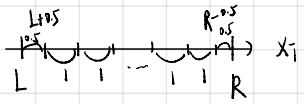
1. $g_{s,i,\theta}(x) = s \cdot \mathrm{sign}(x_i - \theta) \Rightarrow$ 

$S = \{+1, -1\}$

$i = \{1, \dots, d\}$

$\theta = \{L+0.5, L+1.5, \dots R-0.5\}$ $L, R \in \mathbb{Z}$

$$K_{ds}(x, x') = \left(g_{+1,1,\theta_1}(x), g_{+1,1,\theta_2}(x), \dots, g_{+1,1,\theta_k}(x), g_{-1,d,\theta_k}(x)\right) \cdot \left(g_{+1,1,\theta_1}(x'), g_{+1,1,\theta_2}(x'), \dots, g_{+1,1,\theta_k}(x'), g_{-1,d,\theta_k}(x')\right)$$

$$= \sum_{s=\{+1,-1\}} \sum_{i=1}^{d} \sum_{\theta=L+0.5}^{R-0.5} \left[g_{s,i,\theta}(x)\, g_{s,i,\theta}(x')\right]$$

$$= \sum_{i=1}^{d} \sum_{\theta=L+0.5}^{R-0.5} \left[(+1)\,\mathrm{sign}(x_i-\theta)(+1)\,\mathrm{sign}(x_i'-\theta) + (-1)\,\mathrm{sign}(x_i-\theta)(-1)\,\mathrm{sign}(x_i'-\theta)\right]$$

$$= 2\sum_{i=1}^{d} \sum_{\theta=L+0.5}^{R-0.5} \left[\mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta)\right]$$

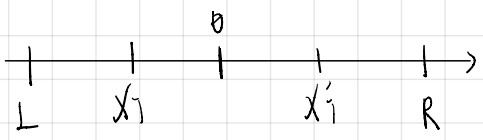
$\Downarrow$

$$\begin{cases} x_i, x_i' > \theta \\ x_i, x_i' < \theta \end{cases} \rightarrow \mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta) = 1$$

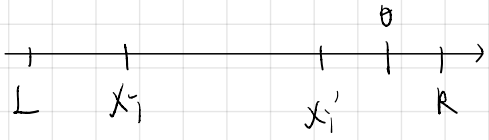
$$\begin{cases} x_i > \theta \ \& \ x_i' < \theta \\ x_i < \theta \ \& \ x_i' > \theta \end{cases} \rightarrow \mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta) = -1$$

$$x_i = 0 \ \text{ or } \ x_i' = 0 \rightarrow \mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta) = 0$$

ex.

θ is in $[x_i, x_i']$ $\Downarrow$ 

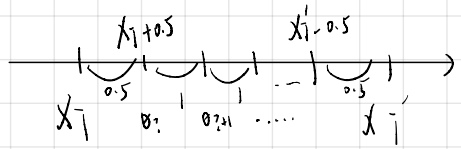
$\Rightarrow -1$

θ is out side of $[x_i, x_i']$ 

$\Rightarrow 1$

Hence we need to find how many θ is in and out of the interval $[x_i, x_i')$ (suppose $x_i \le x_i'$) to get the value of $\sum_{\theta=L+0.5}^{R-0.5} \mathrm{sign}(x_i-\theta)\,\mathrm{sign}(x_i'-\theta)$

there will be $R-L$ θs in the interval $[L+0.5, R-0.5]$



$\Rightarrow$ there will be $|(x_i'-0.5)-(x_i+0.5)+1|$ θs in the interval $[x_i, x_i']$ $\Rightarrow |x_i'-x_i|$

$$= 2\sum_{i=1}^{d} \left(\text{count of } \theta \text{ outside of } [x_i, x_i'] - \text{count of } \theta \text{ in } [x_i, x_i'] \right)$$

$$= 2\sum_{i=1}^{d} \left\{ \left[(R-L) - |x_i'-x_i| \right] - |x_i'-x_i| \right\}$$

$$= 2\sum_{i=1}^{d} \left((R-L) - 2|x_i'-x_i| \right)$$

$$= 2d(R-L) - 4\|x'-x\|_1 \quad \text{※}$$

2.

