CS420, Machine Learning, Lecture 13

Transfer Learning

Weinan Zhang

Shanghai Jiao Tong University

http://wnzhang.net

http://wnzhang.net/teaching/cs420/index.html

Transfer Learning Materials

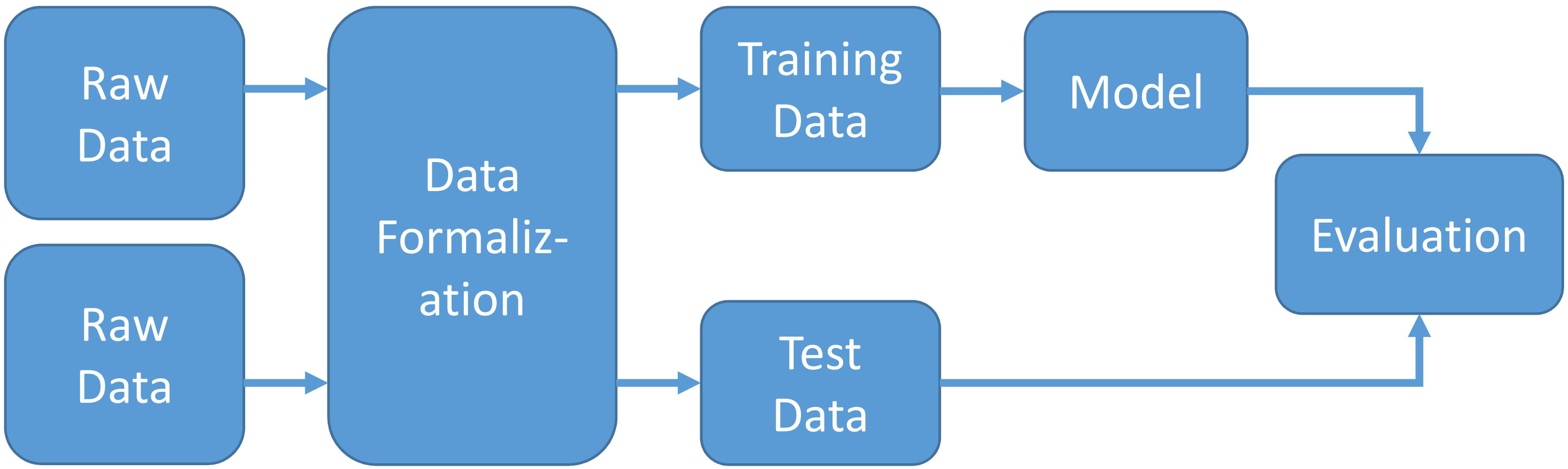
Our course on TL is mainly based on the materials from Prof. Qiang Yang and his students

Prof. Qiang Yang

- Chair Professor, Department Head of CSE, HKUST
- http://www.cs.ust.hk/~qyang/
- SJ Pan, Q Yang. A survey on transfer learning. IEEE TKDE 2010.
- 2800+ citations on this survey paper (May 2017)

Machine Learning Process

$$\min_{\theta} \frac{1}{N} \sum_{(x_i, y_i) \in D_{\text{train}}} \mathcal{L}(y_i, f_\theta(x_i)) + \lambda ||\theta||_2^2$$



$$\text{Test Error} = \frac{1}{N} \sum_{(x_i, y_i) \in D_{\text{test}}} \mathcal{L}(y_i, f_\theta(x_i))$$

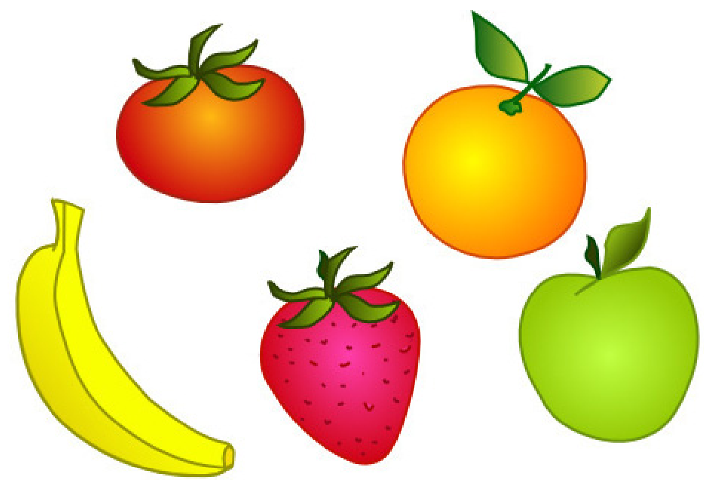
- Assumption: training and test data has the same distribution

Practical Cases

- Data distributions *p*(*x*) change across different domains or vary over time

$$\mathcal{X}_S \neq \mathcal{X}_T \quad \text{or} \quad p_S(x) \neq p_T(x)$$

Real images



Cartoon images

Practical Cases

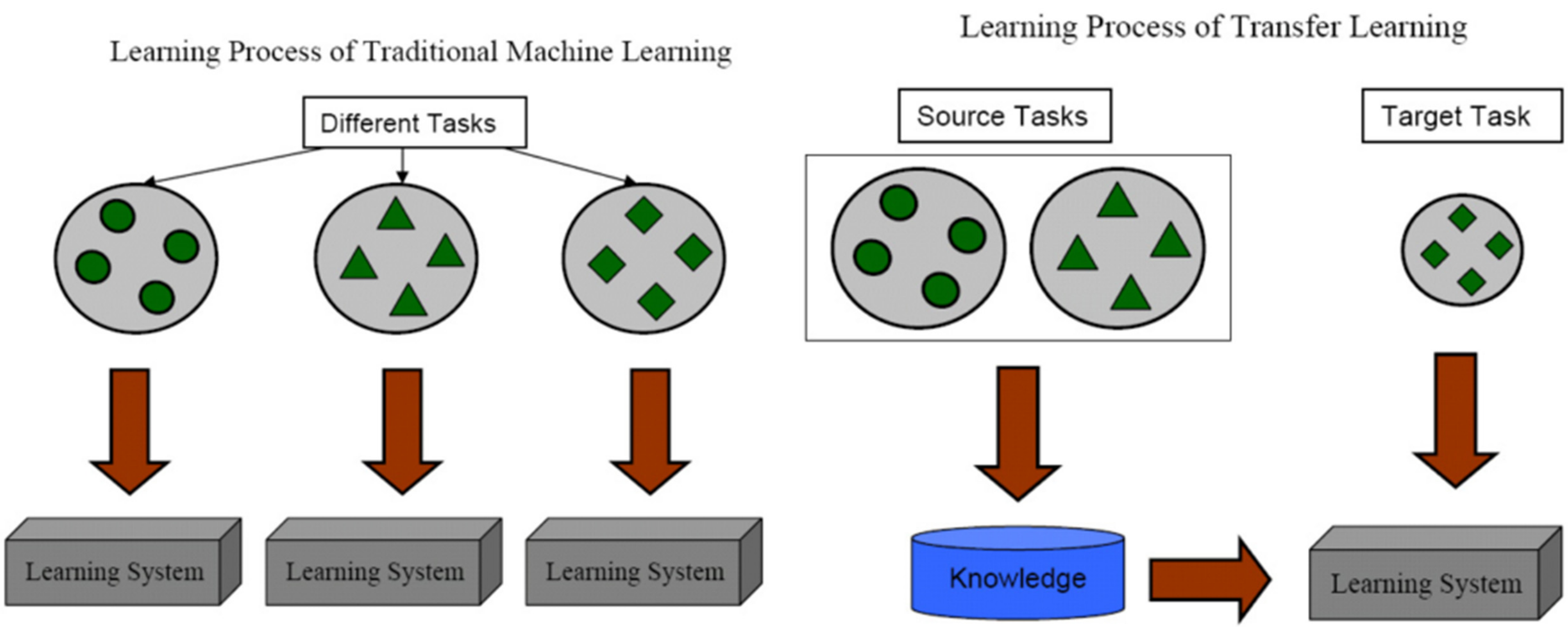
- Data dependencies $p(y|x)$ could be also different

$$\mathcal{Y}_S \neq \mathcal{Y}_T \quad \text{or} \quad p_S(y|x) \neq p_T(y|x)$$

Apple recognition

Pear recognition

Transfer Learning



(a) Traditional Machine Learning (b) Transfer Learning

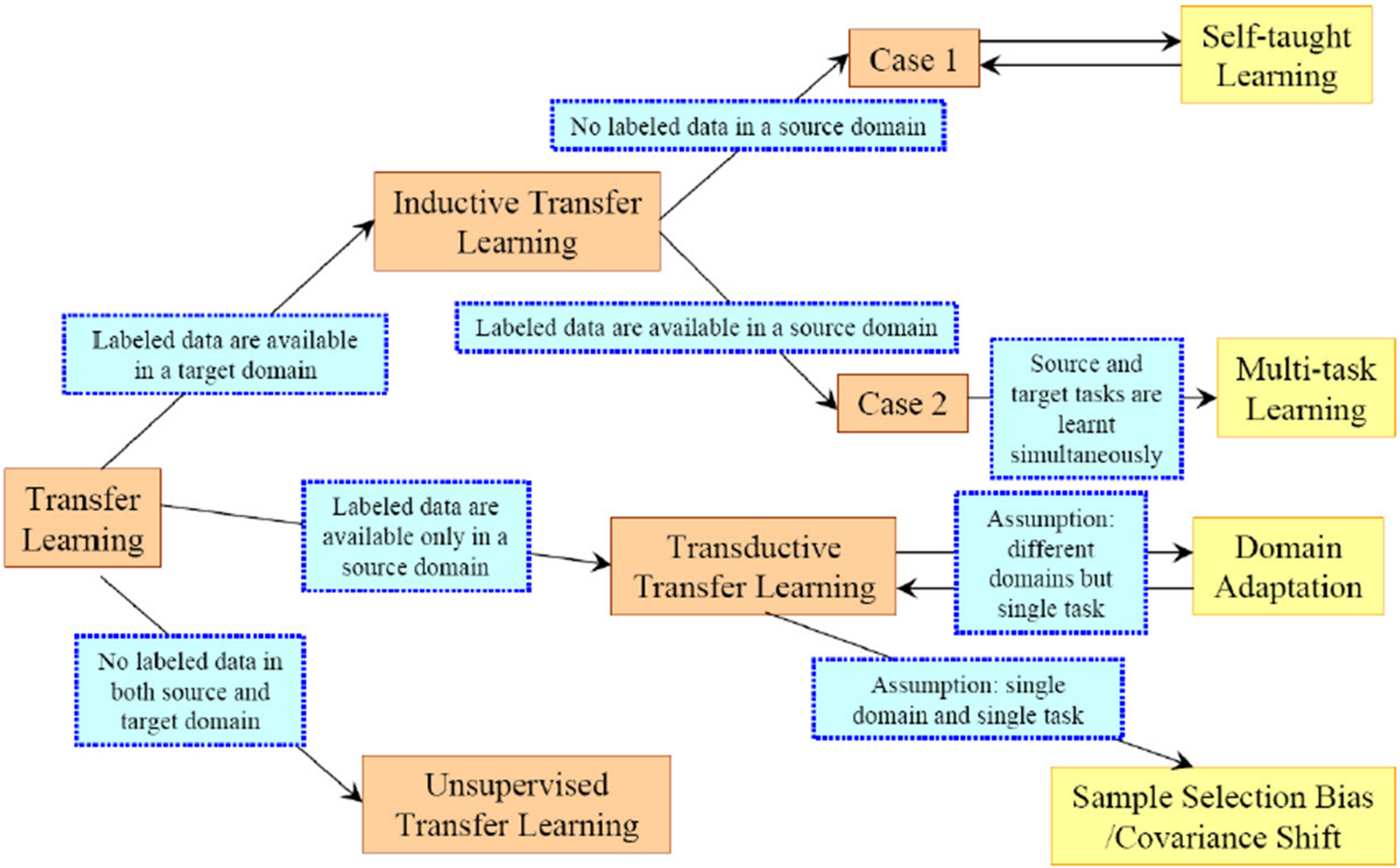
Notation and Definition of TL

- Notation
  - A **domain** $\mathcal{D} = \{\mathcal{X}, p(x)\}$
    - Feature space $\mathcal{X}$
    - Data distribution $p(x)$
  - A **task** $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$
    - Label space $\mathcal{Y}$
    - Objective predictive function $f(\cdot)$
- Definition
  - Given a **source domain** $\mathcal{D}_S$ with corresponding learning task $\mathcal{T}_S$ and a **target domain** $\mathcal{D}_T$ with corresponding learning task $\mathcal{T}_T$
  - **transfer learning** is the process of improving the target predictive function $f_T(\cdot)$ by using the related information from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$

Explanation

- $\mathcal{D}_S \neq \mathcal{D}_T$
  - $\mathcal{X}_S \neq \mathcal{X}_T$
    - Heterogeneous transfer learning
    - Two sets of documents are described in different languages
  - $P(X_S) \neq P(X_T)$
    - Domain adaptation
    - Two sets of documents focus on different topics
- $\mathcal{T}_S \neq \mathcal{T}_T$
  - $\mathcal{Y}_S \neq \mathcal{Y}_T$
    - Source has two classes: positive or negative; target adds one class: neutral
  - $P_S(y|x) \neq P_T(y|x)$
    - A word can have different meanings in two domains

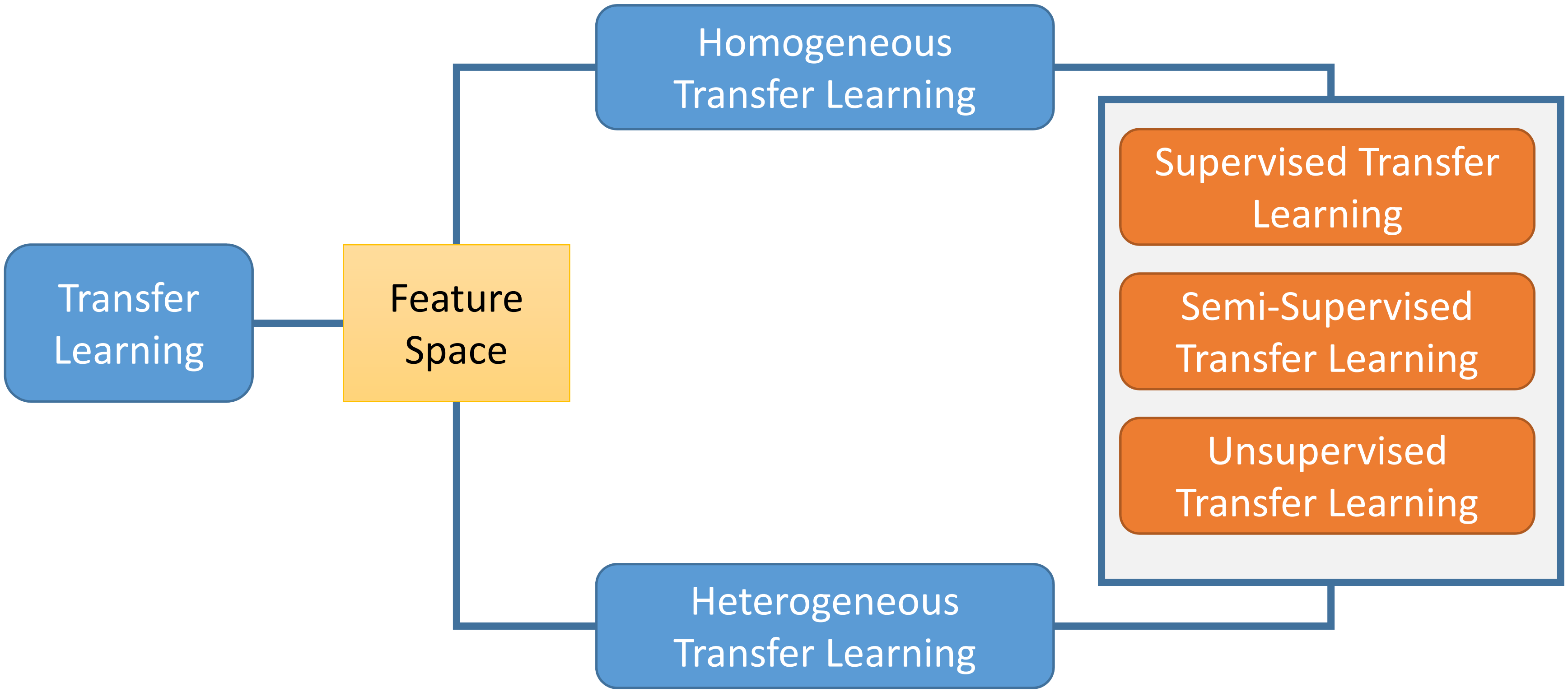
Categorization of Transfer Learning



Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering* 22.10 (2010): 1345-1359.

