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

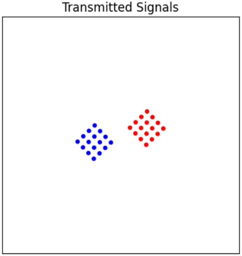
$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).

A Detection Problem

$$Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W$$

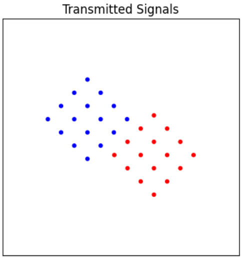
- $X_1 \in$ BPSK, $X_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters



A Detection Problem

$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters

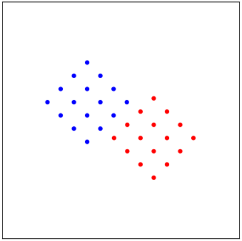


A Detection Problem

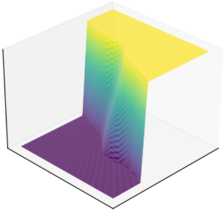
$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
    - Non-linear operation
    - highly depends on the parameters

Transmitted Signals



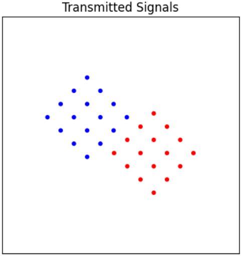
Decision Function

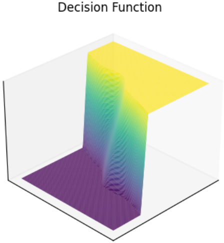


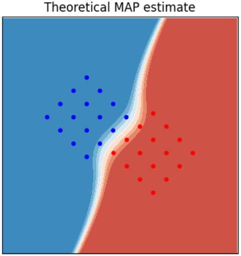
A Detection Problem

$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters



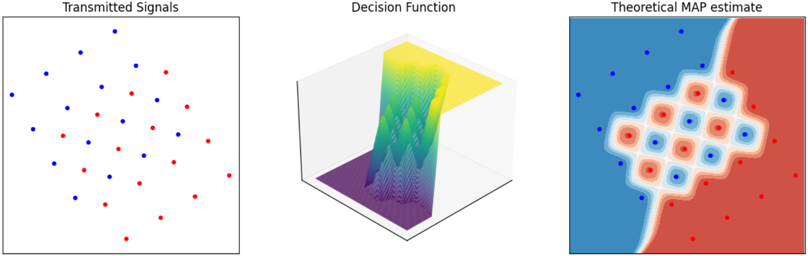




A Detection Problem

$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

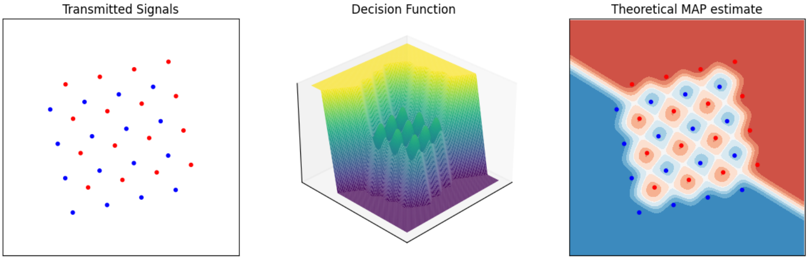
- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters



A Detection Problem

$$\mathsf{Y} = h_1 \cdot \mathsf{X}_1 + h_2 \cdot \mathsf{X}_2 + \mathsf{W}$$

- $\mathsf{X}_1 \in$ BPSK, $\quad \mathsf{X}_2 \in$ 16-QAM;
- Parameters h_1, h_2, σ_W^2 known at the receiver (CSIR).
- A good problem to solve with neural networks
  - Non-linear operation
  - highly depends on the parameters



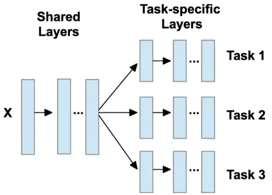
The Current Solutions (Complaints)

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

The Current Solutions (Complaints)

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

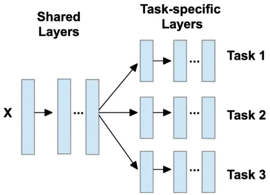
- Parameterized receiver:
  - Training with mixed samples
  - Parameters as input
  - Online vs. Offline



The Current Solutions (Complaints)

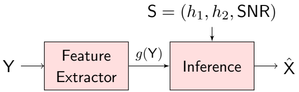
- The common "black box" issues: no guarantee, no metric, etc.

