Homework 2
Graphical Models[1]

CMU 10-707: Topics in Deep Learning (Spring 2022)

https://deeplearning-cmu-10707-2022spring.github.io/

OUT: Thursday, Feb 17th, 2022

DUE: Thursday, March 3rd, 2022, 11:59pm

TAs: Minji Yoon, Zhili Feng, Kin G. Olivares

START HERE: Instructions

Homework 2 covers topics on graphical models. The homework includes short answer questions, derivation questions, and a coding task.

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: https://deeplearning-cmu-10707-2022spring.github.io/index.html#policies
- **Late Submission Policy:** See the late submission policy here: https://deeplearning-cmu-10707-2022spring.github.io/index.html#policies
- **Submitting your work:**
  - **Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (https://gradescope.com/). Please write your solution in the LaTeX files provided in the assignment and submit in a PDF form. Put your answers in the question boxes (between `\begin{soln}` and `\end{soln}`) below each problem. Please make sure you complete your answers within the given size of the question boxes. **Handwritten solutions are not accepted and will receive zero credit.** Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. For more information about how to submit your assignment, see the following tutorial (note that even though the assignment in the tutorial is handwritten, submissions must be

[1] Compiled on Thursday 3rd March, 2022 at 01:51

typed): https://www.youtube.com/watch?v=KMPoby5g_nE&feature=youtu.be

- **Code submission:** All code must be submitted to a Gradescope autograder named as "Assignment 2: Programming". **If you do not submit your code here, you will not receive any credit for your assignment.** Gradescope grader will be used to check for plagiarism. Please make sure you familiarize yourself with the academic integrity information for this course.

Important Notes on the Written Problems:

- Please make sure that the solution boxes do not move from the original places in your writeup pdf. And also please check whether your solution boxes fit into the solution areas when you submit your pdf to Gradescope.
- If you have more text than the solution box, you can resize the box using height option in the latex command $\backslash begin\{soln\}\{height = 10cm\}$.
- You can also put $\backslash pagebreak$ before the solution environment (before $\backslash begin\{soln\}$) to make your final pdf look better-organized.

Important Notes on the Programming Problems:

- Do not use any toolboxes except those already imported in the code template.
- Read the doc-strings/comments in the template very carefully before you start.
- Reach out for help on Piazza or during office hours when you struggle.
- Do not change any function signatures because your code will be auto-graded.
- Try to vectorize the computation as much as possible (e.g. compute in the form of matrix multiplication, utilize numpy functions instead of loops, etc.)
- Use Python 3.6 or above, and the latest version of numpy.

Problem 1 (5 pts)

By marginalizing out the variables in order, show that the representation

$$p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|pa_k)$$

for the joint distribution of a directed graph is correctly normalized, provided each of the conditional distribution is normalized. Note that pa_k denote parent nodes of node x_k in the directed graph.

Solution

From the expression $p(\mathbf{x}) = \prod_{k=1}^{K} p(x_k|pa_k)$, we can get the following normalized expression:
$\int_{\mathbf{x}} p(\mathbf{x})d\mathbf{x}$
$= \int_{\mathbf{x}} \prod_{k=1}^{K} p(x_k|pa_k)d\mathbf{x}$
$= \int_{x,x_2,\ldots,x_{k-1}} \int_{x_K} p(x_K|pa_K)d_{x_K} \prod_{k=1}^{K-1} p(x_k|pa_k)d_{x_{1\ldots K-1}}$
$= \int_{x,x_2,\ldots,x_{k-2}} \int_{x_K} p(x_K|pa_K)d_{x_K} \int_{x_{K-1}} p(x_{K-1}|pa_{K-1})d_{x_{K-1}} \prod_{k=1}^{K-2} p(x_k|pa_k)d_{x_{1\ldots K-2}}$
Keep doing this, we have:
$= \int_{x_K} p(x_K|pa_K)d_{x_K} \int_{x_{K-1}} p(x_{K-1}|pa_{K-1})d_{x_{K-1}} \cdots \int_{x_1} p(x_1|pa_1)d_{x_1}$ (1)
Since, we assume that each conditional distribution is correctly normalized, therefore,
$\int_{x_k} p(x_k|pa_k)d_{x_k} = 1 for k = 1, 2, \ldots K$ Therefore,
(1)=$1^K = 1$
Thus we prove the normalized expression $\int_{\mathbf{x}} p(\mathbf{x})d\mathbf{x} = 1$, it is correctly normalized (QED).

Problem 2 (10 pts)

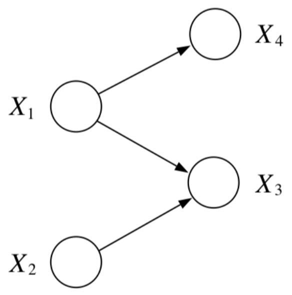


Figure 1: Directed Graphical Model

1. Without making any conditional independence assumption, given that each random variable can take r values, how many parameters would be required to enumerate the joint distribution of the model in Figure 1? You can start with writing down the joint distribution of the model as the product of the local conditional distributions shown in Figure 1.

2. State True or False for the following with required proof or reasons.

   (a)
   $$X_3 \perp\!\!\!\perp X_4 \mid X_1$$

   (b)
   $$X_1 \perp\!\!\!\perp X_2 \mid X_3$$

3. Repeat part 2 assuming that the edges in the graph are undirected.

Solution

1. **answer:**
$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2)p(x_3|x_2, x_1)p(x_4|x_1)$
The parameters needed for $p(x_1), p(x_2), p(x_3|x_2, x_1), p(x_4|x_1)$ are $r-1, r-1, r(r-1), r^2(r-1), r(r-1)$
Therefore, there are total $r-1+r-1+r^3-r^2+r^2-r = 2r-2+r^3-r = r^3+r-2$

2(a) **answer(since it is required not use d-separation, I will also provide formal method):**
to prove the independence, we need to show $P(X_3X_4|X_1) = P(X_3|X_1)P(X_4|X_1)$
d-separation approach:
From the graph, we can see that $X4$ and $X3$ has a common ancestor $X1$, this is a common cause situation. By using d-separation, following the procedure of d-separation: Draw ancestral graph - Moralize - Disorient - Delete givens, we can see that $X1$ is not a collider and the only element in the only path between $X3$ and $X4$.
Thus $X3$ and $X4$ are d-connected but d-separated giving $X1$, thus $X4$ and $X3$ are independent giving $X1$.
formal proof:
This is a common cause scenario, x_1 is the only cause to x_3 and x_4, therefore:
$P(x_3, x_4, x_1) = P(x_1)P(x_3|x_1)P(x_4|x_1)$
thus: $P(x_3, x_4|x_1) = \frac{P(x_3,x_4,x_1)}{P(x_1)} = \frac{P(x_1)P(x_3|x_1)P(x_4|x_1)}{P(x+1)} = P(x_3|x_1)P(x_4|x_1)$ (QED).
Thus, it is True.

