Memoria Práctica 1

Álvaro Fernández García

Marzo, 2018

---

1 Ejercicio sobre la búsqueda iterativa de óptimos

1. Implementar el algoritmo de gradiente descendente.

   El algoritmo se encuentra implementado y comentado en el script.

2. Considerar la función $E(u,v) = (u^3e^{(v-2)} - 4v^3e^{-u})^2$. Usar gradiente descendente para encontrar un mínimo de esta función, comenzando desde el punto $(u,v) = (1,1)$ y usando una tasa de aprendizaje $\eta = 0.05$.

   (a) Calcular analíticamente y mostrar la expresión del gradiente de la función $E(u,v)$:

   La función E se trata de una función polinómica, en cuyo interior nos encontramos términos polinómicos mezclados con exponenciales. Para derivarla, debemos derivar el polinomio principal y multiplicar a continuación por la derivada de lo contenido en el paréntesis principal (a lo que denotaremos como "a"). Nos quedará una forma del tipo $2 \cdot (a) \cdot a'$, donde a' representa la derivada de "a". Por otro lado, como los términos exponenciales presentes tienen por base el número e, su derivada coincide, solo habría que multiplicarla por la derivada del exponente.

   Las derivadas parciales quedarían de la siguiente forma:

   - Con respecto a u:

     $\frac{dE(u,v)}{du} = 2 \cdot (u^3e^{(v-2)} - 4v^3e^{-u}) \cdot (3e^{(v-2)}u^2 + 4v^3e^{-u})$

   - Con respecto a v:

     $\frac{dE(u,v)}{dv} = 2 \cdot (u^3e^{(v-2)} - 4v^3e^{-u}) \cdot (u^3e^{(v-2)} - 12v^2e^{-u})$

   (b) ¿Cuántas iteraciones tarda el algoritmo en obtener por primera vez un valor de $E(u,v)$ inferior a 10^{-14} ? (Usar flotantes de 64 bits)

   El algoritmo necesita un total de 37 interaciones.

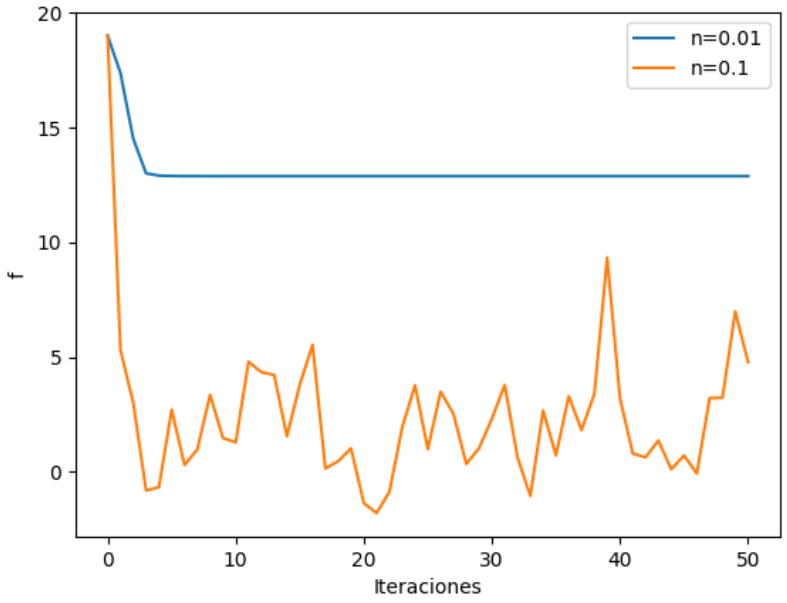
   (c) ¿En qué coordenadas (u, v) se alcanzó por primera vez un valor igual o menor a 10^{-14} en el apartado anterior.

   Se alcanzó por primera vez para un valor de u = 1.1195 y de v = 0.6539. El valor alcanzado es $E(u,v) = 8.7952 \cdot 10^{-15}$

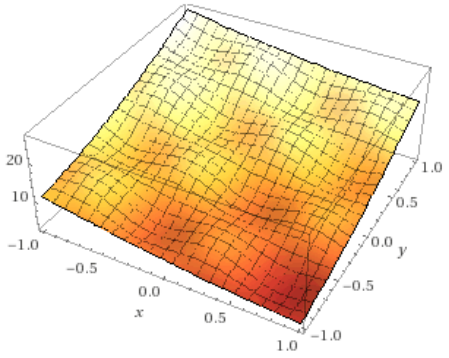
3. Considerar ahora la función: $f(x,y) = (x-2)^2 + 2(y+2)^2 + 2sin(2\pi x)sin(2\pi y)$.

