Stochastic Gradient Method

Department of Statistics

2022, Nov

University of Seoul

Introduction

Optimization problem

- Objective function: $L : \mathbb{R}^p \mapsto \mathbb{R}$
- assume that L is differentiable and strictly convex.
- optimization problem is to obtain

$$w^* = \underset{w \in \mathbb{R}^p}{\text{argmin}}\ L(w)$$

Optimization problem

- solve $\nabla L(w) = 0$ where

$$\nabla L(w) = (\frac{\partial L(w)}{\partial w_1}, \cdots, \frac{\partial L(w)}{\partial w_p})^\top$$

  with $w = (w_1, \cdots, w_p)^\top$.

- When the solution of the equation does not have a closed form, the iterative algorithm is widely used.

Newton-Raphson algorithm: second order algorithm

Assume that $L(w)$ is twice differentiable, and denote the Hessian matrix of $L(w)$ by $H(w)$.

- Set $k = 0$, an initial $w^{(k)} \in \mathbb{R}^p$
- Repeat:
  - $w^{(k+1)} \leftarrow w^{(k)} - H(w^{(k)})^{-1} \nabla L(w^{(k)})$
  - $k \leftarrow k + 1$
  - If $w^{(k)}$ converges, stop the algorithm.
- Return $w^{(k)}$

contour plot

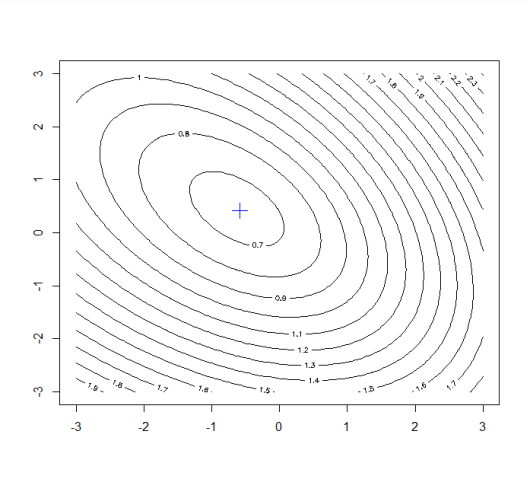

Figure 1: contour plot of an $L(w)$

set an initial

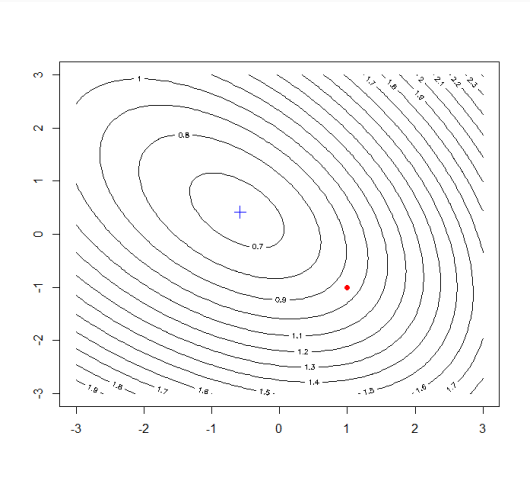

Figure 2: contour plot of an $L(w)$

Quadratic approximation and updating solution

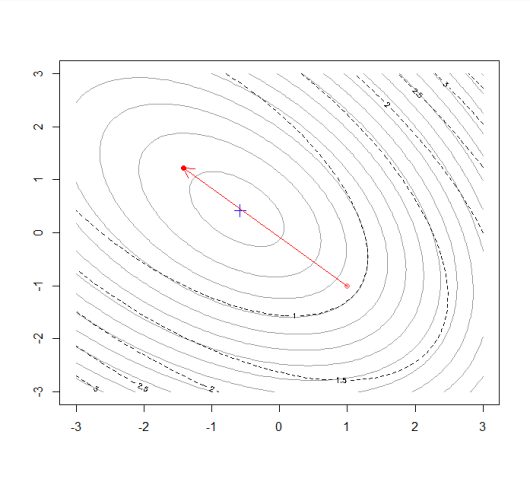

Figure 3: dashed curve is the contour of quadratic function approximated at the initial points

