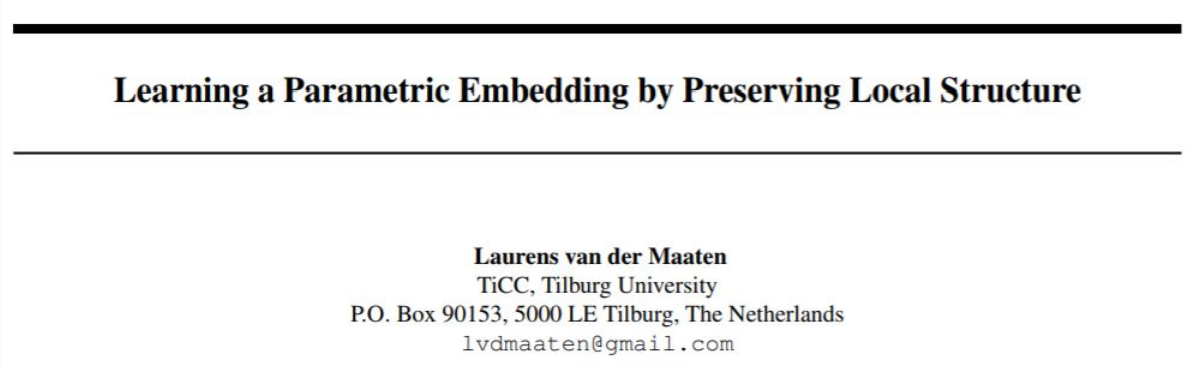
Progress Report

Kuei-Yueh (Clint) Ko

Paper Reading: parametric t-SNE

Once I have a t-SNE map, how can I embed incoming test points in that map?
t-SNE learns a non-parametric mapping, which means that **it does not learn an explicit function that maps data from the input space to the map**. Therefore, it is not possible to embed test points in an existing map (although you could re-run t-SNE on the full dataset).

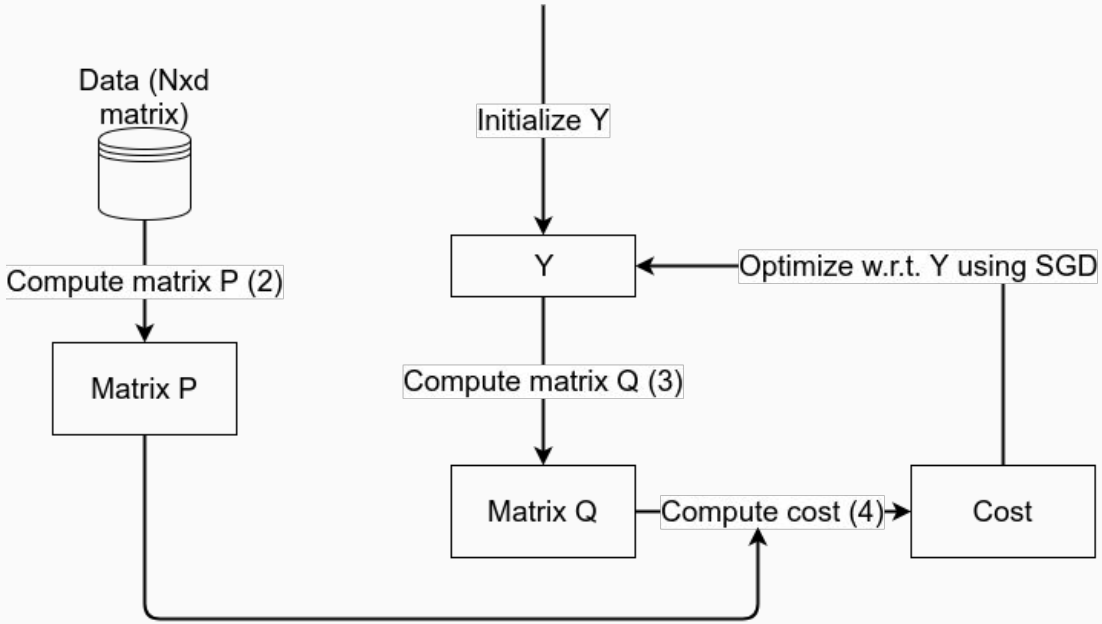
By Laurens van der Maaten



Learning a Parametric Embedding by Preserving Local Structure

Laurens van der Maaten
TiCC, Tilburg University
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
lvdmaaten@gmail.com

