INTRODUCTION TO MACHINE LEARNING

Savannah Thais

CODAS-HEP 2022

08/02/2022

A Quick Intro

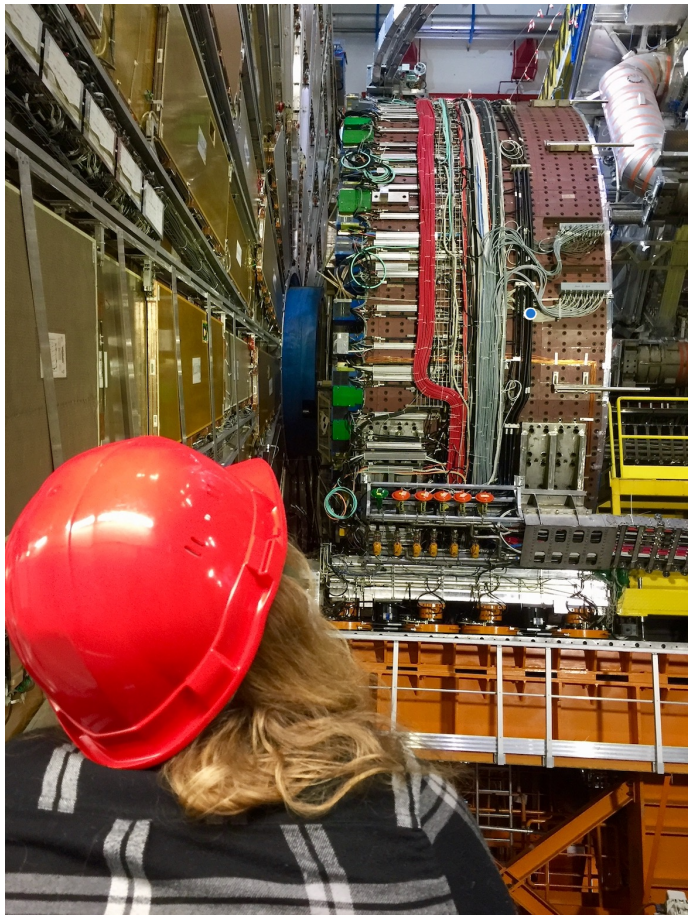
Physics and Math Undergrad (UChicago), PhD in Particle Physics (Yale), ML for Physics Postdoc (Princeton), Incoming Research Scientist at Columbia Data Science Institute

Research in several directions:

- ML for particle reconstruction
- Physics informed machine learning
- Algorithmic interpretability
- Regulation of emerging technology
- 'ML for social good'/community based data advocacy

This 'course':

- Theory-based introduction to key ML methods relevant to (particle) physics
- Demonstration notebooks (applications in physics and beyond)
- Hands-on exercises
- **Goal:** get a broad basis so you know what to look for to dive deeper, a coding starting point for your own projects



Outline for This Session

- **What is ML and What Can It Do?**
- **Supervised Learning: Training**
- **Some 'Simple' ML Models**
- **A Super Quick Preview of Cool Particle Physics Applications**
- **Coding Exercises**
  - Intro to ML libraries
  - Linear regression for auto fuel consumption
  - BDTs for Higgs classification

MACHINE LEARNING

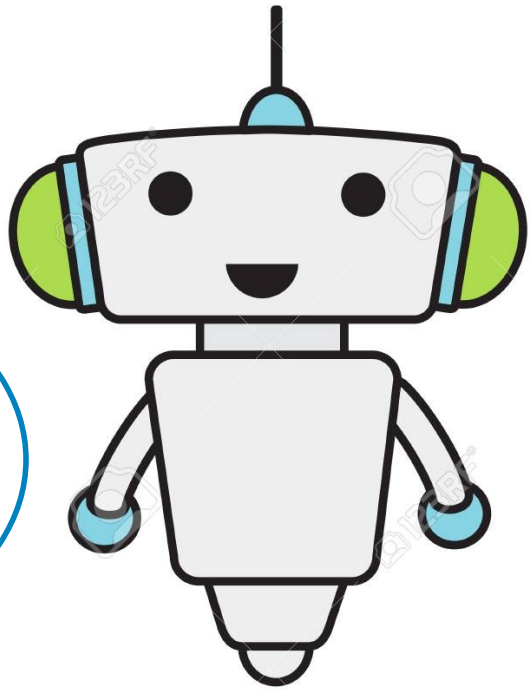
A Conceptual Overview

Killer Robots! Or, What is Machine Learning?

A field of study that gives the ability to the computer to learn without being explicitly programed

Algorithms that improve automatically through experience

Computer programs that can access data and use it to learn for themselves



Using statistics to find patterns in large datasets without the patterns being explicitly stated

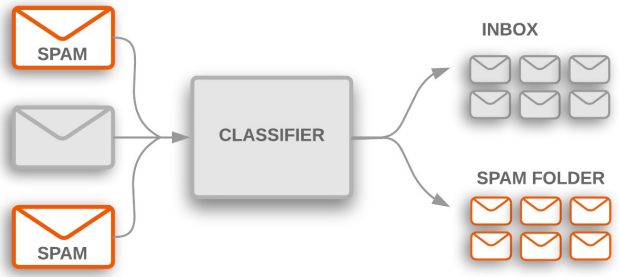
Teaching a computer system how to make accurate predictions when fed data

ML *can* be a great tool to enable scientific (and other) research, but it doesn't know what what it doesn't know

What are Problems for ML?

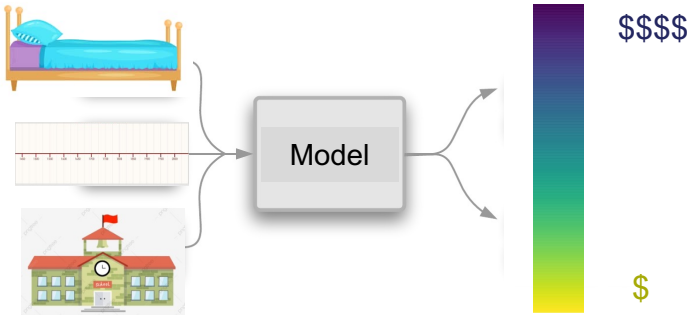
Classification:

Predict a class label for an input



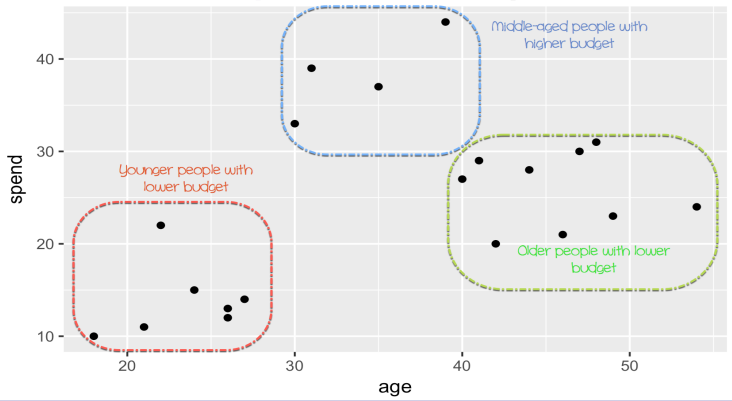
Regression:

Predict a continuous variable



Clustering:

Group similar inputs



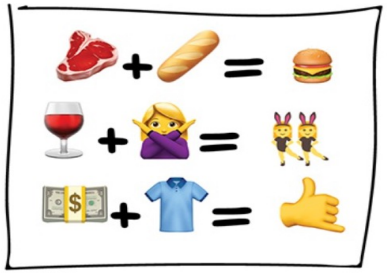
Generation:

Construct new data within pattern

What are Problems for ML?

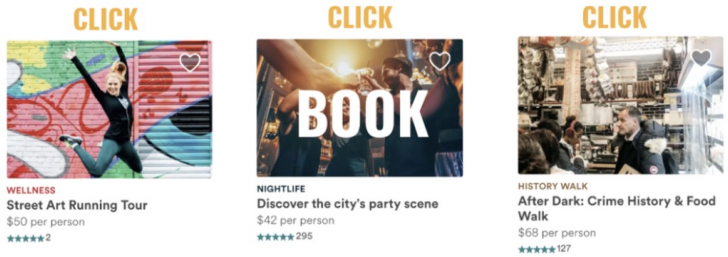
Association Rules:

Identify common patterns in data



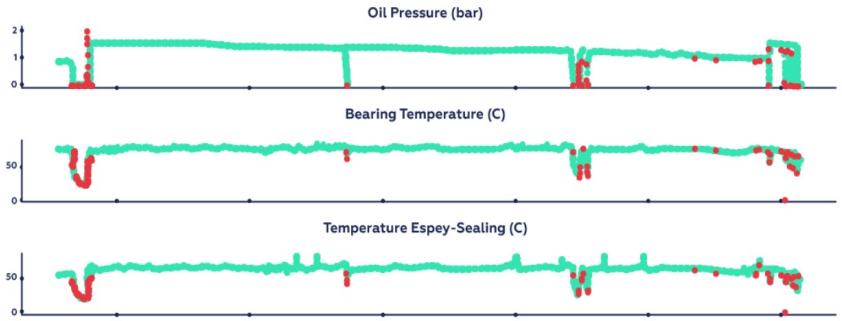
Ranking:

Generate optimal orderings



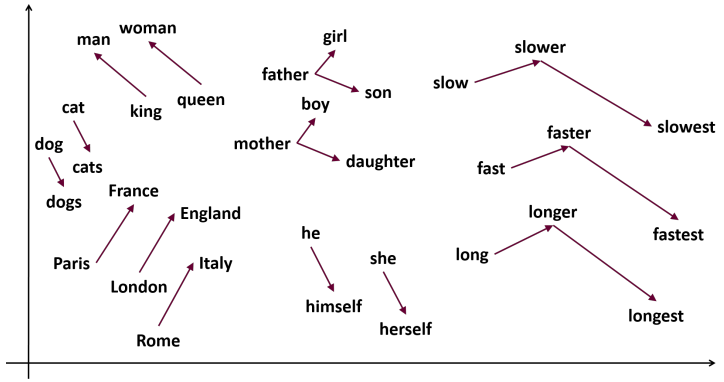
Anomaly Detection:

Identify statistical outliers



Restructuring:

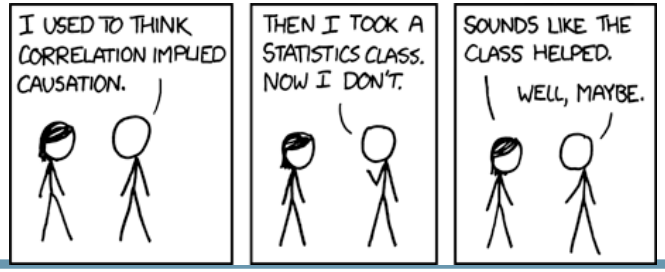
Transform data representations



What are NOT Problems for ML?*

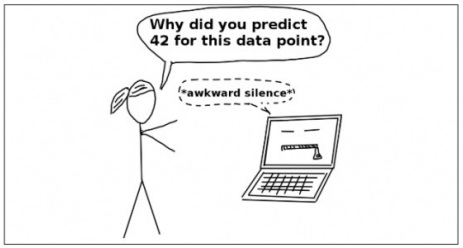
Causation:

Models learn correlations, but can't infer causality or intent



Precise Interpretability:

It's often difficult to understand what a model is learning



Context:

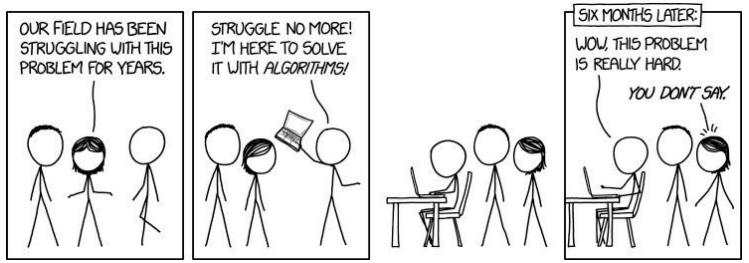
Models are incapable of non-mathematical reasoning

Keaton Patti ✓
@KeatonPatti

I forced a bot to watch over 1,000 episodes of Jerry Springer and then asked it to write an episode of its own. Here is the first page.

Data Limitations:

Models can't fix problems in data or learn without examples

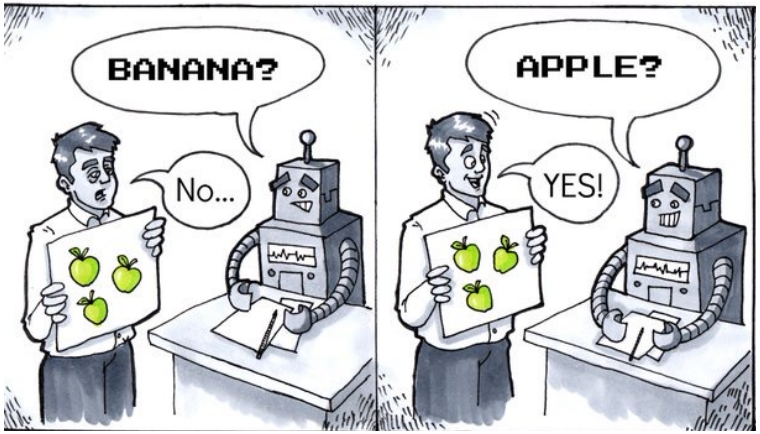


Supervision Required?

