Low n-Rank Tensor Log-Linear Models for Classification

Caleb Nelson, Yulo Leake and Brian Hutchinson
Computer Science Department
Western Washington University
Bellingham, Washington

***Abstract*—Despite the wide array of powerful classification techniques available, simple linear and quadratic models remain important tools in a researcher's toolkit, particularly for problems with relatively little training data. This paper introduces a classification model that allows careful control of model complexity, allowing intermediate levels of expressiveness between linear and quadratic models. Our model is parameterized by a third order tensor that multilinearly maps from input to output. By controlling the n-rank of this weight tensor, we are able to control the complexity of our model and therefore strike a suitable balance between bias and variance. Two variants of our model are proposed. The first induces the low n-rank structure in the weight tensor via tensor nuclear norm regularization, yielding a convex training procedure. The second explicitly constrains the n-rank by parameterizing the tensor in terms of its Tucker decomposition. Experiments are conducted on four standard classification tasks, chosen to offer a variety of training set sizes, feature dimensionality and number of classes. We find that that the second model variant matches or outperforms each of three baselines (majority class, linear classifier, quadratic classifier) on all four tasks: it decreases relative classification error by as much as 15.0% over a linear classifier, 31.6% over a quadratic classifier and 28.9% over the majority class baseline. We also show that despite the non-convexity in training the second variant, it consistently outperforms the first.**

I. Introduction

Navigating the bias-variance trade-off is one of the fundamental challenges of machine learning. To obtain performance that generalizes to unseen data, a machine learning model must be expressive enough to capture the true underlying patterns in the data, but not so expressive that it overfits to the noise present in the finite training set sample. Countless machine learning models, training algorithms, regularization schemes and other assorted techniques have been developed to empower machine learning practitioners to find the balance that is most suitable for their tasks.

This paper introduces the **Te**nsor **L**og-**L**ine**ar** (TELLAR) model for classification, parameterized by a third order weight tensor, $\mathcal{W}$. The TELLAR model multilinearly maps from the input $x \in \mathbb{R}^D$ to a posterior probability vector $y \in \mathbb{R}^C$. In doing so, it captures second order information in the input vector x, enabling it to learn non-linear, quadratic-like decision boundaries. Model complexity can be controlled by manipulating the n-rank of $\mathcal{W}$: model expressiveness decreases as the n-rank decreases. Through the n-rank, we can effectively interpolate between the expressiveness of quadratic and linear models. Additionally, we show that reducing $\mathcal{W}$'s n-rank has an intuitive interpretation; namely, it implicitly learns a dimensionality-reducing feature transformation of x, a low-dimensional continuous distributed representation of the classes, and how the reduced features map to classes via their low-dimensional representations.

Two variants of the TELLAR model are introduced in this paper. The first promotes low n-rank via tensor nuclear norm regularization. The tensor nuclear norm, although non-smooth, is convex, and so training the model is convex optimization problem and can be solved using proximal gradient descent. The second constrains the tensor's n-rank by explicitly parameterizing the model in terms of a Tucker decomposition of $\mathcal{W}$ [1]. This second approach forfeits convexity, but offers substantial improvements in space and time complexity.

II. Background

A. Tensors

Tensors are multidimensional arrays. A vector $w \in \mathbb{R}^I$ is a first order tensor while a matrix $W \in \mathbb{R}^{I \times J}$ is a second order tensor. Third order tensors $\mathcal{W} \in \mathbb{R}^{I \times J \times K}$ are indexed by three indices, and so forth. Let $\mathcal{W}_{\cdot jk}$ denote the vector obtained by fixing the second and third modes of the tensor to j and k, respectively, and allowing the first dimension to vary. $\mathcal{W}_{\cdot jk}$ is referred to as a *mode-1 fiber*. The set of mode-1 fibers are $\{\mathcal{W}_{\cdot jk} : 1 \leq j \leq K, 1 \leq k \leq K\}$. The sets of mode-2 and mode-3 fibers are defined analogously. One can *unfold* a tensor along a mode n into a matrix by taking all of the mode-n fibers and arranging them as columns of a matrix, the mode-n unfolding of $\mathcal{W}$ is denoted $\mathcal{W}_{<n>}$. A tensor can be multiplied along the nth mode by a matrix W to produce a new tensor, $\mathcal{Y}$: $\mathcal{Y} = \mathcal{W} \times_n W$. This operation involves multiplying all of the mode-n fibers by the matrix W. For example, multiplying $\mathcal{W} \in \mathbb{R}^{I \times J \times K}$ along the first mode with matrix $W \in \mathbb{R}^{I \times L}$ yields $\mathcal{Y} \in \mathbb{R}^{L \times J \times K}$, where $\mathcal{Y}_{\ell jk} = \sum_{i=1}^{I} \mathcal{W}_{ijk} W_{i\ell}$.

