Neural Network Learning

Jen-Tzung Chien

National Chiao Tung University

Spatial Temporal Modeling Workshop (STM 2016)
The Institue of Statistical Mathematics

Content

- Part I: Tensor Factorized Network Network
  - Tensor factorization
  - Multilayer perceptron
- Part II: Domain Adaptive Neural Network
  - Semi-supervised learning for domain adaptation
  - Multi-task network
- Part III: Bayesian Unfolding Inference Network
  - Variational inference
  - Unfolding network

Part I: Tensor Factorized Neural Network

Spatial Temporal Modeling Workshop (STM 2016)

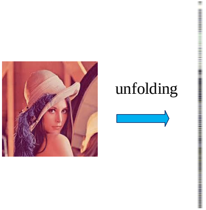
Outline

Introduction

- Neural networks are known as powerful learning machine for supervised learning
  - spatial information: convolutional neural network
  - temporal information: recurrent neural network
- Tensor factorization is successfully applied for various data structures with
  - multiple ways such as trials, conditions, subjects, channels, spaces, times and frequencies could be represented simultaneously

Motivation

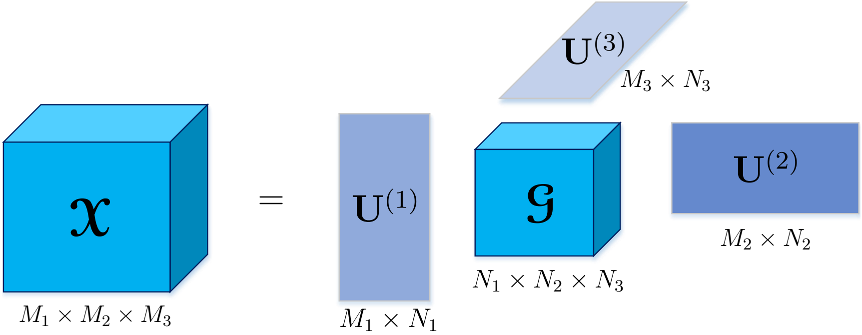
- Multi-way data are unfolded as one-way vectors for NN-based model.
  - neighboring, temporal and spatial information are missing
  - spend extra parameters and training samples



- Tensor analysis aims to keep the multi-way structure in inputs and features
- How to combine neural network and tensor factorization?

Outline

Tucker Decomposition for Three-way Tensor



$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$$

Tucker Decomposition

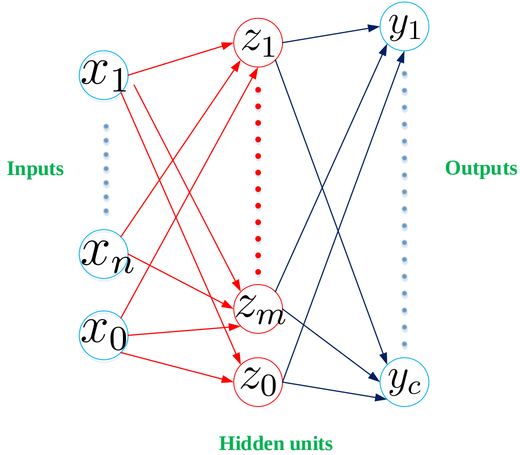
- P-way input tensor $\mathcal{X} \in \mathbb{R}^{M_1 \times \cdots \times M_P}$ is decomposed into core tensor $\mathcal{G}$ and matrices $\mathbf{U}^{(p)}$

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

$$\mathcal{X}_{m_1 m_2 \cdots m_P} = \sum_{n_1=1}^{N_1} \sum_{n_2=1}^{N_2} \cdots \sum_{n_P=1}^{N_P} \mathcal{G}_{n_1 n_2 \cdots n_P} u^{(1)}_{m_1 n_1} u^{(2)}_{m_2 n_2} \cdots u^{(P)}_{m_P n_P}$$

- Two methods to compute the Tucker decomposition (Lathauwer et al., 2000)
  - higher-order singular value decomposition
  - higher-order orthogonal iteration

Neural Network



$$
\begin{aligned}
z_j &= \sigma\Big(\sum_i w_{ji} x_i\Big) \\
&= \sigma(\mathbf{w}_j^T \mathbf{x}) \\
&= \sigma(\langle \mathbf{w}_j, \mathbf{x} \rangle)
\end{aligned}
$$

$$
y_c = \frac{\exp(\sum_j w_{cj} z_j)}{\sum_k \exp(\sum_j w_{kj} z_j)}
$$

Outline

Tensor Factorized Neural Network

- A generalization of conventional neural network (NN) classifier in presence of multi-way data
- Tensor factorized neural network (TFNN) keeps the original multi-way data structure and extracts the features with multiple modes
- Construct meaningful feature representation and classification system
- Key difference between NN classifier and TFNN is the style they handle the high dimensional data in multiple ways

Tensor Factorization & Transformation

- Tucker decomposition:

$$\mathcal{X} = \mathcal{A} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

$$\mathcal{A} = \mathcal{X} \times_1 \mathbf{U}^{(1)^\dagger} \times_2 \mathbf{U}^{(2)^\dagger} \times_3 \cdots \times_P \mathbf{U}^{(P)^\dagger}$$

- Tensor transformation:

$$\mathcal{A} = \mathcal{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

Tensor Feedforward Computation

Given a P-way tensor $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_P}$ as input

1. Tensor transformation layer:

$$\mathcal{A}^{\{1\}} = \mathcal{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

2. Nonlinear activation layer:

$$\mathcal{Z}^{\{2\}} = h(\mathcal{A}^{\{1\}})$$

3. Softmax layer:

$$\mathbf{y} = s(\mathbf{a}) = \frac{\exp(\mathbf{a})}{\sum_c \exp(a_c)}.$$

$$a_c = \langle \mathcal{W}_{:,:,\cdots,:,c}, \mathcal{Z}^{\{l-1\}} \rangle$$

Tensor Feedforward Computation

Given a P-way tensor $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_P}$ as input

1. Tensor transformation layer:

$$\mathcal{A}^{\{1\}} = \mathcal{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

2. Nonlinear activation layer:

$$\mathcal{Z}^{\{2\}} = h(\mathcal{A}^{\{1\}})$$

3. Softmax layer:

$$\mathbf{y} = s(\mathbf{a}) = \frac{\exp(\mathbf{a})}{\sum_c \exp(a_c)}.$$

$$a_c = \langle \mathcal{W}_{:,:,\cdots,:,c}, \mathcal{Z}^{\{l-1\}} \rangle$$

Tensor Feedforward Computation

Given a P-way tensor $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_P}$ as input

1. Tensor transformation layer:

$$\mathcal{A}^{\{1\}} = \mathcal{X} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_P \mathbf{U}^{(P)}$$

2. Nonlinear activation layer:

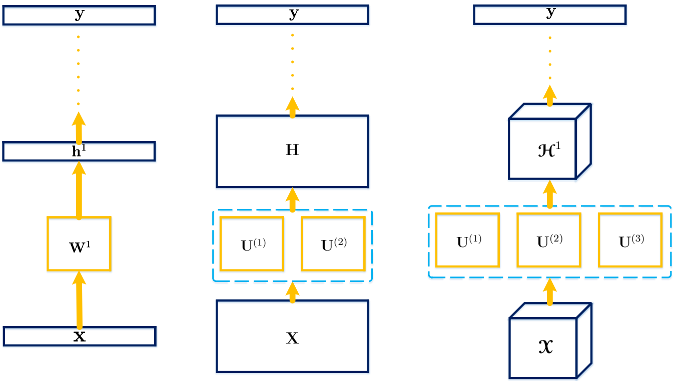
$$\mathcal{Z}^{\{2\}} = h(\mathcal{A}^{\{1\}})$$