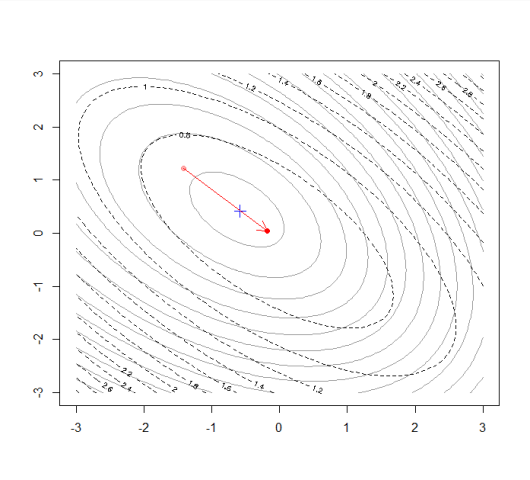

Figure 4: dashed curve is the contour of quadratic function approximated at the updated points

Newton-Raphson algorithm

- Set an initial;
- Approximate $L(w)$ at the current solution by quadratic function
- Update the current solution by minimizing the quadratic function

Newton-Raphson algorithm

- Pros:
  - efficient in the sense of a required number of iterations
- Cons:
  - numerically unstable;
  - computational cost is very high when p is large.

Gradient decent algorithm: first order method

- Set $k = 0$, an initial $w^{(k)} \in \mathbb{R}^p$ and $\eta > 0$
- Repeat:
  - $w^{(k+1)} \leftarrow w^{(k)} - \eta \nabla L(w^{(k)})$
  - $k \leftarrow k + 1$
  - If $w^{(k)}$ converges, stop the algorithm.
- Return $w^{(k)}$

Gradient vector

- Tangent space at $w^{(k)}$: $\mathcal{T}(w^{(k)}) = \{w \in \mathbb{R}^p : \nabla L(w^{(k)})^\top w = 0\}$
- Gradient vector at $w^{(k)}$ satisfies that

$$\nabla L(w^{(k)})^\top w = 0$$

  for $w \in \mathcal{T}(w^{(k)})$ by definition. (orthogonal to tangent vector)
- The gradient vector is the fastest direction to increase the value of the objective function at the point $w^{(k)}$.
- Move the opposite direction of the gradient vector at the current solution.

Gradient decent algorithm: first order method

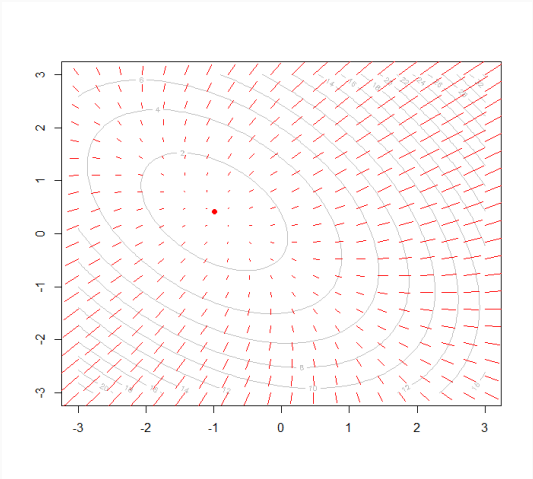

Figure 5: Contour and gradient vector of an $L(w)$: gray curve denotes the contour; the red point denotes w^*; the red lines denote the gradient vectors

Gradient decent algorithm: first order method

- Pros:
  - computational cost for each iteration is relatively low.
  - updating is computationally stable.
- Cons:
  - Required number of iterations for convergence is generally large.

Back-propagation algorithm is a gradient decent method for (deep) neural network model.

Stochastic gradient decent (SGD) method

- Generally $L(w)$ is an empirical risk function. For example, $L(w) = \frac{1}{n}\sum_{j=1}^{n} l(w; z_j)$, where z_j denotes the jth observation.
- Often, applying the GD to some problems is difficult when all z_js are not able to be loaded in the GPU or CPU memory.
  - Case 1: assume that n is very large such that computing $\nabla L(w)$ is intractable.
  - Case 2: assume that $L(w)$ is not fixed since z_j is observed on-line.
- Sampling or partitioning data from $\mathcal{D} = \{z_j : 1 \leq j \leq n\}$ and apply the gradient decent method.

