زنجیره مارکوف مونته کارلو (MCMC)[26]

مقدمه‌ای بر روش مونته کارلو[27]

روش مونته کارلو عموما به روش‌های محاسباتی‌ای گفته می‌شود که از اعداد تصا دفی بهره می‌جوید. به طور خاص‌تر می‌توان آن را دسته‌ای الگوریتم‌ها دانست که اگر شبیه‌سازی کامپیوتری را به مدت کافی ادامه دهیم، به ما جواب درست را می‌دهد. در ادامه چند الگوریتم ساده ولی مشهور که از روش مونته‌کارلو بهره می‌جویند آورده شده است اما قبل از آن بهتر است در رابطه با اعداد تصادفی شفاف‌سازی‌هایی را انجام دهیم.

اعداد تصادفی

به دنباله‌ای از اعداد که با یک توزیع احتمال مشخص تولید شده‌اند، اعداد تصادفی گفته می‌شود. یک نمونه ساده از اعداد تصادفی اعدادی است که پس از پرتاب یک تاس ظاهر می‌گردند. در پرتاب تاس اعداد 1 تا 6 با احتمال‌های برابر یعنی

$P(1)=P(2)=P(3)=P(4)=P(5)=p(6)=\frac{1}{6}$ ظاهر می‌شوند. با توجه به این که عدد ظاهر شده در پرتاب n ام تاس به عدد ظاهر شده در پرتاب $(n-1)$ام تاس بستگی ندارد، می‌گوییم که این آزمایش خاصیت بی‌حافظگی دارد. منظور از تصادفی بودن دارا بودن همین خاصیت بی‌حافظگی است لازم به ذکر است که عبارت «اعداد تصادفی» در این متن و دیگر متون علوم کامپیوتر با اعداد تصادفی‌ای که در ریاضیات شرح داده می‌شوند متفاوت است زیرا که در عمل یک کامپیوتر قادر به تولید اعدادی که واقعا تصادفی باشند نیست و صرفا اعدادی را تولید می‌کنند که به ظاهر تصادفی هستند. به این اعداد، اعداد شبه‌تصادفی[28] گفته می‌شود. الگوریتم‌های زیادی برای تولید اعداد شبه‌تصادفی مطرح شده است که بررسی آن‌ها از هدف این متن دور است. با این وجود توصیه می‌شود که برنامه‌نویسی که از روش مونته‌کارلو استفاده می‌کند، از چگونگی کارکرد و محدودیت‌های الگوریتم‌های تولید اعداد تصادفی باخبر باشد تا در شبیه‌سازی‌های خود به مشکل برنخورد. در ادامه ما دو نوع اعداد تصادفی کاربردی را شرح می‌دهیم.

اعداد تصادفی یکنواخت

اگر احتمال به دست آوردن هر عدد حقیقی در یک بازه (مانند [0,1]) برابر با دیگر اعداد باشد، اعداد به دست آمده را اعداد تصادفی یکنواخت[29] می‌نامیم. چنین توزیعی را می‌توان همچون یک تاس N وجهی سالم در نظر گرفت که هر عددی که بر روی تاس است را با احتمال $\frac{1}{N}$ برمی‌گرداند.

اعداد تصادفی گاوسی (توزیع نرمال)

