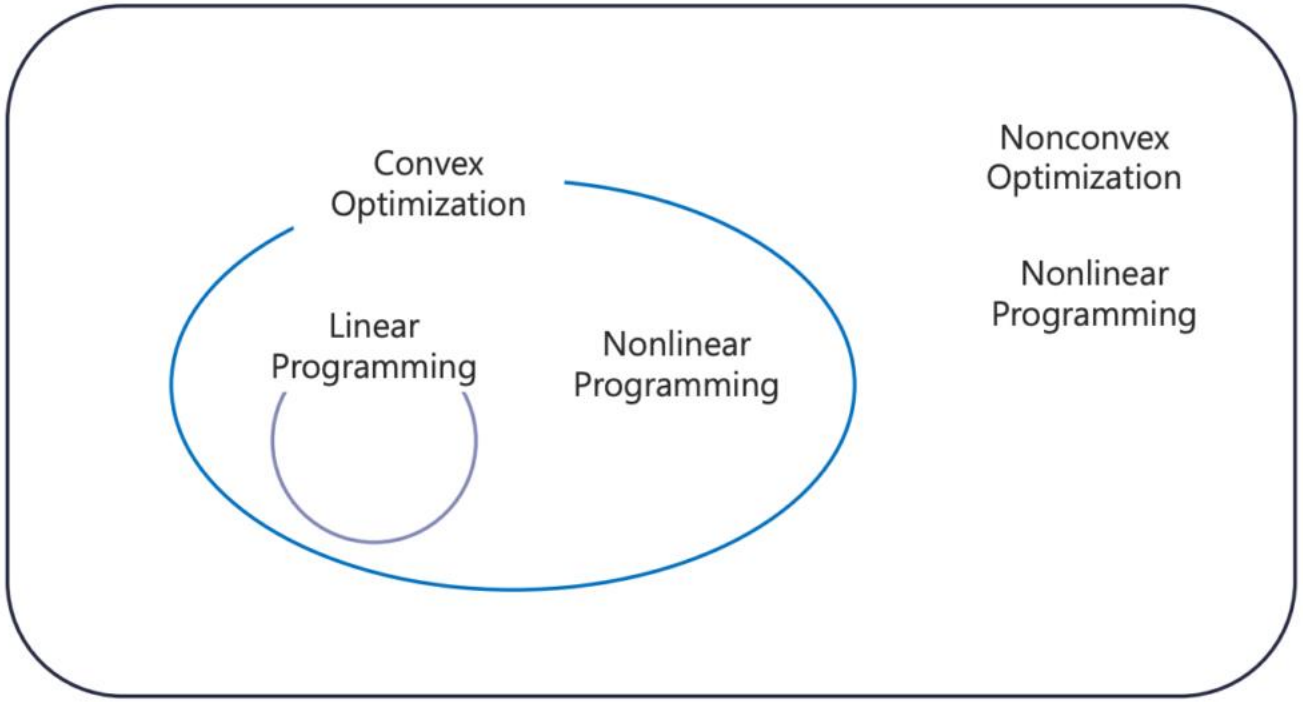
Introduction to Optimization in Deep Learning

(Sejong RCV 임근택)

Convex Optimization?

Optimization Problems



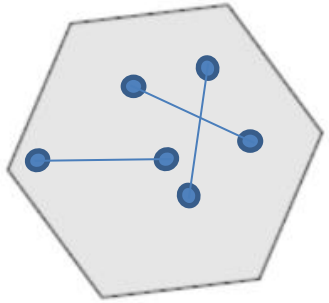
Convex Sets & Function

Convex Set

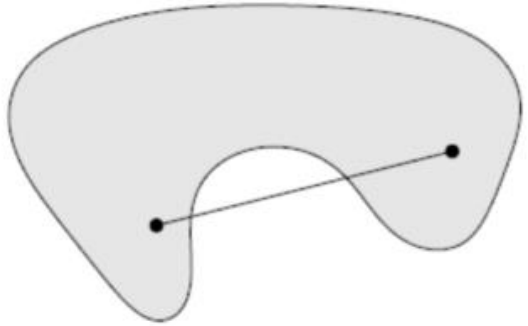
Affine Set : $x_1, x_2 \in C, \theta \in R, \theta x_1 + (1-\theta)x_2$

Convex Set : $x_1, x_2 \in C, \theta \in R, \theta x_1 + (1-\theta)x_2, 0 \leq \theta \leq 1$

- **Examples of convex and nonconvex sets**



Convex

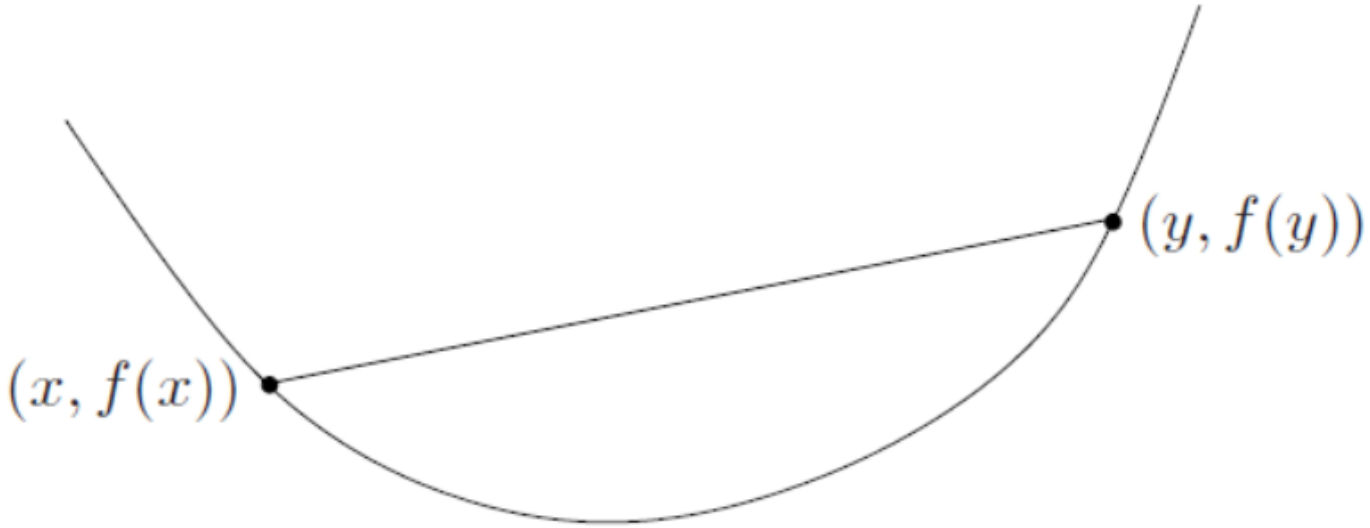


Nonconvex

Nonconvex

Convex Function

Convex Function : $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y), 0 \leq \theta \leq 1$



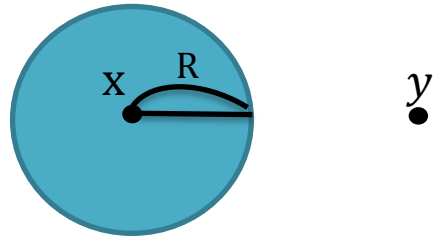
Convex Optimization

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& a_i^T x = b_i, \quad i = 1, \ldots, p,
\end{array}
$$

- The objective function must be convex
- The inequality constraint function must be convex
- The equality constraint function must be affine
- The feasible set of a convex optimization problem is convex

