

Multi-resolution Tensor Learning for Large-Scale Spatial Data

Stephan Zheng
California Institute of Technology
Pasadena, California
stephan@caltech.edu

Rose Yu
California Institute of Technology
Pasadena, California
rose@caltech.edu

Yisong Yue
California Institute of Technology
Pasadena, California
yyue@caltech.edu

ABSTRACT

High-dimensional tensor models are notoriously computationally expensive to train. We present a meta-learning algorithm, MRTL, that can significantly speed up the process for spatial tensor models. MRTL leverages the property that spatial data can be viewed at multiple resolutions, which are related by coarsening and finegraining from one resolution to another. Using this property, MRTL learns a tensor model by starting from a coarse resolution and iteratively increasing the model complexity. In order to not "over-train" on coarse resolution models, we investigate an information-theoretic fine-graining criterion to decide when to transition into higher-resolution models. We provide both theoretical and empirical evidence for the advantages of this approach. When applied to two real-world large-scale spatial datasets for basketball player and animal behavior modeling, our approach demonstrate 3 key benefits: 1) it efficiently captures higher-order interactions (i.e., tensor latent factors), 2) it is orders of magnitude faster than fixed resolution learning and scales to very fine-grained spatial resolutions, and 3) it reliably yields accurate and interpretable models.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**;

ACKNOWLEDGMENTS

This result is supported in part by NSF #1564330, NSF #1637598, and gifts from Bloomberg and Northrop Grumman.

1 INTRODUCTION

We study the problem of learning high-dimensional tensor models from large-scale high-resolution spatial data. Such models can compactly describe *multi-way* correlations between predictive features that are spatially distributed. For example, in competitive basketball play, given the positions of all players in the court, we can predict *whether a player will shoot at the basket* using a high-dimensional tensor model. Previous research on individual player models has demonstrated the effectiveness of matrix latent factor models [24]. For joint team behavior, we can model a basketball player's profile by learning the latent factors among offensive team positions, defending team positions and the player's own profile, which can be modeled by a *tensor latent factor model*.

This paper addresses two major challenges for learning spatial tensor models. First, we study *scalability*, as high resolution spatial tracking data is large-scale. Computational methods usually involve fine-grained spatial discretization, leading to high-dimensional tensors that are computationally slow and expensive to train.

In addition to scalability, we are also interested in learning *interpretable* models. Apart from accurate prediction, we also wish to learn latent factors that can provide insights for domain experts.

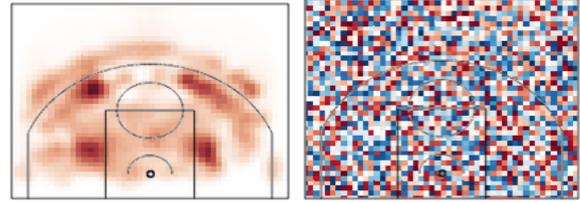


Figure 1: Comparison of learned latent factors for basketball shot prediction. Left: interpretable latent factors learned by our multi-resolution method. Right: uninterpretable factors learned by learning at fixed resolution. Despite similar accuracy, the uninterpretable profile is noisy and not spatially smooth, whereas the interpretable profile shows basketball shooting hotspots that are relevant for prediction.

For instance, the latent factors in basketball player models could correspond to player performance profiles, revealing hidden traits and commonalities among players.

Towards scalable learning of interpretable tensor latent factor models, a key technical challenge is their non-convex nature [23]: optimization algorithms can converge to different local optima, depending on the model initialization. Hence, a good initialization is critical for obtaining accurate and interpretable latent factors. For instance, Figure 1 shows two basketball shooting profiles with similar prediction accuracy, learned using the same optimization algorithm (gradient descent). Using a good initialization yields an interpretable solution, while a random initialization does not.

Our Contributions. Thus we are interested in answering how we can *efficiently* learn both *accurate* and *interpretable* tensor models from high-resolution spatial data. In this paper, we present a novel meta-learning algorithm: multi-resolution tensor learning (MRTL), that can 1) learn a good initialization, 2) automatically control when to fine-grain, and 3) easily scale to high-dimensional tensors. More importantly, the multi-resolution learning scheme yields interpretable solutions that are spatially smooth and relevant for the prediction tasks.

MRTL is based on three key insights. First, to obtain good initializations, instead of directly learning the latent factors from a random initialization, we can first learn a full-rank parameter tensor and use the factors from this parameter tensor as initialization. An analogous approach for matrix latent factor models has been shown in [12, 24] to yield interpretable factors capturing cohesive spatial semantics.

Second, to avoid the curse of dimensionality, we leverage the characteristics of spatial data and learn iteratively at *multiple resolutions*. At a high-level, the multi-resolution training has two stages: in the first stage, we learn a full tensor model using multi-resolution

and factorize it as initialization for the latent factor model. In the second stage, we train the latent factor model using multi-resolutions: starting at a coarse resolution and fine-graining the latent factors during training. We prove that the resulting algorithm is faster by a logarithmic factor of the Lipschitz constant of the objective function and the estimation error.

Third, we investigate fine-graining criteria. One simple criterion is to measure training loss convergence. However, it requires a pre-set threshold, which does not easily reflect the spatial distribution at different granularities. We thus take an information theoretical approach by monitoring the entropy of the gradients distribution for every grid cell, which we refer to as *spatial entropy*. We found this fine-graining criterion to be more effective than loss convergence.

In summary, our main contributions are:

- We present MRTL: a multi-resolution meta-algorithm to learn spatial latent factor models and a number of instantiations using various fine-graining criteria.
- We theoretically analyze MRTL and show that it converges faster to a similar accuracy than training at a fixed resolution.
- We empirically show that using gradient statistics, (e.g. gradient entropy), is an effective transition control method across hyperparameters and can outperform loss convergence.
- We empirically demonstrate orders-of-magnitude faster learning than conventional fixed-resolution training.
- We show that our approach reliably and efficiently yields interpretable spatial latent factor models on real-world basketball and animal tracking data.

2 RELATED WORK

Modeling spatial data enjoys a long history in the data mining community [13]. Recent applications include urban air quality prediction [5], social media recommendation [22], and event detection [25], among others. In this work, we study multi-agent tracking data extracted from high-fidelity tracking cameras. Such spatial data contains high resolution information and are large-scale (cameras operate at high frequency and every frame is one data point).

Latent factor models have been a popular method for spatial data modeling [3, 17, 24]. By projecting the raw data into a low-dimensional space, latent factor models can encode patterns such as spatial clustering. However, most existing latent factor models have only focused on capturing two-way interactions (such as user-location matrix models [10]). For multi-agent behavior modeling, we extend the matrix latent factor model to capture multi-way correlations among agents, resulting in a tensor latent factor model. For instance, we can build a “player \times offense position \times defense position” tensor model for basketball plays.

Tensor models have recently gained considerable attention (cf. [6, 16, 20]), but are still of limited applicability due to computational, data sparsity, and memory efficiency concerns. There are many efforts to scale up tensor computation, such as random projection (sketching) [21], parallel computing [1], or utilizing sparsity structure [15]. In this work, we take a top-down approach to learn a tensor model at multiple resolutions. The multi-resolution training procedure takes advantage of the spatial characteristics of the raw data, and is generically applicable to different tensor models and optimization algorithms.

Our approach bears affinity to other multi-stage meta-algorithms (e.g., [7]) and the multi-grid method in PDE analysis. For instance, [2] studies multi-grid methods in the context of dynamic programming. In the machine learning community, our method is also related to multi-resolution sparse approximation [11]. For example, [9] studies multi-resolution matrix factorization based on wavelets theory. [18] proposes a multi-resolution heuristic for tensor factorization. In contrast, we study multi-resolution learning for *tensor* models and also provide theoretical analysis for its convergence.

