TRANSFER LEARNING FOR BAYESIAN CASE DETECTION SYSTEMS

by

Ye Ye

Bachelor of Medicine, Peking University, 2006

Master of Science, Peking University, 2009

Master of Science in Public Health, Emory University, 2011

Submitted to the Graduate Faculty of

School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Ye Ye

It was defended on

November 12, 2018

and approved by

Dr. Fuchiang Tsui, Associate Professor, Department of Biomedical and Health Informatics, Children's Hospital of Philadelphia

Dr. Michael M. Wagner, Professor, Intelligent Systems Program and Department of Biomedical Informatics, University of Pittsburgh

Dr. Gregory F. Cooper, Professor, Intelligent Systems Program and Department of Biomedical Informatics, University of Pittsburgh

Dr. Jeremy C. Weiss, Assistant Professor, Health Informatics at Heinz College, Carnegie Mellon University

Dissertation Director: Dr. Fuchiang Tsui, Associate Professor, Department of Biomedical and Health Informatics, Children's Hospital of Philadelphia

Dissertation Co-director: Dr. Michael M. Wagner, Professor, Intelligent Systems Program and Department of Biomedical Informatics, University of Pittsburgh

TRANSFER LEARNING FOR BAYESIAN CASE DETECTION SYSTEMS

Ye Ye, PhD

University of Pittsburgh, 2018

In this age of big biomedical data, a variety of data has been produced worldwide. If we could combine that data more effectively, we might well develop a deeper understanding of biomedical problems and their solutions. Compared to traditional machine learning techniques, transfer learning techniques explicitly model differences among origins of data to provide a smooth transfer of knowledge. Most techniques focus on the transfer of data, while more recent techniques have begun to explore the possibility of transfer of models. Model-transfer techniques are especially appealing in biomedicine because they involve fewer privacy risks. Unfortunately, most model-transfer techniques are unable to handle heterogeneous scenarios where models differ in the features they contain, which occur commonly with biomedical data. This dissertation develops an innovative transfer learning framework to share both data and models under a variety of conditions, while allowing the inclusion of features that are unique to and informative about the target context. I used both synthetic and real-world datasets to test two hypotheses: 1) a transfer learning model that is learned using source knowledge and target data performs classification in the target context better than a target model that is learned solely from target data; 2) a transfer learning model performs classification in the target context better than a source model. I conducted a comprehensive analysis to investigate conditions where these two hypotheses hold, and more generally the factors that affect the effectiveness of transfer learning, providing empirical opinions about when and what to share. My research enables knowledge sharing under heterogeneous scenarios and provides an approach for understanding transfer learning performance in terms of differences of features, distributions, and sample sizes between source and target. The model-

transfer algorithm can be viewed as a new Bayesian network learning algorithm with a flexible representation of prior knowledge. In concrete terms, this work shows the potential for transfer learning to assist in the rapid development of a case detection system for an emergent unknown disease. More generally, to my knowledge, this research is the first investigation of model-based transfer learning in biomedicine under heterogeneous scenarios.

Table of Contents

List of Tables

List of Figures

List of Equations

Preface

The PhD study in University of Pittsburgh is an adventure for me. This journey would not have been possible without the support of my family, friends, and professors. I would like to thank my advisor, Dr. Fuchiang Tsui, who introduced me to the Intelligent System Program, guided me all the way through the adventure, and provided me great insight into research question development, study design, and software engineering. I would like to give special thanks to my cochair, Dr. Wagner, who led me towards the path of artificial intelligence for population health, believed my potential in population health informatics, and always supported me to pursue my career. I deeply appreciate the support from my committee, Dr. Cooper, who showed me the most rigorous science, and was always available for research discussions, career advice, and recommendations. I would like to thank my committee, Dr. Weiss and Dr. Day, for their supports on study design and theoretical foundations. I would like to thank the ISP and DBMI program to design the medical informatics track, which is a bridge between artificial intelligence and biomedicine. I learned theoretical foundations and scientific thinking from Drs. Weibe, Hwa, and Litman. I appreciate Dr. Becich's guidance and insights. He is a great coach for DBMI students. I enjoy working in the RODS lab and have learned a lot from John Arnois, Espino, Nick, and John Levander. Our Utah collaborators, Drs. Haug, Ferraro, and Gesteland provided massive high-quality research data, with which I get hands dirty. I would like to thank Dr. Druzdzel and BayesFusion Company to provide JSMILE engine for Bayesian inference in my study. I received a lot of support from Cleat, Howard, Toni, Michele, Lucy, Rob, and Jesse. I would like to thank Yun and Jannie for reviewing my dissertation. My friends and classmates (Yun, Kevin, Yuriy, Mahad, Huma, LingJia, Qing, Luge, Lingyun, Fan, and Zhonghui) always encouraged me, and

provided technique advice. I would especially like to thank my family, my father, Zhonghua, my mother, Yanghua, my younger sister, Yunhan, my husband, Diyang, my mother-in-law, Xiaofang, my father-in-law, Xiaoyong, and my lovely baby, Jane, for unconditional love and support.

1.0 Introduction

In the age of big biomedical data, massive amounts of digital biomedical data have been produced worldwide. If we were able to better integrate data/knowledge from all possible resources, then a deeper understanding of biomedical phenomenon would be possible.

Although many traditional machine learning technologies have been successfully used for knowledge discovery from biomedical data, the discovered knowledge may be not ready for applying to different regions, hospitals, or laboratories. Traditional machine learning technologies usually work under the assumption that the data for model development and the data for model deployment later have the same underlying distribution. This assumption is sometimes too strict if the training and test are from different regions/hospitals. The test data could have a different set of features from the training data. Even if the training and test data have a same set of features, the correlation between predictive features and class variable could be different between training data and test data. When applying a model developed with training data to the test data, their differences could lead to a dramatic performance drop.

On the other hand, there is a need to borrow/learn knowledge from other regions, hospitals, or laboratories. Retraining a model from scratch using target data could be expensive and sometimes infeasible. There may be insufficient historical data available from the target area. Also, the amount of labeled target data for supervised machine learning may be inadequate.

The goal of transfer learning is to provide a smoother transfer of knowledge from the source to the target, which is similar but not identical to the source. Transfer learning algorithms (Lu et al. 2015) consider both the similarities and the differences between the source and the target.

Transfer learning could be a key way to increase the effectiveness of worldwide disease surveillance. The ability to predict, forecast, and control disease outbreaks highly depends on the ability of disease surveillance. Compared to traditional laboratory reporting and physician reporting system, automated case detection based on electronic medical records has much less time delay and may cover a large population. However, development of an automated case detection system may require a large training dataset for modeling, but not all regions have it. One solution is that, when region A were affected by an outbreak first, it could share its developed case detection system to region B before region B was significantly affected. Since both regions cover different populations that are served by healthcare institutions with different electronic medical record systems, their features and distributions for case detection could be different. These differences usually lead to a dramatic performance drop. When region B has a few cases, with these cases, transfer learning techniques could adapt a case detection system from region A with data pattern in region B and thus increase the case detection capability for region B.

Similarly, transfer learning will help enhance the capabilities of nationwide collaborations in secondary use of electronic medical records (Chute et al. 2011), patient-centered outcome research (Selby et al. 2012), and observational scientific research (Hripcsak et al. 2015).

1.1 Hypothesis Statement

This dissertation aims to develop and evaluate a transfer learning framework for case detection using Bayesian networks over discrete variables following multinomial distributions. The developed framework (Chapter 3) is able to conduct transfer learning in two different scenarios: (1) source data and target data are available, (2) source model and target data are available.

I tested two hypotheses. Hypothesis 1: a transfer learning model that is learned using source knowledge and target data performs classification in the target context better than a target model that is learned solely from target data. This hypothesis is explored under several conditions that vary with respect to the degree of difference in features, distributions, and sample sizes in the source and the target contexts. To test this hypothesis, I compared the discriminative ability of a target model developed with the target data only, with the discriminative ability of another target model developed with both the target data and the source data (or the source model) using the proposed transfer learning algorithms. To evaluate model performance, I used a set of target instances that have not been used for model development, measuring each model's discriminative ability for the classification task.

The second hypothesis tested is that a transfer learning model learned using source knowledge and target data performs classification in the target context better than a source model that was learned in the source context. This hypothesis is explored under several conditions, including those in which (1) the source and target distribution differences are relatively large and there is a need to adjust the transferred source model, and (2) the target training data are large enough to adjust the target model to the target context. To test this hypothesis, I compared the

discriminative ability of a source model with the discriminative ability of a target model developed with both the target data and the source data (or the source model).

1.2 Additional Research Questions

To clearly define the conditions of the hypotheses and better understand the conditions where transfer learning improves over conventional machine learning methods, I explored three additional research questions: (1) how does similarity between the source and the target relate to the transfer outcomes (positive or negative)? (2) How does the target data sample size affect the effectiveness of transfer learning? (3) What should be shared, source model or source data, and when?

Question (1): How does similarity between the source and the target relate to the transfer outcomes?

Transfer learning aims to transfer knowledge from the source to the target. However, not every transfer is beneficial for the target. When the two domains are very different (e.g., few generalizable features, completely different distributions), transfer learning could be unnecessary or even harmful (i.e., negative transfer).

The study of the relationship between similarity of domains and the effect of transfer learning could provide insight into how to avoid worthless or negative transfer. In my study, similarity between the two domains is measured using two indicators: (1) the KL divergence between the distributions of the two domains (Eq. 1), which penalizes the situation when $P(i)$ is

large and $Q(i)$ is small, and (2) the proportion of overlapping features among all features (POF) from both domains (Eq. 2).

Equation 1

$$KL(\mathrm{P}||\mathrm{Q}) = \sum_{\mathrm{i}} \mathrm{P(i)} \times \frac{\mathrm{P(i)}}{\mathrm{Q(i)}}$$

where $P(i)$ and $Q(i)$ are joint probability distributions for the ith configuration of feature values.

Equation 2

$$POF = \frac{N_{f \in (F_S \cap F_T)}}{N_{f \in (F_S \cup F_T)}}$$

where F_S represents the feature set of the source data, and F_T represents the feature set of the target data.

The effect of transfer learning is indicated by the difference between the performance of a target model developed with both the target data and the source data (or the source model) and the performance of a target model developed with only the target data. If a transferred model performs significantly better than a local model in testing target instances, then the transfer is *positive*. If a transferred model performs significantly worse than a local model in testing target instances, then the transfer is *negative*. If the two models perform similarly, then the transfer is *unnecessary*.

The relationship between similarity of domains and the effect of transfer learning may depend on the number of instances of the target data. When the target has sufficient data to develop a reliable local model, transfer learning may become unnecessary even if source and target distributions are identical.

This dissertation investigates these relationships in synthetic experiments (Chapter 4) and real-world experiments (Chapter 5).

Question (2): How does the target data sample size impact the effectiveness of transfer learning?

Target models with more features and/or greater complexity usually require a larger training sample in order to learn a model. In other words, similarity between the two domains and complexity of the underlying target model impact how large a target dataset is required to learn a model. If the two domains are very different, a transferred model may not perform better than a less reliable model developed with a few target instances.

To study the effects of target sample size on the effectiveness of transfer learning, I generated synthetic datasets with different target sample sizes (Chapter 4). I also assessed different timelines in real-world datasets, which corresponds to different sizes of target training data (Chapter 5).

Question (3): What should be shared, source model or source data, and when?

Although sharing a model is more feasible than sharing data, researchers may still have concerns about information loss.

The comparison between the effects of sharing a source model and the effects of sharing source data under different transfer learning tasks (Chapter 4 and 5) varying for degree of feature space overlapping, similarities of distributions, and sample sizes provides insight into the critical questions in biomedical knowledge sharing of what to share and when.

1.3 Contributions

This dissertation develops a framework for transfer of knowledge (in the form of a model) across institution boundaries in heterogeneous scenarios (feature space differences and distribution differences). I developed an innovative score for model searching, which considers both source model information (structure and conditional probabilities) and the data pattern (correlations between features and class variables) in the target domain. The score allows for features that were measured in the target domain but not in the source domain, making the localization of a transferred model more flexible for heterogeneous scenarios.

To my knowledge, the developed algorithm is the first transfer learning algorithm for Bayesian network transfer for heterogeneous scenarios. Compared to another linear SVM based model-transfer technique for heterogeneous scenarios (Mozafari et al. 2016), my algorithm does not require relatively large target sample for target model development, while at the same time integrating much more parameter information from the source model rather than just using a one-dimensional offset parameter from the source model. It has fewer restrictions than other algorithms as well: it neither assumes linear correlations between features and the classification task, nor assumes a very similar feature distribution in a transformed common dimension. Compared to popular deep transfer learning techniques, my algorithm does not require a large number of target samples for model tuning, and it can handle the feature space difference issue that is not rare in machine learning tasks and also is very common in medical data. Compared to traditional transfer learning techniques, my algorithm does not necessarily need to use original source data, which makes model reuse an appealing alternative for knowledge sharing.

The developed algorithm can also be viewed as a new Bayesian network learning algorithm for combining knowledge and data. Compared to the classic approach (Heckerman et al. 1995),

the algorithm does not require a prior network to cover all predictive features, and it allows different levels of confidence for different features. These extensions make model learning more flexible and reliable.

I conducted a comprehensive analysis in both synthetic datasets and real-world datasets to investigate critical questions about transfer learning tasks: when to share and what to share (model or data). I recommended four measurements to estimate the success of transfer learning: percentage of shared features among all features, percentage of shared features among target features, average KL divergence, and performance of source model on target data. When the target domain has few samples, sharing the source model was found to archive a performance comparable to that of sharing the source data. When the target domain has many samples, sharing the source data was more flexible for knowledge discovery.

In addition, to my knowledge, this is the first study of model transfer using biomedical data in heterogeneous scenarios. The results demonstrate an impact of task similarity on the success of transfer learning; therefore, I recommend a well-established terminology standard and a generalizable natural language processing parser to enhance knowledge sharing for biomedical research.

Also, this is the first study on transfer learning techniques for infectious disease detection tasks. This dissertation demonstrates the transferability of case detection systems and shows the benefits of sharing in the early stage of outbreaks. Using influenza as a proxy for an emergent unknown disease, it demonstrates the possibility of quickly developing a high-performance case detection system that uses natural language processing to extract clinical findings (features) from routinely collected emergency room reports.

1.4 Dissertation Organization

The rest of the dissertation is organized as follows. Chapter 2 provides transfer learning and offers a review of transfer learning techniques in two categories: data transfer (instance weighting, feature representation, self-labeling, and hyper-parameter), and model transfer. Chapter 2 then describes the design of my Bayesian case detection systems. Chapter 3 explains the developed transfer learning framework, including description of Bayesian transfer learning using the source data algorithm (BTLSD) and Bayesian transfer learning using the source model algorithm (BTLSM). Chapter 4 presents the experimental results of the proposed algorithms on synthetic datasets and Chapter 5 presents the experimental results on real-world datasets. Chapter 6 summarizes the research findings and discusses the dissertation hypotheses in light of the results obtained. Finally, Chapter 7 summarizes the contributions and discusses future lines of research.

2.0 Background

This Chapter provides an introduction of Bayesian case detection system, Bayesian network learning, and current transfer learning techniques.

2.1 Bayesian Case Detection System

The control of epidemic diseases is an increasingly important global problem. The spread of an infectious disease could be very fast in a dense population with low immunity levels. The increasing incidence of domestic and international travel further hastens the speed of contamination. In in multi-region epidemics, efficient and effective infectious disease control strategies strongly rely on the capability of disease surveillance, which must be amenable to widescale rapid deployment.

Traditional case detection mainly relies on notifiable disease reporting and sentinel physician systems; however, issues with time delay and underreporting often occur. As a result, substantial investment and research has been put into public health to include automated surveillance that leverages routinely collected electronic information such as laboratory test orders and results (Panackal et al. 2002, Overhage et al. 2008), chief complaints (Ivanov et al. , Espino et al. 2001, Wagner et al. 2004, Chapman et al. 2005), sales of over-the-counter medications (Wagner et al. 2004, Rexit et al. 2015), and encounter notes (Elkin et al. 2012, Gerbier-Colomban et al. 2013).

Since 2011, our research lab has been developing a type of Bayesian case detection system (BCD) as part of a probabilistic framework for case and outbreak detection (Tsui et al. 2011, Wagner et al. 2011, Cooper et al. 2015). This system uses natural language processing (NLP) to infer the presence or absence of clinical findings from narrative notes. With these findings, a Bayesian network model infers each patient's diagnosis probabilities. The BCD also provides the likelihood of patient clinical evidence supporting outbreak detection and prediction at the population-level. We chose Bayesian network models because they separately represent prior probability of the diagnosis (*i.e.*, *P*(*diagnosis*)) and likelihood of clinical findings (*i.e.*, *P*(*clinical findings* | *diagnosis*)). This representation allows our outbreak detection algorithms (Cooper et al. 2015) to use dynamic priors for the *diagnosis* node, which reflects the changing prevalence of disease during an outbreak.

The initial focus of the disease surveillance was influenza, which was fielded in Allegheny County, PA in 2009. We showed that our influenza case detection system performed well in the location in which it was built (Tsui et al. 2011, López Pineda et al. 2015). For the input of a case detection system, another researcher, Elkin first showed that using whole encounter notes was more accurate for case detection than using only the chief complaint field (Elkin et al. 2012). Ruiz's study found the benefit of using multiple clinical notes associated with an encounter (Ruiz 2014). For the NLP component of the case detection system, I compared performance of BCD using human annotated findings with performance using NLP extracted findings, and found that the latter led to a drop in AUC of about 0.06 (Ye et al. 2014).

We also conducted three studies that analyzed the Bayesian network model component. The first study (Ye et al. 2014) showed that feature selection increases performance. Specifically, models using a subset of findings had better performance than models using a full set of findings.

Although the full set of findings had been defined by experts, NLP extraction changed the strength of associations between many clinical findings and diagnosis. The second study (López Pineda et al. 2015) showed that machine learning alone was as good as a combination of expert knowledge and machine learning, thus providing support that we might be able to eliminate a labor-intensive development step. The third study (Tsui et al. 2017) showed that using the actual dynamic prior of diagnosis could increase the discriminative ability of an influenza detection model comapred to using a constant diagnosis prior.

Overall, effective and efficient management of disease outbreaks would benefit from automated case detection systems such as those described above that can be rapidly deployed across institutional and geographical boundaries. For example, my recent research about transferability demonstrated high influenza case detection performance in two large healthcare systems in two geographically separate regions, the University of Pittsburgh Medical Center (UPMC) and the Intermountain Healthcare (IH) System in Utah, providing support for the use of automated case detection from routinely collected electronic clinical notes in national influenza surveillance (Ye et al. 2017). In addition, I identified the influence of natural language processing on transferability. However, for a BCD developed with IH training data experienced reduced performance. We attribute this to the lower recall of an IH natural language processing parser on the UPMC notes, from which the section-specific rules in IH parser failed to extract influential findings.

These results indicate that case detection using NLP-extracted clinical findings may encounter transferability challenges. NLP-extracted findings of clinical notes from different institutions may have different feature sets with different distributions. Most medical NLP parsers (Friedman et al. 2004, Harkema et al. 2009, Savova et al. 2010) rely on rule-based processing

engines, which rules are usually manually built by knowledge engineers based on expert-annotated sample clinical notes from one institution. It is difficult to automatically refine these rules for another institution. Moreover, different institutions may have different templates in their electronic medical record systems and their clinicians may also write notes differently. Therefore, it is very common that NLP-extracted findings of clinical notes from different institutions have different feature sets with different distributions. Transfer learning of Bayesian network models could thus be helpful.

2.2 Bayesian Network Learning

To increase the transferability of case detection systems, my dissertation study focuses on transfer learning for Bayesian network models. The Bayesian network models represent the uncertainty of a disease using probabilistic graphic models. Typically, the structure of the models represents the correlations between clinical findings and diseases. Their strength is represented by a set of conditional probability tables. Both the structure and the parameters of a Bayesian network model can be automatically induced from data by machine learning algorithms.

Learning a Bayesian network model usually involves both structure learning and parameter learning. Learning Bayesian network structures is NP-hard (Chickering et al. 1994). Three strategies have been used for machine learning: a score-and-search approach, a constraint-based approach, and a dynamic programming approach (Daly et al. 2011), which is a score-based approach without search procedure.

The score-and-search approach uses a scoring measure and a search procedure to find the network or set of networks that is the most supported by the data and possible background

knowledge. Many score-based structure-learning algorithms estimate parameters as part of the scoring process. When calculating a score, an implicit parameterization is always given. The parameter learning is usually a subroutine in structure learning. For Bayesian scores, there is usually not a single parameterization, but rather all possible parameterizations are considered by integrating over them and using a parameter prior while doing so.

The constraint-based approach constructs a dependency structure based on conditional independencies identified by statistical tests on the data. While the score-and-search approach usually works better with less data and with probability distributions with dense graphs and is able to represent probability distributions easily (Daly et al. 2011), the constraint-based approach is typically quicker and is good at modeling hidden common causes and selection bias (Daly et al. 2011).

The dynamic programming approach uses dynamic programming to compute optimal models for a small set of variables (no larger than 30) (Daly et al. 2011). It is similar to the score-and-search approach but performs an exhaustive search.

My proposed transfer learning algorithms uses the Bayesian Dirichlet scoring function and a local search approach, described below.

2.2.1 Bayesian Dirichlet Scoring Functions

Bayesian Dirichlet scoring functions usually compute the relative posterior probability of each candidate network-structure hypothesis given data and prior knowledge, assuming multinomial distributions on the data, parameter independence, parameter modularity, a Dirichlet distribution to represent the parameter priors, and complete independent and identically distributed data.

The Parameter independence assumption includes global and local independence. Global independence means that parameters associated with each node in a network are independent, while local independence means that parameters associated with each state of the parents of a node are independent. With these independence assumptions, I can transform the probability density of all parameters into the multiplication of the individual probability density of each parameter that is associated with each state of the parents of a node. These independence assumptions make the score decomposable: the score of a network structure is the product of the score of each node given its parents.

The parameter modularity assumption assumes identical probability density functions of parameters associated with a node in two distinct networks if the node has the same parents in those two networks.

The assumptions of parameter independence and parameter modularity make the comparison between two different network structures more efficient, enabling a focus solely on those nodes that have different parents in two network structures.

The Dirichlet assumption means that prior parameters associated with each state of the parents of a node follow a Dirichlet distribution.

Under these assumptions, the joint probability of a network structure B_S^h and data D can be calculated as shown in Eq. 3 (Heckerman et al. 1995):

Equation 3

$$p(D, B_S^h | \xi) = p(B_S^h | \xi) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})}$$

The term N'_{ijk} denotes prior parameters of the Dirichlet distribution, where the prior probability of one configuration of parameters is the product of these parameters powered by the corresponding N'_{ijk}, $\rho(\Theta_{ij}|B_S^h, \xi) = c \prod_k \theta_{ijk}^{N'_{ijk}-1}$. The term N'_{ijk} and $p(B_S^h | \xi)$ completely specify

a user's prior knowledge about the model. For the special case when N'_{ijk} =1, the BD measure is the K2 measure (Cooper et al. 1992).

Another example of a BD measure is the Bayesian Dirichlet likelihood equivalent (BDe) measure (Heckerman et al. 1995). The BDe measure is a likelihood-equivalent Bayesian scoring metric. The likelihood-equivalence assumption assumes that network structures representing the same assertions of conditional independence have the same likelihood. The assumptions of likelihood equivalence and parameter independence imply that the parameter priors must follow a Dirichlet distribution (Heckerman et al. 1995).

The definition of the BDe measure is given in (Heckerman et al. 1995): Given a domain U of n discrete variables $x_1, \ldots, x_n$, suppose that prior distribution $\rho(\Theta_U|B_{SC}^h, \xi)$ is Dirichlet with prior equivalent sample size N' for network structure B_{SC} in U. Then, for any network B_S in U, assumptions of a multinomial sample, parameter independence, parameter modularity, complete data, and structure possibility, imply $N'_{ijk} = N'\mathrm{p}(x_i = \mathrm{k}, \pi_i = \mathrm{j}|B_{SC}^h, \xi)$. In situations with uniform joint distribution constraint (i.e., all configurations have same probabilities), N'_{ijk} can be calculated as $\frac{N'}{r_i q_i}$. The BDe with this assignment is called BDeu (Bayesian Dirichlet with likelihood equivalence and a uniform joint distribution) (Daly et al. 2011).

Since the variance of the parameters is proportional to $1/N'$, the prior equivalent sample size reflects a user's confidence for the prior network. Thus, N' can be assessed as the number of observations that would have been seen in order to have the same confidence as a prior knowledge. Using simulation data generated by a gold-standard network, (Heckerman et al. 1995) studied the behavior of a learning Bayesian network initiated using a prior network, and found that the optimal equivalent sample size for a prior network decreases as the difference in distributions between the prior network and the gold-standard network increases.

2.2.2 Search Procedure

For tree-like Bayesian networks (every node has at most one parent), the search for the network structure with the highest score is a "finding maximum branching" problem, which can be solved in polynomial time (Karp 1971, Gabow et al. 1984). When the score assumes likelihood equivalence, a maximum spanning tree algorithm can identify the undirected forest with the highest score, followed by adding any directionality to the arcs to obtain a collection of equivalent network structures.

When allowing some nodes to have more than one parent, the search for the network structure with the highest score is NP-hard (Heckerman et al. 1995). One simple heuristic search algorithm is a local search, which makes the most valuable change (measured by score) at each move in the search process until no change leads to an increase of score. The local search approach is relatively fast when using a decomposable measure (e.g., BD metric), because this type of measure allows us to avoid re-computing all terms after every change. One potential problem with a local search is that it may get struck at a local maximum. To avoid this problem, I can use an iterated local search or simulated annealing. Compared to random initiation, prior structure knowledge is found to be more useful for initiating a local search (Heckerman et al. 1995).

2.2.3 Some Bayesian Network Learning Algorithms

2.2.3.1 Naive Bayes

The Naïve Bayes algorithm assumes conditional independence of predictive nodes given the class node. This assumption dramatically simplifies the structure of a Bayesian network: arcs only connect between predictive nodes and the class node. This assumption also dramatically

reduces the number of parameters (i.e., conditional probabilities) from exponential to linear in the number of nodes.

The conditional probabilities of a Naïve Bayesian network can be estimated using either maximum likelihood estimates or maximum a posterior (MAP) estimates (Mitchell 1997). One potential problem with the maximum likelihood approach is that the estimated conditional probability will be 0 if a particular event does not appear in the training data. This is common when the training data has a small sample size. The MAP approach uses prior distributions to smooth the parameter estimation. Eq. 4 shows Laplace smoothing.

Equation 4

$$P\,(X_j \;=\; x_{jk} \mid Y \;=\; y_m) \;= \frac{\sum I(X_j = x_{jk},\; Y = y_m) + 1}{\sum I(\,Y = y_m) + r_j}$$

The performance Naïve Bayes algorithm has been shown to be comparable to much more complicated models (Domingos et al. 1997). Although the traditional Naïve Bayes learning algorithm does not conduct any feature selection, it is usually very robust against overfitting in terms of discrimination. However, it is sensitive to strong correlations between features that violate the conditional independence assumption. Inference on a Naïve Bayesian network is computationally easy, but the posterior probabilities are usually not well calibrated: they are too close to 0 or 1 (Zadrozny et al. 2001).

2.2.3.2 K2

The Naïve Bayes algorithm has a conditional independence assumption, which may not be able to represent the complicated correlations between variables in biomedical data. After relaxing this assumption, in order to find the most probable network structure in a polynomial time, many Bayes learning algorithms use some restrictions and assumptions to reduce the search space.

The K2 algorithm (Cooper et al. 1992) assumes a uniform prior over structures, and requires input of an ordering of the candidate variables and the maximum number of possible parents. Starting from an empty Bayesian network, the algorithm incrementally adds the parents of nodes as long as the addition increases the K2 score. Candidate parents of a node are restricted by the ordering of the candidate variables. Only variables proceeding a variable in the ordering can be considered as candidates of parents. The number of parents of a node is also restricted by the maximum number of possible parents.

The complexity of the K2 algorithm is O (mn^4r), where m is sample size of the training data, n is the size of the feature space, and r is the maximum number of possible values for any variable.

2.2.3.3 Efficient Bayesian Multivariate Classification Algorithm

The above two algorithms do not conduct feature selection. The efficient Bayesian multivariate classification algorithm (EBMC) (Cooper et al. 2010) was developed to efficiently identify a Bayesian network that predicts a target variable.

EBMC conducts a greedy forward-stepping search. The search starts from an empty model and conducts recurring grow-and-prune cycles. During a growth phrase, EBMC uses a score to add the single best predictive node as a parent of the class node. After that, EBMC continuously adds parent nodes of the class node as long as the addition increases the score. When no additional node improves the score, EBMC starts a pruning phase. It converts the Bayesian network into a statistically equivalent Bayesian network that has the same score. Then, EBMC searches for an arc such that removing the arc increases the score the most. It keeps removing arcs until no single arc removal would increase the score. When the pruning phase stops, the search starts a new grow-and-prune cycle. The recurring cycles stop once no additional node can be added that improves

the score in the growth phrase. In this way, EBMC finds a high-scoring Markov blanket Bayesian network of the class node.

The score in EBMC uses a supervised (prequential) scoring. It also uses the BDeu strategy to incorporate prior knowledge through a prior equivalent sample size parameter. A structure prior of a candidate model in the EBMC search are estimated using a binomial distribution, and it is the product of the probability of including predictive variables in the model and the probability of excluding other variables as candidate predictors (Eq. 5, which is from (Cooper et al. 2010, Jiang et al. 2014),

Equation 5

$$Prior = p^{N_P+N_C} \times (1-p)^{N-(N_P+N_C)}$$

where p is the probability of including any given predictor in the model, and it is estimated as the ratio between the expected number of predictors and the total number of candidate predictors. N_P is the number of parents of the outcome node, N_C is the number of children of the outcome node. The prior is a product of the probability of each predictor in the model, and that probability is estimated using the proportion of expected number of predictors of the class node.

The complexity of an EBMC search is O (rs^2mn), where r is the total number of children node clusters, s is the maximum size of parents of the class node, n is the size of the feature space, and m is the size of the training data.

EBMC has been used to predict clinical outcomes from genome-wide data (Cooper et al. 2010, Jiang et al. 2014), and to detect influenza from emergency department free-text reports (López Pineda et al. 2015). In the studies performed to date, EBMC often has the predictive performance comparable to other traditional machine learning approaches while taking less time to learn a model.

2.3 Transfer Learning Techniques

In this section, I provide a definition of transfer learning, and then introduce five different transfer learning strategies.

2.3.1 Definition of Transfer Learning

(Pan et al. 2010) gave a definition of transfer learning as follows: "Given a source domain D_S and learning task T_S, a target domain D_T, and learning task T_T, transfer learning aims to help improve the learning of the target predictive function (target model) $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$." In this definition, a domain is denoted by D = {χ, P(X)}, where χ is the feature space, and P(X) is the marginal probability distribution of features. A task is denoted by T={Y, f(.)}, where Y is the label space of the class variable, and f(.) is an objective predictive function to be learned.

The main difference between a transfer learning problem and a traditional machine learning problem is whether the source and the target have an identical domain and task. In the definition of transfer learning, the source and the target may have different but related domains or have different but related tasks. Most transfer learning studies only focus on transfer learning tasks under one condition (i.e., domains are different, but tasks are identical; or domains are identical, but tasks are different).

2.3.2 Existing Transfer Learning Methods

Transfer learning techniques aim to provide a smoother transfer of knowledge in the form of models from the source to a different but related target. Existing transfer learning techniques can be divided into two branches: data transfer and model transfer. Data transfer includes four main categories: instance weighting (Huang et al. 2006, Jiang 2008, Sugiyama et al. 2008), feature representation (Aue et al. 2005, Arnold et al. 2007, Jiang et al. 2007, Satpal et al. 2007, Ciaramita et al. 2010, Pan et al. 2010, Pan et al. 2011, Wiens et al. 2014, Ogoe et al. 2015), self-labeling (Dai et al. 2007, Tan et al. 2009), and the hyper-parameter strategy (Roy et al. 2007, Finkel et al. 2009).

2.3.2.1 Instance Weighting

In order to adjust for difference between the marginal distribution of the source and of the target, some transfer learning algorithms assign different weights to instances from the two resources.

Jiang (Jiang 2008) represented a classification problem in the transfer learning setting as an optimization problem, with the aim of finding an optimal solution that minimized the expected loss with respect to the distribution of the target. The optimization process involved assigning different weights to instances from the source and the target in order to adjust the estimated loss. For each instance $(\boldsymbol{x_i}, y_i)$ in the source, the suggested weight was $\frac{\mathrm{P_t}(y_i \mid \boldsymbol{x_i})}{\mathrm{P_s}(y_i \mid \boldsymbol{x_i})}$, which was the ratio between the conditional probability in the target data and the conditional probability in the source data. For each labelled instance in the target, the suggested weight was the ratio between the number of instances in the source and the number of labelled instances in the target.

