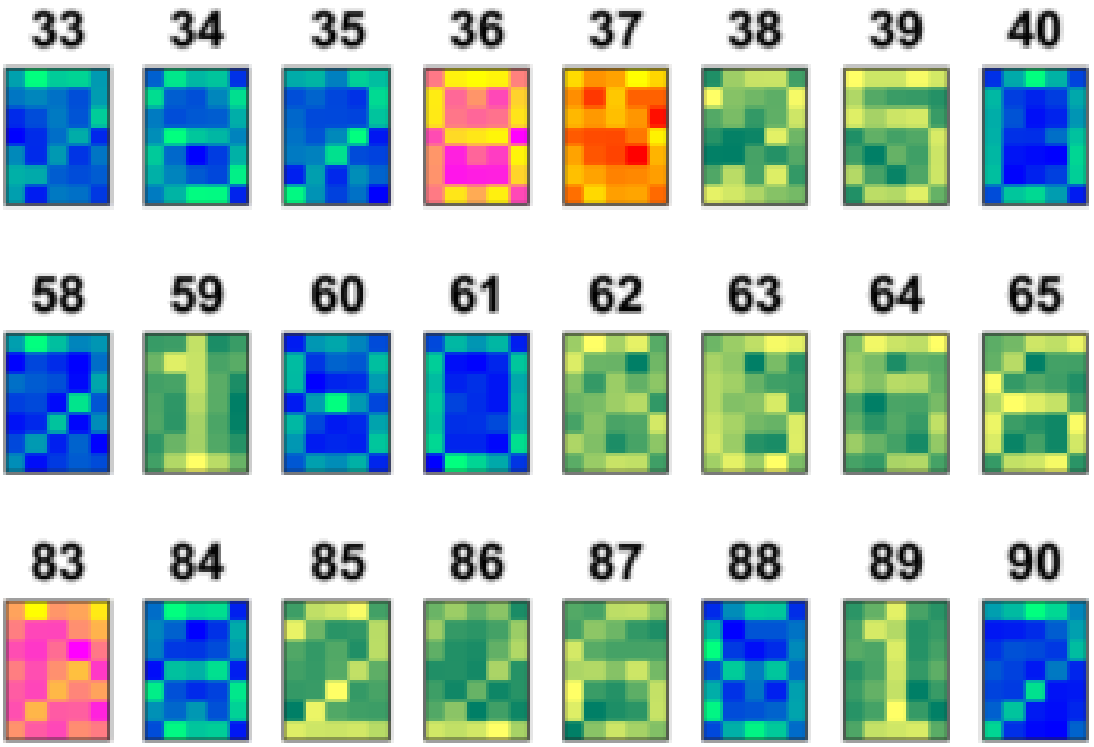
Pattern recognition with Single Layer Neural Network (SLNN)

Optimització Matemàtica, Lab Assignment 1



Gabriel Fortuny Carretero i Pau Mateo Bernadó

Optimització Matemàtica

UPC, Grau en Ciència i Enginyeria de Dades

Curs 2023-2024

tr_seed = 54316377, te_seed= 41606561, sg_seed= 24010204

ÍNDEX

Introducció

L'objectiu del projecte de la primera part de l'assignatura d'Optimització Matemàtica és crear una xarxa neuronal Single-Layer per poder identificar nombres mitjançant mètodes de primeres derivades. El que fem és crear una base de dades de dígits, els quals estan formats per vectors de 35 píxels, amb un nombre de train i un altre de test. Aquests darrers serviran per fer-ne prediccions amb els resultats obtinguts amb el primer conjunt. El nostre objectiu és que la xarxa identifiqui els 10 dígits del 0 al 9. Per això, el que volem és minimitzar la funció de pèrdua, feta a partir de regularització L2 i un paràmetre lambda. Per fer-ho apliquem els mètodes prèviament coneguts del Gradient (GM) i Quasi-Newton BFGS. A més a més, també en fem servir un de nou: el Mètode del Gradient Estocàstic (SGM). Voldrem comparar les qualitats i rendiments de la convergència i la precisió de cada mètode. També veurem l'efecte que té el paràmetre lambda en aquests. És important esmentar que el mètode per trobar el line search emprat és el backtracking line search amb algunes modificacions, perquè es pugui suportar la xarxa neuronal.

