Parametric Information Bottleneck to Optimize Stochastic Neural Networks

The 2nd SAIL@UNIST Workshop

T.T. Nguyen and J. Choi

Statistical Artificial Intelligence Lab @ UNIST

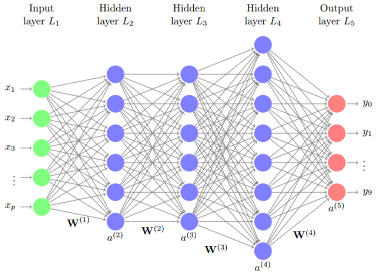
April 30, 2018

Contents

Contents

Motivation

- Deep neural networks (DNNs) have demonstrated competitive performance in several learning tasks;



[1]

- *Learning principle* is important to *guide the learning* in a model;
- The Maximum Likelihood Estimate (MLE) principle is widely used as a standard learning principle for DNNs;
- E.g., squared loss function, cross-entropy loss function.

[1] Figure credit: B.Efron and T.Hastie's

The MLE principle

$$\textit{model distribution} \xrightarrow{\text{match in KL divergence}} \textit{empirical data distribution},$$

MLE

$$\begin{aligned}\boldsymbol{\theta}_{ML} &= \mathop{arg\,max}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\sim p_D(\boldsymbol{x})}\left[\log p_{model}(\boldsymbol{x};\boldsymbol{\theta})\right] \\ &= \mathop{arg\,min}_{\boldsymbol{\theta}} D_{KL}\left[p_D(\boldsymbol{x}) \| p_{model}(\boldsymbol{x};\boldsymbol{\theta})\right] \\ &\approx \mathop{arg\,max}_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_{model}(\boldsymbol{x}_i;\boldsymbol{\theta})\end{aligned}$$

So...

- Does MLE effectively and sufficiently <u>exploit a neural network's representative power?</u>
- Is there any <u>better alternative?</u>

MLE principle for neural networks?

- MLE is generic for all models, NOT specifically tailored for neural networks;
- MLE sees neural network as a whole; hidden layers and neurons are not taken cared of *explicitly*;
- As a result, a neural network with *redundant information* in hidden layers may have a good distribution match in a training set BUT show a poor generalization in a test set

Contents

Our proposition

Our proposition

To better exploit a neural network's representation requires internal information within hidden layers to be considered explicitly during learning phase.

→ e.g., increasing layer-wise informativeness and compactness within a representation captures more regularities.

Contents

Information Bottleneck Principle [Tishby et al., 1999]

- A principled way of extracting relevant information in one variable, X about another variable, Y.
  - Knowing some values of X gives some guidance in prediction of Y, i.e., X provides information about Y (intrinsically via $p(X, Y)$).

Example 1

Whether X_1 or X_2 provides "more information" about Y? Given

$$\begin{cases} \mathcal{Y} = \mathcal{X}_1 = \mathcal{X}_2 = \{0, 1\} \\ p(y) = p(x_1) = p(x_2) = 0.5 \; \forall y \in \mathcal{Y}, x_1 \in \mathcal{X}_1, X_2 \in \mathcal{X}_2, \end{cases}$$

$$p(y, x_1) = \begin{cases} 1 & \text{if } y = x_1 = 0 \\ 0 & \text{otherwise,} \end{cases} ; p(y, x_2) = \frac{1}{4} \; \forall y, x_2$$

Intuition $\rightarrow$ X_2 provides ***no*** information about Y; X_1 provides ***all*** information about Y.

Information Bottleneck Principle [Tishby et al., 1999]

IB core idea

Encode X into an intermediate representation,

$$X \xrightarrow{p(z|x)} Z,$$

in such a way that Z preserves as much of **relevant information** about Y as possible.

Markov assumption

$$Y \to X \to Z, \text{ i.e., } Y \perp Z|X$$

Q:

- How to quantify relevant information in Z?
- How to quantify the representation quality of Z?

A: **mutual information**!

Mutual Information

Definition (Mutual Information)

$$I(Z,Y) = \int p(\boldsymbol{z},\boldsymbol{y}) \log \frac{p(\boldsymbol{z},\boldsymbol{y})}{p(\boldsymbol{z})p(\boldsymbol{y})} d\boldsymbol{z} d\boldsymbol{y}$$

Intuition

$I(Z,Y) =$ the amount of information that Z contains about Y.

