Module 7

Unsupervised and Self-Supervised Learning: From DBNs and Autoencoders to K-Means Clustering

▼ GANs deep dive:

Generative adversarial networks are powerful models that provide many useful applications. A popular use for GAN's is in the generation of fake images given a dataset of images. However, one particular problem with certain models of this type is that they suffer from what is called mode-collapse. This is when the generated images do not represent the diversity of the initial dataset (Rosca et al., 2017).

In order to offset this, researchers have devised a new model paradigm that leverages the characteristics of GAN's and variational autoencoders into a new framework called Variational auto encoder-based GAN's (VAE-GANs) (Rosca et al., 2017). Autoencoders are generally concerned with extracting and deriving relevant characteristics of unlabeled data, by introducing a "bottleneck", a hidden layer smaller than an input layer that forces compression of data to limited dimensions (Jordan, 2018). Autoencoders are modeled in such a way that they are sensitive enough to input data that it can accurately reconstruct it (data reconstruction loss), whilst also being insensitive enough to prevent from overfitting.

Variational autoencoders take this one step further by introducing probabilistic distributions of such attributes in the data rather than discrete values (Jordan, 2018). VAE's are good at accounting for representational variety when generating images, however the images are often blurry. Conversely, GAN's produce crisp images but overall suffer from mode-collapse. Researchers leverage the two models together in order to offset the drawbacks of each.

To do this, they formulated a new function for their model called α-GAN, which combines an adversarial loss with a data reconstruction loss (Rosca, et al., 2017). Image blurriness is solved due to the adversarial loss function, while the reconstruction loss function helps "ground" the generator in all the available training data. According to the article, "GAN's allow few distributional assumptions to be made about the model, whereas VAE's allow for inference of the latent variables which is useful for representation learning, visualization and explanation" (Rosca et al., 2017). In this way, a VAE-GAN can provide robust generative models for images that do not suffer from mode-collapse and have a well-distributed representation of generated images.

▼ Unsupervised or self-supervised learning opens for the possibility of self-teaching machines. From a technical perspective, how does this benefit the field and the development of AI software? How might it hinder it?

Unsupervised learning and self-supervised learning are incredibly beneficial to the field and development of AI software. First, the distinction of unsupervised vs self-supervised learning is generally known to be that self-supervised is a subset of unsupervised. Unsupervised learning is essentially a learning method that requires no supervision. Typically, unsupervised learning involves clustering, grouping, and dimensionality reduction. Self-supervised learning looks for relationships within the data to draw conclusions for regression and classification tasks.

Basically, with these types of learning, as opposed to supervised learning, we are able to take in unlabeled data and give it labels autonomously. In supervised learning, this is done manually by the creator of the datasets/systems that collect data.

These types of learning have many benefits to the AI field, in general, but some of the biggest benefits that they offer are:

- They require a lot **less complexity** than supervised counterparts. The data comes unlabeled (which is more common in the wilderness that is the internet) and these techniques are especially good for handling such data.
- The data is analyzed in **real time.** We get to learn data as it is initially presented to us. It is raw and not filtered in any way. This leaves room for noise but also leaves room for much more insight and discovery of patterns and relationships of different features.

- In terms of benefiting the field of AI, these learning methods can **make the preprocessing step much easier**. We can ask our self-supervised models to label data for us, allowing us to spend more time on training a supervised accurate model.
- We can also discover **patterns and trends** that we weren't previously able to detect manually, especially on **very large datasets**.

Some ways that these learning methods can hinder the AI field:

- More autonomy could mean less human knowledge of the data and hence **more black boxes to be created**. I personally believe that sometimes techniques in which we automate too much can hinder our own abilities and understanding and enable us to miss key steps in the information retrieval process. This can lead to disastrous outcomes.
- We also have to accept less accuracy from the self-supervised models than their supervised counterparts. Since the models do the labeling themselves, we aren't there to guide it when it has experienced a 'technically' correct answer but not the one we, as humans, are looking for. This has been the case in many instances, such as ML Unity Agents and the way that they move across a space as a humanoid. I recall watching a video in which the model didn't use its two legs to run but rather its entire body to 'bounce' and leap further to the finish line.
- Sometimes we cannot ascertain results from an unsupervised model. We get things that don't make sense to us, or, since the number of classes is not known prior to letting the model run, we might not get a result that we can reasonably look at since the number of classes might be too high.

