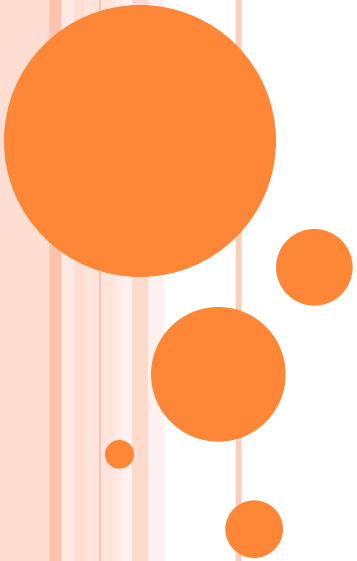


SEMI SUPERVISED LEARNING

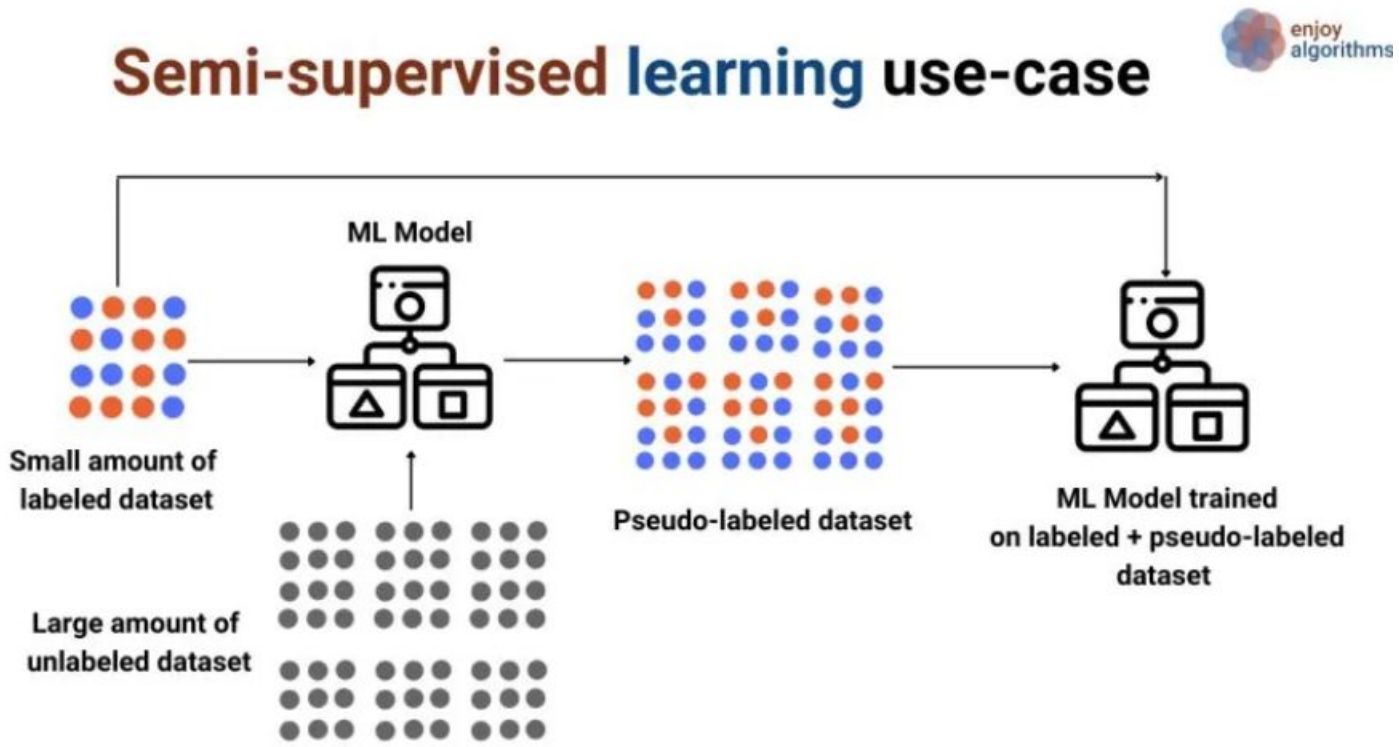
SEMI SUPERVISED LEARNING

- Most of the problems addressed in supervised learning deal with training when we have a lot of labeled data.
- But what if labeled data is expensive?- Semi Supervised learning.
- Semi-supervised learning is the type of machine learning that uses a combination of a small amount of labeled data and a large amount of unlabeled data to train models.
- It is used because getting labeled data is hard in many ways, but it also produces consistent improvement in learning accuracy.

SEMI SUPERVISED LEARNING

- Since acquisition of unlabeled data is relatively inexpensive, adding a small portion of expensive labeled data to it is overall cheap and produce extremely great results in comparison with only unlabelled data.
- Intututively, the learning problem can be seen as an exam and labeled data as sample problems that the teacher solves for the class as an aid in solving another set of problems where you do not have access to the answer, in this case the unlabeled data.
- It is a good practice for the exam since you have seen examples on how to do them and compare them.

TYPES OF SEMI-SUPERVISED LEARNING



Transductive Learning: In this setting, one is given a (labeled) training set and an (unlabeled) test set. The idea of transduction is to perform predictions only for the test points.

This is in contrast to **Inductive Learning**, where the goal is to output a prediction function which is defined on the entire space X.

Comparison using Mathematical representations —

- **X_train** = $\{x_1, x_2, \ldots, x_n\}$ be the set of training inputs.
- **Y_train** = $\{y_1, y_2, \ldots, y_n\}$ be the corresponding labels.
- **X_test** = $\{x_{n+1}, x_{n+2}, \ldots, x_{n+x}\}$ be the test set (unlabeled).

In transductive learning, we aim to find labels **Y_test** = $\{y_{n+1}, y_{n+2}, \ldots, y_{n+x}\}$ for the test set **X_test** without explicitly learning a function $\mathbf{f : X \rightarrow Y}$ for generalization. This kind of a function is learnt during Inductive Learning

WHY TRANSDUCTIVE MODELS DON'T GENERALIZE

- Transductive models are focused on optimizing predictions for a given dataset. They don't attempt to generalize to new, unseen data, because they rely heavily on the structure of the production (or test) data.
- For instance, a graph-based transductive model predicts labels based on the specific structure of a graph, and if the graph changes, the model may not perform as well without retraining.
- SVM follows the first approach(Inductive) of feature scaling, it learns a decision boundary based solely on labeled training data, creating a model that generalizes to new, unseen data.
- **TSVM**, on the other hand, incorporates unlabeled data, often including test data, to refine the boundary. This makes **TSVM** highly effective for the given dataset but at the cost of being specific to it, limiting its ability to generalize.

TRANSDUCTIVE VS. INDUCTIVE LEARNING
DESCRIBE HOW MODELS MAKE PREDICTIONS

Inductive learning

- Aims to create a general rule from the training data, so the model can make predictions on unseen data. For example, a spam filter learns patterns from labeled emails and then applies these patterns to classify new, unseen emails.
- Algorithms like **decision trees** and **neural networks** are typically inductive.