Back to **Example 1**:

$I(X_1, Y) = 2 = H(X) \to X_1$ provides ALL information about Y

$I(X_2, Y) = 0 \to X_2$ contains no information about Y

Information Bottleneck Problem

Definition

- **Compression** (measure) = representation complexity = $I(Z, X)$
- **Relevance** = predictive power = $I(Z, Y)$

IB Problem

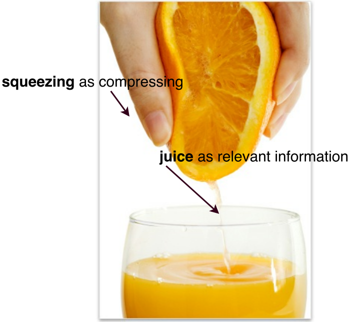
= Trade-off between **compression** and **relevance**

Why compression-relevance tradeoff? → think of this analogy:

- representation Z = some container (e.g., orange)
- $I(Z, X)$ = the container's capacity (e.g., the orange's volume)
- $I(Z, Y)$ = amount of useful items in the container (e.g., orange juice)

Compression-Relevance tradeoff

Orange's Analogy:



- Reducing capacity may cause loss of relevant information
- Larger capacity gives room for more relevant information

Rigorously,

Extreme 1: Maximum Compression $\rightarrow$ Minimum Relevance

$I(Z, X) = 0 \implies Z \perp X \implies Z \perp Y \implies I(Z, Y) = 0$

Extreme 2: Minimum Compression $\rightarrow$ Maximum Relevance

$I(Z, X) = H(X) \implies Z = f(X) \implies I(Z, Y) = I(X, Y)$
where $f(.)$ is a bijective (reversible) function

Proofs: It follows the Markov chain assumption, rules of probability, and definitions of mutual information.

IB Optimization Problem

Compression-Relevance Function [Slonim and Weiss, 2002]

$$R(D) := \min_{Z=Z(X):I(Z,Y)\geq D} I(Z,X),$$

- Equivalent to the Lagrangian multiplier (to be minimized):

Lagrangian multiplier [Tishby et al., 1999]

$$\mathcal{L}_{IB} = I(Z,X) - \beta I(Z,Y)$$

where β is a positive Lagrange multiplier that determines the trade-off between compression and relevance in Z

$\rightarrow$ is in general **NP hard**!

Compression-Relevance tradeoff in action

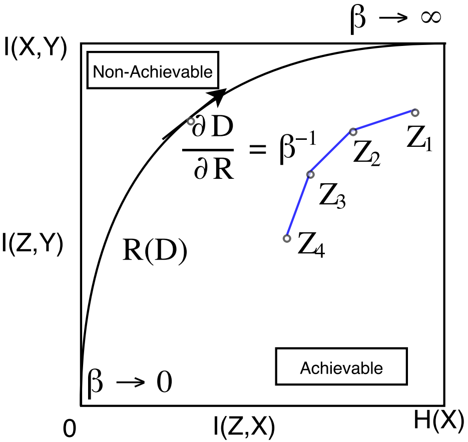


Figure: The information curve $R(D)$ is a non-decreasing concave curve that divides the information plane into achievable region and non-achievable region.

About the IB principle

- No model assumption is required
- Assumption: $Y \to X \to Z$ forms a Markov chain, and $p(X, Y)$ is known (though one can simply use empirical joint distribution $\hat{p}(X, Y)$ in practice)
- Exact solution is known ([Tishby et al., 1999]), but the solution is implicit and highly non-linear that limits its applications in general cases $\to$ NP-hard in general!

