Tutorial: PART 1

Optimization for machine learning

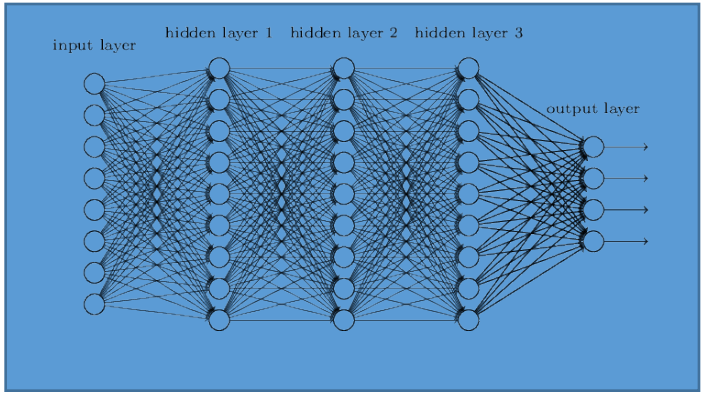
Elad Hazan
Princeton University

+ help from Sanjeev Arora, Yoram Singer

ML paradigm

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

Machine



Chair/car

label

$b = f_{parameters}(a)$

This tutorial - training the machine

- Efficiency
- generalization

Agenda

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

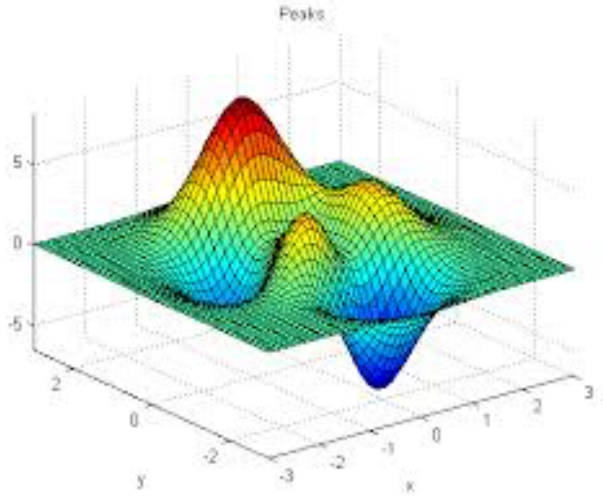
NOT touch upon:

- Parallelism/distributed computation (asynchronous optimization, HOGWILD etc.), Bayesian inference in graphical models, Markov-chain-monte-carlo, Partial information and bandit algorithms

Mathematical optimization

Input: function $f: K \mapsto R$, for $K \subseteq R^d$

Output: minimizer $x \in K$, such that $f(x) \leq f(y)\ \forall\ y \in K$

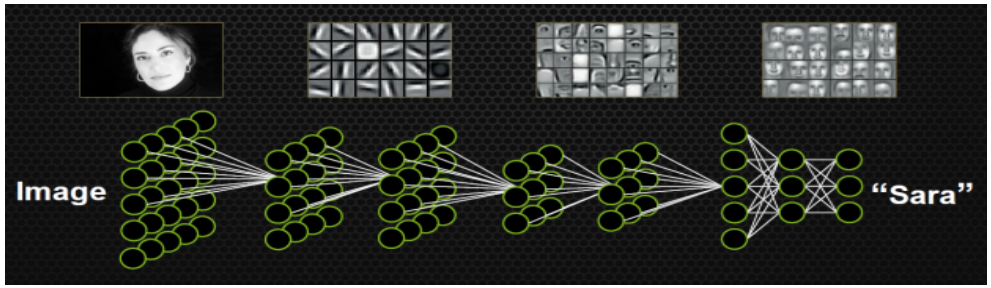


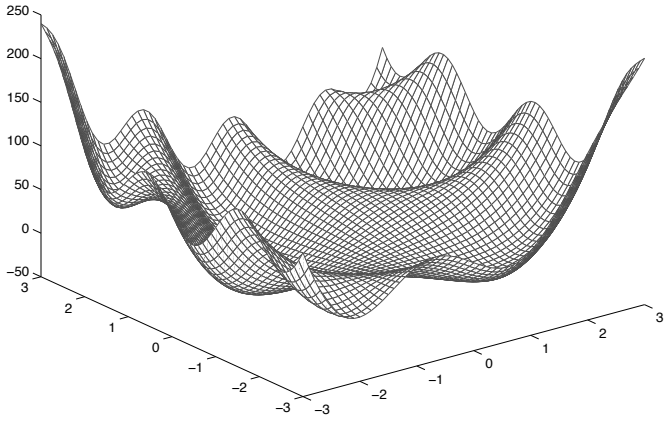
Accessing f? (values, differentials, …)

Generally NP-hard, given full access to function.

Learning = optimization over data (a.k.a. Empirical Risk Minimization)

Fitting the parameters of the model ("training") = optimization problem:



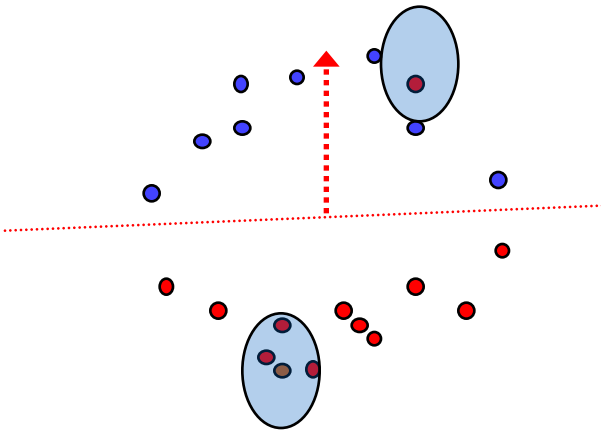


$$\arg\min_{x \in R^d} \frac{1}{m} \sum_{i=1\ to\ m} \ell_i(x, a_i, b_i) + R(x)$$

m = # of examples (a,b) = (features, labels)
d = dimension

Example: linear classification

Given a sample $S = \{(a_1, b_1), \ldots, (a_m, b_m)\}$, find hyperplane (through the origin w.l.o.g) such that:



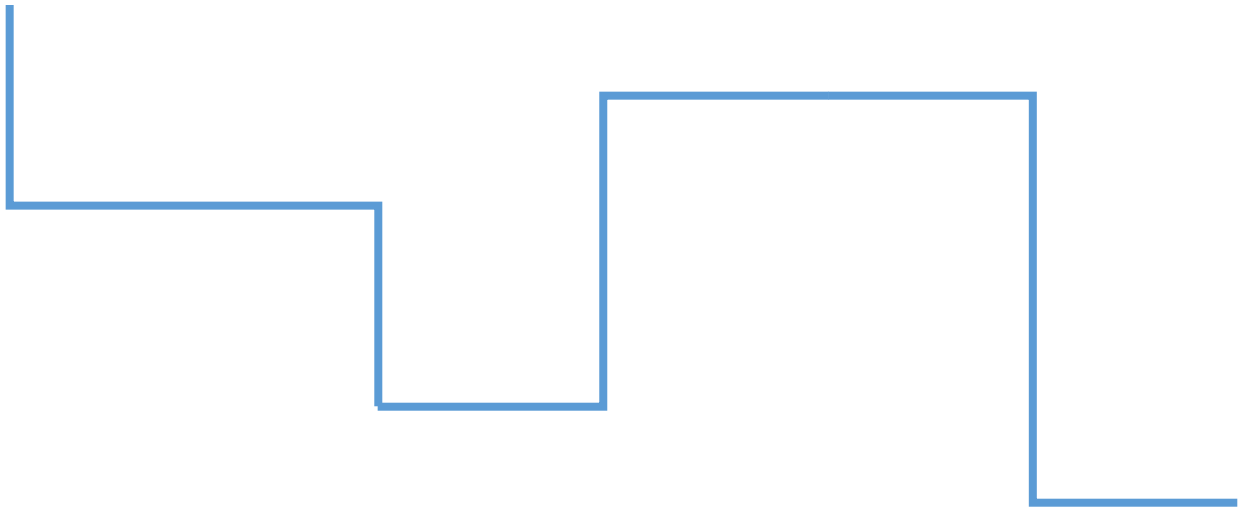
$x = \arg\min_{|x|\leq 1}$ # of mistakes =

$$\arg\min_{|x|\leq 1} |\{i \;\; s.t.\; sign\,(x^T a_i) \neq b_i\}|$$

$$\arg\min_{|x|\leq 1} \frac{1}{m}\sum_i \ell(x, a_i, b_i) \quad \text{for} \quad \ell(x, a_i, b_i) = \begin{cases} 1 & x^\top a \neq b \\ 0 & x^\top a = b \end{cases}$$

NP hard!

Sum of signs → global optimization NP-hard!
but locally verifiable…



Local property that ensures global optimality?

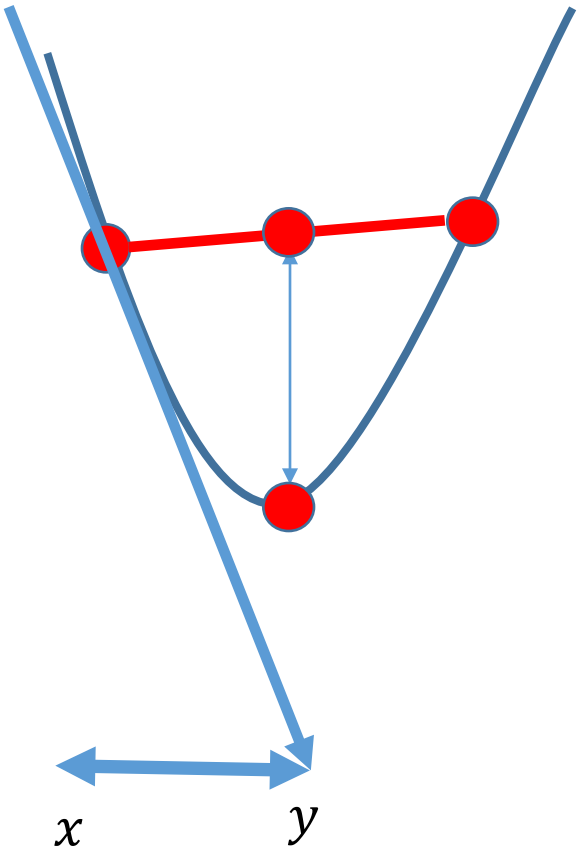
Convexity

A function $f: R^d \mapsto R$ is convex if and only if:

$$f\left(\frac{1}{2}x + \frac{1}{2}y\right) \leq \frac{1}{2}f(x) + \frac{1}{2}f(y)$$

- Informally: smiley ☺
- Alternative definition:

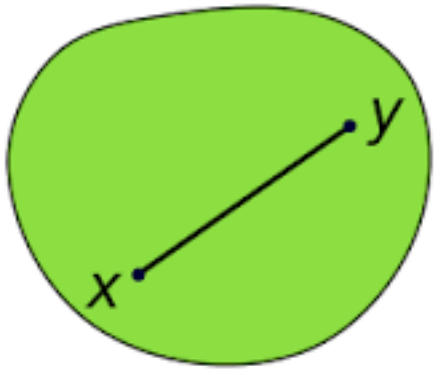
$$\mathrm{f(y)} \geq \mathrm{f(x)} + \nabla f(x)^{\mathsf{T}}(y - x)$$

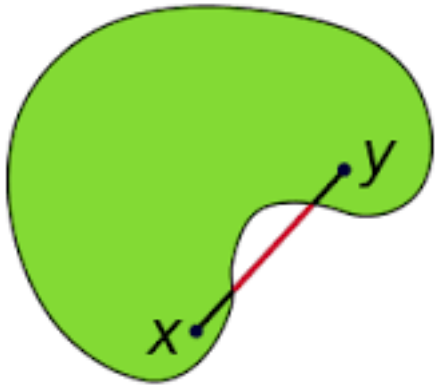


Convex sets

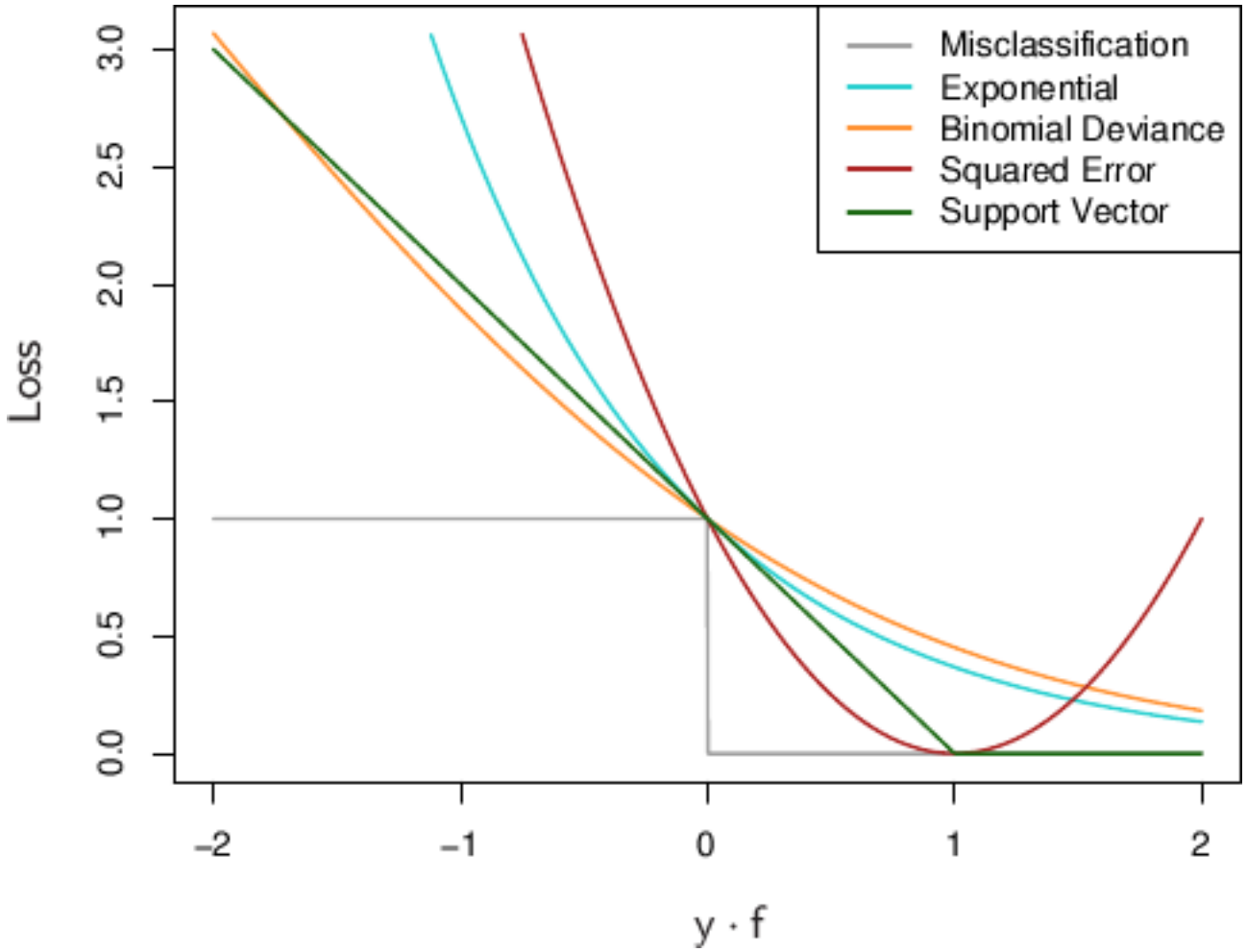
Set K is convex if and only if:

$x, y \in K \Rightarrow (½x + ½y) \in K$



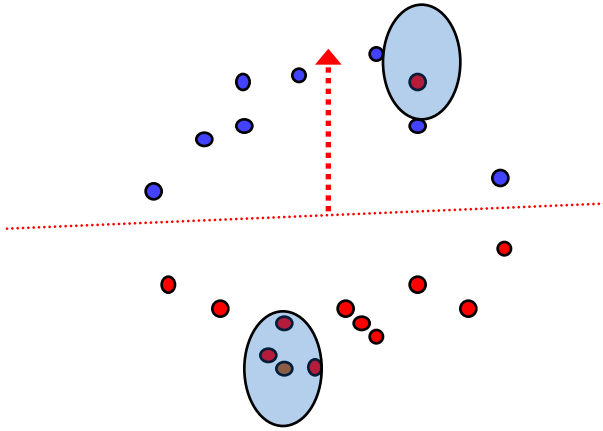


Loss functions $\ell(x, a_i, b_i) = \ell(x^T a_i \cdot b_i)$



Convex relaxations for linear (&kernel) classification

$$x = \arg\min_{|x|\leq 1} |\{i \;\; \text{s.t. } sign\,(x^T a_i) \neq b_i\}|$$



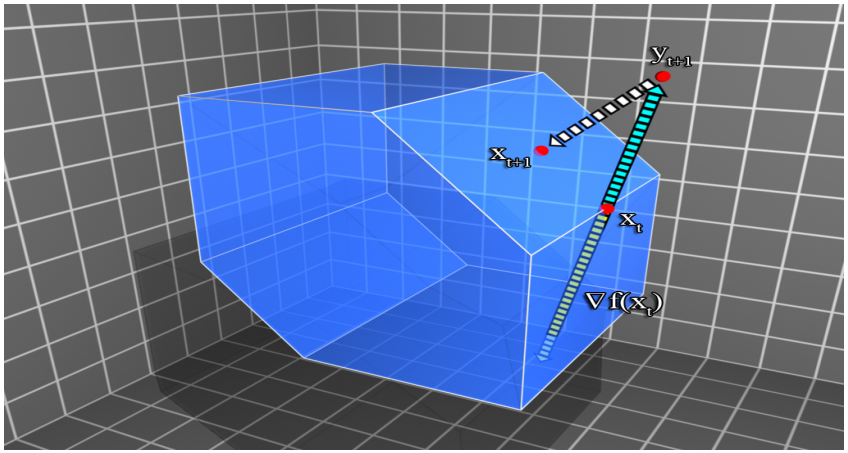
1. Ridge / linear regression $\ell(x^\top a_i, y_i) = (x^\top a_i - b_i)^2$
2. SVM $\ell(x^\top a_i, y_i) = \max\{0, 1 - b_i \;\; x^\top a_i\}$
3. Logistic regression $\ell(x^\top a_i, y_i) = \log(1 + e^{-b_i \cdot x^\top a_i})$

We have: cast learning as mathematical optimization, argued convexity is algorithmically important

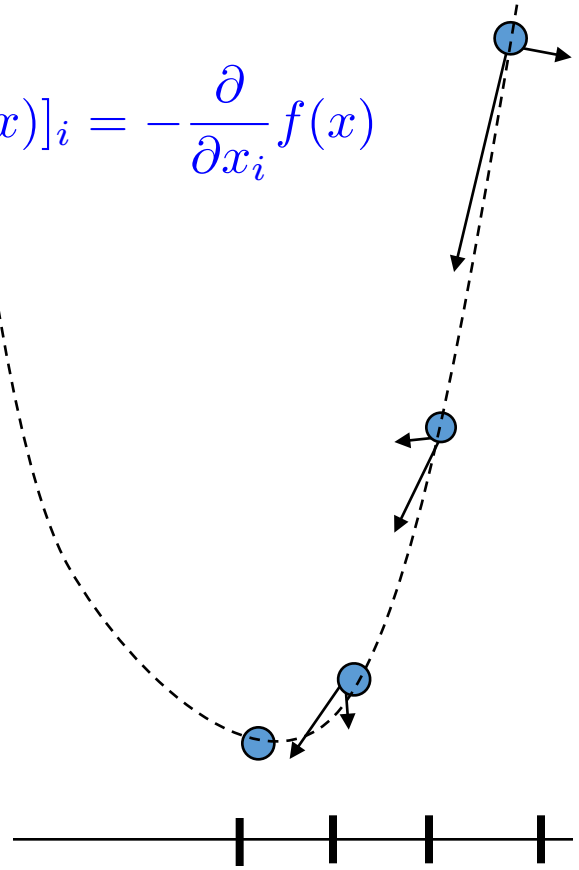
Next ➔ algorithms!

