# CS 536 : Graphical Models

16:198:536

## 1 What is a Graphical Model? Why is a Graphical Model?

So far we've been concerned with problems of learning a feature $Y$ from a set of features $\underline{X}$. Frequently though we may be interested in predicting or learning a collection of features, $\underline{Y}$. We could potentially learn a model for each component of $\underline{Y}$, but this would multiply our models unnecessarily and ignore fundamental interconnections between the variables - a good example of this is in part of speech prediction, where the part of speech of various words may actually be informative about the parts of speech of other words. We could try to embed the problem into a larger dimensional space (for instance if $\underline{Y}$ is three boolean values, consider the space of all $2^3 = 8$ possible value assignments) but this is frequently computationally intractable for even modest sized spaces.

An alternate approach is to construct a model for how all the features in $\underline{X}$ and all the features in $\underline{Y}$ vary together - that is, model the *joint distribution* $\mathbb{P}(\underline{X}, \underline{Y})$. With this joint model in place, we could then perform inference or prediction by computing on some new feature set $\underline{x}$,

$$\mathbb{P}\left(\underline{Y}|\underline{X} = \underline{x}\right). \tag{1}$$

The distinction between the $\underline{X}$ and $\underline{Y}$ features is in fact arbitrary - we can consider a joint model over a number of features, and consider conditioning on available features to make predictions about the remaining features.

In this way, having a model for the joint distribution can be useful for multi-variate prediction, inference, etc. Another advantage that these kinds of models have is the ability to sample or generate data according to a distribution. Given all or some of the available features, we can compute the distribution for the remaining features, and *sample* from this distribution to generate data according to that distribution.

## 2 How to Represent a Graphical Model

A graphical model is nothing more than a representation of the joint distribution between a set of random variables. Given a set of variables $\underline{X} = (X_1, X_2, \ldots, X_k)$, we represent the joint distribution between the variables with a dependency graph $G$, and a consistent conditional distribution $P$ on $G$. The dependency graph $G$ captures the 'immediate' dependencies of the variables on each other - if $(X_i \sim X_j) \in G$, we say that $X_i$ and $X_j$ are immediately dependent on each other. A conditional distribution $P$ is *markov consistent* on $G$ if for any $i$,

$$P\left(X_i|X_1, X_2, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k\right) = P\left(X_i|N_G(X_i)\right), \tag{2}$$

where $N_G(X_i)$ are the immediate neighbors of $X_i$ in $G$. That is, the value of $X_i$, conditioned on every other variable, depends only on the values of its neighbors in $G$. A choice of $G$ and consistent distribution $P$ is known as a **Markov Random Field**.

Any Markov Random Field can be represented (as a data structure) in the following way: given $G$ and a consistent distribution $P$, we have as a theorem that

$$\mathbb{P}\left(X_1 = x_1, \ldots, X_k = x_k\right) \propto \exp\left(\sum_C \theta_C \Phi_C(\underline{x})\right), \tag{3}$$

where the sum is taken over cliques $C$ in $G$, $\theta_C$ is some real valued weight on that clique, and $\Phi_C$ is a 'potential function' that depends only on the specific variables in $C$.

To turn this into a proper probability, we need to normalize it. It is convenient to introduce the following notation:

$$e^{A(\underline{\theta})} = \sum_{\underline{x}} \exp\left(\sum_C \theta_C \Phi_C(\underline{x})\right), \tag{4}$$

hence we can normalize the probabilities to yield

$$\mathbb{P}\left(X_1 = x_1, \ldots, X_k = x_k\right) = \exp\left(\sum_C \theta_C \Phi_C(\underline{x}) - A(\underline{\theta})\right) \tag{5}$$

Hence any Markov Random Field can be represented with a choice of dependency graph $G$, a choice of clique weights $\underline{\theta}$, and potential functions on each clique.

*Author's Note: A common, though not required, choice of potential function is $\Phi_{i\sim j}(\underline{x}) = x_i * x_j$. This is very convenient in the case of binary or boolean values for the x-values, in which case the product simulates an and or xor function (depending on the binary values you choose).*

# 3 How to Compute on a Graphical Model

Given a graphical model (delaying, for a moment, the question of where graphical models come from), one of the most important things you will want to do is be able to evaluate the probabilities of queries - what is the probably that a variable (or multiple variables) have a given value?

Consider as an example a graphical model on five variables, with values $\pm 1$, where the dependency graph $G$ contains $X_1 \sim X_5, X_2 \sim X_5, X_3 \sim X_5, X_4 \sim X_5$. According to the representation theorem, we have that

$$\mathbb{P}\left(X_1 = x_1, \ldots, X_5 = x_5\right) = \exp\left(\sum_{i=1}^{5} \theta_i x_i + \sum_{i=1}^{4} \theta_{i,5} x_i x_5 - A(\underline{\theta})\right). \tag{6}$$

Suppose we wanted to compute the probability that $X_5 = x_5$. In order to do that, we would have to *marginalize* over the remaining variables. That is,

$$\mathbb{P}\left(X_5 = x_5\right) = \sum_{x_1, x_2, x_3, x_4} \exp\left(\sum_{i=1}^{5} \theta_i x_i + \sum_{i=1}^{4} \theta_{i,5} x_i x_5 - A(\underline{\theta})\right), \tag{7}$$

where in the above sum we are summing over all possible values of $x_1$ through $x_4$. This can be simplified in the following way:

$$\begin{aligned}
\mathbb{P}\left(X_5 = x_5\right) &= \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4} \exp\left(\sum_{i=1}^{5} \theta_i x_i + \sum_{i=1}^{4} \theta_{i,5} x_i x_5 - A(\underline{\theta})\right) \\
&= \exp\left(\theta_5 x_5 - A(\underline{\theta})\right) \sum_{x_1}\sum_{x_2}\sum_{x_3}\sum_{x_4} \exp\left(\sum_{i=1}^{4} \theta_i x_i + \sum_{i=1}^{4} \theta_{i,5} x_i x_5\right) \\
&= \exp\left(\theta_5 x_5 - A(\underline{\theta})\right) \sum_{x_1}\sum_{x_2}\sum_{x_3} \exp\left(\sum_{i=1}^{3} \theta_i x_i + \sum_{i=1}^{3} \theta_{i,5} x_i x_5\right) \sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{4,5} x_4 x_5\right) \\
&= \exp\left(\theta_5 x_5 - A(\underline{\theta})\right) \left[\sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{4,5} x_4 x_5\right)\right] \sum_{x_1}\sum_{x_2}\sum_{x_3} \exp\left(\sum_{i=1}^{3} \theta_i x_i + \sum_{i=1}^{3} \theta_{i,5} x_i x_5\right) \\
&\ldots \\
&= \exp\left(\theta_5 x_5 - A(\underline{\theta})\right) \prod_{i=1}^{4} \left[\sum_{x_i} \exp\left(\theta_i x_i + \theta_{i,5} x_i x_5\right)\right]
\end{aligned} \tag{8}$$

If we wanted to compute the probability that $X_5 = x_5$ and $X_4 = x_4$, we wouldn't have to sum over all the possible values of $X_4$ and we would simply have (after similar simplificaiton)

$$\mathbb{P}\left(X_4 = x_4, X_5 = x_5\right) = \exp\left(\theta_5 x_5 - A(\underline{\theta})\right)\left[\exp\left(\theta_4 x_4 + \theta_{4,5} x_4 x_5\right)\right] \prod_{i=1}^{3}\left[\sum_{x_i} \exp\left(\theta_i x_i + \theta_{i,5} x_i x_5\right)\right]. \tag{9}$$

If we are able to evaluate partial queries like the above, we are then able to evaluate more interesting *conditional* queries, such as - what is the probability that $X_4 = x_4$ *given the observation/evidence* that $X_5 = x_5$? In the usual way, we have

$$\begin{aligned}
\mathbb{P}\left(X_4 = x_4 | X_5 = x_5\right) &= \frac{\mathbb{P}\left(X_4 = x_4, X_5 = x_5\right)}{\mathbb{P}\left(X_5 = x_5\right)} \\
&= \frac{\exp\left(\theta_5 x_5 - A(\underline{\theta})\right)\left[\exp\left(\theta_4 x_4 + \theta_{4,5} x_4 x_5\right)\right] \prod_{i=1}^{3}\left[\sum_{x_i} \exp\left(\theta_i x_i + \theta_{i,5} x_i x_5\right)\right]}{\exp\left(\theta_5 x_5 - A(\underline{\theta})\right) \prod_{i=1}^{4}\left[\sum_{x_i} \exp\left(\theta_i x_i + \theta_{i,5} x_i x_5\right)\right]} \\
&= \frac{\left[\exp\left(\theta_4 x_4 + \theta_{4,5} x_4 x_5\right)\right]}{\left[\sum_{x_4'} \exp\left(\theta_4 x_4' + \theta_{4,5} x_4' x_5\right)\right]}
\end{aligned} \tag{10}$$

Hence we see that the ability to evaluate queries and partial queries allows us to compute and answer a lot of interesting inferential questions. But these kinds of evaluations can be quite hard - evaluating the marginalization sums, even for modest numbers of variables, can be quite difficult and require exponentially many terms. The above example illustrates through that these summations can frequently be simplified, for instance factoring into a number of individual sums as in the above case. In the rest of this section, we consider how to algorithmically formalize what was done in the above example, and generalize it to more interesting graphs.

