Università Politecnica delle Marche

Digital Adaptive Circuits And Learning Systems

Sound Event Detection con la tecnica del "*few-shot learning*"

Matteo Orlandini e Jacopo Pagliuca

Prof. Stefano Squartini
Dott.ssa Michela Cantarini

14 luglio 2021

Indice

1 Introduzione

2 Few-shot learning

(https://www.borealisai.com/en/blog/tutorial-2-few-shot-learning-and-meta-learning-i/)

2.1 Introduzione

Gli esseri umani possono riconoscere nuove classi di oggetti da pochissime istanze. Tuttavia, la maggior parte delle tecniche di apprendimento automatico richiedono migliaia di esempi per ottenere prestazioni simili. L'obiettivo dell'apprendimento a breve termine è classificare i nuovi dati dopo aver visto solo pochi esempi di allenamento. All'estremo, potrebbe esserci solo un singolo esempio di ogni classe (one shot learning). In pratica, l'apprendimento a breve termine è utile quando è difficile trovare esempi di training (ad es. casi di una malattia rara) o quando il costo dell'etichettatura dei dati è elevato.

L'apprendimento a pochi colpi viene solitamente studiato utilizzando la classificazione N-way-K-shot. Qui, miriamo a discriminare tra N classi con K esempi di ciascuno. Una tipica dimensione del problema potrebbe essere quella di discriminare tra N=10 classi con solo K=5 campioni da ciascuno da cui allenarsi. Non possiamo addestrare un classificatore usando metodi convenzionali qui; qualsiasi algoritmo di classificazione moderno dipenderà da molti più parametri rispetto agli esempi di addestramento e generalizzerà male. Se i dati non sono sufficienti per ridurre il problema, una possibile soluzione è acquisire esperienza da altri problemi simili. A tal fine, la maggior parte degli approcci caratterizza l'apprendimento a breve termine con un problema di meta-apprendimento.

2.2 The meta learning framework

Nel framework di apprendimento classico, impariamo come classificare dai dati di addestramento e valutiamo i risultati utilizzando i dati di test. Nel quadro del meta-apprendimento, *impariamo come imparare* a classificare in base a una serie di *training task* e valutiamo utilizzando una serie di test task; In altre parole, usiamo un insieme di problemi di classificazione per aiutare a risolvere altri insiemi non correlati. Qui, ogni attività imita lo scenario few-shot, quindi per la classificazione N-way-K-shot, ogni attività include N classi con K esempi di ciascuno. Questi sono noti come support set per il task e vengono utilizzati per apprendere come risolvere questa task. Inoltre, esistono ulteriori esempi delle stesse classi, note come set di query, utilizzate per valutare le prestazioni del task corrente. Ogni task può essere completamente unico; potremmo non vedere mai le classi di un task

in nessuno degli altri. L'idea è che il sistema veda ripetutamente istanze (task) durante l'addestramento che corrispondono alla struttura dell'attività finale di few-shot, ma che contengono classi diverse. Ad ogni fase del meta-apprendimento, aggiorniamo i parametri del modello in base ad un training task selezionato casualmente. La funzione di loss è determinata dalle prestazioni di classificazione sul set di query del task, in base alla conoscenza acquisita dal relativo set di supporto. Poiché la rete viene sottoposta ad un compito diverso in ogni fase temporale, deve imparare a discriminare le classi di dati in generale, piuttosto che un particolare sottoinsieme di classi.

Per valutare le prestazioni in pochi colpi, utilizziamo una serie di attività di test. Ciascuno contiene solo classi invisibili che non erano in nessuna delle attività di formazione. Per ciascuno, misuriamo le prestazioni sul set di query in base alla conoscenza del loro set di supporto.

2.3 Approcci al meta-apprendimento

Gli approcci al meta-apprendimento sono diversi e non c'è consenso sull'approccio migliore. Tuttavia, esistono tre famiglie distinte, ognuna delle quali sfrutta un diverso tipo di conoscenza a priori:

- Conoscenze a priori sulla somiglianza: apprendiamo degli embedding nei training task che tendono a separare classi diverse anche quando non sono visibili.
- Conoscenze a priori sull'apprendimento: utilizziamo le conoscenze a priori per vincolare l'algoritmo di apprendimento a scegliere parametri che generalizzino bene da pochi esempi.
- Conoscenza a priori dei dati: sfruttiamo le conoscenze a priori sulla struttura e la variabilità dei dati e questo ci consente di apprendere modelli praticabili da pochi esempi.

3 Prototypical Network

Nella classificazione few-shot è fornito per ogni episodio un support set costituito da C classi con K istanze $S = \{(x_1, y_1), ..., (x_K, y_K)\}$ dove $x_i \in R^D$ rappresenta il vettore D-dimensionale della feature (nel nostro caso spettrogrammi 128x51) e y_i la rispettiva label.