Gradient descent, constrained set

$$y_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = \arg\min_{x \in K} |y_{t+1} - x|$$



$$-[\nabla f(x)]_i = -\frac{\partial}{\partial x_i} f(x)$$



Convergence of gradient descent

$$y_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = \arg\min_{x \in K} |y_{t+1} - x|$$

Theorem: for step size $\eta = \frac{D}{G\sqrt{T}}$

$$f\left(\frac{1}{T}\sum_t x_t\right) \leq \min_{x^* \in K} f(x^*) + \frac{DG}{\sqrt{T}}$$

Where:

- G = upper bound on norm of gradients

$$|\nabla f(x_t)| \leq G$$

- D = diameter of constraint set

$$\forall x, y \in K \ . \ |x - y| \leq D$$

Proof:

$$y_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = \arg\min_{x \in K} |y_{t+1} - x|$$

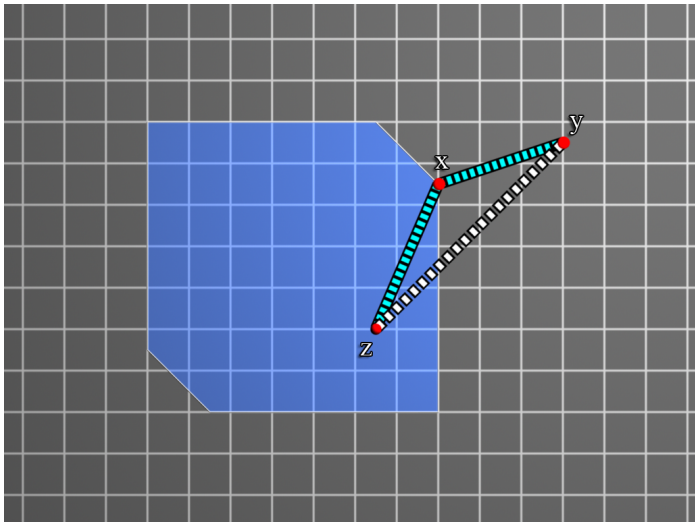
1. Observation 1:

$$|x^* - y_{t+1}|^2 = |x^* - x_t|^2 - 2\eta \nabla f(x_t)(x_t - x^*) + \eta^2 |\nabla f(x_t)|^2$$

2. Observation 2:

$$|x^* - x_{t+1}|^2 \leq |x^* - y_{t+1}|^2$$

This is the Pythagorean theorem:



Proof:

$$y_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$
$$x_{t+1} = \arg\min_{x \in K} |y_{t+1} - x|$$

1. Observation 1:

$$|x^* - y_{t+1}|^2 = |x^* - x_t|^2 - 2\eta \nabla f(x_t)(x_t - x^*) + \eta^2 |\nabla f(x_t)|^2$$

2. Observation 2:

$$|x^* - x_{t+1}|^2 \leq |x^* - y_{t+1}|^2$$

Thus:

$$|x^* - x_{t+1}|^2 \leq |x^* - x_t|^2 - 2\eta \nabla f(x_t)(x_t - x^*) + \eta^2 G^2$$

And hence:

$$f\left(\frac{1}{T}\sum_t x_t\right) - f(x^*) \leq \frac{1}{T}\sum_t [f(x_t) - f(x^*)] \leq \frac{1}{T}\sum_t \nabla f(x_t)(x_t - x^*)$$

$$\leq \frac{1}{T}\sum_t \frac{1}{2\eta}(|x^* - x_{t+1}|^2 - |x^* - x_t|^2) + \frac{\eta}{2}G^2$$

$$\leq \frac{1}{T \cdot 2\eta} D^2 + \frac{\eta}{2} G^2 \leq \frac{DG}{\sqrt{T}}$$

Recap

Theorem: for step size $\eta = \frac{D}{G\sqrt{T}}$

$$f\left(\frac{1}{T}\sum_t x_t\right) \leq \min_{x^* \in K} f(x^*) + \frac{DG}{\sqrt{T}}$$

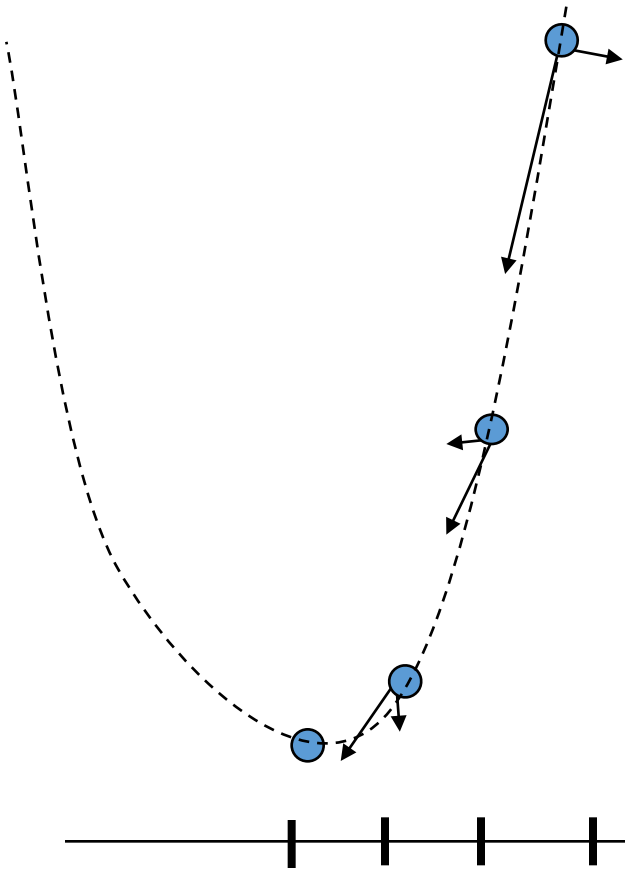
Thus, to get ϵ-approximate solution, apply $O\left(\frac{1}{\epsilon^2}\right)$ gradient iterations.

Gradient Descent - caveat

For ERM problems

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

1. Gradient depends on all data
2. What about generalization?



Next few slides:

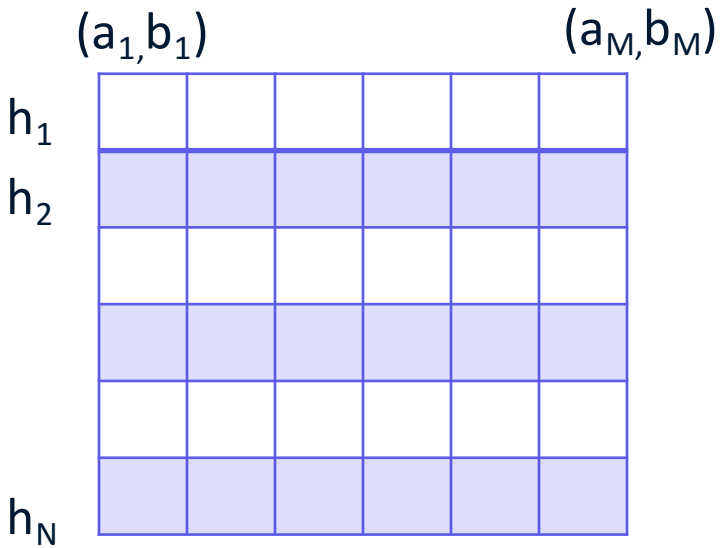
Simultaneous optimization and generalization

➔ Faster optimization! (single example per iteration)

Statistical (PAC) learning

Nature: i.i.d from distribution D over

$A \times B = \{(a, b)\}$



learner:

Hypothesis h

Loss, e.g. $\ell\big(h, (a, b)\big) = (h(a) - b)^2$

$$err(h) = \mathbb{E}_{a,b\sim D}[\ell(h, (a, b)]$$

Hypothesis class H: X -> Y is learnable if $\forall \epsilon, \delta > 0$ exists algorithm s.t. after seeing m examples, for $m = poly(\delta, \epsilon, dimension(H))$

finds h s.t. w.p. 1- δ:

$$\mathrm{err}(h) \leq \min_{h^* \in \mathcal{H}} \mathrm{err}(h^*) + \epsilon$$

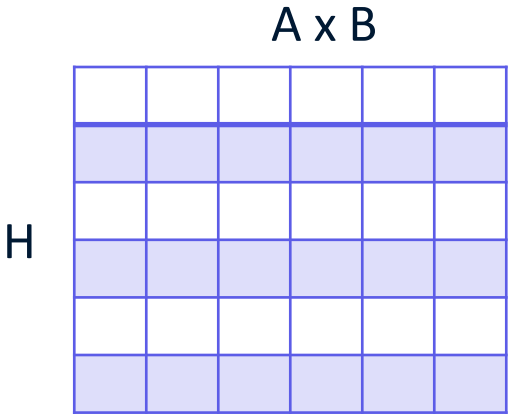
More powerful setting: Online Learning in Games

Iteratively, for $t = 1,2, \ldots, T$

Player: $h_t \in H$

Adversary: $(a_t, b_t) \in A$

Loss $\ell(h_t, (a_t, b_t))$



Goal: minimize (average, expected) regret:

$$\frac{1}{T}\left[\sum_t \ell(h_t, (a_t, b_t) - \min_{h^* \in \mathcal{H}} \sum_t \ell(h^*, (a_t, b_t))\right] \underset{T \to \infty}{\longrightarrow} 0$$

Vanishing regret → generalization in PAC setting! (online2batch)

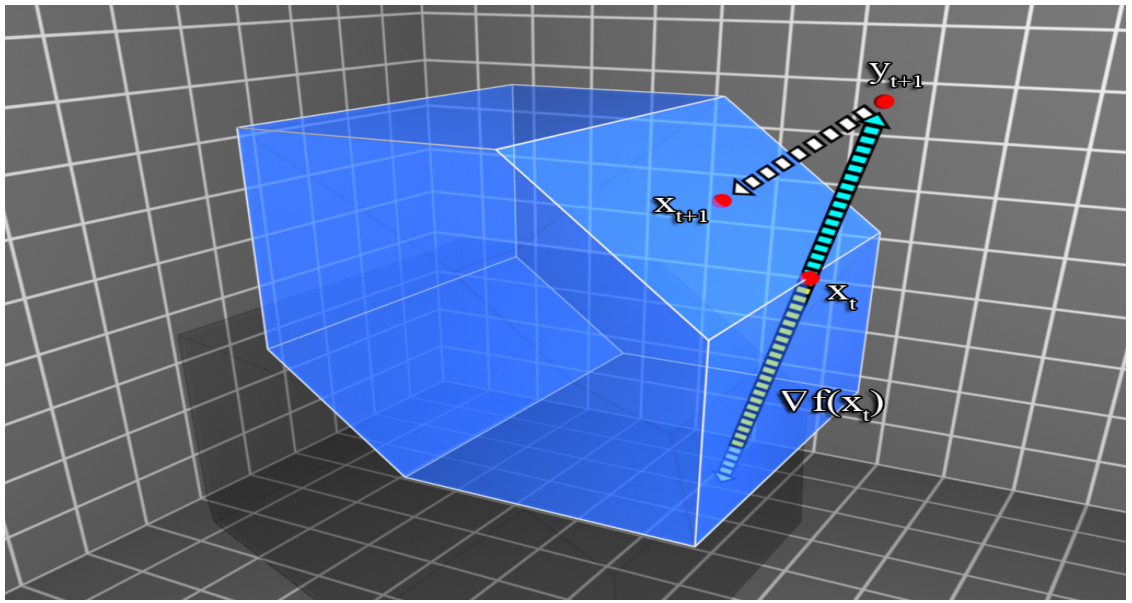
From this point onwards: $f_t(x) = \ell(x, a_t, b_t)$ = loss for one example

Can we minimize regret efficiently?

