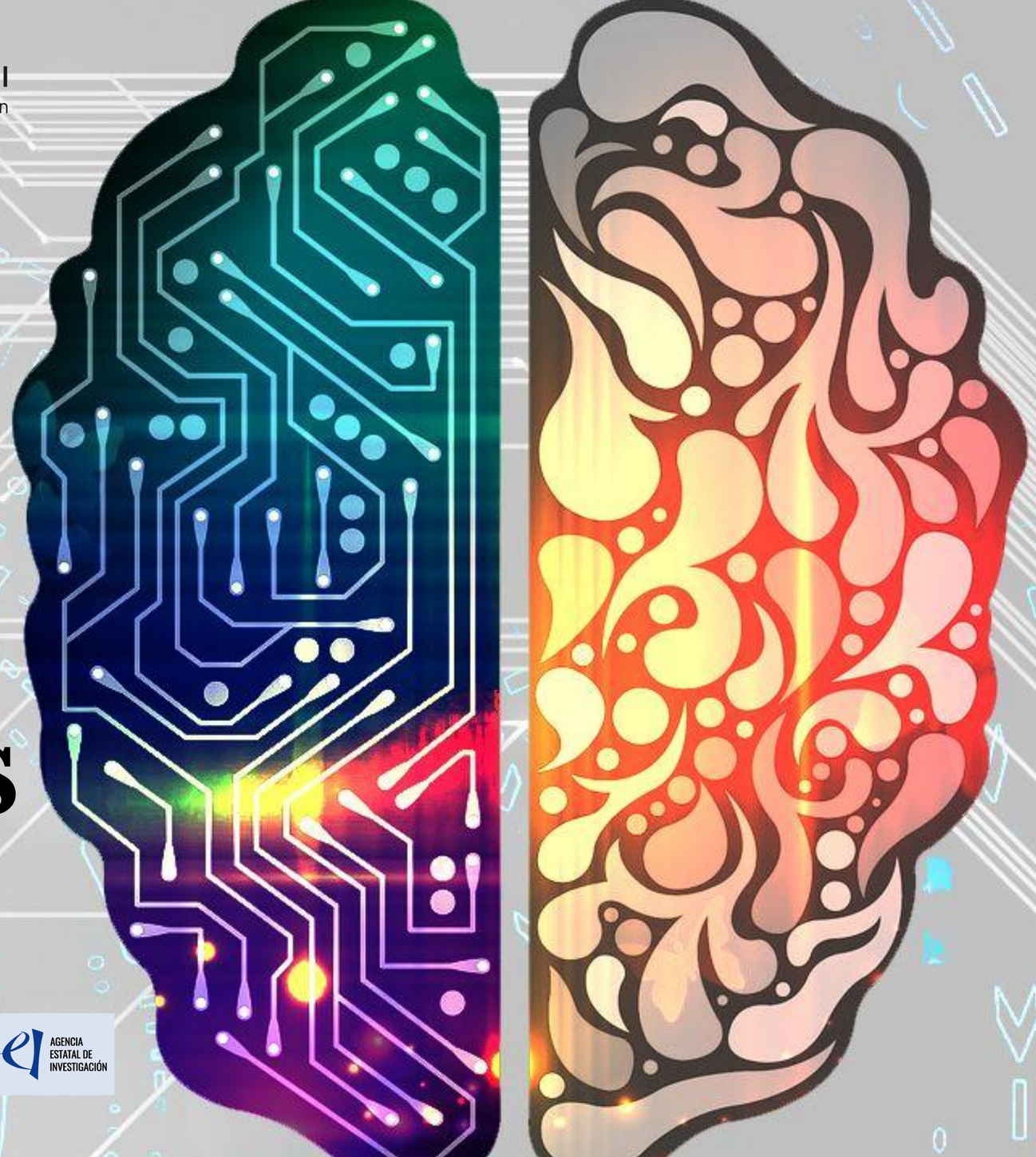
UNIVERSITAT DE BARCELONA

European Research Council
Established by the European Commission

ERC GA: 101077940

COMCHA
Computing challenges

MACHINE LEARNING FUNDAMENTALS



Felipe Luan Souza de Almeida

4th COMCHA School — 09/04/2026

ICCUB
Institut de Ciències del Cosmos
UNIVERSITAT DE BARCELONA

EXCELENCIA MARÍA DE MAEZTU 04/2025-03/2031

MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

AGENCIA ESTATAL DE INVESTIGACIÓN

Outline (lecture 2)

- More on overfitting and generalization
  - Bias-variance trade-off
  - Regularization
- More on models
  - Decision Trees
    - Boosting, random forest, bagging
  - Neural Networks

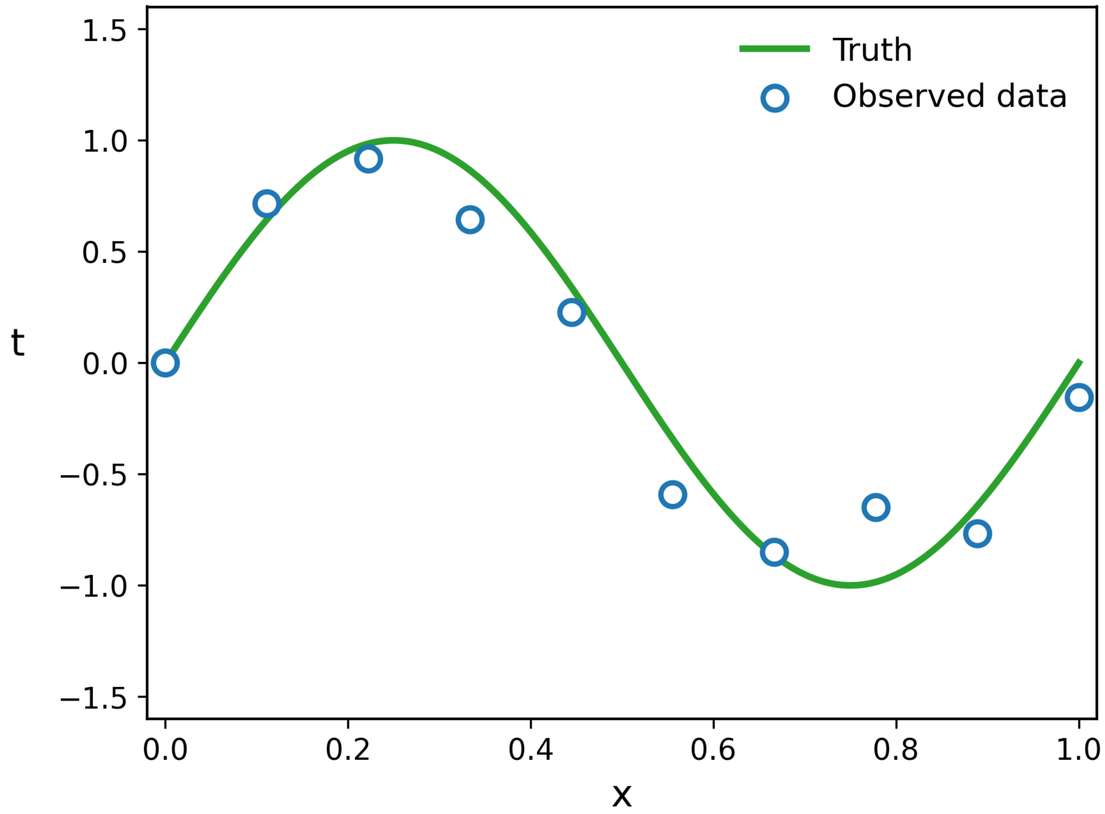
References

- Lectures:
  - *M. Kagan*: CERN Summer Student Lecture (2024) and TRISEP school (2024)
  - *F. Fleuret*: Deep Learning Course (2018-now, EPFL and University of Geneva)
  - *T. Sculac*: CERN Thematic School of Computing on Machine Learning (2024)
  - *F. Vaselli*: CERN Thematic School of Computing on Machine Learning (2024)
  - *D. Noll*: MLPF Berkley school (2024)
  - *L. Moneta*: Machine Learning for Fundamental Physics school (2025)

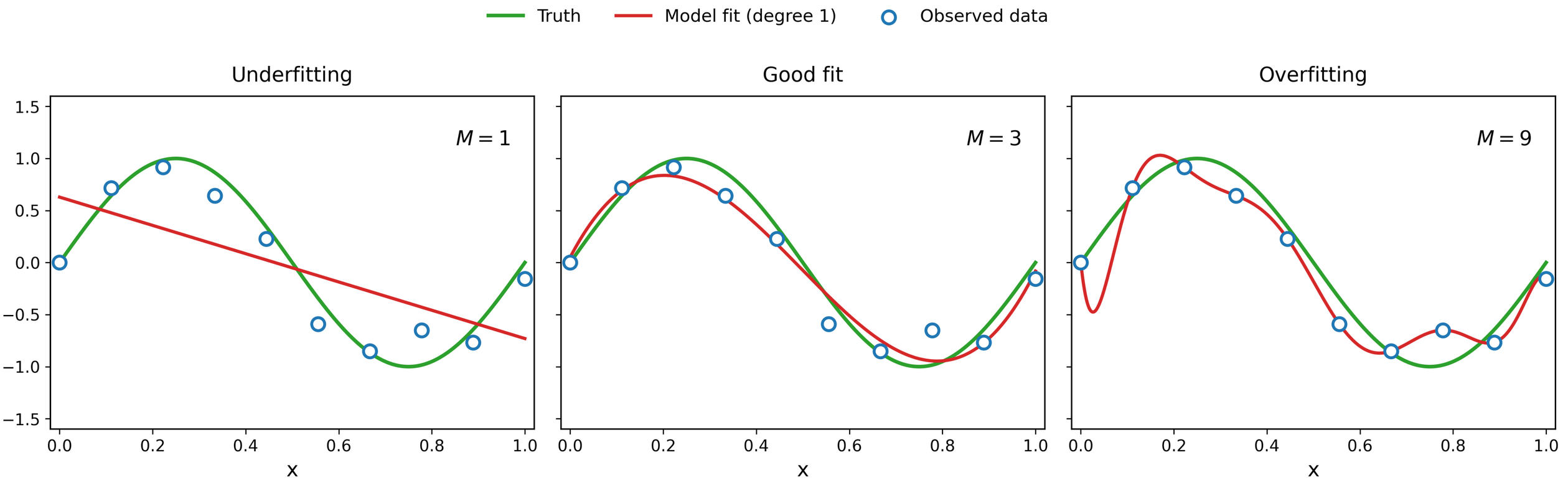
- Books:
  - Deep Learning (*I. Goodfellow, Y. Bengio, A. Courville*)
  - The Elements of Statistical Learning (*T. Hastie, R. Tibshirani, J. Friedman*)
  - Pattern Recognition and Machine learning (*C. Bishop*)

More on underfitting, overfitting, generalization…

- Suppose we want to model the following data:



More on underfitting, overfitting, generalization…



$$f(x \mid \mathbf{w}) = w_0 + w_1 x$$

Large bias

The model does not reproduce well the data

$$f(x \mid \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

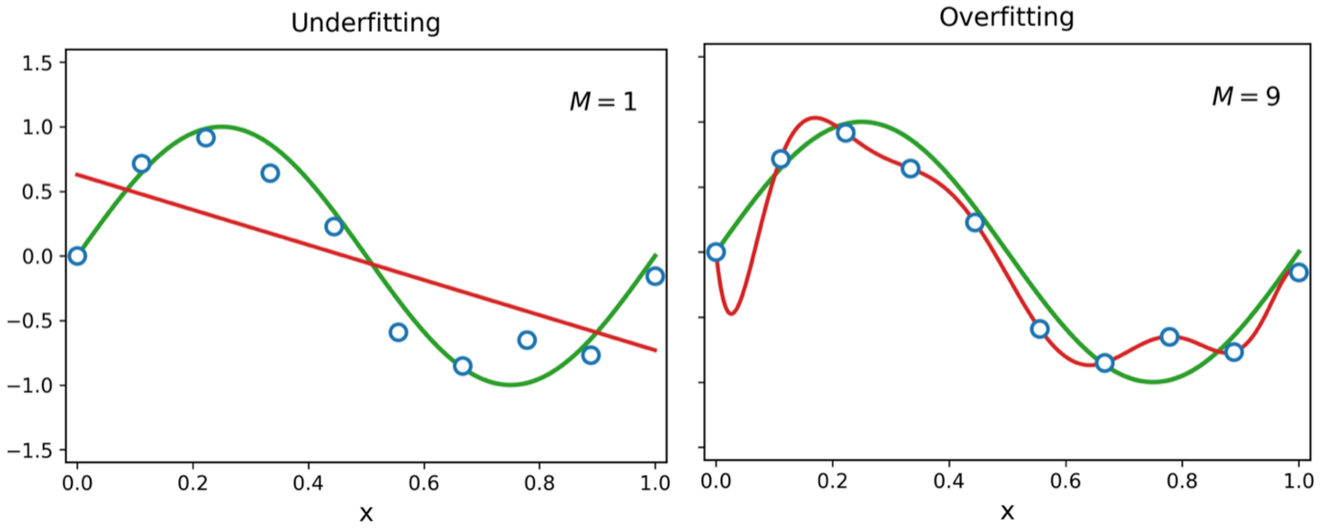
$$f(x \mid \mathbf{w}) = w_0 + w_1 x + \ldots + w_9 x^9$$