La rete Prototypical calcola una rappresentazione M-dimensionale c_k, o prototipo, di ogni classe tramite una funzione di embedding f_Φ rappresentata da una rete convoluzionale. Ogni prototipo è il vettore media tra gli

embedding delle istanze della stessa classe.

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\Phi(x_i)$$

Data una funzione distanza d, la rete Protypical calcola la relazione di una query x rispetto ai prototipi tramite la funzione softmax delle distanze prese con segno negativo.

$$p_\Phi(y = k|x) = \frac{exp(-d(f_\Phi(x), c_k))}{\sum'_k exp(-d(f_P hi(x), c'_k))}$$

Il processo di training avviene minimizzando il negativo della log-probabilità $J(\Phi) = -log(p_\Phi(y = k|x))$ considerando la distanza fra query e il prototipo della sua classe. Gli episodi di training sono formati campionando C classi di parole da un audio e selezionando per ognuna casualmente K istanze e Q query.

4 Relation Network

La Relation Network è costituita da due moduli: un modulo di embedding f_φ (equivalente a quello nella Prototypical) e un modulo di relation g_ϕ. Le istanze del query set x_i e quelle del support set x_j vengono date in ingresso al modulo di embedding producendo dei vettori (feature maps) $f_\varphi(x_i)$ e $f_\varphi(x_j)$. Questi ultimi vengono poi concatenati: $C(f_\varphi(x_i), f_\varphi(x_j))$. Le feature map concatenate passano poi attraverso il modulo di decisione che restituisce uno scalare da 0 a 1 che rappresenta la somiglianza tra x_i e x_j. Per il caso C-way one-shot, viene concatenata la query con le istanze dele C classi producendo C punteggi di somiglianza. Nel caso C-way K-shot invece, la query viene concatenata con la somma elemento per elemento degli embedding di ogni instanza delle classi. Quindi, in ogni caso i confronti $r_{i,j}$ sono C per ogni query.

$$r_{i,j} = g_\phi(C(f_\varphi(x_i), f_\varphi(x_j))$$

Per allenare il modello viene usato l'errore quadratico medio (MSE) in modo che l'uscita del modulo di decisone produca 1 se i vettori concatenati sono della stessa classe e 0 altrimenti.

5 Dataset Spoken Wikipedia Corpora

Il progetto Spoken Wikipedia unisce lettori volontari di articoli di Wikipedia. Centinaia di articoli in inglese, tedesco e olandese sono disponibili per

gli utenti che non sono in grado o non vogliono leggere la versione scritta dell'articolo. Il dataset trasforma i file audio in un corpus allineato nel tempo, rendendolo accessibile per la ricerca.[1]

Questo corpus ha diverse caratteristiche importanti:

- centinaia di ore di audio allineato
- lettori eterogenei
- diversi argomenti
- genere testuale ben studiato
- le annotazioni delle parole possono essere mappate all'html originale
- allineamenti a livello di fonema

Ogni articolo è suddiviso in sezioni, frasi e token. Ogni token è normalizzato e la normalizzazione è allineata all'audio.

INSERIRE SCHEMA SWC.RNC!!

5.1 Annotazioni

```
<article>
  <meta>
    <link key="DC.conformsto" value="http://nats.gitlab.io/swc/schema/swc-1.0.rnc"/>
    <prop key="DC.creator" value="Spoken Wikipedia Corpus Collection Software"/>
    <prop key="DC.publisher" value="Universität Hamburg"/>
    <link key="DC.reference" value="http://nbn-resolving.de/urn:nbn:de:gbv:18-228-7-2209
        "/>
    <prop key="DC.type" value="dataset"/>
    <prop key="DC.license" value="CC-BY-SA"/>
    <prop key="DC.title" value="Limerence"/>
    <prop key="DC.language" value="en"/>
    <prop key="DC.identifier" value="Limerence"/>
    <prop key="DC.date.read" value="2005-04-29 00:00:00"/>
    <link key="DC.source" value="https://en.wikipedia.org/wiki/Limerence"/>
    <prop key="DC.source.wikiID" value="154147"/>
    <prop key="DC.source.revision" value="791627969"/>
    <link key="DC.source.text" value="https://en.wikipedia.org/w/index.php?title=
        Limerence&oldid=13811989"/>
    <link key="DC.source.audio" value="https://upload.wikimedia.org/wikipedia/commons/a/
        aa/Limerence.ogg" group="audio1"/>
    <prop key="DC.source.audio.offset" value="0.0" group="audio1"/>
    <link key="DC.source.audio.page" value="https://en.wikipedia.org/w/index.php?title=
        File%3aLimerence.ogg" group="audio1"/>
    <link key="DC.source.audio.date" value="2007-09-14 19:23:05" group="audio1"/>
    <prop key="reader.name" value="the Epopt"/>
    <prop key="processing.step" value="tokenize" group="tokenize"/>
    <prop key="processing.step.date" value="2017-08-10T07:44:44.095+02:00[Europe/Berlin]
        " group="tokenize"/>
    <prop key="processing.step.options" value="Namespace(output=articles/Limerence/
        tokenized.swc, all_sections=false, null_normalize=false, raw_output=null,
        subparser_name=tokenize, lang=en, no_introduction=false, article_dir=articles/
        Limerence)" group="tokenize"/>
    <prop key="processing.step.git.commit.id" value="7431
        dbf93f212ad828208abaf8f518fb8de11ff3" group="tokenize"/>
    <prop key="processing.step.git.commit.time" value="09.08.2017 @ 15:21:55 CEST" group
        ="tokenize"/>
    <prop key="processing.step" value="align" group="align"/>
    <prop key="processing.step.date" value="2017-08-11T16:12:18.423+02:00[Europe/Berlin]
        " group="align"/>
```

```
    <prop key="processing.step.options" value="Namespace(output=articles/Limerence/
        aligned.swc, transcript=articles/Limerence/tokenized.swc, g2p=../model_en/model.
        fst.ser, phone=false, subparser_name=align, dict=../model_en/empty.dic,
        acoustic_model=../model_en/, audio=articles/Limerence/audio.wav)" group="align"
        />
    <prop key="processing.step.git.commit.id" value="7431
        dbf93f212ad828208abaf8f518fb8de11ff3" group="align"/>
    <prop key="processing.step.git.commit.time" value="09.08.2017 @ 15:21:55 CEST" group
        ="align"/>
  </meta>
```

