Compressive Sensing

Pedro Raigorodsky

2025

Vectores esparsos. 'Norma' cero.

En esta charla vamos a enfocarnos en encontrar soluciones esparsas a problemas (esparsas = ralas = 'muchos ceros').

Vectores esparsos. 'Norma' cero.

En esta charla vamos a enfocarnos en encontrar soluciones esparsas a problemas (esparsas = ralas = 'muchos ceros').Definimos entonces la siguiente cantidad

$$||x||_0 = \#\{i : x_i \neq 0\}$$

es decir, la cantidad de entradas no nulas de un vector. (Ojo, no es una norma!)

Vectores esparsos. 'Norma' cero.

En esta charla vamos a enfocarnos en encontrar soluciones esparsas a problemas (esparsas = ralas = 'muchos ceros').Definimos entonces la siguiente cantidad

$$||x||_0 = \#\{i : x_i \neq 0\}$$

es decir, la cantidad de entradas no nulas de un vector. (Ojo, no es una norma!)

¿Qué es multiplicar un vector esparso por una matriz?

Vectores esparsos. 'Norma' cero.

En esta charla vamos a enfocarnos en encontrar soluciones esparsas a problemas (esparsas = ralas = 'muchos ceros').Definimos entonces la siguiente cantidad

$$||x||_0 = \#\{i : x_i \neq 0\}$$

es decir, la cantidad de entradas no nulas de un vector. (Ojo, no es una norma!)

¿Qué es multiplicar un vector esparso por una matriz? Si $x \in \mathbb{R}^n$ es esparso,

$$Ax = \sum_{i=1}^{n} x_i A_i$$

donde A_i es la columna i-ésima de A.

Vectores esparsos. 'Norma' cero.

En esta charla vamos a enfocarnos en encontrar soluciones esparsas a problemas (esparsas = ralas = 'muchos ceros').Definimos entonces la siguiente cantidad

$$||x||_0 = \#\{i : x_i \neq 0\}$$

es decir, la cantidad de entradas no nulas de un vector. (Ojo, no es una norma!)

¿Qué es multiplicar un vector esparso por una matriz? Si $x \in \mathbb{R}^n$ es esparso,

$$Ax = \sum_{i=1}^{n} x_i A_i$$

donde A_i es la columna i-ésima de A.Si x es esparso, la mayoría de estos sumandos son cero. Luego el resultado es **una combinación lineal de pocas columnas de A.**

El problema de compressive sensing es muy simple: dado un sistema lineal $Ax = b$, encontrar la solución más esparsa.

El problema de compressive sensing es muy simple: dado un sistema lineal $Ax = b$, encontrar la solución más esparsa.Es decir,

Problema de Compressive Sensing

$$P_0 : \min ||x||_0 \quad : Ax = b$$

donde $A \in \mathbb{R}^{m \times n}$ con $n > m$. Parece fácil, pero...

El problema de compressive sensing es muy simple: dado un sistema lineal $Ax = b$, encontrar la solución más esparsa.Es decir,

Problema de Compressive Sensing

$$P_0 : \min ||x||_0 \quad : Ax = b$$

donde $A \in \mathbb{R}^{m \times n}$ con $n > m$. Parece fácil, pero...

Inconveniente

Este problema es NP-Completo.

Compressive Sensing

El problema de compressive sensing es muy simple: dado un sistema lineal $Ax = b$, encontrar la solución más esparsa.Es decir,

Problema de Compressive Sensing

$$P_0 : \min ||x||_0 \quad : Ax = b$$

donde $A \in \mathbb{R}^{m \times n}$ con $n > m$. Parece fácil, pero...

Inconveniente

Este problema es NP-Completo.

En general este problema es díficl de resolver computacionalmente, pero veremos que en muchos contextos sí se puede resolver. Aplicaciones: muchísimas (procesamiento de señales, compresión, procesamiento de imágenes).

Interpretación con Subespacios

Otra manera de pensar el problema es el siguiente: si $Ax = b$ entonces b se escribe como una combinación lineal de ciertas columnas de A.

Interpretación con Subespacios

Otra manera de pensar el problema es el siguiente: si $Ax = b$ entonces b se escribe como una combinación lineal de ciertas columnas de A.

CS - Versión subespacios

Dado $b \in \mathbb{R}^m$ y $\mathcal{F} = \{A_1, \ldots, A_n\} \subseteq \mathbb{R}^m$, ¿cuál es la familia más chica $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{F}$ tal que $b \in \text{gen}(A_{i_1}, \ldots, A_{i_k})$?

