FW X Machina on FPGA for L1 trigger

1. Classification with boosted decision trees
2. Regression with deep boosted decision trees
3. Anomaly detection with decision tree-based autoencoder

Today ⇨

Tae Min Hong
University of Pittsburgh

Workshop // on // Fast Realtime Systems // and // Realtime Machine Learning
April 9, 2024
https://indico.belle2.org/event/10782/contributions/75182/

FW X Machina Papers

Classification

Published by IOP Publishing for Sissa Medialab

Received: *April 9, 2021*
Accepted: *June 29, 2021*
Published: *August 4, 2021*

Nanosecond machine learning event classification with boosted decision trees in FPGA for high energy physics

T.M. Hong,* B.T. Carlson, B.R. Eubanks, S.T. Racz, S.T. Roche, J. Stelzer and D.C. Stumpp

Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.

E-mail: tmhong@pitt.edu

Abstract: We present a novel implementation of classification using the machine learning/artificial intelligence method called boosted decision trees (BDT) on field programmable gate arrays (FPGA). The firmware implementation of binary classification requiring 100 training trees with a maximum depth of 4 using four input variables gives a latency value of about 10 ns, independent of the clock speed from 100 to 320 MHz in our setup. The low timing values are achieved by restructuring the BDT layout and reconfiguring its parameters. The FPGA resource utilization is also kept low at a range from 0.01% to 0.2% in our setup. A software package called fwXmachina achieves this implementation. Our intended user is an expert in custom electronics-based trigger systems in high energy physics experiments or anyone that needs decisions at the lowest latency values for real-time event classification. Two problems from high energy physics are considered, in the separation of electrons vs. photons and in the selection of vector boson fusion-produced Higgs bosons vs. the rejection of the multijet processes.

Keywords: Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software); Data reduction methods

ArXiv ePrint: 2104.03408

*Corresponding author.

https://doi.org/10.1088/1748-0221/16/08/P08016

2021 JINST 16 P08016

Hong et al., JINST **16**, P08016 (2021)
http://doi.org/10.1088/1748-0221/16/08/P08016

Regression + deep

Published by IOP Publishing for Sissa Medialab

Received: *July 13, 2022*
Accepted: *August 23, 2022*
Published: *September 27, 2022*

Nanosecond machine learning regression with deep boosted decision trees in FPGA for high energy physics

B.T. Carlson,[a,b] Q. Bayer,[b] T.M. Hong[b,*] and S.T. Roche[b]

[a] *Department of Physics and Engineering, Westmont College, 955 La Paz Road, Santa Barbara, CA 93108, U.S.A.*
[b] *Department of Physics and Astronomy, University of Pittsburgh, 100 Allen Hall, 3941 O'Hara St., Pittsburgh, PA 15260, U.S.A.*

E-mail: tmhong@pitt.edu

Abstract: We present a novel application of the machine learning / artificial intelligence method called boosted decision trees to estimate physical quantities on field programmable gate arrays (FPGA). The software package fwXmachina features a new architecture called parallel decision paths that allows for deep decision trees with arbitrary number of input variables. It also features a new optimization scheme to use different numbers of bits for each input variable, which produces optimal physics results and ultraefficient FPGA resource utilization. Problems in high energy physics of proton collisions at the Large Hadron Collider (LHC) are considered. Estimation of missing transverse momentum ($E_{\mathrm{T}}^{\mathrm{miss}}$) at the first level trigger system at the High Luminosity LHC (HL-LHC) experiments, with a simplified detector modeled by Delphes, is used to benchmark and characterize the firmware performance. The firmware implementation with a maximum depth of up to 10 using eight input variables of 16-bit precision gives a latency value of $O(10)$ ns, independent of the clock speed, and $O(0.1)$% of the available FPGA resources without using digital signal processors.

Keywords: Data reduction methods; Digital electronic circuits; Trigger algorithms; Trigger concepts and systems (hardware and software)

ArXiv ePrint: 2207.05602

*Corresponding author.

https://doi.org/10.1088/1748-0221/17/09/P09039

2022 JINST 17 P09039

Carlson et al., JINST **17**, P09039 (2022)
http://doi.org/10.1088/1748-0221/17/09/P09039

Anomaly detection

PITT-PACC-2311

Nanosecond anomaly detection with decision trees for high energy physics and real-time application to exotic Higgs decays

S.T. Roche[a,b], Q. Bayer[b], B.T. Carlson[b,c], W.C. Ouligian[b], P. Serhiayenka[b], J. Stelzer[b], and T.M. Hong*[b]

[a] School of Medicine, Saint Louis University
[b] Department of Physics and Astronomy, University of Pittsburgh
[c] Department of Physics and Engineering, Westmont College

April 11, 2023

Abstract

We present a novel implementation of the artificial intelligence autoencoding algorithm, used as an ultrafast and ultraefficient anomaly detector, built with a forest of deep decision trees on FPGA, field programmable gate arrays. Scenarios at the Large Hadron Collider at CERN are considered, for which the autoencoder is trained using known physical processes of the Standard Model. The design is then deployed in real-time trigger systems for anomaly detection of new unknown physical processes, such as the detection of exotic Higgs decays, on events that fail conventional threshold-based algorithms. The inference is made within a latency value of 25 ns, the time between successive collisions at the Large Hadron Collider, at percent-level resource usage. Our method offers anomaly detection at the lowest latency values for edge AI users with tight resource constraints.

Keywords: Data processing methods, Data reduction methods, Digital electronic circuits, Trigger algorithms, and Trigger concepts and systems (hardware and software).

*Corresponding author, tmhong@pitt.edu

Roche et al., accepted for publication
https://arxiv.org/abs/2304.03836

Outline

Introduction

- Autoencoders for anomaly detection
- Machine learning at L1

Decision tree autoencoder

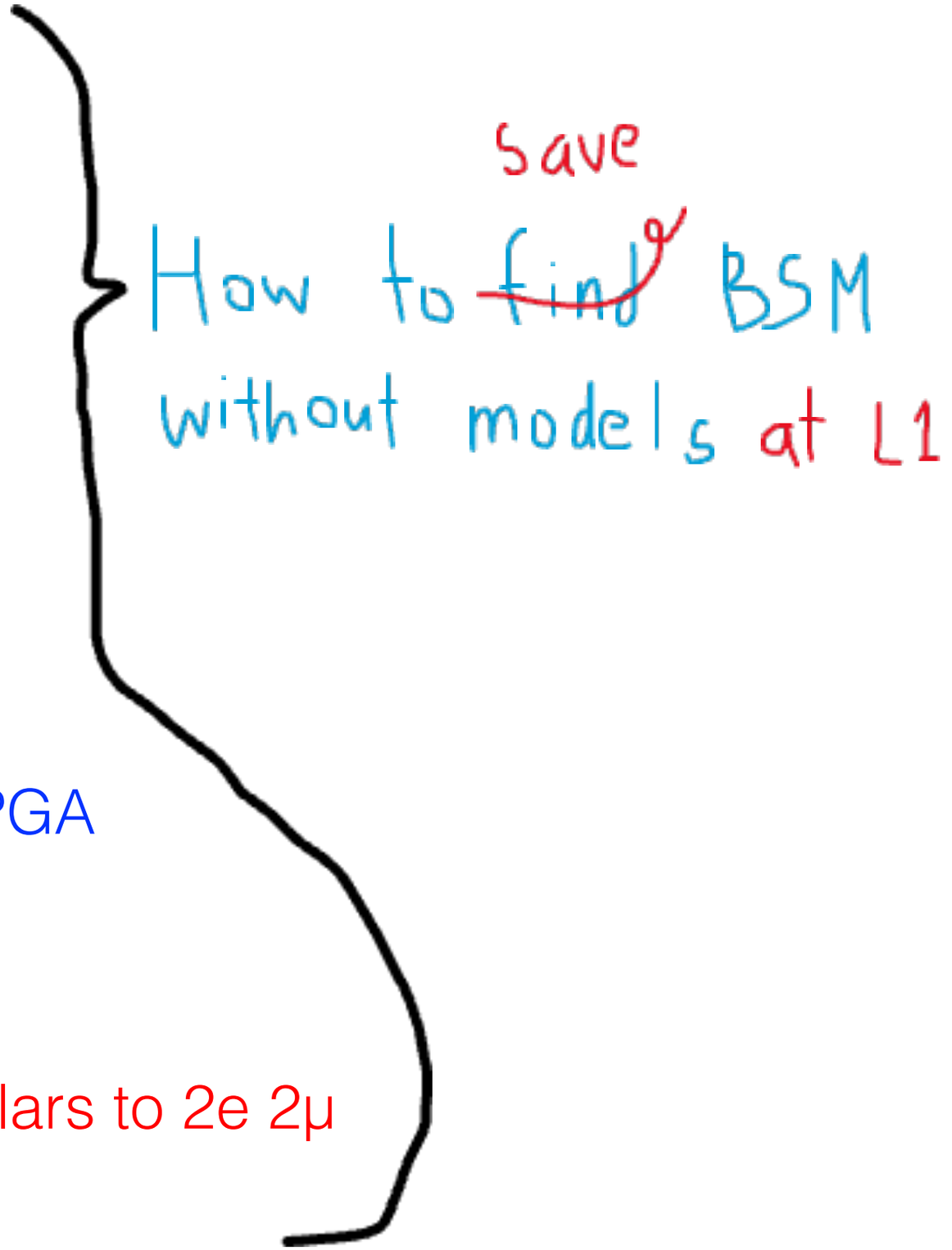
- Novel training method

Firmware design

- Novel latent-spaceless design for FPGA

Physics & FPGA results

- Exotic decay of Higgs to pseudoscalars to 2e 2μ
- “LHC anomaly detection” dataset



Prior work

TM Hong

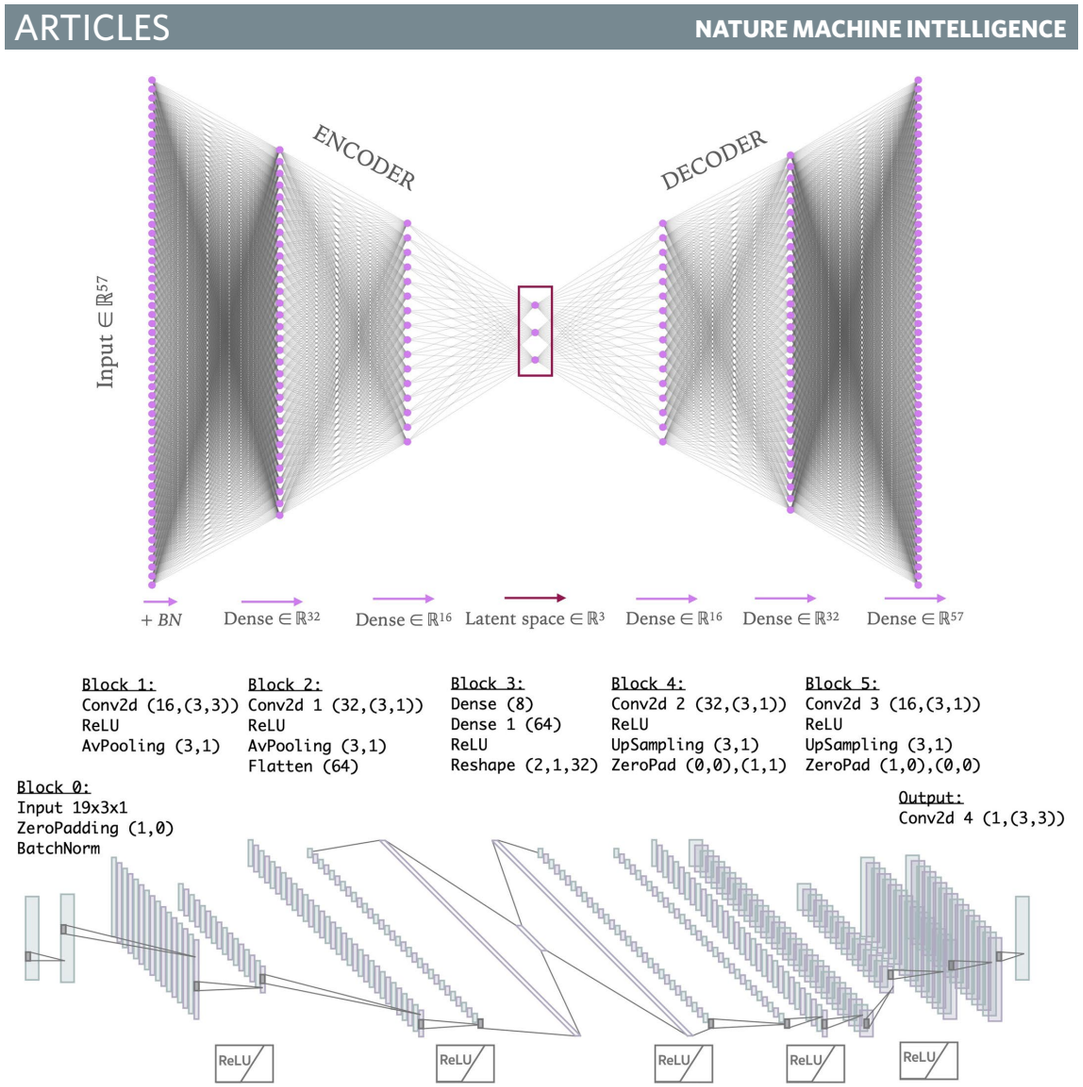
Autoencoder

- Typically constructed using neural networks
- Challenge to implement in pure digital logic on FPGA
- NN example shown on right →

Decision tree?

- Used in our work
- Has certain advantages: technical (no multiplication) & philosophical (interpretable)

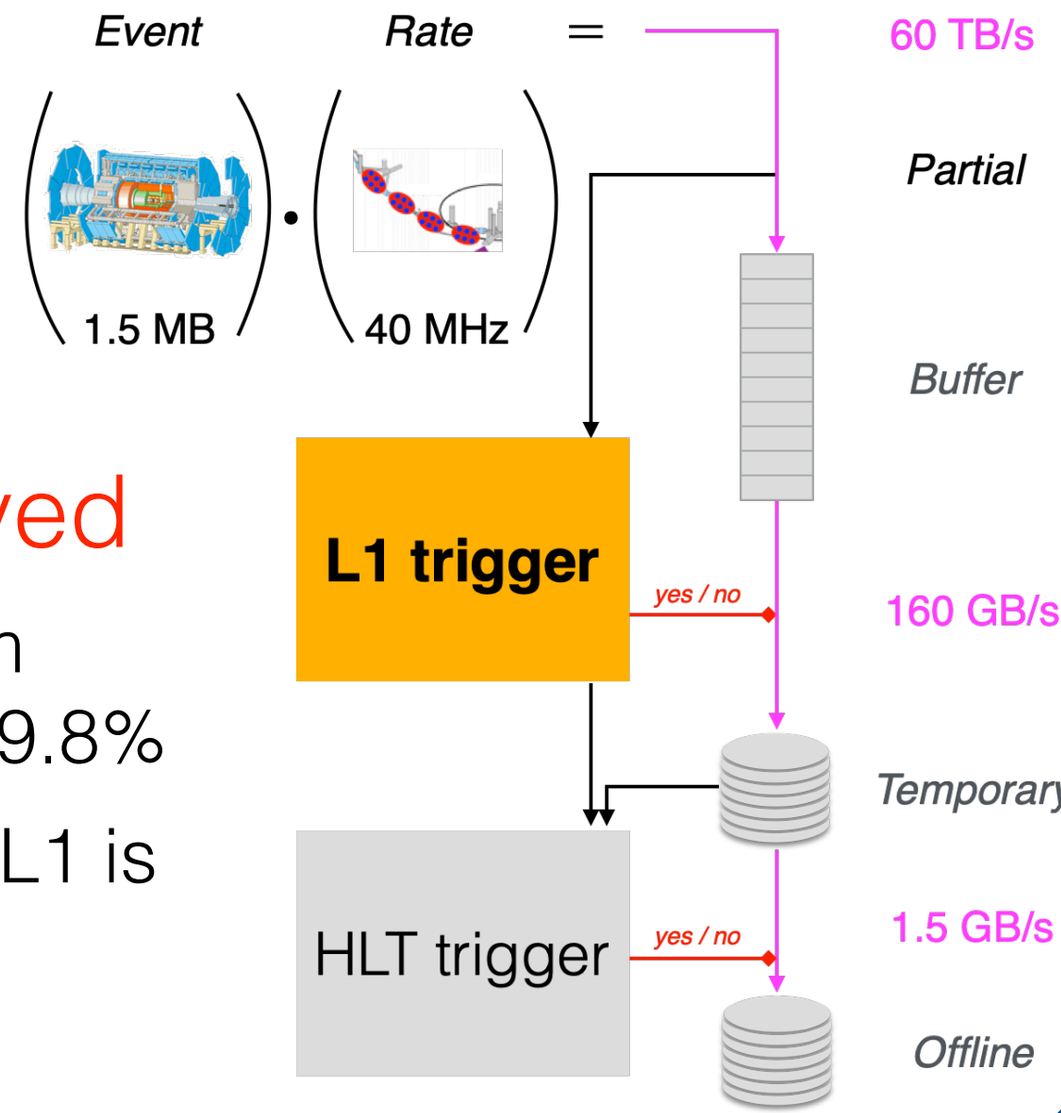
Govorkova et al., *Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 MHz at the Large Hadron Collider,* Nature Mach. Intell. **4** (2022) 154–161
https://doi.org/10.1038/s42256-022-00441-3



Extended Data Fig. 1 | Network architectures. Network architecture for the DNN AE (top) and CNN AE (bottom) models. The corresponding VAE models are derived introducing the Gaussian sampling in the latent space, for the same encoder and decoder architectures (see text).

Model-agnostic detection of BSM signals

- Many anomaly detection methods have been devised and tested on a variety of different HEP problems [https://iml-wg.github.io/HEPML-LivingReview]
- Anomaly detection in ATLAS analysis [ATLAS-CONF-2022-045]



Can’t analyze data that’s not saved

- L1 triggers at ATLAS & CMS use custom electronics such as FPGAs to discard 99.8%
- Implementing anomaly detection at the L1 is challenging and possible (this talk)

Source: http://cern.ch/twiki/pub/Atlas/TDAQSpeakersCommitteeCommonReferences/tdaqFullNew2017.pdf

Decision trees

FW X Machina

ML on FPGA

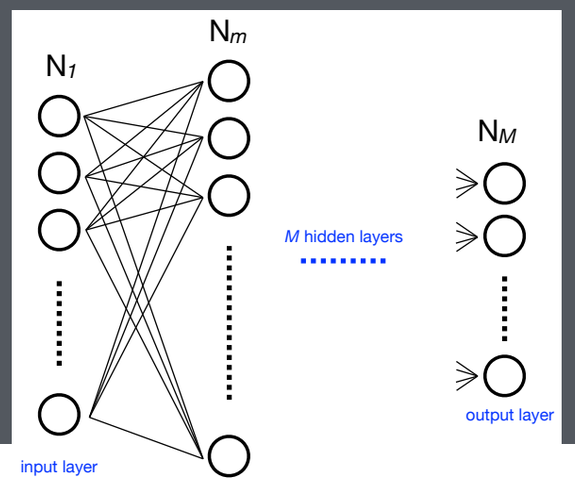
NN vs. DT

[1] Denby, Comp. Phys. Comm. 49-3, 429 (1988)
[2] Duarte et al., J. Instrum. 13, P07027 (2018)
[3] CMS Collaboration, Phys. Lett. B 716, 31 (2012)
[4] Summers et al., J. Instrum 15, P02056 (2020)
[5] Hong et al., J. Instrum. 16, P08016 (2021)
[6] Carlson et al., J. Instrum. 17, P09039 (2022)

- **Neural Network**

Popular	Been around HEP since the 80s[1]
Depth	Challenging, so ~3 on FPGA[2]
Score	$y = \Theta(\mathbb{M} \cdot x + b)$

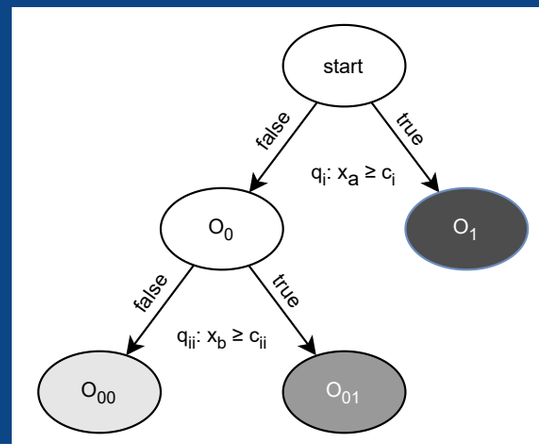
Activation *Multiplication*



- **Decision Tree**

Popular	Discovered the Higgs![3]
Depth	Challenging, so 4 to 8 on FPGA[4,5,6]
Score	$y = \Theta(x < \text{threshold})$

Step fn *Comparison*



- **FWX Decision Tree**

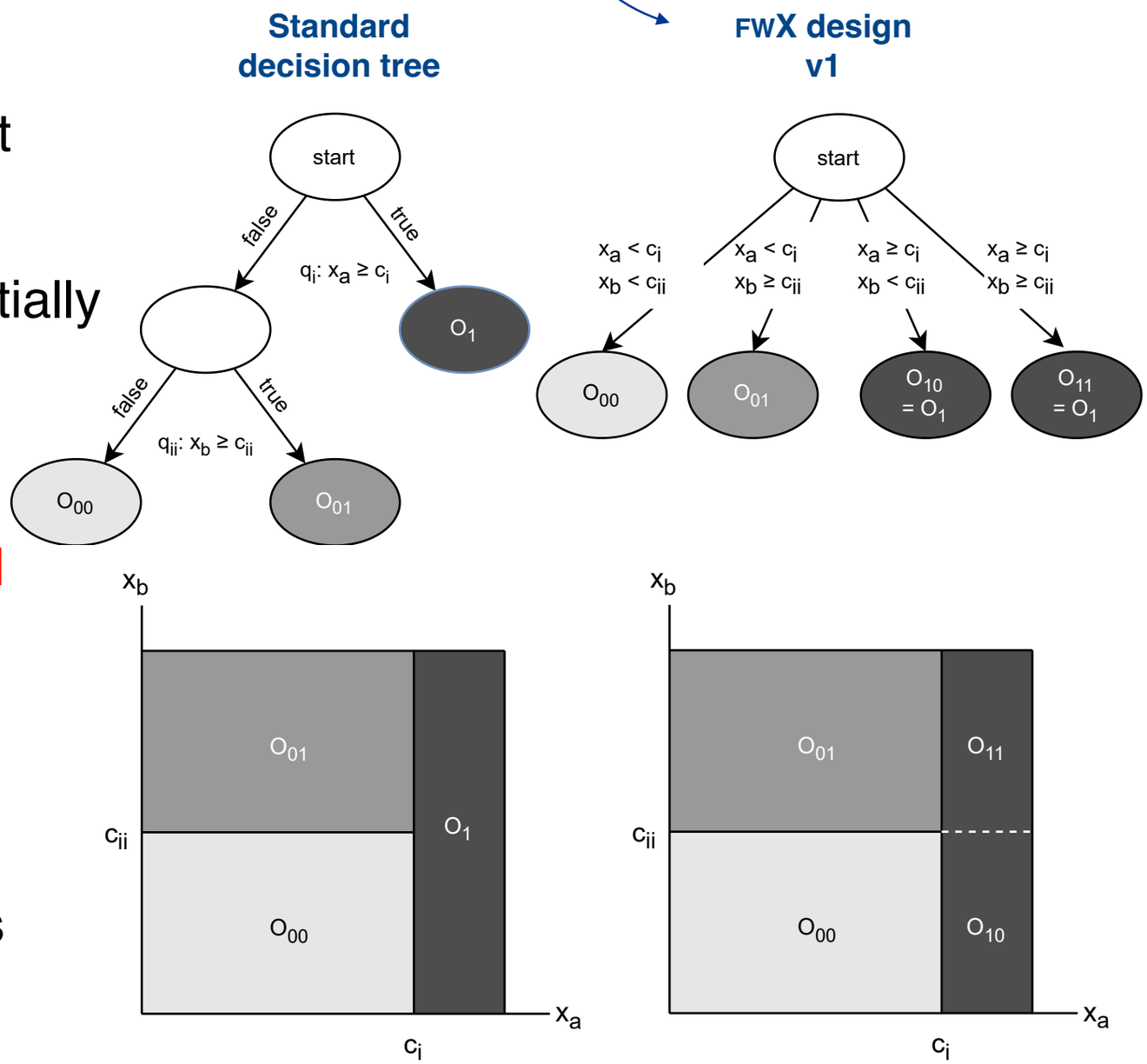
Physics	Comparable results vs. NN on FPGA
~~Float / fixed~~	Bit integer → bit shifts → efficient
Optimized	Parallelize → one step → low latency

FW X Machina

Design v1: Parallelize cuts

2 variable example

- **General**
  - Training — Use TMVA or equivalent
  - Design — Threshold comparisons
  - Challenge — Evaluate layers sequentially
- **FWXv1**
  - Key design — Evaluate cuts in parallel
  - Benefit — Each cut is indpd't
    → Bin search on a grid
    → Bit shift to speed-up
  - Limitations — Does not scale well w/ tree depth & # variables
  - Follow-up — Led to v2 design



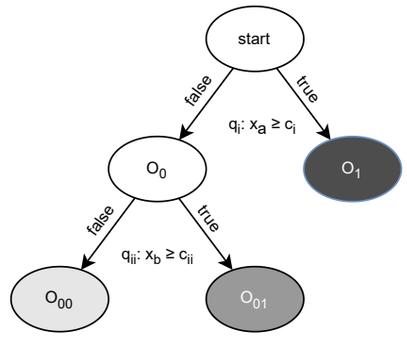
Hong et al., JINST 16, P08016 (2021)

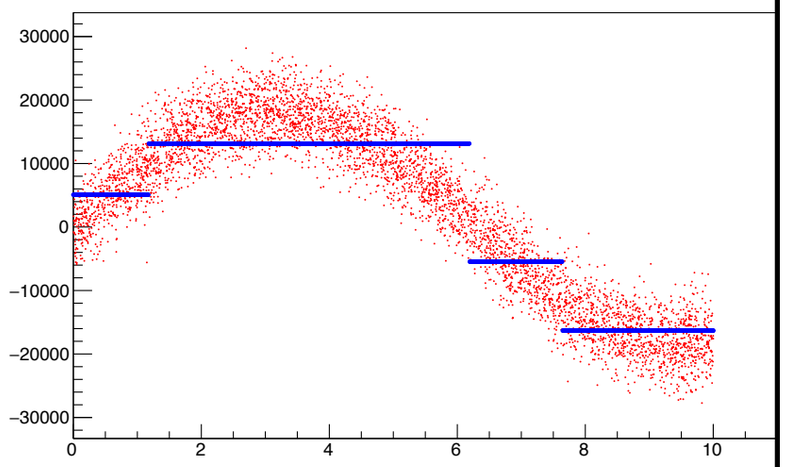
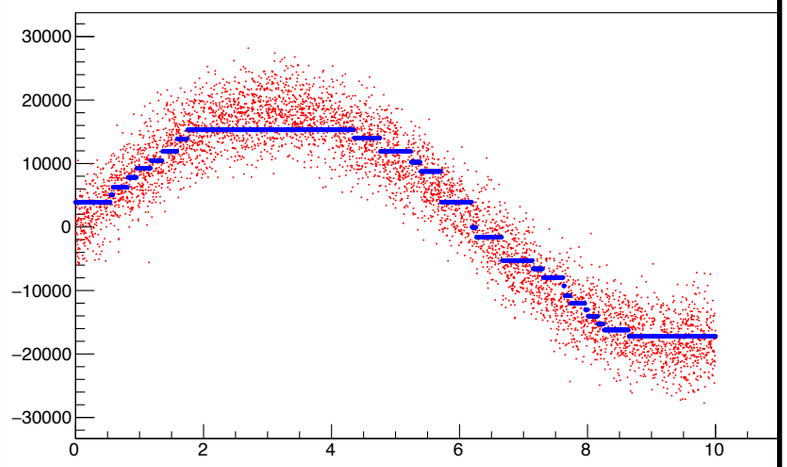
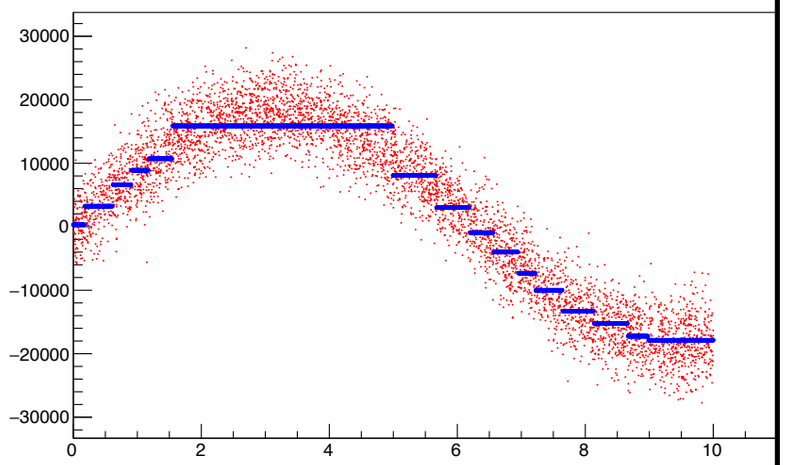
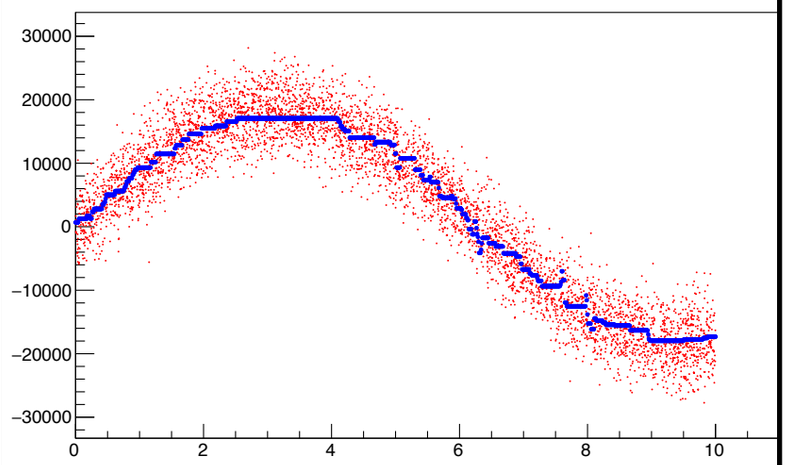
Regression

