$	$\dfrac{TP}{TP+FN}$
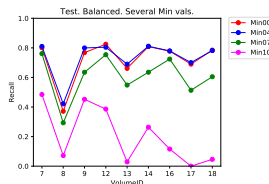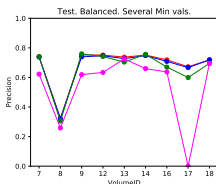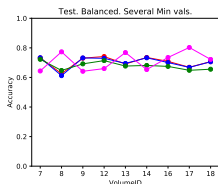
Conclusion. Future plans.

o Conclusions:
- Compared two methods: Unbalanced vs Balanced.
- Balanced: remove buckets such that number of buckets with a given nbPositiveHit is symmetric around 10.
- 0 and 1 no buckets → 19 and 20 are balanced to no buckets (ignored).
- For each Unbalanced and Balanced, did Min00, Min04, Min07, Min10.
- As seen before, Unbalanced Min10 learns all hits to be negative, so precision and recall are zero.
- Min00 and Min04 are very similar.
- Min07 in between Min04 and Min10.
- **Overall, balancing buckets improve performance.**

o Future plans:
- Balance buckets in majority particle η. Is it replacing the balancing of nbPositiveHit, or in addition to it?