Universidad de los Andes

Departamento de Matemáticas

Optimización Convexa

Proyecto de Clustering + Boosting

Juan Nicolás Mendoza Roncancio

Contenido

1 Deducción del dual

El problema de clustering convexo se puede expresar de la forma: Dados n puntos en $\mathbb{R}^p$ queremos minimizar:

$$F_\gamma(U) = \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{i<j} w_{ij}||u_i - u_j||$$

Done γ es un parámetro de penalización positivo, w_{ij} es un peso no negativo y la i-ésima columna de U es el centro del cluster al cual x_i pertenece.

Podemos reformular este problema como sigue:

$$\min \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| \tag{1}$$

$$\text{s.t } u_{l_1} - u_{l_2} - v_l = 0 \tag{2}$$

En este caso, el conjunto $\mathcal{E}$ es el conjunto de las aristas del grafo completo formado por los centros para las cuales w_{ij} es no nulo, formalmente: $\mathcal{E} = \{l = (l_1, l_2) : w_l > 0\}$, indexamos los centroides justamente con las parejas $l = (l_1, l_2)$ tal que $l_1 < l_2$.

El objetivo de esta primera sección es demostrar que el dual del problema anterior es:

$$D_\gamma(\Lambda) = -\frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_l \langle \lambda_l, x_{l_1} - x_{l_2}\rangle - \sum_l \delta_{C_l}(\lambda_l) \tag{3}$$

Con $\Delta_i = \sum_{l:l_1=i} \lambda_l - \sum_{l:l_2=i} \lambda_l$ y $\delta_{C_l}(\lambda_l)$ es la función indicadora del conjunto $C_l = \{\lambda_l : ||\lambda_l||_* \leq \gamma w_l\}$

Primero calculemos el lagrangiano de nuestro problema, las matrices U, V y Λ son respectivamente las matrices de centros, de los vectores v_l definidas en (2) y la matriz de los multiplicadores de Lagrange λ_l:

$$\begin{aligned}
\mathcal{L}(U, V, \Lambda) &= \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| + \sum_{l\in\mathcal{E}} \langle \lambda_l, u_{l_1} - u_{l_2} - v_l\rangle \\
&= \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}} \langle \lambda_l, v_l\rangle + \sum_{l\in\mathcal{E}} \langle \lambda_l, u_{l_1} - u_{l_2}\rangle \\
&= \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}} \langle \lambda_l, v_l\rangle + \sum_{i=1}^{n} \langle u_i, \sum_{l:l_1=i} \lambda_l - \sum_{l:l_2=i} \lambda_l\rangle \\
&= \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}} \langle \lambda_l, v_l\rangle + \sum_{i=1}^{n} \langle u_i, \Delta_i\rangle \\
&= \frac{1}{2}\sum_{i=1}^{n} ||x_i - u_i||_2^2 + \sum_{i=1}^{n} \langle u_i, \Delta_i\rangle + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}} \langle \lambda_l, v_l\rangle
\end{aligned}$$

Ahora vamos a calcular:

$$(\nabla_U \mathcal{L})_i = -(x_i - u_i) + \Delta_i$$

Si u_i^* es óptimo:

$$u_i^* = x_i - \Delta_i \tag{4}$$

Reemplazando en el dual:

$$\begin{aligned}
\mathcal{L}(U^*, V, \Lambda) &= \frac{1}{2}\sum_{i=1}^{n} ||x_i - (x_i - \Delta_i)||_2^2 + \sum_{i=1}^{n}\langle x_i - \Delta_i, \Delta_i\rangle + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}}\langle \lambda_l, v_l\rangle \\
&= \frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_{i=1}^{n}\langle x_i, \Delta_i\rangle - \sum_{i=1}^{n} ||\Delta_i||_2^2 + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}}\langle \lambda_l, v_l\rangle \\
&= -\frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_{i=1}^{n}\langle x_i, \Delta_i\rangle + \gamma \sum_{l\in\mathcal{E}} w_l||v_l|| - \sum_{l\in\mathcal{E}}\langle \lambda_l, v_l\rangle
\end{aligned}$$

Ahora debemos repetir lo mismo con v_l, fijemos entonces l, así queremos encontrar $\inf_{v_l}\{\gamma w_l||v_l|| - \langle \lambda_l, v_l\rangle\}$. Tenemos que:

$$\inf_{v_l}\{\gamma w_l||v_l|| - \langle \lambda_l, v_l\rangle\} = -\sup_{v_l}\{\langle \lambda_l, v_l\rangle - \gamma w_l||v_l||\} = -f^*(\lambda_l),$$

con $f(v_l) = \gamma w_l||v_l||$ y f^* el conjugado convexo de f. Entonces, ahora nuestro problema se reduce a calcular f^*.

