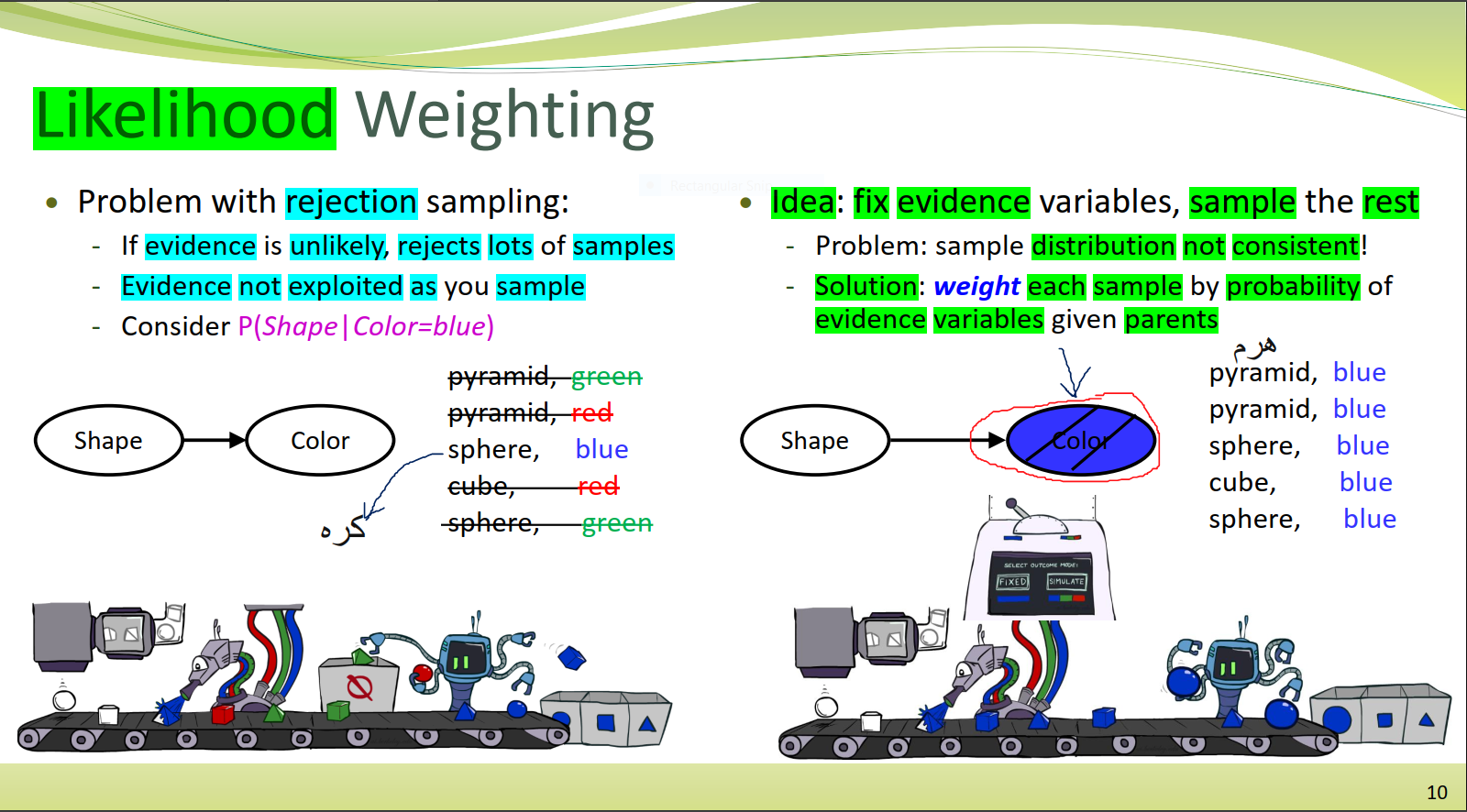
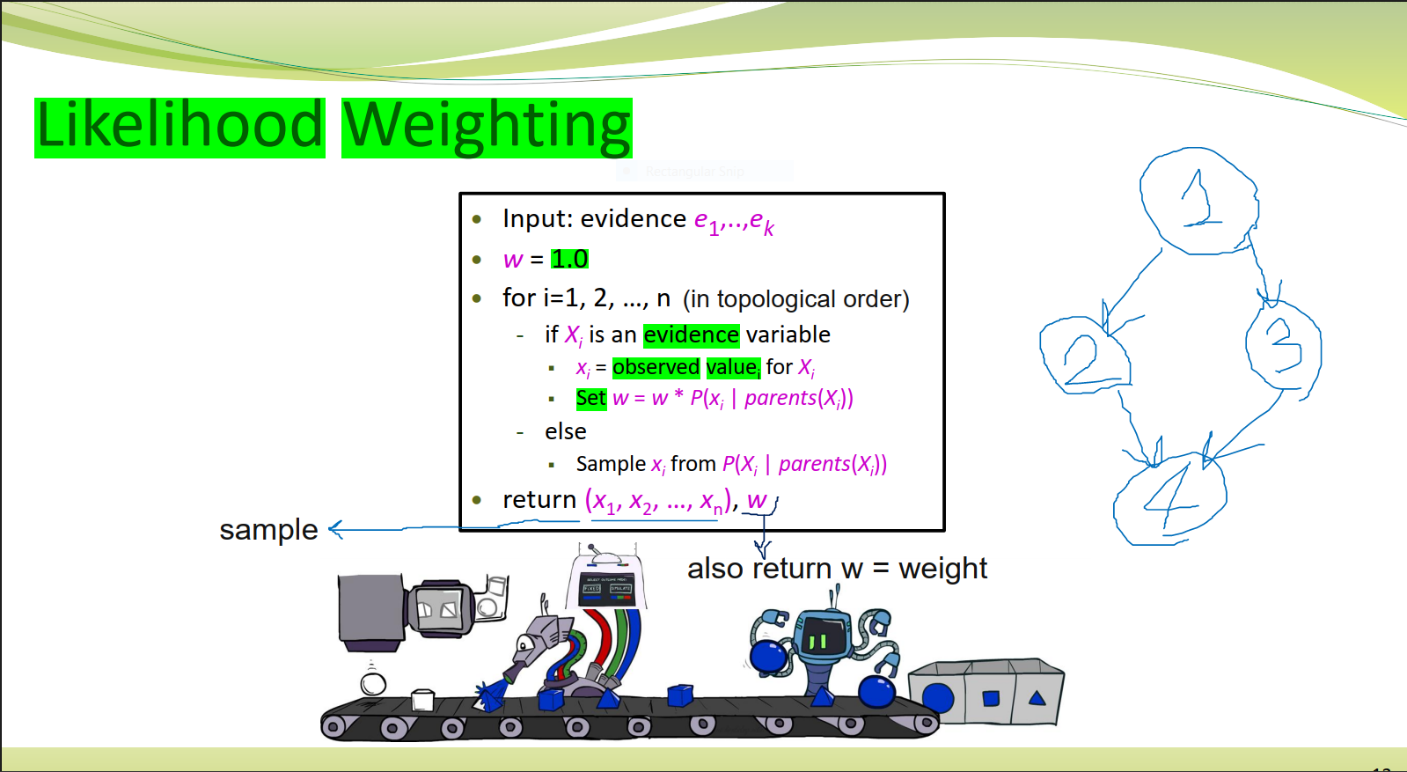