Supervised Learning

- Model is provided with labels
- Algorithm learns relationship* between features and labels during training

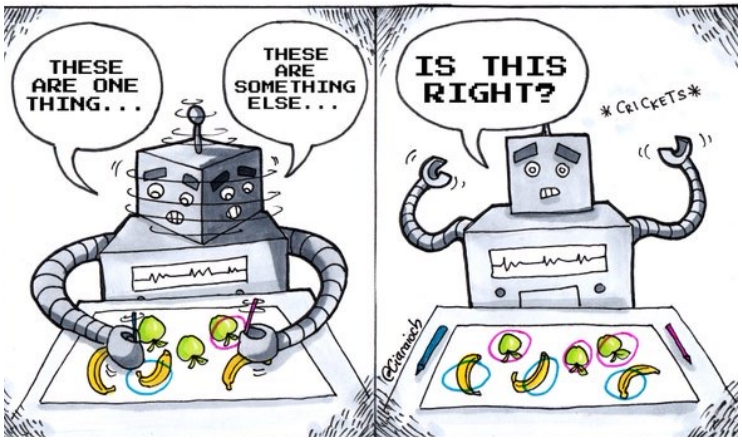
*if that relationship can be expressed as a mathematical function



Un-supervised Learning

- Model takes unlabeled or unclassified data
- Algorithm learns patterns* or groupings in the data during training

*but no promises that those patterns are useful!

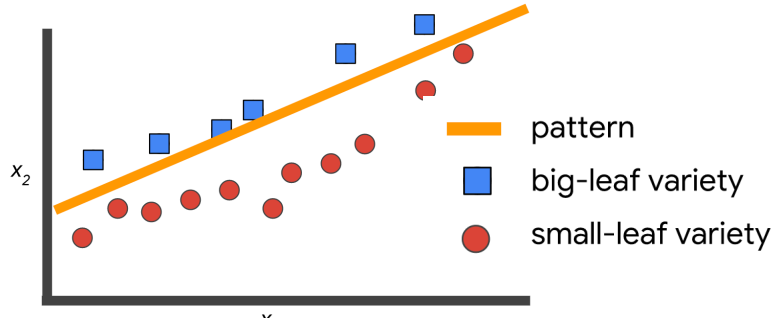


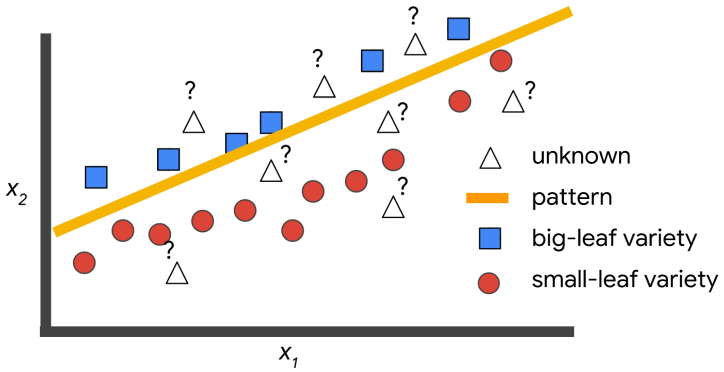
Bonus: Reinforcement Learning – models are rewarded for meeting goals

Supervision Required?

Supervised Learning

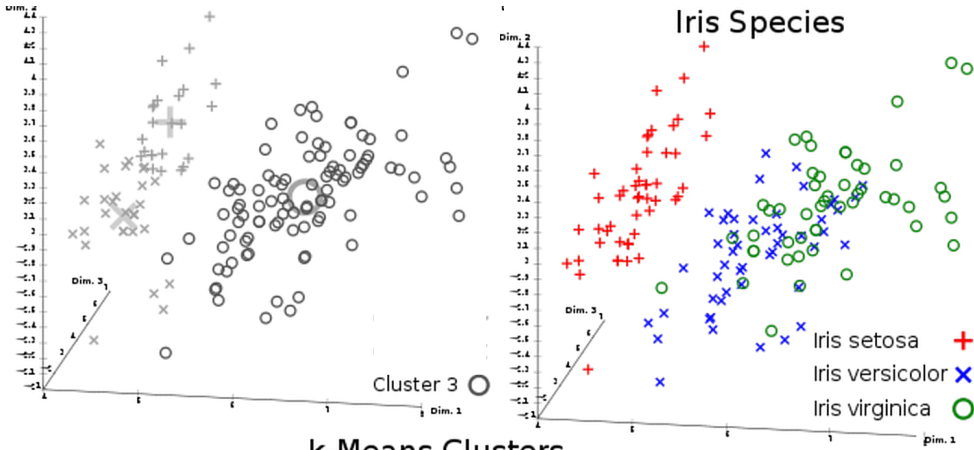
Example: distinguishing between two known classes (learn a decision boundary)

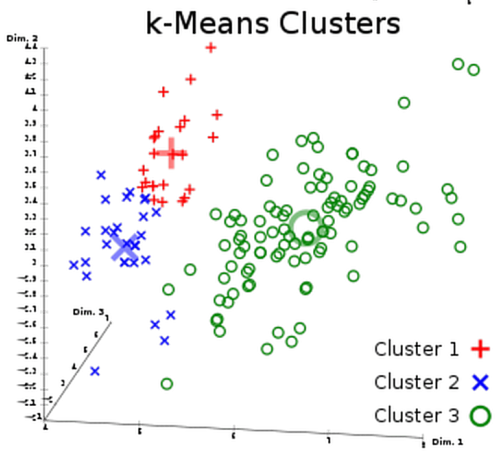




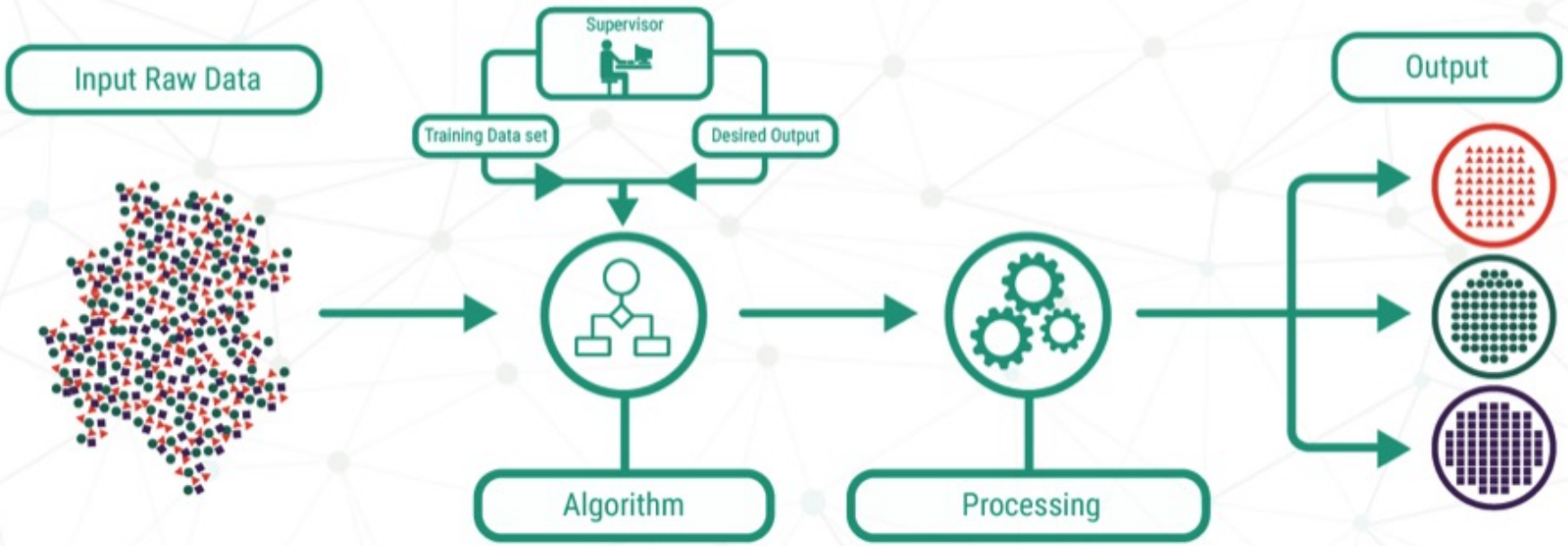
Un-supervised Learning

Example: grouping data into 3 classes (learn relational structure)



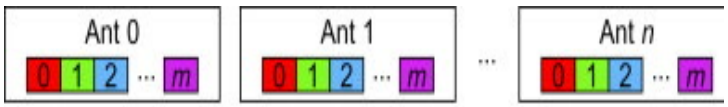


Supervised Algorithms



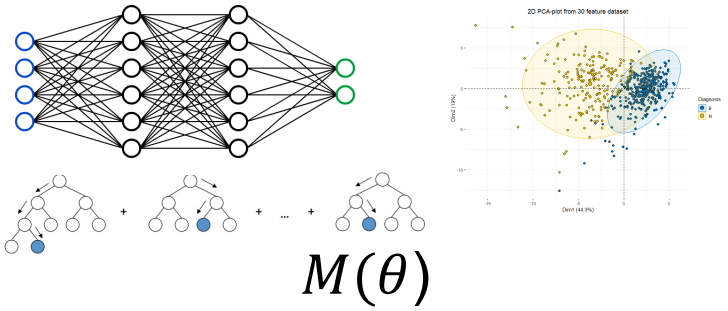
How Do You Actually Do This?

Training data



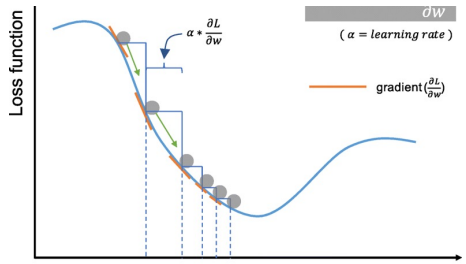
$$D_{train}=(\vec{x}, y)$$

Choose a Model



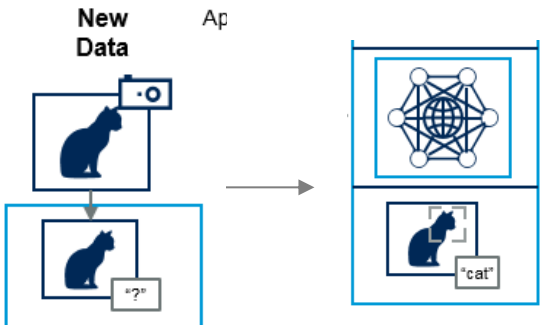
$$M(\theta)$$

Optimize Parameters

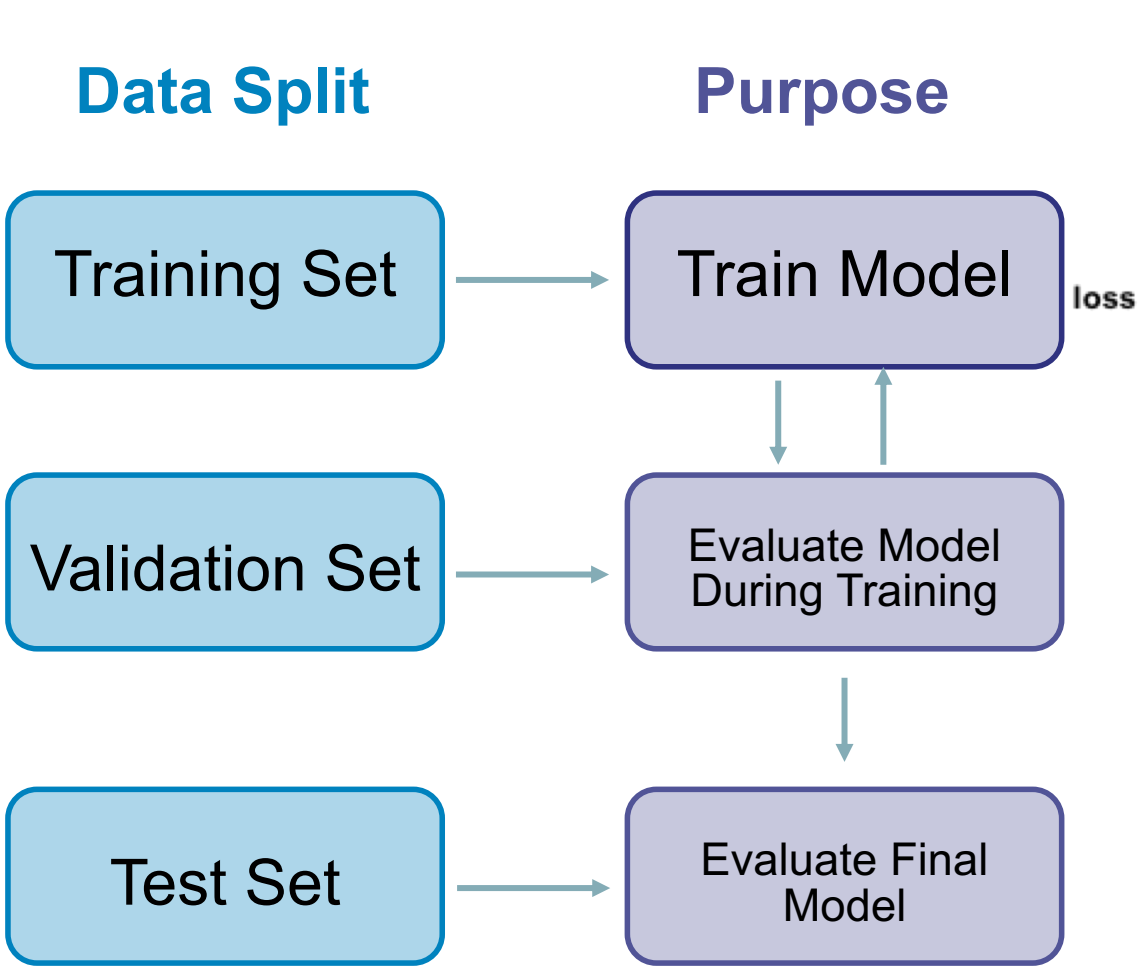


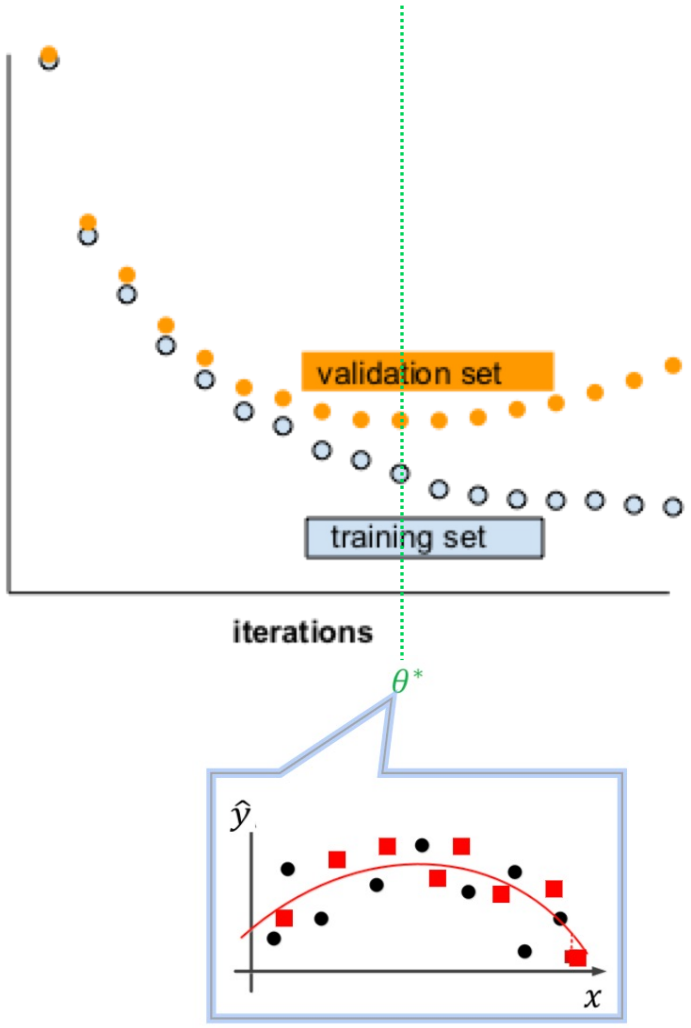
Loss function: L

Make an Inference!



