Physics 693: Modern ML for Physics & Astronomy

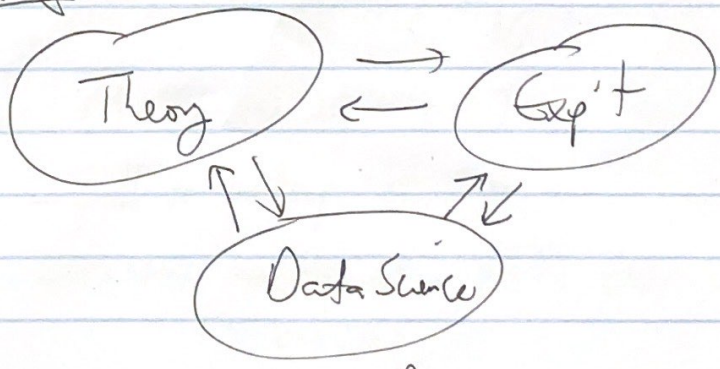
ML is revolutionizing nearly every field of science & society more broadly

Powerful new tool
- enabling new analyses previously impossible
- enhancing sensitivity & precision
- accelerating simulation & inference
- unifying solutions to problems across domains

made possible by
- Big data
- Computing (GPU)
- Sophisticated algorithms (NNs, ...)

ML like a telescope! or microscope

This course:



→ formal mathematics

novel mix of ~~theory~~, statistics, numerics

in ML some papers can have all 3!

This course:

This course: general ML concepts

popular & state of the art architectures

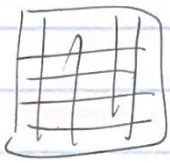
applications to LHC, Astro, Cosm, Condensed Matter (I know more ——→ less)

- No HWs but will be hands on tutorials & exercises (required) & 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) — need input!
- 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 slt?
  - in Nov some travel — either Zoom or guest lectures (4 lectures)
- will provide refs & readings on course website; no textbook
- will not require 568 (future will) but would help if you took it (NN basics ...)
  will try to go fast in overlapping content (SGD, backprop, ...)

please interrupt frequently & ask me lots of ???'s

What is ML?

"learning from data" "glorified curve fitting"

- Data: $\vec{x}_i \in \mathbb{R}^d$ ← d can be high! e.g. $10^2, 10^3, \ldots$
  - ↳ a vector of features
    - "low level" vs "high level" features
      - e.g. particle momenta @ LHC or $x, y, z, v_x, v_y, v_z, G, B-R, \ldots$ @ Gaia
      - or  pixel intensities in image
    - ↳ "tabular data"
  - features could include labels $\vec{y}_i$ e.g. cat vs dog, ...

$i = 1, \ldots N$ # of instances → N could be large $10^6, 10^9, \ldots$

(e.g. events @ LHC, stars in Gaia, images of galaxies)

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

- Goal: minimize some "loss" or "objective" fn

$$L(\theta) = \sum_{i=1}^{N} \mathcal{L}(f(\vec{x}_i; \theta), \vec{x}_i)$$

wrt parameters θ.

parameters ← fit fn $f(\vec{x}_i; \theta)$ to data

Example: Maybe we have $f(x_i;\theta) = y_i$ ← target labels

- 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 fn in principled way)

- Where do labels come from?
  - if data is simulated have access to "ground truth"
  - in actual data — human labels? (common in real world apps like Imagenet, also Astro citizen science)
  - no labels or labels derived from data itself

DNN, RNN, CNN, graph NNs, transformers, ... → building blocks

↑ Categories of ML

pretty much solved problem in ML
- fully supervised — all data labeled & used in training
  - classification, regression

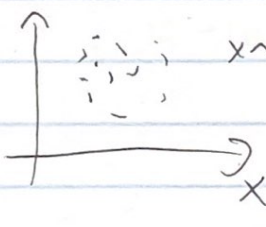
this is where the active ML development is happening
- unsupervised — no labels!
  - generative models, density estimation, some kinds of anomaly detection
- weakly supervised — noisy labels
- semi supervised — mix of labeled & unlabeled data
- self supervised — labels generated from data e.g. contrastive loss

less-than-supervised

Also really important for science — strive to be as data-driven as possible (simulation-free)

In more detail:

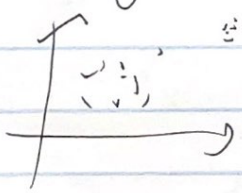
- Generative modeling (GANs, VAEs, normalizing flows, diffusion, ...)