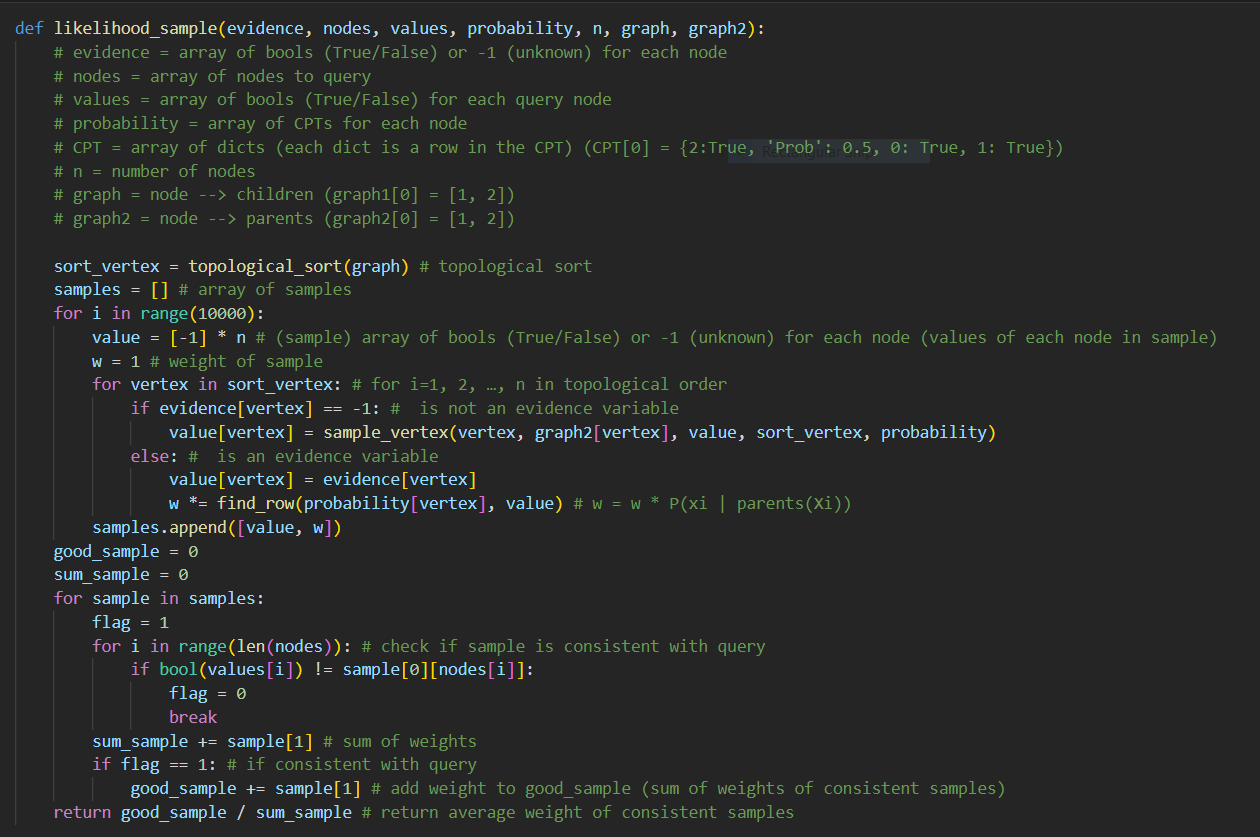
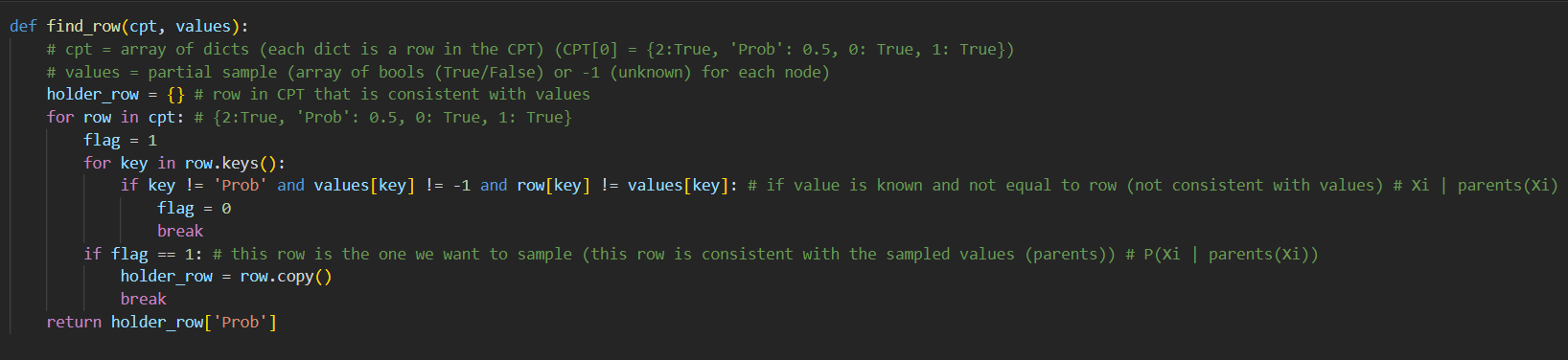
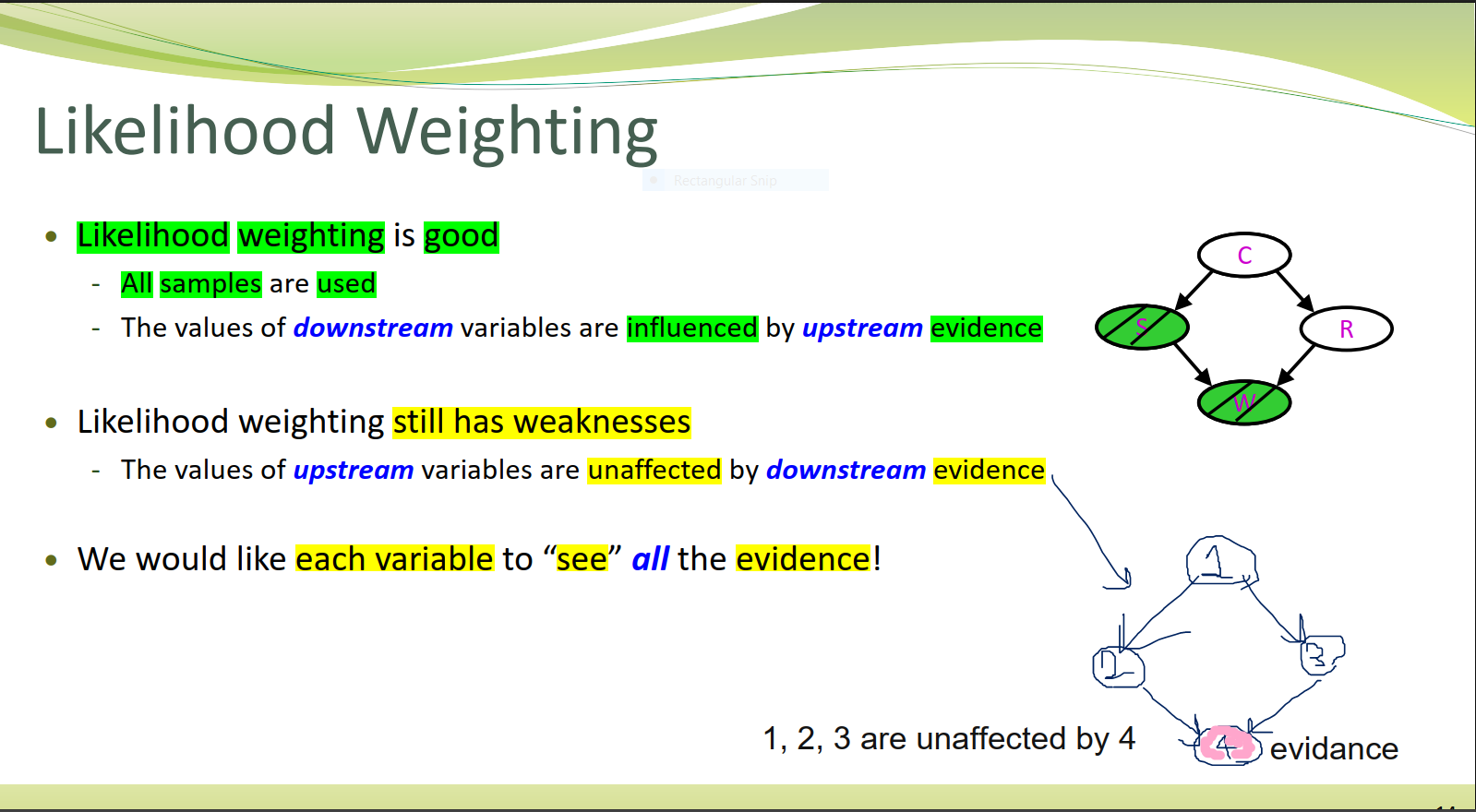
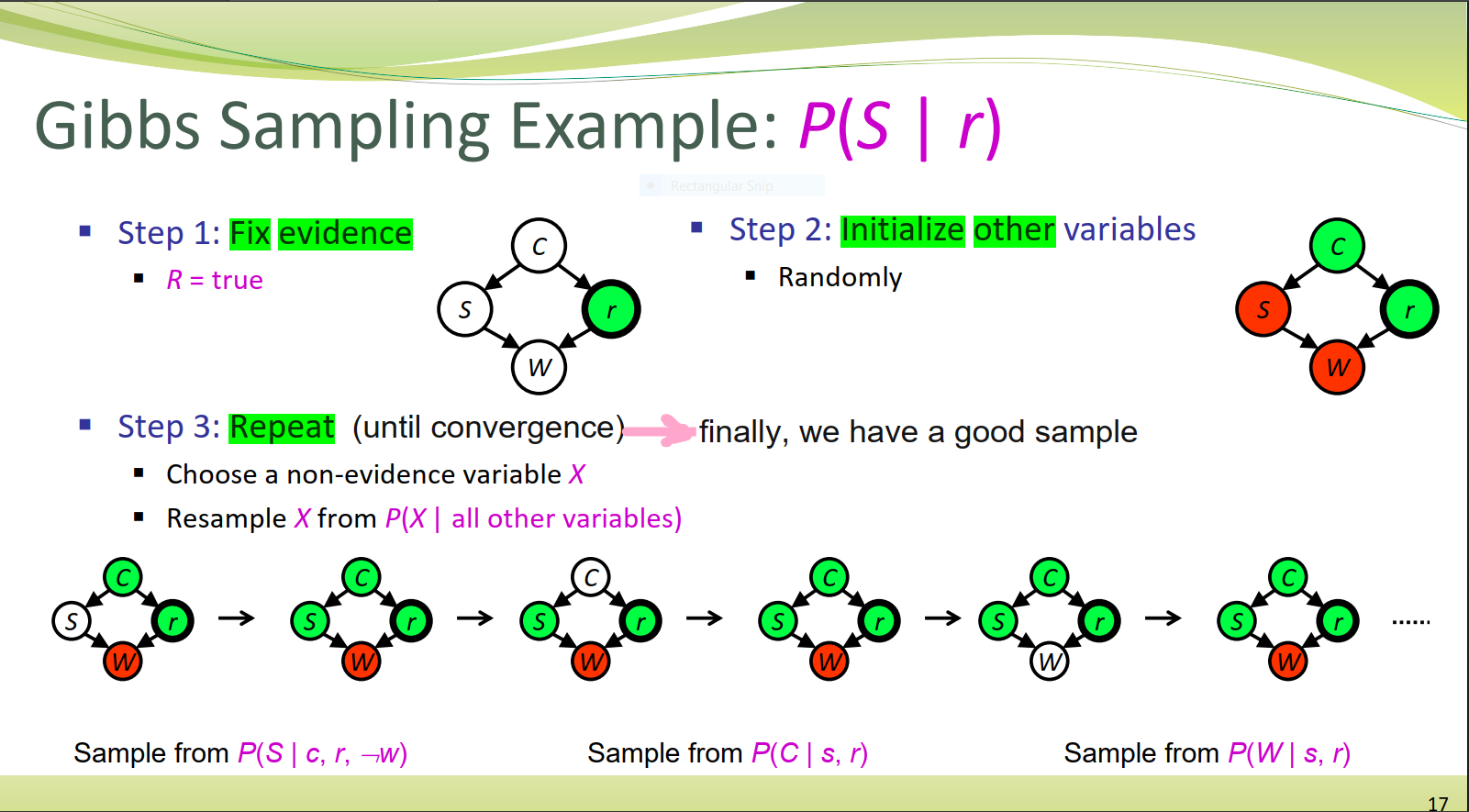
