Machine Learning - Relazione progetto

**Autori V.M. & A.P.**

**Descrizione del dominio di riferimento e obiettivi dell’elaborato**

**Descrizione del dominio:** Il progetto consiste nel creare, perfezionare e confrontare due modelli di apprendimento automatico nel campo del riconoscimento della classe di appartenenza di fotografie satellitari, eventualmente raffiguranti **navi**.

Ci siamo chiesti in quali domini applicativi sarebbe potuta risultare utile questo tipo di apprendimento automatico, individuando due possibili scenari:

* riconoscimento di immagini acquisite da sensore montato su missile antinave, al fine di guidare questo contro il bersaglio navale designato;
* riconoscimento di immagini acquisite da droni impegnati in operazioni di ricerca di navi disperse.

Svilupperemo quindi i modelli nell’ottica di trattare questi due domini applicativi.

**Obiettivi dell’elaborato**:

Induzione di due modelli supervisionati che permettano la classificazione di immagini satellitari in due classi:

* **Classe 1 (ship):** l’immagine contiene la rappresentazione di una nave per intero;
* **Classe 0 (no ship):** l’immagine non contiene la rappresentazione di una nave per intero.

Si noti che saranno classificate con la **classe 1** solo le immagini satellitari che contengono la completa rappresentazione di una nave; al contrario, la **classe 0** conterrà sia le immagini che non contengono navi, sia quelle che ne contengono solamente una parte.

**Scelte di design per la creazione del data set, eventuali ipotesi o assunzioni**

Immagini d’esempio del dataset:

Immagine che contiene schermata, testo

Descrizione generata automaticamente

Al fine di risparmiare risorse, principalmente nell’apprendimento dei modelli, abbiamo scelto di portare le immagini originali in scala di grigi. Si passa quindi da immagini a immagini .

Ultimato il progetto abbiamo deciso performare un’operazione di **ablation**, per capire effettivamente se la nostra scelta di operare sui dati in scala di grigi fosse stata la migliore. Abbiamo ri-eseguito il codice prodotto, con qualche aggiustamento, per creare, addestrare e fare inferenza sui dati originali a colori piuttosto che su quelli in bianco e nero.

Le performance riscontrate (vedi l’ultima sezione) così facendo non sono risultate troppo dissimili da quelle ottenute sul dataset portato in scala di grigi. Questo conferma la scelta di fare questo tipo di **pre-processing** come corretta.

Si ribadisce come i modelli *definitivi* sono quelli creati e addestrati su immagini in scala di grigi.

Immagine che contiene testo, schermata, bianco e nero, nero

Descrizione generata automaticamente

Su questi dati è stata effettuata la **normalizzazione**: ogni feature diventa con media zero e varianza 1, eseguendo su ogni dato l’operazione:

Link al dataset: <https://www.kaggle.com/datasets/rhammell/ships-in-satellite-imagery>

**Descrizione del dataset e relativa analisi esplorativa**

Il dataset che utilizzeremo per l’induzione dei modelli sarà quindi rappresentato da immagini in *scala di grigi* di dimensione , per un totale di 6400 features numeriche (i valori di grigio, appunto). Ogni immagine è etichettata come descritto nel primo paragrafo, ovvero:

* **Classe 1 (ship):** l’immagine contiene la rappresentazione di una nave per intero;
* **Classe 0 (no ship):** l’immagine non contiene la rappresentazione di una nave per intero.

A seguire l’analisi esplorativa del dataset.

