Scalable Machine Learning

4. Optimization

Alex Smola
Yahoo! Research and ANU

http://alex.smola.org/teaching/berkeley2012
Stat 260 SP 12

4. Optimization

Optimization

Basic Techniques

- Gradient descent
- Newton's method
- Conjugate Gradient Descent
- Broden-Fletcher-Goldfarb-Shanno (BFGS)

- Constrained Convex Optimization
  - Properties
  - Lagrange function
  - Wolfe dual
- Batch methods
  - Distributed subgradient
  - Bundle methods
- Online methods
  - Unconstrained subgradient
  - Gradient projections
  - Parallel optimization

Why

Parameter Estimation

- Maximum a Posteriori with Gaussian Prior

$$-\log p(\theta|X) = \frac{1}{2\sigma^2}\|\theta\|^2 + \sum_{i=1}^{m} g(\theta) - \langle \phi(x_i), \theta \rangle + \text{const.}$$

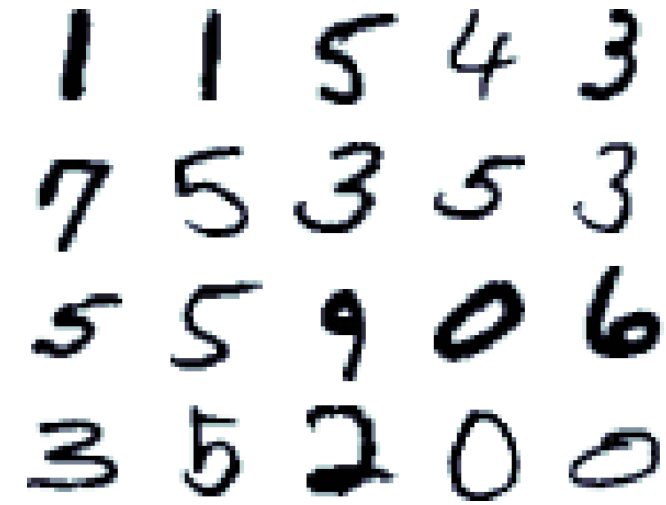
prior

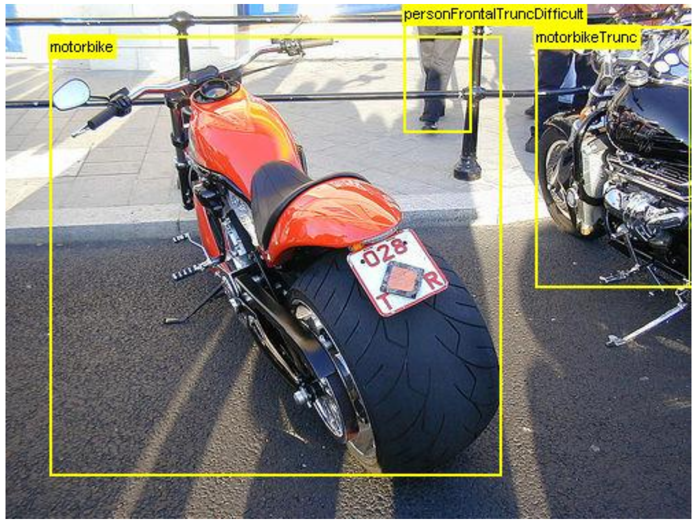
data

- We have lots of data
  - Does not fit on single machine
  - Bandwidth constraints
  - May grow in real time
- Regularized Risk Minimization yields similar problems (more on this in a later lecture)

Batch and Online

- Batch
  - Very large dataset available
  - Require parameter only at the end
    - optical character recognition
    - speech recognition
    - image annotation / categorization
    - machine translation
- Online
  - Spam filtering
  - Computational advertising
  - Content recommendation / collaborative filtering

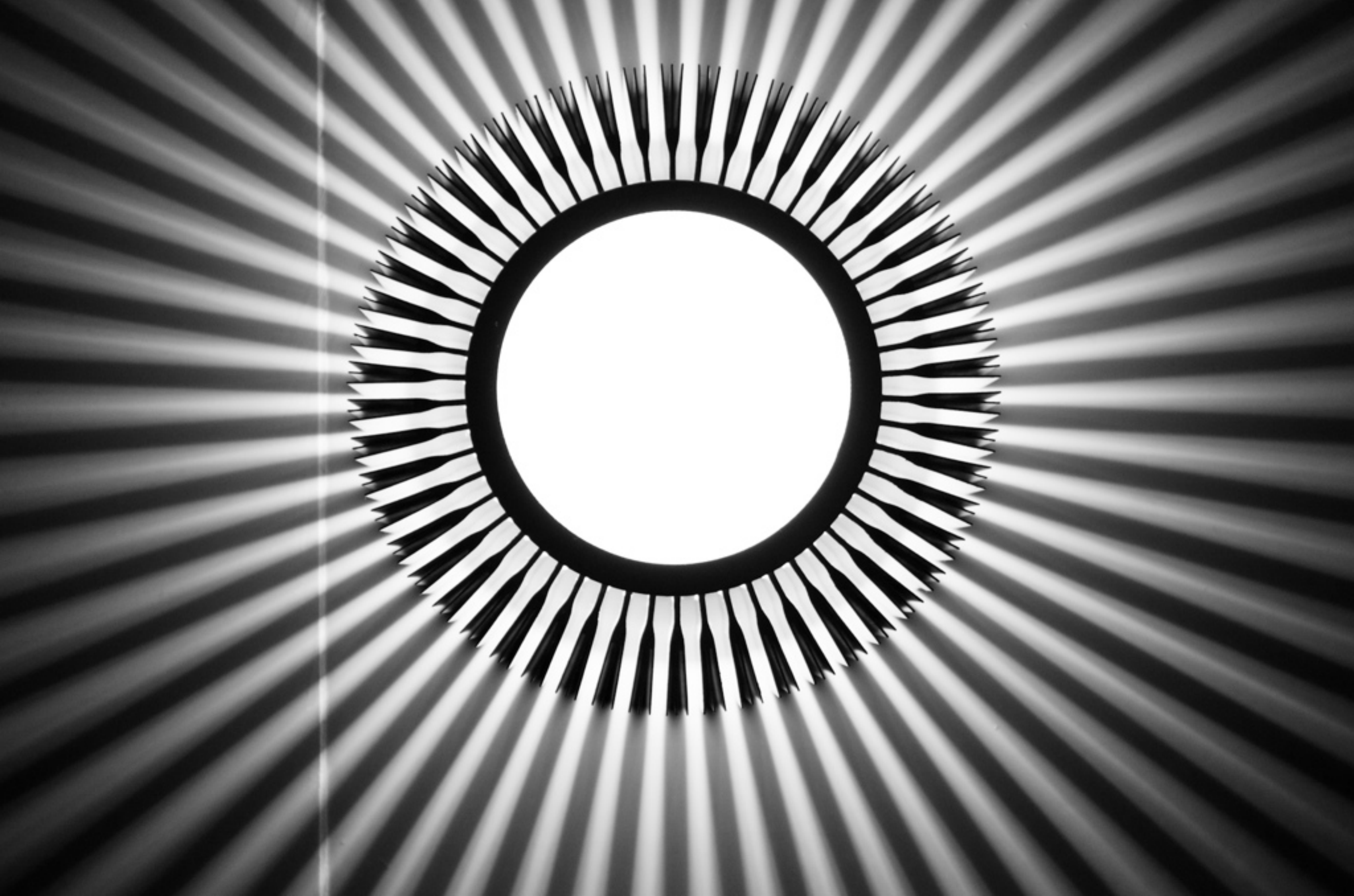




Many parameters

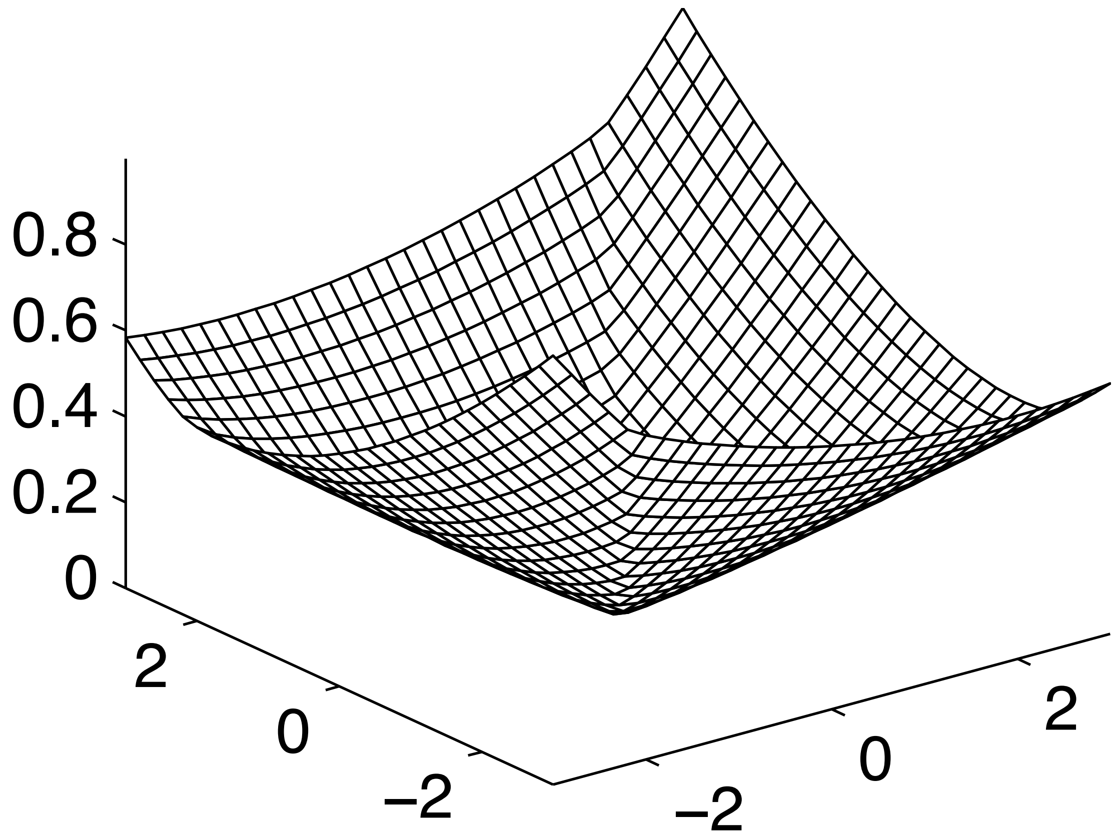
- 100 million to 1 Billion users
  Personalized content provision - impossible to adjust all parameters by heuristic/manually
- 1,000-10,000 computers
  Cannot exchange all data between machines, Distributed optimization, multicore
- Large networks
  Nontrivial parameter dependence structure

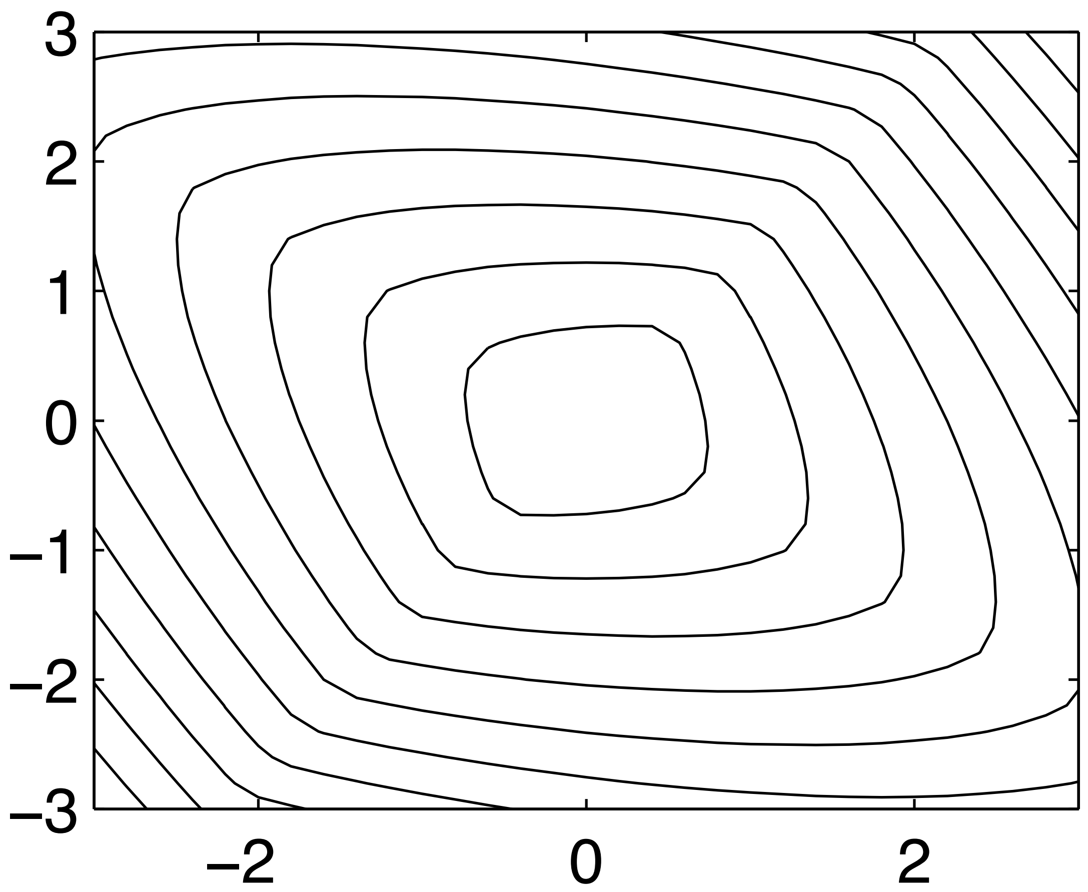
4.1 Unconstrained Problems



Convexity 101

Convexity 101





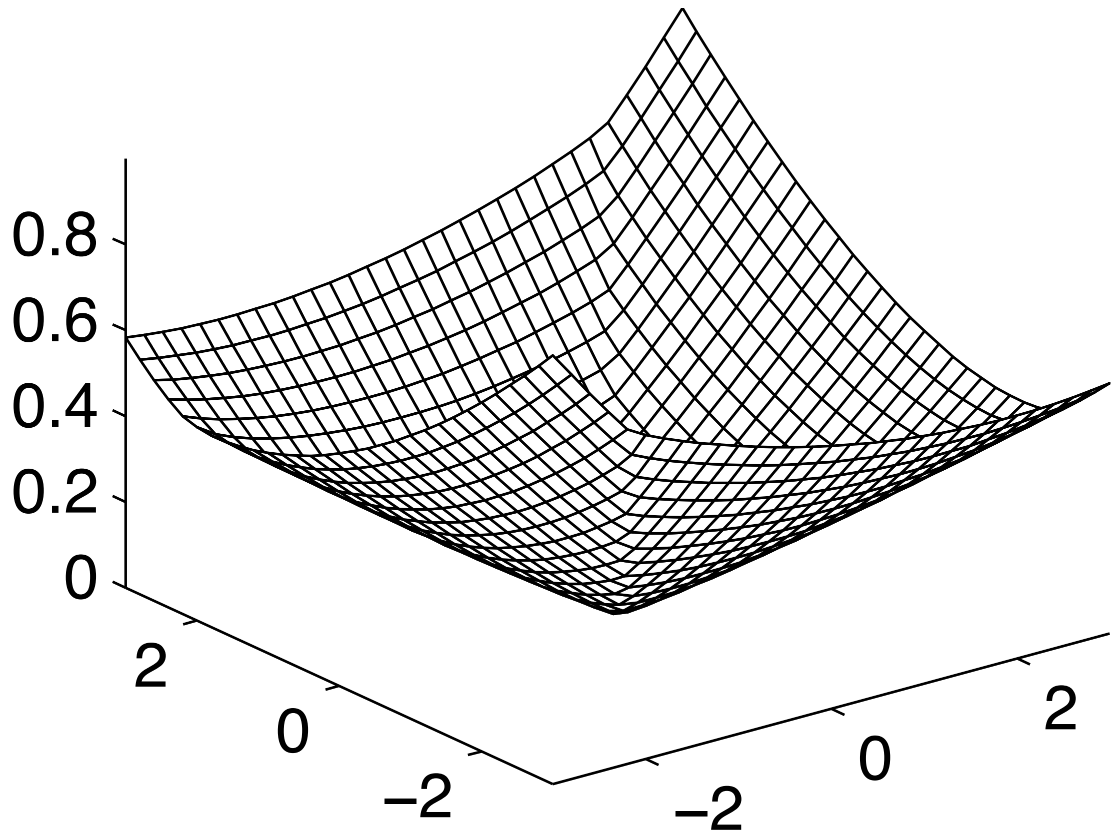
- **Convex set**

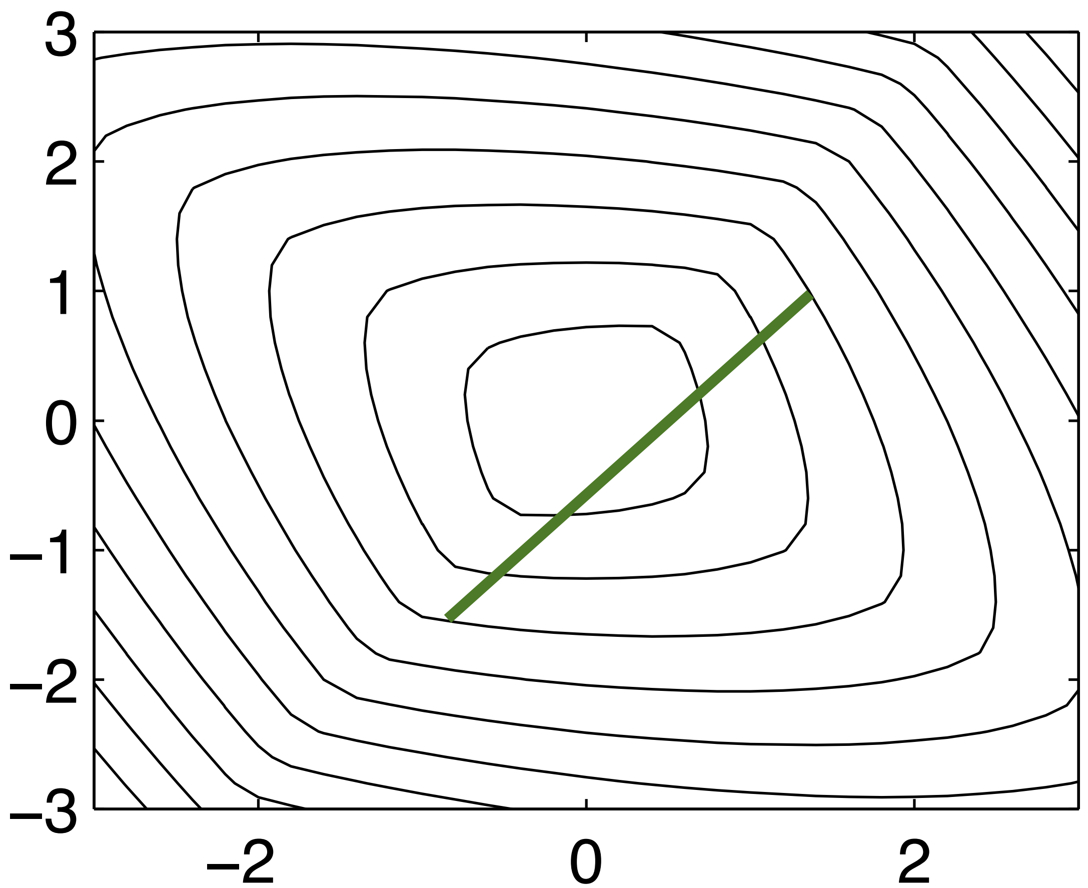
  For $x, x' \in X$ it follows that $\lambda x + (1-\lambda)x' \in X$ for $\lambda \in [0,1]$