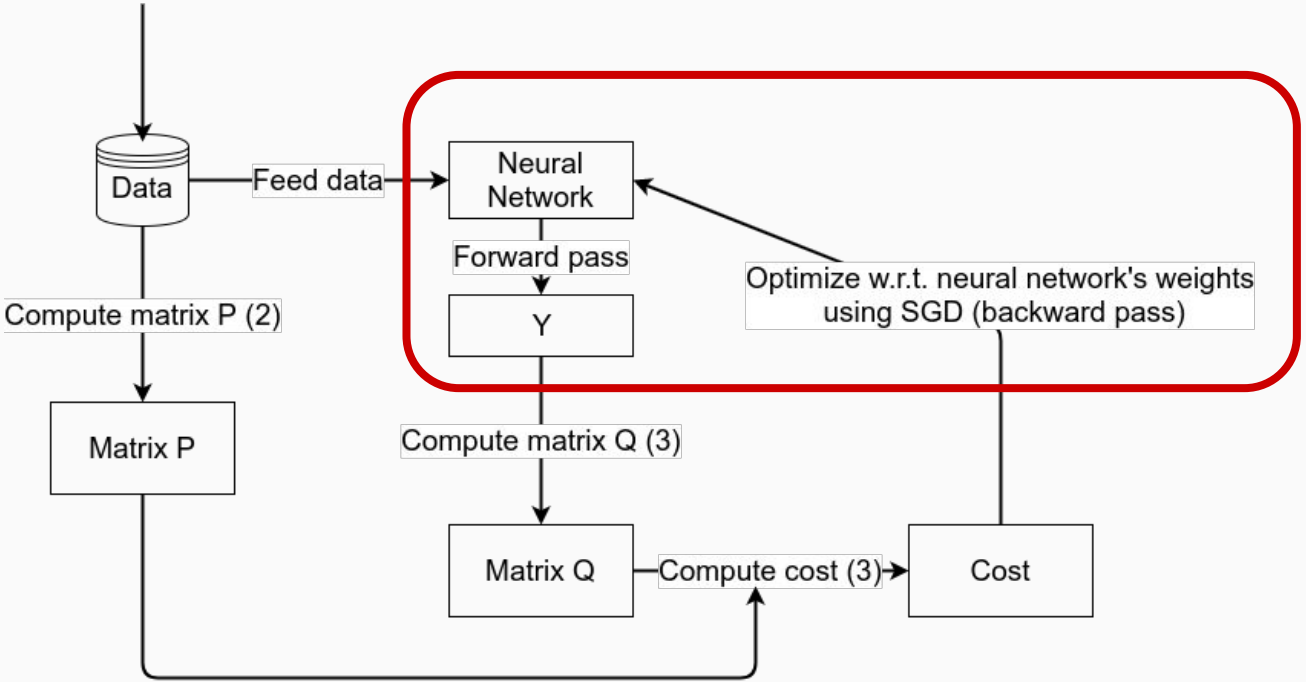
Paper Reading: parametric t-SNE



Original t-SNE Algorithm

https://medium.com/jungle-book/automatic-feature-extraction-with-t-sne-62826ce09268

Paper Reading: parametric t-SNE



Parametric t-SNE Algorithm

https://medium.com/jungle-book/automatic-feature-extraction-with-t-sne-62826ce09268