Transductive learning

- Focuses on making predictions for a specific production (or test) set rather than generalizing to unseen data. For example, if a transductive model is given a set of labeled and unlabeled emails, it may use the structure of both to predict labels for the unlabeled emails in that same set.
- Common algorithms include **Transductive SVMs**, **k-Nearest Neighbors** and **graph-based models**.

SUPERVISED LEARNING/ UNSUPERVISED

Inductive

- Generalizes to unseen data using labeled training data. Eg. Neural network in image classification, decision trees.
- Finds patterns that generalizes to new unseen data. K-means clustering, PCA

Transductive

- Predictions are tailored for a specific test set, often in semi supervised learning scenarios. Eg. KNN, transductive SVM
- Leverages data structure to make predictions specific to the same data set. Eg. Graph based learning

SEMI SUPERVISED LEARNING

Inductive

- Inductive models are best for situations where the goal is to generalize to unseen data, such as spam detection or recommendation systems.
- These models are widely used and versatile.

Transductive

- Transductive models are useful when you know the production set in advance and need tailored predictions, as in semi-supervised learning or graph-based applications.
- Transductive learning performs better than inductive learning on the same data. Because it capture extra data, because of the prediction triage.

WHEN CAN SEMI-SUPERVISED LEARNING WORK? ASSUMPTIONS

- H1: **Smoothness Assumptions**: If two data samples are close in a high-density region of the feature space, their labels should be the same or very similar.
- If two points x1, x2 in a high density region are close, then so should be the corresponding outputs y1, y2. By transitivity, this assumption implies that if two points are linked by a path of high density (e.g., if they belong to the same cluster), then their outputs are likely to be close. If, on the other hand, if they are separated by a low density region, then their outputs need not be close.

WHEN CAN SEMI-SUPERVISED LEARNING WORK? ASSUMPTIONS

2.1.1 Smoothness assumption

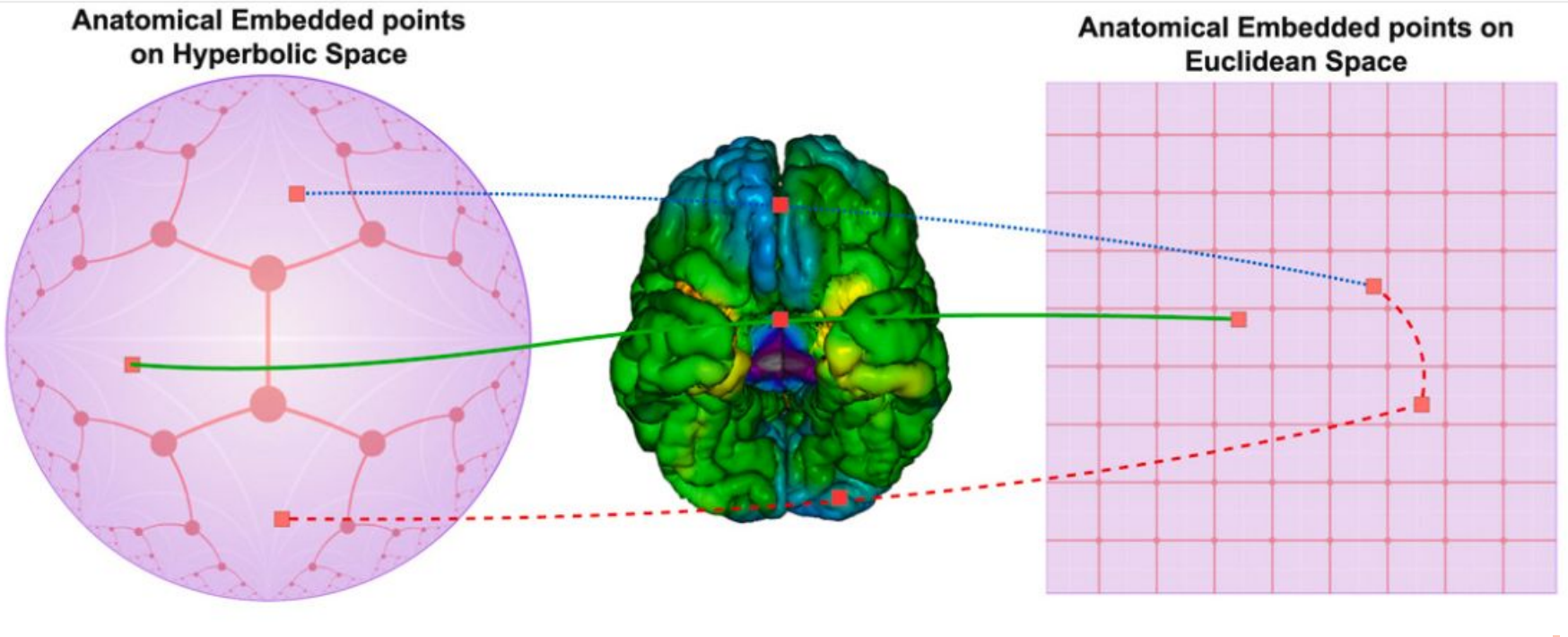
The smoothness assumption states that, for two input points $x, x' \in \mathcal{X}$ that are close by in the input space, the corresponding labels y, y' should be the same. This assumption is also commonly used in supervised learning, but has an extended benefit in the semi-supervised context: the smoothness assumption can be applied transitively to unlabelled data. For example, assume that a labelled data point $x_1 \in X_L$ and two unlabelled data points $x_2, x_3 \in X_U$ exist, such that x_1 is close to x_2 and x_2 is close to x_3, but x_1 is not close to x_3. Then, because of the smoothness assumption, we can still expect x_3 to have the same label as x_1, since proximity—and thereby the label—is transitively propagated through x_2.

