18 nov 2025

Análisis Matemático (5/8) - a241b25

Invitados merrazquin@fi.uba.ar Análisis Matemático - CEIA 24Co2025

Archivos adjuntos Análisis Matemático (5/8) - a241b25

Registros de la reunión Transcripción Grabación

Resumen

lse posgrados condujo la clase, comenzando con una discusión sobre la formación de grupos y la dificultad del material de la Clase 5 reportada por Federico G. Zacchigna, Gabriel Quiroga y Lucia T. Capon Paul. Se revisaron conceptos fundamentales de funciones y campos multivariados, destacando la importancia de los campos escalares para la optimización y definiendo el gradiente como el vector de derivadas parciales que apunta a la dirección de máxima variación, lo cual fue complementado por las consultas de Mariano Jauffroy, Gabriel Quiroga y Alejandro Valle sobre sus propiedades. Finalmente, lse posgrados introdujo la diferenciación automática y el concepto de *back propagation* mediante la implementación de grafos de cómputo, explicando cómo el *forward* y el *backward* pasan valores y derivadas a través de operaciones base, y discutiendo con Gabriela Sol Salazar y Federico G. Zacchigna sobre la implementación de derivadas analíticas y el soporte para derivadas de segundo orden.

Detalles

- **Discusión de Tareas y Revisión de Videos Pregrabados** lse posgrados inició la reunión consultando sobre el progreso del Trabajo Práctico (TP) y la formación de grupos. Federico G. Zacchigna mencionó que no ha formado un grupo, ya que no está seguro de poder dedicar suficiente tiempo al TP. lse posgrados preguntó sobre la experiencia de ver los videos para la clase, ya

que era la clase "más cargada", a lo que Gabriel Quiroga confirmó que fue una clase "pesada". Lucia T. Capon Paul compartió que tuvo poco tiempo para ver los videos y se sintió desorientada, descubriendo más tarde que había visto los videos teóricos de *back propagation* en lugar de los pregrabados específicos para la Clase 5, los cuales estaban en la carpeta de grabaciones (00:00:00).

- **Conceptos de Funciones y Campos Multivariados** lse posgrados revisó la nomenclatura de las funciones multivariadas, distinguiendo entre funciones (entrada escalar) y campos (entrada vectorial). También se explicó la diferencia entre salida escalar y salida vectorial. Se destacó que las funciones escalares (como las de la secundaria) y las vectoriales (como las parametrizaciones de curvas) (00:07:15) contrastan con los campos escalares, que tienen múltiples entradas y una sola salida (considerados "los reyes" porque son optimizables) (00:08:44).
- **Optimización y Campos Escalares** lse posgrados enfatizó que para cualquier tarea de optimización, se debe llegar a un campo escalar, ya que los vectores no pueden ordenarse. Esta limitación implica que la salida debe ser un escalar para que el proceso sea optimizable, siendo el campo escalar la forma más genérica para la optimización (00:09:49).
- **Conjuntos de Nivel y Visualización** Se definió el conjunto de nivel como el conjunto de valores de entrada para los cuales la salida de una función tiene un valor específico, comparable al concepto de raíces en funciones escalares. Para los campos escalares de R2 en R, los conjuntos de nivel permiten visualizar la función en un plano bidimensional (X e Y) utilizando colores para representar la salida (Z), similar a los mapas físicos de altura (00:11:06).
- **Derivadas y Direcciones en Campos Escalares** lse posgrados explicó que el concepto de derivada en funciones escalares se relaciona con la tasa de variación y la pendiente de la recta tangente (00:12:24). La complicación en campos escalares de R^n a R es que existen infinitas direcciones posibles para moverse desde un punto, a diferencia del movimiento unidimensional en el eje X de las funciones escalares (00:13:58).
- **Derivadas Parciales y Gradiente** Para manejar las múltiples direcciones en campos escalares, se introdujo el concepto de derivadas parciales, donde se asume que solo una variable de entrada se mueve (se le suma h) mientras las demás se mantienen constantes (00:16:46). Mariano Jauffroy confirmó que esto se asemeja a examinar el movimiento de la función en la dirección de una variable de entrada específica (00:18:08). El gradiente empaqueta

todas estas n derivadas parciales en un vector de R^n que, de manera crucial, vive en el mismo espacio que las entradas (00:20:50).

