Tutorial: PART 2

Optimization for Machine Learning

Elad Hazan
Princeton University

+ help from Sanjeev Arora & Yoram Singer

Agenda

1. Learning as mathematical optimization
   - Stochastic optimization, ERM, online regret minimization
   - Online gradient descent
2. Regularization
   - AdaGrad and optimal regularization
3. Gradient Descent++
   - Frank-Wolfe, acceleration, variance reduction, second order methods, non-convex optimization

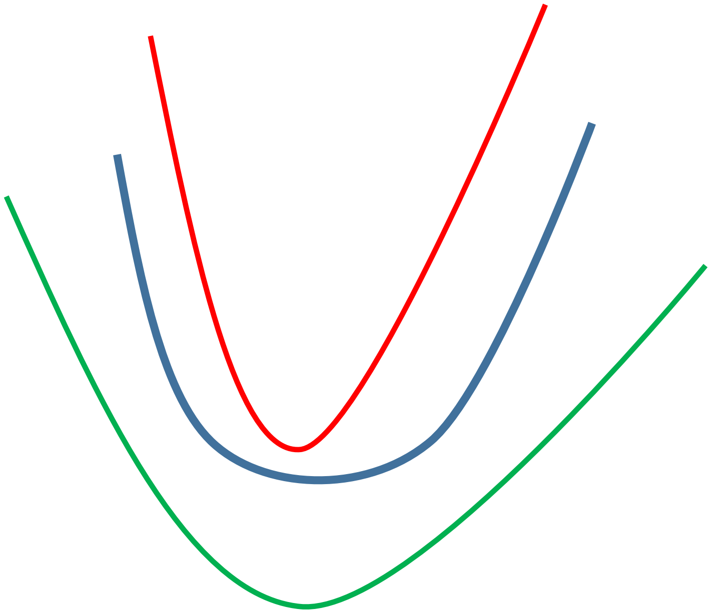
Accelerating gradient descent?

1. Adaptive regularization (AdaGrad)
   works for non-smooth&non-convex
2. Variance reduction
   uses special ERM structure
   very effective for smooth&convex
3. Acceleration/momentum
   smooth convex only, general purpose optimization
   since 80's

Condition number of convex functions

defined as $\gamma = \frac{\beta}{\alpha}$, where (simplified)

$$0 \prec \alpha I \preccurlyeq \nabla^2 f(x) \preccurlyeq \beta I$$



α = strong convexity, β = smoothness

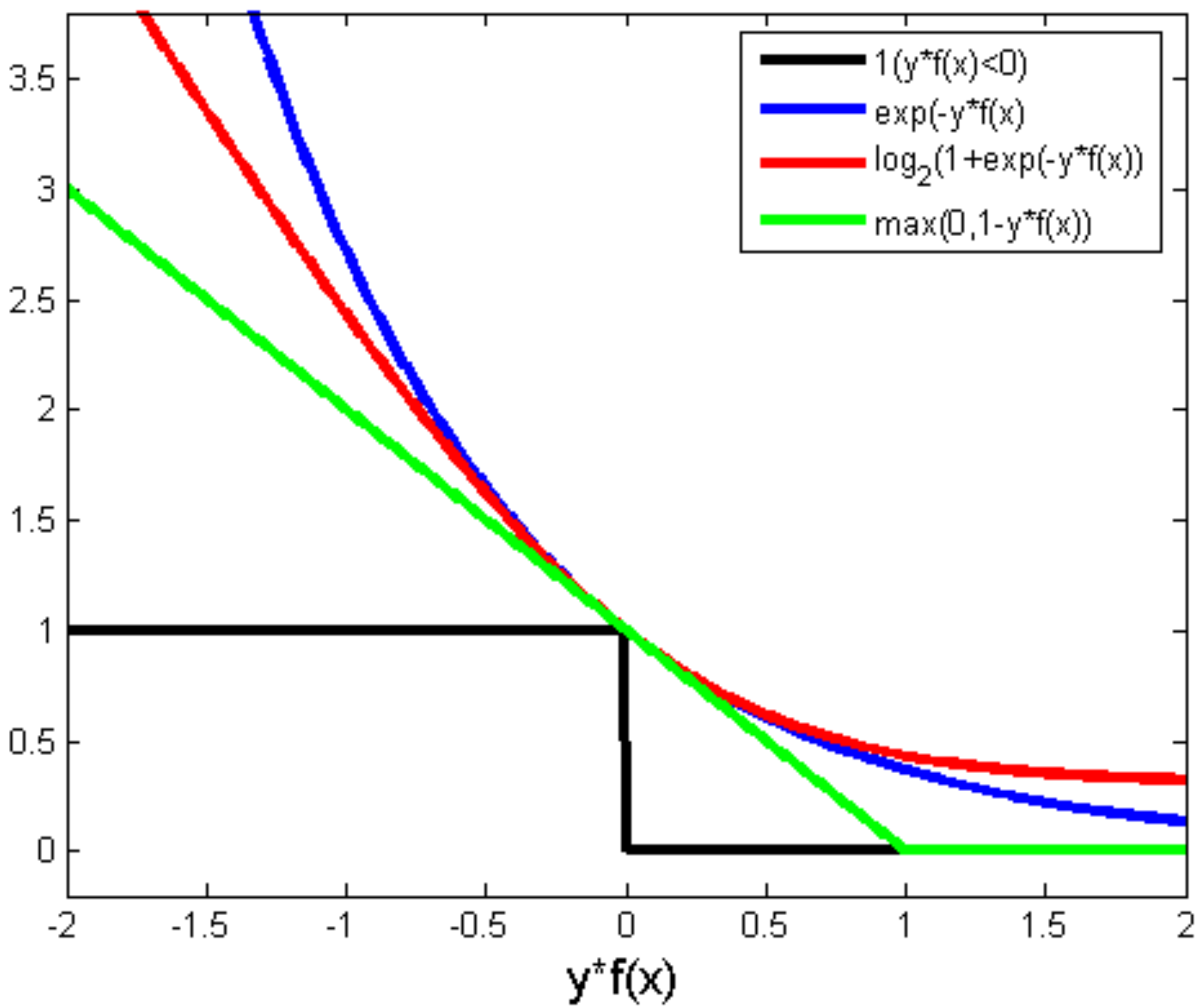
Non-convex smooth functions: (simplified)