3 TENSOR LATENT FACTOR MODELS

We motivate the tensor latent factor model using the example of competitive basketball play. Suppose we have n_a players and want to predict whether a basketball player a ($a = 1 \dots n_a$) takes a shot at the basket (binary label $y_a = \pm 1$). We first discretize the court using a grid with m cells (positions $\mathbf{x} \in \mathbb{R}^m$) and then construct a binary feature vector $\phi(\mathbf{x}) \in \mathbb{R}^m$ using the positions of all other players on the court. The prediction problem can be formulated as:

$$P(y_a = 1|\mathbf{x}) = \langle \mathbf{w}_a, \phi(\mathbf{x}) \rangle + \mathbf{b}_a, \quad \phi(\mathbf{x}) = (0 \dots 1 \dots 0), \quad (1)$$

where $\mathbf{w}_a \in \mathbb{R}^m$ are the weight parameters for the grid cells and \mathbf{b}_a is the bias unit. The weights \mathbf{w}_a encodes how likely a basketball player will shoot from a given position on the basketball court.

Matrix Latent Factor Models. In matrix form, (1) is equivalent to the following model:

$$f_a(\mathbf{x}) = \sum_b W_{ab} \phi_b(\mathbf{x}) + \mathbf{b}_a, \quad (2)$$

where $f_a(\mathbf{x})$ is the prediction function for player a , and the matrix W concatenates the parameter vectors $\{\mathbf{w}_a\}$. Latent factor models assume there exist low-dimensional representations of the weights. Hence the parameter matrix factorizes into two matrices A and B with K components:

$$W_{ab} = \sum_k A_{ak} B_{bk}. \quad (3)$$

These latent factors can represent players’ behavioral profiles at different positions on the court.

Tensor Latent Factor Models. The limitation of the matrix latent factor model in (2) is that it can only capture the correlation between the ballhandler and all other players. In order to learn rich behaviors between teams, we need to explicitly model multi-way dependencies and generalize to high-order models. In the basketball example, we separately construct a feature vector $\phi(\mathbf{x}), \psi(\mathbf{x}) \in \mathbb{R}^d$ for the offense and defense team positions, respectively, and model the multi-way interactions as:

$$f_a(\mathbf{x}) = \sum_{bc} \mathcal{W}_{abc} \phi_b(\mathbf{x}) \psi_c(\mathbf{x}) + \mathbf{b}_a. \quad (4)$$

Here \mathcal{W} is an order 3 weight tensor. To encode low-dimensional structure, we assume the weight tensor \mathcal{W} factorizes:

$$\mathcal{W}_{abc} = \sum_{k=1}^K A_{ak} B_{bk} C_{ck}, \quad (5)$$

which corresponds to a CP tensor model [8]. In this case, K is called the *tensor rank*. In general, given some input \mathbf{x} and feature

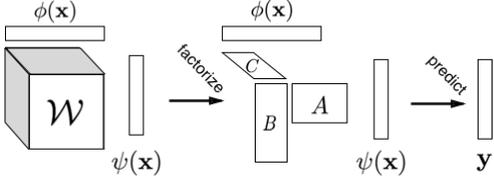


Figure 2: Prediction process using tensor latent factor model, from full-rank tensor to latent factor model.

transformation functions $\phi(\cdot)$ and $\psi(\cdot)$, *tensor latent factor models* aim to learn a function $f_a(\mathbf{x})$ for each prediction task a , such that:

$$f_a(\mathbf{x}) = \sum_{bc} \sum_{k=1}^K A_{ak} B_{bk} C_{ck} \phi_b(\mathbf{x}) \psi_c(\mathbf{x}) + \mathbf{b}_a, \quad (6)$$

where A, B, C are the latent factors, \mathbf{b}_a is a bias unit and a indexes prediction tasks. It is straightforward to generalize (6) to other tensor models and higher orders. Figure 2 depicts the model details.

The dimension of the weight tensor relates to the discretization size of the spatial data. The main motivation for considering discretizations is that they allow us to learn flexible non-parametric models. In contrast, traditional parametric models such as spatial point processes [12] have strong assumptions on the form of the spatial correlations (e.g. the generating process is a distribution $P(y|x)$ with multiple modes or has a cyclic factorization structure), which can be hard to learn using conventional learning methods.

Tensor models explicitly capture higher-order correlations among the tasks and spatial features. The learned latent factors can also be more interpretable due to the multi-linear nature (in the spatial feature dimensions). For example, if we see the prediction function $f(\mathbf{x}) = P(y|x)$ as a distribution and $\phi_b(\mathbf{x}), \psi_c(\mathbf{x})$ are 1-hot vectors that encode spatial occupancy, the columns $B_{\cdot,k}$ and $C_{\cdot,k}$ (for a fixed k) can be interpreted as smooth spatial probability distributions, representing players' shooting profile.

The cost of using such models, however, is that they lead to a hard non-convex optimization problem and can suffer from multiple local optima. Moreover, different initializations can lead to more or less interpretable models. They are also computationally expensive to train, making them infeasible for large-scale high resolution spatial data. Hence, the goal of our multi-resolution method is to efficiently find accurate and interpretable solutions in this setting.

4 MULTI-RESOLUTION LEARNING

We now describe our approach for training interpretable spatial latent factor models, such as the tensor model \mathcal{W} in (4). In this section, for simplicity we consider only fine-graining a single dimension of \mathcal{W} , although our analysis can be easily generalized.

Initialization from Factorization. The non-convex nature of the tensor model requires a good initialization. We obtain this by factorizing a trained full-rank tensor model as in (6) and using the factors as initializations. This approach has been shown to be effective in learning predictive spatial patterns [12, 24]. We also have similar observations in our experiments.

Iterative Fine-graining. It is generally not tractable in memory to learn a high-dimensional tensor latent factor model, that operates

at some high resolution. To make training feasible, the main idea of our algorithm is iterative fine-graining.

We will denote the *resolution* (i.e. size of a grid cell) as d_i , which is inversely proportional to the number of grid cells m_i .

During the first (full-rank) phase, we start training with a *full-rank model* \mathcal{W}_{d_0} that operates at a coarse resolution d_0 and increase the spatial resolution of the data and model using a sequence of resolutions d_0, d_1, \dots, d_f . That is, instead of learning $\mathcal{W} \in \mathbb{R}^{\dots \times m_f \times \dots}$, we learn a sequence of $\{\mathcal{W}_{d_i} \in \mathbb{R}^{\dots \times m_i \times \dots}\}$. When moving to a higher resolution, we use the learned weights \mathcal{W}_{d_i} to initialize the next weights $\mathcal{W}_{d_{i+1}}$, by scaling up \mathcal{W}_{d_i} along the spatial dimension.

We leverage the fact that spatial data can be viewed at multiple resolutions. Lower resolution models can provide good initializations for higher resolution models with high accuracy. To determine when to transition, we study different fine-graining criterion, which we discuss in section 4.2.

In the second phase, we apply a tensor factorization to \mathcal{W}_{d_i} at resolution d_f as initialization. We use a similar fine-graining procedure for the latent factor model to the final resolution d_n . The resolution d_f is determined by considering the memory requirements of the model at different resolutions. This meta-algorithm is outlined in Algorithm 1 and depicted in Figure 3.

4.1 Computational Complexity Analysis

We analyze the computational complexity for MRTL (Algorithm 1). Intuitively, as most of the training iterations are spent on coarser resolutions with fewer number of parameters, multi-resolution training is more efficient than fixed-resolution training.

