Relazione progetto intelligenza artificiale

Andrea Commodi

Matricola: 7131582

1 Introduzione

In questo progetto è stato implementato un algoritmo di apprendimento automatico chiamato Dual Form Perceptron ed è stato utilizzato per effettuare previsioni su quattro datasets differenti. Per maggiori informazioni sui codici e la loro implementazione consultare il file **readme.md**.

2 Pseudo codice

Algorithm 1 Model training

Input: Train data $\mathcal{X}_{train}, \mathbf{y}_{train}$, max epochs T, kernel type $K()$
Initialize:
$\alpha_1, \alpha_2, \ldots, \alpha_n \leftarrow \mathbf{0} \in \mathbb{R}^n$
$b \leftarrow 0 \in \mathbb{R}$
$\mathbb{G}_{ij} \leftarrow K(\mathbf{x_i}, \mathbf{x_j}) \quad \forall i, j \in \{1, \ldots, n\}$ ▷ Compute gram matrix
for t = 1 to T **do**
    for each training example $\mathbf{x_i}$ **do**
        Compute $h(x) \leftarrow \sum_{j=1}^{n} \alpha_j y_j \mathbb{G}_{ij} + b$
        if $y_i \cdot h(x) \leq 0$ **then**
            $\alpha_i \leftarrow \alpha_i + 1$
            $b \leftarrow b + y_i$
        end if
    end for
end for
Output: α, b, $\mathcal{X}_{train}, \mathbf{y}_{train}$.

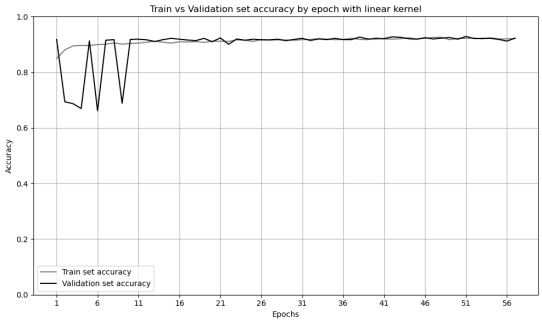
Algorithm 2 Model predict

Input: $\mathcal{X}_{test}$, $\mathbf{y}_{test}$, α, $\mathbf{y}_{train}$, $\mathcal{X}_{train}$, b, $K()$
function PREDICT($\mathcal{X}_{test}, \mathbf{y}_{test}$)
    predictions ← empty array
    for all $x_i \in \mathcal{X}_{test}$ **do**
        $\hat{y_i}$ = PREDICTPOINT(x_i)
        predictions ← $\hat{y_i}$
    end for
    accuracy ← $\frac{1}{n} \sum_{i=1}^{n} (\hat{y_i} == \mathbf{y}_{test}[i])$ ▷ where n=len(predictions)
    Output: predictions ▷ predictions=$(\hat{y_1}, ..., \hat{y_n})$
end function
function PREDICTPOINT(x_i)
    Compute $w \leftarrow \{\sum_{j=1}^{n} a_j \cdot \mathbf{y}_{train}[j] \cdot K(\mathcal{X}_{train}[j], x_i)\} + b$ ▷ where n=len(α)
    Output: sign(w)
end function

