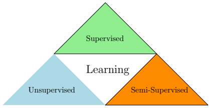
# Semi-Supervised Learning and Applications

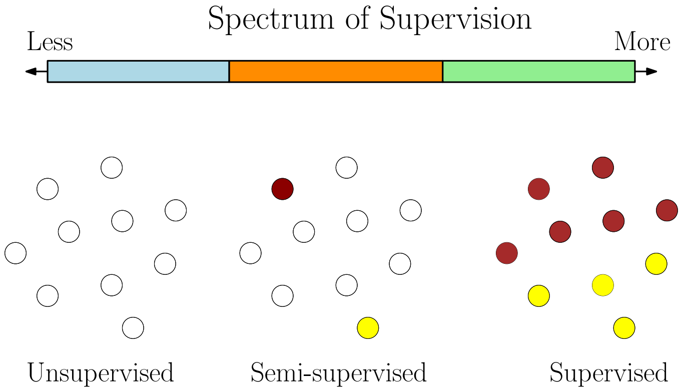
Siamak Mehrkanoon
Assistant professor
siamak.mehrkanoon@maastrichtuniversity.nl

Maastricht University

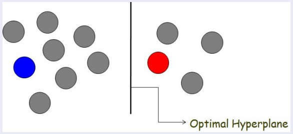
4th Oct 2021

## MSS-KSC


Supervised
Learning
Unsupervised
Semi-Supervised

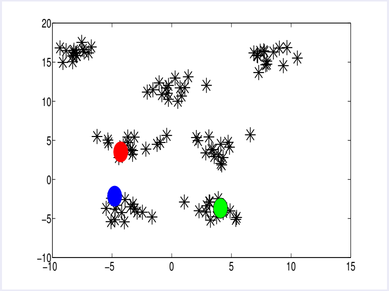

Spectrum of Supervision
Less
More
Unsupervised
Semi-supervised
Supervised

## MSS-KSC



### Semi-supervised classification:

Uses both unlabeled and labeled data to obtain a better classification model, and better predictions on unseen test data points.
# classes = # available class labels.



### Semi-supervised clustering:

Addresses the problem of exploiting additional labeled data to adjust the cluster membership of unlabeled data.
We do not have labeled data points from all the existing clusters.

## MSS-KSC

Consider training data points

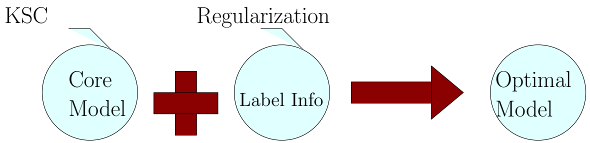
$$\mathcal{D} = \{\underbrace{x_1, \ldots, x_{n_u}}_{\substack{\textit{Unlabeled} \\ (\mathcal{D}_U)}}, \underbrace{x_{n_u+1}, .., x_n}_{\substack{\textit{Labeled} \\ (\mathcal{D}_L)}}\},$$

where $\{x_i\}_{i=1}^{n} \in \mathbb{R}^d$.

- the first $n_u$ data points do not have labels whereas the last $n_L = n - n_u$ points have been labeled.
- assume that there are $Q$ classes: then the label indicator matrix $Y \in \mathbb{R}^{n_L \times Q}$ is defined as follows:

$$Y_{ij} = \begin{cases} +1 & \text{if the } i\text{th point belongs to the } j\text{th class} \\ -1 & \text{otherwise.} \end{cases}$$

## MSS-KSC



**Primal:**

MSS-KSC is formulated as follows: [see[1]]

$$\min_{w^{(\ell)},b^{(\ell)},e^{(\ell)}} \quad \frac{1}{2}\sum_{\ell=1}^{Q} w^{(\ell)^T} w^{(\ell)} - \frac{\gamma_1}{2}\sum_{\ell=1}^{Q} e^{(\ell)^T} V e^{(\ell)} + \frac{\gamma_2}{2}\sum_{\ell=1}^{Q} (e^{(\ell)} - c^{(\ell)})^T A (e^{(\ell)} - c^{(\ell)})$$

$$\text{subject to} \quad e^{(\ell)} = \Phi w^{(\ell)} + b^{(\ell)} 1_n, \ \ \ell = 1, \ldots, Q,$$

**Dual:**

$$\gamma_2 \left( I_n - \frac{R 1_n 1_n^T}{1_n^T R 1_n} \right) c^{(\ell)} = \alpha^{(\ell)} - R \left( I_n - \frac{1_n 1_n^T R}{1_n^T R 1_n} \right) \Omega \alpha^{(\ell)}, \ \ \ell = 1, \ldots, Q.$$

[1] Siamak Mehrkanoon et al. “Multiclass semisupervised learning based upon kernel spectral clustering”. In: *IEEE transactions on neural networks and learning systems* 26.4 (2014), pp. 720–733.

## MSS-KSC

The bias term becomes:

$$b^{(\ell)} = (1/1_n^T R 1_n)(-1_n^T \gamma_2 c^{(\ell)} - 1_n^T R \Omega \alpha^{(\ell)}), \ell = 1, \ldots, Q.$$

Considering the test set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ the score variables of the test points become:

$$e_{test}^{(\ell)} = \Phi_{test} w^{\ell} + b^{(\ell)} 1_{N_{test}} = \Omega_{test} \alpha^{(\ell)} + b^{(\ell)} 1_{N_{test}}, \; \ell = 1, \ldots, Q,$$

where $\Omega_{test} = \Phi_{test} \Phi^T$.

- Realizing low embedding dimension to reveal the existing number of clusters.
- Addressing both multi-class semi-supervised classification and clustering.
- Existence of model selection scheme and out-of-sample property.

## MSS-KSC

**Algorithm 1:** Multi-Class Semi-Supervised Classification

```
Input: Training data set $\mathcal{X}$, labels $Z$, the tuning parameters $\{\gamma_i\}_{i=1}^2$, the kernel parameter (if any), the test set
       $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ and codebook $\mathcal{CB} = \{c_q\}_{q=1}^Q$
Output: Class membership of test data points $\mathcal{X}^{test}$

1 Solve the dual linear system (6) to obtain $\{\alpha^\ell\}_{\ell=1}^Q$ and compute the bias term $\{b^\ell\}_{\ell=1}^Q$ using (7).
2 Estimate the test data projections $\{e_{test}^{(\ell)}\}_{\ell=1}^Q$ using (7).
3 Binarize the test projections and form the encoding matrix $\mathcal{E} = [\text{sign}(e_{test}^{(1)}), \ldots, \text{sign}(e_{test}^{(Q)})]_{N_{test} \times Q}$ for the
    test points (Here $e_{test}^{(\ell)} = [e_{test,1}^{(\ell)}, \ldots, e_{test,N_{test}}^{(\ell)}]^T$.)
4 $\forall i$, assign $x_i^{test}$ to class $q^*$, where $q^* = \underset{q}{\text{argmin}}\, d_H(e_{test,i}^\ell, c_q)$ and $d_H(\cdot, \cdot)$ is the Hamming distance.
```

## MSS-KSC

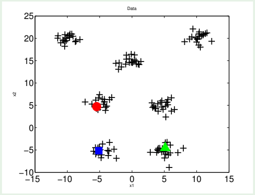
**Algorithm 2:** Semi-Supervised Clustering

**Input:** Training data set $\mathcal{X}$, labels $Z$, the tuning parameters $\{\gamma_i\}_{i=1}^{2}$, the kernel parameter (if any), number of clusters $k$, the test set $\mathcal{D}^{test} = \{x_i^{test}\}_{i=1}^{N_{test}}$ and number of available class labels i.e. $Q$

