1 Introduction

With the rapid development of both the astronomical instruments and various machine learning algorithms, we can apply the spectral characteristics of stars to classify the stars. A great quantity of astronomical observatories have been built to get the spectra, such as the Large Sky Area Multi-Object Fibre Spectroscopic Telescope (LAMOST) in China. A variety of astronomical instrumentation, methods and techniques—methods: analytical—methods: statistical data analysis—methods: statistical multivariate analysis methods, i.e., PCA and discriminant function analysis, are used multi-layer back propagation (BP) neural network for classification.

In 2006, Sarty and Wu applied two well known multivariate analysis methods, i.e., PCA and discriminant function analysis, to analyze the spectroscopic emission data collected by Williams (1983). By using the PCA method, they found that the source of variation had correlation to the binary orbital period. With the discriminant function analysis, they found that the source of variation was connected with the equivalent width of the $H\beta$ line (Sarty & Wu 2006).

In 2010, Rosalie et al. applied PCA to analyze the stellar spectra obtained from SDSS (Sloan Digital Sky Survey) DR6 (McGurk, Kimball, & Ivezić 2010). They found that the first 4 principal components (PCs) could remain enough information of the original data without overpassing the measurement noise. Their work made classifying novel spectra, find-
ing out unusual spectra and training a variety of spectral classification methods etc. not as hard as before.

In 2012, Bazarghan applied self-organizing map (SOM, a kind of unsupervised artificial Neural Network (ANN) algorithm) to stellar spectra obtained from the Jacoby, Hunter and Christian (JHC) library, and the author obtained the accuracy of about 92.4% (Bazarghan et al. 2012). In the same year, Navarro et al. used the ANN method to classify the stellar spectra with low signal-to-noise ratio (S/N) on the samples of field stars which were along the line of sight toward NGC 6781 and NGC 7027 etc. (Navarro, Corradi, & Mampaso 2012). They not only trained, but also tested the ANNs with various S/N levels. They found that the ANNs were insensitive to noise and the ANN’s error rate was smaller when there were two hidden layers in the architecture of the ANN in which there were more than 20 hidden units in each hidden layer.

In the above, some applications of PCA for dimension reduction and ANN for spectral classification were reviewed in astronomy. Furthermore, SVM and decision trees have also been used for spectral classification in astronomy.

In 2004, Zhang and Zhao applied single-layer perceptron (SLP) and support vector machines (SVMs) etc. for the binary classification problem, i.e., the classification of AGNs (active galactic nucleus) and S & G (stars and normal galaxies) (Zhang & Zhao 2004), in which they first selected features using the histogram method. They found that SVM’s performance was as good as or even better than that of the neural network method when there were more features chosen for classification. In 2006, Ball et al. applied decision trees to SDSS DR3 (Ball et al. 2006). They investigated the classification of 143 million photometric objects and they trained the classifier with 477,068 objects. There were three classes, i.e., galaxy, star and neither of the former two classes, in their experiment.

From the perspective of feature extraction methods, some researches in spectral classification based on linear dimension reduction technique, e.g., PCA, have been reviewed. Except from linear dimension reduction method, nonlinear dimension reduction technique has also been applied in spectral classification for feature extraction.

In 2011, Daniel et al. applied locally linear embedding (LLE, a well known nonlinear dimension reduction technique) to classify the stellar spectra coming from the SDSS DR7 (Daniel et al. 2011). There were 85,564 objects in their experiment. They found that most of the stellar spectra was approximately a 1d sequence lying in a 3d space. Based on the LLE method, they proposed a novel hierarchical classification method being free of the feature extraction process.

1.2 Previous application of RBM

In this subsection, we present some representative applications of the RBM algorithm so far.

In 2007, Salakhutdinov et al. (Salakhutdinov, Mnih, & Hinton 2007) applied RBM for collaborative filtering, which is closely related to recommendation system in machine learning community. In 2008, Gunawardana and Meek (Gunawardana & Meek 2008) applied RBM for cold start recommendations. In 2009, Taylor and Hinton (Taylor & Hinton 2009) applied RBM for modeling motion style. In 2010, Dahl et al. (Dahl et al. 2010) applied RBM to phone recognition on the TIMIT dataset. In 2011, Schloter and Osendorf (Schloter & Osendorf 2011) applied RBM to estimate music similarity. In 2012, Tang et al. (Tang, Salakhutdinov, & Hinton 2012) applied RBM for recognition and de-noising on some public face databases.

