Tarea 1: Análisis de sentimientos

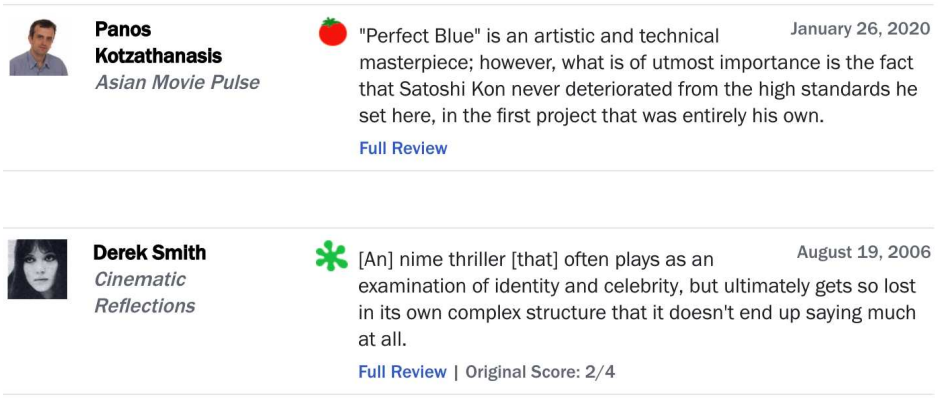
Inteligencia Artificial
Lic. en Ciencias de la Computación
Universidad de Sonora
Semestre 2023-1

Esta tarea consiste de una parte escrita y una parte de programación. Todas tus respuestas deben ser escritas en TEX (o algún derivado), deben aparecen en el mismo orden que las preguntas y estar correctamente etiquetadas. Tus programas deben estar escritos en el lenguaje de programación Python.

Debes cargar tu trabajo al grupo de Microsoft Teams del curso como un archivo comprimido llamado `sentimientos` que contenga tus respuestas en un archivo en formato PDF llamado `sentimientos` y tu código en un archivo de Python llamado `sentimientos`.

1. Construyendo la intuición.

Aquí hay dos reseñas de *Perfect Blue*, del sitio web Rotten Tomatoes:



Rotten Tomatoes ha clasificado estas reseñas como "positivas" y "negativas", respectivamente, como lo indica el tomate fresco en la primer reseña y el tomatazo verde de la segunda. En esta tarea, vas a crear un sistema simple de clasificación de texto que puede realizar esta tarea de forma automática. Vamos a calentar con el siguiente conjunto de cuatro mini-reseñas, cada una etiquetada como positiva $(+1)$ o negativa (-1):

1. (-1) not good
2. (-1) pretty bad
3. $(+1)$ good plot
4. $(+1)$ pretty scenery

Cada reseña x es asociada a un vector de características $\phi(x)$, el cuál asocia cada palabra la cantidad de veces que aparece en la reseña. Por ejemplo, la primer reseña se asocia al vector (disperso) de características

$$\phi(x) = \{\text{not} : 1, \text{good} : 1\}.$$

Recuerda la definición de la pérdida de articulación (hinge loss):

$$\text{Loss}_{\text{hinge}}(x, y, \boldsymbol{w}) = \max\{0, 1 - \boldsymbol{w} \cdot \phi(x)y\},$$

donde x es el texto de la reseña, y es la etiqueta correcta y $\boldsymbol{w}$ es el vector de pesos.

a. Supongamos que corremos descenso de gradiente estocástico una vez por cada una de las cuatro muestras en el orden dado arriba, actualizando los pesos de acuerdo a

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \text{Loss}_{\text{hinge}}(x, y, \boldsymbol{w}).$$

Despues de las actualizaciones, cuáles son los pesos de las seis palabras ("pretty", "good", "bad", "plot", "not", "scenery") que aparecen en las reseñas de arriba?

- Usa $\eta = 0.1$ como tamaño de paso.
- Inicializa $\boldsymbol{w} = [0, 0, 0, 0, 0, 0]$.
- El gradiente $\nabla_{\boldsymbol{w}} \text{Loss}_{\text{hinge}}(x, y, \boldsymbol{w}) = 0$ cuando el margen es exactamente 1.

