Ejercicio 1

Parte a

Se tiene que $u(\theta) = E_{X,y}\left[\|X\theta - y\|^2\right]$. La función $f(\theta)$ corresponde a la derivada de $u(\theta)$, es decir

$$f(\theta) = \nabla_\theta u(\theta)$$
$$\Rightarrow f(\theta) = \nabla_\theta E_{X,y}\left[\|X\theta - y\|^2\right]$$

Como la esperanza y la derivada son lineales es posible intercambiar el orden en el que estas se aplican

$$f(\theta) = E_{X,y}\left[\nabla_\theta(\|X\theta - y\|^2)\right]$$

Recordando que $f(\theta) = E_{X,y}\left[F(\theta, X, y)\right]$, se tiene que en este caso:

$$F(\theta, X, y) = \nabla_\theta\left(\|X\theta - y\|^2\right)$$

Aplicando regla de la cadena vectorial, se tiene que el gradiente anterior es

$$\boxed{F(\theta, X, y) = 2X^T\left(X\theta - y\right)}$$

$$\Rightarrow F(\theta, X, y) = 2X^TX\theta - 2X^Ty$$

Además, por la letra del ejercicio se cumple que $y = X\theta_0 + w$. Entonces, se sustituye esto en la ecuación anterior

$$\Rightarrow F(\theta, X, y) = 2X^TX\theta - 2X^T\left(X\theta_0 + w\right)$$
$$F(\theta, X, y) = 2X^TX\theta - 2X^TX\theta_0 + 2X^Tw \tag{1}$$

En este punto, tomaremos esperanza de la ecuación anterior, para ello primero calculamos $E[X^TX]$ y $E[X^Tw]$.

- $$E[X^TX] = E\left[(A+N)^T(A+N)\right]$$
  $$\Rightarrow E[X^TX] = E\left[A^TA\right] + E\left[A^TN\right] + E\left[N^TA\right] + E\left[N^TN\right]$$

  Como A es deterministico y N de media 0, la esperanza del producto es 0. Por lo tanto

  $$E[X^TX] = A^TA + E[N^TN] = A^TA + 2\sigma^2I$$

- $E[X^TW] = 0$, por tratarse del producto matricial de dos variables aleatorias independientes de media nula.

Sustituyendo lo anterior en 1, se llega a que

$$\boxed{f(\theta) = E[F(\theta, X, y)] = 2(A^TA + 2\sigma^2I)\theta - 2(A^TA + 2\sigma^2I)\theta_0} \tag{2}$$

Parte b

En la parte anterior vimos que $F(\theta, X, y) = 2X^T(X\theta - y)$. Por lo tanto, si se aplica descenso por gradiente estocástico, la regla de actualización sera

$$\boxed{\theta_{k+1} = \theta_k - \alpha_k 2{X_k}^T(X_k\theta_k - y_k)} \tag{3}$$

Parte c

En esta parte, se implementa el algoritmo en el archivo *ej1.py*. Para esto realizan $K = 10000$ iteraciones de SGD partiendo desde el punto $(0, 0)$ para distintos tipos de pasos.

