UNIVERSITÀ DI PISA

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

Localizzazione Indoor Basata su Beacon Bluetooth a Bassa Potenza Attraverso Tecniche di Deep Learning

un progetto realizzato per Consorzio Metis e ASL Toscana

Relatori:

Prof. Gianluigi Ferrari

Presentata da:

Marco Pampaloni

Anno Accademico 2019/2020

Sommario

Il problema della Localizzazione Indoor si è rivelato di particolare interesse pratico negli ultimi anni. Questa tesi mostra come moderne tecniche di Deep Learning possano risultare determinanti nella corretta risoluzione di tale problema.

L'approccio analizzato sfrutta una rete neurale convoluzionale (CNN) profonda: l'input del modello è caratterizzato da una serie temporale di segnali broadcast *Bluetooth Low Energy* (BLE) emessi da un insieme di Beacon disposti all'interno dell'edificio adibito alla Localizzazione Indoor, mentre l'output è una coppia di coordinate relative alla posizione all'interno dell'edificio stesso. Sono state inoltre utilizzate varie tecniche di *data augmentation* per produrre un dataset di grandi dimensioni sulla base dei campionamenti dei segnali effettuati in loco.

A seguito dell'addestramento, il modello utilizzato ha mostrato un errore medio assoluto (MAE) sul dataset di test pari a *30cm*, esibendo una discreta affidabilità anche rispetto a variazioni significative dei segnali dovute al rumore ambientale. Un ensemble di modelli, ognuno addestrato con diversi iperparametri, ha permesso di ridurre l'errore medio fino a circa *26cm*.

Il modello prodotto risulta eseguibile in tempo reale su dispositivi mobile con ridotte capacità computazionali, rendendolo particolarmente adatto alla così detta navigazione *"blue-dot"* all'interno di contesti Indoor. Tuttavia si evidenzia come la variazione dell'output del modello possa risultare in una navigazione poco fluida. Per arginare questo problema viene applicato un filtro di Kalman al modello e viene sfruttato il sensore inerziale dello smartphone per produrre un'euristica utile a individuare i movimenti dell'utente.

Indice

Capitolo 1

Introduzione

1.1 Localizzazione Indoor

Il problema della Localizzazione Indoor consiste nell'individuazione di un utente all'interno di uno spazio chiuso e in riferimento a un sistema di coordinate predefinito. Tale sistema di coordinate, relativo ad un determinato edificio, può essere poi espresso in termini georeferenziali conoscendo la precisa dislocazione geografica del locale in questione.
La localizzazione indoor apre le porte a diverse possibilità nel campo dell'esperienza utente all'interno di edifici pubblici, nel settore della gestione dei flussi di persone, della sicurezza e della contingentazione. Attraverso l'impiego di tale tecnologia è possibile coadiuvare la navigazione degli utenti all'interno di edifici complessi e migliorare l'esperienza individuale di persone affette da disabilità. Per ottenere questi risultati è però richiesto un certo grado di precisione, di affidabilità, di efficienza e di sicurezza nella gestione della privacy dei dati di localizzazione degli utenti. Inoltre la tecnologia scelta per risolvere il problema, per essere fruibile, deve avere come ulteriore requisito il basso impatto economico.

1.2 Soluzioni Tecnologiche

Nel corso degli anni sono state implementati diversi sistemi di localizzazione indoor, che possiamo dividere in due macrocategorie: soluzioni ad-hoc e soluzioni che sfruttano tecnologie esistenti. Nel primo caso si fornisce all'utente l'attrezzatura necessaria ad essere localizzato,

mentre nel secondo si utilizza un dispositivo mobile di proprietà dell'utilizzatore. Spesso tale dispositivo è uno smartphone.
I sistemi che implementano tecnologie sviluppate ad-hoc, sono spesso più efficienti, più precisi e flessibili. Tuttavia il loro impiego rimane limitato dall'alto costo di progettazione, installazione e di gestione. È poi richiesto che ad ogni utente che intende essere localizzato sia assegnato un dispositivo che si interfacci col sistema impiegato.
Per l'impiego su larga scala, un sistema di localizzazione indoor deve essere facilmente utilizzabile dalle masse e non deve richiedere particolari requisiti tecnologici.

