XGBoost for Mass Reconstruction in DarkLight@ARIEL

Sid Gupte, Win Lin

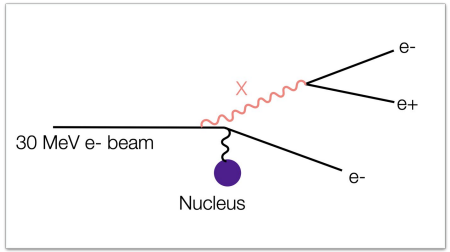
Stony Brook University

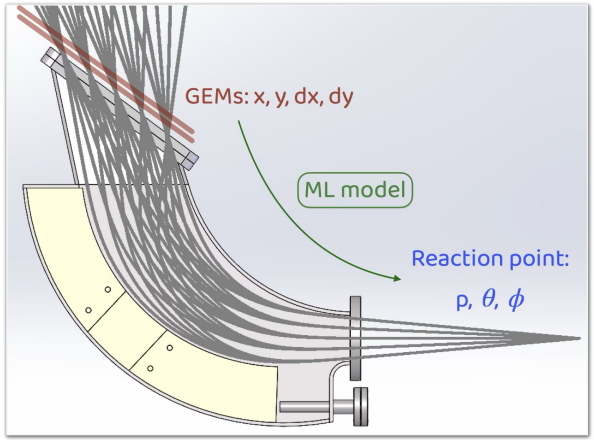
CFNS MLLM meeting 02/28/2025

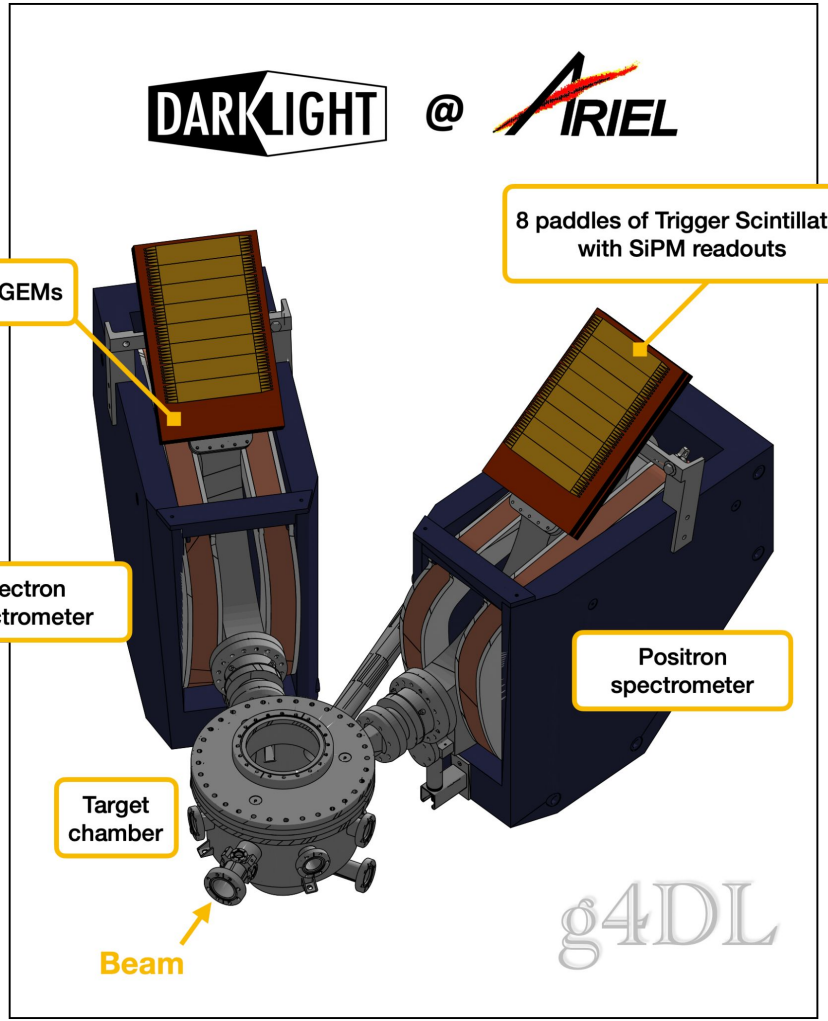
Introduction

- The DarkLight Experiment

Physics process of potential “dark photon” signal



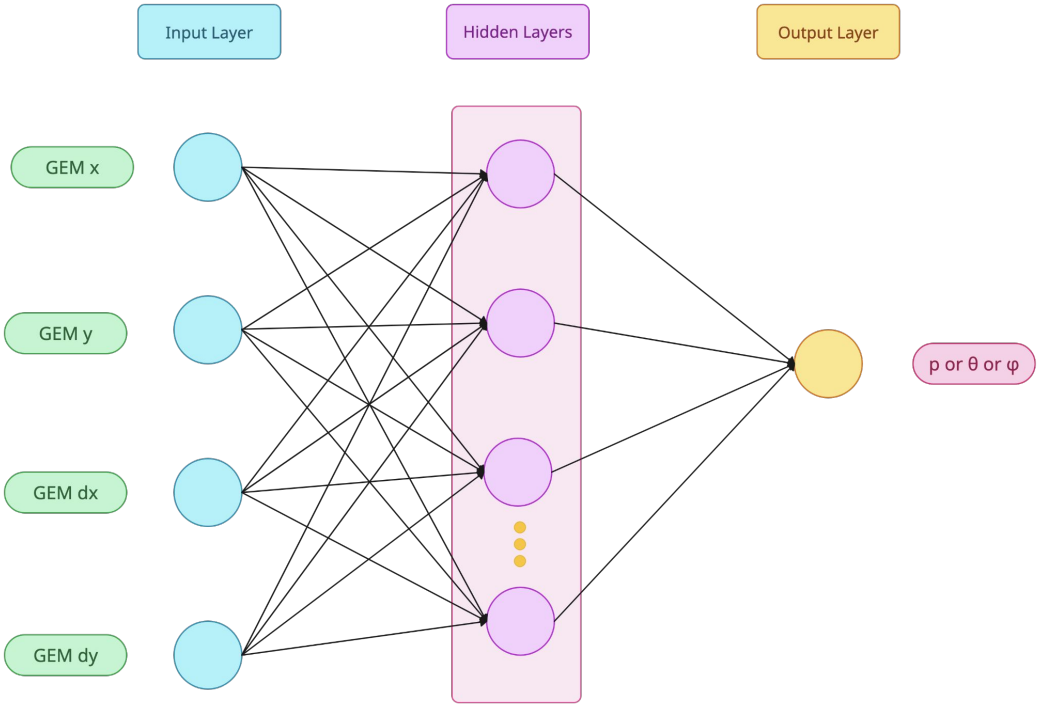




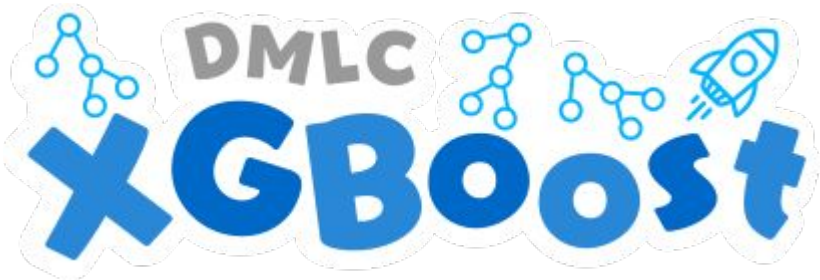
We tried linear neural network:

But:

- Slow to train
- Parameters (layers, neurons etc.) are not necessarily intuitive to set



Tried new model:



- Reference: https://doi.org/10.1145/2939672.2939785
- Webpage: https://xgboost.readthedocs.io/en/stable/
- GitHub: https://github.com/dmlc/xgboost
- Multiple API: Python, R, JVM, C, C++, Ruby, Julia, Swift …

XGBoost:

- Extreme Gradient Boosting

XGBoost:

- Extreme Gradient Boosting

  - Decision tree
  - Ensemble learning

XGBoost:

- Extreme Gradient Boosting
  - Decision tree
  - Ensemble learning
  - Gradient descent

XGBoost:

- Extreme Gradient Boosting
  - Decision tree
  - Ensemble learning
  - Gradient descent
  - Include more regulations

XGBoost:

- Extreme Gradient Boosting

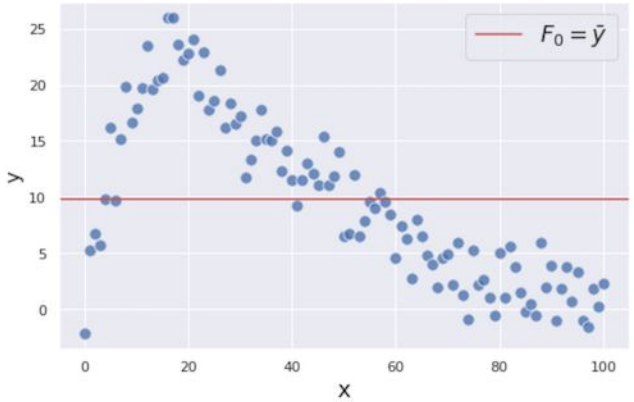
  - Decision tree
  - Ensemble learning
  - Gradient descent
  - Include more regulations

Good for supervised classification and regression.

Yuri's talk on decision tree and gradient boosting:
https://indico.cfnssbu.physics.sunysb.edu/event/388/contributions/1434/attachments/539/846/Trees.pdf

Gradient boosting

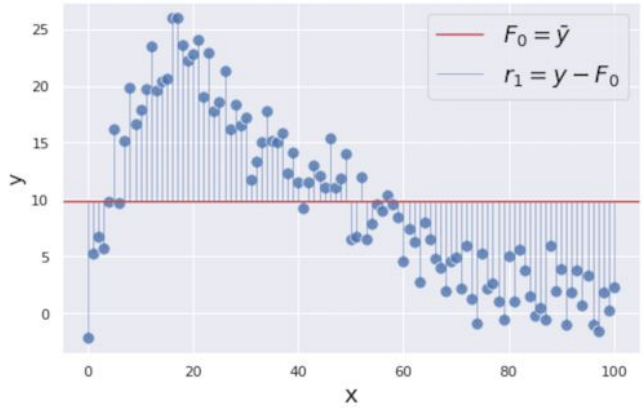
- simple regression example



https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

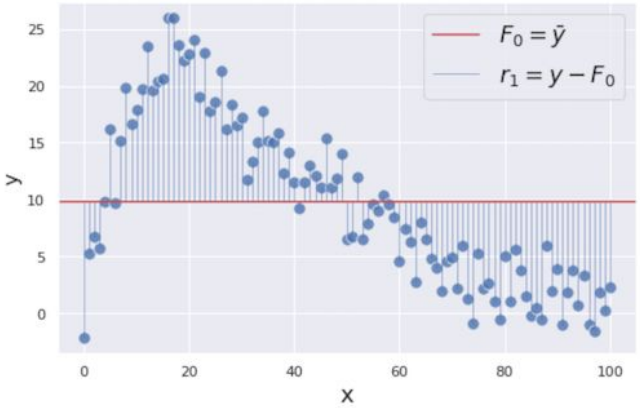
- simple regression example

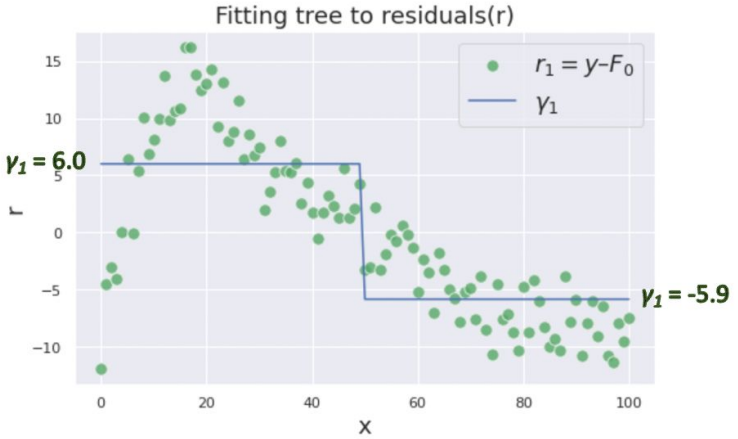


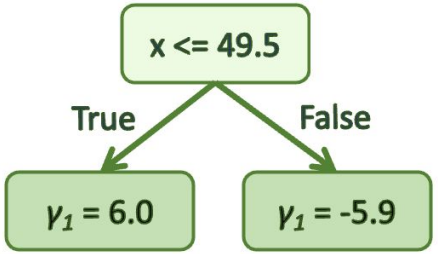
https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

