Machine Learning Hands-on

Shawn Dubey

Belle II Summer Workshop

Duke University

07/25/2023

Introduction by ChatGPT

Artificial Intelligence (AI) and Machine Learning (ML) are related concepts but have distinct differences. Here's an overview:

1. Definitions:
   - Artificial Intelligence (AI): AI refers to the development of computer systems capable of performing tasks that would typically require human intelligence. It involves simulating human-like intelligence in machines to enable them to understand, reason, learn, and make decisions.
   - Machine Learning (ML): ML is a subset of AI that focuses on the development of algorithms and statistical models that enable computer systems to learn from data and make predictions or take actions without being explicitly programmed. ML algorithms learn patterns and relationships within the data to improve their performance over time.

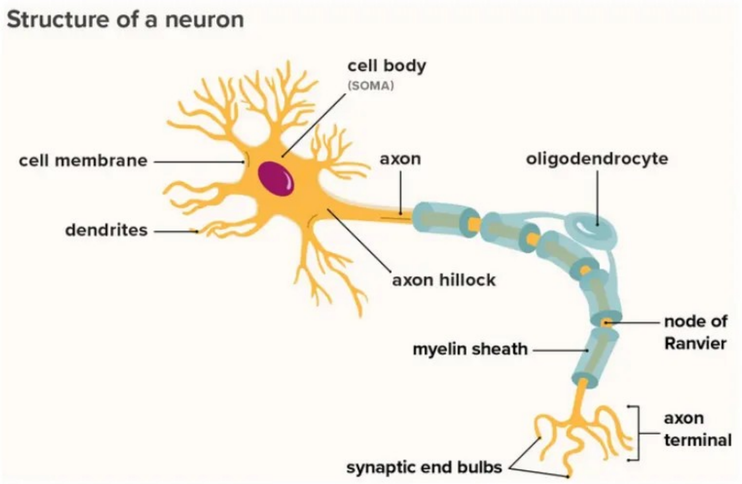
Introduction

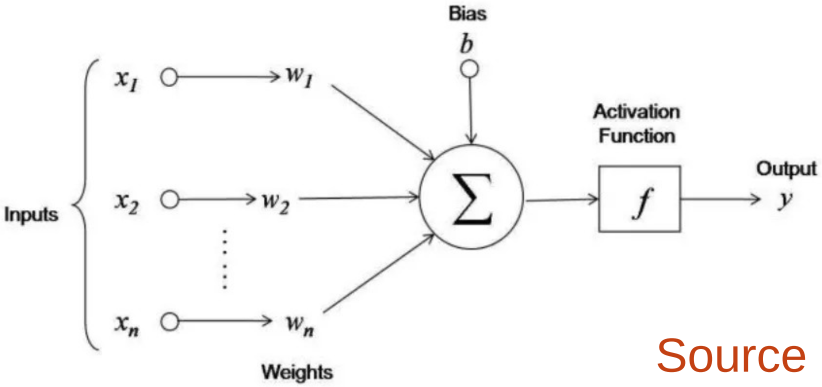
- Different types of machine learning models:
  - Neural Networks (NNs)
  - Decision trees and random forests
  - Various unsupervised learning methods
    - Unsupervised methods (e.g. clustering) learn without using labeled training data
  - Many more

Neural Networks

Artificial Neural Networks

- Models that are inspired by biological neural networks (like the brain), i.e. connections of biological neurons and their behavior





Source

Artificial Neural Networks

- Universal Approximation Theorem: Very simply and approximately, this says that neural networks can approximate any continuous function
  - If there there is a mapping between a set of features and some label/output, the neural network should be able to approximate it.
  - This function can be a function of many variables.

Artificial Neural Networks

- ANNs can be used for *classification* or *regression*.
  - Classification: Predict a class label such as cat vs dog, signal vs background, SM vs NP
    - These are examples of binary classification, but can do multi-class classification, such as with the famous MNIST dataset; common use of machine learning in HEP
  - Regression: Predict a continuous value such as the length of a flower petal.

Artificial Neural Networks

- There are several types of artificial neural networks (ANNs)
  - Fully-connected Networks (FCN; today's exercise)
  - Convolutional Neural Networks (CNN; computer vision)
    - https://arxiv.org/abs/1512.03385
  - Graph Neural Networks (GNN; used in Belle II)
    - https://arxiv.org/abs/2306.04179

Artificial Neural Networks

- **Fully-connected network:** when the neurons in each layer are connected to all neurons in the previous layer
  - Fully-connected or *dense* layers
  - Connections between neurons are given by *weights*