**Dimensione del dataset:**

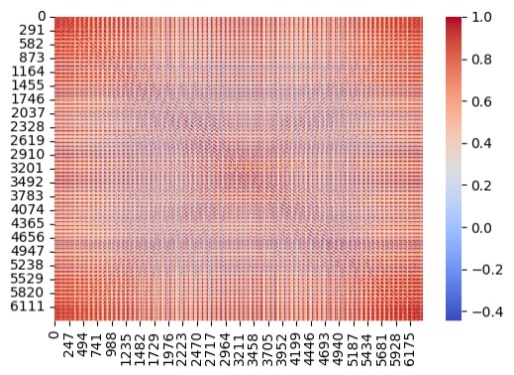
Come già detto, verranno trattate 6400 features numeriche a valori continui. Questo, unito all’elevato numero di esempi disponibili (4000 esempi, per un totale di 25.6 milioni di valori), ci ha portato considerare un importante utilizzo di **tecniche di riduzione della dimensionalità** per rendere il dataset più trattabile e, in particolare, all’utilizzo della ***Principal Component Analysis*** o ***PCA***, trattata di seguito.

**Correlazione delle features:**

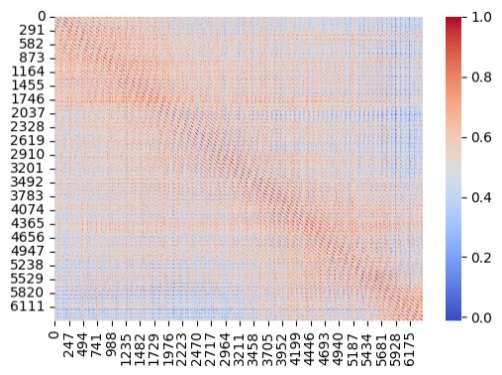
Trattando il dominio delle **immagini**, ci aspettavamo di trovare *correlazione* tra le features, dovuta al fatto che pixel vicini tra loro in un’immagine tendono ad avere valori simili. Aspettativa che è stata confermata attraverso l’utilizzo delle ***matrici di correlazione*** che, a causa della dimensione del dataset, sono state calcolate su diversi sottoinsiemi disgiunti di quest’ultimo:

Immagine che contiene modello, griglia, Simmetria, bianco e nero

Descrizione generata automaticamenteImmagine che contiene modello, Simmetria, quadrato, bianco e nero

Descrizione generata automaticamenteImmagine che contiene modello, Simmetria, quadrato, Rettangolo

Descrizione generata automaticamente

Immagine che contiene testo, schermata

Descrizione generata automaticamente

*Correlazione* delle **prime** **cento** **immagini (classe 1)**

*Correlazione* delle **cento** **immagini** **centrali (classe 0)**

*Correlazione* di cento **immagini** **casuali**

Il **dataset** ha prima tutti gli elementi di **classe 1 (ship)** e successivamente tutti quelli di **classe 0 (no ship).** Risulta quindi che le prime 100 immagini sono tutte di classe ship.

Riscontriamo essere presente *forte correlazione* sulle immagini di questa classe, cosa comprensibile guradano le rispettive immagini(abbastanza simili).

Per quanto riguarda le 100 immagini centrali esse sono tutte di classe no ship ed essendo il dato pittorico più variegato, riscontriamo meno correlazione.

Tuttavia, anche estraendo cento elementi casuali, si continua a notare un certo grado di *correlazione*: sarà necessario tenere conto di questo fattore nella scelta dei modelli più opportuni.

**Suddivisione in training set, validation set e test set:**

Considerato l’elevato numero di esempi disponibili, abbiamo optato per una divisione del tipo:

* **Training set**: 70% del dataset
* **Validation set**: 20% del dataset
* **Test set**: 10 % del dataset

Dove, per evitare ambiguità, con **validation set** intendiamo il set utilizzato per effettuare il *tuning* degli *iperparametri*, mentre con **test set** l’insieme di esempi per valutare il modello dopo l’addestramento.

