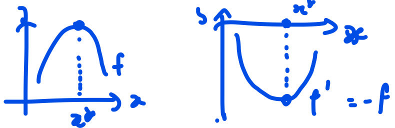
Análisis Matemático para Inteligencia Artificial

Martín Errázquin (merrazquin@fi.uba.ar)

Especialización en Inteligencia Artificial

Optimización en ML, aprendizaje supervisado



$x^* = \arg\max f(x) = \arg\min -f(x)$

NOTA: No se puede optimizar algo que no sea campo escalar

Convención: Todos los casos van a asumirse de minimización, sin pérdida de generalidad ya que maximizar f equivale a minimizar $f' = -f$.

$(1,0) \overset{?}{\gtrless} (0,1)$

Optimización en general: buscamos minimizar $J(\theta)$, tenemos toda la información necesaria disponible.

Optimización en ML: buscamos minimizar $J(\theta)$, sólo disponemos de un $\hat{J}(\theta)$ basado en el dataset disponible.

muestra aleatoria

?!

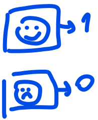
$(x, y) \sim F$

in out

Conclusión: **no** son el mismo problema.

• info parcial

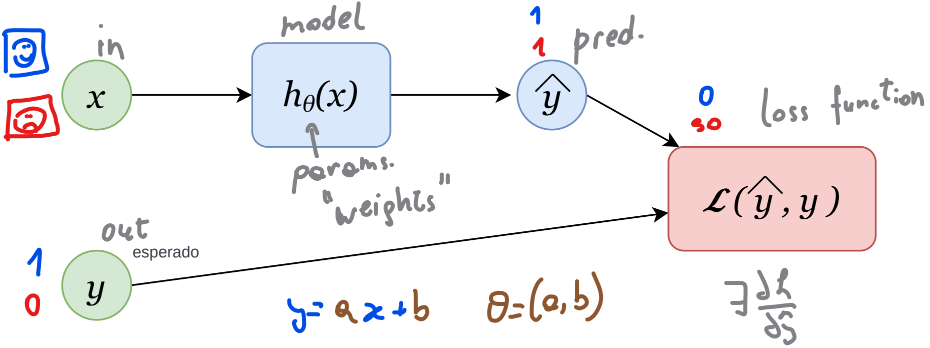
• distorsión



	TRAIN	TEST
A	0.8	0.4
B	0.5	0.5

El modelo B es mejor porque en test dio mejor

Aprendizaje supervisado: esquema



Dada una observación (x, y) fija, entonces la predicción $\hat{y} = h_\theta(x)$ depende puramente de los parámetros θ del modelo, y por lo tanto también la pérdida/error.

Para un dataset $(x_1, y_1), \ldots, (x_n, y_n)$ fijo, definimos entonces una función de costo $J(\theta)$ que sólo depende de los parámetros del modelo, y queremos minimizarla.

$\theta^* = \text{argmin}\ J(\theta)$

J(tita) generalmente es la perdida promedio

Proxy target/surrogate loss

Importante: Definida una función de pérdida por observación $\mathcal{L}(\hat{y}, y)$, la función de costo típicamente se define como

distrib. teórica de los (x, y)

$$J(\theta) = \mathbb{E}[\mathcal{L}(\hat{y}, y)]$$

$, h_\theta(x)$

risk minimization

de donde

aprox. + dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

$$\hat{J}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\hat{y}_i, y_i) = \hat{E}[\mathcal{L}(\hat{y}, y)]$$

empirical risk minimization

Como solo se tiene un conjunto (dataset) se estima la funcion de costo

Denominamos *proxy* o *surrogate* a una función f' que queremos minimizar como *medio* para minimizar otra función f que es la que verdaderamente nos interesa.

El esquema entonces resulta:

nivel de felicidad

- *aprendemos* vía train set $\rightarrow$ *necesitamos* minimizar $J_{train}(\theta)$
- *predecimos* vía test set $\rightarrow$ *queremos* minimizar $J_{test}(\theta)$

feliz vs triste

En el train se puede minimizar con algo totalmente diferente que en el test

Ejemplo

$y \in \{0,1\}$ $\hat{y} \in \{0,1\}$

Supongamos un caso de clasificación binaria donde definimos la función de pérdida como el *accuracy*, definido como

$$J_{TEST} \leftarrow \mathcal{L}(\hat{y}, y) = 1\{\hat{y} \neq y\} = \begin{cases} 1 & \text{si error} \\ 0 & \text{si acierto} \end{cases}$$

Como podemos ver, esta función de pérdida es *muy mala* para minimizar.

Planteamos entonces entrenar sobre la *cross-entropy loss*

$$J_{TRAIN} \leftarrow \underline{\mathcal{L}_{train}}(\hat{y}, y) = -y \cdot log(\hat{y}) - (1-y) \cdot log(1-\hat{y}) = \begin{cases} -\log(\hat{y}) & \text{si } y=1 \\ -\log(1-\hat{y}) & \text{si } y=0 \end{cases}$$

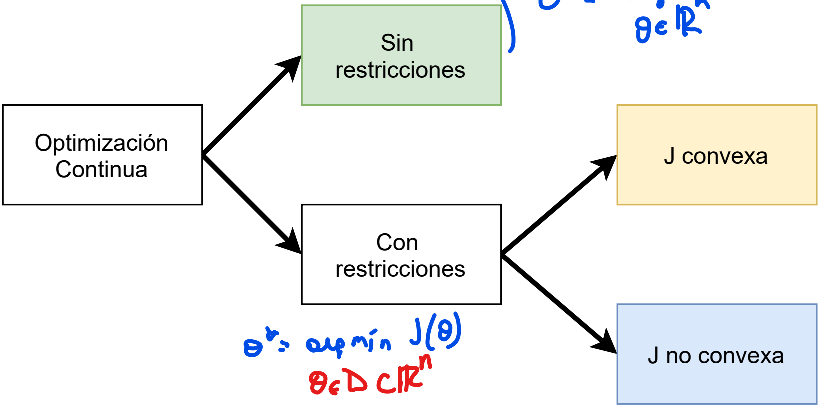
que nos permite ya no sólo trabajar con $\hat{y} \in \{0,1\}$ sino todo el rango continuo $[0,1]$ de probabilidades, además de, especialmente, ser **derivable respecto de** $\hat{y}$.

$$\exists \frac{\partial \mathcal{L}_{train}}{\partial \hat{y}} \qquad \hat{y} \in (0,1)$$

NOTA: lo importante es que la funcion
de perdida sea derivable respecto
de la prediccion

Ahora que nuestro problema es minimizar $J_{train}(\theta)$, podemos separarlo en varios casos:

$\theta^* = \arg\min_{\theta \in \mathbb{R}^n} J(\theta)$



$\theta^* = \arg\min_{\theta \in D \subset \mathbb{R}^n} J(\theta)$

↳ valores INVÁLIDOS