Online gradient descent [Zinkevich '05]

$$y_{t+1} = x_t - \eta \nabla f_t(x_t)$$

$$x_{t+1} = \arg\min_{x \in K} \|y_{t+1} - x_t\|$$



Theorem: Regret = $\sum_t f_t(x_t) - \sum_t f_t(x^*) = O(\sqrt{T})$

Analysis

$\nabla_t := \nabla f_t(x_t)$

Observation 1:

$$\|y_{t+1} - x^*\|^2 = \|x_t - x^*\|^2 - 2\eta\nabla_t(x^* - x_t) + \eta^2\|\nabla_t\|^2$$

Observation 2: (Pythagoras)

$$\|x_{t+1} - x^*\| \leq \|y_{t+1} - x^*\|$$

Thus: $\|x_{t+1} - x^*\|^2 \leq \|x_t - x^*\|^2 - 2\eta\nabla_t(x^* - x_t) + \eta^2\|\nabla_t\|^2$

Convexity: $\sum_t [f_t(x_t) - f_t(x^*)] \leq \sum_t \nabla_t(x_t - x^*)$

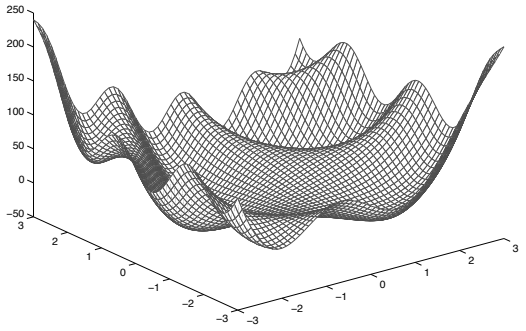
Lower bound

$$\text{Regret} = \Omega(\sqrt{T})$$

- 2 loss functions, T iterations:
  - $K = [-1,1],\ f_1(x) = x\ ,\ f_2(x) = -x$
  - Second expert loss = first * -1
- Expected loss = 0 (any algorithm)
- Regret = (compared to either -1 or 1)

$$E[|\#1's - \#(-1)'s|] = \Omega(\sqrt{T})$$

Stochastic gradient descent



Learning problem $\arg\min_{x \in R^d} F(x) = E_{(a_i,b_i)}[\ell_i(x, a_i, b_i)]$

random example: $f_t(x) = \ell_i(x, a_i, b_i)$

1. We have proved: (for any sequence of ∇_t)

$$\frac{1}{T}\sum_t \nabla_t^\top x_t \leq \min_{x^* \in K} \frac{1}{T}\sum_t \nabla_t^\top x^* + \frac{DG}{\sqrt{T}}$$

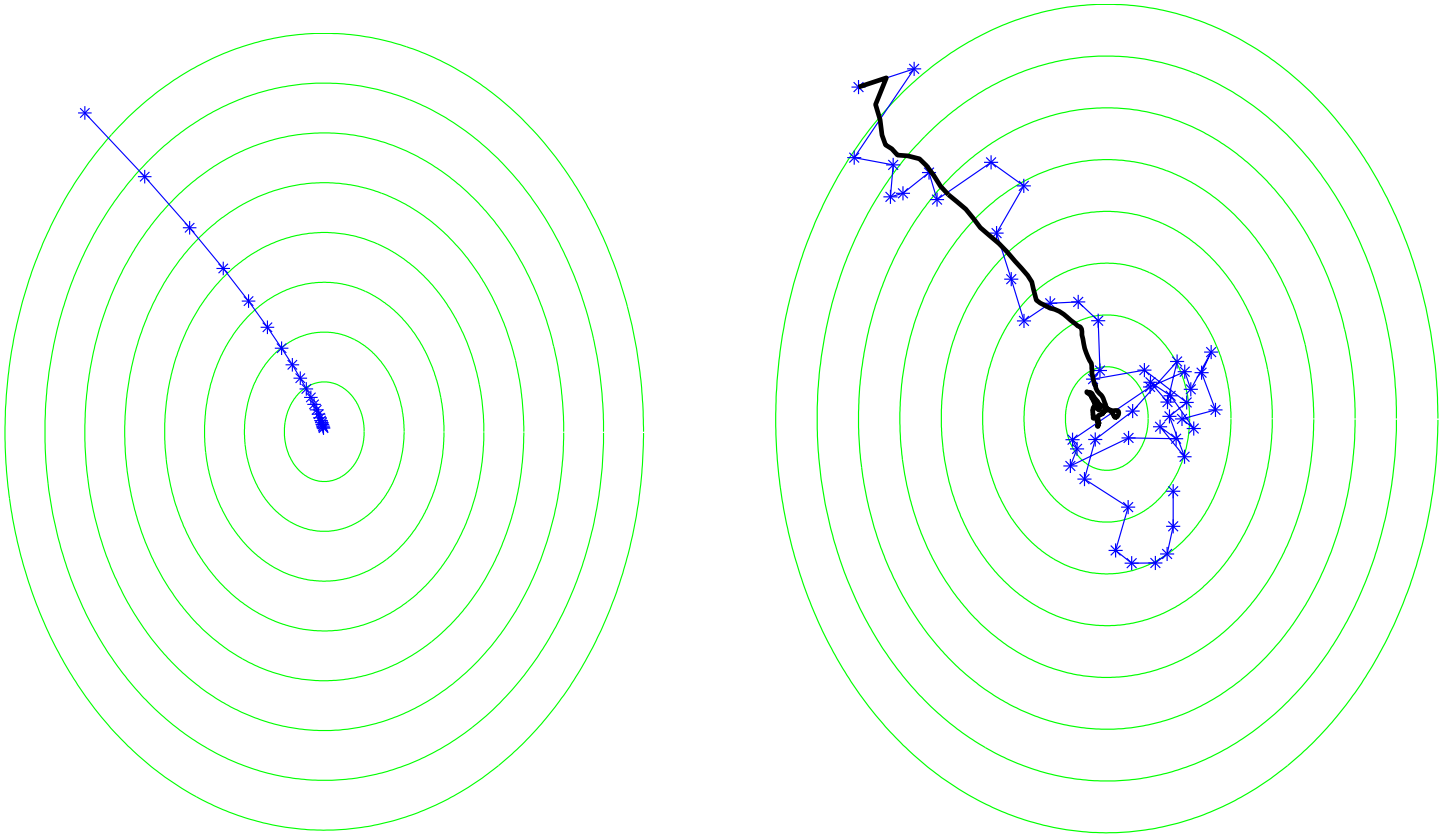
2. Taking (conditional) expectation:

$$E\left[F\left(\frac{1}{T}\sum_t x_t\right) - \min_{x^* \in K} F(x^*)\right] \leq E\left(\frac{1}{T}\sum_t \nabla_t^\top (x_t - x^*)]\right) \leq \frac{DG}{\sqrt{T}}$$

One example per step, same convergence as GD, & gives direct generalization! (formally needs martingales)

$O\left(\frac{d}{\epsilon^2}\right)$ vs. $O\left(\frac{md}{\epsilon^2}\right)$ total running time for ϵ generalization error.

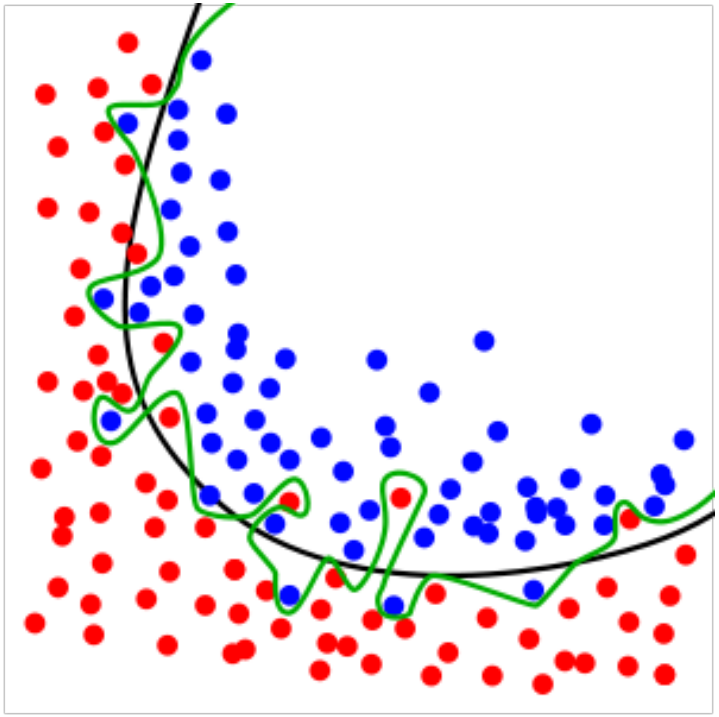
Stochastic vs. full gradient descent



Regularization & Gradient Descent++

Why "regularize"?

- Statistical learning theory / Occam's razor:
  # of examples needed to learn hypothesis class ~ it's "dimension"
  - VC dimension
  - Fat-shattering dimension
  - Rademacher width
  - Margin/norm of linear/kernel classifier



- PAC theory: Regularization <-> reduce complexity
- Regret minimization: Regularization <-> stability

Minimize regret: best-in-hindsight

$$\text{Regret} = \sum_t f_t(x_t) - \min_{x^* \in K} \sum_t f_t(x^*)$$

- Most natural:

$$x_t = \arg\min_{x \in K} \sum_{i=1}^{t-1} f_i(x)$$

- Provably works [Kalai-Vempala'05]:

$$x'_t = \arg\min_{x \in K} \sum_{i=1}^{t} f_i(x) = x_{t+1}$$

- So if $x_t \approx x_{t+1}$, we get a regret bound
- But instability $|x_t - x_{t+1}|$ can be large!

Fixing FTL: Follow-The-Regularized-Leader (FTRL)

- Linearize: replace f_t by a linear function, $\nabla f_t(x_t)^T x$
- Add **regularization:**

$$x_t = \arg\min_{x \in K} \sum_{i=1 \ldots t-1} \nabla_t^\top x \; + \frac{1}{\eta} R(x)$$

- *R(x)* is a strongly convex function, ensures stability:

$$\nabla_t^\top (x_t \; - x_{t+1}) = O(\eta)$$

FTRL vs. gradient descent

- $R(x) = \frac{1}{2} \| x \|^2$

$$x_t = \arg\min_{x \in K} \sum_{i=1}^{t-1} \nabla f_i(x_i)^\top x + \frac{1}{\eta} R(x)$$

$$= \prod_K \left(-\eta \sum_{i=1}^{t-1} \nabla f_i(x_i) \right)$$

- *Essentially* OGD: starting with $y_1 = 0$, for $t = 1, 2, \ldots$

$$x_t = \prod_K (y_t)$$

$$y_{t+1} = y_t - \eta \nabla f_t(x_t)$$

FTRL vs. Multiplicative Weights

- Experts setting: $K = \Delta_n$ distributions over experts
- $f_t(x) = c_t^T x$, where c_t is the vector of losses
- $R(x) = \sum_i x_i \log x_i$: negative entropy

$$x_t = \arg\min_{x \in K} \sum_{i=1}^{t-1} \nabla f_i(x_i)^\top x + \frac{1}{\eta} R(x)$$

$$= \exp\left(-\eta \sum_{i=1}^{t-1} c_i\right) / Z_t$$

Entrywise exponential

Normalization constant

- Gives the Multiplicative Weights method!

