● More about Support Vector Machines

1. kernel on decision stump

$$\phi_{ds}(\mathbf{x}) = \left(g_{+1,1,\theta_1}(\mathbf{x}), g_{+1,1,\theta_2}(\mathbf{x}), \ldots, g_{+1,1,\theta_k}(\mathbf{x}), \ldots g_{-1,d,\theta_k}(\mathbf{x})\right)$$

$$K_{ds}(\mathbf{x},\mathbf{x}') = \phi_{ds}(\mathbf{x})^T\phi_{ds}(\mathbf{x}')$$

$$= g_{+1,1,\theta_1}(\mathbf{x})^T g_{+1,1,\theta_1}(\mathbf{x}') + \cdots + g_{-1,d,\theta_k}(\mathbf{x})^T g_{-1,d,\theta_k}(\mathbf{x}')$$

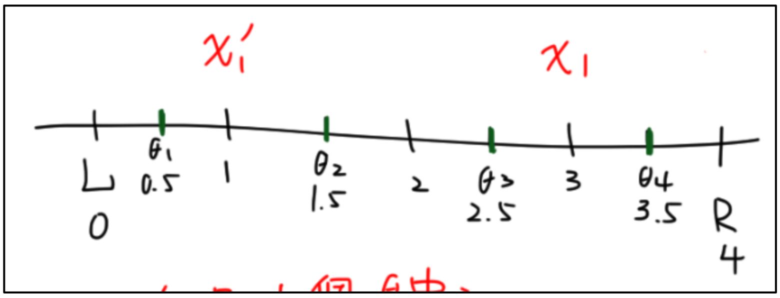
$$= (+1)\text{sign}(x_1 - \theta_1)(+1)\text{sign}(x_1' - \theta_1) + \ldots + (-1)\text{sign}(x_d - \theta_k)(-1)\text{sign}(x_d' - \theta_k)$$

首先 s 的部分因為都是同號，所以可以不用看：

$$= \text{sign}(x_1 - \theta_1)\text{sign}(x_1' - \theta_1) + \ldots + \text{sign}(x_d - \theta_k)\text{sign}(x_d' - \theta_k)$$

所以就是要判別各個$\text{sign}(x_i - \theta_j)\text{sign}(x_i' - \theta_j)$是$+1$還是$-1$。

由於$\mathbf{x}$的值只會是介於 L 跟 R 之間的整數，θ則是介於這些整數之間差 0.5 的值，所以我們可以觀察以下的例子：



$$L = 0, R = 4, x_1' = 1, x_1 = 3$$

假設這是某個在第 1 個維度的時候的情形，可以知道可能的θ有四個，並且根據此題目的特性，θ的個數可以由$R - L$算出來。

在上面的模擬中，可以知道只有當 θ 比 x_1' 小的時候，以及 θ 比 x_1 跟 x_1' 大的時候，$\text{sign}(x_i - \theta_j)\text{sign}(x_i' - \theta_j)$的結果才會是$+1$；$\theta$ 介於 x_1' 跟 x_1 之間時$\text{sign}(x_i - \theta_j)\text{sign}(x_i' - \theta_j)$ 的結果會是-1。一樣根據題目的特性，我們可以知道有 $R - x_1$ 個 θ 比 x_1' 跟 x_1 大，有 $x_1' - L$ 個 θ 比 x_1' 跟 x_1 小，有 $x_1 - x_1'$ 個 θ 介於 x_1' 跟 x_1 之間，因此我們可以列出 $\text{sign}(x_i - \theta_j)\text{sign}(x_i' - \theta_j)$加起來的值為：

$$\underbrace{(x_1' - L) + (R - x_1)}_{+1} - \underbrace{(x_1 - x_1')}_{-1} = R - L - 2(x_1 - x_1')$$

但是上面的例子一旦當x_1在x_1'的左邊，也就是$x_1 < x_1'$的時候，公式就會失效，必須要將 x_1 跟 x_1' 在公式中的位置對調，或者說 $x_1 - x_1'$ 的值取一個負號才會正確，所以公式如果要更為普遍，要改成絕對值：

$$R - L - 2|x_1 - x_1'|$$

此時再考慮 s 有兩種方向，並且總共有 d 個維度，所以要再乘以 2 跟 d：

$$2d\left(R - L - 2\sum_i^d |x_i - x_i'|\right)$$

最後公式可以改成用一範數(one norm)來表示：

$$2d(R - L - 2\|x_1 - x_1'\|_1)$$

2. shift and scale the kernel function

根據題目所述：

$$\tilde{K}(\mathbf{x},\mathbf{x}') = uK(\mathbf{x},\mathbf{x}') + v$$

$$\tilde{C} = \frac{C}{u}$$

列出「縮放的 SVM」的對偶問題：

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m \tilde{K}(\mathbf{x}_n,\mathbf{x}_m{}') - \sum_n \alpha_n$$