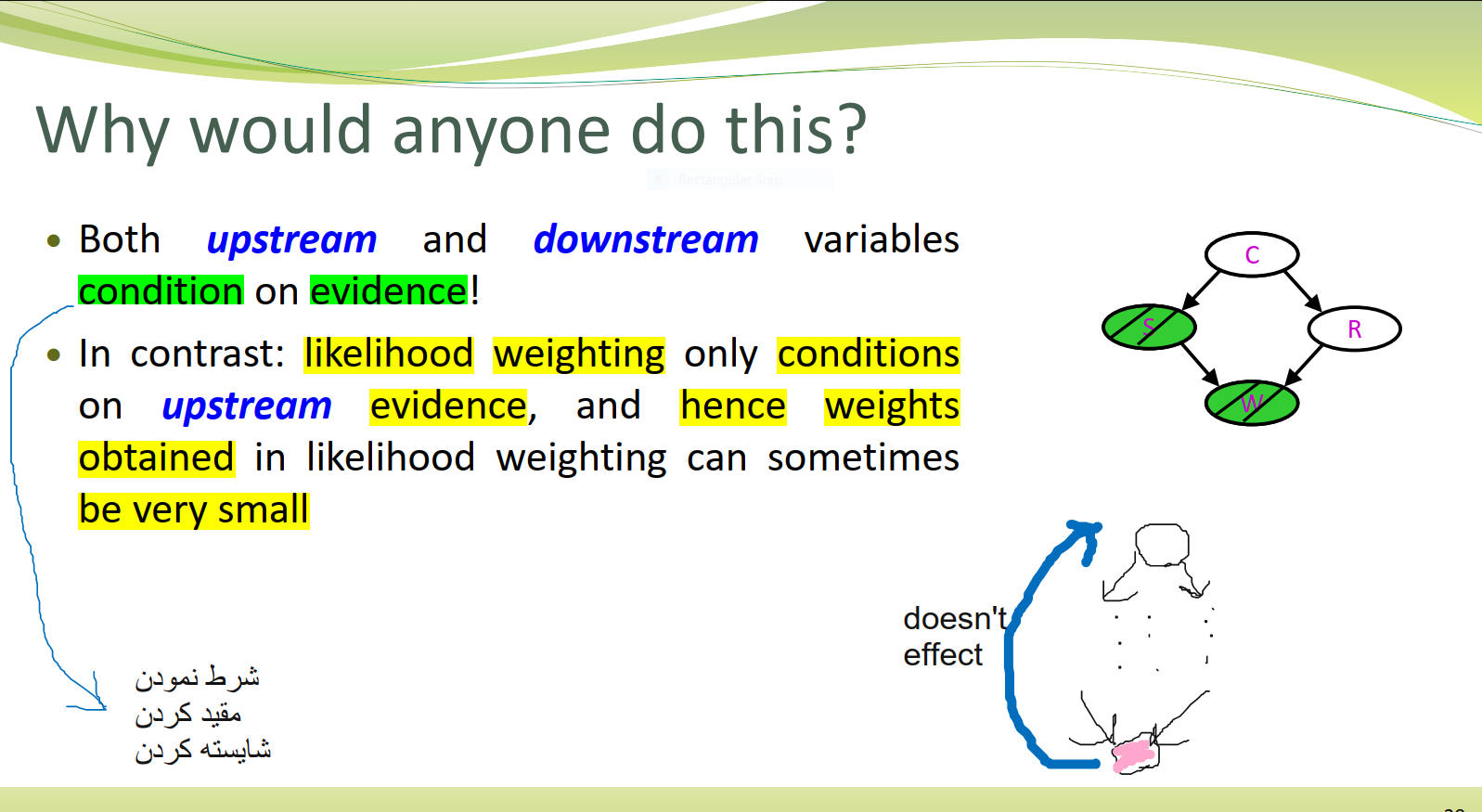
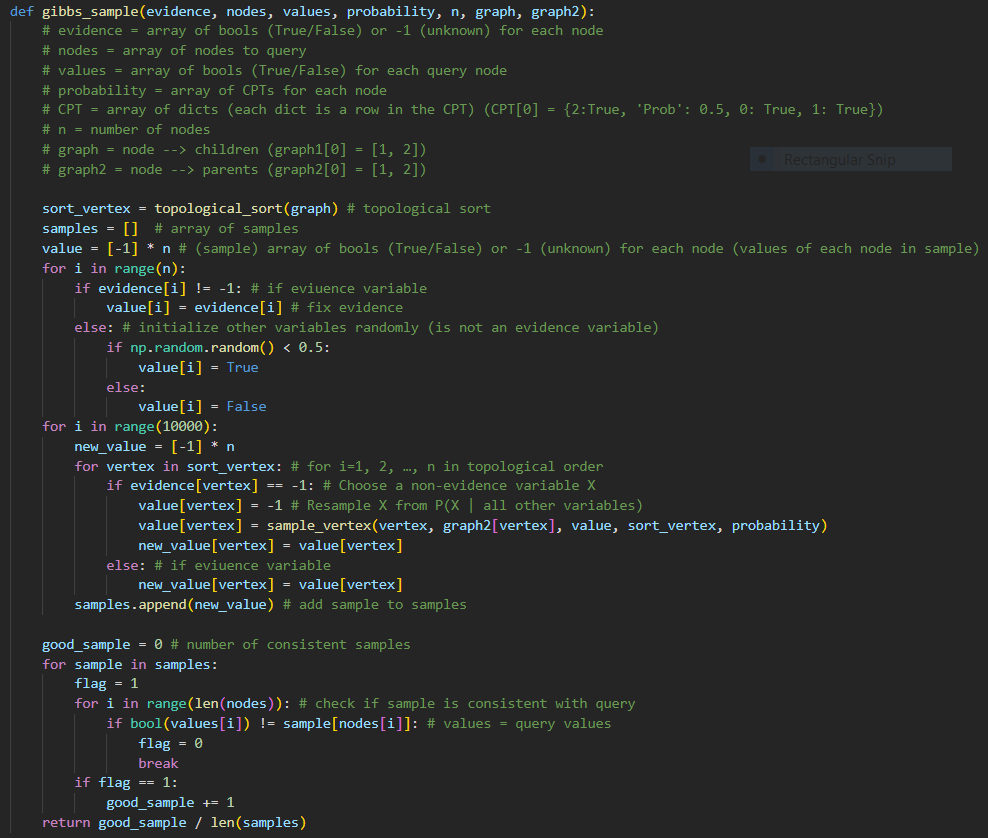
