Outline

1 Introduction

2 Learning Algorithms

3 Learning Models

4 Deep Learning
- 4.1. Deep neural network
- 4.2. Deep belief network
- 4.3. Stacking auto-encoder
- 4.4. Variational auto-encoder
- 4.5. Deep transfer learning

5 Case Studies

6 Future Direction

Outline

1 Introduction

2 Learning Algorithms

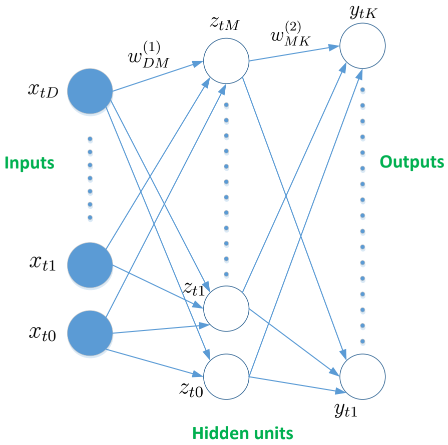
3 Learning Models

4 Deep Learning
- 4.1. Deep neural network
- 4.2. Deep belief network
- 4.3. Stacking auto-encoder
- 4.4. Variational auto-encoder
- 4.5. Deep transfer learning

5 Case Studies

6 Future Direction

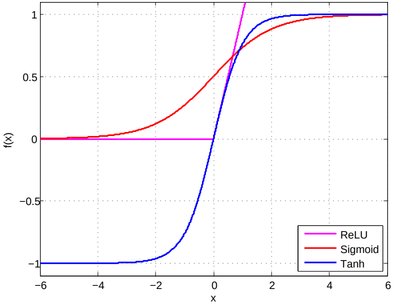
Neural network



Multilayer perceptron

Nonlinear activation

$$y_{tk} = y_k(\mathbf{x}_t, \mathbf{w}) = f\left(\sum_{j=0}^{M} w_{jk}^{(2)} f\left(\sum_{i=0}^{D} w_{ij}^{(1)} x_{ti}\right)\right)$$

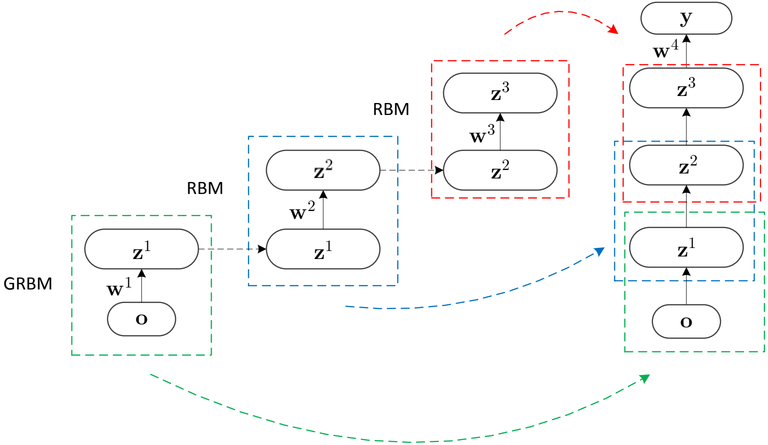


activation function $f(\cdot)$

Deep learning

- Deep belief networks (DBN) obtained great results due to good initialization and deep model structure
  - pre-train each layer from bottom up
  - each pair of layers is a restricted Boltzmann machine
  - jointly fine tune all layers using back-propagation
- Deep neural network (DNN)
  - discriminative model works for classification tasks
  - empirically works well for image recognition, speech recognition, information retrieval and many others
  - no theoretical guarantee

DBN-DNN training

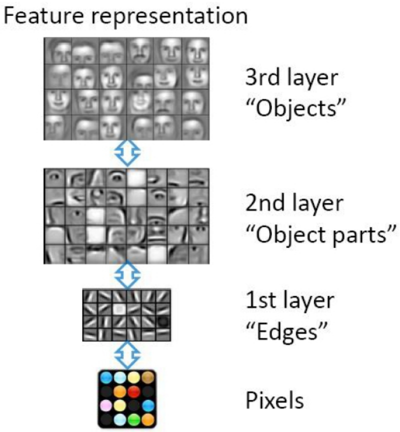


Why go deep?

