2014 22nd International Conference on Pattern Recognition

Feature Learning using Bayesian Linear Regression Model

Siqi Nie
Department of Electrical, Computer and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180
nies@rpi.edu

Qiang Ji
Department of Electrical, Computer and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180
qji@ecse.rpi.edu

***Abstract*—Data representation plays a key role in many machine learning tasks. Specific domain knowledge can help design some features, but it often needs a long time to hand-craft them. On the other hand, unsupervised learning can automatically learn a good representation of either labeled or unlabeled data. Currently one of the dominant approaches is the restricted Boltzmann machine (RBM). In this paper, we investigate an alternative approach for feature learning, which is based on Bayesian linear regression model. This model can also be denoted as Factor analysis, which is a statistical method for modeling the covariance structure of high dimensional data, but has not been used for feature learning. We will compare the proposed framework with RBM on different kinds of computer vision applications. Experiment results on different datasets are reported to demonstrate the effectiveness of the proposed feature learning framework.**

I. Introduction

Feature learning refers to learning the transformation of the 'raw' input to extract patterns or useful structures among the data for further supervised learning tasks such as classification or other purposes. Feature learning techniques can be either supervised or unsupervised, which commonly include auto-encoders, dictionary learning, restricted Boltzmann machine, k-means clustering and many other approaches. During the past few years, restricted Boltzmann machine draws more and more attention from researchers due to its capability of handling different kinds of data and its efficient learning method.

A typical latent variable density model for feature learning is the Gaussian restricted Boltzmann machine (GRBM) [5]. GRBM is based on Markov Random Field, and can be viewed as a mixture of diagonal Gaussians with the number of components exponential in the number of hidden units. In this work, our discussion focuses on Gaussian restricted Boltzmann machine, and GRBM and RBM are used interchanged in the following text.

In this paper, we intend to explore an alternative approach for feature learning. The basic idea is to utilize the mature theories of learning and inference in directed graphic model to perform the feature learning task. Specifically, we propose to use Bayesian linear regression model (BLR) as building blocks for feature learning. Bayesian linear regression model is also denoted as Factor Analyser [2]. Unlike GRBM, Factor Analyser is a directed graphical model where a multivariate standard normal prior is specified for the latent variables. Factor Analyser is an approach for real-valued high-dimensional data modeling, which simultaneously performs clustering and dimensionality reduction of the data. However, to the best of our knowledge, it has not been used for feature learning.

The Expectation-Maximization algorithm for learning Factor Analyser, and a more sophisticated model, mixture of Factor Analysers, are first proposed by Ghahramani and Hinton [2]. As an extension of the Factor Analyser, deep mixture of Factor Analysers is introduced by Tang *et al.* [8]. Instead of assuming the Gaussian distribution of the hidden factors, another layer of Factor Analysers is used to model the prior distribution of the first layer of hidden variables. Such deep model avoids over-fitting, and it outperforms RBM in data representation in terms of the likelihood of data. Furthermore, Tensor Analyser is introduced in [9], in which the loading matrix is replaced with a loading tensor. As the number of parameters grows, the representation power is also enhanced. However, neither work uses the hidden variables as features for supervised tasks such as classification. In fact, with the representation power of Factor Analyser, it is natural to use it as tool for feature learning.

In this work, we will compare the feature learning performance of RBM and Bayesian linear regression model (BLR). The comparison is limited to a double-layer graphical model, which means we only have one layer of hidden variable. We believe that if effectiveness is demonstrated in a double-layer model, it is easy to generalize to deep models.

The remainder of this paper is organized as follows: feature learning using restricted Boltzmann machine is briefly introduced in Section II. In Section III we introduce the method of feature learning using Bayesian linear regression model. These two approaches are compared in Section IV. Section V concludes the paper.

II. Restricted Boltzmann Machine

In this section, we give a brief introduction of restricted Boltzmann machine. A standard RBM is basically a Markov Random Field, in which all the variables are separated into two layers: visible layer and hidden layer. Typically visible layer represent the 'raw' data, which is the observation to the RBM. Observation can be discrete or continuous, depending on the specific application. For example, for digit recognition, the

1051-4651/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPR.2014.267

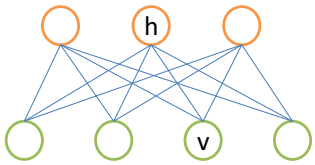

Fig. 1. Illustration of the restricted Boltzmann machine. Upper layer represents hidden variables; bottom layer represents observations.

input is typically assumed to be binary. For natural image patch modeling, the input is usually continuous. In this research, we focus on real-valued data. The hidden layer consists of binary variables, which determines the existence of some pattern in the observation layer. A graphic illustration of RBM is Figure 1. There is a link between each pair of hidden unit and visible variable. A weight w_{ij} is associated with the link between visible variable v_i and hidden unit h_j.