WHEN CAN SEMI-SUPERVISED LEARNING WORK? ASSUMPTIONS

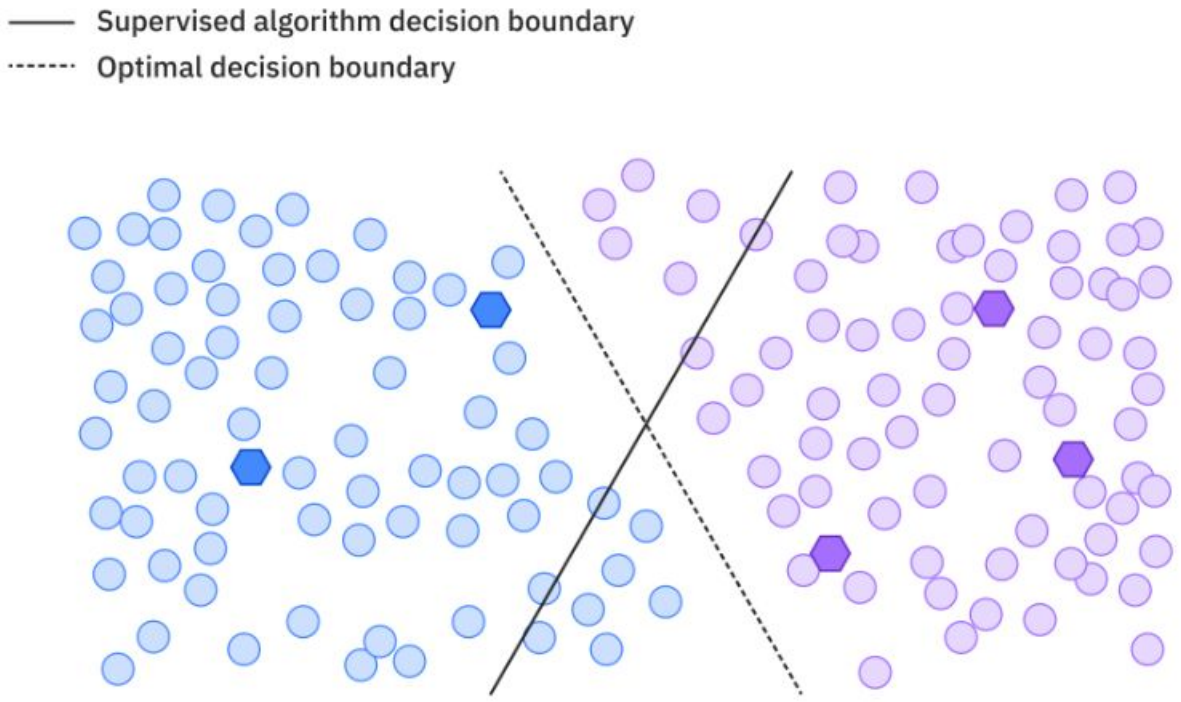
- **H2 Cluster Assumption.** If points are in the same cluster, they are likely to be of the same class. We do not observe objects of two distinct classes in the same cluster.
- **H3: Low Density Separation.** The decision boundary should lie in a low density region.

WHEN CAN SEMI-SUPERVISED LEARNING WORK? ASSUMPTIONS

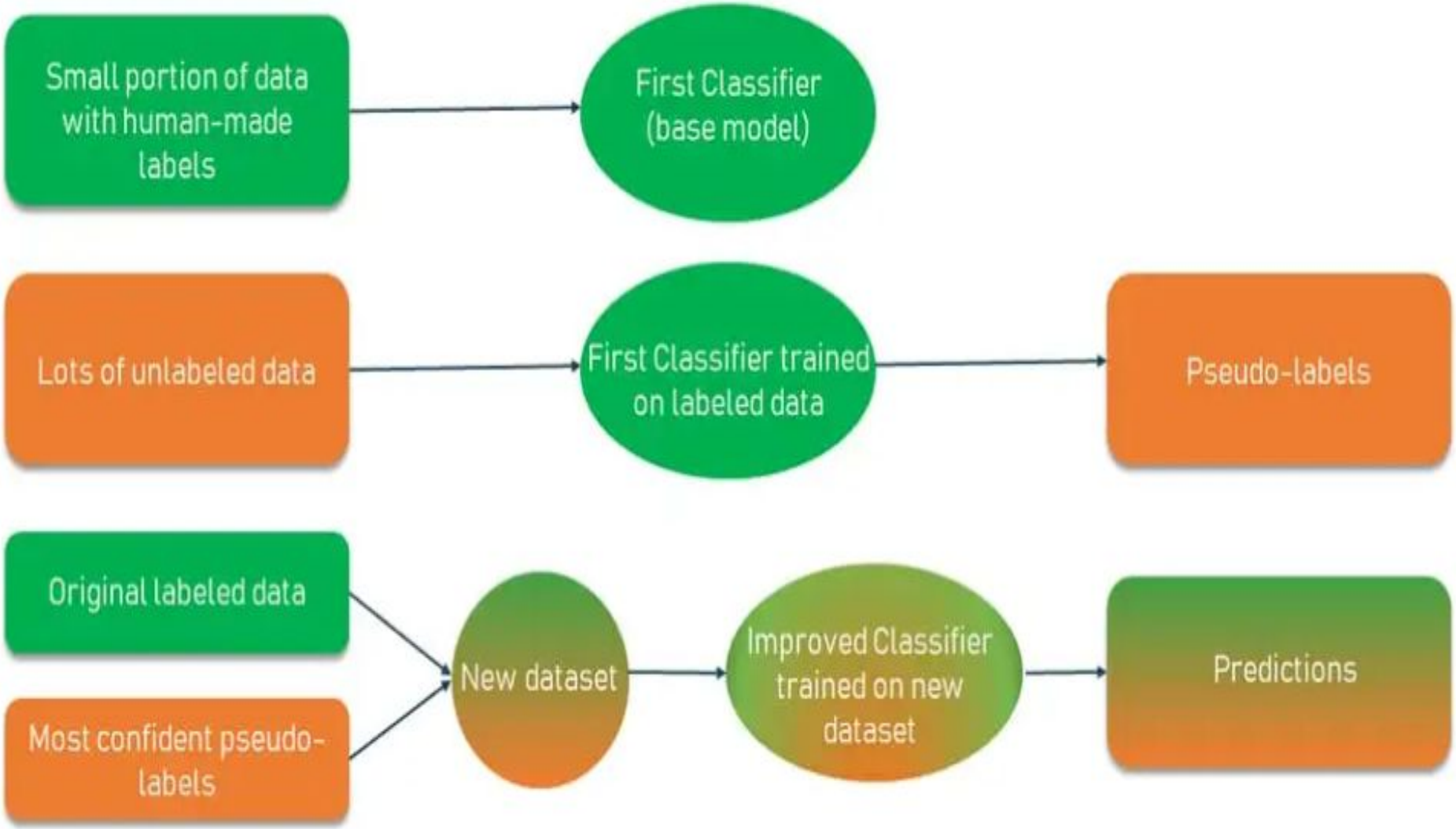
- **H4: Manifold Assumption.** The (high-dimensional) data lie (roughly) on a low dimensional manifold.
- For an intuitive example, consider a piece of paper crumpled up into a ball. The location of any points on the spherical surface can only mapped with three-dimensional x,y,z coordinates. But if that crumpled up ball is now flattened back into a sheet of paper, those same points can now be mapped with two dimensional x, y coordinates. This is called dimensionality reduction, and it can be achieved mathematically using methods like autoencoders or convolutions.


Anatomical Embedded points
on Hyperbolic Space
Anatomical Embedded points on
Euclidean Space

The diagram illustrates how the smoothness and low-density assumptions can inform a far more intuitive decision boundary than would be possible with supervised methods that can only consider the (very few) labeled data points.