- simple regression example



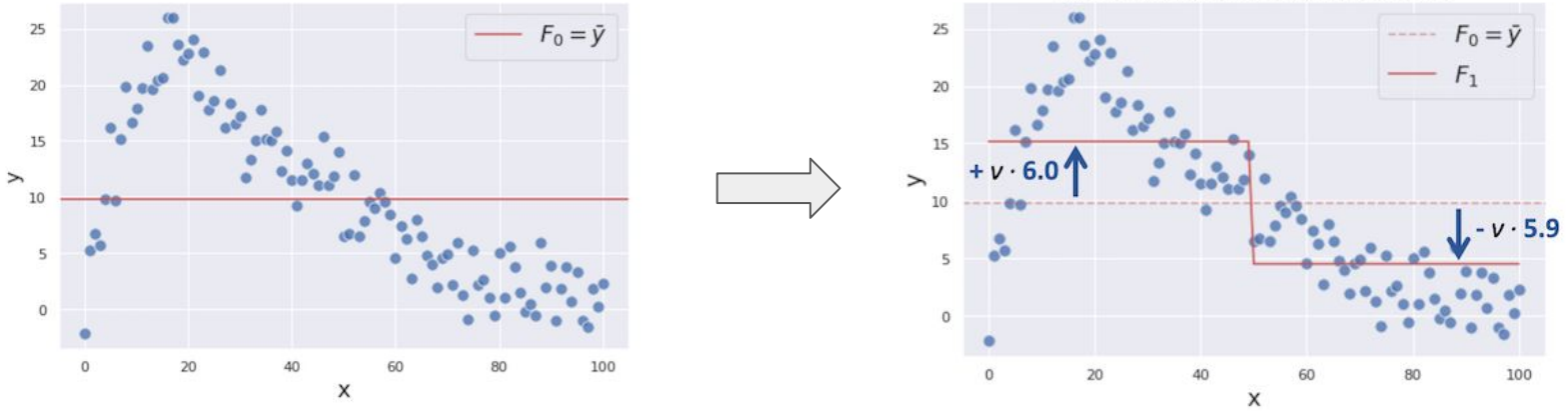




https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

- simple regression example

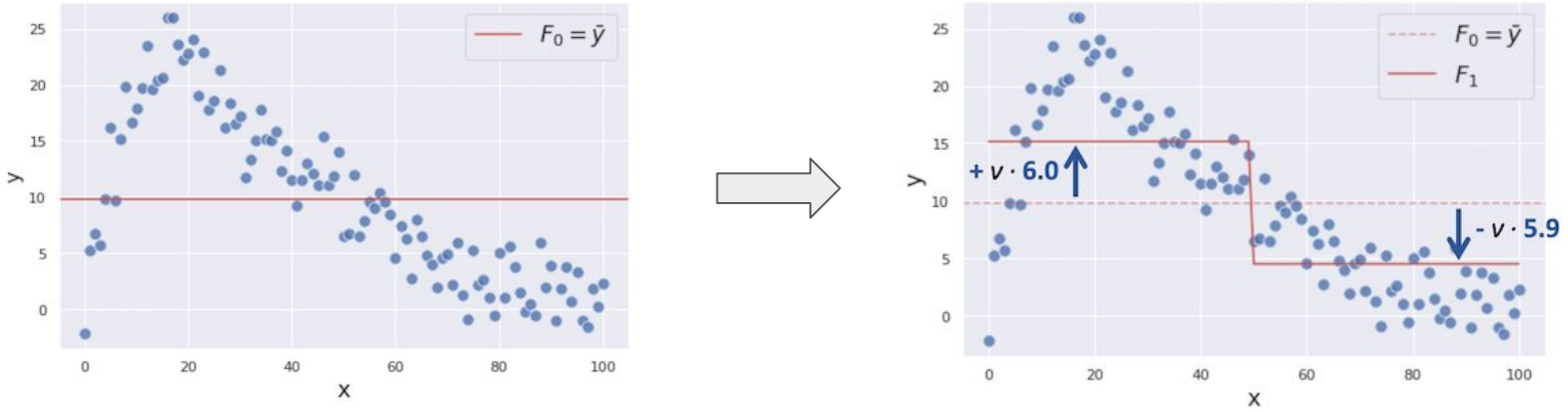


$$F_1 = F_0 + v \cdot \gamma_1 = \begin{cases} F_0 + v \cdot 6.0 & \text{if x} \leq 49.5 \\ F_0 - v \cdot 5.9 & \text{otherwise} \end{cases}$$

https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

- simple regression example



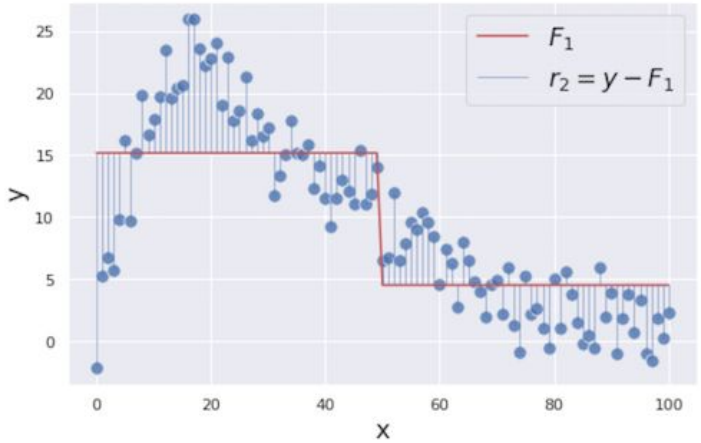
$$F_1 = F_0 + \boxed{v} \cdot \gamma_1 = \begin{cases} F_0 + v \cdot 6.0 & \text{if x} \leq 49.5 \\ F_0 - v \cdot 5.9 & \text{otherwise} \end{cases}$$

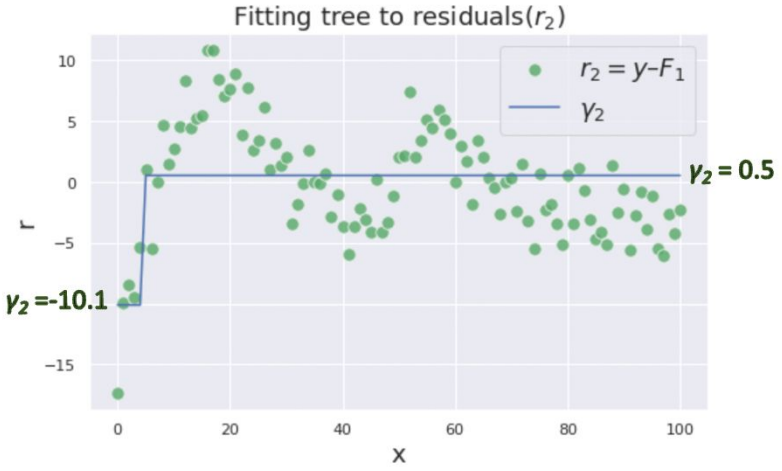
Learning rate

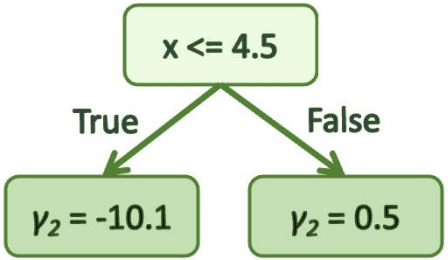
https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

