ML Hands-on session

B. Zhang and T. Lam

UNIVERSITY of HAWAIʻI® MĀNOA

Belle II

VIRGINIA TECH™

Outline

First half:

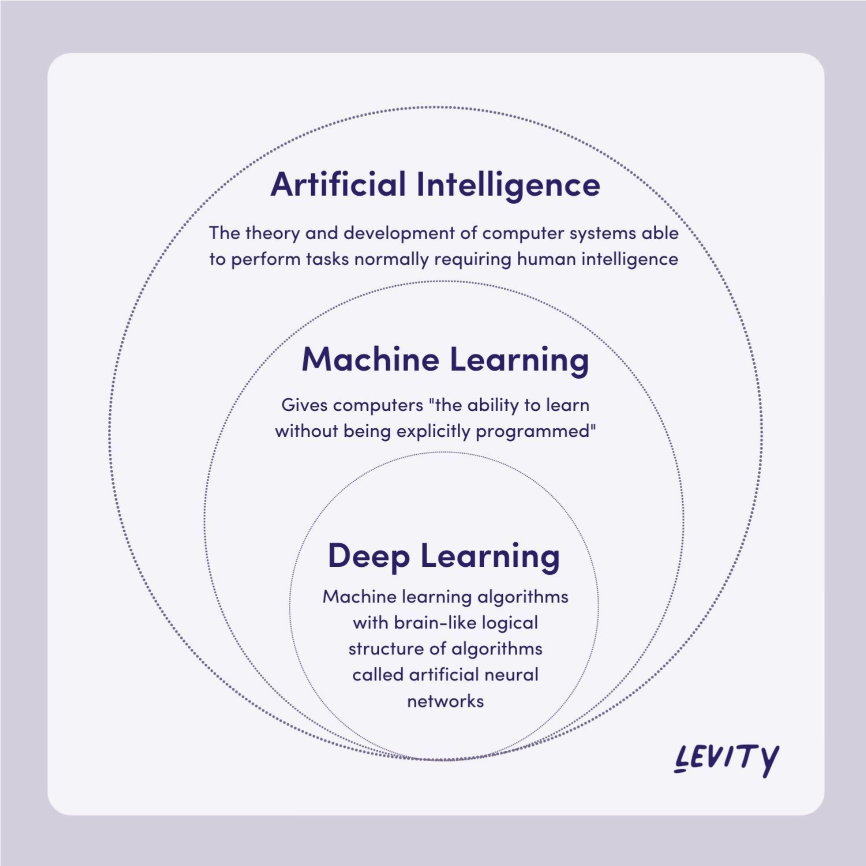
- Resources
- Introduction of Machine Learning
    - Why do we do machine learning?
    - Landscape of models
- How to do Machine Learning?
- Model Example 1: Neural Networks

Second Half:

- Model Example 2: Decision Trees
- PID recap at Belle II
- Auto machine learning
- Hands-on exercise

What is Machine Learning?

- “Machine learning is a form of AI that enables a system to learn from data rather than through explicit programming.”



PC:https://levity.ai/blog/difference-machine-learning-deep-learning

Types of Machine Learning Problems (w/ ex.)

Supervised Learning

Classification

-Particle identification
-Distinguishing between signal vs. background

Regression

-Approximating Functions

Semi-Supervised Learning

Image Generation

-PXD background overlay generation
-ECL Waveform generations

Anomaly Detection

-Computer Vision for DQM at HYDRA

Unsupervised Learning

Clustering

-Improved ECL Cluster Search
-GNN Tracking at Belle 2

Anomaly Detection

-NP Detection for Dark Higgs at Belle 2

Reinforcement Learning

Language Models

-ChatGPT

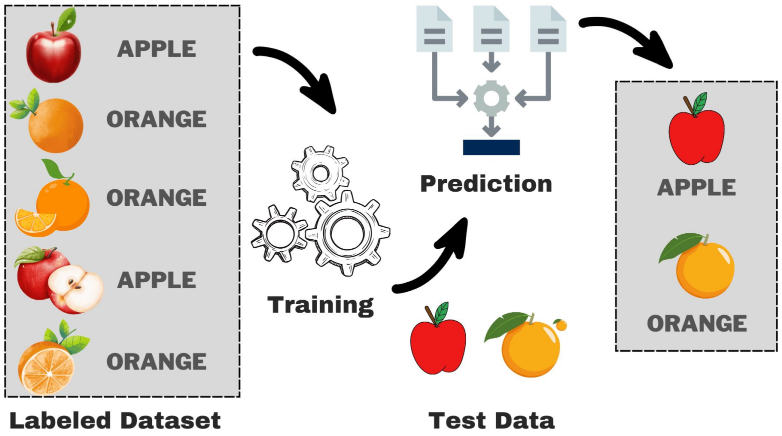
Time-Dependent

-Self Driving Cars

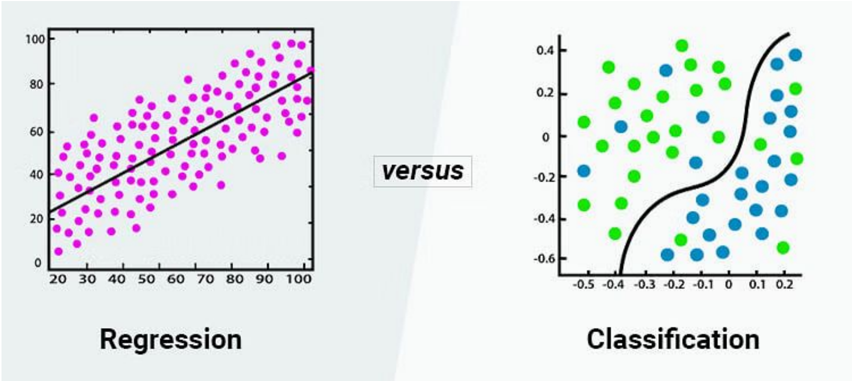
For better view of the landscape: https://iml-wg.github.io/HEPML-LivingReview/

Supervised Learning

- Given a dataset with a set of labels/indices, train your machine to get those labels right.
    - If labels are discrete, then we have classification
    - If labels are continuous, then we have regression
- If not all your dataset is labeled or you don't have labels, then you are semi-supervised or unsupervised



PC:https://www.kdnuggets.com/wp-content/uploads/mehreen_understanding_supervised_learning_theory_overview_6.png



PC:https://www.simplilearn.com/ice9/free_resources_article_thumb/Regression_vs_Classification.jpg

How to do machine learning?

With a focus on supervised learning

0 **Data Collection & Processing**

Get a dataset and process the data to prepare for training

1 **Model Selection**

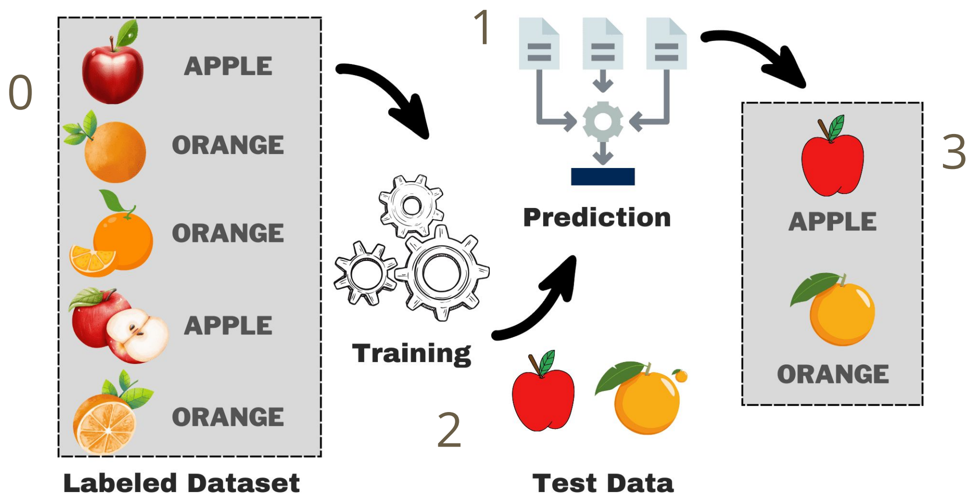
Choose the best set of algorithms that best suits your problem of interest

2 **Training, Validation, & Testing**

Given a dataset, tune your model to best solve the problem of interest.

3 **Deployment**

With your model trained and tested, all that's left to do is to apply it to new data with the trained parameters

