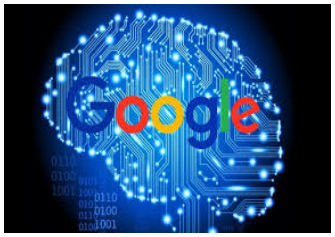
Tutorial:
Optimization for Machine Learning



Elad Hazan
Princeton University and Google Brain
http://www.cs.princeton.edu/~ehazan/tutorial/MLSStutorial.htm

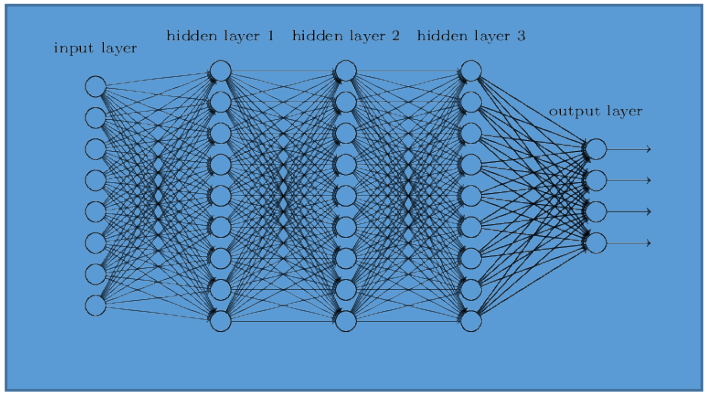
“Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?”

A.M Turing, 1950

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
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm
4. A touch of state-of-the-art

Bibliography + more info: http://www.cs.princeton.edu/~ehazan/tutorial/MLSStutorial.htm

Agenda

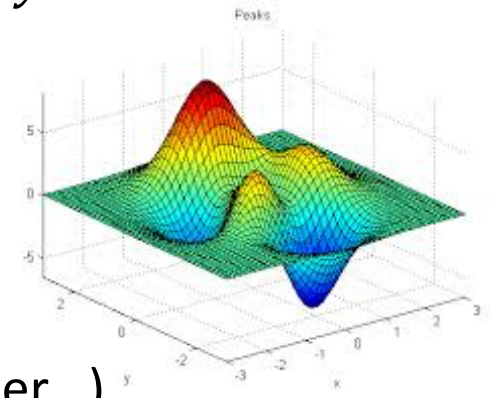
NOT touch upon:

- Parallelism/distributed computation (asynchronous optimization, HOGWILD etc.), Bayesian inference in graphical models, Markov Chain Monte Carlo, Partial information and bandit algorithms, optimization for RL (policy and value iteration, policy gradient, …), Hyperpararmeter optimization, …

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?

- Value oracle (0-order opt.)
- Gradient (1st order opt.), k'th differentials (k'th order…)

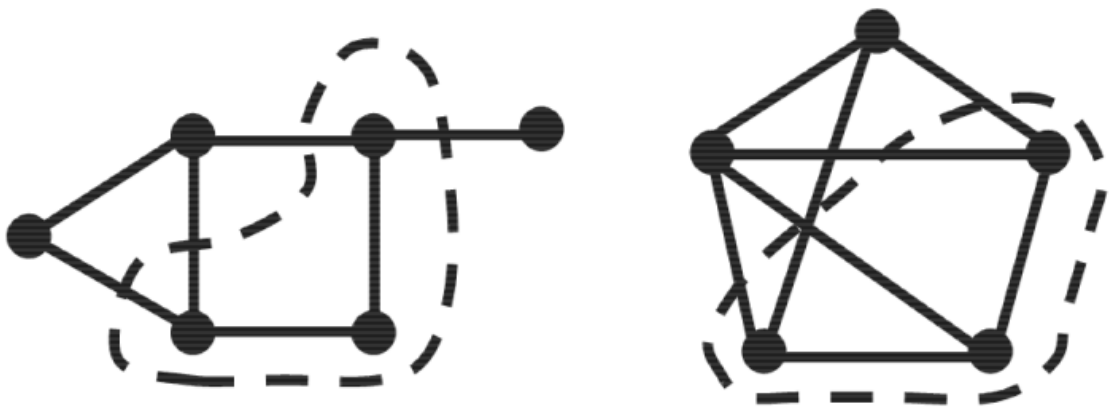
Accessing K? (separation/membership oracle, explicit constraints)

Generally NP-hard, given full access to function.

NP-hardness of Mathematical optimization

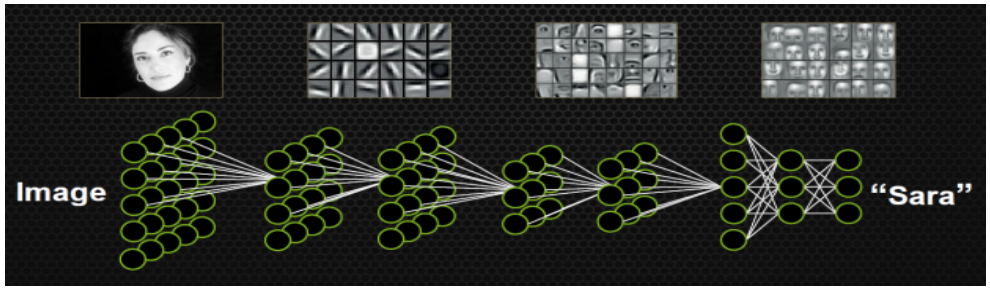
Max Cut over G=(V,E):

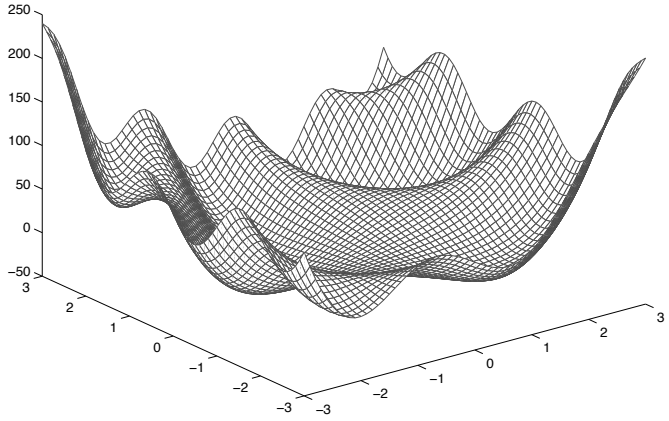
$$\max_{x\in[-1,1]^V} \sum_{(ij)\in E} \frac{1-x_i x_j}{2}$$



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

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



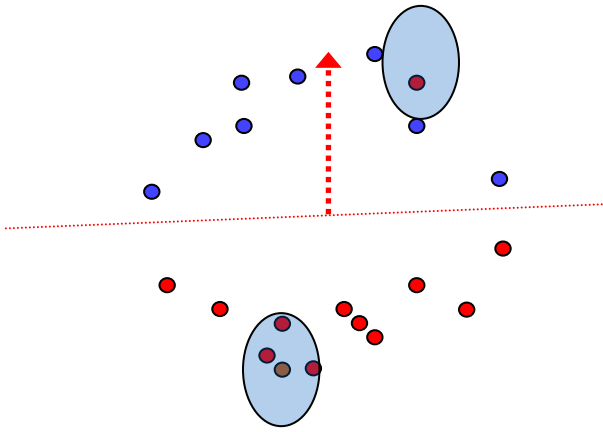


$$\arg\min_{x \in R^k} \frac{1}{m} \sum_{i=1\ to\ m} \ell(x, a_i, b_i) + R(x)$$

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!

Example: deep neural net classification

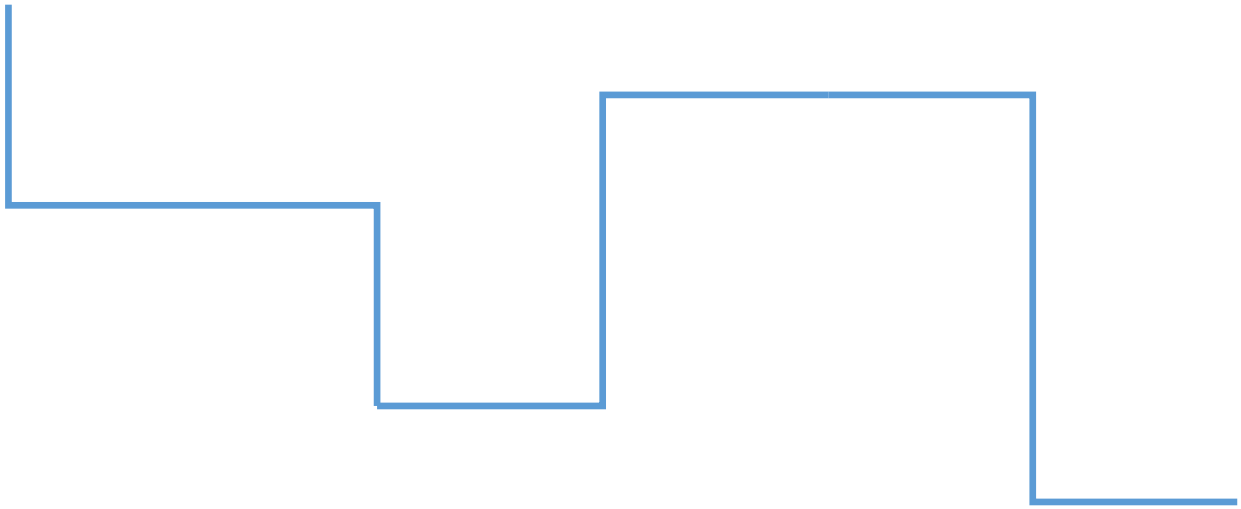
Given a sample $S = \{(a_1, b_1), \ldots, (a_m, b_m)\}$,
find a depth N ResNet such that:

$W = \arg\min_{|W| \le K}$ # of mistakes =

$$\arg\min_{|W| \le K} \left|\left\{i \;\; s.t. sign\left(Net_W(a_i)\right) \neq b_i\right\}\right|$$

$\arg\min_{|W| \le K} \frac{1}{m} \sum_i \ell(W, a_i, b_i)$ for $\ell(W, a_i, b_i) = \begin{cases} 1 & Net_W(a) \neq b \\ 0 & Net_W(a) = b \end{cases}$

Sum of signs → global optimization NP-hard!



Local property that ensures global optimality?

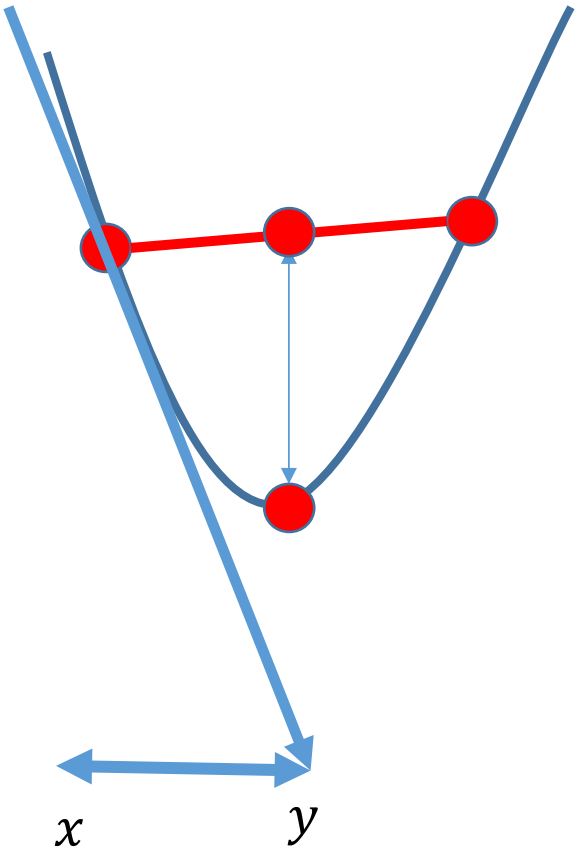
Convexity

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:

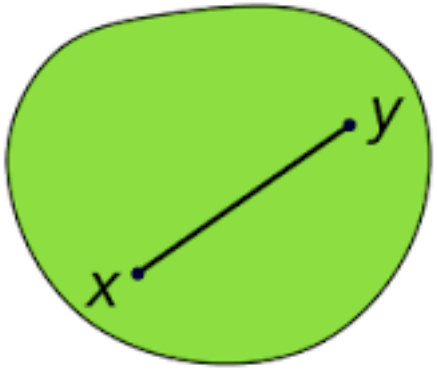
$$\mathrm{f(y)} \geq \mathrm{f(x)} + \nabla f(x)^{\mathsf{T}}(y - x)$$

