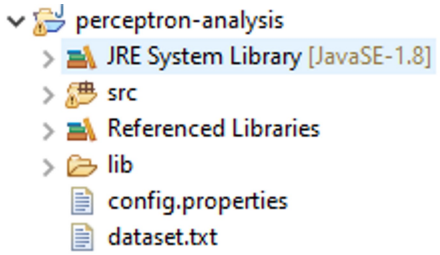
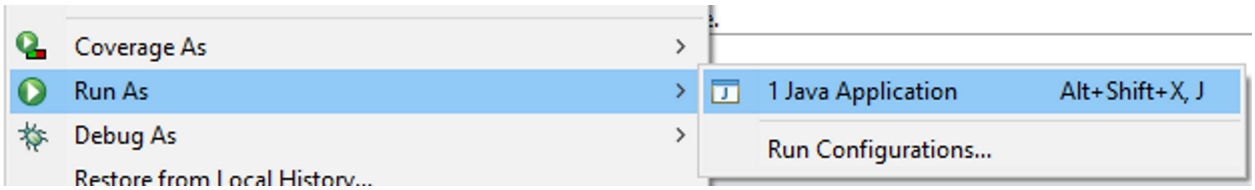
README - Gabriele Nannotti

Per eseguire il progetto:

- Importare il progetto in Eclipse
- Utilizzare un java JRE 1.8 o superiore
- Una volta importato il progetto dentro Eclipse, scaricare il dataset al link https://drive.google.com/drive/folders/1mJVXtLxar-wPL9S8z0wDc0-s3jzaJjcp?usp=sharing
- Copiare il dataset nella directory del progetto



- A questo punto premere col destro sulla cartella del progetto (perceptron-analysis) e selezionare "Run As > Java Application"



- Nel caso in cui venisse richiesto, selezionare la launch configuration di default (quella che esegue il metodo Main)

Descrizioni Classi / Implementazione:

Entry:
Una entry contiene le informazioni di un'esempio del dataset; in particolare contiene due attributi:

float[] x contiene i valori di ogni pixel dei 28x28 pixel che compongono l'immagine (istanza dell'esempio)
Int y contiene il valore della cifra raffigurata all'interno dell'immagine (label dell'esempio) e può assumere valori che vanno da 0 a 9

Il programma addestra dei modelli che effettuano una classificazione binaria delle immagini; è quindi necessario definire due insiemi nei quali le immagini possono essere categorizzate.
Il metodo GetClass restituisce 1 o -1 per categorizzare le istanze di tipo entry in base ai valori contenuti in y, in particolare:

- Se y = 0,6,8,9 allora l'immagine appartiene alla classe 1
- Altrimenti appartiene alla classe -1

LinearPlane:
Gli oggetti LinearPlane contengono i parametri w e b di un piano per la classificazione; inoltre la variabile successes specifica la quantità di esempi classificati bene dal piano durante il training.

Il metodo PlaneValue prende in input una entry e utilizza l'istanza x al suo interno per calcolare il valore della quantità:

$$w\,x + b$$

calcolata in relazione all'array dei pesi w e il bias b contenuti all'interno dell'oggetto.

Il metodo UpdatePlaneParametersOverEntry prende in input una entry e utilizza l'esempio (x,y) contenuto al suo interno per aggiornare l'array dei pesi e il bias del piano.
Questo metodo viene chiamato durante la fase di training.

Model:
Un oggetto model tiene conto dei piani LinearPlanes prodotti durante l'esecuzione del training; quindi fornisce:

- Il piano lastLinearPlane: l'ultimo piano prodotto dal training; questo è utilizzato per calcolare la previsione last (calcolata con lastLinearPlane.PlanePrediction(entry)):

$$h_{last}(x) = sign(w_{last} \cdot x + b_{last})$$

- Il metodo ExecuteModelPrediction: questo produce le predizioni voted e average utilizzando l'intera lista di LinearPlanes e posiziona i risultati negli attributi votedPrediction e avgPrediction; in particolare il loop implementa i classificatori:

$$h_{voted}(x) = sign\left(\sum_{i}^{|listlength|} h_i.successes * sign(h_i(x))\right)$$

$$h_{average}(x) = sign\left(\sum_{i}^{|listlength|} h_i.successes * h_i(x)\right)$$

  sommando iterativamente sulle quantità votedResults e avgResults.

Main:

Il programma può utilizzare più modelli; il numero di modelli utilizzati è specificato da NUMBER_OF_MODELS (eventualmente = 1 se si vuole produrre l'analisi tenendo conto di un solo modello).