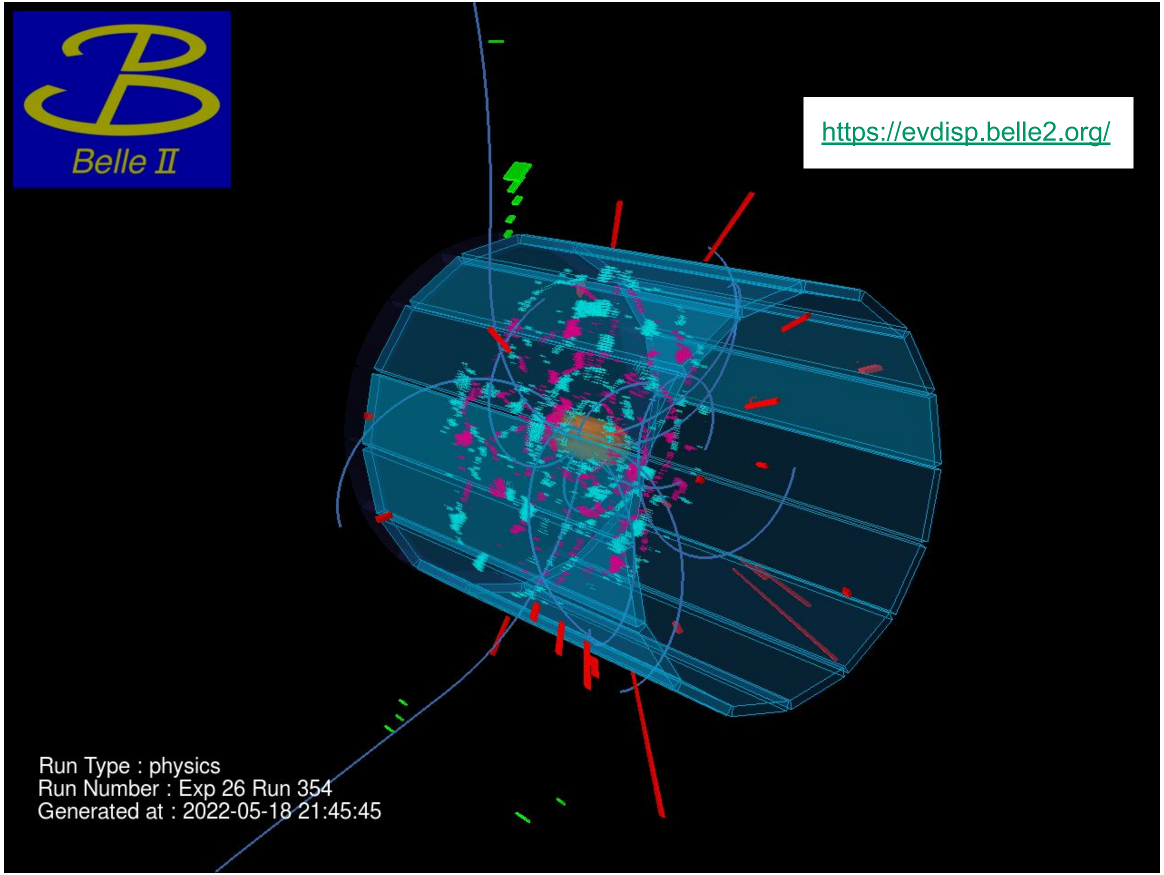


Data Collection

For Belle II, data taking is the most important task!

In general, the more data, the better.

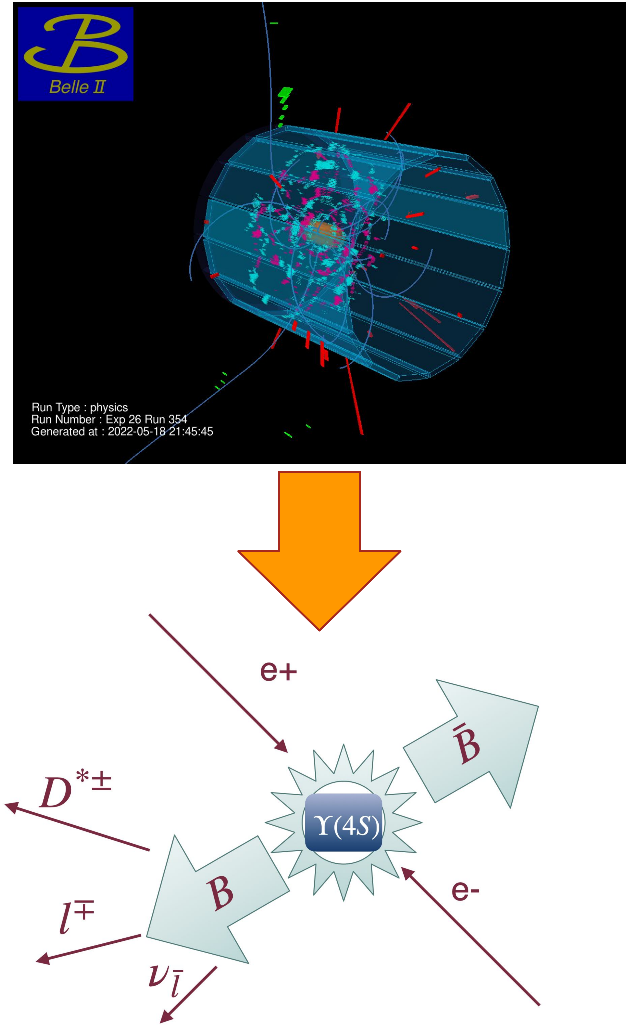
Take shifts to help in with this part!



Data Processing

Basf2 software converts raw data to analysis-level variables as part of processing.

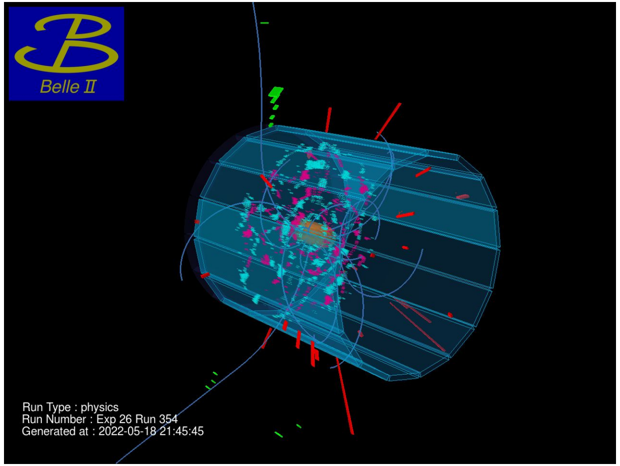
These variables are used as “**features**” or inputs into our machine learning model.

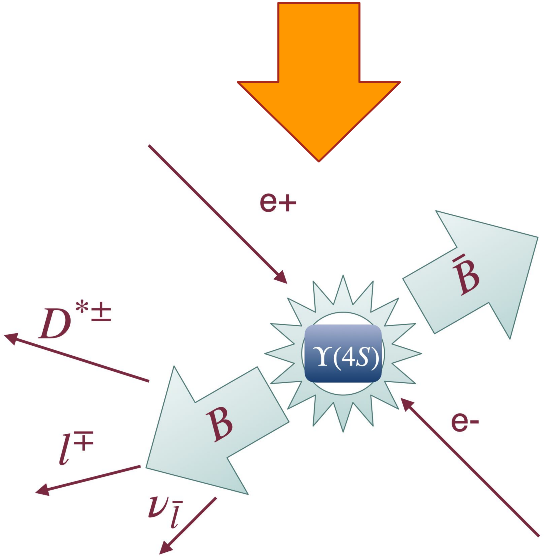


Data Processing

In general, data processing should also consider the following:

- Feature selection:
    - Choose variables with high discriminating power
    - Remove irrelevant variables
        - Little discriminating power
        - Highly correlated with other features
- Data Cleaning:
    - Missing Values/Bad Formatting/Type Conversion
    - Duplicates in dataset
    - Outlier detection (bad reconstructed signal?)
    - Verifying/cross-checking labels



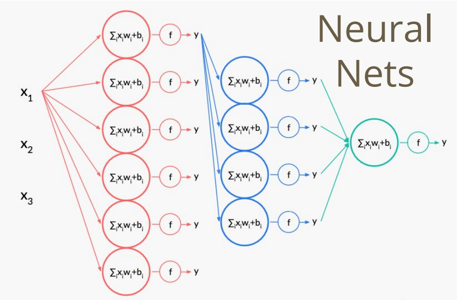


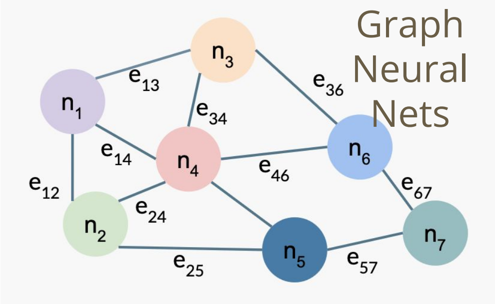
Model Selection

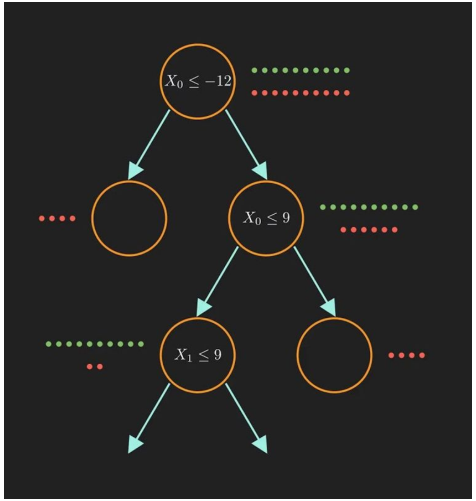
After preparing your datasets, choose the model/algorithm that is best suited for your task.

Things to consider:

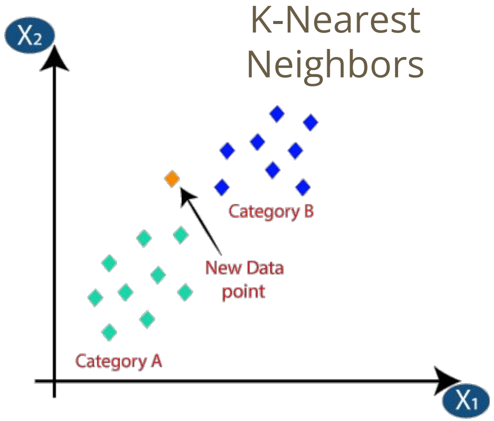
- Availability of resources (time)
- Strengths and weaknesses?
- Evaluation Metrics







Decision Trees



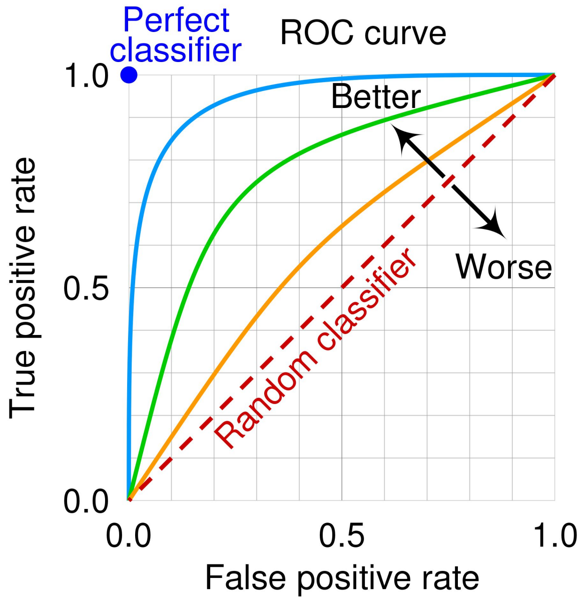
https://machinelearningmastery.com/applied-machine-learning-is-hard/