Global optimal = local optimal

- Suppose that x is locally optimal point and x is feasible



$$f_0(x) = \inf\{f_0(z) \mid z \; is \; feasible, \;\; \| z - x \|_2 \leq R\}$$

- Suppose that x is not globally optimal there is a feasible point y such that $\| y - x \|_2 > R$

$$f_0(y) < f_0(x)$$

- Consider the point z given by

$$z = (1-\theta)x + \theta y \;,\; \theta = \frac{R}{2 \| y - x \|_2}$$

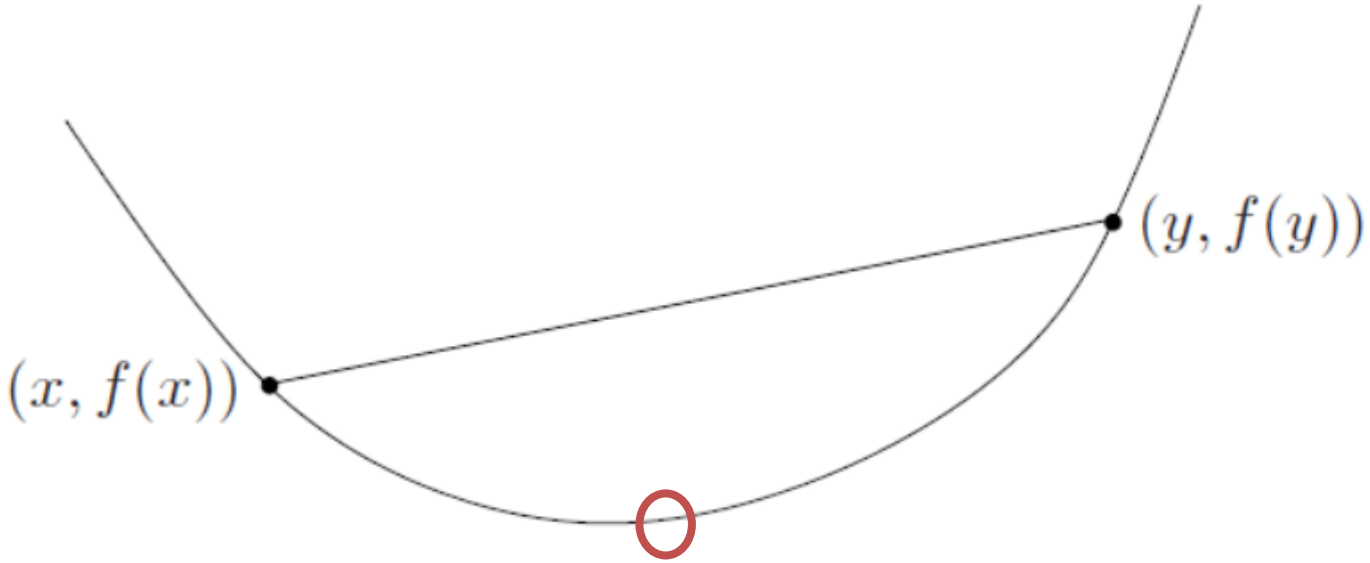
- By Convexity of the function we can have

$$f_0(z) \leq (1-\theta)\, f_0(x) + \theta f_0(y) \leq f_0(x)$$

Unconstrained Convex Optimization

Unconstrained Convex Optimization

Minimize f(x)



$$we\ just\ need\ to\ find\ \nabla_x f_0(x) = 0$$

Example : Least Square

$$minimize \ \| Ax - b \|_2^2$$

$$\| Ax - b \|_2^2 = (Ax - b)^T (Ax - b)$$

$$(Ax - b)^T (Ax - b) = (Ax)^T (Ax) - (Ax)^T b - b^T (Ax) + b^T b$$

$$(Ax)^T (Ax) - (Ax)^T b - b^T (Ax) + b^T b = x^T (A^T A) x - 2(A^T b)^T x + b^T b$$

$$f(x) = x^T (A^T A) x - 2(A^T b)^T x + b^T b$$

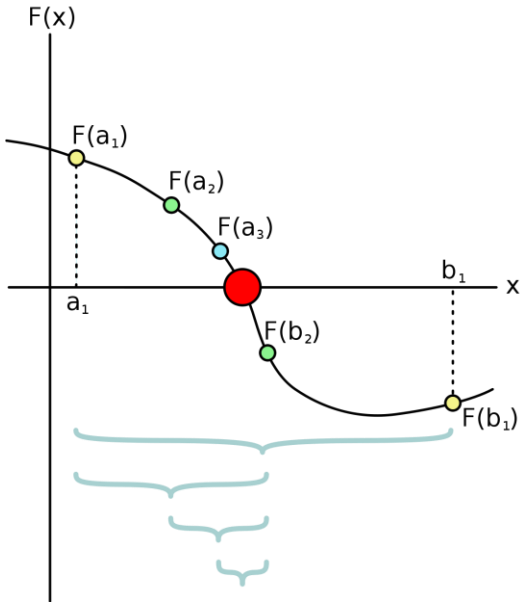
Vector Differentiation

$$\nabla f(x) = 2 (A^T A) x - 2(A^T b) = 0$$

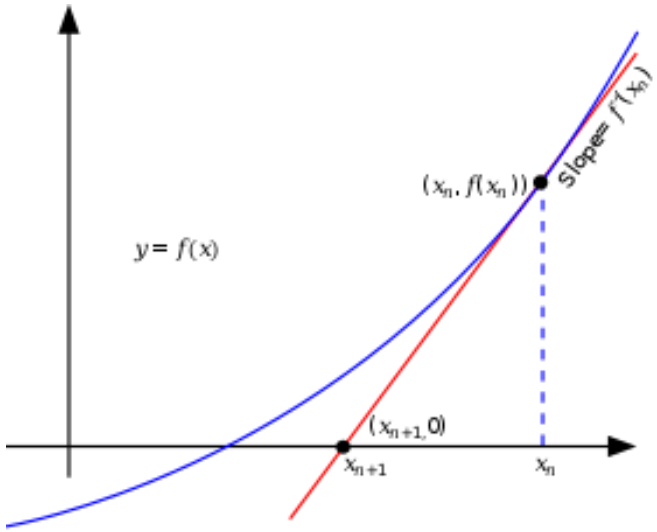
$$x = (A^T A)^{-1} A^T b$$

Unconstrained Convex Optimization (Numerical)

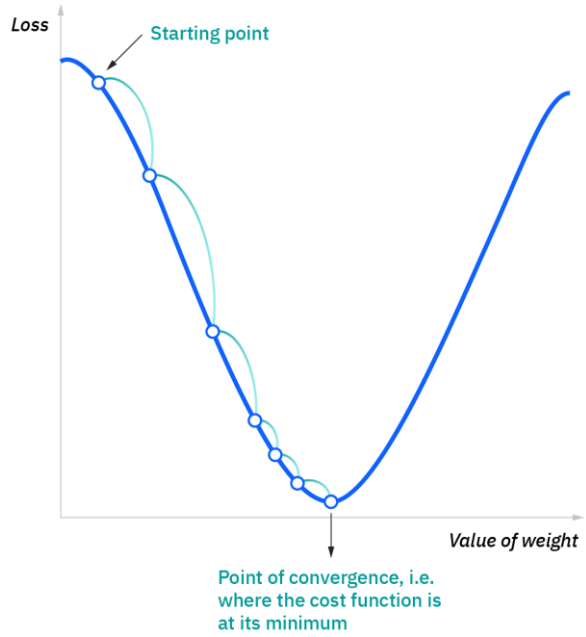
$$we\ just\ need\ to\ find\ \nabla_x f_0(x) = 0$$



Bisection method



Newton's method



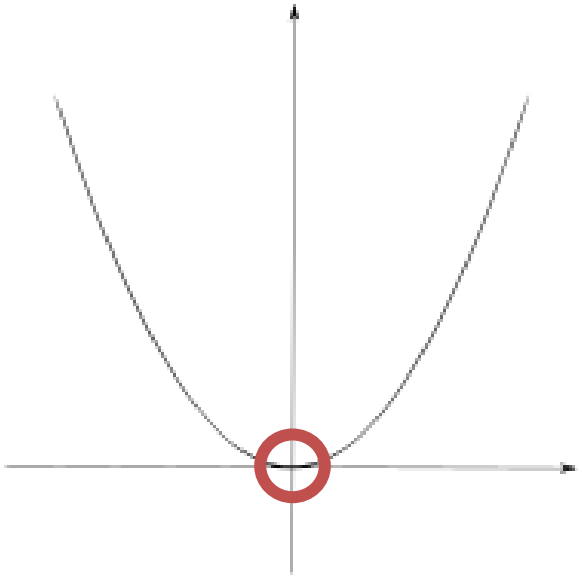
Gradient Descent

- To calculate roots of gradient, the exact solution may not be calculated.
- So, sometimes we use **numerical approach**

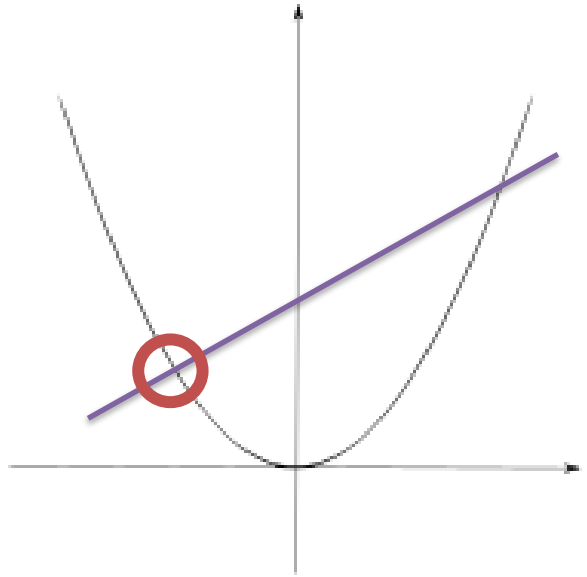
Constrained Convex Optimization

Constrained Convex Optimization

$$
\begin{array}{ll}
\text{minimize} & f_0(x) \\
\text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\
& a_i^T x = b_i, \quad i = 1, \ldots, p,
\end{array}
$$



Unconstrained



Constrained

Lagrange multiplier

An optimization problem

$$\mathcal{D} = \left(\bigcap_{i=0}^{m} \mathbf{dom}\, f_i\right) \cap \left(\bigcap_{i=1}^{p} \mathbf{dom}\, h_i\right)$$

$$\begin{array}{lll} \textbf{(P1)} \quad \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \end{array}$$

- We do not assume this problem is convex.

Lagrangian $L : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^p \to \mathbf{R}$

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

- λ_i : Lagrange multiplier associated with the *i*-th inequality constraint
- ν_i : Lagrange multiplier associated with the *i*-th equality constraint
- $\boldsymbol{\lambda}, \boldsymbol{\nu}$: Lagrange multiplier vectors or dual variables associated with **Problem (P1)**

Lagrange Dual Function

The Lagrange dual function

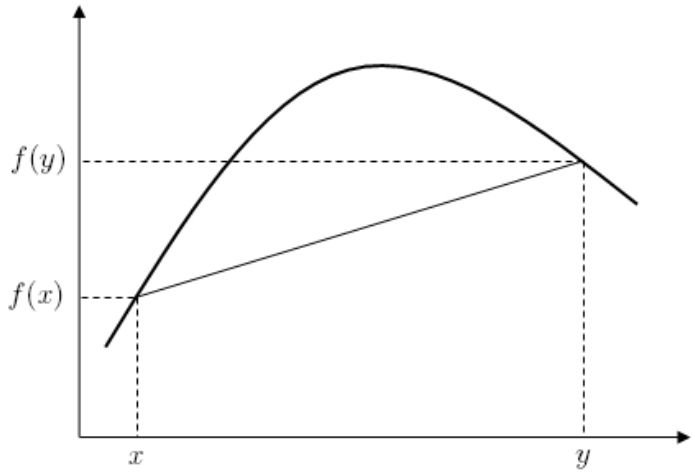
$$g(\lambda,\nu) = \inf_{x\in\mathcal{D}} L(x,\lambda,\nu) = \inf_{x\in\mathcal{D}} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right)$$

- The Lagrange dual function is always concave in (λ,ν) even when **Problem (P1)** is not convex since it is the pointwise infimum of a family of affine functions of (λ, ν).

Lower bound on the optimal value $p^\star$

- The dual function yields lower bound on the optimal value of **Problem (P1)**.
- For any $\lambda \succeq 0$ and any ν, we have

$$g(\lambda,\nu) \le p^\star.$$



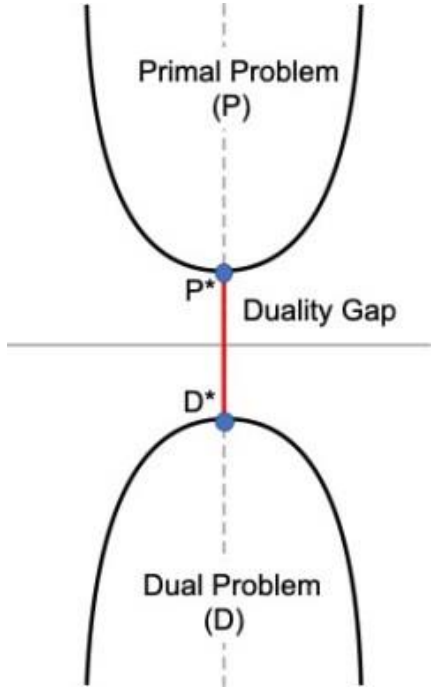
Concave function

Lagrange Dual Problem

The Lagrange dual problem

(P2) $\begin{array}{ll} \text{maximize} & g(\lambda, \nu) \\ \text{subject to} & \lambda \succeq 0 \end{array}$

- **Problem (P1)** is called the primal problem.
- The Lagrange dual problem is a convex optimization problem even when the primal problem is not convex.



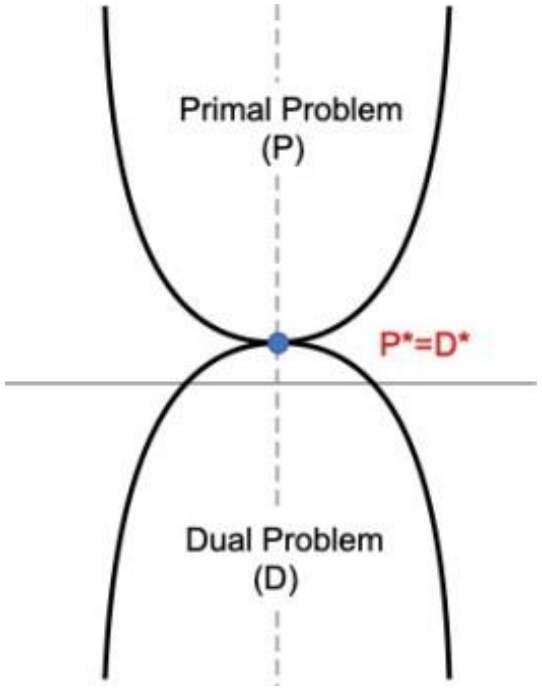
Duality Gap

Strong Duality

Strong duality

- If the equality $d^\star = p^\star$ holds, i.e., the optimal duality gap is zero, then we say that strong duality holds.
- Strong duality does not, in general, hold.
- If the primal problem is convex, we usually (but not always) have strong duality.

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$



Strong duality

Slater's condition

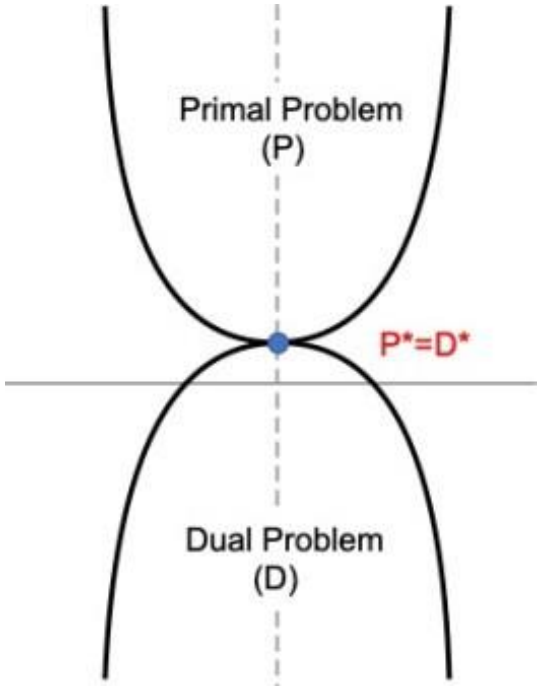
- **Slater's condition (one of constraint qualifications)**
  - There exists a *strictly feasible* $x \in \mathbf{relint}\ \mathcal{D}$, ($\mathcal{D}$: domain of the problem)

$$f_i(x) < 0, \quad i = 1, \ldots, m, \qquad Ax = b.$$

- **Weaker form of Slater's condition**
  - If the first k constraint functions $f_1, \ldots, f_k$ are affine, then strong duality holds provided the following weaker condition holds.
  - There exist an $x \in \mathbf{relint}\ \mathcal{D}$ with

$$f_i(x) \leq 0, \quad i = 1, \ldots, k, \qquad f_i(x) < 0, \quad i = k+1, \ldots, m,$$

$$Ax = b.$$



Strong duality

Karush-Kuhn-Tucker condition(KKT condition)

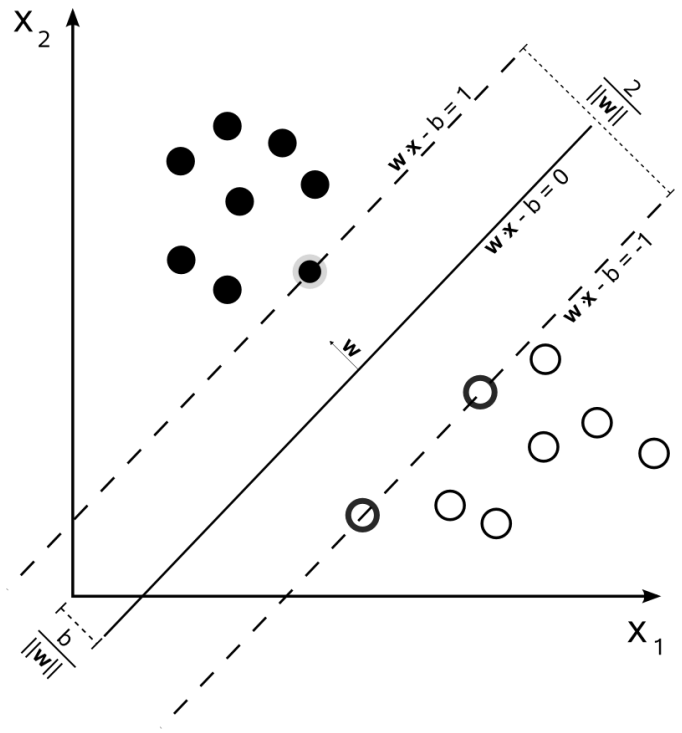
KKT optimality condition for convex problems

- When the primal problem is convex, the KKT conditions are also sufficient.
- Let $\tilde{x}, \tilde{\lambda}, \tilde{\nu}$ be any points that satisfy the KKT conditions.

$$
\begin{aligned}
f_i(\tilde{x}) &\leq 0, \quad i = 1, \ldots, m \\
h_i(\tilde{x}) &= 0, \quad i = 1, \ldots, p \\
\tilde{\lambda}_i &\geq 0, \quad i = 1, \ldots, m \\
\tilde{\lambda}_i f_i(\tilde{x}) &= 0, \quad i = 1, \ldots, m \\
\nabla f_0(\tilde{x}) + \textstyle\sum_{i=1}^{m} \tilde{\lambda}_i \nabla f_i(\tilde{x}) + \sum_{i=1}^{p} \tilde{\nu}_i \nabla h_i(\tilde{x}) &= 0
\end{aligned}
$$

- Then, $\tilde{x}$ and $(\tilde{\lambda}, \tilde{\nu})$ are primal and dual optimal, with zero duality gap.

Example : Support Vector Machine(SVM)



$$\min_{w,b} \frac{1}{2} \| w \|_2^2$$

$$subject\ to\ -\{y_i(w^T x_i + b) - 1\} \leq 0$$

$$L(w, b, \lambda) = \frac{1}{2}\|w\|_2^2 - \sum_{i=1}^{N} \lambda_i \{y_i\,(w^T x_i + b) - 1\}$$

↓ Derive Dual Function

$$g(\lambda) = \inf_{w,b} \left(\frac{1}{2} \| w \|_2^2 - \sum_{i=1}^{N} \lambda_i \{y_i\,(w^T x_i + b) - 1\} \right)$$

Example : Support Vector Machine (Dual Function)

$$\frac{\partial L(w,b,\lambda)}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \;\rightarrow\; w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\frac{\partial L(w,b,\lambda)}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \;\rightarrow\; \sum_{i=1}^{N} \lambda_i y_i = 0$$

Substitute

$$g(\lambda) = \inf_{w,b} \left(\frac{1}{2} \parallel w \parallel_2^2 - \sum_{i=1}^{N} \lambda_i \left\{ y_i \left(w^T x_i + b \right) - 1 \right\} \right)$$

$$g(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j^T$$

Example : Support Vector Machine (Dual Problem)

$$\max_{\lambda} \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j$$

$$subject\ to\ 0 \leq \lambda_i\ and\ \sum_{i=1}^{N} \lambda_i y_i = 0$$

Example : Support Vector Machine (KKT condition)

$$\min_{w,b} \frac{1}{2} \| w \|_2^2$$

$$subject\ to\ -\{y_i(w^T x_i + b) - 1\} \leq 0$$

$$Stationarity\ :\ \nabla_x f_0(x) + \sum_{i=1}^{N} \lambda_i \nabla_x f_i(x) = 0$$

$$Dual\ feasibility\ :\ 0 \leq \lambda_i\ i = 1, ..., N$$

$$\frac{\partial L(w,b,\lambda)}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \ \rightarrow\ w = \sum_{i=1}^{N} \lambda_i y_i x_i$$

$$\frac{\partial L(w,b,\lambda)}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \ \rightarrow\ \sum_{i=1}^{N} \lambda_i y_i = 0$$

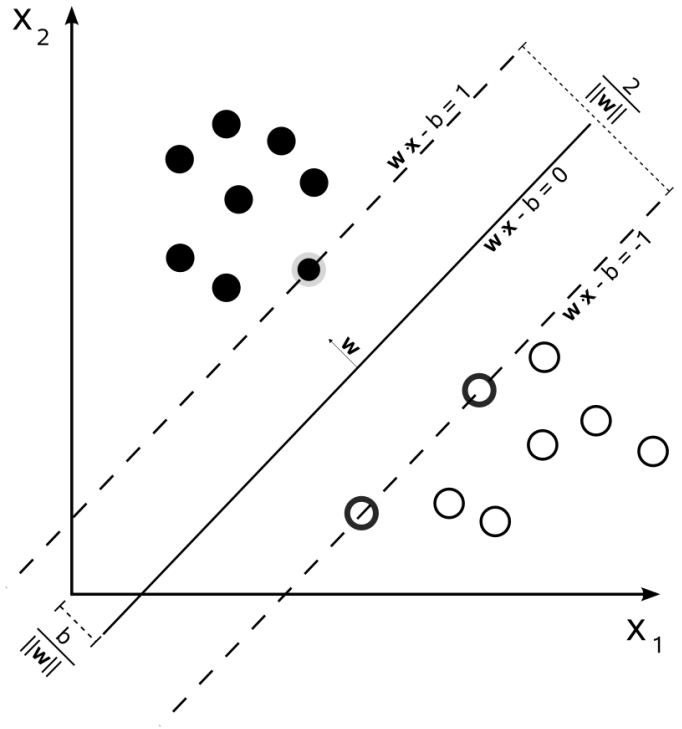
$$Primal\ feasibility\ :\ 1 - y_i(w^T x_i + b) \leq 0$$

$$complmentary\ slackness\ :\ \lambda_i \left(y_i(w^T x_i + b) - 1\right) = 0$$

Example : Support Vector Machine (KKT condition)

$$Primal\ feasibility\ :\ 1 - y_i(w^T x_i + b) \leq 0$$

$$complmentary\ slackness\ :\ \lambda_i \left(y_i(w^T x_i + b) - 1\right) = 0$$



$$if\ \lambda_i \neq 0 \longrightarrow\ y_i\left(w^T x_i + b\right) = 1\ (x_i\ is\ support\ vector)$$

$$w = \sum_{i=1}^{N} \lambda_i y_i x_i \qquad b = \frac{1}{y_i} - w^T x_i$$

Gradient Descent

Loss function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^{N}$$

Where x_i is an vector (image) and
y_i is an integer-valued label

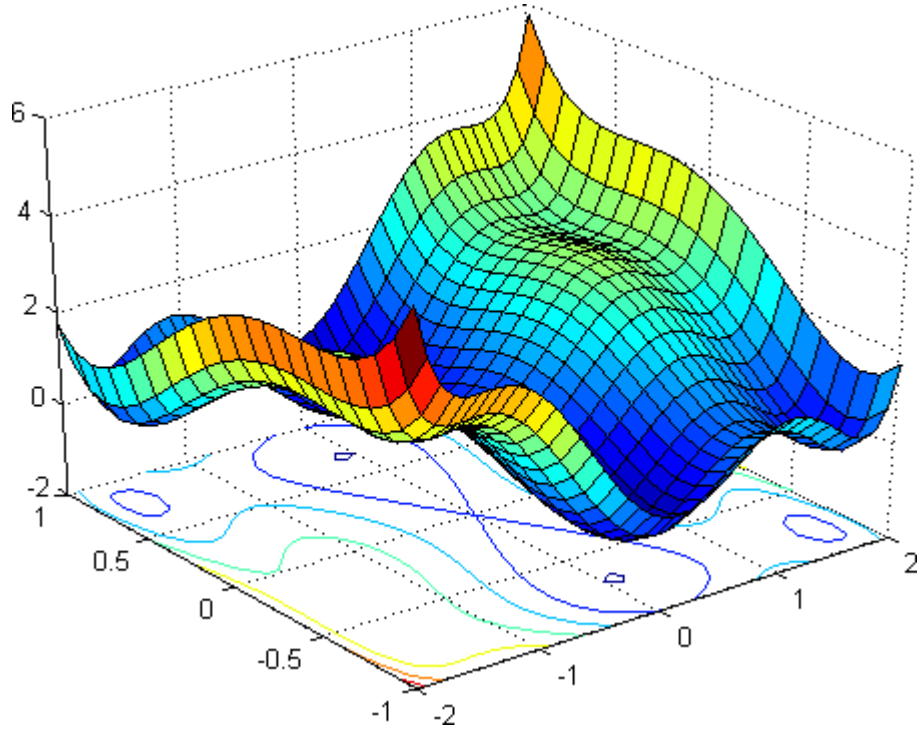
Loss for a single example is

$$L_i = L(f(x_i, W), y_i)$$

Loss for the dataset is average of per-example losses:

분류기의 Loss ➔

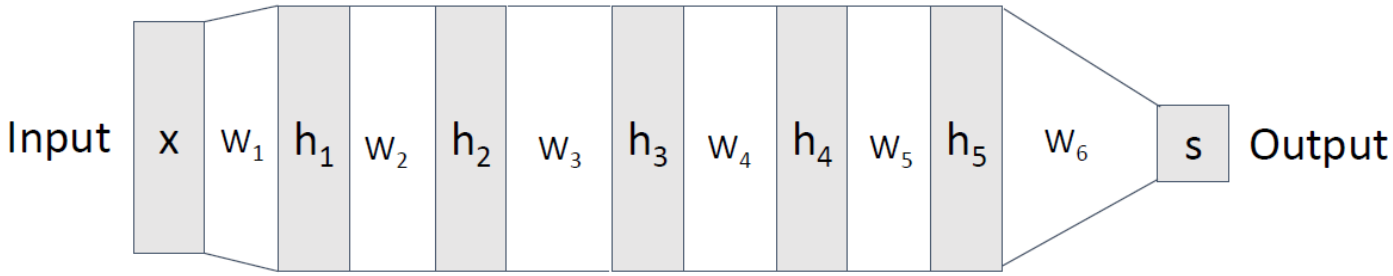
$$\phi(W) = \frac{1}{N}\sum_{i=1}^{N} L(f(x_i, W), y_i)$$

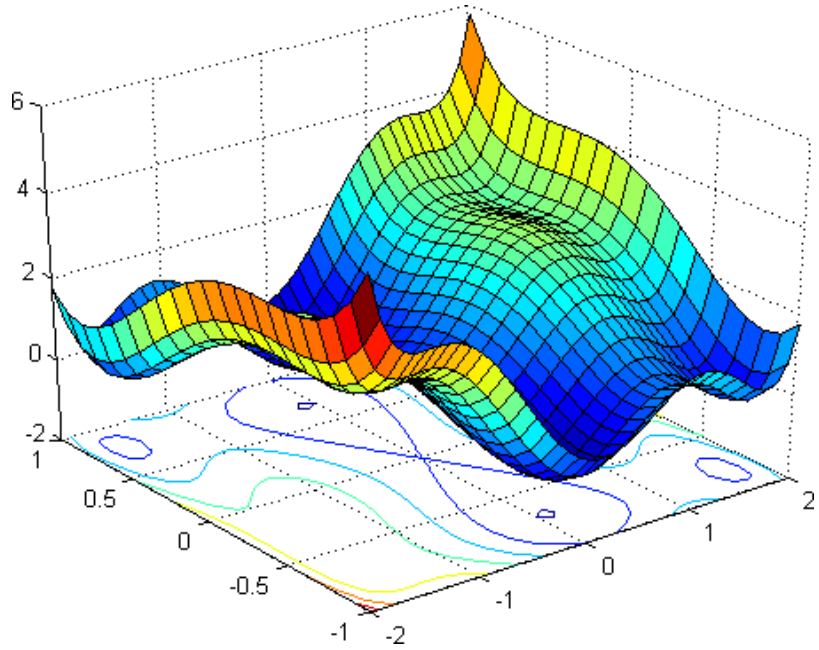


Gradient Descent

Goal:
find the **w** minimizing some loss function L.

$$\arg\min_{\boldsymbol{w}\in R^N} L(\boldsymbol{w})$$





Gradient Descent

Method: at each step, move in direction of negative gradient

```
w0 = initialize()                    #initialize
for iter in range(numIters):
    g = ∇_w L(w)                     # eval gradient
    w = w + -stepsize*g              # update w
return w
```

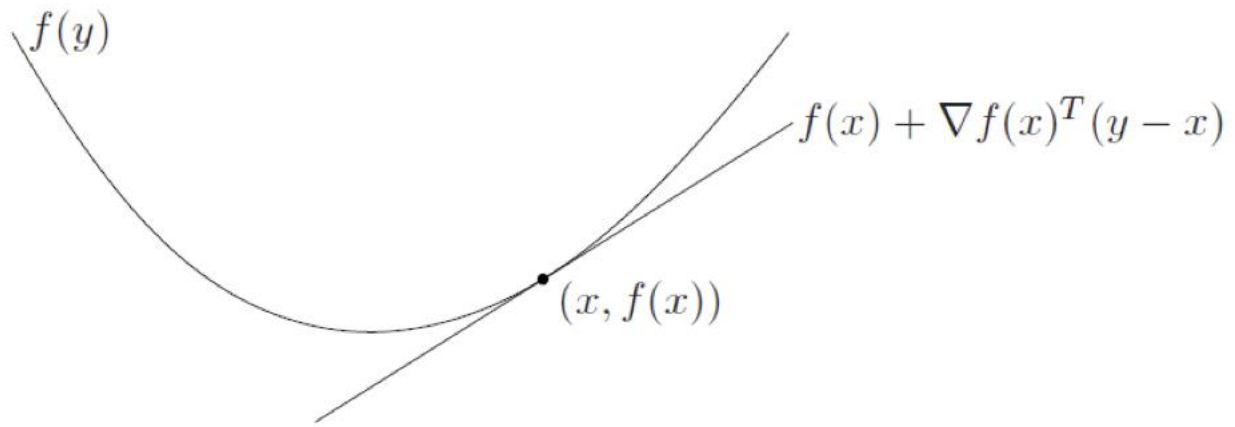
Learning rate

Convergence Analysis of Gradient Descent

Differentiable Convex Function

Lemma 1.2. Let f be twice continuously differentiable. Then f is convex if either of the following hold

- **First-order condition** $f(x) + \nabla f(x)^T (y - x) \leq f(y)$
- **Second-order condition** $0 \preceq \nabla^2 f(x)$



L-Smoothness

A differential function f is said to be L–smooth if its gradients are Lipschitz continuous, that is

$$\|\nabla f(x) - \nabla f(y)\| \quad \leq \quad L\|x - y\|. \tag{9}$$

Lemma 1.3. Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a twice differentiable function. If f is L–smooth then the following holds

$$\left\langle \nabla^2 f(x)v, v \right\rangle \quad \leq \quad L\|v\|_2^2, \quad \forall x, v \in \mathbb{R}^d, \tag{10}$$

$$f(y) \quad \leq \quad f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|_2^2. \tag{11}$$

Convergence for convex and smooth functions

6.1.1 Convergence of gradient descent with fixed step size

Theorem 6.1 *Suppose the function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and that its gradient is Lipschitz continuous with constant $L > 0$, i.e. we have that $\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2$ for any x, y. Then if we run gradient descent for k iterations with a fixed step size $t \leq 1/L$, it will yield a solution $f^{(k)}$ which satisfies*

$$f(x^{(k)}) - f(x^*) \leq \frac{\|x^{(0)} - x^*\|_2^2}{2tk}, \tag{6.1}$$

where $f(x^*)$ is the optimal value. Intuitively, this means that gradient descent is guaranteed to converge and that it converges with rate $O(1/k)$.

Convergence for convex and smooth functions

$$f(y) = f(x) + \nabla f(x)^T + \frac{1}{2} \nabla^2 f(x) \parallel y - x \parallel_2^2$$

$$f(y) \leq f(x) + \nabla f(x)^T + \frac{1}{2} L \parallel y - x \parallel_2^2$$

Now let's plug in the gradient descent update by $x^+ = x - t \nabla f(x)$

$$f(x^+) \leq f(x) + \nabla f(x)^T (x - t \nabla f(x) - x) + \frac{1}{2} L \parallel x - t \nabla f(x) - x \parallel_2^2$$

$$f(x^+) \leq f(x) - \nabla f(x)^T (t \nabla f(x)) + \frac{1}{2} L \parallel t \nabla f(x) \parallel_2^2$$

$$f(x^+) \leq f(x) - t \parallel \nabla f(x) \parallel_2^2 + \frac{1}{2} L t^2 \parallel \nabla f(x) \parallel_2^2$$

$$f(x^+) \leq f(x) - (1 - \frac{1}{2} L t) t \parallel \nabla f(x) \parallel_2^2$$

Convergence for convex and smooth functions

$$f(x^+) \leq f(x) - (1 - \frac{1}{2}Lt)t \parallel \nabla f(x) \parallel_2^2$$

$$\textit{Using } t \leq 1/L, \textit{ we know that } -1(1 - \frac{1}{2}Lt) \leq \frac{1}{2}L(1/L) - 1 = -\frac{1}{2}$$

$$f(x^+) \leq f(x) - \frac{1}{2}t \parallel \nabla f(x) \parallel_2^2$$

$$\textit{Next we can bound } f(x^+) \textit{ in terms of } f(x^*). \textit{ Since } f \textit{ is convex, we can write}$$

$$f(x) + \nabla f(x)^T(x^* - x) \leq f(x^*)$$

$$f(x) \leq f(x^*) + \nabla f(x)^T(x - x^*)$$

$$f(x^+) \leq f(x^*) + \nabla f(x)^T(x - x^*) - \frac{1}{2}t \parallel \nabla f(x) \parallel_2^2$$