In this section, we formalize this intuition and give a rigorous analysis of MRTL. The analysis is based on the multi-grid method [19] commonly used in partial differential equation analysis.

We denote the label $y \in \mathbb{R}^{n_a}$, features $\phi(x) \in \mathbb{R}^m$, $\psi(x) \in \mathbb{R}^{m'}$ and weight tensor $\mathcal{W} \in \mathbb{R}^{n_a \times m \times m'}$. The prediction model is:

$$f(x; \mathcal{W})_a = \sum_{ab} \mathcal{W}_{abc} \phi(x)_b \psi(x)_c + b_a. \quad (7)$$

Denote the objective function as $\mathcal{L}(x, y, f(x; \mathcal{W}))$, MRTL (Algorithm 1) aims to solve the optimization problem:

$$\min_{\mathcal{W}} \mathbb{E}_{(x, y) \sim P} [\mathcal{L}(x, y, f(x; \mathcal{W}))]. \quad (8)$$

where \mathcal{L} is the logistic loss. MRTL follows gradient descent:

$$\mathcal{W} \leftarrow \mathcal{W} - \lambda \nabla \mathcal{L} \mathcal{W}. \quad (9)$$

We first start with the coarsest resolution d_0 (discretization size), compute an initial estimate \mathcal{W}_{d_0} , and keep iterating until \mathcal{W}_{d_0} satisfies certain fine-graining criteria. Then we replace d_0 with $d_0/2$ and use \mathcal{W}_{d_0} as an initialization for the next level.

In general, this iterative fine-graining procedure yields a sequence of discretization sizes $[d_0, d_1, \dots, d_n]$. Suppose for each resolution d , we use the following as the fine-graining criterion:

$$\|\mathcal{W}^{t(d)} - \mathcal{W}^{t(d)-1}\| \leq \frac{C_0 d}{\alpha(1-\alpha)}. \quad (10)$$

where $t(d)$ is the number of iterations needed at level d . The algorithm terminates when the estimation error reaches $\frac{C_0 d}{(1-\alpha)^2}$. We now show that MRTL reduces the number of computation steps.

Algorithm 1 MRTL: Memory-efficient multi-resolution training with spatial entropy control

```

1: Input: Tensor weights  $\mathcal{W}_{d_0}$  (e.g. Equation (4)), data  $D$ , features  $\Psi$ .
2: for each resolution  $d_i \in \{d_1, \dots, d_f\}$  do
3:   # For definition of resolution, see Section 4.
4:   SGD-se( $\mathcal{W}_{d_i}$ ) # see Algorithm 2.
5: end for
6: TensorFactors $_{d_f}$  = Factorize( $\mathcal{W}_{d_f}$ ) # e.g.  $A, B, C$  in Equation (6).
7: for each resolution  $d_i \in \{d_{f+1}, \dots, d_n\}$  do
8:   SGD-se(TensorFactors $_{d_i}$ )
9: end for
10: return TensorFactors $_{d_n}$ 

```

Algorithm 2 SGD-se: Stochastic gradient descent with spatial entropy control

```

1: Input:  $\mathcal{W}_d$ , data-set  $D$ , features  $\Psi$ , length- $T$  rolling window gradient buffer  $H$ .
2: while termination condition (e.g. Condition (28)) not true do
3:   Gradient descent step on  $\mathcal{W}_{d_i}$  using minibatch  $B$ 
4:   Add gradient  $\{g(x, y)\}_{(x, y) \in B}$  to  $H$ 
5: end while
6: Finegrain( $\mathcal{W}_{d_i}$ )
7: return  $\mathcal{W}_{d_{i+1}}$ 

```

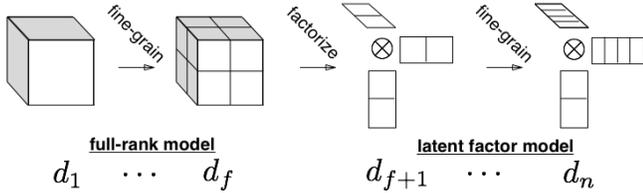


Figure 3: Depicting our multi-stage training, which starts with a coarse-grained full-rank model, and concludes on a fine-grained latent factor model. See Algorithm 1 and Algorithm 2 for more details.

This is done by first formulating a gradient descent update as a fixed point iteration operator F^1 :

$$\mathcal{W} \leftarrow F(\mathcal{W}), \quad F := I - \lambda \nabla \mathcal{L}. \quad (11)$$

Assume the objective function is Lipschitz continuous, with a contraction constant of $\alpha \in (0, 1)$, meaning:

$$\|F(\mathcal{W}) - F(\mathcal{W}')\| \leq \alpha \|\mathcal{W} - \mathcal{W}'\|. \quad (12)$$

The following main theorem characterizes the speed-up gained by multi-resolution training w.r.t. the contraction factor α and the terminal estimation error ϵ .

THEOREM 4.1. *Suppose the fixed point iteration operator (gradient descent) for the optimization algorithm has a contraction factor (Lipschitz constant) of α , the multi-resolution training procedure is faster than that of the fixed resolution algorithm by a factor of $\log \frac{1}{(1-\alpha)\epsilon}$, with ϵ as the terminal estimation error.*

¹Stochastic gradient descent converges to a noise ball instead of a fixed point.

We prove several useful Lemmas before proving the main Theorem 4.1. We first analyze the computational cost of the *fixed-resolution* algorithm.

LEMMA 4.2. *Given a fixed point iteration operator with a contraction factor of α , the computational complexity of a fixed-resolution training for a p -order tensor with rank r is*

$$T = O\left(\frac{1}{|\log \alpha|} \cdot \log \frac{1}{(1-\alpha)\epsilon} \cdot \left(\frac{rp}{(1-\alpha)^2\epsilon}\right)\right). \quad (13)$$

PROOF. At a high level, we prove this by choosing a small enough resolution d such that the approximation error is bounded with a fixed number of iterations. Let \mathcal{W}_d^* be the optimal estimate at level d and \mathcal{W}^t be the estimate at step t . Then we have

$$\|\mathcal{W}^* - \mathcal{W}^t\| \leq \|\mathcal{W}^* - \mathcal{W}_d^*\| + \|\mathcal{W}_d^* - \mathcal{W}^t\| \leq \epsilon. \quad (14)$$

Choose a fixed resolution d that is small enough such that

$$\|\mathcal{W}^* - \mathcal{W}_d^*\| \leq \frac{\epsilon}{2}, \quad (15)$$

then the termination criteria $\|\mathcal{W}^* - \mathcal{W}_d^*\| \leq \frac{C_0 d}{(1-\alpha)^2}$ gives $d = \Omega((1-\alpha)^2\epsilon)$. Initialize $\mathcal{W}^0 = 0$ and iterate over t times such that:

$$\frac{\alpha^t}{2(1-\alpha)} \|F(\mathcal{W}^0)\| \leq \frac{\epsilon}{2}, \quad (16)$$

Since $\mathcal{W}^0 = 0$, $\|F(\mathcal{W}^0)\| \leq 2C$, we obtain that

$$t \leq \frac{1}{|\log \alpha|} \cdot \log \frac{2C}{(1-\alpha)\epsilon}, \quad (17)$$

Note that for an order p tensor with rank r , the computational complexity of every iteration in MRTL is $O(rp/d)$ with d as the discretization size. Hence, the computational complexity of the fixed resolution training is

$$\begin{aligned} T &= O\left(\frac{1}{|\log \alpha|} \cdot \log \frac{1}{(1-\alpha)\epsilon} \cdot \left(\frac{rp}{d}\right)\right) \\ &= O\left(\frac{1}{|\log \alpha|} \cdot \log \frac{1}{(1-\alpha)\epsilon} \cdot \left(\frac{rp}{(1-\alpha)^2\epsilon}\right)\right). \quad \square \end{aligned}$$

In the multi-resolution setting, the spatial weights that are learned can be seen as a distribution that is approximated by its values at finitely many points. Given a spatial discretization d , we can construct an operator F_d that learns discretized tensor weights. The next Lemma relates the estimation error with resolution:

LEMMA 4.3. [14] *For each resolution level $[d_0, d_1, \dots, d_n]$, there exists a constant C_1 and C_2 , such that the fixed point iteration with the discretization size d has an estimation error:*

$$F(\mathcal{W}) - F_d(\mathcal{W}) \leq C_1 + \alpha C_2 \|\mathcal{W}\|_d. \quad (18)$$

PROOF. See [14] for the details. \square

We have obtained the discretization error for the fixed point operation at any resolution. Next we analyze the number of iterations $t(d)$ needed at each resolution d before fine-graining.