Immagine che contiene testo, cerchio, schermata, Carattere

Descrizione generata automaticamente**Suddivisione delle classi:**

Nel dataset originale le classi sono suddivise secondo un rapporto 3:1 per la **classe 0**: nonostante sia una suddivisione non del tutto sbilanciata, abbiamo comunque deciso di applicare tecniche di ridimensionamento del dataset per bilanciare le classi. Il ridimensionamento è stato effettuato solo sul **training set**, in maniera da mantenere delle misure di valutazione sui **validation** e **test** **set** che rispettassero il **dataset** iniziale. In particolare, sono stati utilizzati sia **undersampling** che **oversampling** secondo l’ordine:

Immagine che contiene testo, cerchio, schermata, compact disk

Descrizione generata automaticamente

1. **Undersampling** della **classe 0 (no ship):** attraverso la classe [*RandomUnderSampler*](https://imbalanced-learn.org/stable/references/generated/imblearn.under_sampling.RandomUnderSampler.html) della libreria [*imblearn*](https://imbalanced-learn.org/stable/)sono stati eliminati 688 elementi della classe 0 (il 32.8% di quelli presenti), portando il rapporto tra le classi a 2:1;

Immagine che contiene testo, cerchio, schermata, Carattere

Descrizione generata automaticamente

1. **Oversampling della classe 1 (ship):** attraverso la classe [RandomOverSampler](https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.RandomOverSampler.html) della libreria [*imblearn*](https://imbalanced-learn.org/stable/)sono stati generati 354 esempi della classe 1 (il 25.1% di quelli presenti), portando il rapporto tra le classi a 3:2.

Queste operazioni ci hanno permesso di ottenere un training set maggiormente bilanciato e quindi più adatto alla fase di addestramento, evitando di perdere troppa *varianza* in seguito all’**undersampling** e di generare troppe nuove che istanze (che avrebbero falsato il training set) con l‘**oversampling**.

**PCA:**

|  |  |
| --- | --- |
| **Varianza spiegata** | **Componenti** |
| 0.6 | 4 |
| 0.7 | 9 |
| 0.8 | 24 |
| 0.9 | 84 |
| 0.95 | 195 |
| 0.99 | 615 |

Immagine che contiene testo, schermata, linea, numero

Descrizione generata automaticamenteDato il grande numero di features del dataset, si è fatto affidamento sulla PCA per cercare di ridurne la dimensionalità e avere migliori tempi di addestramento dei modelli. Selezionati una serie di valori critici, si è ricercato il minimo numero di componenti per arrivare a spiegare questi livelli di varianza, ottenendo i seguenti risultati:

Per arrivare al 99% della varianza spiegata si sono rivelate necessarie solamente 615 componenti su 6400, ovvero il 9.61% di queste. Tuttavia, basandosi sul grafico della *varianza spiegata cumulativa,* si è deciso di selezionare **84 componenti (l’1.31%),** necessarie per raggiungere il **90% di varianza spiegata**, valore che rappresenta il punto di gomito della curva.

**Descrizione e motivazione dei modelli di machine learning scelti**

Sono stati indotti due modelli diversi (uno per studente):

1. **Rete neurale** - Volpato Mattia
2. **Support Vector Machine** - Alessandro Preziosa

A seguire la descrizione e le motivazioni per la scelta di ciascuno dei due modelli.

**1 -** **Rete Neurale**

**Motivazione**:

La rete neurale è stata scelta in quanto risulta essere un modello adatto alla gestione di un numero elevato di *features numeriche e continue*; inoltre, sono note diverse altre applicazioni di reti neurali per il *riconoscimento delle immagini* che hanno ottenuto ottimi risultati (**MNIST, ImageNet, …**), le quali hanno portato a preferire questo modello rispetto ad altri. Infine, tra i vantaggi delle reti neurali è presente il riuscire ad avere una buona **tolleranza al rumore** eallo **sbilanciamento delle classi**, vantaggi che si adattano perfettamente alla struttura del dataset di riferimento.

**Descrizione**:

In seguito alla consultazione di materiale online, si è scelto di optare per una **rete a uno strato nascosto**, ritenuta ideale in diversi scenari per la classificazione di immagini in classi binarie. Inoltre, tra gli svantaggi principali delle reti neurali è presente la *difficoltà dell’addestramento* e l’alta probabilità di imbattersi in *ottimi locali*: la scelta di un’*architettura poco complessa* cerca di combattere questa problematica insita nel modello scelto.