- Deep architecture can be representationally efficient
  - fewer computational units for the same function
- Deep representation might allow for a hierarchical representation
  - allows non-local generalization
  - comprehensibility
- Multiple levels of latent variables allow combinatorial sharing of statistical strength
- Deep architecture works well for representation of vision, audio, NLP, music and many other technical data

Different level of abstraction

- Hierarchical learning
  - natural progression from low level to high level structure as seen in natural complexity
  - easier to monitor what is being learnt and to guide the machine to better subspaces
  - a good lower level representation can be used in different tasks

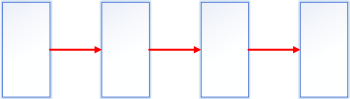


Trainable feature hierarchy

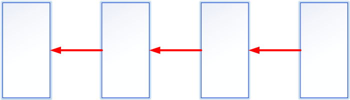
- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → ... → phone → phoneme → word

Deep architecture

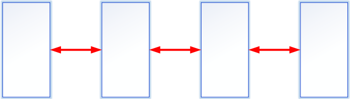
- Feed-forward: multilayer neural nets, convolutional nets



- Feed-back: stacked sparse coding, deconvolutional nets



- Bi-directional: deep Boltzmann machines, stacked auto-encoders

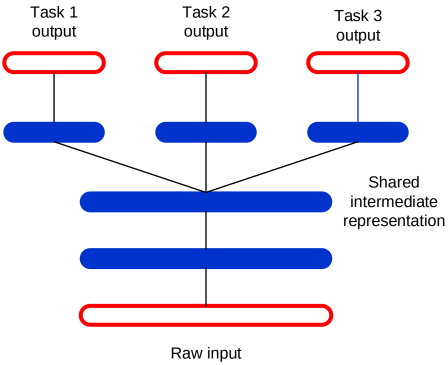


Training strategy

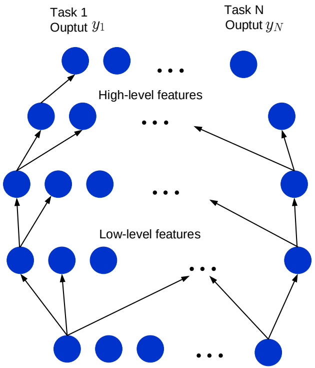
- Purely supervised
  - initialize parameters randomly
  - train in supervised mode
    - typically with SGD, using backprop to compute gradients
  - used in most practical systems for speech and image recognition
- Unsupervised, layerwise + supervised classifier on top
  - train each layer unsupervised, one after the other
  - train a supervised classifier on top, keeping the other layers fixed
  - good when very few labeled samples are available
- Unsupervised, layerwise + global supervised fine-tuning
  - train each layer unsupervised, one after the other
  - add a classifier layer, and retrain the whole thing supervised
  - good when label set is poor
- Unsupervised pre-training often uses the regularized auto-encoders

Generalizable learning

- Shared representation
  - multi-task learning
  - unsupervised training

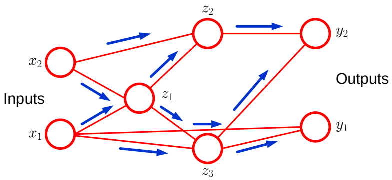


- Partial feature sharing
  - mixed mode learning
  - composition of functions

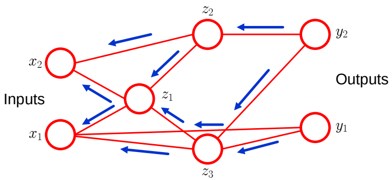


Forward & backward passes

- Forward propagation
  - sum inputs, produce activation, feed-forward

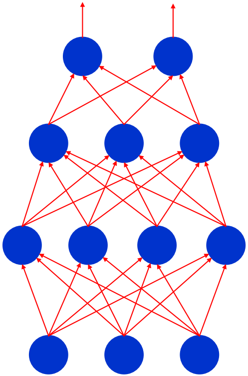


- Training: back propagation of error
  - calculate total error at the top
  - calculate contributions to error at each step going backwards



