A Neural Network Approach to Ordinal Regression

Jianlin Cheng, Zheng Wang, and Gianluca Pollastri

Abstract— **Ordinal regression is an important type of learning, which has properties of both classification and regression. Here we describe an effective approach to adapt a traditional neural network to learn ordinal categories. Our approach is a generalization of the perceptron method for ordinal regression. On several benchmark datasets, our method (NNRank) outperforms a neural network classification method. Compared with the ordinal regression methods using Gaussian processes and support vector machines, NNRank achieves comparable performance. Moreover, NNRank has the advantages of traditional neural networks: learning in both online and batch modes, handling very large training datasets, and making rapid predictions. These features make NNRank a useful and complementary tool for large-scale data mining tasks such as information retrieval, web page ranking, collaborative filtering, and protein ranking in Bioinformatics. The neural network software is available at: http://www.cs.missouri.edu/~chengji/cheng_software.html.**

I. Introduction

Ordinal regression (or ranking learning) is an important supervised problem of learning a ranking or ordering on instances, which has the property of both classification and metric regression. The learning task of ordinal regression is to assign data points into a set of finite ordered categories. For example, a teacher rates students' performance using A, B, C, D, and E (A $>$ B $>$ C $>$ D $>$ E) [9]. Ordinal regression is different from classification due to the order of categories. In contrast to metric regression, the response variables (categories) in ordinal regression is discrete and finite.

The research of ordinal regression dates back to the ordinal statistics methods in 1980s [28], [29] and machine learning research in 1990s [7], [20], [13]. It has attracted the considerable attention in recent years due to its potential applications in many data-intensive domains such as information retrieval [20], web page ranking [24], collaborative filtering [18], [3], [41], image retrieval [40], and protein ranking [8] in Bioinformatics.

A number of machine learning methods have been developed or redesigned to address ordinal regression problem [33], including perceptron [14] and its kernelized generalization [3], neural network with gradient descent [7], [5], Gaussian process [10], [9], [37], large margin classifier (or support vector machine) [21], [22], [24], [38], [11], [2], [12], k-partite classifier [1], boosting algorithm [17], [15], constraint classification [19], regression trees [25], Naive Bayes [42], Bayesian hierarchical experts [32], binary classification approach [16], [26] that decomposes the original ordinal regression problem into a set of binary classifications, and the optimization of nonsmooth cost functions [6].

Most of these methods can be roughly classified into two categories: pairwise constraint approach [22], [24], [15], [5] and multi-threshold approach [14], [38], [9]. The former is to convert the full ranking relation into pairwise order constraints. The latter tries to learn multiple thresholds to divide data into ordinal categories. Multi-threshold approaches also can be unified under the general, extended binary classification framework [26].

The ordinal regression methods have different advantages and disadvantages. Prank [14], a perceptron approach that generalizes the binary perceptron algorithm to the ordinal multi-class situation, is a fast online algorithm. However, like a standard perceptron method, its accuracy suffers when dealing with non-linear data, while a quadratic kernel version of Prank greatly relieves this problem. One class of accurate large-margin classifier approaches [22], [24] convert the ordinal relations into $O(n^2)$ (n: the number of data points) pairwise ranking constraints for the structural risk minimization [39], [36]. Thus, it can not be applied to medium size datasets ($>$ 10,000 data points), without discarding some pairwise preference relations. It may also overfit noise due to incomparable pairs.

The other class of powerful large-margin classifier methods [38], [11] generalize the support vector formulation for ordinal regression by finding $K-1$ thresholds on the real line that divide data into K ordered categories. The size of this optimization problem is linear in the number of training examples. However, like support vector machine used for classification, the prediction speed is slow when the solution is not sparse, which makes it not appropriate for time-critical tasks. Similarly, another state-of-the-art approach, Gaussian process method [9], also has the difficulty of handling large training datasets and the problem of slow prediction speed in some situations.

Here we describe a new neural network approach for ordinal regression that has the advantages of neural network learning: learning in both online and batch mode, training on very large dataset [5], handling non-linear data, good performance, and rapid prediction. Our method can be considered a generalization of the perceptron learning [14] into multilayer perceptrons (neural network) for ordinal regression. Our method is also related to the classic generalized linear models (e.g., cumulative logit model) for ordinal regression [28]. Unlike the neural network method [5] trained on pairs of examples to learn pairwise order relations, our method works on individual data points and uses multiple output nodes to

Jianlin Cheng and Zheng Wang are with the computer science department and informatics institute, University of Missouri, Columbia, MO 65211, USA (email: chengji@missouri.edu, zwyw6@missouri.edu); Gianluca Pollastri is with the school of computer science and informatics, University of College Dublin, Belfield, Dublin 4, Ireland (email: gianluca.pollastri@ucd.ie).

