Rank regularization

Oleh Rybkin
University of Pennsylvania
oleh@seas.upenn.edu

Abstract

I present a regularization algorithm based on rank penalization. When applied to the multiclass classification task, the algorithm has intuitive interpretation. Moreover, I show that the algorithm is equivalent in expressivity to another structured learning algorithm described in (Srikumar and Manning, 2014). As a second experiment, I analyze the embedding of the label space produced by these and classical methods.

1 Introduction

The paper on which I base my work gives a new way for constructing joint feature space in the structured learning problem. The authors argue for the use of interpretable label space embedding and enforcement of structure on them, such as making them linearly dependent.

I rigorously analyze the model and present a simplification of it. Further, inspired by (Mikolov et al., 2013; Liu et al., 2017) I analyze the embeddings of the output space. Embedding of the output space are vectors from the vector space $\mathbb{R}^m$ that in some sense represent the labels of the given classification problem. For example, given document classification task to various theme-based online boards, we could expect the label vector representing "sports.baseball" to be close to the vectors of "sports.hockey" then to "sci.electronics", when judged by some appropriate metric. The possible metrics are, for instance, the cosine distance or the l2 norm.

2 Original Model

Label vectors

The paper introduces a new model called DISTRO which uses a two-layer structure consisting of the weight vector $\mathbf{w}$ and label matrix $\mathbf{A}$.

The output of the first layer is constructed iteratively as tensor product of the current label vector with the previous tensor. In the simplest atomic case, the output is just a tensor product of the features and a label vector. The weights of the first layer are computed from the label matrix $\mathbf{A}$.

$$\Psi_p(\mathbf{x}, \mathbf{y}_p, \mathbf{A}) = \begin{cases} \mathbf{a}_{l_{y_p}} \otimes \phi_p(\mathbf{x}), p \text{ atomic} \\ \mathbf{a}_{l_{y_p}} \otimes \Psi(\mathbf{x}, \mathbf{y}_p^{1:}, \mathbf{A}) \end{cases}$$

The output of the model is computed by viewing the output of the first layer as the features. Standard structured learning techniques apply straightforwardly for training of the second layer weights $\mathbf{w}$.

$$\Phi_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) = \sum_{p \in \Gamma_x} \text{vec}(\Psi_p(\mathbf{x}, \mathbf{y}_p, \mathbf{A}))$$

Optimization

As a regularization technique, the authors employ rank penalization on the $\mathbf{A}$ matrix, approximated as nuclear form. This further reduces the degree of freedom of the label vectors and forces them to share more information.

The model is thus optimized iteratively, with a structured learning method optimizing $\mathbf{w}$ in the first step and a proximal optimization method optimizing $\mathbf{A}$ in the second step.

3 Analysis

I will analyze the model and give a proof that there exist an equivalent model with strictly fewer number of parameters. This gives us a theoretical insight into the workings of the model, but the modification can also be practically implemented.

Model Equivalence

Let us first analyze the trivial case of the model, where each label vector is a one-hot encoding of the corresponding label (part). In that case, the model reduces to the multiclass classification problem, where each label l has a weight vector $\mathbf{w}_a$ that is used for its scoring. We can form these vectors into a tensor $\mathbf{W}$, where the first p indices of $\mathbf{W}$ correspond to the number of classes in a specific label part, and the last index is the number of features. In case where the labels are atomic, $\mathbf{W}$ is just a matrix of size (num_labels $\times$ num_features).

Now let us consider the case which the original paper uses in practical experiments. The label matrix $\mathbf{A}$ is rank-deficient and each label part vector has less dimensions than there is classes. We can form a tensor $\mathbf{V}$ in the same way as $\mathbf{W}$ above: the first p indices of $\mathbf{V}$ correspond to the dimensionalities of the label part vectors $\mathbf{a}_p$. The label score given the label parts $\mathbf{a}_p$ can be then computed as:

$$a = (\mathbf{V} \circ \mathbf{a}^a)\phi(\mathbf{x}) = \sum_{i_1=1}^{\dim(\mathbf{a}_1)} \sum_{i_2=1}^{\dim(\mathbf{a}_2)} \cdots$$

$$\sum_{i_p=1}^{\dim(\mathbf{a}_p)} (\mathbf{a}_1)_{i_1}(\mathbf{a}_2)_{i_2} \dots (\mathbf{a}_p)_{i_p} \mathbf{v}^T_{i_1,i_2,\dots,i_p} \phi(\mathbf{x}),$$

where the vector $\mathbf{a}^a$ is a concatenation of label parts $\mathbf{a}_i$ for a label a. Note that the same formula applies to the formulation of the original model. In that case all but one summands will be zero. In fact, there is a strong connection between the two models, summarized in the next result.

Theorem 1. *For each instance of* DISTRO*, i.e. the pair of label matrix $\mathbf{A}$ and weight tensor $\mathbf{V}$, there exists a instance of structured SVM, i.e. the weight tensor $\mathbf{W}$, that scores the labels in the same way.*

Proof. Using the notation established in this section, for each label a set

$$\mathbf{w}_a = \mathbf{V} \circ \mathbf{a}^a.$$

The weight tensor $\mathbf{W}$ produced in this way does score the labels the same as the DISTRO model.

□

The theorem means that DISTRO has no more expressivity power than SVM. The difference between SVM and DISTRO is in fact in that the DISTRO is more constrained. I explain the precise form of constraints in the next result.

Theorem 2. *If $\mathbf{V}$ is full rank, the rank of the tensor $\mathbf{W}$ constructed as in the proof of the Theorem 1 can be computed from the label vector matrix $\mathbf{A}$ in the next way:*

$$r = \mathit{rank}(\mathbf{W}) = \prod_{i=1}^{p} \mathit{rank}(\mathbf{A}_i),$$

where $\mathbf{A}_i$ is the matrix containing the label parts $\mathbf{a}_i$.

Moreover, for any $\mathbf{W}$ of the rank less or equal then r, we can find a DISTRO *instance that is equivalent to $\mathbf{W}$, in the sense that it scores the labels in the same way.*

Proof. The proof applies by induction on number of label parts p: the theorem is trivially true for atomic labels. Denote by $\mathbf{W}_l$ the tensor constructed from l label parts. Then the elements (tensors with dimensionality smaller by one) of it, $\mathbf{W}'_{l-1}, \mathbf{W}''_{l-1}$ are linearly dependent if and only if their corresponding label parts $\mathbf{a}_l, \mathbf{a}'_l$ are linearly dependent. It follows that the tensor $\mathbf{W}_l$ has the rank

$$\text{rank}(\mathbf{W}_l) = \text{rank}(\mathbf{W}_{l-1})\text{rank}(\mathbf{A}_l),$$

where $\mathbf{W}_{l-1}$ is the tensor constructed in a previous induction step.

I will prove the second part of the theorem for the case of atomic labels only. In that case the correspondence between $\mathbf{W}$ and $\mathbf{V}$ is given by

$$\mathbf{W} = \mathbf{A}^T\mathbf{V}.$$

Trivially, for each matrix $\mathbf{W}$ of rank $k < \text{rank}(\mathbf{A})$ we can find such $\mathbf{W}, \mathbf{A}$ that the equation holds.

□

The theorem says that the expressivity of the DISTRO model is equivalent to the expressivity of an SVM where the weight tensor has constrained rank. For the case of atomic labels this weight tensor is just a rank-deficient matrix.

Rank penalization

The results in the previous paragraphs show that rank penalization plays the same role as decreasing the dimensionality of label vectors. The rank of the results weight tensor $\mathbf{W}$ is given by the rank of the label matrices, which can be decreased by enforcing rank-deficiency as well as by decreasing the label vectors dimensionality.

The nuclear norm minimization as a proxy to rank penalization can, however, be an effective regularization technique on it's own. In fact, the results of the original paper suggest that the nuclear form improves the performance at least in some circumstances.

4 Model

Based on the results above I propose a new model, Rank-regularized SVM, that has equivalent theoretical properties to DISTRO. The model is simpler and contain less trainable parameters.

I pose the problem as learning the weight matrix $\mathtt{W}$ which is rank-deficient and has a certain rank. This is justified by the Theorem 2. I approximate this approach with minimizing the nuclear norm $\mathtt{W}_*$ of the weight matrix.

When the labels are atomic, the weight tensor is naturally a matrix. In the other case, the weight tensor can be unfolded into a matrix by vectorizing the first p indices. A nice theoretical alternative for that would be to perform canonical polyadic decomposition (Kolda and Bader, 2009) to find the singular values of the tensor itself.

The loss on the model is as follows:

$$L(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) = \lambda_1 \left\|\mathbf{W}\right\|_2 + \lambda_2 \left\|\mathbf{W}\right\|_* + \frac{1}{N}\sum_i L_s(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}),$$

where $\left\|\mathbf{X}\right\|_*$ is the nuclear norm and the L_s is a structured learning loss. Note that the loss decomposes to three separate updates which can correspondingly be applied separately. This gives us the sketch of the training procedure for the model in the Alg. 1.

Note that the complexity of the training procedure is notably lower than the complexity of DISTRO. As the nuclear regularization is applied to the weight tensor rather than to the label vectors, the optimization algorithm needs not to be iterative.

Furthermore, with the coefficient λ in the loss function I can control the regularization and the desired rank of the matrix.

Algorithm 1: A training procedure for Rank-regularized SVM

Data: List of inputs $\mathbf{x}_i$ with the corresponding labels $\hat{a}$.

begin

  foreach *training example* **do**

    Do the structural update:

    $a \leftarrow \arg\max_a \mathbf{w}_a^T \phi(\mathbf{x})$;

    $\mathbf{w}_a \leftarrow \mathbf{w}_a - c\phi(\mathbf{x})$;

    $\mathbf{w}_{\hat{a}} \leftarrow \mathbf{w}_a + c\phi(\mathbf{x})$;

    Apply weight decay: $\mathbf{w} \leftarrow \mathbf{w} - \lambda_1 \mathbf{w}$;

    Compute the SVD of the weight matrix $\mathbf{W}$: $\mathbf{U}, \mathbf{S}$

    $\mathbf{S}' \leftarrow \mathbf{S} - \lambda_2 \mathbf{I}$;

    Do the nuclear update:

    $\mathbf{W} \leftarrow \mathbf{U}\mathbf{S}'\mathbf{S}^{-1}\mathbf{U}^T\mathbf{W}$;

  end

end

Discussion

My theoretical results prove the redundancy of some parameters in the DISTRO formulation. Rank-regularized SVM has less parameters, which means that the training process should be faster and more stable. The vectors learned by Rank-regularized SVM are also easily interpretable, while the vectors learned by DISTRO are corrupted because of the redundancy of the formulation (see the Fig. 1).

One can gain some intuition about the problem by considering a parallel of models to MLP. While the original model can be viewed as a two-layer perceptron, there is no non-linearity between the two layers, which leads to the collapse in expressive power. Rank-regularized SVM represents a model with single layer substituting for the two collapsed layers of DISTRO.

The DISTRO model, however, is easily computable without specialized libraries for SVD computation which is needed for Rank-regularized SVM. DISTRO offers a simple formulation for en-

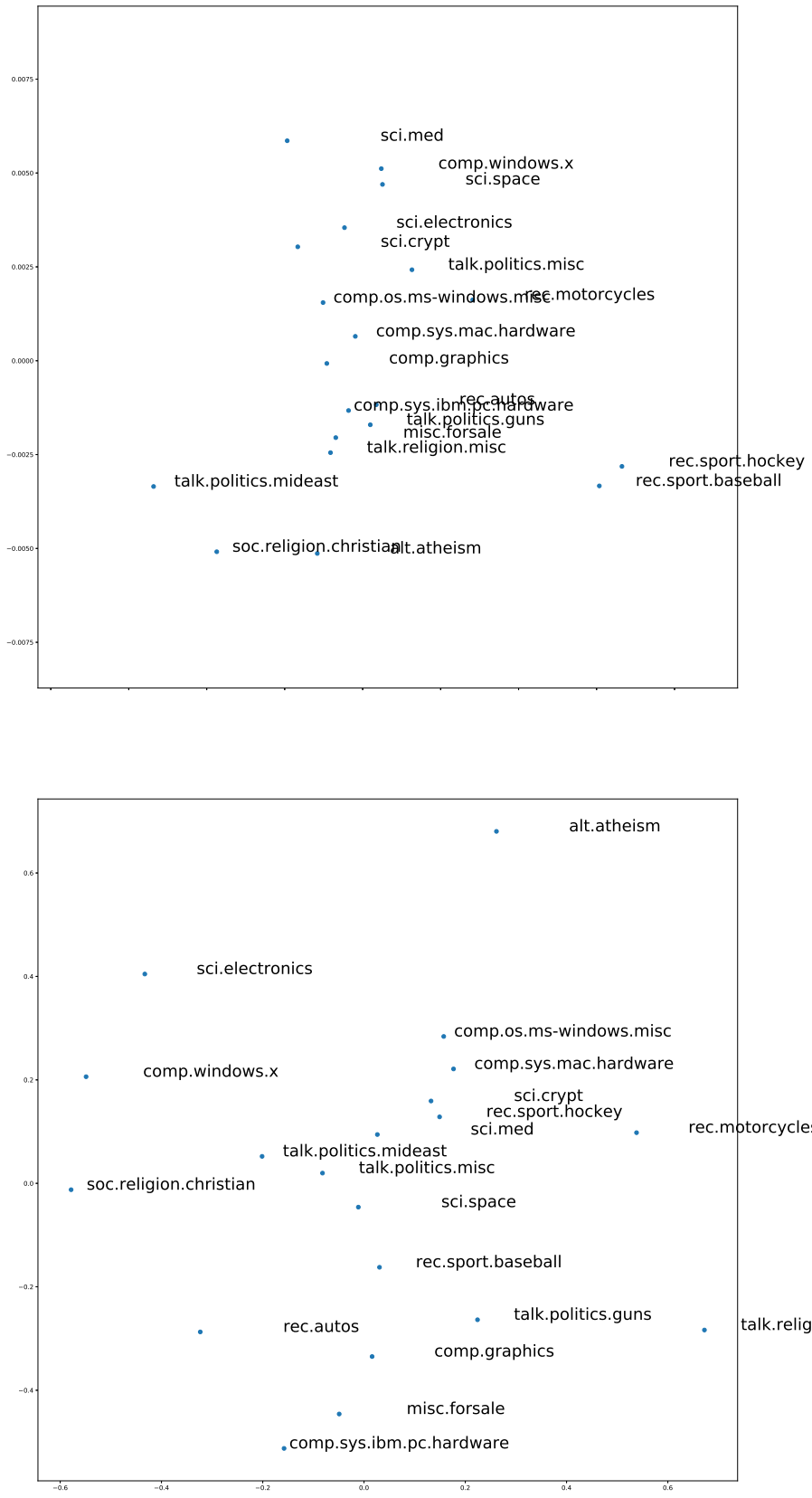

Figure 1: PCA of label embeddings produced by different models. Top: SVM, botton: DISTRO. SVM produces notably better results than DISTRO due to inherent redundancy in DISTRO label vectors.

forcing the rank-deficiency of the weight matrix by introducing redundant parameters.

5 Experiments

A drawback of my model is the need to compute SVD of a larger matrix than in the original DISTRO implementation. The nuclear form still can be computed efficiently as the weight matrix has a small number of rows. However, a specialized library would be needed as we need to avoid computing the right orthogonal matrix. To my knowledge, there is no efficient solutions for this in Scala, in which the original code is written, which is why I do not conduct experiments with my model. I, however, visualize the learned embedding of a standard structured SVM.

Label embeddings

I examine the learned by SVM weights and show that they form to an embedding of the output labels into $\mathbb{R}^n$. When considering the weight matrix $\mathbf{W}$ as defined above, the rows correspond to the embeddings of a particular label. I plot these vectors in a 2-dimensional space found by PCA dimensionality reduction method. The embeddings produced by SVM are in the Fig. 1, top. Note the cluster of sport to the left, the scientific and electronics-related cluster in the upper half, and the religion cluster to the bottom-left. While the clustering is noisy and not reliable, it clearly captures some of the semantics of the embedding space.

Further, I examine the labels produced by DISTRO. As I argue above, the label values contain redundant parameters which are essentially left to fluctuate randomly. This prevents the labels from being semantically meaningful.

6 Conclusion

I analyze the paper (Srikumar and Manning, 2014) and show that it is equivalent to a simpler model with fewer parameters. I present an algorithm of training for such model. Further, I show that the labels produced by DISTRO lose semantic interpretability due to the uncovered redundancy in the weights. I show that even the weight learned by plain structured SVM show more structure of the output space that labels produced by DISTRO.

References

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM REVIEW*, 51(3):455–500.

Shusen Liu, Peer-Timo Bremer, Jayaraman J. Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. 2017. Visual exploration of semantic relationships in neural word embeddings. PP:1–1, 08.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. 2013, 01.

Vivek Srikumar and Christopher D. Manning. 2014. Learning distributed representations for structured output prediction. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3266–3274, Cambridge, MA, USA. MIT Press.