Tarea 4
Redes Neuronales

Profesor: Felipe Tobar
Auxiliar: Mauricio Araneda, Alejandro Cuevas, Mauricio Romero
Alumno: Mario Vicuña
Fecha: 25/06/2019

I. Redes Neuronales y Problema propuesto

En esta sección se abordan brevemente algunos conceptos básicos utilizados en el presente trabajo.

A grandes rasgos, se entiende por Red Neuronal a un modelo computacional de estructura jerárquica, es decir que se organiza en forma de capas o subconjuntos de sus unidades básicas, las neuronas. Por medio de operaciones matriciales y las denominadas "funciones de activación" para incluir no linealides, se propone este tipo de arquitecturas para tareas de modelamiento. En este mismo contexto, se entiende por regularización a cualquier modificación que se hace a un algoritmo de aprendizaje con tal de mejorar su gerenalización. Para este trabajo se considerarán la regularización $L2$ y *dropout*. La primera considera incluir en el funcional de costo la magnitud de los parámetros (pesos) de la red a modo de *prior* sobre el espacio de estos. Por su parte, *dropout* pretende evitar codependencias entre las distintas unidades de la red al incluir un factor aleatorio sobre los pesos que las conectan durante el entrenamiento, eliminandolos azarosamente para cada *batch*. En este trabajo se busca analizar algunos algoritmos de optimización de Redes Neuronales y efectos paramétricos y de regularizadores, considerando el clásico problema del *dataset MNIST*.

II. Algoritmos de Optimización

- **Momentum**: Una extensión natural del gradiente descendente (*SGD*) clásico es la inclusión de *momentum*. La escencia de este algoritmo recide en que el gradiente posea cierto grado de "memoria" o inercia de modo tal que este se vea amplificado en las direcciones en las que continuamente este aumenta su magnitud y disminuido en aquellas en que tenga un comportamiento errático u oscilatorio. Luego, el vector de actualización v_t está regido por la ecuación 1. Se aprecia de dicha ecuación que este es prácticamente idéntico al algoritmo *SGD* salvo por la inclusión de la actualización anterior y un peso asociado a esta por medio de un hiperparámetro.

$$v_t = \gamma v_{t-1} + \eta \nabla J_\theta(\theta), \Delta\theta_t = -v_t \tag{1}$$

- **AdaGrad**: Este algoritmo de optimización sigue un principio similar al de *momentum* en cuanto a adaptar el vector de actualización en base a direcciones "relevantes", obteniendo una tasa de aprendizaje propia para cada una de estas direcciones. Estas direcciones son aquellas correspondientes a parámetros asociados a características poco frecuentes, es decir se fomenta el movimiento en los parámetros más *sparse*.
  En comparación con *SGD*, *AdaGrad* posee la misma cantidad de parámetros, pero el *learning rate* de este último suele no ajustarse manualmente pues es el mismo algoritmo el que se encarga de manipular su valor durante el entrenamiento. Además se destaca por ser particularmente útil para *datasets sparse*. No obstante, tiene la desventaja de que se suele presentar desvanecimiento de la tasa de aprendizaje, dada su definición. Para este algoritmo, la ecuación 1 se convierte en la ecuación 2 (eliminiando el momentum γ).

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\eta}{\sqrt{G_{t,ii}}} \nabla J_\theta(\theta_{t,i}), G_t = \sum_{i=1}^{m} J_\theta(\theta_{t,i}) J_\theta^T(\theta_{t,i}) \tag{2}$$

- **RMSProp**: A grandes rasgos, *RMSProp* es significativamente similar a *AdaGrad*, incluyendo una media móvil (controlada por medio de un hiperparámetro extra) con la cual se pondera la dinámica de la tasa de aprendizaje. Este algoritmo se propuso para palear el desvanecimiento de la tasa de aprendizaje de *AdaGrad*, particularmente enfocado para optimizar Redes Recurrentes.

$$v_t = \gamma v_{t-1} + (1-\gamma)(\nabla J_\theta(\theta))^2, \Delta\theta = \frac{-\eta}{\sqrt{v_t}} \nabla J_\theta(\theta) \tag{3}$$