SEMI-SUPERVISED SELF-TRAINING METHOD



SEMI-SUPERVISED LEARNING TECHNIQUE: SELF-TRAINING

You pick a small amount of labeled data, and you use this dataset to train a base model with the help of ordinary supervised methods.

Then you apply the process known as *pseudo-labeling* — when you take the partially trained model and use it to make predictions for the rest of the database which is yet unlabeled.

The labels generated thereafter are called *pseudo* as they are produced based on the originally labeled data that has limitations (may be resulting in bias).

From this point, you take the most confident predictions made with your model. If any of the pseudo-labels exceed this confidence level, you add them into the labeled dataset and create a new, combined input to train an improved model.

The process can go through several iterations (10) with more and more pseudo-labels being added every time. Provided the data is suitable for the process, the performance of the model will keep increasing at each iteration.

SEMI-SUPERVISED LEARNING TECHNIQUE: CO-TRAINING

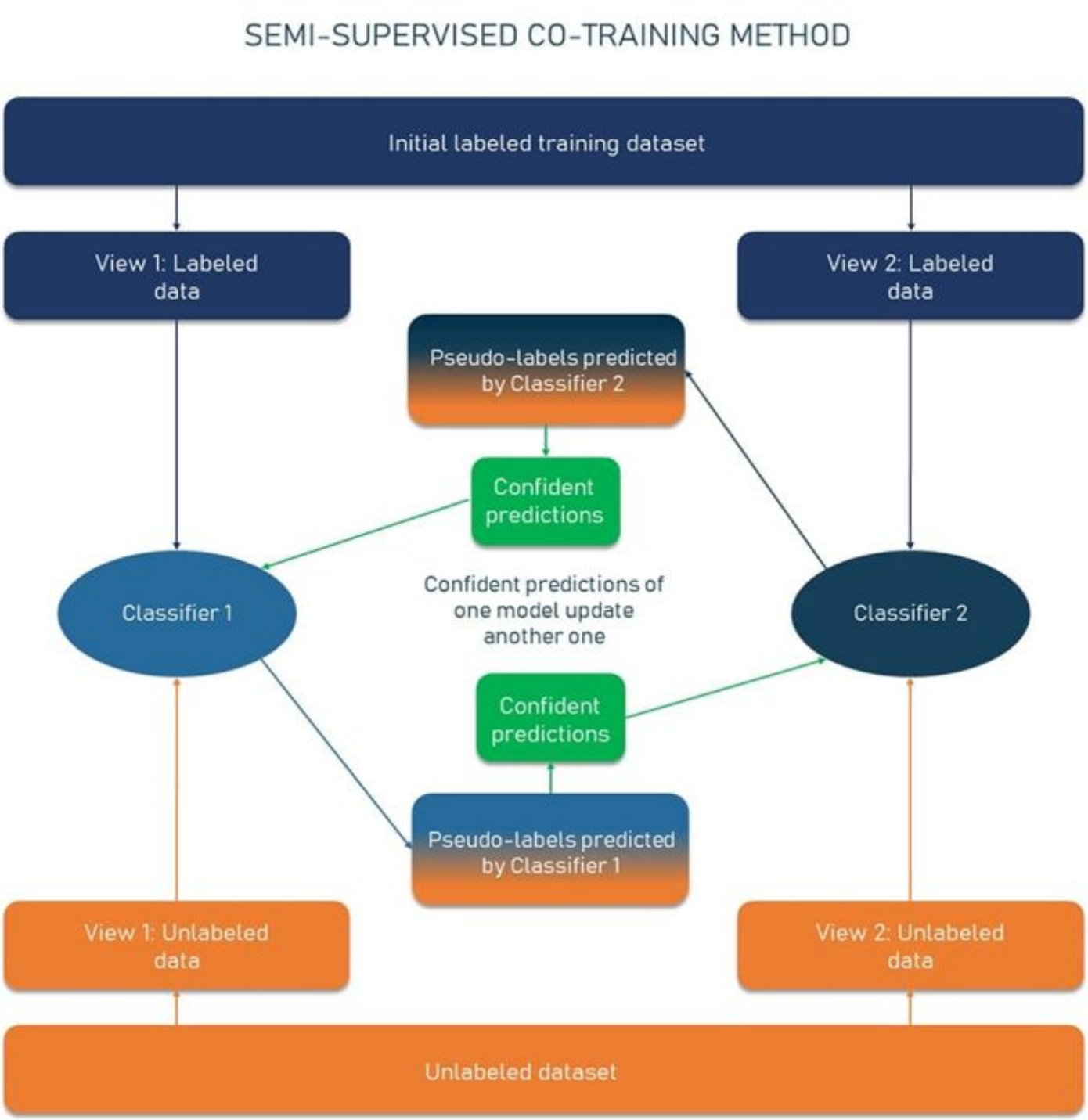
Co-training

Derived from the self-training approach and being its improved version, co-training is another semi-supervised learning technique **used when only a small portion of labeled data is available**.

Unlike the typical process, co-training **trains two individual classifiers based on two *views* of data.**

Co-training capitalizes on the concept of learning from multiple, potentially complementary, views of the data. The fundamental idea is to **train multiple models, each using a different subset of features or representations of the dataset.** The views provide additional information about each instance, meaning they are independent given the class. Also, each view is sufficient — the class of sample data can be accurately predicted from each set of features alone.

These models then collaborate by providing predictions for the unlabeled data, and instances on which the models agree are confidently added to the labeled dataset. This iterative process enhances the robustness and generalization ability of the models.



n be successfully
tent classification
otion of each web
ed into two views:
curring on that
er with anchor words
to it.

SEMI-SUPERVISED LEARNING(SSL) TECHNIQUE: GRAPH-BASED LABEL PROPAGATION

- A popular way to run SSL is to represent labeled and unlabeled data in the form of graphs and then apply a label propagation algorithm. It spreads human-made annotations through the whole data network.

- The Label Propagation algorithm (LPA) is a fast algorithm for finding communities in a graph. It detects these communities using network structure alone as its guide and doesn't require a predefined objective function or prior information about the communities.
- You have the option of assigning preliminary labels to narrow down the range of generated solutions. This means you can use it as a semi-supervised way of finding communities where you handpick some initial communities.