3. Softmax layer:

$$\mathbf{y} = s(\mathbf{a}) = \frac{\exp(\mathbf{a})}{\sum_c \exp(a_c)}.$$

$$a_c = \langle \mathcal{W}_{:,:,\cdots,:,c}, \mathcal{Z}^{\{l-1\}} \rangle$$

TFNN in Different Ways



(a) one-way (b) two-way (c) three-way

Tensor Factorized Error Backpropagation

- Estimate the model parameters $\boldsymbol{\Theta} = \{\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(P)}, \boldsymbol{\mathcal{W}}\}$ by minimizing the cross-entropy error function

$$E(\boldsymbol{\Theta}) = \sum_t E_t(\boldsymbol{\Theta}) = -\sum_t \sum_c r_{tc} \ln y_{tc}(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta})$$

- By using the stochastic gradient descent algorithm, we update the parameters iteratively

$$\boldsymbol{\Theta}^{(\tau+1)} = \boldsymbol{\Theta}^{(\tau)} - \eta \frac{\partial E(\boldsymbol{\Theta})}{\partial \boldsymbol{\Theta}}$$

Differentiation of Softmax Layer

- Softmax layer:

$$\frac{\partial E_t}{\partial a_c^{(l)}} = \sum_k \frac{\partial E_t}{\partial y_{tk}} \frac{\partial y_{tk}}{\partial a_c^{(l)}} = y_{tc} - r_{tc} \triangleq d_c^{(l)}$$

$$\frac{\partial E_t}{\partial \mathcal{W}_{n_1 n_2 \cdots n_P c}} = \frac{\partial E_t}{\partial a_c^{(l)}} \frac{\partial a_c^{(l)}}{\partial \mathcal{W}_{n_1 n_2 \cdots n_P c}} = d_c^{(l)} \mathcal{Z}_{n_1 n_2 \cdots n_P}^{(l-1)}$$

$$\nabla_{\boldsymbol{\mathcal{W}}_{:,:,\cdots,:c}} E_t = d_c^{(l)} \times \boldsymbol{\mathcal{Z}}^{(l-1)}$$

- Backpropagation of local gradients:

$$\frac{\partial E_t}{\partial \mathcal{Z}_{n_1 n_2 \cdots n_p}^{(l-1)}} = \sum_c \frac{\partial E_t}{\partial a_c^{(l)}} \frac{\partial a_c^{(l)}}{\partial \mathcal{Z}_{n_1 n_2 \cdots n_P}^{(l-1)}} = \sum_c d_c^{(l)} \mathcal{W}_{n_1 n_2 \cdots n_P c}$$

$$\boldsymbol{\mathcal{D}}^{(l-1)} = \boldsymbol{\mathcal{W}} \times_{(P+1)} \mathbf{d}^{(l)}$$

Differentiation of Nonlinear Activation Layer

- Nonlinear activation layer:

$$\frac{\partial E_t}{\partial \mathcal{A}^{(l-2)}_{n_1 n_2 \cdots n_P}} = \frac{\partial E_t}{\partial \mathcal{Z}^{(l-1)}_{n_1 n_2 \cdots n_P}} \frac{\partial \mathcal{Z}^{(l-1)}_{n_1 n_2 \cdots n_P}}{\partial \mathcal{A}^{l-2)}_{n_1 n_2 \cdots n_P}} = \mathcal{D}^{(l-1)}_{n_1 n_2 \cdots n_P} h'(\mathcal{A}^{(l-2)}_{n_1 n_2 \cdots n_P})$$

$$\boldsymbol{\mathcal{D}}^{(l-2)} = \boldsymbol{\mathcal{D}}^{(l-1)} * h'(\boldsymbol{\mathcal{A}}^{(l-2)})$$

Differentiation of Tensor Transformation Layer

- Tensor transformation layer :

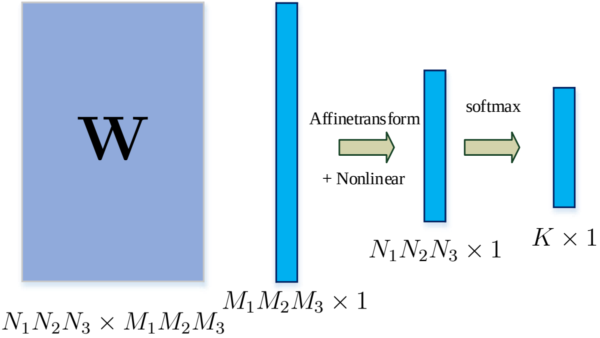
$$\frac{\partial E_t}{\partial U^{(p)}_{n_p m_p}} = \sum_{n_1} \cdots \sum_{n_{p-1}} \sum_{n_{p+1}} \cdots \sum_{n_P} \frac{\partial E_t}{\partial \mathcal{A}^{(l-2)}_{n_1 n_2 \cdots n_P}} \frac{\partial \mathcal{A}^{(l-2)}_{n_1 n_2 \cdots n_P}}{\partial U^{(p)}_{n_p m_p}}$$
$$= \langle \mathfrak{D}^{(l-2)}_{::\cdots n_p \cdots :}, \mathfrak{T}_{::\cdots m_p \cdots :} \rangle$$

$$\mathfrak{T}_{::\cdots m_p \cdots :} = \mathfrak{Z}^{(l-3)}_{::\cdots m_p \cdots :} \times_1 \mathbf{U}^{(1)} \cdots \times_{p-1} \mathbf{U}^{(p-1)} \times_{p+1} \mathbf{U}^{(p+1)} \cdots \times_P \mathbf{U}^{(P)}$$

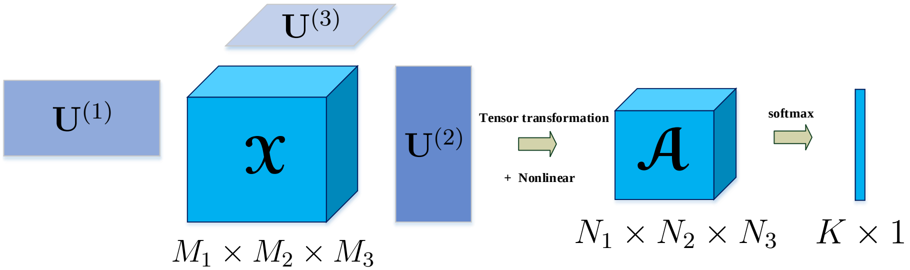
- Backpropagation of local gradients:

$$\mathfrak{D}^{(l-3)} = \mathfrak{D}^{(l-2)} \times_1 \mathbf{U}^{(1)T} \times_2 \cdots \times_P \mathbf{U}^{(P)T}$$

Number of Parameters in NN



Number of Parameters in TFNN



Model	Neural network	Tensor factorized neural network
Parameter size	$\prod_p (M_p N_p) + K \prod_p N_p$	$\sum_p (M_p N_p) + K \prod_p N_p$

- TFNN needs very few parameters

Outline

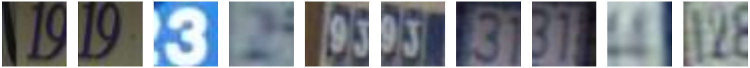
Two-Way TFNN Task

- MNIST dataset:
  - 0-9 digits
  - grayscale images with size 28×28
  - 60,000 training images and 10,000 test images

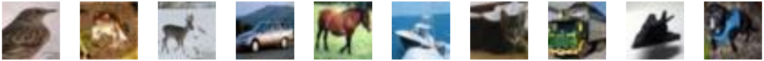
- Preprocessing: normalization into values between 0 and 1
- mini-batch: 50
- $1/6$ training data for held-out validation
- learning rate: 0.001 and 0.005 for $\mathbf{U}^{(1)}, \mathbf{U}^{(2)}$ and $\boldsymbol{\mathcal{W}}$

Three-Way TFNN Task

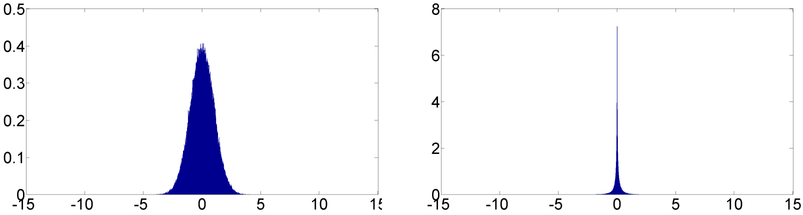
- SVHN dataset:
  - 0-9 digits, but not crop well
  - colour images with size 32 × 32
  - 73,256 training images and 26,032 test images