- simple regression example



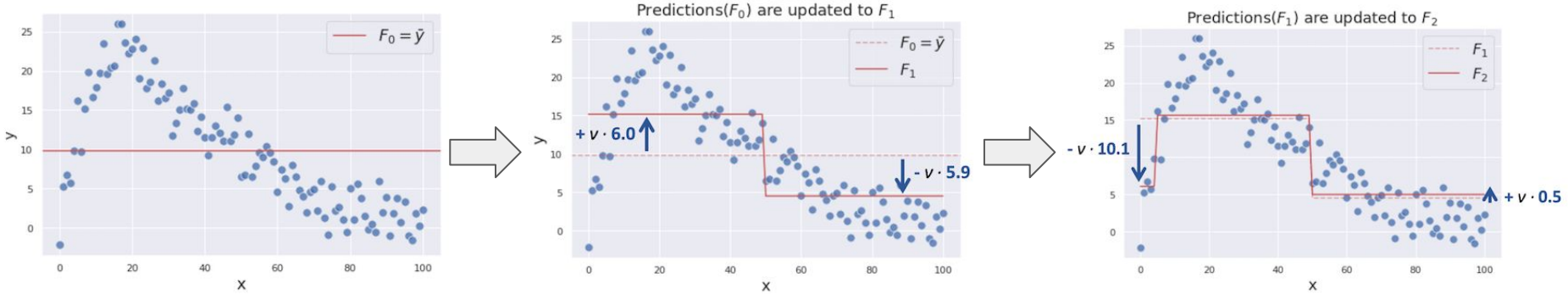




https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

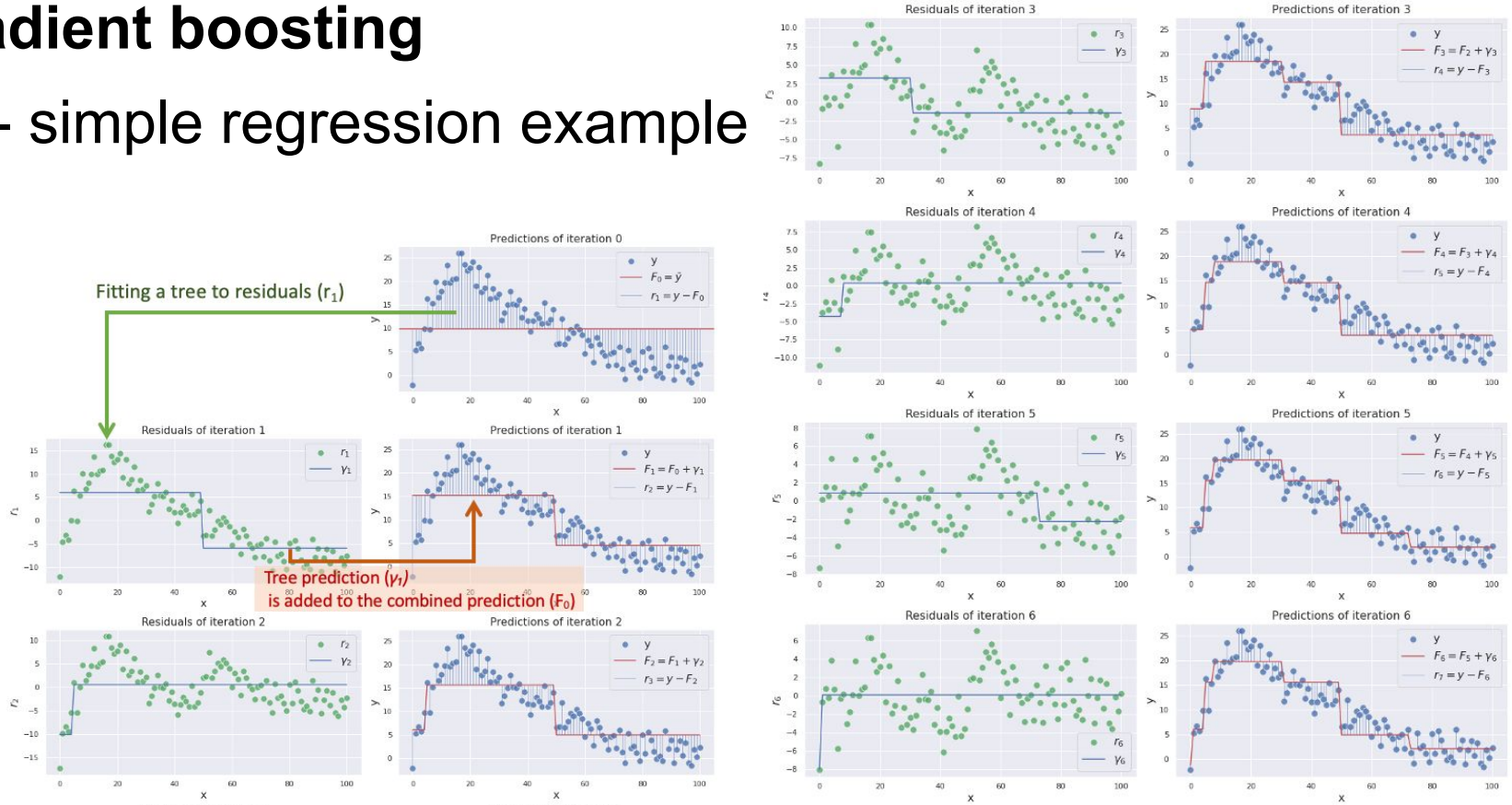
- simple regression example



https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

Gradient boosting

- simple regression example



https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

XGBoost:

- Extreme Gradient Boosting
  - Include more regulations

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

Loss function

Regularization term

XGBoost:

- Extreme Gradient Boosting
  - Include more regulations

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

Loss function

Regularization term

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^T w_j^2$$

Control complexity of tree

n leaves in a tree

Penalize the weight

XGBoost:

- Extreme Gradient Boosting
  - Include more regulations

- Parameters:
  - Learning rate
  - Max depth
  - γ, λ, α
  - Subsample
  - n estimator
  - Colsample bytree
  - Min_child_weight
  - Scale_pos_weight

See: https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/?ref=ml_lbp

Example code:

Load model

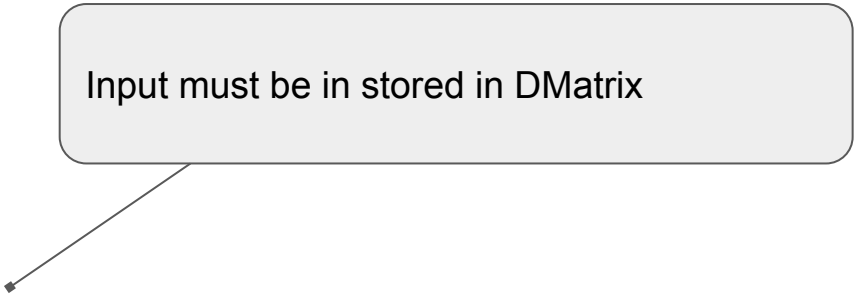
Load Scikit-Learn API

```
# Necessary imports
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE

# Load the data
dataset = pd.read_csv("boston_house.csv")
X, y = dataset.iloc[:, :-1], dataset.iloc[:, -1]

# Splitting
train_X, test_X, train_y, test_y = train_test_split(X, y,
                      test_size = 0.3, random_state = 123)

# Instantiation
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

# Fitting the model
xgb_r.fit(train_X, train_y)

# Predict the model
pred = xgb_r.predict(test_X)

# RMSE Computation
rmse = np.sqrt(MSE(test_y, pred))
print("RMSE : % f" %(rmse))
```

https://www.geeksforgeeks.org/xgboost-for-regression/

Example code:

```
# Necessary imports
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE

# Load the data
dataset = pd.read_csv("boston_house.csv")
X, y = dataset.iloc[:, :-1], dataset.iloc[:, -1]

# Splitting
train_X, test_X, train_y, test_y = train_test_split(X, y,
                      test_size = 0.3, random_state = 123)

# Instantiation
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

# Fitting the model
xgb_r.fit(train_X, train_y)

# Predict the model
pred = xgb_r.predict(test_X)

# RMSE Computation
rmse = np.sqrt(MSE(test_y, pred))
print("RMSE : % f" %(rmse))
```



https://www.geeksforgeeks.org/xgboost-for-regression/

Example code:

```
# Necessary imports
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE

# Load the data
dataset = pd.read_csv("boston_house.csv")
X, y = dataset.iloc[:, :-1], dataset.iloc[:, -1]

# Splitting
train_X, test_X, train_y, test_y = train_test_split(X, y,
                      test_size = 0.3, random_state = 123)

# Instantiation
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

# Fitting the model
xgb_r.fit(train_X, train_y)

# Predict the model
pred = xgb_r.predict(test_X)

# RMSE Computation
rmse = np.sqrt(MSE(test_y, pred))
print("RMSE : % f" %(rmse))
```

