

Descenso de gradiente de Stein ramificado

Arturo Jaramillo Gil

Centro de Investigación en Matemáticas (CIMAT)

Trabajo conjunto con Isaías Banales y Joshué Ricalde-Guerrero

Consideramos una densidad en $\mathbb{R}^d$ dada por

$$\pi(dx) = Z^{-1} \rho(x) dx.$$

Objetivo principal

Consideramos una densidad en $\mathbb{R}^d$ dada por

$$\pi(dx) = Z^{-1} \rho(x) dx.$$

Sólo podemos acceder parcialmente a π , via $\nabla \log \rho(x)$.

Objetivo principal

Consideramos una densidad en $\mathbb{R}^d$ dada por

$$\pi(dx) = Z^{-1}\rho(x)dx.$$

Sólo podemos acceder parcialmente a π , via $\nabla \log \rho(x)$.

Objetivo

Aproximar π numéricamente.

Adicional al problema de dimensionalidad, se presentan los obstáculos

- *Multimodalidad*

Adicional al problema de dimensionalidad, se presentan los obstáculos

- *Multimodalidad*
- *Valles de baja densidad*

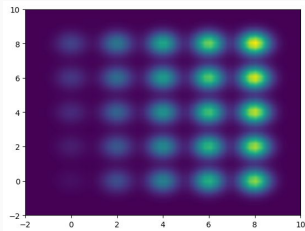
Adicional al problema de dimensionalidad, se presentan los obstáculos

- *Multimodalidad*
- *Valles de baja densidad*

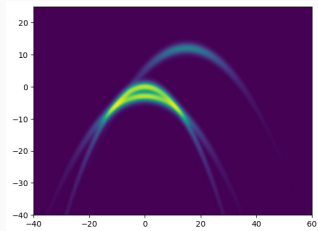
Obstáculos

Adicional al problema de dimensionalidad, se presentan los obstáculos

- *Multimodalidad*
- *Valles de baja densidad*



Mezcla Gaussiana



Mezcla "banana"

Preludio: optimización sobre medidas

En optimización Euclidea movemos puntos $x_t \in \mathbb{R}^d$.

Preludio: optimización sobre medidas

En optimización Euclídeana movemos puntos $x_t \in \mathbb{R}^d$. En optimización de Wasserstein movemos distribuciones $\mu_t \in \mathbb{P}_2(\mathbb{R}^d)$ con densidad f_{μ_t} .

Preludio: optimización sobre medidas

En optimización Euclidea movemos puntos $x_t \in \mathbb{R}^d$. En optimización de Wasserstein movemos distribuciones $\mu_t \in \mathbb{P}_2(\mathbb{R}^d)$ con densidad f_{μ_t} .

Concepto	Sobre $\mathbb{R}^d$	Sobre $\mathcal{P}(\mathbb{R}^d)$
Funcional	$g(x)$, con $x \in \mathbb{R}^d$	$F(\mu)$, con $\mu \in \mathcal{P}(\mathbb{R}^d)$
Dinámica	$\dot{x}_t = -\nabla g(x_t)$	$\partial_t f_{\mu_t} = \nabla \cdot (f_{\mu_t} \nabla \frac{\delta F}{\delta \mu_t})$
Gradiente	∇g	$\nabla \frac{\delta F}{\delta \mu}$
Métrica	$\sum_{i=1}^d v_x^i ^2$	$\int_{\mathbb{R}^d} v_\mu(x) ^2 \mu(dx)$
Dinámica	Flujo gradiente	Flujo gradiente de Wasserstein

Una elección natural de función minimizante es la divergencia de Kullback-Leibler:

$$F(\mu) = KL(\mu||\pi) = \int_{\mathbb{R}^d} \log\left(\frac{f_{\mu}(x)}{f_{\pi}(x)}\right) f_{\mu}(x) dx.$$

Muestreo via optimización

Una elección natural de función minimizante es la divergencia de Kullback-Leibler:

$$F(\mu) = KL(\mu||\pi) = \int_{\mathbb{R}^d} \log\left(\frac{f_{\mu}(x)}{f_{\pi}(x)}\right) f_{\mu}(x) dx.$$

Observaciones

- Minimizar $F(\mu)$ equivale a hacer que μ aproxime π .