Transfer Learning Settings

- Homogeneous/heterogeneous transfer learning



Transfer Learning Methods

- Instance Transfer
  - Reweight instances of target data according to source
- Feature Transfer
  - Mapping features of source and target data in a common space
- Parameter Transfer
  - Learn target model parameters according to source model

Relational approaches are relatively unpopular, thus omitted in this talk

Transfer Learning Methods

- Instance Transfer
  - Reweight instances of target data according to source

- Feature Transfer
  - Mapping features of source and target data in a common space

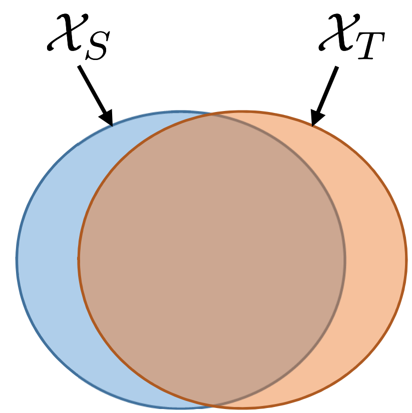
- Parameter Transfer
  - Learn target model parameters according to source model

Relational approaches are relatively unpopular, thus omitted in this talk

Instance-based Transfer Learning

- General assumption
  - Source and target domains have a lot of overlapping features or even share the same feature spaces

$$\mathcal{X}_S \simeq \mathcal{X}_T$$



  - Label space should be the same

$$\mathcal{Y}_S \simeq \mathcal{Y}_T$$

- Example applications
  - Electronic medical record across different departments
  - Sentiment analysis over different topics

Instance TL Case 1: Domain Adaption

- Problem setting
  - Given source domain labeled data $D_S = \{x_{S_i}, y_{S_i}\}_{i=1}^{n_S}$ and target domain data $D_T = \{x_{T_i}\}_{i=1}^{n_T}$
  - learn f_T such that the loss on target data is small

$$\sum_i \mathcal{L}(f_T(x_{T_i}), y_{T_i})$$

  - where y_{T_i} is unknown.
- Assumption
  - The same label space $\mathcal{Y}_S = \mathcal{Y}_T$
  - The same dependency $p(y_S|x_S) = p(y_T|x_T)$
  - (Almost) the same feature space $\mathcal{X}_S \simeq \mathcal{X}_T$
  - Different data distribution $p_S(x) \neq p_T(x)$

Importance Sampling for Domain Adaption

- Importance sampling

$$
\begin{aligned}
\theta^* &= \arg\min_\theta \mathbb{E}_{(x,y)\sim p_T}[\mathcal{L}(y, f_\theta(x))] \\
&= \arg\min_\theta \int_{(x,y)} p_T(x)\mathcal{L}(y, f_\theta(x))dx \\
&= \arg\min_\theta \int_{(x,y)} p_S(x)\frac{p_T(x)}{p_S(x)}\mathcal{L}(y, f_\theta(x))dx \\
&= \arg\min_\theta \mathbb{E}_{(x,y)\sim p_S}\Big[\frac{p_T(x)}{p_S(x)}\mathcal{L}(y, f_\theta(x))\Big]
\end{aligned}
$$

- Re-weight each instance by $\beta(x) = \dfrac{p_T(x)}{p_S(x)}$

Importance Sampling for Domain Adaption

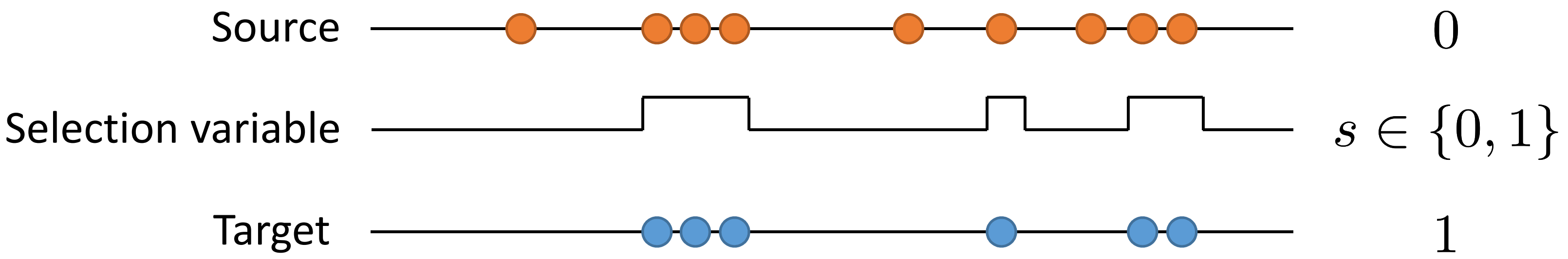
- How to estimate $\beta(x) = \dfrac{p_T(x)}{p_S(x)}$

- A simple solution would be to first estimate $p_S(x)$ and $p_T(x)$ respectively, and then calculate $\beta(x)$
  - May suffer from huge variance problem

- A more practical solution is to estimate $\dfrac{p_T(x)}{p_S(x)}$ directly

Importance Sampling for Domain Adaption

- Imagine a rejection sampling process, and view the target domain as samples from the source domain



- Probabilistic density function (p.d.f.) relationship

$$p_T(x) \propto p_S(x) p(s = 1|x)$$

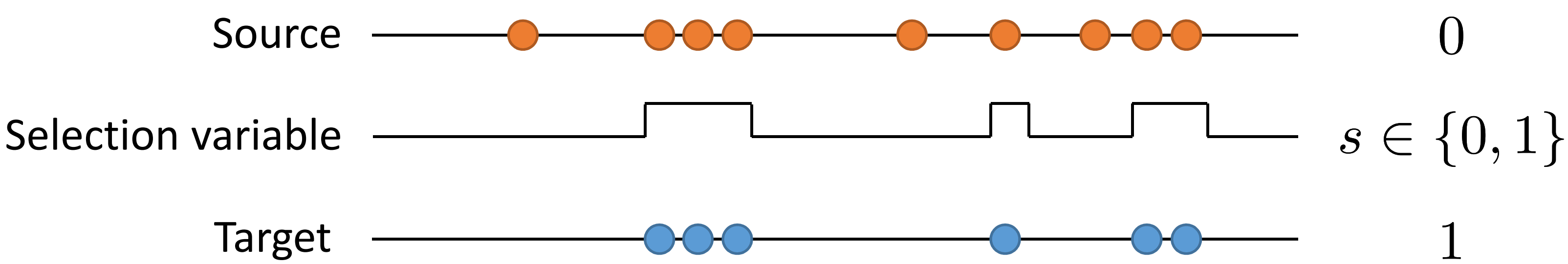
- And we estimate *p*(*s*=1|*x*) as a binary classification model

$$\beta(x) = \frac{p_T(x)}{p_S(x)} \propto p(s = 1|x)$$

Zadrozny, Learning and Evaluating Classifiers under Sample Selection Bias, ICML 2004

Importance Sampling for Domain Adaption

- Imagine a rejection sampling process, and view the target domain as samples from the source domain



- Estimate $p(s=1|x)$ as a binary classification model
  - Label instance from the target domain as 1
  - Label instance from the source domain as 0

$$\beta(x) = \frac{p_T(x)}{p_S(x)} \propto p(s = 1|x)$$

Zadrozny, Learning and Evaluating Classifiers under Sample Selection Bias, ICML 2004

Importance Sampling for Domain Adaption

- How to estimate $\beta(x) = \frac{p_T(x)}{p_S(x)}$

- Build the estimator with a list of basis functions

$$\hat{\beta}(x) = \sum_{l=1}^{b} \alpha_l \psi_l(x)$$

- The estimated target p.d.f. $\hat{p}_T(x) = \hat{\beta}(x) p_S(x)$

- Minimize KL divergence

$$\min_{\{\alpha_l\}_{l=1}^{b}} \mathrm{KL}[p_T(x) \| \hat{p}_T(x)]$$

Sugiyama *et al*., Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation, NIPS 2007

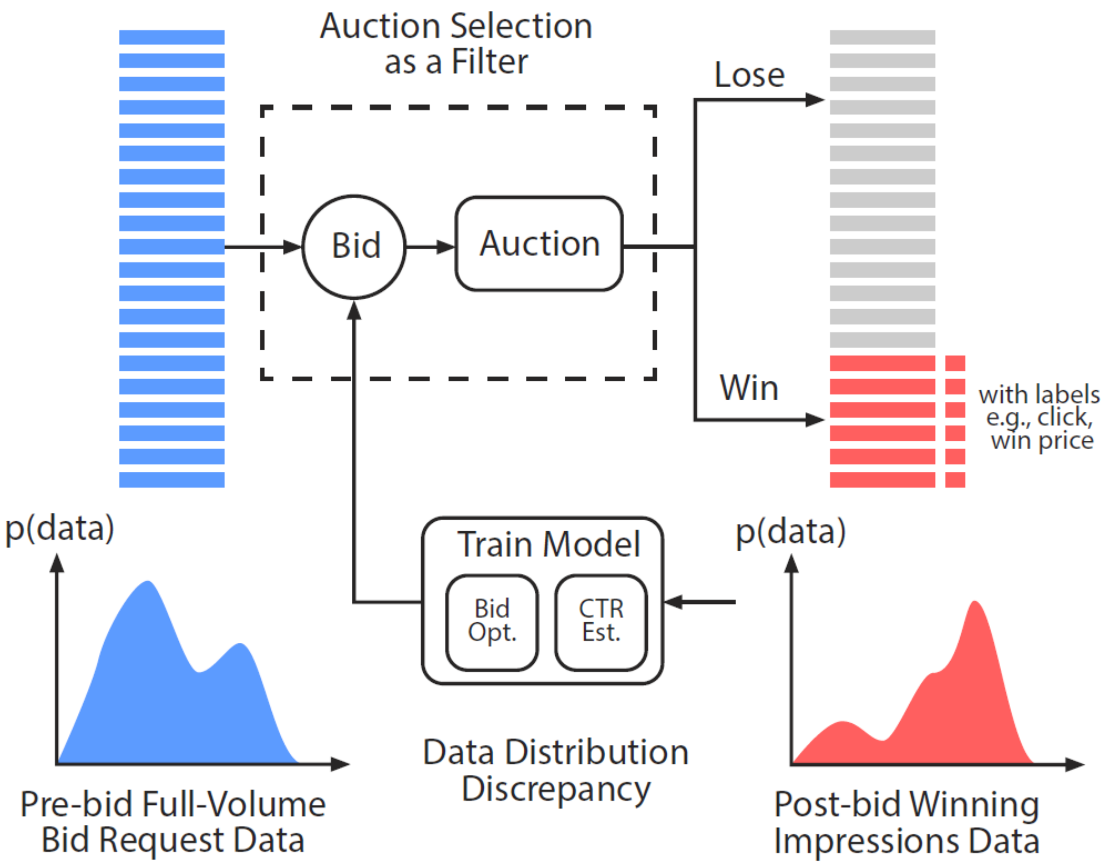
- Minimize squared error

$$\min_{\{\alpha_l\}_{l=1}^{b}} \int_x \left(\hat{\beta}(x) - \beta(x)\right)^2 p_S(x) dx$$

Kanamori et al., A Least-squares Approach to Direct Importance Estimation, JMLR 2009

Unbiased Training in Display Advertising

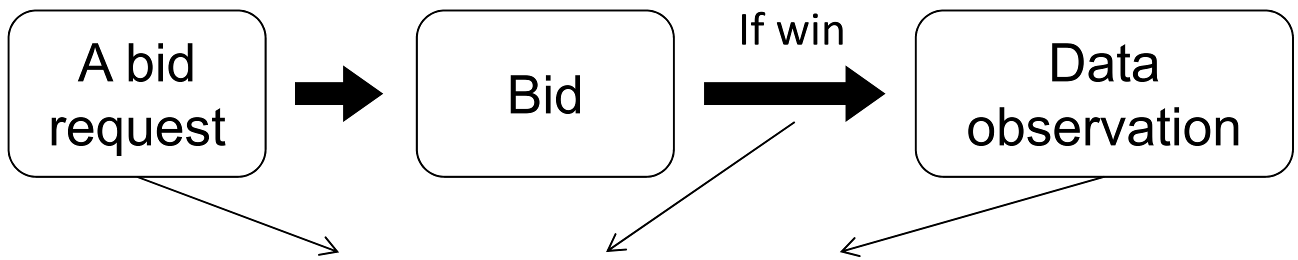
- In display advertising, the label data is observed by an advertiser only when she wins the auction, thus it is biased.



Weinan Zhang et al. Bid-aware Gradient Descent for Unbiased Learning with Censored Data in Display Advertising. KDD 16

Unbiased Learning Framework

- Data observation process



$$p(\boldsymbol{x})w(b_{\boldsymbol{x}}) \propto q(\boldsymbol{x})$$

- Importance sampling

$$\min_{\boldsymbol{\beta}} \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\mathcal{L}(y, f_{\boldsymbol{\beta}}(\boldsymbol{x}))] = \min_{\boldsymbol{\beta}} \mathbb{E}_{\boldsymbol{x}\sim q(\boldsymbol{x})}\Big[\frac{\mathcal{L}(y, f_{\boldsymbol{\beta}}(\boldsymbol{x}))}{w(b_{\boldsymbol{x}})}\Big]$$

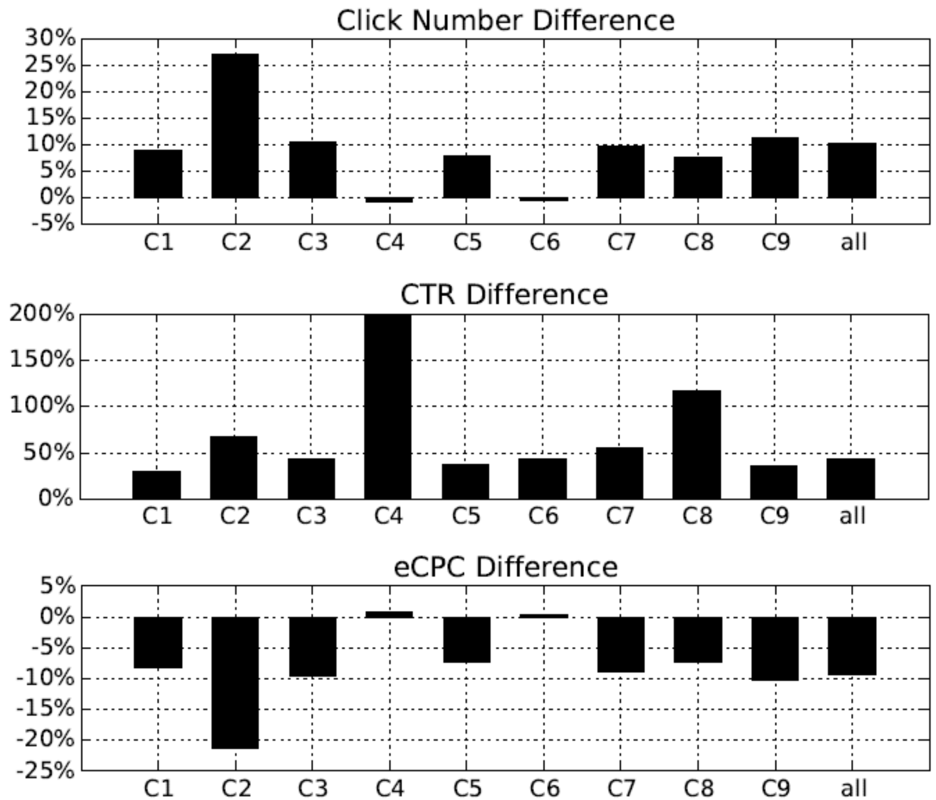
Weinan Zhang et al. Bid-aware Gradient Descent for Unbiased Learning with Censored Data in Display Advertising. KDD 16

Performance Comparison on Yahoo! DSP

- A/B Testing on Yahoo! United States

2.97% AUC lift

Camp.	BIAS AUC.	KMMP AUC	AUC Lift
C1	63.78%	64.12%	0.34%
C2	87.45%	88.58%	1.13%
C3	69.73%	75.52%	5.79%
C4	88.82%	89.55%	0.73%
C5	69.71%	72.29%	2.58%
C6	89.33%	90.70%	1.37%
C7	77.76%	78.92%	1.16%
C8	74.57%	76.98%	2.41%
C9	71.04%	73.12%	2.08%
all	73.48%	76.45%	2.97%



10.3% more clicks

42.8% higher CTR

9.3% lower eCPC

Weinan Zhang et al. Bid-aware Gradient Descent for Unbiased Learning with Censored Data in Display Advertising. KDD 16

Instance TL Case 2: Labels in 2 Domains

- Problem setting
  - Given source domain labeled data $D_S = \{x_{S_i}, y_{S_i}\}_{i=1}^{n_S}$
  - and very limited target domain data $D_T = \{x_{T_i}, y_{T_i}\}_{i=1}^{n_T}$
  - learn f_T such that the loss on target data is small

$$\sum_i \mathcal{L}(f_T(x_{T_i}), y_{T_i})$$

- Assumption
  - The same label space $\mathcal{Y}_S = \mathcal{Y}_T$
  - Different dependency $p(y_S|x_S) \neq p(y_T|x_T)$
  - (Almost) the same feature space $\mathcal{X}_S \simeq \mathcal{X}_T$
  - Different data distribution $p_S(x) \neq p_T(x)$

TrAdaBoost

- For each boosting iteration
  - Use the same strategy as AdaBoost to update the weights of target domain data
  - Use a new mechanism to decrease the weights of misclassified source domain data

