Homework #2

B07902028 資工二 林鶴哲

1. Let $q_n = 1 + \exp(-y_n(Az_n + B))$, we have $F(A, B) = \frac{1}{N}\sum_{n=1}^{N} \ln q_n$. Use chain rule to compute $\frac{\partial F}{\partial A}$ and $\frac{\partial F}{\partial B}$, we obtain

$$\begin{aligned}
\frac{\partial F}{\partial A} &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial F}{\partial q_n}\cdot\frac{\partial q_n}{\partial(-y_n(Az_n+B))}\cdot\frac{\partial(-y_n(Az_n+B))}{\partial A}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{q_n}\cdot\exp(-y_n(Az_n+B))\cdot(-y_nz_n)\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{\exp(-y_n(Az_n+B)}{1+\exp(-y_n(Az_n+B))}(-y_nz_n) \\
&= \frac{1}{N}\sum_{n=1}^{N}(-p_ny_nz_n)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial F}{\partial B} &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial F}{\partial q_n}\cdot\frac{\partial q_n}{\partial(-y_n(Az_n+B))}\cdot\frac{\partial(-y_n(Az_n+B))}{\partial B}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{q_n}\cdot\exp(-y_n(Az_n+B))\cdot(-y_n)\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{\exp(-y_n(Az_n+B)}{1+\exp(-y_n(Az_n+B))}\cdot(-y_n) \\
&= \frac{1}{N}\sum_{n=1}^{N}(-p_ny_n)
\end{aligned}$$

Hence we have $\nabla F(A, B) = (\frac{1}{N}\sum_{n=1}^{N}(-p_ny_nz_n), \frac{1}{N}\sum_{n=1}^{N}(-p_ny_n))$.

2. Before computing the second-order partial derivatives of $F(A, B)$, we first compute the derivative of $\theta(s)$

$$\theta'(s) = \frac{e^{-s}}{(1+e^{-s})^2} = \theta(s)\cdot\frac{e^{-s}}{1+e^{-s}} = \theta(s)\theta(-s) = \theta(s)(1-\theta(s))$$

Then we compute the components in $H(F)$

$$\begin{aligned}
\frac{\partial^2 F}{\partial A^2} &= \frac{\partial}{\partial A}\left(\frac{1}{N}\sum_{n=1}^{N}-p_ny_nz_n\right) = \frac{1}{N}\sum_{n=1}^{N}-y_nz_n(p_n)(1-p_n)(-y_nz_n) = \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2{z_n}^2 \\
\frac{\partial^2 F}{\partial A\partial B} &= \frac{\partial}{\partial B}\left(\frac{1}{N}\sum_{n=1}^{N}-p_ny_nz_n\right) = \frac{1}{N}\sum_{n=1}^{N}-y_nz_n(p_n)(1-p_n)(-y_n) = \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2z_n \\
\frac{\partial^2 F}{\partial B^2} &= \frac{\partial}{\partial B}\left(\frac{1}{N}\sum_{n=1}^{N}-p_ny_n\right) = \frac{1}{N}\sum_{n=1}^{N}-y_n(p_n)(1-p_n)(-y_n) = \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2
\end{aligned}$$

Hence we obtain the Hessian matrix $H(F) = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2{z_n}^2 & \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2z_n \\ \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2z_n & \frac{1}{N}\sum_{n=1}^{N}p_n(1-p_n){y_n}^2 \end{bmatrix}$

3. Since ${y_n}^2 = 1$, the Hessian matrix can also be written as $H(F) = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} p_n(1-p_n){z_n}^2 & \frac{1}{N}\sum_{n=1}^{N} p_n(1-p_n)z_n \\ \frac{1}{N}\sum_{n=1}^{N} p_n(1-p_n)z_n & \frac{1}{N}\sum_{n=1}^{N} p_n(1-p_n) \end{bmatrix}$.
Since $0 < p_n < 1$, we know that $H(F)_{11} > 0$, and by Cauchy's inequality, we have

$$\left(\sum_{n=1}^{N}\left(\sqrt{p_n(1-p_n){z_n}^2}\right)^2\right)\left(\sum_{n=1}^{N}\left(\sqrt{p_n(1-p_n)}\right)^2\right) \geq \left(\sum_{n=1}^{N} p_n(1-p_n)z_n\right)^2,$$

which implies $\det(H(F)) \geq 0$. By $H(F)_{11} \geq 0$ and $\det(H(F)) \geq 0$, we know that $H(F)$ is positive semi-definite.

