

EST-46115: Modelación Bayesiana

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — HMC.

Objetivo. Estudiaremos el uso de modelos de física para simular cadenas de Markov con el objetivo de resolver integrales derivadas de un modelo probabilístico. Esto nos lleva a estudiar el método que utiliza el estado del arte en simulación para resolver inferencia bayesiana. Lo lograremos con lo que ya hemos visto: método de aceptación-rechazo, Metropolis-Hastings y demás.

Lectura recomendada: Capítulo 9 de [?]. Los artículos [?] y [?], aunque mas avanzados, tienen una muy buena explicación del mecanismo teórico y algorítmico que utiliza HMC.

1. INTRODUCCIÓN

El interés es poder resolver

$$\mathbb{E}[f] = \int_{\Theta} f(\theta) \pi(\theta|y) d\theta. \quad (1)$$

donde $\theta_t \sim \pi(\theta|y)$ y existe una relación $\theta_t = h(\theta_{t-1}, \xi_{t-1})$.

Ya vimos que el método Metrópolis-Hastings es **suficientemente general** en el sentido de que no importa la distribución objetivo o la distribución de propuesta. Si las condiciones lo permiten se generarán muestras en el *largo plazo*. Sin embargo, el **tiempo** para llegar a ese estado de equilibrio o la **correlación** entre muestras sucesivas puede impactar nuestros estimadores y en ese caso generar resúmenes pobres.

Como vimos, el método de MH está *atraído* por las zonas de alta densidad. Así que una extensión *natural* es considerar distribuciones de propuesta que *apunten* en dirección donde **crece** la densidad. Esto se ha estudiado como el método de difusiones de Langevin metropolizadas (MALA por sus siglas en ingles, [?]).

1.1. Extensiones: MALA

La distribución propuesta está dada por

$$q(\cdot|\theta_t) = \mathbf{N}(\theta_t + \tau \nabla \log \pi(\theta_t), 2\tau). \quad (2)$$

Nota que es prácticamente una caminata aleatoria salvo que la media apunta en **dirección de ascenso** de la densidad objetivo.

Recordemos nuestra ilustración anterior que corresponde a la siguiente distribución objetivo

$$\theta \sim \mathbf{N}(\mathbf{m}, \mathbf{S}), \quad \mathbf{m} = (1, 2)^\top, \quad \mathbf{S} = \begin{pmatrix} 1 & .75 \\ .75 & 1 \end{pmatrix}. \quad (3)$$

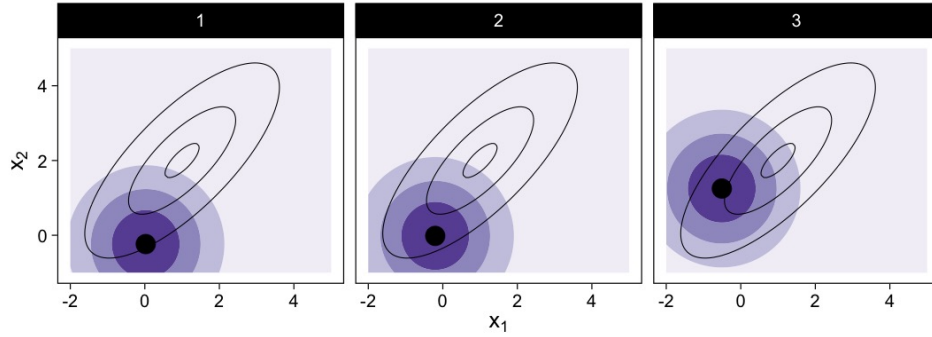


FIGURA 1. *Propuestas Gaussianas (morado) contra densidad objetivo (línea sólida). Tres primeras iteraciones.*