Training, Validation, & Testing

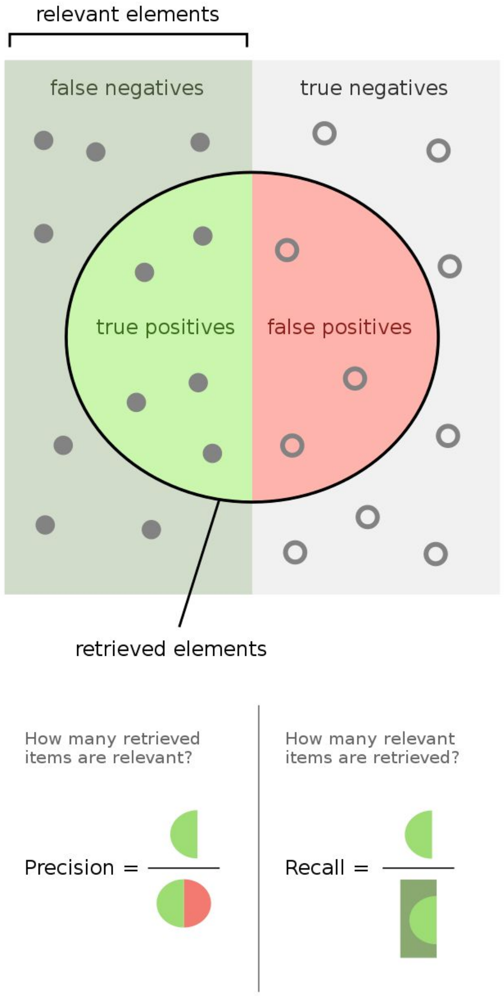
- For a given dataset with a set of features, one typically partitions their dataset into:
  - **Training**: The bulk of your overall dataset should be here to make sure your model learns as many trends as possible.
  - **Validation**: An independent subset used to check for overfitting during training (and hyper-parameter tuning)
  - **Testing**: Another independent subset used for overall comparisons.
- A figure of merit or evaluation metric is important and is dependent on your model and problem.

Performance Evaluation Example

Receiver Operating Characteristic (**ROC**) curves are used to evaluate classification models.



PC: https://en.wikipedia.org/wiki/Receiver_operating_characteristic



Quick Break For Questions

Understanding models

Part 1: Neural Networks

0 Data Collection & Processing

Get a dataset and process the data to prepare for training

1 Model Selection

Choose the best set of algorithms that best suits your problem of interest

2 Training, Validation, & Testing

Given a dataset, tune your model to best solve the problem of interest.

3 Deployment

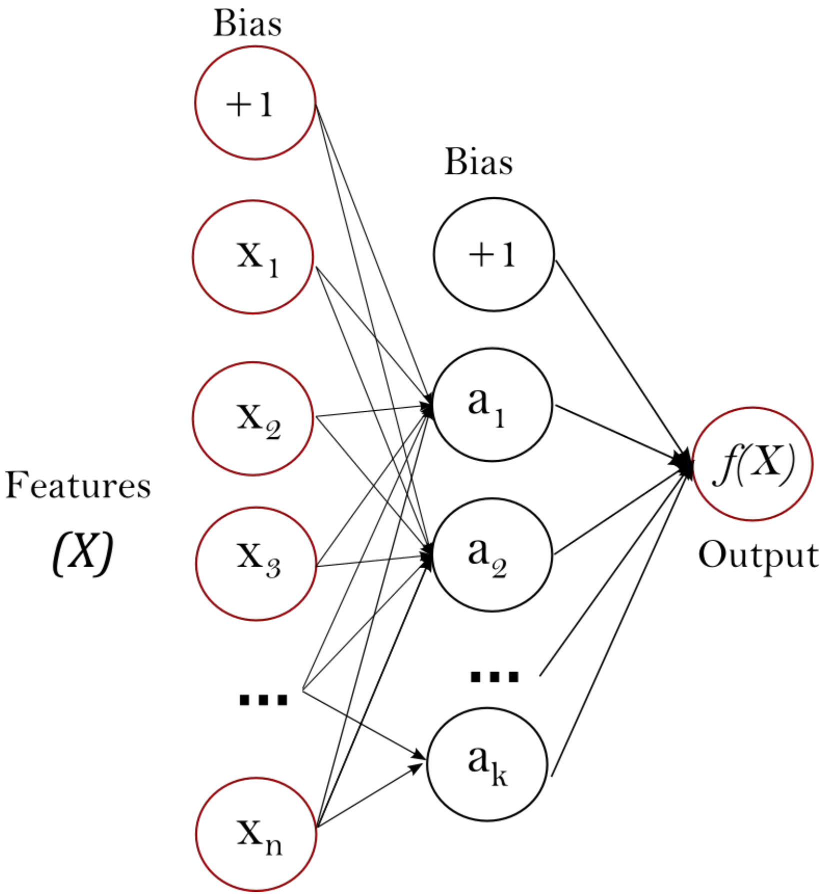
With your model trained and tested, all that's left to do is to apply it to new data with the trained parameters

What is a neural network?

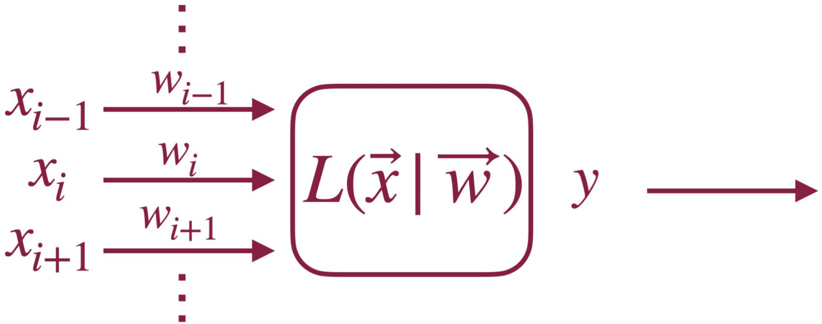
Inspired by structure and function of brains, neural networks (NNs) consists of neurons that process some input and outputs a signal.

Strength of NNs come from their ability to be 'universal approximators'

(https://doi.org/10.1016/0893-6080(89)90020-8)



What's a node/neuron?



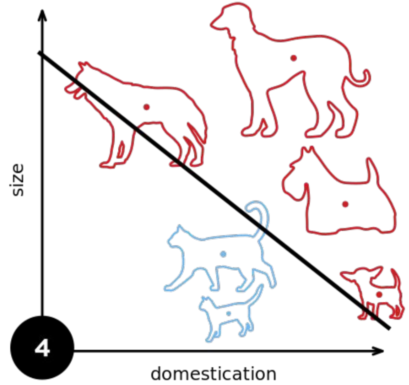
For input features x_i ($i \in [0,N]$) and outputs y, we have some activation function L(x).

Features are scaled/weighted before fed into L(x)

$$L(\vec{x}\,|\,\vec{w}) \equiv H(\vec{x}\cdot\vec{w}) = \begin{cases} 0 \text{ for } \vec{x}\cdot\vec{w} < 0 \\ 1 \text{ for } \vec{x}\cdot\vec{w} > 0 \end{cases}$$

Simplest example of an activation function is the step function H(x).

Note: x·w=0 forms a line like seen on the right!



How to neural networks learn?

Components for NN learning:

Performance Metric:

-Figuring out what to optimize on

-Ex. Mean Squared Error ($\mathscr{L}^2$)

Update Rule:

-Procedure for your model to learn

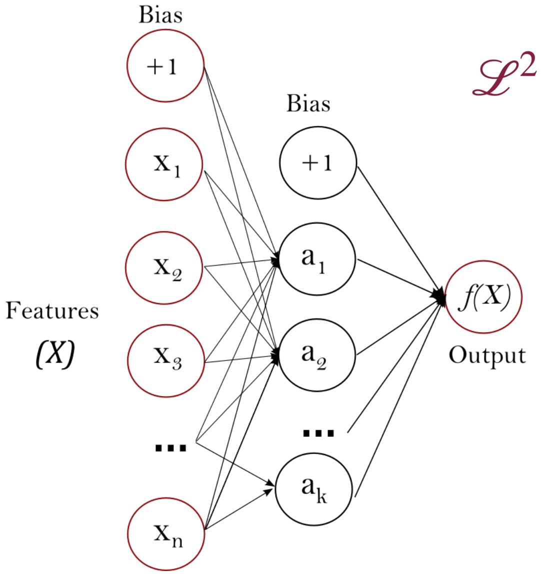
-Ex. Gradient Descent with learning rate η over weights from node n

$$\mathscr{L}^2 = \frac{1}{D}\sum_{d=1}^{D}(Y_d - f(\vec{x}_d))^2$$

$$\vec{w}_n \leftarrow \vec{w}_n - \eta\left(\partial_{\vec{w}_n}\mathscr{L}^2\right)$$

Forward Pass

With one's current NN configuration, go through your dataset and determine f(x) for the d'th data point



Calculate Loss

Determine how close you are to the correct labels (or accuracy)

$$\mathcal{L}^2 = \frac{1}{D}\sum_{d=1}^{D}(Y_d - f(\vec{x}_d))^2$$

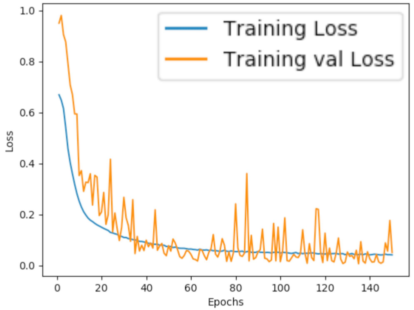
Backpropagate

Depending on the weights, update each node's weights. Usually done via a gradient descent.

$$\vec{w}_n \leftarrow \vec{w}_n - \eta\left(\partial_{\vec{w}_n}\mathcal{L}^2\right)$$

Learn!

Update your model's weights. Rinse and repeat until your model converges (loss is low after some number of epochs)



Quick Break For Questions

Understanding models

Part 2: Decision Trees

(classify singal/bkg by making cuts)

0 **Data Collection & Processing**

Get a dataset and process the data to prepare for training

1 **Model Selection**

Choose the best set of algorithms that best suits your problem of interest

2 **Training, Validation, & Testing**

Given a dataset, tune your model to best solve the problem of interest.

