11. Optimization Algorithms

Junggwon Shin, shinjunggwon@gmail.com

Summary for Dive Into Deep Learning, https://d2l.ai/chapter_preface/index.html

Motivation

- Optimization problems in DL tasks
    1. A loss function is defined first for the DL model
    2. we use an gradient descent-based optimization algorithms for minimizing loss defined by the function.

- Goal of DL: finding a suitable model with minimum **generalization error,** such as recall and precision, for given data.
- Goal of Optimization: minimizing **training error (loss)** for a given objective function.
- For achieve the goal of DL, we use an optimization algorithms to reduce training error (fitting a model to data) with a desire that minimizing **training error** has same effect with minimizing **generalization error**.

- Optimization problem (MLE, p$(x|\theta)$) solving
    - Analytic approach (Normal Equation)
    - Numerical approach (Gradient descent, EM Algorithm)
    - https://github.com/howawindelu/dive-into-deep-learning/blob/master/week3/week3_1_lukeshin.pdf
    - http://faculty.washington.edu/yenchic/18A_stat516/Lec8_EM_SGD.pdf

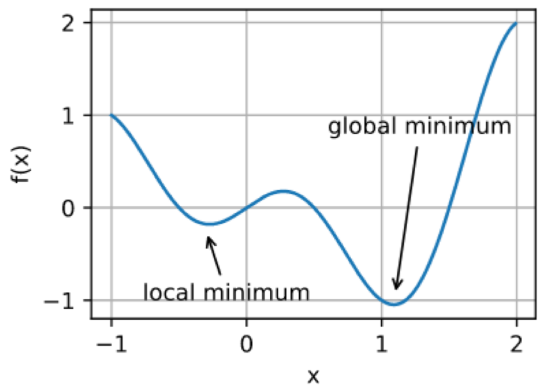
- In deep learning, most objective functions are complicated and do not have analytical solutions.
- Instead, we must use **Gradient descent-based optimization algorithms**.

$$x \leftarrow x - \eta \nabla f(x)$$

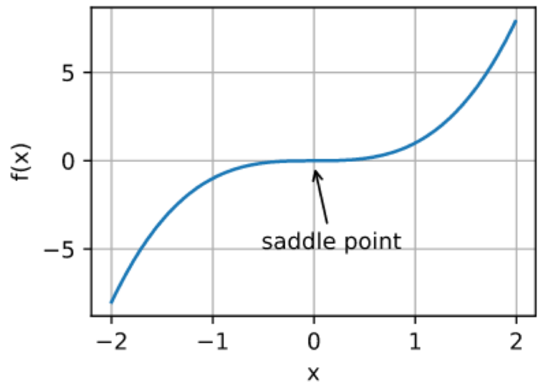
Optimization Challenges in Deep Learning

- Problems in gradient updates for DL model training

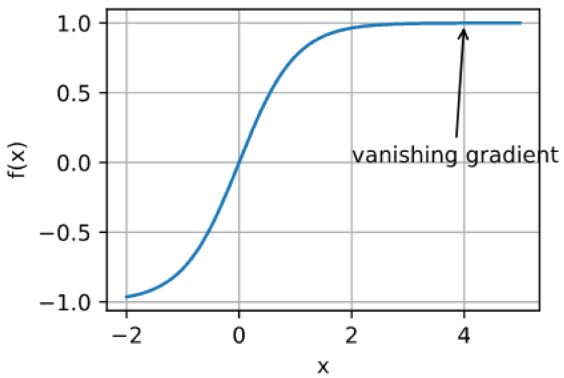
1. Local Minima
    - When the numerical solution of an optimization problem is near the local optimum, it only minimize the objective function locally as the gradient of the objective function's solutions approaches or **becomes zero**.
    - Ways to knock the parameter out of the local minimum.
        - Stochastic gradient descent
        - Add some noise $y = x - \eta \nabla f(x) + \epsilon.$



2. Saddle Points
    - a point where all gradients of a function vanish but which is neither a global nor a local minimum ($f''(x) = 0,\ f'(x) = 0$)