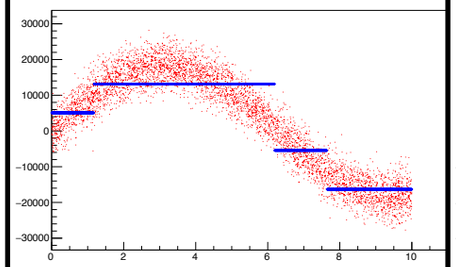
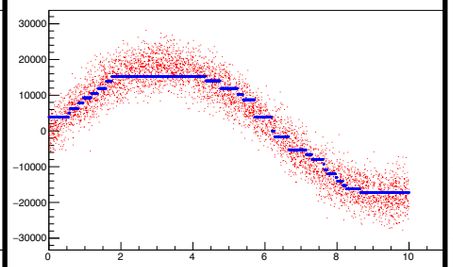
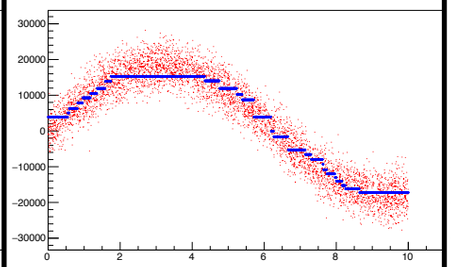
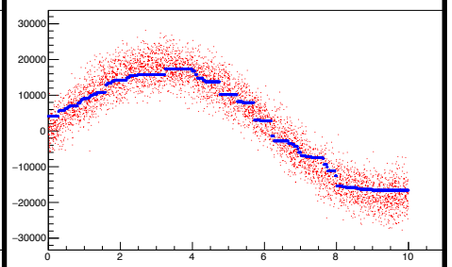
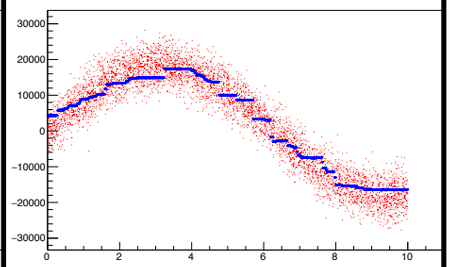
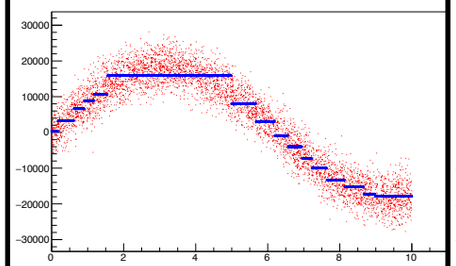
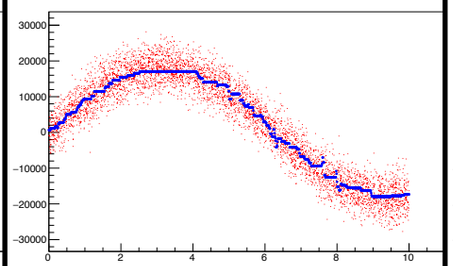
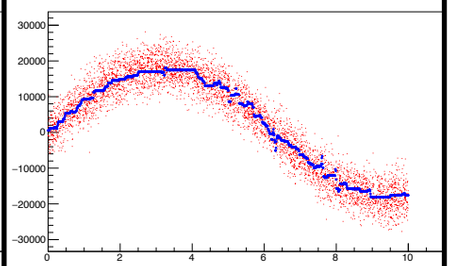
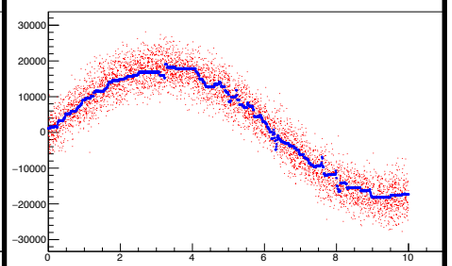
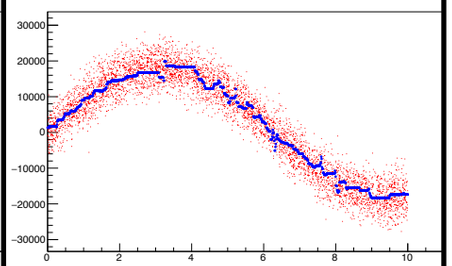
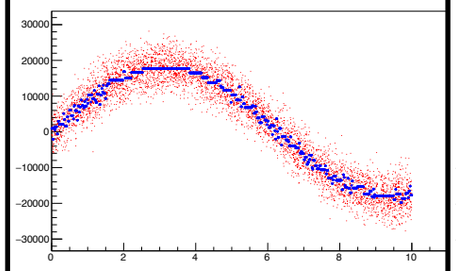
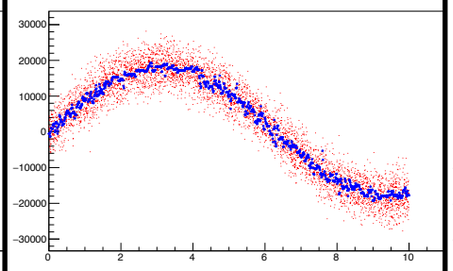
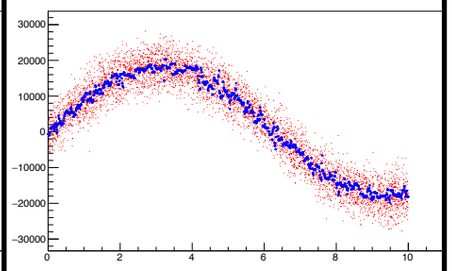
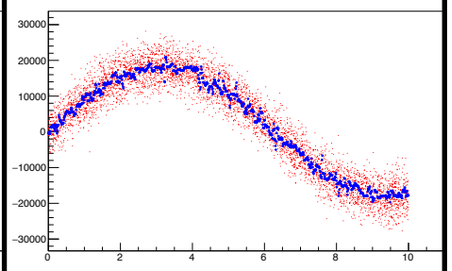
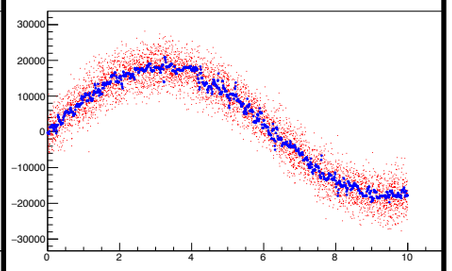
TM Hong

Look-up table

- Toy problem in 1-d
- Train / test on f(x) = sin(x) + Gaussian(x)
- For sample of x: y = f(x) in 16 bits



	$N_{tree} = 1$	$N_{tree} = 10$
$N_{depth} = 2$		
$N_{depth} = 4$		

$(x,y) = (N_{tree}, N_{depth})$	Ntree = 1	10	25	100	400
2					
4					
8					

FWX Machina

Design v2: Parallelize terminal bins

Go deeper from 4 → 8

- **Improve FWXv1**

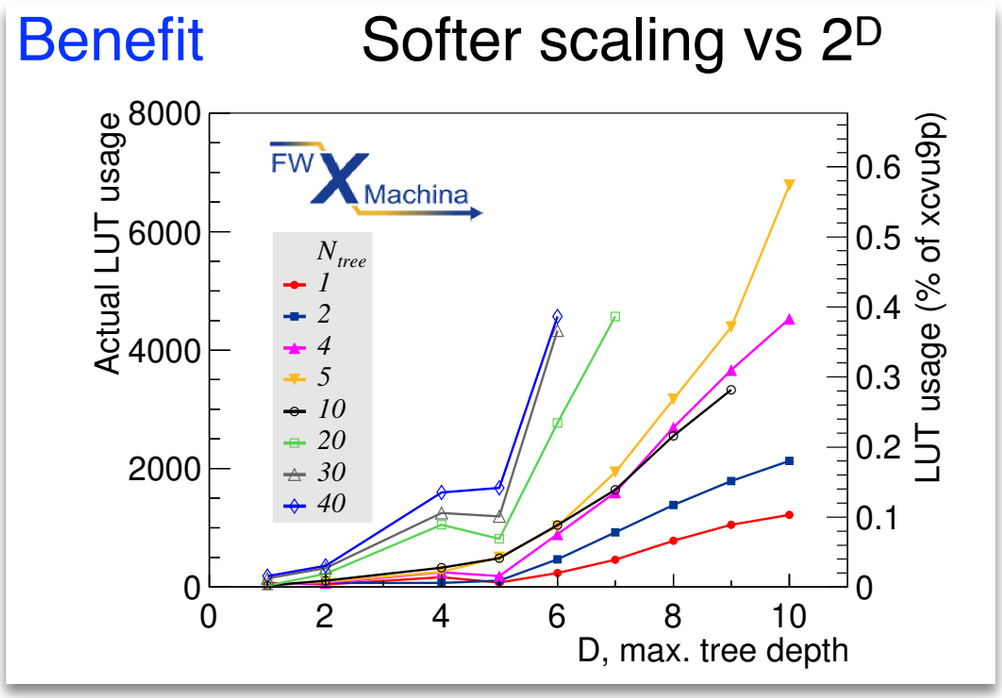
  Challenge — Does not scale well w/ tree depth & # variables

  Cut redundancy 2^D

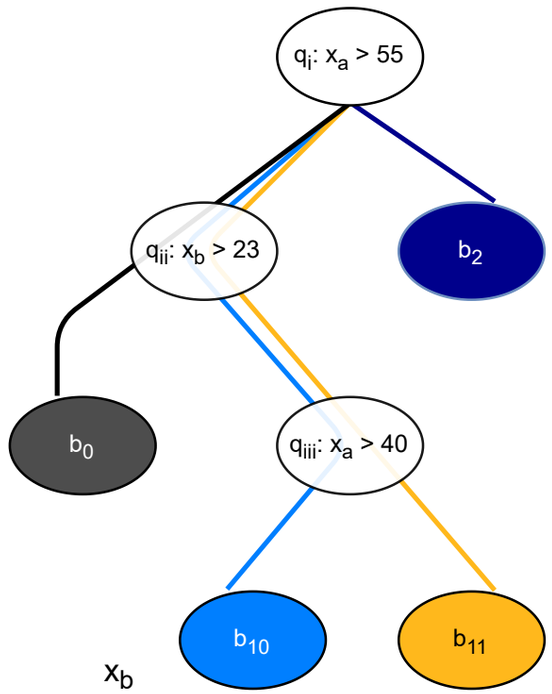
- **FWXv2**

  Key design — Evaluate decision paths

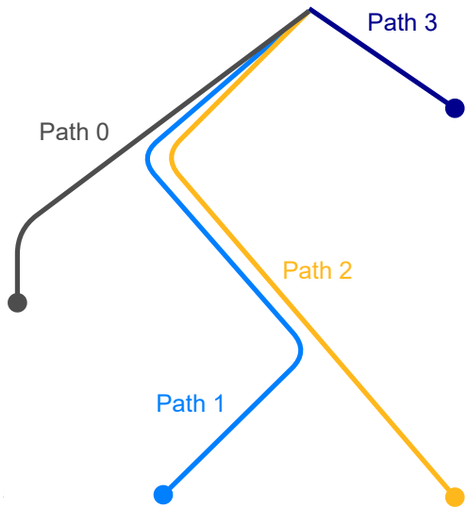
  Benefit — Softer scaling vs 2^D

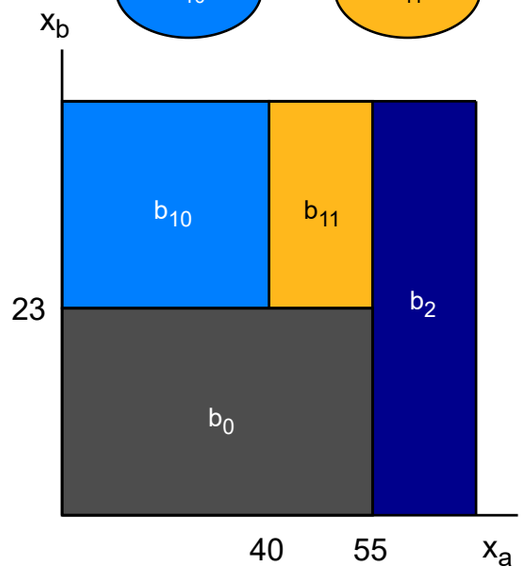


Standard decision tree



FWX design v2





Destination bin	Decision path
b_0	not(q_i) and not(q_{ii})
b_2	q_i
b_{10}	not(q_i) and q_{ii} and not(q_{iii})
b_{11}	not(q_i) and q_{ii} and q_{iii}

Carlson et al., JINST 17, P09039 (2022)

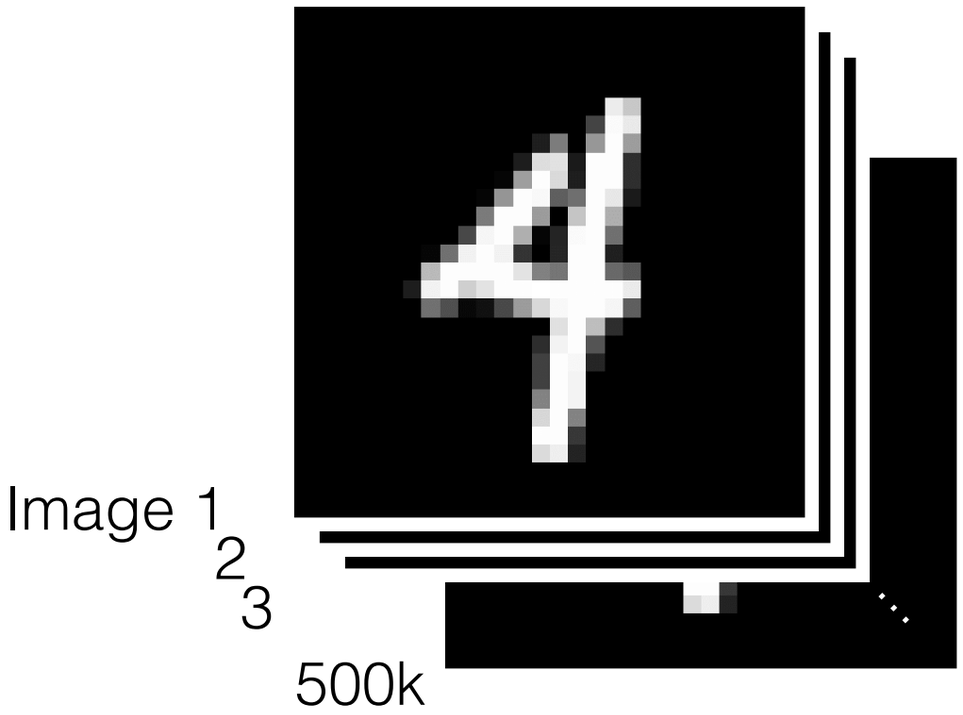
Autoencoder

Tree-based

Example: handwritten numbers

- Teach it about the number 4

784 variables (8-bit)



=

Corresponding data set

Image	Pixel 1	Pixel 2	...	Pixel 300	...	Pixel 783	Pixel 784
1	0	0	...	240	...	0	0
2	0	1	...	255	...	0	0
...	...	...	...	...	...	...	...
500k	0	0	...	231	...	0	0

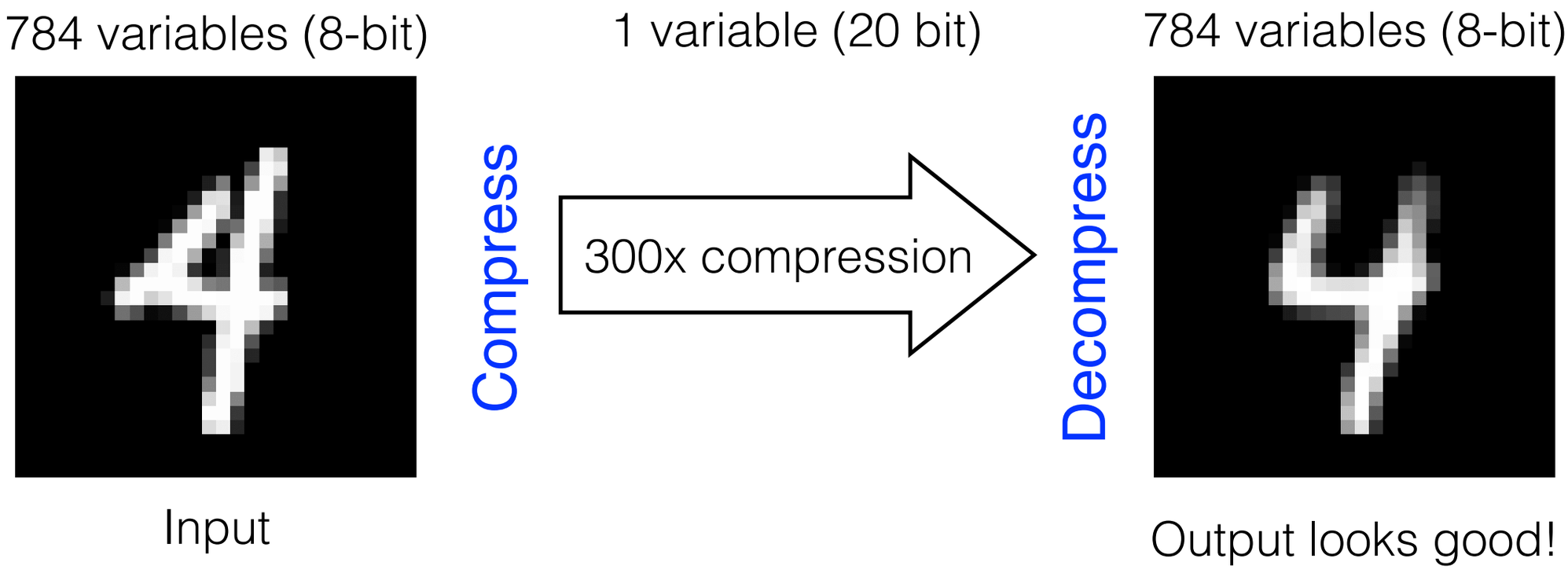
Details

- Each pixel in the data set are unrelated to each other

TM Hong

Example: handwritten numbers

- Teach it 0, 1, 2, 3, 4 with a sample

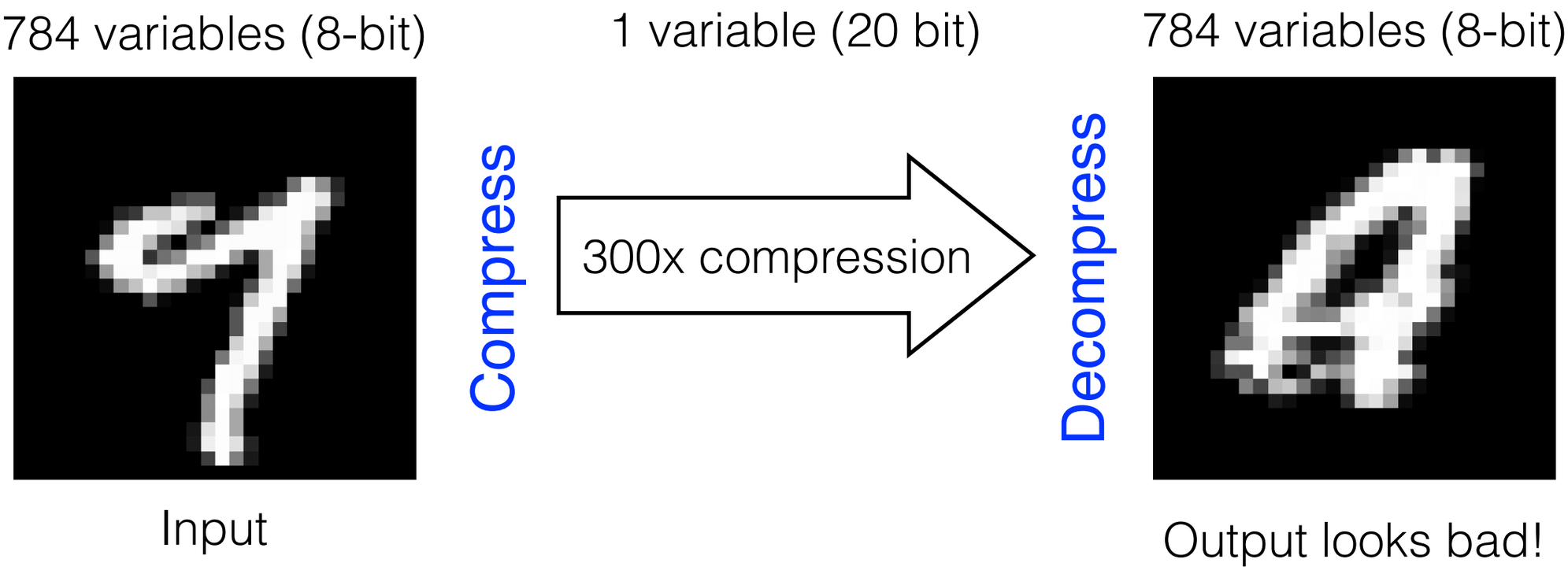


Details

- Input-output distance is relatively small = good compression

Example: handwritten numbers

- Teach it 0, 1, 2, 3, 4 with a sample (doesn't know about 9!)



Details

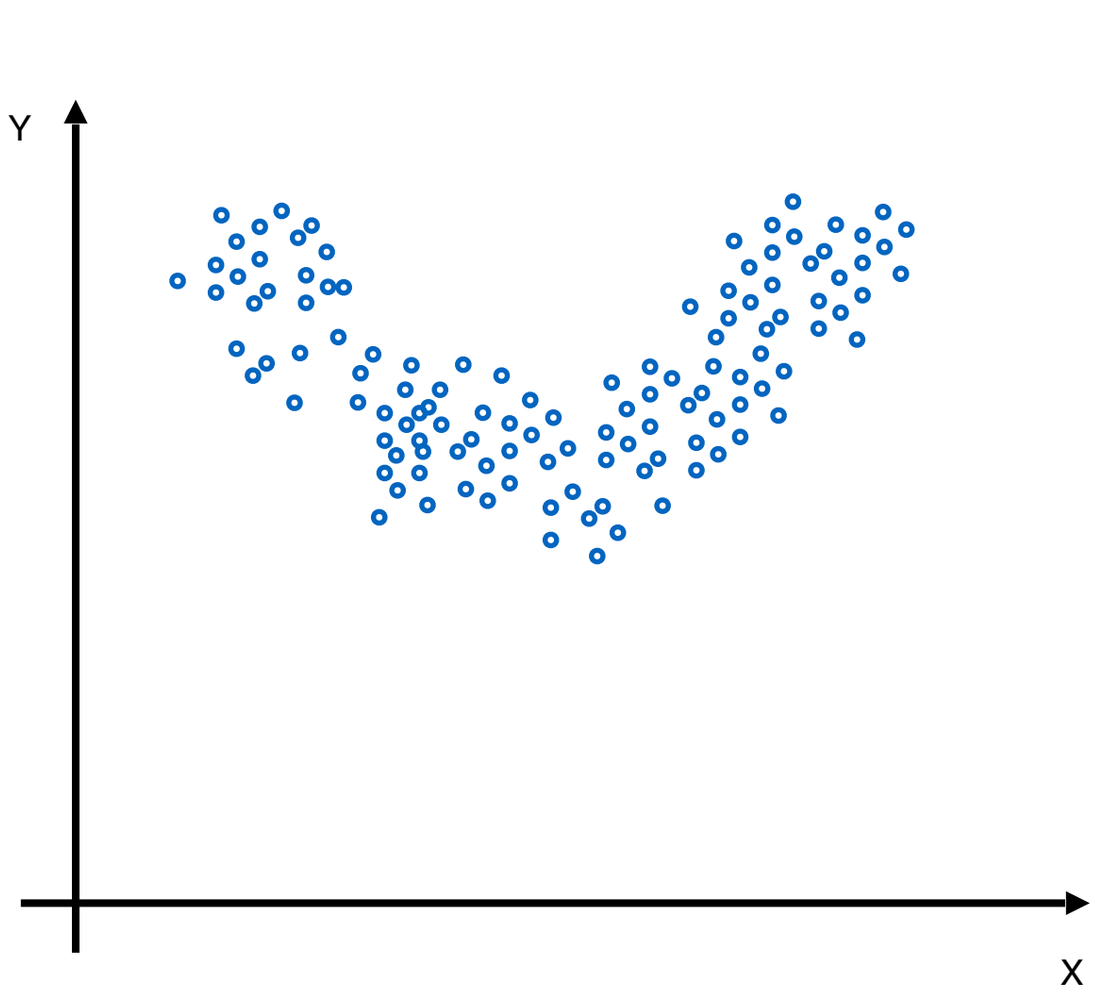
- Input-output distance is relatively large = bad compression

Decision tree autoencoders

TM Hong

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Example
  - 2d toy dataset, say x = p_T and y = eta for some SM sample

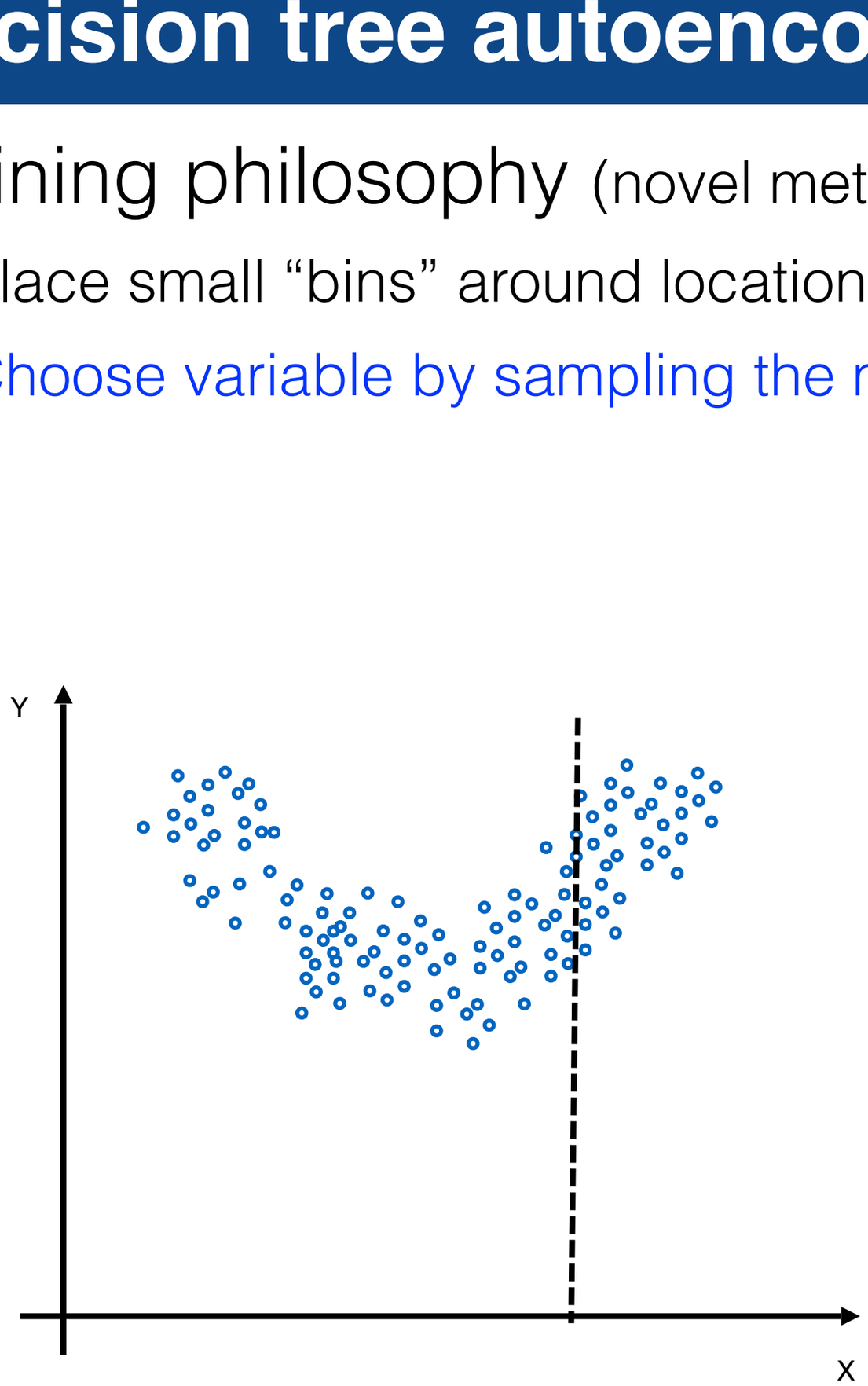


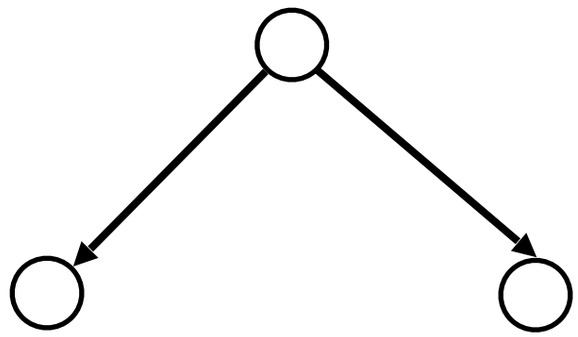
Decision tree autoencoders

TM Hong

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Choose variable by sampling the max of the distributions



