Tarea 1: Perceptrón Multicapa

Fecha de entrega: 28 de Agosto, 23:59

Profesor: Pablo Estévez V.
Auxiliar: Ignacio Reyes J.
Semestre: Primavera 2019

Tarea 1: Perceptrón Multicapa

1. Segunda parte: experimentos

1. ¿Hay alguna diferencia apreciable entre ambos casos? ¿Qué funcional es más adecuado para entrenar esta red? Justifique su respuesta en base a los resultados obtenidos y a los fundamentos teóricos.

   Sol: Al comparar las dos funciones de costo, queda claro a partir de las curvas de aprendizaje que Cross Entropy convergerá con menos iteraciones que MSE. Pero las tasas de acierto, curvas de Roc, DET son más o menos las mismas.

   Pronto podemos concluir que lo mejor es usar Entropía cruzada. El fundamento teórico detrás se basa en que la Entropía cruzada es óptima para entrenar clasificadores, como es el caso, mientras que el Error cuadrático medio es óptimo para entrenar regresión, esto se debe a que se tiene un clasificador de dos elementos (Verdadero/Falso) por lo que tiene distribución binomial y la Entropía cruzada tiene la forma de la función de verosimilitud para este tipo de funciones de distribución, por lo que esta función de costos permite encontrar el estimador de máxima verosimilitud.