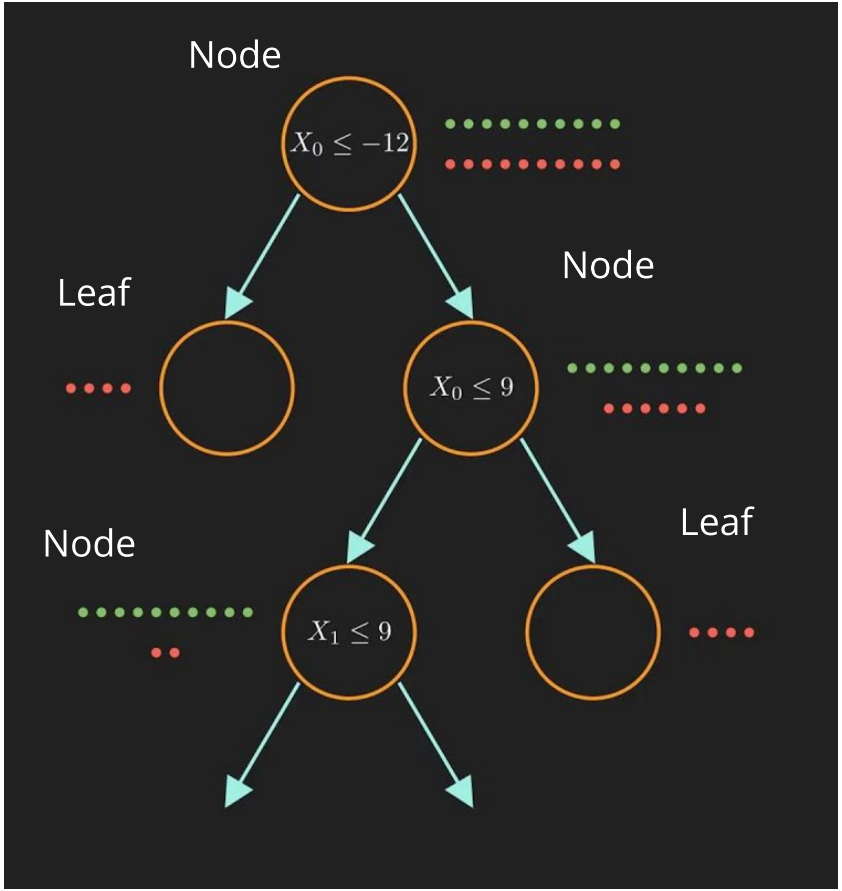
3 **Deployment**

With your model trained and tested, all that's left to do is to apply it to new data with the trained parameters

What is a decision tree?

- A decision tree is a flowchart-like model
- Each node partition the dataset based on one feature.
- Each leaf represents the outcome of the partition.

Example next page



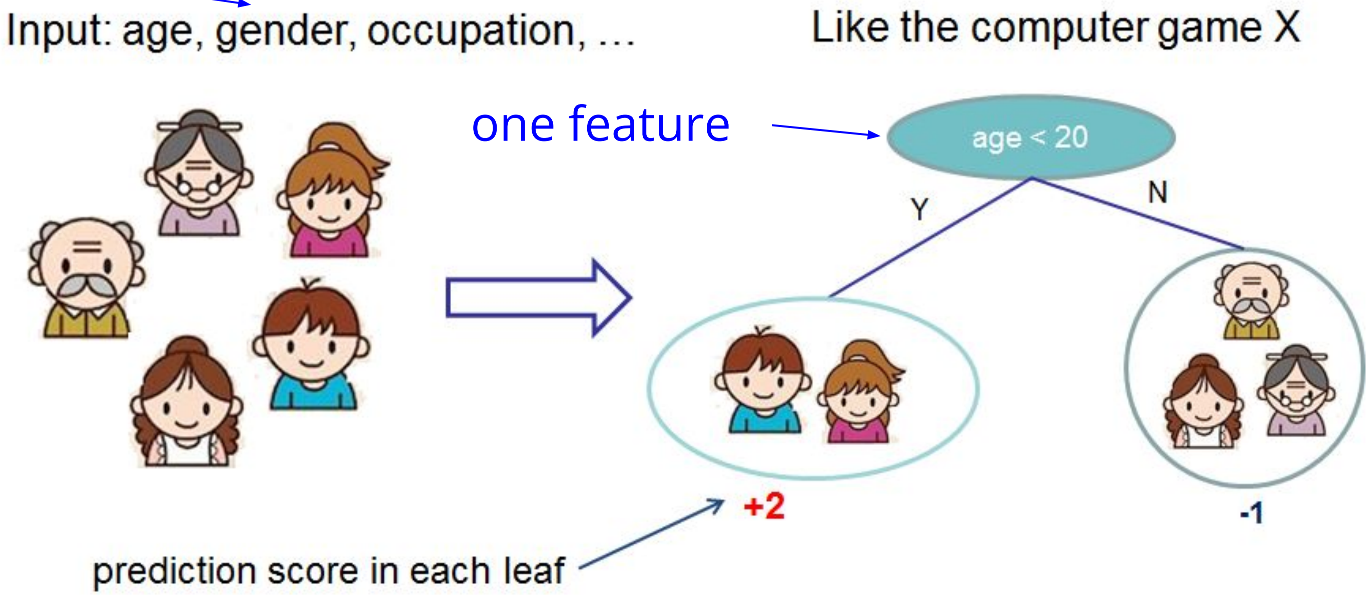
Example classification: Like vs. Dislike computer game

Input features

Decision Tree Model

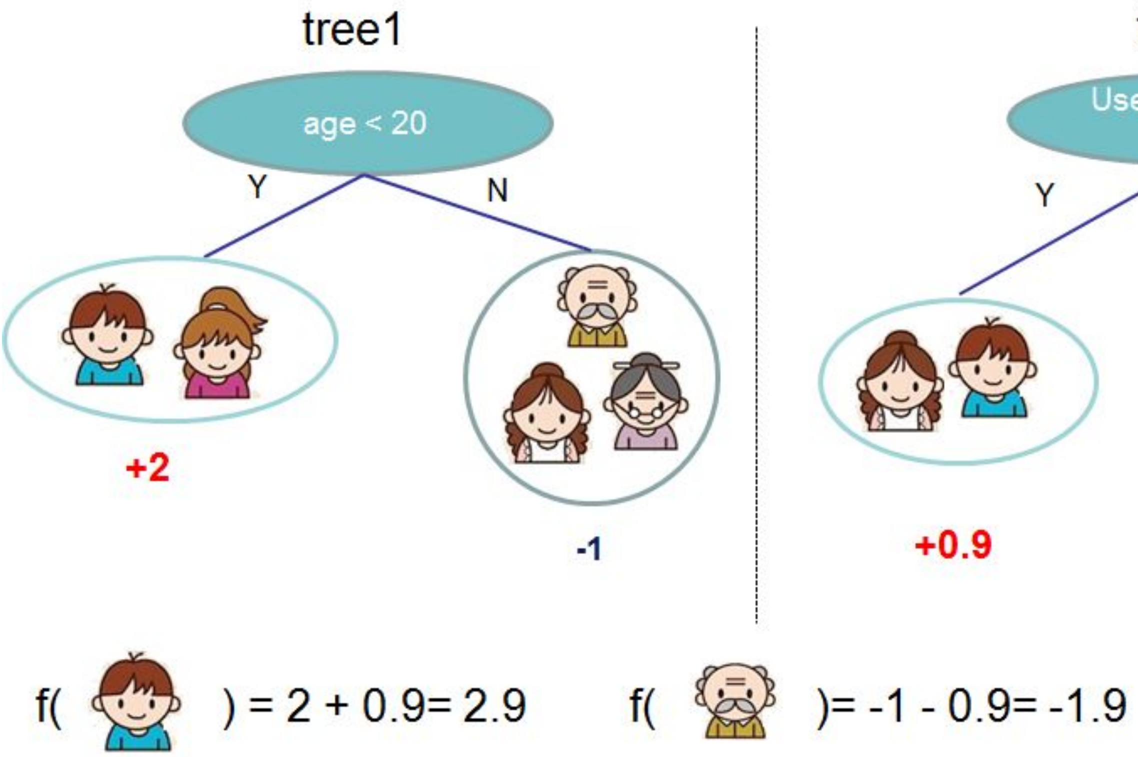
one feature

Input data



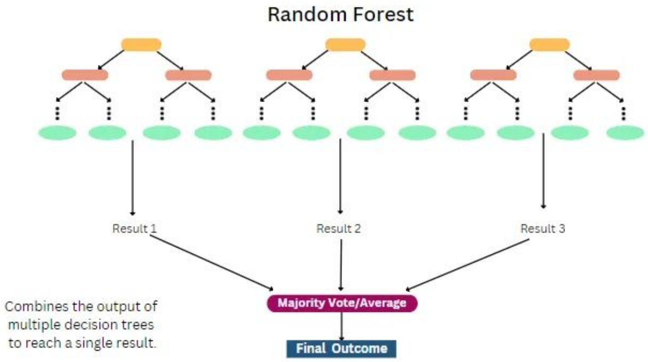
Credit: [Gradient Boost Intro]

