Lecture 21: Unsupervised Learning

Instructor: Aditya Bhaskara *Scribe: Jake Pitkin*

CS 5966/6966: Theory of Machine Learning

March 29th, 2017

Abstract

In this lecture, we introduce the concept of unsupervised learning by looking at a commonly used and broad statistical technique called clustering. We will look at using a mixture of Gaussian distributions to cluster data points as well as introduce the k-means optimization problem.

1 Introduction

Similar to other forms of learning we have studied, the goal of unsupervised learning is to discover patterns in a set of data. It is unique from supervised learning or reinforcement learning in two main ways. First, data points do not come with labels and the goal is rarely to just classify the data. Second, there is no "ground truth" or evaluation of accuracy as the data points are unlabeled. Unsupervised learning is desirable as annotated data is rare and expensive to produce. In contrast, unlabeled data is typically plentiful and much easier to obtain in large quantities.

2 Motivating Examples

As motivation for unsupervised learning techniques, we will briefly look at three example applications.

Movie Recommendations

Consider a service such as Netflix that is interested in making movie recommendations. They could keep a matrix of *movies* $\times$ *users* where the entries in the matrix are say a rating on a scale from $1-5$. As there exists a large number of *movies* and *users* this will be a sparse matrix.

$$\begin{bmatrix} & user_1 & \dots & \dots & user_n \\ movie_1 & & 3 & & \\ \vdots & & & & \\ \vdots & & & & 4 \\ movie_m & 2 & & & \end{bmatrix}$$

Now say we want to determine the likelihood that $user_i$ would enjoy $movie_j$. By using the data points we do have, we could look at the attributes of those movies and users in an attempt to find a hidden structure in the data. This hidden structure could then be used to guide our recommendation process.

Cocktail Party Problem

Imagine being at a cocktail party where the room is noisy from multiple separate conversations occurring simultaneously. There is a microphone in the room that is picking up the superposition of a whole bunch of audio signals. After collecting this audio data, the goal is to break up the signals into "components" in an attempt to isolate each of the individual conversations.

Sparse Coding

In the field of neuroscience, there is work being done to understand how our brains encode sensory input of images. Each image is encoded as a sparse combination of a few basic patterns. Given visual signals, the brain finds a small set of these patterns such that the images we see can be formed as a sparse combination of these basic patterns. This allows the images to be understood by the brain using a relatively small number of neurons.

3 Unsupervised Learning

In unsupervised learning, we assume the data points are generated by some random process with a "few parameters" that are unknown. This assumption is called a generative assumption as we are assuming the data is produced by a unknown generative model. We have seen generative models before in supervised learning and they are not unique to unsupervised learning.

The goal of unsupervised learning is to identify these hidden parameters. These parameters could be central points and how these points relate to nearby points. If we identify these parameters, we could cluster the data into groups. We are operating under the assumption that the data has some underlying explanation. Additionally, there is no "ideal" model and we can overfit the data or underfit the data depending on the model complexity.

4 Gaussian Distributions

To keep our examples simple we will consider a set of points $\mathcal{X} \in \mathbb{R}$ as our data points.

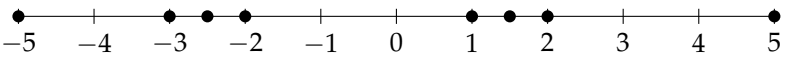

Figure 1: An example set of data points.

As you could imagine, we could find a high degree polynomial to closely fit the data points. This is generally a bad idea as this complicated model overfits the given data and it is likely to generalize poorly on future data points.

Mixture of Gaussian Distributions

Instead, we could express the distribution of points as the superposition of a mixture of Gaussian distributions with weights. Recall the Gaussian distribution is a probability distribution defined by a mean μ and a variance σ^2.

$$(1) \quad f(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Given k Gaussian distributions, our model parameters would be k weights w_i such that $\sum_{i=1}^{k} w_i = 1$ and the means and variances that define the Gaussian distributions $\{ \mu_i, \sigma_i^2 \}_{i=1}^{k}$.

$$(2) \quad \textit{data distribution} \approx w_1 GD_1 + w_2 GD_2 + ... + w_k GD_k$$

A bonus of this model is it gives us a way to generate a potentially infinite amount of synthetic data as we have a distribution. This brings up the question of does this distribution truly approximate the real distribution (assuming the data has a distribution)? An important observation is we have a discrete distribution as we have a finite number of data points. There is no distribution mass between the sampled points (in fig. 1 at -4 for example) and we don't have a true continuous distribution in the formal sense.

