PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
IIC-2613, INTELIGENCIA ARTIFICIAL

TAREA 3: ALGORITMOS DE APRENDIZAJE DE MÁQUINA

LA TAREA ES INDIVIDUAL

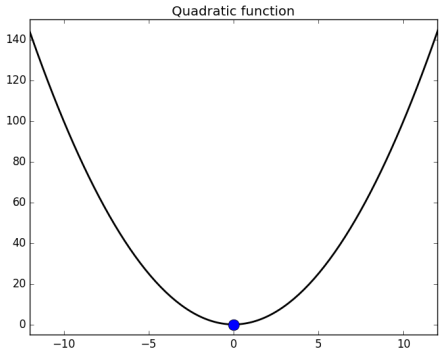
Fecha de Entrega: 25/11, 23:59 hrs., Siding.

1 Objetivo

Muchach@s, en esta actividad tendrán la oportunidad de experimentar con la técnica de minimización de funciones basada en descenso de gradiente, la cual es clave para el entrenamiento de redes neuronales. Adicionalmente, podrán experimentar nuevamente con el poder discriminativo de los SVMs y explorar el mundo de análisis de texto mediante la clasificación de frases según su motivación positiva o negativa (sentiment analysis).

2 Experimentos con Descenso de Gradiente [20%]

El objetivo de esta actividad es ver en la práctica el funcionamiento de técnicas de descenso de gradiente, en particular, ver el efecto de la inicialización y tasa de aprendizaje. Para ello usaremos el código Python en el archivo DescGrdt1.py. Este archivo nos permite encontrar el mínimo de la función $f(x) = x^2$ usando la técnica de descenso de gradiente. La siguiente figura muestra la gráfica de esta función, indicando la posición del mínimo:



Como se aprecia, el mínimo de la función ocurre para el valor $x = 0$. Valor para el cual la función alcanza un valor $f(x = 0) = 0$.

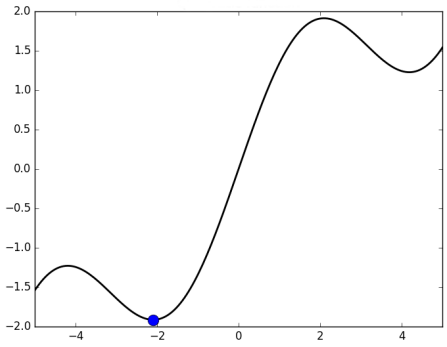
En la sección 3 del código está la definición de los parámetros relevante. En particular se define un valor inicial para la posición del mínimo y la tasa de aprendizaje, siendo los valores por defecto 10 y 0.1, respectivamente.

Realice los siguientes experimentos y comente sus observaciones:

- Ejecute el código en DescGrdt1.py y observe la simulación que muestra la trayectoría seguida por el descenso de gradiente. Primero utilice los valores por defecto para los parámetros relevantes y luego modifique la tasa de aprendizaje a valores más pequeños, comente sus observaciones.

- Repita lo anterior, pero en esta oportunidad incremente lentamente el valor de la tasa de aprendizaje respecto de su valor inicial por defecto, comente sus observaciones.

Veamos ahora una función más entretenida, $f_2(x) = sin(x) + \frac{x}{2}$. La siguiente figura muestra la gráfica de esta función, indicando la posición del mínimo:



- Ejecute el código DescGrdt2.py, el cual implementa el método de descenso de gradiente para minimizar la función $f_2(x)$. Los valores por defecto para la posición inicial del mínimo y la tasa de aprendizaje son 1 y 0.1, respectivamente. Ejecute el código usando los valores por defecto, comente sus observaciones.
- Repita lo anterior, pero en esta oportunidad experimente con distintos valores para el punto de inicio y la tasa de aprendizaje. Comente sus observaciones.

3 Análisis de Sentimientos en Críticas de Películas [80%]

En esta actividad experimentarán la codificación de texto conocida como Word2Vec. Esta codificación será usada para entrenar clasificadores SVM binarios que permitirán identificar si el texto de una crítica a determinada película indica un comentario Positivo (+) o Negativo (-).

3.1 Información general

- **Codificación Word2Vec:** Este modelo consiste en una red neuronal pre-entrenada que permite codificar cada palabra de texto en un espacio de características de 300 dimensiones, de ahí su nombre Words-To-Vectors o simplemente Word2Vec. Esta codificación ha ganado notoriedad recientemente por su simplicidad y por el hecho que logra capturar relaciones semánticas entre las distintas palabras. Por ejemplo, las codificaciones de palabras tales como *clase*, *curso*, *asignatura* son similares. Aún más sorprendente, la codificación resultante permite realizar operaciones algebraicas que capturan relaciones semánticas. Por ejemplo, si al vector que representa la palabra *Rey* se le resta el vector que representa la palabra *Hombre* y se le suma el vector correspondiente a *Mujer*, da como resultado un vector muy similar al que codifica la palabra *Reina*. Algebraicamente:

  Word2Vec(Rey) - Word2Vec(Hombre) + Word2Vec(Mujer) $\approx$ *Word2Vec(Reina).*

  Lamentablemente, no tenemos tiempo en el curso para explicar la teoría detrás de Word2Vec, pero los interesados pueden averiguar más en internet o luego cursar IIC-3423 Big Data, donde veo estos temas en detalle.

3.2 Set de datos

En esta tarea trabajarán con los siguientes set de datos:

- **Polarity dataset (v2.0):** contiene 2.000 críticas de cine. El set fue hecho de manera de equilibrar los casos positivos y negativos (1000 y 1000).