## 3.1 Variable Elimination on Trees

Consider the case where the dependency graph is a tree $T$. The general case is a straight generalization of the tree case, so it is important to nail this part down first.

A **query** is a partial assignment of variables. Let $Q$ be a set of variable (indices) and $\underline{X}_Q = \underline{x}_Q$ is an assignment of values to only the variables mentioned in $Q$. The hidden or latent variables are taken to be everything else, $Q^c$. Naively, the probability of a query is given by the marginalizing sum,

$$\mathbb{P}\left(\underline{X}_Q = \underline{x}_Q\right) = \sum_{\underline{x}_{Q^c}} \exp\left(\sum_C \theta_C \Phi_C(\underline{x}_Q, \underline{x}_{Q^c}) - A(\underline{\theta})\right). \tag{11}$$

In the case of a tree, this is particularly simple, boiling down to (taking the potential functions as products by assumption)

$$\mathbb{P}\left(\underline{X}_Q = \underline{x}_Q\right) = \sum_{\underline{x}_{Q^c}} \exp\left(\sum_i \theta_i x_i + \sum_{i \sim j} \theta_{i,j} x_i x_j - A(\underline{\theta})\right). \tag{12}$$

Suppose $i'$ is a variable that is not assigned in the query, and is a leaf node in the dependency graph. In that case, the above summation can be factored, in much the same way as in the previous section. Suppose that $j'$ is the parent of $i'$ in the dependency graph:

$$\mathbb{P}\left(\underline{X}_Q = \underline{x}_Q\right) = \sum_{\underline{x}_{Q^c \setminus \{i'\}}} \exp\left(\sum_{i \neq i'} \theta_i x_i + \sum_{(i \sim j) \neq (i' \sim j')} \theta_{i,j} x_i x_j - A(\underline{\theta})\right) \sum_{x_{i'}} \exp\left(\theta_{i'} x_{i'} + \theta_{i',j'} x_{i'} x_{j'}\right). \tag{13}$$

It is convenient to introduce a 'variable elimination function' that describes this last sum:

$$m_{j'}^{i'}(x_{j'}) = \sum_{x_{i'}} \exp\left(\theta_{i'} x_{i'} + \theta_{i',j'} x_{i'} x_{j'}\right). \tag{14}$$

This function essentially eliminates the variable $X_{i'}$ from consideration, rolling the weight or likelihood of all the possible values of $X_{i'}$ into a single quantity, a function dependent on the value of $X_{j'}$. We then have

$$\mathbb{P}\left(\underline{X}_Q = \underline{x}_Q\right) = \sum_{\underline{x}_{Q^c \setminus \{i'\}}} \exp\left(\sum_{i \neq i'} \theta_i x_i + \sum_{(i \sim j) \neq (i' \sim j')} \theta_{i,j} x_i, x_j - A(\underline{\theta})\right) m_{j'}^{i'}(x_{j'}), \tag{15}$$

and we see that we have eliminated one of the variables from the sum to compute - the variable elimination function for $i'$ could be computed efficiently in advanced, and simply referenced for each term needed in the summation rather than computing the sum for $x_{i'}$ over and over. We saw in the previous case that in an extreme example, the overall sum factored into a product of variable elimination functions given $X_5$.

For any node $i$ with 'parent' $j$, the 'variable elimination' function for $i$ given $j$ summarizes *everything* that happens down that branch of the tree moving from $j$ to $i$ - potentially the results of other variable eliminations as well. In general, we have the following result:

$$m_j^i(x_j) = \begin{cases} \sum_{x_i} \exp\left(\theta_i x_i + \theta_{i,j} x_i x_j\right) \prod_{(i \sim j'), j' \neq j} m_i^{j'}(x_i) & \text{if } i \notin Q \\ \exp\left(\theta_i x_i + \theta_{i,j} x_i x_j\right) \prod_{(i \sim j'), j' \neq j} m_i^{j'}(x_i) & \text{if } i \in Q. \end{cases} \tag{16}$$

Note in the above, we are specifying that if a variable *is specified in the query*, we do not need to marginalize over it or consider all its possible values.

For a given query $Q$ with $i \in Q$ (as a sort of root of the variable tree), we can then specify

$$\mathbb{P}\left(\underline{X}_Q = \underline{x}_Q\right) = \exp\left(\theta_i x_i - A(\underline{\theta})\right) \prod_{j \sim i} m_i^j(x_i). \tag{17}$$

This gives us a very nice recursive formulation for computing probabilities, based on the query and on the tree. While a naive recursive evaluation would start at the selected root and work outwards, it may be useful (from a sort of dynamic programming perspective) to instead start at the leaves of the tree and work inwards, evaluating the variable elimination functions as you go. This is generally a very efficient approach, giving a complexity like $O(k^2)$ depending on the complexity of the tree and the size of the query. This algorithm is sometimes referred to as belief propagation as well.

## 3.2 General Variable Elimination: The Junction Tree Algorithm

One of the things that makes the previous algorithm so efficient and easy to implement is that it is applicable only to trees - every variable only has a single other variable that it depends on (looking back towards the root). But in more complicated graphs with cycles, a variable may have multiple 'parents', and therefore multiple variables that may factor into its elimination.

Consider, as an example, four variables arranged in a square. The overall distribution is specified by

$$\mathbb{P}\left(\underline{X} = \underline{x}\right) = \exp\left(\sum_{i=1}^{4} \theta_i x_i + \theta_{1,2} x_1 x_2 + \theta_{2,3} x_2 x_3 + \theta_{3,4} x_3 x_4 + \theta_{4,1} x_4 x_1 - A(\underline{\theta})\right). \tag{18}$$

Consider computing the probability that $X_1 = x_1$. In that case, we have

$$\mathbb{P}\left(X_1 = x_1\right) = \sum_{x_2, x_3, x_4} \exp\left(\sum_{i=1}^{4} \theta_i x_i + \theta_{1,2} x_1 x_2 + \theta_{2,3} x_2 x_3 + \theta_{3,4} x_3 x_4 + \theta_{4,1} x_4 x_1 - A(\underline{\theta})\right). \tag{19}$$

If we wanted to eliminate a variable, for instance $X_4$, in the way done in the previous section, that would require manipulating the sum in the following way:

$$\mathbb{P}\left(X_1 = x_1\right) = \sum_{x_2,x_3} \exp\left(\sum_{i=1}^{3} \theta_i x_i + \theta_{1,2}x_1x_2 + \theta_{2,3}x_2x_3 - A(\underline{\theta})\right) \sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{3,4}x_3x_4 + \theta_{4,1}x_4x_1\right). \tag{20}$$

Hence we can't eliminate $X_4$ in quite the same way - it implicitly depends on both $X_1$ and $X_3$. We could introduce a generalized variable elimination function, however,

$$m^4_{1,3}(x_1, x_3) = \sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{3,4}x_3x_4 + \theta_{4,1}x_4x_1\right), \tag{21}$$

in which case we have

$$\mathbb{P}\left(X_1 = x_1\right) = \sum_{x_2,x_3} \exp\left(\sum_{i=1}^{3} \theta_i x_i + \theta_{1,2}x_1x_2 + \theta_{2,3}x_2x_3 - A(\underline{\theta})\right) m^4_{1,3}(x_1, x_3). \tag{22}$$

If we then wanted to eliminate $X_3$:

$$\mathbb{P}\left(X_1 = x_1\right) = \sum_{x_2} \exp\left(\sum_{i=1}^{2} \theta_i x_i + \theta_{1,2}x_1x_2 - A(\underline{\theta})\right) \sum_{x_3} \exp\left(\theta_3 x_3 + \theta_{2,3}x_2x_3\right) m^4_{1,3}(x_1, x_3). \tag{23}$$