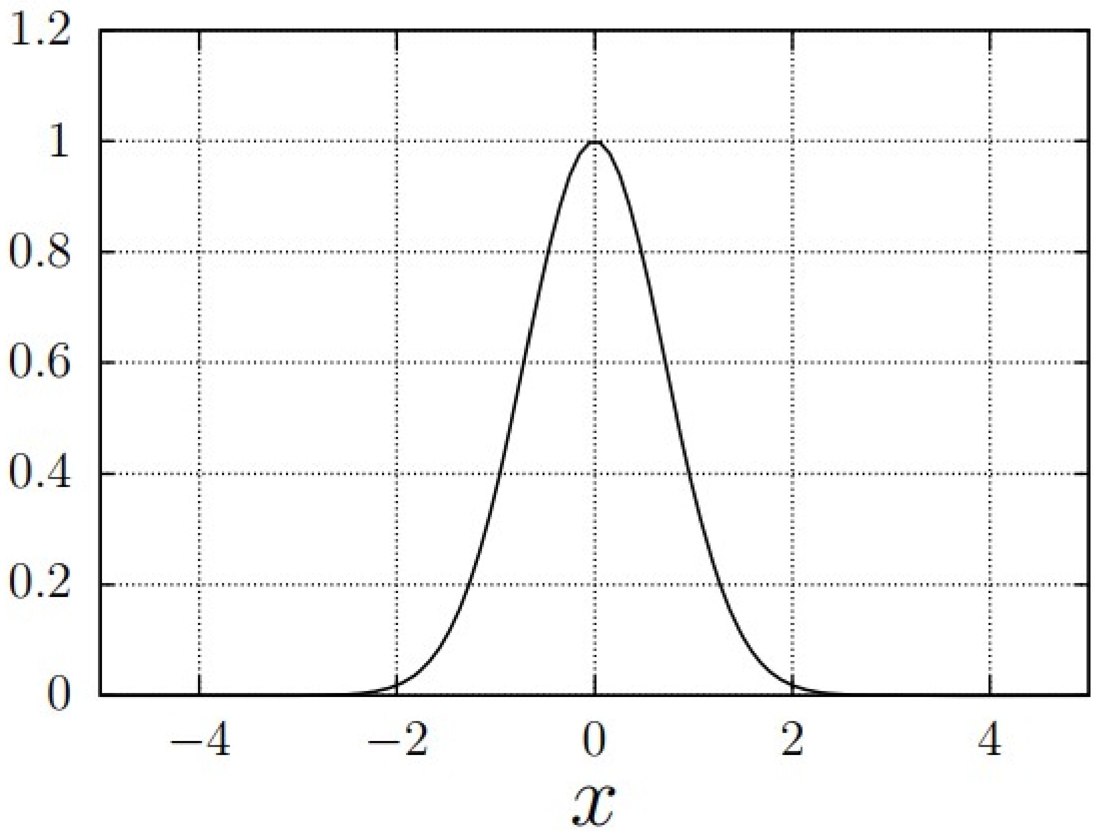
---
[26] Markov chain Monte Carlo
[27] Monte Carlo method
[28] Pseudorandom numbers
[29] Uniform Random Numbers

اعداد تصادفی گاوسی[30] در بسیاری از مسائل ریاضی و کامپیوتری کاربرد دارد. این اعداد تصادفی با تابع گاوسی e^{-x^2} که نمودار آن در زیر آورده شده است، به دست می‌آید. با توجه به این که مقادیر این تابع با بزرگ شدن به سرعت کم می‌شود، آن را طبق فرمول زیر نرمال‌سازی می‌کنیم.

$$P(x)=\frac{e^{-x^2}}{\sqrt{\pi}}$$

که در آن $\sqrt{\pi}$ با انتگرال گیری از این تابع به دست آمده است.

$$\int_{-\infty}^{\infty} e^{-x^2}dx=\sqrt{\pi}$$



تصویر A نمودار تابع گاوسی e^{-x^2}

[30] Gaussian Random Numbers

یک مثال ساده از متد مونته کارلو (بدون زنجیره مارکوف)

