Deep Learning

Gradient Descent Variations

Syed Irtaza Muzaffar

So far . . .

▶ Neural Networks are universal approximators.

▶ Backpropagation allows computation of derivatives in hidden layers.

▶ Gradient descent finds weights corresponding to local minimum of loss surface.

▶ In this lecture: alternative methods of finding local minima of loss surface.

  ▶ First-order methods
    ▶ Rprop
  ▶ Second-order methods
    ▶ Taylor series approximation
    ▶ Newton's method
    ▶ Quickprop

▶ Next lecture:
  ▶ Momentum-based first-order methods

Gradient Descent in Higher Dimensions

▶ Let $\Delta \mathbf{w}^{\tau+1}$ denote the step at time $\tau + 1$.

$$w^{\tau+1} = w^\tau + \Delta w^{\tau+1}$$

▶ For gradient descent

$$\Delta \mathbf{w}^{\tau+1} = -\eta \nabla_{\mathbf{w}}^\tau L$$

▶ For gradient descent in $1D$,

$$\Delta w^{\tau+1} = -\eta \left. \frac{dL}{dw} \right|_\tau$$

The only issue is determining learning rate η.

Gradient Descent in Higher Dimensions

▶ Let $\Delta w^{\tau+1}$ denote the step at time $\tau + 1$.

$$\underline{w^{\tau+1}} = \underline{w^{\tau}} + \underbrace{\underline{\Delta w^{\tau+1}}}_{step}$$

▶ For gradient descent

$$\underline{\Delta w^{\tau+1}} = -\underset{\underset{\text{learning}}{\uparrow}}{\eta} \underset{\underset{\text{rate}}{}}{} \underbrace{\nabla_w^{\mathcal{I}} L}_{\substack{\text{gradient vector} \\ \text{computed at } \vec{w}^{\tau}}}$$

▶ For gradient descent in $1D$,

$$\underline{\Delta w^{\tau+1}} = -\eta \left. \frac{dL}{dw} \right|_{\tau}$$

The only issue is determining learning rate η.

?

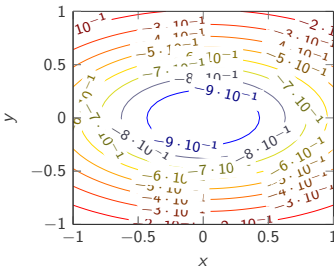$\Delta = $ delta $ = $ change

$\Delta x = $ change in x

$\nabla = $ nabla $ = $ rate of change

$\nabla_x \vec{y} = \begin{bmatrix} \frac{dy_1}{dx} \\ \frac{dy_2}{dx} \\ \vdots \end{bmatrix}$

$$f(x, y) = -\exp\left(-(\tfrac{3}{4}x)^2 - (\tfrac{5}{4}y)^2\right)$$
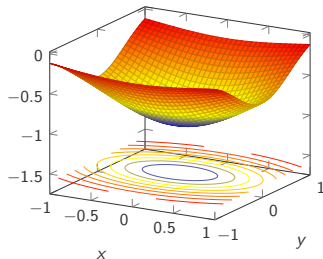
Iso-contours of $f(x, y)$

A function that changes faster in y-direction.

▶ In higher dimensions, if $\left|\frac{\partial L}{\partial w_i}\right| >> \left|\frac{\partial L}{\partial w_j}\right|$ then using the same η *can* result in overshooting in the direction of w_i and very slow convergence in the direction of w_j.

▶ Solution: separate learning rate η_i for each direction w_i.

Resilient Propagation (Rprop)

▶ In Rprop[1], each direction is handled independently.

▶ Increase learning rate for direction i if current derivative has same sign as previous derivative.

▶ Otherwise, you just overshot a minimum.
   ▶ So go back to previous location.
   ▶ Decrease learning rate for that direction.
   ▶ Update parameter with this smaller step.

$$\eta_i = \begin{cases} \alpha\eta_i & \text{if } \left.\frac{\partial L}{\partial w_i}\right|_{\tau} * \left.\frac{\partial L}{\partial w_i}\right|_{\tau-1} > 0 \\ \beta\eta_i & \text{if } \left.\frac{\partial L}{\partial w_i}\right|_{\tau} * \left.\frac{\partial L}{\partial w_i}\right|_{\tau-1} < 0 \\ \eta_i & \text{otherwise} \end{cases}$$

▶ *Hyperparameters* should follow the constraint $\alpha > 1$ and $\beta < 1$.