- **Convex function**

$$\lambda\lambda f(x) + (1-\lambda)f(x') \geq f(\lambda x + (1-\lambda)x') \text{ for } \lambda \in [0,1]$$

Convexity 101





- **Convex set**

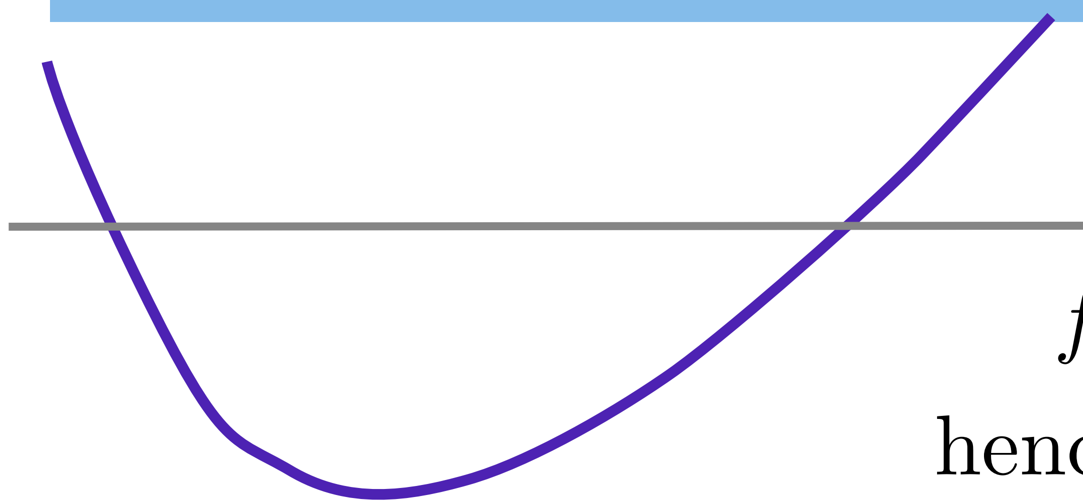
For $x, x' \in X$ it follows that $\lambda x + (1-\lambda)x' \in X$ for $\lambda \in [0,1]$

- **Convex function**

$$\lambda\lambda f(x) + (1-\lambda)f(x') \geq f(\lambda x + (1-\lambda)x') \text{ for } \lambda \in [0,1]$$

Convexity 101

- Below-set of convex function is convex

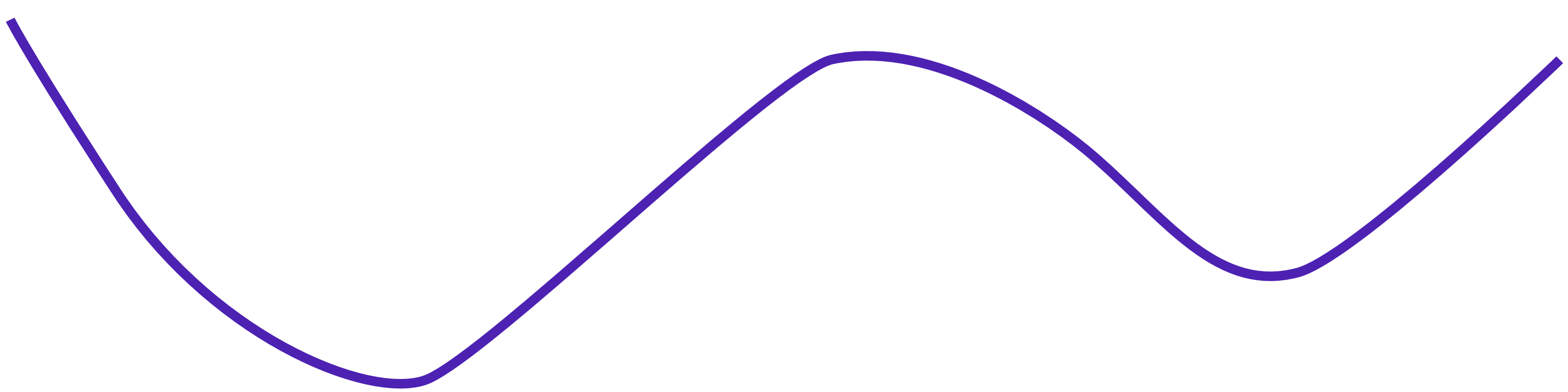


$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda)f(x$$

$$\text{hence } \lambda x + (1-\lambda)x' \in X \text{ for } x, x' \in X$$

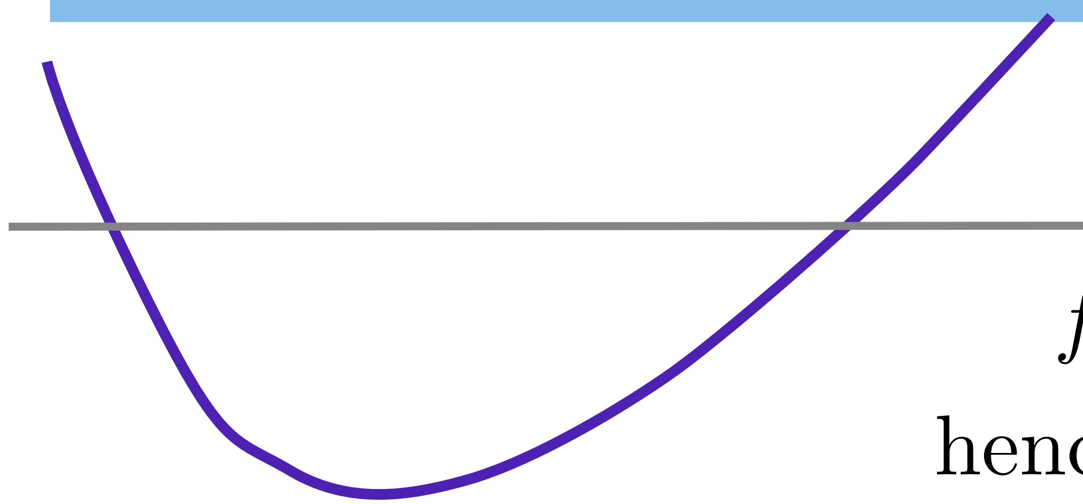
- Convex functions don't have local minima

Proof by contradiction - linear interpolation breaks local minimum condition



Convexity 101

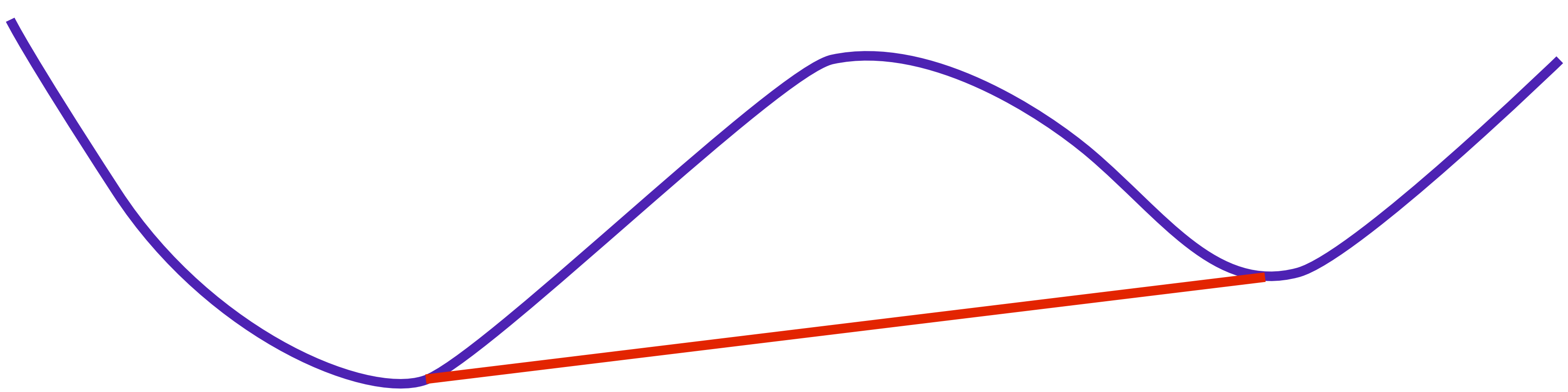
- Below-set of convex function is convex



$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x$$

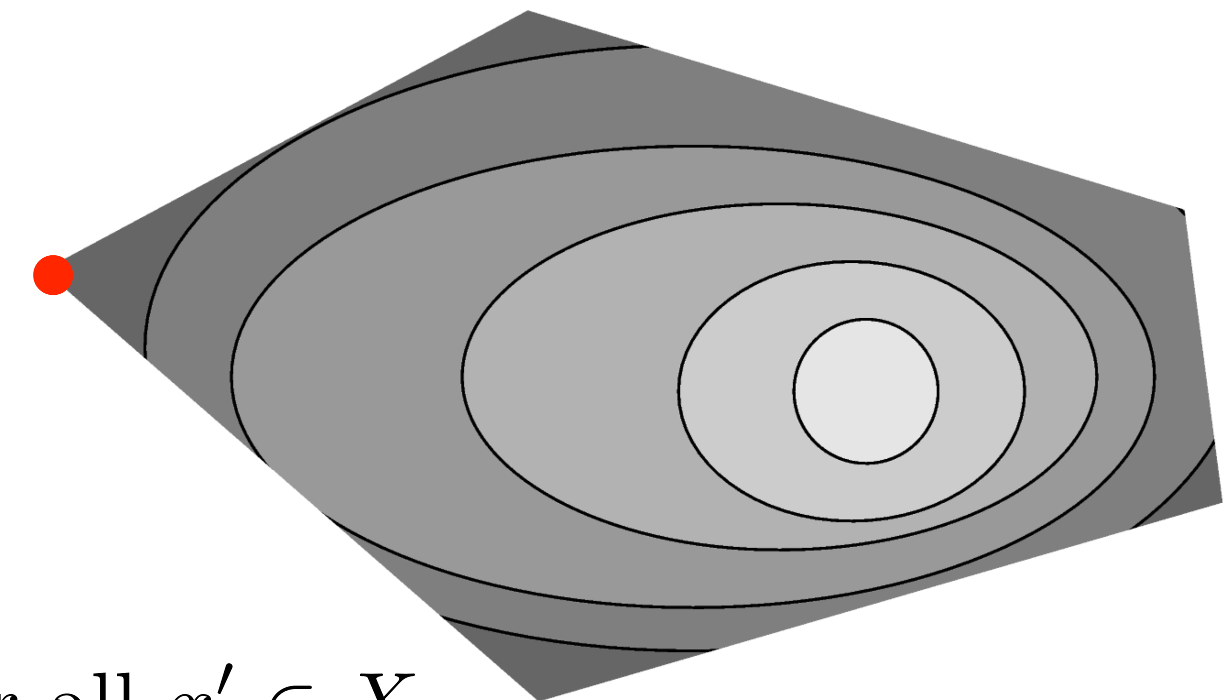
$$\text{hence } \lambda x + (1-\lambda)x' \in X \text{ for } x, x' \in X$$

- Convex functions don't have local minima
Proof by contradiction - linear interpolation breaks local minimum condition



Convexity 101

- **Vertex of a convex set**
  Point which cannot be extrapolated within convex set
  $\lambda x + (1-\lambda)x' \notin X \text{ for } \lambda > 1 \text{ for all } x' \in X$



- **Convex hull**

$$\operatorname{co} X := \left\{ \bar{x} \,\middle|\, \bar{x} = \sum_{i=1}^{n} \alpha_i x_i \; \textit{where } n \in \mathbb{N}, \alpha_i \geq 0 \; \textit{and} \; \sum_{i=1}^{n} \alpha_i \leq 1 \right\}$$

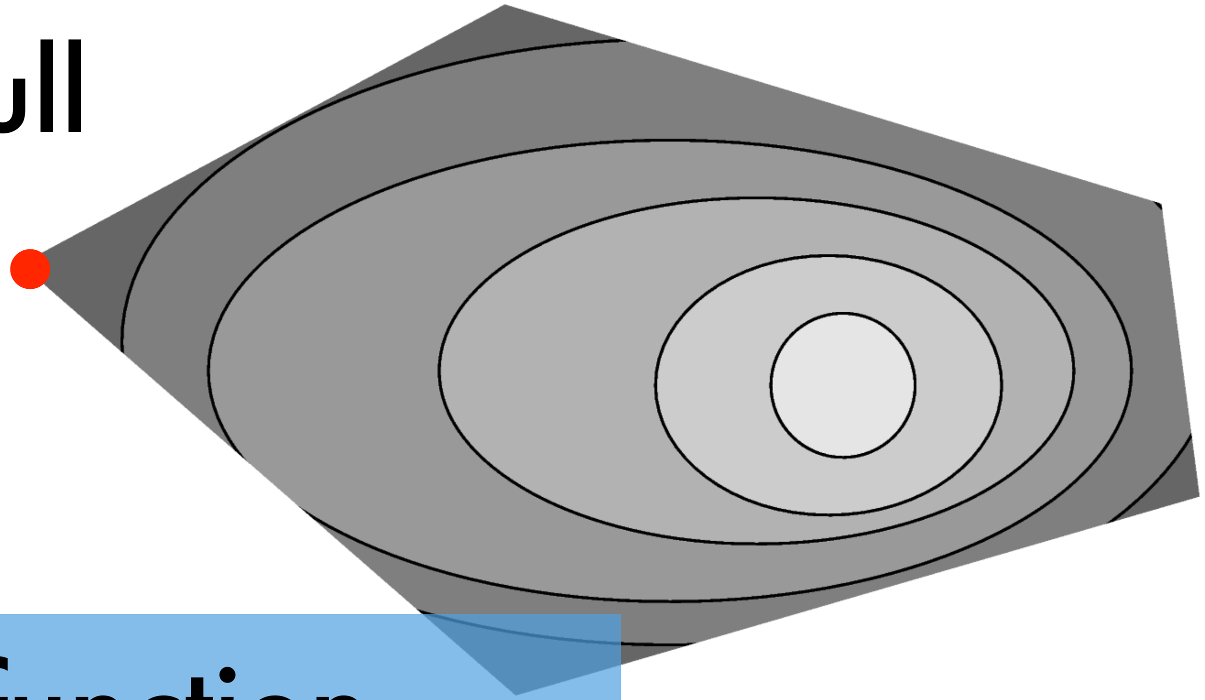
- Convex hull of set is a convex set (proof trivial)