FTRL ⇔ Online Mirror Descent

$$x_t = \arg\min_{x \in K} \sum_{i=1}^{t-1} \nabla f_i(x_i)^\top x + \frac{1}{\eta} R(x)$$

Bregman Projection:

$$\prod_K^R(y) = \arg\min_{x \in K} B_R(x \| y)$$

$$B_R(x \| y) := R(x) - R(y) - \nabla R(y)^\top (x - y)$$

$$x_t = \prod_K^R(y_t)$$

$$y_{t+1} = (\nabla R)^{-1}(\nabla R(y_t) - \eta \nabla f_t(x_t))$$

Adaptive Regularization: AdaGrad

- Consider generalized linear model, prediction is function of $a^T x$

$$\nabla f_t(x) = \ell(a_t, b_t, x) a_t$$

- OGD update: $x_{t+1} = x_t - \eta \nabla_t = x_t - \eta \ell(a_t, b_t, x) a_t$
- features treated equally in updating parameter vector

- In typical text classification tasks, feature vectors a_t are very sparse, Slow learning!

- Adaptive regularization: per-feature learning rates

Optimal regularization

- The general RFTL form

$$x_t = \arg\min_{x \in K} \sum_{i=1 \ldots t-1} f_i(x) + \frac{1}{\eta} R(x)$$

- Which regularizer to pick?
- AdaGrad: treat this as a learning problem!
  Family of regularizations:

$$R(x) = |x|_A^2 \quad s.t. \quad A \succcurlyeq 0 \ , Trace(A) = d$$

- Objective in matrix world: best regret in hindsight!

AdaGrad (diagonal form)

- 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:

$$G_t = \mathrm{diag}\left(\sum_{i=1}^{t} \nabla f_i(x_i) \nabla f_i(x_i)^\top\right)$$

$$y_{t+1} = x_t - \eta G_t^{-1/2} \nabla f_t(x_t)$$

$$x_{t+1} = \arg\min_{x \in K} (y_{t+1} - x)^\top G_t (y_{t+1} - x)$$

- Regret bound: [Duchi, Hazan, Singer '10]

  $O\left(\sum_i \sqrt{\sum_t \nabla_{t,i}^2}\right)$, can be $\sqrt{d}$ better than SGD

- Infrequently occurring, or small-scale, features have small influence on regret (and therefore, convergence to optimal parameter)

Agenda

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

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

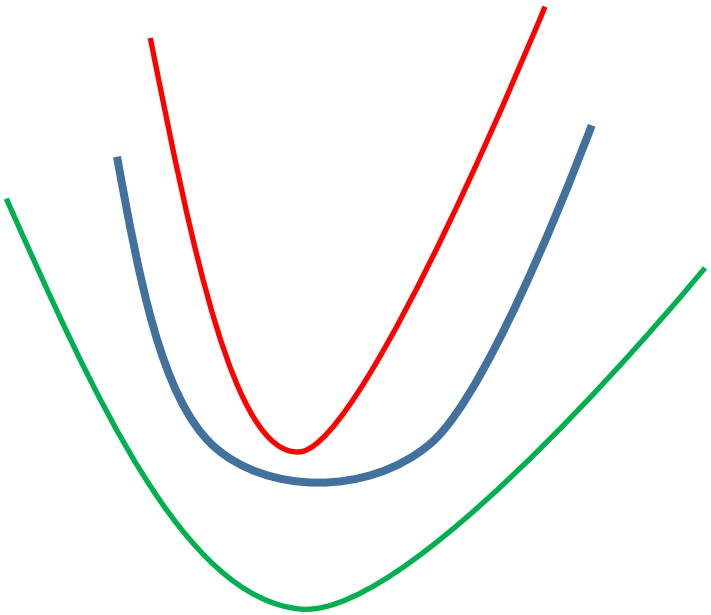
Accelerating gradient descent?

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

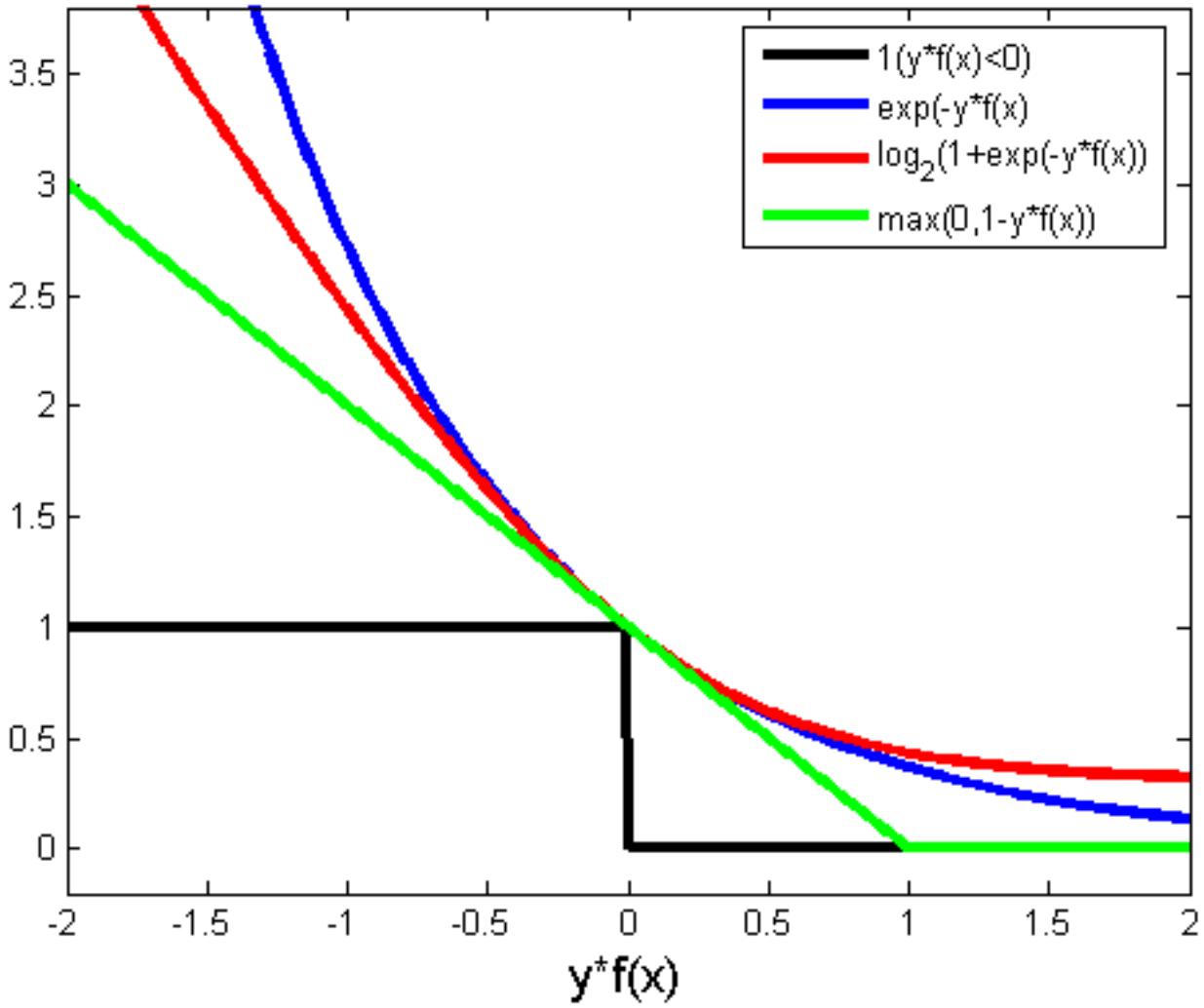
Non-convex smooth functions: (simplified)

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

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: $x_{t+1} = x_t - \eta \nabla_t$)

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

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

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 $x_{t+1} = x_t - \eta \nabla_t$ and $T = \Omega\left(\frac{1}{\epsilon}\right)$, 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: $x_{t+1} = x_t - \eta\widetilde{\nabla}_t$)

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

$$= E\left[-\tilde{\nabla}_t \cdot \eta\nabla_t + \beta\left|\widetilde{\nabla}_t\right|^2\right] = -\eta\nabla_t^2 + \eta^2\beta E\left|\widetilde{\nabla}_t\right|^2$$

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

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

$$T = O\left(\frac{M\beta}{\varepsilon^2}\right) \qquad \Rightarrow \qquad \exists_{t\le T}\,.\ |\nabla_t|^2 \le \varepsilon$$

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)

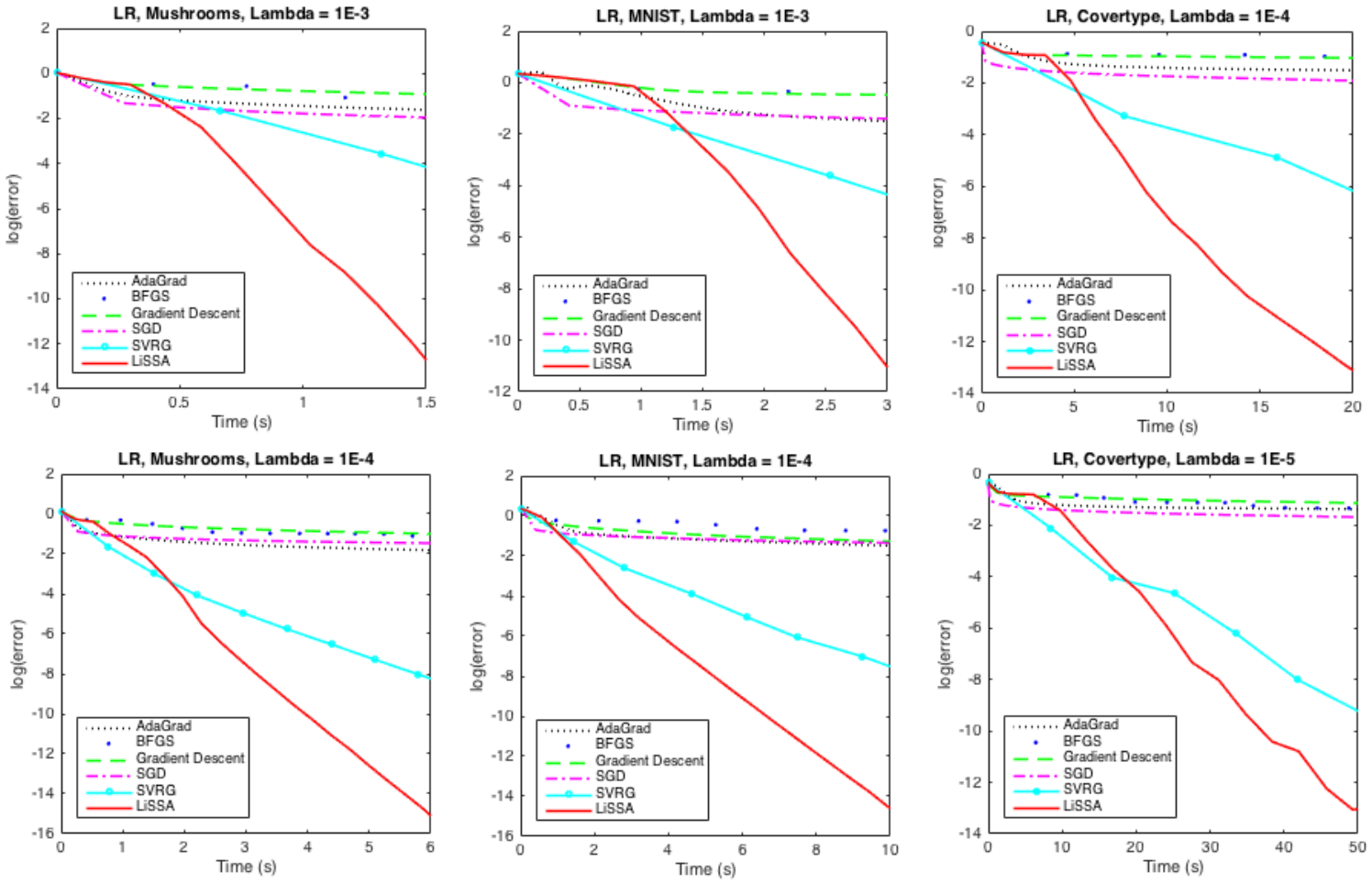
- modest practical improvements, non-convex "momentum" methods.