   (a) Usar gradiente descendente para minimizar esta función. Usar como punto inicial $(x_0 = 1, y_0 = 1)$, tasa de aprendizaje $\eta = 0.01$ y un máximo de 50 iteraciones. Generar un gráfico de cómo desciende el valor de la función con las iteraciones. Repetir el experimento pero usando $\eta = 0.1$, comentar las diferencias y su dependencia de η.

   El gráfico pedido en el enunciado es el siguiente:



Para poder justificar como varía el valor de f conforme avanzan las iteraciones, aquí se muestra una gráfica de la función nombrada en el enunciado para un intervalo que va desde -1 a 1 tanto en x como en y:



Como podemos ver, la función está llena de valles y colinas.

La tasa de aprendizaje determina la "amplitud de los saltos" que va realizando el gradiente a través de la función conforme avanza en su búsqueda de un mínimo. En el caso en el que la tasa es muy pequeña (0.01), los saltos que realiza son muy cortos. Al principio vemos que disminuye bruscamente, pero a partir de ahí, entorno a 12.8, el valor se mantiene prácticamente invariable. Se encuentra en un valle, pero como los saltos son tan cortos, apenas se observa variación.

Por otro lado, cuando la tasa es un valor relativamente grande (0.1), los pasos que realiza son muy amplios. Al ser tan amplios comienza a dar saltos entre los valles y colinas, saltándose incluso los óptimos locales. Esto es lo que provoca las fluctuaciones observadas en el gráfico.

(b) Obtener el valor mínimo y los valores de las variables (x, y) en donde se alcanzan cuando el punto de inicio se fija: (2.1, -2.1), (3, -3), (1.5, 1.5), (1, -1). Generar una tabla con los valores obtenidos:

x_0	y_0	x_{min}	y_{min}	f
2.1	-2.1	2.2888	-2.1481	-1.4295
3	-3	2.2753	-2.2507	-1.7730
1.5	1.5	2.7348	-1.7267	-1.2804
1	-1	1.6987	-1.8053	-1.6170

4. ¿Cuál sería su conclusión sobre la verdadera dificultad de encontrar el mínimo global de una función arbitraria?

   Uno de los principales problemas que tiene el algoritmo del gradiente descendente reside en que para funciones que no son convexas no se garantiza que se encuentre el óptimo. Por tanto, en estas situaciones, es de vital importancia una correcta elección de sus parámetros. Y con parámetros me refiero a los siguientes:

   - El punto de inicio: Ya se ha comprobado en el apartado anterior que dependiendo del punto en el que empecemos, se pueden obtener mejores o peores resultados: puede que estemos cerca de un óptimo (ya sea local o global) y lo encontremos en pocos pasos, o puede que estemos muy lejos y no lo encontremos.
   - La tasa de aprendizaje: Este tiene una gran importancia, ya que como hemos mencionado anteriormente, determina la amplitud de los "saltos" que realizaremos en la función para encontrar el óptimo. Una tasa baja puede ralentizar mucho la búsqueda; una tasa alta puede hacer que nos saltemos los óptimos.

   La combinación de lo anterior ya puede dar más de un quebradero de cabeza. Si eso lo unimos a que también hay que determinar la condición de parada, y que aún así se nos sigue sin garantizar que se encuentre el óptimo global, no hace más que complicar las cosas.