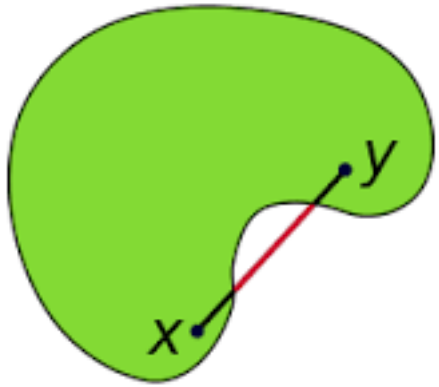


Convex sets

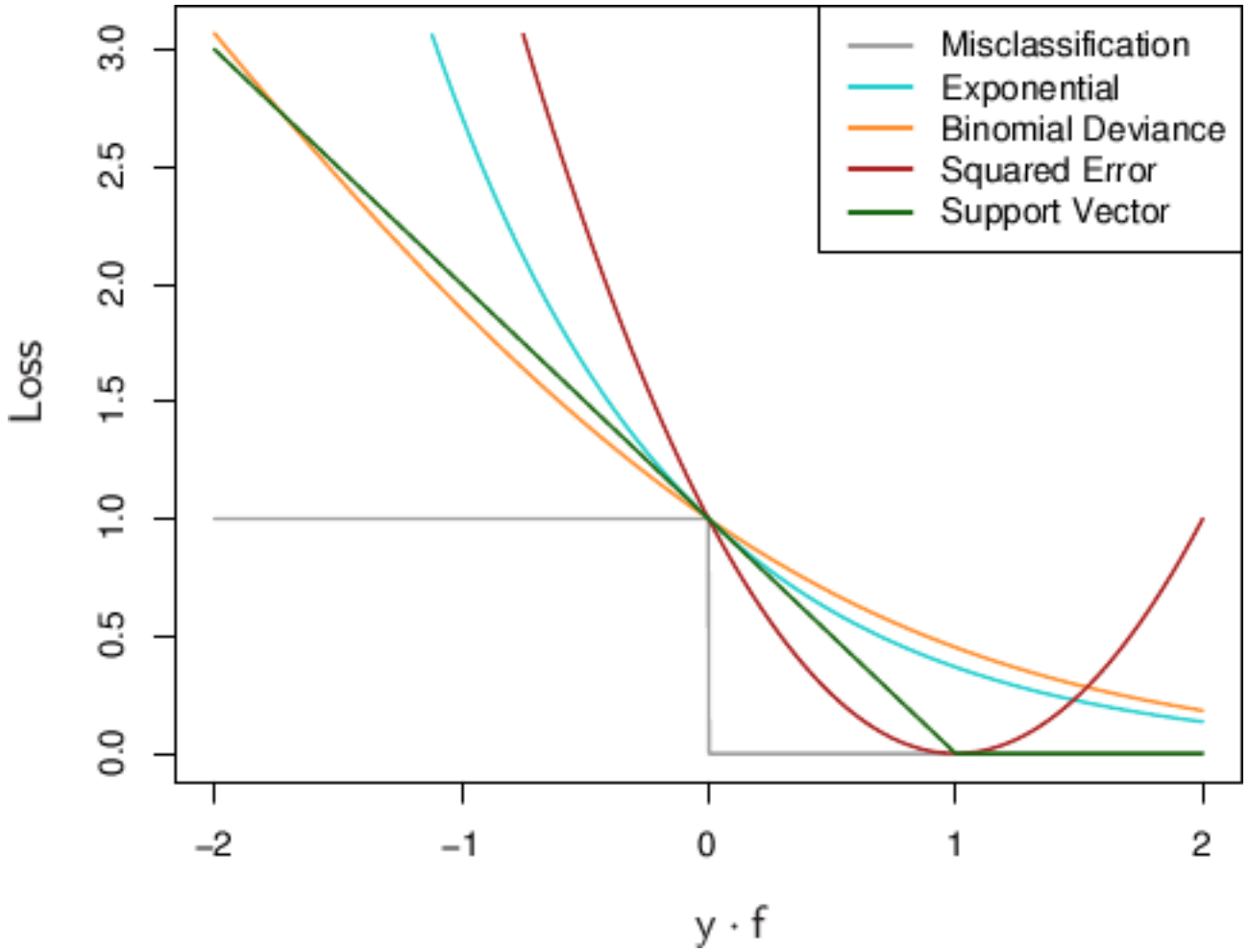
Set K is convex if and only if:

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



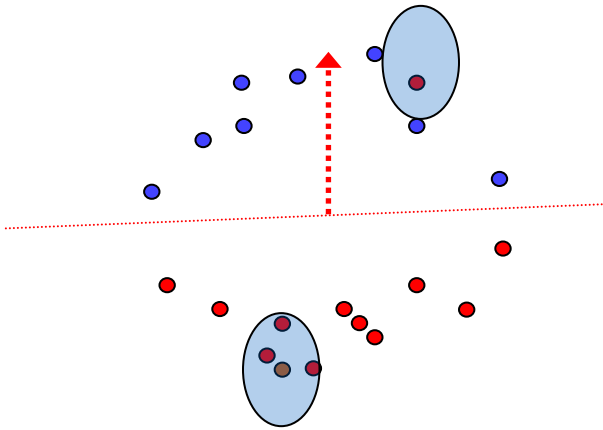


Loss functions $\ell(x, a_i, b_i) = \ell(x^T a_i \cdot b_i)$



Convex relaxations for linear classification

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

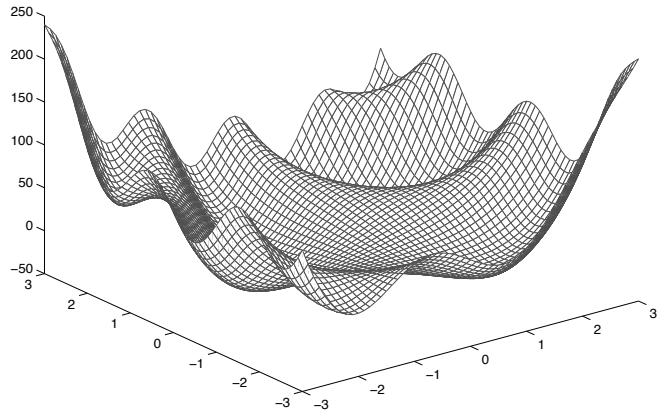


1. Ridge / linear regression $\ell(x^\top a_i, y_i) = (x^\top a_i - b_i)^2$
2. SVM $\ell(x^\top a_i, y_i) = \max\{0, 1 - b_i\ x^\top a_i\}$
3. Logistic regression $\ell(x^\top a_i, y_i) = \log(1 + e^{-b_i \cdot x^\top a_i})$

Non-Convex Optimization

Multiple unsurpassable hurdles.

- Global optimization is NP-hard
- Even deciding whether you are at a local minimum is NP-hard

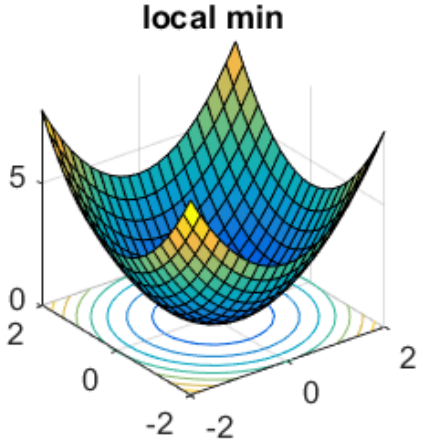


So what can we hope to do?

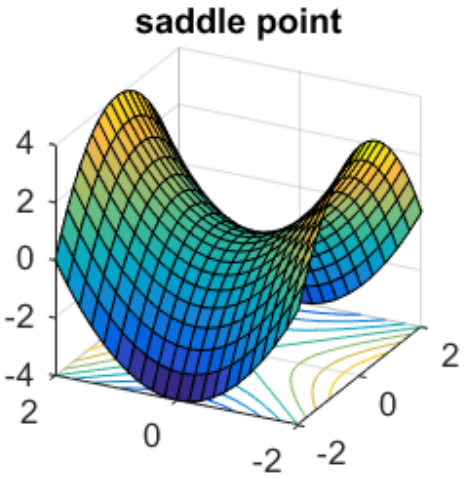
- First Order Critical Points – $\{h \mid ||\nabla f(h)|| \leq \epsilon\}$

Good enough

$\lambda_{min}(\nabla^2 f(h)) > 0$



or



Less ideal

$\lambda_{min}(\nabla^2 f(h)) < 0$

4^{th} order critical points: NP-hard! $f(x) - f(x^*) = O(|x - x^*|^{k+1})$

A better baseline

- $\epsilon\text{-}approximate\ local\ minimum$ (Nesterov and Polyak)

$$||\nabla f(h)|| \leq \epsilon \quad \text{and} \quad \nabla^2 f(h) \succcurlyeq -\sqrt{\epsilon}\, I$$

- Does not guarantee a local minimum even with $\epsilon = 0$ (but for some functions)

We have:

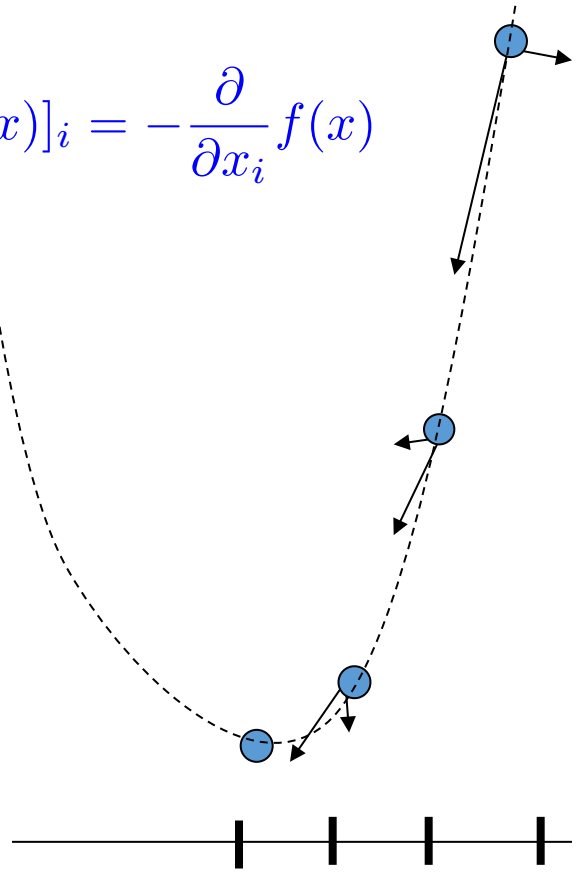
1. motivated learning as mathematical optimization
2. convexity → locally verify global optimality
3. Non-convex opt → k^{th}-order local opt.
   (2^{nd} is reasonable, 4^{th} is NP-hard)

Next → algorithms!

Unconstrained Gradient Descent

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

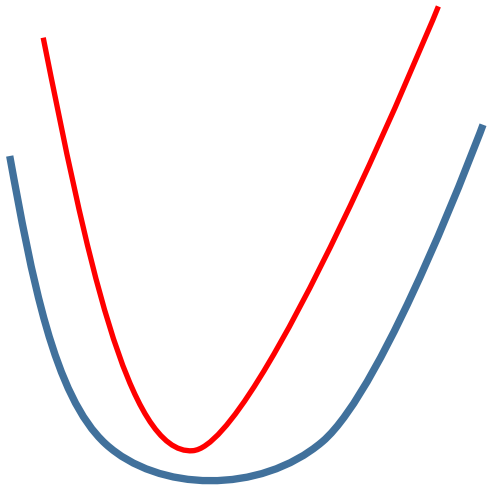
$$x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$



Smooth functions

β = smoothness, simplified:

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



Why is this important?

For gradients to be meaningful at all…

Remark: any function can be “smoothed” by local integration

Convergence of gradient descent

$$x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$$

Theorem: for β-smooth M-bounded functions, step size $\eta = \frac{1}{\beta}$,

$$\frac{1}{T}\sum_t |\nabla f(x_t)|^2 \leq \frac{2M\beta}{T}$$

Thus, there exists a time for which:

$$|\nabla f(x_t)|^2 \leq \frac{2M\beta}{T}$$

convex f → global convergence:

$$f(x_t) - f(x^*) \leq |x_t - x^*||\nabla f(x_t)| = O\left(\frac{D}{\sqrt{T}}\right)$$

Proof: $x_{t+1} \leftarrow x_t - \eta \nabla f(x_t)$

1. The **descent lemma**:

$$f(x_t) - f(x_{t+1}) \geq -\nabla f(x_t)(x_{t+1} - x_t) - \frac{1}{2}\beta |x_t - x_{t+1}|^2$$

$$= \eta |\nabla_t|^2 - \frac{1}{2}\beta \eta^2 |\nabla_t|^2$$

$$= \frac{1}{2\beta} |\nabla_t|^2$$

2. Summing over all iterations:

$$M \geq f(x_1) - f(x^*) \geq \sum_t f(x_t) - f(x_{t+1}) \geq \frac{1}{2\beta} \sum_t |\nabla_t|^2$$

and thus, $\frac{1}{T} \sum_t |\nabla f(x_t)|^2 \leq \frac{2M\beta}{T}$

For convex functions:

$$f(x) - f(x^*) \leq \nabla f(x)(x^* - x) \leq D|\nabla f(x)|$$

Thus, in T iterations, exists an iterate t for which:

$$f(x) - f(x^*) \leq \frac{D\sqrt{2M\beta}}{\sqrt{T}}$$

Our objective function:

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

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

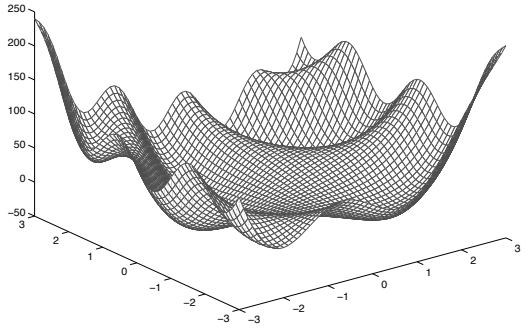
→ Gradients are expensive! (entire data set per iteration)

→ What about non-smooth functions? (non-convex: infeasible)

→ How good is a local minimum in terms of **generalization**?

→ Can we get faster rates?

