Augmented Dictionary Learning for Motion Prediction

Yu Fan Chen, Miao Liu, and Jonathan P. How

***Abstract*— Developing accurate models and efficient representations of multivariate trajectories is important for understanding the behavior patterns of mobile agents. This work presents a dictionary learning algorithm for developing a part-based trajectory representation, which combines merits of the existing Markovian-based and clustering-based approaches. In particular, this work presents the augmented semi-nonnegative sparse coding (ASNSC) algorithm for solving a constrained dictionary learning problem, and shows that the proposed method would converge to a local optimum given a convexity condition. We consider a trajectory modeling application, in which the learned dictionary atoms correspond to local motion patterns. Classical semi-nonnegative sparse coding approaches would add dictionary atoms with opposite signs to reduce the representational error, which can lead to learning noisy dictionary atoms that correspond poorly to local motion patterns. ASNSC addresses this problem and learns a concise set of intuitive motion patterns. ASNSC shows significant improvement over existing trajectory modeling methods in both prediction accuracy and computational time, as revealed by extensive numerical analysis on real datasets.**

I. Introduction

With wide application of onboard sensors and GPS devices, large volumes of pedestrian and vehicular trajectory data has been collected [1], [2]. These datasets are useful for understanding the mobility patterns of human and vehicles, which in turn, benefit applications such as autonomous navigation [2] and mobility-on-demand systems [3]. To ensure safety in these applications, it is important that the autonomous vehicles can accurately anticipate the future paths of their surrounding moving objects. This requires learning the models that can differentiate complex behavior patterns.

Understanding the mobility patterns of dynamic agents is crucial for applications such as autonomous driving [4], traffic planning [5], video camera surveillance [6], and aerial interception tasks [7]. Markovian-based [8]–[10] and clustering-based [6], [7], [11], [12] methods are the two main types of approaches for trajectory modeling. The Markovian-based methods learn a state transition model from the training trajectories, and make predictions based on the current state and the hidden intent (e.g. goal) of an agent. The clustering-based methods group the training trajectories into a few clusters, and make predictions by fitting a predictive motion model, such as a Gaussian Process [13], to each cluster. Using only the current state, Markovian-based methods can be more susceptible to measurement noise. Clustering-based methods have been shown to generally produce better

Laboratory of Information and Decision Systems, Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA, USA {chenyuf2, miaoliu, jhow}@mit.edu

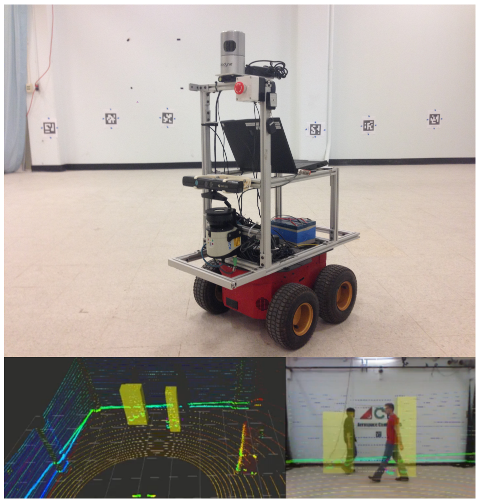
Fig. 1. Detecting and tracking pedestrians using sensors onboard of an autonomous vehicle. Top shows an autonomous vehicle equipped with a 3D Velodyne LIDAR, a Kinect camera, and a 2D SICK LIDAR. Bottom shows the vehicle detecting and tracking pedestrians (yellow bounding boxes) using a combination of camera and LIDAR data. In particular, pedestrians are detected using the histogram of gradient algorithm on an camera image, and pedestrians' position is determined by matching the camera image with the 3D point cloud data.

prediction accuracies, but they are slow to detect changes in an agent's behaviors [14]. More importantly, in real datasets collected by cameras or LIDAR (fig. 1), there often exist a large portion of short, incomplete trajectories due to occluded views. This leads to some inherent ambiguity in trajectory clustering (not clear how to assign partial trajectories) and can lead to creating similar clusters that produce repetitive predictions. Also, clustering-based methods can be computationally expensive, such as using Gibbs sampling in the Dirichlet process mixture of Gaussian processes (DPGP) framework [7], [15].

To address the shortcomings of clustering-based methods, this work develops a part-based trajectory representation with sparse dictionary learning. In particular, this work proposes the augmented semi-nonnegative sparse coding (ASNSC) formulation, which is a flexible framework that allows for specifying different constraints on different parts of the dictionary. By using a novel parametrization of multivariate trajectories in the ASNSC framework, we develop a method for learning dictionary atoms that correspond to visually intuitive local motion patterns.

Learning local motion patterns is contrasted with assigning global cluster labels to each trajectory as in clustering-based methods [7], [14]. We provide a detailed comparison between these two views in section IV-B. Noticeably, this work combines merits of Markovian-based methods and clustering-based methods by finding the *local clusters* characterized by partial trajectory segments, and making predictions using both the local motion models and the *global Markovian transition dynamics*. Multi-class inverse reinforcement learning (IRL) algorithms [16], [17], which are related to clustering-based methods (e.g. [7]), have also been applied to modeling motion patterns. Previous work based on clustering partial trajectory segments [18], [19] was limited to modeling local motion patterns as short straight line segments, whereas this work is more flexible since we do not constrain the shape of local motion patterns. Recent work [20] applied sparse coding to an image representation of hand gesture trajectories, but this work models each dimension independently, which would not be suitable for location-based applications as considered in this paper.

The main contributions of this work are (i) development of the ASNSC formulation, (ii) development of an algorithm for solving ASNSC with convergence properties under some mild conditions, (iii) formulation of the multivariate trajectory modeling problem as a dictionary learning problem, and (iv) empirical results and evaluations that show significant improvement in prediction accuracy over existing methods on real trajectory datasets.

II. Preliminaries

We first review the dictionary learning problem and some sparse coding algorithms for finding parts-based representations. Then, we develop the augmented semi-nonnegative sparse coding (ASNSC) problem formulation and motivate its usefulness with an application to trajectory modeling.