For each configuration $(\mathbf{v}, \mathbf{h})$, the model assigns an energy function:

$$E(\mathbf{v}, \mathbf{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{i,j} \frac{v_i}{\sigma_j} w_{ij} h_j - \sum_j b_j h_j \,, \quad (1)$$

where a_i is the bias for visible unit v_i, σ_i is the standard deviation of the Gaussian distribution, which is typically 1 if we normalize the data, b_j is the bias for the hidden unit h_j.

From the energy function, the joint probability of visible and hidden vector is defined as:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \,, \quad (2)$$

where Z is the partition function. With continuous inputs, the Z is calculated by integrating over all visible nodes and summing over all hidden unit configurations.

From the energy function (Equation 1) and likelihood function (Equation 2), we can derive the probability of an hidden unit being activated, given the visible layer:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i \mathbf{v}_i^T \mathbf{w}_{ij.}) \,, \quad (3)$$

where $\sigma(\cdot)$ is the sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$.

Similarly, given the hidden layer, the distribution of the visible variable is:

$$p(v_i|\mathbf{h}) = \mathcal{N}(\sum_j h_j w_{ij} + a_i, 1) \,. \quad (4)$$

The reason that the hidden layer can be seen as features is that one hidden unit is connected to all visible units, so whether it is activated or not depends stochastically on the visible layer through Equation 3. The weights act like a detector for some specific pattern in the observation. Therefore the hidden layer $\mathbf{h}$ is able to capture important patterns of $\mathbf{v}$.

The likelihood of input data is computed by summing over all configurations of hidden units:

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h})} \,. \quad (5)$$

The parameter learning includes bias for visible and hidden units a_i, b_j, and weights between two layers $\mathbf{w}$. To learn the parameters, we seek to maximize the joint probability of all training data D, $P(D) = \prod_{\mathbf{v} \in D} p(\mathbf{v})$. The derivative of the log-likelihood of a training instance with respect to a parameter is given below:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \,, \quad (6)$$

$$\frac{\partial \log p(\mathbf{v})}{\partial b_j} = \langle h_j \rangle_{data} - \langle h_j \rangle_{model} \,, \quad (7)$$

where the angle brackets denotes expectations under the distribution specified by the subscript that follows: data-dependent expectation and model-dependent expectation. A straightforward learning rule is stochastic steepest ascent in the log-likelihood of training data. Take w_{ij} as an example:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \,, \quad (8)$$

where ϵ is the learning rate. $\langle v_i h_j \rangle_{data}$ is easy to get from the data. $\langle v_i h_j \rangle_{model}$ is much more difficult to compute due to the large dimension of hidden and visible layers. Hinton [4] proposes the Contrastive Divergence (CD) algorithm to approximate $\langle v_i h_j \rangle_{model}$ by using a reconstructed sample, which gives the following weight update rule:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{recon}) \,. \quad (9)$$

Given the visible layer, the hidden units are independent with each other, so sampling the states of hidden units can be performed in parallel using Equation 3. With the reconstructed data, we are able to calculate the approximate gradient of each parameter, and perform the CD learning procedure.

With all the parameters, the probability of the hidden units given the observation is used as features extracted from the data.

III. Bayesian Linear Regression Model

Bayesian liner regression model is a rival model to RBM. A graphical representation of BLR is given in Figure 2. Basically, a BLR is a densely connected Bayesian network, which is the most significant difference from restricted Boltzmann machine.

In BLR, a real-valued data $\mathbf{x}$ of dimension d is represented using a real-valued factor $\mathbf{z}$ in dimension k, where k is generally much smaller than d.

$$\mathbf{x} = W\mathbf{z} + \mathbf{u} \,, \quad (10)$$

where W is a $d \times k$ loading matrix. Following the Bayesian network parameterization, the prior probability of the hidden vector $\mathbf{z}$ is assumed to follow Gaussian distribution with zero mean and unit variance, where the components are independent of each other. d-dimensional vector $\mathbf{u}$ also follows normal distribution, $\mathbf{u} \sim \mathcal{N}(0, \Psi)$, where Ψ is a diagonal matrix.