Large variance

The model represents the training data too well

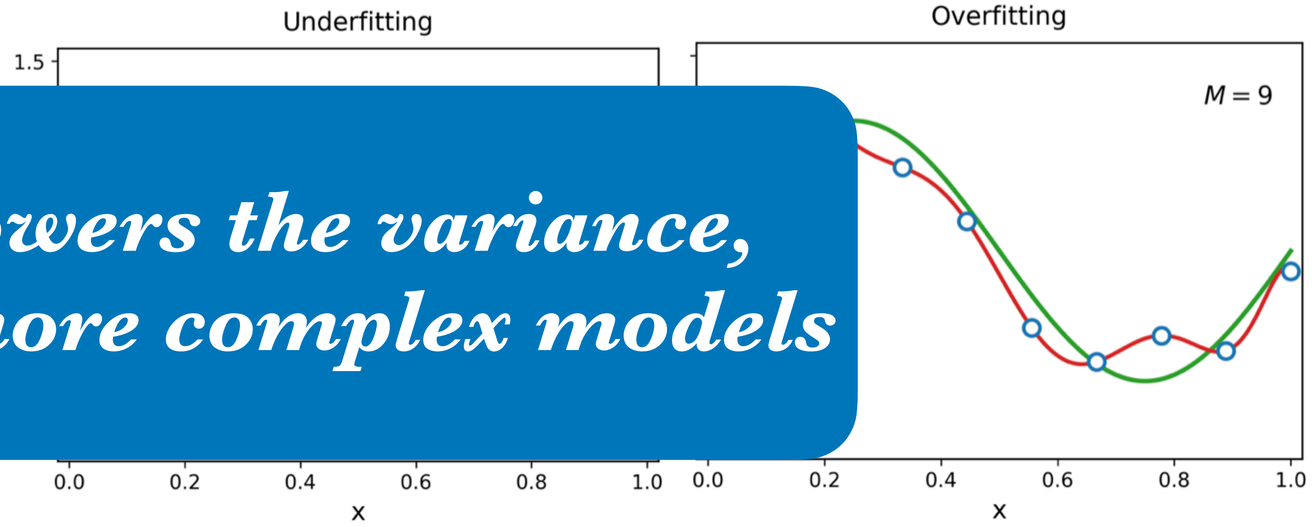
Bias-variance trade-off

- Simple model under-fit:
  - High bias, low variance
  - Fails to capture underlying structure (too
- Complex model over-fit:
  - Low bias, high variance
  - Learns statistical fluctuations
- Bias: systematic error
  - How far predictions are, on average, from the true function
- Variance: sensitivity of the dataset
  - How much predictions change with different training samples



Bias-variance trade-off

- Simple model under-fit:
  - High bias, low variance
  - Fails to capture
- Complex model
  - Low bias, high
  - Learns statisti
- Bias: systematic error
  - How far predictions are, on average, from the true function
- Variance: sensitivity of the dataset
  - How much predictions change with different training samples



More data lowers the variance, allowing for more complex models

Regularization

- Idea: control model complexity by penalizing parameters
  - Strategy: modify the loss function adding a penalty term on the weights
- Example for regression:
  - Regularized square loss function:

$$\mathscr{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(\mathbf{x_i}; \mathbf{w}))^2 + \lambda\Omega(\mathbf{w})$$

- Common choices:
  - L2 norm (Ridge):

$$\Omega(\mathbf{w}) = ||\mathbf{w}||^2 = \sum_i w_i^2$$

    - Keeps weights small
  - L1 norm (Lasso):

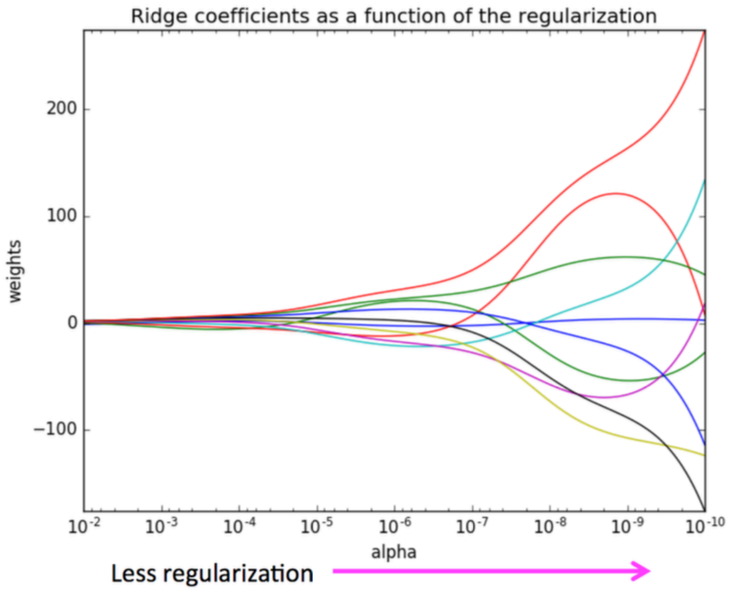
$$\Omega(\mathbf{w}) = ||\mathbf{w}|| = \sum_i |w_i|$$

    - Promotes sparsity (many weights = 0)

L1 vs L2

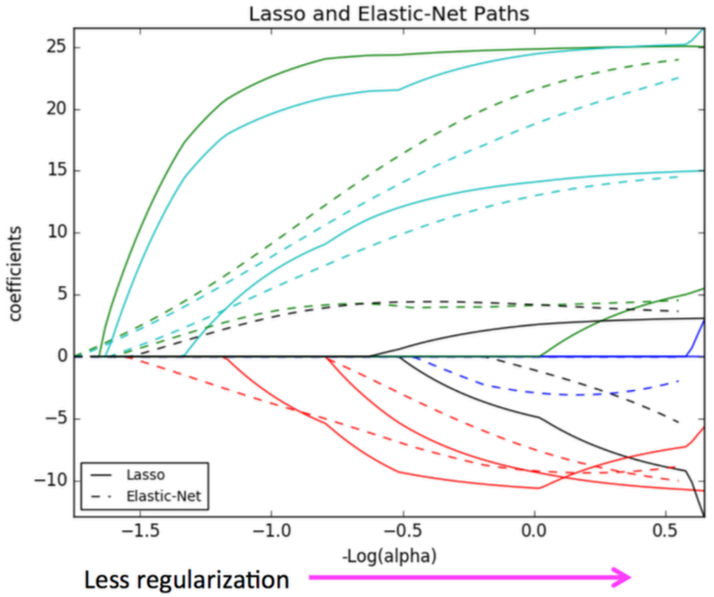
$$L(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^2 + \alpha\Omega(\mathbf{w})$$

$$L2: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||^2$$



- Shrinks all weights smoothly
- No exact zeroes

$$L1: \quad \Omega(\mathbf{w}) = ||\mathbf{w}||$$



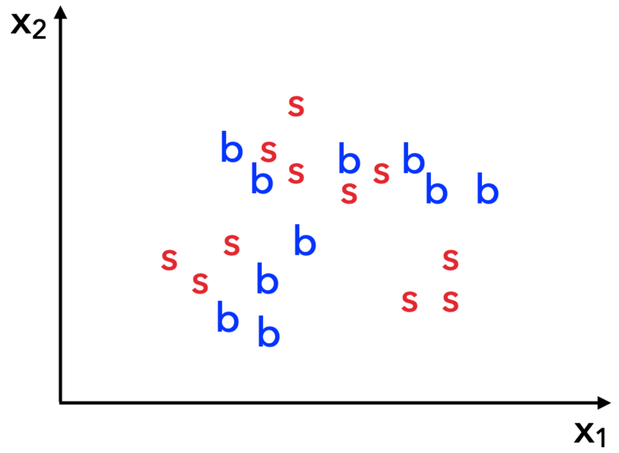
- Drives some weights exactly to 0
- Performs feature selection

From concepts to models

- We now introduce concrete models:
  - Decision Trees
  - Neural Networks (MLPs)
- Key question:
  - How each model control complexity?
- Each model:
  - Has its own inductive bias
  - Has its own regularization mechanisms

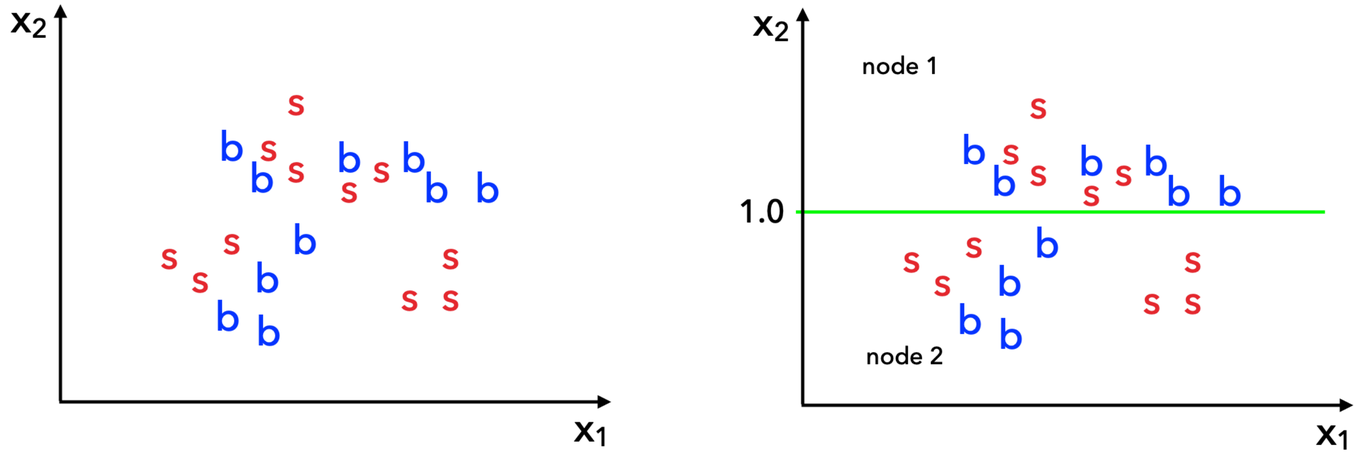
Decision Trees

- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)



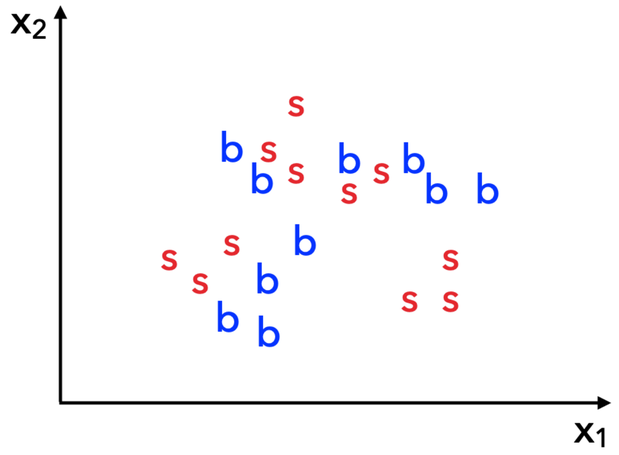
Decision Trees Classification

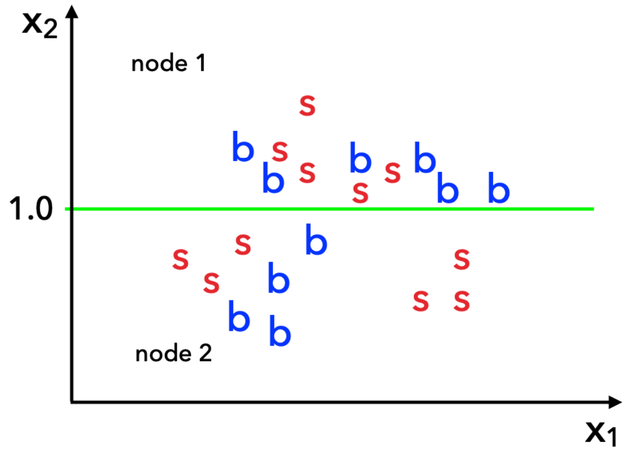
- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)

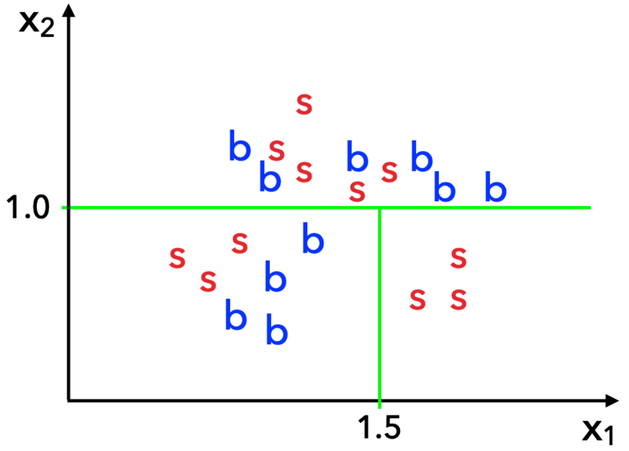


Decision Trees Classification

- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)