Other approaches aim to increase the similarity between the source and target distributions. To reweight instances in the source, Huang et al. (Huang et al. 2006) proposed a kernel mean matching approach, and Sugiyama et al. (Sugiyama et al. 2008) proposed a Kullback-Leibler importance estimation procedure.

2.3.2.2 Feature Representation

Many transfer learning algorithms manipulate features to maximize the similarity between the source and the target distributions. Two strategies (Weiss et al. 2016) may be utilized to maximize the similarity: (1) asymmetric transformation of the source domain to the target domain, (2) symmetric transformation mapping of the source and target domains into a common feature space.

Asymmetric transformation aims to transform features in the source to be similar to features in the target. For example, Wiens et al. (Wiens et al. 2014) conducted two transformations for source data: (1) remove source-specific features, (2) map source data to the target feature space by augmenting with zeros. Other distribution similarity approaches (Aue et al. 2005, Arnold et al. 2007, Jiang et al. 2007, Satpal et al. 2007) penalize or remove features whose distributions are different in the source and the target.

Symmetric transformation aims to map the source and target domains into a common feature space. For example, latent feature methods construct new features by analyzing large amounts of unlabeled source and target data (Ciaramita et al. 2010, Pan et al. 2010). Ogoe et al. (Ogoe et al. 2015) implemented a gene ontology-similarity-based method to identify common variables in the source and target datasets. Using this functional mapping, the performance of their transfer rule learner was improved and was shown to perform better than other integrative models driven by meta-analysis and cross-platform data merging. Other examples are the adversarial-

based deep learning algorithms (Ajakan et al. 2014, Long et al. 2016, Luo et al. 2017, Tzeng et al. 2017), which use an adversarial layer to achieve the lowest performance to discriminate the origin of the data (i.e., source or target)

2.3.2.3 Self-labeling

When the target has few labelled instances and many unlabeled instances, the unlabeled instances can be made valuable by assigning them pseudo-labels. For example, (Dai et al. 2007) applied the Expectation–Maximization (EM) algorithm to iteratively update the conditional probability tables in a transferred Naïve Bayes model using both source instances and unlabeled target instances. The tradeoff parameter between source instances and target instances was estimated using the Kullback-Leibler divergence (KL).

(Tan et al. 2009) conducted other adjustments for the traditional EM algorithm: (1) they used frequently co-occurring entropy to select generalizable features from the source and only used these features in the initial model for EM iteration; (2) they used all of the features that appeared in the target; (3) during the EM iteration, they gradually increased the weight of the data from the target and decreased the weight of the data from the source.

2.3.2.4 Hyper-parameter Strategy

The hyper-parameter strategy for multi-task learning can be applied to transfer learning, by viewing transfer learning as a special case of multi-task learning, where the source and the target are two tasks in a multi-task learning setting.

(Roy et al. 2007) developed a clustered Naïve Bayes model that was a hierarchical extension of a classic Naïve Bayes model. They placed a Dirichlet process prior over the conditional probability tables of many Naïve Bayes models, constraining these conditional

probability tables to be similar. The Dirichlet process coupled the multiple Naïve Bayes models by first partitioning the dataset into a number of clusters, each of which followed the same distribution (shared the same parameterization). Complete parameterizations of all these clusters were then drawn from the same base distribution from a Dirichlet process mixture model with a mixing parameter. Experiments showed that the clustered Naïve Bayes model performed better than the classic Naïve Bayes model when (1) the source and target were related (e.g. similar classification tasks), and (2) there were few instances from the target.

(Finkel et al. 2009) proposed a hierarchical model that added a layer over a general discriminative probabilistic model. For each domain, a Gaussian prior was centered on top-level parameters (hyperparameters). This design had two effects. First, if a feature appeared solely in domain A, domain B would have a similar parameter for the feature, because training instances in domain A will largely determine top-level parameters, which will then determine the parameters in domain B and there is no evidence in domain B to override the effect. Second, if a feature appeared in both domain A and domain B but with different strength in classification task, then the domain-specific evidences from both would eventually outweigh the effect of top-level parameters.

As described in these two examples, hyper-parameter algorithms use high-level parameters to capture the similarity among models for different domains, and at the same time allow variances among these models. When applying multi-task learning algorithms to transfer learning tasks, one potential problem is that these algorithms usually optimize the "average" performance over all tasks, which means that they always consider the source and the target equally important. Therefore, the target model developed by hyper-parameter algorithms may not perform as well as a model built using other types of transfer learning algorithms that aim to achieve a high

performance on target tasks. In addition, the assumptions in these hyper-parameter algorithms may be too strong and models may be too complicated when the source and the target have different feature sets and largely different distributions.

All four transfer learning strategies above require source data to be available.

2.3.2.5 Model-Transfer

Another branch of transfer learning is model-transfer, which focuses on sharing previously trained source models. Model-transfer techniques may reduce the adaptation time when the source sample size is extremely large. When only a model is publicly available for a task, the knowledge represented by this model can be reused for a similar task by anyone with a few training samples for model tuning.

Currently popular transfer learning methods using deep neural networks can be viewed as one type of model-transfer. Network-based deep learning algorithms reuse the structure (and sometimes the connection parameters) of the first few layers of the source network for the target domain. The rationale is that the first few layers are usually more general and the last few layers are more specific for tasks. For deep neural networks in computer vision tasks, the first layer is usually similar to Gabor filter and color blot which are general for different tasks. (Yosinski et al. 2014) experimentally quantified the generality of each layer of a deep convolutional network. They found that transferability was negatively impacted by the specialization of higher layers in the source model and optimization difficulties related to splitting networks occurred between co-adapted neurons when tuning source model for the target domain. The impact of these two issues on performance depends on whether features are transferred from the bottom, middle, or top of the source network.

One major function of deep transfer learning techniques is to tune the task-specific layers using a lot of target samples. (Long et al. 2015) developed a deep adaptation network (DAN), by embedding hidden representations of all the task-specific layers in a reproducing kernel Hilbert space and using a multiple kernel variant of "maximum mean discrepancies based multi-layer adaptation regularizer" to the loss function. The proposed algorithm achieved higher accuracy on unsupervised and semi-supervised transfer learning tasks on the Office-31 dataset and Office-10 plus Caltech-10 dataset, which are commonly used as standard testing datasets of transfer learning in the computer vision area. The researchers later proposed a new loss function, joint maximum mean discrepancy, and they showed that this algorithm outperformed the previous DAN approach on the same tasks (Long et al. 2016).

A main limitation of these deep transfer learning techniques (Long et al. 2015, Long et al. 2016) and other model-transfer learning techniques (Yang et al. 2007, Yang et al. 2007, Aytar et al. 2011) is their inability to perform transfer learning for heterogeneous scenarios where the source and the target have different feature spaces. (Mozafari et al. 2016) developed the first model-transfer method that conducted transfer learning in heterogeneous scenarios. This linear SVM based algorithm adds a regularizer in an objective function that minimizes the distance between target and source model in one-dimensional space. However, this algorithm has an overfitting problem for scenarios with a high feature dimension and a small number of target samples, and it is not robust to noise and outliers (Mozafari et al. 2016).

3.0 Bayesian Transfer Learning Framework

In the definition of transfer learning, the source and the target may have different but related domains or have different but related tasks. Most transfer learning studies only focus on transfer learning tasks under one condition. However, the reality is that it is common in real-world biomedical data for the source and the target to have different domains and different tasks at the same time.

My research focuses on a more general transfer learning scenario, heterogeneous transfer learning, which is learning when both domains and tasks are different for the source and the target. The feature space, marginal probability distribution, and objective predictive function can also be different; however, the label space of the class variable must be the same.

I designed two algorithms that differ in how the source and the target are connected. The first algorithm connects the data, conducting *Bayesian transfer learning using both target data and source data* (BTLSD algorithm), which belongs to the instance weighting category. The second algorithm focuses on model-transfer. Model-transfer involves much fewer privacy risks and so is more appealing for the biomedical field. This second algorithm, called *Bayesian transfer learning using source model* (BTLSM algorithm), processes the shared source information in the format of a Bayesian network model.

These two algorithms both focus on Bayesian network modeling for the classification task, where all predictive features and class are discrete variables. These two algorithms have two key processes: feature selection and model development.

3.1 Summary of Notation

$\boldsymbol{X}$: the vector of input variables. $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$. I use uppercase X to denote a random variable and lowercase x to denote its value. I use bold uppercase $\boldsymbol{X}$ to denote a vector of random variables and bold lowercase $\boldsymbol{x}$ to denote a vector of values of random variables.

Y: the class label. Similarly, lower case y denotes a value of Y.

Source: the party sharing data or a model. The source usually has abundant instances of data with both input variables and class labels. I use the character s to indicate the source. The source dataset is denoted as $D_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$, where Ns is the sample size of the dataset.

Target: the party to which the developed models will be applied. The target usually has a few instances of data with both input variables and class labels (denoted as $D_{t,l} = \{(\boldsymbol{x}_i^{t,l}, y_i^{t,l})\}_{i=1}^{N_{t,l}}$).

$P(\boldsymbol{X}, Y)$: the joint distribution of $\boldsymbol{X}$ and Y. $P_s(\boldsymbol{X}, Y)$ denotes the joint distribution in the source, while $P_t(\boldsymbol{X}, Y)$ denotes the joint distribution in the target. $P(\boldsymbol{x}, y)$ refers to the joint distribution when $\boldsymbol{X}=\boldsymbol{x}$ and $Y=y$.

$P_s(\boldsymbol{X})$ and $P_s(Y)$ are marginal distributions in the source, while $P_t(\boldsymbol{X})$ and $P_t(Y)$ are marginal distributions in the target.

$P_s(\mathbf{X}|Y)$ and $P_s(Y|\mathbf{X})$ are conditional distributions in the source, while $P_t(\mathbf{X}|Y)$ and $P_t(Y|\mathbf{X})$ are conditional distributions in the target.

3.2 Identifying Different Types of Features

Both algorithms have two key processes: feature selection and model development. In this section, I will explain the feature selection procedure, which includes identifying the different types of features constituting the feature pool (or candidate features) and selecting features using two methods.

A source model usually includes both generalizable features and source-specific features. A target model will include generalizable features and target-specific features. Therefore, the intersection of source model and target model is the set of generalizable features (Figure 1).

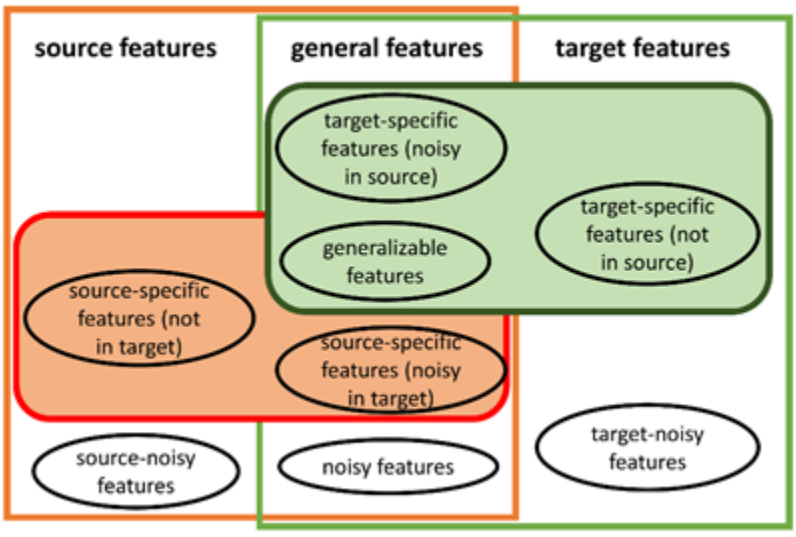

Figure 1 Types of features in the source and the target (Red box: features in the source model. Green box: features in the target model)

Table 1 summarizes eight different types of features of transfer learning tasks. General features are features of both the source and the target. General features include generalizable features, noisy features, source-specific features that are irrelevant in the target, and target-specific features that are irrelevant in the source. Source features are features that only appear in the source, including source-specific features that are unobserved in the target and source-noisy features. Target features are features that only appear in the target, including target-specific features that are unobserved in the source and target-noisy features.

Table 1 Different types of features in transfer learning tasks

Category	Feature type	Symbol	Observed in source data	Correlated to the class variable in source data	Observed in target data	Correlated to the class variable in target data	Included in the source model	Included in the target model
General features	Generalizable features	F_G	Yes	Yes	Yes	Yes	Yes	Yes
	Noisy features	F_N	Yes	No	Yes	No	No	No
	Source-specific features (irrelevant in target data)	F_{SI}	Yes	Yes	Yes	No	Yes	No
	Target-specific features (irrelevant in source data)	F_{TI}	Yes	No	Yes	Yes	No	Yes
Source features	Source-specific features (unobserved in target data)	F_{SU}	Yes	Yes	No	N/A	Yes	No
	Source-noisy features	F_{SN}	Yes	No	No	N/A	No	No
Target features	Target-specific features (unobserved in source data)	F_{TU}	No	N/A	Yes	Yes	No	Yes
	Target-noisy features	F_{TN}	No	N/A	Yes	No	No	No

When I build the target model, a key step is to identify the generalizable features and the target-specific features. Once I have both source data and target data to identify different feature types, I can use some feature selection approaches to identify influential feature sets for source data and target data respectively. In this dissertation, I mainly used information gain score (Kent 1983) or correlation-based feature selection (Hall 1999).

3.2.1 Information Gain Score

The information gain score is commonly used as an indicator of a finding's discriminative ability. It is the expected reduction in entropy after using a candidate feature to divide data into subgroups. Entropy is calculated using $\sum_i -p_i log_2 p_i$, where p_i is the probability of class i, and it is estimated using the proportion of class i in the training dataset.

As described in Table 1, the intersection of relevant feature sets in the source and the target are generalizable features. The remaining features in the relevant feature set of the target are target-specific features. They are either unobserved or irrelevant in the source. I calculate the information gain score of each candidate feature in the source and the target, respectively. Features with information gain scores greater than a threshold are included.

3.2.2 Correlation-based Feature Selection

Another feature selection approach is correlation-based feature selection (CFS). This approach has a central criterion that good feature sets contain features that are highly correlated with the class, yet uncorrelated with each other. The CFS approach has been found to be able to quickly identify noisy features and influential features as long as their relevancies do not strongly depend on other features (Hall 1999).

Similar to the information gain score approach, the CFS approach can be used to select a relevant feature set of the source and the target, respectively. Then, the intersection of the two sets is the set of generalizable features. The remaining features in the relevant feature set of the target is the set of target-specific features.

3.3 BTLSD Algorithm

The BTLSD algorithm aims to combine source data and target data for the classification task in the target. After obtaining data from the source, the simplest way is to mix source and target data and use the shared features of them (general features in Figure 1) for model development. However, this approach is not able to include target-specific features into final models.

The BTLSD algorithm is particularly designed to be able to remove source-specific features (noisy in target) and inject target-specific features into the model. The algorithm first identifies the generalizable features and target-specific features. Some target-specific features may be unobserved (F_{TU} in Table 1), or irrelevant (F_{TI} in Table 1) in the source data. Their counts are to be 0 in each class in the source data. Then, the BTLSD algorithm applies learning algorithms

(Naïve Bayes and K2) to build Bayesian network models with the selected features (i.e., generalizable features and target-specific features) using both target data and source data. The pseudocodes are provided in Appendix A.

Because transfer learning tasks usually have a large source training sample and a relatively small target training sample, it is possible that the source data will dominate the target data for model development. This can be harmful especially when the source and the target distributions are very different. Therefore, I designed three strategies to assign a lower weight to the source instances during the model building process.

3.3.1 BTLSD-R: Weighting Based on Sample Size Ratio

First, I can assign instances from the source a weight, which is the ratio between the sample size of target instances and the sample size of source instances. I use the short term BTLSD-R for this approach.

If the source sample size is larger than the target sample size, then I use the ratio as a weight to reduce the impact of source instances so that the weighted source sample size is equal to the target size. Jiang has also used the inverse of this heuristic to assign a weight for target instances (Jiang 2008).

3.3.2 BTLSD-KL: Weighting Based on Kullback-Leibler Divergence

Second, I can use the KL (Kullback et al. 1951), which is a measurement indicating the distance between two data distributions. It is the expectation of the logarithmic difference between one distribution and another distribution (Eq. 6). The KL divergence is not symmetric. In general,

$KL(P||\ Q)$ does not equal to $KL(Q\ ||\ P)$. $KL(P||\ Q)$ can indicate the amount of information lost when Q is used to approximate P.

Equation 6

$$KL(P||\ Q) = \sum_i P(i) \times \frac{P(i)}{Q(i)}$$

where $P(i)$ and $Q(i)$ are joint probability distributions for the *ith* configuration of feature values.

One nice property of KL is that it particularly penalizes the situation when $P(i)$ is large and $Q(i)$ is small, while allows a small $P(i)$and a large $Q(i)$. Because of this property, I used $KL(\mathrm{P_t}\ ||\ \mathrm{P_s})$ to estimate the difference between target distribution and source distribution (Eq. 7), where $\mathrm{P_t}$ is the target distribution and $\mathrm{P_s}$ is the source distribution. Transfer learning usually is needed when the source dataset has sufficient samples and the target dataset has few samples. When a configuration of the discrete variables frequently appears in the target dataset but rarely appears in the source dataset, the source dataset could be very different from the target dataset. However, I do not want to penalize the situation when a configuration of the discrete variables rarely appears in the target dataset but frequently appears in the source dataset because the low probability of the configuration in the target dataset may result from the small sample size of the target training data rather than the difference between the source and the target distributions.

Equation 7

$$KL\ (P_t\ ||\ P_s\ ) = \sum_{\boldsymbol{x},y} P_t\ (\boldsymbol{X} = \boldsymbol{x}, Y = y)\ log\ \frac{P_t\ (\boldsymbol{X}=\boldsymbol{x}, Y=y)}{P_s(\boldsymbol{X}=\boldsymbol{x}, Y=y)}$$

$$\mathbf{X} = \mathrm{X_1}, \mathrm{X_2}, \ldots, \mathrm{X_k} \quad \mathbf{x} = \mathrm{x_1}, \mathrm{x_2}, \ldots, \mathrm{x_k}$$

X is a vector of predictive features, Y is the class variable.

I weight all source instances with 2^{-KL} using a short term BTLSD-KL for this approach. When the two distributions are identical, KL divergence is 0 and the weight of source instances is 1. When a large difference exists between the distributions of the source and the target, the KL

divergence is large, and the source instances will be assigned a low weight in order to reduce their impacts on development of the target model.

3.3.3 BTLSD-FS-KL: Weighting Based on Feature-Specific KL

A third strategy is to use feature-specific KL. While the two approaches mentioned above assign the same weights to all generalizable features of source instances, another strategy takes into account of the possibility that the target distribution and the source distribution may be similar in some features and different in other features.

Since KL is the sum of all possible configurations, the complexity of KL calculation could be exponential. If two distributions can be fully represented by two Bayesian networks with the same structures, then the KL calculation can be decomposed (Eq. 8), and the complexity can be determined by the maximum number of parents, the maximum number of values, and the feature number.

Equation 8

$$KL\ (T,S) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} P_T(X_i = k, \pi_i = j) \times log \frac{P_T(X_i=k|\pi_i=j)}{P_S((X_i=k|\pi_i=j)}$$

In this way, KL can be viewed as the sum of individual KLs, each of which can be used to assign a feature-specific weight.

3.4 BTLSM Algorithm

The second algorithm, the BTLSM algorithm, conducts model-transfer in heterogeneous scenarios. For biomedical knowledge sharing, where privacy issues are a focus of concern, the BTLSM algorithm, which shares a modal across institutional boundaries, seems to be more feasible than the BTLSD algorithm, which requires sharing of thousands of unprocessed clinical data.

To obtain a final target model, the BTLSM algorithm scores each candidate network structure based on how many structure changes it has and how well this candidate network structure fits the target training data. In particular, the assessment is based on a score that integrates the distribution of target training data and the prior knowledge in the source model.

The BTLSM algorithm includes four main steps: cutting the source model, identifying candidate target-specific features (optional), conducting recurring grow-prune refinements, and assigning conditional probabilities to the final model (see pseudocodes in Appendix B).

3.4.1 Trimming the Source Model

In the "cutting" procedure, the BTLSM algorithm firstly removes features from the source model if these features are completely unobserved in the target data. It then removes predictive nodes that are outside the Markov Blanket of the class node and converts the resulting Bayesian network to a statistically equivalent one by making all parents of the class node its children. Lastly, the BTLSM algorithm removes arcs in the Bayesian network one by one for as long as the cutting increases the score. The approach acts in a greedy way. In each change, the most valuable deletion is chosen; once an arc has been cut, it will not be added back.

3.4.2 Identifying Candidate Target-Specific Features

This step is optional in the BTLSM algorithm, because the recurring grow-prune refinements pick/remove features to select a feature set for the predicting/classification task. The complexity of the algorithm is like that of the EBMC algorithm, which is $O(rs^2mn)$, where n is the size of the feature space. The recurring grow-prune will be polynomial to the size of the feature space.

When conducting this step, the BTLSM algorithm calculates the information gain scores for all of the features that are in the target data but not in the current Bayesian network and then excludes those noisy features whose information gain scores are lower than a certain threshold.

3.4.3 Conducting Recurring Grow-Prune Refinements

The BTLSM recurring grow-prune modification procedure is very similar to the EBMC grow-prune procedure. The main difference is the start point. Whereas the EBMC algorithm starts with the class node, the BTLSM algorithm starts with the refined source model. In the grow period of each grow-prune cycle, the BTLSM algorithm first adds new nodes as parents of the class node one by one if the change increases the score; this is done in a greedy way so that the most "valuable" addition is chosen first. When there is no "valuable" grow available, the Bayesian network is converted to a statistically equivalent one by making all newly added parents of the class node the children of the class node. Then, in the prune period of the grow-prune cycle, the BTLSM algorithm removes arcs one by one if the change increase the score. This is also done in a greedy way, where the largest change is conducted first. When there is no "valuable" prune

available, a new grow-prune cycle begins. The recurring grow-prune cycle ends once no new feature is added into the network during the grow period.

3.4.4 Proposed Score

The searching model process is led by a score. The score is the sum of log marginal likelihood and log structure prior.

Equations 9-13 show how the marginal likelihoods are calculated. Y represents the class variable for the predicting/classification task (e.g., whether a patient has influenza or not). y^N represents a set of class labels in the training data. $\boldsymbol{X}$ represents a set of predictive variables. $\boldsymbol{x}^N$ represents a set of states of prediction variables in the training data. Predictive variables can be divided into two sets: the parent set of the class variable (denoted as π_Y), and the children set of the prediction variable (denoted as c_Y). π_{X_i} denotes the parent set of *Xi*.

Equation 9

$$Marginal\ likelihood\ =\ P(y^N|\boldsymbol{x}^N, M) = \int P(y^N|\boldsymbol{x}^N, \theta, M)P(\theta|\boldsymbol{x}^N, M)d\theta$$

$$\approx \prod_{i=1}^{N} P\left(y_i|\boldsymbol{x}_i, y^{i-1}, \boldsymbol{x}^{i-1}, M\right) = \prod_{i=1}^{N} P\left(y_i|c_{y_i}, \pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)$$

$$= \prod_{i=1}^{N} \frac{P\left(y_i, c_{y_i}|\pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)}{P\left(c_{y_i}|\pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)} = \prod_{i=1}^{N} \frac{P\left(y_i, c_{y_i}|\pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)}{\sum_{y_i} P\left(y_i, c_{y_i}|\pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)}$$

Equation 10

$$\log(\text{marginal likelihood}) = \sum_{\mathrm{i}=1}^{\mathrm{N}} \left\{ \log \frac{\mathrm{P}\left(\mathrm{y_i}, \mathrm{c_{y_i}} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right)}{\sum_{\mathrm{y_i}} \mathrm{P}\left(\mathrm{y_i}, \mathrm{c_{y_i}} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right)} \right\}$$

$$= \sum_{\mathrm{i}=1}^{\mathrm{N}} \left\{ \log \mathrm{P}\left(\mathrm{y_i}, \mathrm{c_{y_i}} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right) - \log[\sum_{\mathrm{y_i}} \mathrm{P}\left(\mathrm{y_i}, \mathrm{c_{y_i}} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right)] \right\}$$

Equation 11

$$\mathrm{P}\left(\mathrm{y_i}, \mathrm{c_{y_i}} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right) = \mathrm{P}\left(\mathrm{y_i} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right) \times \mathrm{P}\left(\mathrm{c_{y_i}} \middle| \mathrm{y_i}, \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right)$$

$$= \mathrm{P}\left(\mathrm{y_i} \middle| \pi_{\mathrm{y_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right) \times \prod_{\mathrm{a} \in \mathrm{c_y}} \mathrm{P}\left(\mathrm{a_i} \middle| \pi_{\mathrm{a_i}}, \mathrm{y}^{\mathrm{i}-1}, \mathbf{x}^{\mathrm{i}-1}, \mathrm{M}\right)$$

Equation 12

$$log\ P\left(y_i, c_{y_i} \middle| \pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)$$

$$= log\ P\left(y_i \middle| \pi_{y_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right) + \sum_{x_i \in c_y} log\ P\left(x_i \middle| \pi_{x_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right)$$

Equation 13

Under the Bayesian Dirichlet assumption,

$$P\left(x_i \middle| \pi_{x_i}, y^{i-1}, \boldsymbol{x}^{i-1}, M\right) = \frac{N'_{ijk} + N_{ijk}}{N'_{ij} + N_{ij}}$$

N_{ijk} denotes the number of cases of the first i-1 instances of the training data $\{y^{i-1}, \boldsymbol{x}^{i-1}\}$ in which x_i is the *k*th state and π_{x_i} is the *j*th state. N_{ij} denotes the number of cases, of which π_{x_i} is the *j*th state. N'_{ijk} denotes the Dirichlet distribution component of the prior distribution for $P(x_i = k | \pi_{x_i} = j)$. $N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk}$

N'_{ijk} can be obtained from a prior probability distribution using $N' \times \mathrm{P}(x_i = \mathrm{k}, \pi_{x_i} = j | \mathrm{Ms})$. The joint probability can be inferred from a transferred source model Ms. One scenario that must be considered is that some features in the target may not appear in the source model. Joint

probabilities involving these features are not able to be directly retrieved from the source model. My strategy is to "inject" these features into the source model without connecting them to any of the other features (including the class node), and the conditional probability table will be non-informative for states of the feature (*e.g.*, setting 1/2 for a binary variable, 1/3 for a variable with three distinct states). The underlying assumption is that the source data has no information about the correlations between the injected feature and other features in the target domain. With this model, any joint probability involving any candidate feature can be inferred through the transferred source model in order to calculate the values of N'_{ijk} for score calculation.

Another component included in N'_{ijk} estimation is the equivalent sample size, N', which indicates the confidence level of a prior model. The BTLSM has four ways to estimate the equivalent sample size. A simple baseline approach (**unadjusted approach**) to use the sample size of the source training data. Another simple approach (**ratio approach**) to use the sample size of the target training data, which is the same as weighting the original source sample size with a ratio (number of target training instances / number of source training instances).

A more sophisticated approach is to weight the source sample size based on the similarity between the target distribution and the source distribution. If the source and the target distributions are very different, it would be better to assign a lower equivalent sample size for a transferred source model.

KL divergence approach: The KL divergence can be used as a multiplier (weight = 2^{-KL}) to get a reduced equivalent sample size. The empirical KL can be estimated by comparing the distribution of the target training data and the distribution of a transferred source model.

Feature-specific KL divergence approach: Another way to get a lower equivalent sample size for the source is to estimate KL for each feature and use a feature-specific equivalent sample size.

The searching score also considers a structure prior of each candidate Bayesian network structure. The BTLSM algorithm uses a modification of Heckerman's approach (Heckerman et al. 1995): $P\left(B_S^h \middle| \xi\right) = c\, k^{\sum_{i=1}^{n} \delta_i}$ where c is a normalization constant, which can be ignored since the score is mainly used for comparisons among candidate Bayesian network structures. Here, k is a constant parameter ($0<k\leq1$), representing the strength of penalizing the difference between a prior network and a candidate network structure. One way to estimate k is using the equivalent sample size: $k = \frac{1}{1+N'}$. The rationale behind this substitution is the consideration of the relationship between the degree of confidence of a prior network and the strength of the penalty of the difference between prior network and candidate network structure. $\delta_i = |[\pi_i(B_s) \cup \pi_i(P)] - [\pi_i(B_s) \cap \pi_i(P)]|$ is the symmetric difference between the parent set of a node in a candidate Bayesian network structure and the parent set of that node in a prior Bayesian network structure.

The Heckerman approach does not consider a scenario where a prior model structure and a candidate model structure have different feature spaces, which is normal in transfer learning tasks. Therefore, I modified the Heckerman approach using different k for different features (Eq. 14-15).

Equation 14

$$P\,(M) = c \prod k_i^{\,\delta_i}$$

Equation 15

$$log\, P\,(M) = log\, c + \sum_i (\delta_i \times log\, k_i) = log\, c + \sum_i (\delta_i \times log \frac{1}{1+N_i'})$$

The structure score is the sum of each feature score. For a target-specific feature (not in the source model), the N_i' is 0 and the feature score is 0. In addition, all δ_i calculations did not include the difference involving any target-specific features. With these rules, the BTLSM algorithm does not penalize any local structure involving any target-specific feature, assuming any local structure involving target-specific features to be equally likely.

Moreover, for shared features (in both the source model and target data), N_i' can use an unadjusted approach, ratio approach, KL approach, and Bayes factor approach.

3.4.5 Assigning Conditional Probabilities to the Selected Model Structure

After determining a graphic model structure, conditional probability tables can be estimated using all target training samples and Dirichlet distribution components (Eq. 16).

Equation 16

$$P\left(x_i \middle| \pi_{x_i}, v^{i-1}, u^{i-1}, M\right) = \frac{N'_{ijk} + N_{ijk}}{N'_{ij} + N_{ij}}$$

For shared features, the Dirichlet distribution component can be inferred from the transferred source model, $N'_{ijk} = N'\mathrm{P}(x_i = \mathrm{k}, \pi_{x_i} = j|\mathrm{B}_S)$, where N' can use an unadjusted, ratio, KL, or Bayes factor approach. For target-specific features, I set $N'_{ijk} = 1$ in order to not get a zero probability for some states.

3.4.6 An Illustration of BTLSM Algorithm

Figure 2 provides an illustration of the BTLSM algorithm. From network 1 to network 5, the algorithm conducts the procedure of cutting the source model. The first network is the source

model. C node is the class node. S1 to S6 are the predictive feature nodes. Because feature S6 is not observed in the target, it is removed. Because node S5 is not in node C's Markov Blanket, it is removed. We then obtain the third network. This network is converted to a statistically equivalent one by making S4 a child of the class node. The arc between S2 and S3 is removed because the deletion increases the score in the target data. From network 5 to network 8, the algorithm conducts recurring grow-prune refinements. From network 5 to network 6, two target-specific features are added in the grow stage. Network 6 is then converted to network 7. Next, the arc between T2 and T3 is removed in the prune stage. After obtaining a final model, the algorithm assigns conditional probabilities by using source information and target data.

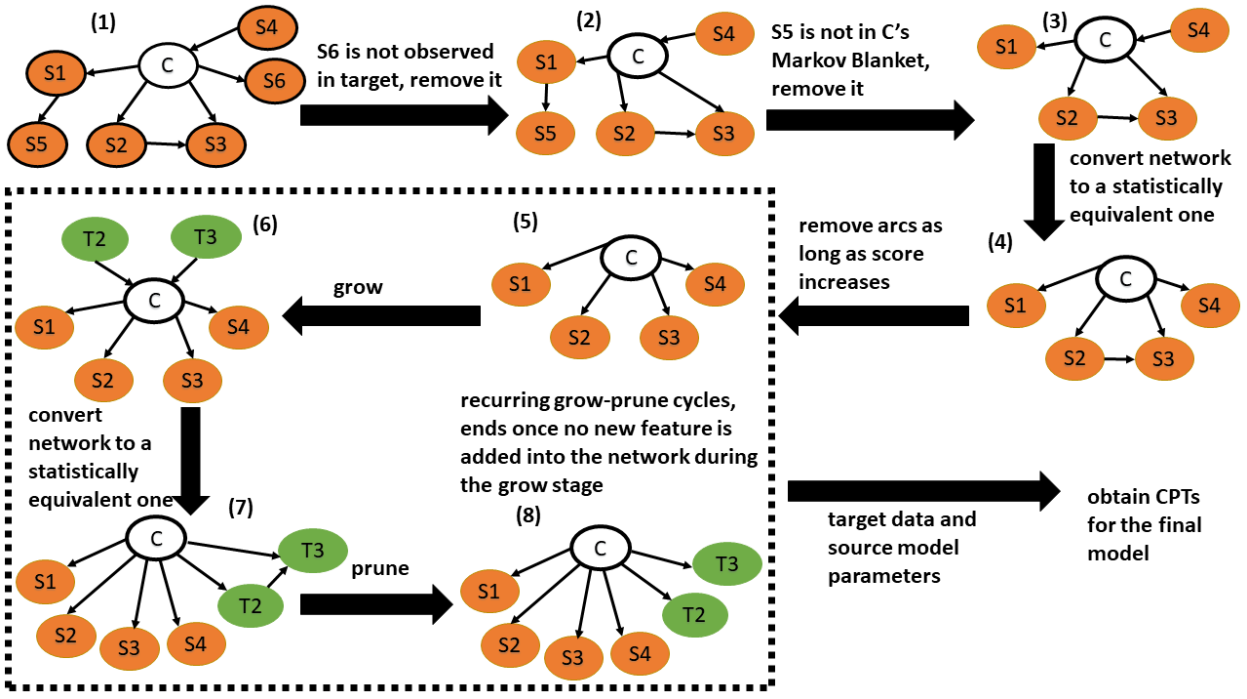

Figure 2 An illustration of the BTLSM algorithm

4.0 Evaluating the Transfer Learning Algorithms Using Synthetic Datasets

In order to have a comprehensive understanding of the effectiveness of the proposed transfer learning algorithms, I evaluated these algorithms using synthetic transfer learning tasks that varied in degrees of feature space overlapping, similarity of distributions, and sample sizes.