SGD algorithm

- Set $k = 0$, an initial $w^{(k)} \in \mathbb{R}^p$ and $\eta > 0$
- Repeat:
  - Sample a set $B \subset \{1, \cdots, n\}$
  - Compute $\nabla L(w^{(k)}) = \sum_{j \in B} \nabla l(w^{(k)}, z_j)/|B|$
  - $w^{(k+1)} \leftarrow w^{(k)} + \eta(-\nabla L(w^{(k)}))$
  - $k \leftarrow k + 1$
  - If $w^{(k)}$ converges, stop the algorithm.
- Return $w^{(k)}$

SGD algorithm by (mini) batch

- Set $k = 0$, an initial $w^{(k)} \in \mathbb{R}^p$ and $\eta > 0$
- Set a partition of $\{1, \cdots, n\}$: $B_1, \cdots, B_m$
- Repeat:
  - Sample $m' \in \{1, \cdots, m\}$
  - Compute $\nabla L(w^{(k)}) = \sum_{j \in B_{m'}} \nabla l(w^{(k)}, z_j)/|B_{m'}|$
  - $w^{(k+1)} \leftarrow w^{(k)} + \eta(-\nabla L(w^{(k)}))$
  - $k \leftarrow k + 1$
  - If $w^{(k)}$ converges, stop the algorithm.
- Return $w^{(k)}$

Expectation of stochastic gradient

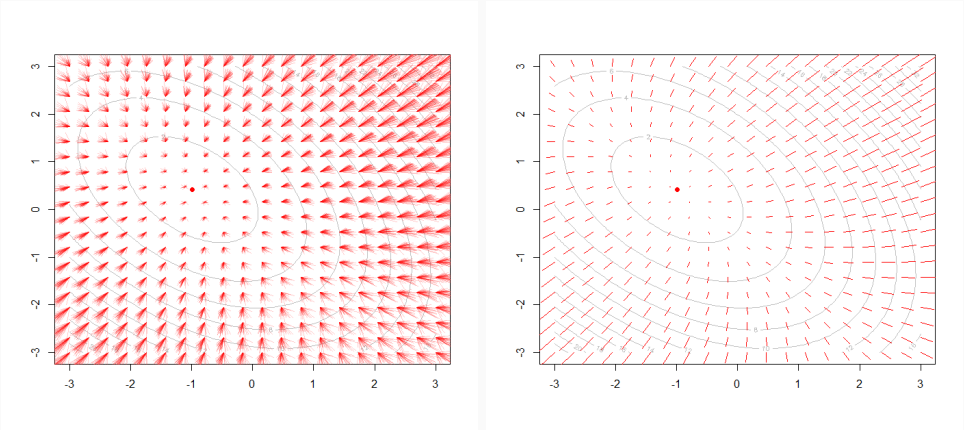

Figure 6: Left panel displays stochastic gradient vectors in batch samples; right panel displays expectation of the stochastic gradients

Since the stochastic gradient is proportional to unbiased estimator of the gradient vector of $L(w)$, we can expect the desired good property of the solution obtained by stochastic gradient method. The convergence will be shown in the last section.

Variants of SGD

Stochastic gradient descent with Moment

- Parameters: Learning rate η, batch size b, initial value of w, moment parameter α
- Input : Sample $z_1, \cdots, z_n$
- Algorithm
  1. set initial $w^{(1)}$, $\Delta w^{(0)} = 0$
  2. **for** $t = 1$ to iteration **do**
     S_t is a index set of b random samples
     $L_t(w) = \frac{1}{b} \sum_{i \in S_t} l(w, z_i)$
     $\Delta w^{(t)} = -\eta \nabla L_i(w^{(t)}) + \alpha \Delta w^{(t-1)}$
     $w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$
     end for
  3. Return w

($\Delta w^{(t)}$ is the updated vector at the tth step.)

For convenience, we ignore the index of batch in the algorithm. The solution of the GD algorithm at t,

$$w^{(t)} = w^{(1)} - \eta \sum_{k=1}^{t-1} \nabla L(w^{(k)})$$

While, the solution of the moment method is given by

$$w^{(t)} = w^{(1)} - \eta \sum_{k=1}^{t-1} \nabla L(w^{(k)}) - \eta \sum_{k=1}^{t-2} \sum_{s=1}^{t-k-1} \alpha^s \nabla L(w^{(k)}).$$

Equally,

$$w^{(t)} = w^{(1)} - \eta \sum_{k=1}^{t-1} \sum_{s=1}^{t-k} \alpha^{s-1} \nabla L(w^{(k)})$$

