CSE 446: Machine Learning **Lecture**

Intro to Probabilistic Graphical Models and The EM Algorithm

Instructor: Sham Kakade

1 Basic idea of 'generative models'

We are now going to specify the method in which we believe our data are generated. This does not tell us how to learn the parameters of the model. However, specifying these procedures are helpful abstractions as they then give us a way to answer questions such as: what are the document groupings? How dow decided upon what is a good 'rule' to use to group our documents? The following approach allows us to address these questions in a principled (and general) approach.

For example, let us view each document as datapoint. And let us view each document as being represented by the word counts in the document. So each datapoint is just a collection of word counts (suppose we have M documents and each document is specified by a big vector of word counts). So how should we group our documents together?

Before we can answer the question, let us take a different viewpoint. For now, let us just specify a procedure for how our documents are generated; a probabilistic generative model is an underlying model of how our data are created. In what follows, we will consider a simple 'single topic' case, we assumed that each datapoint/document has a hidden topic associated with it. And that the words we observed were generated under a distribution over words implied by the topic. The learning question (next class!) is how we figure out the topics and (soft) document assignments given our data, by using our modeling assumptions.

These notes just specify a few generative models.

2 The Expectation Maximization (EM) algorithm for Gaussian Mixtures

2.1 The Mixture of Gaussians Model

This model was first introduced by [1], with an application to evolutionary biology.

Random variables: a "hidden" cluster $i \in \{1 \dots k\}$ and a vector $x \in \mathbb{R}^d$.

Parameters: "mixing weights" $\pi_i = \Pr(\text{topic} = i)$, means: $\mu_1 \dots \mu_j$, noise covariance matrices $\Sigma_1, \Sigma_2, \dots \Sigma_k$

The Generative model for a datapoint:

1. sample a cluster i, which has probability π_i
2. observe x, where x is the mean μ_i corrupted with Gaussian noise:

$$x = \mu_i + \eta$$

where η has a multivariate normal distribution, $N(0, \Sigma_i)$.

2.2 Maximum Likelihood Estimation

Suppose we have datapoints $x_1, \ldots x_N$. The maximum likelihood estimation problem is:

$$\arg\max_{\text{params}} \Pr(x_1 \ldots x_N|\text{params}) = \arg\max_{\text{params}} \log \Pr(x_1 \ldots x_N|\text{params}) \quad (1)$$

$$= \arg\max_{\text{params}} \sum_{n=1}^{N} \log \Pr(x_n|\text{params}) \quad (2)$$

where the params are the "mixing weights" $\pi_i = \Pr(\text{topic} = i)$, means: $\mu_1 \ldots \mu_j$, noise covariance matrices $\Sigma_1, \Sigma_2, \ldots \Sigma_k$.

Let $h_n \in \{1 \ldots k\}$ be the unknown hidden cluster associated with datapoint i. Let us examine how we would compute the probability of any single term:

$$\Pr(x_n|\text{params}) = \sum_{i=1}^{k} \Pr(x_n, h_n = i|\text{params}) = \sum_{i=1}^{k} \pi_i N(x_n|\mu_i, \Sigma_i)\,.$$

where $N(x_n|\mu_i, \Sigma_i)$ is the density of x_n under a Gaussian distribution. Using this, we have that the optimization problem is:

$$\arg\max_{\theta} \sum_{n=1}^{N} \log \sum_{i=1}^{k} \pi_i N(x_n|\mu_i, \Sigma_i)\,.$$

which shows how the log likelihood function depends on the parameters θ.

2.3 The EM algorithm: an example

The EM algorithm is a general iterative algorithm for learning parameters in models, like this mixture of Gaussians problem. It is more generally applicable than just for the mixture of Gaussians.

Warmup: $K = 2$ **and** $d = 1$

Assume $K = 2$ and $d = 1$. Consider an example where our data are in one dimension, i.e. for all i, $x_i \in \mathbb{R}$. Let us also consider the problem of learning a mixture of two Gaussians, so our parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1^2, \sigma_2^2$.