There are many benefits to unsupervised and self-supervised learning methods. I believe that we must use them in tandem with supervised models. When we are looking to find new insights into unknown data, make preprocessing easier, or simply make a more generalized model, we should use these methods. When we are presented with new data, these types of models can find unknown labels unlike a supervised model that is prefixed on predefined labels. This is crucial in getting closer to more scalable, robust, and generalized artificial intelligence.

▼ In what use-cases is k-means clustering more useful than other techniques?

k-means clustering may be more useful than other techniques when an engineer has knowledge that the underlying data process satisfies the assumptions made by the model. For example, *the k*-means model imagines the existence of a "ball" surrounding each of the *k* centroids, within which are all the points belonging to that cluster. If an engineer knows that the data is not so organized and separable then they may prefer another model, as *k*-means is guaranteed to mis-cluster some points in that case.

Tuning *k* is another way an engineer's prior knowledge would be useful. The algorithm will construct *k* clusters no matter how many clusters *should* be found. Only by inspecting the resulting clusters can an engineer determine whether something has gone wrong, and even then there may be some challenges as measuring the goodness of clusters can be subjective.

▼ Why are activation functions so important?

- Without them, an artificial neuron’s output would equal the sum of its weighted input.
- The signal would always be passed to the next neuron and all neurons would be activated all the time.
- By using activation functions, we can introduce thresholds into our artificial neurons that prevents neurons from being activated, if their information can be considered irrelevant.
- Activation functions also enable an artificial neuron to solve more complex problems by introducing non-linearity.

▼ How do activation functions enable an artificial neuron to solve more complex problems by introducing non-linearity? (**activation function breakdown**)

Let's say something like yellow is 1 and blue is 2 etc. The mimic network interprets that data and can assign it a weight. This weighted function can determine whether or not a specific neuron branch gets 'excited' and is activated. Remember that a signal can make an actual neuron excited or not? An artificial neuron has this ability too! We can mimic the natural properties of an actual neuron by introducing so called weights. A weight is multiplied
with the input and results in a new
value. Coming back to our color detection example, we could set the properties of

an artificial neuron so that it always gets excited about the color yellow but not so much over the colors green and blue. In a simplified mathematical way, the artificial neuron multiplies 1 with a weight of 10 but multiplies green and blue only with a weight of 1. In result, this particular artificial neuron will always get **more excited** about the color yellow than any other color. I realize this is a really rudimentary break down. But, it is essential to our understanding of how these function. The process is quite a bit more complex than this. This just seems a bit easier to digest.

▼ What is self-supervised learning?

a form of supervised learning that labels its own data automatically (usually by training on a small labeled dataset)

▼ Unsupervised learning uses what kind of data?

unlabeled data

▼ What are some examples of unsupervised models for NLP?

- autoencoders
- autoregressive language models

▼ Reinforcement learning is not what?

supervised nor unsupervised learning

▼ What is reinforcement learning (RL)?

Reinforcement learning seeks to teach a model (in the form of an agent) to learn the correct decisions (in the form of actions tied to some symbolic planning system, predictions, or some other kind of output) on its own.

▼ How does reinforcement learning work?

- RL is accomplished by developing a programmatic structure that provides either positive or negative rewards based on factors in the environment (as seen by the agent as observations) and its own state.

  - This optimizes the agents proclivity toward actions that maximize reward and away from actions that reduce reward.
  - This ideally leads the agent to find the best series of actions for a variety of situations or scenarios.

▼ What should be considered when building an RL model?

  - The layout of the reward function;
  - How to tune hyperparameters relating to the learning rate, discount factors, and various initial conditions;
  - Delayed rewards and how to handle them using a variety of methods such as the Intrinsic Curiosity Module in Unity's ML Agents library;
  - Time-step functions and history vs. state functions (sometimes utilizing LSTM RNNs which are great for memory functions and time-series data).

▼ Model-based RL is usually modeled as what?

a partially-observable Markov Decision Process (MDP)

▼ What is a Markov Decision Process (MDP)?

  - A decision-making process that is based on the idea that "the effects of an action taken in a state depend only on that state and not on the prior history."
  - In other words, past observations are ignored in favor of current observations.
  - This can lead to many problems.

▼ What are Policy Gradient Methods?

  - Policy Gradient Methods utilize gradient descent to optimize parametrized policies to maximize long-term cumulative rewards, and allow for the addition of domain knowledge to further optimize learning.
  - Policy gradient methods can be inefficient at converging on global minima and instead tend to excel at finding distinct local minima.
  - Policy gradient methods are efficient at problems where continuous control of actions and states is required, such as robotic locomotion and video game AI.

