Optimización sin restricciones: Algoritmos

Objetivo: encontrar un valor x suficientemente cerca de satisfacer las condiciones de optimalidad.

La mayoría de los métodos son iterativos y descendientes, es decir, obtienen puntos $x_0, x_1, x_2, \ldots$ tales que

$$f(x_{k+1}) < f(x_k), \quad k = 0,1,2, \ldots$$

Métodos de descenso: procedimiento iterativo básico

$$x_{k+1} = x_k + \alpha_k p_k,$$

Donde p_k es la dirección que se busca y α_k es la longitud del paso que se da.

Optimización sin restricciones: Algoritmos

- Sea un punto inicial, para comenzar a iterar.
- Se calculan direcciones descendientes de búsqueda p_k.
- Si estamos lejos de la solución, calculamos la longitud de paso $\alpha_k > 0$.
- El punto inicial se mueve:

$$x_{k+1} = x_k + \alpha_k p_k$$

Hasta que converge a una solución local.

Importante: lo deseable sería tener convergencia a una solución local desde cualquier punto x_0.

Optimización sin restricciones: Algoritmos

Métodos de descenso

1. Elegir $x_0 \in \mathbb{R}^n$, $k = 0$.
2. Si $\nabla f(x_k) = 0$, PARAR.
3. Test de convergencia: (por ejemplo $\|\nabla f(x_k)\| < \epsilon$, ϵ pequeño dado)
   - si se verifica, PARAR
   - si no se verifica, continuar
4. Elegir una dirección de descenso d_k.
5. Elegir un paso $t_k > 0$ verificando $f(x_k + t_k d_k) < f(x_k)$
6. Hacer $x_{k+1} = x_k + t_k d_k$, $k = k + 1$ e ir a 2

Optimización sin restricciones: Algoritmos

La dirección es descendiente si: $\nabla f(x_k)p_k < 0$

Método del gradiente

En este caso,

$$p_k = -\nabla f(x_k).$$

Este es el método más simple y con menos coste computacional.

Pero, **en la práctica, converge demasiado lento.**

Optimización sin restricciones: Algoritmos

Método de Newton

Es el método más conocido y **converge muy rápido,** pero el coste computacional asociado puede llegar a ser muy alto (puesto que hay que calcular y almacenar el hessiano).

En este caso,

$$p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k),$$

siempre que $\nabla^2 f(x_k)$ sea no singular.

Optimización sin restricciones: Algoritmos

Método de Cuasi-Newton

Comenzamos con B_0 (aproximación inicial a $H_0 = \nabla^2 f(x_0)$) y entonces

$$B_{k+1} = B_k + regla\ de\ actualización$$

La dirección sería:

$$d_k = -B_k^{-1} \nabla f(x_k).$$

Optimización sin restricciones: Algoritmos

Método de Cuasi-Newton

Existen dos versiones para la regla de actualización:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$ (simétrica de rango 1)

$$B_{k+1} = B_k - \frac{(B_k s_k)(B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$ (BFGS)

donde $s_k = x_{k+1} - x_k$ e $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

Optimización sin restricciones: Algoritmos

Método de Cuasi-Newton

Ventajas

- No es tan caro como el método de Newton: B_k se calcula usando sólo primeras derivadas (no son necesarias segundas derivadas).
- El sistema lineal correspondiente se resuelve en $O(n^2)$ operaciones (versus $O(n^3)$ en el método de Newton) .
- Convergencia rápida.

Optimización sin restricciones: Algoritmos

La longitud de paso (*steplength*)

Ahora sabemos cómo calcular p_k. Pero, ¿cómo podemos elegir el otro elemento básico para el algoritmo?

Es decir, cómo elegir α_k , con $x_{k+1} = x_k + \alpha_k p_k,$ donde p_k es una dirección descendiente.

Elección óptima

$$\alpha_k \equiv \min_{\alpha \geq 0} \; f(x_k + \alpha p_k)$$

La mejor, pero demasiado costosa de calcular en cada iteración.

