Unsupervised Machine Learning

Gabriel Magill[*]

(Dated: November 12, 2017)

In this McMaster Journal Club, we introduce concepts relating to unsupervised machine learning, where explicit answer labels $y^{(i)}$ are not present in the training set. We will discuss algorithms dealing with clustering and dimensionality reduction of the feature space, both in the context of probabilistic and non-probabilistic methods. In particular, we will discuss Principle Component Analysis, Singular Value Decomposition, K-Means clustering, the EM algorithm, and mixture of Gaussians. I gave a similar talk last semester on concepts relating to supervised learning.

I. DISCLAIMER

These set of notes are based heavily on the course "CS229 - Machine Learning", taught be Andrew Ng at Stanford University, from this website https://see.stanford.edu/Course/CS229/.

II. INTRODUCTION

In unsupervised learning, we are given a dataset $\{x^{(1)}, \ldots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$. Each entry in the $x^{(i)}$ vector is called a feature, and the collection of all the $\{x^{(i)}\}$ is the training set. What differentiates supervised from unsupervised learning is the absence of a corresponding "right answer" $y^{(i)}$ corresponding to each $x^{(i)}$. Recalling the story from supervised learning, these $y^{(i)}$ labels were crucial in extremizing the posterior/likelihood distributions, doing regression, figuring out weights in a neural net, etc. What can we say in the absence of $y^{(i)}$? It turns out there are many techniques available. For example, we can reduce the feature space to a lower dimension or we can estimate the underlying density structure of the existing feature space. See Table I for examples. The model based approaches in that table will involve minimizing a posterior distribution by recursively "guessing" $z^{(i)}$ labels (in supervised learning, these were called $y^{(i)}$) and recursively extremizing the parameters θ. In the non-probabilistic approaches, we don't have a likelihood function, so we need to resort to other techniques.

	Model Based	Not Probabilistic
Dim. Reduction	Factor Analysis	PCA
Density Est.	Mixture of Gaussians	K-Means

TABLE I: Classification of popular unsupervised algorithms according to whether they reduce dimensionality of feature space or estimate density within existing feature space, and whether they rely on an explicit probabilistic model or if they are inherently non-probabilistic.

[*] gabriel.magill@gmail.com

III. NON-PROBABILISTIC METHODS

A. K-Means

K-Means is a very straightforward example demonstrating the idea of clustering. Given data $\{x^{(1)}, \ldots, x^{(m)}\}$, where $x^{(i)} \in \mathbb{R}^n$, we try to estimate distinct clusters in the data by recursively estimating the mean (or centroids) of the clusters. The algorithm is as follows

```
1. Initialize cluster centroids $\{\mu_1, \ldots, \mu_k\} \in \mathbb{R}^n$ randomly.
2. Repeat until convergence: {
$\forall i$, set $c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2$
$\forall$j, set $\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^m 1\{c^{(i)}=j\}}$
}
```

So we assume there are k clusters (k needs to be determined). We initially randomly initialize the $\{\mu_k\}$. For each datapoint $x^{(i)}$, we figure out which cluster its closest to. Then we update the $\{\mu_k\}$ centroids with the average of the $x^{(i)}$ that are closest to it. We repeat this until the μ don't change. It can be shown that the distortion function

$$J(c, \mu) \equiv \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}|| \tag{1}$$

is monotonically decreasing for each iteration. So in this sense, K-Means is guaranteed to converge.

B. Principal Component Analysis (PCA)

PCA is a very cool technique where we reduce the dimensionality of the feature space by projecting the data $x^{(i)}$ to a lower dimensional subspace. It is often used as a preprocessing step to clean up data before doing clustering, density estimation, and so on. We start by giving the algorithm behind it, and then discuss useful applications. We begin by taking our data $x^{(i)}$ and subtracting out the mean and standard deviation across each feature.

$$x_j^{(i)} \to \frac{x^{(i)} - \mu_j}{\sigma_j} \tag{2}$$

Then, we project onto a subspace spanned by $u^{(k)}$. We choose this sub-basis to be orthonormal. Expressing our data in this basis amounts to $x^{(i)} \to y^{(i)}$, where (using Einstein summation notation), we have

$$y^{(i)} = u_j^{(k)} x_j^{(i)} u^{(k)}. \tag{3}$$

We want to choose our new u sub-basis such that it maximizes the variance of the projections. This makes sense, because suppose we have minimized the variance, then all the data points would map onto a single point in the subspace, and we would have lost all information. In math, this means that we want to maximize $y_j^{(i)} y_j^{(i)}$ subject to the constraint that u are normalized and orthogonal. Using Lagrange multipliers and defining λ^{kK} to be a diagonal matrix, we extremize:

$$\begin{aligned}\mathcal{L} &= y_j^{(i)} y_j^{(i)} - \lambda^{kK}(u_j^{(k)} u_j^{(K)} - \delta_{kK}) \\ &\equiv u_j^{(k)} x_j^{(i)} u_\ell^{(k)} u_\ell^{(k')} x_m^{(i)} u_m^{(k')} - \lambda_{kK}(\ldots) \\ &= u_j^{(k)} x_j^{(i)} x_m^{(i)} u_m^{(k)} - \lambda_{kK}(\ldots)\end{aligned} \tag{4}$$