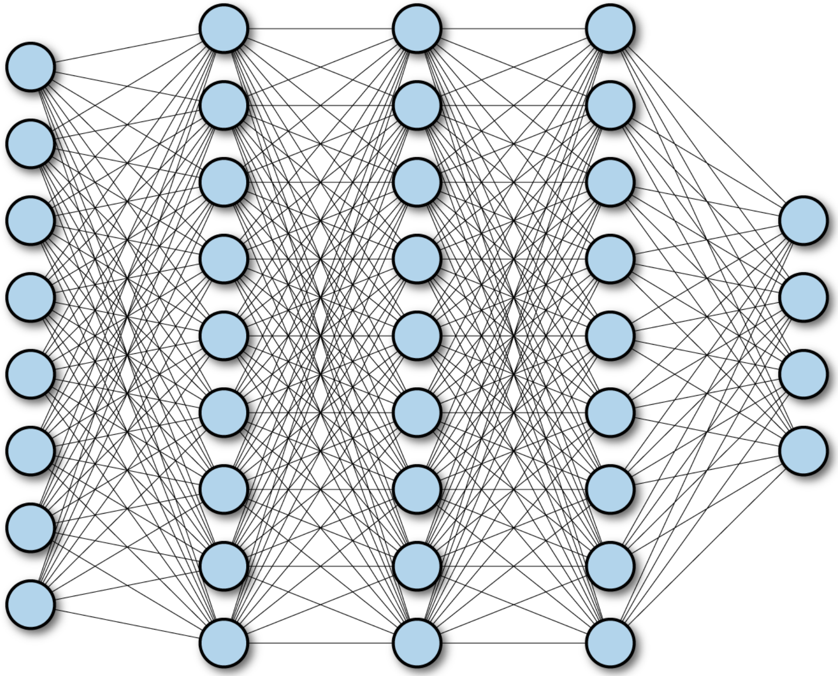
Fully-Connected Networks

- Have at least an *input layer*, where whatever is given to it is simply passed through unaltered, and an *output layer*, which performs the predictive task.

Fully-Connected Networks

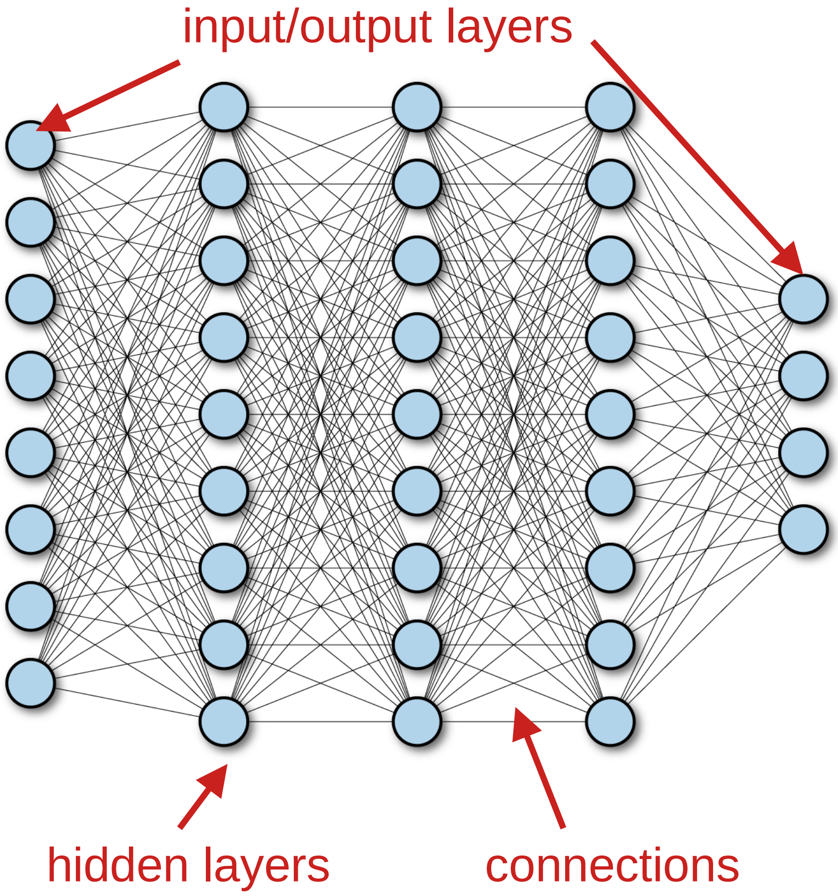
- In between the input and output layers can be any number of *hidden layers*, which perform intermediate computations on their inputs and help obtain the mapping
  - A rule of thumb is that a neural network is deep is if it has more than two hidden layers

Fully-Connected Networks



https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

Fully-Connected Networks



https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

Inner Workings

- The output of a layer (except the input layer) is given by

$$h(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

Inner Workings

- The output of a layer (except the input layer) is given by

$$h(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

Input feature matrix

Inner Workings

- The output of a layer (except the input layer) is given by

$$h(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

Weight matrix

Inner Workings

- The output of a layer (except the input layer) is given by

$$h(\mathbf{X}) = \phi(\mathbf{X}\mathbf{W} + \mathbf{b})$$

bias matrix; add flexibility to the model and allows the model to understand more complicated relationship