En el fitxer .zip es poden trobar tots els codis creats i emprats per poder dur a terme el projecte. Hem fet dos nous codis de uo_nn_batch.m per poder diferenciar la part 3 del projecte i les dues parts del report, que es diuen batch_report1 i batch_report2, de manera que tenim també tres sortides en format csv. Els algorismes del Mètode del Gradient i del Mètode Quasi Newton BFGS estan escrits al document uo_nn_solve_proj.m i el SGM a uo_nn_SGM. La resta hem emprat els codis que es donaven com a plantilla: uo_nn_main.m, uo_nn_batch.m i uo_nn_solve.m i els uo_nn_BLSNW32.m, uo_nn_Xyplot i uo_nn_dataset.m sense modificar-los.

1) Estudi de la convergència

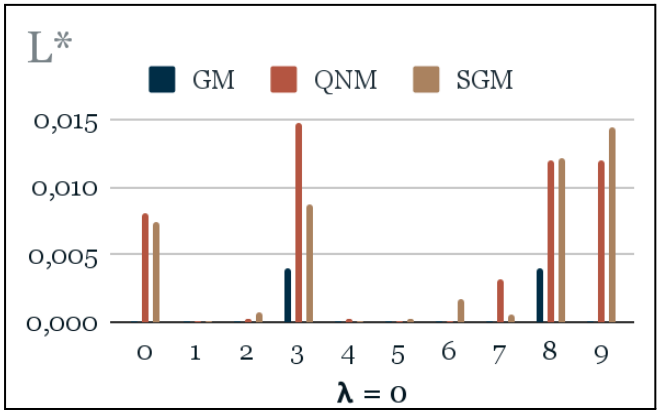
En aquest apartat es discuteix i analitza la convergència dels tres mètodes en relació a la funció objectiu L. Volem veure si hi ha diferències entre els tres mètodes i entre els valors de lambda. Executem el fitxer uo_nn_batch.m amb les seeds demanades. En aquest cas: tr_seed = 54316377, te_seed = 41606561, sg_seed = 24010204.

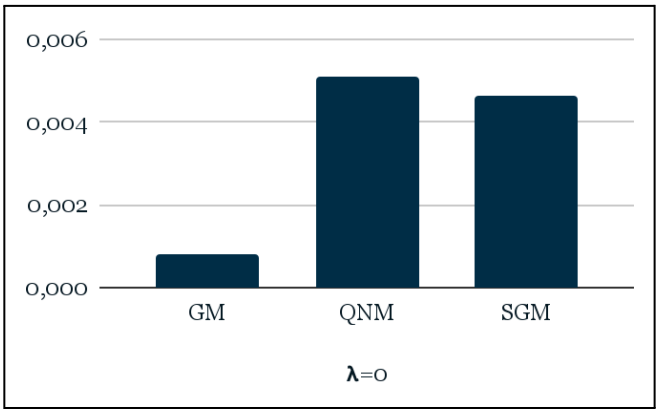
a) Convergència global

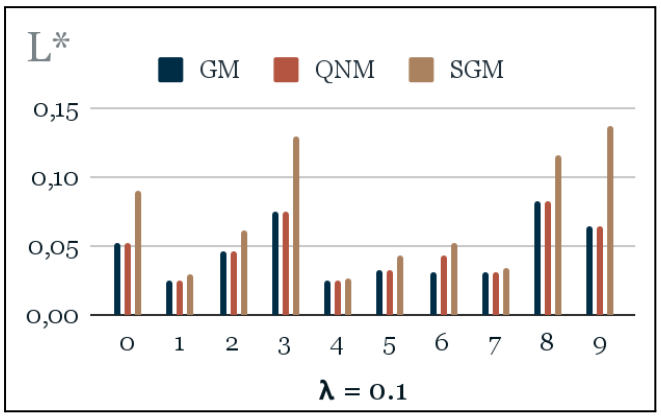
En primer lloc, observem que en la majoria de casos tots els mètodes convergeixen; és a dir, que l'algorisme s'atura abans d'arribar al màxim nombre d'iteracions que hem establert perquè considera que ha trobat un mínim local. El Mètode Quasi-Newton BFGS convergeix sempre, però el Mètode del Gradient no ho fa en dues ocasions: es tracta dels casos on **λ** = 0 i els nombres objectiu són 8 i 9; en aquest el nombre d'iteracions arribat és 1000, el qual ens indica que el programa no arriba a convergir a una solució òptima. El Mètode del Gradient Estocàstic sembla que convergeix quasi sempre, doncs és més mal de veure degut a que el nombre d'iteracions que fa no és constant i depèn d'altres paràmetres. Així i tot, hi ha dos casos on fa moltes més iteracions que a la resta, propers a 125000, que són casos on efectivament les iteracions del bucle *epochs* arriben a les màximes establertes. Són els casos de **λ** = 0, amb els números objectiu 1 i 4. Com que les condicions de terminació del SGM són diferents dels altres dos mètodes, hem d'adoptar una intuïció diferent sobre com es comporta l'algorisme: que el SGM arribi a les itercions màximes vol dir que en cap moment ha fet iteracions (del bucle *epoch*) sense millorar la funció objectiu (*loss function*, en aquest cas) de les permeses. És a dir, que sempre va trobant millors solucions, per poc diferents que siguin. Ara, això tampoc vol dir que el vector de coeficients no sigui bo i que no obtingui les mateixes mètriques de *training accuracy* i *test accuracy* que quan sí que convergeix; al contrari, per als dos únics casos on no convergeix per iteracions, l'algorisme obté 100% tant de *training accuracy* com de *test accuracy* i un valor de la funció d'error de menys de 10^-4.