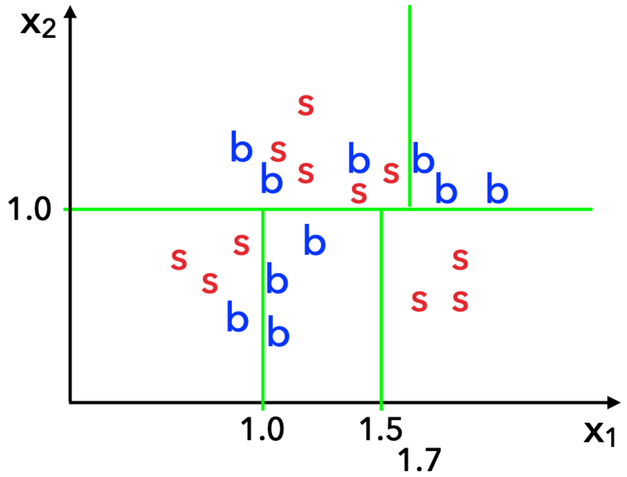


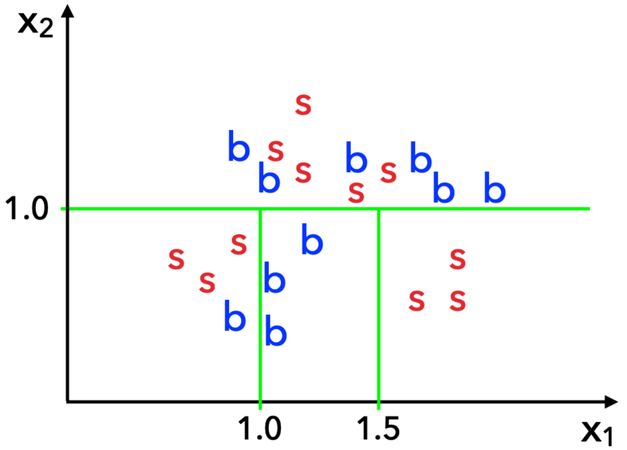


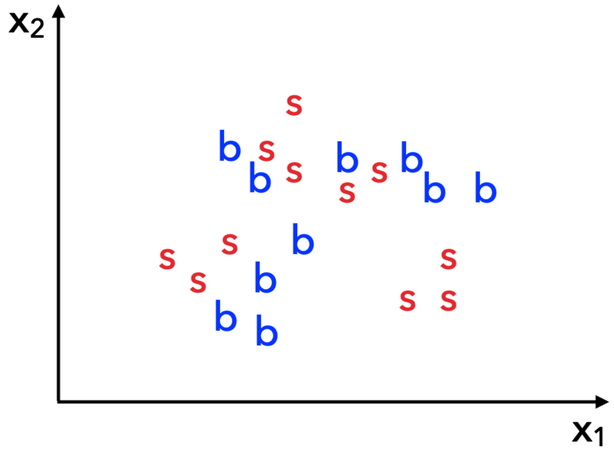


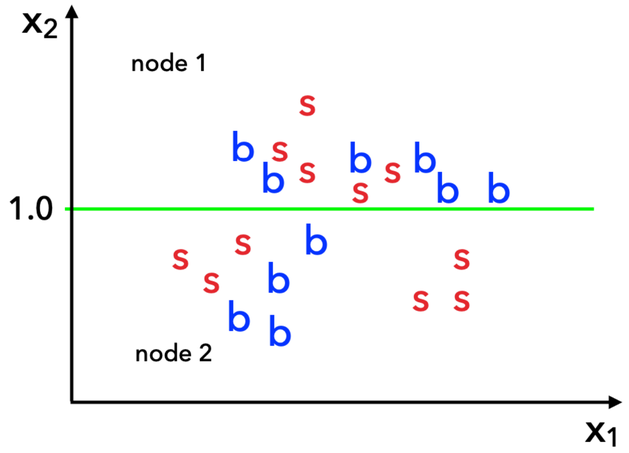
Decision Trees Classification

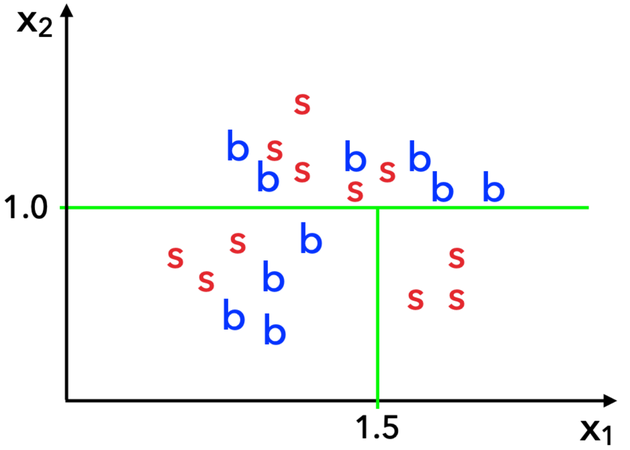
- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)





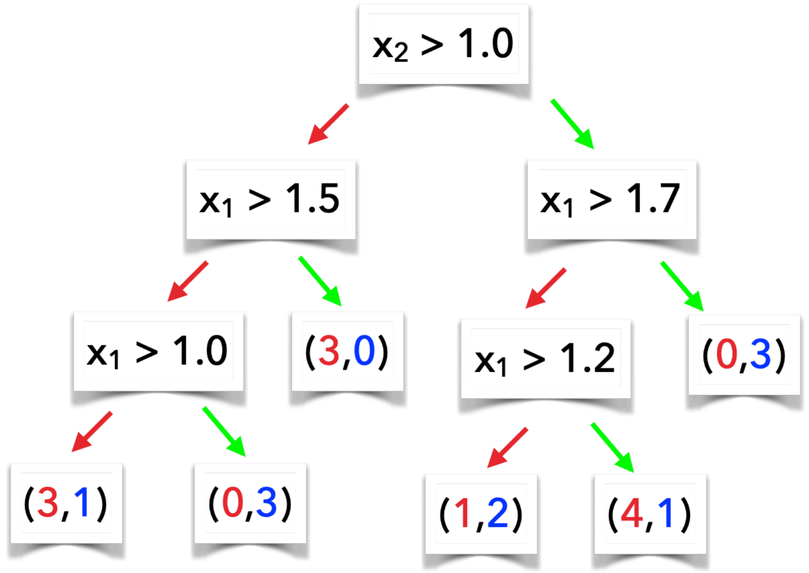


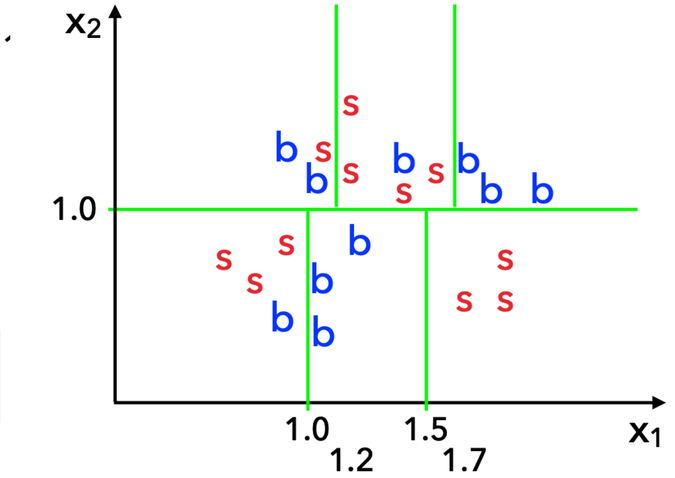




Decision Trees

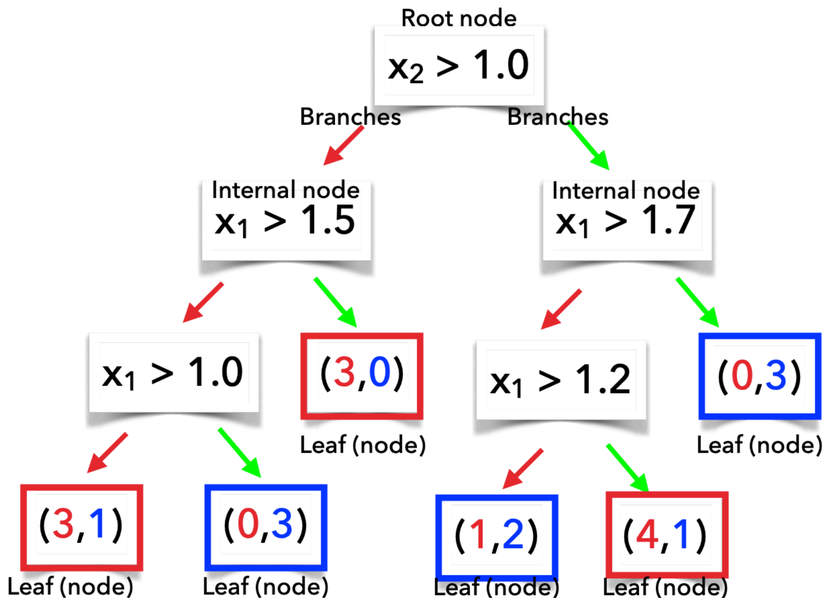
- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)

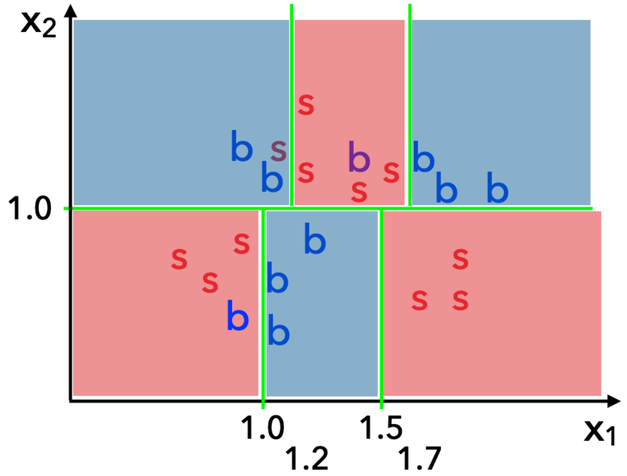




Decision Trees Classification

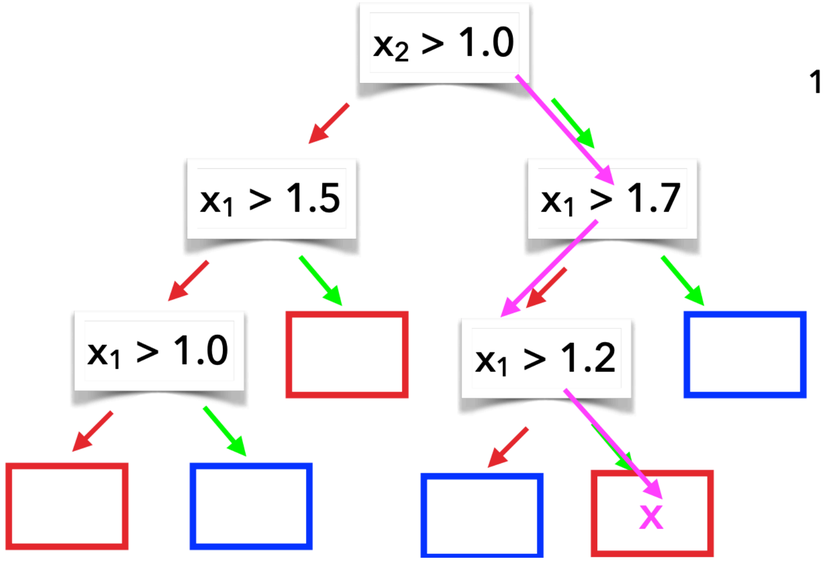
- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)

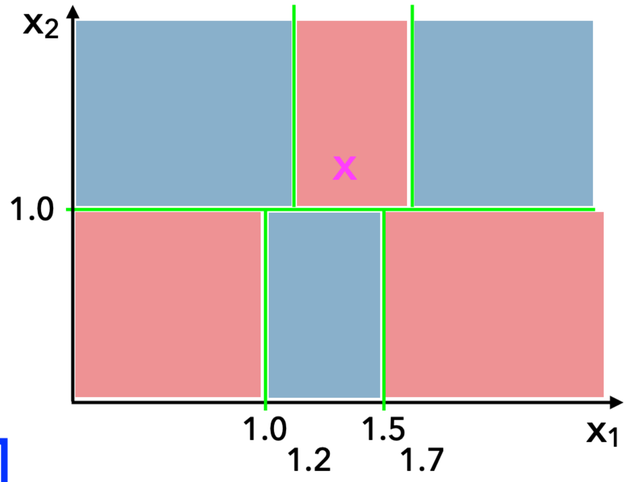




Decision Trees Classification

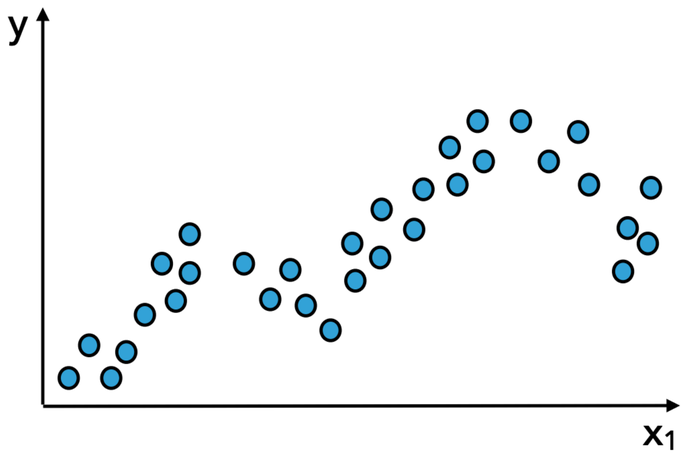
- Model based on recursive partitioning of the feature space
- At each node:
  - Apply a binary split (e.g. $x_j < t$)
- Final prediction:
  - Constant value per region (leaf)





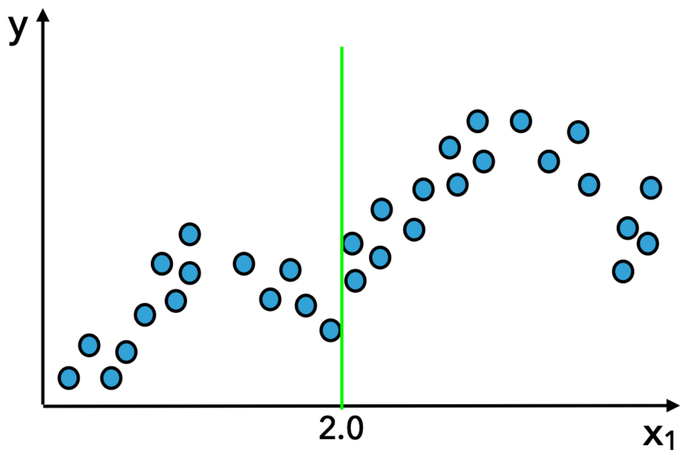
Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