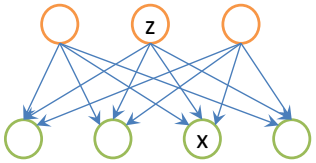

Fig. 2. Illustration of Bayesian linear regression model. Upper layer represents hidden variables; bottom layer represents observations.

According to the model, the conditional probability of $\mathbf{x}$ given $\mathbf{z}$ is:

$$\mathbf{x} \sim \mathcal{N}(0, WW' + \Psi)\,. \tag{11}$$

Since the model involves latent variables, the typical method for learning the parameters is the parameter Expectation-Maximization (EM) algorithm. The EM algorithm consists of two steps: E-step and M-step as follows.

E-step: Compute $E(\mathbf{z}|\mathbf{x}_i)$ and $E(\mathbf{z}\mathbf{z}'|\mathbf{x}_i)$ for each data point $\mathbf{x}_i$, given the current parameters. The expectations are with respect to the current parameters.

M-step: The expected log-likelihood (Q function) for BLR is

$$\begin{aligned} Q \;=\; & c - \frac{n}{2} - \sum_i E\left[\log p(\mathbf{z})\right] \\ & - \sum_i \left(\frac{1}{2}\mathbf{x}_i'\Psi^{-1}\mathbf{x}_i - \mathbf{x}_i'\Psi^{-1}WE\left[\mathbf{z}|\mathbf{x}_i\right]\right) \\ & - \sum_i \frac{1}{2} tr\left[W'\Psi^{-1}WE\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right]\right]\,, \end{aligned} \tag{12}$$

where c is a constant, independent of the parameters, and tr is the trace operator. To maximize the Q function, we take derivative with respect to each parameter.

$$\frac{\partial Q}{\partial W} = -\sum_i \Psi^{-1}\mathbf{x}_i E\left[\mathbf{z}|\mathbf{x}_i\right]' + \sum_i \Psi^{-1}W^{\text{new}}E\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right] = 0\,. \tag{13}$$

Then the loading matrix W can be updated as:

$$W^{\text{new}} = \left(\sum_i \mathbf{x}_i E\left[\mathbf{z}|\mathbf{x}_i\right]'\right)\left(\sum_i E\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right]\right)^{-1}\,. \tag{14}$$

Similarly,

$$\Psi^{\text{new}} = \frac{1}{n} diag\left[\sum_i \mathbf{x}_i\mathbf{x}_i' - W^{\text{new}}E\left[\mathbf{z}|\mathbf{x_i}\right]\mathbf{x}_i'\right]\,. \tag{15}$$

Once we have the parameters of the model, $E(\mathbf{z}|\mathbf{x})$ is used as the feature extracted from the observation. The expectation can be computed through a linear projection:

$$E(\mathbf{z}|\mathbf{x}) = W'(\Psi + WW')^{-1}\,. \tag{16}$$

The above results are based on continuous hidden variables. As a fair rival model for RBM, we can set also the hidden units in BLR to be binary, which leads to the following change in the learning process.

For example, if we have 2 latent variables, z_1 and z_2, with initial prior probability τ_1 and τ_2, respectively. First the posterior probability $p(\mathbf{z}|\mathbf{x}_i)$ for every data point $\mathbf{x}_i$ is computed:

$$\begin{aligned} T_{1,i} &= p(\mathbf{z} = (0,0)^T|\mathbf{x}_i)\,, \\ T_{2,i} &= p(\mathbf{z} = (0,1)^T|\mathbf{x}_i)\,, \\ T_{3,i} &= p(\mathbf{z} = (1,0)^T|\mathbf{x}_i)\,, \\ T_{4,i} &= p(\mathbf{z} = (1,1)^T|\mathbf{x}_i)\,. \end{aligned} \tag{17}$$

The computation is performed using Bayes Rule:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{z},\mathbf{x})}{\sum_{\mathbf{z}'} p(\mathbf{z}',\mathbf{x})}\,. \tag{18}$$

Notice that given data point $\mathbf{x}_i$, the latent variables are not independent, so the posterior probability cannot be factorized into a product of individual latent variables. We have to compute the posterior probability for every vector of latent variables, which means if the latent space is large in dimension, there will be exponential number of posterior probabilities to compute, and this is one limitation for scaling up the model.

