La base de datos consta de dos archivos, train.csv y test.csv, un archivo para entrenar el modelo y otro archivo para testearlo, cada uno con la misma cantidad de ejemplos por cada clase, la base de datos de entrenamiento tiene 1840 ejemplos por cada clase para un total de 9200 ejemplos y la base de datos de test tiene 460 ejemplos por cada clase para un total de 2300 ejemplos.

Los entrenamientos se realizaron variando de forma aleatoria la cantidad de capas ocultas, el número de neuronas por capa, y los demás hiperparametros mencionados en cada entrenamiento.

**Entrenamientos:**

**Entrenamiento 0**

n\_hidden=[2560,1920,1280,960,640,320,160,80,40,20] # 10 capas

learning\_rate = 0.0001

training\_epochs = 1000

batch\_size =50#<=50 & >= 10

display\_step = 10 #cada cuantas iteraciónes imprime

n\_input = 178

dropout\_rate = 0.2

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

train\_op = optimizer.minimize(loss\_op)

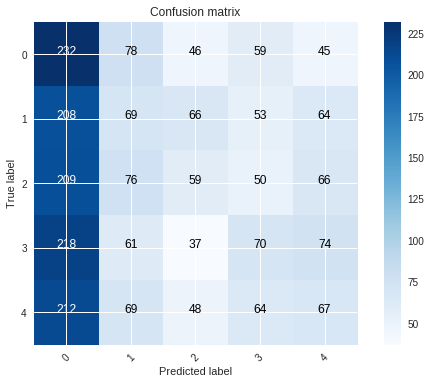
Optimization Finished!

Cost Min: cost=1.513386794 in epoch: 0993 cost\_init=2.172012383 accurrency\_init=0.187826082

Accuracy-train: 0.23467392

Accuracy-test: 0.22130434

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.5396739130434782, 1: 0.8456521739130435, 2: 0.8929347826086956, 3: 0.8771739130434782, 4: 0.8646739130434783}

TPR Sensibilidad o verdaderos positivos {0: 0.5043478260869565, 1: 0.15, 2: 0.1282608695652174, 3: 0.15217391304347827, 4: 0.14565217391304347}

**Entrenamiento 1**

n\_hidden=[2560,1920,1280,960,640,320,160,80,40,20] # 10 capas

learning\_rate = 0.0001

training\_epochs = 1000

batch\_size =50#<=50 & >= 10

display\_step = 10 #cada cuantas iteraciónes imprime

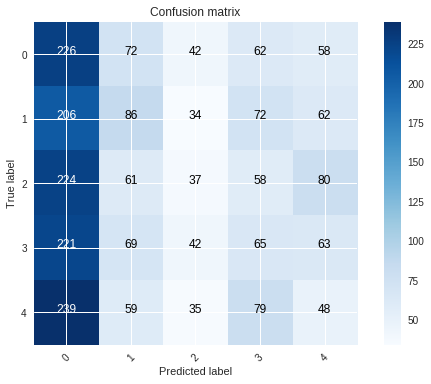
n\_input = 178

dropout\_rate = 0.2

train\_op = optimizer.minimize(loss\_op)

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.510820984 in epoch: 0879 cost\_init=2.640485015 accurrency\_init=0.200000003

Accuracy-train: 0.23445652

Accuracy-test: 0.20434782

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.5163043478260869, 1: 0.8581521739130434, 2: 0.9168478260869565, 3: 0.8527173913043479, 4: 0.8570652173913044}

TPR Sensibilidad o verdaderos positivos

{0: 0.49130434782608695,

1: 0.18695652173913044,

2: 0.08043478260869565,

3: 0.14130434782608695,

4: 0.10434782608695652}

**Entrenamiento 2**

n\_hidden=[2560,1920,1280,960,640,320,160,80,40,20] # 10 capas

learning\_rate = 0.0001

training\_epochs = 1000

batch\_size =50#<=50 & >= 10

display\_step = 10 #cada cuantas iteraciónes imprime

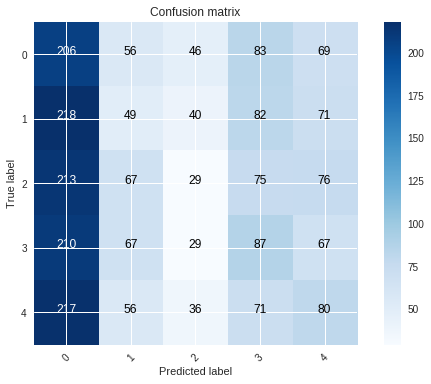
n\_input = 178

dropout\_rate = 0.2

train\_op = optimizer.minimize(loss\_op)