Dual of soft-margin SVM: $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m \tilde{K}(z_n,z_m) - \sum_{n=1}^{N}\alpha_n$

subject to $\sum_{n=1}^{N} y_n\alpha_n = 0$;

$0 \le \alpha_n \le \tilde{C}$, for $n=1,2,\dots,N$;

implicitly $w = \sum_{n=1}^{N}\alpha_n y_n z_n$;

$\beta_n = C - \alpha_n$ for $n=1,2,\dots,N$

$g_{SVM}(x) = \text{sign}\left(\sum_{\text{SV indices } n}\alpha_n y_n \tilde{K}(x_n,x) + b\right)$

expand K' $\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m (u K(z_n,z_m)+v) - \sum_{n=1}^{N}\alpha_n$

subject to $\sum_{n=1}^{N} y_n\alpha_n = 0$, $0 < \alpha_n < \frac{C}{u}$

expand $\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m u K(z_n,z_m) + \frac{v}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m - \sum_{n=1}^{N}\alpha_n$

subject to $\sum_{n=1}^{N} y_n\alpha_n = 0$, $0 < \alpha_n < \frac{C}{u}$

$\times u$ $\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(u\alpha_n)(u\alpha_m) y_n y_m (K(z_n,z_m)+v) - \sum_{n=1}^{N}(u\alpha_n)$

$u > 0$

Subject to $\sum_{n=1}^{N} u y_n\alpha_n = 0$, $0 < (u\alpha_n) < C$

take $\alpha_n' = u\alpha_n$ $\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n'\alpha_n' y_n y_m (K(z_n,z_m)+v) - \sum_{n=1}^{N}\alpha_n'$

Subject to $\sum_{n=1}^{N} y_n\alpha_n' = 0$, $0 < \alpha_n' < C$

expand $\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n'\alpha_n' y_n y_m K(z_n,z_m) + \frac{v}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n'\alpha_n' y_n y_m - \sum_{n=1}^{N}\alpha_n'$

Subject to $\sum_{n=1}^{N} y_n\alpha_n' = 0$, $0 < \alpha_n' < C$

$= \frac{v}{2}\left(\sum_{n=1}^{N}\alpha_n' y_n\right)\left(\sum_{m=1}^{N}\alpha_m' y_m\right) = 0$

$\rightarrow$ $\min_{\alpha} \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n'\alpha_n' y_n y_m K(z_n,z_m) - \sum_{n=1}^{N}\alpha_n'$

Subject to $\sum_{n=1}^{N} y_n\alpha_n' = 0$, $0 < \alpha_n' < C$

Hence we can solve soft-margin SVM by the new optimization problem above.

$$\min_{\alpha} \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m K(z_n, z_m) - \sum_{n=1}^{N} \alpha_n$$

$$\text{subject to } \sum_{n=1}^{N} y_n \alpha_n = 0, \quad 0 \le \alpha_n \le C$$

which is exactly the same problem as using kernel $= K(x, x')$ and $C = C$

Hence $\tilde{g}_{SVM}(x) = g_{SVM}(x)$ ∎

3.

$$E_{out}(g) = avg(E_{out}(g_t)) - avg(\varepsilon(g_t - g)^2)$$

$$= \frac{1}{17}\sum_{t=1}^{17} e_t - avg(\varepsilon(g_t - g)^2)$$

$$\Rightarrow \frac{E_{out}(g)}{E} = \frac{1}{17} - \frac{avg(\varepsilon(g_t - g)^2)}{E}$$

$\because \varepsilon(g_t - g)^2 \geq 0$ (it equals to 0 when under any condition, all $g_t = g$, which means g_t = every other g_i in g)

$\therefore avg(\varepsilon(g_t - g)^2) \geq 0$

Hence $\frac{E_{out}(g)}{E} \geq \frac{1}{17}$ ※

4.