A. Sparse Coding

Sparse coding is a class of algorithms that find a succinct representation of data by learning a set of overcomplete basis vectors [21]–[23]. In this framework, each data sample $\mathbf{x_i} \in \mathbb{R}^p$ is represented as a linear combination of a few basis vectors $\mathbf{d_k} \in \mathbb{R}^p$, such that $\mathbf{x_i} \approx \sum_k s_{ki}\mathbf{d_k}$, where s_{ki} is a set of sparse coefficients. In the past decade, a number of sparse coding algorithms [22]–[24] have been developed and successfully applied to image segmentation and classification, audio signal processing, and image denoising [23]. Since many types of natural data (e.g. image pixel intensity) are non-negative, and motivated by the need to find interpretable part-based representations, subsequent research has developed non-negative sparse coding (NSC) [25] and non-negative matrix factorization (NMF) [26], [27] algorithms by imposing non-negativity constraints on the coefficient s_{ki} and dictionary atoms $\mathbf{d_k}$. However, few work has been focused on semi-nonnegative sparse coding, which allows $\mathbf{d_k}$ to be unconstrained while maintaining s_{ki} non-negative. We believe that this is not because of a lack of algorithms [28] to solve the problem, but an inherent issue with the problem formulation. In particular, the non-negativity of s_{ki} is often imposed to simplify interpretation, such that a non-zero coefficient can be interpreted as the activation of a basis feature. Yet, this problem formulation would allow dictionary atoms with opposite signs to cancel each other, thus making s_{ki} difficult to interpret semantically. In particular, it is ambiguous whether a trajectory exhibits two local motion patterns that cancel each other (fig. 3). It is plausible that an offset can be added to shift the entire dataset into the positive range, which then allows the application of NSC and NMF algorithms. However, this trick cannot be applied for some types of data, such as modeling a motion pattern as a velocity flow field. This is because the sign and magnitude of a velocity variable has different semantic meanings – heading directions and speeds – that can change after a constant shift. A detailed example will be presented in section II-C.

B. Augmented Semi-nonnegative Sparse Coding

For a set of I data samples, $\mathbf{X} = [\mathbf{x_1}, \dots, \mathbf{x}_I]$, where $\mathbf{x_i}$ is a column vector of length p, the goal is to learn a set of K dictionary atoms, $\mathbf{D} = [\mathbf{d_1}, \dots, \mathbf{d}_K]$, and the corresponding non-negative sparse coefficients $\mathbf{S} = [\mathbf{s_1}, \dots, \mathbf{s}_I]$. More precisely, the objective is to solve

$$\underset{\mathbf{D},\mathbf{S}}{\operatorname{argmin}} \quad ||\mathbf{X} - \mathbf{DS}||_F^2 + \lambda \sum_i ||\mathbf{s_i}||_1 \tag{1}$$

$$s.t. \quad \mathbf{d_k} \in \mathcal{Q}, \qquad s_{ki} \geq 0 \quad \forall k, i. \tag{2}$$

where λ is a regularization parameter, and $\mathcal{Q}$ is the feasible set in which $\mathbf{d_k}$ resides.

This problem formulation generalizes prior work in several important ways. In particular, the non-negative sparse coding (NSC) problem [25] is obtained if $\mathcal{Q} = \mathbb{R}_+^p$, and the non-negative matrix factorization (NMF) problem [26] is obtained by further setting λ to zero, and the semi-nonnegative matrix factorization (Semi-NMF) problems [28] is obtained by allowing $\mathbf{d_k}$ to be unconstrained. The key difference of this problem formulation from prior work is that eq. (2) provides the flexibility to characterize different constraints for different parts of the dictionary. This can be very useful, for example, in cases where each data sample consists of two related measurement readings from different sensors. Consider a scenario in which a data sample can be constructed by stacking up two column vectors – each corresponding to a measurement from a different sensor – such that $\mathbf{x_i}^T = [\mathbf{x_{ai}}^T, \mathbf{x_{bi}}^T]$, where $\mathbf{x_{ai}}$ and $\mathbf{x_{bi}}$ could be of different lengths. The corresponding dictionary atom can be partitioned similarly, such that $\mathbf{d_k}^T = [\mathbf{d_{ak}}^T, \mathbf{d_{bk}}^T]$, where $\mathbf{d_{ak}}$ and $\mathbf{d_{bk}}$ can be subjected to different constraints. More precisely, we can define $\mathcal{Q}$ in eq. (2) such that

$$\mathbf{d_{ak}} \in \mathcal{D}_a, \quad \mathbf{d_{bk}} \in \mathcal{D}_b, \quad g\left(\mathbf{d_{bk}}, \mathbf{d_{ak}}\right) \leq 0 \quad \forall k, \tag{3}$$

where $\mathcal{D}_a$ and $\mathcal{D}_b$ are the feasible sets in which $\mathbf{d_{ak}}$ and $\mathbf{d_{bk}}$ reside, and $g(\cdot,\cdot)$ is a joint constraint on $\mathbf{d_{ak}}$ and $\mathbf{d_{bk}}$. In real applications, $\mathcal{D}_a$ and $\mathcal{D}_b$ can be chosen to reflect the properties, such as range and discretization level, of the corresponding sensor measurements.

C. Multivariate Trajectory Modeling

The trajectory of a mobile agent i is denoted by t^i, which is a sequence of two dimensional position measurements

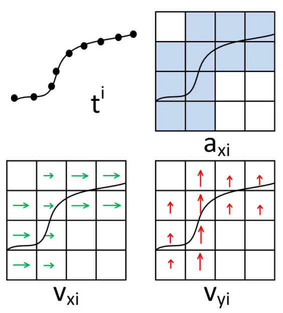

Fig. 2. Vector representation of a trajectory. In a world discretized into $M \times N$ blocks of width w, a trajectory t^i is represented by a vector $\mathbf{x_i}^T = [\mathbf{v_{xi}}^T, \mathbf{v_{yi}}^T, \mathbf{a_i}^T]$, where $\mathbf{v_{xi}}, \mathbf{v_{yi}}, \mathbf{a_i}$ are the normalized x, y velocities, and the activeness variables, respectively.

taken at a fixed time interval Δs. For predictive modeling, the agent's velocities are calculated using the finite difference approximation, that is, $(v_{xi}, v_{yi}) \approx (\frac{\Delta x_i}{\Delta s}, \frac{\Delta y_i}{\Delta s})$. Since mobile agents can move at different speeds depending on environmental factors (e.g. traffic conditions and signals) and inherent variability (e.g. agility) among them, we are more interested in finding the heading direction that determine the *shape* of a set of possible future paths. For online prediction of a mobile agent's motion, we would use this model for predicting heading direction, and use the current observation (speed, traffic conditions) for predicting *speed*. Thereby, we focus on modeling the shape of the trajectories by normalizing the magnitude of the input velocities to one, that is, require $v_x^2 + v_y^2 = 1$.

