# A FAST, UNIVERSAL ALGORITHM TO LEARN PARAMETRIC NONLINEAR EMBEDDINGS

**Miguel Á. Carreira-Perpiñán** and **Max Vladymyrov**, EECS, UC Merced

BayLearn
Bay Area Machine Learning Symposium

## 1 Abstract

Nonlinear embedding algorithms such as stochastic neighbor embedding do dimensionality reduction by optimizing an objective function involving similarities between pairs of input patterns. The result is a low-dimensional projection of each input pattern. A common way to define an out-of-sample mapping is to optimize the objective directly over a parametric mapping of the inputs, such as a neural net. This can be done using the chain rule and a nonlinear optimizer, but is very slow, because the objective involves a quadratic number of terms each dependent on the entire mapping's parameters. Using the method of auxiliary coordinates, we derive a training algorithm that works by alternating steps that train an auxiliary embedding with steps that train the mapping. This has two advantages: the algorithm is universal in that a specific learning algorithm for any choice of embedding and mapping can be constructed by simply reusing existing algorithms for the em-bedding and for the mapping; and the algorithm is fast because reusing N-body methods developed for nonlinear embeddings yields linear-time iterations.

Funded by NSF award IIS–1423515.

## 2 Parametric embeddings

We focus on problems of nonlinear embedding methods, such as Stochastic Neighbor Embedding (SNE), $t$-SNE, Elastic Embedding (EE). The goal of the original methods is to obtain low-dimensional coordinates $\mathbf{X}_{d\times N}$ for a given set of high-dimensional points $\mathbf{Y}_{D\times N}$. For example, in EE:

$$E(\mathbf{X}) = \sum_{n,m=1}^{N} w_{nm} \|\mathbf{x}_n - \mathbf{x}_m\|^2 + \lambda \sum_{n,m=1}^{N} \exp\left(-\|\mathbf{x}_n - \mathbf{x}_m\|^2\right) \quad \lambda > 0.$$

- Often produce high-quality embedding results.
- Require elaborate iterative non-convex optimization, which can be mitigated with (1) the spectral direction, which uses part of the Hessian efficiently, and (2) an $N$-body approximation for the gradient so each each iteration runs in linear time.
- Do not give an out-of-sample mapping for projection of new data.

We can obtain an out-of-sample mapping $\mathbf{F}$ for test data in 3 different ways:

- Variational approach: optimize together for training and test data, but keeping training data fixed. No closed form solution, costly optimization.
- Direct fit: fit $\mathbf{F}$ directly to $(\mathbf{Y}, \mathbf{X})$. The mapping plays no role in the learning of the embedding $\mathbf{X}$.
- Parametric embedding (PE): train $\mathbf{F}$ using the embedding objective function (thus converting the nonparametric embedding into a parametric one):

$$P(\mathbf{F}) = \sum_{n,m=1}^{N} w_{nm} \|\mathbf{F}(\mathbf{y}_n) - \mathbf{F}(\mathbf{y}_m)\|^2 + \lambda \sum_{n,m=1}^{N} \exp\left(-\|\mathbf{F}(\mathbf{y}_n) - \mathbf{F}(\mathbf{y}_m)\|^2\right) \quad \lambda > 0.$$

This ties the mapping to the embedding during the optimization. However, (1) the gradient of $P$ wrt $\mathbf{F}$ must be derived using the chain rule and depends on the form of both $P$ and $\mathbf{F}$, (2) computing the gradient is $\mathcal{O}(N^2)$.

## 3 Method of auxiliary coordinates (MAC)

Convert the nested problem for $P(\mathbf{F})$ into an equivalent constrained problem:

$$\min \bar{P}(\mathbf{F}, \mathbf{Z}) = E(\mathbf{Z}) \quad \text{s.t.} \quad \mathbf{z}_n = \mathbf{F}(\mathbf{y}_n),\ n = 1, \dots, N$$

that is not nested, where $\mathbf{z}_n$ are the auxiliary coordinates (low-dim projection) for an input pattern $\mathbf{y}_n$. Solve it using the quadratic penalty method:

$$\min P_Q(\mathbf{F}, \mathbf{Z}; \mu) = E(\mathbf{Z}) + \frac{\mu}{2} \sum_{n=1}^{N} \|\mathbf{z}_n - \mathbf{F}(\mathbf{y}_n)\|^2 = E(\mathbf{Z}) + \frac{\mu}{2} \|\mathbf{Z} - \mathbf{F}(\mathbf{Y})\|^2, \quad \mu \to \infty.$$

The minimization alternates between two well-studied problems:

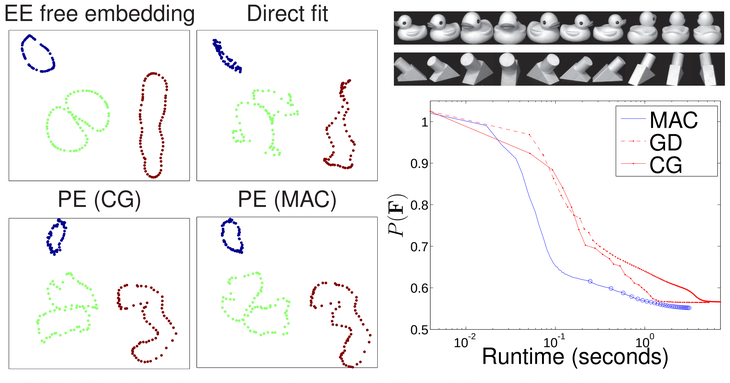
- Over $\mathbf{F}$ given $\mathbf{Z}$: $\min_{\mathbf{F}} \sum_{n=1}^{N} \|\mathbf{z}_n - \mathbf{F}(\mathbf{y}_n)\|^2$. This is a standard least-squares regression for a dataset $(\mathbf{Y}, \mathbf{Z})$ using $\mathbf{F}$, and can be solved using existing, well-developed code for many classes of mappings.
- Over $\mathbf{Z}$ given $\mathbf{F}$: $\min_{\mathbf{Z}} E(\mathbf{Z}) + \frac{\mu}{2} \|\mathbf{Z} - \mathbf{F}(\mathbf{Y})\|^2$. This is a regularized embedding which can be minimized using existing techniques for $E(\mathbf{Z})$ (such as the spectral direction) with simple modifications.

Benefits:

- Easy to develop an algorithm for an arbitrary choice of embedding objective function $E$ and of mapping $\mathbf{F}$: simply reuse existing algorithms for them.
- Deals with the optimization of $E$ and of $\mathbf{F}$ separately. The optimization details (step sizes, etc.) of the nested problem decouple and remain confined within the corresponding steps.
- Allows for non-differentiable mappings (e.g. decision trees).
- Same complexity as using the chain rule. However, the quadratic bottleneck step over $\mathbf{Z}$ can be easily linearized with $N$-body methods.
- Convergence to a minimum guaranteed as $\mu \to \infty$.

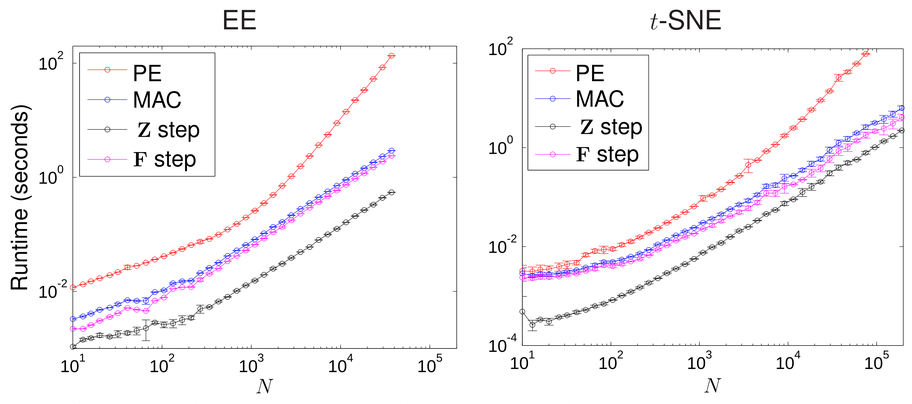
## 4 Experiments

### 1. Illustrative example.




- COIL-20 dataset: $128 \times 128$ images of the rotation of $3$ objects every $5°$.
- We used EE to produce the free embedding $E(\mathbf{X})$ (i.e., $\mu = 0$).
- Direct fit applies a linear mapping directly to a free embedding.
- Parametric embedding (PE) directly optimizes $P(\mathbf{F})$ using Gradient Descent (GD), Conjugate Gradients (CG), or MAC.
- MAC finds better local minima and is faster.

### 2. Cost of the iterations.




- MAC shown by itself and as a split between $\mathbf{Z}$ and $\mathbf{F}$ steps.
- Mapping $\mathbf{F}$: neural net with architecture $3$–$100$–$500$–$2$ with sigmoidal activations.
- $\mathbf{Z}$ step: approximated w/ Barnes-Hut method for $t$-SNE and fast multipole method for EE.
- PE with chain rule is $\mathcal{O}(N^2)$; PE with MAC is $\mathcal{O}(N)$ for EE and $\mathcal{O}(N \log N)$ for $t$-SNE.

### 3. MNIST dataset.

- We train a $t$-SNE embedding with a neural net $28 \times 28$–$500$–$500$–$2000$–$2$ on $N = 60\,000$ MNIST handwritten $28 \times 28$ digits dataset, using entropic affinities.
- We reuse most of the code needed for the experiment:
  - $\mathbf{Z}$ step: spectral direction minimization, $N$-body approximation.
  - $\mathbf{F}$ step: deep net pretraining, minibatch optimization with constant step size and momentum.