Wenyuan Dai et al., Boosting for Transfer Learning, ICML 2007

TrAdaBoost

- Source/target domain data *D* (combined)

$$x_i = \begin{cases} x_{S_i}, & i = 1, \ldots, n \\ x_{T_i}, & i = n+1, \ldots, n+m \end{cases}$$

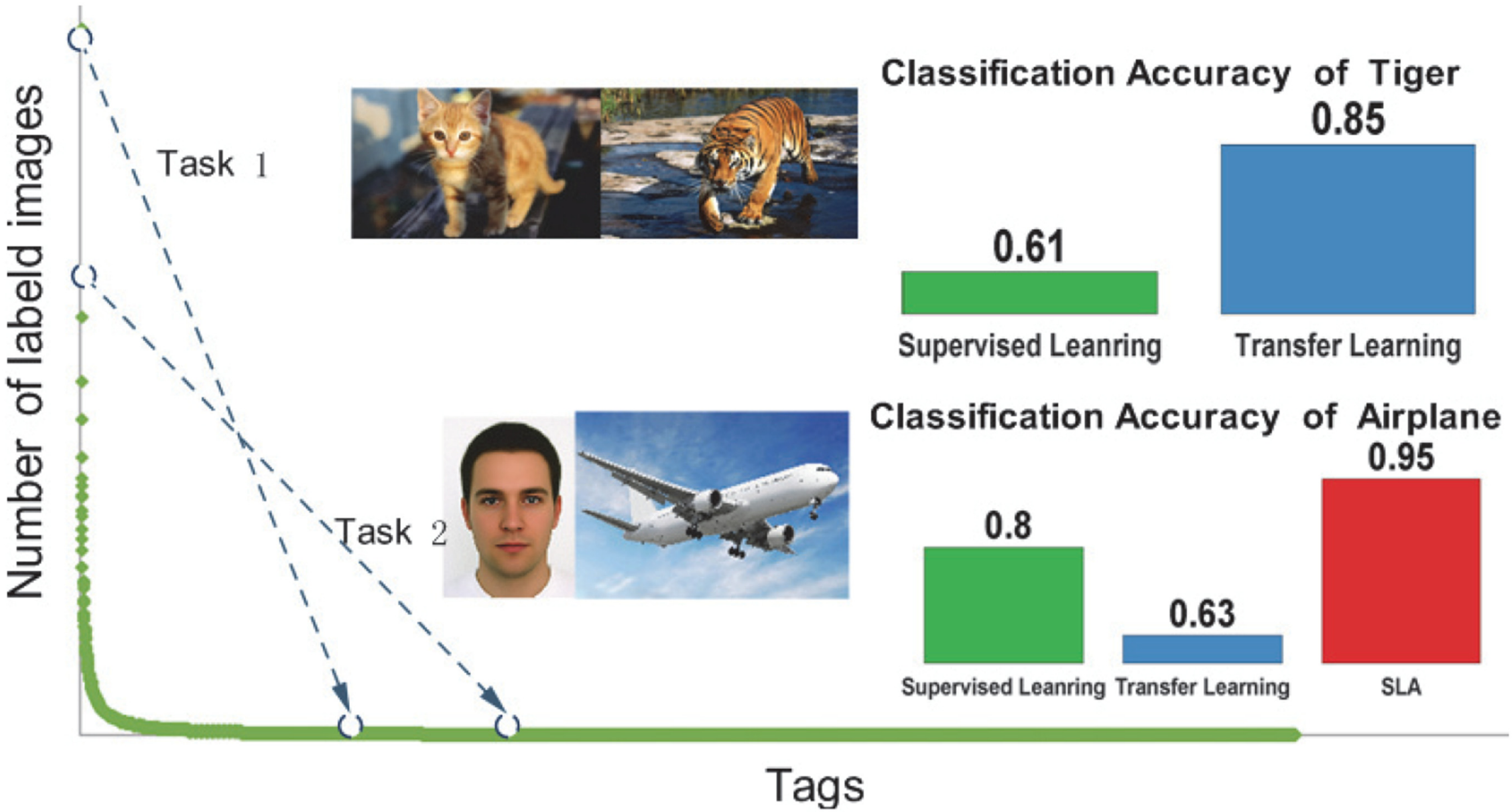
- Initialize the weight vector
- For *t* = 1, …, *N* rounds
  - Set $\mathbf{p}^t = \mathbf{w}^t / (\sum_{i=1}^{n+m} w_i^t)$
  - Learn the model h_t based on the weighted data *D*, $\mathbf{p}^t$
  - Calculate the error on target data $\epsilon_t = \frac{\sum_{i=n+1}^{n+m} w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}$
  - Set $\beta_t = \epsilon_t/(1-\epsilon_t) < 1 \qquad \beta = 1/(1+\sqrt{2 \ln n / N})$
  - Update the new weight vector

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - c(x_i)|}, & i = 1, \ldots, n \\ w_i^t \beta_t^{-|h_t(x_i) - c(x_i)|}, & i = n+1, \ldots, n+m \end{cases}$$

- Output the model $h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \geq \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & \text{otherwise} \end{cases}$

Wenyuan Dai et al., Boosting for Transfer Learning, ICML 2007

Distant Domain Transfer Learning



Ben Tan and Qiang Yang et al. Distant Domain Transfer Learning. AAAI 2017.

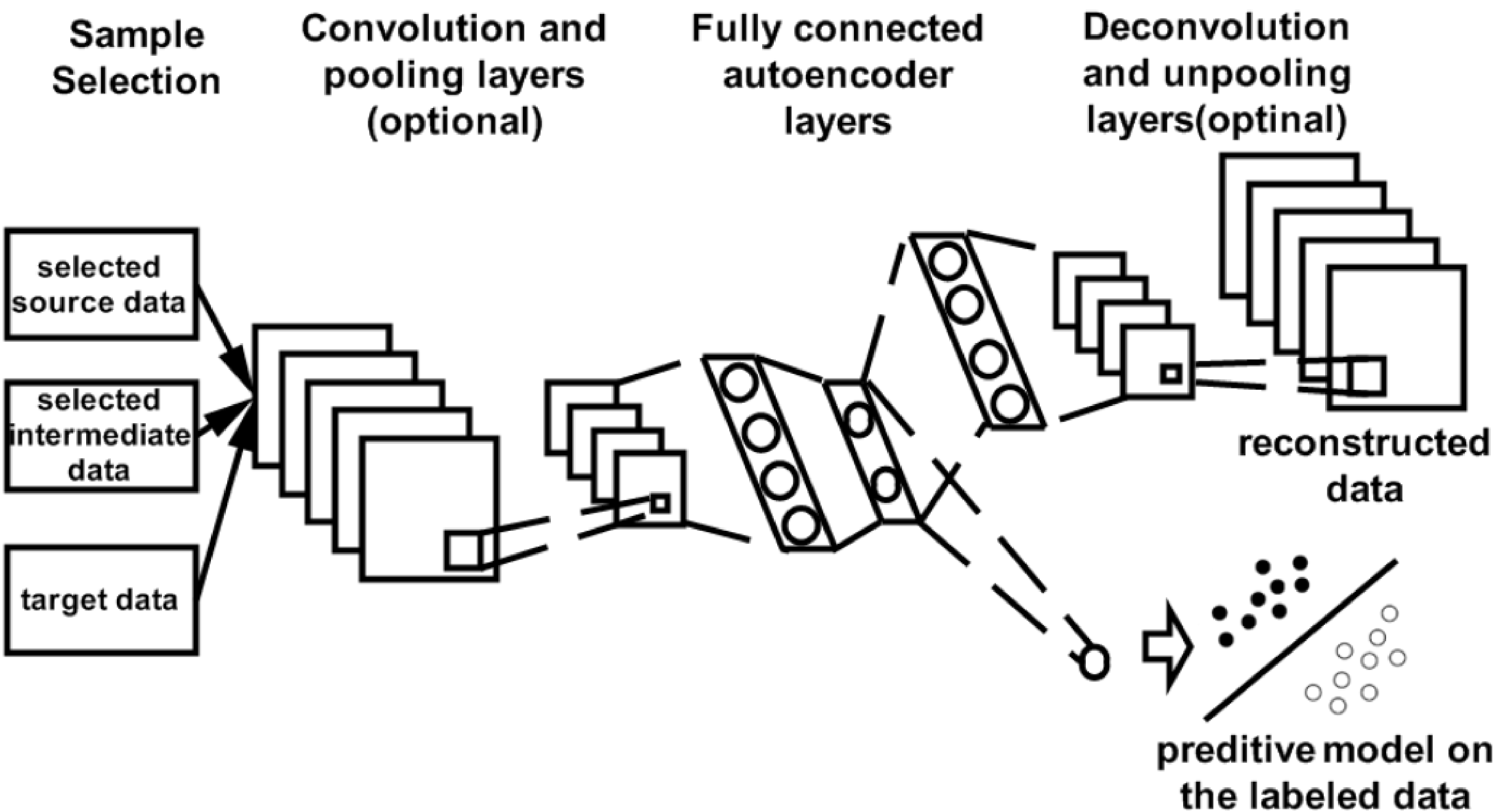
Problem Setting

- Sufficient source domain data $S = \{(x_S^1, y_S^1), ..., (x_S^{n_S}, y_S^{n_S})\}$
- Limited target domain data $T = \{(x_T^1, y_T^1), ..., (x_T^{n_T}, y_T^{n_T})\}$
- Mixture of unlabeled data of multiple intermediate domains $I = \{x_I^1, ..., x_I^{n_I}\}$, n_I is large enough
- Homogeneous: same feature space but different distributions

$$p_T(x) \neq p_S(x)$$
$$p_T(x) \neq p_I(x)$$
$$p_T(y|x) \neq p_S(y|x)$$

Ben Tan and Qiang Yang et al. Distant Domain Transfer Learning. AAAI 2017.

Selective Learning Algorithm



Ben Tan and Qiang Yang et al. Distant Domain Transfer Learning. AAAI 2017.

Selective Learning Algorithm

- Instance selection via reconstruction error by an AE

$$\mathcal{J}_1(f_e, f_d, \mathbf{v}_s, \mathbf{v}_t) = \frac{1}{n_S}\sum_{i=1}^{n_S} v_S^i \left\|\hat{x}_S^i - x_S^i\right\|_2^2 + \frac{1}{n_I}\sum_{i=1}^{n_I} v_I^i \left\|\hat{x}_I^i - x_I^i\right\|_2^2 + \frac{1}{n_T}\sum_{i=1}^{n_T} v_I^i \left\|\hat{x}_T^i - x_T^i\right\|_2^2 + R(\mathbf{v}_s, \mathbf{v}_t)$$

  - selection indicators $v_S^i, v_I^j \in \{0,1\}$
  - regularization term $R(\mathbf{v_s}, \mathbf{v_t}) = -\frac{\lambda_S}{n_S}\sum_{i=1}^{n_S} v_S^i - \frac{\lambda_I}{n_I}\sum_{i=1}^{n_I} v_I^i$

- Incorporation of label information

$$\mathcal{J}_2(f_c, f_e, f_d) = \frac{1}{n_S}\sum_{i=1}^{n_S} v_S^i l(y_S^i, f_c(h_S^i)) + \frac{1}{n_T}\sum_{i=1}^{n_T} v_T^i l(y_T^i, f_c(h_T^i)) + \frac{1}{n_I}\sum_{i=1}^{n_I} v_I^i g(f_c(h_I^i))$$

  - Entropy function $g(z) = -z\log z - (1-z)\log(1-z)$

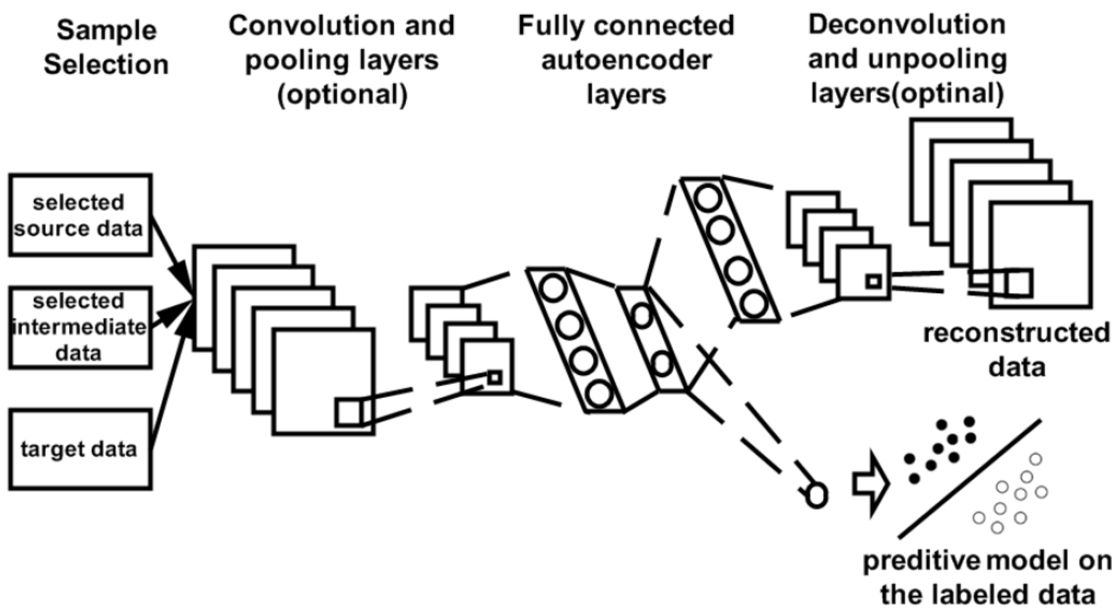
- Overall objective function $\min_{\theta, v} \mathcal{J} = \mathcal{J}_1 + \mathcal{J}_2$

Selective Learning Algorithm

- Update Θ: back propagation
- Update v

$$v_S^i = \begin{cases} 1 & \text{if } \ell(y_s^i, f_c(f_e(\boldsymbol{x}_S^i))) + \|\hat{\boldsymbol{x}}_S^i - \boldsymbol{x}_S^i\|_2^2 < \lambda_S \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$v_I^i = \begin{cases} 1 & \text{if } \|\hat{\boldsymbol{x}}_I^i - \boldsymbol{x}_I^i\|_2^2 + g(f_c(f_e(\boldsymbol{x}_I^i))) < \lambda_I \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

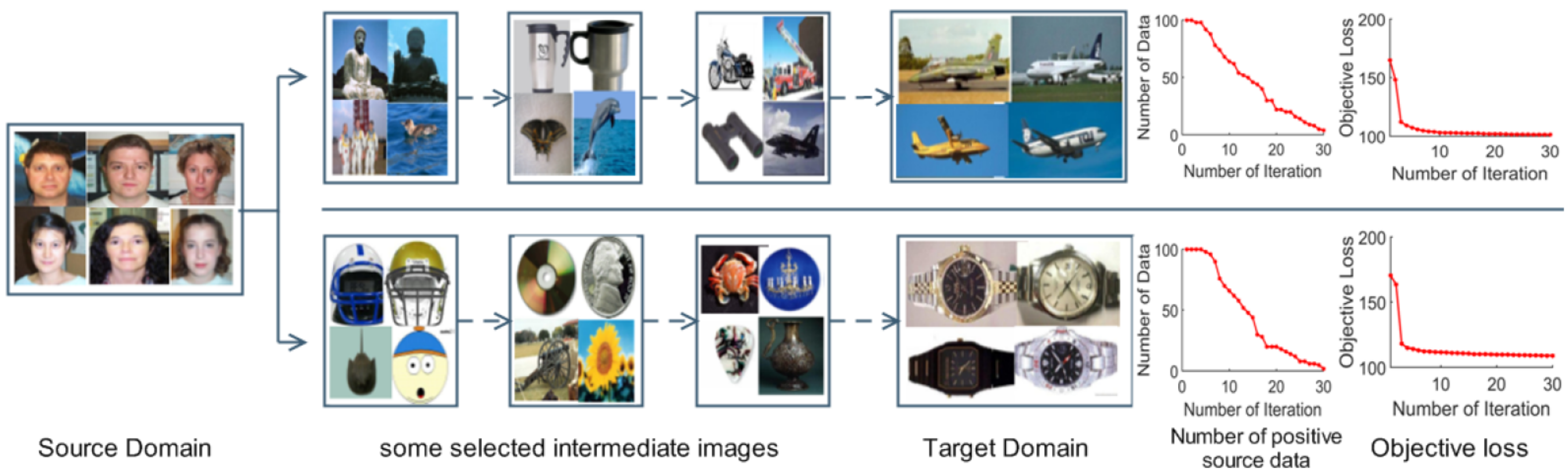


Ben Tan and Qiang Yang et al. Distant Domain Transfer Learning. AAAI 2017.

DDTL by Selective Learning Algorithm

Table 2: Accuracies (%) of selected tasks on Catech-256 and AwA with SIFT features.