[1]Riedmiller and Braun, 'A direct adaptive method for faster backpropagation learning: The RPROP algorithm'.

Resilient Propagation (Rprop)

$$\frac{\partial L}{\partial w_i} = 0$$

$$\frac{\partial L}{\partial w_i} \text{ when } w_i \nearrow \frac{\partial L}{\partial w_i} \neq 0 \text{ changes}$$

- In Rprop[1], each direction is handled independently.

- Increase learning rate for direction i if current derivative has same sign as previous derivative.

- Otherwise, you just overshot a minimum.

  - So go back to previous location.
  - Decrease learning rate for that direction.
  - Update parameter with this smaller step.

$$\frac{\partial L}{\partial w_i}\bigg|_\tau \quad * \quad \frac{\partial L}{\partial w_i}\bigg|_{\tau-1}$$

$$\eta_i = \begin{cases} \alpha\eta_i & \text{if } \frac{\partial L}{\partial w_i}\bigg|_\tau * \frac{\partial L}{\partial w_i}\bigg|_{\tau-1} > 0 \quad \leftarrow \text{when both derivatives had the same direction} \\ \beta\eta_i & \text{if } \frac{\partial L}{\partial w_i}\bigg|_\tau * \frac{\partial L}{\partial w_i}\bigg|_{\tau-1} < 0 \quad \leftarrow \text{different direction} \\ \eta_i & \text{otherwise} \end{cases}$$

$\alpha > 1$ $\beta < 1$

both agree that local minimum was not crossed.

w_i

- *Hyperparameters* should follow the constraint $\alpha > 1$ and $\beta < 1$.

[1] Riedmiller and Braun, 'A direct adaptive method for faster backpropagation learning: The RPROP algorithm'.

Resilient Propagation (Rprop)

▶ Typical values are $\alpha = 1.2$ and $\beta = 0.5$.
  ▶ Increase learning rate slowly but decrease quickly when you overshoot.

▶ In practice, learning rates are bounded via $\eta_{\min}$ and $\eta_{\max}$.

$$\eta_i = \begin{cases} \min(\alpha\eta_i, \eta_{\max}) & \text{if } \left.\frac{\partial L}{\partial w_i}\right|_\tau * \left.\frac{\partial L}{\partial w_i}\right|_{\tau-1} > 0 \\ \max(\beta\eta_i, \eta_{\min}) & \text{if } \left.\frac{\partial L}{\partial w_i}\right|_\tau * \left.\frac{\partial L}{\partial w_i}\right|_{\tau-1} < 0 \\ \eta_i & \text{otherwise} \end{cases}$$

▶ Rprop converges much faster than gradient descent.

▶ But it works well when derivatives are accumulated over large batches.

Taylor Series Approximation

▶ If values of a function $f(a)$ and its derivatives $f'(a), f''(a), \ldots$ are known at a value a, then we can approximate $f(x)$ for <u>x close to a</u> via the *Taylor series expansion*

$$f(x) \approx f(a) + (x-a)^1 \frac{f'(a)}{1!} + (x-a)^2 \frac{f''(a)}{2!} + (x-a)^3 \frac{f'''(a)}{3!} + O((x-a)^4)$$

▶ Using $\Delta x = x - a$, Taylor series can be equivalently expressed as

$$f(a + \Delta x) \approx f(a) + (\Delta x)^1 \frac{f'(a)}{1!} + (\Delta x)^2 \frac{f''(a)}{2!} + (\Delta x)^3 \frac{f'''(a)}{3!} + O((\Delta x)^4)$$
$$= \sum_{n=0}^{\infty} \frac{1}{n!} f^n(a) (\Delta x)^n$$
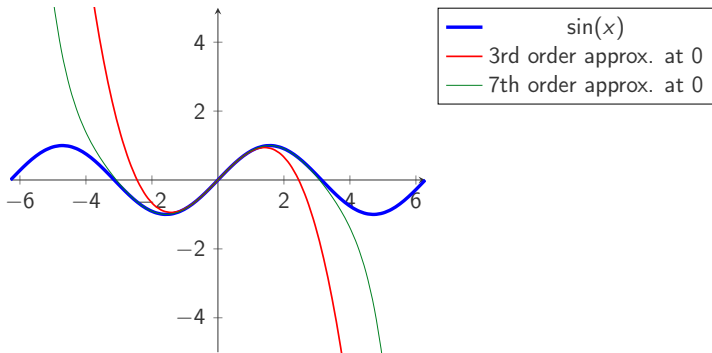
Taylor Series Approximation
Examples

▶ For x around $a = 0$
  ▶ $\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$
  ▶ $e^x \approx 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$

Taylor Series Approximation
Not very useful for x not close to a

The sine function (blue) is closely approximated around 0 by its Taylor polynomials. The 7th order approximation (green) is good for a full period centered at 0. However, it becomes poor for $|x - 0| > \pi$.