La rete presenta la seguente architettura:

* **Strato di input**:
  + 84 nodi;
* **Strato nascosto**:
  + 53 nodi;
  + Funzione di attivazione: [*mish*](https://paperswithcode.com/method/mish)
* **Strato di output**:
  + Singolo nodo;
  + Funzione di attivazione: *sigmoid*

Per tutti gli strati sono stati utilizzati nodi completamente connessi ai nodi dello strato successivo (modello *Dense*). Come **ottimizzatore** dei **parametri** del modello (come il *learning rate*) è stato utilizzato l’algoritmo [*Adam*](https://keras.io/api/optimizers/adam/). Come **funziona di loss** è stata utilizzata la funzione [*binary crossentropy*](https://keras.io/api/losses/probabilistic_losses/)*,* ideale per problemi di classificazione binaria*.* Di seguito una rappresentazione grafica della **rete neurale**:

**Immagine che contiene testo, schermata, Carattere, linea

Descrizione generata automaticamente**

**Funzione di loss**: *binary crossentropy*

**Ottimizzatore SGD***: Adam*

**Epoche di addestramento**: 22

**2 -** **Support Vector Machine**

**Motivazione**:

Abbiamo scelto di usare il modello **Support Vector Machine** considerando:

* la sua buona velocità di addestramento e inferenza.
* semplicità concettuale: è un modello decisamente più semplice e con meno parametri di reti neurali profonde con milioni di iperparametri da ottimizzare. Quindi al posto di usare (magari in un estensione futura di questo progetto) modelli così complessi, vogliamo verificare come performa un modello *storico* ma comunque potente.

**Descrizione**:

L’implementazione di SVM è quella della libreria *sklearn*.

In particolare, si usa SVC, cioè la versione **soft margin** delle SVM originali. SVC Possiede un parametro *C* che controlla quanto penalizzare per le istanze classificate male.

Ovviamente SVC fa uso del **kernel trick**. Sono state fatte varie prove per determinare la miglior funzione kernel e, successivamente, il valore migliore di *C*.

Il modello risultante che verrà utilizzato a seguire come comparazione con l’altro modello (la rete neurale), è un **SVM** con:

* Kernel: ***Radial Basis Function***
* Penalità per gli errori di classificazione: ***C*=8**

**Esperimenti (esecuzione di almeno una modalità di validazione e stima delle misure di performance)**

**1 -** **Rete Neurale**

Nel processo di *tuning* degli *iperparametri* sono stati eseguiti diversi esperimenti; in particolare, gli *iperparametri* sono stati testati sequenzialmente e in maniera indipendente; questo procedimento presenta il rischio di trovare una configurazione **sub-ottima** di questi, ma permette di ottenere grandi vantaggi dal punto di vista della **complessità del modello** e del **tempo di addestramento** richiesto (rispetto a quanto impiegherebbe una *grid search*, per esempio). La ricerca degli *iperparametri* ottimi ha seguito il seguente ordine:

1. **Funzione di attivazione** dello **strato nascosto**;
2. **Numero di nodi** dello **strato nascosto**;
3. **Capacità** del **modello** (numero di ottimale di epoche di addestramento)

giustificato dal fatto che la funziona di attivazione influenza la cardinalità ottimale dello strato nascosto, che a sua volta influisce sulla capacità ottimale del modello.