- CIFAR-10 dataset:
  - contain 10 classes such as airplane, bird, cat, etc.
  - colour images with size 32 × 32
  - 50,000 training images and 10,000 test images



Distribution of Weights



(a) NN (b) TFNN

Experimental Result of MNIST

Method	Topology	Parameter Size	Accuracy
NN	784-70-10	55,580	95.6%
NN	784-196-10	155,624	97.3%
NN	784-1000-10	794,000	97.7%
TFNN	28×28-14×14-10	2,744	96.2%
TFNN	28×28-40×40-10	27,800	96.8%
TFNN	28×28-70×70-10	52,920	97.7%

Experimental Result of SVHN

Method	Topology	Parameter Size	Accuracy
NN	3072-400-10	1,232,800	55%
NN	3072-1000-10	3,082,000	63%
TFNN	$32\times32\times3$-$20\times20\times3$-10	13,289	72%

Experimental Result of CIFAR-10

Method	Topology	Parameter Size	Accuracy
NN	3072-1000-10	3,082,000	33%
TFNN	$32 \times 32 \times 3$-$30 \times 30 \times 2$-10	19,926	43%
TFNN	$32 \times 32 \times 3$-$30 \times 30 \times 6$-10	55,938	46%

Part II: Domain Adaptive Neural Network

Spatial Temporal Modeling Workshop (STM 2016)

Outline

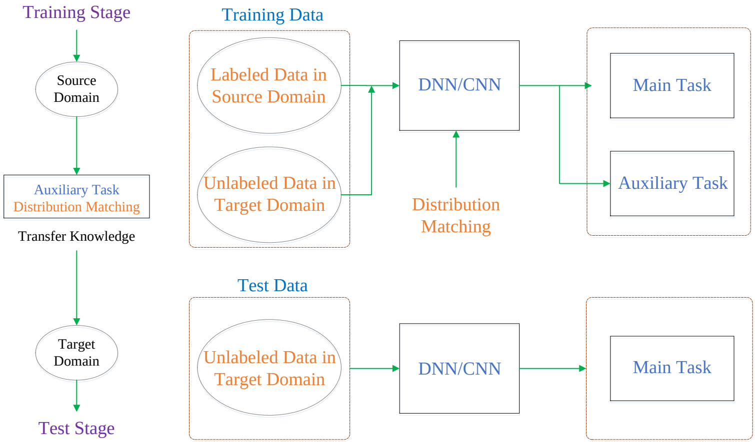
Introduction

- Traditional machine learning works well under an assumption that training and test data follow the same distribution
  - real-world data may not follow this assumption
- Feature-based domain adaptation is a common approach
  - allow knowledge to be transferred across domains through learning a good feature representation

Motivation

- Most of previous studies are restricted to train features and classifier separately under a shallow model structure
- We co-train the feature representation and classifier under neural network without labeling in target domain
- Objective function is based on multi-task learning and distribution matching

Systematic Diagram



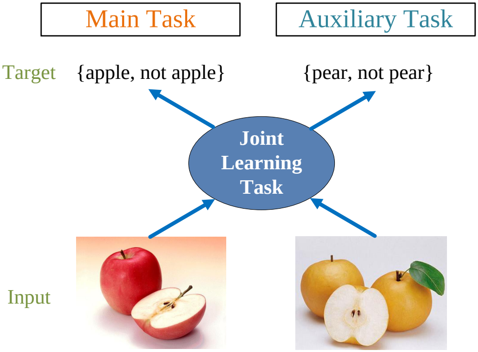
Outline

Transfer Learning

- Let $\mathcal{D} = \{\mathcal{X}, p(X)\}$ denote a domain
  - feature space $\mathcal{X}$
  - marginal probability distribution $p(X)$
  - $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset \mathcal{X}$
- Let $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ denote a task
  - label space $\mathcal{Y}$
  - objective predictive function $f(\cdot)$
    can be written as $p(Y|X)$
- Assumptions in transfer learning
  - source and target domains are different $\mathcal{D}_S \neq \mathcal{D}_T$
  - source and target tasks are different $\mathcal{T}_S \neq \mathcal{T}_T$

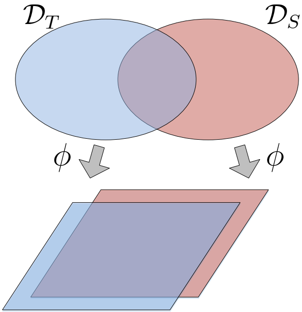
Multi-task Learning

$$\min_{\theta} \ell(\mathcal{D}_m, \theta) + \lambda\Omega(\theta)$$



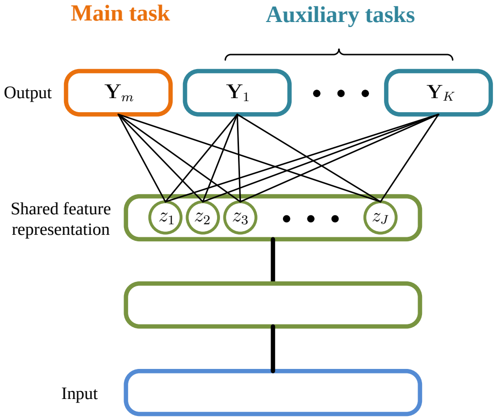
Feature-based Domain Adaptation

- Assume that a domain-invariant feature space exists
- Minimize $Div(p(\phi(X^s)), p(\phi(X^t)))$ to find transformation ϕ



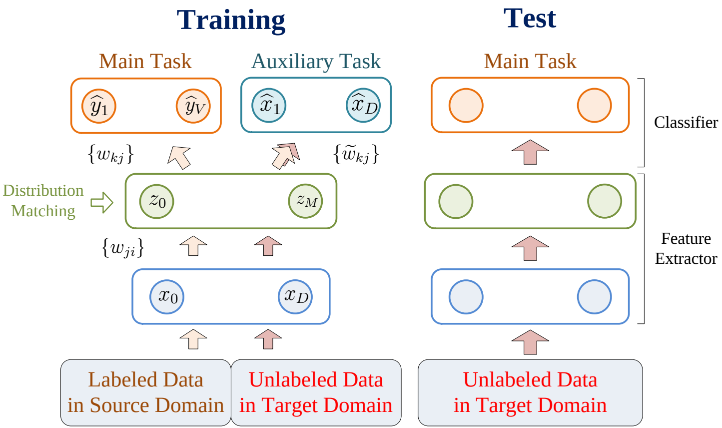
Multi-task Neural Network Learning

$$\min_{\theta} \ell(\mathcal{D}_m, \theta) + \lambda\Omega(\theta)$$



Outline

Learning Strategy and Task



Objective Function

- Semi-supervised model adaptation aims to estimate the neural network parameters **w** by minimizing

$$E(\mathbf{w}) = E_c(\mathbf{w}) + \lambda_r E_r(\mathbf{w}) + \lambda_d E_d(\mathbf{w})$$

where λ_r and λ_d are the empirical regularization parameters

- $E_c(\mathbf{w})$ is classification error for main task
- $E_r(\mathbf{w})$ is reconstruction error for auxiliary task
- $E_d(\mathbf{w})$ is error for matching distribution in hidden layer

Multi-task Learning

- Classification task
  - cross entropy error function
  - training samples and their labels from source domain

$$E_c(\mathbf{w}) = -\sum_{a=1}^{m}\sum_{v} t_{av} \log y_{av}$$

- Regression task
  - squared reconstruction error
  - both datasets in source and target domains

$$E_r(\mathbf{w}) = \frac{1}{m+n}\sum_{a=1}^{m+n} \|\widehat{\mathbf{x}}_a - \mathbf{x}_a\|^2$$