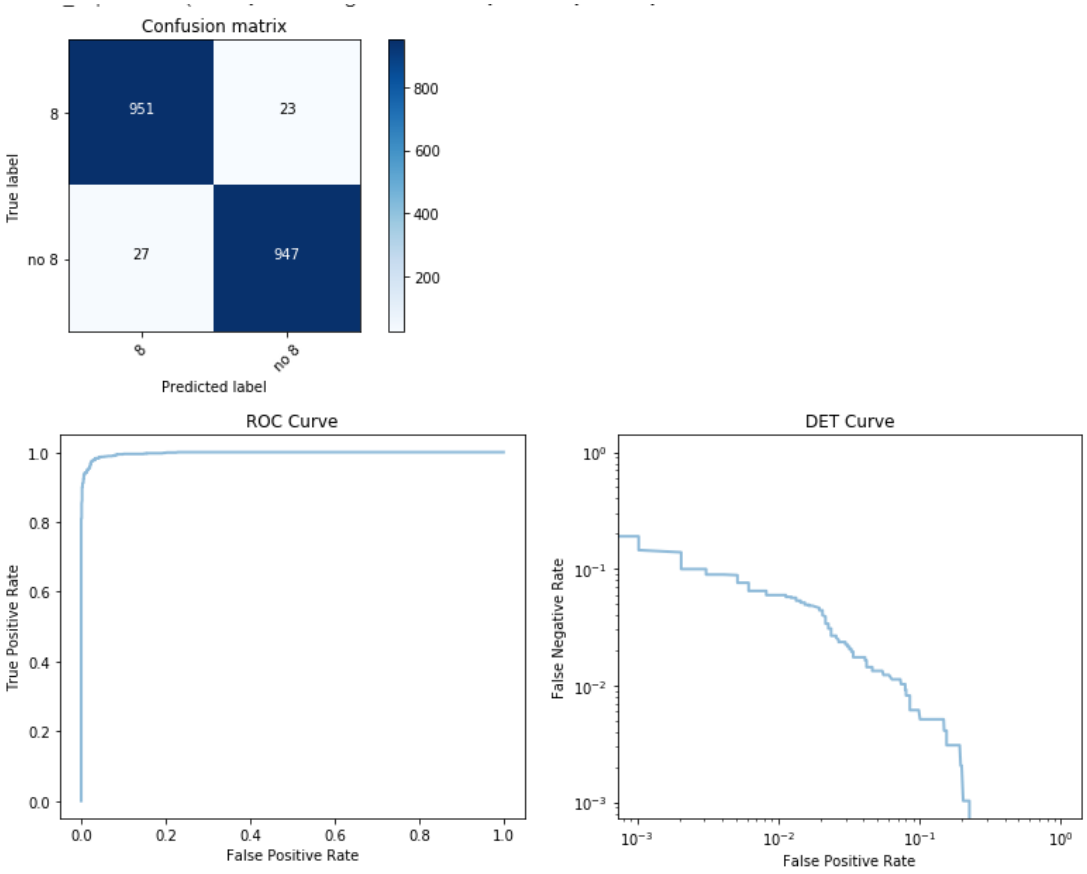

Figura 1: matriz de confusión, curvas ROC y DET para Entropía cruzada

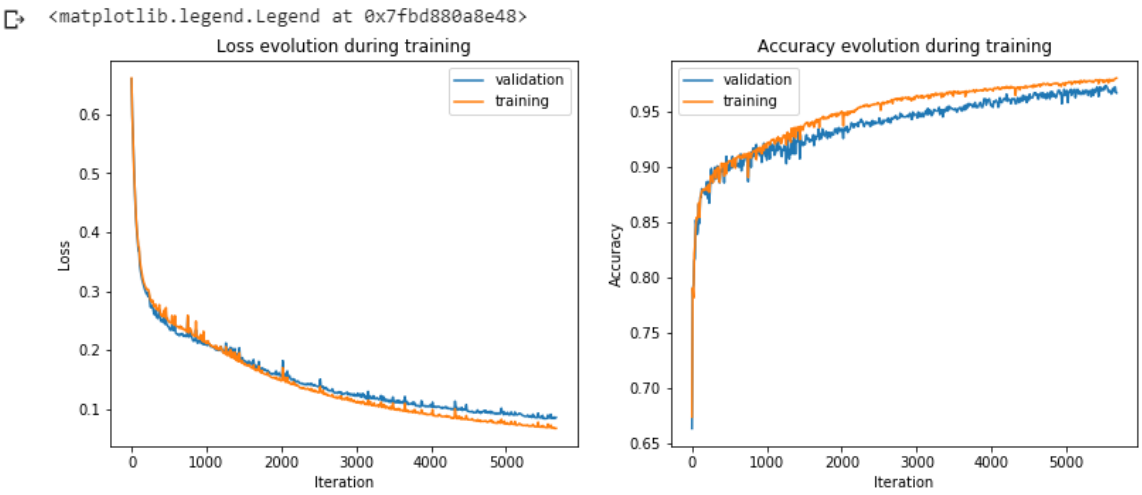

Figura 2: Curvas de aprendizaje para Entropía Cruzada

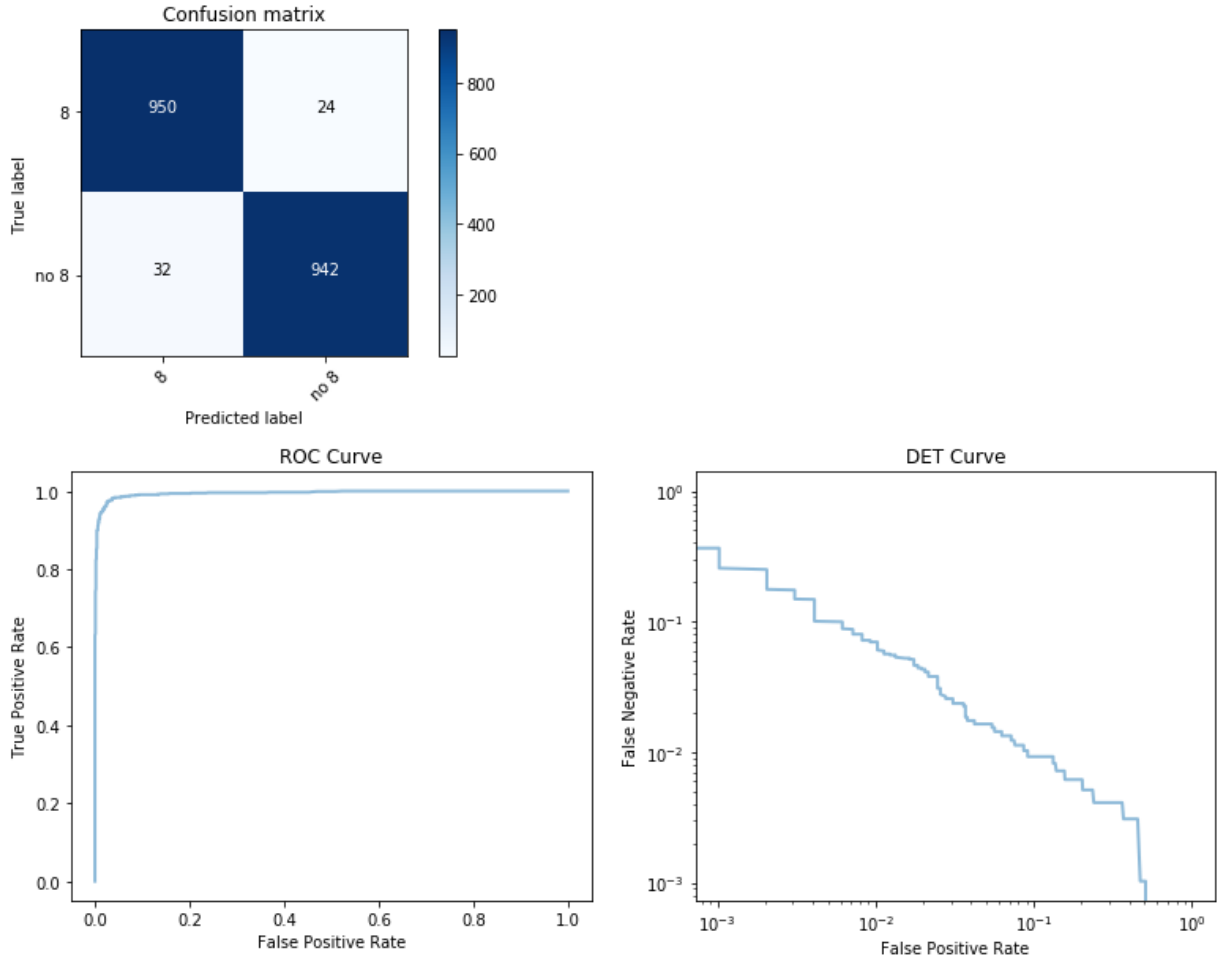

Figura 3: matriz de confusión, curvas ROC y DET para MSE

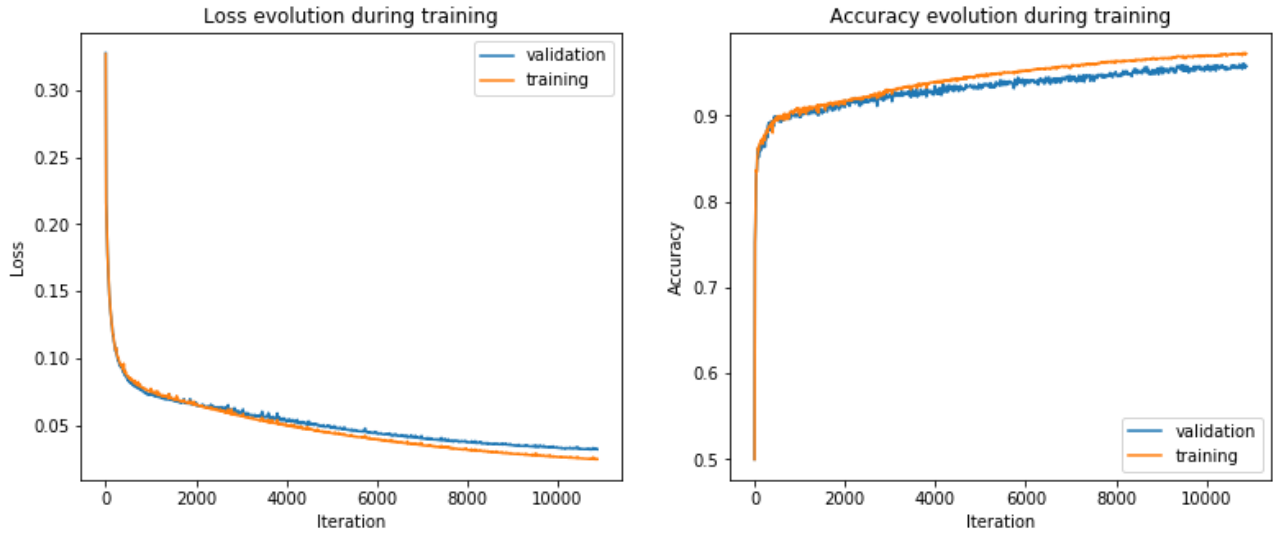

Figura 4: Curvas de aprendizaje para MSE

2. ¿Cómo cambia la tasa de error alcanzada al final del entrenamiento? ¿Qué ocurre con la estabilidad del entrenamiento para tasas altas? ¿Cuántas iteraciones se requieren para la convergencia del modelo en cada caso?. Justifique su respuesta apoyándose en las curvas de aprendizaje, tasas de acierto y otras métricas que considere pertinentes.

   Sol: De una tasa de aprendizaje = 0.01 a 0.1, la tasa de error al final del entrenamiento disminuye. De 0.1 a 1 no se observa mucha variación.

   Para la tasa de aprendizaje = 10 no converge a ningún valor al final de la capacitación.

   Para la tasa de aprendizaje = 0.01, converje aproximadamente 16000 iteraciones

   Para la tasa de aprendizaje = 0.1, converje alrededor de 5000 iteraciones

   Para tasa de aprendizaje = 1, converje exactamente a 1000 iteraciones

   Cuanto mayor es la tasa de aprendizaje, más inestabilidad se observa en el gráfico de la curva de aprendizaje. Cuanto más pequeña, más suave es la curva.

   resultados para $\mu = 0{,}01$:

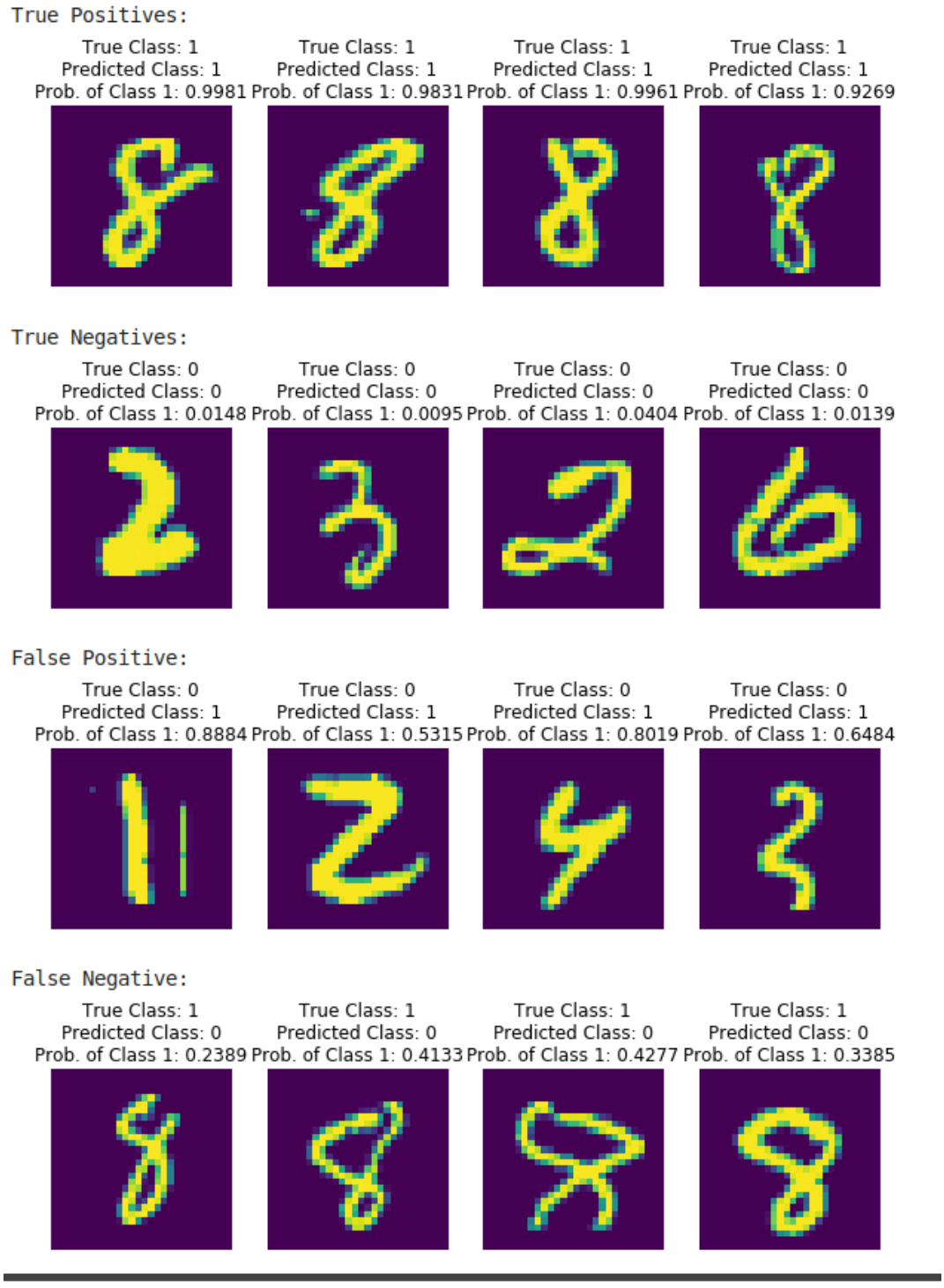

Figura 5: Curvas de aprendizaje para taxa de aprendizaje = 0.01

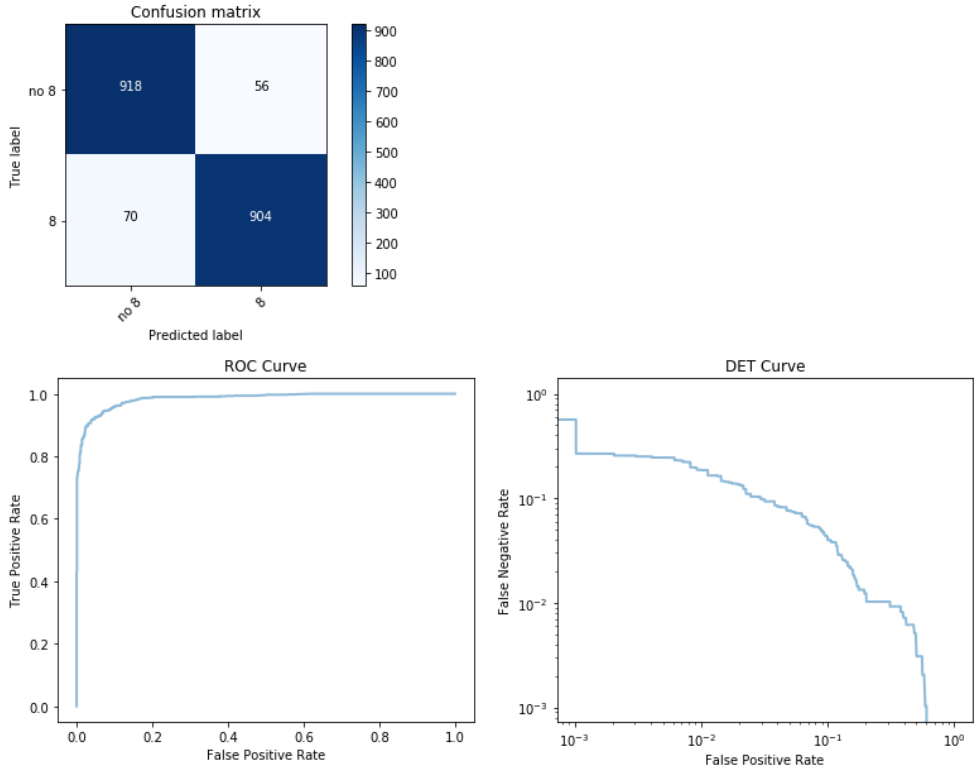

Figura 6: matriz de confusión, curvas ROC y DET para taxa de aprendizaje = 0.01

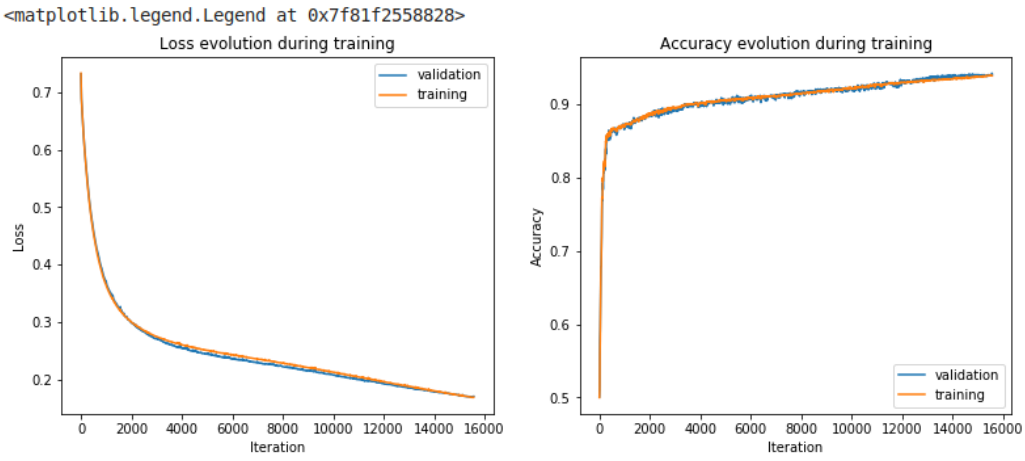

Figura 7: Curvas de aprendizaje para taxa de aprendizaje = 0.01

resultados para $\mu = 0{,}1$:

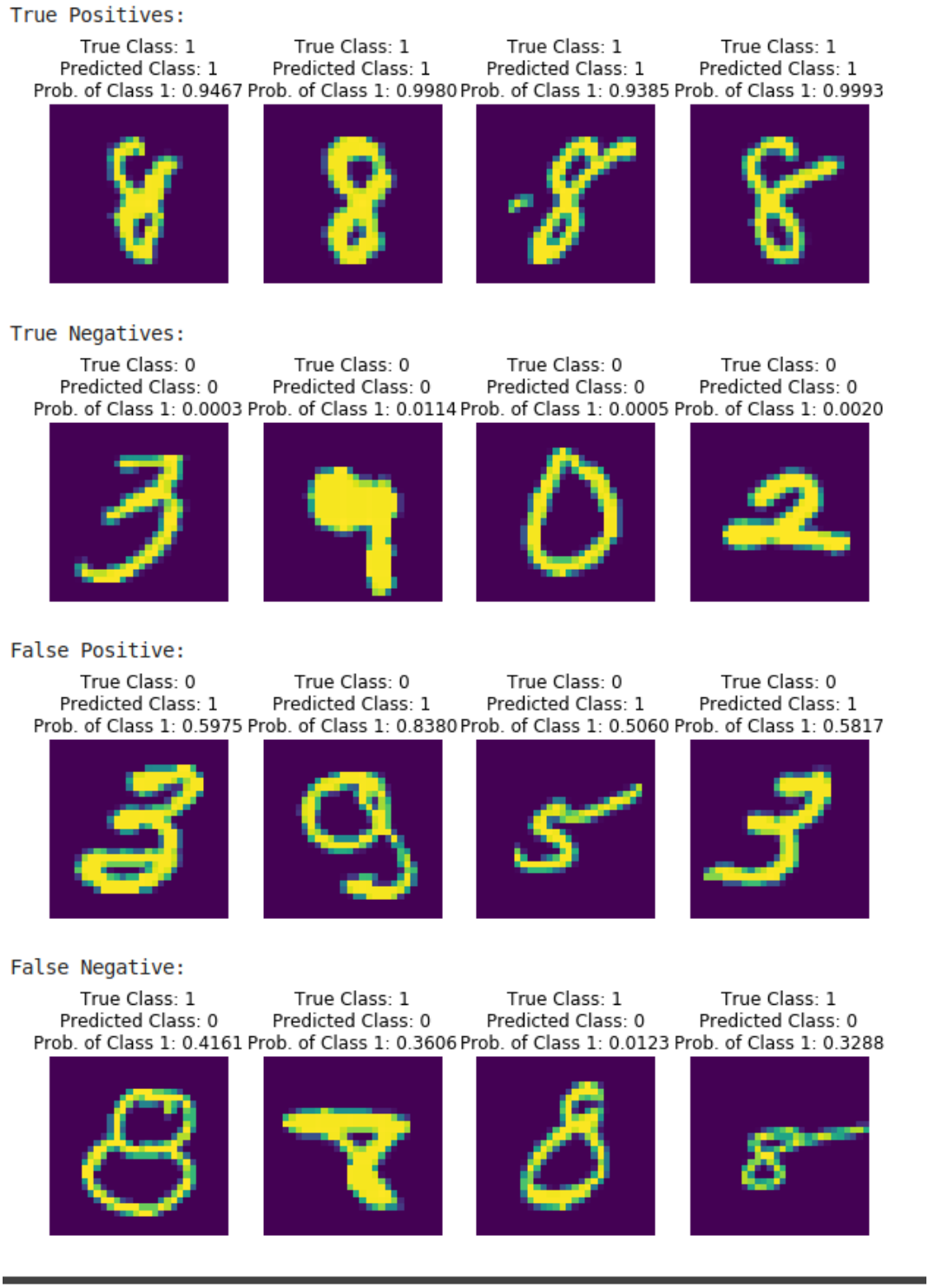

Figura 8: Curvas de aprendizaje para taxa de aprendizaje = 0.1

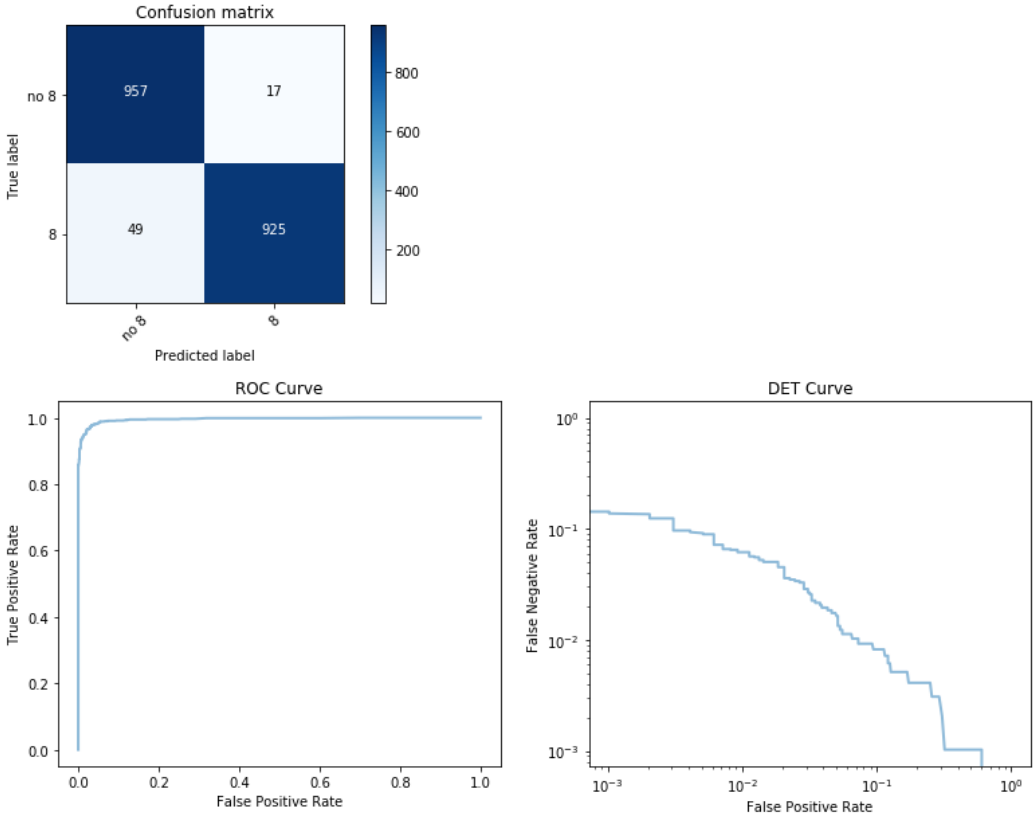

Figura 9: matriz de confusión, curvas ROC y DET para taxa de aprendizaje = 0.1

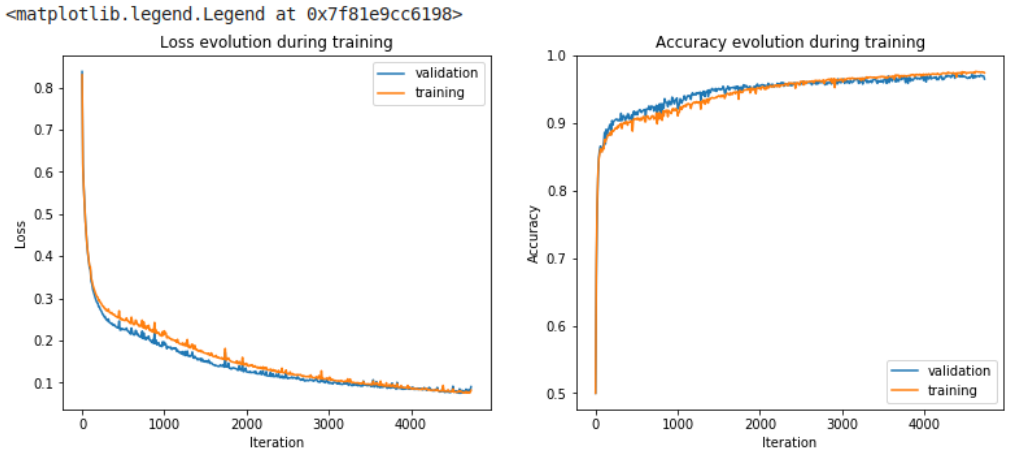

Figura 10: Curvas de aprendizaje para taxa de aprendizaje = 0.1

resultados para $\mu = 1$:

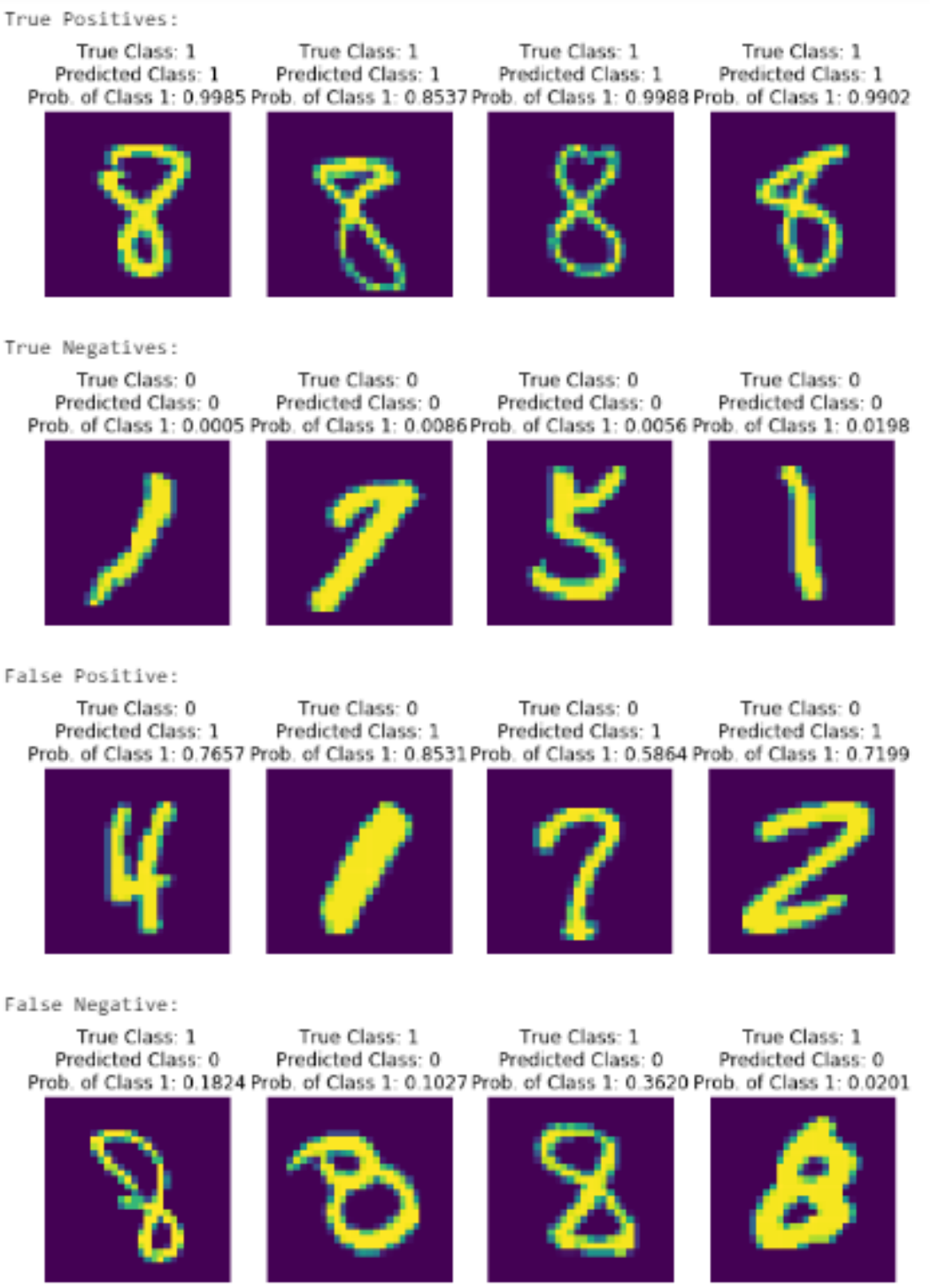

Figura 11: Curvas de aprendizaje para taxa de aprendizaje = 1

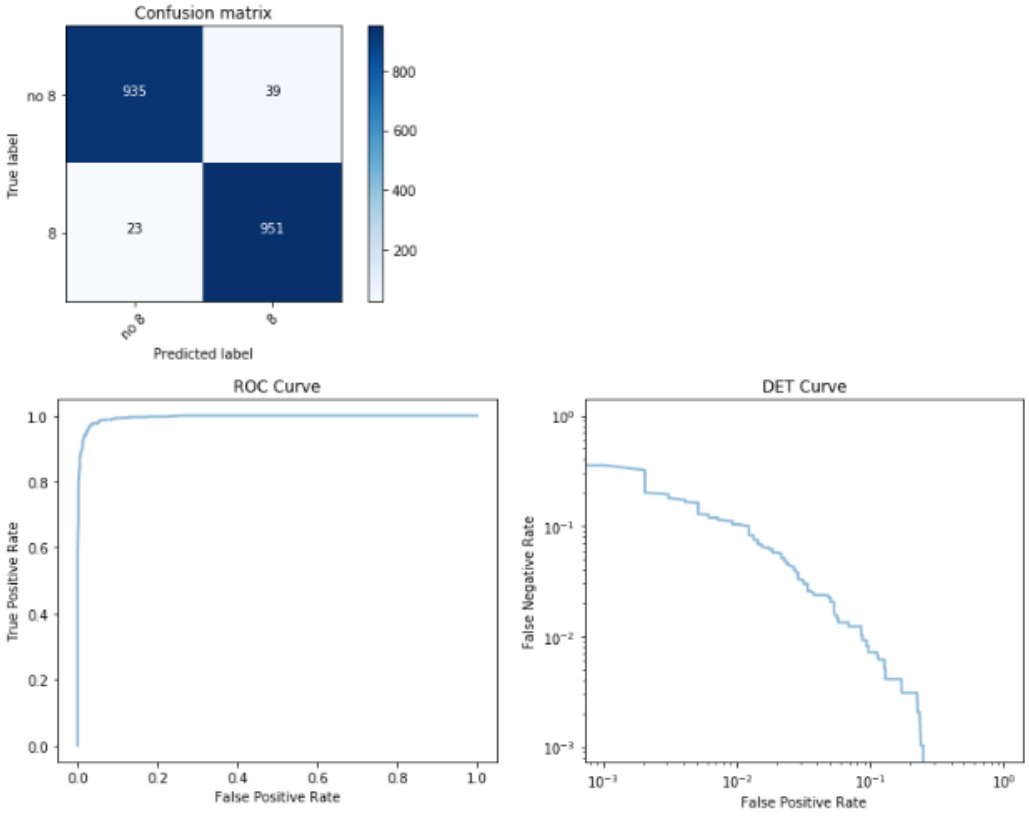

Figura 12: matriz de confusión, curvas ROC y DET para taxa de aprendizaje = 1

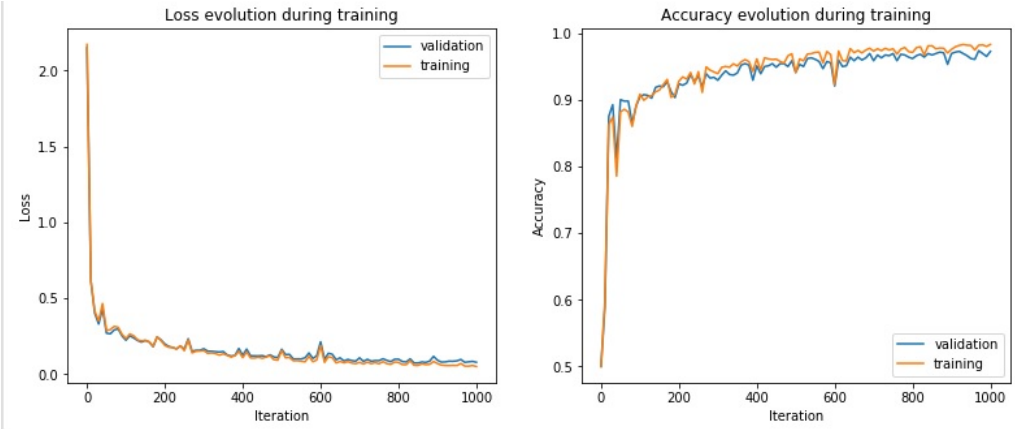

Figura 13: Curvas de aprendizaje para taxa de aprendizaje = 1

resultados para $\mu = 10$:

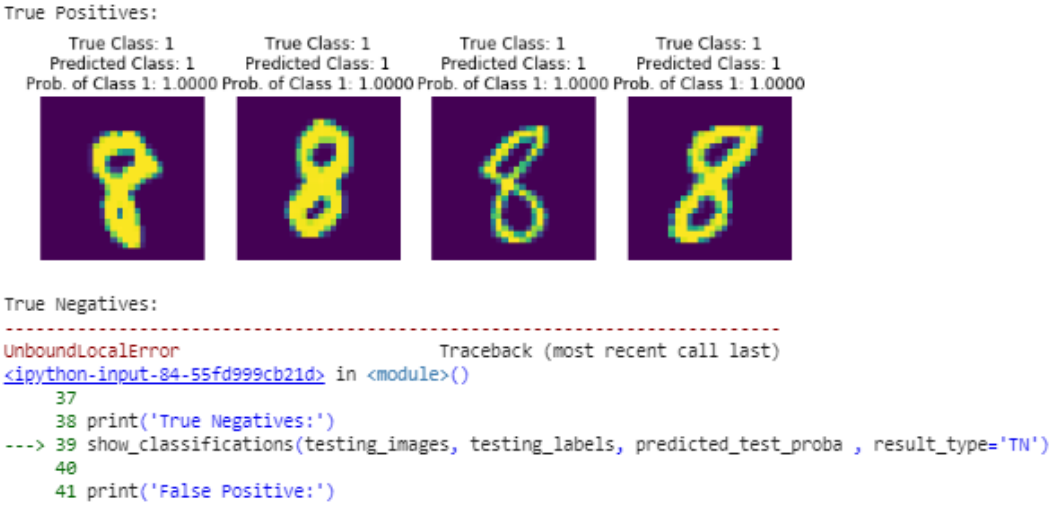

Figura 14: Curvas de aprendizaje para taxa de aprendizaje = 10

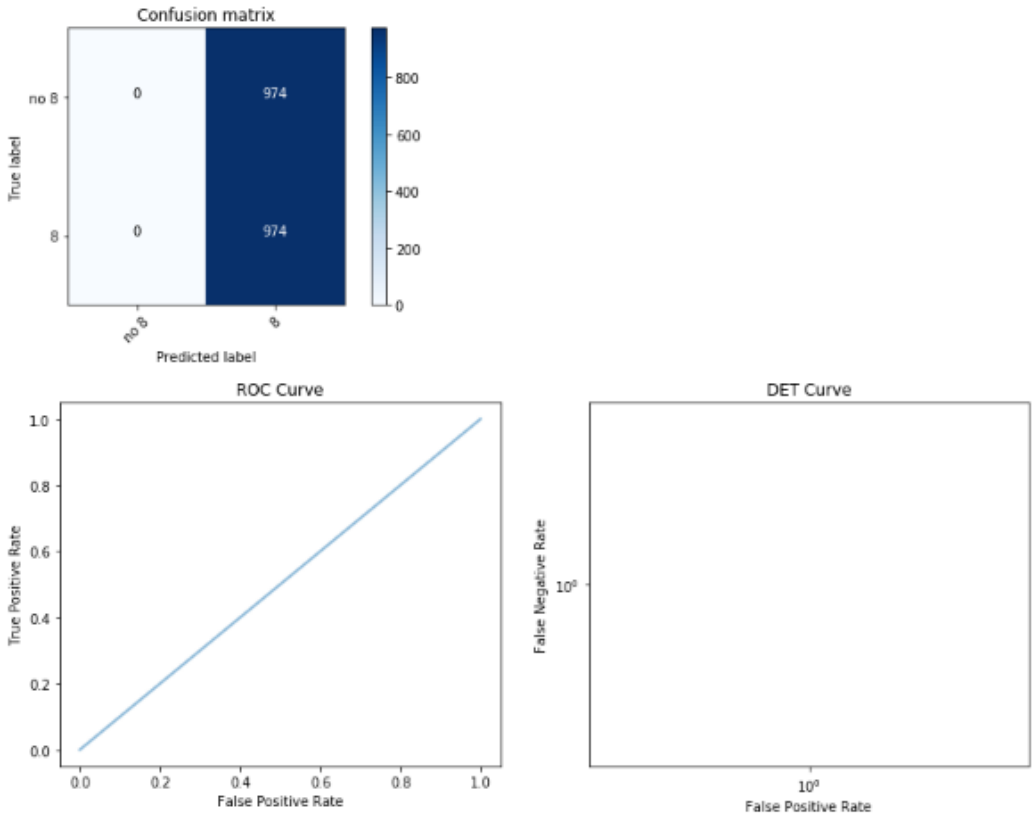

Figura 15: matriz de confusión, curvas ROC y DET para taxa de aprendizaje = 10

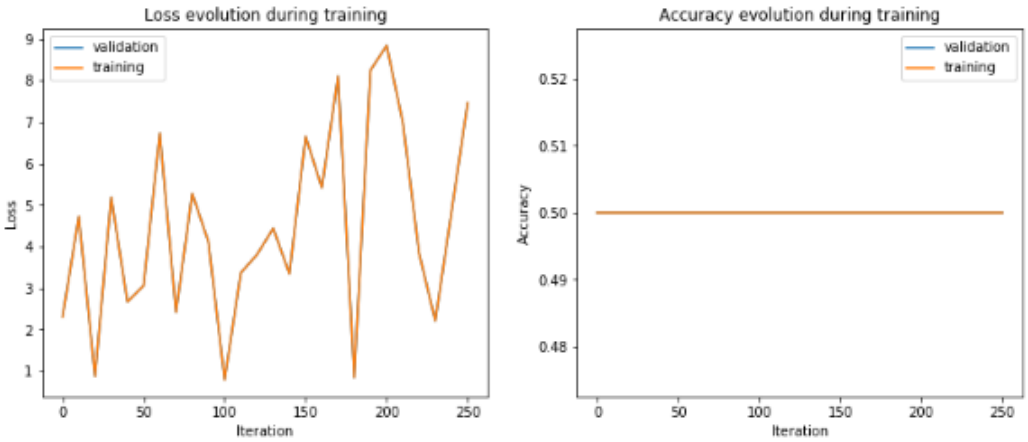

Figura 16: Curvas de aprendizaje para taxa de aprendizaje = 10

3. **Sol:** Comparando los valores de precisión, la capa de 25 neuronas es la más apropiada.
   Resultados N=25:

```
Validation accuracy 0.968 +/- 0.004
[0.9621212  0.9729437  0.969697   0.96753246 0.9686147 ]
Train accuracy 0.975 +/- 0.002
[0.97105217 0.9771757  0.9747634  0.97606236 0.97689736]
```

Figura 17: Promedio de tasa de acierto para N=25

```
Training results:
TP: 5225, TN: 5304, FP: 85, FN: 164
97.6897% Accuracy (Porcentaje de clasificaciones correctas)
98.3992% Precision
96.9568% Recall

Validation results:
TP: 441, TN: 454, FP: 8, FN: 21
96.8615% Accuracy (Porcentaje de clasificaciones correctas)
98.2183% Precision
95.4545% Recall

Test results:
TP: 937, TN: 949, FP: 25, FN: 37
96.8172% Accuracy (Porcentaje de clasificaciones correctas)
97.4012% Precision
96.2012% Recall
```

Figura 18: conjunto de validación para N=25

Resultados N=1:

```
val_acc_history = np.array([run['val_acc_history'][-1] for run in stats_history])
val_acc_mean = val_acc_history.mean()
val_acc_std = val_acc_history.std()
print('Validation accuracy %.3f +/- %.3f' % (val_acc_mean, val_acc_std))
print(val_acc_history)

train_acc_history = np.array([run['train_acc_history'][-1] for run in stats_history])
train_acc_mean = train_acc_history.mean()
train_acc_std = train_acc_history.std()
print('Train accuracy %.3f +/- %.3f' % (train_acc_mean, train_acc_std))
print(train_acc_history)
```

```
Validation accuracy 0.902 +/- 0.009
[0.8950216  0.90800864 0.88852817 0.9101732  0.90909094]
Train accuracy 0.910 +/- 0.002
[0.9072184  0.91250694 0.90740395 0.90981627 0.91297084]
```

Figura 19: Promedio de tasa de acierto para N=1

```
Confusion matrix, without normalization
[[879  95]
 [ 85 889]]
Training results:
TP: 4871, TN: 4969, FP: 420, FN: 518
91.2971% Accuracy (Porcentaje de clasificaciones correctas)
92.0620% Precision
90.3878% Recall

Validation results:
TP: 416, TN: 424, FP: 38, FN: 46
90.9091% Accuracy (Porcentaje de clasificaciones correctas)
91.6300% Precision
90.0433% Recall

Test results:
TP: 889, TN: 879, FP: 95, FN: 85
90.7598% Accuracy (Porcentaje de clasificaciones correctas)
90.3455% Precision
91.2731% Recall
```

Figura 20: conjunto de validación para sobreajuste para N=1

Resultados N=10:

```
Validation accuracy 0.955 +/- 0.005
[0.9534632  0.9491342  0.9512987  0.96428573 0.95454544]
Train accuracy 0.961 +/- 0.005
[0.9603823  0.9561143  0.9570421  0.9688254  0.96335125]
```

Figura 21: Promedio de tasa de acierto para N=10

```
Confusion matrix, without normalization
[[919  55]
 [ 39 935]]
Training results:
TP: 5198, TN: 5185, FP: 204, FN: 191
96.3351% Accuracy (Porcentaje de clasificaciones correctas)
96.2236% Precision
96.4557% Recall

Validation results:
TP: 434, TN: 448, FP: 14, FN: 28
95.4545% Accuracy (Porcentaje de clasificaciones correctas)
96.8750% Precision
93.9394% Recall

Test results:
TP: 935, TN: 919, FP: 55, FN: 39
95.1745% Accuracy (Porcentaje de clasificaciones correctas)
94.4444% Precision
95.9959% Recall
```

Figura 22: conjunto de validación para sobreajuste para N=1

Resultados N=100:

```
Validation accuracy 0.940 +/- 0.018
[0.92424244 0.9556277  0.9134199  0.95238096 0.9534632 ]
Train accuracy 0.944 +/- 0.026
[0.9127853  0.9692893  0.91167194 0.96288735 0.96400076]
```

Figura 23: Promedio de tasa de acierto para N=100

```
Confusion matrix, without normalization
[[932  42]
 [ 46 928]]
Training results:
TP: 5171, TN: 5219, FP: 170, FN: 218
96.4001% Accuracy (Porcentaje de clasificaciones correctas)
96.8171% Precision
95.9547% Recall

Validation results:
TP: 435, TN: 446, FP: 16, FN: 27
95.3463% Accuracy (Porcentaje de clasificaciones correctas)
96.4523% Precision
94.1558% Recall

Test results:
TP: 928, TN: 932, FP: 42, FN: 46
95.4825% Accuracy (Porcentaje de clasificaciones correctas)
95.6701% Precision
95.2772% Recall
```

Figura 24: conjunto de validación para sobreajuste para N=100

Sol: El sobreajuste se encuentra aquí al observar la suavidad y el número de iteraciones en las curvas de aprendizaje y los valores de precisión en el entrenamiento y la validación que sugieren memorización dos dados.