Optimización sin restricciones: Algoritmos

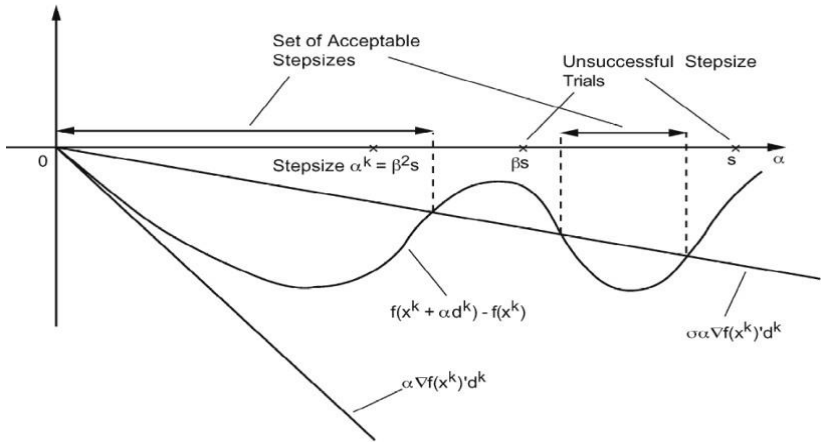
Alternativas

- Regla de Wolfe (demasiado costosa, ya que debe calcular el gradiente para cada prueba de longitud de paso)
- Regla de Armijo
- Se empieza con $\alpha = s, s > 0,$ y la regla continúa con $\alpha = \beta s, \alpha = \beta^2 s, \ldots, (0 < \beta < 1),$ hasta que α cumple:

$$f(x_k + \alpha p_k) \leq f(x_k) + \sigma \alpha p_k^T \nabla f(x_k),$$

para algún $0 < \sigma < 1$.

- Normalmente, $\sigma \in [10^{-5}, 10^{-1}], \beta \in \left[\frac{1}{10}, \frac{1}{2}\right], s = 1.$



3 | Ejemplos

Problemas de regresión

Mínimos cuadrados

$$minimize_{\beta} \frac{1}{2} \sum_{i} \left(y_i - x_i^T \beta\right)^2$$

Solución explícita: $\beta^* = (X^T X)^{-1} X^T y$

Mínimos cuadrados no lineales

$$minimize_{\beta} \ \frac{1}{2} \sum_{i} (y_i - F_i(\beta, x_i))^2$$

No tiene solución explícita: problema no lineal sin restricciones

Problemas de regresión

Regresión Ridge

Cuando el problema tiene demasiadas variables, es conveniente regularizar

$$minimize_{\beta} \ \frac{1}{2}\left|\left|y - X\beta\right|\right|_2^2 + \rho\left|\left|\beta\right|\right|_2^2$$

Solución explícita: $\beta^* = (X^T X + \rho I)^{-1} X^T y$

Regresión Lasso

$$minimize_{\beta} \ \frac{1}{2}\left|\left|y - X\beta\right|\right|_2^2 + \rho\left|\left|\beta\right|\right|_1$$

No tiene solución explícita: problema no lineal sin restricciones

Optimización de portfolios

Desarrollado por Harry **Markowitz** en la década de 1950: se basa en la idea de que un inversor racional siempre buscará maximizar sus retornos y asumir el menor riesgo posible, lo cual se puede lograr a través de la **diversificación** de carteras y principalmente al elegir activos que tengan correlaciones bajas o negativas.

Optimización de portfolios

Desarrollado por Harry **Markowitz** en la década de 1950: se basa en la idea de que un inversor racional siempre buscará maximizar sus retornos y asumir el menor riesgo posible, lo cual se puede lograr a través de la **diversificación** de carteras y principalmente al elegir activos que tengan correlaciones bajas o negativas.

Un inversor tiene *n* activos disponibles donde puede invertir su dinero (comprar y retener). Markowitz mide el riesgo a partir de la varianza de los retornos de los distintos activos del portfolio.

$$\begin{aligned} \text{maximize}_x \quad & \mu^T x - \frac{1}{2}\,\gamma\, x^T \Sigma x \\ \text{subject to} \quad & \sum x_i = 1, \end{aligned}$$

donde $\mu, \sum$ son la esperanza y varianza de los retornos, respectivamente y γ es la aversión al riesgo.

Optimización de portfolios

Desarrollado por Harry **Markowitz** en la década de 1950: se basa en la idea de que un inversor racional siempre buscará maximizar sus retornos y asumir el menor riesgo posible, lo cual se puede lograr a través de la **diversificación** de carteras y principalmente al elegir activos que tengan correlaciones bajas o negativas.

Un inversor tiene *n* activos disponibles donde puede invertir su dinero (comprar y retener). Markowitz mide el riesgo a partir de la varianza de los retornos de los distintos activos del portfolio.

$$\begin{aligned} \text{maximize}_x \quad & \mu^T x - \frac{1}{2}\,\gamma\, x^T \Sigma x \\ \text{subject to} \quad & \sum x_i = 1, \end{aligned}$$

donde $\mu, \sum$ son la esperanza y varianza de los retornos, respectivamente y γ es la aversión al riesgo.

Este modelo permite construir la llamada frontera eficiente (formada por los puntos con menor nivel de riesgo, para un retorno dado).