Inner Workings

- The output of a layer (except the input layer) is given by

$$h(\mathbf{X}) = \phi(\mathbf{XW} + \mathbf{b})$$

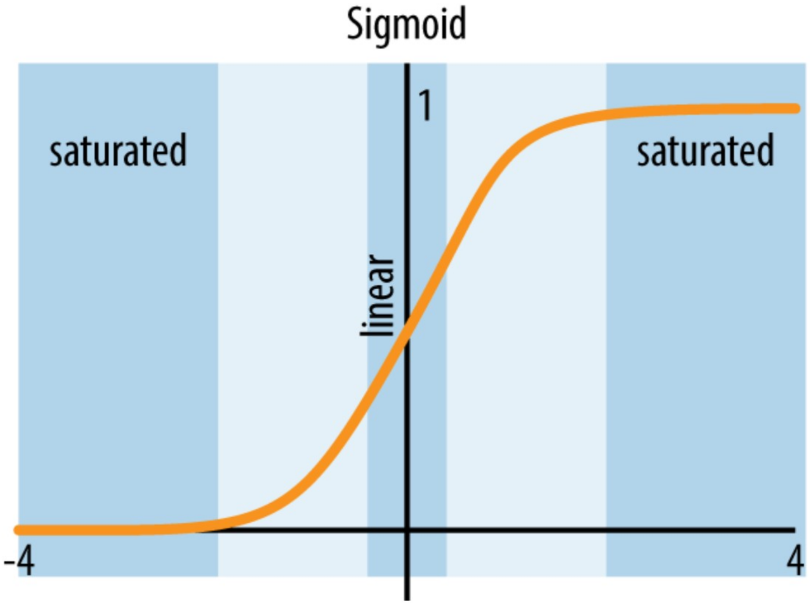
activation function

Activation Functions

- *Activation functions* perform the transformations
- Add non-linearity into the model
  - Helps neural networks model different functions
- Different activation functions for different tasks

Activation Functions

- *Sigmoid function* is a common choice for *binary classification*
  - Saturates at 0 and 1.
- *Softmax function* is a common choice for *multi-class classification*



Learning

- How does all this lead to learning?

Learning

- **Goal**: optimize/minimize a cost/loss function
  - e.g.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

https://en.wikipedia.org/wiki/Mean_squared_error

Learning

- **Goal**: optimize/minimize a cost/loss function
  - e.g.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error

n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

https://en.wikipedia.org/wiki/Mean_squared_error

There are other loss functions that may be better for certain tasks like binary classification (e.g. binary cross-entropy)

Learning

- Use an optimization algorithm like *gradient descent* (GD) to gradually alter parameters that *minimize the loss function*.
- GD measures gradient wrt to a parameter vector **θ**, and moves in the direction of steepest gradient
  - When gradient is zero, you are at a minimum – hopefully global, not local minimum.
    - The goal is to reach a global minimum and not get stuck in a local one
  - Loss functions therefore need to be smooth and convex

Learning

- There are other optimizers, e.g.
  - Stochastic Gradient Descent (SDG): computes gradient based on random instant of training set
  - Adaptive Moment Estimation (Adam)
  - Nesterov-accelerated Adaptive Moment Estimation (Nadam)
  - Many more

Learning

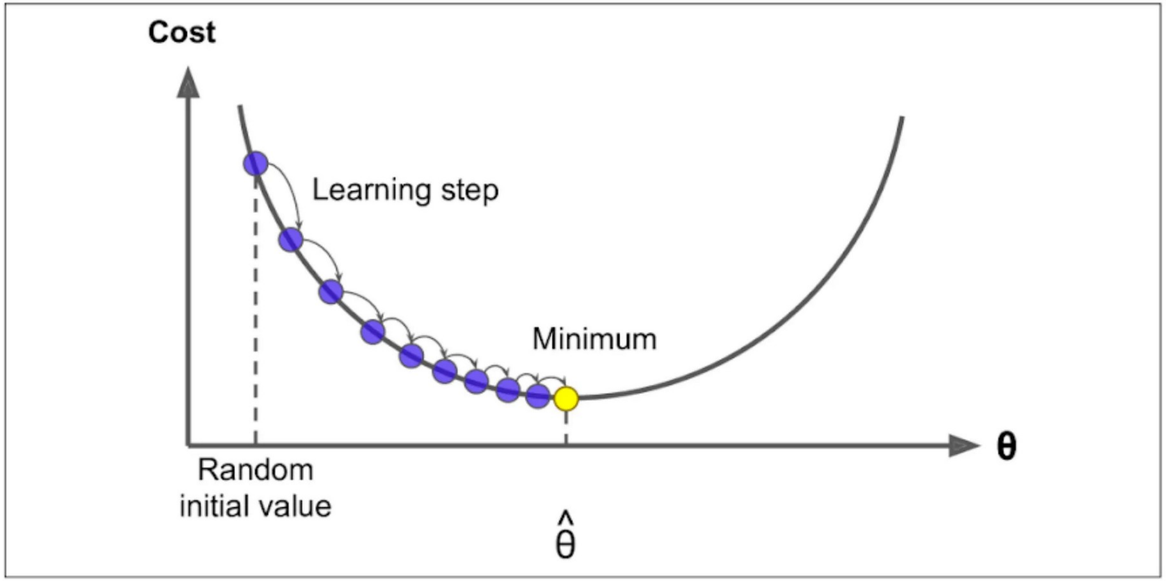
- Importantly, there is a parameter called the *learning rate (LR)* η that affects how optimization algorithms perform.
- The LR controls how much of an adjustment to make to the parameters during learning.
- If the LR is too high, then GD will bounce around the minimum and not converge; too low and it will take too long to converge and you could get stuck in a local minimum.

Learning

- The learning rate is an example of a *hyperparameter*.
- A hyperparameter is a parameter of the learning algorithm and determines behavior and architecture.
  - A few examples are the learning rate, number of hidden layers, and the number of neurons
- This is in contrast to *model parameters* such as weights and biases that are affected by the learning algorithm.