Note that even though there is no explicit dependency in the original graph between $X_1$ and $X_3$, in order to accomplish variable elimination in the above, we have to consider how $X_1$ influences $X_3$ - *through their mutual influence on $X_4$!* It is as though in eliminating $X_4$, we have introduced a pseudo-dependency between $X_1$ and $X_3$. Defining the $X_3$ elimination function, we have

$$m^3_{1,2}(x_1, x_2) = \sum_{x_3} \exp\left(\theta_3 x_3 + \theta_{2,3}x_2x_3\right) m^4_{1,3}(x_1, x_3), \tag{24}$$

so that

$$\mathbb{P}\left(X_1 = x_1\right) = \sum_{x_2} \exp\left(\sum_{i=1}^{2} \theta_i x_i + \theta_{1,2}x_1x_2 - A(\underline{\theta})\right) m^3_{1,2}(x_1, x_2). \tag{25}$$

As a final step, we could consider eliminating $X_2$, which would give the variable elimination function

$$m^2_1(x_1) = \sum_{x_2} \exp\left(\theta_2 x_2 + \theta_{1,2}x_1x_2\right) m^3_{1,2}(x_1, x_2), \tag{26}$$

and

$$\mathbb{P}\left(X_1 = x_1\right) = \exp\left(\theta_1 x_1 - A(\underline{\theta})\right) m^2_1(x_1). \tag{27}$$

This suggests a sort of variable elimination by cliques. We can mimic the variable elimination of the previous section, pushing terms of the sum as far right as possible and grouping, but only if we consider elimination with respect to cliques. And these 'variable elimination cliques' may spawn as a result of other variable eliminations, as the $X_1, X_3$ edge did above.

Note that the variable elimination cliques can be identified in advance, independent of the query:

- Select a variable to eliminate. It and its immediate neighbors represent the first 'elimination clique'.
- Remove that variable, and add edges between all the neighbors of that variable.
- Repeat, recording the 'elimination clique' that results at every step. Termination occurs when one variable is left. This is the 'root' of the evaluation tree.

In this way, the elimination cliques of the above graph were given by $(1,3,4) \leftarrow (1,2,3) \leftarrow (1,2)$, with elimination functions given by

$$\begin{aligned} m_1^2(x_1) &= \sum_{x_2} \exp\left(\theta_2 x_2 + \theta_{1,2} x_1 x_2\right) m_{1,2}^3(x_1, x_2) \\ m_{1,2}^3(x_1, x_2) &= \sum_{x_3} \exp\left(\theta_3 x_3 + \theta_{2,3} x_2 x_3\right) m_{1,3}^4(x_1, x_3) \\ m_{1,3}^4(x_1, x_3) &= \sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{3,4} x_3 x_4 + \theta_{4,1} x_4 x_1\right) \end{aligned} \tag{28}$$

However, we could have also taken the following approach: eliminate $X_2$ with $X_1, X_3$, eliminate $X_4$ with $X_1, X_3$, then eliminate $X_3$ with $X_1$. This represents a different ordering of variables and elimination cliques, captured by $(1,4,3) \leftarrow (1,3) \rightarrow (1,2,3)$, and elimination functions given by

$$\begin{aligned} m_1^3(x_1) &= \sum_{x_3} \exp\left(\theta_3 x_3\right) m_{1,3}^2(x_1, x_3) m_{1,3}^4(x_1, x_3) \\ m_{1,3}^2(x_1, x_3) &= \sum_{x_2} \exp\left(\theta_2 x_2 + \theta_{1,2} x_1 x_2 + \theta_{2,3} x_2 x_3\right) \\ m_{1,3}^4(x_1, x_3) &= \sum_{x_4} \exp\left(\theta_4 x_4 + \theta_{3,4} x_3 x_4 + \theta_{4,1} x_4 x_1\right). \end{aligned} \tag{29}$$

Different orderings of variables and elimination cliques will result in different overall representations for computation. In any case, the choice of elimination cliques gives rise to a tree structure (a **Junction Tree**) and evaluation of the probability via variable elimination over this junction tree exactly mimics the previous computation for variable elimination on trees. The choice of 'best' junction tree to simplify computations is a hard one; additionally, if the graph is sufficiently dense, there may be no 'good' junction trees. The Junction Tree algorithm for probability computation is most efficient when the dependency graph is in some sense sparse.

As a general approach to evaluating these probability queries with an aim for efficiency, we present the following:

**The Junction Tree Algorithm**

- Compute the Junction Tree
  - Select a variable ordering
  - For each variable, in order, determine the variable elimination clique for that variable.
  - Generate the full set of variable elimination cliques for this ordering.
  - Reduce these cliques to the **maximal** cliques, the cliques that cover all the other cliques.
  - Build a graph on these maximal cliques, where two cliques share an edge if they share variables.
  - Weight each edge between cliques according to the number of shared variables.
  - Generate the Junction Tree as the maximum weight spanning tree of this elimination clique graph.
- To Evaluate a Query
  - From the leaves of the junction tree inwards, equip each node of the junction tree with a variable elimination function (as in the previous section).
    - At each node, eliminate variables from that clique in terms of whatever variables that node shares with its parent.
    - If a variable is specified in the query, take that to be its value.
    - If a variable is not specified in the query, marginalize over all values of that variable.
  - Perform belief propagation / variable elimination on the junction tree using these variable elimination functions, as in the previous section.

# 4 How to Learn a Dependency Graph

In general, the problem of learning dependency graphs is hard. As a general approach, we can consider a maximum likelihood approach that asks: given the data, what graph maximizes the (average log) likelihood of the specific data observed? In other words, solve

$$\max_G \ \frac{1}{m}\sum_{i=1}^{m} \ln \mathbb{P}\left(\underline{X} = \underline{x}^i | G\right). \tag{30}$$

The problem, as we have observed, is that computing the probability of a given variable assignment over a graph can be difficult. It is dramatically simpler in the case of trees, however - the likelihood over a tree actually has a very nice, compact form.

Let $T$ be a tree on the variables of $\underline{X}$; let $t_0$ be an arbitrarily chosen root node of $T$, and let $\pi_T(t)$ be the 'parent' of variable $t$ in the tree $T$, moving out from the root. In that case,

$$\mathbb{P}\left(\underline{X} = \underline{x} | T\right) = \mathbb{P}\left(X_{t_0} = x_{t_0}\right) \prod_{t \neq t_0} \mathbb{P}\left(X_t = x_t | X_{\pi_T(t)} = x_{\pi_T(t)}\right), \tag{31}$$

so that

$$\ln \mathbb{P}\left(\underline{X} = \underline{x} | T\right) = \ln \mathbb{P}\left(X_{t_0} = x_{t_0}\right) + \sum_{t \neq t_0, s = \pi_T(t)} \ln \mathbb{P}\left(X_t = x_t | X_s = x_s\right). \tag{32}$$

We can then rearrange this, based on the conditional probability term, and the structure of the graph. Note that we can simplify this to be

$$\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=\ln \mathbb{P}\left(X_{t_0}=x_{t_0}\right)+\sum_{t\neq t_0, s=\pi_T(t)} \ln \mathbb{P}\left(X_t=x_t, X_s=x_s\right)-\sum_{t\neq t_0, s=\pi_T(t)} \ln \mathbb{P}\left(X_s=x_s\right). \tag{33}$$

The first sum, with the probability now symmetric in $s$ and $t$, is simply the sum over all edges in $T$ once. The second sum is effectively looking at every node $s$ that is a parent (i.e., not a leaf), and summing over all the edges to its children. For the root node $s=t_0$, the number of such edges is $\deg_T(t_0)$, since the root has no parents. Every other node as a single parent, so that the number of such edges is $\deg_T(s)-1$.

