Data integration in machine learning

Li, Yifeng; Ngom, Alioune

Publisher's version / Version de l'éditeur:
https://doi.org/10.1109/BIBM.2015.7359925
2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1665-1671, 2015-12-17

NRC Publications Record / Notice d'Archives des publications de CNRC:
https://nrc-publications.canada.ca/eng/view/object/?id=b505f592-59d6-41fb-ad11-24604539569f
https://publications-cnrc.canada.ca/fra/voir/objet/?id=b505f592-59d6-41fb-ad11-24604539569f

Data Integration in Machine Learning

Yifeng Li
Information and Communications Technologies
National Research Council of Canada
Ottawa, Ontario, Canada
Email: yifeng.li@nrc-cnrc.gc.ca

Alioune Ngom
School of Computer Science
University of Windsor
Windsor, Ontario, Canada
Email: angom@uwindsor.ca

***Abstract*—Modern data generated in many fields are in a strong need of integrative machine learning models in order to better make use of heterogeneous information in decision making and knowledge discovery. How data from multiple sources are incorporated in a learning system is key step for a successful analysis. In this paper, we provide a comprehensive review on data integration techniques from a machine learning perspective.**

***Keywords*-data integration, Bayesian network, decision tree, random forest, multiple kernel learning, feature extraction, deep learning.**

I. Introduction

In the big data era, data explosively grow in both volume and variety [1]. For example in biomedical research, it is not technically difficult to access a patient's traditional records (such as age, sex, weight, height, blood pressure, family history, imaging materials, and diagnostic symptoms) and high-throughput genotype data (such gene expression, non-coding RNA transcription, DNA methylation, and whole-genome sequence) under proper multi-party consents. On the one hand, this provides us an unprecedented opportunity to understand a complex object (or system) from multiple angles and make precise data-driven decisions. On the other hand, it poses a challenge for machine learning experts and data scientists to wisely optimize the use of these data.

While it becomes impossible to use a human-based decision making procedure, intelligent learning systems play crucial roles in the deluge of big data. When data from multiple sources are incorporated into a learning process, data fusion techniques can be classified as either early, intermediate, or late integrations [2]. In early integration methods, all features are concatenated into a vector before fitting an unsupervised or supervised model. In late integration, separate models are first learned using their corresponding feature subsets, then their outputs are further combined to make the final determination. An intermediate strategy globally involves data integration in a learning process.

Data integration has been studied in related areas such as multisensor signal processing [3] and bioinformatics [4]. In the frontier of big data studies, it becomes indispensable to investigate the fundamental principles of integrating data from multiple sources. In this paper, we focus on fundamental integrative machine learning principles, particularly for classification. In the following sections, we shall discuss simple feature concatenating, Bayesian methods, Bayesian networks, tree-based ensemble methods, multiple kernel learning, and deep neural networks. Hereafter, we use the phrase "data from multiple sources" to denote any kinds of heterogeneous data that provide complementary information to characterize an object from various aspects. Data from multiple sources may have different data types, follow very different statistical distributions, possess different semantics, and suffer different levels of uncertainties.

II. Features Concatenation

Concatenating features in a vector seems to be the "simplest" principle. It requires additional downstream processing. Since data from multiple sources have different types that may include continuous features, discrete features, characters, even graphics. Converting these features into acceptable types (e.g. continuous to discrete, categorical to 0-1 coding) is inevitable for certain classifiers. Moreover, data from multiple sources are usually in different scales. Particularly for discriminative models, normalization or standardization may be necessary in order to speed up training and improve performance. This integrative strategy is commonly used in linear models such as support vector machines (SVMs) [5] and LASSO-based predictors [6]. Feature concatenation is the easiest (perhaps the oldest) strategy that glues all features together. However, this art often does not work well with modern data, which are either high dimensional or structural. Coarsely stitching all features together would lead to severe loss of key structural information. For instance, converting a text document into a bag of words will certainly ignore semantics that is obviously very important. Moreover, downstream processing, such as normalization and transformation, after feature concatenating is often panic, even impossible.

III. Bayesian Methods

From a Bayesian perspective, we can consider treating some features as prior knowledge. Suppose we separate the input features into prior features and regular features. The

978-1-4673-6799-8/15/$31.00 ©2015 IEEE

corresponding training data can thus be split as $\boldsymbol{X} = [\boldsymbol{P}; \boldsymbol{R}]$ accordingly. Suppose we deal with a two-class problem, that is $y_n \in \{-1, +1\}$. For a sample $\boldsymbol{x}_n = [\boldsymbol{p}_n; \boldsymbol{r}_n]$, its class prior can be defined as a logistic function:

$$p(y_n = +1|\boldsymbol{p}_n, \boldsymbol{\beta}) = \frac{e^{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{p}_n}}{1 + e^{\boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{p}_n}}, \tag{1}$$

where $\boldsymbol{\beta}$ are the coefficients of the corresponding prior features, indicating their contributions to the class prior. Using the Bayesian theorem, the posterior can be written as

$$p(y = +1|\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{\theta}) = \frac{p(\boldsymbol{x}|y = +1, \boldsymbol{\theta})p(y|\boldsymbol{p}, \boldsymbol{\theta})}{p(\boldsymbol{x}|\boldsymbol{\theta})} \tag{2}$$

$$\propto p(\boldsymbol{x}|y = +1, \boldsymbol{\alpha}_{+1})p(y = +1|\boldsymbol{p}, \boldsymbol{\beta}) \tag{3}$$

where $\boldsymbol{\alpha}_{+1}$ is the parameter of the +1 class-conditional distribution (likewise, $\boldsymbol{\alpha}_{-1}$ is the parameter of the -1 class-conditional distribution). The exact form of the class-conditional distribution $p(\boldsymbol{x}|y, \boldsymbol{\alpha}_{+1})$ is formulated by certain models (such as mixture of Gaussians and naïve Bayes). The model parameter $\boldsymbol{\theta} = \{\boldsymbol{\alpha}_{+1}, \boldsymbol{\alpha}_{-1}, \boldsymbol{\beta}\}$ can be learned from training data $\{\boldsymbol{X}, \boldsymbol{y}\}$ by maximum likelihood or maximum *a posteriori* estimation. For instance, CENTIPEDE is an Bayesian mixture model developed for the prediction of transcription factor binding sites [7] by using position weight matrices score, evolutionary conservation score, and transcription start site proximity as prior knowledge, and ChIP-seq as regular features. In case of multi-class problems (that is $y_n \in \{1, \cdots, C\}$), the class prior can be extended as a multinoulli distribution:

$$h_\theta(\boldsymbol{p}, \boldsymbol{\theta}) = \begin{bmatrix} p(y = 1|\boldsymbol{p}, \boldsymbol{\theta}) \\ \vdots \\ p(y = C|\boldsymbol{p}, \boldsymbol{\theta}) \end{bmatrix} = \frac{1}{\sum_{c=1}^{C} e^{-\boldsymbol{\beta}_c^{\mathrm{T}}\boldsymbol{p}}} \begin{bmatrix} e^{-\boldsymbol{\beta}_1^{\mathrm{T}}\boldsymbol{p}} \\ \vdots \\ e^{-\boldsymbol{\beta}_C^{\mathrm{T}}\boldsymbol{p}} \end{bmatrix}. \tag{4}$$

Then the parameter to be learned from training data becomes $\{\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_C, \boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_C\}$. Bayesian methods are well-known with their capability of incorporating various prior knowledge. However, it may be difficult to find informative information as prior features. Furthermore, it is often hard to assume proper class-conditional distributions, especially for complex systems. In case of many-class problems, finding a suitable class-conditional distribution for each individual class becomes unattainable in practice.

IV. Bayesian Networks

Bayesian networks (BNs) [8] can learn on data of multiple sources. As a typical model of probabilistic graphical models (PGMs), a BN can be represented by $\{\boldsymbol{S}, \boldsymbol{\theta}\}$, where $\boldsymbol{S}$ denotes its graphical structure whose nodes represent variables and directed edges represent dependencies between pairs of variables, and $\boldsymbol{\theta}$ the parameters of the variables' conditional distributions. Suppose there are M visible variables (say $\boldsymbol{X} = [X_1, \cdots, X_M]$) and no latent variable, BN decomposes the probability of $\boldsymbol{X}$ to

$$p(\boldsymbol{X}) = \prod_{i}^{M} p(X_i|\Pi(X_i)) \tag{5}$$

where $\Pi(X_i)$ are the parents of X_i. The dependency structure and parameter of the conditional distributions can be learned from data. Given a learned BN, the values of invisible variables can be inferred from partially observed data. The variables in a BN can be discrete [9], continuous [10], or a mixture of them [11]. Thus, variables of different types can be naturally integrated using BN. In the general procedure of applying BN for classification (and regression) purpose (Figure 1a), the network (with the class variable being a node) can be learned from labelled training data. The class of unlabelled samples can be inferred using the learned network [12]. Furthermore, BN can be applied to feature selection by only taking variables in the Markov blanket of the class node. For example in Figure 1a, features X_1, X_2, X_4, X_5, X_6 form the Markov blanket of the class node, thus they can be reported as key features. However, three obstacles challenge us to apply BNs in data integration. First, searching the optimal BN structure is a NP-complete problem [9]. Second, the number of parameters may be much larger than the sample size. Third, inference in a BN is intractable. This obstructs us from using BN for high-dimensional data. Thus, heuristic structure learning algorithms and restrictions (or assumptions) on the model structure and conditional distributions are usually made for high-dimensional data. The model structure is often assumed to be sparse. For example in Naïve Bayes classifier [13] (Figure 1b), the features are assumed to be conditionally independent of each other given the class variable (say C) as common parent, so that the joint probability $p(C, \boldsymbol{X})$ can be factorized as

$$p(C, \boldsymbol{X}) = p(C) \prod_{i=1}^{M} p(X_i|C). \tag{6}$$

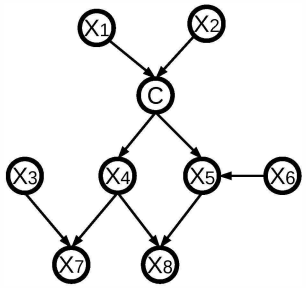
The class label can thus be inferred by

$$p(C|\boldsymbol{X}) = \frac{p(C, \boldsymbol{X})}{p(\boldsymbol{X})} = \frac{p(C) \prod_{i=1}^{M} p(X_i|C)}{p(\boldsymbol{X})}. \tag{7}$$

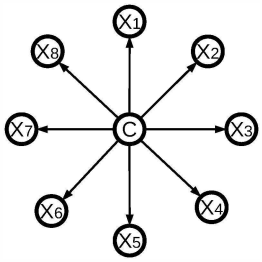
Naïve Bayes classifier is a slim and swift model because there is no need to learn the structure and the inference of class label is straightforward. Tree-augmented Naïve Bayes classifier [14] (Figure 1c) relaxes the independence among features by a tree structure. It outperforms the Naïve Bayes classifier but keeps the efficiency of model learning.

V. Decision Trees and Ensemble Learning

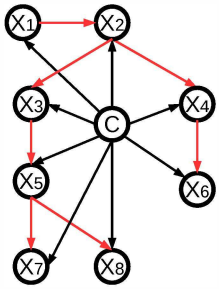
A mixture of discrete and continuous features can be simultaneously fed into decision trees [15]. There is no need to normalize the features. Thus, decision trees should



(a) General Bayesian network classifier. The class variable are treated as an ordinary node.



(b) Naïve Bayes classifier. Features are assumed to be conditional independent given the class variable as their common parent.



(c) Tree-augmented Naïve Bayes classifier. Sharing the class variable as parent, the features have a tree structure (in red).

Figure 1: Bayesian network classifiers.

be considered as integrative models. Decision trees for classification or regression are representatives of rule-based learning. When recursively constructing the tree structure, a feature (even feature subset) that splits the classes the best is selected to create a node. At each node, rules are established to branch out different classes downstream. Different from black-box models, the learned hierarchy of rules (tree) are interpretable. Decision trees can be applied to select features by ranking the features with respect to their summed improvements in class purity. In a decision tree with T internal nodes, the importance score of the i-th feature can be defined by $s(X_i) = \sum_{t=1}^{T} g(t) I(v(t) = i)$, where $I(v(t) = i) \in \{0, 1\}$ indicates whether the i-th feature is selected in the t-th node to split the corresponding region, and $g(t)$ is the gain of class purity measured, for example, by Gini index [15], [16]. Since each feature is used to learn decision rules, various data types (discrete, categorical, and continuous) are acceptable. The values of continuous variables are partitioned into intervals of different lengths, thus decision rules can be created for continuous variables of a variety of distributions. There is no need to standardize the input data. In fact, decision trees are invariant under feature scaling and transformation. However, decision trees are notorious with their high risk of overfitting, thus pruning is a necessary remedy. Moreover, building a decision tree for high-dimensional data is very time-consuming.

As a successful example of collective intelligence, ensemble learning [17], [18] builds a population of weak learners for the state-of-the-art performance. Bagging [19] and Boosting [20], [21] are popular ensemble learning models where decision trees are often used as weak learners. While bagging simply combining the decisions of multiple weak learners, boosting tweaks the weak learners to focus on hard examples.

Different from bagging, random forest quickly constructs decision trees by randomizing the picking of features at each node. When growing a tree in random forest, a single feature or a subset of features are randomly chosen to create a branching node. Feature importance can be ranked by out-of-bag (OOB) randomization or Gini index. In the former method, the importance score of the i-th feature is defined as the difference of OOB errors between using the original OOB samples and using the OOB samples where the values of the i-th feature are permuted. In the latter method, Gini indices [15] of the i-th feature in individual trees in the forest can be averaged as the importance score [16].

There are three ways to integrate data by ensemble learning. The first way is to use the concatenated features as input of random forest. The second way is to build multiple trees for each data source, and then use all trees of all data sources to vote for the final decision [22], [23] An example of using random forest is illustrated in Figure 2. More elegant combination methods are discussed in [24]. One of the advantages of this ensemble-learning based data integration is its good manipulability and interpretability. Class imbalance problems can be elegantly addressed by random forest in its bootstrapping [25]. Also, granularity of features can be carefully considered in the step of sampling features [26]. However, since it is a late-integration principle, the interactions of features from separate sources cannot be detected. In the third way, meta-features learned from different sources can be used to grow trees. This idea is from West's group who incorporate both clinical factors and genomic data in predictive survival assessments [27]. A meta-feature (named meta-gene in [27]) is defined as the first principal component of a cluster of genes grouped by k-mean clustering. Then, the model grows a forest of statistical classification and prediction trees. In each tree, features used in the nodes are decided by the significances of Bayesian factor tests on the features (meta-genes and clinical factors). Multiple significant features can be distributed in multiple trees so that the correlations between trees are reduced. The final decision is determined by a weighted combination of the decisions of all trees, where the probabilities of trees are used as weights. One advantage of the meta-feature based ensemble model is that the information from different

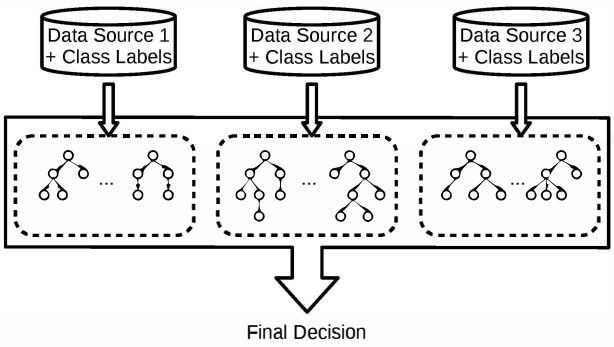


Figure 2: Ensemble learning (e.g. random forest) integrates data from multiple sources.

sources can be incorporated in the model learning. Since the meta-features are used instead of the original features, complexity of trees can thus be reduced.