Muestreo via optimización

Una elección natural de función minimizante es la divergencia de Kullback-Leibler:

$$F(\mu) = KL(\mu \parallel \pi) = \int_{\mathbb{R}^d} \log\left(\frac{f_\mu(x)}{f_\pi(x)}\right) f_\mu(x) dx.$$

Observaciones

- Minimizar $F(\mu)$ equivale a hacer que μ aproxime π .
- El flujo gradiente de Wasserstein asociado a F está gobernado por

$$\partial_t f_{\mu_t} = \Delta f_{\mu_t} + \nabla \cdot (f_{\mu_t} \nabla V),$$

donde $V(x) = -\log f_\pi(x)$.

Muestreo via optimización

Una elección natural de función minimizante es la divergencia de Kullback-Leibler:

$$F(\mu) = KL(\mu \parallel \pi) = \int_{\mathbb{R}^d} \log \left(\frac{f_\mu(x)}{f_\pi(x)} \right) f_\mu(x) dx.$$

Observaciones

- Minimizar $F(\mu)$ equivale a hacer que μ aproxime π .
- El flujo gradiente de Wasserstein asociado a F está gobernado por

$$\partial_t f_{\mu_t} = \Delta f_{\mu_t} + \nabla \cdot (f_{\mu_t} \nabla V),$$

donde $V(x) = -\log f_\pi(x)$.

- El esquema se basa en que $f_{\mu_t} \rightarrow f_\pi$.

La idea principal es cambiar el flujo de Wasserstein

$$\partial_t f_{\mu_t} = \Delta f_{\mu_t} + \nabla \cdot (f_{\mu_t} \nabla V),$$

con $V(x) = -\log f_{\pi}(x)$, por...

La idea principal es cambiar el flujo de Wasserstein

$$\partial_t f_{\mu_t} = \Delta f_{\mu_t} + \nabla \cdot (f_{\mu_t} \nabla V),$$

con $V(x) = -\log f_\pi(x)$, por...

$$\partial_t f_{\mu_t} = \Delta f_{\mu_t} + \nabla \cdot (f_{\mu_t} K_{\mu_t} \nabla V),$$

con

$$K_\nu[\nabla V](x) := \int_{\mathbb{R}^d} K(x, y) \nabla V(y) \mu(dy),$$

donde K es un kernel positivo definido

Una dinámica para μ_t equivale a las dinámicas para $\langle \mu_t, \varphi \rangle$, con φ una función de prueba.

Flujos kernelizados

Una dinámica para μ_t equivale a las dinámicas para $\langle \mu_t, \varphi \rangle$, con φ una función de prueba.

Versión débil del flujo kernelizado:

$$\partial_t \langle \mu_t, \varphi \rangle = - \langle \mu_t, K_{\mu_t} [\nabla \log f_{\mu_t} - \nabla \log f_{\pi}] \cdot \nabla \varphi \rangle.$$

Flujos kernelizados

Una dinámica para μ_t equivale a las dinámicas para $\langle \mu_t, \varphi \rangle$, con φ una función de prueba.

Versión débil del flujo kernelizado:

$$\partial_t \langle \mu_t, \varphi \rangle = - \langle \mu_t, K_{\mu_t} [\nabla \log f_{\mu_t} - \nabla \log f_{\pi}] \cdot \nabla \varphi \rangle.$$

Si suponemos

$$\mu_t = \frac{1}{\ell} \sum_{j=1}^{\ell} \delta_{x_j(t)},$$

el sistema se “resuelve” si

$$\partial_t x_k(t) = K_{\mu_t} [\nabla \log(f_{\pi})](x_k) + \frac{1}{\ell} \sum_{j=1}^{\ell} \nabla K(x_k, x_j),$$

(i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.

