Mind the class weight bias: weighted maximum mean discrepancy for unsupervised domain adaptation

Hongliang Yan[1], Yukang Ding[1], Peihua Li[2], Qilong Wang[2], Yong Xu[3], Wangmeng Zuo[1,*]

Hongliang Yan
2017/06/21

Domain Adaptation

Problem:

Training and test sets are related but under different distributions.

Methodology:

- Learn feature space that combine *discriminativeness* and *domain invariance*.

minimize ***source error*** + ***domain discrepancy***

Figure 1. Illustration of dataset bias.

[1]https://cs.stanford.edu/~jhoffman/domainadapt/

Maximum Mean Discrepancy (MMD)

- representing distances between distributions as distances between mean embeddings of features

$$\mathrm{MMD}^2(s,t) = \sup_{\|\phi\|_H \le 1} \| E_{\mathrm{x}^s \sim s}[\phi(\mathrm{x}^s)] - E_{\mathrm{x}^t \sim t}[\phi(\mathrm{x}^t)] \|_H^2$$

- An empirical estimate

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M} \sum_{i=1}^{M} \phi(\mathrm{x}_i^s) - \frac{1}{N} \sum_{j=1}^{t} \phi(\mathrm{x}_j^t) \|_H^2$$

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M} \sum_{i=1}^{M} \phi(\mathrm{x}_i^s) - \frac{1}{N} \sum_{j=1}^{t} \phi(\mathrm{x}_j^t) \|_H^2$$

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t)\|_H^2$$

$$\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t))\|_H^2$$

$$w_c^s = M_c / M \;\text{ and }\; w_c^t = N_c / N$$

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M} \sum_{i=1}^{M} \phi(\mathrm{x}_i^s) - \frac{1}{N} \sum_{j=1}^{t} \phi(\mathrm{x}_j^t) \|_H^2$$

$$\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \|_H^2$$

$$w_c^s = M_c / M \ \text{ and } w_c^t = N_c / N$$

Effect of class weight bias should be removed:

① Changes in sample selection criteria

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t) \|_H^2$$

$$\Downarrow$$

$$\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \|$$

$$w_c^s = M_c / M \ \text{ and } w_c^t = N_c / N$$

Effect of class weight bias should be removed:

① Changes in sample selection criteria

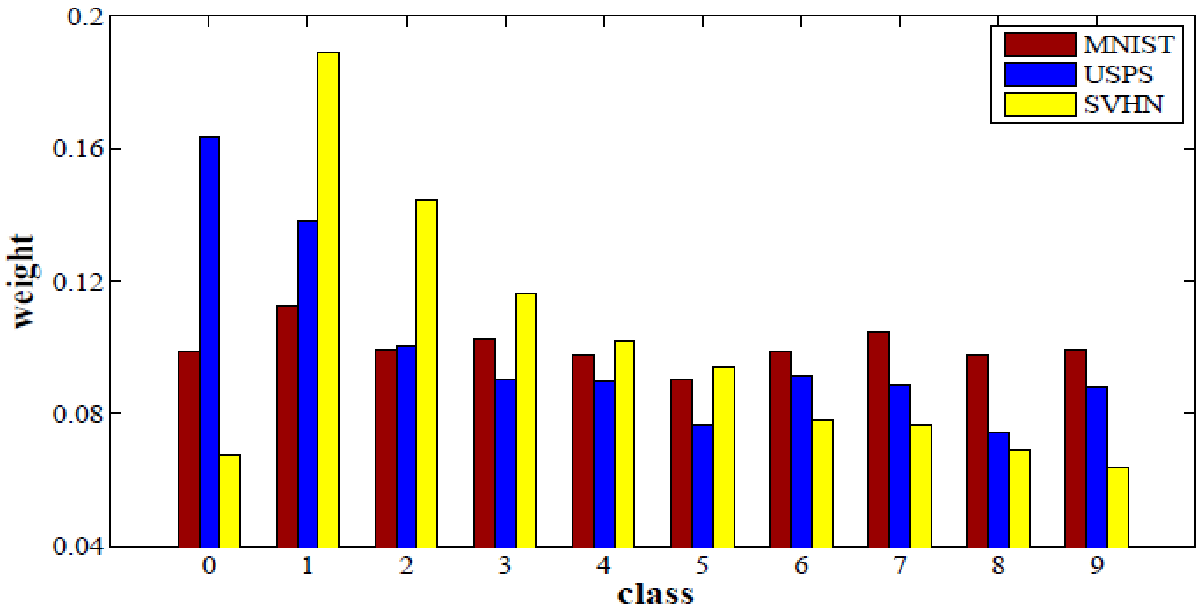

Figure 2. Class prior distribution of three digit recognition datasets.