Deep neural network

- Simple to construct
  - sigmoid nonlinearity for hidden layers
  - softmax for the output layer
- Backpropagation does not work well if randomly initialized [Bengio et al., 2007]
  - deep networks trained without unsupervised pretraining perform worse than shallow networks



	train.	valid.	test
DBN, unsupervised pre-training	0%	1.2%	1.2%
Deep net, auto-associator pre-training	0%	1.4%	1.4%
Deep net, supervised pre-training	0%	1.7%	2.0%
$\boxed{\text{Deep net, no pre-training}}$	.004%	2.1%	2.4%
Shallow net, no pre-training	.004%	1.8%	1.9%

(Bengio et al., NIPS 2007)

Problems and solvers with back propagation

- Gradient is progressively getting more dilute
  - below top few layers, correction signal is minimal
- Gets stuck in local minima
  - random initialization: may start out far from good regions
- In usual settings, we can use only labeled data
  - almost all data are unlabeled
  - the brain can learn from unlabeled data
- Use unsupervised learning via greedy layer-wise training
  - allow abstraction to develop naturally from one layer to another
  - help the network initialize with good parameters
- Perform supervised top-down training as final step
  - refine the features in intermediate layers more relevant for the task

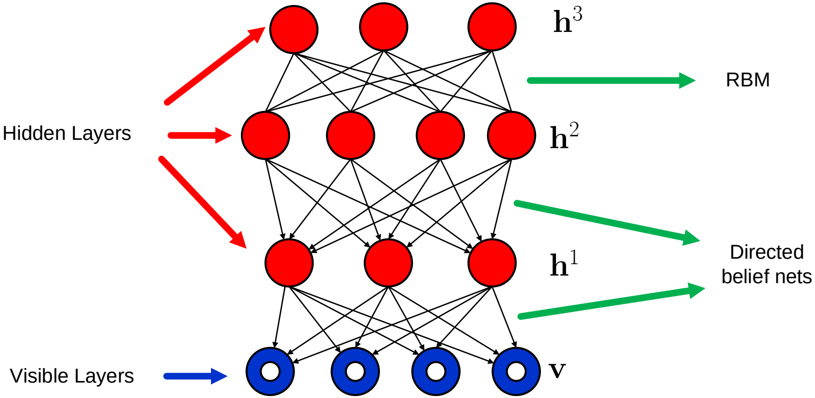
Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning
5. Case Studies
6. Future Direction

Deep belief network

- Deep belief network (DBN) is a probabilistic generative model
- Deep architecture with multiple hidden layers
- Unsupervised pre-learning provides a good initialization
  - maximizing the lower-bound of the log-likelihood of data
- Supervised fine-tuning
  - generative: up-down algorithm
  - discriminative: back propagation

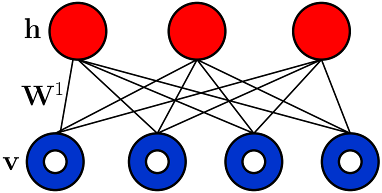
Model structure



$$p(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^l) = p(\mathbf{v}|\mathbf{h}^1)p(\mathbf{h}^1|\mathbf{h}^2)\ldots p(\mathbf{h}^{l-2}|\mathbf{h}^{l-1})p(\mathbf{h}^{l-1}|\mathbf{h}^l)$$