El conjugado convexo de $||v||$ es $\delta_B(v)$ con B la bola unitaria en la métrica dual $||.||_*$, usando que $(af)^*(x^*) = af^*(\frac{x^*}{a})$ obtenemos:

$$f^*(\lambda_l) = \begin{cases} (\gamma w_l)(0), & ||\frac{\lambda_l}{\gamma w_l}||_* \leq 1 \\ (\gamma w_l)(\infty), & ||\frac{\lambda_l}{\gamma w_l}||_* > 1 \end{cases} = \begin{cases} 0 & ||\lambda_l||_* \leq \gamma w_l \\ \infty, & ||\lambda_l||_* > \gamma w_l \end{cases} = \delta_{C_l}(\lambda_l)$$

Entonces $\inf_{v_l}\{\gamma w_l||v_l|| - \langle \lambda_l, v_l\rangle\} = -\delta_{C_l}(\lambda_l)$. Reemplazando en el lagrangiano:

$$\mathcal{L}(U^*, V^*, \Lambda) = -\frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_{i=1}^{n}\langle x_i, \Delta_i\rangle - \sum_{l\in\mathcal{E}} \delta_{C_l}(\lambda_l)$$

Veamos con mayor detalle el segundo sumando:

$$\begin{aligned}
\sum_{i=1}^{n}\langle x_i, \Delta_i\rangle &= \sum_{i=1}^{n}\left\langle \sum_{l:l_1=i} \lambda_l - \sum_{l:l_2=i} \lambda_l, x_i \right\rangle \\
&= \sum_{i=1}^{n}\sum_{l:l_1=i}\langle \lambda_l, x_i\rangle - \sum_{i=1}^{n}\sum_{l:l_2=i}\langle \lambda_l, x_i\rangle \\
&= \sum_{l\in\mathcal{E}}\langle \lambda_l, x_{l_1} - x_{l_2}\rangle
\end{aligned}$$

Con esto, nuestro dual es finalmente:

$$D_\lambda(\Lambda) = \mathcal{L}(U^*, V^*, \Lambda) = -\frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_{l\in\mathcal{E}}\langle \lambda_l, x_{l_1} - x_{l_2}\rangle - \sum_{l\in\mathcal{E}} \delta_{C_l}(\lambda_l) \tag{5}$$

Tal y como se quería.

2 Visualización como un problema de mínimos cuadrados

El objetivo de esta sección es entender el dual como un problema de mínimos cuadrados con restricciones. El dual de nuestro problema es:

$$D_\gamma(\Lambda) = -\frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2 + \sum_{l\in\mathcal{E}} \langle \lambda_l, x_{l_1} - x_{l_2}\rangle - \sum_{l\in\mathcal{E}} \delta_{C_l}(\lambda_l)$$

Queremos verlo como:

$$\min_\beta ||X\beta - y||_2^2 \tag{6}$$

$$\text{s.t } f(X) \leq 0 \tag{7}$$

Es claro que la restricción es el último término, esta nos indica que los valores que encontremos de λ_l deben estar en $C_l = \{\lambda_l : ||\lambda_l|| \leq \gamma w_l\}$, entonces podemos centrarnos en expresar nuestro dual de la forma $\min_\beta ||X\beta - y||_2^2$ para alguna matriz X y vector y.