Decision Tree Ensembles

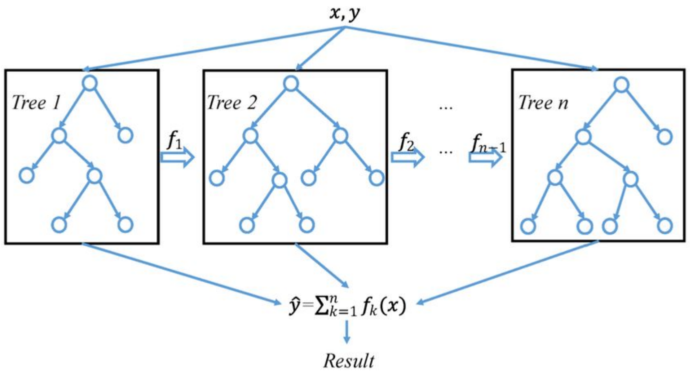


Split nodes **Grow trees** **Combine trees** **Learn!**

Random Forest

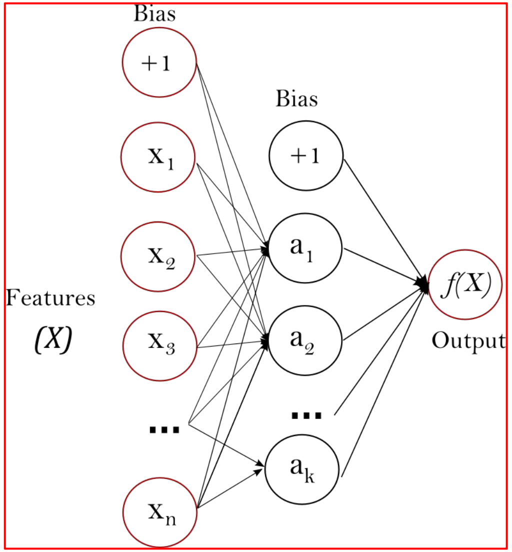


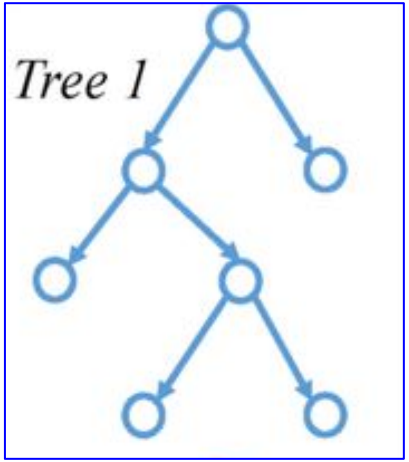
Boosted Decision Trees



Neural Net vs. Decision Tree

- NN can use all input features in a single neuron, DT uses only one feature per node (DT is simpler)
- NN can have deep layers, DT prefers shallow layers with few splits (DT is simpler)
- DT is a weak learner → used in ensembles
  Multiple trees are trained
- Parameters of a NN are updated continuously during training. Once trained, the parameters of a tree are fixed.
- NN have a fixed # of neurons, but DT can grow into a forest





Particle Identification Recap

0 CDC and dE/dx

1 ARICH/TOP and Cherenkov rad

2 ECL and energy deposition

3 KLM and penetrating length

Particle Identification at Belle II:

- Necessary to distinguish 'stable' / final state particles to the detector.(e, μ, π, K, p, d)

- Vital for making **precise measurements** or validating **new physics** models.

- This is achieved through a combination of sophisticated **sub-detector** and **software** algorithms.

- We have used PID in the basf2 hands-on exercise (B -> D* l ν).

General idea for Belle II PID:

- In each sub-detector **d** ∈ D = {CDC, TOP, ARICH, ECL, KLM}, a likelihood $\mathbf{L^d(x|i)}$ is defined for each charged particle hypothesis **i** as a joint probability density function (PDF) of a given set of observables, **x**.

- Assuming sub-detectors' measurements of **x** are independent, a global likelihood for each particle hypothesis **i** is defined by

$$\mathcal{L}(\mathbf{x}|i) = \prod_{d}^{d \in D} \mathcal{L}^d(\mathbf{x}|i) \quad \text{or equivalently,} \quad \mathcal{L}(\mathbf{x}|i) = \exp\left(\sum_{d}^{d \in D} \log \mathcal{L}^d(\mathbf{x}|i)\right)$$

General idea for Belle II PID:

- The ratios of the global likelihood serves as a ‘probability’ for identifying candidates against all other hypotheses, using Bayes’ theorem and the law of total probability:

$$P(A_i|\mathbf{x}) = \frac{P(\mathbf{x}|A_i) \cdot P(A_i)}{\sum_j P(\mathbf{x}|A_j)P(A_j)} \quad \Rightarrow P(i|\mathbf{x}) = \frac{\mathcal{L}_i}{\sum_j \mathcal{L}_j}$$

Credit: [Belle II Note]

What is the problem exactly:

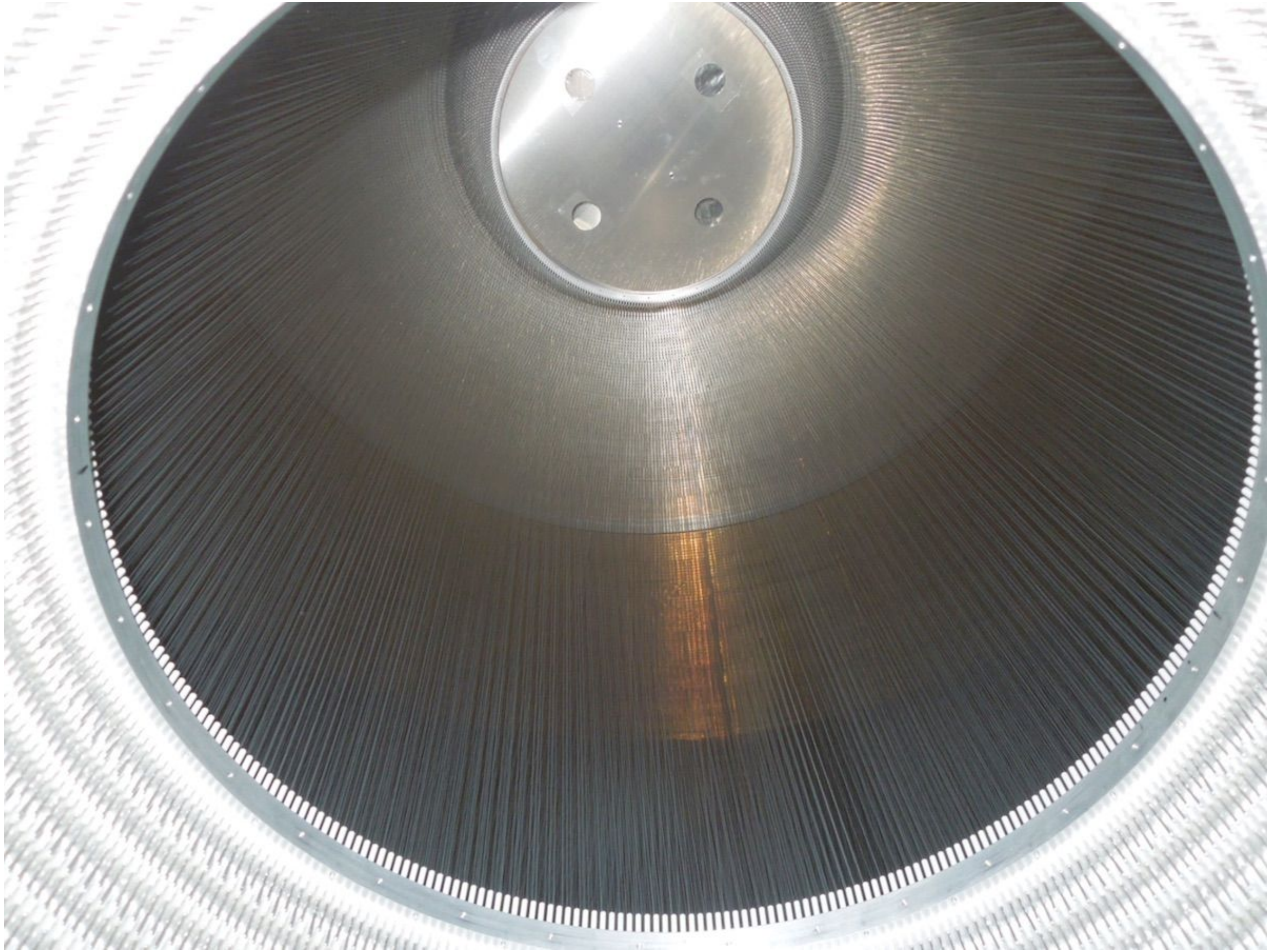
- A track is detected at Belle II, we can measure its **momentum p** thank to the magnetic field, but we don’t know the species, i.e. **mass M**

- **p = M v**

- A simple method is to measure the velocity and then determine the mass
    - Ionization energy loss dE/dx – Bethe Bloch formula → velocity
    - Cherenkov radiation θ_c angle, # of photons → velocity

CDC and dE/dx:

- Charged FSP ionizes the gas in CDC along the trajectory.
- The number of ionized electrons (dE) are collected and measured at each wire segment (dx), which provides dE/dx.
- -> velocity



CDC and dE/dx: (more relevant p<1 GeV)