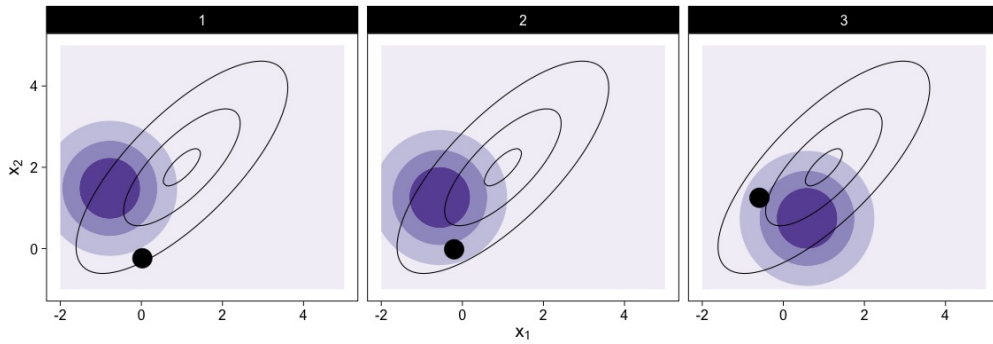


FIGURA 2. *Propuestas dirigidas por gradiente morado) contra densidad objetivo (línea sólida). Tres primeras iteraciones.*

La formulación de las difusiones de Langevin metropolizadas proviene de modelos matemáticos para fenómenos físicos como la difusión de calor en un medio. El algoritmo surge por el interés de resolver numéricamente este modelo formulado como una ecuación diferencial estocástica. La discretización es altamente volátil y, a menos que se utilicen mecanismos de geometría diferencial, su simulación no será exacta. Utilizar el mecanismo de MH ayuda a preservar las buenas propiedades de la solución numérica. Puedes leer mas de esto en [?] o puedes también al libro de [?].

1.2. Observaciones:

La alta dependencia en un parámetro de escala τ puede ocasionar problemas. Especialmente cuando la distribución es muy angosta. Para aliviar esto se puede incorporar información de segundo orden. Sin embargo la teoría necesaria (geometría diferencial) y el *software* aún no permiten una adopción masiva ([? ?]).

2. RESOLVIENDO PROBLEMAS COMUNES

Una alternativa a estos problemas conceptuales, teóricos y computacionales se ha alcanzado con el método de simulación Hamiltoniano o Híbrido (HMC).

Como ejemplo de situaciones que pueden ser complicadas consideremos de nuevo nuestro modelo Normal en \mathbb{R}^2 . Ahora consideraremos un modelo con mayor correlación:

$$\theta \sim N(m, S), \quad m = (1, 2)^\top, \quad S = \begin{pmatrix} 1 & .90 \\ .90 & 1 \end{pmatrix}. \quad (4)$$

Y también consideremos utilizar una normal estandar (multivariada) como distribución propuesta.

En este ejemplo incorporar mayor correlación puede parecer un artificio un tanto arbitrario. Sin embargo, considera que en aplicaciones normalmente no conocemos las dependencias condicionales en nuestro modelo y es usual observar correlaciones altas.

- **El problema:** la alta tasa de rechazo que tendremos. Incluso si estamos en el punto de mayor densidad.

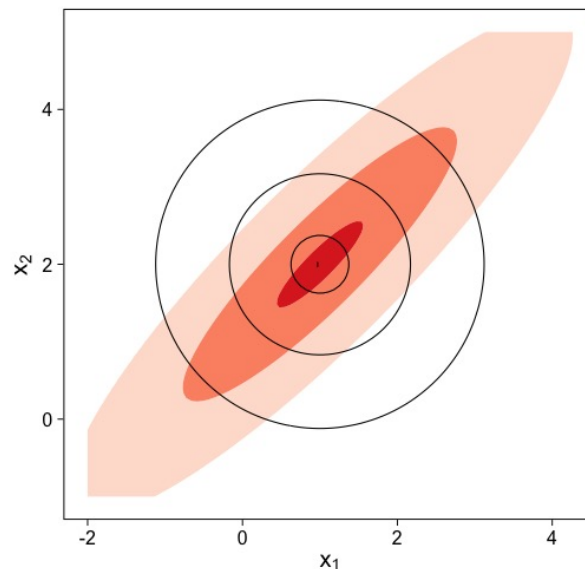


FIGURA 3. Curvas de nivel de un modelo con alta correlación (rojo). Curvas de nivel de un modelo Gaussiano estandar como propuesta (línea sólida).

```

1 mu ← c(1, 2)
2 Sigma ← matrix(c(1, .90, .90, 1), nrow = 2)
3 objetivo ← ModeloNormalMultivariado$new(mu, Sigma)
4 propuesta.std ← ModeloNormalMultivariado$new(c(1,2), diag(c(1,1)))

```

En el ejemplo la distribución propuesta no parece tan mala. Las distribuciones marginales tienen exactamente la misma dispersión que le modelo conjunto. ¿Pero por qué generaría tantas simulaciones rechazadas?