The number of such edges, given that $T$ is a tree, is going to be $\deg_T(s)-1$. Hence the above becomes

$$\begin{aligned}\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=&\ln \mathbb{P}\left(X_{t_0}=x_{t_0}\right)+\sum_{(t\sim s)\in T} \ln \mathbb{P}\left(X_t=x_t, X_s=x_s\right)\\ &-\sum_{s \text{ not a leaf in } T, s\neq t_0} \left(\deg_T(s)-1\right)\ln \mathbb{P}\left(X_s=x_s\right)\\ &-\deg_T(t_0)\ln \mathbb{P}\left(X_{t_0}=x_{t_0}\right).\end{aligned} \tag{34}$$

However, note that if $s$ is a leaf in $T$, then $\deg_T(s)-1=0$, so we can include these terms anyway:

$$\begin{aligned}\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=&\ln \mathbb{P}\left(X_{t_0}=x_{t_0}\right)+\sum_{(t\sim s)\in T} \ln \mathbb{P}\left(X_t=x_t, X_s=x_s\right)\\ &-\sum_{s\neq t_0} \left(\deg_T(s)-1\right)\ln \mathbb{P}\left(X_s=x_s\right)\\ &-\deg_T(t_0)\ln \mathbb{P}\left(X_{t_0}=x_{t_0}\right).\end{aligned} \tag{35}$$

Regrouping gives us the following:

$$\begin{aligned}\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=&\sum_{(t\sim s)\in T} \ln \mathbb{P}\left(X_t=x_t, X_s=x_s\right)\\ &-\sum_{s}\deg_T(s)\ln \mathbb{P}\left(X_s=x_s\right)+\sum_{s}\ln \mathbb{P}\left(X_s=x_s\right).\end{aligned} \tag{36}$$

Note one promising aspect of the above formula - it is actually independent of the initial choice of 'root' in the tree - any choice of root should give rise to the exact same computed probability. But we can simplify this further still. We can replace the second sum we a sum over all the edges as well - summing over all the edges, if we add up the probability associated with each vertex in the edge, the final result will count each probability multiplied by the vertex degree.

$$\begin{aligned}\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=&\sum_{t\sim s} \ln \mathbb{P}\left(X_t=x_t, X_s=x_s\right)\\ &-\sum_{s\sim t}\left(\ln \mathbb{P}\left(X_s=x_s\right)+\ln \mathbb{P}\left(X_t=x_t\right)\right)+\sum_{s}\ln \mathbb{P}\left(X_s=x_s\right).\end{aligned} \tag{37}$$

Bringing this all together, we have

$$\ln \mathbb{P}\left(\underline{X}=\underline{x}|T\right)=\sum_{t\sim s}\ln\left(\frac{\mathbb{P}\left(X_t=x_t, X_s=x_s\right)}{\mathbb{P}\left(X_s=x_s\right)\mathbb{P}\left(X_t=x_t\right)}\right)+\sum_{s}\ln \mathbb{P}\left(X_s=x_s\right). \tag{38}$$

Therefore, for a given tree, the (average) joint likelihood of that tree can be expressed as

$$\frac{1}{m}\sum_{i=1}^{m}\ln \mathbb{P}\left(\underline{X}=\underline{x}^i|T\right)=\sum_{t\sim s}\frac{1}{m}\sum_{i=1}^{m}\ln\left(\frac{\mathbb{P}\left(X_t=x_t^i, X_s=x_s^i\right)}{\mathbb{P}\left(X_s=x_s^i\right)\mathbb{P}\left(X_t=x_t^i\right)}\right)+\sum_{s}\frac{1}{m}\sum_{i=1}^{m}\ln \mathbb{P}\left(X_s=x_s^i\right). \tag{39}$$

It is convenient to define the 'score' of an edge to be

$$\text{score}(s \sim t) = \frac{1}{m}\sum_{i=1}^{m} \ln\left(\frac{\mathbb{P}\left(X_t = x_t^i, X_s = x_s^i\right)}{\mathbb{P}\left(X_s = x_s^i\right)\mathbb{P}\left(X_t = x_t^i\right)}\right). \tag{40}$$

Observing that the second term in the average log likelihood is actually independent of the tree itself, we can write the following:

$$T^* = \text{argmax}_T \ \frac{1}{m}\sum_{i=1}^{m} \ln \mathbb{P}\left(\underline{X} = \underline{x}^i | T\right) = \text{argmax}_T \sum_{(t\sim s)\in T} \text{score}(t \sim s). \tag{41}$$

The above equation illustrates the basis of the Chow-Liu algorithm: consider the complete graph on all variables, with the edge $s \sim t$ weighted according to $\text{score}(s \sim t)$. The above suggests that the most likely tree is in fact the **maximum weight spanning tree** on this weighted complete graph. Any max spanning tree algorithm may be applied here then to recover the tree in efficient time.

However, one glaring flaw remains: we do not actually know what the scores are, because we don't know what the true underlying probabilities are. The solution, in the usual way, is to approximate these from data. Let $\#_j\{x\}$ be the number of times that $X_j$ was observed to have value $x$ in the data set. The Chow-Liu algorithm proceeds as follows:

**The Chow Liu Tree Algorithm:**

- Construct the complete graph on all the variables $\underline{X} = X_1, \ldots, X_k$.
- Weight each edge $s \sim t$ according to the estimated mutual information.

$$\frac{1}{m}\sum_{i=1}^{m} \ln\left(\frac{\#_{t,s}\{x_t^i, x_s^i\}/m}{\left(\#_s\{x_s^i\}/m\right)\left(\#_t\{x_t^i\}/m\right)}\right) \tag{42}$$

- Return $T^*$, the maximum weight spanning tree on this graph.

Another approach, **Graphical Lasso** gives a more general approach of learning the dependencies and parameters jointly, but the Chow-Liu algorithm is nice as it comes with a number of theoretical guarantees, for instance the ability to recover the 'true' dependency tree given enough samples.

# 5 How to Learn Graphical Model Parameters

Given a set of data points $\underline{x}^1, \ldots, \underline{x}^m$, and a proposed dependency graph ( and choice of potential function ), how can we learn the parameters or clique weights $\theta_C$? Here again we take a maximum likelihood approach, that is to find $\underline{\theta}$ to solve

$$\max_{\underline{\theta}} \ \frac{1}{m}\sum_{i=1}^{m} \ln \mathbb{P}\left(\underline{X} = \underline{x}^i | \underline{\theta}\right). \tag{43}$$

Utilizing the previous representation theorem for MRFs, we have

$$\max_{\underline{\theta}} \ \frac{1}{m}\sum_{i=1}^{m} \ln \mathbb{P}\left(\underline{X} = \underline{x}^i | \underline{\theta}\right) = \max_{\underline{\theta}} \ \frac{1}{m}\sum_{i=1}^{m}\left[\sum_C \theta_C \Phi_C\left(\underline{x}^i\right) - A(\underline{\theta})\right] = \max_{\underline{\theta}} \ \sum_C \theta_C \frac{1}{m}\left[\sum_{i=1}^{m} \Phi_C\left(\underline{x}^i\right)\right] - A(\underline{\theta}). \tag{44}$$

Defining the log loss function

$$L(\underline{\theta}) = \sum_C \theta_C \frac{1}{m}\left[\sum_{i=1}^{m} \Phi_C\left(\underline{x}^i\right)\right] - A(\underline{\theta}), \tag{45}$$

we could consider attempting to maximize $L$ with respect to $\underline{\theta}$ in the usual way, for instance using gradient ascent. This requires being able to compute the derivatives of $L$, however. Note then that for any given clique $C$, we have that

$$\frac{\partial L}{\partial \theta_C}(\underline{\theta}) = \frac{1}{m}\left[\sum_{i=1}^{m} \Phi_C\left(\underline{x}^i\right)\right] - \frac{\partial A}{\partial \theta_C}(\underline{\theta}). \tag{46}$$

To complete this derivation (and be able to do gradient descent), we need to compute the derivatives of $A$. Note that

$$A(\underline{\theta}) = \ln\left(\sum_{\underline{x}} \exp\left(\sum_{C} \theta_C \Phi_C(\underline{x})\right)\right), \tag{47}$$

so that

$$\begin{aligned}
\frac{\partial A}{\partial \theta_C} &= \frac{\partial}{\partial \theta_C}\left[\sum_{\underline{x}} \exp\left(\sum_{C'} \theta_{C'} \Phi_{C'}(\underline{x})\right)\right] \frac{1}{\sum_{\underline{x}} \exp\left(\sum_{C'} \theta_{C'} \Phi_{C'}(\underline{x})\right)} \\
&= \left[\sum_{\underline{x}} \Phi_C(\underline{x}) \exp\left(\sum_{C'} \theta_{C'} \Phi_{C'}(\underline{x})\right)\right] \frac{1}{e^{A(\underline{\theta})}} \\
&= \sum_{\underline{x}} \Phi_C(\underline{x}) \exp\left(\sum_{C'} \theta_{C'} \Phi_{C'}(\underline{x}) - A(\underline{\theta})\right) \\
&= \mathbb{E}_{\underline{\theta}}\left[\Phi_C(\underline{X})\right].
\end{aligned} \tag{48}$$

Hence we have

$$\frac{\partial L}{\partial \theta_C}(\underline{\theta}) = \frac{1}{m}\left[\sum_{i=1}^{m} \Phi_C\left(\underline{x}^i\right)\right] - \mathbb{E}_{\underline{\theta}}\left[\Phi_C(\underline{x})\right]. \tag{49}$$

This suggests the following cute interpretation: that the maximum likelihood $\underline{\theta}$ occur (where the gradient of $L$ is zero) when the sample means of all the potential functions are *as closed as possible* to the theoretical mean of all the potential functions.

