Pattern recognition with Single Layer Neural Network Report

Arnau Turch i Inés Broto

8/5/2020

En aquesta pràctica hem realitzat una xarxa neuronal que pretén identificar els digits del 0 al 9 mitjançant l'optimització d'una funció Loss utilitzant 3 mètodes diferents: el Mètode del Gradient, el Mètode Quasi Newton BFGS, i per últim, el Mètode del Gradient Estocàstic.

Podem comprovar en primer lloc, que la funcio que estarem minimitzant és convexa, ja que la seva hessiana és semidefinida positiva. Aquest fet ens permet estar segurs dels resultats dels mètodes ja que sol existirà un mínim local i global.

Hem realitzat dos vegades aquest experiment, modificant el SGM. A l'hora de trobar la direcció d, en un cas la dividiem pel tamany de la mostra utilitzada per trobar l'aproximació del gradient i en l'altre cas no. Comparant els resultats hem observat que obteniem un millor rendiment en el primer cas, així com millor precisió. Per aquesta raó hem decidit treballar amb el primer cas ja que només es tracta d'un escalat dels valors.

Les seeds utilitzades en aquest report són, per al *training*: 369354 i per al *test*: 169306

Carreguem les dades

```
##        ID num_target la  isd niter tex tr_acc   te_acc L.
## 1 0.01  1          0  1    3 0.060 100    100 2.44e-10 NA
## 1 0.03  1          0  3    3 0.029 100    100 2.56e-10 NA
## 1 0.07  1          0  7 1000 0.091 100    100 3.28e-01 NA
## 1 1.01  1          1  1  104 0.638 100    100 3.88e+00 NA
## 1 1.03  1          1  3   21 0.164 100    100 3.88e+00 NA
## 1 1.07  1          1  7 1000 0.104 100    100 1.21e+01 NA
```

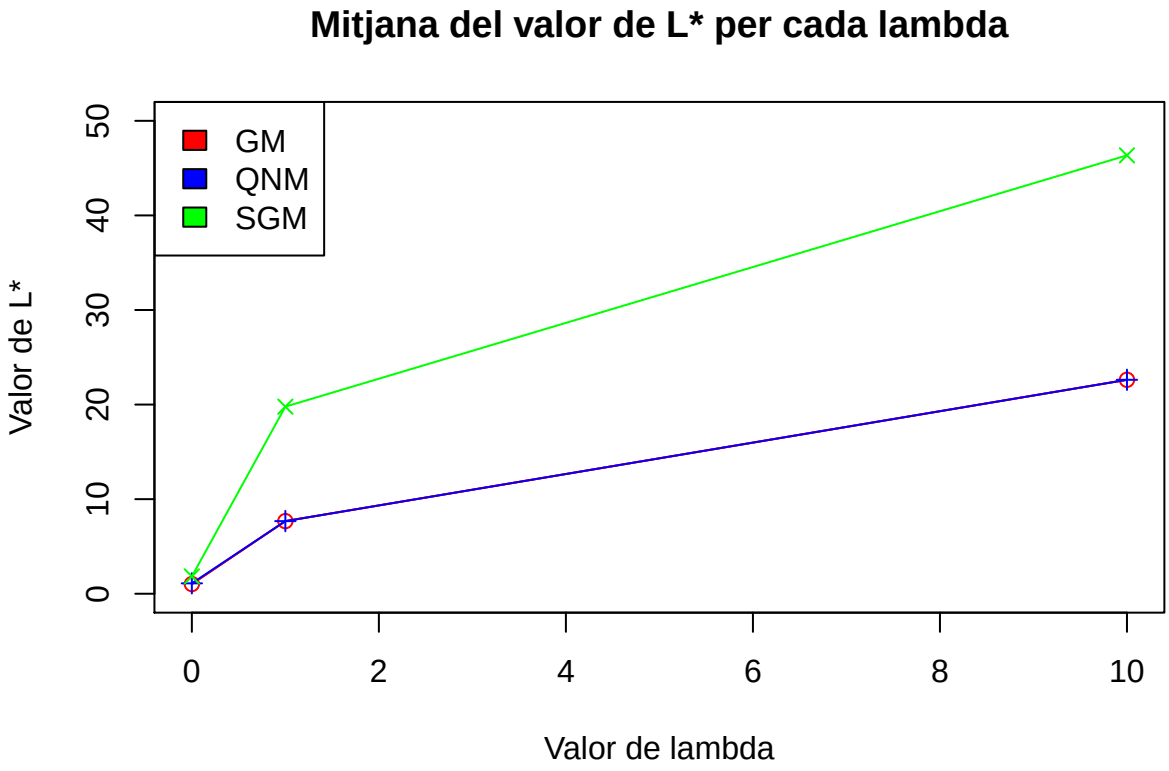
Observem que al llegir les dades el nom de les columnes ens queda desplaçat, corretgim aquest error i afegim una nova variable *speed* que utilitzarem més endavant.

```
##        num_target la isd niter   tex tr_acc te_acc       L.       speed
## 1 0.01          1  0   1     3 0.060    100    100 2.44e-10 0.020000000
## 1 0.03          1  0   3     3 0.029    100    100 2.56e-10 0.009666667
## 1 0.07          1  0   7  1000 0.091    100    100 3.28e-01 0.000091000
## 1 1.01          1  1   1   104 0.638    100    100 3.88e+00 0.006134615
## 1 1.03          1  1   3    21 0.164    100    100 3.88e+00 0.007809524
## 1 1.07          1  1   7  1000 0.104    100    100 1.21e+01 0.000104000
```