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s,\mathrm{D}_t)=\|\frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s)-\frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t)\|_H^2$$

$$\Downarrow$$

$$\|\sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s))-\sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t))\|_H^2$$

$$w_c^s = M_c/M \text{ and } w_c^t = N_c/N$$

Effect of class weight bias should be removed:

① Changes in sample selection criteria

② Applications are not concerned with class prior distribution

Motivation

- *Class weight bias* cross domains remains unsolved but ubiquitous

$$\mathrm{MMD}^2(\mathrm{D}_s,\mathrm{D}_t) = \| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t) \|_H^2$$

$$\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \|_H^2$$

$$w_c^s = M_c / M \ \text{ and } w_c^t = N_c / N$$

Effect of class weight bias should be removed:

① Changes in sample selection criteria

② Applications are not concerned with class prior distribution

MMD can be minimized by either learning domain invariant representation or preserving the class weights in source domain.

Weighted MMD

Main idea: reweighting classes in source domain so that they have the same class weights as target domain

- Introducing an auxiliary weight α_c for each class c in source domain

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \left\| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t) \right\|_H^2$$

$$\left\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \right\|_H^2$$

$$\alpha_c = w_c^t / w_c^s$$

Weighted MMD

Main idea: reweighting classes in source domain so that they have the same class weights as target domain

- Introducing an auxiliary weight α_c for each class c in source domain

$$\mathrm{MMD}^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t) \|_H^2$$

$$\| \sum_{c=1}^{C} w_c^s E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \|_H^2$$

$$\alpha_c = w_c^t / w_c^s$$

$$\mathrm{MMD}_w^2(\mathrm{D}_s, \mathrm{D}_t) = \| \frac{1}{M}\sum_{i=1}^{M}\alpha_{y_i^s}\phi(\mathrm{x}_i^s) - \frac{1}{N}\sum_{j=1}^{t}\phi(\mathrm{x}_j^t) \|_H^2$$

$$\| \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^s)) - \sum_{c=1}^{C} w_c^t E_c(\phi(\mathrm{x}^t)) \|_H^2$$

Weighted DAN

1. Replace MMD with weighted MMD item in DAN[4]:

$$\min_{\mathrm{W}} \frac{1}{M} \sum_{i=1}^{M} \ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \lambda \sum_{l \in \{l_1, \ldots, l_L\}} \mathrm{MMD}_l(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

[4] Long M, Cao Y, Wang J. Learning Transferable Features with Deep Adaptation Networks[J]., 2015.

Weighted DAN

1. Replace MMD with weighted MMD item in DAN[4]:

$$\min_{\mathrm{W}} \frac{1}{M}\sum_{i=1}^{M} \ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \lambda \sum_{l \in \{l_1,\dots,l_L\}} \mathrm{MMD}_l(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

$$\Downarrow$$

$$\min_{\mathrm{W},\boldsymbol{\alpha}} \frac{1}{M}\sum_{i=1}^{M} \ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \lambda \sum_{l \in \{l_1,\dots,l_L\}} \mathrm{MMD}_{l,w}(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

[4] Long M, Cao Y, Wang J. Learning Transferable Features with Deep Adaptation Networks[J]., 2015.

Weighted DAN

1. Replace MMD with Weighted MMD item in DAN[4]:

$$\min_{\mathrm{W}} \frac{1}{M}\sum_{i=1}^{M}\ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \lambda \sum_{l\in\{l_1,\dots,l_L\}} \mathrm{MMD}_l(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

2. To further exploit the unlabeled data in target domain, empirical risk is considered as semi-supervised model in [5]:

$$\min_{\mathrm{W},\{\hat{y}_j\}_{j=1}^N,\boldsymbol{\alpha}} \frac{1}{M}\sum_{i=1}^{M}\ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \gamma\frac{1}{N}\sum_{j=1}^{N}\ell(\mathrm{x}_i^t, \hat{y}_i^t; \mathrm{W}) + \lambda \sum_{l\in\{l_1,\dots,l_L\}} \mathrm{MMD}_{l,w}(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

[4] Long M, Cao Y, Wang J. Learning Transferable Features with Deep Adaptation Networks[J]., 2015.
[5] Amini, Massih-Reza, and Patrick Gallinari. "Semi-supervised logistic regression." *Proceedings of the 15th European Conference on Artificial Intelligence*. IOS Press, 2002.

Optimization: an extension of CEM[6]

Parameters to be estimated including three parts, i.e., $\mathrm{W}, \boldsymbol{\alpha}, \{\hat{y}_j^t\}_{j=1}^N$

The model is optimized by alternating between three steps :

- E-step:

  Fixed W, estimating the class posterior probability $p(y_j^t = c \mid \mathrm{x}_j^t)$ of target samples:

$$p(y_j^t = c \mid \mathrm{x}_j^t) = g(\mathrm{x}_j^t, \mathrm{W})$$

[7] Celeux, Gilles, and Gérard Govaert. "A classification EM algorithm for clustering and two stochastic versions." *Computational statistics & Data analysis* 14.3 (1992): 315-332.