4. Assume $x_0 = +1$. Then we write $\mathbf{w}$ as

$$\mathbf{w} = (w_0, w_1, \ldots, w_d) = (d-1, 1, \ldots, 1)$$

If $x_i = -1$ for all $i = 1, \ldots, d$, $g_A(x) = \text{sign}(d-1-d) = -1$.
If at least one of $x_i = 1$, then $\sum_{i=0}^{d} w_i x_i \geq d-1+1-(d-1) = 1$, hence $\text{sign}\left(\sum_{i=0}^{d} w_i x_i\right) = 1$.
Hence the perceptron is equivalent to the logical `OR`.

5. According to the slides of Lecture 12, we know that

$$\frac{\partial e_n}{\partial w_{ij}^{(l)}} = \begin{cases} \delta_1^{(L)} x_i^{(l-1)} = \left(-2(y_n - \tanh(s_1^{(l)}))\right)\left(\tanh'(s_1^{(l)})\right) x_i^{(l-1)} & \text{if } l = L \\ \left(\sum_k \left(\delta_k^{(l+1)}\right)\left(w_{jk}^{(l+1)}\right)\left(\tanh'(s_j^{(l)})\right)\right) x_i^{(l-1)} & \text{otherwise} \end{cases} \quad (*)$$

For any $1 \leq l \leq L-1$, $w_{jk}^{(l+1)} = 0$ and thus the gradient corresponding to $w_{ij}^{(l)}$ is 0.
Hence we only have to discuss $l = L$, i.e, the gradient component corresponding to $w_{i1}^{(L)}$ for $0 \leq i \leq d^{(l-1)}$: for any $i \neq 0$, $x_i^{(L-1)} = 0$. Thus, the corresponding gradient component is 0.
Now, we focus on $\frac{\partial e_n}{\partial w_{01}^{(L)}}$. If $y_n \neq \tanh(s_1^{(l)}) = 0$, all three terms in $(*)$ is not 0, so the corresponding gradient component might not be 0 in this case.
In sum, all gradient components except the corresponding component of $w_{01}^{(L)}$ are also 0.

6. I solve this problem with Dynamic Programming. Given a NNet with i units (containing the bias term) in layer-0, one output unit and j units left to use (the output unit excluded), define `dp[i][j]` to be the maximum number of weights that such network can have.
Clearly, we want to find `dp[12][48]`. Then we formulate the following DP transition:

$$\texttt{dp[i][j]} = \begin{cases} i & \text{if } j = 0 \text{ or } j = 1 \\ \max\limits_{2 \leq k \leq j} \left(i(k-1) + \texttt{dp[k][j-k]}\right) & \text{otherwise} \end{cases}$$

When $j = 0$ or $j = 1$, i.e, there are only 1 or 0 unit to use(including the output unit), the only possibility to generate weights is to connect the i units in layer-0 to the output unit, that is, an i-1 neuron network.
When there are more than 1 units left to use, we compute the maximum weights if we connect the current i units to the next k units ($2 \leq k \leq j$, and the bias term won't be connected by the current i units), and find the maximum among these possibilities.
After running the program, we find that `dp[12][48]` $= 877$. After tracing back the optimal solution, We find that the maximum number of weights is attained when a 12-29-19-1 NNet is built.
The result meets our intuition: Given fixed number of units, if we build a NNet with lots of layers, the number of units in each layer is few, so the weight such network generates would less than a NNet with less layers(but each has many neurons).
However, if we only build a NNet with few layer, take only 1 layer as an example, then there are only two terms in the final computation of the total generated weights (the last term even has the output # of unit "1"!). When maximum attained, the number of layers of NNet should not be too small or large.

7. Expand $\text{err}_n(\mathbf{w})$, we have

$$\begin{aligned}\text{err}_n(\mathbf{w}) &= \mathbf{x}_n^T\mathbf{x}_n - 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T\mathbf{x}_n + (\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\mathbf{x}_n) \\ &= \mathbf{x}_n^T\mathbf{x}_n - 2\mathbf{x}_n^T\mathbf{w}\mathbf{w}^T\mathbf{x}_n + \mathbf{x}_n^T\mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T\mathbf{x}_n \\ &= \mathbf{x}_n^T\mathbf{x}_n - 2\mathbf{w}^T(\mathbf{x}_n\mathbf{x}_n^T)\mathbf{w} + (\mathbf{w}^T\mathbf{x}_n)^2\mathbf{w}^T\mathbf{w}\end{aligned}$$

Then compute $\nabla\text{err}_n(\mathbf{w})$:

$$\begin{aligned}\nabla\text{err}_n(\mathbf{w}) &= 0 + \big(-2(2(\mathbf{x}_n\mathbf{x}_n^T)\mathbf{w})\big) + \big(2(\mathbf{x}_n^T\mathbf{w})\mathbf{x}_n(\mathbf{w}^T\mathbf{w}) + 2(\mathbf{w}^T\mathbf{x}_n)^2)\mathbf{w}\big) && (*) \\ &= -4\mathbf{x}_n^T\mathbf{w}\mathbf{x}_n + 2(\mathbf{x}_n^T\mathbf{w})(\mathbf{x}_n\mathbf{w}^T\mathbf{w} + (\mathbf{x}_n^T\mathbf{w})\mathbf{w}) && \text{(simplification)}\end{aligned}$$

where (*) is obtained by computing the gradient term by term.
The first term is trivial, since it's a constant to $\mathbf{w}$; the second term is by Lecture 9, MLF, slides #9:

$$\begin{aligned}E_{in}(\mathbf{w}) &= \mathbf{w}^T A\mathbf{w} \\ \nabla E_{in}(\mathbf{w}) &= 2A\mathbf{w}, \end{aligned} \tag{1}$$

and the last term is by chain rule and (1).

8. Expand $E_{in}(\mathbf{w})$, we obtain

$$\begin{aligned}E_{in}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}\big(\mathbf{x}_n^T\mathbf{x}_n - 2(\mathbf{w}\mathbf{w}^T(\mathbf{x}_n+\epsilon_n))^T\mathbf{x}_n + (\mathbf{w}\mathbf{w}^T(\mathbf{x}_n+\epsilon_n))^T(\mathbf{w}\mathbf{w}^T(\mathbf{x}_n+\epsilon_n))\big) \\ &= \frac{1}{N}\sum_{n=1}^{N}\big(\mathbf{x}_n^T\mathbf{x}_n - 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T\mathbf{x}_n + (\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)\big) \\ &\quad + \frac{1}{N}\sum_{n=1}^{N}\big(-2(\mathbf{w}\mathbf{w}^T\epsilon_n)^T\mathbf{x_n} + 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n) + (\mathbf{w}\mathbf{w}^T\epsilon_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n)\big) \\ &= \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2 + \frac{1}{N}\sum_{n=1}^{N}\big(-2(\mathbf{w}\mathbf{w}^T\epsilon_n)^T\mathbf{x_n} + 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n) + (\mathbf{w}\mathbf{w}^T\epsilon_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n)\big)\end{aligned}$$

Then, we compute the expected value of each term. Trivially,

$$\mathbb{E}\left[\frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2\right] = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \mathbf{w}\mathbf{w}^T\mathbf{x}_n\|^2$$