Part 1

Convergència dels algoritmes: primerament, estudiarem la convergència global i local dels tres algoritmes a partir de la funció objectiu L.

a) Convergència Global




```
##     no_conv
## GM        6
## QNM       0
## SGM      30
```

Com es pot observar, el mètode del gradient estocàstic no arriba mai al punt òptim abans de les 1000 iteracions, com podem observar també en la gràfica anterior, ja que el punt on acaba sempre és superior als òptims trobats pels altres dos mètodes.

El mètode Quasi Newton sempre acaba abans de les 1000 iteracions i convergeix globalment.

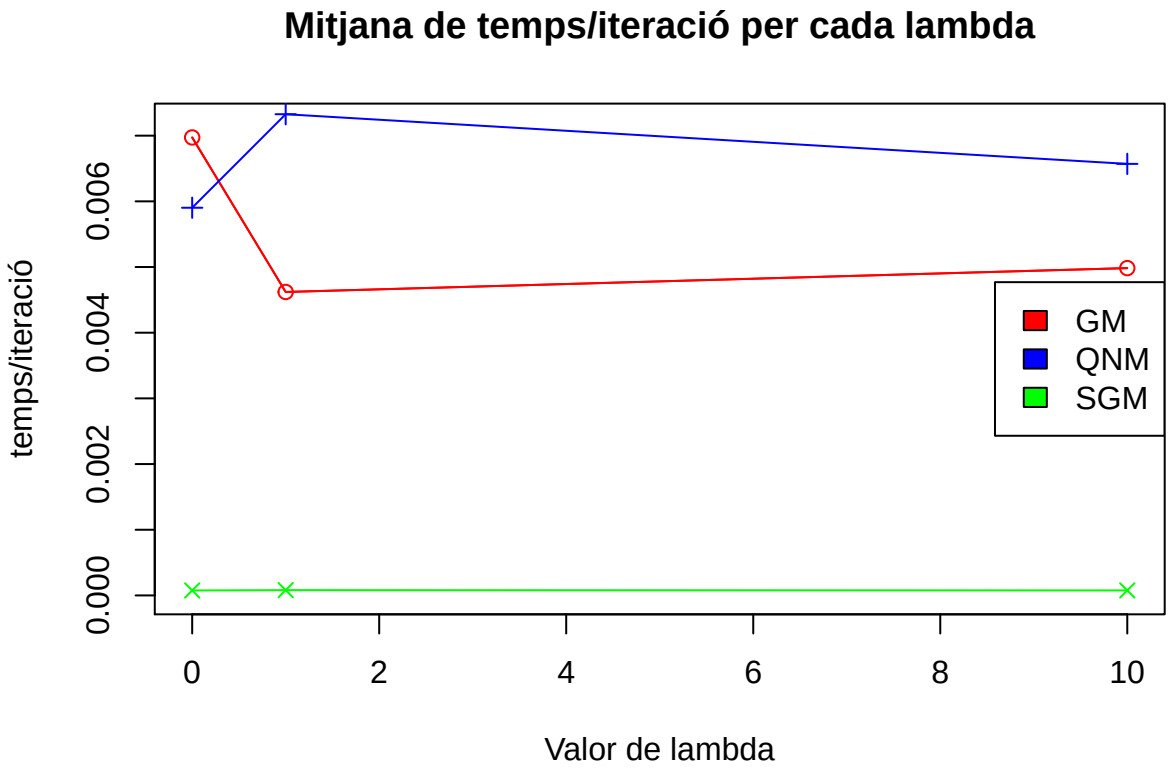
El mètode del gradient observem que en alguns casos no arriba a la solució abans de les 1000 iteracions, podem buscar en quins casos son:

```
##        num_target la isd niter   tex tr_acc te_acc       L.    speed
## 210.01          2 10   1  1000 3.398   96.0   97.2 2.22e+01 0.003398
## 3 0.01          3  0   1  1000 4.204  100.0   97.6 9.37e-02 0.004204
## 8 0.01          8  0   1  1000 3.747  100.0   94.4 1.40e-01 0.003747
## 810.01          8 10   1  1000 3.384   92.4   90.0 3.32e+01 0.003384
## 9 0.01          9  0   1  1000 4.125  100.0   96.8 2.38e-04 0.004125
## 0 0.01          0  0   1  1000 4.164  100.0   98.0 7.04e-06 0.004164
```