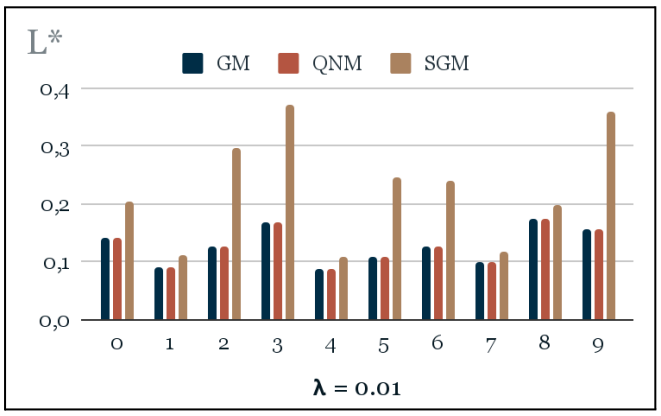
Per tant, veiem que el valor del paràmetre de regularització **λ** està relacionat amb la convergència dels algorismes (els casos on no tenim convergència és justament quan no regularitzem!). No només és crucial per evitar *overfittings* i controlar la complexitat del model, sinò que també té un impacte en la convergència (i en la velocitat de convergència, discutit més endavant), ja que la regularització limita els valors del vector *w* i, per tant, simplifica i fins i tot millora la funció a optimitzar.

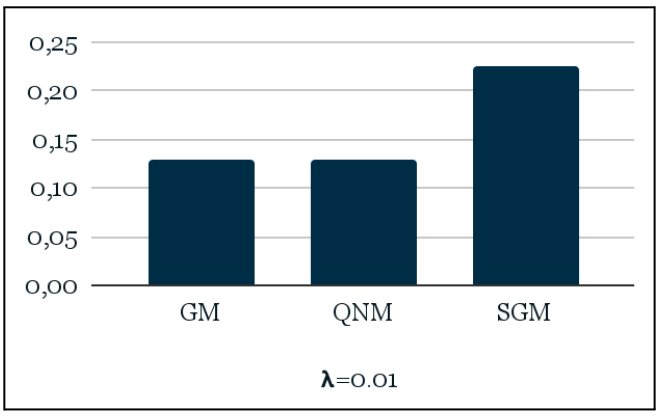
A continuació mirem a quin valor de la funció *loss* convergeixen les diferent configuracions de **λ** - algorisme.











A les gràfiques de l'esquerra es mostra el valor de la funció d'error que cada algorisme obté per cada nombre diferent a identificar, separant pels tres casos de **λ**, i als gràfics de la dreta la mitjana d'aquests valors (separant també per **λ**).

Com es pot veure, els algorismes obtenen, en mitjana, un valor més baix de la funció error amb **λ** = 0, i en particular, el GM és el que obté el valor mínim. Pel que fa als altres valors de

λ, els tres algorismes convergeixen a un valor més baix de la funció error amb **λ** = 0.1, sent així **λ** = 0.01 el pitjor dels tres casos de **λ**.