Nel caso in cui NUMBER_OF_MODELS > 1, ciascun modello viene addestrato su una differente permutazione del dataset e i dati prodotti per l'analisi terranno conto della media dei dati prodotti dall'analisi di ciascun modello separatamente.

Nel Main è contenuta l'inizializzazione dei modelli utilizzati dal programma e il loop principale.

Il loop principale itera sul vettore TRAINING_ENTRIES_SETS:
Ad ogni iterazione i-esima il programma addestra il modello (o i modelli se NUMBER_OF_MODELS > 1) su ulteriori TRAINING_ENTRIES_SETS[i] esempi nel dataset, li testa sull'intero testset e produce i risultati dell'analisi in relazione alle epoche di training effettuate.

Il file di configurazione contiene:

- dataSetSize: numero di esempi nell'intero dataset
- trainSetSize: numero di esempi del dataset destinati al training
- testSetSize: numero di esempi del dataset destinati al test: sono raccolti dal dataset a partire dall'esempio trainSetSize+1
- attributes: numero di attributi delle istanze x degli esempi

DatasetManager:

DatasetManager gestisce la lettura dal dataset; il dataset è un file di testo contenente dataSetSize righe ciascuna delle quali è composta da 784 (28x28) valori separati da virgola,

ognuno dei quali indica il valore di grigio di un determinato pixel, e da una cifra che va da 0 a 9 che indica la cifra rappresentata nell'immagine associata ai 28x28 pixel.

Il dataset è caricato dal metodo LoadDatasetRam sull'array di stringhe datasetRam; ogni elemento di datasetRam corrisponde a una riga del dataset (esempio del dataset).

I primi trainSetSize elementi di datasetRam corrispondono agli esempi di training, i successivi testSetSize corrispondono agli esempi per il testing

Durante il training ogni modello utilizzato nel programma legge dal dataset in un'ordine differente; l'istanza di DatasetManager deve gestire l'ordine di lettura per il training dei vari modelli.

Per non caricare più versioni permutate dello stesso dataset in memoria si crea una permutazione degli indici con cui leggere dal dataset; queste permutazioni sono contenute nell'array datasetIndexesPermutations (inizializzati nel metodo InitializeIndexesPermutations).

Per leggere il j-esimo esempio per il training del modello i-esimo si legge da datasetRam l'elemento: datasetRam[datasetIndexesPermutations [i] [j]]

Oltre all'ordine di lettura, DatasetManager tiene conto per ciascun modello i-esimo il numero j del prossimo esempio da leggere; questa informazione è mantenuta nell'array indiceLettura[i] = j contenuto nell'istanza DatasetManager.

Il metodo ParseNextTrainEntry carica sull'oggetto entry (fornito negli argomenti di input) l'esempio su cui deve essere addestrato il modello i-esimo che corrisponde all'esempio j-esimo contenuto in indiceLettura[i].
Quindi ParseNextTrainEntry incrementa l'indice di lettura del modello i-esimo eventualmente riazzerandolo nel caso si sia raggiunto l'ultimo elemento del trainingSet.

Il metodo ParseInputTestset carica nell'oggetto entry (fornito negli argomenti di input) l'esempio di indice entryIndex del testset.

Infine DatasetManager calcola la massima norma delle norme delle istanze nel dataset attraverso il metodo calculateMaxNorm.

Training:

Questa classe contiene l'implementazione dei passi per l'algoritmo di training.

Il metodo run addestra il modello di indice modelIndex su numberOfEntries entries del trainingset lette secondo l'ordine specificato da DatasetManager.

L'addestramento inizia dall'ultimo LinearPlane contenuto nella lista dell'oggetto Model.
Per ogni iterazione di addestramento si legge una entry dal dataset e si guarda se il piano corrente la classifica correttamente:

- Se è classificata correttamente si aumenta il numero di successi del piano
- Se non è classificata correttamente
    - si crea un nuovo piano clonandolo dal piano corrente
    - Si mette in cima alla lista di piani nel modello il nuovo piano
    - si mette il nuovo piano nel piano corrente e lo si aggiorna sulla entry su cui si è appena sbagliato a classificare; quindi si passa alla prossima iterazione

Testing:

Gli oggetti Testing elaborano e contengono i dati sul test di un modello testato sull'intero testset.

Il metodo run esegue il test del modello di indice modelIndex su tutte le entries del testset e tiene conto di quante volte sbaglia a predire con i diversi classificatori last, voted e average.