

Tarea 1

Inteligencia Artificial



Lic. en Ciencias de la Computación

29 de enero de 2024 – 11 de febrero de 2024

El objetivo de esta tarea es poner en práctica tu conocimiento sobre problemas de regresión y clasificación. Considera que una *palabra* es simplemente una cadena de caracteres no vacía sin espacios en blanco y que dos palabras que difieren únicamente en mayúsculas son consideradas distintas.

Instrucciones generales

Esta tarea tiene una parte escrita y una parte programada. Se utilizan los siguientes indicadores visuales:

-  Significa que debes escribir tus respuestas en `tarea1.pdf`.
-  Significa que debes escribir tus respuestas en `tarea1.py`.

Todas las respuestas escritas deben ser tipografiadas en \LaTeX . Se provee una plantilla en el material de ayuda `tarea1.zip`. Tus respuestas deben estar **en orden**, así como **clara y correctamente etiquetadas** para recibir la puntuación correspondiente.

Debes modificar el código de `tarea1.py` entre los marcadores `# INICIO` y `# FIN`. También puedes agregar funciones auxiliares fuera de estos delimitadores si gustas. No se permite cambiar otros archivos además de `tarea1.py`.


Tu código es evaluado con pruebas unitarias, las cuales puedes ver en `califica.py`. Para correr las pruebas, puedes invocar el comando `python califica.py` en la terminal, el cuál te informa si pasaste las pruebas básicas y señala una alerta si tu código tarda demasiado o truena sin verificar la respuesta de las pruebas ocultas.

Se recomienda que leas y entiendas todos los casos de prueba y que agregues tus propios casos durante el desarrollo de la tarea.

Algunos problemas presentes en esta tarea son marcados como '(Opcional)', puedes omitir resolver estos problemas, pero se recomienda que si los presentes si consideras continuar tu formación en el área de inteligencia artificial.

Desarrollando la intuición

Considera las siguientes dos reseñas de la película *Perfect Blue*, del sitio [Rotten Tomatoes](#).

| | | | |
|---|--|--|------------------|
|  | Panos Kotzathanasis <i>Asian Movie Pulse</i> |  "Perfect Blue" is an artistic and technical masterpiece; however, what is of utmost importance is the fact that Satoshi Kon never deteriorated from the high standards he set here, in the first project that was entirely his own. Full Review | January 26, 2020 |
|  | Derek Smith <i>Cinematic Reflections</i> |  [An] nime thriller [that] often plays as an examination of identity and celebrity, but ultimately gets so lost in its own complex structure that it doesn't end up saying much at all. Full Review Original Score: 2/4 | August 19, 2006 |


Rotten Tomatoes ha clasificado estas reseñas como “positiva” y “negativa” respectivamente, como se indica por el tomate fresco e intacto en la reseña de arriba y la salpicadura de un tomate podrido en la reseña de abajo. En esta tarea, vas a crear un simple sistema de clasificación de texto que puede realizar esta tarea automáticamente. Vamos a considerar 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 asignada a un vector de características $\phi(x)$, el cuál asocia cada palabra a la cantidad de veces que aparece en la reseña. Por ejemplo, la segunda reseña se asigna al vector disperso de características $\phi(x) = \{\text{pretty} : 1, \text{bad} : 1\}$. Recuerda la definición de la *pérdida de articulación*,

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

donde x es el texto de la reseña, y es la clasificación correcta y \mathbf{w} es el vector de pesos.

1.  Supongamos que corres el descenso de gradiente estocástico una vez por cada una de las cuatro reseñas de arriba en el mismo orden, actualizando los pesos de acuerdo a,

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

Después de las actualizaciones, ¿cuáles son los pesos de las seis palabras (“pretty”, “good”, “bad”, “plot”, “not”, “scenery”) que aparecen en las reseñas?

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

Se espera que respondas con: Un vector de pesos que contiene un valor numérico por cada palabra en las reseñas (“pretty”, “good”, “bad”, “plot”, “not”, “scenery”), en ese orden. Por ejemplo: [0.1, 0.2, 0.3, 0.4, 0.5, 0.6].

2. ✎ (Opcional) Dado el segundo conjunto de datos de reseñas:

- (a) (-1) bad
- (b) (+1) good
- (c) (+1) not bad
- (d) (-1) not good

Muestra que ningún clasificador lineal que utilice conteo de palabras como sus características puede obtener un error cero sobre este conjunto de datos. Recuerda que esta es una pregunta sobre clasificadores, no sobre algoritmos de optimización; tu prueba debe ser cierta para cualquier clasificador lineal de la forma $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$, independientemente de cómo se aprenden los pesos.