To generate these synthetic datasets, I assumed that the target data followed a joint distribution encoded by the original gold-standard networks and the source data followed the joint distribution encoded by other variant networks whose feature space, feature dependencies, and conditional probability tables differed from the target model.

To test both hypotheses under different conditions, I conducted two experiments with different gold-standard networks and settings. The first experiment was designed to test the transfer learning algorithms' performance in heterogeneous scenarios where the feature spaces of source and target domains were different. Heterogeneous scenarios are very common in medical data from different institutions, but very few transfer learning algorithms can be applied to them. The first experiment used four Bayesian networks previously learned from real-world datasets. With these four models as ground-truth models for source or target, I designed six transfer learning scenarios which differed with respect to the number of features shared between the source and target models and the KL divergence of the shared features. I then conducted transfer learning for these six scenarios under different source sample size and target sample size situations.

The second experiment was performed to examine to study how transfer learning algorithms would behave when the KL divergences between the source and target distributions increase. For this experiment, the source and target models shared the same feature space and network structure. The true target model's structure and parameters were directly copied from an

intubation network, which is a portion of a medical diagnostic network (the Logical Alarm Reduction Mechanism, ALARM) defined by human experts (Beinlich et al. 1989). There were 141 true source models, which maintained the same Bayesian network structure as the true target model. One of them had the same parameters as the true target model. The remaining 140 source models' parameters were randomly generated through a mechanism to control the amount of the difference.

In these experiments, all features were discrete with multinomial distributions, and these features were completely observed. I assumed time invariance when using a network to generate each synthetic dataset.

In the remaining sections of this chapter, I first describe the design of the first experiment, present the performance of the transfer learning algorithms in different scenarios, and discuss why the transfer learning algorithms performed differently in these scenarios. Then, I introduce the setting of the second experiment, compare different approaches to estimating empirical KL divergences, and draw plots showing how transfer learning algorithms behave when KL divergences increase.

4.1 Experiment 1: Influenza Network

4.1.1 Experiment Design

We previously conducted a study on the transferability of influenza case detection systems between two large healthcare systems, Intermountain Healthcare (IH) in Utah and University of Pittsburgh Medical Center (UPMC) in Pittsburgh, PA (Ye et al. 2017). In that study, I developed four training datasets. The IH training datasets included 47,504 Emergency Department (ED) encounters at IH facilities between January 1, 2008 and May 31, 2010: 1,858 *influenza* encounters (encounters with a positive laboratory test for influenza using polymerase chain reaction, direct fluorescent antibody, or viral culture), and 15,989 *NI-ILI* encounters (encounters with at least one negative result for the three lab tests), and 29756 other encounters (encounters without any laboratory tests for influenza). The IH training datasets were associated with 60,344 notes (1.2 notes per encounter). From these notes, the IH parser identified 934,414 findings; the UPMC parser identified 877,377 (94% of the IH findings). From the same time period, the UPMC training datasets consisted of 41,189 ED encounters, which were constructed in an identical manner as those for IH. These training datasets included 915 *influenza*, 3,040 *NI-ILI*, and 37,234 *other* encounters, all of which were associated with 76,467 notes (1.9 notes per encounter). From these notes, the UPMC parser retrieved 1,031,134 findings; the IH parser identified 849,932 findings (82% of the UPMC findings).

We developed four Bayesian case detection systems (BCD) at Intermountain Healthcare in Utah (BCD_{IH}) and the University of Pittsburgh Medical Center (BCD_{UPMC}). These four models were machine-learned with four training datasets. Each medical center had two training datasets; one used a local parser to retrieve clinical findings, and the other used a non-local parser.

With the four training datasets (IH/UPMC notes + IH/UPMC parser), I used feature selection, machine learning, and an expert debiasing approach to develop four Bayesian network models (Figure 3): (1) BN_{IH}&NLP_{IH} (Bayesian network learned with IH clinical findings extracted by the IH parser), (2) BN_{IH}&NLP_{UPMC} (IH notes and UPMC parser), (3) BN_{UPMC}&NLP_{UPMC} (UPMC notes and UPMC parser), and (4) BN_{UPMC}&NLP_{IH} (UPMC notes and IH parser).

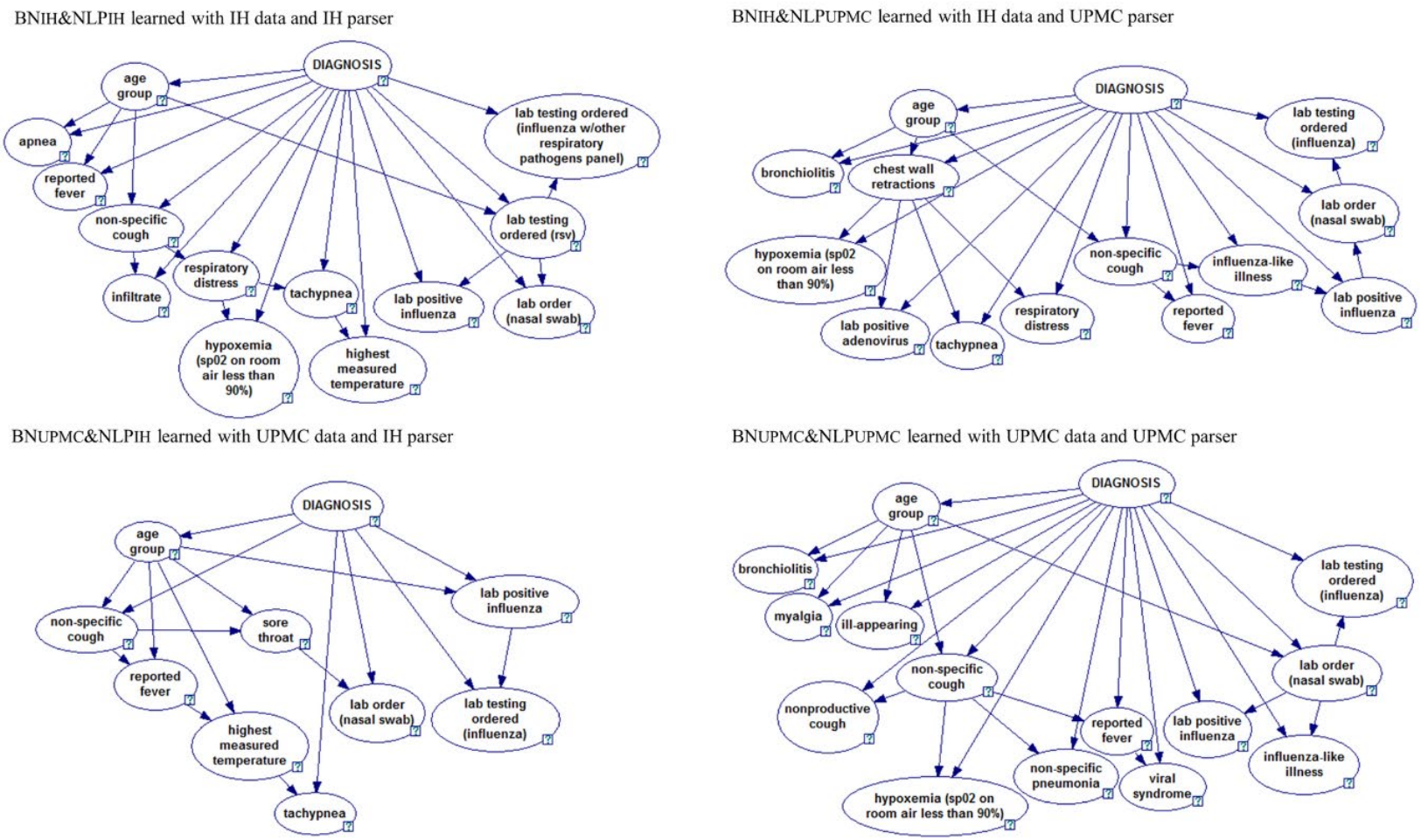

Figure 3 Four Bayesian network models developed using datasets distinguished by data resources (i.e., IH notes or UPMC notes) and NLP parsers (i.e., IH parser or UPMC parser) (from Ye et al. 2017)

These four Bayesian networks can be used to represent the six transfer learning scenarios (Table 2). For example, in scenario 1, I transferred a 14-feature UPMC Bayesian network (developed using UPMC notes and the UPMC parser) to IH, where a different NLP parser is used. The true IH model had 12 predictive features, of which six also appeared in the transferred UPMC model. The KL of these shared features between the source and the target model is 1.8224. In scenario 2, I transferred a UPMC model to IH, and both institutions used the same NLP parser, NLP_{IH}. Using the same NLP parser reduced the KL from 1.8224 to 1.142, with an extra shared feature, the *highest measured temperature* (which was included in the KL calculation of this scenario). Moreover, the distributions of the *influenza lab positive* node and the *nasal swab order* node between source and target became similar, indicating that the differences for these three features in scenario 1 mainly resulted from the NLP parser differences instead of institutional differences. When the two institutions used the same UPMC parser (scenario 3), the *influenza lab positive* node between the source and the target became more similar. The *bronchiolitis* node also became transferable between the source and the target. Scenario 5 and scenario 3 have the smallest KLs, and there are nine features shared between the two networks in these scenarios.

Table 2 Transfer learning scenarios and the difference between the target and the source

Scenario	True Source Model			True Target Model			Number of features in source or target	Number of shared features	Proportion of shared features among all features	KL (target to source)
	Name	Number of features	Percentage of shared features	Name	Number of features	Percentage of shared features				
1	BN_{UPMC}&NLP_{UPMC}	14	0.43	BN_{IH}&NLP_{IH}	12	0.50	20	6	0.30	1.8224
2	BN_{UPMC}&NLP_{IH}	9	0.78	BN_{IH}&NLP_{IH}	12	0.58	15	7	0.47	1.142
3	BN_{UPMC}&NLP_{UPMC}	14	0.64	BN_{IH}&NLP_{UPMC}	13	0.69	21	9	0.43	0.6057
4	BN_{IH}&NLP_{IH}	12	0.50	BN_{UPMC}&NLP_{UPMC}	14	0.43	20	6	0.30	1.2793
5	BN_{IH}&NLP_{UPMC}	13	0.69	BN_{UPMC}&NLP_{UPMC}	14	0.64	21	9	0.43	0.3855
6	BN_{IH}&NLP_{IH}	12	0.58	BN_{UPMC}&NLP_{IH}	9	0.78	15	7	0.47	1.462

I used these four Bayesian networks to generate datasets for the six transfer learning scenarios. For each scenario, I generated datasets under different source sample size (8000, 50) and target sample size (8000, 50) situations. The first situation for transfer learning tasks was when the source had a large number of training samples while the target had only a few, so I simulated 8,000 source training samples and 50 target training samples to represent this situation. I also wanted to investigate whether transfer learning could still be beneficial when the target has many samples, so the second situation was 8,000 source training samples and 8,000 target training samples. In addition, I was interested in whether a few source samples could add extra information for the target, and thus designed another two situations (source sample size: 50, target sample size: 50 or 8,000).

With these synthetic training datasets, I used the information gain approach (threshold 0.0001) and K2 to develop source-only and target-only models. Because these synthetic datasets were generated from models that previously learned using a feature selection approach, almost all of the features were related to the class task. Using a small threshold 0.0001, as with the information gain approach, may be helpful to keep all relevant features. The information gain approach was also indicated from results of a preliminary experiment, which compared different feature selection and modeling strategies for learning with 50 training samples. In that experiment, I used each of the IH and UPMC (four) Bayesian models to simulate ten training datasets (50 samples), built models with different feature selection and modeling strategies. I tested their discriminative ability with a measure of area under receiver operating characteristic curve (AUC) for influenza vs. non-influenza on 10 simulated test datasets (10,000 samples). One good characteristic of AUC is that it is not impacted by disease prevalence. So, it is a good measurement to indicate the discriminative ability of models whose main function is to use dynamic disease

prevalence for disease diagnosis and at the same time provide likelihoods to outbreak detection systems. Table 3 shows that the information gain approach performs better than the CFS approach for three of the four gold standard models.

Table 3 Comparisons of mean AUC of models learned from 50-sample training datasets with different feature selection and machine learning algorithms

Feature selection	Learning algorithm	BN_{UPMC}&NLP_{UPMC}	BN_{UPMC}&NLP_{IH}	BN_{IH}&NLP_{UPMC}	BN_{IH}&NLP_{IH}
CFS	K2	0.8767	0.7005	0.8813	0.8890
	NB	0.8767	0.7005	0.8813	0.8890
IG (0.0001)	K2	0.9430	0.6991	0.9075	0.9195
	NB	0.9430	0.6991	0.9075	0.9195
IG (0.0002)	K2	0.9430	0.6996	0.9075	0.9195
	NB	0.9430	0.6996	0.9075	0.9195
IG (0.0005)	K2	0.9430	0.6996	0.9075	0.9195
	NB	0.9430	0.6996	0.9075	0.9195
IG (0.001)	K2	0.9430	0.6996	0.9075	0.9195
	NB	0.9430	0.6996	0.9075	0.9195
IG (0.01)	K2	0.9271	0.7019	0.9022	0.9138
	NB	0.9271	0.7019	0.9022	0.9138

IG: feature selection based on information gain; the threshold is provided inside the parentheses.

All of the developed Bayesian network models were tested using synthetic datasets, each of which included 10,000 visits simulated from the true target model in each scenario. I conducted 10-fold experiments for situations in each scenario and used average performance to provide more reliable simulation results. The inference engineer used for the Bayesian network models was the SMILE Engine (Druzdzel 1999) from BayesFusion LLC. Each model provided estimations of probability of having influenza for each testing visit, with which the AUC for influenza vs. non-influenza was calculated. To compare two models' performance, I used DeLong's two-sided

statistical tests (DeLong et al. 1988) as implemented in the pROC package of R statistical software (Robin et al. 2011).

To study the calibration ability of models, I calculated the expected calibration error (ECE) (Naeini et al. 2015, Naeini et al. 2015). The lower the values for ECE, the better the calibration ability of a model. The expected calibration error was calculated as follows (Eq. 17): (1) partition the output space of a binary model into K bins (K=10 in my experiment), (2) in each bin, calculate the calibration error as the difference between observed faction and mean of estimated probabilities of instances in the bin, (3) sum up the error by weighting each error with an empirical probability of instances in the corresponding bin.

Equation 17

$$\mathrm{ECE} = \sum_{k=1}^{K} P(k) \ |o_k - e_k|$$

Where *P(k)* is an empirical probability estimated using the fraction of all instances in bin *k*, e_k is the average of the probabilities (estimated by a model) for all instances in bin *k*, and o_k is the observed fraction of positive instances in that bin *k*.

4.1.2 Results Varying Source and Target Sample Sizes Across Six Transferring Scenarios

4.1.2.1 Source Size 8000, Target Size 50

When the source has many samples and the target has a few (the most common situation in transfer learning tasks), my results showed that transferring a source model could be helpful, and that performance could be boosted by using BTLSM algorithms to incorporate information from the transferred source model with the small target training dataset.

In detail, when the source size was 8,000 and the target size was 50, transferring a source model had positive effects in scenarios 1, 2, 3, 5, and 6 (Table 4-5). The learned source models already performed statistically significantly better than the learned target models in scenarios 1, 2, 3, and 6. For scenario 1, the BTLSM-unadjusted algorithm boosted the performance of the learned source models from 0.924 to 0.9283. For scenario 2, the BTLSM-unadjusted, BTLSM-KL, and BTLSM-feature-specific-KL algorithms all significantly boosted the performance of the learned source models, and the BTLSM-KL algorithm's impact was the greatest (AUC increased from 0.9078 to 0.9177). For scenarios 3 and 6, the BTLSM algorithm did not boost the performance. For scenario 5, the learned source models (AUC: 0.9498) did not perform significantly better than the learned target models (AUC: 0.9413). With 50 target training visits, the BTLSM-unadjusted, BTLSM-KL, and BTLSM-feature-specific-KL algorithms significantly increased the performance and the final models (AUC: 0.9555) became significantly better than the learned target models.

In scenario 4, neither the sharing model nor the sharing data were necessary. The learned source model (0.9183) did not perform better than the learned target model (0.9413). Even though the BTLSM algorithm boosted their performance, the final models still did not outperform the learned target models.

On the other hand, although the BTLSD models also performed significantly better than the learned target model in scenario 1, 2, 3, and 6, they did not perform significantly better than the learned source models in these four scenarios.

When the target size is small, and the source size is large, it would be better to use the BTLSM algorithm than the BTLSD algorithm. In scenario 1, the best BTLSM model (prior model: learned source model) performance was 0.9283 (BTLSM-unadjusted), which was significantly better than the best BTLSD model 0.9066 (BTLSD-feature-specific-KL) ($p<0.0001$). The best

BTLSM model always performed significantly better than the best BTLSD model in scenario 2 (0.9177 vs 0.9019, $p<0.0001$), scenario 3 (0.9425 vs. 0.9132, $p<0.0001$), scenario 4 (0.9383 vs. 0.9278, $p=0.0299$), and scenario 5 (0.9555 VS. 0.9359, $p<0.0001$). In scenario 6, the performance of the best BTLSM model and the best BTLSD model were not significantly different (0.7662 vs. 0.7628, $p=0.6926$).

In addition to examining discriminative ability, I further compared the expected calibration error as a measurement of calibration (Table 6). The transfer learning models did not have a lower expected calibration error than the learned target models in scenarios 3, 4, 5, and 6. In scenarios 1 and 2, the BTLSD ratio models' expected calibration error was slightly lower than the learned target model. Compared to the learned source model, most of the error of the BTLSM models was slightly lower.

4.1.2.2 Source Size 8000, Target Size 8000

The results for this situation indicate that when both the target size and the source size are large, the BTLSD algorithm is preferred. Although there were already 8,000 visits in the target training data, in scenarios 1, 2, and 6, the BTLSD models still performed better than the learned target models, indicating that the source data still added extra classification capability (Table 7-8). In scenario 6, the BTLSM models also performed better than the learned target models, and the best BTLSM model and the best BTLSD model did not perform significantly differently (0.7707 vs. 0.7724, $p=0.7922$). In scenarios 3, 4, and 5, sharing model/data became unnecessary, as all of the BTLSM models and BTLSD models did not perform better than the target model.

When I compared the expected calibration error as a measurement of calibration (Table 9), the transfer learning models had a slightly lower error than the learned target model in scenarios

1, 2, and 3. Compared to the learned source models, the transfer learning models greatly reduced the calibration error.

4.1.2.3 Source Size 50, Target Size 50

The results for this situation show that when the target size is small, a few source data or a source model may be useful in some conditions. For example, in scenario 6, the BTLSD models performed statistically significantly better than the learned target models (Table 10-11). In scenario 2, the BTLSM models performed statistically significantly better than the learned target models.

When I compared the models' expected calibration error (Table 12), in scenarios 1 and 2, the BTLSD models had a slightly lower error than the learned target models. In all the scenarios, the BTLSD models had a lower error than the learned source models.

4.1.2.4 Source Size 50, Target Size 8000

To my surprise, in situation where the target size is large, and the source size is small, the BTLSM models still outperformed the learned target model in scenario 6 (Table 13-14). Also, the BTLSD models still outperformed the learned target models in scenario 2.

When I compared models' expected calibration error (Table 15), in all scenarios, the transfer learning models had a slightly lower error than the learned target models and the learned source models.

Table 4 Summative results for 10-fold experiments (source size=8000, target size=50)

Measurement and comparison	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
learned source model > learned target model	yes	yes	yes	no	no	yes
Hypothesis 1: BTLSM (prior model: learned source model) > learned target	unadjusted, KL, feature-specific KL	unadjusted, KL, feature-specific KL	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL	unadjusted, KL, feature-specific KL, ratio
Hypothesis 1: BTLSM (prior model: true source model) > learned target	unadjusted, KL, feature-specific KL	none	unadjusted, KL, feature-specific KL	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, feature-specific KL, ratio
Hypothesis 1: BTLSD > learned target	unadjusted, KL, feature-specific KL	KL	unadjusted, KL, feature-specific KL	none	none	unadjusted, KL, feature-specific KL, ratio
Hypothesis 2: BTLSM (prior model: learned source model) > learned source	unadjusted	unadjusted KL, feature-specific KL	none	unadjusted KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL	none
Hypothesis 2: BTLSD > learned source	none	none	none	KL	none	none
true source > learned target	yes	no	yes	no	no	yes
BTLSM (prior model: true source model) > true source	none	none	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL	none
BTLSD > true source	none	none	none	unadjusted, KL, feature-specific KL, ratio	none	unadjusted KL, feature-specific KL

Only statistically significantly better results are mentioned ($p<0.05$) in this table.

Table 5 Average AUC and confidence interval of models in 10-fold experiments (source size=8000, target size=50)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target 0.969 (0.9669-0.9711)[123]	true target 0.969 (0.9669-0.9711)[123]	true target 0.9643 (0.9612-0.9674)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.8053 (0.7808-0.8299)[123]
learned source BTLSM unadjusted 0.9283 (0.9242-0.9323)[12]	learned source BTLSM KL 0.9177 (0.9136-0.9219)[123]	true source 0.9459 (0.9417-0.9501)[2]	learned target 0.9413 (0.9329-0.9497)[13]	learned source BTLSM KL 0.9555 (0.9475-0.9634)[123]	BTLSD KL 0.7662 (0.739-0.7933)[23]
true source BTLSM KL 0.928 (0.9241-0.932)[12]	learned source BTLSM feature-specific-KL 0.9154 (0.9113-0.9196)[123]	learned source 0.9447 (0.9406-0.9488)[2]	learned source BTLSM KL 0.9383 (0.9277-0.949)[1]3	learned source BTLSM unadjusted 0.9555 (0.9475-0.9634)[123]	BTLSD unadjusted 0.7647 (0.7369-0.7925)[23]
true source BTLSM unadjusted 0.9274 (0.9235-0.9314)[2]	learned source BTLSM unadjusted 0.9141 (0.9099-0.9183)[123]	true source BTLSM unadjusted 0.9433 (0.9391-0.9476)[23]	learned source BTLSM unadjusted 0.9379 (0.9273-0.9486)[1]3	learned source BTLSM feature-specific-KL 0.9555 (0.9475-0.9634)[123]	BTLSD feature-specific-KL 0.7643 (0.7365-0.792)[23]
learned source BTLSM feature-specific-KL 0.9264 (0.9223-0.9306)[2]	learned source 0.9078 (0.9035-0.912)[23]	true source BTLSM feature-specific-KL 0.9433 (0.939-0.9475)[23]	learned source BTLSM feature-specific-KL 0.9379 (0.9272-0.9485)[1]3	true source BTLSM unadjusted 0.9546 (0.947-0.9621)[23]	learned source BTLSM KL 0.7628 (0.7365-0.7891)[23]
learned source BTLSM KL 0.9262 (0.9219-0.9304)[2]	BTLSD KL 0.9019 (0.8976-0.9062)[12]	true source BTLSM KL 0.9425 (0.9383-0.9468)[23]	learned source BTLSM ratio 0.9294 (0.9193-0.9395)[123]	true source BTLSM feature-specific-KL 0.9545 (0.9469-0.9621)[23]	learned source BTLSM feature-specific-KL 0.7612 (0.7344-0.7879)[23]
true source BTLSM feature-specific-KL 0.9258 (0.9218-0.9299)[2]	true source 0.901 (0.8953-0.9066)[1]	learned source BTLSM unadjusted 0.9425 (0.9384-0.9466)[123]	BTLSD KL 0.9278 (0.9173-0.9383)[123]	true source BTLSM KL 0.9544 (0.9468-0.9621)[23]	learned source BTLSM unadjusted 0.7611 (0.7343-0.7879)[23]
true source 0.9255 (0.9215-0.9296)[2]	BTLSD feature-specific-KL 0.8983 (0.8938-0.9028)[1]	learned source BTLSM feature-specific-KL 0.9425 (0.9383-0.9466)[123]	BTLSD feature-specific-KL 0.9208 (0.9095-0.9321)[23]	learned source 0.9498 (0.9396-0.9599)	learned source 0.759 (0.7318-0.7861)[23]

Table 5 Average AUC and confidence interval of models in 10-fold experiments (source size=8000, target size=50) (continued)

learned source 0.924 (0.9197-0.9283)[2]	BTLSD unadjusted 0.8977 (0.8931-0.9022)[1]	learned source BTLSM KL 0.9417 (0.9375-0.9458)[123]	BTLSD unadjusted 0.9202 (0.9088-0.9316)[23]	true source BTLSM ratio 0.9498 (0.942-0.9577)[2]	BTLSD ratio 0.7452 (0.717-0.7733)[2]
BTLSD feature-specific-KL 0.9066 (0.902-0.9111)[123]	learned target 0.8955 (0.8904-0.9005)[1]	BTLSD unadjusted 0.9132 (0.9081-0.9184)[123]	BTLSD ratio 0.9184 (0.9078-0.929)[23]	true source 0.9455 (0.934-0.957)	learned source BTLSM ratio 0.7373 (0.7106-0.764)[12]
BTLSD unadjusted 0.9061 (0.9015-0.9106)[123]	BTLSD ratio 0.8909 (0.8864-0.8955)[123]	BTLSD feature-specific-KL 0.9132 (0.9081-0.9183)[123]	learned source 0.9183 (0.905-0.9316)[23]	learned source BTLSM ratio 0.9443 (0.9342-0.9544)	true source 0.7297 (0.7001-0.7593)[12]
BTLSD KL 0.9014 (0.8968-0.906)[123]	true source BTLSM unadjusted 0.889 (0.8835-0.8946)[123]	BTLSD KL 0.9119 (0.9068-0.917)[123]	true source BTLSM unadjusted 0.8999 (0.887-0.9128)[123]	learned target 0.9413 (0.9329-0.9497)	true source BTLSM feature-specific-KL 0.7272 (0.6979-0.7565)[12]
learned source BTLSM ratio 0.8988 (0.8936-0.904)[13]	true source BTLSM feature-specific-KL 0.8887 (0.8832-0.8943)[123]	learned source BTLSM ratio 0.8886 (0.8829-0.8943)[123]	true source BTLSM feature-specific-KL 0.8984 (0.8856-0.9113)[123]	BTLSD KL 0.9359 (0.9252-0.9466)[1]	true source BTLSM unadjusted 0.7262 (0.6969-0.7555)[12]
learned target 0.8955 (0.8904-0.9005)[13]	true source BTLSM KL 0.8841 (0.8783-0.8898)[123]	true source BTLSM ratio 0.8866 (0.8804-0.8928)[13]	true source BTLSM KL 0.8915 (0.8777-0.9054)[123]	BTLSD unadjusted 0.9303 (0.9193-0.9414)[123]	true source BTLSM ratio 0.7215 (0.6937-0.7493)[12]
BTLSD ratio 0.8808 (0.8757-0.8858)[123]	learned source BTLSM ratio 0.8834 (0.877-0.8898)[123]	learned target 0.8819 (0.8757-0.8882)[13]	true source BTLSM ratio 0.8748 (0.8594-0.8903)[123]	BTLSD feature-specific-KL 0.9303 (0.9193-0.9414)[123]	true source BTLSM KL 0.7107 (0.6806-0.7408)[13]
true source BTLSM ratio 0.8706 (0.8636-0.8777)[123]	true source BTLSM ratio 0.8477 (0.8402-0.8553)[123]	BTLSD ratio 0.8327 (0.8252-0.8402)[123]	true source 0.7086 (0.6805-0.7366)[12]	BTLSD ratio 0.9119 (0.9003-0.9236)[123]	learned target 0.6868 (0.661-0.7125)[13]

From top to bottom, average AUC ranks from high to low. Green: AUC of learned source. Yellow: AUC of learned target.
1. significantly different from learned source model ($p<0.05$)
2. significantly different from learned target model ($p<0.05$)
3. significantly different from true source model ($p<0.05$)

Table 6 Average expected calibration error of models in 10-fold experiments (source size=8000, target size=50)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target, 0.001	true target, 0.001	true target, 0.0004	true target, 0.0007	true target, 0.0007	true target, 0.0013
BTLSD ratio, 0.014	BTLSD ratio, 0.0168	learned target, 0.0085	learned target, 0.0021	learned target, 0.0021	learned target, 0.0069
learned target, 0.017	learned target, 0.017	true source BTLSM ratio, 0.0098	learned source BTLSM ratio, 0.0055	BTLSD ratio, 0.0059	true source BTLSM ratio, 0.0156
learned source BTLSM ratio, 0.0208	learned source BTLSM ratio, 0.025	BTLSD ratio, 0.0105	true source BTLSM ratio, 0.0056	true source BTLSM ratio, 0.0085	learned source BTLSM ratio, 0.0185
true source BTLSM ratio, 0.0218	true source BTLSM ratio, 0.0252	learned source BTLSM ratio, 0.0113	true source BTLSM unadjusted, 0.008	learned source BTLSM ratio, 0.0091	BTLSD ratio, 0.0186
true source BTLSM KL, 0.0223	learned source BTLSM KL, 0.0289	true source, 0.0126	true source BTLSM feature-specific-KL, 0.0081	true source BTLSM KL, 0.0093	true source BTLSM KL, 0.0205
learned source BTLSM KL, 0.0224	BTLSD KL, 0.0291	true source BTLSM KL, 0.0135	learned source BTLSM KL, 0.0084	true source BTLSM unadjusted, 0.0094	learned source BTLSM KL, 0.022
true source, 0.0225	learned source, 0.0296	true source BTLSM feature-specific-KL, 0.0136	true source BTLSM KL, 0.0086	true source BTLSM feature-specific-KL, 0.0094	learned source BTLSM feature-specific-KL, 0.022
true source BTLSM feature-specific-KL, 0.023	true source, 0.03	true source BTLSM unadjusted, 0.0137	BTLSD ratio, 0.0086	BTLSD KL, 0.01	learned source BTLSM unadjusted, 0.0221
true source BTLSM unadjusted, 0.0231	BTLSD unadjusted, 0.03	learned source, 0.0143	learned source BTLSM unadjusted, 0.0092	BTLSD unadjusted, 0.0109	true source BTLSM feature-specific-KL, 0.0246
learned source, 0.0234	BTLSD feature-specific-KL, 0.03	learned source BTLSM KL, 0.015	learned source BTLSM feature-specific-KL, 0.0092	BTLSD feature-specific-KL, 0.0109	true source BTLSM unadjusted, 0.025
learned source BTLSM feature-specific-KL, 0.0234	learned source BTLSM feature-specific-KL, 0.0301	learned source BTLSM unadjusted, 0.0153	BTLSD unadjusted, 0.0115	true source, 0.0126	learned source, 0.0276

Table 6 Average expected calibration error of models in 10-fold experiments (source size=8000, target size=50) (continued)

learned source BTLSM unadjusted, 0.0235	learned source BTLSM unadjusted, 0.0302	learned source BTLSM feature-specific-KL, 0.0153	BTLSD feature-specific-KL, 0.0116	learned source BTLSM KL, 0.0131	BTLSD KL, 0.0288
BTLSD KL, 0.0237	true source BTLSM KL, 0.0305	BTLSD unadjusted, 0.0159	BTLSD KL, 0.0121	learned source BTLSM unadjusted, 0.0132	BTLSD feature-specific-KL, 0.0304
BTLSD feature-specific-KL, 0.0254	true source BTLSM unadjusted, 0.0308	BTLSD KL, 0.0159	learned source, 0.0216	learned source BTLSM feature-specific-KL, 0.0132	BTLSD unadjusted, 0.0306
BTLSD unadjusted, 0.0256	true source BTLSM feature-specific-KL, 0.0308	BTLSD feature-specific-KL, 0.0159	true source, 0.0385	learned source, 0.0155	true source, 0.0377

From top to bottom, average ECE ranks from low to high. Green: ECE of learned source. Yellow: ECE of learned target.