TODO: Inserire riferimenti bibliografici che mettano in comparazione le varie tecnologie utilizzate, ad-hoc e non, e in particolare mostrino i risultati dei sistemi che sfruttano il Bluetooth.

1.3 Bluetooth Low Energy

La tecnologia *Bluetooth* è talmente pervasiva che ogni smartphone in circolazione ne implementa il protocollo, mostrandosi particolarmente adeguata alla risoluzione del problema in esame. Nello specifico, *Bluetooth Low Energy* (BLE) è un protocollo che riduce notevolmente il consumo energetico dei dispositivi che ne sfruttano le capacità.
La soluzione riportata in questo documento prevede l'utilizzo di una serie di beacon BLE programmabili, ciascuno installato in un punto significativo dell'edificio e configurato per emettere un segnale broadcast con una frequenza di circa 50Hz. La potenza dei segnali viene quindi utilizzata per produrre, attraverso l'utilizzo di una rete neurale artificiale, una coppia di coordinate rappresentative della posizione dell'utente all'interno dell'edificio. Ciò viene reso possibile da una fase preliminare in cui viene mappata la superficie del locale raccogliendo i segnali ricevuti dai beacon in vari punti. Per ogni punto della superficie mappato si registra una serie temporale di segnali, dei quali si considera solo il valore *RSSI*, ovvero la potenza del segnale nel punto in cui questo viene ricevuto.
Il modello utilizzato è di fatto completamente agnostico rispetto all'ubicazione dei beacon installati, fin quando questa sia unica e non mutata nel tempo.

L'utilizzo di tale sistema assicura il completo anonimato dell'utente, il quale non necessita di condividere la propria posizione, essendo quest'ultima calcolata direttamente sul suo smartphone in funzione dei segnali che riceve.

Questa tesi si pone l'obiettivo di descrivere nello specifico la rete neurale progettata per risolvere il problema, le tecniche utilizzate per alzare il grado di precisione del modello e le principali differenze rispetto a modelli già esistenti.

TODO: introdurre breve descrizione dei capitoli

1.4 RSSI e propagazione del segnale

1.5 Variabilità e rumore di fondo: requisiti di usabilità

1.6 Installazione dei Beacon e Acquisizione dei Dati

Capitolo 2

Deep Learning

In questo capitolo saranno introdotti i concetti fondamentali alla base delle moderne tecniche di Deep Learning e le strutture matematiche necessarie alla loro comprensione.

2.1 Machine Learning

Il *Machine Learning*, o apprendimento automatico, è un insieme di tecniche e algoritmi che consente a dei programmi di "imparare" a svolgere un determinato compito sulla base di esperienze pregresse, senza bisogno da parte del programmatore di specificare come eseguire tali mansioni.

2.1.1 Regressione Lineare

Un classico esempio di algoritmo di machine learning è quello della regressione lineare. Scopo dell'algoritmo è predire l'output di una determinata funzione. Si consideri quindi un vettore $\boldsymbol{x} \in \mathbb{R}^n$, e un valore scalare $\hat{y} = \boldsymbol{\theta}^\intercal \boldsymbol{x}$. il vettore $\boldsymbol{\theta}$ introduce i parametri del modello, mentre $\hat{y}$ ne rappresenta l'output, che è una funzione lineare di $\boldsymbol{x}$. Siano quindi $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$ dei vettori in $\mathbb{R}^n$ e $y_1, y_2, \ldots, y_m$ i corrispettivi valori della funzione $f(\boldsymbol{x}_i)$ che stiamo cercando di approssimare.

Perchè la funzione $\hat{y}(\boldsymbol{x})$ approssimi $f(\boldsymbol{x})$ è necessario che i parametri $\boldsymbol{\theta}$ del modello si adattino in modo da minimizzare la differenza tra l'output prodotto dal modello e la cosiddetta *ground truth*: $y = f(\boldsymbol{x})$. A questo scopo si definisce una metrica di errore propria del processo di apprendimento: l'errore quadratico medio (MSE dall'inglese)

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y} - y_i)^2 \tag{2.1}$$

Per minimizzare l'errore sul nostro dataset di test è sufficiente porre a zero la derivata, rispetto a $\boldsymbol{\theta}$, della nostra funzione di costo. Nel caso della regressione lineare è possibile risolvere l'equazione risultante ottenendo un sistema di equazioni che prende il nome di *normal equations*. Esistono tuttavia metodi numerici iterativi che si basano sulle informazioni fornite dal gradiente della funzione di costo che permettono di aggiornare i parametri del modello cercando di ridurne l'errore, anche nel caso di modelli non lineari. L'algoritmo su cui si basano molti dei moderni metodi di apprendimento del Deep Learning è il *gradient descent* o metodo del gradiente.

Il metodo del gradiente aggiorna i parametri $\boldsymbol{\theta}$ del modello secondo la seguente regola:

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{2.2}$$

dove $J(\boldsymbol{\theta})$ rappresenta la funzione di costo associata al modello, mentre η è un coefficiente chiamato *learning rate*. Un'interpretazione dell'algoritmo è data dalle informazioni sulla monotonia ottenute dal gradiente della funzione di costo: per η abbastanza piccolo risulta $J(\boldsymbol{\theta}') \leq J(\boldsymbol{\theta})$ poichè il gradiente negativo di J determina la direzione di massima decrescita della funzione[1]. Ne consegue che applicando ricorsivamente la regola di aggiornamento del metodo del gradiente, la nostra funzione $\hat{y}$ tenderà ad avvicinarsi alla funzione originale y, minimizzando la funzione di costo. Possiamo intepretare il coefficiente η come la velocità con cui seguire la pendenza della funzione di errore.

TODO: Inserire un'immagine che fornisca un'intuizione del funzionamento del gradient descent

Nel caso di un modello di regressione lineare che usa l'MSE come funzione di costo, risulta:

$$\begin{aligned}
&\forall j \in 1, \ldots, n : \\
\frac{\partial}{\partial \theta_j} MSE &= \frac{\partial}{\partial \theta_j} \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \\
&= \frac{\partial}{\partial \theta_j} \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{\theta}^\intercal \boldsymbol{x}_i - y_i)^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial \theta_j} (\boldsymbol{\theta}^\intercal \boldsymbol{x}_i - y_i)^2 \\
&= \frac{2}{m} \sum_{i=1}^{m} (\boldsymbol{\theta}^\intercal \boldsymbol{x}_i - y_i) x_i^{(j)}
\end{aligned}$$

Ovvero abbiamo che la regola di aggiornamento è:

$$\theta_j' = \theta_j - \eta \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \theta_j - \eta (\frac{2}{m} \sum_{i=1}^{m} (\boldsymbol{\theta}^\intercal \boldsymbol{x}_i - y_i) x_i^{(j)}), \quad \forall j \in \{1, \ldots, n\}$$

2.2 Perceptron

Il *Perceptron* è il modello che ha posto le basi per le moderne reti neurali artificiali e il deep learning. Esso prende spunto dalla neurologia, cercando di imitare il comportamento dei neuroni del cervello umano, con ovvie limitazioni e senza presunzione di volerne fornire una simulazione accurata del funzionamento. Una schematizzazione del modello è descritta in Figura 2.1

Il perceptron è molto simile al modello di regressione lineare descritto precedentemente, ma si distingue per un fattore fondamentale: la non linearità. L'output del perceptron è infatti definito come:

$$\hat{y}(\boldsymbol{x}) = g(\boldsymbol{\theta}^\intercal \boldsymbol{x})$$

dove g è una funzione non lineare che, in questo caso, è una funzione *sigmoidea*, cioè una funzione che ha un andamento a "S", con due asintoti orizzontali come in Figura 2.2: di solito è utilizzata la funzione $g(x) = \dfrac{1}{1 + e^{-x}}$.

Malgrado la nonlinearità del modello, il perceptron non è in grado di approssimare molte classi di funzioni. È famoso l'esempio della funzione XOR, la quale non può essere imparata

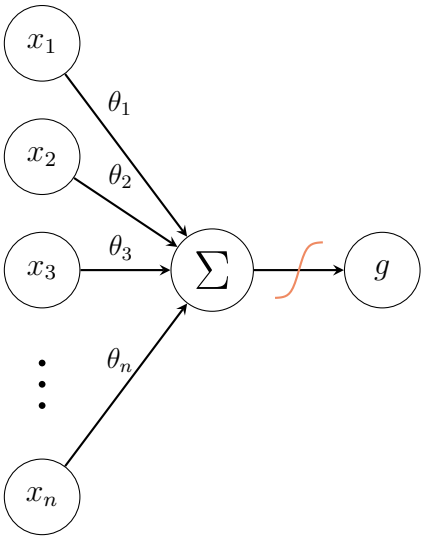

Figura 2.1: Schematizzazione del Perceptron

dal perceptron. Per questo motivo è stata sviluppata un'estensione del modello che prende il nome di *Multi Layer Perceptron*.