Convexity 101

- Supremum on convex hull

$$\sup_{x \in X} f(x) = \sup_{x \in \mathrm{co} X} f(x)$$

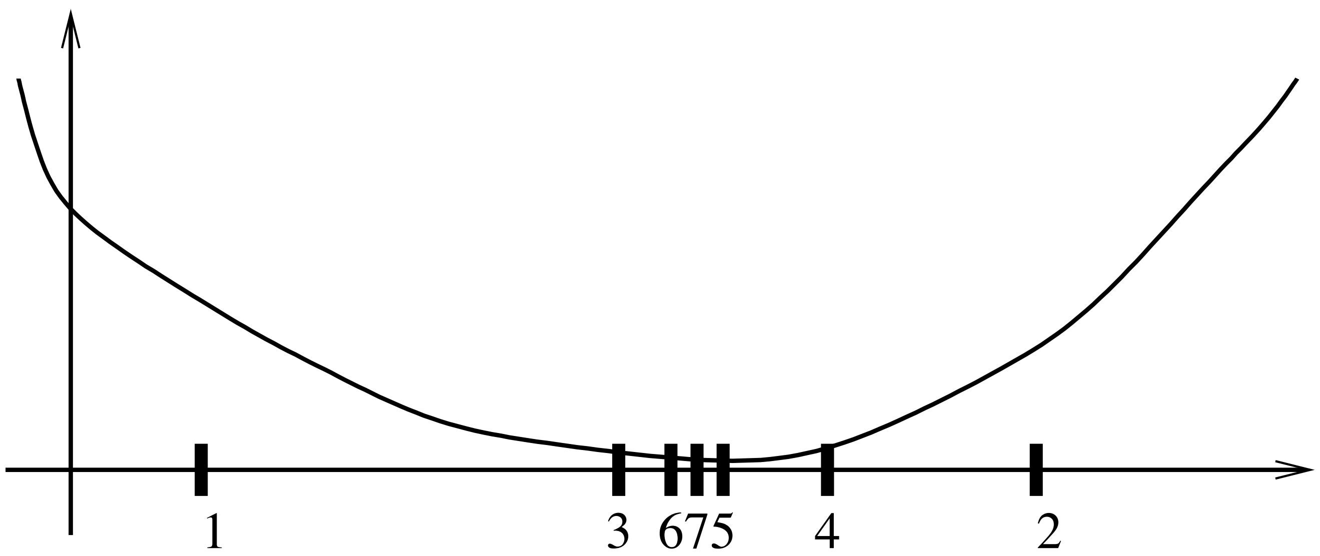
  Proof by contradiction



- Maximum over convex function on convex set is obtained on vertex
  - Assume that maximum inside line segment
  - Then function cannot be convex
  - Hence it must be on vertex

Gradient descent

One dimensional problems

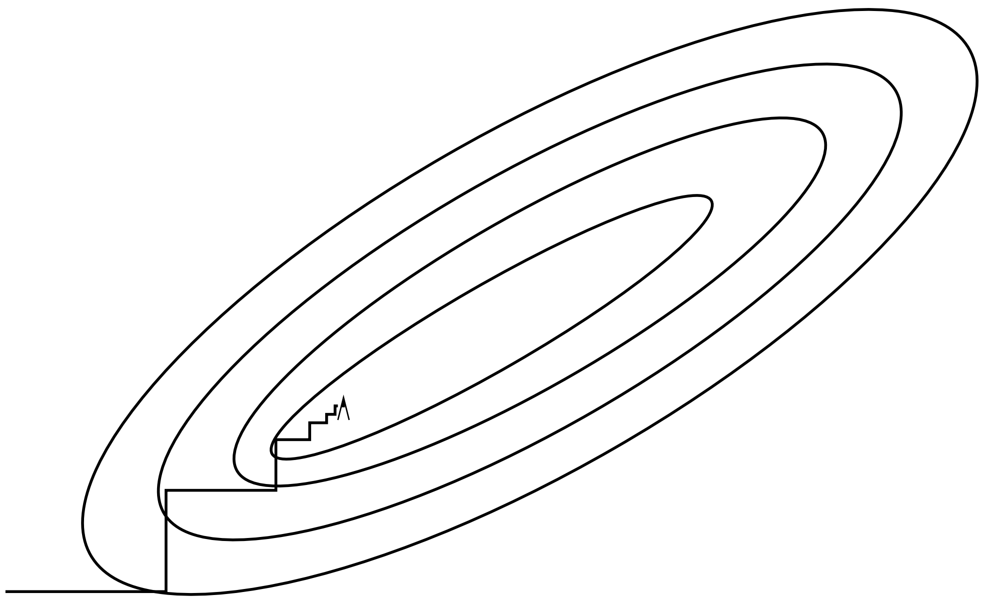


Require: a, b, Precision ϵ
Set $A = a, B = b$
repeat
if $f'\left(\frac{A+B}{2}\right) > 0$ **then**
$B = \frac{A+B}{2}$
else
$A = \frac{A+B}{2}$
end if
until $(B - A)\min(|f'(A)|, |f'(B)|) \leq \epsilon$
Output: $x = \frac{A+B}{2}$

solution on the left

- Key Idea
  - For differentiable f search for x with f′(x) = 0
  - Interval bisection (derivative is monotonic)
  - Need log (A-B) - log ε to converge
- Can be extended to nondifferentiable problems (exploit convexity in upper bound and keep 5 points)

Gradient descent



- Key idea
  - Gradient points into descent direction
  - Locally gradient is good approximation of objective function
- GD with Line Search
  - Get descent direction
  - Unconstrained line search
  - Exponential convergence for strongly convex objective

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Convergence Analysis

- Strongly convex function

$$f(y) \geq f(x) + \langle y - x, \partial_x f(x) \rangle + \frac{m}{2} \|y - x\|^2$$

- Progress guarantees (minimum x*)

$$f(x) - f(x^*) \geq \frac{m}{2} \|x - x^*\|^2$$

- Lower bound on the minimum (set y= x*)

$$\begin{aligned} f(x) - f(x^*) &\leq \langle x - x^*, \partial_x f(x) \rangle - \frac{m}{2} \|x^* - x\|^2 \\ &\leq \sup_y \langle x - y, \partial_x f(x) \rangle - \frac{m}{2} \|y - x\|^2 \\ &= \frac{1}{2m} \|\partial_x f(x)\|^2 \end{aligned}$$

Convergence Analysis

- Bounded Hessian

$$f(y) \leq f(x) + \langle y - x, \partial_x f(x) \rangle + \frac{M}{2} \|y - x\|^2$$

$$\implies f(x + tg_x) \leq f(x) - t \|g_x\|^2 + \frac{M}{2} t^2 \|g_x\|^2$$

$$\leq f(x) - \frac{1}{2M} \|g_x\|^2$$

  Using strong convexity

$$\implies f(x + tg_x) - f(x^*) \leq f(x) - f(x^*) - \frac{1}{2M} \|g_x\|^2$$

$$\leq f(x) - f(x^*) \left[1 - \frac{m}{M}\right]$$

- Iteration bound $\frac{M}{m} \log \frac{f(x) - f(x^*)}{\epsilon}$

Distributed Implementation

Basic steps

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Basic steps

distribute data over
several machines

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Basic steps

distribute data over several machines

compute partial gradients and aggregate

given a starting point $x \in \mathbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Basic steps

distribute data over several machines

compute partial gradients and aggregate

update value in search direction and feed back

given a starting point $x \in \textbf{dom} f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Basic steps

distribute data over several machines

compute partial gradients and aggregate

given a starting point $x \in \mathbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

update value in search direction and feed back

communicate final value to each machine

Basic steps

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- Map: compute gradient on subblock and emit
- Reduce: aggregate parts of the gradients
- Communicate the aggregate gradient back to all machines

Basic steps

distribute data over several machines

given a starting point $x \in \mathbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- Map: compute gradient on subblock and emit
- Reduce: aggregate parts of the gradients
- Communicate the aggregate gradient back to all machines

Basic steps

distribute data over several machines

compute partial gradients and aggregate

given a starting point $x \in \mathbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

- Map: compute gradient on subblock and emit
- Reduce: aggregate parts of the gradients
- Communicate the aggregate gradient back to all machines

Basic steps

- Repeat until converged
  - Map: compute function & derivative at given parameter t
  - Reduce: aggregate parts of function and derivative
  - Decide based on f(x) and f’(x) which interval to pursue
- Send updated parameter to all machines

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Basic steps

- Repeat until converged
  - Map: compute function & derivative at given parameter t
  - Reduce: aggregate parts of function and derivative
  - Decide based on f(x) and f’(x) which interval to pursue
- Send updated parameter to all machines

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

update value in search direction and feed back

Basic steps

- Repeat until converged
  - Map: compute function & derivative at given parameter t
  - Reduce: aggregate parts of function and derivative
  - Decide based on f(x) and f′(x) which interval to pursue
- Send updated parameter to all machines

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

update value in search direction and feed back

communicate final value to each machine

Scalability analysis

- Linear time in number of instances
- Linear storage in problem size, not data
- Logarithmic time in accuracy
- ‘perfect’ scalability

- 10s of passes through dataset for each iteration (line search is very expensive)
- MapReduce loses state at each iteration
- Single master as bottleneck (important if the state space is several GB)

A Better Algorithm

- Avoiding the line search
  - Not used in convergence proof anyway
  - Simply pick update

$$x \leftarrow x - \frac{1}{M} \partial_x f(x)$$

- Only single pass through data per iteration
- Only single MapReduce pass per iteration
- Logarithmic iteration bound (as before)

$$\frac{M}{m} \log \frac{f(x) - f(x^*)}{\epsilon}$$

Newton's Method

Isaac Newton

Newton Method

- Convex objective function f
- Nonnegative second derivative

$$\partial_x^2 f(x) \succeq 0$$

- Taylor expansion

$$f(x+\delta) = f(x) + \langle \delta, \underbrace{\partial_x f(x)}_{\text{gradient}} \rangle + \frac{1}{2}\delta^\top \underbrace{\partial_x^2 f(x)}_{\text{Hessian}}\delta + O(\delta^3)$$

- Minimize approximation & iterate til converged

$$x \leftarrow x - \left[\partial_x^2 f(x)\right]^{-1} \partial_x f(x)$$

Convergence Analysis

- There exists a region around optimality where Newton's method converges quadratically if f is twice continuously differentiable
- For some region around x* gradient is well approximated by Taylor expansion

$$\left\|\partial_x f(x^*) - \partial_x f(x) - \left\langle x^* - x, \partial_x^2 f(x) \right\rangle\right\| \leq \gamma \left\|x^* - x\right\|^2$$

- Expand Newton update

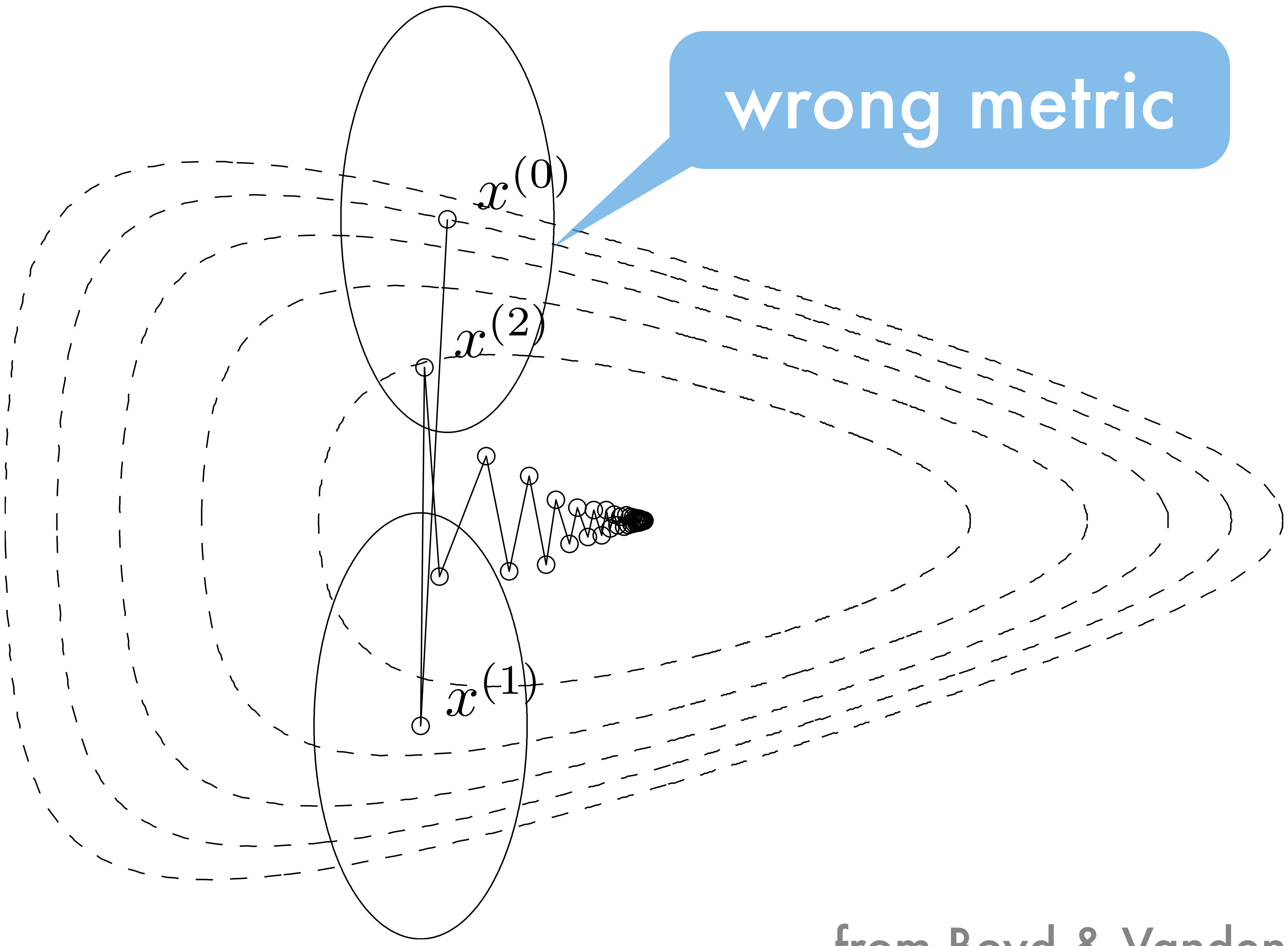
$$\begin{aligned}
\|x_{n+1} - x^*\| &= \left\| x_n - x^* - \left[\partial_x^2 f(x_n)\right]^{-1} \left[\partial_x f(x_n) - \partial_x f(x^*)\right] \right\| \\
&= \left\| \left[\partial_x^2 f(x_n)\right]^{-1} \left[\partial_x^f (x_n)[x_n - x^*] - \partial_x f(x_n) + \partial_x f(x^*)\right] \right\| \\
&\leq \gamma \left\| \left[\partial_x^2 f(x_n)\right]^{-1} \right\| \|x_n - x^*\|^2
\end{aligned}$$

Convergence Analysis

- Two convergence regimes
  - As slow as gradient descent outside the region where Taylor expansion is good
$$\left\|\partial_x f(x^*) - \partial_x f(x) - \left\langle x^* - x, \partial_x^2 f(x)\right\rangle\right\| \leq \gamma \left\|x^* - x\right\|^2$$
  - Quadratic convergence once the bound holds
$$\|x_{n+1} - x^*\| \leq \gamma \left\|\left[\partial_x^2 f(x_n)\right]^{-1}\right\| \|x_n - x^*\|^2$$
  - Newton method is affine invariant (proof by chain rule)

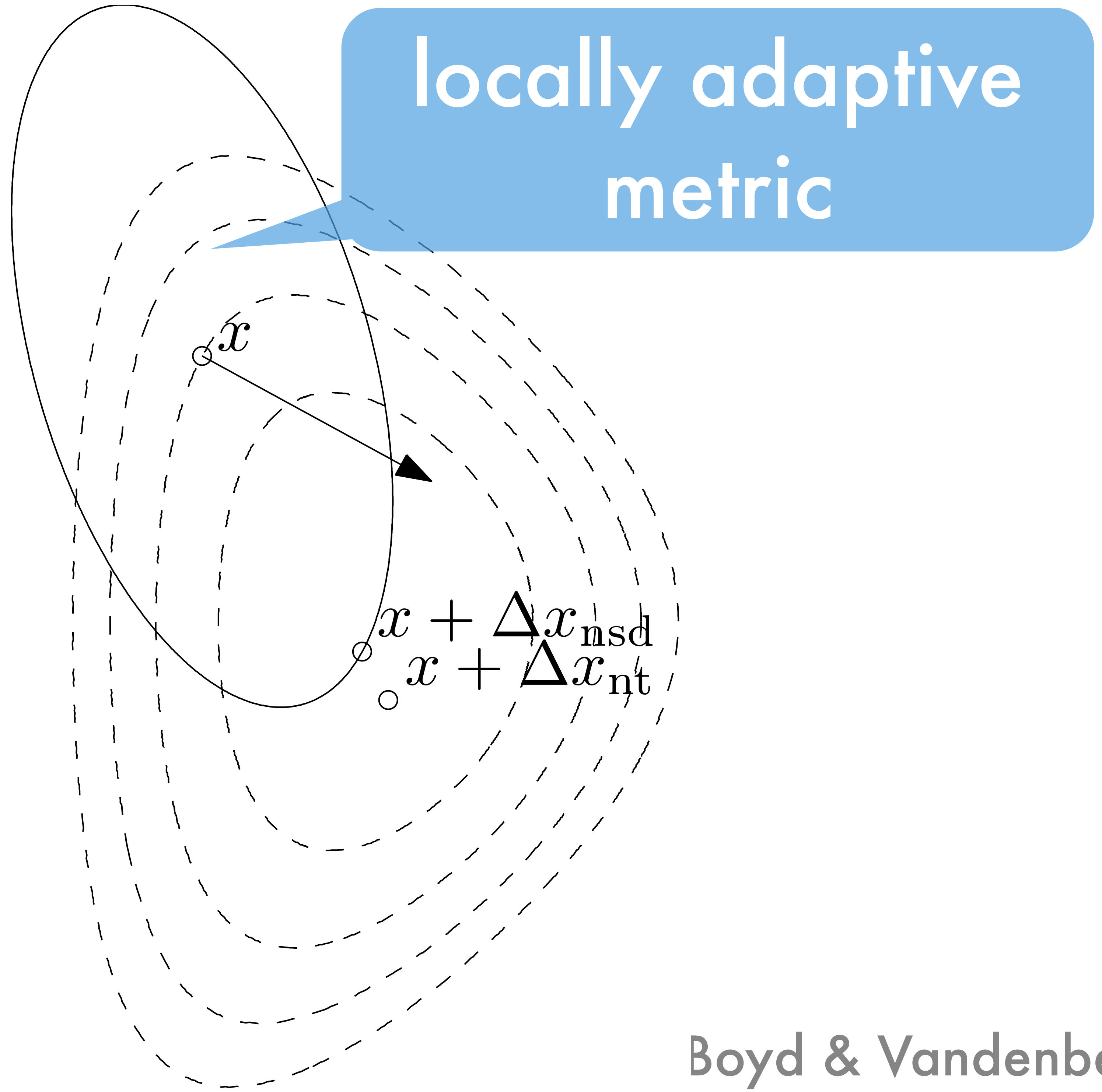
See Boyd and Vandenberghe, Chapter 9.5 for much more