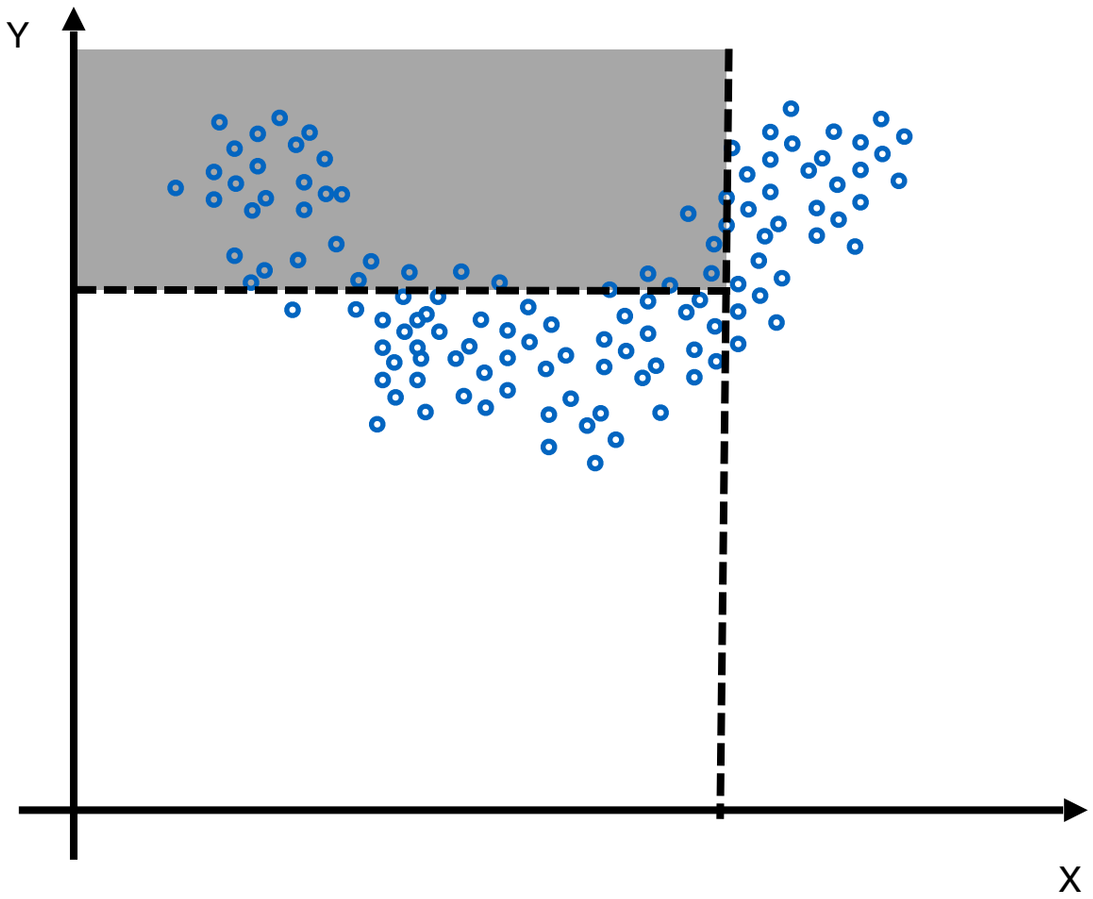


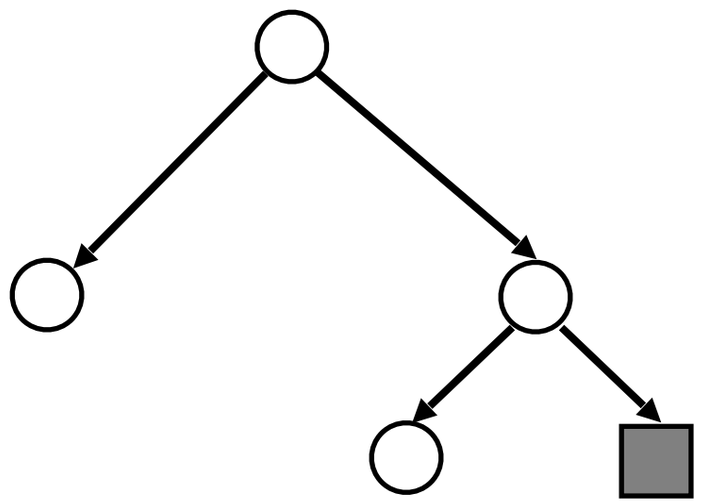
Decision tree autoencoders

TM Hong

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Sample the variable for a cut, then repeat



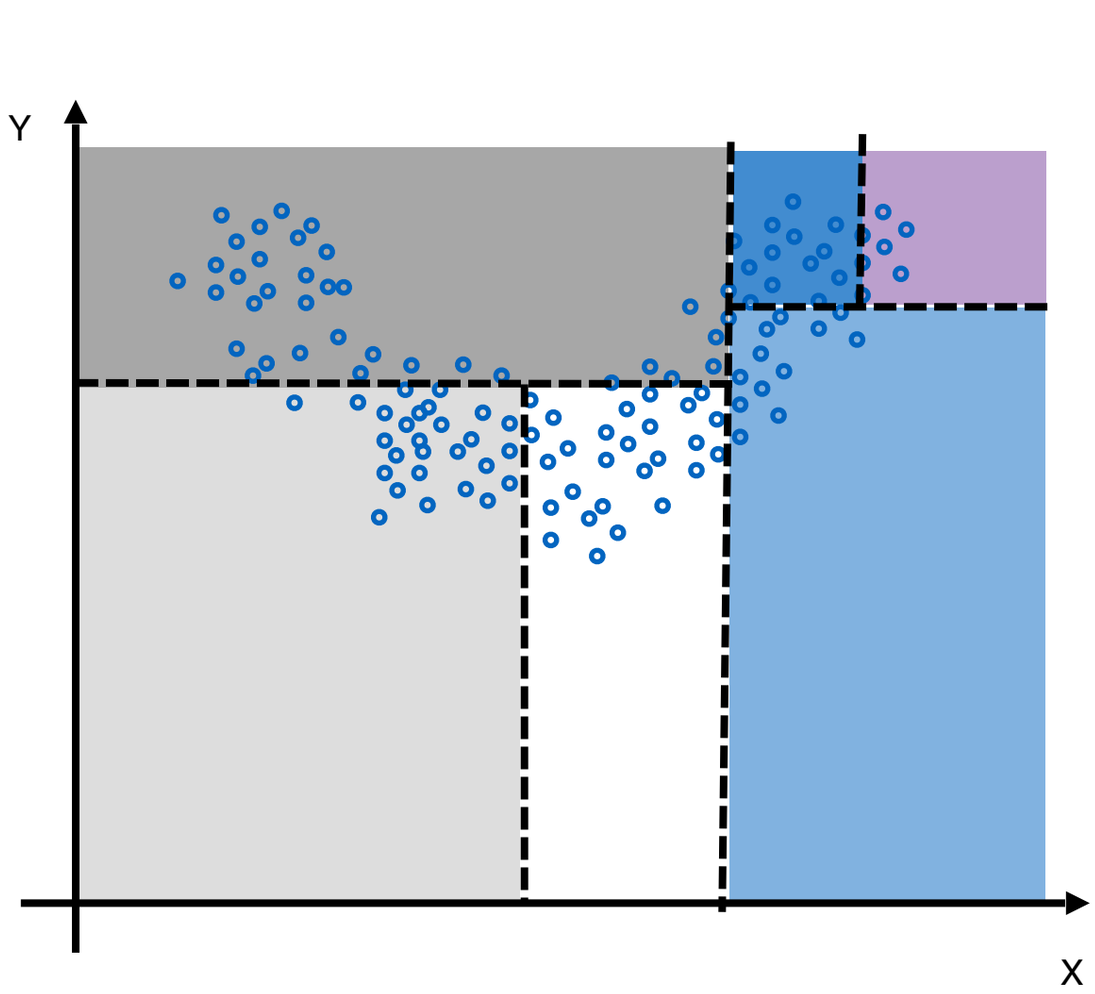


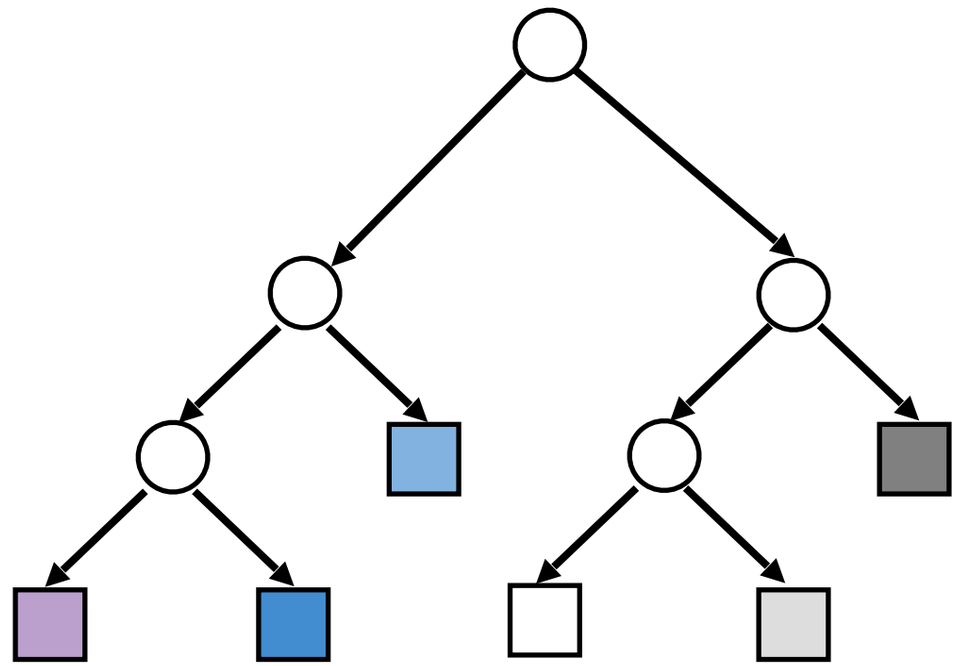
Decision tree autoencoders

TM Hong

Training philosophy (novel method described in paper)

- Place small “bins” around locations of high event density
- Iteratively repeat for subsamples





Decision tree autoencoders

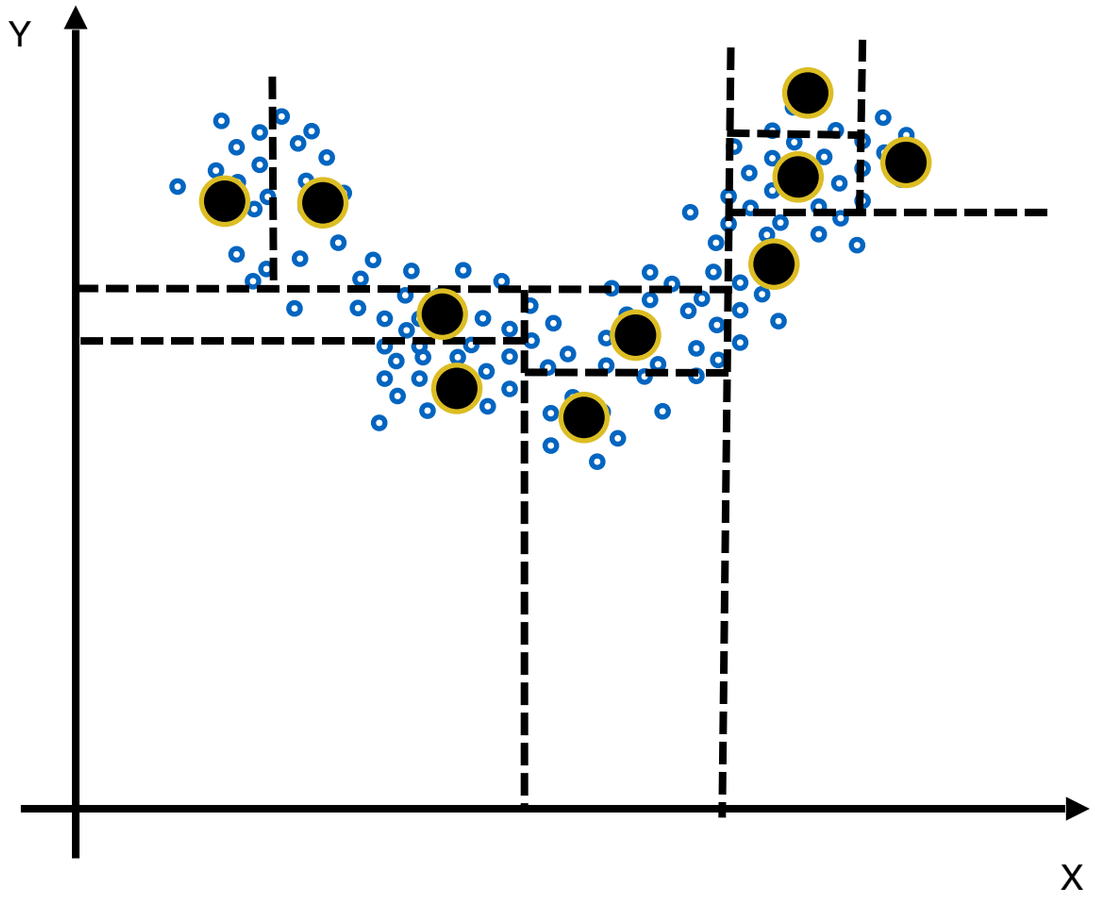
TM Hong

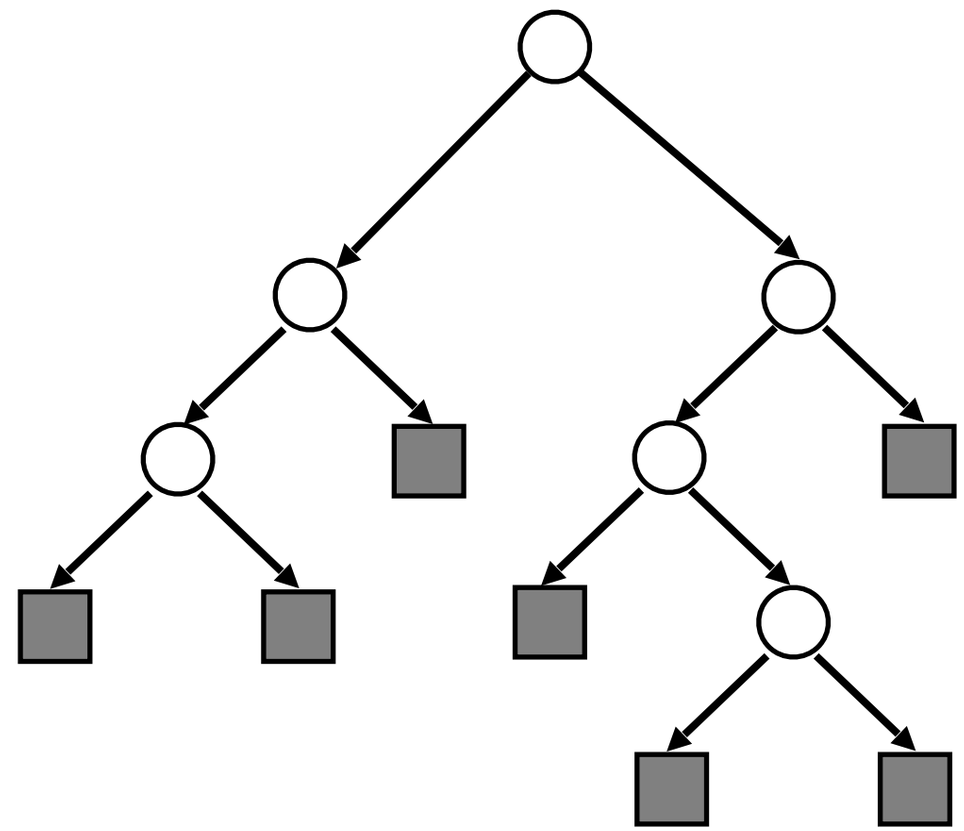
Latent space is bin number

- Encoding: Event → which bin it's in

Decode by returning a "reconstruction point"

- Decoding: Bin → median of the training data in bin



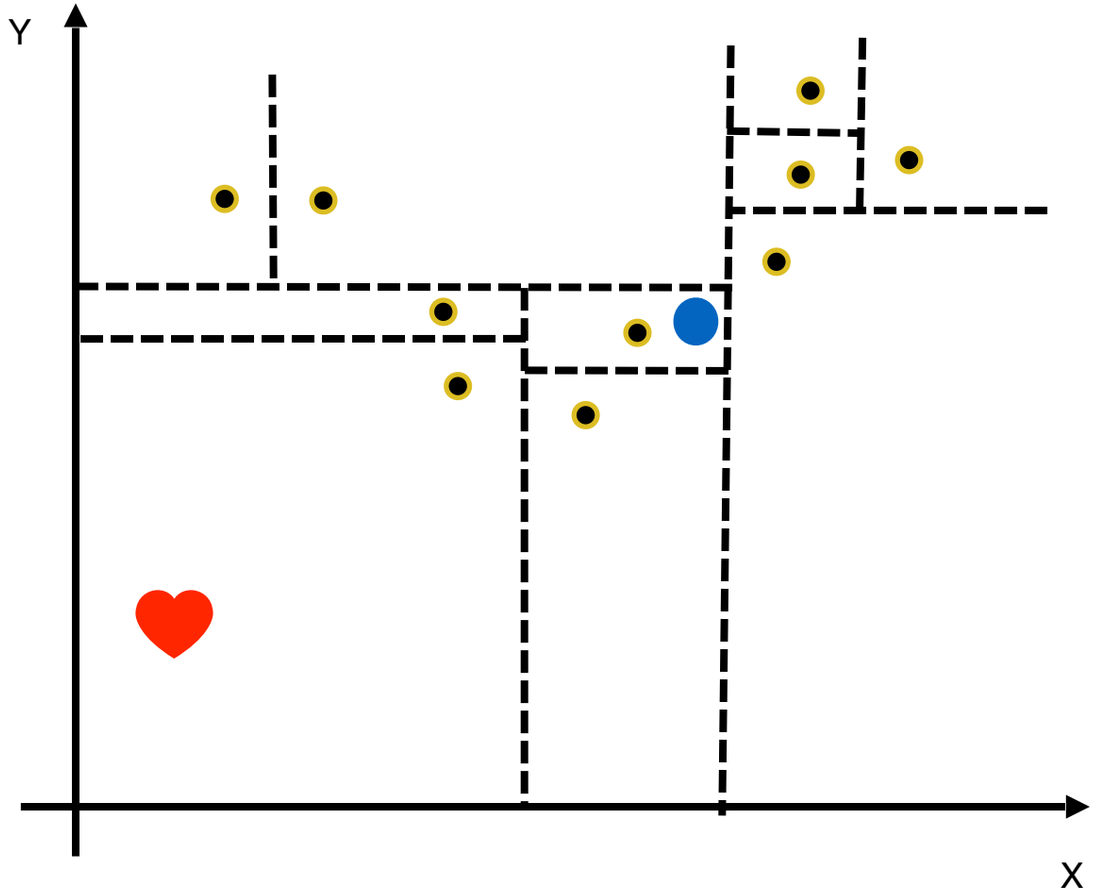


Decision tree autoencoders

TM Hong

How does this detect anomalies?

- Define: Distance between input – output = anomaly score

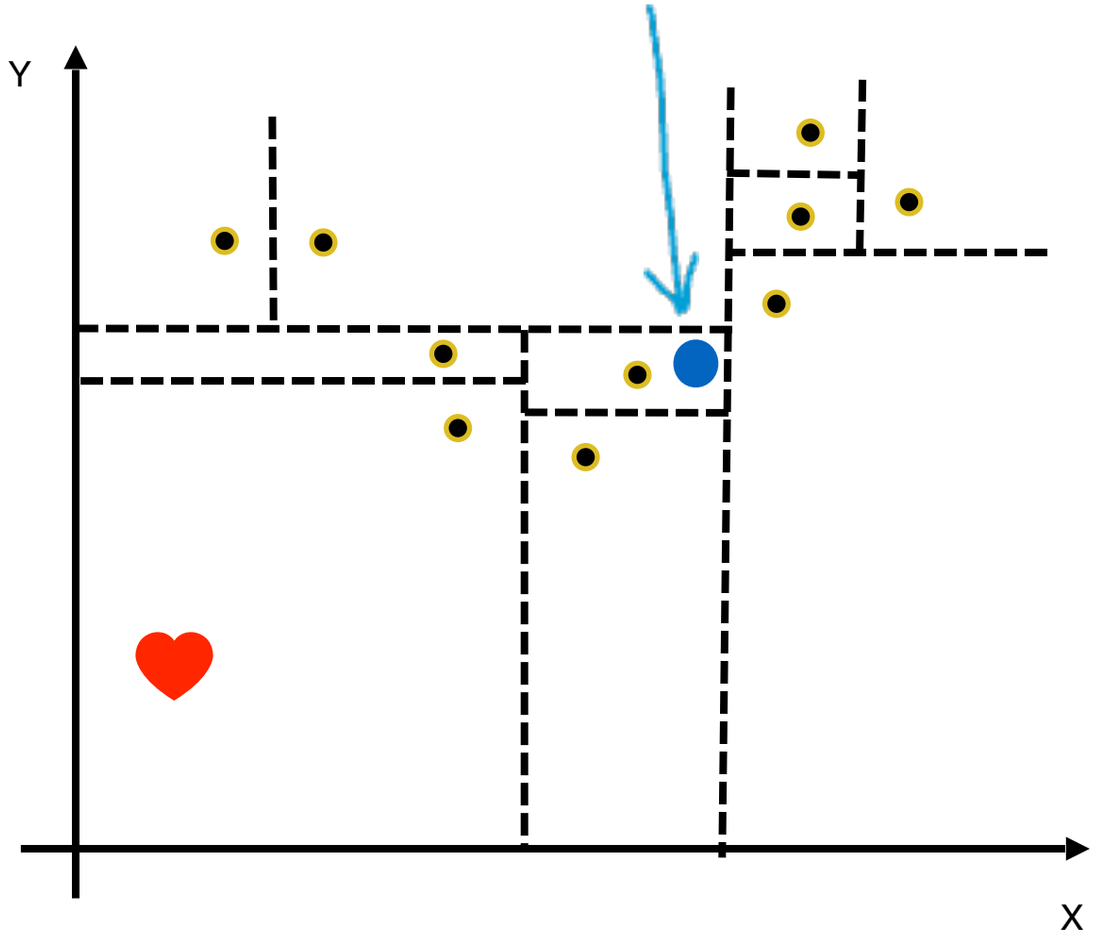


Decision tree autoencoders

TM Hong

How does this detect anomalies?

- Define: Distance between input – output = anomaly score
- Non-anomaly
  - Input is similar to training data
  - Will likely land in a small bin → close to reconstruction point

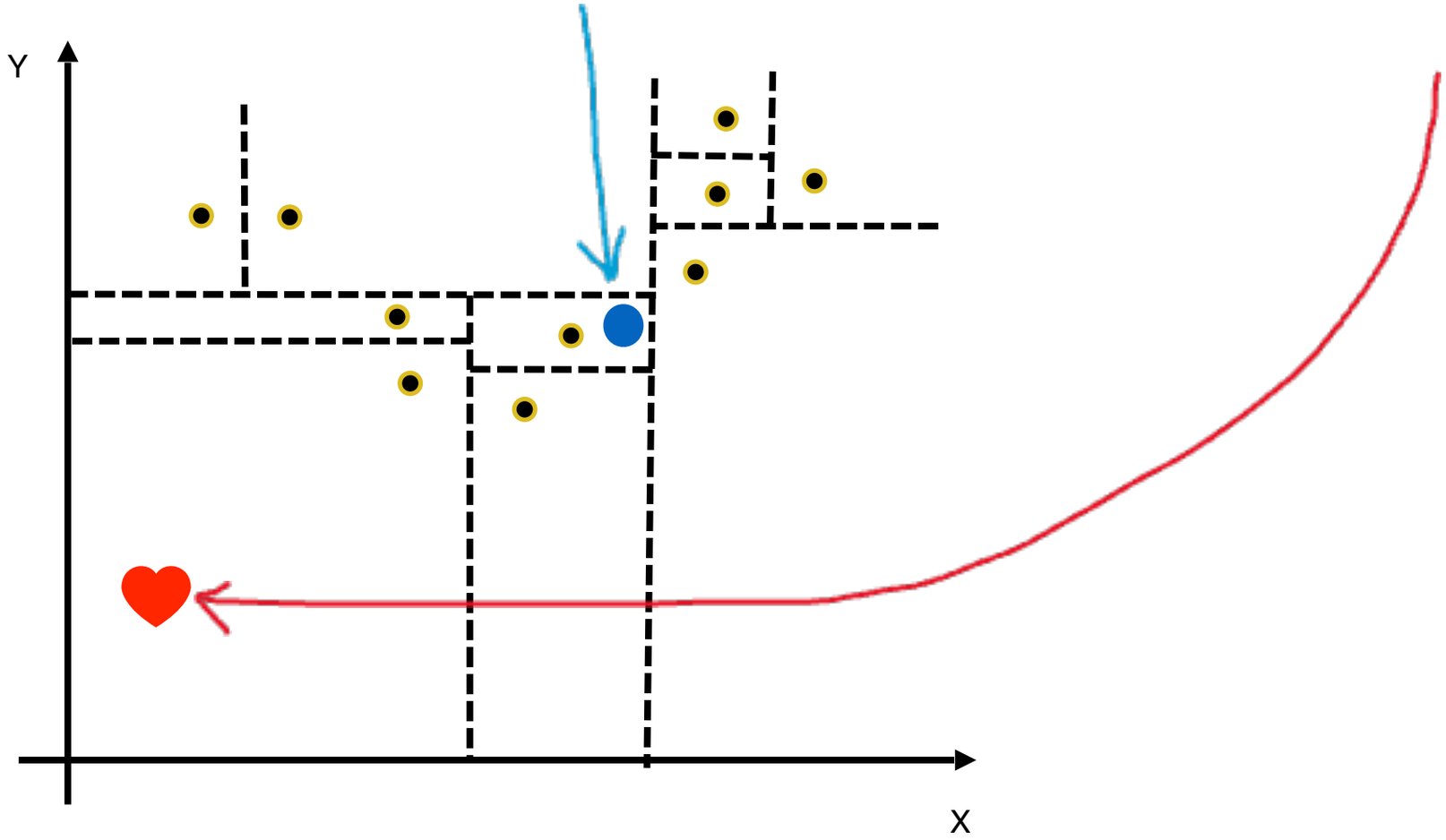


Decision tree autoencoders

TM Hong

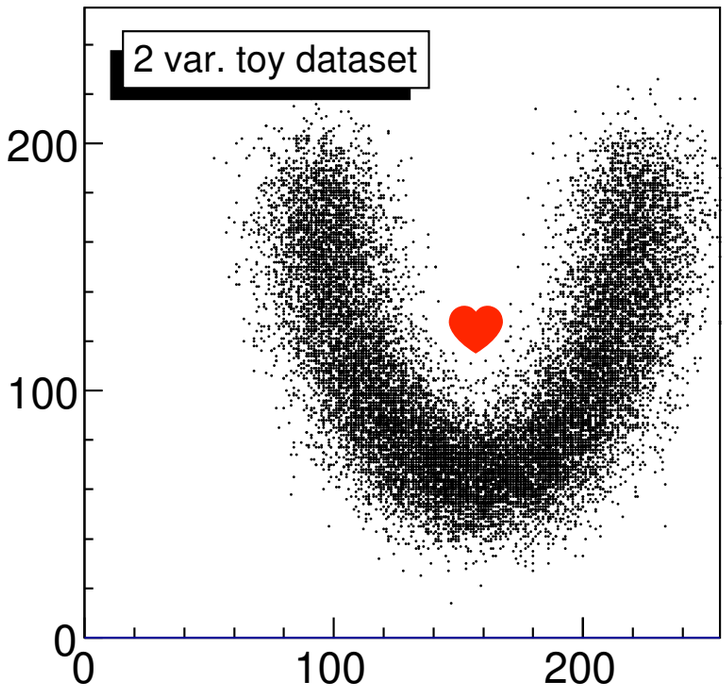
How does this detect anomalies?