Training a Model



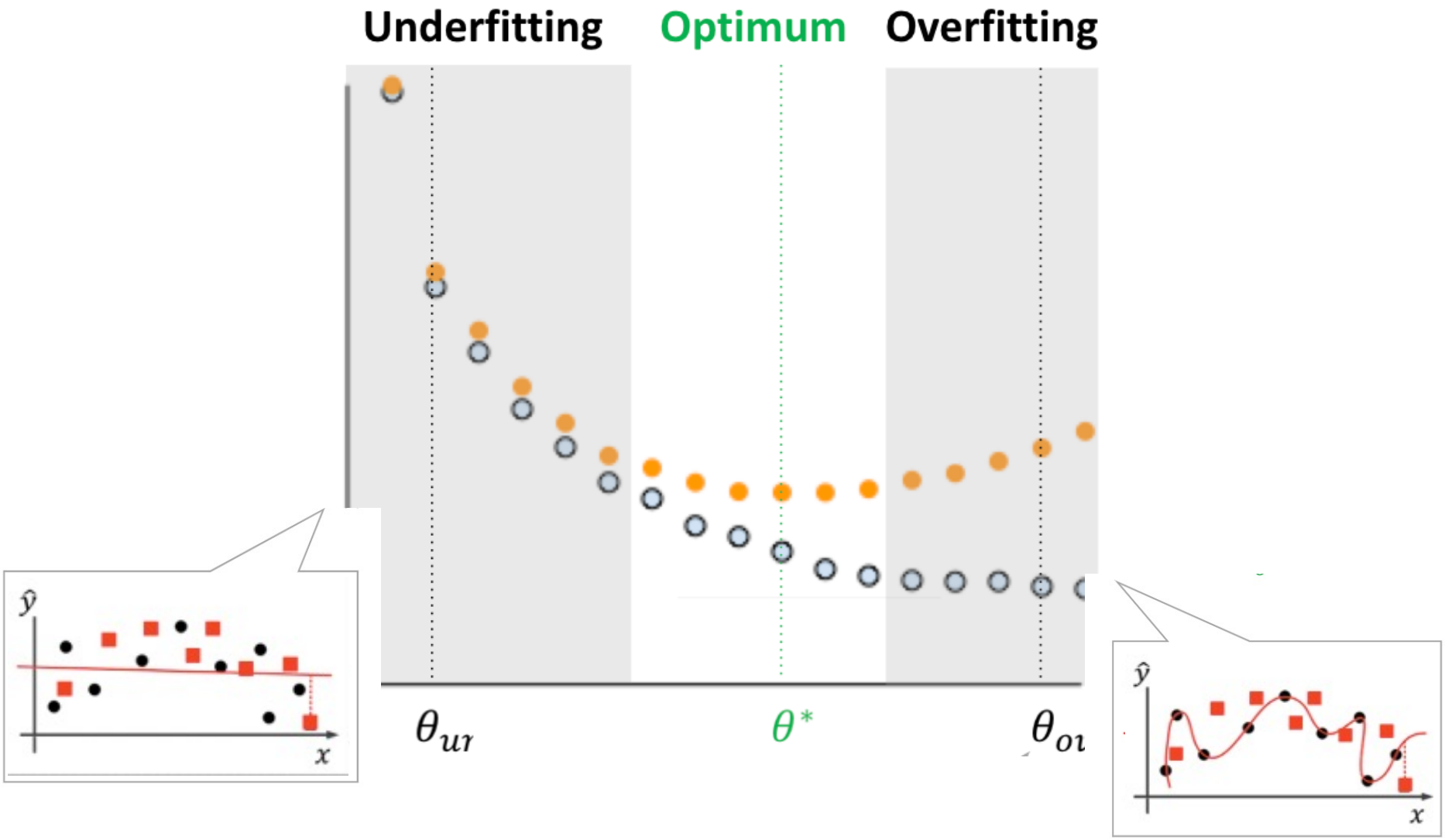


Over and Under Fitting

- Didn't learn general patterns in data
- High bias
- Low variance
- Inaccurate inference

- Learned specific details of training set
- Low bias
- High variance
- Inaccurate inference

Over and Under Fitting



Choosing a Cost Function + Evaluation

Loss Functions

- Common choices:
  - Mean Squared Error

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

  - Hinge Loss

$$SVMLoss = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + 1)$$

  - Cross Entropy

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

- **Considerations**:
  - Type of model
  - Presence of outliers in dataset
  - Sensitive outcomes
  - Training behavior

further reading

Evaluation Metrics

- Common choices:
  - Accuracy $\frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$
  - Confusion matrix
  - Sensitivity $\frac{TruePositive}{FalseNegative + TruePositive}$
  - Specificity $\frac{TrueNegative}{TrueNegative + FalsePositive}$
  - False positive rate $\frac{FalsePositive}{TrueNegative + FalsePositive}$
  - Precision $\frac{TruePositives}{TruePositives + FalsePositives}$
  - Recall $\frac{TruePositives}{TruePositives + FalseNegatives}$
  - F1 Score $F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$
- **Considerations**:
  - What type of outcomes matter?
  - Typically should report multiple

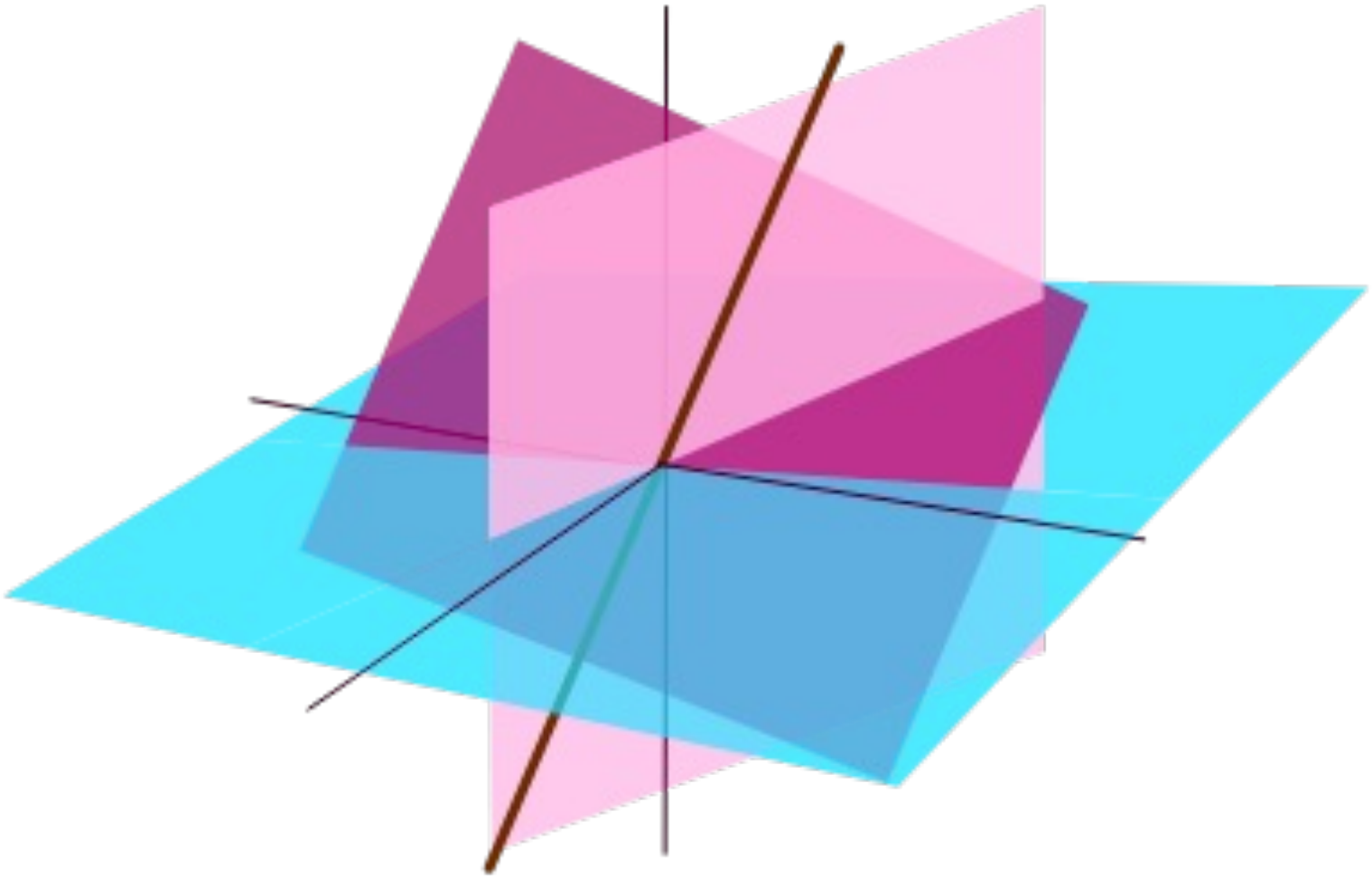
further reading

Question Time!

Can you think of an example of ML in your daily life?:

- Do you think it's a supervised or un-supervised model?
  - What data could have been used to train it?
  - What might have been the learning objective?
- Can you imagine some important training considerations?

A Quick Linear Algebra Refresher



Matrix Basics

$$\begin{bmatrix} 460 \\ 430 \\ 480 \\ \dots \\ \dots \end{bmatrix}$$

Vectors

- A single column of data
- Can represent a single variable, training labels, model parameters

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$$ Where $a_{ij} \in R$

Matrices

- m*n matrix: m columns of n elements
- Often represents training data, model parameters, convolutions

Matrix Multiplication

- Dot product of rows and columns
- Rows of first matrix multiplied by columns of second matrix
- Allows dimension transformations

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} * \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 9 & 12 \end{pmatrix} = \begin{pmatrix} 52 & 64 \\ 127 & 154 \end{pmatrix}$$

Where,

[1, 2, 3] * [8, 10, 12] = 1 * 8 + 2 * 10 + 3 * 12 = 64

source

Matrix Types + Operations

$$A = \begin{pmatrix} 1 & 4 \\ -2 & 3 \end{pmatrix} \quad A^T = \begin{pmatrix} 1 & -2 \\ 4 & 3 \end{pmatrix}$$

Matrix Transpose

B is transpose of A if $a_{ij} = b_{ji}$

Matrix Inverse

B is inverse of A if AB=BA

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 4 & 5 \\ 6 & 7 & 7 \end{pmatrix} \quad B = \begin{pmatrix} -7 & -7 & 6 \\ 2 & 1 & -1 \\ 4 & 5 & -4 \end{pmatrix}$$

$$AA^T = I = A^TA$$

Where, $A^{-1} = A^T$

Orthogonal Matrix

Square matrix with columns of unit length

Diagonal Matrix

$a_{ij} = 0$ for i!=j, $a_{ij}! = 0$ for i=j

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Matrices + Models

Matrix multiplication for model inference

No. of Rooms (x0)	Size (Square feet) (x1)	Year Built (x2)	No. of Floors (x3)	Price (y)
2	1434	2010	1	8500
3	1534	2019	2	9600
2	962	1996	3	25880
...	...	...	...	...

