ST456 Deep Learning

Lecture 3

Optimization Algorithms

Milan Vojnovic

https://github.com/lse-st456/lectures2022

Topics of this lecture

- Empirical loss function minimization
- Gradient descent algorithm
- Stochastic gradient descent algorithm
- More on gradient descent algorithm

Loss minimization and gradient descent

Empirical risk minimization

- Model:

$$\underbrace{y}_{\text{output}} = h_{\underbrace{w}_{\text{parameter}}}(\underbrace{x}_{\text{input}})$$

- The expected loss function:

$$f(w) = \mathbf{E}_{(x,y)\sim D}\left[\underbrace{\ell}_{\text{a given loss function}}\big(y, h_w(x)\big)\right]$$

- Empirical risk minimization:

$$f(w) = \frac{1}{m}\sum_{i=1}^{m} \ell\big(y_i, h_w(x_i)\big) + \underbrace{\lambda\phi(w)}_{\text{regularization term, } \lambda \geq 0}$$

Loss functions for binary classification

True class y (0 or 1), $\hat{y}$ prediction (in [0,1])

- Misclassification loss (negative accuracy):

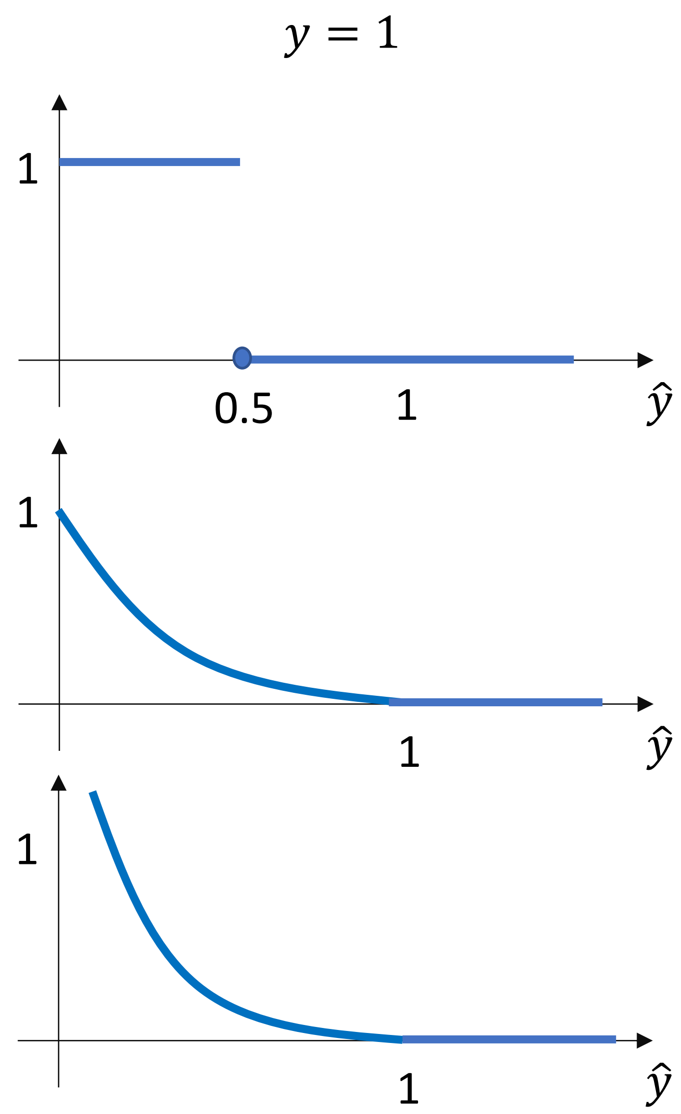
$$\ell(y, \hat{y}) = 1_{\left\{\left(y-\frac{1}{2}\right)\left(\hat{y}-\frac{1}{2}\right)<0\right\}}$$

- Squared loss:

$$\ell(y, \hat{y}) = (\hat{y} - y)^2$$

- Cross-entropy loss:

$$\ell(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$



Loss functions for multi-class case

True output: $\boldsymbol{y} \in \{0,1\}^d$ such that $\sum_{i=1}^{d} y_i = 1$

Prediction distribution: $\hat{\boldsymbol{y}}$

- Squared loss: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2 = \sum_{i=1}^{d} (y_i - \hat{y}_i)^2$
- Cross-entropy: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{i=1}^{d} y_i \log(\hat{y}_i)$

- Cross-entropy interpretations:
  - Penalty $\log(1/\hat{y}_i)$ if y_i is the true class
  - Negative log-likelihood function
  - KL-divergence: $\mathrm{KL}(\boldsymbol{y}||\hat{\boldsymbol{y}}) = \sum_{i=1}^{d} y_i \log\left(\frac{y_i}{\hat{y}_i}\right) = -\underbrace{H(\boldsymbol{y})} + \ell(\boldsymbol{y}, \hat{\boldsymbol{y}})$

Entropy $H(\boldsymbol{y}) = -\sum_{i=1}^{d} y_i \log(y_i)$

$H(\boldsymbol{p}) = 0$ when $\boldsymbol{p}$ has all mass on one element
$H(\boldsymbol{p}) = \log(d)$ maximum value, when $\boldsymbol{p}$ is uniform distribution over a set of d elements

Regularization

- Regularization is used to mitigate overfitting (improve generalization)

- Common examples:
  - Lasso (or L_1) regularization: $\phi(\boldsymbol{w}) = \|\boldsymbol{w}\|_1$
  - Ridge (or L_2) regularization: $\phi(\boldsymbol{w}) = \|\boldsymbol{w}\|_2$

- Lasso regularization favors parameter vectors with zero elements (feature selection)

- Note that $\|\boldsymbol{w}\|_1$ is not differentiable for all $\boldsymbol{w}$ while $\|\boldsymbol{w}\|_2$ is