- **Interpretación del Gradiente** Ise posgrados explicó que el vector gradiente apunta en la dirección de máxima variación local (crecimiento) de la función (00:22:19). Gabriel Quiroga preguntó si el gradiente está definido para un solo punto o para toda la superficie; Ise posgrados aclaró que la expresión genérica del gradiente es válida para todos los puntos donde la función es diferenciable, pero debe evaluarse en un punto específico para obtener un vector (00:23:48). Alejandro Valle preguntó sobre la dirección de mínimo crecimiento, a lo que Ise posgrados respondió que simplemente es el negativo del gradiente (00:25:19).
- **Generalización de Derivadas (Jacobiano y Hessiano)** Para campos vectoriales (R^n a R^m), la derivada de orden uno es el Jacobiano, una matriz de derivadas parciales de $m \times n$ o $n \times m$, dependiendo de la convención (00:26:54). Ise posgrados mencionó que para las derivadas de orden dos se limita el estudio a campos escalares, resultando en la matriz Hessiana, una matriz de derivadas parciales segundas. Se evita ir más allá de derivadas de orden dos (que resultarían en tensores de rango mayor a 2) por razones de complejidad y porque las herramientas comunes de optimización usan como máximo el Hessiano (00:29:33) (00:32:26).
- **Tensores en Deep Learning** Alejandro Valle preguntó sobre el uso de tensores en *deep learning* (00:29:33). Ise posgrados afirmó que la barrera de aprendizaje de la matemática tensorial es alta y que, si bien se utilizan tensores (a menudo de rango 3), en la práctica del TP estos se manejan como "matrices apiladas" para computar cálculos en paralelo, sin requerir álgebra tensorial avanzada (00:30:51).
- **Polinomio de Taylor y Optimización** Ise posgrados introdujo la generalización del polinomio de Taylor hasta orden dos para campos escalares, destacando su utilidad para modelar funciones localmente, incluso si la expresión analítica es desconocida (00:33:46). Esta aproximación local utiliza la función y sus derivadas (incluido el Hessiano) evaluadas en un punto para formar una aproximación polinomial. Se señaló que esta técnica es fundamental para desarrollar algoritmos de optimización (00:36:43).
- **Diferenciación Automática y Regla de la Cadena** La diferenciación automática se presentó como una técnica para obtener derivadas de funciones compuestas de manera eficiente y flexible (00:38:02). Mediante el uso de la regla de la cadena, se pueden combinar el gradiente de la función de salida y el Jacobiano de las funciones intermedias para calcular el gradiente

de la función compuesta (00:41:53). Esta técnica permite "encadenar" las derivadas de cada parte para obtener la derivada del todo (00:43:18).

- **Eficiencia y Grafos de Cómputo** lse posgrados contrastó el cálculo analítico puro (máxima eficiencia, mínima flexibilidad) con la aproximación numérica (mínima eficiencia, máxima flexibilidad) (00:48:09) (00:51:04). El enfoque intermedio es la diferenciación automática, que utiliza un grafo de cómputo para modelar la función compuesta como bloques de construcción (*bloques de Lego*), donde cada bloque tiene una función y su derivada analítica. Esto evita que el programador deba derivar la función completa analíticamente (00:52:16) (00:55:03).
- **Estructura del Grafo de Cómputo** El grafo de cómputo es un Grafo Acíclico Dirigido (DAG) que representa las dependencias entre variables (entradas y salidas de cada función) (00:53:51). Se requiere que cada nodo sepa calcular su valor y su gradiente en función de sus entradas (00:55:03). Federico G. Zacchigna preguntó si esto asume un campo escalar (una sola salida), a lo que lse posgrados respondió que, si bien puede extenderse a campos vectoriales, el interés principal en este contexto es la función de costo, que siempre es un campo escalar (00:56:15).
- **Derivación y Gráficos de Cómputo** lse posgrados explicó que la flexibilidad de la derivación se basa en que cada bloque puede calcular su propia derivada y las derivadas de los bloques se unen mediante un producto matricial (00:58:55). Este proceso se realiza con una pasada hacia adelante para obtener valores y luego una pasada hacia atrás para calcular la derivada de J con respecto a cada entrada (01:00:21). El requisito fundamental es que cada bloque sea "diferenciable de punta a punta" (end to end), lo que significa que debe saber calcular tanto su pasada hacia adelante (evaluar la función) como su pasada hacia atrás (la derivada) (01:01:42).
- **Implementación de Operaciones Base** lse posgrados presentó ejemplos simplificados de código para una clase de "operación base" que requiere definir los métodos *forward* y *backwards* (01:01:42). Por ejemplo, la clase *producto* devuelve el producto de dos valores y su gradiente es IX, mientras que una función escalar devuelve el valor y se piensa como un campo escalar de dimensión uno (01:02:52). Se ilustró cómo construir un grafo de cómputo para una expresión matemática más compleja ($x^2 \cdot e^{x + \cos(y)}$) usando operaciones elementales sucesivas (01:04:02).
- **Proceso de Forward y Backward en el Grafo** La pasada hacia adelante (*forward*) calcula los valores de las variables intermedias, lo cual es crucial porque los valores de entrada de una operación son necesarios para evaluar