	SVM	DTL	GFK	LAN	ASVM	TTL	STL	SLA
‘horse-to-face’	84 ± 2	88 ± 2	77 ± 3	79 ± 2	76 ± 4	78 ± 2	86 ± 3	$\mathbf{92} \pm 2$
‘airplane-to-gorilla’	75 ± 1	62 ± 3	67 ± 5	66 ± 4	51 ± 2	65 ± 2	76 ± 3	$\mathbf{84} \pm 2$
‘face-to-watch’	75 ± 7	68 ± 3	61 ± 4	63 ± 4	60 ± 5	67 ± 4	75 ± 5	$\mathbf{88} \pm 4$
‘zebra-to-collie’	71 ± 3	69 ± 2	56 ± 2	57 ± 3	59 ± 2	70 ± 3	72 ± 3	$\mathbf{76} \pm 2$



Ben Tan and Qiang Yang et al. Distant Domain Transfer Learning. AAAI 2017.

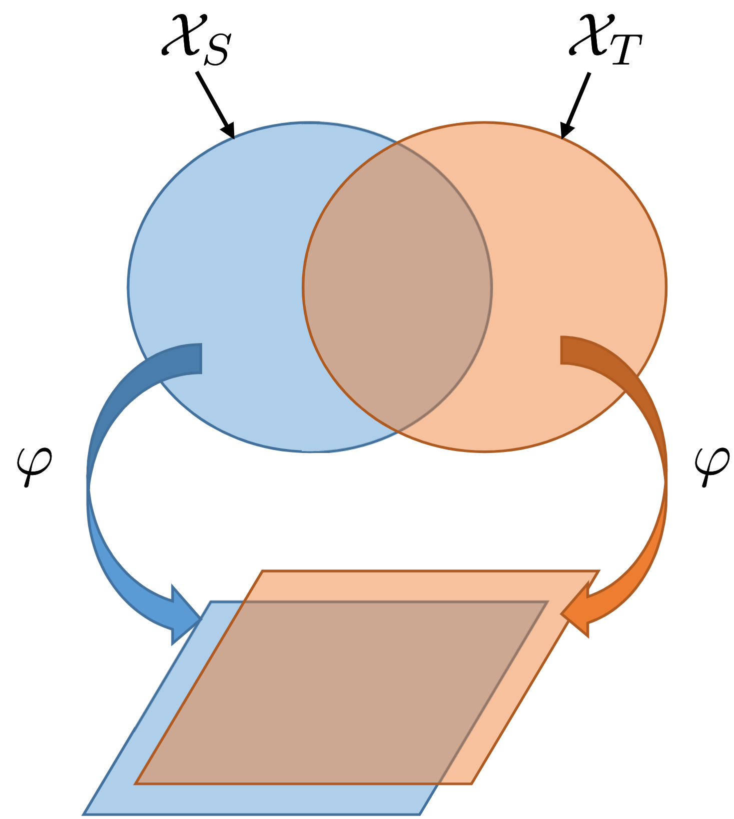
Transfer Learning Methods

- Instance Transfer
  - Reweight instances of target data according to source
- Feature Transfer
  - Mapping features of source and target data in a common space
- Parameter Transfer
  - Learn target model parameters according to source model

Relational approaches are relatively unpopular, thus omitted in this talk

Feature-based Transfer Learning

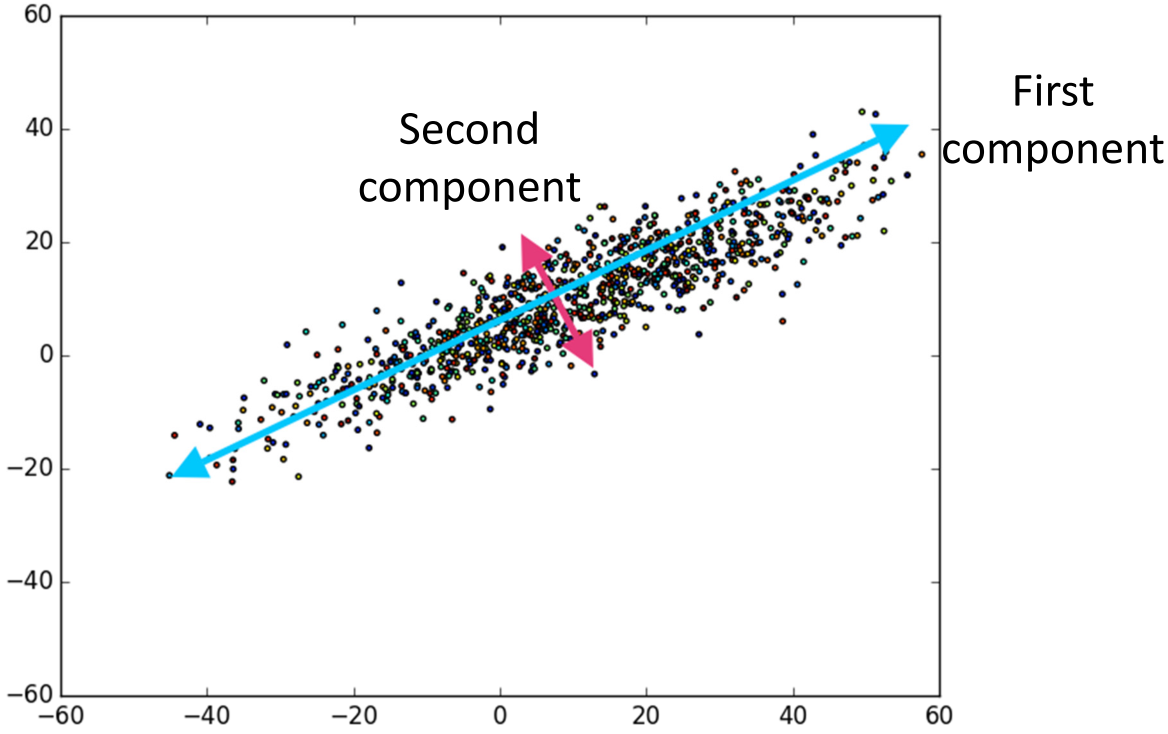
- When source and target domains only have some overlapping features
  - Lots of features only have support in either the source or the target domain
- Possible solutions
  - Encode application-specific knowledge
  - General approaches to learn the transformation φ



General Feature-Based TL Approach

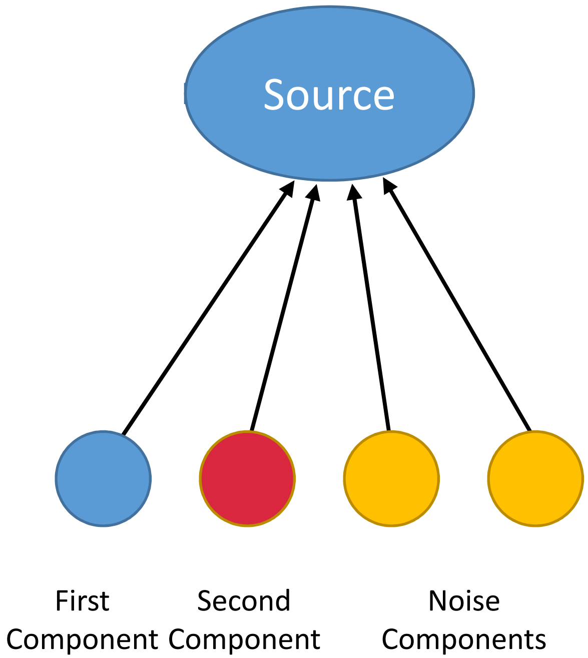
- Learning new data representations by minimizing the distance between two domain distributions
- Learning new data representations by multi-task learning
- Learning new data representations by self-taught learning

Principle Component Analysis (PCA)



- PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components

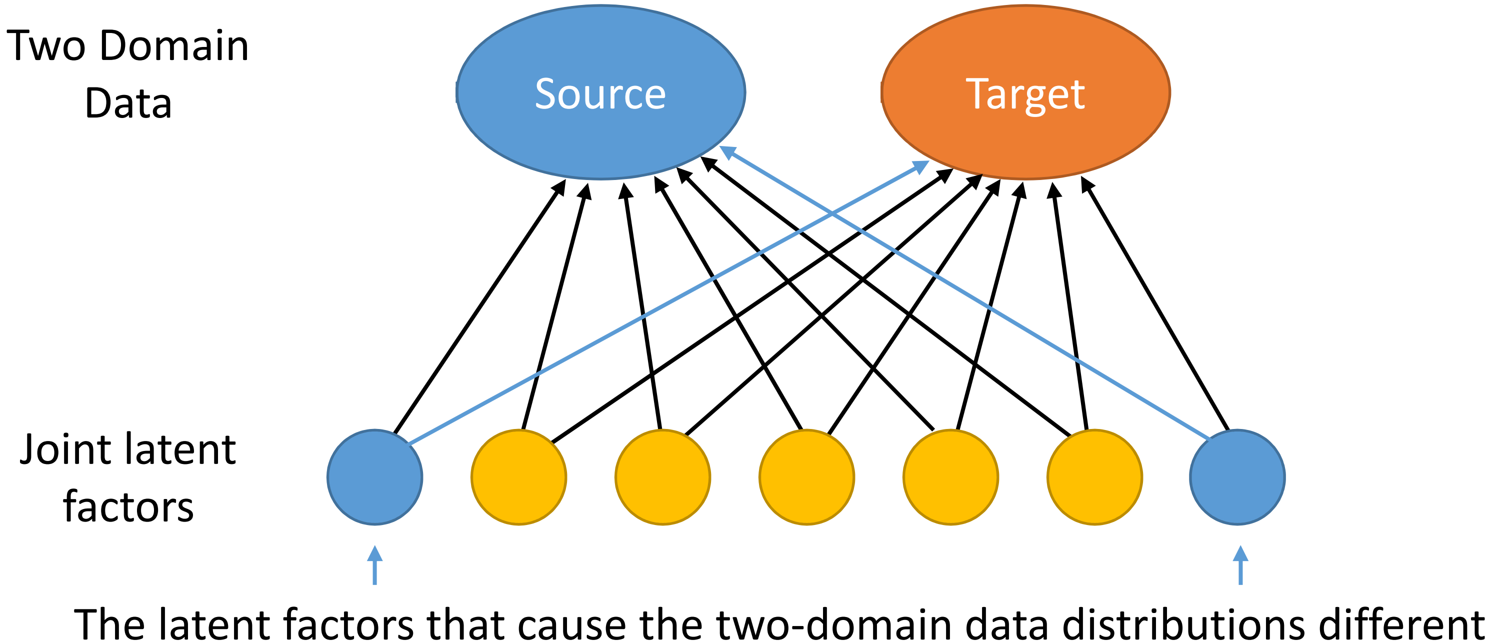
Principle Component Analysis (PCA)



- PCA uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components

Transfer Component Analysis

- Motivation
  - Minimize the distance between domain distributions by projecting data onto the learned transfer components



Pan, Sinno Jialin, et al. "Domain adaptation via transfer component analysis." *IEEE Transactions on Neural Networks* 22.2 (2011): 199-210.

Transfer Component Analysis

- Main idea
  - Learn φ to map the source and target domain data to the latent space spanned by the factors which can reduce domain difference and preserve original data structure

$$\begin{aligned} \min_{\varphi} \quad & \mathrm{Dist}(\varphi(\mathbf{X}_S), \varphi(\mathbf{X}_T)) + \lambda\Omega(\varphi) \\ \text{s.t.} \quad & \text{constraints on } \varphi(\mathbf{X}_S) \text{ and } \varphi(\mathbf{X}_T) \end{aligned}$$

Transfer Component Analysis

- Maximum Mean Discrepancy (MMD)
  - Given the source and target domain data

$$\mathbf{X}_S = \{x_{S_i}\}_{i=1}^{n_S} \qquad \mathbf{X}_T = \{x_{T_i}\}_{i=1}^{n_T}$$

drawn from $P_S(x)$ and $P_T(s)$ respectively

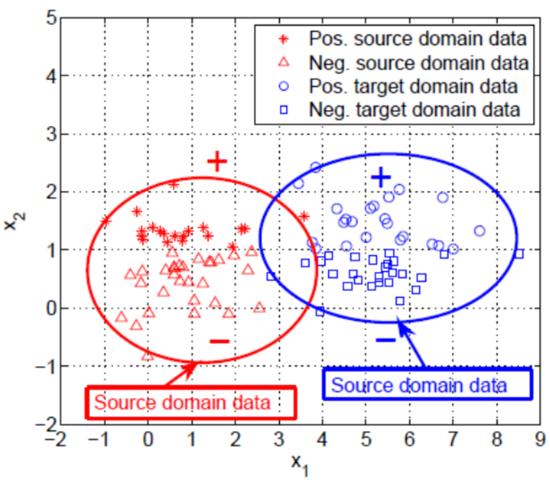
$$\mathrm{Dist}(\varphi(\mathbf{X}_S), \varphi(\mathbf{X}_T)) = \left\| \frac{1}{n_S} \sum_{i=1}^{n_S} \Phi(\varphi(x_{S_i})) - \frac{1}{n_T} \sum_{i=1}^{n_T} \Phi(\varphi(x_{T_i})) \right\|_{\mathcal{H}}$$

Mapping

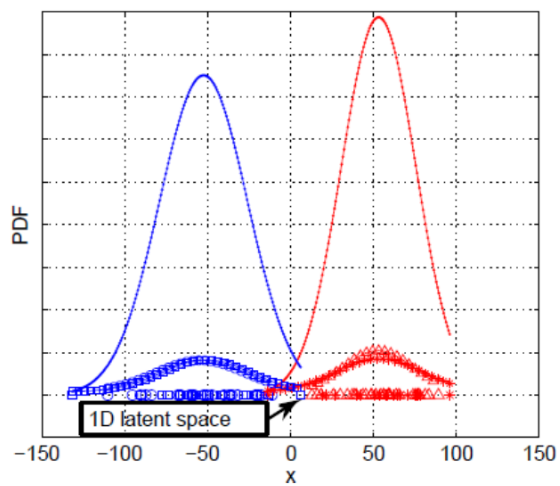
Kernel function

Transfer Component Analysis

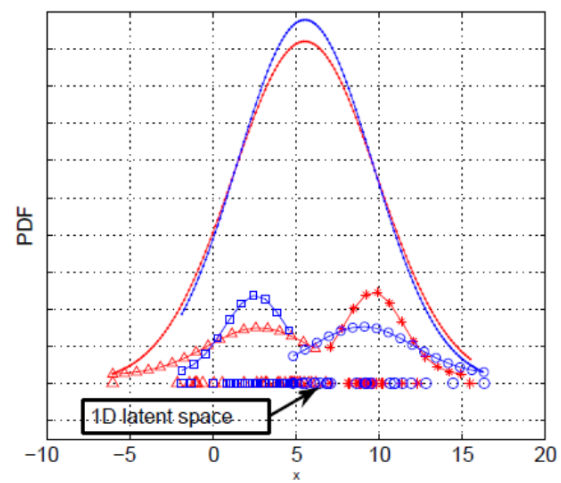
- An illustrative example Latent features learned by PCA and TCA



Original feature space



PCA



TCA

Maximum Mean Discrepancy

Problem 1 *Let x and y be random variables defined on a topological space $\mathcal{X}$, with respective Borel probability measures p and q . Given observations $X := \{x_1, \ldots, x_m\}$ and $Y := \{y_1, \ldots, y_n\}$, independently and identically distributed (i.i.d.) from p and q, respectively, can we decide whether $p \neq q$?*

Lemma 1 *Let $(\mathcal{X}, d)$ be a metric space, and let p, q be two Borel probability measures defined on $\mathcal{X}$. Then $p = q$ if and only if $\mathbf{E}_x(f(x)) = \mathbf{E}_y(f(y))$ for all $f \in C(\mathcal{X})$, where $C(\mathcal{X})$ is the space of bounded continuous functions on $\mathcal{X}$.*

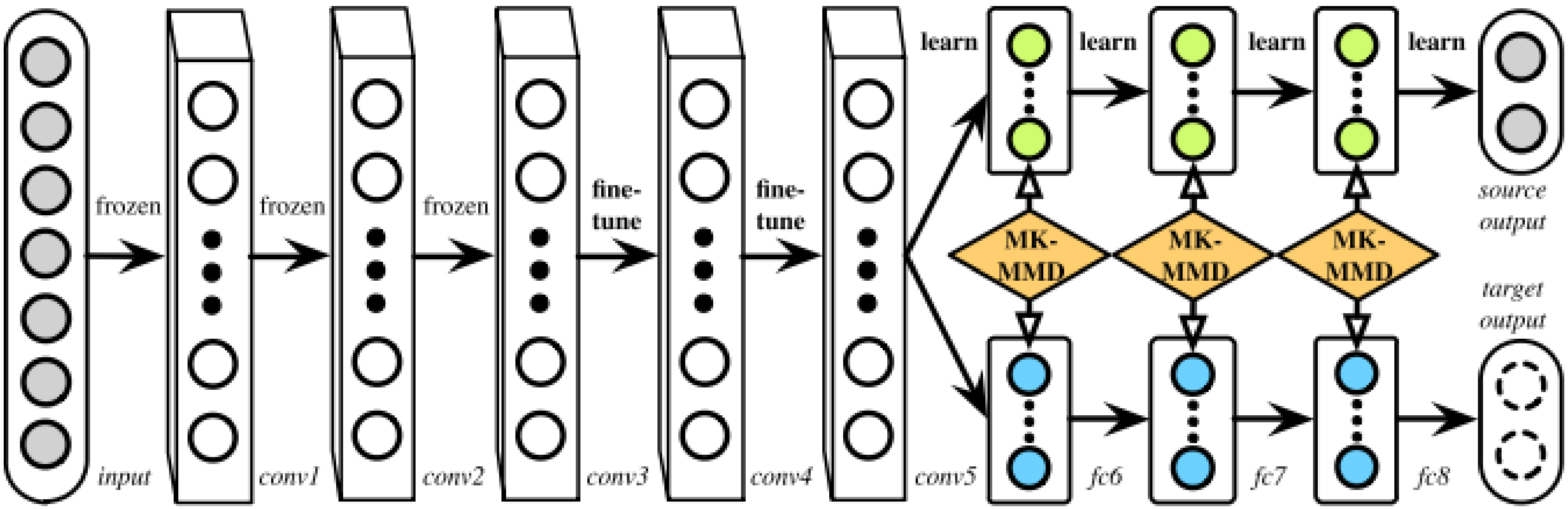
Definition 2 *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ and let p, q, x, y, X, Y be defined as above. We define the maximum mean discrepancy (MMD) as*

$$\mathrm{MMD}[\mathcal{F}, p, q] := \sup_{f \in \mathcal{F}} \left(\mathbf{E}_x[f(x)] - \mathbf{E}_y[f(y)]\right). \tag{1}$$

$$\mathrm{MMD}_b[\mathcal{F}, X, Y] := \sup_{f \in \mathcal{F}} \left(\frac{1}{m}\sum_{i=1}^{m} f(x_i) - \frac{1}{n}\sum_{i=1}^{n} f(y_i)\right). \tag{2}$$

MMD in Transfer Learning

Deep Adaptation Network



- Multi-kernels, e.g., some RBF kernels with different standard deviations

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

Long et al. Learning Transferable Features with Deep Adaptation Networks. ICML 2015.