Convergence for convex and smooth functions

$$f(x^+) \leq f(x^*) + \nabla f(x)^T(x - x^*) - \frac{1}{2}t \parallel \nabla f(x) \parallel_2^2$$

$$f(x^+) - f(x^*) \leq \frac{1}{2t}(2t \nabla f(x)^T(x - x^*) - t^2 \parallel \nabla f(x) \parallel_2^2)$$

$$f(x^+) - f(x^*) \leq \frac{1}{2t}(2t \nabla f(x)^T(x - x^*) - t^2 \parallel \nabla f(x) \parallel_2^2 - \parallel x - x^* \parallel_2^2 + \parallel x - x^* \parallel_2^2)$$

$$\textit{Since the square of } \parallel x - t \nabla f(x) - x^* \parallel_2^2 \textit{ yields } \parallel x - x^* \parallel_2^2 - 2t \nabla f(x)^T(x - x^*) + t^2 \parallel \nabla f(x) \parallel_2^2$$

$$f(x^+) - f(x^*) \leq \frac{1}{2t}(\parallel x - x^* \parallel - \parallel x - t \nabla f(x) - x^* \parallel_2^2)$$

$$\textit{Notice that by definition we have } x^+ = x - t \nabla f(x)$$

$$f(x^+) - f(x^*) \leq \frac{1}{2t}(\parallel x - x^* \parallel_2^2 - \parallel x^+ - x^* \parallel_2^2)$$

Convergence for convex and smooth functions

$$f(x^+) - f(x^*) \leq \frac{1}{2t}(\| x - x^* \|_2^2 - \| x^+ - x^* \|_2^2)$$

This inequality holds for x^+ *on every iteration of gradient desecent. Summing over iterations, we get :*

$$\sum_{i=1}^{k} f(x^{(i)}) - f(x^*) \leq \sum_{i=1}^{k} \frac{1}{2t}(\| x^{(i-1)} - x^* \|_2^2 - \| x^{(i)} - x^* \|_2^2)$$

$$\sum_{i=1}^{k} f(x^{(i)}) - f(x^*) \leq \frac{1}{2t}(\| x^{(0)} - x^* \|_2^2 - \| x^{(k)} - x^* \|_2^2)$$

$$\sum_{i=1}^{k} f(x^{(i)}) - f(x^*) \leq \frac{1}{2t}(\| x^{(0)} - x^* \|_2^2)$$

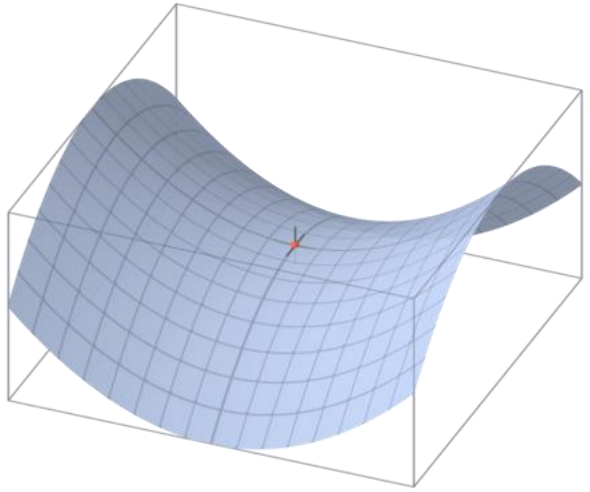
$$f(x^{(k)}) - f(x^*) \leq \frac{1}{k}\sum_{1}^{k} f(x^{(i)}) - f(x^*) \leq \frac{\| x^{(0)} - x^* \|_2^2}{2tk}$$

QED

Advanced Gradient Descent

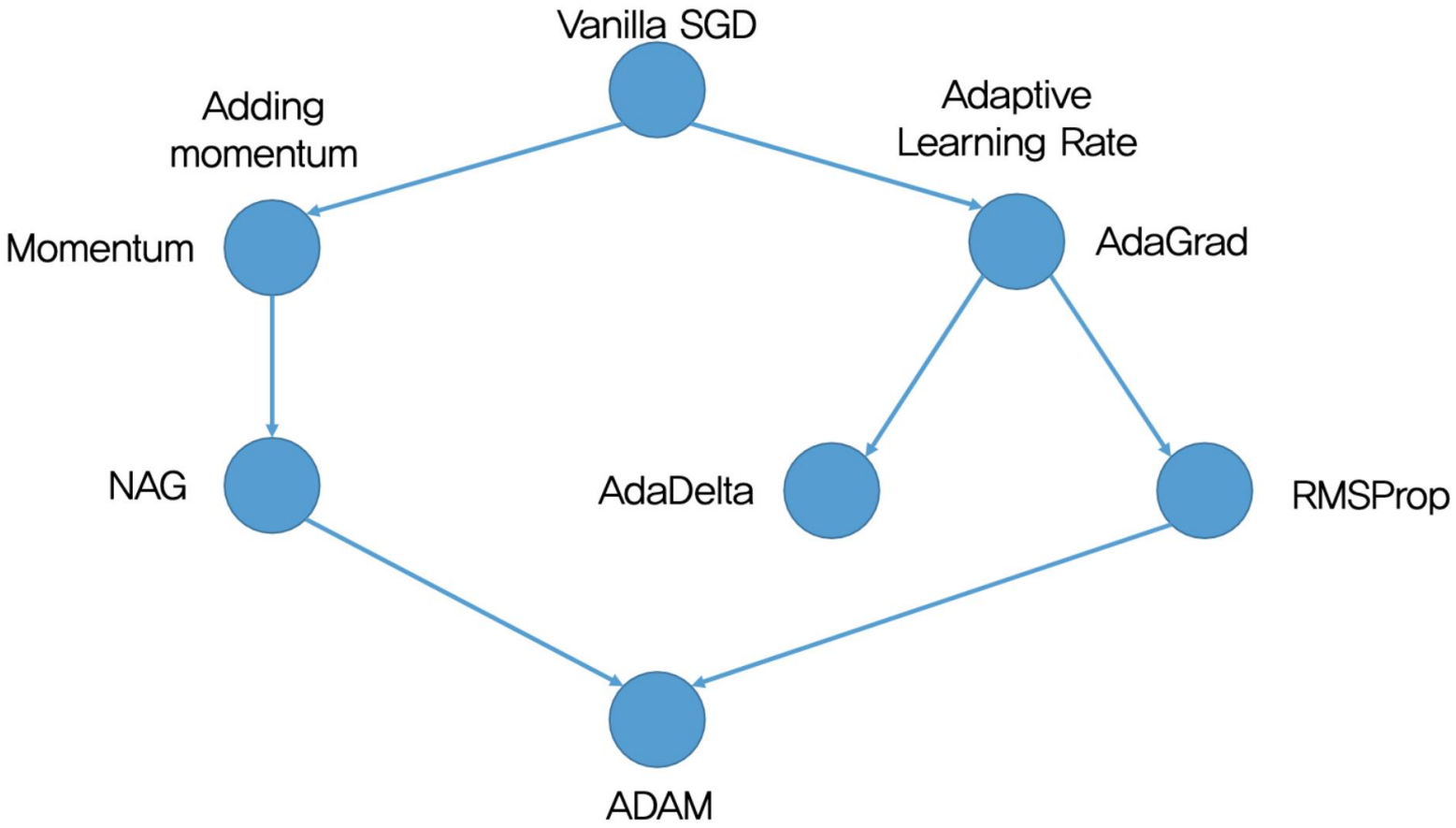
Challenges of Gradient Descent in Non-Convex

- Choosing a proper learning rate can be difficult
- The same learning rate applies to all parameter update
- It is difficult to avoid getting trapped in their numerous local minima, saddle point