Newton method rescales space



from Boyd & Vandenberghe

Newton method rescales space



Boyd & Vandenberghe

Parallel Newton Method

- Good rate of convergence
- Few passes through data needed
- Parallel aggregation of gradient and Hessian
- Gradient requires O(d) data
- Hessian requires $O(d^2)$ data
- Update step is $O(d^3)$ & nontrivial to parallelize
- Use it only for low dimensional problems

Conjugate Gradient Descent

Key Idea

- Minimizing quadratic function $(K \succeq 0)$
$$f(x) = \frac{1}{2}x^\top K x - l^\top x + c$$
takes cubic time (e.g. Cholesky factorization)
- Matrix vector products and orthogonalization
  - Vectors x, x' are K orthogonal if $x^\top K x' = 0$
  - m mutually K orthogonal vectors $x_i \in \mathbb{R}^m$
    - form a basis
    - allow expansion
    - solve linear system

$$z = \sum_{i=1}^{m} x_i \frac{x_i^\top K z}{x_i^\top K x_i}$$

$$z = \sum_{i=1}^{m} x_i \frac{x_i^\top y}{x_i^\top K x_i} \text{ for } y = Kz$$

Proof

- m mutually K orthogonal vectors $x_i \in \mathbb{R}^m$
  - form a basis
  - allow expansion
  - solve linear system

$$z = \sum_{i=1}^{m} x_i \frac{x_i^\top K z}{x_i^\top K x_i}$$

$$z = \sum_{i=1}^{m} x_i \frac{x_i^\top y}{x_i^\top K x_i} \text{ for } y = Kz$$

- Show linear independence by contradiction

$$\sum_i \alpha_i x_i = 0 \text{ hence } 0 = x_j^\top K \sum_i \alpha_i x_i = x_j^\top K x_j \alpha_j$$

- Reconstruction - expand z into basis

$$z = \sum_i \alpha_i x_i \text{ hence } x_j^\top K z = x_j^\top K \sum_i \alpha_i x_i = x_j^\top K x_j \alpha_j$$

- For linear system plug in y = Kz

???

- Need vectors x_i
- Need to orthogonalize the vectors
- How to select them

- K-orthogonal vectors whiten the space since

$$f(x) = \frac{1}{2}x^\top x - l^\top x + c$$

has trivial solution $x = l$

Conjugate Gradient Descent

- **Gradient computation**

$$f(x) = \frac{1}{2} x^\top K x - l^\top x + c \text{ hence } g(x) = Kx - l$$

- **Algorithm**

$$\begin{aligned}
&\textbf{initialize } x_0 \text{ and } v_0 = g_0 = Kx_0 - l \text{ and } i = 0 \\
&\textbf{repeat} \\
&\quad x_{i+1} = x_i - v_i \frac{g_i^\top v_i}{v_i^\top K v_i} \\
&\quad g_{i+1} = K x_{i+1} - l \\
&\quad v_{i+1} = -g_{i+1} + v_i \frac{g_{i+1}^\top K v_i}{v_i^\top K v_i} \\
&\quad i \leftarrow i + 1 \\
&\textbf{until } g_i = 0
\end{aligned}$$

deflation step

K orthogonal

Proof - Deflation property

$$x_{i+1} = x_i - v_i \frac{g_i^\top v_i}{v_i^\top K v_i}$$
$$g_{i+1} = K x_{i+1} - l$$
$$v_{i+1} = -g_{i+1} + v_i \frac{g_{i+1}^\top K v_i}{v_i^\top K v_i}$$

- First assume that the v_i are K orthogonal and show that x_{i+1} is optimal in span of $\{v_1 .. v_i\}$
- Enough if we show that $v_j^\top g_i = 0 \text{ for all } j < i$
  - For j=i expand $$v_i^\top g_{i+1} = v_i^\top \left[K x_i - l - K v_i \frac{g_i^\top v_i}{v_i^\top K v_i} \right]$$ $$= v_i^\top g_i - v_i^\top K v_i \frac{g_i^\top v_i}{v_i^\top K v_i} = 0$$
  - For smaller j a consequence of K orthogonality

Proof - K orthogonality

$$x_{i+1} = x_i - v_i \frac{g_i^\top v_i}{v_i^\top K v_i}$$
$$g_{i+1} = K x_{i+1} - l$$
$$v_{i+1} = -g_{i+1} + v_i \frac{g_{i+1}^\top K v_i}{v_i^\top K v_i}$$

- Need to check that v_{i+1} is K orthogonal to all v_i (rest automatically true by construction)

$$v_j^\top K v_{i+1} = -v_j^\top K g_{i+1} + v_j^\top K v_i \frac{g_{i+1}^\top K v_i}{v_i^\top K V_i}$$

0 by deflation

0 by K orthogonality

Properties

- Subspace expansion method for optimality (g, Kg, K^2g, K^3g, ...)
- Focuses on leading eigenvalues
- Often sufficient to take only a few steps (whenever the eigenvalues decay rapidly)

Extensions

Generic Method	Compute Hessian $K_i := f''(x_i)$ and update α_i, β_i with $\alpha_i = -\frac{g_i^\top v_i}{v_i^\top K_i v_i}$ $\beta_i = \frac{g_{i+1}^\top K_i v_i}{v_i^\top K_i v_i}$ This requires calculation of the Hessian at each iteration.
Fletcher–Reeves [163]	Find α_i via a line search and use Theorem 6.20 (iii) for β_i $\alpha_i = \mathrm{argmin}_\alpha f(x_i + \alpha v_i)$ $\beta_i = \frac{g_{i+1}^\top g_{i+1}}{g_i^\top g_i}$
Polak–Ribiere [398]	Find α_i via a line search $\alpha_i = \mathrm{argmin}_\alpha f(x_i + \alpha v_i)$ $\beta_i = \frac{(g_{i+1} - g_i)^\top g_{i+1}}{g_i^\top g_i}$ Experimentally, Polak–Ribiere tends to be better than Fletcher–Reeves.

x and v updates

BFGS algorithm
Broyden-Fletcher-Goldfarb-Shanno

Basic Idea

- Newton-like method to compute descent direction

$$\delta_i = B_i^{-1} \partial_x f(x_{i-1})$$

- Line search on f in direction

$$x_{i+1} = x_i - \alpha_i \delta_i$$

- Update B with rank 2 matrix

$$B_{i+1} = B_i + u_i u_i^\top + v_i v_i^\top$$

- Require that Quasi-Newton condition holds

$$B_{i+1}(x_{i+1} - x_i) = \partial_x f(x_{i+1}) - \partial_x f(x_i)$$

$$B_{i+1} = B_i + \frac{g_i g_i^\top}{\alpha_i \delta_i^\top g_i} - \frac{B_i \delta_i \delta_i^\top B_i}{\delta_i^\top B_i \delta_i}$$

Properties

- Simple rank 2 update for B
- Use matrix inversion lemma to update inverse
- Memory-limited versions L-BFGS
- Use toolbox if possible (TAO, MATLAB)
(typically slower if you implement it yourself)
- Works well for nonlinear nonconvex objectives
(often even for nonsmooth objectives)

4.2 Constrained Convex Problems

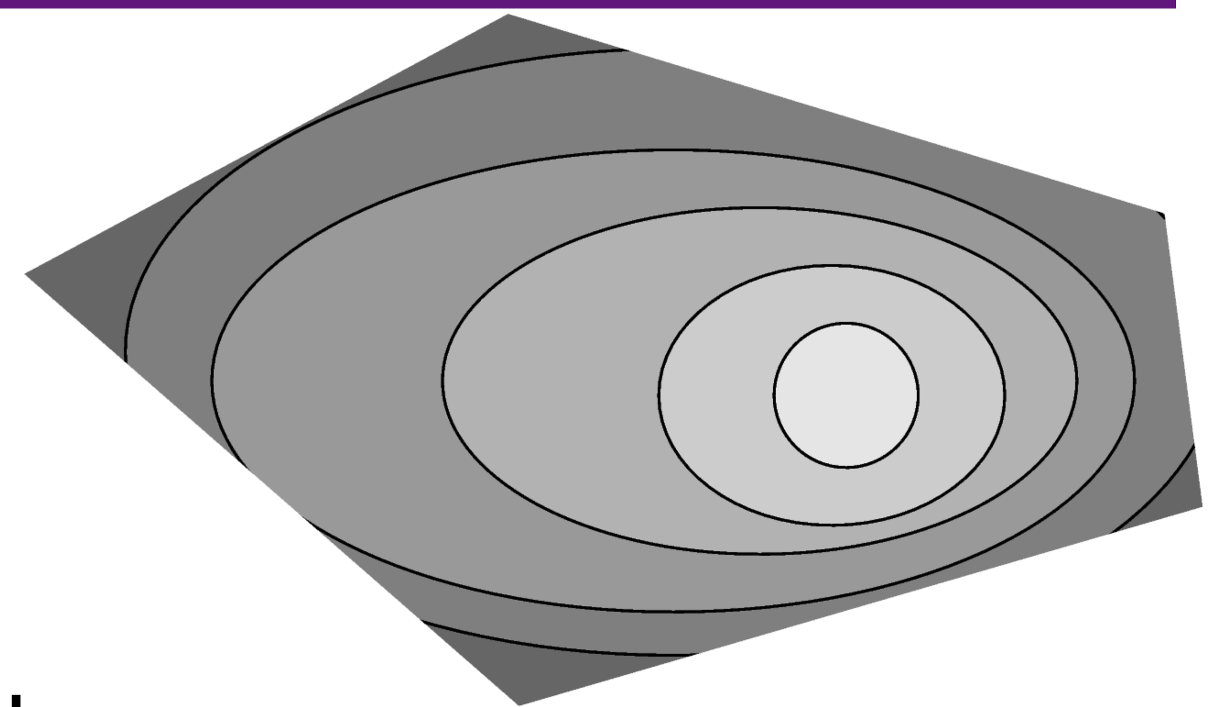
Basic Convexity

Constrained Convex Minimization

- Optimization problem

$$\underset{x}{\text{minimize}}\ f(x)$$

$$\text{subject to } c_i(x) \leq 0 \text{ for all } i$$

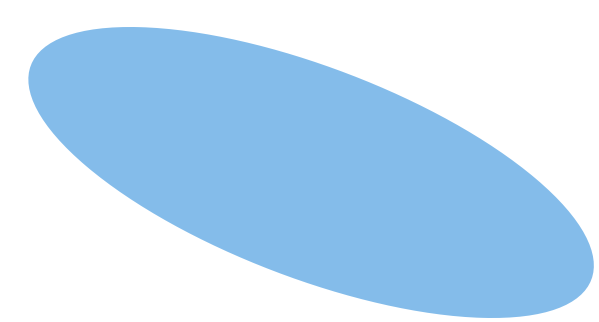


- Common constraints
  - linear inequality constraints

$$\langle w_i, x \rangle + b_i \leq 0$$

  - quadratic cone constraints

$$x^\top Q x + b^\top x \leq c \text{ with } Q \succeq 0$$



  - semidefinite constraints

$$M \succeq 0 \text{ or } M_0 + \sum_i x_i M_i \succeq 0$$

Constrained Convex Minimization

- Optimization problem

$$\underset{x}{\text{minimize}}\ f(x)$$

$$\text{subject to } c_i(x) \leq 0 \text{ for}$$

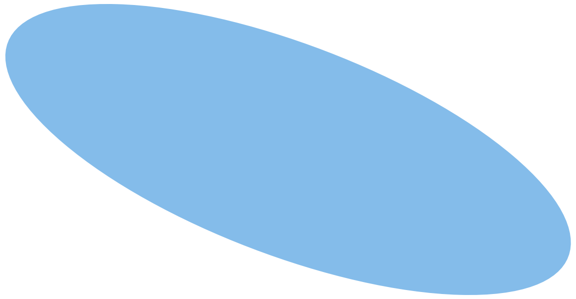
Equality is special case
Why?

- Common constraints
  - linear inequality constraints

$$\langle w_i, x \rangle + b_i \leq 0$$

  - quadratic cone constraints

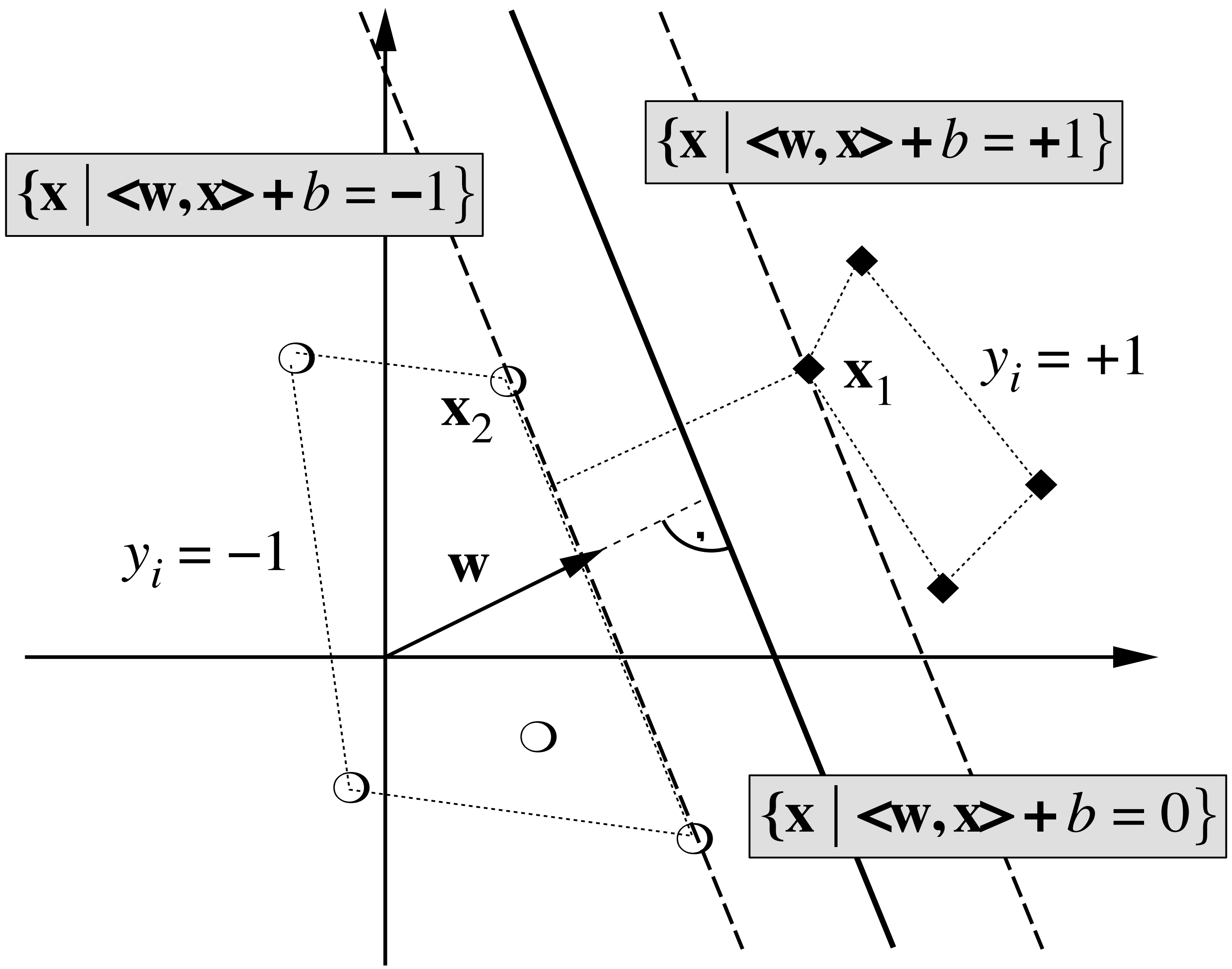
$$x^\top Q x + b^\top x \leq c \text{ with } Q \succeq 0$$