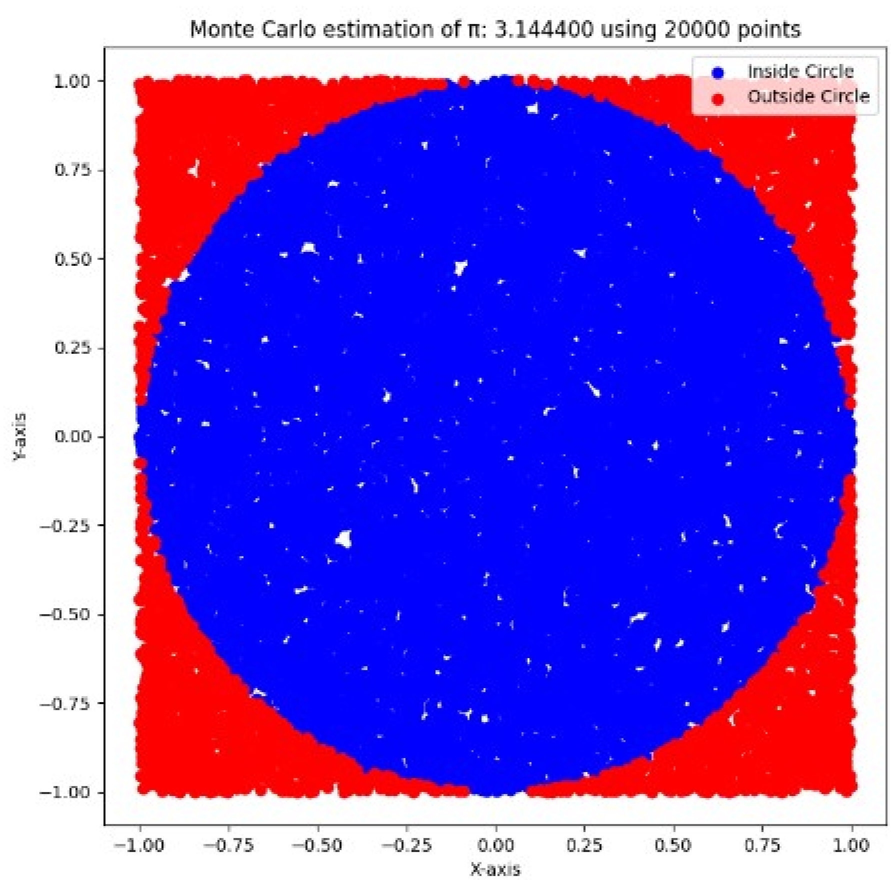
قبل از این که به حالت خاص‌تر MCMC بپردازیم بهتر است برای آشنایی با سازوکار روش مونته‌کارلو، یک مثال از آن بدون استفاده از زنجیره مارکوف بزنیم. آن چه پیش رو می‌آید یک مقدمه ساده و کلاسیک برای آشنایی با روش مونته کارلو است که در آن به کمک تکرارهای طولانی تولید اعداد تصادفی، به تقریب محاسبه عدد پی می‌پردازیم.

روش بدین گونه است:

دایره‌ای به شعاع واحد و مرکز مبدای مختصات را در نظر بگیرید که درون یک مربع با طول ضلع ۲ قرار دارد. نقاط تصادفی زیادی را تولید می‌کنیم و بررسی می‌کنیم که آیا این نقاط درون دایره می‌افتند و یا بیرون دایره. با توجه به نسبتی که بین مساحت دایره و مربع و تعداد نقاطی که درون دایره و خارج از دایره می‌افتند، می‌توان نوشت:

$$\frac{\textit{Number of points inside the } \mathrm{circle}}{\textit{Number of all points}} \approx \frac{\textit{Area of the } \mathrm{circle}}{\textit{Area of the } \mathrm{square}}$$

بدیهی است که هر چقدر تعداد نقاط تصادفی تولید شده بیش‌تر باشد، برنامه ما نتیجه‌ای دقیق‌تر را طولانی‌تر بیرون می‌دهد. خروجی برنامه pi.py در فولدر mcmc به ازای 20000 نقطه تصادفی چنین خواهد بود.



الگوریتم متروپلیس-هستینگز

الگوریتم متروپلیس-هستینگز الگوریتمی برای به دست آوردن یک سری نمونه‌های تصادفی از یک تابع توزیع احتمالی است. این الگوریتم زمانی به کار می‌آید که نمونه‌گرفتن به صورت مستقیم از یک توزیع احتمال نشدنی یا هزینه‌بر و سخت باشد. الگوریتم متروپلیس-هستینگز خصوصا در مواقعی که ما تابع توزیع احتمال یک پدیده تصادفی را نداریم و یا حدس پارامترهای آماری (مانند واریانس و میانگین) برای یک جمعیت بزرگ سخت است بسیار ترجیح داده می‌شود. با نمونه‌برداری از یک توزیع احتمال می‌توان رفتار و ماهیت آن تابع را تقریب زد و به کمک این تقریب عملیات‌های مربوط به آمار و احتمالات یک سیستم پیچیده (مثل شبیه‌سازی حرکات کاتوده‌ای مولکول‌های گاز) را انجام داد .از همین جهت به کمک این الگوریتم به یک هیستوگرام احتمال پسین[31] دست خواهیم یافت. نمونه‌برداری گیبز[32] یک حالت تعمیم‌داده شده از الگوریتم متروپلیس-هستینگز برای نمونه‌برداری از توابع احتمالی چند متغیره است.

الگوریتم متروپلیس-هستینگز مانند کوهنوردی در یک روز مه‌آلود است. کوهنورد که جلوی خودش را نمی‌تواند ببیند به صورت تصادفی قدم برمی‌دارد. اگر آن قدم او را بالاتر از جای قبلی خود برد پس این قدم را می‌پذیرد ولی اگر این قدم او را پایین‌تر از جای خود برد تنها با احتمال خاصی آن قدم را می‌پذیرد. علت این که چرا این الگوریتم یک الگوریتم مونته‌کارلو زنجیره مارکوف است مشخص است: زنجیره مارکوف است چرا که هر قدم به قدم قبلی بستگی دارد و مونته‌کارلو است زیرا کوهنورد به صورت تصادفی قدم‌های زیادی برمی‌دارد.

شبه‌کد این الگوریتم به صورت کلی چنین است:

1. Initialise $x^{(0)}$.
2. For $i = 0$ to $N-1$
   - Sample $u \sim \mathcal{U}_{[0,1]}$.
   - Sample $x^\star \sim q(x^\star | x^{(i)})$.
   - If $u < \mathcal{A}(x^{(i)}, x^\star) = \min\left\{1, \frac{p(x^\star)q(x^{(i)}|x^\star)}{p(x^{(i)})q(x^\star|x^{(i)})}\right\}$

     $x^{(i+1)} = x^\star$

     else

     $x^{(i+1)} = x^{(i)}$

[31] Posterior probability
[32] Gibbs sampling

در شبه‌کد بالا u یک عدد تصادفی با توزیع یک‌نواخت و x^* یک نمونه تصادفی از q^* که تابع توزیع احتمالی پیش‌نهاد شده برای تقریب زدن است می‌باشد. q^* متغیری است که توسط برنامه‌نویس تعیین می‌شود به طوری که هر چقدر برنامه‌نویس اطلاعات بیش‌تری از تابع توزیع احتمالی مجهول داشته باشد، آن را با دقت و شباهت بیش‌تری می‌تواند انتخاب کند. خط سوم بررسی می‌کند که آیا عدد تصادفی نمونه‌برداری شده کم‌تر از احتمال پذیرش[33]حالت پیش‌نهاد شده است یا خیر. در صورت مثبت بودن جواب پس حالت پیش‌نهاد شده را به عنوان حالت بعدی اتخاذ می‌کنیم و در غیر این صورت گامی بر نمی‌داریم. محاسبه احتمال پذیرش و مقایسه آن با یک عدد تصادفی در واقع تکنیکی برای ایجاد موازنه بین بهره‌برداری-کشف[34] می‌باشد تا گاهی اوقات به زنجیره مارکوف اجازه حرکت به یک حالت با احتمال کم‌تر را بدهد. به صورت ساده‌تر می‌توان احتمال پذیرش را این گونه بیان کرد:

acceptance_prob = min(1, likelihood(proposed_state, observed_data_mean, observed_data_std) / likelihood(current_state, observed_data_mean, observed_data_std))

به عبارت ساده‌تربا ایجاد یک نسبت و تناسب بین مقدار درست‌نمایی[35] حالت پیش‌نهاد شده و مقدار درست‌نمایی حالت کنونی و سپس کمینه گرفتن بین آن کسر و عدد ۱ (برای این که مطمئن شویم که مقدار احتمال ما یک مقدار احتمال معتبر با حداکثر مقدار ۱ است) به این مقدار دست می‌یابیم که پس از مقایسه آن با یک عدد تصادفی از تکنیک بهره‌برداری-کشف استفاده می‌کنیم.