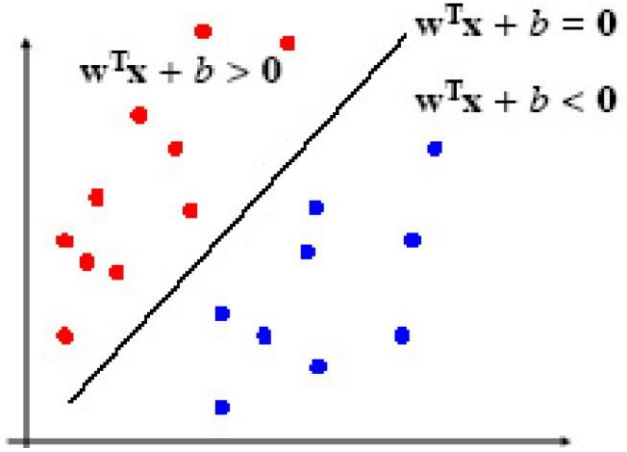
Support Vector Machines

Las SVM se usan en clasificación, para separar clases.

Se usan, por ejemplo en algoritmos para detectar riesgo de crédito y algoritmos de *credit scoring,* algoritmos de reconocimiento facial, filtros de *spam…*

Pongamos como ejemplo la clasificación binaria.
Según los valores de las variables de entrada, definimos dos conjuntos de puntos: buenos clientes y malos clientes.

Cuando llega un cliente nuevo ¿cómo lo clasificamos?



Support Vector Machines

- En este caso concreto, debemos encontrar un clasificador lineal, pero hay infinitos… ¿cuál es el mejor?

- La distancia desde un punto x al clasificador lineal se calcula como:

$$r = \frac{w^T x + b}{||w||}$$

Los vectores soporte son los puntos que están más cerca del clasificador.

Support Vector Machines

- En este caso concreto, debemos encontrar un clasificador lineal, pero hay infinitos… ¿cuál es el mejor?

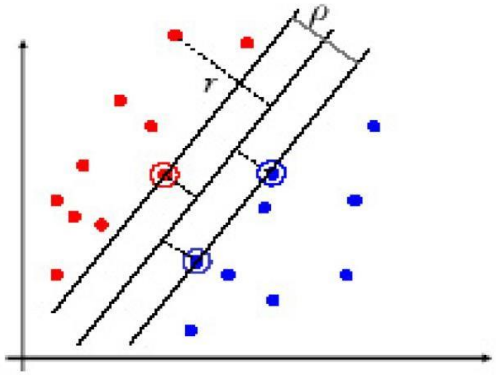
- La distancia desde un punto x al clasificador lineal se calcula como:

$$r = \frac{w^T x + b}{||w||}$$

Los vectores soporte son los puntos que están más cerca del clasificador.

- El margen ρ asociado al clasificador es la distancia (ancho de la banda) entre las dos clases.

Objetivo: **maximizar** el margen.



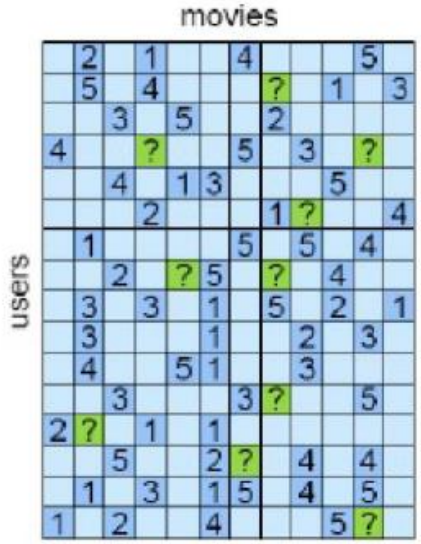
Completación de matrices

- **The Netflix challenge**: predecir el rating (1-5 estrellas) de cada usuario para cada película.

- Basándose en ratings previos, donde algunos usuarios calificaron películas, que podían haber visto o no.

- Usar la predicción para recomendar a los usuarios las películas que les gustaría ver.