While matrix rank is a familiar concept, there are two distinct notions of rank when generalizing to higher order tensors. The *rank* of a tensor $\mathcal{W}$ is minimum the number of rank-1 tensors into which it can be decomposed. This notion of rank can be difficult to work with; e.g., even determining a tensor's rank is NP-hard. The second notion of rank, which we use in this work, is the n-rank, which is a vector of length n for an nth order tensor. The ith value is the matrix rank of $\mathcal{W}_{<i>}$; i.e., the dimension of the space spanned by the mode-i

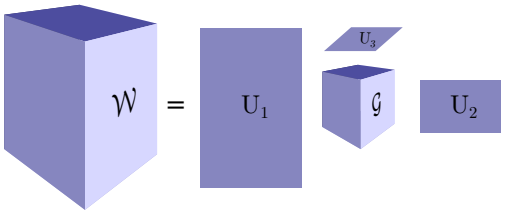


Fig. 1. The Tucker decomposition.

fibers. Unlike matrix rank, in which the column rank and the row rank are equal, the n-rank is generally different for each mode. A tensor $\mathcal{W} \in \mathbb{R}^{I \times J \times K}$ with n-rank (n_1, n_2, n_3) can be compactly represented using a Tucker decomposition [1], as shown in Figure 1. That is,

$$\mathcal{W} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3 \tag{1}$$

where $\mathcal{G} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a compact *core tensor*, and the columns of matrix U_i form a basis for the mode-i fiber space.

Because the n-rank function is non-convex and therefore difficult to optimize, researchers often instead use a convex relaxation known as tensor nuclear norm[1] [2], [3]. The nuclear norm of an Nth order tensor, $\|\mathcal{W}\|_*$ is defined to be

$$\|\mathcal{W}\|_* = \|\mathcal{W}_{<1>}\|_* + \|\mathcal{W}_{<2>}\|_* + \cdots + \|\mathcal{W}_{<N>}\|_* \tag{2}$$

where $\|\mathcal{W}_{<n>}\|_*$ is the matrix nuclear norm of the mode-n unfolding; i.e. the sum of the singular values in that unfolding. Like low rank matrices, low n-rank tensors can be thought of as "simple" in a sense; we will see that using low n-rank parameterized classifiers will lend itself to an intuitive interpretation of the relationship between model input and output. For an excellent introduction to tensors and a survey of their use, we refer the reader to [4].

B. Log-Linear Models

Log-linear models have been used extensively for classification. These models map an input vector $x \in \mathbb{R}^D$ to a vector of posterior probabilities $y \in \mathbb{R}^C$ via the following simple relationship, where vectors $w_1, w_2, \ldots, w_C \in \mathbb{R}^D$ and scalars $b_1, b_2, \ldots, b_c$ are the model parameters:

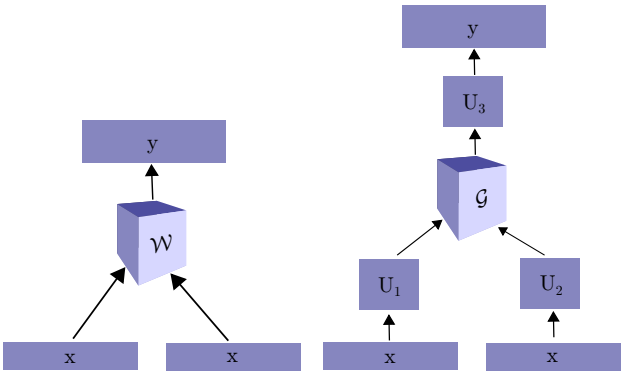
$$P(Y = i|x) = y_i = \frac{\exp(w_i^T x + b_i)}{\sum_{j=1}^{C} \exp(w_j^T x + b_j)}. \tag{3}$$