Try parametric t-SNE

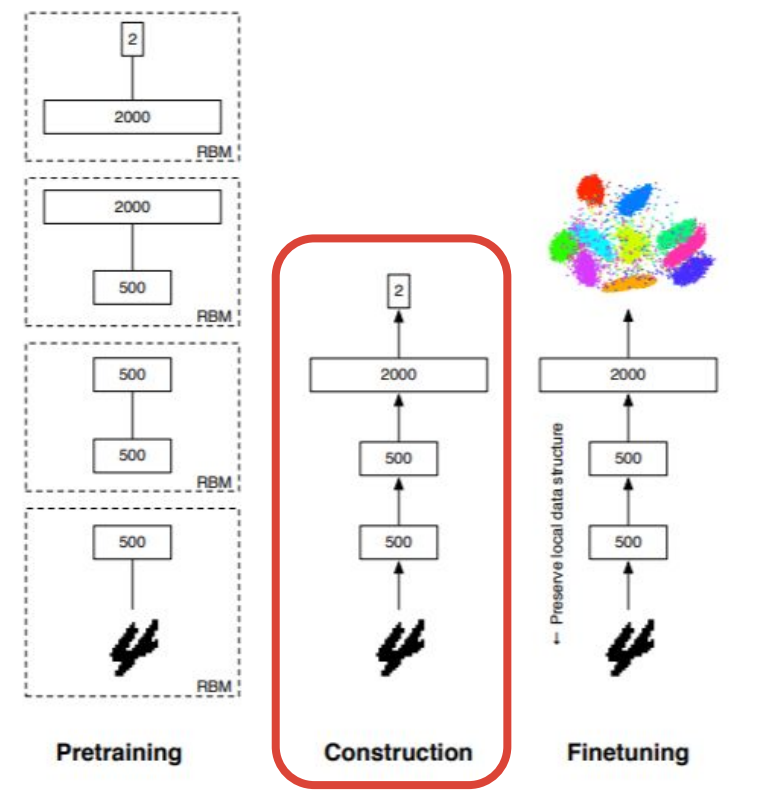


Figure 1: Overview of the three-stage training procedure of a parametric t-SNE network.

```
model = Sequential()
model.add(Dense(500, activation='relu', input_shape=(X_train.shape[1],)))
model.add(Dense(500, activation='relu'))
model.add(Dense(2000, activation='relu'))
model.add(Dense(2))
sgd = SGD(lr=0.1) # Stochastic gradient descent optimizer
%time model.compile(loss=tsne, optimizer=sgd)
```

```
CPU times: user 198 ms, sys: 148 ms, total: 345 ms
Wall time: 337 ms
```

Self defined loss function

Loss = tsne

- https://github.com/kylemcdonald/Parametric-t-SNE/blob/master/Parametric%20t-SNE%20(Keras).ipynb
- https://groups.google.com/forum/#!topic/keras-users/FfPg50m1ozs

Try parametric t-SNE

Distance matrix of t-SNE

Result of the output layer

```
# P is the joint probabilities for this batch (Keras loss functions call this y_true)
# activations is the low-dimensional output (Keras loss functions call this y_pred)
def tsne(P, activations):
#     d = K.shape(activations)[1]
    d = 2 # TODO: should set this automatically, but the above is very slow for some reason
    n = batch_size # TODO: should set this automatically
    v = d - 1.
    eps = K.variable(10e-15) # needs to be at least 10e-8 to get anything after Q /= K.sum(Q)
    sum_act = K.sum(K.square(activations), axis=1)
    Q = K.reshape(sum_act, [-1, 1]) + -2 * K.dot(activations, K.transpose(activations))
    Q = (sum_act + Q) / v
    Q = K.pow(1 + Q, -(v + 1) / 2)
    Q *= K.variable(1 - np.eye(n))
    Q /= K.sum(Q)
    Q = K.maximum(Q, eps)
    C = K.log((P + eps) / (Q + eps))
    C = K.sum(P * C)
    return C
```

Try parametric t-SNE

```
Y_train = P.reshape(X_train.shape[0], -1)
print(X_train.shape)
print(Y_train.shape)
```

```
(60000, 784)
(60000, 5000)
```

Batch_size = 5000

```
%time model.fit(X_train, Y_train, batch_size=batch_size, shuffle=False, epochs=100)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<timed eval> in <module>()
```

```
ValueError: Error when checking target: expected dense_24 to have shape (None, 2) but got array with shape (60000, 5000)
```

Try parametric t-SNE

ValueError: Error when checking target: expected dense_24 to have shape (None, 2) but got array with shape (60000, 5000)

?

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 500)               392500
_________________________________________________________________
dense_22 (Dense)             (None, 500)               250500
_________________________________________________________________
dense_23 (Dense)             (None, 2000)              1002000
_________________________________________________________________
dense_24 (Dense)             (None, 2)                 4002
=================================================================
Total params: 1,649,002
Trainable params: 1,649,002
Non-trainable params: 0
_________________________________________________________________
```