Table 7 Summative results for 10-fold experiments (source size=8000, target size=8000)

Measurement and comparison	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
learned source model > learned target model	no	no	no	no	no	yes
Hypothesis 1: BTLSM (prior model: learned source model) > learned target	none	none	none	none	none	unadjusted, KL, feature-specific KL, ratio
Hypothesis 1: BTLSM (prior model: true source model) > learned target	none	none	none	none	none	unadjusted, KL, ratio
Hypothesis 1: BTLSD > learned target	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	none	none	unadjusted, KL, feature-specific KL, ratio
Hypothesis 2: BTLSM (prior model: learned source model) > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, feature-specific KL, ratio
Hypothesis 2: BTLSD > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none
true source > learned target	no	no	no	no	no	no
BTLSM (prior model: true source model) > true source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio
BTLSD > true source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio

Only statistically significantly better results are mentioned ($p<0.05$) in this table.

Table 8 Average AUC and confidence interval of models in 10-fold experiments (source size=8000, target size=8000)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target 0.969 (0.9669-0.9711)[123]	true target 0.969 (0.9669-0.9711)[123]	true target 0.9643 (0.9612-0.9674)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.8053 (0.7808-0.8299)[123]
BTLSD unadjusted 0.9644 (0.9621-0.9668)[123]	BTLSD KL 0.964 (0.9617-0.9663)[123]	learned target 0.9591 (0.9558-0.9624)[13]	learned target 0.9647 (0.9576-0.9717)[13]	BTLSD KL 0.9661 (0.9596-0.9726)[13]	BTLSD unadjusted 0.7724 (0.7453-0.7994)[23]
BTLSD KL 0.9644 (0.962-0.9667)[123]	BTLSD feature-specific-KL 0.9636 (0.9613-0.966)[123]	BTLSD KL 0.9581 (0.9548-0.9614)[13]	BTLSD KL 0.9585 (0.952-0.965)[123]	BTLSD unadjusted 0.9655 (0.959-0.972) [13]	BTLSD ratio 0.7724 (0.7453-0.7994)[23]
BTLSD feature-specific-KL 0.9644 (0.9621-0.9668)[123]	BTLSD unadjusted 0.9632 (0.9608-0.9656)[123]	BTLSD unadjusted 0.9565 (0.9532-0.9599)[123]	BTLSD feature-specific-KL 0.9558 (0.9494-0.9623)[123]	BTLSD feature-specific-KL 0.9655 (0.959-0.972)[13]	BTLSD feature-specific-KL 0.7721 (0.7451-0.7992)[23]
BTLSD ratio 0.9644 (0.9621-0.9668)[123]	BTLSD ratio 0.9632 (0.9608-0.9656)[123]	BTLSD feature-specific-KL 0.9565 (0.9532-0.9599)[123]	BTLSD unadjusted 0.9556 (0.9491-0.9621)[123]	BTLSD ratio 0.9655 (0.959-0.972)[13]	learned source BTLSM feature-specific-KL 0.7707 (0.7437-0.7976)[123]
true source BTLSM KL 0.9622 (0.9597-0.9646)[13]	learned target 0.9618 (0.9594-0.9643)[13]	BTLSD ratio 0.9565 (0.9532-0.9599)[123]	BTLSD ratio 0.9556 (0.9491-0.9621)[123]	learned target 0.9647 (0.9576-0.9717)[13]	learned source BTLSM unadjusted 0.7706 (0.7436-0.7975)[123]
learned target 0.9618 (0.9594-0.9643)[13]	learned source BTLSM KL 0.9599 (0.9573-0.9625)[123]	true source 0.9459 (0.9417-0.9501)[2]	learned source BTLSM KL 0.9549 (0.9458-0.9641)[123]	true source BTLSM KL 0.961 (0.9532-0.9689)[13]	learned source BTLSM ratio 0.7706 (0.7436-0.7975)[123]
true source BTLSM feature-specific-KL 0.9613 (0.9588-0.9638)[13]	learned source BTLSM unadjusted 0.9592 (0.9565-0.9618)[123]	true source BTLSM KL 0.9458 (0.9419-0.9498)[2]	learned source BTLSM unadjusted 0.9536 (0.9441-0.9631)[123]	true source BTLSM unadjusted 0.9605 (0.9524-0.9686)[13]	BTLSD KL 0.7696 (0.7423-0.7968)[23]
learned source BTLSM KL 0.9611 (0.9585-0.9636)[13]	learned source BTLSM feature-specific-KL 0.9592 (0.9566-0.9619)[123]	learned source BTLSM unadjusted 0.9456 (0.9417-0.9495)[2]	learned source BTLSM ratio 0.9536 (0.9441-0.9631)[123]	true source BTLSM feature-specific-KL 0.9605 (0.9524-0.9687)[13]	learned source BTLSM KL 0.7667 (0.7394-0.794)[23]
true source BTLSM unadjusted 0.961 (0.9585-0.9635)[13]	learned source BTLSM ratio 0.9592 (0.9565-0.9618)[123]	learned source BTLSM feature-specific-KL 0.9456 (0.9417-0.9495)[2]	learned source BTLSM feature-specific-KL 0.9534 (0.9439-0.9629)[123]	true source BTLSM ratio 0.9605 (0.9524-0.9686)[13]	true source BTLSM unadjusted 0.761 (0.7323-0.7897)[23]

Table 8 Average AUC and confidence interval of models in 10-fold experiments (source size=8000, target size=8000) (continued)

true source BTLSM ratio 0.961 (0.9585-0.9635)[13]	true source BTLSM unadjusted 0.9234 (0.919-0.9278)[123]	learned source BTLSM ratio 0.9456 (0.9417-0.9495)[2]	true source BTLSM KL 0.9528 (0.944-0.9615)[123]	learned source BTLSM KL 0.9567 (0.9473-0.9662)[23]	true source BTLSM ratio 0.761 (0.7323-0.7897)[23]
learned source BTLSM feature-specific-KL 0.9601 (0.9576-0.9627)[123]	true source BTLSM feature-specific-KL 0.9234 (0.919-0.9278)[123]	true source BTLSM unadjusted 0.9455 (0.9415-0.9494)[2]	true source BTLSM unadjusted 0.9455 (0.9372-0.9539)[123]	learned source BTLSM unadjusted 0.9542 (0.9445-0.9639)[2]	true source BTLSM KL 0.7598 (0.7314-0.7883)[23]
learned source BTLSM unadjusted 0.9598 (0.9572-0.9624)[123]	true source BTLSM ratio 0.9234 (0.919-0.9278)[123]	true source BTLSM ratio 0.9455 (0.9415-0.9494)[2]	true source BTLSM ratio 0.9455 (0.9372-0.9539)[123]	learned source BTLSM ratio 0.9542 (0.9445-0.9639)[2]	true source BTLSM feature-specific-KL 0.7595 (0.7308-0.7882)[3]
learned source BTLSM ratio 0.9598 (0.9572-0.9624)[123]	true source BTLSM KL 0.9227 (0.9182-0.9272)[123]	true source BTLSM feature-specific-KL 0.9454 (0.9415-0.9494)[2]	true source BTLSM feature-specific-KL 0.9452 (0.9364-0.954)[123]	learned source BTLSM feature-specific-KL 0.9541 (0.9444-0.9639)[2]	learned source 0.759 (0.7318-0.7861)[23]
true source 0.9255 (0.9215-0.9296)[2]	learned source 0.9078 (0.9035-0.912)[23]	learned source BTLSM KL 0.9451 (0.9412-0.9491)[2]	learned source 0.9183 (0.905-0.9316)[23]	learned source 0.9498 (0.9396-0.9599)[2]	learned target 0.7412 (0.7129-0.7694)[1]
learned source 0.924 (0.9197-0.9283)[2]	true source 0.901 (0.8953-0.9066)[12]	learned source 0.9447 (0.9406-0.9488)[2]	true source 0.7086 (0.6805-0.7366)[12]	true source 0.9455 (0.934-0.957)[2]	true source 0.7297 (0.7001-0.7593)[1]

From top to bottom, average AUC ranks from high to low. Green: AUC of learned source. Yellow: AUC of learned target.

1. significantly different from learned source model ($p<0.05$)
2. significantly different from learned target model ($p<0.05$)
3. significantly different from true source model ($p<0.05$)

Table 9 Average expected calibration error for models in 10-fold experiments (source size=8000, target size=8000)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target, 0.001	true target, 0.001	true target, 0.0004	true target, 0.0007	true target, 0.0007	true target, 0.0013
learned source BTLSM KL, 0.0016	learned source BTLSM KL, 0.0023	learned source BTLSM KL, 0.0027	learned target, 0.0011	learned target, 0.0011	learned target, 0.0027
true source BTLSM KL, 0.0017	learned target, 0.003	learned source BTLSM unadjusted, 0.0033	true source BTLSM KL, 0.0018	BTLSD KL, 0.0025	true source BTLSM KL, 0.0046
BTLSD KL, 0.0029	learned source BTLSM feature-specific-KL, 0.0045	learned source BTLSM feature-specific-KL, 0.0033	learned source BTLSM KL, 0.0031	BTLSD feature-specific-KL, 0.0029	BTLSD KL, 0.0072
learned target, 0.003	learned source BTLSM unadjusted, 0.0046	learned source BTLSM ratio, 0.0033	BTLSD KL, 0.0031	BTLSD unadjusted, 0.003	learned source BTLSM KL, 0.0086
learned source BTLSM unadjusted, 0.0035	learned source BTLSM ratio, 0.0046	true source BTLSM KL, 0.0045	true source BTLSM feature-specific-KL, 0.004	BTLSD ratio, 0.003	BTLSD feature-specific-KL, 0.0128
learned source BTLSM ratio, 0.0035	BTLSD KL, 0.0092	BTLSD KL, 0.0049	learned source BTLSM unadjusted, 0.0043	true source BTLSM KL, 0.0042	true source BTLSM feature-specific-KL, 0.0134
true source BTLSM unadjusted, 0.0038	true source BTLSM KL, 0.0102	true source BTLSM unadjusted, 0.0051	learned source BTLSM feature-specific-KL, 0.0043	true source BTLSM unadjusted, 0.0049	BTLSD unadjusted, 0.0134
true source BTLSM ratio, 0.0038	true source BTLSM feature-specific-KL, 0.0129	true source BTLSM feature-specific-KL, 0.0051	learned source BTLSM ratio, 0.0043	true source BTLSM feature-specific-KL, 0.0049	BTLSD ratio, 0.0134
learned source BTLSM feature-specific-KL, 0.0043	true source BTLSM unadjusted, 0.0134	true source BTLSM ratio, 0.0051	true source BTLSM unadjusted, 0.0044	true source BTLSM ratio, 0.0049	learned source BTLSM feature-specific-KL, 0.0138
true source BTLSM feature-specific-KL, 0.0049	true source BTLSM ratio, 0.0134	learned target, 0.0053	true source BTLSM ratio, 0.0044	learned source BTLSM KL, 0.0053	learned source BTLSM unadjusted, 0.0143
BTLSD feature-specific-KL, 0.008	BTLSD feature-specific-KL, 0.0142	BTLSD unadjusted, 0.0059	BTLSD feature-specific-KL, 0.005	learned source BTLSM unadjusted, 0.0058	learned source BTLSM ratio, 0.0143

Table 9 Average expected calibration error for models in 10-fold experiments (source size=8000, target size=8000) (continued)

BTLSD unadjusted, 0.0081	BTLSD unadjusted, 0.0147	BTLSD feature-specific-KL, 0.0059	BTLSD unadjusted, 0.0051	learned source BTLSM feature-specific-KL, 0.0058	true source BTLSM unadjusted, 0.0147
BTLSD ratio, 0.0081	BTLSD ratio, 0.0147	BTLSD ratio, 0.0059	BTLSD ratio, 0.0051	learned source BTLSM ratio, 0.0058	true source BTLSM ratio, 0.0147
true source, 0.0225	learned source, 0.0296	true source, 0.0126	learned source, 0.0216	true source, 0.0126	learned source, 0.0276
learned source, 0.0234	true source, 0.03	learned source, 0.0143	true source, 0.0385	learned source, 0.0155	true source, 0.0377

From top to bottom, average ECE ranks from low to high. Green: ECE of learned source. Yellow: ECE of learned target.

Table 10 Summative results for 10-fold experiments (source size=50, target size=50)

Measurement and comparison	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
learned source model > learned target model	no	no	no	no	no	no
Hypothesis 1: BTLSM (prior model: learned source model) > learned target	none	unadjusted, feature-specific KL, ratio	none	none	none	none
Hypothesis 1: BTLSM (prior model: true source model) > learned target	unadjusted, KL, feature-specific KL	none	unadjusted, KL, feature-specific KL	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, feature-specific KL, ratio
Hypothesis 1: BTLSD > learned target	none	none	none	none	none	unadjusted, feature-specific KL, ratio
Hypothesis 2: BTLSM (prior model: learned source model) > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio
Hypothesis 2: BTLSD > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio
true source > learned target	yes	no	yes	no	no	yes
BTLSM (prior model: true source model) > true source	none	none	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL	none
BTLSD > true source	none	none	none	unadjusted, KL, feature-specific KL, ratio	none	none

Only statistically significantly better results are mentioned ($p<0.05$) in this table.

Table 11 Average AUC and confidence interval of models in 10-fold experiments (source size=50, target size=50)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target 0.969 (0.9669-0.9711)[123]	true target 0.969 (0.9669-0.9711)[123]	true target 0.9643 (0.9612-0.9674)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.8053 (0.7808-0.8299)[123]
true source BTLSM KL 0.928 (0.9241-0.932)[12]	learned source BTLSM feature-specific-KL 0.9018 (0.8973-0.9064)[12]	true source 0.9459 (0.9417-0.9501)[12]	learned target 0.9413 (0.9329-0.9497)[13]	true source BTLSM unadjusted 0.9546 (0.947-0.9621)[123]	true source 0.7297 (0.7001-0.7593)[12]
true source BTLSM unadjusted 0.9274 (0.9235-0.9314)[12]	learned source BTLSM unadjusted 0.9016 (0.897-0.9062)[12]	true source BTLSM unadjusted 0.9433 (0.9391-0.9476)[123]	learned source BTLSM unadjusted 0.9237 (0.9132-0.9342)[123]	true source BTLSM feature-specific-KL 0.9545 (0.9469-0.9621)[123]	true source BTLSM feature-specific-KL 0.7272 (0.6979-0.7565)[12]
true source BTLSM feature-specific-KL 0.9258 (0.9218-0.9299)[12]	learned source BTLSM ratio 0.9016 (0.897-0.9062)[12]	true source BTLSM feature-specific-KL 0.9433 (0.939-0.9475)[123]	learned source BTLSM ratio 0.9237 (0.9132-0.9342)[123]	true source BTLSM KL 0.9544 (0.9468-0.9621)[123]	true source BTLSM unadjusted 0.7262 (0.6969-0.7555)[12]
true source 0.9255 (0.9215-0.9296)[12]	true source 0.901 (0.8953-0.9066)[1]	true source BTLSM KL 0.9425 (0.9383-0.9468)[123]	learned source BTLSM feature-specific-KL 0.9226 (0.9118-0.9333)[123]	true source BTLSM ratio 0.9498 (0.942-0.9577)[12]	BTLSD unadjusted 0.7222 (0.6948-0.7496)[12]
learned target 0.8955 (0.8904-0.9005)[13]	learned target 0.8955 (0.8904-0.9005)[1]	true source BTLSM ratio 0.8866 (0.8804-0.8928)[3]	learned source BTLSM KL 0.9189 (0.9079-0.93)[123]	true source 0.9455 (0.934-0.957)[1]	BTLSD ratio 0.7222 (0.6948-0.7496)[12]
learned source BTLSM unadjusted 0.8944 (0.8895-0.8993)[13]	true source BTLSM unadjusted 0.889 (0.8835-0.8946)[123]	learned source 0.8837 (0.8771-0.8903)[3]	true source BTLSM unadjusted 0.8999 (0.887-0.9128)[123]	learned target 0.9413 (0.9329-0.9497)[1]	BTLSD feature-specific-KL 0.7216 (0.6937-0.7495)[12]
learned source BTLSM ratio 0.8944 (0.8895-0.8993)[13]	true source BTLSM feature-specific-KL 0.8887 (0.8832-0.8943)[123]	learned target 0.8819 (0.8757-0.8882)[3]	true source BTLSM feature-specific-KL 0.8984 (0.8856-0.9113)[123]	learned source BTLSM unadjusted 0.9208 (0.9093-0.9323)[123]	true source BTLSM ratio 0.7215 (0.6937-0.7493)[12]
learned source BTLSM feature-specific-KL 0.8906 (0.8855-0.8957)[13]	learned source BTLSM KL 0.8877 (0.882-0.8935)[123]	learned source BTLSM unadjusted 0.8793 (0.8731-0.8855)[3]	BTLSD feature-specific-KL 0.8951 (0.8827-0.9075)[123]	learned source BTLSM ratio 0.9208 (0.9093-0.9323)[123]	true source BTLSM KL 0.7107 (0.6806-0.7408)[13]

Table 11 Average AUC and confidence interval of models in 10-fold experiments (source size=50, target size=50) (continued)

BTLSD feature-specific-KL 0.8869 (0.8823-0.8916)[123]	true source BTLSM KL 0.8841 (0.8783-0.8898)[123]	learned source BTLSM ratio 0.8793 (0.8731-0.8855)[3]	BTLSD unadjusted 0.8939 (0.8812-0.9066)[123]	learned source BTLSM feature-specific-KL 0.9206 (0.909-0.9321)[123]	learned source BTLSM unadjusted 0.7041 (0.6767-0.7314)[1]
BTLSD unadjusted 0.885 (0.8802-0.8898)[123]	BTLSD unadjusted 0.8796 (0.8746-0.8846)[123]	learned source BTLSM feature-specific-KL 0.8775 (0.8713-0.8838)[3]	BTLSD ratio 0.8939 (0.8812-0.9066)[123]	learned source BTLSM KL 0.919 (0.9074-0.9307)[123]	learned source BTLSM ratio 0.7041 (0.6767-0.7314)[1]
BTLSD ratio 0.885 (0.8802-0.8898)[123]	BTLSD ratio 0.8796 (0.8746-0.8846)[123]	learned source BTLSM KL 0.8681 (0.8618-0.8744)[123]	BTLSD KL 0.8917 (0.8788-0.9045)[123]	BTLSD feature-specific-KL 0.8915 (0.8779-0.9051)[23]	learned source BTLSM feature-specific-KL 0.7021 (0.6744-0.7299)[13]
learned source BTLSM KL 0.8809 (0.8752-0.8865)[123]	BTLSD feature-specific-KL 0.8795 (0.8745-0.8845)[123]	BTLSD unadjusted 0.8402 (0.8327-0.8478)[123]	true source BTLSM KL 0.8915 (0.8777-0.9054)[123]	BTLSD KL 0.8899 (0.8763-0.9034)[23]	BTLSD KL 0.6986 (0.6719-0.7253)[13]
BTLSD KL 0.8752 (0.8703-0.8801)[123]	BTLSD KL 0.8753 (0.8704-0.8802)[23]	BTLSD ratio 0.8402 (0.8327-0.8478)[123]	true source BTLSM ratio 0.8748 (0.8594-0.8903)[123]	BTLSD unadjusted 0.8887 (0.8752-0.9022)[23]	learned source BTLSM KL 0.6892 (0.6611-0.7173)[13]
true source BTLSM ratio 0.8706 (0.8636-0.8777)[123]	learned source 0.8713 (0.8664-0.8762)[23]	BTLSD feature-specific-KL 0.8396 (0.8321-0.847)[123]	learned source 0.8306 (0.8128-0.8484)[23]	BTLSD ratio 0.8887 (0.8752-0.9022)[23]	learned target 0.6868 (0.661-0.7125)[13]
learned source 0.8113 (0.8041-0.8184)[23]	true source BTLSM ratio 0.8477 (0.8402-0.8553)[123]	BTLSD KL 0.8136 (0.8057-0.8216)[123]	true source 0.7086 (0.6805-0.7366)[12]	learned source 0.8875 (0.8738-0.9013)[23]	learned source 0.6613 (0.6322-0.6903)[23]

From top to bottom, average AUC ranks from high to low. Green: AUC of learned source. Yellow: AUC of learned target.

1. significantly different from learned source model ($p<0.05$)
2. significantly different from learned target model ($p<0.05$)
3. significantly different from true source model ($p<0.05$)

Table 12 Average expected calibration error for models in 10-fold experiments (source size=50, target size=50)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target, 0.001	true target, 0.001	true target, 0.0004	true target, 0.0007	true target, 0.0007	true target, 0.0013
BTLSD unadjusted, 0.0103	BTLSD feature-specific-KL, 0.0146	learned target, 0.0085	learned target, 0.0021	learned target, 0.0021	learned target, 0.0069
BTLSD ratio, 0.0103	BTLSD KL, 0.0158	learned source BTLSM unadjusted, 0.0088	learned source BTLSM KL, 0.004	learned source BTLSM KL, 0.0041	learned source BTLSM KL, 0.0081
BTLSD feature-specific-KL, 0.0126	BTLSD unadjusted, 0.0161	learned source BTLSM ratio, 0.0088	learned source BTLSM feature-specific-KL, 0.0044	learned source BTLSM unadjusted, 0.0041	learned source BTLSM unadjusted, 0.012
learned source, 0.0136	BTLSD ratio, 0.0161	learned source BTLSM feature-specific-KL, 0.0092	learned source BTLSM unadjusted, 0.0046	learned source BTLSM feature-specific-KL, 0.0041	learned source BTLSM ratio, 0.012
learned source BTLSM KL, 0.016	learned target, 0.017	true source BTLSM ratio, 0.0098	learned source BTLSM ratio, 0.0046	learned source BTLSM ratio, 0.0041	learned source BTLSM feature-specific-KL, 0.0131
BTLSD KL, 0.0161	learned source BTLSM unadjusted, 0.0181	BTLSD unadjusted, 0.0098	BTLSD feature-specific-KL, 0.0055	BTLSD unadjusted, 0.0075	BTLSD unadjusted, 0.0147
learned source BTLSM unadjusted, 0.0166	learned source BTLSM ratio, 0.0181	BTLSD ratio, 0.0098	true source BTLSM ratio, 0.0056	BTLSD ratio, 0.0075	BTLSD ratio, 0.0147
learned source BTLSM ratio, 0.0166	learned source BTLSM feature-specific-KL, 0.0187	BTLSD feature-specific-KL, 0.0106	BTLSD unadjusted, 0.0057	learned source, 0.0076	BTLSD KL, 0.0148
learned target, 0.017	learned source BTLSM KL, 0.0196	learned source, 0.0109	BTLSD ratio, 0.0057	BTLSD feature-specific-KL, 0.0076	BTLSD feature-specific-KL, 0.0152
learned source BTLSM feature-specific-KL, 0.017	learned source, 0.0204	true source, 0.0126	true source BTLSM unadjusted, 0.008	BTLSD KL, 0.0081	true source BTLSM ratio, 0.0156
true source BTLSM ratio, 0.0218	true source BTLSM ratio, 0.0252	true source BTLSM KL, 0.0135	true source BTLSM feature-specific-KL, 0.0081	true source BTLSM ratio, 0.0085	true source BTLSM KL, 0.0205
true source BTLSM KL, 0.0223	true source, 0.03	true source BTLSM feature-specific-KL, 0.0136	BTLSD KL, 0.0084	true source BTLSM KL, 0.0093	true source BTLSM feature-specific-KL, 0.0246
true source, 0.0225	true source BTLSM KL, 0.0305	true source BTLSM unadjusted, 0.0137	true source BTLSM KL, 0.0086	true source BTLSM unadjusted, 0.0094	true source BTLSM unadjusted, 0.025

Table 12 Average expected calibration error for models in 10-fold experiments (source size=50, target size=50) (continued)

true source BTLSM feature-specific-KL, 0.023	true source BTLSM unadjusted, 0.0308	learned source BTLSM KL, 0.0141	learned source, 0.0273	true source BTLSM feature-specific-KL, 0.0094	learned source, 0.0373
true source BTLSM unadjusted, 0.0231	true source BTLSM feature-specific-KL, 0.0308	BTLSD KL, 0.0142	true source, 0.0385	true source, 0.0126	true source, 0.0377

From top to bottom, average ECE ranks from low to high. Green: ECE of learned source. Yellow: ECE of learned target.

Table 13 Summative results for 10-fold experiments (source size=50, target size=8000)

Measurement and comparison	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
learned source model > learned target model	no	no	no	no	no	no
Hypothesis 1: BTLSM (prior model: learned source model) > learned target	none	none	none	none	none	unadjusted, KL, feature-specific KL, ratio
Hypothesis 1: BTLSM (prior model: true source model) > learned target	none	none	none	none	none	unadjusted, KL, ratio
Hypothesis 1: BTLSD > learned target	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	none	none	none
Hypothesis 2: BTLSM (prior model: learned source model) > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio
Hypothesis 2: BTLSD > learned source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio
true source > learned target	no	no	no	no	no	no
BTLSM (prior model: true source model) > true source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio
BTLSD > true source	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	unadjusted, KL, feature-specific KL, ratio	none

Only statistically significantly better results are mentioned ($p<0.05$) in this table.

Table 14 Average AUC and confidence interval of models in 10-fold experiments (source size=50, target size=8000)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target 0.969 (0.9669-0.9711)[123]	true target 0.969 (0.9669-0.9711)[123]	true target 0.9643 (0.9612-0.9674)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.9742 (0.9686-0.9798)[123]	true target 0.8053 (0.7808-0.8299)[123]
BTLSD KL 0.9646 (0.9622-0.9669)[123]	BTLSD unadjusted 0.9646 (0.9622-0.9669)[123]	BTLSD KL 0.96 (0.9567-0.9632)[13]	BTLSD feature-specific-KL 0.9652 (0.9586-0.9718)[13]	learned target 0.9647 (0.9576-0.9717)[13]	true source BTLSM unadjusted 0.761 (0.7323-0.7897)[123]
BTLSD feature-specific-KL 0.9646 (0.9623-0.9669)[123]	BTLSD KL 0.9646 (0.9622-0.9669)[123]	BTLSD unadjusted 0.9599 (0.9567-0.9632)[13]	BTLSD unadjusted 0.9648 (0.9583-0.9714)[13]	BTLSD unadjusted 0.9642 (0.9577-0.9708)[13]	true source BTLSM ratio 0.761 (0.7323-0.7897)[123]
BTLSD unadjusted 0.9645 (0.9622-0.9668)[123]	BTLSD feature-specific-KL 0.9646 (0.9622-0.9669)[123]	BTLSD feature-specific-KL 0.9599 (0.9567-0.9632)[13]	BTLSD ratio 0.9648 (0.9583-0.9714)[13]	BTLSD feature-specific-KL 0.9642 (0.9576-0.9708)[13]	true source BTLSM KL 0.7598 (0.7314-0.7883)[123]
BTLSD ratio 0.9645 (0.9622-0.9668)[123]	BTLSD ratio 0.9646 (0.9622-0.9669)[123]	BTLSD ratio 0.9599 (0.9567-0.9632)[13]	learned target 0.9647 (0.9576-0.9717)[13]	BTLSD ratio 0.9642 (0.9577-0.9708)[13]	true source BTLSM feature-specific-KL 0.7595 (0.7308-0.7882)[13]
true source BTLSM KL 0.9622 (0.9597-0.9646)[13]	learned target 0.9618 (0.9594-0.9643)[13]	learned target 0.9591 (0.9558-0.9624)[13]	BTLSD KL 0.9641 (0.9575-0.9708)[13]	BTLSD KL 0.9641 (0.9575-0.9708)[13]	learned source BTLSM unadjusted 0.7589 (0.7305-0.7874)[123]
learned source BTLSM unadjusted 0.962 (0.9596-0.9645)[13]	learned source BTLSM KL 0.9616 (0.959-0.9641)[13]	learned source BTLSM unadjusted 0.9508 (0.9471-0.9546)[123]	learned source BTLSM unadjusted 0.96 (0.9522-0.9677)[123]	true source BTLSM KL 0.961 (0.9532-0.9689)[13]	learned source BTLSM ratio 0.7589 (0.7305-0.7874)[123]
learned source BTLSM feature-specific-KL 0.962 (0.9595-0.9644)[13]	learned source BTLSM unadjusted 0.9616 (0.9591-0.9641)[13]	learned source BTLSM feature-specific-KL 0.9508 (0.9471-0.9546)[123]	learned source BTLSM ratio 0.96 (0.9522-0.9677)[123]	true source BTLSM unadjusted 0.9605 (0.9524-0.9686)[13]	learned source BTLSM feature-specific-KL 0.7586 (0.7302-0.7871)[123]
learned source BTLSM ratio 0.962 (0.9596-0.9645)[13]	learned source BTLSM feature-specific-KL 0.9616 (0.9591-0.9641)[13]	learned source BTLSM ratio 0.9508 (0.9471-0.9546)[123]	learned source BTLSM feature-specific-KL 0.9595 (0.9516-0.9674)[123]	true source BTLSM feature-specific-KL 0.9605 (0.9524-0.9687)[13]	learned source BTLSM KL 0.7577 (0.7293-0.7861)[123]
learned target 0.9618 (0.9594-0.9643)[13]	learned source BTLSM ratio 0.9616 (0.9591-0.9641)[13]	learned source BTLSM KL 0.9495 (0.9457-0.9533)[123]	learned source BTLSM KL 0.9588 (0.9507-0.967)[123]	true source BTLSM ratio 0.9605 (0.9524-0.9686)[13]	BTLSD unadjusted 0.7544 (0.7271-0.7818)[1]

Table 14 Average AUC and confidence interval of models in 10-fold experiments (source size=50, target size=8000) (continued)

learned source BTLSM KL 0.9618 (0.9594-0.9643)[13]	true source BTLSM unadjusted 0.9234 (0.919-0.9278)[123]	true source 0.9459 (0.9417-0.9501)[12]	true source BTLSM KL 0.9528 (0.944-0.9615)[123]	learned source BTLSM feature-specific-KL 0.9584 (0.9498-0.967)[123]	BTLSD ratio 0.7544 (0.7271-0.7818)[1]
true source BTLSM feature-specific-KL 0.9613 (0.9588-0.9638)[13]	true source BTLSM feature-specific-KL 0.9234 (0.919-0.9278)[123]	true source BTLSM KL 0.9458 (0.9419-0.9498)[12]	true source BTLSM unadjusted 0.9455 (0.9372-0.9539)[123]	learned source BTLSM unadjusted 0.9582 (0.9495-0.9669)[123]	BTLSD feature-specific-KL 0.7542 (0.727-0.7815)[1]
true source BTLSM unadjusted 0.961 (0.9585-0.9635)[13]	true source BTLSM ratio 0.9234 (0.919-0.9278)[123]	true source BTLSM unadjusted 0.9455 (0.9415-0.9494)[12]	true source BTLSM ratio 0.9455 (0.9372-0.9539)[123]	learned source BTLSM ratio 0.9582 (0.9495-0.9669)[123]	BTLSD KL 0.7533 (0.7262-0.7803)[1]
true source BTLSM ratio 0.961 (0.9585-0.9635)[13]	true source BTLSM KL 0.9227 (0.9182-0.9272)[123]	true source BTLSM ratio 0.9455 (0.9415-0.9494)[12]	true source BTLSM feature-specific-KL 0.9452 (0.9364-0.954)[123]	learned source BTLSM KL 0.9576 (0.949-0.9662)[123]	learned target 0.7412 (0.7129-0.7694)[1]
true source 0.9255 (0.9215-0.9296)[12]	true source 0.901 (0.8953-0.9066)[12]	true source BTLSM feature-specific-KL 0.9454 (0.9415-0.9494)[12]	learned source 0.8306 (0.8128-0.8484)[23]	true source 0.9455 (0.934-0.957)[12]	true source 0.7297 (0.7001-0.7593)[1]
learned source 0.8113 (0.8041-0.8184)[23]	learned source 0.8713 (0.8664-0.8762)[23]	learned source 0.8837 (0.8771-0.8903)[23]	true source 0.7086 (0.6805-0.7366)[12]	learned source 0.8875 (0.8738-0.9013)[23]	learned source 0.6613 (0.6322-0.6903)[23]

From top to bottom, average AUC ranks from high to low. Green: AUC of learned source. Yellow: AUC of learned target.

1. significantly different from learned source model ($p<0.05$)
2. significantly different from learned target model ($p<0.05$)
3. significantly different from true source model ($p<0.05$)

Table 15 Average expected calibration error for models in 10-fold experiments (source size=50, target size=8000)

Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6
true target, 0.001	true target, 0.001	true target, 0.0004	learned source BTLSM KL, 0.0005	true target, 0.0007	true target, 0.0013
BTLSD unadjusted, 0.0011	BTLSD unadjusted, 0.0011	BTLSD KL, 0.0015	learned source BTLSM feature-specific-KL, 0.0005	learned source BTLSM KL, 0.0007	learned source BTLSM feature-specific-KL, 0.0016
BTLSD ratio, 0.0011	BTLSD feature-specific-KL, 0.0011	BTLSD unadjusted, 0.0016	learned source BTLSM unadjusted, 0.0006	learned source BTLSM unadjusted, 0.0007	learned source BTLSM unadjusted, 0.0017
BTLSD KL, 0.0012	BTLSD ratio, 0.0011	BTLSD feature-specific-KL, 0.0016	learned source BTLSM ratio, 0.0006	learned source BTLSM feature-specific-KL, 0.0007	learned source BTLSM ratio, 0.0017
BTLSD feature-specific-KL, 0.0012	BTLSD KL, 0.0012	BTLSD ratio, 0.0016	true target, 0.0007	learned source BTLSM ratio, 0.0007	learned source BTLSM KL, 0.0018
true source BTLSM KL, 0.0017	learned source BTLSM KL, 0.002	learned source BTLSM KL, 0.0043	BTLSD feature-specific-KL, 0.001	BTLSD unadjusted, 0.001	BTLSD unadjusted, 0.002
learned source BTLSM unadjusted, 0.0017	learned source BTLSM unadjusted, 0.002	true source BTLSM KL, 0.0045	learned target, 0.0011	BTLSD feature-specific-KL, 0.001	BTLSD KL, 0.002
learned source BTLSM feature-specific-KL, 0.0017	learned source BTLSM feature-specific-KL, 0.002	learned source BTLSM unadjusted, 0.0045	BTLSD unadjusted, 0.0011	BTLSD ratio, 0.001	BTLSD feature-specific-KL, 0.002
learned source BTLSM ratio, 0.0017	learned source BTLSM ratio, 0.002	learned source BTLSM feature-specific-KL, 0.0045	BTLSD KL, 0.0011	learned target, 0.0011	BTLSD ratio, 0.002
learned source BTLSM KL, 0.0018	learned target, 0.003	learned source BTLSM ratio, 0.0045	BTLSD ratio, 0.0011	BTLSD KL, 0.0011	learned target, 0.0027
learned target, 0.003	true source BTLSM KL, 0.0102	true source BTLSM unadjusted, 0.0051	true source BTLSM KL, 0.0018	true source BTLSM KL, 0.0042	true source BTLSM KL, 0.0046
true source BTLSM unadjusted, 0.0038	true source BTLSM feature-specific-KL, 0.0129	true source BTLSM feature-specific-KL, 0.0051	true source BTLSM feature-specific-KL, 0.004	true source BTLSM unadjusted, 0.0049	true source BTLSM feature-specific-KL, 0.0134
true source BTLSM ratio, 0.0038	true source BTLSM unadjusted, 0.0134	true source BTLSM ratio, 0.0051	true source BTLSM unadjusted, 0.0044	true source BTLSM feature-specific-KL, 0.0049	true source BTLSM unadjusted, 0.0147
true source BTLSM feature-specific-KL, 0.0049	true source BTLSM ratio, 0.0134	learned target, 0.0053	true source BTLSM ratio, 0.0044	true source BTLSM ratio, 0.0049	true source BTLSM ratio, 0.0147

Table 15 Average expected calibration error for models in 10-fold experiments (source size=50, target size=8000) (continued)

learned source, 0.0136	learned source, 0.0204	learned source, 0.0109	learned source, 0.0273	learned source, 0.0076	learned source, 0.0373
true source, 0.0225	true source, 0.03	true source, 0.0126	true source, 0.0385	true source, 0.0126	true source, 0.0377

From top to bottom, average ECE ranks from low to high. Green: ECE of learned source. Yellow: ECE of learned target.

4.1.3 Discussion: Transfer Learning for Different Scenarios

In this research, the first hypothesis is that transfer learning is better than using A model constructed with target data only when the two domains have similar distributions and the target has insufficient data. The results of synthetic influenza datasets show that transfer learning was useful in some scenarios and was unnecessary in other scenarios. For example, in scenario 2 and 6, transfer learning models performed statistically significantly better than the learned target models no matter the size of the source and target. In contrast, in scenario 4, transfer learning did not show any significant benefit even when the source size was large, and the target size was small. So, what was different among these scenarios?

Since each scenario was simulated with ground-truth source and target models, I compared these two models for each scenario (Table 3). Of all the scenarios, scenarios 2 and 6 each had the highest proportion of shared features among the feature union in the scenario (seven out of fifteen features were shared, 47%). In scenario 2, the target model had five target specific features. In scenario 6, the target model had two target specific features.

In the "non-transferrable" scenario, scenario 4, the true source model only reached an AUC of 0.7086 for the target test data, while the true target model AUC reached 0.9742. The source model only had 30% of features (six) shared between the source and the target. This small percent of shared features and their low discriminative ability for the classification task in the target may explain the non-transferability.

However, the results for scenario 1 show that when there was a low percent of sharing features, transfer learning could also be beneficial. For example, scenario 1 also had 30% shared features, but unlike scenario 4, it was not completely "non-transferrable" in all situations. It

showed benefits of transfer learning when the source size was 8000 (target size=50 or 8000). The difference between scenario 1 and the scenario 4 was that scenario 1 had fewer target specific features than scenario 4 (Figure 4). What is more, the true source model with these six features could reach an AUC of 0.9255 in scenario 1, where the true target model had an AUC of 0.969.

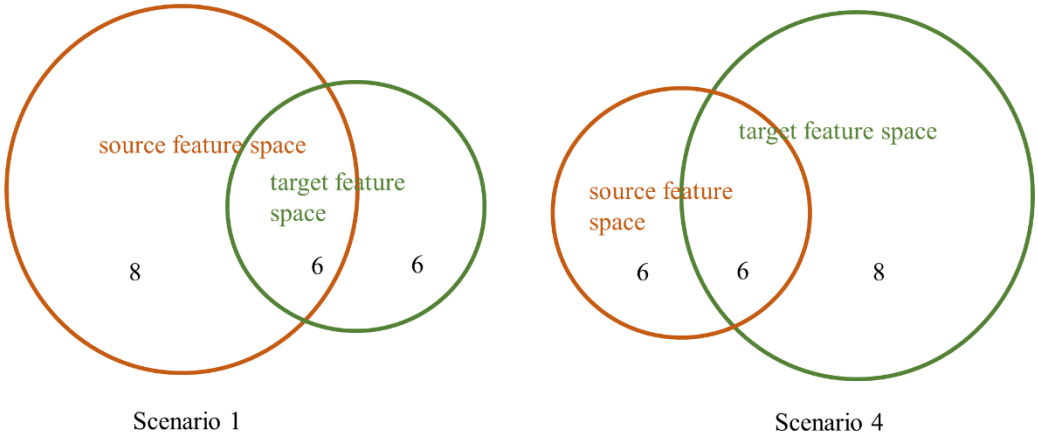

Figure 4 Comparison of feature spaces of the source model and the target model between scenario 1 and 4

Now, let us test the second hypothesis: transfer learning is better than direct adoption of the source model. This hypotheses is valid under some conditions, when the learned source model is not good enough, when the source and target's distribution difference is relatively large and there is a need to adjust the transferred source model, and when the target training data is large enough to make the right adjustment. In fact, these conditions can be common when the source size is too small to develop a well-performed model and the target size is large enough to make the adjustment.

My research results validated the second hypothesis in most scenarios. All of the BTLSM and BTLSD models significantly outperformed the learned source model in all six scenarios when the source size was 50 and target size was 8000. When the source size was increased to 8000, the BTLSM models and BTLSD models still outperformed the learned source model in all six scenarios, and most of differences were statistically significant. When the target size was reduced to 50, in the 50-source sample size situation, the BTLSM models and BTLSD models still outperformed the learned source model in five scenarios – all except scenario 3, where the learned source model performed slightly better than the learned target model. This may indicate a lack of enough target data to boost the source models' performance for this scenario. When the source size was 8000 and target size was 50, to my surprise, the BTLSM models were still better than the learned source model for scenarios 1, 2, 4, and 5. The BTLSD-KL model performed better in scenario 4.

Regarding calibration error, the transfer learning models' error usually was slightly higher than the learned target model. But the transfer learning models' error was always lower than the learned source models'.

4.2 Experiment 2: Intubation Network

4.2.1 Experiment Design

The influenza simulation experiment above tested the performance of transfer learning algorithms in heterogeneous scenarios where the feature spaces of the source and target domains were different. In this experiment, two factors impacted the similarity between target and source.

The first was the degree of overlapping features. The second was the distribution differences of the shared features.

My second experiment, the intubation simulation experiment, was designed to focus on the distribution differences. It studied how transfer learning algorithms would behave when the KL divergences between source and target distributions increased. In this experiment, I assumed the source and the target model to have the same feature space and Bayesian network structure. The only difference between the source and target model was the conditional probability tables. This scenario does occur in the real world, especially for the same institution. The classification task could change after some intervention or population drifting.

In the second experiment, I used the ALRAM network developed for monitoring patients in intensive care. It is often used as a gold-standard network to evaluate Bayesian network algorithms. For prediction purposes, I used a subgraph from the ALARM network, which includes the intubation node and nodes in the Markov Blanket of the intubation node (Figure 5). The subgraph has nine nodes and 12 arcs.

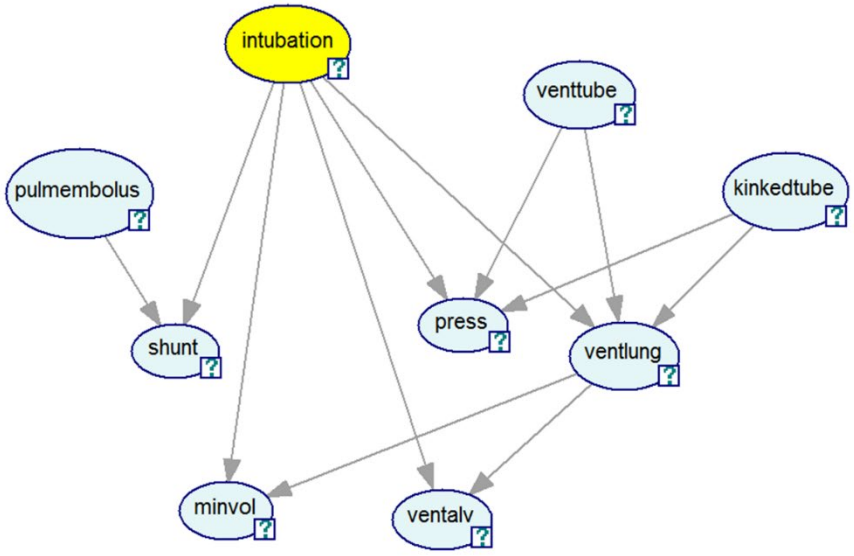

Figure 5 Intubation network (a subgraph of the ALARM network, (Beinlich et al. 1989))

The original expert-defined intubation network was used as the underlying target model. I assumed the underlying source model to have the same feature set and network structure as the target model. The source model was different from the target model in conditional probabilities. To study the relationship between source-target similarity and the performance of transfer learning algorithms, I simulated 140 different source models with different conditional probability tables.

The simulation procedure lists as follows (Eq. 18):

(1) Convert conditional probabilities into log odds

(2) Add some randomly generated noise to the log odds, where the noise follows a normal distribution, $N(0, \sigma^2)$. When σ is small, the generated noise has a smaller variance; when σ is large, the generated noise has a larger variance.

(3) Convert the new log odds back to condition probabilities

For example, if a variable *X* has three possible values, *v1*, *v2*, and *v3*, the conditional probabilities, when X's parent Pa(X)=j, can be written as: P(X=v1 | Pa(X)=j), P(X=v2 | Pa(X)=j), P(X=v3 | Pa(X)=j), and the sum of the three is one. Therefore, there are only two parameters.

Let P(X=v1 | Pa(X)=j)=p1 and P(X=v2 | Pa(X)=j)= p2 in the target model. First, I calculated log odd *r1* for *p1* and *r2* for *p2*. And then, I generated two random noise values, *k1*, *k2*, from a normal distribution N(0, σ^2). Then, I converted the new log odds, r_1' and r_2' to new probabilities and assigned them to the source model.

Equation 18

$$r_1 = \frac{p_1}{1-p_1} \qquad r_2 = \frac{p_2}{1-p_1-p_2}$$

$$r_1' = e^{\log r_1 + k_1} \quad r_2' = e^{\log r_2 + k_2}$$

$$p_1' = \frac{r_1'}{1+r_1'} \qquad p_2' = \frac{(1-p_1')r_2'}{1+r_2'} \qquad p_3' = 1 - p_1' - p_2'$$

With this procedure, I generated 140 source models, by simulating 10 times for each $\sigma \in$ {0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 1.5, 1.8, 2}. Adding in the situation when the source model is the same as the target model, there are 141 scenarios (141 pairs of ground truth source and target model). From the 141 source models, I generated 705 source datasets with different source dataset sizes (500, 1000, 2000, 4000, or 8000). Since I would like to focus on the situation when the target dataset size is small, all of the target training datasets had 50 samples. With these training datasets, I used the CFS approach or information gain (threshold 0.0001) and K2 to develop source and target models.

To get a reliable performance, the target testing dataset had 10,000 samples. Since the task variable has three values, normal, esophageal, and one-sided, I used weighted AUC to indicate the overall performance. The weighted AUC is a weighted sum of three AUC for binary classification

(normal or not, esophageal or not, one-sided or not), where the weight is the prevalence of the category (e.g., proportion of normal intubation samples in 10,000 samples).

4.2.2 Results: Comparisons of Different Approaches for Empirical KL Divergence Estimation

The goal of this section is to compare different approaches for empirical KL divergence estimation. The estimation can be challenging when the target distribution and the source distribution are unknown, and in transfer learning scenarios, it is common that the target distribution is unknown, and few target cases are available for estimation.

I calculated the true KL for the 705 different transfer learning scenarios in this intubation experiment. I also used different strategies to estimate KLs and calculated their differences from the true KL. Table 16 shows that the KL calculated using the target data and the learned source model was the closest to the true KL on average. Therefore, I chose this approach for KL estimation in the following intubation experiments. The KL calculated using the target data and the true source model had the smallest variance.

Table 16 Summary statistics of the differences between estimated KLs and true KLs in 705 runs

KL estimation approach	Mean	Standard Deviation	Minimum	25th Percentile	Median	75th Percentile	Maximum
average KL approach 1 (average KL, each KL was estimated using simulated data from true source and target models)	0.2509	0.1545	0.1458	0.1599	0.1790	0.2719	0.9344
average KL approach 2 (average KL, each KL was estimated using target cases and simulated source cases from true source model)	0.7255	0.4409	0.1434	0.4726	0.5804	0.8160	5.7707
average KL approach 3 (average KL, each KL was estimated using target cases and simulated source cases from learned source model)	0.1756	0.5166	-1.2934	0.0204	0.1375	0.2595	2.9951
KL estimated using learned source model and target training data	**0.1023**	0.3883	-1.3591	0.0155	0.1346	0.2436	1.4047
KL estimated using true source model and target training data	0.4620	0.1396	0.0398	0.4085	0.4491	0.4930	1.0906

4.2.3 Results: the Relationship between Transfer Learning and KL

One set of baseline models, the learned target models, were learned from 50 target training samples, using CFS feature selection and a K2 algorithm. Firstly, I tested whether the BTLSM models performed significantly better than the learned target models when the source size was large (8000) and the target size was small (50). The Wilcoxon signed rank tests on the weighted AUC (141 scenarios) showed that the BTLSM-unadjusted models performed statistically significantly better than the learned target models ($p<0.0001$). In addition, the BTLSM-KL, BTLSM-feature-specific-KL, and BTLSM-ratio models all performed statistically significantly better than the learned target models ($p<0.0001$).

Another set of baseline models, the learned source models, were learned from source training samples (500, 1000, 2000, 4000 or 8000), using CFS feature selection and K2 algorithm. They also performed significantly worse than the BTLSM models ($p<0.0001$) when the source size was large (8000) and the target size was small (50).

The performance of modeling strategy changed at different rates as KL increased. This is demonstrated by the lines with different slopes in the linear regression plots in Figures 6 and 7. The KL values on the horizontal axis were calculated using the joint probability distributions represented in the true source and target model. The performance of the learned source models dropped fastest as KL increased. The BTLSM-ratio models dropped more slowly than the other BTLSM models: they performed worse than other BTLSM models when KL was small. However, when KL was large, the performance of the BTLSM-ratio model became slightly better. The advantages of the BTLSM-KL models and BTLSM-Bayes-factor models over the BTLSM-

feature-specific-KL models and BTLSM-unadjusted models became more obvious when KL became large than 1, but these differences were still not statistically significant.

Negative transfers and unnecessary transfers can happen when the source and the target distributions are very different. As shown in the Figure 6, BTLSM models perform worse than learned target model when KL becomes large than 3.

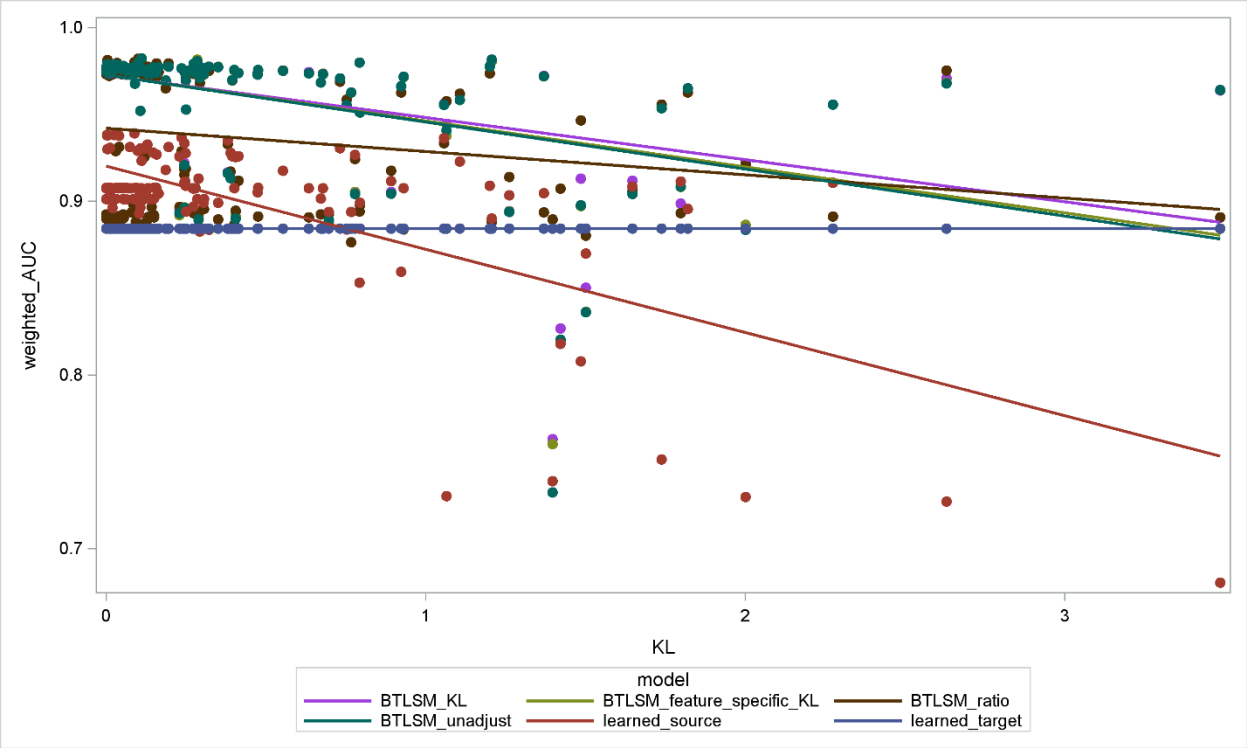

Figure 6 Performance changes at different rates when KL increases (feature selection=CFS, source size=8000, target size=50)

I found similar pattern of performance changes when using information gain (threshold 0.0001) as feature selection approach. In Figure 7, BTLSM models perform worse than learned target model when KL becomes large than 1.

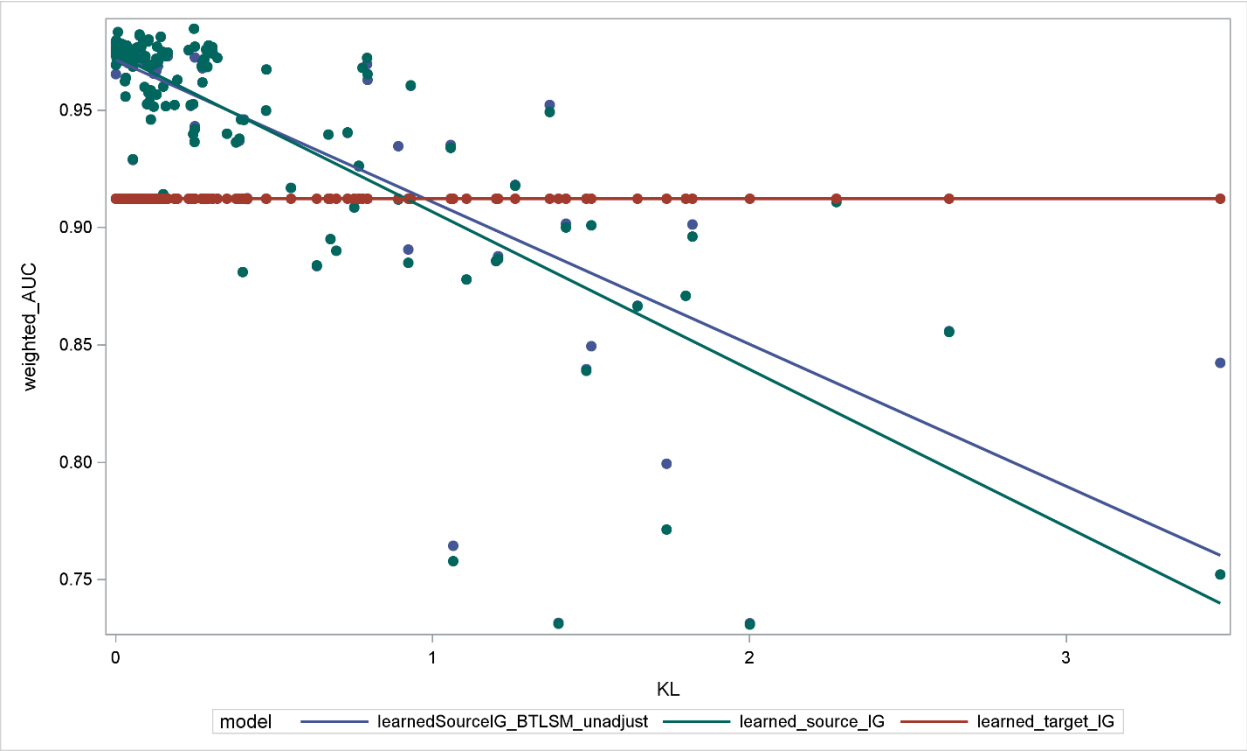

Figure 7 Performance changes at different rates when KL increases (feature selection=information gain with threshold 0.0001, source size=8000, target size=50)

This chapter shows the validity of hypothesis 1 (transfer learning models > target model) and hypothesis 2 (transfer learning models > source model) in most simulation scenarios when the source sample size is large and the target sample size is small, and examines the behaviors of transfer learning models as KL between the target and the source increases. The next chapter will further test the hypotheses on real-world datasets.

5.0 Evaluating the Transfer Learning Algorithms Using Real-World Datasets

In this chapter, I discuss how I used real-world datasets to further test the two hypotheses proposed in Chapter 1. I studied the benefits and limitations of sharing data or a model between two regions where an outbreak of an emergent unknown disease is happening concurrently.

Our previous research (Tsui et al. 2017) demonstrated that a machine-learned CDS could distinguish cases of disease from general ED visits (for biosurveillance purposes), and from symptomatically similar visits when considering dynamic disease prevalence (for clinical differential diagnosis purposes); it also demonstrated that a CDS built in one region was transferable to another region. These studies were conducted for regular infectious diseases (e.g. influenza) that have a lot of historical data available for machine learning CDS.

For an emergent unknown disease, historical data is scarce, i.e., there will be limited data in a region when an outbreak just starts in that region. Data sharing then is critical between regions that are experiencing an outbreak.

Figure 8 shows three scenarios for outbreak timelines in two regions. In scenario 1, the outbreak happens in the two regions concurrently. Region A and region B have the same start date. In scenario 2, the first region has an outbreak before the second region, and the first region's case detection system can be directly transferred to the second region. The second region does not have any cases to share with the first region. This scenario mainly relies on the transferability of a case detection system, so it is not the focus of this dissertation. Scenario 3 is the most common one: two outbreaks happen in two regions, with their epidemic curves partially overlapping. In fact, scenario 3 can be decomposed into the two previous scenarios. Therefore, this dissertation mainly

focused on scenario 1 and scenario 3, where different degrees of overlapping can be achieved by shifting the timeline of an outbreak in one region.

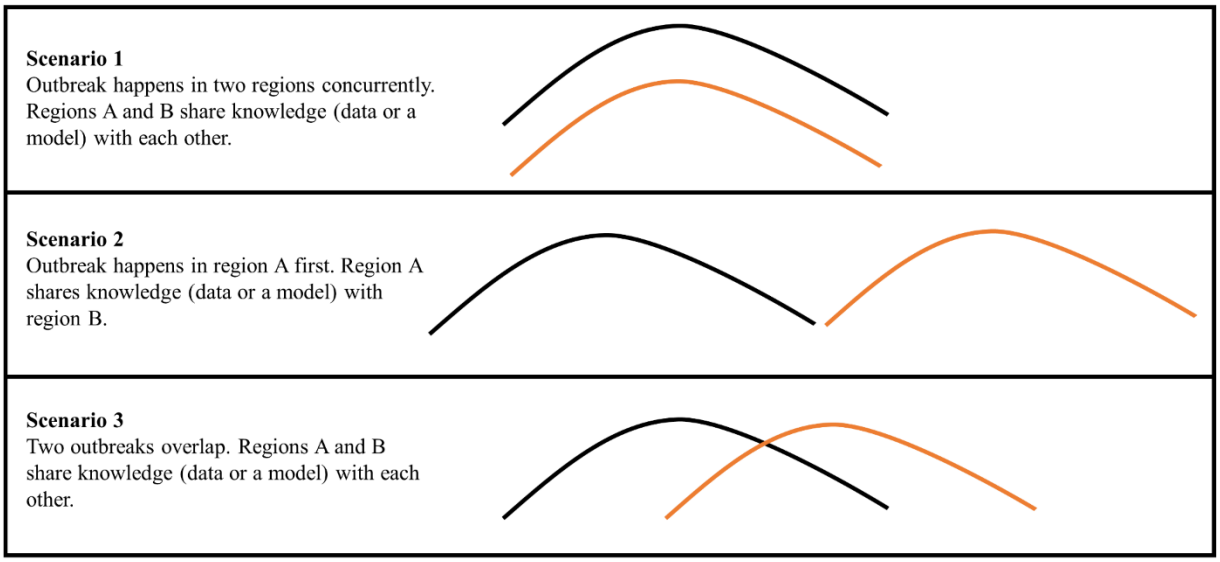

Figure 8 Three types of outbreak timelines in two regions

In the early stages of an outbreak, neither region A nor region B accumulates enough cases to learn a discriminative case detection system. Therefore, I hypothesized that the Bayesian case detection system using cases from the two regions (or source model + target cases) could perform better than a system only leveraging visits from the target region, when this region has a limited number of cases at the beginning of a similar outbreak (hypothesis 1 of the dissertation). I also hypothesized that localization with the target region's few samples could increase the performance of a source case detection system on target region data (hypothesis 2 of the dissertation).

To test these hypotheses, I machine learned five types of Bayesian case detection systems (BCDs): (1) $BN_{d,A}$: a BCD using visits from region A, including both infectious cases and other visits from week 1 to week d; (2) $BN_{d,B}$: a BCD using visits from region B; (3) $BN_{d,B\rightarrow A}$: a BCD focusing on case detection tasks in region A, by tuning $BN_{d,B}$ with visits from region A using a BTLSM algorithm ($BTLSM_{d,B\rightarrow A}$) or by using a BTLSD algorithm to combine region A and region B data for case detection in region A ($BTLSD_{d,B\rightarrow A}$); (4) $BN_{d,A\rightarrow B}$: a BCD focusing on case detection tasks in region B, by tuning $BN_{d,A}$ with visits from region B ($BTLSM_{d,A\rightarrow B}$) or by combining region A and region B data for case detection in region B ($BTLSD_{d,A\rightarrow B}$). (5) $BN_{d,A+B}$: a BCD developed using visits from the two regions (general model) to indicate how much extra information the visits from one region could bring to another region and vice versa. This model also served as a baseline to assess whether the BTLSD approach was better.

Hypothesis 1 is valid when (1) $BN_{d,B\rightarrow A}$ performs better than $BN_{d,A}$ for the scenario of transferring from region B to region A, and (2) $BN_{d,A\rightarrow B}$ performs better than $BN_{d,B}$ for the scenario of transferring from region A to region B. Hypothesis 2 is valid when (1) $BN_{d,A\rightarrow B}$ performs better than $BN_{d,A}$ for region B, and (2) $BN_{d,B\rightarrow A}$ performs better than $BN_{d,B}$ for region A.

5.1 Datasets

In my experiments, I used influenza as a proxy of an emergent unknown diseases, by assuming that both Allegheny County (region A) and Salt Lake County (region B) did not have cases before the 2014-15 season, which was the property of an emergent unknown disease. The research datasets were retrieved from the electronic medical records of emergency department

encounters at University of Pittsburgh Medical Center (UPMC) in Allegheny county and Intermountain healthcare (IH) in Salt Lake County between June 2014 and May 2015 (Table 17).

Table 17 Counts of encounters in IH and UPMC institutions

Disease	IH	UPMC
laboratory confirmed influenza	640	691
laboratory confirmed influenza negative	7,853	4,844
non-influenza (including influenza laboratory negative and visits without influenza tests)	196,914	279,596

Figure 9 plots the seven-day average counts of lab-confirmed cases in both institutions. Table 18 lists the duration of outbreaks. I defined the start day of an outbreak as the first of seven consecutive days with at least seven confirmed cases in those days. I defined the end day of an outbreak as the day when seven consecutive days after that day have fewer than seven confirmed cases. For example, in the IH dataset, the average daily lab-confirmed influenza count for the seven-day (Nov 15, 2014 to Nov 21, 2014) was 1.14, and this was the first time when this measurement was greater than or equal to one in this season (June 1, 2014 to May 31, 2015). Therefore, the start date was Nov 15, 2014. The average daily lab-confirmed influenza count of the seven-day (April 29, 2015 to May 5, 2015) was 1.14, and this was the last time when this measure was greater than or equal to one in this season, so the end date was May 5, 2015.

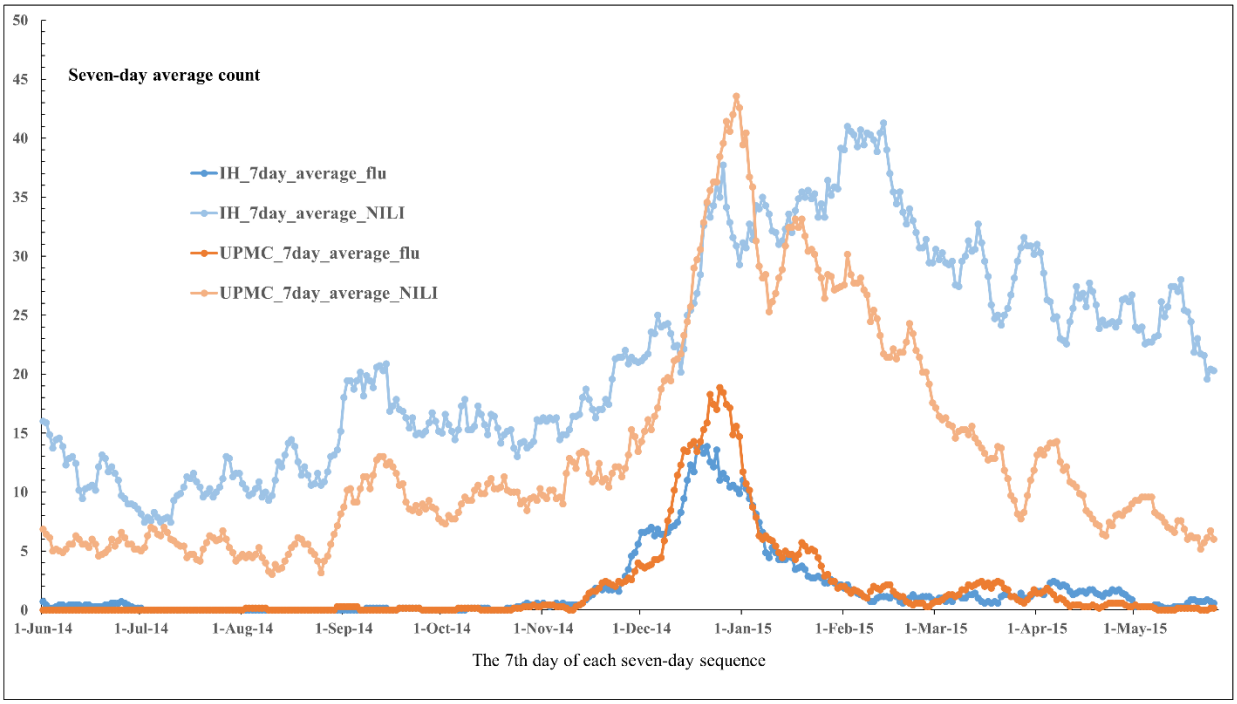

Figure 9 Seven-day average counts of lab-confirmed cases in IH and UPMC

Table 18 Duration of outbreaks in IH and UPMC institutions

Measurement	IH	UPMC
Duration	Nov 15, 2014 to May 5, 2015 (172 days)	Nov 14, 2014 to April 14, 2015 (152 days)
Count of influenza cases (positive influenza lab result)	600	670
Count of non-influenza influenza-like illness encounters (negative influenza lab result)	5,012	3,276
Count of non-influenza encounters (including laboratory negative)	96,124	114,778

Duration: (a) Start: the first day of first seven-day average count ≥1. (b) End: the 7th day of last seven-day average count ≥1.

