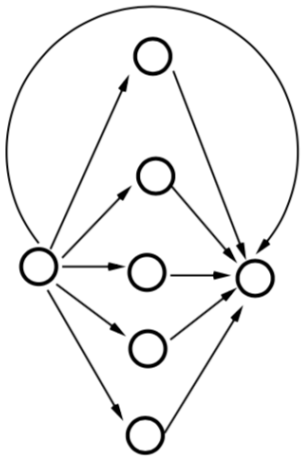
Redes Neuronales y Aprendizaje Profundo para Visión Artificial

Cuatrimestre: 2do de 2020

PRÁCTICA 4: INTRODUCCIÓN A KERAS

NOTA: Para todos los problemas explique porque utiliza la función o funciones de activación, y sobre todo porqué elige usar la función de costo utilizada. En caso que haya una relación entre la elección de la función de costo y la función de activación de la última capa indíquelo.

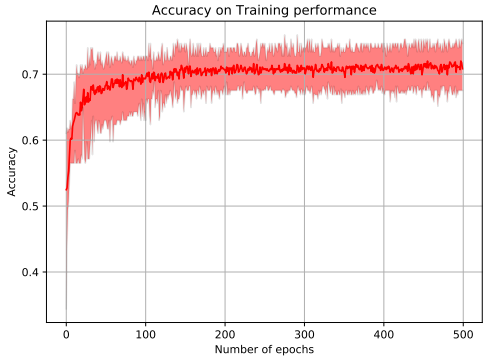
1. Usando los datos provisto por sklearn sobre el precio de los inmuebles en Boston (`from sklearn.datasets import load_boston`), entrenar un modelo de regresión lineal que permita predecir el precio de los inmuebles ($Wx + b$). Graficar la función de costo sobre los datos de validación y el ajuste final utilizando la función de matplotlib `scatter`. Normalizar los datos de entrada y utilizar $\sim 25\%$ para testing.
2. Implementar los problemas 3, 4 y 6 de la práctica 2 utilizando Keras.
3. El objetivo de este problema consiste en implementar una red neuronal que permita clasificar la revisiones realizadas en IMDB como positivas o negativas. Para ello vamos a trabajar con los datos que provee la librería keras (`datasets.imbd.load_data(num_words=10000)`). Los datos se encuentran codificados según el diccionario `imdb.get_word_index()`, donde cada palabra esta codificada con un número. Antes de poder trabajar con los datos es necesario preprocesar los mismos de forma similar a cómo se hizo con la BD mnist. Analizar el rendimiento de la red propuesta, y en caso de overfitting estudie el impacto que tienen las distintas variantes de regularización vistas en clase para mejorar la generalización de la red (L_2, BN y Drop out).
4. Resolver el problema anterior utilizando Embeddings y recordar hacer un padding. Luego, armar una nueva arquitectura utilizando capas convoluciones 1D en lugar de capas densas. Comparar la precisión entre ambos problemas y el mejor resultado obtenido del punto anterior.
5. Implementar la arquitectura de la figura para resolver el mapeo logístico ($x(t+1) = 4x(t)(1 - x(t))$ donde $x \in [0, 1]$). La función de activación de la neurona de salida es lineal. Estudiar el error de entrenamiento y el de generalización. (Nota: utilizar como función de activación tanh en la capa oculta.)



6. A partir de los registros de los pacientes almacenados en el archivo ”pima-indians-diabetes.csv" diseñar una red neuronal que permita predecir si el paciente puede tener diabetes o no. Para cada paciente la información que se tiene es la siguiente:

   - Cantidad de embarazos.
   - Concentración de glucosa plasmática a 2 horas en una prueba oral de tolerancia a la glucosa.
   - Presión arterial diastólica (mm Hg).
   - Grosor del pliegue de la piel del tríceps (mm).
   - Insulina en suero de 2 horas (mu U/ml).
   - Indice de masa corporal (peso en kg/(altura en m)^2).
   - Función de pedigrí de diabetes (información sobre la historia familiar y la influencia genética para tener diabetes).
   - Edad.