Learning

- If "*f*" is the loss function, then the gradient descent step is given by $\vec{\theta}_{\text{updated}} = \vec{\theta} - \eta \nabla_{\vec{\theta}} f(\vec{\theta})$



"Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow (Concepts, Tools, and Techniques to Build Intelligent Systems)" by Aurelien Geron

Learning

- If the information flows from only input to output, then this is called a *feedforward* neural network.
- The process of *backpropagation* is added to this to efficiently compute the gradients of the loss. Very approximately, it is
  - Forward pass and make prediction
  - Determine error
  - Error gradient is propagated backward and the error contribution from each layer is determined
  - GD is used to modify weights
- Loss and activation functions need to be differentiable
- The learning process is an iterative one that repeats many times.

Problems with Learning

- Vanishing or exploding gradients:
  - **Vanishing** – gradient becomes too small during backpropagation and the model doesn't learn; can be mitigated by e.g. choosing a better activation function
  - **Exploding** – gradients become exceedingly large, causing instability in the model; can mitigate by e.g. tweaking learning rate, changing activation, or, in deep CNNs, adding a "skip" connection (ResNet, outside of today's scope)

Problems with Learning

- Overfitting – the model memorizes features and patterns of the training set and cannot generalize to data it hasn't seen before (test data)
  - Can be mitigated by adding more training data
  - make your model simpler
  - Use early stopping
  - Add a dropout layer
  - etc
- Underfitting – the model doesn't learn
  - Check if the model can overfit on a few training events/instances, maybe 1-10 events
  - Introduce or engineer other features
  - Alter model architecture
  - Add more training data
  - Select additional or different features
  - etc

Hands-on Exercise

Hands-on Exercise

- Your mission, should you choose to accept it, is to classify the continuum background in $B \to K$ pi^0 decays.
- But what is continuum background?

Hands-on Exercise

- Continuum background comes from the process $e^+e^- \rightarrow q\bar{q}$ (q = u, d, s, c).

Hands-on Exercise

The continuum particles are strongly collimated due to the large available momentum for the decay to light hadrons. In contrast, the particles from the BB event are uniformly distributed.

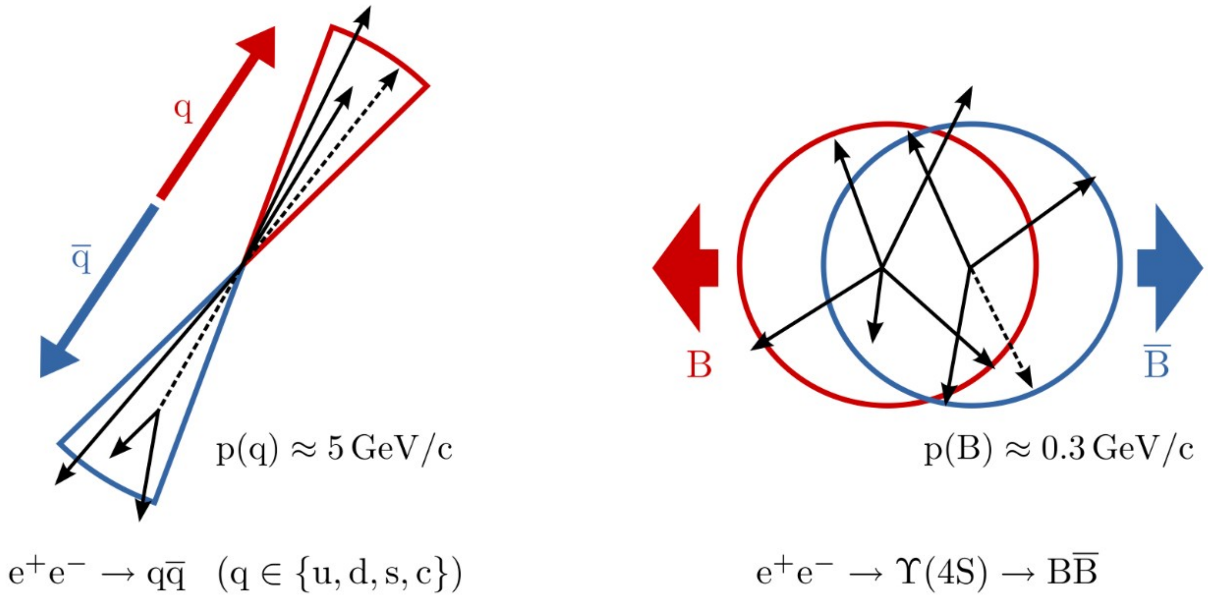


Fig. 3.27 (Credit: Markus Röhrken)

https://software.belle2.org/development/sphinx/online_book/basf2/cs.html

Hands-on Exercise

- Use features of the event topology to build a (binary) classifier in Tensorflow/Keras
- It is up to you to choose the appropriate features
  - Think wisely about what to use (e.g. do we use event shape variables or something else?)
- Main objective: train the neural network to distinguish between B events and continuum events.

Hands-on Exercise

Which properties can we use? A popular one is the ratio of the second and zeroth Fox-Wolfram moment:

$$R_2 = \frac{H_2}{H_0}$$

This variable is called `R2` in basf2 (not to be confused with `foxWolframR2` which is the same property but from the Event Shape Framework).