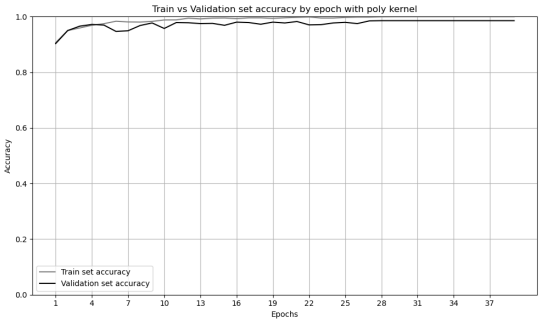
3 Datasets

3.1 Musk dataset

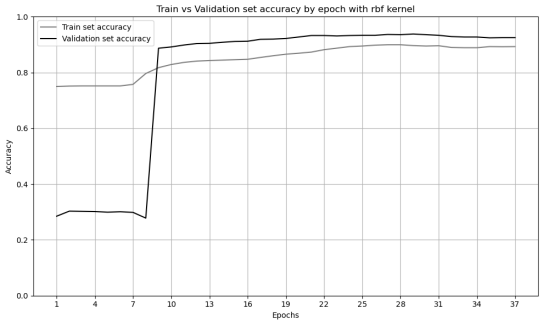
L'obiettivo è prevedere se una molecola è **musk** (profumata) o **non-musk**. Il dataset presenta 6598 esempi (15% **musk** e 85% **non musk**) ed è stata fatta una divisione dell'intero dataset in 60% train set, 20% validation set e 20% test set. Il modello è stato allenato sul set di train e ad ogni epoca è stata calcolata l'accuracy sul validation set per monitorare accuratezza ed eventuali overfitting o underfitting. Successivamente viene salvato il modello che ha ottenuto migliore accuracy nel validation set e viene testato sui dati di test. Infine vengono comparati i kernel sia in accuracy che in recall, precision ed F1 dato che il dataset utilizzato è sbilanciato. Questo perché in un dataset con queste proporzioni, se si implementa un classificatore che predice sempre la classe **non musk** si ottiene un accuracy circa dell'85%, ma ciò non implica che il classificatore sia un buon classificatore perché è necessario valutare anche la capacità di prevedere la classe meno dominante.



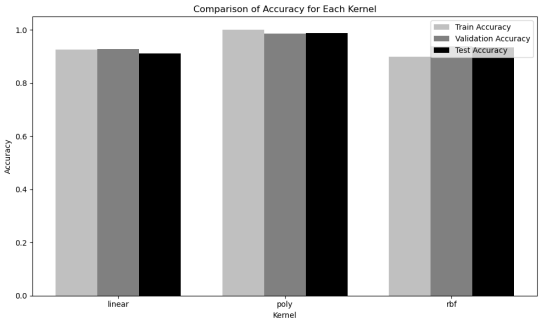
(a) Linear kernel



(b) Polynomial kernel



(c) RBF kernel



(d) Kernels compare

Figure 1: Confronto tra i kernel utilizzati

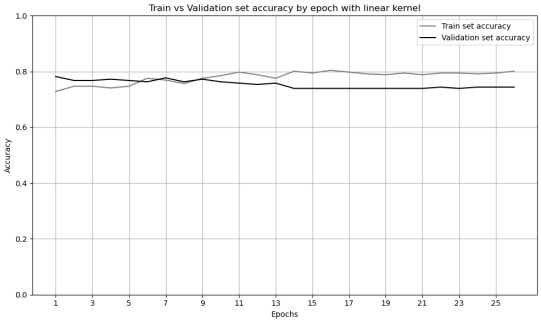
	Kernel	Precision (class 1)	Recall (class 1)	F1 (class 1)	Precision (class -1)	Recall (class -1)	F1 (class -1)	Test accuracy
0	Linear	0.663043	0.884058	0.757764	0.977011	0.916442	0.945758	0.911364
1	Poly	0.975248	0.951691	0.963325	0.991055	0.995508	0.993277	0.988636
2	RBF	0.767544	0.845411	0.804598	0.970696	0.952381	0.961451	0.935606

Figure 2: Statistiche dettagliate dei tre modelli

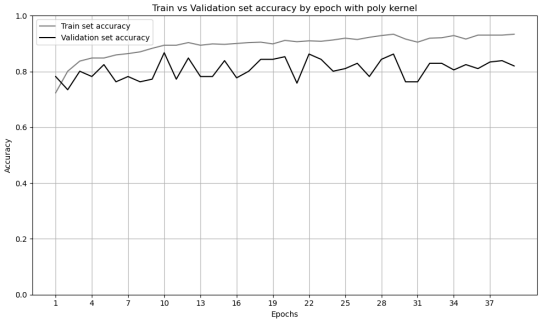
Come è possibile osservare, seppur a livello di accuracy (sia in validation che in test) tutti e tre i kernel superino lo 0.90%, osservando i valori nella figura 3 si evince facilmente che il modello con il kernel polinomiale è quello che performa meglio. In questo caso sia la **recall** che la **precision** della classe 1 sono notevolmente superiori nel kernel polinomiale, il che significa che il modello con kernel polinomiale riduce al minimo i falsi positivi e falsi negativi ed ha predetto circa il 97% dei positivi (classe minoritaria) in maniera corretta, a differenza del kernel lineare (66%) e del kernel RBF (76%), perciò in questo dataset è il modello migliore.