$$\text{subject to } \sum_n \alpha_n y_n = 0$$

$$\tilde{C} \geq \alpha_n \geq 0$$

解完 QP 問題後可以得到這個問題的 $\boldsymbol{\alpha}$。這時我令$\boldsymbol{\alpha} = u\boldsymbol{\alpha}$，並且將上面的問題做一些修正：

$$\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m \tilde{K}(\mathbf{x}_n,\mathbf{x}_m{}') - \sum_n \alpha_n$$

$$= \frac{1}{2}\sum_n\sum_m \frac{\alpha_n}{u}\frac{\alpha_m}{u} y_n y_m (uK(\mathbf{x},\mathbf{x}') + v) - \sum_n \frac{\alpha_n}{u}$$

$$= \frac{1}{u}\left[\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m K(\mathbf{x},\mathbf{x}') + \frac{1}{u}\alpha_n\alpha_m y_n y_m v - \sum_n \alpha_n\right]$$

$$= \frac{1}{u}\left[\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m K(\mathbf{x},\mathbf{x}') + \frac{v}{u}\left(\sum_n \alpha_n y_n\right)\left(\sum_m \alpha_m y_m\right) - \sum_n \alpha_n\right]$$

並且將限制改為：

$$\sum_n \alpha_n y_n = \frac{1}{u}\sum_n \alpha_n y_n = 0, \Rightarrow \sum_n \alpha_n y_n = 0$$

$$\tilde{C} \geq \alpha_n \geq 0$$

$$\Rightarrow \frac{C}{u} \geq \alpha_n \geq 0 \Rightarrow C \geq \alpha_n \geq 0$$

$\frac{v}{u}(\sum_n \alpha_n y_n)(\sum_m \alpha_m y_m)$部分可以透過$\sum_n \alpha_n y_n = 0$消除掉，所以最終得到：

$$\min_{\boldsymbol{\alpha}} \frac{1}{u}\left[\frac{1}{2}\sum_n\sum_m \alpha_n\alpha_m y_n y_m K(\mathbf{x},\mathbf{x}') - \sum_n \alpha_n\right]$$

$$\text{subject to } \sum_n \alpha_n y_n = 0$$

$$C \geq \alpha_n \geq 0$$

這個新的 SVM 問題，我將他稱為「還原的 SVM」問題。

而且我令「原本的 SVM」問題為：

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_{n}\sum_{m} \alpha_n \alpha_m y_n y_m K(\mathbf{x},\mathbf{x}') - \sum_{n} \alpha_n$$

$$\text{subject to } \sum_{n} \alpha_n y_n = 0$$

$$C \geq \alpha_n \geq 0$$

我們可以知道「還原的 SVM」的解($\boldsymbol{\alpha}$的部分)，跟「原本的 SVM」的解是一樣的，因為只差了$\frac{1}{u}$這個正的常數倍。

這時我們來從 $\boldsymbol{\alpha}$ 還原出 b 以及 $g(\mathbf{x})$，根據講義的公式，還原 b 只要找其中一個自由支持向量(free SV) s：

$$b = y_s - \sum_{SV} \alpha_n y_n \tilde{K}(\mathbf{x}_n, \mathbf{x}_s)$$

$$g(\mathbf{x}) = sign\left(\sum_{SV} \alpha_n y_n \tilde{K}(\mathbf{x}_n, \mathbf{x}) + b\right)$$

當我們透過 $\boldsymbol{\alpha} = u\boldsymbol{\alpha}$ 這個關係就可以得到「還原的 SVM」的 b 以及 $g(\mathbf{x})$：

$$b = y_s - \sum_{SV} \alpha_n y_n (uK(\mathbf{x}_n, \mathbf{x}_s) + v)$$

$$= y_s - \sum_{SV} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_s) + \frac{1}{u}\sum_{SV} \alpha_n y_n v$$

$$\Rightarrow b = b = y_s - \sum_{SV} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}_s)$$

s 依舊是自由支持向量，因為$\alpha_s < \tilde{C} \Rightarrow \alpha_s < C$

$$g(\mathbf{x}) = sign\left(\sum_{SV} \alpha_n y_n \tilde{K}(\mathbf{x}_n, \mathbf{x}) + b\right)$$

$$= sign\left(\sum_{SV} \alpha_n y_n (uK(\mathbf{x}_n, \mathbf{x}) + v) + b\right)$$

$$\Rightarrow g(\mathbf{x}) = g(\mathbf{x}) = sign\left(\sum_{SV} \alpha_n y_n K(\mathbf{x}_n, \mathbf{x}) + b\right)$$

