Proceedings of the 6th INFORMS Workshop on Data Mining and Health Informatics (DM-HI 2011)
P. Qian, Y. Zhou, C. Rudin, eds.

A latent feature factor graph for cancer diagnosis using microarray gene expression data

Mohammad Khoshneshin

Department of Management Sciences
University of Iowa
Iowa City, IA
mohammad-khoshneshin@uiowa.edu

W. Nick Street

Department of Management Sciences
University of Iowa
Iowa City, IA

Abstract

In this paper, we propose a latent feature model — Latent feature factor graph (LFFG) — for the gene expression microarray classification problem. LFFG is a factor graph in which each entity is represented via a latent feature variable — a node in the factor graph. Then the relationship between entities is captured via a function of latent features — a factor in the factor graph. We use maximum a posteriori estimation to learn the latent features for the gene expression microarray classification problem. The experiments over a small dataset are promising as LFFG outperforms SVM — the state-of-the-art classification method in microarray classification.

Keywords: microarray classification, factor graph, latent variables

Introduction

Because of the availability of huge gene expression microarray datasets, many machine learning and pattern recognition algorithms have emerged to analyze them [12]. One of the important applications of gene expression microarray data is in cancer diagnosis [11]. In this paper, we propose a new classification approach — the latent feature factor graph — which works well in microarray classification.

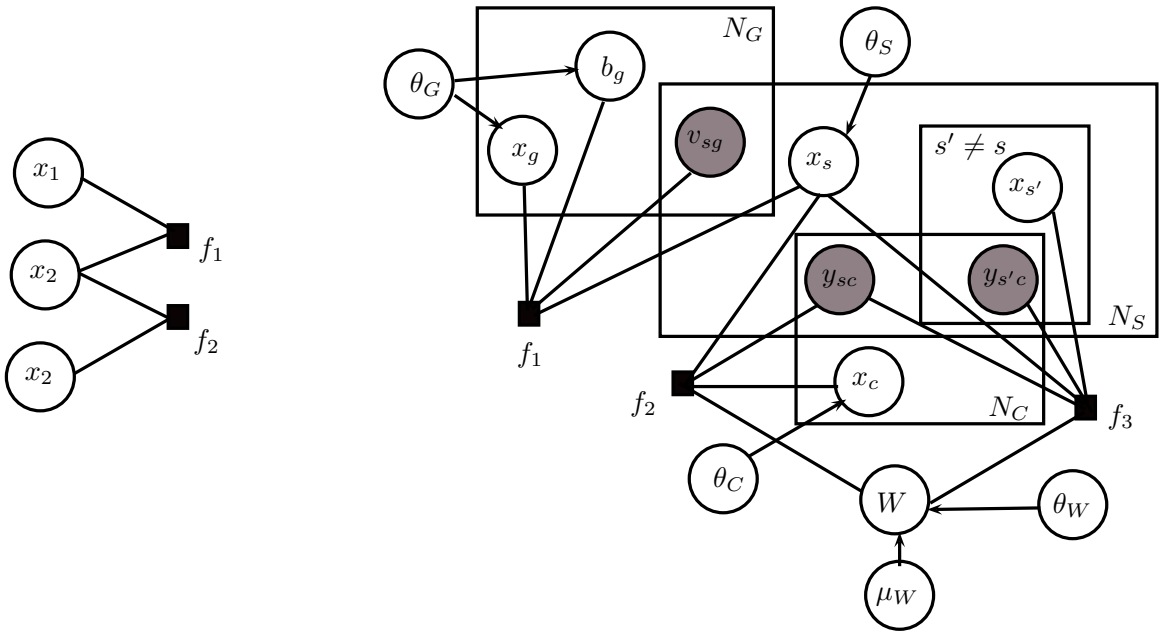
The latent feature factor graph is a graphical model. Graphical models [9] are used to represent complex probability distributions via a graph. Nodes denote random variables and edges denote dependencies between random variables. There are three different types of graphical models. Bayesian networks are directed acyclic graphs. Markov networks are undirected graphs. Factor graphs [8] are bipartite graphs with two different types of nodes. The first type of nodes are random variables and the second type of nodes represent factors which are a function of random variables. In this paper, we use factor graphs due to their expressive power [4].

Figure 1-(a) represents a typical factor graph with 3 random variables (x_1, x_2 and x_3) and two factors (f_1 and f_2). Given a factor graph, the probability distribution over random variables factors based on a potential function ϕ_i for each factor f_i:

$$P(x_1, x_2, ..., x_n) = \frac{1}{Z} \prod_i \phi_i(S_i), \tag{1}$$

where S_i denotes the set of random variables connected to factor i and $Z = \sum_x \prod_i \phi_i(S_i)$ is the partition function. The only condition is $\phi_i \geq 0$ for all factors. Given 1, the joint probability distribution for the factor graph in Figure 1-(a) is:

$$P(x_1, x_2, x_3) = \frac{1}{Z} \phi_1(x_1, x2) \phi_2(x_2, x_3).$$



(a) A factor graph

(b) The latent feature factor graph

Figure 1: The graphical model of (a) a typical factor graph and (b) the latent feature factor graph. In (b), observed variables are shown in gray while latent variables are white. Note that y_{sc} and $y_{s'c}$ are observed only in the training set.

In this paper, we propose a factor graph model — latent feature factor graphs — to learn the relationship between entities. In LFFG, we try to discover the latent features of entities in a low-dimensional space via random variable or nodes in the factor graph. That is, per each entity, a random variable is defined which is latent variable. The relationship between entities is captured via a function of the latent features — representative random variables — which is a factor in the factor graph. The LFFG graphical model for the gene expression microarray classification problem, is shown in Figure 1-(b) where there are 3 types of entities: genes, samples and cancer categories. x_s, x_g and x_c are the random variables representing latent features of samples, genes and categories. Two types of relationships are of interest: expression level (shown by v_{sg}) and samples' cancer category (shown by y_{sc}). The expression level of a gene for a sample will be a function of the latent features of the gene and the sample (factor f_1). Similarly, the cancer category relationship is a function of the latent features of the sample and cancer categories (factor f_2). To boost classification power, we force similarity between the latent features of two samples with the same cancer category (factor f_3).

We use maximum a posteriori probability (MAP) estimation to learn the latent features for the gene expression microarray classification problem. The experiments over a small dataset are promising as LFFG outperforms support vector machine (SVM) — the state-of-the-art classification method in microarray classification [11]. We conjecture that LFFG is successful because it behaves as a supervised dimensionality reduction method. It embeds entities in a low-dimensional dot product space where the samples from the same class are more similar to each other. In comparison to SVM that uses kernel trick to transfer original space to a dot product space, LFFG constructs the dot product space from scratch using both features and labels.

The proposed latent feature factor graph is similar to collective matrix factorization [10] in some aspects as we use weighted linear functions for factors. However, collective matrix factorization is capable of predicting values from several matrices while the goal of our model is classification. Collective matrix factorization assumes a generative model from latent features to relationships — a directed graph — which is limiting, while LFFG assumes correlation between latent feature variables — which can be directed or undirected given the definition of potentials. Therefore,

applying collective matrix factorization to gene expression microarray classification is not straightforward. However, non-negative matrix factorization has been used for clustering microarray data [2] where the learning is only based on the microarray data — an unsupervised approach. While this approach is worthwhile since it does not need labels for training, it is not as powerful as supervised learning approaches.

Latent feature factor graph

In the latent feature factor graph, we introduce one random variable per entity. In classification using microarray gene expression data, there are 3 groups of entities: x_s — a $(1 \times d)$ vector — denotes the latent feature for sample s, x_g — a $(1 \times d)$ vector — denotes the latent feature for gene g, b_g — a scalar — denotes the bias in expression for gene g, and x_c — a $(1 \times d)$ vector — denotes the latent feature for category c. Generally, latent features are vectors with the same size. However, it is possible to assume different sizes for the entities of different types. As a result, the relationship between entities can be captured as a function of entities' latent feature variables — a factor. Here, two types of relationships exist. The expression level of each gene g for each sample s, denoted by v_{sg}, and the category c of each sample s, denoted by y_{sc} which is a binary variable and is one if sample s is in category c and zero otherwise. For the expression level relationship, we define the factor as

$$\phi_1(x_g, x_s, v_{sg}) = \exp\left(-\tau \frac{(x_g x_s^T + b_g - v_{sg})^2}{2}\right). \tag{2}$$

