Tarea 4 de Aprendizaje de Máquinas

Redes Neuronales

Miguel Espinoza Pereira
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
Santiago, Chile

I. Parte A)

Se cargaron los datos de MNIST mediante Torchvision, y se modificaron para poder trabajar con vectores de 784 features en vez de imagenes de 28x28. El detalle de la implementación se encuentra en el Jupyter Notebook.

II. Parte B)

Además del algoritmo típico SGD, *Stochastic Gradient Descent*, se investigaron los algoritmos *Adagrad*, *RMSProp* y *Adam*.

- **Adagrad**: Adagrad es un algoritmo de optimización que actualiza el *learning rate* en cada iteración. El learning rate actual depende de la suma de los gradientes de las iteraciones pasadas elevados al cuadrado. En específico, como se ve en la ecuación 1, el learning se divide por esta suma y se le agrega un término ϵ para que nunca ocurra que el gradiente se divida por cero.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t. \tag{1}$$

  La diagonal de la matriz G_t contiene tiene el cuadrado de la suma de los gradientes hasta el tiempo t, como aparece en la figura 1.

$$G_t^{(i,i)} = \sum_{\tau=1}^{t} (g_\tau^{(i)})^2,$$

Fig. 1. Matriz G_t.

  La idea de Adagrad es que haya un learning rate distinto por cada dimensión, ya que es de esperar que no todas las dimensiones converjan de igual manera al mínimo, por lo que lo que hace la ecuación 1 es disminuir el learning rate para direcciones muy frecuentes y aumentar el learning rate para direcciones poco frecuentes.
- **RMSProp**: Este algoritmo de optimización intenta remediar un problema de Adagrad, ya que al dividir el learning rate por una suma que cada vez se va haciendo más grande, se provoca que nel learning rate tienda a cero, y la red no aprenda nada. RMSProp utiliza el promedio de los gradientes pasados elevados al cuadrado, como se aprecia en la ecuación 2.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2}$$

  Además, para calcular el promedio, multiplica los gradientes de las iteraciones pasadas por un hiperparámetro que provoca que mientras más antiguos sean los gradientes, menos influencia tendrán sobre el promedio, de forma similar al algoritmo *Momentum*, como se ve en la ecuación 3.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{3}$$

- **Adam**: *Adam* (Adaptive Moment Estimation), es uno de los algoritmos de optimización más utilizados por sus buenos resultados. Es una unión entre los algoritmos RMSProp y Momentum, ya que guarda un promedio decadente del cuadrado de los gradientes pasados y un promedio decadente los gradientes pasados, respectivamente. Sus hiperparámetros son β_1 y β_2, β_1 es la tasa de decrecimiento exponencial para el estimador del primer momento y β_2 para el estimador del segundo momento, ecuación 4.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{4}$$

  Sin embargo estos momentos están sesgados, por lo que se calculan los estimadores, ecuación 5.

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{5}$$

  Finalmente el óptimo para la iteración actual está dado por la ecuación 6.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{6}$$

Para resolver la tarea se decidió usar Adam, debido a la superioridad que presenta con respecto al resto de los métodos de optimización.

III. Parte C)

Se creó una clase que hereda la clase `nn.module` de Pytorch, en esta clase se puede elegir las neuronas de la capa de entrada y de salida, si serán 1 o 2 capas ocultas y de cuántas neuronas cada una, y por último la probabilidad de dropout. Más detalles en el Jupyter Notebook.

IV. Parte D)

A. Metodologia

Se crearon 4 arquitecturas de redes neuronales de acuerdo a lo especificado en el enunciado, que se presentan en la tabla I.

Arquitectura	Entrada	Oculta 1	Oculta 2	Salida
1 capa	784	100	-	10
2 capas	784	100	100	10
1 capa con reg. L2	784	100	-	10
1 capa con dropout	784	100	-	10

TABLE I
ARQUITECTURAS USADAS EN LA TAREA.

En cuanto a los hiperparámetros, para la capa con regularización L2, se ocupó un *weight_decay* de 10^{-5}. Para la capa con dropout se utilizó una probabilidad de 0.5.

Se graficó el error en función de las épocas para cada arquitectura, cada 50 iteraciones en los datos de entrenamiento y cada una época en los datos de validación. Cada época contenía 500 iteraciones, ya que los 50000 datos de entrenamiento se dividieron en batches de 100 vectores cada uno.

Se calculó la precisión en los datos de test para cada arquitectura y también la matriz de confusión, la cual permitió ver cuales eran las clases que más costaba discriminar entre sí y cuales eran las más precisas.