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



- Receiver with side-information, multi-varite dependence

$$\mathsf{Y}, \mathsf{X}, \underbrace{(h_1, h_2, \mathsf{SNR})}_{\mathsf{S}}$$



A Little Bit Tools

- Feature functions as vectors, $f : \mathcal{X} \to \mathbb{R}; \quad f \in \mathcal{F}_{\mathcal{X}}$
- Inner product

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

  - 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:

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

- Reference distribution $R_{\mathsf{XY}} = P_{\mathsf{X}} \cdot P_{\mathsf{Y}}$

Low Rank Approximation of Dependence

- Dependence between X and Y:

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

- Reference distribution $R_{\mathsf{XY}} = P_{\mathsf{X}} \cdot P_{\mathsf{Y}}$
- Subtract 1 to make zero-mean, (local approx. of PMI)

Low Rank Approximation of Dependence

- Dependence between X and Y:

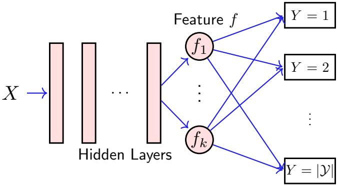
$$\mathfrak{i}_{\mathsf{X};\mathsf{Y}}(x,y) = \frac{P_{\mathsf{XY}}(x,y)}{P_{\mathsf{X}}(x)P_{\mathsf{Y}}(y)} - 1; \quad \mathfrak{i}_{X;Y} \in \mathcal{F}_{\mathcal{X}\times\mathcal{Y}}$$

- Reference distribution $R_{\mathsf{XY}} = P_{\mathsf{X}} \cdot P_{\mathsf{Y}}$
- Subtract 1 to make zero-mean, (local approx. of PMI)
- Low-rank approximation

$$f^*, g^* = \arg \min_{f \in \mathcal{F}_{\mathcal{X}}^k, g \in \mathcal{F}_{\mathcal{Y}}^k} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g \right\|^2$$

$$\iff \quad P_{\mathsf{XY}}(x,y) \approx P_{\mathsf{X}}(x)P_{\mathsf{Y}}(y) \left(1 + \sum_{i=1}^{k} f_i^*(x) \cdot g_i^*(y) \right), \forall x, y$$

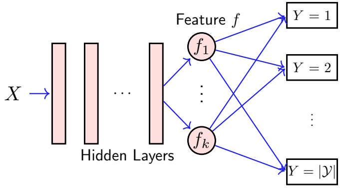
Feature Selection in Neural Networks



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

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

Feature Selection in Neural Networks

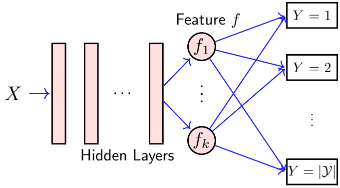


- A NN learns pairs of feature functions to form the "learned model"

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

- Cross-Entropy as the loss metric.

Feature Selection in Neural Networks



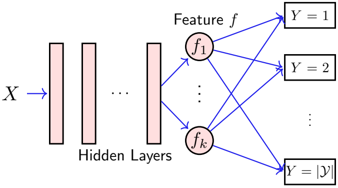
- A NN learns pairs of feature functions to form the "learned model"

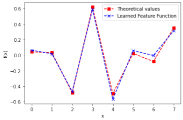
$$P_{\mathsf{Y}|\mathsf{X}} \propto P_{\mathsf{Y}}(y) \cdot \left(1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y) \right)$$

- Cross-Entropy as the loss metric.
- Approximately solves

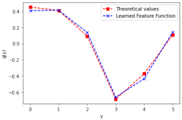
$$\min_{f \in \mathcal{F}_{\mathcal{X}}^k, g \in \mathcal{F}_{\mathcal{Y}}^k} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g \right\|^2$$

Feature Selection in Neural Networks





f_1



g_1

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

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

- Cross-Entropy as the loss metric.
- Approximately solves

$$\min_{f \in \mathcal{F}_{\mathcal{X}}^k, g \in \mathcal{F}_{\mathcal{Y}}^k} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g \right\|^2$$

