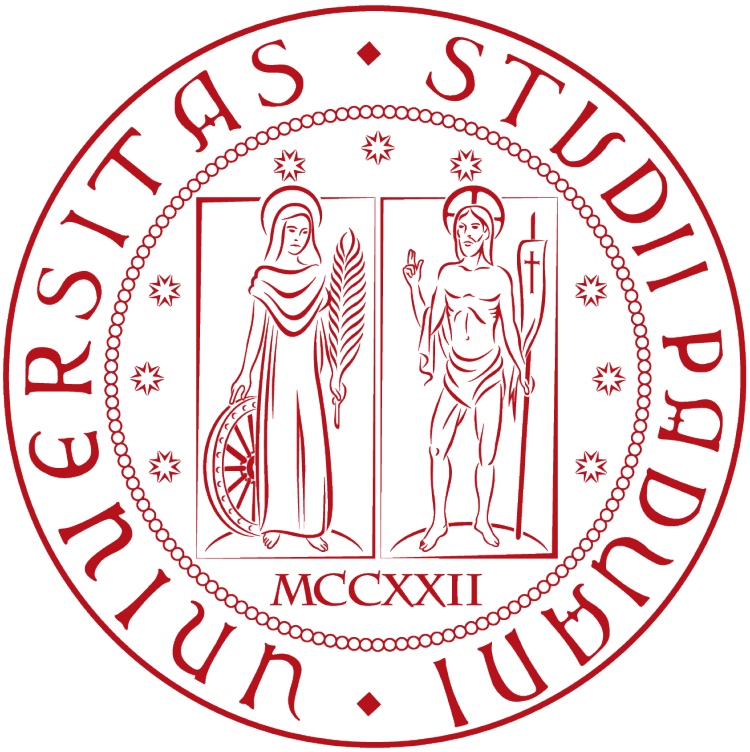

Relazione progetto di Intelligenza Artificiale

Informazioni sul progetto

Redatto	Francesco Corti - 1142525 Giovanni Sorice - 1144558
Referente	Giovanni Sorice - 1144558 giovanni.sorice@studenti.unipd.it

Link al sito

http://tecweb1819.studenti.math.unipd.it/gsorice/

Descrizione

Documento riportante le informazioni relative al progetto di intelligenza artificiale.

Indice

1 Introduzione

1.1 Scopo dell'analisi

Questo progetto consiste nell'applicazione e confronto di due metodi apprendimento supervisionato, per sottolineare le potenzialità e i problemi dei suddetti metodi.

1.2 Il problema dello spam

Il problema dello spam affligge da molti anni tutte le persone che dispongano di una casella di posta elettronica oppure di uno smartphone.
In passato è stato constato come l'eliminazione manuale dei messaggi spam, data la quantità di messaggi inviati, presentasse costi di tempo insostenibili.
Questo ha portato ad uno sviluppo di tecniche algoritmiche che permettessero di classificare automaticamente un messaggio ricevuto, come spam o ham.
Si è però scoperto che un approccio di tipo *offline learning*, presentava dei problemi.
Gli spammer, persone o bot che spediscono messaggi spam, riuscivano a modificare i messaggi in modo da renderli classificati come ham dai sistemi anti-spam presenti.
Questo era possibile in quanto gli algoritmi non evolvevano nel tempo, cambiando quindi la struttura del messaggio di spam questo veniva erroneamente identificato come un messaggio non spam.
Si è quindi passati a un approccio di tipo online che si è visto essere quello più ottimale.
Gli algoritmi in questo modo non smettono di imparare una volta terminato l'input dei dati ma evolvono nel corso del tempo imparando a classificare nuove tipologie di messaggi come spam. Abbiamo scelto questo problema dato che si adatta molto bene all'applicazione di algoritmi supervisionati.

2 Panoramica

Molte aziende e molti esperti del settore hanno studiato il problema dello spam, ciò ha portato ad algoritmi sempre più affidabili ed efficienti. Gli algoritmi presi in considerazione all'interno del nostro progetto sono stati:

- Logistic Regression;
- Neural Network (con e senza dropout).

Lo sviluppo di tali algoritmi è stato svolto all'interno di Google Colab.

*****Scrivere obiettivi********* Gli obiettivi principali che ci siamo prefissati trattano come la fase di progettazione e sviluppo doveva avvenire (cioè in modo corretto, efficace ed efficiente) e dei confronti in termini di velocità di training, precision e recall. *****Scrivere Fasi e tempi di lavoro********** Abbiamo deciso di dividere il progetto in 7 fasi:

- Scelta e studio dell'ambito
- Scelta e studio del problema
- Scelta e studio degli algoritmi
- Progettazione
- Implementazione
- Tracciamento dei risultati
- Confronto tra gli algoritmi

alle quali abbiamo dedicato n1,n2,...,n7 ore/lavoro persona rispettivamente. Il progetto ha avuto inizio gg/mm/aaaa ed è stato terminato gg/mm/aaaa. ****Scrivere Persone coinvolte nel lavoro e loro ruolo******

3 Metodologie utilizzate

Per il progetto didattico abbiamo deciso di utilizzare i seguenti approcci: l'algoritmo di *Logistic Regression* e le *Neural Networks*.
Tuttavia sono possibili altri approcci che, per rimanere all'interno dei tempi stabiliti, non siamo riusciti ad affrontare.