Optimization: an extension of CEM[6]

Parameters to be estimated including three parts, i.e., $\mathrm{W}, \boldsymbol{\alpha}, \{\hat{y}_j^t\}_{j=1}^N$

The model is optimized by alternating between three steps :

- E-step:

  Fixed W, estimating the class posterior probability $p(y_j^t = c \mid \mathrm{x}_j^t)$ of target samples:

$$p(y_j^t = c \mid \mathrm{x}_j^t) = g(\mathrm{x}_j^t, \mathrm{W})$$

- C-step:

  ① Assign the pseudo labels $\{\hat{y}_j^t\}_{j=1}^N$ on target domain: $\hat{y}_j^t = \arg\max_c p(y_j^t = c \mid \mathrm{x}_j^t)$

  ② update the auxiliary class-specific weights $\boldsymbol{\alpha}$ for source domain:

$$\alpha_c = \hat{w}_c^t \big/ w_c^s \quad \text{where } \hat{w}_c^t = \sum_j \mathbf{1}_c(\hat{y}_j^t) \big/ N$$

$\mathbf{1}_c(x)$ is an indictor function which equals 1 if x = c, and equals 0 otherwise.

[7] Celeux, Gilles, and Gérard Govaert. "A classification EM algorithm for clustering and two stochastic versions." *Computational statistics & Data analysis* 14.3 (1992): 315-332.

Optimization: an extension of CEM[6]

Parameters to be estimated including three parts, i.e., $\mathrm{W}, \boldsymbol{\alpha}, \{\hat{y}_j^t\}_{j=1}^N$

The model is optimized by alternating between three steps :

- M-step:

  Fixed $\{\hat{y}_j^t\}_{j=1}^N$ and $\boldsymbol{\alpha}$, updating W. The problem is reformulated as:

$$\min_{\mathrm{W}} \frac{1}{M}\sum_{i=1}^{M} \ell(\mathrm{x}_i^s, y_i^s; \mathrm{W}) + \gamma \sum_{j=1}^{N} \ell(\mathrm{x}_i^t, y_i^t; \mathrm{W}) + \lambda \sum_{l \in \{l_1, \ldots, l_L\}} \mathrm{MMD}_{l,w}(\mathrm{D}_s^l, \mathrm{D}_t^l)$$

  The gradient of the three items is computable and W can be optimized by using a mini-batch SGD.

[7] Celeux, Gilles, and Gérard Govaert. "A classification EM algorithm for clustering and two stochastic versions." *Computational statistics & Data analysis* 14.3 (1992): 315-332.

Experimental results

- Comparison with state-of-the-arts

Method	A→C	W→C	D→C	C→A	C→W	C→D	Avg.
AlexNet [21]	84.0±0.3	77.9±0.4	81.0±0.4	91.3±0.2	83.2±0.3	89.1±0.2	84.0
LapCNN (AlexNet) [38]	83.6±0.6	77.8±0.5	80.6±0.4	92.1±0.3	81.6±0.4	87.8±0.4	83.9
DDC (AlexNet) [36]	84.3±0.5	76.9±0.4	80.5±0.2	91.3±0.3	85.5±0.3	89.1±0.3	84.6
DAN (AlexNet) [25]	86.0±0.5	81.5±0.3	82.0±0.4	92.0±0.3	92.6±0.4	90.5±0.2	87.3
WDAN (AlexNet)	**86.9**±0.1	**84.1**±0.2	**83.9**±0.1	**93.1**±0.2	**93.6**±0.2	**93.4**±0.2	**89.2**
WDAN⋆ (AlexNet)	87.1±0.2	85.1±0.3	85.2±0.2	93.2±0.1	93.5±0.3	94.5±0.2	89.8
GoogLeNet [34]	91.3±0.2	88.2±0.3	88.9±0.3	95.2±0.1	92.5±0.2	94.7±0.3	91.8
DDC (GoogLeNet) [36]	91.4±0.2	88.7±0.3	89.0±0.4	95.3±0.2	93.0±0.1	94.9±0.4	92.1
DAN (GoogLeNet) [25]	91.4±0.3	89.7±0.2	89.1±0.4	95.5±0.2	93.1±0.3	95.3±0.1	92.3
WDAN (GoogLeNet)	**92.2**±0.2	**91.0**±0.5	**89.8**±0.3	**95.5**±0.3	**95.4**±0.2	**95.5**±0.5	**93.2**
VGGnet-16 [32]	89.6±0.4	88.1±0.4	85.4±0.5	93.7±0.2	94.3±0.2	93.7±0.2	90.8
DAN (VGGnet-16) [25]	91.2±0.2	90.6±0.3	87.1±0.4	95.7±0.2	95.3±0.3	94.7±0.1	92.4
WDAN (VGGnet-16)	**91.4**±0.2	**91.0**±0.2	**89.0**±0.3	**95.7**±0.1	**95.8**±0.2	**95.9**±0.3	**93.1**

