Unsupervised Learning

F. Song

1 k mean clusters

In the clustering problem, we are given a training set$\{x^{(1)}, ...x^{(m)}\}$ with no class label(unsupervised learning), and want to group the data into a few cohesive "clusters". The k-means clustering algorithm is as follows:

1. Initialize **cluster centroids** $\mu_1, \mu_2, ...\mu_k \in R^n$ randomly;
2. Repeat until convergence:
   For every i, set
   $$c^{(i)} := argmin_j \|x^{(i)} - \mu_{(j)}\|^2$$
   for each j, set
   $$\mu_j := \frac{\sum_{i=1}^m 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1} m1\{c^i = j\}}$$

In the algorithm above, k (a parameter of the algorithm) is the number of clusters we want to find. To initialize the cluster centroids, we could choose k training examples randomly, and set the cluster centroids to be equal to the values of these k examples. Then algorithms repeatedly carries out two steps: (i) "Assigning" each training examples $x^{(i)}$ to the clusters centroid $\mu_{(i)}$ and (ii) moving each cluster centroid μ_j to the mean of the points assigned to it. Let define the **distortion function** to be:

$$J(c, \mu) = \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2$$

K means is the cooridinate descent on J. Specifically, the inner loop of the $k - means$ repeatedly minimize J with respect to c while holding μ fixed, and then minimize J with respect to μ while holding c fixed.

A similar algorithm is **Mixture of Gaussian Model**

2 Mixture of Gaussian model

Suppose that we are given a training set$\{x^{(1)}, ...x^{(m)}\}$, we wish to model the data by specifying a joint distribution$p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$. Here, $z^i \sim Multinomial(\phi)$ and the parameter ϕ_j gives $p(z^i = j)$, and $x^i|(z^i = j) \sim N(\mu_i, \Sigma_j)$,$k$ denote the number of values that the $z^{(i)}$ can take on. This model

posits that each x^i was generated by randomly choosing $z^{(i)}$ from $\{1, ..k\}$, and then x^i was drawn from one of k Gaussians depending on $z^{(i)}$.Note that the $z^{(i)}$'s are **hidden/hiden** random variables. The parameters of our model are ϕ, μ, Σ. To estimate them, we write down the likelihood of our data:

$$\begin{aligned} l(\phi, \mu, \Sigma) &= \sum_{i=1}^{m} logp(x^{(i)}, z^{(i)}; \phi, \mu, \Sigma) \\ &= \sum_{i=1}^{m} log \sum_{z^i=1}^{k} p(x^{(i)}|z^i; \mu, \Sigma)p(z^i, \phi) \end{aligned}$$

However, if we set the derivatives of this formula with respect to the parameters to zero and try to solve, we will find that it is not possible to find the maximum likelihood estimates of the parameters in closed form, because the class label $z^{(i)}$ were unknown. If the z^i were given, the maximum likelihood estimation becomes identical to the GDA model. So solve this problem, we can use the EM(Expectation - Maximization) algorithm.

The EM algorithm is an iterative algorithm that has two steps. In the E-step, it tries to "guess" the values of the $z^{(i)}$ s. In the M-step, it updates the parameters of our model based on our guesses. Specifically, for the mixtures of Gaussian problem, in the E-step, we calculate the posterior probability of $z^{(i)}$ given the $x^{(i)}$ and using the current setting of our parameters. I.e., using Bayes rule. The values $w_j^{(i)}$ calculated in the E-step represents our soft guesses(guess being probabilities) for the value of $z^{(i)}$; In the M-step, it's identical to GDA. The EM algorithms is also reminiscent of the K-means clustering algorithm, except that instead of the "hard" cluster assignment, we have the soft guess. Similar to K-clusters, it is also susceptible to local optima, so reinitializing at several different initial parameters may be a good idea.

Repeat until convergence: { (E-step) For each i, j, set

$$w^{(i)} := p(z^i = j|x^i; \phi, \mu,) \tag{1}$$

(M-step) Update teh parameters:

$$\begin{array}{rcl} \phi_j & := & \frac{1}{m}\sum_{i=1} m w_j^{(i)} \\ \mu_j & := & \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)} x^{(i)} \\ \Sigma_i & := & \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1} m w_j^{(i)}} \end{array} \quad \}$$

2.1 The EM algorithm

EM algorithm can be applied to a large family of estimation problems with latent variables.

Jensen's inequality Let f be a convex function, and let X be a random variable. Then:

$$E[f(x)] \leq f(EX)$$

Suppose we have an estimation problem in which we have a training set $x^{(1)}, ...x^{(m)}$, we wish to fit the parameters of a model $p(x, z)$ to the data. The likelihood is

$$
\begin{aligned}
l(\theta) &= \sum_{i=1}^{m} logp(x^{(i)};\theta) \\
&= \sum_{i=1}^{m} log \sum_{z} p(x^{(i)}, z^{(i)};\theta)
\end{aligned}
$$

For each i, let Q_i be some distribution over z. Consider:

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

To make the bound tight for a particular value of θ, we set $Q_i(z^{(i)})$ to be the posterior distribution of the $z^{(i)}$ given $x^{(i)}$ and θ. Thus we get the lower bound on l that we're trying to maximize. This is the E-step. In the M-step, we maximize l with respect to the parameters to obtain a new setting of θ. Repeatedly carrying out these two steps gives the EM algorithm:
Repeat until convergence{
(E-step) For each i: set $Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)};\theta)$
(M-step) set $\theta := argmax_\theta \sum_{i} \sum_{z^{(i)}} Q_i(z^{(i)}) log \frac{p(x^{(i)}, z^{(i)};\theta)}{Q_i(z^{(i)})}$

3 Factor Analysis

To estimate a model which explains variance/ co-variance between a set of observed variables by a set of unobserved factors + weighting

$$
\begin{aligned}
l(\phi, \mu,) &= \sum_{i=1}^{m} logp(x^{(i)}, z^{(i)};\phi, \mu, \Sigma) \\
&= \sum_{i=1}^{m} log \sum_{z^i=1}^{k} p(x^{(i)}|z^i;\mu, \Sigma) p(z^i, \phi)
\end{aligned}
$$

However, if we set the derivatives of this formula with respect to the parameters to zero and try to solve, we will find that it is not possible to find the maximum likelihood estimates of the parameters in closed form, because the class label $z^{(i)}$ were unknown. If the z^i were given, the maximum likelihood estimation becomes identical to the GDA model. So solve this problem, we can use the EM(Expectation - Maximization) algorithm.

The EM algorithm is an iterative algorithm that has two steps. In the E-step, it tries to "guess" the values of the $z^{(i)}$ s. In the M-step, it updates the parameters of our model based on our guesses. Specifically, for the mixtures of Gaussian problems, in the E-step, we calculate the posterior probability of $z^{(i)}$ given the $x^{(i)}$ and using the current setting of our parameters. I.e., using Bayes rule. The values $w_j^{(i)}$ calculated in the E-step represents our soft guesses(guess being probabilities) for the value of $z^{(i)}$; In the M-step, it's identical to GDA.

4 Principle Component Analysis

PCA can speed up the running time of an algorithm, helps visualization, choose k by figuring out % of variance retained. Bad use of PCA includes: prevent overfitting, use regularization instead. Before implementing PCA, first try running whatever you want to do with the original raw data. Only if that does not do what you want, then implement PCA.

Reduce data from n dimension to k dimension: find k vectors onto which to project the data so as to minimize the projection error.