Cal notar que per **λ** = 0 es veu una clara diferència entre diferents números: per als números 1, 2, 4, 5, 6 i 7 els tres algorismes obtenen un valor de la funció error molt baix (per sota de 0.0025 en pràcticament tots els casos) però després per a la resta de números arriben a valors bastant més alts (tots per sota de 0.15, tot i així). Aquesta tendència també es veu, tot i que no de forma tan marcada, amb els altres valors de **λ**. Això ens fa intuïr que aquests darrers números són més difícils de predir correctament, potser degut per a la similitud entre ells.

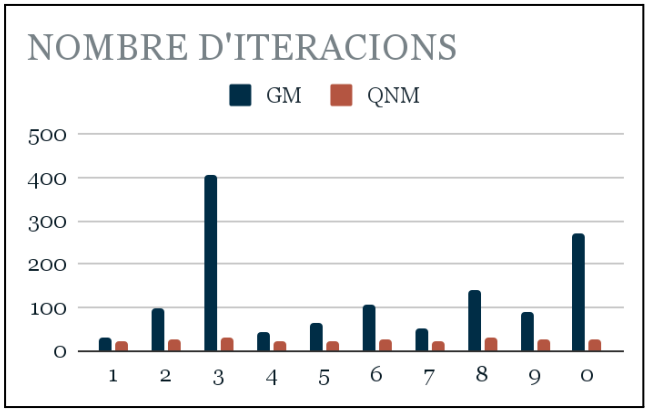
I pel que fa a la diferència de convergència entre els algorismes, l'algorisme SGM tendeix a estar sempre per sobre dels altres dos algorismes, els quals sempre convergeixen a valors molt similars (exceptuant el cas **λ** = 0, on obtenen valors molt diferents per certs números target).

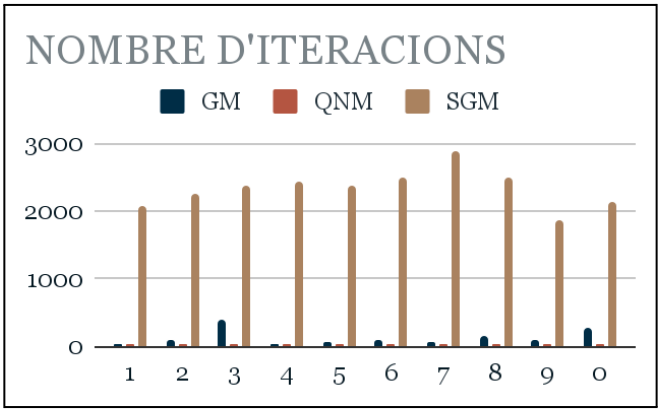
b) Convergència local

Per poder comparar la convergència local per cada algorismes i valor diferent de lambda ens fixem en els valors dels paràmetres *tex* (temps d'execució de l'algorisme) i *niter* (nombre d'iteracions). A més a més, hem tret els 4 casos on l'algorisme no convergeix per tal que aquest anàlisi sigui més rigurós i es basi realment en la velocitat de convergència i no en si convergeix, el qual té més a veure amb la convergència global.

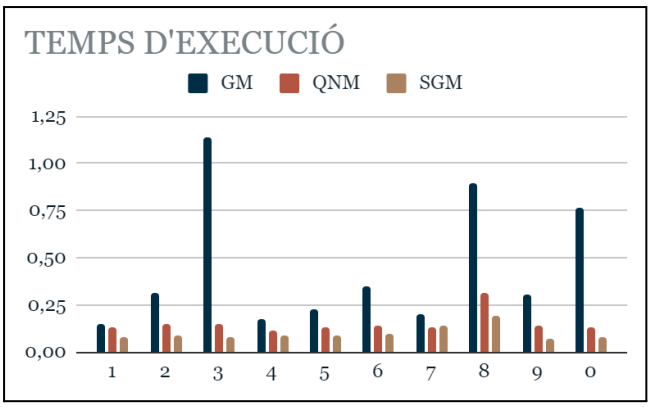
i.

En primer lloc, volem comparar la convergència local en funció de l'algorisme emprat.





Aquests gràfics mostren, per a cada algorisme i per a cada número *target*, el nombre d'iteracions i el temps d'execució mitjà dels tres valors de **λ**. Hem decidit fer un segon gràfic de les iteracions per poder veure millor la diferència entre l'algorisme BFGS i GM.