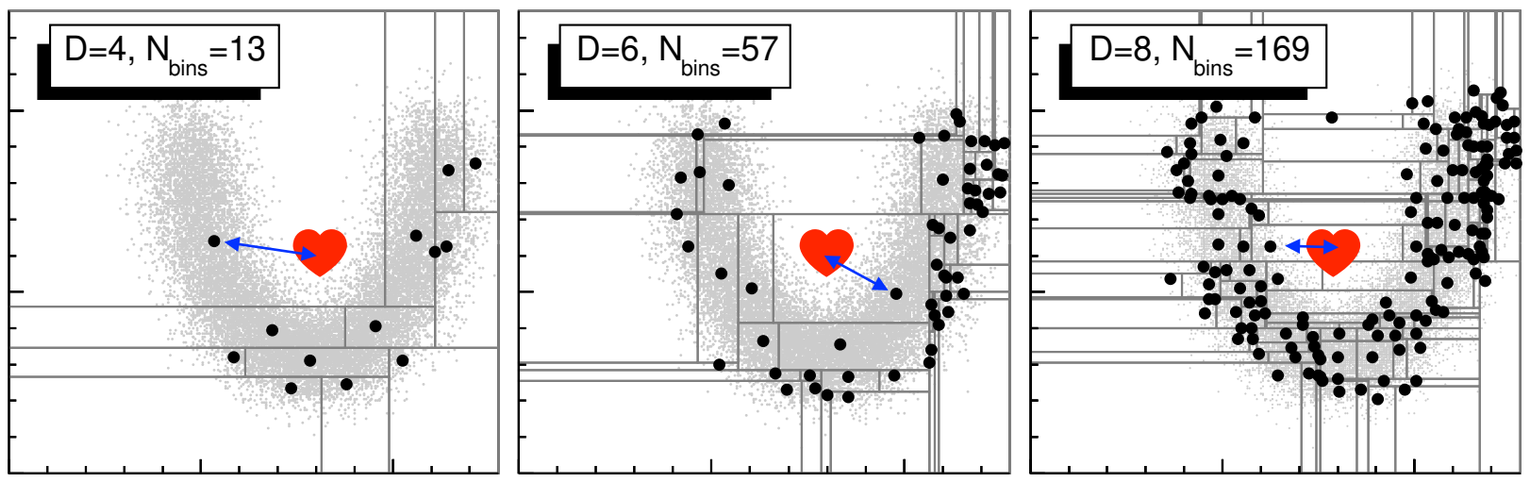
- Define: Distance between input – output = anomaly score
- Non-anomaly
  - Input is similar to training data
  - Will likely land in a small bin → close to the reconstruction point
- Anomaly
  - Input is not similar to training data
  - Will likely land in a large bin → far from the reconstruction point



Toy dataset (2 input variables)

TM Hong

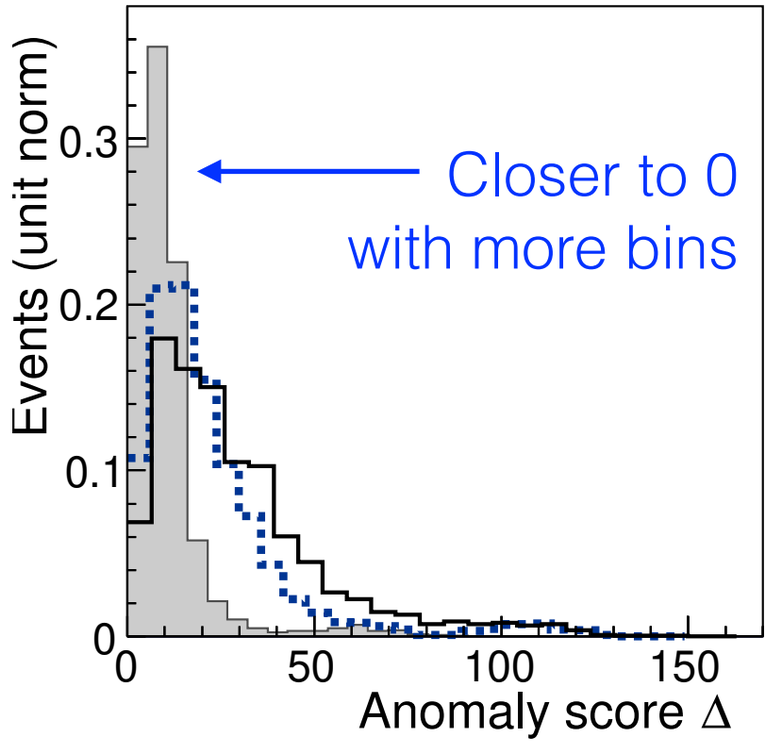




more bins

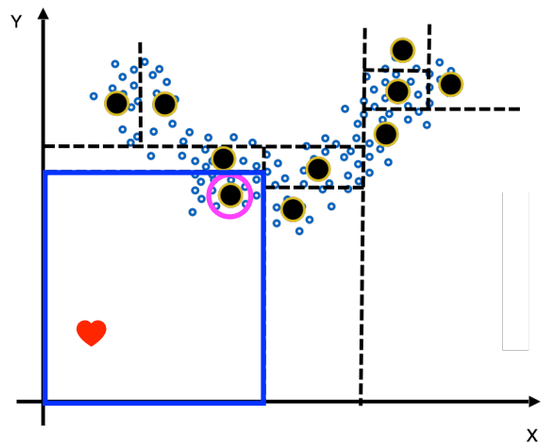
Anomaly score

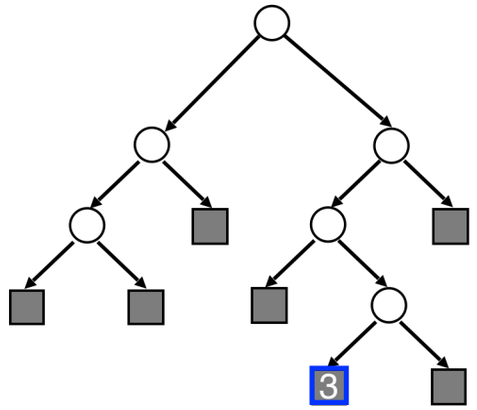
- Feed back in the training sample
  - Should be near 0, like E_T^{miss} resolution



Latent spaceless implementation

- Closer look at what it means to encode

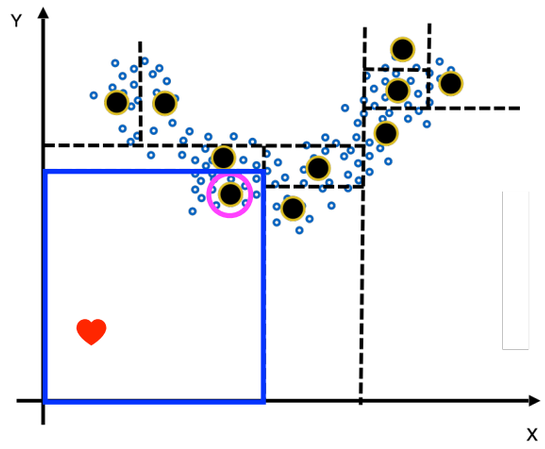


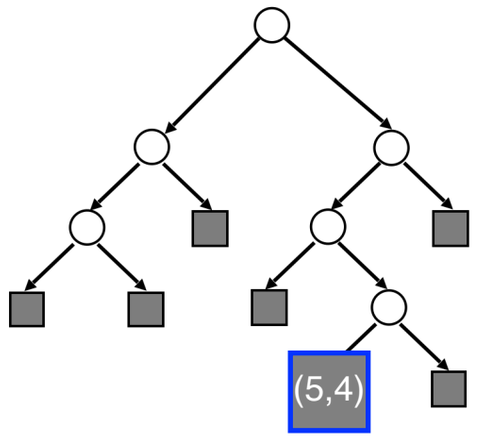


Bin	x	y
1	...	...
2	...	...
3	5	4
...		

Incoming heart → Encode: return bin 3 → Decode bin 3: return (5,4)

- Skip the encoding & decoding





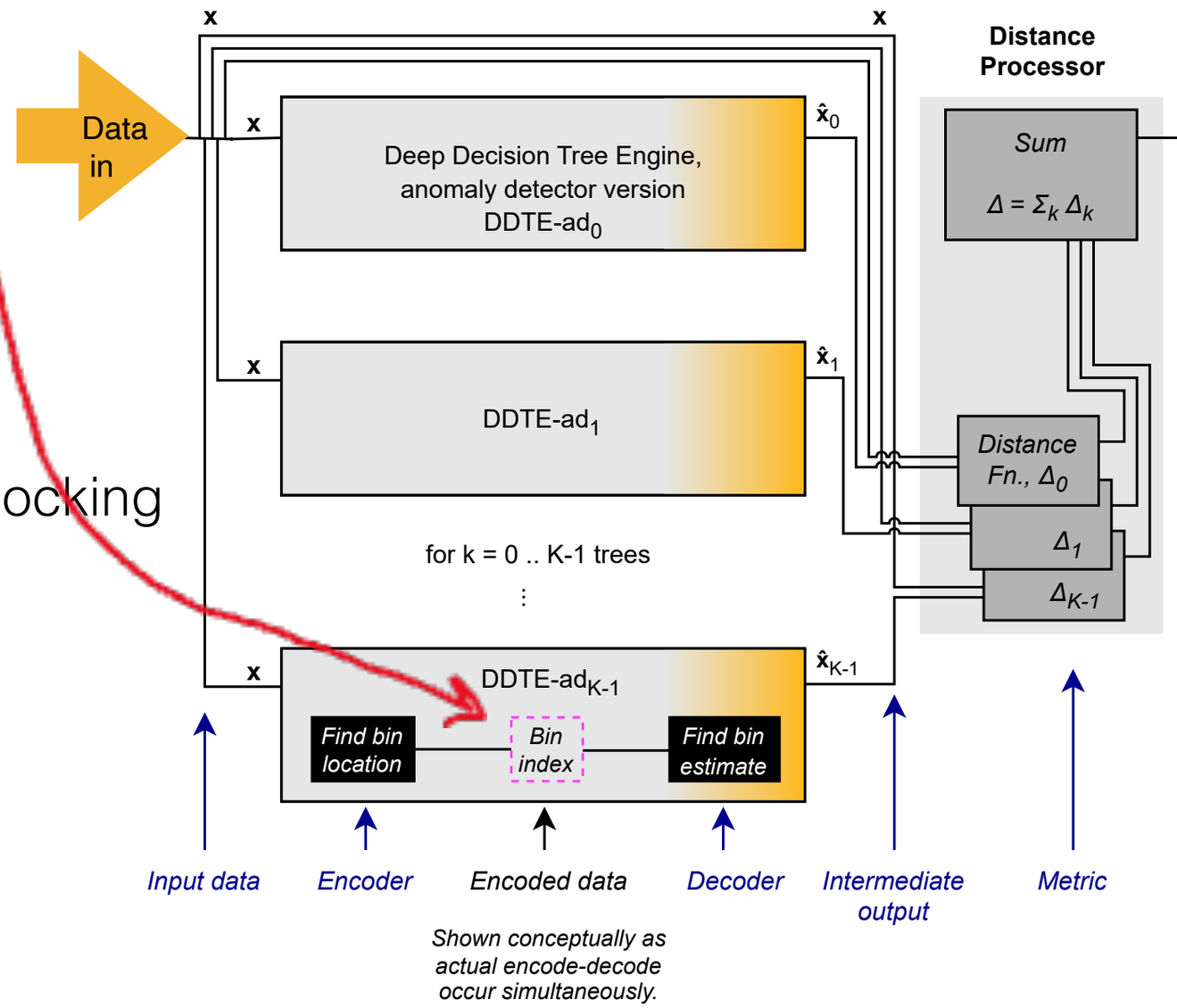
Incoming heart → Encode is Decode: return (5,4)

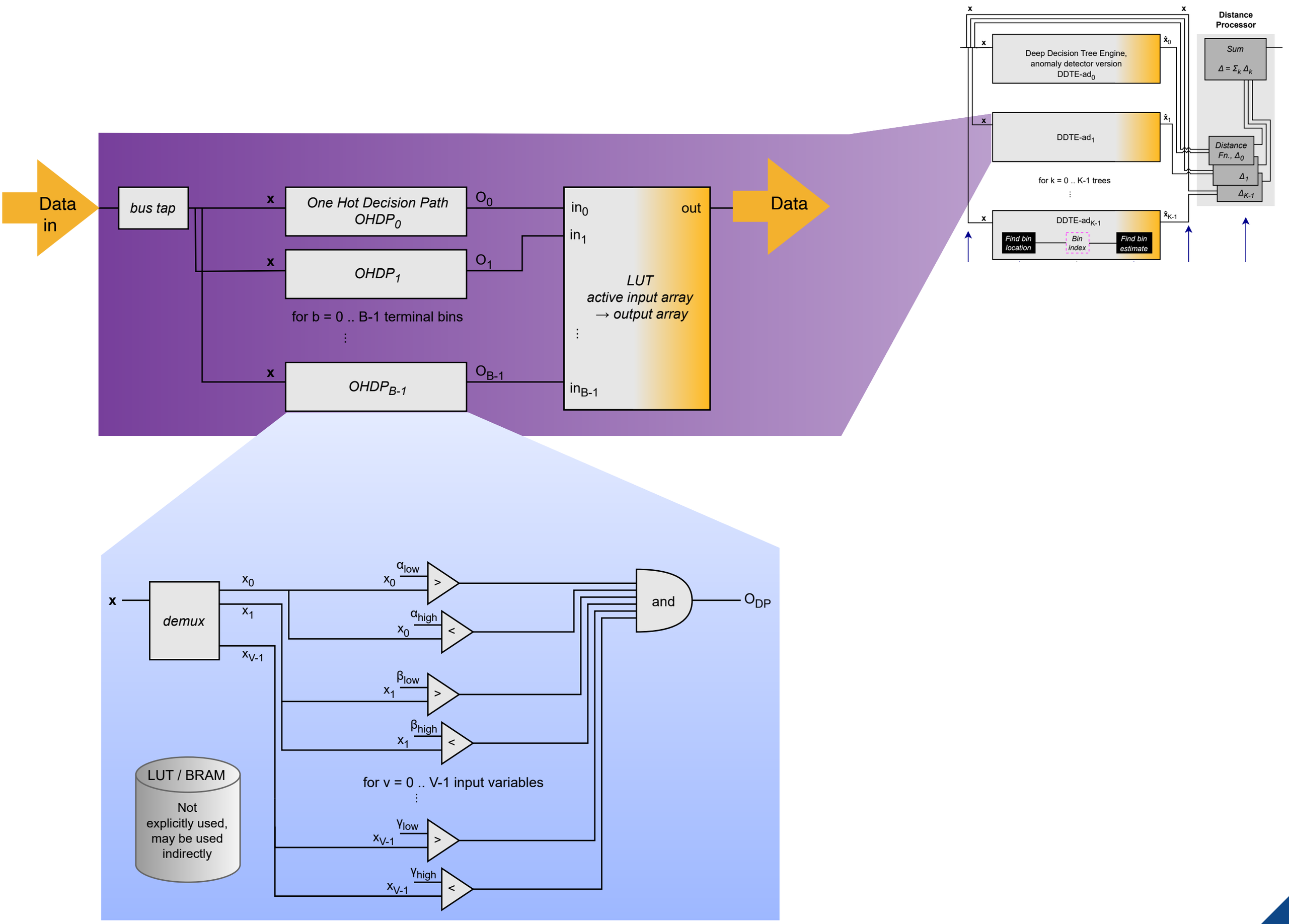
Logic flow

- Left-to-right data flow (see right)
- Realized that we can bypass the latent space!
  - Encoding = Decoding

Details

- Parallel computing
  - TREE ENGINES eval. in parallel
  - All combinatoric logic, so no clocking between steps = fast
  - Mostly comparisons = fast
  - No multiplication = fast
- Technical info in backup & [2304.03836]



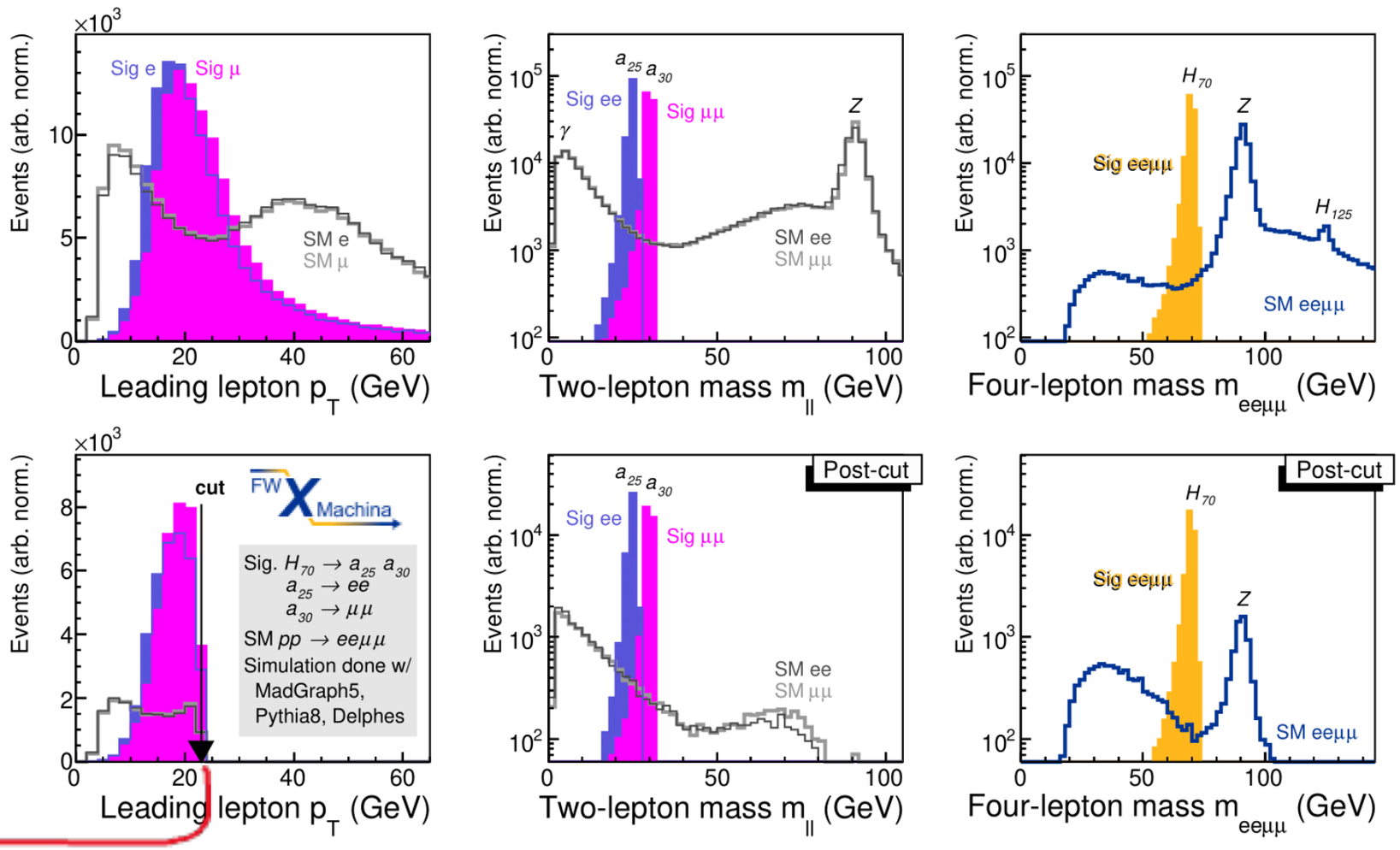

Data in
bus tap
x
One Hot Decision Path
OHDP0
O0
x
OHDP1
O1
for b = 0 .. B-1 terminal bins
x
OHDPB-1
OB-1
in0
in1
inB-1
out
LUT
active input array
→ output array
Data
x
x
Deep Decision Tree Engine,
anomaly detector version
DDTE-ad0
x̂0
x
DDTE-ad1
x̂1
for k = 0 .. K-1 trees
x
DDTE-adK-1
x̂K-1
Find bin location
Bin index
Find bin estimate
Distance Processor
Sum
Δ = Σk Δk
Distance Fn., Δ0
Δ1
ΔK-1
x
demux
x0
x1
xV-1
αlow
x0
>
αhigh
x0
<
βlow
x1
>
βhigh
x1
<
for v = 0 .. V-1 input variables
γlow
xV-1
>
γhigh
xV-1
<
and
ODP
LUT / BRAM
Not explicitly used, may be used indirectly

SM 2e 2μ vs. ?

TM Hong

Proof of concept problem

- Background: we generate all SM with 2e 2μ (predominantly ZZ*)
- Signal: ggF H→ a_1 a_2 → e^+ e^- μ^+ μ^- (different m_H & m_a)



Veto events with lepton p_T > 23 GeV

- Consider only events that won't be already captured by L1 trigger

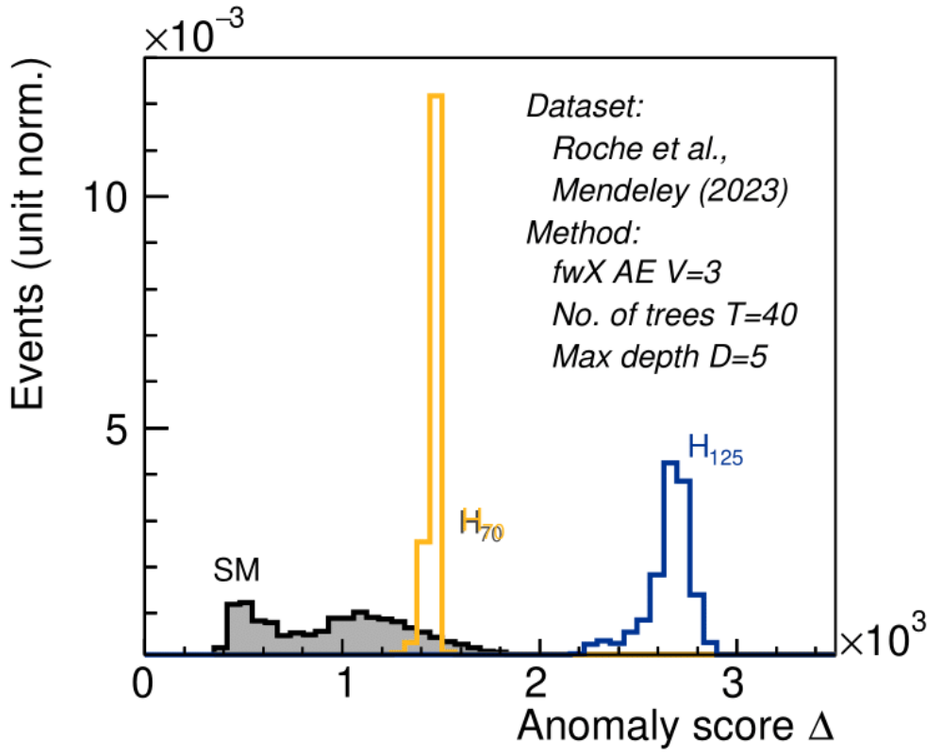
SM 2e 2μ vs. ?

TM Hong

Proof of concept problem

- Design
  - 40 decision trees with maximum depth of 5
  - 3 variables: m_{ee}, $m_{\mu\mu}$, m_{4l}
- Physics results (see figure)
  - Great separation for H_{125}
  - May need a “window selection” for H_{70}
- FPGA results (see table)
  - Latency within 25 ns = 1 BC
  - Percent-level (or smaller) resource usage
  - No multiplications!

Distribution



Parameter	Value
Clock speed	320 MHz
Latency	8 ticks (25 ns)
Interval	1 tick (3.125 ns)
FF	10k (0.4 %)
LUT	31k (2.6%)
DSP	3 (0.04%)
BRAM	0

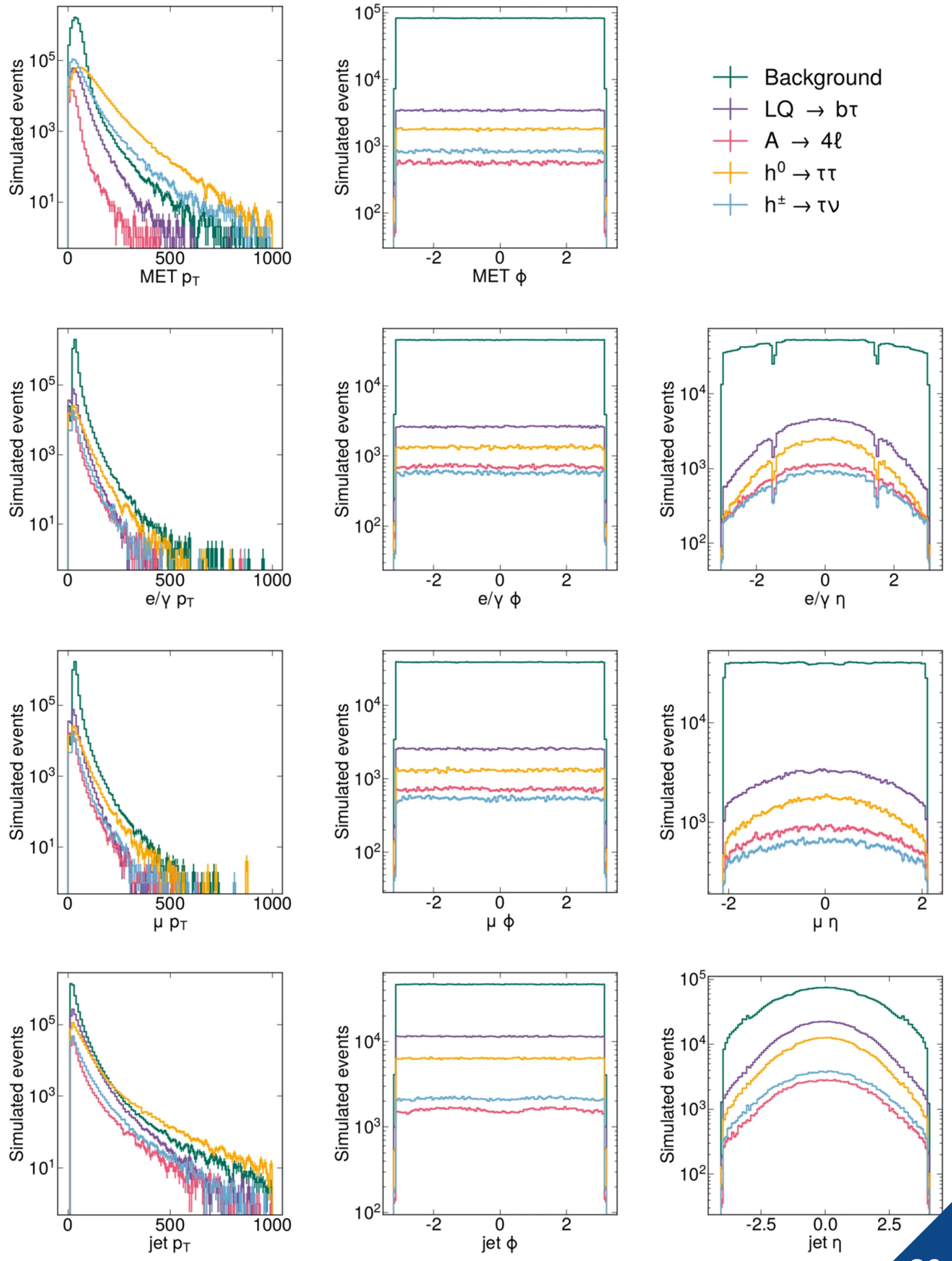
Compare with hls4ml

TM Hong

LHC anomaly detection ds

[Sci Data 9, 118]

- Background
  - W → lν, Z → ll, multijet, ttbar
- Signal
  - 4 BSM scenarios
- Input variables
  - 54 variables
  - p_T, η, ϕ of the 4 leading μ, 4 leading e, 10 leading jets, MET
  - See distributions on the right
- Sample selection
  - Require ≥1 lepton w/ p_T > 23 GeV
  - (L1 will already save these...)



