Accelerazione mediante Graphics Processing Units delle reti di Kohonen per apprendimento automatico non supervisionato

Tutor universitario: Mauri Giancarlo
Tutor aziendale: Nobile Marco
Sessione di laurea prevista: Luglio

Mistri Matteo

5 aprile 2018

1 Materiale di riferimento

Per documentarmi sull'argomento SOM ho utilizzato principalmente i papers forniti nella cartella condivisa con noi stagisti, dove è presente diverso materiale. In particolare mi sono concentrato sui documenti *ChangTeng-Exploiting SelfOrganizingMap MedicalImageSegmentation* e *Kohonen-The SelfOrganizingMap*, da cui ho appreso la teoria alla base delle SOM e il metodo di funzionamento di queste ultime.
Ho in seguito ricercato online le diverse implementazioni disponibili di SOM funzionanti, per poter capire quali fossero le funzionalità di interesse e le possibili varianti da implementare. A questo punto mi sono confrontato con il mio tutor aziendale per decidere quali funzioni fossero effettivamente di interesse tra quelle trovate online, e ad eventuali aggiunte.
I prototipi di SOM analizzati per estrapolare le funzionalità sono stati:

- Minisom, dell'utente github JustGlowing
- SOMPY, dell'utente github sevamoo
- kohonen, pacchetto per *R* utilizzato attualmente dal laboratorio

Per quanto riguarda CUDA, ho fatto riferimento al manuale *CUDAbyExample* per la parte pratica, mentre per un background teorico ho utilizzato la parte introduttiva del paper *LASSIE* ed i file linkati nel paper *Graphics processing units in bioinformatics, computational biology and systems biology*.

2 Attività svolte

Inizialmente ho svolto una attività di preparazione e ricerca sull'argomento, documentandomi sui principi di funzionamento delle SOM e sulle varianti presenti.
Presa confidenza con CUDA, ho iniziato a progettare la SOM, limitandomi ad implementare le funzionalità di base e fissando i paramentri a quelli utilizzati nell'esempio fornito dal mio tutor. Dopo aver corretto il codice e aver verificato la compatibilità dei miei risultati con quelli della simulazione, ho inizato ad aggiungere flessibilità alla SOM, permettendo all'utente di modificare i parametri della simulazione e aggiungendo le funzionalità discusse in precedenza con il tutor. Ogni funzionalità aggiunta è stata confrontata con gli esempi forniti dal tutor e nel caso corretta, per mantenere coerenza e correttezza.

3 Problemi risolti

L'implementazione inizale presentava problemi di convergenza del risultato che sono stati risolti modificando le funzioni di decrescita del learning rate e del raggio. Ottenuta la convergenza, la distanza media della SOM dalle reads risultava essere di molto superiore rispetto all'esempio fornito dal tutor, seppur la convergenza risultava procedere allo stesso modo, con steps di decrescita simili.
Inizialmente ho supposto che la distanza della BMU dalla read venisse divisa per il numero di features della read; l'ipotesi, seppur permettesse di ottenere gli stessi risultati della simulazione per il primo esempio, si è rivelata errata a seguito di altri test compiuti su dati differenti.
Data l'assenza di una documentazione dettagliata sul pachetto utilizzato nella simulazione fornita dal tutor, per risolvere i problema sono ricorso al reverse engineering del codice eseguito nella simulazione di esempio(pacchetto kohonen per R), per riuscire a estrapolare la tecnica di calcolo della distanza. Analizzando la porzione di codice interessata, ho scoperto che la distanza tra BMU e read è calcolata come somma dei quadrati delle differenze delle singole componenti; il valore viene poi diviso per il numero di features della read, ne viene fatta la radice quadrata e a questo punto viene nuovamente diviso per il numero di neuroni presenti nella SOM. Utilizzando questa nuova formula per il calcolo della distanza tra BMU e read, ho ottenuto risultati aderenti alle simulazioni sui diversi dati di esempio.
Confrontandomi con il tutor, ho deciso di permettere la scelta del metodo di calcolo della distanza all'utente, così da poter garantire la massima flessibilità del codice.
L'implementazione attuale della SOM fornisce tutte le funzionalità utilizzate nella simulazione su R con il pacchetto kohonen, garantendo una maggiore flessibilità e personalizzazione dei parametri e delle funzioni caratteristiche utilizzate e fornendo funzionalità aggiuntive.

4 Problemi da risolvere

La versione attuale del codice si rivela essere ancora troppo lenta, specialmente su piccole moli di dati. Questo sembra essere dovuto all'overhead introdotto dalla computazione su GPU di piccole quantità di dati, che risulta essere molto svantaggioso rispetto alla CPU.

5 Riusltati premliminari e prossimi passi

Deve essere indagato il problema di prestazioni evidenziato in precedenza, verificando se l'overhead introdotto dalla GPU vada scemando con il crescere della quantità di dati forniti alla SOM.

Il testing dell'implementazione deve essere esteso a nuovi esempi di dati, possibilmente di dimensioni nell'oridne di grandezza dei dati realmente prodotti dalla features extraction su immagini medicali.
Deve inoltre essere indagata la possibilità di effettuare la modifica dei pesi dei neuroni direttamente sulla GPU, in quanto essa si è dimostrata svantaggiosa per piccole moli di dati, ma potrebbe risultare sensata su quantità di dati considerevoli.