Neural Feature Learning

Lizhong Zheng

NIST, October 2023
Where Are We on the BlackBox Issue?

- Deep neural networks learn "informative functions": information/statistical measures used in loss and regulator.
- LLM and AI for Science: no model, no repetition.
- How to control, measure, or certify the internal operations?
- Aligned with many engineering applications: domain knowledge, structure, constraints, parameterized solutions.
- New mathematical tools.
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM}; \)
- Parameters \( h_1, h_2, \sigma_W^2 \) known at the receiver (CSIR).
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM}; \)
- Parameters \( h_1, h_2, \sigma_W^2 \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM}; \)
- Parameters \( h_1, h_2, \sigma^2_W \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM} \);
- Parameters \( h_1, h_2, \sigma_W^2 \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM}; \)
- Parameters \( h_1, h_2, \sigma_W^2 \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in \text{16-QAM}; \)
- Parameters \( h_1, h_2, \sigma_W^2 \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters
A Detection Problem

\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W \]

- \( X_1 \in \text{BPSK}, \quad X_2 \in 16\text{-QAM}; \)
- Parameters \( h_1, h_2, \sigma^2_W \) known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - Highly depends on the parameters
The Current Solutions (Complaints)

- The common "black box" issues: no guarantee, no metric, etc.
The Current Solutions (Complaints)

- The common "black box" issues: no guarantee, no metric, etc.

- Parameterized receiver:
  - Training with mixed samples
  - Parameters as input
  - Online vs. Offline
The Current Solutions (Complaints)

- The common "black box" issues: no guarantee, no metric, etc.

- Parameterized receiver:
  - Training with mixed samples
  - Parameters as input
  - Online vs. Offline

- Receiver with side-information, multi-variate dependence

\[
\begin{align*}
\mathbf{Y}, \mathbf{X}, (h_1, h_2, \text{SNR}) & \quad \mathbf{S} \\
\end{align*}
\]
Feature functions as vectors, $f : \mathcal{X} \rightarrow \mathbb{R}$; $f \in \mathcal{F}_\mathcal{X}$

Inner product

$$\langle f_1, f_2 \rangle \triangleq \mathbb{E}_{X \sim R_X}[f_1(X) \cdot f_2(X)]$$

- Zero-mean w.r.t. $R_X$,
- $R_X$: metric distribution, reference, ...
- Fisher information metric
Low Rank Approximation of Dependence

- Dependence between $X$ and $Y$:

$$i_{X;Y}(x, y) = \frac{P_{XY}(x, y)}{P_X(x)P_Y(y)} - 1; \quad i_{X;Y} \in \mathcal{F}_{X \times Y}$$

- Reference distribution $R_{XY} = P_X \cdot P_Y$
Low Rank Approximation of Dependence

- Dependence between $X$ and $Y$:

$$i_{X;Y}(x, y) = \frac{P_{XY}(x, y)}{P_X(x)P_Y(y)} - 1; \quad i_{X;Y} \in \mathcal{F}_{X \times Y}$$

- Reference distribution $R_{XY} = P_X \cdot P_Y$

- Subtract 1 to make zero-mean, (local approx. of PMI)
Low Rank Approximation of Dependence

- Dependence between $X$ and $Y$:
  \[
  i_{X;Y}(x, y) = \frac{P_{XY}(x, y)}{P_X(x)P_Y(y)} - 1; \quad i_{X;Y} \in \mathcal{F}_{X \times Y}
  \]

- Reference distribution $R_{XY} = P_X \cdot P_Y$

- Subtract 1 to make zero-mean, (local approx. of PMI)

- Low-rank approximation
  \[
  f^*, g^* = \arg \min_{f \in \mathcal{F}_X^k, g \in \mathcal{F}_Y^k} \|i_{X;Y} - f \otimes g\|^2
  \]

\[\iff\]
\[
P_{XY}(x, y) \approx P_X(x)P_Y(y) \left(1 + \sum_{i=1}^{k} f_i^*(x) \cdot g_i^*(y)\right), \forall x, y
\]
A NN learns pairs of feature functions to form the "learned model"

\[ P_{Y|X} \propto P_Y(y) \cdot \left( 1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y) \right) \]
Feature Selection in Neural Networks

A NN learns pairs of feature functions to form the "learned model"

\[ P_{Y|X} \propto P_Y(y) \cdot \left( 1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y) \right) \]

Cross-Entropy as the loss metric.
Feature Selection in Neural Networks

A NN learns pairs of feature functions to form the "learned model"

\[ P_{Y|X} \propto P_Y(y) \cdot \left(1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y)\right) \]

Cross-Entropy as the loss metric.