2(b) **answer:**
to prove the independence, we need to show $P(X_1X_2|X_3) = P(X_1|X_3)P(X_2|X_3)$
From the graph, we can see that $X1$ and $X2$ has a common child $X3$, this is a common effect situation. By using d-separation, following the procedure of d-separation, we can see X_1 is a collider and the only element in the only path between $X1$ and $X2$.
Therefore $X1$ and $X2$ are d-separated but d-connected given $X3$, there is connection between $X1$ and $X2$, thus $X1$ and $X2$ are not independent giving $X3$.
formal proof:
This is the common effect scenario, both X_1 and X_2 affect X_3 and they are the only two causes to X_3.
Therefore, I can provide example and counter example, suppose being rich needs you to either earn a lot or inherit a huge amount of money from family. These two factor (earn a lot and inherit wealth) are independent as either of it can lead you to be rich. However, they are not conditionally independent provided you are rich. Suppose we know that you are rich and we know you do not earn a lot, then we know that you must have a very rich parents. Therefore, we can see that $P(wealthyparent|youarerich)$, $P(earnalot|youarerich)$ are not independent.
Thus, it is False.

3. **answer:**
if the graph is undirected, then we could not know whether the x_1 and x_3 are collider or not

collider, we could only confirm that x_4 and x_3 are separated by x_1, x_1 and x_2 are separated by x_3, thus, this generalized the Markov property for chains like this. Actually, in the undirected graph, the conditional distribution of each node only dependent on the nodes in the Markov blanket. In this context, the pair x_4 and x_3 as well as the pair x_1 and x_2 are not in the Markov blanket of each other, thus they should be independent. We can see that both (a) and (b) statements become true that x_4 and x_3 are independent giving x_1, x_1 and x_2 are independent giving x_3.

Problem 3 (10 pts)

Consider the use of iterated conditional modes (ICM) to minimize the energy function (that we considered in class for image denoising example) given by:

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{i,j} x_i x_j - \gamma \sum_i x_i y_i$$

where $x_i \in \{-1, 1\}$ is a binary variable denoting the state of pixel i in the unknown noise-free image, i and j are indices of neighboring pixels, and $y_i \in \{-1, 1\}$ denotes the corresponding value of pixel i in the observed noisy image. The joint distribution is defined as:

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} exp(-E(\mathbf{x}, \mathbf{y}))$$

1. (5 pts) Write down an expression for the difference in the values of the energy associated with the two states of a particular variable x_j, with all other variables held fixed, and show that it depends only on quantities that are local x_j in the graph.
2. (5 pts) Consider a particular case of the energy function above in which the coefficients $\beta = h = 0$. Show that the most probable configuration of the latent variables is given by $x_i = y_i$ for all i.

Solution

1. **answer:**
The difference between two states of x_j is 1 as $x_j \in \{-1, 1\}$.
Therefore, suppose we have a particular x_j with value -1 and 1, we denote this specific variable as x'_j
Then, we know that:
$E(\mathbf{x}, \mathbf{y})$
$= h\sum_i x_i - \beta\sum_{i,j} x_i x_j - \gamma\sum_i x_i y_i$
$= h\sum_{i\neq j'} x_i - \beta\sum_{i\in\mathbf{ne}(j), j\neq j'} x_i x_j - \gamma\sum_{i\neq j'} x_i y_i + hx_{j'} - \beta\sum_{i'\in\mathbf{ne}(j')} x_{i'} x_{j'} - \gamma x_{j'} y_{j'}$
Therefore, we can compute the difference:
$E(\mathbf{x}, \mathbf{y})_{x_{j'}=1} - E(\mathbf{x}, \mathbf{y})_{x_{j'}=-1}$
$= h\sum_{i\neq j'} x_i - \beta\sum_{i\in\mathbf{ne}(j), j\neq j'} x_i x_j - \gamma\sum_{i\neq j'} x_i y_i + h * 1 - \beta\sum_{i'\in\mathbf{ne}(j')} x_{i'} * 1 - \gamma 1 * y_{j'}$
$- h\sum_{i\neq j'} x_i + \beta\sum_{i,j\neq j'} x_i x_j + \gamma\sum_{i\neq j'} x_i y_i - h * (-1) + \beta\sum_{i\in\mathbf{ne}(j')} x_{i'} * (-1) + \gamma(-1) * y_{j'}$
$= 2h - 2\beta\sum_{i\in\mathbf{ne}(j')} x_{i'} - 2\gamma y_{j'}$
Therefore, we can see that the difference only depends on $x_{i'}$ which are neighbors of this particular $x_{j'}$, and $y_{j'}$ which is the observed value for pixel j' (QED).

2. **answer:**
For $\beta = h = 0$, we know that $E(\mathbf{x}, \mathbf{y}) = -\gamma\sum_i x_i y_i$
Therefore, $p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} exp(\gamma\sum_i x_i y_i)$
As $x_i, x_j \in \{-1, 1\}$ can only take value between -1 and 1, we can see that: if $x_i \neq x_j$ then $x_i, x_j = -1$
if $x_i = x_j$ then $x_i, x_j = 1$
Thus, when $x_i = x_j$, $p(\mathbf{x}, \mathbf{y})$ is higher. Therefore, the most probable configuration is to keep the energy function lowest which is -1, therefore, we need $x_i = y_i$ for all i (QED).

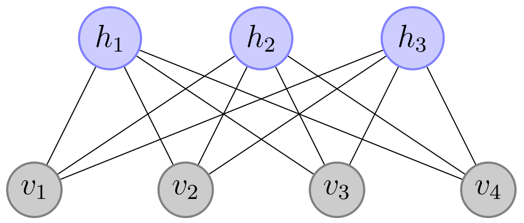
Problem 4 (15 pts)

We learn that a deep belief network (DBN) as a generative graphical model, consists of multiple layers of latent variables with connections between the layers. One DBN can learn to probabilistically reconstruct its inputs and extract a deep hierarchical representation of the training data. DBNs can be viewed as a composition of simple, unsupervised networks such as restricted Boltzmann machines (RBMs), where each sub-network's hidden layer serves as the visible layer for the next.