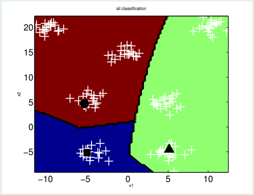
**Output:** Cluster membership of test data points $\mathcal{X}^{test}$

```
1 Solve the dual linear system (6) to obtain $\{\alpha^\ell\}_{\ell=1}^{Q}$ and compute the bias term $\{b^\ell\}_{\ell=1}^{Q}$ using (7).
2 Binarize the solution matrix $\mathcal{S} = [\text{sign}(\alpha^{(1)}), \ldots, \text{sign}(\alpha^{(Q)})]_{M \times Q}$, where $\alpha^\ell = [\alpha_1^\ell, \ldots, \alpha_M^\ell]^T$.
3 Form the codebook $\mathcal{CB} = \{c_q\}_{q=1}^{p}$, where $c_q \in \{-1, 1\}^Q$, using the $k$ most frequently occurring encodings
    from unique rows of solution matrix $\mathcal{S}$.
4 Estimate the test data projections $\{e_{test}^{(\ell)}\}_{\ell=1}^{Q}$ using (7).
5 Binarize the test projections and form the encoding matrix $\mathcal{E} = [\text{sign}(e_{test}^{(1)}), \ldots, \text{sign}(e_{test}^{(Q)})]_{N_{test} \times Q}$ for the
    test points (Here $e_{test}^\ell = [e_{test,1}^\ell, \ldots, e_{test,N_{test}}^\ell]^T$).
6 $\forall i$, assign $x_i^{test}$ to class/cluster $q^*$, where $q^* = \underset{q}{\text{argmin}}\, d_H(e_{test,i}^\ell, c_q)$ and $d_H(\cdot,\cdot)$ is the Hamming distance.
```

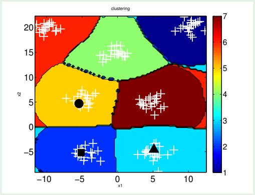
## MSS-KSC



(a)

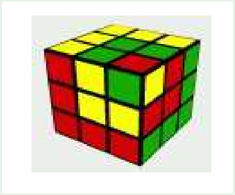


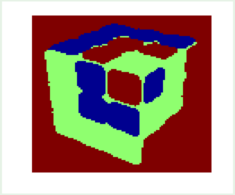
(b)

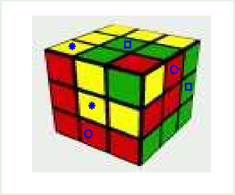


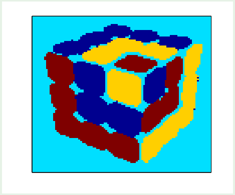
(c)

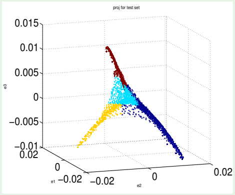
## MSS-KSC


(d)

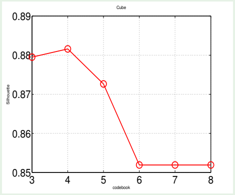

(e)


(f)


(g)

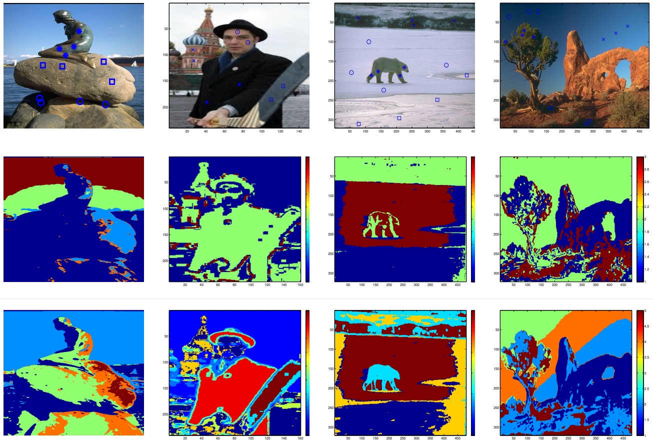


(h)



(i)

## MSS-KSC



## MSS-KSC

Table: Average Accuracy for four data sets

| Dataset | # of CL | $\mathcal{D}^{\text{train}}_{\text{Labeled}}/\mathcal{D}^{\text{train}}_{\text{Unlabeled}}/\mathcal{D}^{\text{test}}$ | Method: MSS-KSC | LapSVMp | means3vm-iter | means3vm-mkl |
|---|---|---|---|---|---|---|
| Wine | 3 | 36/107/35 | **$0.96 \pm 0.02$** | $0.94 \pm 0.03$ | $0.95 \pm 0.02$ | $0.94 \pm 0.07$ |
| Iris | 3 | 30/90/30 | $0.89 \pm 0.08$ | $0.88 \pm 0.05$ | **$0.90 \pm 0.03$** | $0.89 \pm 0.01$ |
| Zoo | 7 | 21/60/20 | **$0.93 \pm 0.05$** | $0.90 \pm 0.06$ | $0.88 \pm 0.02$ | $0.89 \pm 0.07$ |
| Seeds | 3 | 42/126/42 | **$0.90 \pm 0.04$** | $0.89 \pm 0.03$ | $0.88 \pm 0.07$ | $0.89 \pm 0.02$ |

Table: Comparison of KSC and MSS-KSC for image segmentations

| | | $\mathcal{D}$ | | $\mathcal{D}^{val}$ | | F-measure | | Variation of information | |
|---|---|---|---|---|---|---|---|---|---|
| **Image ID** | $Q$ | $\mathcal{D}_u$ | $\mathcal{D}_L$ | $\mathcal{D}_u$ | $\mathcal{D}_L$ | **KSC** | **MSS-KSC** | **KSC** | **MSS-KSC** |
| 372019 | 3 | 500 | 6 | 3000 | 6 | 0.40 | **0.44** | 2.83 | **2.44** |
| 385039 | 5 | 500 | 14 | 3000 | 12 | 0.48 | **0.48** | 3.20 | **3.18** |
| 388067 | 3 | 500 | 6 | 3000 | 6 | 0.60 | **0.74** | 4.61 | **4.50** |
| 8049 | 3 | 500 | 6 | 3000 | 7 | 0.70 | **0.75** | 2.22 | **2.07** |

## Fixed-Size MSSKSC (FS-MSS-KSC)

- Fixed-Size MSS-KSC (FS-MSSKSC). [see[2]]
- Reduced Kernel Techniques (RD-MSSKSC)
- Random Fourier Features (RFF-MSSKSC) [see[3]]

[2] Siamak Mehrkanoon and Johan AK Suykens. “Large scale semi-supervised learning using KSC based model”. In: *IEEE International Joint Conference on Neural Networks (IJCNN)*. 2014.

[3] Siamak Mehrkanoon and Johan AK Suykens. “Scalable Semi-supervised kernel spectral learning using random Fourier features”. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. 2016.

## Fixed-Size MSSKSC (FS-MSS-KSC)

### Nyström approximation

- We work with a subsample (prototype vectors) of size $m \ll n$.
- The *i*-th component of the *m*-dimensional feature map, for any point $x \in \mathbb{R}^d$, can be obtained as follows:

$$\hat{\varphi}_i(x) = \frac{1}{\lambda_i^{(s)}} \sum_{k=1}^{m} u_{ki}\, K(x_k, x)$$

where $\lambda_i^{(s)}$ and $u_i$ are eigenvalues and eigenvectors of the kernel matrix $\Omega_{m\times m}$.

