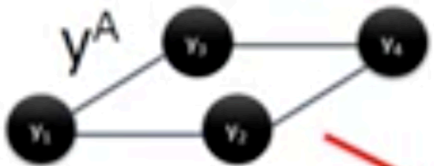
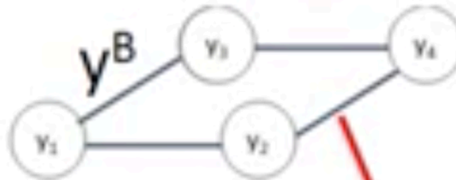
總共有16種可能的組合，這邊列舉三種實際上算出來的機率分佈如下

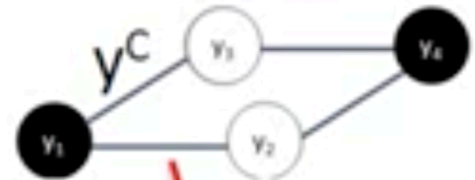
$P(y^A|x) \approx 0.33$



$P(y^B|x) \approx 0.22$



$P(y^C|x) \approx 0.004$



No. of samples	y^A	y^B	y^C
10	2	$\boxed{3}$	0
100	23	$\boxed{37}$	0
1000	$\boxed{315}$	230	8
10000	$\boxed{3307}$	2225	40
100000	$\boxed{32637}$	22129	422

不滿十次是因為剩下的事sample到別的組合

$P(y^A|x) \approx 0.33$ $P(y^B|x) \approx 0.22$ $P(y^C|x) \approx 0.004$

From sampling: y^A would be the results of inference.

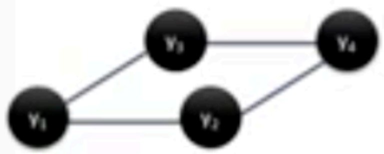
在iteration越大的時候越能表示真實的機率分佈

不同initial對結果沒有差

How about starting from different initialization?

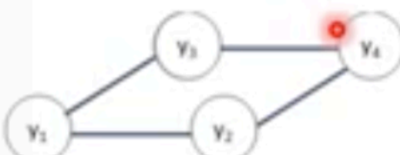
Not really change the final results.

Starting from ...
case A



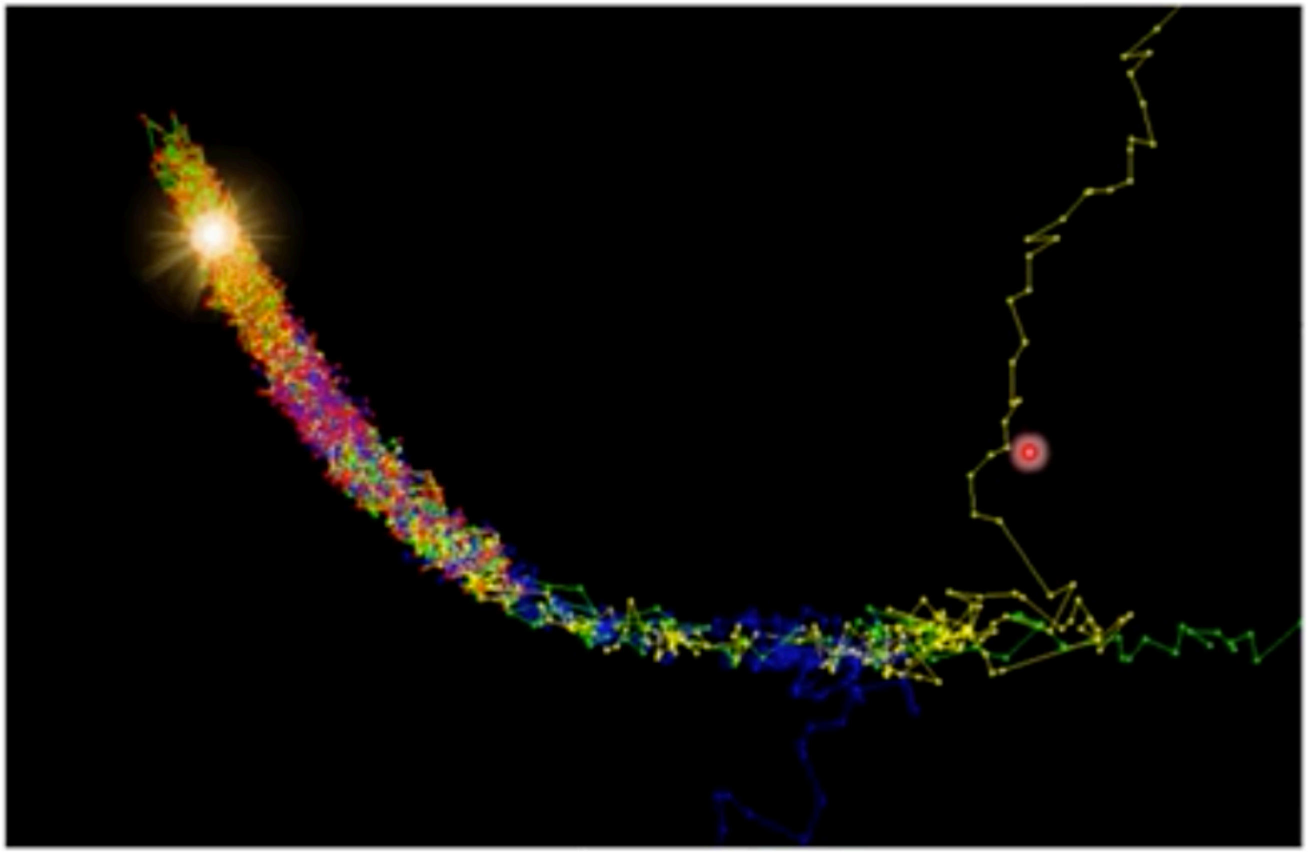
No. of samples	A	B	C
10	3	1	0
100	40	11	1
1000	331	237	2
10000	3251	2176	31
100000	32911	21845	385