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

Cost Min: cost=0.711314980 in epoch: 0998 cost\_init=1.014492849 accurrency\_init=0.195217386

Accuracy-train: 0.22836956

Accuracy-test: 0.20043479

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.533695652173913, 1: 0.866304347826087, 2: 0.9179347826086957, 3: 0.8309782608695652, 4: 0.846195652173913}

TPR Sensibilidad o verdaderos positivos {0: 0.44782608695652176, 1: 0.10652173913043478, 2: 0.06304347826086956, 3: 0.1891304347826087, 4: 0.17391304347826086}

**Entrenamiento 3**

n\_hidden=[2560,1920,1280,960,640,320,160,80,40,20] # 10 capas

learning\_rate = 0.0001

training\_epochs = 1000

batch\_size =50#<=50 & >= 10

display\_step = 10 #cada cuantas iteraciónes imprime

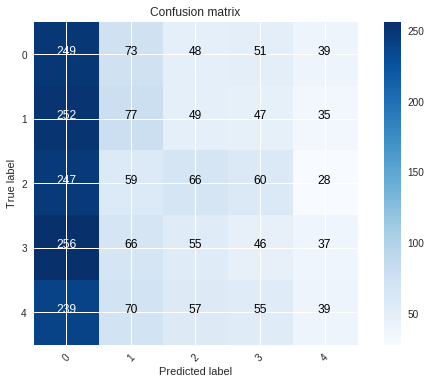
n\_input = 178

dropout\_rate = 0.2

train\_op = optimizer.minimize(loss\_op)

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=0.691820696 in epoch: 0985 cost\_init=0.986866643 accurrency\_init=0.186086953

Accuracy-train: 0.18315217

Accuracy-test: 0.19217391

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.4597826086956522, 1: 0.8543478260869565, 2: 0.8864130434782609, 3: 0.8842391304347826, 4: 0.9244565217391304}

TPR Sensibilidad o verdaderos positivos {0: 0.5413043478260869, 1: 0.1673913043478261, 2: 0.14347826086956522, 3: 0.1, 4: 0.08478260869565217}

**Entrenamiento 4**

learning\_rate = 0.0001 #

training\_epochs = 1000 #

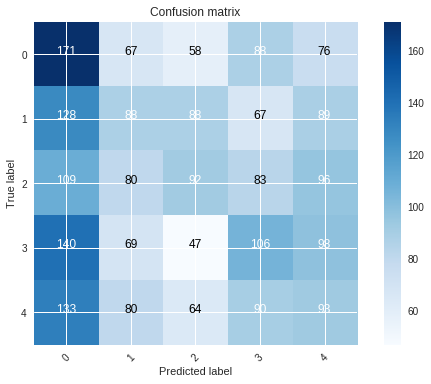
n\_hidden=[2560,5120,10240,5120,2560,1280,640,320,160,80]

dropout\_rate = 0.3

batch\_size =50

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)# optimizador

Cost Min: cost=1.513967712 in epoch: 0994 cost\_init=2.052056408 accurrency\_init=0.200434789

Accuracy-train: 0.27380434

Accuracy-test: 0.24347825

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7228260869565217, 1: 0.8391304347826087, 2: 0.8603260869565217, 3: 0.8217391304347826, 4: 0.8048913043478261}

TPR Sensibilidad o verdaderos positivos {0: 0.3717391304347826, 1: 0.19130434782608696, 2: 0.2, 3: 0.23043478260869565, 4: 0.20217391304347826}

**Entrenamiento 5**

n\_hidden=[600,500,400,300,350,450,550,650,550,700,600,500,400,300,150,80]

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

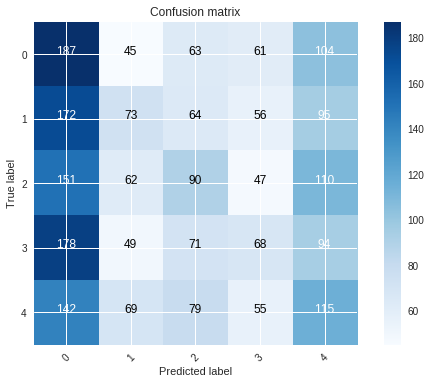
display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

n\_input = 178

dropout\_rate = 0.3

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.433175411 in epoch: 1000 cost\_init=1.723486026 accurrency\_init=0.210869566