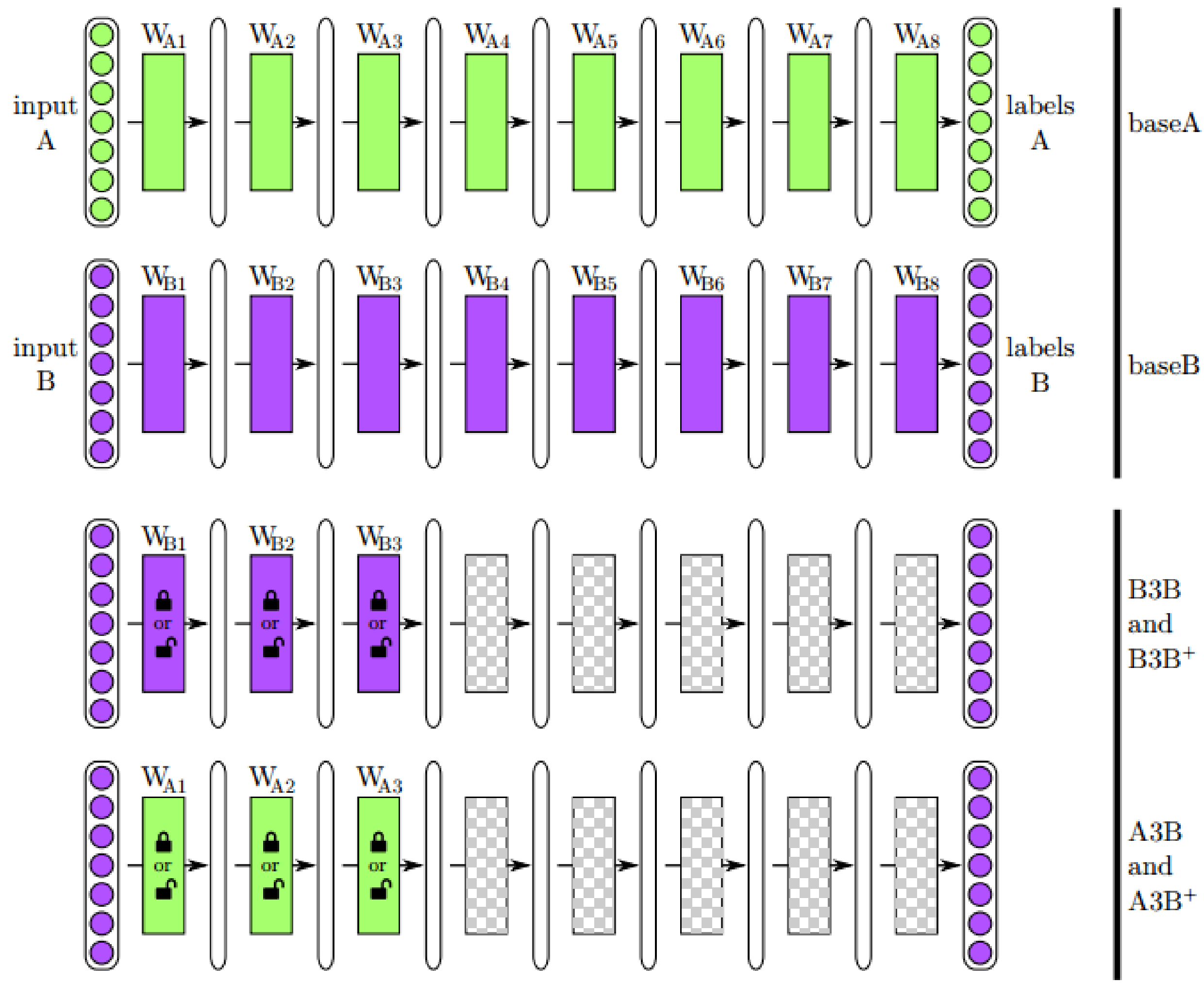
How transferable are features in deep neural networks? [NIPS 2014]

Standard BP training on dataset A

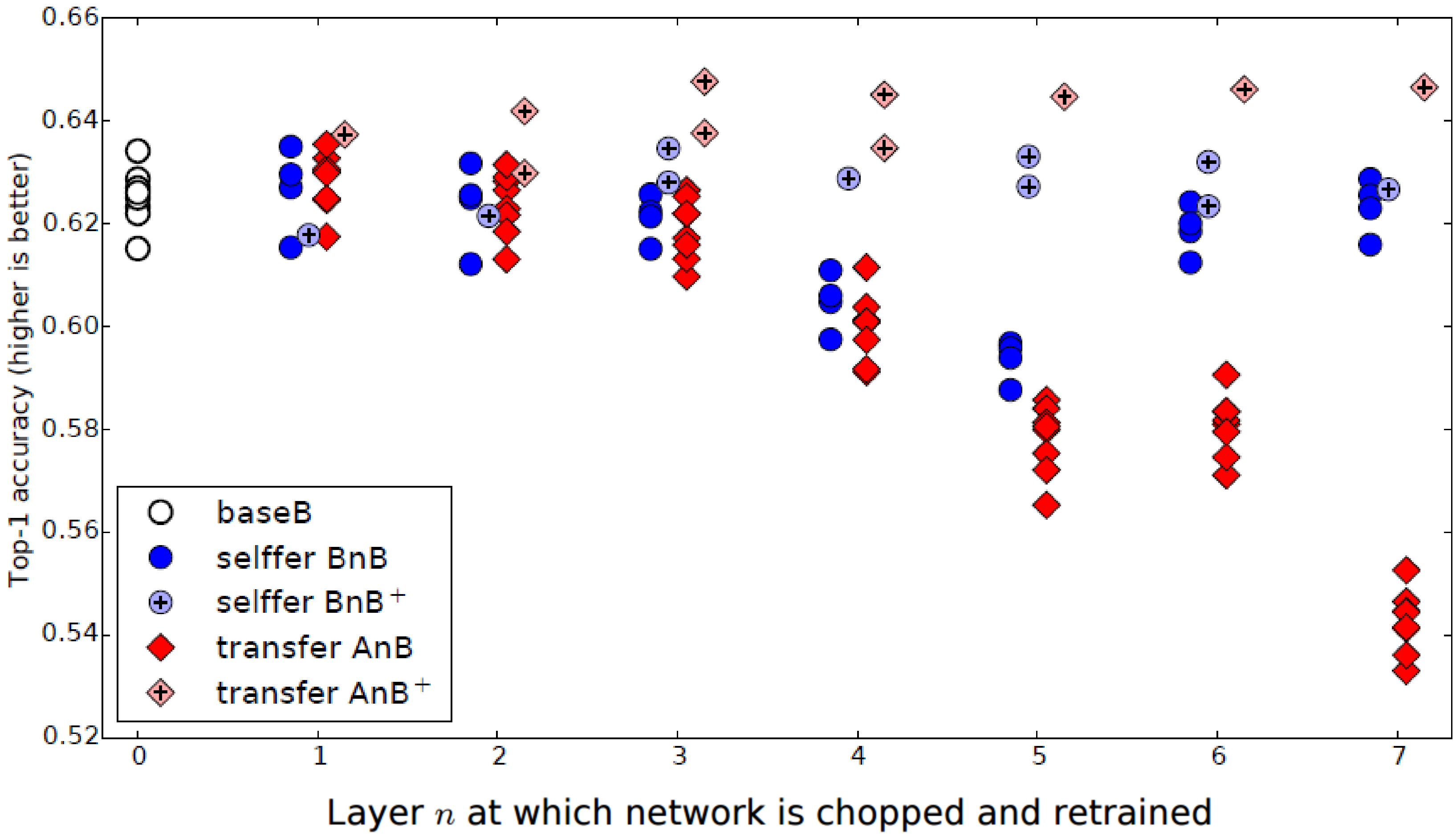
Standard BP training on dataset B

Reuse first n=3 layers weights of baseB and train on B

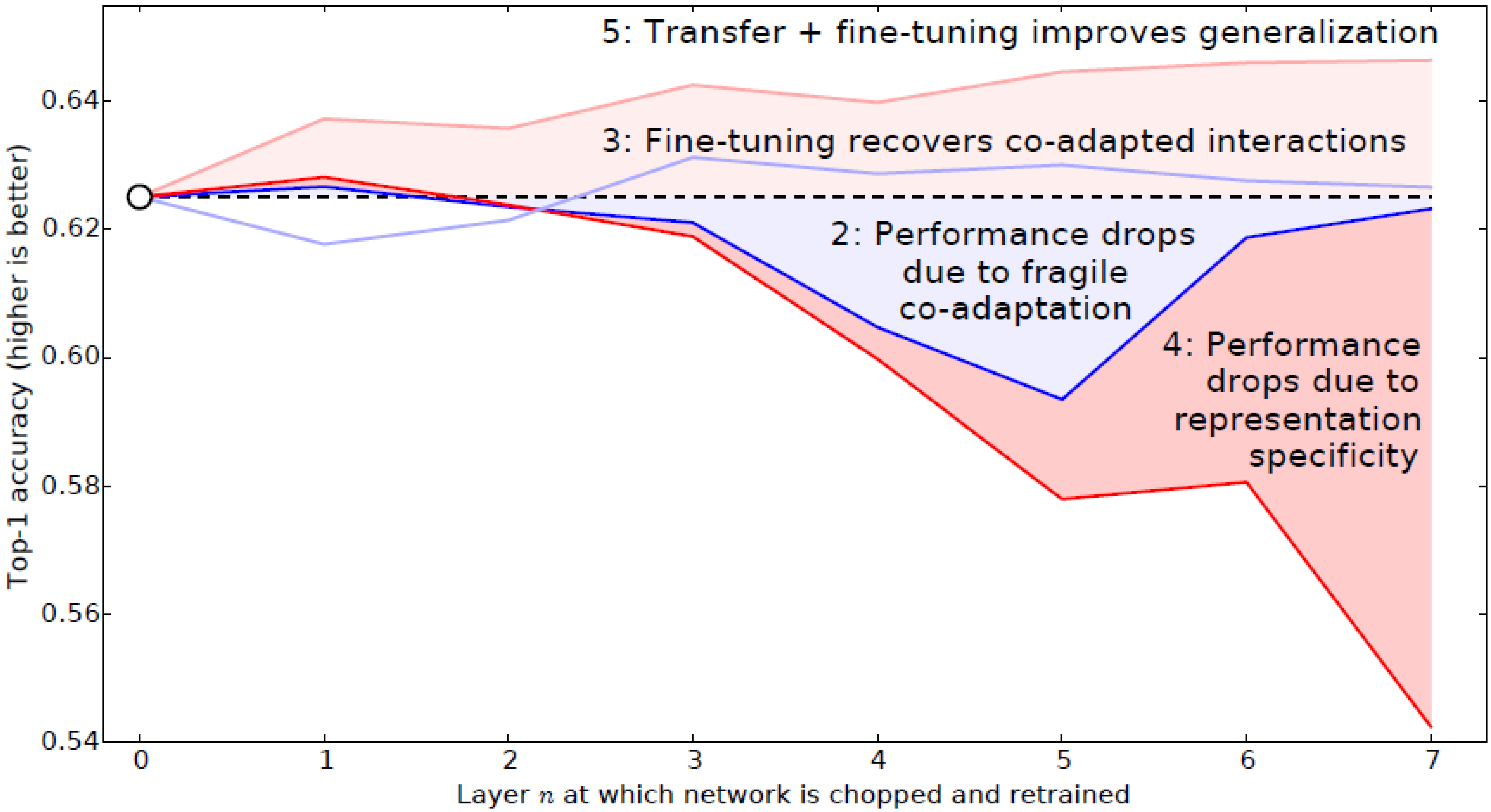
Reuse first n=3 layers weights of baseA and trained on B



How transferable are features in deep neural networks? [NIPS 2014]



How transferable are features in deep neural networks?



Domain Adversarial Neural Network

Definition 1 (Ben-David et al., 2006, 2010; Kifer et al., 2004) *Given two domain distributions $\mathcal{D}_{\mathrm{S}}^{X}$ and $\mathcal{D}_{\mathrm{T}}^{X}$ over X, and a hypothesis class $\mathcal{H}$, the $\mathcal{H}$-*divergence *between $\mathcal{D}_{\mathrm{S}}^{X}$ and $\mathcal{D}_{\mathrm{T}}^{X}$ is*

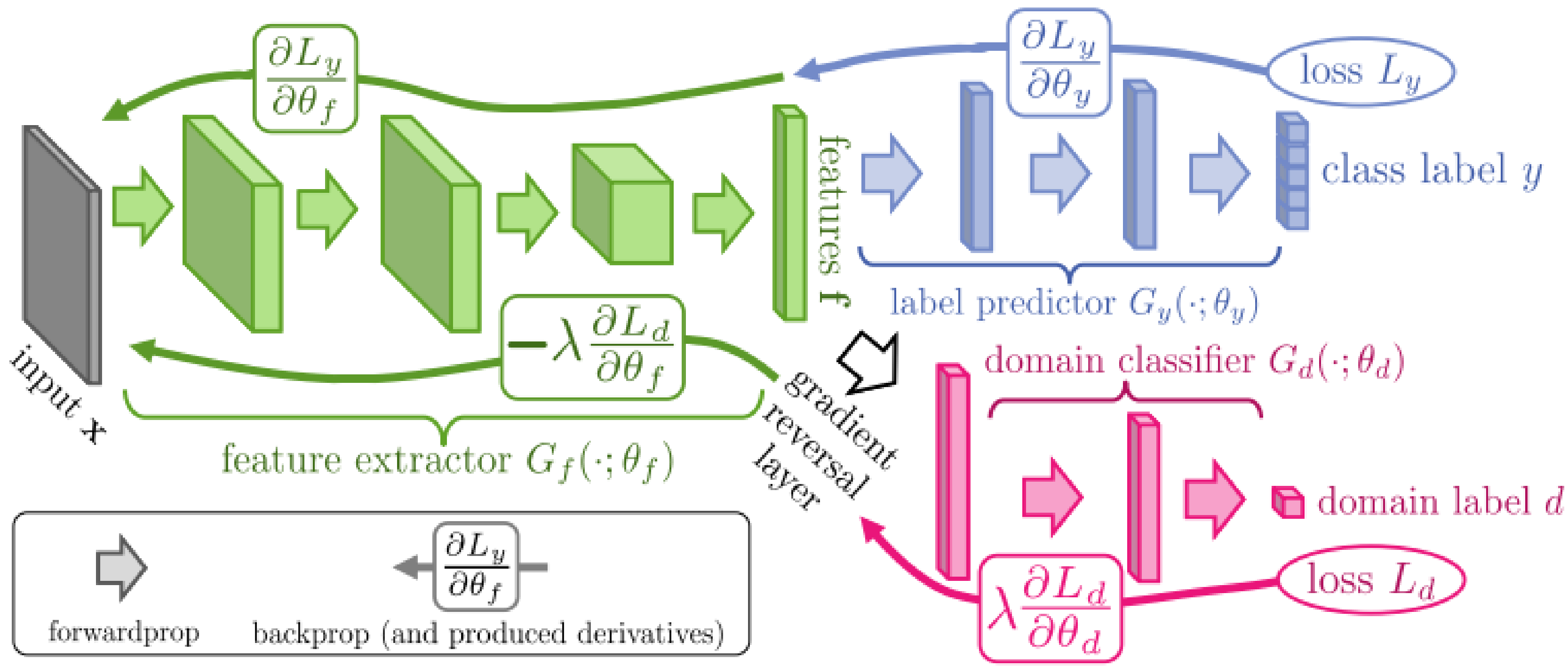
$$d_{\mathcal{H}}(\mathcal{D}_{\mathrm{S}}^{X}, \mathcal{D}_{\mathrm{T}}^{X}) \quad = \quad 2 \sup_{\eta \in \mathcal{H}} \left| \Pr_{\mathbf{x} \sim \mathcal{D}_{\mathrm{S}}^{X}} \left[\eta(\mathbf{x}) = 1\right] - \Pr_{\mathbf{x} \sim \mathcal{D}_{\mathrm{T}}^{X}} \left[\eta(\mathbf{x}) = 1\right] \right| .$$

$$\Pr_{x \sim \mathcal{D}_S^X}[\eta(x) = 1] + \Pr_{x \sim \mathcal{D}_S^X}[\eta(x) = 0] = 1$$

$$\hat{d}_{\mathcal{H}}(S, T) \quad = \quad 2\left(1 - \min_{\eta \in \mathcal{H}} \left[\underbrace{\frac{1}{n} \sum_{i=1}^{n} I[\eta(\mathbf{x}_i) = 0]}_{\text{Source domain}} + \underbrace{\frac{1}{n'} \sum_{i=n+1}^{N} I[\eta(\mathbf{x}_i) = 1]}_{\text{Target domain}}\right]\right) ,$$