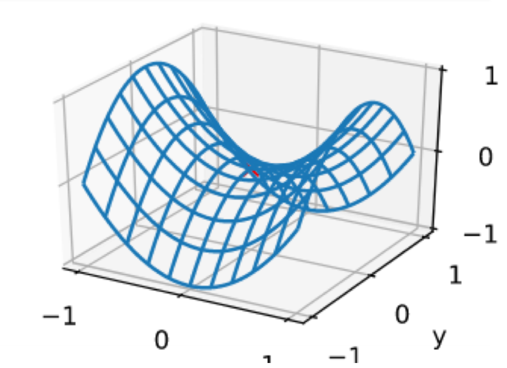
3. Vanishing Gradients
    - Problems from the derivative of nonlinear activation functions
    - $f(x) = \tanh(x),\ f'(x) = 1 - \tanh^2(x),\ f'(4) = 0.0013$



11.2 Convexity

Mathematical interpretation of saddle points

- When all **eigenvalues** of the function's Hessian matrix at the zero-gradient position are
  - All **eigenvalues** are positive – local minimum
  - All eigenvalues are negative - local maximum
  - All eigenvalues are both negative and positive – saddle point
- For high-dimensional problems the likelihood that at least some of the eigenvalues are negative is quite high. This makes saddle points more likely than local minima.
- **Convex** functions are those where the eigenvalues of the Hessian are **never** negative, but most deep learning problems do not fall into this category



$$f(x, y) = x^2 - y^2$$

$$\mathbf{H}_{f(x,y)} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$\lambda_{\mathbf{H}_{f(x,y)}} = 2, -2$$

Convexity

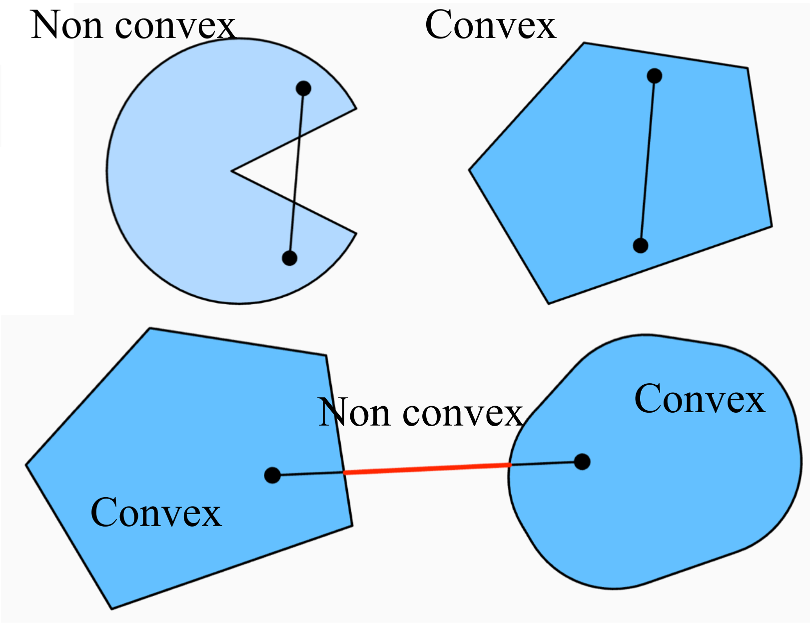
- Even though the optimization problems in deep learning are generally nonconvex, they often exhibit some properties of convex ones near local minima.
- Convexity provides easier analysis and algorithm test.
- If the algorithm performs poorly even in the convex setting we should not hope to see great results otherwise.