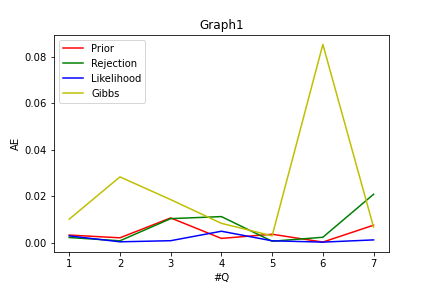
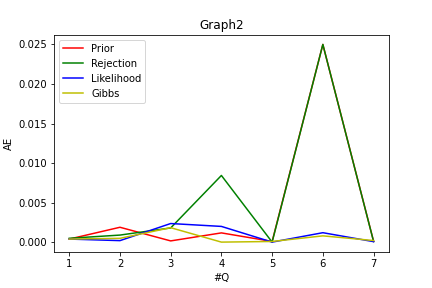
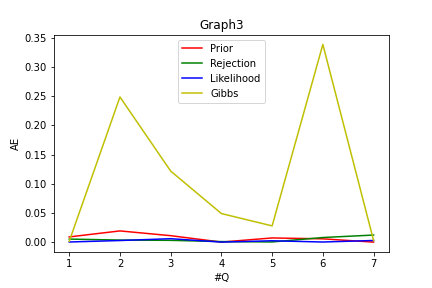
به نام خدا

# تمرین دهم هوش مصنوعی و سیستم های خبره فرزان رحمانی – 99521271

* بخش 1  
  ایده این روش برای بهتر کردن روش Rejection sampling می باشد. همان طور که در توضیح زیر آمده است این روش فقط نمونه هایی رو تولید می کند که با evidence ها سازگار هستند و نمونه ای دور ریخته نمی شود. فقط برا اینکه توزیع احتمال بهم نخورد و درست باشد برای هر evidence یک وزن در نظر میکنیم و وزن ها را در هم ضرب میکنیم و نمونه های موزون داریم.  
    
    
    
    
    
  برای پیاده سازی این روش نیز از سودو کدی که در زیر آمده است استفاده میکنیم. ابتدا یک وزن برابر با 1 در نظر میگیریم. گره های شبکه را topological sort می کنیم تا برای تعیین احتمال نود های فرزند نود های والد از قبل مقدار داشته باشند. سپس اگر گره evidence variable بود، مقدار ثابت و دلخواه را در نظر میگیریم و فقط وزن آن (احتمال رخداد به شرط والد ها) در w ضرب میکنیم. اگر هم evidence variable نبود، نمونه ای تصادفی از گره برمیداریم. در انتها چند تایی نمونه های هر گره به همراه وزن نمونه را به عنوان خروجی return می کنیم.  
    
    
  هم چنین در کد زیر پیاده سازی عملی این روش به همراه کامنت هایی که نشان می دهند هر خط چه کاری انجام می دهد آمده است:  
    
  در اینجا ابتدا گرافی که هر گره را به فرزندانش متصل می کند را topological sort می کنیم. سپس آرایه تمام نمونه ها را می سازیم. یک حلقه 10000 تایی که 10000 نمونه تولید میکند را آغاز میکنیم. آرایه value که مقدار هر گره را در نمونه نهایی نشان می دهد. value دارای سه مقدار True,False و -1 برای مقادیری که مقدار ندارند می باشد. سپس وزن w را برابر 1 تعریف میکنیم. شروع به پیمایش گره در topological order می کنیم. اگر گره evidence variable بود، مقدار ثابت و دلخواه را در نظر میگیریم و فقط وزن آن (احتمال رخداد به شرط والد ها) در w ضرب در مساوی میکنیم. اگر هم evidence variable نبود، نمونه ای تصادفی از گره برمیداریم. در انتها نمونه را به همراه وزنش در آرایه samples میریزیم. متغیری برای مجموع وزن ها و مجموع وزن هایی که query می خواهد تعریف میکنیم. سپس در نمونه ها for میزنیم. برای هر نمونه مقادیر کویری را با نمونه مقایسه میکنیم تا ببینیم که با query سازگار هست یا خیر. وزن نمونه را با مجموع وزن ها جمع میکنیم. همچنین اگر با query سازگار بود با good\_sample جمع میکنیم. در انتها احتمال query را return می کنیم(وزن مطلوب تقسیم بر وزن کل نمونه ها ). همچنین در این تابع از یک تابع کمکی که در زیر آماده است استفاده کرده ایم:  
    
    
    
  در این تابع ابتدا متغیری برای سطر مطلوب در cpt که با مقادیری از نمونه که مقدار دهی شده اند (parents) سازگارند تعریف میکنیم. سپس در cpt پیمایش میکنیم. برای هر row که یک دیکشنری در کلید هایش for میزنیم. اگر مقداری از row فعلی با مقادیر مقدار داده شده همخوانی نداشته باشند سراغ سطر بعدی می رویم. در غیر این صورت این row همان سطری است که مد نظر ماست. سپس از آن کپی میگریم و در متغیری که اول تعریف کردیم میریزیم. در انتها در این row که یک دیکشنری است مقدار احتمال را بر میداریم وreturn می کنیم.  
    
    
    
  این روش نمونه برداری همان طور که در زیر آمده است خوب است چون همه نمونه ها استفاده می شوند . ولیکن فقط گره هایی که در بالای شبکه هستند روی گره های پایینی اثر دارند و اگر evidence های ما در پایین شبکه باشند روی گره های بالایی تاثیر نمی گذارند. و بعضی مواقع وزن نمونه هایی که میگیریم بسیار کوچک و عملا بی استفاده هستند. به همین دلیل روش نمونه برداری دیگری را در ادامه معرفی میکنیم بنام Gibbs .
* بخش 2  
  این روش داری 3 گام می باشد. در ابتدا evidence variable ها را با مقادیر داده شده ثابت میکنیم. سپس بقیه متغیر ها را به صورت تصادفی مقدار دهی میکنیم. در مرحله آخر که به تعداد دلخواه تکرار می شود (تا همگرایی) یکی از non-evidence variable ها انتخاب میکنیم. سپس آن را به شرط مقدار همه متغیر های دیگر باز نمونه برداری میکنیم. در محاسبات فقط گره هایی که متغییر انتخابی در CPT شان وجود دارند نیاز به بررسی و join شدن دارند.   
    
    
  مزیت این روش این است که evidence variables ها رو تمام گره ها تاثیر خود را می گذارند و فقط از بالا به پایین مانند روش likelihood sampling نمی باشد.  
    