Stochastic gradient descent



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

"Random approximation" (Robbins/Monro '51)

estimate the gradient using a single example: $f_t(x) = \ell_i(x, a_i, b_i)$

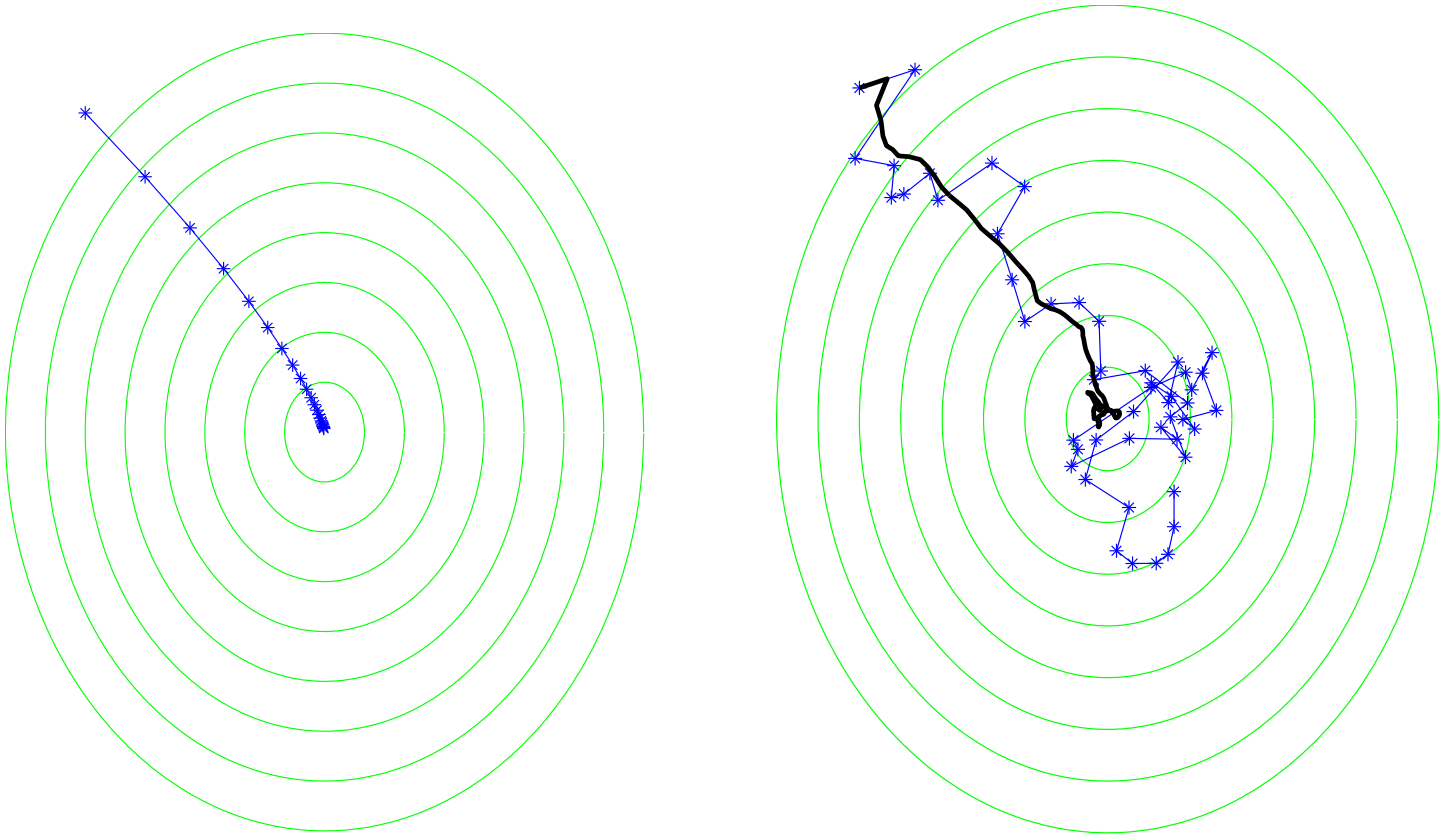
$$\tilde{\nabla} \; = \; \mathrm{E}[\nabla f(x)] = \; E_{(a_i,b_i)}[\nabla \, \ell_i(x, a_i, b_i)]$$

move in the direction of estimator! $x_{t+1} \leftarrow x_t - \eta \widetilde{\nabla(x_t)}$

Convergence?

Variance of gradient estimator: $\mathrm{E}[\|\nabla\|^2] = \sigma^2$

Stochastic vs. full gradient descent



Convergence of SGD

$$x_{t+1} \leftarrow x_t - \eta \tilde{\nabla} f(x_t)$$

Theorem: for β-smooth M-bounded functions, step size $\eta = \sqrt{\frac{M}{T\beta\sigma^2}}$,

$$\frac{1}{T}\sum_t \|\nabla f(x_t)\|^2 \leq \sqrt{\frac{M\beta\sigma^2}{T}}$$

Proof:

$$x_{t+1} \leftarrow x_t - \eta\tilde{\nabla} f(x_t)$$

1. The **descent lemma**:

$$\mathrm{E}[f(x_t) - f(x_{t+1})] \geq \mathrm{E}\left[-\nabla f(x_t)(x_{t+1} - x_t) - \frac{1}{2}\beta|x_t - x_{t+1}|^2\right]$$

$$= E\left[\nabla_t \cdot \eta\tilde{\nabla}_t - \frac{1}{2}\beta\eta^2\left|\tilde{\nabla}\right|^2\right] = \eta|\nabla_t|^2 - \frac{1}{2}\beta\eta^2\sigma^2$$

2. Summing over all iterations:

$$\mathrm{M} \geq \mathrm{f}(\mathrm{x}_1) - \mathrm{f}(\mathrm{x}^*) \geq \sum_t f(x_t) - f(x_{t+1}) \geq \eta\sum_t|\nabla_t|^2 - \frac{1}{2}\beta\eta^2\sigma^2 T$$

and thus, $\frac{1}{T}\sum_t|\nabla f(x_t)|^2 \leq \frac{M}{\eta T} + \eta\beta\sigma^2 \leq \sqrt{\frac{M\beta\sigma^2}{T}}$

Our objective function:

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

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

→ Gradients are expensive! (entire data set per iteration)

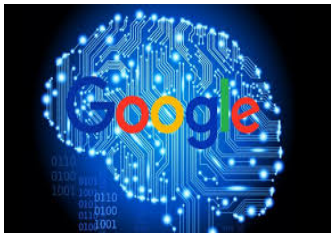
→ What about non-smooth functions? (non-convex: infeasible)

→ How good is a local minimum in terms of **generalization**?

→ Can we get faster rates?

Tutorial: PART 2

Optimization for Machine Learning



Elad Hazan
Princeton University and Google Brain

Bibliography + more info:
http://www.cs.princeton.edu/~ehazan/tutorial/MLSStutorial.htm

Agenda

1. Learning as mathematical optimization
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm
4. A touch of state-of-the-art

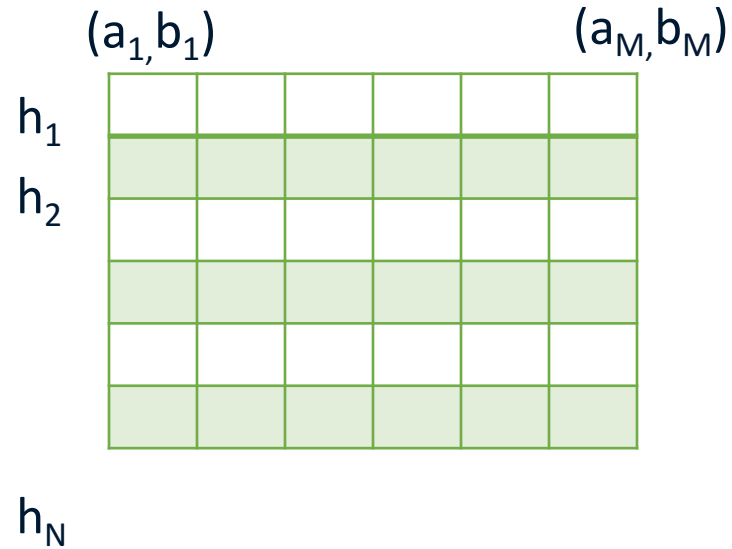
- → What about non-smooth functions? (non-convex: infeasible)
- → Gradients are expensive! (entire data set per iteration)
- → How good is a local minimum in terms of **generalization**?
- → Can we get faster rates?

→ convex optimization

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:

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

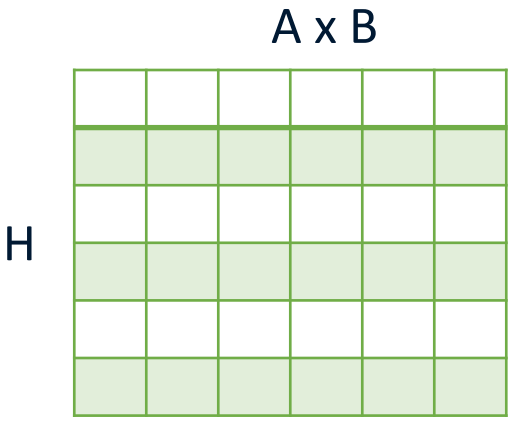
More powerful setting: Online Learning in Games

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)

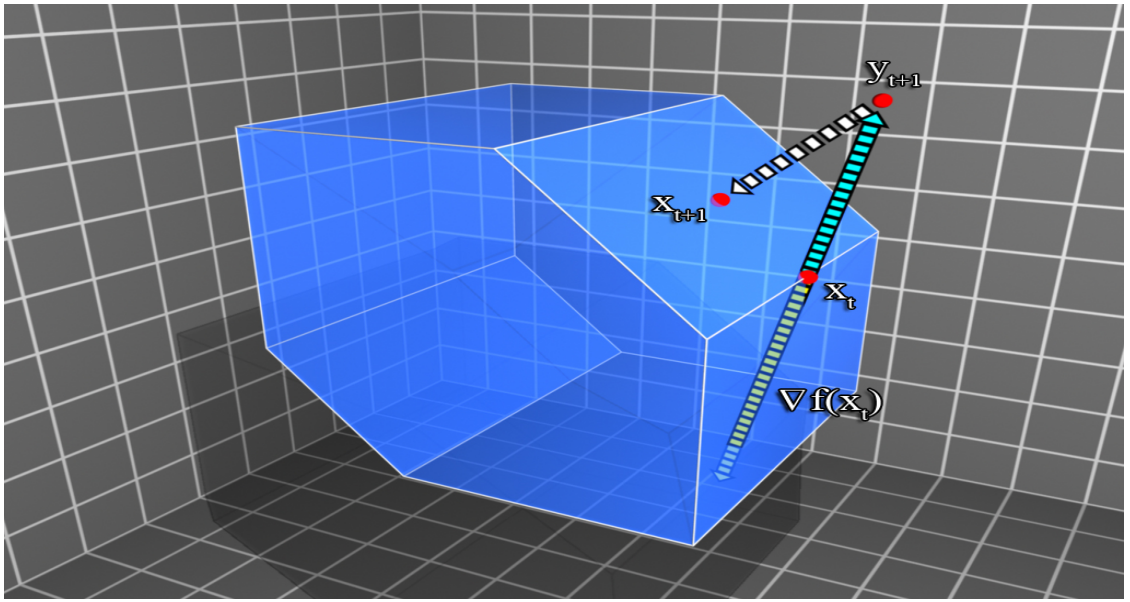
From this point onwards: $f_t(x) = \ell(x, a_t, b_t)$ = loss for one example

Can we minimize regret efficiently?

Online gradient descent

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

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



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

Online gradient descent

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

Theorem: for step size $\eta = \frac{D}{G\sqrt{T}}$, NO SMOOTHNESS REQUIRED!!

$$\sum_t f_t(x_t) - \min_{x^*} \sum_t f_t(x^*) \leq DG\sqrt{T}$$

Where:

- G = upper bound on norm of gradients

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

- D = diameter of constraint set

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

Proof:

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

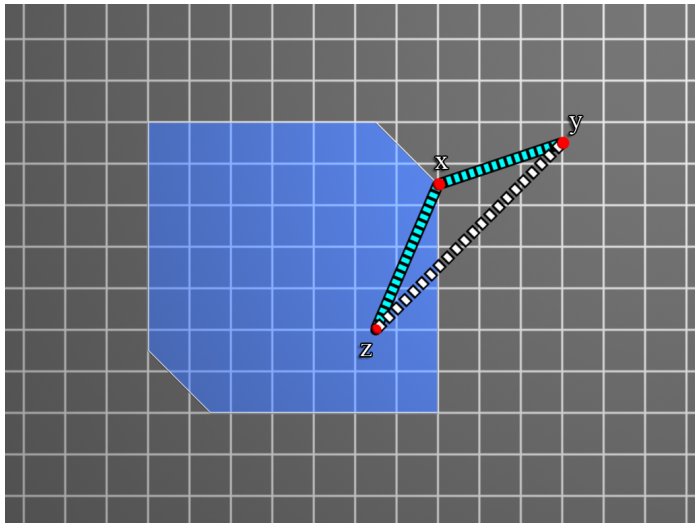
1. Observation 1:

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

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:

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

2. Observation 2:

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

Thus:

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

And hence:

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

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

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

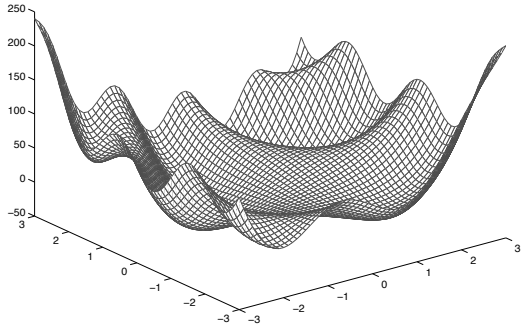
Lower bound

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

- 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)

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

SGD: non-smooth case



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)

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

2. Taking (conditional) expectation:

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

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.