B. Resultados y Análisis

Como se ve en los gráficos del anexo, el error va disminuyendo a medida que pasan las épocas. El loss inicialmente disminuye muý rápido, pero a medida que pasan las iteraciones se va estancando, lo que quiere decir que alcanzó un mínimo, ya sea local o global, aunque también pudo haber llegado a un punto silla. De cualquier forma, una vez que llega a esos puntos le es bastante díficil salir y encontrar otro punto con menor error. Se aprecia además, que el error en los datos de entrenamiento no es estrictamente decreciente, sino que sube y baja al pasar las iteraciones, principalmente cuando se estanca. Esto ocurre porque el gradiente puede dar un salto grande que sobrepase el mínimo y quedar en un punto con mayor error que el inicial. El error de los datos de validación si es decreciente, por lo que a pesar del comportamiento errático el error efectivamente desciende.

La arquitectura con menor error fue la arquitectura de dos capas esto se debe a la complejidad de la red, al tener dos capas tiene más parámetros que optimizar en cada iteración, por lo que puede conseguir llegar a un error más bajo. En la tabla II se presenta el error en los datos de validación para cada arquitectura.

Arquitectura	Error
1 capa	0.1192
2 capas	0.1009
1 capa con reg. L2	0.1164
1 capa con dropout	0.1306

TABLE II
ERROR EN LAS DISTINTAS ARQUITECTURAS.

En general para todas las arquitecturas se obtuvo una precisión mayor al 96%, siendo la mejor la arquitectura de dos capas, con una precisión de 97.12%. Es interesante notar que no se produce overfitting a pesar de que haya una capa extra, ya que los resultados en los datos de test son excelentes. Si se hubiera sobrejustado a los datos de entrenamiento el error hubiera sido el más bajo, pero al calcular la precisión se hubiera obtenido un resultado menor que el resto de arquitecturas, ya que hubiera aprendido características específicas de los datos de entrenamiento. El accuracy obtenido en cada red se muestra en la tabla III.

Arquitectura	Precisión
1 capa	96.43%
2 capas	97.12%
1 capa con reg. L2	96.54%
1 capa con dropout	96.13%

TABLE III
PRECISIÓN PARA DATOS DE TEST EN LAS ARQUITECTURAS.

Por último, al analizar la matriz de confusión, se aprecia que las diagonales concentran los valores más altos, lo que quiere decir que la mayoría de las clases son predichas correctamente. En la figura 2 se muestra la matriz de confusión para la arquitectura de 2 capas, que como se explicó anteriormente fue con la se obtuvo menor error y mayor precisión.

```
[[ 962    0    2    2    0    2    9    1    2    0]
 [   0 1119    4    0    0    1    4    0    7    0]
 [   3    1 1008    4    0    0    5    5    6    0]
 [   0    0    8  991    0    1    0    5    5    0]
 [   2    0    6    0  952    1    5    2    2   12]
 [   2    1    1   13    2  858    7    0    6    2]
 [   5    3    2    1    3    2  941    0    1    0]
 [   0    5   13    7    0    1    0  991    3    8]
 [   3    0    7   11    3    6    6    4  932    2]
 [   3    5    1   13   15    4    1    7    2  958]]
Precision de 0 : 98.16 %
Precision de 1 : 98.59 %
Precision de 2 : 97.67 %
Precision de 3 : 98.12 %
Precision de 4 : 96.95 %
Precision de 5 : 96.19 %
Precision de 6 : 98.23 %
Precision de 7 : 96.40 %
Precision de 8 : 95.69 %
Precision de 9 : 94.95 %
```

Fig. 2. Matriz de confusión para la red de 2 capas ocultas. Las filas de la matriz representan la clase real y las columnas la clase predicha por la red. También se muestra la precisión para cada clase.

En el anexo se muestran el resto de gráficos por si son de interés.

V. Anexo

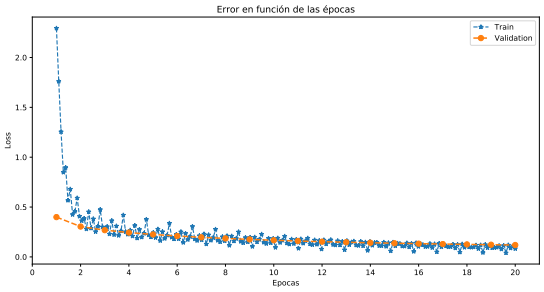

Fig. 3. Error en red de 1 capa.

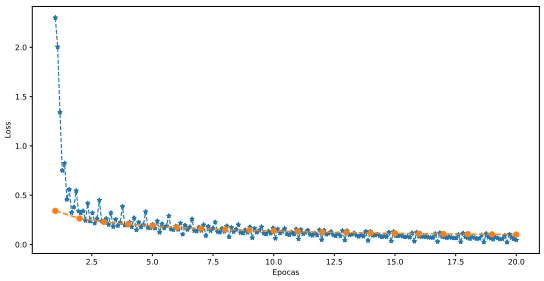

Fig. 4. Error en red de 2 capas.