Expectativa: Un vector de pesos que contiene un valor numérico por cada palabra en las reseñas con el siguiente orden ("pretty", "good", "bad", "plot", "not", "scenery"). Por ejemplo: $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]$.

b. Dado el siguiente conjunto de datos de reseñas:

1. (-1) bad
2. $(+1)$ good
3. $(+1)$ not bad
4. (-1) not good

Muestra que no hay clasificador lineal que utilice características de palabras que tenga cero error sobre este conjunto de datos. Recuerda que esta es una pregunta sobre clasificadores, no sobre algoritmos de optimización; tu demostración debe ser correcta para cualquier clasificador lineal, sin importar en cómo se aprenden los pesos.

Propón una sola característica adicional para tu conjunto de datos con la que pudieramos arreglar este problema.

Expectativa: Una demostración corta (de tres a cinco oraciones) y una característica viable que pudiera permitir a un clasificador lineal tener cero error en el conjnuto de datos (clasificar todos los ejemplos correctamente).

2. Prediciendo calificaciones de películas. Supongamos que ahora estamos interesados en predecir una calificación numérica para reseñas de películas. Vamos a usar un predictor no-lineal que toma una reseña de película x y regresa $\sigma(\boldsymbol{w} \cdot \phi(x))$, donde $\sigma(z) = (1 + e^{-z})^{-1}$ es la función logística que aplasta un número real al rango $(0, 1)$. Para este problema, supón que la calificación de película y es una variable con valor real en el rango $[0, 1]$.

a. Si quisiéramos usar la pérdida cuadrática (squared loss), ¿Cuál sería la expresión para $\text{Loss}(x, y, \boldsymbol{w})$ para un solo punto (x, y).

   Expectativa: Una expresión matemática para la pérdida. Puedes usar σ en la expresión.

b. Considerando $\text{Loss}(x, y, \boldsymbol{w})$ de la anterior parte, calcula el gradiente de la pérdida con respecto a $\boldsymbol{w}$, $\nabla_{\boldsymbol{w}}\text{Loss}(x, y, \boldsymbol{w})$. Escribe tu respuesta en términos del valor predecido $p = \sigma(\boldsymbol{w} \cdot \phi(x))$.

   Expectativa: Una expresión matemática para el gradiente de la pérdida.

c. Si hay un dato (x, y) con algúna $\phi(x)$ arbitraria y $y = 1$. Especifica las condiciones para $\boldsymbol{w}$ para hacer la magnitud del gradiente de la pérdida con respecto a $\boldsymbol{w}$ arbitrariamente pequeño (es decir, minimizar la magnitud del gradiente). ¿Es posible que la magnitud del gradiente con respecto a $\boldsymbol{w}$ sea exactamente cero? Puedes hacer la magnitud de $\boldsymbol{w}$ arbitrariamente grande pero no infinita.

   Nota: Intenta entender intuitivamente qué está ocurriendo y a qué contribuye cada parte de la expresión. Si te piérdes en mucha álgebra, probablemente estés haciendo algo subóptimo.

   Motivación: La razón por la que nos interesa la magnitud de gradientes es porque gobierna el tamaño de la zancada en el descenso de gradiente. Por ejemplo, si el gradiente es cercano a cero, entonces $\boldsymbol{w}$ está muy lejos del óptimo, entonces puede tardar mucho tiempo para que el descenso de gradiente llegue al óptimo (si es que llega). Esto se conoce como el *problema de desvanecimiento de gradiente* cuando entrenamos redes neuronales.

   Expectativa: Un par de oraciones describiendo las condiciones de $\boldsymbol{w}$ para minimizar la magnitud del gradiente y un par de oraciones explicando si el gradiente puede ser exactamente cero.

3. Clasificación de sentimientos. En este problema, vamos a construir un clasificador lineal que lee reseñas de películas y adivina si son "positivos" o "negativos".

a. Implementa la función `extractWordFeatures`, que toma una reseña en forma de cadena como entrada y regresa un vector de características $\phi(x)$, representado como un

diccionario de Python.

b. Implementa la función `learnPredictor` usando descenso de gradiente estocástico y minimiza la pérdida de articulación. Imprime el error de entrenamiento y el error de validación después de cada época para asegurarte que tu código funciona. Debes obtener una tasa de error menor al 4% sobre el conjunto de entrenamiento y menor que 30% en el de validación.

c. Escribe la función `generateExample` para generar muestras de datos artificiales.

Usa esta función para volver a verificar que `learnPredictor` funciona. Puedes hacer esto usando `generateDataset()` para generar ejemplos de entrenamiento y validación. Luego puedes pasar estos ejemplos a `learnPredictor` como los argumentos `trainExamples` y `validationExamples` respectivamente y la función **lambda** x: x como el argumento `featureExtractor`.

d. Algunos lenguajes son escritos sin espacios entre las palabras, entonces ¿es realmente necesario dividir las palabras o podemos ingenuamente considerar cadenas de caracteres que se extienden a lo largo de las palabras? Implementa la función `extractCharacterFeatures`, la cuál asocia cada cadena de n caracteres a la cantidad de veces que aparece, ignorando espacios en blanco.

e. Corre tu predictor lineal con el extractor de características `extractCharacterFeatures`. Experimenta con distintos valores de n para ver cuál produce el menor error de validación. Debes observar que este error es casi tan pequeño como el que se produce con características de palabras. ¿Por qué este es el caso?

Construye una reseña (a lo más una oración) en donde n-gramas de caracteres probablemente sea mejor que características de palabras. Brevemente explica por qué este es el caso.

> **Expectativa:** Un párrafo corto de ($\sim$4–6 oraciones) en donde presentes cuál valor de n produce el error de validación más pequeño y por qué este probablemente produce el error más pequeño. Una reseña de una oración y la explicación para cuando n-gramas de caracteres probablemente sean mejores que características de palabras.

4. Clasificación de toxicidad y pérdida máxima de grupo. Recordemos que los modelos entrenados (de la forma estándar) para minimizar la pérdida promedio funcionan bien en promedio pero mal para ciertos grupos, y que podemos mitigar este problema minimizando la pérdida máxima de grupo. En este problema, vamos a comparar los objetivos de la pérdida promedio y de la pérdida máxima de grupo con un problema de juguete inspirado en un problema real con modelos de clasificación de toxicidad.

Los claisificadores de toxicidad son diseñados para asistir en la moderación de foros en línea prediciendo si un comentario es tóxico o no, de tal forma que los comentarios predecidos como tóxicoss puedan ser marcados para revisión humana. Desafortunadamente, se ha

observado que estos modelos están sesgados: comentarios no-tóxicos que mencionan identidades demográficas suelen ser mal clasificados como tóxicos. Estos sesgos surgen porque comentarios tóxicos usualmente mencionan y atacan identidades demográficas, y resultado de esto, los modelos aprenden a correlacionar falsamente la toxicidad con la mención de estas identidades.

En este problema, vamos a estudiar una situación de juguete para ilustrar el problema de la falsa correlación: La entrada x es un comentario en forma de cadena publicado en un foro de internét; la etiqueta $y \in \{-1, 1\}$ es la toxicidad del comentario ($y = 1$ es tóxico, $y = -1$ es no-tóxico); $d \in \{0, 1\}$ indica si el texto contiene una palabra que se refiere a una identidad demográfica; y $t \in \{0, 1\}$ indica si el comentario incluye algunas palabras "tóxicas". El comentario x es asociado al vector de características $\phi(x) = [1, d, t]$ donde 1 es un término de sesgo (para prevenir el caso de que $\boldsymbol{w} \cdot \phi(x) = 0$ en las preguntas de abajo). Como ejemplo concreto, se proveen algunos comentarios simples abajo, donde se subrayan las palabras tóxicas y las que se refieren a una identidad demográfica:

Comentario (x)	**Toxicidad (y)**	**Presencia de menciones demográficas (d)**	**Presencia de palabras tóxicas (t)**
"Stanford sucks!"	1	0	1
"I'm a woman in computer science!"	-1	1	0
"The hummingbird sucks nectar from the flower"	-1	0	1

Supongamos que se nos proveen los siguientes datos de entrenamiento, donde enlistamos la cantidad de veces que cada combinación (y, d, t) aparece en el conjunto de entrenamiento.

y	d	t	**# datos**
-1	0	0	63
-1	0	1	27
-1	1	0	7
-1	1	1	3
1	0	0	3
1	0	1	7
1	1	0	27
1	1	1	63
		Total	200

De la tabla de arriba, podemos ver que 70 de los 100 comentarios tóxicos incluyen palabras tóxicas, y 70 de los 100 comentarios no-tóxicos no contienen palabras tóxicas.

Adicionalmente, la toxicidad del comentario t está altamente correlacionada con menciones de identidades demográficas d (porque los comentarios tóxicos tienden a dirigirse a ellas) — 90 de los 100 comentarios incluyen menciones de identidades demográficas, y 90 de los 100 comentarios no-tóxicos no las incluyen.

Vamos a considerar clasificadores lineales de la forma

$$f_{\boldsymbol{w}}(x) = \text{sign}(\boldsymbol{w} \cdot \phi(x)),$$

con $\phi(x)$ definida como arriba. Normalmente entrenaríamos los clasificadores para minimizar ya sea la pérdida promedio o la pérdida máxima de grupo, pero por simplicidad, vamos a comparar dos clasificadores lineales (los cuáles pudieran no minimizar ninguno de los objetivos):

- Clasificador D: $\boldsymbol{w} = [-0.1, 1, 0]$
- Clasificador T: $\boldsymbol{w} = [-0.1, 0, 1]$

Para nuestra función de pérdida, vamos a estar usando la pérdida cero-uno, de tal forma que la pérdida por grupo es

$$\text{TrainLoss}_g(\boldsymbol{w}) = \frac{1}{|\mathcal{D}_{\text{train}}(g)|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}(g)} \mathbf{1}[f_{\boldsymbol{w}}(x) \neq y].$$

Recuerda la definición de la pérdida máxima de grupo:

$$\text{TrainLoss}_{\max}(\boldsymbol{w}) = \max_g \text{TrainLoss}_g(\boldsymbol{w}).$$

Para capturar el problema de la falsa correlación, definamos grupos basados en el valor de (y, d). Entonces tenemos cuatro grupos: $(y = 1, d = 1)$, $(y = 1, d = 0)$, $(y = -1, d = 1)$, y $(y = -1, d = 0)$. Por ejemplo, el grupo $(y = -1, d = 1)$ se refiere a comentarios no-tóxicos con menciones a identidades demográficas.

a. En palabras, describe el comportamiento del Claisificador D y el clasificador T.

   Expectativa: Por cada clasificador (D y T) una proposición bicondicional que describe la salida del clasificador en términos de sus características cuando $y = 1$.

b. Calcula las siguientes tres cantidades sobre el Clasificador D usando el conjunto de datos de arriba:

   1. La pérdida promedio del clasificador
   2. La pérdida promedio de cada grupo
   3. La pérdida máxima de grupo del clasificador

> **Expectativa:** Un valor para la pérdida promedio, las cuatro pérdidas promedio de grupo (una por cada grupo) y un valor para la pérdida máxima de grupo.

c. Ahora calcula las siguientes cantidades sobre el Clasificador T usando el mismo conjunto de datos:

   1. La pérdida promedio del clasificador
   2. La pérdida promedio de cada grupo
   3. La pérdida máxima de grupo del clasificador ¿Qué clasificador tiene menor pérdida promedio? ¿Qué clasificador tiene máxima pérdida de grupo?

   > *Nota:* Los grupos siguen siendo definidos por d, la etiqueta de mención demográfica.
   >
   > **Expectativa:** Un valor para la périda promedio, las cuatro pérdidas promedio de grupo (una por cada grupo) y un valor para la pérdida máxima de grupo. Indica qué clasificador tiene menor pérdida promedio, luego indica qué clasificador tiene menor pérdida máxima de grupo.

d. Cómo vimos arriba, distintos clasificadores nos llevan a diferentes números de predicciones precisas y diferentes comentarios de gente incorrectamente rechazadas. La clasificación precisa de un comentario no-tóxico es bueno para el comentarista, pero cuando no existe un clasificador perfecto, ¿Cómo deben distribuirse las clasificaciones correctas entre los comentaristas? Aquí hay tres principios de distribución justa:

   1. Tu elección de algoritmo debe resultar en el mayor beneficio neto (beneficiar a la mayor cantidad de gente con el menor costo posible).
   2. Tu elección de algoritmo debe priorizar el bienestar de la gente más desfavorecida.
   3. Tu elección de algoritmo debe ser tal que no imponga una desventaja en los miembros de grupos que han enfrentado discriminación histórica.

   > *Nota:* Los puntos de arriba son solo un subconjunto de los ajustes éticos existentes. Si sientes que otro principio es más apropiado, puedes utilizarlo.

Primero argumenta por el uso de de la pérdida promedio apelando a uno de los principios de arriba. ¿Cuál de los Clasificadores D o T implementarías en una plataforma web social real para identificar publicaciones etiquetadas como tóxicas para su revisión? Luego, haz lo mismo pero argumentando por el uso de la pérdida máxima de grupo como tu objetivo, igual apelando a uno de los tres principios.

> **Expectativa:** Una explicación breve que justifique usar la pérdida promedio refiriendo a uno de los principios. Luego una explicación breve que justifique usar la pérdida máxima de grupo refiriendo a uno de los

principios. Hay varias maneras de responder esta pregunta, una buena respuesta explica la conexión entre un clasificador y un principio de forma clara y concisa.

e. Hemos discutido sobre el aprendizaje máquina como el proceso de transformar datos en modelos, pero ¿de dónde vienen los datos? En el contexto de recolección de datos para entrar modelos de aprendizaje máquina para clasificación de toxicidad, quien determina si un comentario *debería* ser identificado como tóxico o no es muy importante (es decir, si $y = 1$ o $y = -1$ en la tabla de datos). Aquí hay algunas opciones:

- Recluta personas en una plataforma de *crowdsourcing* (término que viene de "crowd" y "outsourcing") para antoar cada comentario.
- Contrata científicos sociales para establecer criterios de toxicidad y anotar cada comentario.
- Pide a los usuarios de la plataforma que califiquen comentarios.
- Pide a los usuarios de la plataforma que deliberen al respecto y decidan en estándares y criterios de toxicidad para la comunidad, quizá usando un proceso de diseño participativo.

¿Qué métodos usarías para determinar la toxicidad de comentarios para usarlos como datos de entrenamiento con un clasificador de toxicidad? Explica por qué eligirías esos métodos sobre otros enlistados.

Expectativa: Un par de oraciones explicando qué métodos usarías y por qué. Un par de oraciones donde contrastes los métodos elegidos con alternativas.

5. Agrupación con K-medias. Supongamos que tenemos un extractor de características ϕ que produce vectores de características bidimensionales, y que tenemos un conjunto de datos de juguete $\mathcal{D}_{\text{train}} = \{x_1, x_2, x_3, x_4\}$ con,

$$\begin{aligned}\phi(x_1) &= [0, 0] \\ \phi(x_2) &= [4, 0] \\ \phi(x_3) &= [6, 0] \\ \phi(x_4) &= [11, 0]\end{aligned}$$

a. Corre 2-medias sobre este conjunto de datos hasta la convergencia. Muestra tu trabajo. ¿Cuáles fueron las asignaciones finales de agrupamiento z y los centros de agrupamiento μ? Corre este algoritmo dos veces con los siguientes centros centros iniciales:

1. $\mu_1 = \phi(x_1) = [0, 0]$ y $\mu_2 = \phi(x_4) = [11, 0]$
2. $\mu_1 = \phi(x_1) = [0, 0]$ y $\mu_2 = \phi(x_2) = [4, 0]$

> **Expectativa:** Muestra los centros de agrupamiento y asignaciones para cada paso, así como la pérdida final de cada proceso de agrupación.

b. Implementa la función `kmeans`. Debes inicializar tus k centros a elementos aleatorios de `examples`.

Después de unas cuantas iteraciones de k-medias, tus centros van a ser vectores muy densos. Para que tu código corra de forma eficiente debes precalcular algunos productos punto. Puedes encontrar útil la función `generateClusteringExamples` para probar tu código.

c. Si escalamos todas las dimensiones en nuestros centroides iniciales y los datos por un factor distinto a cero. ¿Garantizamos que recuperaremos los mismos agrupamientos después de correr k-medias (es decir, los mismos puntos de datos van a pertenecer al mismo grupo antes y después de escalar)? ¿Qué pasa si escalamos solo ciertas dimensiones? Si tu respuesta es afirmativa, provee una explicación corta, si no, presenta un contraejemplo.

> **Expectativa:** Esta respuesta debe tener dos partes. La primera debe ser una respuesta de si/no y una explicación o contraejemplo que la sustente para la primer pregunta (escalar todas las dimensiones). La segunda parte debe ser una respuesta de si/no y una explicación o contraejemplo para la segunda pregunta (escalar solo ciertas dimensiones).