LEMMA 4.4. *For every resolution $d \in [d_0, d_1, \dots, d_m]$, there exists a constant C' , such that the number of iterations $t(d)$ before fine-graining satisfies:*

$$t(d) \leq C' / \log |\alpha|. \quad (19)$$

PROOF. We know that $d_0 = 2d_1 = 4d_2 = \dots$. At resolution d , by using the estimate from the last level as initialization, we have:

$$\mathcal{W}_{2d}^{t(2d)} = \mathcal{W}_d^0. \quad (20)$$

where we use the subscript to index the solution at a given resolution. By combining Lemma 18 and the fine-graining criteria in (10), we can guarantee the estimation error per iteration:

$$\begin{aligned} \|F_d(\mathcal{W}_d^0) - \mathcal{W}_d^0\| &= \|F_d(\mathcal{W}_{2d}^{t(2d)}) - \mathcal{W}_{2d}^{t(2d)}\| \\ &\leq \|F_d(\mathcal{W}_{2d}^{t(2d)}) - F(\mathcal{W}_{2d}^{t(2d)})\| + \|F(\mathcal{W}_{2d}^{t(2d)}) - \mathcal{W}_{2d}^{t(2d)}\| \\ &\leq (C_1 + \alpha C_2 \|\mathcal{W}_{2d}^{t(2d)}\|)d \\ &+ \|F(\mathcal{W}_{2d}^{t(2d)}) - F_{2d}(\mathcal{W}_{2d}^{t(2d)})\| + \|F_{2d}(\mathcal{W}_{2d}^{t(2d)}) - \mathcal{W}_{2d}^{t(2d)}\| \\ &\leq (C_1 + \alpha C_2 \|\mathcal{W}_{2d}^{t(2d)}\|)d + (C_1 + \alpha C_2 \|\mathcal{W}_{2d}^{t(2d)}\|)2d + \alpha \frac{C_0 d}{\alpha(1-\alpha)}. \end{aligned}$$

According to the fixed point iteration definition, we have:

$$\|F_d(\mathcal{W}^{t(d)}) - \mathcal{W}^{t(d)}\| \leq \alpha^{t(d)-1} \|F_d(\mathcal{W}_d^0) - \mathcal{W}_d^0\| \leq C'd$$

Thus the number of iterations satisfies $t(d) \leq C'/\log|\alpha|$ for all resolutions. \square

Proof of Theorem 4.1. By combining Lemmas 4.4 and the computational cost per iteration, we can compute the total computational cost for our MRTL algorithm, which is proportional to the total number of iterations for all resolutions:

$$\begin{aligned} T &= O\left(\frac{1}{|\log \alpha|} \left[(d_n/rp)^{-1} + (2d_n/rp)^{-1} + (4d_n/rp)^{-1} + \dots \right]\right) \\ &= O\left(\frac{1}{|\log \alpha|} \left(\frac{rp}{d_n}\right) \left[1 + \frac{1}{2} + \frac{1}{4} + \dots\right]\right) \\ &= O\left(\frac{1}{|\log \alpha|} \left(\frac{rp}{d_n}\right) \left[\frac{1 - (\frac{1}{2})^n}{1 - \frac{1}{2}}\right]\right) \\ &= O\left(\frac{1}{|\log \alpha|} \left(\frac{rp}{(1-\alpha)^2 \epsilon}\right)\right), \quad (21) \end{aligned}$$

where the last step uses the termination criterion in (10) $d_n = \Omega((1-\alpha)^2 \epsilon)$. Comparing with the complexity analysis for the fixed resolution algorithm in Lemma 4.2, we complete the proof. \square

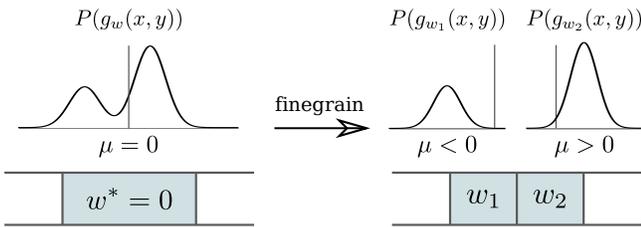


Figure 4: Gradient distribution for a toy model during multi-resolution training. At the coarse level, the distribution has $\mu = 0$ at convergence, whereas at a higher resolution, the distributions for the finegrained weights have non-zero mean.

4.2 Fine-Graining Criteria

Given the structure of MRTL (Algorithm 1), the key technical question is which fine-graining criterion to use at each stage. In Theorem 4.1, we showed the improved convergence speed when using loss-convergence as a transition criterion. We now propose several fine-graining criteria to instantiate MRTL, that empirically outperform loss-convergence.

Loss convergence. Intuitively, one simple fine-graining condition is when training at the present resolution converges (which mimics the termination condition for iterative training of sparse models [7]). That is, at training step t , we check whether:

$$|\mathcal{L}^t - \mathcal{L}^{t-1}| < \tau_L, \quad (22)$$

where \mathcal{L}^t is the historical mean loss and τ_L is a threshold. However, since the model at each resolution is used to initialize the training for the next resolution, the converged model at the current resolution might not be the best initialization for the next resolution (i.e., training might be overfitting to the coarser resolution).

Using Gradient Statistics. Given the spatial nature of the model weights (6), we can use more finegrained information during training by analyzing the gradient distribution during training. Intuitively, spatial resolution d_i is too coarse when the data prefers much more fine-grained curvature, as evidenced by substantial disagreement in the gradients between resolution d_i and d_{i+1} . For example, consider learning a linear binary classifier f as in (1):

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \mathbb{E}_{(x,y) \sim P} [\mathcal{L}(x, y, f(x; \mathbf{w}))], \quad (23)$$

where $x \in \mathbb{R}$ and we consider two resolutions: $m_1 = 1$ cells and $m_2 = 2$ cells. Suppose we optimize \mathbf{w} using

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta \mathbf{w}, \quad \Delta \mathbf{w} = -\lambda h(g_{\mathbf{w}}; \theta), \quad (24)$$

where λ is the learning-rate, h is a (nonlinear) optimizer with parameters θ and the true gradient is

$$g_{\mathbf{w}} = \mathbb{E}_{(x,y) \sim P} [\nabla_{\mathbf{w}} \mathcal{L}(x, y, f(x; \mathbf{w}))]. \quad (25)$$

Consider the situation that 1) during training at resolution d_1 the model has converged to $\mathbf{w}^* = 0$ and 2) the ground truth model at resolution d_2 is $\mathbf{w}^* = (+1, -1)$. In this case, at the coarse resolution d_1 the distribution $P(g_{\mathbf{w}}(x, y))$ of the point-wise gradient $g_{\mathbf{w}}(x, y) = \nabla_{\mathbf{w}} \mathcal{L}(x, y, f(x; \mathbf{w}))$ has mean $\mu \approx 0$ (and some variance $\sigma > 0$). However, if we fine-grain the model, the (distribution of) gradients for the higher-resolutions weights w_1 and w_2 have non-zero mean: the mean gradient $g_{w_1}(x, y)$ will be negative and $g_{w_2}(x, y)$ positive. This is illustrated in Figure 4.