In this problem, we will learn how RBMs and DBNs are trained. Based on what you induce here, you will implement RBMs and DBNs in the following programming part.

Restricted Boltzmann Machine

In the RBM, visible units are conditionally independent on hidden units and vice versa.



For a given RBM, we relate the units with the energy function as follow:

$$Energy(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T\mathbf{h} - \mathbf{c}^T\mathbf{v} - \mathbf{h}^T\mathbf{W}\mathbf{v}$$

where $\mathbf{b} \in \mathbb{R}^H, \mathbf{c} \in \mathbb{R}^V$ are offset/bias vectors and $\mathbf{W} \in \mathbb{R}^{H\times V}$ comprises the weights connecting units.

The joint probability $P(\mathbf{v}, \mathbf{h})$ is presented as follows:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}e^{-Energy(\mathbf{v},\mathbf{h})}$$

where Z is the normalization term.

(a) Conditional probabilities $P(\mathbf{v}|\mathbf{h})$ and $P(\mathbf{h}|\mathbf{v})$ (2 pts)

Assuming $\mathbf{v}$ and $\mathbf{h}$ are binary units, show $P(v_i = 1|\mathbf{h})$ and $P(h_j = 1|\mathbf{v})$ are presented as follow:

$$P(v_i = 1|\mathbf{h}) = sigm(c_i + \mathbf{h}^T\mathbf{W}_{:,i})$$
$$P(h_j = 1|\mathbf{v}) = sigm(b_j + \mathbf{W}_{j,:}\mathbf{v})$$

where $sigm(x) = \frac{1}{1+e^{-x}}$.

Solution

We have condiitonal probability like this:

$P(\mathbf{v}|\mathbf{h}) = \frac{P(\mathbf{v},\mathbf{h})}{P(\mathbf{h})} = \frac{\frac{1}{Z}e^{-(-\mathbf{b}^T\mathbf{h}-\mathbf{c}^T\mathbf{v}-\mathbf{h}^T\mathbf{W}\mathbf{v})}}{P(\mathbf{h})} = \frac{\frac{1}{Z}e^{\mathbf{b}^T\mathbf{h}+\mathbf{c}^T\mathbf{v}+\mathbf{h}^T\mathbf{W}\mathbf{v}}}{P(\mathbf{h})}$

Since $P(\mathbf{h})$ is provided,therefore,$\frac{e^{\mathbf{b}^T\mathbf{h}}}{P(\mathbf{h})}$ should be a constant, thus we can denoted $\frac{\frac{1}{Z}e^{\mathbf{b}^T\mathbf{h}}}{P(\mathbf{h})}$ as Z_h

$P(\mathbf{v}|\mathbf{h}) = \frac{1}{Z_h}e^{\mathbf{c}^T\mathbf{v}+\mathbf{h}^T\mathbf{W}\mathbf{v}}$

Similarly, we can see that:

$P(\mathbf{h}|\mathbf{v}) = \frac{P(\mathbf{v},\mathbf{h})}{P(\mathbf{v})} = \frac{\frac{1}{Z}e^{\mathbf{b}^T\mathbf{h}+\mathbf{c}^T\mathbf{v}+\mathbf{h}^T\mathbf{W}\mathbf{v}}}{P(\mathbf{v})} = \frac{1}{Z_v}e^{\mathbf{b}^T\mathbf{h}+\mathbf{h}^T\mathbf{W}\mathbf{v}}$, we denote $\frac{\frac{1}{Z}e^{\mathbf{c}^T\mathbf{v}}}{P(\mathbf{v})}$ as Z_v

Therefore we cam compute:

$P(v_i = 1|\mathbf{h}) = \frac{P(v_i=1|\mathbf{h})}{P(v_i=0|\mathbf{h})+P(v_i=1|\mathbf{h})} = \frac{\frac{1}{Z_h}e^{c_i+\mathbf{h}^T\mathbf{W}_{:,i}}}{\frac{1}{Z_h}(e^0+e^{c_i++\mathbf{h}^T\mathbf{W}_{:,i}})} = \frac{e^{c_i+\mathbf{h}^T\mathbf{W}_{:,i}}}{1+e^{c_i++\mathbf{h}^T\mathbf{W}_{:,i}}} = \frac{1}{1+e^{-(c_i++\mathbf{h}^T\mathbf{W}_{:,i})}}$

$= sigm(c_i + \mathbf{h}^T\mathbf{W}_{:,i})$

$P(h_j = 1|\mathbf{v}) = \frac{P(h_j=1|\mathbf{v})}{P(h_j=0|\mathbf{v})+P(h_j=1|\mathbf{v})} = \frac{\frac{1}{Z_v}e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}{\frac{1}{Z_v}(e^0+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}})} = \frac{e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}{1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}} = \frac{1}{1+e^{-(b_j+\mathbf{W}_{j,:}\mathbf{v})}}$

$= sigm(b_j + \mathbf{W}_{j,:}\mathbf{v})$ (QED).

(b) Free energy (3 pts)

We obtain $P(\mathbf{v})$ by marginalizing $\mathbf{h}$ as follow:

$$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}\in\{0,1\}^H} e^{-Energy(\mathbf{v},\mathbf{h})}}{Z} = \frac{e^{-FreeEnergy(\mathbf{v})}}{Z}$$

where $FreeEnergy(\mathbf{v})$ form makes it easier to compute gradients with visible units only.

When we rewrite the energy function as follow:

$$Energy(\mathbf{v},\mathbf{h}) = -\beta(\mathbf{v}) - \sum_{j=1}^{H} \gamma_j(\mathbf{v}, h_j)$$

Show $P(\mathbf{v})$ and $FreeEnergy(\mathbf{v})$ could be presented as follow:

$$P(\mathbf{v}) = \frac{e^{\beta(\mathbf{v})}}{Z} \prod_{j=1}^{H} \sum_{h_j\in\{0,1\}} e^{\gamma_j(\mathbf{v},h_j)}$$

$$\begin{aligned} FreeEnergy(\mathbf{v}) &= -\beta(\mathbf{v}) - \sum_{j=1}^{H} \log \sum_{h_j\in\{0,1\}} e^{\gamma_j(\mathbf{v},h_j)} \\ &= -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H} \log\left(1 + e^{b_j + \mathbf{W}_{j,:}\mathbf{v}}\right) \end{aligned}$$

Solution

From the context we can see that:

$P(\mathbf{v}) = \frac{\sum_{\mathbf{h}\in\{0,1\}^H} e^{\beta(\mathbf{v})+\sum_{j=1}^{H}\gamma_j(\mathbf{v},h_j)}}{Z} = \frac{e^{\beta(\mathbf{v})}}{Z}\sum_{\mathbf{h}\in\{0,1\}^H} e^{\sum_{j=1}^{H}\gamma_j(\mathbf{v},h_j)}$

$= \frac{e^{\beta(\mathbf{v})}}{Z}\sum_{\mathbf{h}\in\{0,1\}^H} e^{\gamma_1(\mathbf{v},h_1)\gamma_2(\mathbf{v},h_2)\ldots\gamma_H(\mathbf{v},h_H)} = \frac{e^{\beta(\mathbf{v})}}{Z}\sum_{\mathbf{h}\in\{0,1\}^H} e^{\gamma_1(\mathbf{v},h_1)}e^{\gamma_2(\mathbf{v},h_2)}\ldots e^{\gamma_H(\mathbf{v},h_H)}$

$= \frac{e^{\beta(\mathbf{v})}}{Z}\sum_{\mathbf{h}_1\in\{0,1\}^H} e^{\gamma_1(\mathbf{v},h_1)}\sum_{\mathbf{h}_2\in\{0,1\}^H} e^{\gamma_2(\mathbf{v},h_2)}\ldots\sum_{\mathbf{h}_H\in\{0,1\}^H} e^{\gamma_H(\mathbf{v},h_H)}$

$= \frac{e^{\beta(\mathbf{v})}}{Z}\prod_{j=1}^{H}\sum_{\mathbf{h}_j\in\{0,1\}^H} e^{\gamma_j(\mathbf{v},h_j)}$

From $P(\mathbf{v})$, We can know that:

$e^{\beta(\mathbf{v})}\prod_{j=1}^{H}\sum_{\mathbf{h}_j\in\{0,1\}^H} e^{\gamma_j(\mathbf{v},h_j)}$

$= \frac{e^{\beta(\mathbf{v})}}{Z}\sum_{\mathbf{h}_1\in\{0,1\}^H} e^{\gamma_1(\mathbf{v},h_1)}\sum_{\mathbf{h}_2\in\{0,1\}^H} e^{\gamma_2(\mathbf{v},h_2)}\ldots\sum_{\mathbf{h}_H\in\{0,1\}^H} e^{\gamma_H(\mathbf{v},h_H)} = e^{-FreeEnergy(\mathbf{v})}$

take log on both sides:

$\beta(\mathbf{v}) + \log\sum_{\mathbf{h}_1\in\{0,1\}^H} e^{\gamma_1(\mathbf{v},h_1)} + \log\sum_{\mathbf{h}_2\in\{0,1\}^H} e^{\gamma_2(\mathbf{v},h_2)} + \cdots + \log\sum_{\mathbf{h}_H\in\{0,1\}^H} e^{\gamma_H(\mathbf{v},h_H)}$

$= -FreeEnergy(\mathbf{v})$

Therefore: $FreeEnergy(\mathbf{v}) = -\beta(\mathbf{v}) - \sum_{j=1}^{H}\log\sum_{h_j\in\{0,1\}} e^{\gamma_j(\mathbf{v},h_j)}$

From the rewrite: $-\beta(\mathbf{v}) - \sum_{j=1}^{H}\gamma_j(\mathbf{v},h_j) = -\mathbf{b}^T\mathbf{h} - \mathbf{c}^T\mathbf{v} - \mathbf{h}^T\mathbf{W}\mathbf{v}$, we know that:

$\beta(\mathbf{v}) = \mathbf{c}^T\mathbf{v}$ and $\gamma_j(\mathbf{v},h_j) = b_jh_j + h_j\mathbf{W}_{j,:}\mathbf{v}$

thus we have: $FreeEnergy(\mathbf{v}) = -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log(e^{\gamma_j(\mathbf{v},h_j=0)} + e^{\gamma_j(\mathbf{v},h_j=1)})$

$= -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log(e^{(b_j+\mathbf{W}_{j,:}\mathbf{v})*0} + e^{(b_j+\mathbf{W}_{j,:}\mathbf{v})*1}) = -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log\left(1 + e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}\right)$ (QED).

(c) Log-likelihood gradient of $P(v)$ (5 pts)

Show log-likelihood gradient of $P(v)$ could be presented as follows:

"Positive Phase" "Negative Phase"

$$\frac{\partial \log P(\mathbf{v})}{\partial \theta} = -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} + \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} P(\tilde{\mathbf{v}}) \frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial \theta}$$

Hint: present $\frac{\partial \log Z}{\partial \theta}$ in terms of $FreeEnergy(\mathbf{v})$ and $P(\mathbf{v})$.

Solution

Because $Z = \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}$, we have:

$\log P(\mathbf{v}) = \log \left(\frac{e^{-FreeEnergy(\mathbf{v})}}{Z} \right) = \log(e^{-FreeEnergy(\mathbf{v})}) - \log(\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})})$

Thus, we can compute the gradient:

$\frac{\partial \log P(\mathbf{v})}{\partial \theta} = \frac{\partial}{\partial \theta} \log e^{-FreeEnergy(\mathbf{v})} - \frac{\partial}{\partial \theta} \log(\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})})$

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} - \frac{1}{\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})} \ln(e)} \frac{\partial}{\partial \theta} \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}$

(from (b), we know $\sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})} = e^{-FreeEnergy(\tilde{\mathbf{v}})}$, therefore:)

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} - \frac{1}{\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}} \frac{\partial}{\partial \theta} \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} e^{-FreeEnergy(\tilde{\mathbf{v}})}$

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} + \frac{1}{\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}} \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} e^{-FreeEnergy(\tilde{\mathbf{v}})} \frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial \theta}$

(from (b) we also know: $P(\mathbf{v}) = \frac{e^{-FreeEnergy(\mathbf{v})}}{Z}$ so that $e^{-FreeFreEnergy(\tilde{\mathbf{v}})} = ZP(\tilde{\mathbf{v}})$, therefore:)

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} + \frac{1}{\sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}} * \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} ZP(\tilde{\mathbf{v}}) \frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial \theta}$

(Since $Z = \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} \sum_{\tilde{\mathbf{h}} \in \{0,1\}^H} e^{-Energy(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})}$, therefore:)

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} + \frac{1}{Z} ZP(\tilde{\mathbf{v}}) \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} (\tilde{\mathbf{v}}) \frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial \theta}$

$= -\frac{\partial FreeEnergy(\mathbf{v})}{\partial \theta} + \sum_{\tilde{\mathbf{v}} \in \{0,1\}^V} P(\tilde{\mathbf{v}}) \frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial \theta}$ (QED).