Probability for an example not to be sampled, if we only sample 1 example: $\frac{N-1}{N}$

Probability for an example not to be sampled, if we sample $\frac{3}{4}N$ examples: $\left(\frac{N-1}{N}\right)^{\frac{3}{4}N}$

$$\lim_{N\to\infty}\left(\frac{N-1}{N}\right)^{\frac{3}{4}N} = \lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{\frac{3}{4}N} = \lim_{N\to\infty}\left[\left(1-\frac{1}{N}\right)^{N}\right]^{\frac{3}{4}} = \left[\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{\frac{1}{N}}\right]^{-\frac{3}{4}} = e^{-0.75} \doteq 0.47237$$

※

5. 98% data +, 2% data −

suppose $N=100$, we'll have 98 $y_n>0$, 2 $y_n<0$

then $\frac{\sum_{n:y_n>0} u_n^{(2)}}{\sum_{n:y_n<0} u_n^{(2)}} = \frac{\sum_{n:y_n>0} u_n^{(1)} / \diamond_1}{\sum_{n:y_n<0} u_n^{(1)} \times \diamond_1}$, where $\diamond_1 = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{98\%}{2\%}} = 7$

$$= \frac{\frac{1}{100} \times 98 \times \frac{1}{7}}{\frac{1}{100} \times 2 \times 7} = 1$$ ※

b. $U_t = \sum_{n=1}^{N} u_n^{(t)}$

$$\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} \left[|y_n \neq g_t(x_n)| \right]}{\sum_{n=1}^{N} u_n^{(t)}}$$

$$\Rightarrow U_t \cdot \epsilon_t = \sum_{n=1}^{N} u_n^{(t)} \left[|y_n \neq g_t(x)| \right], \quad U_t \cdot (1-\epsilon_t) = \sum_{n=1}^{N} u_n^{(t)} \left[|y_n = g_t(x)| \right]$$

$$U_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} = \sum_{n=1}^{N} u_n^{(t)} \left[|y_n \neq g_t(x)| \right] \times \diamond_t + \sum_{n=1}^{N} u_n^{(t)} \left[|y_n = g_t(x)| \right] / \diamond_t$$

$$= U_t \cdot \epsilon_t \times \diamond_t + U_t \cdot (1-\epsilon_t) / \diamond_t$$

$$= U_t \cdot \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \times \epsilon_t + U_t \cdot (1-\epsilon_t) \cdot \sqrt{\frac{\epsilon_t}{1-\epsilon_t}}$$

$$= 2 U_t \cdot \sqrt{\epsilon_t (1-\epsilon_t)}$$

$$\Rightarrow \frac{U_{t+1}}{U_t} = 2\sqrt{\epsilon_t (1-\epsilon_t)}$$

Then $\frac{U_{T+1}}{U_T} \times \frac{U_T}{U_{T-1}} \times \cdots \times \frac{U_2}{U_1} = 2\sqrt{\epsilon_T (1-\epsilon_T)} \cdot 2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdots 2\sqrt{\epsilon_1 (1-\epsilon_1)}$

$$= 2^T \sqrt{\epsilon_T (1-\epsilon_T)} \sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \cdots \sqrt{\epsilon_1 (1-\epsilon_1)}$$ ※※

1. let $\vec{S}_t = [s_1, s_2, \dots, s_N]$, $\vec{G}_t = [g_t(x_1), g_t(x_2), \dots g_t(x_N)]$, $\vec{Y} = [y_1, y_2 \dots y_N]$
   s at round t

the optimal η is obtain by solving $\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} ((y_n - s_n) - \eta g_t(x_n))^2$

hence $\eta = \frac{\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)}{\sum_{n=1}^{N} g_t^2(x_n)} = \frac{\vec{G}_t (\vec{Y} - \vec{S}_{t-1})^T}{\vec{G}_t \vec{G}_t^T} = \alpha_t$

we know $\vec{S}_t = \alpha_t \vec{G}_t + \vec{S}_{t-1}$

then $\sum_{n=1}^{N} s_n g_t(x_n) = \vec{S}_t \vec{G}_t^T = (\alpha_t \vec{G}_t + \vec{S}_{t-1}) \vec{G}_t^T = \alpha_t \vec{G}_t \vec{G}_t^T + \vec{S}_{t-1} \vec{G}_t^T$