For each trajectory t^i, we find a vector representation $\mathbf{x_i}$ as a column in the data matrix $\mathbf{X}$. In particular, we discretize the world into $M \times N$ blocks of width w, and compute the x-y velocities $(v_{xi}^{mn}, v_{yi}^{mn})$ of each trajectory t^i going through each block mn. If a trajectory does not go through a grid position, the corresponding velocities are default to zero. Further, we compute a set of binary activeness variables a_i^{mn}, such that $a_i^{mn} = 1$ if trajectory t^i goes through a grid position mn, and $a_i^{mn} = 0$ otherwise. In terms of the ASNSC formulation in eq. (3), we let

$$\mathbf{x_{ai}} = \left[v_{xi}^{11}, \ldots, v_{xi}^{MN}, v_{yi}^{11}, \ldots, v_{yi}^{MN}\right]^T \in \mathbb{R}^{2MN}, \quad (4)$$

$$\mathbf{x_{bi}} = \left[a_i^{11}, \ldots, a_i^{MN}\right]^T \in \{0,1\}^{MN}. \quad (5)$$

An example of the discretized representation of a simple trajectory is illustrated in fig. 2.

We choose to model velocities because we need to distinguish trajectories that are traveling in the opposite directions; and we choose to enforce the non-negativity of sparse coefficients (s_{ki}) for interpret-ability, which would be important for the prediction step. A problem with using velocities in the dictionary learning framework is that dictionary atoms containing opposite velocities can cancel each other by addition, as illustrated in fig. 3. This is undesirable because it can lead to learning noisy dictionary atoms and thus making the sparse coefficients less interpretable. In particular, the black data sample has a coefficient 1 corresponding to the green dictionary atom, but it clearly does not exhibit the corresponding motion pattern. We address this issue

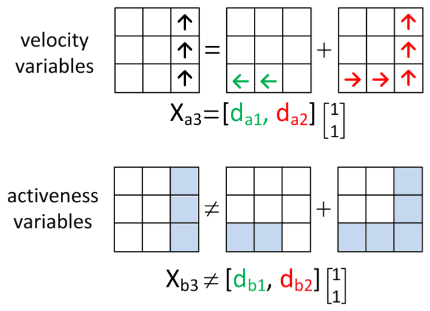

Fig. 3. Effect of augmenting the activeness variables. We represent a data sample (black) on the left as a linear combination of the two dictionary atoms (green and red) on the right. Top row shows summing dictionary atoms with opposite velocities can numerically fit the data. This is undesirable because it makes the sparse codes less interpretable. Bottom row shows that augmenting the positive activeness variables addresses this problem, because when velocities of opposite direction add to reduce error, the activeness variables would add to increase error.

by augmenting the binary activeness variables, a_i^{mn}, which specifies whether a trajectory goes through a particular grid. When relaxed to be a non-negative continuous variable, it can be interpreted as the degree of confidence that a motion primitive (dictionary atom) goes through a particular grid. Further, we require the magnitude of the velocity components to be upper bounded by the activeness variable, that is, $|v_{xi}^{mn}| \le a_i^{mn}$, $|v_{yi}^{mn}| \le a_i^{mn}$. This formulation addresses the cancellation problem because when velocities of opposite signs add to reduce error, the positive activeness variables would add to increase error, thereby discouraging combinations dictionary atoms with opposite signs. Rewrite these conditions in terms of eq. (3) of the ASNSC formulation, we specify

$$\mathbf{d_{ak}} \in \mathbb{R}^{2MN}, \quad \mathbf{d_{bk}} \in \mathbb{R}_+^{MN}, \quad (6)$$

$$|\mathbf{d_{ak}}[j]| \le \mathbf{d_{bk}}[j], \quad |\mathbf{d_{ak}}[2j]| \le \mathbf{d_{bk}}[j], \quad (7)$$

where $[j]$ denotes the jth element of a vector, and eq. (7) is the coupling constraint between the velocity variables and the corresponding activeness variables.

III. Semi-Nonnegative Sparse Coding and Dictionary Learning

In this section, we develop an algorithm for solving the augmented semi-nonnegative sparse coding problem as defined in eq. (1). Further, we establish that the proposed algorithm would converge to a local optimum if the feasible set $\mathcal{Q}$ in eq. (2) is convex. The proposed algorithm is outlined in Alg. 1, which solves the constrained optimization problem by iterating between three major steps – (i) learning the non-negative sparse coefficients given the current dictionary in line 8, (ii) updating the dictionary given the current sparse codes in line 9, and (iii) modifying the current dictionary to enforce the constraints in line 10. Further, adapting to the complexity of the data matrix, we expand the size of the dictionary (number of columns) incrementally. In particular, if a data column is poorly represented as indicated by a large residual, we add the data column into the current dictionary (lines 4-7). The addition of new dictionary atoms happens once in a few iterations to allow for the current set of

Algorithm 1: Augmented Non-negative Sparse Coding

```
1  $\mathbf{D} \leftarrow \mathbf{0}$ , $\mathbf{S} \leftarrow \mathbf{0}$
2  while not converged do
3      // add new dictionary atom if needed
4      $r_{max} \leftarrow \max_{\mathbf{x_i}} ||\mathbf{x_i} - \mathbf{D}\mathbf{s_i}||_2/||\mathbf{x_i}||_2$
5      if $r_{max} > thres$ then
6          $\mathbf{d_{new}} \leftarrow \arg\max_{\mathbf{x_i}} ||\mathbf{x_i} - \mathbf{D}\mathbf{s_i}||$
7          $\mathbf{D} \leftarrow [\mathbf{D}, \mathbf{d_{new}}]$
8      $\mathbf{S} \leftarrow$ constrQP($\mathbf{X}$, $\mathbf{D}$)
9      $\mathbf{D}' \leftarrow$ gradientDescent($\mathbf{X}$, $\mathbf{S}$)
10     $\mathbf{D} \leftarrow$ projection($\mathbf{D}'$)
11 return $\mathbf{D}$, $\mathbf{S}$
```

dictionary atoms to converge, as indicated by changes in the objective value. The following details each of the three major steps.