Debido a las buenas propiedades sobre *datasets sparse*, y al relativo poco efecto del hiperparámetro del optimizador, se utilizará *AdaGrad* para este trabajo. No se considerará *RMSProp* debido a que considera el ajuste de un hiperparámetro extra y pese a que en principio comparte las mismas propiedades que *AdaGrad*, el hecho de que la inclusión de este hiperparámetro extra tiene como objetivo adicional tratar Redes Recurrentes y series de tiempo (lo cual no es el caso de *MNIST*) no lo hace (en principio) el optimizador más adecuado.

III. Resultados sobre *MNIST*

En esta sección se discuten y analizan los resultados obtenidos. Como indicación del cuerpo docente, las figuras respectivas se encuentran en el Apéndice.

Dentro de las consideraciones hechas para poder comparar los modelos probados de la mejor forma posible, se ajustó el modelo base, una red *MLP* de una capa oculta, de tal forma que presentara una precisión del orden del $90\,\%$. Considerando

el optimizador escogido, *AdaGrad*, se optó por el valor de su hiperparámetro usado típicamente en la literatura. Además para todos los parámetros mencionados en el enunciado de la presente tarea, salvo por la tasa de aprendizaje y cantidad de neuronas por capa, se optó por los mencionados en dicho enunciado. La diferencia sustancial es que se eligió que cada capa oculta constara de 200 unidades, añadiendo más parámetros a la red y por ende más expresividad.

Del entrenamiento de esta red y de una equivalente pero con dos capas ocultas se obtuvieron las figuras 1 y 2, donde se muestra la evolución de la función de costo y de la precisión de ambas arquitecturas.

Para la selección de los modelos regularizados se presenta la tabla III, donde se muestra la precisión lograda sobre el conjunto de validación para los distintos casos. Escogiendo aquellos que presenten una mayor precisión, se aprecia en las figuras 3 y 4 como evoluciona la función pérdida y la precisión de los modelos regularizados durante el entrenamiento, y su comparación con el modelo base.

En las cuatro figuras mencionadas, se aprecia que a medida que avanzan las épocas, disminuye la pérdida de la red y aumenta su precisión, demostrando que está aprendiendo de la distribución de datos durante el proceso de entrenamiento, hasta alcanzar el desempeño que se muestra en las matrices de confusión de las figuras 5 a 8 correspondientes a la red de una capa oculta, la de dos capas, la regularizada con penalización $L2$ y con *Dropout*, respectivamente. Se hace evidente en estas cuatro imágenes que con un efectivo proceso de entrenamiento, las matrices de confusión se concentran en la diagonal, lo cual corresponde a los casos de correcta clasificación, mientras que los errores tienten a volverse relativamente marginales en proporción.

En cuanto a los efectos de las variables de cada experimento, basta con ver las figuras 1 y 2 para ver el efecto sustancial de mejora provocado por el aumento de parámetros entrenables, perceptible como un *gap* entre las curvas asociadas a cada modelo.

En este caso, el efecto de los regularizadores no es tan evidente a simple vista y requiere del análisis tanto de las curvas de pérdida y precisión durante el entrenamiento como del desempeño sobre el conjunto de prueba, resumido como precisión en la tabla III.

Si bien a primera vista las figuras 3 y 4 parecieran indicar que el modelo base sin regularizar se desempeña mejor dadas sus curvas de evolución, cabe recordar que estas curvas se realizan únicamente a partir del entrenamiento, mientras que el principal objetivo de las técnicas de regularización es atacar el sobreajuste y propiciar la capacidad de generalización de la red fuera del entrenamiento, lo cual se aprecia levemente si se compara la diferencia que presenta la precisión de los modelos durante el entrenamiento con la que presentan sobre el conjunto de prueba, siendo esta diferencia menor. Es decir, la deficiencia de los modelos regularizados en comparación el modelo base durante el entrenamiento se ve reducida durante prueba, lo cual implica que aún en desmedro del desemepeño en entrenamiento, la regularización favorece los buenos resultados sobre conjuntos de datos aún no observados por la red.