978-1-4244-1821-3/08/$25.00

estimate the probabilities of ordinal categories. Thus, our method falls into the category of multi-threshold approach. The learning of our method proceeds similarly as traditional neural networks using back-propagation [35].

On the same benchmark datasets, our method yields the performance better than the standard classification neural networks and comparable to the state-of-the-art methods using support vector machines and Gaussian processes. In addition, our method can learn on very large datasets and make rapid predictions.

II. METHOD

A. Formulation

Let D represent an ordinal regression dataset consisting of n data points (x, y) , where $x \in R^d$ is an input feature vector and y is its ordinal category from a finite set Y. Without loss of generality, we assume that $Y = 1, 2, ..., K$ with "$<$" as order relation.

For a standard classification neural network without considering the order of categories, the goal is to predict the probability of a data point x belonging to one category k $(y = k)$. The input is x and the target of encoding the category k is a vector $t = (0, ..., 0, 1, 0, ..., 0)$, where only the element t_k is set to 1 and all others to 0. The goal is to learn a function to map input vector x to a probability distribution vector $o = (o_1, o_2, ...o_k, ...o_K)$, where o_k is closer to 1 and other elements are close to zero, subject to the constraint $\sum_{i=1}^{K} o_i = 1$.

In contrast, like the perceptron approach [14], our neural network approach considers the order of the categories. If a data point x belongs to category k, it is classified automatically into lower-order categories $(1, 2, ..., k-1)$ as well. So the target vector of x is $t = (1, 1, .., 1, 0, 0, 0)$, where t_i $(1 \leq i \leq k)$ is set to 1 and other elements zeros, as shown in Figure 1. Thus, the goal is to learn a function to map the input vector x to a probability vector $o = (o_1, o_2, ..., o_k, ...o_K)$, where o_i $(i \leq k)$ is close to 1 and o_i $(i \geq k)$ is close to 0. $\sum_{i=1}^{K} o_i$ is the estimate of number of categories (i.e. k) that x belongs to, instead of 1. The formulation of the target vector is similar to the perceptron approach [14]. It is also related to the classical cumulative probit model for ordinal regression [28], in the sense that we can consider the output probability vector $(o_1, ...o_k, ...o_K)$ as a cumulative probability distribution on categories $(1, ..., k, ..., K)$, i.e., $\frac{\sum_{i=1}^{K} o_i}{K}$ is the proportion of categories that x belongs to, starting from category 1.

The target encoding scheme of our method is related to but, different from multi-label learning [4] and multiple label learning [23] because our method imposes an order on the labels (or categories).

B. Learning

Under the formulation, we can use the almost exactly same neural network machinery for ordinal regression. We construct a multi-layer neural network to learn ordinal relations from D. The neural network has d inputs corresponding to the number of dimensions of input feature vector x and K output nodes corresponding to K ordinal categories. There can be one or more hidden layers. Without loss of generality, we use one hidden layer to construct a standard two-layer feedforward neural network. Like a standard neural network for classification, input nodes are fully connected with hidden nodes, which in turn are fully connected with output nodes. Likewise, the transfer function of hidden nodes can be linear function, sigmoid function, and tanh function that is used in our experiment. The only difference from traditional neural network lies in the output layer. Traditional neural networks use softmax $\frac{e^{-z_i}}{\sum_{i=1}^{K} e^{-z_i}}$(or normalized exponential function) for output nodes, satisfying the constraint that the sum of outputs $\sum_{i=1}^{K} o_i$ is 1. z_i is the net input to the output node O_i.