Optimizing and Evaluating the Model

Given k Gaussian distributions and their weights, how do we optimize the model and evaluate how good a "fit" it is? For a single Gaussian distribution, the probability mass is 1. A natural thing to do would be to take an interval of say $1/4$ the Gaussian distribution and this interval should contain $1/4th$ the points. This is an ok approach but it is hard to generalize.

Instead, we will frame optimization as a "likelihood" problem. That is, given a collection of parameters that define our model what is the likelihood this model generated the data?

$$(3) \quad Pr[\mathcal{X} \mid f(\mu_i, \sigma_i, w_i, k)]$$

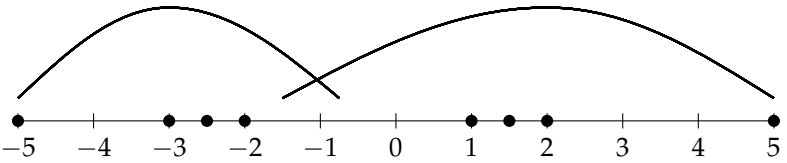

Figure 2: What is the likelihood the model generated these points?

We want to maximize this likelihood and this is called the "maximum-likelihood" approach written as

$$(4) \quad \arg\max_{\theta} \; Pr[\mathcal{X} \mid \theta] = \prod_{i=1}^{n} f(x_i \mid \theta) \quad \textit{where} \;\; \theta = \{\mu_i, \sigma_i, w_i, k\}$$

We prefer to work with the natural logarithm of the likelihood function to avoid working with products. This is called the "log-likelihood" and we can now perform stochastic gradient descent to solve this optimization problem.

$$(5) \quad \arg\max_{\theta} \; \sum_{i=1}^{n} \ln f(x_i \mid \theta) \quad \textit{where} \;\; \theta = \{\mu_i, \sigma_i, w_i, k\}$$

This is a traditional statistics approach to this problem. We guess a good model class and then find parameters in that class that maximize the log-likelihood. Unfortunately, finding the optimal parameters is NP-hard even in simple cases. To cope with this, we use a heuristic algorithm called the Expectation-maximization (EM) algorithm to approximate the optimal parameters.

While we can't prove this outside of very simple cases, this is similar to solving the Maximum likelihood estimation (MLE) problem. Unsupervised learning is often used as a preprocessor to supervised learning to learn something about the data by clustering it.

Single Gaussian Distribution Example

Even when trying to fit a single Gaussian distribution to a set of points, things are not simple and can be NP-hard. The probability of a given Gaussian distribution producing N points is given as

$$Pr[\mathcal{X} \mid f(\mu,\sigma)] = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{\sigma^2}} = \left(\frac{1}{2\pi\sigma^2}\right)^{N/2} e^{-[\frac{(x_1-\mu)^2}{\sigma^2}+\ldots+\frac{(x_N-\mu)^2}{\sigma^2}]} \tag{6}$$

As such, we can frame maximizing the likelihood as a minimization problem

$$\arg\max_{\mu,\sigma} Pr[\mathcal{X} \mid f(\mu,\sigma)] = \arg\min_{\mu,\sigma} \frac{\sum_{i=1}^{N}(x_i-\mu)^2}{\sigma^2} \tag{7}$$

That is, minimize the part of e's power from eq. 6 that is in the brackets to maximize the entire expression.

Things only become more complicated when we consider a mixture of k Gaussian distributions. As a heuristic, we try to minimize the objective from eq. 7 only for the nearest μ for each $x \in \mathcal{X}$.

5 The k-Means Problem

To finish this lecture, we will introduce the k-means problem. Given points $x_1,\ x_2, \ldots,\ x_N \in \mathcal{X}$ we want to find $\mu_1,\ \mu_2, \ldots,\ \mu_k \in \mathcal{X}'$ (μ_i doesn't necessarily need to be a data point but a point from the same space) such that the following is minimized:

$$\sum_{i=1}^{N}(x_i - \mu_{c(i)})^2 \tag{8}$$

Where $c(i)$ is the index of the μ closest to point x_i. It turns out finding the optimal $\mu_1,\ \mu_2, \ldots,\ \mu_k$ solution is infeasible and NP-hard. Next lecture, we will take a closer look at this optimization problem and also an iterative algorithm (the k-means algorithm) to approximate a solution.