Starting from ...
case B



No. of samples	A	B	C
10	0	3	0
100	28	31	0
1000	318	226	2
10000	3277	2169	46
100000	32319	21751	[illegible]

All rivers run into the sea.



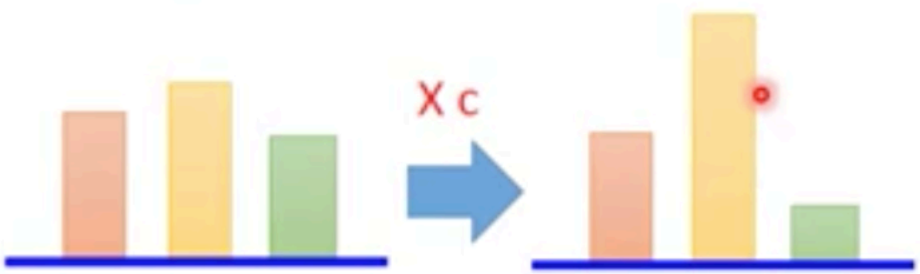
Practical Suggestion

- "burn-in"　假設擔心初始值有影響的話就丟掉前幾個sample
  - "burn-in" period: The first few of samples would be influenced by the initialization
  - Discard the samples in the "burn-in" period
- Modify the sampling distribution

如果有其他方法就不要用Gibbs，因為只優於窮舉

c > 1

$$P(y_i|y_1,y_2,\cdots,y_{i-1},y_{i+1},\cdots,y_N,\boldsymbol{x}) = \frac{e^{F(x,y_{-i},y_i)\times c}}{\sum_{y_i'} e^{F(x,y_{-i},y_i')\times c}}$$

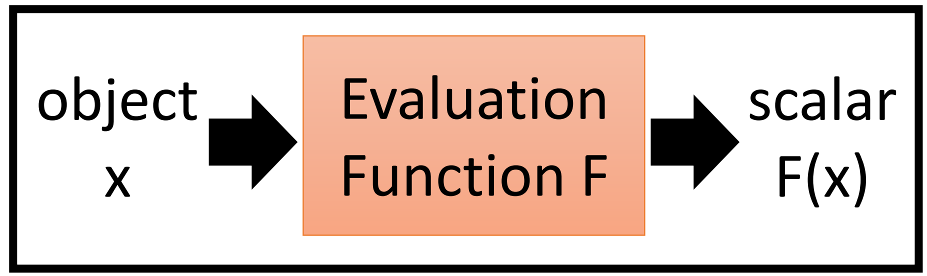


Increase c after each interaction

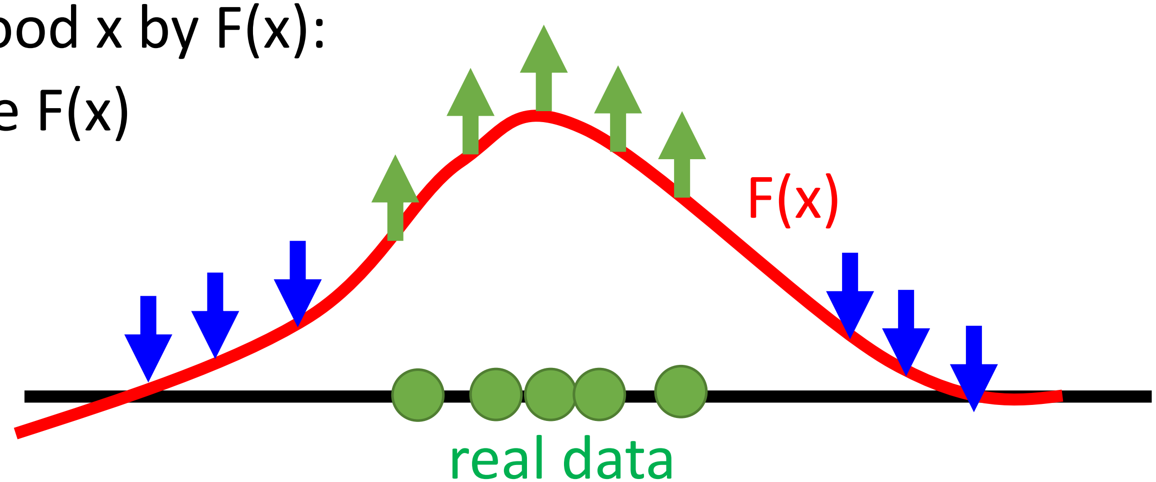
另一個問題是如果今天機率分佈非常平均，space又很大，很重負sample到相同的點
因此我們乘上一個scale放大他們之間的差異（修改distribution）

Evaluation Function

- We want to find an evaluation function F(x)
  - Input: object x, output: scalar F(x) (how “good” the object is)
  - E.g. x are images
    - Real x has high F(x)
  - F(x) can be a network



- We can generate good x by F(x):
  - Find x with large F(x)
- How to find F(x)?



In practice, you cannot decrease all the x other than real data.

Evaluation Function
- Structured Perceptron

- **Input**: training data set $\{(x^1,\hat{y}^1),(x^2,\hat{y}^2),\ldots,(x^r,\hat{y}^r),\ldots\}$
- **Output**: weight vector w
- **Algorithm**: Initialize w = 0

$$F(x,y)=w\cdot\phi(x,y)$$

  - do
    - For each pair of training example $(x^r,\hat{y}^r)$
      - Find the label $\tilde{y}^r$ maximizing F(x^r,y)