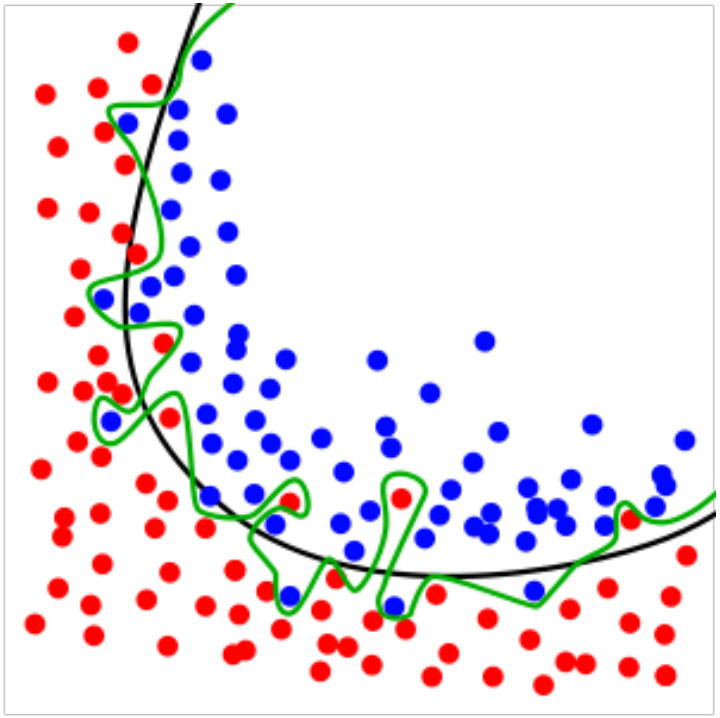
Agenda

1. Learning as mathematical optimization
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm
4. A touch of state of the art

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

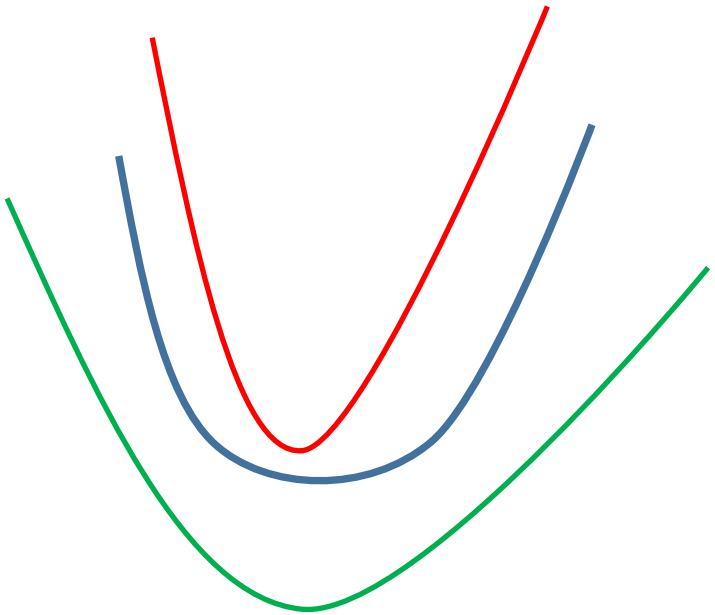


- PAC theory: Regularization <-> reduce complexity
- Regret minimization: Regularization <-> stability

Condition number of convex functions

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

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



α = strong convexity, β = smoothness

Non-convex smooth functions: (simplified)

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

Why do we care?

well-conditioned functions exhibit much faster optimization!
Important for regularization

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!

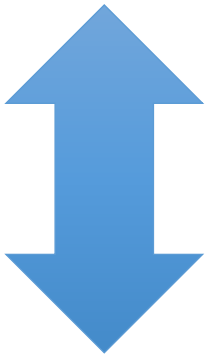
FTRL ⇔ Online Mirror Descent

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

Bregman Projection:

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

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



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

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

GD++

#1: Adaptive Regularization

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

- Set $x_1 \in K$ arbitrarily
- For $t = 1, 2, \ldots$,
  1. use x_t obtain f_t, and gradient $g_t = \nabla f(x_t)$
  2. compute x_{t+1} as follows:

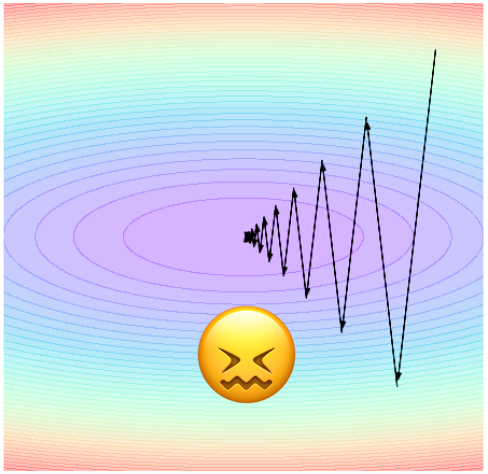
$$G_t = \sum_{i=1 \ldots t} g_i g_i^\top$$

$x_{t+1} = x_t - \eta G_t^{-\frac{1}{2}} g_t$ or diagonal version $x_{t+1} = x_t - \eta \, diag(G_t)^{-\frac{1}{2}} g_t$

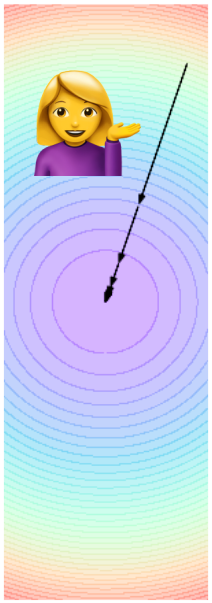
$$y_{t+1} = \arg\min_{x \in K} (y_{t+1} - x)^\top G_t (y_{t+1} - x)$$

Intuition: Adaptive Preconditioning

- Per-coordinate scaling matrix $D_t = [\sum_{s=1}^{t} \mathrm{diag}(g_s[i])^2]^{-1/2}$
- Adaptive methods learn D which makes the loss surface more isotropic



$f(x) \mapsto f(Dx)$



The ratio of adaptivity

- Theorem 1: (informal) AdaGrad has regret within factor at most O(d) from the best regularization in hindsight from the family

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

- Define ratio of adaptivity

$$\mu \ := \ \frac{\sum_{t=1}^{T} \langle g_t, x_t - x^* \rangle}{\|x_1 - x^*\| \sqrt{\sum_{t=1}^{T} \|g_t\|^2}} \ = \ \frac{\text{AdaGrad regret}}{\text{worst-case OGD regret}}$$

- Theorem 2: $\mu = O\left(\sum_i \sqrt{\sum_t \nabla_{t,i}^2}\right) \in [\frac{1}{\sqrt{d}}, \sqrt{d}]$
- Infrequently occurring, or small-scale, features have small influence on regret (and therefore, convergence to optimal parameter)

Agenda

1. Learning as mathematical optimization
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm

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

GD++

#2: Variance Reduction

Recall: 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 $\mathrm{T} = \Omega\left(\frac{1}{\epsilon}\right)$, finds

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

Holds even for non-convex functions

Non-convex stochastic gradient descent

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

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

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

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

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

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

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]

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 $\frac{1}{\epsilon}$

Variance Reduction: analysis

For smooth functions (**non-stochastic**):

$$\frac{1}{\beta}|\nabla_t|^2 \leq h_t = f(x_t) - f(x^*)$$

Variance-reduced estimator advantage:

$$\left|\tilde{\nabla}_t\right|^2 = \left|\tilde{\nabla} f(x_t) - \tilde{\nabla} f(x_0) + \nabla f(x_0)\right|^2 \leq$$
$$\leq 2\left|\tilde{\nabla} f(x_t) - \tilde{\nabla} f(x_0)\right|^2 + 2|\nabla f(x_0)|^2$$

Since $(\mathrm{a}+\mathrm{b})^2 \leq 2a^2 + 2b^2$

$$\leq 2G^2|x_t - x_0|^2 + 2\beta h_t$$

G-smoothness for $\tilde{\nabla} f$, smoothness

$\leq 2\frac{G^2}{\alpha}\mathrm{h_t} + 2\beta h_t$ by strong convexity

$$= O(\beta h_T)$$

Variance Reduction: analysis

Regret minimization for strongly convex functions:

$$h^{k+1} = \frac{1}{T}\sum_t h_t \leq \frac{1}{\alpha T}\sum_t \frac{\left|\tilde{\nabla}_t\right|^2}{t} \leq \frac{\beta \log T}{\alpha T} h^k$$

Thus, after $O\left(\gamma = \frac{\beta}{\alpha}\right)$ iterations, we have that the average distance ½ of what it (on average) was:

$$E[h^{k+1}] \leq \frac{1}{2} E[h^k]$$

Thus $O\left(\log \frac{1}{\epsilon}\right)$ epochs. Each epoch: one full gradient, T stochastic gradients.
Total running time:

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

GD++

#3: Momentum and Nesterov acceleration

Acceleration/momentum [Nesterov '83]

- Optimal gradient complexity (smooth, convex)

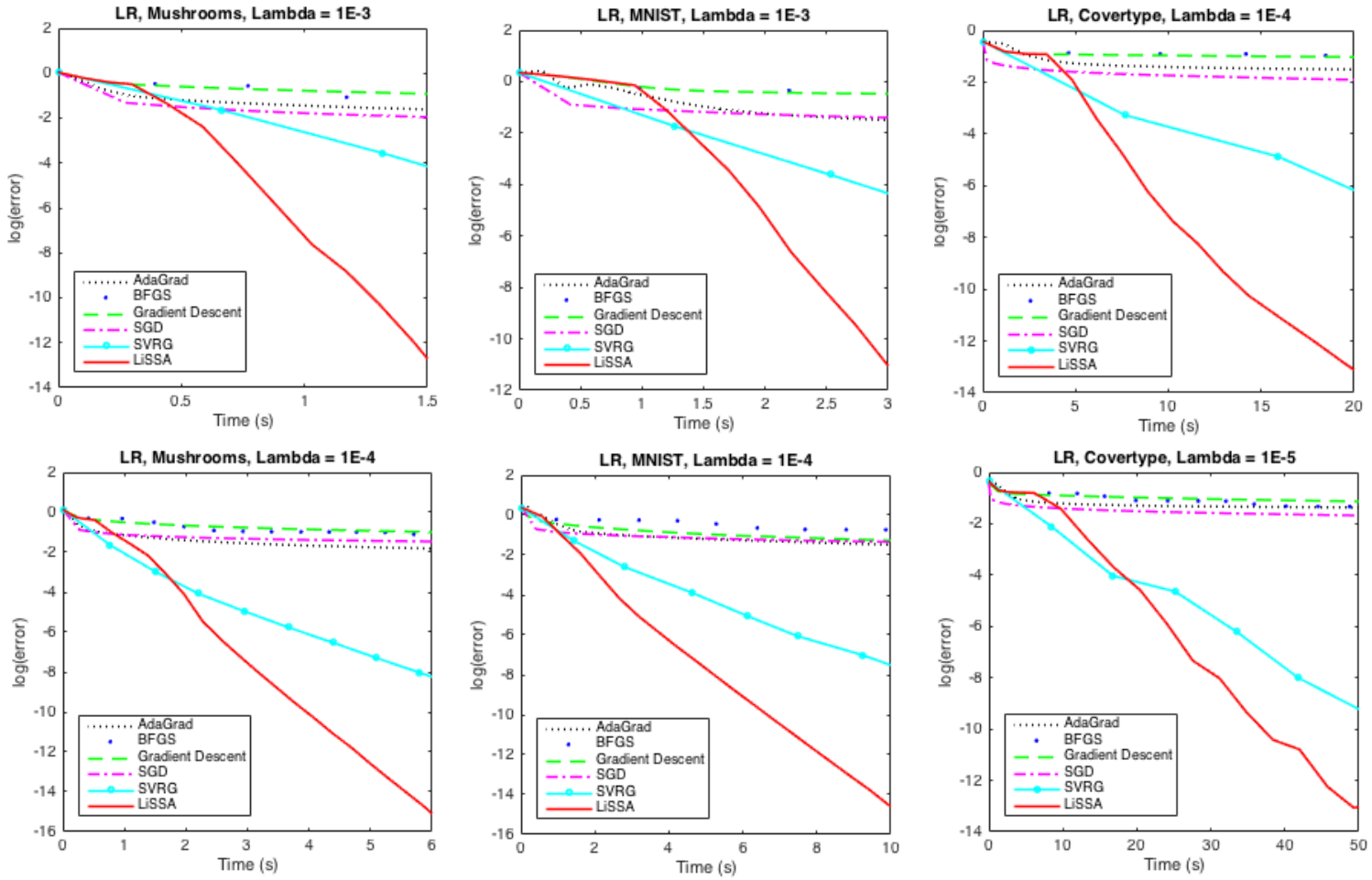
- modest practical improvements, non-convex "momentum" methods.

- 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



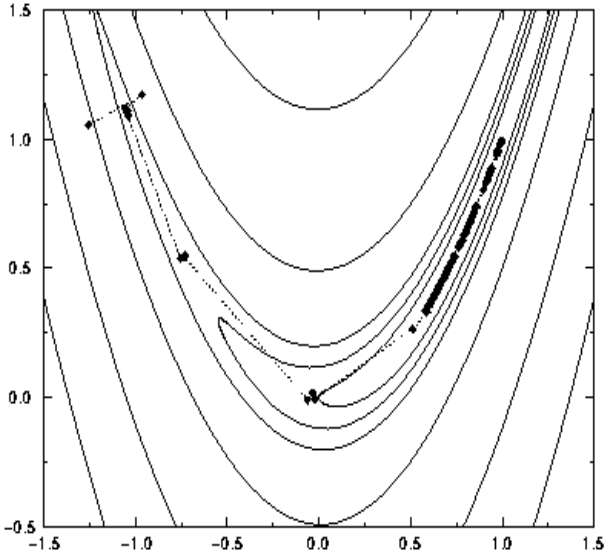
Improve upon GradientDescent++ ?