Adagrad

- Parameters: Learning rate η, batch size b, initial value of w
- Input : Sample $z_1, \cdots, z_n$
- Algorithm
  set $w^{(1)}, G^{(1)} = 0 \in \mathbb{R}^{p \times p}$, where p is dimension of parameter
  for $t = 1$ to iteration **do**
  $S_t := b$ size of random sample from $\{1, \cdots, n\}$
  $L_t(w) = \frac{1}{b} \sum_{i \in S_t} l(w, z_i)$
  $\nabla w^{(t)} = \nabla L_t(w^{(t)})$
  $G^{(t)} = G^{(t-1)} + \nabla w^{(t)} \nabla w^{(t)^\top}$
  $w^{(t+1)} = w^{(t)} - \eta diag(G^{(t)})^{-\frac{1}{2}} \odot \nabla w^{(t)}$, where $\odot$ denotes Hadamard product
  end for
  Return w

Note that

$$(G^{(t)})_{jj} = \sum_{k=1}^{t} (\nabla w^{(k)})_j^2,$$

thus

$$\left(diag(G^{(t)})^{-\frac{1}{2}} \odot \nabla w^{(t)} \right)_j = \frac{\nabla w_j^{(t)}}{\sqrt{\sum_{k=1}^{t} \nabla (w^{(k)})_j^2}}.$$

The Adagrad control the learning rate coordinatewisely according to the variation of the gradient along the past trajectory. Adagrad reduces the learning rate in the direction with high variation.

RMSprop

- Parameters: Learning rate η, batch size b, initial value of w,$\gamma \in [0, 1)$
- Input : Sample $z_1, \cdots, z_n$
- Algorithm

  set $w^{(1)}, v^{(0)} = 0 \in \mathbb{R}^p$

  for $t = 1$ to iteration **do**

  $S_t := b$ size of random sample from $\{1, \cdots, n\}$

  $L_t(w) = \frac{1}{b} \sum_{i \in S_t} l(w, z_t)$

  $\nabla w^{(t)} = \frac{\partial L_t(w^{(t)})}{\partial w}$

  $v^{(t)} = \gamma v^{(t-1)} + (1 - \gamma)(\nabla w^{(t)} \odot \nabla w^{(t)})$

  $\Delta w^{(t)} = -\eta \frac{\nabla w^{(t)}}{\sqrt{v^{(t)}}}$

  $w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$

  end for

  Return w

Note that

$$v^{(t)} = (1-\gamma)(\sum_{k=1}^{t-1} \gamma^k (\Delta w^{(t-k+1)})^2),$$

which is exponential moving average of $(\Delta w^{(t-k+1)})^2$s. While the adagrad employs the arithmetic average.

It is trivial that the adagrad adopts the idea of the moment method to increase the weight of the variation of recently updated solutions.

Adam

- Parameters: Learning rate η, batch size b, initial value of $w, \gamma_1, \gamma_2 \in [0, 1), \epsilon > 0$
- Input : Sample $z_1, \cdots, z_n$
- Algorithm set $m^{(0)} = v^{(0)} = 0 \in \mathbb{R}^p$

  for $t = 1$ to iteration **do**

  $S_t :=$ b size of random sample from $\{1, \cdots, n\}$

  $L_t(w) = \frac{1}{b} \sum_{i \in S_t} l(w, z_i)$

  $\nabla w^{(t)} = \frac{\partial L_t(w^{(t)})}{\partial w}$

  $m^{(t)} = \gamma_1 m^{(t-1)} + (1 - \gamma_1) \nabla w^{(t)}$

  $v^{(t)} = \gamma_2 v^{(t-1)} + (1 - \gamma_2)(\nabla w^{(t)} \odot \nabla w^{(t)})$

  $\hat{m^{(t)}} = \frac{m^{(t)}}{1 - \gamma_1^t}, \hat{v^{(t)}} = \frac{v^{(t)}}{1 - \gamma_2^t}$

  $\Delta w^{(t)} = -\eta \frac{\hat{m^{(t)}}}{\sqrt{\hat{v^{(t)}}} + \epsilon}$

  $w^{(t+1)} = w^{(t)} + \Delta w^{(t)}$

  end for

  Return w

Adam compromises the updating strategies of the moment method and RMSprop.

