2 dic 2025

Análisis Matemático (7/8) - a241b25

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

Archivos adjuntos Análisis Matemático (7/8) - a241b25

Registros de la reunión Transcripción Grabación

Resumen

lse posgrados, Gabriel Quiroga, Alejandro Valle y Pa. Santiago Rodriguez Castro discutieron el contenido de la semana, centrándose en el concepto de Medias Móviles Exponenciales (EMA) como una herramienta computacionalmente eficiente y la lógica detrás de su ponderación y reescalado para evitar problemas numéricos. lse posgrados detalló la aplicación de EMA en la optimización de *gradient descent* a través de Momentum y RMSprop, culminando en la descripción de Adam como el estándar de la industria y la comparación con métodos de segundo orden como Newton y BFGS, destacando que la elección del optimizador depende de la problemática específica. La discusión también incluyó una revisión práctica de la implementación de un Perceptrón Multicapa (MLP) en Python usando solo NumPy y el análisis de optimizadores de vanguardia, como AdamW, una corrección para la regularización L2 en Adam, y Muon, que utiliza *momentum* con ortogonalización para una mayor eficiencia en el entrenamiento de Large Language Models (LLMs).

Detalles

Longitud de las notas: Estándar

- **Discusión sobre el Contenido de la Semana** lse posgrados inició la discusión preguntando a los participantes sobre el material de video de la semana. Gabriel Quiroga mencionó que los videos fueron "sencillos" y fáciles de

entender. El grupo reconoció que esta era la penúltima clase y que el material era una continuación de la clase anterior (00:00:00).

- **Medias Móviles Exponenciales (EMA)** Ise posgrados explicó que las EMA son una herramienta matemática y un artificio computacionalmente "superficiente" y "baratísimo" (00:04:35). La lógica es realizar un promedio ponderado donde la ponderación no es uniforme, lo cual soluciona el problema de tener que guardar todos los valores en una ventana para un promedio uniforme; con EMA solo se guarda el último valor estimado, manteniendo un consumo de memoria constante (00:06:07). La fórmula de actualización es una combinación lineal del valor anterior y la nueva medición, suavizando la sucesión y dándole más peso a los valores recientes (00:07:16).
- **Consideraciones sobre la Ponderación en EMA** Ise posgrados explicó que, técnicamente, la suma de las ponderaciones de las EMA no da uno, ya que está ponderada por $1 - \lambda$ (00:09:37). Esto se nota especialmente cuando λ es alto y n (el número de mediciones) es pequeño, lo que puede "achicar" los valores de manera significativa (00:10:49). La solución a este problema es reescalar el resultado dividiendo por $1 - \lambda^n$. Gabriel Quiroga consultó si para n muy grande el resultado se alejaría de uno, y Ise posgrados aclaró que, dado que λ está entre 0 y 1, cuanto más grande sea n, más se acerca a cero la resta, y por lo tanto, el resultado se acerca a uno (00:12:04).
- **Aclaración sobre la Eficiencia Computacional de EMA** Alejandro Valle expresó su preocupación de que, al solo guardar el último valor estimado, se pudiera perder información (00:14:36). Ise posgrados explicó que, aunque la formulación es computacionalmente eficiente porque solo requiere guardar el valor anterior en memoria, matemáticamente el término enésimo tiene en cuenta todos los valores anteriores, por lo que no se pierde información (00:16:06).
- **Aplicación de EMA en Optimización: Momentum y RMSprop** Ise posgrados introdujo la aplicación de las medias exponenciales en *gradient descent* a través de Momentum y RMSprop (00:17:38). Momentum aplica la media exponencial sobre los gradientes para suavizar el avance, amplificando la consistencia en una dirección y anulando la inconsistencia, lo que ayuda con el ruido (00:19:08). RMSprop trabaja sobre la variabilidad o escala de los gradientes, aplicando una media exponencial a los elementos al cuadrado de cada componente del gradiente para adaptar el *learning rate* (00:20:15). Esto es útil para transformar superficies de curvas de nivel estiradas en algo más circular, acelerando la convergencia (00:21:40).