VI. Kernel Learning and Metric Learning

Kernel matrix $k(\boldsymbol{X}, \boldsymbol{X})$, rather than the original input features, is only required as input of kernel methods. Thus, the problem of data integration is transformed to kernel integration. *Multiple kernel learning* (MKL) [28] is an intermediate integration technique that first computes kernel (or similarity) matrices separately for each of the multiple data sources, then combines these matrices to generate the kernel matrix to be used in a kernel model. Suppose there are K kernel functions, $\{k_1(\cdot,\cdot), \cdots, k_K(\cdot,\cdot)\}$, representing K sources of data, $\boldsymbol{x} = \{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(K)}\}$. The combined similarity between samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be computed by

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_{\boldsymbol{\eta}}\big(k_1(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_j^{(1)}), \cdots, k_K(\boldsymbol{x}_i^{(K)}, \boldsymbol{x}_j^{(K)})\big), \quad (8)$$

where $f_{\boldsymbol{\eta}}$ is either a linear or non-linear function. In the simplest case, it is a weighted linear combination of kernel matrices:

$$k(\boldsymbol{X}, \boldsymbol{X}) = \sum_{i=1}^{K} \lambda_i k(\boldsymbol{X}_i, \boldsymbol{X}_i), \quad (9)$$

where $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_K]$ can be either assigned prior to learning, or determined in the procedure of learning. The parameter, $\boldsymbol{\eta}$, can either be predefined, partially, or completely learned during training. Figure 3 shows an example of MKL-based integration system. The individual similarity (kernel) matrix of a data source can be computed by an off-the-shelf kernel function semantically sensible for the specific data source or by sophisticated metric learning [29]. Metric learning [30], [31], [32] aims to learn a metric function from data such that the distances between within-class samples are closer, and the distances between inter-class samples are farther. The key strength of kernel methods is that their optimizations are independent of the number of features, which is known as dimension-free [33]. However, large-scale optimization corresponding to a large sample size remains the main bottleneck. For example, an optimal MKL learning may be essentially a semidefinite programming problem [34].

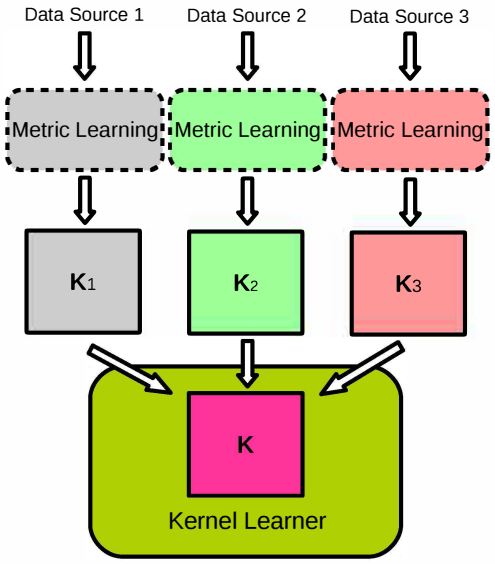


Figure 3: Multiple kernel learning. Metric learning may be applied to learn suitable similarity matrices.

VII. Feature Extraction

While it is often challenging to combine features in the original input space, the new features extracted by feature extraction methods can be easily combined. Illustrated in Figure 4, the idea is to extract new features from each data source first, and then combine these new features together. Finally, a classifier can be applied on the combined features. Depending on the nature of an individual data source, a feature extraction method learns the representations of samples in a new feature space. Matrix factorization methods, such as principal component analysis (PCA) [35], [36], factor analysis (FA) [37], [38], non-negative matrix factorization (NMF) [39], [40], [41], sparse representation (SR) [42], and tensor decomposition methods [43], [44], are commonly used feature extraction models. There are several benefits of using feature extraction in data integration. First, the natures of heterogeneous data from multiple sources can be separately well considered. In spite of the original data types, the new features in the corresponding feature spaces are usually numeric. Second, the high-dimensionality is dramatically reduced so that the downstream analysis will be more efficient. Third, extracting features separately for each data source implements the principle of divide-and-conquer, thus computational complexity can be significantly reduced. Forth, relational data can be well incorporated by kernel feature extraction methods [45]. However, one pitfall

of the feature-extraction based integrative principle is that the interactions (correlation) between features from different sources cannot be considered in the feature extraction procedures.