Accuracy-train: 0.27934784

Accuracy-test: 0.23391305

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.6505434782608696, 1: 0.8777173913043478, 2: 0.8494565217391304, 3: 0.8809782608695652, 4: 0.7809782608695652}

TPR Sensibilidad o verdaderos positivos {0: 0.40652173913043477, 1: 0.15869565217391304, 2: 0.1956521739130435, 3: 0.14782608695652175, 4: 0.25}

**Entrenamiento 6**

n\_hidden=[600,500,400,300,350,450,550,650,550,700,600,500,400,300,150,80]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

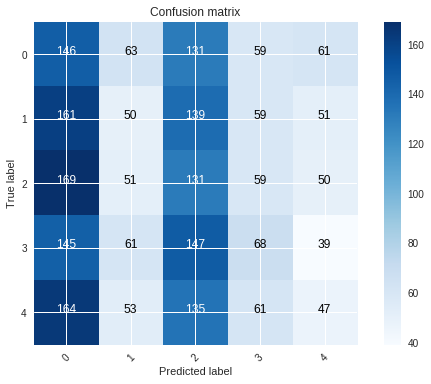
n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.3

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.912183874 in epoch: 0874 cost\_init=2.142123218 accurrency\_init=0.197826087

Accuracy-train: 0.21021739

Accuracy-test: 0.20956522

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.6527173913043478, 1: 0.8760869565217392, 2: 0.7, 3: 0.8706521739130435, 4: 0.8907608695652174}

TPR Sensibilidad o verdaderos positivos {0: 0.3173913043478261, 1: 0.10869565217391304, 2: 0.2847826086956522, 3: 0.14782608695652175, 4: 0.10217391304347827}

**Entrenamiento 7**

n\_hidden=[600,500,400,300,350,450,550,650,550,700,600,500,400,300,150,80]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

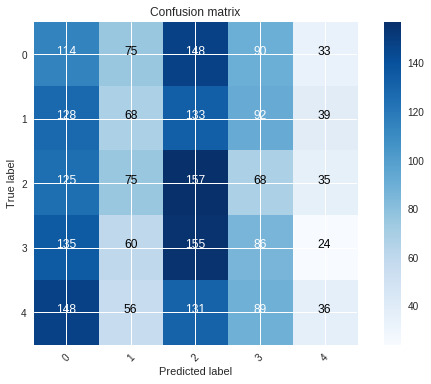
n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.3

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=0.789823175 in epoch: 0997 cost\_init=0.934686455 accurrency\_init=0.205217391

Accuracy-train: 0.21728261

Accuracy-test: 0.2073913

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7086956521739131, 1: 0.8554347826086957, 2: 0.6918478260869565, 3: 0.8157608695652174, 4: 0.928804347826087}

TPR Sensibilidad o verdaderos positivos {0: 0.24782608695652175, 1: 0.14782608695652175, 2: 0.34130434782608693, 3: 0.18695652173913044, 4: 0.0782608695652174}

**Entrenamiento 8**

n\_hidden=[600,500,400,300,350,450,550,650,550,700,600,500,400,300,150,80]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.3

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

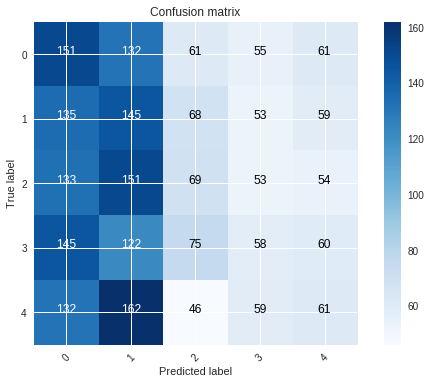
optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=0.693147235 in epoch: 0121 cost\_init=0.962858280 accurrency\_init=0.193913043

Accuracy-train: 0.2025

Accuracy-test: 0.20043479

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7038043478260869, 1: 0.6918478260869565, 2: 0.8641304347826086, 3: 0.8804347826086957, 4: 0.8728260869565218}

TPR Sensibilidad o verdaderos positivos {0: 0.3282608695652174, 1: 0.31521739130434784, 2: 0.15, 3: 0.12608695652173912, 4: 0.13260869565217392}