- The explicit *m*-dimensional feature map $\hat{\varphi} : \mathbb{R}^d \to \mathbb{R}^m$ for the given point $x \in \mathbb{R}^d$:

$$\hat{\varphi}(x) = [\hat{\varphi}_1(x), \ldots, \hat{\varphi}_m(x)]^T$$

## Fixed-Size MSSKSC (FS-MSS-KSC)

Given *m*-dimensional approximation to the feature map, i.e.

$$\hat{\Phi} = [\hat{\varphi}(x_1), \ldots, \hat{\varphi}(x_n)]^T \in \mathbb{R}^{n\times m}$$

one can rewrite the optimization problem as an unconstrained optimization problem:

$$\min_{w^{(\ell)}, b^{(\ell)}} J(w^{(\ell)}, b^{(\ell)}) = \frac{1}{2}\sum_{\ell=1}^{Q} w^{(\ell)^T} w^{(\ell)} -$$

$$\frac{\gamma_1}{2}\sum_{\ell=1}^{Q} (\hat{\Phi} w^{(\ell)} + b^{(\ell)} 1_N{}^T)^T V (\hat{\Phi} w^{(\ell)} + b^{(\ell)} 1_N) +$$

$$\frac{\gamma_2}{2}\sum_{\ell=1}^{Q} (c^{(\ell)} - \hat{\Phi} w^{(\ell)} + b^{(\ell)} 1_n)^T A (c^{(\ell)} - \hat{\Phi} w^{(\ell)} + b^{(\ell)} 1_n)$$

## Fixed-Size MSSKSC (FS-MSS-KSC)

- The solution is given by: [see[4]]

$$\begin{bmatrix} w^{(\ell)} \\ b^{(\ell)} \end{bmatrix} = \left(\Phi_e^T R \Phi_e + I_{(m+1)}\right)^{-1} \gamma_2 \Phi_e^T c^{(\ell)}, \ell = 1, \dots, Q,$$

- The score variables evaluated at the test set $\mathcal{D}^{\text{test}} = \{x_i\}_{i=1}^{n_{\text{test}}}$ become:

$$e_{\text{test}}^{(\ell)} = \hat{\Phi}_{\text{test}} w^{(\ell)} + b^{(\ell)} 1_{n_{\text{test}}} \ \ell = 1, \dots, Q,$$

where $\hat{\Phi}_{\text{test}} = [\hat{\varphi}(x_1), \dots, \hat{\varphi}(x_{n_{\text{test}}})]^T \in \mathbb{R}^{n_{\text{test}} \times m}$.

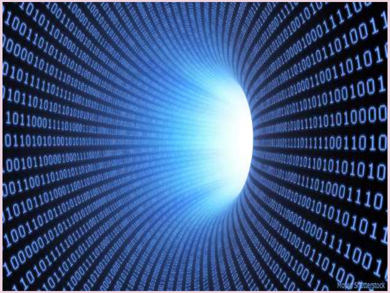
[4] Siamak Mehrkanoon and Johan AK Suykens. "Large scale semi-supervised learning using KSC based model". In: *IEEE International Joint Conference on Neural Networks (IJCNN)*. 2014.

## Fixed-Size MSSKSC (FS-MSS-KSC)

Table: The average test accuracy and computation time of the FS-MSSKSC models [Mehrkanoon & Suykens, *WCCI-IJCNN 2014*] on real-life datasets over 10 simulation runs.

| **Dataset** | $\mathcal{D}_{\text{tr}}^{L}$ | $\mathcal{D}_{\text{tr}}^{U}$ | $\mathcal{D}_{\text{test}}$ | **Test Accuracy** | **(Training/Test) time (S)** |
|---|---|---|---|---|---|
| Shuttle | 4,000 | 12,000 | 11,599 | 0.993 | 0.05/0.08 |
| | 8,000 | 24,000 | 11,599 | 0.995 | 0.12/0.04 |
| IJCNN | 4,000 | 20,000 | 28,338 | 0.935 | 0.16/0.30 |
| | 16,000 | 80,000 | 28,338 | 0.955 | 1.31/0.60 |
| Skin | 8,000 | 40,000 | 49,011 | 0.997 | 0.18/0.29 |
| | 32,000 | 160,000 | 49,011 | 0.998 | 1.50/0.76 |
| Cod-rna | 8,000 | 40,000 | 66,230 | 0.959 | 0.27/0.40 |
| | 32,000 | 160,000 | 66,230 | 0.962 | 2.26/1.21 |
| Covertype | 8,000 | 40,000 | 116,202 | 0.732 | 0.25/0.85 |
| | 64,000 | 320,000 | 116,202 | 0.781 | 7.76/3.09 |
| **SUSY** | 500,000 | 1,000,000 | 1,000,000 | 0.771 | 2.05/1.61 |
| | 1,000,000 | 2,000,000 | 1,000,000 | 0.783 | 5.87/2.61 |

## I-MSS-KSC



An incremental semi-supervised clustering algorithm (I-MSS-KSC):

- Initialize the model using (MSS-KSC)
- Read new data points
- Provide the labels for some of the data points (Optional)
- Predict the cluster membership of arriving data points
- Update the model

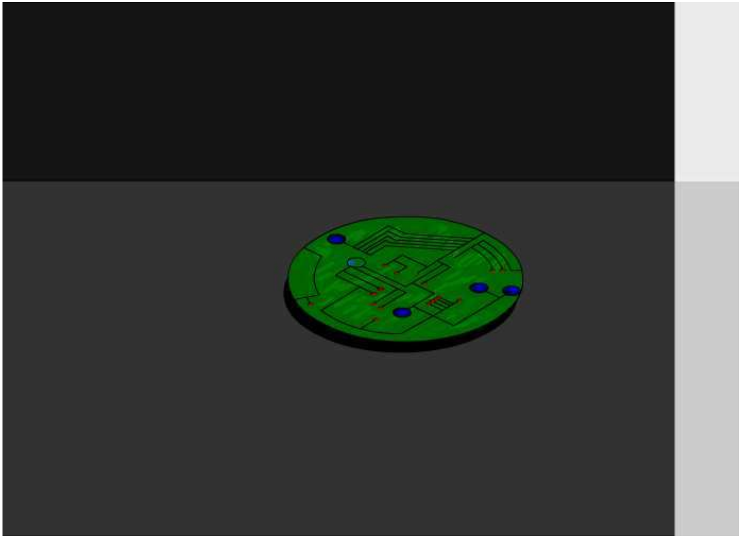
## Modes of implementation:

Two possibilities: [see[5]]

- I-MSS-KSC (-): the labels are only provided in the first stage, i.e. just for obtaining the initial cluster representatives and the subsequent set of data points do not have any label information.
- I-MSS-KSC (+): the user can also provide the labels for some of the subsequent set of data points.

[5] Siamak Mehrkanoon, Oscar Mauricio Agudelo, and Johan AK Suykens. "Incremental multi-class semi-supervised clustering regularized by Kalman filtering". In: *Neural Networks* (2015).

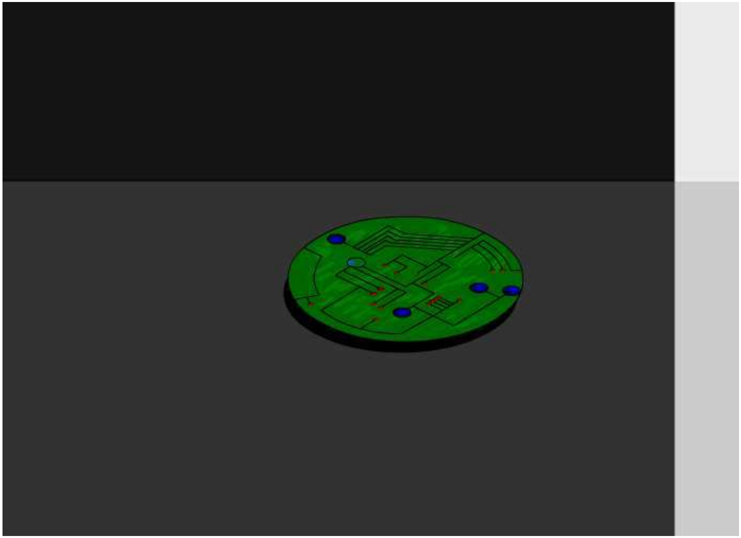
## I-MSS-KSC



## I-MSS-KSC

Fixed labels in time...

## I-MSS-KSC



## Common assumptions of ML algorithms:

- Both training and test data exhibit the same distribution.
- Fully labeled training data.
- Same feature domains.
- Stationary datasets.