Contents

Parametric Information Bottleneck (PIB)

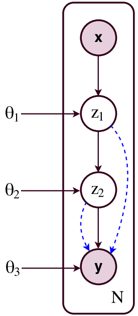
What is this?

$$\boxed{\text{PIB} = \text{neural network architecture} + \text{IB approximations},}$$

- A learning framework to **better exploit the neural network's intermediate representation**
- Learning a neural network's representation is now interpreted as jointly and explicitly **inducing compression and relevance into every level of neural networks**

Our perspective: Neural network as sequential encodings

Neural network architecture = **series of stochastic encoders** that sequentially modify the original data space



- $Z_{l-1} \xrightarrow{p(z_l|z_{l-1})} Z_l$ where Z_l is the l^{th} layer [2]
- **Layer transform** $p(\boldsymbol{z}_{l+1}|\boldsymbol{z}_l)$: induces a soft partitioning of the space, $\mathcal{Z}_l$, into a new encoding space, $\mathcal{Z}_{l+1}$
- Markov Assumption:

$$Y \to X \to Z_l \to Z_{l+1}$$

[2] Stochastic Neural Networks

Why is this perspective useful?

1. Property 1:

   Enables easy sampling of Z_l given X.

   $\rightarrow$ useful for estimating mutual information in the later parts.

2. Property 2:

   Makes a lossy representation (due to Data-Processing Inequality):

   $$H(X) \geq I(X, Z_l) \geq I(X, Z_{l+1})$$
   $$I(X, Y) \geq I(Y, Z_l) \geq I(Y, Z_{l+1})$$

Property 2: Being lossy is useful (if we can <u>control</u> it)

- The original data space, $\mathcal{X}$ is continuous $\rightarrow$ requires infinite precision to represent it precisely
- **preciseness** $\neq$ **usefulness**!
- Let it (representation) be lossy, but <u>in a beneficial manner</u>: **lossy in irrelevant information and gain in relevant information**

Q:

How to *mathematically* "implement" (quantify) this idea?

A: Borrow the idea of the IB principle!

Parametric Information Bottleneck

PIB objective

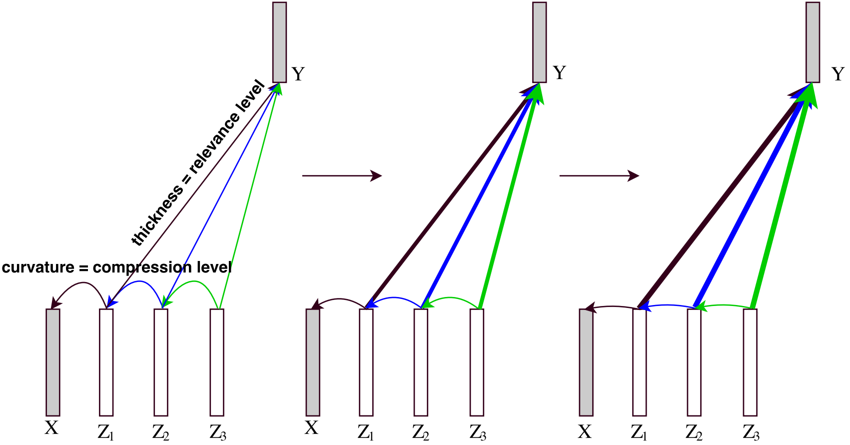
$$\mathcal{L}_{PIB}(Z) := \mathcal{L}_{PIB}(\boldsymbol{\theta}) := \sum_{l=0}^{L} \left[\beta_l^{-1} I(Z_l, Z_{l-1}) - I(Z_l, Y) \right]$$

where the layer-specific Lagrangian multiplier β_l^{-1} controls the tradeoff between relevance and compression in each bottleneck.

- $\mathcal{L}_{PIB}$ = a joint version of the theoretical analysis in [Tishby and Zaslavsky, 2015]
- $\mathcal{L}_{PIB}$ **encourages compression and relevance simultaneously** within each layer and within the entire network (i.e., *super layer* or l = 0) as a whole.

PIB Intuition: Tightening the information knots

Think of minimizing the PIB objective as **tightening the information knots** of a network architecture:



Solving for $\min J_{PIB}(Z)$?

- Much harder than the original IB problem since the information terms are inter-dependent
- Still inherits the intractability of mutual information

Intractability of Mutual Information

- **Relevance**:

$$I(Z,Y) = H(Y) - H(Y|Z)$$
$$H(Y|Z) = -\int p(\boldsymbol{y},\boldsymbol{z}) \log p_{true}(\boldsymbol{y}|\boldsymbol{z}) d\boldsymbol{y} d\boldsymbol{z}$$

Relevance decoder (def):

$$\boxed{p_{true}(\boldsymbol{y}|\boldsymbol{z}_l) = \int p_D(\boldsymbol{x},\boldsymbol{y}) \frac{p(\boldsymbol{z}_l|\boldsymbol{x})}{p(\boldsymbol{z}_l)} d\boldsymbol{x}}$$

$\rightarrow$ intractable!