Alternativas como muestreo con el **método de Gibbs** (implementado en BUGS ó JAGS) resuelven el problema de altas tasas de rechazo al muestrear de las marginales. De hecho, se consigue una tasa de aceptación de 100 %. El inconveniente es que usualmente esto crea cadenas con mayor correlación entre iteraciones y en consecuencia sufre de **exploración ineficiente** de la distribución objetivo.

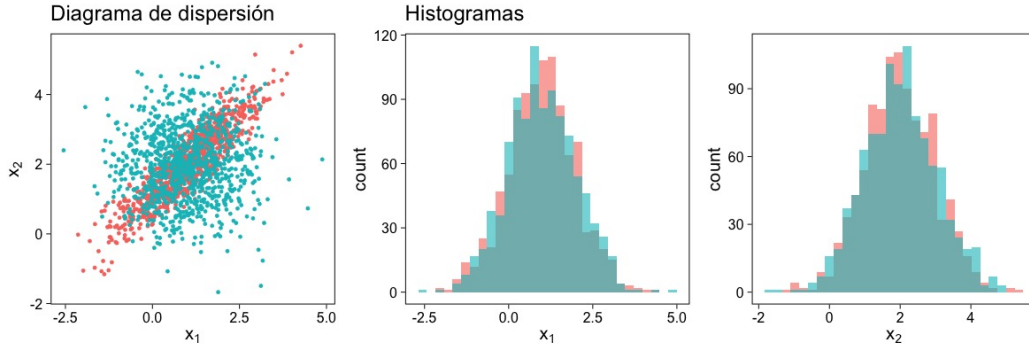


FIGURA 4. Muestras de la distribución objetivo (salmón) y la distribución propuesta (azul).

El muestreador de Gibbs fue el que popularizó el cómputo de muestreo en aplicaciones mas diversas (en comparación con Metropolis-Hastings) en la década de los 90s. Sin duda sin la contribución de este muestreador se hubiera retrasado la adopción de métodos Bayesianos.

Se pueden aliviar estos problemas de muchas formas. Una de ellas es **re-parametrizando** el problema. Por ejemplo, podemos utilizar la técnica de **cambio de variables**. Es decir, cambiar de

$$\theta \sim N(m, S), \quad \text{a} \quad \tilde{\theta} \sim N(0, I), \quad (5)$$

donde $I \in \mathbb{R}^{p \times p}$ denota la matriz identidad, y $\tilde{\theta}$ la variable con entradas de-correlacionadas.

Se puede utilizar descomposición en valores singulares o descomposición de Cholesky para expresar nuestro problema de muestreo en términos de una variable aleatoria con media 0 y varianza 1. Por ejemplo, consideremos la descomposición de Cholesky de la matriz de covarianzas $S = LL^T$. Por propiedades del operador varianza para vector, tenemos que

$$\mathbb{V}(\theta) = \mathbb{V}(L\tilde{\theta}) = L\mathbb{V}(\tilde{\theta})L^T = S, \quad (6)$$

donde $\tilde{\theta} \sim N(0, I)$.

Se pueden utilizar, además, técnicas de **Gaussianización** de variables (como la **transformación Rosenblatt**) pero esto implica conocer la estructura de correlación del problema. En aplicaciones es inusual tener conocimiento de esto.

3. EXPLORACIÓN CON UN POCO DE FÍSICA

Imaginemos que la función de densidad corresponde ahora a un *bowl*. Podemos explorar esa superficie rodando una pelota. Donde denotaremos su **posición** en el *bowl* por medio de

$$\theta(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^p. \quad (7)$$

El argumento lo consideraremos un **tiempo ficticio** t que nos ayudará a registrar la **posición** de la pelota en cualquier momento, $\theta(t)$.