Such a factor implies that the distribution over v_{sg} is Gaussian with mean $x_g x_s^T + b_g$ and precision (one over variance) τ. For the category relationship, we define the factor

$$\phi_2(x_s, x_c, y_{sc}) = \exp(x_s W x_c^T y_{sc}), \tag{3}$$

where W — a $(d \times d)$ matrix — is a weight matrix to transfer x_s from the microarray gene expression data space to the classification space. Note that (3) resembles the numerator of the multinomial logistic regression probability. However, they are not equivalent due to our third factor definition. In the third factor, the relationship between categories and latent features of two different samples is captured with the factor

$$\phi_3(x_s, x_{s'}, y_{sc}, y_{s'c}) = \exp\left(\frac{x_s W W^T x_{s'}^T y_{sc} y_{s'c}}{\sum_{s''} y_{s''c}}\right), \tag{4}$$

where $\sum_{s''} y_{s''c}$ is a normalizing term for unbalanced categories.

The factor graph representing the explained model is shown in Figure 1-(b) via plate notation. In plate notation, the enumeration over random variables is denoted by plates. In classification based on gene expression data, there are N_G genes, N_S samples, and N_C categories. For the relationship between two distinct samples, the plate with index $s' \neq s$ is used. The priors over latent features are shown by directed relationships. The Gaussian prior with mean zero is assumed for all latent variables (relaxing the zero-mean assumption is straightforward). Each θ in the graphical model is the precision for the relevant variable where we assumed Multivariate Gaussian with independent elements. The prior mean of matrix W is the matrix μ_W.

The probability distribution over all random variables is

$$\begin{aligned} P(X, W, Y, V|\Theta) = \\ \frac{1}{Z}\exp(-\alpha\tau \sum_{gs} \frac{(x_g x_s^T + b_g - v_{sg})^2}{2} + (1-\alpha)\left(\sum_{sc} x_s W x_c^T y_{sc} + \sum_{s<s',c} \frac{x_s W W^T x_{s'}^T y_{sc} y_{s'c}}{\sum_{s''} y_{s''c}}\right) \\ - \frac{\theta_G}{2}\sum_g (||x_g||_2^2 + b_g^2) - \frac{\theta_S}{2}\sum_s ||x_s||_2^2 - \frac{\theta_C}{2}\sum_c ||x_c||_2^2 - \frac{\theta_W}{2}||W - \mu_W||_2^2), \end{aligned} \tag{5}$$

where Z is the partition function and $0 < \alpha < 1$ is the weight for combining expression level factor and classification factor. Such a weighting is necessary since the data is usually unbalanced — very fat in the gene expression side.

For learning the latent features, we only use the samples for which the category is given — the training set. However, it is possible to use the unlabeled data given the latent feature factor graph in a semi-supervised learning

manner — similar to the work of Zhu and Ghahramani [13] — which we leave for future directions. Therefore, in the learning phase, the latent feature parameters are learned given the observed variables v_{sg} and y_{sc}. Here we use maximum a posteriori probability (MAP) estimation which is the mode of the posterior distribution over the latent feature variables.

As another strategy, instead of finding the right weight parameter α, we optimize the probability distribution by alternating between the factors for the expression level and the category. That is, in an iterative way, first we optimize the log-posterior function over variables x_g, b_g, and x_s based on the factor f_1, and then over variables x_s, x_c, and W based on the factors f_2 and f_3. This is because of the fact that based on the joint probability distribution (5), the only common random variable between f_1, f_2, and f_3 is x_s. For making x_s satisfying both groups, we update the prior of x_s between alternating. As we mentioned, the prior over x_s is Gaussian with mean zero and precision θ_S. Given $\hat{x}_s$ as the output of one half of the algorithm, the prior over x_s will be a Gaussian with mean $\hat{x}_s/2$ and precision θ_S which is the posterior of the x_s given the first prior and the new recieved $\hat{x}_s$. That way, the prior mean over x_s for updating based on factor 1 is half of the x_s as the output of updating based on factors 2 and 3 and vice versa. We show the updated prior over x_s by μ_s.