5.2 Experiment Setting

Infectious disease case detection has two functions (Tsui et al. 2017). Firstly, I hope to identify each case from suspected visits for effective treatment and isolation (experiments in section 5.3). Secondly, I hope to identify cases from general visits to have a good estimation of regional prevalence, outbreak scope, and speed of spread (experiments in section 5.4).

To accomplish two tasks, I studied data/model transfer learning in both directions, from IH to UPMC, and from UPMC to IH, at the first, second, fourth, and eighth week of the epidemic as definition previously. In the 2014-15 influenza season, UPMC and IH started having influenza around Nov 14, 2014. Their weekly counts were also close (Table 19).

Table 19 Cumulative counts of IH and UPMC emergency room visits during 2014-15 influenza season.

Week	Institution	Dataset	Laboratory confirmed influenza	Laboratory confirmed influenza negative	Non-influenza (lab negative + not tested)	Date Duration
Week 1	UPMC	Train	7	93	5,079	2014-11-14 to 2014-11-20
	IH	Train	6	114	3,020	2014-11-15 to 2014-11-20
Weeks 1-2	UPMC	Train	23	166	10,115	2014-11-14 to 2014-11-27
	IH	Train	18	236	6,565	2014-11-15 to 2014-11-27
Weeks 1-4	UPMC	Train	71	388	20,970	2014-11-14 to 2014-12-11
	IH	Train	94	550	14,395	2014-11-15 to 2014-12-11
Weeks 1-8	UPMC	Train	455	1,311	42,489	2014-11-14 to 2015-01-08
	IH	Train	396	1,385	30,613	2014-11-15 to 2015-01-08
Weeks 8-	UPMC	Test	215	1,965	72,289	2015-01-09 to 2015-04-14
	IH	Test	204	3,627	65,511	2015-01-09 to 2015-05-05

With each pair of UPMC and IH datasets (each row in table 19), I built nine models across three situations: 1) two local models in the non-transferring situations, 2) two BTLSM models where only the source model was shared, 3) four BTLSD models and one general model where source data was shared. Among these, the two local models served as baseline models for hypothesis testing. The general model was used to indicate how much extra information the visits from one region could bring to another region.

The two local models, BN_{IH} and BN_{UPMC}, represent the best case detection each institution can do using only data from its own institution. To create these models, I firstly used feature selection to remove features whose information gain scores were less than 0.0001 on the training data and sorted all remaining features from the highest information gain score to the lowest. Then, I used the K2 learning algorithm to build a Bayesian network model. These two local models were later tested on the testing datasets for both IH and UPMC, showing local performance as well as their inherent transferability.

In situations where the model from one institution could be shared to another, I used the BTLSM algorithm for model learning (Figure 10). For the IH to UPMC transferring scenario ($BTLSM_{IH \rightarrow UPMC}$), after receiving the BN_{IH} (source model), the algorithm firstly removed the IH specific features (not appearing in UPMC data or information gain less than 0.0001 in UPMC data), and then conducted pruning and growing steps to modify the model structure. After that, the algorithm used pseudo counts estimated from the IH model and the actual counts in UPMC training data to calculate the conditional probabilities for each feature given its parents. Since the simulation experiment showed the BTLSM algorithm using the original IH training data size same as the equivalent sample size performs as well as other weighting strategies, I used the unadjusted

approach in the following experiments. The UPMC to IH transferring scenario ($BTLSM_{UPMC \rightarrow IH}$) used the same model building procedure.

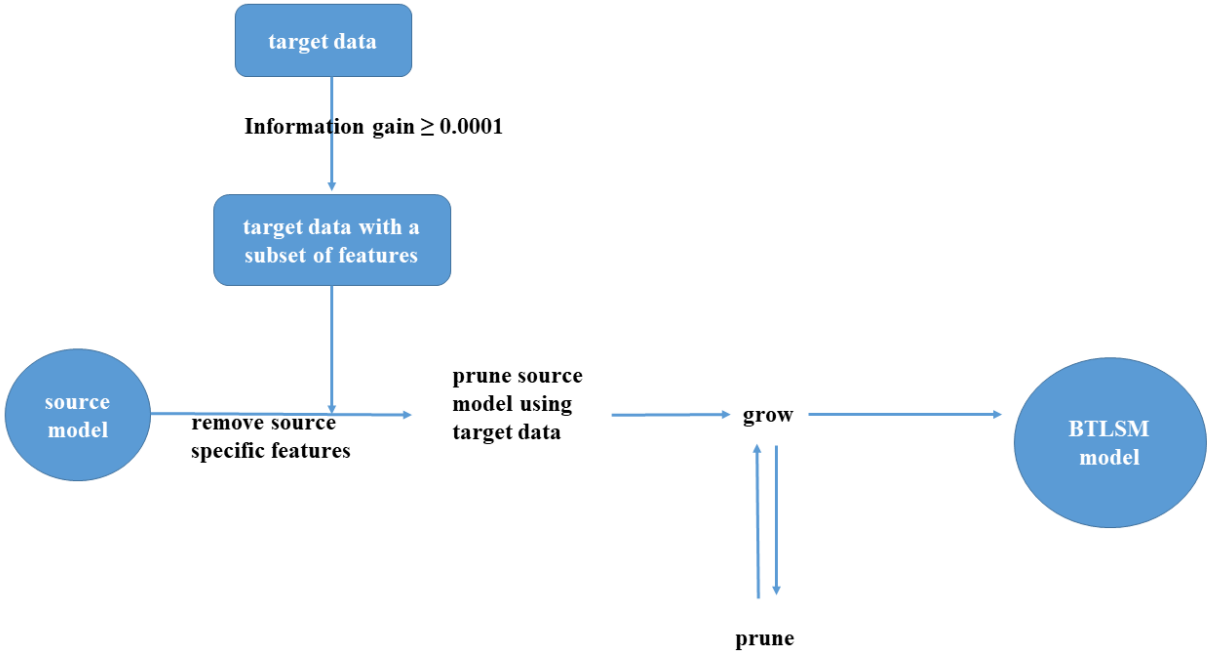

Figure 10 BTLSM approach used in the real-world influenza experiment

When data from one institution can be shared to another, I have three options: (1) combining the two datasets into one training dataset and using information gain and K2 to develop a general model, $BN_{IH+UPMC}$, (2) using a BTLSD algorithm with naïve Bayesian structure (BTLSD-NB), and (3) using the BTLSD algorithm with a K2 searching strategy (BTLSD-K2) (Figure 11). The difference between the BTLSD algorithm and the general model approach is that the BTLSD algorithm conducts feature selection for each training dataset and can remove source-specific features and add target-specific features when developing the final model for the target task. By

comparing the general model with BTLSD models, I may assess whether the feature selection procedure in the BTLSD algorithm is better than conventional feature selection on mixed target and source data in cases where source data is sharable.

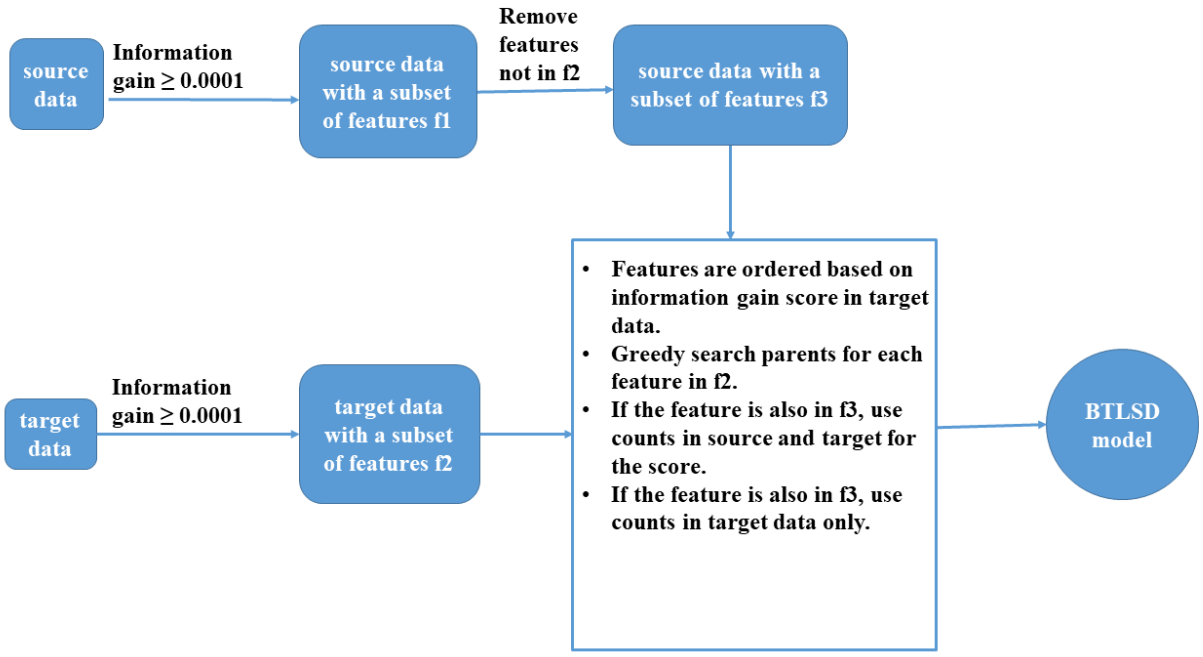

Figure 11 BTLSD approach used in the real-world influenza experiment

5.3 Results: Influenza Detection among Suspected Visits

The first set of experiments aims to detect influenza cases among all visits that have influenza laboratory tests, which are considered to be suspected visits. All of the training visits involved testing with influenza laboratory tests, from which the visits were labelled influenza, or

influenza negative. I used one NLP parser to extract 81 clinical findings from the notes for these visits. Because this case detection task is to differentiate laboratory-confirmed influenza cases from other suspected visits, all of the case detection models were built without the 14 NLP-extracted clinical findings that mentioned laboratory orders or results, including lab order (nasal swab), lab testing ordered (rsv), lab testing ordered (influenza w/other respiratory pathogens panel), lab positive coronavirus, lab positive rsv, lab positive strep a, lab positive rhinovirus, lab positive parainfluenza, lab positive adenovirus, lab positive enterovirus, lab testing ordered (influenza), lab positive influenza, lab positive hmpv, and lab positive vrp - pathogen not specified.

At an early stage of an outbreak, each local hospital usually identifies very few cases among a small number of suspected visits. For example, during the first week of the 2014-15 influenza season (from Nov 14, 2014 to Nov 20, 2014), UPMC had seven laboratory confirmed influenza visits and 93 laboratory confirmed non-influenza visits, while IH had six and 114, respectively.

I conducted two experiments, one addressing transfer learning from UPMC to IH, and another addressing transfer learning from IH to UPMC.

In the UPMC to IH experiment (Table 20), transfer learning showed benefits on the first two weeks. At week 2, trained with 254 visits, the local IH model, BN_{IH}, reached an AUC of 0.6727. The UPMC model (trained with 189 visits from UPMC), BN_{UPMC}, reached an AUC of 0.6077 on IH visits. This UPMC model was then shared to IH, the BTLSM algorithm used the 254 IH training visits to adjust the UPMC model and obtained $BTLSM_{UPMC \rightarrow IH}$, for which the AUC reached 0.7118. This AUC was statistically significantly better than both the local IH model and the transferred UPMC model. If sharing data at week 2, the performance of BTLSD-$NB_{UPMC \rightarrow IH}$ model could reach 0.7198 and the performance of the general model, $BN_{IH+UPMC}$, could reach

0.7165. Both were significantly better than the local IH model and the transferred UPMC model, but they were not statistically significantly different from the BTLSM model.

After week 2, sharing data or a model from UPMC to IH became unnecessary. Neither the BTLSD model nor the BTLSM model performed significantly better than the local IH model, BN_{IH}. On the other hand, transfer learning techniques were able to significantly boost the performance of the transferred UPMC model. In chapter 6, I further explain the conditions where transfer learning works and where it does not.

Table 20 Classification performance when transferring knowledge from UPMC to IH for influenza detection among suspected visits

Week	Target: BN_{IH}	Source: BN_{UPMC}	$BTLSM_{UPMC \rightarrow IH}$	$BN_{IH+UPMC}$	BTLSD-$NB_{UPMC \rightarrow IH}$	BTLSD-$K2_{UPMC \rightarrow IH}$
Week 1	0.6419 (0.6035-0.6803)[2456]	0.5834 (0.5432-0.6236)[1345]	0.6669 (0.6297-0.7040)[26]	0.6689 (0.6323-0.7056)[126]	0.668 (0.6312-0.7047)[126]	0.5989 (0.5551-0.6426)[1345]
Weeks 1-2	0.6727 (0.6335-0.7119)[2345]	0.6077 (0.5690-0.6465)[13456]	0.7118 (0.6770-0.7466)[12]	0.7165 (0.6825-0.7505)[12]	0.7198 (0.6859-0.7537)[12]	0.6918 (0.6523-0.7313)[2]
Weeks 1-4	0.7402 (0.7059-0.7745)[2]	0.6596 (0.6219-0.6972)[13456]	0.7291 (0.6936-0.7647)[245]	0.7468 (0.7141-0.7795)[23]	0.7463 (0.7133-0.7793)[23]	0.7213 (0.6833-0.7593)[2]
Weeks 1-8	0.7625 (0.73-0.795)[236]	0.7043 (0.6701-0.7385)[1456]	0.7131 (0.6752-0.7509)[1456]	0.7512 (0.7199-0.7826)[236]	0.7528 (0.7215-0.7841)[236]	0.658 (0.6155-0.7005)[12345]

Measurement: AUC for influenza positive and negative classification on 204 influenza positive and 3,627 influenza negative IH visits between Jan 9 and May 5, 2015. Yellow: significantly better than the target model.
1. significantly different from BN_{IH} ($p<0.05$).
2. significantly different from BN_{UPMC} ($p<0.05$).
3. significantly different from $BTLSM_{UPMC \rightarrow IH}$ ($p<0.05$).
4. significantly different from $BN_{IH+UPMC}$ ($p<0.05$).
5. significantly different from BTLSD-$NB_{UPMC \rightarrow IH}$ ($p<0.05$).
6. significantly different from BTLSD-$K2_{UPMC \rightarrow IH}$ ($p<0.05$).

Regarding the calibration error, the BTLSD-K2 models showed the lowest error at week 2 and 4, and the general model showed the lowest error at week 8 (Table 21). The BTLSM models' error was slightly higher than the learned target model, BN_{IH}, and lower than the learned source model, BN_{UPMC}.

Table 21 Calibration performance when transferring knowledge from UPMC to IH for influenza detection among suspected visits

Week	Target: BN_{IH}	Source: BN_{UPMC}	$BTLSM_{UPMC \rightarrow IH}$	$BN_{IH+UPMC}$	BTLSD-$NB_{UPMC \rightarrow IH}$	BTLSD-$K2_{UPMC \rightarrow IH}$
Week 1	0.0464	0.3843	0.0901	0.0839	0.0777	0.0569
Weeks 1-2	0.0616	0.502	0.0684	0.0924	0.0879	0.0534
Weeks 1-4	0.0911	0.4344	0.0974	0.1179	0.1201	0.0888
Weeks 1-8	0.1298	0.1284	0.1526	0.1084	0.1114	0.1523

Measurement: expected calibration error for influenza positive and negative classification on 204 influenza positive and 3,627 influenza negative IH visits between Jan 9 and May 5, 2015. The lowest calibration error in each row is marked yellow.

Similarly, in the IH to UPMC experiment (Table 22), transfer learning showed benefits on the second week when sharing model/data. At week 2, the local target model performed significantly worse than the $BTLSM_{IH \rightarrow UPMC}$ (0.5906 vs. 0.6231, p=0.0018), the general model, $BN_{IH+UPMC}$ (0.6152, p=0.0097), and the BTLSD-$NB_{IH \rightarrow UPMC}$ (0.6120, p=0.0225). At week 4, the local target model still performed significantly worse than the $BTLSM_{IH \rightarrow UPMC}$ (0.6035 vs. 0.6434, p=0.0002), the general model, BNIH+UPMC (0.6318, p=0.0003), and the BTLSD-$NB_{IH \rightarrow UPMC}$

(0.6313, p=0.0003). At week 1 and week 8, the target model was not statistically significantly different from the BTLSM model, general model, or BTLSD models.

In addition, the transfer learning models performed significantly better than the source (IH) model at week 1 except for BTLSD-K2$_{IH\rightarrow UPMC}$. After that, the transfer learning models did not significantly boost the performance of the source model except for weeks 1-8 of BN$_{IH+UPMC}$.

Table 22 Classification performance when transferring knowledge from IH to UPMC for influenza detection among suspected visits

Week	Target: BN$_{UPMC}$	Source: BN$_{IH}$	BTLSM$_{IH\rightarrow UPMC}$	BN$_{IH+UPMC}$	BTLSD-NB$_{IH\rightarrow UPMC}$	BTLSD-K2$_{IH\rightarrow UPMC}$
Week 1	0.5968 (0.5548-0.6389)[2]	0.5625 (0.5213-0.6036)[1345]	0.5962 (0.5544-0.638)[245]	0.5898 (0.5481-0.6314)[23]	0.5906 (0.5489-0.6323)[23]	0.5866 (0.5447-0.6284)
Weeks 1-2	0.5906 (0.5461-0.6352)[345]	0.5969 (0.5559-0.6380)	0.6231 (0.5823-0.6638)[145]	0.6152 (0.5737-0.6567)[135]	0.6120 (0.5704-0.6537)[134]	0.5927 (0.5514-0.6339)
Weeks 1-4	0.6035 (0.5583-0.6486)[2345]	0.6591 (0.6211-0.6971)[1456]	0.6434 (0.6037-0.6831)[16]	0.6318 (0.5899-0.6737)[126]	0.6313 (0.5894-0.6732)[12]	0.6082 (0.5655-0.6508)[234]
Weeks 1-8	0.6724 (0.6355-0.7094)	0.6626 (0.6245-0.7008)[4]	0.6628 (0.6251-0.7005)	0.6846 (0.6474-0.7219)[2]	0.6801 (0.6429-0.7174)	0.6744 (0.6363-0.7124)

Measurement: AUC for influenza positive and negative classification on 215 influenza positive and 1,965 influenza negative UPMC visits between Jan 9 and April 14, 2015. Yellow: significantly better than the target model.
1. significantly different from BN$_{UPMC}$ ($p<0.05$).
2. significantly different from BN$_{IH}$ ($p<0.05$).
3. significantly different from BTLSM$_{IH\rightarrow UPMC}$ ($p<0.05$).
4. significantly different from BN$_{IH+UPMC}$ ($p<0.05$).
5. significantly different from BTLSD-NB$_{IH\rightarrow UPMC}$ ($p<0.05$).
6. significantly different from BTLSD-K2$_{IH\rightarrow UPMC}$ ($p<0.05$).

Regarding the calibration error, the BTLSD-K2 models always had lowest error (Table 23). The error for the BTLSM models and BTLSD-NB models was between the error of the learned source model and the learned target model.

Table 23 Calibration performance when transferring knowledge from IH to UPMC for influenza detection among suspected visits

Week	Target: BN_{UPMC}	Source: BN_{IH}	$BTLSM_{IH \rightarrow UPMC}$	$BN_{IH+UPMC}$	BTLSD-$NB_{IH \rightarrow UPMC}$	BTLSD-$K2_{IH \rightarrow UPMC}$
Week 1	0.2155	0.0946	0.1421	0.1306	0.1401	0.0890
Weeks 1-2	0.2954	0.1254	0.1553	0.1749	0.1772	0.076
Weeks 1-4	0.2794	0.1529	0.1565	0.1913	0.1904	0.0867
Weeks 1-8	0.1937	0.1514	0.1756	0.1847	0.1825	0.1243

Measurement: expected calibration error for influenza positive and negative classification on 204 influenza positive and 3,627 influenza negative IH visits between Jan 9 and May 5, 2015. The lowest calibration error in each row is marked yellow.

Tables 20 and 22 do not show any advantage of BTLSD algorithms over the general model. This may be because the source data and target data shared the same feature space (using one NLP parser to extract clinical findings from both institutions). Compared to the general model approach, the BTLSD's main advantage is its ability to add target-specific features. Unfortunately, in these real-world experiments, there were no target-specific features to be added.

In addition, the BTLSD-NB algorithm was found to perform slightly better than the BTLSD-K2 algorithm. That may be because I did not restrict the number of maximum parents for the K2 algorithm here, so I may end up with very complicated models when the feature size was large. So, I excluded BTLSD-K2 in the following analysis.

Since transfer learning showed benefits at week 2 in both scenarios (from IH to UPMC and, from UPMC to IH), I further compared the predictive features of the developed models in order to further understand the advantages of BTLSM and BTLSD models (Table 24). The union of features had 64 features in total. The local BN_{IH} included 62 features and did not include *chills* and *decreased activity*. The local BN_{UPMC} also included 62 features and did not include

parainfluenza and *viral pneumonia*. The general model included all features that were influential for at least one individual institution. The general model did show benefits on feature selection, and its better performance on both institutions may have resulted from relatively larger training samples from using data from both institutions compared to only leveraging data from one institution.

Table 24 Features of models to detect influenza among suspected visits using first two weeks data

64 Features	**BN_{IH} (62)**	**BN_{UPMC} (62)**	**$BN_{IH+UPMC}$ (64)**	**BTLSM $_{IH→UPMC}$ (58)**	**BTLSM $_{UPMC→IH}$ (48)**	**BTLSD-NB $_{IH→UPMC}$ (62)**	**BTLSD-NB $_{UPMC→IH}$ (62)**
influenza-like illness							
other cough				*			
bilateral acute conjunctivitis					*		
vomiting					*		
stuffy nose							
conjunctivitis							
age group							
stridor							
abnormal chest radiograph findings							
highest measured temperature							
pharyngitis on exam							
hemoptysis							
runny nose							
anorexia							
productive cough				*			
wheezing							
hoarseness							
cervical lymphadenopathy					*		
malaise							
barking cough							
nausea							
tachypnea							
viral syndrome					*		
seizure							
acute onset							
cyanosis					*		
poor feeding							

Table 24 Features of models to detect influenza among suspected visits using first two weeks data (continued)

reported fever							
nonproductive cough							
rigor							
sore throat					*		
rhonchi							
parainfluenza		*		*	*	*	
pharyngitis diagnosis							
respiratory distress							
viral pneumonia		*		*	*	*	
poor response of fever to antipyretics							
arthralgia					*		
abdominal tenderness					*		
grunting							
chest pain					*		
uri					*		
rales							
other pneumonia							
toxic appearance							
infiltrate							
headache					*		
croup							
diarrhea							
myalgia					*		
dyspnea							
nasal flaring							
ill-appearing							
weakness or fatigue							
bronchitis							
abdominal pain							
apnea							
bronchiolitis							
other abnormal breath sounds							
hypoxemia (sp02 on room air < 90%)							
chest wall retractions							
crackles							
chills	*			*	*		*
decreased activity	*			*	*		*

Star indicates absence of the feature

When transferring BN_{IH} to UPMC, the BTLSM algorithm removed *chills* and *decreased activity* because they were not influential for the IH data, and it also removed *other cough* and *productive cough* because the change led to a lower score calculated using information from the BN_{IH} and UPMC data. The BTLSM models ended up with a smaller feature set than the other models, so their better performance may result from a more relevant feature set as well as a better estimation of correlations between features and the class for the target task.

BTLSD-$NB_{IH \rightarrow UPMC}$ had the same feature set as the BN_{UPMC} model and BTLSD-$NB_{UPMC \rightarrow IH}$ had the same feature set as the BN_{IH} model, so their better performance over the local model may result from a better estimation of conditional probability tables using training data from two institutions.

5.4 Results: Influenza Detection among General Emergency Room Visits

The second set of experiments aims to detect influenza cases among all emergency visits in order to support a better estimation of population prevalence, outbreak scope, and outbreak timeline. In these experiments, all training visits have been labelled as influenza or other non-influenza visits (including influenza laboratory results negative visits and visits without laboratory tests).

I conducted two experiments: one analyzed transferring data or a model from IH to UPMC, and the other analyzed transferring data or a model from UPMC to IH. I studied these two situations using the real timelines during 2014-15. In addition, I also shifted the source region's start date and assumed the source region to start one, two, or three weeks earlier than the target

region. This shifting was aimed at assessing the different degrees of overlapping in scenario 3 introduced at the beginning of this chapter.

In the IH to UPMC experiment (Table 25), transfer learning showed benefits over the target model at the first week. The target model BN_{UPMC} (0.7294) performed statistically worse than the transferred source model BN_{IH} (0.7882, $p<0.0001$), the general model $BN_{IH+UPMC}$ (0.7763, $p<0.0001$), and the BTLSD-$NB_{IH\rightarrow UPMC}$ (0.7659, $p<0.0001$). If the IH flu season started one week earlier than UPMC, then for the first week at UPMC, sharing the IH data/model to UPMC shows even more benefits (BN_{IH}: 0.8112, $BTLSM_{IH\rightarrow UPMC}$: 0.8023, $BN_{IH+UPMC}$: 0.8054, BTLSD-$NB_{IH\rightarrow UPMC}$: 0.7791). In none of these scenarios, did combining the IH data and UPMC data show better performance than the source model, suggesting that the first week of UPMC data did not add extra information to the IH data for UPMC model development.

After the first week at UPMC, transfer learning became unnecessary. For example, at week 2, BN_{UPMC} reached an AUC of 0.8356, and it was developed with 10,138 visits (23 influenza cases). Adding 6583 visits (18 influenza cases) from IH for model development, the general model $BN_{UPMC+IH}$ reached an AUC of 0.8372, which was not statistically better than the BN_{UPMC}. The data for the first two-week for UPMC was already enough to develop a well-performed UPMC model and the IH data did not add extra value. After the first week, the BTLSM models and BTLSD models also did not show significant benefits in the IH to UPMC transferring situation.

Table 25 Classification performance when transferring knowledge from IH to UPMC for influenza detection among emergency department visits

Scenario	IH training dataset	UPMC training dataset	Target: BN_{UPMC}	Source: BN_{IH}	$BTLSM_{IH \rightarrow UPMC}$	$BN_{IH+UPMC}$	BTLSD-$NB_{IH \rightarrow UPMC}$
UPMC and IH have influenza concurrently	Week 1	Week 1	0.7294 (0.6913-0.7675)[245]	0.7882 (0.7625-0.8139)[1345]	0.7406 (0.7085-0.7727)[245]	0.7763 (0.7478-0.8047)[1235]	0.7659 (0.7348-0.797)[1234]
	Weeks 1-2	Weeks 1-2	0.8356 (0.8146-0.8566)[23]	0.8112 (0.7879-0.8345)[1345]	0.7843 (0.7532-0.8154)[1245]	0.8372 (0.8166-0.8578)[235]	0.8343 (0.813-0.8556)[234]
	Weeks 1-4	Weeks 1-4	0.8613 (0.8426-0.88)[2345]	0.8069 (0.7821-0.8318)[1345]	0.7742 (0.7441-0.8043)[1245]	0.8423 (0.8224-0.8622)[1235]	0.8440 (0.8242-0.8638)[1234]
	Weeks 1-8	Weeks 1-8	0.8581 (0.8367-0.8795)[23]	0.8178 (0.7955-0.8401)[145]	0.7961 (0.7615-0.8307)[145]	0.8638 (0.8454-0.8821)[23]	0.8623 (0.8435-0.8811)[23]
IH starts one week earlier	Weeks 1-2	Week 1	0.7294 (0.6913-0.7675)[2345]	0.8112 (0.7879-0.8345)[145]	0.8023 (0.7744-0.8302)[1]	0.8054 (0.7809-0.83)[125]	0.7791 (0.7494-0.8087)[124]
	Weeks 1-3	Weeks 1-2	0.8356 (0.8146-0.8566)[2345]	0.7942 (0.7672-0.8212)[1345]	0.7186 (0.6787-0.7549)[1245]	0.8094 (0.785-0.8338)[1235]	0.8205 (0.7977-0.8434)[1234]
IH starts two weeks earlier	Weeks 1-3	Week 1	0.7294 (0.6913-0.7675)[2345]	0.7942 (0.7672-0.8212)[1345]	0.7685 (0.7367-0.8003)[12]	0.7713 (0.7407-0.8018)[12]	0.7726 (0.7419-0.8033)[12]
	Weeks 1-4	Weeks 1-2	0.8356 (0.8146-0.8566)[2345]	0.8069 (0.7821-0.8318)[1345]	0.7181 (0.6809-0.7552)[1245]	0.8033 (0.7782-0.8284)[1235]	0.8128 (0.789-0.8365)[1234]
IH starts three weeks earlier	Weeks 1-4	Week 1	0.7294 (0.6913-0.7675)[2345]	0.8069 (0.7821-0.8318)[1345]	0.7864 (0.7585-0.8142)[12]	0.7842 (0.756-0.8124)[125]	0.7794 (0.7503-0.8085)[124]
	Weeks 1-5	Weeks 1-2	0.8356 (0.8146-0.8566)[2345]	0.8094 (0.7852-0.8335)[134]	0.7086 (0.6712-0.746)[1245]	0.7997 (0.7744-0.825)[1235]	0.8078 (0.7836-0.8321)[134]

Measurement: AUC for influenza positive and negative classification on 215 influenza positive and 72,289 influenza negative UPMC visits between Jan 9 and April 14, 2015. Yellow: significantly better than the local target model.
1. significantly different from target model BN_{UPMC} ($p<0.05$).
2. significantly different from source model BN_{IH} ($p<0.05$).
3. significantly different from $BTLSM_{IH \rightarrow UPMC}$ ($p<0.05$).
4. significantly different from $BN_{IH+UPMC}$ ($p<0.05$).
5. significantly different from BTLSD-$NB_{IH \rightarrow UPMC}$ ($p<0.05$).

Regarding the calibration error, to my surprise, the BTLSM models had the lowest error in all situations when transferring from IH to UPMC for influenza detection from general visits (Table 26).

Table 26 Calibration performance when transferring knowledge from IH to UPMC for influenza detection among emergency department visits

Scenario	IH training dataset	UPMC training dataset	Target: BN_{UPMC}	Source: BN_{IH}	$BTLSM_{IH \rightarrow UPMC}$	$BN_{IH+UPMC}$	$BTLSD\text{-}NB_{IH \rightarrow UPMC}$
UPMC and IH have influenza concurrently	Week 1	Week 1	0.0285	0.0247	0.0057	0.0348	0.0294
	Weeks 1-2	Weeks 1-2	0.0514	0.0452	0.0055	0.0514	0.0499
	Weeks 1-4	Weeks 1-4	0.0548	0.0556	0.0035	0.057	0.0588
	Weeks 1-8	Weeks 1-8	0.068	0.0533	0.0061	0.0575	0.0582
IH starts one week earlier	Weeks 1-2	Weeks 1	0.0285	0.0452	0.0055	0.042	0.0372
	Weeks 1-3	Weeks 1-2	0.0514	0.0507	0.0046	0.053	0.0526
IH starts two weeks earlier	Weeks 1-3	Week 1	0.0285	0.0507	0.0048	0.0472	0.0421
	Weeks 1-4	Weeks 1-2	0.0514	0.0556	0.0047	0.0537	0.0545
IH starts three weeks earlier	Weeks 1-4	Week 1	0.0285	0.0556	0.0058	0.0502	0.048
	Weeks 1-5	Weeks 1-2	0.0514	0.0504	0.0046	0.05	0.0532

Measurement: expected calibration error for influenza positive and negative classification on 215 influenza positive and 72,289 influenza negative UPMC visits between Jan 9 and April 14, 2015. The lowest calibration error in each row is marked yellow.

In the UPMC to IH experiment (Table 27), the IH data was also enough for its local case detection task. For example, with only the data from the first week of training data (6 influenza, 3020 non-influenza visits), the developed BN_{IH} already reached an AUC of 0.8765. Although the developed BN_{UPMC} reached about 0.88 on IH data, it may be unnecessary to share the data/model because the local IH model/data was sufficient.

The second set of experiments only showed positive transfer in the first week for the IH to UPMC situation and showed many unnecessary transfers for the UPMC to IH situation. Chapter 6 further compares these scenarios and investigates the reasons for positive transfer and unnecessary transfer.