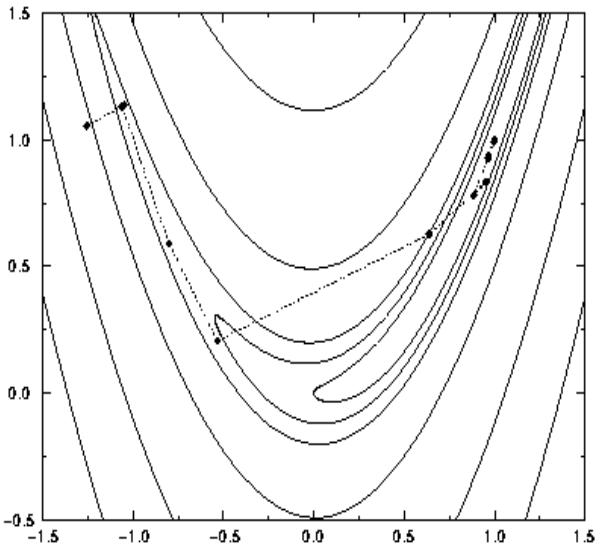
Next few slides:

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

Higher Order Optimization

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

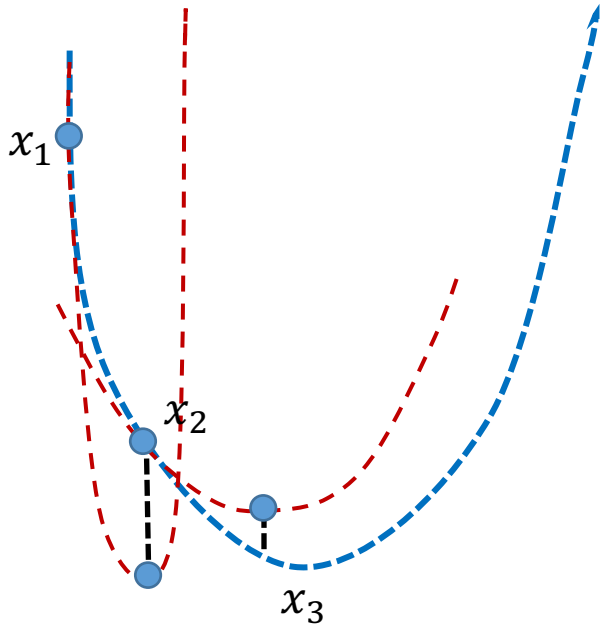




Newton's method (+ Trust region)

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

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

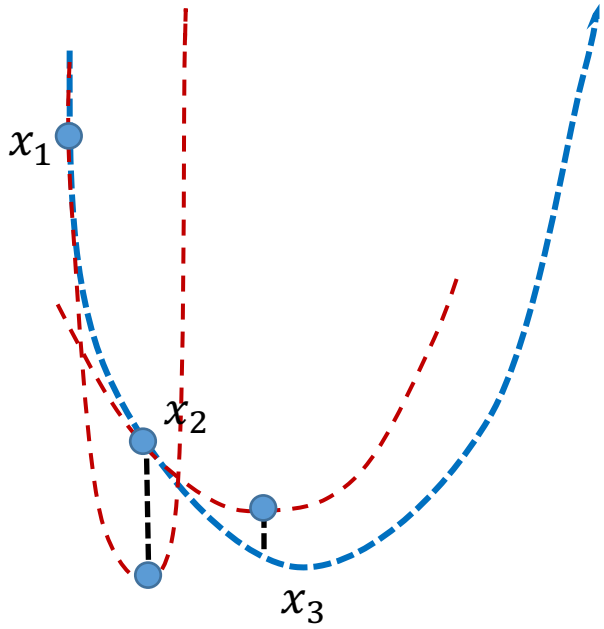


Newton's method (+ Trust region)

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

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?

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

- 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:

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

- 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]

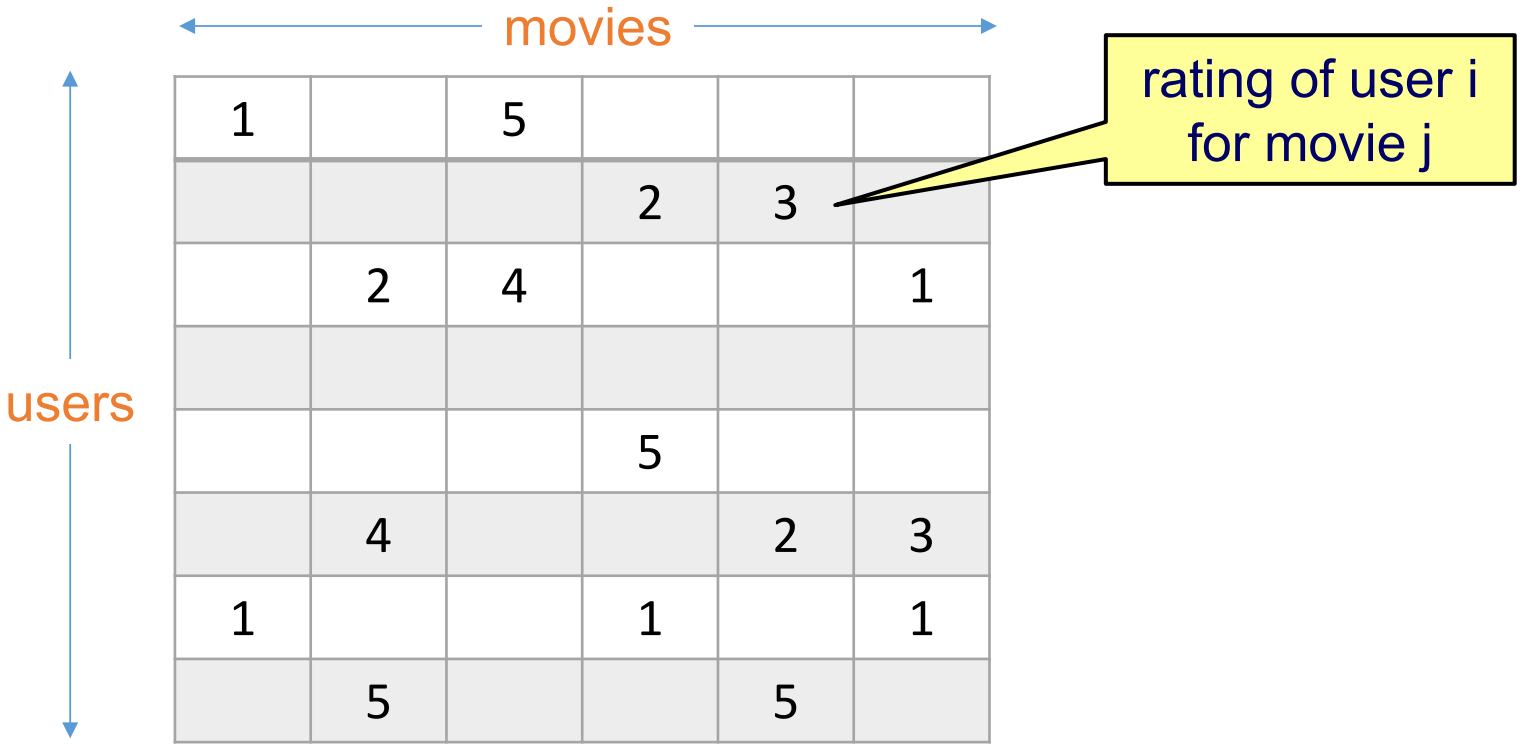
$$O\left(dm \log \frac{1}{\epsilon} + \sqrt{\gamma d}\, d \log \frac{1}{\epsilon}\right)$$

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

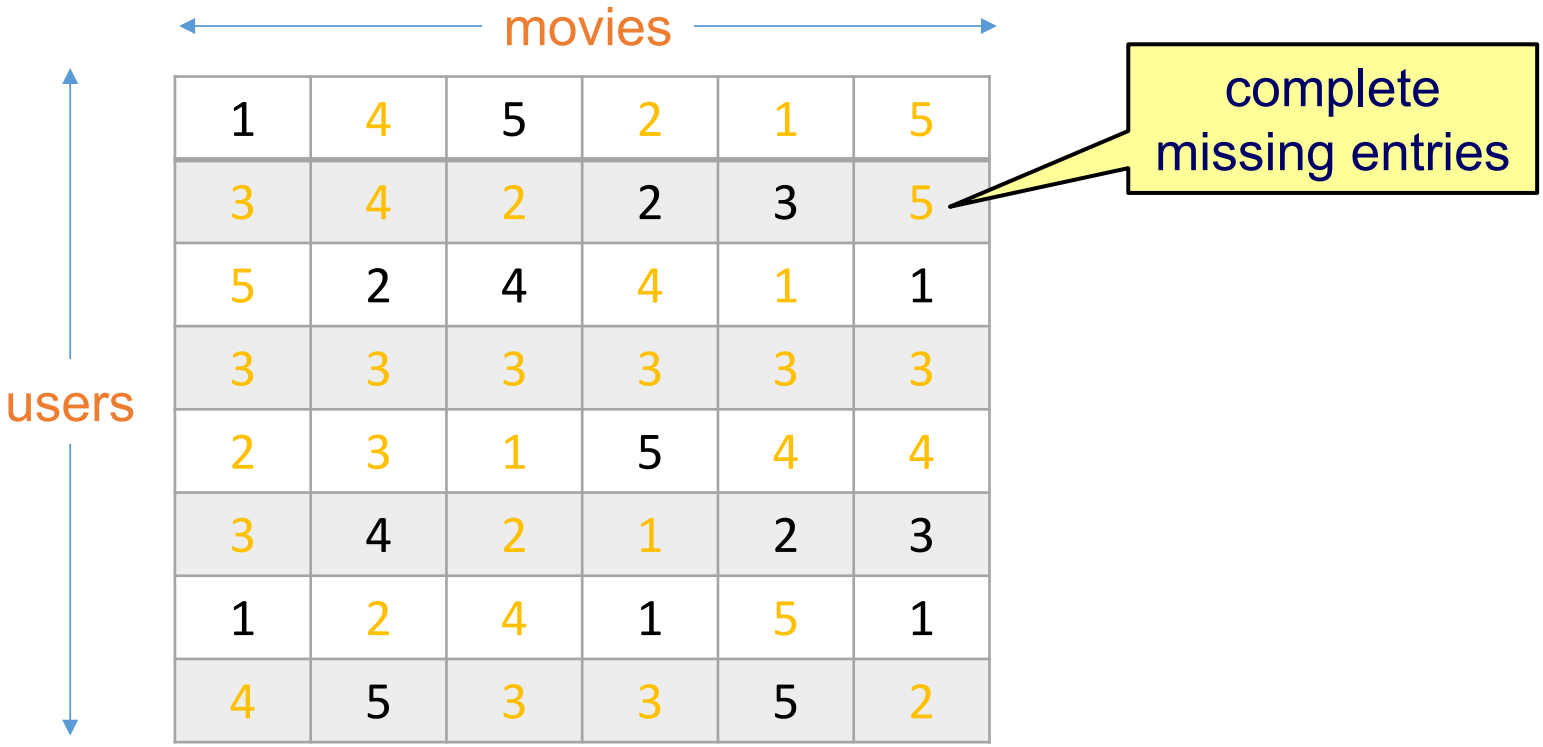
What about constraints??