Theoretically, gradient ascent could be applied here, *as long as it is easy to calculate the* $\mathbb{E}_{\underline{\theta}}\left[\Phi_C(\underline{X})\right]$. Computing this expectation is an act of inference, and generally we see a tight coupling between learning and inference. Unfortunately this is generally difficult as it would require summing over exponentially many terms.

Other **variational methods** attempt to estimate $A(\underline{\theta})$ and maximize $L(\underline{\theta})$ based on that.

## 5.1 Graphical Lasso: Learning Parameters and Dependencies Jointly

This notion of learning parameters via maximum likelihood estimation (exact or approximate) suggests an alternate way of learning the dependency graph. Recall in regression that **Lasso** was used to identify features that were likely irrelevant, and eliminate them by setting the corresponding weight to zero. An analogy can be drawn here towards identifying 'irrelevant' dependencies from the data, and setting the corresponding weight or coupling to zero. An algorithm for this might initially assume a complete graph between all the variables, and then try to solve for the maximum likelihood $\theta_C$ values - with an additional Lasso regularizing term that pushes the $\theta_C$ toward zero. Any cliques that yield $\theta_C^* = 0$ in the MLE solution, those cliques could be 'killed' by the appropriate deletion of edges in the graph.

The difficulty then is that starting with a complete graph and considering all possible cliques is then a semi-impossible computational task: in a graph with $k$ variables, there are going to be $2^k$ possible cliques to consider. This suggests

an alternative approach: start by assuming that the weights on all cliques of size $> 2$ are zero. That is, for a complete graph $G$, the likelihood is given by

$$\mathbb{P}\left(\underline{X} = \underline{x}\right) \propto \exp\left(\sum_i \theta_i \Phi_i(x_i) + \sum_{(i \sim j) \in G} \theta_{i,j} \Phi_{i,j}(x_i, x_j)\right). \tag{50}$$

This gives a more economical $k + \binom{k}{2} = O(k * k)$ variables to solve for, each variable uniquely identifying the weight of a single edge in $G$ (or importance of a vertex in $G$). Solving for the MLEs for these $\underline{\theta}$, an additional Lasso regularizer will force some of these towards $\theta_{i,j} = 0$, at which point these edges could be pruned from the graph.

Now, once these edges have been pruned from the graph, the resulting graph may or may not have higher order cliques in it. But if it does, then having identified a useful dependency graph, we could re-approach learning the parameters on this graph (with all cliques), in much the same way that elastic net regression first learns relevant features using Lasso, then finds the best fit on those relevant features.

# 6 How to Learn with Missing Variables: EM Algorithm

Suppose that some of the data is actually missing. We usually try to separate data into 'hidden' or 'latent' and observed variables, $\underline{Z}$ and $\underline{X}$ respectively. A good example of this would be a hidden Markov model, where we observe the observation variables $Y$, but are unable to observe the hidden state variables $X$. In NLP, we might have a sequence of words or text representing the observed variables, but some hidden or underlying topic or sentiment that gave rise to those variables.

In this case, we may still want to learn a model (find a model that makes the observed data as likely as possible), but the problem we run into is this: if we only have values for the observed data, we need to marginalize or consider *all possible values* for the unobserved data. A joint model might specify $P(\underline{X}, \underline{Z})$, how these variables relate to each other, but we want to maximize the *marginal likelihood of the observed data*. That is, given a distribution $P$ parameterized by some parameters $\underline{\theta}$, we want to find $\underline{\theta}$ to maximize

$$\max_{\underline{\theta}} \sum_{i=1}^{m} \ln P_{\underline{\theta}}(\underline{X} = \underline{x}^i), \tag{51}$$

and therefore equivalently solve

$$\max_{\underline{\theta}} \sum_{i=1}^{m} \ln \left[\sum_{\underline{z}^i} P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z} = \underline{z}^i)\right]. \tag{52}$$

Again, since we don't know the values of the hidden variables, we must marginalize with respect to all possible values of the marginal variables.

But maximization problems like this are typically incredibly hard to solve - part of this is the fact that the log of a sum is incredibly difficult to work with, mathematically. The **EM Algorithm** proposes an alternate approach to trying to maximize the above sum that potentially avoids computing these sums over all possible values of the hidden variables. Essentially, while we may not know the true values of the hidden variables, we may know something about their distribution and the values they are likely to have.

The EM (or Expectation-Maximization) Algorithm proceeds in the following way: Given a current model $\underline{\theta}_{\text{old}}$, it attempts to build a new model $\underline{\theta}_{\text{new}}$ with a higher log likelihood of the observed data. In steps:

- **The Expectation Step:** For each $i$, compute *the conditional distribution of $\underline{Z}^i$ given the current model, and the value of the observed variables*:

$$q_i\left(\underline{z}\right) = P_{\underline{\theta}_{\text{old}}}\left(\underline{Z} = \underline{z} | \underline{X} = \underline{x}^i\right) \tag{53}$$

This represents **a Bayesian Update of the old model, conditioned on the observed data.** We then want to build a new model that accounts for this Bayesian Update.

- **The Maximization Step:** Find a new model $\underline{\theta}_{\text{new}}$ that maximizes the *expected log likelihood* of the data, with respect to this distribution over the hidden variables. That is, solve

$$\max_{\underline{\theta}} \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[ \ln P_{\underline{\theta}} \left( \underline{X} = \underline{x}^i, \underline{Z} \right) \right]. \tag{54}$$

  This new model maximizes the (expected) likelihood, factoring in the Bayesian Update of the previous step.

- Repeat, sequentially improving models, until convergence.

Two questions remain at this point, roughly:

- Why does this algorithm work?
- How can we use this algorithm?

We can answer the first one with a short proof, and we can answer the second one by example. Recall that what we want to do is maximize the average log likelihood, i.e., find $\underline{\theta}$ to maximize

$$L(\underline{\theta}) = \frac{1}{m} \sum_{i=1}^{m} \ln P_{\underline{\theta}}(\underline{X} = \underline{x}^i) = \frac{1}{m} \sum_{i=1}^{m} \ln \left[ \sum_{\underline{z}^i} P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z} = \underline{z}^i) \right] \tag{55}$$

It is convenient to modify this by introducing a secondary or intermediate distribution $q_i(\underline{z})$ in the following way:

$$\begin{aligned} L(\underline{\theta}) &= \frac{1}{m} \sum_{i=1}^{m} \ln \left[ \sum_{\underline{z}^i} q_i(\underline{z}^i) \frac{P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z} = \underline{z}^i)}{q_i(\underline{z}^i)} \right] \\ &= \frac{1}{m} \sum_{i=1}^{m} \ln \left[ \mathbb{E}_{q_i} \left[ \frac{P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z})}{q_i(\underline{Z})} \right] \right] \end{aligned} \tag{56}$$

This is equivalent to the original formulation, so we have not really gained anything. However, at this point we can apply the following result: **Jensen's Inequality** tells us that $\ln \mathbb{E}[f(Z)] \geq \mathbb{E}[\ln f(Z)]$. Applying this here, we have that

$$L(\underline{\theta}) \geq \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[ \ln \left[ \frac{P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z})}{q_i(\underline{Z})} \right] \right], \tag{57}$$

or

$$L(\underline{\theta}) \geq \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[ \ln \left[ P_{\underline{\theta}}(\underline{X} = \underline{x}^i, \underline{Z}) \right] \right] - \frac{1}{m} \sum_{i=1}^{m} \mathbb{E}_{q_i} \left[ \ln \left[ q_i(\underline{Z}) \right] \right], \tag{58}$$

Note that in the above, the right hand term is actually independent of $\underline{\theta}$. That is, if we want to try to find $\underline{\theta}$ to increase $L(\underline{\theta})$, we could consider instead trying to find $\underline{\theta}$ to maximize this expected log likelihood term, given the distributions $\{q_i\}$. This justifies somewhat the maximization step of the EM algorithm.