Table 1. Experimental results on office-10+Caltech-10

Experimental results

- Empirical analysis

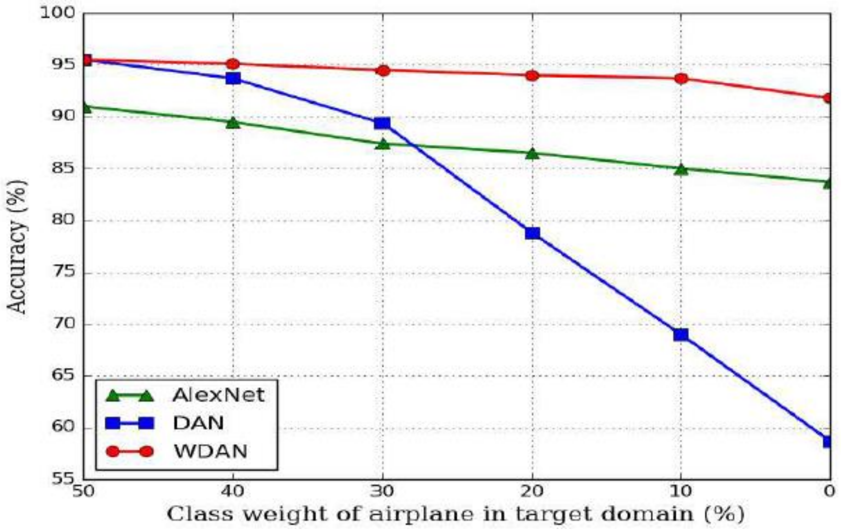

Figure 3. Performance of various model under different class weight bias.

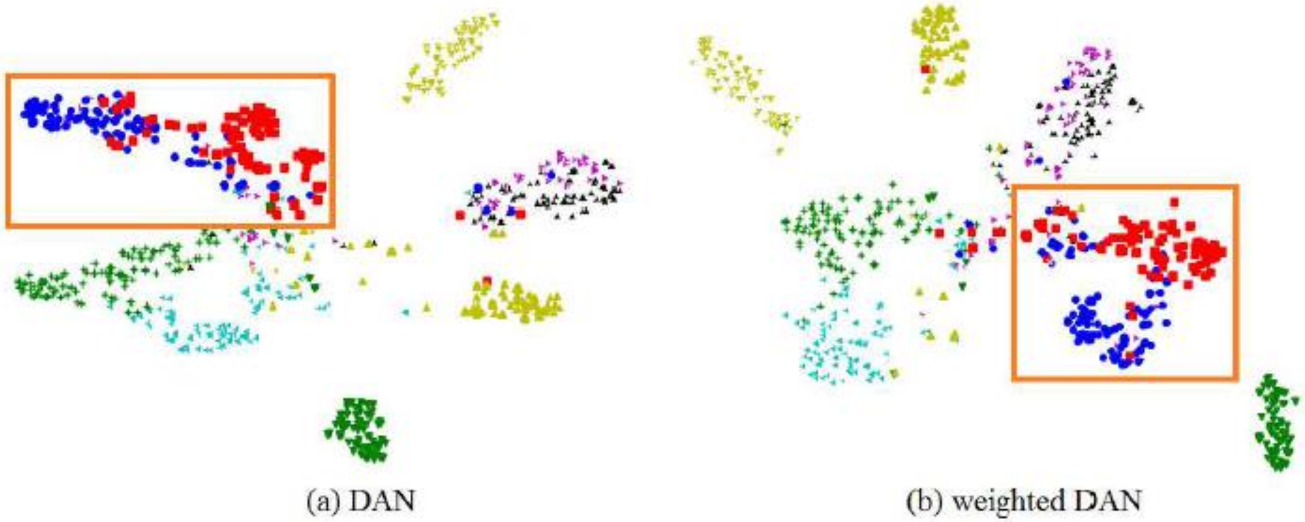

Figure 4. Visualization of the learned features of DAN and weighted DAN.

Summary

- Introduce class-specific weight into MMD to reduce the effect of class weight bias cross domains.
- Develop WDAN model and optimize it in an CEM framework.
- Weighted MMD can be applied to other scenarios where MMD is used for distribution distance measurement, e.g., image generation

Thanks!

Paper & code are available

Paper: https://arxiv.org/abs/1705.00609

Code: https://github.com/yhldhit/WMMD-Caffe