Es detecta que aquest fet succeeix sovint quan el valor de λ és igual a 0, però encara que en aquests casos no convergeixi, el valor de la funció objectiu en la ultima iteració és molt proper a l'òptim com podem comprovar en la gràfica ja que els valors de la funció objectiu utilitzant aquest mètode són molt semblants als del mètode Quasi-Newton, que si que convergeix.

També podem observar que al augmentar la λ, el valor del punt òptim de la funció objectiu augmenta. De la mateixa manera quan la λ augemnta, el mètode del Gradient Estocàstic queda més lluny de l'òptim.

b) Convergència Local




```
##       local_conv
## GM  5.525412e-03
## QNM 6.599893e-03
## SGM 7.743333e-05
```

El mètode que va més ràpid és el SGM, aixo es deu a que fa els càlculs amb una part de les dades per calcular el gradient de manera aproximada però amb menys temps.

El GM és el segon més ràpid cosa que podiem esperar perque realitza els mateixos càlculs que el mètode del gradient però amb totes les dades, de manera que troba el gradient exacte a cada iteració.

Per últim el QNM és el més lent ja que realitza càlculs extra a l'hora de buscar la direcció de descens i a més a més també ho fa amb totes les dades.

c) Discusió

GM

El mètode del gradient assegura la seva convergència global ja que per definició la d és sempre de descens $d^k = -\nabla f^k < 0, \forall k$ i sempre es satisfà la CAC : $cos(-\nabla f^k, d^k) = cos(0) = 1 \geq \delta > 0, \forall k$. Pel que fa a la convergència local, sabem que aquest metode convergeix linealment, és per això que en la majoria de casos troba el punt òptim en poques iteracions. De totes maneres, tal i com hem vist, no sempre acaba en menys de 1000 iteracions i això es deu a les variacions que el valor de lambda genera a la funció objectiu i a la seva Hessiana en el punt òptim.

QNM

Per aquest mètode també podem assegurar la convergència global tenint en compte com hem definit la matriu que utilitzem per calcular la direcció de descens. Com que aquesta matriu la calculem pel mètode de *BFGS update* estem aconseguint una aproxicació de la matriu Hessiana definida positiva si la longitud de pas satisfà les SWC. D'aquesta manera, la direcció és sempre de descens $(\nabla f^k)^t * d^k = -(\nabla f^k)^t * H^k BFGS * \nabla f^k < 0, \forall k)$ A més a més, la CAC també es compleix sempre que la matriu $H^k BFGS$ sigui definida positiva i tingui una condició numèrica fitada superiorment per una constant positiva $cos(-\nabla f^k, d^k) \geq \delta > 0$. Si considerem la convergència local del QNM, hem vist a les lliçons de teoria que és superlineal. Això fa que convergeixi en tots els experiments que hem realitzat i que a més a més, la mitjana d'iteracions que triga a trobar l'òptim és notablement més baixa que pels altres dos mètodes.

```
##      iter_mean
## GM   348.76667
## QNM   28.63333
## SGM 1000.00000
```

SGM

En aquest mètode no podem garantir la convergència global i per tant tampoc la local ja que de la manera com definim la direcció d, utilitzant sol una part de les dades, no podem assegurar que la direcció que utilitzarem serà de descens. A més a més tampoc podrem garantir que complim les CAC. Es per això que veiem que el SGM sempre arriba a les 1000 interacions.