For optimizing the log-posterior function based on the first factor, first we update x_g and b_g to fit them to the x_s coming from factors 2 and 3 phase. The equation for updating x_g is

$$x_g = (-\sum_s (b_g - v_{sg})x_s)(\sum_s x_s^T x_s + \theta_G I)^{-1} \tag{6}$$

for all g, where I is the identity matrix. The equation for updating b_g is

$$b_g = \frac{-\sum_s (x_s x_g^T - v_{sg})}{N_S + \theta_G} \tag{7}$$

for all g. The equation for updating x_s is

$$x_s = (\theta_S \mu_s - \sum_g (b_g - v_{sg})x_g)(\sum_g x_g^T x_g + \theta_S I)^{-1}. \tag{8}$$

Optimizing the log-posterior function based on the second and third factors is more complex due to the difficulty of computing the partition function. This is because of the exponential growth for summing over all possible y_{sc} variables in computing the partition function. To remedy this problem, instead of optimizing the log-posterior function given all variables, we work on the conditional probability of one sample given the labels for the rest of samples:

$$P(y_{sc}, X, W | y_{s' \neq s, c}) = (const) \frac{\exp\left(\sum_c x_s W x_c^T y_{sc} + \sum_{s' \neq s, c} \frac{x_s W W^T x_{s'}^T y_{sc} y_{s'c}}{\sum_{s'} y_{s'c}}\right)}{\sum_{y_{sc}} \exp\left(\sum_c x_s W x_c^T y_{sc} + \sum_{s' \neq s, c} \frac{x_s W W^T x_{s'}^T y_{sc} y_{s'c}}{\sum_{s'} y_{s'c}}\right)}$$
$$\exp\left(-\frac{\theta_S}{2}\sum_s ||x_s - \mu_s||_2^2 - \frac{\theta_C}{2}\sum_c ||x_c||_2^2 - \frac{\theta_W}{2}||W - \mu_W||_2^2\right), \tag{9}$$

where $const$ does not depend on any decision variable and sum over y_{sc} means enumerating over all categories for sample s. In this phase, we update the variables with regard to one sample via gradient ascent. The order of going over different samples is randomized. Note that the $x_{s'} | \forall s' \neq s$ variables are updated given (9) for sample s. The inside loop for updating $x_{s'}$ is randomized as well.

The algorithm for learning latent features is presented in Figure 2. Note that we repeat phase 2 until convergence since in phase 1 we use closed-form updates while in phase 2 we use gradient ascent and changes are slower. Finally, given latent feature variables, we can classify test sample t. First, using

$$x_t = (-\sum_g (b_g - v_{tg})x_g)(\sum_g x_g^T x_g + \theta_G I)^{-1}, \tag{10}$$

we map the sample t in the new space. Then using

$$P(y_{tc} | Y_{train}, X, W) = \frac{\exp\left(\sum_c x_t W x_c^T y_{tc} + \sum_{sc} \frac{x_t W W^T x_s^T y_{tc} y_{sc}}{\sum_s y_{sc}}\right)}{\sum_{y_{tc}} \exp\left(\sum_c x_t W x_c^T y_{tc} + \sum_{sc} \frac{x_t W W^T x_s^T y_{tc} y_{sc}}{\sum_s y_{sc}}\right)}, \tag{11}$$

```
Input [{y_sc}_sc, {v_sg}_sg, θ_C, θ_S, θ_G, θ_W, μ_W]
Output [{x_s}_s,{x_g}_g,{b_g}_g,{x_c}_c,W]
Initialize [{x_s}_s,{x_g}_g,{b_g}_g,{x_c}_c,W]
∀s μ_s ← x_s/2
Repeat
   Phase 1
      Update {x_g}_g using (6)
      Update {b_g}_g using (7)
      Update {x_s}_s using (8)
   ∀s μ_s ← x_s/2
   Phase 2
      Repeat
         For s ∈ RandomizedOrder
            optimize (9) over {x_s}_s,{x_c}_c,W by gradient ascent
         End
      Until(convergence)
   ∀s μ_s ← x_s/2
Until (convergence)
```

Figure 2: Iterative algorithm for learning latent features.

we can predict the class of the sample t.