Given the posterior probabilities for each configuration of the latent variables, the expectation is easy to compute. For the same example:

$$\begin{aligned} E(\mathbf{z}|\mathbf{x}_i) = T_{1,i}\begin{bmatrix}0\\0\end{bmatrix} + T_{2,i}\begin{bmatrix}0\\1\end{bmatrix} + \\ T_{3,i}\begin{bmatrix}1\\0\end{bmatrix} + T_{4,i}\begin{bmatrix}1\\1\end{bmatrix}\,, \end{aligned} \tag{19}$$

$$\begin{aligned} E(\mathbf{z}\mathbf{z}'|\mathbf{x}_i) = T_{1,i}\begin{bmatrix}0&0\\0&0\end{bmatrix} + T_{2,i}\begin{bmatrix}0&0\\0&1\end{bmatrix} + \\ T_{3,i}\begin{bmatrix}1&0\\0&0\end{bmatrix} + T_{4,i}\begin{bmatrix}1&1\\1&1\end{bmatrix}\,. \end{aligned} \tag{20}$$

The updating rule for the prior probability of each hidden unit τ_j is:

$$\tau_j^{\text{new}} = \frac{1}{n}\sum_i p(z_j = 0|\mathbf{x}_i, \theta^{\text{old}})\,. \tag{21}$$

For binary hidden variables, the features extracted from the data are $p(\mathbf{z}|\mathbf{x})$. For continuous hidden variables, the features are $E(\mathbf{z}|\mathbf{x})$.

The BLR and RBM have several differences. First, in RBM, the distribution of a visible variable is determined by the states of hidden layer through Equation 4. Suppose the size

of the hidden layer is m, then there are 2^m possible Gaussian components in the input space. However, in BLR, the hidden units are continuous, which means there are infinite number of possible Gaussian components. This fact may give BLR more representation power. Another difference is the property of the graphical representation. In undirected model as in RBM, given the observation, the hidden units (features) are independent of each other, so different features do not have correlations. While in BLR, since the hidden units are parents of the observations, they are related given the input data, such that we may learn some certain feature correlations.

IV. Experiment

In this section, we compare the performances of RBM and BLR in feature learning for classification. The experiments are performed in three different domains: widely used datasets in UCI machine learning repository [1], CIFAR-10 dataset [6] for object recognition, and the CK+ dataset [7] for expression recognition. To fairly compare RBM and BLR, we do not apply any fine tuning, or other process. Both models are assigned the same number of hidden variables, with only one hidden layer. Classification is performed using svm with the features extracted from data.

A. UCI Datasets

First, we use two datasets available at the UCI machine learning repository to evaluate the proposed model. The datasets are wine (14 variables, 178 instances, 2 classes) and breast cancer (31 variables, 569 instances, 3 classes). We use a 5-fold cross-validation setting, each time $4/5$ of the data as training data, $1/5$ as testing data. The hidden variables are set to be binary, both in RBM and BLR. The classification accuracy is the criterion for comparison. Due to the uncertainty brought by random initialization of both models, the experiment is performed 10 times with different initializations. The maximum, minimum, and average recognition accuracy are recorded. Details can be found in Table I and II. From Table 3, for small size of hidden layer, BLR is a better representation in both training and testing phases. Considering the dimension of the dataset, BLR can simultaneously learns the features and reduces the dimensionality of data, while RBM needs much more hidden units to capture good patterns in the input data. For the Breast Cancer dataset, BLR significantly outperforms RBM in terms of both training and testing accuracy. It captures better patterns in the data.

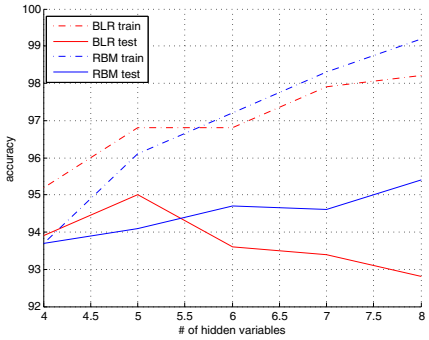

Fig. 3. Recognition accuracy with different number of hidden nvariables, Wine dataset. (Best viewed in color)

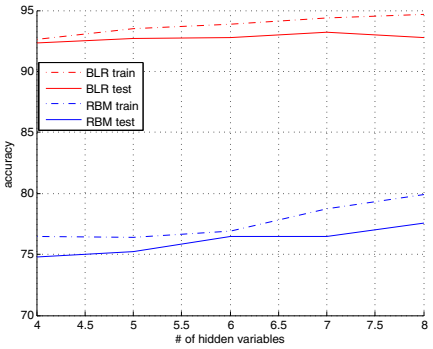

Fig. 4. Recognition accuracy with different number of hidden nvariables, Breast Cancer dataset. (Best viewed in color)

Notice that the max, min, and average recognition accuracy are in terms of the training phase. The testing performances are from the corresponding models, which means that testing accuracy may not necessarily be max, min, or average.