Interpretación con Subespacios

Otra manera de pensar el problema es el siguiente: si $Ax = b$ entonces b se escribe como una combinación lineal de ciertas columnas de A.

CS - Versión subespacios

Dado $b \in \mathbb{R}^m$ y $\mathcal{F} = \{A_1, \ldots, A_n\} \subseteq \mathbb{R}^m$, ¿cuál es la familia más chica $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{F}$ tal que $b \in \text{gen}(A_{i_1}, \ldots, A_{i_k})$?

Una búsqueda naive requiere estudiar $\binom{n}{k}$ suboncjuntos

Interpretación con Subespacios

Otra manera de pensar el problema es el siguiente: si $Ax = b$ entonces b se escribe como una combinación lineal de ciertas columnas de A.

CS - Versión subespacios

Dado $b \in \mathbb{R}^m$ y $\mathcal{F} = \{A_1, \dots, A_n\} \subseteq \mathbb{R}^m$, ¿cuál es la familia más chica $\{A_{i_1}, \dots, A_{i_k}\} \subseteq \mathcal{F}$ tal que $b \in \mathrm{gen}(A_{i_1}, \dots, A_{i_k})$?

Una búsqueda naive requiere estudiar $\binom{n}{k}$ suboncjuntos, que crece como n^k. En general k crece (y de hecho, es común que $k \sim cm$ por lo que la complejidad es de orden $n^{c \cdot m}$, ¡demasiado!)

Spark de una Matriz

Dada una matriz $A \in \mathbb{R}^{m \times n}$ con $m < n$, definimos su **sparse rank** o **spark** como la menor cantidad de columnas de A que forman un conjunto linealmente dependiente.

Spark de una Matriz

Dada una matriz $A \in \mathbb{R}^{m \times n}$ con $m < n$, definimos su **sparse rank** o **spark** como la menor cantidad de columnas de A que forman un conjunto linealmente dependiente.

Inconveniente

Dada una matriz, determinar su spark es NP completo.

Spark de una Matriz

Dada una matriz $A \in \mathbb{R}^{m \times n}$ con $m < n$, definimos su **sparse rank** o **spark** como la menor cantidad de columnas de A que forman un conjunto linealmente dependiente.

Inconveniente

Dada una matriz, determinar su spark es NP completo.

Notar que el spark es exactamente el problema de compressive sensing pero con $b = 0$ (donde exlcuimos la solución trivial)

Spark de una Matriz

Dada una matriz $A \in \mathbb{R}^{m \times n}$ con $m < n$, definimos su **sparse rank** o **spark** como la menor cantidad de columnas de A que forman un conjunto linealmente dependiente.

Inconveniente

Dada una matriz, determinar su spark es NP completo.

Notar que el spark es exactamente el problema de compressive sensing pero con $b = 0$ (donde exlcuimos la solución trivial)

Teorema - Unicidad de Esparsidad Máxima

Sea $A \in \mathbb{R}^{m \times n}$ y sea x una solución con $Ax = b$ tal que $||x||_0 < \frac{1}{2}\text{spark}(A)$. Entonces x es la única solución más esparsa de $Ax = b$.

Spark de una Matriz

Dada una matriz $A \in \mathbb{R}^{m \times n}$ con $m < n$, definimos su **sparse rank** o **spark** como la menor cantidad de columnas de A que forman un conjunto linealmente dependiente.

Inconveniente

Dada una matriz, determinar su spark es NP completo.

Notar que el spark es exactamente el problema de compressive sensing pero con $b = 0$ (donde exlcuimos la solución trivial)

Teorema - Unicidad de Esparsidad Máxima

Sea $A \in \mathbb{R}^{m \times n}$ y sea x una solución con $Ax = b$ tal que $||x||_0 < \frac{1}{2}\text{spark}(A)$. Entonces x es la única solución más esparsa de $Ax = b$.

Ok, pero incluso si el problema está bien definido, ¿cómo lo resolvemos?

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Iteramos sobre k hasta tener que $||r_k|| < \delta$:

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Iteramos sobre k hasta tener que $||r_k|| < \delta$:

1. **Sweep:** Buscamos la columna más cercana al resudio (en ángulo). La guardamos j_0.

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Iteramos sobre k hasta tener que $||r_k|| < \delta$:

1. **Sweep:** Buscamos la columna más cercana al resudio (en ángulo). La guardamos j_0.
2. **Update Support:** Usamos esta coordenada para definir el nuevo soporte, $S^k = S^{k-1} \cup \{j_0\}$.