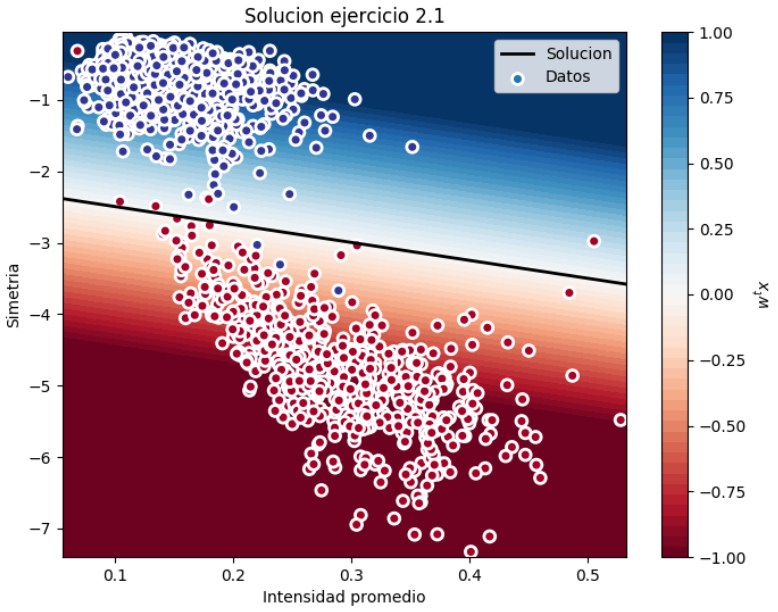
   Estos son los principales motivos por los que el algoritmo del Gradiente Descendente no es tan bueno, como a priori puede parece. Para funciones convexas es de los mejores. Para las demás funciones no tanto.

2 Ejercicio sobre Regresión Lineal

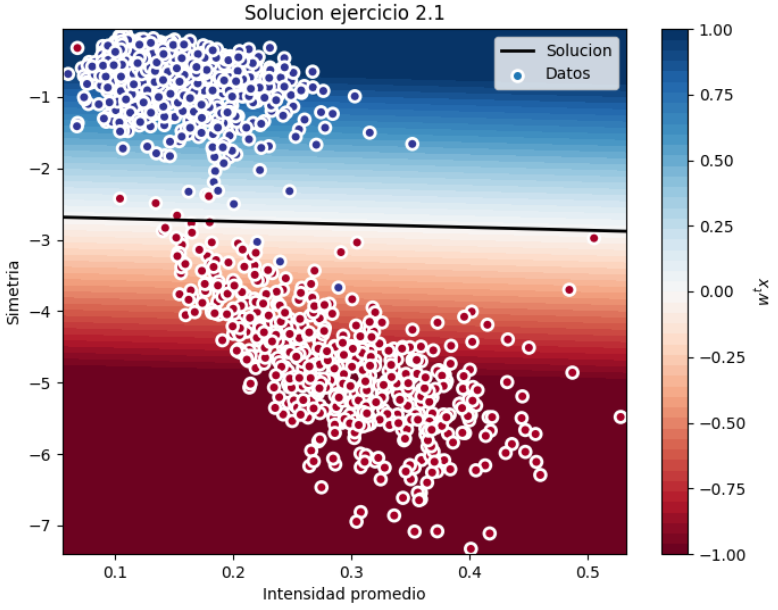
Este ejercicio ajusta modelos de regresión a vectores de características extraidos de imágenes de digitos manuscritos. En particular se extraen dos característcas concretas: el valor medio del nivel de gris y simetría del número respecto de su eje vertical. Solo se seleccionarán para este ejercicio las imágenes de los números 1 y 5.

1. Estimar un modelo de regresión lineal a partir de los datos proporcionados de dichos números (Intensidad promedio, Simetria) usando tanto el algoritmo de la pseudo inversa como Gradiente descendente estocástico (SGD). Las etiquetas serán $\{-1, 1\}$, una para cada vector de cada uno de los números. Pintar las soluciones obtenidas junto con los datos usados en el ajuste. Valorar la bondad del resultado usando E_{in} y E_{out} (para E_{out} calcular las predicciones usando los datos del fichero de test). (usar Regress_Lin(datos, label) como llamada para la función (opcional)).