3.2 QSAR dataset

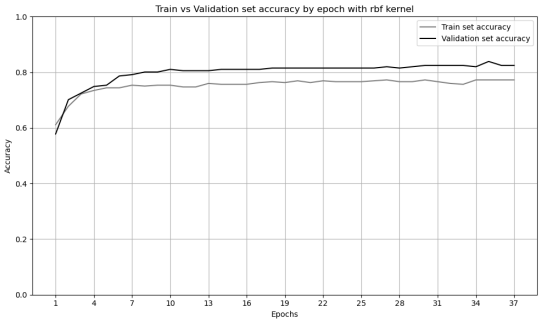
Il dataset QSAR sulla biodegradazione contiene informazioni riguardanti 1055 sostanze chimiche, di cui 356 sono classificate come "pronte per la biodegradazione" (0.34%) e 699 come "non pronte per la biodegradazione" (0.66%). L'obiettivo in questo dataset è classificare le molecole che sono biodegradabili e quelle che non lo sono.



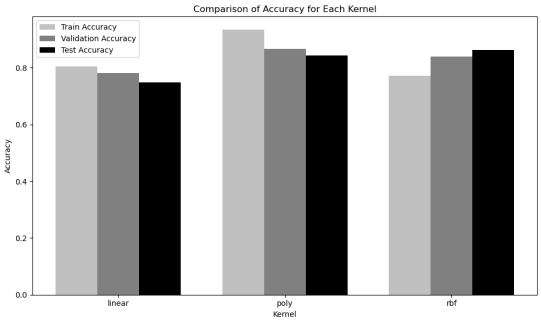
(a) Linear kernel



(b) Polynomial kernel



(c) RBF kernel



(d) Kernels compare

Figure 3: Confronto tra i kernel utilizzati

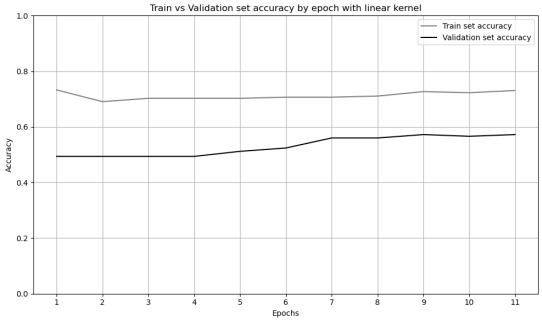
	Kernel	Precision (class 1)	Recall (class 1)	F1 (class 1)	Precision (class -1)	Recall (class -1)	F1 (class -1)	Test accuracy
0	Linear	0.988764	0.628571	0.768559	0.573770	0.985915	0.725389	0.748815
1	Poly	0.902256	0.857143	0.879121	0.743590	0.816901	0.778523	0.843602
2	RBF	0.959016	0.835714	0.893130	0.741573	0.929577	0.825000	0.867299