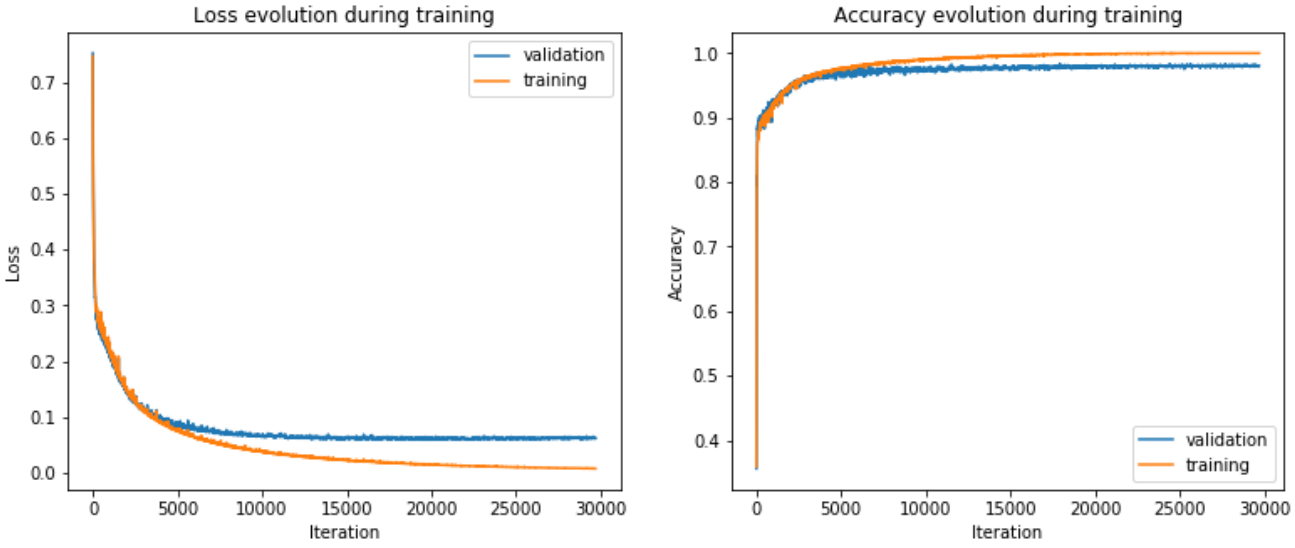

Figura 25: Promedio de tasa de acierto

```
Validation accuracy 0.989 +/- 0.000
[0.98917747]
Train accuracy 1.000 +/- 0.000
[1.]
```

Figura 26: conjunto de validación para sobreajuste

4. .

(a) **Sol:** $TPR = \frac{TP}{TP+FN}$
$FPR = \frac{FP}{FP+TN}$
El siguiente experimento se realizó con valores tomados del modelo.
Umbral=0.8:
$TPR = \frac{TP}{TP+FN} = \frac{435}{435+27} = 0{,}94156$
$FPR = \frac{FP}{FP+TN} = \frac{4}{4+458} = 0{,}0086580$
Umbral=0.5:
$TPR = \frac{TP}{TP+FN} = \frac{451}{451+11} = 0{,}976$
$FPR = \frac{FP}{FP+TN} = \frac{12}{12+450} = 0{,}0256$
Umbral=0.2:

$TPR = \frac{TP}{TP+FN} = \frac{457}{457+5} = 0{,}9891$
$FPR = \frac{FP}{FP+TN} = \frac{24}{24+438} = 0{,}0519$
Observamos que las tasas de FPR y TPR son inversamente proporcionales al umbral de clasificación. Aumentar el valor umbral disminuye las tasas y viceversa.

En este caso, nos movemos en la curva ROC desde el umbral = 0 a 1 de derecha a izquierda, de arriba a abajo.

(b) **Sol:** $TPR = \frac{TP}{N_1}$
$TPR = \frac{FP}{N_2}$
$N_1 = TP + FP$
$N_2 = TN + FP$
Como es balanceado: $N_1 = N_2 = N/2$, $N_1 + N_2 = N$
Además $T = TP + FP = N_1 * TPR + N_2 * FPR$

$$T = \frac{N}{2} * (TPR + FPR)$$

A medida que avanzamos sobre la curva ROC, TP y FP disminuyen. A medida que avanzamos sobre la curva ROC, TP y FP disminuyen. T cambia con TP y FP, cambiando el umbral, cambiamos TP y FP. Bajar el umbral aumenta T.

(c) **Sol:** Si queremos T=400 y N=1000:

$$400 = \frac{1000}{2} * (TPR + FPR)$$

$$(TPR + FPR) = \frac{4}{5} = 0{,}8$$

Con el siguiente loop, se ha alcanzado el umbral=0.913.

```
predicted_validation_proba = mlp.predict_proba(validation_images)
threshold = 0.5
TP, FP, FN, TN = detection_performance_given_threshold(validation_labels, predicted_validation_proba[:, 1], threshold=threshold)
print('TP: %d, TN: %d, FP: %d, FN: %d' %(TP,TN,FP,FN))

T= TP+FP
TPR= TP/(TP+FN)
FPR= FP/(FP+TN)
threshold = 0

while 1:
  TP, FP, FN, TN = detection_performance_given_threshold(validation_labels, predicted_validation_proba[:, 1], threshold=threshold)
  T= TP+FP
  TPR= TP/(TP+FN)
  FPR= FP/(FP+TN)
  if (TPR+FPR)<=0.8: #T==400:
    print('umbral= ',threshold)
    print('TPR', (TPR))
    print('FPR', (FPR))

    print('soma das tasas=', (TPR+FPR))
    break
  else:
    threshold = threshold+ 0.001
```

```
TP: 437, TN: 449, FP: 13, FN: 25
umbral=  0.9130000000000007
TPR 0.7943722943722944
FPR 0.0021645021645021645
soma das tasas= 0.7965367965367965
```

Figura 27: Umbral con loop

2. Tercera parte: programación

Sol: Cuanto menor sea el valor de la razon por iteración en los siguientes gráficos, menor será el cambio porcentual en la actualización de peso. Lo que significa que se acerca a un mínimo local