Can be an issue

$$\tilde{y}^r=\arg\max_{y\in Y}F(x^r,y)$$

      - If $\tilde{y}^r\neq\hat{y}^r$, update w

Increase $F(x^r,\hat{y}^r)$, decrease $F(x^r,\tilde{y}^r)$

$$w\rightarrow w+\phi(x^r,\hat{y}^r)-\phi(x^r,\tilde{y}^r)$$

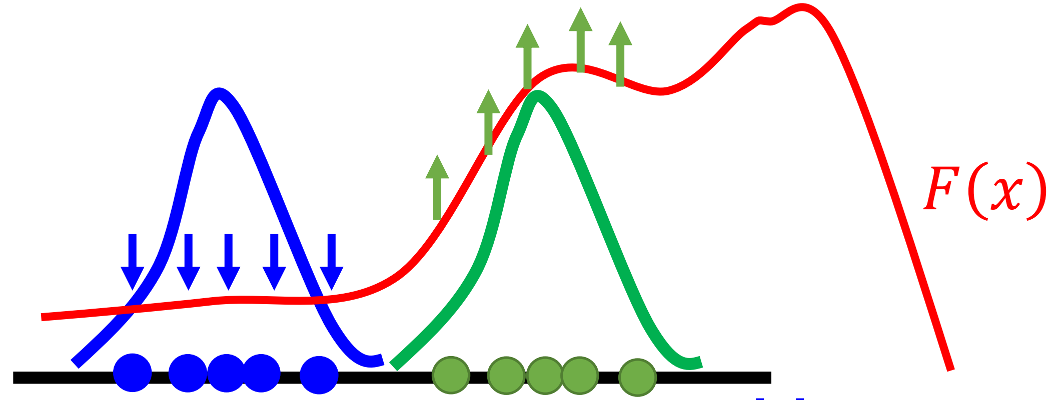
  - until w is not updated

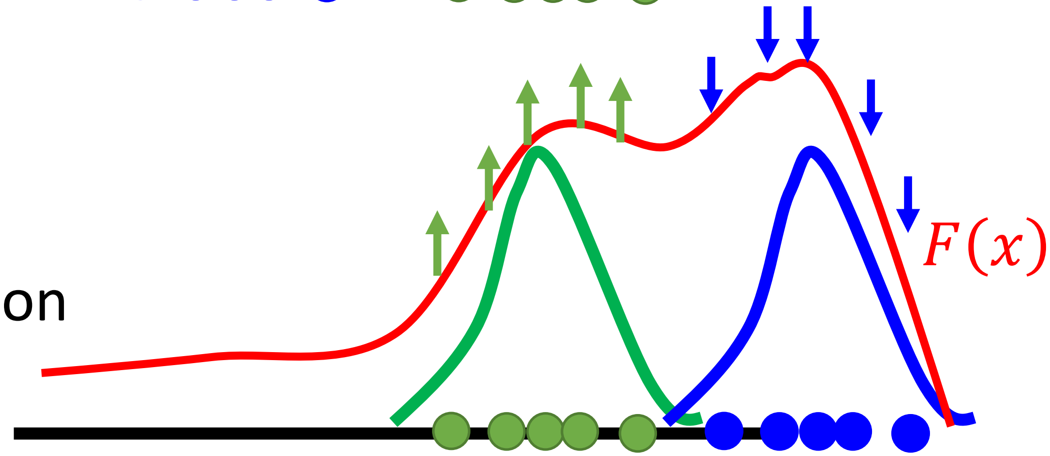
We are done!

How about GAN?

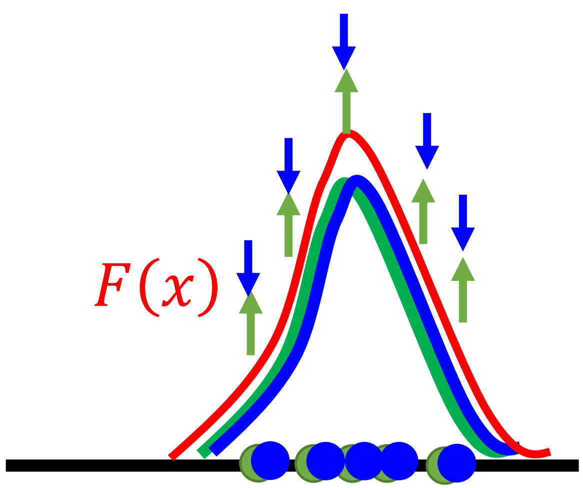
- Generator is an intelligent way to find the negative examples.

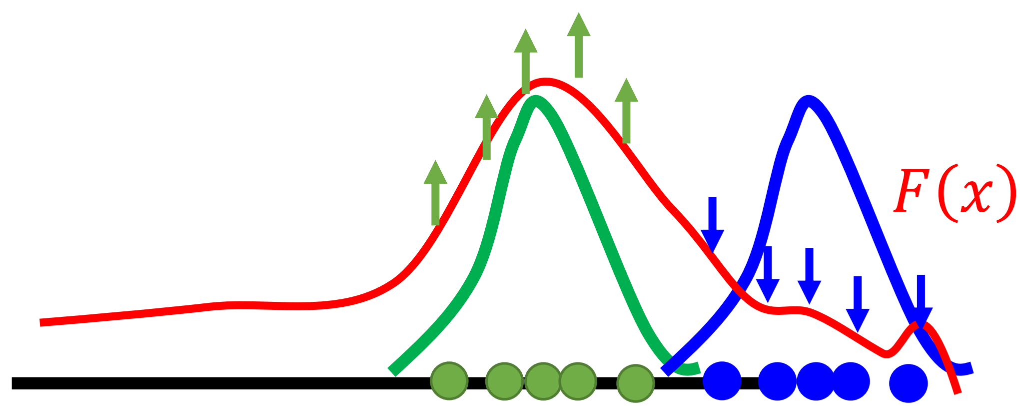
"Experience replay", parameters from last iteration