$$-\beta I \preccurlyeq \nabla^2 f(x) \preccurlyeq \beta I$$

Why do we care?

well-conditioned functions exhibit much faster optimization!
(but equivalent via reductions)

Examples



1(y*f(x)<0)
exp(-y*f(x)
log2(1+exp(-y*f(x))
max(0,1-y*f(x))
3.5
3
2.5
2
1.5
1
0.5
0
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
y*f(x)

Smooth gradient descent

The descent lemma, β-smooth functions: (algorithm: $x_{t+1} = x_t - \eta \nabla_{\mathrm{t}}$)

$$f(x_{t+1}) - f(x_t) \leq -\nabla_t (x_{t+1} - x_t) + \beta |x_t - x_{t+1}|^2$$

$$= -(\eta + \beta \eta^2) |\nabla_t|^2 = -\frac{1}{4\beta} |\nabla_t|^2$$

Thus, for M-bounded functions: ($|f(x_t)| \leq M$)

$$-2M \leq f(x_T) - f(x_1) = \sum_t [f(x_{t+1}) - f(x_t)] \leq -\frac{1}{4\beta} \sum_t |\nabla_t|^2$$

Thus, exists a t for which,

$$|\nabla_t|^2 \leq \frac{8M\beta}{T}$$

Smooth gradient descent

Conclusions: for $x_{t+1} = x_t - \eta \nabla_{\mathrm{t}}$ and $\mathrm{T} = \Omega\left(\frac{1}{\epsilon}\right)$, finds

$$|\nabla_t|^2 \leq \epsilon$$

1. Holds even for non-convex functions
2. For convex functions implies $f(x_t) - f(x^*) \leq O(\epsilon)$ (faster for smooth!)

Non-convex stochastic gradient descent

The descent lemma, β-smooth functions: (algorithm: $x_{t+1} = x_t - \eta \tilde{\nabla}_t$)

$$E[f(x_{t+1}) - f(x_t)] \leq E[-\nabla_{\mathrm{t}}(x_{t+1} - x_t) + \beta|x_t - x_{t+1}|^2]$$

$$= E\left[-\tilde{\nabla}_t \cdot \eta \nabla_{\mathrm{t}} + \beta\left|\tilde{\nabla}_{\mathrm{t}}\right|^2\right] = -\eta \nabla_{\mathrm{t}}^2 + \eta^2 \beta E\left|\tilde{\nabla}_{\mathrm{t}}\right|^2$$

$$= -\eta \nabla_{\mathrm{t}}^2 + \eta^2 \beta(\nabla_{\mathrm{t}}^2 + var(\tilde{\nabla}_t))$$

Thus, for M-bounded functions: ($|f(x_t)| \leq M$)

$$\mathrm{T} = O\left(\frac{\mathrm{M}\beta}{\varepsilon^2}\right) \quad \Rightarrow \quad \exists_{t \leq T} . \ |\nabla_t|^2 \leq \varepsilon$$

Controlling the variance: Interpolating GD and SGD

Model: both full and stochastic gradients. Estimator combines both into lower variance RV:

$$x_{t+1} = x_t - \eta\left[\tilde{\nabla} f(x_t) - \tilde{\nabla} f(x_0) + \nabla f(x_0)\right]$$

Every so often, compute full gradient and restart at new x_0.