Codice 1: Metadati delle annotazioni delle parole in un audio

```
<d>
  <extra text="LimerenceFrom wikipedia, the free encyclopedia at e n dot wikipedia dot
      org.">
    <s text="LimerenceFrom wikipedia, the free encyclopedia at e n dot wikipedia dot org
        .">
      <t text="LimerenceFrom">
        <n pronunciation="LimerenceFrom" start="140" end="1190"/>
      </t>
      <t text="wikipedia">
        <n pronunciation="wikipedia" start="1250" end="1950"/>
      </t>
      <t text=","/>
      <t text="the">
        <n pronunciation="the" start="1950" end="2070"/>
      </t>
      <t text="free">
        <n pronunciation="free" start="2070" end="2300"/>
      </t>
      <t text="encyclopedia">
        <n pronunciation="encyclopedia" start="2300" end="3220"/>
      </t>
      <t text="at">
        <n pronunciation="at"/>
      </t>
      <t text="e">
        <n pronunciation="e" start="3490" end="3710"/>
      </t>
```

Codice 2: Annotazioni delle parole in un audio

5.2 Lettori

```python
import json

def write_json_file(filename, list_of_dict, indent = 0):
  f = open(filename, "w")
  f.write(json.dumps(list_of_dict, indent = indent))
  f.close

def read_json_file(filename):
  f = open(filename, "r")
  list_of_dict = json.load(f)
  f.close
  return list_of_dict
```

Codice 3: json_manager.py

```
{
    "reader_name": "the epopt",
    "folder": [
        "(I_Can%27t_Get_No)_Satisfaction",
        "Ceremonial_ship_launching",
        "Limerence",
```

```
        "Revolt_of_the_Admirals",
        "Ship_commissioning"
    ]
},
{
    "reader_name": "wodup",
    "folder": [
        "0.999..%2e",
        "Execution_by_elephant",
        "Hell_Is_Other_Robots",
        "Tom_Bosley",
        "Truthiness"
    ]
},
```


Codice 4: Formato JSON dei lettori

5.3 Parole


```
[
{
    "word": "i",
    "frequency": 12,
    "start": [
        660,
        8800,
        115050,
        116300,
        117910,
        228560,
        273900,
        497150,
        518270,
        534740,
        543420,
        589280
    ],
    "end": [
        870,
        8940,
        115240,
        116360,
        118020,
        228690,
        274000,
        497340,
        518520,
        534860,
        543700,
        589360
    ]
},
```


Codice 5: Formato JSON delle parole allineate

Riferimenti bibliografici

[1] Arne Köhn, Florian Stegen e Timo Baumann. «Mining the Spoken Wikipedia for Speech Data and Beyond». Inglese. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* (23–28 mag. 2016). A cura di Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk e Stelios Piperidis. Portorož, Slovenia: European Language Resources Association (ELRA). ISBN: 978-2-9517408-9-1.

[2] Jake Snell, Kevin Swersky e Richard S. Zemel. *Prototypical Networks for Few-shot Learning*. 2017. arXiv: 1703.05175 [cs.LG].

[3] Yu Wang, Justin Salamon, Nicholas J. Bryan e Juan Pablo Bello. *Few-Shot Sound Event Detection*. 2020, pp. 81–85. DOI: 10.1109/ICASSP40776.2020.9054708.