De esta manera, pensemos que la pelota la dejamos correr desde un punto inicial $\theta(0)$ y nos fijamos en dónde va al tiempo T . Es decir, registramos el punto $\theta(T)$. En nuestro contexto de muestreo, la **posición inicial** es el valor actual de nuestra cadena de Markov y la posición final de la pelota es la propuesta para nuestra nueva iteración. Lo podemos denotar como

$$\theta_n = \theta(0), \quad \theta_\star = \theta(T). \quad (8)$$

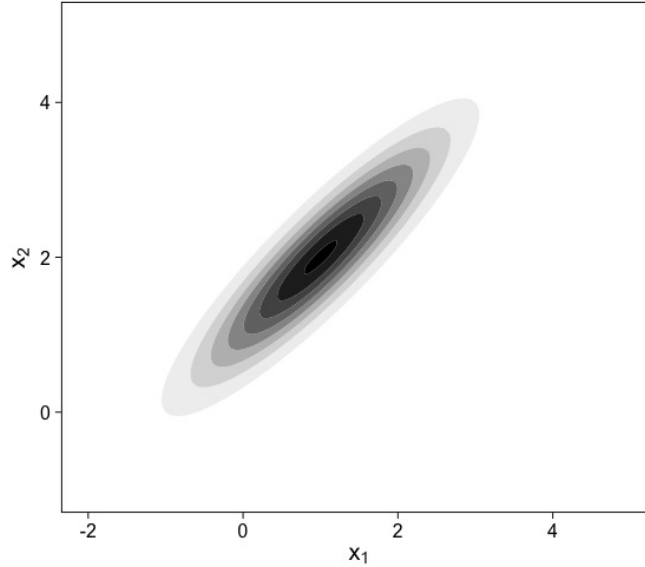


FIGURA 5. Curvas de nivel del modelo Gaussiano.

3.1. ¿Cómo lo logramos?

La idea es la misma que ha funcionado en optimización numérica. Primero, necesitamos **información de gradiente** para mover la pelota en dirección del fondo del *bowl*. Segundo, necesitamos incorporar **información sobre la curvatura** del *bowl*.

Para esto, aumentamos el espacio de variables e incorporamos información de inercia junto con el gradiente.

3.2. Idea general

Extendemos el espacio de variables $\theta \in \mathbb{R}^p$ al sistema en $(\theta, \vartheta) \in \mathbb{R}^p \times \mathbb{R}^p$ por medio de la distribución conjunta

$$\pi(\theta, \vartheta) = \pi(\vartheta|\theta) \cdot \pi(\theta),$$

donde, como antes, $\pi(\theta)$ denota la distribución objetivo.

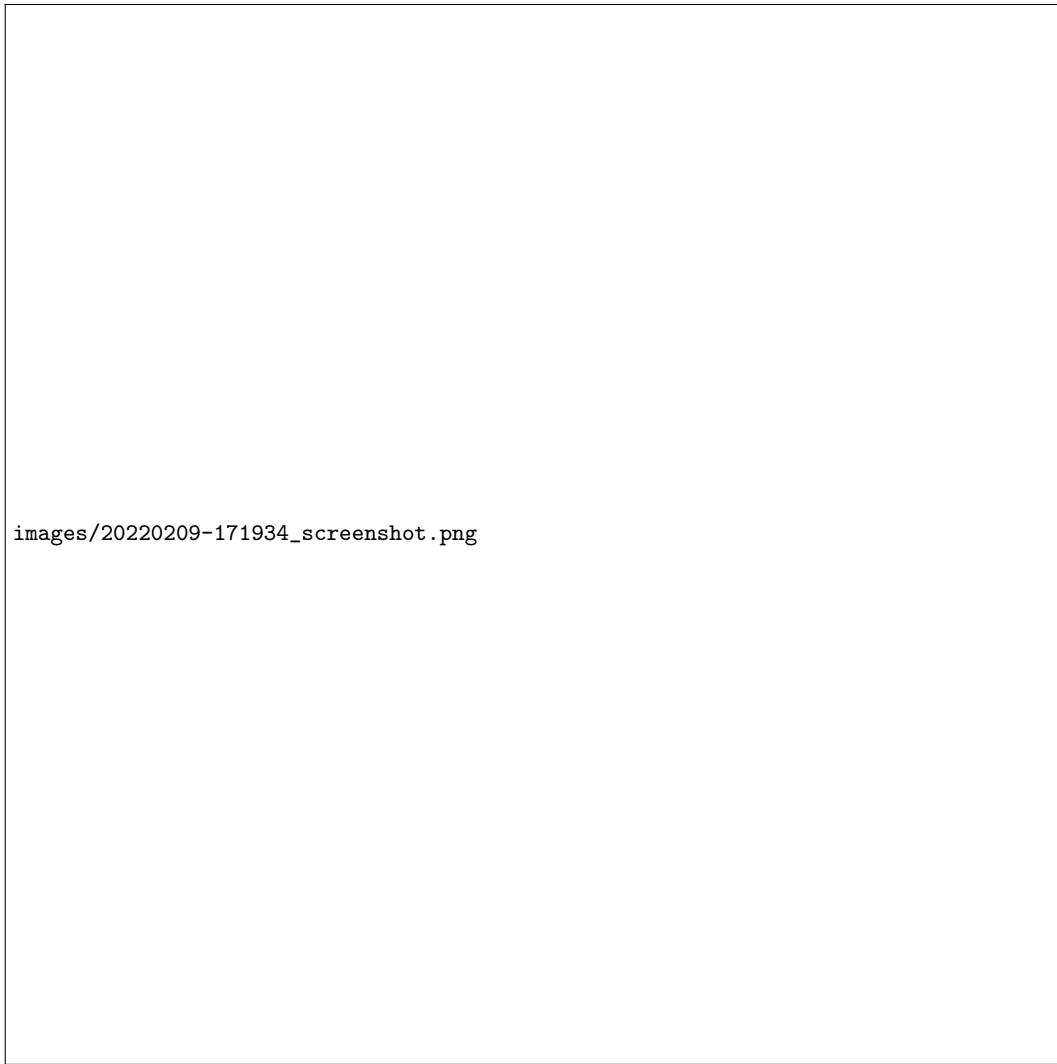
Es usual en mecánica clásica identificar un modelo probabilístico –la densidad $\pi(\cdot)$ – con un potencial de energía –el negativo, $(-1) \times \log(\pi(\cdot)) - [?]$. De esta manera, podemos formular la densidad conjunta en términos del potencial de energía