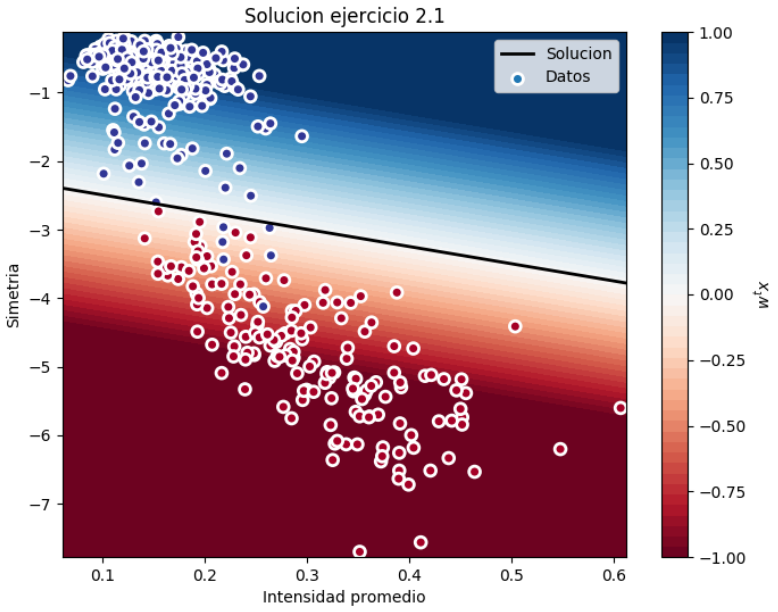
   - Los algoritmos se encuentran implementados y comentados en el script.
   - Pesos obtenidos por la pseudo-inversa: [1.11588016, 1.24859546, 0.49753165]
   - Pesos obtenido por el gradiente descendente estocastico: [1.24097524, 0.19295413, 0.46639675]



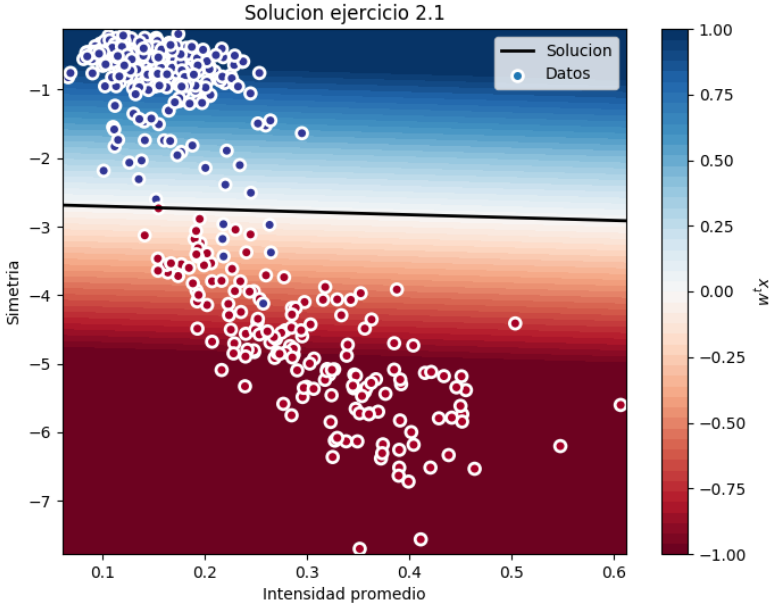
Ajuste de la pseudoinversa y datos de Training.



Ajuste del gradiente descendente estocástico y datos de Training.



Ajuste de la pseudoinversa y datos de Test.



Ajuste del gradiente descendente estocástico y datos de Test.

Con respecto a los errores, se tiene lo siguiente:

- Para la Pseudo-inversa:
  - $E_{in} = 0.07918658628900388$
  - Predicciones de -1 erroneas para los datos de train: 5
  - Predicciones de 1 erroneas para los datos de train: 3
  - Total de errores en los datos de train: 8 de 1561 muestras

- $E_{out} = 0.13095383720052575$
- Predicciones de -1 erroneas para los datos de test: 0
- Predicciones de 1 erroneas para los datos de test: 7
- Total de errores en los datos de test: 7 de 424 muestras

- Para el Gradiente Descendente Estocástico (con un $\eta = 0.01$, 128 para el tamaño del batch y 100 iteraciones):
  - $E_{in} = 0.0820707712293483$
  - Predicciones de -1 erroneas para los datos de train: 5
  - Predicciones de 1 erroneas para los datos de train: 3
  - Total de errores en los datos de train: 8 de 1561 muestras
  - $E_{out} = 0.13791659850084104$
  - Predicciones de -1 erroneas para los datos de test: 0
  - Predicciones de 1 erroneas para los datos de test: 7
  - Total de errores en los datos de test: 7 de 424 muestras

