definite Hessian. Nonetheless they can be applied to minimize general functions.[1] Minimization of a quadratic function

Estimating the curvature $f''(0)$ by differencing first derivatives gives

$$f''(0) \approx \mathbf{s}_n \cdot \nabla E($$

if $f''(0) + \lambda_n^{(\text{old})} \|\mathbf{s}_n\|^2 \leq 0$, reset

$$\lambda_n^{(\text{new})} = \lambda_n^{(\text{old})} - \frac{f''(0)}{\|\mathbf{s}_n\|^2}$$

following which

$$f''(0) + \lambda_n^{(\text{new})} \|\mathbf{s}_n\|^2 = \lambda_n^{(\text{old})} \|\mathbf{s}_n\|^2 > 0.$$

Approximating the second derivative

The simplest way of determining σ in (4) is to choose a small constant > 0 and set

$$\sigma_n = \frac{}{\|\mathbf{s}_n\|}$$

on the n'th iteration. The renormalization ensures uniform scaling for varying directions and gradients. If f were in fact quadratic the exact choice of σ would be unimportant and there would be no need for small . Figure 1, however, shows that there is no particular advantage in having $\sigma_n \ll 1$ even when f is not quadratic. When $\lambda_n \approx 0$ the method proposed by (2), (3), (4) is equivalent to fitting a straight line to $f'(0)$ and $f'(\sigma_n)$ to give α_n as the estimated zero crossing of f'. The ideal σ_n would be one for which $\sigma_n = \alpha_n$. A

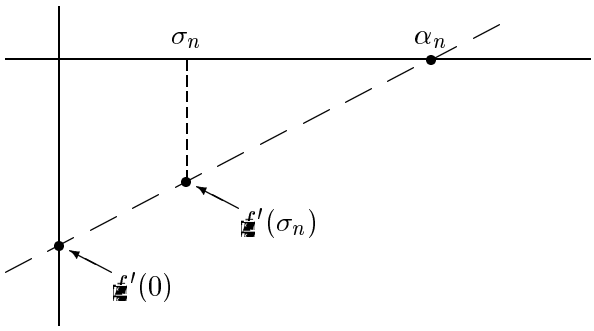

Figure 1: The extrapolation used to determine α_n.

The algorithm

The algorithm can be stated as follows:

0. choose weight vector $\mathbf{w}_0$, scalars $_0 > 0$, $\lambda_0 > 0$, $\pi \geq 0$ and initialize search direction:

$$\mathbf{g}_0 = \nabla E(\mathbf{w}_0)$$

$$\mathbf{s}_0 = -\mathbf{g}_0$$

$$\text{success} = \text{true},\ S = 0,\ n = 0$$

1. if success = true calculate first and second order directional derivatives:

$$\mu_n = \mathbf{s}_n \cdot \mathbf{g}_n \qquad \text{(directional gradient)}$$

$$\text{if } \mu_n \geq 0, \text{ set } \mathbf{s}_n = -\mathbf{g}_n,\ \mu_n = \mathbf{s}_n \cdot \mathbf{g}_n,\ S = 0$$

$$\kappa_n = \mathbf{s}_n \cdot \mathbf{s}_n, \quad \sigma_n = \frac{n}{\sqrt{\kappa_n}}$$

$$\gamma_n = \mathbf{s}_n \cdot \frac{\nabla E(\mathbf{w}_n + \sigma_n \mathbf{s}_n) - \nabla E(\mathbf{w}_n)}{\sigma_n} \qquad \text{(directional curvature)}$$

2. increase the working curvature: $\delta_n = \gamma_n + \lambda_n \kappa_n$
3. if $\delta_n \leq 0$ make δ_n positive and increase λ_n:

$$\delta_n = \lambda_n \kappa_n$$

$$\lambda_n = \lambda_n - \frac{\gamma_n}{\kappa_n}$$

4. calculate step size and adapt :

$$\alpha_n = -\frac{\mu_n}{\delta_n}$$

$$_{n+1} = \ _n \left(\frac{\alpha_n}{\sigma_n}\right)^{\pi}$$

5. calculate the comparison ratio:

$$\rho_n = \frac{2[E(\mathbf{w}_n + \alpha_n \mathbf{s}_n) - E(\mathbf{w}_n)]}{\alpha_n \mu_n}$$

$$\text{success} \ = \ \rho_n \geq 0$$

6. if $\rho_n < 0.25$, set $\lambda_{n+1} = \min\{\lambda_n + \frac{\delta_n(1-\rho_n)}{\kappa_n},\ \lambda_{\max}\}$

   if $\rho_n > 0.75$, set $\lambda_{n+1} = \max\{\lambda_n/2,\ \lambda_{\min}\}$

   otherwise, set $\lambda_{n+1} = \lambda_n$

7. if success = true then adjust weights:

   $$\mathbf{w}_{n+1} = \mathbf{w}_n + \alpha_n \mathbf{s}_n$$

   $$\mathbf{g}_{n+1} = \nabla E(\mathbf{w}_{n+1})$$

   $$S = S + 1$$

   else leave weights unchanged:

   $$\mathbf{w}_{n+1} = \mathbf{w}_n$$

   $$\mathbf{g}_{n+1} = \mathbf{g}_n$$

8. choose new search direction:

   if $S = S_{\max}$ restart algorithm in direction of steepest descent:

   $$\mathbf{s}_{n+1} = -\mathbf{g}_{n+1}$$

   success = true, $S = 0$

   else

   if success = true create new conjugate direction:

   $$\beta_n = \frac{(\mathbf{g}_n - \mathbf{g}_{n+1}) \cdot \mathbf{g}_{n+1}}{\mu_n}$$
   $$\mathbf{s}_{n+1} = -\mathbf{g}_{n+1} + \beta_n \mathbf{s}_n$$

   else use current direction again:

   $$\mathbf{s}_{n+1} = \mathbf{s}_n$$
   $$\mu_{n+1} = \mu_n,\ \kappa_{n+1} = \kappa_n,\ \sigma_{n+1} = \sigma_n,\ \gamma_{n+1} = \gamma_n$$

9. if $\|\mathbf{g}_{n+1}\| < \tau$ return $\mathbf{w}_{n+1}$ as desired minimum, else go to 1 with $n = n + 1$.

Notes on the algorithm

0. $_0 = 10^{-3}$ is satisfactory though not critical if $\pi > 0$. If a non-zero value of π is chosen then $\pi = 0.05$ or $\pi = 0.1$ are recommended. The initial value of λ is not critical though $\lambda_0 = 1$ is a natural choice. The algorithm starts in the direction of steepest descent.

1. Apart from the initial cycle, this step is only executed if the last cycle succeeded in error reduction. Otherwise no change in the weight vector has been made and this information is already known. Neither the Hestenes-Stiefel formula nor the Polak-Ribière formula guarantees that $\mathbf{s}_n$ is a *descent* direction, though usually it is. If $\mu_n \geq 0$, a restart is made in the direction of steepest descent for which $\mu_n = -\mathbf{g}_n \cdot \mathbf{g}_n$ is negative, otherwise the algorithm would have terminated at the last step of the previous cycle, assuming the two-norm is used.

3. After this step, $\delta_n = \gamma_n + \lambda_n \kappa_n$ as before, but with the new value of λ_n.

5. Remember that $\mu_n < 0$. The choice of $\rho_n \geq 0$ rather than $\rho_n > 0$ is deliberate. It safeguards against the algorithm getting stuck owing to limited floating-point precision. An alternative is to restart in the direction of steepest descent after a given number, 10 say, of consecutive failures.

6. λ_n must stay in the range $0 < \lambda_n < \infty$, otherwise no further rescaling is possible. $\lambda_{\min}$ and $\lambda_{\max}$ can be of the order of the smallest and largest positive floating point numbers provided by the implementation. When $\rho_n < 0.25$, the proposed rule increases λ_n by more than a factor 4, the intention being to avoid the possibility of more than 2 or 3 successive failures.

8. S is the total number of successes since the last restart in the direction of steepest descent. By default $S_{\max}$ is the dimension of the weight vector (the total number of weights and biases). For large210Td(largest)Tg3980Td(tha0Td.813.4402Tdfitor)]TJ33

function evaluation $E(\mathbf{w}_n + \alpha_n \mathbf{s}_n)$ though it is worth performing the full gradient evaluation $\nabla E(\mathbf{w}_n + \alpha_n \mathbf{s}_n)$ at this stage. If an error reduction results, $\mathbf{w}_n + \alpha_n \mathbf{s}_n$ will become the new weight vector $\mathbf{w}_{n+1}$ in step 7 and $\nabla E(\mathbf{w}_{n+1})$ will then already be known. If no error reduction occurs, the extra computation will have been wasted. On the other hand, if an error reduction does occur, the work involved in calculating only the function value $E(\mathbf{w}_n + \alpha_n \mathbf{s}_n)$ in step 5 will have to be redone. Assuming that successes are more common than failures, it is better on average to calculate the gradient in step 5. Note that at the end of the cycle both $E(\mathbf{w}_{n+1})$ and $\nabla E(\mathbf{w}_{n+1})$ are known.

All other significant calculations in a cycle are inner products. Each requires N multiplications and additions, where N is the number of weights. This is comparable to a forward pass of a single pattern. If $P \gg 1$, where P is the number of patterns in a batch, the cost of the inner product calculations is not significant.

Space Memory