A. Learning the Non-negative Sparse Coefficients

We solve for the sparse coefficients $\mathbf{S}$ assuming a fixed dictionary $\mathbf{D}$. Since each column $\mathbf{s_i}$ corresponding to data sample $\mathbf{x_i}$ is independent from each other, we can optimize with respect to each $\mathbf{s_i}$ individually, that is,

$$\begin{aligned} &\underset{\mathbf{s_i}}{\operatorname{argmin}} \; ||\mathbf{x_i} - \mathbf{D}\mathbf{s_i}||_2^2 + \lambda \, ||\mathbf{s_i}||_1 \\ &s.t. \quad s_{ki} \geq 0 \quad \forall k, i. \end{aligned} \tag{8}$$

Note that the L_1-norm can be replaced with a summation term, $\sum_{ki} s_{ki}$, given the non-negativity constraints. This reduces eq. (8) to a constrained quadratic program (QP), which can be solved using many off-the-shelf optimization packages. Lee et al. [23] developed the feature-sign algorithm for finding sparse coefficients without the non-negativity constraints ($s_{ki} > 0$). In this work, we develop the non-negative feature-sign (NFS) algorithm, an adaptation of the feature-sign algorithm to accommodate for the non-negativity constraints. We derive a NFS update step and show that it strictly decreases the objective value, which allows for finding the optimum in a finite number of steps. NFS's pseudocode and detailed convergence proof are provided in the supplementary materials (`https://goo.gl/dSMCom`). This adaptation was compared with Matlab's *quadprog* solver [29] using various optimization setting, and the results showed similar accuracy and better run time in solving eq. (8). The non-negative feature-sign algorithm was used to finding the sparse coefficient matrix $\mathbf{S}$.

B. Updating the Dictionary

The dictionary $\mathbf{D}'$ can be solved for assuming fixed sparse coefficients $\mathbf{S}$. Since the dictionary is shared across all data samples, all dictionary atoms must be solved for all data samples jointly by minimizing the following,

$$f(\mathbf{D}') = ||\mathbf{X} - \mathbf{D}'\mathbf{S}||_F^2 + \eta \, ||\mathbf{D}'||_F^2 , \tag{9}$$

where $\eta \, ||\mathbf{D}'||_F$ is a regularization term.

We derive a gradient descent update step for solving eq. (9). In particular, let $\mathbf{D}_{old}$ be the previous iterate, we update the dictionary such that,

$$\mathbf{D}' = \mathbf{D}_{old} - \alpha \nabla_{\mathbf{D}'} f, \tag{10}$$

where $\nabla_{\mathbf{D}'} f = -2(\mathbf{X} - \mathbf{D}'\mathbf{S})\mathbf{S}^T + 2\mathbf{D}'$, and α is a positive scalar step size. In essence, the algorithm takes a step in the steepest descent direction.

C. Enforcing Constraints on the Dictionary Atoms

The previous step learns a dictionary $\mathbf{D}$ by solving eq. (1) without considering the constraints in eq. (2). Here, we enforce the constraints by projecting each column of $\mathbf{D}'$ back onto the feasible space. Let $\mathbf{d_k}'$ be the kth column of $\mathbf{D}'$, we update each dictionary column such that,

$$\begin{aligned} \mathbf{d_k} = \; &\underset{\mathbf{d_k}}{\operatorname{argmin}} \; ||\mathbf{d_k} - \mathbf{d}'_\mathbf{k}||_2 \\ &s.t. \quad \mathbf{d_k} \in \mathcal{Q}. \end{aligned} \tag{11}$$

For the trajectory modeling application, eq. (7) specifies that only the variables at same grid position, $(a_k^{mn}, v_{xk}^{mn}, v_{yk}^{mn})$, are jointly constrained; whereas variables at different grid positions are independent from each other (e.g. no joint constraints on a_k^{mn}, a_k^{hl} for $mn \neq hl$). This allows for a simple projection operation by solving for the variables at each grid position independently.

D. Convergence Analysis

Let $C(\mathbf{D}, \mathbf{S})$ denote the objective function $||\mathbf{X} - \mathbf{D}\mathbf{S}||_F^2 + \lambda \sum_i ||\mathbf{s_i}||_1 + \eta \, ||\mathbf{D}||_F^2$. Here, we prove that Alg. 1 will converge to a local minimum of the objective function.

Fact 1. The objective function eq. (1) is biconvex in $\mathbf{S}$ and $\mathbf{D}$ [23].

Fact 2. The non-negative feature-sign algorithm finds the global optimum of eq. (8) in a finite number of steps, and each feature-sign step strictly reduces the objective value (see the supplementary materials `https://goo.gl/dSMCom`).

Lemma 1: Given that $\mathcal{Q}$ is convex, for any $(\mathbf{D}_{old} \in \mathcal{Q}, \mathbf{S})$, the gradient-projection step (lines 9-10 in Alg. 1) reduces the objective value, that is, $C(\mathbf{D}, \mathbf{S}) \leq C(\mathbf{D}_{old}, \mathbf{S})$, where $\mathbf{D}$ is obtained by solving eq. (9) and eq. (11). Further, if $C(\mathbf{D}, \mathbf{S}) = C(\mathbf{D}_{old}, \mathbf{S})$, then $\mathbf{D} = \mathbf{D}_{old}$.