Propón una característica adicional para el conjunto de datos con la que podemos ampliar el vector de características que resolvería este problema.

Se espera que respondas con: Una breve demostración y una característica viable que permitiría a un clasificador lineal tener error cero sobre el conjunto de datos.

Prediciendo calificaciones de películas

Supongamos que ahora nos interesa predecir una calificación numérica para reseñas de películas. Utilizaremos un predictor no-lineal que toma una reseña x y regresa $\sigma(\mathbf{w} \cdot \phi(x))$, 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 $[0, 1]$.

1. ✎ Supongamos que queremos usar la *pérdida cuadrática*. Escribe la expresión para $\text{Loss}(x, y, \mathbf{w})$ para un dato (x, y) .

Se espera que respondas con: Una expresión matemática para la pérdida.
Puedes usar σ en la expresión.

2. ✎ Dada la pérdida $\text{Loss}(x, y, \mathbf{w})$ del problema anterior, calcula el gradiente de la pérdida con respecto a \mathbf{w} , $\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$. Escribe la respuesta en términos del valor predicho $p = \sigma(\mathbf{w} \cdot \phi(x))$.

Se espera que respondas con: Una expresión matemática para el gradiente de la pérdida.

3. ✎ (Opcional) Supongamos que hay un dato (x, y) con algún $\phi(x)$ arbitrario y $y = 1$. Especifica las condiciones para \mathbf{w} para hacer que la magnitud del gradiente de la pérdida con respecto a \mathbf{w} sea arbitrariamente pequeño (es decir, que minimice la magnitud del gradiente). ¿Es posible que la magnitud del gradiente con respecto a \mathbf{w} sea exactamente cero? Puedes considerar que la magnitud de \mathbf{w} sea arbitrariamente grande pero no infinita.

Intenta entender intuitivamente qué está pasando y qué contribuye cada parte de la expresión. Si terminas haciendo muchos movimientos algebraicos, probablemente estás explorando una respuesta demasiado larga.

La razón por la que nos interesa la magnitud de los gradientes es porque esta influencia qué tanto ajusta los pesos el descenso de gradiente. Por ejemplo, si el gradiente es cercano a cero cuando \mathbf{w} está muy lejos del óptimo, entonces el descenso de gradiente puede tardar mucho en alcanzar el óptimo (en caso que lo alcance). Esto se conoce como el *problema de desvanecimiento de gradiente* al entrenar redes neuronales.






Se espera que respondas con: Un par de oraciones describiendo las condiciones para \mathbf{w} que minimicen la magnitud del gradiente, un par de oraciones explicando si el gradiente puede ser exactamente cero.

Clasificación de sentimientos

En este problema vamos a construir un clasificador lineal binario que lee reseñas de películas y adivina si son “positivos” o “negativos”.

Lee cuidadosamente el archivo `util.py`, contiene algunas funciones de utilidad que puedes utilizar para implementar tu código. En tus respuestas, no modifi-

ques los diccionarios de Python directamente, mejor utiliza las operaciones de `dotProduct` e `increment`.

1.  Implementa la función `extractWordFeatures`, la cuál toma una reseña (como cadena) de entrada y regresa un vector de características $\phi(x)$ representado como un `dict` en Python.
2.  Implementa la función `learnPredictor` utilizando el descenso de gradiente estocástico y minimizando 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 de a lo más 4% sobre el conjunto de entrenamiento y a lo más 30% sobre el conjunto de validación para que tu implementación sea correcta.
3.  Escribe la función `generateExample` (anidada en la función `generateDataset`) para generar ejemplos de datos artificiales.
Utilízala para doble verificar que tu `learnPredictor` funciona. Puedes hacer esto utilizando `generateDataset()` para generar ejemplos de entrenamiento y validación, puedes luego pasar estos ejemplos como los argumentos `trainExamples` y `validationExamples` respectivamente a la función `learnPredictor` con la función de identidad como `featureExtractor`.
4.  Algunas lenguas son escritas sin espacios entre palabras, entonces ¿es necesario separar las palabras o podemos ingenuamente considerar subcadenas de caracteres a lo largo del texto? Implementa la función `extractCharacterFeatures` (escribiendo la función `extract`), la cuál asocia a cada cadena de n caracteres a la cantidad de veces que aparece, ignorando espacios en blanco (espacios y tabulaciones).
5.  (Opcional) 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 producido con el uso de conteo de palabras como características. ¿Por qué es este el caso?
Construye una reseña (una oración) en donde el conteo de n -gramas probablemente sea mejor que el conteo de palabras y explica brevemente por qué es el caso.