(d) Gradients with regard to each variables (5 pts)

Show the gradients w.r.t. each variable $\mathbf{W}, \mathbf{b}, \mathbf{c}$ could be presented as follows:

$$-\frac{\partial \log P(\mathbf{v})}{\partial W_{ji}} = -P(h_j = 1|\mathbf{v}) \cdot v_i + E_{\tilde{\mathbf{v}}}[P(\tilde{h}_j = 1|\tilde{\mathbf{v}}) \cdot \tilde{v}_i]$$
$$-\frac{\partial \log P(\mathbf{v})}{\partial b_j} = -P(h_j = 1|\mathbf{v}) + E_{\tilde{\mathbf{v}}}[P(\tilde{h}_j = 1|\tilde{\mathbf{v}})]$$
$$-\frac{\partial \log P(\mathbf{v})}{\partial c_i} = -v_i + E_{\tilde{\mathbf{v}}}[\tilde{v}_i]$$

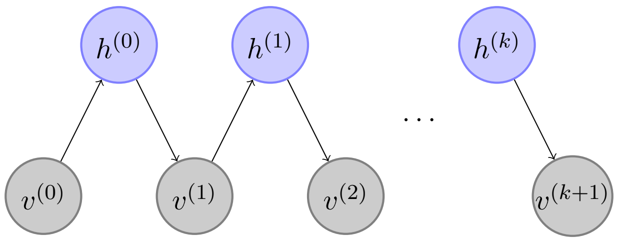
Hint: use the results from (a).

Solution

$P(h_j = 1|\mathbf{v}) = sigm(b_j + \mathbf{W}_{j,:}\mathbf{v}) = \frac{e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}{1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}$
From (b) and (c) and above, replace θ with derivative target parameter (W_{ji}, b_j, c_i), we know:
$-\frac{\partial \log P(\mathbf{v})}{\partial W_{ji}} = \frac{\partial FreeEnergy(\mathbf{v})}{\partial W_{ji}} - \sum_{\mathbf{v}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{\partial FreeEnergy(\tilde{\mathbf{v}})}{\partial W_{ji}}$
$= \frac{\partial -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}\right)}{\partial W_{ji}} - \sum_{\tilde{\mathbf{v}}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{\partial -\mathbf{c}^T\tilde{\mathbf{v}} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}}\right)}{\partial W_{ji}}$
terms other than the j-th term not matter in derivation of W_{ji}:
$= -\frac{1}{1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}(e^{b_j+\mathbf{W}_{j,:}\mathbf{v}})v_i + \sum_{\tilde{\mathbf{v}}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{1}{1+e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}}}(e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}})\tilde{v}_i$
$= -P(h_j = 1|\mathbf{v})\cdot v_i + \sum_{\tilde{\mathbf{v}}\in\{0,1\}^V} P(\tilde{\mathbf{v}})P(h_j = 1|\tilde{\mathbf{v}})\cdot \tilde{v}_i = -P(h_j = 1|\mathbf{v})\cdot v_i + E_{\tilde{\mathbf{v}}}[P(\tilde{h}_j = 1|\tilde{\mathbf{v}})\cdot \tilde{v}_i]$
Similarly: $-\frac{\partial \log P(\mathbf{v})}{\partial b_j} = \frac{\partial -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}\right)}{\partial b_j} - \sum_{\mathbf{v}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{\partial -\mathbf{c}^T\tilde{\mathbf{v}} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}}\right)}{\partial b_j}$
$= -\frac{1}{1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}}e^{b_j+\mathbf{W}_{j,:}\mathbf{v}} * 1 + \sum_{\tilde{\mathbf{v}}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{1}{1+e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}}}e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}} * 1$
$= -P(h_j = 1|\mathbf{v}) + \sum_{\tilde{\mathbf{v}}\in\{0,1\}^V} P(\tilde{\mathbf{v}})P(h_j = 1|\tilde{\mathbf{v}}) = -P(h_j = 1|\mathbf{v}) + E_{\tilde{\mathbf{v}}}[P(\tilde{h}_j = 1|\tilde{\mathbf{v}})]$
Similarly: $-\frac{\partial \log P(\mathbf{v})}{\partial c_j} = \frac{\partial -\mathbf{c}^T\mathbf{v} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\mathbf{v}}\right)}{\partial c_j} - \sum_{\mathbf{v}\in\{0,1\}^V} P(\tilde{\mathbf{v}})\frac{\partial -\mathbf{c}^T\tilde{\mathbf{v}} - \sum_{j=1}^{H}\log\left(1+e^{b_j+\mathbf{W}_{j,:}\tilde{\mathbf{v}}}\right)}{\partial c_j}$
$= -v_j - \sum_{\mathbf{v}\in\{0,1\}^V} P(\tilde{\mathbf{v}})(-\tilde{v}_j) = -v_i + E_{\tilde{\mathbf{v}}}[\tilde{v}_i]$ (QED).

Contrastive Divergence

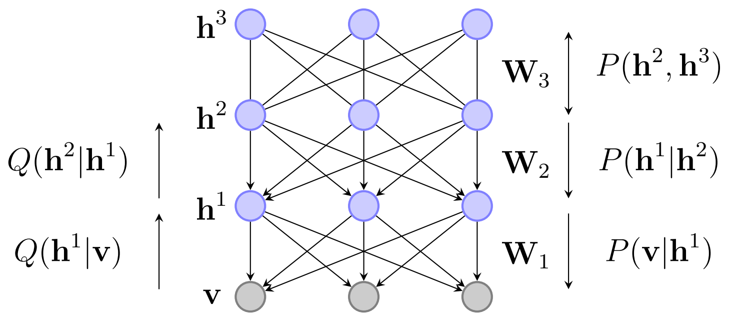
In positive phase, we can compute the term with a training sample $\mathbf{v}$. In negative phase, obtaining the expectation $E_{\tilde{\mathbf{v}}}[\cdot]$ is intractable. Contrastive Divergence replaces the expectation with a point estimate $(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$. We obtain the point $(\tilde{\mathbf{v}}, \tilde{\mathbf{h}})$ by Gibbs sampling. Given initial $\mathbf{v}^{(0)}$, we sample $\mathbf{h}^{(0)} \sim P(\mathbf{h}|\mathbf{v}^{(0)})$. Then we can sample $\mathbf{v}^{(1)} \sim P(\mathbf{v}|\mathbf{h}^{(1)})$. After k steps, we obtain $(\mathbf{v}^{(k)}, \mathbf{h}^{(k)})$. As $k \to \infty$, sample $(\mathbf{v}^{(k)}, \mathbf{h}^{(k)})$ are guaranteed to be accurate sample of $P(\mathbf{v}, \mathbf{h})$. Then we can replace $E_{\tilde{\mathbf{v}}}[P(\tilde{h}_j|\tilde{\mathbf{v}})]$ with $P(h_j^{(k)}|\mathbf{v}^{(k)})$.



The initial sample in Gibbs sampling can be random values. Since we eventually want $P(\mathbf{v}) \approx P_{train}(\mathbf{v})$, we start with a training example. Contrastive Divergence does not wait for the chain to converge. Only k-steps of Gibbs sampling is enough (so-called CD-k). $k = 1$ has been shown to work surprisingly well in practice.

Deep Belief Network

Top most layer is RBM. Others are directed belief networks.



$\mathbf{h}^3$
$\mathbf{h}^2$
$\mathbf{h}^1$
$\mathbf{v}$
$Q(\mathbf{h}^2|\mathbf{h}^1)$
$Q(\mathbf{h}^1|\mathbf{v})$
$\mathbf{W}_3$
$\mathbf{W}_2$
$\mathbf{W}_1$
$P(\mathbf{h}^2, \mathbf{h}^3)$
$P(\mathbf{h}^1|\mathbf{h}^2)$
$P(\mathbf{v}|\mathbf{h}^1)$

DBN training: We will use the layer-wise training algorithm. Namely, we first learn an RBM with an input layer $\mathbf{v}$ and a hidden layer $\mathbf{h}^1$, then freeze this RBM. We generate new training data $\mathbf{h}^1$ by sampling $p(\mathbf{h}^1|\mathbf{v})$ for every $\mathbf{v}$ in the training set (one per sample in the training set), and use this as the input to the second RBM ($\mathbf{h}^1$ and $\mathbf{h}^2$). The third RBM ($\mathbf{h}^2$ and $\mathbf{h}^3$) is also trained in this way.

DBN generation: After training a DBN, we could sample a vector $\mathbf{h}^2$ using Gibbs sampling from the third RBM. Given initial $\mathbf{h}^{2^{(0)}}$, we sample $\mathbf{h}^{3^{(0)}} \sim P(\mathbf{h}^3|\mathbf{h}^{2^{(0)}})$. Then we can sample $\mathbf{h}^{2^{(1)}} \sim P(\mathbf{h}^2|\mathbf{h}^{3^{(0)}})$. After k steps, we obtain $\mathbf{h}^{2^{(k)}}$. Then, using $\mathbf{h}^{2^{(k)}}$, we sample lower layers, $\mathbf{h}^1$ followed by $\mathbf{v}$, using sigmoid belief networks.

Programming (60 pts)

Two files, rbm.py and dbn.py, are provided. Only rbm.py will be evaluated in autograder, but you need to implement both to answer all questions below. You should read the instructions on top of these files, and the docstrings very carefully. You can change anything as you see fit in dbn.py, as this file will not be autograded. **Zip only *rbm.py* file and upload the zip file to GradeScope as shown in Figure 2.**

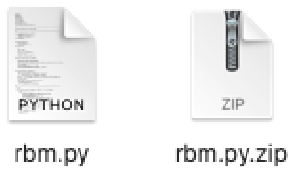


Figure 2: Submission Example.

As assistance through the programming assignment, we provided access to the hw2.ipynb jupyter notebook, where the paths and dependencies are managed. The final submission will still be this pdf.

In this assignment, the reconstruction error refers to:

$$\frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{m}(x_{ij}-\tilde{x}_{ij})^2} \tag{1}$$

where N is the number of samples in training set or validation set, m is the dimension of data vector, x is the input vector and $\tilde{x}$ is the reconstructed vector. Perform Gibbs sampling with $k = 1$ to obtain $\tilde{x}$.

We will build and train generative models (RBM and DBN) for MNIST dataset, which are images of 10 handwritten numbers. The images are 28 by 28, but they will be represented as 1D vectors with length 784. Data for training, validation and testing can be found in digitstrain.txt, digitsvalid.txt, and digitstest.txt.

For training, choose a reasonable learning rate (e.g. 0.1 or 0.01) and use your own judgement to decide when to stop training (i.e number of epochs or convergence criteria).

1. RBM task

Initialize an RBM with 100 hidden units.

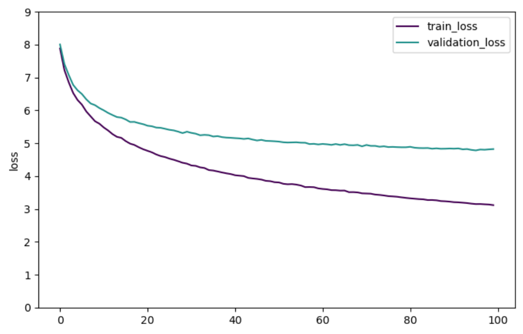
a.1) Autograder (20 points)

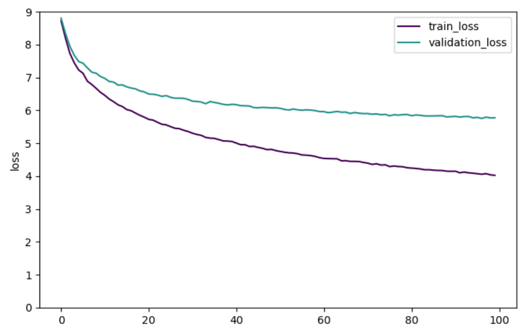
This question only requires your `rbm.py` Autograder submission.

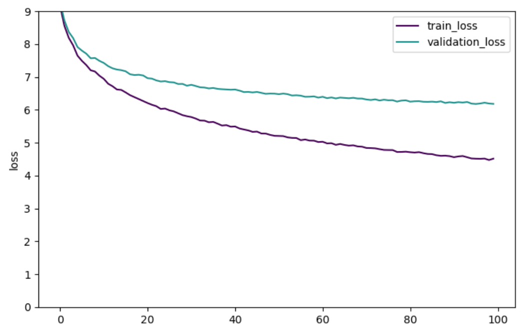
a.2) Training(5 points)

Try the RBM model with gibbs steps k as 1, 3, and 5. For each k, plot reconstruction error against the epoch number for training and validation on one plot. So you should include 3 plots here, each contains two curves for training and validation. How does k affect training convergence of the model?