Note that for 1 dimensional Gaussian distributions, the convention is to use σ^2 to denote their variances, while for multivariate Gaussian distributions, the convention is to use Σ for the covariance matrix.

Initialization. The EM algorithm is an *alternating minimization* algorithm. We start with some guess of these parameters, say $\widehat{\pi}_1, \widehat{\pi}_2, \widehat{\mu}_1, \widehat{\mu}_2, \widehat{\sigma}_1^2, \widehat{\sigma}_2^2$, and then alternate between the E and M steps as follows:

The E-step. Using our current estimate of the parameters, estimate the "soft weights" as follows:

$$z_n := \Pr(h_n = 1|x_n) = \frac{\Pr(h_n = 1, x_n)}{\Pr(x_n)} = \frac{\widehat{\pi}_1 N(x_n|\widehat{\mu}_1, \widehat{\sigma}_1^2)}{\widehat{\pi}_1 N(x_n|\widehat{\mu}_1, \widehat{\sigma}_1^2) + \widehat{\pi}_2 N(x_n|\widehat{\mu}_2, \widehat{\sigma}_2^2)}$$

The M-step. Now update our parameters as follows. For the weights,

$$\widehat{\pi}_1 \leftarrow \frac{\sum_n z_n}{N}, \qquad \widehat{\pi}_2 \leftarrow \frac{\sum_n (1 - z_n)}{N}$$

the means,

$$\widehat{\mu}_1 \leftarrow \frac{\sum_n z_n x_n}{\sum_n z_n}, \qquad \widehat{\mu}_2 \leftarrow \frac{\sum_n (1 - z_n) x_n}{\sum_n (1 - z_n)}$$

and for the variances:

$$\widehat{\sigma}_1^2 \leftarrow \frac{\sum_n z_n (x_n - \widehat{\mu}_1)^2}{\sum_n z_n}, \qquad \widehat{\sigma}_2^2 \leftarrow \frac{\sum_n (1 - z_n)(x_n - \widehat{\mu}_2)^2}{\sum_n (1 - z_n)}.$$

General K and d.

Now suppose our data are in d dimensions and that we would like to fit our model with K Gaussians.

Initialization. Again, we start with some guess of these parameters, say $\widehat{\pi}_1, \ldots \widehat{\pi}_K, \widehat{\mu}_1, \ldots \widehat{\mu}_K, \widehat{\Sigma}_1^2, \ldots \widehat{\Sigma}_K^2$. Note that $\widehat{\pi}_j$ is scalar, $\widehat{\mu}_j$ is a d-dimensional vector, and $\widehat{\Sigma}_j$ is a $d \times d$ matrix. We then alternate between the E and M steps as follows:

The E-step. Using our current estimate of the parameters, our estimate of the "soft weights" our now K dimensional. These are:

$$z_{n,j} := \Pr(h_n = j | x_n) = \frac{\Pr(h_n = j, x_n)}{\Pr(x_n)} = \frac{\widehat{\pi}_j N(x_n | \widehat{\mu}_j, \widehat{\Sigma}_j)}{\widehat{\pi}_1 N(x_n | \widehat{\mu}_1, \widehat{\Sigma}_1) + \ldots + \widehat{\pi}_K N(x_n | \widehat{\mu}_K, \widehat{\Sigma}_K)}$$

which is computed for all $j \in \{1, \ldots K\}$ and for all datapoints n.

The M-step. For all $j \in \{1, \ldots K\}$, the our parameter updates are as follows: for the weights,

$$\widehat{\pi}_j \leftarrow \frac{\sum_n z_{n,j}}{N},$$

the means,

$$\widehat{\mu}_j \leftarrow \frac{\sum_n z_{n,j} x_n}{\sum_n z_{n,j}},$$

and for the variances:

$$\widehat{\Sigma}_j \leftarrow \frac{\sum_n z_{n,j} (x_n - \widehat{\mu}_j)(x_n - \widehat{\mu}_j)^\top}{\sum_n z_{n,j}}.$$

which where $a^\top$ denotes the transpose of a vector a.

3 Other common generative models

3.1 "Bag of words" model: a (single) topic model

Suppose every document has T words.