- Exercise: revisit the binary classification example from the last lecture for the cross-entropy loss function with Lasso, and then with Ridge regularization
  - Recall $\mathbf{Pr}[y_i = 1] = 1 - \mathbf{Pr}[y_i = 0] = p_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = a(\boldsymbol{x}_i^\top \boldsymbol{w} + b)$
  - $f_{\mathrm{CE}}(\boldsymbol{\theta}) = -\sum_{i=1}^{m} y_i \log\big(p_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\big) + (1 - y_i) \log(1 - p_{\boldsymbol{\theta}}(\boldsymbol{x}_i))$

Linear algebra refresher

- Gradient vector: Assume $f: \mathbf{R}^n \to \mathbf{R}$ is differentiable. Then, the gradient vector $\nabla f(\boldsymbol{x})$ at $\boldsymbol{x}$ is defined by

$$\nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(\boldsymbol{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\boldsymbol{x}) \end{pmatrix}$$

- Hessian matrix: Assume $f: \mathbf{R}^n \to \mathbf{R}$ is twice-differentiable. Then, the Hessian matrix $\nabla^2 f(\boldsymbol{x})$ at $\boldsymbol{x}$ is defined by

$$\nabla^2 f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2}{\partial^2 x_1} f(\boldsymbol{x}) & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n} f(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1} f(\boldsymbol{x}) & \cdots & \frac{\partial^2}{\partial^2 x_n} f(\boldsymbol{x}) \end{pmatrix}$$

Linear algebra refresher (cont'd)

- Eigenvalues: For any $n \times n$ matrix $\boldsymbol{A}$, λ is an eigenvalue with corresponding eigenvector $\boldsymbol{x}$ if the following holds

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

- Positive-definite matrices: A symmetric real matrix $\boldsymbol{A}$ is said to be positive-definite if

$$\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} > 0 \text{ for all non-zero } \boldsymbol{x} \in \mathbf{R}^n$$

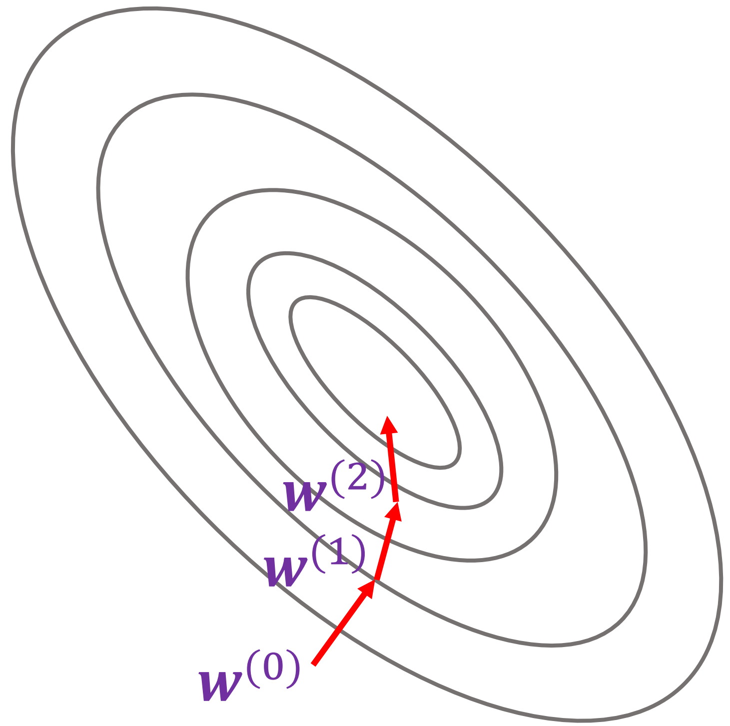
- Positive semi-definite matrices: A symmetric real matrix $\boldsymbol{A}$ is said to be positive-definite if

$$\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} \geq 0 \text{ for all } \boldsymbol{x} \in \mathbf{R}^n$$

- "Negative-definite" and "negative semi-definite" are defined analogously with reverse signs

- Fact 1: $\boldsymbol{A}$ positive-definite $\Leftrightarrow$ all eigenvalues of $\boldsymbol{A}$ are positive ($\lambda_1, \ldots, \lambda_n > 0$)
- Fact 2: $\boldsymbol{A}$ positive semi-definite $\Leftrightarrow$ all eigenvalues of $\boldsymbol{A}$ are non-negative ($\lambda_1, \ldots, \lambda_n \geq 0$)

Gradient descent algorithm



- Gradient descent algorithm update:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \underbrace{\eta^{(t)}}_{\text{step size}} \boldsymbol{B}^{(t)} \nabla f\left(\boldsymbol{w}^{(t)}\right)$$

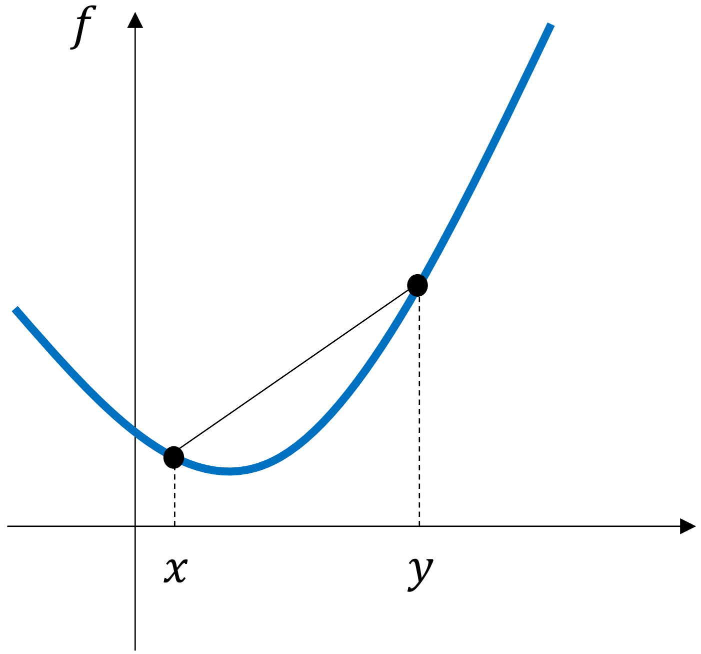
- Common step sizes:
  - A small positive constant
  - A decreasing sequence such that $\sum_{t=1}^{\infty} \eta^{(t)} = \infty$ and $\sum_{t=1}^{\infty} \left(\eta^{(t)}\right)^2$ is finite