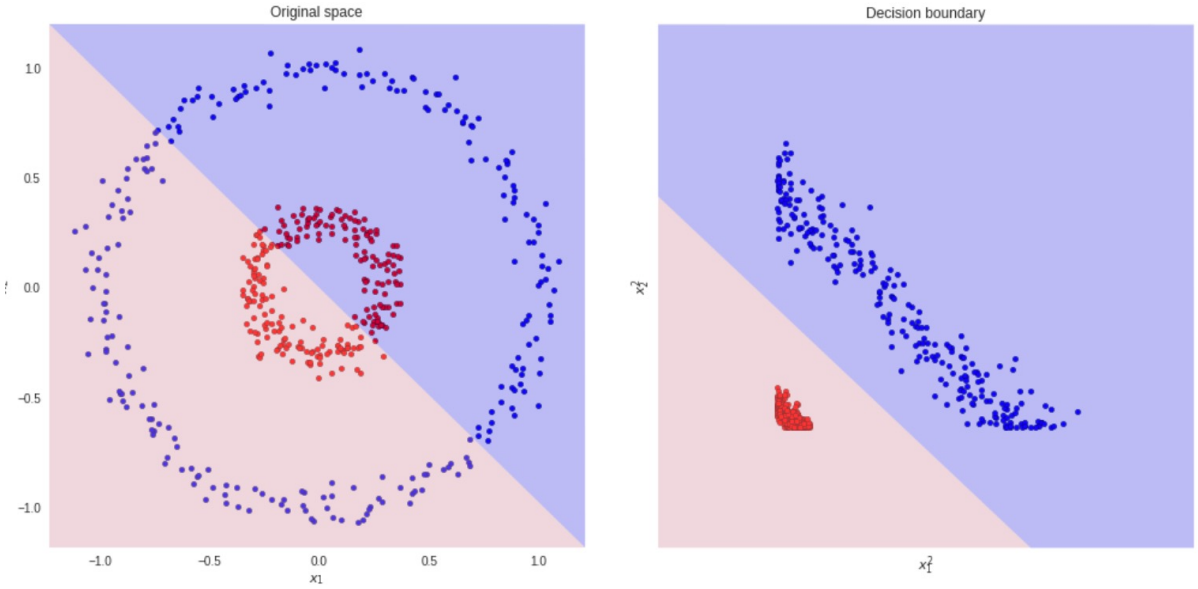
x_i = features of a house

y = target variable

$$h_\theta(x) = \theta_0 x0 + \theta_1 x1 + \dots + \theta_n xn = y'$$

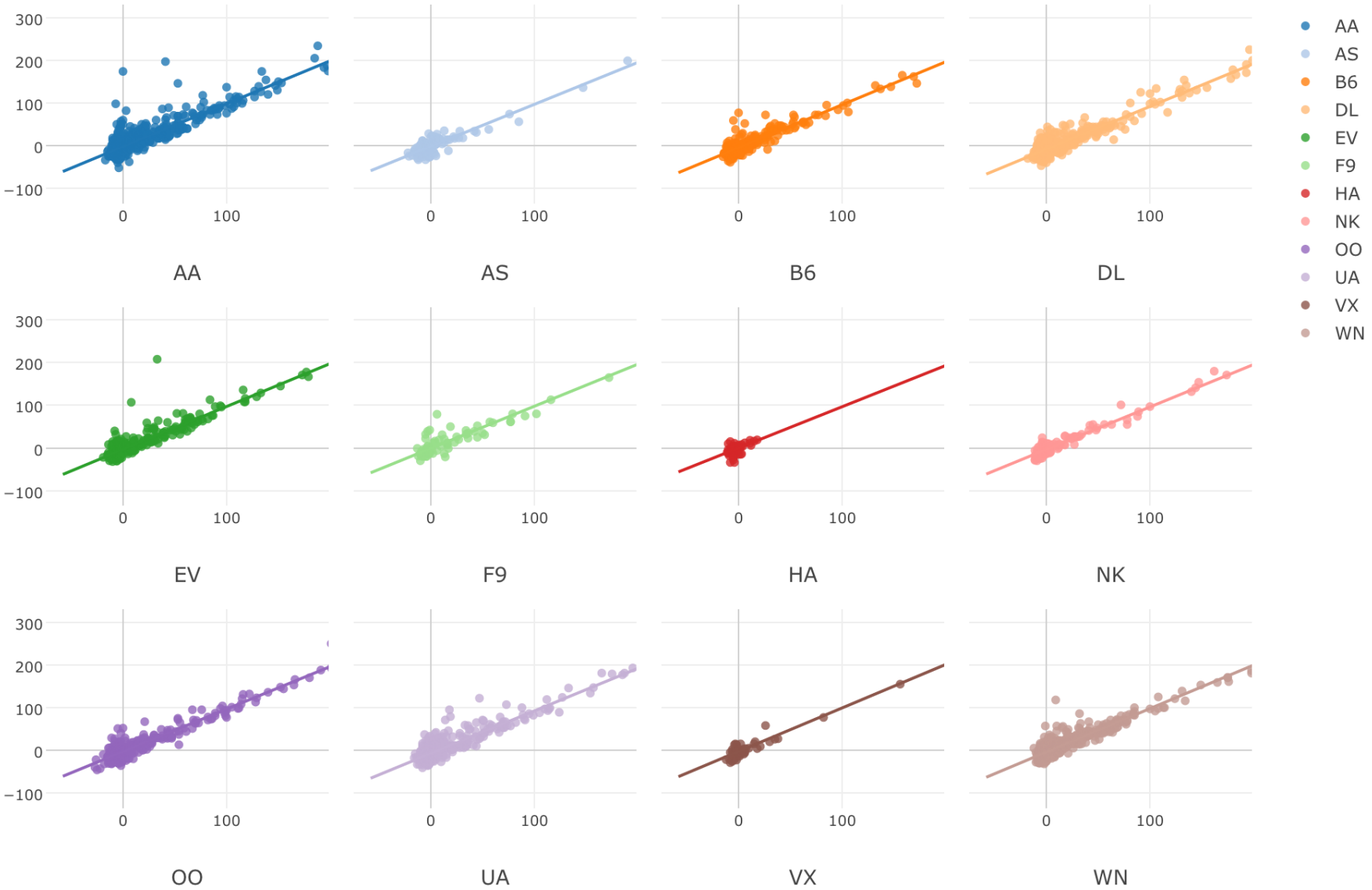
$$x = \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \end{pmatrix} \quad \& \quad \theta = \begin{pmatrix} \theta0 \\ \theta1 \\ \theta2 \\ \theta3 \end{pmatrix}$$

Matrix multiplication for data transformation



$$y_n = \boldsymbol{W}^T x_n$$

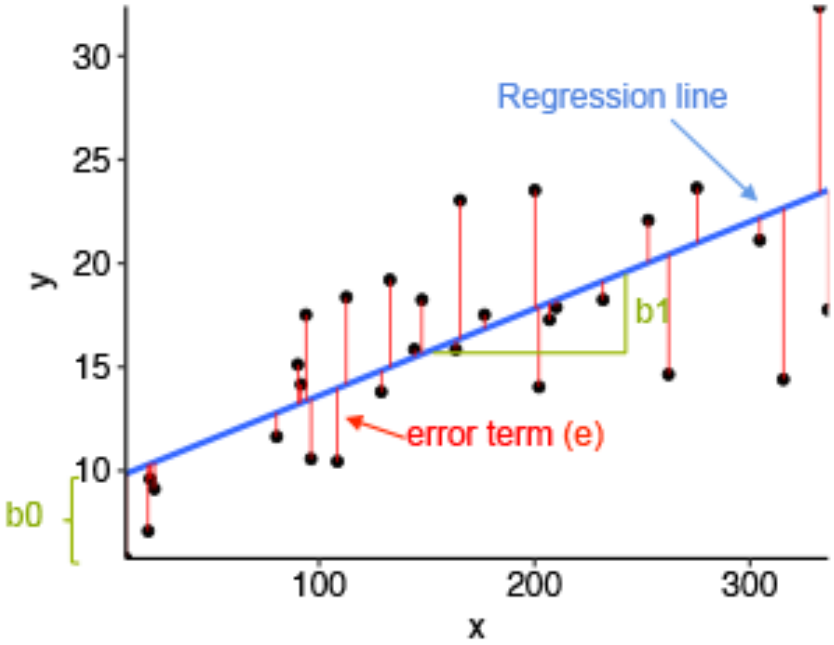
The Simplest ‘ML’ Models



Linear Regression as ML

Fit the set of dependent and independent data of a dataset to a linear line function

Allows prediction of continuous variables



A specific case of linear models

$$y = y(x) \rightarrow y(x_i) = \tilde{y}_i + \epsilon_i = \sum_{j=0}^{n-1} \beta_j x_i^j + \epsilon_i,$$

$$\begin{aligned}
y_0 &= \beta_0 + \beta_1 x_0^1 + \beta_2 x_0^2 + \cdots + \beta_{n-1} x_0^{n-1} + \epsilon_0 \\
y_1 &= \beta_0 + \beta_1 x_1^1 + \beta_2 x_1^2 + \cdots + \beta_{n-1} x_1^{n-1} + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2^1 + \beta_2 x_2^2 + \cdots + \beta_{n-1} x_2^{n-1} + \epsilon_2 \\
&\cdots\cdots \\
y_{n-1} &= \beta_0 + \beta_1 x_{n-1}^1 + \beta_2 x_{n-1}^2 + \cdots + \beta_{n-1} x_{n-1}^{n-1} + \epsilon_{n-1}.
\end{aligned}$$

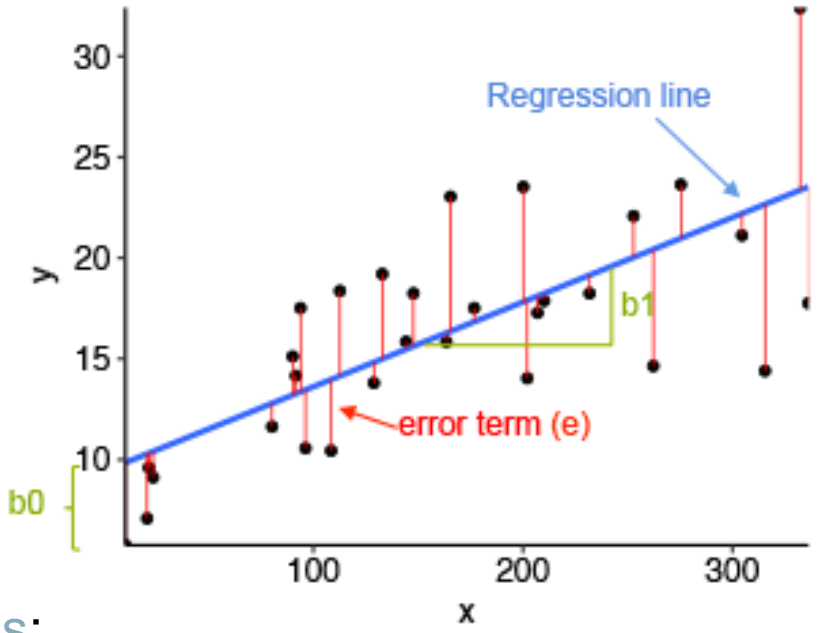
Learn parameters of β and ε

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}.$$

Linear Regression as ML

Fit the set of dependent and independent data of a dataset to a linear line function

Allows prediction of continuous variables



- Training: Often Ordinary Least Squares
- Inference: use parameters of new data to calculate regression value
- Loss Function: $\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$

further reading

Pros:

- Easy to implement and train
- Fast inference
- Interpretable

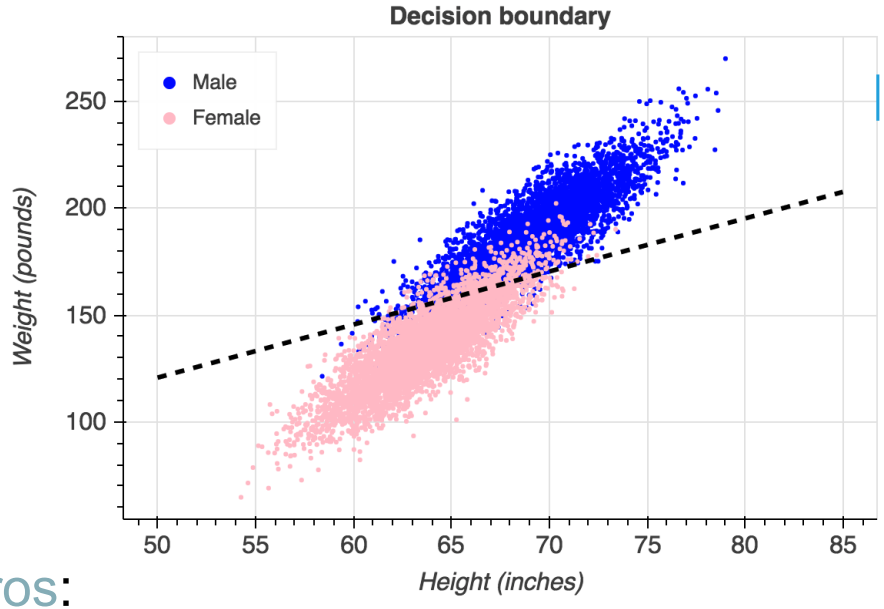
Cons:

- Assumes linear relationship
- Overfits or doesn't converge with correlated features

Logistic Regression

Estimate the logarithmic probability of a class from linear combination of inputs

Assume probability depends linearly on features: $h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$



- Training: GD to adjust slope parameters
- Inference: select a classification threshold
- Loss Function:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

further reading

Pros:

- Easy to implement and train
- Fast inference
- interpretable

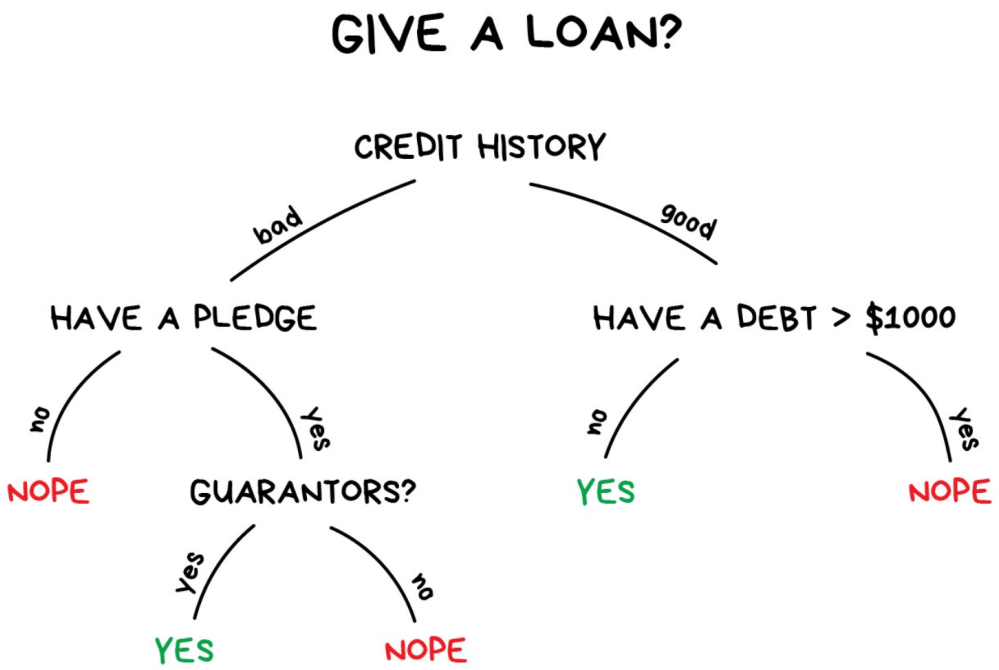
Cons:

- Assumes linear relationship
- Overfits or doesn't converge with correlated features

Decision Trees

Decision Trees

Input data is processed through a series of linear cuts



1. Try a range of cuts on each feature
2. Select cut with lowest cost
3. Repeat 1+2 for all branchings
4. Meet stopping criteria
5. Assign a label to each final leaf

- No global loss function, uses 'greedy local metric'
- Pros: interpretable, easy to build
- Cons: assumes simple linear relationship, easy to overtrain, high variance

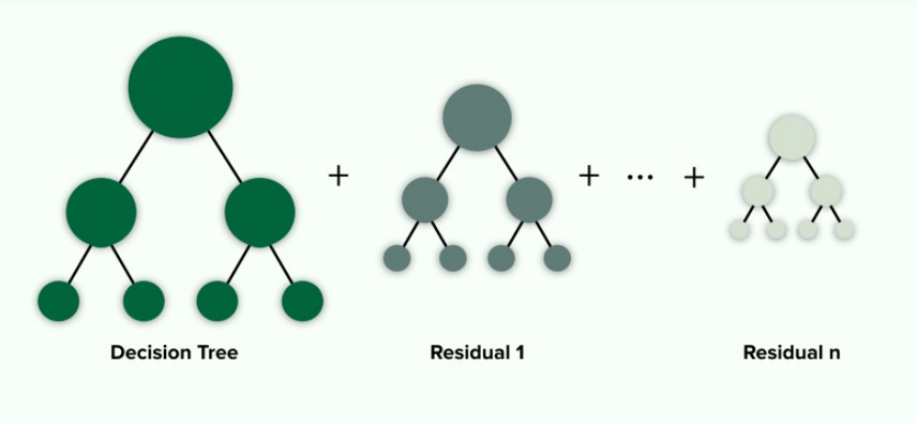
further reading

Forests of Trees

Multiple decisions trees are combined into ensemble classifier

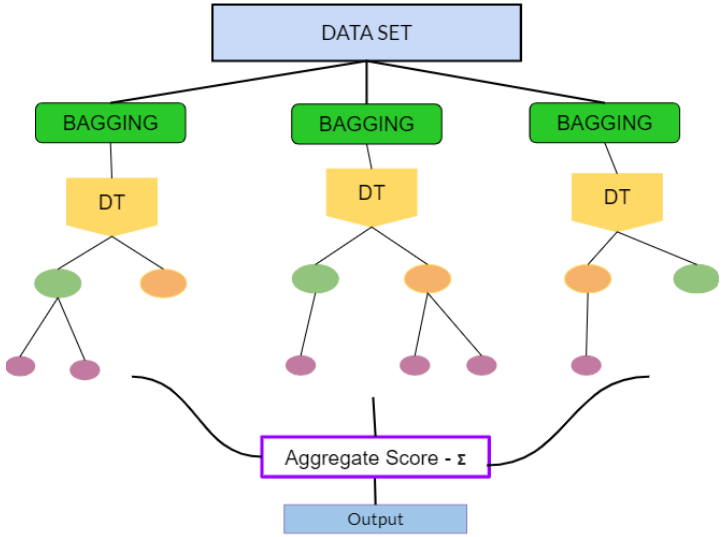
Boosting:

Misclassified data points are given a higher weight in the next tree



Bagging:

Each tree is trained on a random subset of data or input features

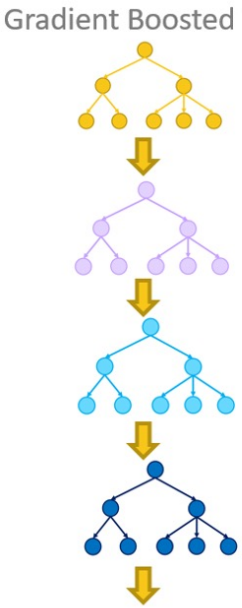


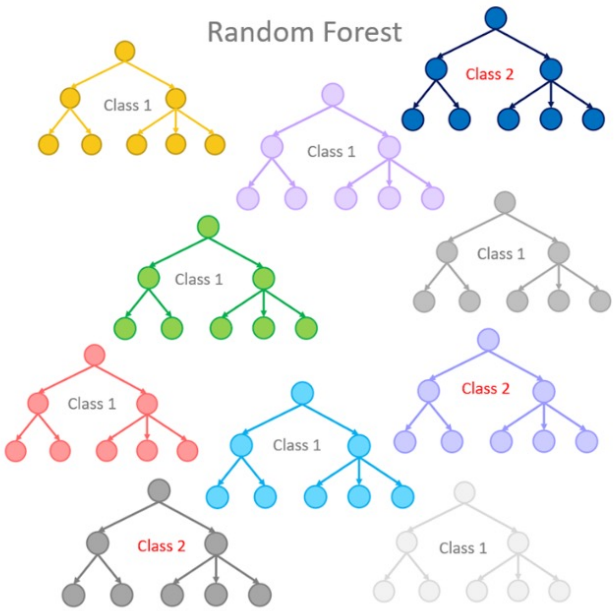
- Final label is typically (weighted) average of label from all trees
- Both methods reduce variance in final decision
- Boosting has a global loss function and can be optimized with gradient descent

Decision Tree Forests Model Card

Multiple decisions trees are combined into ensemble classifier

- Boosting: misclassified data points given a higher weight in next tree
- Bagging: train trees on random subsets of data/subsets of features





- Training: gradient descent (boosting) or greedy local (bagging)
- Classification: assign label to each final leaf in aggregate
- Loss Function:

$$h_m(x) = -\frac{\partial L_{\text{MSE}}}{\partial F} = y - F(x).$$

- Hyperparameters:
  - Depth and pruning
  - Combination function
  - Number of trees

further reading

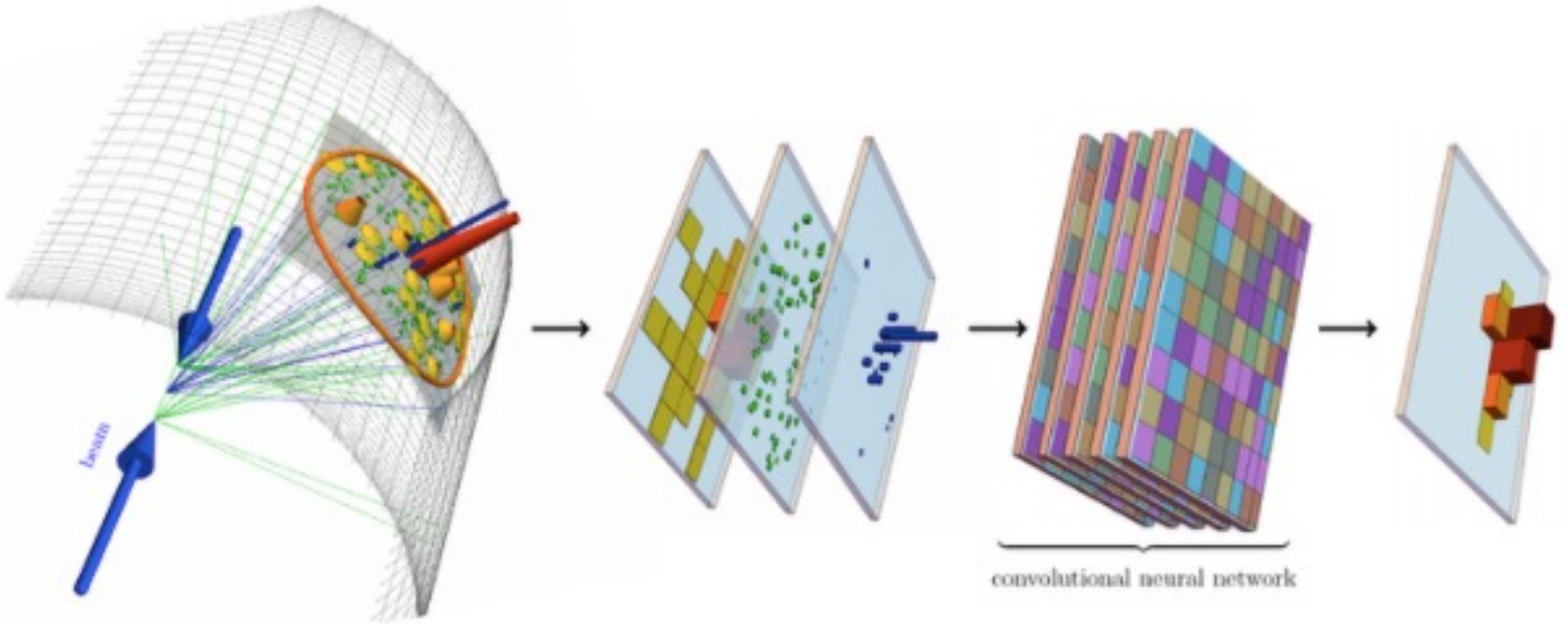
Pros:

- ~Interpretable
- Reduced variance
- Allows higher dimensional data

Cons:

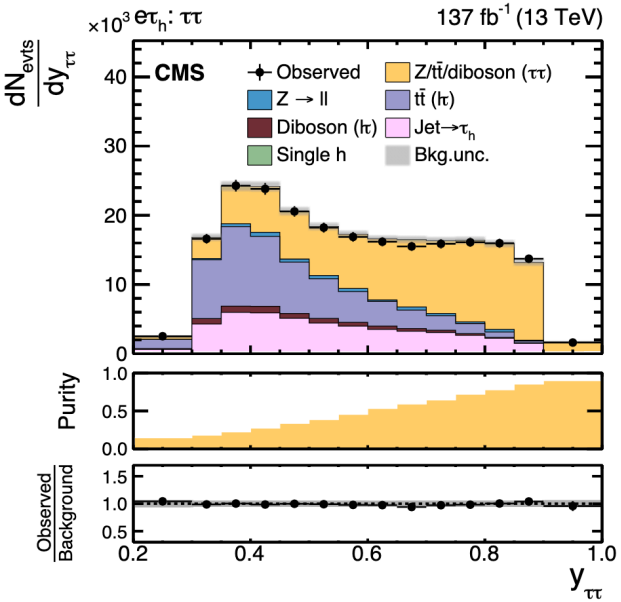
- Less precise regression
- Only allows linear discriminant

Some of the Super Cool Applications of ML in HEP!



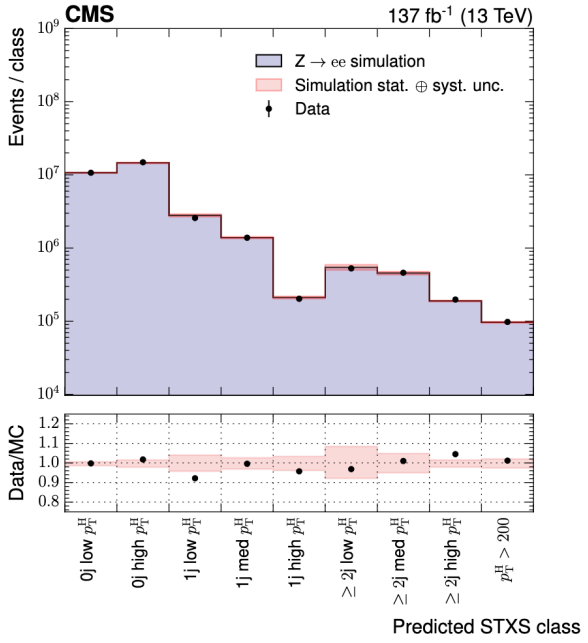
Higgs Event Classification

- Tensorflow NN used for multiclass classification in heavy Higgs decay search
  - Four background and one signal class
  - Multiple networks trained for each final state to allow for range of H mass



CMS Publication

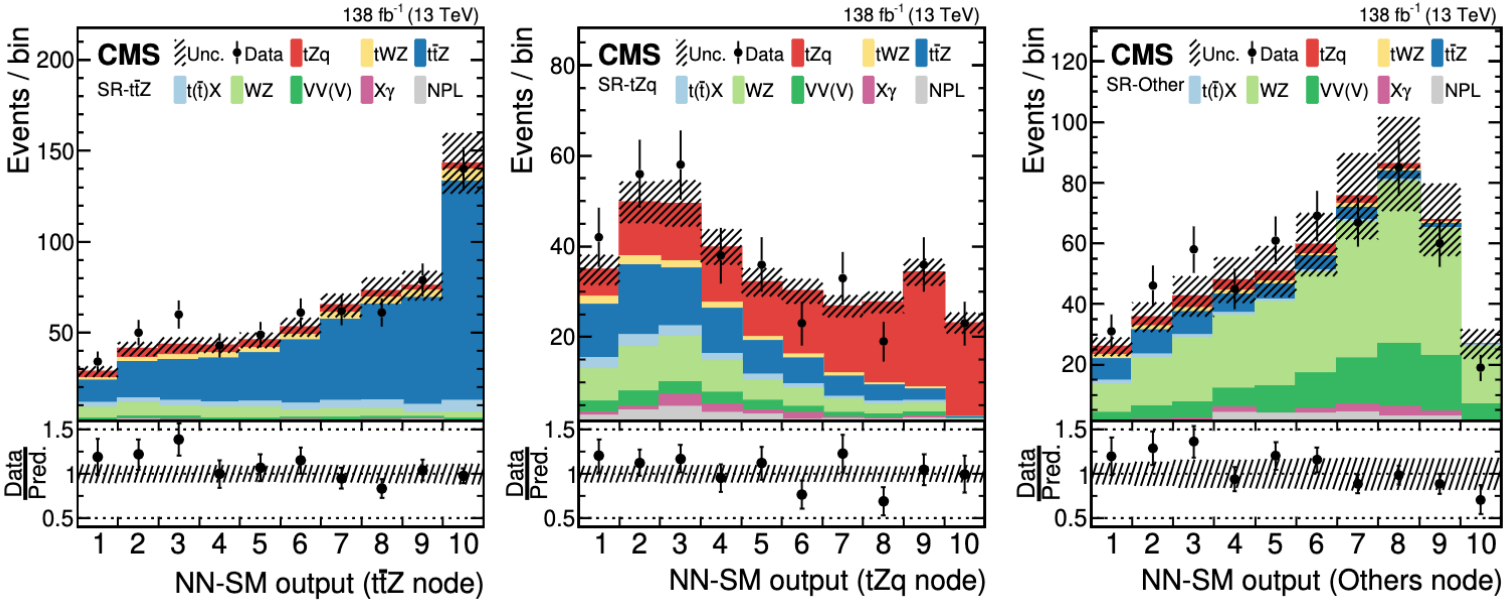
- BDT used to select H→$\gamma\gamma$ events from different production modes
  - Each production mode has a dedicated, optimized BDT
  - Each BDT classifies events into bins based on decay kinematics



