Lecture II

Neyman-Pearson Lemma

\[ L = \frac{1}{n} \log f(x_i) + \frac{1}{m} \log (1 - f(x_i)) \]

What \( f \) minimizes \( L \) in ideal case?

Continuum limit (infinite data)

\[ L = -\int dx \left[ p_s(x) \log f(x) + p_B(x) \log (1 - f(x)) \right] \]

\[ \int d\phi(x) = \sum_{x \in \Omega} \phi \rightarrow f + \delta f, \quad \frac{\partial L}{\partial f \delta f} = 0 \]

\( f = f^* \) is optimal \( f \)

\[ 0 = \frac{p_s(x) - p_B(x)}{f^*_x - 1 - f^*_x} \Rightarrow f^*_x = \frac{p_s(x)}{p_B(x)} \]

\[ \frac{p_s(x)}{p_B(x)} + \frac{p_s(x)}{p_B(x)} = 1 \]

\( f^*_x \) is monotonically increasing

\( \Rightarrow f^*_x > f_c \Rightarrow R(x) > R_c \)

Optimal classifier is likelihood ratio

"Neyman-Pearson Lemma"
Powerful results! Fundamental result in statistics

(LR uniformly most powerful test for simple hypothesis)

Later will see:
- Canonical: learn ML classifier from samples → approach LR
  "likelihood ratio trick"

- Another perspective: Bayes Theorem

\[ f_x = \frac{P_S(x)}{P_S(x) + P_B(x)} = \frac{\frac{P(x|S)}{P(x)}}{\frac{P(x|S)}{P(x)} + \frac{P(x|B)}{P(x)}} = \frac{P(x|S)}{P(x|S) + P(x|B)} \]

\[ = \frac{P(S|x)}{2} \]

So, \( f_x \) is the prob of signal given \( x \), which is where we started!

i.e., BCE is minimized when our model predicts the prob = true prob.

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

→ metrics for binary classification
AUC: Area Under the Curve

AUC = 1: Ideal
dAUC = 0: Random

ROC curve: True positive rate (TPR) vs. False positive rate (FPR)

- Accuracy = max f(p) + (1-f(p))

- ROC curve:
  - TPR = \frac{TP}{TP + FN}
  - FPR = \frac{FP}{FP + TN}

- Thresholds:
  - f(x) = \frac{N_S(x) \delta}{N_S(x) \delta + N_B(x) \alpha}
  - \delta = 1 - \frac{N_S(x)}{N_B(x)}
  - \alpha = \frac{N_B(x)}{N_S(x) + N_B(x)}

- How do we use known classifier?
When \( N_y \gg N_f \) (as in many cases, e.g., LHC),
care more about \( \text{FPR} < 1 \% \) rather than \( \text{ROC curve} \).
→ Often report for \( \text{FPR} \leq 5\% \) or \( \leq 3\% \)
or rejection factor: \( R_{50} = \frac{1}{\text{FPR} \leq 5\%} \)
\( R_{30} \) etc.

- Neyman–Pearson LR classifier is optimal
  → Best possible ROC curve
  \[ \text{LR} \quad \text{all other classifiers} \]

- In practice cannot achieve NP optimality
  - Exact likelihoods unknown
  - Finite training data → Bias of big data
  - Limited model capacity (expressivity)

  \( \Rightarrow \) Thus very flexible, expressive, etc.

  Cannot be set very close to optimal.
Neural Networks

- So far have not specified family of fits for $f(x; \theta)$
- May choices (BOTS, SVMs, Bayesian models, ...)
  - Deep learning / "modern ML": $f(x; \theta) = \text{NN}$
  - Why? Universal approximator: roughly, "NN can approximate differentiable (e.g. high model can be trained on a bit of data that is well understood, using backprop & SGD), but surprisingly resistant to overfitting". "Generalization puzzle" "Inductive bias"

- Basic structure of NN "Feed-forward"
  - "Nodes" or "units"
  - "MLP" "DNN" "Fully connected"
  - Input $x^{(1)}$ outputs $x^{(2)}$ ..., $x^{(L)}$ outputs $z = F(x; \theta)$ "Biases"
  - Each connection: "weight" $x^{(i)} = A^{(i)}(w^{(i)}x^{(i-1)} + b^{(i)})$
  - All other weights & bias $w_i$, $b_i$
Activation: only source of non-linearity.

Examples: \[ A(x) = \frac{e^x}{1 + e^x} \]

"Sigmoid" function:

\[ A(x) = \begin{cases} 0 & x < -3 \\ 3 & x > 3 \\ \tan h & \text{best choice} \\ \text{otherwise lead to "vanishing gradient problem" - more later} \end{cases} \]

\[ \text{ReLU(x)} = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \]

Note: activation acts elementwise.

\[ A^{(1)}(x_1, x_2, \ldots, x_n) = \left( A^{(1)}(x_1), A^{(1)}(x_2), \ldots, A^{(1)}(x_n) \right) \]

Structure of NN is recursive:

\[ x^{(n)} = A^{(n)}(w^{(n)} x^{(n-1)} + b^{(n)}) \]

\[ z = A^{(n+1)}(w^{(n+1)} x^{(n)} + b^{(n+1)}) \]

Final activation depends on problem:

\[ z = \begin{cases} \text{p(x|y)} & \text{for binary classification} \\ \text{p(y|x)} = 1 - \text{p(x|y)} & \text{for regression} \end{cases} \]

\[ \text{output} \text{ maps to 1} \]

Popular choice: Softmax:

\[ \frac{e^{x_i}}{\sum e^{x_j}} \]