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Iteramos sobre k hasta tener que $||r_k|| < \delta$:

1. **Sweep:** Buscamos la columna más cercana al resudio (en ángulo). La guardamos j_0.
2. **Update Support:** Usamos esta coordenada para definir el nuevo soporte, $S^k = S^{k-1} \cup \{j_0\}$.
3. **Update Provisional Solution:** Resolvemos $x_k = \min ||Ax - b||_2$ con $\text{Sup}(x) \subseteq S^k$

Idea 1: Algoritmo Greedy

Resolver CS se reduce a encontrar el soporte (las columnas de A) que realmente necesito. Un enofque greedy lo que hace es agregar a cada paso 'la mejor columna' que puede.

OMP

Inicializamos $x_0 = 0$, un soporte $S_0 = \emptyset$ y el residuo $r_0 = Ax_0 - b = b$.

Iteramos sobre k hasta tener que $||r_k|| < \delta$:

1. **Sweep:** Buscamos la columna más cercana al resudio (en ángulo). La guardamos j_0.
2. **Update Support:** Usamos esta coordenada para definir el nuevo soporte, $S^k = S^{k-1} \cup \{j_0\}$.
3. **Update Provisional Solution:** Resolvemos $x_k = \min ||Ax - b||_2$ con $\text{Sup}(x) \subseteq S^k$
4. **Compute Residue:** $r_k = Ax_k - b$.

Idea 2: Relajación ℓ_1.

Motivación: Es cierto que $||x||_0 = \lim_{p \to 0} ||x||_p$.

Idea 2: Relajación ℓ_1.

Motivación: Es cierto que $||x||_0 = \lim_{p \to 0} ||x||_p$.

Problema en ℓ_1

Sea $A \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^n$. Definimos el problema ℓ_1 asociado cómo:

$$P_1 : \min ||Wx||_1 \text{ con } Ax = b$$

donde W es una matriz cuadrada de pesos. Si $W = I$, lo denominamos Basis Pursuit (BP).

Idea 2: Relajación ℓ_1.

Motivación: Es cierto que $||x||_0 = \lim_{p \to 0} ||x||_p$.

Problema en ℓ_1

Sea $A \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^n$. Definimos el problema ℓ_1 asociado cómo:

$$P_1 : \min ||Wx||_1 \text{ con } Ax = b$$

donde W es una matriz cuadrada de pesos. Si $W = I$, lo denominamos Basis Pursuit (BP).

Mediante multiplicadores de Lagrange, se puede demostrar que para cierta elección del parámetro λ, el problema es equivalente a

$$P_1' : \min \lambda ||Wx||_1 + \frac{1}{2}||Ax - y||_2^2$$

Y este se puede resolver, por ejemplo, con descenso gradiente.

Algunos Problemas

Tanto Greedy como algoritmos para resolver el problema en ℓ_1 (por ej, descenso gardiente) son muy eficientes en la práctica, pero no siempren resuelven el problema original. La pregunta luego es: **¿bajo que condiciones estos algoritmos encuentran la respuesta a P?**.

Tanto Greedy como algoritmos para resolver el problema en ℓ_1 (por ej, descenso gardiente) son muy eficientes en la práctica, pero no siempren resuelven el problema original. La pregunta luego es: **¿bajo que condiciones estos algoritmos encuentran la respuesta a P?**.

En general, para que haya recuperación, el problema tiene que estar 'bien condicionado'. En general, si varios vectores estan cerca de ser colineales o degenerados, los algoritmos pueden llegar a fallar, ya que es 'díficil' escoger qué columnas usar.

Algunos Problemas

Tanto Greedy como algoritmos para resolver el problema en ℓ_1 (por ej, descenso gardiente) son muy eficientes en la práctica, pero no siempren resuelven el problema original. La pregunta luego es: **¿bajo que condiciones estos algoritmos encuentran la respuesta a P?**.

En general, para que haya recuperación, el problema tiene que estar 'bien condicionado'. En general, si varios vectores estan cerca de ser colineales o degenerados, los algoritmos pueden llegar a fallar, ya que es 'díficil' escoger qué columnas usar.

Rule of Thumb

Las cosas funcionan bien si A es 'similar a ortogonal'.

Mutual Coherence

Sea $A \in \mathbb{R}^{m\times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i\neq j} \frac{|C_i(A)C_j(A)|}{||C_i(A)||_2||C_j(A)||_2}$$

Mutual Coherence

Sea $A \in \mathbb{R}^{m \times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i \neq j} \frac{|C_i(A) C_j(A)|}{||C_i(A)||_2 ||C_j(A)||_2}$$

Alternativamente, si $\overline{A}$ proviene de normalizar las columnas de A

$$\mu(A) = ||\overline{A}^t \cdot \overline{A} - I_{n \times n}||_\infty$$

Mutual Coherence

Sea $A \in \mathbb{R}^{m \times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i \neq j} \frac{|C_i(A) C_j(A)|}{||C_i(A)||_2 ||C_j(A)||_2}$$

Alternativamente, si $\overline{A}$ proviene de normalizar las columnas de A

$$\mu(A) = ||\overline{A}^t \cdot \overline{A} - I_{n \times n}||_\infty$$

- $\mu(\cdot)$ es continua con respecto a A.