Stated equivalently, $y = \text{softmax}(W^T x + b)$, where $W \in \mathbb{R}^{D \times C}$ is the matrix whose columns are the various w_i and $b \in \mathbb{R}^C$ is the vector whose elements are b_i. Training these models to maximize likelihood involves solving a convex optimization problem:

$$\arg\max_{W,b} \prod_{i=1}^{N} P(Y = t_i|x_i) \tag{4}$$

where t_i is the true class label for datapoint i. As defined, the decision boundaries are linear, and the model makes use of only first order information of the inputs. Non-linear classification can be performed in this framework by replacing x in the above equation with $\phi(x)$, where ϕ denotes a non-linear feature expansion. For example, for quadratic feature expansion (i.e. polynomial expansion with degree 2), ϕ maps from $\mathbb{R}^D$ to $\mathbb{R}^{D^2}$, where the elements of $\phi(x)$ correspond to all pairs of products of elements in x.

[1]Sometimes also referred to as the tensor trace norm.



(a) cTELLAR model. (b) fTELLAR model.

Fig. 2. Parameterizations for the TELLAR models.

III. The Tensor Log-Linear (TELLAR) Model

Our basic TELLAR model is visualized in Fig. 2a. The input vector $x \in \mathbb{R}^D$ is multiplied along two of the modes of tensor $\mathcal{W} \in \mathbb{R}^{D \times D \times C}$, and then the softmax function is applied to the result to produce a probability distribution over the C output classes:

$$y = \text{softmax}(z), \text{ where} \tag{5}$$

$$z = \mathcal{W} \times_1 x \times_2 x. \tag{6}$$

Because x is multiplied along two modes of the tensor, the model is capable of extracting second order features of the input and thus learning non-linear decision boundaries. The approach generalizes to higher orders, although only second order features are explored in this work.

If $\mathcal{W}$ has full n-rank, this model is equivalent to doing a quadratic feature expansion $\hat{x} = \phi(x)$ ($\hat{x}$ contains all pairs of products of elements in x) and then feeding $\hat{x}$ into a standard log-linear model. Non-linear feature expansion is a standard technique for obtaining non-linear classifiers using underlying linear methods. The primary downside to this explicit feature expansion is the explosion of parameters (our model now has $O(D^2C)$ parameters instead of the $O(DC)$ used for a linear classifier), which puts the model at risk of overfitting.

Either encouraging or constraining the tensor to have low n-rank serves to regularize the model and decreases the risk of overfitting. When $\mathcal{W}$ has n-rank $[n_1, n_2, n_3]$ the number of free parameters is $O(n_1 n_2 n_3 + D(n_1 + n_2) + Cn_3)$, as shown by the Tucker decomposition. When $n_1 \ll D$, $n_2 \ll D$ and $n_3 \ll C$ this results in a dramatic reduction in parameters. In short, tensor n-rank gives us a mechanism to carefully control model complexity to achieve better balance between bias and variance for the task at hand.

As shown in Fig 2b, when $\mathcal{W}$ has low n-rank with Tucker decomposition $\mathcal{W} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3$, it can be interpreted as having three components: 1) the linear transformations of x via U_1 and U_2, 2) a linear transformation of the output y via U_3, and 3) a multilinear interaction between the low dimensional representations of the data and label via core tensor $\mathcal{G}$. If we let e_i be the ith standard basis vector, this interpretation is illustrated by the following:

$$\begin{aligned} z_i &= [\mathcal{W} \times_1 x \times_2 x]_i && (7)\\ &= \mathcal{W} \times_1 x \times_2 x \times_3 e_i && (8)\\ &= (\mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3) \times_1 x \times_2 x \times_3 e_i && (9)\\ &= \mathcal{G} \times_1 (U_1 x) \times_2 (U_2 x) \times_3 (U_3 e_i) && (10)\\ &= \mathcal{G} \times_1 \tilde{x}_1 \times_2 \tilde{x}_2 \times_3 \tilde{y} && (11)\\ &= \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \mathcal{G}_{ijk} \tilde{x}_{1,i} \tilde{x}_{2,j} \tilde{y}_k && (12) \end{aligned}$$

