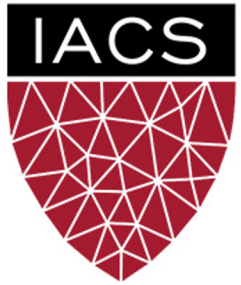
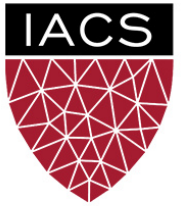


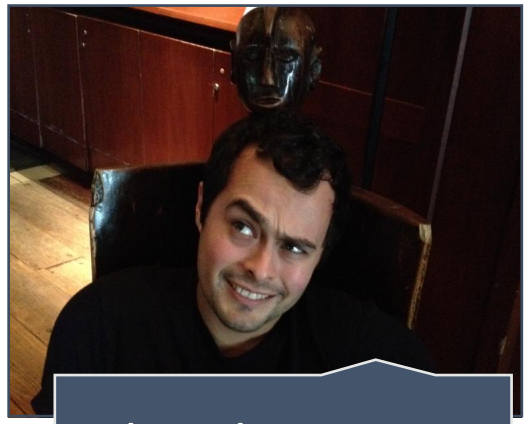
A Comprehensive Survey on Transfer Learning

$\alpha\beta normal$ Distriβution

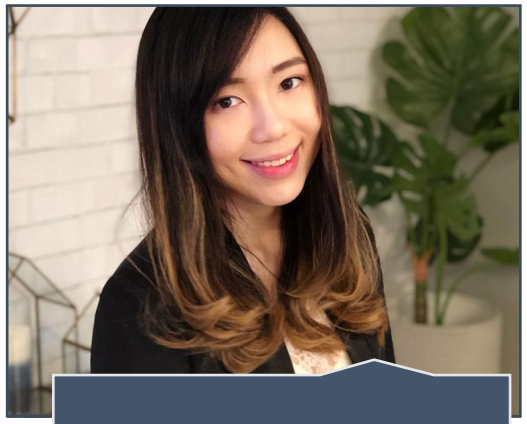
AC295/CS115



$\alpha\beta normal$ Distriβution



Eduardo Peynetti

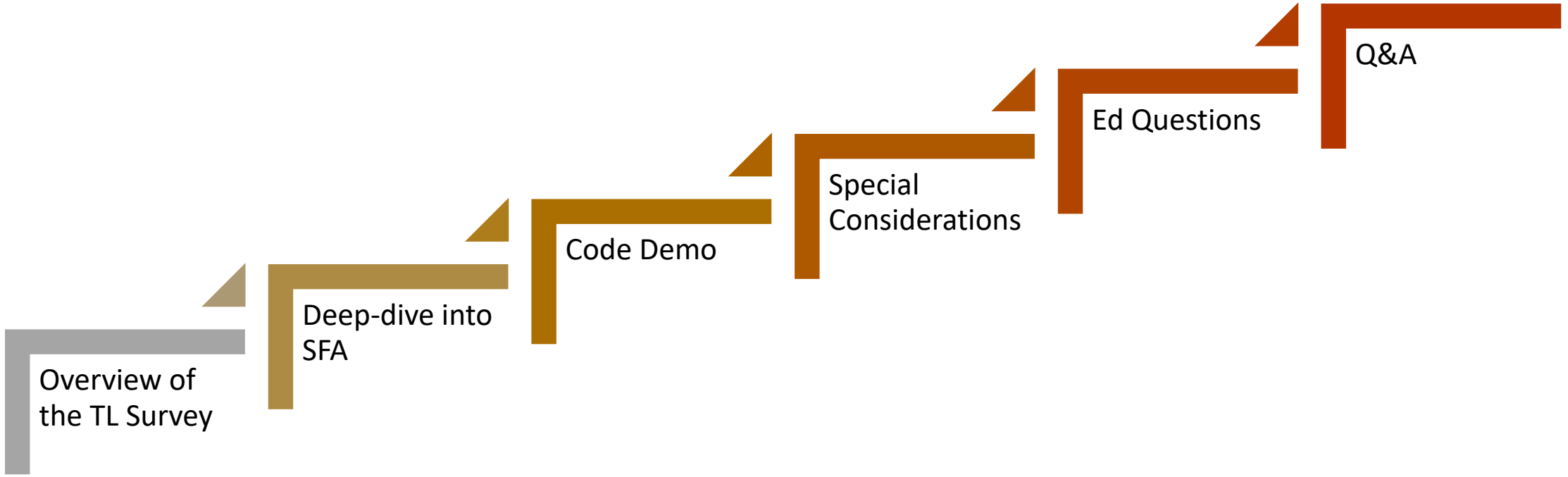


Jessica Wijaya

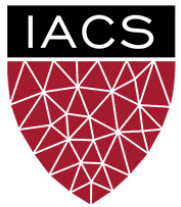
Rohit Beri

Stuart Neilson

Contents


Overview of the TL Survey
Deep-dive into SFA
Code Demo
Special Considerations
Ed Questions
Q&A

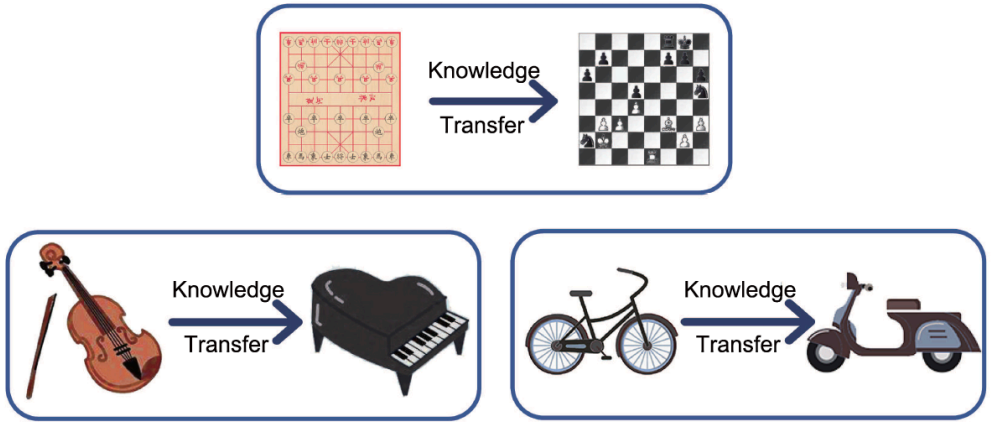
Overview of the Transfer Learning Survey (Zhuang et al.)



Transfer Learning

Generalization of Experience

- Prerequisite
  - There needs to be a connection between two learning activities



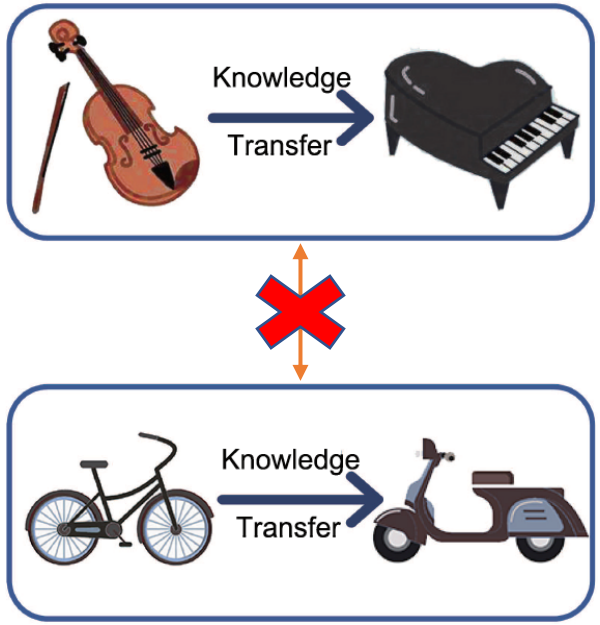
Categories of Transfer Learning

- Homogeneous TL
  - Source and Target in the same feature space
  - Differ only in marginal distributions
- Heterogeneous TL
  - Domains have different feature space
  - Requires feature space adaptation in addition to distribution adaptation
- RTL, LTL, OTL