Solution





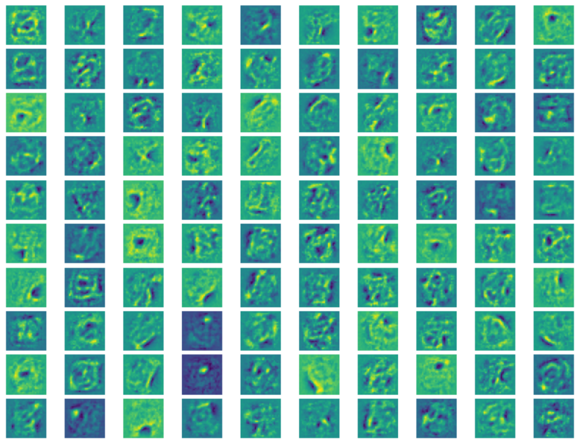


(top left K=1, top right k=3, bottom left k=5)
The validation loss for K=1,3,5 are: 4.8211, 5.7739, 6.1812 for 100 epochs and lr=0.01 and 300 hidden units. We can see that with small K=1, the model works incredibly well and as k increases loss increases.

b) Visualizing and understanding learned parameters (5 points)

Choose one model that you like, and visualize its learned W as 100 images that are 28-by-28 in pixel. Plot all of them in one figure. What are being plotted here? Do they exhibit any structure?

Solution



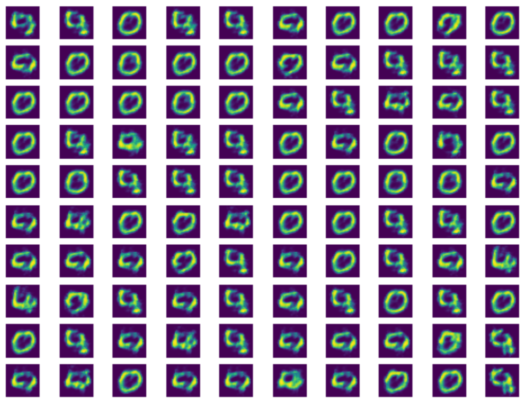
I choose the first model with k=1, it does not seem to some abstract structures.

c) Generation (5 points)

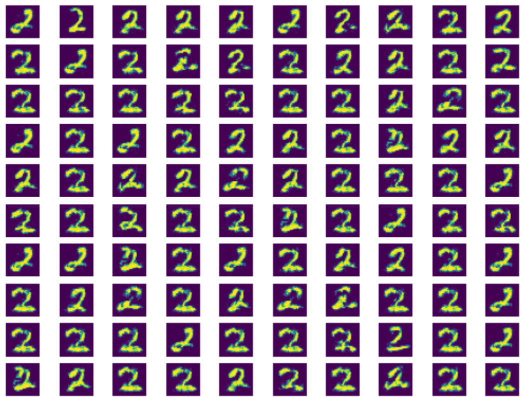
Set $k > 1000$ for this task. Display the 100 generated samples for digit images in one figure. Do they look like handwritten digits? What if you retrain your RBM on only 3 digits, say **1, 2** and **3**? If you train with $k = 1$ vs $k = 5$, do you see a difference in generated figures?

Solution

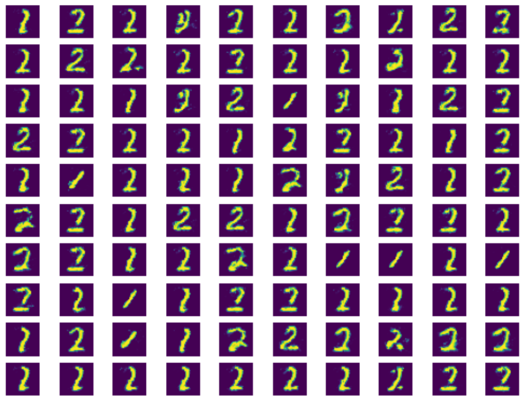
The figure below is 100 generated samples for the model trained by 10 digits and k=1 for training. The input for generation is using np.random.binomial() and k=20000 for generating. The plot is for v_{prob} not v_{sample} for smoother plotting. They look pretty good like handwritten digits, some like 4, some like 9, many like 0.



The figure below is 100 generated samples for the model trained by 3 digits (1,2,3) and k=1 for training. The input for generation is using np.random.binomial() and k=20000 for generating. Since the model is trained by only 1,2,3; therefore most of the generated digits look like 2.



The figure below is 100 generated samples for the model trained by 3 digits (1,2,3) and k=5 for training. The input for generation is using np.random.binomial() and k=20000 for generating. Unlike the graph generated by model with k=1, the model with k=5 have many 1s and 2s, not purely 2s.

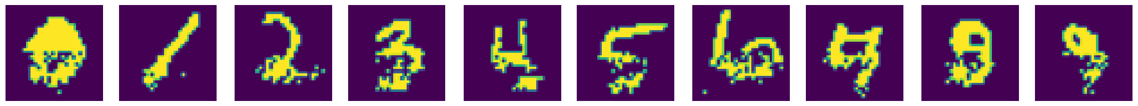


d) Conditional generation (5 points)

Only reveal the top half of MNIST images (data generation code is provided to you), and use the RBM to reconstruct the bottom half of the image. Note here when you do gibbs sampling, when you sample $\mathbf{v}$ condition on $\mathbf{h}$, part of $\mathbf{v}$ is known for sure. You need to inject these known value to the newly sampled $\mathbf{v}$.

Solution

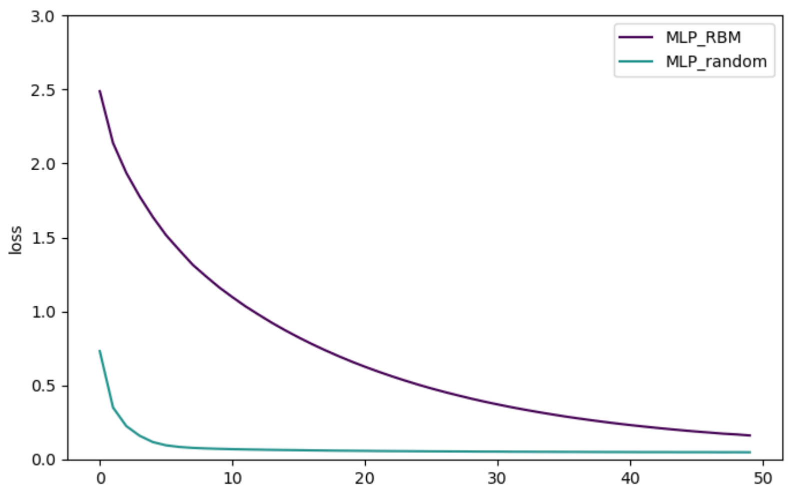
This is the conditional generated images based on the 10 digit code provided:



e) Supervised learning with RBM (10 points)

Take the RBM you have trained and initialize a 2-layer neural network, of which the first layer's weights are initialized using the RBM's weight. Compare the training trajectory of this RBM-initialized network with a randomly initialized network. Does the RBM-initialized network converge faster? Plot the training loss of these two networks in one figure.