CMS Publication

Probing Effective Field Theory Operators

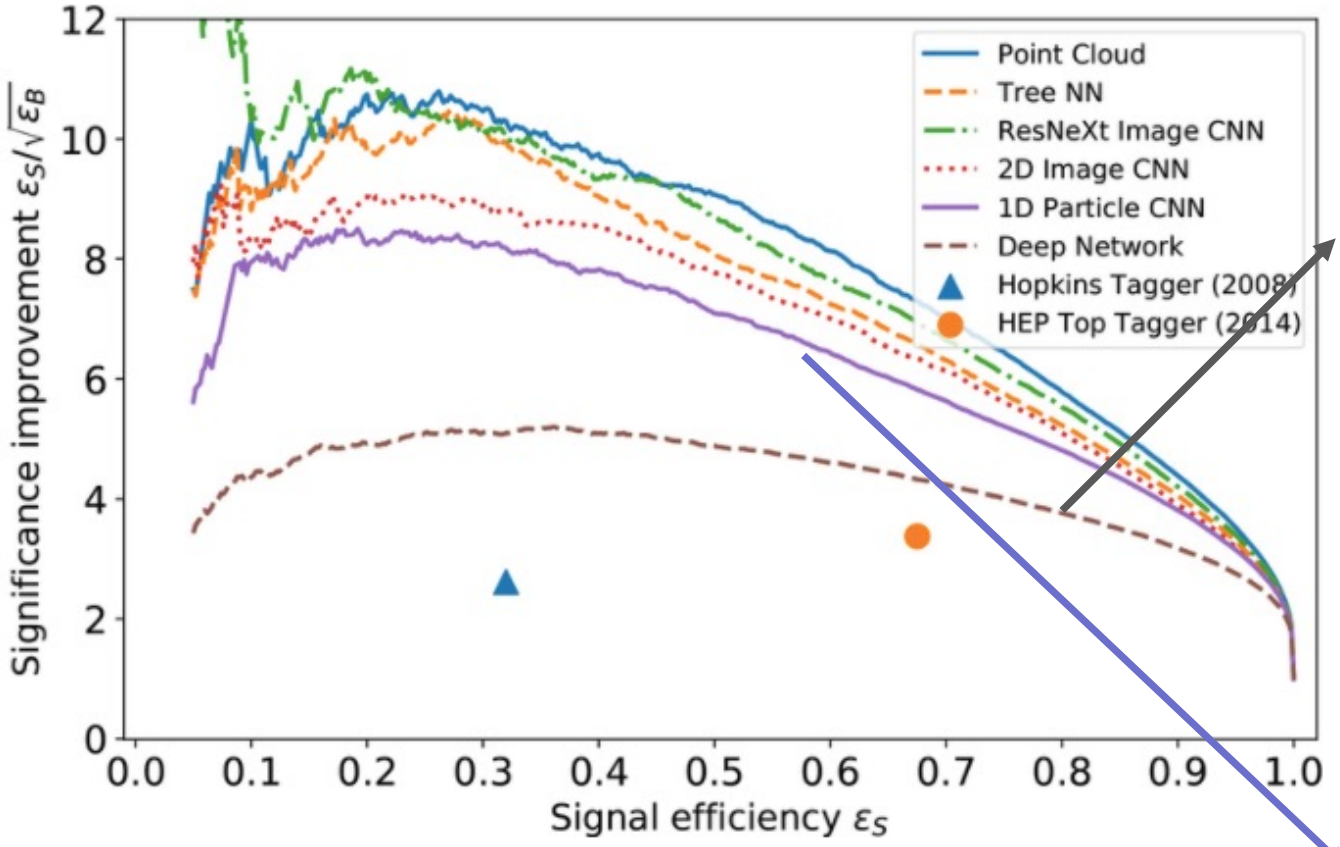
- Looking for new physics in the top and Z boson couplings
  - Many BSM models predict modifications to this coupling
  - EFT extends the SM Lagrangian by introducing a higher-order operator with interaction strength described by a Wilson Coefficient (WC)
- Tensorflow NN used in two stages in analysis
  - Multiclass NN used to separate tZq and ttZ events from background
  - Three binary NNs used to separate SM events from non-zero WC events



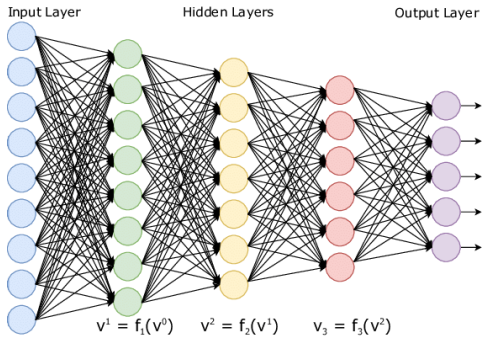
CMS Publication

Jet Tagging (and Reconstruction/Particle ID)

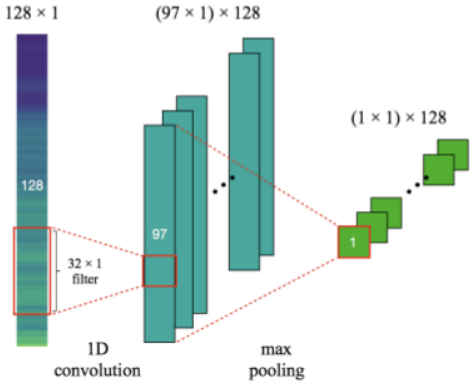
Comparison of ML-based top-taggers



Deep Neural Network using constituent particle momenta

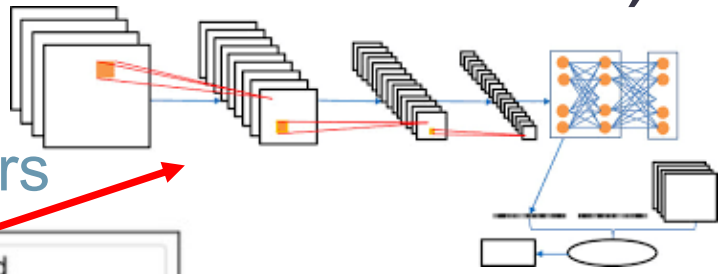


1D Convolutional Network using constituent particle momenta

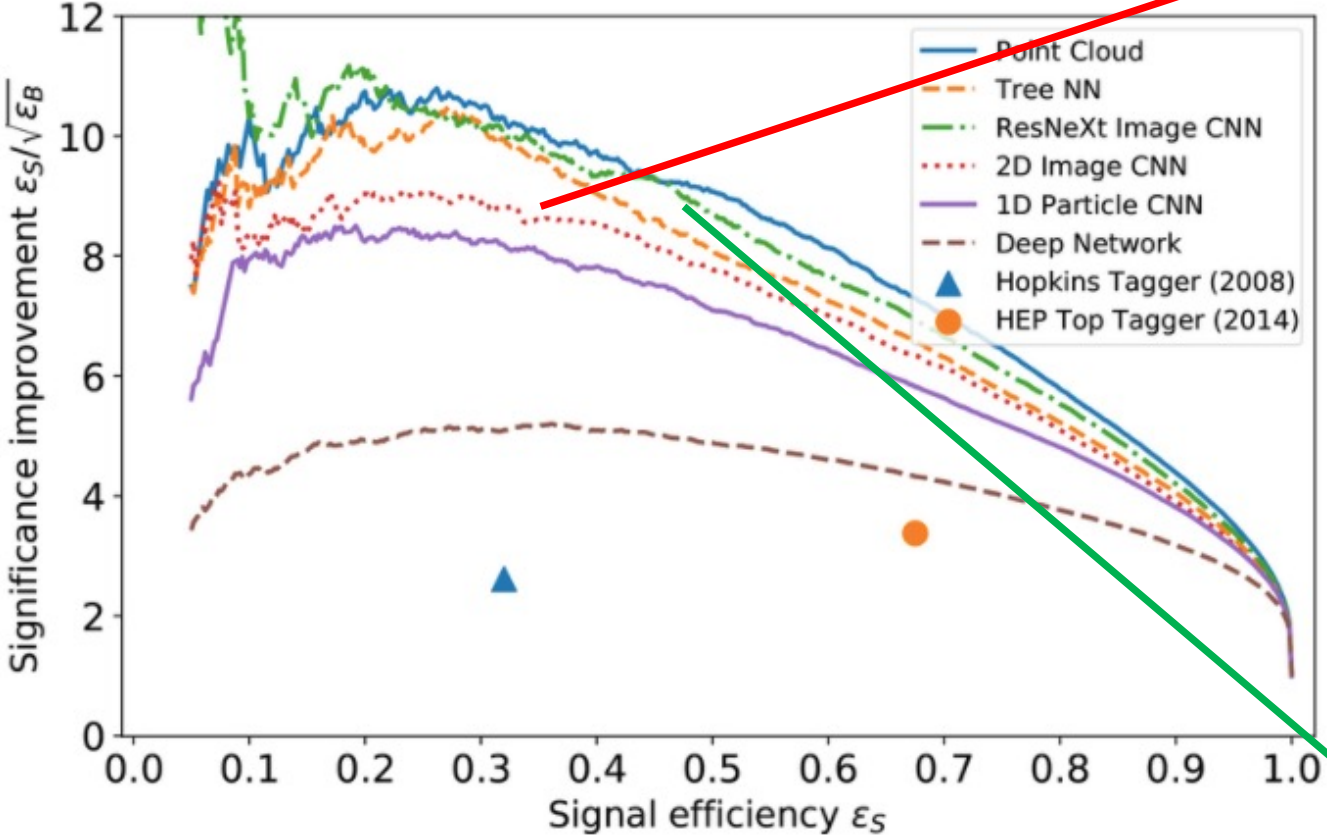


From the excellent paper [Modern Machine Learning and Particle Physics]

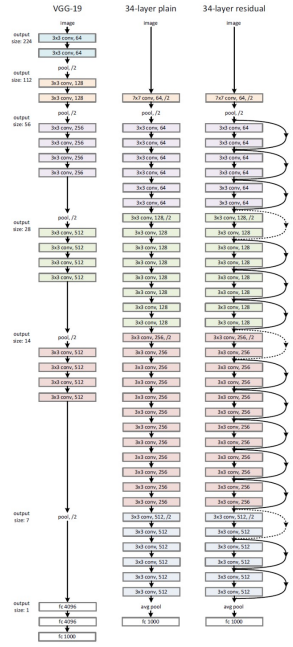
Jet Tagging (and Reconstruction/Particle ID)



Comparison of ML-based top-taggers



2D CNN using calorimeter images

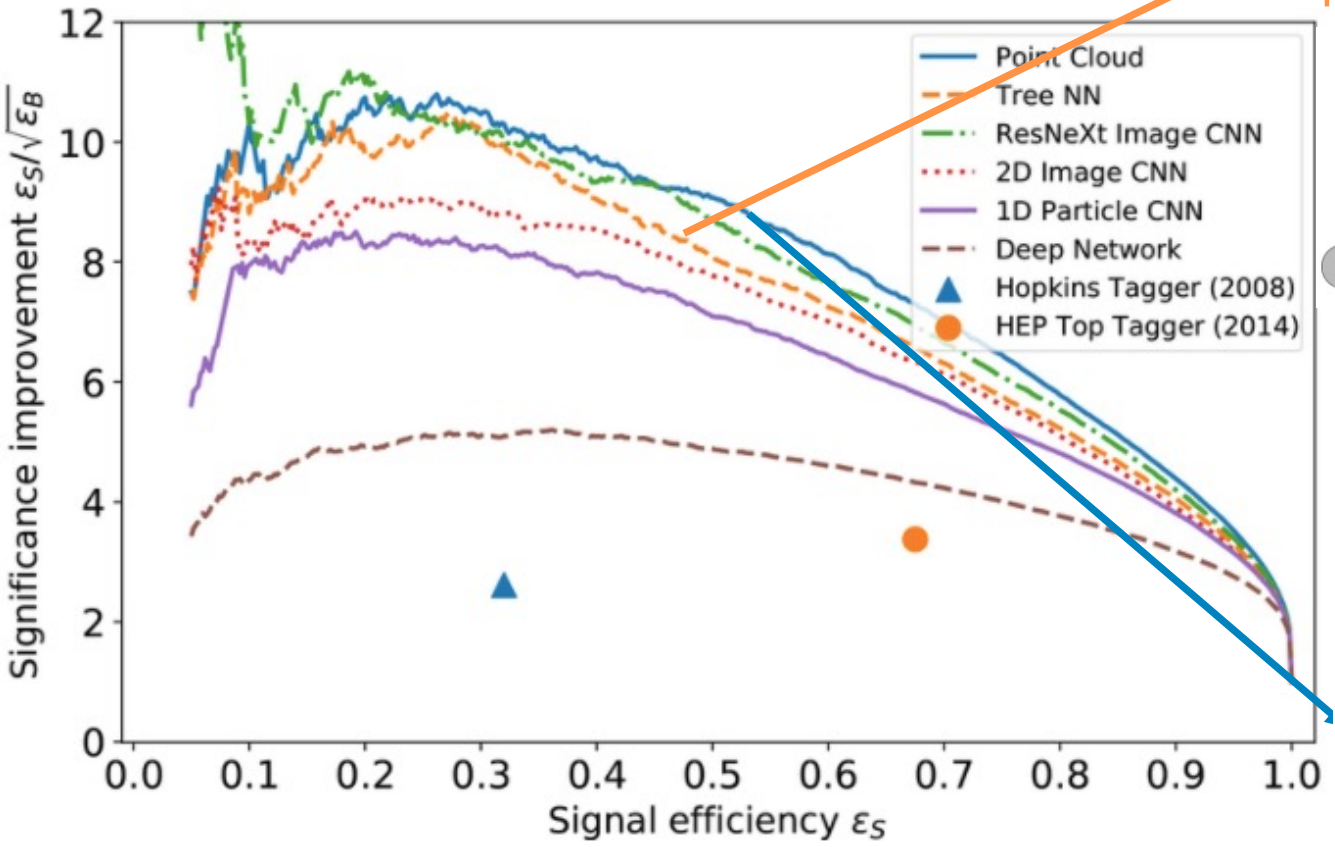


2D industry standard CNN using calorimeter images

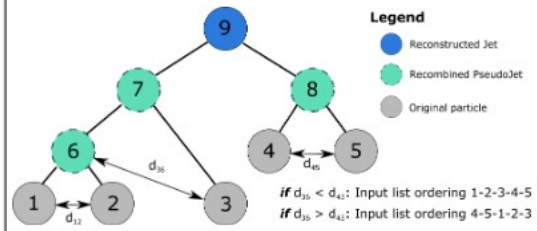
From the excellent paper Modern Machine Learning and Particle Physics