Here $\tilde{x}_1$ denotes the low dimensional continuous representation of x induced by the model by multiplying along the first mode, $\tilde{x}_2$ the same for the second mode and $\tilde{y}$ denotes the low dimensional representation of class i. During training, the linear transformations of x will be learned to enhance discrimination, while the low dimensional continuous representations of classes $\tilde{y}$ will learn to exploit similarity/overlap in the tasks by placing similar classes near to each other in this low dimensional space. The latter means that this can be viewed as a multi-task learning problem: our model can pool knowledge of several tasks together to make more accurate predictions on each task individually (multitask learning has been explored previously with low rank matrices [5], [6]). These three terms, $(\tilde{x}_1, \tilde{x}_2, \tilde{y})$ are multilinearly combined via the core tensor, which governs how terms interact to produce value z_i. Two types of TELLAR models are considered.

A. cTELLAR

The convex TELLAR (cTELLAR) model is parameterized by $\mathcal{W}$ directly (the parameterization illustrated in Fig. 2a). Simply using maximum-likelihood as a training objective, however, is unlikely to produce a result with low n-rank. To encourage this property in our trained model, we can add a convex tensor nuclear norm term to our objective and train as follows:

$$\arg\min_{\mathcal{W}} \gamma \|\mathcal{W}\|_* - LL(\mathcal{D}|\mathcal{W}) \tag{13}$$

Here $LL(\mathcal{D}|\mathcal{W})$ denotes the log-likelihood of our training data, $\mathcal{D}$. While the smooth, differentiable log-likelihood term is easy to optimize, minimizing the tensor nuclear norm is somewhat more complicated. To train the model, we employ the Convex Multilinear Estimation Algorithm introduced by Signoretto [2]. In practice, we use a slightly modified objective (inspired by [7]) that allows us to more or less heavily regularize different elements in the n-rank vector:

$$\arg\min_{\mathcal{W}} \sum_{i=1}^{3} \gamma_i \|\mathcal{W}_{<i>}\|_* - LL(\mathcal{D}|\mathcal{W}) \tag{14}$$

The primary advantage of the cTELLAR approach is that training is convex, guaranteeing that we can converge to a globally optimal solution. The major disadvantage is the computational complexity: $O(D^2C)$ space is required and each iteration of the algorithm requires a singular value decomposition on all three tensor unfoldings.

B. fTELLAR

As a more computationally efficient alternative, we also explore the use of a Tucker-factorized parameterization (the parameterization in Fig. 2b). In this model, our parameters are matrices $U_1 \in \mathbb{R}^{D \times n_1}$, $U_2 \in \mathbb{R}^{D \times n_2}$ and $U_3 \in \mathbb{R}^{C \times n_3}$ along with core tensor $\mathcal{G} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Tensor $\mathcal{W} = \mathcal{G} \times_1 U_1 \times_2 U_2 \times_3 U_3$ is never explicitly constructed. Our training objective then is

$$\arg\min_{\mathcal{G}, U_1, U_2, U_3} -LL(\mathcal{D}|\mathcal{W}) \tag{15}$$

Implicitly, this is a constrained optimization problem, restricted to solutions for which the n-rank of $\mathcal{W}$ is upper bounded by the n_1, n_2 and n_3. Complementary to the cTELLAR model, the fTELLAR model has advantages in space and time complexity, but in exchange forfeits convexity. We trained the model using stochastic minibatch gradient descent using the Tensorflow toolkit [8].

