Unsupervised Feature Learning on CyTOF Data with Stacked Auto-Encoders

Krishnan Srinivasan
May 3, 2017

Yale University | Krishnaswamy Lab
1. Introduction

2. Model

3. Experiments

4. Related Work

5. Ongoing Work
Introduction
• The history of artificial neural networks dates back to the 1950s and 60s.
• Largely influenced by the work of Marvin Minsky on *Perceptrons* (1969).
• The prototypical neural network is generally a feed-forward model.
Feed-Forward Networks

• At each "layer," take affine transformations of input features, and perform non-linear "activation," of output to feed to next layer.

• Final layer produces an output that can be evaluated by a loss function, $\mathcal{L}$.

• Standard non-linear functions must be differentiable, as this is how back-propogation is used to update the transformation weights to reduce final loss.

Example of a single hidden-layer network:

$$f(x) = \tanh(Wx + b)$$
• A way of adjusting the weights in a neural network to minimize (or maximize) loss (or reward) function.
• Typically viewed as a convex optimization problem, finding the best setting of parameters $W_i$ and $b_i$ ($i$ being the $i$th layer) to minimize the loss $\mathcal{L}$.
• Many optimizers to choose from, the most common is stochastic gradient descent, which performs batched updates to parameters.

Gradient descent parameter update:

$$\beta_{t+1} = \beta_t - \eta \nabla_{\beta_t} \mathcal{L}$$
Back-Propogation: Notation

- Let $w_{j,k}^l$ denote the weight of node $k$ in layer $l - 1$ applied by node $j$ in layer $l$
- $b_j^l$ denote the bias of node $j$ of layer $l$
- $z_j^l = \sum_k (x_k^{l-1} \cdot w_{j,k}^l) + b_j^l$
- $a_j^l$ denotes the activation from applying a non-linear function to node $j$ in layer $l$.
- Sigmoid activation function: $\sigma$, where $a_j^l = \sigma(z_j^l) = \frac{1}{1+e^{-z_j^l}}$
- $M$: minibatch (subset) of input data samples
- $\delta_j^l = \frac{\partial L}{\partial z_j^l}$: empirical error partial derivative at node $j$ in layer $l$
  (which we use to update the parameters)
- $L$: Final layer
The loss changes with respect to the final layer’s output, so we calculate the updates to the final layer using the chain rule:

\[
\delta^L_j = \frac{\partial \mathcal{L}}{\partial z^L_j} = \sum_k \frac{\partial \mathcal{L}}{\partial a^L_k} \cdot \frac{\partial a^L_k}{\partial z^L_j} = \frac{\partial \mathcal{L}}{\partial a^L_j} \cdot \frac{\partial a^L_j}{\partial z^L_j} = \frac{\partial \mathcal{L}}{\partial a^L_j} \cdot \sigma'(z^L_j)
\]
Lastly, note that the error $\delta_j^l$ for any previous layer $l < L$ is

$$\delta_j^l = \frac{\partial L}{\partial z_j^l}$$

$$= \sum_k \frac{\partial L}{\partial z_k^{l+1}} \cdot \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

$$= \sum_k \delta_k^{l+1} \cdot \frac{\partial z_k^{l+1}}{\partial z_j^l}$$

Finally, the general gradients for weights and biases are as follows:

- for weights: $\frac{\partial L}{\partial w_{j,k}^l} = a_k^{l-1} \delta_j^l$
- for biases: $\frac{\partial L}{\partial b_j^l} = \delta_j^l$

From this, we can calculate the gradients for all of the parameters in the model by propagating the loss signal from the final layer’s parameters all the way to the parameters in the first hidden layer.
1. \( l_i = \|y^{(i)} - \tilde{y}^{(i)}\|_2^2 \) is the empirical loss for element \( x^{(i)} \) in minibatch \( M \)

2. Calculate errors \( \delta \) for each layer, working backwards from last layer.

3. Repeat calculating all gradients for every element \( x^{(i)} \).

4. Calculate the minibatch gradient update by
\[
\sum_{x^{(i)} \in M} \nabla_{\beta} l_i \approx \nabla_{\beta} \mathcal{L}
\]
for all parameters \( \beta \) (weights and biases).