而「還原的 SVM」的 b 以及 $g(\mathbf{x})$，跟「原本的 SVM」的 b 以及 $g(\mathbf{x})$是一樣的，所以我們就可以知道「縮放的 SVM」的 b 以及 $g(\mathbf{x})$，跟「原本的 SVM」的 b 以及 $g(\mathbf{x})$是一樣的。

- Blending and Bagging

3. Upper bound of Error

對於每個 $g_t(\mathbf{x})$ 的錯誤率 e_t，假設總共有 N 筆資料，那麼$g_t(\mathbf{x})$犯錯的筆數就是$N \cdot e_t$。而在 17 個 $g_t(\mathbf{x})$ 的投票過程當中，一筆資料要被分類錯誤，必須要 17 個 $g_t(\mathbf{x})$ 中一半的人都犯錯，該筆資料才會是投票後犯錯的，也就是要$(17+1)/2 = 9$個$g_t(\mathbf{x})$ 都在該筆資料犯錯。

而總共犯錯的筆數是：

$$\sum_{t=1}^{17} N \cdot e_t$$

可以知道我們最多可以弄出：

$$\frac{\sum_{t=1}^{17} N \cdot e_t}{9}$$

這麼多筆資料是投票之後犯錯的，所以可以知道$E_{out}(G)$最大可以是：

$$E_{out}(G) = \frac{\frac{\sum_{t=1}^{17} N \cdot e_t}{9}}{N} = \frac{1}{9}\sum_{t=1}^{17} e_t$$

因此：

$$\frac{E_{out}(G)}{E} = \frac{\frac{1}{9}\sum_{t=1}^{17} e_t}{\sum_{t=1}^{17} e_t} = \frac{1}{9}$$

4. OOB Probability

$$\left(1-\frac{1}{N}\right)^{\frac{3}{4}N} = \left(\left(1-\frac{1}{N}\right)^{N}\right)^{\frac{3}{4}}$$

這裡直接用講義(隨機森林第 8 頁)推導的結果：

$$\left(1-\frac{1}{N}\right)^{N} = \frac{1}{e}$$

所以可知：

$$\left(\left(1-\frac{1}{N}\right)^{N}\right)^{\frac{3}{4}} = \left(\frac{1}{e}\right)^{\frac{3}{4}}$$

● Adaptive Boosting and Gradient Boosting

5. AdaBoost can deal with “imbalanced” data immediately

首先我們可以知道：

$$u_n^{(1)} = \frac{1}{N}$$

並且在第一回合，由於 AdaBoost algorithm 吐了$g_1(\mathbf{x}) = +1$回來，所以可以知道有 2%的資料是犯錯的，它們的$y_n = -1$；98%的資料是預測對的$y_n = +1$，所以首先可以算出錯誤率 ϵ ：

$$\epsilon = \frac{\sum_n u_n^{(1)} [\![y_n \neq g_1(\mathbf{x_n})]\!]}{\sum_n u_n^{(1)}} = \frac{0.02N \cdot \frac{1}{N}}{N \cdot \frac{1}{N}} = 0.02$$

接著可以算出縮放係數 α ：

$$\alpha = \sqrt{\frac{1-\epsilon}{\epsilon}} = \sqrt{\frac{1-0.02}{0.02}} = \sqrt{\frac{98}{2}} = \sqrt{49}$$

所以我們可以將$u_n^{(1)}$更新成$u_n^{(2)}$：

$$u_n^{(2)} = \frac{u_n^{(1)}}{\alpha} \text{ for } y_n = +1$$
$$u_n^{(2)} = u_n^{(1)} \cdot \alpha \text{ for } y_n = -1$$

所以我們可以算出：

$$\sum_{n:y_n=+1} u_n^{(2)} = \sum_{n:y_n=+1} \frac{u_n^{(1)}}{\alpha} = \frac{0.98N \cdot \frac{1}{N}}{\alpha} = \frac{0.98}{\sqrt{49}}$$

$$\sum_{n:y_n=-1} u_n^{(2)} = \sum_{n:y_n=-1} u_n^{(1)}\alpha = \alpha \cdot 0.02N \cdot \frac{1}{N} = \sqrt{49} \cdot 0.02$$

$$\frac{\sum_{n:y_n=+1} u_n^{(2)}}{\sum_{n:y_n=-1} u_n^{(2)}} = \frac{\frac{0.98}{\sqrt{49}}}{\sqrt{49} \cdot 0.02} = \frac{0.98}{49 \cdot 0.02} = 1$$