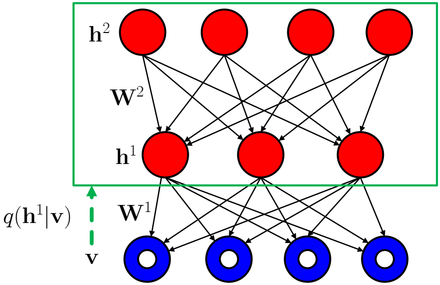
Greedy training

- First step:
  - construct an RBM with an input layer **v** and a hidden layer **h**
  - train the RBM



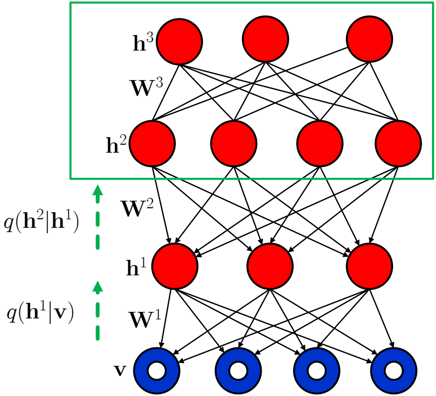
Greedy training

- Second step:
  - Stack another hidden layer on top of the RBM to form a new RBM
  - Fix $\mathbf{W}^1$, sample $\mathbf{h}^1$ from $q(\mathbf{h}^1|\mathbf{v})$ as input. Train $\mathbf{W}^2$ as RBM



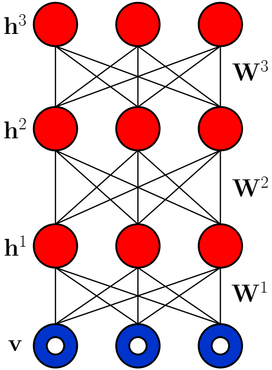
Greedy training

- Third step:
  - continue to stack layers on top of the network, train it as previous step, with sample sampled from $q(\mathbf{h}^2|\mathbf{h}^1)$
- And so on...



Deep Boltzmann machine

$$p(\mathbf{v}) = \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} \frac{1}{Z} \exp[\mathbf{v}^T \mathbf{W}^1 \mathbf{h} + (\mathbf{h}^1)^\top \mathbf{W}^2 \mathbf{h}^2 + (\mathbf{h}^2)^\top \mathbf{W}^3 \mathbf{h}^3]$$

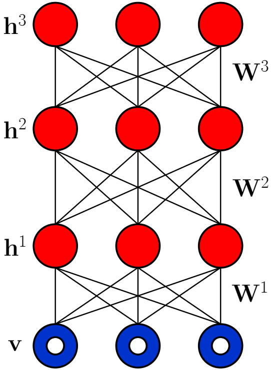


- Undirected connections between all layers. No connections between the nodes in the same layer
- High-level representations are built from unlabeled inputs. Labeled data is used to only slightly fine-tune the model

[Salakhutdinov and Hinton, 2009]

Training procedure

- Pre-training
  - initialize from stacked RBMs
- Generative fine-tuning
  - positive phase: variational or mean-field approximation
  - negative phase: persistent chain & stochastic approximation
- Discriminative fine-tuning
  - back-propagation



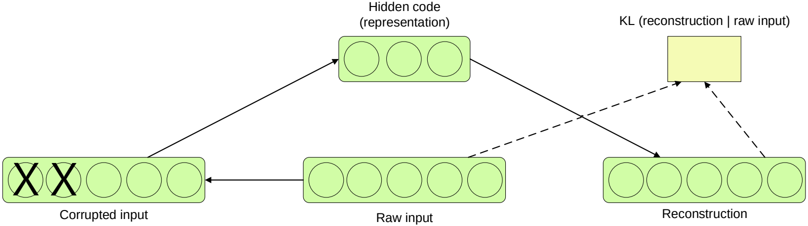
Why greedy layer wise training works

- Regularization hypothesis
  - pre-training is constraining the parameters in a region relevant to unsupervised dataset
  - better generalization - representations that better describe unlabeled data are more discriminative for labeled data
- Optimization hypothesis
  - unsupervised training initializes lower level parameters near localities of better minima than random initialization can

Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning
5. Case Studies
6. Future Direction

Denoising auto-encoder

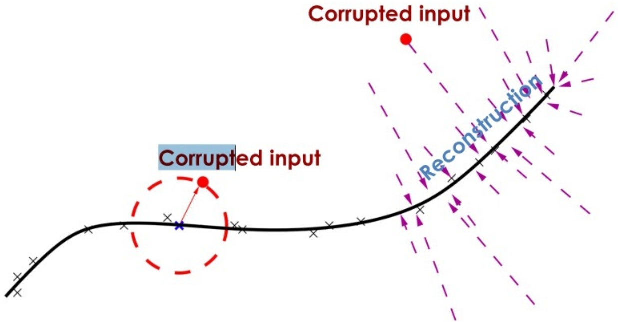


[Vincent et al., 2008]

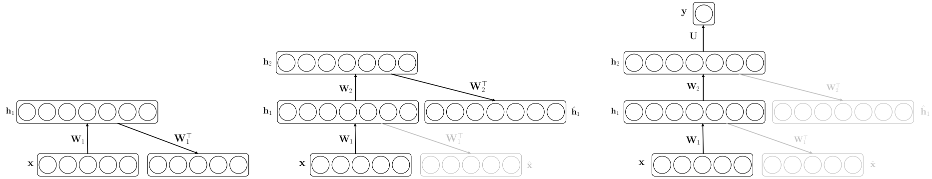
- Corrupt the input, e.g. set 25% of inputs to 0
- Reconstruct the uncorrupted input
- Use the uncorrupted encoding as input to next level

Manifold learning perspective

- Learn a vector field towards higher probability regions
- Minimize the variational lower bound on a generative model
- Correspond to the regularized score matching on an RBM

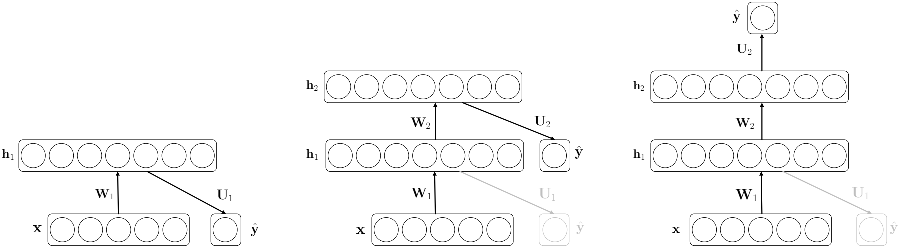


Stacked denoising auto-encoders



Greedy layer-wise learning

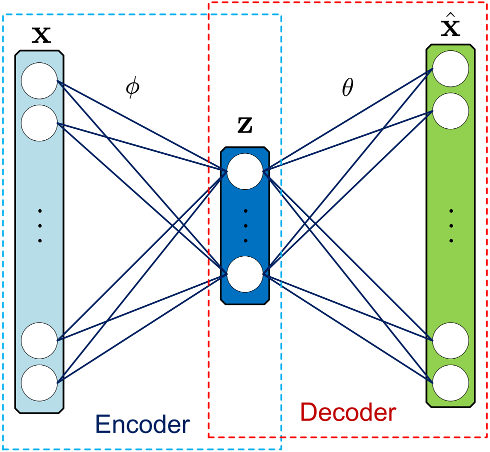
- Start with the lowest level and stack upwards
- Train each layer of auto-encoder using the intermediate codes or features from the layer below
- Top layer can have a different output, e.g. softmax non-linearity, to provide an output for classification