Table 27 Classification performance when transferring knowledge from UPMC to IH for influenza detection among emergency department visits

Scenario	UPMC training dataset	IH training dataset	target: BN_{IH}	source: BN_{UPMC}	$BTLSM_{UPMC \rightarrow IH}$	$BN_{IH+UPMC}$	BTLSD-$NB_{UPMC \rightarrow IH}$
UPMC and IH have influenza concurrently	Week 1	Week 1	0.8765 (0.8625-0.8904)[3]	0.8824 (0.8654-0.8994)[3]	0.8299 (0.8044-0.8554)[1245]	0.8795 (0.8646-0.8944)[3]	0.8776 (0.8632-0.892)[3]
	Week 1-2	Week 1-2	0.8856 (0.8714-0.8998)[3]	0.8823 (0.8637-0.9009)[3]	0.8164 (0.784-0.8487)[1245]	0.8879 (0.8723-0.9036)[35]	0.8828 (0.8671-0.8986)[34]
	Week 1-4	Week 1-4	0.8879 (0.8739-0.9019)[23]	0.8643 (0.8415-0.8872)[145]	0.8404 (0.8121-0.8687)[145]	0.8891 (0.8733-0.905)[235]	0.8871 (0.8717-0.9025)[234]
	Week 1-8	Week 1-8	0.8824 (0.8673-0.8976)[23]	0.8260 (0.7985-0.8534)[145]	0.7893 (0.761-0.8175)[145]	0.8821 (0.8648-0.8994)[23]	0.8807 (0.8636-0.8977)[23]
UPMC starts one week earlier	Week 1-2	Week 1	0.8765 (0.8625-0.8904)[34]	0.8823 (0.8637-0.9009)[34]	0.8200 (0.7902-0.8498)[1245]	0.8868 (0.8707-0.903)[1235]	0.8767 (0.8614-0.892)[34]
	Week 1-3	Week 1-2	0.8856 (0.8714-0.8998)[35]	0.8728 (0.8521-0.8936)[34]	0.8313 (0.7999-0.8626)[1245]	0.8824 (0.8648-0.8999)[235]	0.8774 (0.8602-0.8946)[134]
UPMC starts two weeks earlier	Week 1-3	Week 1	0.8765 (0.8625-0.8904)[3]	0.8728 (0.8521-0.8936)[3]	0.8445 (0.8183-0.8708)[1245]	0.8784 (0.86-0.8968)[3]	0.8769 (0.8605-0.8933)[3]
	Week 1-4	Week 1-2	0.8856 (0.8714-0.8998)[235]	0.8643 (0.8415-0.8872)[14]	0.8485 (0.8198-0.8772)[14]	0.8753 (0.8552-0.8954)[23]	0.8687 (0.8494-0.888)[1]
UPMC starts three weeks earlier	Week 1-4	Week 1	0.8765 (0.8625-0.8904)[3]	0.8643 (0.8415-0.8872)$_{34}$	0.8141 (0.7818-0.8464)[1245]	0.8708 (0.8497-0.8918)[23]	0.8718 (0.8537-0.8899)[3]
	Week 1-5	Week 1-2	0.8856 (0.8714-0.8998)[2345]	0.8585 (0.8348-0.8822)[13]	0.8165 (0.783-0.85)[1245]	0.8531 (0.8286-0.8776)[13]	0.8591 (0.8374-0.8808)[13]

Measurement: AUC for influenza positive and negative classification on 204 influenza positive and 65,511 other IH visits between Jan 9 and May 5, 2015. Yellow: statistically significantly better than the local target model.

1. significantly different from target model BN_{IH} ($p<0.05$).
2. significantly different from source model BN_{UPMC} ($p<0.05$).
3. significantly different from $BTLSM_{UPMC \rightarrow IH}$ ($p<0.05$).
4. significantly different from $BN_{IH+UPMC}$ ($p<0.05$).
5. significantly different from BTLSD-$NB_{UPMC \rightarrow IH}$ ($p<0.05$).

Regarding the calibration error, the BTLSM models also had the lowest error in all situations when transferring from UPMC to IH (Table 28).

Table 28 Calibration performance when transferring knowledge from UPMC to IH for influenza detection among emergency department visits

Scenario	UPMC training dataset	IH training dataset	Target: BN_{IH}	Source: BN_{UPMC}	$BTLSM_{UPMC \rightarrow IH}$	$BN_{IH+UPMC}$	$BTLSD\text{-}NB_{UPMC \rightarrow IH}$
UPMC and IH have influenza concurrently	Week 1	Week 1	0.0507	0.0633	0.0111	0.0799	0.0678
	Week 1-2	Week 1-2	0.0677	0.1024	0.0045	0.085	0.0826
	Week 1-4	Week 1-4	0.0952	0.0804	0.0075	0.1094	0.1042
	Week 1-8	Week 1-8	0.1015	0.0689	0.0112	0.0956	0.0963
UPMC starts one week earlier	Week 1-2	Week 1	0.0507	0.1024	0.0099	0.0955	0.081
	Week 1-3	Week 1-2	0.0677	0.0912	0.0043	0.0869	0.0837
UPMC starts two weeks earlier	Week 1-3	Week 1	0.0507	0.0912	0.0099	0.0932	0.0727
	Week 1-4	Week 1-2	0.0677	0.0804	0.0068	0.0791	0.0768
UPMC starts three weeks earlier	Week 1-4	Week 1	0.0507	0.0804	0.0053	0.0811	0.0621
	Week 1-5	Week 1-2	0.0677	0.0719	0.0049	0.0653	0.0653

Measurement: expected calibration error for influenza positive and negative classification on 204 influenza positive and 65,511 other IH visits between Jan 9 and May 5, 2015. The lowest calibration error in each row is marked yellow.

6.0 Discussion and Conclusions

In the first two sections of this chapter, I summarize the results of the simulated experiments and real-world experiments for the two hypotheses: (1) a transfer learning model that is learned using source knowledge and target data performs classification in the target context better than a target model that is learned solely from target data; (2) a transfer learning model that is learned using source knowledge and target data performs classification in the target context better than a source model. Then, I discuss the experimental findings for the three research questions: (1) how does similarity between the source and the target relate to transfer (positive, unnecessary, or negative)? (2) How does the target data sample size affect the effectiveness of transfer learning? (3) What should be shared, source model or source data, and when?

6.1 Whether and When Transfer Learning Is Beneficial for the Target

The primary goal of transfer learning is to reach a higher case detection capability for one region/institution by borrowing knowledge/data from another region/institution. Therefore, the first research hypothesis is that transfer learning models perform better for the target task than local target models. This hypothesis could be valid when two conditions are met: (1) the two domains have similar distributions; (2) the target has insufficient data.

In the influenza simulation experiments, the condition of similar distributions was not met in scenario 4, where the true source model and the true target model were very different in their classification ability in the target data (true source model: 0.7086, true target model: 0.9742) and

the feature space (only 43% of target features appeared in the source). The condition of similar distributions was met in the other five scenarios. The true source model and the true target model were similar in their classification ability in the target data (scenario 1: true source mode 0.9225, true target model 0.969; scenario 2: 0.901, 0.969; scenario 3: 0.9459, 0.9643; scenario 5: 0.9455, 0.9742; scenario 6: 0.7297, 0.8053). At least half of the target features were available in the source data. In these five scenarios, the first hypothesis was supported. When the source had 8000 visits and the target had 50 visits, the BTLSM models (weighting strategy: unadjusted, KL, or feature-specific-KL) and BTLSD-KL models all performed statistically significantly better than the learned target models in the five scenarios. To my surprise, when the target size was increased to 8000 visits, transfer learning still showed benefits in scenarios 1,2, and 6, where all BTLSD models (unadjusted, KL, feature-specific-KL, or ratio) performed statistically significantly better than the learned target models. In scenario 6, BTLSM (unadjusted, KL, ratio) also performed statistically significantly better than the learned target models. Results also showed that if the source size is small (50), transfer learning may still bring some advantages in some scenarios. In scenario 2, the BTLSM models performed statistically significantly better than the learned target model when there were 50 target visits; and in scenario 6, the BTLSD models performed statistically significantly better than the learned target model.

In the intubation experiment (141 scenarios), hypothesis 1 is supported. When the source size was 8000 and the target was 50, the BTLSM models performed statistically significantly better than the learned target models. As distance between the target and source distributions (measured by KL) increased, the benefits of transfer learning decreased.

For influenza detection among suspected visits (real-world data) experiments, hypothesis 1 is supported in the first two weeks. At the second week, there were about 20 influenza cases and

200 laboratory test negative suspected visits in each region (IH or UPMC). Sharing data or model from UPMC to IH improved the AUC from 0.67 to 0.71 and sharing data or model from IH to UPMC improved the AUC from 0.59 to 0.61. At week 4, UPMC had 71 influenza cases and 388 laboratory test negative suspected visits; transferring from IH to UPMC was still beneficial (local UPMC model: 0.60, BTLSM or BTLSD model: over 0.63). Transfer learning from UPMC to IH became unnecessary, at week 4, IH had already cumulated 94 influenza cases and 550 laboratory test negative suspected visits, which may be sufficient for local model development. More discussions about similarity between the source and target distributions and sample sizes are provided in later Sections 6.3 and 6.4 below.

For influenza detection among general visits (real-world data) experiments, hypothesis 1 is supported when transferring data or a model from IH to UPMC for the first week. Sharing a model boosted AUC from 0.73 to 0.74 (not statistically significant), and sharing data boosted AUC to 0 78 (statistically significant). If IH had started the influenza season one week earlier, with more IH transferred knowledge, the final model could further reach 0.80. Transferring learning did not show significant improvement in some other situations. This may be for three reasons: (1) influenza detection among emergency room visits is a relatively easy task because influenza visits usually show different symptoms than the general emergency room visits, the majority of which are injury cases and cardiovascular events; (2) even at the first week, each region has already accumulated over 3,000 controls (non-influenza cases) in the training dataset, which may be large enough for model development; (3) When the target region has accumulated enough training samples, extra information from the source could become noise for the target pattern discovery, leading to a slight performance drop.

6.2 Whether and When Transfer Learning Is Better Than the Source Model

The main difference between transfer learning and traditional machine learning approaches is whether to assume the training and test data follow the same distributions or not. If the data from two institutions/regions have identical distribution, one institution can directly adopt the model from another institution, without concern for performance loss. If not, transfer learning algorithms aim to reduce the performance loss by integrating the knowledge in the transferred model with the patterns hidden in the few training cases of the target institution.

The second hypothesis is that transfer learning is better than direct adoption of the source model. This hypothesis could be valid when three conditions are met: (1) the learned source model is not good enough; (2) the source and target's distribution difference is relatively large and there is a need to adjust the transferred source model; (3) the target training data is large enough to make the right adjustment. In fact, these conditions tend to be common when the source size is too small to develop a well-performed model and the target size is large enough to make the adjustment.

The intubation simulation experiment demonstrated that after integrating with 50 target samples, the BTLSM models could perform significantly better than the learned source models. The advantage of the BTLSM models over the source models became larger as the KL increased. In this experiment, 50 target samples seemed to be large enough for the right adjustment.

In the influenza simulation experiments, when source size was 8000 and target size was 50, the BTLSM-unadjusted models performed better than the learned source model in scenarios 1, 2, 4, 5, and 6 (the first four were statistically significant). In scenario 3, the learned source model already performed well on the target data (learned source model: 0.9447, true source model: 0.9459; true target model: 0.9643). When both source size and target size reached 8000, the BTLSM models significantly boosted the performance of the source model in scenarios 1, 2, 4,

and 6. While in the remaining scenarios 3 and 5, the BTLSD models outperformed the learned source models.

For the influenza detection from suspected visits (real-world data) experiments, at the first week, the transferred source model's performance could be significantly boosted by the BTLSM algorithm with the target institution's training data. For the influenza detection from general visits (real-world data) experiments, when transferring from IH to UPMC, models developed with data from two institutions performed significantly better than the learned IH models at week 2, week 4, and week 8. When transferring from UPMC to IH, models developed with data from two institutions performed significantly better than learned IH models at week 4, and week 8.

6.3 Similarity Measurements for Transfer Learning Tasks

Similarity between source and target distributions is an important factor when determining the effects of transfer learning. This study has identified four measurements to indicate similarity.

The proportion of shared features among all features from both domains is the first indicator. Features that only appear in source training data or the source model do not have any discriminative ability for the target task because they are all missing in the target data. If the source and target only share a very small portion of features, then the transferred data/model may have very little useful information to leverage, and transfer learning may not be effective. For example, in the influenza simulation experiment, scenario 4 only had 30% (six out of twenty features) shared features, and this scenario did not show positive transfer effects over the target model.

Another indicator of similarity is the proportion of shared features in the target feature space. To the extreme, if the target feature space is a subset of the source feature space, then all of

the target features have information with the shared data even if the first indicator has a small number. In this situation, if the source model was developed through feature selection techniques with the source training data, it is still not guaranteed that all target features are included in the shared source model.

The third indicator is the average KL divergence for the shared features. Like the transfer learning task, KL divergence is asymmetric. The $\mathrm{KL}(\mathrm{target} \,||\, \mathrm{source})$ measures the amount of information lost when source distribution is used to approximate the target distribution. As shown in Figure 6 and 7, continuous decreases in performance of BTLSM models and source models happened as the KL increased. In the intubation simulation experiment, when the KL reached about 3 (average KL 0.33), transfer learning became unnecessary.

Since the KL could be impacted by the number of shared features, the average KL (KL divided by the number of features) may be a better indicator of similarity. In influenza season 2014-15 (real-world data), for influenza vs. NILI, the KL between target and source was about 14, but the average KL over features was about 0.20 for the first two weeks (Table 29).

Table 29 KL between target and source distributions in datasets of UPMC and IH

Classification Task	Datasets	Source	Target	KL	Average KL for each node (KL divided by 69 features)
Influenza vs. NILI	Week1 training datasets	IH	UPMC	14.74	0.21
		UPMC	IH	13.63	0.20
	Week1-2 training datasets	IH	UPMC	14.25	0.21
		UPMC	IH	12.88	0.19
	Week 1-4 training datasets	IH	UPMC	13.7	0.20
		UPMC	IH	11.95	0.17
	Week 1-8 training datasets	IH	UPMC	12.66	0.18
		UPMC	IH	10.94	0.16
	Testing datasets	IH	UPMC	11.36	0.16
		UPMC	IH	10.17	0.15
Influenza vs. Non-influenza	Week1 training datasets	IH	UPMC	9.5	0.14
		UPMC	IH	10.35	0.15
	Week1-2 training datasets	IH	UPMC	8.85	0.13
		UPMC	IH	9.6	0.14
	Week 1-4 training datasets	IH	UPMC	8.22	0.12
		UPMC	IH	8.82	0.13
	Week 1-8 training datasets	IH	UPMC	7.59	0.11
		UPMC	IH	8.06	0.12
	Testing datasets	IH	UPMC	7.14	0.10
		UPMC	IH	7.34	0.11

Sometimes, a low average KL may indicate that it is not necessary to change the source model. The intubation experiment showed that the learned source model's performance was closer to the BTLSM models when the KL was low. In table 21, the KLs of datasets for the influenza vs. non-influenza task are lower than the KLs of datasets for influenza vs. NILI task, which may

partially explain the fact that the source model did not need to be changed much in the influenza vs. non-influenza tasks.

However, the empirical estimation of KL could be affected by the availability of target data. Since KL measures the distance between target and source, a few target training samples may not provide a reliable estimation. From the KL calculation formula, KL will be very large when the probability of one particular instantiation is high in the target and low in the source. If an instantiation appears in the very few target training samples but it never appears in the large source dataset, then this results in a large KL between target and source.

A large percentage of shared features and a low KL could indicate a potential positive transfer over the local target model, but it is not guaranteed. A fourth measurement could be the performance of the source model on the training target dataset. If a transferred model could reach an expected performance on a few training target samples, then that would further increase my confidence in the success of transferring knowledge. If this were already good enough, further adjustment of the source model might be unnecessary.

In summary, if a transfer learning task has a high percentage of shared features among all features, a high percentage of shared features among target features, a low average KL, and a good performance on a preliminary test of the source model on target data, then the transferring knowledge/data may be successful.

6.4 How Does the Target Data Size Impact the Effectiveness of Transfer Learning

The classification task in the target and the similarity between target and source impact whether a target training dataset is enough or not. For example, with the "non-transferrable" scenario 4 in the influenza simulation experiments, local target models developed with 50 samples had already reached an AUC of 0.94, while the true source model only had an AUC of 0.71. In the intubation simulation experiments (target size =50), after the KL reached 3 (average KL reached 0.33), the benefits of transfer learning disappeared. For the influenza vs. non-influenza task in the 2014-15 season, the target training data contained over 3000 samples at week1 and transfer learning did not show much benefit in this task. For the influenza vs. NILI task, the target training data contained around 120 samples at week 1, and fewer than 260 samples at week 2, where transfer learning showed benefits.

6.5 What Should Be Shared: Source Model or Source Data

The idea of sharing models across institutional boundaries is appealing for the biomedical domain because it usually is associated with less concern about patient privacy and data security. One may worry about the possibility of information loss when sharing a model instead of original data. However, a well-developed model can have fewer noisy features and may provide a more compact way to represent the essential knowledge that is worth sharing.

My experiments support that sharing models is feasible. In the first week of influenza season 2014-15, a UPMC case detector for influenza vs. non-influenza visits reached an AUC of 0.88 when it was tested on IH data. In cases where the target institution does not have accumulated

data or the capability to retrieve data and develop a model currently, then a well-developed model from another institution could be a feasible alternative.

Moreover, this study showed that the BTLSM transfer learning algorithm was able to further boost the performance of the shared source model. In the influenza simulation experiments (source size=8000, target size=50), the BTLSM models (with the learned source model as prior model) performed significantly better than the learned source model in scenarios 1, 2, 4, and 5. And these BTLSM models even performed better than the BTLSD models. Therefore, it can be inferred that when the target dataset is small, it may be a better idea to start the model searching process with a prior source model.

Unfortunately, the designed weighting strategies did not show significant benefits for transfer learning algorithms over the unadjusted approach. One explanation is that almost all similarity related weighting strategies rely on accurate estimation of the empirical KL, which can be a very challenging task when there are very few target training samples available. The chosen weighting strategy for source equivalent sample size, 2^{-KL}, becomes 1 when KL is 0 and is near 0 when KL is extremely large. This heuristic and Bayes factor are used to shrink the source information based on distance. However, the extent of shrinkage may need to be further explored in future work.

Sharing source data could be more valuable than sharing source model when the target has many samples. In the influenza simulation experiments, the BTLSD models were shown to perform better than BTLSM models when the target size increased to 8000 in all six scenarios. After region A receives data from region B, region A can certainly develop a region B model first and then use a BTLSM model to integrate region A's information into the region B model, or region A can use the mixed data for model development. Directly using source data instead of the

source model allows for fewer restrictions regarding the independences and parameterization of source distributions and allows the combination of source and target information more smoothly for a more complicated model development.

After sharing source data, I may either use mixed data to obtain a general model or use the BTLSD algorithm to obtain a model customized for the target domain. I conducted some preliminary experiments on real-world datasets. In general, I did not find any advantage of the BTLSD algorithm over the general model approach. That may be because the feature space of the source and target was identical in these datasets (by using one NLP parser for both regions), and BTLSD algorithm's biggest advantage of adding target-specific features did not apply here. A further investigation may involve experimentally removing some influential features from source data.

6.6 Calibration Error Comparison

In addition to the discriminative ability comparisons, I compared models' expected calibration error to study their calibration capability. In the influenza synthetic experiments, the learned target models usually had a lower error even when they were built with a small training sample size. The transfer learning models usually had a higher calibration error than the learned target models, and a lower error than the learned source models.

The results for the real-world data experiments are slightly different. When detecting influenza from suspected visits, the BTLSD-K2 models usually had the lowest error of all the models, including both the learned source and learned target models. When detecting influenza from general visits, the BTLSM models usually had the lowest error.

These findings indicate that the proposed transfer learning algorithms could always reduce the calibration error of the transferred source models, and their calibration may be better at times than the learned target models'.

6.7 Limitations

One limitation of the evaluation in the experiments is that I mainly used a discrimination measure, AUC, and a calibration measure, expected calibration error. To use a case detection model as a diagnosis tool, I need to further identify an appropriate probability threshold and estimate the positive predictive values to indicate the tool's practical value. Other classification measurements can be calculated and compared, such as prevision recall area under curve, mean absolute error, and F1. Moreover, how the proposed transfer learning algorithm will impact the threshold and precision is worthy to further explore.

Another limitation of the experiments is a lack of exhaustively study of all feature selection approaches. A further exploration of the relationship between feature selection and transfer learning performance in different heterogeneous scenarios would be very valuable.

This dissertation conducted experiments in different scenarios, and for some the proposed transfer learning algorithms did not bring benefits for the target domain. Although I further investigated these scenarios, it would be worthwhile to try many other transfer learning algorithms to further confirm the non-transferability of these scenarios in future work. Moreover, it is also worthwhile to try other data transfer algorithms and compare their performance with the performance of the BTLSM algorithm to further confirm the feasibility of the BTLSM algorithm when the target size is small.

6.8 Conclusions

My dissertation developed an innovative transfer learning framework to share data or a model under heterogeneous scenarios. Heuristic scores were designed to integrate source information with target data, while allowing the inclusion of target-specific features for improved local customization. The developed BTLSM algorithm can be viewed as a new Bayesian network learning algorithm that combines (partial) knowledge and data, where partial coverage of feature spaces and various confidence levels of features in that prior model are allowed.

Both synthetic and real-world datasets were used to test two hypotheses: 1) a transfer learning model that is learned using source knowledge and target data performs classification in the target context better than a target model that is learned solely from target data; 2) a transfer learning model performs classification in the target context better than a source model. An extensive analysis was conducted to investigate the conditions in which these two hypotheses are supported and not supported, and more generally, provided insight into the factors that affect the effectiveness of transfer learning, providing empirical guidance about when and what to share.

I recommend calculations of percentage of shared features among all features, percentage of shared features among target features, average KL, and performance of source model on target data indicators of the potential of success of transfer learning tasks. When the target domain has few samples, the sharing model strategy reaches a comparable or even better performance than the sharing data strategy. When the target domain has many samples, sharing data brings more benefits for knowledge discovery.

To my knowledge, this research is the first study on biomedical model transfer in heterogeneous scenarios. Experimental results indicate an impact of task similarity on the success of transfer learning. Results showed that transfer learning models usually perform slightly better

than the learned target model, especially when the source sample size is large, and the target sample size is small. Moreover, results indicate that proposed transfer learning algorithms can increase the discriminative ability of the shared source model and at the same time reduce the calibration error in most synthetic experiments.

This is also the first study on transfer learning techniques for infectious case detection tasks. This dissertation demonstrated the capability of quick development of a case detection system after accumulating about 20 cases among 10,000 general emergency room visits with good transferability and the ability to be localized. For differential diagnosis between cases and suspected visits, this study also indicated the value of sharing data or a model when training data has fewer than 500 visits.

7.0 Contributions and Future Work

This research developed an innovative transfer learning framework to share data or a model under heterogeneous scenarios. A comprehensive analysis was conducted on synthetic and real-world biomedical datasets, especially for infectious case detection tasks. This work potentially contributes to the fields of machine learning, medical informatics and disease surveillance.

7.1 Contributions to Machine Learning

This dissertation developed a framework to transfer knowledge (in a format of a mathematical model) across institutional boundaries. The framework fully considers feature space differences and distribution differences. It is able to localize the transferred model by integrating the represented source pattern with target training samples in a new domain. An innovative score has been developed for model searching. This score considers both source model information (structure and conditional probabilities) and data pattern in the target. It further integrates target-specific features, making the localization of a transferred model more flexible.

To the best of my knowledge, the BTLSM algorithm is the first transfer learning algorithm for Bayesian network transfer for heterogeneous scenarios. Compared to the first model-transfer technique for heterogeneous scenarios (Mozafari et al. 2016), the BTLSM algorithm does not require large target samples for target model development, and it integrates much more parameter information from the source model rather than just a one-dimensional offset parameter. Our algorithm has much fewer restrictions as well: it neither assumes linear correlations between

features and the classification task, nor assumes a very similar feature distribution in a transformed common dimension.

In fact, the BTLSM algorithm can also be viewed as a new algorithm for Bayesian network learning from a combination of knowledge and statistical data, where knowledge is represented as a prior Bayesian network. Compared to Heckerman's approach (Heckerman et al. 1995), the BTLSM algorithm reduces two constrains of the prior network: (1) it does not need to cover all predictive features, allowing partial knowledge from another institution or from human experts; (2) it allows different levels of confidence for different features (by setting different equivalent sample sizes). Compared to the EBMC algorithm (Cooper et al. 2010), the BTLSM algorithm uses a prior model to initiate the model search, and the prior model's knowledge is used as both a parameter prior and a structure prior. For the parameter prior, the marginal likelihoods calculation now uses pseudo counts estimated from the prior model and equivalent sample sizes instead of the K2 or BDeu approach in EBMC. The structure prior is calculated based on the symmetric difference between the parent set of a node in a candidate Bayesian network structure and the parent set of that node in a prior model, so users do not need to provide estimates of the expected number of predictors.

Finally, by conducting a comprehensive evaluation in both synthetic datasets and real-world datasets, this thesis provides empirical guidance about when to share and what to share (model or data). The degree of overlapping feature sets, the similarity of distributions of shared features (average KL), and the discriminative ability of the source model on target data are indicators for the success of transfer learning. To our surprise, sharing model achieved comparable performance as sharing data when the target domain had few samples. Sharing the data was superior when the target domain had many samples for a more complicated knowledge discovery.

7.2 Contributions to Biomedical Informatics

In this big data era, sharing medical data and knowledge has become a key strategy for secondary use of electronic medical records, research on patient-centered outcomes, and knowledge discovery from observational health data. Regarding sharing of data, collaborators are still exploring ways to address concerns about patient privacy, to establish rules to govern data usage, and to provide mutual benefits to different stakeholders.

To meet these challenges, this dissertation establishes a new sharing channel, sharing well-developed probabilistic models. Compared to sharing data, sharing models across institutions' boundaries involves fewer restrictions and privacy concerns. The shared probabilistic models could still have immense value because they are able to capture the uncertainties of correlations between medical findings and conditions. Our experiments demonstrated the value of model-sharing in both synthetic data experiments and real-world data experiments.

Moreover, this dissertation developed and evaluated an automatic mechanism to localize a transferred model for a new institution. Experimental results demonstrated that this mechanism could align a source model's feature space with data elements in the new institution, and also adjust the source model's parameters to better reflect the data patterns in the new institution.

Our results also showed the impact of task similarity on the success of transfer learning. To eliminate dissimilarities, it would be better for collaborators to establish standard definitions of terminologies, and to share natural language processing parsers.

Finally, the developed framework provides a mechanism to enable a learning healthcare system, where models for one institution could be tuned using the most recent data. This adjustment is usually needed when the electronic medical systems are upgraded, new medications and procedures are added, or when hospital policies are changing.

7.3 Contributions to Disease Surveillance

Population disease control and prevention heavily rely on quick and accurate disease surveillance. We have successfully developed probabilistic case detection systems that identify suspected cases from emergency room visits using natural language processing parsers and Bayesian network models (López Pineda et al. 2015, Tsui et al. 2017, Ye et al. 2017). Leveraging routinely collected medical records, these systems enable earlier detection, lower costs, and larger population coverage than traditional surveillance systems.

This dissertation demonstrates that a case detection system may be quickly developed in about two weeks after accumulating about 20 cases among 10,000 general emergency room visits. If the symptoms of an emergent unknown disease have been covered by a general purpose medical natural language processing parser (e.g., MedLEE, cTakes, etc), then a quick development of a probabilistic case detection system could be possible. Moreover, this system could serve as a new format of case definition for sharing across different regions for disease control and prevention.

This is the first study on transfer learning techniques for an infectious disease case detection. My results showed that sharing data/model could be very valuable during the first two weeks for differential diagnosis between cases and suspected visits, which is a more challenging task given the availability of fewer training samples in the early stages of an outbreak. The

performance of transferred models can be further boosted using my transfer learning algorithms. This type of Bayesian case detection system may serve as a clinical decision support tool to assist differential diagnosis and it is sharable and localizable.

7.4 Future Work

This study only conducted experiments on two region sharing scenarios. A more common scenario in biomedical collaborations and infectious disease outbreaks is one target domain with multiple source domains. In fact, the transfer learning framework here can be extended for these scenarios. For the BTLSM algorithm, the score consists of two parts: the marginal likelihood and the structure prior. From Eq. 10, it is easy to see that the marginal likelihood is decomposable into one component for each feature, for each of which I use the sum of pseudo counts from all source models that include the feature. The structure prior part of the equation is a little complicated, but the calculation still can be decomposed into one component for each feature, for each of which the symmetric difference may be estimated as the difference between a candidate Bayesian network structure and the union of parent sets of that node in all source models that include the feature. This extension is worth further exploration. For the BTLSD algorithm, the K2 score is also decomposable, and the calculation for each feature may only include data from the source domains that include the feature.

Another change to explore for the BTLSM algorithm is to build a more relevant model for the target using the source data. The first step of the current BTLSM algorithm is to remove the source-specific features. An alternative would be to request the source to build a model for the target using features that are shared by the source and target datasets. In the example in Figure 2,

for instance, suppose that {S2, S3, S4, S5} are variables shared by the source and target. I would learn a BN on these four variables using the source data. That BN might include S5, whereas the current method shown in Figure 2 would not because removing S1 will also remove S5. This alternative algorithm is worth future study because more target related features could be kept in the source model.

Another natural extension of the transfer learning framework here is to reduce the restrictions on the source model. The current BTLSM model uses a Bayesian network as parameter priors and a structure prior. If another structure prior is used, the parameter prior does not necessarily require the source model to be in the format of a Bayesian network model. The prior model could be of any format as long as it represents a joint probability distribution. This would loosen the constraints on the BTLSM algorithm. The source model could be any probabilistic models or even a simple contingency table.

A further extension of the work is to explore other model formats for model-transfer, such as deep neural networks. It has been shown that the first few layers of deep neural networks represent more general patterns that could be sharable between source and target. Current transfer deep learning techniques have not considered the scenario when the source and target feature spaces are different, yet this phenomenon is very common in medical data. The exploration of deep transfer learning for heterogeneous scenarios is important for medical knowledge discovery, especially when multiple data sources (e.g., images, natural language parsed clinical findings, and laboratory results) are available. Because deep neural networks usually have a large number of parameters that usually require a large training dataset, semi-supervised transfer learning with a large unlabeled target training datasets and a few labeled target samples could be the first scenario for exploration.

This dissertation developed and applied the transfer learning techniques for region A-B scenarios. In fact, even in the same hospital, the shift of feature space and distributions is also very common. For example, newly approved medicines and treatment procedures are new features that never appear in historical data. The adaption of electronic medical systems may lead to many template updates that are associated with the changes of structured datasets. These changes of patient population and medical practice activities always lead to degrees of distribution shift. Transfer learning techniques can be leveraged to smooth the knowledge transition in individual hospital. The potential of transfer learning for a learning healthcare system will be a very interesting topic for further exploration.

It also seems worthwhile to explore applying the transfer learning algorithms developed here to other classification tasks, such as automated patient cohort identification for clinical research, patient readmission risk profiling, adverse clinical events identification, and adverse medication error detection.

For public health surveillance, automatic case detection techniques and their transferability can be further studied for other infectious diseases, including other respiratory diseases, gastrointestinal infections, and sexually transmitted diseases.

Appendix A Pseudocodes of BTLSD Algorithm

ALGORITHM BTLSD (D_s, D_t, T_s, T_t, ML, W, u)

D_s: source dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: target dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

T_s: the threshold used to determine the set of relevant features in the source

T_t: the threshold used to determine the set of relevant features in the target

ML: the machine learning approach, Naïve Bayes or K2, to build Bayesian network model

W: weighting approach for the source instances

u: maximum number of parents

V_s ← the selected source feature set

V_t ← the selected target feature set

/*general feature set is the intersection between relevant source features and relevant target features*/

F_G ← $V_s \cap V_t$

/*target-specific feature set is the remaining features in relevant target feature set*/

F_{TU} ← V_t not in the source

F_{TI} ← $V_t - F_G - F_{TU}$

/*develop Bayesian network model for the target using source dataset and target dataset*/

If (W=BTLSD-R) { $\mathbf{W}_s$ ← BTLSD-R (D_s, D_t) }

Else if (W=BTLSD-KL) { $\mathbf{W}_s$ ← BTLSD-KL (D_s, D_t, F_G) }

Else if (W=BTLSD-FS-KL) {$\mathbf{W}$s ← BTLSD-FS-KL (D_s, D_t, F_G)}

If (ML=naïve Bayes) { BN_t ← DevelopNaiveBayes (D_s, D_t, F_G, F_{TU}, F_{TI}, $\mathbf{W}_s$) }

Else if (ML=K2) { BN_t ← DevelopK2 (D_s, D_t, F_G, F_{TU}, F_{TI}, u, $\mathbf{W}_s$) }

Return BN_t

END ALGORITHM

PROCEDURE DevelopNaiveBayes (D_s, D_t, F_G, F_{TU}, F_{TI}, $\mathbf{W}_s$)

D_s: Source dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: Target dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

F_G: generalizable feature set, including the class node O

F_{TU}: Target-specific features that are unobserved in source data

F_{TI}: Target-specific features that are irrelevant in source data

$\mathbf{W}_s$: a vector containing weights for all features in F_G. W_{sj} is the weight for feature X_j.