Initialize model:
Choose the training type and set parameters

https://www.geeksforgeeks.org/xgboost-for-regression/

Example code:

```
# Necessary imports
import numpy as np
import pandas as pd
import xgboost as xg
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as MSE

# Load the data
dataset = pd.read_csv("boston_house.csv")
X, y = dataset.iloc[:, :-1], dataset.iloc[:, -1]

# Splitting
train_X, test_X, train_y, test_y = train_test_split(X, y,
                        test_size = 0.3, random_state = 123)

# Instantiation
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

# Fitting the model
xgb_r.fit(train_X, train_y)

# Predict the model
pred = xgb_r.predict(test_X)

# RMSE Computation
rmse = np.sqrt(MSE(test_y, pred))
print("RMSE : % f" %(rmse))
```

Use OPTUNA to optimize parameters:

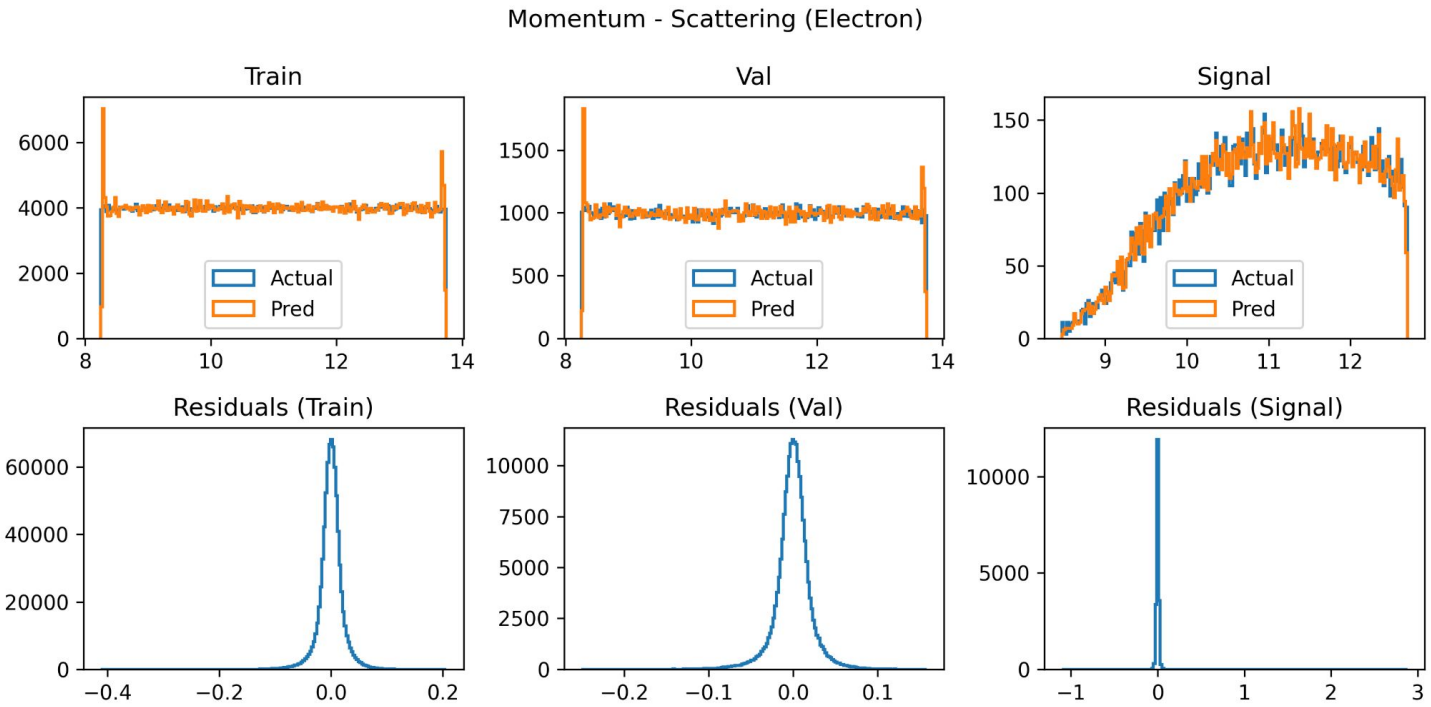
```
def objective(self, trial):
    """Objective function for Optuna parameter tuning."""
    params = {
        'objective': 'reg:squarederror',
        'n_estimators': trial.suggest_int('n_estimators', 200, 1500),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.2, log=True),
        'max_depth': trial.suggest_int('max_depth', 3, 20),
        'min_child_weight': trial.suggest_int('min_child_weight', 1, 10),
        'gamma': trial.suggest_float('gamma', 0, 5.0),
        'reg_alpha': trial.suggest_float('reg_alpha', 1e-3, 10.0, log=True),
        'reg_lambda': trial.suggest_float('reg_lambda', 1e-3, 10.0, log=True),
        'subsample': trial.suggest_float('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.6, 1.0),
        'n_jobs': self.n_jobs,
        'early_stopping_rounds': 10
    }
```

See our slides last time for more OPTUNA details:
https://indico.cfnssbu.physics.sunysb.edu/event/391/

https://www.geeksforgeeks.org/xgboost-for-regression/

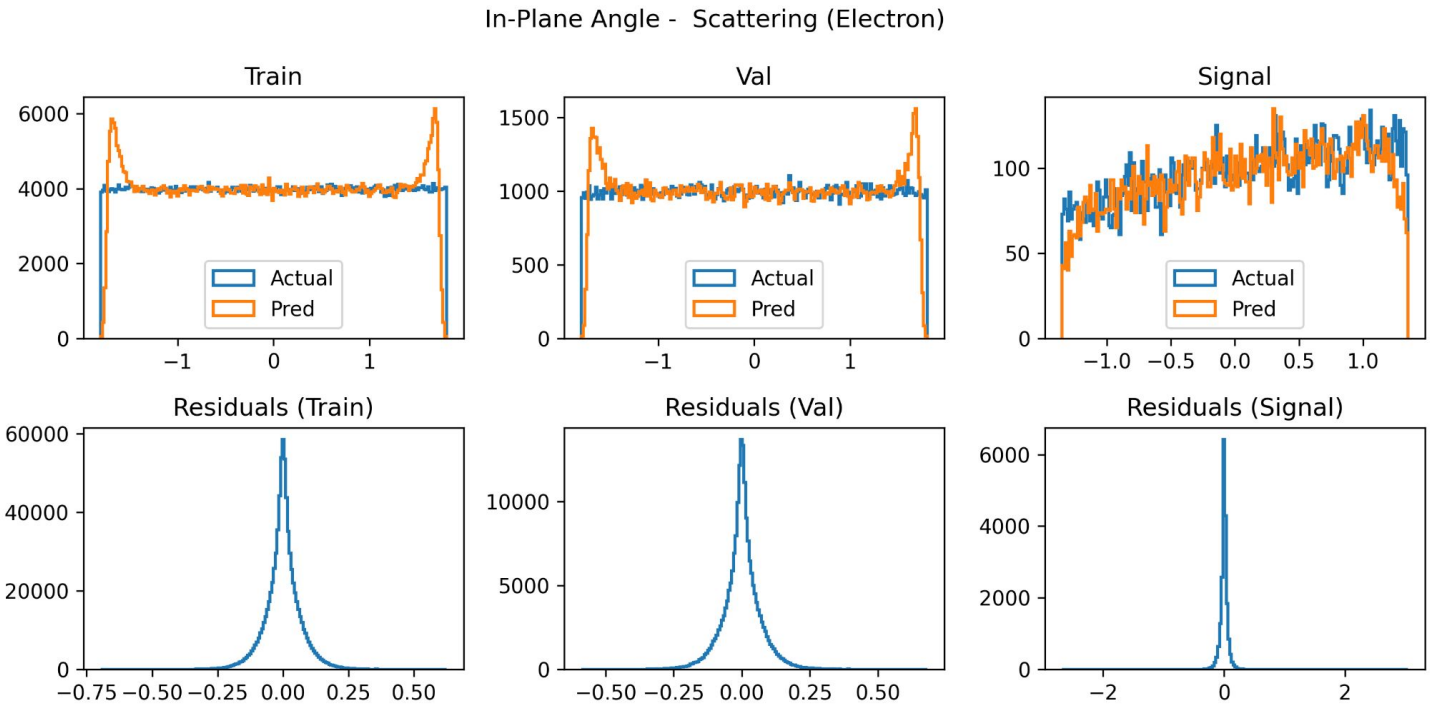
Result:

- Momentum



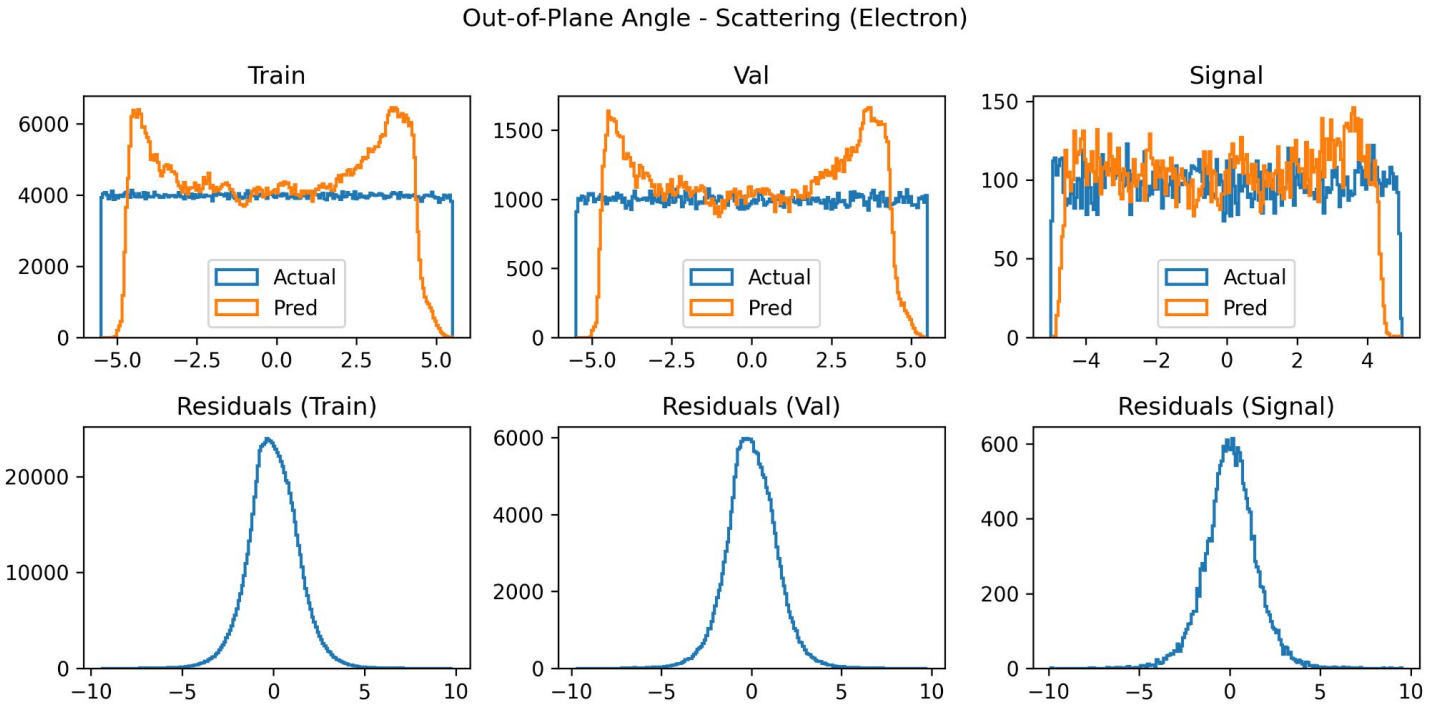
Result:

- In-plane angle



Result:

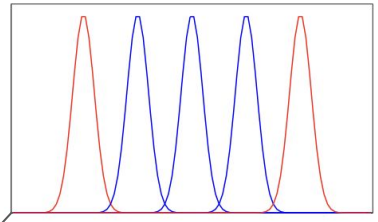
- Out-of-plane angle

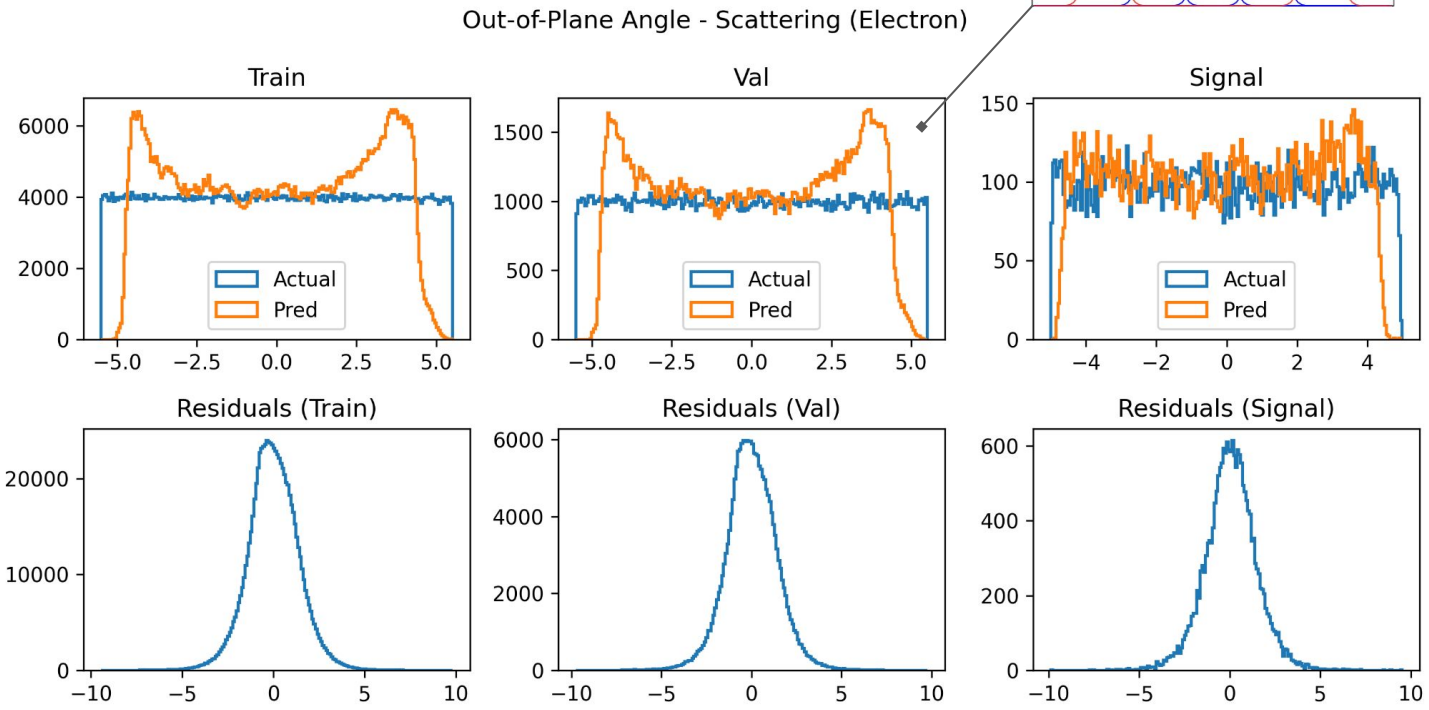


Result:

- Out-of-plane angle

Disagreement at edges is caused by smearing of data

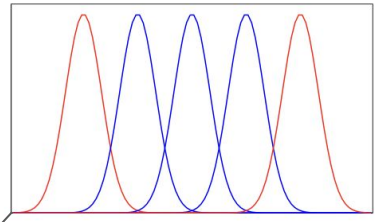


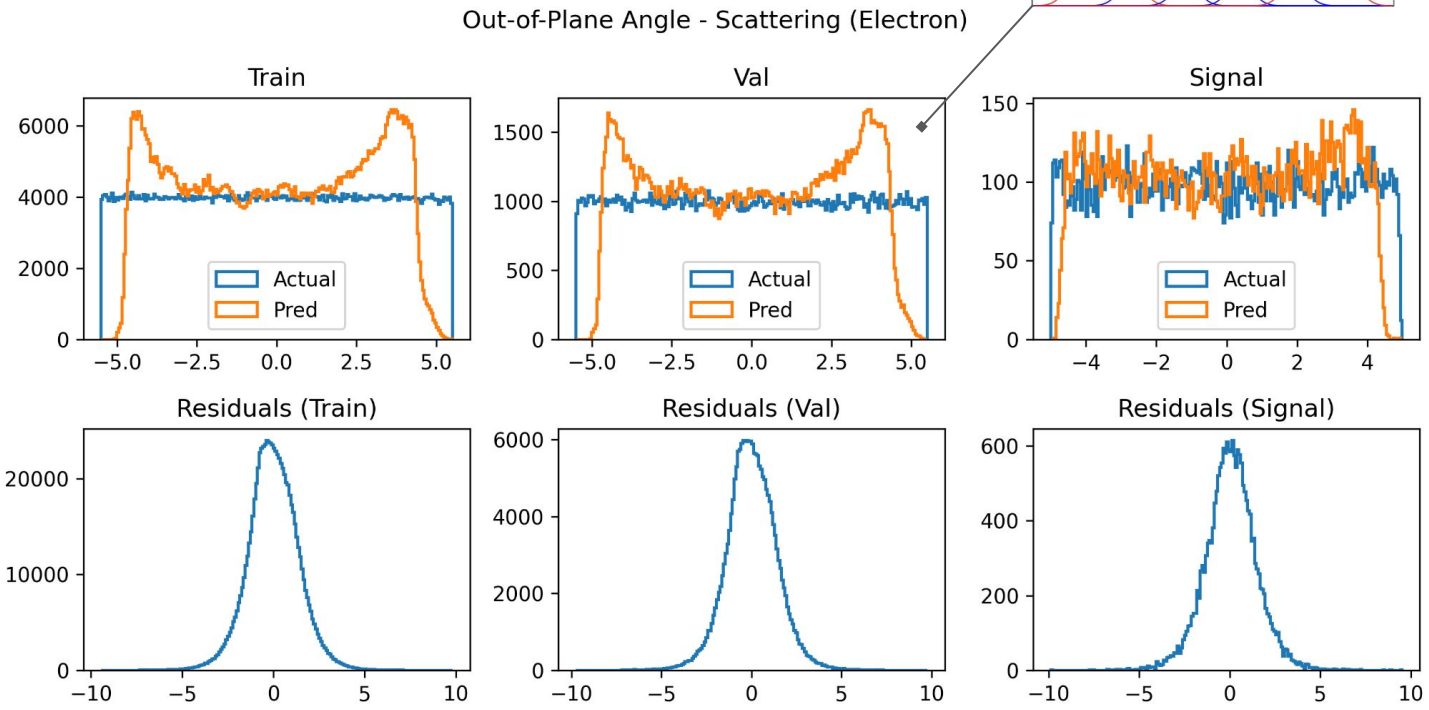


Result:

- Out-of-plane angle

Disagreement at edges is caused by smearing of data

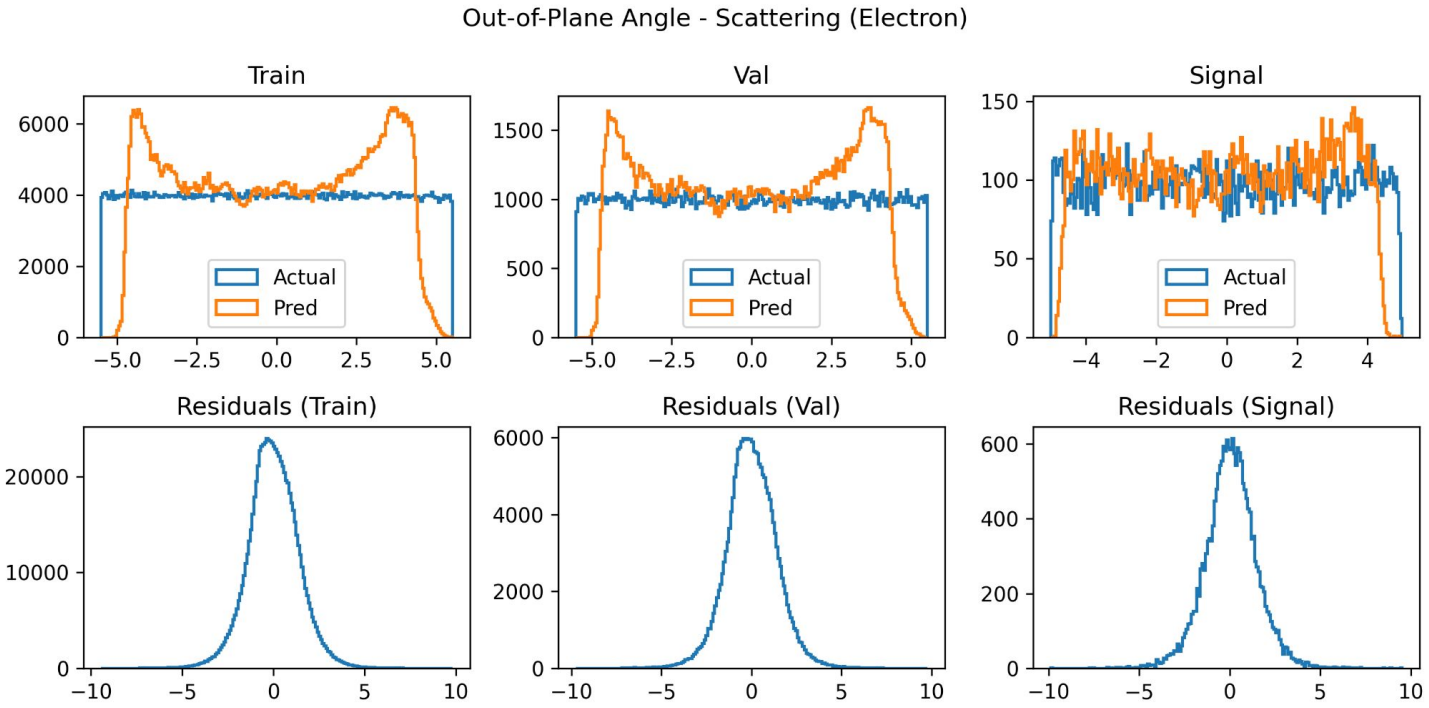




Result:

- Out-of-plane angle

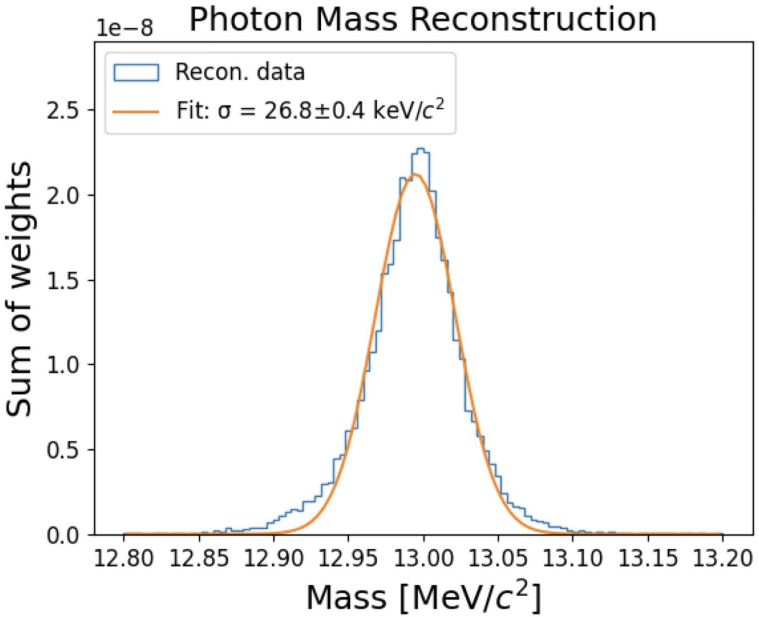
Increase data range in simulation to be wider than signal range so the signal is not affected by the “cat ear”



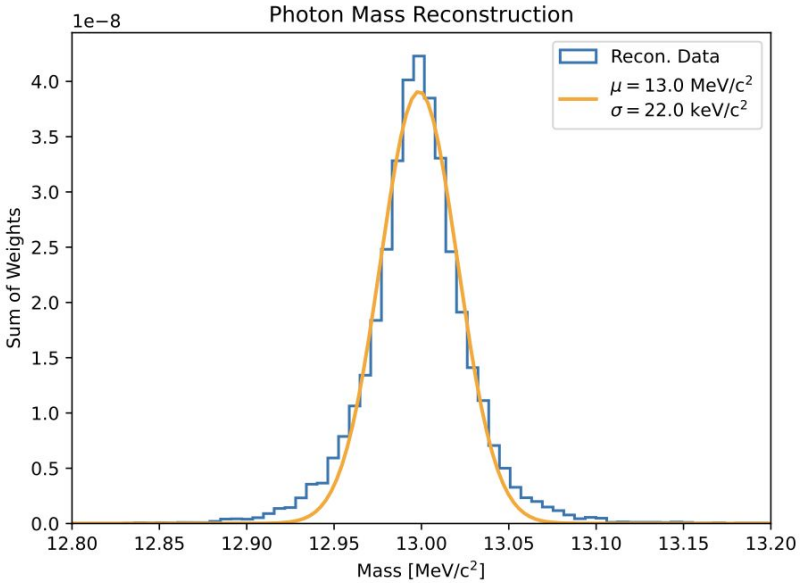
Result:

- Reconstructed mass

Polynomial fit



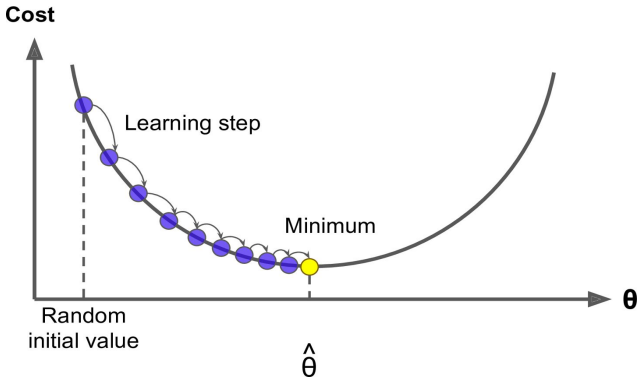
XGBoost



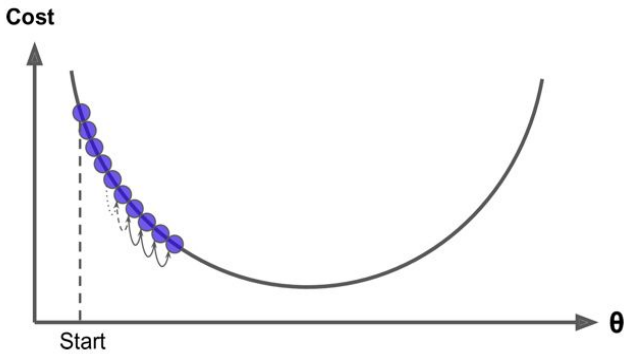
Backups

Gradient descent

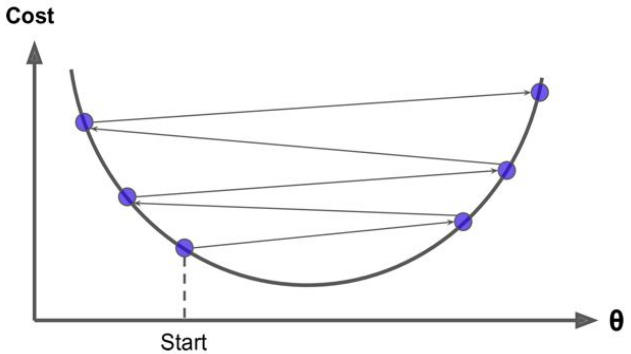
https://uc-r.github.io/gbm_regression



$$r_{im} = -\left[\frac{\partial L(y_i,\ F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad for\ i = 1,\ldots,n$$



a) too small



a) too big