More generally, we can quantify disagreement between gradients at multiple resolutions via the statistics μ, σ and entropy S of the (empirical) gradient distribution of *all* weights \mathbf{w} at resolutions d_i :

$$S(g_{\mathbf{w}^i}) = \mathbb{E}_{g_{\mathbf{w}}(x,y)} [\log P(g_{\mathbf{w}}(x, y))]. \quad (26)$$

Intuitively, when the entropy $S(g_{\mathbf{w}^i})$ is high, the gradients of the current discretization are increasingly disagreeing with each other and training can benefit from finegraining. More formally, we can express this via the information gain:

$$I_{i, i+1} = S(g_{\mathbf{w}^i}) - S(g_{\mathbf{w}^{i+1}}), \quad (27)$$

Name	Range	Name	Range
Learning-rate λ	$10^{-5} - 10^{-1}$	τ_I	$10^{-6} - 10^{-2}$
Gradient buffer window T	10 - 1000	τ_S	$10^{-6} - 10^{-0}$
L_1 -regularization	$10^{-5} - 10^{-1}$	τ_μ	$10^{-8} - 10^{-2}$
L_2 -regularization	$10^{-5} - 10^{-1}$	τ_σ	$10^{-8} - 10^{-1}$
Criterion check frequency ω	10 - 1000		

Table 1: Search range for SGD-se hyperparameters and fine-graining criterion hyperparameters.

If $I_{i,i+1} > 0$, the gradient will have lower entropy at the higher resolution and finegraining could be beneficial.²

However, in practice it is infeasible to compute information gain, as this requires gradient statistics at a higher resolution. As a proxy to (27) we can instead use a simpler condition: finegrain when for $\geq p\%$ of the weights in \mathbf{w} the entropy S exceeds a margin τ :

$$\text{for } \geq p\% \text{ weights } w_j : S(g_{w_j^i}) > \tau, \quad (28)$$

where τ and p are tunable hyperparameters.

Moment-based thresholds. We can define other criteria based on gradient statistics: finegrain if for at least $p\%$ of the weights:

$$\sigma\text{-threshold: } \sigma_t > \tau_\sigma, \quad (29)$$

$$\mu, \sigma\text{-threshold: } \sigma_t > \tau_\sigma \text{ and } |\mu_t| < \tau_\mu. \quad (30)$$

where μ_t, σ_t^2 are the gradient mean and variance at step t and $\tau.s$ are tunable hyperparameters. σ -thresholding is a more coarse statistic compared to the entropy S : if the gradient distribution is non-Gaussian or multimodal S can capture higher-order statistics as well. μ, σ -thresholding also tracks whether training has converged ($\mu \approx 0$) according to the gradients.

5 BENCHMARK EXPERIMENTS

To validate our approach, we demonstrate that our multi-resolution method converges faster than fixed-resolution training on two real-world datasets: basketball shots and fruit-fly behavior prediction. Furthermore, we show that using spatial gradient statistics, such as its entropy, outperforms using loss convergence and provide a sensitivity analysis of the various transition criteria. Finally, in Section 6, we show that our learned models are interpretable.

For both datasets, we learn a tensor model for *multi-task* prediction, where we instantiate the model (6) as a sum of low-rank \mathcal{W}^L and sparse \mathcal{W}^S tensors:

$$P(y_a|\mathbf{x}) = \sum_{abc} (\mathcal{W}_{abc}^L + \mathcal{W}_{abc}^S) \phi_b(x) \psi_c(x), \quad (31)$$

$$\mathcal{W}_{abc}^L = \sum_{k=1}^K A_{ak} B_{bk} C_{ck}, \quad \mathcal{W}_{abc}^S = \sum_{k=1}^K U_{ak} V_{bk} W_{ck},$$

where $a \in \mathcal{A}$ indexes the tasks. A key motivation for this decomposition is that L, S can capture semantically meaningful profiles (spatially dense and smooth, or sparse and peaky).

²This is analogous to the use of information gain for decision tree regularization.

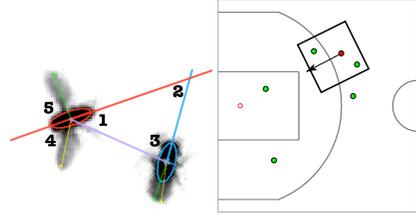


Figure 5: Left: Fly-vs-Fly angles: facing angles (1), (3), inter-fly angle (2), minimal (4) and maximal wing angle (5). Right: sample frame with ballhandler (red) and defensive players (green). Only players close to the ballhandler are used.

Empirical Gradient Distribution. During training, we estimate the gradient distribution $P(g_w(x, y))$ by recording the empirical minibatch gradients and their statistics μ, σ, S over a rolling window of T training steps. While collecting gradients, the weights \mathbf{w} are typically updated too, introducing a bias in the gradient statistics, as gradients are computed for different models at each step. However, we found empirically that using (28) remains effective.

Experiments. First, we compared MRTL with fixed-resolution training. Secondly, we compared the various fine-graining criteria (entropy divergence, loss convergence, σ -threshold, μ, σ -threshold) as described in section 4.2. We instantiated MRTL by adaptively upscaling the model B, C, V, W and features ϕ_b and ψ_c as in Algorithm 1, checking transition criteria each $\omega = 100$ steps for $p = 10\%$ and various τ . All models were trained using Adam using cross-entropy loss and L_2 (L_1) regularization for the dense (sparse) factors. We selected optimal hyperparameters for all models using a grid search over hyperparameters (see Table 1). For the sensitivity analysis, we used the $\tau.s$ as in Table 1.

5.1 Datasets

Basketball tracking data. We evaluated our method on *basketball shot prediction*, where the tasks correspond to unique basketball players, indexed by a , and our multi-task model predicts *whether player a will shoot at the basket immediately after frame \mathbf{x}_α* (α indexes the dataset). We used a large player tracking dataset *BBSHOT* [24, 26] that includes hundreds of players and covers millions of game frames captured during competitive basketball gameplay. For each frame \mathbf{x}_α , we have binary labels $y_{\alpha,a} \in \{-1, +1\}$: whether player a will (not) shoot at the basket in frame \mathbf{x}_α . The features are 1-hot vectors $\phi(\mathbf{x}_\alpha) \in \{0, 1\}^{2000}$ for the location b of the ballhandler and $\psi(\mathbf{x}_\alpha) \in \{0, 1\}^{144}$ for the defenders in a 12×12 grid around the ballhandler. At full resolution, the court is discretized using a 50×40 grid of 2000 cells of size 1×1 . For lower resolutions, we used $2 \times 2, 3 \times 3$ and 4×4 cells.

Caltech Fly-vs-Fly. The Caltech Fly-vs-Fly behavior dataset *Fly-vs-Fly* [4] consists of several million video-frames of 2 fruit-flies, labeled by neuro-biologists with 10 behavioral fly actions: *touch, wing threat, charge, lunge, hold, tussle, wing extension, circle, copulation attempt, copulation*. The goal is to predict whether either fly performs any of these 10 actions in video frame \mathbf{x}_α . Here, the tasks are the actions indexed by a and the binary labels $y_{\alpha,a} \in \{-1, +1\}$ correspond to whether an action a is present in frame \mathbf{x}_α or not.