Theorem: [Schmidt, LeRoux, Bach '12; Johnson and Zhang '13; Mahdavi, Zhang, Jin '13]

Variance reduction for well-conditioned functions

$$0 \prec \alpha I \preccurlyeq \nabla^2 f(x) \preccurlyeq \beta I \ , \qquad \gamma = \frac{\beta}{\alpha}$$

Produces an ϵ approximate solution in time

$$O\left((m+\gamma) d \log \frac{1}{\epsilon}\right)$$

γ should be interpreted as $\frac{1}{\epsilon}$

Acceleration/momentum [Nesterov '83]

- Optimal gradient complexity (smooth, convex)

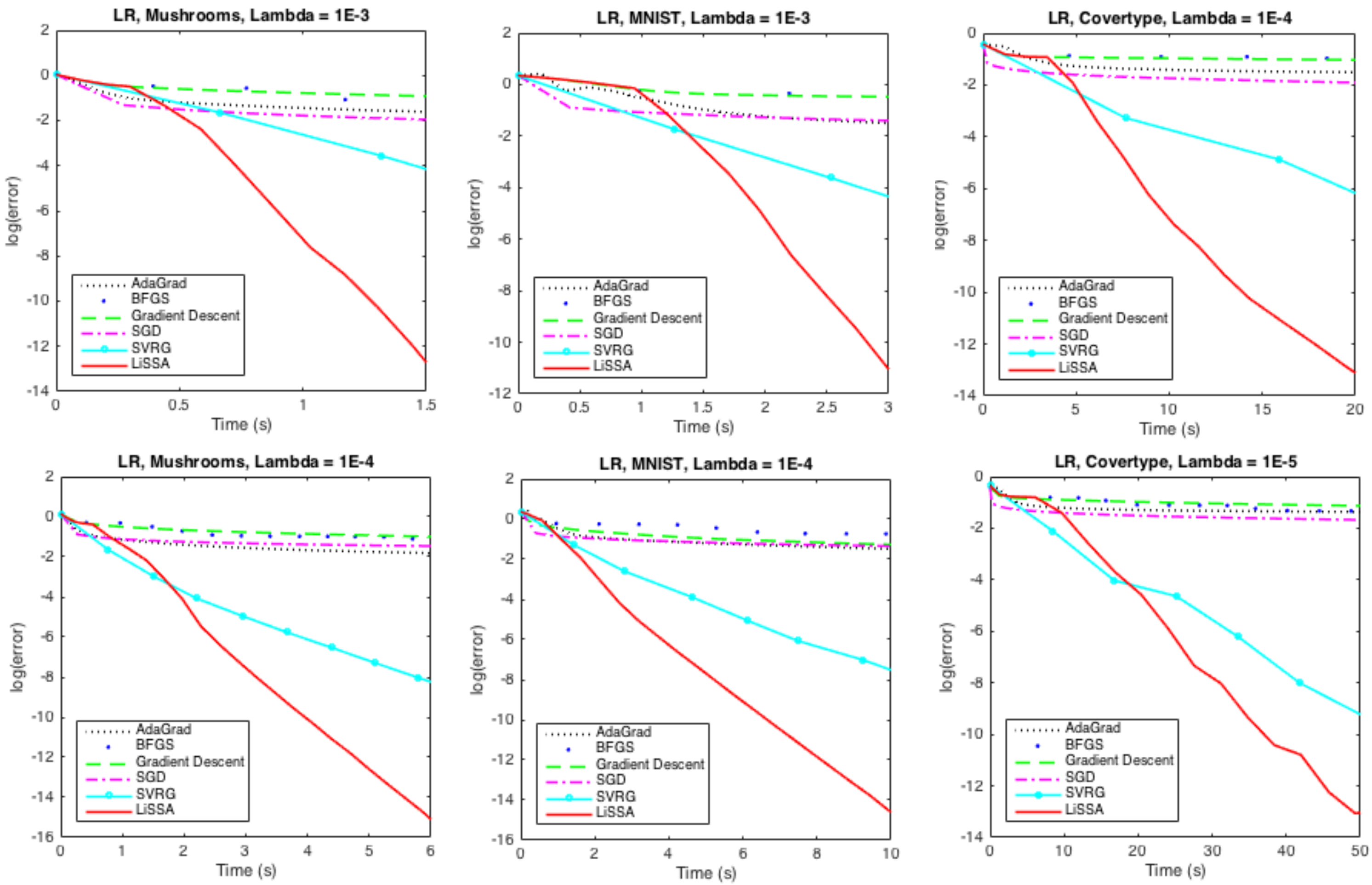
- modest practical improvements, non-convex "momentum" methods.