SEMI-SUPERVISED LEARNING(SSL) TECHNIQUE: GRAPH-BASED LABEL PROPAGATION

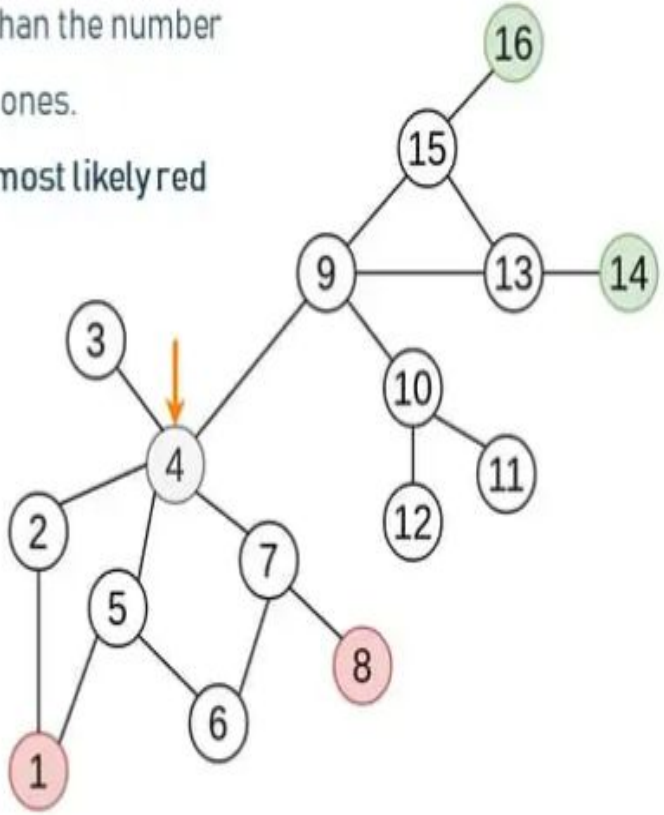
- It is a Near linear time algorithm to detect community structures in large-scale networks. It works by propagating labels throughout the network and forming communities based on this process of label propagation.
- The intuition behind the algorithm is that a single label can quickly become dominant in a densely connected group of nodes, but it will have trouble crossing a sparsely connected region. Labels will get trapped inside a densely connected group of nodes, and those nodes that end up with the same label when the algorithm finishes are considered part of the same community.

SEMI-SUPERVISED LEARNING(SSL) TECHNIQUE: GRAPH-BASED LABEL PROPAGATION

GRAPH-BASED LABEL PROPAGATION

The number of walks to red nodes is greater than the number of walks to green ones.

Assumption: **4 is most likely red**



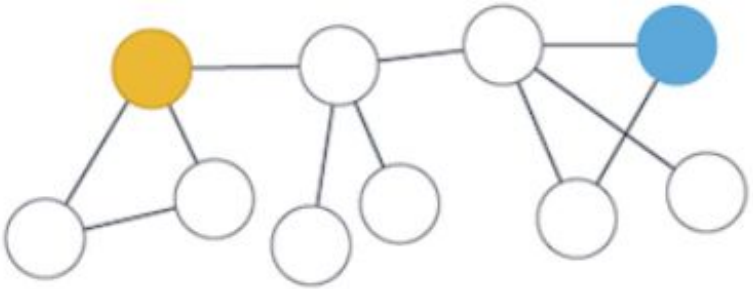
Walks that end up in green nodes

1. $4 \rightarrow 9 \rightarrow 15 \rightarrow 16$
2. $4 \rightarrow 9 \rightarrow 13 \rightarrow 14$
3. $4 \rightarrow 9 \rightarrow 13 \rightarrow 15 \rightarrow 16$
4. $4 \rightarrow 9 \rightarrow 15 \rightarrow 13 \rightarrow 14$

Walks that end up in red nodes

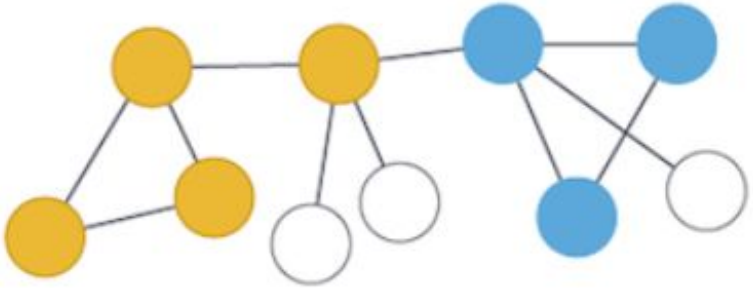
1. $4 \rightarrow 7 \rightarrow 8$
2. $4 \rightarrow 7 \rightarrow 6 \rightarrow 5 \rightarrow 1$
3. $4 \rightarrow 5 \rightarrow 1$
4. $4 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 8$
5. $4 \rightarrow 2 \rightarrow 1$