	$\lambda = 0.0001$	$\lambda = 0.001$	$\lambda = 0.01$
L2	96.04 %	95.22 %	90.46 %
	p = 0.3	**p = 0.5**	**p = 0.8**
Dropout	95.8 %	95.87 %	95.99 %

Tabla I
PRECISIÓN SOBRE EL CONJUNTO DE VALIDACIÓN PARA DISTINTOS REGULARIZADORES.

	Precisión en *Test*
MLP 1 Capa	96.43 %
MLP 2 Capas	97.04 %
MLP c/Reg. L2	96.12 %
MLP c/Dropout	96.06 %

Tabla II
PRECISIÓN SOBRE EL CONJUNTO DE PRUEBA PARA LOS DISTINTOS MODELOS ESCOGIDOS.

No obstante, respecto al efecto particular de cada regularización sobre las curvas de pérdida y precisión no se puede decir nada, debido en primer lugar al desempeño casi idéntico obtenido (particularmetne si se observa la curva de pérdida), sino que dichos efectos particulares se son apreciables en otros aspectos de la red, como la relación y dependencia entre pesos o la magnitud de estos, lo cual requiere otro análisis aparte de las curvas y métricas de desempeño.

APÉNDICE

A. *Gráficos obtenidos*

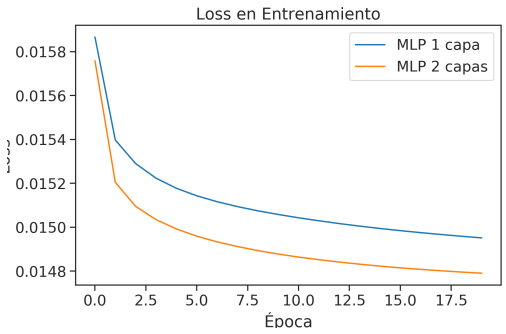

Figura 1. Función de costo durante entrenamiento para redes de 1 y 2 capas.

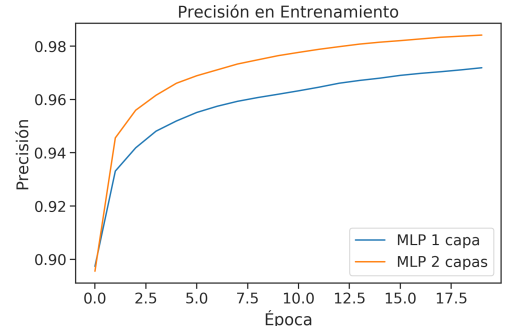

Figura 2. Precisión durante entrenamiento para redes de 1 y 2 capas.

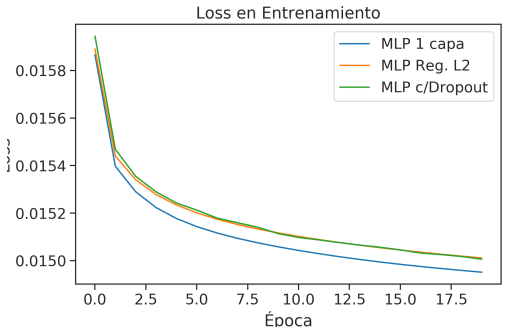

Figura 3. Función de costo durante entrenamiento para redes regularizadas, con regularización $L2$ y dropout, respectivamente.

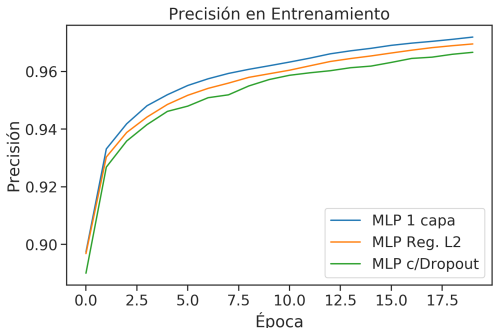

Figura 4. Precisión durante entrenamiento para redes regularizadas, con regularización $L2$ y dropout, respectivamente.