  - semidefinite constraints

$$M \succeq 0 \text{ or } M_0 + \sum_i x_i M_i \succeq 0$$

Example - Support Vectors



$$\langle w, x_1 \rangle + b = 1$$

$$\langle w, x_2 \rangle + b = -1$$

$$\text{hence} \ \ \langle w, x_1 - x_2 \rangle + b = 2$$

$$\text{hence} \ \left\langle \frac{w}{\|w\|}, x_1 - x_2 \right\rangle = \frac{2}{\|w\|}$$

margin

$$\underset{w,b}{\text{minimize}} \frac{1}{2} \|w\|^2 \ \text{subject to} \ y_i \left[\langle w, x_i \rangle + b\right] \geq 1$$

Lagrange Multipliers

- Lagrange function

$$L(x, \alpha) := f(x) + \sum_{i=1}^{n} \alpha_i c_i(x) \text{ where } \alpha_i \geq 0$$

- Saddlepoint Condition
  If there are x* and nonnegative α* such that

$$L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*)$$

  then x* is an optimal solution to the constrained optimization problem

Proof

$$L(x^*, \alpha) \leq L(x^*, \alpha^*) \leq L(x, \alpha^*)$$

- From first inequality we see that x* is feasible

$$(\alpha_i - \alpha_i^*) c_i(x^*) \leq 0 \text{ for all } \alpha_i \geq 0$$

- Setting some $\alpha_i = 0$ yields KKT conditions

$$\alpha_i^* c_i(x^*) = 0$$

- Consequently we have

$$L(x^*, \alpha^*) = f(x^*) \leq L(x, \alpha^*) = f(x) + \sum_i \alpha_i^* c_i(x) \leq f(x)$$

This proves optimality

Constraint gymnastics (all three conditions are equivalent)

- Slater's condition
  There exists some x such that for all i
  $$c_i(x) < 0$$
- Karlin's condition
  For all nonnegative α there exists some x such that
  $$\sum_i \alpha_i c_i(x) \leq 0$$
- Strict constraint qualification
  The feasible region contains at least two distinct elements and there exists an x in X such that all $c_i(x)$ are strictly convex at x with respect to X

Necessary Kuhn-Tucker Conditions

- Assume optimization problem
  - satisfies the constraint qualifications
  - has convex differentiable objective + constraints
- Then the KKT conditions are necessary & sufficient

$$\partial_x L(x^*, \alpha^*) = \partial_x f(x^*) + \sum_i \alpha_i^* \partial_x c_i(x^*) \qquad = 0 \text{ (Saddlepoint in } x^*)$$

$$\partial_{\alpha_i} L(x^*, \alpha^*) = c_i(x^*) \qquad \leq 0 \text{ (Saddlepoint in } \alpha^*)$$

$$\sum_i \alpha_i^* c_i(x^*) \qquad = 0 \text{ (Vanishing KKT-gap)}$$

Yields algorithm for solving optimization problems
Solve for saddlepoint and KKT conditions

Proof

$$
\begin{aligned}
f(x) - f(x^*) &\geq \left[\partial_x f(x^*)\right]^\top (x - x^*) && \text{(by convexity)} \\
&= -\sum_i \alpha_i^* \left[\partial_x c_i(x^*)\right]^\top (x - x^*) && \text{(by Saddlepoint in } x^*\text{)} \\
&\geq -\sum_i \alpha_i^* (c_i(x) - c_i(x^*)) && \text{(by convexity)} \\
&= \sum_i \alpha_i^* c_i(x) && \text{(by vanishing KKT gap)} \\
&\geq 0
\end{aligned}
$$

Linear and Quadratic Programs

Linear Programs

- **Objective**

$$\underset{x}{\text{minimize}}\, c^\top x \text{ subject to } Ax + d \leq 0$$

- **Lagrange function**

$$L(x, \alpha) = c^\top x + \alpha^\top (Ax + d)$$

- **Optimality conditions**

$$\begin{aligned} \partial_x L(x, \alpha) &= A^\top \alpha + c = 0 \\ \partial_\alpha L(x, \alpha) &= Ax + d \leq 0 \\ 0 &= \alpha^\top (Ax + d) \\ 0 &\leq \alpha \end{aligned}$$

- **Dual problem**

$$\underset{i}{\text{maximize}}\, d^\top \alpha \text{ subject to } A^\top \alpha + c = 0 \text{ and } \alpha \geq 0$$

Linear Programs

- **Objective**

$$\underset{x}{\text{minimize}}\, c^\top x \text{ subject to } Ax + d \le 0$$

- **Lagrange function**

$$L(x, \alpha) = c^\top x + \alpha^\top (Ax + d)$$

- **Optimality conditions**

$$\partial_x L(x, \alpha) = A^\top \alpha + c = 0$$
$$\partial_\alpha L(x, \alpha) = Ax + d \le 0$$
$$0 = \alpha^\top (Ax + d)$$
$$0 \le \alpha$$

plug into L

- **Dual problem**

$$\underset{i}{\text{maximize}}\, d^\top \alpha \text{ subject to } A^\top \alpha + c = 0 \text{ and } \alpha \ge 0$$

Linear Programs

- **Objective**

$$\underset{x}{\text{minimize}}\, c^\top x \text{ subject to } Ax + d \leq 0$$

- **Lagrange function**

$$L(x, \alpha) = c^\top x + \alpha^\top (Ax + d)$$

- **Optimality conditions**

$$\partial_x L(x, \alpha) = A^\top \alpha + c = 0$$
$$\partial_\alpha L(x, \alpha) = Ax + d \leq 0$$
$$0 = \alpha^\top (Ax + d)$$
$$0 \leq \alpha$$

plug into L

- **Dual problem**

$$\underset{i}{\text{maximize}}\, d^\top \alpha \text{ subject to } A^\top \alpha + c = 0 \text{ and } \alpha \geq 0$$

Linear Programs

- Primal

$$\underset{x}{\text{minimize}}\, c^\top x \text{ subject to } Ax + d \leq 0$$

- Dual

$$\underset{i}{\text{maximize}}\, d^\top \alpha \text{ subject to } A^\top \alpha + c = 0 \text{ and } \alpha \geq 0$$

- Free variables become equality constraints
- Equality constraints become free variables
- Inequalities become inequalities
- Dual of dual is primal

Quadratic Programs

- **Objective**

$$\underset{x}{\text{minimize}}\ \frac{1}{2}x^\top Q x + c^\top x \text{ subject to } Ax + d \leq 0$$

- **Lagrange function**

$$L(x, \alpha) = \frac{1}{2}x^\top Q x + c^\top x + \alpha^\top (Ax + d)$$

- **Optimality conditions**

$$\begin{aligned}
\partial_x L(x, \alpha) &= Qx + A^\top \alpha + c = 0 \\
\partial_\alpha L(x, \alpha) &= Ax + d \leq 0 \\
0 &= \alpha^\top (Ax + d) \\
0 &\leq \alpha
\end{aligned}$$

plug into L

Quadratic Program

- Eliminating x from the Lagrangian via

$$Qx + A^\top \alpha + c = 0$$

- Lagrange function

$$\begin{aligned}
L(x,\alpha) &= \frac{1}{2}x^\top Q x + c^\top x + \alpha^\top (Ax + d) \\
&= -\frac{1}{2}x^\top Q x + \alpha^\top d \\
&= -\frac{1}{2}(A^\top \alpha + c)^\top Q^{-1}(A^\top \alpha + c) + \alpha^\top d \\
&= -\frac{1}{2}\alpha^\top A Q^{-1} A^\top \alpha + \alpha^\top \left[d - AQ^{-1}c\right] - \frac{1}{2}c^\top Q^{-1} c
\end{aligned}$$

$$\text{subject to } \alpha \geq 0$$

Quadratic Program

- Eliminating x from the Lagrangian via

$$Qx + A^\top \alpha + c = 0$$

- Lagrange function

$$\begin{aligned}
L(x, \alpha) &= \frac{1}{2} x^\top Q x + c^\top x + \alpha^\top (Ax + d) \\
&= -\frac{1}{2} x^\top Q x + \alpha^\top d \\
&= -\frac{1}{2} (A^\top \alpha + c)^\top Q^{-1} (A^\top \alpha + c) + \alpha^\top d \\
&= -\frac{1}{2} \alpha^\top A Q^{-1} A^\top \alpha + \alpha^\top \left[d - A Q^{-1} c \right] - \frac{1}{2} c^\top Q^{-1} c
\end{aligned}$$

$$\text{subject to } \alpha \geq 0$$

dual

Quadratic Programs

- Primal

$$\underset{x}{\text{minimize}}\ \frac{1}{2}x^\top Qx + c^\top x \text{ subject to } Ax + d \leq 0$$

- Dual

$$\underset{\alpha}{\text{minimize}}\ \frac{1}{2}\alpha^\top AQ^{-1}A^\top\alpha + \alpha^\top\left[AQ^{-1}c - d\right] \text{ subject to } \alpha \geq 0$$

  - Dual constraints are simpler
  - Possibly many fewer variables
  - Dual of dual is not (always) primal (e.g. in SVMs x is in a Hilbert Space)

Interior Point Solvers

Constrained Newton Method

- **Objective** $\underset{x}{\text{minimize}}\, f(x) \text{ subject to } Ax = b$
- **Lagrange function and optimality conditions**

$$L(x,\alpha) = f(x) + \alpha^\top [Ax - b]$$

$$\partial_x L(x,\alpha) = \partial_x f(x) + A^\top \alpha = 0$$

$$\partial_\alpha L(x,\alpha) = Ax - b = 0$$

yields optimality

- **Taylor expansion of gradient**

$$\partial_x f(x) = \partial_x f(x_0) + \partial_x^2 f(x_0)[x - x_0] + O(\|x - x_0\|^2)$$

- **Plug back into the constraints and solve**

$$\begin{bmatrix} \partial_x^2 f(x_0) & A^\top \\ A & \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} = \begin{bmatrix} \partial_x^2 f(x_0) x_0 - \partial_x f(x_0) \\ b \end{bmatrix}$$

No need to be initially feasible!

General Strategy

- Optimality conditions

$$\partial_x L(x^*, \alpha^*) = \partial_x f(x^*) + \sum_i \alpha_i^* \partial_x c_i(x^*) = 0 \text{ (Saddlepoint in } x^*)$$

$$\partial_{\alpha_i} L(x^*, \alpha^*) = c_i(x^*) \leq 0 \text{ (Saddlepoint in } \alpha^*)$$

$$\sum_i \alpha_i^* c_i(x^*) = 0 \text{ (Vanishing KKT-gap)}$$

- Solve equations repeatedly.
- Yields primal and dual solution variables
- Yields size of primal/dual gap
- Feasibility not necessary at start
- KKT conditions are problematic - need approximation

Quadratic Programs

- Optimality conditions

$$\begin{aligned} Qx + A^\top \alpha + c &= 0 \\ Ax + d + \xi &= 0 \\ \alpha_i \xi_i &= 0 \\ \alpha, \xi &\geq 0 \end{aligned}$$

slack

- Relax KKT conditions

$$\alpha_i \xi_i = 0 \text{ relaxed to } \alpha_i \xi_i = \mu$$

- Solve linearization of nonlinear system

$$\begin{bmatrix} Q & A^\top \\ A & -D \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \alpha \end{bmatrix} = \begin{bmatrix} c_x \\ c_\alpha \end{bmatrix}$$

- Predictor/corrector step for nonlinearity
- Iterate until converged

Implementation details

- Dominant cost is solving reduced KKT system

$$\begin{bmatrix} Q & A^\top \\ A & -D \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \alpha \end{bmatrix} = \begin{bmatrix} c_x \\ c_\alpha \end{bmatrix}$$

  Solve linear system with (dense) Q and A
- Solve linear system twice (predictor / corrector)
- Update steps are only taken far enough to ensure nonnegativity of dual and slack
- Tighten up KKT constraints by decreasing μ
- Only 10-20 iterations typically needed

Solver Software

- OOQP
  http://pages.cs.wisc.edu/~swright/ooqp/
  Object oriented quadratic programming solver
- LOQO
  http://www.princeton.edu/~rvdb/loqo/LOQO.html
  Interior point path following solver
- HOPDM
  http://www.maths.ed.ac.uk/~gondzio/software/hopdm.html
  Linear and nonlinear infeasible IP solver
- CVXOPT
  http://abel.ee.ucla.edu/cvxopt/
  Python package for convex optimization
- SeDuMi
  http://sedumi.ie.lehigh.edu/
  Semidefinite programming solver

Solver Software

- OOQP
  http://pages.cs.wisc.edu/~swright/ooqp/
  Object oriented quadratic programming solver
- LOQO
  http://www.princeton.edu/~rvdb/loqo/LOQO.html
  Interior point path following solver
- HOPDM
  http://www.maths.ed.ac.uk/~gondzio/software/hopdm.html
  Linear and nonlinear infeasible IP solver
- CVXOPT
  http://abel.ee.ucla.edu/cvxopt/
  Python package for convex optimization
- SeDuMi
  http://sedumi.ie.lehigh.edu/
  Semidefinite programming solver

nontrivial to parallelize

Bundle Methods

simple parallelization

Some optimization problems

- **Density estimation**

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} -\log p(x_i|\theta) - \log p(\theta)$$

$$\text{equivalently} \quad \underset{\theta}{\text{minimize}} \sum_{i=1}^{m} \left[g(\theta) - \langle \phi(x_i), \theta \rangle\right] + \frac{1}{2\sigma^2} \|\theta\|^2$$

- **Penalized regression**

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} l\left(y_i - \langle \phi(x_i), \theta \rangle\right) + \frac{1}{2\sigma^2} \|\theta\|^2$$

e.g. squared loss

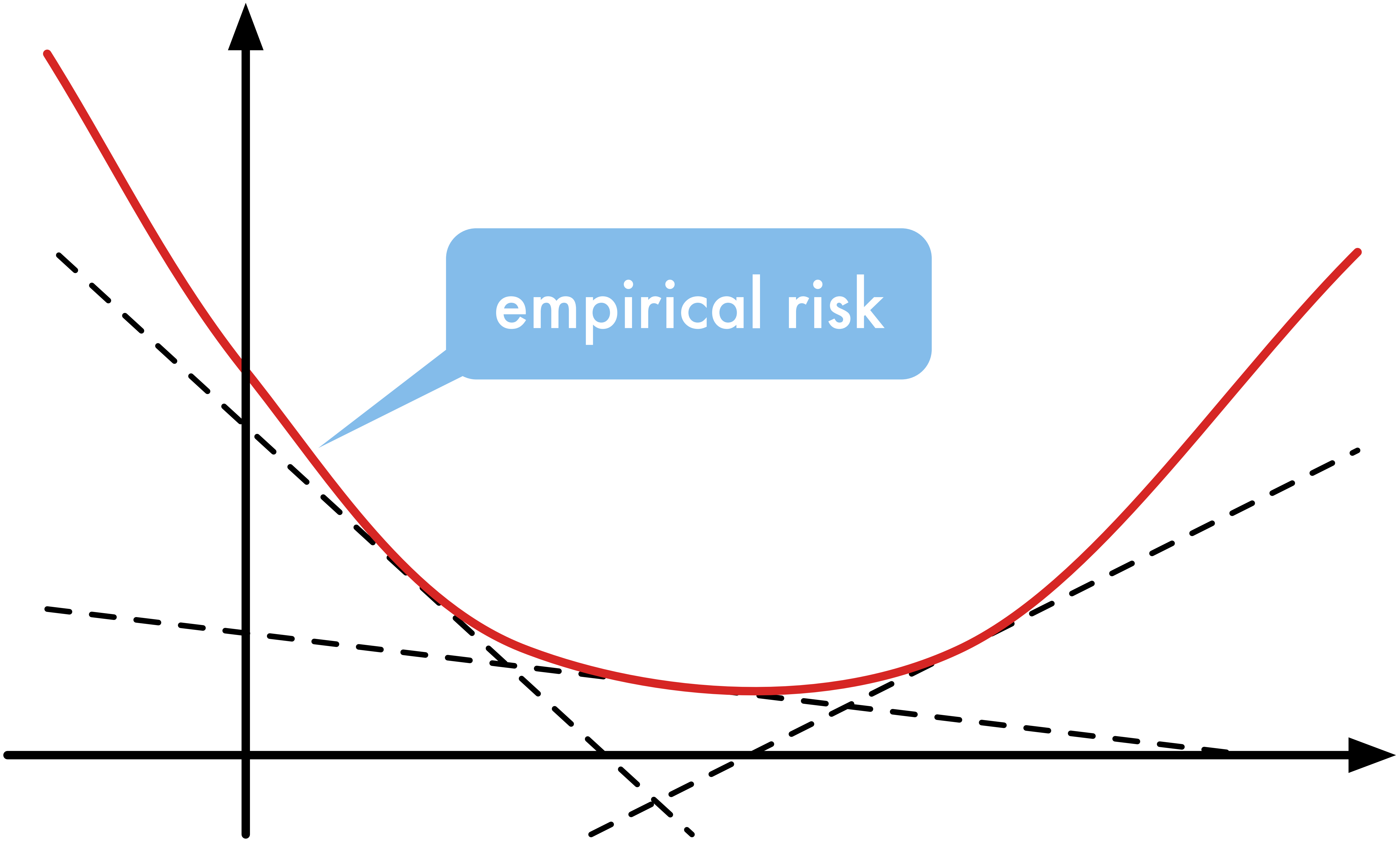
regularizer

Basic Idea

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{m} l_i(\theta) + \lambda\Omega[\theta]$$

- Loss
  - Convex but expensive to compute
  - Line search just as expensive as new computation
  - Gradient almost free with function value computation
  - Easy to compute in parallel
- Regularizer
  - Convex and cheap to compute and to optimize
- Strategy
  - Compute tangents on loss
  - Provides lower bound on objective
  - Solve dual optimization problem (fewer parameters)

Bundle Method



Lower bound

Regularized Risk Minimization

$$\underset{w}{\text{minimize}}\, R_{\mathrm{emp}}[w] + \lambda\Omega[w]$$

Taylor Approximation for $R_{\mathrm{emp}}[w]$

$$R_{\mathrm{emp}}[w] \geq R_{\mathrm{emp}}[w_t] + \langle w - w_t, \partial_w R_{\mathrm{emp}}[w_t]\rangle = \langle a_t, w\rangle + b_t$$

where $a_t = \partial_w R_{\mathrm{emp}}[w_{t-1}]$ and $b_t = R_{\mathrm{emp}}[w_{t-1}] - \langle a_t, w_{t-1}\rangle$.

