MSDS spring2020

Deep learning and data mining Robert Azencott

HW2

1 Describe your Data set and the associated classification task

i suggest to reduce the number of classes to the 3 largest classes C1 C2 C3 of sizes siz1 > siz2 >siz3

if siz3 is much smaller than siz2 and siz1, inflate C3 artificially by adjoining to C3 one or two or even three copies of itself to get a better balanced classification problem

- indicate the number of cases N in the data set (after reduction of number of classes to 3)

- present in words the distinct classes C1 C2 C3 involved and their sizes

- Select a training set and a test set ; give the size of each class within the training set and within the test set ; give the proportions of cases in each class for the test set and for the training set; these two proportions should be similar for each class

2 Define the MLP architecture of an Automatic Classifier with r =3 classes

- select an MLP architecture with three layers L1, H, L2, extended by a Softmax function and its final output layer OUT

L1 => H => L2=> softmax => OUT

of respective dimensions

p = dim(L1) = # descriptors, h = dim(H) , dim(L2) = 3 ; dim(OUT) =3

- select a response function (either the sigmoid function or the RELU function) to be set for all neurons

- the goal of the MLP classifier is to provide for each input vector Xk an output probability vector OUTk very close to the binary vector encoding the true class of the input Xk

- explain precisely what is the part played by the softmax function, and how the final classification of Xk is computed from OUTk

3. Select 2 tentative sizes h for the hidden layer

To estimate one small but plausible value for h, namely h = h95 < p , apply PCA analysis to your training set of input vectors to generate p eigenvalues ordered in decreasing order

Plot this decreasing sequence of eigenvalues in yr report

Compute the smallest number "h95" of eigenvalues preserving 95% of the total sum of eigenvalues

To estimate one larger plausible value hL for the size h , proceed as follows

- apply PCA analysis to the set of Mj input vectors corresponding to the class Cj, with j=1,2 3 to generate Mj eigenvalues in decreasing order, and compute the smallest number "Uj" of eigenvalues preserving 95% of the total sum of these Mj eigenvalues;

- then define hL = U1 + U2 +U3.

3. for each one of the 2 values h= h95, h=hL implement automatic training

- use gradient descent to minimize avCRE= average CROSS ENTROPY error between computed and true values of the probability OUTk

- explain precisely what is aCRE during training and after each epoch

- indicate which software environment you will use for HW2

- indicate precisely which software functions you are choosing to implement MLP learning with aCRE loss function;

-list clearly what are the options offered for this task, namely for initialization of the weights, for batch learning , for the successive gradient descent steps sizes ε(n), for stopping the learning, for intermediary outputs to monitor learning quality

- indicate your selections for batch size B, for time dependent gradient descent step size, for gradient descent algorithm, for stopping the learning, for *random* initialization of weights and thresholds

automatic learning typically implements successive steps

- at STEP(n-1) select a new batch BATn containing B cases, and apply the learning rule to update the last vector of weights and thresholds W(n-1) into a new vector Wn which includes both thresholds and weights .

- compute the batch average cross-entropy error bavCREn by running the MLP parametrized by Wn on the current Batch BATn, and compute the vector Gn = gradient of BACREn at step n by the formula Gn = (1/ ε(n)) [W(n+1)- W(n)]

- compute and plot the curve n➔ baCREn

- compute and plot the curve n➔ || W(n+1)- W(n)|| / ||Wn||

- compute the gradient norm ||Gn|| = (1/ ε(n)) || W(n+1)- W(n)|| and

the (fixed) dimension D of the gradient vector Gn,

- plot the curve n➔ ||Gn|| / d where d is the square root of D

- comment on these three curves

4 Performance analysis

Do the following after each Epoch(m) , m= 1,2, ...

denote MLP(m) the MLP classifier parametrized by the last weight vector computed during Epoch(m)

compute the performance indicator trainPER(m) = percentage of correct classifications computed by MLP(m) on the whole training set and the similar indicator testPER(m) computed on the whole test set

on the same graph plot the two curves trainPER(m) and testPER(m) versus m; compare the two curves to select the best epoch Epoch(m*) after which the learning should be stopped to avoid overfit

after the last epoch Epoch(m*) compute and interpret the 3x3 confusion matrix of MLP(m*)

5 Impact of various learning options

Evaluate experimentally the impact on final performance of various factors such as

changes in batch size , initialization, gradient descent step size , dimension h of H

6 Analysis of the hidden layer behaviour after completionof automatic learning

- call h* the best of the two values h95 and hL from the point of view of performance evaluations

- fix h* and m* and MLP*= MLP(m*) ; perform and interpret a PCA analysis on the hidden layer activity vectors H(1) ...H(k)... H(N) generated by all the training inputs X(k)

- display and compare the average hidden neurons activity profiles PROF1 , PROF2, PROF3 where PROFj is the average of the H(k) over all cases case(k) belonging to class j

- list the hidden neurons which achieve best DIFFERENTIATION between class C1 versus C2 ;
- do the same for C1 versus C3 , and then for C2 versus C3; interpret these results