$= \vec{G}_t (\vec{Y} - \vec{S}_{t-1})^T + \vec{G}_t \vec{S}_{t-1}^T$

$= \vec{G}_t \vec{Y}^T$

$= \sum_{n=1}^{N} g_t(x_n) y_n$

8.

By gradient descent update function : $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta\, x_i^{(l-1)}\, \delta_i^{(l)}$

Because $w^{(l)} = 0$ (initialization), we know $w_{ij}^{(l)} = 0$, for any i, j, l.

Get $\delta_j^{(l)} = \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)})(w_{jk}^{(l+1)})(\tanh' s_j^{(l)})$

$= \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)})(w_{jk}^{(l+1)})\left(\tanh'\left(\sum_{i=1}^{d^{(l-1)}} \underbrace{w_{ij}^{(l)}}_{0}\, x_i^{(0)}\right)\right)$

$= 0$

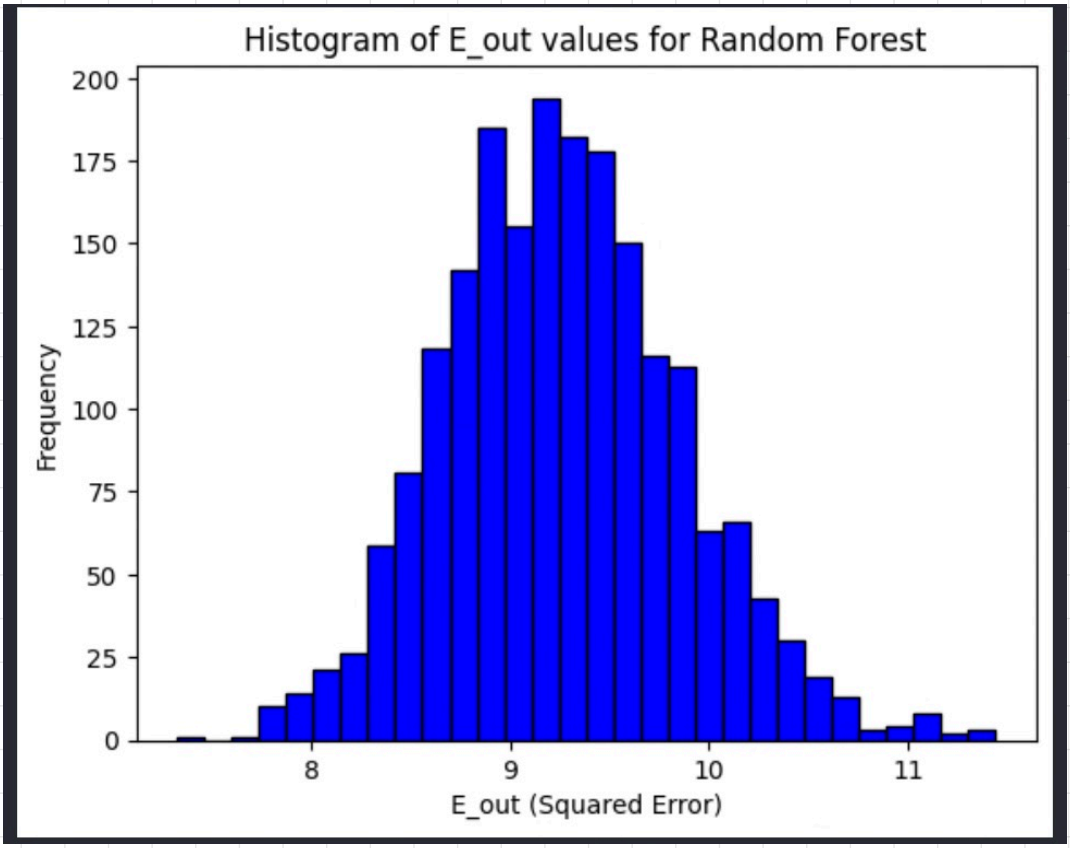
Hence, the update function becomes $w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - 0$, for every i, j, l.