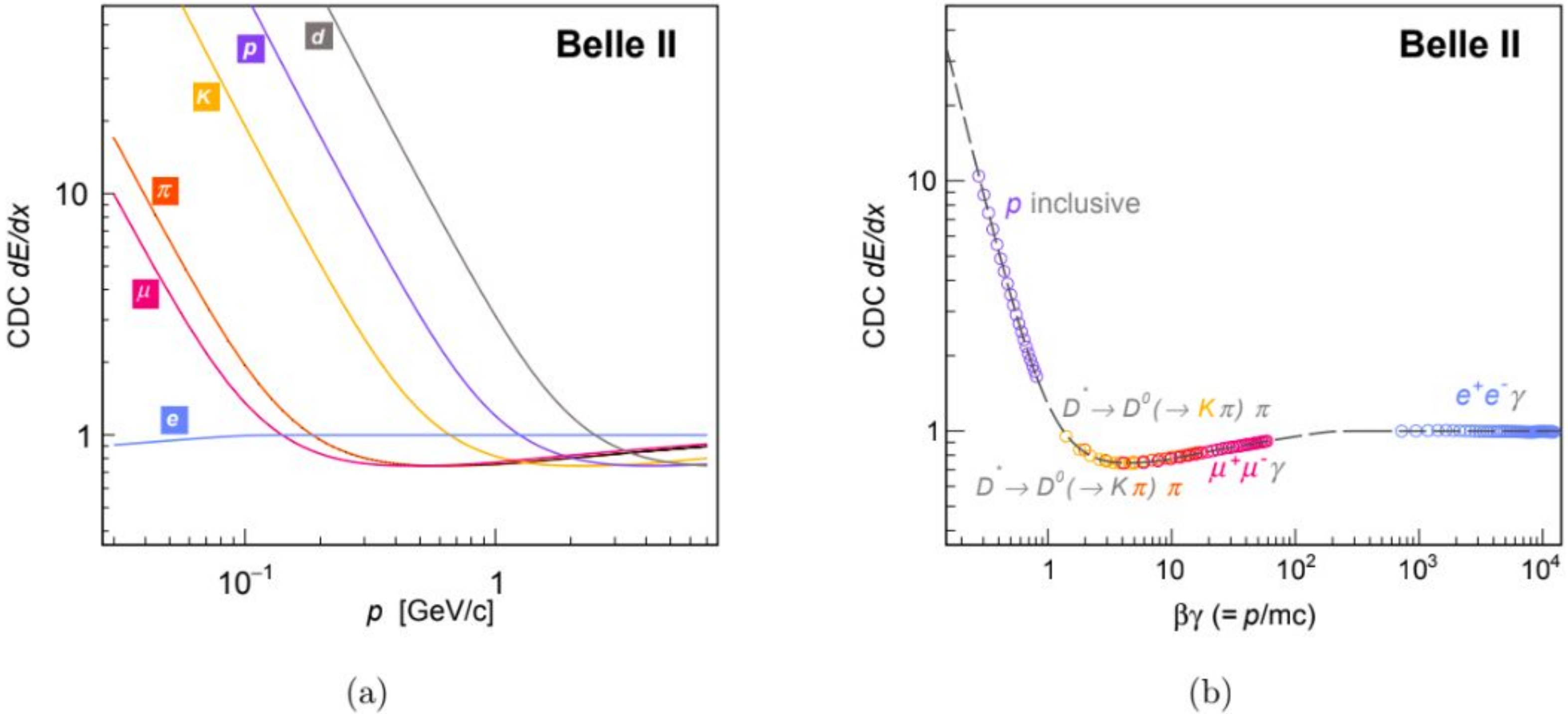
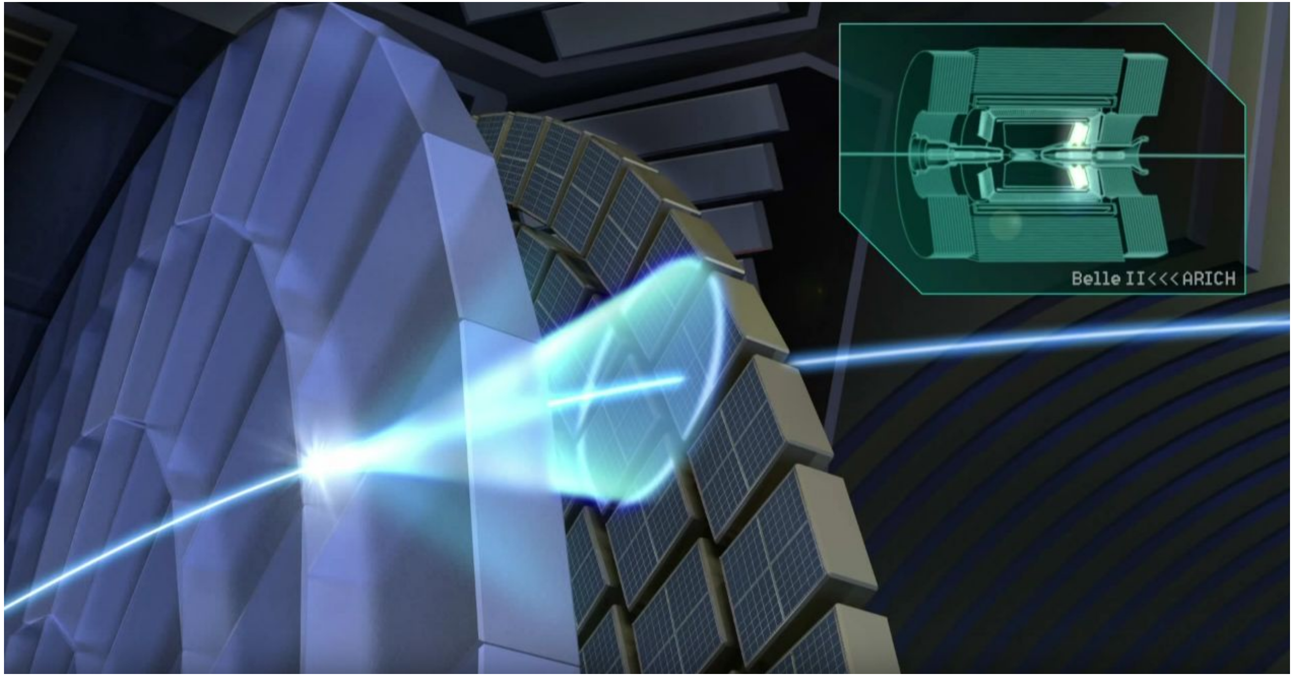


FIG. 1. $\beta\gamma$ universality curve (right) and CDC-based dE/dx curve predictions (left) for different charged particle species.

ARICH and Cherenkov Radiation:

- Measure the radiation cone opening angle and # of photons emitted
- -> velocity



ARICH

- Mass hypotheses have **large** impact on velocity for **low** momentum tracks

- -> large difference on opening angle

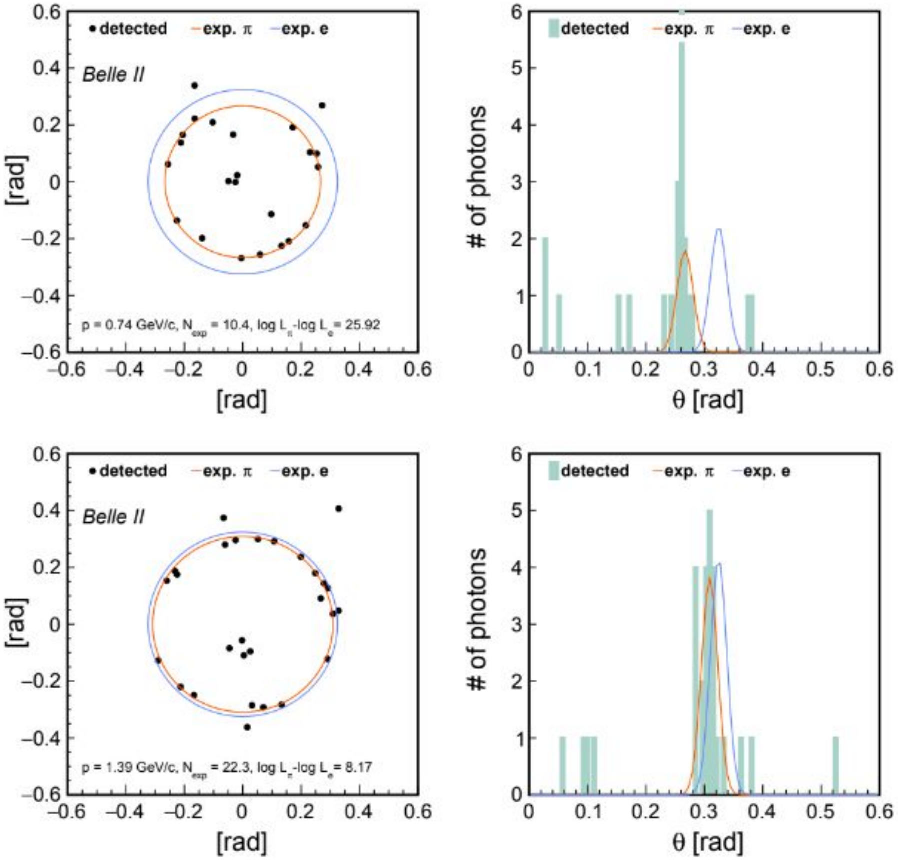
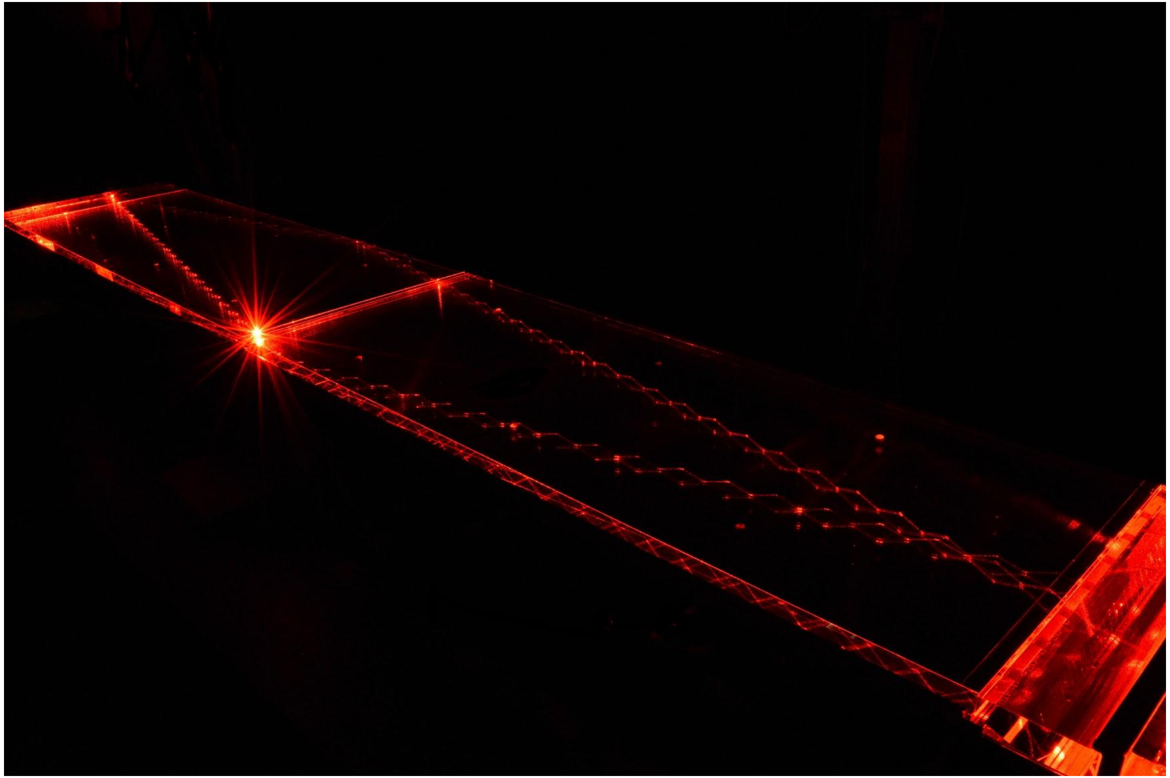


FIG. 6. The observed Cherenkov rings for two pion tracks from $K_S^0 \to \pi^+\pi^-$ decay (on the top with $p = 0.74$ GeV/c and below with $p = 1.39$ GeV/c). The red and blue rings show the expected rings for the pion and electron hypothesis respectively.