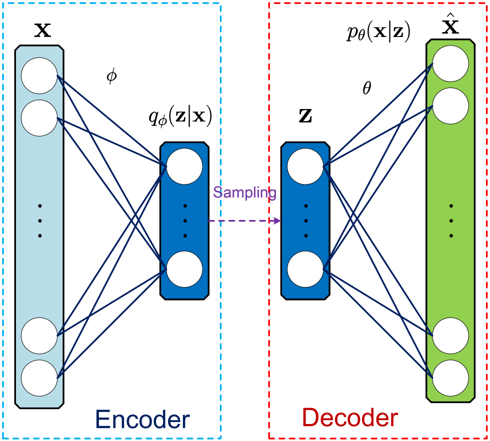
Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning
5. Case Studies
6. Future Direction

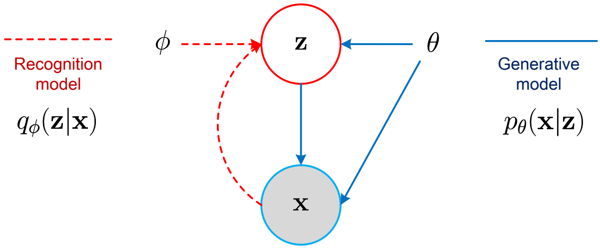
Auto-encoder



Variational auto-structure



Graphical model



[Kingma and Welling, 2014]

- Mean-field approach requires analytical solutions $\mathbb{E}_q$, which are intractable in the case of neural network
- Use neural network and sample the latent variables $\mathbf{z}$ from variational posterior

Variational inference

- Variational Bayesian inference aims to find a variational distribution $q(\mathbf{z}|\mathbf{x})$ that is maximally close to the original true posterior distribution $p(\mathbf{z}|\mathbf{x})$
- According to the evidence decomposition, we have

$$p(\mathbf{x}) = \mathcal{L}(q) + \mathrm{KL}(q\|p)$$

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(\mathbf{x}, \mathbf{z})] + \mathbb{H}_q[\mathbf{z}]$$

$$\mathrm{KL}(q\|p) = -\mathbb{E}_q[\log p(\mathbf{z}|\mathbf{x})] - \mathbb{H}_q[\mathbf{z}]$$

Mean field variational inference

- Assume that $q(\mathbf{z}|\mathbf{x})$ can be factorized into the product of individual probability distributions

$$q(\mathbf{z}|\mathbf{x}) = \prod_{n=1}^{N} q(z_n|x_n)$$

- We can perform the coordinate ascent for each factorized variational distributions by

$$\hat{q}(z_j|x_j) \propto \exp(\mathbb{E}_{q(z_{i\neq j})}[\log p(\mathbf{x}, \mathbf{z})])$$

- Model parameters are learned by maximizing the variational lower bound

$$\begin{aligned} \log p(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}) || p_\omega(\mathbf{z})) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})] \\ &\triangleq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f_{\boldsymbol{\Theta}}(\mathbf{x}, \mathbf{z})] \\ &\triangleq \mathcal{L}_{\boldsymbol{\Theta}} \end{aligned}$$

where $\boldsymbol{\Theta} = \{\theta, \phi, \omega\}$

Stochastic backpropagation

Objective:

$$\mathcal{L}_{\Theta} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f_{\Theta}(\mathbf{x}, \mathbf{z})]$$

Gradient:

Step1: sample $\mathbf{z}^{(l)}$ from $q_\phi(\mathbf{z}|\mathbf{x})$

Step2: $\mathcal{L}_{\Theta} \simeq f_{\Theta}(\mathbf{x}|\mathbf{z}^{(l)})$

Step3: $\nabla_{\Theta}\mathcal{L}_{\Theta} \simeq \nabla_{\Theta} f_{\Theta}(\mathbf{x}, \mathbf{z}^{(l)})$

- Problem: high variance by directly sampling $\mathbf{z}$ [Rezende et al., 2014]

Stochastic gradient variational Bayes

Objective: $\mathcal{L}_{\boldsymbol{\Theta}} = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f_{\boldsymbol{\Theta}}(\mathbf{x}, \mathbf{z})]$

Gradient:

Step1: sample $\boldsymbol{\epsilon}^{(l)}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$