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

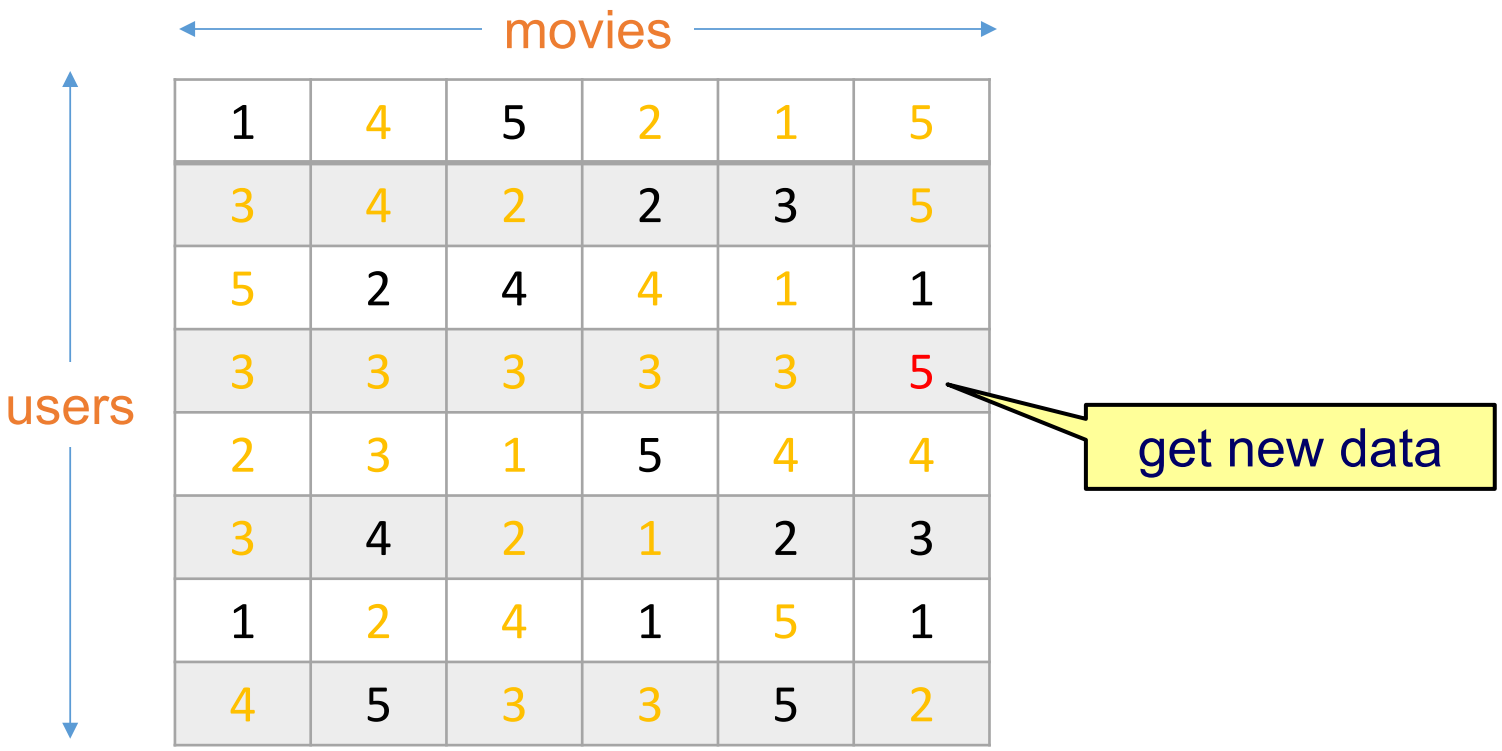
Recommendation systems



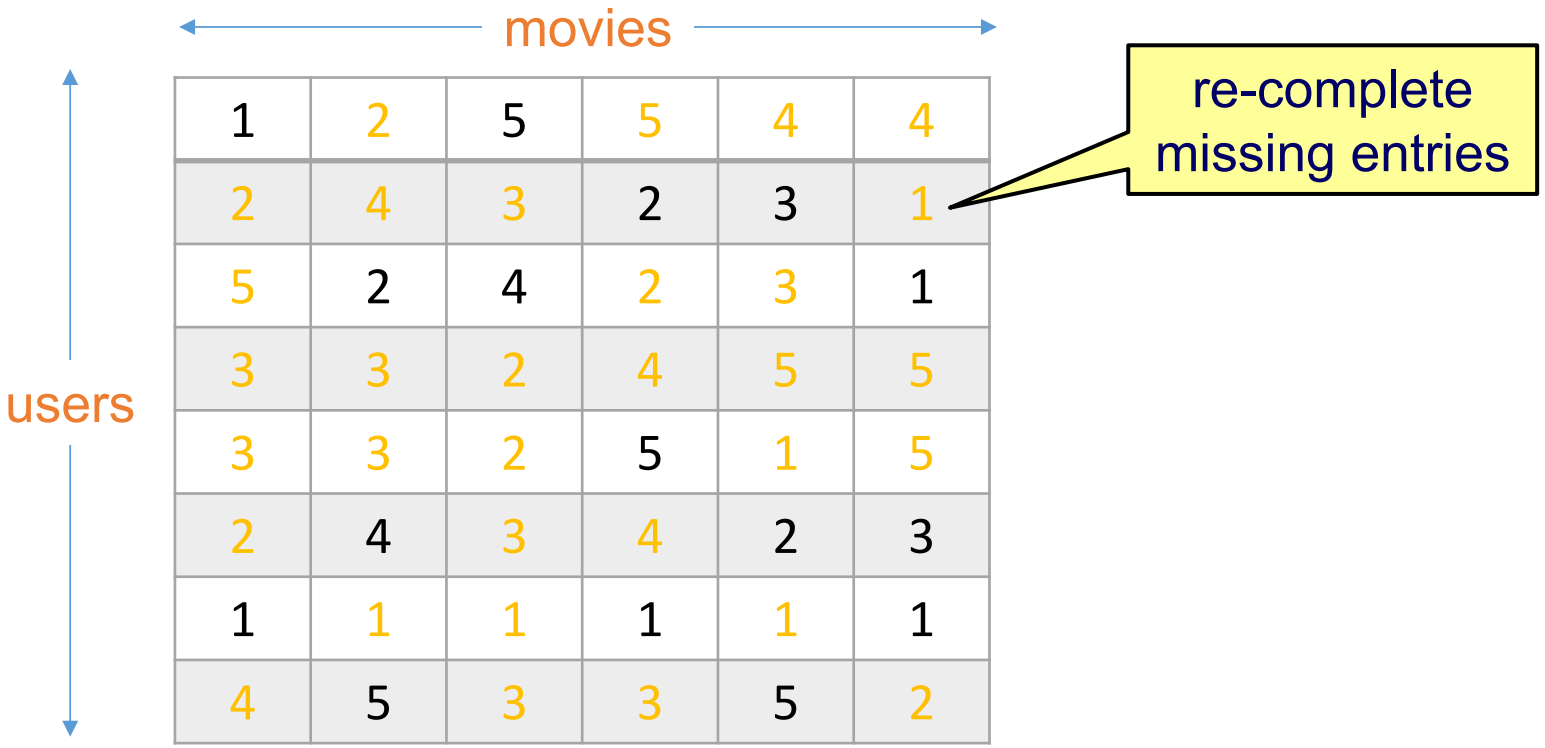
Recommendation systems



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

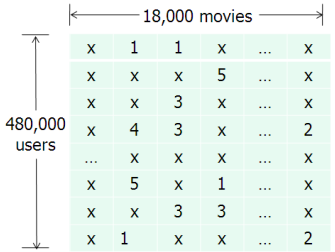
- Trace norm of a matrix = sum of singular values
- K = { X | X is a matrix with trace norm at most D }

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

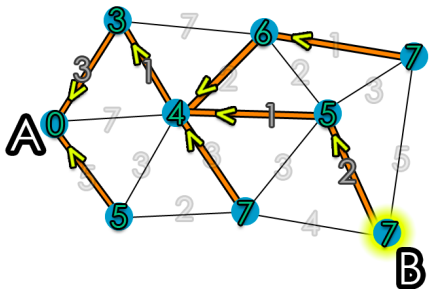
- But: linear optimization over K is easier computing top eigenvector; O(sparsity) time

Projections → linear optimization

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

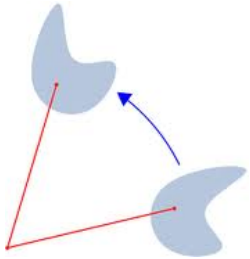


2. Online routing (*K* = flow polytope)
   ~~conic optimization over flow polytope~~



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

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

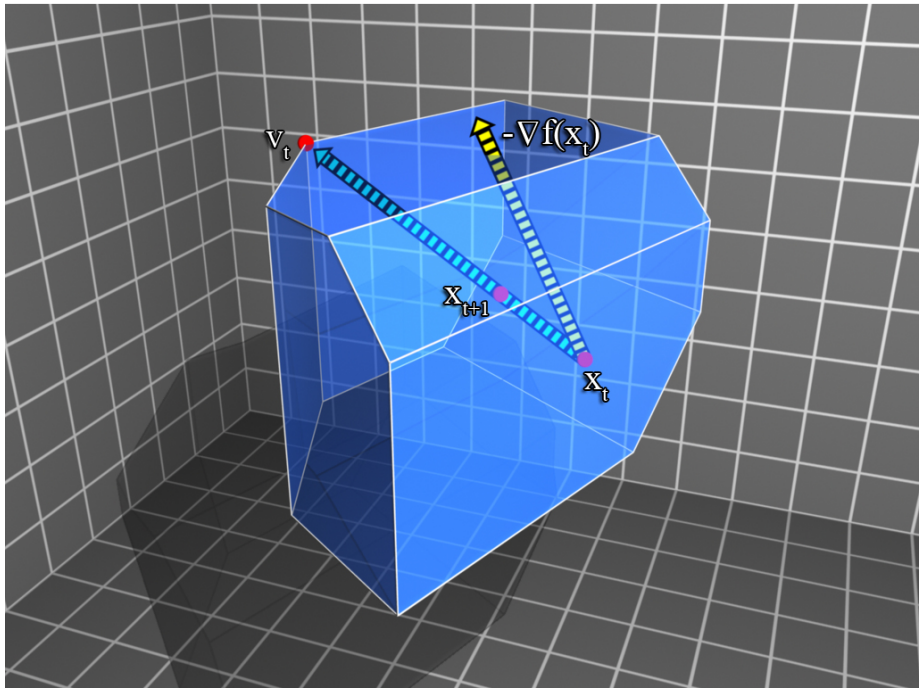


Conditional Gradient algorithm [Frank, Wolfe '56]

Convex opt problem:

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

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



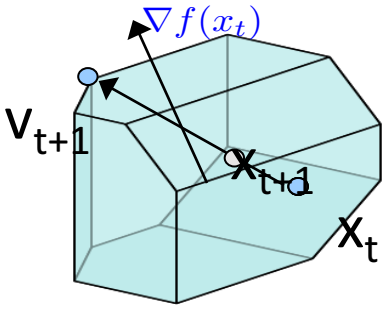
$$v_t = \arg\min_{x \in K} \nabla f(x_t)^\top x$$

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

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

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

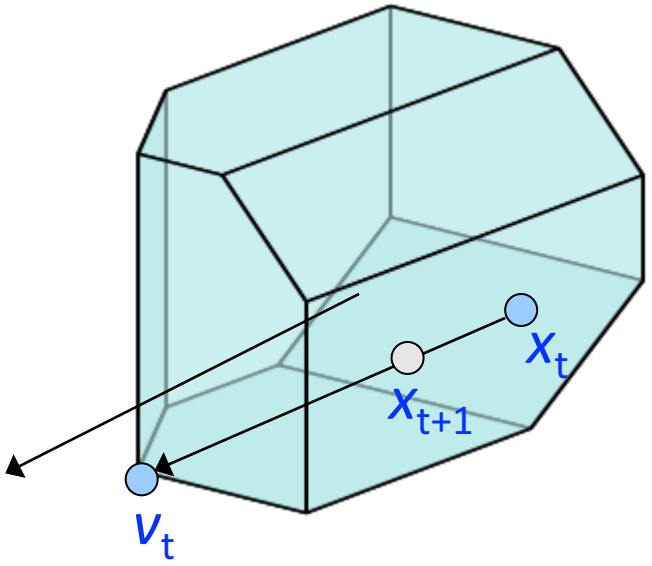
Online Conditional Gradient

- 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: $Regret = O(\sqrt{T})$

Agenda

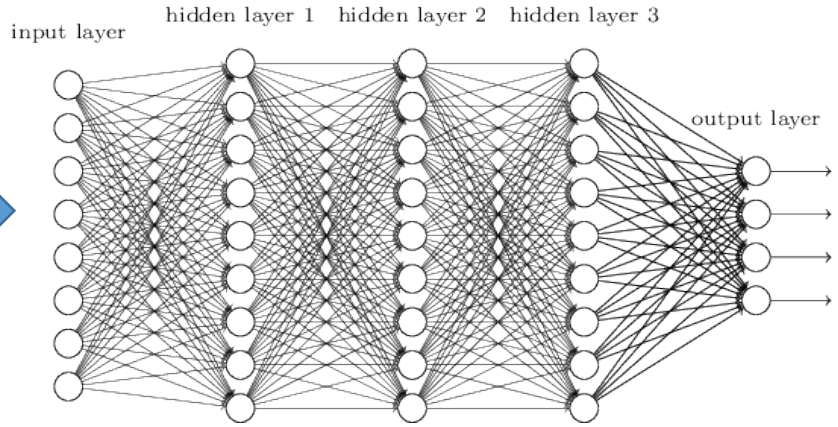
1. Learning as mathematical optimization
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm
4. A touch of state of the art

   * Some slides by Cyril Zhang

Non-convex optimization in ML

Machine

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



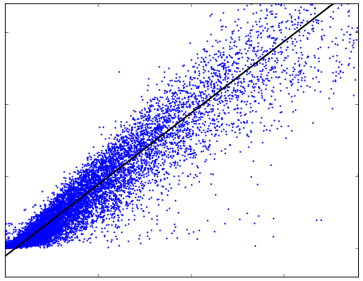
label

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

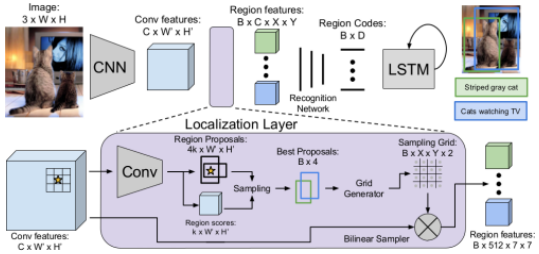
Stochastic Optimization in ML

- **Problem:** find $\operatorname{argmin}_{x \in \mathbb{R}^d} f(x)$
- **Stochastic first-order oracle:** $\mathbb{E}\left[\widetilde{\nabla f}(x)\right] = \nabla f(x), \quad \mathbb{E}\left[\left\|\widetilde{\nabla f}(x)\right\|^2\right] \leq \sigma^2$
- **SGD** [Robbins & Monro '51]:

$f(x) = x^\top \mathbf{A} x + b^\top x$



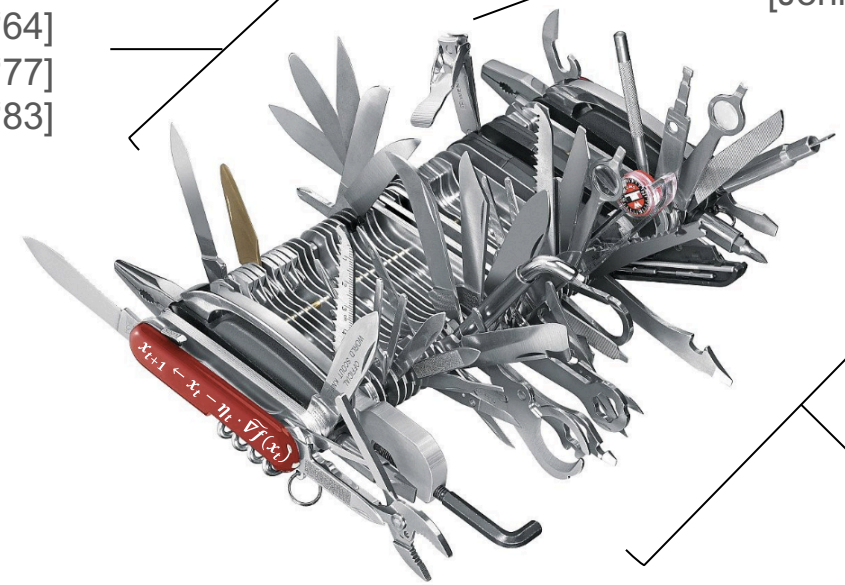
$f(x) = \text{OutrageouslyDeepNeuralNet}(x)$



source: Andrej Karpathy's PhD thesis

“Modern ML is SGD++”

Momentum
[Polyak ‘64]
[Nemirovskii & Yudin ‘77]
[Nesterov ‘83]



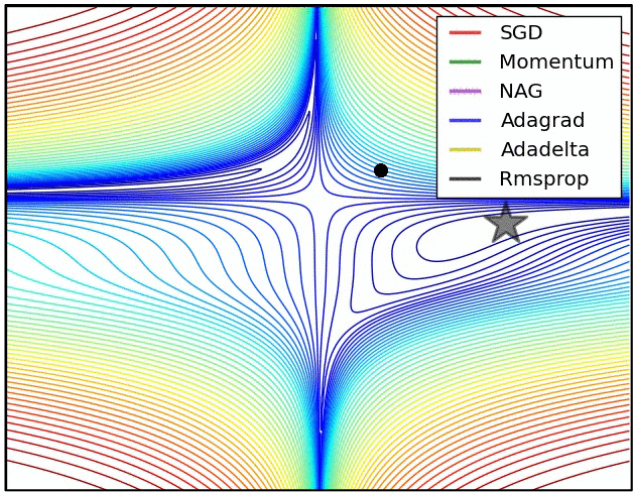
Variance Reduction
[Le Roux et al. ‘12]
[Shalev-Shwartz & Zhang ‘12]
[Johnson & Zhang ‘13]

Adaptive Regularization
[Duchi, Hazan, Singer ‘10]
[Kingma & Ba ‘14]