Initial State



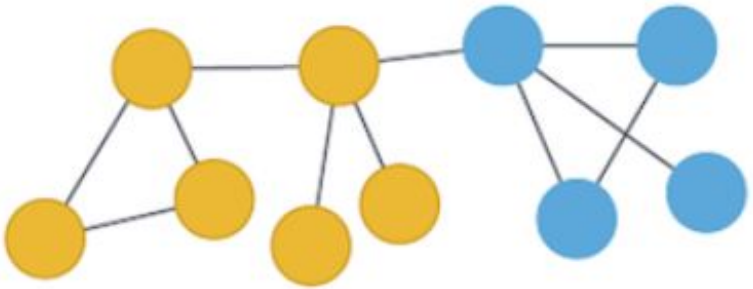
Some nodes have labels

Pass 1



More labels added

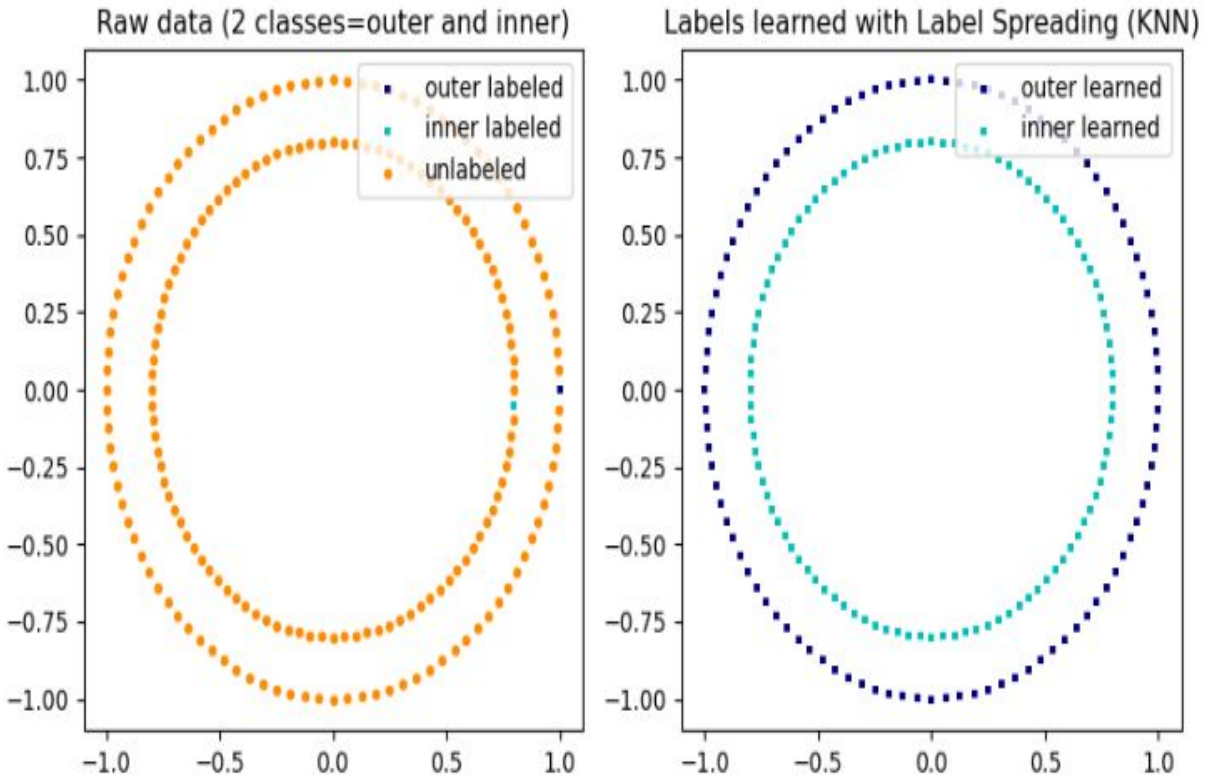
Pass 2



Iterations continue until there is convergence on a solution, a set solution range, or a set number of iterations.

Label Propagation Algorithm

Semi-supervised learning(SSL) technique: graph-based label propagation



An illustration of label-propagation: the structure of unlabeled observations is consistent with the class structure, and thus the class label can be propagated to the unlabeled observations of the training set.

SEMI-SUPERVISED LEARNING(SSL) TECHNIQUE: GRAPH-BASED LABEL PROPAGATION

- The algorithm works as follows:
  - Every node is initialized with a unique label (an identifier).
  - These labels propagate through the network.
  - At every iteration of propagation, each node updates its label to the one that the maximum number of its neighbors belongs to. Ties are broken uniformly and randomly.
  - LPA reaches convergence when each node has the majority label of its neighbors. As labels propagate, densely connected groups of nodes quickly reach a consensus on a unique label. At the end of the propagation, only a few labels will remain – most will have disappeared. Nodes that have the same label at convergence are said to belong to the same community.

LABEL PROPAGATION- APPLICATIONS

- Label Propagation has been used to assign polarity of tweets, as a part of semantic analysis that uses seed labels from a classifier trained to detect positive and negative emoticons in combination with the Twitter follower graph.
- Label Propagation has been used to estimate potentially dangerous combinations of drugs to co-prescribe to a patient, based on the chemical similarity and side effect profiles.
- Label Propagation has been used to infer features of utterances in a dialogue for a machine learning model to track user intention with the help of a Wikidata knowledge graph of concepts and their relations.

SEMI-SUPERVISED LEARNING TECHNIQUE: APPLICATIONS

Speech recognition: annotating the input audio data with corresponding text transcriptions

Labeling audio is a very resource- and time-intensive task, so semi-supervised learning can be used to overcome the challenges and provide better performance. Facebook (now Meta) has successfully applied semi-supervised learning (namely the self-training method) to its speech recognition models and improved them. They started off with the base model that was trained with 100 hours of human-annotated audio data. Then 500 hours of unlabeled speech data was added and self-training was used to increase the performance of the models. As far as the results, the word error rate (WER) decreased by 33.9 percent, which is a significant improvement.

SEMI-SUPERVISED LEARNING TECHNIQUE: APPLICATIONS

Web content classification: Web classification is the method of classifying a website main content or topic according to a set of defined categories.

With billions of websites presenting all sorts of content out there, classification would take a huge team of human resources to organize information on web pages by adding corresponding labels. The variations of semi-supervised learning are used to annotate web content and classify it accordingly to improve user experience. Many search engines, including Google, apply SSL to their ranking component to better understand human language and the relevance of candidate search results to queries. With SSL, Google Search finds content that is most relevant to a particular user query.

SEMI-SUPERVISED LEARNING TECHNIQUE: APPLICATIONS

Text document classification: takes raw text like documents and classifies them as spam or not spam, Netflix(Comedy, Drama)

Another example of when semi-supervised learning can be used successfully is in the building of a text document classifier. Here, the method is effective because it is really difficult for human annotators to read through multiple word-heavy texts to assign a basic label, like a type or genre.

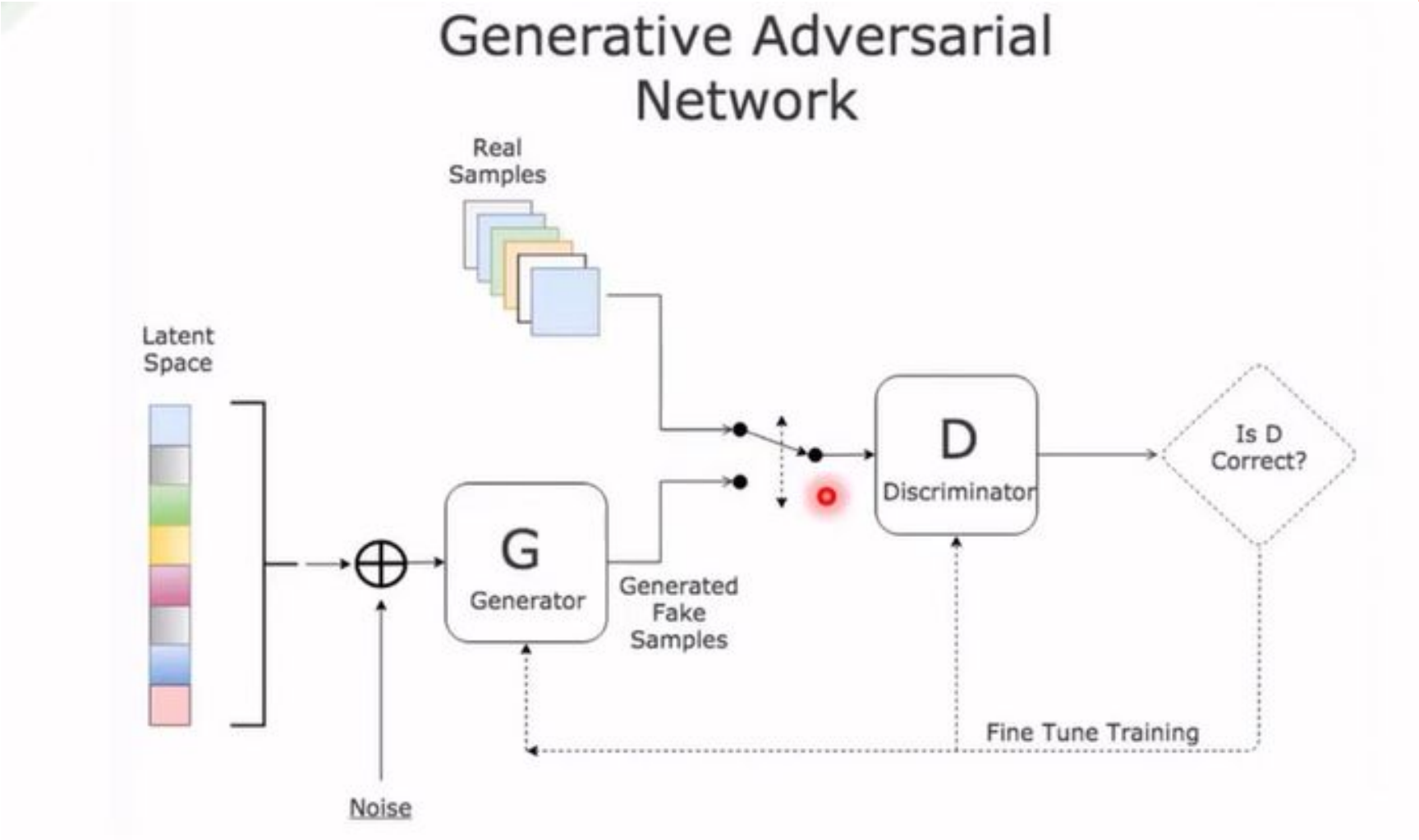
For example, a classifier can be built on top of deep learning neural networks like LSTM (long short-term memory) networks that are capable of finding long-term dependencies in data and retraining past information over time. Usually, training a neural net requires lots of data with and without labels. A semi-supervised learning framework works just fine as you can train a base LSTM model on a few text examples with hand-labeled most relevant words and then apply it to a bigger number of unlabeled samples.