su derivada en la pasada hacia atrás (*backward*). El proceso de derivación comienza en la última variable (Z en el ejemplo), estableciendo su derivada respecto a sí misma como 1, y luego encadenando las derivadas hacia atrás (01:05:14). Ise posgrados mencionó que se omite el numerador (derivada de Z) ya que siempre es el mismo en este contexto, y al encontrar una bifurcación se debe sumar sobre todos los caminos (01:06:31).

- **Consultas sobre Diferenciación** Gabriela Sol Salazar consultó si se usan funciones analíticas o numéricas para las derivadas, a lo que Ise posgrados respondió que se usan funciones analíticas, que no se vuelven más complicadas que descomponer una función compleja en partes simples analíticamente resolubles y usar la regla de la cadena para orquestarlas (01:08:13). Federico G. Zacchigna preguntó sobre el soporte para derivadas de segundo orden en *frameworks* como PyTorch o Jax, a lo que Ise posgrados indicó que lo duda debido a la complejidad de la dimensión, aunque existen métodos de segundo orden "truchos" o pseudo-segundo orden (01:21:00).
- **Introducción a Back Propagation** Back propagation fue identificado como el algoritmo crucial que permite el entrenamiento eficiente de redes neuronales, que en sí mismas son funciones compuestas que deben ser diferenciables de punta a punta (01:23:57). Ise posgrados explicó que durante el entrenamiento, se busca encontrar los valores de los parámetros del modelo (como A y B en una regresión) al pasar valores conocidos de entrada (X) y salida esperada (Y), convirtiendo a los parámetros en las incógnitas (01:25:37). La función de pérdida (J o L) compara la predicción con la salida esperada y debe ser derivable respecto a la predicción (01:28:13).
- **Back Propagation y Redes Neuronales** Back propagation utiliza el esquema del grafo de cómputo para obtener las derivadas del error respecto a cada parámetro del modelo (01:30:28). La arquitectura más simple es el Perceptrón Multicapa (MLP), una concatenación de capas lineales seguidas de funciones de activación. Ise posgrados explicó que la capa lineal es una transformación afín y proporcionó las expresiones matriciales para las derivadas de J respecto a los parámetros (W y B) y la entrada (X) para encadenar hacia atrás (01:31:58).
- **Uso de Funciones de Activación** El propósito de la función de activación es introducir no linealidad, ya que la composición de funciones lineales resulta en una única función lineal, lo que limitaría la capacidad del modelo (01:34:45). La función de activación típicamente es una función escalar no lineal, fácil de derivar y aplicar elemento a elemento, resultando en que el Jacobiano sea una matriz diagonal (01:35:58). La función de activación ReLU

(máximo entre 0 y x) es un ejemplo común, destacando que su derivada es barata de calcular (01:37:21).