1.3 Our work

In this study, we applied the binary RBM algorithm to classify spectra of CVs and non-CVs obtained from the SDSS.

Generally, before applying a classifier for classification, the researchers always preprocess the original data, for example, normalization to get better features and thus to get better performance. Thus, firstly, we normalize the spectra with unit norm. Then, to apply binary RBM for spectral classification, we binarize the normalized spectra by some rule which we will discuss in the experiment. Finally, we use the binary RBM for classification of the data, one half of all the given data for training and the other half for testing. The experiment result shows that the classification accuracy is 100%, which is state-of-the-art. And RBM outperforms the prevalent classifier, SVM, with accuracy of 99.14% (Bu et al. 2013).

The rest of this paper is organized as follows. In section 2, we review the prerequisites for training restricted Boltzmann machine. In section 3, we introduce the binary RBM and the training algorithm for RBM. In section 4, we present the experiment result. Finally, in section 5, we conclude our work in this study and also present the future work.

2 Prerequisites

2.1 Markov Chain

A Markov chain is a sequence composed of a number of random variables. Each element in the sequence can transit from one state to another one randomly. Indeed, a Markov chain belongs to a stochastic process (Andrieu et al. 2003). In general, the number of possible states for each element or random variable in a Markov chain is finite. And a Markov chain is a random process without memory. It is the current state rather than the states preceding the current state that can influence the next state of a Markov chain. This is the well known Markov Property (Xiong, Jiang, & Wang 2012).

Mathematically, a Markov chain is a sequence, $X_1$, $X_2$, $X_3$, ..., with the following property:

$$P(X_{n+1} = x_{n+1} | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n),$$

2We say the norm of a vector $x = (x_1, \ldots, x_n)$ is unit, if $\sum x_i^2 = 1.$
where the $X_i$ $(i = 1, 2, \ldots)$ is a random variable and it usually can take on finite values for a specific problem in the real world. And all the values as a whole can form a denumerable set $S$, which is commonly called the state space of the Markov chain (Yang et al. 2009).

Generally, all the probabilities of the transition from one state to another one can be represented as a whole by a transition matrix. And the transition matrix has the following three properties:

- square: both the row number of the matrix and the column number of the matrix equal the total number of the states that the random variable in the Markov chain can take on;
- the value of a specific element is between 0 and 1: it represents the transition probability from one state to another one;
- all the elements in each row sum to 1: The sum of the transition probabilities from any specific one state to all the states equals 1.

If the initial vector, a row vector, is $X_0$, and the transition matrix is $T$, then after $n$ steps of inference, we can get the final vector $X_0 \cdot T^n$.

Then we introduce the equilibrium of a Markov chain. If there exists an integer $N$, which renders all the elements in the resulting matrix $T^N$ nonzero, or rather, greater than 0, then we say that the transition matrix is a regular transition matrix (Greenwell, Ritchey, & Lial 2003). If the transition matrix $T$ is a regular transition matrix, and there exists one and only one row vector $V$ satisfying the condition that $v \cdot T^n$ approximately equals $V$, for any probability vector $v$ and large enough integer $n$, then we call the vector $V$ as the equilibrium vector of the Markov chain.

### 2.2 MCMC

Markov chain Monte Carlo (MCMC) is a sampling algorithm from a specific probability distribution. For the detailed information of MCMC, the readers are referred to Andrieu et al. (2003). The sampling process proceeds in the form of a Markov chain and the goal of MCMC is to get a desired distribution, or rather, the equilibrium distribution via running many inference steps. The larger the number of iterations is, the better the performance of the MCMC is. And MCMC can be applied for unsupervised learning with some hidden variables or maximum likelihood estimation (MLE) learning of some unknown parameters (Andrieu et al. 2003).

### 2.3 Gibbs Sampling

Gibbs sampling method can be used to obtain a sequence of approximate samples from a specific probability distribution, in which sampling directly is usually not easy to implement. For the detailed information of Gibbs sampling, the readers are referred to Gelfand (2000). The sequence obtained via the Gibbs sampling method can be applied to approximate the joint distribution and the marginal distribution with respect to (w.r.t) one of all the variables etc. In general, Gibbs sampling method is a method for probabilistic inference.