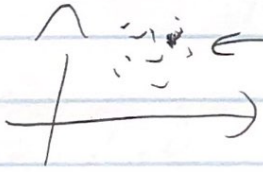
- Density estimation — learn $p_{data}(x)$ (normalizing flows, ...)

(Gen modeling $\not\Leftrightarrow$ Density est)
one doesn't guarantee the other

- Anomaly detection



(autoencoders, ...)



General principle for loss fns: Maximum Likelihood Estimation (MLE)

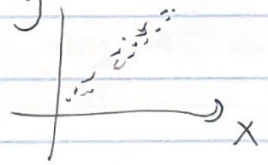
want to maximize $P(\text{data} \mid \text{model}, \theta)$

if data iid $\prod_{i=1}^{N} P(x_i \mid \theta)$

$$L = -\log P(\text{data} \mid \text{model}) = -\sum_{i=1}^{N} \log P(x_i \mid \theta)$$

- MLE has properties in lot of large N:
  - consistency (as $N \to \infty$, estimated $\theta \to$ true θ)
  - efficiency (as $N \to \infty$, minimum variance estimator of θ)

Example:



Suppose want to predict y given x (regression)

model: y is gaussian distributed around $f(x;\theta)$ w/ some width σ

then $P(y \mid x, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-f(x;\theta))^2}{2\sigma^2}}$

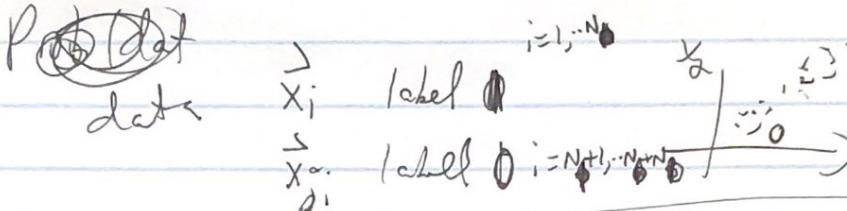
$$L = -\log P = \sum \frac{(y_i - f(x_i;\theta))^2}{2\sigma^2} + \log\sqrt{2\pi\sigma^2}$$

if σ known $\to$ MSE! (otherwise diff for σ)

Ex: binary classification

labels $y_i = 0$ or 1 from data x_i

model: $f(x_i;\theta)$ = prob of x_i being label 1.



$$P(\text{data}|\text{model}) = \prod_{i=1}^{N_1} f(x_i;\theta) \prod_{i=N_1+1}^{N_1+N_0} (1-f(x_i;\theta))$$

$$L = -\log P = -\sum_{i\in 1} \log f(x_i;\theta) - \sum_{i\in 0} \log(1-f(x_i;\theta))$$

$$= -\sum_i \left(y_i \log f(x_i;\theta) + (1-y_i)\log(1-f(x_i;\theta)) \right)$$

"binary cross entropy loss" → best loss for classification

→ could use MSE but it would be suboptimal

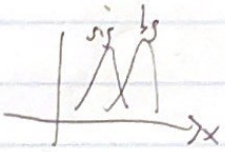
Lecture II

- email list
- slack
- pitches for applications

Neyman-Pearson Lemma

$$L = -\left(\sum_{i \in S} \log f(x_i) + \sum_{i \in B} \log(1 - f(x_i)) \right)$$

What f minimizes L in ideal case?