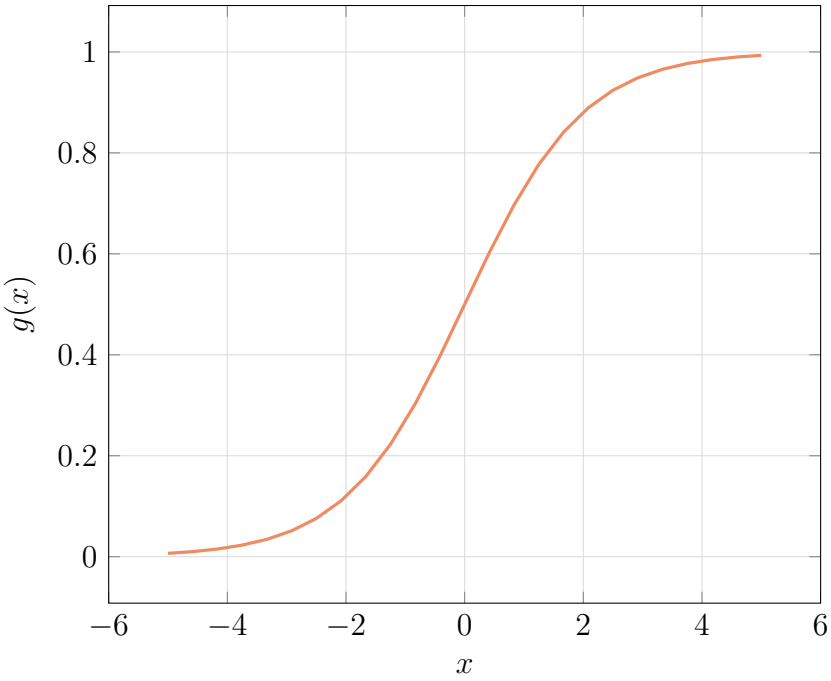

Figura 2.2: Funzione sigmoidea $g(x) = \dfrac{1}{1+e^{-x}}$

2.3 Multi Layer Perceptron

2.4 Attivazione: ReLU

2.5 BackPropagation

2.6 Reti Neurali Convoluzionali

2.7 Regolarizzazione

2.7.1 Overfitting e Underfitting

2.7.2 Regolarizzazione L2

2.7.3 Dropout

2.8 Dataset Augmentation e Preprocessing

2.8.1 Jittering

2.8.2 Ridimensionamento (Scaling)

Capitolo 3

Architettura Software

3.1 TensorFlow

3.2 Keras

3.3 Google Colab

3.4 Weights & Biases

3.5 Modello di Apprendimento

3.5.1 Input del Modello

3.5.2 Blocco Convoluzionale

3.5.3 Uso della Bussola e Output Ausiliario

3.5.4 Coefficiente di Memoria Residua e Input Ausiliario

3.5.5 Output del Modello

3.6 Addestramento del Modello

3.7 Ensembling

Capitolo 4

Applicazione Mobile

4.1 Flutter

4.2 Planimetrie e Poligoni

4.3 Backend TensorFlow

4.3.1 TensorFlow Lite

4.3.2 Implementazione del Bridge di Comunicazione

4.4 Stabilizzazione del Modello

4.4.1 Utilizzo di Sensori Inerziali

4.4.2 Filtro di Kalman

Capitolo 5

Conclusioni

5.1 Risultati Sperimentali

5.1.1 Metriche di Errore: MSE, MAE, MaxAE

5.2 Lavori futuri

5.2.1 Input a Lunghezza Variabile

5.2.2 Reti Neurali Residuali

5.2.3 Variational Autoencoder: Generazione di nuovi dati

5.2.4 Transfer Learning

5.2.5 Input Masking e Ricostruzione dei Segnali

5.2.6 Transformers per Problemi di Regressione

5.2.7 Simulatore BLE

5.2.8 Posizionamento Magnetico

Bibliografia

[1] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.