- **Compression**:

$$I(Z_l, Z_{l-1}) = H(Z_l) - H(Z_l|Z_{l-1})$$
$$H(Z_l) = -\int p(\boldsymbol{z}_l) \log p(\boldsymbol{z}_l) d\boldsymbol{z}_l$$
$$p(\boldsymbol{z}_l) = \mathbb{E}_{p(\boldsymbol{z}_{l-1})}[p(\boldsymbol{z}_l|\boldsymbol{z}_{l-1})]$$

$\rightarrow$ remains exponentially in terms of $|\mathcal{Z}_{l-1}|$

Approximate Relevance

- Use ***variational approximation*** to derive a lower bound on $I(Z_l, Y)$ (as in [Alemi et al., 2016])
- Variational approximation = to posit a family of distributions and then find a member of that family that is closest to the intractable distribution in terms of KL divergence
- However,

> here we propose to re-use the network architecture to define variational distributions

Variational Approximation to Mutual Information

$$H(Y|Z_l) = -\int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log \overset{\text{relevance decoder}}{\boxed{p_{true}(\mathbf{y}|\mathbf{z}_l)}} d\mathbf{y} d\mathbf{z}_l$$

$$= -\int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log \overset{\textit{variational}\text{ relevance decoder}}{\boxed{p_v(\mathbf{y}|\mathbf{z}_l)}} d\mathbf{y} d\mathbf{z}_l - \int p(\mathbf{z}_l) D_{KL}[p_{true}(\mathbf{y}|\mathbf{z}_l) || p_v(\mathbf{y}|\mathbf{z}_l)] d\mathbf{z}_l$$

$$\leq -\int p(\mathbf{z}_l) p_{true}(\mathbf{y}|\mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y} d\mathbf{z}_l$$

$$= -\int p(\mathbf{y}, \mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y} d\mathbf{z}_l$$

$$= -\int p(\mathbf{x}, \mathbf{y}, \mathbf{z}_l) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y} d\mathbf{z}_l d\mathbf{x}$$

$$= -\int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_l|\mathbf{x}, \mathbf{y}) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{y} d\mathbf{z}_l d\mathbf{x}$$

$$= -\int p_D(\mathbf{x}, \mathbf{y}) p(\mathbf{z}_l|\mathbf{x}) \log p_v(\mathbf{y}|\mathbf{z}_l) d\mathbf{z}_l d\mathbf{x} d\mathbf{y}$$

$$= -\mathbb{E}_{p_D(\mathbf{x},\mathbf{y})} \left[\mathbb{E}_{p(\mathbf{z}_l|\mathbf{x})} [\log p_v(\mathbf{y}|\mathbf{z}_l)] \right] =: \tilde{H}(Y|Z_l)$$

Approximate Relevance

- Re-use the higher-level part of the existing network architecture at each layer to define the *variational relevance decoder* for that layer, i.e.,

$$p_v(\boldsymbol{y}|\boldsymbol{z}_l) = p(\boldsymbol{y}|\boldsymbol{z}_l)$$

  where $p(\boldsymbol{y}|\boldsymbol{z}_l)$ is determined by the network architecture
- Then,

$$p_v(\boldsymbol{y}|\boldsymbol{z}_l) = \int \prod_{i=l}^{L+1} p(\boldsymbol{z}_{i+1}|\boldsymbol{z}_i) d\boldsymbol{z}_L \ldots d\boldsymbol{z}_{l+1} = \mathbb{E}_{p(\boldsymbol{z}_L|\boldsymbol{z}_l)}\left[p(\boldsymbol{y}|\boldsymbol{z}_L)\right]$$

Variational Conditional Relevance (VCR) (def.)

$$\tilde{H}(Y|Z_l) = -\mathbb{E}_{p_D(\boldsymbol{x},\boldsymbol{y})}\left[\mathbb{E}_{p(\boldsymbol{z}_l|\boldsymbol{x})}\left[\log \mathbb{E}_{p(\boldsymbol{z}_L|\boldsymbol{z}_l)}\left[p(\boldsymbol{y}|\boldsymbol{z}_L)\right]\right]\right]$$

$\rightarrow$ Use Monte Carlo sampling to estimate VCR because sampling is straightforward in the network!

