Trace Norm Regularised Deep Multi-Task Learning

Yongxin Yang, Timothy M. Hospedales
Queen Mary, University of London
{yongxin.yang, t.hospedales}@qmul.ac.uk

Abstract

In this paper, we propose a framework for training multiple neural networks simultaneously. The parameters from all models are regularised by the tensor trace norm, so that one neural network is encouraged to reuse others' parameters if possible – this is the main motivation behind multi-task learning. In contrast to many deep multi-task learning work, we do not predefine a parameter sharing strategy by tying some (usually bottom) layers' parameters, instead, our framework allows the sharing for all shareable layers thus the sharing strategy is learned from a pure data-driven way.

1 Introduction and Related Work

The paradigm of multi-task learning (MTL) [Caruana, 1997] is to learn multiple tasks jointly, and the knowledge obtained from one task could be reused by others. In this section, we will briefly review some studies in this area.

1.1 Matrix-based Multi-Task Learning

Matrix-based MTL is usually built on linear model, i.e., each task is parameterised by a D-dimensional weighting vector w, and the model prediction is $\hat{y} = x \cdot w = x^T w$, where x is a D-dimensional feature vector representing an instance. The objective for the matrix-based MTL can be written as,

$$\sum_{i=1}^{T} \sum_{j=1}^{N^{(i)}} \ell(y_j^{(i)}, x_j^{(i)} \cdot w^{(i)}) + \lambda \Omega(W) \tag{1}$$

Here $\ell(y, \hat{y})$ is a loss function for the true label y and the predicted label $\hat{y}$. T is the number of tasks, and for the i-th task there are $N^{(i)}$ training instances. Assuming the dimensionality of every task's feature is the same, the models – $w^{(i)}$s – are of the same size. Then the collection of $w^{(i)}$s forms a $D \times T$ matrix W of which the i-th column is the linear model for the i-t task. $\Omega(W)$ is a kind of regularisation that encourages W to be a low-rank matrix. Some choices could be the $\ell_{2,1}$ norm [Argyriou et al., 2008], and the trace norm [Ji and Ye, 2009]. An alternative approach [Kumar and Daumé III, 2012] is to explicitly formulate W as a low-rank matrix, i.e., $W = LS$ where L is a $D \times K$ matrix and S is a $K \times T$ matrix with $K < \min(D, T)$ as a hyper-parameter (matrix rank).

1.2 Tensor-based Multi-Task Learning

Though it is a classic setting that each task is indexed by a single factor, in many real-world problems, tasks are indexed by multiple factors. For example, to build a restaurant recommendation system, we want a regression model that predicts the scores for different aspects (food quality, environment) by different customers. Then the task is indexed by aspects $\times$ customers. For this case, the collection of all linear models for all tasks is then a 3-way tensor $\mathcal{W}$ of size $D \times T_1 \times T_2$, where T_1 and T_2 is the number of aspects and the number

of customers respectively. Consequently $\Omega(\mathcal{W})$ has to be a kind of tensor regularisation [Tomioka et al., 2010]. For example, sum of the trace norms on all matriciations[1] [Romera-paredes et al., 2013], and scaled latent trace norm [Wimalawarne et al., 2014]. An alternative solution is to concatenate the one-hot encodings of task factors and feed it as input into a two-branch neural network model [Yang and Hospedales, 2015], in which there are two input channels for feature vector and encoded task factor.

1.3 Multi-Task Learning for Neural Networks

With the great success of deep learning, many researches are proposed for deep multi-task learning. Zhang et al. [2014] use a convolutional neural network to find facial landmarks as well as recognise face attributes (e.g., emotions). Liu et al. [2015] propose a deep neural network for query classification and information retrieval (ranking for web search). One of the key commonalties of these work is that they all use a kind of predefined parameter sharing strategy. A typical design is to use the same set of parameters for the bottom layers of the deep neural network and use the task-specific parameters for the top layers. This kind of architecture design can be traced back to 2000s [Bakker and Heskes, 2003]. However, modern neural network models usually contain a large number of layers, which make the decision after which layer the neural network is split for different tasks extremely hard.