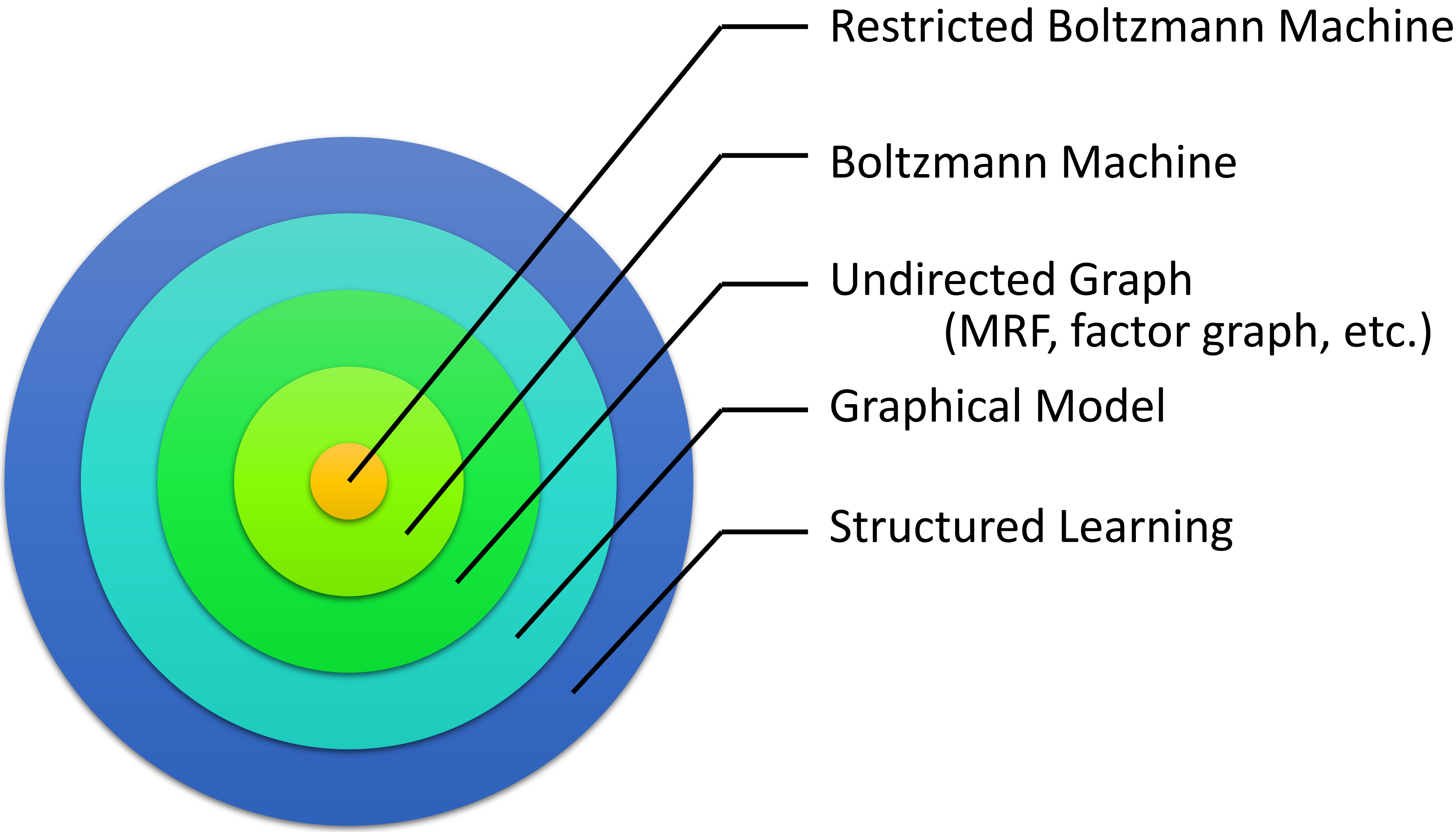


In the end ……





Where are we?


Restricted Boltzmann Machine
Boltzmann Machine
Undirected Graph
(MRF, factor graph, etc.)
Graphical Model
Structured Learning