A continuación se presenta la trayectoria de θ_k obtenida en cada caso

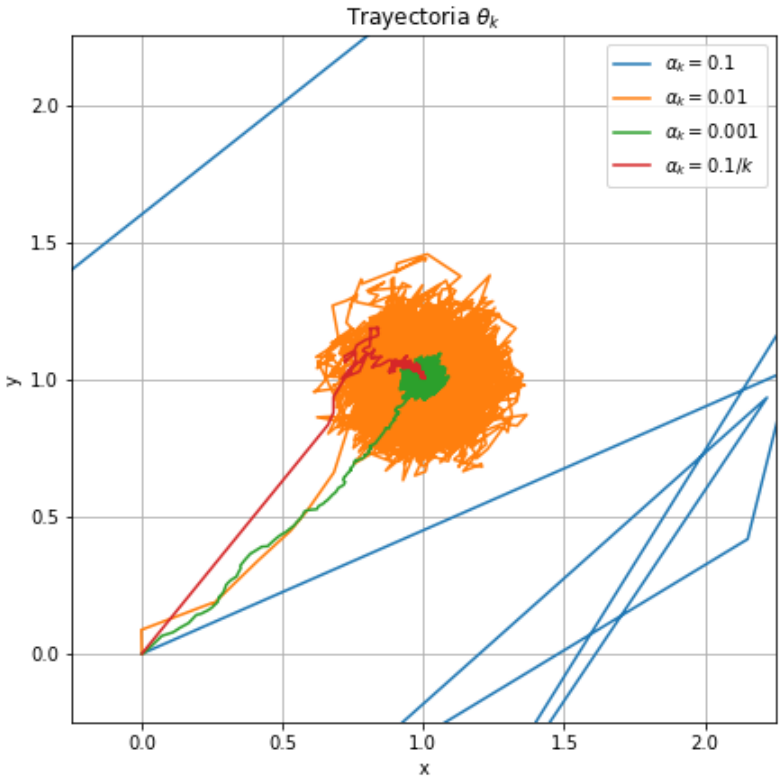

Figure 1: Trayectoria obtenida para cada uno de los pasos probados. Las trayectorias siguen la regla de actualización de la Ecuación 3.

Para analizar estos resultados, es útil recordar el resultado que indica que si se cumple la condición A1 (correspondiente a c_1) y A3 (correspondiente a c_2) del teorema de Robbins-Monro, y además el paso es constante entonces

$$e_{k+1} \leq (1 - 2c_2\alpha)^{k+1} e_0 + c_1\alpha^2 \sum_{l=0}^{k} (1 - 2c_2\alpha)^l$$

con $e_k = E_{X,y}\left[(\theta_k - \theta^\star)^2\right]$.

Si se toma limite con $k \to \infty$ este existe si y solo si la serie converge. Por lo tanto se tienen dos casos posibles

- Si $|\alpha| < \frac{1}{2c_2}$ y entonces se cumple que

  $$\liminf_{k\to\infty} e_k \leq \frac{c_1}{2c_2}\alpha$$

  Por lo tanto, e_k este acotado en infinito, indicando que la sucesión de puntos en infinito estará contenida dentro de una bola de radio dado por c_1, c_2 y α.

- Si $|\alpha| \geq \frac{1}{2c_2}$ no tenemos garantías de que e_k este acotado, pudiendo no converger.

Lo anterior indica que ante un paso constante lo suficientemente chico la sucesión de puntos estará contenida en una bola, cuyo radio máximo sera proporcional al tamaño del paso. Experimentalmente se observa que la bola en la cual esta contenida la solución es más grande con pasos más grandes, ver que la bola que encierra a la trayectoria naranja ($\alpha_k = 0.01$) es más grande que la bola que encierra a la trayectoria verde ($\alpha_k = 0.001$).

Para el caso de $\alpha_k = 0.1$ se observa que no existe una bola que encierre a la trayectoria. Esto coincide con la teoría, de que luego de cierto α la trayectoria no tiene porque ser acotada a partir de algún k.

Por otra parte, en este problema con un paso decreciente del tipo $\alpha_k = \frac{c}{k}$ se cumple A2 del Teorema de Robbins-Monro. Esto implica que

$$\liminf_{k\to\infty} e_k = 0$$

Por lo tanto para el paso decreciente $\alpha_k = \frac{0.1}{k}$ debería converger al óptimo. Esto se verifica experimentalmente en la la trayectoria roja de la Figura 1 en la que el SGD converge al punto $(1.0028914, 1.0015467)$, el cual es el óptimo.

Notar que en todos los casos, el descenso hacia el punto óptimo $(1, 1)$ se da en una dirección que no es suave. Esto se debe a la aleatoriedad de en cada paso estar sorteando un X e y. Se concluye que este ruido es propio del método SGD.

Ejercicio 2

Parte a

Se tiene que

$$G(a, x, y) = \left(y - \text{Relu}\left(a^T x\right)\right)^2$$

Notar que G es derivable en todo punto salvo cuando $a^T x = 0$. Calculemos el gradiente de G en las regiones que sea derivable.

$$\nabla_a G(a, x, y) = 2\left(y - \text{Relu}\left(a^T x\right)\right) \frac{\partial}{\partial a}\left(-\text{Relu}(a^T x)\right)$$