As for the second term, we have that

$$\begin{aligned}
\Omega(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}\left[-2(\mathbf{w}\mathbf{w}^T\epsilon_n)^T\mathbf{x_n} + 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n) + (\mathbf{w}\mathbf{w}^T\epsilon_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n)\right] \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbb{E}\left[-2(\mathbf{w}\mathbf{w}^T\epsilon_n)^T\mathbf{x_n}\right] + \mathbb{E}\left[2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n)\right] + \mathbb{E}\left[(\mathbf{w}\mathbf{w}^T\epsilon_n)^T(\mathbf{w}\mathbf{w}^T\epsilon_n)\right]\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbb{E}\left[\epsilon_n^T\right](-2\mathbf{w}\mathbf{w}^T\mathbf{x_n}) + 2(\mathbf{w}\mathbf{w}^T\mathbf{x}_n)^T(\mathbf{w}\mathbf{w}^T)\mathbb{E}\left[\epsilon_n\right] + \mathbb{E}\left[\epsilon_n^T\mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T\epsilon_n\right]\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left(0 + 0 + (\mathbf{w}^T\mathbf{w})\mathbb{E}\left[\mathbf{w}^T\epsilon_n\epsilon_n^T\mathbf{w}\right]\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left((\mathbf{w}^T\mathbf{w})\mathbf{w}^T\left(\mathbb{E}\left[\begin{pmatrix}\epsilon_{n1}\epsilon_{n1} & \dots & \epsilon_{n1}\epsilon_{nn}\\ \vdots & & \vdots \\ \epsilon_{nn}\epsilon_{n1} & \dots & \epsilon_{nn}\epsilon_{nn}\end{pmatrix}\right]\right)\mathbf{w}\right) \\
&= \frac{1}{N}\sum_{n=1}^{N}\left((\mathbf{w}^T\mathbf{w})\mathbf{w}^T I_n\mathbf{w}\right) \\
&= (\mathbf{w}^T\mathbf{w})^2
\end{aligned}$$

9. We first compute the p-th component of $g(\mathbf{x})$ as follows:

$$g(\mathbf{x})_p = \sum_{q=1}^{\tilde{d}} u_{pq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)$$

Hence the error function E is

$$\sum_{p=1}^{d}\left(x_p - \sum_{q=1}^{\tilde{d}} u_{pq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)\right)^2$$

10. From 9, we compute $\frac{\nabla E_9(\mathbf{u})}{u_{ij}}$

$$\begin{aligned}
\frac{\partial E_9(\mathbf{u})}{u_{ij}} &= \sum_{p\neq i} 2\left(x_p - \sum_{q=1}^{\tilde{d}} u_{pq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)\right)\left(-u_{pj}\tanh'\left(\sum_{r=1}^{d} u_{rj}x_r\right)x_i\right) && (p\neq i) \\
&\quad + 2\left(x_i - \sum_{q=1}^{\tilde{d}} u_{iq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)\right)\left(-\tanh'\left(\sum_{r=1}^{d} u_{rj}x_r\right) - \tanh'\left(\sum_{r=1}^{d} u_{rq}x_r\right)x_i u_{ij}\right) \\
&&& (p=i, q=j) \\
&= \sum_{p=1}^{d} 2\left(x_p - \sum_{q=1}^{\tilde{d}} u_{pq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)\right)\left(-u_{pj}\tanh'\left(\sum_{r=1}^{d} u_{rj}x_r\right)x_i\right) \\
&\quad + 2\left(x_i - \sum_{q=1}^{\tilde{d}} u_{iq}\tanh\left(\sum_{r=1}^{d} u_{rq}x_r\right)\right)\left(-\tanh'\left(\sum_{r=1}^{d} u_{rj}x_r\right)\right) && (*)
\end{aligned}$$

Then, write $E_{10}(\mathbf{w})$ as

$$\sum_{p=1}^{d}\left(x_p - \sum_{q=1}^{\tilde{d}} w_{qp}^{(2)}\tanh\left(\sum_{r=1}^{d} w_{rq}^{(1)}x_r\right)\right)^2$$

Compute $\frac{\partial E_{10}(\mathbf{w})}{\partial w_{ij}^{(1)}}$:

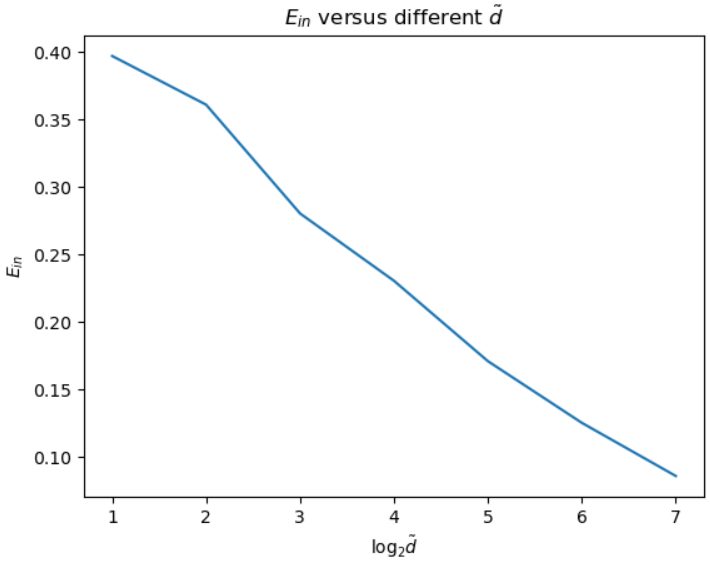
$$\frac{\partial E_{10}(\mathbf{w})}{\partial w_{ij}^{(1)}} = \sum_{p=1}^{d} 2 \left(x_p - \sum_{q=1}^{\tilde{d}} w_{qp}^{(2)} \tanh \left(\sum_{r=1}^{d} w_{rq}^{(1)} x_r \right) \right) \left(w_{jp}^{(2)} \tanh' \left(\sum_{r=1}^{d} w_{rq}^{(1)} x_r \right) x_i \right) \qquad (r = i, q = j)$$

and

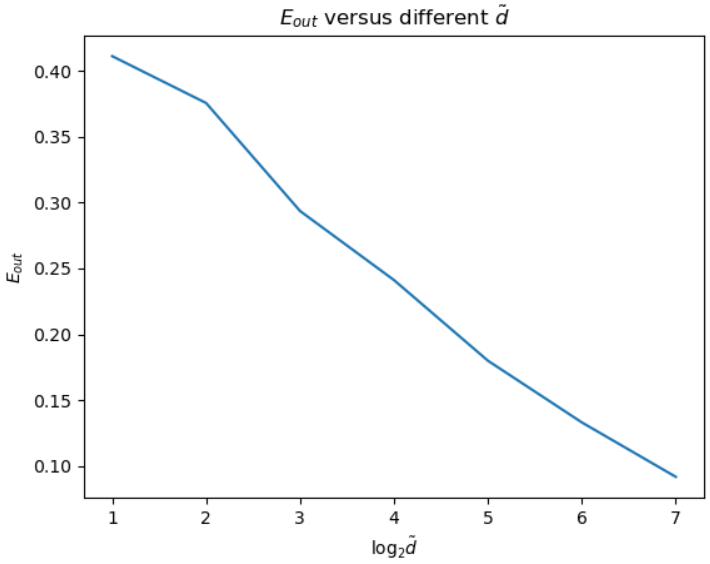
$$\frac{\partial E_{10}(\mathbf{w})}{\partial w_{ji}^{(2)}} = 2 \left(x_i - \sum_{q=1}^{\tilde{d}} w_{qi}^{(2)} \tanh \left(\sum_{r=1}^{d} w_{rq}^{(1)} x_r \right) \right) \left(- \tanh' \left(\sum_{r=1}^{d} w_{rj}^{(1)} x_r \right) \right) \qquad (q = i, p = j)$$

Now, with the condition that $u_{ij} = w_{ij}^{(1)} = w_{ji}^{(2)}$, we can see that the first term with $\sum$ in (*) is equivalent to $\frac{\partial E_1 0(\mathbf{w})}{w_{ij}^{(1)}}$, and the second term is equivalent to $\frac{\partial E_1 0(\mathbf{w})}{w_{ji}^{(2)}}$, which is the desired result.