- Common choices of $\boldsymbol{B}^{(t)}$:
  - Standard gradient descent: $\boldsymbol{B}^{(t)} = I$
  - Other: Newton's method $\boldsymbol{B}^{(t)} = \nabla^2 f\left(\boldsymbol{w}^{(t)}\right)^{-1}$, AdaGrad, RMSProp, Adam, …

Convex functions

- A function $f: \mathbf{R}^n \to \infty$ is said to be convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \text{ for all } \lambda \in [0,1]$$



"every chord from x to y lies on or above the function"

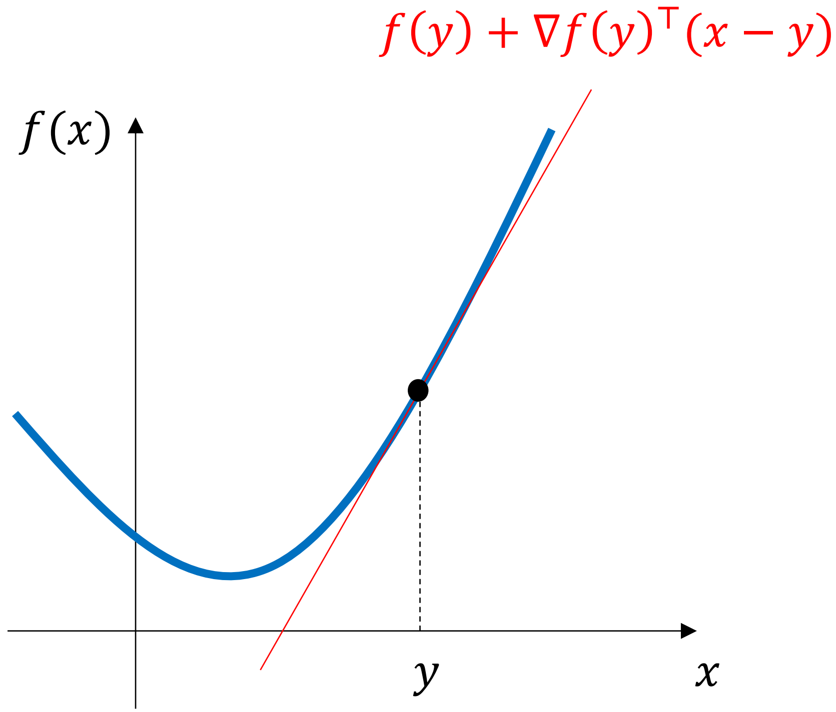
- f is said to be strictly convex if strict inequality holds for all $\lambda \in (0,1)$

Convex functions (cont'd)

- If f is differentiable, then f is convex iff for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$

$$f(\boldsymbol{x}) \geq f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^{\top}(\boldsymbol{x} - \boldsymbol{y})$$

"function is lower bounded by its tangents"



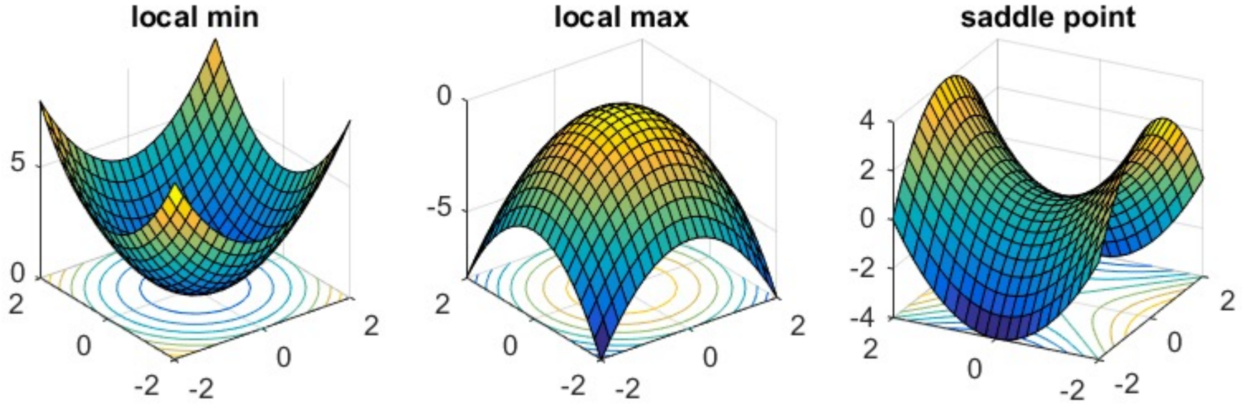
- If f is twice-differentiable, then
  - f is convex iff its Hessian $\nabla^2 f(\boldsymbol{x})$ at $\boldsymbol{x}$ is positive semidefinite for all $\boldsymbol{x} \in \mathbf{R}^n$
  - f is strictly convex iff its Hessian $\nabla^2 f(\boldsymbol{x})$ at $\boldsymbol{x}$ is positive definite for all $\boldsymbol{x} \in \mathbf{R}^n$

Note: a matrix is

- positive semidefinite if all its eigenvalues are real non-negative
- positive definite if all its eigenvalues are real and greater than zero

Non-convex loss functions

- Non-convex functions are neither convex nor concave
- Non-convex functions may have
  - Multiple local minima
  - Local minima that are globally suboptimal
  - Saddle points



- Non-convex functions are much more challenging for optimization
- Loss functions of neural networks are typically non-convex functions !