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

11.2 Convexity

Convex Sets

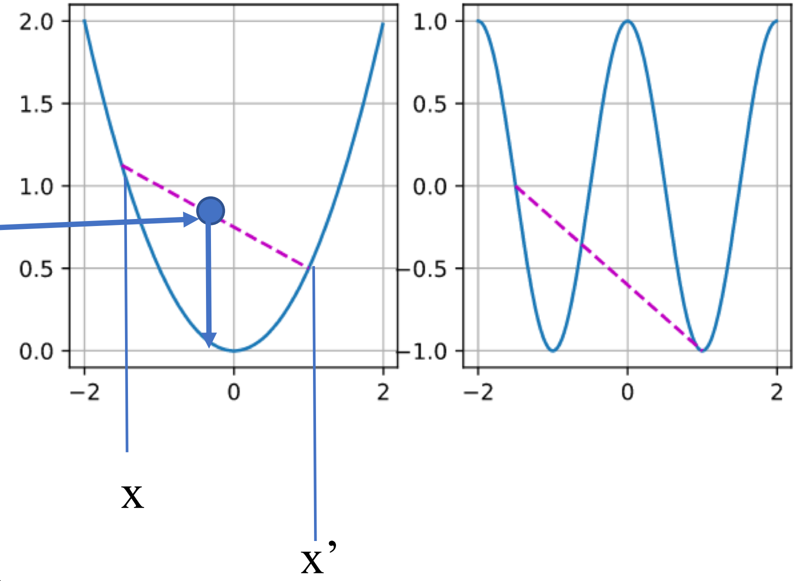
- A set $\boldsymbol{X}$ in a vector space is convex if $\lambda \cdot a + (1-\lambda) \cdot b \in X$ whenever $a, b \in X$.
  - for any a, $b \in \boldsymbol{X}$ and $\lambda \in [0,1]$
  - That is, the line segment connecting a and b also in $\boldsymbol{X}$
- Intersection
  - Given convex sets X_i their intersection $\cap X_i$ is convex.
- Union
  - General unions of convex sets need not be convex.
  - Thus, the problems in deep learning can be defined on convex domains, even though the function's entire domain is not convex



Convex Functions

- Generalization of the definition of **convexity**
- $f: X \rightarrow \mathbb{R}$ is convex if for all x, $x' \in X$ and for all $\lambda \in [0,1]$ we have

$$\boxed{\lambda f(x) + (1-\lambda) f(x')} \geq f(\lambda x + (1-\lambda) x').$$

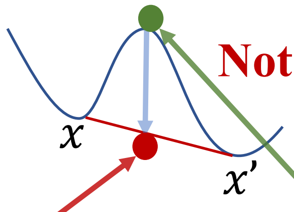


Jensen's Inequality

$$\sum_i \alpha_i f(x_i) \geq f\left(\sum_i \alpha_i x_i\right) \text{ or } E_x[f(x)] \geq f\left(E_x[x]\right) \text{ where } a_i \geq 0 \text{ and } \sum_i \alpha_i = 1$$

The expectation of a convex function is larger than the convex function of an expectation (Jensen's inequality).

Properties of Convex functions



Not satisfying Jensen's inequality

1. No Local Minima
   - Proof with the contrary (귀류법)
     - if $f(x),\ x \in \boldsymbol{X}$ is local **minima**, that means $f(x) > f(x')$ where x' is another value in $x' \in \boldsymbol{X}$.

$$f(x) > \boxed{\lambda f(x) + (1-\lambda) f(x')} \geq \boxed{f(\lambda x + (1-\lambda) x'),} \qquad \lambda \in [0,1)$$

     - This contradicts the assumption that $f(x)$ is a local minimum. Thus, convex function has no local minima

2. Second Derivatives of Convex Function
   - $f''(x) >= 0$, for a convex function, its second derivative must be **nonnegative**
     - Eigenvalues of its Hessian Hessian matrix must be nonnegative
     - $f'(x)$ is a monotonically increasing function

3. Easier Constraints Handling

Solving convex optimization as Lagrangian.

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x})$$
$$\text{subject to } c_i(\mathbf{x}) \leq 0 \text{ for all } i \in \{1, \ldots, N\}.$$

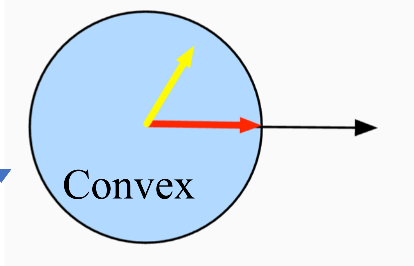
$$L(\mathbf{x}, \alpha) = f(\mathbf{x}) + \sum_i \alpha_i c_i(\mathbf{x}) \text{ where } \alpha_i \geq 0.$$