**Entrenamiento 9**

n\_hidden=[600,500,400,300]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.001 #

training\_epochs = 500 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

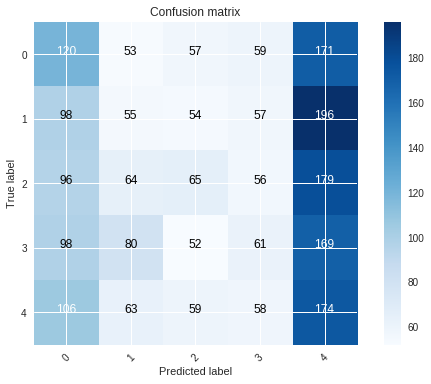
n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.4

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.741002681 in epoch: 0489 cost\_init=1.973916676 accurrency\_init=0.206521735

Accuracy-train: 0.22043478

Accuracy-test: 0.2173913

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.783695652173913, 1: 0.8586956521739131, 2: 0.8793478260869565, 3: 0.875, 4: 0.6114130434782609}

TPR Sensibilidad o verdaderos positivos {0: 0.2608695652173913, 1: 0.11956521739130435, 2: 0.14130434782608695, 3: 0.13260869565217392, 4: 0.3782608695652174}

**Entrenamiento 10**

n\_hidden=[600,500,400,300]

learning\_rate = 0.001 #

training\_epochs = 500 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

n\_input = 178

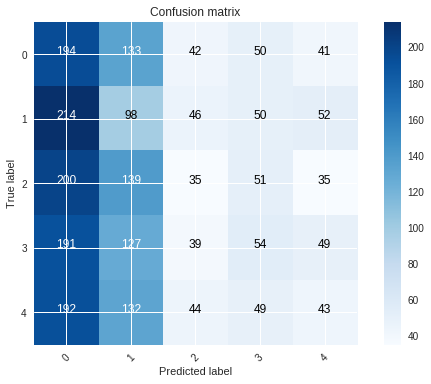
n\_classes = 5 # numero de clases

dropout\_rate = 0.4

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y)

optimizer =tf.train.GradientDescentOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=0.909044848 in epoch: 0490 cost\_init=1.127364654 accurrency\_init=0.203478262

Accuracy-train: 0.20913044

Accuracy-test: 0.20608696

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.5668478260869565, 1: 0.7114130434782608, 2: 0.9070652173913043, 3: 0.8913043478260869, 4: 0.903804347826087}

TPR Sensibilidad o verdaderos positivos {0: 0.4217391304347826, 1: 0.21304347826086956, 2: 0.07608695652173914, 3: 0.11739130434782609, 4: 0.09347826086956522}

**Entrenamiento 11**

n\_hidden=[600,500,400,300]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.001 #

training\_epochs = 500 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

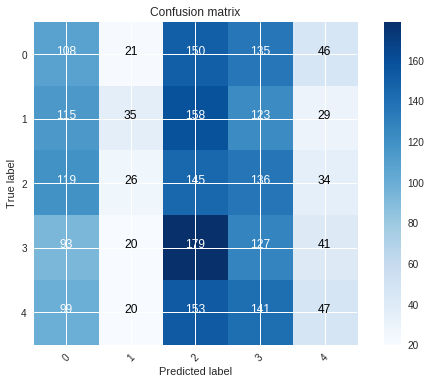
n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.4

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=0.693015031 in epoch: 0113 cost\_init=0.742471015 accurrency\_init=0.196521744

Accuracy-train: 0.2

Accuracy-test: 0.2

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7684782608695652, 1: 0.9527173913043478, 2: 0.6521739130434783, 3: 0.7092391304347826, 4: 0.9184782608695652}

TPR Sensibilidad o verdaderos positivos {0: 0.23478260869565218, 1: 0.07608695652173914, 2: 0.31521739130434784, 3: 0.27608695652173915, 4: 0.10217391304347827}

**Entrenamiento 12**

n\_hidden=[600,500,400,300]

learning\_rate = 0.001 #

training\_epochs = 500 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

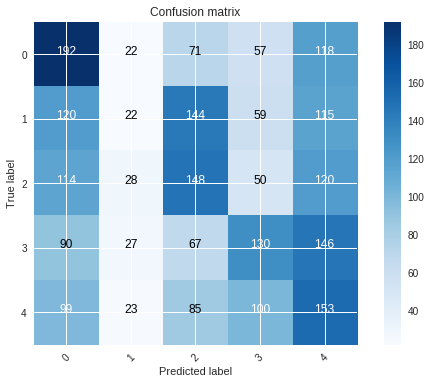
n\_input = 178

n\_classes = 5 # numero de clases

dropout\_rate = 0.4

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.285008143 in epoch: 0496 cost\_init=1.642515669 accurrency\_init=0.200000003