Ajakan, Hana, et al. "Domain-adversarial neural networks." JMLR 2016

Domain Adversarial Neural Network



Experiment Result

		Original data			mSDA representation		
SOURCE	TARGET	DANN	NN	SVM	DANN	NN	SVM
BOOKS	DVD	.784	.790	**.799**	.829	.824	**.830**
BOOKS	ELECTRONICS	.733	.747	**.748**	**.804**	.770	.766
BOOKS	KITCHEN	**.779**	.778	.769	**.843**	.842	.821
DVD	BOOKS	.723	.720	**.743**	.825	.823	**.826**
DVD	ELECTRONICS	**.754**	.732	.748	**.809**	.768	.739
DVD	KITCHEN	**.783**	.778	.746	.849	**.853**	.842
ELECTRONICS	BOOKS	**.713**	.709	.705	**.774**	.770	.762
ELECTRONICS	DVD	**.738**	.733	.726	**.781**	.759	.770
ELECTRONICS	KITCHEN	**.854**	**.854**	.847	.881	**.863**	.847
KITCHEN	BOOKS	**.709**	.708	.707	.718	.721	**.769**
KITCHEN	DVD	**.740**	.739	.736	**.789**	**.789**	.788
KITCHEN	ELECTRONICS	**.843**	.841	.842	.856	.850	**.861**
	AVG	0.763	0.761	0.760	0.813	0.803	0.801

Experiment Result

Method	Source / Target	Amazon / Webcam	DSLR / Webcam	Webcam / DSLR
GFK(PLS, PCA) (Gong et al., 2012)		.197	.497	.6631
SA* (Fernando et al., 2013)		.450	.648	.699
DLID (Chopra et al., 2013)		.519	.782	.899
DDC (Tzeng et al., 2014)		.618	.950	.985
DAN (Long and Wang, 2015)		.685	.960	.990
Source only		.642	.961	.978
DANN		**.730**	**.964**	**.992**

Table 3: Accuracy evaluation of different DA approaches on the standard Office (Saenko et al., 2010) data set. All methods (except SA) are evaluated in the "fully-transductive" protocol (some results are reproduced from Long and Wang, 2015). Our method (last row) outperforms competitors setting the new state-of-the-art.

1.6.a: Amazon: Laptop

1.6.b: Amazon: Bottle

1.6.c: Amazon: Phone

1.6.d: DSLR: Laptop

1.6.e: DSLR: Bottle

1.6.f: DSLR: Phone

1.6.g: Webcam: Laptop

1.6.h: Webcam: Bottle

1.6.i: Webcam: Phone

Figure 1.6: Examples from Office dataset

Transfer Learning Methods

- Instance Transfer
  - Reweight instances of target data according to source
- Feature Transfer
  - Mapping features of source and target data in a common space
- Parameter Transfer
  - Learn target model parameters according to source model

Relational approaches are relatively unpopular, thus omitted in this talk

Parameter based Transfer Learning

- The ϑ-parameterized function $f_\vartheta(x)$ learned on two domains

$$\theta_S^* = \arg\min_\theta \sum_{i=1}^{n_S} \mathcal{L}(y_{S_i}, f_\theta(x_{S_i})) + \lambda\Omega(\theta)$$

$$\theta_T^* = \arg\min_\theta \sum_{i=1}^{n_T} \mathcal{L}(y_{T_i}, f_\theta(x_{T_i})) + \lambda\Omega(\theta)$$

- Motivation
  - A well-trained model $f_{\theta_S^*}(x)$ has learned a lot of structure on the source domain.
  - If two tasks are related, this structure can be transferred to learn the model $f_{\theta_T^*}(x)$ on the target domain

Multi-Task or Collective Learning

- Minimize the joint loss on two tasks and the model parameters distance

$$\min_{\theta_S,\theta_T} \alpha \frac{1}{N_S} \sum_{i=1}^{N_S} \mathcal{L}(y_i, f_{\theta_S}(x_i)) + (1-\alpha) \frac{1}{N_T} \sum_{j=1}^{N_T} \mathcal{L}(y_j, f_{\theta_T}(x_j)) + \lambda \Omega(\theta_S, \theta_T)$$

Source task loss   Target task loss   Parameter distance

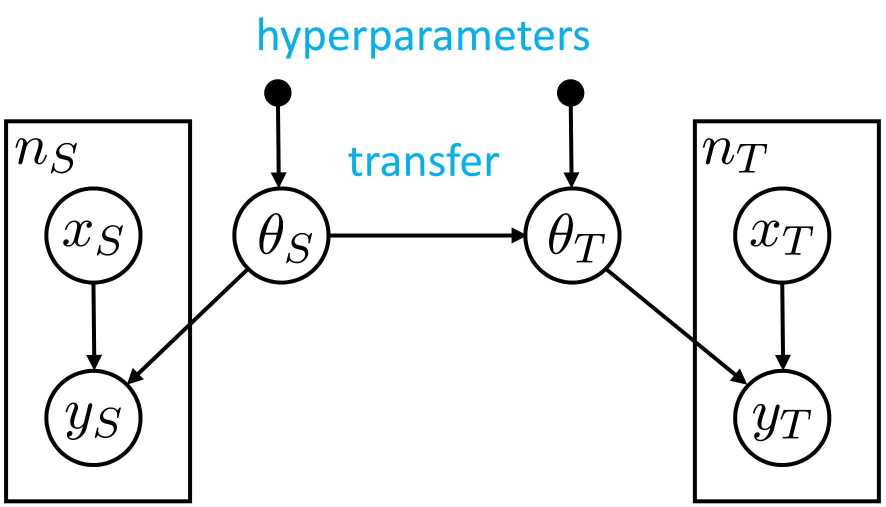
- Different parameter distance definitions

$$\Omega(\theta_S, \theta_T) = \|\theta_S - \theta_T\|^2$$

$$\Omega(\theta_S, \theta_T) = \sum_{t \in \{S,T\}} \|\theta_t - \frac{1}{2} \sum_{s \in \{S,T\}} \theta_s\|^2$$

Hierarchical Bayesian Network

- Idea: source domain parameters, regarded as random variables, act as the prior of the target domain parameters



Case Study: from web browsing to ad click

- Source task
  - Data: user browsed webpage ids
  - Task: predict whether a user likes a webpage
- Target task
  - Data: user browsed webpage ids
  - Task: predict whether a user likes to click an ad

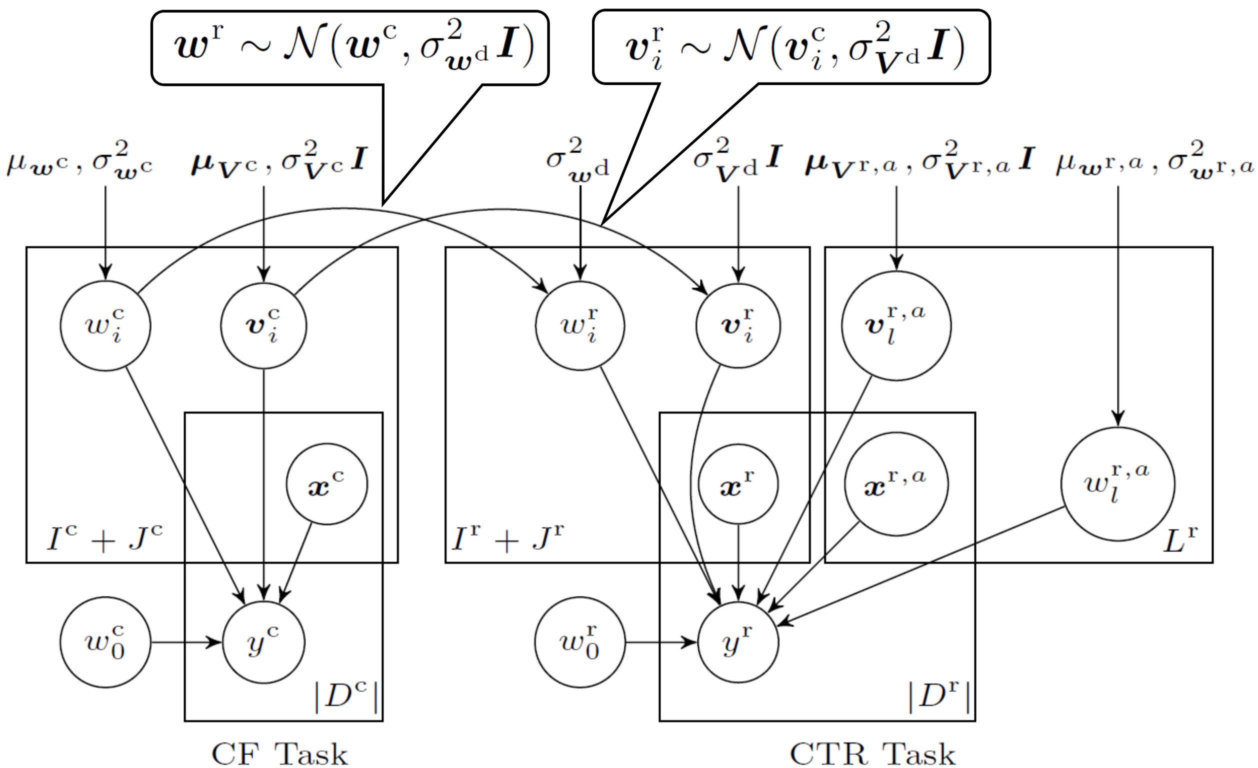
$$\min_{\theta_S,\theta_T} \alpha \frac{1}{N_S} \sum_{i=1}^{N_S} \mathcal{L}(y_i, f_{\theta_S}(x_i)) + (1-\alpha) \frac{1}{N_T} \sum_{j=1}^{N_T} \mathcal{L}(y_j, f_{\theta_T}(x_j)) + \lambda \|\theta_S - \theta_T\|^2$$

Logistic regression　　　　Logistic regression

[Perlich, Claudia, et al. "Machine learning for targeted display advertising: Transfer learning in action." *Machine learning* 95.1 (2014): 103-127.]

Case Study: from web browsing to ad click

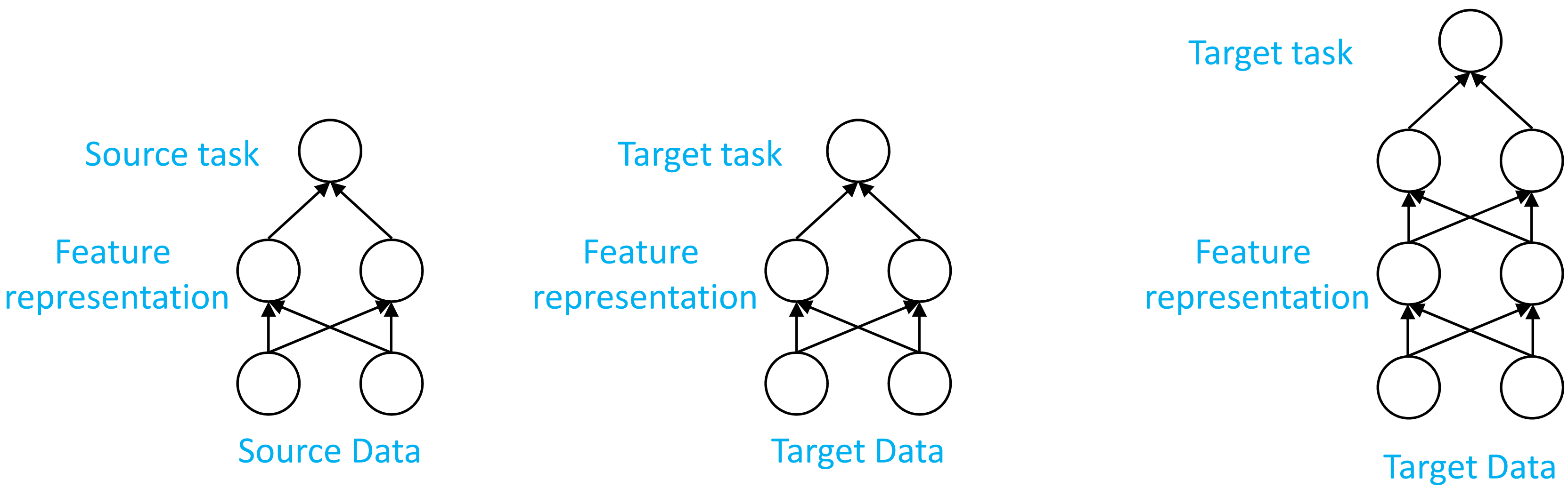
- Illustrated in a hierarchical Bayesian graphical model



[Zhang, Weinan et al. Implicit Look-alike Modelling in Display Ads: Transfer Collaborative Filtering to CTR Estimation. ECIR 2016]

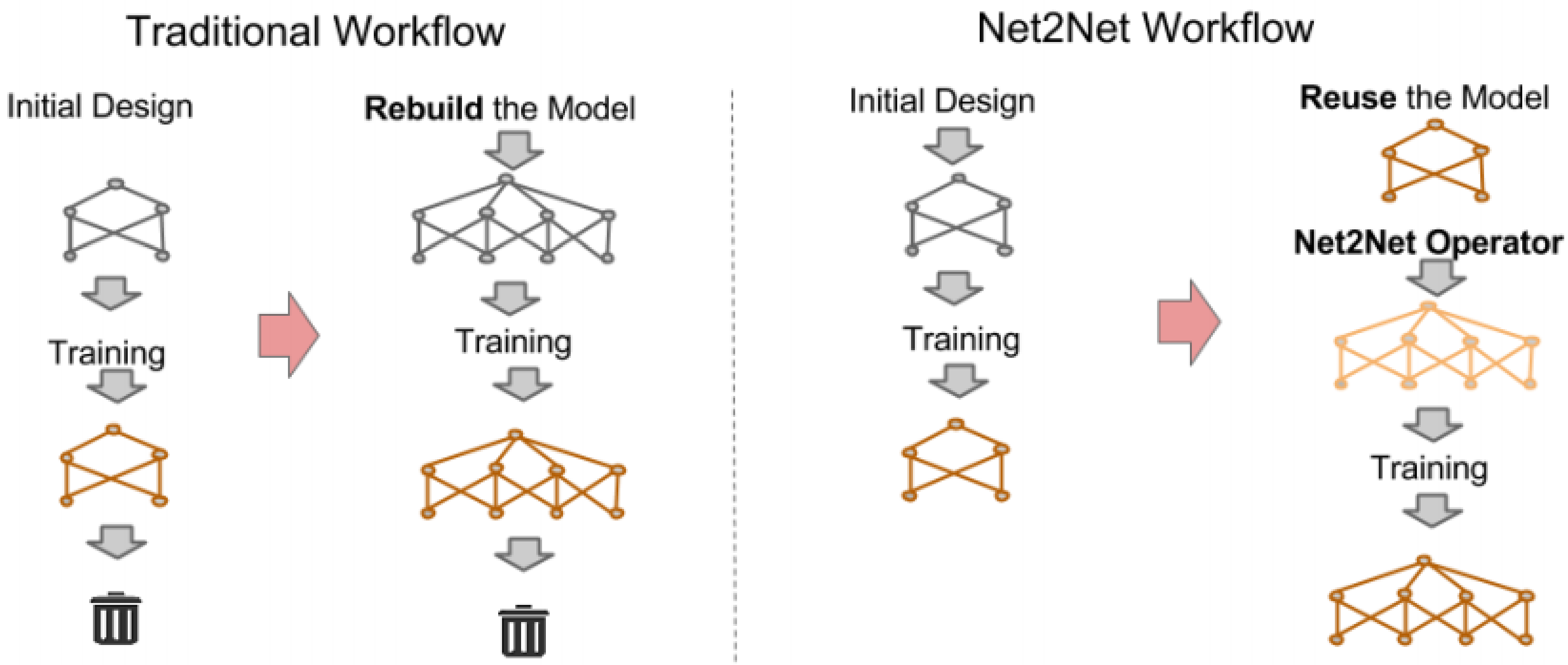
Transfer Learning in Deep Learning

- Mostly, neural network reusing
  - Feed new data for domain adaptation
  - Build higher layers for training another task (feature transfer)