Distribution Matching

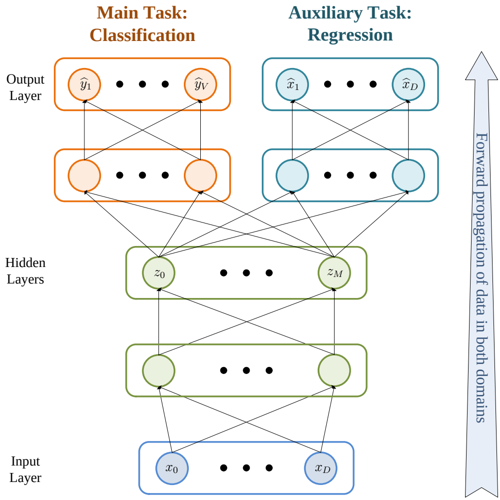
- Maximum mean discrepancy (MMD) is defined with $f(x) = \langle \phi(x), f \rangle$ and $\phi(x)\colon \mathcal{X} \to \mathcal{H}$

$$\mathrm{MMD}(X^s, X^t) = \left\| \frac{1}{m} \sum_{a=1}^{m} \phi(x_a^s) - \frac{1}{n} \sum_{a=1}^{n} \phi(x_a^t) \right\|_{\mathcal{H}}$$

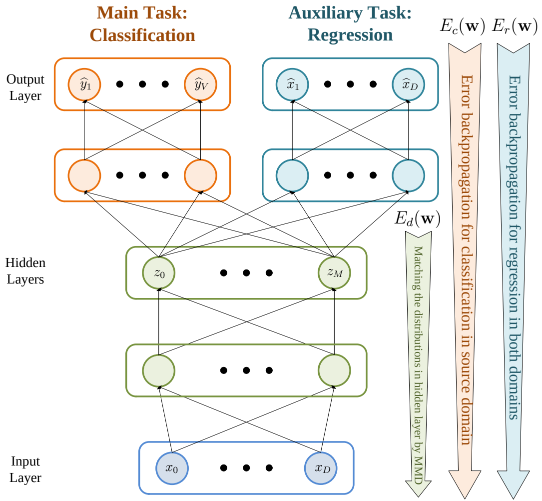
$$= \left[\frac{1}{m^2} \sum_{a,b=1}^{m} k(x_a^s, x_b^s) - \frac{2}{mn} \sum_{a,b=1}^{m,n} k(x_a^s, x_b^t) + \frac{1}{n^2} \sum_{a,b=1}^{n} k(x_a^t, x_b^t) \right]^{\frac{1}{2}}$$

- Gaussian kernel $k(\cdot, \cdot)$ is used

Learning Procedure



Learning Procedure



Derivation of Differentiations

- Differentiation for second term $E_{d2}(\mathbf{w})$ is shown below

$$\frac{\partial E_{d2}}{\partial w_{ji}} = \sum_{a,b=1}^{m,n} \sum_{j} \frac{\partial E_{d2}}{\partial z_j} \frac{\partial z_j}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}$$

where z_j comes from both domain data z^s_{aj} and z^t_{bj} and

$$\begin{aligned}\frac{\partial E_{d2}}{\partial z_j} &= -\frac{2}{\sigma^2} \exp(-\|\mathbf{z}^s_a - \mathbf{z}^t_b\|^2 / 2\sigma^2)(z^s_{aj} - z^t_{bj}) \\ &= g(\mathbf{z}^s_a, \mathbf{z}^t_b)(z^s_{aj} - z^t_{bj})\end{aligned}$$

Derivation of Differentiations

- We can find differentiations for three terms as

$$\frac{\partial E_{d2}}{\partial w_{ji}} = \sum_{a,b=1}^{m,n} \sum_{j} g(\mathbf{z}_a^s, \mathbf{z}_b^t) \left(z_{aj}^s \frac{\partial z_{aj}^s}{\partial a_{aj}^s} \frac{\partial a_{aj}^s}{\partial w_{ji}} - z_{bj}^t \frac{\partial z_{bj}^t}{\partial a_{bj}^t} \frac{\partial a_{bj}^t}{\partial w_{ji}} \right)$$

$$\frac{\partial E_{d1}}{\partial w_{ji}} = 2 \sum_{a=1}^{m} \sum_{j} g(\mathbf{z}_a^s, \mathbf{z}_a^s) \left(z_{aj}^s \frac{\partial z_{aj}^s}{\partial a_{aj}^s} \frac{\partial a_{aj}^s}{\partial w_{ji}} \right)$$

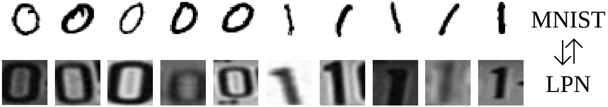
$$\frac{\partial E_{d3}}{\partial w_{ji}} = 2 \sum_{a=1}^{n} \sum_{j} g(\mathbf{z}_a^t, \mathbf{z}_a^t) \left(z_{aj}^t \frac{\partial z_{aj}^t}{\partial a_{aj}^t} \frac{\partial a_{aj}^t}{\partial w_{ji}} \right)$$

- $\frac{\partial E_{d2}}{\partial w_{ji}}$ involves both domain data, $\frac{\partial E_{d1}}{\partial w_{ji}}$ and $\frac{\partial E_{d3}}{\partial w_{ji}}$ only involve source domain data and target domain data, respectively

Outline

Experimental Data

- MNIST dataset:
  - MNIST uses a subset of original MNIST dataset
  - 6000 images per class (0 and 1)
  - size of images is 28×28
- LPN dataset:
  - license plate numbers with different angles and illuminations captured from different surveillance cameras
  - 4000 images per class (0 and 1)
  - size of images is 28×28



Experimental Setup

- Subtract mean over the dataset and normalize over the dataset to a stand normal distribution in each pixel
- Four-layer classification network (784-500-300-2)
- Three-layer auto-encoder regression network (784-500-784)
  - activation function: sigmoid function
  - output function: softmax function
- Stochastic gradient descent (SGD) with momentum
  - $\lambda_r = 1$, $\lambda_d = 1.2$ (MNIST→LPN), $\lambda_d = 2.2$ (LPN→MNIST)
  - batchsize: 2000 and 200
  - epoch: 350
  - momentum: 0.5
  - learning rate: 1 (learning rate is multiplied by a factor of 0.7 each 10 epochs after 300th epoch)

Experimental Result

- Classification error rates (%)

	MNIST→LPN	LPN→MNIST
NN	27.3	15.0
NN+SSL (distribution matching)	18.8	13.5
NN+SSL (multi-task learning)	25.3	5.4
NN+SSL (both)	*16.2*	*3.3*

Experimental Data

- Amazon dataset
  - Amazon product reviews on four domains or product types (kitchen appliances, DVDs, books, electronics)
  - 1000 positive and 1000 negative reviews on each product type
- Pre-processing
  - ignore the words appearing less than 10 occurrences (dictionary size 40K words)
  - use tf-idf reweighting method to extract feature vectors
  - transform feature vector into low-dimensional vector with 2300 dimensions by PCA

Experimental Setup

- Four-layer classification network (2300-300-50-2)
- Three-layer auto-encoder regression network (2300-300-2300)
  - activation function: sigmoid function
  - output function: softmax function
- Stochastic gradient descent with momentum
  - $\lambda_r = 1$ and $\lambda_d = 0.8$
  - batchsize: 1000 and 1000
  - epoch: 500
  - momentum: 0.5
  - learning rate: 1 (learning rate is multiplied by a factor of 0.5 each 30 epochs after 300th epoch)

Experimental Result

- Classification error rates (%) for adaptation among different domains (K: Kitchen appliances, D: DVDs, B: Books, E: Electronics)

	K→D	D→B	B→E	E→K
NN	31.8	23.3	24.3	36.5
CODA	26.0	21.4	18.6	*27.2*
NN+SSL	*22.7*	*20.6*	*13.9*	27.4

Part III: Bayesian Unfolding Network

Spatial Temporal Modeling Workshop (STM 2016)

Outline

Outline

Topic Models

- Topic models are tools for discovering the abstract topics that occur in collection of documents. For example,

  a document consists in

  - **90%** of tokens $\in$ { **medicine**, doctor, patients, ... }
  - **10%** of tokens $\in$ { **baseball**, runner, bat, ball, ... }

Input:

Statistical approaches help in the determination of significant configurations in protein and nucleic acid sequence data. Three recent statistical methods are discussed: (i) score-based sequence analysis that provides a means for characterizing anomalies in local sequence text and for evaluating sequence comparisons; (ii) quantile distributions of amino acid usage that reveal general compositional biases in proteins and evolutionary relations; and (iii) r-scan statistics that can be applied to the analysis of spacing of sequence markers.