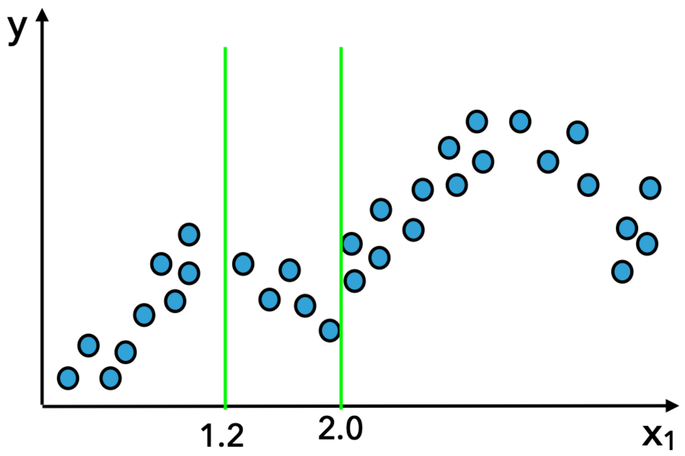
Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



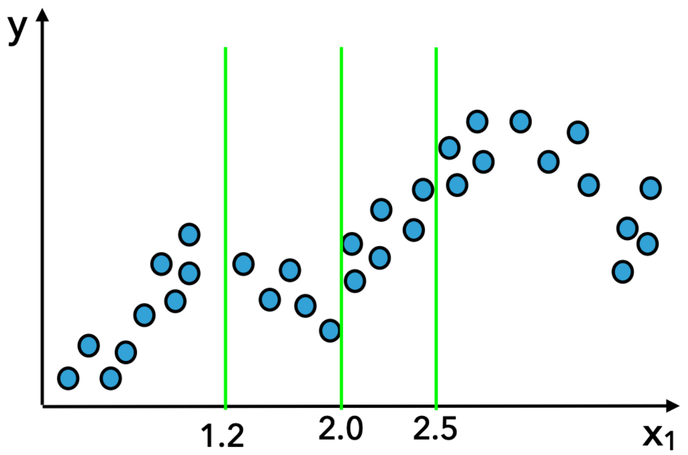
Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



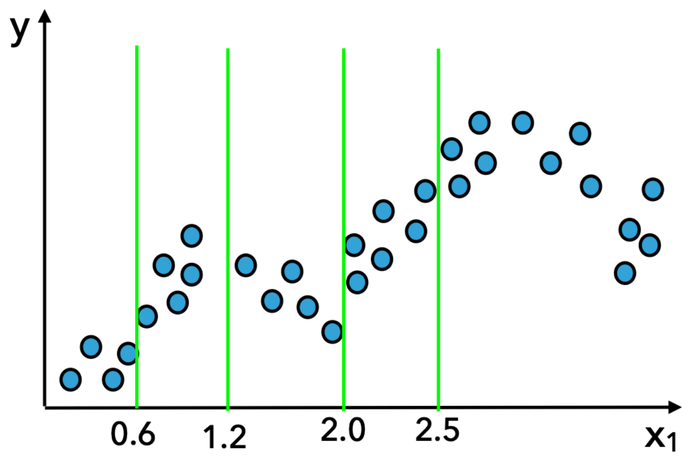
Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



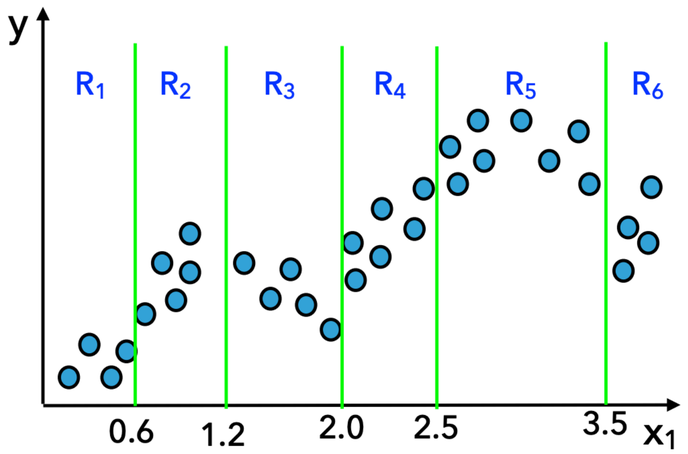
Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



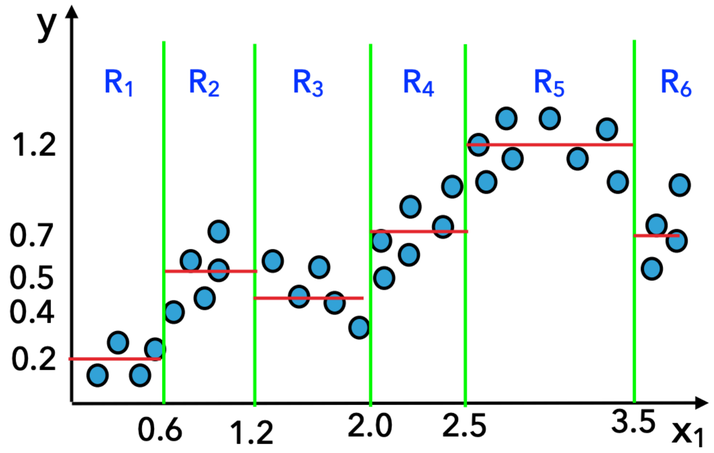
Decision Tree Regression

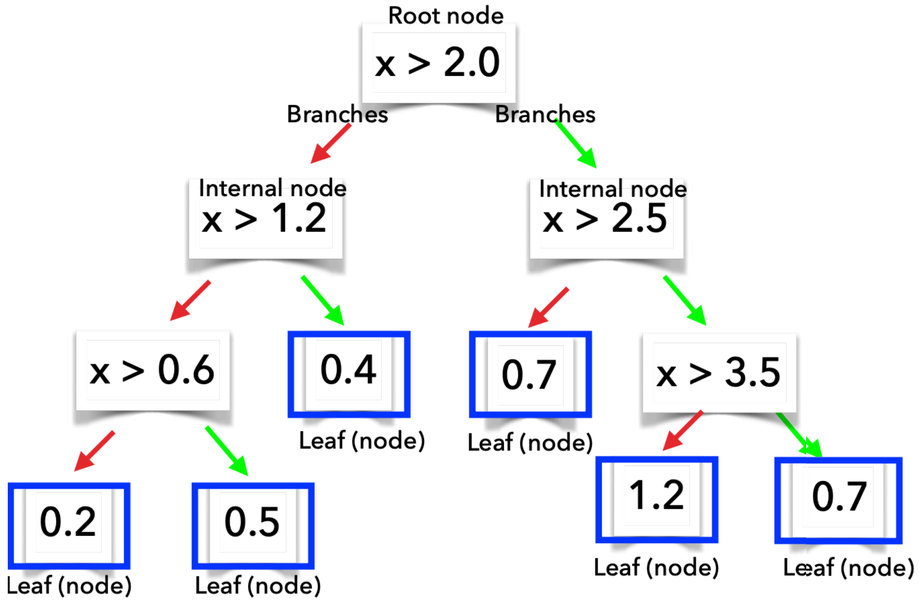
- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead



Decision Tree Regression

- From a 2D training dataset (x,y) we want to predict the y value for each new data point x
- Now we need to predict a continuous value instead
- Average in each region gives you the prediction

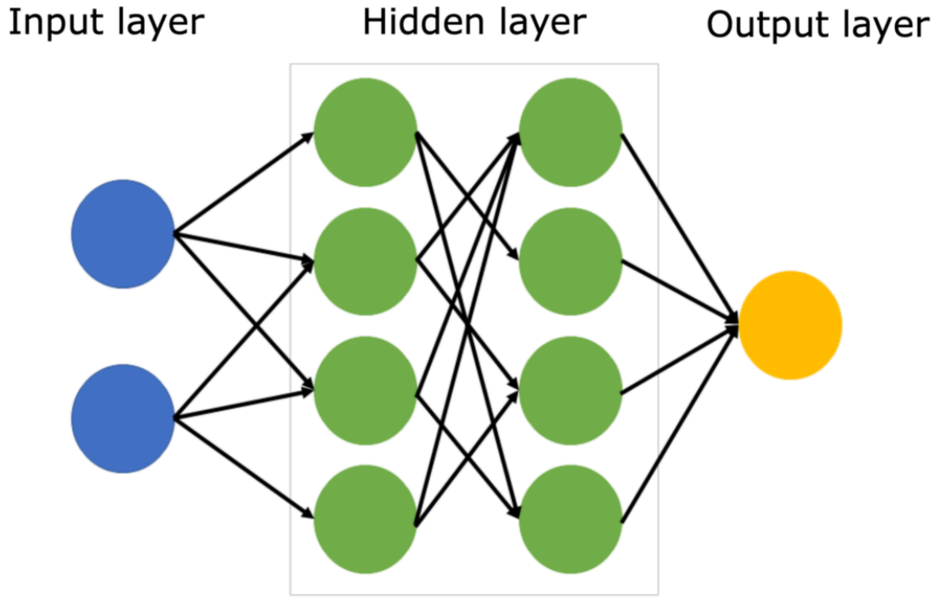




Neural Networks

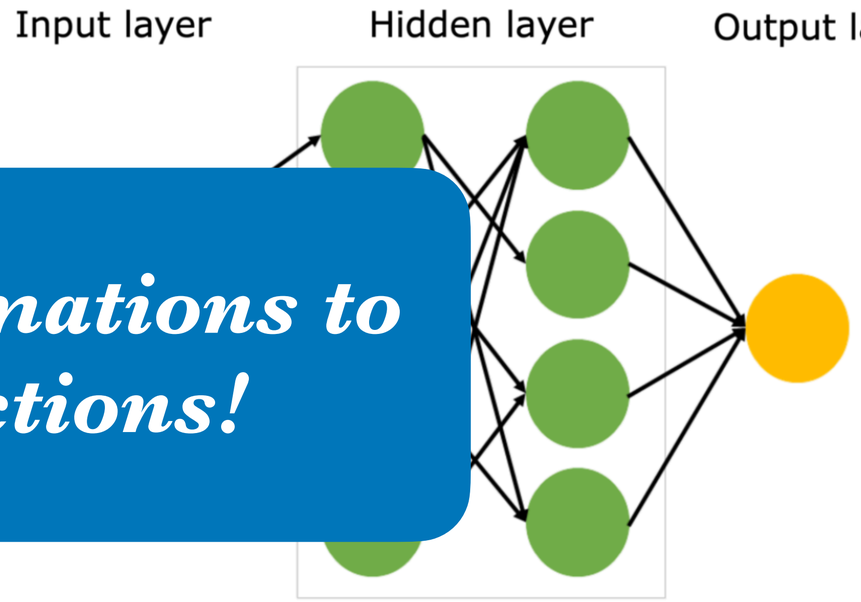
- Trees:
  - Piecewise constant predictions
- What if we want:
  - Smoother functions?
  - Continuous approximations?
- Multilayer Perceptron (MLP):
  - Composition of layers
  - Each layer:
    - Linear transformation
    - Non-linear activation function

$$\mathbf{h}^{(l)} = \sigma\left(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right)$$



Neural Networks

- Trees:
  - Piecewise constant predictions
- What if we want:
  - Smoother funct
  - Continuous app
- Multilayer Percept
  - Composition of layers
  - Each layer:
    - Linear transformation
    - Non-linear activation function



Stack simple transformations to build complex functions!

$$\mathbf{h}^{(l)} = \sigma\left(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right)$$

Why non-linearity matters?

- Without non-linearity:
  - Multiple layers colapse into a single linear model
- With non-linearity:
  - Model can approximate highly complex functions
- Universal approximation:
  - *MLPs can approximate any continuous function (with enough capacity)!*

MLP complexity and regularization

- MLP complexity:
  - Number of layers
  - Number of neurons
  - Activation functions
- Regularization
  - Weight decay
  - Dropout
  - Early stopping

MLP complexity and regularization

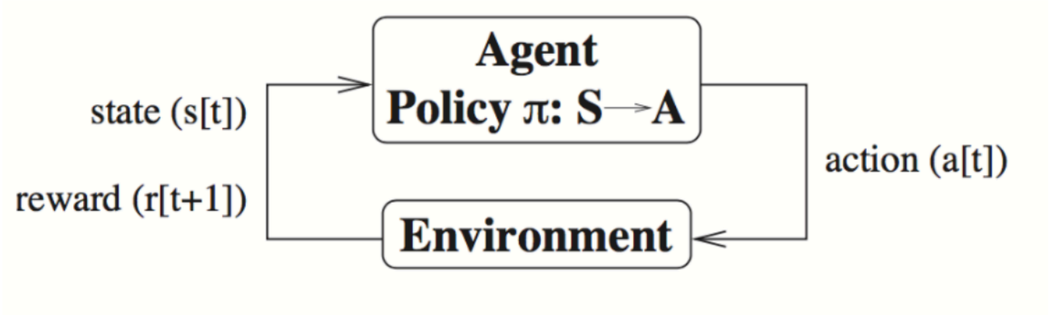
- MLP complexity:
  - Number of layers
  - Number of ne
  - Activation fu
- Regularization
  - Weight decay
  - Dropout
  - Early stopping

MLPs are flexible and powerful, BUT require tuning

Thank you!

Backup

Reinforcement Learning



[Ravikumar]

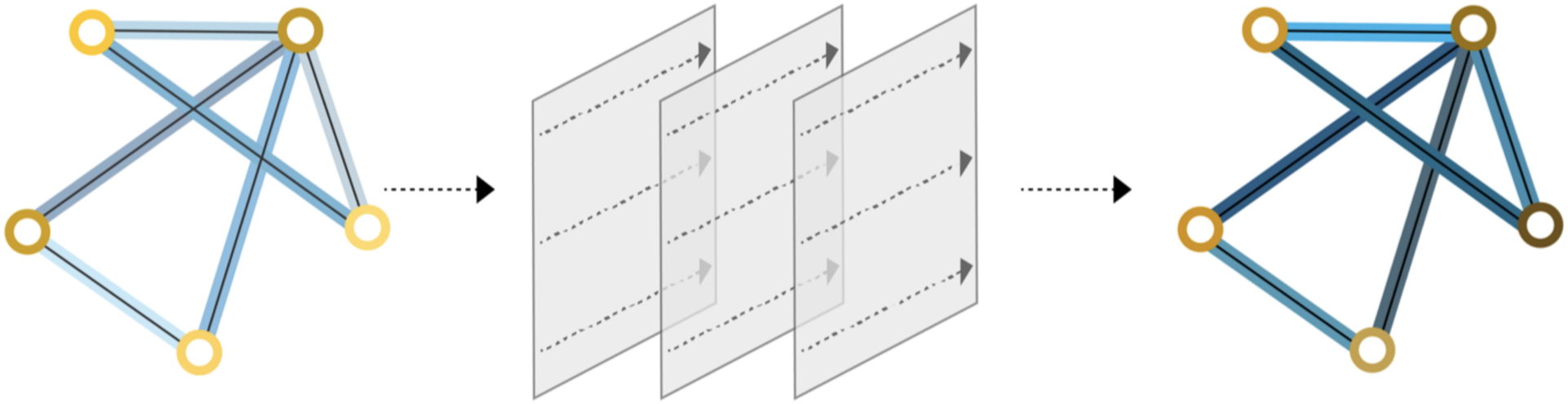
- Models for agents that take actions depending on current state
  - Actions incur rewards, and affect future states ("feedback")

- Learn to make the best sequence of decisions to achieve a given goal when feedback is often delayed until you reach the goal

[M. Kagan]

GNNs for calorimeters

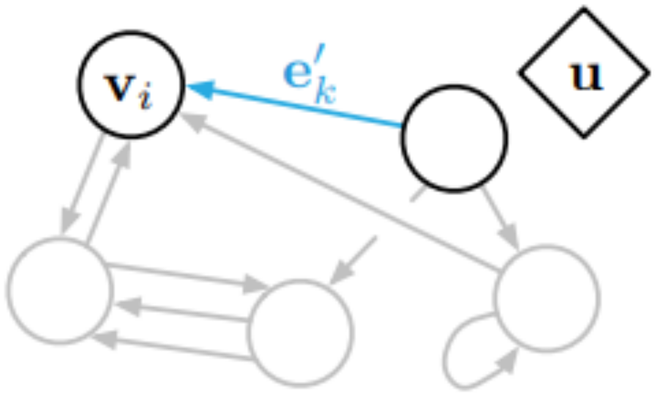
- Inspired by Belle II: GNN reconstruction [Comput Software Big Sci 7, 13 (2013)]
  - GNNs do not require the same concessions to handle irregular geometries
  - Clusters represented as graphs: cells as nodes, edges describing relationships among cells (e.g. spatial distance), and global features for the cluster (e.g. total energy)



The message passing algorithm

- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



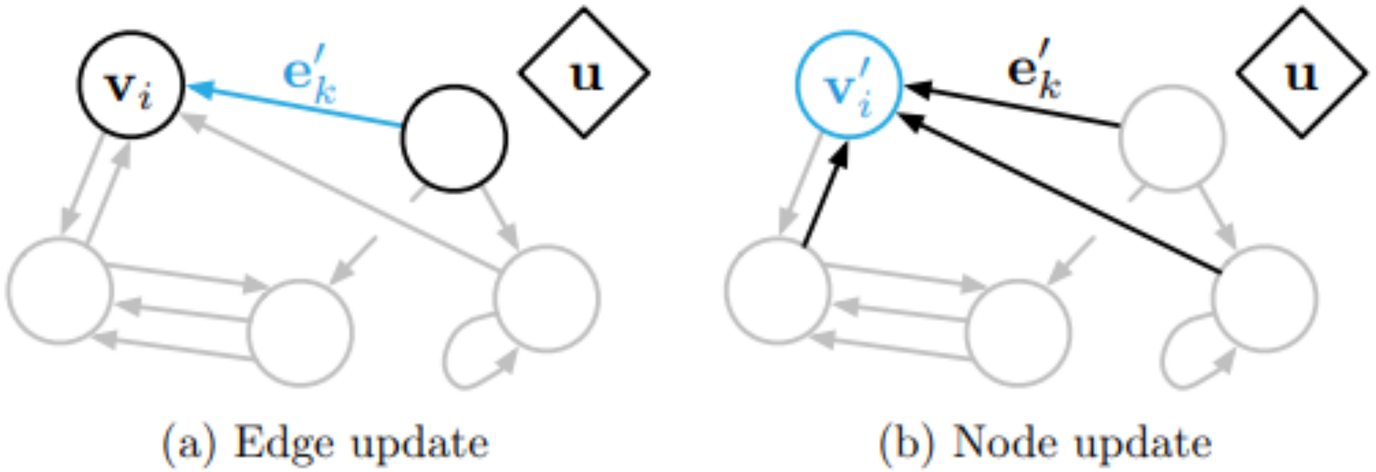
(a) Edge update

Update edges using
connected nodes and globals

The message passing algorithm

- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



(a) Edge update

(b) Node update

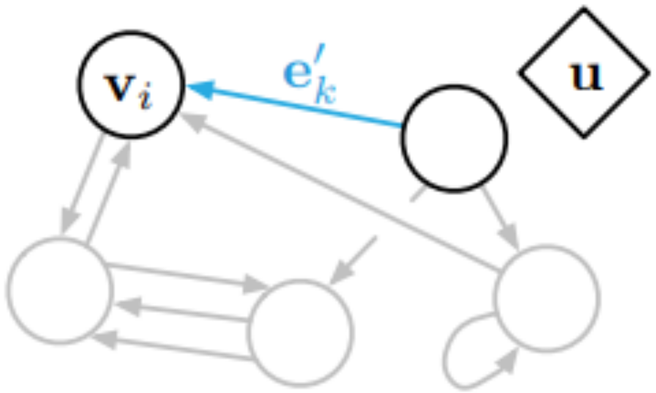
Update edges using connected nodes and globals

Update nodes using (new) edges and globals

The message passing algorithm

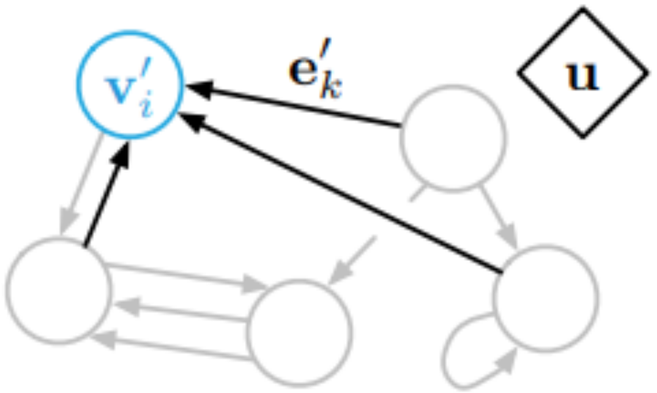
- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



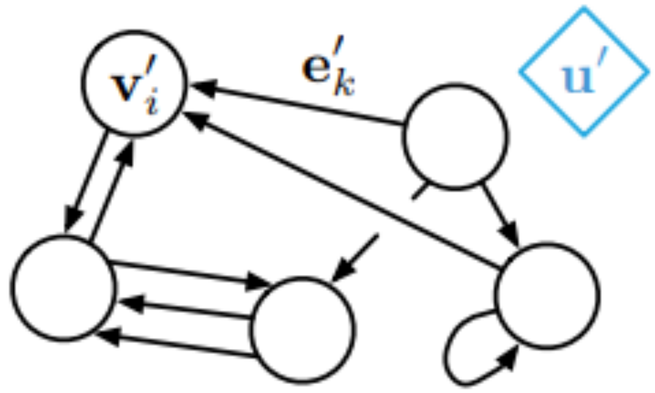
(a) Edge update

Update edges using connected nodes and globals



(b) Node update

Update nodes using (new) edges and globals



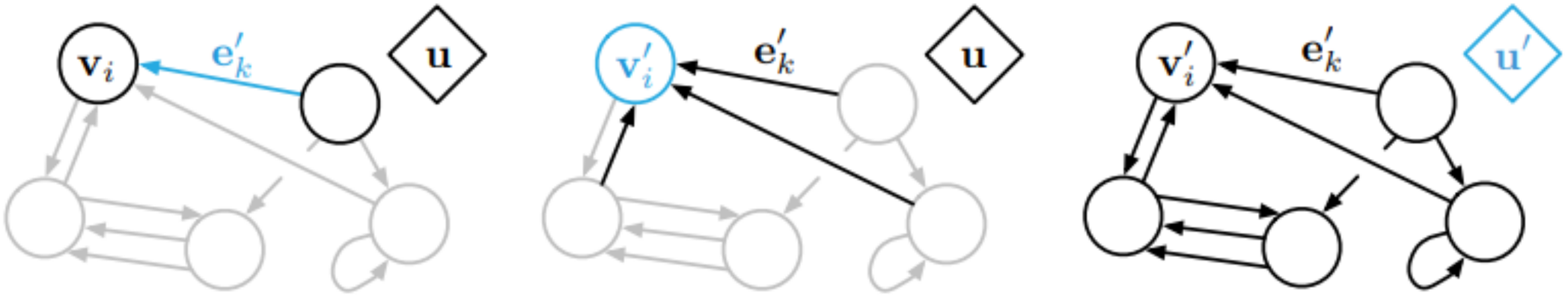
(c) Global update

Update globals using (new) nodes and (new) edges

The message passing algorithm

- Nodes, edges, and globals are updated through aggregation and MLPs

BLUE updated by BLACK (not utilizing GREY)



(a) Edge update

(b) Node update

(c) Global update

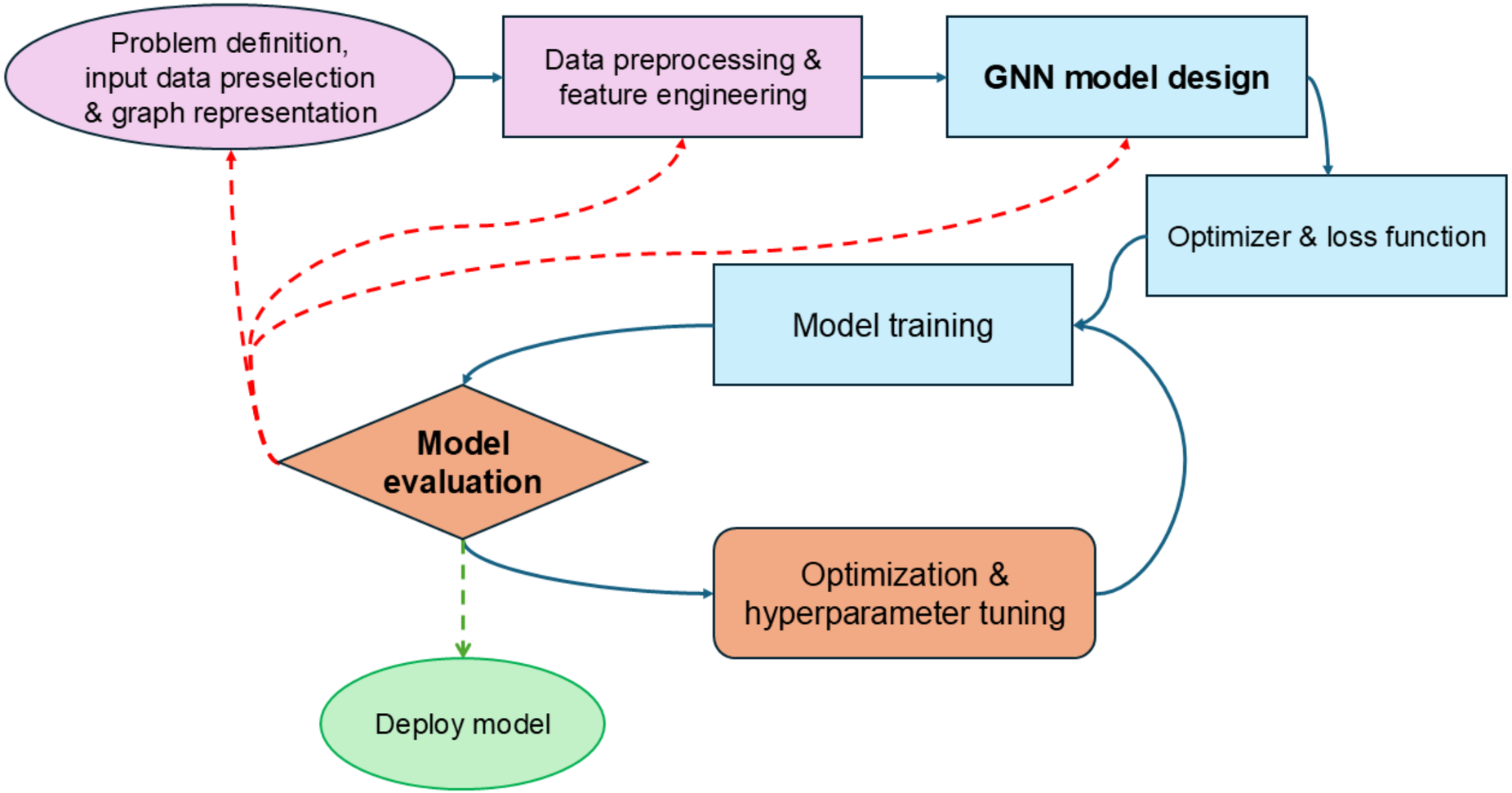
Update edges using connected nodes and globals

