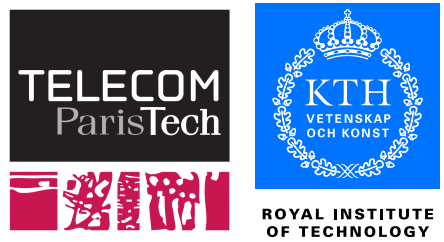


Deep Learning

Lecture 2: Learning and Convex Optimization

University of Agder,
Kristiansand, Norway

Prepared by: Hadi Ghauch[⋆], Hossein S. Ghadikolaei[†]

[⋆] Telecom ParisTech

[†] Royal Institute of Technology, KTH

`https://sites.google.com/view/fundl/home`

April 2019

Learning outcomes

Supervised/unsupervised learning as optimization

Definitions and properties of smooth and convex functions

Deterministic iterative algorithms for convex problems

Advantages and limitations of each

Convergence analysis

Intro to Reinforcement Learning

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Machine Learning

- **Supervised learning**

  learning from **labeled training set**, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$
  $\boldsymbol{x}_i$ is feature vector, y_i label for sample $i \in [N]$
  e.g., deep learning for **regression/classification** task

- **Unsupervised learning**

  learning from **unlabeled training set**, $\{\boldsymbol{x}_i\}_{i=1}^N$
  identify inherent structure of the data

  is it low-dimensional (PCA, MF) ? does it lie on manifold (manifold learning)

- **Reinforcement learning (a.k.a online learning)**

  learning by interacting with an unknown environment (modeled by a Markov decision process), e.g., Q-learning

Supervised learning: Recap

A dataset of N training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$

hidden (unknown) fnc that generates training set

learned func from the training set (a.k.a **model**)

Prediction on sample i: and loss on each sample

Empirical risk (training error):

Expected risk (true risk):

necessary cond on the **shattering coeff** to ensure empirical risk and expected (true) risk are uniformly close

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Learning and Optimization

Linear ridge regression:
Given set of feature vectors with labels: $(\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_N, y_N)$
find the hyperplane $\boldsymbol{w} \in \mathbb{R}^d$ that min the squared loss (board)

$$\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \boldsymbol{w}^T\boldsymbol{x}_i\right)^2 \quad + \quad \lambda\|\boldsymbol{w}\|_2^2 \qquad\qquad := f(\boldsymbol{w})$$

training error + regularizer regularized loss

$f()$: **loss function** + regularization ($\ell(\ )$ in Lec 1)

$\boldsymbol{w}$: **function/model** to be learned ($f(\ )$ in Lec 1)

increasing λ makes training error larger while reducing overfitting

Linear LASSO regression: $f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \boldsymbol{w}^T\boldsymbol{x}_i\right)^2 + \lambda\|\boldsymbol{w}\|_1$

Supervised learning always solves an optimization problem
- Convex problem for linear/logistic regression, SVM

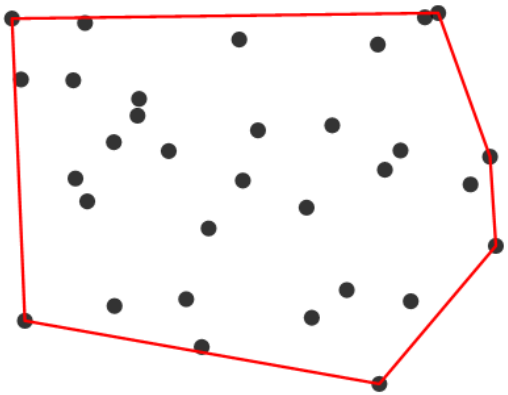
Convexity: Basic definitions

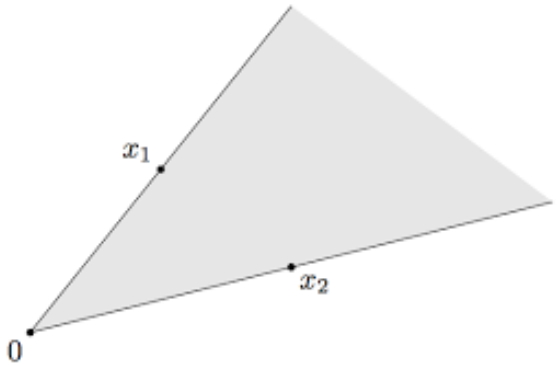
Convex combination of points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i\in[n]}$ is $\sum_{i\in[n]} \theta_i \boldsymbol{x}_i$
where $\{\theta_i\}$ are *convex weights*, $\sum_i \theta_i = 1$

Conic combination of points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i\in[n]}$ is $\sum_{i\in[n]} \theta_i \boldsymbol{x}_i$
where $\{\theta_i\}$ are *affine weights*, $\theta_i \geq 0$

Convex hull of $\mathcal{X}$: set of all convex combinations of points in $\mathcal{X}$

Convex cone of $\mathcal{X}$: set of all conic combinations of points in $\mathcal{X}$





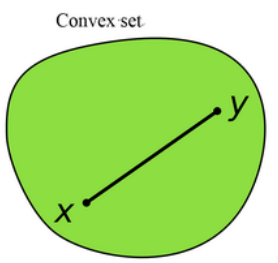
Convex Sets

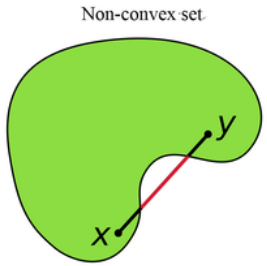
Convex set $\mathcal{X}$: $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\theta \in [0, 1]$, $\theta \boldsymbol{x}_1 + (1 - \theta)\boldsymbol{x}_2 \in \mathcal{X}$.

