Trabajo práctico 1 – Análisis multivariado
UNaB 2023 cuat. 1

G. Sebastián Pedersen

sebastian.pedersen@unab.edu.ar

Jueves 04 de Mayo de 2023

Notas:

1. Ponderación de este TP respecto a la nota final: 40 %.
2. Entregar las resoluciones de las partes teóricas en un único archivo `pdf` con nombre `tp1_teo_AM_apellido_nombre.pdf`.
3. Entregar las resoluciones de las partes de programación en un único archivo `zip` con nombre `tp1_prog_AM_apellido_nombre.zip`. El archivo `zip` debe tener únicamente dos carpetas conteniendo:
   - Una carpeta `src` conteniendo los scripts de Python. Los scripts de Python deben contener únicamente código y comentarios.
   - Otra carpeta `output` conteniendo las salidas en `txt` y los gráficos en `png`.
4. Entregar por el campus.
5. Fecha y hora límite de entrega: Dom 25/Jun/2023 23:59 hs. (GMT-3).
6. Es válido entregar aunque no se haya resuelto todo el TP.
7. Sea lo más conciso y claro que pueda.
8. El TP es de entrega individual.
9. **El cumplimiento de las pautas de entrega es parte de la resolución del TP.**

1. **Clasificadores lineales (regresión logística y análisis del discriminante gaussiano)**

   En este problema trabajamos con dos clasificadores lineales probabilísticos. Primero, un clasificador lineal discriminativo: regresión logística. En segundo lugar, un clasificador lineal generativo: análisis discriminante gaussiano (GDA). Ambos modelos predictivos encuentran un borde de decisión lineal que separa los datos en dos clases, pero hace suposiciones diferentes. Nuestro objetivo en este problema es obtener una comprensión más profunda de las similitudes y diferencias (y fortalezas y debilidades) de estos dos modelos predictivos.

   Para este problema consideraremos dos conjuntos de datos, provistos en los siguientes archivos:

   (I) `data/ds1_{train,valid}.csv`

   (II) `data/ds2_{train,valid}.csv`

   Cada archivo contiene m ejemplos, un ejemplo $(x^{(i)}, y^{(i)})$ por fila. En particular, la i-ésima fila contiene columnas $x_0^{(i)} \in \mathbb{R}$, $x_1^{(i)} \in \mathbb{R}$ e $y^{(i)} \in \mathbb{R}$. En los subproblemas que siguen, investigaremos utilizando regresión logística y análisis discriminante gaussiano (GDA) la clasificación binaria en estos dos conjuntos de datos.

   a) En clase vimos que la pérdida empírica promedio para la regresión logística es:

   $$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta\left(x^{(i)}\right)\right)$$

   donde $y^{(i)} \in \{0, 1\}$, $h_\theta(x) = g(\theta^t x)$ y $g(z) = 1/(1 + \mathrm{e}^{-z})$. Encuentre el Hessiano H de esta función, y pruebe que es semi definido positivo.

   Sugerencia: puede servir empezar mostrando que $\sum_i \sum_j z_i x_i x_j z_j = (x^t z)^2 \geq 0$. Recordar también que $g'(z) = g(z)(1 - g(z))$

   Comentario: como ya vimos en el TP0, esta es una de las formas estándar de mostrar que la matriz H es semi positiva, y lo notamos como $H \succeq 0$. Esto implica que J es convexo y por lo tanto no tiene mínimos locales, y solamente tiene un mínimo global. Si tiene alguna otra forma de mostrar que $H \succeq 0$, es bienvenide a hacerlo como más le guste.

   b) **Programación.** Siga las instrucciones en `src/p01b_logreg.py` para entrenar un clasificador de regresión logística utilizando el método de Newton. Comenzando con $\theta = 0$, ejecute el método de Newton hasta que las actualizaciones de θ sean pequeñas: específicamente, entrenar hasta la primera iteración k tal que $|\theta_k - \theta_{k-1}| < \epsilon$, donde $\epsilon = 10^{-5}$. Asegúrese de escribir las predicciones de su modelo en el archivo especificado en el código.

   c) Recordar que en GDA modelamos la distribución conjunta de (x, y) de la siguiente manera:

   $$p(y) = \begin{cases} \phi & \text{si } y = 1 \\ 1 - \phi & \text{si } y = 0 \end{cases}$$

   $$p(x|y=0) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_0)^t \Sigma^{-1}(x - \mu_0)\right)$$

   $$p(x|y=1) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_1)^t \Sigma^{-1}(x - \mu_1)\right)$$

   donde ϕ, μ_0, μ_1 y Σ son los parámetros del modelo.