Completación de matrices

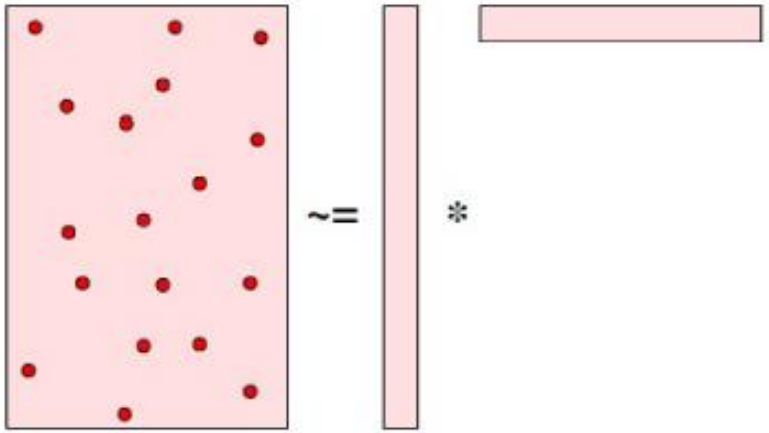


- ¿Al usuario i le gustará la película j?
- **Objetivo**: completar la matriz basada en la información rellena parcialmente.

Netflix proporcionó un data set de entrenamiento con 100.480.507 puntuaciones, que 480.189 usuarios habían dado a 17.770 películas.

Completación de matrices

- La matriz incompleta, Y, aproximada por una matriz de rango bajo (que representa, por ejemplo, el valor cómico de la película, dramatismo, violencia ...).



Completación de matrices

- Por tanto, debemos encontrar la matrix $X \in \mathbb{R}^{mxn}$ con un subconjunto de entradas (M) y con el rango más pequeño:

$$\min_X rango(X)$$
$$s.a. X_{ij} = M_{ij},\ (i,j) \in I$$

- Es un problema **NP-hard,** esto quiere decir que no admite soluciones exactas eficientes.
- Se puede aproximar el rango por:

$$\left|\left|X\right|\right|_* = suma\ de\ los\ valores\ singulares.$$

- Por tanto, lo convertimos en un problema convexo:

$$\min_X \left|\left|X\right|\right|_* + \rho \sum_{i,j \in I} \left(X_{ij} - M_{ij}\right)^2$$

4 | Algoritmos para Big Data

Optimización convexa para Big Data

- Aumento de la importancia de formulación convexa → nuevos modelos de aprendizaje estadístico.
- Internet, problemas de texto, problemas de imágenes, meteorología → gran volumen de datos (terabytes o exabytes).
- Algoritmos clásicos → problema de insolubilidad
- Nuevos enfoques basados en aleatorización de métodos de primer orden y explotación de computación paralela y distribuida.

Optimización convexa para Big Data

- La formulación “estándar” o clásica de los algoritmos de optimización no es apropiada cuando hablamos de gran volumen de datos, ya que algunas de las operaciones son costosas: evaluaciones de gradiente, Hessianos…

- En este contexto, en ocasiones, no se necesita minimizar exactamente la función objetivo, ya que podría llevar a *overfitting.* Basta con una aproximación de la solución.

- Los algoritmos más usados son:

    - Gradiente acelerado
    - Gradiente estocástico
    - Descenso coordinado
    - Método de Newton y cuasi-Newton

Métodos de primer orden

Funciones objetivos regulares (*smooth)*

- Consideramos el problema:

$$\min_{\{x \in \mathbb{R}^p\}} f(x)$$

donde $f(x)$ es convexa y *Lipshitz* continua, es decir,

$$\forall x, y \in \mathbb{R}^p, \|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$$

para alguna constante L.

- Entonces, el método del gradiente

$$x^{k+1} = x^k - \alpha_k \nabla f(x)$$

necesita $O\left(\frac{1}{\epsilon}\right)$ iteraciones para una precisión ϵ. Los métodos *Newton-like* necesitan menos iteraciones pero tienen un mayor coste (calcular el Hessiano).

Métodos de primer orden

Método gradiente acelerado de Nesterov.

Algoritmo

$$1: \; x^{k+1} = v^k - \alpha_k \nabla f(v^k)$$
$$2: \; v^{k+1} = x^{k+1} + \beta_k (x^{k+1} - x^k)$$