Accuracy-train: 0.35293478

Accuracy-test: 0.2969565

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7701086956521739, 1: 0.9456521739130435, 2: 0.8005434782608696, 3: 0.8554347826086957, 4: 0.7288043478260869}

TPR Sensibilidad o verdaderos positivos {0: 0.41739130434782606, 1: 0.04782608695652174, 2: 0.3217391304347826, 3: 0.2826086956521739, 4: 0.33260869565217394}

**Entrenamiento 13**

n\_hidden=[600,500,400,300]

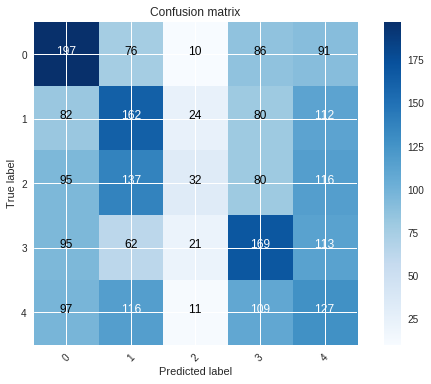
total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.001 #

training\_epochs = 1000 #

batch\_size =50# <=10 & <= 50 # numero de ejemplos que van a ser mostrados a la red en cada itereación

display\_step = 10 #cada cuantas iteraciónes imprime los resultados de la función de costo

n\_input = 178

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

ost Min: cost=1.252722198 in epoch: 0943 cost\_init=1.636424004 accurrency\_init=0.190434784

Accuracy-train: 0.39065218

Accuracy-test: 0.30173913

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7994565217391304, 1: 0.7875, 2: 0.9641304347826087, 3: 0.8070652173913043, 4: 0.7652173913043478}

TPR Sensibilidad o verdaderos positivos {0: 0.4282608695652174, 1: 0.3521739130434783, 2: 0.06956521739130435, 3: 0.3673913043478261, 4: 0.27608695652173915}

**Entrenamiento 14**

n\_hidden=[600,500,400,300]

learning\_rate = 0.001 #

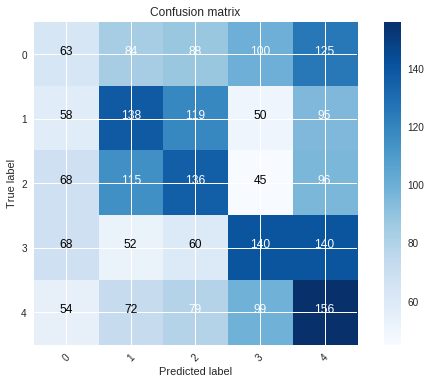
training\_epochs = 2000 #

batch\_size =50

dropout\_rate = 0.4

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)



Cost Min: cost=1.226615241 in epoch: 1957 cost\_init=1.628229029 accurrency\_init=0.207826093

Accuracy-train: 0.42826086

Accuracy-test: 0.34130436

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.8652173913043478, 1: 0.8244565217391304, 2: 0.8119565217391305, 3: 0.8402173913043478, 4: 0.7521739130434782}

TPR Sensibilidad o verdaderos positivos {0: 0.13695652173913042, 1: 0.3, 2: 0.2956521739130435, 3: 0.30434782608695654, 4: 0.3391304347826087}

**Entrenamiento 15**

n\_hidden=[600,500,400,300]

learning\_rate = 0.0001 #

training\_epochs = 3000 #

batch\_size =30# <=1

n\_classes = 5 # numero de clases

dropout\_rate = 0.5

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

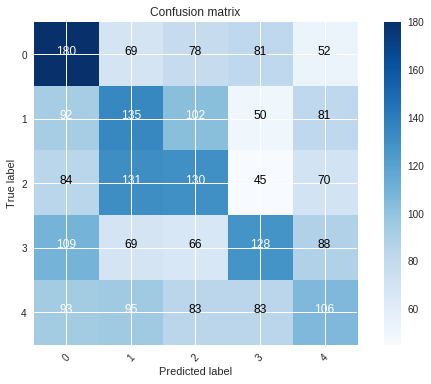
optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.006911298 in epoch: 2947 cost\_init=1.922930865 accurrency\_init=0.211304352