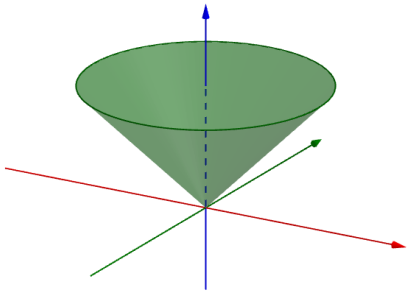
Intution: convex combination of any two points in the set is within the set.

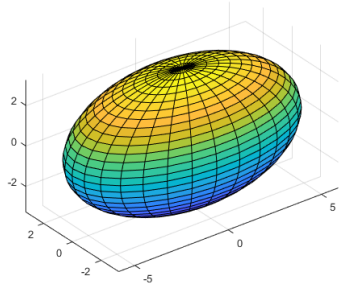
Examples:

- Euclidean ball with radius r at $\boldsymbol{x}_c$: $\{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}_c\|_2 \leq r\}$
- Norm cone: $\{(\boldsymbol{x}, r) \mid \|\boldsymbol{x}\| \leq r\}$
- Ellipsoid centered at $\boldsymbol{x}_c$: $\{(\boldsymbol{x} - \boldsymbol{x}_c)^T \boldsymbol{P}^{-1} (\boldsymbol{x} - \boldsymbol{x}_c) \leq 1\}$, where $\boldsymbol{P}$ is positive definite









Convex Functions

Convex function $f : \mathcal{X} \to \mathbb{R}$: if $\mathcal{X}$ is convex and $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ and $\theta \in [0, 1]$:

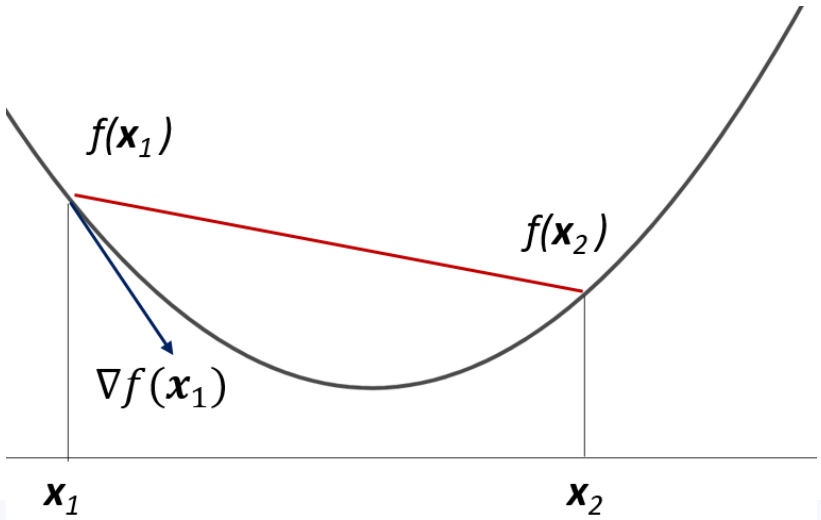
$$f\left(\theta \boldsymbol{x}_1 + (1-\theta)\boldsymbol{x}_2\right) \leq \theta f(\boldsymbol{x}_1) + (1-\theta) f(\boldsymbol{x}_2) \tag{1}$$

If f differentiable, (1) is equivalent to

$$f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f(\boldsymbol{x}_1)^T (\boldsymbol{x}_2 - \boldsymbol{x}_1), \ \forall (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{X} \tag{2}$$

Intuition? grad is a **global linear lower bound** what about non-smooth convex ?

If f twice differentiable, (1) $\leftrightarrow \nabla^2 f(\boldsymbol{x}) \succeq \mathbf{0}, \forall \boldsymbol{x} \in \mathcal{X}$: PSD Hessian, non-negative curvature everywhere.