- With variance reduction, fastest possible running time of first-order methods:

$$O\left((m + \sqrt{\gamma m})\, d \log \frac{1}{\epsilon}\right)$$

[Woodworth, Srebro ' 15] – tight lower bound w. gradient oracle

Experiments w. convex losses



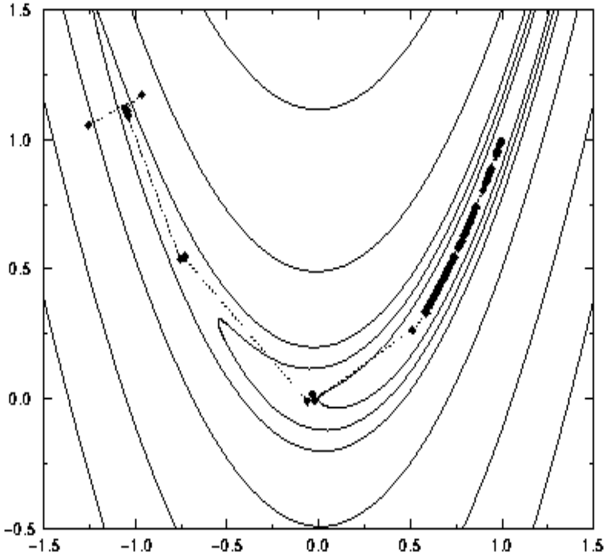
Improve upon GradientDescent++ ?

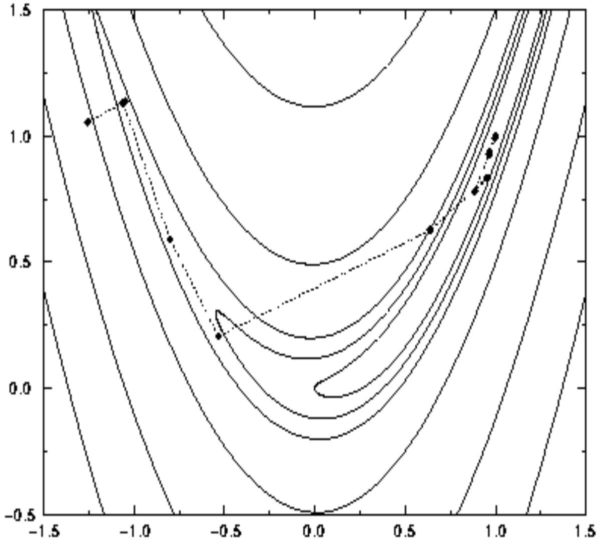
Next few slides:

Move from first order (GD++) to second order

Higher Order Optimization

- Gradient Descent – Direction of Steepest Descent
- Second Order Methods – Use Local Curvature

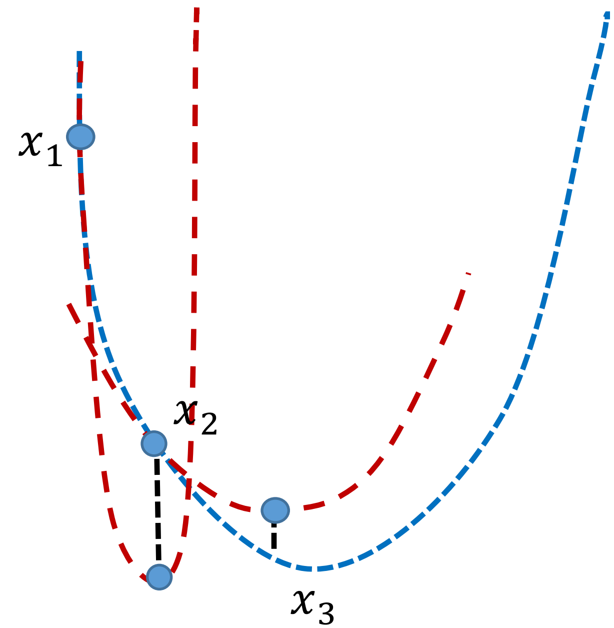




Newton's method (+ Trust region)

$$x_{t+1} = x_t - \eta\, [\nabla^2 f(x)]^{-1}\, \nabla f(x)$$

For non-convex function: can move to ∞
Solution: solve a quadratic approximation in a local area (trust region)