Limitantes:

- (i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.
- (ii) Para cada paso, hacemos

$$\dot{x}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \left[k(x_j, x_i) \nabla \log \rho(x_j) + \nabla_{x_j} k(x_j, x_i) \right], \quad i = 1, \dots, \ell.$$

Limitantes:

Algoritmo SVGD

(i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.

(ii) Para cada paso, hacemos

$$\dot{x}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \left[k(x_j, x_i) \nabla \log \rho(x_j) + \nabla_{x_j} k(x_j, x_i) \right], \quad i = 1, \dots, \ell.$$

(iii) Actualizar $x_i \leftarrow x_i + \varepsilon \dot{x}_i$

Limitantes:

Algoritmo SVGD

(i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.

(ii) Para cada paso, hacemos

$$\dot{x}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \left[k(x_j, x_i) \nabla \log \rho(x_j) + \nabla_{x_j} k(x_j, x_i) \right], \quad i = 1, \dots, \ell.$$

(iii) Actualizar $x_i \leftarrow x_i + \varepsilon \dot{x}_i$

(iv) Detenemos cuando haya bajo desplazamiento entre pasos.

Limitantes:

Algoritmo SVGD

(i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.

(ii) Para cada paso, hacemos

$$\dot{x}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \left[k(x_j, x_i) \nabla \log \rho(x_j) + \nabla_{x_j} k(x_j, x_i) \right], \quad i = 1, \dots, \ell.$$

(iii) Actualizar $x_i \leftarrow x_i + \varepsilon \dot{x}_i$

(iv) Detenemos cuando haya bajo desplazamiento entre pasos.

Limitantes:

- *Colapso por mala inicialización.*

(i) Inicializar partículas $x_1, \dots, x_\ell \sim \mu_0$.

(ii) Para cada paso, hacemos

$$\dot{x}_i = \frac{1}{\ell} \sum_{j=1}^{\ell} \left[k(x_j, x_i) \nabla \log \rho(x_j) + \nabla_{x_j} k(x_j, x_i) \right], \quad i = 1, \dots, \ell.$$

(iii) Actualizar $x_i \leftarrow x_i + \varepsilon \dot{x}_i$

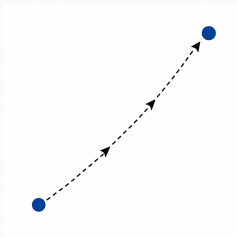
(iv) Detenemos cuando haya bajo desplazamiento entre pasos.

Limitantes:

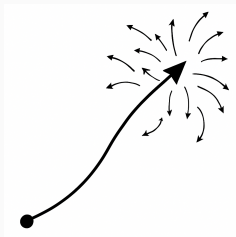
- *Colapso por mala inicialización.*
- *Modas aislados.*

Heurística de la versión ramificada

SVGD



SVGD ramificado



Moraleja

Se intercalan una ramificación aleatoria controlada y un refinamiento determinista del sistema.

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

Arquitectura de partículas ajustada

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

- **E**: Exploradores

Generan descendencia aleatoria en posiciones aleatorias.

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

- **E**: Exploradores

Generan descendencia aleatoria en posiciones aleatorias.

- **O**: Optimizadores

No generan descendencia, sólo ajustan el modelo.

Arquitectura de partículas ajustada

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

- **E**: Exploradores
Generan descendencia aleatoria en posiciones aleatorias.
- **O**: Optimizadores
No generan descendencia, sólo ajustan el modelo.
- **S**: Espina
Garantiza la no extinción del linaje.

Arquitectura de partículas ajustada

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

- **E**: Exploradores
Generan descendencia aleatoria en posiciones aleatorias.
- **O**: Optimizadores
No generan descendencia, sólo ajustan el modelo.
- **S**: Espina
Garantiza la no extinción del linaje.