Cross-check with public data

TM Hong

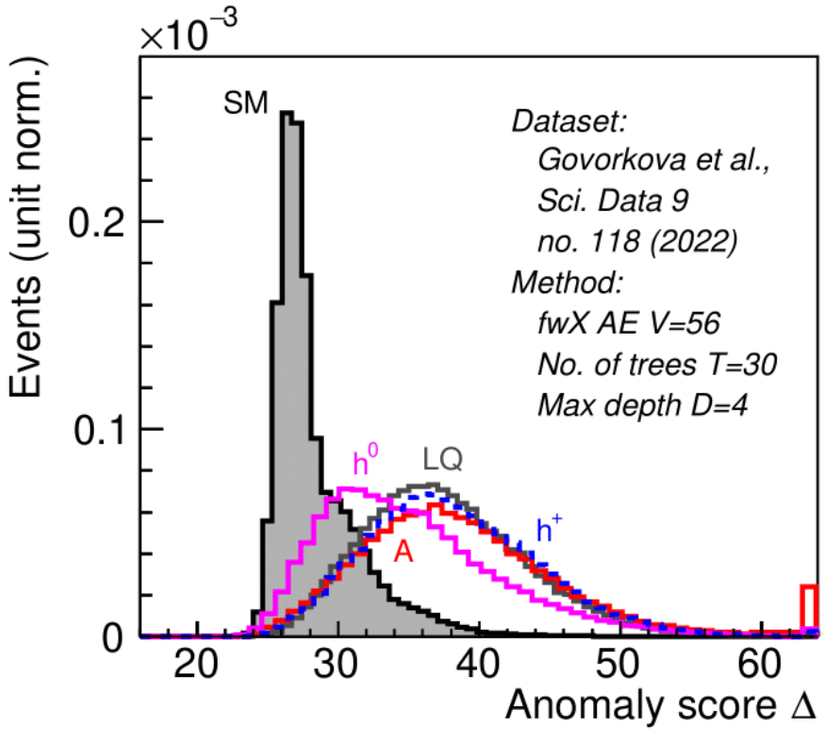
Works well

- Physics (plots)
- FPGA (table)

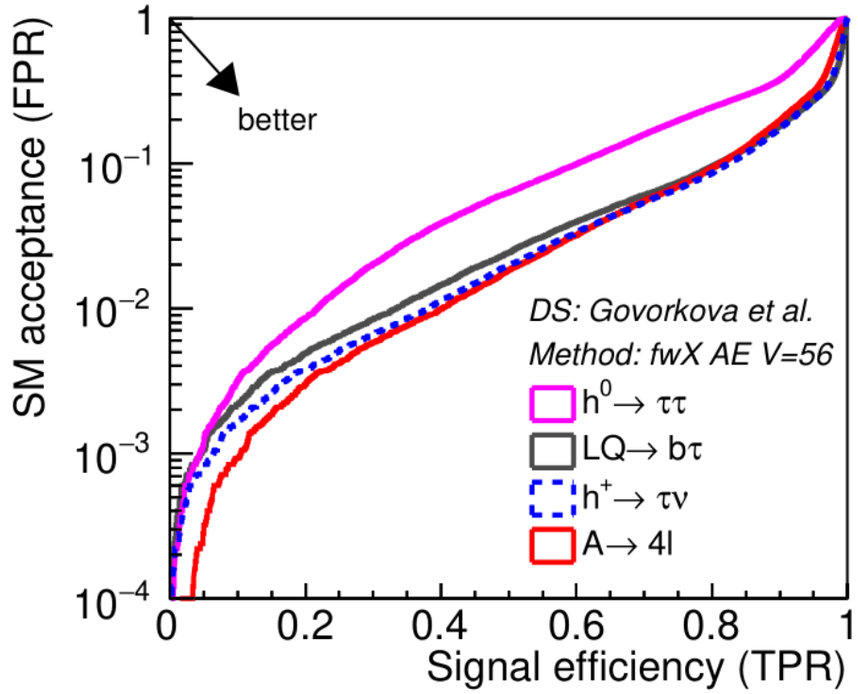
Comparison

- Hls4ml NN-AE
  [Nature Mach. Intell. 4 (2022) 154–161]
- Physics: comparable AUC
- FPGA results →

Distribution



ROC curve



	hls4ml	fwX (this)
Clock speed	200 MHz	200 MHz
Latency	80 ns	30 ns
Interval	5 ns	5 ns
FF	0.5%	0.6%
LUT	3%	9%
DSP	1%	0.8%
BRAM	0.3%	0

What I presented

Decision tree-based autoencoder

- New training method by sampling, it's density estimation
- More transparent (to me) than neural network-based designs
- Can do problems in high energy physics (3 - 50 variables)
- Competitive performance vs. hls4ml

Efficient implementation

- Latent space-less design where encoding = decoding
- Performance on Xilinx Virtex Ultrascale+ VU9P
  - O(1)% level resource usage
  - Fast at 30 ns latency
  - Try it yourself with the provided testbench & IP available online

What I think about

TM Hong

Then what

- What are we going to do with the events that we save?
  - Everyone is saving rare events that are uncategorized. Who's going to categorize them? CMS recently showed an event display of the most anomalous event. Will we go through one-by-one to try to guess at the physics?
  - There are ideas, but more needed

What about benchmarks?

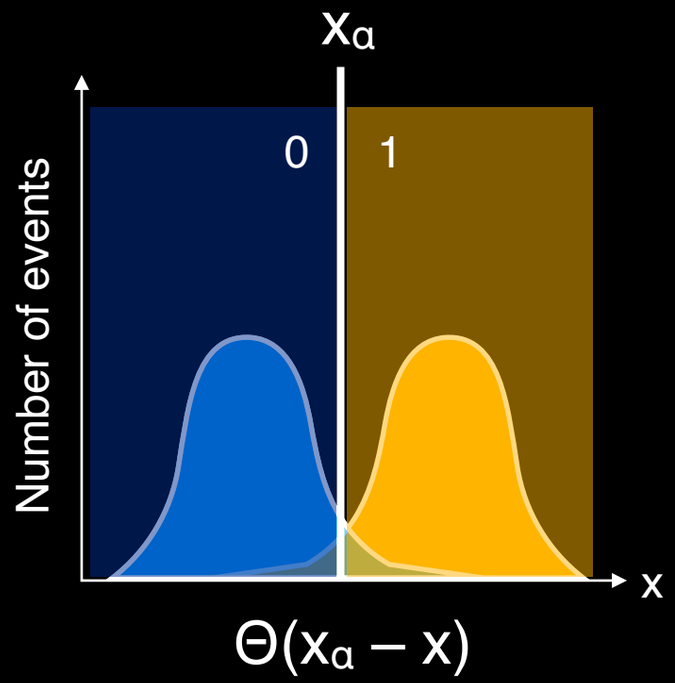
- By construction, it's supposed to pick up events that we don't know about. But to benchmark it, we choose models that we know about. Is this a contradiction? How do we avoid it? Who gets to choose?
- How much trigger bandwidth do we devote to it if we don't know what may be in it?

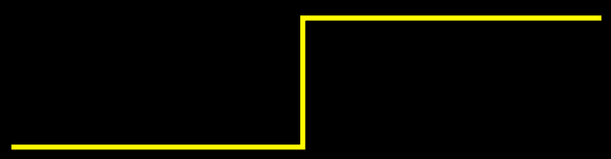
Backup slides

Neural networks basics

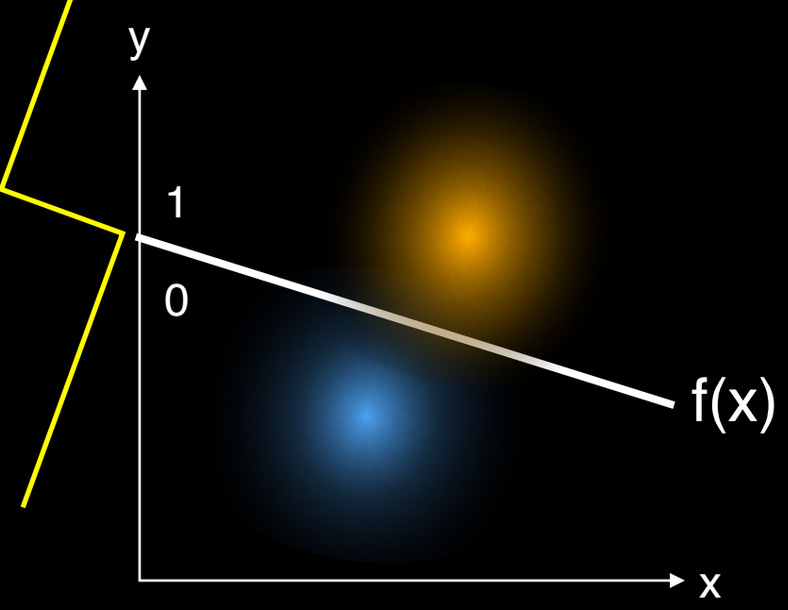
From Bruce Denby, *Tutorial on Neural Network Applications in High Energy Physics: A 1992 Perspective*, FERMILAB-CONF-92 / 121-E

Step function for 1d

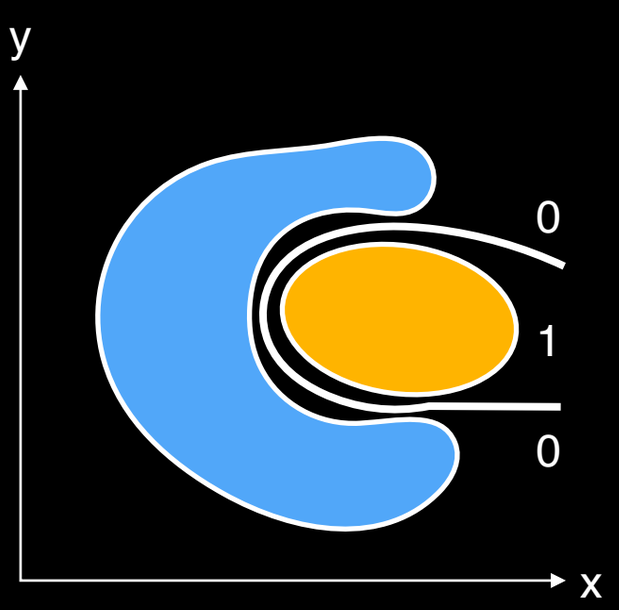




Step function for 2d



Curved step fn? for 2d



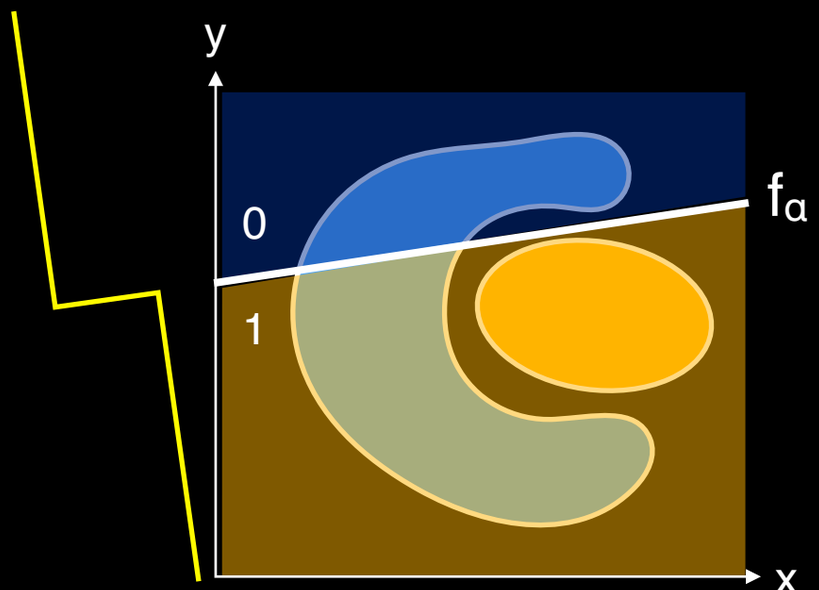
$= \Theta(y - (mx + b))$ substitute

$= \Theta(c_1 y + c_2 x + b)$ multiply by c_1 & define c_2

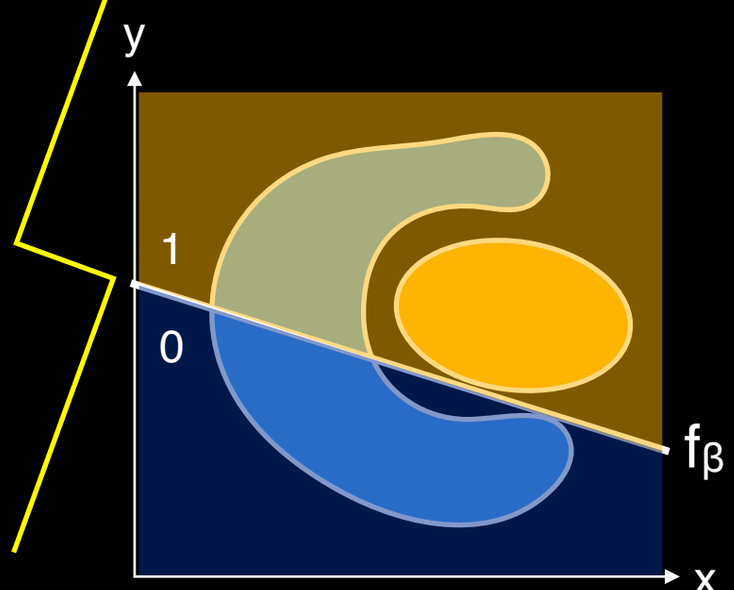
$= \Theta(c_1 x_1 + c_2 x_2 + b)$ generalized notation

$= \Theta(\mathbf{c} \cdot \mathbf{x} + b)$ vector notation

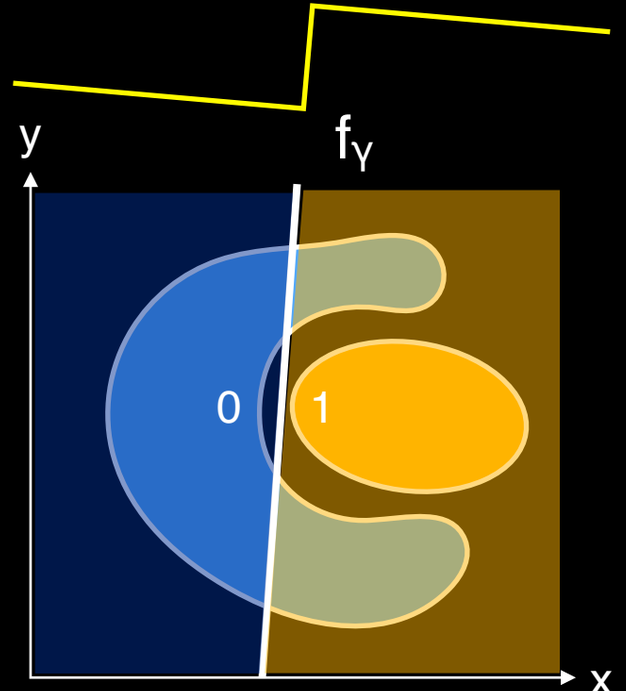
Step functions divide samples given a desired true / false positive rates



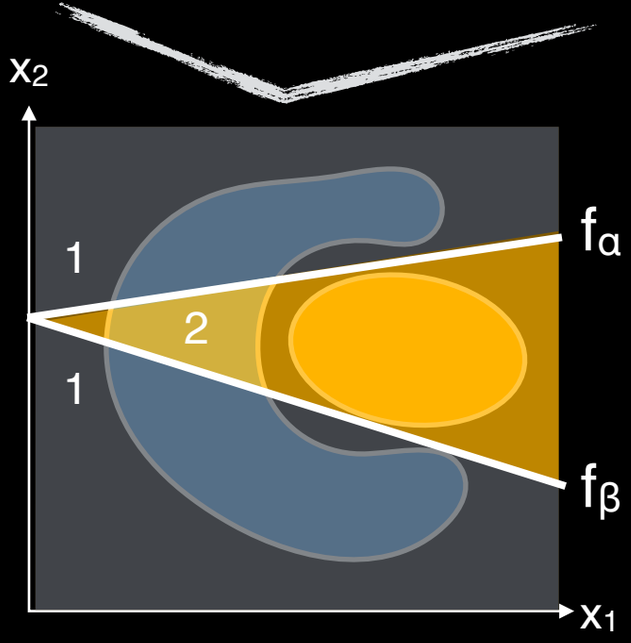
$\Theta(c_{\alpha 1}\, x_1 + c_{\alpha 2}\, x_2 + b_\alpha)$



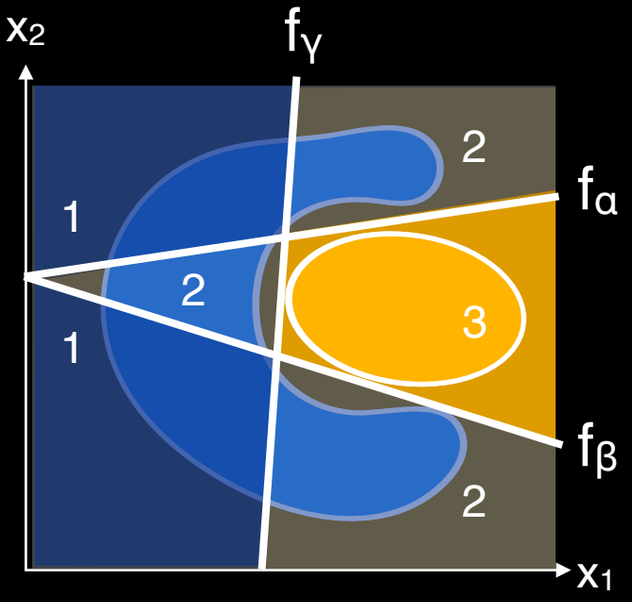
$\Theta(c_{\beta 1}\, x_1 + c_{\beta 2}\, x_2 + b_\beta)$



$\Theta(c_{\gamma 1}\, x_1 + c_{\gamma 2}\, x_2 + b_\gamma)$

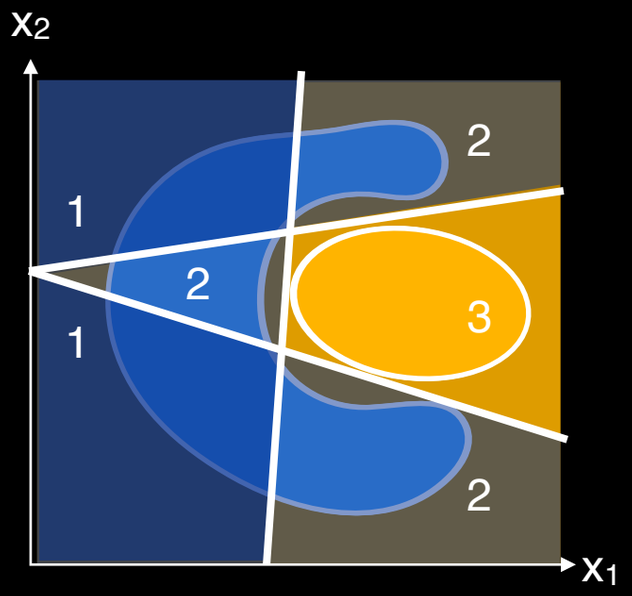


$\Theta(c_{\alpha 1}\, x_1 + c_{\alpha 2}\, x_2 + b_\alpha) +$
$\Theta(c_{\beta 1}\, x_1 + c_{\beta 2}\, x_2 + b_\beta)$

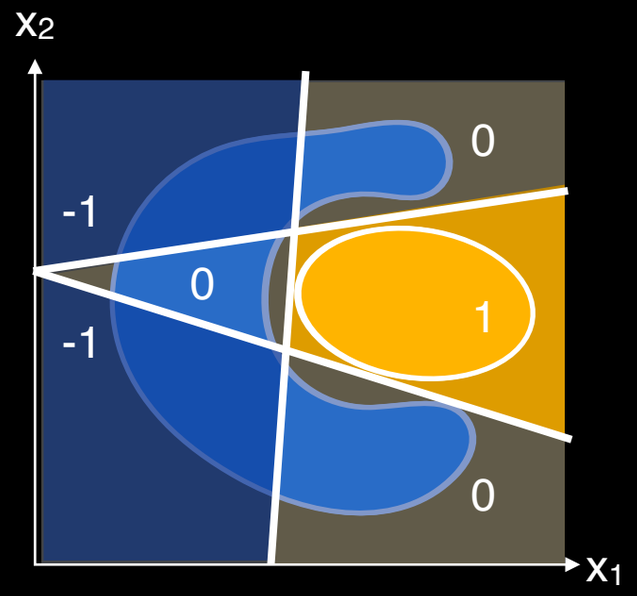


$\Theta(c_{\alpha 1}\, x_1 + c_{\alpha 2}\, x_2 + b_\alpha) +$
$\Theta(c_{\beta 1}\, x_1 + c_{\beta 2}\, x_2 + b_\beta) +$
$\Theta(c_{\gamma 1}\, x_1 + c_{\gamma 2}\, x_2 + b_\gamma)$

Sum of step functions can approximate the desired contour

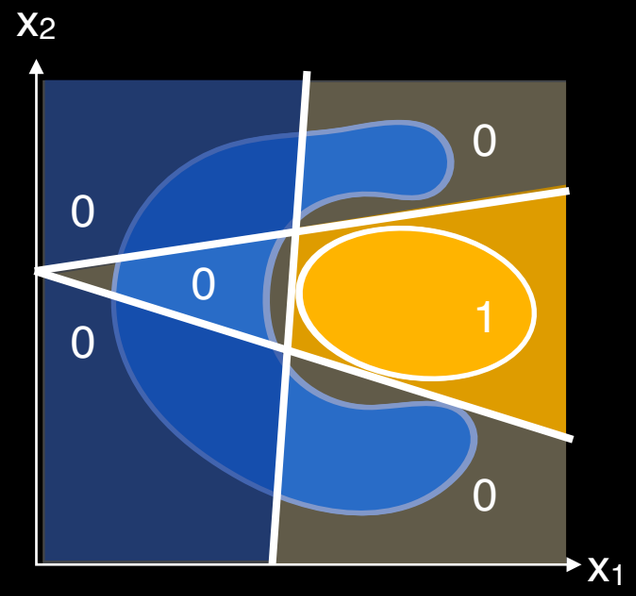


$$\Theta(\mathbf{c}_\alpha \cdot \mathbf{x} + b_\alpha) + \Theta(\mathbf{c}_\beta \cdot \mathbf{x} + b_\beta) + \Theta(\mathbf{c}_\gamma \cdot \mathbf{x} + b_\gamma)$$



$$\Theta(\mathbf{c}_\alpha \cdot \mathbf{x} + b_\alpha) + \Theta(\mathbf{c}_\beta \cdot \mathbf{x} + b_\beta) + \Theta(\mathbf{c}_\gamma \cdot \mathbf{x} + b_\gamma) - 2$$

subtract 2



$$\Theta(\,\Theta(\mathbf{c}_\alpha \cdot \mathbf{x} + b_\alpha) + \Theta(\mathbf{c}_\beta \cdot \mathbf{x} + b_\beta) + \Theta(\mathbf{c}_\gamma \cdot \mathbf{x} + b_\gamma) - 2)$$