Proof: Let $\mathbf{D}'$ be the solution to eq. (9) obtained via solving eq. (10). Since $||\mathbf{D}' - \mathbf{D}||_F^2 = \sum_k ||\mathbf{d}'_\mathbf{k} - \mathbf{d_k}||_2^2$, solving for eq. (11) is equivalent to finding the l_2 projection of $\mathbf{D}$ onto the feasible space. Further, since $\mathcal{Q}$ is convex, lines 9-10 of algorithm 1 is a gradient-projection step that is guaranteed to reduce the objective value [30]. Further, the gradient-projection update step returns the same $\mathbf{D} = \mathbf{D}_{old}$ if and only if $\mathbf{D}_{old}$ is a stationary point [30]. ■

Theorem 1: Alg. 1 converges to a local minimum of the optimization problem eq. (1) if $\mathcal{Q}$ is convex.

Proof: Given a data matrix with a finite number of columns, the algorithm would stop introducing new dictionary columns after a finite number of iterations (lines 4-7), allowing us to only consider lines 8-10 for showing convergence. We establish convergence using the monotonic convergence theorem by showing that the feature-sign step (line 8) and the gradient-projection step (lines 9-10) both decrease the objective function $C(\mathbf{D}, \mathbf{S})$. Let $(\mathbf{D}_{old}, \mathbf{S}_{old})$ be values from the previous iteration. The feature-sign algorithm computes $\mathbf{S}$ given $(\mathbf{S}_{old}, \mathbf{D}_{old})$. Fact 2 establishes $C(\mathbf{D}_{old}, \mathbf{S}) \leq C(\mathbf{D}_{old}, \mathbf{S}_{old})$, and Lemma 1 establishes $C(\mathbf{D}, \mathbf{S}) \leq C(\mathbf{D}_{old}, \mathbf{S})$. If either equality holds, then the algorithm has converged; otherwise, the objective value strictly decreases, that is, $C(\mathbf{D}_{old}, \mathbf{S}_{old}) < C(\mathbf{D}, \mathbf{S})$. Further, when the algorithm has converged at $(\mathbf{D}, \mathbf{S})$, fact 1 and lemma 1 imply that this solution is a stationary point. ■

We claim that the feasible space $\mathcal{Q}$ as defined in eq. (7) is convex. In particular, it is straightforward to verify algebraically that for any $\mathbf{d_1}, \mathbf{d_2} \in \mathcal{Q}$, a convex combination of $\mathbf{d_1}, \mathbf{d_2}$ also belongs to $\mathcal{Q}$. The algebraic details of the proof are omitted for brevity. Thus, we can establish the following corollary by invoking Theorem 1.

Corollary 2: Alg. 1 converges to a local minimum of the trajectory modeling problem characterized by eq. (7).

E. Computational Complexity

For each of the three major steps of ASNSC (line 8-10) in algorithm 1, the per iteration time complexity is provided below. Assume there are n_t samples, each of which has length p, and the number of dictionary elements to be k_d. For the problem considered in our paper, $p > n_t \gg k_d$. [1].

In the nonnegative feature-sign (line 8 in algorithm 1) update, the major computation loads are matrix operations. For a dictionary matrix $\mathbf{D} \in \mathbb{R}^{p \times n_t}$, it requires calculating $\mathbf{D}^T\mathbf{D}$, $(\mathbf{D}^T\mathbf{D})^{-1}$ and $\mathbf{D}^T\mathbf{X}$, which has complexity $O(k_d^2 p)$, $O(k_d^3)$ and $O(pk_d n_t)$, respectively. Overall, the time complexity for updating sparse codes is $O(pk_d n_t)$.

The gradient descent update (line 9 in algorithm 1) requires computing the gradient information, in which the major computation loads are for calculating $\mathbf{XS^T}$ and $\mathbf{DS}$, which has complexity $O(pk_d n_t)$ and $O(pk_d n_t)$, respectively. The projection step requires solving for a l_2 minimization problem for each dictionary column and depends on the complexity of the convex set. For the trajectory modeling problem, the complexity of the projection step is $O(pk_d)$. Overall, the time complexity for updating the dictionary is $O(pk_d n_t)$.

Combining the three steps, ASNSC has computational complexity $O(pk_d n_t)$.

F. A Faster Dictionary Update Step

While the proposed dictionary learning step has good theoretical guarantee, the gradient-projection update can be a bit slow because it takes a sequence of small steps to find the local optimum. Since the algorithm alternates between solving for $\mathbf{D}$ and $\mathbf{S}$, we find empirically that it is often better to find a good dictionary quickly and move on to the next iteration, compared with finding the locally optimal dictionary for the current iterate of $\mathbf{S}$. We describe a Lagrange-dual-projection step that yields better computational speed at the expense of theoretical rigor.

Recall the data matrix $\mathbf{X}$ consists of H rows and I columns, which correspond to the dimension and number of data samples, respectively. Due to domain discretization as described in section II-C, our parametrization of a trajectory typically has more features than samples ($H = 3MN \gg I$). This observation motivates using of the Lagrange dual approach [23] to solving eq. (9), because the number of dual variables is much less than the number of primal variables. After solving for the diagonal dual variable matrix $\mathbf{\Lambda}$ as described in [23], we can reconstruct the dictionary by finding

$$\mathbf{D}'^T = \left(\mathbf{S}\mathbf{S}^T + \mathbf{\Lambda} + \eta \mathbf{I}\right)^{-1} \left(\mathbf{S}\mathbf{X}^T\right). \qquad (12)$$

Compared with [23], our formulation in eq. (9) contains an extra regularization term, $\eta \, ||\mathbf{D}'||_F$, whose effect can be understood by inspection of eq. (12). In particular, since $H \gg I$, the matrix $\mathbf{S}\mathbf{S}^T + \mathbf{\Lambda}$ can sometimes be singular. Compared with a gradient update, a Lagrange-dual update takes a large step and finds the unconstrained global optimum $\mathbf{D}^*$ of the objective function (eq. (9)). Then, we perform a line search from the previous dictionary $\mathbf{D}_{old}$ to the global optimum $\mathbf{D}^*$ to ensure that the update step would decrease the objective value [2]. In practice, we interweave the two update steps to achieve faster convergence and still attain theoretical convergence properties. Empirically, we found this update rule can reduce time to convergence by approximately three times.