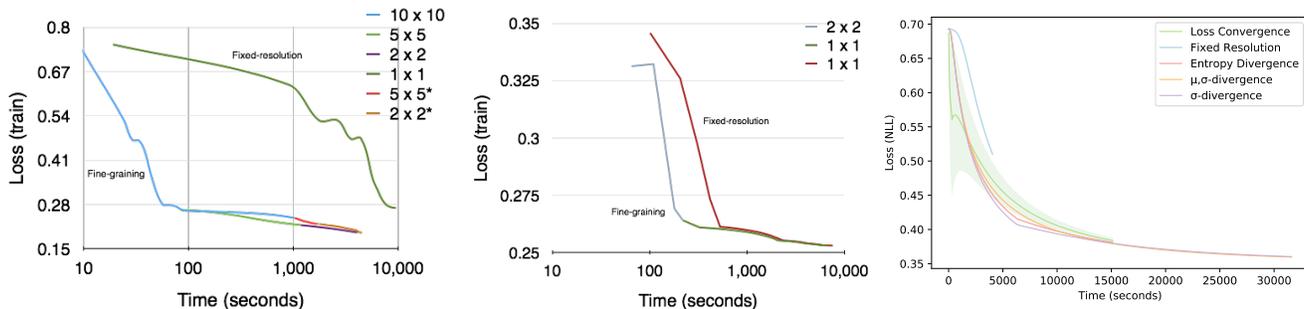


Figure 6: Left: training a full-rank tensor model using MRTL (1) with $\omega = 100$ on *BBSHOT*. Colors indicate training stages. For visual clarity, only best runs for fixed-resolution, loss-convergence and entropy-thresholding are shown. Baseline 1 (dark green): training a fixed-resolution fine-grained model is orders of magnitude slower. Baseline 2 (indicated by *): fine-graining using gradient entropy control (green, purple) with 20 histogram bins converges faster than loss-convergence (red, orange). Middle: training a factored tensor model using iteratively fine-graining outperforms a fixed-resolution approach. Right: Learning a factorized model (31) on *Fly-vs-Fly*. Using gradient statistics converges $\approx 50\%$ faster than fixed-resolution learning.

Loss-threshold:	0.62	Full-rank	0.64	Factor
Criterion	Minimal time (s)	Mean	Minimal	Mean
Fixed Resolution	18358	-	106	-
Loss-conv	639	10141	96	99
Entropy-threshold	541	4258	40	42
σ -threshold	356	2453	55	56
μ, σ -threshold	292	677	880	894

Table 2: Minimal and mean time to reach a loss when overfitting starts on *BBSHOT-200k* across 20 runs, with full and factorized model (6) (as in Figure 7). Using gradient statistics reaches the loss threshold consistently significantly faster.

The dataset includes 12 pose and spatial features, including the *velocity and the wing angles* of a single fly, and pairwise features, such as the *distance and angle* between the two flies (see Figure 5).

5.2 Accuracy and Convergence Speed

We compare MRTL with fixed-resolution learning. Moreover, we compare a number of fine-graining criteria: fine-graining when the loss has converged versus spatial entropy control. For the latter, we used condition (28) to determine when to fine-grain. Figure 6 and Table (2) show our results using models with total latent dimension 20.³ The left plot shows the results for the full-rank model, which is usually the more computationally intensive part. We see that our multi-stage approach dramatically outperforms (by multiple orders-of-magnitude) a naive approach which only uses the finest resolution.⁴ Moreover, spatial entropy control outperforms, by an order-of-magnitude, using the loss as a termination criterion.

The right plot shows the performance of MRTL on the latent factor model after initializing using a factorized model of the previous stage. We see that the learning objective continues to decrease as learning enters what is essentially a fine-tuning phase.⁵

³Performance trend is stable for a wide range of latent dimensions, omitted for brevity.

⁴Note that for more complex models, the naive approach would not even fit in memory.

⁵Note the absolute objective of the un-factorized model is lower than the latent-factor model because the former has more degrees of freedom and is overfitting.

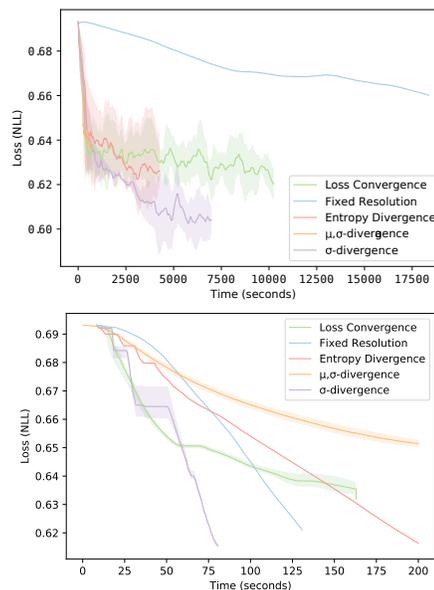


Figure 7: Sensitivity of MRTL for 4 transition criteria (e.g. (28)) to hyperparameters τ and $p = 10\%$ on *BBSHOT-200k* using a full-rank (top) and a factorized model (6) (bottom). For each criterion, we sampled 20 τ s uniformly and show the mean and variance of the converging runs as they start to overfit.

5.3 Sensitivity Analysis

To evaluate the efficiency and transition behavior using the transition criteria from Section 4.2, we evaluated their sensitivity to the threshold parameters τ . For this, we used *BBSHOT-200k* with 200k *BBSHOT* examples and evaluated Algorithm 1 using a random search over τ . The results are in Table 2 and Figure 7. We see that using

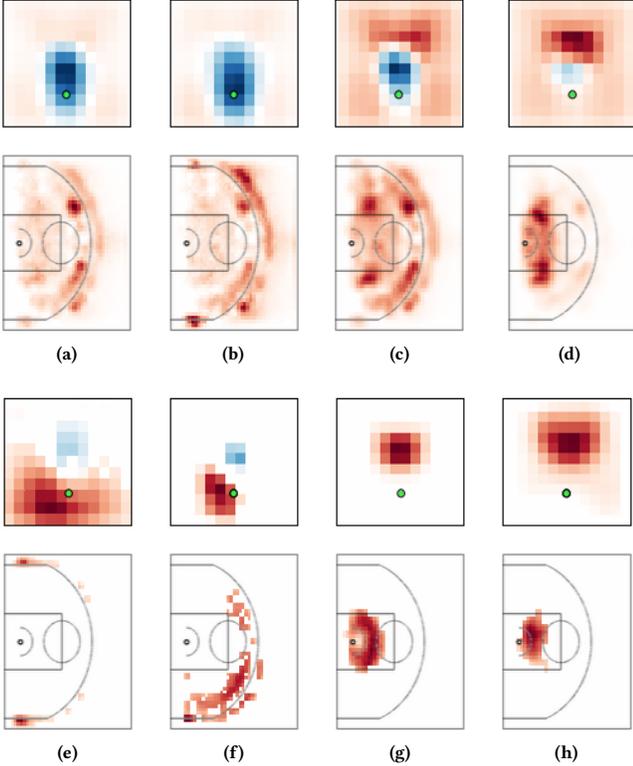


Figure 8: Row 1: smooth defender profiles C_{ck} ($k = 1, 2, 3, 4$), row 3: sparse W_{ck} ($k = 5, 6, 7, 8$) showing how defenders around the ballhandler influence the probability of a shot at the basket. The ballhandler is located at $(6, 3)$ (green marker) and the vertical points towards the basket. Row 2: smooth ballhandler shooting profiles B_{bk} ($k = 1, 2, 3, 4$), row 4: sparse V_{bk} ($k = 5, 6, 7, 8$). The basket is located at the dot at $(5.25, 20)$ and the 3-point line is the large arc. All profiles are normalized (red: positive, blue: negative).

gradient statistics ((28), (29)) consistently outperforms loss convergence in the short-term (σ -threshold) and long-term (entropy-threshold). We empirically find that μ, σ -threshold is harder to stabilize than other criteria, although it can outperform other methods.