C. Baselines

The view of our model as providing a means to interpolate between the expressiveness of linear and quadratic models motivates the following three baselines. The first is a standard log-linear exponential model (denoted "Linear" in our experiments), as defined in Section II-B, which represents a practical (but not theoretical) lower-bound on the expressiveness of the TELLAR model. Our second baseline is a full n-rank TELLAR model; as noted above, this is equivalent to doing a full quadratic feature expansion and then using a standard log-linear model. This can be seen as follows, where z is defined in Eqn. 6:

$$z_k = \sum_{i,j} \mathcal{W}_{ijk} x_i x_j = \sum_{\ell} \Lambda_{k\ell} \phi(x)_\ell = \Lambda_k^T \phi(x) \tag{16}$$

where $\Lambda \in \mathbb{R}^{C \times D^2} = \mathcal{W}_{<3>}$ is just a reparameterization of the weights, $\phi(x) \in \mathbb{R}^{D^2}$ is an explicit quadratic feature expansion, and ℓ iterates over $1, 2, \ldots D^2$. This is now in the form of a log-linear exponential model, with class dependent weights Λ_k, applied to features with an explicit feature expansion. This baseline is denoted "Quadratic" in the experimental results, and represents the upper-bound on expressiveness on the expressiveness of TELLAR. Finally, we also report the majority class baseline; that is, the model that always predicts the class which was most common in the training set. This baseline is denoted "Majority" in our experiments.

IV. Prior Work

There have been several machine learning models that, like TELLAR, involve some parameterization by a tensor. Many of these are energy-based models [9], [10], [11], [12], [13].

	Wine	MSD	Geo	CTG
N_{tr}	3214	463715	530	1063
N_{de}	1628	25815	-	-
N_{te}	1655	25815	529	1063
D	11	90	116	36
C	7	89	33	3

TABLE I
DATASET STATISTICS: TRAINING SET (N_{tr}), DEVELOPMENT SET (N_{de}) AND TEST SET (N_{te}) SIZES, DIMENSION OF THE INPUT DATA (D) AND NUMBER OF CLASSES (C). WHEN N_{de} IS NOT LISTED, MODEL IS TUNED USING 5-FOLD CROSS-VALIDATION THE TRAINING SET.

Model	Wine	MSD	Geo	CTG
Majority	0.451	0.082	0.070	**0.866**
Linear	0.452	0.079	0.301	**0.866**
Quadratic	**0.534**	0.080	0.034	**0.866**
cTELLAR	0.458	0.083	0.096	**0.866**
fTELLAR	**0.534**	**0.087**	**0.339**	**0.867**

TABLE II
TEST SET ACCURACIES ON THE WINE QUALITY (WINE), YEAR PREDICTION MILLION SONG DATASET (MSD), GEOLOCATION OF MUSIC (GEO) AND CARDIOTOCOGRAPY (CTG) DATASETS.

Notably, like TELLAR, the Gated Softmax Classifier [11] is a parametric classification model. However, it uses a third-order tensor to model interactions between input x, output y and a latent hidden vector h. The hidden vector is efficiently marginalized over, yielding a model that acts as a product of experts over linear models. Like TELLAR, the Factored 3-Way Restricted Boltzmann Machine [12], [13] models captures second order structure by multiplying a tensor along two modes by an input vector x. In contrast, however, it is a generative model and the third mode corresponds to a hidden vector h, which must then be marginalized over. In these energy-based models, the parameter tensor is often represented in a CANDECOMP/PARAFAC [14], [15] factored form, which can be seen as a special case of the Tucker form in which $n_1 = n_2 = n_3$ and core tensor $\mathcal{G}$ is superdiagonal [4]. More recently, tensors have been used to parameterize various neural network models, including deep stacking networks [16], [17], deep neural networks [18] and recursive neural networks [19], [20]. In language processing applications, low rank (not low n-rank) tensor-parameterized models were shown to be effective in scenarios where the training data is limited, requiring more careful control over model complexity [21].

V. EXPERIMENTS

We conducted a set of experiments to assess the effectiveness of the cTELLAR and fTELLAR models.

A. Data

Our experiments make use of four standard, publicly available datasets from the UCI Machine Learning repository [22]. These datasets were selected to offer variety in terms of the number of datapoints, input dimension and number of classes. Dataset statistics are summarized in Table I.