prob density of s & b

Continuum limit (∞ data)

$$L = -\int dx \left[\rho_S(x) \log f(x) + \rho_B(x) \log(1 - f(x)) \right]$$

$\int dx\, p(x) \approx \sum_{x \sim p(x)}$ ●

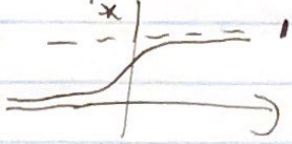
$f \to f + \delta f$, $\left. \frac{\partial L}{\partial \delta f} \right|_{f = f_*} = 0$ ← optimal f

(assume $N_S = N_B$ WLOG)

$$0 = \frac{\rho_S(x)}{f_*} - \frac{\rho_B(x)}{1 - f_*} \Rightarrow \boxed{f_* = \frac{\rho_S(x)}{\rho_S(x) + \rho_B(x)}}$$

So optimal classifier ◉ $f_* = \frac{R(x)}{R(x)+1}$ ◉

$R(x) = \frac{\rho_S(x)}{\rho_B(x)}$



f_* is monotonic w/ $R(x)$

$$\Rightarrow \boxed{f_* > f_c \iff R(x) > R_c}$$

cut on f equiv to cut on R

So optimal classifier is likelihood ratio

"Neyman-Pearson Lemma"

- powerful result! fundamental result in statistics
  (LR Uniformly most powerful test for simple hypothesis)

later will see:
- Can turn around → learn (ML) classifier from samples
  → approach LR.
  "likelihood ratio trick"

- Another perspective: Bayes Thm

$$f_x = \frac{p_S(x)}{p_S(x)+p_B(x)} = \frac{p(x|S)}{p(x|S)+p(x|B)} = \frac{p(x|S)p(S)}{p(x)} \quad (p(S) = \tfrac{1}{2})$$

$$= \frac{p(x|S)+p(x|B)}{2}$$

$$= p(S|x)$$

∴ f_x is the prob of signal given x which is where we started!

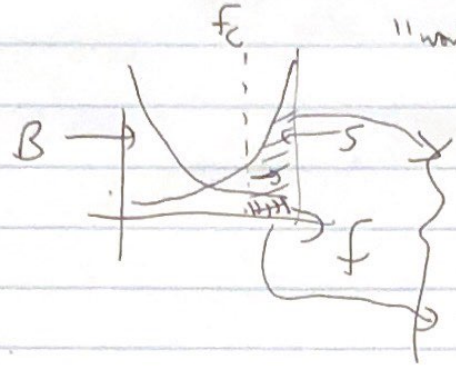
i.e. BCE is minimized when our model for this prob = true prob. ✓

Back to NP lemma: what does "most powerful" test mean?

→ metrics for binary classification

How do we use binary classifier?

Usually: cut $f(x) > f_c$ ("working pt") — all x satisfy cut are "signal" (classified as S); all x fail cut are "bg"

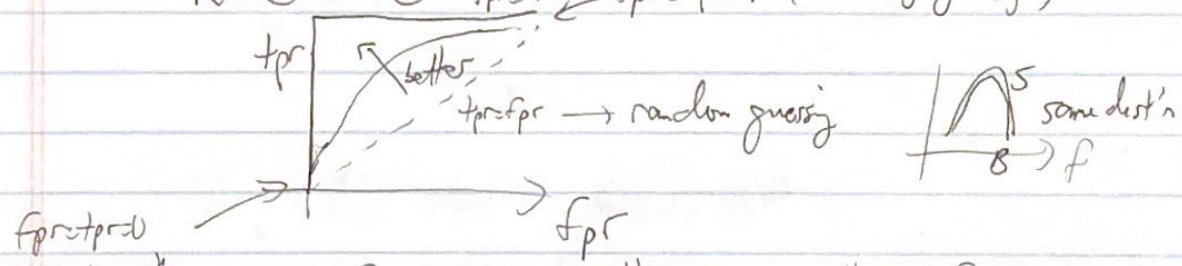


$$tpr = \frac{N_S(f > f_c)}{N_S} \qquad fnr = 1 - tpr$$

$$fpr = \frac{N_B(f > f_c)}{N_B} \qquad tnr = 1 - fpr$$

- $\text{accuracy} = \max_{f_c} \frac{tpr + (1 - fpr)}{2}$

- ROC curve: $tpr = 1$ ← $fpr = tpr = 1$ (let everything through)



Vary f_c, sweep out "ROC curve" $(fpr(f_c), tpr(f_c))$

→ AUC "Area under the curve"

- AUC = 1 perfect
- AUC = 0.5 random

(same for ACC but AUC ≠ ACC)

- AUC, ACC mostly sensitive to tpr, fpr ~ O(1) part of ROC curve

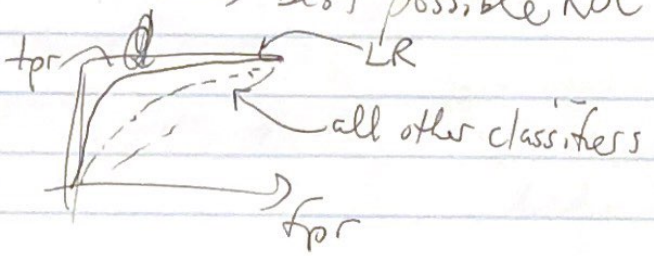
When $N_{bg} \gg N_{sig}$ (as in many cases, eg LHC)

care more about $fpr \ll 1$ part of ROC curve

→ often report fpr @ fixed tpr eg $tpr = 50\%$ or 30%

or rejection factor { $R50 = \frac{1}{fpr @ tpr = 50\%}$; R30 etc

- Neyman-Pearson LR classifier is optimal
  → best possible ROC curve



- In practice cannot achieve NP optimality
  - exact likelihods unknown
  - finite training data → Dera of Big data
  - limited model capacity (expressivity)
    ↳ NNs very ~~powerful~~ expressive
    ⇒ can maybe get very close to optimal!

Then possibly biased on val set

→ final metrics on another held out dataset

"test set"

Example 1: Gaussian toy model (10d)

$$\begin{cases} b_{ij}\ (0) & P_{b_{ij}}(x) = \mathcal{N}(0,1)^{10} \\ s_{ij}\ (1) & P_{s_{ij}}(x) = \mathcal{N}(0.5,1)^{10} \end{cases}$$

- can compare w/ NP lemma

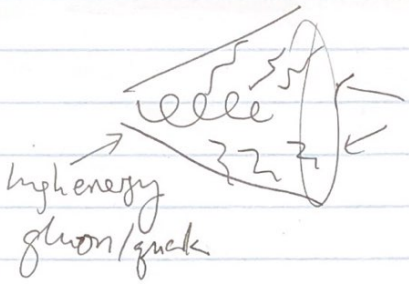
$$P_{NP}(s_{ij}|x) = \frac{P_{s_{ij}}(x)}{P_{b_{ij}}(x) + P_{s_{ij}}(x)}$$

- train simple DNN on samples
  - plot losses
  - ROC curve
  - AUC
  - classifier output

- Jupyter nb demo

Example 2: Jet classification @ LHC

- Jets are essential objects @ LHC
  Product of strong interaction (QCD) → showers of hadrons



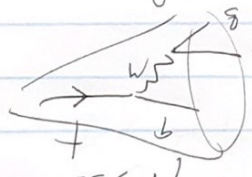
↓ fills out a cone of energy "jet"

- Many different flavors of jets — important to tell them apart!
  q, g, b, W, Z, top, ...

↓ SM measurements ↓ BSM searches

Typically QCD jets (q, g) are background & everything else is signal

- Especially rich example: boosted top jets



$m = 175$ GeV
Heaviest ptcl in SM

- Decays to bW
  W can decay to quarks
- If top is very energetic → boosted decay products collimated "fat jet"

- Top jet has substructure → expect 3 "prongs" of energy $\simeq b, q, q'$
  - expect $m_{jet} \simeq m_{top}$
  - expect $q q'$ mass $\simeq m_W$.
  - expect b jet (won't use here) — its own w/ special properties

Note: QCD xsec thousands times more than top (or more) dependent on energy (~1000x @ m_{jj} ~ 1TeV)

- Meanwhile QCD jets are relatively structureless.

→ can use these properties to classify Tops vs QCD!

(unlike Gaussian toy model, here we do not know true likelihoods, so do not know NP optimal classifier)

- Jupyter nb example demos
  - (m_j, m_W, τ_{32}) → cuts
    → DNN
  - jet constituents → DNN
  - ⋮

Metric: R30
1/fpr @ tpr = 30%
more informative than AUC/ACC

A jet = collection of ptcl 4-vectors "LLFs"

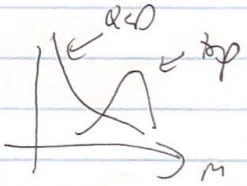
$$\text{jet}_i = \begin{pmatrix} p_x^1, p_y^1, p_z^1, E^1 \\ \vdots \\ p_x^{N_i}, p_y^{N_i}, p_z^{N_i}, E^{N_i} \end{pmatrix} \quad N_i \text{ diff from jet to jet}$$

$(E^a)^2 \geq (\vec{p}^{\,a})^2$

HLFs like jet mass $m_i^2 = \left(\sum_{a=1}^{N_i} E^a\right)^2 - \left(\sum_{a=1}^{N_i} \vec{p}^{\,a}\right)^2$

↓ and N-subjettiness τ_{32}

physics motivated fns of LLFs



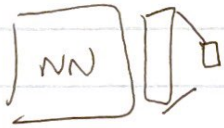
We have seen:

cut based classifier on HLFs $\simeq$ DNN on HLFs

$\simeq$ DNN on LLFs (30 highest p_T constituents)

"Automated feature engineering"

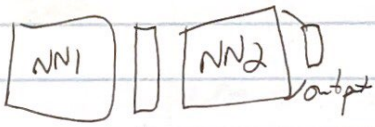
- DNN can construct HLFs from LLFs that performs as well as physics HLFs!
  - Q: is it learning the physics HLFs or something else? How would we know?
    A: Could add physics HLFs to inputs — perf. improves?

- 

last layer → these or like NN engineered HLFs!

↓ motivated the HLF engineer

not necessarily very last layer

- 

↳ could interpret these as HLFs

- Is this all or can we surpass physics HLF performance?

→ yes but need more powerful NN architectures!

general lesson: → leverage symmetry / structure of data!

↓

like getting more data for free

"inductive bias" → helps machine learn more effectively

our DNN p_T ordered constituents

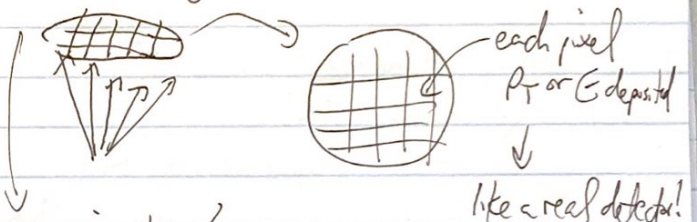
selected 30 hardest

But jets have permutation invariance! ← order of constituents doesn't matter

also: Lorentz symmetry → spatial rotations & boosts

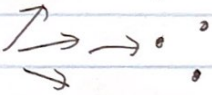
"point cloud"

- A rich playground for different ML architectures
  - CNNs — jets as images



perm inv't ✓
Lorentz sym? → spatial rotations & translations

  - RNNs, LSTMs, ... — jets as sequences
    — not perm inv't! what ordering?
  - Graph NNs — perm inv't!
    (can add Lorentz)



  - Deepsets — a perm inv't DNN
  - Transformers — a perm inv't sequence NN.

→ Plan for next few lectures + (detours into Astro w/ CNNs)