Relation between Adam and previous algorithms

The first moment estimate m at the i step in Adam :

$$
\begin{aligned}
m^{(0)} &= 0 \\
m^{(1)} &= \gamma_1 \times m^{(0)} + (1-\gamma_1) \times \nabla w^{(1)} = (1-\gamma_1) \times \nabla w^{(1)} \\
m^{(2)} &= \gamma_1 \times m^{(1)} + (1-\gamma_1) \times \nabla w^{(2)} \\
&= \gamma_1(1-\gamma_1) \times \nabla w^{(1)} + (1-\gamma_1) \times \nabla w^{(2)} \\
&\vdots \\
m^{(i)} &= \gamma_1^{i-1}(1-\gamma_1)\nabla w^{(1)} + \cdots + (1-\gamma_1)\nabla w^{(i)}
\end{aligned}
$$

So, we can get general solution of $m^{(i)}$. Using this, we also can get general solution of $\hat{m^{(i)}}$.

$$\hat{m^{(t)}} = \frac{m^{(t)}}{1-\gamma_1^t} = \frac{\gamma_1^{t-1}(1-\gamma_1)\nabla w^{(1)} + \cdots + (1-\gamma_1)\nabla w^{(t)}}{(1-\gamma_1)(1+\gamma_1+\cdots+\gamma_1^{t-1})}$$
$$= \frac{\gamma_1^{t-1}\nabla w^{(1)} + \cdots + \nabla w^{(t)}}{1+\gamma_1+\cdots+\gamma_1^{t-1}}$$

That is,

$$\hat{m^{(t)}} = a_1\nabla w^{(1)} + \cdots + a_t\nabla w^{(t)}$$

where, $a_k = \frac{\gamma_1^{t-k}}{1+\gamma_1+\cdots+\gamma_{t-1}}, k = 1, \cdots, t.$

Note that

- $a_j > 0$
- $\sum_{j=1}^{t} a_j = 1$
- $a_j / a_{j+1} = \gamma_1 \leq 1$

We can easily see that $1 - \gamma_1^t$ is the normalizing term and that the above result of $\hat{m}^{(t)}$ is a exponential moving average of $\nabla w^{(j)}$, where $j = 1, \cdots, t$.

Similarly, we can get the general solution of $v^{(t)}$ and $\hat{v}^{(t)}$. Denote $(\nabla w^{(t)})^2 = \nabla w^{(t)} \odot \nabla w^{(t)}$ and if $\epsilon = 0$, they are expressed as the following.

$$v^{(t)} = \gamma_2^{t-1}(1-\gamma_2)(\nabla w^{(1)})^2 + \cdots + (1-\gamma_2)(\nabla w^{(t)})^2$$
$$\hat{v}^{(t)} = \frac{\gamma_2^{t-1}(\nabla w^{(1)})^2 + \cdots + (\nabla w^{(t)})^2}{1 + \gamma_2 + \cdots + \gamma_2^{t-1}}$$

We can easily see that the above result of $\hat{v}^{(t)}$ is a exponential moving average of $(\nabla w^{(j)})^2$ where $j = 1, \cdots, t$.

Finally, we can get the $\Delta w^{(i)}$.

$$\begin{aligned}\Delta w^{(t)} &= -\eta \left(\frac{\gamma_1^{t-1} \nabla w^{(1)} + \cdots + \nabla w^{(t)}}{1 + \gamma_1 + \cdots + \gamma_1^{t-1}} \right) \\ &\quad \odot \left(\frac{\gamma_2^{t-1} (\nabla w^{(1)})^2 + \cdots + (\nabla w^{(t)})^2}{1 + \gamma_2 + \cdots + \gamma_2^{t-1}} \right)^{-1/2}.\end{aligned}$$

- if $\gamma_1 = 0$ and denominator of $\Delta w^{(t)}, \sqrt{\frac{\gamma_2^{t-1}(\nabla w^{(1)})^2 + \cdots + (\nabla w^{(t)})^2}{1+\gamma_2+\cdots+\gamma_2^{t-1}}}$ is 1, then Adam is same as Stochastic gradient descent algorithm with random batch.
- if $\gamma_1 = 0$ then $\Delta w^{(t)} = -\eta_t \frac{\nabla w^{(t)}}{\sqrt{v}}$ with $\eta_t = \eta(1 - \gamma_2^t)$.

Adam is extended RMSprop which contains moment parameter and adaptive learning rate parameter.

- Stochastic Gradient Methods in Pytorch : torch.optim
  - https://pytorch.org/docs/stable/optim.html
- SGD example: https://arxiv.org/pdf/1512.03385.pdf
- RMSprop example: https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf