EECS 445 Discussion 8: RNNs and Unsupervised Methods

March 18, 2021

1 Recurrent Neural Network

- Recall Recurrent Neural Networks can be used to analyze sequences of data.
- The idea behind RNNs is to make use of sequential information. RNNs are called recurrent because they perform the same task for every element of sequence. The output at each step depends on the previous inputs.

Example 1. Construct a simple RNN model to perform the following sequence classification problem: Generate sequences $\bar{x}^{(i)}_{1\cdots t}$ of sequence length between [5,15] with random values between 0 and 1. Assign a binary label $\in \{0, 1\}$ to each input. If the cumulative sum of the values from $\bar{x}^{(i)}_1$ to $\bar{x}^{(i)}_j$ lies above 3 then classify $\bar{x}^{(i)}_j$ as 1 and 0 otherwise.

For example, if $\bar{x}^{(i)} = [0.6, 0.9, 0.4, 0.0, 0.9, 0.1, 0.8, 1.0, 0.9, 0.5]$.
Its cumulative sum is $[0.6, 1.5, 1.9, 1.9, 2.8, 2.9, 3.7, 4.7, 5.6, 6.1]$.
Then $\bar{y}^{(i)}$ will be $[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0]$

For the loss function,

- We use Binary Class Entropy (BCE) between the ground truth $\bar{y}^{(i)}_{true}$ and $\bar{y}^{(i)}_{pred}$.

For initialization,

- For the hidden state at timestamp 0, h_0, which will be fed into the RNN with $\bar{x}^{(i)}_1$, h_0 is initialized to 0.

Solution: Run `RNN Jupyter Notebook`. Generally, in RNN, the output y_t and hidden state h_t can be abstracted as

$$h_t = f(h_{t-1}, x_t)$$
$$y_t = g(h_{t-1}, x_t)$$

The hidden state is set to be a vector of size 4. $h \in \mathbb{R}^4$.
In the Jupyter Notebook, we use a fully connected layer with `nn.Linear(input_size + hidden_size, hidden_size)` for $h_t = f(h_{t-1}, x_t)$. We use another fully connected layer after we yield the hidden state h_t with `nn.Linear(hidden_size, output_size)` for y_t.
At timestamp t, the input is $\bar{x}^{(i)}_t$, the hidden input is the hidden output from $t-1$ which we denote as h_{t-1}. We combine the input $\bar{x}^{(i)}_t$ and h_{t-1} as the final input to the building block. Two sequential fully

connected layers are trained for the combined input. The first fully connected layer yeilds the result for h_t and the fully connected layer with sigmoid activation that follows the first fully connect layer yield the the output for timestamp t, y_t.

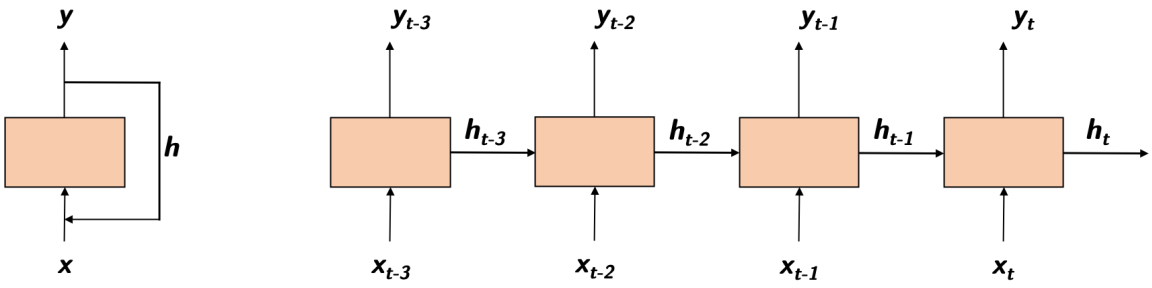


Figure 1: RNN Structure

Example 2. How could we combine a convolutional neural network with an RNN to classify videos?

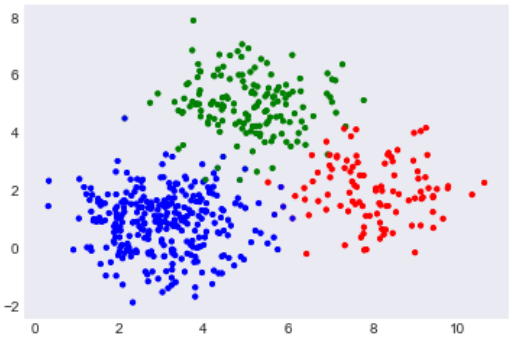
Solution: A video is a sequence of images or frames, and hence can be regarded as a high-level RNN/LSTM. Hence, one possible architecture would be as follows,
The CNN-LSTM architecture involves using Convolutional Neural Network layers for feature extraction on input data combined with LSTMs to support sequence prediction. One possible choice is that the CNN will output the flattened output of its late convolutional layer as the input to the LSTM.

2 Unsupervised Learning Overview

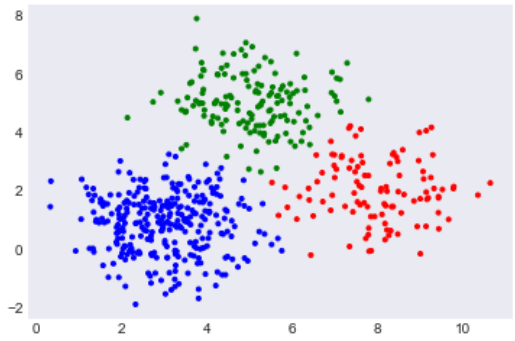
Thus far throughout the course, we have focused on *supervised* learning methods, which learn a mapping from inputs $\{\bar{x}^{(i)}\}_{i=1}^n$ to some semantic label $\{y^{(i)}\}_{i=1}^n$ in either a discrete or continuous domain. In practice, these semantic labels can pose a problem: they may require arduous human annotation, and can be a source of noise. Here, we discuss powerful methods for *unsupervised* learning, which are attractive given their lack of reliance on explicit labels. We focus on two applications: clustering and matrix completion. Common to both of these is a notion of *similarity* or *affinity* between data points. Without labels, it is this similarity between data points that allows us to extract meaningful information from our data.

3 Clustering

The goal of a clustering algorithm is to uncover underlying structure within our data. For example, consider a data distribution consisting of points sampled from one of three Gaussian distributions (Figure 2a). In this case, the goal of a clustering algorithm is to assign each point to the cluster representing the Gaussian distribution from which it was sampled.



(a) Data sampled from 3 independent Gaussians



(b) A k-means clustering on independent Gaussians

Here, the data clearly fits into one of three distributions, so when fitting our unsupervised algorithm it is a natural choice to search for a division of data into three clusters. However, it is not always the case that we know the correct number of clusters. To determine a suitable value for k, the number of clusters, we define a metric for evaluating a cluster and perform a hyperparameter search for a value of k that maximizes this value under the constraint that we prefer "simpler" solutions with fewer clusters. Visualizing the data by creating a low-dimensional representation—via PCA or t-SNE, for instance—can also provide some intuition for finding a sensible number of clusters. Naive clustering methods such as k-means perform very well on globular clusters with similar variance (Figure 2b). Other clustering methods are needed for more complex distributions.

3.1 K-Means Clustering

The typical objective of clustering is to minimize the distance between points in the same cluster and maximize the distance between points in different clusters. Since this optimal clustering cannot be efficiently found, k-means serves as a relatively fast, easy to understand approximation. The algorithm is as follows:

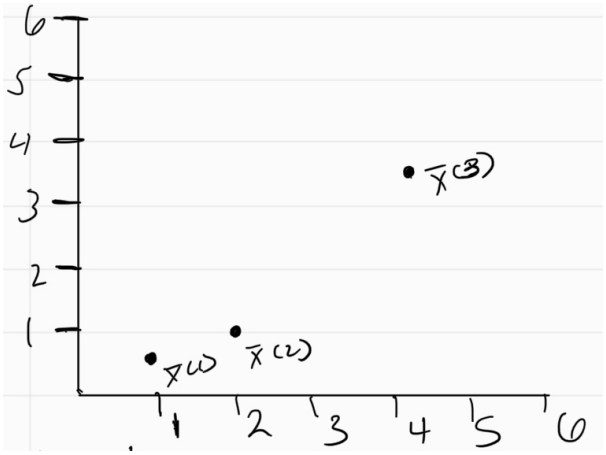
Given a data set with n points, $\bar{x}^{(1)} \ldots \bar{x}^{(n)}$:

1. Randomly choose k cluster means, $\bar{\mu}^{(1)} \ldots \bar{\mu}^{(k)}$
2. Iterate between two main steps until convergence:
   (a) Assign each data point $\bar{x}^{(j)}$ to the cluster i with the mean $\bar{\mu}^{(i)}$ that is closest to $\bar{x}^{(j)}$

(b) Recompute each cluster mean $\bar{\mu}^{(i)}$ based on all data points now assigned to that cluster i

K-means converges when the cluster assignments or means do not change between iterations.

Discussion Question 1. *Consider a dataset consisting of three data points as plotted below.*

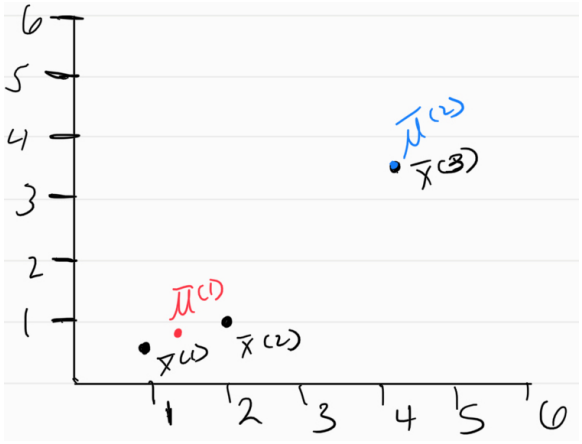


From the image, we have $\bar{x}^{(1)} = [1, 0.5]^T$, $\bar{x}^{(2)} = [2, 1]^T$, $\bar{x}^{(3)} = [4, 3.5]^T$. *Run K-means clustering (using Euclidean distance) with k=2 and using the following initial centroids:* $\bar{\mu}^{(1)} = \bar{x}^{(1)}$ *and* $\bar{\mu}^{(2)} = \bar{x}^{(2)}$.

1. *What are the final centroid values after convergence?*

Solution:

1. *The image below shows the original dataset with the centroids plotted after convergenc.*



Iteration 1 $\bar{\mu}^{(1)} = \bar{x}^{(1)}$ *and* $\bar{\mu}^{(2)} = \bar{x}^{(2)}$

- *Assign clusters:* $\bar{x}^{(1)} \to 1$, $\bar{x}^{(2)} \to 2$, $\bar{x}^{(3)} \to 2$
- *Recompute means:*
  $\bar{\mu}^{(1)} = \frac{\bar{x}^{(1)}}{1} = [1, 0.5]^T$
  $\bar{\mu}^{(2)} = \frac{\bar{x}^{(2)} + \bar{x}^{(3)}}{2} = \frac{1}{2}([2, 1]^T + [4, 3.5]^T) = [3, 2.25]^T$

Iteration 2 $\bar{\mu}^{(1)} = [1, 0.5]^T$ *and* $\bar{\mu}^{(2)} = [3, 2.25]^T$

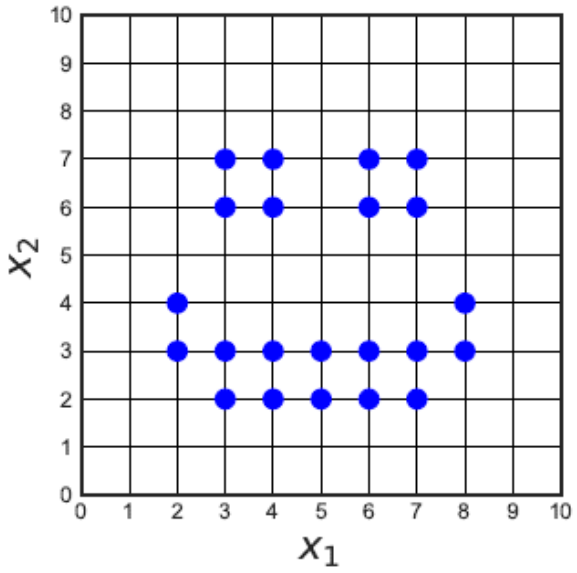
- *Assign clusters:* $\bar{x}^{(1)} \to 1$, $\bar{x}^{(3)} \to 2$
  For $\bar{x}^{(2)}$*, it is not immediately clear which cluster it belongs to, so we can compute the euclidean distance to each point:*
  $dist(\bar{x}^{(2)}, \bar{\mu}^{(1)}) = \sqrt{(2-1)^2 + (1-0.5)^2} = \sqrt{1.25}$
  $dist(\bar{x}^{(2)}, \bar{\mu}^{(2)}) = \sqrt{(2-3)^2 + (1-2.25)^2} = \sqrt{2.5625}$
  $\bar{x}^{(2)}$ *is closer to* $\bar{\mu}^{(1)}$*, so we can conclude* $\bar{x}^{(2)} \to 1$
- *Recompute means:*
  $\bar{\mu}^{(1)} = \frac{\bar{x}^{(1)} + \bar{x}^{(2)}}{2} = \frac{1}{2}([1, 0.5]^T + [2, 1]^T) = [1.5, 0.75]^T$
  $\bar{\mu}^{(2)} = \bar{x}^{(3)} = [4, 3.5]^T$

Iteration 3 $\bar{\mu}^{(1)} = [1.5, 0.75]^T$ *and* $\bar{\mu}^{(2)} = [4, 3.5]^T$

- *Assign clusters:* $\bar{x}^{(1)} \to 1$,$\bar{x}^{(1)} \to 1$, *and* $\bar{x}^{(3)} \to 2$
  Since the cluster assignment has not changed, our algorithm has converged!

The final centroid values are $\bar{\mu}^{(1)} = [1.5, 0.75]^T$ *and* $\bar{\mu}^{(2)} = [4, 3.5]^T$

Discussion Question 2. *Consider the data plotted in the figure below, which consists of 22 points that make up a smiley face.*



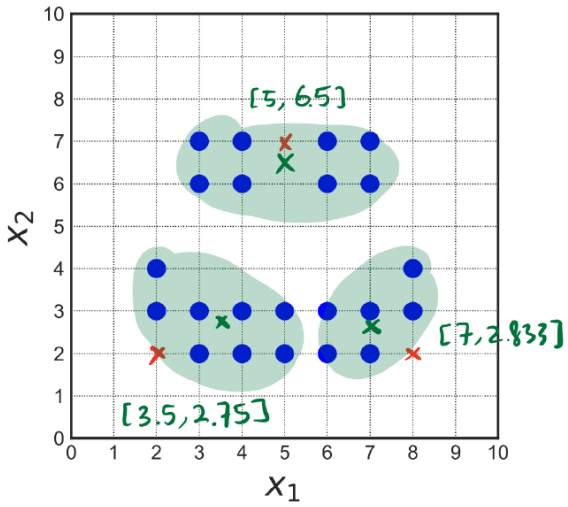
K-means clustering (with $k = 3$ and using Euclidean distance) is initialized with three centroids at coordinates $\bar{z}^{(1)} = [2, 2]^T$,$\bar{z}^{(2)} = [8, 2]^T$, and $\bar{z}^{(3)} = [5, 7]^T$.

1. *What are the final clusters and corresponding centroids after convergence? Ties are broken consistently, where cluster i is favored over cluster j in a tie if $i < j$.*

2. *Would we be able to cluster our smiley face into two 'eye' clusters and one 'mouth' cluster by changing our centroid initialization? Why or why not?*

Solution:

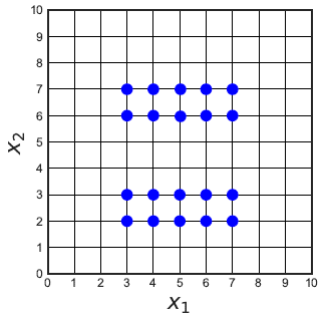
1. *The figure is provided after one iteration, which is all that is required because the cluster assignments do not change.*

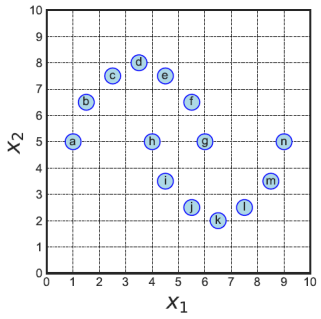


2. *Sadly, we cannot. If we know each each of the four points corresponding to 'eyes' is its own cluster, the centroids must be* $\bar{z}^{(left)} = [3.5, 6.5]^T$,$\bar{z}^{(right)} = [6.5, 6.5]^T$,,$\bar{z}^{(mouth)} = [5, 2.78]^T$, *at convergence. However, this means that the point at each tip of the mouth will be closer to one of the 'eye' centroids than the 'mouth' centroid, making the assignments incorrect.*

   For example, on the right side: $(6.5 - 8)^2 + (6.5 - 4)^2 = 8.5 < (5 - 8)^2 + (2.78 - 4)^2 = 10.54$

Discussion Question 3. *Consider two more example data sets in the following two figures:*





For each figure, provide initial values for centroids (k = 2) for which k-means will ***not*** *converge to an result that best captures the trend in the data. Explain why this initialization, or the data in general, causes an indesirable outcome.*

Solution:

1. For the left figure, setting centroids equal to [4, 4.5] and [6.5, 4.5] will result in a horizontal split between clustering, when splitting them based on the X_2 axis is clearly better. Even if we separated the two rectangles by a near infinite amount vertically, we can still choose two clusters that will separate them based on X_1, thus causing the total distance between points and their centroids to be arbitrarily worse than the global minimum.
2. For the right figure, **any** centroid intialization will converge to a sub-optimal solution because the clusters are not globular clusters.