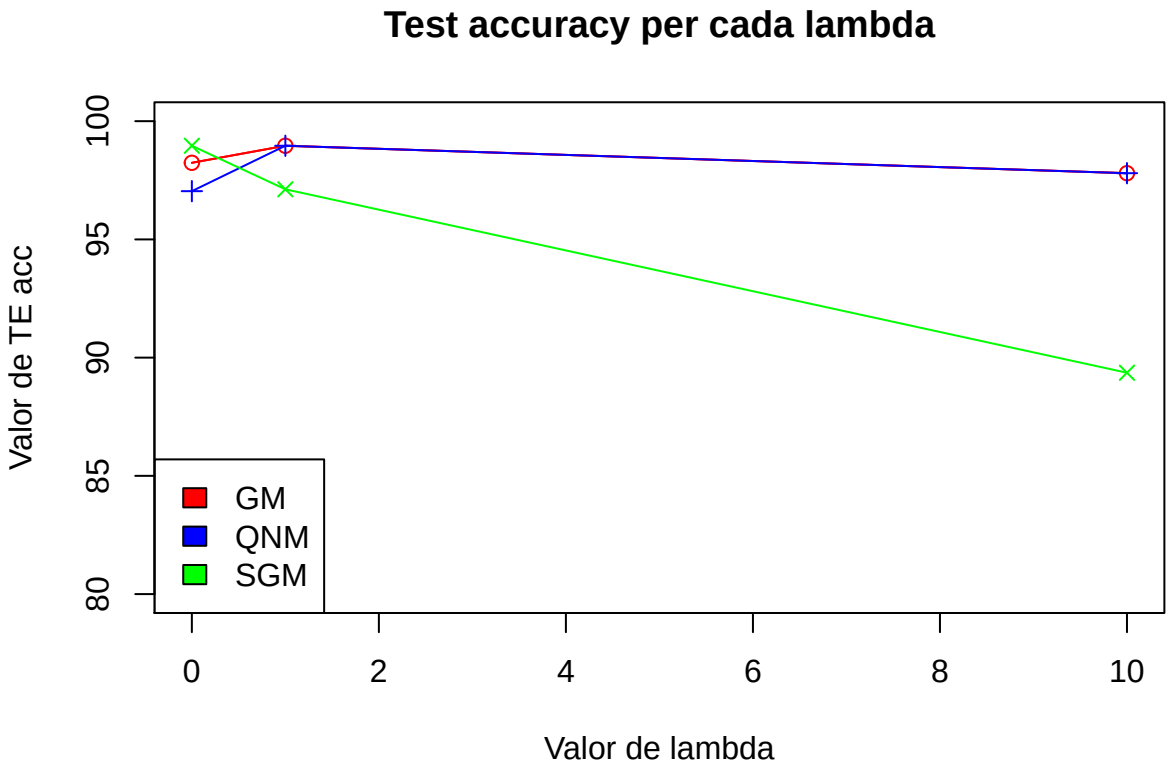
Si comparem totes tres metodologies podem concloure que segurament, la més adecuada pel que fa a l'eficiència és el mètode del gradient estocàstic amb $\lambda = 0$ ja que, malgrat no podem assegurar la seva convergència global, arriba a un valor de la funció molt proper a l'òptim real en polt poc temps.

Part 2

Precisió de reconeixement: Ara analitzarem la precisió de reconeixement de la nostra xarxa neuronal comparant l'encert de test tant per cada λ-algoritme com per cada dígit de manera individual.

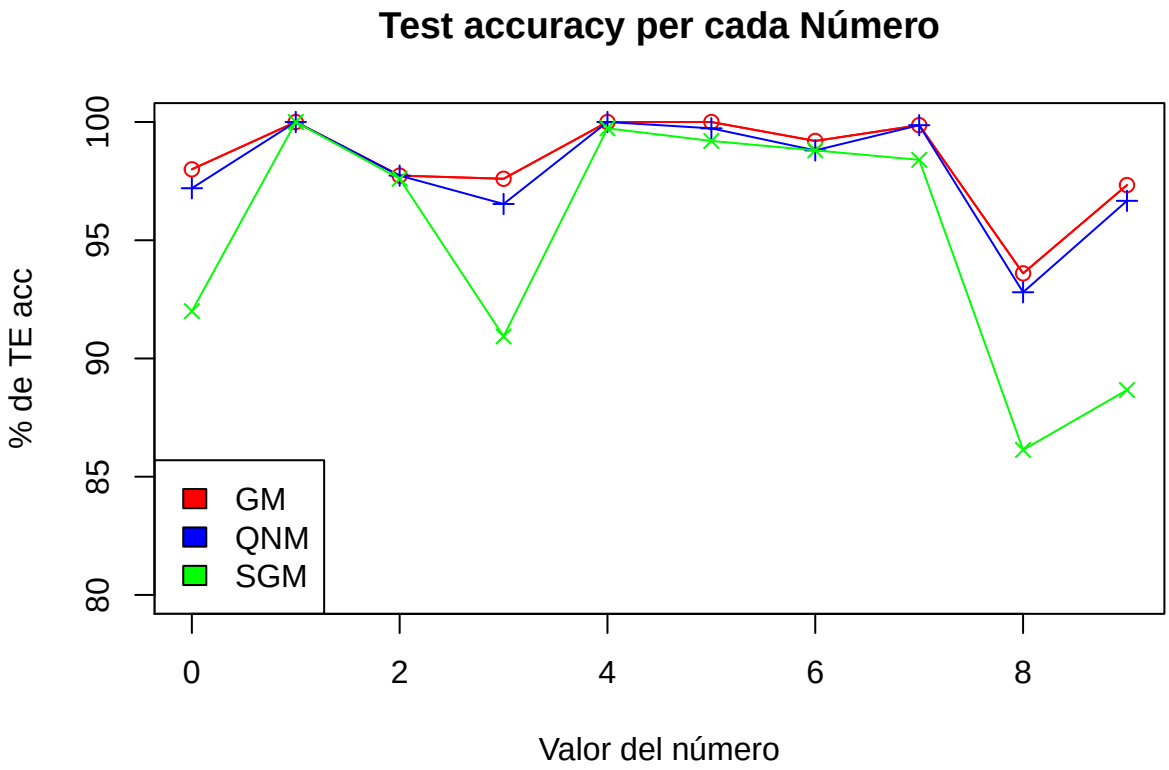
a)