We know that every gradient component in this NN is 0.

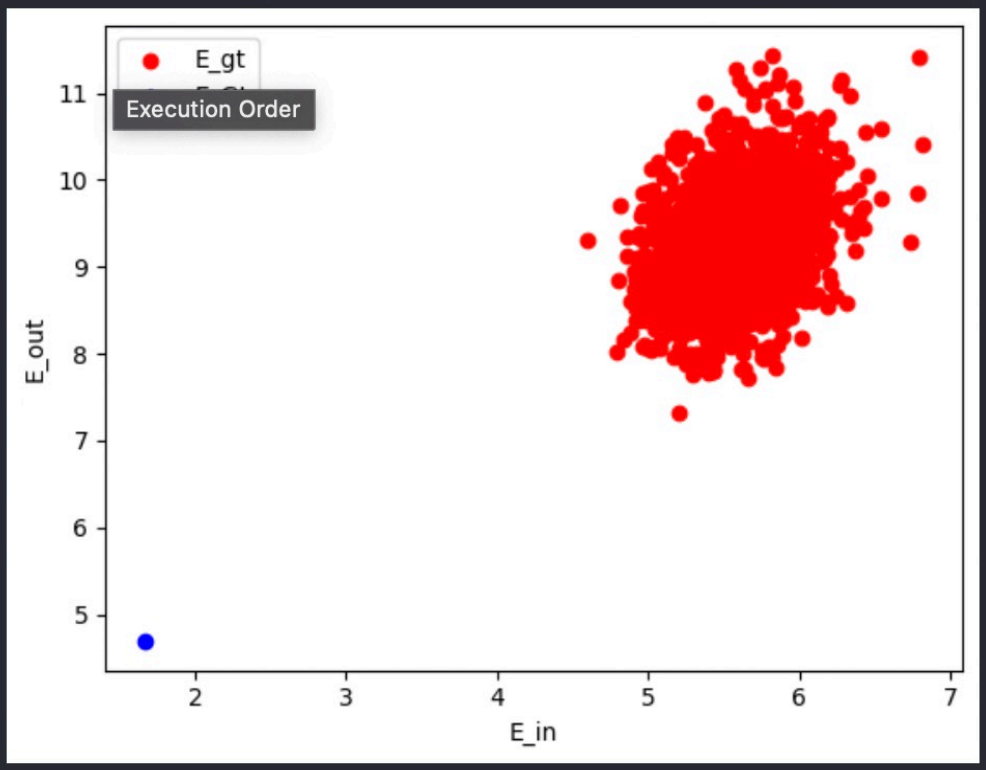
9.

8.79324462640737

10.