跟原本的值相比：

$$\frac{\sum_{n:y_n=+1} u_n^{(1)}}{\sum_{n:y_n=-1} u_n^{(1)}} = 49$$

減少了許多。

6. Some result AdaBoost

首先我們知道：

$$\epsilon_t = \frac{\sum_n [\![y_n \neq g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t)}}{\sum_n u_n^{(t)}}$$

所以可以巧妙的發現：

$$\sqrt{\epsilon_t(1-\epsilon_t)} = \epsilon_t \times \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \frac{\sum_n [\![y_n \neq g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t)} \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{\sum_n u_n^{(t)}}$$

$$= \frac{\sum_n [\![y_n \neq g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t+1)}}{\sum_n u_n^{(t)}}$$

$$\sqrt{\epsilon_t(1-\epsilon_t)} = (1-\epsilon_t) \div \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \left(1 - \frac{\sum_n [\![y_n \neq g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t)}}{\sum_n u_n^{(t)}} - \right) \div \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$$

$$= \frac{\sum_n [\![y_n = g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t)} \div \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}}{\sum_n u_n^{(t)}} = \frac{\sum_n [\![y_n = g_t(\mathbf{x}_{\boldsymbol{n}})]\!] u_n^{(t+1)}}{\sum_n u_n^{(t)}}$$

$$2\sqrt{\epsilon_t(1-\epsilon_t)} = \epsilon_t \times \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t) \div \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$$

$$= \frac{\sum_n \left[\!\left[y_n \neq g_t(\mathbf{x}_{\boldsymbol{n}})\right]\!\right] u_n^{(t+1)}}{\sum_n u_n^{(t)}} + \frac{\sum_n \left[\!\left[y_n = g_t(\mathbf{x}_{\boldsymbol{n}})\right]\!\right] u_n^{(t+1)}}{\sum_n u_n^{(t)}}$$

$$\Rightarrow 2\sqrt{\epsilon_t(1-\epsilon_t)} = \frac{\sum_n u_n^{(t+1)}}{\sum_n u_n^{(t)}}$$

所以只要一直往回推到$t = 1$，就可以知道：

$$\frac{\sum_n u_n^{(T+1)}}{\sum_n u_n^{(T)}} \times \frac{\sum_n u_n^{(T)}}{\sum_n u_n^{(T-1)}} \times \ldots \times \frac{\sum_n u_n^{(2)}}{\sum_n u_n^{(1)}} = \frac{U_{T+1}}{U_1}$$

$$= 2\sqrt{\epsilon_T(1-\epsilon_T)} \times 2\sqrt{\epsilon_{T-1}(1-\epsilon_{T-1})} \times \ldots \times 2\sqrt{\epsilon_1(1-\epsilon_1)}$$

$$\Rightarrow \frac{U_{T+1}}{U_1} = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

7. Some result of gradient boosted decision tree

首先列出更新公式：

$$s_n^{t+1} = s_n^t + \alpha_t g_t(\mathbf{x_n})$$

所以可以知道：

$$\sum_{n=1}^{N} s_n^{t+1} g_t(\mathbf{x_n}) = \sum_{n=1}^{N} \left(s_n^t + \alpha_t g_t(\mathbf{x_n})\right) g_t(\mathbf{x_n})$$

$$= \sum_{n=1}^{N} s_n^t g_t(\mathbf{x_n}) + \alpha_t g_t^2(\mathbf{x_n}) = \sum_{n=1}^{N} s_n^t g_t(\mathbf{x_n}) + \alpha_t \sum_{n=1}^{N} g_t^2(\mathbf{x_n})$$

s_n^{t+1}其實就是題目說的更新後的s_n^t。這時候回顧α_t計算的方式：

$$\alpha_t = \frac{\sum_{n=1}^{N} g_t(\mathbf{x_n})(y_n - s_n^t)}{\sum_{n=1}^{N} g_t^2(\mathbf{x_n})}$$

代進去：

$$\sum_{n=1}^{N} s_n^t g_t(\mathbf{x_n}) + \frac{\sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n^t)}{\sum_{n=1}^{N} g_t^2(\mathbf{x_n})} \times \sum_{n=1}^{N} g_t^2(\mathbf{x_n})$$

$$= \sum_{n=1}^{N} s_n^t g_t(\mathbf{x_n}) + \sum_{n=1}^{N} g_t(\mathbf{x}_n)(y_n - s_n^t)$$

$$= \sum_{n=1}^{N} g_t(\mathbf{x}_n) y_n$$

Red correction：跟改後的題目為$\sum_{n=1}^{N}(y_n - s_n^{t+1}) g_t(\mathbf{x_n})$，而這點可以從上面的結論$\sum_{n=1}^{N} s_n^{t+1} g_t(\mathbf{x_n}) = \sum_{n=1}^{N} g_t(\mathbf{x}_n) y_n$清楚的推論出來