- 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



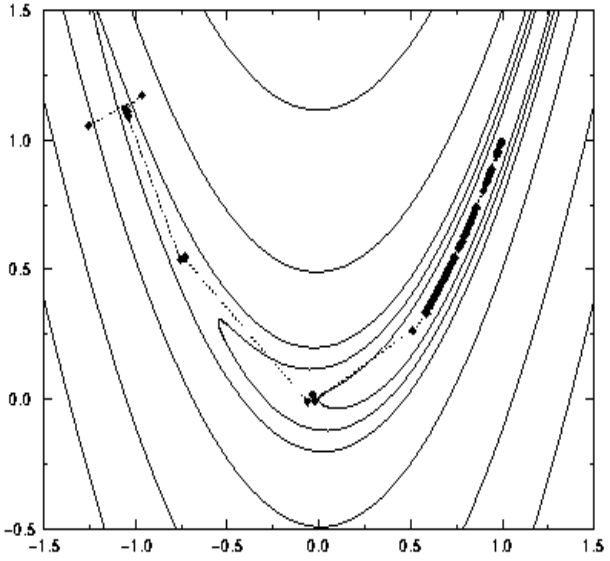
Improve upon GradientDescent++ ?

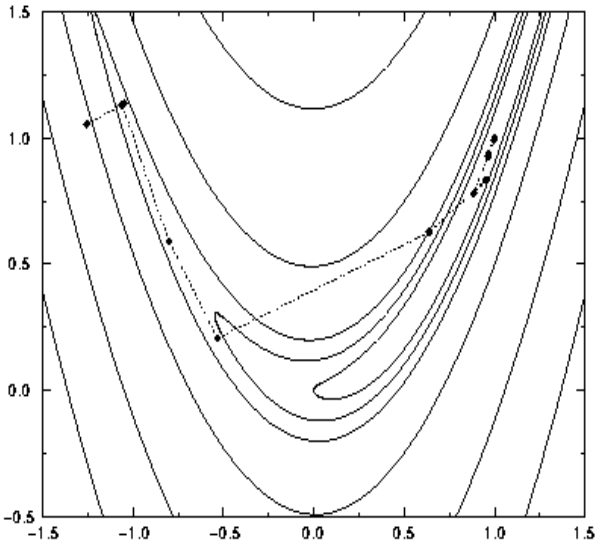
Next few slides:

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

Higher Order Optimization

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

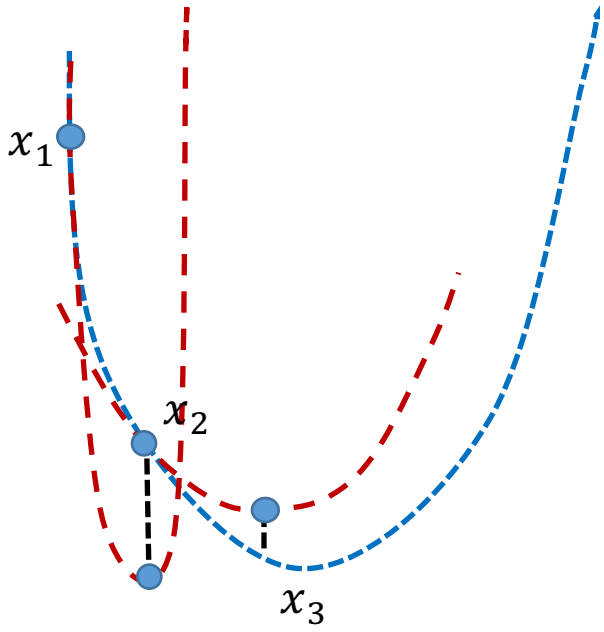




Newton's method (+ Trust region)

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

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

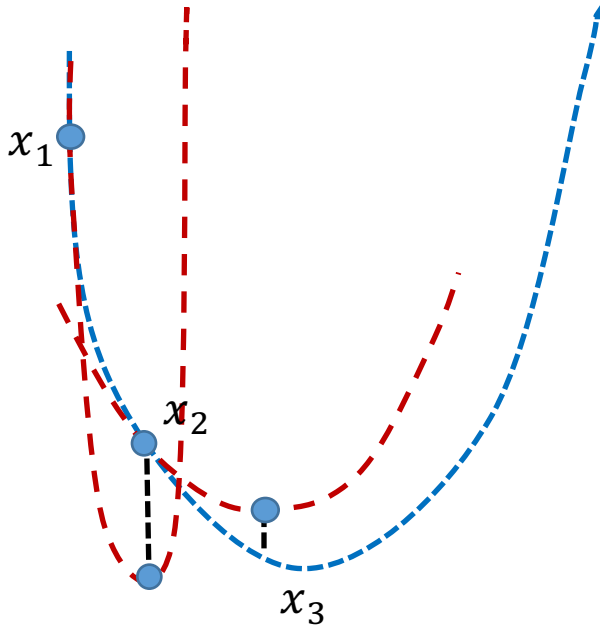


Newton's method (+ Trust region)

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

d^3 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?

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

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

- unbiased estimator of the Hessian:

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

- 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

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

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

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

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]

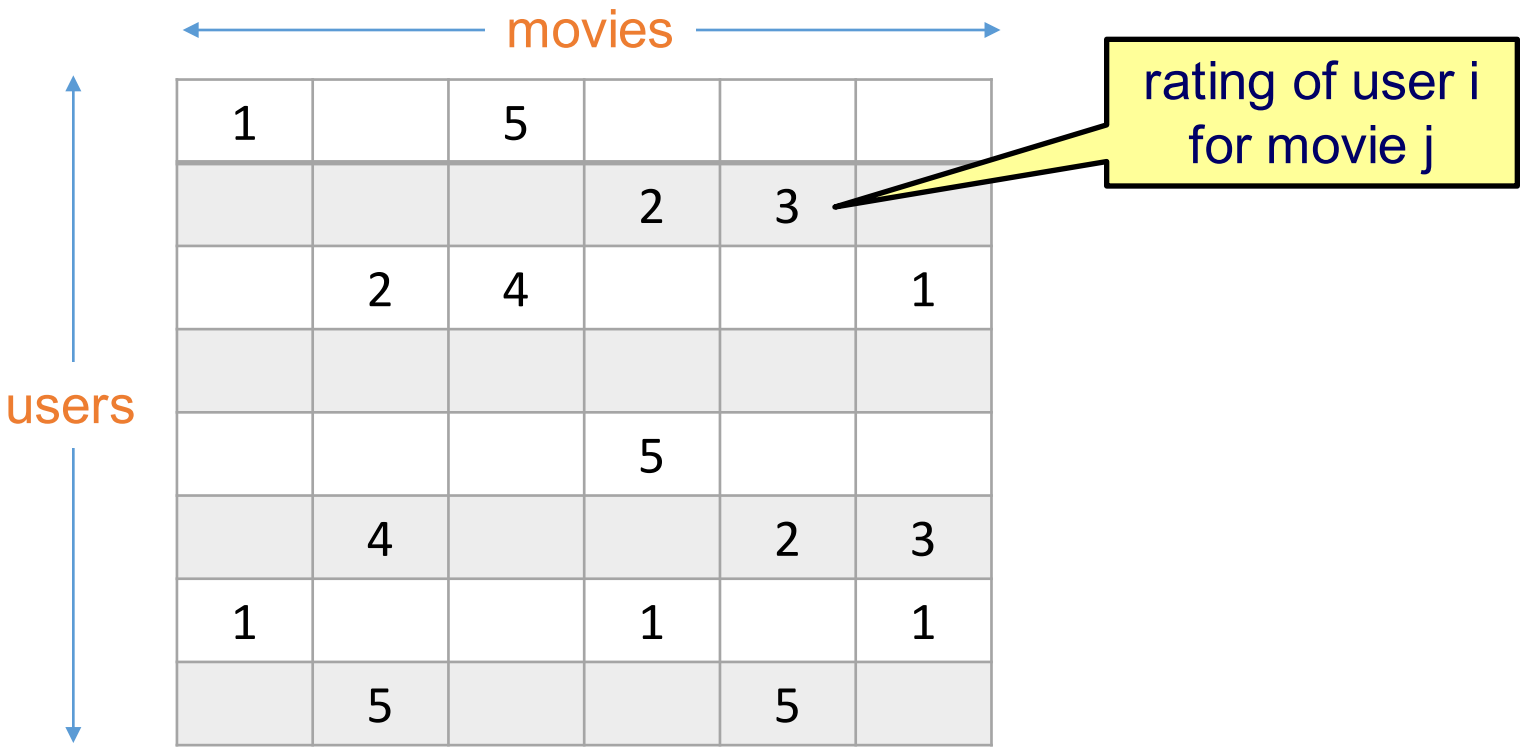
$$O\left(dm \log \frac{1}{\epsilon} + \sqrt{\gamma d}\ d \log \frac{1}{\epsilon}\right)$$

