B003725 Intelligenza Artificiale (2018/19)

Studente: Alberto Baldrati (6174233) — *<2019-01-22 Tue>*

Elaborato assegnato per l'esame finale

Istruzioni generali

Il lavoro svolto sarà oggetto di discussione durante l'esame orale e dovrà essere sottomesso per email due giorni prima dell'esame, includendo:

1. Sorgenti e materiale sviluppato in autonomia (non includere eventuali datasets reperibili online, per i quali basta fornire un link);
2. Un file `README` che spieghi:
   - come usare il codice per riprodurre i risultati sottomessi
   - se vi sono parti del lavoro riprese da altre fonti (che dovranno essere **opportunamente citate**);
3. Una breve relazione (massimo 4 pagine in formato pdf) che descriva il lavoro ed i risultati sperimentali. Non è necessario ripetere in dettaglio i contenuti del libro di testo o di eventuali articoli, è invece necessario che vengano fornite informazioni sufficienti a *riprodurre* i risultati riportati.

La sottomissione va effettuata preferibilmente come link ad un repository **pubblico** su github, gitlab, o bitbucket. In alternativa è accettabile allegare all'email un singolo file zip; in questo caso è **importante evitatare di sottomettere files eseguibili** (inclusi files `.jar` o `.class` generati da Java), al fine di evitare il filtraggio automatico da parte del software antispam di ateneo!

Apprendimento di parametri in modelli grafici

Nella prima parte di questo elaborato si sviluppa del codice (in un linguaggio di programmazione a scelta) per l'apprendimento dei parametri in reti orientate utilizzando l'approccio a massima verosimiglianza descritto in Russell & Norvig 20.2.1 ed utilizzando Laplace smoothing per evitare stime degeneri. Nella seconda parte, si generano datasets di n esempi a partire da una rete nota che esprime una distribuzione p. I dati possono essere generati ad esempio utilizzando il software Hugin Educational dal menu File/Simulate cases. impostando a zero la percentuale di missing data (la cartella degli esempi del software Hugin contiene reti usabili per gli esperimenti). Alternativamente i dati possono essere campionati dalla rete. Si apprendono quindi i parametri con il codice sviluppato nella prima parte, usando come priors pseudo-counts unitari (Laplace smoothing). Si misura infine la "distanza" tra p e la distribuzione appresa su un data set di dimensione n, q_n, tramite la divergenza di Kullback-Leibler definita come

$$KL(p||q_n) = \sum_U p(U) \log \frac{p(U)}{q_n(U)}$$

Si ripeta per diversi valori di n, formando una learning curve, per verificare che la divergenza si riduce al crescere di n.