Step2: $\mathbf{z}^{(l)} = \boldsymbol{\mu}_{\mathbf{z}} + \boldsymbol{\sigma}_{\mathbf{z}} \odot \boldsymbol{\epsilon}^{(l)}$

Step3: $\mathcal{L}_{\boldsymbol{\Theta}} \simeq f_{\boldsymbol{\Theta}}(\mathbf{x}|\mathbf{z}^{(l)})$

Step4: $\nabla_{\boldsymbol{\Theta}}\mathcal{L}_{\boldsymbol{\Theta}} \simeq \nabla_{\boldsymbol{\Theta}} f_{\boldsymbol{\Theta}}(\mathbf{x}, \mathbf{z}^{(l)})$

- Reduce the variance caused by directly sampling **z**

Outline

1. Introduction
2. Learning Algorithms
3. Learning Models
4. Deep Learning
   - 4.1. Deep neural network
   - 4.2. Deep belief network
   - 4.3. Stacking auto-encoder
   - 4.4. Variational auto-encoder
   - 4.5. Deep transfer learning
5. Case Studies
6. Future Direction

Why transfer learning?

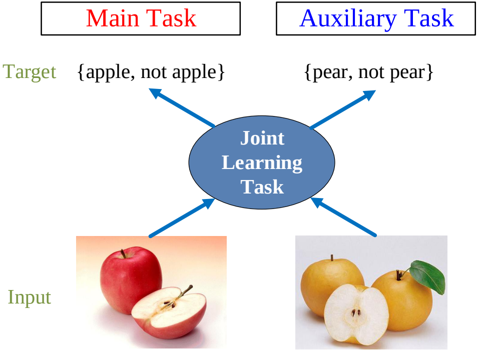
- Mismatch between training and test data in speaker recognition always exists
- Traditional machine learning works well under an assumption that training and test data follow the same distribution
  - real-world data may not follow this assumption
- Feature-based domain adaptation is a common approach
  - allow knowledge to be transferred across domains through learning a good feature representation
- Co-train for feature representation and speaker recognition without labeling in target domain

Transfer learning

- Let $\mathcal{D} = \{\mathcal{X}, p(X)\}$ denote a domain
  - feature space $\mathcal{X}$
  - marginal probability distribution $p(X)$
  - $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_T\} \subset \mathcal{X}$
- 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 $\mathcal{T}_S \neq \mathcal{T}_T$

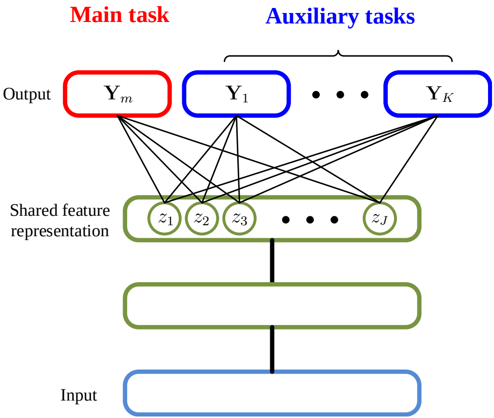
Multi-task learning

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

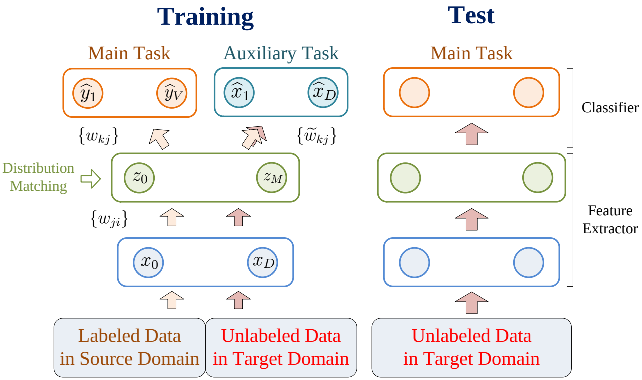


Multi-task neural network learning

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



Learning strategy and task



- Semi-supervised learning is conducted under multiple objectives