Accuracy-train: 0.52347827

Accuracy-test: 0.3047826

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7945652173913044, 1: 0.8021739130434783, 2: 0.821195652173913, 3: 0.8592391304347826, 4: 0.8418478260869565}

TPR Sensibilidad o verdaderos positivos {0: 0.391304347826087, 1: 0.29347826086956524, 2: 0.2826086956521739, 3: 0.2782608695652174, 4: 0.23043478260869565}

**Entrenamiento 16**

dropout\_rate = 0.5

learning\_rate = 0.0001 #

training\_epochs = 3000 #

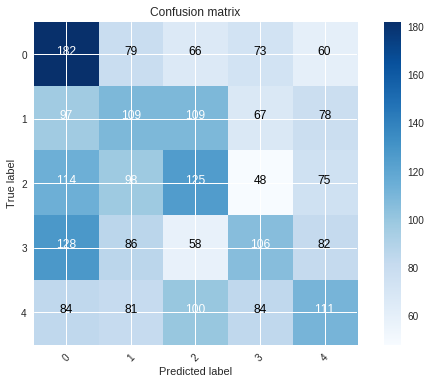
batch\_size =50

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.010744223 in epoch: 2894 cost\_init=1.924905914 accurrency\_init=0.207391307

Accuracy-train: 0.5202174

Accuracy-test: 0.27130434

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7701086956521739, 1: 0.8130434782608695, 2: 0.8190217391304347, 3: 0.8521739130434782, 4: 0.8396739130434783}

TPR Sensibilidad o verdaderos positivos {0: 0.39565217391304347, 1: 0.23695652173913043, 2: 0.2717391304347826, 3: 0.23043478260869565, 4: 0.24130434782608695}

**Entrenamiento 17**

n\_hidden=[300,400,500,400,300,400,500,400,300]

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50

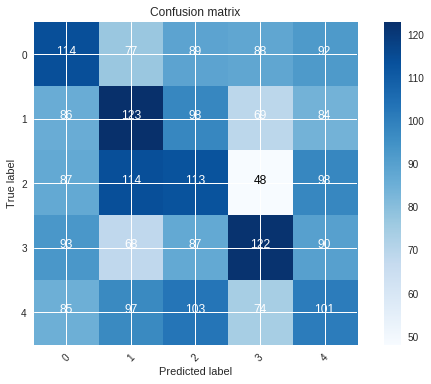
loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.101244644 in epoch: 0996 cost\_init=1.960285008 accurrency\_init=0.204782605

Accuracy-train: 0.48934782

Accuracy-test: 0.24608696

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.8092391304347826, 1: 0.8065217391304348, 2: 0.7951086956521739, 3: 0.8483695652173913, 4: 0.8021739130434783}

TPR Sensibilidad o verdaderos positivos {0: 0.24782608695652175, 1: 0.2673913043478261, 2: 0.24565217391304348, 3: 0.26521739130434785, 4: 0.21956521739130436}

**Entrenamiento 18**

n\_hidden=[300,400,500,400,300,400,500,400,300]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

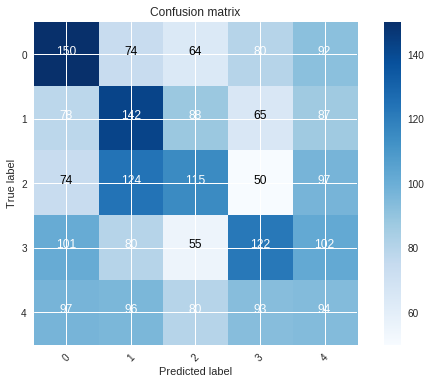
learning\_rate = 0.0001 #

training\_epochs = 3000 #

batch\_size =40

loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.029384277 in epoch: 2211 cost\_init=1.662210825 accurrency\_init=0.202173918

Accuracy-train: 0.5679348

Accuracy-test: 0.2721739

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.8097826086956522, 1: 0.7967391304347826, 2: 0.8440217391304348, 3: 0.8434782608695652, 4: 0.7945652173913044}

TPR Sensibilidad o verdaderos positivos {0: 0.32608695652173914, 1: 0.30869565217391304, 2: 0.25, 3: 0.26521739130434785, 4: 0.20434782608695654}