GENERATED ADVERSARIAL NETWORKS

- Why? There are many times where we just do not have enough data to create a model . GANs can learn about your data and learn to synthesize or generate never-before-seen data to augment your dataset.
- It is an approach to semi supervised learning when some of your data is labeled while the rest are not. We may have the labeled dataset set but it may not be enough to train a model. In this case, we can generate unlabeled data using GAN's.
- Furthermore, the generated data could also be used as is (image/audio synthesis).

GENERATED ADVERSARIAL NETWORKS

- Why it is Adversarial? It is a competition between two models that play against each other.
- Model:A Generator and a Discriminator.
- Generator(Counterfeiter): Replicates real input data to produce fake data.
- Discriminator(cop): Distinguishes real data from fake data (to catch the counterfeiter).
- The Loss of the Discriminator is used as the objective function for the Generator in the next round.


Generative Adversarial Network
Real Samples
Latent Space
G
Generator
Generated Fake Samples
D
Discriminator
Is D Correct?
Noise
Fine Tune Training

HOW DOES A GAN WORK?

- GAN train by having two networks the Generator (G) and the Discriminator (D) compete and improve together. Here's the step-by-step process
- **1. Generator's First Move**
- The generator starts with a random noise vector like random numbers. It uses this noise as a starting point to create a fake data sample such as a generated image. The generator’s internal layers transform this noise into something that looks like real data.
- **2. Discriminator's Turn**
- The discriminator receives two types of data:Real samples from the actual training dataset. Fake samples created by the generator.

HOW DOES A GAN WORK?

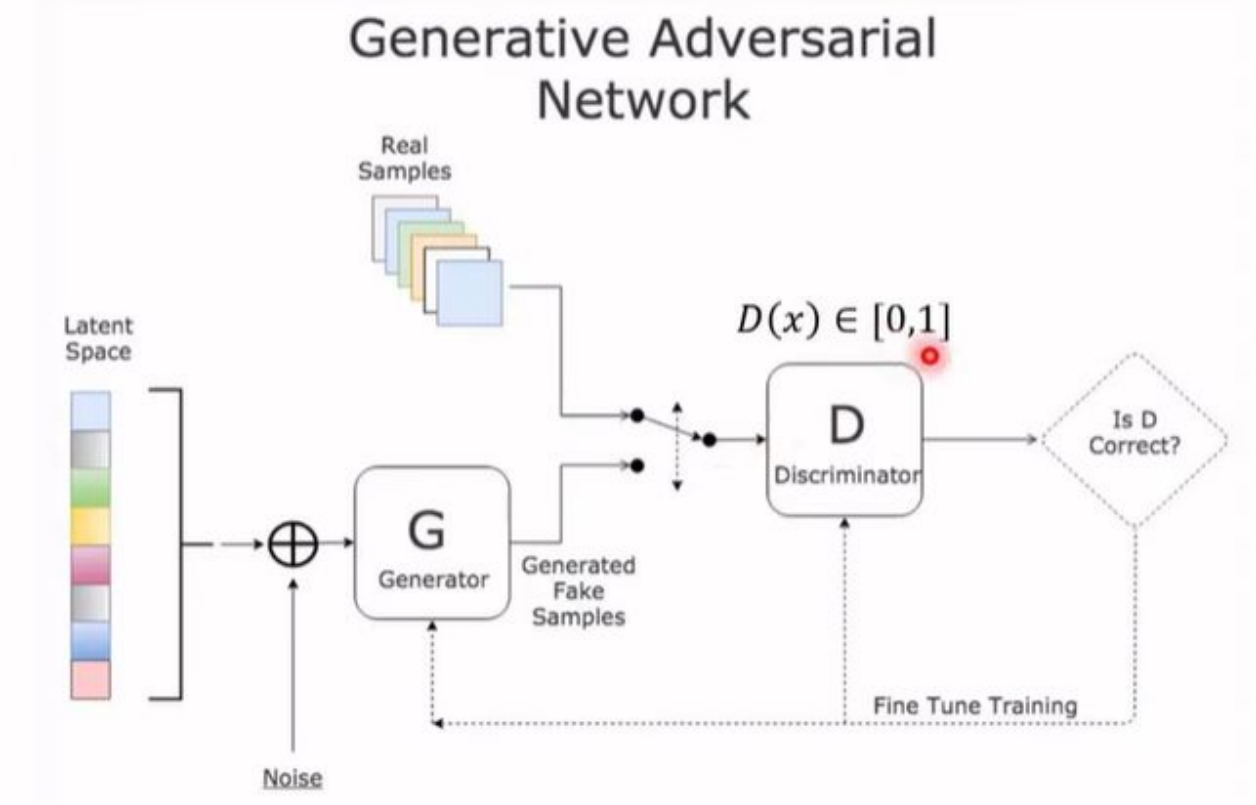
- D's job is to analyze each input and find whether it's real data or something G cooked up. It outputs a probability score between 0 and 1. A score of 1 shows the data is likely real and 0 suggests it's fake.
- **3. Adversarial Learning**
- If the discriminator correctly classifies real and fake data it gets better at its job. If the generator fools the discriminator by creating realistic fake data, it receives a positive update and the discriminator is penalized for making a wrong decision.
- **4. Generator's Improvement**
- Each time the discriminator mistakes fake data for real, the generator learns from this success.Through many iterations, the generator improves and creates more convincing fake samples.

HOW DOES A GAN WORK?