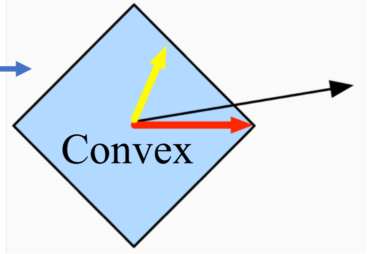
$\frac{\lambda}{2}\|\mathbf{w}\|^2$ Penalties

- Lagrangian function
  - $f^* \geq min_x L(x, a)$, Lagrangian function (dual function) L provides the lower bound of optimal primal function f^*
  - *maximize* Lagrangian function L with respect to α can simultaneously *minimize* it with respect to $\mathbf{x}$
  - When primal function f is convex, $f^* = L^*$ (strong duality)
  - Role of Lagrange Multiplier a_i: regulating constraint function $c_i(x)$ for suitable optimization

Handling Constraints

- Lagrangian
- Penalties (Regularization)
- Projections: a vector onto a convex set
  - Gradient clipping, $\mathbf{g} \leftarrow \mathbf{g} \cdot \min(1, c/\|\mathbf{g}\|)$. It equals to projecting g onto a ball with radius c.
  - L1 regularization ($\lambda|\mathbf{w}|$) equals to project a vector onto a diamond-shaped convex set





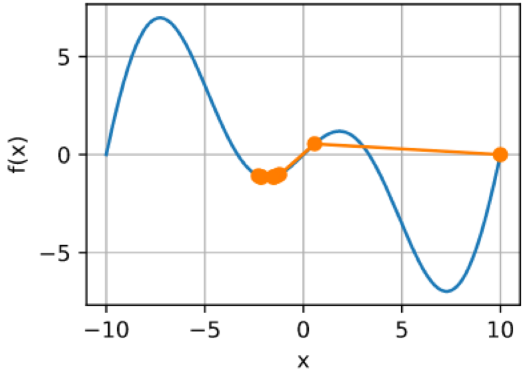
Gradient Descent in One Dimension

- From Taylor expansion, For a continuously differentiable real-valued function $f:\mathbb{R}\to\mathbb{R}$, $f(x+\epsilon) = f(x) + \epsilon f'(x) + \mathcal{O}(\epsilon^2).$
- We hope to move ϵ a little in the direction of the negative gradient will decrease. Thus, we set $\epsilon = -\eta f'(x)$ where $\eta > 0$.

$$f(x - \eta f'(x)) = f(x) - \cancel{\eta f'^2(x) + \mathcal{O}(\eta^2 f'^2(x))}.$$

- we can always choose η small enough for the higher order terms ($\mathcal{O}(\eta^2 f'^2(x))$) become irrelevant.

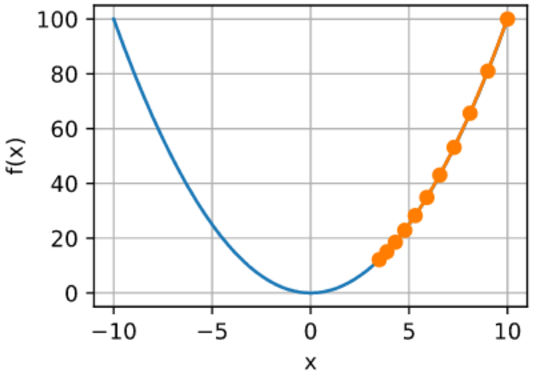
$$f(x - \eta f'(x)) \lessapprox f(x).$$



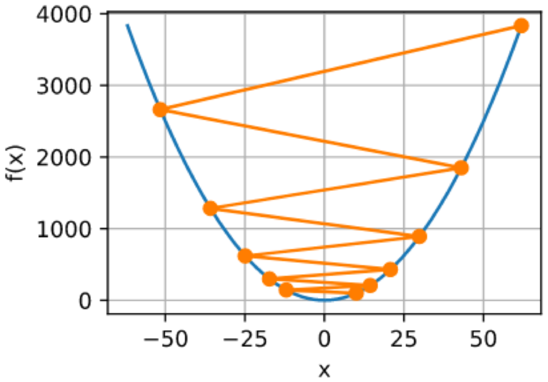
- Thus, if we use $x \leftarrow x - \eta f'(x)$ to iterate x, the value of function $f(x)$ might decline.
- In gradient descent
  - 1. choose an initial value x and a constant η>0
  - 2. use them to continuously and iterate x until the stop condition is reached