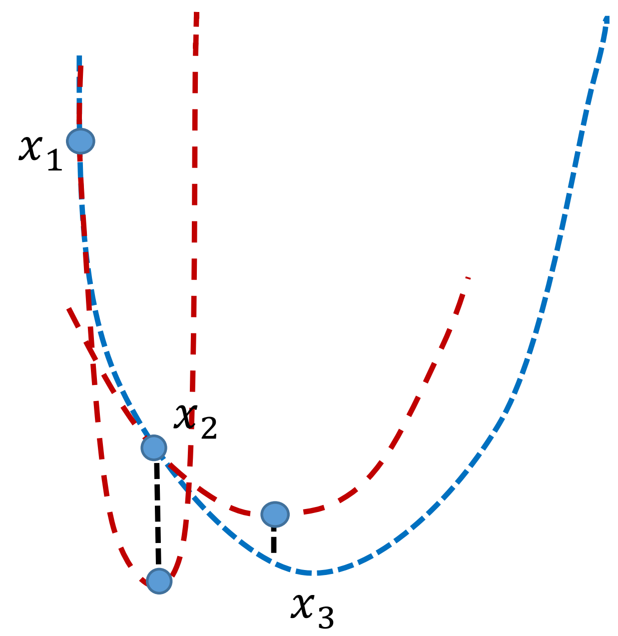


Newton's method (+ Trust region)

$$x_{t+1} = x_t - \eta\,[\nabla^2 f(x)]^{-1}\,\nabla f(x)$$

d³ time per iteration!
Infeasible for ML!!

Till recently…



Speed up the Newton direction computation??

- Spielman-Teng '04: diagonally dominant systems of equations in linear time!
  - 2015 Godel prize
  - Used by Daitch-Speilman for faster flow algorithms

- Erdogu-Montanari '15: low rank approximation & inversion by Sherman-Morisson
  - Allow stochastic information
  - Still prohibitive: rank * d^2

Stochastic Newton?

$$x_{t+1} = x_t - \eta \overbrace{[\nabla^2 f(x)]^{-1} \nabla f(x)}^{\boxed{\tilde{n}}}$$

- ERM, rank-1 loss: $\arg\min_{\mathrm{x}} E_{\{i\sim m\}}[\ell(x^T a_i, b_i) + \frac{1}{2}|x|^2]$

- unbiased estimator of the Hessian:

$$\widetilde{\nabla^2} = \mathrm{a_i a_i^T} \cdot \ell'(x^T a_i, \mathrm{b_i}) + I \qquad i \sim U[1, \dots, m]$$

- clearly $E\left[\widetilde{\nabla^2}\right] = \nabla^2 f$, but $E\left[\widetilde{\nabla^2}^{-1}\right] \neq \nabla^2 f^{-1}$

Circumvent Hessian creation and inversion!

- 3 steps:
  - (1) represent Hessian inverse as infinite series

$$\nabla^{-2} = \sum_{i=0 \text{ to } \infty} (I - \nabla^2)^i$$

For any distribution on naturals $i \sim N$

  - (2) sample from the infinite series (Hessian-gradient product) , ONCE

$$\nabla^2 f^{-1} \nabla f = \sum_i (I - \nabla^2 f)^i \, \nabla \mathrm{f} = E_{i \sim N} \, (I - \nabla^2 f)^i \, \nabla \mathrm{f} \cdot \frac{1}{\Pr[i]}$$

  - (3) estimate Hessian-power by sampling i.i.d. data examples

$$= \mathrm{E}_{i \sim N, k \sim [i]} \left[\prod_{k=1 \text{ to } i} (I - \nabla^2 f_k) \nabla \mathrm{f} \cdot \frac{1}{\Pr[i]} \right]$$

Single example
Vector-vector
products only

Linear-time Second-order Stochastic Algorithm (LiSSA)

- Use the estimator $\widetilde{\nabla^{-2} f}$ defined previously
- Compute a full (large batch) gradient $\nabla \mathrm{f}$
- Move in the direction $\widetilde{\nabla^{-2} f}\ \nabla f$
- Theoretical running time to produce an ϵ approximate solution for γ well-conditioned functions (convex): [Agarwal, Bullins, Hazan '15]

$$O\left(dm \log \frac{1}{\epsilon} + \sqrt{\gamma d}\ d \log \frac{1}{\epsilon}\right)$$

1. Faster than first-order methods!
2. Indications this is tight [Arjevani,Shamir '16]

What about constraints??

Next few slides – projection free (Frank-Wolfe) methods

Recommendation systems

movies

users

1		5			
			2	3	
	2	4			1
			5		
	4			2	3
1			1		1
	5			5	

rating of user i
for movie j

Recommendation systems

movies

users

1	4	5	2	1	5
3	4	2	2	3	5
5	2	4	4	1	1
3	3	3	3	3	3
2	3	1	5	4	4
3	4	2	1	2	3
1	2	4	1	5	1
4	5	3	3	5	2

complete missing entries

Recommendation systems

movies

users

1	4	5	2	1	5
3	4	2	2	3	5
5	2	4	4	1	1
3	3	3	3	3	5
2	3	1	5	4	4
3	4	2	1	2	3
1	2	4	1	5	1
4	5	3	3	5	2