In contrast, each output node O_i of our neural network uses a standard sigmoid function $\frac{1}{1+e^{-z_i}}$, without including the outputs from other nodes, as shown in Figure 1. Output node O_i is used to estimate the probability o_i that a data point belongs to category i independently, without subjecting to normalization as traditional neural networks do. Thus, for a data point x of category k, the target vector is $(1, , 1, .., 1, 0, 0, 0)$, in which the first k elements is 1 and others 0. This sets the target value of output nodes O_i $(i \leq k)$ to 1 and O_i $(i > k)$ to 0. The targets instruct the neural network to adjust weights to produce probability outputs as close as possible to the target vector. It is worth pointing out that using independent sigmoid functions for output nodes does not guaranteed the monotonic relation $(o_1 >= o_2 >= ... >= o_K)$, which is not necessary but, desirable for making predictions [26]. A more sophisticated approach is to impose the inequality constraints on the outputs to improve the performance.

Training of the neural network for ordinal regression proceeds very similarly as standard neural networks. The cost function for a data point x can be relative entropy or square error between the target vector and the output vector. For relative entropy, the cost function for output nodes is $f_c = \sum_{i=1}^{K} (t_i \log o_i + (1 - t_i) \log(1 - o_i))$. For square error, the error function is $f_c = \sum_{i=1}^{K} (t_i - o_i)^2$. Previous studies [34] on neural network cost functions show that relative entropy and square error functions usually yield very similar results. In our experiments, we use square error function and standard back-propagation to train the neural network. The errors are propagated back to output nodes, and from output nodes to hidden nodes, and finally to input nodes.

Since the transfer function f_t of output node O_i is the independent sigmoid function $\frac{1}{1+e^{-z_i}}$, the derivative of f_t of output node O_i is $\frac{\partial f_t}{\partial z_i} = \frac{e^{-z_i}}{(1+e^{-z_i})^2} = \frac{1}{1+e^{-z_i}}(1 - \frac{1}{1+e^{-z_i}})$ $= o_i(1 - o_i)$. Thus, the net error propagated to output node O_i is $\frac{\partial f_c}{\partial o_i}\frac{\partial f_t}{\partial z_i} = \frac{t_i - o_i}{o_i(1-o_i)} \times o_i(1 - o_i) = t_i - o_i$ for relative entropy cost function, $\frac{\partial f_c}{\partial o_i}\frac{\partial f_t}{\partial z_i} = -2(t_i - o_i) \times o_i(1 - o_i) = -2o_i(t_i - o_i)(1 - o_i)$ for square error cost function. The net errors are propagated through neural networks to adjust weights using gradient descent as traditional neural networks

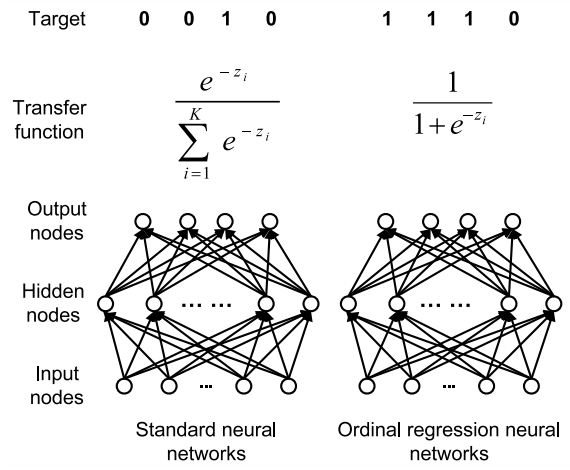

Fig. 1. Comparison between a standard classification neural network and an ordinal regression neural network. Without loss of generality, the neural networks are assumed to have one hidden layer and one output layer with four output nodes. For a data point in category three, the target vector of the standard neural network is (0, 0, 1, 0), while the target vector of the ordinal regression neural network is (1, 1, 1, 0). The transfer function of output node i of the standard neural network is the normalized exponential function $\frac{e^{-z_i}}{\sum_{i=1}^{K} e^{-z_i}}$. In contrast, the ordinal regression neural network uses the sigmoid function $\frac{1}{1+e^{-z_i}}$.

do.

Despite the small difference in the transfer function and the computation of its derivative, the training of our method is the same as traditional neural networks. The network can be trained on data in the online mode where weights are updated per example, or in the batch mode where weights are updated per bunch of examples.

C. Prediction

In the test phase, to make a prediction, our method scans output nodes in the order $O_1, O_2, ..., O_K$. It stops when the output of a node is smaller than the predefined threshold T (e.g., 0.5) or no nodes left. The index k of the last node O_k whose output is bigger than T is the predicted category of the data point.