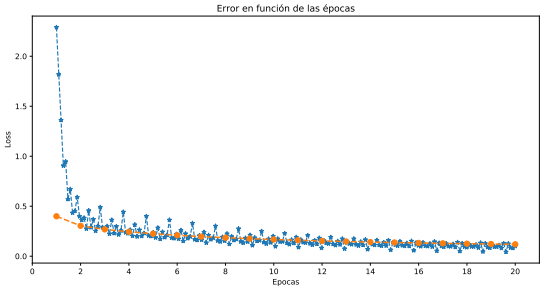

Fig. 5. Error en red de 1 capa con regularización L2.

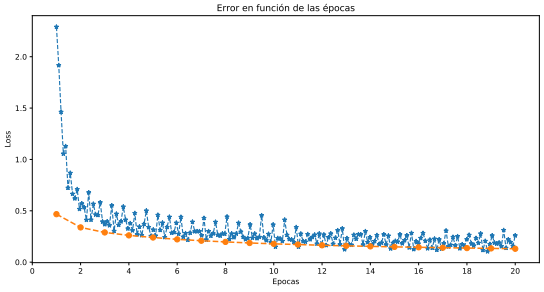

Fig. 6. Error en red de 1 capa con dropout.

```
[[ 961    0    3    2    0    4    7    1    2    0]
 [   0 1118    4    0    0    1    5    0    7    0]
 [   4    2 1000    5    5    0    3    5    8    0]
 [   0    0   10  972    0    7    0    6   11    4]
 [   1    0    7    1  954    0    3    3    2   11]
 [   4    1    0   19    3  840   10    1    9    5]
 [   6    3    3    1    6    6  930    0    3    0]
 [   0    6   14    6    0    1    0  989    1   11]
 [   3    1    7   14    7    3    4    7  924    4]
 [   3    6    1   11   16    4    1    9    3  955]]
Precision de 0 : 98.06 %
Precision de 1 : 98.50 %
Precision de 2 : 96.90 %
Precision de 3 : 96.24 %
Precision de 4 : 97.15 %
Precision de 5 : 94.17 %
Precision de 6 : 97.08 %
Precision de 7 : 96.21 %
Precision de 8 : 94.87 %
Precision de 9 : 94.65 %
```

Fig. 7. Matriz de confusión en red de 1 capa.

```
[[ 965    0    2    1    0    2    7    1    2    0]
 [   0 1121    4    0    0    1    4    0    5    0]
 [   5    1 1001    4    0    1    6    7    7    0]
 [   0    0    8  981    0    3    1   10    5    2]
 [   0    1    7    0  951    0    8    2    3   10]
 [   3    2    1   19    2  841    9    2    9    4]
 [   4    3    4    0    5    5  934    0    3    0]
 [   0    6   14    7    0    1    0  991    0    9]
 [   3    3    5   17    7    7    5    7  917    3]
 [   5    6    2   13   18    2    0    9    2  952]]
Precision de 0 : 98.47 %
Precision de 1 : 98.77 %
Precision de 2 : 97.00 %
Precision de 3 : 97.13 %
Precision de 4 : 96.84 %
Precision de 5 : 94.28 %
Precision de 6 : 97.49 %
Precision de 7 : 96.40 %
Precision de 8 : 94.15 %
Precision de 9 : 94.35 %
```

Fig. 8. Matriz de confusión en red de 1 capa con regularización.

```
[[ 967    0    1    2    0    3    5    1    1    0]
 [   0 1123    2    2    0    1    3    1    3    0]
 [   4    1  989    6    7    1    4    8   12    0]
 [   0    1    8  975    1    5    0   11    9    0]
 [   1    0    4    0  952    0    8    1    3   13]
 [   5    1    1   24    3  827   10    3   11    7]
 [   8    3    1    1    7    5  929    0    4    0]
 [   2    7   17    7    2    0    0  983    0   10]
 [   5    4    5   11    6    6    7    8  918    4]
 [   5    6    1   11   16    5    0   10    5  950]]
Precision de 0 : 98.67 %
Precision de 1 : 98.94 %
Precision de 2 : 95.83 %
Precision de 3 : 96.53 %
Precision de 4 : 96.95 %
Precision de 5 : 92.71 %
Precision de 6 : 96.97 %
Precision de 7 : 95.62 %
Precision de 8 : 94.25 %
Precision de 9 : 94.15 %
```

Fig. 9. Matriz de confusión en red de 1 capa con dropout.