Convergence properties of gradient descent

- Gradient descent algorithm has certain convergence guarantees depending on the properties of the loss function

- For convex loss functions, gradient descent algorithm has faster convergence guarantees, under certain conditions on the loss function (smoothness and strong convexity)

- For non-convex loss functions, gradient descent algorithm has some convergence guarantees

- We explore this in the next slides

Smooth functions

- A function f is said to be β-smooth if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|$$

- Example: quadratic function $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x}$

$$\nabla f(\boldsymbol{x}) = \boldsymbol{H}\boldsymbol{x}$$

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| = \|\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y})\| \leq \underbrace{\|\boldsymbol{H}\|}_{\beta} \|\boldsymbol{x} - \boldsymbol{y}\|$$

Smooth functions (cont'd)

- If f is a β-smooth function, then [Bubeck L 3.4 or Nesterov L 1.2.3]

  (S0) $|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \nabla f(\boldsymbol{x})^\top(\boldsymbol{y} - \boldsymbol{x})| \leq \frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2$

- Equivalent conditions for f convex and β-smooth [Nesterov Thm 2.1.5]:

  (S1) $0 \leq f(\boldsymbol{y}) - f(\boldsymbol{x}) - \nabla f(\boldsymbol{x})^\top(\boldsymbol{y} - \boldsymbol{x}) \leq \frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2$

  (S2) $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top(\boldsymbol{y} - \boldsymbol{x}) + \frac{1}{2\beta}\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^2$

  (S3) $\left(\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\right)^\top(\boldsymbol{y} - \boldsymbol{x}) \geq \frac{1}{\beta}\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^2$

Gradient descent: smooth convex functions

- Consider gradient descent algorithm with a constant step size:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \nabla f(\boldsymbol{x}^{(t)})$$

- Thm 1: For any β-smooth convex function f, the gradient descent algorithm with constant step size $0 < \eta < 2/\beta$ converges to a global minimizer of f

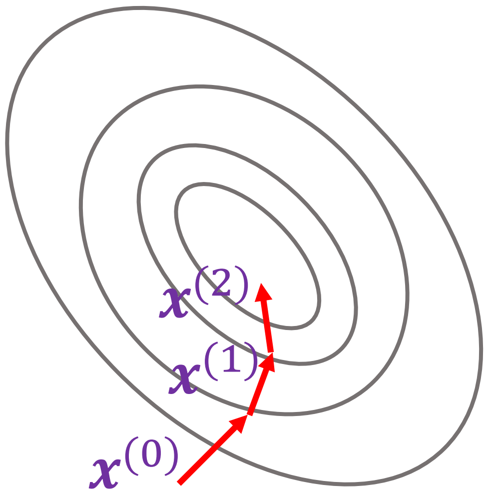
In fact, for any initial point $\boldsymbol{x}^{(0)}$ and an optimum point $\boldsymbol{x}^*$ such that $\left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\right\| \le R$,

$$f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*) \le \underbrace{\frac{2R^2}{2R^2 + \left(f(\boldsymbol{x}^{(0)}) - f(\boldsymbol{x}^*)\right)\eta(2-\beta\eta)t}}_{= O\left(\frac{1}{t}\right)} \left(f\left(\boldsymbol{x}^{(0)}\right) - f(\boldsymbol{x}^*)\right)$$

Proof sketch – Thm 1

- Claim 1: If $0 < \eta < 2/\beta$, then the Euclidean distance between $\boldsymbol{x}^{(t)}$ and $\boldsymbol{x}^*$ decreases with the number of iterations t, i.e.

$$\left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\right\| < \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\| \text{ whenever } \left\|\nabla f(\boldsymbol{x}^{(t)})\right\| \neq 0$$



- $\left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\right\|^2 = \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^* - \eta\nabla f(\boldsymbol{x}^{(t)})\right\|^2$

$$= \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2 - 2\eta\nabla f\left(\boldsymbol{x}^{(t)}\right)^\top\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right) + \eta^2\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2$$

$$\leq \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2 - 2\eta\frac{1}{\beta}\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2 + \eta^2\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2 \quad (f \text{ is } \beta\text{-smooth})$$

$$= \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2 - \eta\left(\frac{2}{\beta} - \eta\right)\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2$$

Proof sketch – Thm 1 (cont'd)

- Claim 2: Since f is β-smooth, we have

$$f\left(\boldsymbol{x}^{(t+1)}\right) \leq f\left(\boldsymbol{x}^{(t)}\right) + \nabla f\left(\boldsymbol{x}^{(t)}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right) + \frac{\beta}{2}\left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right\|^2$$

$$= f\left(\boldsymbol{x}^{(t)}\right) - \eta\frac{\beta}{2}\left(\frac{2}{\beta} - \eta\right)\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2$$

- Claim 3: Since f is convex , we have

$$f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*) \leq \nabla f\left(\boldsymbol{x}^{(t)}\right)^{\mathsf{T}}\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right)$$

$$\leq \left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\| \quad \text{(Cauchy-Schwarz inequality)}$$

$$\leq R\left\|\nabla f(\boldsymbol{x}^{(t)})\right\|$$

Proof sketch – Thm 1 (cont'd)

- By Claim 2 and Claim 3, we have

$$f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*) \le f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*) - \underbrace{\frac{1}{R^2}\eta\frac{\beta}{2}\left(\frac{2}{\beta} - \eta\right)}_{:= C > 0}\left(f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*)\right)^2$$

- But this is equivalent to

$$\frac{1}{f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*)} \ge \frac{1}{f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*)} + C\,\frac{f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*)}{f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*)}$$

- Since by Claim 2, $f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*) \le f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*)$, it follows

$$\frac{1}{f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*)} \ge \frac{1}{f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*)} + C$$

$$\Rightarrow \quad \frac{1}{f\left(\boldsymbol{x}^{(t+1)}\right) - f(\boldsymbol{x}^*)} \ge \frac{1}{f\left(\boldsymbol{x}^{(0)}\right) - f(\boldsymbol{x}^*)} + Ct \quad \Leftrightarrow \quad \text{claim of the theorem}$$

Strongly convex convex functions

- Function f is said to be α-strongly convex if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^\top (\boldsymbol{y} - \boldsymbol{x}) + \frac{\alpha}{2} \|\boldsymbol{y} - \boldsymbol{x}\|^2$$