- **Large Movie Review Dataset (v1.0):** contiene 25.000 críticas de entrenamiento y 25.000 de test. Además de 50.000 sin rotular, que pueden ser usadas para delinear mejor el diccionario de palabras. El set fue hecho de manera de equilibrar los casos positivos y negativos. Dado que usaremos modelos de palabras preentrenados, sólo utilizaremos los conjuntos de entrenamiento y test.

3.3 Modelos

En esta tarea utilizarán modelos preentrenados que nos permitirán operar sobre textos. A continuación los detalles:

- *Word2Vec*: este modelo permite obtener la codificación Word2Vec de cada palabra en idioma inglés. Para esta tarea usaremos los modelos obtenidos por una implementación de la empresa Google que fueron entrenados con texto de noticias que incorporan un total de 3.000 millones de palabras. El número es correcto!, usando la nomenclatura inglés serían 3 billones de palabras.
- *Stopwords*: este modelo consiste de una lista de palabras no informativas, como artículos y ciertas preposiciones, las cuales son eliminadas de los textos de entrada por no aportar a la discriminación.
- *PorterStemmer*: este modelo permite transformar sustantivos y verbos a su forma base o raíz. Por ejemplo, este modelo transforma palabras como *cats* a *cat*, o *running* a *run*. En particular, en esta actividad usaremos el stemmer propuesto por Martin Porter, el cual es ampliamente usado en el análisis de texto.

A modo de ejemplo, el siguiente código Python permite cargar los modelos anteriores.

```
# Loading Word2Vec, this takes a while..."
from gensim.models import Word2Vec
model = Word2Vec.load_word2vec_format('./google_news.bin', binary=True)

## Loading Stopwords model
from nltk.corpus import stopwords

# Loading Stemmer model
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()          # stemmer
st = stemmer.stem                  # get the stemming function
```

Las librerías necesarias se pueden descargar de los siguientes links:

- gensim: https://radimrehurek.com/gensim/
- nltk: http://www.nltk.org/
- google_news.bin: estará disponible para descarga desde el sitio web del curso.

Dado que los ejemplos de entrenamiento contienen frases, para aplicar la codificación Word2Vec es necesario identificar las palabras de cada frase y luego obtener una codificación de la frase completa. En esta tarea usaremos la siguiente estrategia:

- Para cada palabra en la frase de entrada obtener su codificación word2vec aplicando el modelo correspondiente.
- Obtener la codificacón de la frase de entrada calculando el vector promedio de las codificaciones de sus palabras.

Para la clasificación, dado que el dataset *Polarity dataset* es pequeño y no tiene una partición especial en sets de entrenamiento y test, se realiza la evaluación usando la técnica de k-folds, más detalles sobre esto en la ayudantía de la tarea. En el caso del dataset *Large Movie Review Dataset* se utiliza la evaluación tradicional usando un set independiente de test.

3.4 Actividades

- Usando Word2Vec vea si las siguientes relaciones son efectivas. Compare con relaciones aleatorias para tener una noción de que significa cercanía en este espacio.
  - *Word2Vec(Rey) - Word2Vec(Hombre) + Word2Vec(Mujer)* $\approx$ *Word2Vec(Reina).*
  - *Word2Vec(Paris) - Word2Vec(France) + Word2Vec(Italy)* $\approx$ *Word2Vec(Rome).*
  - *Word2Vec(Breakfast)* $\approx$ *Word2Vec(Cereal).*
  - *Word2Vec(Class)* $\approx$ *Word2Vec(Course).*
  - Agregue alguna relación que crea relevante.
- Usando un editor de texto mire los archivos PolarityExamples.txt y MovieReviewExamples.txt disponibles en el sitio web del curso. Estos archivos contienen algunos ejemplos del tipo de frases que incluyen los datasets Polarity y MovieReview. Comente sus observaciones.
- Pruebe el rendimiento de clasificación obtenido por un SVM sobre la codificación de frases descrita anteriormente. Pruebe con kernel lineal y RBF. ¿Nota diferencias entre los rendimientos obtenidos en los 2 datasets?, comente.
- Analice la matriz de confusión reportada para cada dataset, ¿Cuál es el error más común?, ¿Qué categoría obtiene el mejor rendimiento?, ¿Cuál es el más bajo?.
- Indique 2 acciones que podrían mejorar el rendimiento de clasificación.

Genere un archivo de texto con 10 críticas ficticias de película: 5 críticas positivas y 5 negativas. Pruebe en estas críticas el rendimiento del mejor modelo obtenido. En particular, realice experimentos orientados a evaluar robustez respecto a situaciones como las siguientes:

- Robustez de los algoritmos respecto a cambios en el orden de las palabras en la crítica. Por ejemplo, “Is this a good critic?” en lugar de “This is a good critic”.
- Robustez respecto a críticas que mezclan aspectos positivos y negativos. Por ejemplo, “This is a good movie, but it is too slow”.
- Robustez respecto a comentario irónicos. Por ejemplo, “The actor was great, a real great joke”.

Para los que queden motivados y durante las vacaciones quieran entrenar sus propios modelos de Word2Vec usando palabras en español. Por ejemplo sus emails, facebook, cursos, info de pasatiempos, etc, pueden pueden descargar código abierto desde el siguiente link: https://code.google.com/archive/p/word2vec/. Esta página tiene además detalles de como entrenar un nuevo modelo. Suerte con sus tarea!.