Fox-Wolfram moments are rotationally-invariant parametrisations of the distribution of particles in an event. They are defined by

$$H_l = \sum_{i,j} \frac{|p_i||p_j|}{E^2_{\text{event}}} P_l(\cos\theta_{i,j})$$

with the momenta $p_{i,j}$, the angle $\theta_{i,j}$ between them, the total energy in the event E_{event} and the Legendre Polynomials P_l.

https://software.belle2.org/development/sphinx/online_book/basf2/cs.html

Hands-on Exercise

Which properties can we use? A popular one is the ratio of the second and zeroth Fox-Wolfram moment:

$$R_2 = \frac{H_2}{H_0}$$

This variable is called `R2` in basf2 (not to be confused with `foxWolframR2` which is the same property but from the Event Shape Framework).

Fox-Wolfram moments are rotationally-invariant parametrisations of the distribution of particles in an event. They are defined by

We have KSFW moments for this exercise.

$$H_l = \sum_{i,j} \frac{|p_i||p_j|}{E^2_{\text{event}}} P_l(\cos\theta_{i,j})$$

with the momenta $p_{i,j}$, the angle $\theta_{i,j}$ between them, the total energy in the event E_{event} and the Legendre Polynomials P_l.

https://software.belle2.org/development/sphinx/online_book/basf2/cs.html

Hands-on Exercise

Which properties can we use? A popular one is the ratio of the second and zeroth Fox-Wolfram moment:

$$R_2 = \frac{H_2}{H_0}$$

This variable is called `R2` in basf2 (not to be confused with `foxWolframR2` which is the same property but from the Event Shape Framework).

Fox-Wolfram moments are rotationally-invariant parametrisations of the distribution of particles in an event. They are defined by

$$H_l = \sum_{i,j} \frac{|p_i||p_j|}{E^2_{\text{event}}} P_l(\cos\theta_{i,j})$$

with the momenta $p_{i,j}$, the angle $\theta_{i,j}$ between them, the total energy in the event E_{event} and the Legendre Polynomials P_l.

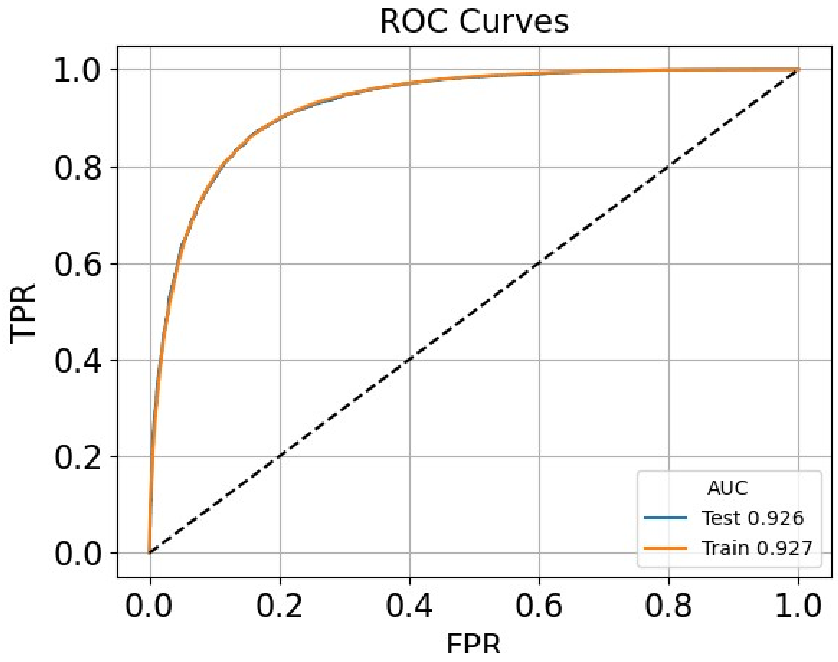
Go here for more variable/feature suggestions

https://software.belle2.org/development/sphinx/online_book/basf2/cs.html

Hands-on Exercise

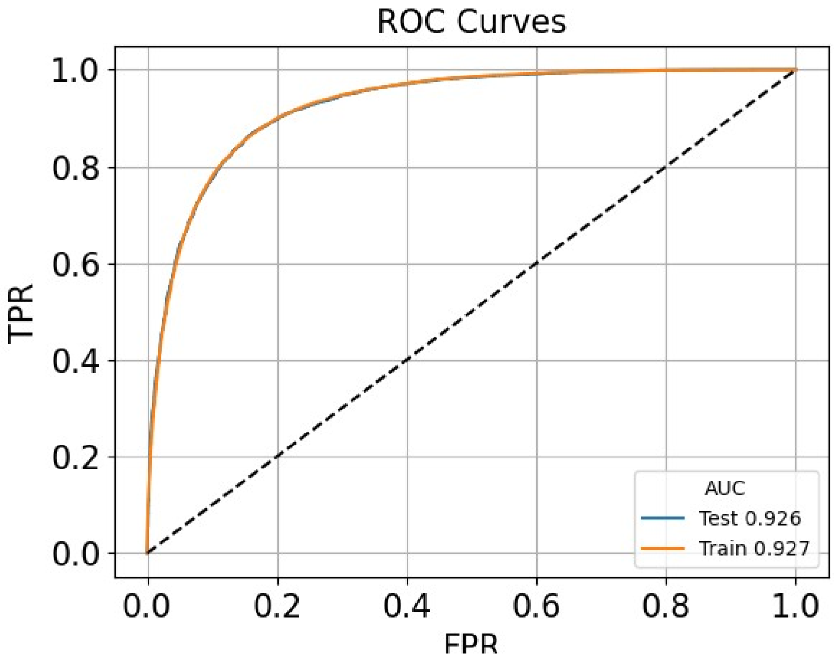
- We will implement a competition
  - The team with the best evaluation metrics will win a prize
  - Teams are max 3 people each
  - Evaluation will be done with AUC
  - Use the Jupyter notebook to produce a list of predictions for each event in the test set and submit it to the Kaggle page.
    - Code is already written for you.

Hands-on Exercise



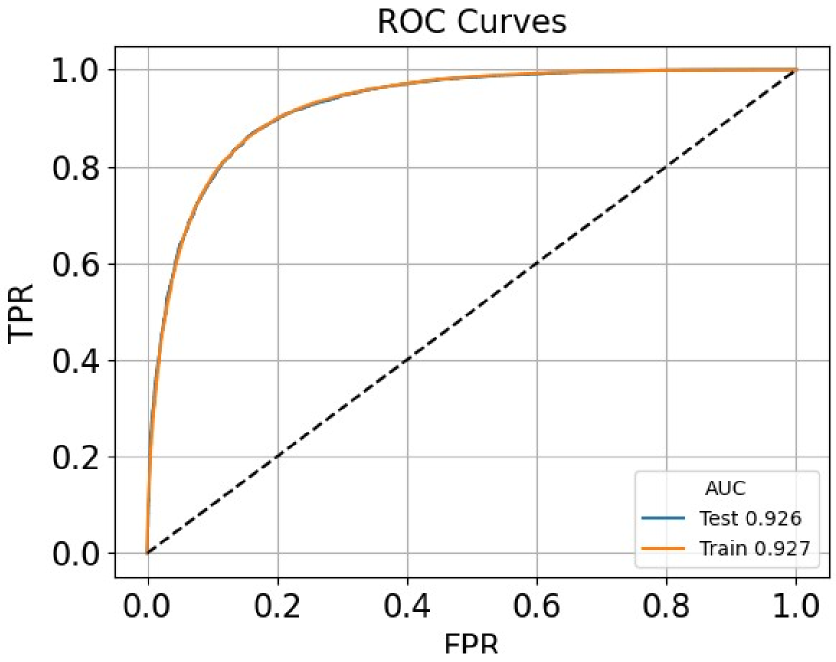
Hands-on Exercise

Receiver Operating Characteristics



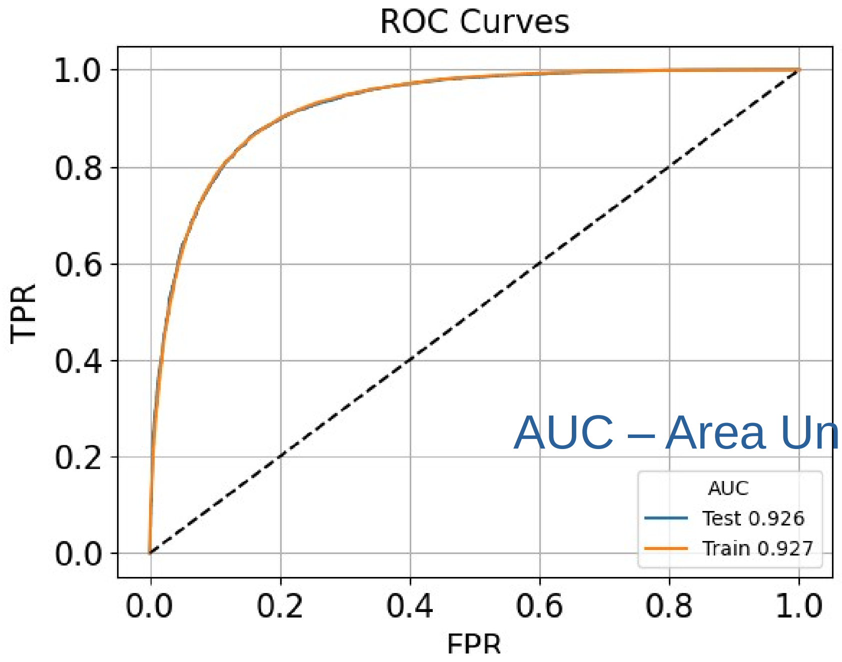
Hands-on Exercise

TPR – True Positive Rate



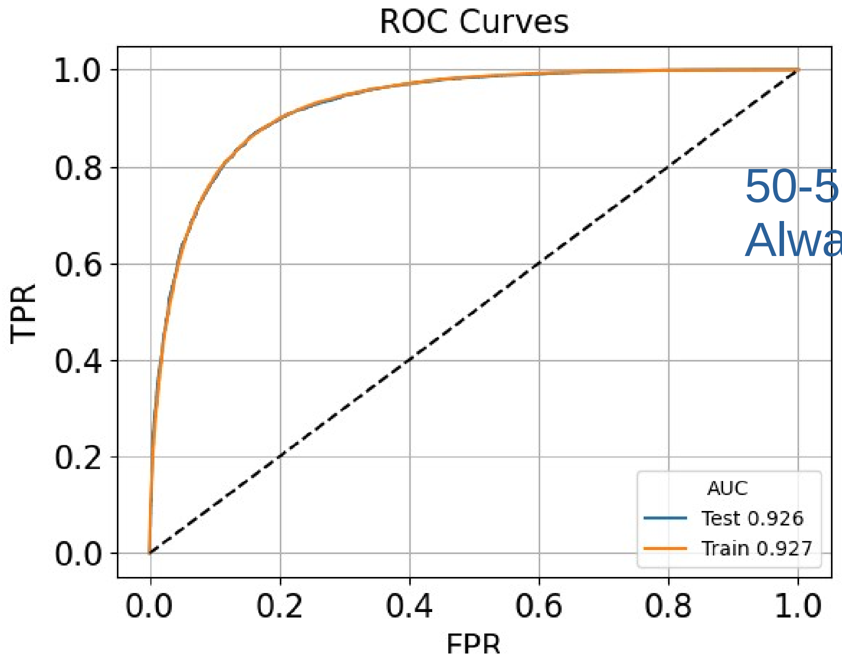
FPR – False Positive Rate

Hands-on Exercise



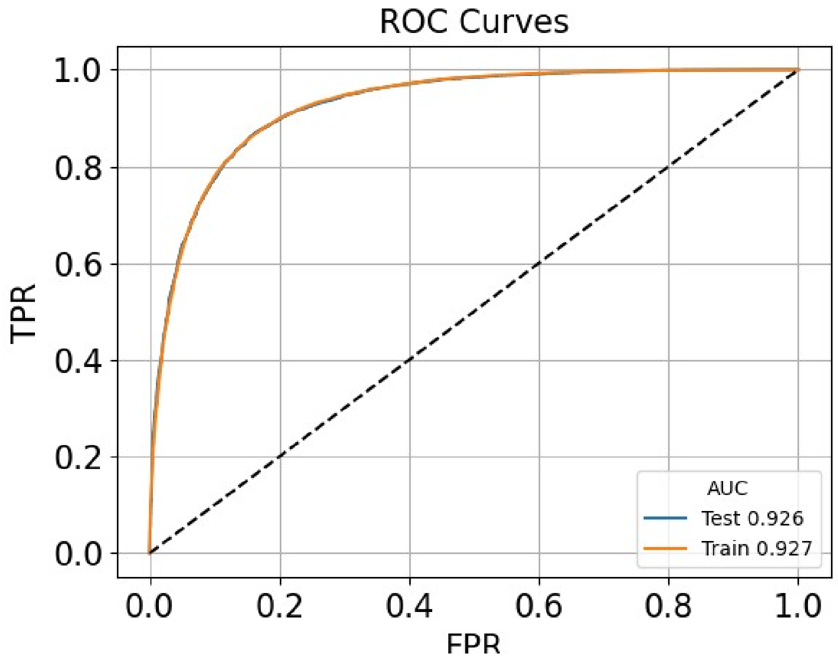
AUC – Area Under Curve

Hands-on Exercise



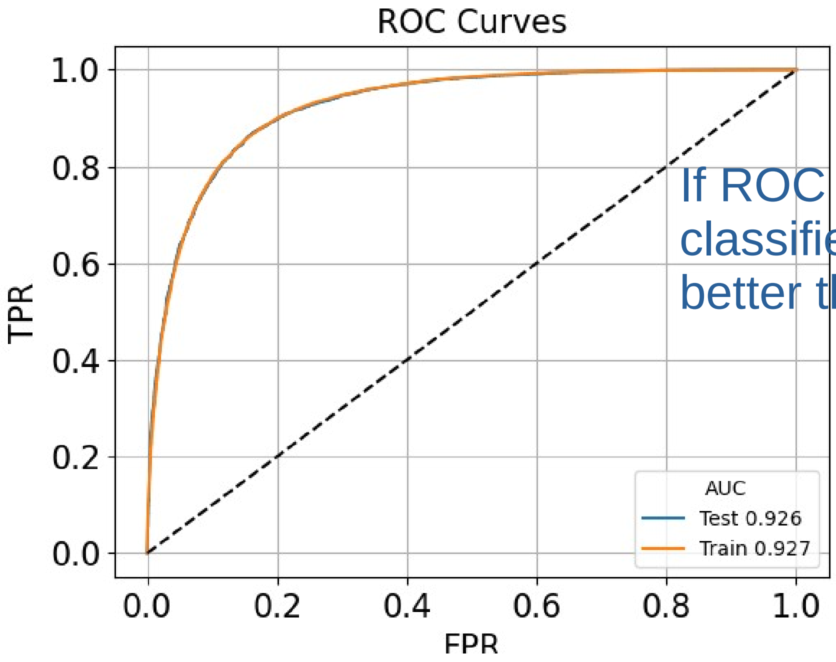
50-50 Line – Random Guess
Always want to be above this line