● Neural Networks

8. gradient components

下面的表記法採用老師講義的格式：

w_{ij}^d，配上第$d-1$層的x_i^{d-1}送往第 d 層的s_j^d的權重

x_i^d，第d層的第 i 個 x

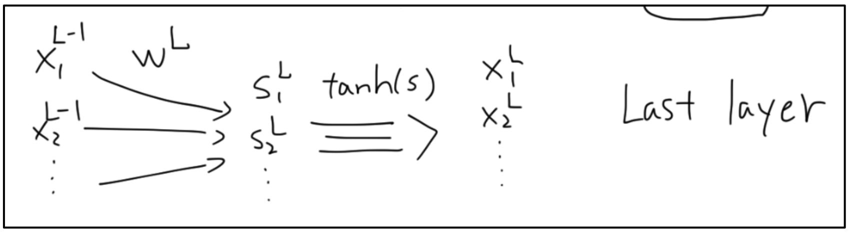
s_i^d，第d層的第 i 個 s

先列出通往最後一層 L 的權重 w_{ij}^L 的偏導數

$$\frac{\partial e_n}{\partial w_{ij}^L} = \frac{\partial e_n}{\partial x_j^L} \times \tanh'(s_j^L) \times \frac{\partial s_j^L}{\partial w_{ij}^L}$$

$$= \frac{\partial e_n}{\partial x_j^L} \times \tanh'(s_j^L) \times x_i^{L-1}$$

圖示的部分如下：



因為題目有說 output neuron 也有經過 tanh，所以才會有 x_j^L，並且這裡考慮的是更普遍的情況，輸出不只一個而是很多個。

$\frac{\partial e_n}{\partial x_j^L}$ 會根據 error function 的不同而不同，例如講義是用均方誤差。

接著列出中間層的 d 的權重 w_{ij}^d 的偏導數：

$$\frac{\partial e_n}{\partial w_{ij}^d} = \frac{\partial e_n}{\partial s_j^d} \times \frac{\partial s_j^d}{\partial w_{ij}^d} = \frac{\partial e_n}{\partial s_j^d} \times x_i^{d-1}$$

$$\frac{\partial e_n}{\partial s_j^d} = \underbrace{\tanh'(s_j^d)}_{x_j^d\text{是}s_j^d\text{函數}} \times \underbrace{\sum_{k=1}^{M} \frac{\partial s_k^{d+1}}{\partial x_j^d} \times \frac{\partial e_n}{\partial s_k^{d+1}}}_{x_j^d\text{分別是}M\text{個}s_k^{d+1}\text{的函數}}$$

$$= \tanh'(s_j^d) \times \sum_{k=1}^{M} w_{jk}^{d+1} \times \frac{\partial e_n}{\partial s_k^{d+1}}$$

由於初始權重都是 0，所以可以知道任何算出來的 s_j^d 都會是 0。
因此根據公式可以推得：

$$\frac{\partial e_n}{\partial s_j^d} = \tanh'(s_j^d) \times \sum_{k=1}^{M} w_{jk}^{d+1} \times \frac{\partial e_n}{\partial s_k^{d+1}} = 0$$

$$\frac{\partial e_n}{\partial w_{ij}^d} = \frac{\partial e_n}{\partial s_j^d} \times x_i^{d-1} = 0$$

所以中間層的偏導數都會是 0。
而最後一層的情形為：

$$\frac{\partial e_n}{\partial w_{ij}^L} = \frac{\partial e_n}{\partial x_j^L} \times \tanh'(s_j^L) \times x_i^{L-1}$$

這時候情況會有些不一樣：對於x_0^{L-1}來說，他是我們補上去的常數項，如果他是 0 的話，那麼 $\frac{\partial e_n}{\partial w_{0j}^L}$ 就會是 0。