VCR vs. MLE

Proposition 1

$\tilde{H}(Y|Z_{l=0})$ = negative log-likelihood (NLL) function

Proposition 2

VCR for the *compositional* bottleneck $Z = (Z_1, Z_2, ..., Z_L)$ places an upper bound on the NLL function, i.e.,

$$\underset{\substack{\uparrow\\\text{VCR for } Z_L}}{\boxed{\tilde{H}(Y|Z_L)}} = \underset{\substack{\uparrow\\\text{VCR for } Z}}{\boxed{\tilde{H}(Y|Z)}} \geq -\mathbb{E}_{p_D(\boldsymbol{x},\boldsymbol{y})}\left[\log p(\boldsymbol{y}|\boldsymbol{x})\right]$$

Proof of Prop. 1: a direct result of VCR.

Proof of Prop. 2: It follows the Markov assumption and Jensen's inequality

VCR vs. MLE: Interpretation

- VCR for the entire network (i.e., at super layer $l = 0$) $=$ NLL of $p(\boldsymbol{y}|\boldsymbol{x})$
- VCR for layer l $=$ NLL of $p(\boldsymbol{y}|\boldsymbol{z}_l)$

Approximate Compression

- Decomposition: $I(Z_1, Z_0) = H(Z_1) - H(Z_1|Z_0)$
- Conditional entropy:

$$H(Z_1|Z_0) = \mathbb{E}_{p(\boldsymbol{z}_0)} \left[\sum_{i=1}^{N_1} H(Z_{1,i}|Z_0 = \boldsymbol{z}_0) \right]$$

- Resort to <u>empirical samples of $\boldsymbol{z}_1$</u> generated by Monte Carlo sampling to estimate the entropy:

$$H(Z_1) = -\mathbb{E}_{p(\boldsymbol{z}_1)}[\log p(\boldsymbol{z}_1)] \approx -\frac{1}{M} \sum_{k=1}^{M} \log p(\boldsymbol{z}_1^{(k)}) =: \hat{H}_{MLE}(Z_1)$$

$$\boldsymbol{z}_1^{(k)} \sim p(\boldsymbol{z}_1) = \mathbb{E}_{p(\boldsymbol{z}_0)}[p(\boldsymbol{z}_1|\boldsymbol{z}_0)]$$

$\rightarrow$ However, <u>$\log p(\boldsymbol{z}_1)$ is numerically unstable</u> since $p(\boldsymbol{z}_1) \approx 0$ for high-dimensional $\boldsymbol{z}_1$!

Approximate Compression

Numerically-stable estimation of entropy:

$$H(Z_1) \underset{}{\overset{\text{Jensen's inequality}}{\overset{\downarrow}{\boxed{\leq}}}} -\mathbb{E}_{p(\boldsymbol{z}_1)}\left[\mathbb{E}_{p(\boldsymbol{z}_0)}\left[\log p(\boldsymbol{z}_1|\boldsymbol{z}_0)\right]\right] := \tilde{H}(Z_1)$$

$\rightarrow$ more stable since $p(\boldsymbol{z}_1|\boldsymbol{z}_0)$ factorizes!

Monte Carlo Sampling

$$s = \int p(x)f(x)dx = \mathbb{E}_{p(x)}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) =: \hat{s}_N$$

where $x^{(i)} \sim p(x)$.