   Supongamos que ya ajustamos ϕ, μ_0, μ_1 y Σ, y ahora queremos predecir y dado un nuevo x. Para mostrar que GDA da como resultado un clasificador que tiene un límite de decisión lineal, muestre que la distribución posterior se puede escribir como:

   $$p(y = 1|x; \phi, \mu_0, \mu_1, \Sigma) = \frac{1}{1 + \exp(-(\theta^t x + \theta_0))}$$

   donde $\theta \in \mathbb{R}^n$ y $\theta_0 \in \mathbb{R}$ son funciones de ϕ, μ_0, μ_1 y Σ.

d) Para esta parte del problema puede suponer que n (la dimensión de x) es 1, por lo que que $\Sigma = [\sigma^2]$ es solo un número real, y por lo tanto el determinante de Σ está dado por $|\Sigma| = \sigma^2$. Dado el conjunto de datos, afirmamos que las estimaciones de máxima verosimilitud de los parámetros son dadas por:

$$\phi = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m} \left(x^{(i)} - \mu_{y^{(i)}}\right)\left(x^{(i)} - \mu_{y^{(i)}}\right)^t$$

El logaritmo de la verosimilitud de los datos es:

$$\begin{aligned} l(\phi, \mu_0, \mu_1, \Sigma) &= \log \prod_{i=1}^{m} p\left(x^{(i)}, y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) \\ &= \log \prod_{i=1}^{m} p\left(x^{(i)} | y^{(i)}; \phi, \mu_0, \mu_1, \Sigma\right) p(y^{(i)}; \phi) \end{aligned}$$

Maximizando l con respecto a los cuatro parámetros, demuestre que la máxima verosimilitud las estimaciones de ϕ, μ_0, μ_1 y Σ son las que se dan en las fórmulas anteriores. (Puede suponer que hay al menos un ejemplo positivo y uno negativo, de modo que los denominadores de μ_0 y μ_1 no sean cero).

e) **Programación**. En `src/p01e_gda.py`, complete el código para calcular ϕ, μ_0, μ_1 y Σ, use estos parámetros para derivar θ y use el modelo GDA resultante para hacer predicciones sobre el conjunto de test.

f) Para el conjunto de datos 1, cree un gráfico de los datos de entrenamiento con x_1 en el eje horizontal y x_2 en el eje vertical. Para visualizar las dos clases, use un símbolo diferente para los ejemplos $x^{(i)}$ con $y^{(i)} = 0$ que para aquellos con $y^{(i)} = 1$. En la misma figura, trace el borde de decisión encontrado por regresión logística en la parte $1b$. Hacer un gráfico idéntico con el borde de decisión encontrado por GDA en la parte $1e$.

g) Repita los pasos de la parte $1f$ para el conjunto de datos 2. ¿En qué conjunto de datos parece que GDA funciona peor que regresión logística? ¿Por qué podría ser este el caso?

h) Para el conjunto de datos en el que GDA se desempeñó peor, ¿puede encontrar una transformación de las x tal que GDA funcione significativamente mejor? ¿Cómo es esta transformación y por qué ayuda a mejor el funcionamiento de GDA?

2. **Etiquetas sólo de los positivos e incompletas.** En este problema consideraremos entrenar clasificadores binarios en situaciones donde no tenemos acceso total a las etiquetas. En particular, consideramos un escenario, que no es demasiado infrecuente en la realidad, donde tenemos etiquetas sólo para un subconjunto de los ejemplos positivos. Todos los ejemplos negativos y el resto de los ejemplos positivos no están etiquetados.