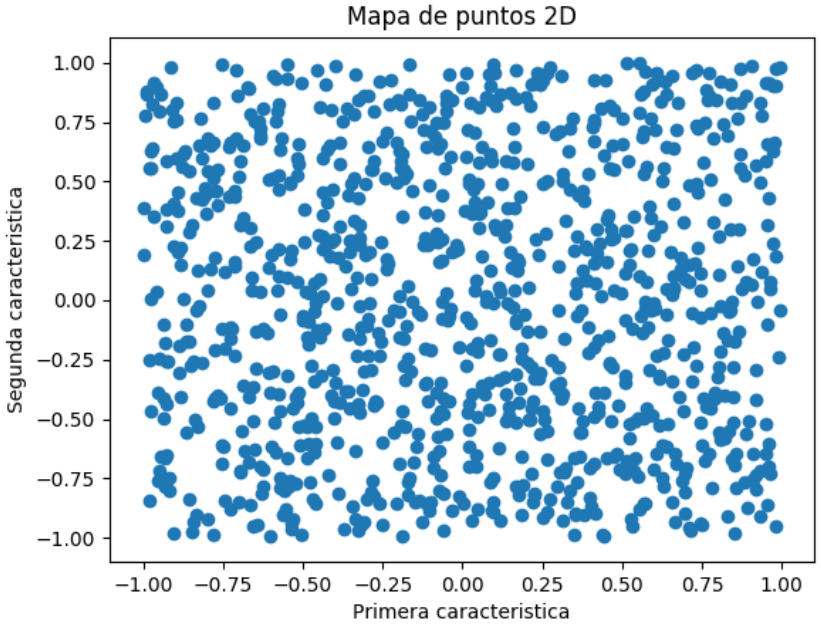
Con respecto a la bondad de los resultados, los ajustes está bastante bien. Ambos ajustes, tanto el de la pseudoinversa, como el del gradiente descendente estocástico son muy similares, teniendo ambos un E_{in} de aproximadamente 0.08 y tan solo 8 errores de 1561 muestras. Como era de esperar, el E_{out} es mayor que el E_{in} siendo este de 0.14 aproximadamente y 7 errores de 424 muestras, lo que sigue siendo bastante aceptable. Por último, si observamos la gráfica, vemos que ambos algoritmos consiguen establecer una línea que separe ambos conjuntos de forma razonable, tanto en el Training como en el Test.

2. En este apartado exploramos como se transforman los errores E_{in} y E_{out} cuando aumentamos la complejidad del modelo lineal usado. Ahora hacemos uso de la función simula_unif (N, 2, size) que nos devuelve N coordenadas 2D de puntos uniformemente muestreados dentro del cuadrado definido por $[-size, size] \times [-size, size]$

   EXPERIMENTO:

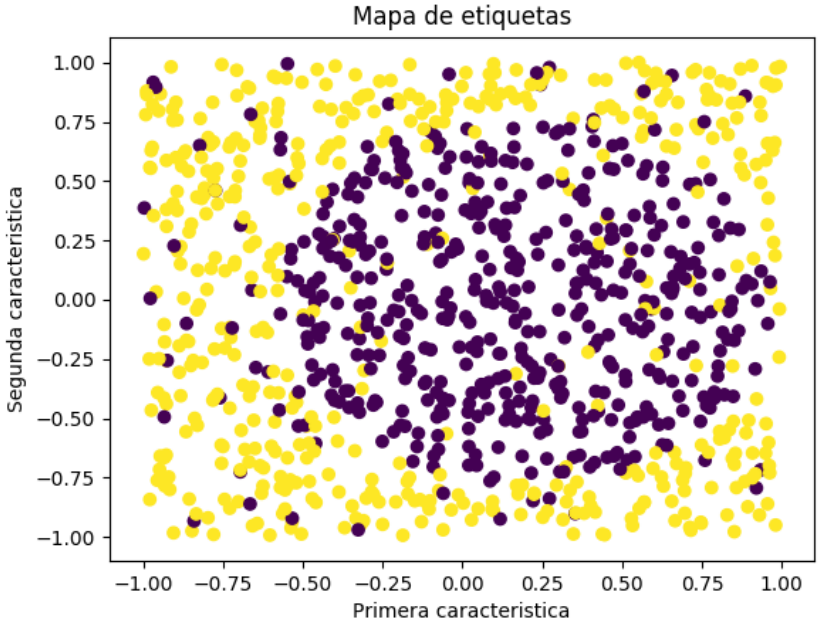
   (a) Generar una muestra de entrenamiento de N = 1000 puntos en el cuadrado $X = [-1, 1] \times [-1, 1]$. Pintar el mapa de puntos 2D. (ver función de ayuda)

