Policy Gradients utilizando CUDA aplicado al juego Breakout de ATARI

Amaury Gutiérrez Acosta, Ixchel G. Meza Chávez

May 2, 2017

Introducción

Uso de algoritmos de optimización numericos en el contexto de aplicaciones de machine learning.

Hay dos tipos de problemas de optimización que surgen en machine learning: - problemas de optimización convexa -derivados de el uso de regresión logística o support vector machines - problemas no convexos y altamente no lineales -derivvados del uso de redes neuronales DNN.

En el contexto de machine learning a gran escala se ha tenido gran interés la propuesta de Robbins y Monro de algoritmos estocásticos, como el método de gradiente estocástico SG.

En DNN se describe una función de predicción h cuyo valor es calculado aplicando transformaciones sucesivas a un vector de entrada x. Estas transformaciones son capas o layers. Por ejemplo una capa canónica conectada completamente realiza un cálculo donde x es el vector de entrada, la matriz W y el vector b contienen los parámetros de la capa y s es una función de activación no lineal component-wise.

$$x^{(j)} = s(W_j x^{(j-1)} + b_j)$$

La función sigmoidea $s(x) = \frac{1}{(1+exp(-x))}$ y la función hinge $s(x) = max\{0, x\}$ conocida como ReLU son funciones populares para utilizarse como función de activación s.

El problema de optimización en este contexto involucra un training set $(x_1, y_1)...(x_n, y_n)$ y la elección de una función de pérdida l de tal forma que

$$min \frac{1}{n} \sum_{i=1}^{n} l(h(x_i; w), y_i)$$

Como ya se mencionó dicho problema es no convexo altamente no lineal, sin embargo se han claculado soluciones aproximadas por medio de métodos basados en el gradiente., pues el gradiente del objetivo en la ecuación anterior con respecto al vector parámetro w se puede calcular por medio de la regla de la cadena usando diferenciación algorítmica. Ésta técnia de diferenciación es conocida como *back propagation*.

Los problemas de optimización en macine learning surgen a través de la definición de las funciones de pérdida y predicción que aparecen en medidas de riesgo empírico y esperado que intentamos minimizar.

Métodos de optimización

Método estocástico de gradiente

En el contexto de minimizar el riesgo empírico R_n con $w_1 \in \mathbb{R}^d$ el método estocástico de gradiente SG se define como

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla f_{ik}(w_k)$$

donde para todo $k \in \mathbb{N} := \{1, 2, ...\}$, el índice i_k correspondiente a la seed como el par muestra (x_{ik}, y_{ik}) es escogido aleatoriamente de $\{1, ..., n\}$ y α_k es un paso positivo. Cada iteración de este método involucra sólo el cálculo del gradiente $\nabla f_{ik}(w_{ik})$ correspondiente a una muestra. A diferencia de un algoritmo de

optimización determinística, en éste método la secuencia de iteración no está determinada únicamente por la función R_n, el punto de inicio w_1 y el paso de la secuencia $\{\alpha_k\}$, sino que $\{\alpha_k\}$ es un proceso estocástico cuyo comportamiento está determinado por la secuencia aleatoria $\{i_k\}$. Aunque cada dirección $-\nabla f_{ik}(w_k)$ pueda no ser descendiente de w_k al producir una derivada direccional negativa para R_n de w_k, si es una dirección descendiente *en espera*, entonces la secuencia $\{w_k\}$ puede ser guiada hacia un mínimo de R_n.

Método batch de gradiente

El método más simple es el algoritmo *steepest descent* o *gradiente batch* definido como

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla R_n(w_k) = w_k - \frac{\alpha_k}{n} \sum_{i=1}^{n} \nabla f_i(w_k)$$

donde el cálculo del paso $-\alpha_k \nabla R_n(w_k)$ es más costoso que el del método estocástico, sin embargo se podría esperar un mejor paso cuando se consideran todas las muestras en la iteración.