get new data

Recommendation systems

movies

users

1	2	5	5	4	4
2	4	3	2	3	1
5	2	4	2	3	1
3	3	2	4	5	5
3	3	2	5	1	5
2	4	3	4	2	3
1	1	1	1	1	1
4	5	3	3	5	2

re-complete missing entries

Assume low rank of “true matrix”, convex relaxation: bounded trace norm

Bounded trace norm matrices

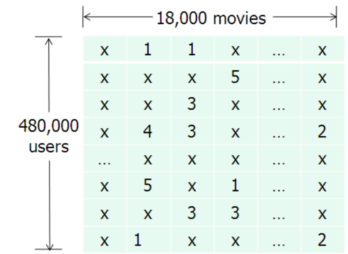
- Trace norm of a matrix = sum of singular values
- K = { X | X is a matrix with trace norm at most D }

- Computational bottleneck: projections on K require eigendecomposition: $O(n^3)$ operations

- But: linear optimization over K is easier computing top eigenvector; $O(\text{sparsity})$ time

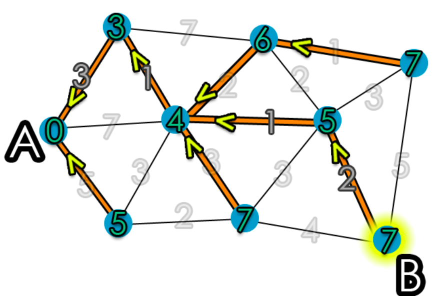
Projections → linear optimization

1. Matrix completion (K = bounded trace norm matrices)
~~eigen decomposition~~



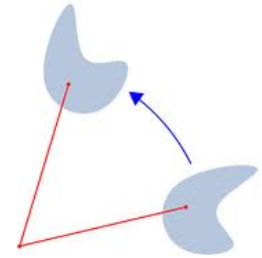
x	1	1	x	...	x
x	x	x	5	...	x
x	x	3	x	...	x
x	4	3	x	...	2
...	x	x	x	...	x
x	5	x	1	...	x
x	x	3	3	...	x
x	1	x	x	...	2

2. Online routing (K = flow polytope)
~~conic optimization over flow polytope~~



3. Rotations (K = rotation matrices)
~~conic optimization over rotations set~~

4. Matroids (K = matroid polytope)
~~convex opt. via ellipsoid method~~

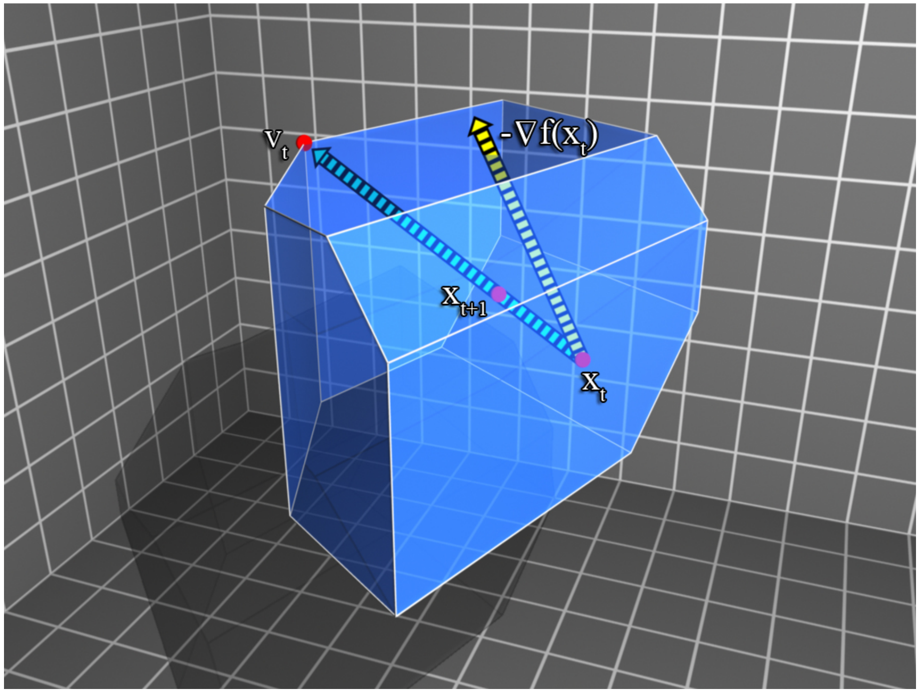


Conditional Gradient algorithm [Frank, Wolfe '56]

Convex opt problem:

$$\min_{x \in K} f(x)$$

- f is smooth, convex
- linear opt over K is easy

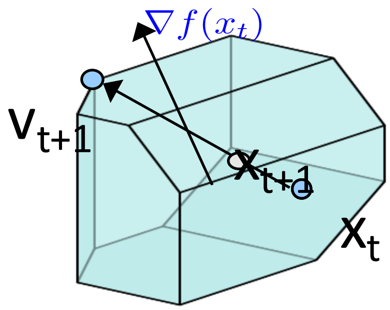


$$v_t = \arg\min_{x \in K} \nabla f(x_t)^\top x$$

$$x_{t+1} = x_t + \eta_t (v_t - x_t)$$

1. At iteration t: convex comb. of at most t vertices **(sparsity)**
2. No learning rate. $\eta_t \approx \frac{1}{t}$ (independent of diameter, gradients etc.)