Jet Tagging (and Reconstruction/Particle ID)

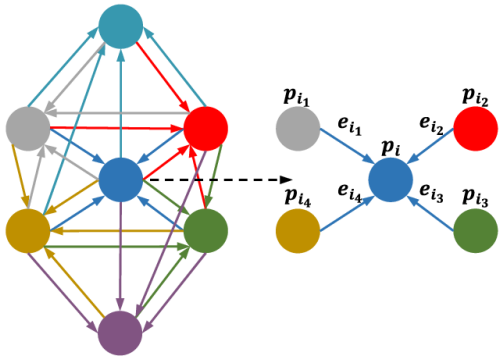
Comparison of ML-based top-taggers



RNN using tree structured constituent particle momenta



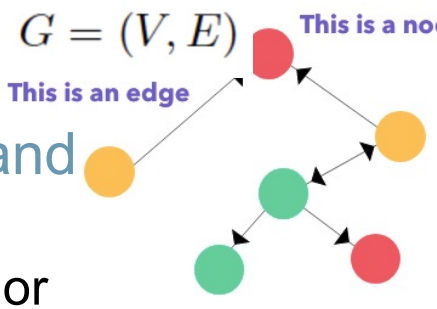
Graph Convolution Network using constituent particles as nodes

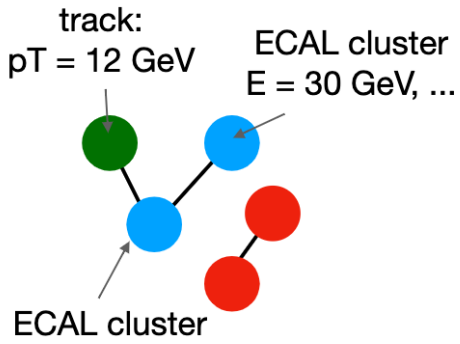


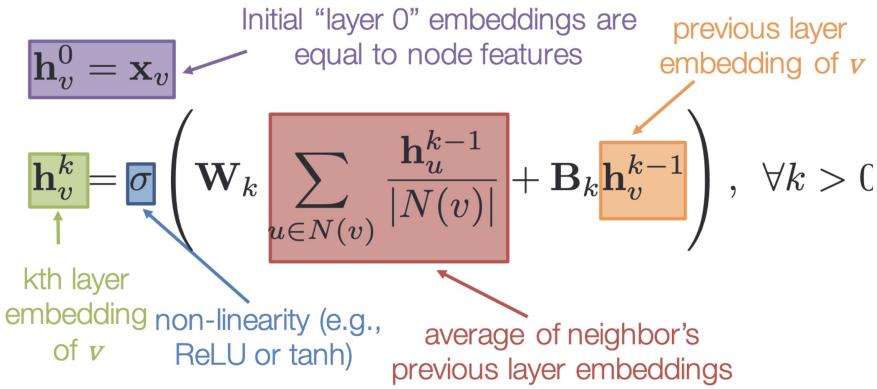
From the excellent paper [Modern Machine Learning and Particle Physics]

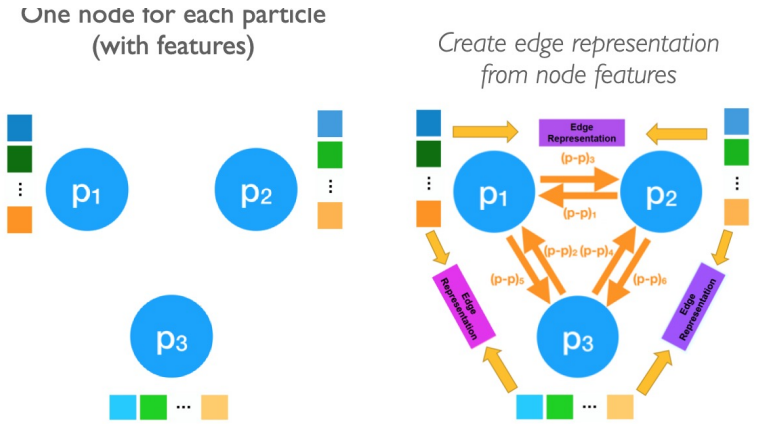
Graph Methods for Particles

- Graphs are a mathematical structure composed of nodes and edges
  - Great for represented structured or relational data!
- GNNs learn a smart embedding of graph structure
  - Leverage geometric information by passing and aggregating messages from local neighborhoods
- Many use cases in HEP
  - Tagging (particles as nodes, learned edges)
  - Clustering (detector deposits as nodes, geometry informed edges)
  - Reconstruction (particle components as nodes, physics-motivated distance metric)



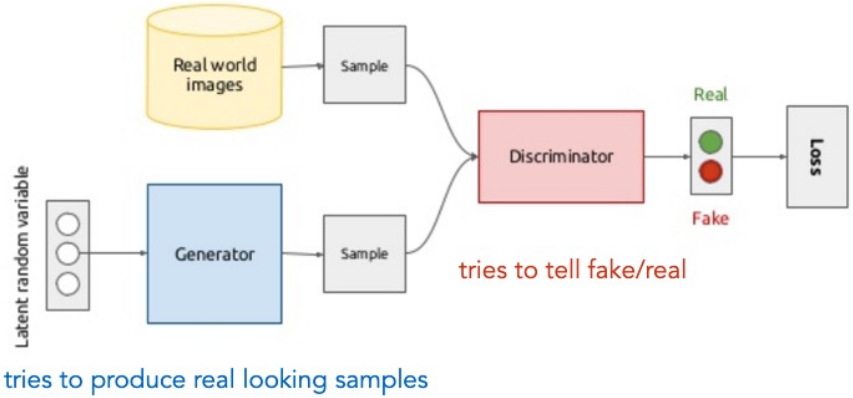


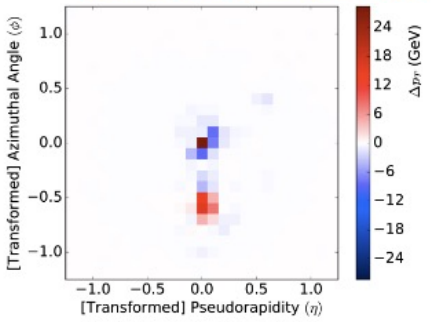




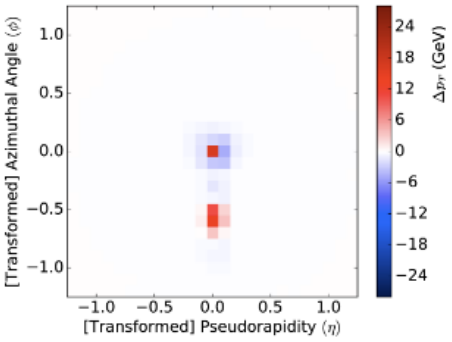
Generative Adversarial Networks

- GANs pit two networks against each other in a non-cooperative game
- Can be used to generate detector response samples
  - Well studied example: calorimeter images
  - Can include effects like pileup by summing images or condition on variables like mass
- Recently extended to use graph data representations

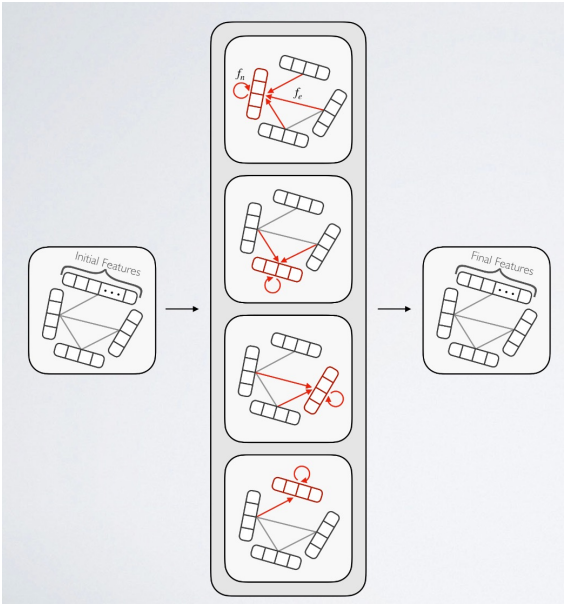




GAN-generated signal - background

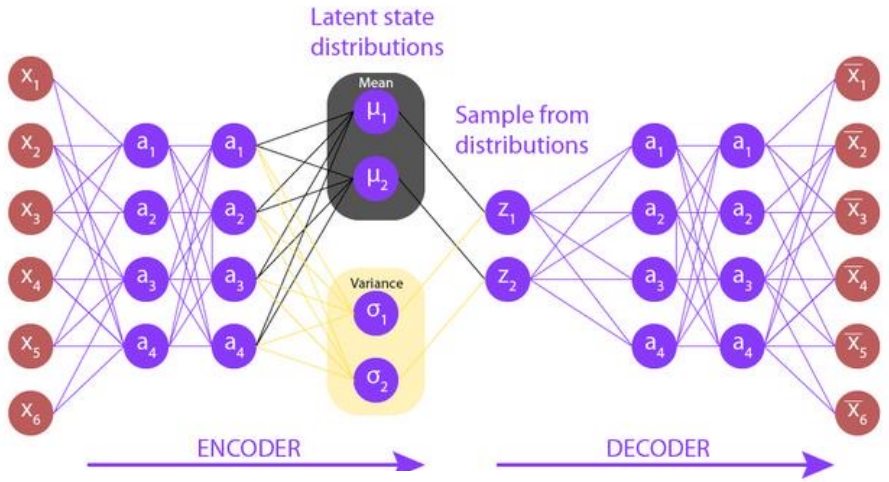


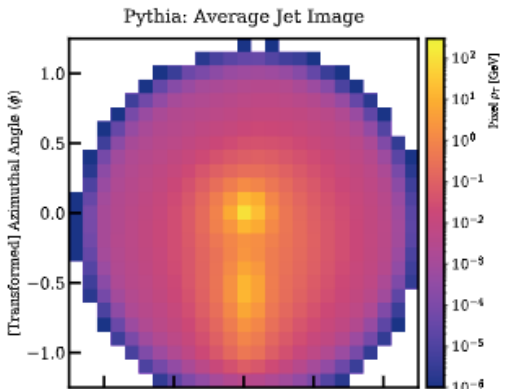
Real signal - background

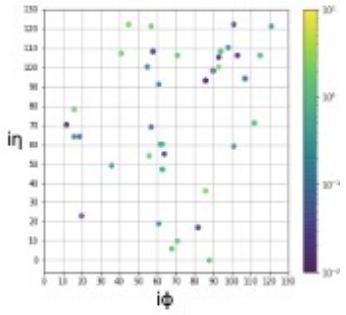


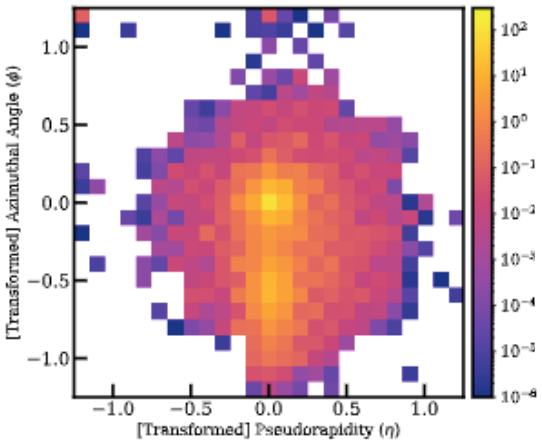
Autoencoders

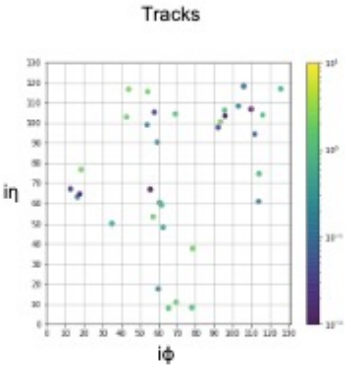
- Autoencoders reduce dimensionality of data to a learned latent space that can be reconstructed
  - Variational AEs allow a continuous latent space by assigning a distribution to each encoded feature
  - New simulations constructed by sampling from the latent space and decoding
- Several HEP applications
  - Jet calorimeter images
  - Parton showers from matrix elements
  - Particle decay chain products
  - Graph representations of full events
- Both methods can substantially speed up simulation