Gibbs sampling method can generate a Markov chain of random samples under the condition that each of the sample is correlated with the nearby sample, or rather, the probability of choosing the next sample equals to 1 in Gibbs sampling (Andrieu et al. 2003).

### 3 RBM

Considering that RBM is a generalized version of Boltzmann Machine (BM), we first review BM in this section. For the detailed information of BM, the readers are referred to Ackley, Hinton, & Sejnowski (1985).

BM can be regarded as a bipartite graphical generative model composed of two layers in which there are a number of units with both inter-layer and inner-layer connections. One layer is a visible layer $v$ with $m$ binary visible units $v_i$, i.e., $v_i = 0$ or $v_i = 1$ $(i = 1, 2, \ldots, m)$. For each unit in the visible layer, the corresponding value is observable. The other layer is a hidden (latent) layer $h$ with $n$ binary hidden units $h_j$.

As in the visible layer, $h_j = 0$ or $h_j = 1$ $(j = 1, 2, \ldots, n)$. For each unit or neuron in the hidden layer, the corresponding value is hidden, latent or unobservable, and it needs to be inferred.

The units coming from the two layers of a BM are connected with weighted edges completely, with the weights $w_{ij}$ $(v_i \leftrightarrow h_j)$ $(i = 1, 2, \ldots, m, j = 1, 2, \ldots, n)$. For the two layers, the units within each specific layer are also connected with each other, and also with weights.

For a BM, the energy function can be defined as follows:

$$
E(v, h) = - \sum_{i,j=1}^{m} v_i a_{ij} v_j - \sum_{i,j=1}^{n} h_i d_{ij} h_j - \sum_{i=1}^{m} v_i w_{ij} h_j - \sum_{i=1}^{n} h_i c_i - \sum_{j=1}^{n} h_j b_j,
$$

where $a_{ij}$ is the weight of the edge connecting visible units $v_i$ and $v_j$, $d_{ij}$ the weight of the edge connecting hidden units $h_i$ and $h_j$, $w_{ij}$ the weight of the edge connecting visible unit $v_i$ and hidden unit $h_j$. For a RBM, the $b_j$ is the bias for the hidden unit $h_j$ in the following activation function (Sigmoid function $f(x) = \text{sigmoid}(x) = 1/(1 + e^{-x})$)

$$
p(h_j = 1|v) = \frac{1}{1 + e^{-b_j - \sum_{i=1}^{m} w_{ij} v_i}}.
$$

And in a RBM, the $c_i$ is the bias for the visible unit $v_i$ in the following formula:

$$
p(v_i = 1|h) = \frac{1}{1 + e^{-c_i - \sum_{j=1}^{n} w_{ij} h_j}}.
$$

Then for each pair of a visible vector and a hidden vector $(v, h)$, the probability of this pair can be defined as follows:

$$
p(v, h) = \frac{e^{-E(v,h)}}{P_F},
$$
where the denominator $PF$ in the fraction (a partition function) is:
\[
PF = \sum_{\bar{v}, \bar{h}} p(\bar{v}, \bar{h}).
\]  

Besides, a RBM is a graphical model with the units for both layers not connected within a specific layer, i.e., there are only connections between the two layers for the RBM (Hinton & Salakhutdinov 2006). Mathematically, for a RBM, $a_{ij} = 0$ for $i, j = 1, 2, \ldots , m$ and $d_{ij} = 0$ for $i, j = 1, 2, \ldots , n$. Thus, the states of all the hidden units $h_i$'s are independent given a specific visible vector $v$ and so are the visible units $v_i$'s given a specific hidden vector $h$. Then we can obtain the following partial derivative:
\[
p(h|v) = \prod_j p(h_j|v) \quad \text{and} \quad p(v|h) = \prod_i p(v_i|h).
\]

### 3.1 Contrastive Divergence

Contrastive Divergence (CD) is proposed by Hinton and it can be used to train RBM (Hinton, Osindero, & Teh 2006). Initially, we are given $v_i$ ($i = 1, 2, \ldots , m$), then we can obtain $h_j$ ($j = 1, 2, \ldots , n$) by the sigmoid function given in the above. And the value of $h_j$ is determined by comparing a random value $r$ ranging from 0 to 1 with the probability $p(h_j = 1|v)$. Then we can reconstruct $v$ by $p(v_i = 1|h)$.

We can repeat the above process backward and forward until the reconstruction error is small enough and it has reached the maximum number of iterations which is set beforehand. To update the weights and biases in a RBM, it is necessary to compute the following partial derivatives:
\[
\frac{\partial \log p(v,h)}{\partial v_i} = E_{\text{data}}[v_i|h] - E_{\text{recon}}[v_i|h], \tag{3}
\]
\[
\frac{\partial \log p(v,h)}{\partial h_j} = v_i - E_{\text{recon}}[v_i], \tag{4}
\]
\[
\frac{\partial \log p(v,h)}{\partial b_i} = E_{\text{data}}[h_i] - E_{\text{recon}}[h_i], \tag{5}
\]

where $E[\star]$ represents the expectation of $\star$, and the subscript 'data' means that the probability is original-data-driven while the subscript 'recon' means that the probability is reconstructed-data-driven.

Then the weight can be updated according to the following rule:
\[
\Delta w_{ij} = \eta (E_{\text{data}}[v_i|h_j] - E_{\text{recon}}[v_i|h_j]),
\]

where $\eta$ is a learning rate, which influences the speed of convergence. And the biases can be updated similarly.

In equations (3)-(5), $E_{\text{data}}[\star]$'s are easy to compute. To compute or inference the latter term $E_{\text{recon}}[\star]$, we can resort to MCMC.

### 3.2 Free energy and Soft-max

To apply RBM for classification, we can resort to the following technique. We can train a RBM for each specific class. And for classification, we need the free energy and the soft-max function for help. For a specific visible input vector $v$, its free energy equals to the energy that a single configuration must own and it equals the sum of the probabilities of all the configurations containing $v$. In this study, the free energy (Hinton 2012) for a specific visible input vector $v$ can be computed as follows:
\[
F(v) = -\sum_i v_i c_i + \sum_j \log[1 + e^{x_j}], \tag{6}
\]

where $x_j = b_j + \sum_i v_i w_{ij}$.

For a given specific test vector $v$, after training the RBMs on a specific class $c$, the log probability that $RBMs$ assigns to $v$ can be computed according to the following formula:
\[
\log p(v|c) = -F_c(v) - \log PF_c,
\]

where the $PF_c$ is the partition function of $RBMs$. For a specific classification problem, if the total number of classes is small, there will be no difficulty for us to get the unknown log partition function. In this case, given a specific training set, we can just train a "soft-max" model to predict the label for a visible input vector $v$ resorting to the free energies of all the class-dependent RBMs:
\[
\log p(label = c|v) = \frac{e^{-F_{c}(v)} - \log PF_{c}}{\sum_{d} e^{-F_{d}(v)} - \log PF_{d}}. \tag{7}
\]

In the above formula Equation (7), all the partition functions $PF_{c}$’s can be learned by maximum likelihood (ML) training of the "soft-max" function, where the maximum likelihood method is a kind of parameter estimation method generally with the help of the log probability. Here, the "soft-max" function for a specific unit is generally defined in the following form:
\[
p_j = \frac{e^{x_j}}{\sum_{i=1}^{M} e^{x_i}},
\]

and the parameter $k$ means that there are totally $k$ different states that the unit can take on.

For clarity, we show the complete RBM algorithm in the following. The RBM algorithm as a whole based on the CD method can be summarized as follows:

- **Input**: a visible input vector $v$; the size of the hidden layer $n_h$; the learning rate $\eta$ and the maximum epoch $M$;
- **Output**: a weight matrix $W$, a biases vector for the hidden layer $b$ and a biases vector for the visible layer $c$;
- **Training**:
  - Initialization: Set the visible state with $v_1 = x$, and set $W$, $b$ and $c$ with small (random) values, For $t = 1, \ldots , M$,
  - For $j = 1, \ldots , n_h$, Compute the following value $p(h_j = 1|v_1) = \text{sigmoid}(b_j + \sum_i v_i W_{ij})$;
  - Sample $h_{ij}$ from the conditional distribution $P(h_{ij}|v_1)$ with Gibbs sampling method;
  - End
For $i = 1, 2, \ldots, n_v$, here, the $n_v$ is the size of the visible input vector $v$.
Compute the following value
\[ p(v_j = 1|h_i) = \text{sigmoid}(c_i + \sum_j W_{ij}h_j); \]
Sample $v_j$, from the conditional distribution $P(v_j|h_i)$ with Gibbs sampling method;
End
For $j = 1, \ldots, n_h$,
Compute the following value
\[ p(h_j = 1|v_i) = \text{sigmoid}(b_j + \sum_i v_iW_{ij}); \]
Update the parameters:
\[ W = W + \eta [P(h_i = 1|v_1)v_1 - P(h_i = 1|v_2)v_2]; \]
\[ c = c + \eta (v_1 - v_2); \]
\[ c = c + \eta [P(h_i = 1|v_1) - P(h_i = 1|v_2)]; \]
End

For classification, after training the RBM using the above algorithm, we need to compute the free energy function by Equation (6) and then we can assign a label for the sample $v$ with Equation (7).

4 Experiment

4.1 Data description

There have been a large amount of surveys in astronomy. SDSS is one of those surveys and it is one of the most not only ambitious but also influential ones (The official website of SDSS is http://www.sdss.org/). The SDSS has begun collecting data since 2000. From 2000 to 2008, the SDSS collected deep and multi-color images containing no less than a quarter of the sky and it also created 3d maps for over 930,000 galaxies and also for over 120,000 quasars. Data Release 7 (DR7) is the seventh major data release and it provides spectra and redshifts etc. for downloading.

All the data used in our experiment is coming from the SDSS. All the samples in the entire data set are divided into two classes, one class composed of non-CVs while the other class composed of CVs. There are totally 6818 non-CVs and 208 CVs in our data set. Each sample is composed of 3522 variables, or rather, spectral components. Among the total 6818 non-CVs, there are 1,559 belonging to Galaxies, 3,981 belonging to QSOs (Quasi-stellar objects) and redshifts etc. for downloading.

For clarity, we show the number of the CVs they found using the SDSS in Table 1. And the spectrum of a CV in our data set is shown in Figure 1.

| Paper                 | # of CVs |
|-----------------------|----------|
| Szkody et al. 2002    | 22       |
| Szkody et al. 2003    | 42       |
| Szkody et al. 2004    | 36       |
| Szkody et al. 2005    | 44       |
| Szkody et al. 2006    | 41       |
| Szkody et al. 2007    | 28       |

In our experiment, we chose randomly half of the whole data for training and the remaining half for testing for both non-CVs and CVs. In detail, for non-CVs, half of the total 6818 samples (i.e. 3414) were randomly chosen to train the RBM classifier and the remaining half to test the RBM classifier. Similarly, for CVs, half of all the 208 samples (i.e. 104) were randomly chosen to train the RBM classifier and the remaining half to test the RBM classifier. To explain it clearly, we showed the data used for training and testing the RBM classifier in the following table (Table 2).
Table 2: The number of the original data for training and for testing respectively, where the number 3522 is the dimension of the original data.

| CV/non-CV | Train/Test | #×Dim   |
|-----------|------------|---------|
| non-CV    | Train      | 3414×3522 |
| non-CV    | Test       | 3414×3522 |
| CV        | Train      | 104×3522  |
| CV        | Test       | 104×3522  |

4.2 Parameter chosen

In this subsection, we present the parameters in our experiment. We chose all the parameters referring to Hinton (2012). The learning rate in the process of updating was set to be 0.1. The momentum for smoothness and to prevent over-fitting was chosen to be 0.5. The maximum number of epochs was chosen to be 50. The weight decay factor, penalty, was chosen to be $2 \times 10^{-4}$. The initial weights were randomly generated from the standard normal distribution, while the biases vectors $b$ and $c$ were initialized with $0$. For clarity, we present them in the following table (Table 3).

| Parameter            | Value |
|----------------------|-------|
| learning rate        | 0.1   |
| momentum             | 0.5   |
| maximum epochs       | 50    |
| number of hidden units | 100   |
| initial biases vector | $0$   |

4.3 Experiment result

We first normalized the data to make it have unit $l_2$ norm, i.e., for a specific vector $x = [x_1, x_2, \ldots, x_n]$, the $l_2$ norm of the vector satisfies $\sum x_i^2 = 1$. Then we could get two matrices, one was $A = 6818 \times 3522$ and the other was $B = 208 \times 3522$. Then we found out the maximum element and the minimum element for CVs and non-CVs respectively. Finally, to apply binary RBM for classification, we found a parameter to assign the value of the variable in our experiment with 0 or 1, or rather, binarization.

