Semi-Supervised Learning and Look-Alike Models

(https://github.com/learn-co-curriculum/dsc-semi-supervised-learning-and-look-alike-models) (https://github.com/learn-co-curriculum/dsc-semi-supervised-learning-and-look-alike-models/issues/new)

Introduction

In this lesson, we'll learn about some unsupervised learning techniques we can use to supplement our supervised learning techniques.

Objectives

You will be able to:

- Identify appropriate use cases for semi-supervised learning
- Identify appropriate use cases for look-alike models

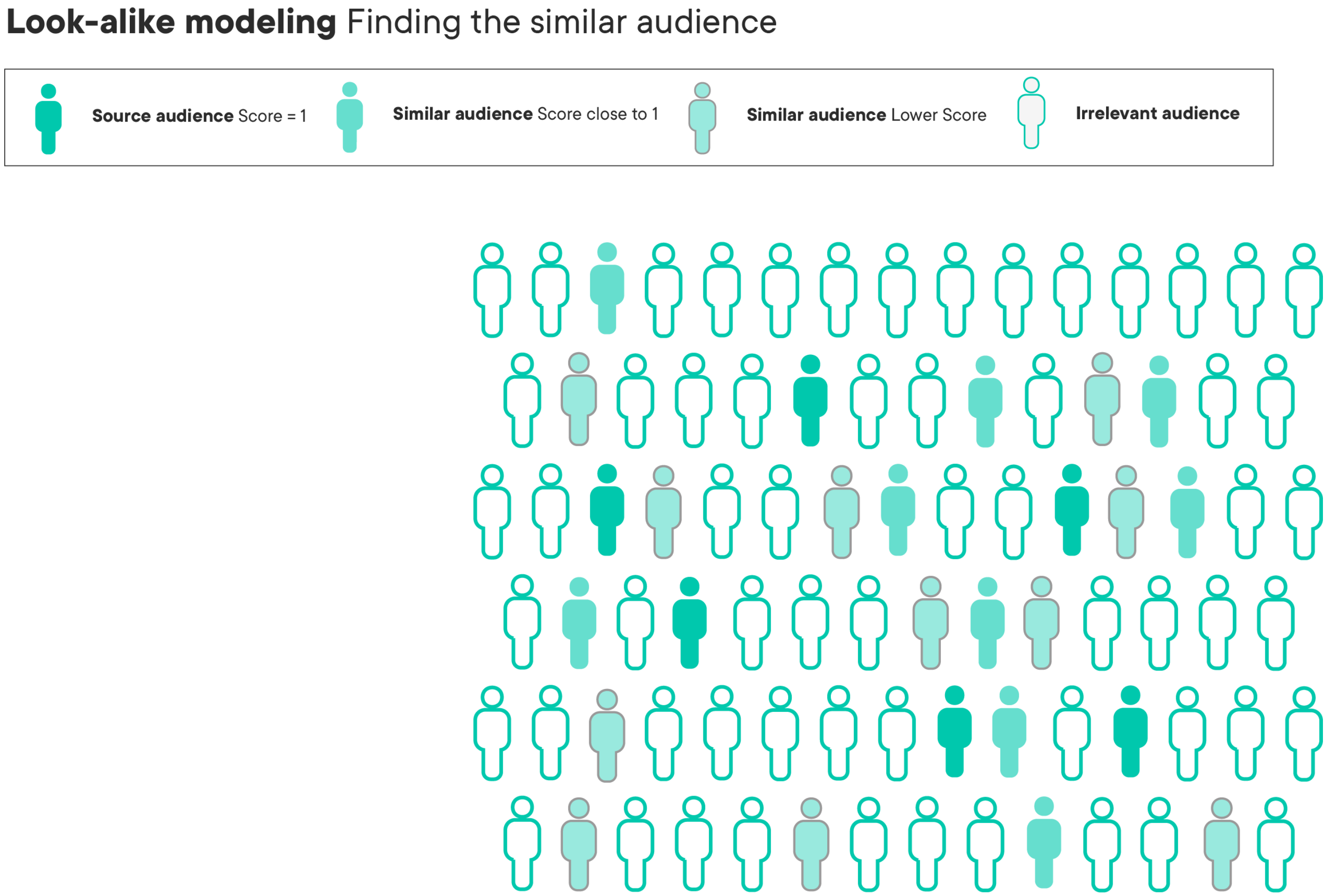
Combining Supervised and Unsupervised Learning

For the majority of this section, we've focused exclusively on popular unsupervised learning techniques and their most common use cases. However, in the real world, there are also plenty of examples where it works to our advantage to bring supervised and unsupervised learning algorithms together to supplement each other. In this lesson, we'll look at two common areas combining supervised and unsupervised learning algorithms that allow us to be more effective than just using them on their own.

Use Case 1: Look-Alike Models

As we've learned when working with clustering algorithms, one of their most common use cases is for market segmentation. A more advanced, but similar use case is to then use these market segments to create ***look-alike models*** to help us identify more customers or market segments that we can plausibly assume are equally valuable, due to their similarity with valuable customers or market segments we've already identified.