Taylor Series Approximation

▶ It is often convenient to use the first-order Taylor expansion

$$f(a + \Delta x) \approx f(a) + \Delta x f'(a)$$

or the second order Taylor expansion

$$f(a + \Delta x) \approx f(a) + \Delta x f'(a) + \frac{1}{2}(\Delta x)^2 f''(a)$$

▶ In d-dimensional input space

$$f(\mathbf{a} + \Delta \mathbf{x}) \approx f(\mathbf{a}) + \Delta \mathbf{x}^T \nabla f + \frac{1}{2} \Delta \mathbf{x}^T \mathbf{H} \Delta \mathbf{x}$$

where $\mathbf{H} \in \mathbb{R}^{d \times d}$ is the Hessian matrix composed from second derivatives.

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d} \end{bmatrix}$$

Newton's Method for finding stationary points

▶ Starting from a_0, we want to find a stationary point of f.

▶ Instead of actual function f, use a quadratic approximation (second-order Taylor expansion) of f at a_0.

▶ Find a step Δx such that $a_0 + \Delta x$ minimizes the quadratic approximation of f.

$$\frac{d}{d\Delta x}\left(f(a_0) + f'(a_0)\Delta x + \frac{1}{2}f''(a_0)(\Delta x)^2\right) = 0$$
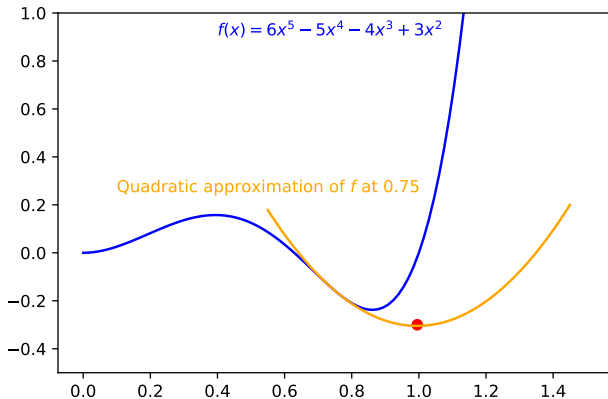
$$f'(a_0) + f''(a_0)\Delta x = 0$$

$$\Delta x = -\frac{f'(a_0)}{f''(a_0)}$$

▶ Move to $a_1 = a_0 + \Delta x$ and repeat the process at a_1.
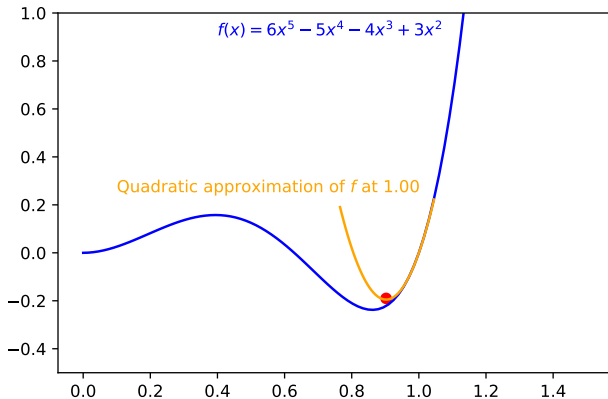
▶ Continue until convergence to a stationary point a_n.
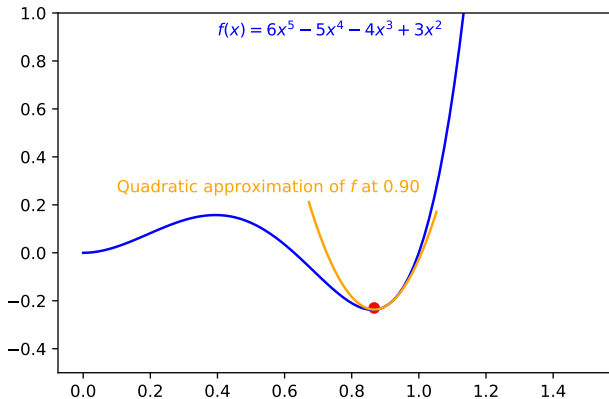
Newton's Method for finding stationary points