In the second line, we have used orthogonality condition which follows from doing $d\mathcal{L}/d\lambda = 0$. Calculating $d\mathcal{L}/du_{j'}^{(k')}$, we obtain after a little algebra that:

$$x_{j'}^{(i)} x_m^{(i)} u_m^{(k')} = \lambda^{k'k'} u_{j'}^{(k')} \tag{5}$$

where the term of the RHS is not Einstein summed. We recognize this as an eigenvalue problem for the matrix $X^T_{j'(i)} X_{(i)m}$. In particular, doing PCA amounts to finding a number of principle eigenvalues of the correlation matrix $\Sigma_{ab} = \frac{1}{m}\sum_{i=1}^m x_a^{(i)} x_b^{(i)}$ which is true in the 0-mean limit. The collection of the corresponding eigenvectors spans the subspace, and we project all our data points onto this subspace. The fact that the correlation matrix is a real symmetric square matrix ensures that we were indeed able to chose our eigenvalues so as to form an orthonormal basis. In practice, the eigenvalues / eigenvectors are found using Singular Value Decomposition (SVD), a matrix decomposition routine implemented efficiently in virtually all programming languages. SVD states that a matrix X (which can be real or complex, but let's assume it's real) can be decomposed as:

$$X_{n\times p} = U_{n\times n} S_{n\times p} V^T_{p\times p} \tag{6}$$

where

$$\begin{aligned}&U^T U = 1,\ V^T V = 1 \\ &\text{Eigenvectors of } X^T X\ (XX^T) \text{ are columns of } V\ (U) \\ &\sqrt{\text{Eigenvalues}} \text{ of } X^T X \text{ or } XX^T \text{ are entries of } S\end{aligned} \tag{7}$$

This relates exactly to our problem. A nice intuition behind all of this is that linear operators such a matrix A rescales and rotates when acted upon vectors. Eigenvectors are special vectors which are only rescaled. The eigenvector corresponding to the principle eigenvalue is therefore the direction in the whole metric space which gets rescaled by the largest amount when acted upon by A. If A is a correlation matrix, the interpretation is then the direction which explains the largest spread in correlations. So if we take the n principle eigenvectors, we are saying in a sense that mostly all the correlation in data is captured by these, and everything else is noise on top.

Lastly, we discuss some examples. Suppose in our data, one feature measures a distance in km, and the other in miles. In reality, our data is $n-1$ dimensional. Doing PCA would eliminate this redundancy. Another (very cool) example of PCA is this Nature Paper, in which a series of different genetic markers are collected from various European individuals. Doing PCA to 2 dimensions, the data becomes organized in the (x,y) plane in clusters closely resembling the map of Europe! We see that PCA helps

- Visualize data to 2 or 3 dimensions
- Remove highly correlated features
- Improves convergence of other ML algorithms
- Removes noise

IV. MODEL BASED APPROACH

A. EM Algorithm

In everything described above, we had no PDFs or underlying probabilistic models organizing the data. Here, we assume that the data can be assumed (for whatever reason) to follow a PDF/likelihood function $p(x;\theta)$, and the goal is to estimate the parameters θ of the data $\{x^{(1)}, \ldots, x^{(m)}\}$ in the absence of labels $y^{(i)}$. Instead, we will consider latent/unobserved labels $z^{(i)}$, which are kind of like $y^{(i)}$ but they are unobserved. We describe the EM (Estimation Maximization) Algorithm, which is a widely used algorithm that recursively guesses $z^{(i)}$, maximizes $p(x;\theta)$ and updates θ. This is the sense in which the likelihood function can be maximized, because taking derivatives of $p(x;\theta)$ wrt θ and setting to 0 is analytically intractable.

Lemma: Jensen's inequality: If $f(x)$ is a concave function ($f''(x) \leq 0$) and x a random variable, then $f(E[x]) \geq E[f(x)]$. This inequality saturates to an equality when the expectation E is taken over a "constant" random variable.

The proof for this can be found in an analysis or probability book. We now derive the EM algorithm. For each i, let Q_i be a distribution over the latent z random variable, such that $\int Q_i(z)dz = 1$ or $\sum_z Q_i(z) = 1$. We want to maximize the likelihood function

$$\begin{aligned}
\ell(\theta) &= \sum_i \log p(x^{(i)};\theta) \\
&= \sum_i \log \sum_{z^{(i)}} p(x^{(i)}, z^{(i)};\theta) \\
&= \sum_i \log \sum_{z^{(i)}} Q_i(z^{(i)}) \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})} \\
&\geq \sum_i \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}
\end{aligned} \tag{8}$$

In the last step, we have used Jensen's inequality, where the expectation E is taken over the distribution Q_i and $f(x)$ has become $\log p(x^{(i)}, z^{(i)};\theta)/Q_i(z^{(i)})$, with random variable $z^{(i)}$. We now want to saturate the inequality to provide a tight bound. Choosing

$$\begin{aligned}
Q_i(z^{(i)}) &= \frac{p(x^{(i)}, z^{(i)};\theta)}{\sum_z p(x^{(i)}, z;\theta)} \\
&= p(z^{(i)}|x^{(i)};\theta)
\end{aligned} \tag{9}$$