Hands-on Exercise



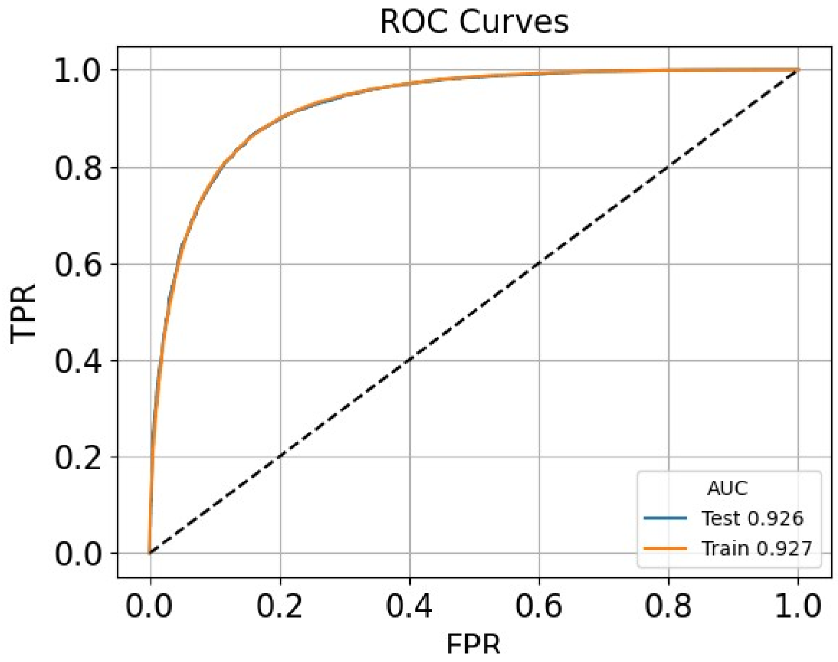
Want AUC closer to 1

Hands-on Exercise



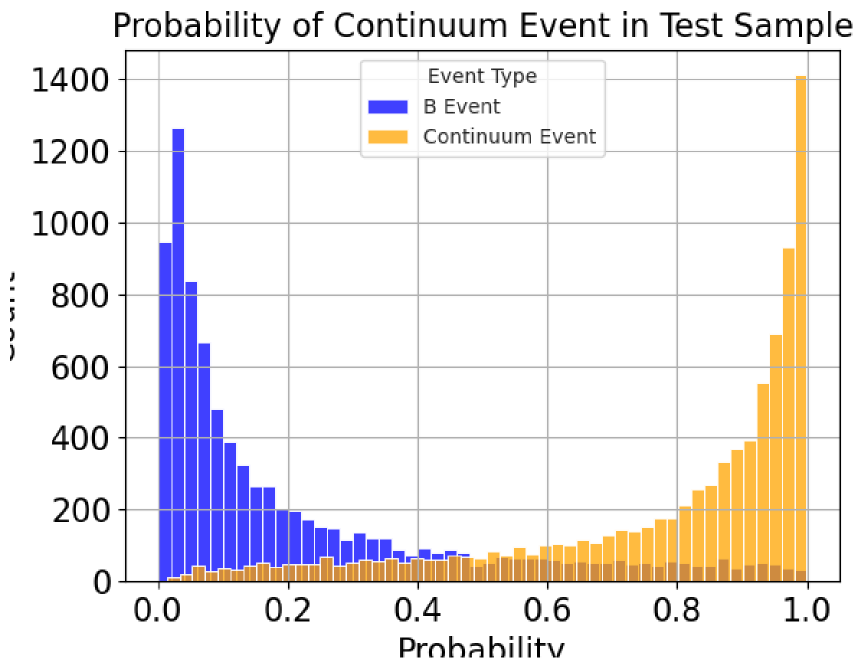
If ROC curve’s at 50-50 line, classifier is no better than a coin flip

Hands-on Exercise



Different AUC for both curves is an indication of overfitting

Hands-on Exercise



Hands-on Exercise

1) Navigate to the Kaggle page and follow the instructions. You will need to sign in.
https://www.kaggle.com/t/1892b4abfb5e4b46bc8f9b0acf70d550

Hands-on Exercise

- You must…
  1) Download the Jupyter notebook from the Indico page or ~sdubey/public/US_BelleII_Summer_Workshop_2023/notebooks on KEKCC
  2) Download the training and test data csv files from the Kaggle page, Indico, or ~sdubey/public/US_BelleII_Summer_Workshop_2023/data on KEKCC
  3) Open the notebook and be sure to read the markdown cells.

Hands-on Exercise

1) Follow each code cell
    1) Select your features
    2) Build you model
        1) There is a build_model() function whose hyperparameters and optimizers you can modify
            1) You can do this by hand, or if ambitious, use scikit-learn's GridSearchCV or RandomSearchCV to select your hyperparameters (Keras has something similar called Keras Tuner)
        2) You will have to write the code for this yourself but it is not required.
2) The notebook will output a CSV file with a column for event IDs and a column for the classifier prediction for each value.
    1) You will upload this to the Kaggle page as your competition submission
    2) You are able to make 20 submissions per day. So you can improve your score if you want

Hands-on Exercise

1) **WARNING!** DOING A SEARCH FOR BEST HYPERPARAMETERS USING THESE FUNCTIONS MAY TAKE A LONG TIME, DEPEDING ON THE PARAMETER SPACE YOU SEACH.
2) GridSearchCV, for example, does not scale well as it performs an exhaustive search; RandomSearchCV is better, as it randomly selects from the parameter space, but may still take a while.

Hands-on Exercise

- If this seems daunting, don't worry, most of this has been implemented for you in the Jupyter notebook (except the parameter tuner implementations).
- All you have to do is fill in the blanks with whatever you think is best.

Hands-on Exercise

Id	B_isContinuumEvent
11246	0.5
43945	0.5
41103	0.5
13420	0.5
16675	0.5
44537	0.5
31795	0.5
13637	0.5
17770	0.5
66153	0.5
81048	0.5

You will produce a file that looks like this
Notebook produces it for you.
To be submitted to Kaggle.

Hands-on Exercise

Hands-on Exercise

- If you have questions, just ask.