III. Experiments and Results

A. Standard Benchmark Data and Evaluation Metric

We test our method on eight benchmark datasets for ordinal regression [9]. The eight datasets (Diabetes, Pyrimidines, Triazines, Machine CUP, Auto MPG, Boston, Stocks Domain, and Abalone) are originally used for metric regression. Chu and Ghahramani [9] discretized the real-value targets into five equal intervals, corresponding to five ordinal categories. The authors randomly split each dataset into training/test datasets and repeated the partition 20 times independently. We use the exactly same partitions as in [9] to train and test our method.

We use the online mode to train neural networks. The parameters to tune are the number of hidden units, the number of epochs, and the learning rate. We create a grid for these three parameters, where the hidden unit number is in the range $[1..15]$, the epoch number in the set $(50, 200, 500, 1000, 1500, 2000)$, and the initial learning rate in the range $[0.01..0.5]$. During the training, the learning rate is halved if training errors continuously go up for a pre-defined number (40, 60, 80, or 100) of epochs. For experiments on each data split, the neural network parameters are *fully* optimized on the training data without using any test data.

For each experiment, after the parameters are optimized on the training data, we train five models on the training data with the optimal parameters, starting from different initial weights. The ensemble of five trained models are then used to estimate the generalized performance on the test data. That is, the average output of five neural network models is used to make predictions.

We evaluate our method using zero-one error and mean absolute error as in [9]. Zero-one error is the percentage of wrong assignments of ordinal categories. Mean absolute error is the root mean square difference between assigned categories (k') and true categories (k) of all data points. For each dataset, the training and evaluation process is repeated 20 times on 20 data splits. Thus, we compute the average error and the standard deviation of the two metrics as in [9].

B. Comparison with Neural Network Classification on Standard Bencharmks

We first compare our method (NNRank) with a standard neural network classification method (NNClass). We implement both NNRank and NNClass using C++. NNRank and NNClass share most code with minor difference in the transfer function of output nodes and its derivative computation as described in Section II-B.

As Table I shows, NNRank outperforms NNClass in all but one case in terms of both the mean-zero error and the mean absolute error. And on some datasets the improvement of NNRank over NNClass is sizable. For instance, on the Stock and Pyrimidines datasets, the mean zero-one error of NNRank is about 4% less than NNClass; on four datasets (Stock, Pyrimidines, Triazines, and Diabetes) the mean absolute error is reduced by about .05. The results show that the ordinal regression neural network consistently achieves the better performance than the standard classification neural network.

C. Comparison with Neural Network Classification on a Protein Similarity Dataset

To further verify the effectiveness of neural network ordinal regression approach, we evaluate NNRank and NNClass on a protein similarity dataset - a real ordinal regression dataset used in protein fold recognition [8]. Each data point in the dataset corresponds to a protein pair (a query protein and a template protein). The data points in the dataset are classified into three ordinal similarity categories (*fold* $<$ *super family* $<$ *family*), depending on the similarity levels

of the protein pairs. The category of each data point was assigned by biologists [31].

A data point representing a query-template protein pair is labeled as *fold* if the two proteins have similar tertiary structures but do not have evolutionary relationship; *super family* if they have similar structures and weak revolutionary relationship; and *family* if they have similar structures and strong revolutionary relationship. Each data point has 62 features, corresponding to specific criteria used to measure the similarities between a query and a template protein.

The data points are splitted into a training dataset consisting of 6018 data points (2910 in fold, 1810 in super family, and 1298 in family) and a test dataset containing 747 data points (395 in fold, 166 in super family, and 186 in family). Both NNRank and NNClass are trained on the training dataset and evaluated on the test dataset.

The mean zero one error and mean absolute error of NNRank are 23.96% and 0.258, respectively. The mean zero one error and mean absolute error of NNClass are 25.03% and 0.277, respectively. The mean zero one error of NNRank is 1.1% lower than NNClass. The mean absolute error of NNRank is 0.019 less than NNClass. The experiment shows that NNRank performs better than NNClass on a large, real ordinal regression dataset.

D. Comparison with Gaussian Processes and Support Vector Machines on Standard Benchmarks

To further evaluate the performance of our method, we compare NNRank with two Gaussian process methods (GP-MAP and GP-EP) [9] and a support vector machine method (SVM) [38] implemented in [9]. The results of the three methods are quoted from [9]. Table II reports the zero-one error on the eight datasets. NNRank achieves the best results on Diabetes, Triazines, and Abalone, GP-EP on Pyrimidines, Auto MPG, and Boston, GP-MAP on Machine, and SVM on Stocks.