$$H(\theta, \vartheta) = -\log \pi(\theta, \vartheta).$$

El cual podemos descomponer como

$$H(\theta, \vartheta) = -\log \pi(\vartheta|\theta) - \log \pi(\theta) \quad (9)$$

$$= K(\vartheta, \theta) + V(\theta). \quad (10)$$

FIGURA 6. Tomado de *Towards Data Science*.

En este sistema, el vector θ representa la posición de un objeto y ϑ la inercia que tiene en su movimiento. Las funciones K y V pueden ser interpretadas como las funciones de energía cinética y potencial, respectivamente, del sistema Hamiltoniano.

El sistema descrito arriba se puede simular en tiempo ficticio por medio del sistema de ecuaciones de movimiento, las cuales son:

$$\frac{d\theta}{dt} = \frac{\partial H}{\partial \vartheta}, \quad \frac{d\vartheta}{dt} = -\frac{\partial H}{\partial \theta},$$

lo cual pone en evidencia que es un sistema que **conserva la energía** dentro de la trayectoria.

Esto último es de suma importancia pues quisiéramos que, para un nivel de inercia dado ϑ_* , la trayectoria del sistema (θ, ϑ_*) se mantenga dentro de la curva $H(\theta, \vartheta_*)$.

En la práctica sistema de ecuaciones Hamiltonianas se resuelve en un tiempo discreto ficticio. Estos se llaman integradores simplécticos y tienen la particularidad de aproximar muy bien las trayectorias, incluso en sistemas de dimensiones altas. Puedes consultar [?] para mayores detalles.

Cualquier **patología** que se encuentre en esta simulación determinista puede indicar problemas con el modelo $\pi(\theta)$ en sí (lo cual veremos más adelante).

El punto clave de utilizar el sistema extendido para simulación de cadenas de Markov viene de la siguiente observación. El sistema Hamiltoniano nos permite recuperar realizaciones aleatorias (ya veremos cómo) de

$$\pi(\theta, \vartheta) = \pi(\vartheta|\theta) \cdot \pi(\theta).$$

Ya hemos visto antes que dada una colección de valores aleatorios de una distribución conjunta podemos recuperar la distribución marginal de un componente **descartando** los demás componentes. Esto lo utilizamos en **muestreo por aceptación-rechazo**.

3.3. ¿Cómo incorporamos el componente aleatorio en la simulación?

El proceso estocástico lo construimos como sigue. Consideremos que estamos en el estado θ_n . Incorporamos el movimiento aleatorio en la cadena al **simular** el componente de inercia ϑ_n de la distribución $\pi(\vartheta|\theta)$. Usualmente se considera una variable aleatoria Gaussiana

$$\vartheta_n | \theta_n \sim N(0, M).$$

Una vez que tenemos nuestro estado de inicio, consideramos

$$(\theta(0), \vartheta(0)) = (\theta_n, \vartheta_n),$$

y obtenemos el candidato

$$(\theta(T), \vartheta(T)) = (\theta_*, \vartheta_*), \quad (11)$$

simulando el sistema Hamiltoniano de manera determinista.

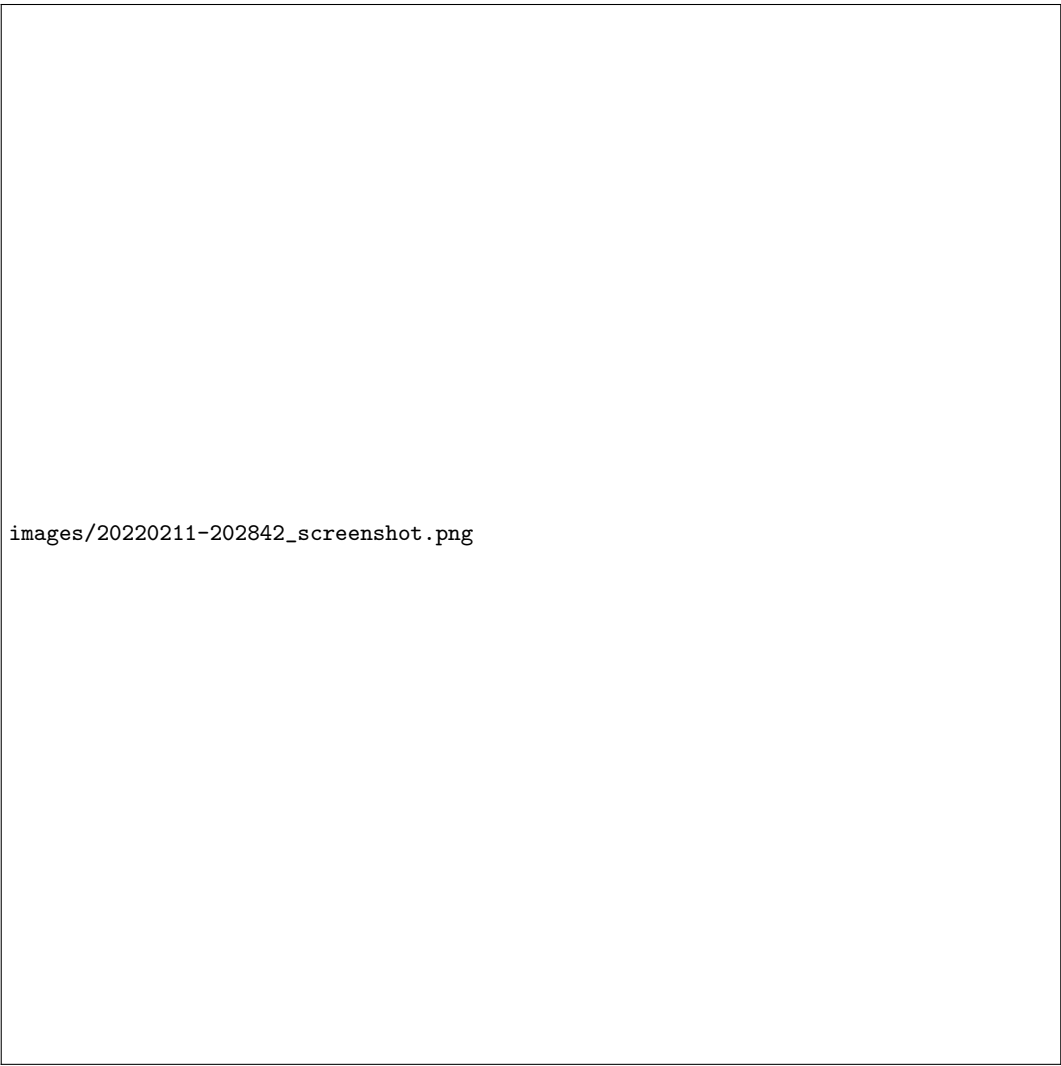
La idea de combinar un proceso aleatorio (simular el componente de inercia) y un proceso determinista (seguir la trayectoria de las ecuaciones de Hamilton) es lo que originalmente motivó [?] a llamarle **Monte Carlo Híbrido**.

La simulación se ve así

4. CONCLUSIONES

HMC es **computacionalmente más costoso** que Metropolis o Gibbs, sin embargo, sus propuestas suelen ser más eficientes, y por consiguiente no necesita un tamaño de muestra tan grandes. En particular cuando se ajustan modelos grandes y complejos (por ejemplo, con variables con correlación alta) HMC supera a otros.

HMC ha sido desarrollado y materializado en **Stan** el cual usa **rutinas automáticas** para determinar la función de energía cinética adecuada y ajusta el tiempo de simulación determinista en cada paso del algoritmo. El método derivado de HMC que se utiliza se conoce como el **No U-Turn Sampler** [? ?].



images/20220211-202842_screenshot.png

FIGURA 7. *Simulación de HMC. Imagen tomada de [?].*

5. EL ESTADO DEL ARTE

El método de Metropolis-Hastings es muy flexible y existe una colección numerable de versiones que pueden ser empleadas en contextos muy particulares. Una buena referencia que incluye métodos de simulación por medio de cadenas de Markov se encuentra en [?], donde incluso se pueden encontrar generalizaciones con **transiciones Markovianas asimétricos** y extensiones a **problemas de dimensión variable**. El libro [?] presenta el estado del arte al 2010.

El cómputo Bayesiano se popularizó con el muestreador de Gibbs. En particular, el avance en teoría de grafos para representar una distribución conjunta como un Grafo Acíclico Dirigido (DAG) que se implementó en software como BUGS o WinBUGS. Pueden consultar el libro de [?] para su explicación.

La desventaja del muestreador de Gibbs es que tiende a ser muy lento en problemas de tamaño grande. Ha habido estrategias que aceleran la simulación aunque al **costo de utilizar aproximaciones**. Estas estrategias han sido materializadas en lenguajes de programación mas generales como Infer.NET.

JAGS (Just Another Gibbs Sampler), es una generalización donde se implementan méto-

dos MCMC para generar simulaciones de distribuciones posteriores. Los paquetes `rjags` y `R2jags` permiten ajustar modelos en JAGS desde R [?]. Es muy fácil utilizar estos programas pues uno simplemente debe especificar las distribuciones iniciales, la verosimilitud y los datos observados. Igual el libro de [?].

Al depender de gradientes para construir propuestas para las cadenas de Markov ha sido natural el desarrollo de herramientas de muestreo basadas en diferenciadores automáticos. Por ejemplo, `Pyro` utiliza `PyTorch`. Tenemos también `Tensorflow Probability` que utiliza `Tensorflow`. `Pymc` (antes `Pymc3`) utiliza Theano (ahora llamado `Aesara`). `NumPyro` utiliza `numpy` y `JAX` como *backend*.

`Pymc` es un muestreador híbrido que permite utilizar Metropolis-Hastings, Gibbs y HMC para la simulación de la posterior [?]. También es mucho más flexible y brinda muestreadores más modernos basados en partículas e información de primer orden (gradientes).

Además, hay herramientas que utilizan las librerías de muestreo para análisis específicos. Por ejemplo, tenemos `cmdstanarm` ajusta **modelos de regresión** utilizando `Stan` como *backend*. La herramienta de Facebook, `Prophet`, utiliza `Stan` (ver [aquí](#)) como *backend* y se especializa en **series de tiempo**. `fenics-pymc3` se especializa en soluciones de **ecuaciones diferenciales** escritas en FEniCS. También tenemos `beat` para **análisis probabilístico de terremotos** y `exoplanet` para series de tiempo en **astronomía**. Por supuesto, no podía faltar una integración `scikit` que se llama `Pymc-Learn`.

Existen otras alternativas para construir cadenas de Markov. Por ejemplo, hay algoritmos que buscan evolucionar una colección de muestras de θ como un enjambre que se comunican entre sí para generar una caminata aleatoria en el espacio del soporte de la distribución. Ejemplos de éstos son el `t-walk` [?] o un ensamble de cadenas linealmente relacionadas como en la herramienta de `emcee` [?],

Finalmente, hay muchos más mecanismos que tienen como objetivo aproximar la distribución posterior. En problemas donde la verosimilitud es **computacionalmente costosa** existen alternativas para crear aproximaciones. El artículo [?] provee de una alternativa utilizando una combinación de técnicas bien establecidas (difusiones Langevin, ensamble de partículas interactivas y filtros de Kalman).