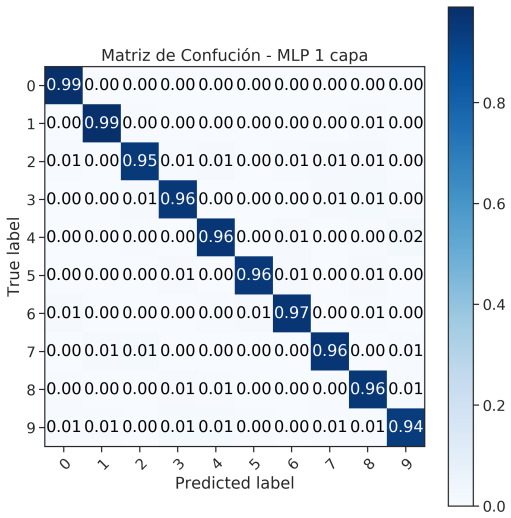

Figura 5. Matriz de Confusión para red *MLP* de una capa.

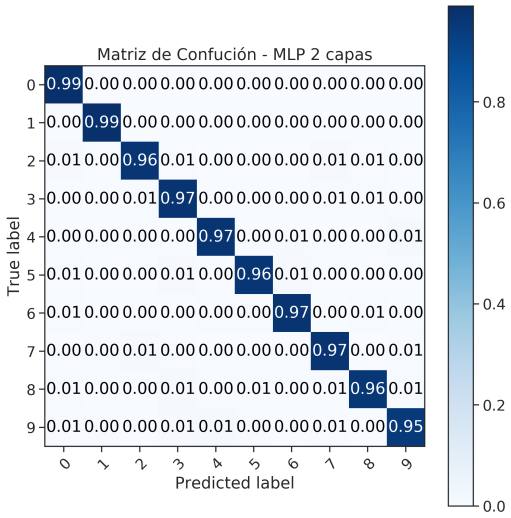

Figura 6. Matriz de Confusión para red *MLP* de dos capas ocultas.

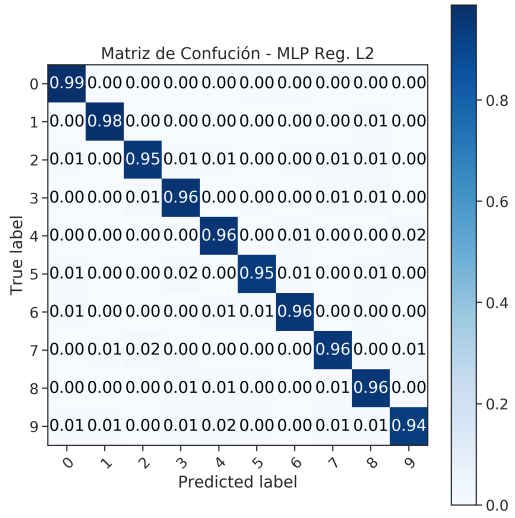

Figura 7. Matriz de Confusión para red *MLP* de una capa con regularización $L2$ de parámetro $= 0.0001$.

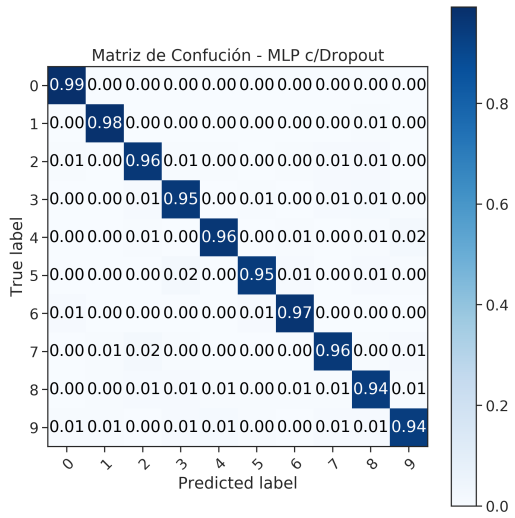

Figura 8. Matriz de Confusión para red *MLP* de una capa con regularización por *Dropout* de probabilidad $p = 0.8$.