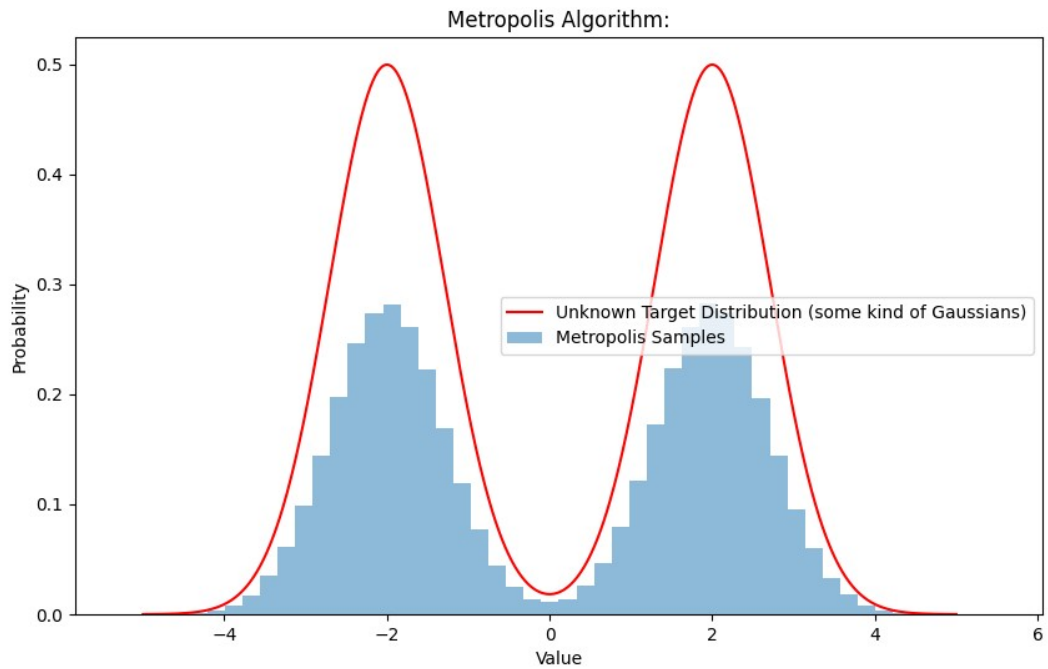
فایل metropolis_plot.py در پوشه mcmc برنامه‌ای برای نمونه‌برداری تصادفی از یک تابع توزیع احتمال نسبتا پیچیده گاوسی می‌باشد. در نهایت این برنامه هیستگورام نمونه‌های تصادفی گرفته شده را چاپ می‌کند:

---

[33]Acceptance probability

[34]رویکرد exploration - exploitation : بهره برداری – کشف : هر زمان که با امتحان کردن چیزها در مورد جهان بیاموزید، موازنه کشف یا بهره برداری یک معضل اساسی است. دوراهی بین انتخاب آنچه می دانید و چیزی نزدیک به آنچه انتظار دارید exploitation (بهره برداری) و انتخاب چیزی است که در مورد آن مطمئن نیستید و احتمالاً یادگیری بیشتر دارد exploration (کشف) . برگرفته از https://abadis.ir/entofa/exploration-exploitation/ نعیم ابراهیمیان

