Machine Learning
Esercitazione di laboratorio su Reti Neurali

1) MATERIALE NECESSARIO (contenuto nel file .ZIP scaricabile dal sito del corso)

- common.py
- Mlp.py
- Amianto_TrainingData.txt
- Amianto_ValDataMlp.txt
- Amianto_TestDataMlp.txt (fornito durante l'esercitazione)

2) CONOSCENZE UTILI

La libreria scikit learn mette a disposizione la classe `MLPClassifier` per l'addestramento di un classificatore basato su rete neurale multilayer perceptron (pag. 8-32 delle dispense "Reti Neurali").

```
MLPClassifier(  hidden_layer_sizes=(100, ), # livelli hidden e neuroni per livello hidden
                activation='relu',          # funzione di attivazione: provare "tanh" e "relu"
                solver='adam',              # provare "sgd" "adam" (pag. 31)
                alpha=0.0001,               # regolarizzazione L2 (lambda pag. 29)
                batch_size='auto',          # pattern in ogni minibatch (size_mb a pag. 31)
                learning_rate='constant',   # politica di "eventuale" riduzione learning rate
                learning_rate_init=0.001,   # learning rate iniziale (eta a pag. 15-16)
                power_t=0.5,                # usato solo per learning_rate = "invscaling"
                max_iter=200,               # numero massimo di EPOCHE
                shuffle=True,               # se eseguire random sort dei pattern ad ogni epoca
                random_state=None,          # seme casuale
                tol=0.0001,                 # tolleranza per criterio di arresto
                verbose=False,              # se stampare a video dettagli esecuzione
                warm_start=False,           # per un maggior controllo esecuzione (vedi sotto)
                momentum=0.9,               # parametro mu (pag. 30)
                nesterovs_momentum=True,    # ottimizzazione metodo del momento (lasciare True)
                early_stopping=False,       # se attivo, utilizza parte del training set pari a
                validation_fraction=0.1,    #   validation_fraction per implementare arresto
                beta_1=0.9,                 # coefficiente usato solo da Adam
                beta_2=0.999,               # coefficiente usato solo da Adam
                epsilon=1e-08)              # coefficiente usato solo da Adam
```

Nel file Mlp.py l'inizializzazione di un MLP è fatta con `warm_start=True` e `max_iter=1`. In questo modo ad ogni chiamata di `mlp.fit` viene eseguita una sola epoca, consentendo di calcolare la loss e le accuratezze su training e validation set ed eventualmente arrestare l'apprendimento. Alla chiamata successiva di `mlp.fit` l'addestramento proseguirà con i pesi individuati al passo precedente.

3) IL PROBLEMA

Il problema di classificazione da risolvere è lo stesso già affrontato nell'esercitazione dei classificatori. Si tratta di classificare come "Amianto" e "Non amianto" (2 classi) pattern provenienti da un set di immagini aeree acquisite con un drone professionale. I file Amianto_TrainingData.txt e Amianto_ValDataMlp.txt contengono rispettivamente 20000 e 5000 pattern 25-dimensionali etichettati. Ogni pattern è relativo ad una posizione spaziale (x,y) in un'immagine aerea e codifica attraverso 25 feature le caratteristiche di colore e tessitura in un intorno 31×31 centrato in x,y.

4) ADDESTRAMENTO

Addestrare un MLP a partire dal training set e validation set sopra specificati. Il codice Mlp.py implementa già un addestramento di base con iperparametri non ottimizzati. Al termine dell'esecuzione viene visualizzato un grafico che evidenzia l'andamento di loss e accuracy su training set e validation set. Su console è inoltre stampata l'epoca nella quale si è ottenuta l'accuratezza massima sul validation set. Si consiglia di intervenire sugli iperparametri nel seguente ordine (eventualmente riconsiderando successivamente i parametri già valutati):

a. Il file Mlp.py è incompleto e contiene puntini di sospensione "…" laddove è richiesto di completare il codice.
b. Assegnare alle variabili `train_file_path` e `validation_file_path` i percorsi dei file contenenti rispettivamente il traininig e il validation set.
c. Aumentare/diminuire il numero massimo di epoche (nel codice `max_epochs`) per garantire convergenza evitando al tempo stesso *overfitting*.
d. Tarare l'iperparametro `learning_rate_init`.
e. Tarare l'iperparametro `momentum`.
f. Provare `solver='adam'` (nota: Adam non usa momentum).
g. Provare a modificare il numero di livelli hidden e il numero di neuroni sui diversi livelli. Il formato di input richiesto è una tupla contenente il numero di neuroni per ciascun livello hidden. Il default prevede un solo livello hidden con 100 neuroni.

Annotare gli iperparametri della configurazione che ha consentito di massimizzare l'accuratezza sul validation set, e l'epoca e_{max} alla quale si è ottenuta tale accuratezza massima.

5) <u>TEST</u>
Misurare l'accuratezza ottenuta dal classificatore MLP (utilizzando gli iperparametri trovati al punto precedente e arrestandosi all'epoca e_{max} individuata) sul dataset di Test (scaricabile dal sito del corso durante l'esercitazione) per verificarne l'effettiva capacità di generalizzazione. Il file Amianto_TestDataMlp.txt contiene 15000 pattern 25-dimensionali etichettati.