Learning Rate

- The learning rate η can be set by the algorithm designer
- Small learning rate: it will cause x to update very slowly, requiring more iterations to get a better solution
- High learning rate: higher order terms become significant. it cannot be guaranteed that the iteration will make lower $f(x)$.



small learning rate



high learning rate

Newton's Method

- Determine η automatically using second derivative of an objective function
- Taylor expansion, $f(\mathbf{x}+\epsilon) = f(\mathbf{x}) + \boxed{\epsilon^\top \nabla f(\mathbf{x}) + \frac{1}{2}\epsilon^\top \nabla\nabla^\top f(\mathbf{x})\epsilon} + \mathcal{O}(\|\epsilon\|^3)$.
- For Hessian of f, $H_f := \nabla\nabla^\top f(\mathbf{x})$ (d × d matrix)
  - For deep networks, cost for Hessian may be large, due to the cost of storing O(d^2) and calculation O($d^{2.xxx}$)
- The objective of gradient descent is $f(\mathbf{x}+\epsilon) = f(\mathbf{x})$

$$\boxed{\nabla f(\mathbf{x}) + H_f \epsilon = 0} \text{ and hence } \epsilon = -H_f^{-1}\nabla f(\mathbf{x}).$$

- Thus, we update $x_{k+1} = x_k - f'(x_k)/f''(x_k)$

Convergence Analysis

- Let the distance from optimal x^* as $e_k := x_k - x^*$ where x_k is the value of x at k-th iteration
- We hope to the first order $f'(x^*) = 0$ and when using Taylor series expansion,
- f'(x)=0 = f'(xk-ek)

$$0 = f'(x_k - e_k) = f'(x_k) - e_k f''(x_k) + \frac{1}{2}e_k^2 f'''(\xi_k). \Rightarrow e_k - \boxed{f'(x_k)/f''(x_k)} = \boxed{\frac{1}{2}e_k^2 f'''(\xi_k)/f''(x_k)}.$$

- First order gradient update: $x_{k+1} - x_k = e_k = \eta f'(x) \rightarrow e_{k+1} \leq a e_k$ (linear convergence)
- Second order gradient update: $x_{k+1} - x_k = e_k = \eta f'(x)/f''(x) \rightarrow e_{k+1} \leq a {e_k}^2$ (quadratic convergence)
  - For Newton's method, faster convergence is possible.
- *Preconditioning*: to relieve computation complexity for Hessian we can only compute the diagonal entries,

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \operatorname{diag}(H_f)^{-1}\nabla f(\mathbf{x}).$$

Batch Gradient Update

- Update weights once with the mean of gradients for all data, complexity O(n)
- High cost for a large dataset

$$\nabla f(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}).$$

Stochastic Gradient Update

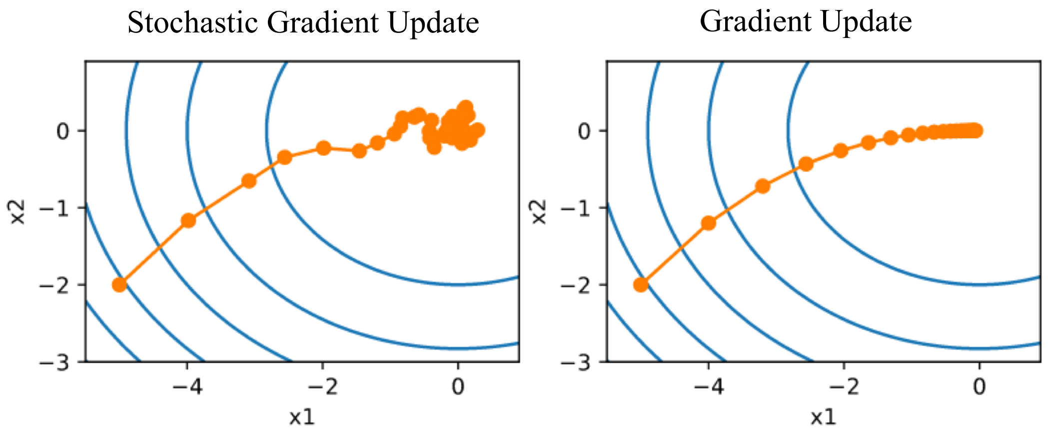
- Uniformly sample an index $i \in \{1, \ldots, n\}$ for data examples at random, and compute the gradient $\nabla f_i(\mathbf{x})$ to update