Approximately solves

\[
\min_{f \in \mathcal{F}_X^k, g \in \mathcal{F}_Y^k} ||i_{X;Y} - f \otimes g||^2
\]
Feature Selection in Neural Networks

A NN learns pairs of feature functions to form the "learned model"

\[ P_{Y|X} \propto P_Y(y) \cdot \left( 1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y) \right) \]

Cross-Entropy as the loss metric.

Approximately solves

\[ \min_{f \in \mathcal{F}_X^k, g \in \mathcal{F}_Y^k} \| i_{X;Y} - f \otimes g \|^2 \]
More Directly: the H-Score Network

\[ \mathcal{H}(f, g) = \|i_{X;Y}\|^2 - \|i_{X;Y} - f \otimes g\|^2 \]

\[ = \text{cov}[f(X)g(Y)] - \frac{1}{2} \mathbb{E}[f^2(X)] \mathbb{E}[g^2(Y)] \]

- Model approx. equivalent to maximize the H-score;
More Directly: the H-Score Network

\[ \mathcal{H}(f, g) = \| i_{X;Y} \|^2 - \| i_{X;Y} - f \otimes g \|^2 \]

\[ = \operatorname{cov}[f(X)g(Y)] - \frac{1}{2} \mathbb{E}[f^2(X)]\mathbb{E}[g^2(Y)] \]

- Model approx. equivalent to maximize the H-score;
- When we don’t have the model but only have data samples, we can use empirical averages to approximate;
More Directly: the H-Score Network

$$\mathcal{H}(f, g) = \|i_{X,Y}\|^2 - \|i_{X,Y} - f \otimes g\|^2$$

$$= \text{cov}[f(X)g(Y)] - \frac{1}{2} \mathbb{E}[f^2(X)]\mathbb{E}[g^2(Y)]$$

- Model approx. equivalent to maximize the H-score;
- When we don’t have the model but only have data samples, we can use empirical averages to approximate;
- Individual NN-modules for feature functions $f(\cdot)$ and $g(\cdot)$
What’s Good About This?

- Directly metrics of feature functions is conceptually the “right” thing to do.
  - Learn without reconstruction (weak dependence example);
  - Learn functions, not predictors.
- Control of individual feature functions.
Control of Feature Functions: Put a Constraint

\[
\arg \min_{f,g : f \perp f} \|\text{i}_{\mathbf{X};\mathbf{Y}} - (f \otimes g)\|^2
\]

Examples: symmetry, band-limited, stability, etc., Can use a regulator, post-processing projection, etc.

A network that only generates \( f \) satisfying the constraint

Generic solution without the constraint
Control of Feature Functions: Put a Constraint

\[ X \xrightarrow{f} \mathcal{H} \xrightarrow{g} Y \]

constraint: \( f \perp \bar{f} \)

\[ \arg \min_{f,g : f \perp \bar{f}} \| i_{X,Y} - (f \otimes g) \|^2 \]

- Examples: symmetry, band-limited, stability, etc.,
Control of Feature Functions: Put a Constraint

\[
\begin{align*}
X \xrightarrow{f} \mathcal{H} \xrightarrow{g} Y \\
\text{constraint: } f \perp \bar{f}
\end{align*}
\]

\[
\text{arg min}_{f,g:\mathcal{f}} \| \mathbf{i}_{X,Y} - (f \otimes g) \|^2
\]

- Examples: symmetry, band-limited, stability, etc.,
- Can use a regulator, post-processing projection, etc.
Control of Feature Functions: Put a Constraint

\[ \text{constraint: } f \perp \bar{f} \]

\[ \arg \min_{f,g : f \perp \bar{f}} \| i_{X,Y} - (f \otimes g) \|^2 \]