- Unbiased mean: $\mathbb{E}[\hat{s}_N] = s$
- Variance: $Var[\hat{s}_N] = \frac{Var[f(x)]}{N}$

Application to Network Architecture Domain

We applied our PIB framework to feed-forward neural networks with binary bottlenecks,

- $Z_l \in \{0, 1\}^{|\mathcal{Z}_l|}$
- stochastic encoder $p(\boldsymbol{z}_{l+1}|\boldsymbol{z}_l)$ is simply modeled by network weights and **sigmoid** function for **binary** $\boldsymbol{z}_{l+1}$
- $H(Z_{1,i}|Z_0 = \boldsymbol{z}_0)$ can be computed analytically

Q:
Derivative wrt discrete-valued variables is impossible. How to compute gradients via sampling of binary variables?
A:
Use Raiko gradient estimator![3]

[3] *Updated* (30/04/2018): *Our modified PIB futher uses Straight-Through and REINFORCE gradient estimator*

Raiko gradient estimator [Raiko et al., 2014]

Idea: Decompose the binary variable into the sum of a deterministic term and a noise term (to be ignored when taking derivatives)

$$\boldsymbol{z}_l = (z_{l,1}, z_{l,2}, ..., z_{l,k}),$$
$$\boldsymbol{a}^{(l)} = (a_1^{(l)}, a_2^{(l)}, ..., a_k^{(l)}) = W^{(l)}\boldsymbol{z}_{l-1} + \boldsymbol{b}_i^{(l)}$$
$$p(z_{l,i} = 1|\boldsymbol{z}_{l-1}) = \sigma(a_i^{(l)})$$

Rewrite $z_i^{(l)}$ as if it is continuous: $\boxed{z_{l,i} = \sigma(a_i^{(l)}) + \epsilon_i^{(l)}}$
where

$$\epsilon_i^{(l)} = \begin{cases} 1 - \sigma(a_i^{(l)}) \text{ with probability } \sigma(a_i^{(l)}) \\ -\sigma(a_i^{(l)}) \text{ with probability } 1 - \sigma(a_i^{(l)}) \end{cases}$$

Gradient-based Algorithm: GRAD-PIB

Algorithm 1 GRAD-PIB

Input: Labeled training dataset S_0.
$\boldsymbol{\theta} \leftarrow$ Initialize the parameters.
repeat
  $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i=1}^{N} \leftarrow N$ samples randomly drawn from S_0.
  for $i = 1$ **to** L **do**
    $S_i \leftarrow$ Generate M samples of $\boldsymbol{z}_i$ given each sample of $\boldsymbol{z}_{i-1}$ from S_{i-1}.
  end for
  $\tilde{\mathcal{L}}_{PIB}(\boldsymbol{\theta}) \leftarrow$ Equations (13), (14), (19), and $\{S_i\}_{i=0}^{L}$.
  $\boldsymbol{g} \leftarrow \frac{\partial}{\partial \boldsymbol{\theta}} \tilde{\mathcal{L}}_{PIB}(\boldsymbol{\theta})$ using the Raiko estimator.
  $\boldsymbol{\theta} \leftarrow$ Update $\boldsymbol{\theta}$ using $\boldsymbol{g}$ and SGD.
until convergence condition.
Output: $\boldsymbol{\theta}$

$\rightarrow$ **Main advantage**: only one pass of (hierarchical ancestral) sampling for computation of all information terms in J_{PIB}

Contents

Statistical Artificial Intelligence Laboratory @UNIST

Experiment Setup

- Used the **same** neural network architecture for PIB and MLE;
- The architecture guided by the MLE is named as Stochastic Feed-forward Neural Networks (SFNN) (e.g., [Tang and Salakhutdinov, 2013])
- Three tasks in the MNIST dataset [LeCun et al., 1998]:
  - **Image classification**: to evaluate generalization,
  - **Odd-even decision problem**: to visualize the network's representation in terms of mutual information,
  - **Multi-modal learning**: to evaluate capability to learn a multi-modal distribution.

Exp. 1: Classification

- **Task**: To classify digit images of the MNIST dataset into 10 categories
- **Input**: A 28x28 gray (digit) image
- **Output**: One of 10 labels
- **Architecture**: $784 \times 512 \times 512 \times 10$

Table: Test error (%) for permutation-invariant MNIST. In the TRAIN/TEST column, D and S stand for deterministic and stochastic, respectively.