IACS

Not Transfer Learning

CANNOT TRANSFER

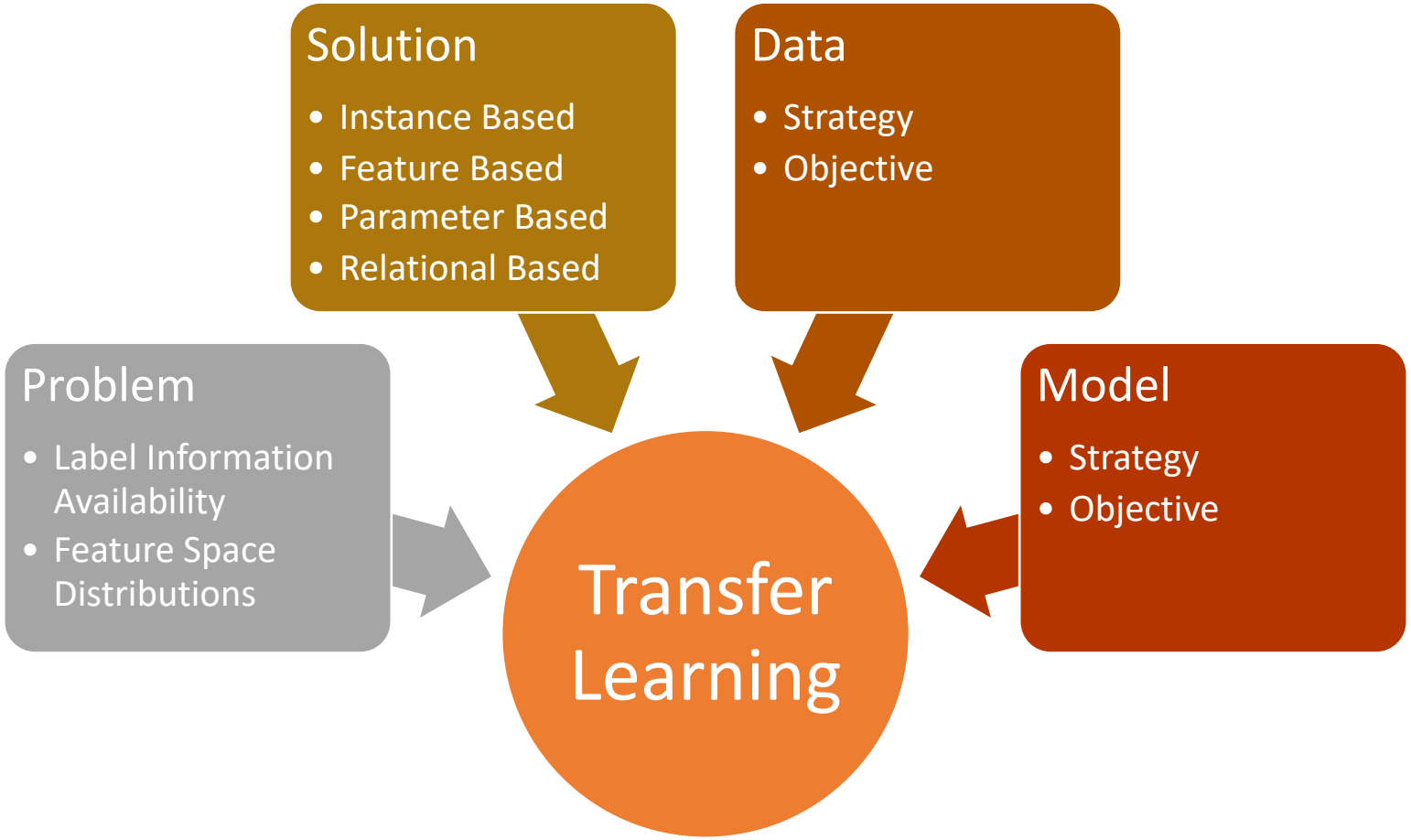


NEGATIVE TRANSFER

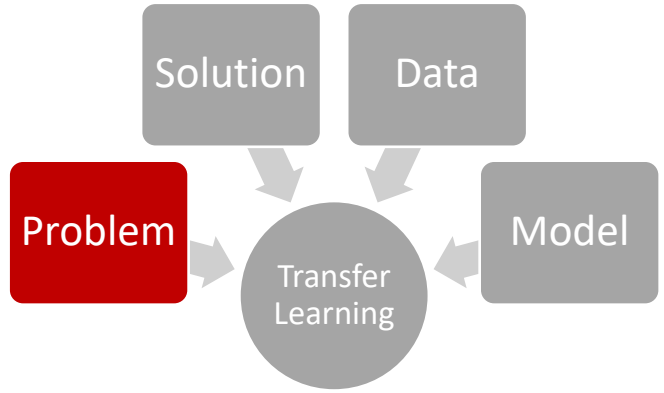
Knowing Good Spanish	Makes it difficult to learn French