▼ What are examples of actor-critic policy gradient methods?

PPO (used in Unity ML Agents as well as by DeepMind), A3C and TRPO

▼ What are model-free RL methods?

- An alternative to policy-based methods are model-free methods such as Q-learning.
- Q-learning is considered off-policy. This means that the agent is not fed any sort of labeled representation of its environment or state; instead, the agent derives an optimal policy directly from its interactions with its environment in an unlabeled manner.
- Model-free learning attempts to predict the value (Q) function of a certain policy.
- Other examples of model-free RL algorithms include TD learning and Monte Carlo methods.

▼ What is Hebbian theory and how does it relate to RL?

- Hebbian theory is a neuroscientific theory explaining associative learning and synaptic plasticity.
- It can be summed up as "neurons wire together if they fire together."
- Synaptic strength increases as neurons activate simultaneously, leading to the creation of memory engrams.
- Hebbian dynamics underlie the theoretical basis of RL.

▼ What is an attention model?

- By using a technique called "Attention" a model can be enabled to focus only on relevant parts of an input sequence.
- An attention model's encoder passes much more data to the decoder than in traditional sequence-to-sequence, including all hidden states (representations of previous states stored by LSTM RNNs in the form of a memory circuit).
- It also scores these states for relevancy and uses mathematical transformations to amplify highly-scored hidden states and reduce the impact of hidden states with low scores.

▼ What are autoencoders?

A type of encoder-decoder that directly maps inputs to outputs

▼ What are some applications of autoencoders?

- dimensionality reduction and information retrieval
- anomaly detection
- image compression
- dataset augmentation

▼ What are Restricted Boltzmann Machines (RBM)?

- Restricted Boltzmann Machines (RBM) are generative, stochastic ANNs that learn a probability distribution of its inputs.
- RBMs have application in a variety of tasks including basic classification, feature learning for other models, and dimensionality reduction.
- You can stack together RBMs and "pre-train" layer-by-layer with a greedy unsupervised technique called Contrastive Divergence to become a Deep Belief Network (DBN).
- Weights can then be fine-tuned in a supervised manner.

▼ What are Deep Belief Networks (DBN)?

- DBNs have a different topology than deep ANNs.
- One large difference is that DBNs do not have a distinct input layer, rather all layers learn the entire input.
- Also, differently than deep ANNs, the nodes in each layer are connected to all nodes in the preceding and anteceding layers.
- The top layers in a DBN are undirected, but all successive layers are directed.

▼ What are Generative Adversarial Networks (GAN)?

- GANs are capable of generating convincing new data by utilizing two competing networks.
- The generative network attempts to replicate an input and pass it to its corresponding adversarial network, which simultaneously attempts to classify

inputs given to it as "fake" (produced by the generative network) or "real" (an actual real-world input).

- If the adversarial network accepts the output from the generative network "real," it is passed as the final output.
- This gives GANs a paradigm-shifting creative ability.

▼ What are some applications of GANs?

- You can use GANs in scientific simulations in areas such as astronomy, cosmology, genetic analysis, high energy physics, and weather to accurately approximate high-cost simulation functions.
- You can use GANs to visualize 2d images such as maps, building layouts, industrial designs, and more into accurately representative 3d images.
- You can use GANs to provide supersampling effects to video game textures as well as dynamically create video game content.
- You can use GANs to create realistic, lifelike images of people and objects that have never existed in reality, opening up an entirely new realm of possibility in art, advertising, and modeling.

▼ What are clustering techniques?

- Clustering techniques are unsupervised techniques that group data points together based on similarity of their features/attributes.
- For example, Google News uses clustering techniques to correlate and group together news stories that are similar or regarding the same event or topic.

▼ What kinds of analysis are well suited for clustering techniques?

Sentiment analysis, trend analysis for market research, or analysis of astronomical or genetic data

▼ What are some clustering methods?

k-means clustering, mean shift clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Expectation–Maximization (EM) Clustering using Gaussian Mixture Models (GMM)

▼ What is k-means clustering?

- K-means clustering is extremely similar to k-Nearest Neighbor regression as it is based on the same methods.
- In k-means clustering, the assumption is there are *k* groups and each of these groups has a distinct center.
- The input data is then separated according to those groups based on distance of each data point from each center, choosing the closest one.
- The centers are then recomputed and the process iterates again.
- Once the change in the value of the centers begins to become negligible, then a good solution has been converged on.