1) Wine Quality: The Wine Quality dataset [23] takes 11 attributes of red and white wines and includes subjective ratings by experts. Ratings were made on a scale of 1-10, but only ratings of 3-9 appear in the data; we therefore frame this as a seven-way classification task. The 4898 red wine samples and 1599 white wine samples were combined, randomly shuffled and then split at a roughly 50-25-25 ratio into training, development and test sets.

2) Year Prediction MSD: In this this subset of Million Song Dataset (MSD) [24], the task is to predict the release year of a song (out of the set of 89 distinct years between 1922 and 2011) given 90 dimensional acoustic features produced by the Echo Nest API. The large number of classes and similarity between subsets thereof poses an interesting test of our model's ability to learn meaningful low-dimensional representations of classes. The standard split between training and test sets ensures that no artist shows up in both sets; we evenly split the standard test set into our development and test sets to inherit this property for our development set as well.

3) Geographical Origin of Music: This dataset [25] consists of 116-dimensional acoustic features extracted using the MARSYAS tool on 1059 songs. The task is to predict where, out of a set of 33 distinct areas, the music originated. Given the small size of this data, we performed a random 50-50 split between training and test sets, and used cross-validation on the training set in lieu of a development set.

4) Cardiotocograpy (CTG): The CTG dataset includes 2126 fetal cardiotocograms, from which 36 features are extracted. Each cardiotocogram is labeled as normal, suspect or pathological. Like the "Geo" task above, due to the small number of samples, we split 50-50 between training and test, and perform cross-validation on the training set to tune.

B. Tuning

Several hyper-parameters were tuned, either on the development set (Wine and MSD) or on the training set using five-fold cross-validation (Geo and CTG). All hyper-parameters were tuned with the Bayesian hyperparameter optimization toolkit Spearmint [26]. For the cTELLAR model, we tuned the regularization parameters γ_1, γ_2 and γ_3. For the fTELLAR model, we tuned the n-rank (n_1, n_2 and n_3), learning rate, minibatch size and weight initialization range.

C. Results and Analysis

The results of our classification experiments on the four datasets outlined in Section V-A are summarized in Table II. There are several interesting observations.

First, as expected, when there are relatively many samples (N_{tr}) compared to the feature dimension (D), the more complicated quadratic models offer better performance. Likewise, when there are relatively few samples compared to the feature dimension, the linear model greatly outperforms. This result agrees with standard intuition about the effect of model complexity on the bias and variance trade-off.

Second, the fTELLAR model ties or exceeds the performance of all other models on all four tasks, including the linear model for small N_{tr}/D tasks and the quadratic model for large N_{tr}/D tasks, suggesting that the ability to carefully control

model complexity via tensor n-rank provides a superior bias-variance trade-off. On the Wine dataset, the optimal fTELLAR model is in fact a full n-rank model (i.e. quadratic model). This makes some intuitive sense, since this dataset has the fewest features (D) and therefore increases model complexity the least by doing this feature expansion. This property appears to allow the model to avoid overfitting, even with the modest training set size.

Third, the cTELLAR performance lags behind that of the fTELLAR, sometimes exceeding linear and quadratic model performance, but not always. With the cTELLAR model, we exactly solve an approximation of our true objective (minimizing tensor nuclear norm instead of tensor n-rank). We hypothesize that, in at least some case, the advantages of the convex objective are more than offset by the approximation we have to make to the objective. This result, combined with the fact that the fTELLAR model has both better space and time complexity compared to the cTELLAR, points to the fTELLAR model as being superior for the tasks considered in this paper.

VI. Conclusions

In this paper we introduce two low n-rank parameterized log-linear models: cTELLAR, in which the low n-rank structure is encouraged through tensor nuclear norm regularization and fTELLAR, in which it is explicitly constrained by parameterizing the weight tensor by a Tucker decomposition. These models multilinearly map the input vector (twice) to a prediction, allowing the models to use second order feature information in their predictions. With full n-rank structure, the models act equivalently to a log-linear model applied to features after an explicit quadratic feature expansion. Controlling the n-rank gives us a precise mechanism to control model complexity, and thus achieve a superior bias-variance trade-off. In a set of experiments on four standard classification datasets, specifically selected to offer a range of sizes in terms of number of features, classes and datapoints, the factored low n-rank model (fTELLAR) was found to consistently yield the best performance. Notably, the fTELLAR model decreased error by as much as 15.0%, 31.6% and 28.9% relative over linear, quadratic and majority class baselines, respectively.

There are many ways this work could be extended. First, it could be of interest to explore higher order features (beyond quadratic); both TELLAR models would generalize naturally by using higher order tensors. There are additional training techniques one could apply, including different tensor n-rank minimization algorithms (e.g. [27]) and other regularization strategies (e.g. ℓ_2 regularization on the factored weights). Finally, it would be of interest to explore these models on a wider range of classification tasks to further explore the effect of training set size, feature dimension and number of classes.

Acknowledgments

The authors would like to thank the Nvidia corporation for their donation of a Titan X GPU used in this research.

References

[1] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, no. 31, pp. 279–311, 1966.

[2] M. Signoretto, L. De Lathauwer, and J. A. K. Suykens, "Nuclear norms for tensors and their use for convex multilinear estimation," ESAT-SISTA, K.U.Leuven (Leuven, Belgium), Tech. Rep., 2010.

[3] R. Tomioka, K. Hayashi, and H. Kashima, "On the extension of trace norm to tensors," Oct 2010, http://arxiv-web3.library.cornell.edu/abs/1010.0789v1.

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

[5] Y. Amit, M. Fink, N. Srebro, and S. Ullman, "Uncovering shared structures in multiclass classification," in *Proc. ICML*, 2007, pp. 17–24.

[6] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Mach. Learn.*, vol. 73, no. 3, pp. 243–272, Dec 2008.

[7] R. Tomioka, K. Hayashi, and H. Kashima, "Estimation of low-rank tensors via convex optimization," 2011. [Online]. Available: http://arxiv.org/abs/1010.0789

[8] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[9] R. Memisevic and G. Hinton, "Learning to represent spatial transformation with factored higher-order boltzmann machines," Department of Computer Science, University of Toronto, Tech. Rep. UTML TR 2009-003, July 2009.

[10] G. W. Taylor and G. E. Hinton, "Factored conditional restricted boltzmann machines for modeling motion style," in *Proc. ICML*, 2009.

[11] R. Memisevic, C. Zach, G. Hinton, and M. Pollefeys, "Gated softmax classification," in *Proc. NIPS*, 2010, pp. 1603–1611.

[12] M. Ranzato and G. E. Hinton, "Modeling pixel means and covariances using factorized third-order boltzmann machines," in *Proc. CVPR*, 2010, pp. 2551–2558.

[13] M. Ranzato, A. Krizhevsky, and G. E. Hinton, "Factored 3-way restricted boltzmann machines for modeling natural images," in *Proc. AISTATS*, 2010.

[14] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of the "eckart-young" decomposition," *Psychometrika*, no. 35, pp. 283–319, 1970.

[15] A. Harshman, "Foundations of the PARAFAC procesdure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, no. 16, pp. 1–84, 1970.

[16] B. Hutchinson, L. Deng, and D. Yu, "A deep architecture with bilinear modeling of hidden representations: Applications to phonetic recognition," in *Proc. ICASSP*, 2012.

[17] ——, "Tensor deep stacking networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1944–1957, 2013.

[18] D. Yu, L. Deng, and F. Seide, "The deep tensor neural network with applications to large vocabulary speech recognition," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, pp. 388–396, 2013.

[19] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. EMNLP*, 2013.

[20] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Proc. NIPS*, 2013.

[21] B. Hutchinson, M. Ostendorf, and M. Fazel, "Low rank language models for small training sets," *IEEE Signal Processing Letters*, vol. 18, no. 9, pp. 489–492, 2011.

[22] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[23] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physiochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 2009.

[24] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc ISMIR*, 2011.

[25] F. Zhou, C. Q, and R. D. King, "Predicting the geographic origin of music," in *Proc. ICDM*, 2014.

[26] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proc. NIPS*, 2012.

[27] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, Feb 2011.