Makes the random variable $p(x^{(i)}, z^{(i)};\theta)/Q_i(z^{(i)})$ constant with respect to z under the Q_i distribution. In the last step, we have used the law of conditional probabilities, that $p(A|B) = p(A,B)/p(B)$. Since the "random variable" is now constant, the inequality becomes an equality and we are left with

$$\bar{\ell}(\theta) = \sum_i \sum_{z^{(i)}} p(z^{(i)}|x^{(i)};\theta) \log \frac{p(x^{(i)}, z^{(i)};\theta)}{p(z^{(i)}|x^{(i)};\theta)}. \tag{10}$$

The EM algorithm is as follows. Letting f denote the index of the current iteration, we repeat until convergence {

E-Step: $\forall i$, $Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta_{f-1})$
Where we consider $Q_i(z^{(i)})$ to be completely fixed in terms of the parameters θ_{f-1} of the previous iteration.

M-Step: Set $\theta_f := \arg\max_\theta \bar{\ell}(\theta)$

}
The bar on $\bar{\ell}(\theta)$ is used to emphasize that $\bar{\ell}(\theta) = \ell(\theta)$ only at $\theta = \theta_{f-1}$. This is true because we have saturated Jensen's inequality at this value of θ. However, maximizing $\bar{\ell}(\theta)$ doesn't return the maximum of $\ell(\theta)$ since these two functions are different. I believe that the initial value for the parameters θ is a random assignment.

B. Mixture of Gaussians

Let's consider an example of EM algorithm in which the data follows a mixture of Gaussians. We begin by defining the model, and then relating each component to the EM algorithm. The model posits that the data inherently follows a product of k different Gaussian distributions (like a mountain landscape). For data i, we begin by picking a $z^{(i)}$ (i.e. the Gaussian j that generated $x^{(i)}$). The $z^{(i)}$ follow a Multinomial(ϕ) distribution with $\phi_j > 0$ and $\sum_j \phi_j = 1$, and $p(z^{(i)} = j) = \phi_j$. Supposing we have picked Gaussian j as generating $x^{(i)}$, then $x^{(i)}|(z^{(i)} = j) \sim \mathcal{N}(\mu_j, \sigma_j)$. It is important to note that $z^{(i)}$ is never directly observed in the dataset. It is just used as a latent variable for optimizing. Notice how similar this is to K-Means, with the replacement of hard assignments to different clusters with soft assignments based on a probability model. Relating this to the EM algorithm, the **E-Step** becomes

$$\begin{aligned}
Q_i(z^{(i)} = j) &= p(z^{(i)} = j|x^{(i)};\phi,\mu,\Sigma) \\
&= \frac{p(x^{(i)}|z^{(i)} = j;\mu,\Sigma)p(z^{(i)} = j;\phi)}{\sum_{l=1}^k p(x^{(i)}|z^{(i)} = l;\mu,\Sigma)p(z^{(i)} = l;\phi)}
\end{aligned} \tag{11}$$

In the above, $p(z^{(i)} = j;\phi) = \phi_j$ and we have used Bayes theorem and the current best guess for the model parameters. For the **M-Step**, we want to maximize

$$\begin{aligned}
&\bar{\ell}(\theta) = \\
&\quad \sum_{i=1}^m \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)};\phi,\mu,\Sigma)}{Q_i(z^{(i)})} \\
&\quad = \sum_{i,j}^{m,k} Q_i(z^{(i)} = j) \log \frac{p(x^{(i)}|z^{(i)} = j;\phi,\mu,\Sigma)p(z^{(i)} = j;\phi)}{Q_i(z^{(i)} = j)} \\
&\quad = \sum_{ij} \ldots \log \frac{\exp\left(-\frac{1}{2}(x^{(i)} - \mu_j)^T \Sigma_j^{-1}(x^{(i)} - \mu_j)\right) \cdot \phi_j}{\cdots}
\end{aligned} \tag{12}$$

We now want to obtain update rules. Setting the derivative $\nabla_{\mu_l}\ell(\theta) = 0$, we get the update rule:

$$\mu_l := \frac{\sum_{i=1}^m Q_i(z^{(i)} = l)x^{(i)}}{\sum_{i=1}^m Q_i(z^{(i)} = l)}. \tag{13}$$

Any parameters showing up on the RHS are from the previous iteration. Similarly, there will be update rules for ϕ_j and for Σ_j.

C. Factor Analysis

Factor analysis also assumes that the data follows a probabilistic model and runs the EM algorithm. In mixture of Gaussians, we assumed that there are m data examples with n features. We assume that $m \gg n$ so the algorithm can clearly resolve each of the Gaussian peaks in the feature space. Suppose on the other hand that $m \ll n$; we have only a few data points, each with a large number of features. Factor Analysis deals with this case, and attempts to make broad educated guesses on the distribution of data based on a few examples. It's used for example for targeting ads for relevant products when the user has clicked on only a few products.