對於其他的 x_i^{L-1} 來說，他們是從$\tanh(s_i^{L-1})$算出來的，而$s_i^{L-1} = 0$

所以可以知道 $\frac{\partial e_n}{\partial w_{ij}^L}$ 必定為 0。

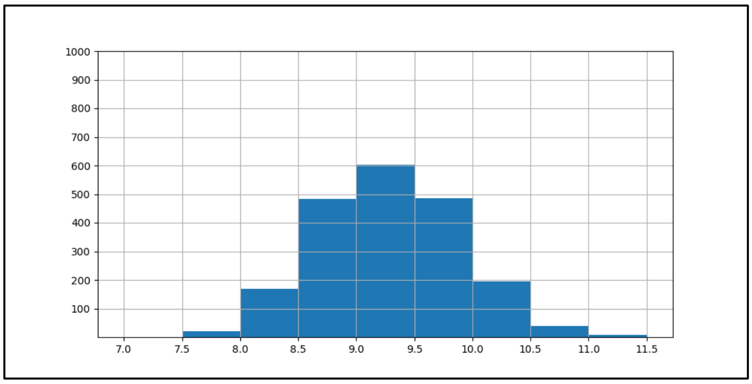
因此可以知道只有 $\frac{\partial e_n}{\partial w_{0j}^L}$ 是有可能不會是 0 的，其他都必定為 0。

● Experiments with Decision Trees and Random Forests

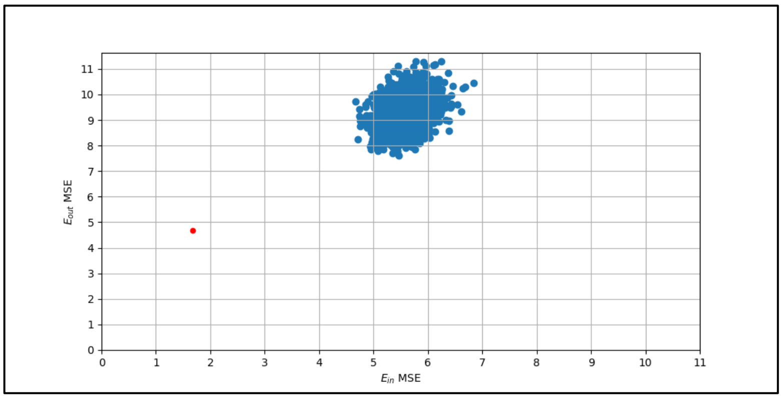
9. $E_{out}(g)$ of unpruned decision tree

$$E_{out}(g) = 8.71$$

10. $E_{out}(g)$ of 2000 unpruned decision trees

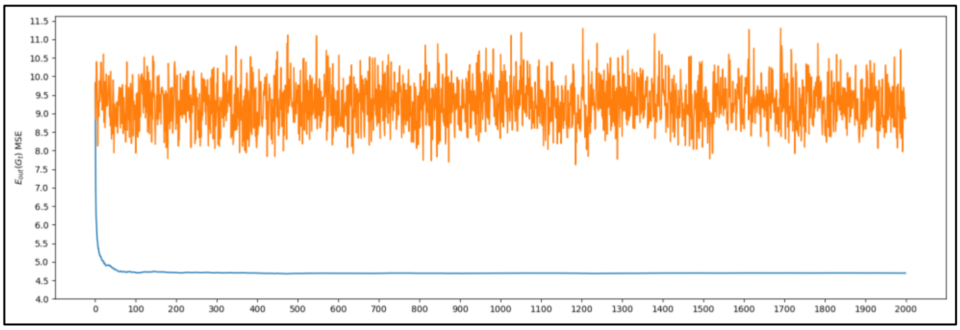


11. E_{in} and E_{out}scatter plot



紅色點是$(E_{in}(G), E_{out}(G))$的所在位置，可以看到他離資料分布的位置有一段距離，並且是 Error 小的區域，這很好的顯示了 voting 所達到的降低錯誤效果，並且也不易受離群值的影響。

12. $E_{out}(G_t)$function



隨著參與投票的人越來越多，除了結果越趨於穩定，不易受離群值影響，錯誤率也跟著降低。不過可以發現錯誤率在大約 100 棵樹的時候就不再下降了，而那些有誤差的地方我認為應該是「尚未學到」的部分，因為都經過這麼多人投票決定了卻依舊有誤差。