$f(x) = 6x^5 - 5x^4 - 4x^3 + 3x^2$

Newton's Method for finding stationary points

$f(x) = 6x^5 - 5x^4 - 4x^3 + 3x^2$

Quadratic approximation of f at 0.75
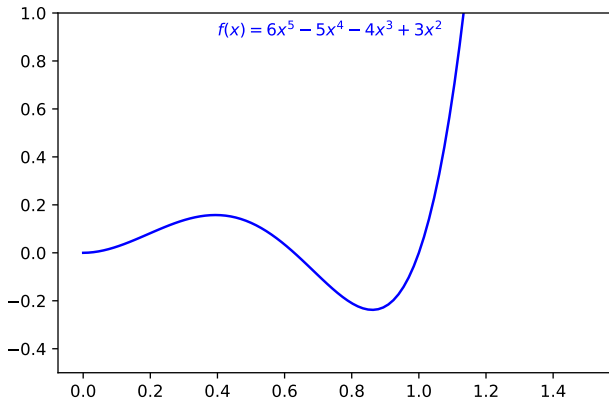
Newton's Method for finding stationary points

Newton's Method for finding stationary points

$f(x) = 6x^5 - 5x^4 - 4x^3 + 3x^2$

Quadratic approximation of f at 0.90

Newton's Method for finding stationary points

$f(x) = 6x^5 - 5x^4 - 4x^3 + 3x^2$
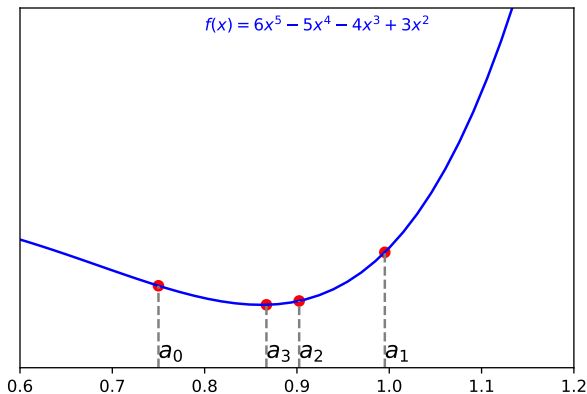
Newton's Method
Role of the 2nd-derivative

▶ For weights of a neural network, Newton's update corresponds to

$$w^{\tau+1} = w^{\tau} - \left(\frac{\partial^2 L}{\partial w^2}\right)^{-1} \frac{\partial L}{\partial w}$$

▶ In other words, gradient descent learning rate η corresponds to inverse of 2nd-derivative.

▶ Division by 2nd-derivative can also be viewed as normalising the gradient.

▶ In higher dimensions

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \mathbf{H}^{-1}\nabla_{\mathbf{w}}L$$

The inverse Hessian matrix normalises the gradient vector.

Newton's Method
Role of the 2nd-derivative

▶ Complete Hessian matrix is rarely used because of its size and computational cost of inverting it.

▶ Common assumption: diagonal Hessian matrix.

$$
\mathbf{H} = \begin{bmatrix}
\frac{\partial^2 f}{\partial x_1 \partial x_1} & 0 & \cdots & 0 \\
0 & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \frac{\partial^2 f}{\partial x_d \partial x_d}
\end{bmatrix}
$$

▶ Inverse of diagonal matrix is cheap (reciprocal of entries on the diagonal).

Quickprop

▶ Decouple all directions.

▶ Perform Newton updates in each direction.

$$w_i^{\tau+1} = w_i^\tau - \left(\frac{\partial^2 L}{\partial w_i^2}\right)^{-1} \frac{\partial L}{\partial w_i}$$

▶ Approximate 2nd-derivative *numerically* by finite difference of 1st-derivatives.

$$\frac{\partial^2 L}{\partial w_i^2} \approx \frac{\left.\frac{\partial L}{\partial w_i}\right|_\tau - \left.\frac{\partial L}{\partial w_i}\right|_{\tau-1}}{\Delta w_i^{\tau-1}}$$

▶ Leads to very fast convergence.

▶ Some instability where loss is non-convex since everything is based on assumptions of convexity (quadratic approximation in Newton's method).

Fahlman, *An empirical study of learning speed in back-propagation networks.*

Summary

▶ For complex and non-convex loss functions of deep networks, vanilla gradient descent can get stuck in poor local minima and saddle points.

▶ It can also converge very slowly.

▶ Different directions require different learning rates.

▶ Adaptive learning rates are very important.

▶ Next lecture: momentum-based first-order methods.