El que es veu a simple vista és que per a tots els números *target*, el mètode SGM sempre és el que més ràpid va, a continuació el mètode BFGS, i el mètode GM és el que més tarda.

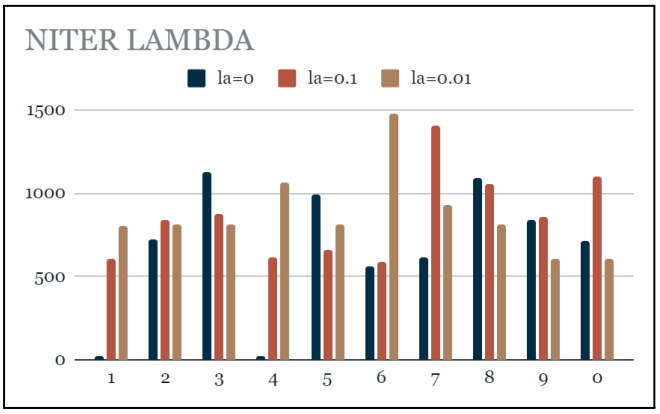
S'ha de tenir en compte però que aquests resultats són sense tenir en consideració els casos on els algorismes no convergien: pel SGM són 1 i 4 amb **λ**=0, que tenen un temps de 2,7 i 2,6 segons, respectivament, i pel GM, també amb **λ**=0 i pels números *target*, que tenen un temps de 2.9 i 4.0 segons.

Pot semblar que amb el Mètode del Gradient i pel cas del número *target* 3 l'algorisme no ha convergit perquè el promig del temps d'execució i de les iteracions són molt alts, però és un cas particular on, amb **λ**=0, el Mètode del Gradient fa 975 iteracions i triga 2.66 segons, arribant a convergir com a tal però per poc.

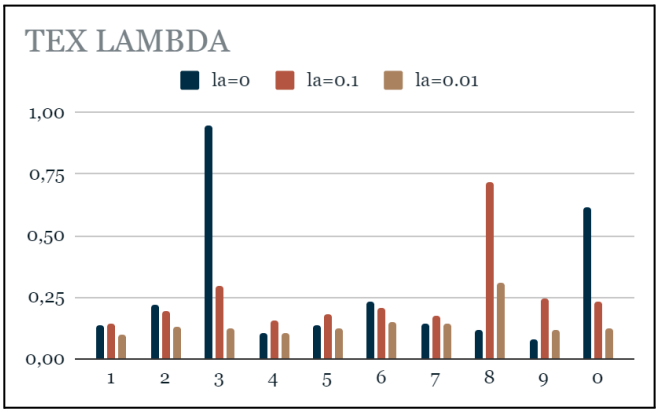
En relació al nombre d'iteracions, veiem que les iteracions del Mètode del Gradient Estocàstic tenen un altre ordre de magnitud. En mitjana, aquest fa 2300 iteracions, mentre que els altres dos en fan 129 (GM) i 24 (QN-BFGS). Això es deu a que les iteracions es calculen de manera diferent; en aquest model comptem una iteració cada vegada que el mètode fa una volta en el bucle interior (*minibatches*), mentres que per als altres mètodes només comptem les iteracions principals (no comptem les iteracions de la cerca lineal, per exemple). Aquest fet ens fa preveure que el rati temps per iteració del mètode SGM serà significativament menor al dels altres dos. Entre els altres dos mètodes, el que menys iteracions fa és el Quasi-Newton, que en general en fa bastant poques i troba resultats similars al GM. Per això, tot i ser ambdós mètodes de primeres derivades, el QN-BFGS té una convergència local més ràpida.

ii.

A continuació volem comparar la rapidesa dels algorismes tenint en compte les lambdes; per això calculem la mitjana dels temps d'execució i nombre d'iteracions per cada nombre objectiu.



El primer que observem del gràfic del nombre d'iteracions segons la lambda emprada és que no sembla que hi hagi massa relació entre les lambdes i els nombres. Sí es veuen alguns valors que despunten però no és una tendència que es repeteixi en tots els nombres. Cal destacar que hi ha dos casos on el nombre d'iteracions és especialment baix. Es tracta de quan lambda val zero i l'objectiu són els nombres 1 i 4, en els quals entre el GM i BFGS fan menys de 50 iteracions. Recordem que per fer el gràfic s'han tret les iteracions del SGM en aquests nombres degut a que no convergeix.