Se espera que respondas con: Un párrafo breve donde describas cuál valor de n produce el menor error de validación, explicando la posible razón por la que este valor produce el menor error. Una reseña de una oración y una explicación para cuando el conteo de n -gramas probablemente es mejor que el conteo de palabras.

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 las funciones objetivo de la pérdida promedio y la pérdida máxima de grupo con un contexto de juguete inspirado en un problema real con modelos de clasificación de toxicidad.

Los clasificadores 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 tóxicos puedan ser marcados para revisión humana. Desafortunadamente, se ha mostrado que algunos modelos [clasifican comentarios no-tóxicos que mencionan identidades demográficas como tóxicos](#). Este comportamiento puede surgir si suponemos que los comentarios tóxicos en los datos mencionan frecuentemente identidades demográficas, y como resultado, los modelos aprenden a correlacionar falsamente la toxicidad con la mención de estas identidades.

En este problema, vamos a estudiar un contexto de juguete que ilustra el problema de la correlación espuria: La entrada x es un comentario (como cadena) realizado en un foro virtual; 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 ciertas palabras “tóxicas”. El comentario x es asignado a un vector de características $\phi(x) = [1, d, t]$ donde 1 es el término de sesgo (para prevenir el caso $w \cdot \phi(x) = 0$ en las preguntas de abajo). Se proveen algunos ejemplos simples abajo, donde se subraya las palabras tóxicas y las que se refieren a una identidad demográfica:

| Comentario (x) | Toxicidad (y) | Menciona identidad (d) | Menciona tóxica (t) |
|--|-------------------|-------------------------------|----------------------------|
| “Hermosillo <u>sucks</u> !” | +1 | 0 | 1 |
| “I’m a <u>woman</u> in computer science” | -1 | 1 | 0 |
| “The hummingbird <u>sucks</u> nectar from the flower” | -1 | 0 | 1 |

Supongamos que se nos proveen los siguientes datos de entrenamiento, resumido con 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 |

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 (bajo el supuesto que los comentarios tóxicos tienden a dirigirse a ellas) — 90 de los 100 comentarios tóxicos incluyen menciones de identidades demográficas, y 90 de los 100 comentarios no-tóxicos no las incluyen.

Vamos a considerar clasificadores lineales de la forma

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

donde se define a $\phi(x)$ 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 : $\mathbf{w} = [-0.1, 1, 0]$
- Clasificador T : $\mathbf{w} = [-0.1, 0, 1]$

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

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

Recuerda la definición de la pérdida máxima de grupo:

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

Para capturar el problema de la correlación espuria, 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.

1. ✎ (Opcional) Describe el comportamiento del clasificador D y del clasificador T .

Se espera que respondas con: Una oración de “si y solo si” describiendo la salida del clasificador en términos de sus características cuando $f_w(x) = 1$.

2. ✎ (Opcional) Calcula las siguientes cantidades del clasificador D usando el conjunto de datos de arriba:

- (a) La pérdida promedio del clasificador D en general
- (b) La pérdida promedio del clasificador D por cada grupo
- (c) La pérdida máxima de grupo del clasificador D

Se espera que respondas con: Un valor para la pérdida promedio, el valor de las pérdidas promedio por cada grupo, un valor de la pérdida máxima de grupo.

3. ✎ (Opcional) Calcula las siguientes cantidades del clasificador T usando el conjunto de datos de arriba:

- (a) La pérdida promedio del clasificador T en general
- (b) La pérdida promedio del clasificador T por cada grupo
- (c) La pérdida máxima de grupo del clasificador T


¿Qué clasificador tiene la menor pérdida promedio? ¿Qué clasificador tiene la menor pérdida máxima de grupo?

Se espera que respondas con: Un valor para la pérdida promedio, el valor de las pérdidas promedio por cada grupo, un valor de la pérdida máxima de grupo. Indica cuál clasificador tiene la menor pérdida promedio y cuál tiene la menor pérdida máxima de grupo.

4. ✎ (Opcional) Diferentes clasificadores nos llevan a diferentes predicciones acertadas y a diferentes comentarios siendo incorrectamente rechazados (falsos positivos). La clasificación correcta de un comentario no-tóxico es bueno para quien escribe el comentario, pero cuando ningún clasificador tiene precisión perfecta, ¿Cómo deben distribuirse las clasificaciones correctas entre los comentaristas?