MODEL	PRINCIPLE	TRAIN / TEST	ERROR (%)
A	MLE	D / D	1.73
B	MLE	D / S	2.30 ± 0.070
C	MLE	S / S	1.94 ± 0.036
D	MLE	S / D	1.88
E	PIB	S / S	$\mathbf{1.47 \pm 0.034}$
F	PIB	S / D	**1.46**

Classification: stochastic vs deterministic prediction

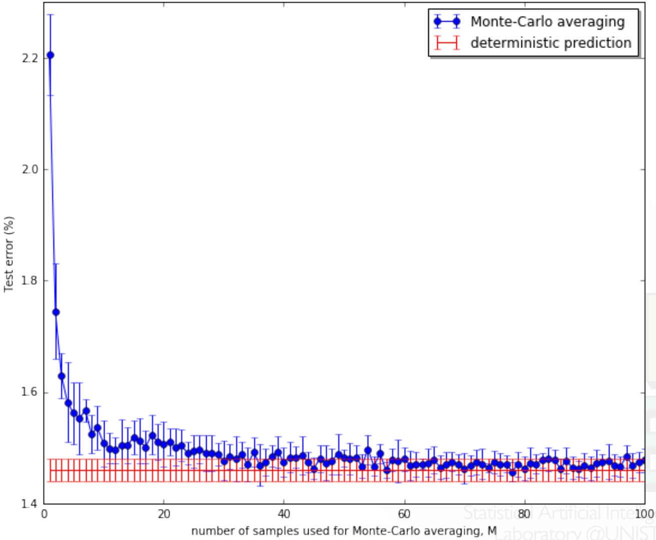


Figure: A comparison of Monte-Carlo averaging and deterministic prediction of PIB.

Classification: To conclude

- PIB outperforms MLE by large margin (for the considered architecture, 1.47% vs. 1.94%)
- The generalization of PIB is attributed to, of course, compression (as a regularization), and its ability to encode more relevant information into the representation Z (next experiment).

Exp. 2: Learning dynamics

- **Task**: To determine if the digit in an MNIST image is odd or even.
- **Input**: A 28×28 gray (digit) MNIST image
- **Output**: 0/1 (i.e., even/odd)
- **Architecture**: $784 \times 10 \times 10 \times 10 \times 1$
- **Purpose**: To visualize a neural network's representation in the learning process

$\rightarrow$ Used a small network architecture so that we can compute $I(Z_l, X)$ and $I(Z_l, Y)$ precisely, and plot them on the *information plane* over training epochs

Learning dynamics

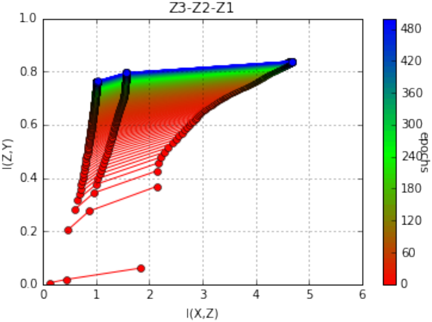


Figure: MLE.

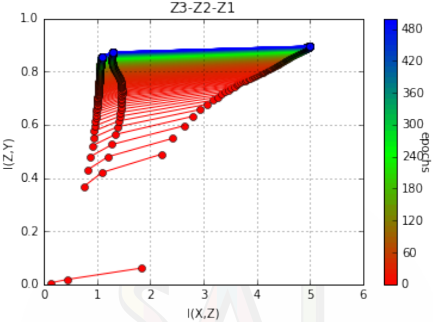


Figure: PIB.

Figure: Each point represents a hidden layer while the color indicates training epochs. Because of the Markov assumption, we have $H(X) \geq I(Z_i, X) \geq I(Z_{i+1}, X)$ and $I(X, Y) \geq I(Z_l, Y) \geq I(Z_{l+1}, Y)$.

Learning dynamics: PIB has selective encoding

- PIB selectively encodes information while MLE does not.
- Compression is not dominant (only layer 2 goes under apparent compression phase)
  - Probably our compression estimate is not yet good enough!