Mutual Coherence

Sea $A \in \mathbb{R}^{m\times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i\neq j} \frac{|C_i(A)C_j(A)|}{||C_i(A)||_2||C_j(A)||_2}$$

Alternativamente, si $\overline{A}$ proviene de normalizar las columnas de A

$$\mu(A) = ||\overline{A}^t \cdot \overline{A} - I_{n\times n}||_\infty$$

- $\mu(\cdot)$ es continua con respecto a A.
- $\mu(A)$ se calcula en $O(n^2)$.

Mutual Coherence

Sea $A \in \mathbb{R}^{m \times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i \neq j} \frac{|C_i(A) C_j(A)|}{||C_i(A)||_2 ||C_j(A)||_2}$$

Alternativamente, si $\overline{A}$ proviene de normalizar las columnas de A

$$\mu(A) = ||\overline{A}^t \cdot \overline{A} - I_{n \times n}||_\infty$$

- $\mu(\cdot)$ es continua con respecto a A.
- $\mu(A)$ se calcula en $O(n^2)$.
- $\mu(A) = 0$ si y solo si A es ortogonal. En general queremos que $\mu(A)$ sea bajo, pero si $n > m$, claramente $\mu(A) \neq 0$.

Mutual Coherence

Sea $A \in \mathbb{R}^{m \times n}$ sin columnas nulas. La **mutual coherence de** A es

$$\mu(A) = \max_{i \neq j} \frac{|C_i(A) C_j(A)|}{||C_i(A)||_2 ||C_j(A)||_2}$$

Alternativamente, si $\overline{A}$ proviene de normalizar las columnas de A

$$\mu(A) = ||\overline{A}^t \cdot \overline{A} - I_{n \times n}||_\infty$$

- $\mu(\cdot)$ es continua con respecto a A.
- $\mu(A)$ se calcula en $O(n^2)$.
- $\mu(A) = 0$ si y solo si A es ortogonal. En general queremos que $\mu(A)$ sea bajo, pero si $n > m$, claramente $\mu(A) \neq 0$.
- $\mu(A) = 1$ si y solo si A tiene dos columnas que son múltiplos. Este sería el caso degenerado.

Mutual Coherence

Si X es una matriz que proviene de tomar k columnas de A, notar que X tiene rango máximo si y solo si X^tX es inversible.

Mutual Coherence

Si X es una matriz que proviene de tomar k columnas de A, notar que X tiene rango máximo si y solo si X^tX es inversible.

Corolario de Disco de Gershwin: Si B es una matriz cuadrada con diagonal dominante (o sea $|B_{ii}| > \sum_{j \neq i} |B_{ij}|$) entonces es inversible.

Mutual Coherence

Si X es una matriz que proviene de tomar k columnas de A, notar que X tiene rango máximo si y solo si X^tX es inversible.

Corolario de Disco de Gershwin: Si B es una matriz cuadrada con diagonal dominante (o sea $|B_{ii}| > \sum_{j\neq i} |B_{ij}|$) entonces es inversible.

Notar que, si $k < 1 + \dfrac{1}{\mu(A)}$, luego X^tX es diagonal dominante.

Mutual Coherence

Si X es una matriz que proviene de tomar k columnas de A, notar que X tiene rango máximo si y solo si X^tX es inversible.

Corolario de Disco de Gershwin: Si B es una matriz cuadrada con diagonal dominante (o sea $|B_{ii}| > \sum_{j\neq i} |B_{ij}|$) entonces es inversible.

Notar que, si $k < 1 + \dfrac{1}{\mu(A)}$, luego X^tX es diagonal dominante.Por ende tenemos el siguiente resultado escencial:

Coherencia Mutua y Spark

Si $A \in \mathbb{R}^{m\times n}$ entonces

$$\text{spark}(A) \geq 1 + \frac{1}{\mu(A)}$$

Mutual Coherence y relajaciones del problema

Mutual Coherence y Reconstrucción de la Solución

Sea $A \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^n$. Si $||x||_0 < \dfrac{1}{2}(1 + \frac{1}{\mu(A)})$. Entonces, x es la solución más esparsa. ¡Además el algoritmo greedy y la relajación ℓ_1 encuentran x correctamente!