- **Dudas sobre Funciones de Activación y Arquitectura** Gabriel Quiroga expresó su confusión sobre el uso de ReLU, ya que en gran parte es lineal, a lo que lse posgrados aclaró que mientras la función no sea lineal se rompe la propiedad de que la composición de funciones lineales es lineal (01:41:16). Respecto a la no derivabilidad en el origen de ReLU, lse posgrados mencionó que se resuelve de forma pragmática, asignando una de las derivadas o considerando que numéricamente es improbable que se dé exactamente el valor de cero (01:42:32). Federico G. Zacchigna preguntó sobre capas famosas que no sean lineales más activación, y lse posgrados citó las capas de atención (*attention blocks*) (01:40:00).
- **Implementación Explícita de Grafos de Cómputo** lse posgrados introdujo una implementación de grafos de cómputo en código (Colab), aclarando que es un grafo explícito (como el enfoque anterior de TensorFlow) para una mejor visualización, a diferencia del enfoque de grafos implícitos de PyTorch (01:45:28). El grafo de cómputo se modela como una clase orquestadora que usa un diccionario para almacenar operaciones (01:48:37). Se destacó que Python mantiene el orden de inserción en los diccionarios, lo que permite recorrer el grafo correctamente para el cálculo *forward* (01:50:47).
- **Mecánica del Forward y Backward en Código** El método *forward* inicializa los valores y calcula el valor de cada variable; si no es una entrada, se calcula el *forward* de la operación asociada (01:52:03). El método *backwards* no recibe valores de entrada, asumiendo que ya se ejecutó el *forward*, y recorre el grafo a la inversa. La derivada de la variable de salida (output) se establece en 1 (01:53:11). La contribución a las derivadas acumuladas se realiza pasando los aportes de los padres a los hijos para evitar la necesidad de sumar sobre todas las bifurcaciones explícitamente (01:54:18).
- **Definición de Clases de Operación** La implementación incluye la definición de clases de operación que heredan de una clase base y definen sus propios métodos *forward* y *backwards* (01:55:32). Como ejemplos, la clase *Producto* define su *forward* como `a * b` y su *backwards* como `(b, a)`, mientras que funciones escalares como el coseno o elevar al cuadrado también definen sus correspondientes *forward* y *backwards* con sus derivadas (01:56:52).
- **Derivadas Analíticas y Gradiente** lse posgrados explicó que las derivadas analíticas se aplican a bloques pequeños, y que el gradiente para un exponente es más complejo que para la base. Para la base (x a la k), la derivada se trata como si el exponente fuera una constante ($k \cdot

x^{k-1}$), mientras que para el exponente (k a la x), la derivada es k^x por el logaritmo natural de K (01:58:03).

- **Implementación del Cálculo de Gradiente** lse posgrados detalló la implementación del cálculo de gradiente utilizando un lenguaje que define entradas, un grafo de cómputo con modo de depuración activado, y operaciones explícitas como coseno (I es coseno de X) y potencia (Z es X a la I) (01:59:15). El proceso se ilustra con el ejemplo de $x^{\cos(x)}$ evaluado en $x=3$, cuya derivada da -0.1634 (01:58:03).
- **Ejecución Forward y Backward** El proceso de cálculo se demostró con un ejemplo donde el pase "forward" calcula los valores de X, I y Z secuencialmente, sirviendo el último valor calculado (Z) como salida, ya que el modo de depuración está activo (01:59:15). Posteriormente, la función "backwards" calcula la derivada de Z respecto a X, incluyendo las contribuciones de Z a sí misma, I, y I a X (02:00:20).
- **Ejemplo Más Complejo y Visualización** lse posgrados presentó un ejemplo más complejo, $x^{\cos(y) \cdot z^2}$, donde se desglosó la función en operaciones intermedias (A, B, C) y se definió el grafo de cómputo con múltiples entradas (X, Y, Z) (02:00:20). Se mostró cómo el uso de la librería 'grafis' permite visualizar el grafo de cómputo, tanto con los *inputs* y operaciones hasta la salida, como en un formato más "humano" que lista las operaciones secuencialmente o de izquierda a derecha (02:01:39).
- **Naturaleza del Código y la Programación Didáctica** lse posgrados señaló que, aunque la ejecución parece "magia", es la aplicación de la matemática vista previamente y que el código implementado no es eficiente, sino didáctico, pero la lógica subyacente es la misma que en implementaciones eficientes (02:01:39). Se afirmó que la parte difícil del trabajo la realizan las operaciones individuales, y que el programa solo se encarga de multiplicar y sumar, demostrando que la lógica de la cuenta programada funciona (02:02:46).

Pasos siguientes recomendados

- [ ] Cesar Orellana mirará los videos de la clase seis para entender el ajuste de los pesos con las derivadas.