A seguire una breve descrizione degli esperimenti effettuati per ogni *iperparametro*:

* **Funzione di attivazione** dello **strato** **nascosto**: sono state testate tutte le funzioni messe a disposizione dalla libreria *[Keras](https://keras.io/api/layers/" \l "layer-activations)*; le migliori sono risultate essere
  + *ReLU*
  + *SeLU*
  + *ELU*
  + *softplus*
  + *swish*
  + *mish*

e tra queste è stata selezionata la funzione [*mish*](https://paperswithcode.com/method/mish)*,* definita come*:*

* **Numero di nodi** dello **strato nascosto:** partendo da un parametro (pari alla metà del numero dei nodi di input) si è cercata la cardinalità ottimale dello strato nascosto, aggiungendo e sottraendo fino al 50% di questo parametro iniziale, esplorando quindi l’intervallo : la migliore cardinalità è risultata essere 53;
* **Capacità** del **modello:** dato il numero elevato di esempi disponibili nel dataset, era cruciale trovare una capacità del modello che permettesse di evitare l’*overfitting;* per far ciò si è fatto ricorso alle funzionalità di *callback* offerte da *Keras*, in particolare al modulo [*EarlyStopping*](https://keras.io/api/callbacks/early_stopping/), che permette di fermare l’addestramento dopo un certo numero di epoche in cui non si verificano miglioramenti in una certa metrica (per la quale è stata usata la *validation\_lost*). Di conseguenza, il numero ideale di epoche individuato è 22.

Determinati gli *iperparametri* migliori, si è eseguita una **10-folds *stratified cross validation*** sul **training set** originale (non ridimensionato), ottenendo i seguenti risultati sul **validation set**:

|  |  |  |
| --- | --- | --- |
| **Metrica** | **Media** | **Intervallo di confidenza (95%)** |
| **Accuracy** | 0.969 | (0.9607, 0.9782) |
| **Recall** | 0.937 | (0.9189, 0.9541) |
| **Precision** | 0.941 | (0.9196, 0.9829) |
| **F1-Score** | 0.939 | (0.9214, 0.9560) |

**2 -** **Support Vector Machine**

Come per il precedente modello gli *iperparametri* sono stati testati sequenzialmente e in maniera indipendente: questo implica che potrebbe esistere una loro combinazione migliore. Questa scelta di ottimizzazione è dettata dal voler bilanciare le prestazioni finali con tempo di ricerca dei parametri ottimi.

La ricerca degli iperparametri ottimi ha seguito il seguente ordine:

1. Ricerca della funzione **kernel** migliore,
2. Ricerca del valore di **C** migliore.

Abbiamo addestrato le SVM sul **training set** e fatto inferenza sul **validation set,** in vari esperimenti.

Per confrontare le performance abbiamo guardato la **metrica F1**; il modello che massimizza questa metrica verrà ritenuto l’ottimo/migliore per quel batch di esperimenti.

Esperimenti effettuati per ogni **iperparametro**:

* ricerca funzione **kernel:** SVM con C=1 e kernel in {*Lineare,* *Radial Basis Function, Polinomiale, Sigmoideo*}. Confrontando l’F1-score dei vari SVM sugli stessi dati, si determina che il tipo di kernel migliore risulta essere ***Radial Basis Function***.
* ricerca valore di **C** migliore**:** SVM con **kernel** = ***Radial Basis Function*** e **C** in {1, 2, 3, …, 10}. Confrontando l’F1-score dei vari SVM sugli stessi dati, si determina che il **C ottimo** è di **8**.

Determinati gli *iperparametri* migliori, si è eseguita una **10-folds *stratified cross validation*** sul **training set** originale (non ridimensionato), ottenendo i seguenti risultati sul **validation set**:

|  |  |  |
| --- | --- | --- |
| **Metrica** | **Media** | **Intervallo di confidenza (95%)** |
| **Accuracy** | 0.967 | (0.9588, 0.9745) |
| **Recall** | 0.932 | (0.9127, 0.9513) |
| **Precision** | 0.935 | (0.9152, 0.9541) |
| **F1-Score** | 0.933 | (0.9176, 0.9486) |

**Analisi dei risultati ottenuti**

**Rete neurale:**

I valori delle metriche ottenute dalla valutazione sul **test set** della rete neurale sono i seguenti

|  |  |  |
| --- | --- | --- |
| **Metrica** | **Valore** | **Variazione dalla cross-validation** |
| **Accuracy** | 0.968 | - 0.10 % |
| **Recall** | 0.931 | - 0.64 % |
| **Precision** | 0.941 | + 0.00 % |
| **F1-Score** | 0.936 | - 0.32 % |

valori che riflettono minime variazioni da quelli ottenuti tramite **stratified cross validation** e che quindi riteniamo affidabili.

**SVM:**

I valori delle metriche ottenute dalla valutazione sul **test set** dell’SVM sono i seguenti

|  |  |  |
| --- | --- | --- |
| **Metrica** | **Valore** | **Variazione dalla cross-validation** |
| **Accuracy** | 0.953 | - 1.44 % |
| **Recall** | 0.931 | - 0.10% |
| **Precision** | 0.888 | - 5.02 % |
| **F1-Score** | 0.909 | - 2.57 % |

Notiamo che il valore della precisione cala abbastanza sul **test set**, rispetto a quanto performato sulla **stratified cross validation**; tuttavia, in generale rimangono valori soddisfacenti.

**Comparazione dei due modelli:**

**Metriche**

Immagine che contiene testo, schermata, diagramma, Diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma, numero

Descrizione generata automaticamente

Dai risultati ottenuti sulle metriche più importanti si nota un leggero vantaggio di performance sulla rete neurale rispetto al modello SVM.

**Complessità – tempi di addestramento e valutazione**

Immagine che contiene testo, schermata, schermo, diagramma

Descrizione generata automaticamente

Dal punto di vista della complessità dei modelli risulta invece superiore la SVM, che ottiene tempi di addestramento e di valutazione decisamente inferiori a quelli della rete neurale.

Immagine che contiene testo, schermata, linea, schermo

Descrizione generata automaticamente**Soglia di valutazione**

Immagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamente

Infine, analizzando il comportamento dei due modelli a diverse soglie di valutazione, si osservano ottimi valori di **AUC** per entrambe le curve **ROC** e **Precision-Recall**; questo ci permette di avere una buona flessibilità sulla soglia decisionale, permettendo di favorire le prestazioni su ***precision*** o ***recall*** a seconda dello specifico scenario. In particolare, rispetto alle **due possibili applicazioni** considerate:

* riconoscimento di immagini acquisite da sensore montato su **missile antinave**: sarebbe possibile prediligere la ***precision***(alzando la soglia decisionale sopra il valore 0.5, fino ad un valore che ci permette di raggiungere la precisione desiderata),in maniera da rendere il modello **più** **conservativo** e da limitare il più possibile i **falsi** **positivi**;
* riconoscimento di immagini acquisite da droni impegnati in operazioni di **ricerca di navi disperse**: sarebbe possibile prediligere la ***recall***(abbassando la soglia decisionale sotto il valore 0.5),in maniera da rendere il modello **meno** **conservativo** e da limitare il più possibile i **falsi** **negativi**.

**Ablation sul pre-processing dei dati:**

Per confermare la scelta di portare il dataset il scala di grigi, mostriamo gli indici di performance dei due modelli quando addestrati e provati sui dati in formato **RGB**:

**Immagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, diagramma, Diagramma

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, diagramma, numero

Descrizione generata automaticamenteImmagine che contiene testo, schermata, schermo, numero

Descrizione generata automaticamente**

Immagine che contiene testo, schermata, diagramma, schermo

Descrizione generata automaticamenteTutte le misure di performance confermano miglioramenti minimi rispetto al caso in scala di grigi, per i quali riteniamo giustificata la scelta fatta che permettere di lavorare con dati di dimensione ridotta e ridurre la complessità dei modelli, come dimostrato dalla comparazione dei tempi di addestramento e valutazione:

**Conclusioni**

In conclusione, riteniamo di aver sviluppato due modelli di apprendimento automatico con buone performance nel riconoscimento di navi da immagini satellitari, che potrebbero già utilizzati in specifici ambiti o considerati come un ottimo punto di partenza per lo sviluppo di modelli più complessi.