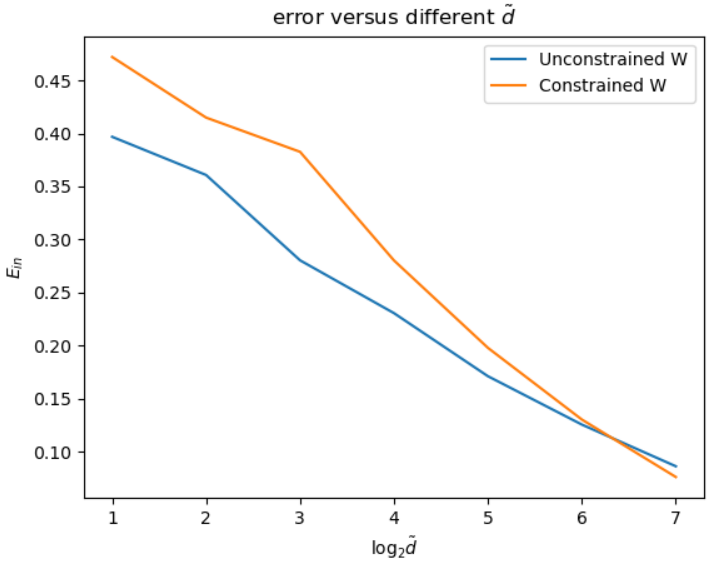
11. We can see that E_{in} decreases as $\tilde{d}$ increases. As $\tilde{d}$ increases, we "preserve" more information in the hidden layer. Thus, we can get a more similar $g(\mathbf{x})$ when more neurons in the middle layer is given.



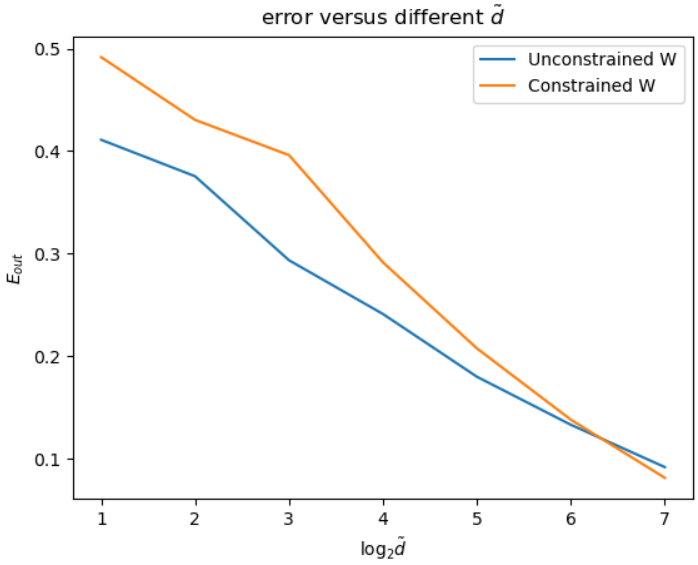
12. No matter what $\tilde{d}$ is, E_{out} is slightly larger than E_{in} in problem 11, which shows that there's no overfitting. The findings is similar to the previous problem: as $\tilde{d}$ grows, more features can be kept in the model so that it's more likely that the autoencoder generates a similar $g(\mathbf{x})$ with $\mathbf{x}$.



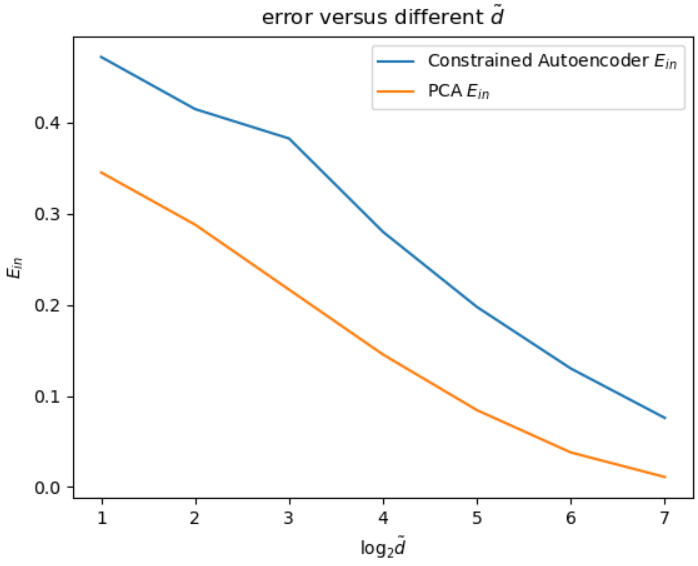
13. When $\log_2 \tilde{d}$ is smaller than 7, E_{in} of unconstrained W is smaller than E_{in} of constrained W, which shows that if we choose to preserve not so many features in the autoencoder, unconstrained W could have more choice to choose which information to preserve. However, when $\tilde{d} = 7$, constrained W performs better than the unconstrained one, showing that a little regularization is helpful when there are more neurons in the middle layer.



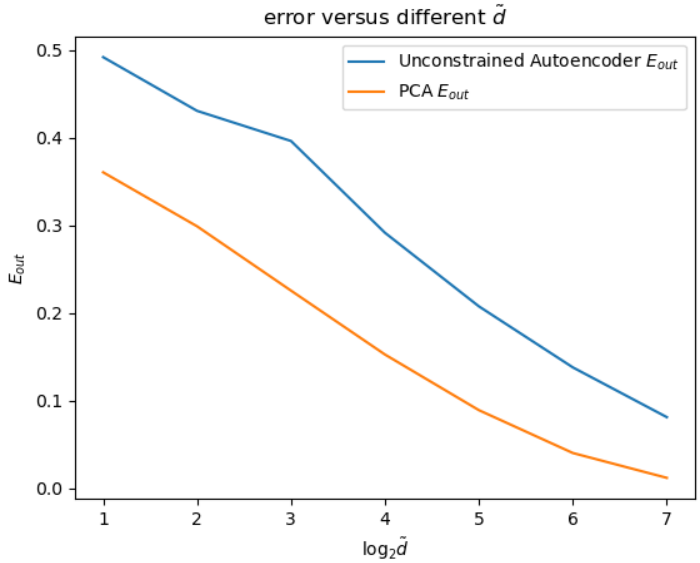
14. Again, E_{in} in problem 11, 13 is both slightly smaller than the E_{out} in problem in problem 12, 14, respectively, showing that the model performs not bad. The comparison of unconstrained W and constrained W is similar to the previous problem: When $\log_2 \tilde{d} \leq 6$, unconstrained W model performs better than the constrained W one; when $\log_2 \tilde{d} = 7$, constrained W model performs better. For the two problem (13, 14), we conclude that when the neuron in the middle layer is not so many, the autoencoder should have more weight to choose; when there are more neurons, then some regularization adding on the model will improve the performance.



15. No matter what the value of $\tilde{d}$ is, the E_{in} of PCA is smaller than the E_{in} of autoencoder, showing that linear model is enough for this problem. Also, one can see that the difference of "E_{in} of Autoencode minus E_{in} of PCA" decreases as $\tilde{d}$ increases, showing that with more information preserved, more powerful models can be used in such cases.



16. E_{out} of PCA decreases as $\tilde{d}$ increases. Also, the E_{out} is slightly higher than E_{in} of autoencoder. No matter what the value of $\tilde{d}$ is, the E_{out} of PCA is smaller than the E_{out} of autoencoder. Combining both 15 and 16, we find that for this problem, linear model is enough, since PCA is a linear model and the autoencoder is a non-linear one.



17. We prove the statement by showing that

$$\Delta \log_2 N + \log_2 \frac{3}{2} < N. \tag{17.1}$$

If (17.1) holds, take exponential with base 2 then we have

$$N^\Delta + 1 < 2^{(\Delta \log_2 N)+\log_2 \frac{3}{2}} = \frac{3}{2} N^\Delta < 2^N \tag{17.2}$$