Solution



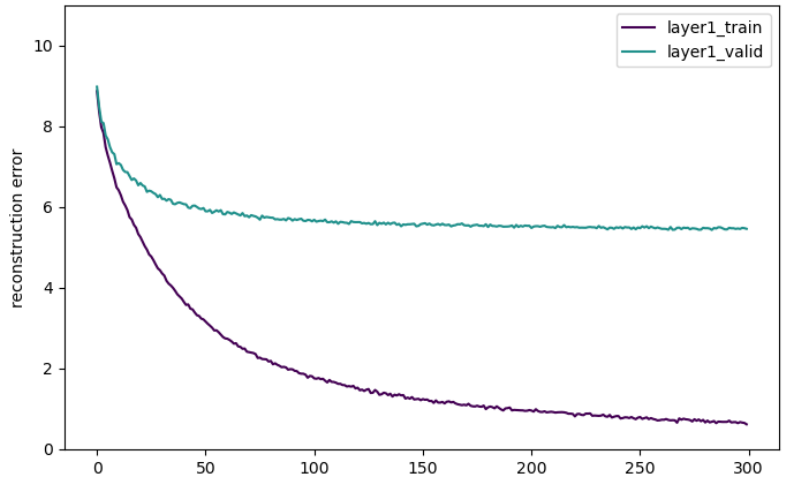
Actually, MLP with random initiation converges faster than MLP with RBM weights. I tried hidden layer with hidden units from 20,100, 300 to 1000; as number of hidden units increases, the original RBM has a smaller reconstruction error, but in contrast the starting error for MLP model if initialized by RBM weight also increases (the graph shows intial error of 2.5 for hidden units =100, but will be 8 if hidden units=1000); Therefore I suspect that RBM and MLP look at the image data from different abstractions, because RBM is a unsupervised approach focus on extracting the internal patterns of the training X, but MLP is a supervised approach focus on classfication. Therefore if the unsupervised pre-training is unrelated to the classification problem, it may hurt performance. The more precise the pre-train (means better RBM performance), the faraway the pre-train pattern deviates from (less relate to) classification problem. Thus, we have worse convergence.

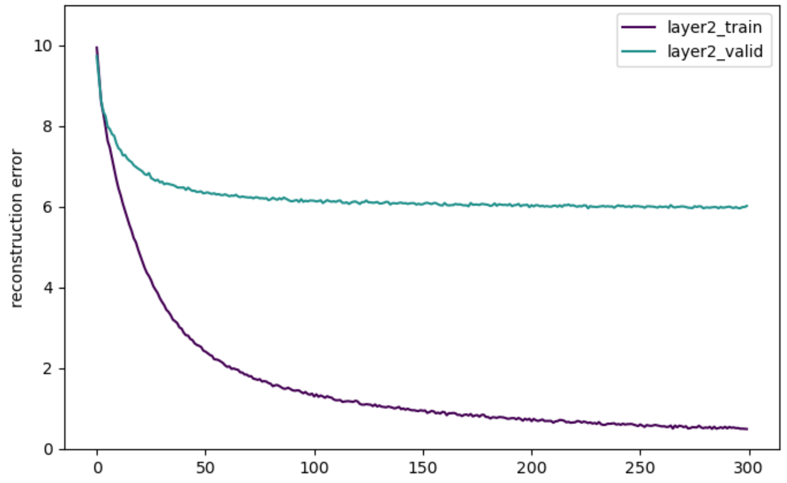
2. DBN task

Truncate our dataset and only retain images of digits **7**, **8**, and **9**. Build a DBN with two hidden layers with 500 and 784 units respectively, so there are two RBMs with 500 and 784 hidden units.

a) Training (5 points) Training this DBN with gibbs steps $k = 3$. For each RBM, plot reconstruction error against the epoch number for training and validation on one plot. So you should include 2 plots here, each contains two curves for training and validation.

Solution

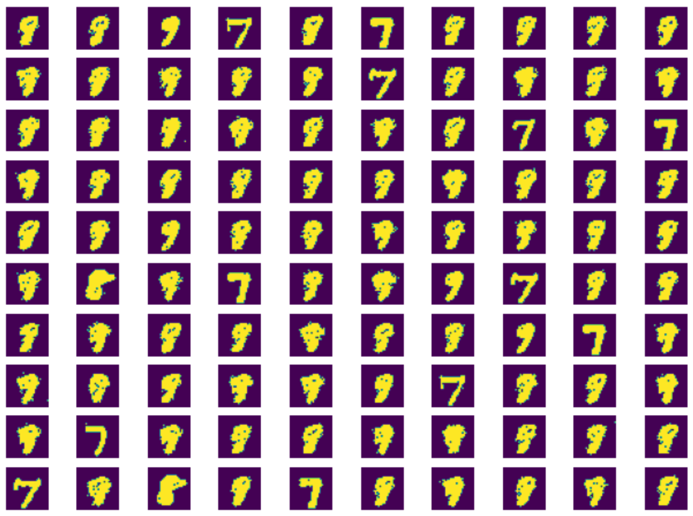




b) Generation (5 points)

Set $k > 1000$ for this task. Display the 100 generated samples for digit images in one figure. Do they look like handwritten digits?

Solution



Graph generated by use gibbs sampling from random data for top most RBM and sigmoid for lower layers. They look good, since the model is train by only 7, 8, 9 digit, the generated digit also looks like 7,8 or 9.

Collaboration Questions Please answer the following:

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies found here.

1. Did you receive any help whatsoever from anyone in solving this assignment? Is so, include full details.
   No
2. Did you give any help whatsoever to anyone in solving this assignment? Is so, include full details.
   No
3. Did you find or come across code that implements any part of this assignment ? If so, include full details even if you have not used that portion of the code.
   No