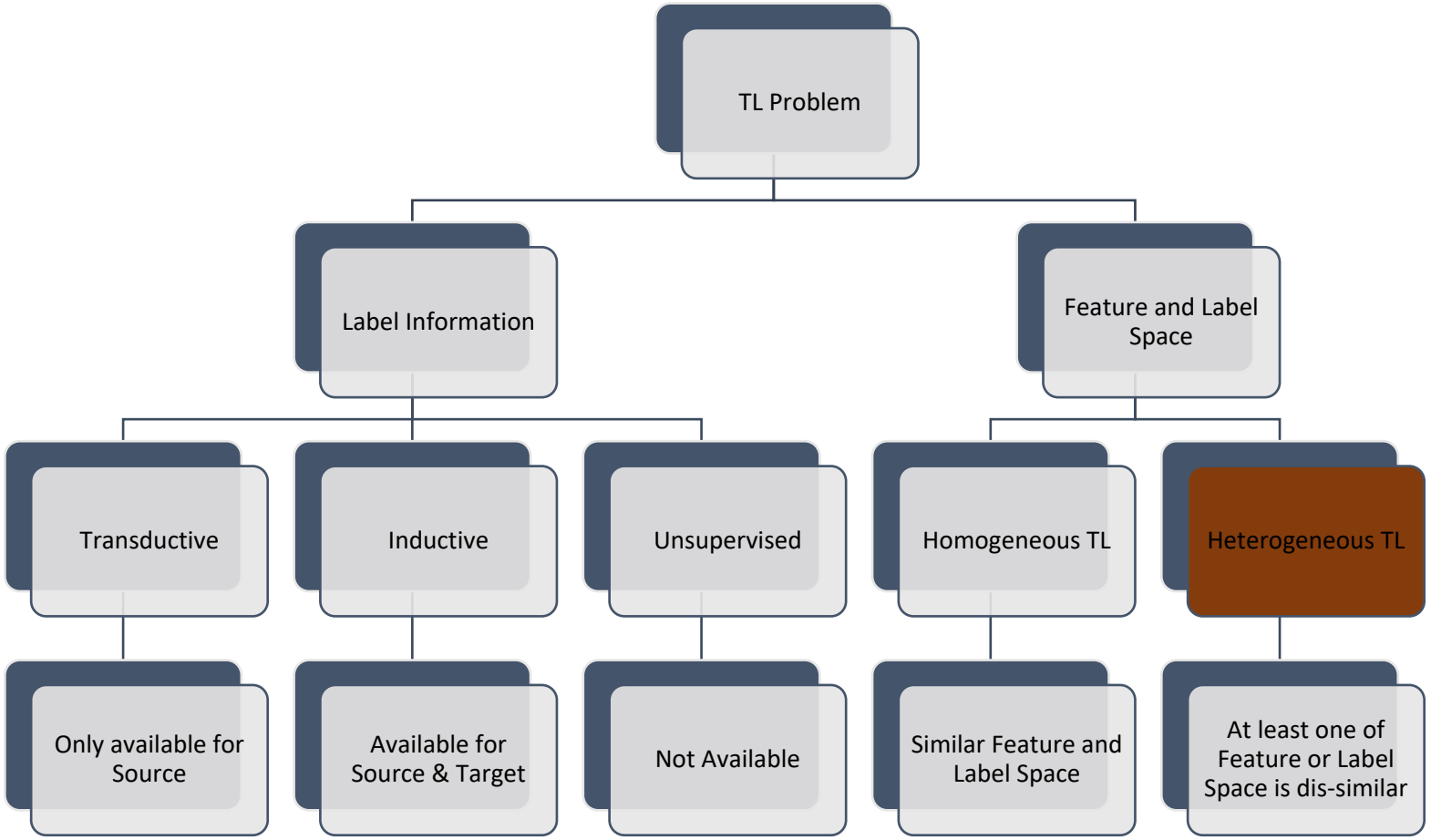
IACS

Four views of Transfer Learning



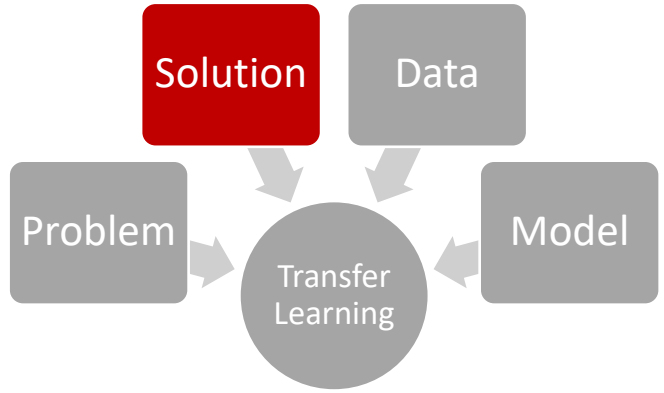
TL Problem Categorization

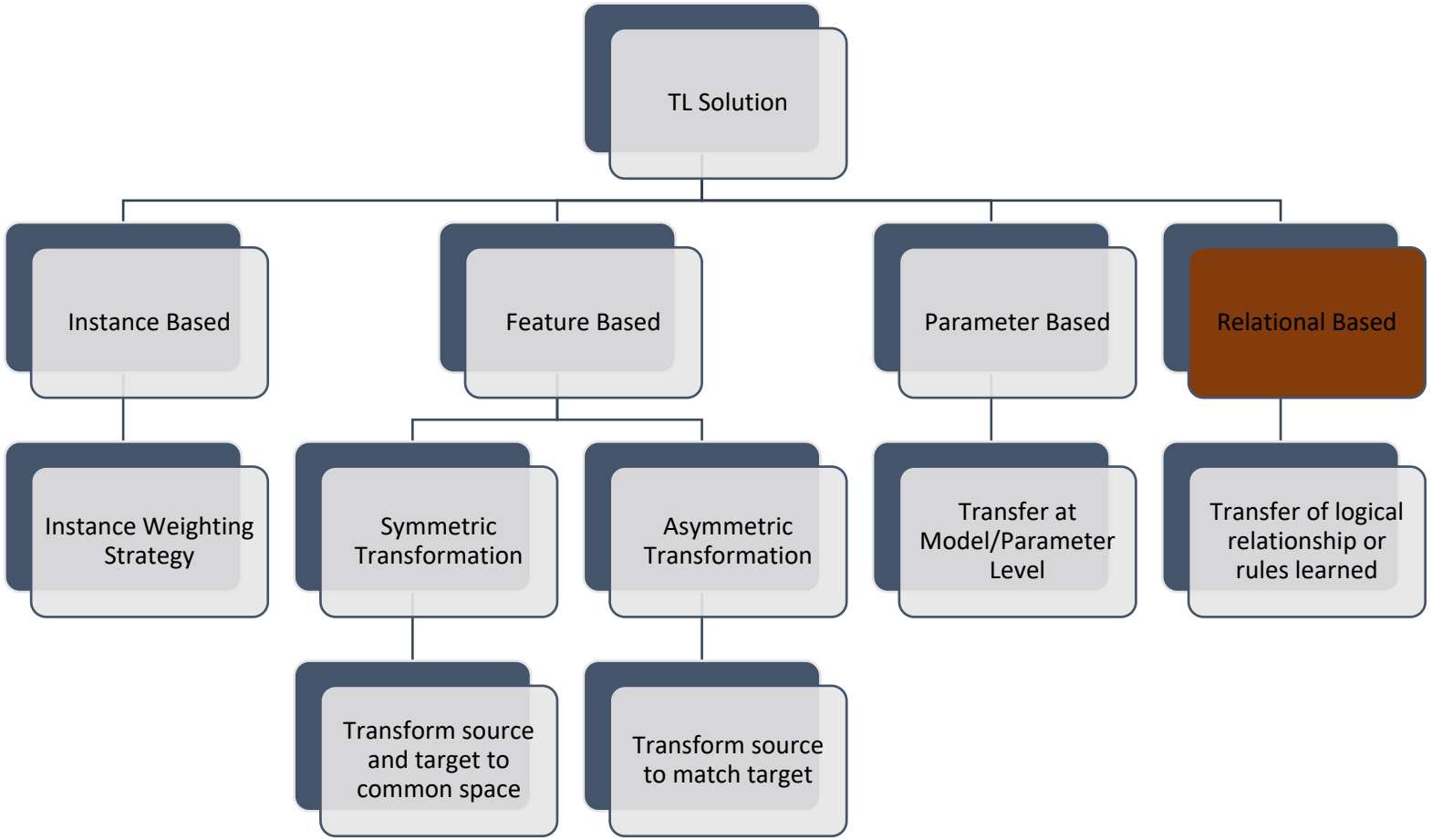




IACS

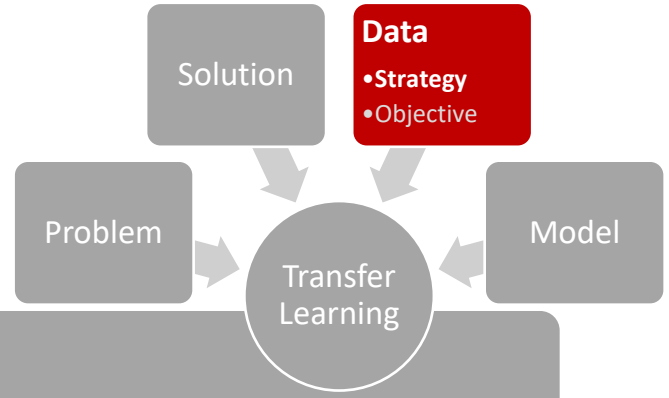
TL Solution Categorization





IACS

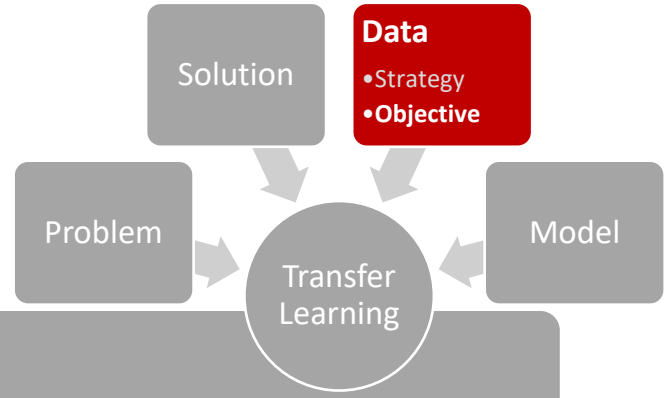
Data-based Interpretation



Strategy

- Instance Weighting
  - Estimation Method
  - Heuristic Method
- Feature Transformation
  - Augmentation
    - Replication
    - Stacking
  - Reduction
    - Mapping
    - Clustering
    - Selection
    - Encoding
  - Alignment
    - Subspace
    - Statistic
    - Spectral

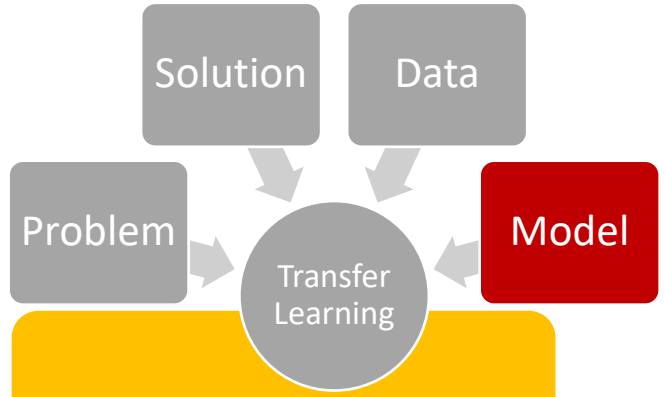
Data-based Interpretation



Objective

- **Space Adaptation**
  - Feature Space
  - Label Space
- **Distribution Adaptation**
  - Type
    - Marginal Distribution
    - Conditional Distribution
    - Joint Distribution
  - Measurement
    - Max Mean Discrepancy
    - KL Divergence
    - JS Divergence
    - Bregman Divergence
- **Data Property Preserve/Adj.**
  - Statistical Property
    - Mean
    - Covariance
  - Geometric Structure
    - Manifold
    - Cluster

Model-based Interpretation



Strategy

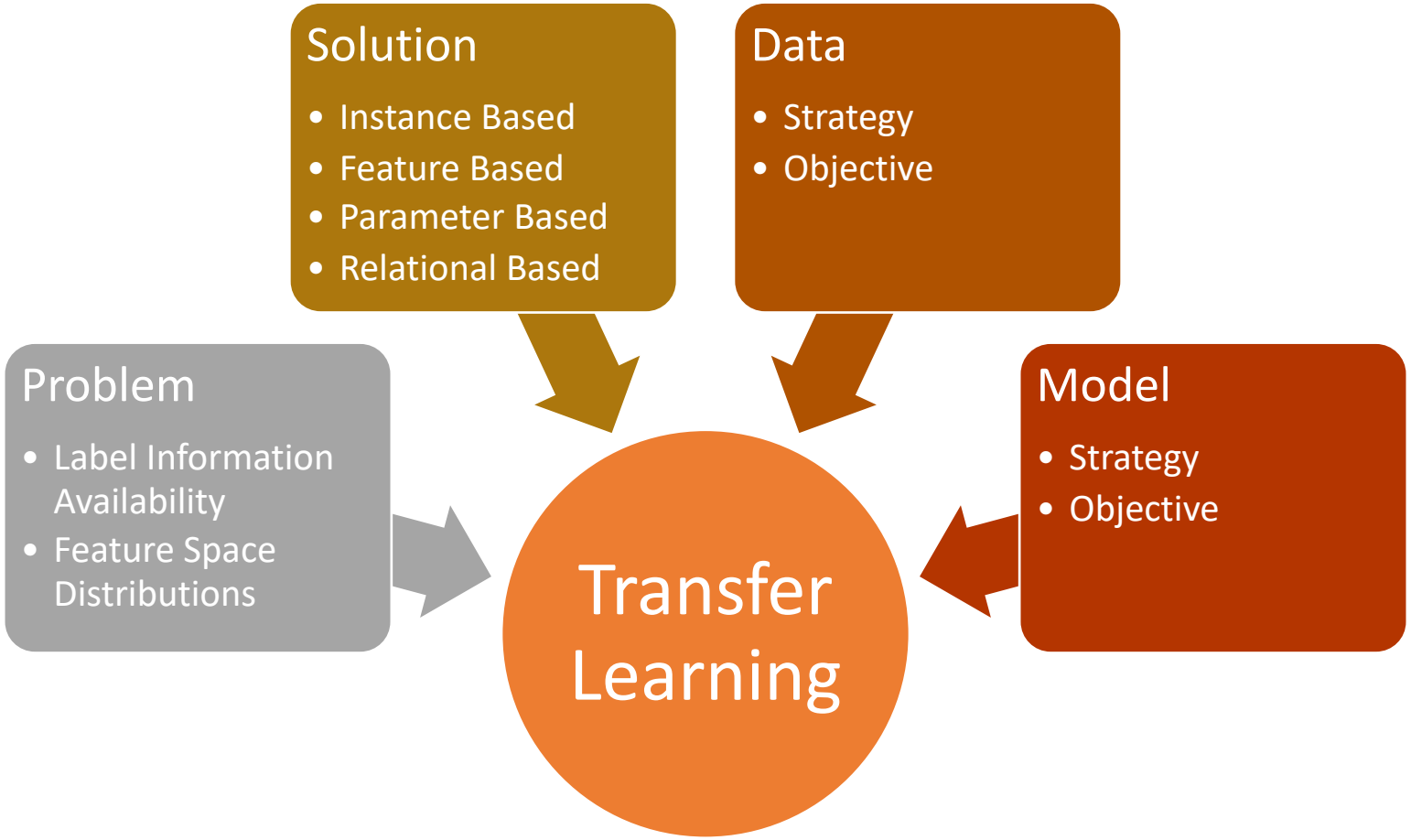
- Model Control
- Parameter Control
- Model Ensemble
- Deep Learning Techniques
  - Consensus Regularizer
  - Domain-dependent Regularizer
  - Sharing
  - Restriction
  - Voting Strategy
  - Weighting Strategy
  - Model Selection
  - Traditional DL
  - Adversarial DL

Objective

- Prediction Making
- Domain Adaptation
- Pseudo Label Generation

IACS

Four Pillars of Transfer Learning



IACS

Summary

Key Findings

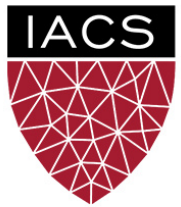
- Suitability of the algorithms is domain dependent
- Adversarial models tend to outperform others
- Co-Clustering can be powerful tool to uncover common latent space

What's missing?

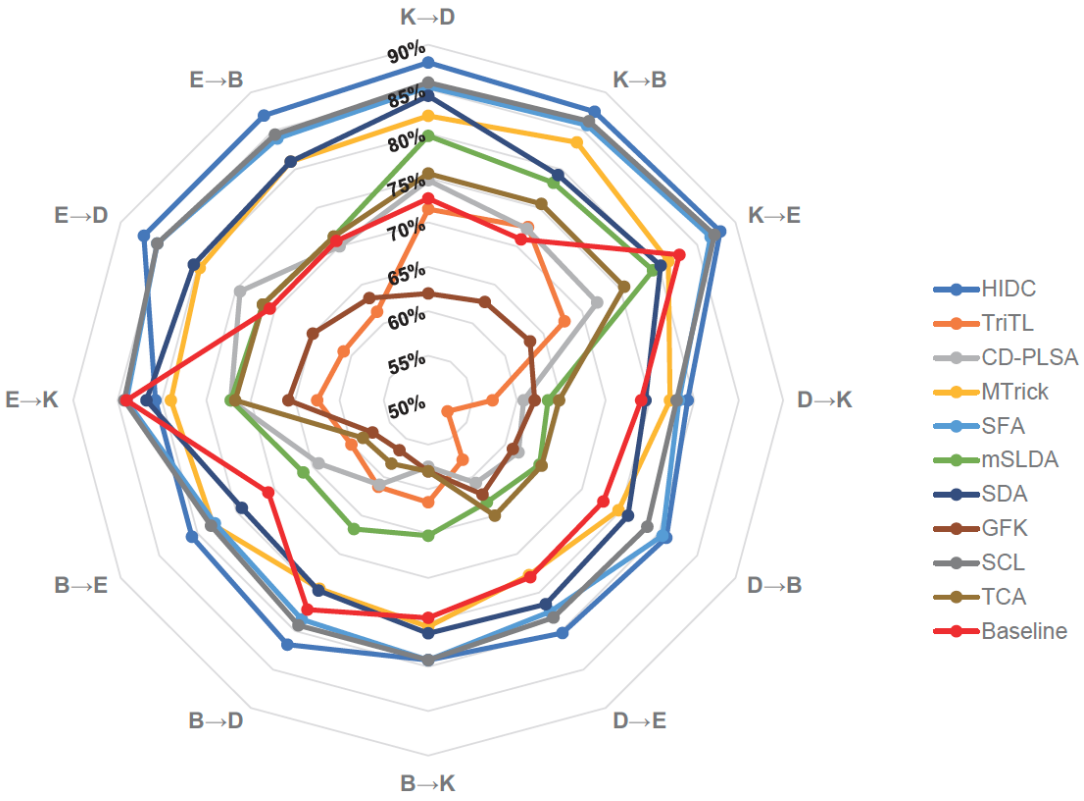
- How to avoid negative transfer?
- Theoretical underpinnings for Transfer Learning
- How to do model selection?

IACS

Deep Dive – Spectral Feature Alignment (Pan et al.)