IV. Results

We evaluated the proposed algorithm on two trajectory datasets of different length scales. The first dataset consists of 147 taxi trajectories (adapted from [1]), each of which was converted to a vector representation of length 2484. Using dataset I, we discuss the importance of augmenting the activeness variables for dictionary learning. The second dataset consists of 72 pedestrian trajectories (collected using a Velodyne LIDAR), each of which are converted to a vector representation of length 375. Using dataset II, we describe how to make predictions using the learned dictionary. We evaluate the algorithm on a computer with an Intel i7-4510U CPU and 16GB of memory.

A. Learning Traffic Patterns

We ran the proposed algorithm on the taxi dataset to examine the importance of using the ASNSC framework by augmenting the activeness variables for dictionary learning.

[1]Recall $p = M \times N$ for a map discretized into $M \times N$ blocks. A preprocessing step is run to omit the grid positions at which no trajectory passes through (e.g. occupied by static obstacles). Given this simple preprocessing step, we note $p \ll M \times N$.

[2]by convexity of the objective function.

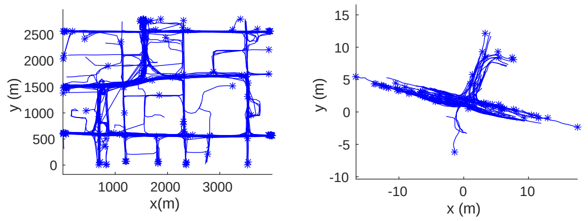

Fig. 4. Two real world trajectory datasets: 147 taxi trajectories (dataset I, left) and 72 pedestrian trajectories (dataset II, right). In dataset I, we processed (ex. parsing and filtering) a subset of the taxi GPS data published by Yuan et al. [1]. In dataset II, we used a Velodyne Lidar to detect and track pedestrians at an intersection.

The algorithm converged in 168 iterations within 33.1 seconds, and learned a 26 column dictionary and the associated sparse coefficients. Recall that the number of dictionary atoms are learned automatically by the algorithm. For each point p^i_j in a trajectory t^i, we find a projection variable $r^i_j \in \{1, \ldots, K\}$, which correspond to the dictionary atom that mostly likely generated this point. More precisely,

$$r^i_j = \underset{k}{\operatorname{argmin}} \left|\left|p^i_j - Grid(\mathbf{d_k}, p^i_j)\, s_{ki}\right|\right|_2 \tag{13}$$

where s_{ki} is (k, i)th element in the sparse coefficient matrix $\mathbf{S}$, and $Grid(\mathbf{d_k}, p^i_j)$ is the x-y velocities of the dictionary atom $\mathbf{d_k}$ in the grid position containing p^i_j. The projection variable r^i_j can be interpreted as an assignment variable of the point p^i_j to a dictionary atom $\mathbf{d_k}$. Thus, the projection vector $\mathbf{r^i}$ specifies the segmentation pattern of a trajectory t^i.

For brevity, we only illustrate the first six dictionary atoms learned from dataset I in fig. 5. Visualization of the dictionary atoms and the projection onto these dictionary atoms are shown in the left and middle subfigures, respectively. We also show the projection vector, $\mathbf{r^i}$, of a single trajectory in the right subfigure. The top and bottom rows correspond to solving eq. (1) with ASNSC, and with classical semi-nonnegative sparse coding without augmenting the activeness variables. By augmenting the activeness variables, we address the problem of having dictionary atoms with opposite signs cancel each other, and were able to learn a set of less noisy and more localized dictionary atoms. Also, we were able to better identify commonly shared segments and obtain an intuitive segmentation pattern (right subfigure of fig. 5).

B. Making Predictions at an Intersection

For the pedestrian dataset, the ASNSC algorithm converged in 59 iterations within 4.98 seconds, and learned a 9 column dictionary and the associated sparse code. The learned dictionary correspond well to human intuition – each dictionary atom specifies a motion pattern that either enters or exits the intersection. Visualization of all nine dictionary atoms and the corresponding projection are shown in fig. 6.

Besides being useful for segmentation, the local motion patterns learned by ASNSC are also useful for making accurate predictions. After learning a dictionary with K columns, we built a transition matrix $\mathbf{T} \in \mathbb{Z}^{K \times K}$ to capture the temporal ordering of the dictionary atoms using the projection vectors $\mathbf{r^i}$. In particular, the (k, l)th element of $\mathbf{T}$ corresponds to the number of trajectories that exhibited transition from kth dictionary atom to the lth dictionary atom.

We evaluate the prediction quality on a separate test set of 24 trajectories by comparing with a set of hand-labeled predictions, as illustrated on the left of fig. 7. In particular, given the observed path of pedestrian going into an intersection (shown in black), the algorithm finds a set of possible future paths. We categorize each predicted path as either correct (red), incorrect (green), missing or repetitive (orange).

Recall in clustering-based trajectory modeling approaches, given an observed path, predictions are made by using (forward propagation) the n clusters that agree the most with the observation. The problem of this approach is that in real datasets, there often exist many fragmented trajectories due to occlusion in the data collection process. The fragmented trajectories are often classified into their own clusters. Consequently, making predictions using these cluster can lead to poor results, as illustrated on the right of fig. 7. In contrast, this work models each trajectory as a concatenation of a few dictionary atoms. Given an observed path, we first find the dictionary atom k that most likely generated this observation. Then, we can make predictions by finding the set of possible subsequent dictionary atoms $\{\mathbf{d_j} | T_{kj} > 0\}$.

TABLE I

PREDICTION PERFORMANCE OF ASNSC VS DPGP.