6 ANALYSIS OF INTERPRETABLE SOLUTIONS

We now qualitatively evaluate a learned model (31) that demonstrates that our meta-algorithm MRTL (Algorithm (1)) can learn compact representations of semantic knowledge. We show that:

- The latent factors are interpretable: they capture characteristic basketball shooting profiles and spatial fly behavior.
- Smooth and sparse latent factors correspond to various types of basketball behavior and spatial configurations of fruit flies.

6.1 Basketball shot prediction

Ball handler shooting profiles. In competitive basketball play, shots at the basket happen throughout the court and peak at certain

hot-spot locations at short range (close to the basket), medium range (between the basket and 3-point line) and long range (at the 3-point line or beyond). Inspecting the smooth shooting profiles in Figure 8, we see that all profiles are spatially cohesive and have small peaks around different subsets of hot-spots. Profiles 1, 2 and 3 capture medium range and long range hot-spots, while profile 4 covers the short-range zone. In contrast, the sparse shooting profiles are more spatially concentrated and activate only on specific hot-spots. For example, profile 5 covers 2 hot-spots to the far left and right side at the back of the court and profiles 7 and 8 only activate on the short range zone directly around the basket.

This leads to an attractive semantic interpretation: smooth shooting profiles describe *inconsistent players* that tend to shoot from many locations throughout the basketball court, whereas sparse shooting profiles capture *consistent players* that shoot from only specific hot-spots. Figure 9 show the latent factor activations A_{ak} and $U_{a,k}$ of 6 players that can be grouped as such: players 1, 2, and 3 are *consistent*; players 4, 5, and 6 are more *inconsistent*.

Defender influence profiles. The bottom row of Figure 8 depicts four dense defender profiles C and four sparse defender profiles W . The first two dense profiles capture defender suppression in front of the ballhandler across the entire court, since the companion shooting profiles (1 and 2) are diffused throughout. In contrast, the first two sparse defender profiles (5 and 6) describe defender influence specifically at long range, which has a more peaked behavior. The last two sparse profiles (7 and 8) describe ballhandlers that are prone to shoot with a defender close by. This is likely due to confusing correlation with causation, since players close to the basket tend to shoot, even though typically a defender is close by.

6.2 Fruit fly behavior

Configuration profiles. Figure 11 depicts a sample dense and a sparse learned profile that are interpretable. In the dense profile, the fly extends *at least one wing* which defines *wing extension*, a sign of courtship. The sparse profile similarly captures more concentrated spatial characteristics, when the flies are close. Such wing configurations are important in both aggressive and courtship behavior, when the two flies interact closely to each other.

Actions and behavioral types. Figure 10 provides a more holistic view across all actions in the *Fly-vs-Fly* dataset. For example, we see that *lunge* and *wing threat* have similar activations. This is natural: the former is active physical aggression, while the latter can be interpreted as a fly showing a preview of physical aggression to intimidate the other fly. We observe that both dense and sparse profiles are useful for modeling a wide range of actions/tasks.

7 DISCUSSION

Our adaptive multi-resolution approach learns models at a fixed *global* resolution at each stage. An open problem remains how to learn at different resolutions *locally*, i.e. by finegraining individual cells rather than all. Such methods could be used to learn the dynamics in systems that intrinsically behave differently at different resolution, such as many dynamical systems. Local gradient statistics could be used in this case to decide when and which cells to finegrain to optimize model performance.

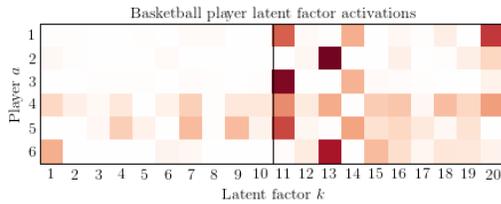


Figure 9: Basketball player latent factor activation weights A_{ak} for the same model as in Figure 8 with 10 smooth factors ($k = 1 \dots 10$) and 10 sparse factors ($k = 11 \dots 20$).

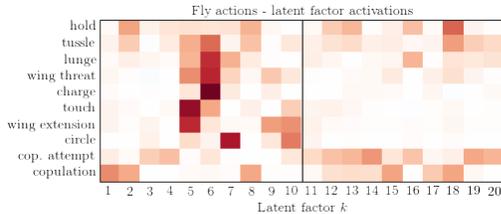


Figure 10: Fly action latent factor activation weights A_{ak} for a model with 10 smooth ($k = 1 \dots 10$) and 10 sparse latent factors ($k = 11 \dots 20$). Similar actions activate similar factors.

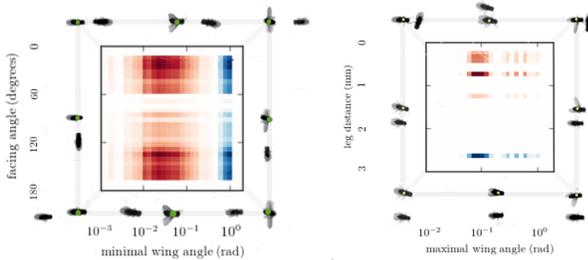


Figure 11: Left: Smooth normalized profile in $B_{bk}C_{ck}$ over facing angles and wing angles. Right: Sparse profile. Red (blue) means positive (negative). Example spatial configurations of a pair of flies are shown for the extremal regions of the profile. The reference fly (green) always points to the right, with the second fly at various facing angles. The dense profile describes a fly that extends exactly one wing, the maximal (minimal) wing length is large (small) and is turned towards the other fly. The sparse profile activates highly when the legs of the two flies are close and the reference fly keeps both wings close to his body.

REFERENCES

- [1] Woody Austin, Grey Ballard, and Tamara G Kolda. 2016. Parallel tensor compression for large-scale scientific data. In *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, 912–922.
- [2] C-S Chow and John N Tsitsiklis. 1991. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE transactions on automatic control* 36, 8 (1991), 898–914.
- [3] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. 2016. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1525–1534.
- [4] Eyrun Eyjolfssdottir, Steve Branson, Burgos-A, Xavier P Rtizzu, Eric D Hooper, Jonathan Schor, David J Anderson, and Pietro Perona. 2014. Detecting Social Actions of Fruit Flies. *Computer Vision & ECCV 2014* (2014), 772–787.

