Trabajo Práctico Nro 2:
Optimización

Para este trabajo se busca la implementación de una SVM (Support Vector Machine). Dado que las SVM son un ejemplo de clasificadores que responden a la solución de un problema de optimización convexo, el objetivo es que el alumno pueda internalizarse con conceptos de optimización a partir de un ejemplo práctico y de interés.

Planteo general del problema

Dado un conjunto de datos de entrenamiento de la forma $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$, donde cada par (x_i, y_i) se conforma de una observación, $x_i \in \mathbb{R}^d$, y su etiqueta correspondiente, $y_i = \{-1, 1\}$.

El objetivo es hallar el hiperplano en $\mathbb{R}^d$ de la forma $w^T x + b$ que separe las observaciones de acuerdo a su clase. En el caso donde los datos no son linealmente separables, buscamos $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ que satisfagan

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ s.t. \quad & y_i(w^T x + b) \geq 1 - \xi_i \\ & \xi_i \geq 0. \end{aligned}$$

Alternativamente, el problema puede verse como la minimización de una función costo dada por

$$J(w, b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max\{0, 1 - y_i(w^T x + b)\}. \tag{1}$$

A la función de pérdida dada por $\ell(t) = \max\{0, 1 - t\}$ se la conoce como *hinge loss* (fig. 1.

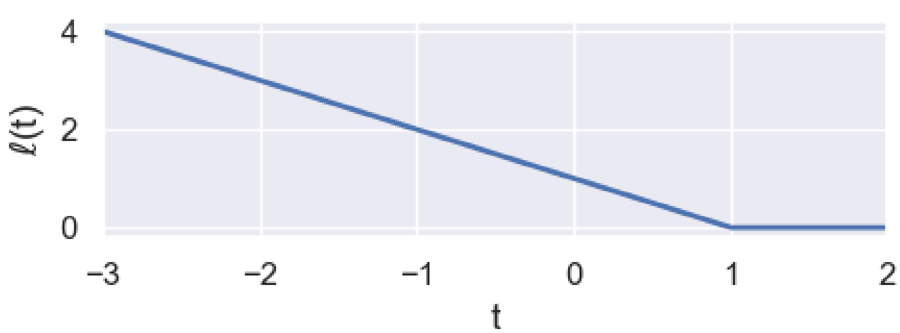

Figure 1: Hinge loss

Bajo esta perspectiva, estamos frente a un problema de optimización sin restricciones donde la función a optimizar es derivable (salvo en un punto). Podemos resolver este problema usando la familia de métodos de (sub)gradiente descendiente. Definimos el (sub)gradiente para la *hinge loss* como: $\frac{d\ell}{dt} = \begin{cases} -1 & t < 1 \\ 0 & t \geq 1 \end{cases}$.

Una vez hallados w y b óptimos podemos evaluar la performance del clasificador entrenado sobre un nuevo conjunto de datos (test) estimando la etiqueta mediante $\hat{y} = sgn(w^T x + b)$.

1 Problema a resolver

1. Escribir una función que calcule el gradiente para la función de costo $J(w, b)$ dada por (1).

2. Usar la función del item anterior para implementar el algoritmo de gradiente descendiente estocástico (SGD). Usar como criterio de corte la variación en la función de pérdida entre un paso y el siguiente (cortar si $\Delta J < \epsilon$). Opcional: implementarlo para distintos tamaños de *batch* para luego analizar el comportamiento respecto de este parámetro.

3. Cargar los datos que se encuentran en el archivo `train.csv`, y usarlos para entrenar una SVM basándose en el SGD implementado.

4. Importar la función `SVC` del paquete `svm` de la librería Scikit-Learn (`from sklearn.svm import SVC`). Usarlo para entrenar una SVM. Comparar los valores de w y b obtenidos con esta librería contra los hallados por gradiente descendiente. Analizar los vectores soporte hallados.

5. Cargar los datos que se encuentran en el archivo `test.csv` y analizar la performance de la SVM entrenada por ambos métodos. Usar como métrica de error la proporción de etiquetas predichas correctamente. ¿Coinciden las predicciones?

6. Analizar que ocurre cuando se quita del dataset de entrenamiento un vector soporte. ¿Y si sacamos uno que no lo era?

Sobre el use de la implementación de Scikit-Learn: A continuación se presentan los algunos de los atributos y métodos implementados sobre la clase SVC que serán de utilidad para el análisis pedido.

```
from sklearn.scvm import SVC
...
Xtrain = ...
ytrain = ...
Xtest = ...
ytest = ...
...
svm = SVC() # aca pueden pasarle distintos parametros, ver documentacion
svm.fit(Xtrain,ytrain) # entrena la SVM
svm.support_vectors_ # me devuelve los vectores soporte
svm.supprt_ # guarda los indices de los datos de entrenamiento
svm.coef_ # aca se guarda el vector w
svm.intercept_ # aca se guarda el valor de b
svm.score(Xtest, ytest) # calcula el score (proporcion de instancias bien clasificadas)
svm.predict(Xtest) # me devuelve la prediccion de etiquetas.
```