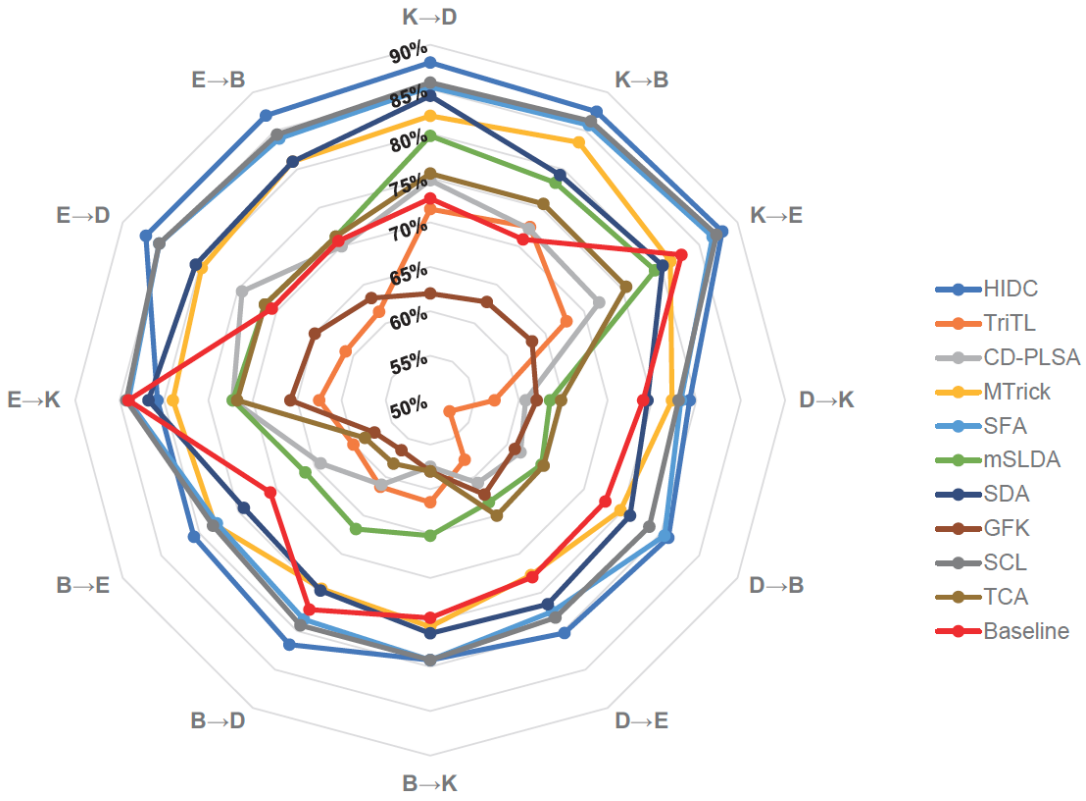
Amazon Reviews



- Models that performed well in all 12 experiments
  - HDIC: feature reduction – clustering
  - SCL: feature selection
  - SFA: feature alignment
  - MTrick: feature reduction – clustering
  - SDA: feature encoding
- Source domain: electronics or kitchen
  - All models did well
  - Domains may contain more transferable information

IACS

Amazon Reviews



- Models that performed well in all 12 experiments
  - HDIC: feature reduction – clustering
  - SCL: feature selection
  - SFA: feature alignment
  - MTrick: feature reduction – clustering
  - SDA: feature encoding
- Source domain: electronics or kitchen
  - All models did well
  - Domains may contain more transferable information

IACS

Spectral Feature Alignment

Challenge

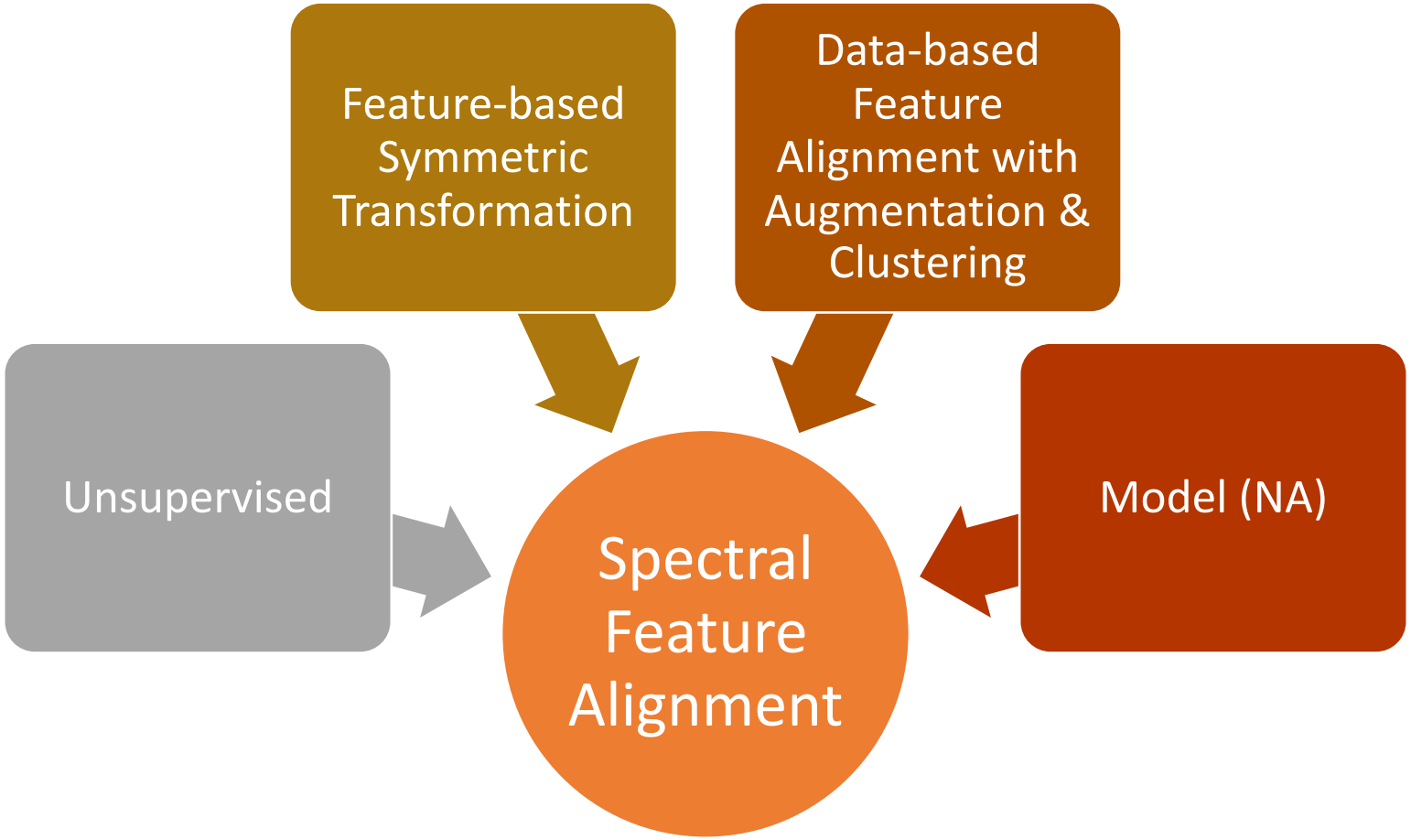
- Mismatch between domain-specific words in source and target domains
- Sentiment classifier trained on the source domain may not work well when directly applied to the target domain

Solution

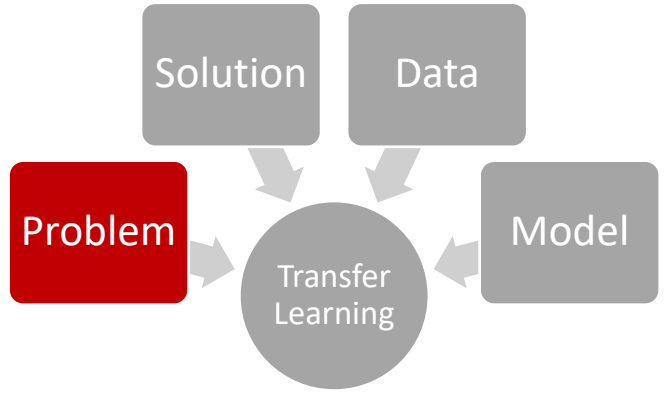
- Adapted Spectral clustering algorithm, based on graph spectral theory
- Co-align domain-specific and domain-independent words into a set of feature-clusters

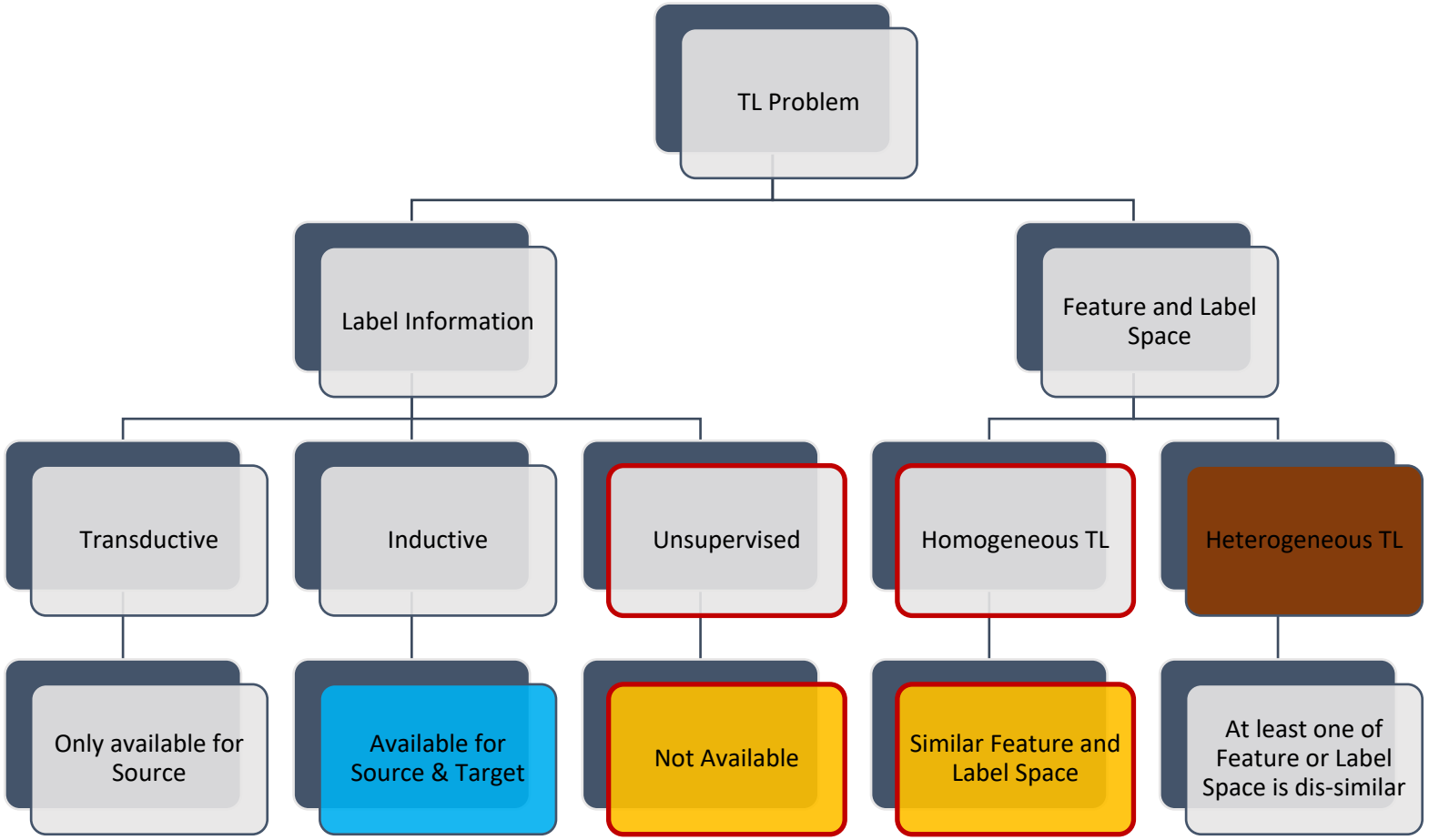
IACS

Spectral Feature Alignment



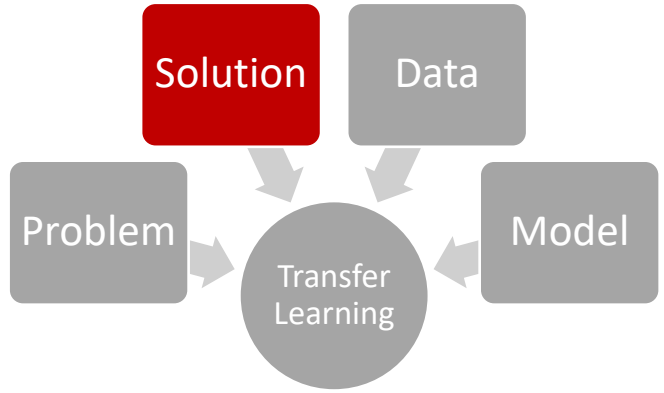
TL Problem Categorization

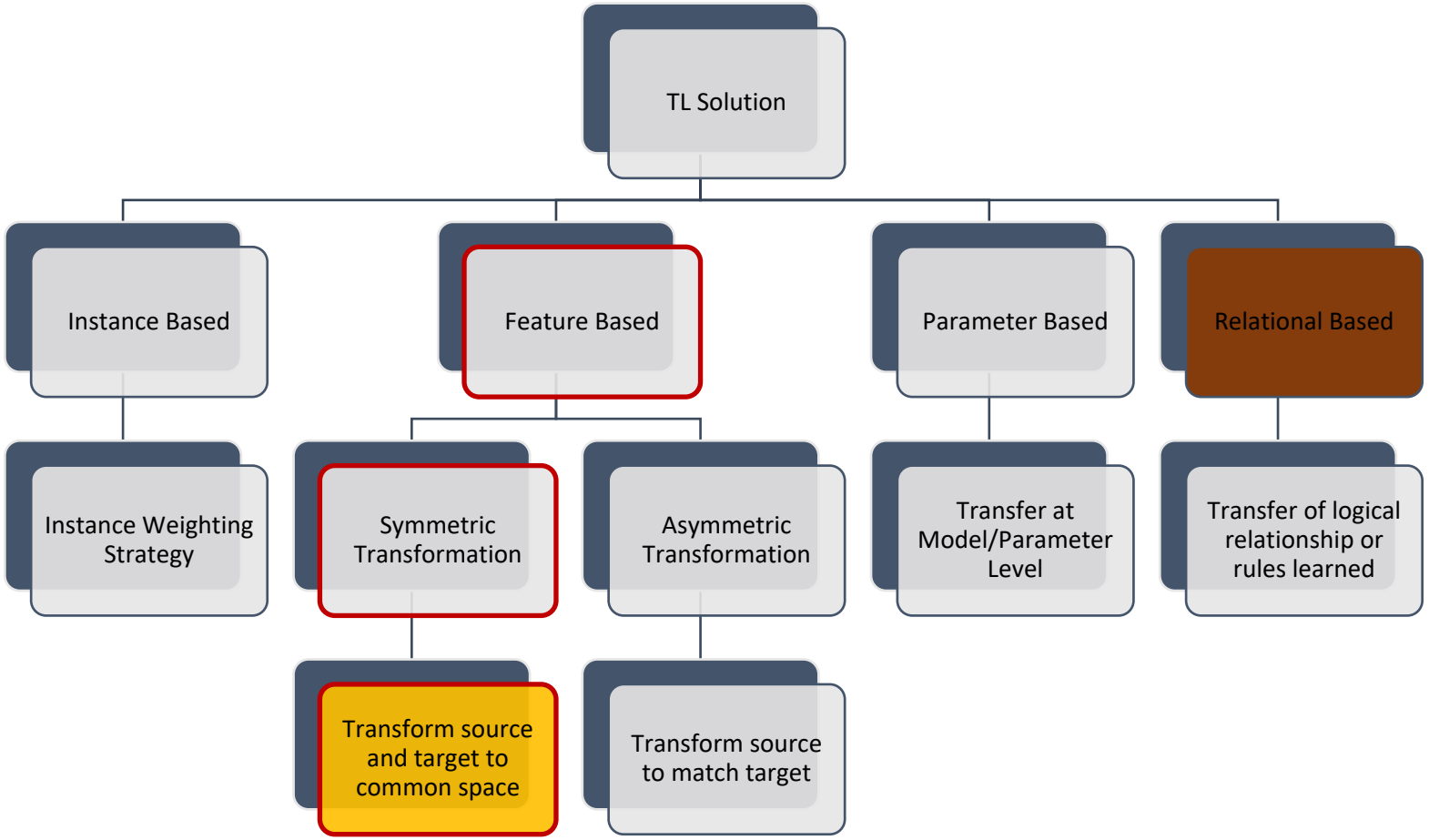




IACS

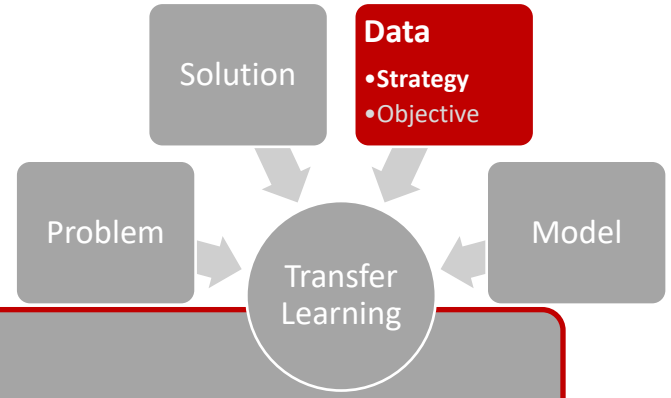
TL Solution Categorization





IACS

Data-based Interpretation



Strategy

Instance Weighting

Feature Transformation

Estimation Method

Heuristic Method

Augmentation

Reduction

Alignment

Replication

Stacking

Mapping

Clustering

Selection

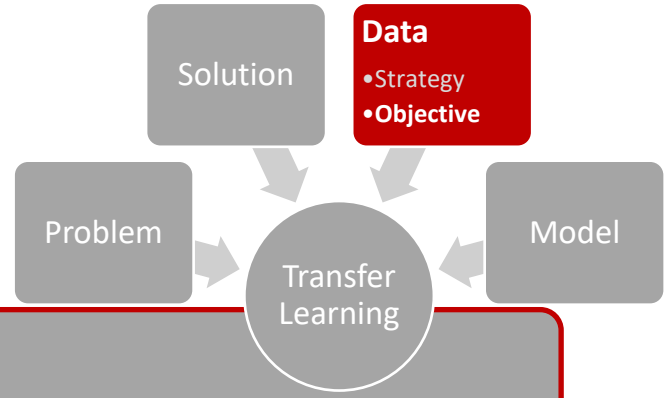
Encoding

Subspace

Statistic

Spectral

Data-based Interpretation



Objective

Space Adaptation

Distribution Adaptation

Data Property Preserve/Adj.

Feature Space

Label Space

Type

Measurement

Statistical Property

Geometric Structure

Marginal Distribution

Conditional Distribution

Joint Distribution

Max Mean Discrepancy

KL Divergence

JS Divergence

Bregman Divergence

Mean

Covariance

Manifold

Cluster

Cross-Domain Sentiment Classification

Table 1: Cross-domain sentiment classification examples: reviews of *electronics* and *video games* products. Boldfaces are domain-specific words, which are much more frequent in one domain than in the other one. Italic words are some domain-independent words, which occur frequently in both domains. "+" denotes positive sentiment, and "-" denotes negative sentiment.

	electronics	*video games*
+	**Compact**; easy to operate; very *good* picture quality; looks **sharp**!	A very *good* game! It is action packed and full of *excitement*. I am very much **hooked** on this game.
+	I purchased this unit from Circuit City and I was very *excited* about the quality of the picture. It is really *nice* and **sharp**.	Very **realistic** shooting action and *good* plots. We played this and were **hooked**.
-	It is also quite **blurry** in very dark settings. I will *never buy* HP again.	The game is so **boring**. I am extremely unhappy and will probably *never buy* UbiSoft again.

IACS

Cross Domain Sentiment Classification

What do we have?

Table 2: Bag-of-words representations of *electronics* (E) and *video games* (V) reviews. Only domain-specific features are considered. "..." denotes all other words.

		...	compact	sharp	blurry	hooked	realistic	boring
E	+	...	1	1	0	0	0	0
	+	...	0	1	0	0	0	0
	-	...	0	0	1	0	0	0
V	+	...	0	0	0	1	0	0
	+	...	0	0	0	1	1	0
	-	...	0	0	0	0	0	1

What do we want

Table 3: Ideal representations of domain-specific words.

		...	sharp_hooked	compact_realistic	blurry_boring
E	+	...	1	1	0
	+	...	1	0	0
	-	...	0	0	1
V	+	...	1	0	0
	+	...	1	1	0
	-	...	0	0	1

Cross Domain Sentiment Classification

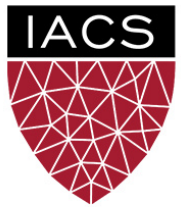
How do we get there?