3.1 Studio del problema

3.2 Reti neurali

Il primo approccio che abbiamo scelto di utilizzare è stato quello delle reti neurali. Per farlo ci siamo appoggiati alla libreria *TensorFlow* la quale, dalla versione 2.0.0, integra *Keras*.
Maggiori informazioni riguardanti l'integrazione di Keras sono presenti al seguente link *tf.keras*.
Questo ci ha permesso di:

1. Utilizzare TensorFlow(*tf*) come ecosistema;
2. Definire la rete tramite la libreria *tf.keras*;

3.2.1 Struttura

È stata scelto di utilizzare il tipo di rete *Sequential*.
Abbiamo creato due reti, con tre layer di tipo *Dense* con funzione di attivazione:

1. *Relu* per i nodi interni;
2. *Sigmoid* per l'output della rete.

La prima rete presenta la seguente struttura:

```
1    model = Sequential()
2    model.add(Dense(512, activation = relu ))
3    model.add(Dense(256, activation = relu ))
4    model.add(Dense(1, activation = sigmoid ))
```

La seconda rete presenta la seguente struttura:

```
1    model = Sequential()
2    model.add(Dense(512, activation = relu ))
3    model.add(Dropout(0.2))
4    model.add(Dense(256, activation = relu ))
5    model.add(Dropout(0.2))
6    model.add(Dense(1, activation = sigmoid ))
```

3.2.2 Configurazione

Tramite il metodo *compile()* presente in Keras è possibile stabilire:

1. **Loss functions**: *binary crossentropy* nel nostro caso, maggiori info al seguente link *loss functions*;
2. **Optimizer**: l'ottimizzatore che verrà utilizzato per l'aggiustamento dei pesi e per minimizzare la loss function. Nel nostro caso *Adam*.
3. **Metrics**: lista di metriche che verranno valutate dal modello durante la fase di training e testing. Nel nostro caso è stata scelta la metrica di *binary accuracy*.

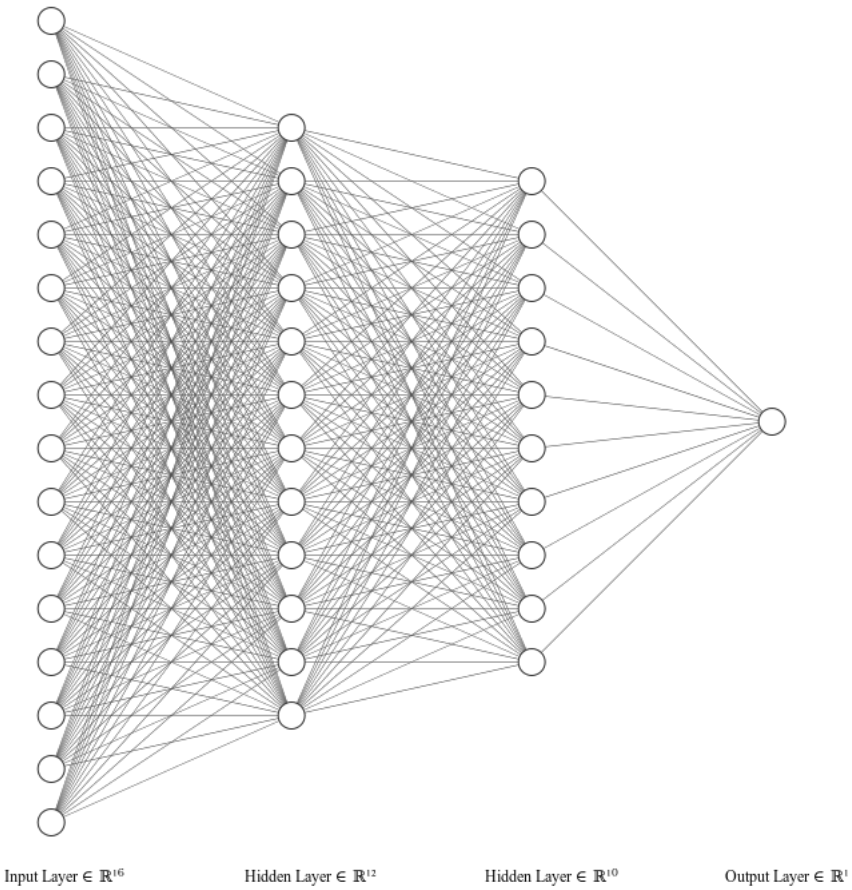

Figura 1: Esempio di rete neurale

3.3 Logistic Regression

s

4 Realizzazione

Suddivisione dataset

5 Validazione

6 Comparazioni

7 Conclusioni

A Tabelle riassuntive

Riferimenti