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

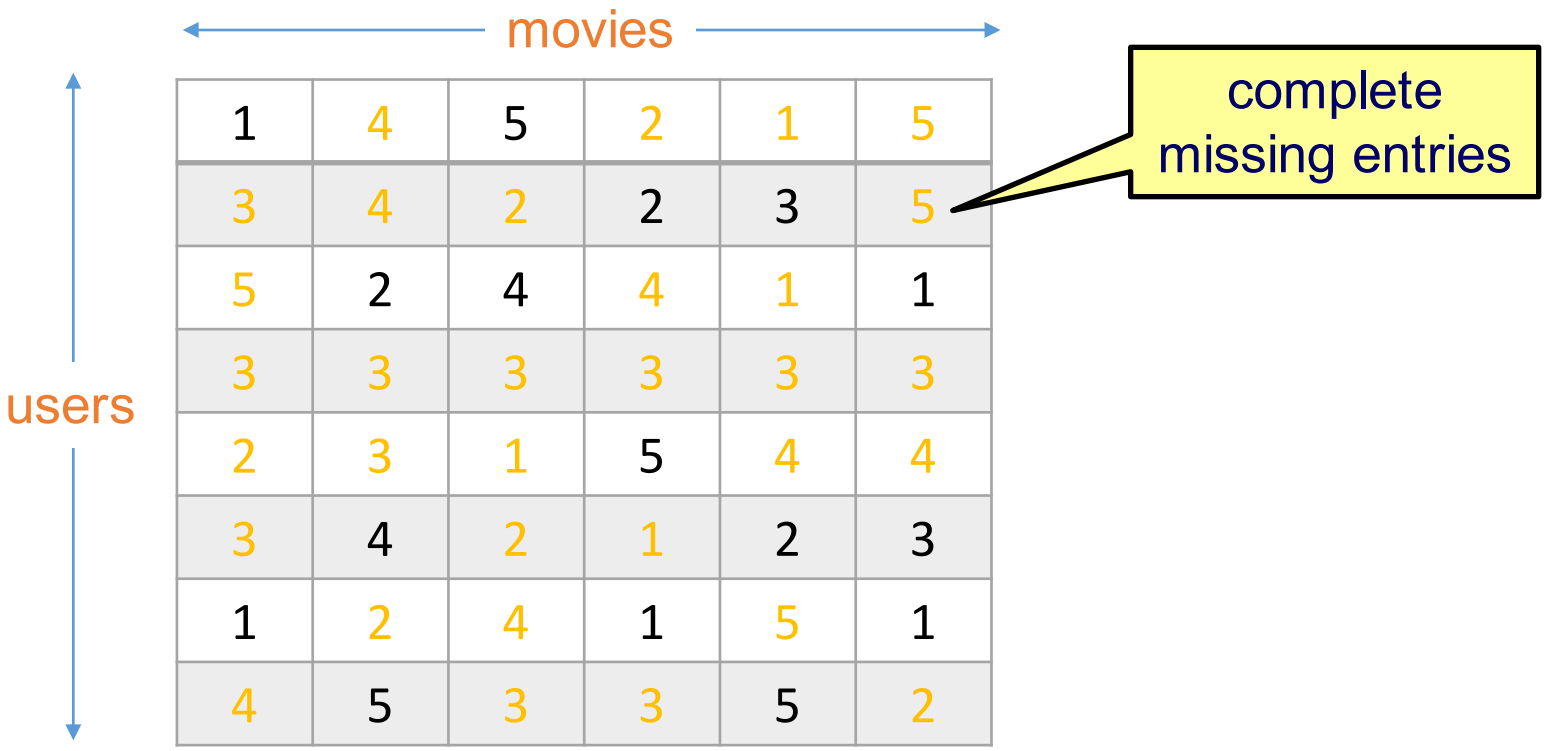
What about constraints??

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

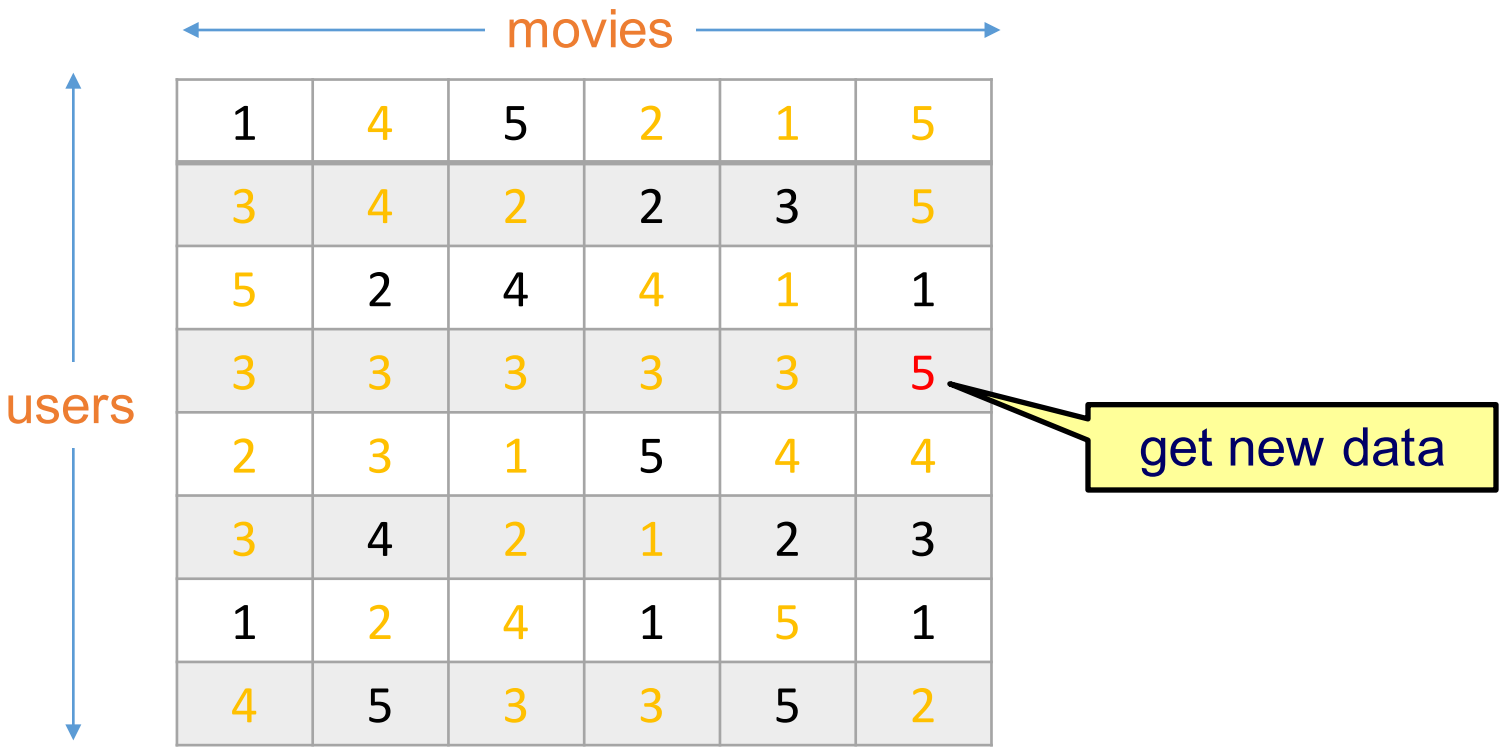
Recommendation systems



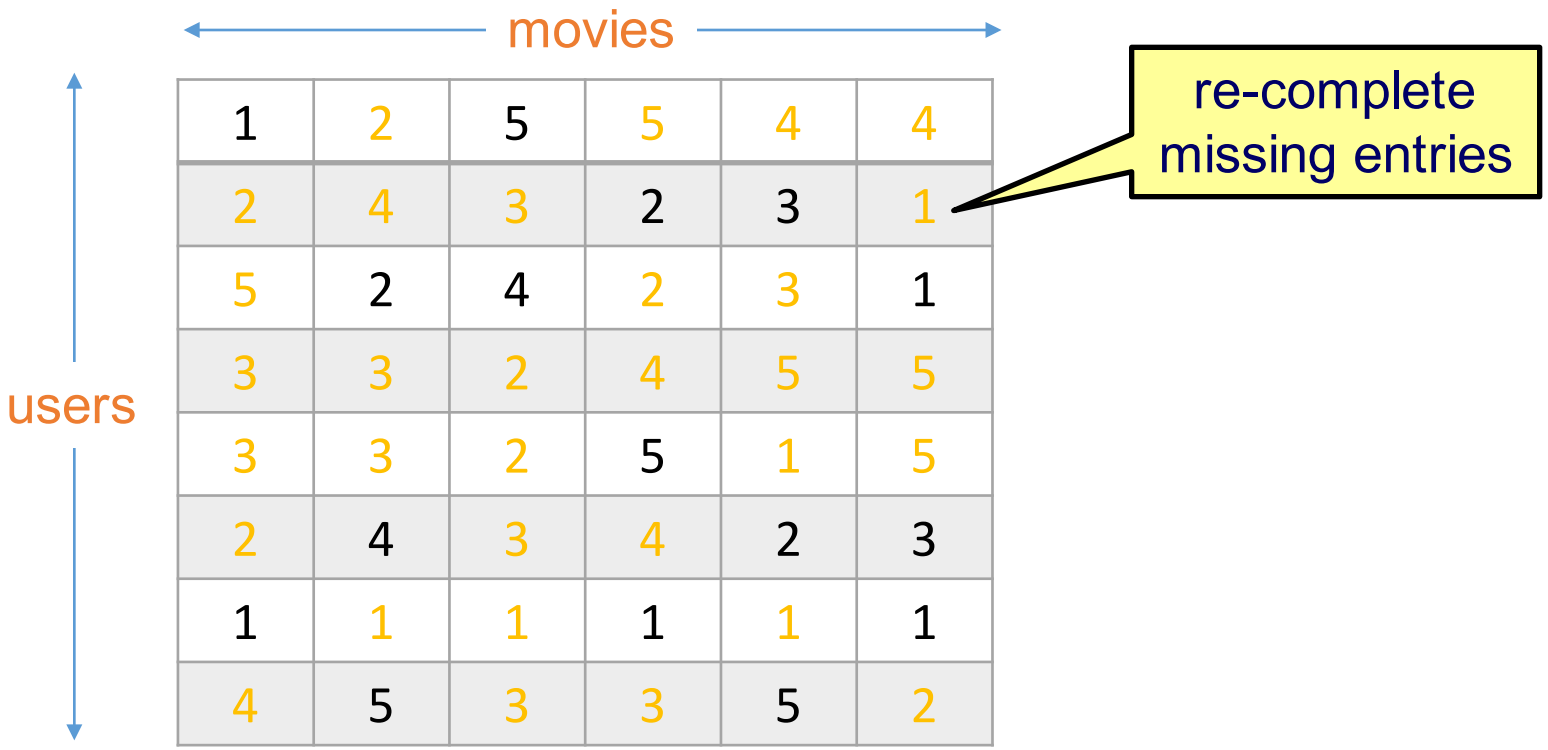
Recommendation systems



Recommendation systems



Recommendation systems



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

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

Bounded trace norm matrices

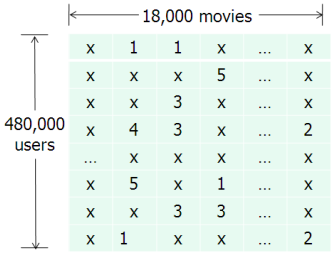
- Trace norm of a matrix = sum of singular values
- K = { X | X is a matrix with trace norm at most D }

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

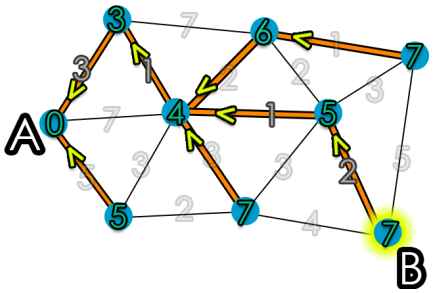
- But: linear optimization over K is easier computing top eigenvector; $O(\text{sparsity})$ time