	DPGP (ξ, n)				ASNSC
	$-1.5, 3$	$-1.3, 3$	$-1.3, 3$	$-1.5, 5$	-
missed (%)	27.3	36.4	31.8	18.2	4.5
correct (%)	72.7	63.6	68.2	81.8	95.5
repetitive (%)	38.6	31.8	38.6	31.8	0
incorrect (%)	9.1	4.5	13.6	50.0	4.5

We compare our algorithm against using Gibbs sampling for the Dirichlet Process Gaussian Process (DPGP) model [14], which clusters training trajectories with a DP prior and models each cluster using a GP. In contrast, we model each transition (concatenation of two dictionary atoms $\{\mathbf{d_k}, \mathbf{d_j} | T_{kj} > 0\}$) as a GP. The prediction results on a few representative trajectories from the test set are shown in fig. 8. The proposed algorithm produced correct predictions in nearly all test cases, whereas DPGP made some repetitive and erroneous predictions. To find the set of possible paths using DPGP, we identify the set of clusters that agree with observation (above a likelihood value ξ) and pick the best n clusters. We tuned the parameters ξ and n through a grid search procedure. In comparison, the proposed method does not require parameter tuning for prediction because the transition matrix $\mathbf{T}$ specifies the number of possible future behaviors. Table I compares the performance of DPGP at 4 different parameter settings with the proposed algorithm.

Recall from section III-E that the computational complexity of ASNSC is $O(pk_dn_t)$. The computation complexity of Gibbs sampler for DP-GP model has been discussed in [15], in which each iteration takes $O((pn_t)^3/M)$, with M being

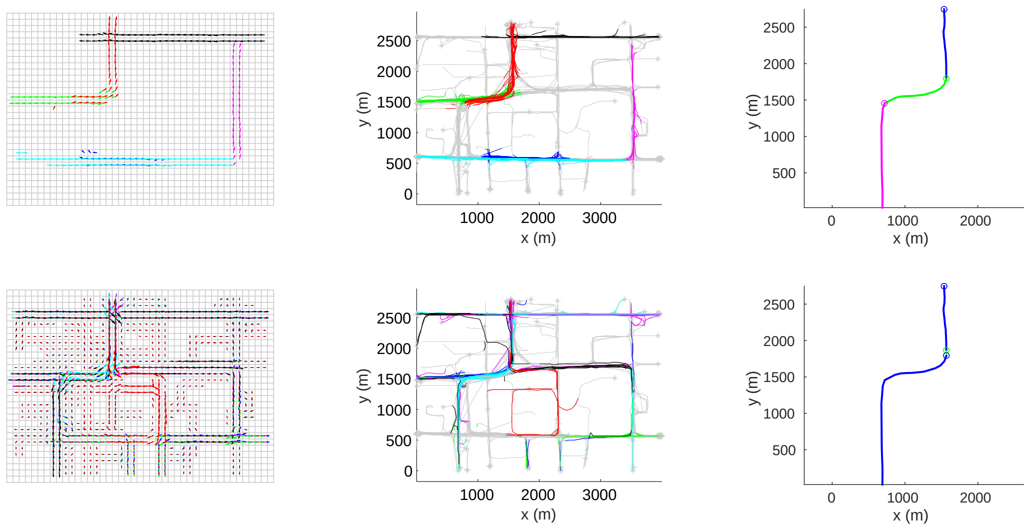

Fig. 5. Effect of augmenting the activeness variables (AVs) in the ASNSC framework (Top: with augmentation of the AVs; Bottom: without augmentation of the AVs.). Visualization of the dictionary atoms (the first six in different color), and projection of the training trajectories onto the dictionary atoms, are shown on the left and middle subfigures, respectively. The right subfigure shows the segmentation (projection) of one training trajectory, with each color representing a different dictionary atom (color is uncorrelated to the left and middle subfigures). The gray color in the middle subfigures shows all trajectories in the dataset.

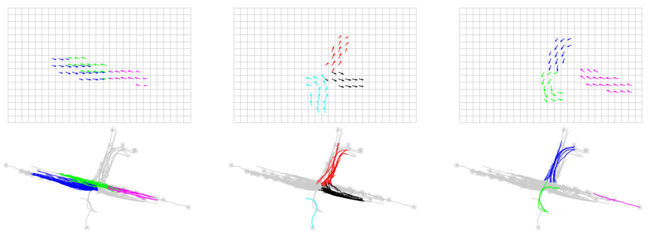
Fig. 6. Dictionary atoms found by solving ASNSC on dataset II. Top: visualization of dictionary atoms. Bottom: projection of training trajectories onto the dictionary atoms. We found 9 dictionary atoms (top row), which are shown in groups of three. The bottom row shows the projection of training trajectories onto the corresponding dictionary atoms.

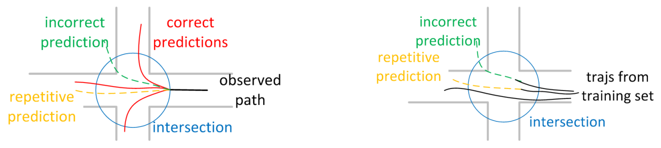

Fig. 7. Metric for evaluating prediction quality. Left: as a pedestrian enters an intersection, an algorithm makes predictions about the set of possible future paths the pedestrian may take. The predictions are evaluated by a hand-labeled set as ground truth. Right: poor predictions due to trajectory segments. Due to sensor occlusion, real datasets often contain fragmented trajectories, as shown in black. Predictions based on motion patterns (clusters) formed by fragmented trajectories can be repetitive (orange) or incorrect (green).

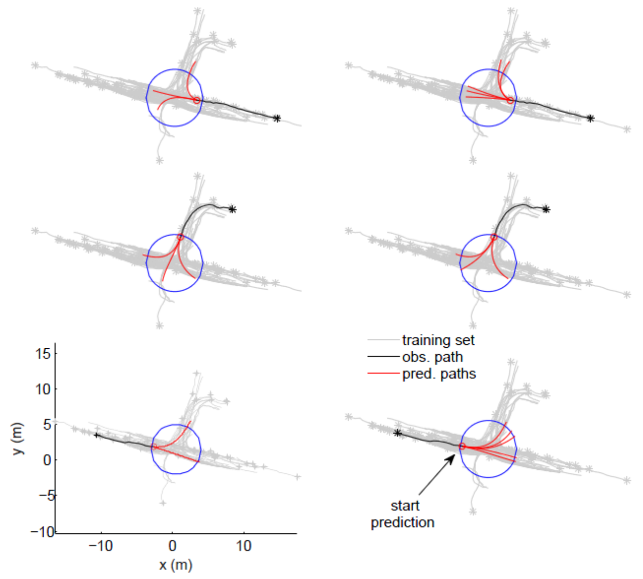

Fig. 8. Examples of prediction results from ANSSC (left) and DPGP (right). We've trained the algorithms on dataset II, and made predictions for a separate test set of 24 trajectories. For brevity, three trajectories are shown here. Given the observation history of a pedestrian (black line), we want to find the set of possible future paths. Subfigures on the right show that DPGP made some repetitive and incorrect predictions. In contrast, subfigures on the bottom show that ASNSC correctly predicted the pedestrian's future paths in all three cases (consistent with the gray training set).

the average number of mixture components $(M < n_t)$. Thus, the comparison shows that the per iteration complexity of the proposed method is an order of magnitude lower than using Gibbs sampling for the DP-GP model, which corroborates with the empirical result. In particular, on dataset II, learning the DPGP model via Gibbs sampling (200 steps) took 377.3 seconds to complete, while the proposed algorithm took 4.98 seconds to learn a set of dictionary atoms.