Table III reports the mean absolute error on the eight datasets. NNRank yields the best results on Diabetes and Abalone, GP-EP on Pyrimidines, Auto MPG, and Boston, GP-MAP on Triazines and Machine, SVM on Stocks.

In summary, on the eight datasets, the performance of NNRank is comparable to the three state-of-the-art methods for ordinal regression.

IV. Discussion and Future Work

We have described a novel approach to adapt traditional neural networks for ordinal regression. Our neural network approach can be considered a generalization of one-layer perceptron approach [14] into multi-layer. On the standard benchmark of ordinal regression, our method outperforms standard neural networks used for classification. Furthermore, on the same benchmark, our method achieves the similar performance as the two state-of-the-art methods (support vector machines and Gaussian processes) for ordinal regression.

Compared with existing methods for ordinal regression, our method has several advantages of neural networks. First, like the perceptron approach [14], our method can learn in both batch and online mode. The online learning ability makes our method a good tool for adaptive learning in the real-time. The multi-layer structure of neural network and the non-linear transfer function give our method the stronger fitting ability than perceptron methods.

Second, the neural network can be trained on very large datasets iteratively, while training is more complex than support vector machines and Gaussian processes. Since the training process of our method is the same as traditional neural networks, average neural network users can use this method for their tasks.

Third, neural network method can make rapid prediction once models are trained. The ability of learning on very large dataset and predicting in time makes our method a useful and competitive tool for ordinal regression tasks, particularly for time-critical and large-scale ranking problems in information retrieval, web page ranking, collaborative filtering, and the emerging fields of Bioinformatics. To facilitate the application of this new approach, we make both NNRank and NNClass to accept a general input format and freely available to the community at http://www.cs.missouri.edu/~chengji/cheng_software.html.

There are some directions to further improve the neural network (or multi-layer perceptron) approach for ordinal regression. One direction is to design a transfer function to ensure the monotonic decrease of the outputs of the neural network; the other direction is to derive the general error bounds of the method under the binary classification framework [26]. Furthermore, the other flavors of implementations of the multi-threshold multi-layer perceptron approach for ordinal regression are possible. Since machine learning ranking is a fundamental problem that has wide applications in many diverse domains such as web page ranking, information retrieval, image retrieval, collaborative filtering, bioinformatics and so on, we believe the further exploration of the neural network (or multi-layer perceptron) approach for ranking and ordinal regression is worthwhile.

References

[1] S. Agarwal and D. Roth. Learnability of bipartite ranking functions. In *Proc. of the 18th annual conference on learning theory (COLT-05)*. 2005.

[2] F. Aiolli and A. Sperduti. Learning preferences for multiclass problems. In *Advances in Neural Information Processing Systems 17 (NIPS)*. 2004.

[3] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on machine learning (ICML)*, page 9. ACM press, New York, USA, 2004.

[4] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, USA, 1996.

[5] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. of Internaltional Conference on Machine Learning (ICML-05)*, pages 89–97. 2005.

[6] C.J.C. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. In *Advances in Neural Information Processing Systems (NIPS) 20*. MIT press, Cambridge, MA, 2006.

[7] R. Caruana, S. Baluja, and T. Mitchell. Using the future to sort out the present: Rankprop and multitask learning for medical risk evaluation. In *Advances in neural information processing systems 8 (NIPS)*. 1996.

[8] J. Cheng and P. Baldi. A machine learning information retrieval approach to protein fold recognition. *Bioinformatics*, 22:1456–1463, 2006.

[9] W. Chu and Z. Ghahramani. Gaussian processes for ordinal regression. *Journal of Machine Learning Research*, 6:1019–1041, 2005.

[10] W. Chu and Z. Ghahramani. Preference learning with Gaussian processes. In *Proc. of International Conference on Machine Learning (ICML-05)*, pages 137–144. 2005.

[11] W. Chu and S.S. Keerthi. New approaches to support vector ordinal regression. In *Proc. of International Conference on Machine Learning (ICML-05)*, pages 145–152. 2005.

[12] W. Chu and S.S. Keerthi. Support vector ordinal regression. *Neural Computation*, 19(3), 2007.