Convex Functions: examples

- $\|\boldsymbol{x}\|_p$ for any $p \geq 1$
- $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$ is convex for any $\boldsymbol{A}$ (since $\nabla^2 f(\boldsymbol{x}) := 2\boldsymbol{A}^T\boldsymbol{A} \succeq \boldsymbol{0}$)
- $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p$ for any $p \geq 1$
- Set of PSD matrices $\boldsymbol{X} \succeq \boldsymbol{0}$ is a cone (why?). It is convex.
- $\lambda_{\max}(\boldsymbol{X})$ convex in $\boldsymbol{X}$, for $\boldsymbol{X} \succeq \boldsymbol{0}$
- $\mathrm{proj}(\boldsymbol{x}, \mathcal{C}) := \inf_{y \in \mathcal{C}} \|\boldsymbol{y} - \boldsymbol{x}\|$ for convex $\mathcal{C}$
- Check Boyd and Vandenberghe (2003) for more examples

  Nonnegative weighted sum, composition with affine function, pointwise maximum and supremum, composition, minimization, and perspective

Convex Problems and Duality

$$\textbf{Primal problem:} \quad \begin{aligned} \text{minimize} \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq 0, i \in [m] \\ & h_i(\boldsymbol{x}) = 0, i \in [p] \end{aligned} \tag{3}$$

f_0 convex cost fnc, f_i convex inequality constraints, h_i affine equality constraints.

Lagrange dual function: $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

$$g(\lambda, \nu) = \inf_{x \in \mathcal{X}} \; L(x, \lambda, \nu) := f_0(\boldsymbol{x}) + \sum_{i \in [m]} \lambda_i f_i(\boldsymbol{x}) + \sum_{i \in [p]} \nu_i h_i(\boldsymbol{x})$$

$$\textbf{Lagrange dual problem:} \quad d^\star = \begin{aligned} \text{maximize} \quad & g(\lambda, \nu) \\ \text{s.t.} \quad & \lambda_i \geq 0, i \in [m] \quad , \; \mu_i \neq 0, i \in [p] \end{aligned}$$

Weak duality: $d^\star \leq f^\star$ (holds for any (3))

Strong duality: $d^\star = f^\star$ (holds when (3) strongly convex)

Duality

- Check Boyd and Vandenberghe (2003) for more details

  Constraint qualifications

  Slater's constraint qualification

  Complementary slackness

  Karush-Kuhn-Tucker (KKT) conditions

Strong convexity

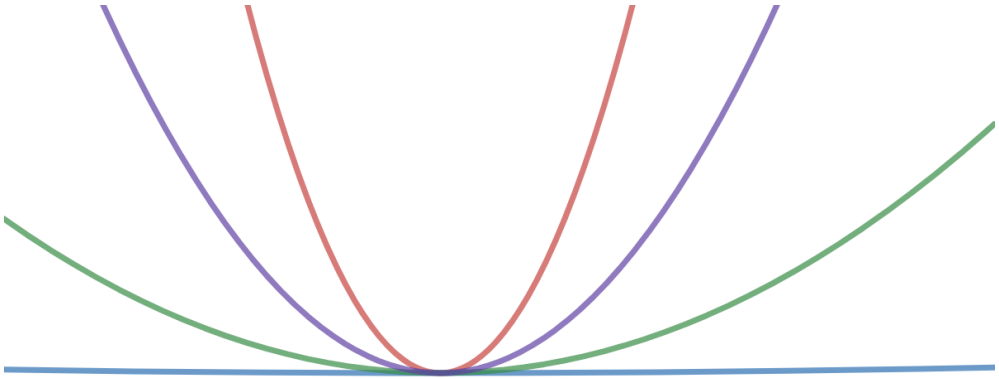
Differentiable function f is μ-strongly convex iff $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \mu > 0$

$$\underbrace{f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f(\boldsymbol{x}_1)^T (\boldsymbol{x}_2 - \boldsymbol{x}_1)}_{\text{definition of convex } f} + \frac{\mu}{2} \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2$$

Intuition:
for convex f: gradient is global linear lowerbound
for strongly convex f: quadratic lower bound with strong convexity

Global definitions not local ($\forall \boldsymbol{x} \in \mathcal{X}$)

Gradient can be replaced by sub-gradient for non-smooth functions



Strong convexity: Equivalent definitions

$$f(\boldsymbol{x}_2) - f(\boldsymbol{x}_1) \geq \nabla f(\boldsymbol{x}_1)^T(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{\mu}{2}\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2 \tag{4}$$

- (4) is equivalent to a **minimum positive curvature**
  $\nabla^2 f(\boldsymbol{x}) \succeq \mu \boldsymbol{I}_d, \forall \boldsymbol{x} \in \mathcal{X}$
- (4) is equivalent to $\left(\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\right)^T (\boldsymbol{x}_2 - \boldsymbol{x}_1) \geq \mu \|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2$
- (4) implies

  (a) $f(\boldsymbol{x}) - f(\boldsymbol{x}^\star) \leq \frac{1}{2\mu}\|\nabla f(\boldsymbol{x})\|_2^2, \forall \boldsymbol{x}$

  dist bw any points $f(\boldsymbol{x})$ and $f(\boldsymbol{x}^\star)$ upperbounded by $\|\nabla f(\boldsymbol{x})\|$

  (b) $\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2 \leq \frac{1}{\mu}\|\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\|_2, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$

  dist bw any two points upperbounded by dist of their grad

  (c) $\left(\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\right)^T (\boldsymbol{x}_2 - \boldsymbol{x}_1) \leq \frac{1}{\mu}\|\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\|_2^2, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$

Smoothness

A function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth iff it is diffrentiable and its gradient is L-Lipschitz-continuous (usually w.r.t. norm-2):

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^d, \|\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\|_2 \leq L\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2 \tag{5}$$

If f twice differentiable and smooth:

(5) equivalent $\nabla^2 f(\boldsymbol{x}) \preceq L\boldsymbol{I}_d$: upperbound on max curvature of f

L upperbound on largest eigenvalue of $\nabla^2 f(\boldsymbol{x})$

recall: for strongly convex: μ lowerbound on min eigenval of $\nabla^2 f(\boldsymbol{x})$

Strongly convex and L-smooth:

$$\mu\boldsymbol{I}_d \preceq \nabla^2 f(\boldsymbol{x}) \preceq L\boldsymbol{I}_d$$

L and μ upperbound and lowerbound on curvature of f

(5) implies

(a) $f(\boldsymbol{x}_2) \leq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^T(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{L}{2}\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$

(b) $f(\boldsymbol{x}_2) \geq f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^T(\boldsymbol{x}_2 - \boldsymbol{x}_1) + \frac{1}{2L}\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|_2^2, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$

(c) $(\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1))^T (\boldsymbol{x}_2 - \boldsymbol{x}_1) \geq \frac{1}{L}\|\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\|_2^2, \forall \boldsymbol{x}_1, \boldsymbol{x}_2$

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Descent methods

Problem: $\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\ f(\boldsymbol{w})$

$f : \mathbb{R}^d \to \mathbb{R} \cup \{\pm\infty\}$ differentiable and convex

Intuition: iteratively move along direction where f decreases. Formally,

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{d}(\boldsymbol{w}_k),$$

- $(\alpha_k)_{k\in\mathbb{N}}$ non-negative step sizes
- $\boldsymbol{d}(\boldsymbol{w}_k)$, decent direction is such that $f(\boldsymbol{w}_{k+1}) < f(\boldsymbol{w}_k)$

How to choose $\boldsymbol{d}(\boldsymbol{w}_k)$?

$f(\boldsymbol{w}_{k+1}) < f(\boldsymbol{w}_k)$ implies $-\nabla f(\boldsymbol{w}_k)^T \boldsymbol{d}(\boldsymbol{w}_k) > 0$

- any direction in half-space of the negative gradient

Gradient/Steepest descent: Set $\boldsymbol{d}(\boldsymbol{w}_k) = -\nabla f(\boldsymbol{w}_k)$ (obvious choice)

Newton's method: Set $\boldsymbol{d}(\boldsymbol{w}_k) = -[\nabla^2 f(\boldsymbol{w}_k)]^{-1} \nabla f(\boldsymbol{w}_k)$

- adjust $\boldsymbol{d}(\boldsymbol{w}_k)$ to account for curvature: it is descent direction

Compare different iterative methods ?

-**Computational Complexity:** number of calculations/flops per iteration

-**Iteration Complexity:** number of iterations to get within ϵ-ball around optimal

- both done using order analysis: $\mathcal{O}()$

Gradient descent

Gradient descent (GD), aka, batch GD, full GD, steepest descent

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k) \tag{6}$$

for some sequence of non-negative step sizes $(\alpha_k)_{k \in \mathbb{N}}$.

Intuition: Approximate f at each iteration, with the gradient

Theorem 1: Convergence of GD with constant step size

a) **Convex and L-smooth** f with $\alpha \leq 1/L$ satisfies
$f(\boldsymbol{w}_k) - f(\boldsymbol{w}^\star) \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^\star\|_2^2}{2k\alpha} \approx \mathcal{O}(1/k)$.

b) **μ-strongly convex and L-smooth** f with $\alpha = 2/(\mu + L)$, we have
$\|\boldsymbol{w}_k - \boldsymbol{w}^\star\|_2 \leq \left(1 - \frac{2}{1+L/\mu}\right)^k \|\boldsymbol{w}_0 - \boldsymbol{w}^\star\|_2 = c^k \|\boldsymbol{w}_0 - \boldsymbol{w}^\star\|_2$, where $c < 1$
dist to opt decays geometrically with k: **linear convergence**

▸ Proofs

GD for smooth and strongly convex functions

Iteration Complexity : Number of iterations to get within ϵ to $\boldsymbol{w}^\star$

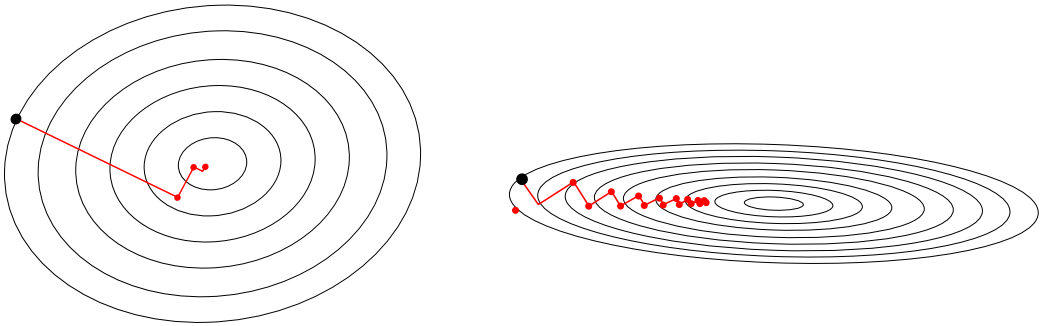
Smooth strongly-convex: $\mathcal{O}(L/\mu \log(1/\epsilon))$ iterations

Computational Complexity$= O(N)$ (evaluate grad of N samples)

Convergence Rate $c = 1 - \dfrac{2}{1 + L/\mu}$ is linear

L/μ large: iteration complexity $\mathcal{O}(L/\mu \log(1/\epsilon))$ increases

Preconditioning to change the space geometry, to make sub-levels similar in all coordinates



small L/μ large L/μ

- small L/μ: $L/\mu = 1 \Leftrightarrow$ all eigenvals of Hessian equal. Why?
- ideal setting for GD

Newton method

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k [\nabla^2 f(\boldsymbol{w}_k)]^{-1} \nabla f(\boldsymbol{w}_k)$$

Intuition: Approximate f at each iteration, with its Hessian

Theorem 2: Quadratic convergence of Newton's method

Assume f is a twice continuously differentiable and set $\alpha_k = 1$. If $\|\boldsymbol{w}_k - \boldsymbol{w}^\star\|$ is small enough, there is a constant c $(0 < c < 1)$ such that $\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^\star\|_2 \leq c\|\boldsymbol{w}_k - \boldsymbol{w}^\star\|_2$.

Iteration complexity: $\mathcal{O}(\log\log(1/\epsilon))$ iterations for ϵ-optimality

convergence is logarithmicaly faster than GD

Computationally expensive. $[\nabla^2 f(\boldsymbol{w}_k)]^{-1}$ costs $\mathcal{O}(d^3)$. Never used in large-scale learning.

'Lighter' methods: quasi-Newton or subsampled Hessian

Accelerated First-order Methods

Heavy-ball method: (B. Polyak 64)

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k) + \underbrace{\eta_k(\boldsymbol{w}_k - \boldsymbol{w}_{k-1})}_{\text{momentum}} \tag{7}$$

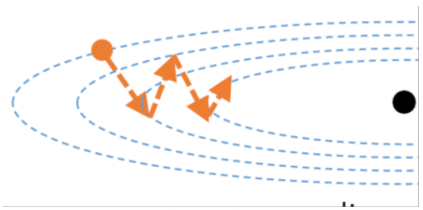
Intuition: Add momentum term to GD to reduce zigzag

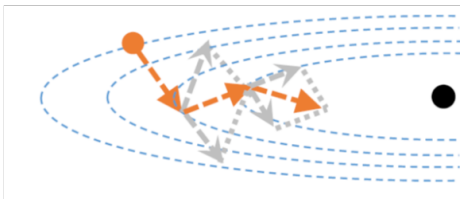
Theorem 3: Convergence of heavy-ball (strongly convex problem)
Suppose f is μ-strongly convex and L-smooth. Then,

$$\|\boldsymbol{w}_{k+1} - \boldsymbol{w}^\star\|_2 + \|\boldsymbol{w}_k - \boldsymbol{w}^\star\|_2 \leq \underbrace{\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k}_{\text{const } <1} (\|\boldsymbol{w}_1 - \boldsymbol{w}^\star\|_2 + \|\boldsymbol{w}_0 - \boldsymbol{w}^\star\|_2)$$

$\kappa := L/\mu$, $\alpha_k := 4/(\sqrt{L}+\sqrt{\mu})^2$, and $\eta_k = \max\left(|1-\sqrt{\alpha_k L}|, |1-\sqrt{\alpha_k \mu}|\right)^2$

- **Iteration complexity:** $\mathcal{O}(\sqrt{\kappa}\log(1/\epsilon))$ (much better than GD)
- **Computational Complexity**$= O(N)$ (evaluate grad of N samples)
- cons: need min and max eigenvalues of Hessian: L, μ


Gradient descent


Heavy-ball method

Accelerated First-order Methods

Nesterov Acceleration: (Y. Nesterov 83)

$$\begin{aligned} \boldsymbol{y}_k &= \boldsymbol{w}_k + \eta_k(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}) \\ \boldsymbol{w}_{k+1} &= \boldsymbol{y}_k - \alpha_k \nabla f(\boldsymbol{y}_k) \end{aligned} \tag{8}$$

Intuition: mix between gradient and extrapolation steps

Theorem 4: Convergence of Nesterov accel (strongly convex problem) Suppose f is μ-strongly convex and L-smooth. Set $\eta_k = k/(k+3)$ and $\alpha_k = \alpha = 1/L$. Then,

$$|f(\boldsymbol{w}_k) - f(\boldsymbol{w}^\star)| \leq 2L \, \|\boldsymbol{w}_k - \boldsymbol{w}^\star\|_2^2 \frac{1}{(k+1)^2} \approx \mathcal{O}(1/k^2)$$

- **convergence rate** is quadratic (like Newton): $\mathcal{O}(1/k^2)$
- **computational complexity** (like GD): $\approx \mathcal{O}(N)$
- not a descent method (iterations not necessarily monotonic) yet converges faster than GD
- no min or max eigenvalues of Hessian needed

Choice of Method

Assume 'best-case' scenario: strong convexity.

Method	**Convergence rate**	**Computational Complexity**
GD	$\mathcal{O}(1/k)$	$\mathcal{O}(N)$
Newton	$\mathcal{O}(1/k^2)$	$\mathcal{O}(d^3)$ too high
Nesterov	$\mathcal{O}(1/k^2)$	$\mathcal{O}(N)$

Number of training samples, N, in **modern datasets**:
- **Large-scale learning ?** MovieLens 100M ($N = 10^8$), Alexnet ($N = 1.2 \times 10^6$), etc.
- even smallest computational complexity, $\mathcal{O}(N)$, is too large
  $\mathcal{O}(N)$ is due to **computing gradient over all samples**
- motivation stochastic gradient descent, workhorse for large-scale learning

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Unsupervised Learning

Learn from **unlabeled data**: training samples $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$

Goal: learn **inherent feature/structure** of the data.

- clustering, matrix factorization, dictionary learning, etc

Low-rank Matrix Factorization (MF) for recommendation systems

- Data matrix $\boldsymbol{X}$ of size $M \times N$
- $[\boldsymbol{X}]_{u,i}$ is rating (score) that user u gave to item i
- In recommendation sys model rating as: $[\boldsymbol{X}]_{u,i} = \boldsymbol{p}_i^T \boldsymbol{q}_u$,
  $\boldsymbol{p}_i$ vector of **item characteristics** , $\boldsymbol{q}_u$ vector of **user preferences**
- factorize $\boldsymbol{X}$ with low-rank matrices: $\boldsymbol{P} \in \mathbb{R}^{M \times d}$, $\boldsymbol{Q} \in \mathbb{R}^{N \times d}$, $d \ll (M, N)$ such that $\boldsymbol{X} = \boldsymbol{P}\boldsymbol{Q}^T$
- rows of $\boldsymbol{Q}$ (resp $\boldsymbol{P}$) are user prefs (resp item characteristics)
- Unsupervised learning also solves optimization prob (sol. in Lec 4)

$$\arg\min_{\boldsymbol{P},\boldsymbol{Q}} \|\boldsymbol{X} - \boldsymbol{P}\boldsymbol{Q}^T\|_F^2$$

In general MF represents training set X into low-dim subspaces,
- compression of large data matrix, and prediction

Sequential Decision Making

Markov process $(\mathcal{S}, p)$, $\mathcal{S}$ is state space, p transition probability

Markov decision process $(\mathcal{S}, \mathcal{A}, p, r)$, $\mathcal{A}$ is action space

reward, r, depends on state and action, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$

maximize reward fnc: closed-form (high complexity), iteratively (low complexity)

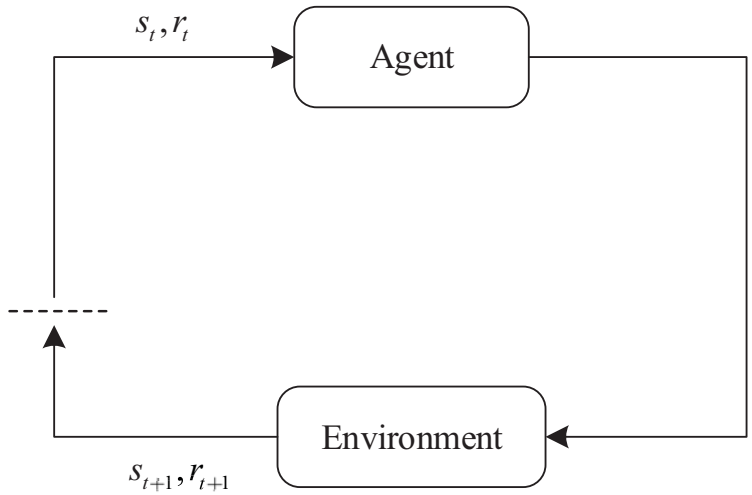
Policy: $\pi : \mathcal{S} \mapsto \mathcal{A}$

Optimal value function:
Let $\gamma \in [0, 1]$. Definite the **state-value function (discounted reward)** starting from state s following policy π is $v_\pi(s) = \mathbb{E}\left[\sum_{i=1}^{T} \gamma^{i-1} r_i \mid s_0 = s\right]$.
For any given policy and state, we can compute $v_\pi(s)$ in closed-form.
The optimal value function is $v^\star = \max_\pi v_\pi(s)$. Computing $v^\star$ needs inverse of large matrix (scales with $\mathcal{S}|$): impossible in modern application (Go)