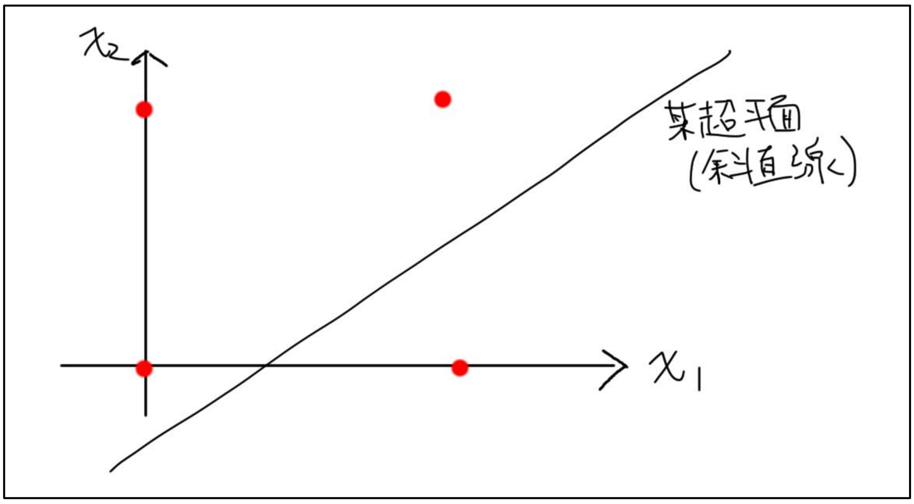
● Bonus

13. Crazy XOR

從這份作業發佈的那一天我就把前 12 題弄完了，全力對付這題，但是就算如此，經過這麼多天，甚至我使用了 6 張金牌延長 3 天的時限，同時伴隨著其他科期末的壓力，我依舊想不到這題要如何證明對所有的 d 是不可能的。

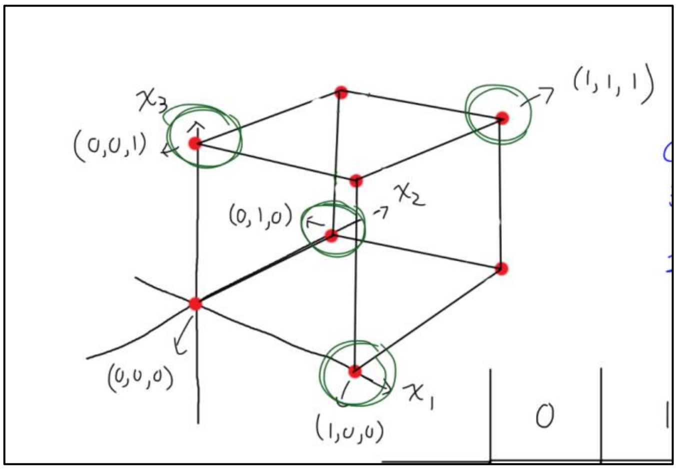
這裡我的設定是 x 只會是 0 或 1。

目前我只能證明出$d = 2$的時候是不可能的，因為$d = 2$的時候可以看做是二維平面的 4 個點：



然而無論選了哪條斜直線，(1,0)或(0,1)這至少有一點就是無法被正確判定是+1。因此$d = 2$的情形可以用清楚的方式說明。

$d = 3$的時候其實也是可以用類似的手法進行說明：



但是也就僅止於$d = 3$了，再上去就超越人類極限了，必須要用更普遍的方法。

下面列舉我曾經的思路。

思路一：歸納法

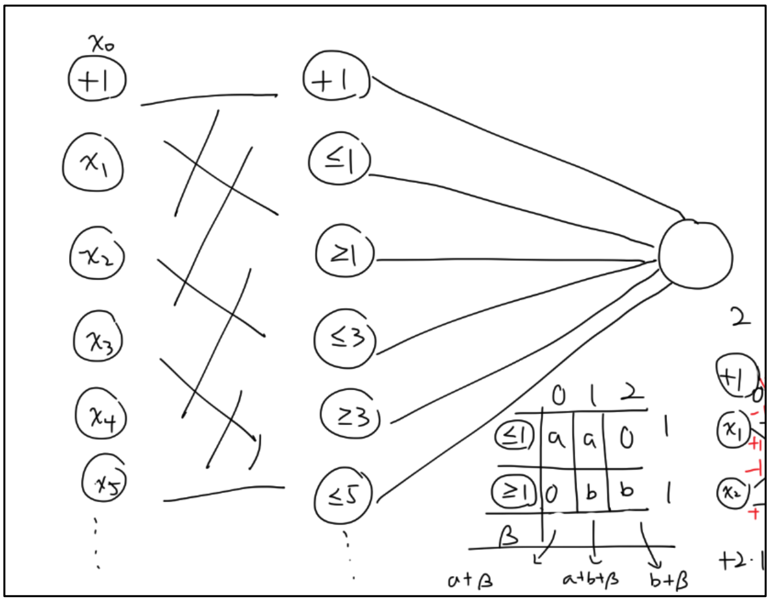
由於有上面$d=2$的明確例子可以使用，所以我曾經想使用歸納法做證明，我的 induction hypothesis 是「$d-(d-1)-1$網路不能正確的達成 XOR 的要求，也就是某些情形的輸出是錯誤的」。