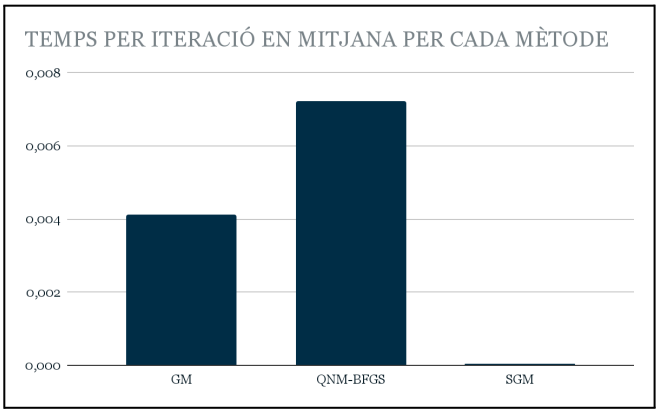


Quan ens fixem en el temps d'execució veiem el mateix; no existeix una lambda que tengui un temps mitjà significativament menor a la resta. Així i tot cal destacar que la λ = 0 té els valors més grans i els valors atípics. A més a més, la lambda $\boldsymbol{\lambda}$ = 0.01 és amb la qual s'observen temps menors. Potser serà més interessant comparar conjuntament els mètodes amb les diferents lambdes. Es fa més endavant a l'apartat 1c).

iii.

El darrer que analitzem per tenir una idea de la convergència lineal és el rati de temps per iteració.

Això és important, doncs podem tenir un mètode que fa moltes iteracions però és ràpid, a diferència d'un que en fa poques però té un temps d'execució molt major. El gràfic de la dreta ens dona una idea molt simple de que el temps per iteració de cada mètode és molt diferent. De fet,



és aquí on més observem que cada mètode té un funcionament molt diferent. Clarament, el Mètode del Gradient Estocàstic té un rati molt inferior, de l'ordre de 10^{-5} segons per iteració. Això es deu al que ja havíem comentat: la manera de comptar les iteracions i el fet que és un mètode ràpid fan que tingui el menor rati. Com hem anat veient, els altres dos mètodes són més comparables. En aquest cas, el QN-BFGS té un temps per iteració major que el GM, degut a que és un mètode que empra molt poques iteracions, tot i que és un poc més ràpid en general que el GM. Aquest darrer té una mitjana de 0,004 segons per iteració.

c) Rendiment general

En general, en l'estudi de la convergència global hem constatat que quan lambda és igual a zero, els mètodes difereixen molt. El que millors solucions troba és el Mètode del Gradient i, dels altres dos, el Quasi-Newton és el que té resultats un poc pitjors. Així i tot, amb lambda igual a zero s'aconsegueixen els valors menors per la funció de pèrdua pels tres mètodes. Això canvia pels altres dos valors de lambda, on aquests dos mètodes donen resultats molt semblants i el que en dona de pitjors és el Mètode del Gradient Estocàstic. En relació a la convergència local, estudiar segons la lambda únicament no ha donat resultats massa clars i l'únic que hem tret és que el mètode més ràpid i a la vegada el que més iteracions fa és el SGM. A més, el GM té alguns valors atípics i és el que més triga i més iteracions fa dels altres dos.

Per això, hem volgut comparar en una taula els valors de mitjana de temps per les combinacions de lambda i mètode. N'hi ha dues que donen uns resultats molt propers, que són amb lambda zero el QN-BFGS i el SGM.

RENDIMENT		**λ = 0**	**λ = 0.01**	**λ = 0.1**
GM	**NITER**	**419,6**	**109,2**	**35,6**
	TEMPS EX (s)	**1,3266**	**0,421**	**0,1838**
QN-BFGS	**NITER**	**12,8**	**40,9**	**17,5**
	TEMPS EX (s)	**0,08**	**0,235**	**0,13**
SGM	**NITER**	**1969,75**	**2426**	**2563,5**
	TEMPS EX (s)	**0,079**	**0,1055**	**0,1088**

D'aquesta manera, es conclou que la relació lambda-algorisme per la minimització de la funció L més eficient és, per molt poc, la combinació **λ = 0** i Mètode Quasi Newton. Ens