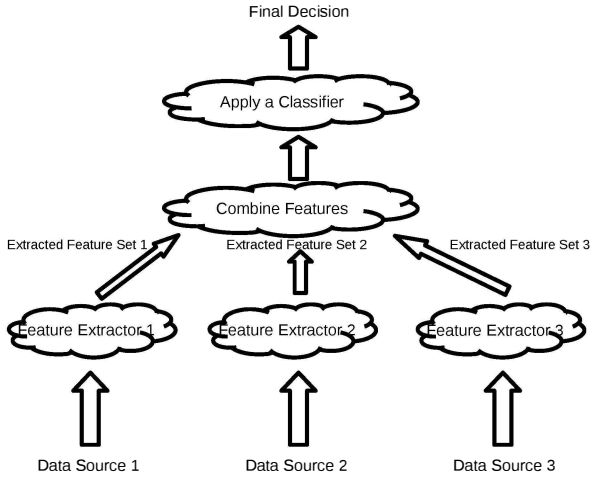


Figure 4: Data integration procedure based on feature extraction.

In order to consider the interactions between features from different sources, Bayesian matrix factorization methods can be applied to extract new features on the feature-wise concatenated matrix from multiple sources. A key to realize it is to assume separate distributions with different variations for features in different scales. By inducing group-wise (that is source-wise) sparsity on the basis matrix (that is factor loading matrix) as in Bayesian group factor analysis (BGFA) [46] (see Figure 5), ubiquitous and source-specific factors can be detected, which is beneficial to the understanding of the data. Bayesian canonical correlation analysis (BCCA) [47] is a special case of BGFA when there are only two data sources.

VIII. Deep Neural Networks and Multi-Modal Learning

While the feature-extraction based integration principle, illustrated in Figure 4, is incapable of learning the interactions between features from different sources, the deep neural network [48] based multi-modal structure, illustrated in Figure 6, integrates the output of individual sub-networks in higher layers. The sub-networks provide the flexibility of choosing appropriate deep learning models respectively for individual data sources, such as deep belief net [49] for binary data, convolutional network [50] for image data, recurrent neural network [51] for speech signal, and deep feature selection [52] for choosing discriminative features. The sub-networks can be either directed or undirected. The whole model can be supervised or unsupervised. [53] is an example of multi-modal learning. When learning the model

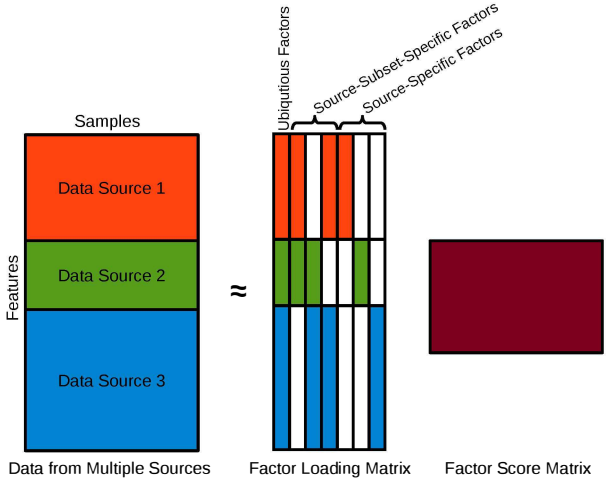


Figure 5: Data integration based on Bayesian group factor analysis. Zero blocks are marked in white.

parameter, the sub-networks can be pretrained on different data sources, then the parameter of the whole integrative network (including the integrative network and sub-networks) can be globally fine-tuned. This component-wise learning can significantly reduce the cost of computation.

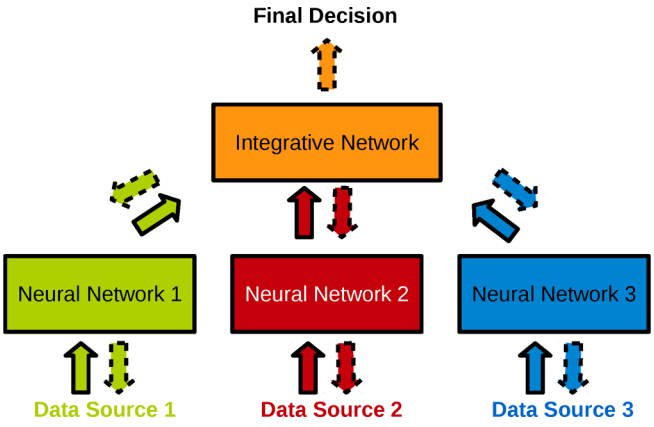


Figure 6: Deep neural network for multi-modal learning. Different deep learning models can be applied to the individual data sources. The integrative network combines the information from the sub-networks. The model can be either directed or undirected; either supervised or unsupervised.