threshold

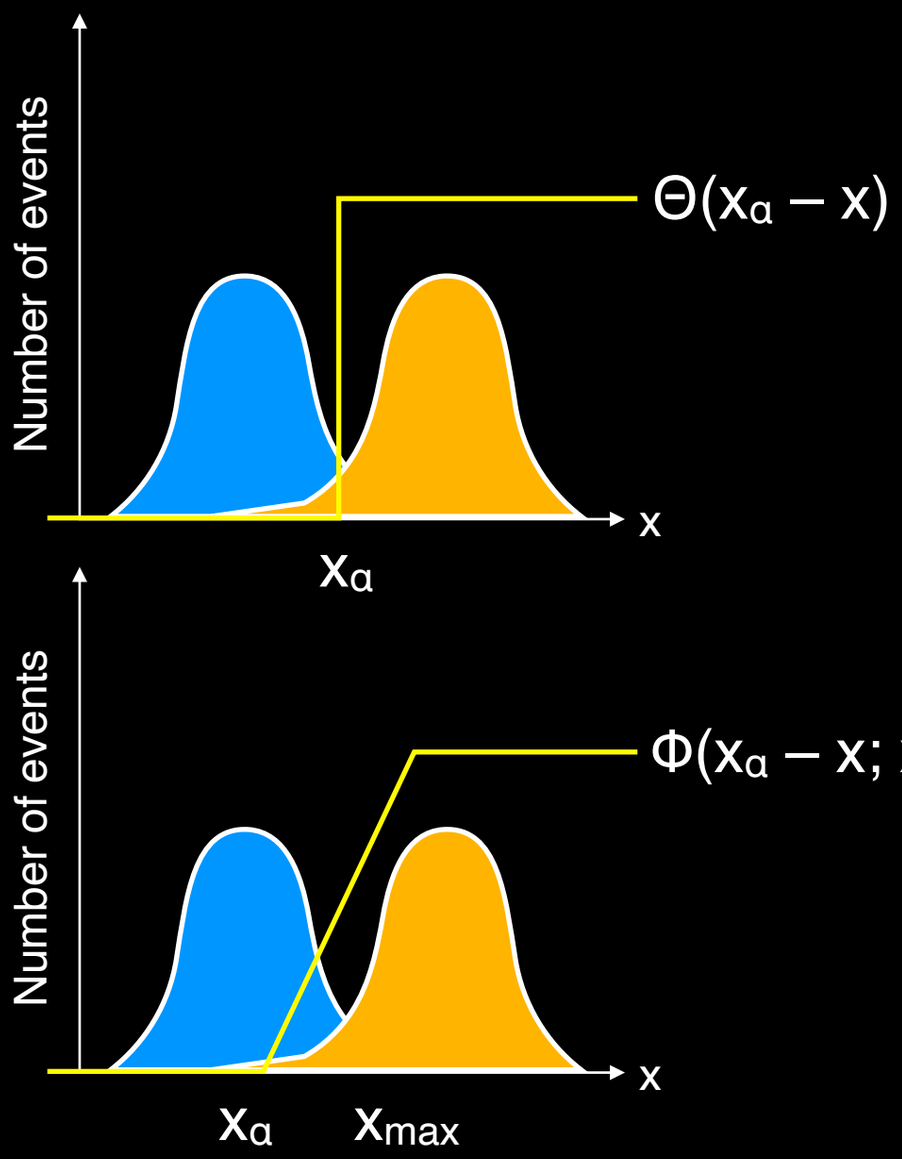
Step function for 2-dim inputs

The contour is converted to the final step function

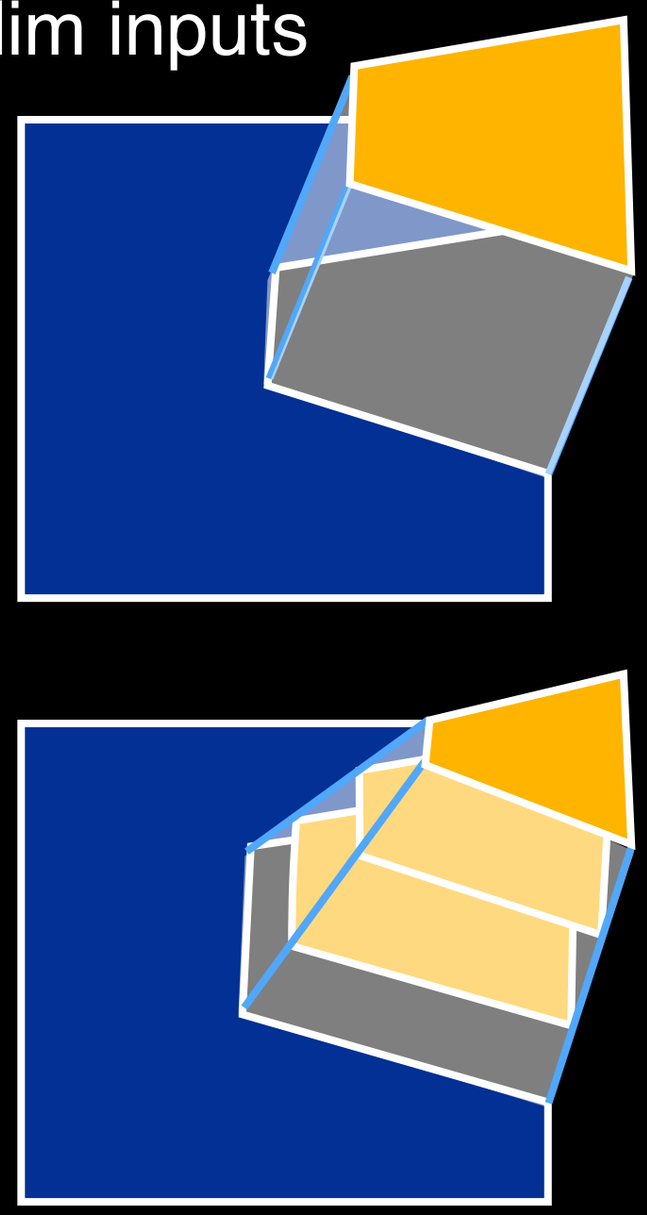
Activation function

Fuzzy boundary using a function

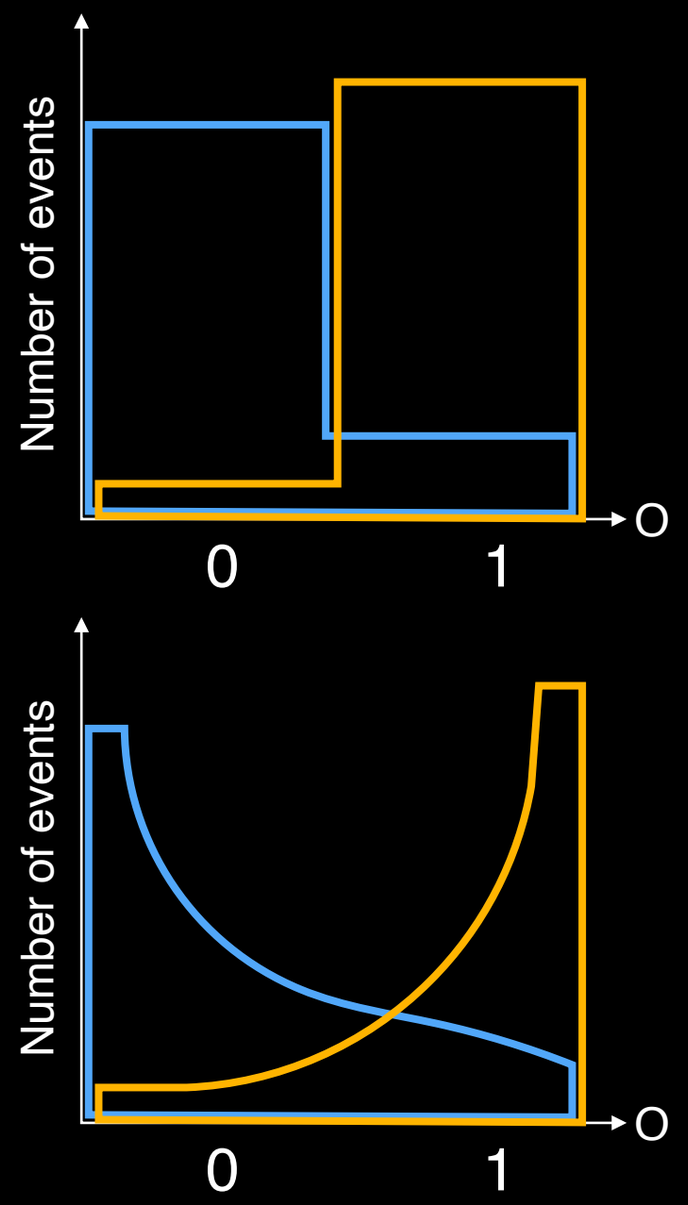
1-dim input



2-dim inputs



Output score

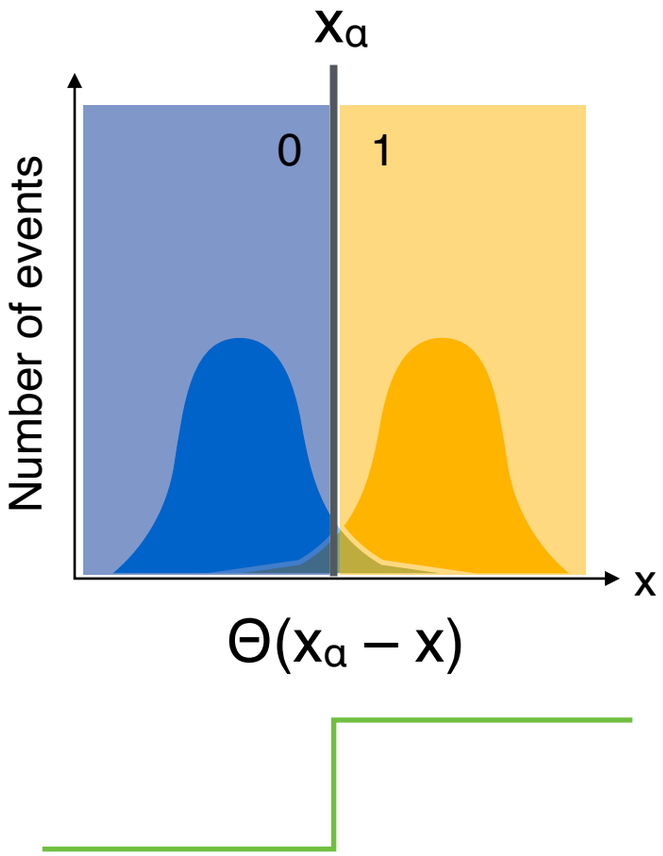


Activation fn gives users a handle to control true / false positive rates

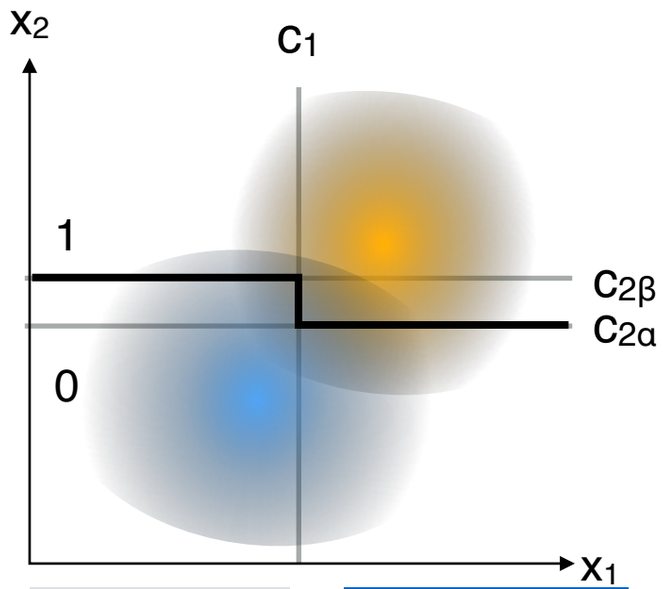
Decision tree basics

And how it achieves the same result as NN

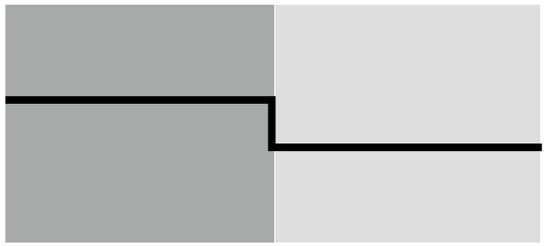
Step function for 1d



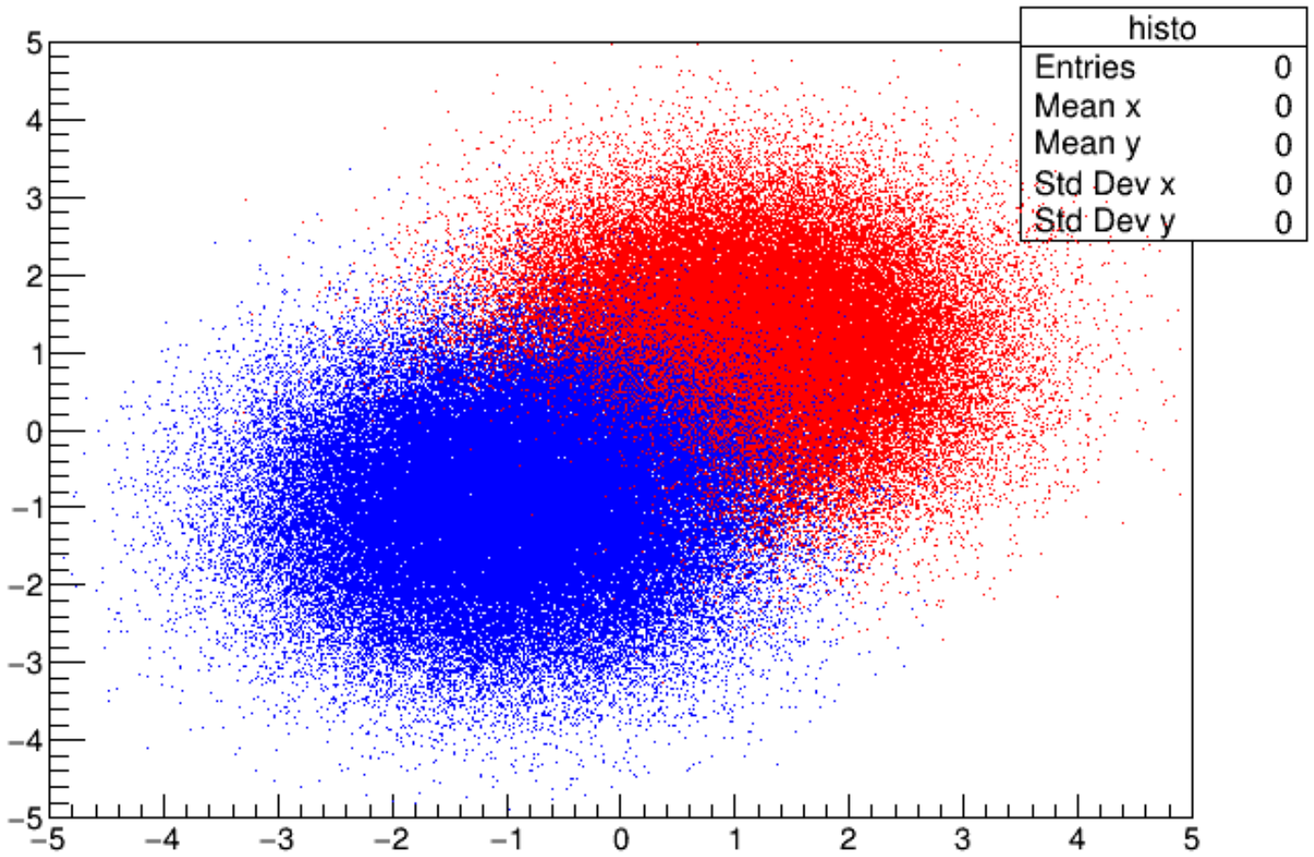
Step function for 2d



$$\Theta(x_1 - c_1) \cdot \Theta(x_2 - c_{2\alpha}) +$$
$$\Theta(c_1 - x_1) \cdot \Theta(x_2 - c_{2\beta})$$