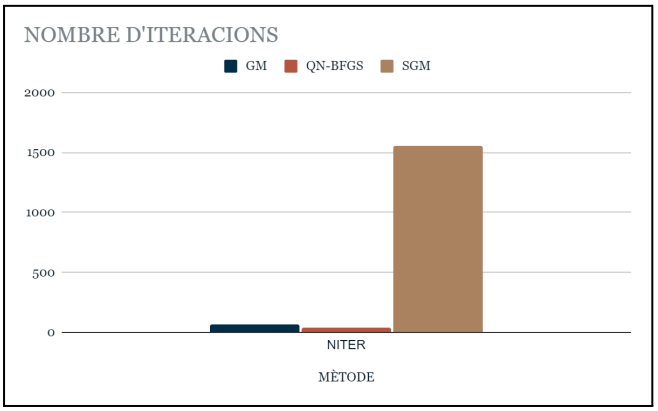
decantem per aquesta opció perquè és el mètode més ràpid en general per la convergència local i dona bons resultats en la convergència global per aquest valor de lambda. A més a més, és el que menys iteracions fa i és l'únic que convergeix sempre, amb una mitjana de 12,8 iteracions i 0,08 segons d'execució.

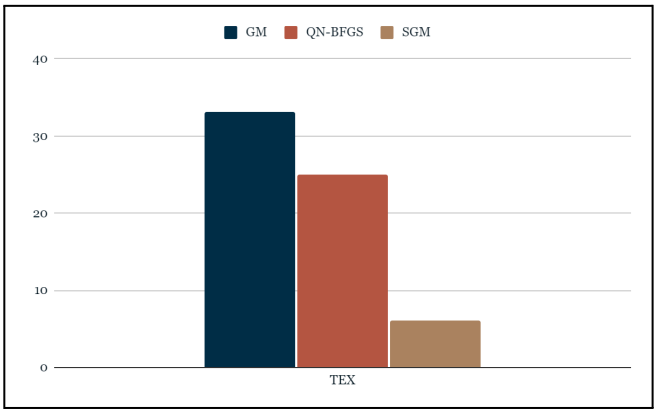
2) Estudi de precisió de reconeixement

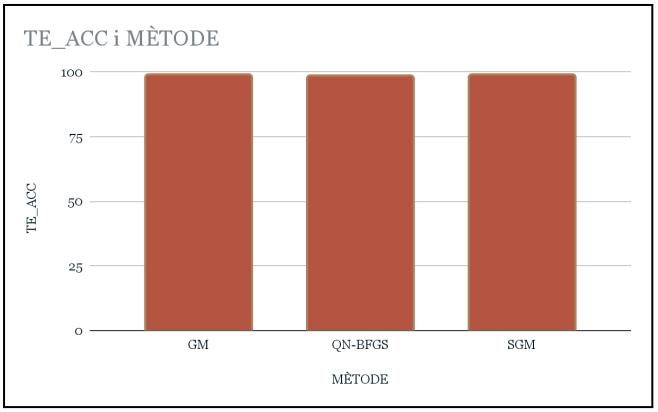
Per a aquest apartat hem utilitzat, per a cada algorisme, el valor de lambda que, per al dataset de p = 250, ho ha fet millor en quant a test accuracy. Aquests valors han resultat ser $\boldsymbol{\lambda}$ = 0.1 per als algorismes GM i QN-BFGS i $\boldsymbol{\lambda}$ = 0 per al SGM.

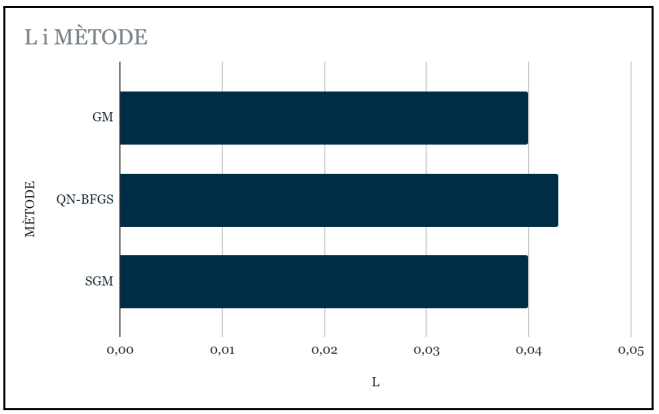
a) Anàlisi dels resultats

Volem analitzar el rendiment dels diferents algorisme per aquests valors més realistes. Per això tornem a graficar els resultats, en aquest cas amb les mitjanes dels temps d'execució, el nombre d'iteracions, el valor de la funció trobat quan convergeix i, sobretot, el valor de la precisió (accuracy) per la base de dades del test. Tots els algorismes han convergit, tot i tardar més que abans degut a la major quantitat d'informació que han manejat.