En primer lloc ens centrarem en la presició de cada algoritme per les diferents lambdes.



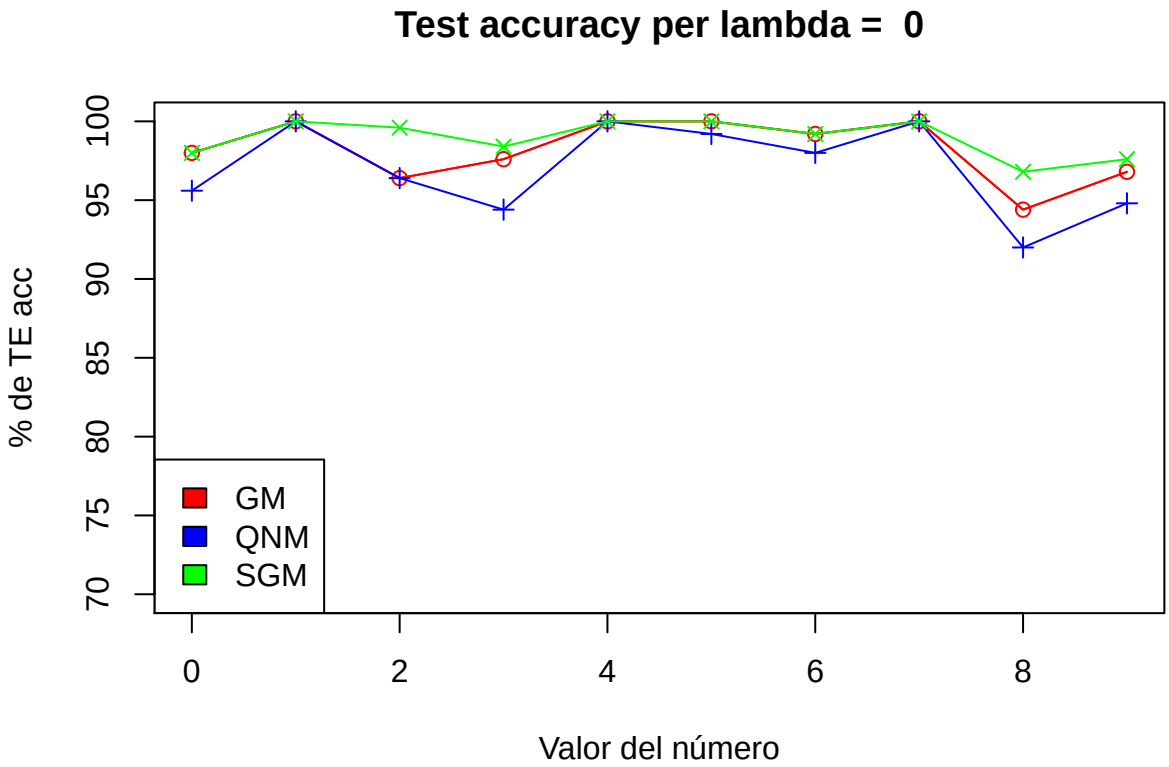
Com es pot observar al gràfic, amb $\lambda = 0$ tots tres algoritmes mostren un bon percentatge d'encert. A mesura que la lambda creix, el GM i el QNM mantenen o inclús milloren la seva precisió, en canvi el SGM empitjora molt notablement.

