CSE6243: Advanced Machine Learning **Fall 2024**

Lecture 14: Autoregressive Model

Lecturer: Bo Dai *Scribes: Ruchi Patel, Ayushi Rajpoot*

Note: *LaTeX template courtesy of UC Berkeley EECS Department.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

14.1 Recap

In a previous lecture, we talked about energy-based models. We can denote the probability density function as

$$p(x) = \frac{\exp(f(x))}{Z(f)}$$

where $f(x)$ is the energy function and $Z(f)$ is the partition function that ensures $p(x)$ sums to 1 over all possible values of x:

$$z(f) = \int \exp(f(x))dx$$

We can build connections from EBMs to diffusion models. This connection is particularly relevant when considering continuous parameterization. There's an important relationship between sampling/generation processes and diffusion models, which involves the gradient of the energy function with respect to x: $\nabla_x f(x)$.

For discrete cases, Gibbs sampling becomes a relevant technique, which we'll explore in more detail in this lecture.

14.2 New Content

14.2.1 Gibbs Sampling

Gibbs sampling is extremely helpful when it comes to understanding the joint distribution across several random variables by iterative sampling conditional distributions of each variable given all other variable conditions.

Algorithm 1 Gibbs Sampling algorithm

```
$x_0 \sim p_0(x)$
for t = 1 ... do
   for i = 1 ... L do
      $x_i \sim p(x_i|x_{-i})$
   end for
end for
```

Let us denote $x = \{x_i\}_{i=1}^{L}$ where $x_i \in \mathbb{R}^{1\times k}$ is a one-hot vector, filled with 0s and 1s. We use the standard notation $x_{-i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_k)$, where x_{-i} is every value in the vector other than x_i.

From this algorithm, we can say:

$$\begin{aligned} p(x_i|x_{-i}) &\propto \exp(f(x_i, x_{-i})) \\ &= \frac{\exp(f(x_i, x_{-i}))}{\sum\limits_{y\in X} \exp(f(y, x_{-i}))} \end{aligned}$$

In this context, f represents the energy function of our model. It's an arbitrary function that parameterizes the probability distribution we're working with. This tells us how likely it is to sample x_i given the other fixed variables conditions.

And we see that the probability of sampling x_i ends up being proportional to our energy function. We divide by $\sum\limits_{y\in X} \exp(f(y, x_{-i}))$ in order to normalize our energy function values.

Note that this algorithm is equivalent to softmax$(f(x_i, x_{-i}))$. Thus, the sampling algorithm is setting each x_i to softmax$(f(x_i, x_{-i}))$.

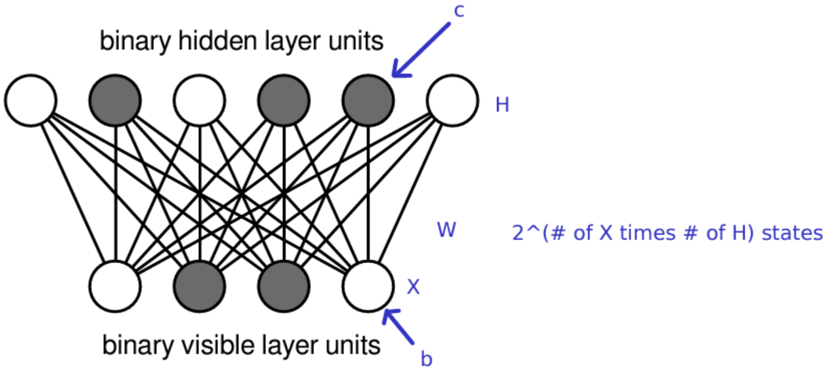
Gibbs sampling can be computationally intensive. An approximation that is easier to calculate is:

$$p(x_i|x_{-i}) \approx p(x_i|x_{<i})$$

Here, we only take variables we have already generated.

14.2.2 Restricted Boltzmann Machine

Restricted Boltzmann machines are probability distributions for binary data with many tunable parameters. Boltzmann machines are in the same family as entropy models. They consist of a visible layer which consists of the input data and a hidden layer which captures patterns from the input data. The connection between these two layers have weights that depict how the hidden and visible layer units interact and connect with each other. And so, the Restricted Boltzman machine is helpful when it comes to understanding underlying patterns within data.


binary hidden layer units
c
H
W
2^(# of X times # of H) states
X
binary visible layer units
b

Here, $h \in \{0,1\}^p$. This graph is restricted because we only have quadratic terms between h and x. There is no dependency among x's or h's. We can denote the potential function as:

$$p(x,h) \propto \exp(h^T W x)$$

where $h \in \mathbb{R}^{1\times p}, W \in \mathbb{R}^{p\times(L\times k)}$ and $x \in \mathbb{R}^{(L\times k)\times 1}$

The probability of a certain x and h configuration is proportional to the exponential of $h^T W x$ (the interaction between the two in relation to W).

Now, let us talk about some properties of the model.

$$p(h_1|x) = \frac{\exp(h^T W x)}{\sum_h \exp(h^T W x)}$$

We divide by $\sum_h \exp(h^T W x)$ here to normalize the values for the probability to ensure it lies between 0 and 1.

Let us take a look at the term $h^T W x$. We can denote this as:

$$h^T W x = h^T y$$

where $y = Wx \in \mathbb{R}^{p\times 1}$. We can rewrite the term:

$$h^T W x = h^T y = \sum_{i=1}^{p} h_i y_i$$

Substituting this term into our equation for p, we get:

$$\begin{aligned} p(h_i = 1|x) &\propto \exp(h_i y_i) \\ &= \frac{\exp(h_i y_i)}{1 + \exp(y_i)} \end{aligned}$$

Note that since $h \in \{0,1\}^p$, the term $\exp(h_i y_i)$ basically gives us a binary distribution used to calculate how likely that the hidden unit h_i will be on or off. Additionally, this equation is a sigmoid function.

To further understand the RBM, we can consider the relationship between the visible units x and the hidden units h. By substituting $\mu = h^T W$ into $h^T W$, we get $\mathrm{tr}(\mu x) = \mu^T x_i$. This leads to an important result $\mu = h^T W$ into $h^T W x$ so $tr(\mu x) = \mu^T x_i$ and then we get

$$p(x_i|h) = \mathrm{softmax}(\mu_i x_i)$$

This relationship shows how the hidden units influence the probability of the visible units, which is crucial for understanding the generative process in RBMs.

14.2.3 Block-wise Gibbs Sampling

The Block-wise Gibbs sampling algorithm is a variation of Gibbs sampling explained above that instead variables are updated in block segments instead of being considered one at a time. This approach is generally more efficient as a result.

Algorithm 2 Block-wise Gibbs Sampling algorithm

```
  $x_0 \sim p_0(x)$
  for t = 1 ... T do
      $\{h_i\}_{i=1}^{p} \sim \prod_{i=1}^{p} p(h_i = 1|x)$
      $\{x_i\}_{i=1}^{L} \sim \prod_{i=1}^{L} p(x_i|h)$
  end for
```

As shown above, in Block-wise Gibbs Sampling, you first calculate the probability of each hidden unit being on based on current visible layer units values and then update all hidden units from these calculation as a block at once. And then do the same for the visible units but based on the newly calculated hidden values.

How can we change this produced to an auto-regressive model?

In this algorithm, ζ denotes a sigmoid function that outputs a vector with dimension $\mathbb{R}^{p \times 1}$.

Algorithm 3 Auto-regressive model construction

```
  $\tilde{h} \leftarrow \zeta(W^T x)$
  $x = \text{softmax}(h^T W x)$
  for t = 1 ... L do
      $\{h_i\}_{i=1}^{p} \sim \prod_{i=1}^{p} p(h_i = 1|x)$
      $\{x_i\}_{i=1}^{L} \sim \prod_{i=1}^{L} p(x_i|h)$
  end for
```

This modified algorithm for auto-regressive model construction will be based on $p(x_i, x_{<i}) = \text{softmax}(\sigma(W^T x_{<i})^T W x_i))$ where $h = \sigma(W^T x_{<i})$.

As for, a key difference between the two is that while Gibbs sampling doesn't care about order in which variables are sampled, the equation for $p(x_{T(i)}, x_{\pi(<i)})$ does. Here, π denotes the permutation that determins the order of the variables. This ordering is crucial for the autoregressive nature of the model, as each variable depends on all previous variables in the sequence. This distinction between Gibbs sampling and the autoregressive approach is important for understanding the model's behavior.