Figure 4: Statistiche dettagliate dei tre modelli

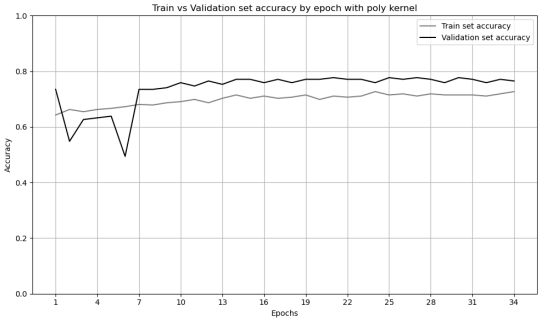
In questo caso, seppur nessuno dei tre modelli performa in maniera convincente, si nota immediatamente che c'è una differenza non banale tra il kernel lineare e gli altri due. Il kernel lineare non riesce a separare i dati mentre gli altri due fanno un lavoro migliore, segno evidente che i dati non sono linearmente seprabili nello spazio definito dal kernel lineare ma lo diventano di più in quello definito dagli altri due kernel. Tra i tre modelli, seppur quelli con kernel polinomiale e RBF hanno la stessa accuracy, quest'ultimo sembra performare meglio in termini di **recall** e **F1** nella classe minore (in questo caso quella negativa). Inoltre confrontando il grafico **3b** con il **3c** vediamo che mentre il modello con kernel RBF ha un'accuracy a bassa varianza e sottostima in train le performance che raggiunge in validation (underfitting), il modello con kernel polinomiale ha un'accurcay ad alta varianza e sovrastima in train le performace rispetto a quelle che raggiunge in validation (overfitting), evidenziando che in questo caso ha "imparato" troppo bene gli esempi di train e non ha capacità generalizzativa. Perciò in questo caso il modello con kernel RBF, seppur con un'accuracy in test di 0.867%, è il modello migliore.

3.3 Mammographic mass dataset

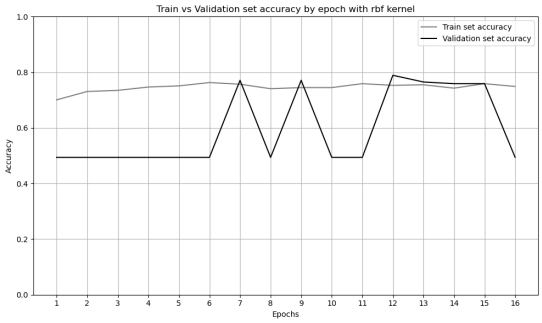
Il dataset in questione è relativo a un'analisi delle masse mammografiche e include informazioni sui tumori benigni e maligni. L'obiettivo è quello di prevedere se il tumore è maligno o beningo sulla base delle features. Il dataset contiene circa 1000 esempi e la proporzione è 55% tumori benigni e restante 45% tumori maligni.



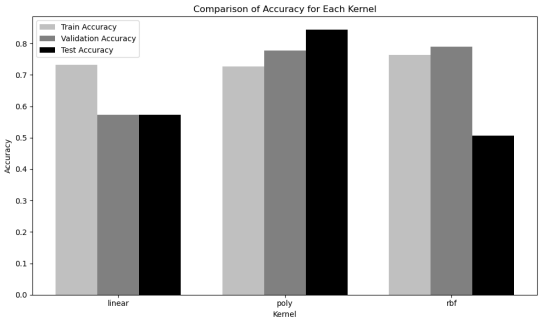
(a) Linear kernel



(b) Polynomial kernel



(c) RBF kernel



(d) Kernels compare

Figure 5: Confronto tra i kernel utilizzati

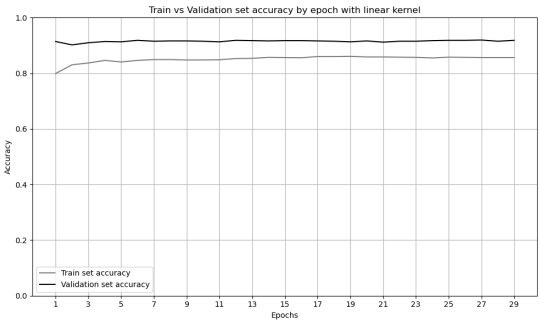
	Kernel	Precision (class 1)	Recall (class 1)	F1 (class 1)	Precision (class -1)	Recall (class -1)	F1 (class -1)	Test accuracy
0	linear	0.541935	1.000000	0.702929	1.000000	0.134146	0.236559	0.572289
1	poly	0.815217	0.892857	0.852273	0.878378	0.792683	0.833333	0.843373
2	rbf	0.506024	1.000000	0.672000	0.000000	0.000000	0.000000	0.506024