Mathematically, if

$$S(i, j) - \min S(i, j) < \alpha (\max S(i, j) - \min S(i, j)),$$

then we set $S(i, j)$ with 0, otherwise we set $S(i, j)$ with 1. Here we used $S(i, j)$ (after binarization) to denote the element of the matrix $A$ and $B$ in the $i$th row and the $j$th column. The parameter $\alpha$ satisfied $0 < \alpha < 1$.

To investigate the influence of the parameter $\alpha$ on the final performance of the binary RBM algorithm, we first chose it to be 1/2 heuristically. Then we chose it to be 1/3. The experiment result shows that the classification accuracy is 100%, which is state-of-the-art and it outperforms the prevalent classifier SVM (Bu et al. 2013).

For clarity, we show the result in Table 4, in which we also show the performance the binary RBM algorithm based on other values for the variable $\alpha$. From Table 4 we can see that the classification accuracy is 97.2% when $\alpha = 1/2$. However, almost all of the CVs for testing is labeled as non-CVs.

Table 4 shows the classification accuracy computed by the following formula:

$$\text{Acc} = \frac{\sum \left| \hat{y} == y \right|}{\text{Card}(y)},$$

where $y$ is a vector denoting the label of all the test samples. In our experiment, there are 3413 (3409 non-CVs + 104 CVs) test samples. And “Card($y$)” represents the number of elements in vector $y$. In Equation (8), the denominator $\hat{y}$ is the label of all the test samples predicted by Equation (7), in which $c = +1$ or $c = -1$. In this paper, $c = +1$ means that the sample belongs to non-CVs, while $c = -1$ means that the sample belongs to CVs. And $\sum [\hat{y} == y]$ means the total number of equal elements in vector $y$ and vector $\hat{y}$.

5 Conclusion and future work

Restricted Boltzmann machine is a bipartite generative graphical model which can extract features representing the original data well. By introducing free energy and soft-max function, RBM can be used for classification. In this paper we apply restricted Boltzmann machine (RBM) for spectral classification of non-CVs and CVs. And the experiment result shows that the classification accuracy is 100%, which is state-of-the-art and it outperforms the prevalent classifier SVM (Bu et al. 2013).

For clarity, we show the result in Table 4, in which we also show the performance the binary RBM algorithm based on other values for the variable $\alpha$. From Table 4 we can see that the classification accuracy is 97.2% when $\alpha = 1/2$. However, almost all of the CVs for testing is labeled as non-CVs.

Table 4 shows the classification accuracy computed by the following formula:

$$\text{Acc} = \frac{\sum \left| \hat{y} == y \right|}{\text{Card}(y)},$$

where $y$ is a vector denoting the label of all the test samples. In our experiment, there are 3413 (3409 non-CVs + 104 CVs) test samples. And “Card($y$)” represents the number of elements in vector $y$. In Equation (8), the denominator $\hat{y}$ is the label of all the test samples predicted by Equation (7), in which $c = +1$ or $c = -1$. In this paper, $c = +1$ means that the sample belongs to non-CVs, while $c = -1$ means that the sample belongs to CVs. And $\sum [\hat{y} == y]$ means the total number of equal elements in vector $y$ and vector $\hat{y}$.
classification accuracy is 100 %, which is the state-of-the-art and outperforms the rather prevalent classifier, SVM.

Since RBM is the building block of deep belief nets (DBNs) and deep Boltzmann machine (DBM), then we can infer that deep Boltzmann machine (Salakhutdinov & Hinton 2009) and deep belief net can also perform well on spectral classification, which is our future work.

Acknowledgments

The authors are very grateful to the anonymous reviewer for a thorough reading, many valuable comments and rather helpful suggestions. The authors thank the editor Bryan Gaensler a lot for the helpful suggestions on the organization of the manuscript. The authors also thank Jiang Bin for providing the CV data.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. 1985, Cognitive science, 9, 147.