Arquitectura de partículas ajustada

Reemplazamos las partículas $x \in \mathbb{R}^d$ por partículas coloreadas (x, c) con c un color en los símbolos E, O, S .

- **E**: Exploradores
Generan descendencia aleatoria en posiciones aleatorias.
- **O**: Optimizadores
No generan descendencia, sólo ajustan el modelo.
- **S**: Espina
Garantiza la no extinción del linaje.

El propósito es generar exploración aleatoria de la arquitectura.

Reglas de ramificación

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción
 $\gamma_i \sim q_E, q_O, q_S$.

Reglas de ramificación

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción
 $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.

Reglas de ramificación

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción

$$\gamma_i \sim q_E, q_O, q_S.$$

- Pedimos q_O degenerada.
- Pedimos q_S con masa ero en cero.

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción

$$\gamma_i \sim q_E, q_O, q_S.$$

- Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:
 - El único hijo de un optimizador es la misma del de su padre

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:
 - El único hijo de un optimizador es la misma del de su padre
 - Los hijos de x_i que es explorador o espina se posicionan de acuerdo a un kernel $Q(\cdot|x_i)$

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción
 $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:
 - El único hijo de un optimizador es la misma del de su padre
 - Los hijos de x_i que es explorador o espina se posicionan de acuerdo a un kernel $Q(\cdot|x_i)$
- Recoloreo

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:
 - El único hijo de un optimizador es la misma del de su padre
 - Los hijos de x_i que es explorador o espina se posicionan de acuerdo a un kernel $Q(\cdot|x_i)$
- Recoloreo
 - Paso I: los hijos de O se colorean O , y los de E y S se colorean E .

Reglas de ramificación

Iteramos algoritmo SVGD con algoritmo de ramificación

- Reproducción: cada color tiene una ley de reproducción
 $\gamma_i \sim q_E, q_O, q_S$.
 - Pedimos q_O degenerada.
 - Pedimos q_S con masa ero en cero.
- Posición: Dependiendo de la descendencia, la ubicación de los hijos es como sigue:
 - El único hijo de un optimizador es la misma del de su padre
 - Los hijos de x_i que es explorador o espina se posicionan de acuerdo a un kernel $Q(\cdot|x_i)$
- Recoloreo
 - Paso I: los hijos de O se colorean O , y los de E y S se colorean E .
 - Paso II: Mostramos una particular aleatoria y la coloreamos S .

- *Mejora*: correr SVGD hasta estabilización.

- *Mejora*: correr SVGD hasta estabilización.
- *Ramificación*: reproducimos las partículas de acuerdo a su color y posicionamos su descendencia de acuerdo a $P(\cdot|x)$.

- *Mejora*: correr SVGD hasta estabilización.
- *Ramificación*: reproducimos las partículas de acuerdo a su color y posicionamos su descendencia de acuerdo a $P(\cdot|x)$.
- *Recoloreo*: partículas optimizadoras o exploradoras dan hijos con el mismo color. El hijo de la espina se colorea E.

- *Mejora*: correr SVGD hasta estabilización.
- *Ramificación*: reproducimos las partículas de acuerdo a su color y posicionamos su descendencia de acuerdo a $P(\cdot|x)$.
- *Recoloreo*: partículas optimizadoras o exploradoras dan hijos con el mismo color. El hijo de la espina se colorea E.
- *Selección de la espina*.

- *Mejora*: correr SVGD hasta estabilización.
- *Ramificación*: reproducimos las partículas de acuerdo a su color y posicionamos su descendencia de acuerdo a $P(\cdot|x)$.
- *Recoloreo*: partículas optimizadoras o exploradoras dan hijos con el mismo color. El hijo de la espina se colorea E.
- *Selección de la espina*.
- Actualizar μ_{n+1} .

