Binarized Neural Network

Report sul lavoro fatto per completare il progetto relativo al paper:
https://arxiv.org/pdf/1602.02830.pdf

Intro:

Implementare una rete neurale di tipo Binario significa utilizzare “pesi binari” (+1, -1) e “activations binarie” che durante il training sono utilizzate per calcolare i parametri del gradiente.

Gli esperimenti sono eseguiti con due framework diversi (Theano e Torch), andando ad implementare 3 diversi dataset di riferimento per la classificazione di immagini: MNIST, CIFAR-10 e SVHN.

Il MNIST è un vasto database di cifre scritte a mano che è comunemente impiegata come set di training in vari sistemi per l’elaborazione di immagini (impiegata anche in ML).
Comprende 60000 immagini di addestramento e 10.000 immagini di testing.

Il CIFAR-10 è molto simile, ma contiene 60000 (32x32) immagini a colori in 10 differenti classi (airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks).
E’ molto utile impiegare questo tipo di training per insegnare a una macchina a riconoscere oggetti differenti.

Lo SVHN è un dataset simile a MNIST (ad esempio, le immagini sono di piccole cifre ritagliate), ma incorpora un ordine di grandezza di più dati etichettati (oltre 600.000 immagini di cifre) e proviene da un problema del mondo reale significativamente più difficile e irrisolto: il riconoscimento di cifre e numeri nelle immagini di scene naturali.
(10 classes, 1 for each digit: Digit '1' has label 1, '0' has label 10).

OTTIMIZZAZIONE RETE: SHIFT BASED ADAMAX

Si tratta di un'estensione del classico algoritmo Vanilla Adam
(ottima alternativa alla classica procedura di discesa del gradiente, che tiene conto della media mobile dei momenti m e del gradiente v, usando degli stimatori corretti per contrastare il bias verso zero nelle prime iterazioni)
caratterizzato principalmente dalle seguenti formule:

$$m_t = \beta_1 . m_{t-1} + (1 - \beta_1) . g_t$$

$$v_t = \max(\beta_2 . v_{t-1}, |g_t|)$$

$$\theta_t = \theta_{t-1} - (\alpha \oslash (1 - \beta_1^t)) . (m_t \oslash v_t^{-1})$$

gt is the gradient, $\theta t-1$ is the previous parameter;
α,β1,β2 are learning rate and the betas of Adam optimizer.

Adamax usa la "norma del massimo" del gradiente invece della norma classica nella stima del momento secondo.
Nella versione shift based (formule sopra) notiamo come nel terzo termine è presente il simbolo ⊘ per indicare l'operazione di shifting (non più una semplice moltiplicazione tra float).

Nel framework Keras esiste una versione gia implementata dell'ottimizzatore Adamax:

```
keras.optimizers.Adamax(lr=0.002, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0)
```

Ma non è sufficiente. (A noi serve una versione shift based)

Quindi attraverso la classe **ShiftBasedAdaMaxOptimizer(optimizer.Optimizer)** è possibile ottenerla. Qui abbiamo:
-metodo init in cui (come per adam) sono definiti lr beta1 beta2 ed epsilon (come nel paper), più un richiamo alla super classe SBAMAX che ci permette di riferirci sempre a tale classe (classe base) senza farlo in modo esplicito.

-metodo ap2(x) richiamato che permette il calcolo della potenza di 2 del valore x in cui viene moltiplicato il segno di x per la potenza di 2 di x (arrotondato a int, definito da log e abs).
E' fondamentale per calcolare poi lo shifting! ! ! !

-metodo prepare che converte in tensori i valori specificati (cosi possono essere usati come input futuri).

-con get_slot posso passare esplicitamente i valori delle variabili da inizializzare (var list).

-metodo get_beta_accumulator per ottenere il graph tensor di beta.

-metodo Dense: genera nuovi dtype per lavorare con il gradiente, applica formule sopra (NB forse invertiti v e m); poi c'è l'operazione di shifting che come definito nelle formule sopra viene ottimizzato grazie all'uso di ap2; infine aggiorna tutto. (MAIN METHOD)