Andrieu et al. 2003, Machine learning, 50, 5.

Ball, N. M. et al. 2006, AJ, 650, 497.

Bazarghan, M. 2012, Astrophysics and Space Science, 337, 93.

Dahl et al. 2010, Advances in NIPS, 23, 469.

Daniel et al. 2011, AJ, 142, 203.

Gelfand, A. E. 2000. Journal of the American Statistical Association, 95, 1300.

Greenwell, R. N., Ritchey, N. P., & Lial, M. L. 2003, Calculus with applications for the life sciences (Addison Wesley)

Gunawardana, A., & Meek, C. 2008, in Proceedings of the 2008 ACM conference on Recommender systems, ACM, 19.

Hinton, G. E. 2012, in Neural Networks: Tricks of the Trade, ed. Editors by Grégoire Montavon, Geneviève B. Orr, Klaus-Robert Müller (Springer Berlin Heidelberg), 599

Hinton, G. E., Osindero, S., & Teh, Y. W. 2006, Neural computation, 18, 1527.

Hinton, G. E., & Salakhutdinov, R. R. 2006, Science, 313, 504.

Muno M. P. et al. 2009, ApJS, 181, 110.

McGurk, R. C., Kimball, A. E., & Ivezić, Ž. 2010, AJ, 139, 1261.

Navarro S. G., Corradi R. L. M. & Mampaso A. 2012, A & A, 538, A76.

Salakhutdinov, R., & Hinton, G. E. 2009, in AISTATAS, Vol. 5, Cambridge, MA: MIT Press, 448.

Salakhutdinov, R., Mnih, A., & Hinton, G. 2007, in Proceedings of the 24th ICML, ACM, 791.

Sarty G. & Wu, K. 2006, PASA, 23, 106

Schluter, J., & Osendorfer, C. 2011, in ICMLA, IEEE, 118.

Singh, H. P., Gulati, R. K., & Gupta, R. 1998, MN-RAS, 295, 312.

Szokody, P. et al. 2002. AJ, 123, 430.

Szokody, P. et al. 2003, AJ, 583, 430.

Szokody, P. et al. 2004, AJ, 128, 1882.

Szokody, P. et al. 2005, AJ, 129, 2386.

Szokody, P. et al. 2006, AJ, 131, 973.

Szokody, P. et al. 2007, AJ, 134, 185.

Tang, Y., Salakhutdinov, R., & Hinton, G. 2012, in CVPR, IEEE, 2264.

Taylor, G. W., & Hinton, G. E. 2009, in ICML, ACM, 1025.

Warner, B. 2003, in Cataclysmic variable stars (Cambridge: Cambridge University Press)

Williams, G. W. 1983, ApJS, 53, 523

Wu, K. 2000, Sp. Sci. Rev., 93, 611.

Xiong, Z., Jiang, W., & Wang, G. 2012, in Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on, 81, 11, ed. Editors byGeyong Min, Yueqi Wu, Lei (Chris) Liu, Xiaolong Jin, Stephen Jarvis, Ahmed Y. Al-Dubai (Washington, DC: IEEE Computer Society), 640

---

Table 4: The classification accuracy with different $\alpha$’s.

| $\alpha$ | Accuracy |
|----------|----------|
| 1/5      | 97%      |
| 2/5      | 100%     |
| 3/5      | 97%      |
| 4/5      | 97%      |
| 1/4      | 97%      |
| 3/4      | 97%      |
| 1/3      | 100%     |
| 2/3      | 97%      |
Yang, N., Tang, C., Wang, Y., Tang, R., Li, C., Zheng, J., & Zhu, J. (2009). in Advances in Data and Web Management, 5446, Lecture Notes in Computer Science ed. Editors by Qing Li, Ling Feng, Jian Pei, Sean X. Wang, Xiaofang Zhou, Qiao-Ming Zhu, Springer Berlin Heidelberg. 297

Yude, B., Jingchang, P., Bin, J., Fuqiang, C., & Peng, W. 2013, PASA, 30, e24.

Zhang Y. & Zhao Y. 2004, A & A, 422, 1113.