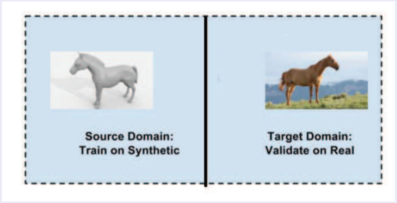
## However:

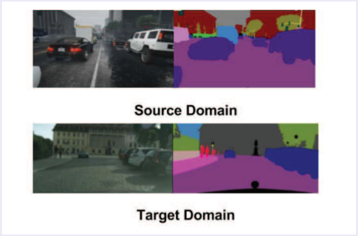
- Collecting training data in different domains is costly.
- Statistical properties evolve in time.
- Performance often drops significantly when the model is presented with data from a new domain.

## RSP-KCCA

**Aim:**

To transfer source knowledge and adapt it to new target domains ( with different distribution, feature domains and dimesntions).





- Spam filtering problem.
- Cross-domain action recognition, document classification, speech recognition.

## Office dataset

**First row:** Images of keyboard and monitors from webcam domain. **Second row:** Images of keyboard and monitors from dslr domain.

- Webcam and DSLR domains
- 10 classes: Calculator, Headphones, Computer-Keyboard, Computer-Monitor, Coffee mug, and etc.
- Homogeneous domains (800 dimensional features)

## Learning types:

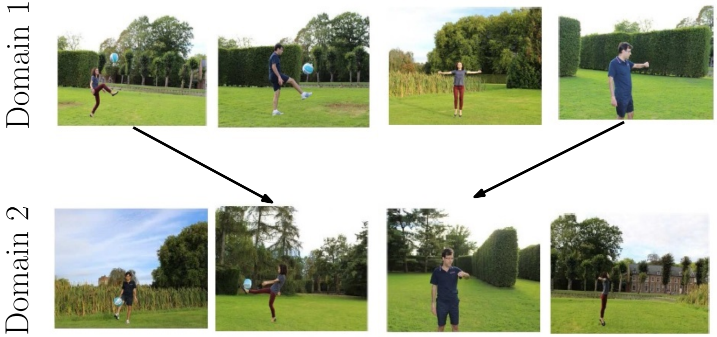
- Unsupervised domain-adaptation
- Supervised domain-adaptation
- Semi-Supervised domain-adaptation

## Domain types:

- Homogeneous domains
  $X_s = X_t$ (same feature space), but $P(X_s) \neq P(X_t)$
- Heterogeneous domains
  $X_s \neq X_t$, but $P(X_s) \neq P(X_t)$

## Side Information

Often side information in the form of correspondence instances (paired instances) are available for either unlabeled or labeled instances across domains.



## RSP-KCCA

Let $\mathcal{D}^{(1)} = \{ \underbrace{x_1^{(1)}, ..., x_{n_p}^{(1)}}_{\substack{\textit{paired and unlabeled} \\ (\mathcal{D}_{p,ul}^{(1)})}} \}$, and $\mathcal{D}^{(2)} = \{ \underbrace{x_1^{(2)}, ..., x_{n_p}^{(2)}}_{\substack{\textit{paired and unlabeled} \\ (\mathcal{D}_{p,ul}^{(2)})}} \}$,

denote $n_p$ training observations of $x^{(1)}$ and $x^{(2)}$ from two domains.

### CCA

The objective of CCA is to find basis vectors $w^{(1)}$ and $w^{(2)}$ such that the projected variables ${w^{(1)}}^T x^{(1)}$ and ${w^{(2)}}^T x^{(2)}$ are maximally correlated:

$$\max_{w^{(1)}, w^{(2)}} \quad \rho = \frac{{w^{(1)}}^T C_{12} w^{(1)}}{\sqrt{{w^{(1)}}^T C_{11} w^{(1)}} \sqrt{{w^{(2)}}^T C_{22} w^{(2)}}}$$

where $C_{11}$, $C_{12}$ and $C_{22}$ are the covariance matrices.

- CCA measures the linear relationship.
- Kernel CCA.

## RSP-KCCA

Primal:

LS-SVM formulation to kernel CCA [see[6]]

$$\begin{aligned} \max_{w^{(1)},w^{(2)},e,r} \quad & \mu e^T r - \frac{\gamma_1}{2} e^T e - \frac{\gamma_2}{2} r^T r - \frac{1}{2} {w^{(1)}}^T w^{(1)} - \frac{1}{2} {w^{(2)}}^T w^{(2)} \\ \text{subject to} \quad & e = \Phi_c^{(1)} w^{(1)}, \\ & r = \Phi_c^{(2)} w^{(2)}, \end{aligned} \tag{1}$$

Dual:

$$\left[\begin{array}{c|c} 0 & \Omega_c^{(2)} \\ \hline \Omega_c^{(1)} & 0 \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \end{array}\right] = \lambda \left[\begin{array}{c|c} \gamma_1 \Omega_c^{(1)} + I_N & 0 \\ \hline 0 & \gamma_2 \Omega_c^{(2)} + I_N \end{array}\right] \left[\begin{array}{c} \alpha \\ \hline \beta \end{array}\right]$$

[6]**Johan_book**.

## RSP-KCCA

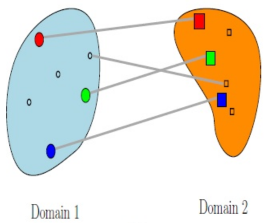
### Unlabeled + labeled semi-paired data

Assume that we are given two training datasets

$$\mathcal{D}^{(1)} = \{ \underbrace{x_1^{(1)}, \ldots, x_{n_p}^{(1)}}_{\substack{\textit{paired} \\ (\textit{labeled}\,/\,\textit{unlabeled}) \\ (\mathcal{D}^{(1)}_{p,ul} \cup \mathcal{D}^{(1)}_{p,l})}}, \underbrace{x_{n_p+1}^{(1)}, \ldots, x_{n_1}^{(1)}}_{\substack{\textit{unpaired} \\ (\textit{unlabeled}) \\ (\mathcal{D}^{(1)}_{up,ul})}}, \underbrace{x_{n_p+1}^{(1)}, \ldots, x_{N_1}^{(1)}}_{\substack{\textit{unpaired} \\ (\textit{labeled}) \\ (\mathcal{D}^{(1)}_{up,l})}} \},$$

and

$$\mathcal{D}^{(2)} = \{ \underbrace{x_1^{(2)}, \ldots, x_{n_p}^{(2)}}_{\substack{\textit{paired} \\ (\textit{labeled}\,/\,\textit{unlabeled}) \\ (\mathcal{D}^{(2)}_{p,ul} \cup \mathcal{D}^{(2)}_{p,l})}}, \underbrace{x_{n_p+1}^{(2)}, \ldots, x_{n_2}^{(2)}}_{\substack{\textit{unpaired} \\ (\textit{unlabeled}) \\ (\mathcal{D}^{(2)}_{up,ul})}} \},$$



(c)

Unlabeled + partially labeled paired data

## RSP-KCCA

**Setup**

- Only a small number of paired labeled data points from both domains are available.
- Source dataset is equipped with additional unpaired labeled instances during training.
- Assuming that there are $Q$ classes, the label indicator matrix $Y^{(1)} \in \mathbb{R}^{n_{L_1} \times Q}$ for the source domain:

$$Y_{ij}^{(1)} = \begin{cases} +1 & \text{if the } i\text{-th point belongs to the } j\text{-th class} \\ -1 & \text{otherwise,} \end{cases} \tag{2}$$

- Similarly one can define the label indicator matrix $Y^{(2)} \in \mathbb{R}^{n_{L_2} \times Q}$ for the target domain.