Mutual Coherence y relajaciones del problema

Mutual Coherence y Reconstrucción de la Solución

Sea $A \in \mathbb{R}^{m \times n}$ y $b \in \mathbb{R}^n$. Si $||x||_0 < \dfrac{1}{2}(1 + \frac{1}{\mu(A)})$. Entonces, x es la solución más esparsa. ¡Además el algoritmo greedy y la relajación ℓ_1 encuentran x correctamente!

No lo vamos a demostrar (está en las notas de Donoho, prueba independientemente que ambos métodos encuentran la solución de Compressive Sensing), pero nos dice lo siguiente:

- Cuánto más baja la mutual coherence, los algoritmos encuentran soluciones forzando menos esparcidad.

Cota de Welch

La cota $\text{spark}(A) < 1 + \dfrac{1}{\mu(A)}$ es buena considerando que podemos asegurar unicidad, pero tenemos el siguiente resultado:

Cota de Welch

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

Cota de Welch

La cota $\text{spark}(A) < 1 + \dfrac{1}{\mu(A)}$ es buena considerando que podemos asegurar unicidad, pero tenemos el siguiente resultado:

Cota de Welch

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

Demostración. Spdg asumimos que las columnas están normalizadas.

Cota de Welch

La cota $\operatorname{spark}(A) < 1 + \dfrac{1}{\mu(A)}$ es buena considerando que podemos asegurar unicidad, pero tenemos el siguiente resultado:

Cota de Welch

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

Demostración. Spdg asumimos que las columnas están normalizadas. Consideramos $G = A^t A$ la matriz de productos internos de las columnas.

Cota de Welch

La cota $\text{spark}(A) < 1 + \dfrac{1}{\mu(A)}$ es buena considerando que podemos asegurar unicidad, pero tenemos el siguiente resultado:

Cota de Welch

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

Demostración. Spdg asumimos que las columnas están normalizadas. Consideramos $G = A^t A$ la matriz de productos internos de las columnas.Como G es definida positiva, tiene autovalores no nulos $\lambda_1 > \lambda_2 \ldots > \lambda_r > 0$.

Cota de Welch

La cota $\text{spark}(A) < 1 + \frac{1}{\mu(A)}$ es buena considerando que podemos asegurar unicidad, pero tenemos el siguiente resultado:

Cota de Welch

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

Demostración. Spdg asumimos que las columnas están normalizadas. Consideramos $G = A^t A$ la matriz de productos internos de las columnas.Como G es definida positiva, tiene autovalores no nulos $\lambda_1 > \lambda_2 \ldots > \lambda_r > 0$.Ahora, recordamos que la traza verifica que

$$n^2 = Tr(G)^2 = \left(\sum_{i=1}^{r} \lambda_i\right)^2 \leq r \sum_{i=1}^{r} \lambda_i^2 \leq m \sum_{i=1}^{r} \lambda_i^2$$

Cota de Welch

Pero la norma de Frobenius verfica que

$$||G||_F^2 = \sum_{i,j} | \langle A_i, A_j \rangle |^2 = \sum_{i=1}^{r} \lambda_i^2$$

Cota de Welch

Pero la norma de Frobenius verfica que

$$||G||_F^2 = \sum_{i,j} |\langle A_i, A_j \rangle|^2 = \sum_{i=1}^{r} \lambda_i^2$$

Luego juntando todo

$$\frac{n^2}{m} \leq n + \sum_{i \neq j} |\langle A_i, A_j \rangle|^2$$

Cota de Welch

Pero la norma de Frobenius verfica que

$$||G||_F^2 = \sum_{i,j} |\langle A_i, A_j \rangle|^2 = \sum_{i=1}^{r} \lambda_i^2$$

Luego juntando todo

$$\frac{n^2}{m} \leq n + \sum_{i \neq j} |\langle A_i, A_j \rangle|^2$$

Ahora, la idea es usar que $\mu(A)^2$ es el mayor de los cuadrados de los productos internos, por lo que que seguro que es más grande que el promedio, o sea

$$\mu(A)^2 \geq \frac{1}{n(n-1)} \sum_{i \neq j} |\langle A_i, A_j \rangle|^2 \geq \frac{n-m}{m(n-1)}$$

¿Qué nos dice la cota de Welch?

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

La cota de Welch es un límite de los argumentos de mutual coherence, que no pueden ser 'demasiado buenos'.

¿Qué nos dice la cota de Welch?

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

La cota de Welch es un límite de los argumentos de mutual coherence, que no pueden ser 'demasiado buenos'.

- Si $n = cm$ (caso estándar), asíntoticamente no sdice que $\mu(A) \geq \dfrac{1}{\sqrt{m}}$ por lo que la demostración anterior nos dice $k < \frac{1}{2}\sqrt{m}$.

¿Qué nos dice la cota de Welch?

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

La cota de Welch es un límite de los argumentos de mutual coherence, que no pueden ser 'demasiado buenos'.

- Si $n = cm$ (caso estándar), asíntoticamente no sdice que $\mu(A) \geq \dfrac{1}{\sqrt{m}}$ por lo que la demostración anterior nos dice $k < \frac{1}{2}\sqrt{m}$.

Moraleja: en general los argumentos de mutual coherence permiten garantizar correctitud solo para problemas con esparsidad $k \sim \sqrt{m}$ ('coherence bound').

¿Qué nos dice la cota de Welch?

$$\mu(A) \geq \sqrt{\frac{n-m}{n(m-1)}}$$

La cota de Welch es un límite de los argumentos de mutual coherence, que no pueden ser 'demasiado buenos'.

- Si $n = cm$ (caso estándar), asíntoticamente no sdice que $\mu(A) \geq \dfrac{1}{\sqrt{m}}$ por lo que la demostración anterior nos dice $k < \frac{1}{2}\sqrt{m}$.

Moraleja: en general los argumentos de mutual coherence permiten garantizar correctitud solo para problemas con esparsidad $k \sim \sqrt{m}$ ('coherence bound').

Restricted Isometric Property

Restricted Isometric Property

En 2006, Tao y Candes introducen la conocida noción de **Restriced Isometry Property** (o RIP) de la siguiente manera: decimos que A es (δ, k)-RIP si para cualquier conjunto de k columnas X y $c \in \mathbb{R}^k$:

$$(1-\delta)||c||_2 \leq ||X(c)||_2 \leq (1+\delta)||c||_2$$

Restricted Isometric Property

En 2006, Tao y Candes introducen la conocida noción de **Restriced Isometry Property** (o RIP) de la siguiente manera: decimos que A es (δ, k)-RIP si para cualquier conjunto de k columnas X y $c \in \mathbb{R}^k$:

$$(1-\delta)||c||_2 \leq ||X(c)||_2 \leq (1+\delta)||c||_2$$

Tambien introducen la noción de las constantes de ortogonalidad $\theta_{k,k'}$ de modo que

$$|\left\langle X(c), X'(c') \right\rangle| \leq \theta_{k,k'}||c||||c'||$$

donde X y X' son conjuntos de k y k' columnas.

Restricted Isometric Property

En 2006, Tao y Candes introducen la conocida noción de **Restriced Isometry Property** (o RIP) de la siguiente manera: decimos que A es (δ, k)-RIP si para cualquier conjunto de k columnas X y $c \in \mathbb{R}^k$:

$$(1-\delta)||c||_2 \leq ||X(c)||_2 \leq (1+\delta)||c||_2$$

Tambien introducen la noción de las constantes de ortogonalidad $\theta_{k,k'}$ de modo que

$$|\left\langle X(c), X'(c')\right\rangle| \leq \theta_{k,k'}||c||||c'||$$

donde X y X' son conjuntos de k y k' columnas.

Es claro que si A es (δ, k)-RIP, entonces su spark es como mínimo k.

RIP y Basis Pursuit

RIP y recuperación vía BP

Sea A un (δ_k, k)-RIP y supongamos que:

$$\delta_k + \theta_{k,2k} + \theta_k < 1$$

Entonces BP recupera las soluciones de esparcidad k.

RIP y Basis Pursuit

RIP y recuperación vía BP

Sea A un (δ_k, k)-RIP y supongamos que:

$$\delta_k + \theta_{k,2k} + \theta_k < 1$$

Entonces BP recupera las soluciones de esparcidad k.

Usando un lema sencillo que dice $\theta_{k,k'} < \delta_{k+k'} < \theta_{k,k'} + \max\{\delta_k, \delta_{k'}\}$ (donde δ_k son valores que hacen que A sea k-RIP), nuestra condición es implicada por $\delta_k + \delta_{2k} + \delta_{3k} < \frac{1}{4}$.

RIP y Basis Pursuit

RIP y recuperación vía BP

Sea A un (δ_k, k)-RIP y supongamos que:

$$\delta_k + \theta_{k,2k} + \theta_k < 1$$

Entonces BP recupera las soluciones de esparcidad k.

Usando un lema sencillo que dice $\theta_{k,k'} < \delta_{k+k'} < \theta_{k,k'} + \max\{\delta_k, \delta_{k'}\}$ (donde δ_k son valores que hacen que A sea k-RIP), nuestra condición es implicada por $\delta_k + \delta_{2k} + \delta_{3k} < \frac{1}{4}$.

Inconvenientes

Calcular RIP es NP completo.

RIP y Basis Pursuit

RIP y recuperación vía BP

Sea A un (δ_k, k)-RIP y supongamos que:

$$\delta_k + \theta_{k,2k} + \theta_k < 1$$

Entonces BP recupera las soluciones de esparcidad k.

Usando un lema sencillo que dice $\theta_{k,k'} < \delta_{k+k'} < \theta_{k,k'} + \max\{\delta_k, \delta_{k'}\}$ (donde δ_k son valores que hacen que A sea k-RIP), nuestra condición es implicada por $\delta_k + \delta_{2k} + \delta_{3k} < \frac{1}{4}$.

Inconvenientes

Calcular RIP es NP completo.

Dos enfoques:

- Construcciones probabilísticas.
- Construcciones deterministicas.

Matrices Gaussianas y RIP

Matrices Gaussianas y RIP

Sea $A \in \mathbb{R}^{m \times n}$ una matriz donde sus coordenadas son Gaussianas $\mathcal{N}(0, \frac{1}{m})$ iid. La idea es acotar los autovalores de $X^t X$ para conjuntos de k columnas de A.

Matrices Gaussianas y RIP

Sea $A \in \mathbb{R}^{m \times n}$ una matriz donde sus coordenadas son Gaussianas $\mathcal{N}(0, \frac{1}{m})$ iid. La idea es acotar los autovalores de $X^t X$ para conjuntos de k columnas de A. Afortunadamente, tenemos el siguiente resultado conocido: si $k/m \to \alpha \leq 1$ entonces resulta que

$$\lambda_{\min}(X^t X) \to (1 - \sqrt{\alpha})^2 \quad y \quad \lambda_{\max}(X^t X) \to (1 + \sqrt{\alpha})^2 \quad c.s.$$

Matrices Gaussianas y RIP

Sea $A \in \mathbb{R}^{m \times n}$ una matriz donde sus coordenadas son Gaussianas $\mathcal{N}(0, \frac{1}{m})$ iid. La idea es acotar los autovalores de $X^t X$ para conjuntos de k columnas de A. Afortunadamente, tenemos el siguiente resultado conocido: si $k/m \to \alpha \leq 1$ entonces resulta que

$$\lambda_{\text{min}}(X^t X) \to (1 - \sqrt{\alpha})^2 \quad y \quad \lambda_{\text{max}}(X^t X) \to (1 + \sqrt{\alpha})^2 \quad c.s.$$

y usando desigualdades de concentración, se puede argumentar que la probabilidad de que **no sea** (δ, k)-RIP decrece exponencialmente.

Matrices Gaussianas y RIP

Sea $A \in \mathbb{R}^{m \times n}$ una matriz donde sus coordenadas son Gaussianas $\mathcal{N}(0, \frac{1}{m})$ iid. La idea es acotar los autovalores de $X^t X$ para conjuntos de k columnas de A. Afortunadamente, tenemos el siguiente resultado conocido: si $k/m \to \alpha \leq 1$ entonces resulta que

$$\lambda_{\min}(X^t X) \to (1 - \sqrt{\alpha})^2 \quad y \quad \lambda_{\max}(X^t X) \to (1 + \sqrt{\alpha})^2 \quad c.s.$$

y usando desigualdades de concentración, se puede argumentar que la probabilidad de que **no sea** (δ, k)-RIP decrece exponencialmente.

Resultado Preciso

Si $A \in \mathbb{R}^{m \times n}$ como descripta, si $r = S/n$ entonces definiendo $f(r) = \sqrt{n/m}(\sqrt{r} + \sqrt{2H(r)})$, para $\varepsilon > 0$ vale que $P(1 + \delta_S > (1 + (1 + \varepsilon)f(r))^2) \leq 2e^{-\frac{1}{2}mH(r)\varepsilon}$

Caso no lineal

Muchas veces no nos importa encontrar la solución más esparsa, sino alguna solución esparsa a un problema particular. En este caso, el problema que tenemos el siguiente

$$P_0 : \min H(x) \quad ||x||_0 < k$$

Caso no lineal

Muchas veces no nos importa encontrar la solución más esparsa, sino alguna solución esparsa a un problema particular. En este caso, el problema que tenemos el siguiente

$$P_0 : \min H(x) \quad ||x||_0 < k$$

Y en muchos casos, al igual que el caso lineal, es conveniente trabajar con la regularización ℓ_1 de este problema

$$P_1 : \min H(x) + \lambda ||Wx||_1$$

Caso no lineal

Muchas veces no nos importa encontrar la solución más esparsa, sino alguna solución esparsa a un problema particular. En este caso, el problema que tenemos el siguiente

$$P_0 : \min H(x) \quad ||x||_0 < k$$

Y en muchos casos, al igual que el caso lineal, es conveniente trabajar con la regularización ℓ_1 de este problema

$$P_1 : \min H(x) + \lambda||Wx||_1$$

En este contexto general, es muy raro que obtengamos para algún λ la verdadera solución más esparsa, pero la penalización de la norma ℓ_1 tiende a matar coordenadas chicas.