- If f is twice differentiable than for all $\boldsymbol{x}$ all eigenvalues of the Hessian matrix

$$\nabla^2 f(\boldsymbol{x})$$

are larger than or equal to α

Gradient descent: smooth and strongly convex

- Thm 2. If f is a α-strongly and β-smooth convex function, then for gradient descent algorithm with step size $\eta = \frac{2}{\alpha+\beta}$ and $\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\| \le R$, we have

$$f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*) \le R^2 \frac{\beta}{2} e^{-\frac{4}{\kappa+1}t}$$

  where $\kappa = \beta/\alpha$

- Think of κ as of the condition number of the Hessian matrix $\nabla^2 f(\boldsymbol{x})$

Proof sketch – Thm 2

- Claim 1: If f is an α-strongly convex and β-smooth convex function, then

$$\left(\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\right)^{\top}(\boldsymbol{y} - \boldsymbol{x}) \geq \frac{\alpha\beta}{\alpha + \beta}\|\boldsymbol{y} - \boldsymbol{x}\|^2 + \frac{1}{\alpha + \beta}\|\nabla f(\boldsymbol{y}) - \nabla f(\boldsymbol{x})\|^2$$

- The claims follows from:
  - $\phi(\boldsymbol{x}) := f(\boldsymbol{x}) - \frac{\alpha}{2}\|\boldsymbol{x}\|^2$ is a convex function
  - $\phi(\boldsymbol{x})$ is $(\beta - \alpha)$-smooth, thus

$$\left(\nabla \phi(\boldsymbol{y}) - \nabla \phi(\boldsymbol{x})\right)^{\top}(\boldsymbol{y} - \boldsymbol{x}) \geq \frac{1}{\beta - \alpha}\|\nabla \phi(\boldsymbol{y}) - \nabla \phi(\boldsymbol{x})\|^2$$

  which yields the claim by straightforward calculus

- Exercise: try prove Claim 1

Proof sketch – Thm 2 (cont'd)

$$\left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^*\right\|^2 = \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^* - \eta \nabla f(\boldsymbol{x}^{(t)})\right\|^2$$

$$= \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2 - 2\eta \nabla f\left(\boldsymbol{x}^{(t)}\right)^{\top}\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right) + \eta^2 \left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2$$

$$\leq \left(1 - 2\frac{\eta\alpha\beta}{\alpha+\beta}\right)\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2 + \left(\eta^2 - 2\frac{\eta}{\alpha+\beta}\right)\left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\|^2 \qquad \text{(Claim 1)}$$

$$= \left(\frac{\kappa-1}{\kappa+1}\right)^2 \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2$$

$$\leq e^{-\frac{4}{\kappa+1}} \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2$$

$$\leq e^{-\frac{4}{\kappa+1}(t+1)} \left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\right\|^2$$

Proof sketch – Thm 2 (cont'd)

$$f\left(\boldsymbol{x}^{(t)}\right) - f(\boldsymbol{x}^*) \leq \nabla f\left(\boldsymbol{x}^{(t)}\right)^{\top}\left(\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right)$$

$$\leq \left\|\nabla f\left(\boldsymbol{x}^{(t)}\right)\right\| \left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|$$ (Cauchy-Schwarz inequality)

$$\leq \frac{\beta}{2}\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}^*\right\|^2$$ (f is β-smooth)

$$\leq R^2 \frac{\beta}{2} e^{-\frac{4}{\kappa+1}t}$$ (previous slide)

Stochastic gradient descent

Scalability issues of gradient descent

- Scalability issue: computing the gradient vector $\nabla f(\boldsymbol{w})$ requires one pass through all the training data points – this is expensive!

$$\nabla f(\boldsymbol{w}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{w}} \ell\big(\boldsymbol{y}_i, h_{\boldsymbol{w}}(\boldsymbol{x}_i)\big) + \lambda \nabla \phi(\boldsymbol{w})$$

Solution: estimate the gradient vector (stochastic gradient descent)

- Gradient vector: how to compute the gradient for a multi-layer neural network?

Solution: use the chain rule (backpropagation algorithm) – we cover in the next lecture

Stochastic gradient descent

- Stochastic gradient descent algorithm estimates the gradient vector of the loss function by using a sample of training examples

- Stochastic gradient: for a sample training example $(\boldsymbol{x}_s, \boldsymbol{y}_s)$ compute

$$\hat{\nabla} f(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \ell\left(\boldsymbol{y}_s, h_{\boldsymbol{w}}(\boldsymbol{x}_s)\right) + \lambda \nabla \phi(\boldsymbol{w})$$

- For a random sample of a training example, stochastic gradient is an unbiased estimator of the true gradient $\nabla f(\boldsymbol{w})$

Robbins and Monro, A stochastic approximation method, Ann. Math. Stat., Vol 22, No 3, 400-407, 1951

Stochastic gradient descent algorithm

- **Initialization**: $t = 1$, step size sequence $\eta^{(t)}$, initial parameter vector $\boldsymbol{w}$

while stopping criterion is not met **do**

Sample a training example $(\boldsymbol{x}_{s_t}, \boldsymbol{y}_{s_t})$

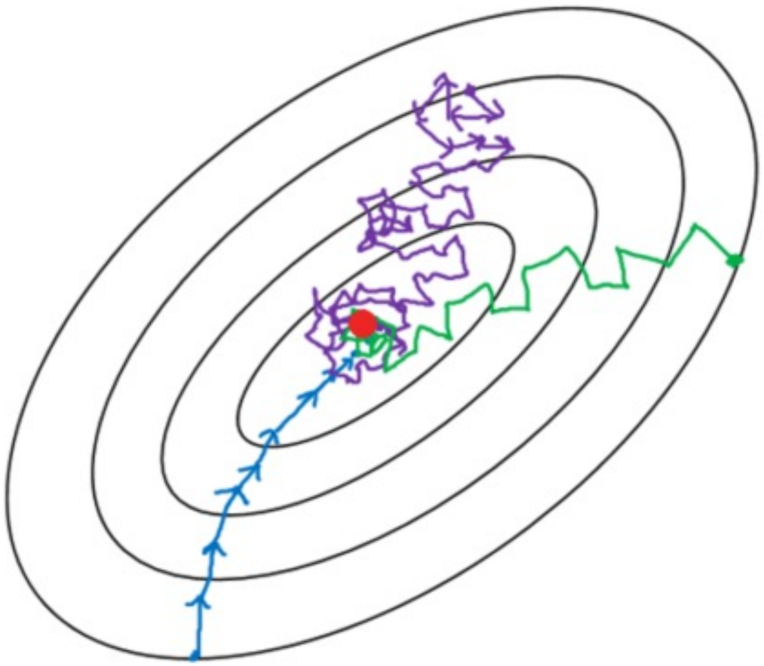
Compute stochastic gradient vector:

$$\hat{\nabla} f(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \ell\left(\boldsymbol{y}_{s_t}, h_{\boldsymbol{w}}(\boldsymbol{x}_{s_t})\right) + \lambda \nabla \phi(\boldsymbol{w})$$

apply update: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta^{(t)} \hat{\nabla} f(\boldsymbol{w})$

$t \leftarrow t + 1$

end while



Gradient descent

Mini-batch gradient descent

Stochastic gradient descent

Mini-batch gradient descent
with mini-batch size = 1

SGD convergence for smooth convex functions

Thm 3. Assume

- W is a convex set
- f is a convex β-smooth function
- $\boldsymbol{w}^{(0)} \in W$ and $R = \sup_{\boldsymbol{w} \in W} \left\| \boldsymbol{w} - \boldsymbol{w}^{(0)} \right\|$.
- Stochastic gradient vector $\widehat{\nabla} f(\boldsymbol{w})$ satisfies $\mathbf{E}\left[\left\| \widehat{\nabla} f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}) \right\|^2 \right] \leq \sigma^2$ for all $\boldsymbol{w}$
- Step size: $\eta = 1/(\beta + \sigma/R\sqrt{t/2})$.

Then,

$$\mathbf{E}\left[f\left(\frac{1}{t} \sum_{s=1}^{t} \boldsymbol{w}^{(s)} \right) \right] - f(\boldsymbol{w}^*) \leq \underbrace{\sqrt{2}\sigma R \frac{1}{\sqrt{t}} + \beta R^2 \frac{1}{t}}_{= O\left(\frac{1}{\sqrt{t}}\right)}$$

[Bubeck Theorem 6.3]

Convergence rates for convex functions

f	Algorithm	Rate	# Iter	Cost/iter
non-smooth	center of gravity	$\exp\left(-\frac{t}{n}\right)$	$n \log\left(\frac{1}{\varepsilon}\right)$	1 ∇, 1 n-dim $\int$
non-smooth	ellipsoid method	$\frac{R}{r} \exp\left(-\frac{t}{n^2}\right)$	$n^2 \log\left(\frac{R}{r\varepsilon}\right)$	1 ∇, mat-vec $\times$
non-smooth	Vaidya	$\frac{Rn}{r} \exp\left(-\frac{t}{n}\right)$	$n \log\left(\frac{Rn}{r\varepsilon}\right)$	1 ∇, mat-mat $\times$
quadratic	CG	exact $\exp\left(-\frac{t}{\kappa}\right)$	n $\kappa \log\left(\frac{1}{\varepsilon}\right)$	1 ∇
non-smooth, Lipschitz	PGD	$RL/\sqrt{t}$	R^2L^2/ε^2	1 ∇, 1 proj.
smooth	PGD	$\beta R^2/t$	$\beta R^2/\varepsilon$	1 ∇, 1 proj.
smooth	AGD	$\beta R^2/t^2$	$R\sqrt{\beta/\varepsilon}$	1 ∇
smooth (any norm)	FW	$\beta R^2/t$	$\beta R^2/\varepsilon$	1 ∇, 1 LP
strong. conv., Lipschitz	PGD	$L^2/(\alpha t)$	$L^2/(\alpha\varepsilon)$	1 ∇ , 1 proj.
strong. conv., smooth	PGD	$R^2 \exp\left(-\frac{t}{\kappa}\right)$	$\kappa \log\left(\frac{R^2}{\varepsilon}\right)$	1 ∇ , 1 proj.
strong. conv., smooth	AGD	$R^2 \exp\left(-\frac{t}{\sqrt{\kappa}}\right)$	$\sqrt{\kappa} \log\left(\frac{R^2}{\varepsilon}\right)$	1 ∇
$f+g$, f smooth, g simple	FISTA	$\beta R^2/t^2$	$R\sqrt{\beta/\varepsilon}$	1 ∇ of f Prox of g
$\max_{y \in \mathcal{Y}} \varphi(x,y)$, φ smooth	SP-MP	$\beta R^2/t$	$\beta R^2/\varepsilon$	MD on $\mathcal{X}$ MD on $\mathcal{Y}$
linear, $\mathcal{X}$ with F ν-self-conc.	IPM	$\nu \exp\left(-\frac{t}{\sqrt{\nu}}\right)$	$\sqrt{\nu} \log\left(\frac{\nu}{\varepsilon}\right)$	Newton step on F
non-smooth	SGD	$BL/\sqrt{t}$	B^2L^2/ε^2	1 stoch. ∇, 1 proj.
non-smooth, strong. conv.	SGD	$B^2/(\alpha t)$	$B^2/(\alpha\varepsilon)$	1 stoch. ∇, 1 proj.
$f = \frac{1}{m}\sum f_i$ f_i smooth strong. conv.	SVRG	–	$(m+\kappa) \log\left(\frac{1}{\varepsilon}\right)$	1 stoch. ∇

[Bubeck]

Batch and mini-batch algorithms

- Batch gradient descent: algorithm calculates the gradient vector for each example in the training dataset and then updates the parameter vector by using the mean value of the computed gradient vectors
  - This is gradient descent algorithm
  - One pass through the entire training dataset is called a training epoch

- Mini-batch gradient descent: algorithm splits the training datasets into small batches that are used to calculate the gradient vector and update the parameter vector
  - Implementations may choose to sum the gradient over the mini-batches or take the mean of the gradients (variance reduction)