Update nodes using (new) edges and globals

Update globals using (new) nodes and (new) edges

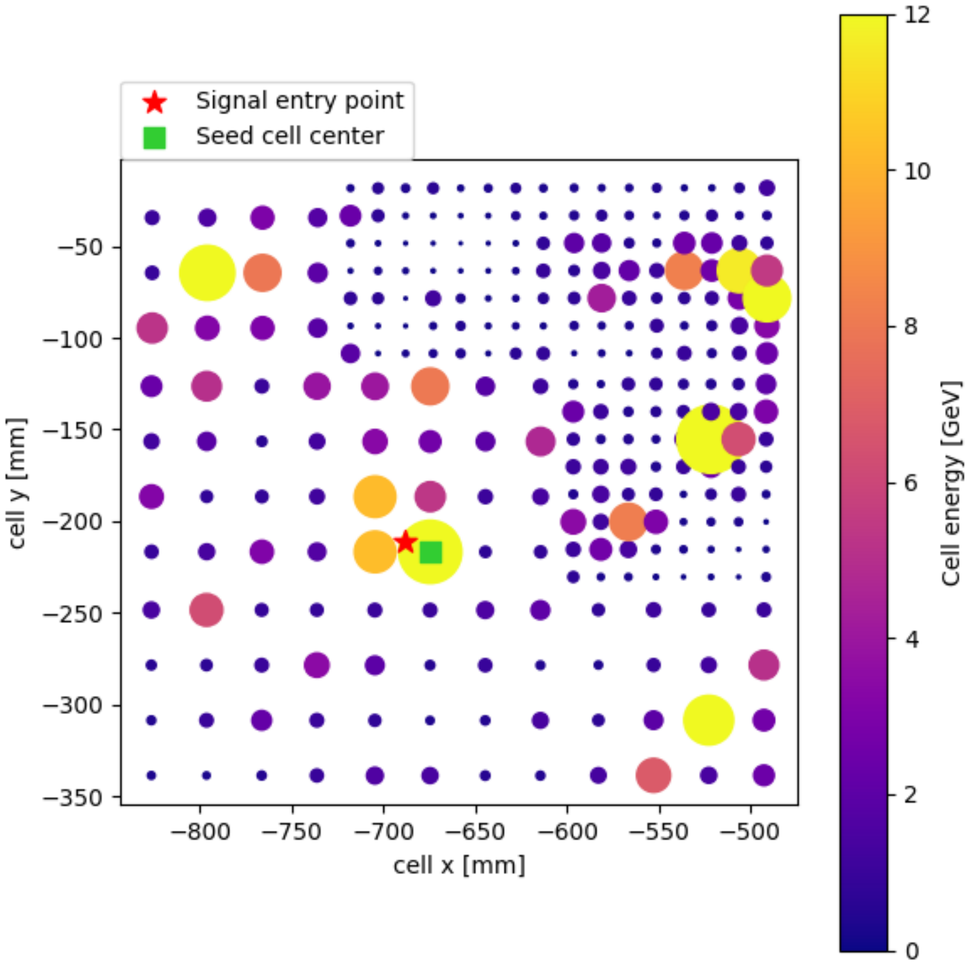
This process can be repeated many times
Each iteration can have a unique set of NNs

Workflow


Problem definition, input data preselection & graph representation
Data preprocessing & feature engineering
GNN model design
Optimizer & loss function
Model training
Model evaluation
Optimization & hyperparameter tuning
Deploy model

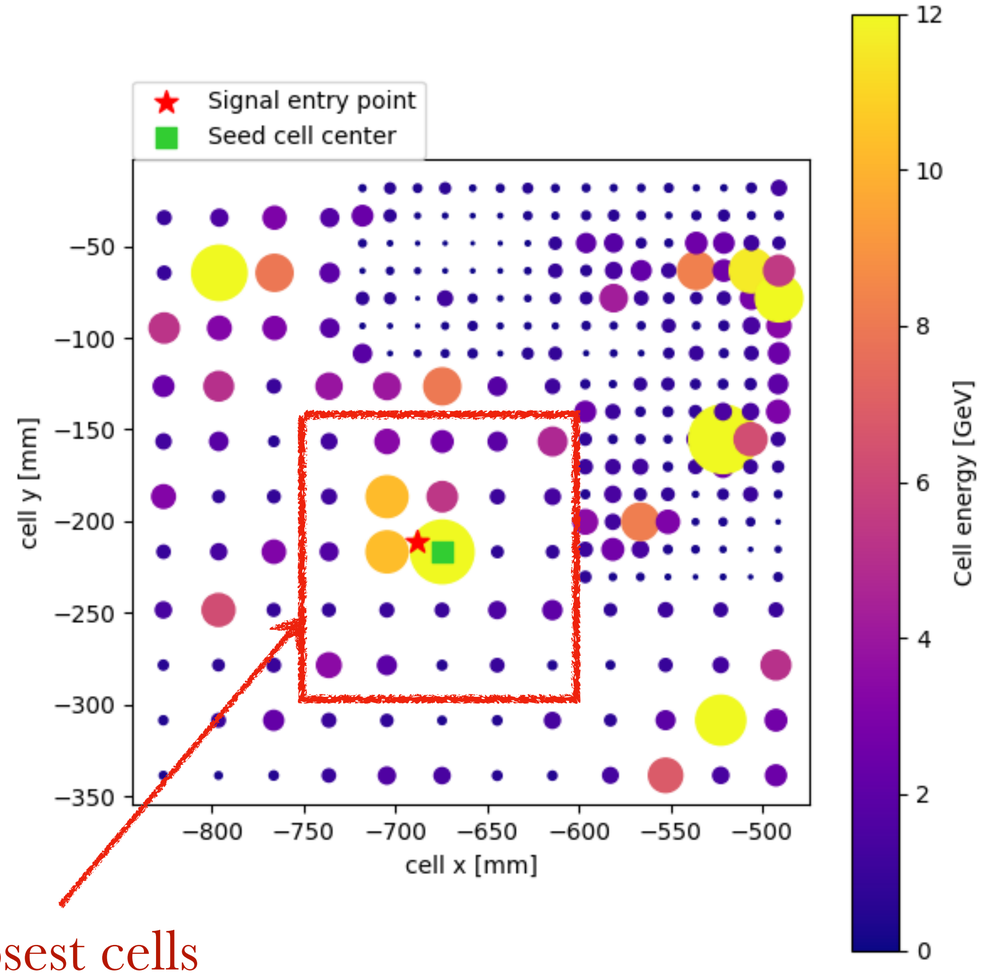
Our approach

- Training sample
  - Single photons in the particle-collision conditions expected for LHCb Run 5
  - $0.5 \leq E_T \leq 5$ GeV photons
- Preselection
  - Photons produced within 100 mm of interaction point
  - Clusters with seed in the central innermost region of the ECAL
  - 5x5 modules around the seed module, reduced to 25 central cells



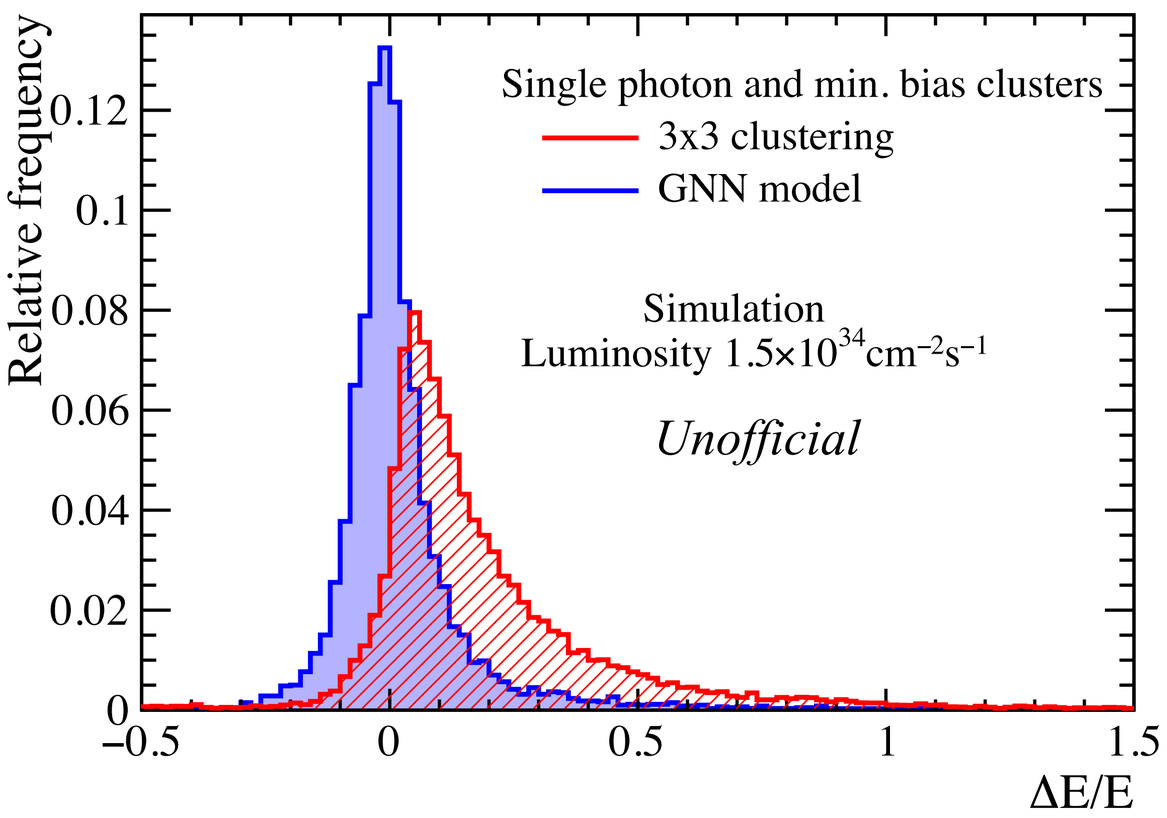
Our approach

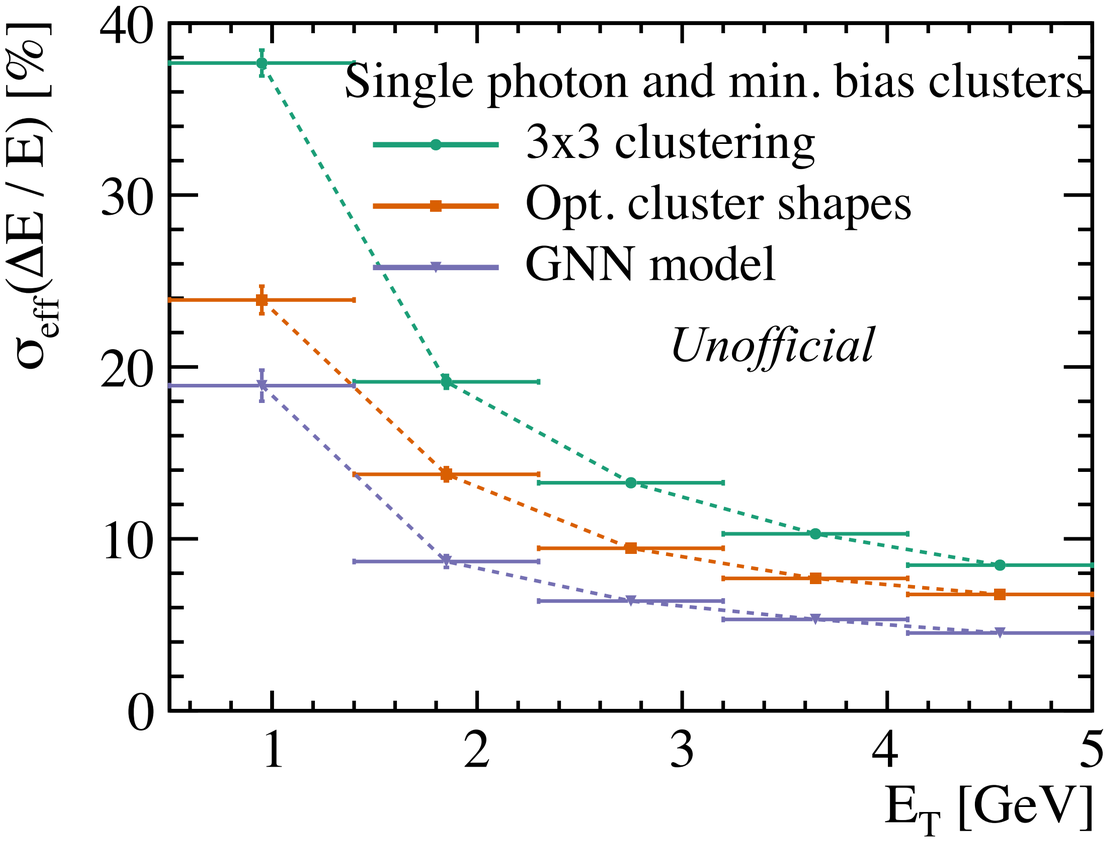
- Training sample
  - Single photons in the particle-collision conditions expected for LHCb Run 5
  - $0.5 \leq E_T \leq 5$ GeV photons
- Preselection
  - Photons produced within 100 mm of interaction point
  - Clusters with seed in the central innermost region of the ECAL
  - 5x5 modules around the seed module, reduced to 25 central cells