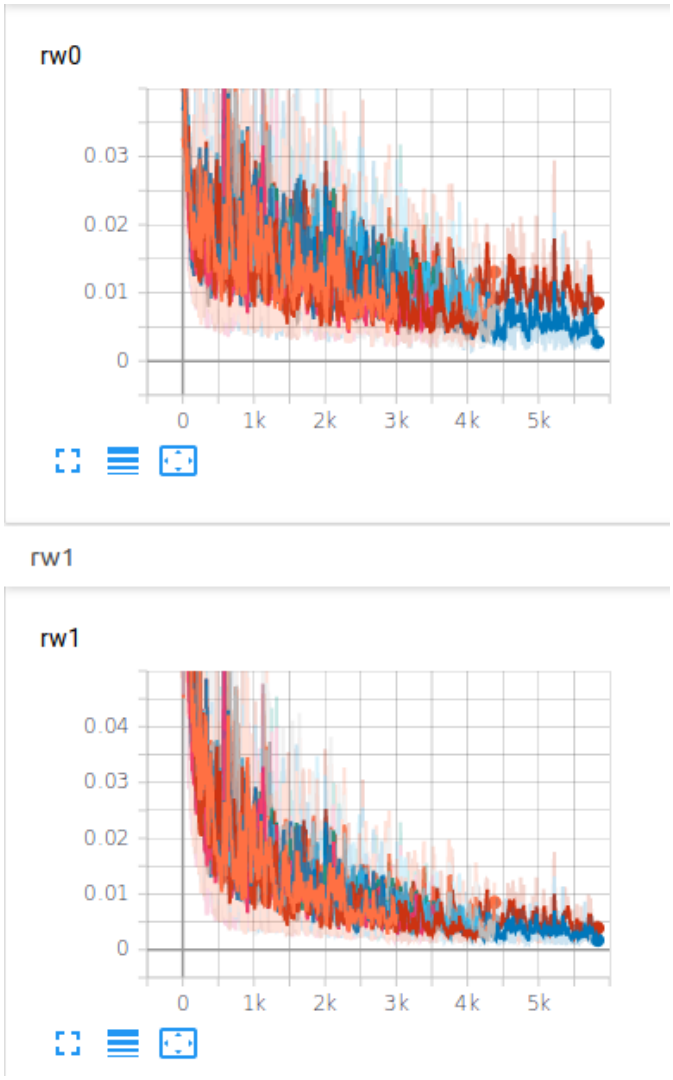

Figura 28: Graficos de Razon dw/d para μ=1, rw0=capa oculta, rw1= capa salida

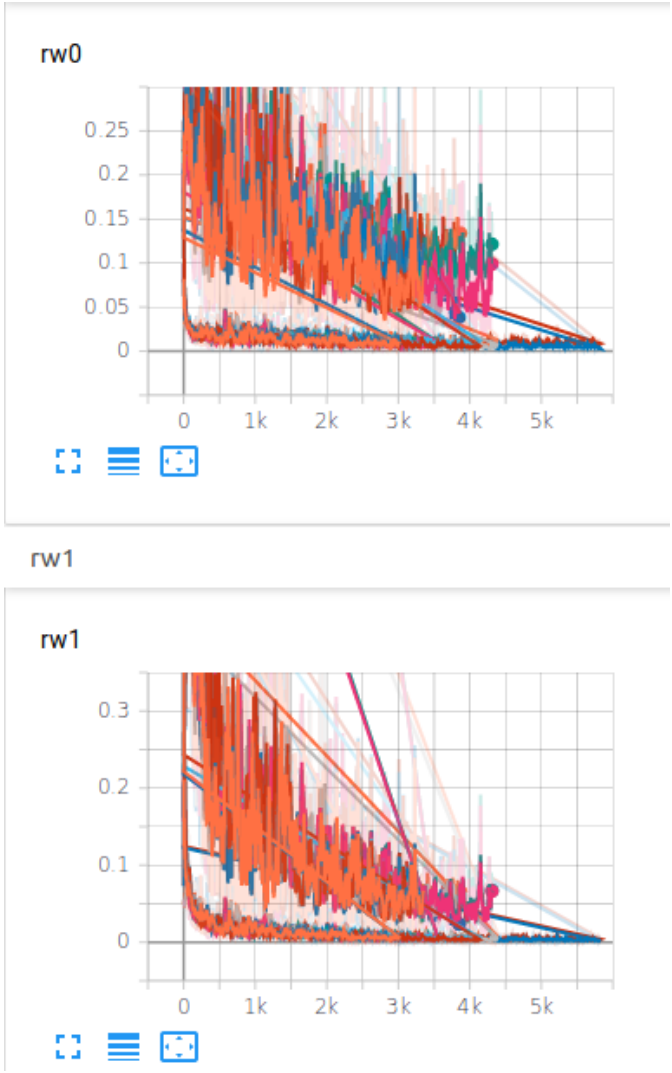

Figura 29: Graficos de Razon dw/d para μ=10, rw0=capa oculta, rw1= capa salida

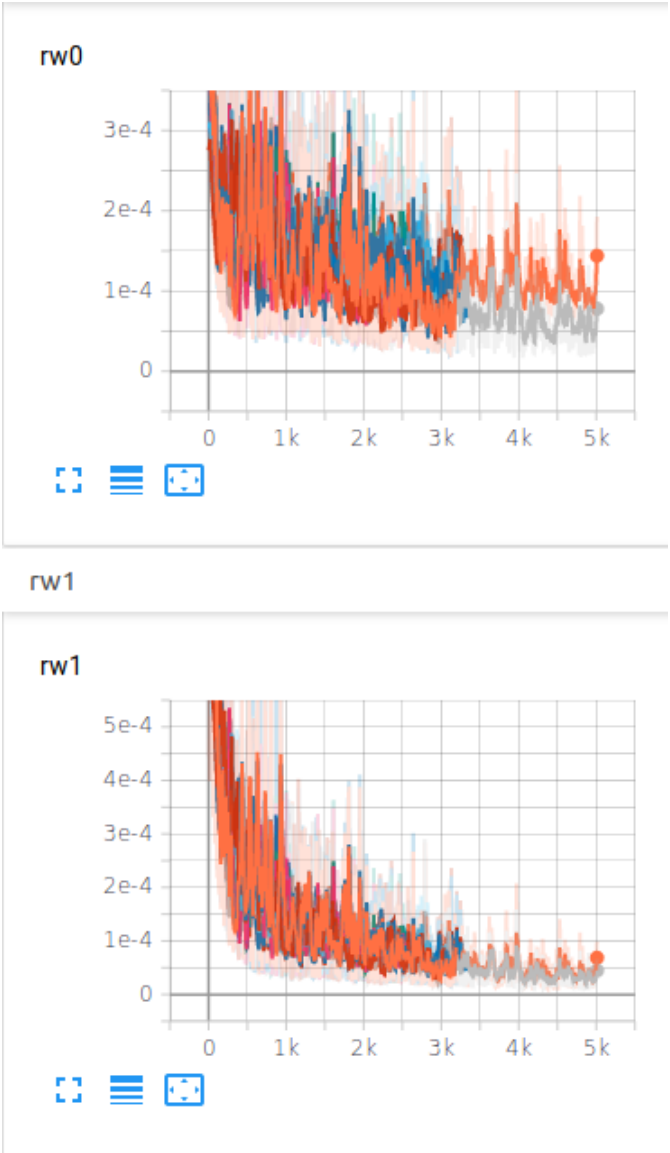

Figura 30: Graficos de Razon dw/d para μ=0.01, rw0=capa oculta, rw1= capa salida

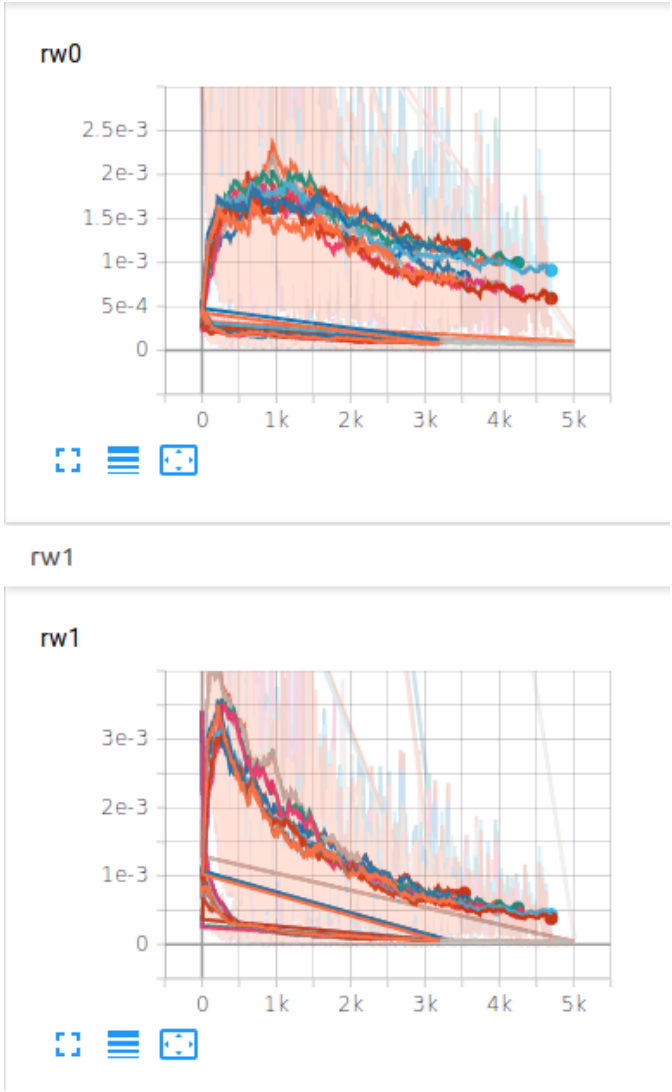

Figura 31: Graficos de Razon dw/d para μ=0.1, rw0=capa oculta, rw1= capa salida

Sol: Observamos que, en términos generales, para una mayor tasa de aprendizaje, menor será la razón de $\Delta w/w$ para el mismo número de iteraciones.

Código en el link:

$https://colab.research.google.com/gist/AndreTeixeira1998/cf20baeb25f9bb7513cc2af5a40cb821/tarea1\_notebook\_la\_luz.ipynb$