Ejemplo: Regularización ℓ_1 de Redes Neuronales

En general, dada una red neuronal que depende de pesos $x \in \mathbb{R}^N$ uno puede agregar un regularizador (por ejemplo, para evitar overfitting). La versión clásica es

$$\min \mathcal{L}(x) + \lambda ||Wx||_2$$

Ejemplo: Regularización ℓ_1 de Redes Neuronales

En general, dada una red neuronal que depende de pesos $x \in \mathbb{R}^N$ uno puede agregar un regularizador (por ejemplo, para evitar overfitting). La versión clásica es

$$\min \mathcal{L}(x) + \lambda ||Wx||_2$$

Pero si queremos forzar a que muchos pesos sean 0 (y así posiblemente mejorar la eficiencia) podemos hacer la siguiente regularización:

$$\min \mathcal{L}(x) + \lambda ||Wx||_1$$

Ejemplo: Regularización ℓ_1 de Redes Neuronales

En general, dada una red neuronal que depende de pesos $x \in \mathbb{R}^N$ uno puede agregar un regularizador (por ejemplo, para evitar overfitting). La versión clásica es

$$\min \mathcal{L}(x) + \lambda ||Wx||_2$$

Pero si queremos forzar a que muchos pesos sean 0 (y así posiblemente mejorar la eficiencia) podemos hacer la siguiente regularización:

$$\min \mathcal{L}(x) + \lambda ||Wx||_1$$

En este contexto, el λ decide el balance entre los dos sumandos:

- Si λ es chico, el modelo aproxima bien los datos pero no es esparso (y puede tener parámetros grandes).

Ejemplo: Regularización ℓ_1 de Redes Neuronales

En general, dada una red neuronal que depende de pesos $x \in \mathbb{R}^N$ uno puede agregar un regularizador (por ejemplo, para evitar overfitting). La versión clásica es

$$\min \mathcal{L}(x) + \lambda ||Wx||_2$$

Pero si queremos forzar a que muchos pesos sean 0 (y así posiblemente mejorar la eficiencia) podemos hacer la siguiente regularización:

$$\min \mathcal{L}(x) + \lambda ||Wx||_1$$

En este contexto, el λ decide el balance entre los dos sumandos:

- Si λ es chico, el modelo aproxima bien los datos pero no es esparso (y puede tener parámetros grandes).
- Si λ es grande, el modelo es esparso pero estima peor los datos.

Ejemplo: Regularización ℓ_1 de Redes Neuronales

En general, dada una red neuronal que depende de pesos $x \in \mathbb{R}^N$ uno puede agregar un regularizador (por ejemplo, para evitar overfitting). La versión clásica es

$$\min \mathcal{L}(x) + \lambda ||Wx||_2$$

Pero si queremos forzar a que muchos pesos sean 0 (y así posiblemente mejorar la eficiencia) podemos hacer la siguiente regularización:

$$\min \mathcal{L}(x) + \lambda ||Wx||_1$$

En este contexto, el λ decide el balance entre los dos sumandos:

- Si λ es chico, el modelo aproxima bien los datos pero no es esparso (y puede tener parámetros grandes).
- Si λ es grande, el modelo es esparso pero estima peor los datos.

Ojo. λ juega un doble papel: eliminar coordenadas chicas, pero también

Referencias

- Alfred M. Bruckstein, David L. Donoho, and Michael Elad. *From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images.* 2009
- Ronald A. DeVore, *Deterministic constructions of compressed sensing matrices*, Journal of Complexity, Volume 23, Issues 4–6, 2007.
- Emmanuel Candes and Terence Tao. *Decoding by Linear Programming*. 2005.
- J. A. Tropp, *Greed is good: algorithmic results for sparse approximation*, in IEEE Transactions on Information Theory, vol. 50, no. 10, pp. 2231-2242, Oct. 2004.
- Huan Li and Zhouchen Lin. *Construction of Incoherent Dictionaries via Direct Babel Function Minimization*. 2018.