Figure 3 and 4 show the influence of different number of hidden variables on the recognition performance. Generally, with more hidden nodes, both models are able to capture more accurate patterns in the input data, and achieve better recognition accuracy. It seems that for BLR, there exists a threshold, beyond which the testing performance will decrease as the number of hidden nodes increases, although training accuracy will continue to increase. To prevent over-fitting

TABLE I. Comparison of the performances of feature learning using RBM and BLR with various number of hidden variables, Wine dataset (%)

		RBM			BLR		
# hidden		max	min	average	max	min	average
4	train	95.8	91.4	93.7	98.7	85.5	95.2
	test	92.6	93.2	93.7	97.2	88.9	93.9
5	train	98.0	94.1	96.1	99.4	93.1	96.8
	test	92.7	95.5	94.1	97.2	92.6	95.0
6	train	99.0	94.7	97.2	99.2	90.0	96.8
	test	92.7	92.1	94.7	94.3	89.3	93.6
7	train	99.3	96.9	98.3	99.4	95.4	97.9
	test	92.1	94.4	94.6	95.5	91.5	93.4
8	train	99.9	97.9	99.2	99.6	95.9	98.2
	test	96.6	96.6	95.4	90.5	90.4	92.8

TABLE II. Comparison of the performance of feature learning using RBM and BLR with various number of hidden variables, Breast Cancer dataset (%)

		RBM			BLR		
# hidden		max	min	average	max	min	average
4	train	78.8	74.2	76.5	94.1	91.6	92.6
	test	76.8	71.5	74.8	93.2	91.1	92.3
5	train	78.6	73.3	76.4	94.5	92.1	93.5
	test	76.7	73.8	75.2	93.7	91.4	92.7
6	train	79.5	73.7	76.9	94.8	92.6	93.9
	test	76.5	75.1	76.5	92.6	92.0	92.8
7	train	80.6	76.6	78.7	95.4	93.4	94.4
	test	75.5	76.3	76.5	94.2	93.3	93.2
8	train	82.8	77.2	79.9	95.5	93.7	94.7
	test	81.4	73.5	77.6	92.1	92.3	92.8

TABLE III. COMPARISON OF THE AVERAGE PERFORMANCE OF FEATURE LEARNING USING RBM AND BLR WITH VARIOUS NUMBER OF HIDDEN VARIABLES, CIFAR-10 DATASET (%)

# hidden	RBM	BLR
10	21.5	24.7
15	22.4	24.15
20	24.5	25.8
25	24.2	26.7
30	25.6	27.3

problem, the number of hidden nodes in BLR need to be limit to a certain amount.

B. CIFAR-10 dataset

CIFAR-10 dataset consists of 60,000 natural color images with dimension 32×32 in 10-classes, 50,000 images for training and 10,000 for testing. As a pre-processing procedure, we normalized the images by subtracting the per-pixel mean computed over the training set from each image.

Since our testing is based on a relatively small number of hidden variables, and a shallow structure, we make some simplification of the dataset. We use the 10,000 images of the first batch, 8,000 images as training data, 2,000 as testing data. Only Red channel is kept, and the size of images is down-sampled to 8×8. Common approaches learn features from small image patches, while we learn the features from the whole images. The learning of BLR is terminated when the likelihood converges, and the maximum epoch of RBM is set to be 150 according to the suggestion in [3]. The features learned from RBM or BLR is used as input to svm for classification.

The classification results are given in Table III. The features learned from BLR have comparable discriminative capability as features learned from RBM. With limited number of hidden units, BLR has a better interpretation power over RBM. Due to the non-convex property of the objective function of RBM, different initialization may end up in local optima. Therefore RBM needs a lot of hidden units to get good representation of the input data. However, as a tool for dimensionality reduction, BLR only needs a small number of hidden layers to capture the important patterns in the data. Although the overall classification accuracy is low, after some commonly used complicated process, both results can be boosted to much higher level.