But which $q_i$? Intuitively we would like this 'intermediate' latent distribution to match the 'true' distribution, i.e.,

$q_i(\underline{z}) = P_{\underline{\theta}}(\underline{Z} = \underline{z}|\underline{X}^i = \underline{x})$. Note that we could additionally have written the initial Jensen Bound above as

$$\begin{aligned}
L(\underline{\theta}) &\geq \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{q_i}\left[\ln\left[\frac{P_{\underline{\theta}}(\underline{X}^i = \underline{x}^i)P_{\underline{\theta}}(\underline{Z}|\underline{X} = \underline{x}^i)}{q_i(\underline{Z})}\right]\right], \\
&= \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{q_i}\left[\ln\left[P_{\underline{\theta}}(\underline{X}^i = \underline{x}^i)\right]\right] + \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{q_i}\left[\ln\left[\frac{P_{\underline{\theta}}(\underline{Z}|\underline{X} = \underline{x}^i)}{q_i(\underline{Z})}\right]\right], \\
&= \frac{1}{m}\sum_{i=1}^{m}\ln\left[P_{\underline{\theta}}(\underline{X}^i = \underline{x}^i)\right] - \frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{q_i}\left[\ln\left[\frac{q_i(\underline{Z})}{P_{\underline{\theta}}(\underline{Z}|\underline{X} = \underline{x}^i)}\right]\right], \\
&= \frac{1}{m}\sum_{i=1}^{m}\ln\left[P_{\underline{\theta}}(\underline{X}^i = \underline{x}^i)\right] - \frac{1}{m}\sum_{i=1}^{m}\mathrm{KL}\left(q_i||P_{\underline{\theta}}(\cdot|\underline{X} = \underline{x}^i)\right) \\
&= L(\underline{\theta}) - \frac{1}{m}\sum_{i=1}^{m}\mathrm{KL}\left(q_i||P_{\underline{\theta}}(\cdot|\underline{X} = \underline{x}^i)\right),
\end{aligned} \tag{59}$$

where in the above, the KL-Divergence measures how different the distribution $q_i$ is from the conditional distribution for the hidden variables. The KL divergence is always positive, and the greater the difference between the two distributions the greater the KL divergence. It effectively acts as a measure of similarity.

We see from this that the more different $q_i$ is from the conditional distribution given the current model, the greater the discrepancy is in the use of the Jensen inequality - to get a tight bound on this inequality, we want $q_i$ to be as close to the conditional distribution as possible. This somewhat justifies the Expectation step of the EM algorithm.

The EM algorithm effectively separates these two steps: finding the 'best' conditional distribution for the hidden variables, freezing it, then finding the new 'best' model given this conditional distribution. Repeat until convergence.

## 6.1 Gaussian Mixture Modeling

We are frequently confronted with the problem of fitting a model to data. This is often of the form mapping input data to some corresponding output data, but is also often of the form of selecting a density function to try to describe the distribution of some data. Given a sequence of data $x_1, x_2, x_3, \ldots, x_N$, this frequently takes the form of constructing a maximum likelihood estimator, i.e., the density function that maximizes the likelihood of the observed data. Given some family of densities $\mathcal{F}$, we have

$$f^* = \mathrm{argmax}_{f\in\mathcal{F}}\prod_{i=1}^{N} f(x_i), \tag{60}$$

though this is frequently formulated in terms of maximizing the log of the likelihood for numerical and theoretical reasons:

$$f^* = \mathrm{argmax}_{f\in\mathcal{F}}\sum_{i=1}^{N}\ln f(x_i). \tag{61}$$

In the case that $\mathcal{F}$ is a parameterized family of distributions, this becomes simply a matter of finding the optimal parameter values to maximize this function.

However, this is not always an easy task. Suppose that $x_1, \ldots, x_n$ were generated according to a *Gaussian mixture model*, multiple gaussian bumps superimposed on each other with some weights, e.g., in a simple case, with probability $p$, $X$ is generated according to a normal distribution with mean $\mu_1$ and variance 1; with probability $1-p$, $X$ is generated according to a normal distribution with mean $\mu_2$ and variance 1. This model has three parameters, $p, \mu_1, \mu_2$, and for any choice of parameters has a density function of the form

$$f(x) = p\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(x-\mu_1)^2\right) + (1-p)\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(x-\mu_2)^2\right). \tag{62}$$

To solve for the maximum likelihood estimators would require solving

$$p^*, \mu_1^*, \mu_2^* = \text{argmax}_{p,\mu_1,\mu_2} \sum_{i=1}^{N} \ln \left[ p \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_1)^2 \right) + (1-p) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_2)^2 \right) \right]. \tag{63}$$

Solving for this minimizer can be quite difficult given the highly non-linear nature of the function and sum over some (potentially large) number of terms. So what can be done?

### 6.1.1 Introducing Latent or Class Variables

It is convenient to model the data as not just been generated by a density function, but in fact generated by a process: consider introducing a latent variable $Z$ such that $Z = 1$ with probability $p$, and $Z = 0$ with probability $1 - p$; then define the distribution of $X$ *conditionally* in terms of $Z$, in particular that if $Z = 1$, then $X$ is normal with unit variance and mean $\mu_1$, and if $Z = 0$, then $X$ is normal with unit variance and mean $\mu_2$. This is frequently discussed in terms of classes - everything in class 1 is distributed according to a normal distribution with mean $\mu_1$, and everything in class 0 is distributed with a normal distribution with mean $\mu_2$; the class a given data point will be in is determined with probability $p$.

Without the introduction of these latent variables, the likelihood of a given set of parameters given the data would be given as this:

$$\text{lik}(p, \mu_1, \mu_2 | \underline{x}) = \prod_{i=1}^{N} \left[ p \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x - \mu_1)^2 \right) + (1-p) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x - \mu_2)^2 \right) \right]. \tag{64}$$

As seen previously, attempting to maximize this likelihood leads to very strange non-linear functions in terms of $p, \mu_1, \mu_2$.

However, suppose that the values of the corresponding latent variables were known: in particular, we are given that $X_i = x_i$, suppose we additionally knew that $Z_i = z_i$. In that case, we could express the likelihood of a given model in the following way:

$$\text{lik}(p, \mu_1, \mu_2 | \underline{x}, \underline{z}) = \prod_{i=1}^{N} \left[ p \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_1)^2 \right) \right]^{z_i} \left[ (1-p) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_2)^2 \right) \right]^{1-z_i}. \tag{65}$$

In this case, the likelihood becomes a pure product of all the likelihood of the individual variable values. We can simplify it further in the following way:

$$\begin{aligned}
&\ln \text{lik}(p, \mu_1, \mu_2 | \underline{x}, \underline{z}) \\
&= \sum_{i=1}^{N} \ln \left( \left[ p \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_1)^2 \right) \right]^{z_i} \left[ (1-p) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_2)^2 \right) \right]^{1-z_i} \right) \\
&= \sum_{i=1}^{N} z_i \ln \left( p \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_1)^2 \right) \right) + \sum_{i=1}^{N} (1 - z_i) \ln \left( (1-p) \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{1}{2}(x_i - \mu_2)^2 \right) \right) \\
&= \sum_{i=1}^{N} z_i \ln \left( p \frac{1}{\sqrt{2\pi}} \right) - \frac{1}{2} \sum_{i=1}^{N} z_i (x_i - \mu_1)^2 + \sum_{i=1}^{N} (1 - z_i) \ln \left( (1-p) \frac{1}{\sqrt{2\pi}} \right) - \frac{1}{2} \sum_{i=1}^{N} (1 - z_i)(x_i - \mu_2)^2
\end{aligned} \tag{66}$$

At this point, *if the latent variable values $z_i$ were known*, it would actually be relatively straightforward to solve for

the maximum likelihood estimators for $p, \mu_1, \mu_2$. In particular, note that:

$$\begin{aligned}
\frac{\partial}{\partial p} \ln \text{lik}(p, \mu_1, \mu_2 | \underline{x}, \underline{z}) &= \frac{1}{p} \sum_{i=1}^{N} z_i - \frac{1}{1-p} \sum_{i=1}^{N} (1 - z_i) = 0 \\
\frac{\partial}{\partial \mu_1} \ln \text{lik}(p, \mu_1, \mu_2 | \underline{x}, \underline{z}) &= \sum_{i=1}^{N} z_i (x_i - \mu_1) = 0 \\
\frac{\partial}{\partial \mu_2} \ln \text{lik}(p, \mu_1, \mu_2 | \underline{x}, \underline{z}) &= \sum_{i=1}^{N} (1 - z_i)(x_i - \mu_2) = 0,
\end{aligned} \tag{67}$$

which solves to yield

$$\begin{aligned}
p^* &= \frac{1}{N} \sum_{i=1}^{N} z_i \\
\mu_1^* &= \frac{\sum_{i=1}^{N} z_i x_i}{\sum_{i=1}^{N} z_i} \\
\mu_2^* &= \frac{\sum_{i=1}^{N} (1 - z_i) x_i}{\sum_{i=1}^{N} (1 - z_i)}.
\end{aligned} \tag{68}$$

Of course, these mechanics are somewhat moot as *we do not know the value of the latent variables.* It is worth noting, however, the rough interpretation of the above formulas, since $z_i = 0$ or $z_i = 1$: $p^*$, the probability of being in class 1, is simply the frequency of being in class 1 over the data. In Class 1, $\mu_1^*$ represents a weighted average of the data points - but in fact, any data point not in class 1 is given a weight of 0 - this formula for $\mu_1^*$ then is simply the sample mean of all data points in class 1; similarly, the formula for $\mu_2^2$ is simply the sample mean of all the data points in class 0.