Saddle Point

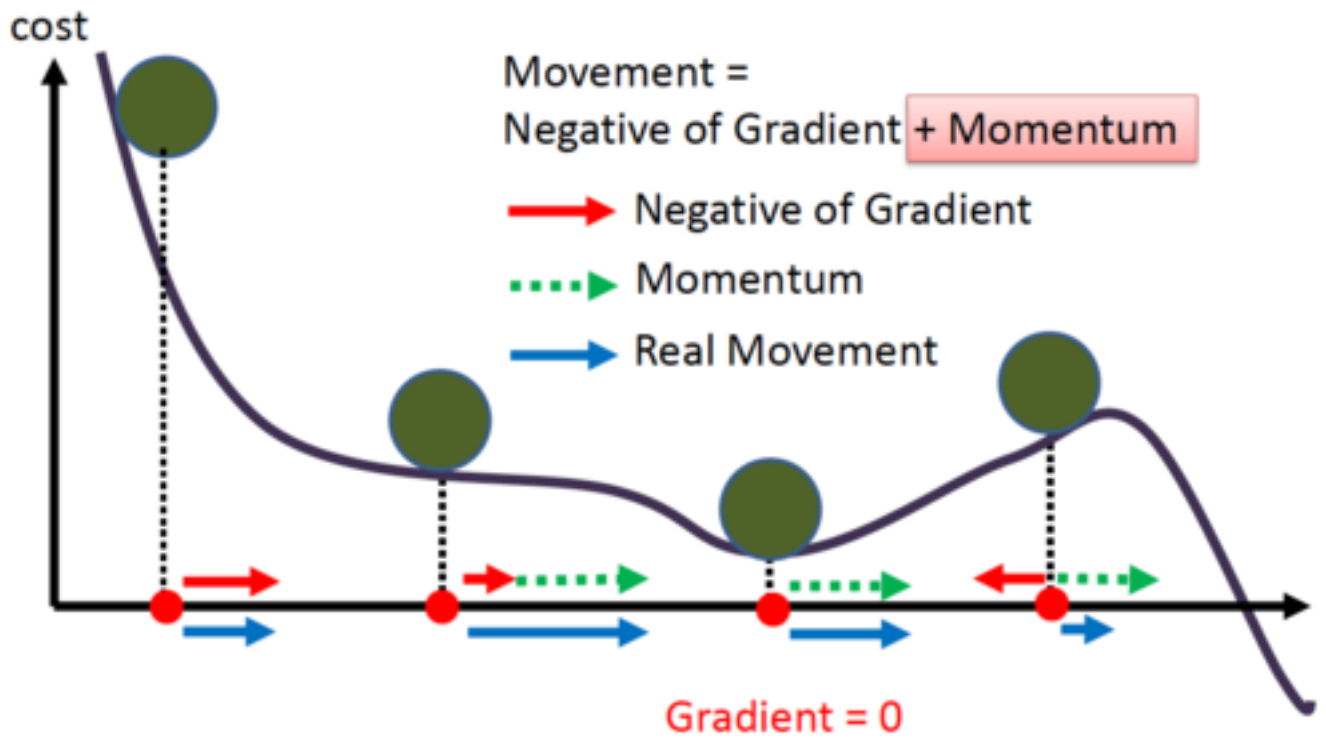
Overview of GD algorithms



Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta)$$

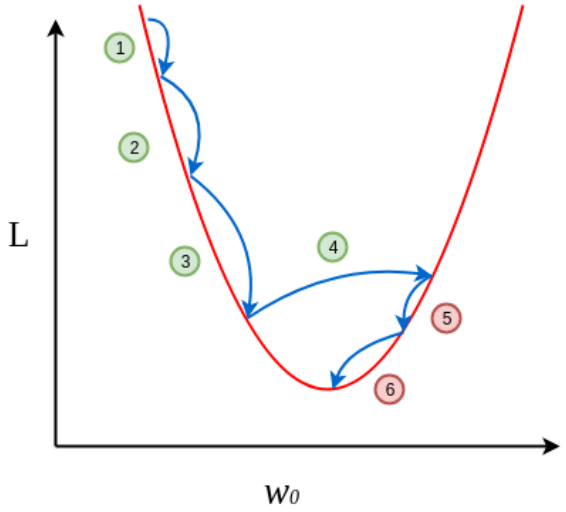
$$\theta = \theta - v_t$$



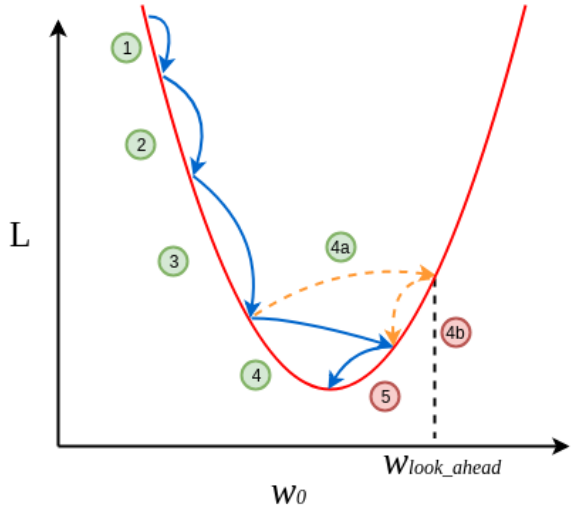
Nesterov Momentum

$$v_t = \gamma v_{t-1} + \eta \nabla J_\theta(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$



(a) Momentum-Based Gradient Descent



(b) Nesterov Accelerated Gradient Descent

○ $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Positive(+)}$

○ $\Longrightarrow \frac{\partial L}{\partial w_0} = \frac{Negative(-)}{Negative(-)}$

AdaGrad

$$G_t = G_{t-1} + (\nabla_\theta J(\theta))^2$$

$$G_t = [G_t^{(1)}, G_t^{(2)}, ..., G_t^{(n-1)}, G_t^{(n)}]$$

$$G_t = [\sum_{\tilde{t}=0}^{t} (\frac{\partial J(\theta_{\tilde{t}})}{\partial \theta_{\tilde{t}}^{(1)}})^2, ..., \sum_{\tilde{t}=0}^{t} (\frac{\partial J(\theta_{\tilde{t}})}{\partial \theta_{\tilde{t}}^{(n)}})^2]$$

$$\theta_t - \frac{\eta}{\sqrt{G_t} + \varepsilon} \odot \nabla_{\theta_t} J(\theta_t)$$

- Large update of a parameter -> small learning rate
- Small update of a parameter -> large learning rate

RMSprop

$$G_t = \gamma G_{t-1} + (1-\gamma)\left(\nabla_\theta J(\theta)\right)^2$$

$$\theta_t - \frac{\eta}{\sqrt{G_t + \varepsilon}} \odot \nabla_{\theta_t} J(\theta_t)$$

Algorithm 8.5 The RMSProp algorithm

Require: Global learning rate ϵ, decay rate ρ.
Require: Initial parameter $\boldsymbol{\theta}$
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
Initialize accumulation variables $\boldsymbol{r} = 0$
while stopping criterion not met **do**
Sample a minibatch of m examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1-\rho)\boldsymbol{g} \odot \boldsymbol{g}$
Compute parameter update: $\Delta\boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta+\boldsymbol{r}}}$ applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end while

Adam

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_\theta J(\theta)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left(\nabla_\theta J(\theta)\right)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \varepsilon}} \hat{m}_t$$

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Machine learning is just Mathematics