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f_i(\mathbf{x}).$$

- Complexity: O(1)
- The stochastic gradient $\nabla f_i(\mathbf{x})$ is the unbiased estimate of gradient $\nabla f(\mathbf{x})$
  - On average, the stochastic gradient is a good estimate of the gradient.
  - The difference between the true value of the parameter being estimated and the estimator's expected value (bias) is zero.

$$\mathbb{E}_i \nabla f_i(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\mathbf{x}) = \nabla f(\mathbf{x}).$$

- The trajectory of the variables in the SGD is much more noisy than that of gradient descent
  - Thus, we reduce the learning rate *dynamically* as optimization progresses



Dynamic Learning Rate

- a time-dependent learning rate function $\eta(t)$

$\eta(t) = \eta_i$ if $t_i \le t \le t_{i+1}$	piecewise constant
$\eta(t) = \eta_0 \cdot e^{-\lambda t}$	exponential
$\eta(t) = \eta_0 \cdot (\beta t + 1)^{-\alpha}$	polynomial

Convergence Analysis for Convex Objectives

- R(w): expected loss for weight parameter w, data sampling distribution P(x),

$$R(\mathbf{w}) = E_{\mathbf{x}\sim P}[l(\mathbf{x}, \mathbf{w})]$$

- w*: optimal parameter

NLL loss

$$l(\mathbf{x}_t, \mathbf{w}^*) \ge l(\mathbf{x}_t, \mathbf{w}_t)$$

$$l(\mathbf{x}_t, \mathbf{w}^*) \boxed{- \mathbf{w}^* \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w}_t)} \ge l(\mathbf{x}_t, \mathbf{w}_t) - \mathbf{w}_t \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w})$$

$$\boxed{\boxed{l(\mathbf{x}_t, \mathbf{w}^*) \ge l(\mathbf{x}_t, \mathbf{w}_t)} + \langle \mathbf{w}^* - \mathbf{w}_t, \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w}_t) \rangle .}$$

w*l(x,w) – wtl(x,w)

$$\begin{aligned} \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 &= \|\mathbf{w}_t - \eta_t \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w}) - \mathbf{w}^*\|^2 \\ &= \|\mathbf{w}_t - \mathbf{w}^*\|^2 + \boxed{\eta_t^2 \|\partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w})\|^2} - 2\eta_t \boxed{\langle \mathbf{w}_t - \mathbf{w}^*, \partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w}) \rangle .} \end{aligned}$$

inner product

$$\boxed{\eta_t^2 \|\partial_{\mathbf{w}} l(\mathbf{x}_t, \mathbf{w})\|^2 \le \eta_t^2 L^2 .}$$

$d(f(x), f(y)) \le k \cdot d(x, y)$, k is **Lipschitz constant**

https://en.wikipedia.org/wiki/Lipschitz_continuity

$$\|\mathbf{w}_t - \mathbf{w}^*\|^2 - \|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2 \ge 2\eta_t (l(\mathbf{x}_t, \mathbf{w}_t) - l(\mathbf{x}_t, \mathbf{w}^*)) - \eta_t^2 L^2 .$$

- Taking expectations over this expression

$$E_{\mathbf{w}_t}\left[\|\mathbf{w}_t - \mathbf{w}^*\|^2\right] - E_{\mathbf{w}_{t+1}|\mathbf{w}_t}\left[\|\mathbf{w}_{t+1} - \mathbf{w}^*\|^2\right] \ge 2\eta_t [E[R[\mathbf{w}_t]] - R^*] - \eta_t^2 L^2 .$$

11.4 Stochastic Gradient Descent

Convergence Analysis for Convex Objectives (Cont.)