Experimental results

For experimental evaluation, we used the 9-tumors dataset explained in [11]. In this dataset, there are 60 samples, 5726 genes, and 9 categories. We compared LFFG to SVM as a best known classifier for this problem [11]. LIBSVM [3] was used for evaluating SVM.

For SVM, we optimized over RBF and polynomial kernels via cross-validation. The best result was a linear kernel with $C = 200$. As preprocessing, we tried different normalizing procedures and for SVM, casting data to [0,1] interval worked best.

For LFFG, we optimized over $\theta_G = \theta_S = \theta_s = \theta$ (equal θ for all entities for simplicity of optimization) and $\theta = 30$ gave the best result. The step size for gradient ascent part was set to $1.0e-5$. Latent features were initialized by standard normal distribution. As preprocessing, we tried different normalizing procedures and normalizing by dividing data by the standard deviation of each gene worked best.

Due to the small size of the data, we repeated 10-fold cross-validation experiments 10 times — a total of 100 experiments. In each run, data was split into 10 mutual exclusive groups randomly. We did not implement stratified splitting since it impaired randomness given the small size of the dataset.

The results are given in Table 1. We use accuracy — the percentage of correctly labeled samples — for comparing LFFG and SVM. LFFG outperforms SVM in all runs and for eight runs the differences significant at 0.05. In total, LFFG outperforms SVM with p-value equal to 1.4e-10.

	run 1	run 2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	total
SVM	48.33	53.33	53.33	51.67	55.00	55.00	51.67	53.33	51.67	58.33	53.17
LFFG	**65.00**	63.33	**71.67**	**63.33**	**71.67**	**68.33**	**70.00**	**63.33**	**66.67**	68.33	**67.17**
p-value	0.002	0.109	0.006	0.044	0.032	0.026	0.009	0.012	0.041	0.130	1.4e-10

Table 1: The accuracy results of ten runs of 10-fold cross-validation experiments (total of 100 runs). The bold numbers are significant at 0.05 in paired t-test.

Conclusion

In this paper, we propose a novel classification approach for gene expression microarray classification — latent feature factor graph. LFFG assumes a latent feature variable for each entity — a random variable in a factor graph — and captures the relationship between entities with a function of the latent feature variables — a factor in a factor graph. We used MAP estimation for learning the latent features. Experimental results show that LFFG outperforms the popular classification algorithm SVM.

One important future direction is implementing Bayesian inference over latent feature variables instead of MAP. However, computing the posterior is intractable. We plan to examine two main approximate inference approaches for graphical models for LFFG: variational approximations [5] and Markov chain Monte Carlos [1].

In this paper, we only used linear functions. However, it is possible to use the Euclidean distance function which is useful for visualizing samples. Such an approach has been used in collaborative filtering [6] and information retrieval [7].

References

[1] C. Andrieu, N. De Freitas, A. Doucet, and M.I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[2] J.P. Brunet, P. Tamayo, T.R. Golub, and J.P. Mesirov. Metagenes and molecular pattern discovery using matrix factorization. *Proceedings of the National Academy of Sciences of the United States of America*, 101(12):4164, 2004.

[3] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):2–27, 2011.

[4] B.J. Frey. Extending factor graphs so as to unify directed and undirected graphical models. In *Proc. 19th Conf. Uncertainty in Artificial Intelligence*, pages 257–264, 2003.

[5] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[6] Mohammad Khoshneshin and W. Nick Street. Collaborative filtering via Euclidean embedding. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 87–94, New York, NY, USA, 2010. ACM.

[7] Mohammad Khoshneshin, W. Nick Street, and Padmini Srinivasan. Bayesian embedding of co-occurrence data for query-based visualization. In *Proceedings of IEEE International Conference on Machine Learning and Applications*, 2011.

[8] F.R. Kschischang, B.J. Frey, and H.A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

[9] S.L. Lauritzen. *Graphical models*, volume 17. Oxford University Press, USA, 1996.

[10] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM.

[11] A. Statnikov, C.F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 21(5):631, 2005.

[12] F. Valafar. Pattern recognition techniques in microarray data analysis. *Annals of the New York Academy of Sciences*, 980(1):41–64, 2002.

[13] X. Zhu and Z. Ghahramani. Towards semi-supervised classification with markov random fields. Technical report, Technical Report CMU-CALD-02-106, Carnegie Mellon University, 2002.