Bundle Bound

$$R_{\mathrm{emp}}[w] \geq R_t[w] := \max_{i \leq t} \langle a_i, w\rangle + b_i$$

Regularizer $\Omega[w]$ solves stability problems.

Pseudocode

Initialize $t = 0, w_0 = 0, a_0 = 0, b_0 = 0$
repeat
  Find minimizer

$$w_t := \operatorname*{argmin}_w R_t(w) + \lambda\Omega[w]$$

  Compute gradient a_{t+1} and offset b_{t+1}.
  Increment $t \leftarrow t + 1$.
until $\epsilon_t \leq \epsilon$

Convergence Monitor $R_{t+1}[w_t] - R_t[w_t]$

Since $R_{t+1}[w_t] = R_{\mathrm{emp}}[w_t]$ (Taylor approximation) we have

$$R_{t+1}[w_t] + \lambda\Omega[w_t] \geq \min_w R_{\mathrm{emp}}[w] + \lambda\Omega[w] \geq R_t[w_t] + \lambda\Omega[w_t]$$

Dual Problem

Good News

Dual optimization for $\Omega[w] = \frac{1}{2}\|w\|_2^2$ is Quadratic Program regardless of the choice of the empirical risk $R_{\mathrm{emp}}[w]$.

Details

$$\underset{\beta}{\text{minimize}}\ \frac{1}{2\lambda}\beta^\top A A^\top \beta - \beta^\top b$$

$$\text{subject to } \beta_i \geq 0 \text{ and } \|\beta\|_1 = 1$$

The primal coefficient w is given by $w = -\lambda^{-1} A^\top \beta$.

General Result

Use Fenchel-Legendre dual of $\Omega[w]$, e.g. $\|\cdot\|_1 \to \|\cdot\|_\infty$.

Very Cheap Variant

Can even use simple line search for update (almost as good).

Properties

Parallelization

- Empirical risk sum of many terms: MapReduce
- Gradient sum of many terms, gather from cluster.
- Possible even for multivariate performance scores.
- Data is **local**. Combine data from competing entities.

Solver independent of loss

No need to change solver for new loss.

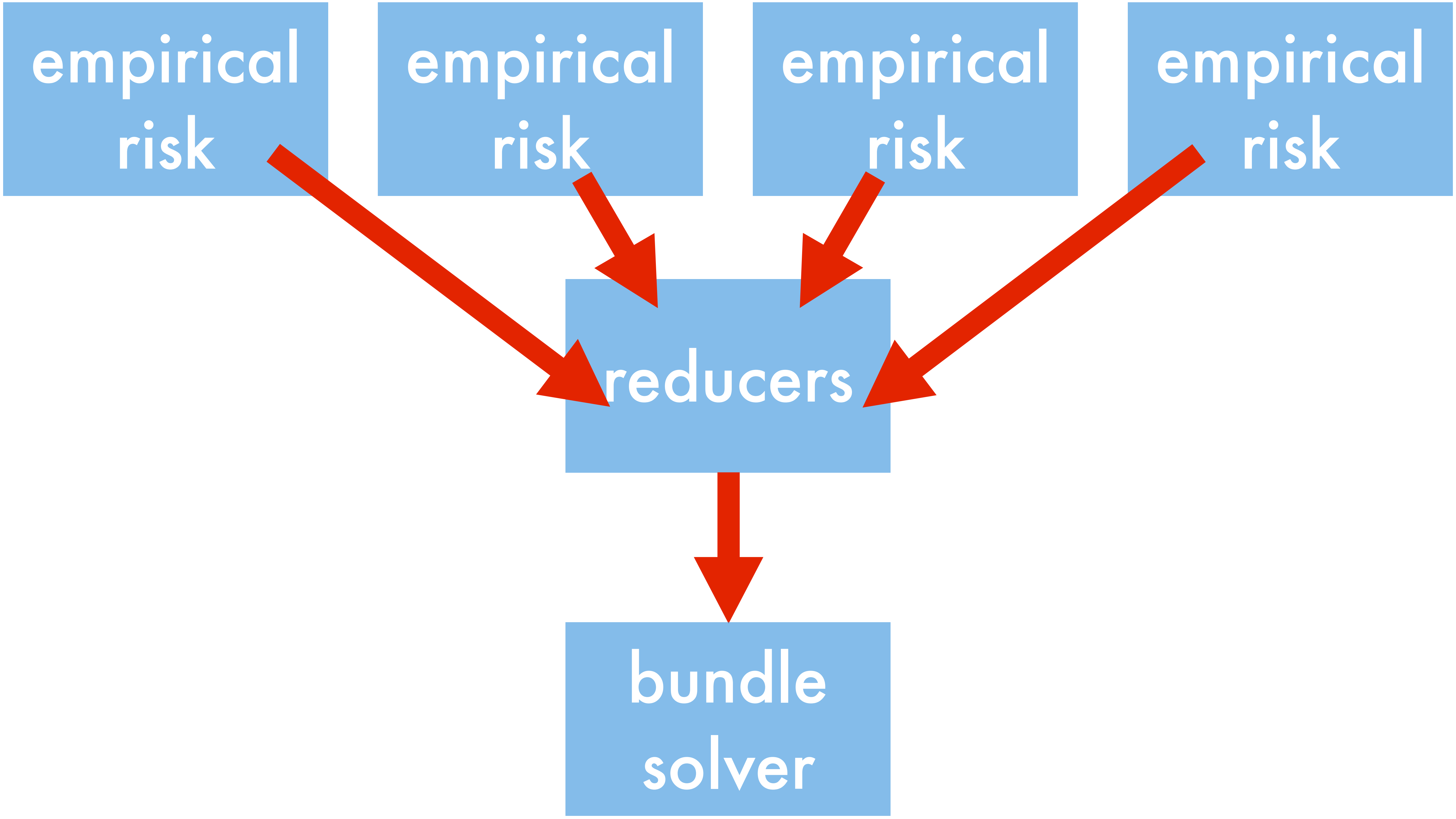
Loss independent of solver/regularizer

Add new regularizer without need to re-implement loss.

Line search variant

- Optimization does not require QP solver at all!
- Update along gradient direction in the dual.
- We only need inner product on gradients!

Implementation



Guarantees

Theorem

The number of iterations to reach ϵ precision is bounded by

$$n \leq \log_2 \frac{\lambda R_{\text{emp}}[0]}{G^2} + \frac{8G^2}{\lambda\epsilon} - 4$$

steps. If the Hessian of $R_{\text{emp}}[w]$ is bounded, convergence to any $\epsilon \leq \lambda/2$ takes at most the following number of steps:

$$n \leq \log_2 \frac{\lambda R_{\text{emp}}[0]}{4G^2} + \frac{4}{\lambda} \max\left[0, 1 - 8G^2 H^* / \lambda\right] - \frac{4H^*}{\lambda} \log 2\epsilon$$

Advantages

- Linear convergence for smooth loss
- For non-smooth loss almost as good in practice (as long as smooth on a course scale).
- Does not require primal line search.

Proof idea

Duality Argument

- Dual of $R_i[w] + \lambda\Omega[w]$ lower bounds minimum of regularized risk $R_{\mathrm{emp}}[w] + \lambda\Omega[w]$.
- $R_{i+1}[w_i] + \lambda\Omega[w_i]$ is upper bound.
- **Show that the gap $\gamma_i := R_{i+1}[w_i] - R_i[w_i]$ vanishes.**

Dual Improvement

- Give lower bound on increase in dual problem in terms of γ_i and the subgradient $\partial_w \left[R_{\mathrm{emp}}[w] + \lambda\Omega[w]\right]$.
- For unbounded Hessian we have $\delta\gamma = O(\gamma^2)$.
- For bounded Hessian we have $\delta\gamma = O(\gamma)$.

Convergence

- Solve difference equation in γ_t to get desired result.

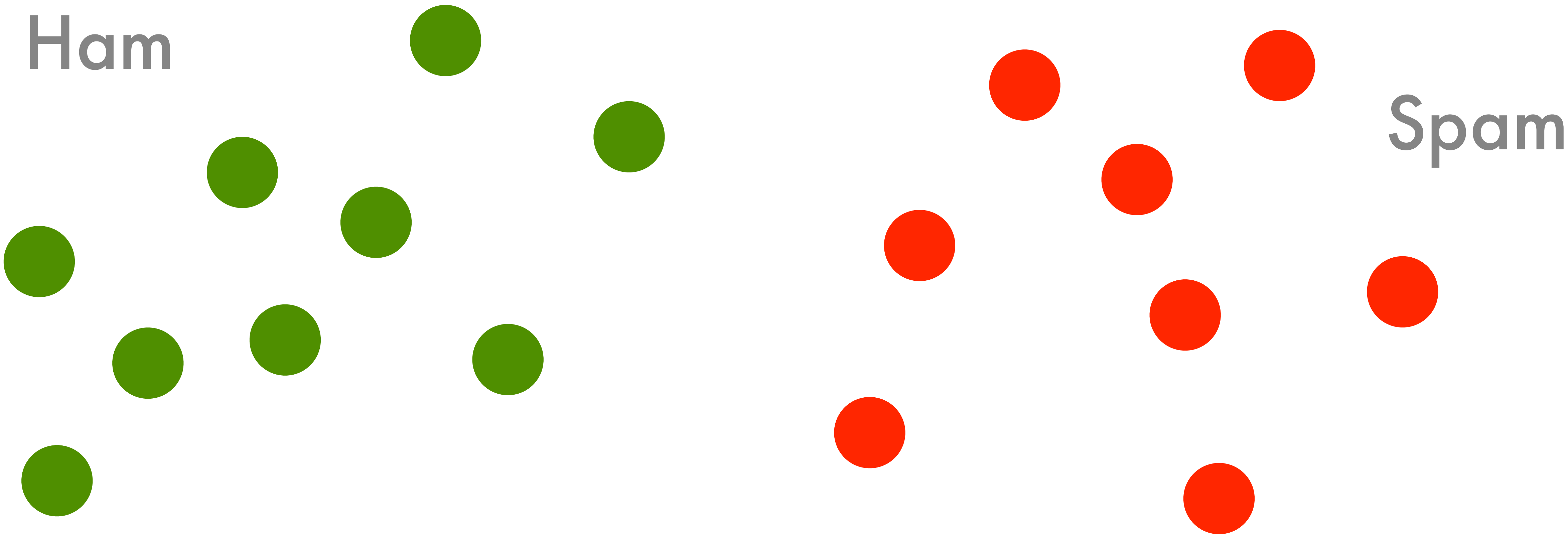
More

- Dual decomposition methods
  - Optimization problem with many constraints
  - Replicate variable & add equality constraints
  - Solve relaxed problem
  - Gradient descent in dual variables
- Prox operator
  - Problems with smooth & nonsmooth objective
  - Generalization of Bregman projections

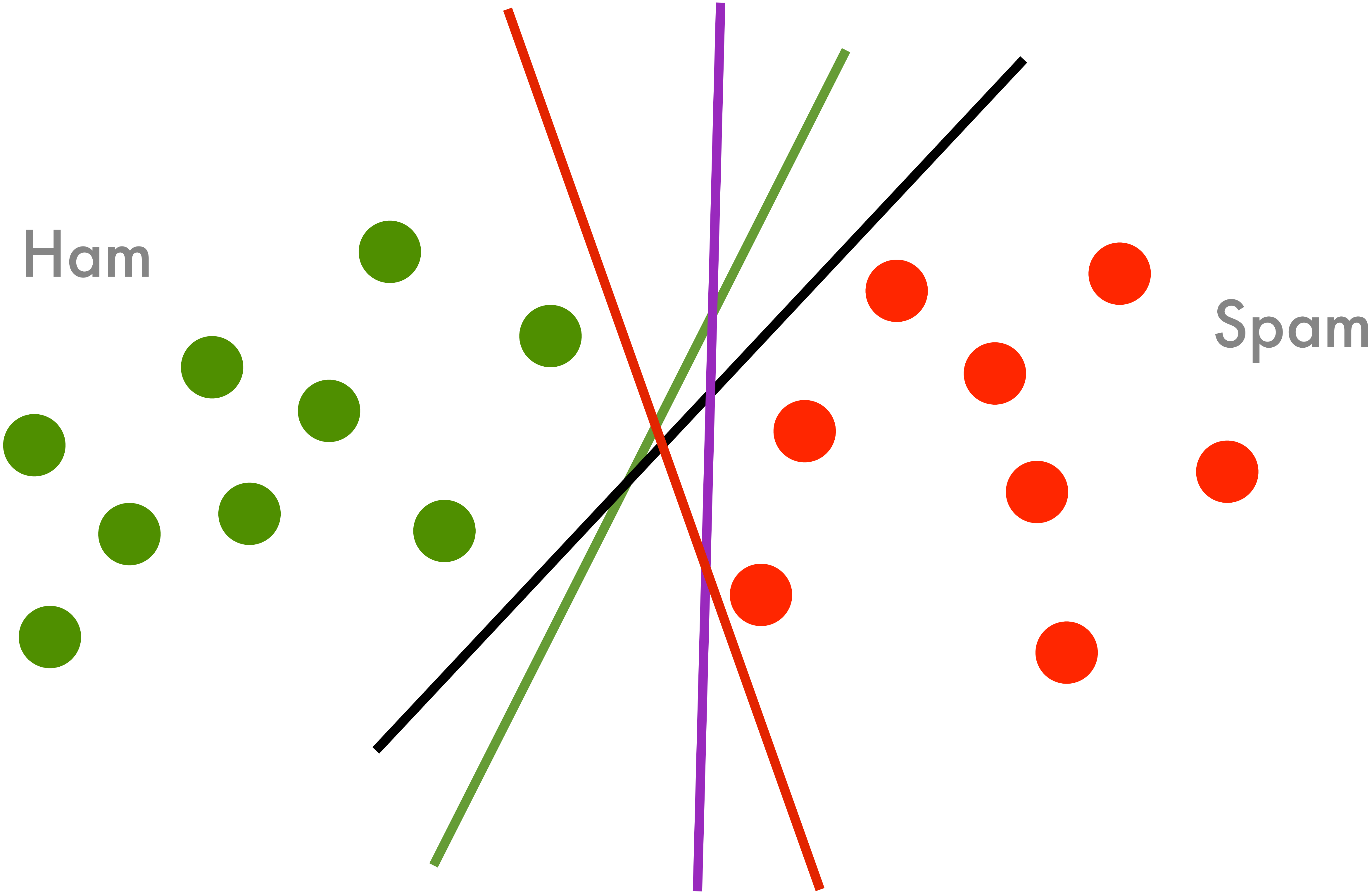
4.3 Online Methods

The Perceptron

The Perceptron



The Perceptron



The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i \left[\langle w, x_i \rangle + b\right] \leq 0$ **then**
    $w \leftarrow w + y_i x_i$ and $b \leftarrow b + y_i$
  end if
until all classified correctly

- Nothing happens if classified correctly
- Weight vector is linear combination $w = \sum_{i \in I} x_i$
- Classifier is linear combination of inner products $f(x) = \sum_{i \in I} \langle x_i, x \rangle + b$

Convergence Theorem

- If there exists some (w^*, b^*) with unit length and

$$y_i \left[\langle x_i, w^* \rangle + b^*\right] \geq \rho \text{ for all } i$$

then the perceptron converges to a linear separator after a number of steps bounded by

$$\left({b^*}^2 + 1\right) \left(r^2 + 1\right) \rho^{-2} \text{ where } \|x_i\| \leq r$$

- Dimensionality independent
- Order independent (i.e. also worst case)
- Scales with 'difficulty' of problem

Proof

Starting Point

We start from $w_1 = 0$ and $b_1 = 0$.

Step 1: Bound on the increase of alignment

Denote by w_i the value of w at step i (analogously b_i).