Net2Net Transfer

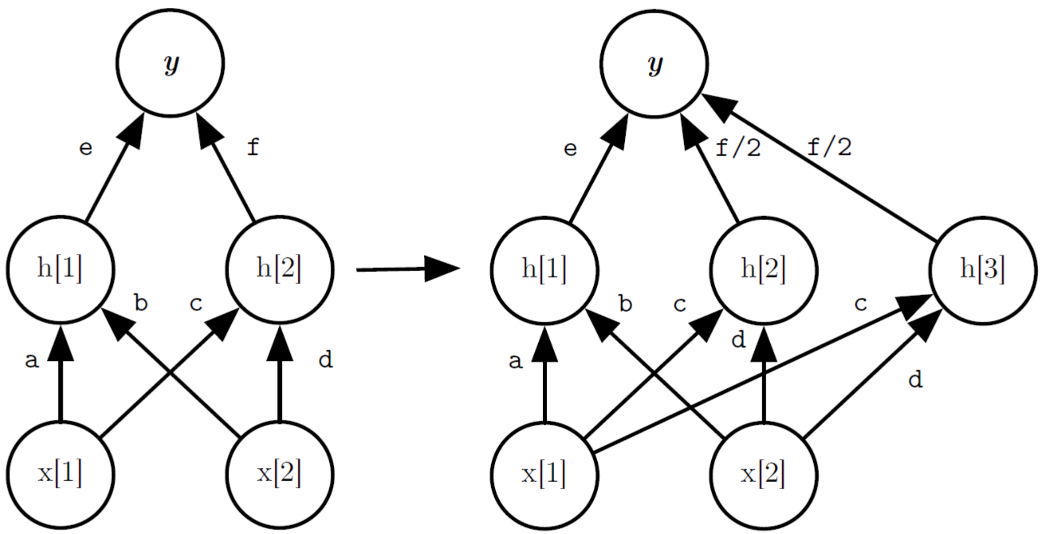
- Net2Net reuses information of already trained model to speedup training of new model



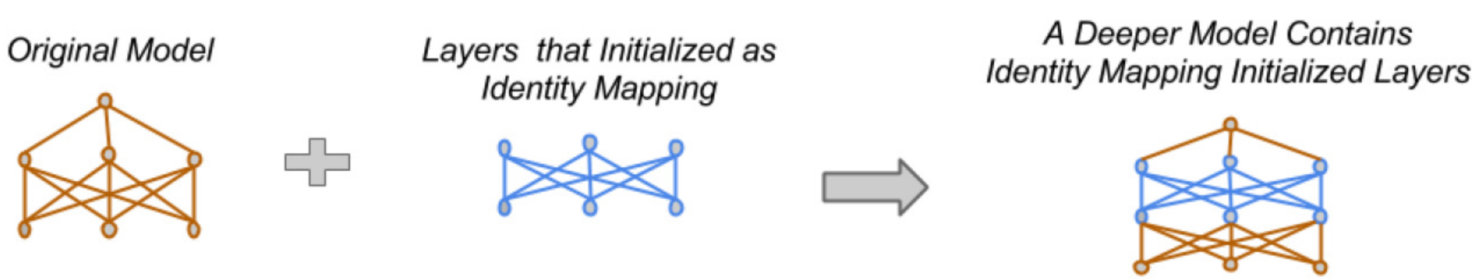
[Chen, Tianqi, Ian Goodfellow, and Jonathon Shlens. "Net2net: Accelerating learning via knowledge transfer.“ ICLR 2016.

Net2Net Transfer: Growing Network

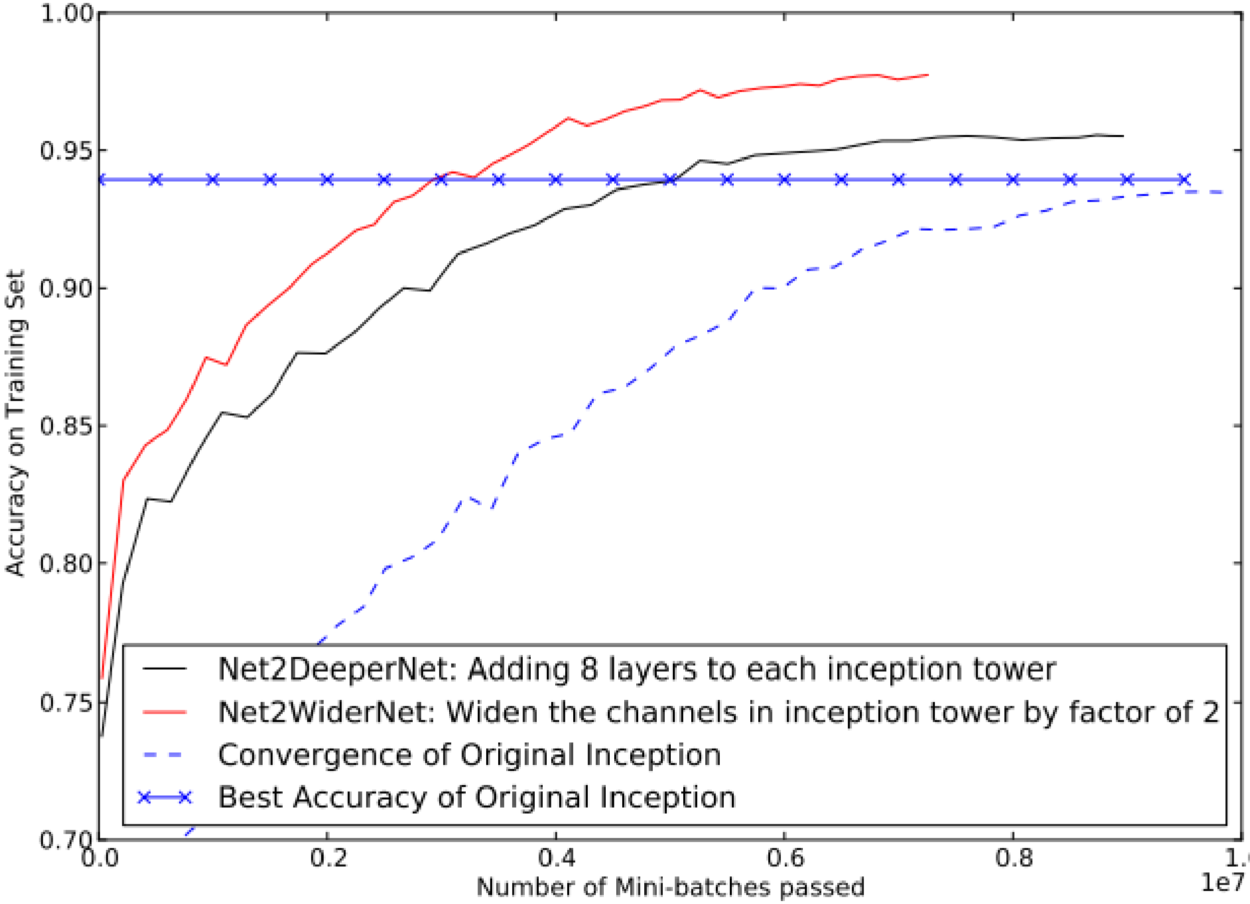
- Wider



- Deeper

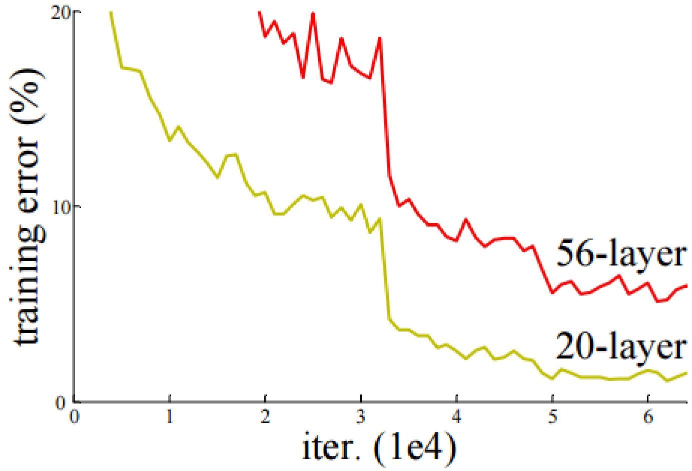


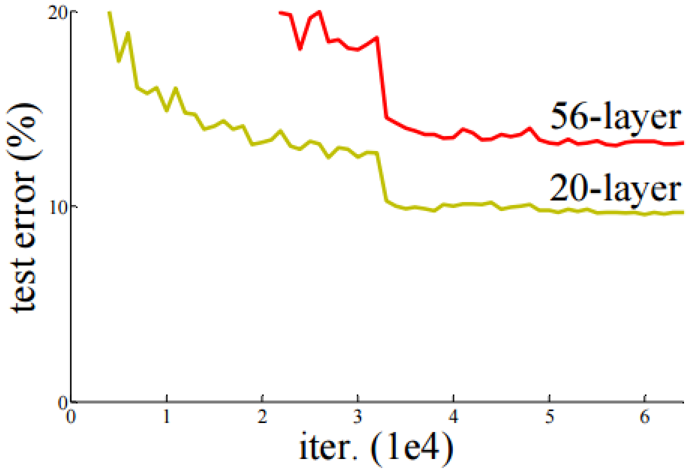
Net2Net over Inception-BN on ImageNet



ResNet: Deep Residual Networks

- Difficulty of training DEEP networks





ResNet: Deep Residual Networks

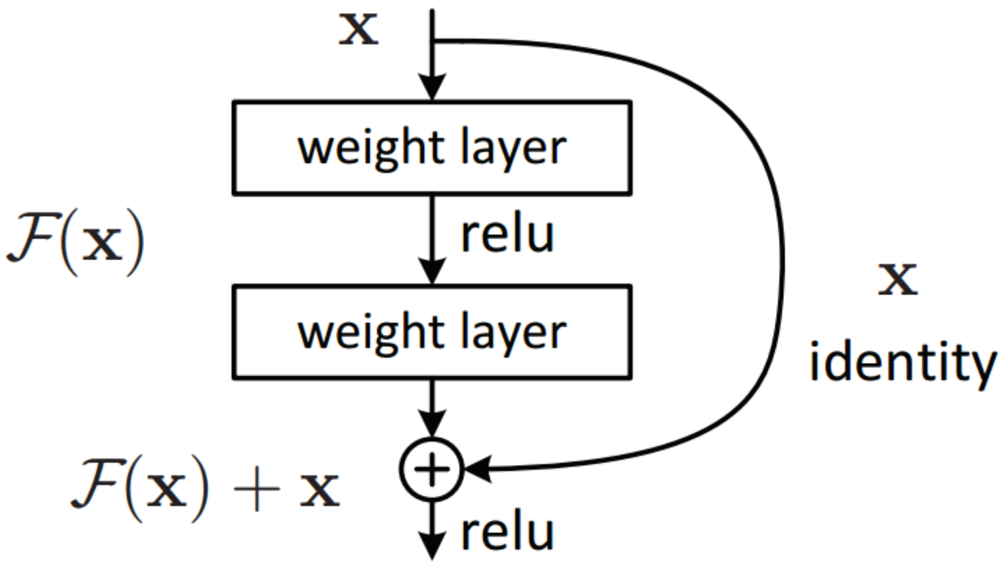
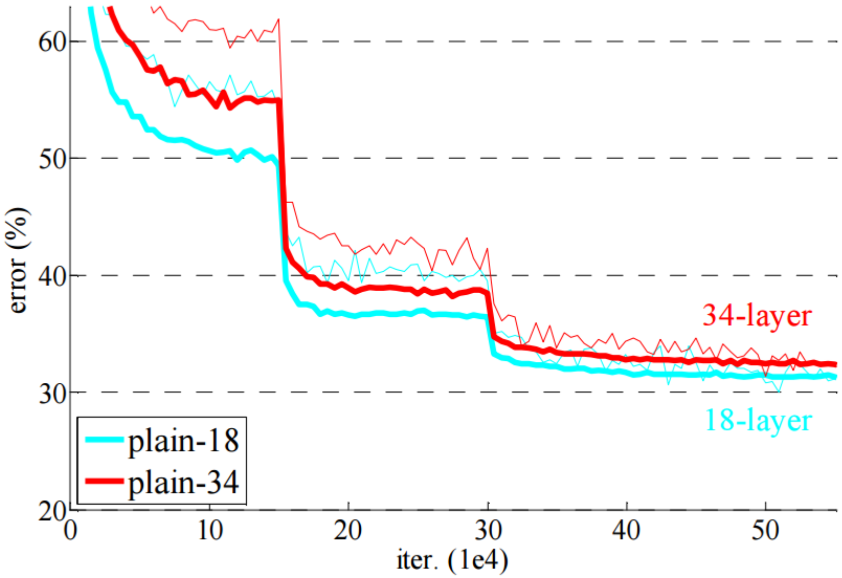
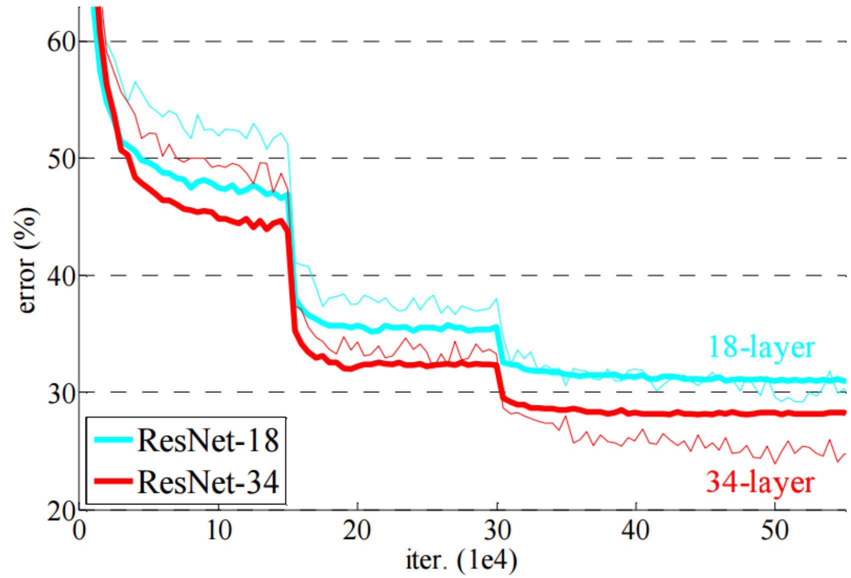


Figure 2. Residual learning: a building block.

Performance on ImageNet



Plain networks of 18 and 34 layers.



ResNets of 18 and 34 layers.

- Thin curves denote training error, and bold curves denote validation error of the center crops.
- The residual networks have no extra parameter compared to their plain counterparts.