25 closest cells

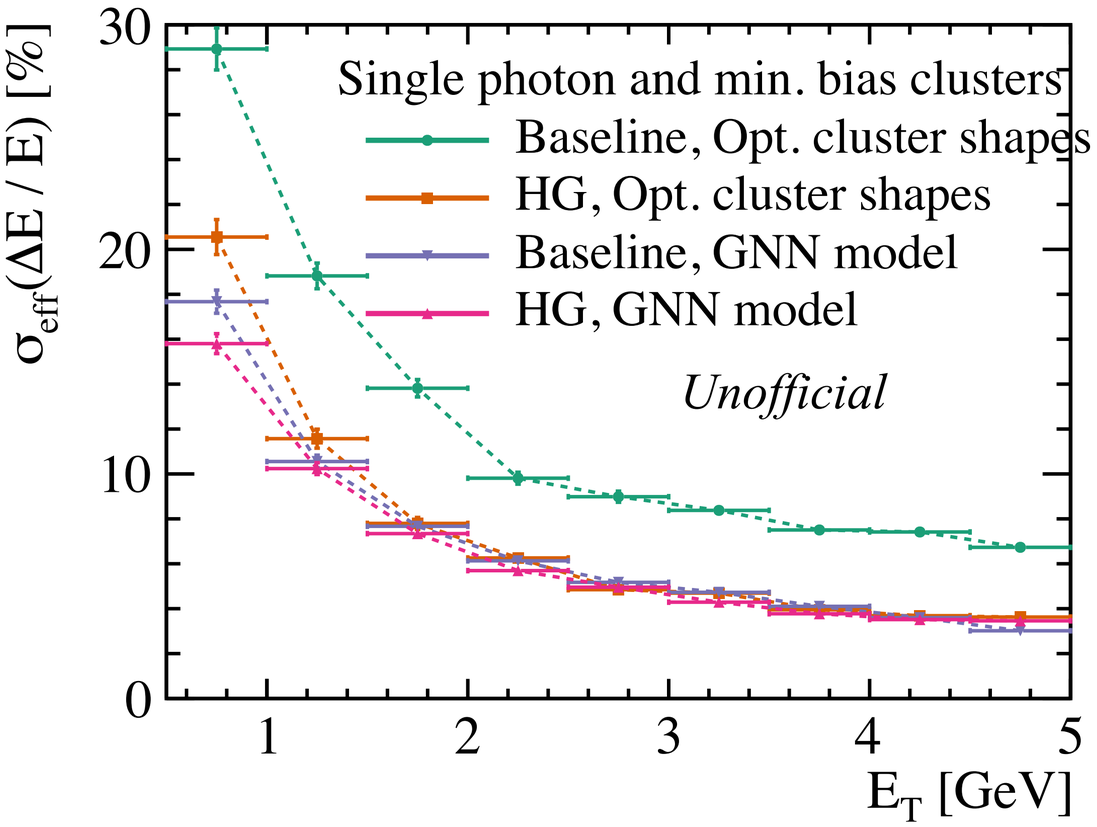
Results





The GNN outperforms the standard approaches over the full E_T range

Higher-granularity alternative

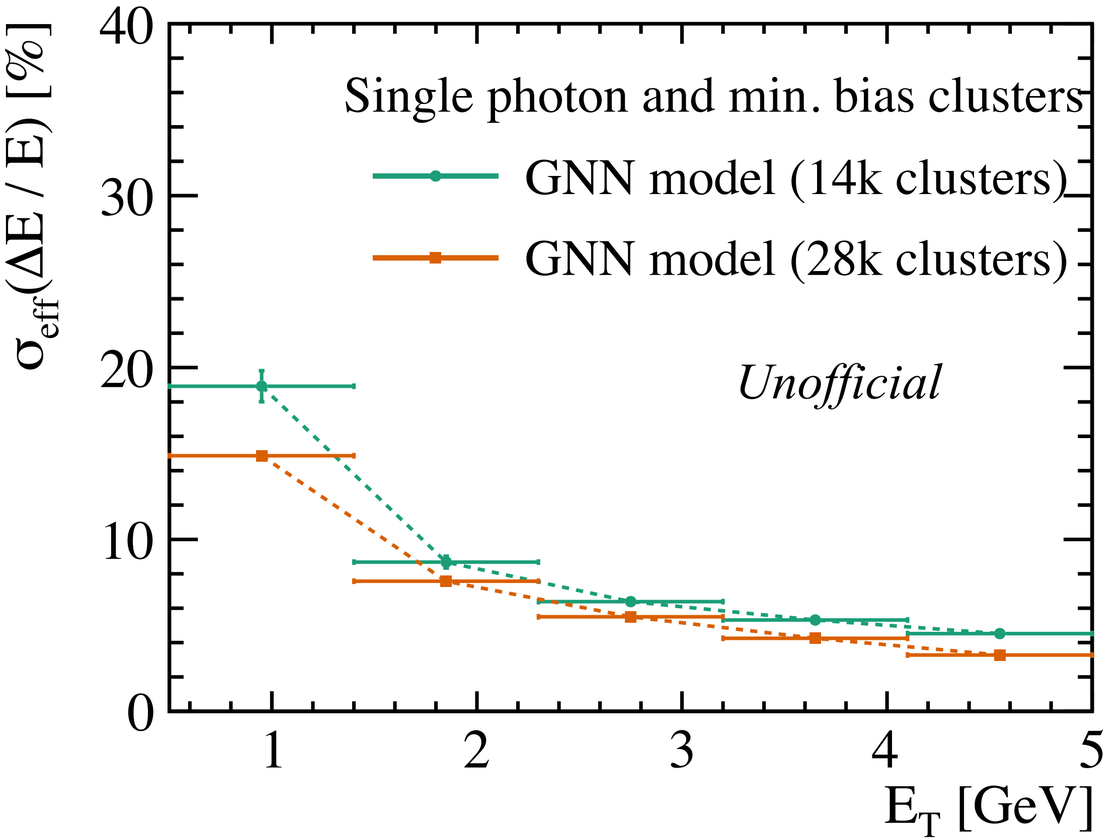


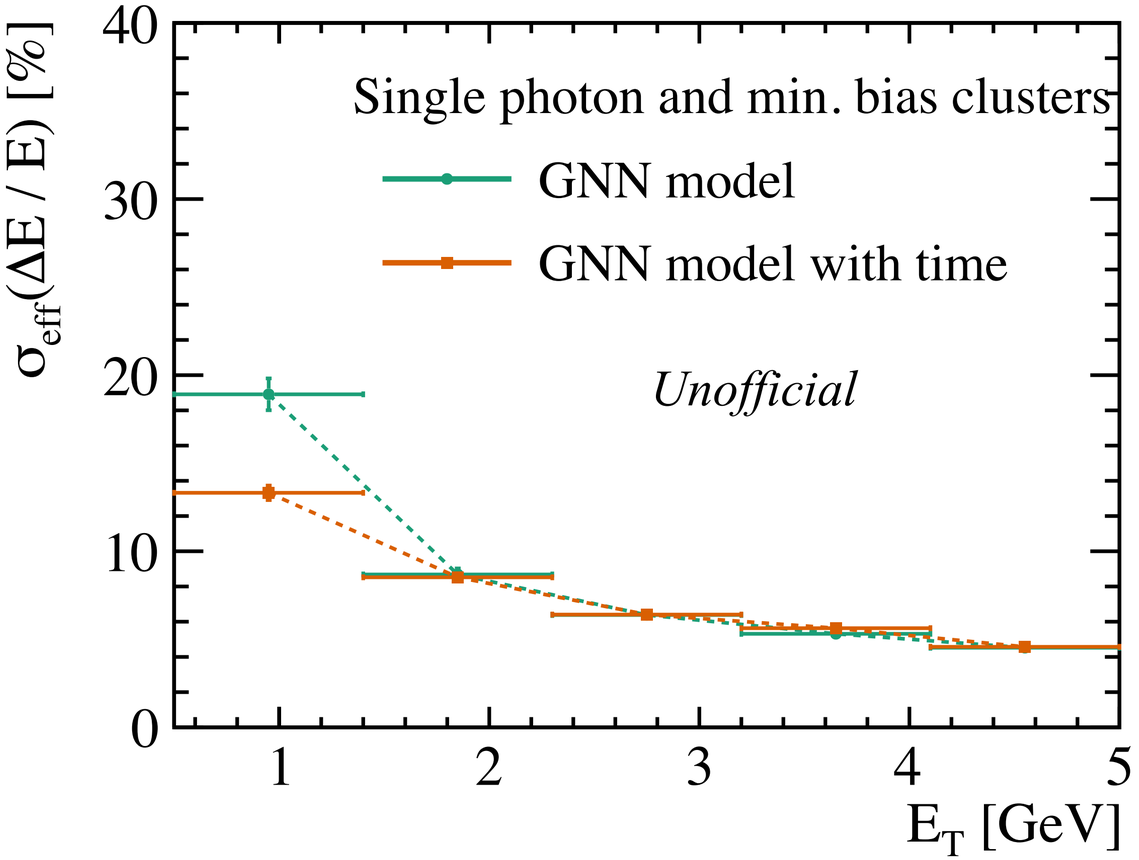
- The GNN achieves better performance than the standard approach even with lower granularity
- Not much improvement observed in the GNN with higher transverse granularity configuration
- But it further demonstrates that the GNN can handle irregular geometries
- Further improvements might be achievable with geometry-aware features

Conclusions

- We successfully developed novel GNN architectures for PicoCal reconstruction
  - Challenging yet highly promising development
  - The approach already outperforms standard reconstruction methods
  - Further improvements expected as new models, features and strategies are explored
  - Active ongoing research
- Flexible PyTorch implementation
  - Currently being tested for LHCb Runs 3&4
    - Further development required for real-time deployment — Check Uzziel's talk!

Further improvements

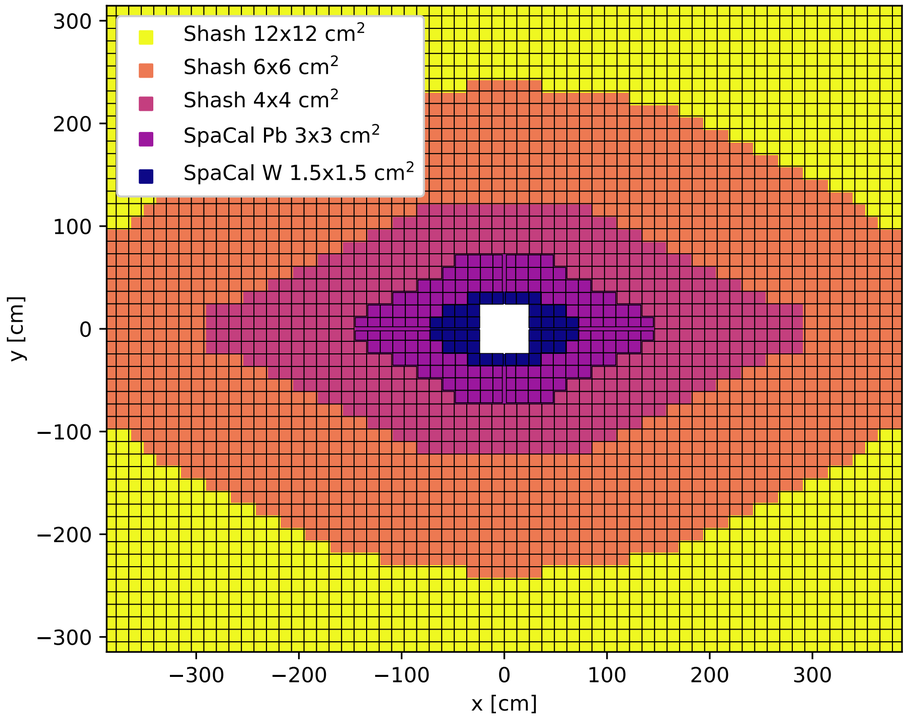




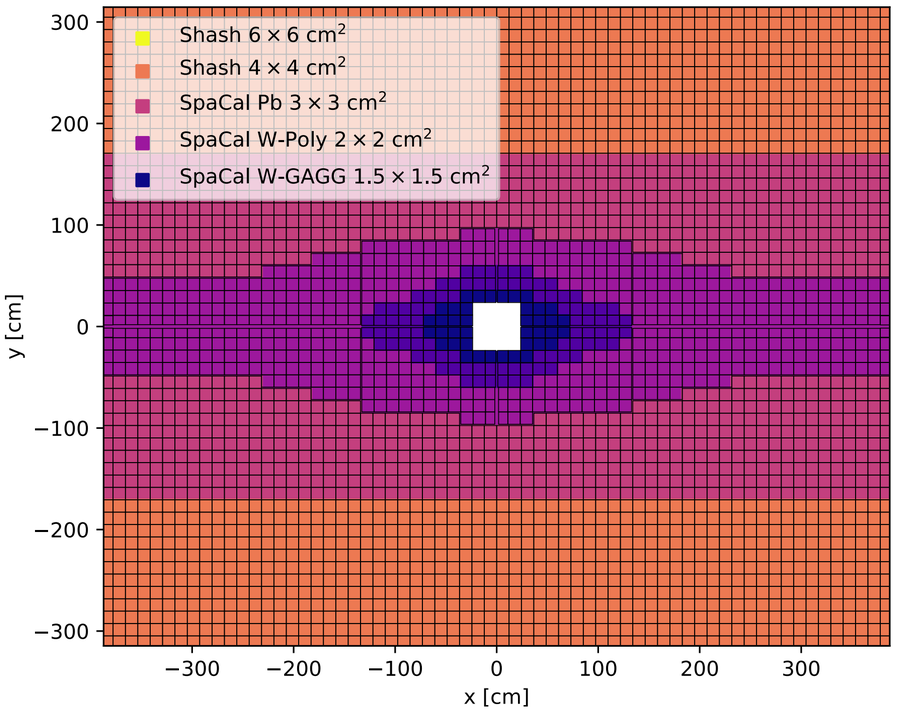
The GNN performance can still be improved with more data, and timing information can also help