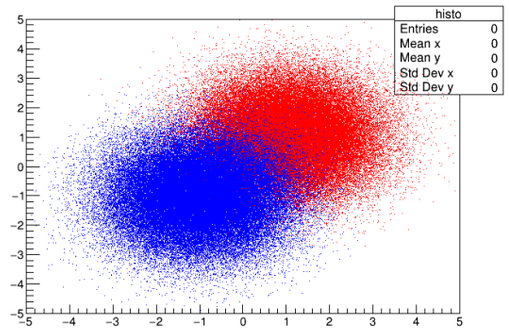


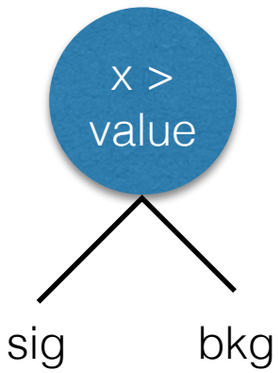
Flip book

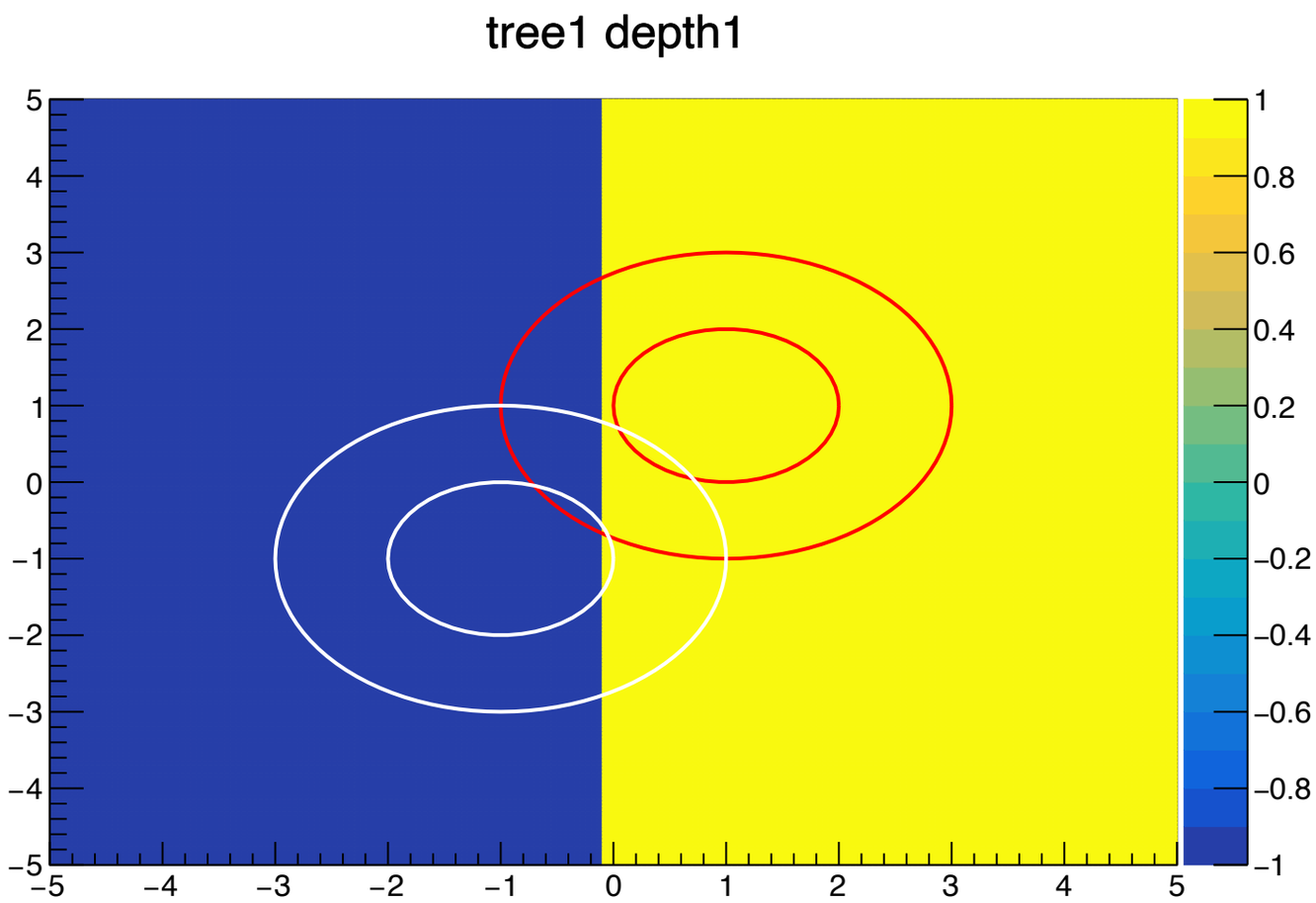


Unit gaussians of two variables

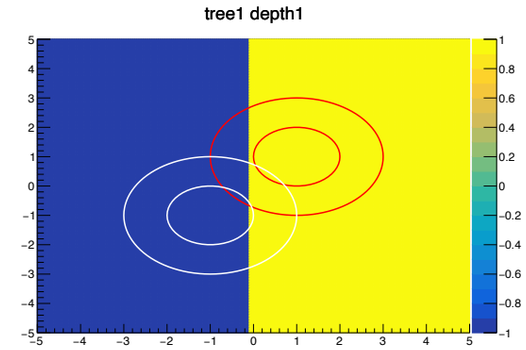
One decision tree

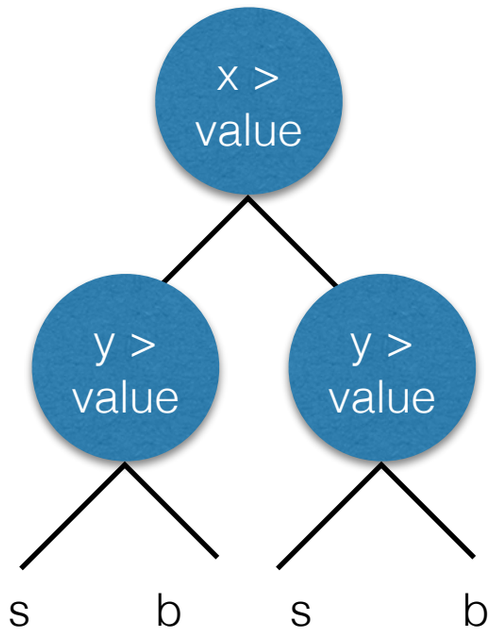


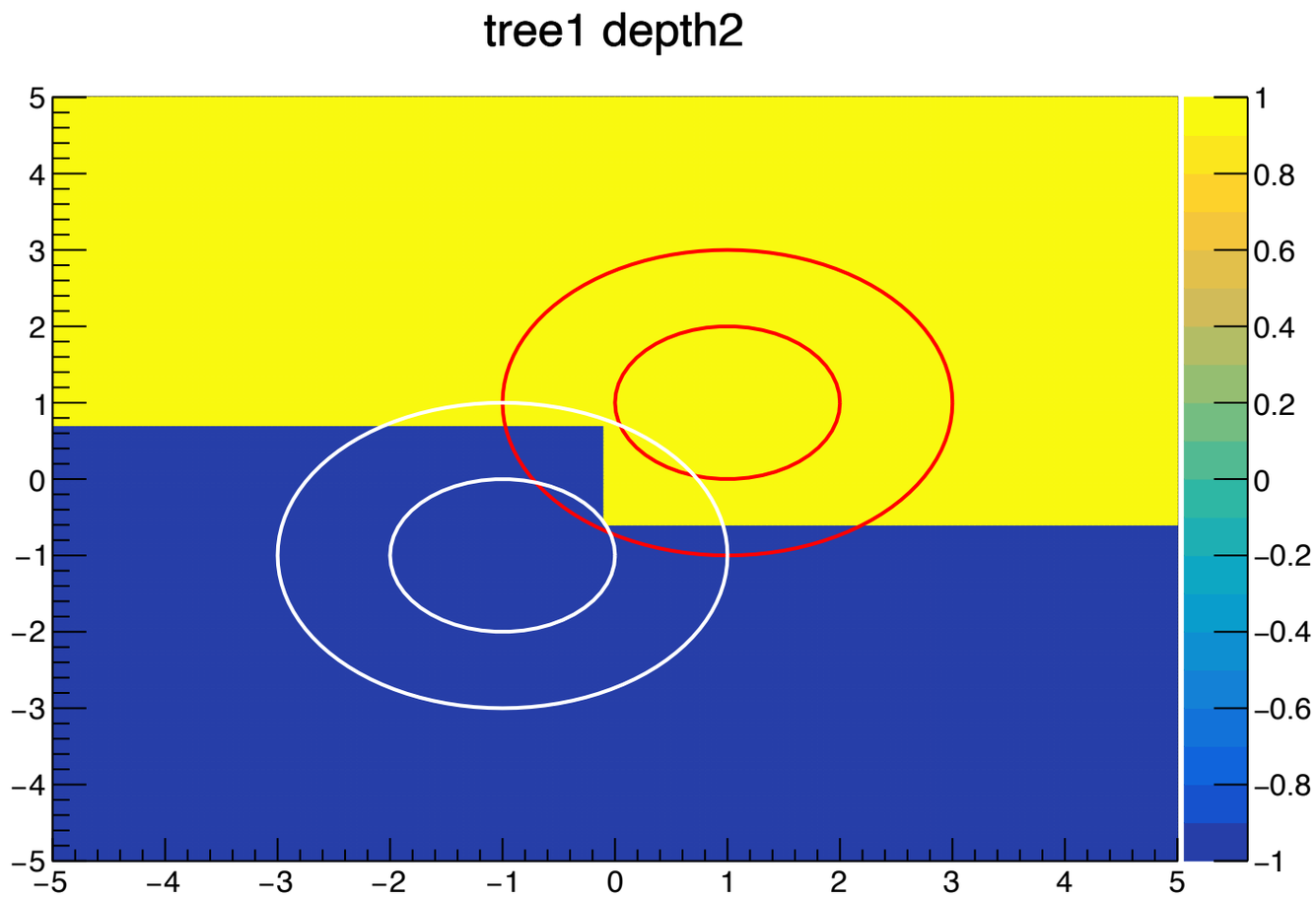




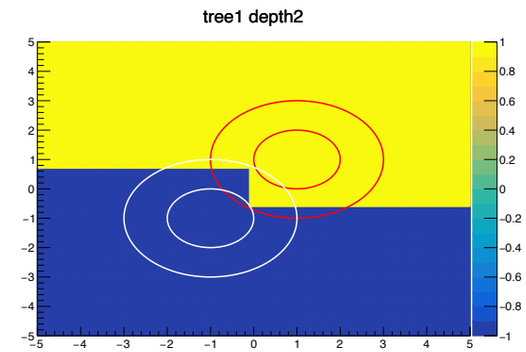
One decision tree

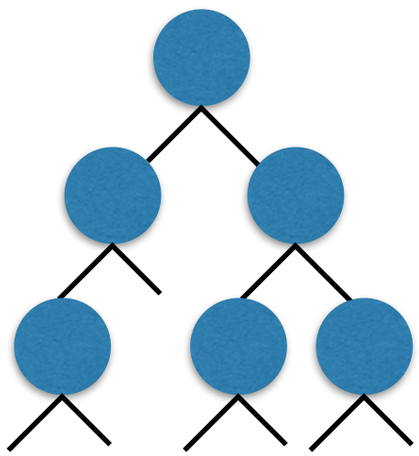


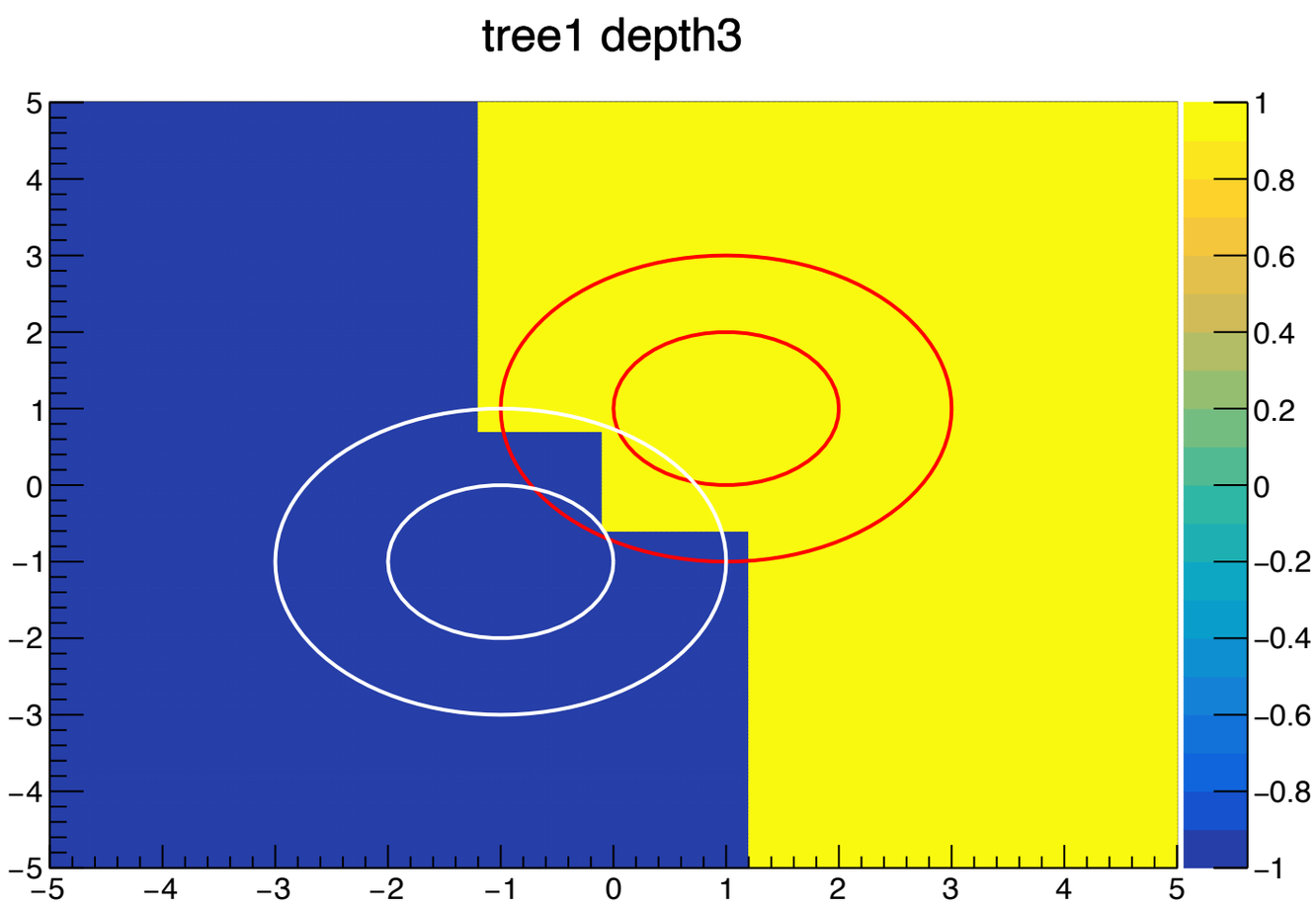




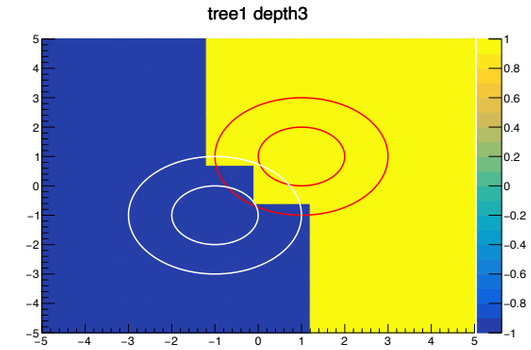
One decision tree

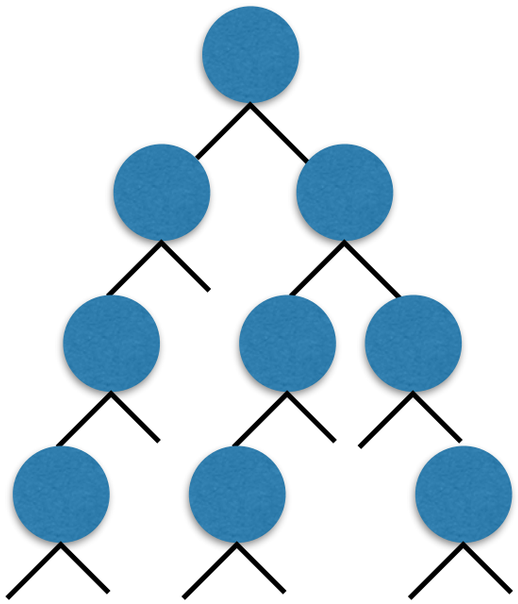


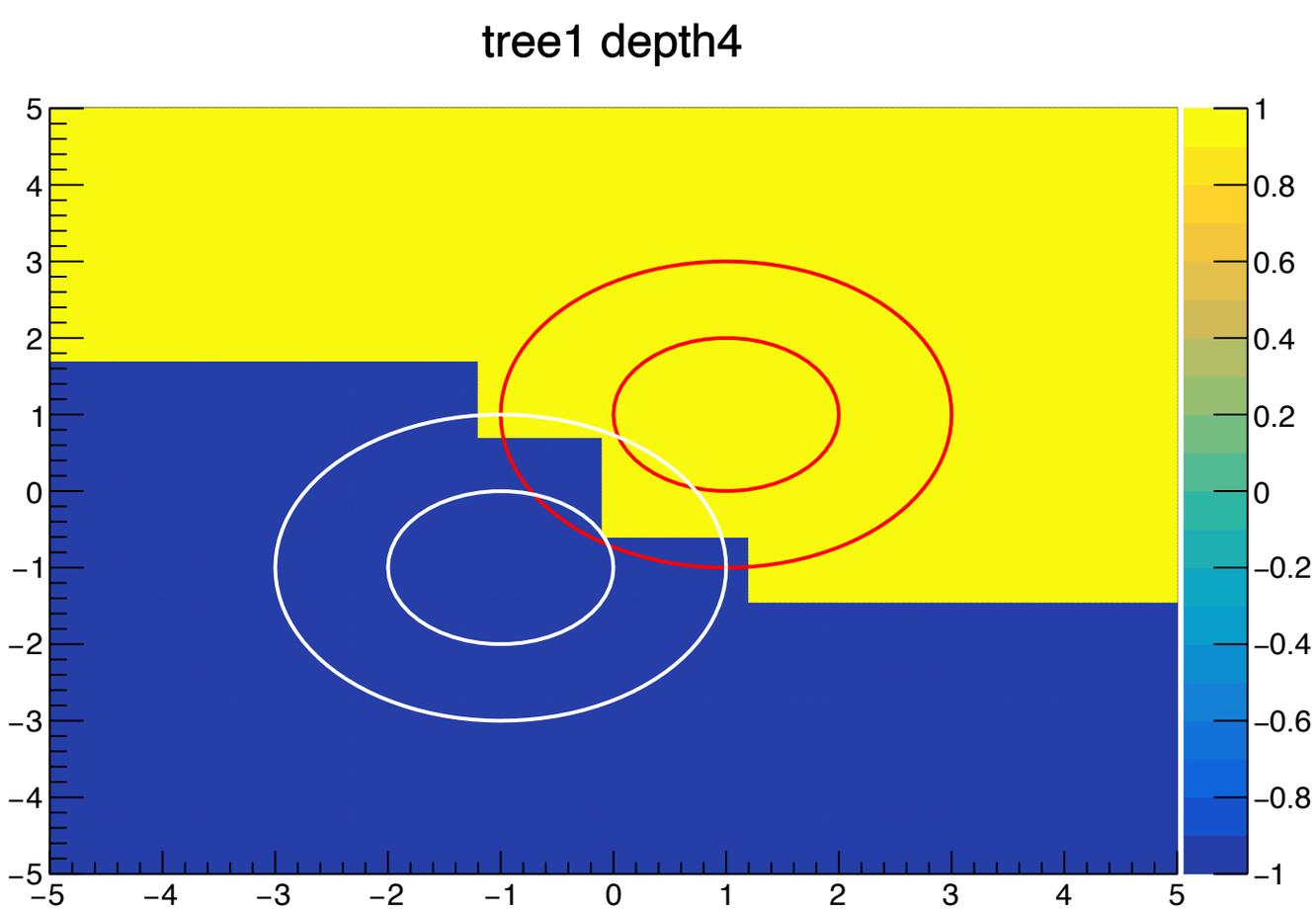




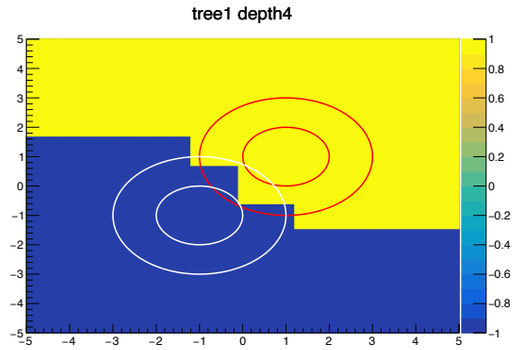
One decision tree

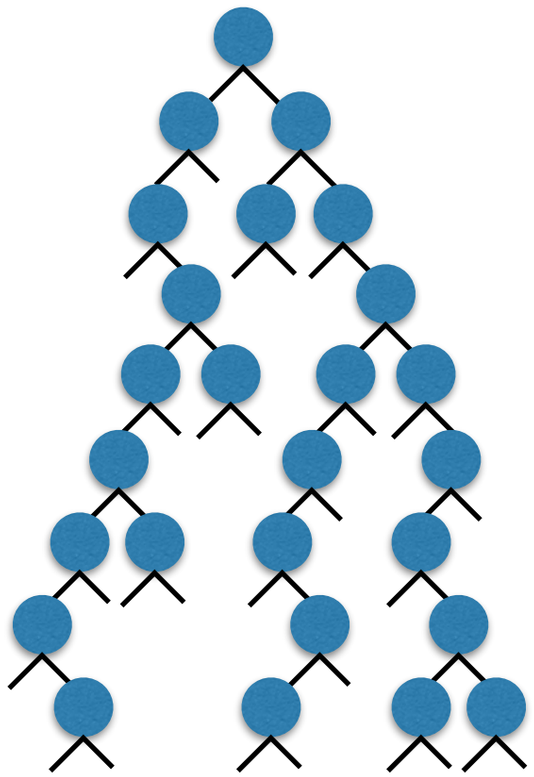


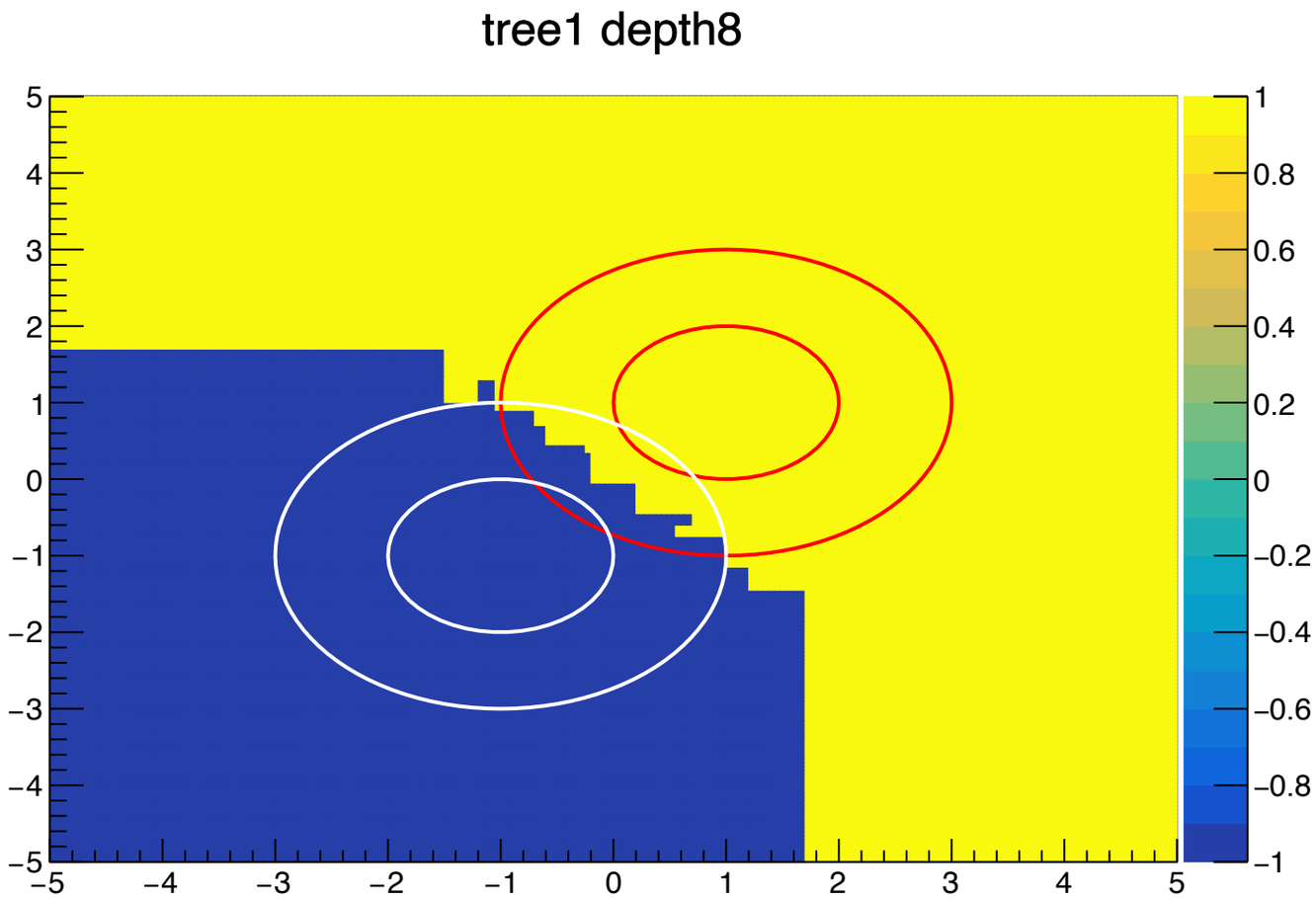




One decision tree





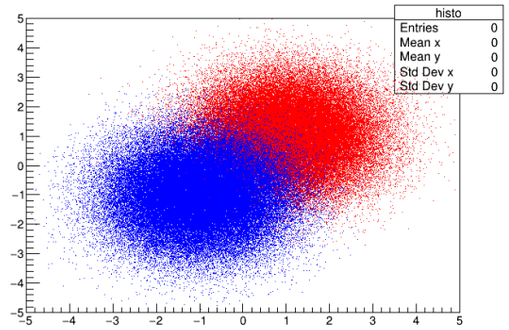


Draws diagonal

Depth 2

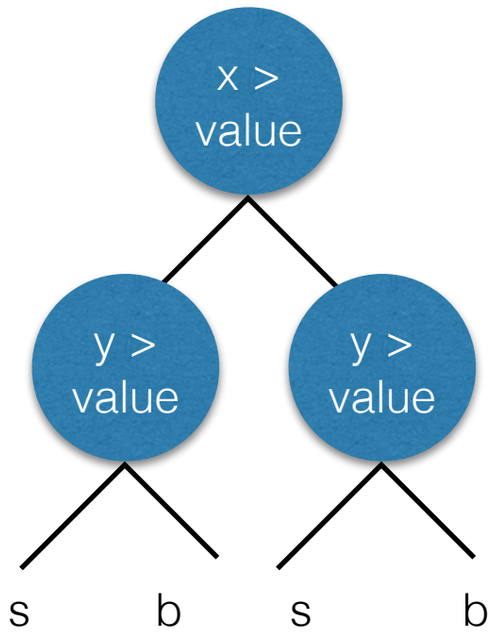
vary trees

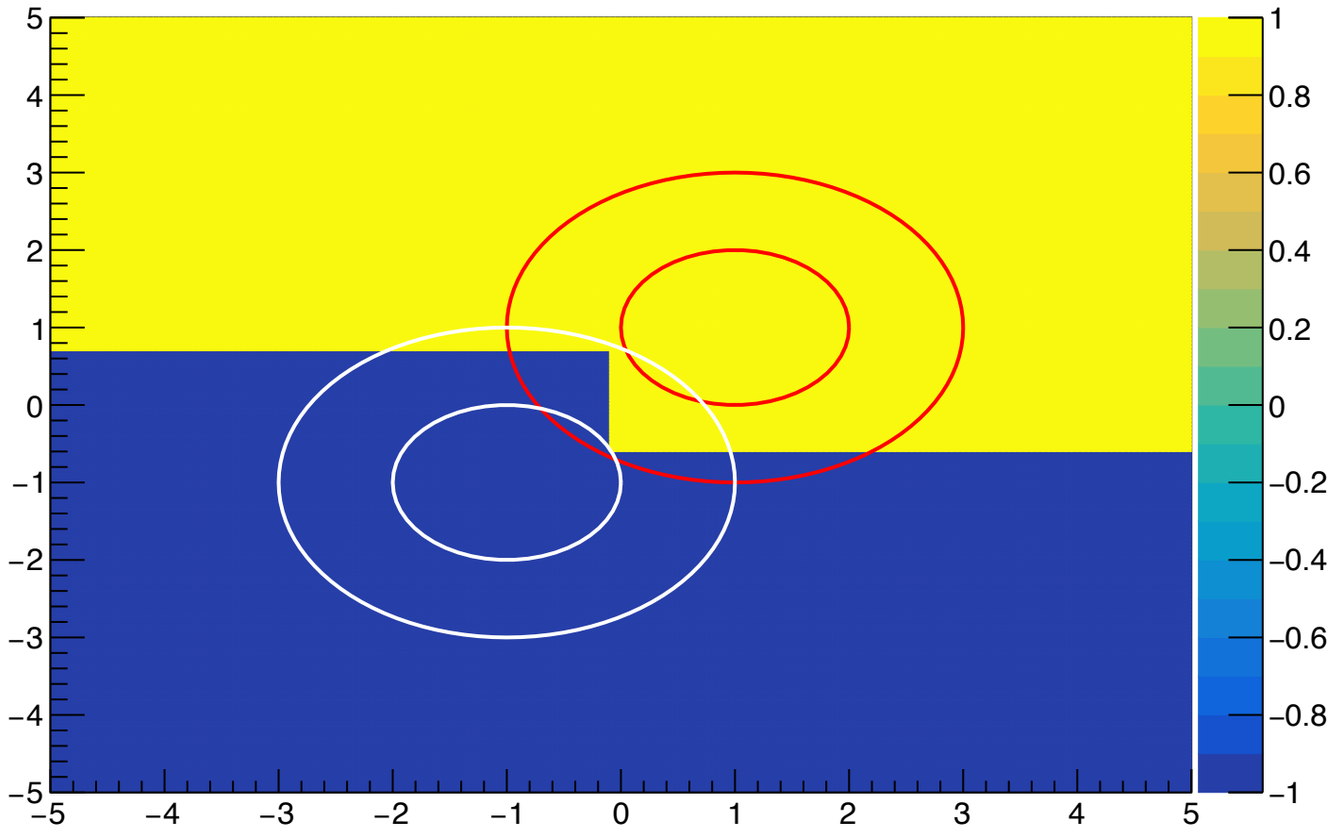
Depth 2



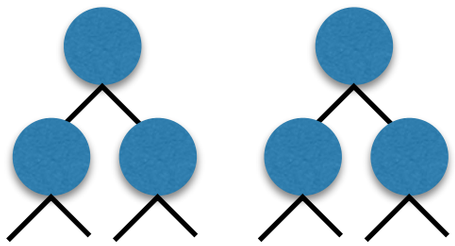
vary

tree1 depth2



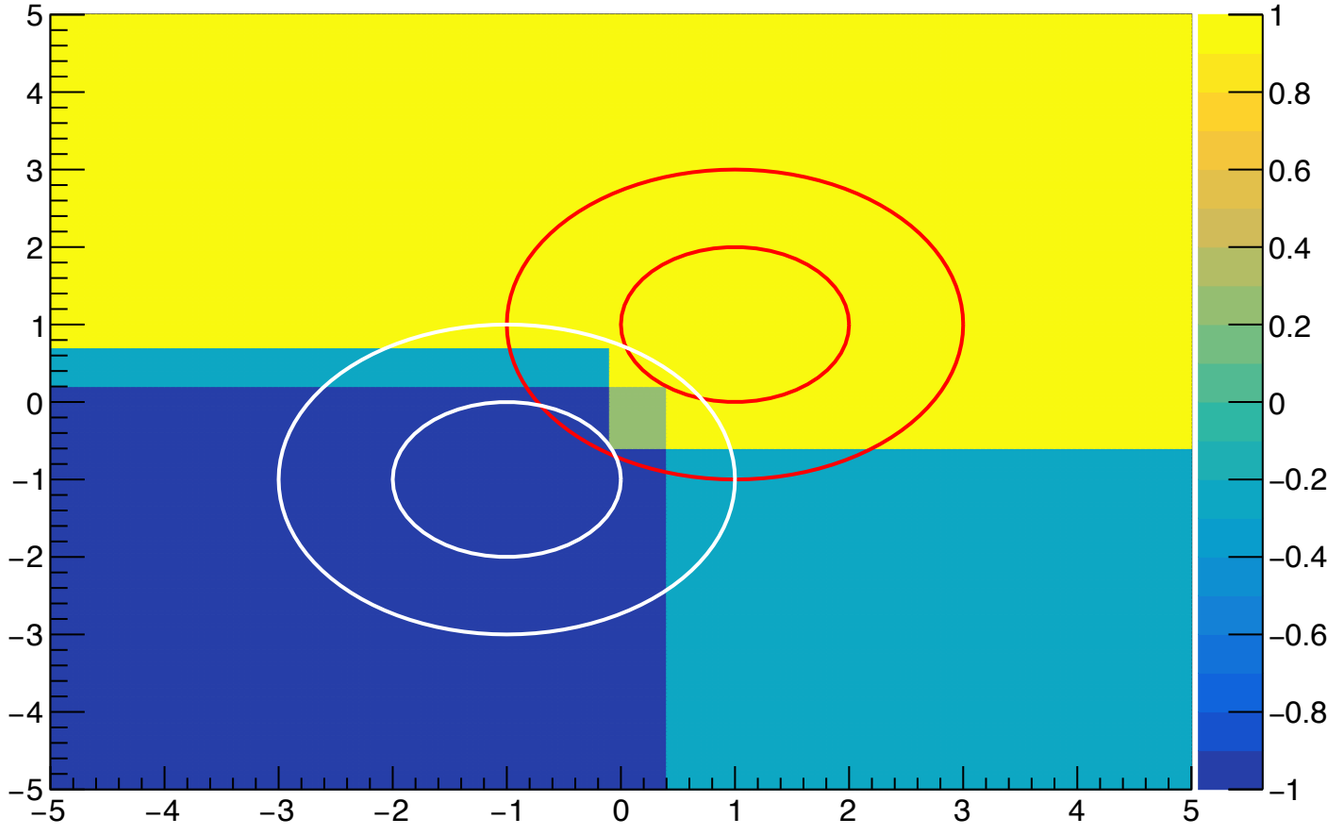


Depth 2

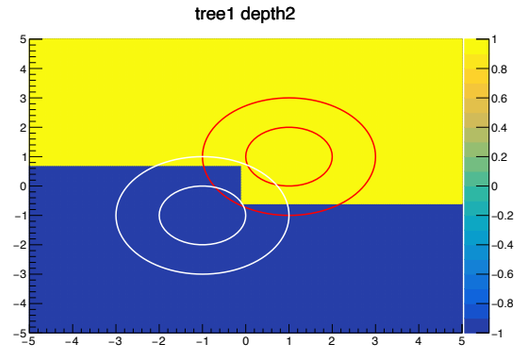


vary

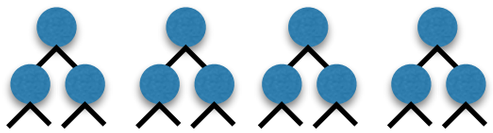
tree2 depth2



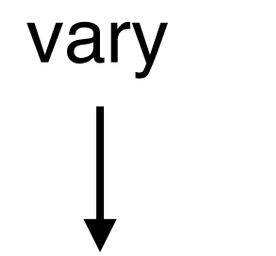
tree1 depth2



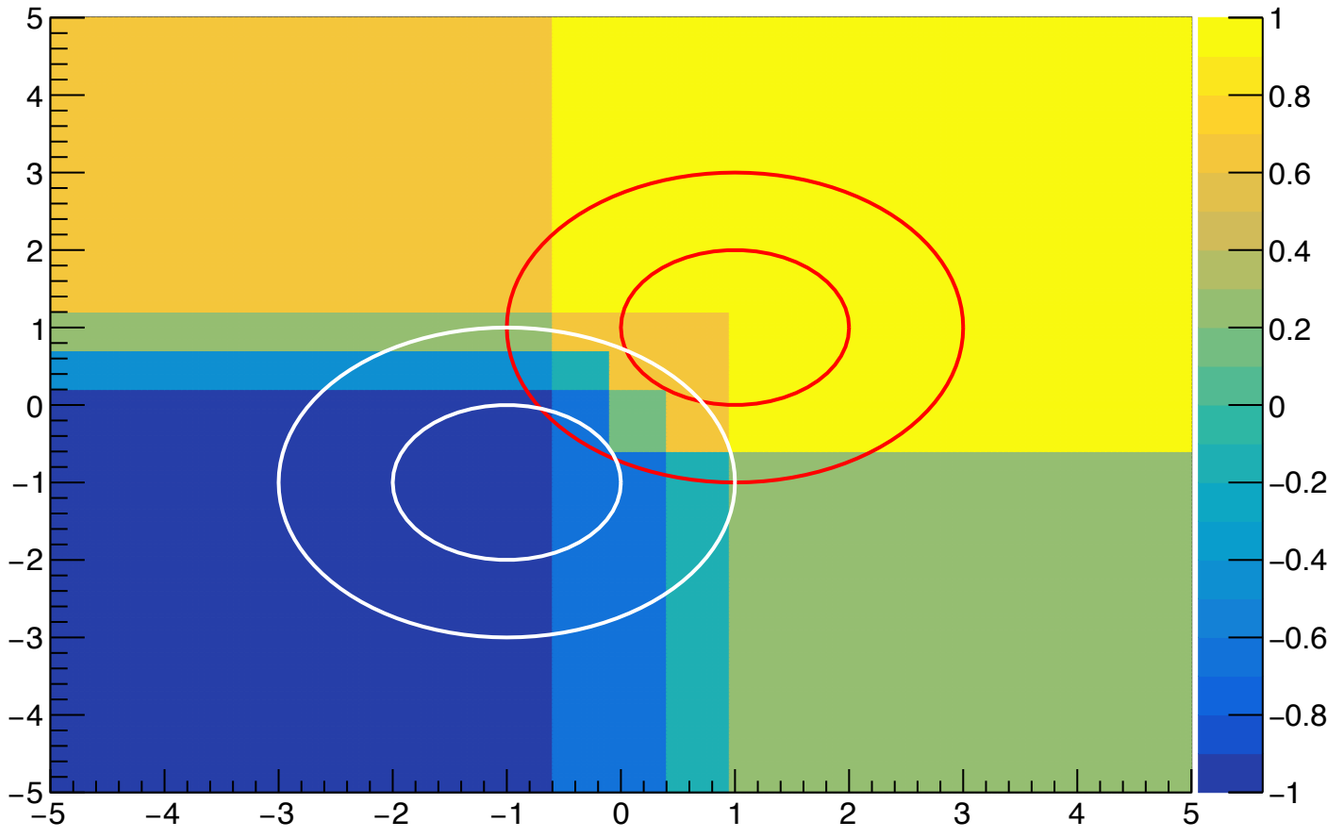
Depth 2

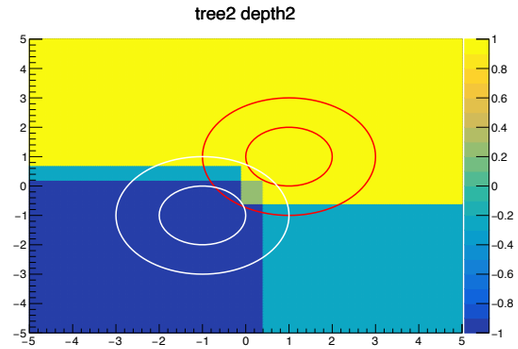


vary

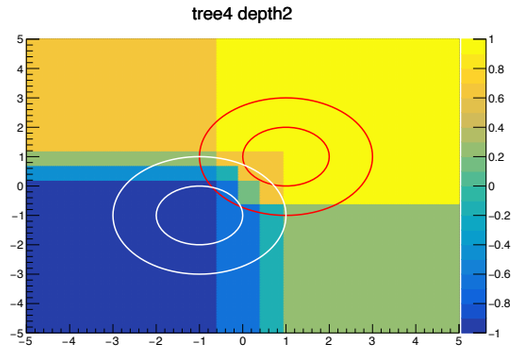


tree4 depth2

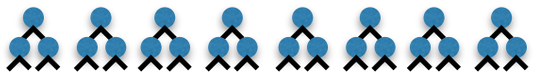




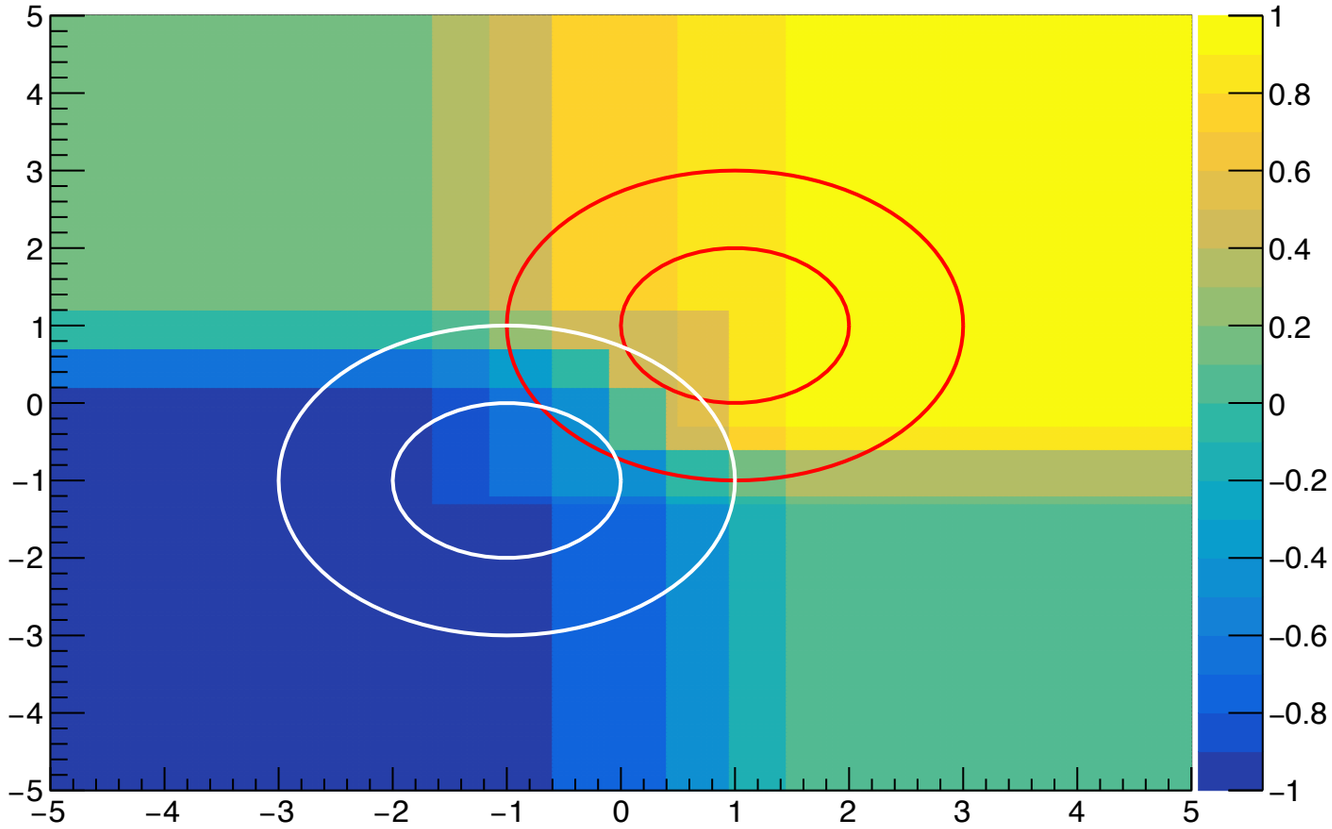
Depth 2



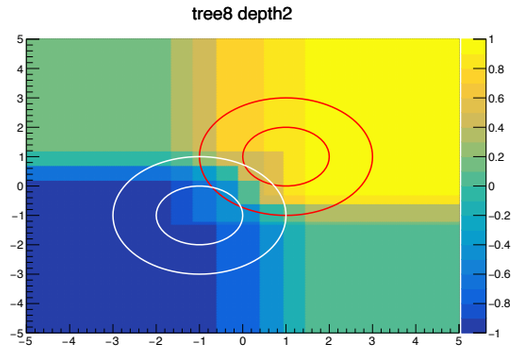
vary



tree8 depth2

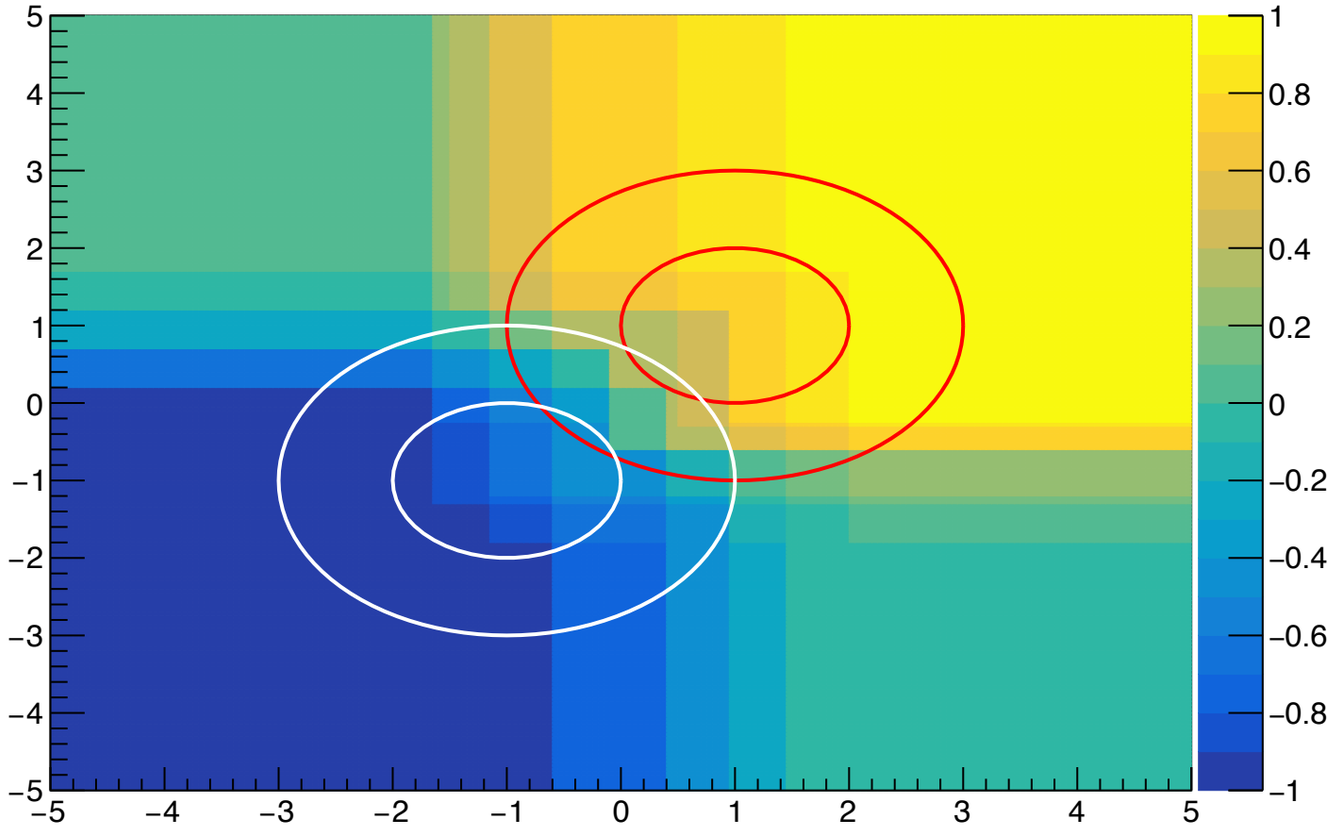


Depth 2