- Examples: symmetry, band-limited, stability, etc.,
- Can use a regulator, post-processing projection, etc.
- A network that only generates \( f \) satisfying the constraint
Control of Feature Functions: Put a Constraint

\[ f \perp \bar{f} \]

\[ \text{constraint: } f \perp \bar{f} \]

\[ \arg \min_{f, g: f \perp \bar{f}} \| \mathbf{i}_{X,Y} - (f \otimes g) \|^2 \]

- Examples: symmetry, band-limited, stability, etc.,
- Can use a regulator, post-processing projection, etc.
- A network that only generates \( f \) satisfying the constraint
- Generic solution without the constraint
A Projection Operation

- Freeze one feature to get linear subspace

\[ \bar{g}^* = \arg \min_{\bar{g}} \| i_{X;Y} - \bar{f} \otimes \bar{g} \|^2 \]
A Projection Operation

Freeze one feature to get linear subspace

$$\bar{g}^* = \arg \min_{\bar{g}} \| i_{X;Y} - \bar{f} \otimes \bar{g} \|^2$$

Projection error \( (i_{X;Y} - \bar{f} \otimes \bar{g}^*) \) is orthogonal;
A Projection Operation

- Freeze one feature to get linear subspace
  \[
  \bar{g}^* = \arg\min_{\bar{g}} \|i_{X;Y} - \bar{f} \otimes \bar{g}\|^2
  \]

- Projection error \((i_{X;Y} - \bar{f} \otimes \bar{g}^*)\) is orthogonal;

- Low-rank approximation of this

\[
\min_{f,g} \| (i_{X;Y} - \bar{f} \otimes \bar{g}^*) - f \otimes g \|^2
\]

\[
\iff \min_{f,g} \| i_{X;Y} - (\bar{f} \otimes \bar{g}^* + f \otimes g) \|^2
\]
The Nested H-Score Network

\[ i_{X,Y} - \bar{f} \otimes \bar{g}^* \]
The Nested H-Score Network

\[ \bar{g}^* = \arg \min_{\bar{g}} \|i_{X;Y} - \bar{f} \otimes \bar{g}\|^2 \]
The Nested H-Score Network

\[ f^*, g^* = \arg \min_{f, g} \| \text{PMI} - \bar{f} \otimes \bar{g}^* - f \otimes g \|^2 \]

\[ = \arg \max_{f, g} \mathcal{H} \left( \begin{bmatrix} f \\ f \end{bmatrix}, \begin{bmatrix} \bar{g}^* \\ g \end{bmatrix} \right) \]
The Nested H-Score Network

\[
f^*, g^* = \arg \min_{f, g} \| \text{PMI} - \bar{f} \otimes \bar{g}^* - f \otimes g \|^2
\]

\[
= \arg \max_{f, g} \mathcal{H} \left( \begin{bmatrix} \bar{f} \\ f \end{bmatrix}, \begin{bmatrix} \bar{g}^* \\ g \end{bmatrix} \right)
\]
\[ Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W, \quad X_1 \in \{+1, -1\}, \quad X_2 \in 16 - \text{QAM} \]

- Input is \( Y \), output is \( \hat{X}_1 \).
- 3-way dependence: \( Y, X, S \)
- A decomposition

\[ I(Y; (S, X)) = I(Y; S) + I(Y; X|S) \]
The Decomposition of Multi-Variate Dependence

- The three-way dependence $i_{Y;(S,X)}$, reference $P_Y \cdot P_{S,X}$,
The Decomposition of Multi-Variate Dependence

- The three-way dependence \( i_{Y;(S,X)} \), reference \( P_Y \cdot P_{S,X} \),
- The Markov Component

\[
\pi_M = \arg\min_{\hat{i}:Y-S-X} \|i_{Y;(S,X)} - \hat{i}\|^2
\]
The Decomposition of Multi-Variate Dependence