- *Mejora*: correr SVGD hasta estabilización.
- *Ramificación*: reproducimos las partículas de acuerdo a su color y posicionamos su descendencia de acuerdo a $P(\cdot|x)$.
- *Recoloreo*: partículas optimizadoras o exploradoras dan hijos con el mismo color. El hijo de la espina se colorea E.
- *Selección de la espina*.
- Actualizar μ_{n+1} .
- Repetir dependiendo del presupuesto disponible.

Theorem

Sean $AC_r(\mathbb{R}^d)$ las medidas absolutamente continuas con densidad acotada por r . Si se cumple

$$d_W(\mu_n, AC_r(\mathbb{R}^d)) \rightarrow 0, \tag{1}$$

entonces μ_n converge en ley a π .

Theorem

Sean $AC_r(\mathbb{R}^d)$ las medidas absolutamente continuas con densidad acotada por r . Si se cumple

$$d_W(\mu_n, AC_r(\mathbb{R}^d)) \rightarrow 0, \quad (1)$$

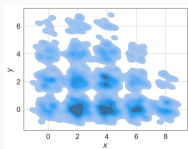
entonces μ_n converge en ley a π .

Observación

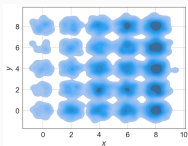
Heurísticamente, la condición (1) se lee del histograma.

Comparación con SVGD en el mismo presupuesto de tiempo

Mezcla Gaussiana

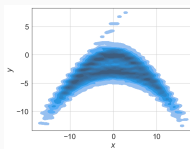


SVGD

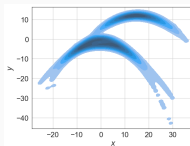


BSVG

Mezcla Banana



SVGD

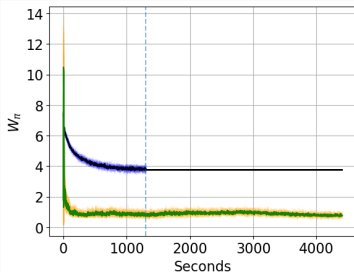


BSVG

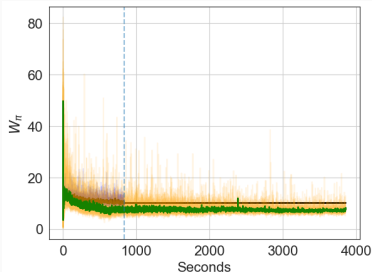
Tiempo de máquina

Distancia de Wasserstein empírica en eje y vs tiempo en eje x

Mezcla Gaussiana



Mezcla Banana



Oservemos que el BSVGd tardará mucho más en estabilizarse, pero en el mismo presupuesto de tiempo, alcanza mejores resultados

- Garantías de convergencia más robustas

Conclusiones y problemas abiertos

- Garantías de convergencia más robustas
- Tasas de convergencia.

Conclusiones y problemas abiertos

- Garantías de convergencia más robustas
- Tasas de convergencia.
- Todos los elementos deberían poder hacerse adaptativos: reproducciones Q , posiciones $P(\cdot|x)$ y coloreos.

Conclusiones y problemas abiertos

- Garantías de convergencia más robustas
- Tasas de convergencia.
- Todos los elementos deberían poder hacerse adaptativos: reproducciones Q , posiciones $P(\cdot|x)$ y coloreos.
- Ajustes en geometrías distintas a Wasserstein (el caso Fisher-Rao se encuentra en proceso).

Gracias!

Arturo Jaramillo Gil

Centro de Investigación en Matemáticas (CIMAT)

`jagil@cimat.mx`



Q. Liu y D. Wang.

Stein Variational Gradient Descent: A general purpose Bayesian inference algorithm.

Advances in Neural Information Processing Systems, 2016.



Q. Liu.

Stein Variational Gradient Descent as Gradient Flow.

preprint, 2017.



I. Banales, A. Jaramillo y J. Ricalde-Guerrero.

Branching Stein Variational Gradient Descent for Sampling Multimodal Distributions.

Preprint, 2025.