Input:

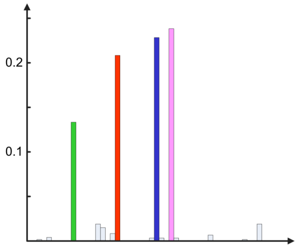
Statistical approaches help in the determination of significant configurations in protein and nucleic acid sequence data. Three recent statistical methods are discussed: (i) score-based sequence analysis that provides a means for characterizing anomalies in local sequence text and for evaluating sequence comparisons; (ii) quantile distributions of amino acid usage that reveal general compositional biases in proteins and evolutionary relations; and (iii) r-scan statistics that can be applied to the analysis of spacing of sequence markers.

Input:

Statistical approaches help in the determination of significant configurations in protein and nucleic acid sequence data. Three recent statistical methods are discussed: (i) score-based sequence analysis that provides a means for characterizing anomalies in local sequence text and for evaluating sequence comparisons; (ii) quantile distributions of amino acid usage that reveal general compositional biases in proteins and evolutionary relations; and (iii) r-scan statistics that can be applied to the analysis of spacing of sequence markers.

Output:

sequence	measured	residues	computer
region	average	binding	methods
pcr	range	domains	number
identified	values	helix	two
fragments	different	cys	principle
two	size	regions	design
genes	three	structure	access
three	calculated	terminus	processing
cdna	two	terminal	advantage
analysis	low	site	important

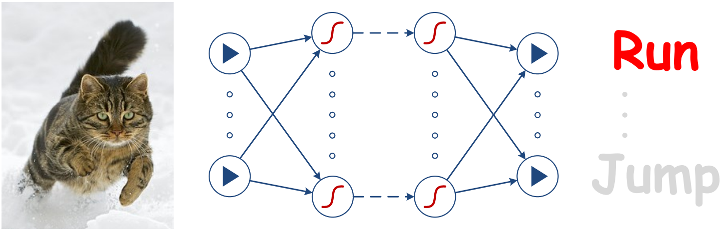


Topic words

Topic proportions

Neural Network Learning

- Deep structured/hierarchical learning
- Rapidly developed and widely applied for many applications
- Multiple layers of nonlinear processing units
- High-level abstraction



Motivation

Topic Model + **Neural Network**

Outline

1 Introduction
- Topic model
- Neural network
- Motivation

2 Bayesian Unfolding Inference
- Bayesian unfolding
- Unfolding for unsupervised topic model
- Unfolding for supervised topic model

3 Conclusions and Future Works

Generative Models vs. Neural Nets

	Generative Models	Neural Nets
Structure	Top-down	Bottom-up
Representation	Intuitive	Distributed
Interpretation	**Easy**	**Harder**
Semi/unsupervised	**Easier**	**Harder**
Incorp. domain knowl.	**Easy**	**Hard**
Incorp. constraint	**Easy**	**Hard**
Incorp. uncertainty	**Easy**	**Hard**
Learning	Many algorithms	Back-propagation
Inference/decode	**Harder**	**Easier**
Evaluation on	int. quantity	**End performance**

Generative Models vs. Neural Nets

	Generative Models	Neural Nets
Structure	Top-down	Bottom-up
Representation	Intuitive	Distributed
Interpretation	**Easy**	**Harder**
Semi/unsupervised	**Easier**	**Harder**
Incorp. domain knowl.	**Easy**	**Hard**
Incorp. constraint	**Easy**	**Hard**
Incorp. uncertainty	**Easy**	**Hard**
Learning	Many algorithms	Back-propagation
Inference/decode	**Harder**	**Easier**
Evaluation on	int. quantity	**End performance**

Generative Models vs. Neural Nets

	Generative Models	Neural Nets
Structure	Top-down	Bottom-up
Representation	Intuitive	Distributed
Interpretation	**Easy**	**Harder**
Semi/unsupervised	**Easier**	**Harder**
Incorp. domain knowl.	**Easy**	**Hard**
Incorp. constraint	**Easy**	**Hard**
Incorp. uncertainty	**Easy**	**Hard**
Learning	Many algorithms	Back-propagation
Inference/decode	**Harder**	**Easier**
Evaluation on	int. quantity	**End performance**

Goal of Bayesian Unfolding Framework

	Generative Models	Neural Nets
Structure	Top-down	Bottom-up
Representation	Intuitive	Distributed
Interpretation	**Easy**	Harder
Semi/unsupervised	**Easier**	Harder
Incorp. domain knowl.	**Easy**	Hard
Incorp. constraint	**Easy**	Hard
Incorp. uncertainty	**Easy**	Hard
Learning	Many algorithms	Back-propagation
Inference/decode	Harder	**Easier**
Evaluation on	int. quantity	**End performance**

Bayesian Unfolding Framework

Model-based method

$$\max_{\boldsymbol{\Theta}} \ \mathcal{J}_{\boldsymbol{\Theta}}(\{y_n\})$$

Bayesian Unfolding Framework

Model-based method

$$\max_{\mathbf{\Theta}} \ \mathcal{J}_{\mathbf{\Theta}}(\{y_n\})$$

where

$$\max_{\mathbf{\Psi}} \ \mathcal{F}_{\mathbf{\Theta}}(\{x_n\}, \mathbf{\Psi}_{\text{best}})$$

estimate y_n given $\mathbf{\Psi}_{\text{best}}$

Bayesian Unfolding Framework

Model-based method

$$\max_{\boldsymbol{\Theta}} \ \mathcal{J}_{\boldsymbol{\Theta}}(\{y_n\})$$

repeat

$\boldsymbol{\Psi}_n = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

until convergence

$y_n = \text{estimate}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

$\boldsymbol{\Theta} = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

Bayesian Unfolding Framework

Model-based method

$$\max_{\boldsymbol{\Theta}} \; \mathcal{J}_{\boldsymbol{\Theta}}(\{y_n\})$$

repeat

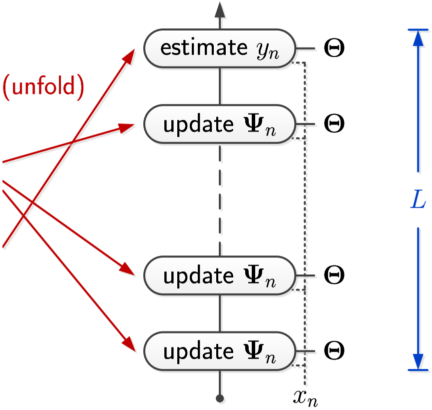
$\boldsymbol{\Psi}_n = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

until convergence

$y_n = \text{estimate}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

$\boldsymbol{\Theta} = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

Bayesian unfolding framework



Bayesian Unfolding Framework

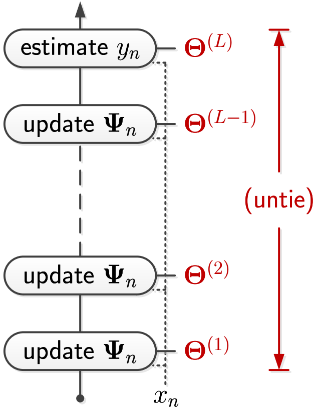
Model-based method

$$\max_{\boldsymbol{\Theta}} \mathcal{J}_{\boldsymbol{\Theta}}(\{y_n\})$$

repeat
$\boldsymbol{\Psi}_n = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$
until convergence
$y_n = \text{estimate}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

$\boldsymbol{\Theta} = \text{update}(x_n, \boldsymbol{\Psi}_n, \boldsymbol{\Theta})$

Bayesian unfolding framework



Network Training: Feed-forward

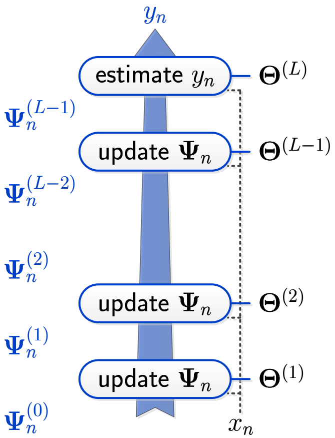
Model-based method

$$\max_{\mathbf{\Theta}} \ \mathcal{J}_{\mathbf{\Theta}}(\{y_n\})$$

repeat
$\mathbf{\Psi}_n = \text{update}(x_n, \mathbf{\Psi}_n, \mathbf{\Theta})$
until convergence
$y_n = \text{estimate}(x_n, \mathbf{\Psi}_n, \mathbf{\Theta})$

$\mathbf{\Theta} = \text{update}(x_n, \mathbf{\Psi}_n, \mathbf{\Theta})$

Bayesian unfolding network



Network Training: Back-propagation

Back-propagation

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(L)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Theta}^{(L)}}$$

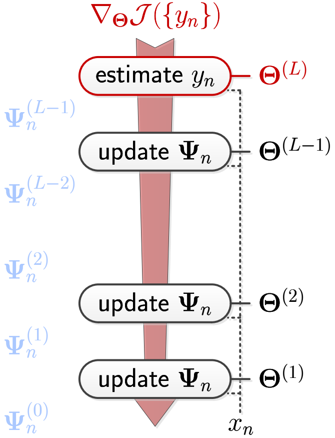
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(L)}} = \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Psi}_n^{(L)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(l)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Theta}^{(l)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l)}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Psi}_n^{(l)}}$$

for $l = L-1, \dots, 1$

Bayesian unfolding network



Network Training: Back-propagation

Back-propagation

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(L)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Theta}^{(L)}}$$

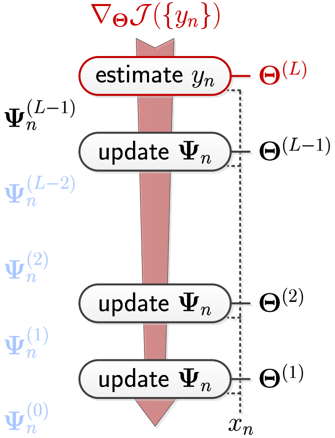
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(L)}} = \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Psi}_n^{(L)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(l)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Theta}^{(l)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l)}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Psi}_n^{(l)}}$$

for $l = L-1, \ldots, 1$

Bayesian unfolding network



Network Training: Back-propagation

Back-propagation

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(L)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Theta}^{(L)}}$$

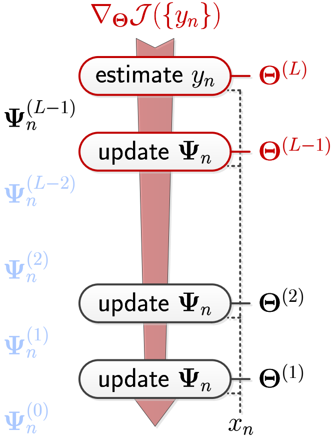
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(L)}} = \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Psi}_n^{(L)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(l)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Theta}^{(l)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l)}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Psi}_n^{(l)}}$$

for $l = L-1, \ldots, 1$

Bayesian unfolding network



Network Training: Back-propagation

Back-propagation

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(L)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Theta}^{(L)}}$$

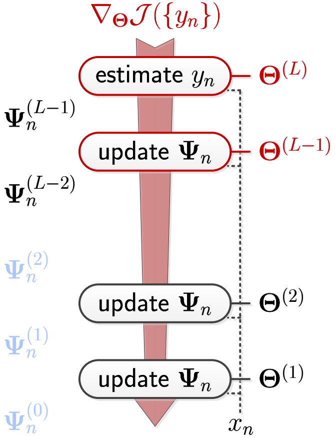
$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(L)}} = \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \boldsymbol{\Psi}_n^{(L)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Theta}^{(l)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Theta}^{(l)}}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l)}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\Psi}_n^{(l+1)}} \frac{\partial \boldsymbol{\Psi}_n^{(l+1)}}{\partial \boldsymbol{\Psi}_n^{(l)}}$$

for $l = L-1, \ldots, 1$

Bayesian unfolding network



Network Training: Back-propagation

Back-propagation

$$\frac{\partial \mathcal{J}}{\partial \mathbf{\Theta}^{(L)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \mathbf{\Theta}^{(L)}}$$

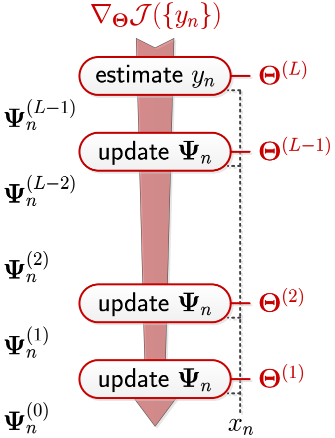
$$\frac{\partial \mathcal{J}}{\partial \mathbf{\Psi}_n^{(L)}} = \frac{\partial \mathcal{J}}{\partial \widehat{y}_n} \frac{\partial \widehat{y}_n}{\partial \mathbf{\Psi}_n^{(L)}}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{\Theta}^{(l)}} = \sum_n \frac{\partial \mathcal{J}}{\partial \mathbf{\Psi}_n^{(l+1)}} \frac{\partial \mathbf{\Psi}_n^{(l+1)}}{\partial \mathbf{\Theta}^{(l)}}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{\Psi}_n^{(l)}} = \frac{\partial \mathcal{J}}{\partial \mathbf{\Psi}_n^{(l+1)}} \frac{\partial \mathbf{\Psi}_n^{(l+1)}}{\partial \mathbf{\Psi}_n^{(l)}}$$

for $l = L-1, \ldots, 1$

Bayesian unfolding network

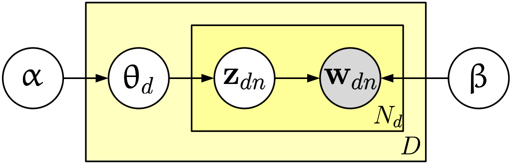


Bayesian Unfolding Inference for Topic Models

- Topic model $\rightarrow$ Bayesian unfolding inference
- Model parameters are inferred by maximizing the end performance of network
  - Unsupervised topic model $\rightarrow$ maximize empirical likelihood
  - Supervised topic model $\rightarrow$ maximize cross-entropy

Bayesian Unfolding Inference for Unsupervised Topic Model - Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation



- True posterior distribution

$$p(\mathbf{z}, \theta \mid \mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\mathbf{w}, \mathbf{z}, \theta \mid \boldsymbol{\alpha}, \boldsymbol{\beta})}{p(\mathbf{w} \mid \boldsymbol{\alpha}, \boldsymbol{\beta})}$$

Variational Bayesian Learning for LDA

- Evidence lower bound (ELBO)

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\gamma}; \boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&= \mathbb{E}_q[\ln p(\mathbf{w}, \mathbf{z}, \boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta})] + \mathbb{H}[\mathbf{z}, \boldsymbol{\theta}] \\
&= \underbrace{\mathbb{E}_q[\ln p(\mathbf{w}|\mathbf{z}, \boldsymbol{\beta})] + \mathbb{E}_q[\ln p(\mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_q[\ln q(\mathbf{z}|\boldsymbol{\phi})]}_{\text{word probability constructed by topic model}} \\
&+ \underbrace{\mathbb{E}_q[\ln p(\boldsymbol{\theta}|\boldsymbol{\alpha})] - \mathbb{E}_q[\ln q(\boldsymbol{\theta}|\boldsymbol{\gamma})]}_{\text{Dirichlet prior}}
\end{aligned}$$

- Variational posterior distribution

$$q(\mathbf{z}, \boldsymbol{\theta} \mid \boldsymbol{\phi}, \boldsymbol{\gamma}) = \prod_d q(\boldsymbol{\theta}_d \mid \boldsymbol{\gamma}_d) \prod_n q(\mathbf{z}_{dn} \mid \boldsymbol{\phi}_{dn})$$