[13] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. *Journal of Artificial Intelligence Research*, 10:243–270, 1999.

[14] K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS) 14*, pages 641–647. MIT press, Cambridge, MA, 2002.

[15] O. Dekel, J. Keshet, and Y. Singer. Log-linear models for label ranking. In *Proc. of the 21st international conference on machine learning (ICML-06)*, pages 209–216. 2004.

[16] E. Frank and M. Hall. A simple approach to ordinal classification. In *Proc. of the European Conference on Machine Learning*. 2001.

[17] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[18] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:61–70, 1992.

[19] S. Har-Peled, D. Roth, and D. Zimak. Constraint classification: a new approach to multiclass classification and ranking. In *Advances in Neural Information Processing Systems 15 (NIPS)*. 2002.

[20] R. Herbrich, T. Graepel, P. Bollmann-Sdorra, and K. Obermayer. Learning preference relations for information retrieval. In *Proc. of ICML workshop on text categorization and machine learning*, pages 80–84. 1998.

[21] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. In *Proc. of 9th International Conference on Artificial Neural Networks (ICANN)*, pages 97–102. 1999.

[22] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In A. J. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[23] R. Jin and Z. Ghahramani. Learning with multiple labels. In *Advances in Neural Information Processing Systems (NIPS) 15*. MIT press, Cambridge, MA, 2003.

[24] I. Joachims. Optimizing search engines using clickthrough data. In David Hand, Daniel Keim, and Raymond NG, editors, *Proc. of 8th ACM SIGKDD International conference on knowledge discovery and data mining*, pages 133–142. 2002.

[25] S. Kramer, G. Widmer, B. Pfahringer, and M. DeGroeve. Prediction of ordinal classes using regression trees. *Fundamenta Informaticae*, 47:1–13, 2001.

[26] L. Li and H. Lin. Ordinal regression by extended binary classification. In *Advances in Neural Information Processing Systems (NIPS) 20*. MIT press, Cambridge, MA, 2006.

[27] D. J. C. MacKay. A practical bayesian framework for back propagation networks. *Neural Computation*, 4:448–472, 1992.

[28] P. McCullagh. Regression models for ordinal data. *Journal of the Royal Statistical Society B*, 42:109–142, 1980.

[29] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, 1983.

[30] T. P. Minka. A family of algorithms for approximate bayesian inference. *PhD Thesis, Massachusetts Institute of Technology*, 2001.

[31] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[32] U. Paquet, S. Holden, and A. Naish-Guzman. Bayesian hierarchical ordinal regression. In *Proc. of the international conference on artifical neural networks*. 2005.

[33] S. Rajaram, A. Garg, X.S. Zhou, and T.S. Huang. Classification approach towards ranking and sorting problems. In *Machine Learning: ECML 2003, vol. 2837 of Lecture Notes in Artificail Intelligence (N. Lavrac, D. gamberger, H. Blockeel and L. Todorovski eds.)*, pages 301–312. Springer-Verlag, 2003.

[34] M.D. Richard and R.P. Lippman. Neural network classifiers estimate bayesian a-posteriori probabilities. *Neural Computation*, 3:461–483, 1991.

[35] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning Internal Representations by Error Propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, pages 318–362. Bradford Books/MIT Press, Cambridge, MA., 1986.

[36] B. Schölkopf and A.J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*. MIT University Press, Cambridge, MA, 2002.

[37] A. Schwaighofer, V. Tresp, and K. Yu. Hiearachical bayesian modelling with gaussian processes. In *Advances in Neural Information Processing Systems 17 (NIPS)*. MIT press, 2005.

[38] A. Shashua and A. Levin. Ranking with large margin principle: two approaches. In *Advances in Neural Information Processing Systems 15 (NIPS)*. 2003.

[39] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, Berlin, Germany, 1995.

[40] H. Wu, H. Lu, and S. Ma. A practical svm-based algorithm for ordinal regression in image retrieval. pages 612–621, 2003.

[41] S. Yu, K. Yu, V. Tresp, and H. P. Kriegel. Collaborative ordinal regression. In *Proc. of 23rd international conference on machine learning*, pages 1089–1096. 2006.

[42] H. Zhang, L. Jiang, and J. Su. Augmenting naive bayes for ranking. In *International Conference on Machine Learning (ICML-05)*. 2005.