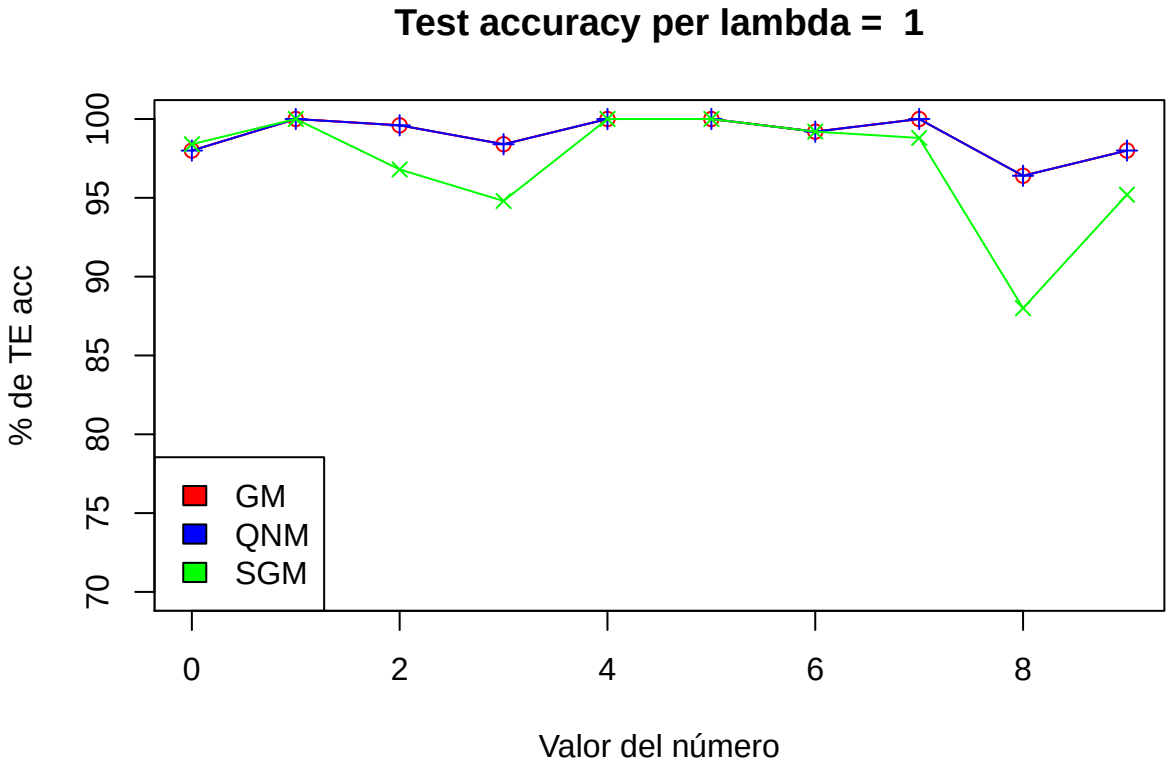
Ara compararem la presició d'encert per cada dígit de manera individual per tots tres mètodes.