Take a look at the following infographic that provides a visual representation of look-alike modeling:


Look-alike modeling Finding the similar audience
Source audience Score = 1
Similar audience Score close to 1
Similar audience Lower Score
Irrelevant audience

In the example above, the dark blue smiley faces represent customer segments that we already know are valuable. These are customers that we have identified in our data, and know for a fact have been good for us. Under normal circumstances, this would mean that we can divide our customers (or, more often, potential customers) into two groups: the group we know is valuable, and everyone else, who are all unknown to us.

This is where *look-alike modeling* comes in. A look-alike model uses a distance metric of our choice to rate the similarity of each customer in our group of unknowns to customers in our known, valuable group. For customers that look extremely similar to customers in own known valuable group, we can assume with a very high likelihood that these customers will also be valuable, and should direct resources at capturing them! We'll likely also see customers that are only somewhat similar to our valuable group, which tells us that they *could possibly be valuable*, but we aren't sure. And finally, customers that look nothing like our known valuable customers segment, should probably be left alone.

If this sounds suspiciously like clustering to you, you are absolutely correct! Although this could also be framed as a classification or regression problem, it's quite common to see clustering used to help determine similarity. After all, if we want to build a supervised learning model to predict if an unknown customer looks like our known valuable customers, then we need plenty of labeled examples, and we don't always have that luxury!

In the real-world, using look-alike models to find other customers that could potentially be valuable to us is often referred to as ***prospecting***. Viewed in terms of the infographic above, we would choose direct resources to market to the customers that look like our valuable customers to increase our ***top-of-funnel***, meaning that we are trying to increase the number of potential customers that haven't shown interest in our product or company yet but are likely to, due to their similarity to customers that already have.