Adaptive Optimizers: The Usual Recipe

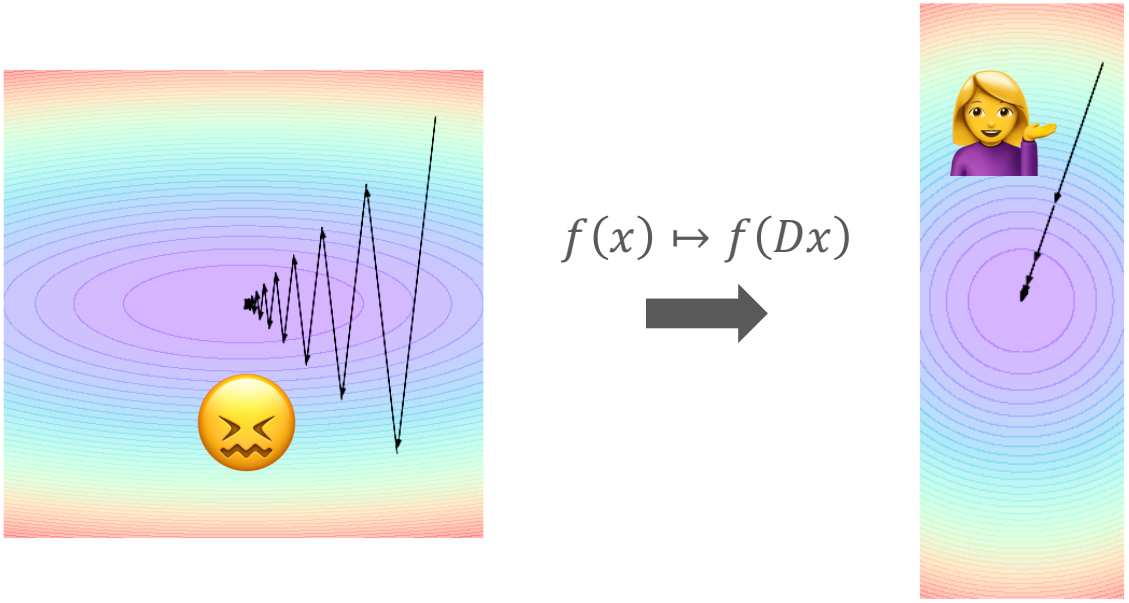
- Each coordinate $x[i]$ gets a learning rate $\mathbf{D}_t[i]$
- $\mathbf{D}_t[i]$ chosen "adaptively" using $\widetilde{\nabla f}(x_{1:t})[i]$ $\quad (g_{1:t}[i]$ for short)

- **AdaGrad:** $D_t[i] := \dfrac{1}{\sqrt{\sum_{s=1}^{t}(g_s[i])^2}}$
- **RMSprop:** $D_t[i] := \dfrac{1}{\sqrt{\sum_{s=1}^{t} 0.99^{t-s}\,(g_s[i])^2}}$
- **Adam:** $D_t[i] := \dfrac{1}{(1-0.99^t)\sqrt{\sum_{s=1}^{t} 0.99^{t-s}\,(g_s[i])^2}}$

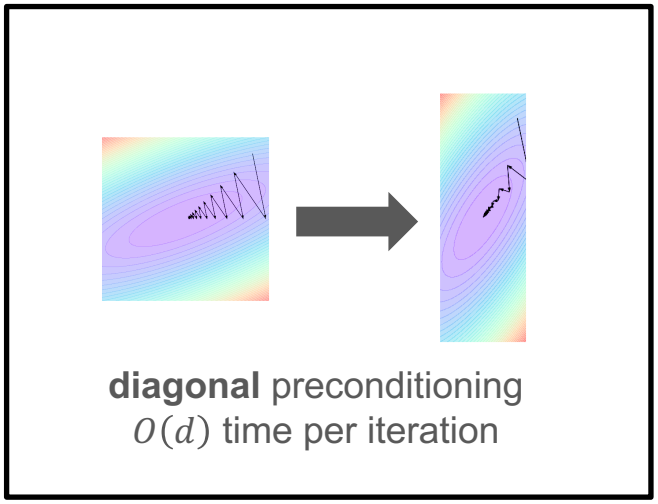


Intuition: Adaptive Preconditioning

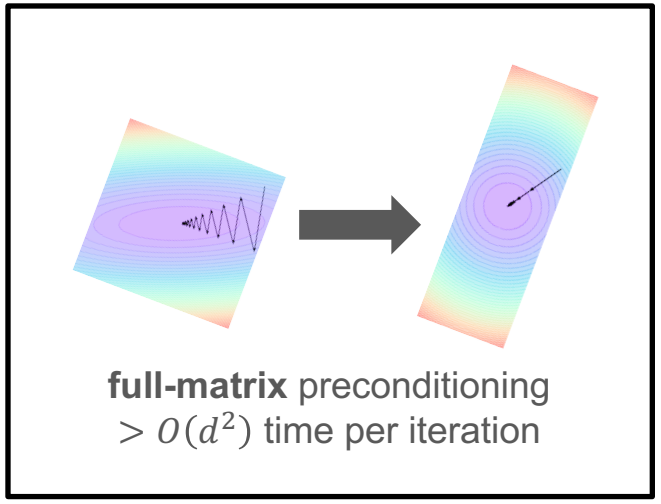
- Per-coordinate scaling matrix $D_t = [\sum_{s=1}^{t} \text{diag}(g_s[i])^2]^{-1/2}$
- Adaptive methods learn D which makes the loss surface more isotropic



What about the *other* AdaGrad?



diagonal preconditioning
$O(d)$ time per iteration



full-matrix preconditioning
$> O(d^2)$ time per iteration

What does adaptive regularization even *do*?!

Theorem 5 *Let the sequence $\{x_t\}$ be defined by Algorithm 1. For x_t generated using the primal-dual subgradient update (3) with $\delta \geq \max_t \|g_t\|_\infty$, for any $x^* \in \mathcal{X}$,*

$$R_\phi(T) \leq \frac{\delta}{\eta}\|x^*\|_2^2 + \frac{1}{\eta}\|x^*\|_\infty^2 \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \eta \sum_{i=1}^{d} \|g_{1:T,i}\|_2 .$$

Main theorem, AdaGrad

- No analysis for *non-convex* optimization

The Case for Full-Matrix Adaptive Regularization

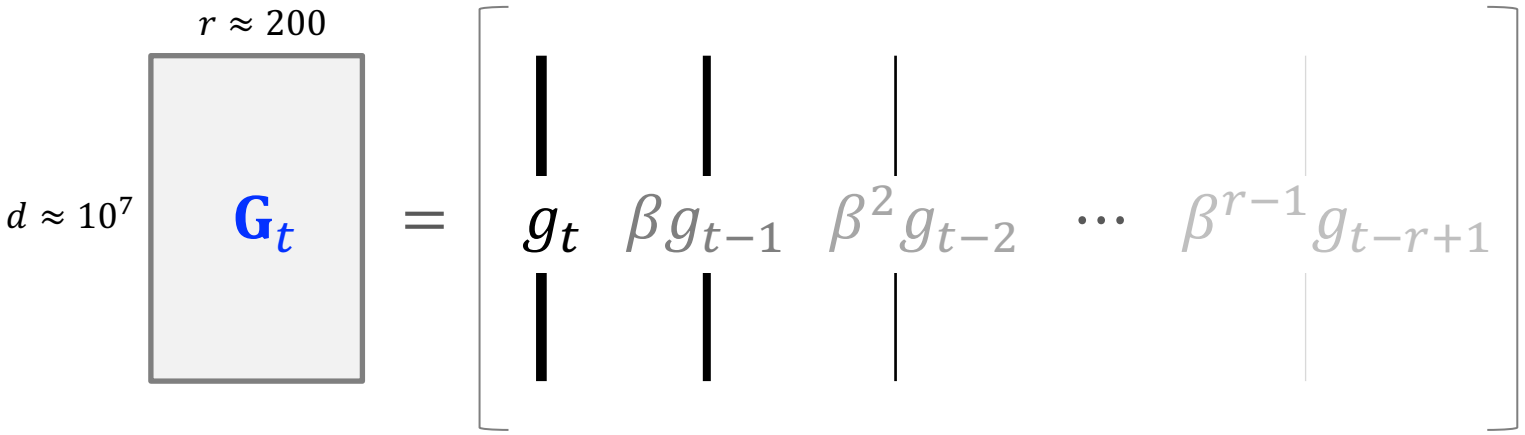
- **GGT**, a new adaptive optimizer
- Efficient full-matrix (low-rank) AdaGrad

- **Theory:** “Adaptive” convergence rate on convex & non-convex f

- **Experiments:** viable in the deep learning era
- GPU-friendly; not much slower than SGD on deep models
- Accelerates training in deep learning benchmarks
- Empirical insights on anisotropic loss surfaces, real and synthetic

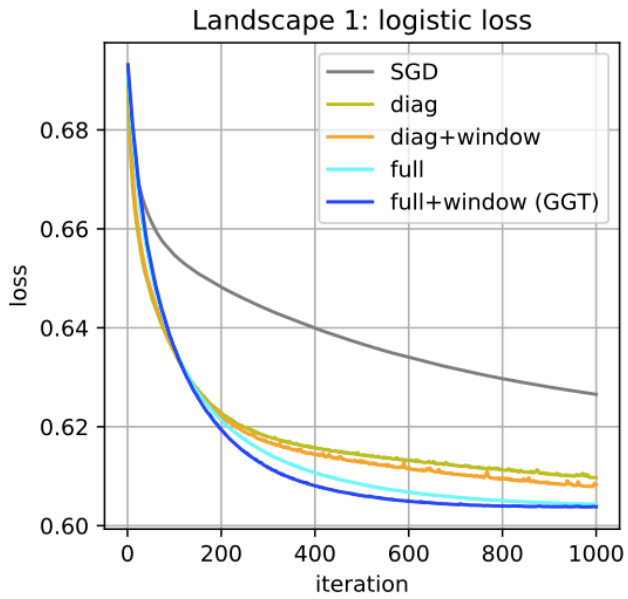
The GGT Algorithm

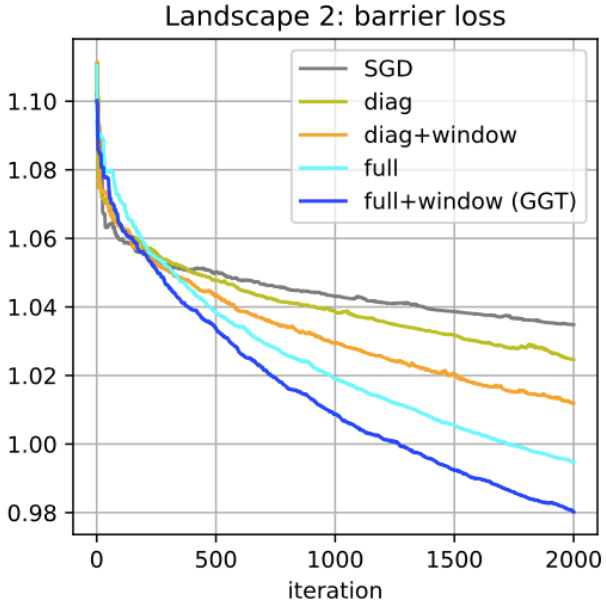
- **SGD:** $x_{t+1} \leftarrow x_t - \eta_t \cdot g_t$
- **AdaGrad:** $x_{t+1} \leftarrow x_t - [\mathrm{diag}(\sum_{s=1}^{t} g_s^2)]^{-1/2} \cdot g_t$
- **Full-Matrix AdaGrad:** $x_{t+1} \leftarrow x_t - [\sum_{s=1}^{t} g_s g_s^\top]^{-1/2} \cdot g_t$
- **GGT:** $x_{t+1} \leftarrow x_t - [\mathbf{G}_t \mathbf{G}_t^\top]^{-1/2} \cdot g_t$



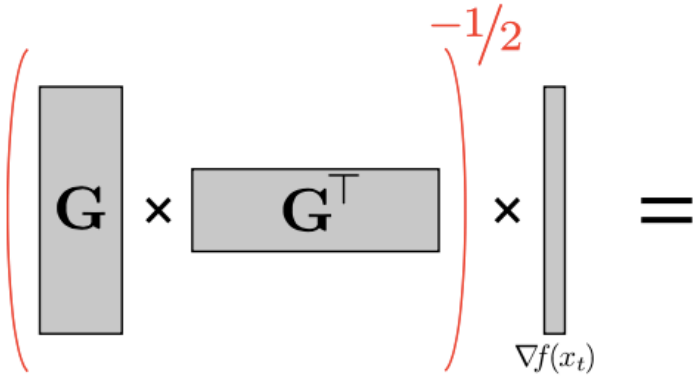
Why a low-rank preconditioner?