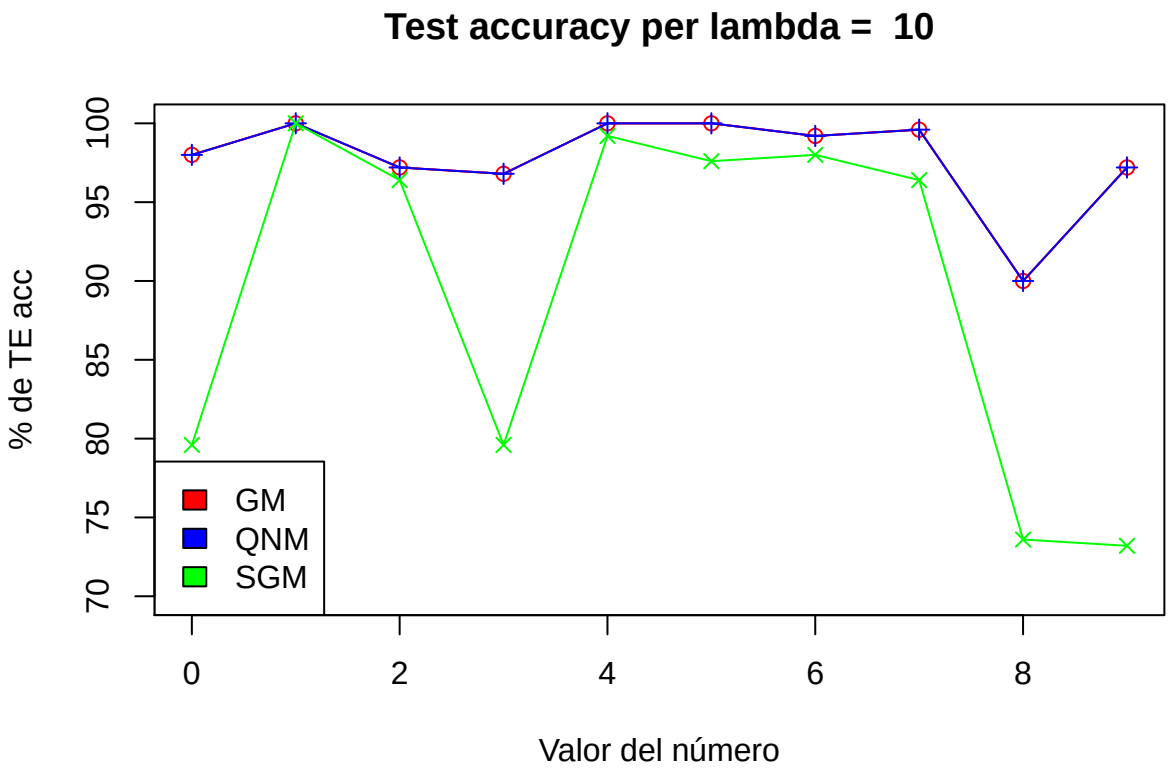


Si observem els resultats podem concloure que els nombres més difícils de reconeixer són el 0, el 3 i el 8 per tots tres mètodes. Això es deu a que tenen formes molt similars i per tant són dificils de diferenciar. També podem veure que en general, sense tenir en compte els diferents valors de λ, el SGM és el mètode que presenta un error de test més alt.

A continuació realitzarem el mateix anàlisi que abans, presició d'encert per cada dígit de manera individual, però diferenciant cada valor de λ per veure com actuen els diferents mètodes en funció d'aquest paràmetre i quin és el seu valor òptim.


Test accuracy per lambda = 0
% de TE acc
100
95
90
85
80
75
70
GM
QNM
SGM
0
2
4
6
8
Valor del número


Test accuracy per lambda = 1
% de TE acc
100
95
90
85
80
75
70
GM
QNM
SGM
0
2
4
6
8
Valor del número