## RSP-KCCA

**Primal:**

Regularized Semi-Paired kernel canonical correlation analysis

$$\max_{w_\ell^{(1)}, w_\ell^{(2)}, r_\ell, e_\ell} \quad \mu \sum_{\ell=1}^{Q} e_\ell^T A r_\ell - \frac{1}{2}\sum_{j=1}^{2}\sum_{\ell=1}^{Q} {w_\ell^{(j)}}^T w_\ell^{(j)} - \frac{1}{2}\sum_{\ell=1}^{Q} {e_\ell}^T V_1 e_\ell - \frac{1}{2}\sum_{\ell=1}^{Q} {r_\ell}^T V_2 r_\ell + \frac{\gamma_3}{2}\sum_{\ell=1}^{Q} e_\ell^T c_\ell^{(1)} + r_\ell^T c_\ell^{(2)}$$

$$\text{subject to} \quad e_\ell = \Phi_c^{(1)} w_\ell^{(1)}, \ \ell = 1, \dots, Q, \qquad r_\ell = \Phi_c^{(2)} w_\ell^{(2)}, \ \ell = 1, \dots, Q, \tag{3}$$

Here:

$$C^{(1)} = [c_1^{(1)}, \dots, c_Q^{(1)}]_{n_1 \times Q} = \left[\frac{Y^{(1)}}{0_{n_{u_1} \times Q}}\right]$$

$$C^{(2)} = [c_1^{(2)}, \dots, c_Q^{(2)}]_{n_2 \times Q} = \left[\frac{Y^{(2)}}{0_{n_{u_2} \times Q}}\right]$$

$$A = \left[\begin{array}{c|c} I_{np \times np} & 0_{np \times n_2 - np} \\ \hline 0_{N_1 - np \times np} & 0_{N_1 - np \times n_2 - np} \end{array}\right]$$

$V_1 = \gamma_1 P_1 + (1 - \gamma_1) L_1$ and $V_2 = \gamma_2 P_2 + (1 - \gamma_2) L_2$ where $P_2$ is defined as previously and $P_1$ is defined as follows:

$$P_1 = \left[\begin{array}{c|c} I_{np \times np} & 0_{np \times N_1 - np} \\ \hline 0_{N_1 - np \times np} & 0_{N_1 - np \times N_1 - np} \end{array}\right]$$

see[a] for more details.

[a] Siamak Mehrkanoon and Johan AK Suykens. "Regularized Semipaired Kernel CCA for Domain Adaptation". In: *IEEE Transactions on Neural Networks and Learning Systems* (2017).

## RSP-KCCA

Dual:

$$\left[\begin{array}{c|c} V_1\Omega_c^{(1)} + I_{N_1} & -\mu A\Omega_c^{(2)} \\ \hline -\mu A^T\Omega_c^{(1)} & V_2\Omega_c^{(2)} + I_{n_2} \end{array}\right] \left[\begin{array}{c} \alpha_\ell \\ \hline \beta_\ell \end{array}\right] = \gamma_3 \left[\begin{array}{c} c_\ell^{(1)} \\ \hline c_\ell^{(2)} \end{array}\right], \; \ell = 1, \ldots, Q, \qquad (4)$$

- $\alpha_\ell$, $\beta_\ell$ are Lagrange multiplier vectors.
- $\Omega_c^{(1)}$ and $\Omega_c^{(2)}$ are the centered kernel matrices.

## RSP-KCCA

The score variables for the training dataset of the source and target domains can be expressed as follows:

$$\begin{cases} z_e^\ell = \Phi_c^{(1)} w_\ell^{(1)} = \Omega_c^{(1)} \alpha_\ell \ \ell = 1, \ldots, Q, \\ z_r^\ell = \Phi_c^{(2)} w_\ell^{(2)} = \Omega_c^{(2)} \beta_\ell \ \ell = 1, \ldots, Q. \end{cases}$$

Here, $\Omega_c^{(1)}$ and $\Omega_c^{(2)}$ are the centered kernel matrix calculated as

$$\Omega_c^{(1)} = M_c^{(1)} \Omega^{(1)} M_c^{(1)}$$

$$\Omega_c^{(2)} = M_c^{(2)} \Omega^{(2)} M_c^{(2)}$$

with centering matrix

$$M_c^{(1)} = I_{N_1} - \frac{1}{N_1} 1_{N_1} 1_{N_1}^T \text{ and } M_c^{(2)} = I_{n_2} - \frac{1}{n_2} 1_{n_2} 1_{n_2}^T.$$

## RSP-KCCA

### Synthetic example

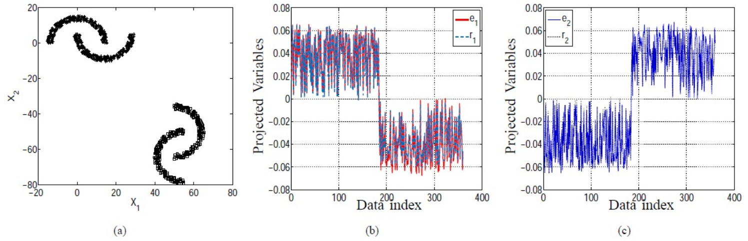


Fig. 6. Learned domain-invariant features using RSP-KCCA model. **(a)** Two-moons datasets. The left two-moons are source domains data and the right two-moons are considered to be target data. **(b)** Projected variables $e_1$ and $r_1$ for the source domain (left two-moons) and target domain (right two-moons). **(c)** Projected variables $e_2$ and $r_2$ for the source domain (left two-moons) and target domain (right two-moons).

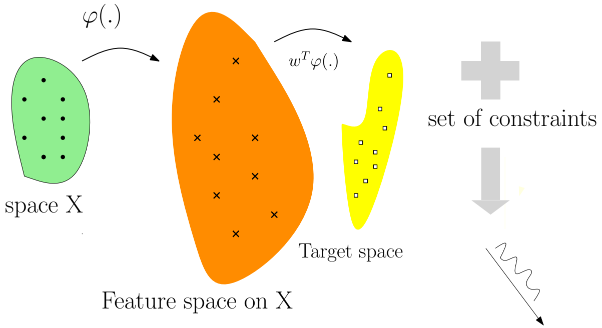
## RSP-KCCA

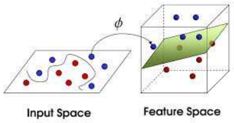
### Multiple Features Dataset

- It contains ten classes, $(0-9)$, of handwritten digits
- 200 images per class, thus for a total of 2000 images.
- six different types of features (heterogeneous features)

Table: Comparing the average test accuracy of the proposed RSP-KCCA model with those of mSDA model.

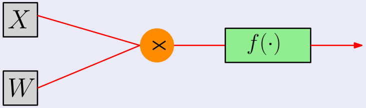
| | mSDA | | | | | | RSP-KCCA | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Source domain** | **fac** | **fou** | **kar** | **mor** | **pix** | **zer** | **fac** | **fou** | **kar** | **mor** | **pix** | **zer** |
| **fac** | 0.9601 | 0.1840 | 0.2453 | 0.3000 | 0.6217 | 0.3743 | 0.9560 | 0.7683 | 0.8987 | 0.6510 | 0.9233 | 0.7190 |
| **fou** | 0.6597 | 0.7953 | 0.3350 | 0.3003 | 0.6123 | 0.4147 | 0.8573 | 0.7993 | 0.8520 | 0.6360 | 0.7853 | 0.7860 |
| **kar** | 0.6457 | 0.3170 | 0.9440 | 0.2810 | 0.5483 | 0.3487 | 0.9270 | 0.7780 | 0.9454 | 0.6583 | 0.9540 | 0.7947 |
| **mor** | 0.5410 | 0.4083 | 0.2420 | 0.6976 | 0.5167 | 0.3230 | 0.7503 | 0.6857 | 0.8290 | 0.7344 | 0.7920 | 0.7347 |
| **pix** | 0.6243 | 0.1733 | 0.3480 | 0.2313 | 0.9644 | 0.3823 | 0.9503 | 0.7817 | 0.9177 | 0.6253 | 0.9590 | 0.7773 |
| **zer** | 0.6920 | 0.2697 | 0.2333 | 0.2943 | 0.5770 | 0.8031 | 0.8400 | 0.7837 | 0.8040 | 0.6340 | 0.8773 | 0.8063 |





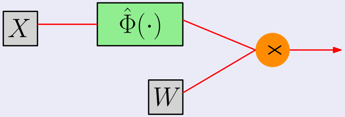
- RKHS
- Gaussian process (probabilistic setting)
- LSSVM (optimization setting)

## ANNs

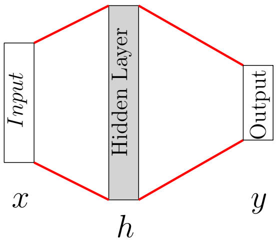


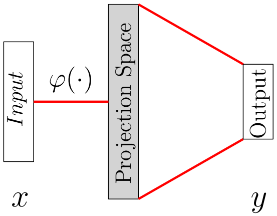
- The nonlinear activation function is applied on the weighted sum of the given input instance.
- Nonlinear in W.

## Kernel



- Weighted sum of the projected sample.
- Linear in the W.
- Capable of learning nonlinear decision boundaries.





- Implicit feature mapping: the projection space corresponds to a hidden layer in a neural network with infinite number of neurons.
- The dimension of the explicit feature map corresponds to the number of hidden units in the hidden layer of a neural network architecture.

Assuming that an explicit feature map is given, we formulate a two layer hybrid architecture as follows

$$h_1 = W_1 x + b_1, \quad h_2 = \hat{\varphi}(h_1), \quad S = W_2 h_2 + b_2,$$

where $W_1 \in \mathbb{R}^{d_1 \times d}$ and $W_2 \in \mathbb{R}^{Q \times d_2}$ are weight matrices and the bias vectors are denoted by $b_1$ and $b_2$. see[7]

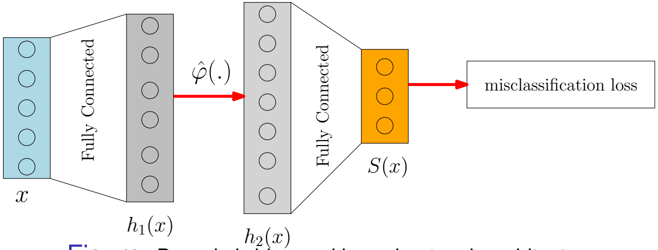


Figure: Deep hybrid neural kernel network architecture

[7] Siamak Mehrkanoon and Johan AK Suykens. "Deep hybrid neural-kernel networks using random Fourier features". In: *Neurocomputing* 298 (2018), pp. 46–54.

The formulation of the proposed method is as follows:

$$\min_{W_1, W_2, b_1, b_2} J(W_1, W_2, b_1, b_2) = \frac{1}{2} Tr(W_1 W_1^T) + \frac{1}{2} Tr(W_2 W_2^T) + \frac{\gamma_1}{n} \sum_{i=1}^{n} L(x_i, y_i).$$

- Here the cross-entropy loss is used.
- The stochastic gradient descent algorithm is used to train the model.
- The score variable for the test point $x_{\text{test}}$ are computed as follows:

$$S_{\text{test}} = W_2 \, \tilde{\varphi}(W_1 x_{\text{test}} + b1) + b_2$$

- The final class label for the test point $x_{\text{test}}$ is computed by:

$$\hat{y}_{\text{test}} = \underset{\ell=1,\ldots,Q}{\operatorname{argmax}} \, (S_{\text{test}})$$

.

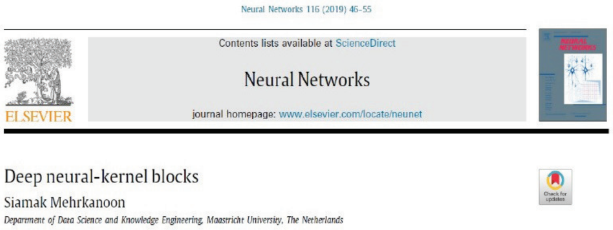
## Deep Neural Kernel Networks

Table: The average accuracy of the proposed deep model and the shallow LS-SVM with implicit and explicit feature maps on several real-life datasets.

| | | | | Shallow LS-SVM | |
|---|---|---|---|---|---|
| **Dataset** | ***n*** | ***d*** | **Hybrid Deep Model** | **Primal** | **Dual** |
| Australian | 690 | 14 | **0.85 ± 0.01** | 0.81 ± 0.01 | 0.83 ± 0.01 |
| Sonar | 208 | 60 | **0.75 ± 0.04** | 0.69 ± 0.07 | 0.70 ± 0.05 |
| Titanic | 2201 | 3 | **0.78 ± 0.01** | 0.77 ± 0.02 | **0.78 ± 0.01** |
| Monk2 | 432 | 6 | **1.00 ± 0.00** | 0.93 ± 0.05 | 0.95 ± 0.02 |
| Balance | 625 | 4 | **0.96 ± 0.01** | 0.93 ± 0.02 | 0.94 ± 0.01 |
| Magic | 19,020 | 10 | **0.86 ± 0.04** | 0.84 ± 0.01 | 0.84 ± 0.01 |
| Covertype | 581,012 | 54 | **0.85 ± 0.03** | 0.78 ± 0.01 | N.A |
| SUSY | 5,000,000 | 18 | **0.80 ± 0.02** | 0.78 ± 0.01 | N.A |

**Note:** “N.A” stands for Not Applicable due the large size of the dataset.

## Deep Neural Kernel Networks



Neural Networks 116 (2019) 46–55

Contents lists available at ScienceDirect

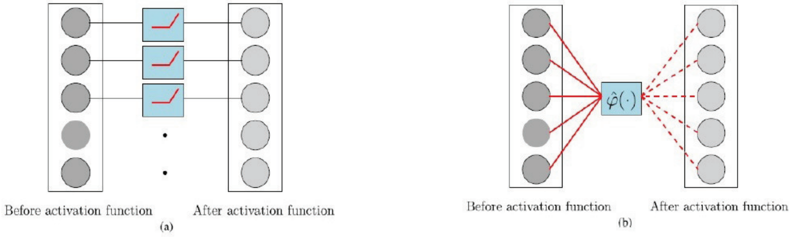
Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Deep neural-kernel blocks

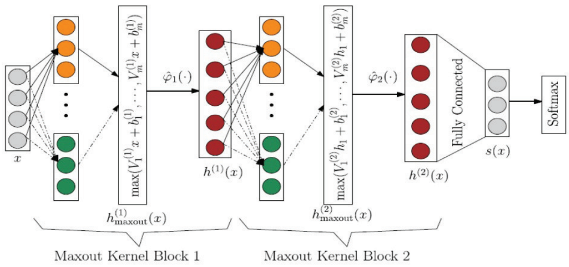
Siamak Mehrkanoon

*Department of Data Science and Knowledge Engineering, Maastricht University, The Netherlands*


Before activation function
After activation function
(a)
$\hat{\varphi}(\cdot)$
Before activation function
After activation function
(b)

## Maxout Kernel Blocks:

$$h^{(1)}_{\text{maxout}} = \max_{k \in \{1,\ldots,m\}} \quad V^{(1)}_k x + b^{(1)}_k,$$
$$h^{(1)} = \hat{\varphi}_{(1)}(h^{(1)}_{\text{maxout}}),$$
$$h^{(2)}_{\text{maxout}} = \max_{k \in \{1,\ldots,m\}} \quad V^{(2)}_k h^{(1)} + b^{(2)}_k,$$
$$h^{(2)} = \hat{\varphi}_{(2)}(h^{(2)}_{\text{maxout}}),$$
$$s(x) = Wh^{(2)} + b,$$



$x$
$\max(V^{(1)}_1 x + b^{(1)}_1, \ldots, V^{(1)}_m x + b^{(1)}_m)$
$\hat{\varphi}_1(\cdot)$
$h^{(1)}(x)$
$\max(V^{(2)}_1 h_1 + b^{(2)}_1, \ldots, V^{(2)}_m h_1 + b^{(2)}_m)$
$\hat{\varphi}_2(\cdot)$
$h^{(2)}(x)$
Fully Connected
$s(x)$
Softmax
$h^{(1)}_{\text{maxout}}(x)$
$h^{(2)}_{\text{maxout}}(x)$
Maxout Kernel Block 1
Maxout Kernel Block 2

**Fig. 3.** Deep neural-kernel network architecture with maxout kernel blocks.

## Conv-Kernel Blocks:

$$
\begin{aligned}
h^{(1)} &= \hat{\varphi}_{(1)}(h^{(1)}_{\text{Conv}}), \\
h^{(2)}_{\text{Conv}} &= Z^{(2)} R 1_m, \\
h^{(2)} &= \hat{\varphi}_{(2)}(h^{(2)}_{\text{Conv}}), \\
s(x) &= W h^{(2)} + b,
\end{aligned}
$$

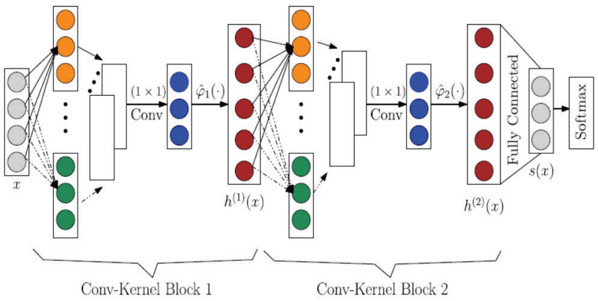


**Fig. 5.** Deep neural-kernel network architecture with convolutional kernel blocks.

## Deep Neural Kernel Networks

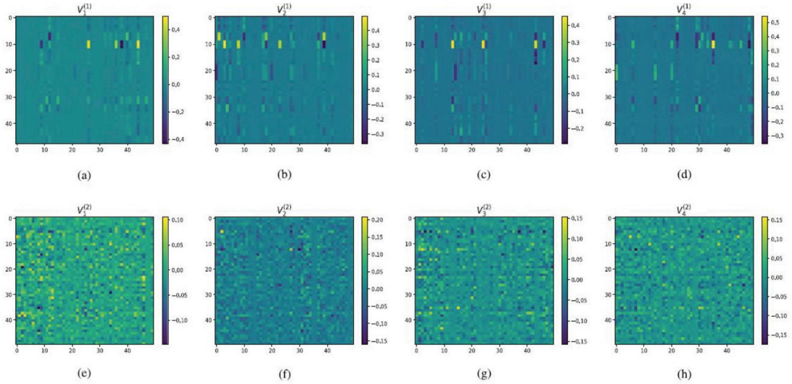


**Fig. 8.** Motor dataset. (a,b,c,d) The learned weights of the maxout kernel block 1. (e,f,g,h) The learned weights of the maxout kernel block 2. (See Fig. 3)

## Deep Neural Kernel Networks

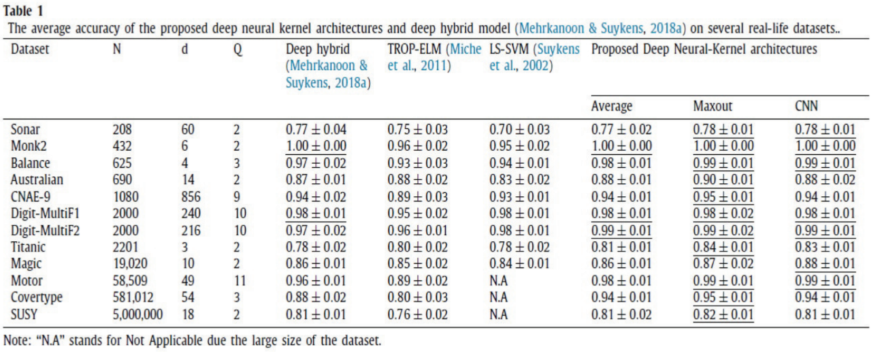


**Table 1**
The average accuracy of the proposed deep neural kernel architectures and deep hybrid model (Mehrkanoon & Suykens, 2018a) on several real-life datasets..

| Dataset | N | d | Q | Deep hybrid (Mehrkanoon & Suykens, 2018a) | TROP-ELM (Miche et al., 2011) | LS-SVM (Suykens et al., 2002) | Proposed Deep Neural-Kernel architectures | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | Average | Maxout | CNN |
| Sonar | 208 | 60 | 2 | $0.77 \pm 0.04$ | $0.75 \pm 0.03$ | $0.70 \pm 0.03$ | $0.77 \pm 0.02$ | $0.78 \pm 0.01$ | $0.78 \pm 0.01$ |
| Monk2 | 432 | 6 | 2 | $1.00 \pm 0.00$ | $0.96 \pm 0.02$ | $0.95 \pm 0.02$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ | $1.00 \pm 0.00$ |
| Balance | 625 | 4 | 3 | $0.97 \pm 0.02$ | $0.93 \pm 0.03$ | $0.94 \pm 0.01$ | $0.98 \pm 0.01$ | $0.99 \pm 0.01$ | $0.99 \pm 0.01$ |
| Australian | 690 | 14 | 2 | $0.87 \pm 0.01$ | $0.88 \pm 0.02$ | $0.83 \pm 0.02$ | $0.88 \pm 0.01$ | $0.90 \pm 0.01$ | $0.88 \pm 0.02$ |
| CNAE-9 | 1080 | 856 | 9 | $0.94 \pm 0.02$ | $0.89 \pm 0.03$ | $0.93 \pm 0.01$ | $0.94 \pm 0.01$ | $0.95 \pm 0.01$ | $0.94 \pm 0.01$ |
| Digit-MultiF1 | 2000 | 240 | 10 | $0.98 \pm 0.01$ | $0.95 \pm 0.02$ | $0.98 \pm 0.01$ | $0.98 \pm 0.01$ | $0.98 \pm 0.02$ | $0.98 \pm 0.01$ |
| Digit-MultiF2 | 2000 | 216 | 10 | $0.97 \pm 0.02$ | $0.96 \pm 0.01$ | $0.98 \pm 0.01$ | $0.99 \pm 0.01$ | $0.99 \pm 0.02$ | $0.99 \pm 0.01$ |
| Titanic | 2201 | 3 | 2 | $0.78 \pm 0.02$ | $0.80 \pm 0.02$ | $0.78 \pm 0.02$ | $0.81 \pm 0.01$ | $0.84 \pm 0.01$ | $0.83 \pm 0.01$ |
| Magic | 19,020 | 10 | 2 | $0.86 \pm 0.01$ | $0.85 \pm 0.02$ | $0.84 \pm 0.01$ | $0.86 \pm 0.01$ | $0.87 \pm 0.02$ | $0.88 \pm 0.01$ |
| Motor | 58,509 | 49 | 11 | $0.96 \pm 0.01$ | $0.89 \pm 0.02$ | N.A | $0.98 \pm 0.01$ | $0.99 \pm 0.01$ | $0.99 \pm 0.01$ |
| Covertype | 581,012 | 54 | 3 | $0.88 \pm 0.02$ | $0.80 \pm 0.03$ | N.A | $0.94 \pm 0.01$ | $0.95 \pm 0.01$ | $0.94 \pm 0.01$ |
| SUSY | 5,000,000 | 18 | 2 | $0.81 \pm 0.01$ | $0.76 \pm 0.02$ | N.A | $0.81 \pm 0.02$ | $0.82 \pm 0.01$ | $0.81 \pm 0.01$ |

Note: "N.A" stands for Not Applicable due the large size of the dataset.

## Deep Neural Kernel Networks

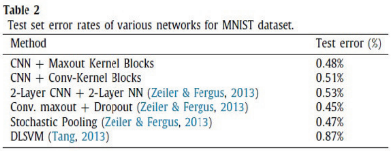


**Table 2**
Test set error rates of various networks for MNIST dataset.

| Method | Test error (%) |
|---|---|
| CNN + Maxout Kernel Blocks | 0.48% |
| CNN + Conv-Kernel Blocks | 0.51% |
| 2-Layer CNN + 2-Layer NN (Zeiler & Fergus, 2013) | 0.53% |
| Conv. maxout + Dropout (Zeiler & Fergus, 2013) | 0.45% |
| Stochastic Pooling (Zeiler & Fergus, 2013) | 0.47% |
| DLSVM (Tang, 2013) | 0.87% |

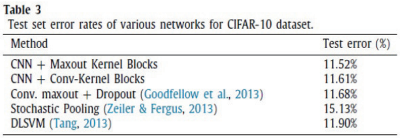


**Table 3**
Test set error rates of various networks for CIFAR-10 dataset.

| Method | Test error (%) |
|---|---|
| CNN + Maxout Kernel Blocks | 11.52% |
| CNN + Conv-Kernel Blocks | 11.61% |
| Conv. maxout + Dropout (Goodfellow et al., 2013) | 11.68% |
| Stochastic Pooling (Zeiler & Fergus, 2013) | 15.13% |
| DLSVM (Tang, 2013) | 11.90% |

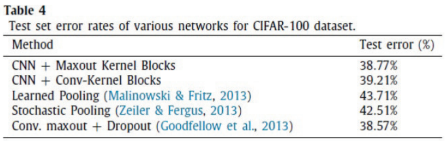


**Table 4**
Test set error rates of various networks for CIFAR-100 dataset.

| Method | Test error (%) |
|---|---|
| CNN + Maxout Kernel Blocks | 38.77% |
| CNN + Conv-Kernel Blocks | 39.21% |
| Learned Pooling (Malinowski & Fritz, 2013) | 43.71% |
| Stochastic Pooling (Zeiler & Fergus, 2013) | 42.51% |
| Conv. maxout + Dropout (Goodfellow et al., 2013) | 38.57% |

## Deep Neural Kernel Networks

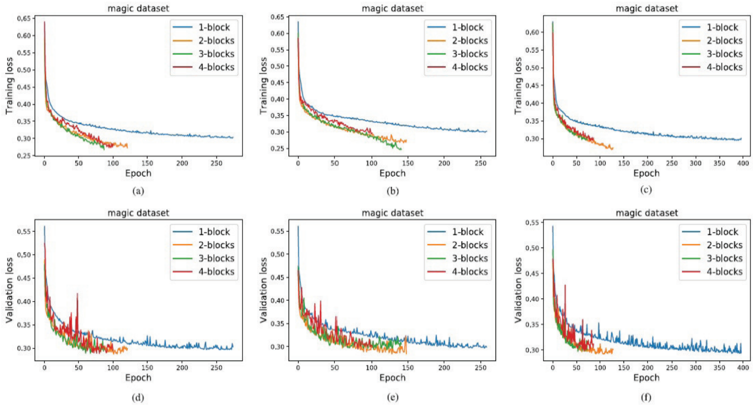


**Fig. 11.** The training and validation loss of the proposed deep maxout kernel blocks architecture with different combinations of the number of blocks and linear transformations within each block for the Magic dataset. (a,d) two linear transformations are used within each block. (b,e) three linear transformations are used within each block. (c,f) four linear transformations are used within each block.

## Cross-Domain Neural-Kernel Netwroks

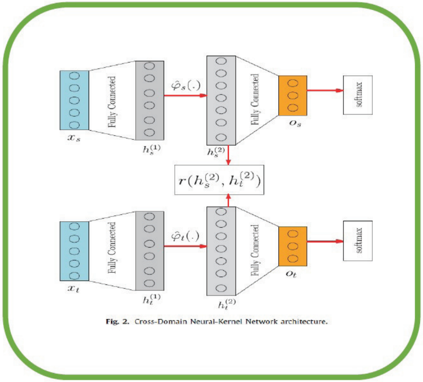
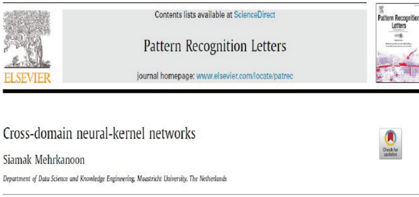


**Fig. 2.** Cross-Domain Neural-Kernel Network architecture.



Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

ELSEVIER

Cross-domain neural-kernel networks

Siamak Mehrkanoon

*Department of Data Science and Knowledge Engineering, Maastricht University, The Netherlands*

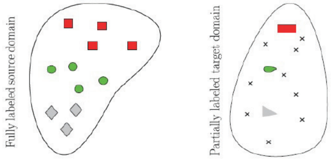


**Fig. 1.** Schematic illustration of the semi-supervised domain adaptation problem. The unlabeled target data points are shown by symbol ×.

## Cross-Domain Neural-Kernel Netwroks

$$\min_{\Theta} J(\Theta) = \gamma\Omega(\Theta) + \sum_{i=1}^{n_s} L_s(x_s^i, y_s^i) + \sum_{i=1}^{n_t^l} L_t(x_t^i, y_t^i) + \mu r(h_s^{(2)}, h_t^{(2)}).$$

$$r(h_s^{(2)}, h_t^{(2)}) = \frac{1}{2}\left(1 - \frac{{h_s^{(2)}}^T h_t^{(2)}}{\|h_s^{(2)}\|\ \|h_t^{(2)}\|}\right)$$

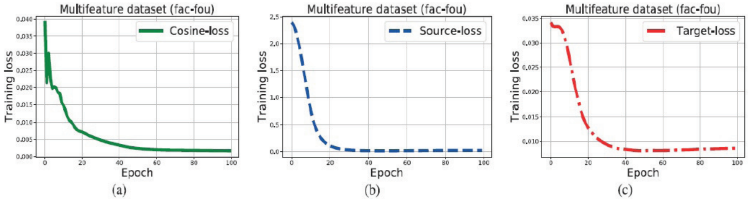


Fig. 6. Training source, target and cosine losses for the studied dataset.

## Cross-Domain Neural-Kernel Netwroks

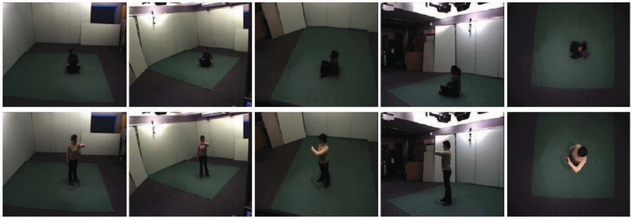


Fig. 5. Two example actions of the IXMAS dataset. **First row:** Represents 'sit down' action at f ve different domains. **Second row:** Represents 'check watch' action at f ve different domains.

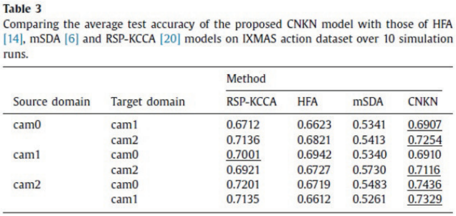


**Table 3**
Comparing the average test accuracy of the proposed CNKN model with those of HFA [14], mSDA [6] and RSP-KCCA [20] models on IXMAS action dataset over 10 simulation runs.

| | | Method | | | |
|---|---|---|---|---|---|
| Source domain | Target domain | RSP-KCCA | HFA | mSDA | CNKN |
| cam0 | cam1 | 0.6712 | 0.6623 | 0.5341 | 0.6907 |
| | cam2 | 0.7136 | 0.6821 | 0.5413 | 0.7254 |
| cam1 | cam0 | 0.7001 | 0.6942 | 0.5340 | 0.6910 |
| | cam2 | 0.6921 | 0.6727 | 0.5730 | 0.7116 |
| cam2 | cam0 | 0.7201 | 0.6719 | 0.5483 | 0.7436 |
| | cam1 | 0.7135 | 0.6612 | 0.5261 | 0.7329 |

## Conclusion

- Overview of the Multi-Class Semi-Supervised Classification/Clustering algorithm (MSS-KSC), [IEEE-TNNLS 2015].
- Overview of the Fixed-size MSS-KSC for making MSS-KSC applicable for large scale data, [WCCI-IJCNN 2014].
- Overview of the Incremental MSS-KSC model for semi-supervised classification/clustering of streaming datasets [Neural Networks 2015].
- Overview of the Regularized Semi-Paired Kernel Canonical Correlation Analysis model for Domain Adaptation Problems. [IEEE-TNNLS 2017].
- Overview of the neural-kernel networks. [Neurocomputing 2018, Neural Networks 2019, Pattern Recognition Letters 2019].

- Some other research works: Link

**Thank you for your attention!**