Alignment: $\langle (w_i, b_i), (w^*, b^*) \rangle$

For error in observation (x_i, y_i) we get

$$
\begin{aligned}
&\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle \\
&= \langle [(w_j, b_j) + y_i(x_i, 1)], (w^*, b^*) \rangle \\
&= \langle (w_j, b_j), (w^*, b^*) \rangle + y_i \langle (x_i, 1) \cdot (w^*, b^*) \rangle \\
&\geq \langle (w_j, b_j), (w^*, b^*) \rangle + \rho \\
&\geq j\rho.
\end{aligned}
$$

Alignment increases with number of errors.

Proof

Step 2: Cauchy-Schwartz for the Dot Product

$$\begin{aligned}\langle (w_{j+1}, b_{j+1}) \cdot (w^*, b^*) \rangle &\leq \|(w_{j+1}, b_{j+1})\| \, \|(w^*, b^*)\| \\ &= \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\|\end{aligned}$$

Step 3: Upper Bound on $\|(w_j, b_j)\|$

If we make a mistake we have

$$\begin{aligned}\|(w_{j+1}, b_{j+1})\|^2 &= \|(w_j, b_j) + y_i(x_i, 1)\|^2 \\ &= \|(w_j, b_j)\|^2 + 2y_i \langle (x_i, 1), (w_j, b_j) \rangle + \|(x_i, 1)\|^2 \\ &\leq \|(w_j, b_j)\|^2 + \|(x_i, 1)\|^2 \\ &\leq j(R^2 + 1).\end{aligned}$$

Step 4: Combination of first three steps

$$j\rho \leq \sqrt{1 + (b^*)^2} \|(w_{j+1}, b_{j+1})\| \leq \sqrt{j(R^2 + 1)((b^*)^2 + 1)}$$

Solving for j proves the theorem.

Consequences

- Only need to store errors.
  This gives a compression bound for perceptron.
- Stochastic gradient descent on hinge loss

$$l(x_i, y_i, w, b) = \max(0, 1 - y_i [\langle w, x_i \rangle + b])$$

- Fails with noisy data

do NOT train your avatar with perceptrons

Stochastic Gradient Descent

Stochastic gradient descent

- Empirical risk as expectation

$$\frac{1}{m}\sum_{i=1}^{m} l\left(y_i - \langle \phi(x_i), \theta \rangle\right) = \mathbf{E}_{i \sim \{1,..m\}}\left[l\left(y_i - \langle \phi(x_i), \theta \rangle\right)\right]$$

- Stochastic gradient descent (pick random x,y)

$$\theta_{t+1} \leftarrow \theta_t - \eta_t \partial_\theta \left(y_t, \langle \phi(x_t), \theta_t \rangle\right)$$

- Often we require that parameters are restricted to some convex set X, hence we project on it

$$\theta_{t+1} \leftarrow \pi_x \left[\theta_t - \eta_t \partial_\theta \left(y_t, \langle \phi(x_t), \theta_t \rangle\right)\right]$$

$$\text{here } \pi_X(\theta) = \operatorname*{argmin}_{x \in X} \|x - \theta\|$$

Convergence in Expectation

initial loss

$$\mathbf{E}_{\bar{\theta}}\left[l(\bar{\theta})\right] - l^* \leq \frac{R^2 + L^2 \sum_{t=0}^{T-1} \eta_t^2}{2 \sum_{t=0}^{T-1} \eta_t} \text{ where}$$

$$l(\theta) = \mathbf{E}_{(x,y)}\left[l(y, \langle \phi(x), \theta \rangle)\right] \text{ and } l^* = \inf_{\theta \in X} l(\theta) \text{ and } \bar{\theta} = \frac{\sum_{t=0}^{T-1} \theta_t \eta_t}{\sum_{t=0}^{T-1} \eta_t}$$

expected loss

parameter average

- Proof
  Show that parameters converge to minimum

$$\theta^* \in \operatorname*{argmin}_{\theta \in X} l(\theta) \text{ and set } r_t := \|\theta^* - \theta_t\|$$

from Nesterov and Vial

Proof

$$
\begin{aligned}
r_{t+1}^2 &= \left\|\pi_X[\theta_t - \eta_t g_t] - \theta^*\right\|^2 \\
&\leq \left\|\theta_t - \eta_t g_t - \theta^*\right\|^2 \\
&= r_t^2 + \eta_t^2 \left\|g_t\right\|^2 - 2\eta_t \left\langle \theta_t - \theta^*, g_t \right\rangle \\
\text{hence } \mathbf{E}\left[r_{t+1}^2 - r_t^2\right] &\leq \eta_t^2 L^2 + 2\eta_t \left[l^* - \mathbf{E}[l(\theta_t)]\right] \\
&\leq \eta_t^2 L^2 + 2\eta_t \left[l^* - \mathbf{E}[l(\bar{\theta})]\right]
\end{aligned}
$$

by convexity

- Summing over inequality for t proves claim
- This yields randomized algorithm for minimizing objective functions (try log times and pick the best / or average median trick)

Rates

- Guarantee

$$\mathbf{E}_{\bar{\theta}}\left[l(\bar{\theta})\right] - l^* \leq \frac{R^2 + L^2 \sum_{t=0}^{T-1} \eta_t^2}{2 \sum_{t=0}^{T-1} \eta_t}$$

- If we know R, L, T pick constant learning rate

$$\eta = \frac{R}{L\sqrt{T}} \text{ and hence } \mathbf{E}_{\bar{\theta}}[l(\bar{\theta})] - l^* \leq \frac{R[1 + 1/T]L}{2\sqrt{T}} < \frac{LR}{\sqrt{T}}$$

- If we don't know T pick $\eta_t = O(t^{-\frac{1}{2}})$
  This costs us an additional log term

$$\mathbf{E}_{\bar{\theta}}[l(\bar{\theta})] - l^* = O\left(\frac{\log T}{\sqrt{T}}\right)$$

Strong Convexity

$$l_i(\theta') \geq l_i(\theta) + \langle \partial_\theta l_i(\theta), \theta' - \theta \rangle + \frac{1}{2}\lambda \left\|\theta - \theta'\right\|^2$$

- **Use this to bound the expected deviation**

$$\begin{aligned} r_{t+1}^2 &\leq r_t^2 + \eta_t^2 \left\|g_t\right\|^2 - 2\eta_t \langle \theta_t - \theta^*, g_t \rangle \\ &\leq r_t^2 + \eta_t^2 L^2 - 2\eta_t \left[l_t(\theta_t) - l_t(\theta^*)\right] - 2\lambda\eta_t r_k^2 \end{aligned}$$

$$\text{hence } \mathbf{E}[r_{t+1}^2] \leq (1 - \lambda h_t)\mathbf{E}[r_t^2] - 2\eta_t \left[\mathbf{E}\left[l(\theta_t)\right] - l^*\right]$$

- **Exponentially decaying averaging**

$$\bar{\theta} = \frac{1-\sigma}{1-\sigma^T} \sum_{t=0}^{T-1} \sigma^{T-1-t}\theta_t$$

and plugging this into the discrepancy yields

$$l(\bar{\theta}) - l^* \leq \frac{2L^2}{\lambda T} \log\left[1 + \frac{\lambda R T^{\frac{1}{2}}}{2L}\right] \text{ for } \eta = \frac{2}{\lambda T} \log\left[1 + \frac{\lambda R T^{\frac{1}{2}}}{2L}\right]$$

More variants

- Adversarial guarantees

$$\theta_{t+1} \leftarrow \pi_x \left[\theta_t - \eta_t \partial_\theta \left(y_t, \langle \phi(x_t), \theta_t \rangle\right)\right]$$

  has low regret (average instantaneous cost) for arbitrary orders (useful for game theory)

  - Ratliff, Bagnell, Zinkevich
    $O(t^{-\frac{1}{2}})$ learning rate
  - Shalev-Shwartz, Srebro, Singer (Pegasos)
    $O(t^{-1})$ learning rate (but need constants)
  - Bartlett, Rakhlin, Hazan
    (add strong convexity penalty)

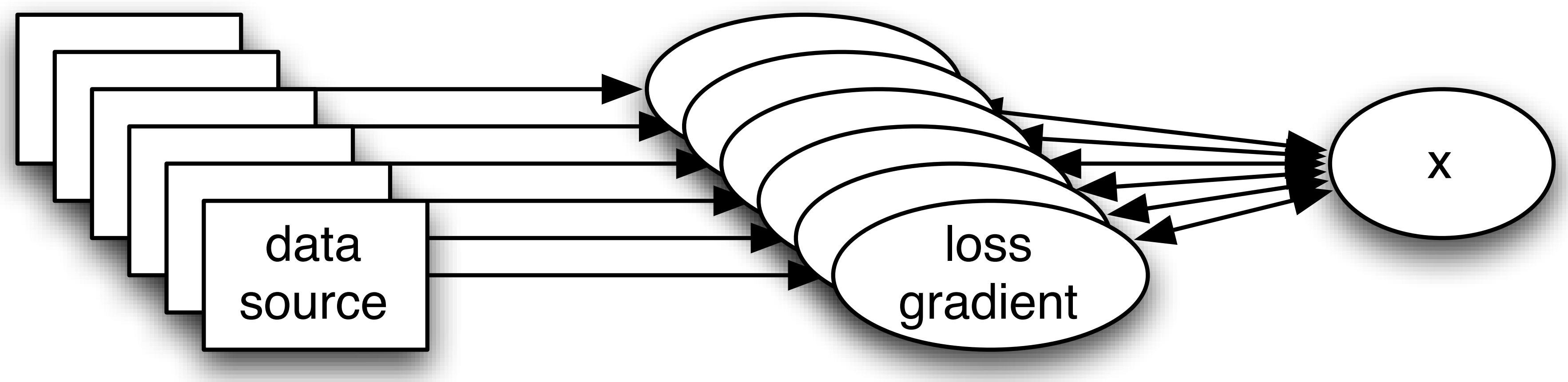
Parallel distributed variants

Online Learning

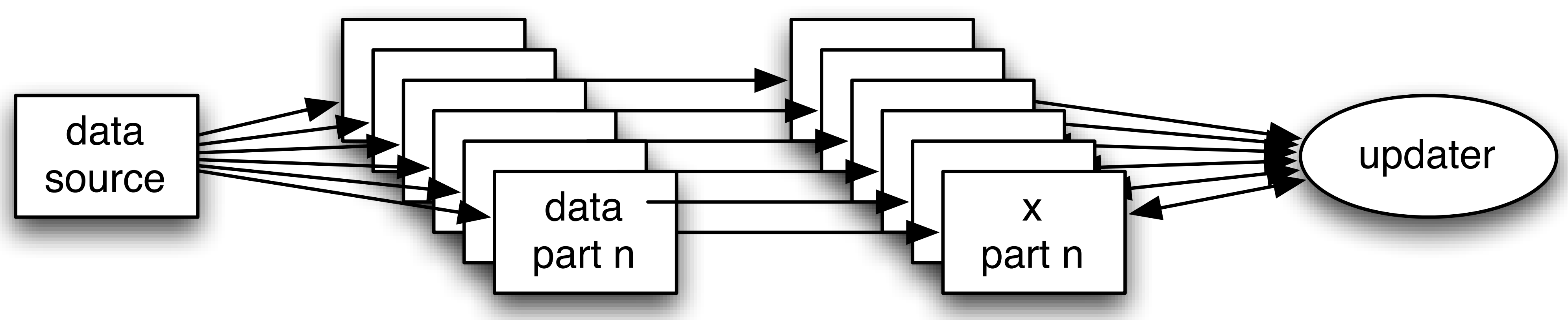
- **General Template**
  - Get instance
  - Compute instantaneous gradient
  - Update parameter vector
- **Problems**
  - Sequential execution (single **core**)
  - CPU **core** speed is no longer increasing
  - Disk/network bandwidth: 300GB/h
  - Does **not** scale to TBs of data
  - Batch subgradient has 50x penalty

Parallel Online Templates

- **Data parallel**



- **Parameter parallel**



Delayed Updates

- **Data parallel**
  - n processors compute gradients
  - delay is n-1 between gradient computation and application
- **Parameter parallel**
  - delay between partial computation and feedback from joint loss
  - delay logarithmic in processors

Delayed Updates

- Optimization Problem

$$\underset{w}{\text{minimize}} \sum_i f_i(w)$$

- Algorithm

Input: scalar $\sigma > 0$ and delay τ
for $t = \tau + 1$ **to** $T + \tau$ **do**
  Obtain f_t and incur loss $f_t(w_t)$
  Compute $g_t := \nabla f_t(w_t)$ and set $\eta_t = \frac{1}{\sigma(t-\tau)}$
  Update $w_{t+1} = w_t - \eta_t g_{t-\tau}$
end for

Theoretical Guarantees

- **Worst case guarantee**
  SGD with delay τ on τ processors is no worse than sequential SGD
- **Lower bound is tight**
  Proof: send same instance τ times
- **Better bounds with iid data**
  - Penalty is covariance in features
  - Vanishing penalty for smooth f(w)

Theoretical Guarantees

- **Linear function classes**

$$\mathbf{E}[f_i(w)] \leq 4RL\sqrt{\tau T}$$

Algorithm converges no worse than with serial execution. Up to a factor of 4 as tight.

- **Strong convexity**

$$R[X] \leq \lambda\tau R + \left[\tfrac{1}{2} + \tau\right] \frac{L^2}{\lambda} (1 + \tau + \log T)$$

Each loss function is strongly convex with modulus λ. Constant offset depends on the degree of parallelism.

Nonadversarial Guarantees

- **Lipschitz continuous loss gradients**

  $$\mathbf{E}[R[X]] \leq \left[28.3R^2H + \frac{2}{3}RL + \frac{4}{3}R^2H\log T\right]\tau^2 + \frac{8}{3}RL\sqrt{T}.$$

  Asymptotic rate does **no longer** depend on amount of parallelism

- **Strong convexity and Lipschitz gradients**

  $$\mathbf{E}[R[X]] \leq O(\tau^2 + \log T)$$

  This only works when the objective function is very close to a parabola (upper and lower bound)

- **Lock-free updates**

  (Hogwild - Recht, Wright, Re http://pages.cs.wisc.edu/~brecht/papers/hogwildTR.pdf)

Lazy updates & sparsity

- Sparse gradients (easy with l_2 regularizer)

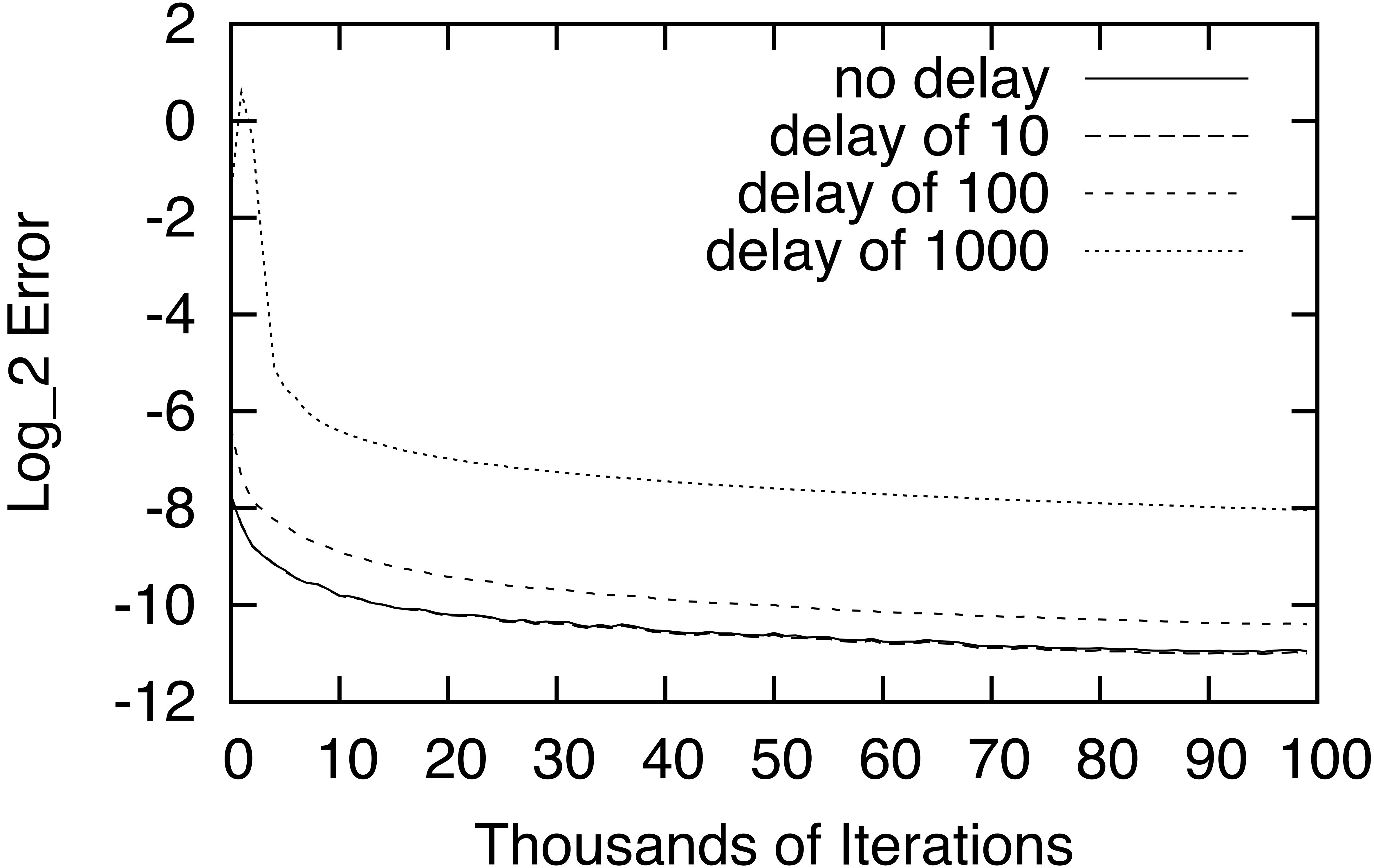
$$w \leftarrow w - \eta_t g(w, x_t) x_t$$