11



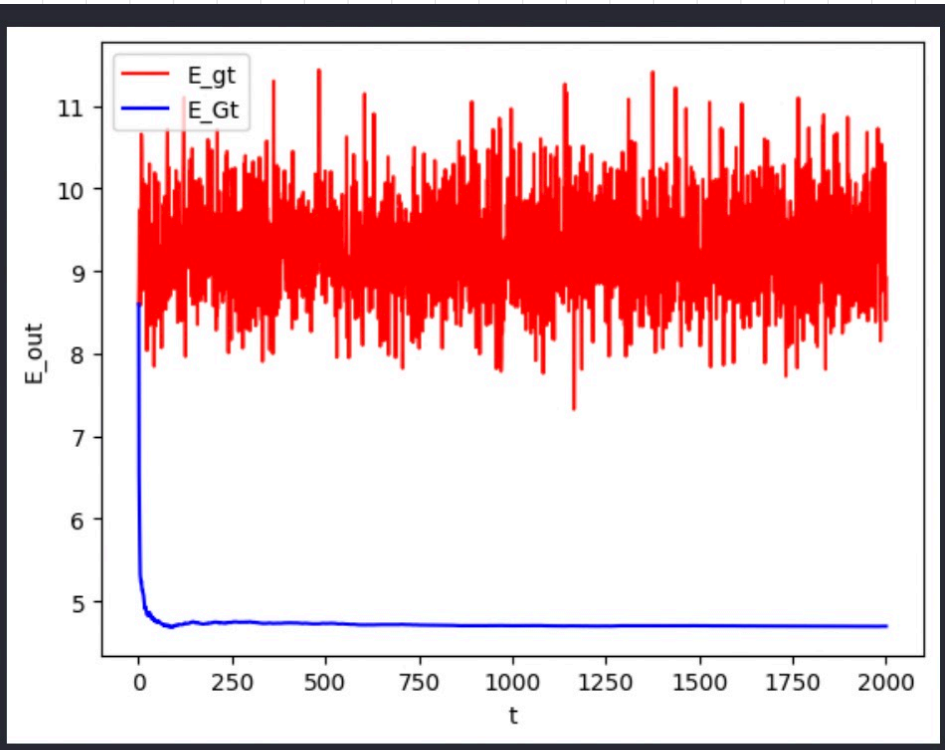
We can see through getting the avg of our random forest, we can significantly let E_out and E_in to drop.

The E_in dropped from 6 to less than 2, and E_out dropped from 10 to less than 5.

The E_out basicly become the half of those RF.

We can conclude that those RF trained by random sampling are pretty different, so that Eout can be reduce that much.

12



As we can see, we don't really need 2000 trees to achieve this amazing performance.

Less than 100 trees would've already do the work.

We can see that the difference for 100 trees are already sufficiently enough.

With more trees, the value only goes towards convergence, which is not a lot of improvement.

If we trained with only 100 trees, we could've save a lot time and achieve almost the same performance.

13.

For $d=2$, $XOR(g_1, g_2) = OR(AND(-g_1, g_2), AND(g_1, -g_2))$

the number of AND() is the number of $1=1$, which is 2.

So, we can see only a 2-2-1 NN works for $d=2$, because we can't remove any AND() in the XOR().

For $d=3$, $\oplus$ represents XOR, + represents OR, $g_1 g_2$ represents and, g' represents not g

Hence $g_1 \oplus g_2 \oplus g_3 = (g_1 \oplus g_2) \oplus g_3$

$= (g_1' g_2 + g_1 g_2') \oplus g_3$

$= (g_1' g_2 + g_1 g_2') g_3' + (g_1' g_2 + g_1 g_2')' g_3$

$= g_1' g_2 g_3' + g_1 g_2' g_3' + \underline{(g_1' g_2)' (g_1 g_2')' g_3}$

$\downarrow$

$\underline{(g_1 + g_2')(g_1' + g_2) g_3}$

$\downarrow$

$g_1 \cancel{g_1' g_3} + g_1 g_2 g_3 + g_1' g_2' g_3 + \cancel{g_2' g_2 g_3}$

$= g_1 g_2' g_3' + g_1' g_2 g_3' + g_1' g_2' g_3 + g_1 g_2 g_3$

Hence, we can see for $d=3$ to work, it'll yield 4 neurons

meaning the second layer must have size of 4, hence a 3-4-1 NN.

Then we can see the formula for $d=d$,

$g_1 \oplus g_2 \oplus \dots \oplus g_d = g_1' g_2 \dots g_d + g_1 g_2' g_3 \dots g_d + \dots + g_1 g_2 \dots g_d' + g_1 \dots g_d$

We will need d+1 neurons in the middle layer, hence a d-(d-1)-1 NN won't do the work.