Figure 6: Statistiche dettagliate dei tre modelli

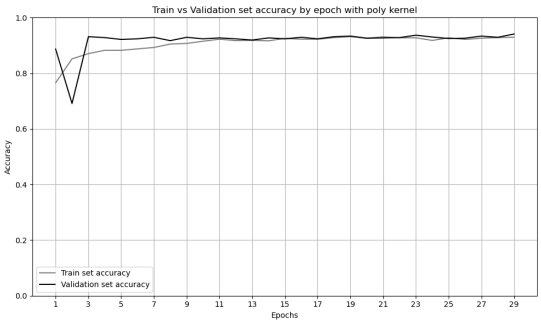
Dai risultati ottenuti vediamo chiaramente che i kernel RBF e lineare non riescono a separare bene i dati, contrariamente al kernel polinomiale. Per il kernel RBF sono stati provati $n = 50$ parametri con $\gamma \in [0.001, 1]$, ottenendo il miglior risultato con $\gamma = 0.5$. Per il kernel polinomiale invece la combinazione di parametri con accuracy più elevata (di circa 0.85) è $c = 0.12$ e $d = 2$.

3.4 Spambase dataset

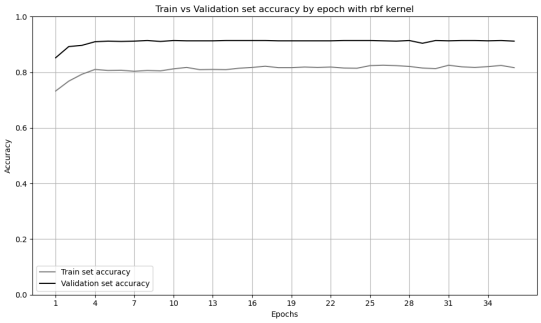
L'obiettivo principale è determinare se un'email sia **spam** (posta indesiderata) o **non-spam** (email legittima) sulla base di caratteristiche estratte dal testo del messaggio (57 features). Queste features includono la frequenza di parole specifiche, simboli speciali e schemi di utilizzo delle lettere maiuscole.



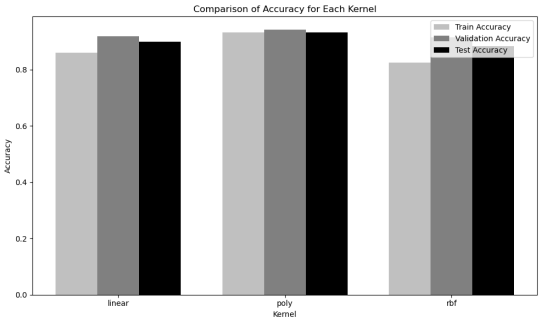
(a) Linear kernel



(b) Polynomial kernel



(c) RBF kernel



(d) Kernels compare

Figure 7: Confronto tra i kernel utilizzati

	Kernel	Precision (class 1)	Recall (class 1)	F1 (class 1)	Precision (class -1)	Recall (class -1)	F1 (class -1)	Test accuracy
0	linear	0.826966	0.958333	0.887817	0.966387	0.856611	0.908193	0.899023
1	poly	0.942149	0.890625	0.915663	0.924731	0.960894	0.942466	0.931596
2	rbf	0.809417	0.940104	0.869880	0.951579	0.841713	0.893281	0.882736

Figure 8: Statistiche dettagliate dei tre modelli

In questo caso osserviamo come tutti e tre i kernel danno ottimi risultati in accuracy, recall e precision. Il kernel RBF è quello che performa peggio in precision per la classe dei positiviti (**spam**) seguito dal kernel lineare. I migliori risultati si ottengono utilizzando il kernel polinomiale, non tanto in termini di accuracy quanto più in termini di precision nel prevedere la classe positiva (ovvero ha meno falsi positivi, quindi è meno probabile che email legittime vengano erroneamente etichettate come spam). Inoltre una cosa positiva è che il modello con kernel polinomiale non è soggetto nè ad overfitting nè ad underfitting, come si può osservare bene dal grafico **7c**.