Recuerda los principios de distribución justa discutidos en clase y con base en ellos: Elabora un argumento para el uso de la pérdida promedio como función objetivo, ¿cuál clasificador (D o T) elegirías para una red social real con la finalidad de identificar publicaciones tóxicas para revisión humana? Luego haz lo mismo pero argumentando para el uso de la pérdida máxima de grupo como función objetivo, igualmente apela a alguno de los principios de distribución justa.

Se espera que respondas con: Una explicación breve que justifique usar la pérdida promedio invocando alguno de los principios. Una explicación breve que justifique usar la pérdida máxima de grupo invocando alguno de los principios. Una buena explicación explica la conexión entre un clasificador y un principio de forma clara y concisa.

5.  El aprendizaje máquina puede pensarse como el proceso de transformar datos en modelos, pero ¿de dónde vienen los datos? En el contexto de recolección de datos para el entrenamiento de los modelos de aprendizaje máquina para la clasificación de toxicidad, quien determina si un comentario *debe* ser identificado como tóxico o no es muy importante (es decir, si $y = 1$ o si $y = -1$ en las tablas de datos de arriba). Algunas de las opciones comúnmente empleadas son:

- Recluta personas en una plataforma de *crowdsourcing* para anotar cada comentario.
- Contratar científicos sociales para establecer criterios de toxicidad y anotar cada comentario.
- Solicitarle a los usuarios de la plataforma que clasifiquen comentarios.
- Solicitarle a los usuarios de la plataforma que deliberen acerca de y decidan sobre estándares comunitarios y criterios de toxicidad, utilizando un proceso de diseño participativo.

¿Qué métodos usarías para determinar la toxicidad de comentarios para usarlos como parte del entrenamiento de un clasificador? ¿Por qué? Explica por qué los métodos que eliges son mejores que los otros enlistados.

Se espera que respondas con: Un par de oraciones explicando qué métodos usarías y por qué. Un par de oraciones contrastando los métodos elegidos con las alternativas.

Agrupamiento con K -medias


Supongamos que tienes un extractor de características ϕ que produce vectores bidimensionales de características, y un conjunto de entrenamiento de juguete $D_{\text{train}} = \{x_1, x_2, x_3, x_4\}$ donde,

$$\phi(x_1) = [0, 0]$$



$$\phi(x_2) = [4, 0]$$

$$\phi(x_3) = [6, 0]$$

$$\phi(x_4) = [11, 0]$$

1.  Corre el método de 2-medias sobre este conjunto de datos hasta llegar a la convergencia. Escribe el desarrollo de tu trabajo. ¿Cuáles son las asignaciones finales z y los centros de los grupos μ ? Corre este algoritmo dos veces con los siguientes centros iniciales:
(a) $\mu_1 = \phi(x_1) = [0, 0]$ y $\mu_2 = \phi(x_4) = [11, 0]$
(b) $\mu_1 = \phi(x_1) = [0, 0]$ y $\mu_2 = \phi(x_2) = [4, 0]$

Se espera que respondas con: Los centros y asignaciones en cada paso del método de 2-medias, así como la pérdida final para cada agrupamiento.

2.  Implementa la función `kmeans`. Después de unas cuantas iteraciones de k -medias, tus centros van a ser vectores muy densos. Para que tu código sea eficiente y tu respuesta a este problema sea válida, deberás precalcular algunos productos puntos para el cálculo de las distancias cuadradas. Puedes encontrar útil la función `generateClusteringExamples` para probar tu código.
3.  (Opcional) Si escalamos todas las dimensiones de nuestros centroides iniciales y los datos de los puntos por algún factor distinto a cero, ¿garantizamos que podremos reconstruir las mismas agrupaciones después de correr k -medias? es decir ¿los mismos puntos en los datos pertenecerán a los mismos grupos antes y después de escalarlos? ¿Y si escalamos solo algunas dimensiones? Si tu respuesta es *si*, provee una explicación breve, si tu respuesta es *no*, provee un contraejemplo.

Se espera que respondas con: dos partes. La primera es una respuesta si/no y explicación o contraejemplo para la primera subpregunta (escalar todas las dimensiones). La segunda es una respuesta si/no y explicación o contraejemplo para la segunda subpregunta (escalar solo algunas dimensiones).