More Directly: the H-Score Network

$$\mathscr{H}(f,g) = \|\mathsf{i}_{\mathsf{X};\mathsf{Y}}\|^2 - \|\mathsf{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g\|^2$$
$$= \mathrm{cov}[f(\mathsf{X})g(\mathsf{Y})] - \frac{1}{2}\mathbb{E}[f^2(\mathsf{X})]\mathbb{E}[g^2(\mathsf{Y})]$$

- Model approx. equivalent to maximize the H-score;

More Directly: the H-Score Network

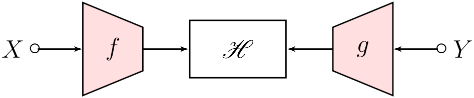
$$\mathscr{H}(f,g) = \|\mathfrak{i}_{\mathsf{X};\mathsf{Y}}\|^2 - \|\mathfrak{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g\|^2$$
$$= \mathrm{cov}[f(\mathsf{X})g(\mathsf{Y})] - \frac{1}{2}\mathbb{E}[f^2(\mathsf{X})]\mathbb{E}[g^2(\mathsf{Y})]$$

- 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;

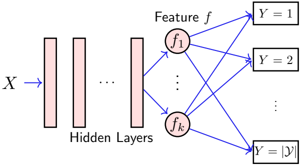
More Directly: the H-Score Network

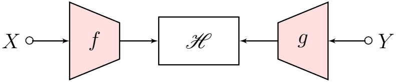
$$\mathscr{H}(f, g) = \|\mathrm{i}_{\mathsf{X};\mathsf{Y}}\|^2 - \|\mathrm{i}_{\mathsf{X};\mathsf{Y}} - f \otimes g\|^2$$
$$= \mathrm{cov}[f(\mathsf{X})g(\mathsf{Y})] - \frac{1}{2}\mathbb{E}[f^2(\mathsf{X})]\mathbb{E}[g^2(\mathsf{Y})]$$

- 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 $g(\cdot)$



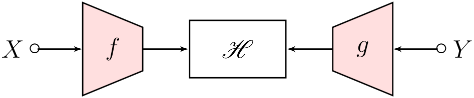
What's Good About This?





- 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.

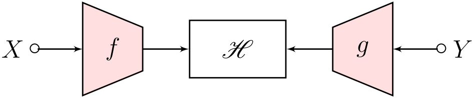
Control of Feature Functions: Put a Constraint



constraint: $f \perp \bar{f}$

$$\arg \min_{f, g : f \perp \bar{f}} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - (f \otimes g) \right\|^2$$

Control of Feature Functions: Put a Constraint

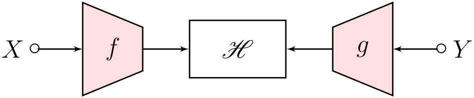


constraint: $f \perp \bar{f}$

$$\arg \min_{f,g:\, f \perp \bar{f}} \left\| \mathsf{i}_{\mathsf{X};\mathsf{Y}} - (f \otimes g) \right\|^2$$

- Examples: symmetry, band-limited, stability, etc.,

Control of Feature Functions: Put a Constraint

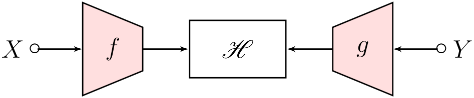


constraint: $f \perp \bar{f}$

$$\arg \min_{f,g: f \perp \bar{f}} \|\mathfrak{i}_{\mathsf{X};\mathsf{Y}} - (f \otimes g)\|^2$$

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

Control of Feature Functions: Put a Constraint

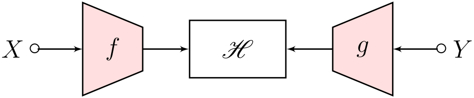


constraint: $f \perp \bar{f}$

$$\arg \min_{f,g:\, f \perp \bar{f}} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - (f \otimes g) \right\|^2$$

- 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

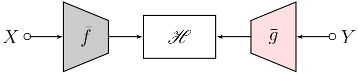


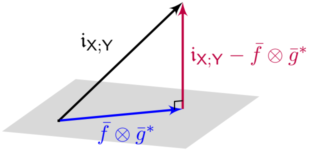
constraint: $f \perp \bar{f}$

$$\arg \min_{f,g:\, f \perp \bar{f}} \left\| \mathfrak{i}_{\mathsf{X};\mathsf{Y}} - (f \otimes g) \right\|^2$$