   Evaluar la cantidad de capas y el tipo de las mismas. En caso de observar overfitting proponga una forma de corregirlo. Utilice un esquema de 5-folding para obtener una medida más precisa sobre la generalización del método propuesto. Graficar la precisión del método propuesto por ejemplo como se presenta en la siguiente figura



¿A partir de una observación rápida de los datos, puede observar algo extraño en los mismos ? ¿Cómo se puede corregir este problema (evaluar el modelo con y sin corrección)? ¿Qué pasa si se incrementamos el número de neuronas, y si se introducen más capas ocultas (armar los modelos y evaluarlos)?

7. A partir de la base de datos de MNIST implementar un autoencoder que permita eliminar el ruido en las imágenes de dígitos manuscritos. Preprocesar los datos para que la imagen $I \in [0, 1]$. Agregar un ruido con una distribución Gaussiana de media 0 y std=0.5 y utilizar la función clip para fijar los valores nuevamente entre [0,1].

8. Proponga una arquitectura basada en capas densas para clasificar los dígitos de la base de datos MNIST. Ahora implementar una arquitectura que utilice capas convolucionales y compare los resultados. Graficar la precisión sobre los datos de entrenamiento y los de prueba para cada época de ambas arquitecturas. A partir de los gráficos de exactitud y error, analizar pros y cons de las distintas redes neuronales. Explicar a qué se deben las diferencias del rendimiento entre ambas arquitecturas. A su vez, explique para qué se utiliza la capa Dropout. ¿Describa la diferencia de utilizar una red con capas densas y otra con capas convolucionales, porque es mejor usar una u otra? ¿ Qué diferencia hay en la cantidad de parámetros de estas dos arquitecturas?

9. Utilizando el código del problema anterior analice que pasa cuando se utiliza una capa densa con la información de los pixels permutada espacialmente y que pasa si en lugar de capa densa se usa una convolucional donde en las capas convolucionar se asume un prior de forma espacial. Compare los resultados del problema anterior (densa y conv. para las curvas training y testing) con los resultados obtenidos haciendo la permutaciones espaciales de los pixels. Usar este código para permutar espacialmente los pixeles:

```
permutation = np.random.permutation(28*28)

x_train_perm = x_train.reshape(x_train.shape[0], -1)
x_train_perm = x_train_perm[:,permutation]
x_train_perm = x_train_perm.reshape(x_train.shape)

x_test_perm = x_test.reshape(x_test.shape[0], -1)
x_test_perm = x_test_perm[:,permutation]
x_test_perm = x_test_perm.reshape(x_test.shape)
```

10. Implementar la arquitectura de Alexnet y VGG16 para resolver el problema de la base de datos CIFAR-10 y CIFAR-100. Utilizar aumentación de datos y analizar los resultados obtenidos. Ajustar el tamaño de las capas (strides, padding, neuronas, etc.) y la profundidad de la red según los datos y los recursos de hardware disponibles.

    – AlexNet:

Layer	Features	size	Kernel size	Strides	Activation
Imagen	1	227x227x3	-	-	-
Conv	96	55x55	11x11	4	relu
Max Pooling	96	27x27	3x3	2	-
Conv	256	27x27	5x5	1	relu
Max Pooling	256	13x13	3x3	2	-
Conv	384	13x13	3x3	1	relu
Conv	384	13x13	3x3	1	relu
Conv	256	13x13	3x3	1	relu
Max Pooling	256	6x6	3x3	2	-
FC	-	4096	-	-	relu
FC	-	4096	-	-	relu
FC	-	1000	-	-	softmax

– VGG16:

Layer	Features	size	Kernel size	Strides	Activation
Imagen	1	224x224x3	-	-	-
2 x Conv	64	224x224	3x3	1	relu
Max Pooling	64	112x112	3x3	2	-
2 x Conv	128	112x112	3x3	1	relu
Max Pooling	128	56x56	3x3	2	-
3 x Conv	256	56x56	3x3	1	relu
Max Pooling	256	28x28	3x3	2	-
3 x Conv	512	28x28	3x3	1	relu
Max Pooling	512	14x14	3x3	2	-
3 x Conv	512	14x14	3x3	1	relu
Max Pooling	512	7x7	3x3	2	-
FC	-	25088	-	-	relu
FC	-	4096	-	-	relu
FC	-	4096	-	-	relu
FC	-	1000	-	-	softmax