TOP and Cherenkov Radiation:

- Measure the **time** and **position** (x vs t) of Cherenkov photon hits on the MCP-PMT (located at one end of the bar)

- -> velocity



TOP and Cherenkov Radiation:

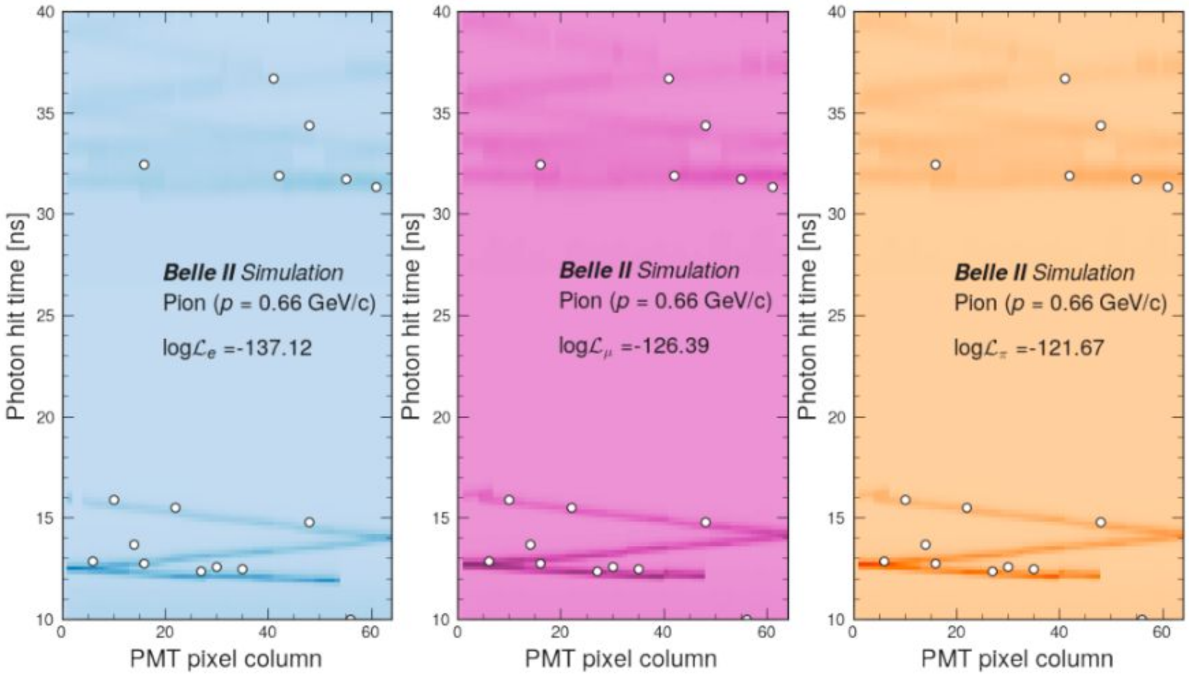


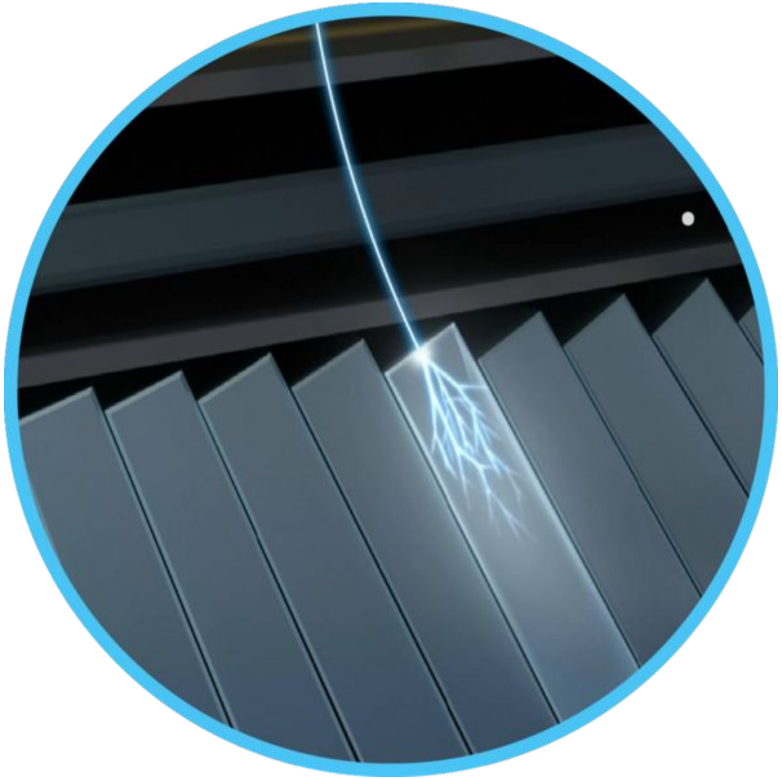
FIG. 5. Comparison between the electron, muon and pion TOP PDFs with the observed signal left by a pion carrying a momentum of 0.66 GeV/c. The eight PMT pixels located at the same transverse position along the array are grouped together for better readability.

ECL and energy deposition:

- Measure the **energy deposition and the width of the shower**, e.g. E/p

- Electrons will create electromagnetic showers and deposit all their energy in the ECL

- Hadrons will likely pass through and not lose much energy

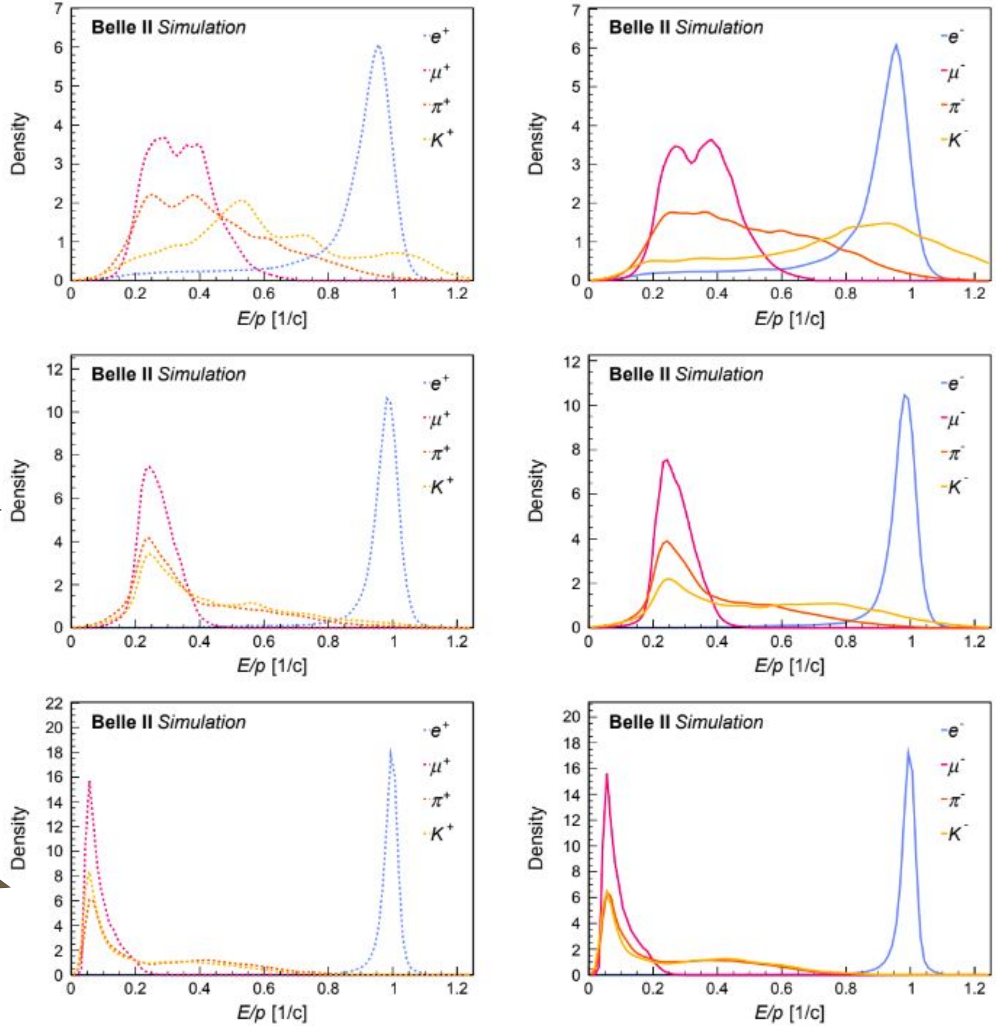


ECL and E/p:

0.2< p < 0.6 GeV

0.6< p < 1 GeV

p > 1 GeV
Most powerful region
Compensate CDC



KLM and penetrating length:

- Measure the longitudinal **penetration depth** and transverse scattering
- Muons penetrate the KLM and leave tracks
- Hadrons create hadronic showers

KLM and penetrating length:

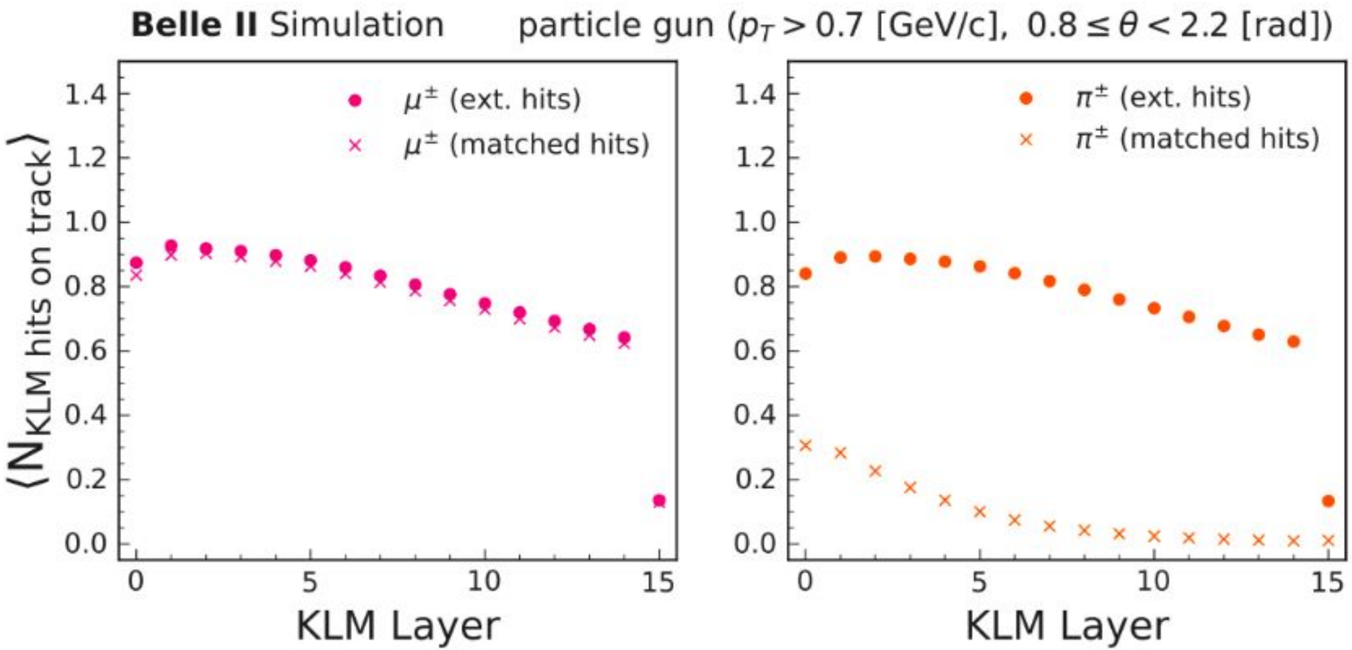


FIG. 8. Average number of extrapolated (solid points) and matched (cross points) KLM hits on track per KLM layer for a sample of muon tracks (magenta) and pion tracks (orange). These are taken from single-particle ("particle gun") samples.

What we get from sub-detectors:

- All the information is converted into a likelihood $\mathbf{L^d(x|i)}$ for each sub-detector.
- Good news: they are defined in basf2 and saved into our exercise samples.
- E.g. eID_CDC, muID_KLM

Auto machine learning

Hands-on exercise: kaggle

https://www.kaggle.com/competitions/2024-b2sw-ml

1 **Auto preprocessing**

Handling nan, large/small numbers, .etc

2 **Auto model selection**

Boosted decision trees or neural network?

3 **Auto hyperparameter tuning**

How many neurons or trees

Backup

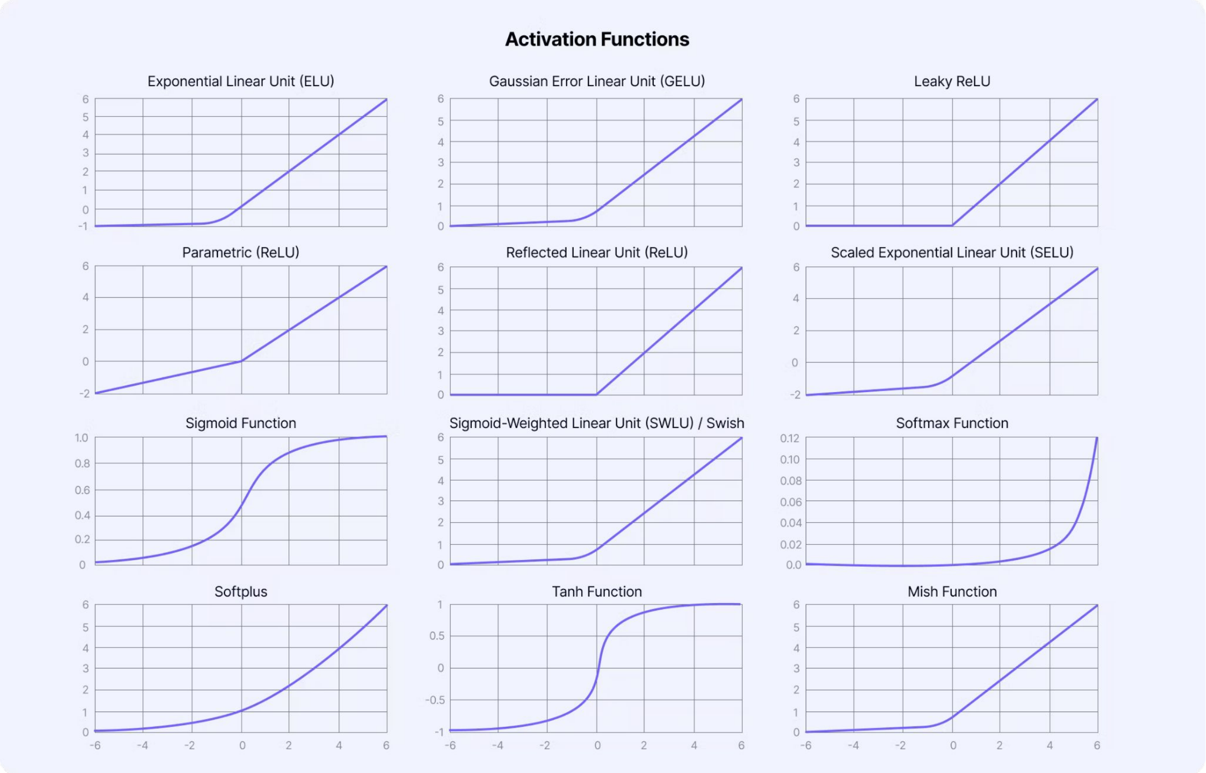
Pitfalls (what to watch out for)

- Vanishing Gradients
- Exploding Gradients
- Overfitting
- High Dimensionality

Hand full of techniques to deal with these:

- Different loss functions or activation functions
    - Regularization Terms
- Dropout or hyper-parameter tuning
- Better feature selection

Ranges of Activation Functions



https://encord.com/blog/activation-functions-neural-networks/

Universal Approximation Theorem

Universal approximation theorem — Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \| f(x) - g(x) \| < \varepsilon$$

where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$

Simple English: Any continuous function f can be approximated given some level of precision.

Similar theorems given for unbounded domains

Regularization

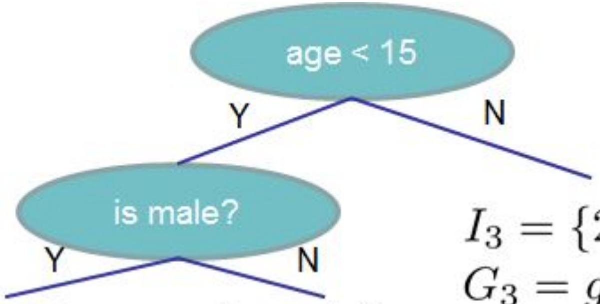
ex. $\mathbb{L} = \mathscr{L}^2 + \boxed{\lambda \sum_n |\vec{w}_n|^2}$

For certain models, one way to tackle overfitting and exploding gradients is by introducing a regularizing term in the loss function.

Sometimes, λ is normalized by the number of nodes/weights you have (but is up to the developer/user) since it is a constant throughout training anyways

New node split and new tree?

Instance index	gradient statistics
1	g1, h1
2	g2, h2
3	g3, h3
4	g4, h4
5	g5, h5



$I_1 = \{1\}$
$G_1 = g_1$
$H_1 = h_1$

$I_2 = \{4\}$
$G_2 = g_4$
$H_4 = h_4$

$I_3 = \{2, 3, 5\}$
$G_3 = g_2 + g_3 + g_5$
$H_3 = h_2 + h_3 + h_5$

$$Obj = -\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$

The smaller the score is, the better the structure is

Gradient Boosting

Algorithm 1: Gradient_Boost

$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N L(y_i, \rho)$

For $m = 1$ to M do:

$\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \ i = 1, N$

$\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^N [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$

$\quad \rho_m = \arg\min_\rho \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

Algorithm 2: LS_Boost

$F_0(\mathbf{x}) = \bar{y}$

For $m = 1$ to M do:

$\quad \tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i), \quad i = 1, N$

$\quad (\rho_m, \mathbf{a}_m) = \arg\min_{\mathbf{a},\rho} \sum_{i=1}^N [\tilde{y}_i - \rho h(\mathbf{x}_i; \mathbf{a})]^2$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

https://jerryfriedman.su.domains/ftp/trebst.pdf

Resources (in general)

In the last 5 years, there is a major abundance of resources to learn ML. Let put a subset of resources that I used to inspire this talk:

S. Still. "Machine Learning: For scientists" (2019).

I. Haide and L. Reuter "Machine Learning: Current Projects at Belle 2" (2023).

S. Dubey. "Machine Learning Hands-On" (2023).

S. Vallecorsa. "Artificial Intelligence and Machine Learning" CHEP 2023.

J. Hurwitz and D. Kirsch "Machine Learning for dummies: IBM Limited Edition."

P. Wittek. "Quantum Machine Learning: What Quantum Computing Means to Data Mining" (2014).

Resources (Neural Networks)

- Neural Networks – State of Art, Brief History, Basic Models and Architecture: https://libguides.aurora.edu/ChatGPT/History-of-AI-and-Neural-Networks
- IBM: What is a neural network: https://www.ibm.com/topics/neural-networks
- Neural Networks (Machine Learning): https://en.wikipedia.org/wiki/Neural_network_(machine_learning)
- Neural network models (supervised): https://scikit-learn.org/stable/modules/neural_networks_supervised.html

Resources (Decision Trees)

- “Boosted Decision Trees.” https://arxiv.org/pdf/2206.09645
- “What is a decision tree?.” https://www.ibm.com/topics/decision-trees
- “Decision Tree.” https://en.wikipedia.org/wiki/Decision_tree
- “FastBDT: A speed-optimized and cache-friendly implementation of stochastic gradient-boosted decision trees for multivariate classification.” https://arxiv.org/abs/1609.06119
- “XGBoost: A Scalable Tree Boosting System.” https://arxiv.org/abs/1603.02754