La idea seria encontrar un β tal que $X\beta = y$ esto es equivalente a encontrar el mínimo de la siguiente función: $\beta^t X\beta - y^t\beta$, esta última función se parece más a lo que ya tenemos, de hecho es posible identificar que, los superindices indican una enumeración de las aristas del grafo:

$$y = \begin{bmatrix} x_{l_1}^{(1)} - x_{l_2}^{(1)} \\ x_{l_1}^{(2)} - x_{l_2}^{(2)} \\ \vdots \\ x_{l_1}^{(|\mathcal{E}|)} - x_{l_2}^{(|\mathcal{E}||)} \end{bmatrix}$$

Es decir, el vector y es el vector cuyas componentes son las diferencias entre las nodos de las aristas del grafo. Esto fuerza a que:

$$\beta = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_{\mathcal{E}} \end{bmatrix}$$

Definimos también:

$$\beta^{||\cdot||} = \begin{bmatrix} ||\lambda_1||_2 \\ ||\lambda_2||_2 \\ \vdots \\ ||\lambda_{\mathcal{E}}||_2 \end{bmatrix}$$

Ahora nos gustaría encontrar una matriz X de tal forma que:

$$\beta^t X\beta = \frac{1}{2}\sum_{i=1}^{n} ||\Delta_i||_2^2$$

Definimos entonces la matriz B. Cada columna de B corresponde a una arista del grafo. Si estamos en la columna $l^i = (l^i_1, l^i_2)$ de la matriz entonces: $B_{l^i_1,l^i} = 1, B_{l^i_2,l^i} = -1$ y $B_{k,l^i} = 0$ en otro caso, entonces B es la matriz de incidencia de nuestro grafo. Es importante recordar que la matriz B tiene n filas por $m := |\mathcal{E}|$ columnas.

Ahora, para construir la matriz $\bar{B}$ lo hacemos de la siguiente manera:

$$\bar{B} = B \otimes I_d$$

Con I_d la matriz identidad de dimensión d, de esta manera:

$$\bar{B}\beta = \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \vdots \\ \Delta_n \end{bmatrix}$$

Como queremos obtener $\sum_{i=1}^{n} ||\Delta_i||_2^2$ podemos hacer:

$$(\bar{B}\beta)^t \bar{B}\beta = \beta^t \bar{B}^t \bar{B}\beta \tag{8}$$

$$= \sum_{i=1}^{n} ||\Delta_i||_2^2 \tag{9}$$

Tal como se quería. Ahora queremos expresar nuestro problema como un problema de mínimos cuadrados podemos ver que:

$$\bar{B}^t x = (B \otimes I_d)^t x = B^t x = y$$

También:

$$\frac{1}{2}||\bar{B}\beta - x||_2^2 = \frac{1}{2}\beta^t \bar{B}^t \bar{B}\beta - \beta^t \bar{B}^t x + \frac{1}{2}||x||_2^2$$

Que es justamente nuestro dual, salvo el último factor, que como es constante podemos ignorarlo. Entonces:

$$\frac{1}{2}||\bar{B}\beta - x||_2^2 = -D_\gamma(\Lambda)$$

Maximizar $D_\gamma(\Lambda)$ equivale a minimizar $-D_\gamma(\Lambda)$ es decir: $\frac{1}{2}||\bar{B}\beta - x||_2^2$. Hemos encontrado nuestro problema de mínimos cuadrados:

$$\min_\beta ||\bar{B}\beta - x||_2^2 \tag{10}$$

$$s.t\, \beta_i^{||.||} \leq \gamma w_{l_i} \tag{11}$$

Con $\bar{B}$ y β definidas arriba. Verifiquemos que este bien definido: el vector x es el vector de los datos, como son n datos en $\mathbb{R}^p$ obtenemos que, $x \in \mathbb{R}^{np}$. $\bar{B} = B \otimes I_d \in \mathbb{R}^{np \times md}$ y $\beta \in \mathbb{R}^{md}$, luego el problema esta bien definido.

3 Algoritmo de boosting

Hagamos un esbozo de como sería el algoritmo de boosting más sencillo en este caso, Para aplicarlo debemos tener que las columnas de $\bar{B}$ tienen media cero y norma 2 unitaria, demas que el vector de datos también tiene norma cero (esto tiene centrido porque sería centrar los datos en cero):

Algoritmo 1 Algoritmo boosting en el caso de clustering

1: Ingresan un conjunto de datos en $\mathbb{R}^p$, un grafo que une a estos datos, su conjunto de aristas $\mathcal{E}$, el peso para cada arista $l \in \mathcal{E}, w_l$,el parámetro de castigo γ, el learning rate $\epsilon > 0$ y el número máximo de iteraciones para Boosting.

2: Ahora debemos construir la matriz $\bar{B}$, que se construye a partir de la matriz B, que a su vez se construye con el conjunto $\mathcal{E}$.

3: Aplicamos Boosting al problema (10), iniciamos con $\hat{r}^0 = x$, $\hat{\beta}^0 = 0, k = 0$ Sí, esto indica que cada punto comienza siendo su centro: Si $\hat{\beta} = 0$, entonces cada λ es cero, y de acuerdo a (4), cada punto es su centro

4: **Repita**

5: Para $0 \leq k \leq M$

6: Encuentre $j_k, \bar{c}_{j_K}$:

$$\bar{c}_m = \arg\min_{c \in \mathbb{R}} \left(\sum_{i=1}^{np} (\hat{r}_i^k - x_{il} c)^2 \right) \text{ Para } l = 1, ..., md$$

$$j_k \in \arg\min_{1 \leq l \leq md} \sum_{i=1}^{np} (\hat{r}_i^k - x_{il} \hat{c}_l)^2$$

7: Actualizamos los residuos:

$$\hat{r}^{k+1} \longleftarrow \hat{r}^k - \epsilon \bar{B}_{j_k} \bar{c}_{j_k}$$

$$\hat{\beta}_{j_k}^{k+1} \longleftarrow \hat{\beta}_{j_k}^k + \epsilon \bar{c}_{j_k} \text{ y no se alteran los demás indices } j$$

8: Hacemos que se cumpla lasso proyectando el conjunto formado por las restricciones.

9: **Hasta** Se llegan a M iteraciones

10: Boosting nos da un vector β. Ahora usamos la ecuación (4) para recuperar los centros de los datos

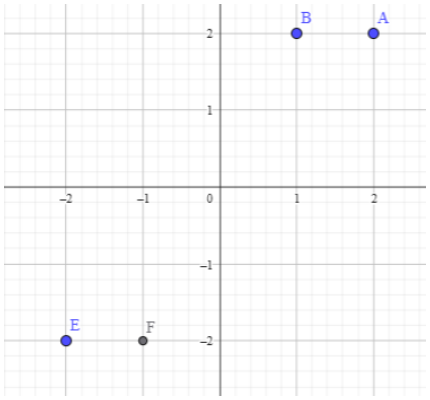
El paso 6 creo que se interpretaría como sigue: Para cada columna de $\bar{B}$ (serían como los datos) buscamos un c que se ajuste a mejor a los residuos, sería como dejar fijo un punto y vamos moviendo los λ actuales de tal forma que se ajusten mejor a los residuos actuales, sería como apagar o encender las conexiones entre los datos, luego vemos en que punto fue en el que más movimos los λ

En el paso 7, a los residuos actuales les quitamos lo que más se modifico, es decir, dejamos encendidas o apagadas las conexiones entre los datos. En los λ hacemos algo similar: Apagamos o encendemos los que más se modificaron

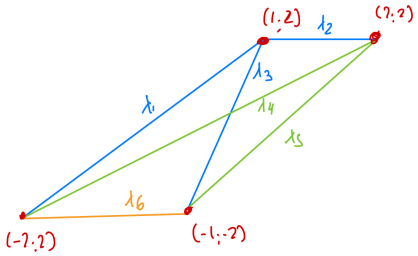
4 Ejemplos computacionales en el plano

4.1 Ejemplo con 4 puntos

Hagamos un ejemplo con los siguientes datos en $\mathbb{R}^2$: $\{(2,2),(1,2),(-2,-2),(-1,-2)\}$:



La idea es obtener 2 centros en $(\frac{3}{2},2)$ Y $(\frac{-3}{2},2)$. El grafo que une a estos puntos es el siguiente, cada arista tiene peso de 1, el parámetro $\gamma = 1/2$ y $\epsilon = 1/2$:



La matriz B sería:

$$B = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

Entonces la matriz (preliminar) $\bar{B}$ es:

$$\bar{B} = \begin{bmatrix} I_2 & I_2 & I_2 & 0 & 0 & 0 \\ 0 & -I_2 & 0 & I_2 & I_2 & 0 \\ -I_2 & 0 & 0 & -I_2 & 0 & -I_2 \\ 0 & 0 & -I_2 & 0 & -I_2 & -I_2 \end{bmatrix}$$

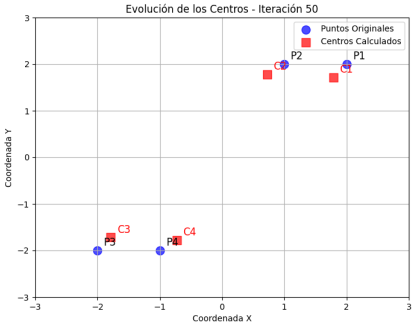
Esta última matriz es una matriz 8×12. Nuestro vector de datos será entonces:

$$x = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ -2 \\ -2 \\ -1 \\ -2 \end{bmatrix}$$

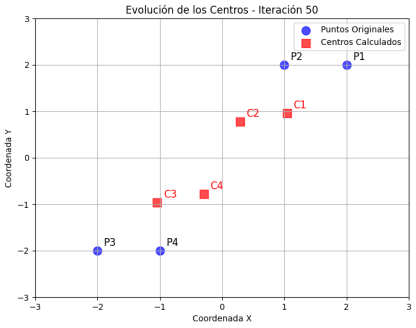
Note que x ya tiene media cero, resta normalizar. También ya todas las columnas de $\bar{B}$ tienen media cero, para normalizarlas multiplicamos por $\frac{1}{\sqrt{2}}$, obteniendo así, redefinimos:

$$\bar{B} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_2 & I_2 & I_2 & 0 & 0 & 0 \\ 0 & -I_2 & 0 & I_2 & I_2 & 0 \\ -I_2 & 0 & 0 & -I_2 & 0 & -I_2 \\ 0 & 0 & -I_2 & 0 & -I_2 & -I_2 \end{bmatrix}$$

Para Boosting, el número máximo de iteraciones será 50. Luego de correr el primer código incluido en los anexos (que aplica el Algoritmo 1) encontramos los siguientes centros:

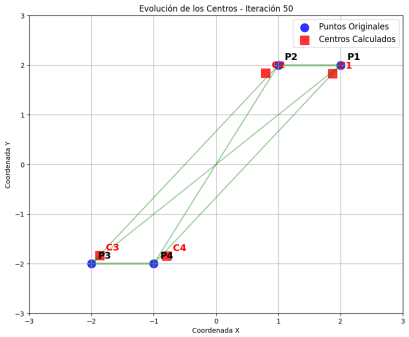


Podemos preguntarnos también que sucede si usamos un γ mayor, por ejemplo $\gamma = 2$, obtenemos entonces: Vemos entonces que, a medida que γ crece, los centros tienden a



converger al centro mismo de los datos. Este es un resultado coherente con los obtenidos por [1] y [2].

Ahora podemos preguntarnos qué pasaría si los pesos no son uniformes, qué pasaría por ejemplo si el centro entre los datos x_i, x_j es $w_{ij} = \frac{1}{1+d_{ij}}$ con d_{ij} la distancia entre estos dos datos. Entonces obtenemos: El grosor de las aristas es proporcional al peso de las

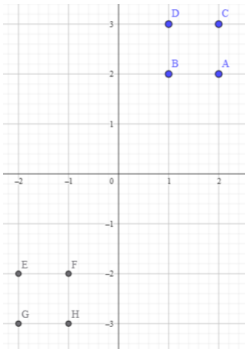


mismas, los valores para los hiperparametros usados en el experimento anterior son los mismos que los usados en los pesos uniformes. Entonces podríamos concluir dos cosas de los experimentos anteriores:

1. Los pesos de las aristas deben ser de alguna manera "función de los datos" para poder un resultado más cercano al esperado, por ejemplo, si los pesos son asignados de manera inversamente proporcional a la longitud entre los extremos de la aristas, obtenemos resultados mejores que si fueran uniformes.
2. El parámetro γ delimita la variabilidad de los centros: Entre mayor sean los centros, mayor será el movimiento de los centros.

4.2 Ejemplo con 8 puntos:

Para poder explotar las conclusiones anteriores proponemos agrupar el siguiente conjunto de datos: $\{(2,2),(1,2),(-2,-2),(-1,-2),(2,3),(1,3),(-2,-3),(-1,-3)\}$: Para



los experimentos, que se implementaron con el segundo código de los anexos, usamos los siguientes parámetros: el máximo de iteraciones será siempre 1500, $\epsilon = 0.01$ siempre. y variaremos γ entre $2, 5$ y 10. Veamos como construimos los pesos entonces:

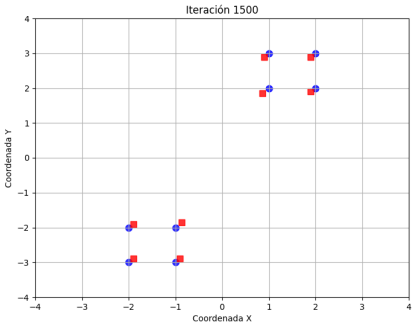
- Primero calculamos las distancias entre los puntos, esto genera una matriz cuadrada con de cada entrada ij es la distancia entre x_i, x_j

- Luego seleccionamos los 3 vecinos más cercanos a cada punto.
- Para cada punto x_i calculamos el promedio de las distancias entre x_i y sus 3 vecinos más cercanos, llamemos a este número m_{ij}.
- Para cada punto x_j diferente de x_i escogemos el mínimo entre la distancia $d_{ij} = d(x_i, x_j)$ y el promedio calculado en el punto anterior, llamemos este número k_{ij}, luego asignamos:

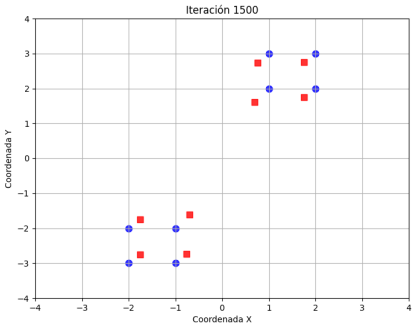
$$w_{ij} = e^{\frac{-d_{ij}}{2k_{ij}}}$$

Entonces hay tres casos naturales: Si $m_{ij} = d_{ij}$ entonces obtenemos que $w_{ij} = e^{\frac{-1}{2}}$, si $m_{ij} < d_{ij}$ entonces obtenemos que $w_{ij} = e^{\frac{-d_{ij}}{2m_{ij}}} \longrightarrow 0$, por último si: $m_{ij} > d_{ij}$ entonces $w_{ij} = e^{\frac{-d_{ij}}{2m_{ij}}} \longrightarrow 1$. Esta función entonces considera la densidad de la distribución de puntos para cada uno de los datos, asignando valores de pesos altos a los puntos cercanos y bajos a los lejanos. Ademas, estos valores están normalizados.

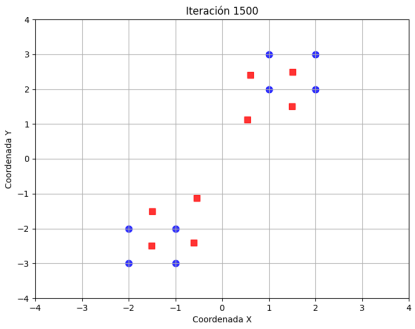
Veamos ahora los resultados, primero, para $\gamma = 2$:



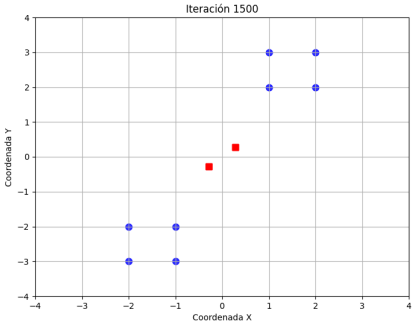
Y para $\gamma = 5$ obtenemos respectivamente:



También para $\gamma = 10$



Vemos entonces cierto patrón, que pasaría entonces si $\gamma = 100$? obtenemos en este caso: Concluimos entonces que, nuevamente, si γ es mayor, los centros convergen al centro de



los datos. Es importante resaltar que en los experimentos anteriores los centros no se movían significadamente, indicando que para cada γ los centros tienen su propio camino hasta llegar al optimo, esto podría ser por las restricciones impuestas en el problema de regresión lineal.

5 Anexos

Codigó 1:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML


puntos = np.array([[2, 2], [1, 2], [-2, -2], [-1, -2]])
gamma = 2
epsilon = 0.5
max_iter = 50

# Construir matriz B
B = np.array([
    [1, 1, 1, 0, 0, 0],
    [0, -1, 0, 1, 1, 0],
    [-1, 0, 0, -1, 0, -1],
    [0, 0, -1, 0, -1, -1]
])

# Construir matriz BarB
I_d = np.eye(2)
Bar = np.kron(B, I_d) * (1/np.sqrt(2))


x = puntos.flatten().astype(float)

n, d = puntos.shape
m = B.shape[1]
psi = x.copy()
beta = np.zeros(m * d)
w = np.ones(m)

all_centros = []

# Calcular centros iniciales
centros_actual = (x - Bar @ beta).reshape(-1, 2)
all_centros.append(centros_actual.copy())

# Funcion para proyectar a la restriccion
def project(beta, gamma, w):
    beta_projected = beta.copy()
    for l in range(m):
        idx = slice(l*d, (l+1)*d)
        beta_l = beta[idx]
        norm = np.linalg.norm(beta_l)
        max_norm = gamma * w[l]
        if norm > max_norm:
            beta_projected[idx] = beta_l * max_norm / norm
    return beta_projected

# Algoritmo de boosting
for k in range(max_iter):
    # Paso 6: Calcular tau_j para cada columna j
```

```
    taus = Bar.T @ psi

    # Encontrar j_k
    j_k = np.argmax(taus**2)

    # Actualizar residuos y beta
    Bar_jk = Bar[:, j_k]
    psi_new = psi - epsilon * Bar_jk * taus[j_k]

    beta_new = beta.copy()
    beta_new[j_k] += epsilon * taus[j_k]

    # Proyectar a las restricciones
    beta_projected = project(beta_new, gamma, w)

    # Actualizar para siguiente iteracion
    psi = psi_new
    beta = beta_projected

    # Calcular nuevos centros
    centros_actual = (x - Bar @ beta).reshape(-1, 2)
    all_centros.append(centros_actual.copy())

# Animacion
all_centros = np.array(all_centros)


fig, ax = plt.subplots(figsize=(8, 6))
ax.set_xlim(-3, 3)
ax.set_ylim(-3, 3)
ax.grid(True)
ax.set_title('Evolucion de los Centros')
ax.set_xlabel('Coordenada X')
ax.set_ylabel('Coordenada Y')

scatter_puntos = ax.scatter(puntos[:, 0], puntos[:, 1], c='blue', s
    =100,
                            label='Puntos Originales', alpha=0.7)


scatter_centros = ax.scatter([], [], c='red', s=100, marker='s',
                             label='Centros Calculados', alpha=0.7)


for i, punto in enumerate(puntos):
    ax.text(punto[0]+0.1, punto[1]+0.1, f'P{i+1}', fontsize=12)


ax.legend(loc='upper right')
ax.annotate(f'gamma = {gamma}, epsilon = {epsilon}',
            xy=(0.05, 0.95), xycoords='axes fraction',
            fontsize=10, bbox=dict(boxstyle="round,pad=0.3", fc="white"
                , ec="gray", alpha=0.7))


def init():
    scatter_centros.set_offsets(np.empty((0, 2)))
    return scatter_centros,
```

```

def animate(i):
    scatter_centros.set_offsets(all_centros[i])
    ax.set_title(f'Evolucion de los Centros - Iteracion {i}')

    for txt in ax.texts[len(puntos):]:
        txt.remove()

    for j, centro in enumerate(all_centros[i]):
        ax.text(centro[0]+0.1, centro[1]+0.1, f'C{j+1}',
                fontsize=12, color='red')

    return scatter_centros,

ani = animation.FuncAnimation(
    fig, animate, frames=len(all_centros),
    init_func=init, blit=True, interval=1000
)

plt.close(fig)