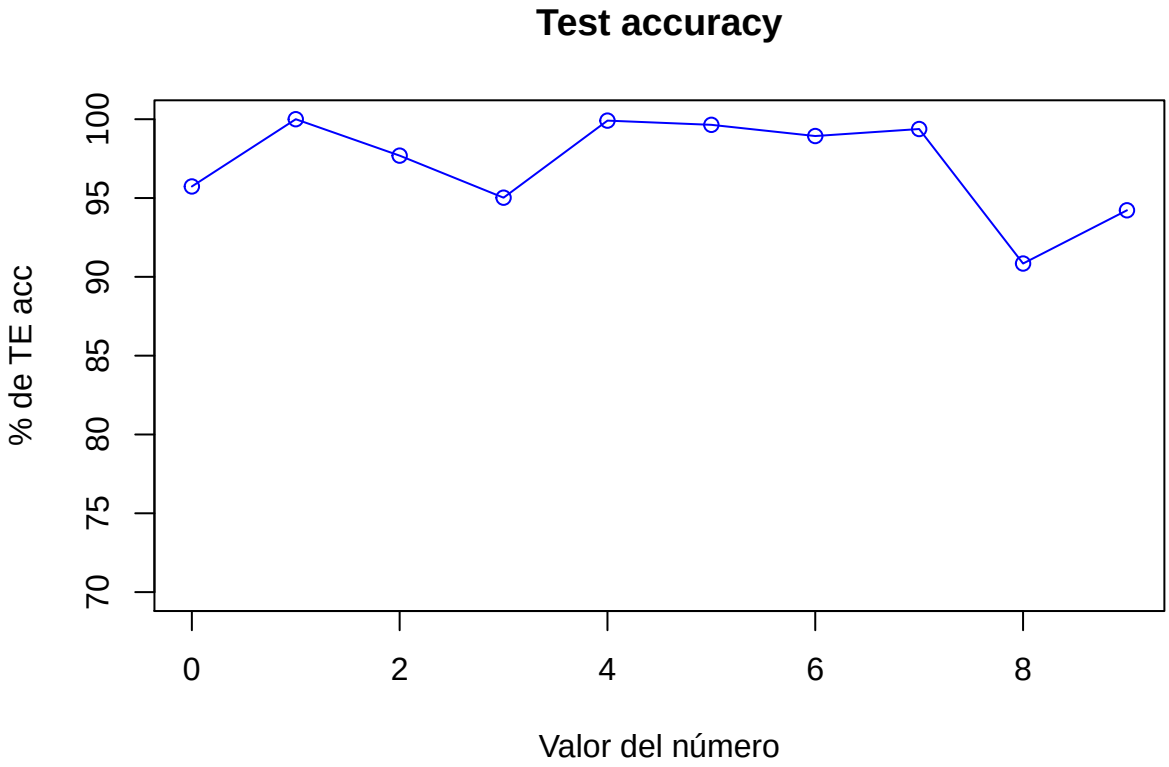


Aquests tres gràfic són molt útils ja que encara que abans podíem pensar que el mètode de SGM cometia molt error, veiem que per $\lambda = 0$ és el que més encert presenta. De totes maneres, a mesura que la λ augmenta, la presició d'encert del SGM disminueix dràsticament, cometent errors molt grans pel cas de $\lambda = 10$. Podem relacionar aquest fet amb el primer gràfic que hem realitzat, on podíem observar com al augmentar el valor de λ, el valor de la funció objectiu que aquest mètode oferia després de les 1000 iteracions quedava molt lluny de l'òptim real. Per contra, els altres dos mètodes mantenen o inclús milloren el seu rendiment en termes generals. Una altra vegada veiem com ens dígits més difícils de diferenciar són el 0, el 3 i el 8.

Un cop hem completat aquest anàlisi, el λ-algoritme que ens ofereix una major presició és el SGM amb $\lambda = 0$, a més a més, aquest mètode té un temps d'execició molt petit tal i com havíem vist anteriorment. Per tant, recomanaríem utilitzar aquesta combinació que ens ofereix resultats molt bons en molt poc temps.

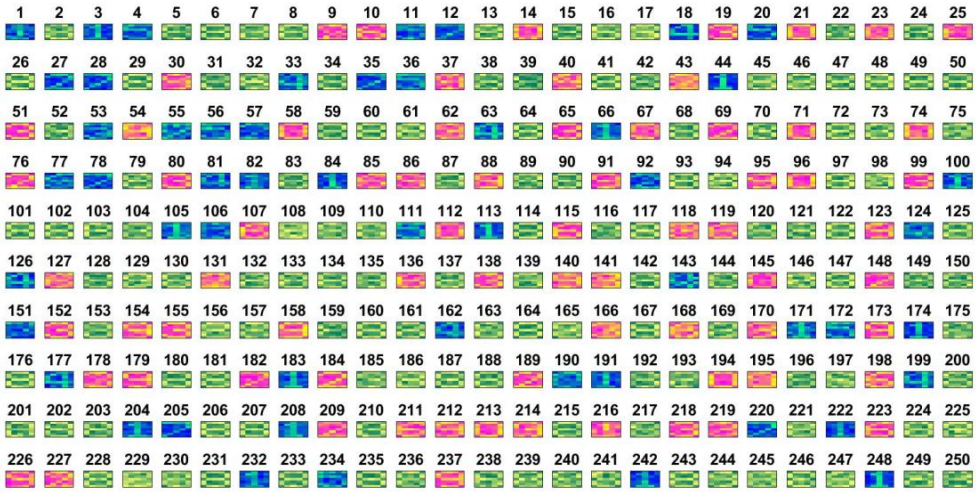
b)

Ara calcularem la mitjana d'encert pel reconeixement de cada dígit independentment de l'algoritme i el valor de la λ.




```
##       means
## 0  95.73333
## 1 100.00000
## 2  97.68889
## 3  95.02222
## 4  99.91111
## 5  99.64444
## 6  98.93333
## 7  99.37778
## 8  90.84444
## 9  94.22222
```

Podem veure com el número 8 és el més difícil de reconeixer independentment de l'algoritme i el valor de λ usats.



A la imatge podem veure els resultats d'intentar reconeixer el número 8 utilitzant l'algoritme SGM amb $\lambda = 10$. En verd, trobem tots aquells 8 correctament identificats, en blau els dígits diferents de 8 també ben classificats i en rosa els falsos positius. Per tant, veiem que el principal problema de detecció de 8 recau en que la xarxa classifica molts dígits com a 8 que realment no ho són. En canvi, no observem cap fals negatiu, l'algorisme no es deixa cap 8 sense detectar.

Això es deu a que la seva forma es pot confondre fàcilment amb molts altres números i tots aquests es converteixen en falsos positius quan posem en marxa la nostra xarxa.