Physics 693: Modern ML for Physics & Astronomy

ML is revolutionizing nearly every field of science & society more broadly.

- Powerful new tool
- Enabling new analyses previously impossible
- Enhancing sensitivity & precision
- Accelerating simulation & inference
- Unifying solutions to problems across domains

Made possible by
- By data
- Computing (GPU)
- Sophisticated algorithms (NNs, ...)

ML like a telescope
or microscope

Task Forces:

Theory ↔ Exp't

Data Science

ML paper: novel mix of statistics, machine learning, etc; in each paper can have all 3!
This course: general ML concepts

popular & state of the art architectures

applications to LHC, Astro, Cosm, Condensed Matter

- No PWS but will be hands-on tutorials & exercises
  - No exams
  - BB + demos/slides

- Will ask people to pitch applications to present in class
  - either you can present, or we can learn it together & I will present

- No domain knowledge of any subfield required

- Prior experience w/ python, numpy, matplotlib would be good

- Lectures M-Th 10:20-11:40
  - poll for makeup sht?
  - in new same travel - either Zoom or guest lectures
  - 4 lectures

- Will provide refs already on course website, no textbook

- Will not require 568 (if at all) but would help if you took it
  - will try to go fast in overlapping content (569 backup, etc.)
What is ML?

"Learning from data" "glomified curve fitting"

- Data: $X_i \in \mathbb{R}$
- A vector of features
- Features could include labels $y_i$, e.g., cat vs dog, ...

"Low level" e.g.
- Particle momenta or $x_i y_i z_i x_i' y_i' z_i'$, e.g., CLIC, Gaia

"High level" or
- Features
- Pixel intensities in image
- "Table data"
- $N = \ldots \#$ of instances
- E.g., events CLIC stars, Gaia images of galaxies
- $i = 1, \ldots, N$

Assume in most applications: each $X_i \sim p_{data}(x)$ drawn iid

Parameters

Fit $f(\theta; \phi \phi^T \phi) \downarrow $ data

Goal: minimize some "loss" or "objective" $L$

$L(\theta) = \sum_{i=1}^{N} L(f(x_i; \phi), x_i)$

wrt parameters $\theta$
Example: maybe we use \( f(x_i; \theta) = y_i \) - discrete "classification"

Could use \( \mathcal{L}(\theta) = (f(x_i; \theta) - y_i)^2 \) - continuous "regression"

"Mean squared error (MSE)"

- many more examples to follow (how to choose loss function)

Where do labels come from?

- if data is simulated - have access to "ground truth"
- in actual data - human labels? (common in real world)
  - no labels or labels derived from data itself

Categories of ML

- fully supervised - all data labeled & used training
  - classification, regression
  - unsupervised - no labels!
  - generative modeling, density estimation, etc. (some kinds)
  - weakly supervised - noisy labels
  - semi-supervised - mix of labeled & unlabeled data
  - self-supervised - labels generated from data e.g. intrinsic loss
less than experienced

Also really important for science - strive to be as data-driven as possible

(simulation-free)

In more detail:

- Generative modeling (GANs, VAEs, generative flows, etc.)
  
  $x \xrightarrow{\text{update (x)}} \xrightarrow{\text{learn to sample from p(data)}} x_i$

- Density estimation - learn p(data (x)) (density flows),
  (Generically $\propto$ density est)

  one doesn't guarantee the other

- Anomaly detection

  $\xrightarrow{\text{outlier detection}}$ (autoencoders, deep
  $\xrightarrow{\text{point to anomalies}}$

  $\xrightarrow{\text{overdensity detect}}$

  $\xrightarrow{\text{gray anomalies}}$
General principle for loss funs: Maximum Likelihood Estimate (MLE)

want to maximize \( P(\text{data} | \theta) \)

if data iid

\[
\prod_{i=1}^{N} P(x_i | \theta) \]

\[
L = -\log P(\text{data} | \theta) = -\sum_{i=1}^{N} \log P(x_i | \theta)
\]

- MLE has properties in lot of large N:
  - consistency (as N→∞, estimated θ → true θ)
  - efficiency (as N→∞, minimum variance estimator)

Example:

Suppose want to predict \( y \) given \( x \) (regression)

\( f(x; \theta) \) is Gaussi dist'd and

\[
P(y | x; \theta) = \frac{1}{\sqrt{2\pi \sigma^2}} e^{-\frac{(y - f(x; \theta))^2}{2\sigma^2}}
\]

\[
L = -\log P = \sum \frac{(y_i - f(x_i; \theta))^2}{2\sigma^2} + \log \sqrt{2\pi \sigma^2}
\]

if \( \sigma^2 \) known \( \rightarrow \text{MSE} \) (otherwise do not fix \( \sigma^2 \))
Ex: binary classification

labels \( y_i = 0 \) or 1 for data \( x_i \)

model: \( f(x_i; \theta) = \text{prob of } x_i \text{ is } y = 1 \)

\[
P(\text{data} | \text{model}) = \prod_{i=1}^{N} f(x_i; \theta)^y_i (1-f(x_i; \theta))^{1-y_i}
\]

\[
L = -\log P = -\sum_{i} \log f(x_i; \theta) \leq \sum_{i} y_i (1-f(x_i; \theta))
\]

\[
= -\sum_{i} (y_i \log f(x_i; \theta) + (1-y_i) \log (1-f(x_i; \theta))
\]

"binary cross entropy loss" → best loss for classification

Could use MSE, but it would be suboptimal