### 6.1.2 The EM Algorithm

Given that we do not know the value of the latent variables, the above approach cannot be applied directly. However, for a particular model, we do know (or can work out) the *distribution* of the latent variables. To begin with, for a specified model $(p, \mu_1, \mu_2)$, we know that for any given $i$,

$$\begin{aligned}
\mathbb{P}(Z_i = 1 | p, \mu_1, \mu_2) &= p \\
\mathbb{P}(Z_i = 0 | p, \mu_1, \mu_2) &= 1 - p.
\end{aligned} \tag{69}$$

But in fact we know slightly more - we can pull in the data that we have, essentially observations that tell us something about the underlying latent variable values. In particular, if $X_i$ was observed to have the value $x_i$, usual Bayesian updating yields

$$\begin{aligned}
\mathbb{P}(Z_i = 1 | X_i = x_i, p, \mu_1, \mu_2) &= \frac{p \exp\left(-\frac{1}{2}(x_i - \mu_1)^2\right)}{p \exp\left(-\frac{1}{2}(x_i - \mu_1)^2\right) + (1-p) \exp\left(-\frac{1}{2}(x_i - \mu_2)^2\right)} \\
\mathbb{P}(Z_i = 0 | X_i = x_i, p, \mu_1, \mu_2) &= \frac{p \exp\left(-\frac{1}{2}(x_i - \mu_1)^2\right)}{p \exp\left(-\frac{1}{2}(x_i - \mu_1)^2\right) + (1-p) \exp\left(-\frac{1}{2}(x_i - \mu_2)^2\right)}.
\end{aligned} \tag{70}$$

Instead of using the values of the latent variables to determine the MLE for $(p^*, \mu_1^*, \mu_2^*)$, as in the previous section, the EM algorithm proposes the following cycle, using the *updated distribution for the latent variables* in place of the values of the latent variables themselves:

- Take a given model $(p, \mu_1, \mu_2)$.

- Compute the posterior distribution of the latent variables $Z_1, Z_2, \ldots, Z_N$ under this model.
- Compute a new model $(p', \mu_1', \mu_2')$ by maximizing the *expected* log likelihood, under the determined distribution for the latent variables.

This process is repeated until convergence.

Effectively, this disconnects the model for the latent variables from the model for the data itself. 1) You compute an alternative, updated model for the latent variables, 2) fix it, 3) determine a better model for the data variables given the fixed latent distribution, 4) repeat.

Applying this here:

- Given a model $(p, \mu_1, \mu_2)$, let $p_i = \mathbb{P}(Z_i = 1 | X_i = x_i, p, \mu_1, \mu_2)$ as above, essentially the updated probability that $X_i$ is of class 1. Consider these $\{p_i\}$ as fixed, specifying the posterior distribution of the latent variables.
- Consider not the (log) likelihood itself, but the expected value of it, under this distribution for the $Z_i$ variables:

$$\begin{aligned}
&\mathbb{E}\left[\ln \text{lik}(p', \mu_1', \mu_2' | \underline{x}, \underline{Z})\right] \\
&= \sum_{i=1}^{N} \mathbb{E}[Z_i] \ln\left(p' \frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}\sum_{i=1}^{N} \mathbb{E}[Z_i](x_i - \mu_1')^2 + \sum_{i=1}^{N} \mathbb{E}[1 - Z_i] \ln\left((1-p') \frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}\sum_{i=1}^{N} \mathbb{E}[1 - Z_i](x_i - \mu_2')^2 \\
&= \sum_{i=1}^{N} p_i \ln\left(p' \frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}\sum_{i=1}^{N} p_i (x_i - \mu_1')^2 + \sum_{i=1}^{N} (1 - p_i) \ln\left((1-p') \frac{1}{\sqrt{2\pi}}\right) - \frac{1}{2}\sum_{i=1}^{N} (1 - p_i)(x_i - \mu_2')^2
\end{aligned} \tag{71}$$

- Solve for the maximizing $p', \mu_1', \mu_2'$ in the above, i.e., solve:

$$\begin{aligned}
\frac{\partial}{\partial p'} \mathbb{E}\left[\ln \text{lik}(p', \mu_1', \mu_2' | \underline{x}, \underline{Z})\right] &= \frac{1}{p'} \sum_{i=1}^{N} p_i - \frac{1}{1-p'} \sum_{i=1}^{N} (1 - p_i) = 0 \\
\frac{\partial}{\partial \mu_1'} \mathbb{E}\left[\ln \text{lik}(p', \mu_1', \mu_2' | \underline{x}, \underline{Z})\right] &= \sum_{i=1}^{N} p_i (x_i - \mu_1') = 0 \\
\frac{\partial}{\partial \mu_2'} \mathbb{E}\left[\ln \text{lik}(p', \mu_1', \mu_2' | \underline{x}, \underline{Z})\right] &= \sum_{i=1}^{N} (1 - p_i)(x_i - \mu_2') = 0,
\end{aligned} \tag{72}$$

  which solves to

$$\begin{aligned}
(p')^* &= \frac{1}{N} \sum_{i=1}^{N} p_i \\
(\mu_1')^* &= \frac{\sum_{i=1}^{N} p_i x_i}{\sum_{i=1}^{N} p_i} \\
(\mu_2')^* &= \frac{\sum_{i=1}^{N} (1 - p_i) x_i}{\sum_{i=1}^{N} (1 - p_i)}.
\end{aligned} \tag{73}$$

  Note, this is the seemingly obvious extension of the previous solution in the case of known $z_i$ values, 'softened' in some sense based on knowing the distribution rather than the values themselves. They still have the nice interpretation as the 'frequency' of being in class 1, the 'weighted average' of the data in class 1 (weighted, in this case, by the likelihood of being in class 1), and the weighted average of the data in class 0.

- At this point, having built a new model $((p')^*, (\mu_1')^*, (\mu_2')^*)$, replace the old model, and repeat the process until convergence.

### 6.1.3 The General Gaussian Mixture EM Solution

Consider extending the previous problem to a set of data $x_1, x_2, \ldots, x_N$ generated by a $k$-class Gaussian mixture model, where $X$ is of class $i$ with probability $p_i$, and if it is in class $i$ it has a normal distribution with mean $\mu_i$ and variance $\sigma_i^2$. Note that now to specify a model, we need to specify a vector of probabilityes $\underline{p}$, a vector of means $\underline{\mu}$, and a vector of variances, $\underline{\sigma}^2$. Again we can introduce hidden or latent variables $Z = i$ to indicate that a given $X$ belongs to class $i$. We can generalize the above approach in the following way: The likelihood, assuming the values of the latent variables are known, is given by

$$\text{lik}(\underline{p}, \underline{\mu}, \underline{\sigma}^2 | \underline{x}, \underline{z}) = \prod_{i=1}^{N} \prod_{j=1}^{k} \left[ p_j \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left( -\frac{1}{2\sigma_j^2} (x_i - \mu_j)^2 \right) \right]^{\mathbf{1}\{z_i = j\}}, \tag{74}$$

or a log likelihood of

$$\ln \text{lik}(\underline{p}, \underline{\mu}, \underline{\sigma}^2 | \underline{x}, \underline{z}) = \sum_{i=1}^{N} \sum_{j=1}^{k} \mathbf{1}\{z_i = j\} \left[ \ln\left( p_j \frac{1}{\sigma_j \sqrt{2\pi}} \right) - \frac{1}{2\sigma_j^2} (x_i - \mu_j)^2 \right] \tag{75}$$

Again, if the values of the latent or class variables were known - the above could be used quite easily to generate maximum likelihood estimators for $\underline{p}, \underline{\mu}, \underline{\sigma}^2$. But we don't know them - hence we fall back to the idea that given a model $(\underline{p}, \underline{\mu}, \underline{\sigma}^2)$, we can compute an updated, posterior distribution for the latent variables given the data. In particular,

$$q_{i,j} = \mathbb{P}\left( Z_i = j | X_i = x_i, \underline{p}, \underline{\mu}, \underline{\sigma}^2 \right) = \frac{p_j \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left( -\frac{1}{2\sigma_j^2} (x_i - \mu_j \right)^2}{\sum_{j'=1}^{k} p_j \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left( -\frac{1}{2\sigma_j^2} (x_i - \mu_j \right)^2}. \tag{76}$$

Fixing this updated distribution, we can now consider the expected log likelihood, using this distribution for the latent variables:

$$\begin{aligned} \mathbb{E}\left[ \ln \text{lik}(\underline{p}, \underline{\mu}, \underline{\sigma}^2 | \underline{x}, \underline{Z}) \right] &= \sum_{i=1}^{N} \sum_{j=1}^{k} \mathbb{P}\left( Z_i = j | X_i = x_i, \underline{p}, \underline{\mu}, \underline{\sigma}^2 \right) \left[ \ln\left( p_j \frac{1}{\sigma_j \sqrt{2\pi}} \right) - \frac{1}{2\sigma_j^2} (x_i - \mu_j)^2 \right] \\ &= \sum_{i=1}^{N} \sum_{j=1}^{k} q_{i,j} \left[ \ln\left( p_j \frac{1}{\sigma_j \sqrt{2\pi}} \right) - \frac{1}{2\sigma_j^2} (x_i - \mu_j)^2 \right] \end{aligned} \tag{77}$$

Attempting to find a new model, $(\underline{p}', \underline{\mu}', \underline{\sigma'}^2)$ to maximize this, we want:

$$\begin{aligned} \frac{\partial}{\partial p_j'} \mathbb{E}\left[ \ln \text{lik}(\underline{p}', \underline{\mu}', \underline{\sigma'}^2 | \underline{x}, \underline{Z}) \right] &= \frac{1}{p_j'} \sum_{i=1}^{N} q_{i,j} \\ \frac{\partial}{\partial \mu_j'} \mathbb{E}\left[ \ln \text{lik}(\underline{p}', \underline{\mu}', \underline{\sigma'}^2 | \underline{x}, \underline{Z}) \right] &= \frac{1}{\sigma'^2_j} \sum_{i=1}^{N} q_{i,j} (x_i - \mu_j') \\ \frac{\partial}{\partial \sigma_j'} \mathbb{E}\left[ \ln \text{lik}(\underline{p}', \underline{\mu}', \underline{\sigma'}^2 | \underline{x}, \underline{Z}) \right] &= \sum_{i=1}^{N} q_{i,j} \left[ \frac{1}{\sigma'^3_j} (x_i - \mu_j')^2 - \frac{1}{\sigma_j'} \right] \end{aligned} \tag{78}$$

The derivatives for $p_j'$, combined with the fact that $\sum_j p_j' = 1$, solves for a maximizer of

$$(p_j')^* = \frac{1}{N} \sum_{i=1}^{N} q_{i,j}. \tag{79}$$

The other parameters can be solved for in an unconstrained way, giving

$$0 = \frac{1}{\sigma'^2_j} \sum_{i=1}^{N} q_{i,j} (x_i - \mu_j') \mapsto (\mu_j')^* = \frac{\sum_{i=1}^{N} q_{i,j} x_i}{\sum_{i=1}^{N} q_{i,j}} \tag{80}$$

and

$$\begin{aligned}
0 &= \sum_{i=1}^{N} q_{i,j} \left[ \frac{1}{\sigma'^3_j}(x_i - \mu'_j)^2 - \frac{1}{\sigma'_j} \right] \\
&= \sum_{i=1}^{N} q_{i,j} \left[ \frac{1}{\sigma'^2_j}(x_i - \mu'_j)^2 - 1 \right] \\
&= \frac{1}{\sigma'^2_j} \sum_{i=1}^{N} q_{i,j}(x_i - \mu'_j)^2 - \sum_{i=1}^{N} q_{i,j} \mapsto (\sigma'^2_j)^* = \frac{\sum_{i=1}^{N} q_{i,j}(x_i - \mu'_j)^2}{\sum_{i=1}^{N} q_{i,j}} \mapsto (\sigma'^2_j)^* = \frac{\sum_{i=1}^{N} q_{i,j}(x_i - (\mu'_j)^*)^2}{\sum_{i=1}^{N} q_{i,j}}.
\end{aligned} \tag{81}$$

Hence we have the following update formulas: at any time, given a model $(\underline{p}, \underline{\mu}, \underline{\sigma}^2)$, we compute the $\{q_{i,j}\}$ based on this model, and update it an exchange it for a model given by

$$\begin{aligned}
(p'_j)^* &= \frac{1}{N} \sum_{i=1}^{N} q_{i,j} \\
(\mu'_j)^* &= \frac{\sum_{i=1}^{N} q_{i,j} x_i}{\sum_{i=1}^{N} q_{i,j}} \\
(\sigma'^2_j)^* &= \frac{\sum_{i=1}^{N} q_{i,j}(x_i - (\mu'_j)^*)^2}{\sum_{i=1}^{N} q_{i,j}},
\end{aligned} \tag{82}$$

effectively interpretable as calculating the posterior frequency of being in class $j$, the weighted sample mean of class $j$ (weighted by the likelihood of being in class $j$), and the weighted variance of class $j$ (again, waited by the likelihood of being in class $j$).

### 6.1.4 $k$-Means Clustering as the EM Algorithm

It is interesting to observe that the classical $k$-Means clustering algorithm can be viewed as an extremely specialized case of the EM algorithm for Gaussian mixture models. In particular, suppose that (for the moment) the variances for each class $j$ were known and constant $\sigma^2$. In that case, we have

$$\text{lik}(\underline{p}, \underline{\mu} | \underline{x}, \underline{z}) = \prod_{i=1}^{N} \prod_{j=1}^{k} \left[ p_j \frac{1}{\sigma\sqrt{2\pi}} \exp\left( -\frac{1}{2\sigma^2}(x_i - \mu_j)^2 \right) \right]^{\mathbf{1}\{z_i = j\}}, \tag{83}$$

or

$$\ln \text{lik}(\underline{p}, \underline{\mu} | \underline{x}, \underline{z}) = \sum_{i=1}^{N} \sum_{j=1}^{k} \mathbf{1}\{z_i = j\} \left[ \ln\left( p_j \frac{1}{\sigma\sqrt{2\pi}} \right) - \frac{1}{2\sigma^2}(x_i - \mu_j)^2 \right], \tag{84}$$

and the posterior latent distributions are given by:

$$q_{i,j} = \mathbb{P}\left( Z_i = j | X_i = x_i, \underline{p}, \underline{\mu} \right) = \frac{p_j \exp\left( -\frac{1}{2\sigma^2}(x_i - \mu_j \right)^2}{\sum_{j'=1}^{k} p_j \exp\left( -\frac{1}{2\sigma^2}(x_i - \mu_j \right)^2}. \tag{85}$$

Following the same process gets you the following update equations:

$$\begin{aligned}
(p'_j)^* &= \frac{1}{N} \sum_{i=1}^{N} q_{i,j} \\
(\mu'_j)^* &= \frac{\sum_{i=1}^{N} q_{i,j} x_i}{\sum_{i=1}^{N} q_{i,j}}
\end{aligned} \tag{86}$$

But notice what happens in the limit as $\sigma \to 0$, essentially increasingly penalizing distance from a given class's mean. In this case, $q_{i,j}$ in the limit becomes

$$q_{i,j} = \begin{cases} 1 & \text{if } j = \text{argmin}_{j'}(x_i - \mu_{j'})^2 \\ 0 & \text{else ,} \end{cases} \tag{87}$$

i.e., $q_{i,j}$ becomes an indicator variable indicating which class $j$ the data point $i$ is closest to. In this case, the estimate $(p'_j)^*$ now represents the fraction of the population closest to current class mean $\mu_j$, and $(\mu'_j)^*$ represents the re-centered mean or average of everything currently in class $j$ (by proximity to the old mean).

Iterating this process in the spirit of the EM algorithm accomplishes the following:

- Start with some random collection of means $\underline{\mu}$.
- Partition the data points $x_i$ based on which mean $\mu_j$ they are closest to.
- Compute new means for each class $j$, based on averaging the $x_i$ currently in that class.
- Repartition, recompute, repeat.

This is the classical solution to $k$-Means clustering known as 'Lloyd's Algorithm'.