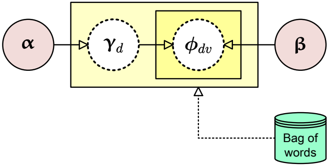
Variational Bayesian Learning for LDA

```
Initialize α, β
repeat
    for all document d do
        φ_dvk = 1/K
        repeat
            γ_dk = α_k + Σ_v N_dv φ_dvk
            φ_dvk ∝ exp{ln β_kv + ψ(γ_dk)}
        until γ_dk and φ_dvk converged
    end for
    β_kv ∝ Σ_d N_dv φ_dvk
    α ← Newton-Raphson(α, g(α), H(α))
until ELBO converged
```

Bayesian Unfolding Inference for LDA

$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

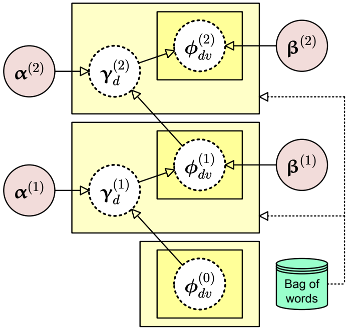
$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$



$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Bayesian Unfolding Inference for LDA



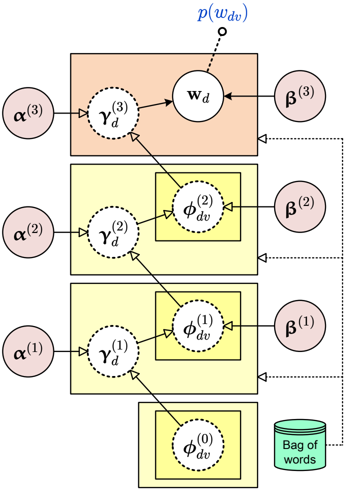
$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Bayesian Unfolding Inference for LDA



$$p(w_{dv}) = \sum_k \beta_{vk} \mathbb{E}_q[\theta_{dk}]$$

$$\mathbb{E}_q[\theta_{dk}] = \frac{\gamma_{dk}}{\sum_j \gamma_{dj}}$$

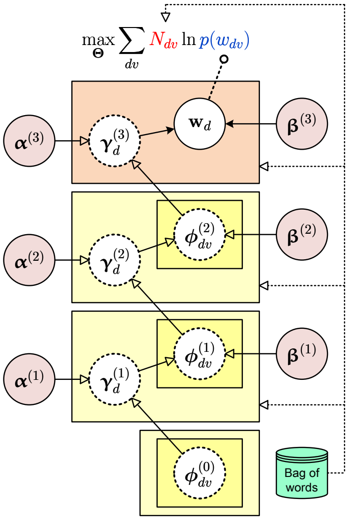
$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Bayesian Unfolding Inference for LDA



$$p(w_{dv}) = \sum_k \beta_{vk} \mathbb{E}_q[\theta_{dk}]$$

$$\mathbb{E}_q[\theta_{dk}] = \frac{\gamma_{dk}}{\sum_j \gamma_{dj}}$$

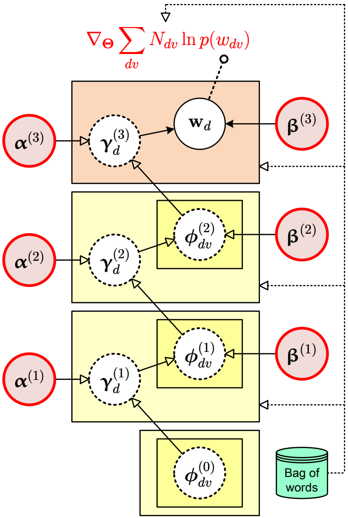
$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Bayesian Unfolding Inference for LDA



$$p(w_{dv}) = \sum_k \beta_{vk} \mathbb{E}_q[\theta_{dk}]$$

$$\mathbb{E}_q[\theta_{dk}] = \frac{\gamma_{dk}}{\sum_j \gamma_{dj}}$$

$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Constraint Optimization for Topic Models

- Exponentiated gradient method (EG)
  (probability simplex constraint)

$$\max_{\boldsymbol{\Theta}^{(\tau+1)}} \quad \mathcal{J}(\boldsymbol{\Theta}^{(\tau+1)}) - \frac{1}{\rho}\,\mathsf{KL}(\boldsymbol{\Theta}^{(\tau+1)} \| \boldsymbol{\Theta}^{(\tau)})$$

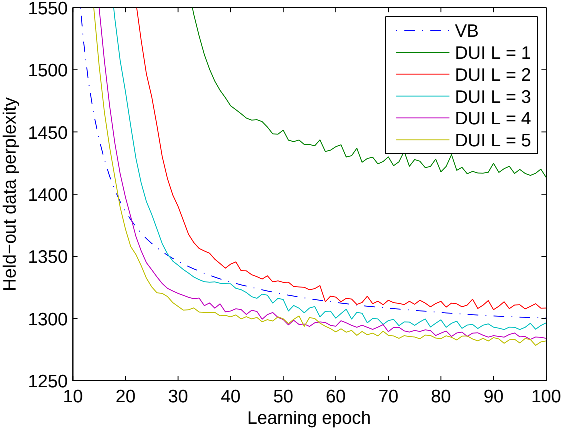
$$\text{s.t.} \quad \Theta_i^{(\tau+1)} \geq 0 \text{ and } \sum_i \Theta_i^{(\tau+1)} = 1$$

$$\Theta_i^{(t+1)} \propto \Theta_i^{(t)} \exp\left(-\rho \frac{\partial \mathcal{J}(\boldsymbol{\Theta}^{(t+1)})}{\partial \Theta_i^{(t)}}\right)$$

Experimental Data

- 20 newsgroups data set
  - rare, common an stop words are removed
  - random select 15,000 documents for document modelling
  - keep 5,000 frequent words
  - 9,000 documents for training
  - 6,000 documents for testing
- BUI for LDA
  - $\alpha = 1$
  - $K = 40$
  - model parameters are tied for all layers
  - minibatch $= 3,000$

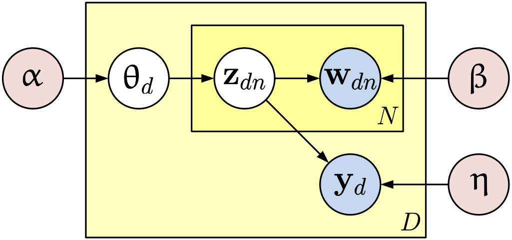
Evaluation for Perplexity



Bayesian Unfolding Inference for Supervised Topic Model - Supervised Latent Dirichlet Allocation (sLDA)

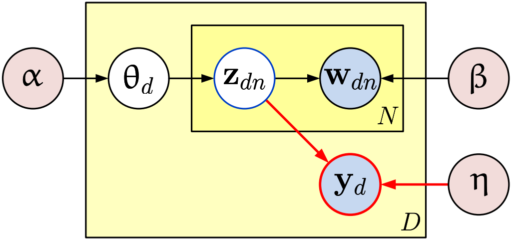
Supervised Topic Models for Multi-class Classification

- sLDA



Supervised Topic Models for Multi-class Classification

- sLDA



$$\bar{\mathbf{z}}_d = \frac{1}{N_d} \sum_n \mathbf{z}_{dn} \qquad \mathbf{y}_d \sim \frac{\exp(\boldsymbol{\eta}_m^\top \bar{\mathbf{z}}_d)}{\sum_m \exp(\boldsymbol{\eta}_m^\top \bar{\mathbf{z}}_d)}$$