FW theorem



$$x_{t+1} = x_t + \eta_t(v_t - x_t) \ , \ v_t = \arg\min_{x \in K} \nabla_t^\top x$$

Theorem: $$f(x_t) - f(x^*) = O(\frac{1}{t})$$

Proof, main observation:

$$f(x_{t+1}) - f(x^*) = f(x_t + \eta_t(v_t - x_t)) - f(x^*)$$

$$\leq f(x_t) - f(x^*) + \eta_t(v_t - x_t)^\top \nabla_t + \eta_t^2 \frac{\beta}{2} \|v_t - x_t\|^2 \qquad \beta\text{-smoothness of } f$$

$$\leq f(x_t) - f(x^*) + \eta_t(x^* - x_t)^\top \nabla_t + \eta_t^2 \frac{\beta}{2} \|v_t - x_t\|^2 \qquad \text{optimality of } v_t$$

$$\leq f(x_t) - f(x^*) + \eta_t(f(x^*) - f(x_t)) + \eta_t^2 \frac{\beta}{2} \|v_t - x_t\|^2 \qquad \text{convexity of } f$$

$$\leq (1 - \eta_t)(f(x_t) - f(x^*)) + \frac{\eta_t^2 \beta}{2} D^2.$$

Thus: $h_t = f(x_t) - f(x^*)$

$$h_{t+1} \leq (1 - \eta_t) h_t + O(\eta_t^2) \quad \Rightarrow \quad \eta_t, h_t = O(\frac{1}{t})$$

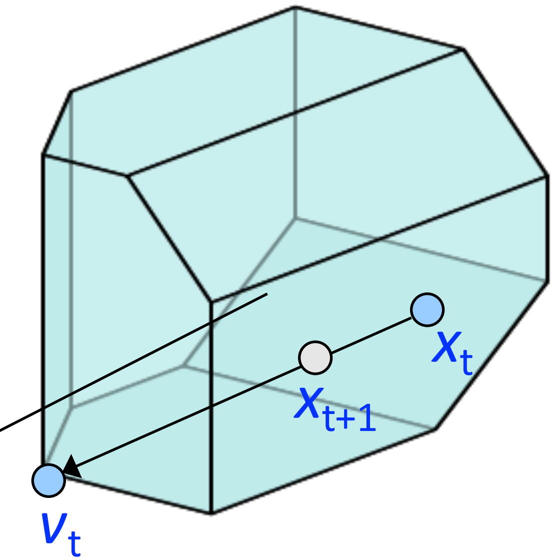
Online Conditional Gradient

- Set $x_1 \in K$ arbitrarily
- For $t = 1, 2, \ldots,$
  1. Use x_t, obtain f_t
  2. Compute x_{t+1} as follows

$$v_t = \arg\min_{x \in K} \left(\sum_{i=1}^{t} \nabla f_i(x_i) + \beta_t x_t \right)^{\top} x$$

$$x_{t+1} \leftarrow (1 - t^{-\alpha}) x_t + t^{-\alpha} v_t$$

$\sum_{i=1}^{t} \nabla f_i(x_i) + \beta_t x_t$



Theorem:[Hazan, Kale '12] $Regret = O(T^{3/4})$

Theorem: [Garber, Hazan '13] For polytopes, strongly-convex and smooth losses,

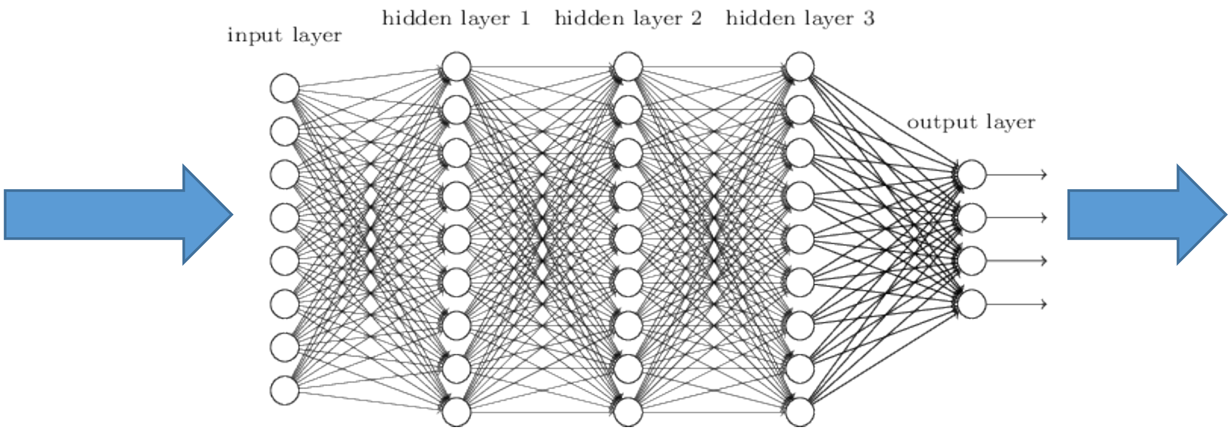
1. Offline: convergence after t steps: $e^{-\Omega(t)}$
2. Online: $Regret = O(\sqrt{T})$