Exp. 3: Multi-modal learning

- **Task**: To predict the lower haft of a digit image given the upper haft
- **Input**: The upper half of a digit image
- **Output**: The lower half of a digit image
- **Architecture**: $392 \times 512 \times 512 \times 392$
- **Purpose**: To evaluate the ability of PIB and MLE to help a neural network model a structured output space (i.e., 1-to-many mapping, e.g., a mixture of Gaussian, mixture of Bernoulli)

Multi-modal learning

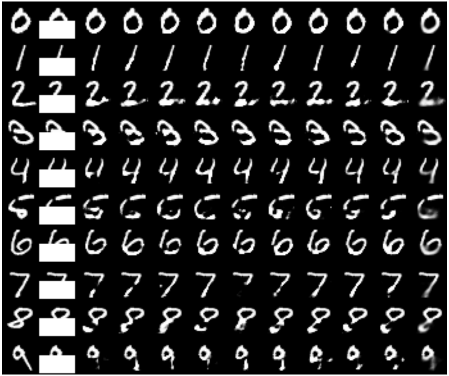


Figure: MLE (after 1000 epochs).

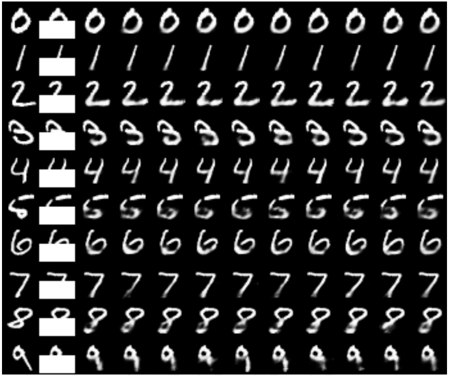


Figure: PIB (after 300 epochs).

Figure: The leftmost column is the original MNIST test digit images followed by the masked out ones and 9 samples. The rightmost column is obtained by averaging over all generated bottleneck samples drawn from the prediction.

Multi-modal learning: To conclude

- PIB can model multi-modal distributions better than MLE (e.g., the generated digits in PIB are more complete and recognizable)
- Multi-modal learning requires information of different modes from all layers:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z} = \int p(\boldsymbol{y}|\boldsymbol{z})p(\boldsymbol{z}|\boldsymbol{x})d\boldsymbol{z} = \int \prod_{l=1}^{L+1} p(\boldsymbol{z}_l|\boldsymbol{z}_{l-1})d\boldsymbol{z}$$

  while PIB assures **selective encodings** occur at all levels.
- PIB **exploits** the neural network's **representation** to an extent that MLE cannot

Contents

Summary

- provided arguments about inefficiency of MLE for learning neural networks
- encouraged a re-thinking of a new learning principle that is specifically tailored for neural networks;
- proposed PIB framework, to
  better exploit a neural network's representation by jointly and explicitly considering representation complexity and predictive power for every layer;
- PIB is followed by *GRAD-PIB*, the first algorithm that learns all parameters using Information Bottleneck principle;
- supported the effectiveness and robustness of our PIB with the qualitative empirical results.

Limitations

- Hierarchical ancestral sampling in *GRAD-PIB* requires an exponential number of samples as the number of hidden layers grow, which currently causes computational burden in large neural network architectures;
- Since our algorithm is of gradient-based learning, it inherits the weakness of gradient-based learning which fails to guarantee the theoretical learning bound and underestimate the variance of the underlying data distribution; this property makes it difficult to analyze neural network architectures;
- Here only fully-connected feed-forward architecture with binary hidden layers are considered and larger neural network architecture is not yet exploited.

Acknowledgements

We thank Prof. Se Young Chun, Prof. Jun Moon, SAIL@UNIST members, and anonymous reviewers for the valuable comments.

Thank you for listening!
Q & A Section

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016).
Deep variational information bottleneck.
CoRR, abs/1612.00410.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

Raiko, T., Berglund, M., Alain, G., and Dinh, L. (2014).

Techniques for learning binary stochastic feedforward neural networks.
CoRR, abs/1406.2989.

Slonim, N. and Weiss, Y. (2002).
Maximum likelihood and the information bottleneck.
In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 335–342.

Tang, Y. and Salakhutdinov, R. (2013).
Learning stochastic feedforward neural networks.
In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting*

held December 5-8, 2013, Lake Tahoe, Nevada, United States., pages 530–538.

Tishby, N., Pereira, F. C., and Bialek, W. (1999).
The information bottleneck method.
In Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing.

Tishby, N. and Zaslavsky, N. (2015).
Deep learning and the information bottleneck principle.
pages 1–5.