HTML(ani.to_jshtml())
```

Listing 1: Codigó I

Codigó II

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.display import HTML
from scipy.spatial.distance import cdist
import matplotlib

matplotlib.rcParams['animation.embed_limit'] = 100

np.random.seed(42)
gamma = 100.0
epsilon = 0.01
max_iter = 1500

puntos = np.array([
    [2, 2], [1, 2], [-2, -2], [-1, -2],
    [2, 3], [1, 3], [-2, -3], [-1, -3]
])

media = np.mean(puntos, axis=0)
puntos_centrados = puntos - media
n, d = puntos_centrados.shape

#Construir matriz de incidencia B
m = n * (n - 1) // 2
B = np.zeros((n, m))
aristas = []
idx = 0
for i in range(n):
    for j in range(i+1, n):
```

```
        B[i, idx] = 1
        B[j, idx] = -1
        aristas.append((i, j))
        idx += 1

#Calculo de Pesos
distancias = cdist(puntos_centrados, puntos_centrados)

vecinos_cercanos = {}
for i in range(n):
    dists = distancias[i].copy()
    dists[i] = np.inf
    vecinos_cercanos[i] = np.argsort(dists)[:3]

distancias_promedio = {}
for i in range(n):
    distancias_vecinos = distancias[i, vecinos_cercanos[i]]
    distancias_promedio[i] = np.mean(distancias_vecinos)

w = np.zeros(m)
for idx, (i, j) in enumerate(aristas):
    dist_ij = distancias[i, j]
    avg_dist = min(distancias_promedio[i], distancias_promedio[j])
    w[idx] = np.exp(-dist_ij / (2 * avg_dist))

w = np.clip(w, 0.1, 1.0)

#Matriz BarB
I_d = np.eye(d)
Bar = np.kron(B, I_d)
column_norms = np.linalg.norm(Bar, axis=0)
Bar = Bar / np.where(column_norms > 0, column_norms, 1)

x = puntos_centrados.flatten().astype(float)

#Boosting
psi = x.copy()
beta = np.zeros(m * d)

# Almacenar cada 5 frames
store_every = 5
all_centros = []
centros_actual = (x - Bar @ beta).reshape(-1, d)
all_centros.append(centros_actual.copy())

# Funcion de proyeccion
def project(beta, gamma, w):
    beta_projected = beta.copy()
    for l in range(m):
        idx = slice(l * d, (l + 1) * d)
        beta_l = beta[idx]
        norm = np.linalg.norm(beta_l)
        max_norm = gamma * w[l]
        if norm > max_norm:
            beta_projected[idx] = beta_l * max_norm / norm
    return beta_projected


```

```python
# Algoritmo de Boosting
for k in range(max_iter):
    taus = Bar.T @ psi
    j_k = np.argmax(taus**2)
    Bar_jk = Bar[:, j_k]
    psi_new = psi - epsilon * Bar_jk * taus[j_k]
    beta_new = beta.copy()
    beta_new[j_k] += epsilon * taus[j_k]
    beta_projected = project(beta_new, gamma, w)
    psi = psi_new
    beta = beta_projected
    centros_actual = (x - Bar @ beta).reshape(-1, d)

    if k % store_every == 0:
        all_centros.append(centros_actual.copy())


# Resultados
print("="*50)
print("Resultados finales:")
print(f"Numero de iteraciones: {k+1}")
print(f"Centros calculados:")
for i, centro in enumerate(centros_actual):
    print(f"C{i}: {centro}")
print("="*50)

# Paso 8: Animacion
all_centros = np.array(all_centros)

fig, ax = plt.subplots(figsize=(8, 6))
ax.set_xlim(-4, 4)
ax.set_ylim(-4, 4)
ax.grid(True)
ax.set_title('Evolucion de los Centros')
ax.set_xlabel('Coordenada X')
ax.set_ylabel('Coordenada Y')

scatter_puntos = ax.scatter(puntos_centrados[:, 0], puntos_centrados[:, 1],
                            c='blue', s=60, label='Puntos Originales', alpha=0.8)

scatter_centros = ax.scatter([], [], c='red', s=60, marker='s',
                             label='Centros Calculados', alpha=0.8)

def init():
    scatter_centros.set_offsets(np.empty((0, 2)))
    return scatter_centros,

def animate(i):
    centros = all_centros[i]
    scatter_centros.set_offsets(centros)
    ax.set_title(f'Iteracion {i*store_every}')
    return scatter_centros,

ani = animation.FuncAnimation(
    fig, animate, frames=len(all_centros),
    init_func=init, blit=True, interval=500
```

```
)

plt.close(fig)

HTML(ani.to_jshtml())
```

Listing 2: Codigó II

6 Referencías

[1] Chi, E. C., & Lange, K. (2015). Splitting methods for convex clustering. Journal of Computational and Graphical Statistics, *24*(4), 994–1013.
[2] Tan, K. M., & Witten, D. (2015). Statistical properties of convex clustering. Electronic Journal of Statistics, *9*(2), 2324–2347.
[3] Freund, R. M., Grigas, P., & Mazumder, R. (2013). A New Perspective on Boosting in Linear Regression via Subgradient Optimization and Relatives. arXiv preprint arXiv:1505.04243v1. https://arxiv.org/abs/1505.04243v1