[35]Likelihood value

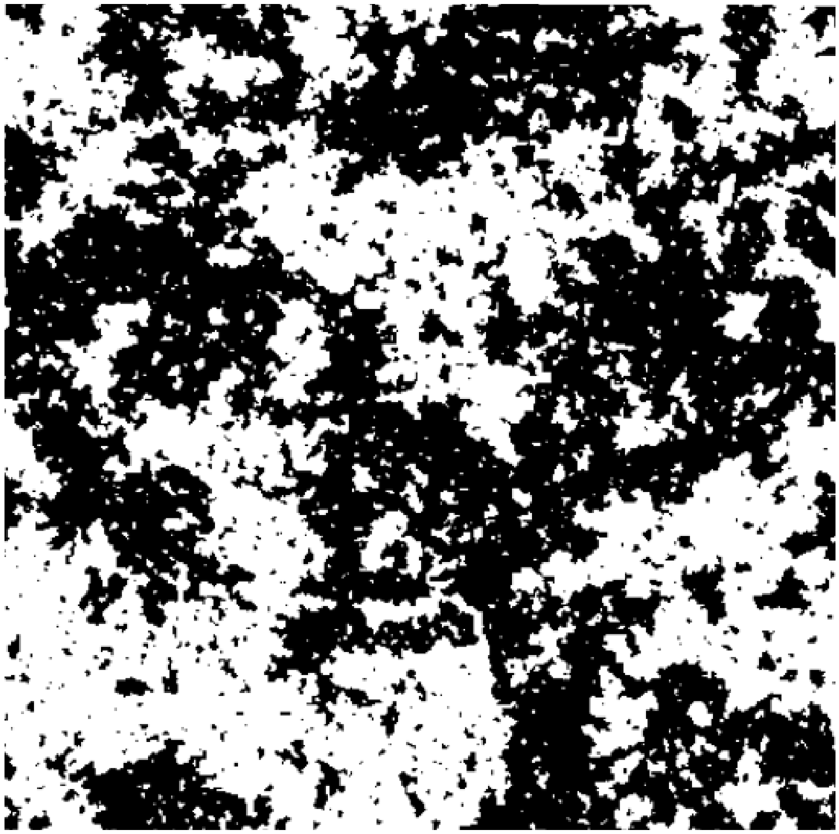


این خروجی با ۴۰ هزار تکرار به دست آمده است. بدیهی است که هر چقدر تعداد تکرارها بیش‌تر باشد نمونه‌برداری دقیق‌تر خواهد بود.

مثال: حذف نویز از عکس سیاه و سفید

برای حذف نویز در عکس ابتدا بهتر است مدل آیزینگ[36] را معرفی نمود. این مدل که به افتخار فیزیک‌دان برجسته ارنست آیزینگ نام‌گذاری شده است در واقع مدلی ریاضی برای توصیف آرایش فرومغناطیس‌ها می‌باشد. در این مدل ما با اشیایی سروکار داریم که دو حالت گسسته بیش‌تر ندارند: یا 1 یا -1. این اشیا در یک مشبکه[37] قرار می‌گیرند و از همین رو می‌توانند با همسایه‌هایشان کنش و واکنش داشته باشند. مدل آیزینگ از مدل‌های مطرح در الگوریتم‌های زنجیره مارکوف مونته کارلو است. در کاربرد مورد نظر ما (حذف نویز از عکس سیاه و سفید) مدل آیزینگ از آن جهت سودمند است که اولا می‌توان مقدار هر پیکسل را با دو مقدار 1 (سیاه) و -1 مشخص نمود و دوما هر پیکسل با همسایه خود در ارتباط است. در واقع از مشخصه دوم می‌توان هدف الگوریتم را حدس زد: احتمال این که یک پیکسل سیاه توسط پیکسل‌های سیاه احاطه شده باشد بیش‌تر از آن است که آن پیکسل سیاه تماما توسط پیکسل‌های سفید احاطه شده باشد.در صورت رخداد حالت دوم می‌توان با اطمینان نسبتا بالایی آن پیکسل را یک داده پرت و در نتیجه نویز در نظر گرفت.

[36]Ising model
[37]Lattice



یک پیکربندی عادی نزدیک دمای بحرانی از مدل آیزینگ دوبعدی

تابع جرمی احتمال تغییر حالت پیکسل i (که آن را با X_i نمایش می‌دهیم)

به کمک الگوریتم متروپلیس-هستینگز می‌توان نویز و داده‌های پرت را با توجه به نمونه‌های تصادفی تقریب زده حذف کرد؛ کافیست که در هر مرحله یک پیکسل را به تصادفی انتخاب کرد و سپس به کمک تابع توزیع احتمال پسین بررسی کرد که آیا تغییر حالت این پیکسل (از سیاه به سفید یا از سفید به سیاه) امتیازتوزیع احتمال پسین را بالا می‌برد یا خیر. این امتیاز را می‌توان با محاسبه احتمال پذیرش آن پیکسل در نمونه‌برداریمان حساب کرد. در صورت مثبت بودن این سوال پس می‌توان آن پیکسل را نویز در نظر گرفت و رنگ آن را معکوس کرد تا هم‌رنگ همسایه‌هایش شود.این مرحله را تا حد ممکن تکرار می‌کنیم تا به عکس اصلی بدون نویز مد نظرمان بیش‌تر نزدیک شویم.

در این برنامه ما به سه پارامتر نیاز داریم:

- پارامتر α که بیان‌گر احتمال تغییرحالت پیکسل‌ها در یک تصویر است. هر چه این مقدار بیش‌تر باشد عکس نویزی‌تر است.
- پارامتر β که بیان‌گر میزان تکیه کردن ما بر مدل آیزینگ ماست. هر چه این مقدار بیش‌تر باشد یک‌پارچگی و هم‌آهنگی بین پیکسل‌های همسایه بیش‌تر است.