Some of the learned features from two models are shown in Figure 5. We can discover vaguely that the features learned by BLR is more concentrated than the features learned by RBM. Features from RBM has a trend to spread over the whole image area.

C. CK+ dataset

The Extended Cohn-Kanade Dataset (CK+) is a complete dataset for action units and emotion-specified expressions, including 593 sequences from 123 subjects. From all the sequences, 327 are associated with expression labels: *angry, disgust, fear, happy, sadness, surprise and contempt*. The facial landmarks (*e.g.,* the position of eye corner, lips, *etc.*) are provided together with the video sequences. In this work, we use the facial landmarks on the peak frame, in which the subject

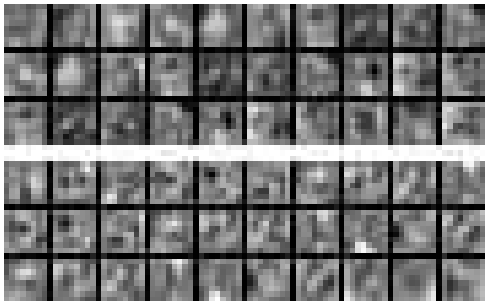

Fig. 5. Some of the features learned from BLR (upper) and RBM (lower).

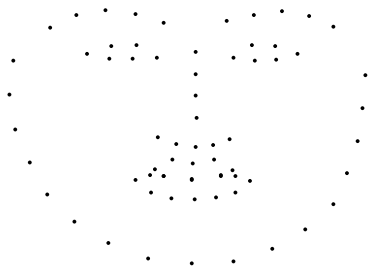

Fig. 6. Illustration of the facial landmarks.

performs the expression to the peak level, to classify it into one of seven expressions. An illustration of the facial landmarks is given in Figure 6. Since expressions are only related to the inner points, we remove the landmarks on the outline of the face, which results in 76-dimension feature vectors. We follow the same leave-one-subject-out cross-validation settings as in [7].

As the pose of each subject are slightly varying all the time, we apply a pose rectification procedure to make each face a frontal one. Then we use a normalization procedure by making the size of facial area the same throughout all subjects, and align the features by moving the centers of eyes to the same position.

The classification performance with different number of hidden variables are given in Table IV. BLR outperforms RBM for most settings. Such result is already better than the baseline method (83.3%) as reported in [7], and the state-of-the-art method (86.3%) as reported in [10].

From the result, one peak image is enough for facial

TABLE IV. PERFORMANCE OF RBM AND BLR WITH VARIOUS NUMBER OF HIDDEN VARIABLES, ON CK+ DATASET (%)

# hidden	RBM	BLR
10	85.3	86.3
15	86.8	87.2
20	87.3	87.2
25	87.6	88.8
30	87.8	86.6

expression recognition. BLR, as well as RBM, can effectively extract some important patterns in input data, for classification purpose.

V. Conclusion and Future Work

Restricted Boltzmann machine is commonly used for unsupervised feature learning in recent research. In this paper, we investigate an alternative model, Bayesian linear regression model, for feature learning. Experiments on different kinds of applications and datasets demonstrate the potential of BLR as the building block for feature learning. Bayesian linear regression model can simultaneously learn the important patterns and reduce the dimensionality of the input data, while RBM tends to need much more hidden variables to get a good interpretation of the data. The models in this work are limited to a shallow 2-layer structure, but with some sophisticated modification, it can obtain much more representation power with additional layers and act as an important tool for feature learning. One limitation of BLR is that the EM algorithm takes more time to converge for larger domains. For future work, we intend to extend the model to larger data sets, as well as to feature learning in a deep manner.

Acknowledgment

The work described in this paper is supported in part by the grant N00014-12-1-0868 from the Office of Navy Research.

References

[1] A. Asuncion and D. Newman. UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

[2] Z. Ghahramani and G. E. Hinton. The em algorithm for mixtures of factor analyzers. Technical report, Technical Report CRG-TR-96-1, University of Toronto, 1996.

[3] G. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1), 2010.

[4] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[5] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[6] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[7] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews. The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 94–101. IEEE, 2010.

[8] Y. Tang, G. E. Hinton, and R. Salakhutdinov. Deep mixtures of factor analysers. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 505–512, 2012.

[9] Y. Tang, R. Salakhutdinov, and G. Hinton. Tensor analyzers. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013.

[10] Z. Wang, S. Wang, and Q. Ji. Capturing complex spatio-temporal relations among facial muscles for facial expression recognition. In *CVPR*, 2013.