- **Answer 1:** want to forget stale gradients (like Adam)
- **Synthetic experiments:** logistic regression, polytope analytic center





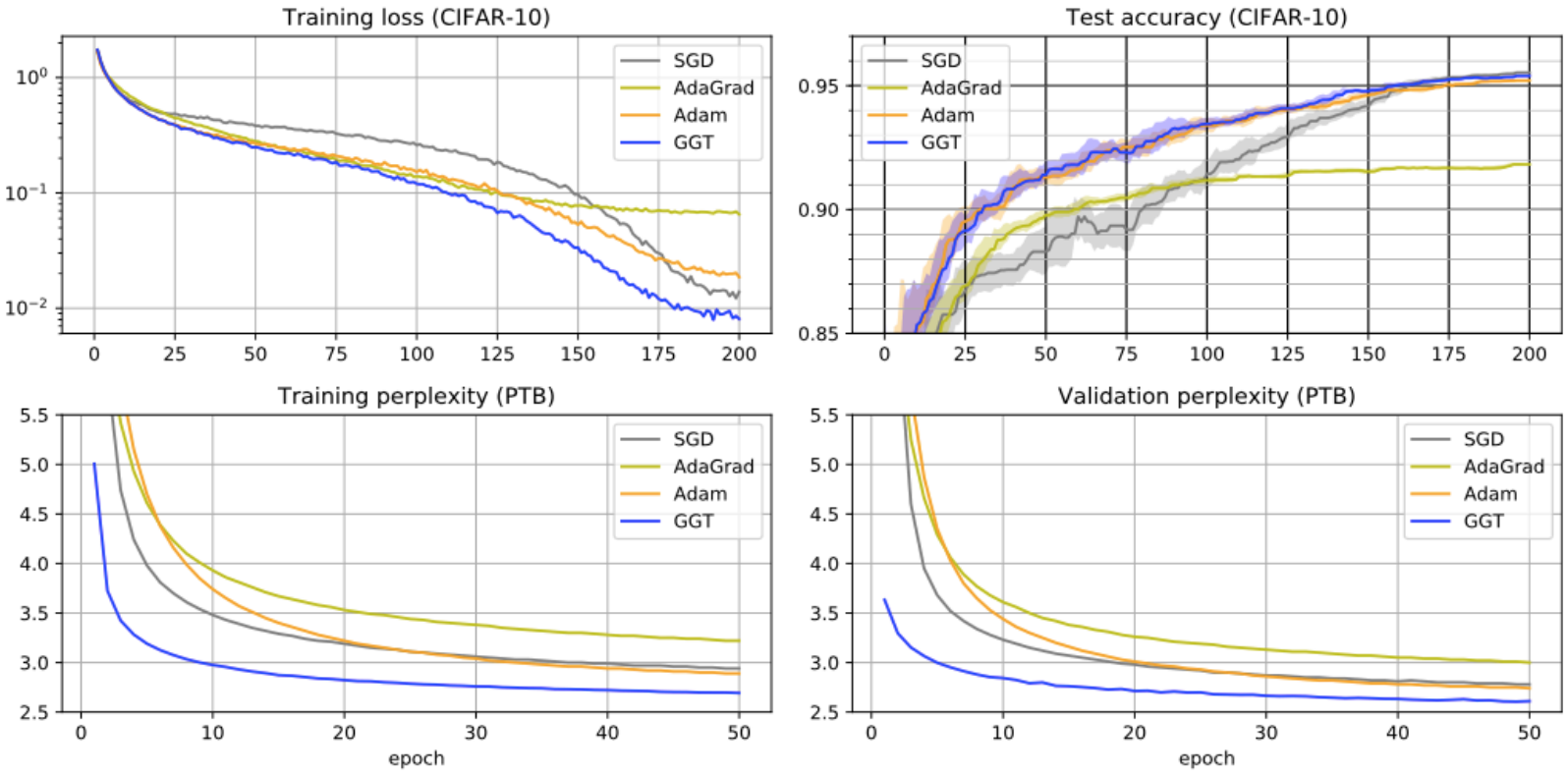
Why a low-rank preconditioner?



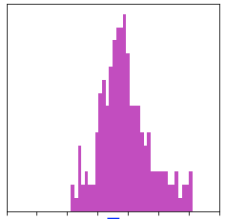
Matrix ops: $O(rd^2)$
Huge SVD: $O(d^3)$

Matrix ops: $O(r^2d)$
Tiny SVD: $O(r^3)$

Large-Scale Experiments (CIFAR-10, PTB)

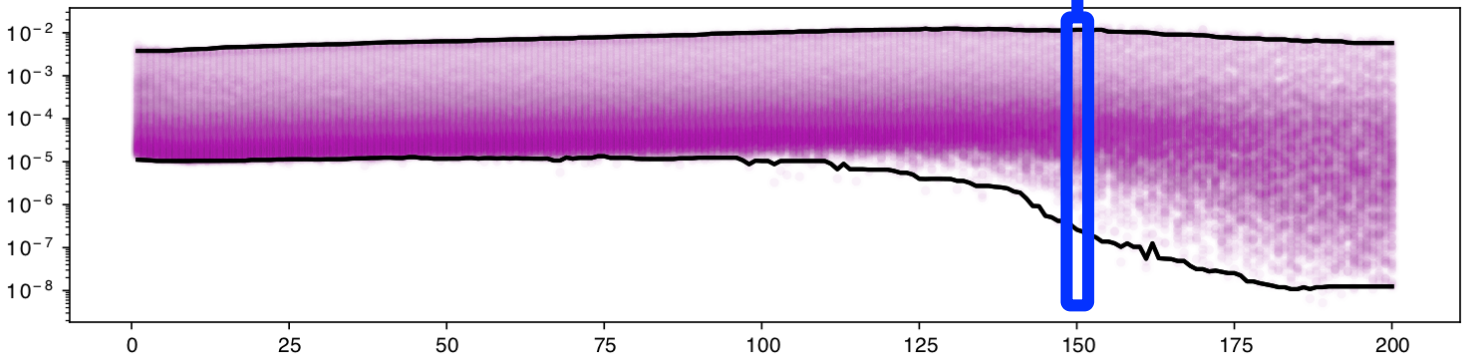


Visualizing Gradient Spectra

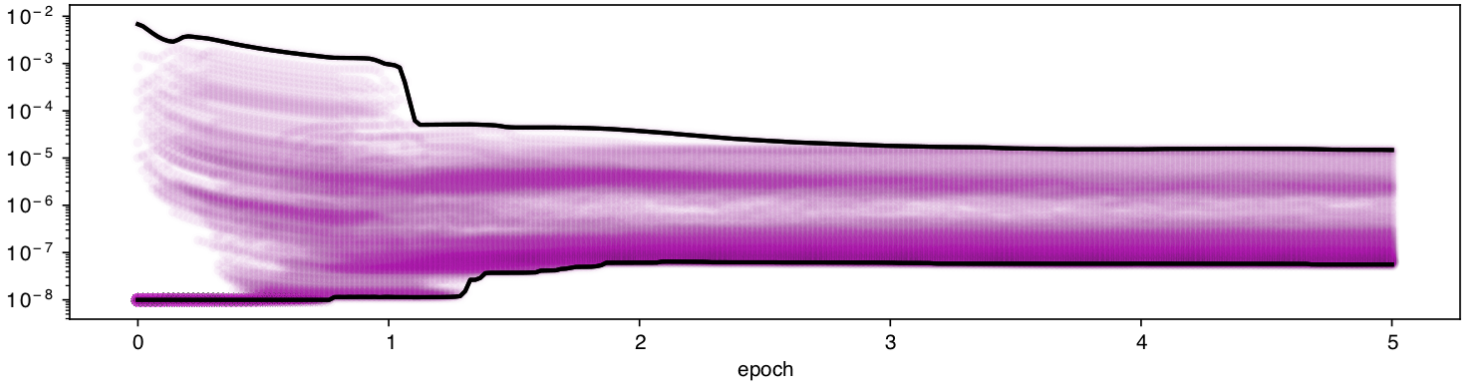


$\text{eigs}(\mathbf{G}_t^\top \mathbf{G}_t)$
$@\ t = 150$

26-layer ResNet
CIFAR-10

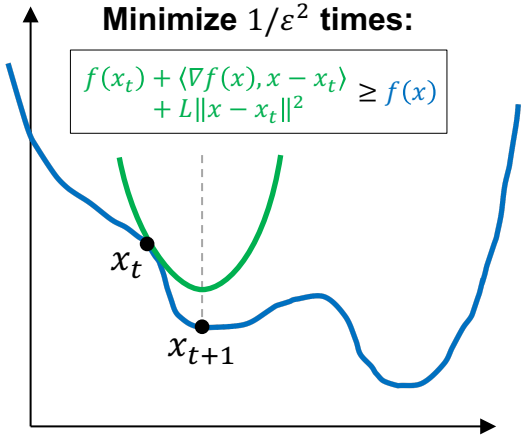


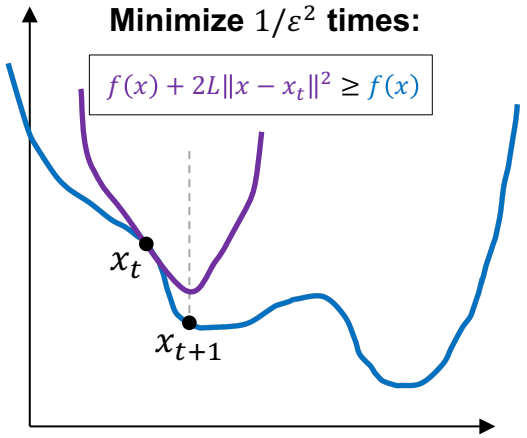
3-layer LSTM
Penn Treebank
(char-level)



Theory: Convergence of Non-Convex SGD

- **Convex:** $f(x_T) \leq \text{argmin}_x f(x) + \varepsilon$ in $O\left(\frac{\sigma^2}{\varepsilon^2}\right)$ steps
- **Non-convex:** $\exists t: \|\nabla f(x_t)\| \leq \varepsilon$ within $O\left(\frac{\sigma^2}{\varepsilon^4}\right)$ steps
- **Reduction via modified descent lemma:**





Theory: Adaptive Convergence Rate of GGT

- Define the *adaptivity ratio* μ:

$$\mu \coloneqq \frac{\sum_{t=1}^{T}\langle g_t, x_t - x^*\rangle}{\|x_1 - x^*\|\sqrt{\sum_{t=1}^{T}\|g_t\|^2}} = \frac{\text{AdaGrad regret}}{\text{worst-case OGD regret}}$$

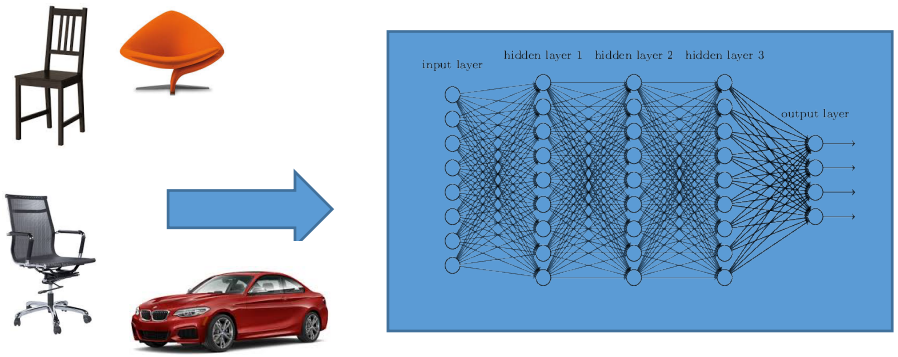
- [DHS10]: $\mu \leq \left[\frac{1}{\sqrt{d}}, \sqrt{d}\right]$ for diag-AdaGrad, sometimes smaller for full AdaGrad
- **Strongly convex losses:** GGT* converges in $\tilde{O}\left(\frac{\mu^2\sigma^2}{\varepsilon}\right)$ steps
- **Non-convex reduction:** GGT* converges in $\tilde{O}\left(\frac{\mu^2\sigma^2}{\varepsilon^4}\right)$ steps
- First step towards analyzing adaptive methods in non-convex optimization

Agenda

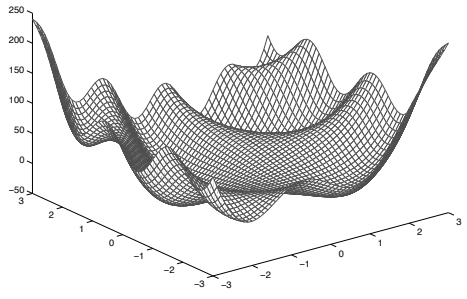
1. Learning as mathematical optimization
   - Empirical Risk Minimization
   - Basics of mathematical optimization
   - Gradient descent + SGD
2. Regularization and Generalization
   - Convex, non-smooth opt.
   - Regret minimization in games, PAC learnability
3. Gradient Descent++
   - Regularization, Adaptive Regularization and AdaGrad
   - Momentum and variance reduction
   - Second order methods
   - Constraints and the Frank-Wolfe algorithm
4. A touch of state of the art

   * Some slides by Cyril Zhang

Recap



Chair/car

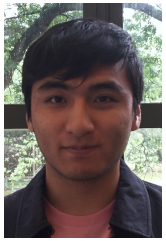


1. Basic non-convex optimization
   - GD and SGD
2. Online learning and stochastic optimization
   - Online gradient descent, non-smooth optimization
3. Regularization and generalization
   - Multiplicative update, mirrored-descent
4. Advanced optimization
   - Adaptive Regularization, Acceleration, variance reduction, second order methods, Frank-Wolfe
5. New algorithms: GGT

collaborators: & Google Princeton-Brain



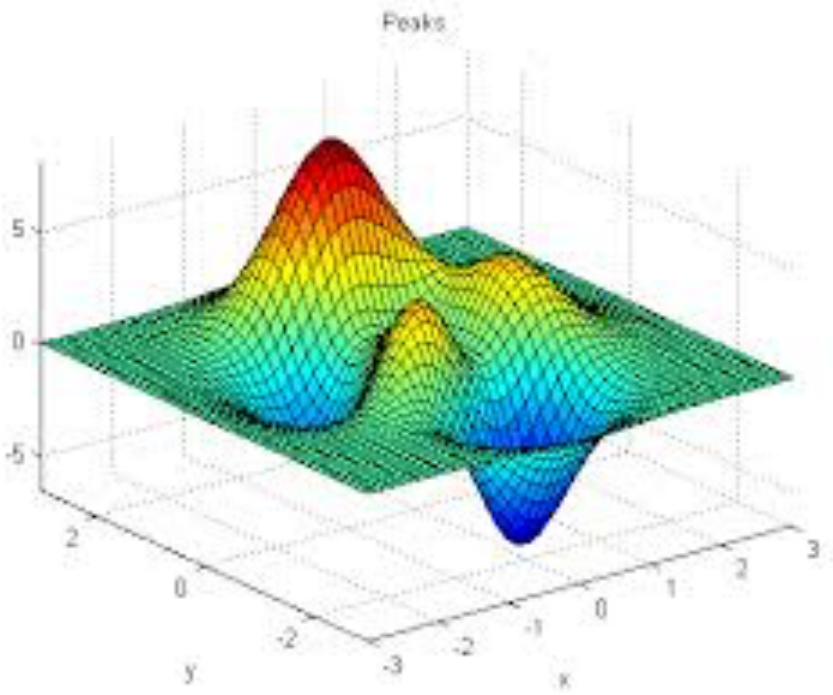





Naman Agarwal, Brian Bullins, Xinyi Chen,
Karan Singh, Cyril Zhang, Yi Zhang

Bibliography & more information, see:

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



Thank you!