Por otro lado se tiene que $\frac{\partial}{\partial a}\left(\text{Relu}(a^T x)\right)$ tiene la expresión siguiente:

$$\frac{\partial}{\partial a}\left(\text{Relu}(a^T x)\right) = \frac{\partial}{\partial a}\begin{cases}\epsilon a^T x & \text{si } a^T x < 0 \\ a^T x & \text{si } a^T x > 0\end{cases}$$

$$\Rightarrow \frac{\partial}{\partial a}\left(\text{Relu}(a^T x)\right) = \begin{cases}\epsilon x & \text{si } a^T x < 0 \\ x & \text{si } a^T x > 0\end{cases}$$

Por lo tanto, se tiene que el gradiente es

$$\boxed{\nabla_a G(a, x, y) = \begin{cases}-2\left(y - \text{Relu}\left(a^T x\right)\right)\epsilon x & \text{si } a^T x < 0 \\ -2\left(y - \text{Relu}\left(a^T x\right)\right) x & \text{si } a^T x > 0\end{cases}} \tag{4}$$

El subgradiente de G coincide con el gradiente anterior salvo para $a^T x = 0$, ya que el gradiente no esta definido mientras que el subgradiente si. De todas formas, en el campo del aprendizaje profundo no es de interés el calculo del subgradiente en el punto donde la Relu no es derivable, por lo tanto en la implementación si $a^T x = 0$ se escoge arbitrariamente el segundo de los casos.

Parte b

Notar que $F(a, x, y) = \nabla_a G(a, x, y)$. Por lo tanto la iteración del algoritmo SGD sera

$$a_{k+1} = a_k - \alpha_k F(a_k, x_k, y_k)$$

$$\boxed{a_{k+1} = a_k - 2\alpha_k\left(\text{Relu}\left({a_k}^T x_k\right) - y_k\right) x_k \begin{cases}\epsilon & \text{si } {a_k}^T x_k < 0 \\ 1 & \text{si } {a_k}^T x_k > 0\end{cases}} \tag{5}$$

Parte c

El código correspondiente a esta parte se puede observar en *ej2.py*.

En primer lugar, al probar con los parámetros por defecto e inicializando a de forma aleatoria, no se obtenían resultados correctos. Por lo tanto, para esta parte se realizaron algunas modificaciones para obtener un correcto resultado sobre validación:

- Se inicializo a como el vector nulo.
- Se decidió aumentar la cantidad de épocas. Es decir, se utilizan varias pasadas sobre el conjunto de entrenamiento para entrenar nuestra neurona. En particular se utilizaron 100 épocas (4000 ejemplos de entrenamiento).

Se fijo como umbral de decisión el valor 0.5, es decir si $a^T x > 0.5$ se clasifica como conejo y en caso contrario como gato.

Parte d

A continuación, se presenta el vector a que el algoritmo aprende.

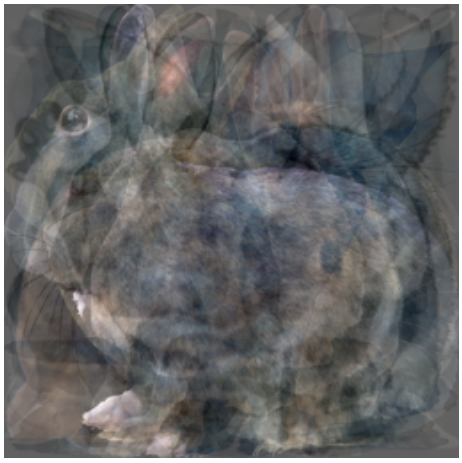

Figure 2: Vector **a** obtenido luego de entrenar el algoritmo

En esta imagen se observa que el algoritmo aprende una combinación lineal de los ejemplos de entrada. Esto ya se podía esperar a partir de la Ecuación 5. Notar que lo que hace el algoritmo es una correlación entre este vector y la imagen que se quiere clasificar.

Además, en las Figuras 3-4 se presentan las gráficas de ${a_K}^T x$ para los conjuntos de train y validación. En las mismas se observa el valor de $a^T x$ para todo x junto con la región de decisión, que es la linea vertical azul sobre el valor 0.5,

indicando que si se esta a la derecha de esta se clasifica como conejo, mientras que si se esta a la izquierda como gato.