IX. Conclusion

Recent development in many areas, such as image processing, computer vision, bioinformatics, social network mining, and finance, have a keen need for integrative machine learning models to incorporate data from multiple data. In this review, we investigate a variety of data integration

principles from a machine learning perspective. Their basic ideas, structures, strengths, and limitations are discussed.

ACKNOWLEDGMENT

We thank Drs Youlian Pan (NRC) and Raymond Ng (UBC) for providing valuable comments in the improvement of this paper.

REFERENCES

[1] Z. Zhou, N. Chawla, Y. Jin, and G. Williams, "Big data opportunities and challenges: Discussions from data analytics perspectives," *IEEE Computational Intelligence Magazine*, vol. 9, no. 4, pp. 62–74, 2014.

[2] W. Nobel, "Support vector machine applications in computational biology," in *Kernel Methods in Computational Biology*, B. Scholkopf, K. Tsuda, and J.-P. Vert, Eds. The MIT Press, 2004, ch. 3, pp. 71–92.

[3] M. L. II, D. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion: Theory and Practice*, 2nd ed. Boca Raton, Floria: CRC Press, 2008.

[4] M. Ritchie, E. Holzinger, R. Li, S. Pendergrass, and D. Kim, "Methods of integrating data to uncover genotype-phenotype interactions," *Nature Review Genetics*, vol. 15, pp. 85–97, 2015.

[5] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[6] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[7] R. Pique-Regi, J. Degner, A. Pai, D. Gaffney, Y. Gilad, and J. Pritchard, "Accurate inference of transcription factor binding from DNA sequence and chromatin accessibility data," *Genome Research*, vol. 21, pp. 447–455, 2011.

[8] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[9] D. Chickering, "Learning Bayesian networks is NP-complete," in *Learning from Data: AI and Statistics V*, ser. Lecture Notes in Statistics, D. Frisher and H.-J. Lenz, Eds. Springer, 1996, ch. 12, pp. 121–130.

[10] G. Elidan, I. Nachman, and N. Friedman, ""Ideal Parent" structure learning for continuous variable Bayesian networks," *Journal of Machine Learning Research*, vol. 8, pp. 1799–1833, 2007.

[11] S. Davies and A. Moore, "Mix-nets: Factored mixtures of Gaussians in Bayesian networks with mixed continuous and discrete variables," in *Proceedings of The Sixteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers Inc, 2000, pp. 168–175.

[12] J. Cheng and R. Greiner, "Comparing Bayesian network classifiers," in *Proceedings of The Fifteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA: Morgan Kaufmann Publishers Inc, 1999, pp. 101–108.

[13] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of AAAI*, 1992, pp. 223–228.

[14] N. Friedman, D. Geiger, and M. Goldszmith, "Bayesian network classifiers," *Machine Learning*, vol. 29, pp. 103–130, 1997.

[15] L. Breiman, J. Friedman, C. Stone, and R. A. Olshen, *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[16] J. Friedman, R. Tibshirani, and T. Hastie, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. Springer, 2009.

[17] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[18] M. Sewell, "Ensemble learning," 2011.

[19] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 3, pp. 123–140, 1996.

[20] M. Kearns, "Thoughts on hypothesis boosting," 1988.

[21] L. Breiman, "Arcing classifiers," *The Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[22] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[23] ——, "Bootstrap inspired techniques in computational intelligence: Ensemble of classifiers, incremental learning, data fusion and missing features," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 59–72, 2007.

[24] M. Wozniak, M. Grana, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[25] C. Chen, A. Liaw, and L. Breiman, "Using Random Forest to Learn Imbalanced Data," Department of Statistics, University of California, Berkeley, Tech. Rep., 2004.

[26] D. Popovic, A. Sifrim, J. Davis, Y. Moreau, and B. D. Moor, "Problems with the nested granularity of feature domains in bioinformatics: The eXtasy case," *BMC Bioinformatics*, vol. 16, no. S-4, p. S2, 2015.