Random variables: a "hidden" topic $i \in \{1 \dots k\}$ and a T-word outcomes $w_1, w_2, \dots w_T$ which take on some discrete values.

Parameters: the "mixing weights" $\pi_i = \Pr(\text{topic} = i)$, the "topics" $b_{wi} = \Pr(\text{word} = w|\text{topic} = i)$

The generative model for an T word "document", where every document is only about one topic.

1. sample a "topic" i, which has probability π_i
2. gererate T words $w_1, w_2, \dots w_T$, independently. in particular, we choose word w_t as the t-th word with probability $b_{w_t i}$.

Note this generative model ignores the word order, so it is not a particularly faithful generative model.

Due to the 'graph' (i.e. the conditional independencies implied by the generative model procedure), we can write the *joint* probability of the outcome topic i occurring with a document containing the words $w_1, w_2, \dots w_T$ as:

$$\begin{aligned}\Pr(\text{topic} = i \text{ and } w_1, w_2, \dots w_T) &= \Pr(\text{topic} = i) \Pr(w_1, w_2, \dots w_T|\text{topic} = i) \\ &= \Pr(\text{topic} = i) \Pr(w_1|\text{topic} = i) \Pr(w_2|\text{topic} = i) \Pr(w_T|\text{topic} = i) \\ &= \pi_i b_{w_1 i} b_{w_2 i} \dots b_{w_T i}\end{aligned}$$

where the second to last step follows due to the fact that the words are generated independently given the topic i.

3.1.1 Inference

Suppose we were given a document with $w_1, w_2, \dots w_T$. One *inference* question would be what is the probability the underlying topic is i. By Bayes rule, we have:

$$\begin{aligned}\Pr(\text{topic} = i|w_1, w_2, \dots w_T) &= \frac{1}{\Pr(w_1, w_2, \dots w_T)} \Pr(\text{topic} = i \text{ and } w_1, w_2, \dots w_T) \\ &= \frac{1}{Z} \pi_i b_{w_1 i} b_{w_2 i} \dots b_{w_T i}\end{aligned}$$

where Z is a number chosen so that the probabilities sum to 1. Critically, note that Z is not a function of i.

3.1.2 LDA: latent Dirichlet allocation

This is a popular model which allows documents to contain more than one topic.

3.2 Hidden Markov models

Random variables: a "hidden" state sequence $z_1, z_2, \dots z_T$ (which take on some discrete values in some 'hidden state space') and T-discrete sequential outcomes $w_1, w_2, \dots w_T$. Suppose each z_i can take one of k outcomes and each w_i can take one of d outcomes.

Parameters: π_i, $A_{ji} = \Pr(z_{n+1} = j|z_t = i)$, $b_{mi} = \Pr(x_n = m|z_t = i)$

The Generative model for an t word "document": for each time t,

1. sample a "hidden" state sequence z_{n+1}, using only the previous outcome z_t. The sampling is determined solely by the parameters $\{A_{ji}\}$.

2. Sample w_{n+1} using only z_{n+1}. This sampling is based only on the probabilities $\{b_{mi}\}$.

Due to the 'graph' (i.e. the conditional independencies implied by the generative model procedure), we can write the joint probability of the hidden state sequence $z_1, z_2, \ldots z_T$ and the word sequence $w_1, w_2, \ldots w_T$ as:

$$\Pr(z_1, z_2, \ldots z_T \text{ and } w_1, w_2, \ldots w_T) = b_{w_1 z_1} A_{z_2 z_1} b_{w_2 z_2} A_{z_3 z_2} \ldots b_{w_T z_T}$$

One inference question would be to determine the probability that hidden state is $z_t = j$ given some observed sequence $w_1, w_2, \ldots w_T$, i.e.

$$\Pr(z_t = j | w_1, w_2, \ldots w_T)$$

Naively, this computation might look difficult. However, this can be done in a computationally efficiently manner using the Baum-Welch algorithm, sometimes known as the "Forward-Backward" algorithm.

References

[1] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of The Royal Society*, 185:71–110, 1894.