Heterogeneous TL

- Different feature space
- Examples
  - Cross-language document classification
  - Cross-system recommendation
- Approaches
  - Symmetric transformation mapping
  - Asymmetric transformation mapping

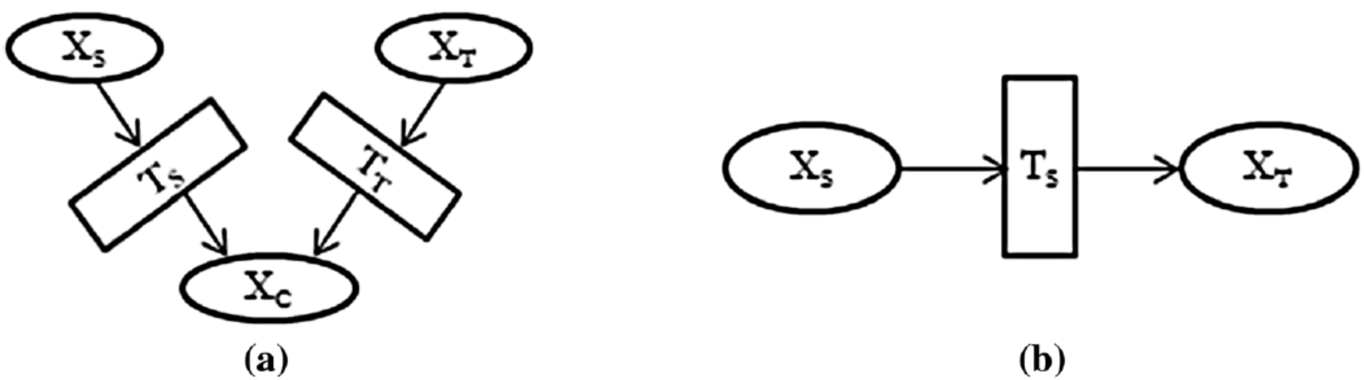


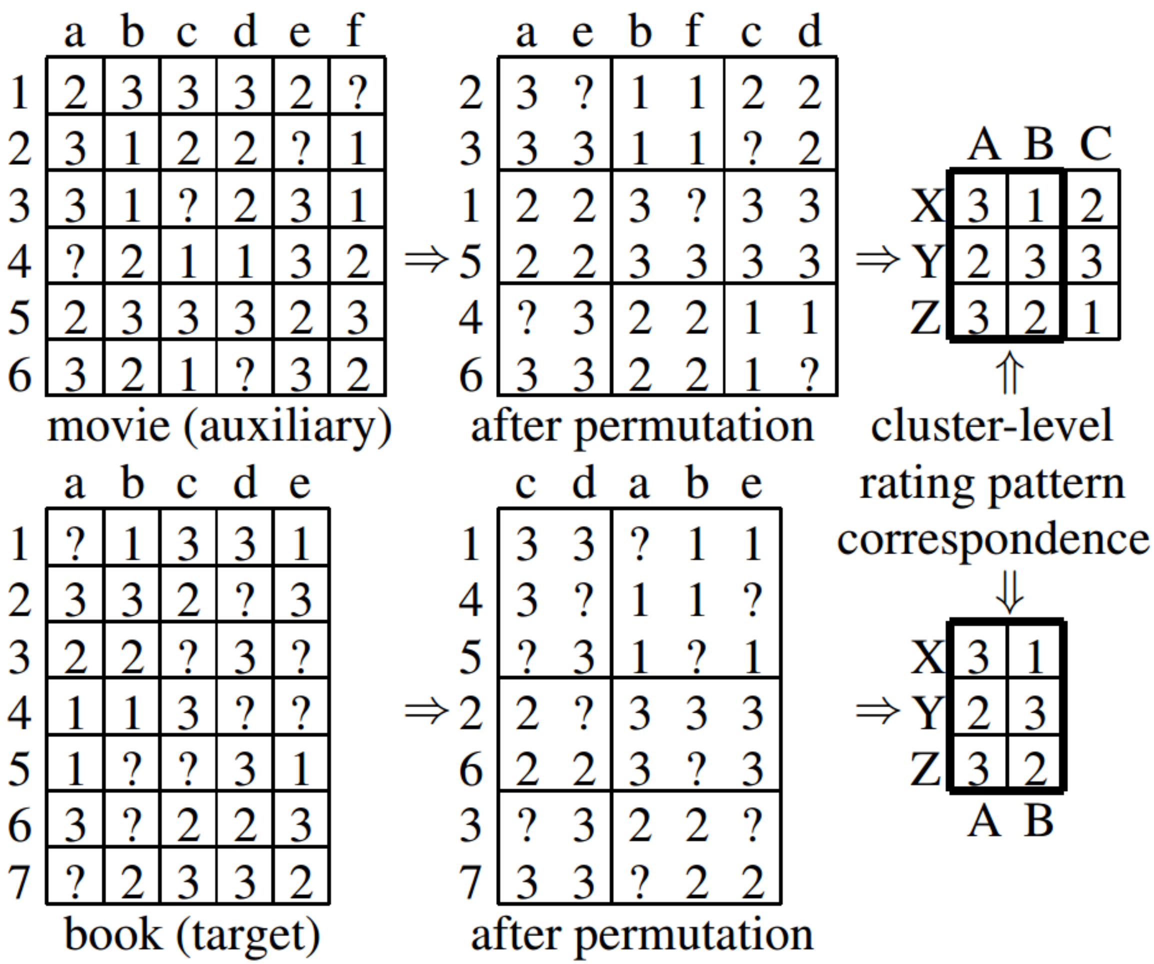
Fig. 1 **a** The symmetric transformation mapping (T_S and T_T) of the source (X_S) and target (X_T) domains into a common latent feature space. **b** The asymmetric transformation (T_T) of the source domain (X_S) to the target domain (X_T)

Cross-system Recommendation

NETFLIX

amazon

Transfer Learning via CodeBook



Li, Bin, Qiang Yang, and Xiangyang Xue. "Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction." *IJCAI*. Vol. 9. 2009.

Transfer Learning via CodeBook

$$\underset{\text{reconstructed}}{\begin{array}{|c|c|c|c|c|}\hline 1^* & 1 & 3 & 3 & 1 \\ \hline 3 & 3 & 2 & 2^* & 3 \\ \hline 2 & 2 & 3^* & 3 & 2^* \\ \hline 1 & 1 & 3 & 3^* & 1^* \\ \hline 1 & 1^* & 3^* & 3 & 1 \\ \hline 3 & 3^* & 2 & 2 & 3 \\ \hline 2^* & 2 & 3 & 3 & 2 \\ \hline \end{array}} \Leftarrow \overset{\mathbf{U}_{tgt}}{\begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}} \times \underset{\text{codebook}}{\begin{array}{c|c|c|c|} & \text{A} & \text{B} & \text{C} \\ \cline{2-4} \text{X} & 3 & 1 & 2 \\ \cline{2-4} \text{Y} & 2 & 3 & 3 \\ \cline{2-4} \text{Z} & 3 & 2 & 1 \\ \cline{2-4} \end{array}} \times \overset{\mathbf{V}_{tgt}^{\top}}{\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 1 & 1 & 0 \\ \hline 1 & 1 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}}$$

Table 1: MAE on MovieLens (average over 10 splits)

Training Set	Method	Given5	Given10	Given15
ML100	PCC	0.930	0.883	0.873
	CBS	0.874	0.845	0.839
	WLR	0.915	0.875	0.890
	CBT	**0.840**	**0.802**	**0.786**
ML200	PCC	0.905	0.878	0.878
	CBS	0.871	0.833	0.828
	WLR	0.941	0.903	0.883
	CBT	**0.839**	**0.800**	**0.784**
ML300	PCC	0.897	0.882	0.885
	CBS	0.870	0.834	0.819
	WLR	1.018	0.962	0.938
	CBT	**0.840**	**0.801**	**0.785**

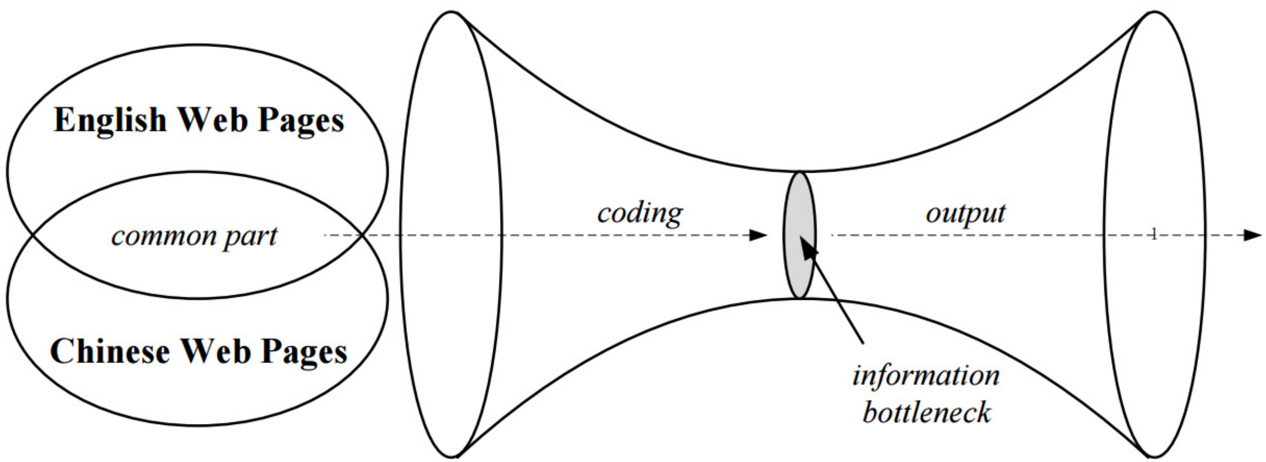
Table 2: MAE on Book-Crossing (average over 10 splits)

Training Set	Method	Given5	Given10	Given15
BX100	PCC	0.677	0.710	0.693
	CBS	0.664	0.655	0.641
	WLR	1.170	1.182	1.174
	CBT	**0.614**	**0.611**	**0.593**
BX200	PCC	0.687	0.719	0.695
	CBS	0.661	0.644	0.630
	WLR	0.965	1.024	0.991
	CBT	**0.614**	**0.600**	**0.581**
BX300	PCC	0.688	0.712	0.682
	CBS	0.659	0.655	0.633
	WLR	0.842	0.837	0.829
	CBT	**0.605**	**0.592**	**0.574**

Li, Bin, Qiang Yang, and Xiangyang Xue. "Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction." *IJCAI*. Vol. 9. 2009.

Cross-Language Text Classification

- A large number of labeled English webpages
- A small number of labeled Chinese webpages
- Solution: information bottleneck

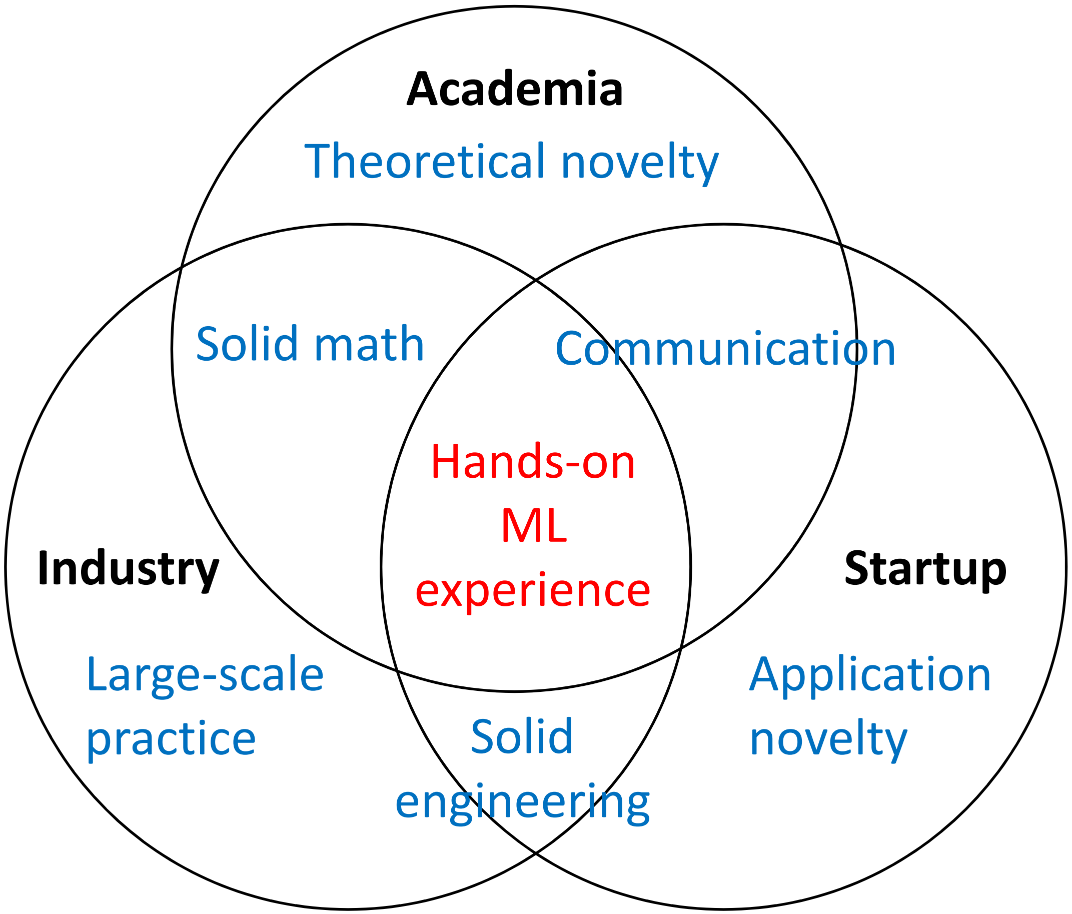


Ling, Xiao, et al. "Can chinese web pages be classified with english data source?." *WWW* 2008.

Summary of CS420

1. ML Introduction
2. Linear Models
3. SVMs and Kernels
4. Neural Networks
5. Tree Models
6. Ensemble Models
7. Collaborative Filtering
8. Graphic Models
9. Unsupervised Learning
10. Model Selection
11. RL Introduction
12. Model-free RL
13. Transfer Learning
14. Poster Session

Summary of CS420



- Play with the data and get your hands dirty!

APPENDIX

RKHS

- MMD function class $\mathcal{F}$: the unit ball in RKHS
- Hilbert Space
  - $given\ k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \exists\ \mathcal{H}\ and\ \phi : \mathcal{X} \to \mathcal{H}$

    $$k(x, x') = < \phi(x), \phi(x') >_{\mathcal{H}}, \forall x, x' \in \mathcal{X}$$
  - k: kernel function
- Reproducing Kernel Hilbert Space
  - $f \in \mathcal{H} : \mathcal{X} \to \mathbb{R}; \phi : \mathcal{X} \to \mathcal{H}$
  - If $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfies
    - (1) $\forall x \in \mathcal{X}, k(\cdot, x) \in \mathcal{H}$
    - (2) $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, f(x) = < f, k(\cdot, x) >_{\mathcal{H}}$
  - k: reproducing kernel
- Define $\phi(x) = k(x, \cdot)$

  $$k(x, x') = < k(\cdot, x'), k(\cdot, x) >_{\mathcal{H}} = < \phi(x'), \phi(x) >_{\mathcal{H}}$$

Transfer Component Analysis

$$\begin{aligned}\text{Dist}(\varphi(\mathbf{X}_S), \varphi(\mathbf{X}_T)) &= \left\| \mathbb{E}_{x\sim P_T(x)}[\Phi(\varphi(x))] - \mathbb{E}_{x\sim P_S(x)}[\Phi(\varphi(x))] \right\| \\ &\approx \left\| \frac{1}{n_S}\sum_{i=1}^{n_S} \Phi(\varphi(x_{S_i})) - \frac{1}{n_T}\sum_{i=1}^{n_T} \Phi(\varphi(x_{T_i})) \right\|\end{aligned}$$

Assume $\Psi = \Phi \circ \varphi$ a RKHS, with kernel $k(x_i, x_j) = \Psi(x_i)^\top \Psi(x_j)$

$$\text{Dist}(\varphi(\mathbf{X}_S), \varphi(\mathbf{X}_T)) = \text{tr}(KL)$$

$$K = \begin{bmatrix} K_{S,S} & K_{S,T} \\ K_{T,S} & K_{T,T} \end{bmatrix} \in \mathbb{R}^{(n_S+n_T)\times(n_S+n_T)}, L_{ij} = \begin{cases} \frac{1}{n_S^2} & x_i, x_j \in X_S, \\ \frac{1}{n_T^2} & x_i, x_j \in X_T, \\ -\frac{1}{n_S n_T} & \text{otherwise.} \end{cases}$$

Transfer Component Analysis

$K = \widetilde{K}WW^\top\widetilde{K}$ where $W \in \mathbb{R}^{(n_S+n_T)\times m}$ and $m \ll n_S + n_T$.

Parametric kernel

Learning K $\Rightarrow$ learning a low-rank matrix W

Regularization term

Minimize distance between domains

$$\min_W \ \mathrm{tr}(W^\top \widetilde{K}L\widetilde{K}W) + \lambda \mathrm{tr}(W^\top W)$$

$$\text{s.t.} \ W^\top \widetilde{K}H\widetilde{K}W = I$$

Maximize data variance

$W^* \Leftrightarrow$ m leading eigenvectors of $(\widetilde{K}L\widetilde{K} + \lambda I)^{-1}\widetilde{K}H\widetilde{K}$

MMD in RKHS

- MMD function class $\mathcal{F}$: the unit ball in RKHS
- Let $\mu_p = \mathbb{E}_{x\sim p}[k(x,\cdot)]$, called mean embedding
- $\mathbb{E}_p[f(x)] = \mathbb{E}_p[< k(x,\cdot), f >_{\mathcal{H}}] =< \mu_p, f >_{\mathcal{H}}$

$$
\begin{aligned}
\mathrm{MMD}^2[\mathcal{F},p,q] &= \left[\sup_{\|f\|_{\mathcal{H}}\le 1} \left(\mathbf{E}_x[f(x)] - \mathbf{E}_y[f(y)]\right)\right]^2 \\
&= \left[\sup_{\|f\|_{\mathcal{H}}\le 1} \langle \mu_p - \mu_q, f\rangle_{\mathcal{H}}\right]^2 \\
&= \|\mu_p - \mu_q\|^2_{\mathcal{H}}.
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{MMD}^2[\mathcal{F},p,q] &= \|\mu_p - \mu_q\|^2_{\mathcal{H}} \\
&= \langle \mu_p, \mu_p\rangle_{\mathcal{H}} + \langle \mu_q, \mu_q\rangle_{\mathcal{H}} - 2\langle \mu_p, \mu_q\rangle_{\mathcal{H}} \\
&= \mathbf{E}_{x,x'}\langle \phi(x), \phi(x')\rangle_{\mathcal{H}} + \mathbf{E}_{y,y'}\langle \phi(y), \phi(y')\rangle_{\mathcal{H}} - 2\mathbf{E}_{x,y}\langle \phi(x), \phi(y)\rangle_{\mathcal{H}},
\end{aligned}
$$

$$
\mathrm{MMD}^2[\mathcal{F},p,q] = \mathbf{E}_{x,x'}\left[k(x,x')\right] - 2\mathbf{E}_{x,y}[k(x,y)] + \mathbf{E}_{y,y'}\left[k(y,y')\right],
$$