- With the help of domain independent words as a bridge
- By simultaneously co-clustering them in a common latent space

Intermediate Step

Table 4: A co-occurrence matrix of domain-specific and domain-independent words.

	compact	realistic	sharp	hooked	blurry	boring
good	1	1	1	1	0	0
exciting	0	0	1	1	0	0
never_buy	0	0	0	0	1	1



Cross Domain Sentiment Classification

Domain-Independent Feature Selection

Three strategies

- Based on their frequency in both domains
- Based on the mutual dependence between features and labels on the source domain data
- Modified mutual-information criterion

Bipartite Feature Graph Construction

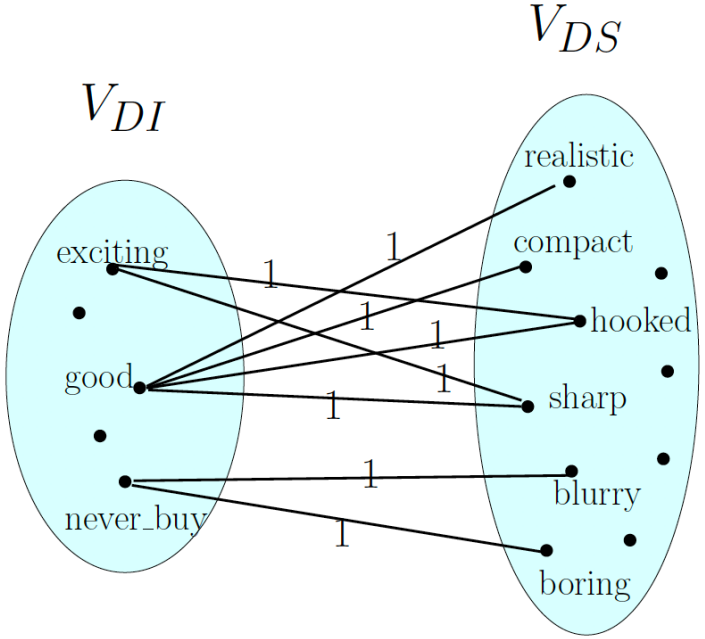
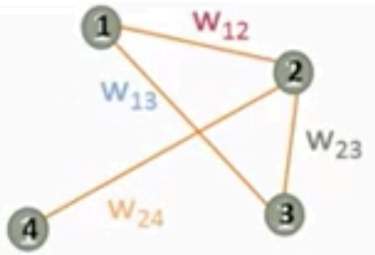


Figure 1: A bipartite graph example of domain-specific and domain-independent features based on Table 4.

Spectral Clustering

Adjacency Matrix: n x n symmetric matrix

$$A_{ij} = \begin{cases} w_{ij} & : \text{weight of edge } (i,j) \\ 0 & : \text{if no edge between } i,j \end{cases}$$



The Laplacian

$$L = D - A$$

$$d_i = \sum_{\{j|(i,j)\in E\}} w_{ij}$$

where D is the diagonal matrix of degrees

$d_1 = w_{12} + w_{13}$
$d_2 = w_{12} + w_{23} + w_{24}$
$d_3 = w_{13} + w_{23}$
$d_4 = w_{24}$

$$L_{ij} = \begin{cases} d_i & : \text{if } i = j \\ -w_{ij} & : \text{if } (i,j) \text{ is an edge} \\ 0 & : \text{if no edge between } i,j \end{cases}$$

A =

	1	2	3	4
1	0	w_{12}	w_{13}	0
2	w_{12}	0	w_{23}	w_{24}
3	w_{13}	w_{23}	0	0
4	0	w_{24}	0	0

L =

	1	2	3	4
1	d_1	$-w_{12}$	$-w_{13}$	0
2	$-w_{12}$	d_2	$-w_{23}$	$-w_{24}$
3	$-w_{13}$	$-w_{23}$	d_3	0
4	0	$-w_{24}$	0	d_4

IACS

Spectral Clustering

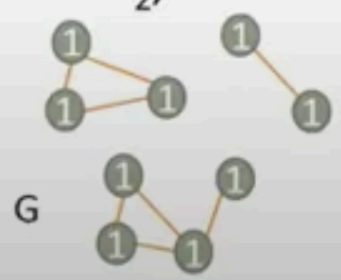
The graph Laplacian of G, L_G, also has

- eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ where $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, *and*
- eigenvectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$

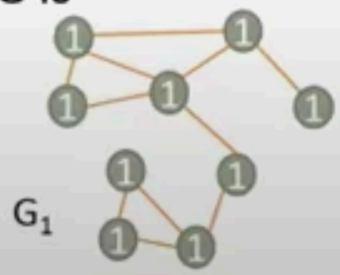
Spectrum of the Laplacian:
$0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

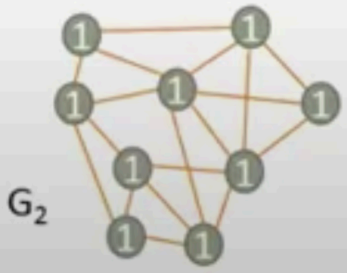
Eigenvalues reveal global graph properties not apparent from edge structure

- If 0 is the eigenvalue of L with *k* different eigenvectors, i.e., $0 = \lambda_1 = \lambda_2 \ldots = \lambda_k$, then G has *k* connected components
- If the graph is connected, $\lambda_2 > 0$ and λ_2 is the **algebraic connectivity** of G
  - The greater λ_2, the more connected G is



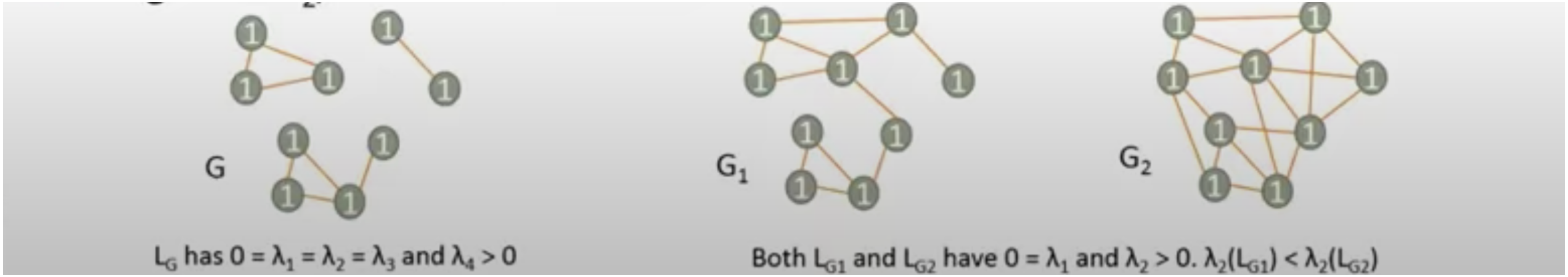
L_G has $0 = \lambda_1 = \lambda_2 = \lambda_3$ and $\lambda_4 > 0$



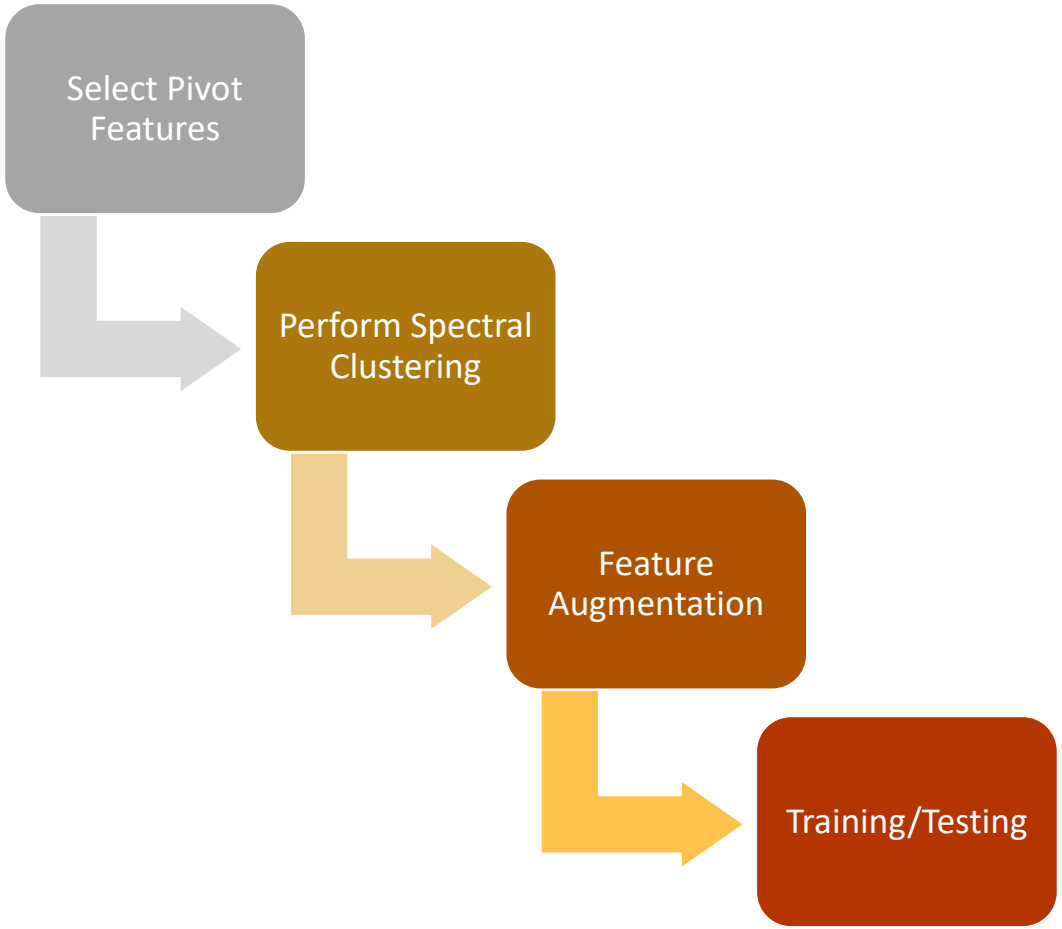


Both L_{G1} and L_{G2} have $0 = \lambda_1$ and $\lambda_2 > 0$. $\lambda_2(L_{G1}) < \lambda_2(L_{G2})$

Spectral Clustering



Spectral Feature Alignment - Algorithm



Algorithm 1 Spectral Domain-Specific Feature Alignment for Cross-Domain Sentiment Classification

Input: labeled source domain data $\mathcal{D}_{src} = \{(x_{src_i}, y_{src_i})\}_{i=1}^{n_{src}}$, unlabeled target domain data $\mathcal{D}_{tar} = \{x_{tar_j}\}_{j=1}^{n_{tar}}$, the number of clusters K and the number of domain-independent features m.

Output: adaptive classifier $f : X \to Y$.

1: Apply the criteria mentioned in Section 4.1 on $\mathcal{D}_{src}$ and $\mathcal{D}_{tar}$ to select l domain-independent features. The remaining $m - l$ features are treated as domain-specific features.
$\Phi_{DI} = \begin{bmatrix} \phi_{DI}(x_{src}) \\ \phi_{DI}(x_{tar}) \end{bmatrix}$ and $\Phi_{DS} = \begin{bmatrix} \phi_{DS}(x_{src}) \\ \phi_{DS}(x_{tar}) \end{bmatrix}$.

2: By using Φ_{DI} and Φ_{DS}, calculate (*DI-word*)-(*DS-word*) co-occurrence matrix $\mathbf{M} \in \mathbb{R}^{(m-l)\times l}$.

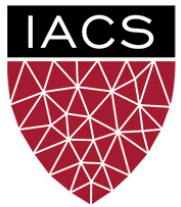
3: Construct matrix $\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$,
where $\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M}^T & \mathbf{0} \end{bmatrix}$.