- For summing over the inequalities for $t \in \{0,\ldots,T\}$ $\quad \|\mathbf{w}_0 - \mathbf{w}^*\|^2 \geq 2\sum_{t=1}^{T} \eta_t [E[R[\mathbf{w}_t]] - R^*] - \boxed{L^2 \sum_{t=1}^{T} \eta_t^2}.$

$$\|w_t - w^*\| - \|w_{t+1} - w^*\| \geq 2\eta l(x_t, w_t) - 2\eta l(x_t, w^*) - \eta^2 L^2$$
$$\|w_{t-1} - w^*\| - \|w_t - w^*\| \geq 2\eta l(x_{t-1}, w_t) - 2\eta l(x_{t-1}, w^*) - \eta^2 L^2$$
$$\|w_{t-2} - w^*\| - \|w_{t-1} - w^*\| \geq 2\eta l(x_{t-2}, w_t) - 2\eta l(x_{t-2}, w^*) - \eta^2 L^2$$
$$\ldots$$
$$\|w_0 - w^*\| - \|w_1 - w^*\| \geq 2\eta l(x_{t-3}, w_t) - 2\eta l(x_0, w^*) - \eta^2 L^2$$

- $\mathbf{w}_0$ is given and let $\|w_0 - w^*\| = r$
- Due to convexity, $\sum_t \eta_t E[R[\mathbf{w}_t]] \geq \boxed{\sum \eta_t} R[E[\bar{\mathbf{w}}]]$

$$R[E[\bar{\mathbf{w}}]] - R^* \leq \frac{r^2 + L^2 \sum_{t=1}^{T} \eta_t^2}{\boxed{2\sum_{t=1}^{T} \eta_t}}.$$

R: expected loss

$$0 = r^2 + TL^2\eta^2$$
$$\eta^2 = r^2 / TL^2$$
$$\eta = r / L\sqrt{T}$$

- The speed of convergence
  - how far away from optimality the initial value, r
  - Lipschitz constant, L (upperbound of gradient)
  - choice of the **learning rate, $\boldsymbol{\eta}$**
- Expectation of weight parameter, $\bar{\mathbf{w}} := \frac{\sum_{t=1}^{T} \eta_t \mathbf{w}_t}{\sum_{t=1}^{T} \eta_t}.$

- For given L, T, and r, we can pick $\eta = \frac{r}{L\sqrt{T}}$ with time complexity $O(1/\sqrt{T})$
- Even though T is not given, we can obtain good solution for any time T with $\eta = O(1/\sqrt{T})$

Stochastic Gradients for Finite Samples

- Since our sample is limited, we instead iterated over **all instances exactly once**
- If we choose an element i from a uniform distribution with replacement for N trials,
  - Probability of the element is selected for each trial: N^{-1}

$$P(\text{choose } i) = 1 - P(\text{omit } i) = 1 - (1 - N^{-1})^N \approx 1 - e^{-1} \approx 0.63.$$

- The probability of picking a sample exactly once is given by

$$\binom{N}{1} N^{-1}(1 - N^{-1})^{N-1} = \frac{N-1}{N}(1 - N^{-1})^N \approx e^{-1} \approx 0.37.$$

- This leads to an **increased variance** and decreased data efficiency relative to sampling without replacement

Minibatch Stochastic Gradient Descent

- Stochastic Gradient Descent is not particularly computationally efficient since CPUs and GPUs cannot exploit the full power of vectorization.
    - Reading a single byte incurs the cost of a much wider access.
- The main contribution of minibatch is increasing computational efficiency
    - all elements of the minibatch $|B_t|$ are drawn uniformly at random from the training set, the expectation of the gradient remains unchanged.
    - The standard deviation of gradients is reduced by a factor of $\boxed{|B_t|^{-1/2}}$

$$\mathbf{w} \leftarrow \mathbf{w} - \eta_t \mathbf{g}_t, \quad \mathbf{g}_t = \partial_{\mathbf{w}} \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} f(\mathbf{x}_i, \mathbf{w})$$

$$\sigma_M = \frac{\sigma}{\sqrt{N}}$$

$\mathrm{std}[\bar{g}] = \mathrm{std}[g_B] / \mathrm{sqrt}(|B_t|^{-1/2})$

standard error of the mean gradient