Richiami formule Hi-Fi

$$L_Z(X \to Y) = \epsilon(Y|Z) - \epsilon(Y|X,Z)$$

$$L_\emptyset(X \to Y) = \sigma^2(y) - \epsilon(Y|X)$$

$$L_{z_{min}}(X \to Y) = \epsilon(Y|z_{min}) - \epsilon(Y|X,z_{min})$$

$$L_{z_{max}}(X \to Y) = \epsilon(Y|z_{max}) - \epsilon(Y|X,z_{max})$$

$$R = L_\emptyset(X \to Y) - L_{z_{min}}$$

$$S = L_{z_{max}} - L_\emptyset(X \to Y)$$

$$L_{z_{max}} = U + R + S$$

Gli insiemi $L_{z_{min}}$ e $L_{z_{max}}$ vengono trovati attraverso la greedy search: Una feature candidata viene aggiunta all'insieme corrente e viene scelta quella che massimizza o minimizza il loco, il criterio di stop si basa su un calcolo statistico (surrogate test).

Scalabilità

L'algoritmo di ricerca prevede il riaddestramento del modello su ogni sottoinsieme ad esempio sulle feature $x, z_{min}^{(curr)}$ questo è uno svantaggio se il modello è computazionalmente difficile da addestrare in maniera iterativa, quindi si rende necessaria una modifica per stimare le componenenti di U, R, S mantenendo l'approccio originale ossia quello di essere agnostico rispetto al modello.

Possibili approcci

Approccio A

Se il modello da addestrare è troppo complesso, si addestra un modello surrogato (più semplice) e si applica il metodo HI-Fi con greedy search sul modello surrogato per la decomposizione.

- Vantaggio: Non richiede alcuna modifica di Hi-Fi
- Svantaggio: Il modello rischia di essere troppo semplice e di non riuscire a spiegare le relazioni apprese dal modello originale.

Approccio B

Invece di riaddestrare il modello, andiamo a mascherare le feature di cui vogliamo calcolare l'importanza. Ad esempio $S \subseteq \{ 1.. . d\}$ è il sottoinsieme di feature da mantenere attive, $j \notin S$ è la feature da mascherare. Il dataset sarà costruito come:

$$x_{S,j}^{(i)} = \begin{cases} x_{S,j}^{(i)} = x_j^{(i)} se\ j \in S \\ \tilde{x}_j se\ j \notin S \end{cases}$$

$\tilde{x}_j$ può essere:

- Sostituito sempre con la media di quella feature
- Campionato da un dataset di background
- Stimato attraverso la probabilità $p(x_j|x_S)$
    - Ad esempio tramite una rete generativa condizionata solo dalle features da mantenere attive cioè quelle in S

Dopodiché l'errore viene calcolato facendo inferenza sulle features da mantenere attive e le feature "mascherate"

$$\hat{y}_S^{(i)} = f(X_S^{(i)})$$

$$\epsilon(Y|S) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, \hat{y}_S^{(i)})$$

Il problema che si riscontra è che spesso:

$$\epsilon(Y|S \cup \{j\}) > \epsilon(Y|S)$$

Cioè il modello produce un errore minore quando gli diamo meno features e questo è controintuivo perché ci aspetteremmo il contrario infatti nell'approccio originale degli autori questo non si verifica.

Il motivo è che diamo in input al modello una distribuzione che il modello non si aspetta (covariate shift) e quindi reagisce in una maniera inaspettata e instabile. Una possibile soluzione è di allenare il modello su un dataset corrotto in modo da renderlo robusto a features mancanti cosi che non reagisca in maniera incoerente quando avviamo l'algoritmo di decomposizione (curriculum masking).

Approccio C

- Addestrare il modello f_θ sul dataset originale (X, y)
- Dopo che ha appreso la relazione tra feature e target lo si adatta con il fine-tuning con campioni mancanti cosi che riesca a gestire scenari con feature rimosse.

Approccio D

Se un modello è differenziabile (es. rete neurale), possiamo usare i gradienti della predizione rispetto alle feature in input per stimare l'importanza di ciascuna feature $g_j(x) = \frac{\partial f(x)}{\partial x_j}$

- Se una piccola variazione della feature x_j cambia molto l'output del modello $f(x)$ significa che x_j è importante.
- Se invece il gradiente è quasi nullo, il modello non dipende molto da quella feature.