2 Methodology

Instead of predefining a parameter sharing strategy, we propose the following framework:
For T tasks, each is modelled by a neural network of the same architecture[2]. We collect the parameters in a layer-wise fashion, and put a tensor norm on every collection.
We illustrate our framework by a simple example: assuming that we have $T = 2$ tasks, and each is modelled by a 4-layer convolution neural network (CNN). The CNN architecture is: (1) convolutional layer ('conv1') of size $5 \times 5 \times 3 \times 32$, (2) convolutional layer ('conv2') of size $3 \times 3 \times 32 \times 64$, (3) fully-connected layer ('fc1') of size 256×256, (4) fully-connected layer ('fc2'$^{(1)}$) of size 256×10 for the first task and fully-connected layer ('fc2'$^{(2)}$) of size 256×20 for the second task.
Note that we assume these two tasks have different number of outputs. In our framework, the shareable layers are 'conv1', 'conv2', and 'fc1'.
For single task learning mode, the parameters are 'conv1'$^{(1)}$, 'conv2'$^{(1)}$, 'fc1'$^{(1)}$, and 'fc2'$^{(1)}$ for the first task; 'conv1'$^{(2)}$, 'conv2'$^{(2)}$, 'fc1'$^{(2)}$, and 'fc2'$^{(2)}$ for the second task. We can see that there are not any parameter sharing between these two tasks.
For one of the possible predefined deep MTL, the parameters could be 'conv1', 'conv2', 'fc1'$^{(1)}$, and 'fc2'$^{(1)}$ for the first task; 'conv1', 'conv2', 'fc1'$^{(2)}$, and 'fc2'$^{(2)}$ for the second task, i.e., the first and second layer are fully shared in this case.
For our proposed method, the parameter setting is the same as single task learning mode, but we put three tensor norms on the stacked {'conv1'$^{(1)}$, 'conv1'$^{(2)}$} (a tensor of size $5 \times 5 \times 3 \times 32 \times 2$), the stacked {'conv2'$^{(1)}$, 'conv2'$^{(2)}$} (a tensor of size $3 \times 3 \times 32 \times 64 \times 2$), and the stacked {'fc1'$^{(1)}$, 'fc1'$^{(2)}$} (a tensor of size $256 \times 256 \times 2$) respectively.

2.1 Tensor Norm

We choose to use the trace norm, which is defined as the sum of matrix's singular values.

$$||X||_* = \sum_{i=1} \sigma_i \tag{2}$$

[1] Matriciation is also known as tensor unfolding or flattening.
[2] For the case that each task has a different number of outputs, the parameters of topmost layers from neural networks should be of different size, thus they are opted out for sharing.

The trace norm is named by the fact that when X is a positive semidefinite matrix, it is the trace of X. It is sometimes referred to as nuclear norm. Trace norm is the tightest convex relation of matrix rank [Recht et al., 2010].
The extension of trace norm from matrix to tensor is not unique, just like the rank of tensor has different definitions. How to define the rank of tensor largely depends on how the tensor is factorised, e.g., Tucker decomposition [Tucker, 1966] and Tensor-Train decomposition Oseledets [2011].
We propose three tensor trace norm designs here, which are corresponding to three variants of the proposed method.
For an N-way tensor $\mathcal{W}$ of size $D_1 \times D_2 \times \cdots \times D_N$. We define
Tensor Trace Norm Tucker

$$||\mathcal{W}||_* = \sum_{i=1}^{N} \gamma_i ||\mathcal{W}_{(i)}||_* \tag{3}$$

Here $\mathcal{W}_{(i)}$ is the mode-i tensor flattening/unfolding, which is obtained by,

$$\mathcal{W}_{(i)} = \text{reshape}(\text{transpose}(\mathcal{W}, [D_i, D_1, \ldots, D_{i-1}, D_{i+1} \ldots, D_N]), [D_i, \prod_{j \neg i} D_j]) \tag{4}$$

Tensor Trace Norm TT

$$||\mathcal{W}||_* = \sum_{i=1}^{N-1} \gamma_i ||\mathcal{W}_{[i]}||_* \tag{5}$$

Here $\mathcal{W}_{[i]}$ is yet another way to unfold the tensor, which is obtained by,

$$\mathcal{W}_{[i]} = \text{reshape}(\mathcal{W}, [D_1 \times D_2 \ldots D_i, D_{i+1} \times D_{i+2} \ldots D_N]) \tag{6}$$

Tensor Trace Norm Matrix

$$||\mathcal{W}||_* = \gamma ||\mathcal{W}_{(N)}||_* \tag{7}$$

This is the simplest definition. Given that in our framework, the last axis of tensor indexes the tasks, i.e., $D_N = T$, it is the most straightforward way to adapt the technique in matrix-based MTL – reshape the $D_1 \times D_2 \times \cdots \times T$ tensor to $D_1 D_2 \cdots \times T$ matrix.

2.2 Optimisation

Using gradient-based method for optimisation involved with trace norm is *not* a common choice, as there are better solutions based on semi-definite programming or proximal gradients because the trace norm is essentially non-differentiable. However, deep neural network is usually trained by gradient descent, and it is relatively harder to modify the training process of neural network. Therefore we use the (sub-)gradient descent for trace norm terms. The sub-gradient for trace norm can be derived as,

$$\frac{\partial ||X||_*}{\partial X} = X(X^T X)^{-\frac{1}{2}} \tag{8}$$

A more numerical stable method instead of computing $(X^T X)^{-\frac{1}{2}}$ is to use SVD. For an $N \times P$ matrix X, SVD computes

$$X = U \Sigma V^T \tag{9}$$

Here U is an $N \times K$ matrix, Σ is a $K \times K$ matrix and V is a $P \times K$ matrix, and $K = \min(N, P)$.

$$X(X^T X)^{-\frac{1}{2}} = U V^T \tag{10}$$

3 Experiment

We implement the proposed method by TensorFlow [Abadi et al., 2015], and the code is available on Github[3].