Hence we start to prove (17.1). Write (17.1) to be

$$\Delta < \frac{N - \log_2 \frac{3}{2}}{\log_2 N} \tag{17.3}$$

for $\Delta \geq 2$ and $N \geq 3\Delta \log_2 \Delta \geq 6$.
We first compute the derivative of the right hand side of (17.3). For $N \geq 6$,

$$\left(\frac{N - \log_2 \frac{3}{2}}{\log_2 N}\right)' = \frac{\log_2 N - \frac{N - \log_2(1.5)}{N \ln 2}}{(\log_2 N)^2} \geq \frac{\log_2 N - \frac{N}{N \ln 2}}{(\log_2 N)^2} \geq \frac{\log_2 N - \frac{1}{\ln 2}}{(\log_2 N)^2} > 0,$$

which implies that the function is increasing for $N \geq 6$. So, it suffices to show that

$$\Delta < \frac{3\Delta \log_2 \Delta - \log_2 \frac{3}{2}}{\log_2(3\Delta \log_2 \Delta)} \tag{17.4}$$

Observe that

$$\frac{\frac{3\Delta \log_2 \Delta - \log_2 \frac{3}{2}}{\log_2(3\Delta \log_2 \Delta)}}{\Delta} = \frac{3 \log_2 \Delta - \frac{\log_2 \frac{3}{2}}{\Delta}}{\log_2(3\Delta \log_2 \Delta)} \geq \frac{3 \log_2 \Delta - \frac{\log_2 \frac{3}{2}}{2}}{\log_2(3\Delta \log_2 \Delta)} = \frac{\log_2\left(\sqrt{\frac{2}{3}}\Delta^3\right)}{\log_2(3\Delta \log_2 \Delta)} > 1$$

The last inequality is by

$$\frac{\sqrt{\frac{2}{3}}\Delta^3}{3\Delta \log_2 \Delta} = \frac{\sqrt{\frac{2}{3}}\Delta^2}{3 \log_2 \Delta} \geq \frac{\sqrt{2} \times 4}{3\sqrt{3}} > 1$$

Thus, we complete the proof.

18. Assume for contradiction that the VC dimension of $\mathcal{H}_{3A}$ is no less than $3 \cdot (3(d+1)+1)\log_2(3(d+1)+1)$. That is, the hypothesis set can shatter some $N = 3 \cdot (3(d+1)+1)\log_2(3(d+1)+1)$ points. That is, for each dichotomy among all 2^N possibilities, there exists a corresponding $g \in \mathcal{H}_{3A}$ such that g generates such dichotomy on these N points.

Now, let's look at the input layer and the first layer of the NNet. If we just consider all d+1 neurons in L_0(the bias unit included) and a single neuron in L_1, it works as a d-dimensional perceptron, which has a VC dimension of d+1.

From Lecture 7 of Machine Learning Foundation, we know that for any N points, the number of dichotomies generated by a d-perceptron is bounded by the bounding function $B(N, d+2)$.

Now, we focus on layer 1. If we regard each different $(\mathbf{x}_1^{(1)}, \mathbf{x}_2^{(1)}, \ldots, \mathbf{x}_N^{(1)})$ as a dichotomy, by the number of dichotomies of a d-perceptron is bounded by $B(N, d+2)$, at most $B(N, d+2)^3$ dichotomies was generated by different assignment of $\mathbf{w}^{(0)}$. Since the # of dichotomies is less than $B(N, d+2)^3$, the # of dichotomies of layer 2 is less than $B(N, d+2)^3$, either.

At page 2 of Lecture 7, we also have that for $N \geq 2$ and $d \geq 1$,

$$B(N, d+2) \leq \sum_{n=0}^{d+1} \binom{N}{n} \leq N^{d+1}. \tag{1}$$

In this problem, $d \geq 1$ and hence $N \geq 2$, so (1) can be applied.

From the above derivation, we know that the number of dichotimies that $\mathcal{H}_{3A}$ can generates on any N points is no more than $B(N, d+2)^3$.

Hence we then have

$$m_{\mathcal{H}_{3A}}(N) \leq B(N, d+2)^3 \leq N^{3(d+1)+1} + 1 < 2^N \tag{2}$$

The last inequality is by what we've proved in problem 17.

(2) shows a contradiction from our assumption. Hence, we complete the proof.