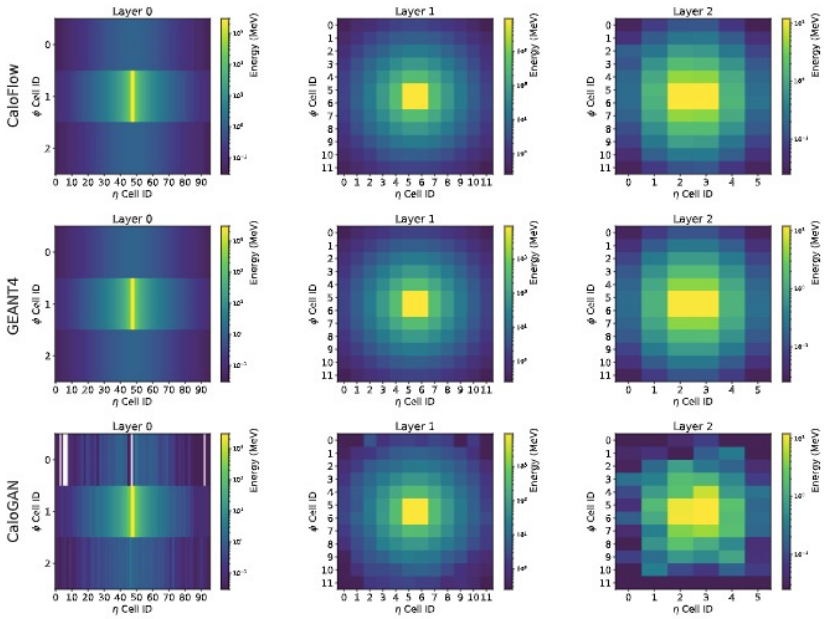


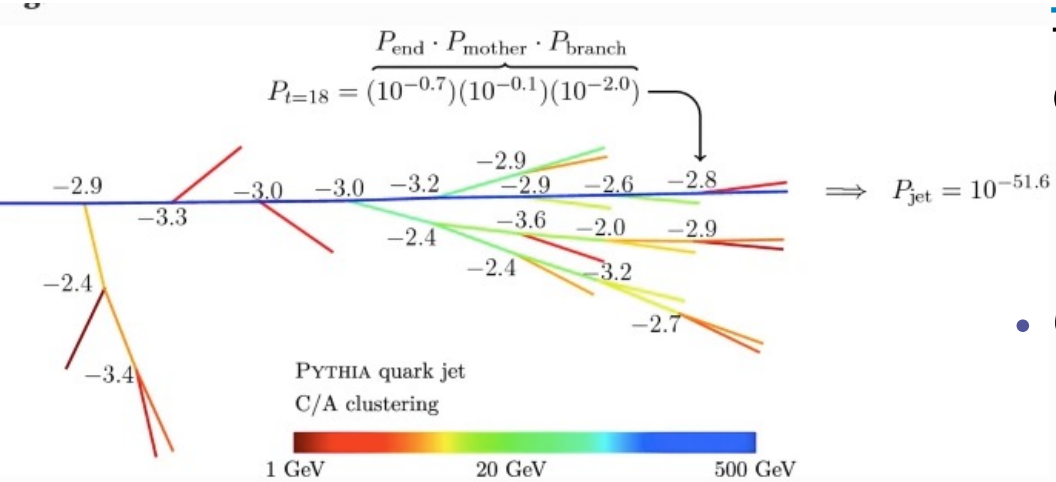




Likelihood Based Simulation

- Normalizing Flows: construct probability distributions to sample from by learning a series of invertible mappings
  - Can also be used to condition the latent space in a VAE
- Successfully demonstrated on calorimeter images
  - Better approximation of true probability distribution than GANs, but slower

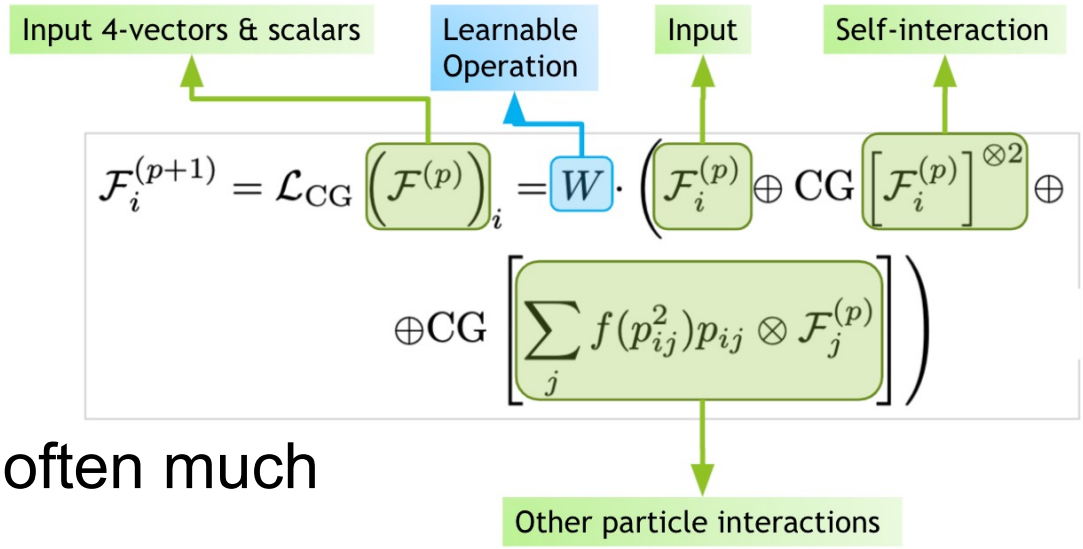




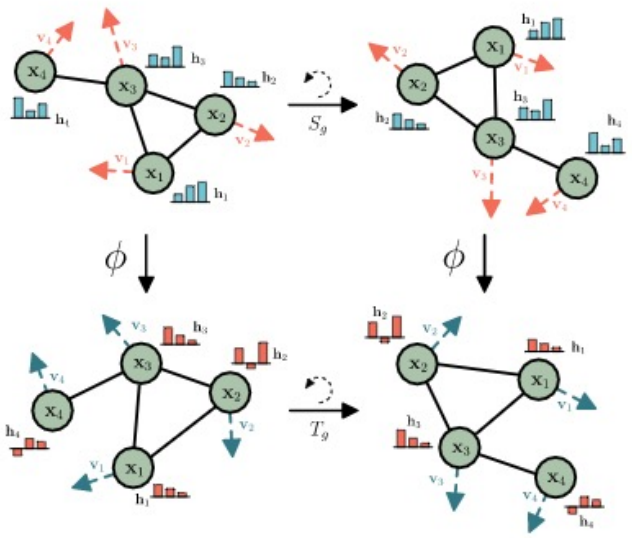
- JUNIPR Framework: learns the full differential distribution of the data
  - Learns distribution for jet showering using RNN on a tree-structured jet
- Other relevant use cases:
  - Classification using likelihood ratio
  - Event reweighting for imperfect simulators

Symmetry Aware Learning

- Constrain the functions learned by a model by requiring them to respec known symmetries or conservation laws

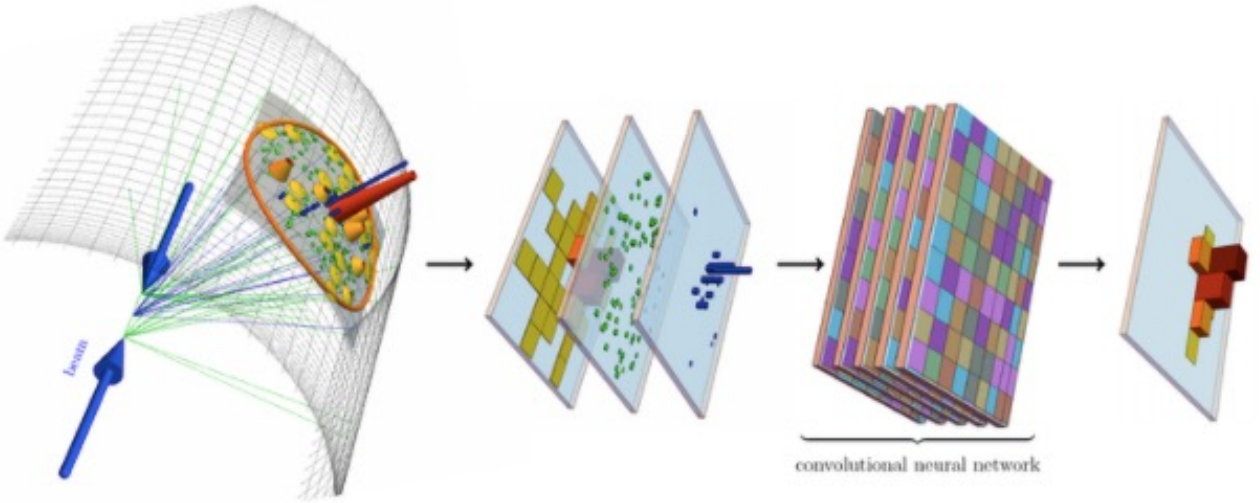


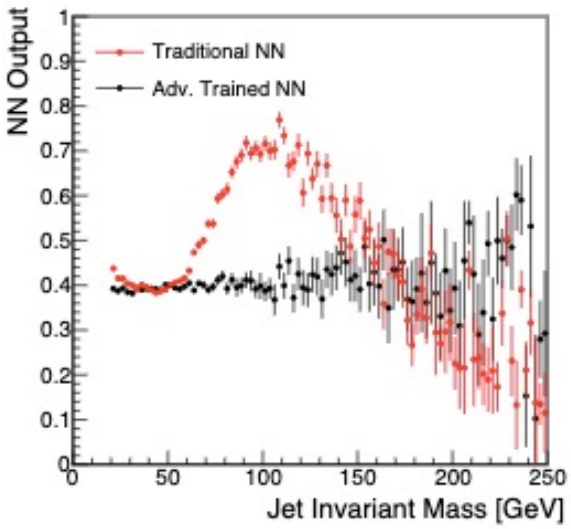
- Constrained models are often much smaller
  - Rotation equivariance well studied in general ML research
  - Recent work demonstrated Lorentz invariant classifier for jet tagging
  - Exploring symmetry-aware GNNs for tracking



Nuisance Parameter Mitigation/Decorrelation

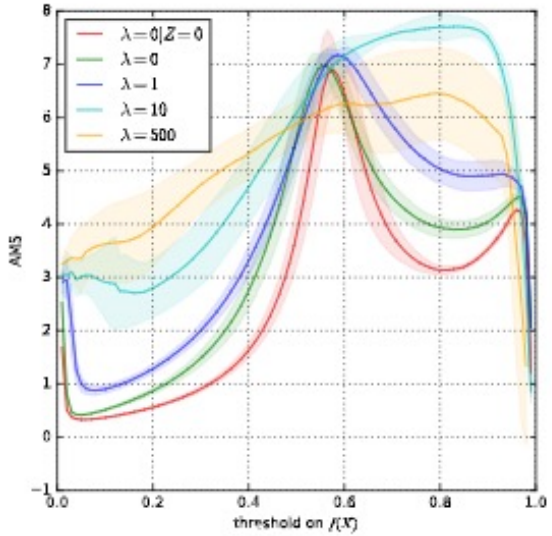
- Pileup Mitigation with ML (PUMML) using event images
  - Separate into images of primary and secondary charged particles and neutral particles
  - Predict a new image showing neutral particles from primary interaction
- Pileup scoring using GNNs
  - Regression on graph nodes representing individual particles
- Add regularizers to loss function
  - Demonstrated in Higgs event selection for removing dependence on tau momentum
- Adversarial methods with classifiers
  - Tries to reproduce nuisance parameter based on input to and output of the classifier

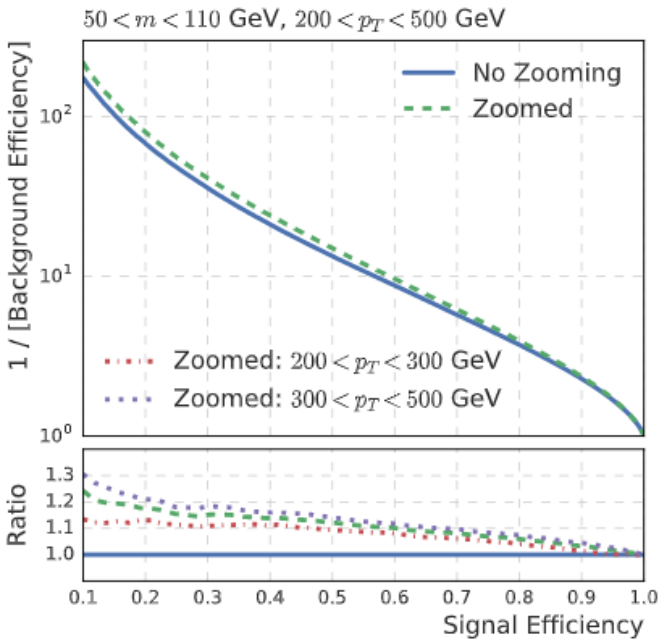




Uncertainty Reduction

- Systematic uncertainty can be lowered by reducing nuisance parameters → increased sensitivity
- Model uncertainty can be assessed in the 'standard' way by varying components of the testing data
  - E.g. varying the showering software used impacted background rejection up to 50%
- Additional uncertainty reduction possible by understanding non-physics or non-relevant signals algorithms are exploiting and removing them from the dataset
  - E.g. scaling jet images to standardize the separation between sub-jets (normally dependent on pt)
  - Model interpretability is an important area of study





Coding Time!

Happy to answer any questions!

savannah.thais@gmail.com @basicsciencesav