5. Update all parameters \( \beta \) at \( t \)th minibatch using gradient descent step:
\[
\beta_{t+1} = \beta_t - \eta \nabla_{\beta_t} \mathcal{L}
\]
Auto-Encoders (AE)

- Hinton and Salakhutdinov 2006, *Reducing the Dimensionality of Data with Neural Networks*
- Example of a feed-forward network formed by two subcomponents:
  - Encoder: maps input to a (usually) lower-dimensional embedding
  - Decoder: takes the transformed input and attempts to reconstruct the original input
- To facilitate learning to encode and decode, the loss function generally used is either MSE or cross-entropy.
- The goal is to produce a code $y$ for ea. $x$ that captures main factors of variation in the data.
- This is similar to projecting the data with PCA dimensions, can be done by AEs by using MSE loss and omitting non-linear activation.
Auto-Encoders

Figure 1: A traditional autoencoder
• Early on, AEs used for layer-wise initialization of deep networks (Bengio et al. 2006).
• Addition of gaussian noise and dropout of input values enabled use of AEs to denoise input (Vincent et al. 2008).
• Denoising Auto-Encoders (DAEs) learn invariant, robust features from the data.
• Later on, Bayesian approach to learning latent variable probability distributions was brought to AEs, allowing them to become generative models known as Variational Auto-encoders (VAEs) (Kingma and Welling 2013).
Figure 1: Figure from Kingma and Welling paper, visualizing learned latent variable manifold
Sparse Auto-Encoders (SAE)

- Extracts useful features from high-dimensional data by using a sparse high-dimensional code
- Sparsity is attained by adding $l_1$-norm of hidden layer activation to loss function

New MSE loss function for SAE:

$$\mathcal{L}(x, \hat{x}) = \frac{1}{n} \|x - \hat{x}\|_2^2 + \lambda \|f(x)\|_1$$

Where $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ represents the output of the encoding layer, with $m > n$. 
CyTOF Data

- Mass cytometry is a method of measuring gene expression at a cellular level
- Yale School of Medicine has a CyTOF group
- Can be used to study subpopulations of cells among patients (such as T cells and B cells, which are related to the immune system)
- The datasets I have worked with are:
  - Clean blood sample from Dr. Kang’s Lab
  - Blood samples from kidney transplant patients (young and old) from Dr. Montgomery’s lab.
  - Blood samples from flu patients prior to vaccination from Dr. Kang’s lab
Model
Motivation

• t-SNE is a commonly used model by biologists to visualize high-dimensional CyTOF datasets.
• Has two main problems:
  • Fails to preserve meaningful inter or intra "group" distances, which can give misleading or unhelpful visualizations.
  • Takes a very long time to fit on datasets with more than 100,000 points.

Figure 2: t-SNE embedding of brain tissue data, Zitnik and Leskovec 2017
Motivation

- Neural networks can take care of the second problem, since they use minibatches to update.
- Auto-encoders are a useful unsupervised method to learn robust features from high-dimensional data, we hope to find combinations of marker expression that can be composed to find the main factors of variation to reduce the dimensionality of our data.
- Manifold hypothesis of deep learning leads us to believe that these stacked auto-encoder transformations can capture invariant properties of the manifold that allows us to perform this dimensionality reduction. (Olah 2014).
Goals

- Identifying clusters in gaussian mixture models (which have similar properties to biological marker data), and other test datasets (like MNIST)
- Evaluating the learned feature representations by clustering embeddings (high and low-dimensional)
- Identifying cell-type clusters, labeled using traditional gating methods
- Clustering and visualizing auto-encoder embeddings to see if there are observable cell population differences between patients of different phenotypic traits
  - young vs. old
  - healthy vs. unhealthy
The model we started with was a sparse auto-encoder with a two-dimensional middle layer with linear activation to linearly project the most important/variant features of the high dimensional input into a visualizable plot. Figure 3 shows the general architecture used for the experiments that follow.

**Figure 3:** Sparse Auto-encoder used in experiments
Experiments
Gaussian Mixture Model

Test dataset of initially 3 dimensions sampled from two multivariate gaussians with overlap, and train the auto-encoder.
MNIST Data

MNIST is a large dataset of 28x28 images of handwritten digits, labeled with the correct digit number (Lecun and Cortes, 1998). Below is the visualization of the data, colored by MNIST digit labels, showing both cluster separation and inter-cluster distance similarity between digits. The left was made with an SAE with 1 layer, while the embedding to the right is a stacked SAE.
Healthy Blood Data

- Healthy blood CyTOF dataset from Dr. Kang’s lab.
- Used gating to label cells, and calculated rand index to determine accuracy of cluster assignment (done by k-means on the embedding).
- Compared rand index of embedding + k-means to Phenograph, which is a single-cell data clustering method using network representation of cell-cell similarities.