- Pros for small batches: smaller memory footprint, may improve generalization
- Pros for large batches: parallelization

Minibatch stochastic gradient descent

- Minbatch SGD: instead of using only one example to estimate the gradient vector, a batch of examples in a set $S(t)$ is used:

$$\hat{\nabla} f(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} \frac{1}{|S(t)|} \sum_{s \in S(t)} \ell\big(\boldsymbol{y}_s, h_{\boldsymbol{w}}(\boldsymbol{x}_s)\big) + \lambda \nabla \phi(\boldsymbol{w})$$

  where $|S(t)|$ is batch size, a fixed constant

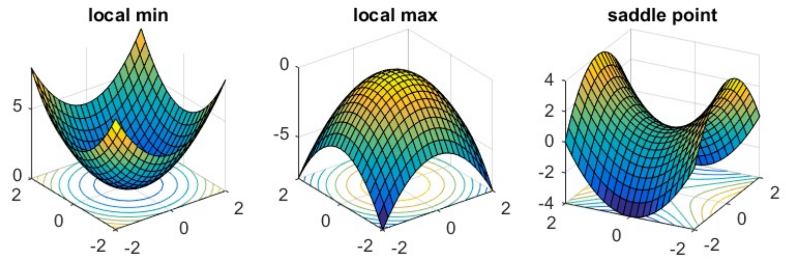
- Larger batch size decreases the variance of the stochastic gradient, and may also increase computation efficiency

- See this: https://d2l.ai/chapter_optimization/minibatch-sgd.html

Gradient descent and non-convex functions

Non-convex functions: strict saddle property

- Suppose $f: \mathbf{R}^n \to \mathbf{R}$ is a twice differentiable function



- A point $\boldsymbol{x}^* \in \mathbf{R}^n$ is a critical point of f if $\nabla f(\boldsymbol{x}^*) = 0$

- Local minimum: a critical point $\boldsymbol{x}^*$ is a local minimum if there exists a neighborhood X around $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*)$ for **all** $\boldsymbol{x} \in X$

- Saddle point: a critical point $\boldsymbol{x}^*$ is a saddle point if for each neighborhood set X around $\boldsymbol{x}^*$ such that $f(\boldsymbol{x}) \leq f(\boldsymbol{x}^*) \leq f(\boldsymbol{y})$ for **some** $\boldsymbol{x}, \boldsymbol{y} \in X$

- A function f satisfies the strict saddle property if each critical point $\boldsymbol{x}^*$ is either:
  - A local minimizer, or
  - A strict saddle, i.e. $\nabla^2 f(\boldsymbol{x}^*)$ has at least one negative eigenvalue

Importance of random initialization

- Thm 4. If $f: \mathbf{R}^n \to \mathbf{R}$ is a twice-differentiable function that satisfies the strict saddle property, then gradient descent algorithm with a random initialization and sufficiently small constant step size **either**:

  (a) converges to a local minimizer, or

  (b) diverges

  almost surely

- Gradient descent escapes strict saddle points !

[Lee, Simchowitz, Jordan, and Recht 2016]

Quadratic function example

- Consider $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x}$ where $\boldsymbol{H} = \mathbf{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ such that

  for $1 \le k < n$: $\lambda_1, \dots, \lambda_k > 0$ and $\lambda_{k+1}, \dots, \lambda_n < 0$

- For such function f, $\boldsymbol{x}^* = \mathbf{0}$ is the unique critical point (which is a strict saddle point)

- Consider the gradient descent algorithm

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta \boldsymbol{H}\boldsymbol{x}^{(t)}$$

  with step size $0 < \eta < 1/\beta$ where $\beta = \max_i |\lambda_i|$

- **Q**: What is the limit point of the gradient descent algorithm for a given $\boldsymbol{x}^{(0)}$?

Quadratic function example (cont'd)

- Note that $\boldsymbol{x}^{(t+1)} = (I - \eta \boldsymbol{H})\boldsymbol{x}^{(t)}$

  where $I - \eta \boldsymbol{H} = \mathbf{diag}(1 - \eta\lambda_1, \dots, 1 - \eta\lambda_n)$

  Hence, we have

$$\boldsymbol{x}^{(t)} = \mathbf{diag}\left((1 - \eta\lambda_1)^t, \dots, (1 - \eta\lambda_n)^t\right)\boldsymbol{x}^{(0)}$$

  i.e.

$$\boxed{\boldsymbol{x}^{(t)} = \sum_{i=1}^{n}(1 - \eta\lambda_i)^t\left(\boldsymbol{e}_i^\top \boldsymbol{x}^{(0)}\right)\boldsymbol{e}_i}$$

  where $\boldsymbol{e}_i$ is a standard basis vector with the i-th element equal to 1 and other equal to 0

Quadratic function example (cont'd)

- Note

$$1 - \eta\lambda_i \begin{cases} < 1 & \text{for } i = 1,2,\dots,k \\ > 1 & \text{for } i = k+1,\dots,n \end{cases}$$

- Hence, if $\underbrace{\boldsymbol{x}^{(0)} \in \operatorname{span}(\boldsymbol{e}_1,\dots,\boldsymbol{e}_k)}_{\text{i.e. } \boldsymbol{x}^{(0)} \text{ is a linear combination of } \boldsymbol{e}_1,\dots,\boldsymbol{e}_k}$, then $\lim_{t\to\infty} \boldsymbol{x}^{(t)} = \mathbf{0}$

  otherwise, $\lim_{t\to\infty} \left\| \boldsymbol{x}^{(t)} \right\| = \infty$

- If $\boldsymbol{x}^{(0)}$ is a random point, then $\mathbf{Pr}[\boldsymbol{x}^{(0)} \in \operatorname{span}(\boldsymbol{e}_1,\dots,\boldsymbol{e}_k)] = 0$