- The three-way dependence $i_{Y;(S,X)}$, reference $P_Y \cdot P_{S,X}$,
- The Markov Component

$$\pi_M = \arg\min_{\hat{i} : Y \rightarrow S \rightarrow X} \|i_{Y;(S,X)} - \hat{i}\|^2$$

- Markov linear subspace, with conditional indep. constraints
The Decomposition of Multi-Variate Dependence

- The three-way dependence $i_{Y;(S,X)}$, reference $P_Y \cdot P_{S,X}$,
- The Markov Component
  
  $$\pi_M = \arg \min_{\hat{i} : Y \rightarrow S \rightarrow X} \| i_{Y;(S,X)} - \hat{i} \|^2$$

  Markov linear subspace, with conditional indep. constraints

- The Chain rule:
  
  $$\left\| i_{Y;(S,X)} \right\|^2_{I(Y;(S,X))} = \left\| \pi_M \right\|^2_{I(Y;S)} + \left\| \pi_C \right\|^2_{I(Y;X|S)}$$
We Know How to Do Projections

\[ g(Y) \rightarrow S \rightarrow \hat{X} \]

\[ i_{Y,(S,X)} \rightarrow \pi_C, \pi_M \]
We Know How to Do Projections

\[ \hat{X} = g(Y) \downarrow S \]

\[ \pi_C, \pi_M \]

\[ i_{Y,(S,X)} \]

\[ Y \rightarrow \text{Feature Extractor} \rightarrow \text{Inference} \rightarrow \hat{X} \]

\[ Y \]

\[ \bar{g} \rightarrow H \rightarrow \bar{f} \rightarrow S \]
We Know How to Do Projections

\[
Y \xrightarrow{g(Y)} S \xrightarrow{\pi} C
\]

\[
\hat{X} \leftarrow X \xrightarrow{f} \hat{S}
\]

\[
\bar{g} \quad \mathbb{H} \quad \bar{f}
\]

\[
Y \xrightarrow{g} \mathbb{H} \xrightarrow{f} \hat{X}
\]

\[
\bar{g} \quad \mathbb{H} \quad \bar{f}
\]

\[
Y \xrightarrow{g} \mathbb{H} \xrightarrow{f} \hat{X}
\]
We Know How to Do Projections

\[
Y \xrightarrow{g(Y)} \text{Feature Extractor} \xrightarrow{S} \text{Inference} \xrightarrow{\hat{X}} X
\]

\[
\hat{X} = g(Y) \circ S \circ X
\]
Assemble the Learning Results

We have learned 4 feature functions, $f, g, \tilde{f}, \tilde{g}$.
Assemble the Learning Results

- We have learned 4 feature functions, $f, g, \bar{f}, \bar{g}$
- Two good approximations:
  \[
P_{S|Y} \approx P_S \cdot (1 + f(s) \cdot g(y))
  \]
  \[
P_{SX|Y} \approx P_{SX} \cdot (1 + f(s) \cdot g(y) + f(s, x) \cdot g(y))
  \]
Assemble the Learning Results

- We have learned 4 feature functions, \( f, g, \bar{f}, \bar{g} \)
- Two good approximations:
  \[
P_{S|Y} \approx P_S \cdot (1 + \bar{f}(s) \cdot \bar{g}(y))
  
P_{SX|Y} \approx P_{SX} \cdot (1 + \bar{f}(s) \cdot \bar{g}(y) + f(s, x) \cdot g(y))
  \]
- Assemble into the predictor we need:
  \[
P_{X|S,Y} \approx P_X \cdot \left( 1 + \frac{f(s, x) \cdot g(y)}{1 + \bar{f}(s) \cdot \bar{g}(y)} \right)
  \]
Performance

BER vs. SINR

H-Score vs. Network Size
Some Special Cases

The "almost linear" case:
Harder Cases
Remarks

- Plug-and-play, no retrain, no adaptation, no few-shots, ...

Using DNNs in engineering problems:
- Learn feature functions, not predictors;
- Measure quality of features, not tasks;
- Features are high-dimensional geometric objects.
The Code
\[
\min_{f,g: f \perp f'} \left\| P_{XY} - P_X P_Y \cdot \left( 1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y) \right) \right\|_2
\]