3.1 Omniglot

We demonstrate the proposed method using the Omniglot Lake et al. [2015] dataset. Omniglot contains handwritten characters in 50 different alphabets (e.g., Cyrillic, Korean, Tengwar), each with its own number of unique characters ($14 \sim 55$). In total, there are 1623 unique characters, and each has exactly 20 instances. Here each task corresponds to an alphabet, and the goal is to recognise its characters.

The images are monochrome of size 105×105. We design a CNN with 3 convolutional and 2 FC layers. The first conv layer has 8 filters of size 5×5; the second conv layer has 12 filters of size 3×3, and the third convolutional layer has 16 filters of size 3×3. Each convolutional layer is followed by a 2×2 max-pooling. The first FC layer has 64 neurons, and the second FC layer has size corresponding to the number of unique classes in the alphabet. The activation function is *tanh*.

We compare the three variants of the proposed framework (based on Eq. 7), TNRDMTL-2 (based on Eq. 3), and TNRDMTL-3 (based on Eq. 5) with single task learning (STL). For every layer, there are one (TNRDMTL-1) or more (TNRDMTL-2 and TNRDMTL-3) γ that control the trade-off between the classification loss (cross-entropy) and the trace norm terms, for which we set all $\gamma = 0.01$.

The experiments are repeated 10 times, and every time 10% training data and 90% testing data are randomly selected. The results are shown in Table 1.

	STL	TNRDMTL-1	TNRDMTL-2	TNRDMTL-3
Mean Acc. (Std. Acc.)	34.52 (0.90)	36.07 (0.11)	35.91 (0.24)	35.55 (0.57)

Table 1: Mean Accuracies % in Omniglot Dataset

Besides, we plot the changes of cross-entropy loss in training set and the values of norm terms with the neural networks' parameters updating. As we can see in Fig 1, STL has a lower training loss, but its performance is the worst in Table 1, this indicates over-fitting. With the help of regularisation, MTL suffers less from this issue.

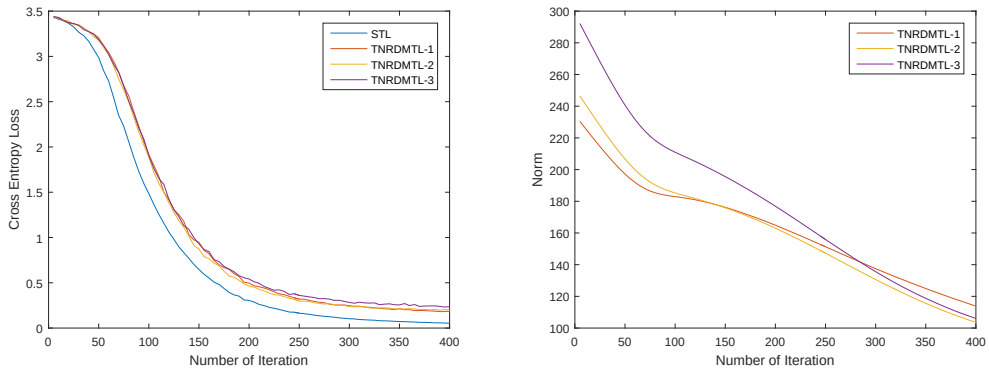

Figure 1: Cross-Entropy Loss (Left) and Norm (Right)

[3] `https://github.com/wOOL/TNRDMTL`

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 2008.

B. Bakker and T. Heskes. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research (JMLR)*, 2003.

R. Caruana. Multitask learning. *Machine Learning*, 1997.

S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *International Conference on Machine Learning (ICML)*, 2009.

A. Kumar and H. Daumé III. Learning task grouping and overlap in multi-task learning. In *International Conference on Machine Learning (ICML)*, 2012.

B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. *NAACL*, 2015.

I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 2011.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 2010.

B. Romera-paredes, H. Aung, N. Bianchi-berthouze, and M. Pontil. Multilinear multitask learning. In *International Conference on Machine Learning (ICML)*, 2013.

R. Tomioka, K. Hayashi, and H. Kashima. On the extension of trace norm to tensors. In *NIPS Workshop on Tensors, Kernels, and Machine Learning*, 2010.

L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 1966.

K. Wimalawarne, M. Sugiyama, and R. Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. In *Neural Information Processing Systems (NIPS)*, 2014.

Y. Yang and T. M. Hospedales. A unified perspective on multi-domain and multi-task learning. In *International Conference on Learning Representations (ICLR)*, 2015.

Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *European Conference on Computer Vision (ECCV)*, 2014.