High-granularity single-sided configuration

- Designed to keep # of channels ~constant
  - Reduce overal occupancy



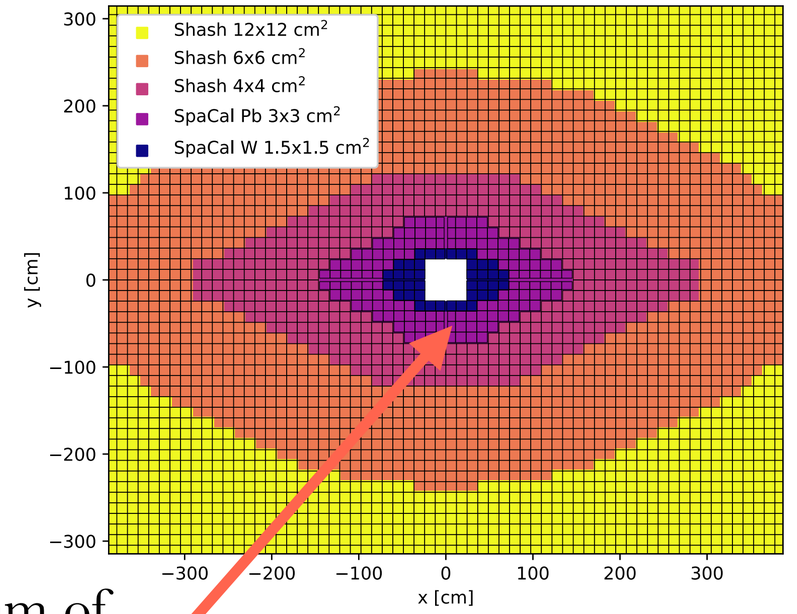
Baseline



HG

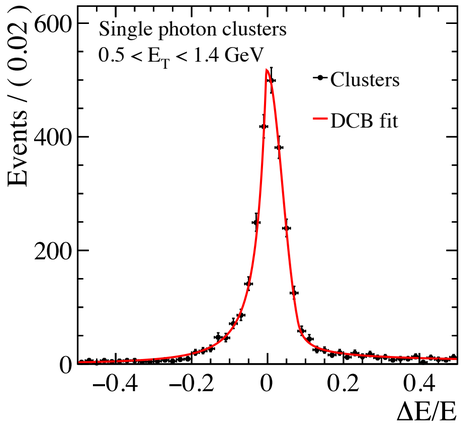
Our approach

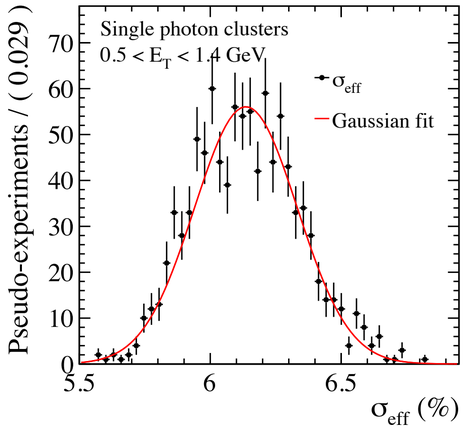
- Nodes
  - Deposited energy (front, back)
  - Cell index in module
  - Cell position (x, y)
  - Distance to seed cell
- Edges
  - ΔE, Δx, Δy, total distance
- Globals
  - Total energy
  - Seed cell position (x, y)
- Loss
  - MSE
- Target
  - True energy
- Preselection
  - Photons produced within 100 mm of interaction point
  - Clusters with seed in SpaCal-Pb region
  - 3x3 modules around the seed module, reduced to 25 central cells

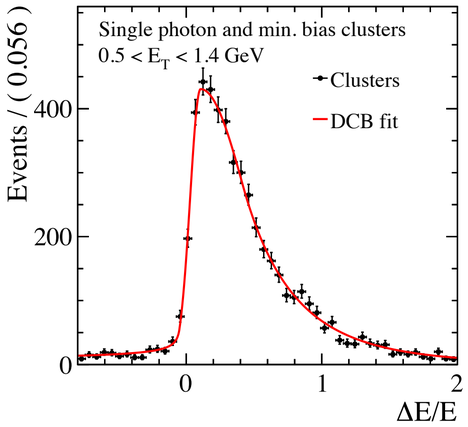


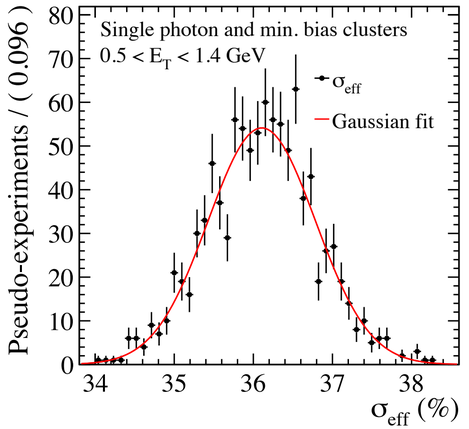
How to evaluate the performance?

- Asymmetric distributions due to pile-up: long right tails
  - Defining the ‘resolution’ is not straightforward
  - We define the resolution (σ_{eff}) as the half-width of the central 68% interval
- Fits to the $\Delta E/E$ distributions
  - Generate toys from the p.d.f., recompute metric
  - Gaussian fit to distribution of metric to extract mean and width (uncertainty)







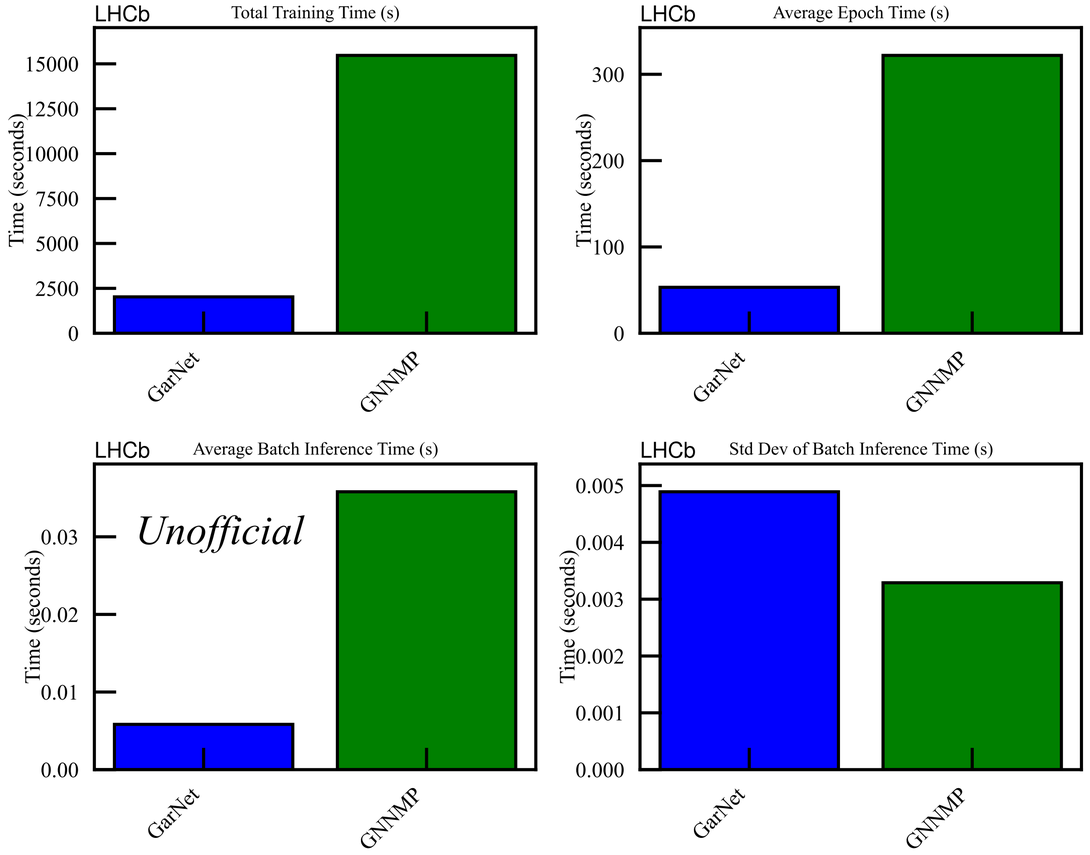


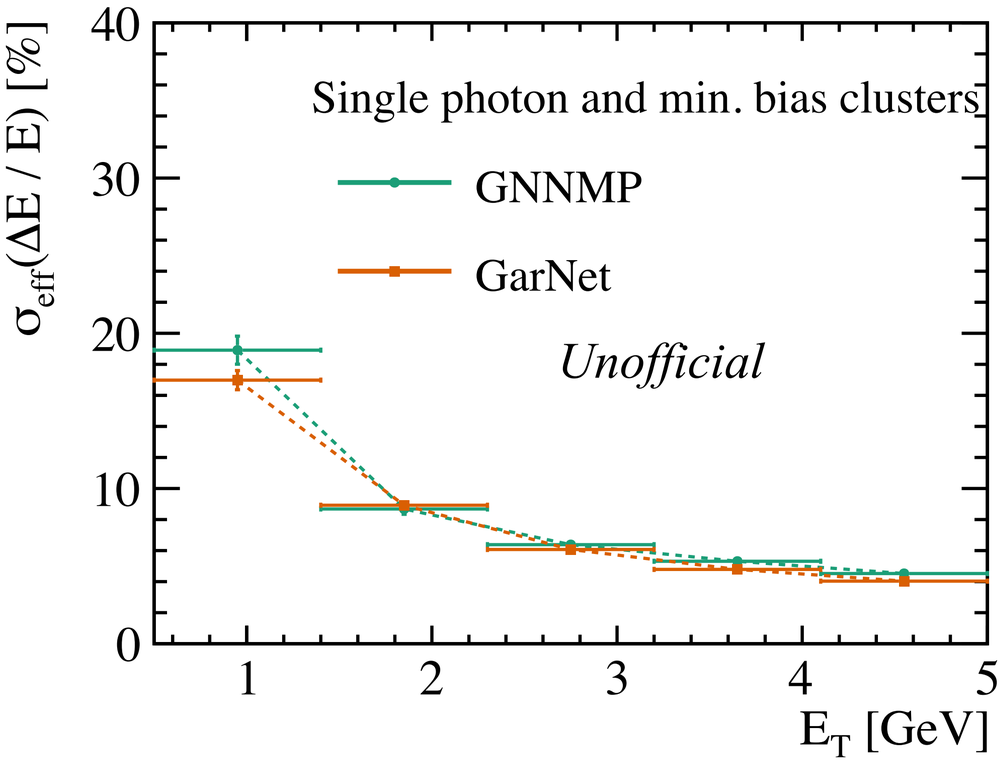
GarNet

- A lightweight, attention-enhanced variant GNN architecture
  - Introduced for real-time particle reconstruction at CMS by Iiyama et al. (Eur. Phys. J. C 79, 608 (2019); Front. Big Data 3, 598927 (2021))
  - Explicit edges are replaced by learnable aggregators connecting nodes through latent vertices
    - This reduces the number of mathematical operations, decreasing training and inference time
- GarNet has a know ONNX implementation for FPGA deployment

GarNet results

- GarNet has similar performance, but is 6-7x faster!





Ongoing studies

- Comparison for performance in and out of boundary regions
- Alternative strategies for connecting nodes in the full GNNMP
- Spatial symmetries
- Node-level targets
- Curriculum learning
- GarNet:
  - Distillation, quantization-aware training, post quantization
  - FPGA implementation, GPU parallelization

Multitarget regression

- Simultaneous regression of energy, time, and position
  - Each target contribution must be properly weighted in the loss
  - Uncertainty-aware losses can help the model dynamically balance target contributions
    - However, regularization may be needed to prevent degenerate minima and stabilize training
  - Strategies range from simple weighted MSE to multivariate Gaussian NLL with full covariance and Cholesky decomposition

Multi-target MSE (predefined weights)

$$\mathcal{L} = \sum_i w_i (y_i - \hat{y}_i(x))^2$$

learn one uncertainty per target

Homoscedastic (learned)

$$\mathcal{L} = \sum_i \left(\frac{1}{2\hat{\sigma}_i^2} (y_i - \hat{y}_i(x))^2 + \frac{1}{2} \log \hat{\sigma}_i^2 \right)$$

make uncertainty input-dependent

Heteroscedastic (diagonal)

$$\mathcal{L} = \sum_i \left(\frac{1}{2\hat{\sigma}_i^2(x)} (y_i - \hat{y}_i(x))^2 + \frac{1}{2} \log \hat{\sigma}_i^2(x) \right)$$

include target correlations

Multivariate Gaussian NLL

$$\mathcal{L} = \frac{1}{2} (\boldsymbol{y} - \hat{\boldsymbol{y}}(x))^\top \hat{\boldsymbol{\Sigma}}(x)^{-1} (\boldsymbol{y} - \hat{\boldsymbol{y}}(x)) + \frac{1}{2} \log \det \hat{\boldsymbol{\Sigma}}(x)$$