Linearization of Temporal Trajectories via Prediction

Anonymous Author(s)
Affiliation
Address
email

Abstract

Abstract abstract

1 Introduction

2 Prior Work

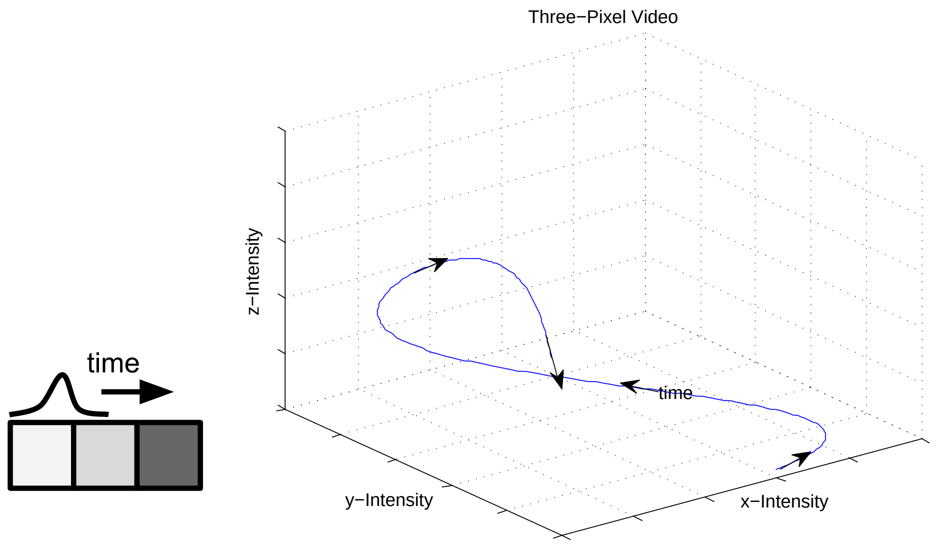
This work was heavily inspired by the philosophy revived by Hinton et al. in the transforming auto-encoder, which introduced "capsule" units. In that work, a linearly covariant representation is learned by the capsules by providing the network with the true latent states as implicit targets. Our work allows us to move to a more unsupervised setting in which the true latent states are not only unknown, but represent completely arbitrary qualities. This was made possible with two assumptions: (1) that temporally adjacent samples also correspond to neighbors in the latent space, (2) predictions of future samples can be formulated as *linear* operations in the latent space. In theory, the representation learned by our method is very similar to the representation learned by the "capsules"; in that the representation has a locally stable –*"what"* component and locally linear, or covariant – *"where"* component.

3 Offset Pooling

Assume that the dataset is parameterized by a temporal index t so the dataset is described by the sequence $X = \{..., x^{t-1}, x^{t}, x^{t+1}, ...\}$ with a corresponding feature map sequence $Z = \{..., z^{t-1}, z^{t}, z^{t+1}, ...\}$. Each feature map z^t can be partitioned into k potentially overlapping neighborhoods. We define a soft version of the max operator within each pool group. Although we can define an arbitrary topology in feature space, for now assume that we have the familiar three-dimensional spatial feature map representation where each activation is a function $z(f,x,y)$, where x and y correspond to the spatial location, and f is the feature map index. Assuming that the feature activations are positive, we define our soft "$max - pooling$" operator for the k^{th} neighborhood N_k as:

$$m_k = \int_{N_k} z(f,x,y) \frac{e^{\beta z(f,x,y)}}{\int_{N_k} e^{\beta z(f\prime,x\prime,y\prime)} \mathrm{d}v\prime} \, \mathrm{d}v \approx \max_{N_k} z(f,x,y) \tag{1}$$

Where $\beta > 1$ and $\mathrm{d}v = \mathrm{d}f\mathrm{d}x\mathrm{d}y$. Assuming that the activation pattern within each neighborhood is approximately unimodal, we can define a soft versions of the $argmax$ operator. The vector $\mathbf{p}_k$ approximates the location in the feature topology at which the max activation value occurred. Assuming that pooling is done volumetrically, that is, spatially *and* across features, $\mathbf{p}_k$ will have



(a) (b)

Figure 1: (a) A video generated by translating a Gaussian intensity bump over a three pixel array (x,y,z), (b) the corresponding manifold parametrized by time in three dimensional space

three components. In general, the number of components in $\mathbf{p}_k$ is equal to the dimension of the topology of our feature space induced by the pooling neighborhood. The dimensionality of $\mathbf{p}_k$ also has the interpretation as being the *maximal intrinsic dimension of the data*. If we define a local standard coordinate system in each pooling volume which is bounded between -1 and +1, the soft "$argmax - pooling$" operator is defined by the vector-valued integral:

$$\mathbf{p}_k = \int_{-1}^{1} \begin{bmatrix} f \\ x \\ y \end{bmatrix} \frac{e^{\beta z(f,x,y)}}{\int_{N_k} e^{\beta z(f\prime,x\prime,y\prime)} \mathrm{d}v\prime} \mathrm{d}v \approx \arg\max_{N_k} z(f,x,y) \tag{2}$$

To motivate the definition of these operations consider a video generated by translating a Gaussian "intensity bump" over a three pixel array at constant speed. The video corresponds to a one dimensional manifold in three dimensional space, i.e. a curve parameterized by time. Next, assume that some feature detector activates when centered on such a bump. Applying the max-pooling operator over this region reveals that the Gaussian bump is present in somewhere in this region (*i.e. the what*). Applying the $argmax$ pooling operator over the region returns the position (*i.e. the where*) with respect to a local coordinate frame defined over the pooling region. This position variable varies linearly with respect to time, and locally parameterizes the data manifold. More generally offset-pooling can potentially locally parametrize arbitrary transformations if the features we pool over are learned. Moreover, linear predictions can be formulated in the parameter space provided that there is a mapping from the parameter space back to the input space (decoder).