    
    
    
    
    
    
    
  هم چنین در کد زیر پیاده سازی عملی این روش به همراه کامنت هایی که نشان می دهند هر خط چه کاری انجام می دهد آمده است:  
    
    
  در اینجا ابتدا گرافی که هر گره را به فرزندانش متصل می کند را topological sort می کنیم. سپس آرایه تمام نمونه ها را می سازیم. آرایه value که مقدار هر گره را در نمونه نهایی نشان می دهد. value دارای سه مقدار True,False و -1 برای مقادیری که مقدار ندارند می باشد. سپس درتعداد گره for میزنیم. اگر evidence variable بود مقدار ثابت آن را میگذاریم در غیر این صورت به صورت تصادفی مقدار گره را در ارایه value قرار می دهیم. سپس یک حلقه 10000 تایی میزنیم تا 10000 نمونه برداریم. آرایه new\_value را شبیه به آرایه value تعریف میکنیم. در topological order راس ها را پیمایش میکنیم. اگر evidence بود مقدار قبلی و ثابت را نگه میداریم در غیر این صورت به شرط تمامی دیگر متغیر ها resample انجام میدهیم. بعد از این حلقه آرایه new\_value را به نمونه ها اضافه میکنیم. متغیری برای تعداد نمونه هایی که query می خواهد تعریف میکنیم. سپس در نمونه ها for میزنیم. برای هر نمونه مقادیر کویری را با نمونه مقایسه میکنیم تا ببینیم که با query سازگار هست یا خیر. همچنین اگر با query سازگار بود تعداد good\_sample را با 1 جمع میکنیم. در انتها احتمال query (تعداد مطلوب تقسیم بر تعداد کل نمونه ها ) را return می کنیم.
* بخش 3  
    
  درون پوشه inputs پوشه ای با نام 3 ایجاد کردیم. که input.txt, q-input.txt و همچنین عکس شبکه bn.PNG می باشند. همچنین خروجی اجرای این ورودی ها در پوشه outputs با نام 3.png,3.txt قابل مشاهده است.  
    
    
  برای نمونه های پوشه 1:  
  با توجه به AE در این نمودار میبینیم که از نظر عملکرد Likelihood بهترین عملکرد را داشته (کوئری های اول، دوم، سوم ، پنجم، ششم و هفتم) و همچین Gibbs بد ترین عملکرد را. بین Rejection و Prior در کل Prior بهتر بوده است (کوئری های چهارم ، ششم و هفتم) ولی در بعضی query ها Rejection خطای کمتری داشته است(کوئری های پنجم و دوم). همچنین در query های دوم و ششم روش Gibbs خطای بسیار زیادی نسبت به بقیه دارد و عملکرد بدی داشته است بخصوص در query ششم.   
  کوئری هایی که Gibbs خطای چشمگیر و قابل ملاحظه ای دارد:  
  کوئری دوم: [{"A": 1 , "B" : 1}, {"C": 1 , "D" : 1}]  
  کوئری ششم: [{"A": 1,"D":1}, {"C": 0 , "B" : 1}]  
    
    
  برای نمونه های پوشه2 :  
  با توجه به AE در این نمودار میبینیم که از نظر عملکرد روش Rejection بدترین عملکرد را داشته است به خصوص در ورودی چهارم و ششم که نسبت به بقیه روش ها خطای چشمگیری دارد. سه روش دیگر تقریبا خطای نزدیک به همه داشته اند و روی بعضی query ها هر یک کمترین خطا را داشته اند. اما در کل روش Gibbs بهتر از بقیه عمل کرده است (کوئری های اول، چهارم و ششم). از بین Prior و Likelihood نیز روش Prior در کل بهتر بوده است(سوم و چهارم) و همچنین Likelihood در کوئری دوم و هفتم خطای کمی داشته است.   
  کوئری هایی که Rejection خطای چشمگیر و قابل ملاحظه ای دارد:  
  کوئری چهارم: [{"A": 1,"D":1}, {"S": 0}]  
  کوئری ششم: [{"A": 1,"S":1}, {"E": 0 , "B" : 1}]  
    
    
  برای نمونه های پوشه3 :  
  با توجه به AE در این نمودار میبینیم که از نظر عملکرد روش Gibbs بدترین عملکرد را داشته است به خصوص در ورودی دوم و سوم و ششم که نسبت به بقیه روش ها خطای چشمگیری دارد. سه روش دیگر تقریبا خطای نزدیک به هم داشته اند و روی بعضی query ها هر یک کمترین خطا را داشته اند. اما در کل روش Likelihood بهترین عملکرد را داشته است(کوئری اول ، دوم ، چهارم و ششم). از بین Prior و Rejection نیز روش Rejection در کل بهتر بوده است (کوئری دوم، سوم و پنجم) و خطای کمتری نسبت به Prior داشته است. Prior در کوئری هفتم کمترین خطا را داشته است.  
  کوئری هایی که Gibbs خطای چشمگیر و قابل ملاحظه ای دارد:  
  کوئری دوم: [{"C": 1 , "R" : 1}, {"S": 1 , "W" : 1}]  
  کوئری چهارم: [{"C": 1,"S":1}, {"W": 0}]  
  کوئری ششم: [{"C": 1,"W":1}, {"S": 0 , "R" : 1}]

پایان