Apuntes Semana 8, Clase

Gerardo Alberto Gómez Brenes - 2022089271

***Resumen*—Resumen compacto y organizado de la clase. Incluye: motivación breve, pautas para la entrega, conceptos clave de redes neuronales y las fórmulas que el profesor mencionó o utilizó como referencia.**

I. Motivación y notas generales

Se compartió un ejemplo de robótica donde un modelo adapta el comportamiento del robot ante la pérdida o modificación de una pata. Esto ilustra la capacidad de adaptación (aprendizaje por refuerzo y transferencia) y su potencial en aplicaciones como prótesis. Mensaje práctico: hay áreas de ML que no son solo modelos de lenguaje; robótica y manipulación son opciones reales.

Se enfatizó también la forma correcta de entregar tareas: documentos cortos y autocontenidos, con figuras y tablas dentro del texto, y máximo unas pocas páginas para esta actividad (usar el grupo para dudas).

II. Pautas para la entrega

El trabajo debe ser claro y compacto. Incluir dentro del documento:

- Resultados relevantes (figuras, tablas) y su interpretación breve.
- Referencias a notebooks o repositorios solo como complemento, no como sustituto.
- Selección crítica de gráficos: mostrar los que aporten a la conclusión.

III. Entrada y primer modelo: regresión logística

Imágenes 28×28 se representan como vectores de 784 píxeles. La regresión logística usa la transformación lineal seguida de la sigmoide:

$$z = w^T x + b, \qquad \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Para clasificación binaria (ej.: "¿es 5 o no?") $\sigma(z)$ da una probabilidad entre 0 y 1.

IV. De binario a multiclase y notación matricial

Para 10 clases se puede entrenar una regresión por clase o usar una salida vectorial. Notación común:

$$Z = XW^T + \mathbf{b},$$

donde, por ejemplo, W puede tener forma 10×784 (10 neuronas de salida y 784 entradas). Con un *batch* de tamaño B la entrada X es $B \times 784$ y el resultado Z es $B \times 10$.

Ejemplo numérico: con 10 salidas y 784 entradas hay $10 \times 784 = 7\,840$ parámetros sólo en esa capa.

V. One-hot, softmax y decisión

Las etiquetas multi-clase se representan como vectores *one-hot*. Para obtener una distribución de probabilidad sobre clases se usa *softmax*:

$$\mathrm{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

La predicción final corresponde al índice con mayor probabilidad.

VI. Arquitectura: capas y conexiones

Una red densa (fully connected / dense) conecta todas las salidas de una capa con todas las entradas de la siguiente. Añadir capas y activaciones no lineales permite resolver relaciones no lineales que un perceptrón simple no puede (ejemplo clásico: XOR).

VII. Funciones de activación y gradientes

Funciones mencionadas en clase:

- Sigmoide: $\sigma(z) = 1/(1 + e^{-z})$. Derivada: $\sigma'(z) = \sigma(z)\,(1 - \sigma(z))$. Tiene problemas de *vanishing gradient* en extremos.
- ReLU: $\mathrm{ReLU}(z) = \max(0, z)$. Es eficiente, pero puede generar neuronas "muertas" cuando la derivada es cero.
- Leaky ReLU: variante con pequeña pendiente negativa para evitar neuronas muertas.
- $\tanh(z)$: acotada en $(-1, 1)$, útil en algunos contextos.

VIII. Forward, pérdida y retropropagación

El *forward* calcula salidas capa a capa. Con una función de pérdida L se aplica retropropagación para obtener derivadas parciales $\partial L/\partial w$ y actualizar parámetros. Regla de actualización (descenso de gradiente):

$$w \leftarrow w - \eta\, \frac{\partial L}{\partial w},$$

donde η es la tasa de aprendizaje. Para el perceptrón se mencionó el *hinge loss*:

$$L_{\text{hinge}} = \max(0,\, 1 - y\,(w^T x + b)).$$

La retropropagación usa la regla de la cadena para propagar sensibilidades hacia atrás; por eso es necesario que las capas sean diferenciables.

IX. Costos computacionales y dimensionalidad

Aumentar neuronas y capas incrementa parámetros y costo de optimización. La *maldición de la dimensionalidad* complica la búsqueda de soluciones óptimas. Ejemplo: si una capa tiene 256 neuronas y la siguiente tiene 10, los pesos entre ellas son $10 \times 256 = 2\,560$.

Recomendación práctica: reducir dimensiones innecesarias (filtrado de features, PCA) cuando sea posible.

X. Notas sobre representaciones y CNN/embeddings

Para imágenes, las CNN aplican kernels que extraen patrones locales (bordes, texturas, formas). En lenguaje, *embeddings* condensan palabras o frases en vectores de dimensión fija; la similitud semántica se mide por distancia en ese espacio vectorial.