Projections → linear optimization

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

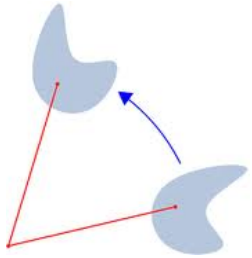


2. Online routing (*K* = flow polytope)
   ~~conic optimization over flow polytope~~



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

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

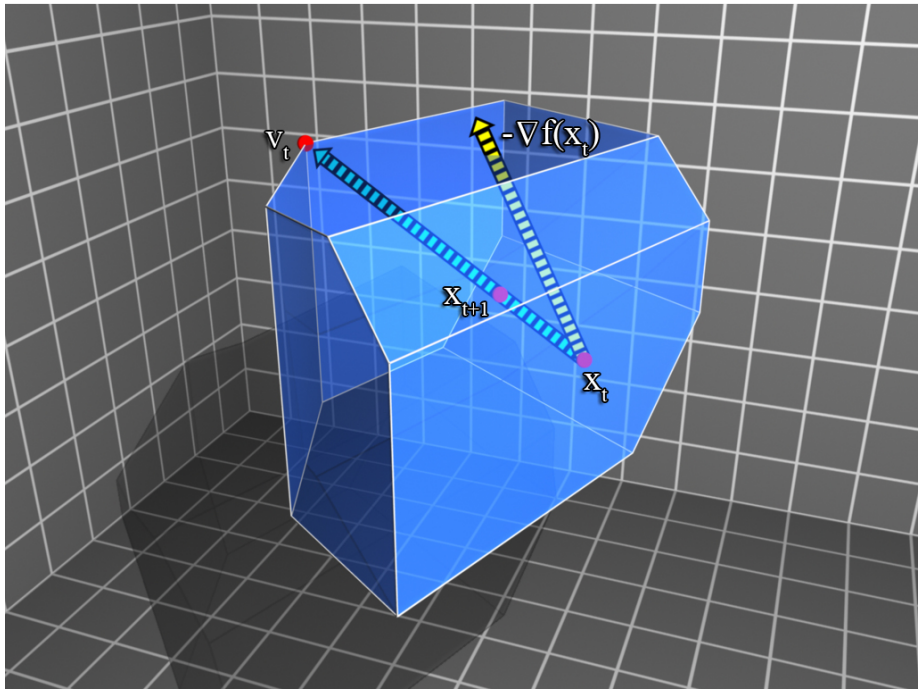


Conditional Gradient algorithm [Frank, Wolfe '56]

Convex opt problem:

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

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



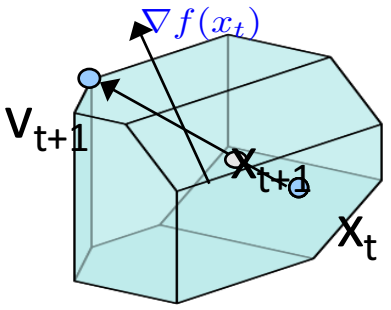
$$v_t = \arg\min_{x \in K} \nabla f(x_t)^\top x$$

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

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

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

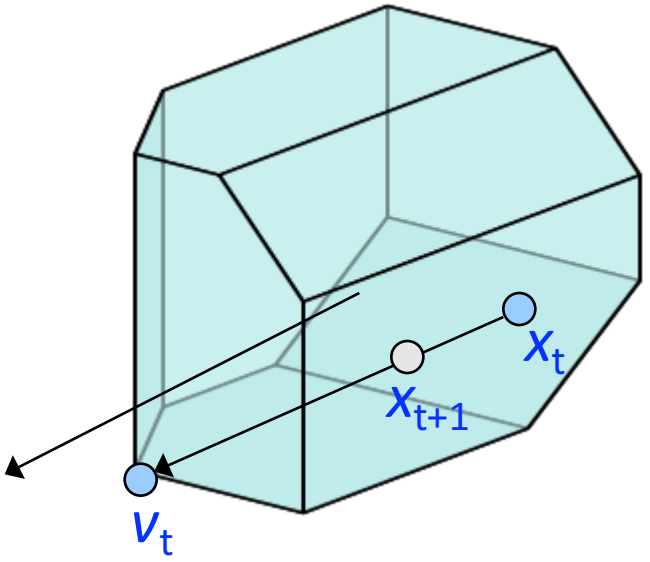
Online Conditional Gradient

- 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,

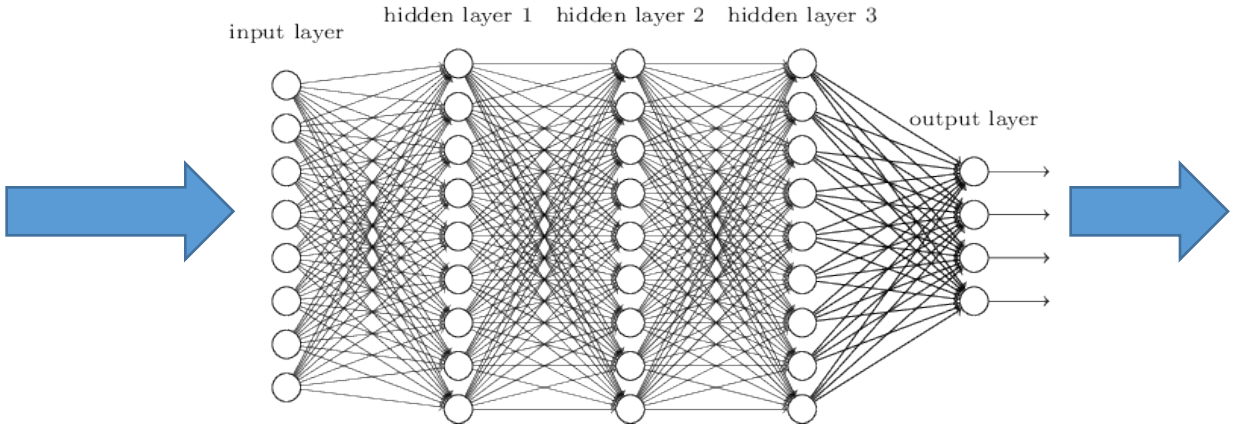
1. Offline: convergence after t steps: $e^{-\Omega(t)}$
2. Online: $Regret = O(\sqrt{T})$

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

Non-convex optimization in ML

Machine

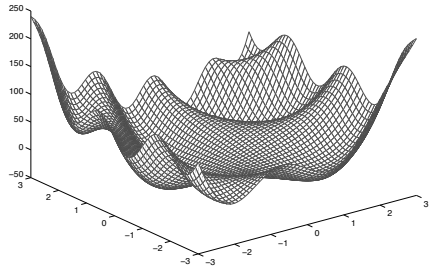
Distribution over $\{a\} \in R^n$



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 4^{th} 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,…)

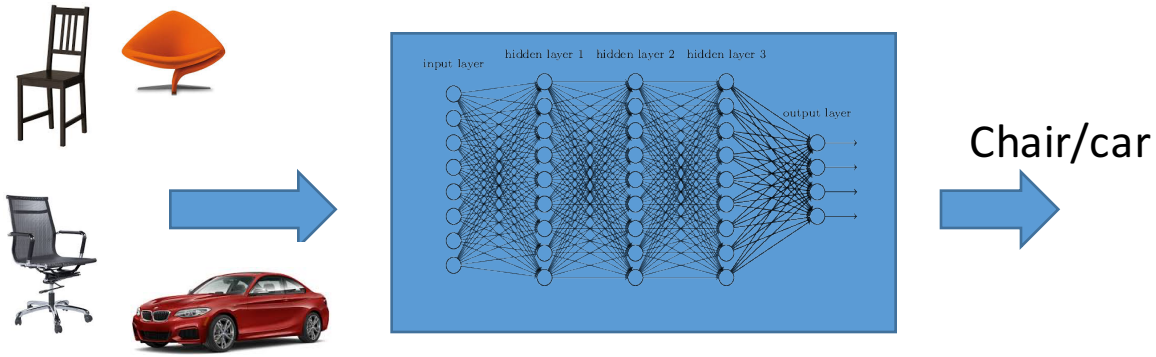
Gradient/Hessian based methods

Goal: find point x such that

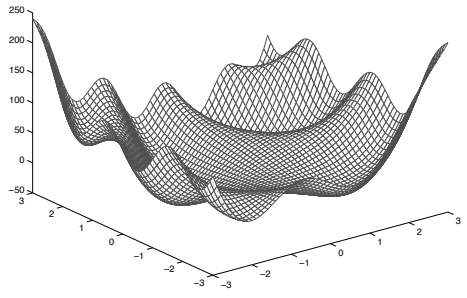
1. $|\nabla f(x)| \leq \varepsilon$ (approximate first order optimality)
2. $\nabla^2 f(x) \succcurlyeq -\varepsilon I$ (approximate second order optimality)

1. (we've proved) GD algorithm: $x_{t+1} = x_t - \eta \nabla_t$ finds in $O\left(\frac{1}{\epsilon}\right)$ (expensive) iterations point (1)
2. (we've proved) SGD algorithm: $x_{t+1} = x_t - \eta \tilde{\nabla}_t$ 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



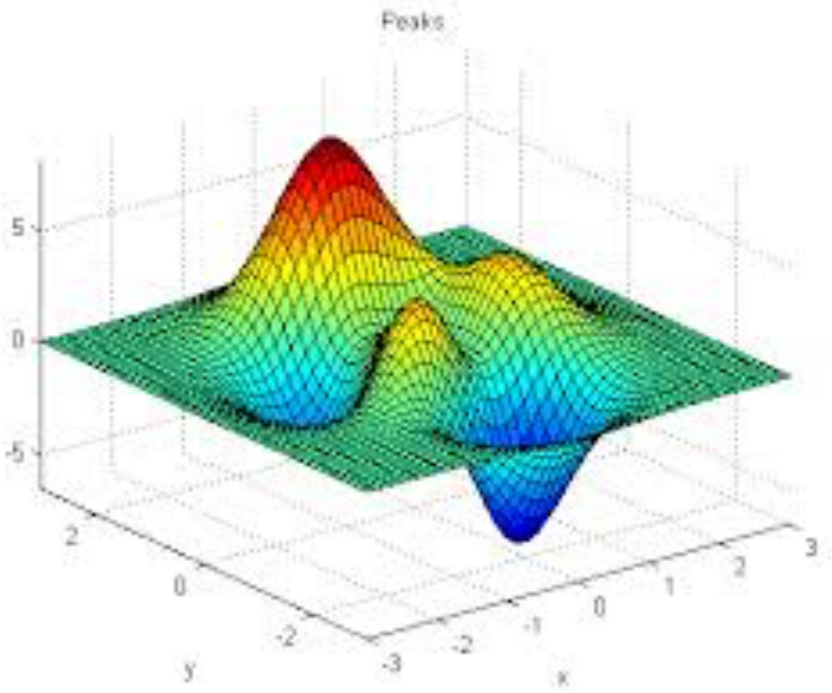
Chair/car



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!