**Entrenamiento 19**

learning\_rate = 0.0001 #

training\_epochs = 1000 #

batch\_size =50

n\_hidden=[300,350,400,450,400,350,300,250,200]

dropout\_rate = 0.5

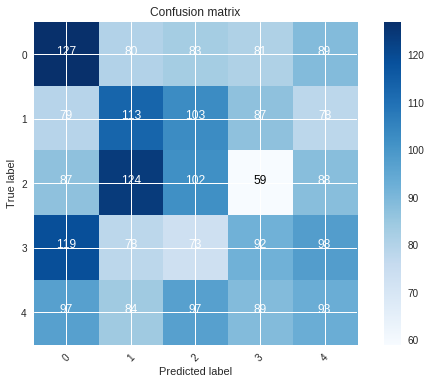
loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.069706586 in epoch: 0877 cost\_init=1.831158485 accurrency\_init=0.200869560

Accuracy-train: 0.4954348

Accuracy-test: 0.23043478

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.7923913043478261, 1: 0.8010869565217391, 2: 0.8065217391304348, 3: 0.8282608695652174, 4: 0.8081521739130435}

TPR Sensibilidad o verdaderos positivos {0: 0.27608695652173915, 1: 0.24565217391304348, 2: 0.2217391304347826, 3: 0.2, 4: 0.20217391304347826}

**Entrenamiento 20**

n\_hidden=[350,300,2500,200]

total\_examples = len(train\_x\_orig) # numero de ejemplos de la BD

learning\_rate = 0.0001 #

training\_epochs = 3000 #

batch\_size =30

dropout\_rate = 0.5

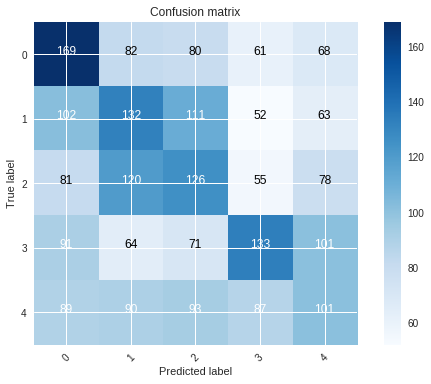
loss\_op = tf.reduce\_mean(tf.losses.softmax\_cross\_entropy(logits=logits, onehot\_labels=Y))#

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Cost Min: cost=1.000508346 in epoch: 2346 cost\_init=1.912574027 accurrency\_init=0.190434784

Accuracy-train: 0.51152176

Accuracy-test: 0.29347825

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.8027173913043478, 1: 0.8065217391304348, 2: 0.8070652173913043, 3: 0.8614130434782609, 4: 0.8315217391304348}

TPR Sensibilidad o verdaderos positivos {0: 0.3673913043478261, 1: 0.28695652173913044, 2: 0.27391304347826084, 3: 0.2891304347826087, 4: 0.21956521739130436}

**Entrenamiento 21**

n\_hidden=[356,260,178,130,89,45,20]

learning\_rate = 0.0001

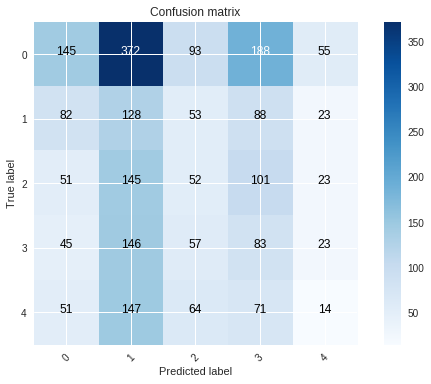
training\_epochs = 300

batch\_size =50

loss\_op = tf.reduce\_mean(tf.losses.sigmoid\_cross\_entropy(logits=logits, multi\_class\_labels=Y))

optimizer = tf.train.AdamOptimizer(learning\_rate=learning\_rate)

Accuracy-train: 0.35630435

Accuracy-test: 0.37086958

<class 'numpy.ndarray'> <class 'numpy.ndarray'>

TNR Especificidad o verdadesros negativos {0: 0.8417415342087077, 1: 0.5794392523364486, 2: 0.8615145228215768, 3: 0.7697841726618705, 4: 0.9365079365079365}

TPR Sensibilidad o verdaderos positivos {0: 0.16998827667057445, 1: 0.3422459893048128, 2: 0.13978494623655913, 3: 0.2344632768361582, 4: 0.040345821325648415}