C. Sensitivity Analysis

ASNSC does not assume a fixed size dictionary. Instead, the proposed algorithm incrementally expands the size of the dictionary as dictated by the parameter $thres \in (0, 1)$ in line 4 of algorithm 1. When the relative representational error of a data sample exceeds a certain threshold, then the data sample is added to the dictionary (with some random noise). Thus, the $thres$ parameter lends a very natural interpretation – the

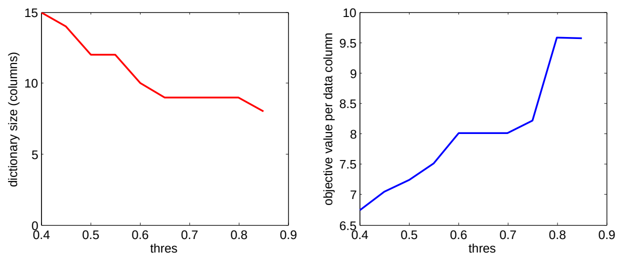

Fig. 9. Effect of varying the $thres$ parameter in ANSC in dataset II. Left shows the number of dictionary atoms learned from data as a function of the $thres$ parameter. Right shows the objective value per data sample as a function of the $thres$ parameter.

largest accepted representational error. Consequently, with a larger value of the $thres$ parameter, we expect learning a dictionary with fewer columns and higher representational error. This expectation is corroborated with empirical evaluation on both trajectory datasets; for brevity, only result from dataset II is shown in fig. 9. Further, the segmentation result does not change drastically in the neighborhood of the selected $thres$ value. For instance, fig. 9 shows that ASNSC finds 9 dictionary atoms on dataset II for $thres \in [0.65, 0.8]$.

V. Conclusion

In this paper, we presented a novel semi-nonnegative sparse coding framework (ASNSC) that allows for specifying coupling constraints on different parts of the dictionary. We developed an algorithm for solving the ASNSC problem, and showed that the algorithm would converge to a local optimum under a convexity condition. Further, a trajectory modeling problem is formulated as a dictionary learning problem in the ASNSC framework. Solution to this problem using the proposed algorithm showed significant improvement in both prediction accuracy and computational time over a state-of-art clustering-based method, DPGP, on real datasets. For future work, we will integrate the proposed predictive model with a risk-aware motion planner for improving the safety of autonomous vehicles.

References

[1] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 316–324.

[2] S. Choi, E. Kim, and S. Oh, "Real-time navigation in crowded dynamic environments using Gaussian process motion control," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 3221–3226.

[3] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 454–460.

[4] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[5] C. Chen, H. Su, Q. Huang, L. Zhang, and L. Guibas, "Pathlet learning for compressing and planning trajectories," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 392–395.

[6] Z. Zhang, K. Huang, and T. Tan, "Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3. IEEE, 2006, pp. 1135–1138.

[7] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A Bayesian nonparametric approach to modeling motion patterns," *Autonomous Robots*, vol. 31, no. 4, pp. 383–400, 2011.

[8] D. Makris and T. Ellis, "Spatial and probabilistic modelling of pedestrian behaviour." in *BMVC*. Citeseer, 2002, pp. 1–10.

[9] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden Markov models," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 3, pp. 403–416, 2009.

[10] A. Bruce and G. Gordon, "Better motion prediction for people-tracking," in *Proc. of the Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain*, 2004.

[11] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1164–1171.

[12] Y. F. Chen, M. Liu, S.-Y. Liu, J. Miller, and J. P. How, "Predictive modeling of pedestrian motion patterns with bayesian nonparametrics," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1861.

[13] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with Gaussian processes," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1229–1234.

[14] S. Ferguson, B. Luders, R. C. Grande, and J. P. How, "Real-time predictive modeling and robust avoidance of pedestrians with uncertain, changing intentions," in *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, August 2014.

[15] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of Gaussian process experts," *Advances in neural information processing systems*, vol. 2, pp. 881–888, 2002.

[16] J. Almingol, L. Montesano, and M. Lopes, "Learning multiple behaviors from unlabeled demonstrations in a latent controller space," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 136–144.

[17] J. Choi and K.-E. Kim, "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions," in *Advances in Neural Information Processing Systems*, 2012, pp. 305–313.

[18] C. Sung, D. Feldman, and D. Rus, "Trajectory clustering for motion prediction," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1547–1552.

[19] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: a partition-and-group framework," in *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. ACM, 2007, pp. 593–604.

[20] C. Vollmer, S. Hellbach, J. Eggert, and H.-M. Gross, "Sparse coding of human motion trajectories with non-negative matrix factorization," *Neurocomputing*, vol. 124, pp. 22–32, 2014.

[21] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[22] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4311–4322, 2006.

[23] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Advances in neural information processing systems*, 2006, pp. 801–808.

[24] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[25] P. O. Hoyer, "Non-negative sparse coding," in *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*. IEEE, 2002, pp. 557–565.

[26] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in neural information processing systems*, 2001, pp. 556–562.

[27] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 25, no. 6, pp. 1336–1353, 2013.

[28] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 1, pp. 45–55, 2010.

[29] "Quadratic programming," http://www.mathworks.com/help/optim/ug/quadprog.html.

[30] D. P. Bertsekas, "Nonlinear programming," 1999.