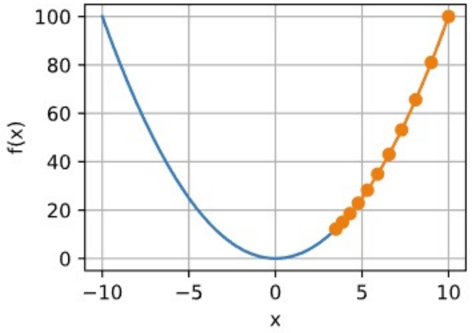
  thus, gradient descent escapes the saddle point

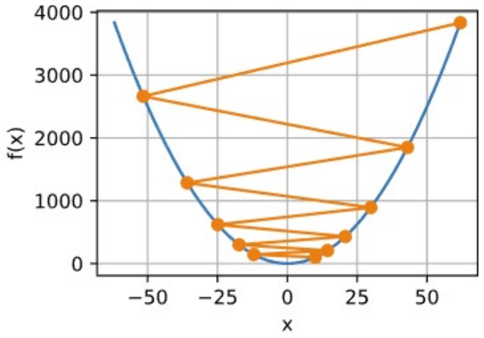
References

- Zhang, Lipton, Li, and Smola, Dive into Deep Learning, https://d2l.ai/chapter_optimization/index.html
- Bubeck, Convex optimization: algorithms and convexity, Now publishers, 2015, https://arxiv.org/pdf/1405.4980.pdf
- Nesterov, Introductory Lectures on Convex Programming, 1998
- Sutskever, Martens, Dahl, and Hinton, On the Importance of Initialization and Momentum in Deep Learning, ICML 2013
- Bottou and O. Bousquet, The trade-offs of large scale learning, NIPS 2007
- Lee, Simehowitz, Jordan, and Recht, Gradient descent only converges to minimizers, COLT 2016

Seminar 3

- Gradient descent
- Stochastic gradient descent
- Optimization framework and functions in TensorFlow





Solution to exercise in proof sketch Thm 2

- P1: $\phi(x) = f(x) - \frac{\alpha}{2}\|x\|^2$ is convex

$$
\begin{aligned}
&\phi(y) = f(y) - \frac{\alpha}{2}\|y\|^2 \\
&\geq f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}\|y - x\|^2 \ - \frac{\alpha}{2}\|y\|^2 \\
&= f(x) + \nabla f(x)^\top (y - x) - \alpha x^\top y + \frac{\alpha}{2}\|x\|^2 \\
&= \phi(x) + \frac{\alpha}{2}\|x\|^2 + (\nabla\phi(x) + \alpha x)^\top (y - x) - \alpha x^\top y + \frac{\alpha}{2}\|x\|^2 \\
&= \phi(x) + \nabla\phi(x)(y - x) + \alpha\|x\|^2 + \alpha x^\top y \\
&= \phi(x) + \nabla\phi(x)(y - x)
\end{aligned}
$$

- We have shown $\phi(y) \geq \phi(x) + \nabla\phi(x)(y - x)$ which means ϕ is convex

Solution to exercise in proof sketch Thm 2 (cont'd)

- P2: ϕ is $(\beta - \alpha)$-smooth

$$\phi(x) - \phi(y) - \nabla\phi(y)^\top(x - y)$$

$$= f(x) - f(y) - \frac{\alpha}{2}(\|x\|^2 - \|y\|^2) - (\nabla f(y) - \alpha y)^\top(x - y)$$

$$\leq \nabla f(y)^\top(x - y) + \frac{\beta}{2}\|x - y\|^2 - \frac{\alpha}{2}(\|x\|^2 - \|y\|^2) - (\nabla f(y) - \alpha y)^\top(x - y)$$ (f is β-smooth)

$$= \frac{\beta}{2}\|x - y\|^2 - \frac{\alpha}{2}(\|x\|^2 - \|y\|^2) + \alpha y^\top x - \alpha\|y\|^2$$

$$= \frac{\beta}{2}\|x - y\|^2 - \frac{\alpha}{2}\|x\|^2 + \alpha y^\top x - \frac{\alpha}{2}\|y\|^2$$

$$= \frac{\beta-\alpha}{2}\|x - y\|^2$$

Solution to exercise in proof sketch Thm 2 (cont'd)

- P3: $\left(\nabla f(y) - \nabla f(x)\right)^{\top}(y - x) \geq \frac{\alpha\beta}{\alpha+\beta}\|y - x\|^2 + \frac{1}{\alpha+\beta}\|\nabla f(y) - \nabla f(x)\|^2$
- Case $\alpha = \beta$:

By (P2), ϕ is 0-smooth, hence

$\phi(x) - \phi(y) - \nabla\phi(y)^{\top}(x - y) = 0$, which is equivalent to

$$f(x) - f(y) - \nabla f(y)^{\top}(x - y) = \frac{\alpha}{2}\|x - y\|^2$$

It follows $\nabla f(x) - \nabla f(y) = \alpha(x - y)$

from which (P3) follows

Solution to exercise in proof sketch Thm 2 (cont'd)

- P3: $\left(\nabla f(y) - \nabla f(x)\right)^{\top}(y - x) \geq \frac{\alpha\beta}{\alpha+\beta}\|y - x\|^2 + \frac{1}{\alpha+\beta}\|\nabla f(y) - \nabla f(x)\|^2$

- Case $\beta > \alpha$:

$$\left(\nabla f(x) - \nabla f(y)\right)^{\top}(x - y) \geq \alpha\|x - y\|^2 + \frac{1}{\beta-\alpha}\|\nabla f(x) - \nabla f(y) - \alpha(x - y)\|^2 \quad \text{(from P1)}$$

$$= \alpha\|x - y\|^2 + \frac{1}{\beta - \alpha}\left(\|\nabla f(x) - \nabla f(y)\|^2 - 2\alpha\left(\nabla f(x) - \nabla f(y)\right)^{\top}(x - y) + \alpha^2\|x - y\|^2\right)$$

$$= \frac{\alpha\beta}{\beta - \alpha}\|x - y\|^2 + \frac{1}{\beta - \alpha}\|\nabla f(x) - \nabla f(y)\|^2 - \frac{2\alpha}{\beta - \alpha}\left(\nabla f(x) - \nabla f(y)\right)^{\top}(x - y)$$

from which (P3) follows