   Mapa de puntos 2D obtenido para la muestra:



(b) Consideremos la función $f(x_1, x_2) = sign((x_1 - 0,2)^2 + x_2^2 - 0,6)$ que usaremos para asignar una etiqueta a cada punto de la muestra anterior. Introducimos ruido sobre las etiquetas cambiando aleatoriamente el signo de un 10 mismas. Pintar el mapa de etiquetas obtenido.

Mapa de etiquetas resultante:



(c) Usando como vector de características $(1, x_1, x_2)$ ajustar un modelo de regresion lineal al conjunto de datos generado y estimar los pesos w. Estimar el error de ajuste E_{in} usando Gradiente Descendente Estocástico (SGD).

- Estimacion de los pesos utilizando Gradiente Descendente Estocástico (con un $\eta = 0.01$, 128 para el tamaño del batch y 100 iteraciones): [-0.00168855, -0.38003511, 0.05285513]

- E_{in} obtenido: 0.9529354347469904

(d) Ejecutar todo el experimento definido por (a)-(c) 1000 veces (generamos 1000 muestras diferentes) y

- Calcular el valor medio de los errores E_{in} de las 1000 muestras.
- Generar 1000 puntos nuevos por cada iteración y calcular con ellos el valor de E_{out} en dicha iteración. Calcular el valor medio de E_{out} en todas las iteraciones.

  Aquí se muestra un resúmen de los errores obtenidos en los 1000 experimentos (con un $\eta = 0.01$, 128 para el tamaño del batch y 30 iteraciones):

  - Ein medio: 0.9279539076662311
  - Media de predicciones erróneas de -1 en el train: 219.094
  - Media de predicciones erróneas de 1 en el train: 180.561
  - Media de errores totales en el train: 399.655
  - Eout medio: 0.932549959355549
  - Media de predicciones erróneas de -1 en el test: 220.51
  - Media de predicciones erróneas de 1 en el test: 182.054
  - Media de errores totales en el test: 402.5639

(e) Valore que tan bueno considera que es el ajuste con este modelo lineal a la vista de los valores medios obtenidos de E_{in} y E_{out}.

Observando los errores obtenidos tanto en el ajuste como en el test utilizando gradiente descendente estocástico, se pueder afirmar que la regresión ha sido bastante satisfactoria.

Quizás no es un valor de error lo suficientemente pequeño, pero un error prácticamente de uno tanto en E_{in} como en E_{out} parece razonable. Sin embargo no debemos olvidar que nuestro problema real se trata de un problema de clasificación y que el error obtenido es el derivado de realizar el ajuste con un modelo de regresión lineal, (no está considerando las etiquetas, está considerando valores reales entre -1 y 1).

El ajuste pierde la razonabilidad cuando se asigna una etiqueta a la predicción realizada. (Nota: para asignar las etiquetas hemos considerado el umbral en 0). Como podemos ver, tanto en el ajuste como en el test se producen de media unos 400 errores (prácticamente se equivoca un 40%). No son unos resultados muy alentadores, sin embargo tiene un motivo. Si miramos el mapa de etiquetas dibujado en ejercicios anteriores, podemos ver que estos presentan un patrón circular (al menos dentro de la muestra). Por mucho que queramos, no vamos a poder hacer una separación de ambas clases utilizando una línea. Una de las cosas que podemos hacer sería buscar una transformación del espacio de datos de la muestra que hiciese a los datos linealmente separables, (por ejemplo, utilizando los cuadrados de las características).