- مقدار γ که میزان نفوذ مدل آیزینگ بر پارامتر α را مشخص می‌کند. از آن جا که توابع جرمی احتمال حالات اشیای مدل آیزینگ توابع سیگمویدی[38] می‌باشد، می‌توان نوشت:

$P(X_i=1)=\frac{1}{1+e^{(-2\gamma)}}$ و $P(X_i=1)=\frac{1}{1+e^{(2\gamma)}}$. همچنین داریم:

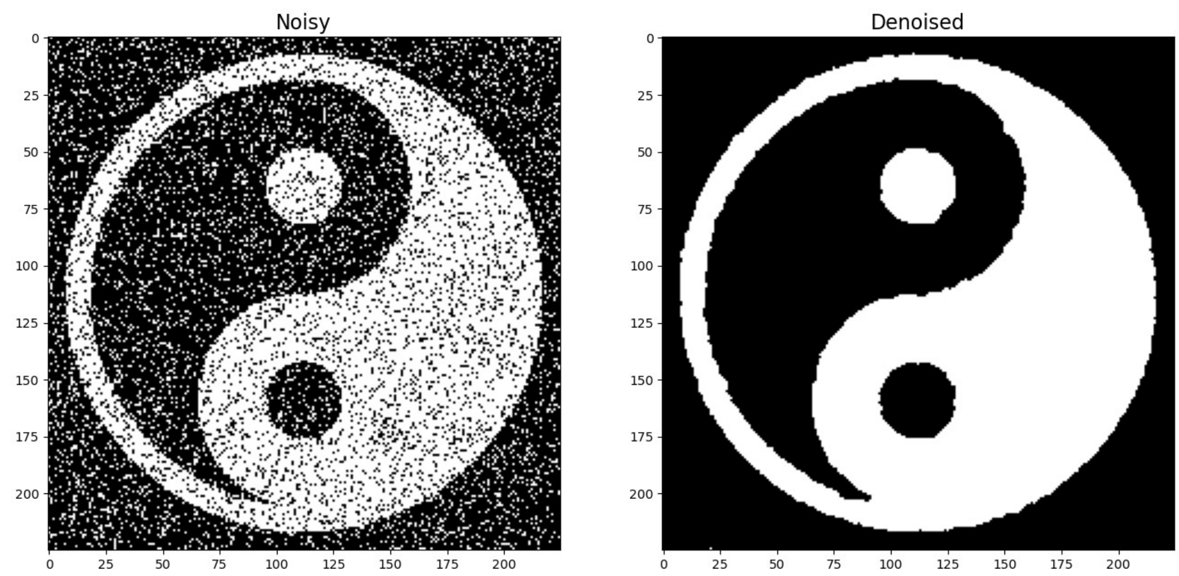
$P(flipping\, a\ pixel)=p.\frac{1}{1+e^{(2\gamma)}}+(1-p)\frac{1}{1+e^{(-2\gamma)}}$ که از آنجا که احتمال‌تغییر حالت پیکسل‌ها همان α می‌باشد با جای‌گذاری و حل به ازای γ خواهیم داشت:

$$\gamma=\frac{1}{2}\log(\frac{1-\alpha}{\alpha})$$

- پارامتر تکرار شبیه‌سازی مونته کارلو که آن را با I نشان می‌دهیم.

برنامه noise.py در پوشه mcmc دو حالت دارد که کاربر به عنوان ورودی وارد می‌کند. در حالت noisify کاربر با آدرس دادن تصویر و مشخص نمودن α تصویر مورد نظر را نویزدار می‌کند. در حالت de-noise کاربر با وارد کردن آدرس تصویر نویز دار و همچنین مشخص نمودن α و β و I تصویر نویز-زدایی شده را تحویل می‌گیرد.

پردازش‌های مربوط به تصویرهای ورودی توسط کتابخانه PIL و numpy صورت می‌گیرد؛ عکس در ابتدا سیاه-سفید می‌شود و سپس اطلاعات پیکسلی آن درون یک آرای ریخته می‌شودو در یک فایل متنی ذخیره می‌گردد. عکس این عمل (تبدیل متن به عکس) نیز درون برنامه کدنویسی شده است. عکس زیر نتیجه حاصل شده با پارامترهای $\alpha=0.1, \beta=0.8$ و با ۵۰۰ هزار تکرار است.



[38]Sigmoid function