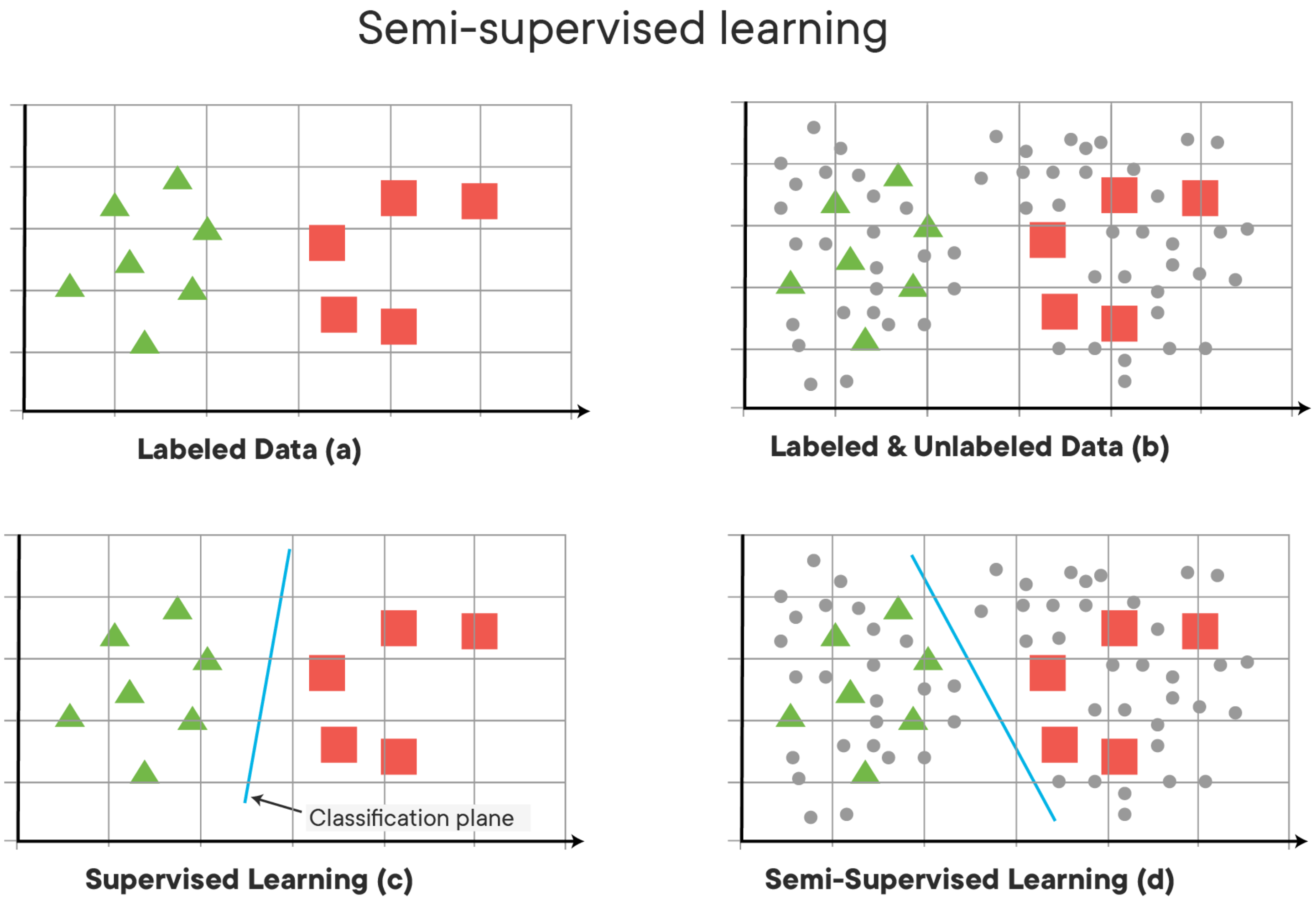
Use Case 2: Semi-Supervised Learning

The second use case we'll talk about combines supervised and unsupervised learning to allow us access to more (pseudo) labeled data so that we can better train our supervised learning models. This technique is called ***semi-supervised learning***. You may also hear it commonly referred to as ***weakly supervised learning***, but it means the same thing.

Picture the following scenario:

We are trying to build a supervised learning model, and we have 100,000 observations in our dataset. However, labels are exceedingly expensive, so only 5,000 of these 100,000 observations are labeled. In traditional supervised learning, this means that in a practical sense, we really only have a dataset of 5,000 observations, because we can't do anything with the 95,000 unlabeled examples -- or can we?

The main idea behind *semi-supervised learning* is to generate ***pseudo-labels*** that are possibly correct (at least better than random chance). To do this, we don't usually use clustering algorithms -- instead, we use our supervised learning algorithms in an unsupervised way.



Supervised learning typically follows a set pattern:

1. ***Train your model on your labeled training data***. In the case of our example above, we would build the best model possible with our tiny dataset of 5,000 labeled examples.

2. ***Use your trained model to generate pseudo-labels for your unlabeled data***. This means having our trained model make predictions on our 95,000 unlabeled examples. Since our trained model does better than random chance, this means that our generated pseudo-labels will be at least somewhat more correct than random chance. We can even put a number to this, by looking at the performance our trained model had on the test set. For example, if our trained model had an accuracy of ~70%, then we can assume that ~70% of the pseudo-labels will be correct, ~30% will be incorrect.

3. ***Combine your labeled data and your pseudo-labeled data into a single, new dataset.***. This means that we concatenate all our labeled data of 5,000 examples with the 95,000 pseudo-labeled examples.

4. ***Retrain your model on the new dataset***. Although some of the pseudo-labeled data will certainly be wrong, it's likely that the amount that is correct will be more useful, and the signal that these correctly pseudo-labeled examples provide will outweigh the incorrectly labeled ones, thereby resulting in better overall model performance.

Benefits and Drawbacks of Semi-Supervised Learning

If semi-supervised learning sounds a bit risky to you, you're not wrong. When done correctly, semi-supervised learning can increase overall model performance by opening up access to much more data than we would have access to, and more data almost always results in better performance, but without the exorbitant costs of paying to have humans generate labels for the data needed.

However, there are definitely some problems that can arise from using a semi-supervised learning approach, if we're not careful and thoughtful throughout.

Feedback Loops and Self-Fulfilling Prophecies

Semi-supervised learning tends to work fairly well in many use cases and has become quite a popular technique in the field of Deep Learning, which requires massive amounts of labeled data that is often very expensive to obtain. But what happens when our dataset is extremely noisy to begin with? In that case, our incorrect pseudo-labels may skew the model by introducing more "noise" than "signal". This is partially because we can end up in a feedback loop of sorts. Think about an example where the model has generated an incorrect pseudo-label. If a model trained only on the real data with no pseudo-labels got this example wrong, then what happens when you train the model on the same example, but this time provide a pseudo-label that "confirms" this incorrect belief? When done correctly, we can hope that the signal provided by all the correctly pseudo-labeled examples will generalize to help the model correct its mistakes on the ones it got wrong. However, if the dataset is noisy, or the original model wasn't that good to begin with (or both), then it can be quite likely that we are introducing even more incorrect information than correct information, moving the model in the wrong direction.

So how do we make sure that we're not making these mistakes when using a semi-supervised approach? ***Use a holdout set!*** You should definitely have a test set that the model has never seen before to check the performance of your semi-supervised model. Obviously, make sure that your test set only contains actual, ground-truth labeled examples, no pseudo-labels allowed! Also, the noisier your dataset or more complicated your problem, the more likely you are to run into trouble with semi-supervised learning. When possible, try to structure your tasks as binary classification tasks, rather than multi-categorical, and make sure that your dataset is as clean as possible before attempting semi-supervised learning. Although it seems risky, there's a reason companies that are heavy into deep learning and AI research such as Google, Microsoft, and Facebook make heavy use of semi-supervised learning -- when done correctly, it works wonders, without costing an arm and a leg to pay for labeling!

Summary

In this lesson, we learned about two popular methodologies for using unsupervised learning in applied, focused ways to help companies generate more revenue, get more customers, or increase model performance without paying for more labeled training data!