$$ R = \frac{\#\text{same} + \#\text{different}}{\binom{n}{2}} $$

<table>
<thead>
<tr>
<th>Middle Layer Size</th>
<th>Phenograph</th>
<th>Sparse Layer Clust.</th>
<th>Middle Layer Clust.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>39 %</td>
<td>53 %</td>
<td>53 %</td>
</tr>
<tr>
<td>7</td>
<td>” ”</td>
<td>69 %</td>
<td>69 %</td>
</tr>
</tbody>
</table>

Table 1: Comparing Rand Scores between true clusters and clusters from Phenograph and Sparse/Middle Layer codes
Figure 4: Embedding colored by cell type
Figure 5: Activation of sparse coding dimension, plotted over middle layer visualization.
• We began to find further into our experiments that the embeddings generated from the model struggled to separate clusters when combining data from different patients.

• We contribute the following loss function to reduce within cluster distance and increase cluster separation.

\[ \mathcal{L}(x, \hat{x}, f(x)) = \|x - \hat{x}\|_2^2 + \frac{\text{dist}_{\text{within}}}{\text{dist}_{\text{outside}}} \]

• \( f(x) \) is the transformation mapping the sparse coding of the input to clusters, using a softmax activation function.

• We later found that a similar clustering concept was used in a recent paper attempting to cluster image data using auto-encoders (Xie et al. 2016).
Figure 6: New model using cluster assignments from parallel network
Ruth Data

- Looking at old and young kidney-transplant patient data, initial sparse auto-encoder (SAE) was separating patients into clusters.
- When we apply the cluster encoder, we find cluster separation to be better, and more indicative of patient groups.
Figure 7: Marker heatmap showing gene expression in each cluster
Related Work
Related Work

- Unsupervised Deep Embedding for Clustering Analysis (ICML 2016)
- Metric Learning with Adaptive Density Discrimination (ICLR 2016)
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (ICLR 2016)
- Learning a Parametric Embedding by Preserving Local Structure (AISTAT 2009)
Unsupervised Deep Embedding for Clustering Analysis

- Very similar in network structure to our approach, but use a very un-intuitive clustering method and loss function.
- Cluster with KL divergence between a locally sampled neighborhood cluster assignment ($Q$) from a centroid with a target cluster assignment ($P$).
- Use unsupervised clustering accuracy ($\text{ACC}$) to evaluate.

$$\text{ACC} = \max_m \sum_{i=1}^n \mathbb{1}\{l_i = m(c_i)\}$$
Metric Learning with Adaptive Density Discrimination

- Develop a loss function called "magnet loss" used for distance metric learning.
- Compare three different loss functions using GoogLe-Net as the transformation mapping images to the representation space.
- Show that Magnet Loss is able to min. w/in cluster dist. while also having distance respecting inter-class similarity.

\[ \mathcal{L}_{\text{magnet}} = \frac{1}{MD} \sum_{m} \sum_{d} \left\{ -\log \frac{\exp\left(-\frac{1}{2\delta^2} \left\| r_m^d - \hat{\mu}_m \right\|_2^2 - \alpha \right)}{\sum_{\hat{\mu}:C(\hat{\mu}) \neq C(r_m^d)} \exp\left(-\frac{1}{2\delta^2} \left\| r_m^d - \hat{\mu} \right\|_2^2 \right)} \right\} + \]
Metric Learning with Adaptive Density Estimation

Figure 1: Distance metric learning approaches sculpt a representation space where distance is in correspondence with a notion of similarity. Traditionally, similarity is specified a-priori and often strictly semantically. In contrast, Magnet Loss adaptively sculpts its representation space by autonomously identifying and respecting intra-class variation and inter-class similarity.

Figure 8: Showing inter-cluster similarity in ImageNet samples
Ongoing Work
• Looking at Flu patient single-cell data with young and old patients
• Interested in finding if we can separate patients into clusters (like with kidney patient data)
• Results indicate clear cluster separation.
Acknowledgements
&
Questions
• **Convex Clustering** is a clustering method that also is able to do imputation, and uses a loss function similar to ours with a graph theoretic base
• Using cosine dissimilarity instead of euclidean distance in loss function (since we see the data separates into cones in the embedding)
• Use l1-regularization in the cluster loss
• Come up with a silly name for this model