- **Adam: Combinación de Momentum y RMSprop** Ise posgrados describió a Adam como una combinación de Momentum y RMSprop, considerado un "estándar de la industria" y muy robusto con valores por defecto para sus hiperparámetros β_1 y β_2 (00:22:57). Adam utiliza las dos medias exponenciales y aplica el reescalado (dividiendo por $1 - \lambda^n$) debido a que usa valores de λ muy cercanos a uno, lo que evita que se "malgasten" las primeras iteraciones (00:24:06).
- **Comparación de Adam con Gradient Descent Clásico** Pa. Santiago Rodriguez Castro preguntó si había alguna razón para usar *gradient descent* clásico en lugar de Adam. Ise posgrados respondió que, aunque Adam es la norma general y ofrece un "modelo base" razonable por defecto, existen casos particulares en "visión" donde se ha observado que un *gradient descent* con un *learning rate* bien ajustado da mejores resultados (00:25:26).
- **Métodos de Segundo Orden: Newton y BFGS** Ise posgrados explicó que los métodos de segundo orden, como el método de Newton, utilizan el Hessiano para medir la curvatura de la superficie (00:28:27). El método de Newton tiene convergencia local cuadrática, pero es muy costoso de estimar en la práctica, ya que requiere muchas más observaciones que el gradiente. El método BFGS es una aproximación que se plantea para ser más barato e iterable, logrando una convergencia semi-cuadrática (00:31:34). Ise posgrados destacó que la implementación del BFGS en librerías como SciPy utiliza la matriz identidad para arrancar la estimación si no se proporciona un valor inicial (00:34:00).
- **BFGS de Memoria Limitada (LBFGS)** Ise posgrados abordó el problema de escala de BFGS, donde el consumo de memoria es cuadrático ($N \times N$) con el tamaño de los parámetros. LBFGS (Limited-memory BFGS) aproxima la matriz H_t basándose en una ventana de tiempo hacia atrás, guardando una cantidad M de pares de diferencias de parámetros y gradientes (00:35:13). Esto reduce la complejidad de la memoria a $O(M \times N)$, resultando en un método más eficiente computacionalmente, aunque con una peor aproximación (00:36:35). Gabriel Quiroga preguntó sobre la necesidad de guardar los parámetros en LBFGS, a lo que Ise posgrados confirmó que en LBFGS se guardan los pares de diferencias porque no se guarda H_t, y esto resulta más eficiente (00:45:18).
- **Dependencia de la Elección del Optimizador** Ise posgrados enfatizó que la elección del método de optimización ("primer orden" o "segundo orden") "depende" del problema. Los métodos de segundo orden son más estables y, por lo general, requieren entrenamiento en *batch* (00:38:19). Por ejemplo, en modelos de árboles potenciados (GBDT), el cuello de botella es el entrenamiento del árbol, por lo que usar métodos de segundo orden resulta

"básicamente gratis" en términos de velocidad (00:41:10). En cambio, para modelos o *datasets* muy grandes, como las redes neuronales, donde no se puede entrenar en *batch* y se requiere velocidad, se suelen utilizar métodos de primer orden como Adam (00:43:56).

- **Revisión del Colab de Multilayer Perceptron (MLP)** Ise posgrados inició la discusión con el colab del Perceptrón Multicapa (MLP) (00:49:03), que ilustra un ejemplo de implementación de una red neuronal utilizando solo NumPy, con una API pseudo-PyTorch. El objetivo de esto era mostrar la implementación de operaciones base como `linear` y `relu`, enfocándose en las funciones de *forward* y *backward*, y la gestión de gradientes. En la clase `linear`, se define la inicialización de pesos con una distribución normal y se explica el proceso de guardar entradas y salidas para su uso en el paso *backward* (01:02:01).
- **Detalles de Implementación del Colab** Se detallaron las operaciones clave, como la definición de los gradientes para los pesos y el sesgo (`W` y `B`) en la capa `linear`, utilizando la retropropagación de la derivada de la función de pérdida con respecto a la salida (`DZ`). Ise posgrados destacó que la función de activación ReLU es popular debido a la simplicidad y bajo costo computacional de su cálculo *forward* y *backward*, que implica establecer la derivada en 1 o 0 dependiendo de si el valor de entrada era positivo o no (01:04:55). Además, se explicó la estructura de la clase `MLP` para el grafo de cómputo secuencial, incluyendo el proceso de *forward*, la puesta a cero de los gradientes, y el *backward* que comienza con la derivada proporcionada por la función de pérdida (01:07:40).
- **Gradient Descent y Estabilidad Numérica** Ise posgrados describió la implementación del paso de *gradient descent* (descenso de gradiente) en el MLP, que actualiza los parámetros restando el *learning rate* multiplicado por el gradiente (01:08:52). Se presentó un ejemplo de red (`lineal` 5-4, `relu`, `lineal` 4-2) para ilustrar cómo el *forward* y el *backward* resultan en pequeños cambios en los parámetros después de un paso de entrenamiento (01:10:07). También se introdujo el concepto de la estabilidad numérica, ejemplificado por la técnica `logsumexp` utilizada en el cálculo de la función de pérdida *cross-entropy*, para evitar *overflow* numérico que puede ocurrir con exponenciales de números grandes (01:11:32).
- **Entrenamiento y Resultados del Colab** Se revisó la función de *loss* (pérdida) de *cross-entropy*, señalando que utiliza `logsumexp` como truco de estabilidad. El código de entrenamiento, que utiliza un dataset, muestra cómo se calcula el costo en *train* y *test* a lo largo de las épocas, lo que indica que el modelo está aprendiendo, aunque un poco lento (01:13:47) (01:16:18). Ise