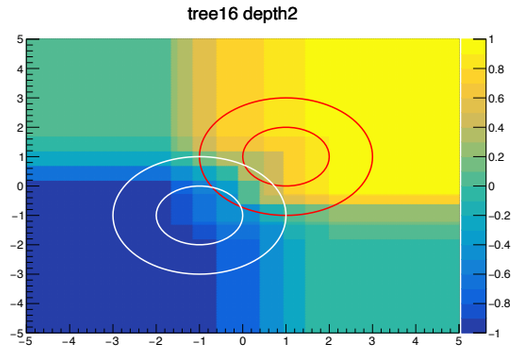


vary

tree16 depth2

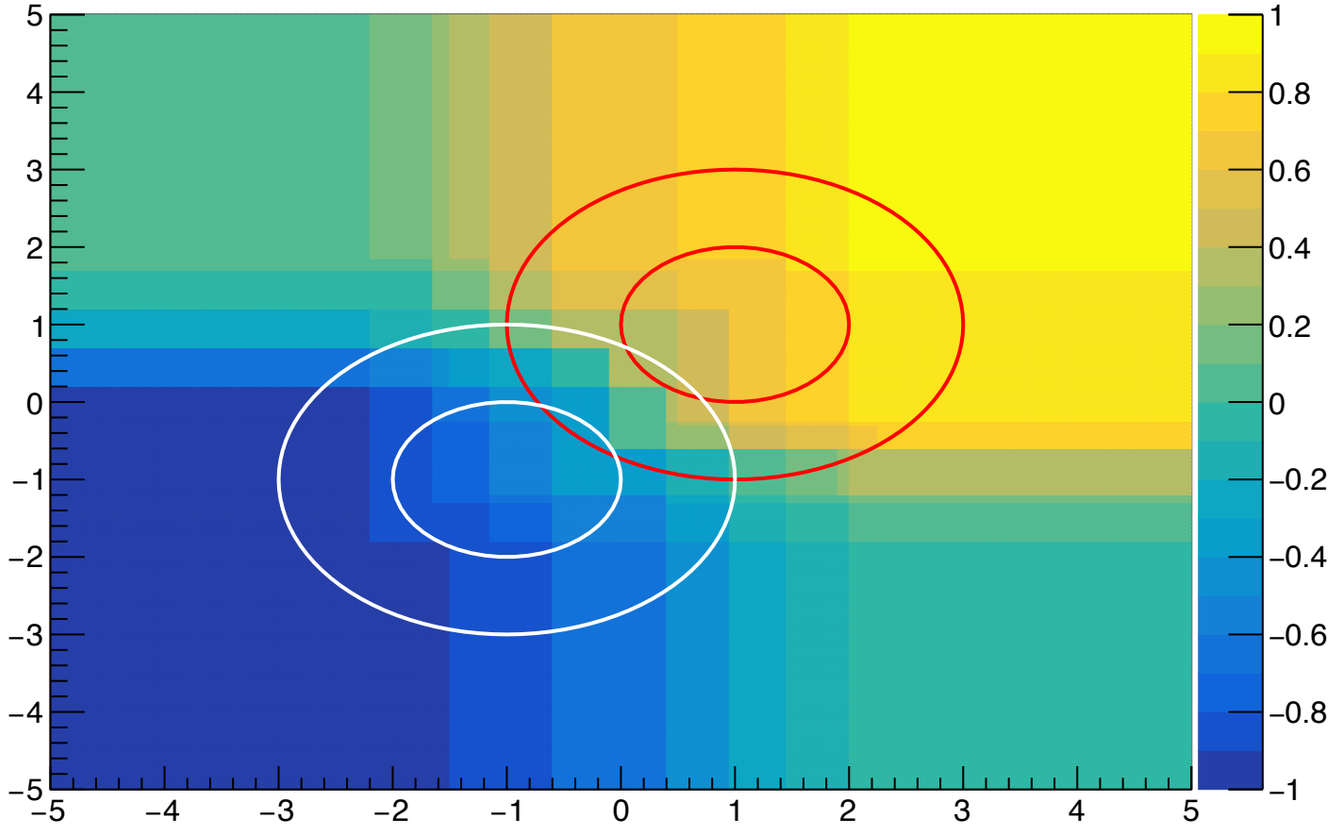


Depth 2

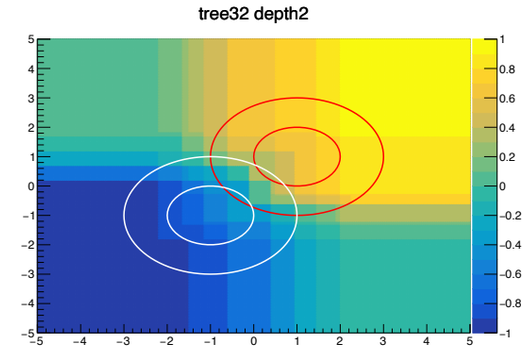


vary

tree32 depth2

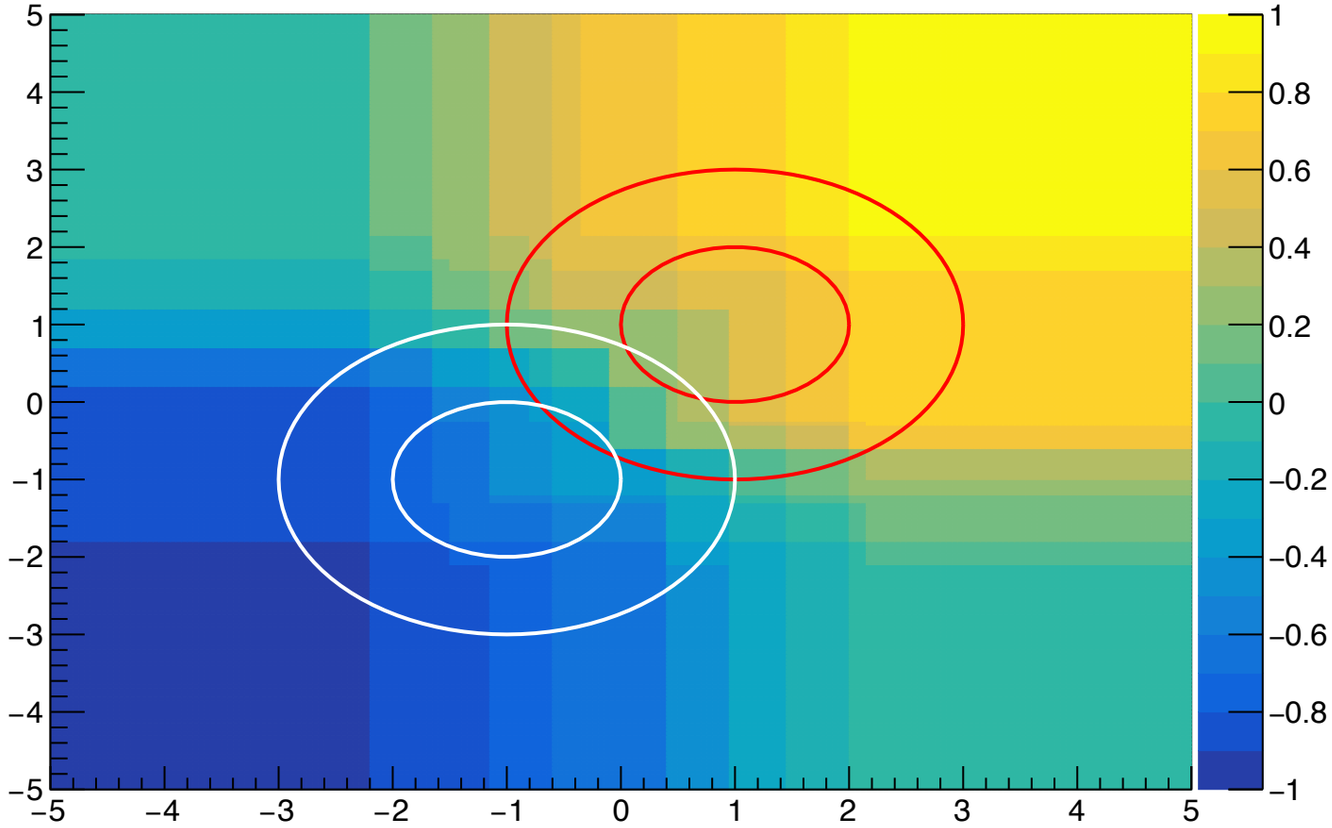


Depth 2

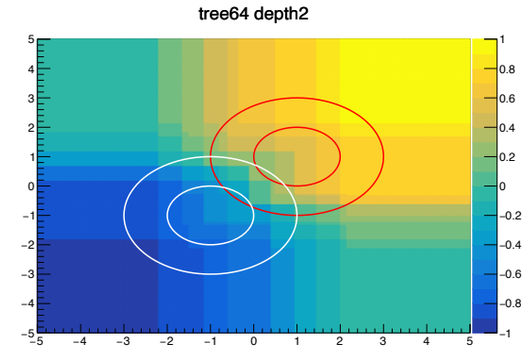


vary

tree64 depth2

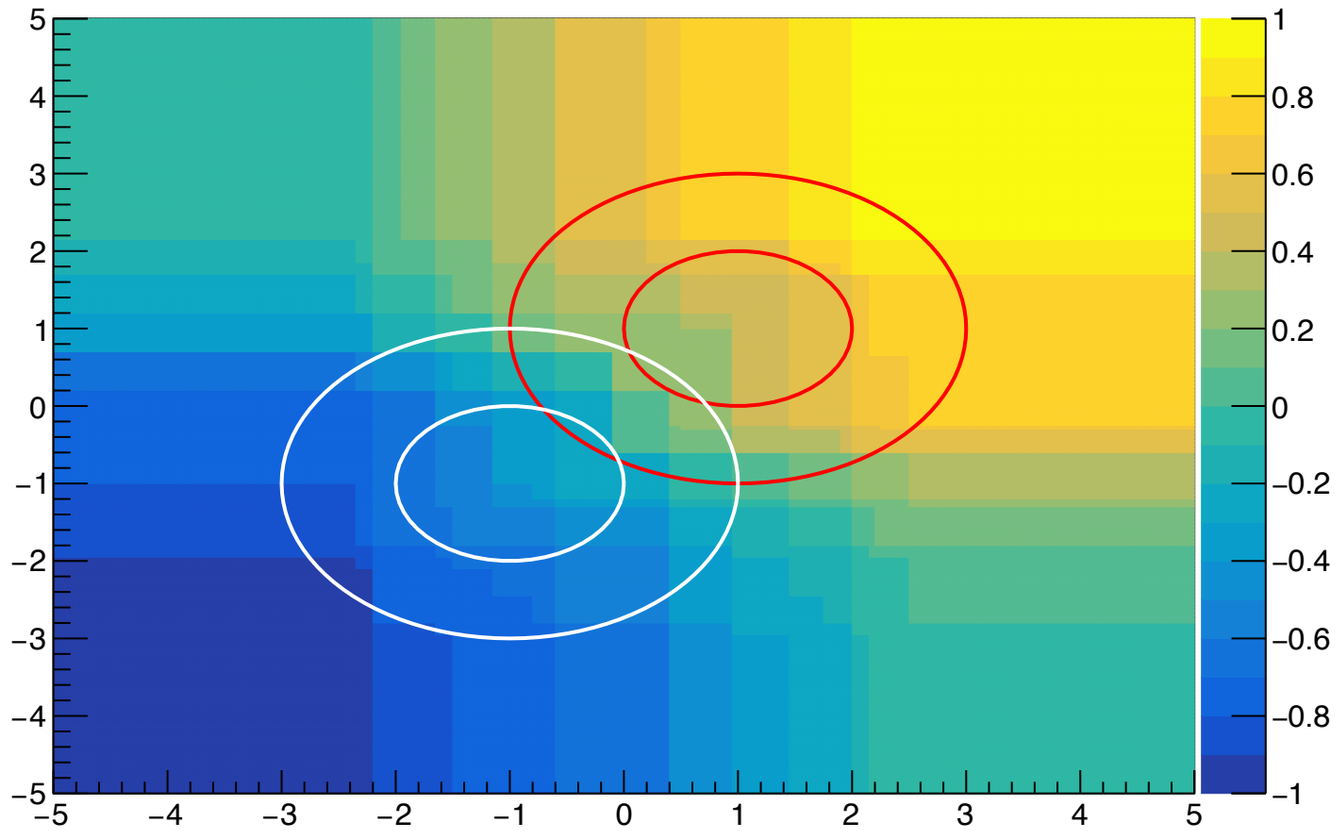


Depth 2



vary

tree128 depth2

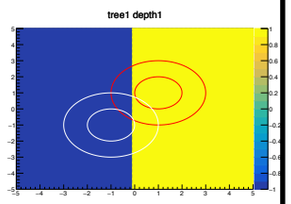
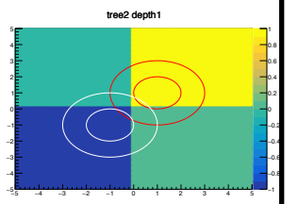
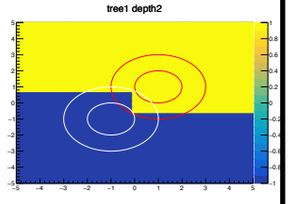
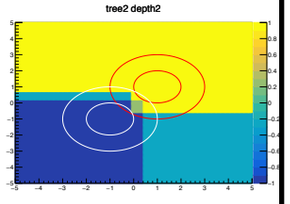
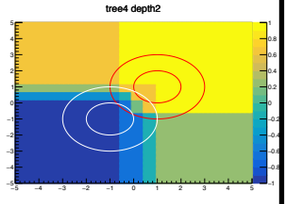
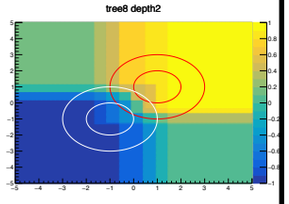
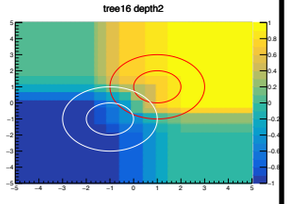
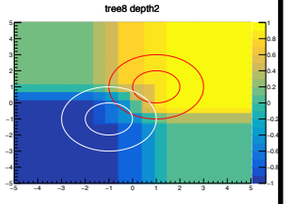
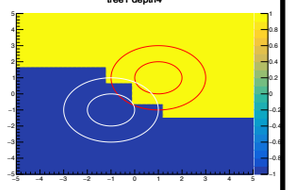
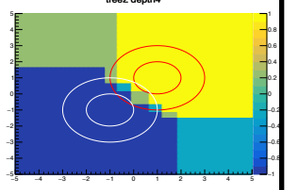
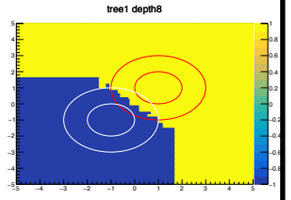
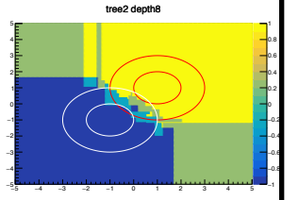
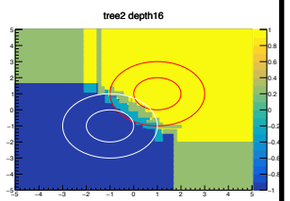


becomes very blurry

Put it together on one slide

Tree →

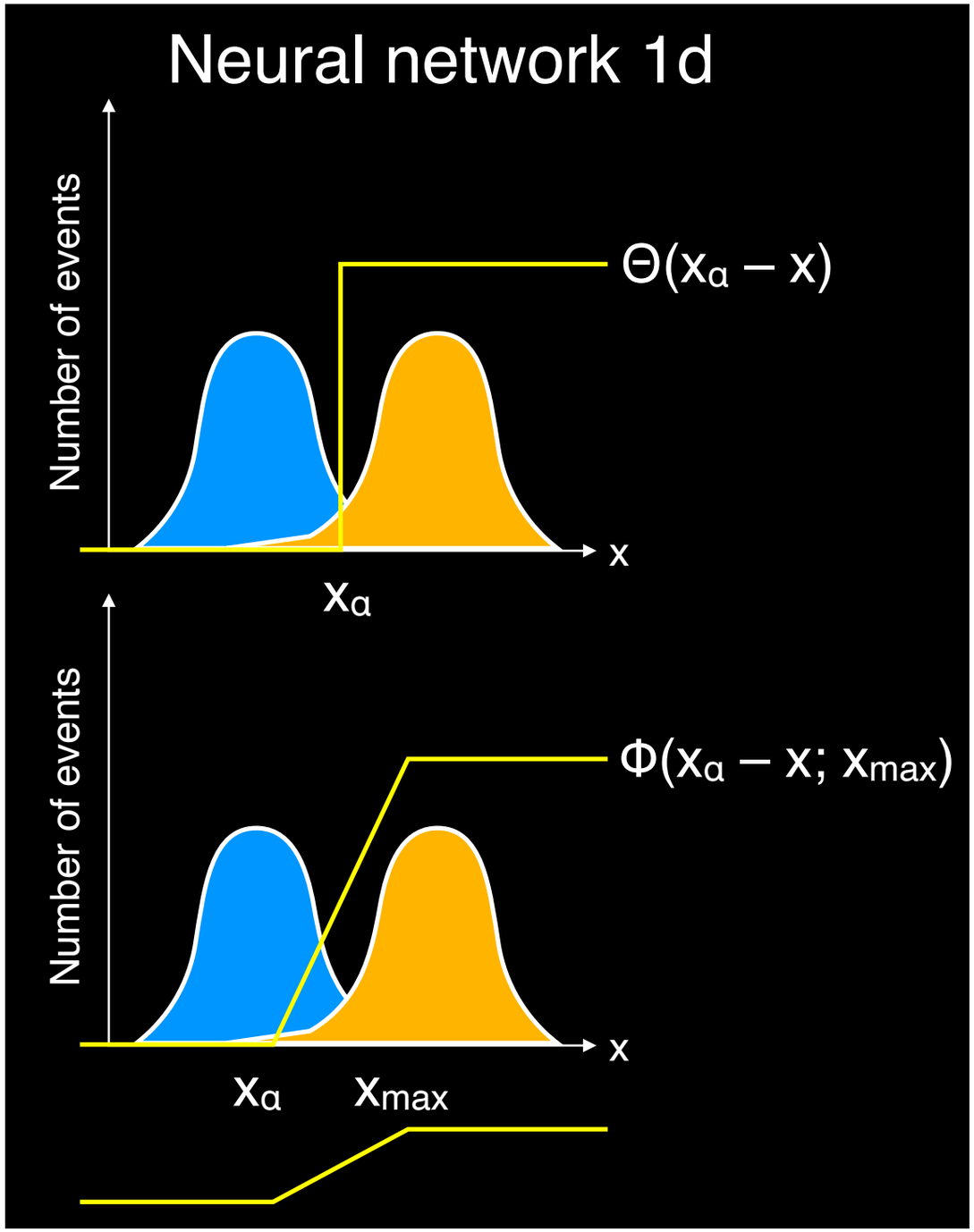
Depth →

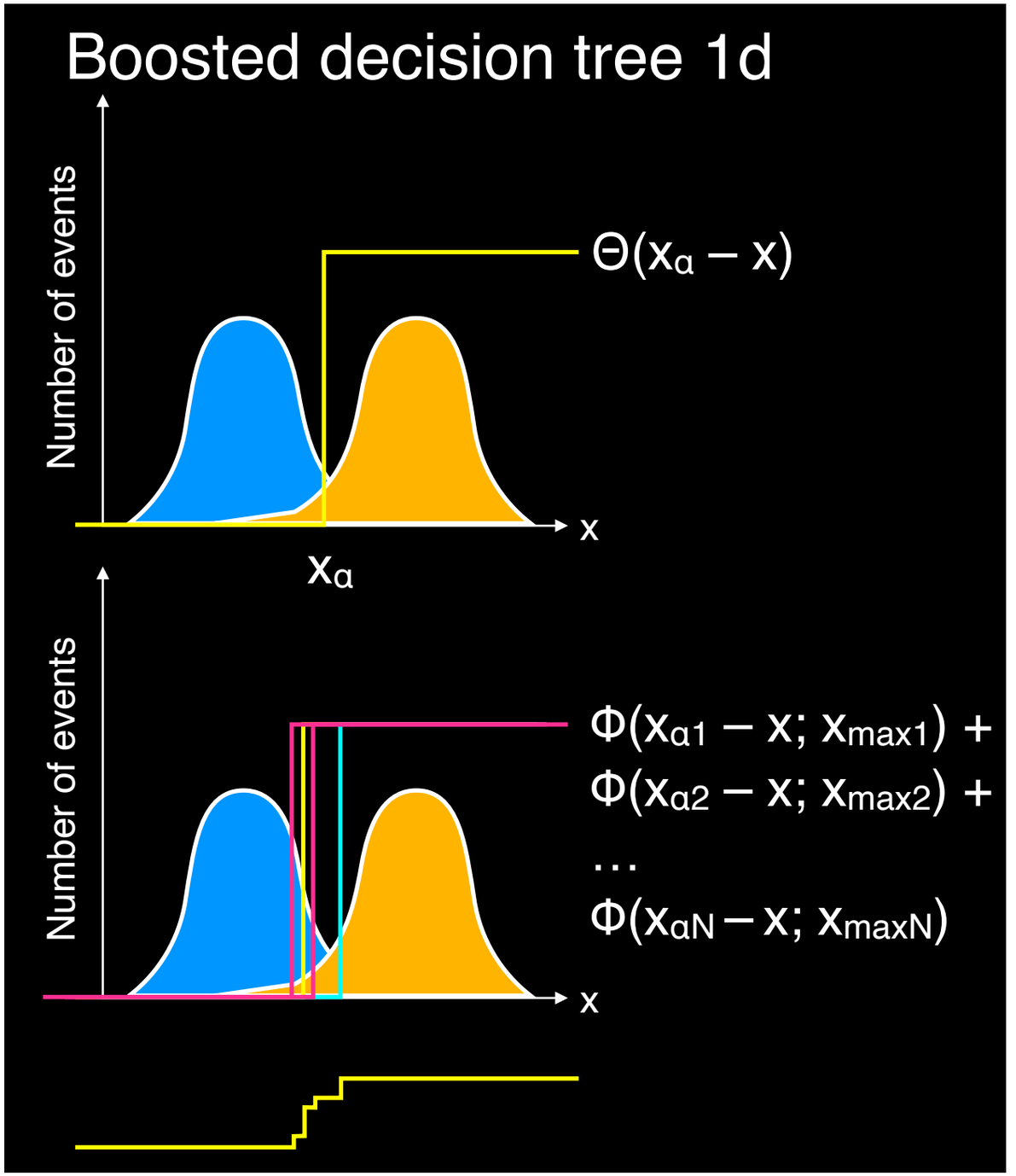
	1	2	4	8	16	32
1						
2						
4						
8						
16						

Sweet spot depends on the physics problem

Forest of decision trees

Fuzzy boundary by averaging step functions





Forest of decision trees provides the gradient

Activation function

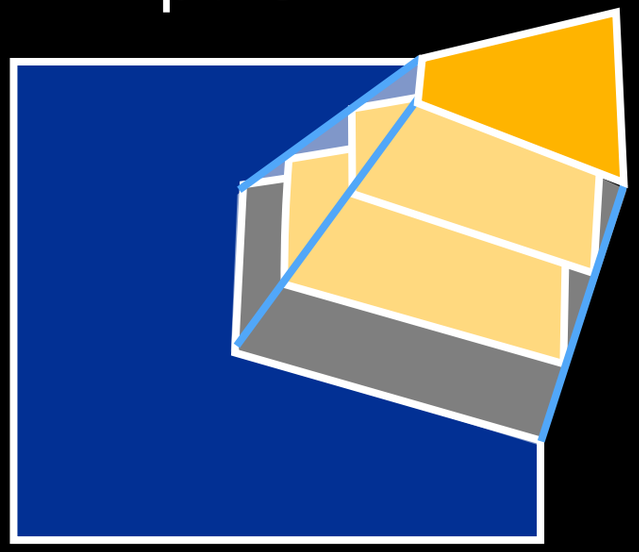
Fuzzy boundary using a function

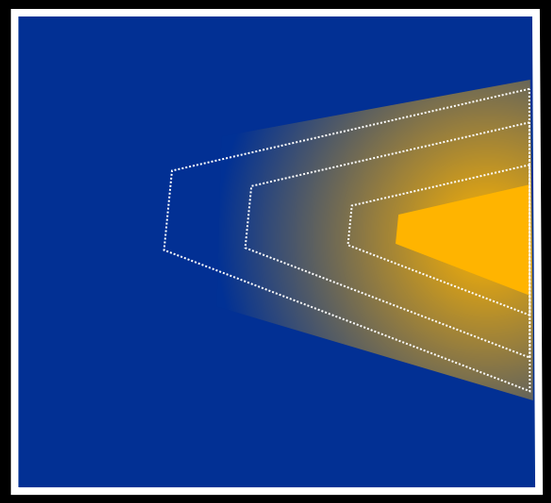
2-dim inputs

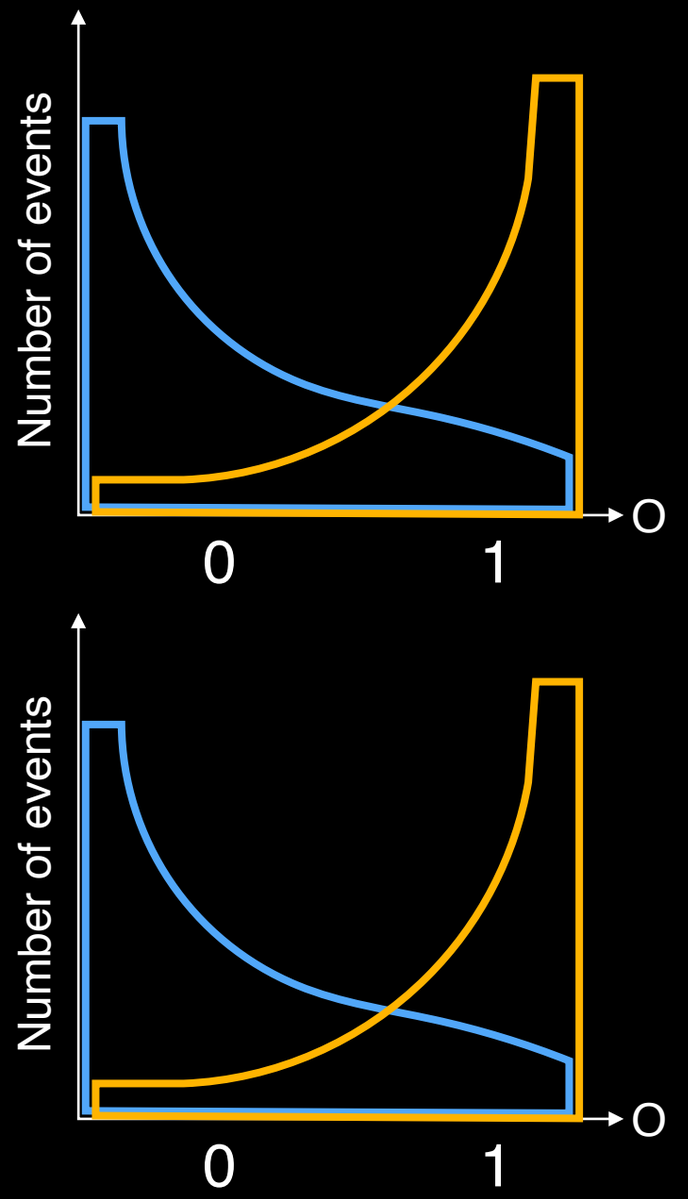
Projection

Output score

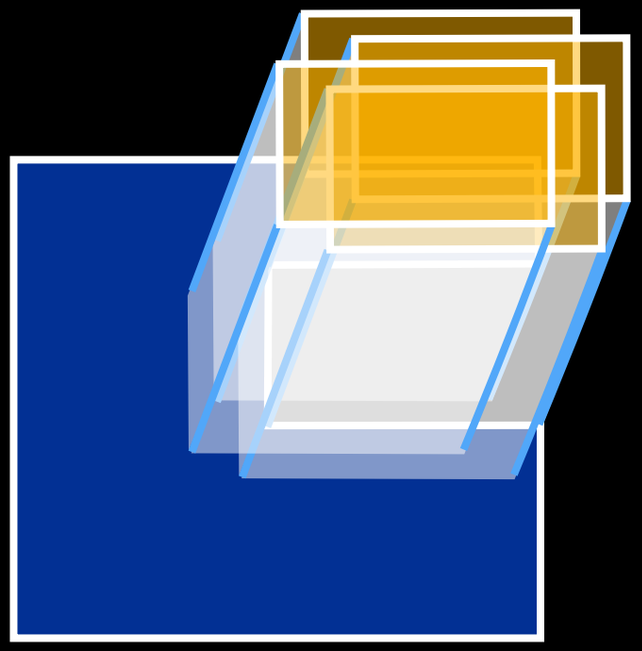
NN

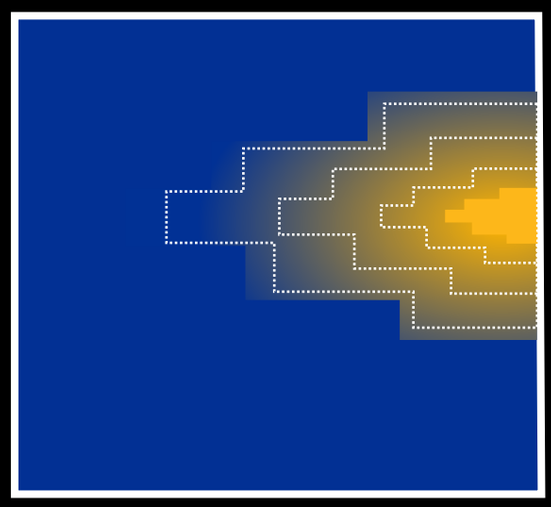






BDT





Different approach, but same result