Variational Bayesian Learning for sLDA

- Evidence lower bound (ELBO)

$$\begin{aligned}
&\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\gamma}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}) \\
&= \mathbb{E}_q[\ln p(\mathbf{w}, \mathbf{y}, \mathbf{z}, \boldsymbol{\theta} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta})] + \mathbb{H}[\mathbf{z}, \boldsymbol{\theta}] \\
&= \underbrace{\mathbb{E}_q[\ln p(\mathbf{w}|\mathbf{z}, \boldsymbol{\beta})] + \mathbb{E}_q[\ln p(\mathbf{z}|\boldsymbol{\theta})] - \mathbb{E}_q[\ln q(\mathbf{z}|\boldsymbol{\phi})]}_{\text{word probability constructed by topic model}} \\
&+ \underbrace{\mathbb{E}_q[\ln p(\boldsymbol{\theta}|\boldsymbol{\alpha})] - \mathbb{E}_q[\ln q(\boldsymbol{\theta}|\boldsymbol{\gamma})]}_{\text{Dirichlet prior}} + \underbrace{\mathbb{E}_q[\ln p(\mathbf{y}|\mathbf{z}, \boldsymbol{\eta})]}_{\text{supervision}}
\end{aligned}$$

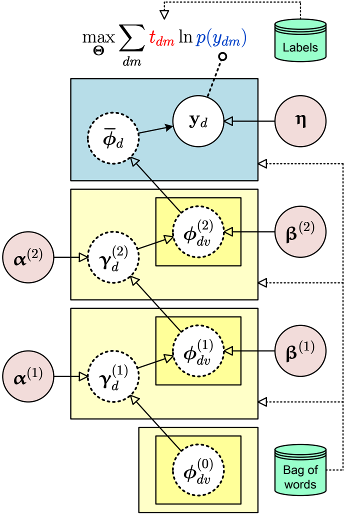
- Variational posterior distribution

$$q(\mathbf{z}, \boldsymbol{\theta} | \boldsymbol{\phi}, \boldsymbol{\gamma}) = \prod_d q(\boldsymbol{\theta}_d | \boldsymbol{\gamma}_d) \prod_n q(\mathbf{z}_{dn} | \boldsymbol{\phi}_{dn})$$

Variational Bayesian Learning for sLDA

Initialize $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$
repeat
    for all document d **do**
        $\phi_{dnk} = 1/K$
        repeat
            $\gamma_{dk} = \alpha_k + \sum_n \phi_{dnk}$
            $\phi_{dnk} \propto \exp\left\{\ln \beta_{kw_n} + \psi(\gamma_{dk}) + \frac{\eta_{mk}}{N_d} - (\mathbf{h}^\top \boldsymbol{\phi}_{dw_n})^{-1} h_k\right\}$
            where $h_k = \sum_c \prod_{i \neq n} (\sum_j \phi_{dnj} \exp(\eta_{cj}/N_d)) \exp(\eta_{ck}/N_d)$
        until γ_{dk} and ϕ_{dnk} converged
    end for
    $\beta_{kv} \propto \sum_{dn} \phi_{dnk} w_{dnv}$
    $\boldsymbol{\alpha} \leftarrow$ Newton-Raphson$(\boldsymbol{\alpha}, \mathbf{g}(\boldsymbol{\alpha}), \mathbf{H}(\boldsymbol{\alpha}))$
    $\boldsymbol{\eta} \leftarrow$ update$(\boldsymbol{\eta}, \boldsymbol{\phi})$
until ELBO converged

Bayesian Unfolding Inference for sLDA



$$p(y_{dm}) = \frac{\exp(\boldsymbol{\eta}_m^\top \bar{\boldsymbol{\phi}}_d)}{\sum_m \exp(\boldsymbol{\eta}_m^\top \bar{\boldsymbol{\phi}}_d)}$$

$$\bar{\phi}_{dk} = \frac{1}{N_d} \sum_v N_{dv} \phi_{dvk}$$

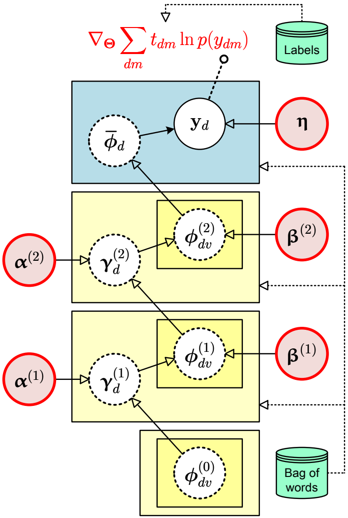
$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\eta}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Bayesian Unfolding Inference for sLDA



$$p(y_{dm}) = \frac{\exp(\boldsymbol{\eta}_m^\top \bar{\boldsymbol{\phi}}_d)}{\sum_m \exp(\boldsymbol{\eta}_m^\top \bar{\boldsymbol{\phi}}_d)}$$

$$\bar{\phi}_{dk} = \frac{1}{N_d} \sum_v N_{dv} \phi_{dvk}$$

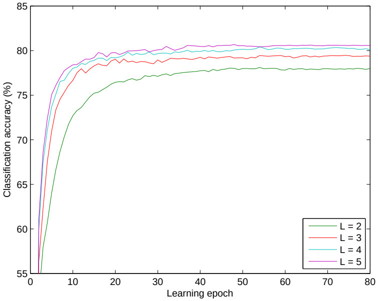
$$\gamma_{dk}^{(l)} = \alpha_k^{(l)} + \sum_v N_{dv} \phi_{dvk}^{(l-1)}$$

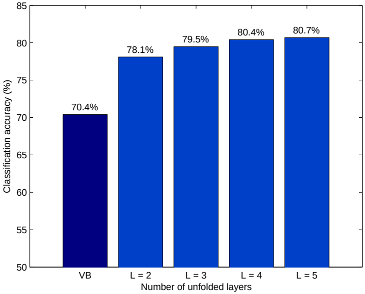
$$\phi_{dvk}^{(l)} \propto \exp\left(\ln \beta_{kv}^{(l)} + \psi(\gamma_{dk}^{(l)})\right)$$

$$\boldsymbol{\Psi}_d = \{\boldsymbol{\gamma}_d, \{\boldsymbol{\phi}_d\}\}$$

$$\boldsymbol{\Theta} = \{\boldsymbol{\eta}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$$

Document Classification





Comparison of BUI and VB

- Variational Bayesian (VB) Inference
  - VB E-step iteration
  - VB M-step
  - Inference with one model parameters
  - Model inference criteria based on evidence lower bound
- Bayesian Unfolding Inference (BUI)
  - Propagation layer-by-layer
  - Back-propagation
  - Inference with a cascade of untied models
  - Model inference criteria based on end performance criteria

Comparison of BUI and VB

- Variational Bayesian (VB) Inference
  - VB E-step iteration
  - VB M-step
  - Inference with one model parameters
  - Model inference criteria based on evidence lower bound
- Bayesian Unfolding Inference (BUI)
  - Propagation layer-by-layer
  - Back-propagation
  - Inference with a cascade of untied models
  - Model inference criteria based on end performance criteria

Comparison of BUI and VB

- Variational Bayesian (VB) Inference
  - VB E-step iteration
  - VB M-step
  - Inference with one model parameters
  - Model inference criteria based on evidence lower bound

- Bayesian Unfolding Inference (BUI)
  - Propagation layer-by-layer
  - Back-propagation
  - Inference with a cascade of untied models
  - Model inference criteria based on end performance criteria

Comparison of BUI and VB

- Variational Bayesian (VB) Inference
  - VB E-step iteration
  - VB M-step
  - Inference with one model parameters
  - Model inference criteria based on evidence lower bound
- Bayesian Unfolding Inference (BUI)
  - Propagation layer-by-layer
  - Back-propagation
  - Inference with a cascade of untied models
  - Model inference criteria based on end performance criteria

Outline

Conclusions

- We have presented a novel tensor factorized neural network which is a generalization of multilayer perceptron
- Tensor factorized error backpropagation for optimization of parameters
- We have presented a domain adaptive neural network that transfers knowledge though jointly training a classifier and a domain-invariant feature extractor
- We turned the VB inference procedure of topic model into a Bayesian unfolding network
- This enables us to exploit an error back-propagation algorithm to meet the end performance

Future Works

- Convolutional or recurrent tensor factorized neural network
  - use convolution or recurrent operation to get extra temporal or spatial information
- Bayesian tensor factorized neural network & Bayesian domain adaptive neural network - model regularization
- Extend current domain adaptation to active transfer learning
- Variational auto-encoder for stochastic error back-propagation - manifold learning, transfer learning, etc
- Applications to different types of information system

Thank you for listening!