   Es decir, asumimos un conjunto de datos $\{(x^{(i)}, t^{(i)}, y^{(i)})\}_{i=1}^m$, donde $t^{(i)} \in \{0, 1\}$ es la etiqueta verdadera, y donde:

$$y(i) = \begin{cases} 1 & \text{si } x(i) \text{ está etiquetado} \\ 0 & \text{si no.} \end{cases}$$

   Todos los ejemplos etiquetados son positivos, es decir, $p(t^{(i)} = 1|y^{(i)} = 1) = 1$, pero los ejemplos no etiquetados pueden ser positivos o negativos. Nuestro objetivo en el problema es construir un clasificador binario h de la etiqueta verdadera t, sólo teniendo acceso a las etiquetas parciales y. En otras palabras, queremos construir h tal que $h(x^{(i)}) \sim p(t^{(i)} = 1|x^{(i)})$ sea lo más aproximado posible, usando sólo x e y.

   Ejemplo del mundo real: supongamos que mantenemos una base de datos de proteínas que están involucradas en la transmisión de señales a través de las membranas. Cada ejemplo agregado a la base de datos está involucrado en una proceso de señalización, pero hay muchas proteínas involucradas en la señalización a través de la membrana que faltan de la base de datos. Sería útil entrenar a un clasificador para identificar proteínas que deberían ser añadidas a la base de datos. En nuestra notación, cada ejemplo $x^{(i)}$ corresponde a una proteína, $y^{(i)} = 1$ si la proteína está en la base de datos y 0 en caso contrario, y $t^{(i)} = 1$ si la proteína está involucrada en un proceso de señalización a través de la membrana y por lo tanto debe agregarse a la base de datos, y 0 de lo contrario.

   a) Suponga que cada $y^{(i)}$ y $x^{(i)}$ son condicionalmente independientes dado $t^{(i)}$:

$$p(y^{(i)} = 1|t^{(i)} = 1, x^{(i)}) = p(y^{(i)} = 1|t^{(i)} = 1)$$

   Notar que esto es equivalente a decir que los ejemplos etiquetados se seleccionaron uniformemente al azar del conjunto de ejemplos positivos. Demuestre que la probabilidad de que un ejemplo sea etiquetado difiere por un factor constante de la probabilidad de que un ejemplo sea positivo. Es decir, demostrar que $p(t^{(i)} = 1|x^{(i)}) = p(y^{(i)} = 1|x^{(i)})/\alpha$ para alguna constante α.

   b) Supongamos que queremos estimar α usando un clasificador entrenado h y un conjunto de validación V. Sea V_+ el conjunto de ejemplos etiquetados (y por lo tanto positivos) en V, dado por $V_+ = \{x^{(i)} \in V|y^{(i)} = 1\}$. Suponiendo que $h(x^{(i)}) \simeq p(y^{(i)} = 1|x^{(i)})$ para todos los ejemplos $x^{(i)}$, mostrar que:

$$h(x^{(i)}) \simeq \alpha \quad \text{para todo } x^{(i)} \in V_+$$

   Puede suponer que $p(t^{(i)} = 1|x^{(i)}) \simeq 1$ cuando $x^{(i)} \in V_+$.

   c) **Programación**. Los siguientes tres problemas tratarán con conjuntos de datos que se proporcionan en los siguientes archivos:

   `datos/ds3_{train,test,valid}.csv`

   Cada archivo contiene las siguientes columnas: x_1, x_2, y y t. Como en el problema 1, hay un ejemplo por fila. Primero consideraremos el caso ideal, donde tenemos acceso a las etiquetas t verdaderas para el entrenamiento. En `src/p02cde_posonly.py`, escriba un clasificador de regresión logística que use x_1 y x_2 como entrada, y entrénelo usando las etiquetas t (puede ignorar las etiquetas y para esta parte). Escriba las predicciones del modelo sobre el conjunto de test en el archivo especificado en el código.

   d) **Programación**. Ahora consideramos el caso en el que las etiquetas t no están disponibles, por lo que sólo se tiene acceso a las etiquetas y en el momento del entrenamiento. Agregue a su código en `p02cde_posonly.py` para volver a entrenar el clasificador (todavía usando x_1 y x_2 como características de entrada), pero usando las etiquetas y solamente.

e) **Programación**. Usando el conjunto de validación, estime la constante α por promedio de las predicciones de su clasificador sobre todos los ejemplos etiquetados en el conjunto de validación:

$$\alpha \simeq \frac{1}{|V_+|} \sum_{x^{(i)} \in V_+} h(x^{(i)})$$

Agregue a su código en `src/p02cde_posonly.py` para cambiar la escala de las predicciones de su clasificador de la parte $2c$ usando el valor estimado para α.

Finalmente, usando un umbral de $p(t^{(i)} = 1|x^{(i)}) = 0.5$, haga tres gráficos separados con los bordes de decisión de las partes $2c$, $2d$ y $2e$, sobre el conjunto de test. Grafique x_1 en el eje horizontal y x_2 en el eje vertical, y use dos símbolos diferentes para el positivo ($t^{(i)} = 1$) y negativo ($t^{(i)} = 0$). En cada gráfico, indique el hiperplano de separación con una línea roja.

Comentario: vimos que la verdadera probabilidad $p(t|x)$ estaba sólo a un factor constante de distancia de $p(y|x)$. Esto significa que, si nuestra tarea es solo rankear los ejemplos (es decir, ordenarlos) en un orden particular (p. ej., clasificar las proteínas en el orden en que es más probable que participen en la transmisión de señales a través de membranas), entonces de hecho ni siquiera necesitamos estimar α. El ranking basado en $p(y|x)$ será el mismo que el basado en $p(t|x)$.

3. **Regresión Poisson.** Considere la distribución de Poisson parametrizada por λ:

$$p(y;\lambda) = \frac{\mathrm{e}^{-\lambda}\lambda^{y}}{y!}$$

a) Muestre que la distribución de Poisson está en la familia exponencial y establezca claramente los valores para $b(y)$, η, $T(y)$ y $a(\eta)$.

b) Considere realizar una regresión utilizando un modelo GLM con una respuesta y de Poisson. ¿Cuál es la función de respuesta canónica para la familia? (Puede usar el hecho que una variable aleatoria de Poisson con parámetro λ tiene esperanza λ).

c) Para un conjunto de entrenamiento $\{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$, sea el log de la verosimilitud de un ejemplo $\log p(y^{(i)}|x^{(i)};\theta)$. Tomando la derivada del logaritmo de la verosimilitud con respecto a θ_j, obtenga la regla de actualización de ascenso por gradiente estocástico para aprender los parámetros de un modelo GLM con una respuesta y de Poisson y con la función de respuesta canónica.

d) **Programación**. Considere un sitio web que quiere predecir su tráfico diario. Los propietarios del sitio web han recopilado un conjunto de datos del tráfico pasado de su sitio web, junto con algunas características que creen que son útiles para predecir el número de visitantes por día. Los conjuntos de datos están en los siguientes archivos:

`data/ds4_{train,valid}.csv`

Aplicaremos la regresión de Poisson para modelar el número de visitantes por día. Tenga en cuenta que aplicar regresión de Poisson en particular supone que los datos siguen una distribución de Poisson cuyo parámetro natural es una combinación lineal de las características de entrada (es decir, $\eta = \theta^t x$). En `src/p03d_poisson.py`, implemente la regresión de Poisson para este conjunto de datos y use ascenso por gradiente para maximizar el logaritmo de la verosimilitud de θ.

4. **Convexidad de Modelos Lineales Generalizados.** En este problema exploraremos y mostraremos algunas lindas e interesantes propiedades de los modelos lineales generalizados, específicamente aquellos relacionados con su uso de distribuciones de Familia Exponencial para modelar la salida. Más comúnmente, los GLM se entrenan utilizando el logaritmo de verosimilitud negativa (NLL) como la función de pérdida. Esto es matemáticamente equivalente a la Estimación de Máxima Verosimilitud (es decir, maximizar el logaritmo de la verosimilitud es equivalente a minimizar el negativo del logaritmo de la verosimilitud). En este problema, nuestro objetivo es mostrar que la pérdida de NLL de un GLM es una función convexa con respecto a los parámetros del modelo. Como recordatorio, esto es conveniente porque una función convexa es aquella para la cual cualquier mínimo local es también un mínimo global.

   Recordar, una distribución de la familia exponencial es aquella cuya densidad de probabilidad se puede representar como:

$$p(y;\eta) = b(y)\exp(\eta^t T(y) - a(\eta))$$

   donde η es el parámetro natural de la distribución. Además, en un modelo lineal generalizado, η se modela como $\theta^t x$, donde $x \in \mathbb{R}^n$ son las características de entrada del ejemplo, y $\theta \in \mathbb{R}^n$ son los parámetros a aprender. Para mostrar que la pérdida de NLL es convexa para los GLM, desglosamos el proceso en subpartes, y los abordamos de uno a la vez. Nuestro enfoque es mostrar que la segunda derivada (es decir, el Hessiano) de la pérdida con respecto a los parámetros del modelo es semidefinida positiva (SDP) en todos los valores de los parámetros del modelo. También mostraremos algunas buenas propiedades de distribuciones de Familia Exponencial como pasos intermedios.

   Por conveniencia nos limitamos al caso donde η es un escalar. Asumimos que $p(Y|X;\theta) \sim$ Familia Exponencial(η), donde $\eta \in \mathbb{R}$ es un escalar, y $T(y) = y$. Esto hace que la representación de la familia exponencial tome la forma:

$$p(y;\eta) = b(y)\exp(\eta y - a(\eta))$$

   a) Obtenga una expresión para la media de la distribución. Demuestre que $E[Y|X;\theta]$ puede representarse como el gradiente de la función de partición logarítmica a con respecto al parámetro natural η.

   Sugerencia: comience observando que $\frac{\partial}{\partial \eta}\int p(y;\eta)dy = \int \frac{\partial}{\partial \eta} p(y;\eta)dy$.

   b) A continuación, obtenga una expresión para la varianza de la distribución. En particular, demuestre que $Var(Y|X;\eta)$ se puede expresar como la derivada de la media con respecto a η (es decir, la segunda derivada de la función de partición logarítmica $a(\eta)$ con respecto al parámetro natural η).

   c) Finalmente, escriba la función de pérdida $l(\theta)$, el NLL de la distribución, como una función de θ. Luego, calcule el Hessiano de la pérdida con respecto a θ y demuestre que siempre es SDP. Esto concluye la demostración de que la pérdida de NLL de GLM es convexa.

   Sugerencia: use la regla de la cadena junto con los resultados de las partes anteriores para simplificar sus derivaciones.

   Comentario: las conclusiones principales de este problema son:

   - Cualquier modelo GLM es convexo en sus parámetros del modelo.
   - La familia exponencial de distribuciones de probabilidad es matemáticamente linda. Mientras que calcular la media y la varianza de distribuciones en general involucra integrales (difícil), sorprendentemente para familias exponenciales podemos calcular media y varianza usando derivadas (fácil).

5. **Análisis de imágenes usando regresión logística**

En este problema entrenaremos un clasificador lineal (regresión logística) para identificar números escritos a mano en imágenes de 28 por 28 píxeles. Si bien la regresión logística no es un clasificador usado extensivamente para este fin, el objetivo del problema es profundizar en el entendimiento del mencionado clasificador e introducirnos en el problema de identificar figuras y/o patrones en imágenes.

Para este problema consideraremos dos conjuntos de datos, provistos en los siguientes archivos:

(I) `mnist_test.csv`

(II) `mnist_train.csv`

Los datos se encuentran accesibles desde acá:

`https://drive.google.com/drive/folders/1DAoor-RqKL-zJRNu28jYHoX8QsZZXZxT?usp=sharing`

Cada fila en los archivos corresponde a una imagen y cada dato en ella representa un píxel de la misma y su color (en una escala de blanco y negro los valores más chicos representarían colores más oscuros, etc) a excepción de la primera columna, que corresponde al label de la imágen.

a) **Programación.** Programe y entrene un clasificador de regresión logística utilizando descenso por gradiente. Puede usarse de guía el archivo `src/p05_img.py`. Como la regresión logística es un clasificador binario, se clasificarán las imágenes como *1* si dicha imágen contiene un *1* y *0* en caso contrario. Para optimizar el funcionamiento del clasificador, se lo debe entrenar con partes iguales de imágenes con labels *1* y *0*. Se recomienda usar los siguientes híperparámetros:

1) $\epsilon = 1e-5$ (Distancia de corte entre θ^{nuevo} y θ^{viejo})
2) Tasa de aprendizaje $= 5e-06$
3) Cantidad máxima de iteraciones $= 1000$

Utilice métricas que considere adecuadas para evaluar su performance en test y escriba conclusiones. Guarde las predicciones en el archivo `p05_pred1.csv`

b) Entrene 10 modelos distintos, uno para cada label del dataset, y realice predicciones sobre los datos de testeo. Nuevamente, guarde las predicciones en el archivo `p05_predtot.csv`, utilice métricas para evaluar la performance del modelo y realice un análisis de los resultados obtenidos.