/*calculate conditional probability tables*/

for (X_j in F_G) {

$$\mathrm{BN_t}\left[\left(X_i = x_{ik} \mid O = O_j\right)\right] = \frac{D_t\left(\mathrm{X_j} = \mathrm{x_{jk}},\ O = O_j\right) + \mathrm{N_i} \times \mathrm{P_s}\left(\mathrm{X_j} = \mathrm{x_{jk}},\ O = O_j\right) + 1}{D_t\left(O = O_j\right) + \mathrm{N_i} \times \mathrm{P_s}\left(O = O_j\right) + r_j}$$

}

for (X_j in F_{TU} or F_{TI}) {

$$\mathrm{BN_t}\left[\left(X_i = x_{ik} \mid O = O_j\right)\right] = \frac{D_t\left(\mathrm{X_j} = \mathrm{x_{jk}},\ O = O_j\right) + 1}{D_t\left(O = O_j\right) + r_j}$$

}

return $\mathrm{BN_t}$

END PROCEDURE

PROCEDURE DevelopK2 (D_s, D_t, F_G, F_{TU}, F_{TI}, u, $\mathbf{W}_s$)

D_s: source training dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: target training dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

F_G: generalizable feature set

F_{TU}: target-specific features that are unobserved in source data

F_{TI}: target-specific features that are irrelevant in source data

u: maximum number of parents

$\mathbf{W}_s$: is a vector containing weights for all features in F_G. W_{si} is the weight for feature X_i

PARENTS_SET ← {} /*this set will contain a set of sets; each will contain the parent set of a node */

V ← $F_G \cup F_{TU} \cup F_{TI}$

for (X_i in V) {

SET_i ← {}

if ($X_i \in \{F_{TU} \cup F_{TI}\}$) { $\text{score}(i, \text{SET}_i) = \prod_{j=1}^{q_i} \frac{\Gamma(r_i)}{\Gamma[r_i + D_t(pa(X_i) = \pi_{ij})]} \prod_{k=1}^{r_i} \frac{\Gamma[1 + D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij})]}{\Gamma(1)}$ }

else if ($X_i \in F_G$)

{ $\text{score}(i, \text{SET}_i) =$

$\prod_{j=1}^{q_i} \frac{\Gamma(r_i)}{\Gamma[r_i + W_{si} \times D_s(pa(X_i) = \pi_{ij}) + D_t(pa(X_i) = \pi_{ij})]} \prod_{k=1}^{r_i} \frac{\Gamma[1 + W_{si} \times D_s(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij})]}{\Gamma(1)}$

}

$S_{current}$ ← $\text{score}(i, \text{SET}_i)$

SEACH ← true

While SEACH is true and | SET_i | < u {

a is the node maximizing score $(i, \text{SET}_i \cup \{a\})$

S_{temp} ← $\text{score}(i, \text{SET}_i \cup \{a\})$

If $S_{temp} > S_{current}$ {

$S_{current} = S_{temp}$

$\text{SET}_i \leftarrow \text{SET}_i \cup \{a\}$

}

Else SEACH ← false

}

put SET_i in Parents_Set

}

BN_t ← Generate the structure of BN_t use Parents_Set

for (X_j in F_G) {

$$BN_t\left[\left(X_i = x_{ik} \middle| pa(X_i) = \pi_{ij}\right)\right] = \frac{D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + N_i \times P_s(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + 1}{D_t(pa(X_i) = \pi_{ij}) + N_i \times P_s(pa(X_i) = \pi_{ij}) + r_j}$$

}

for (X_j in F_{TU} or F_{TI}) {

$$BN_t\left[\left(X_i = x_{ik} \middle| pa(X_i) = \pi_{ij}\right)\right] = \frac{D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + 1}{D_t(pa(X_i) = \pi_{ij}) + r_j}$$

}

return BN_t

END PROCEDURE

PROCEDURE BTLSD-R (D_s, D_t)

D_s: source dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: target dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

If $N_t < N_s$ {

$W \leftarrow N_t/N_s$

}

Else W = 1

$\mathbf{W}_s$ is a vector containing weights for all features in F_G. Here, all weights are equal to W

Return $\mathbf{W}_s$

END PROCEDURE

PROCEDURE BTLSD-KL (D_s, D_t, F_G)

D_s: source dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: target dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

$P_t(\boldsymbol{x}_k, y_m) \leftarrow \frac{D_t(\mathbf{X}=\boldsymbol{x}_k, Y=y_m)}{N_t}$ /* $\boldsymbol{x}$ is a vector including all predictive nodes in F_G*/

$P_s(\boldsymbol{x}_k, y_m) \leftarrow \frac{D_s(\mathbf{X}=\boldsymbol{x}_k, Y=y_m)}{N_s}$

If $P_s(\boldsymbol{x}_k, y_m) = 0$ { $P_s(\boldsymbol{x}_k, y_m) \leftarrow 0.00000001$}

$KL(P_t \,||\, P_s) = \sum_m \sum_k P_t(\boldsymbol{x}_k, y_m)\, log \frac{P_t(\boldsymbol{x}_k, y_m)}{P_s(\boldsymbol{x}_k, y_m)}$

$W = 2^{-KL(P_t \,||\, P_s)}$

$\mathbf{W}_s$ is a vector containing weights for all features in F_G. Here, all weights are equal to W

Return $\mathbf{W}_s$

END PROCEDURE

PROCEDURE BTLSD-FS-KL (D_s, D_t, F_G)

D_s: source dataset $\{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{N_s}$

D_t: target dataset $\{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{N_t}$

for each X_i in F_G {

$$P_t\left(X_i = x_{ik},\ pa(X_i) = \pi_{ij}\right) \leftarrow \frac{D_t(X_i = x_{ik},\ pa(X_i) = \pi_{ij})}{N_t}$$

$$P_t\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right) \leftarrow \frac{D_t(X_i = x_{ik},\ pa(X_i) = \pi_{ij})}{D_t(pa(X_i) = \pi_{ij})}$$

$$P_s\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right) \leftarrow \frac{D_s(X_i = x_{ik},\ pa(X_i) = \pi_{ij})}{D_s(pa(X_i) = \pi_{ij})}$$

If $P_s\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right) = 0$ $\{ P_s\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right) \leftarrow 0.00000001\}$

$$KL_i(P_t \,||\, P_s) = \sum_j \sum_k P_t\left(X_i = x_{ik},\ pa(X_i) = \pi_{ij}\right) log \frac{P_t\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right)}{P_s\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right)}$$

$$W_i = 2^{-KL_i(P_t \,||\, P_s)}$$

Save W_i to $\mathbf{W}_s$

}

For the class node O {

$$KL_O(P_t \,||\, P_s) = \sum_k P_t\,(O = O_k)\, log \frac{P_t\,(O = O_k)}{P_s\,(O = O_k)}$$

$$W_o = 2^{-KL_O(P_t \,||\, P_s)}$$

Save W_o to $\mathbf{W}_s$

}

Return $\mathbf{W}_s$

END PROCEDURE

Appendix B Pseudocodes of BTLSM Algorithm

```
ALGORITHM BTLSM (BN_s, N_s, D_t, O, T, Approach)
        BN_s: a Bayesian network model developed with the instances of the source
        N_s: the size of training instances that were used to develop the source model
        D_t: instances of the target
        O: the class node
        T: the threshold to determine the target-specific feature set
        Approach: weight strategy /*unadjusted, ratio, KL, features-specific KL*/
        V_t ← selected feature set using information gain approach with threshold T
        BN_sc ← from BN_s, remove features that are completely missing in V_t
        BN_sci ← add target-specific features to BN_sc. These features do not connect to any feature.
        N ← GetEquivalentSampleSize (BN_sc, N_s, D_t, Approach) /* see procedure code */
        BN_t ← CUT (BN_sc, BN_sci, D_t, O, BN_c, N) /* see procedure code */
        SEACH ← True
        While SEACH is True {
                BN_t, CHANGE, V_t ← GROW (V_t, BN_sc, BN_sci, D_t, O, BN_c, N) /* see procedure code */
                If (CHANGE = false) { SEACH ← false}
                Else {BN_t ← PRUNE (BN_sc, BN_sci, D_t, O, BN_c, N) } /* see procedure code */
        } /*End While*/
        BN_t ← PARAMETERIZATION (BN_sc, BN_t, D_t, N)
        Return BN_t
END ALGORITHM


PROCEDURE CUT (BN_sc, BN_sci, D_t, O, BN_c, N)
        BN_sc: the cleaned source model that is obtained by removing source-specific nodes from source model
        BN_sci: the prior model for source knowledge transfer. It is obtained by adding target-specific nodes as
                orphan nodes to BN_sc
        D_t: instances of the target
        O: the class node
        BN_c: current target model
        V ← a set of arcs in BN
        SEACH ← true
        While SEACH is true {
            Let C be an arc in V that increases SCORE (BN_sc, BN_sci, D_t, O, BN_c, N) the most when being removed
            If C exists {
                BNc ← remove C from current BN_c
                V ← V - C
             }
            Else {SEACH ← false}
        }
END PROCEDURE
```

PROCEDURE GROW (V_t, BN_{sc}, BN_{sci}, D_t, O, BN_c, **N**)

V_t: candidate features to be added into model

BN_{sc}: the cleaned source model that is obtained by removing source-specific nodes from source model

BN_{sci}: the prior model for source knowledge transfer. It is obtained by adding target-specific nodes as orphan nodes to BN_{sc}

D_t: instances of the target

O: the class node

BN_c: current target model

N: a vector containing equivalent sample size for each node in BN_{sc}

CHANGE ← false

SEARCH ← true

while SEARCH is true {

Let X be a feature in V_t that increases score SCORE (BN_{sc}, BN_{sci}, D_t, O, BN_c, **N**) the most when being added as a parent of O

If X exists {

BN_t ← to current BN_t, add X as a parent of O

CHANGE ← true

V_t ← V_t - X

}

Else SEARCH ← false

} /* End while */

Return BN_t, CHANGE, V_t

END PROCEDURE

```
PROCEDURE PRUNE (BN_sc, BN_sci, D_t, O, BN_c, N)
        BN_sc: the cleaned source model that is obtained by removing source-specific nodes from source model
        BN_sci: the prior model for source knowledge transfer. It is obtained by adding target-specific nodes as
              orphan nodes to BN_sc
        D_t: instances of the target
        O: the class node
        BN_c: a candidate Bayesian network model
        N: a vector containing equivalent sample size for each node in BN_sc
        V ← a node set that includes parents of the class node O
        BN ← convert BN to statistically equivalent BN by making all parents of O be children of O,
               and create a full connected set of arcs among these children nodes
        for each feature X in these children nodes {
              SEARCH ← true
              while SEARCH is true{
                 Let C be an arc (connecting to X) that increases SCORE (BN_sc, BN_sci, D_t, O, BN_c, N) the most when
                 being removed
                 If exists C {
                      BN_c ← remove C from current BN_c
                     SEARCH ← true
                  }
                  Else SEARCH ← false
             }/*end while */
         }/*end for */
         Return BN
END PROCEDURE


PROCEDURE SCORE (BN_sc, BN_sci, D_t, O, BN_c, N)
        BN_sc: the cleaned source model that is obtained by removing source-specific nodes from source model
        BN_sci: the prior model for source knowledge transfer. It is obtained by adding target-specific nodes as
              orphan nodes to BN_sc
        D_t: instances of the target
        O: the class node
        BN_c: a candidate Bayesian network model for scoring
        N: a vector containing equivalent sample size for each node in BN_sc
        Score ← logMarginalLikelihood (BN_sci, D_t, O, BN_c, N) + logStructureScore (BN_sc, BN_c, N)
        Return Score
END PROCEDURE
```

```
PROCEDURE LogMarginalLikelihood (BN_sci, D_t, O, BN_c, N)
        BN_sci: the prior model for source knowledge transfer. It is obtained by removing source-specific nodes
              from the source model and then adding target-specific nodes.
        D_t: instances of the target training data
        O: the class node
        BN_c: a candidate Bayesian network model for scoring
        N: a vector containing equivalent sample size for each node in BN_sc
        logMarginal = 0
        for C from the first instance to the last instance of  D_t {
           iNodeValue ← iNode value in C
           logNominator = 0
           logDenominator = 0
           for  iNodeValue from first state to last state of iNode {
             logJointProb = 0
             logJointProb←logJointProb
                          +GetLogProb (BN_sci, BN_c, iNode, N_i, iNodeValue, iNodeParentValue, D_t^{c-1})
             for (each child of iNode, denoate as aNode){
                 logJointProb←logJointProb
                              + GetLogProb(BN_sci, BN_c, aNode, N_a, aNodeValue, aNodeParentValue, D_t^{c-1})
             }
              logDenominator = logPlus (logDenominator, logJointProb) /*logPlus(logX, logY) = log(X+Y)*/
              if (iNodeValue = iNode value in C)
              {   logNominator = logJointProb   }
           }
           logMarginal ← logMarginal + ( logNominator – logDenominator )
        }
        Return logMarginal
END PROCEDURE
```

PROCEDURE GetLogProb (BN_{sci}, BN_c, iNode, N_i, iNodeValue, iNodeParentValue, D_t^i)

BN_{sci}: the prior model for source knowledge transfer. It is obtained by removing source-specific nodes from the source model and then adding target-specific nodes.

BN_c: a candidate Bayesian network model for scoring

iNode: the node for probability calculation

N_i: the equivalent sample size for iNode

iNode: the node for probability calculation

iNodeValue: the value of the node for probability calculation

iNodeParent: the parent set of the node in BN_c

iNodeParentValue: the value of the parent set

D_t^i: a set of instances of the target domain, from the 1st instance to the ith instance

$N_{ij\_prior} = 0$

$N_{ij} = 0$

for V from first state to last state of iNode {

$N_{ijk\_prior\_temp} \leftarrow$ GetPriorCount (BN_{sci}, N_i, BN_c, iNode, iNodeValue, iNodeParent, iNodeParentValue)

$N_{ijk\_temp} \leftarrow$ GetCount (iNode, V, iNodeParent, iNodeParentValue, D_t^i)

$N_{ij\_prior} \leftarrow N_{ij\_prior} + N_{ijk\_prior\_temp}$

$N_{ij} \leftarrow N_{ij} + N_{ijk\_temp}$

if (V= iNodeValue) {

$N_{ijk\_prior} \leftarrow N_{ijk\_prior\_temp}$

$N_{ijk} \leftarrow N_{ijk\_temp}$

}

}

Return log $[(N_{ijk\_prior} + N_{ijk}) / (N_{ij\_prior} + N_{ij})]$

END PROCEDURE

PROCEDURE GetCount (iNode, V, iNodeParent, iNodeParentValue, D_t^i)

Return count of iNode=V and iNodeParent = iNodePareentValue in the D_t^i

END PROCEDURE

PROCEDURE GetPriorCount (BN_{sci}, N_i, BN_c, iNode, iNodeValue, iNodeParent, iNodeParentValue)

BN_{sci}: the prior model for source knowledge transfer. It is obtained by removing source-specific nodes from the source model and then adding target-specific nodes.

N_i: the equivalent sample size for iNode

BN_c: a candidate Bayesian network model for scoring

iNode: the node for probability calculation

iNodeValue: the value of the node for probability calculation

iNodeParent: the parent set of the node in BN_c

iNodeParentValue: the value of the parent set

prob $\leftarrow$ P (iNode=iNodeValue, iNodeParent=iNodeParentValue | BN_{sci})

Return $N_i \times$ prob

END PROCEDURE

```
PROCEDURE LogStructureScore (BN_sc, BN_c, N)
        BN_sc: the cleaned source model obtained by removing source-specific nodes from the shared source model
        BN_c: a candidate Bayesian network model for scoring
        N: a vector containing equivalent sample size for each node in BN_sc
        sum ← 0
        for (iNode in BN_sc) {
            A ← the parent set in BN_sc
            B ← the parent set in BN_c (exclude target-specific nodes that do not appear in source model)
            C ← A union B
            D ← A insect B
            E ← C – D
            k ← 1/ (1+Ni)  /*Ni is the corresponding equivalent sample size for iNode*/
            sum ← sum + k^|E|
        }
        // ignore the target-specific node
        Return sum
END PROCEDURE
```

```
PROCEDURE GetEquivalentSampleSize (BN_sc, N_s, D_t, Approach)
        BN_sc: the cleaned source model obtained by removing source-specific nodes from source model
        BN_c: a candidate Bayesian network model for scoring
        D_t: instances of target data; N_t: number of instances in D_t
        Approach: weight strategy /*unadjusted, ratio, KL, features-specific KL*/
        Initiate an empty vector N. Its size is the number of nodes in BN_sc.
        If (Approach=unadjusted) {set each element in N to be N_s}
        Else if (Approach = ratio and N_s ≥ N_t) {set each element in N to be N_t}
        Else if (Approach = ratio and N_s < N_t) {set each element in N to be N_s}
        /*KL is calculated using Dt and BN_sc */
        Else if (Approach = KL) {set each element in N to be N_s × 2^-KL(Dt, BNsc)}
        Else if (Approach = feature-specific KL) {
            For (iNode in BNsc) {
                Ni ←N_s × 2^-KLi(Dt, BNsc, iNode)
                save Ni to N
            }
        }
        Return N
END PROCEDURE
```

PROCEDURE PARAMETERIZATION (BN_{sc}, BN_t, D_t, **N**)

BN_{sc}: the cleaned source model obtained by removing source-specific nodes from source model

BN_t: learned structure of the target Bayesian network

D_t: instances of the target training data

N: a vector containing equivalent sample size for each node in BN_{sc}

$F_G \leftarrow$ a set of shared predictive nodes between BN_s and BN_t

$F_T \leftarrow$ a set of predictive nodes only in BN_t, not in BN_s

/* each cell in CPTs of generalizable nodes is estimated using both weighted source Dirichlet component and counts of target instances. */

for (X_i in F_G) {

$P_s\left(X_i = x_{ik},\ pa(X_i) = \pi_{ij}\right)$: the joint probability in the BN_s where $pa(X_i)$ is the parent set of X_i

N_i: the equivalent sample size for X_i, which was saved in **N**

$$BN_t\left[\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right)\right] = \frac{D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + N_i \times P_s\left(X_j = x_{jk}, pa(X_i) = \pi_{ij}\right) + 1}{D_t(pa(X_i) = \pi_{ij}) + N_i \times P_s\left(pa(X_i) = \pi_{ij}\right) + r_j}$$

}

/* each cell in CPTs of target-specific nodes is estimated using target instances only. */

for (X_i in F_T) {

$$BN_t\left[\left(X_i = x_{ik} \mid pa(X_i) = \pi_{ij}\right)\right] = \frac{D_t(X_j = x_{jk},\ pa(X_i) = \pi_{ij}) + 1}{D_t(pa(X_i) = \pi_{ij}) + r_j}$$

}

Return BN_t

End PROCEDURE

Bibliography

Ajakan, H., P. Germain, H. Larochelle, F. Laviolette and M. Marchand (2014). "Domain-adversarial neural networks." arXiv preprint arXiv:1412.4446.

Arnold, A., R. Nallapati and W. W. Cohen (2007). A comparative study of methods for transductive transfer learning. Seventh IEEE International Conference on Data Mining Workshops IEEE.

Aue, A. and M. Gamon (2005). Customizing sentiment classifiers to new domains: A case study. Proceedings of recent advances in natural language processing

Aytar, Y. and A. Zisserman (2011). Tabula rasa: Model transfer for object category detection. Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE.

Beinlich, I. A., H. J. Suermondt, R. M. Chavez and G. F. Cooper (1989). The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks, Springer.

Chapman, W. W., J. N. Dowling and M. M. Wagner (2005). "Classification of emergency department chief complaints into 7 syndromes: a retrospective analysis of 527,228 patients." Annals of emergency medicine 46(5): 445-455.

Chickering, D. M., D. Geiger and D. Heckerman (1994). Learning Bayesian networks is NP-hard, Citeseer.

Chute, C. G., J. Pathak, G. K. Savova, K. R. Bailey, M. I. Schor, L. A. Hart, C. E. Beebe and S. M. Huff (2011). The SHARPn project on secondary use of Electronic Medical Record data: progress, plans, and possibilities. AMIA Annu Symp Proc.

Ciaramita, M. and O. Chapelle (2010). Adaptive parameters for entity recognition with perceptron HMMs. Workshop on Domain Adaptation for Natural Language Processing Association for Computational Linguistics.

Cooper, G. F., P. Hennings-Yeomans, S. Visweswaran and M. Barmada (2010). An efficient Bayesian method for predicting clinical outcomes from genome-wide data. AMIA Annual Symposium proceedings, American Medical Informatics Association.

Cooper, G. F. and E. Herskovits (1992). "A Bayesian method for the induction of probabilistic networks from data." Machine learning 9(4): 309-347.

Cooper, G. F., R. Villamarin, F.-C. R. Tsui, N. Millett, J. U. Espino and M. M. Wagner (2015). "A method for detecting and characterizing outbreaks of infectious disease from clinical reports." Journal of biomedical informatics 53: 15-26.

Dai, W., G.-R. Xue, Q. Yang and Y. Yu (2007). Transferring naive bayes classifiers for text classification. Proceedings of the national conference on artificial intelligence Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Daly, R., Q. Shen and S. Aitken (2011). "Learning Bayesian networks: approaches and issues." The Knowledge Engineering Review 26(02): 99-157.

DeLong, E. R., D. M. DeLong and D. L. Clarke-Pearson (1988). "Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach." Biometrics: 837-845.

Domingos, P. and M. Pazzani (1997). "On the optimality of the simple Bayesian classifier under zero-one loss." Machine learning 29(2-3): 103-130.

Druzdzel, M. J. (1999). SMILE: Structural Modeling, Inference, and Learning Engine and GeNIe: a development environment for graphical decision-theoretic models. Aaai/Iaai.

Elkin, P. L., D. A. Froehling, D. L. Wahner-Roedler, S. H. Brown and K. R. Bailey (2012). "Comparison of natural language processing biosurveillance methods for identifying influenza from encounter notes." Annals of Internal Medicine 156(1_Part_1): 11-18.

Espino, J. U. and M. M. Wagner (2001). Accuracy of ICD-9-coded chief complaints and diagnoses for the detection of acute respiratory illness. Proceedings of the AMIA Symposium, American Medical Informatics Association.

Finkel, J. R. and C. D. Manning (2009). Hierarchical bayesian domain adaptation. Annual Conference of the North American Chapter of the Association for Computational Linguistics Association for Computational Linguistics.

Friedman, C., L. Shagina, Y. Lussier and G. Hripcsak (2004). "Automated encoding of clinical documents based on natural language processing." J Am Med Inform Assoc 11(5): 392-402.

Gabow, H. N., Z. Galil and T. H. Spencer (1984). Efficient implementation of graph algorithms using contraction. Foundations of Computer Science, 1984. 25th Annual Symposium on, IEEE.

Gerbier-Colomban, S., Q. Gicquel, A.-L. Millet, C. Riou, J. Grando, S. Darmoni, V. Potinet-Pagliaroli and M.-H. Metzger (2013). "Evaluation of syndromic algorithms for detecting patients with potentially transmissible infectious diseases based on computerised emergency-department data." BMC medical informatics and decision making 13(1): 1.

Hall, M. A. (1999). Correlation-based feature selection for machine learning, The University of Waikato.

Harkema, H., J. N. Dowling, T. Thornblade and W. W. Chapman (2009). "ConText: an algorithm for determining negation, experiencer, and temporal status from clinical reports." J Biomed Inform 42(5): 839-851.

Heckerman, D., D. Geiger and D. M. Chickering (1995). "Learning Bayesian networks: The combination of knowledge and statistical data." Machine learning 20(3): 197-243.

Hripcsak, G., J. D. Duke, N. H. Shah, C. G. Reich, V. Huser, M. J. Schuemie, M. A. Suchard, R. W. Park, I. C. K. Wong and P. R. Rijnbeek (2015). "Observational Health Data Sciences and Informatics (OHDSI): opportunities for observational researchers." Studies in health technology and informatics 216: 574.

Huang, J., A. Gretton, K. M. Borgwardt, B. Schölkopf and A. J. Smola (2006). Correcting sample selection bias by unlabeled data. Advances in neural information processing systems

Ivanov, O., P. H. Gesteland and W. Hogan "Detection of pediatric respiratory and gastrointestinal outbreaks from free-text chief complaints. AMIA Annu Symp Proc. 2003: 318–322. Date accessed: 4/12/2017. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1480317/."

Jiang, J. (2008). Domain adaptation in natural language processing, ProQuest.

Jiang, J. and C. Zhai (2007). A two-stage approach to domain adaptation for statistical classifiers. Sixteenth ACM conference on Conference on information and knowledge management ACM.

Jiang, X., B. Cai, D. Xue, X. Lu, G. F. Cooper and R. E. Neapolitan (2014). "A comparative analysis of methods for predicting clinical outcomes using high-dimensional genomic datasets." Journal of the American Medical Informatics Association 21(e2): e312-e319.

Karp, R. M. (1971). "A simple derivation of Edmonds' algorithm for optimum branchings." Networks 1(3): 265-272.

Kent, J. T. (1983). "Information gain and a general measure of correlation." Biometrika 70(1): 163-173.

Kullback, S. and R. A. Leibler (1951). "On information and sufficiency." The annals of mathematical statistics 22(1): 79-86.

Long, M., Y. Cao, J. Wang and M. I. Jordan (2015). "Learning transferable features with deep adaptation networks." arXiv preprint arXiv:1502.02791.

Long, M., H. Zhu, J. Wang and M. I. Jordan (2016). "Deep transfer learning with joint adaptation networks." arXiv preprint arXiv:1605.06636.

López Pineda, A., Y. Ye, S. Visweswaran, G. F. Cooper, M. M. Wagner and F. R. Tsui (2015). "Comparison of machine learning classifiers for influenza detection from emergency department free-text reports." Journal of Biomedical Informatics 58: 60-69.

Lu, J., V. Behbood, P. Hao, H. Zuo, S. Xue and G. Zhang (2015). "Transfer learning using computational intelligence: a survey." Knowledge-Based Systems 80: 14-23.

Luo, Z., Y. Zou, J. Hoffman and L. F. Fei-Fei (2017). Label efficient learning of transferable representations acrosss domains and tasks. Advances in Neural Information Processing Systems.

Mitchell, T. M. (1997). machine learning

Mozafari, A. S. and M. Jamzad (2016). "A SVM-based model-transferring method for heterogeneous domain adaptation." Pattern Recognition 56: 142-158.

Naeini, M. P., G. F. Cooper and M. Hauskrecht (2015). Binary classifier calibration using a Bayesian non-parametric approach. Proceedings of the 2015 SIAM International Conference on Data Mining, SIAM.

Naeini, M. P., G. F. Cooper and M. Hauskrecht (2015). Obtaining Well Calibrated Probabilities Using Bayesian Binning. AAAI.

Ogoe, H. A., S. Visweswaran, X. Lu and V. Gopalakrishnan (2015). "Knowledge transfer via classification rules using functional mapping for integrative modeling of gene expression data." BMC bioinformatics 16(1): 226.

Overhage, J. M., S. Grannis and C. J. McDonald (2008). "A comparison of the completeness and timeliness of automated electronic laboratory reporting and spontaneous reporting of notifiable conditions." American journal of public health 98(2): 344-350.

Pan, S. J., X. Ni, J.-T. Sun, Q. Yang and Z. Chen (2010). Cross-domain sentiment classification via spectral feature alignment. Nineteenth international conference on world wide web ACM.

Pan, S. J., I. W. Tsang, J. T. Kwok and Q. Yang (2011). "Domain adaptation via transfer component analysis." Neural Networks, IEEE Transactions on 22(2): 199-210.

Pan, S. J. and Q. Yang (2010). "A survey on transfer learning." Knowledge and Data Engineering, IEEE Transactions on 22(10): 1345-1359.

Panackal, A. A., N. M. M'ikanatha, F.-C. Tsui, J. McMahon, M. M. Wagner, B. W. Dixon, J. Zubieta, M. Phelan, S. Mirza and J. Morgan (2002). "Automatic electronic laboratory-based reporting of notifiable infectious diseases at a large health system." Emerging infectious diseases 8(7): 685-691.

Rexit, R., F. R. Tsui, J. Espino, P. K. Chrysanthis, S. Wesaratchakit and Y. Ye (2015). "An analytics appliance for identifying (near) optimal over-the-counter medicine products as health indicators for influenza surveillance." Information Systems 48: 151-163.

Robin, X., N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez and M. Müller (2011). "pROC: an open-source package for R and S+ to analyze and compare ROC curves." BMC bioinformatics 12(1): 77.

Roy, D. M. and L. P. Kaelbling (2007). Efficient Bayesian Task-Level Transfer Learning. International joint conference on artifical intelligence

Ruiz, V. M. (2014). The Use of Multiple Emergency Department Reports per Visit for Improving Influenza Case Detection. Master, University of Pittsburgh.

Satpal, S. and S. Sarawagi (2007). Domain adaptation of conditional probability models via feature subsetting. European Conference on Principles of Data Mining and Knowledge Discovery Springer.

Savova, G. K., J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler and C. G. Chute (2010). "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications." J Am Med Inform Assoc 17(5): 507-513.

Selby, J. V., A. C. Beal and L. Frank (2012). "The Patient-Centered Outcomes Research Institute (PCORI) national priorities for research and initial research agenda." Jama 307(15): 1583-1584.

Sugiyama, M., S. Nakajima, H. Kashima, P. V. Buenau and M. Kawanabe (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. Advances in neural information processing systems

Tan, C., F. Sun, T. Kong, W. Zhang, C. Yang and C. Liu (2018). A Survey on Deep Transfer Learning. International Conference on Artificial Neural Networks, Springer.

Tan, S., X. Cheng, Y. Wang and H. Xu (2009). Adapting naive bayes to domain adaptation for sentiment analysis. European Conference on Information Retrieval Springer.

Tsui, F., M. Wagner, G. Cooper, J. Que, H. Harkema, J. Dowling, T. Sriburadej, Q. Li, J. Espino and R. Voorhees (2011). "Probabilistic case detection for disease surveillance using data in electronic medical records." Online journal of public health informatics 3(3).

Tsui, F., Y. Ye, V. Ruiz, G. F. Cooper and M. M. Wagner (2017). "Automated influenza case detection for public health surveillance and clinical diagnosis using dynamic influenza prevalence method." Journal of Public Health: 1-8.

Tzeng, E., J. Hoffman, K. Saenko and T. Darrell (2017). Adversarial discriminative domain adaptation. Computer Vision and Pattern Recognition (CVPR).

Wagner, M., F. Tsui, G. Cooper, J. Espino, H. Harkema, J. Levander, R. Villamarin, R. Voorhees, N. Millett and C. Keane (2011). "Probabilistic, decision-theoretic disease surveillance and control." Online journal of public health informatics 3(3).

Wagner, M. M., J. Espino, F. Tsui, P. Gesteland, W. Chapman, O. Ivanov, A. Moore, W. Wong, J. Dowling and J. Hutman (2004). "Syndrome and outbreak detection using chief-complaint data—experience of the Real-Time Outbreak and Disease Surveillance project." Morbidity and Mortality Weekly Report: 28-31.

Wagner, M. M., F.-C. Tsui, J. Espino, W. Hogan, J. Hutman, J. Hersh, D. Neill, A. Moore, G. Parks and C. Lewis (2004). "National retail data monitor for public health surveillance." Morbidity and Mortality Weekly Report: 40-42.

Weiss, K., T. M. Khoshgoftaar and D. Wang (2016). "A survey of transfer learning." Journal of Big Data 3(1): 1-40.

Wiens, J., J. Guttag and E. Horvitz (2014). "A study in transfer learning: leveraging data from multiple hospitals to enhance hospital-specific predictions." Journal of the American Medical Informatics Association 21(4): 699-706.

Yang, J., R. Yan and A. G. Hauptmann (2007). Adapting SVM classifiers to data with shifted distributions. Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on, IEEE.

Yang, J., R. Yan and A. G. Hauptmann (2007). Cross-domain video concept detection using adaptive svms. Proceedings of the 15th ACM international conference on Multimedia, ACM.

Ye, Y., F. Tsui, M. Wagner, J. U. Espino and Q. Li (2014). "Influenza detection from emergency department reports using natural language processing and Bayesian network classifiers." Journal of the American Medical Informatics Association 21(5): 815-823.

Ye, Y., M. M. Wagner, G. F. Cooper, J. P. Ferraro, H. Su, P. H. Gesteland, P. J. Haug, N. E. Millett, J. M. Aronis and A. J. Nowalk (2017). "A study of the transferability of influenza case detection systems between two large healthcare systems." PloS one 12(4): e0174970.

Yosinski, J., J. Clune, Y. Bengio and H. Lipson (2014). How transferable are features in deep neural networks? Advances in neural information processing systems.

Zadrozny, B. and C. Elkan (2001). Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. ICML, Citeseer.