- 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

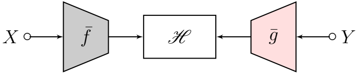


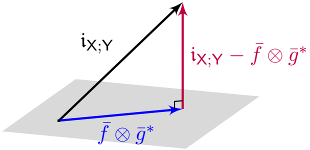


- Freeze one feature to get linear subspace

$$\bar{g}^* = \arg\min_{\bar{g}} \left\| \mathfrak{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g} \right\|^2$$

A Projection Operation



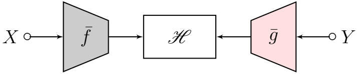


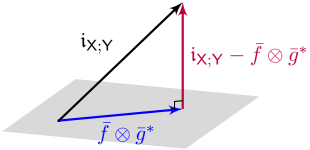
- Freeze one feature to get linear subspace

$$\bar{g}^* = \arg\min_{\bar{g}} \left\| \mathsf{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g} \right\|^2$$

- Projection error $\left(\mathsf{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g}^*\right)$ is orthogonal;

A Projection Operation





- Freeze one feature to get linear subspace

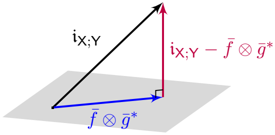
$$\bar{g}^* = \arg\min_{\bar{g}} \left\|\mathfrak{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g}\right\|^2$$

- Projection error $\left(\mathfrak{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g}^*\right)$ is orthogonal;
- Low-rank approximation of this

$$\min_{f,g} \left\|\left(\mathfrak{i}_{\mathsf{X;Y}} - \bar{f} \otimes \bar{g}^*\right) - f \otimes g\right\|^2$$

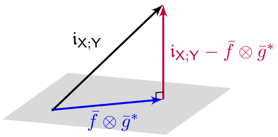
$$\Longleftrightarrow \min_{f,g} \left\|\mathfrak{i}_{\mathsf{X;Y}} - \left(\bar{f} \otimes \bar{g}^* + f \otimes g\right)\right\|^2$$

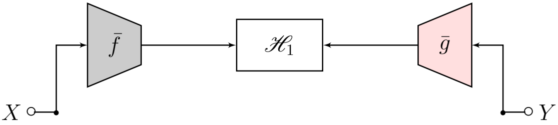
The Nested H-Score Network



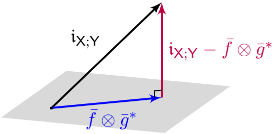
The Nested H-Score Network

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

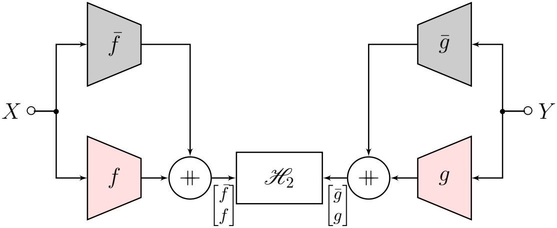




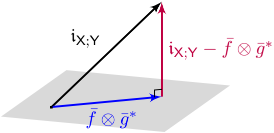
The Nested H-Score Network



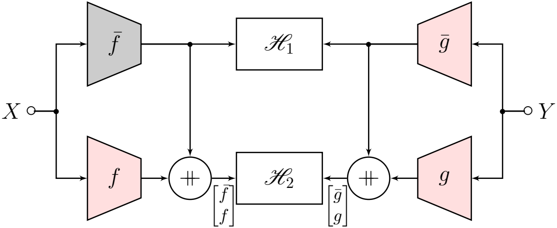
$$
\begin{aligned}
f^*, g^* &= \arg\min_{f,g} \left\|\mathrm{PMI} - \bar f \otimes \bar g^* - f \otimes g\right\|^2 \\
&= \arg\max_{f,g} \ \mathscr{H}\left(\begin{bmatrix}\bar f \\ f\end{bmatrix}, \begin{bmatrix}\bar g^* \\ g\end{bmatrix}\right)
\end{aligned}
$$