4: Find the K largest eigenvectors of $\mathbf{L}$, u_1, u_2, ..., u_K, and form the matrix $\mathbf{U} = [u_1 u_2 ... u_K] \in \mathbb{R}^{m\times K}$.
Let mapping $\varphi(x_i) = x_i \mathbf{U}_{[1:m-l,:]}$, where $x_i \in \mathbb{R}^{m-l}$

5: Return a classifier f, trained on
$\{([x_{src_i}\ \gamma\varphi(\phi_{DS}(x_{src_i}))], y_{src_i})\}_{i=1}^{n_{src}}$

IACS

Code Demo



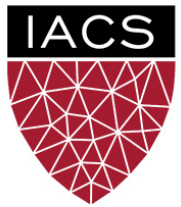
Data Parsing - Individual Data Point

```
cat = 'books'
cl = 'positive.review'

path = 'processed_acl/{}/{}'.format(cat, cl)
df = pd.read_csv(path, header=None)

df[0][2]
```

```
'woman_the:1 contains_the:1 fan_i:1 alex_ross(superman:1 justice:1 read:1 comics_fan:1 again:1 league_etc:1 fans:1 re
cieved:1 hanna-barbera!)_a:1 book_fans:1 wonder:1 gift:1 gorgeous_artwork:1 gift_and:1 contains:1 i_recieved:1 artwor
k:2 christmas:1 read_it:1 wonder_woman:1 justice_league:1 a_comics:1 again_and:1 even:1 i_read:1 the_most:2 gorgeous:
1 of_alex:1 i:2 extraordinary:1 most_gorgeous:1 most:2 it_again:1 comic_books:1 and i:1 ross(superman_batman:1 etc_ev
en:1 etc:1 the_justice:1 fan:1 beautiful:1 again.a:1 even_hanna-barbera!):1 comics:1 batman_wonder:1 for_comic:1 in_c
omic:1 artwork_in:1 books_contains:1 woman:1 a_christmas:1 extraordinary_artwork:1 books:1 christmas_gift:1 ross(supe
rman:1 league:1 artwork_of:1 most_extraordinary:1 comic_book:1 book:1 recieved_this:1 batman:1 must-have for:1 hanna-
barbera!):1 must-have:1 again.a_must-have:1 alex:1 and_again.a:1 comic:2 #label#:positive'
```

Top 20 words by frequency

'i', 'you', 'not', 'was', '', 'my', 'one', 'book', 'so', 'they', 'all', 'if', 'very', 'about', 'just', 'like', 'great', 'his', 'out', 'good',



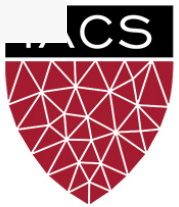
Data Parsing

Extracting Data

```
# Stopword list from NLTK
stopword_list = stopwords.words('english')
stopword_list.append('<num>')
stopword_list = list(set(stopword_list))

# Word counter for all words in dataset
word_counter = Counter()

categories = ['books', 'dvd', 'electronics', 'kitchen']
classifications = ['positive.review', 'negative.review', 'unlabeled.review']

for cat in categories:
    for cl in classifications:
        path = 'processed_acl/{}/{}'.format(cat, cl)
        df = pd.read_csv(path, header=None)
        raw_split = [[rev.split(':') for rev in df.iloc[i].values[0].split()]
                     for i in range(len(df))]
        for review in raw_split:
            for w_count in review:
                # Ignore stopwords
                if w_count[1].isdigit():# and w_count[0] not in stopword_list:
                    word_counter[w_count[0]] += int(w_count[1])

# 5000 most common words across all dataset
common_words = [word[0] for word in word_counter.most_common(5000)]
# Dictionary mapping words to index in numpy array
common_dict = {word:i for i, word in enumerate(common_words)}
```

Creating Features from Labeled Data

```
# Make feature arrays for all labeled categories

X_dict = {}

for cat in categories:
    X = np.zeros((2000,5000))
    for cl_num, cl in enumerate(classifications[0:2]):
        path = 'processed_acl/{}/{}'.format(cat, cl)
        df = pd.read_csv(path, header=None)
        raw_split = [[rev.split(':') for rev in df.iloc[i].values[0].split()]
                     for i in range(len(df))]
        for r_num, review in enumerate(raw_split):
            if r_num%500==0:
                print(r_num)
            for w_count in review:
                if w_count[1].isdigit() and w_count[0] in common_words:
                    X[cl_num*1000 + r_n No file chosen _dict[w_count[0]]] = w_count[1]

    X_dict[cat] = X.copy()
```



Data Parsing

Creating Features from Unlabeled Data

```
# Make feature arrays for all unlabeled categories

X_unlabeled_dict = {}

for cat in categories:
    cl = classifications[2]

    path = 'processed_acl/{}/{}'.format(cat, cl)
    df = pd.read_csv(path, header=None)
    X = np.zeros((len(df),5000))
    raw_split = [[rev.split(':') for rev in df.iloc[i].values[0].split()]
                 for i in range(len(df))]
    for r_num, review in enumerate(raw_split):
        if r_num%1000==0:
            print(r_num)
        for w_count in review:
            if w_count[1].isdigit() and w_count[0] in common_words:
                X[r_num, common_dict[w_count[0]]] = w_count[1]

    X_unlabeled_dict[cat] = X.copy()
```

No file chosen

Saving Features Data

```
[ ] Y=np.ones(2000)
    Y[1000:] = 0
```

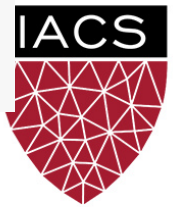
```
Y=np.ones(2000)
Y[1000:] = 0
categories = ['books', 'dvd', 'electronics', 'kitchen']
cat_initial = ['B', 'D', 'E', 'K']

for c1, i1 in zip(categories, cat_initial):
    for c2, i2 in zip(categories, cat_initial):

        if c1!= c2:
            d = [X_dict[c1], Y.copy(), X_dict[c2], Y.copy(), X_unlabeled_dict[c2]]
            pickle.dump(d, open( "{}-{}.pkl".format(i1,i2), "wb" ))
```

IACS

SFA Algorithm

```python
class SFA:
    '''
    spectral feature alignment
    '''
    def __init__(self,l=500,K=100, base_classifer=svm.SVC()):
        self.l = l # number of domain-independent features
        self.K = K # number of clusters
        self.m = 0 # number of domain-specific features
        self.ut = None # eigen-vectors from spectral decomposition
        self.gamma = 1 # tradeoff parameter
        self.base_classifer = base_classifer
        self.ix = None # index of domain-independent features
        self._ix = None # index of domain-specific features
        return

    def fit(self, Xs,Xt):

        # 1. Select domain-specific and domain-dependent features
        # ____________________________________________________

        # Sort indices by highest sum of columns (highest word frequency)
        ix_s = np.argsort(np.sum(Xs, axis=0))
        ix_t = np.argsort(np.sum(Xt, axis=0))
        ix_s = ix_s[::-1][:self.l]
        ix_t = ix_t[::-1][:self.l]

        # Instersect words with highest word frequency in both source and target
        ix = np.intersect1d(ix_s, ix_t)

        # Complement of previous index
        _ix = np.setdiff1d(range(Xs.shape[1]), ix)
        self.ix = ix # index of domain-independent features
        self._ix = _ix # index of domain-specific features
        self.l = len(ix) # number of domain-independent features
        self.m = len(_ix) # number of domain-specific features

        X = np.concatenate((Xs, Xt), axis=0)
```

```python
        # 2. Construct co-occurrence matrix between domain specific / independent
        #_____________________________________________________________________

        DI = (X[:, ix]>0).astype('float') # Domain independent
        DS = (X[:, _ix]>0).astype('float') # Domain specific

        M = np.zeros((self.m,self.l))
        for i in range(X.shape[0]):
            tem1 = np.reshape(DS[i], (1, self.m))
            tem2 = np.reshape(DI[i], (1, self.l))
            M += np.matmul(tem1.T, tem2)
        M = M/np.linalg.norm(M, 'fro')
        M = scp.lil_matrix(M)

        # 3. Create Laplacian and obtain top K eigenvectors
        #_____________________________________________________________________
        D1 = scp.lil_matrix((self.m, self.m))
        D2 = scp.lil_matrix((self.l, self.l))
        for i in range(self.m):
            D1[i,i] = 1.0/np.sqrt(np.sum(M[i,:]))
        for i in range(self.l):
            D2[i,i] = 1.0/np.sqrt(np.sum(M[:,i]))
        B = (D1.tocsr().dot(M.tocsr())).dot(D2.tocsr())
        ut, s, vt = SVD(B.tocsc(), k=self.K)
        self.ut = ut
        return ut

    def transform(self, X):
        # Feature alignment mapping function
        return np.concatenate((X, self.gamma*X[:, self._ix].dot(self.ut)), axis=1)

    def fit_predict(self,Xs, Xt, X_test, Ys, Y_test):
        # Obtained tranformed features through spectral alignment
        ut = self.fit(Xs, Xt)
        Xs = self.transform(Xs)
        # Build classifier with concatenated Xs features and new features
        self.base_classifer.fit(Xs, Ys)
        X_test = self.transform(X_test)
        y_pred = self.base_classifer.predict(X_test)
        acc = accuracy_score(Y_test, y_pred)
        return acc
```



SFA Algorithm

Training

```
datasets = ['K', 'D', 'B', 'E']
#datasets = ['K', 'D']

results = {}
for dataset1 in datasets:
    for dataset2 in datasets:
        if dataset1 == dataset2:
            continue
        [Xs, Ys, X_test, Y_test, Xt]=joblib.load(
                    os.path.join(basepath, dataset1+'-'+dataset2+'.pkl'))
        Xs = Xs.astype('float')
        X_test = X_test.astype('float')
        Xt = Xt.astype('float')
        model = SFA()
        acc = model.fit_predict(Xs, Xt, X_test, Ys, Y_test)
        print(dataset1, dataset2, acc)
        results[dataset1+dataset2] = acc
```