- **5. Discriminator's Adaptation**
- The discriminator also learns continuously by updating itself to better spot fake data.This constant back-and-forth makes both networks stronger over time.
- **6. Training Progression**
- As training continues, the generator becomes highly proficient at producing realistic data. Eventually the discriminator struggles to distinguish real from fake shows that the GAN has reached a well-trained state.
- At this point, the generator can produce high-quality synthetic data that can be used for different applications.

OBJECTIVE FUNCTION FOR THE DISCRIMINATOR

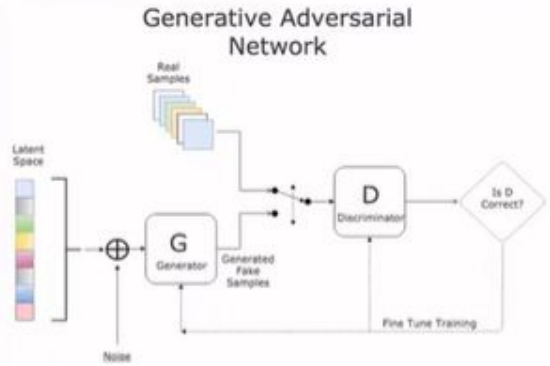
- The discriminator outputs some real number between 0 and 1 which is actually a probability.
- If the output is 1 it means that this instance is real and 0 means it is fake.



OBJECTIVE FUNCTION FOR THE DISCRIMINATOR

- We want to maximize the objective function of the Discriminator(D). There are two components for the discriminator.
- E stands for the expectation value. ~ sign represents "follow the distribution/ is distributed as". x~pdata(x) means instance taken from real data distribution. z~pz(z) means instance taken from noisy distribution of some latent space.
- D(x) represents predicted output of the Discriminator
- Since we do not know how the samples are fed into the discriminator, we represent them as expectation E (average) rather than sums.
- First term represents expectation of instances taken for the real data and the second term represents expectations of the instances taken from noisy distribution

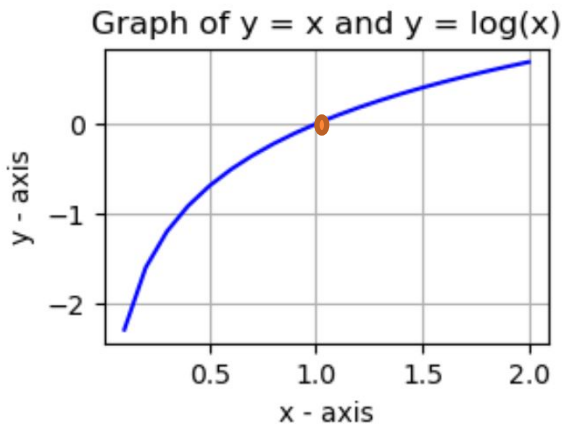
MinMax Loss



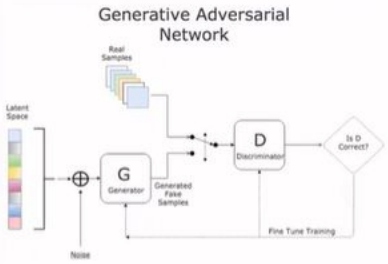
$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

OBJECTIVE FUNCTION FOR THE DISCRIMINATOR

- First part of the term is concerned with the real data. So x comes from the real data. For the first term, we want the Discriminator to output 1 as it is a real sample and not output 0.
- For the purpose of scaling, we use log function. If we plot the log function of D(x), we get a graph as shown below. We are interested for values between 0 and 1 only. Maximum value of log D(x) is when D(x) is 1.
- Better the discriminator, larger the first term. So we can say that if we want to optimize the discriminator, we have to maximize the first term. So we have max D in the objective function.



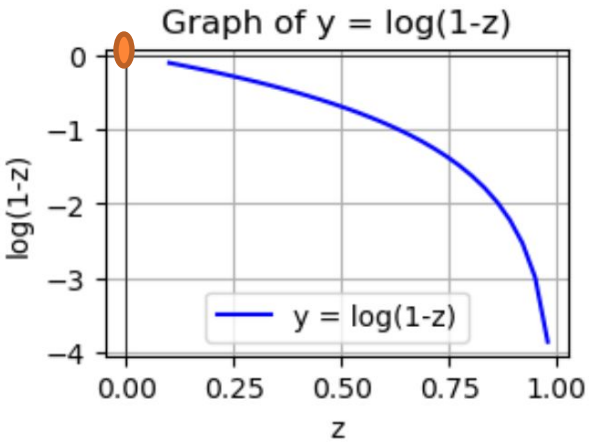
MinMax Loss



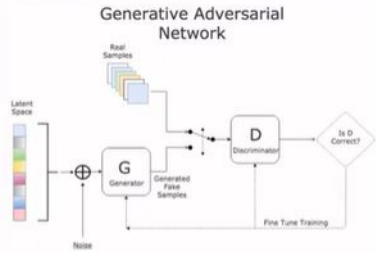
$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

OBJECTIVE FUNCTION FOR THE DISCRIMINATOR

- In the second term, we have the instances generated by the generator. We want the discriminator to be able to tell that they are fake i.e., we want the output of D(G(z))=0 . Therefore, the second term contributes to the Loss function when it is 1.
- Better the discriminator, smaller the value of D(G(z)). It should minimize this part.
- But the first part says maximize. So that is why we use 1 in the second part. Smaller the value of D(G(z)), greater the value of 1-D(G(z)). Better the Discriminator, better the second part.
- If we plot the log function of 1-D(z), we get a graph as shown below. Maximum value of log(1-D(G(z))) is when D(G(z))=0



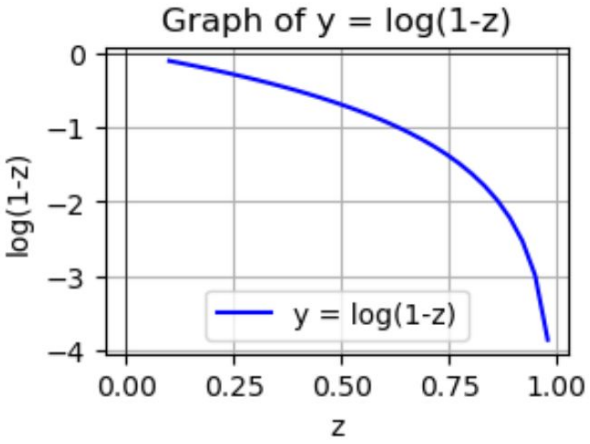
MinMax Loss



$$\max_D V(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

OBJECTIVE FUNCTION OF THE GENERATOR

- In the first part, the Generator does not appear.
- In the second part, if the Generator is good, the Discriminator would have hard time to say that the data is fake, so it will output values close to 1, thinking it is real. Closer the D(G(z)) to 1 , smaller the value is. We then get log(0) which is minus infinity.
- Better the generator, the more negative the second part.
- That is why we want to minimize this term regarding the generator.



$$\min_G V(G) = \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

MIN MAX GANS LOSS

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$