但是會遇到許多問題：

就算我有了$d=k-1$的情形，我很難推導出$d=k$的時候也可以成立，很難從小的網路變成大的網路，因為會有新的輸入權重接上舊的神經元，舊的神經元會接上新的中間層神經元；亦或是這樣的方法，因為並沒有確定權重是多少，可能就會違反歸納法的要求了。

思路二：證明$d-d-1$ 網路「解結構」的唯一性

由於我有已知確定形式的$d-d-1$ 網路的「解結構」：



也就是中間層的各個神經元，或者說超平面，用途是判別當前 1 的個數是「小於等於 1」、「大於等於 1」、「小於等於 3」、「大於等於 3」... 以此類推，如果 d 是偶數的話最後一個就是「大於等於$d-1$」，但如果 d 是奇數的話，最後一個要是「大於等於 d 」。

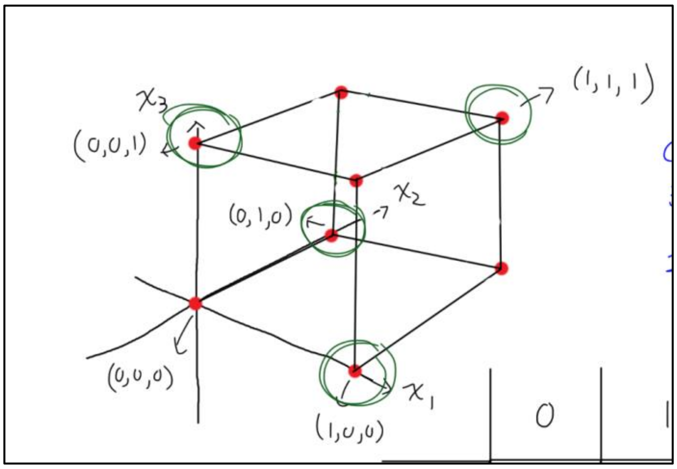
這樣的形式我有準確的配置權重方式，讓輸出滿足 XOR 的要求：

	0	1	2	3	4	5	6	7	8
≤1	a	a	0	0	0	0	0	0	0
≥1	0	b	b	b	b	b	b	b	b
≤3	c	c	c	c	0	0	0	0	0
≥3	0	0	0	d	d	d	d	d	d
≤5	e	e	e	e	e	e	0	0	0
≥5	0	0	0	0	0	f	f	f	f
≤7	g	g	g	g	g	g	g	g	0
≥7	0	0	0	0	0	0	0	h	h

中間的字母 a,b,...代表左方 column 的中間層神經元，如果輸出是 1 的話再乘上其前往輸出層的權重的結果，這裡我令由上到下的神經元權重分別是 a,b,...以此類推。最上面的 row 的 0,1,2,...代表當前的輸入有

幾個 1。可以看到如果輸入是奇數個 1，那麼總共一定有 5 個中間層神經元輸出是 1，如果是偶數個 1，則會是 4 個；這個現象可以推廣出一個結論：偶數個 1 會有$\left\lfloor\frac{d}{2}\right\rfloor$個神經元輸出是 1，奇數個 1 則是有$\left\lfloor\frac{d}{2}\right\rfloor+1$個神經元輸出是 1。
從這個結論可以得知只要讓每個字母的值都是 1，最後 bias 項的權重剛好是$-\left\lfloor\frac{d}{2}\right\rfloor$，這樣子就是個合法的 XOR 神經網路。

上面是我已知的合法結構，並且我可以根據對稱性，也就是經過旋轉或鏡射得到等價的結構，例如上面 3 維的結構：



我可以將整個圖繞x_3軸順時針旋轉 90 度，並將這些旋轉後的點當作暫時該位置的點，然後再用這個點，搭配上面的網路架構，一樣會得到合法解，再轉回去後就是一個新的合法解了。

但是我不知道怎麼證明這是個唯一形式的「解架構」。
如果我確定了這是唯一形式的解架構，那麼我就可以說因為隨便砍掉其中一個中間神經元，就無法藉由調整 bias 的手段使得輸出滿足 XOR，因為會導致有些偶數 1 的輸入跟奇數 1 的輸入，經過中間神經元後的輸出加總是一樣的。

思路三：另一種角度
如果有個中間層神經元，所有的輸入，他的輸出都一樣，那麼它的作用其實就跟 bias 項是一樣的，並且因此可以刪除。所以我有嘗試去證明對於$d-d-1$網路來說，所有中間層的神經元，對所有的輸入其輸出一定不會都一樣。但是要證明這件事我一樣想不到 orz。

以上是心路歷程，如果有個苦勞分我會感到很欣慰：_)。這題的解法還懇請助教到時標記在 gradescope 一下，真的很好奇是怎麼證明的。