More efficient way using **Bellman's optimality eqt**
$v^\star(s) = \max_a \sum_{s',a} p\left(s', r \mid s, a\right)\left[r + \gamma v^\star(s')\right]$
equivalent to optimal value func. fixed point eqt (both sides depend of s)
Solve iteratively until converge (fixed point exist)
So far no learning, just sequential decision making

Reinforcement Learning (RL)

In SDM: p, r, are known. Only need the policy that maximizes value function



- In reinforcement learning (RL). r and p unknown
- Agent interacts with the environment:
  In state, s_t, agent takes action (a_t), gets a reward, r_t (fnc of s_t and a_t)
  agent observes next state, s_{t+1}
- **Objective:** find the optimal policy (with unknown r and p)

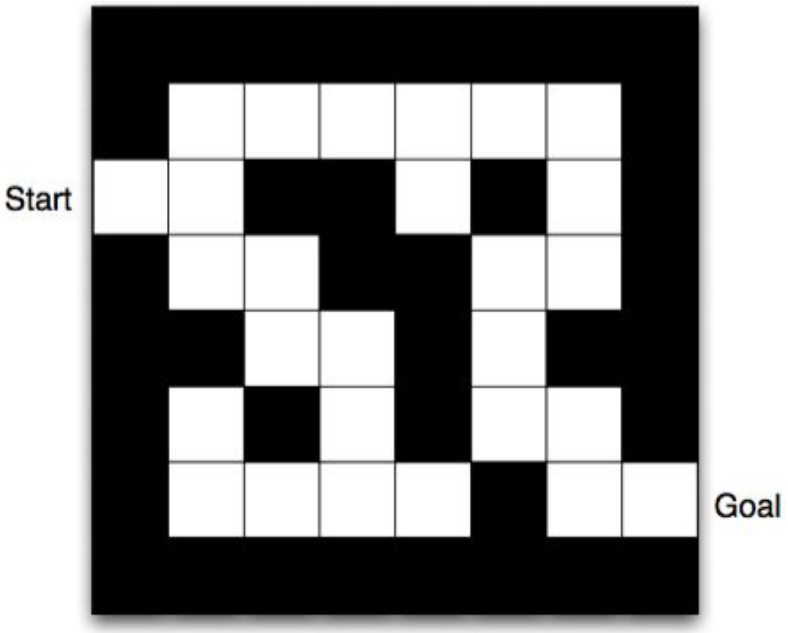
Agent interacts with dataset. Compare with SL where learning done with a fixed dataset.
RL more adaptive/efficient: online learning

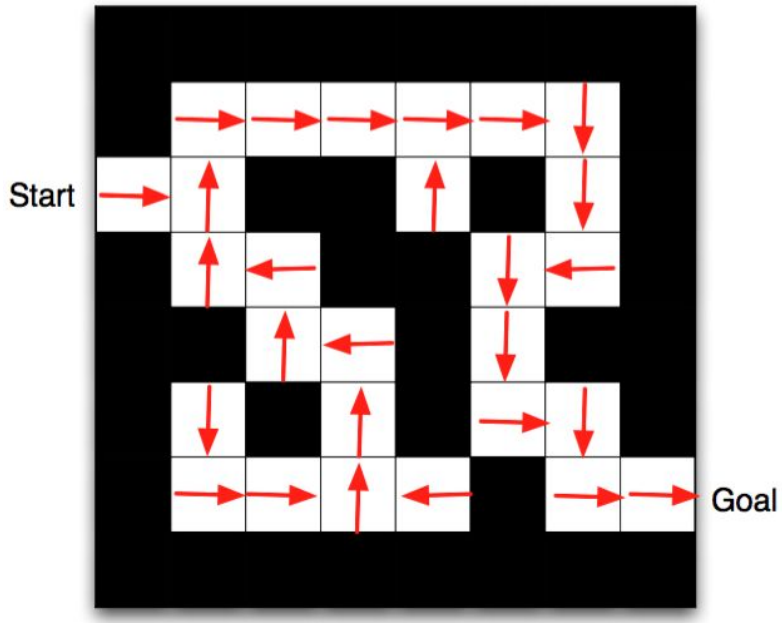
Reinforcement Learning (example)

Actions: taking one step (top/bottom/left/right) at a time

Objective: reach destination with a minimum number of steps

Assume reward known: -1 per step





Reinforcement Learning

Sample mean and stochastic approximation

estimate sample mean of random process, $\{r_n\}$, using stochastic approx

$$q_{n+1} = \frac{r_1 + r_2 + \cdots + r_{n+1}}{n+1} = q_n + \frac{1}{n+1}\left[r_{n+1} - q_n\right]$$

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}\left[\text{Target} - \text{OldEstimate}\right]$$

Reinforcement Learning

$$\text{NewEstimate} \leftarrow \text{OldEstimate} + \text{StepSize}\left[\text{Target} - \text{OldEstimate}\right]$$