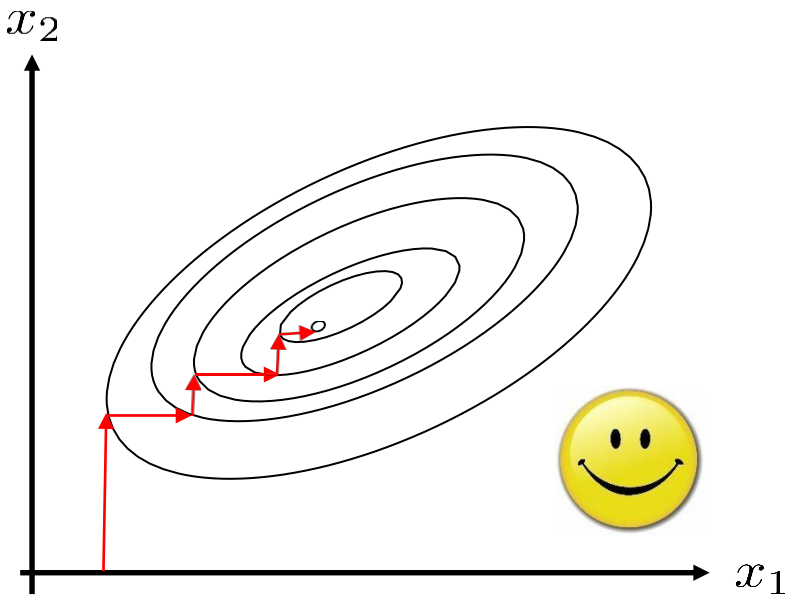
donde $\alpha_k = 1/L$ y $\beta_k = \frac{k}{k+3}$.

- Se alcanza convergencia óptima (la mejor tasa de error posible en el peor de los casos).

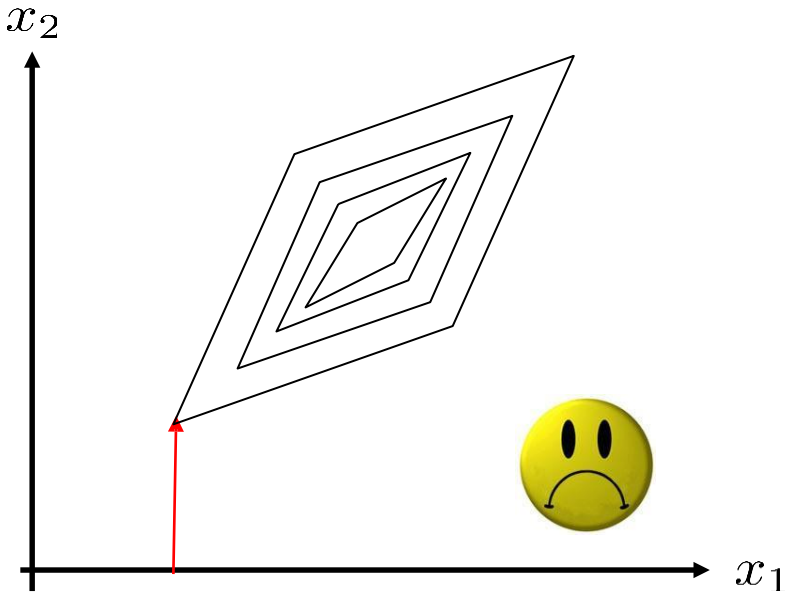
Métodos de primer orden → Aleatorización

Método del descenso coordinado

Se optimiza cada componente de x al mismo tiempo.



Funciones Smooth



Funciones Non-smooth

Métodos de primer orden → Aleatorización

Método del descenso coordinado

- Sólo se modifica x_i para mejorar la función objetivo.

Algoritmo

1. Elegir un índice $i_k \in \{1,2, \dots p\}$
2. $x^{k+1} = x^k - \alpha \nabla_{i_k} f(x^k) e_{i_k}$

donde e_i es la i^{th} coordenada del vector canónico y $\nabla_i f(x)$ es la la i^{th} coordenada del gradiente.

- ¿Cómo seleccionar i?
  - El que tiene mayor $\nabla_i f(x)$ (alto coste computacional).
  - Recorrer todas las coordenadas (convergencia lenta).

Métodos de primer orden → Aleatorización

Método del gradiente estocástico

- Útil para funciones objetivos que se pueden descomponer:

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad f(x) = \frac{1}{n}\sum_{j=1}^{n} f_j(x)$$

Algoritmo

1. Elegir un índice $j_k \in \{1,2,\ldots p\}$ **aleatorio**
2. $x^{k+1} = x^k - \alpha_k \nabla_{j_k} f\left(x^k\right)$

Buena relación coste/beneficio.

Útil para problemas de regresión logística, SVM, *Deep Learning*…

5 Referencias

Referencias

- Bubeck, S., Lee, Y. T., and Singh, M. *A geometric alternative to Nesterov's accelerated gradient descent.* Microsoft Research, 2015.
- Chen, Y. and Wainwright, M. J. *Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees*. University of California-Berkeley, 2015.
- R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.
- Wright, Stephen. *Optimization Methods for Machine Learning*. University of Wisconsin-Madison, 2017.
- D.G. LUENBERGER : *Programación lineal y no lineal*. Addison-Wesley Iberoamericana, 1989.
- Vitoriano, B. y Ramos, A. *Programación matemática: modelos de optimización.* Universidad complutense de Madrid, 2019.