[27] J. Pittman, E. Huang, H. Dressman, C.-F. Horng, S. Cheng, M.-H. Tsou, C.-M. Chen, A. Bild, E. Iversen, A. Huang, J. Nevins, and M. West, "Integrated modeling of clinical and gene expression information for personalized prediction of disease outcomes," *PNAS*, vol. 101, no. 22, pp. 8431–8436, 2004.

[28] M. Gonen and E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, July 2011.

[29] J. Wang, H. T. Do, A. Woznica, and A. Kalousis, "Metric learning with multiple kernels," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., 2011, pp. 1170–1178.

[30] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2003, pp. 521–528.

[31] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *CoRR*, vol. abs/1306.6709, 2013.

[32] B. Kulis, "Metric learning: A survey," *Foundations and Trends in Machine Learning*, vol. 5, no. 4, pp. 287–364, 2012.

[33] Y. Li, R. Caron, and A. Ngom, "A decomposition method for large-scale sparse coding in representation learning," in *World Congress on Computational Intelligence (IJCNN/WCCI)*. IEEE, July 2014, pp. 3732–2738.

[34] S.-J. Kim, A. Magnani, and S. Boyd, "Optimal kernel selection in kernel fisher discriminant analysis," in *International Conference on Machine Learning*, 2006, pp. 465–472.

[35] I. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[36] M. Wall, A. Rechtsteiner, and L. Rocha, "Singular value decomposition and principal component analysis," in *A Practical Approach to Microarray Data Analysis*, D. Berrar, W. Dubitzky, and M. Granzow, Eds. Norwell, MA: Kluwer, 2003, pp. 91–109.

[37] D. Lawley, "The estimation of factor loadings by the method of maximum likelihood," *Proceedings of the Royal Society of Edinburgh*, vol. 60, pp. 64–82, 1940.

[38] M. West, "Bayesian factor regression models in the "large p, small n" paradigm," *Bayesian Statistics*, vol. 7, pp. 723–732, 2003.

[39] D. D. Lee and S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[40] E. Fertig, J. Ding, A. Favorov, G. Parmigiani, and M. Ochs, "CoGAPS: an R/C++ package to identify patterns and biological process activity in transcriptomic data," *Bioinformatics*, vol. 26, no. 21, pp. 2792–2793, 2010.

[41] Y. Li and A. Ngom, "The non-negative matrix factorization toolbox for biological data mining," *BMC Source Code for Biology and Medicine*, vol. 8, no. 1, p. 10, 2013, https://sites.google.com/site/nmftool.

[42] Y. Li, C. Chen, A. Kaye, and W. Wasserman, "The identification of *cis*-regulatory elements: A review from a machine learning perspective," *BioSystems*, 2015, under revision.

[43] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[44] Y. Li and A. Ngom, "Non-negative matrix and tensor factorization based classification of clinical microarray gene expression data," in *IEEE International Conference on Bioinformatics & Biomedicine*, IEEE. Piscataway, NJ: IEEE Press, Dec. 2010, pp. 438–443.

[45] ——, "Versatile sparse matrix factorization: Theory and applications," *Neurocomputing*, vol. 145, pp. 23–29, 2014.

[46] S. Virtanen, A. Klami, S. Khan, and S. Kaski, "Bayesian group factor analysis," in *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics*, 2012, pp. 1269–1277.

[47] A. Klami, S. Virtanen, and S. Kaski, "Bayesian cononical correlation analysis," *Journal of Machine Learning Research*, vol. 14, pp. 965–1003, 2013.

[48] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[49] G. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[50] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA: MIT Press, 1995, pp. 255–258.

[51] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *IEEE International Conference on Machine Learning (ICML)*, 2014, pp. 1764–1772.

[52] Y. Li, C. Chen, and W. Wasserman, "Deep feature selection: Theory and application to identify enhancers and promoters," *Journal of Computational Biology*, 2015, accepted (RECOMB 2015 Special Issue).

[53] N. Srivastava and R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," *Journal of Machine Learning Research*, vol. 15, pp. 2949–2980, 2014.