	Mean zero-one error		Mean absolute error	
Dataset	NNRank	NNClass	NNRank	NNClass
Stocks	12.68±1.8%	16.97± 2.3%	0.127±0.01	0.173±0.02
Pyrimidines	37.71±8.1%	41.87±7.9%	0.450±0.09	0.508±0.11
Auto MPG	27.13±2.0%	28.82±2.7%	0.281±0.02	0.307±0.03
Machine	17.03±4.2%	17.80±4.4%	0.186±0.04	0.192±0.06
Abalone	21.39±0.3%	21.74± 0.4%	0.226±0.01	0.232±0.01
Triazines	52.55±5.0%	52.84±5.9%	0.730±0.06	0.790±0.09
Boston	26.38±3.0%	26.62±2.7%	0.295±0.03	0.297±0.03
Diabetes	44.90±12.5%	43.84±10.0%	0.546±0.15	0.592±0.09

TABLE I

THE RESULTS OF NNRANK AND NNCLASS ON THE EIGHT DATASETS. THE RESULTS ARE THE AVERAGE ERROR OVER 20 TRIALS ALONG WITH THE STANDARD DEVIATION.

Data	NNRank	SVM	GP-MAP	GP-EP
Triazines	**52.55±5.0%**	54.19±1.5%	52.91±2.2%	52.62±2.7%
Pyrimidines	37.71±8.1%	41.46±8.5%	39.79±7.2%	**36.46±6.5%**
Diabetes	**44.90±12.5%**	57.31±12.1%	54.23±13.8%	54.23±13.8%
Machine	17.03±4.2%	17.37±3.6%	**16.53±3.6%**	16.78±3.9%
Auto MPG	27.13±2.0%	25.73±2.2%	23.78±1.9%	**23.75±1.7%**
Boston	26.38±3.0%	25.56±2.0%	24.88±2.0%	**24.49±1.9%**
Stocks	12.68±1.8%	**10.81±1.7%**	11.99±2.3%	12.00±2.1%
Abalone	**21.39±0.3%**	21.58±0.3%	21.50±0.2%	21.56±0.4%

TABLE II

ZERO-ONE ERROR OF NNRANK, SVM, GP-MAP, AND GP-EP ON THE EIGHT DATASETS. SVM DENOTES THE SUPPORT VECTOR MACHINE METHOD [38], [9]. GP-MAP AND GP-EP ARE TWO GAUSSIAN PROCESS METHODS USING LAPLACE APPROXIMATION [27] AND EXPECTATION PROPAGATION [30] RESPECTIVELY [9]. THE RESULTS ARE THE AVERAGE ERROR OVER 20 TRIALS ALONG WITH THE STANDARD DEVIATION. WE USE BOLDFACE TO DENOTE THE BEST RESULTS.

Data	NNRank	SVM	GP-MAP	GP-EP
Triazines	0.730±0.07	0.698±0.03	**0.687±0.02**	0.688±0.03
Pyrimidines	0.450±0.10	0.450±0.11	0.427±0.09	**0.392±0.07**
Diabetes	**0.546±0.15**	0.746±0.14	0.662±0.14	0.665±0.14
Machine	0.186±0.04	0.192±0.04	**0.185±0.04**	0.186±0.04
Auto MPG	0.281±0.02	0.260±0.02	0.241±0.02	**0.241±0.02**
Boston	0.295±0.04	0.267±0.02	0.260±0.02	**0.259±0.02**
Stocks	0.127±0.02	**0.108±0.02**	0.120±0.02	0.120±0.02
Abalone	**0.226±0.01**	0.229±0.01	0.232±0.01	0.234±0.01

TABLE III

MEAN ABSOLUTE ERROR OF NNRANK, SVM, GP-MAP, AND GP-EP ON THE EIGHT DATASETS. SVM DENOTES THE SUPPORT VECTOR MACHINE METHOD [38], [9]. GP-MAP AND GP-EP ARE TWO GAUSSIAN PROCESS METHODS USING LAPLACE APPROXIMATION AND EXPECTATION PROPAGATION RESPECTIVELY [9]. THE RESULTS ARE THE AVERAGE ERROR OVER 20 TRIALS ALONG WITH THE STANDARD DEVIATION. WE USE BOLDFACE TO DENOTE THE BEST RESULTS.