The Nested H-Score Network



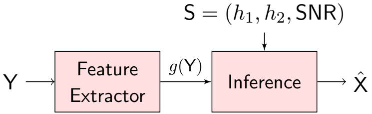
$$
\begin{aligned}
f^*, g^* &= \arg\min_{f,g} \left\| \mathrm{PMI} - \bar{f} \otimes \bar{g}^* - f \otimes g \right\|^2 \\
&= \arg\max_{f,g} \; \mathscr{H}\left(\begin{bmatrix} \bar{f} \\ f \end{bmatrix}, \begin{bmatrix} \bar{g}^* \\ g \end{bmatrix} \right)
\end{aligned}
$$



Back to The Problem

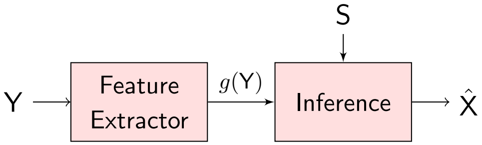
$$Y = h_1 \cdot X_1 + h_2 \cdot X_2 + W, \quad X_1 \in \{+1, -1\},\ X_2 \in 16 - \mathrm{QAM}$$

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



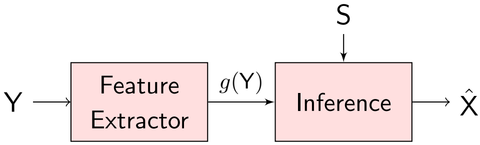
$$I(Y; (S, X)) = I(Y; S) + I(Y; X|S)$$

The Decomposition of Multi-Variate Dependence



- The three-way dependence $\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}$, reference $P_{\mathsf{Y}} \cdot P_{\mathsf{S},\mathsf{X}}$,

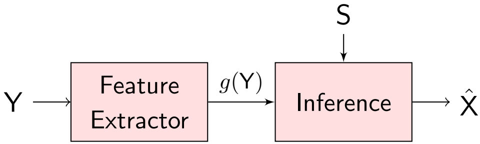
The Decomposition of Multi-Variate Dependence

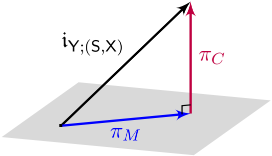


- The three-way dependence $\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}$, reference $P_{\mathsf{Y}} \cdot P_{\mathsf{S},\mathsf{X}}$,
- The Markov Component

$$\pi_M = \arg \min_{\widehat{\mathfrak{i}}\,:\mathsf{Y}-\mathsf{S}-\mathsf{X}} \|\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})} - \widehat{\mathfrak{i}}\|^2$$

The Decomposition of Multi-Variate Dependence



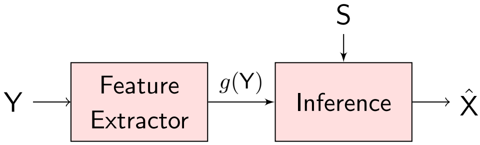


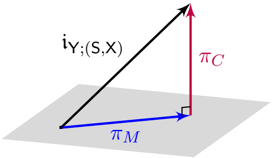
- The three-way dependence $\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}$, reference $P_{\mathsf{Y}} \cdot P_{\mathsf{S},\mathsf{X}}$,
- The Markov Component

$$\pi_M = \arg \min_{\widehat{\mathfrak{i}} : \mathsf{Y}-\mathsf{S}-\mathsf{X}} \|\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})} - \widehat{\mathfrak{i}}\|^2$$

- Markov linear subspace, with conditional indep. constraints

The Decomposition of Multi-Variate Dependence



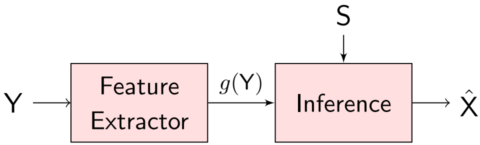


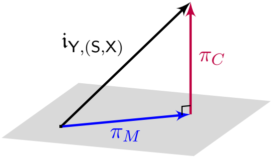
- The three-way dependence $\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}$, reference $P_{\mathsf{Y}} \cdot P_{\mathsf{S},\mathsf{X}}$,
- The Markov Component

$$\pi_M = \arg \min_{\widehat{\mathfrak{i}}\,:\mathsf{Y}-\mathsf{S}-\mathsf{X}} \|\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})} - \widehat{\mathfrak{i}}\|^2$$

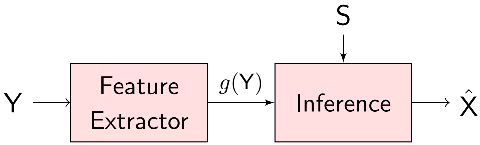
- Markov linear subspace, with conditional indep. constraints
- The Chain rule: $\underbrace{\|\mathfrak{i}_{\mathsf{Y};(\mathsf{S},\mathsf{X})}\|^2}_{I(\mathsf{Y};(\mathsf{S},\mathsf{X}))} = \underbrace{\|\pi_M\|^2}_{I(\mathsf{Y};\mathsf{S})} + \underbrace{\|\pi_C\|^2}_{I(\mathsf{Y};\mathsf{X}|\mathsf{S})}$

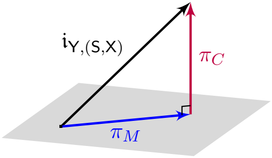
We Know How to Do Projections

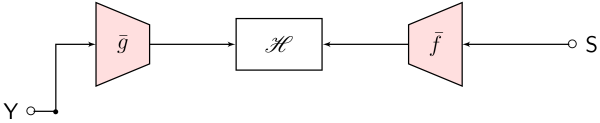




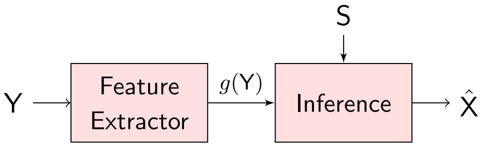
We Know How to Do Projections

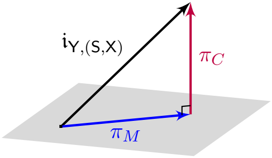


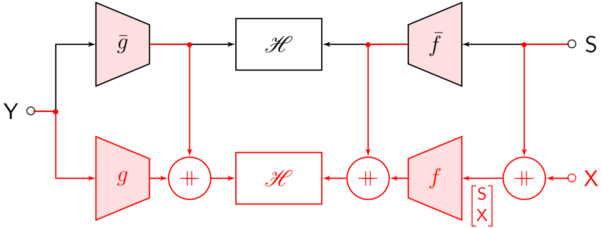




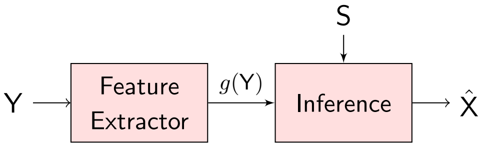
We Know How to Do Projections

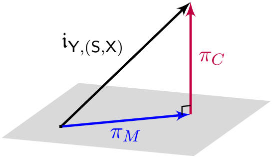


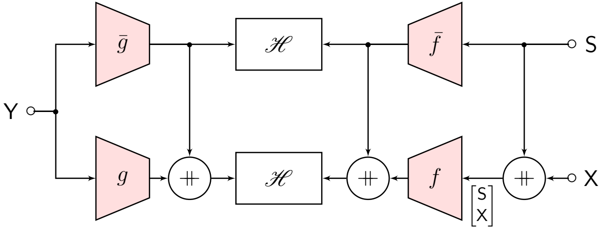




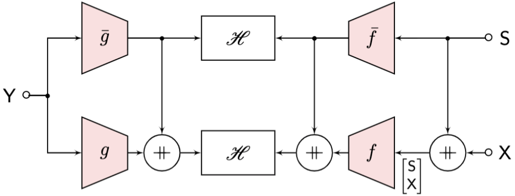
We Know How to Do Projections





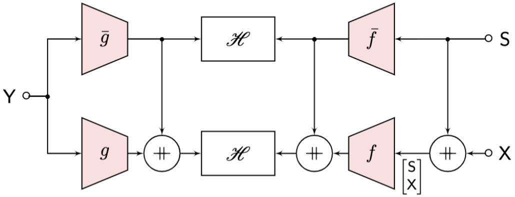


Assemble the Learning Results



- We have learned 4 feature functions, $f, g, \bar{f}, \bar{g}$

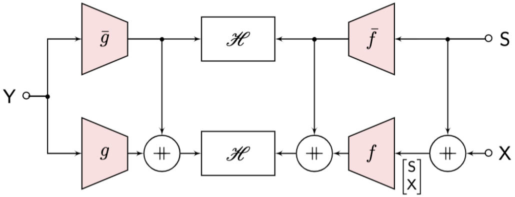
Assemble the Learning Results



- We have learned 4 feature functions, $f, g, \bar{f}, \bar{g}$
- Two good approximations:

$$P_{\mathsf{S}|\mathsf{Y}} \approx P_{\mathsf{S}} \cdot \left(1 + \bar{f}(s) \cdot \bar{g}(y)\right)$$
$$P_{\mathsf{SX}|\mathsf{Y}} \approx P_{\mathsf{SX}} \cdot \left(1 + \bar{f}(s) \cdot \bar{g}(y) + f(s, x) \cdot g(y)\right)$$