TD: temporal-difference

$$v(s_t) \leftarrow v(s_t) + \alpha_{n(s)}\left[r_{t+1} + \gamma v(s_{t+1}) - v(s_t)\right]$$

SARSA: state-action-reward-state-action

From state s, take action a, observe immediate reward r and next state s'. Then choose action a' using policy derived from Q, update

$$q(s,a) \leftarrow q(s,a) + \alpha_{n(s,a)}\left[r + \gamma q(s',a') - q(s,a)\right]$$

Q-learning

$$q(s,a) \leftarrow q(s,a) + \alpha_{n(s,a)}\left[r + \gamma \max_j q(s',j) - q(s,a)\right]$$

Some references

- S. Bubeck, "Convex optimization: Algorithms and complexity," Foundations and Trends in Machine Learning, 2015.

- Y. Nesterov, "Introductory lectures on convex optimization: A basic course, " Springer Science & Business Media, 2004.

- F. Bach, "Large-scale machine learning and convex optimization," Machine Learning Summer School, 2018.

- Y. Chen, " Large-Scale Optimization for Data Science," ELE538B course, spring 2018.

- S. Trivedi and R. Kondor, "Deep Learning", CMSC 35246 course, spring 2017.

- L. Bottou, F. Curtis, and J. Norcedal, "Optimization methods for large-scale machine learning," SIAM Rev., 2018.

- N. Parikh and S. Boyd, "Proximal algorithms", gradient methods," Foundations and Trends in Optimization, 2013.

- A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM Journal on Imaging Sciences, 2009.

Outline

1. Supervised Learning
2. Optimization and Learning
3. Iterative solution approaches
4. Unsupervised and Reinforcement Learning
5. Supplements

Proof sketch for convex and L-smooth f

By convexity of f, $f(\boldsymbol{x}_i) \leq f(\boldsymbol{x}^\star) + \langle \nabla f(\boldsymbol{x}_i), \boldsymbol{x}_i - \boldsymbol{x}^\star \rangle \qquad (*)$

Use smoothness property $f(\boldsymbol{x}_{i+1}) \leq f(\boldsymbol{x}_i) + \nabla f(\boldsymbol{x}_i)^T(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) + \frac{L}{2}\|\boldsymbol{x}_{i+1} - \boldsymbol{x}_i\|_2^2$ and $\alpha \leq 1/L$ to conclude $f(\boldsymbol{x}_{i+1}) \leq f(\boldsymbol{x}_i) - \frac{\alpha}{2}\|\nabla f(\boldsymbol{x}_i)\|_2^2 \qquad (**)$

Substitute $(*)$ into $(**)$ and note that from (6): $\nabla f(\boldsymbol{x}_i) = \dfrac{\boldsymbol{x}_i - \boldsymbol{x}_{i+1}}{\alpha}$

Observe $f(\boldsymbol{x}_{i+1}) \leq f^\star + \dfrac{1}{2\alpha}\left(\|\boldsymbol{x}_i - x^\star\|_2^2 - \|\boldsymbol{x}_{i+1} - x^\star\|_2^2\right)$

Summing over iterations: $\dfrac{1}{k}\sum_{i\in[k]}\left(f(\boldsymbol{x}_i) \leq f^\star\right) \leq \dfrac{1}{2\alpha k}\|\boldsymbol{x}_0 - x^\star\|_2^2$

Conclude from the non-increasing property of GD iterates:

$$f(\boldsymbol{x}_k) - f^\star \leq \frac{1}{k}\sum_{i\in[k]} f(\boldsymbol{x}_i) - f^\star \leq \frac{1}{2\alpha k}\|\boldsymbol{x}_0 - x^\star\|_2^2.$$

Return

Proof sketch for strongly-convex and L-smooth f

From smoothness and vanishing gradient of the optimal point, conclude
$f(\boldsymbol{x}_i) - f(\boldsymbol{x}^\star) \leq \frac{L}{2}\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|_2^2 \qquad (*)$

Use the coercivity of the gradient

$$(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}))^T(\boldsymbol{x} - \boldsymbol{y}) \geq \frac{\mu L}{\mu + L}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \frac{1}{\mu + L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2^2$$

Use $\alpha < 2/(L + \mu)$ to obtain $\|\boldsymbol{x}_i - \boldsymbol{x}^\star\|^2 \leq \left(1 - 2t\alpha\frac{\mu L}{\mu + L}\right)\|\boldsymbol{x}_i - \boldsymbol{x}^\star\|_2^2$

Iterate over i and use $(*)$ to obtain

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star) \leq \frac{L}{2}\prod_{i\in[k]}\left(1 - 2t\alpha\frac{\mu L}{\mu + L}\right)\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|_2^2$$

Use $e^{-x} \geq 1 - x$ to conclude $f(\boldsymbol{x}_k) - f(\boldsymbol{x}^\star) \leq \frac{L}{2}e^{-ck}\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|_2^2$ for $c = \frac{2\alpha\mu L}{\mu + L}$

▸ Return