- [5] Hsun-Ping Hsieh, Shou-De Lin, and Yu Zheng. 2015. Inferring air quality for station location recommendation based on urban big data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 437–446.
- [6] Rodolphe Jenatton, Antoine Bordes, Nicolas Le Roux, and Guillaume Obozinski. 2012. A Latent Factor Model for Highly Multi-relational Data. *Advances in Neural Information Processing Systems* 25 (2012), 3167–3175.
- [7] Tyler B Johnson and Carlos Guestrin. 2015. BLITZ: A Principled Meta-Algorithm for Scaling Sparse Optimization. In *International Conference on Machine Learning (ICML)*.
- [8] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev* 51, 3 (2009), 455–500. <https://doi.org/10.1137/07070111X>
- [9] Risi Kondor, Nedelina Teneva, and Vikas Garg. 2014. Multiresolution matrix factorization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 1620–1628.
- [10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.
- [11] Stephane G Mallat. 1989. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbb{R})$. *Transactions of the American mathematical society* 315, 1 (1989), 69–87.
- [12] Andrew Miller, Luke Bornn, Ryan Adams, and Kirk Goldsberry. 2014. Factorized Point Process Intensities: A Spatial Analysis of Professional Basketball. In *International Conference on Machine Learning (ICML)*.
- [13] Harvey J Miller and Jiawei Han. 2009. *Geographic data mining and knowledge discovery*. CRC Press.
- [14] Stephen G Nash. 2000. A multigrid approach to discretized optimization problems. *Optimization Methods and Software* 14, 1-2 (2000), 99–116.
- [15] Ioakeim Perros, Evangelos E Papalexakis, Fei Wang, Richard Vuduc, Elizabeth Searles, Michael Thompson, and Jimeng Sun. 2017. SPARTan: Scalable PARAFAC2 for Large & Sparse Data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 375–384.
- [16] Ariadna Quattoni and Xavier Carreras. 2014. Spectral Regularization for Max-Margin Sequence Tagging. *Proceedings of The 31st International Conference on Machine Learning* 32 (2014).
- [17] Brian J Reich and Dipankar Bandyopadhyay. 2010. A latent factor model for spatial data with informative missingness. *The annals of applied statistics* 4, 1 (2010), 439.
- [18] Claudio Schifanello, K Selçuk Candan, and Maria Luisa Sapino. 2014. Multiresolution tensor decompositions with mode hierarchies. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 8, 2 (2014), 10.
- [19] Klaus Stüben. 2001. A review of algebraic multigrid. In *Partial Differential Equations*. Elsevier, 281–309.
- [20] Koh Takeuchi, Ryota Tomioka, Katsuhiko Ishiguro, Akisato Kimura, and Hiroshi Sawada. 2013. Non-negative multiple tensor factorization. In *IEEE International Conference on Data Mining (ICDM)*.
- [21] Yining Wang, Hsiao-Yu Tung, Alexander J Smola, and Anima Anandkumar. 2015. Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*. 991–999.
- [22] Hongzhi Yin and Bin Cui. [n. d.]. *Spatio-temporal recommendation in social media*. Springer.
- [23] Rose Yu and Yan Liu. 2016. Learning from multiway data: Simple and efficient tensor regression. In *International Conference on Machine Learning*. 373–381.
- [24] Yisong Yue, Patrick Lucey, Peter Carr, Alina Bialkowski, and Iain Matthews. 2014. Learning Fine-Grained Spatial Models for Dynamic Sports Play Prediction. In *IEEE International Conference on Data Mining (ICDM)*.
- [25] Liang Zhao, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2016. Hierarchical incomplete multi-source feature learning for spatiotemporal event forecasting. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2085–2094.
- [26] Stephan Zheng, Yisong Yue, and Jennifer Hobbs. 2016. Generating long-term trajectories using deep hierarchical networks. In *Advances in Neural Information Processing Systems*. 1543–1551.

Multi-resolution Tensor Learning for Large-Scale Spatial Data

Stephan Zheng
California Institute of Technology
Pasadena, California
stephan@caltech.edu

Rose Yu
California Institute of Technology
Pasadena, California
rose@caltech.edu

Yisong Yue
California Institute of Technology
Pasadena, California
yyue@caltech.edu

1 SUPPLEMENTARY MATERIAL

1.1 Computational Complexity Analysis

LEMMA 1.1. *Given a fixed point iteration operator with a contraction factor of $\alpha \in (0, 1)$, the computational complexity of a fixed-resolution training for a P dimensional tensor with rank r is*

$$\mathcal{O}\left(\frac{1}{|\log \alpha|} \cdot \frac{1}{\log(1-\alpha)\epsilon} \cdot \left(\frac{1}{rp(1-\alpha)^2\epsilon}\right)\right)$$

PROOF. at high level, we prove this by choosing a small enough resolution d such that the approximation error is bounded with a fixed number of iterations.

Let \mathcal{W}_d^* be the optimal estimate at level d and \mathcal{W}^t be the estimate at step t . Then we have

$$\|\mathcal{W}^* - \mathcal{W}^t\| \leq \|\mathcal{W}^* - \mathcal{W}_d^*\| + \|\mathcal{W}_d^* - \mathcal{W}^t\| \leq \epsilon. \quad (1)$$

Choose a fixed resolution d that is small enough such that

$$\|\mathcal{W}^* - \mathcal{W}_d^*\| \leq \frac{\epsilon}{2}, \quad (2)$$

then the termination criteria

$$\|\mathcal{W}^* - \mathcal{W}_d^*\| \leq \frac{C_0 d}{(1-\alpha)^2} \quad (3)$$

gives

$$d = \Omega((1-\alpha)^2\epsilon). \quad (4)$$

Initialize $\mathcal{W}^0 = 0$ and iterate over t times such that:

$$\frac{\alpha^t}{2(1-\alpha)} \|T_d(\mathcal{W}^0)\| \leq \frac{\epsilon}{2}, \quad (5)$$

Given that $\mathcal{W}^0 = 0$, hence $\|T_d(\mathcal{W}^0)\| \leq 2C$, we obtain that

$$t \leq \frac{1}{|\log \alpha|} \cdot \log \frac{2C}{(1-\alpha)\epsilon}, \quad (6)$$

the computational complexity of the fixed resolution training is

$$t = \mathcal{O}\left(\frac{1}{|\log \alpha|} \cdot \frac{1}{\log(1-\alpha)\epsilon} \cdot \left(\frac{1}{drp}\right)\right) \quad (7)$$

$$= \mathcal{O}\left(\frac{1}{|\log \alpha|} \cdot \frac{1}{\log(1-\alpha)\epsilon} \cdot \left(\frac{1}{rp(1-\alpha)^2\epsilon}\right)\right). \quad (8)$$

LEMMA 1.2. [1] *For each resolution level $[d_0, d_1, \dots, d_n]$, there exists a constant C_1 and C_2 , such that the fixed point iteration with discretization size d has an estimation error:*

$$T(\mathcal{W}) - T_d(\mathcal{W}) \leq C_1 + \alpha C_2 \|\mathcal{W}\|_d. \quad (9)$$

PROOF. The approximation error of a function with discretization is bounded if the target function is Lipschitz continuous. We have

$$T(\mathcal{W} + \Delta\mathcal{W}) = T(\mathcal{W}) + \alpha\Delta\mathcal{W}, \quad (10)$$

and

$$\|T(\mathcal{W}) - T(\Delta\mathcal{W})\| \leq \alpha \|\mathcal{W} - \Delta\mathcal{W}\|. \quad (11)$$

Here $T(\mathcal{W})$ is bounded by C_1 and $\Delta\mathcal{W}$ is bounded by C_2d . By definition of the fine-graining criterion, we know that for every resolution $d \in [d_0, d_1, \dots, d_n]$, the discretized operator T_d satisfies:

$$\|T_d(\mathcal{W}) - \mathcal{W}\| \leq \frac{C_0 d}{\alpha(1-\alpha)}. \quad (12)$$

If the estimation error is small enough, the algorithm terminates. Otherwise, we fine-grain and use \mathcal{W}^d to initialize the algorithm at the next resolution. Let the termination criterion at level d_n be

$$\frac{C_0 d_n}{(1-\alpha)^2} \leq \frac{\epsilon}{2}. \quad (13)$$

REFERENCES

- [1] Stephen G Nash. 2000. A multigrid approach to discretized optimization problems. *Optimization Methods and Software* 14, 1-2 (2000), 99–116.