Assemble the Learning Results



- We have learned 4 feature functions, $f, g, \bar{f}, \bar{g}$
- Two good approximations:

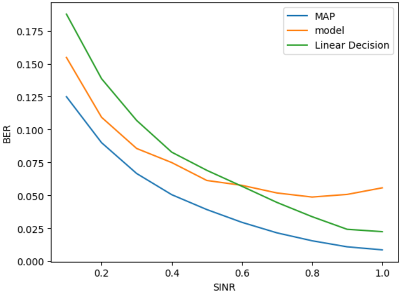
$$P_{\mathsf{S|Y}} \approx P_{\mathsf{S}} \cdot \left(1 + \bar{f}(s) \cdot \bar{g}(y)\right)$$

$$P_{\mathsf{SX|Y}} \approx P_{\mathsf{SX}} \cdot \left(1 + \bar{f}(s) \cdot \bar{g}(y) + f(s,x) \cdot g(y)\right)$$

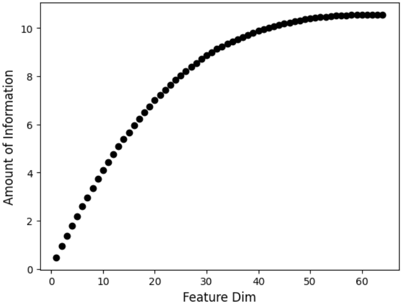
- Assemble into the predictor we need:

$$P_{\mathsf{X|S,Y}} \approx P_{\mathsf{X}} \cdot \left(1 + \frac{f(s,x) \cdot g(y)}{1 + \bar{f}(s) \cdot \bar{g}(y)}\right)$$

Performance



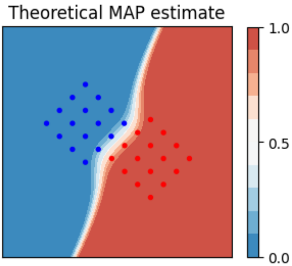
BER vs. SINR

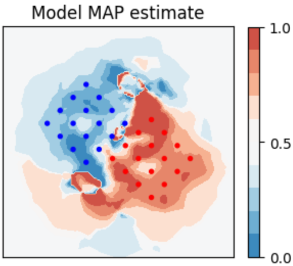


H-Score vs. Network Size

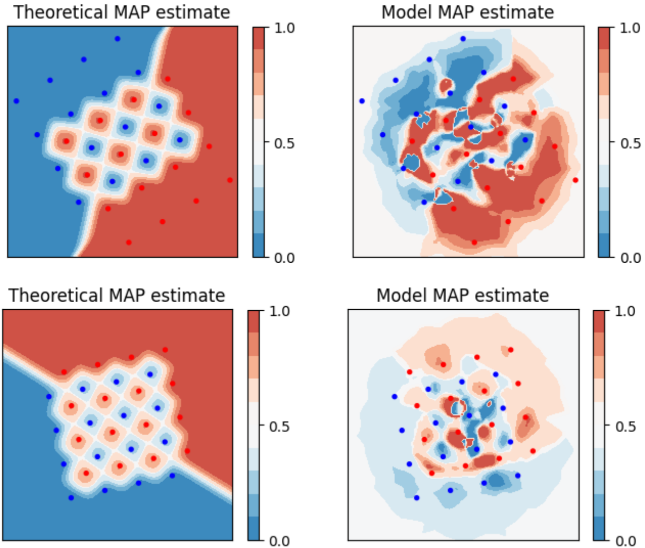
Some Special Cases

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

backup

$$\min_{f,g:\, f \perp \bar{f}} \left\| P_{XY} - \underbrace{P_X P_Y \cdot \left(1 + \sum_{i=1}^{k} f_i(x) \cdot g_i(y)\right)}_{\hat{P}_{XY}} \right\|^2$$