posgrados enfatizó que este código demostraba la aplicación práctica de las cuentas vistas en clases anteriores, incluyendo el uso de un *batch size* de uno y el cálculo de la predicción, la pérdida y el *backward* (01:15:04).

- **Arquitectura de Redes Neuronales de Moda: LLMs y Optimizadores** Ise posgrados pasó a discutir la arquitectura de redes neuronales de moda, destacando que los Large Language Models (LLMs) son omnipresentes, aunque a menudo están sobrevalorados y son difíciles de entrenar (01:17:52). Se mencionó la dificultad de obtener información sobre las últimas innovaciones de laboratorios líderes, como OpenAI, lo que obliga a la gente a inferir sus métodos de entrenamiento (01:19:46). La discusión se centró en los optimizadores de vanguardia, específicamente AdamW, que se usa en la mayoría de los modelos de 2024, incluyendo data de GPT 3.5 y GPT-4 (01:21:36).
- **AdamW: Corrección de Adam y Regularización L2** Ise posgrados explicó que Adam se comporta mal cuando se le agrega la regularización L2 (similar a *Ridge regression*), ya que el término de penalización interfiere con la estimación de la media exponencial al cuadrado del gradiente (01:21:36). La solución, conocida como AdamW, es simple: se elimina el término de regularización L2 del cálculo del gradiente y se aplica directamente a la actualización de los pesos durante el paso de descenso de gradiente. Este cambio permite que el término de corrección se aplique como un *bypass* de los demás cálculos de Adam, y se considera un algoritmo de vanguardia por su simplicidad y eficacia (01:24:18).
- **Muon: Momentum con Ortogonalización** Ise posgrados introdujo Muon, un optimizador avanzado que usa *momentum* con ortogonalización y es considerado un método pseudo-segundo orden porque ajusta la curvatura del gradiente (01:25:58). Este optimizador es específico para los parámetros matriciales (como la matriz W en una capa lineal, pero no para el sesgo B) en LLMs que presentan problemas de estabilidad. La motivación detrás de Muon fue la observación de que las matrices de gradiente en LLMs tienen un rango efectivo muy bajo, con pocos valores singulares grandes y muchos muy pequeños, lo que resultaba en una actualización ineficiente del parámetro (01:27:37).
- **Implementación y Ventajas de Muon** Muon busca compensar las direcciones de valores singulares más bajos al planchar la matriz de valores singulares (Σ), aproximándola a la identidad, a través de un algoritmo numérico llamado Newton-Schuls (01:29:05). Esto se hace para que, efectivamente, se aumente el *learning rate* en las direcciones con menos información (01:30:24). Ise posgrados enfatizó que Muon ha sido probado por laboratorios

para entrenar modelos grandes y que funciona bien (01:31:46). Además, se señaló la ventaja en el consumo de memoria: mientras que Adam multiplica el consumo de memoria por cuatro debido a la necesidad de almacenar parámetros, gradientes y dos medias exponenciales (01:33:15), Muon lo reduce a un factor de tres, lo que resulta en una reducción de costos de entrenamiento (01:34:35).

- **Validación y Relevancia de Muon** Muon vincula conceptos de álgebra lineal (DBS y valores singulares) con optimizadores y se ha utilizado para entrenar modelos de gran escala, como los modelos Kimi (01:36:18). Aunque no se sabe si laboratorios como Open AI lo están utilizando, su validación para modelos de 10^{12} parámetros lo establece como un método de vanguardia (01:37:38). Ise posgrados concluyó la clase recordando a los participantes la encuesta de clase y la importancia de completar autoevaluaciones y ver los videos para la próxima semana (01:39:05).

Pasos siguientes recomendados

No se encontraron próximos pasos sugeridos para esta reunión.