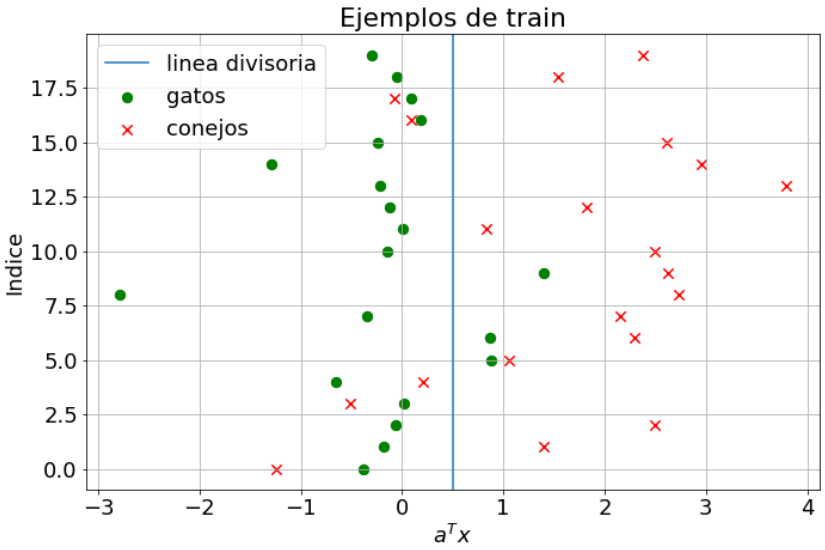

Figure 3: Gráfica de ${a_K}^T x$ contra el numero de muestra para train

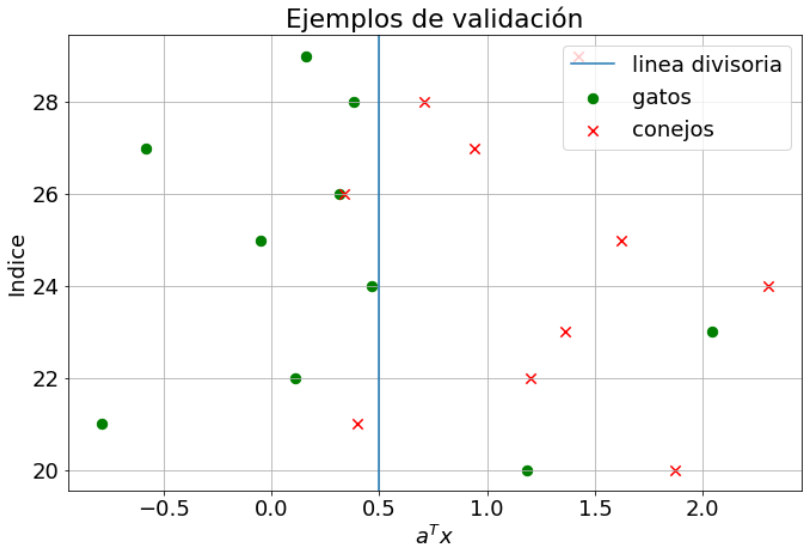

Figure 4: Gráfica de ${a_K}^T x$ contra el numero de muestra para validación

En las figuras anteriores se observa, que en la mayoría de los casos, tanto para train como validación, los gatos y conejos se encuentran en la región correcta (son clasificados correctamente). Notar que el algoritmo es entrenado para que se prediga un 1 en caso de que sea conejo y un 0 en caso de gato, sin embargo los valores predichos para conejos no parecen estar en torno de 1. De todas formas, este extraño comportamiento no impide que se logre cualitativamente un buen desempeño.

Como se menciono, el algoritmo tiene una buena performance por más de su sencillez. Numéricamente se tienen los siguientes resultados:

- 32 ejemplos entre 40 correctamente (accuracy de 80%) en train.
- 16 ejemplos entre 20 correctamente (accuracy de 80%) en validación.

A continuación, se muestran ejemplos de los casos sobre validación que se obtuvieron resultados de clasificación incorrectos.

(a) (b)

Figure 5: Ejemplos de gatos mal clasificados como conejos.

(a) (b)

Figure 6: Ejemplos de conejos mal clasificados como gatos.

El porque clasifica mal los ejemplos anteriores resulta difícil de explicar. Una de las posibles causas podría ser la baja cantidad de ejemplos de entrenamiento. También, el utilizar una única neurona implica que el modelo sea muy sencillo y no tenga la capacidad de aprender la distribución de los datos.