En els gràfics observem que el nombre d'iteracions segueix la mateixa tendència que anteriorment: el SGM en fa moltes més que els altres dos, del qual el QN-BFGS és el que en fa menys en mitjana. Recordem que estem emprant els valors de lambda que millor accuracy han donat en la part 1. Si ens fixem en el gràfic del temps d'execució veiem de manera molt clara els tres algorisme diferenciats. En mitjana, el SGM és l'algorisme més ràpid amb diferència, on el segueixen el QN-BFGS, quatre vegades més lent i, per últim, el GM. On hi ha molta igualtat és en els resultats obtinguts. Doncs com podem observar, els valors de la funció de pèrdua trobat quan els algorismes convergeixen és, en mitjana, molt similar. Finalment, si mirem les precisions (accuracy) dels diferents algorismes veiem que en general tots obtenen prediccions molt bones, per sobre del 96% en quasi tots els casos. A més a més, les mitjanes de tots els mètodes s'assemblen força. Cal destacar que pel nombre 1 tots tres obtenen una accuracy del 99,9% i l'única que baixa del 96% l'obté el mètode QN-BFGS pel nombre 0, amb un 91,2%.

En conclusió, si hem de declarar un dels algorismes com el que millor rendiment proporciona aquest és el Mètode del Gradient Estocàstic. Tot i que tots tres tenen molta

precisió i donen resultats de L bons, el SGM és el mètode més ràpid. Aquesta millora en la convergència local és la que fa que es diferenciï dels altres dos.

b) Discussió AccuracyTE - L

La millor combinació que havíem trobat a l'apartat 1.c) per a termes de minimitzar la funció d'error ha estat el mètode de Quasi-Newton BFGS amb $\boldsymbol{\lambda} = 0$, ja que en general era el que millor comportament tenia per a la convergència local, i obtenia també molts bons resultats en quant a la convergència global, similars als altres dos mètodes, només que aquest convergia per a tots els casos i els altres no.

En canvi, a l'apartat anterior (2.b) hem conclòs que la millor combinació de cara a maximitzar l'error de test era el mètode SGM, amb $\boldsymbol{\lambda} = 0$, que no és la mateixa combinació. Per una banda, no podríem haver triat la mateixa combinació que abans perquè, tot i que per al mètode BFGS el valor de lambda que minimitzava la funció d'error era $\boldsymbol{\lambda} = 0$, de cara a maximitzar la precisió de test anava millor $\boldsymbol{\lambda} = 0.1$, i per aquest segon apartat ja no hem considerat la combinació BFGS més $\boldsymbol{\lambda}=0$. Per l'altra, els tres algorismes han mostrat tenir una precisió del test molt similars, i el mètode SGM ha destacat només pel seu baix temps d'execució.

3. Conclusions

Hem aconseguit crear i ajustar una xarxa neuronal d'una sola capa (SLNN) que és capaç d'identificar els dígits del 0 al 9 correctament amb alta precisió. Hem comparat les diferents configuracions de la xarxa amb els algorismes Mètode del Gradient, Quasi-Newton BFGS i Mètode del Gradient Estocàstic. Hem vist com varia l'ajust de la xarxa en funció de l'algorisme emprat i de l'hiperparàmetre λ corresponent a la regularització de la funció objectiu. Hem conclòs que la millor configuració en quant a maximització de la precisió en el conjunt de test és utilitzar el Mètode del Gradient Estocàstic (SGM) sense regularitzar (λ = 0), amb la qual hem obtingut una precisió del 99.9%. A més, l'entrenament d'aquest algorisme era bastant més ràpid al dels altres algorismes d'optimització.

Com a línies de futur, seria interessant explorar més el paràmetre de regularització, provant més valors per veure si és possible obtenir millors configuracions, i analitzar el rendiment de la xarxa en funció del desenfocament dels dígits.