Next few slides:
survey state-of-the-art

Non-convex optimization in ML

Machine

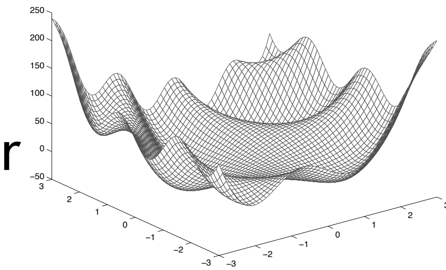
Distribution over $\{a\} \in R^n$



label

$$\arg\min_{x \in R^d} \frac{1}{m} \sum_{i=1 \; to \; m} \ell_i(x, a_i, b_i) + R(x)$$

Solution concepts for non-convex optimization?



- Global minimization is NP hard, even for degree 4 polynomials. Even local minimization up to 4th order optimality conditions is NP hard.

- Algorithmic stability is sufficient [Hardt, Recht, Singer '16]

- Optimization approaches:
  - Finding vanishing gradients / local minima efficiently
  - Graduated optimization / homotopy method
  - Quasi-convexity
  - Structure of local optima (probabilistic assumptions that allow alternating minimization,…)

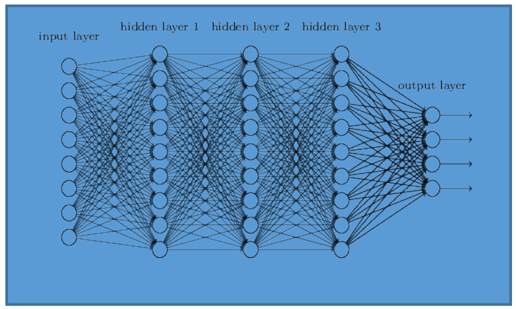
Gradient/Hessian based methods

Goal: find point x such that

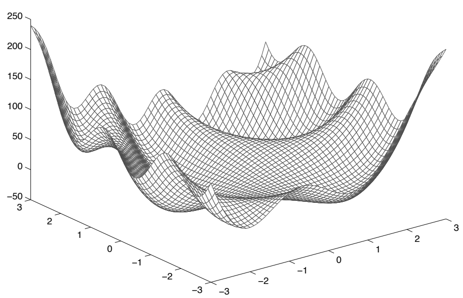
1. $|\nabla f(x)| \leq \varepsilon$ (approximate first order optimality)
2. $\nabla^2 f(x) \succcurlyeq -\varepsilon I$ (approximate second order optimality)

1. (we've proved) GD algorithm: $x_{t+1} = x_t - \eta \nabla_{\mathrm{t}}$ finds in $O\left(\frac{1}{\epsilon}\right)$ (expensive) iterations point (1)
2. (we've proved) SGD algorithm: $x_{t+1} = x_t - \eta \tilde{\nabla}_{\mathrm{t}}$ finds in $O\left(\frac{1}{\epsilon^2}\right)$ (cheap) iterations point (1)
3. SGD algorithm with noise finds in $O\left(\frac{1}{\epsilon^4}\right)$ (cheap) iterations (1&2) [Ge, Huang, Jin, Yuan '15]
4. Recent second order methods: find in $O\left(\frac{1}{\epsilon^{7/8}}\right)$ (expensive) iterations point (1&2)
   [Carmon,Duchi, Hinder, Sidford '16]
   [Agarwal, Allen-Zuo, Bullins, Hazan, Ma '16]

Recap



Chair/car



1. Online learning and stochastic optimization
   - Regret minimization
   - Online gradient descent
2. Regularization
   - AdaGrad and optimal regularization
3. Advanced optimization
   - Frank-Wolfe, acceleration, variance reduction, second order methods, non-convex optimization

Bibliography & more information, see:

http://www.cs.princeton.edu/~ehazan/tutorial/SimonsTutorial.htm

Thank you!