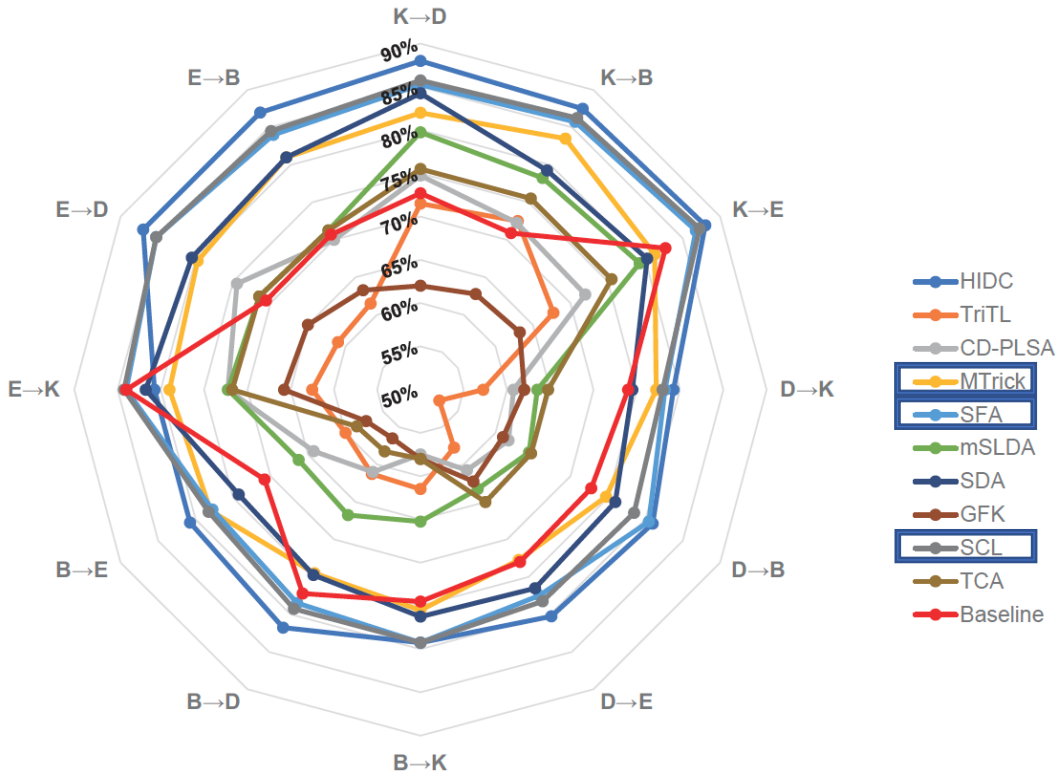
Accuracy

```
K D 0.723
K B 0.7105
K E 0.819
D K 0.7815
D B 0.7525
D E 0.7585
B K 0.787
B D 0.7705
B E 0.74
E K 0.8255
E D 0.716
E B 0.7005
```

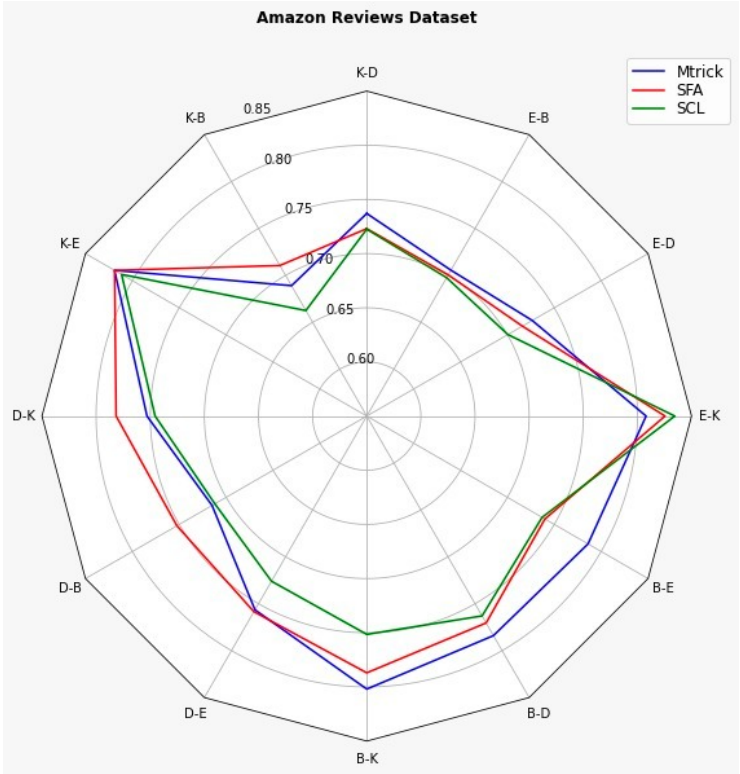
IACS

Overall Results

Survey Experiments



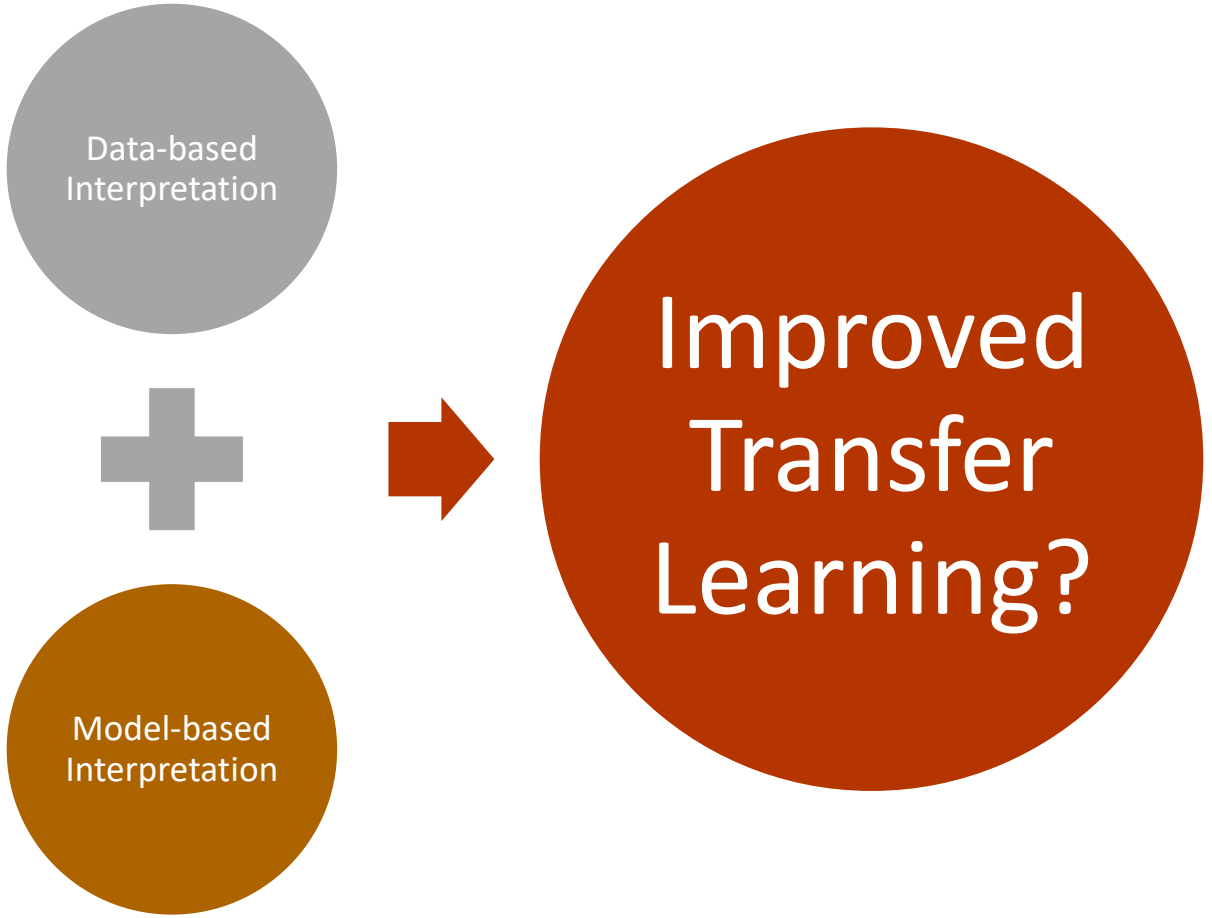
Our Experiments



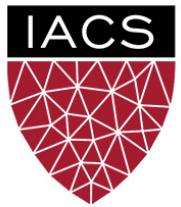
Insights

- Data pre-processing, like tokenization, can affect transferability
- Success dependent on proper choice of domain specific features
- Models in this space are similar: feature choice -> clustering -> augmentation
- Can achieve decent performance without labels

Pertinent Question



Special Considerations



TL Survey GitHub Repo

Challenges

- Part implementation in Python, part in MATLAB
- Lack of information on data preprocessing for replication
- Obsolete, sometime broken code and libraries

IACS

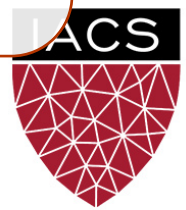
MATLAB

Why are we discussing MATLAB?

- Authors provided the code for the algorithms
- Unfortunately, some of them are written in MATLAB

How to get around it?

- Install MATLAB
- In folder `/extern/engine/python/` under the MATLAB root, run `python setup.py install`
- Need admin rights



MTrick with Python & MATLAB

Importing MATLAB in Python

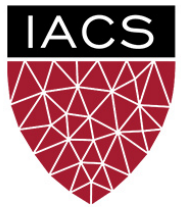
```
import numpy as np
import matlab
import matlab.engine
import scipy.io as sio
```

MTrick: Python with MATLAB backend

```
# MTrick class code taken from the Transfer-Learning-Toolkit github repo, by FuzhenZhuang

class Mtrick:
    def __init__(self, alpha=2.4, beta=2.4, numCluster=15, maxIter=200):
        self.alpha = alpha
        self.beta = beta
        self.numCluster = numCluster
        self.maxIter = maxIter
        return

    def fit_predict(self, Xs, Xt, Ys):
        inputPath = 'data.mat'
        eng = matlab.engine.start_matlab()
        sio.savemat('data.mat',{'TrainData': Xs.T, 'TrainLabel': Ys, 'TestData': Xt.T})
        Y_pred = eng.MTrick_enterFunc(self.alpha, self.beta, self.numCluster, self.maxIter, inputPath)
        eng.exit()
        Y_pred = np.asarray(Y_pred)
        Y_pred = np.reshape(Y_pred, (Y_pred.shape[1],))
        return Y_pred
```

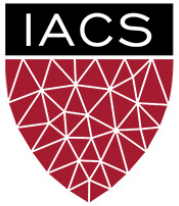
IACS

Ed Questions



TaskTrAdaBoost - Question

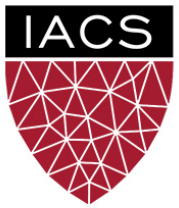
- The authors justify the use of ensemble strategies such as TaskTrAdaBoost on the grounds that combining data from multiple product domains would not be successful because the data are generated from different distributions. Why then can we use models trained on ImageNet for medical applications, given that ImageNet does not contain medical images?



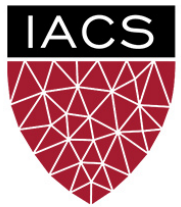
TaskTrAdaBoost – Response

- There are multiple reason for the same:
  1. Firstly, Keyword author uses is “may not work”
  2. Secondly, ImageNet to Medical images is not a multi-source domain problem
  3. Thirdly, don’t forget deep learning models do not work for every domain
  4. Finally, amount of data required for models to learn similar level of generalization as (ImageNet Models), may not be available in the source domain

- Our final thought:
  - How about combining TaskTrAdaBoost with Deep Learning Models



Questions Please!



Thanks!