- General coordinate-based penalty

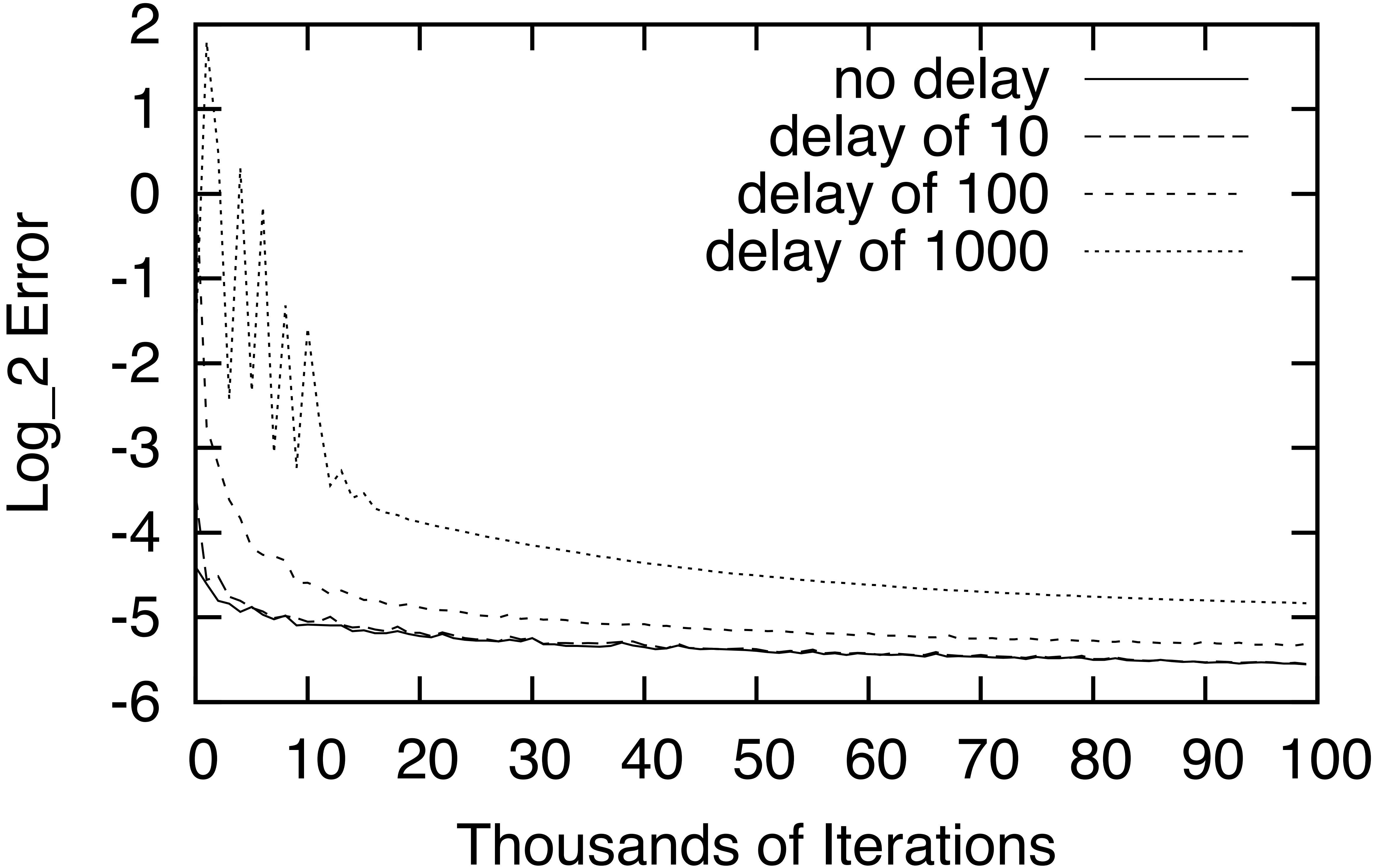
$$\mathbf{E}_{\text{emp}} \left[l(x_i, y_i, w) \right] + \lambda \sum_j \gamma_j(w_j)$$

- Key insight - we only need to know the accurate value of w_j whenever we use it
  - Store w_j with timestamp of last update
  - Before using w_j update using past stepsizes
  - Approximate sum over stepsizes by integral (Quadrianto et al, 2010; Li and Langford, 2009)

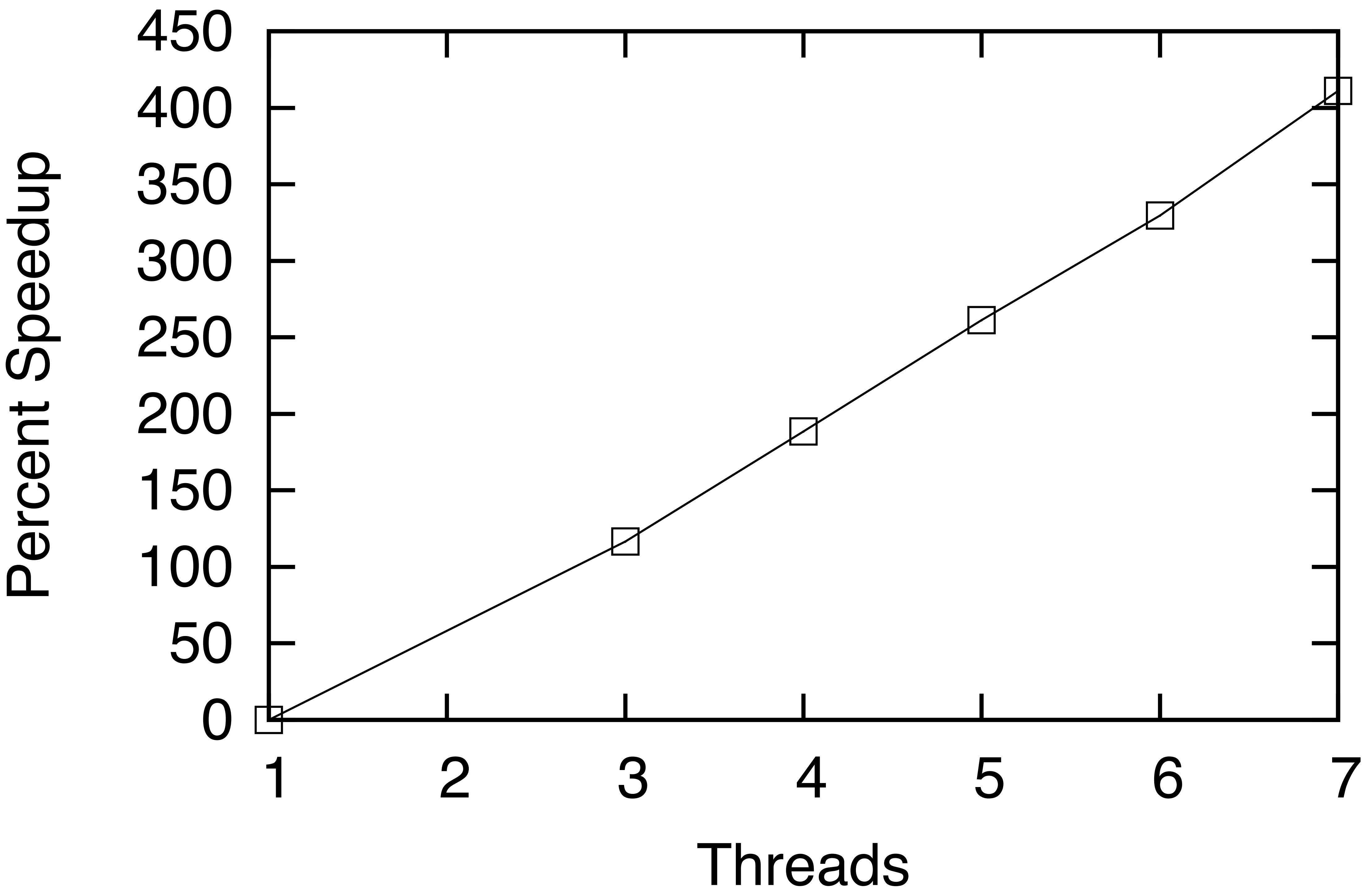
Convergence on TREC



Convergence on Y!Data



Speedup on TREC



Multiple Machines

MapReduce variant

- **Idiot proof simple algorithm**
  - Perform stochastic gradient on each computer for a random subset of the data (drawn with replacement)
  - Average parameters
- **Benefits**
  - No communication during optimization
  - Single pass MapReduce
  - Latency is not a problem

Guarantees

- **Requirements**
  - Strongly convex loss
  - Lipschitz continuous gradient
- **Theorem**

$$\mathbf{E}_{w \in D_\eta^{T,k}}[c(w)] - \min_w c(w) \leq \frac{8\eta G^2}{\sqrt{k\lambda}} \sqrt{\|\partial c\|_L} + \frac{8\eta G^2 \|\partial c\|_L}{k\lambda} + (2\eta G^2)$$

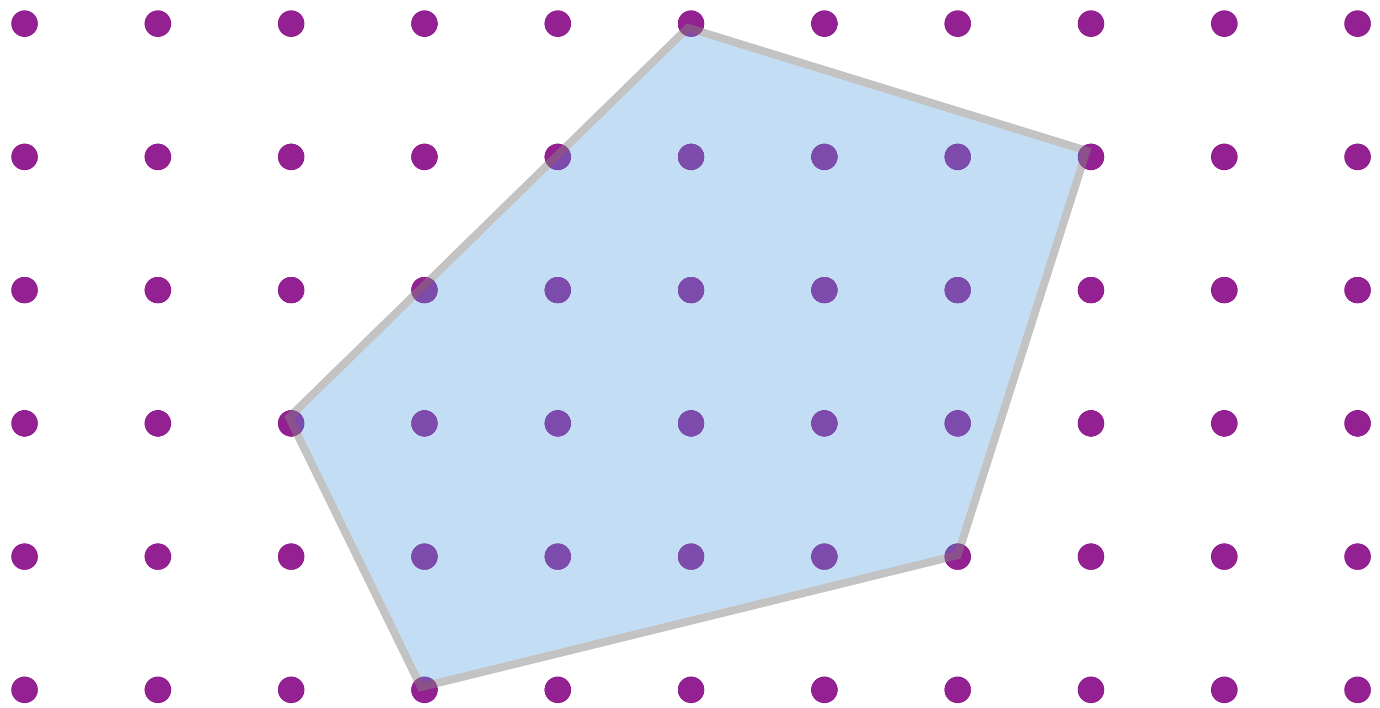
  - Not sample size dependent
  - Regularization limits parallelization
  - For runtime $T = \frac{\ln k - (\ln \eta + \ln \lambda)}{2\eta\lambda}$

4.4 Discrete Problems

Integer programming relaxations

- Optimization problem

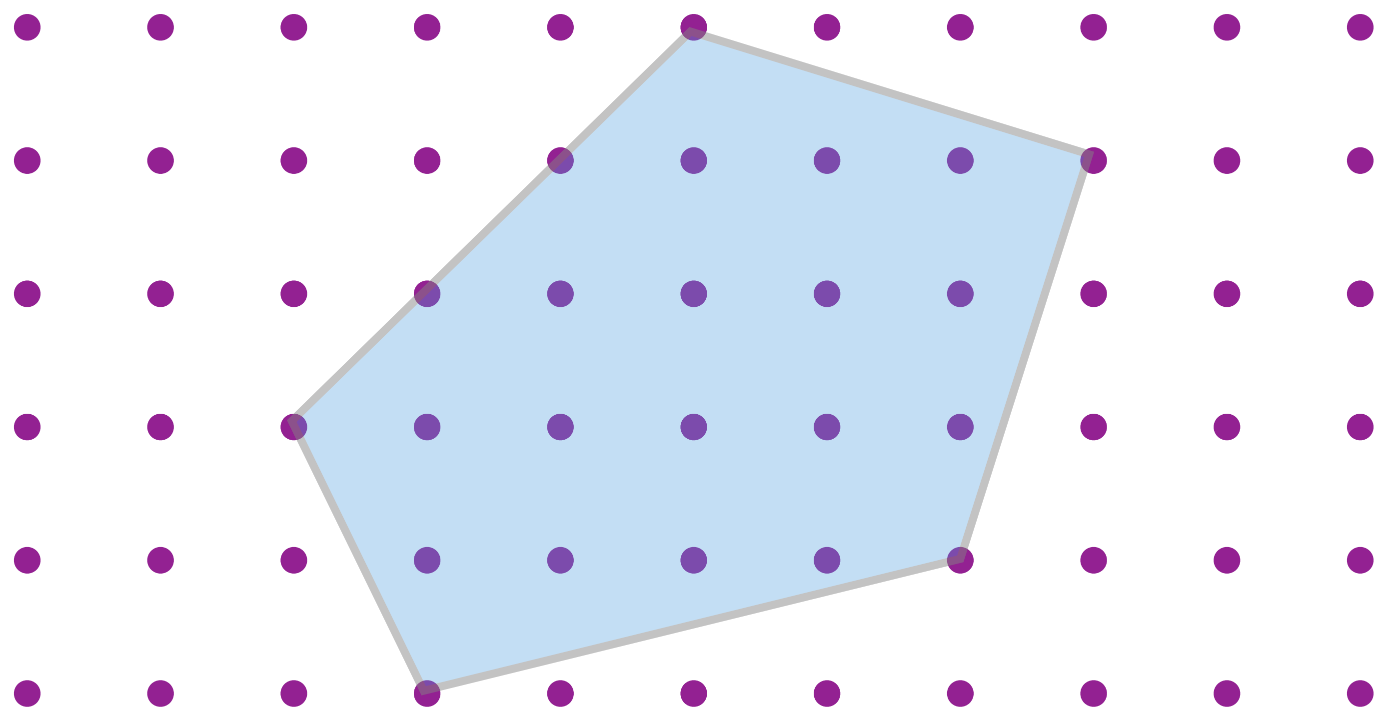
$$\underset{x}{\text{minimize}} \; c^\top x \text{ subject to } Ax \leq b \text{ and } x \in \mathbb{Z}^n$$



- Relax to linear program if vertices are integral since LP has vertex solution

Integer programming relaxations

- Totally unimodular constraint matrix A
  - Inverse of each submatrix must be integral
  - RHS of constraints must be integral
  - Many useful sufficient conditions for TU.



Example - Hungarian Marriage

- Optimization Problem
  - n Hungarian men
  - n Hungarian women
  - Compatibility c_{ij} between them

- Find optimal matching

$$\underset{\pi}{\text{maximize}} \sum_{ij} \pi_{ij} C_{ij}$$

$$\text{subject to } \pi_{ij} \in \{0,1\} \text{ and } \sum_i \pi_{ij} = 1 \text{ and } \sum_j \pi_{ij} = 1$$

- All vertices of the constraint matrix are integral

Randomization

- Maximum finding
  - Very large set of instances
  - Find approximate maximum

- Draw a random set of n terms
- Take maximum over subset
  (59 for 95% with 95% confidence)

$$\Pr\left\{F[\max_i x_i] < \epsilon\right\} = (1-\epsilon)^n = \delta$$

$$\text{hence } n = \frac{\log \delta}{\log(1-\epsilon)} \leq \frac{-\log \delta}{\epsilon}$$

Randomization

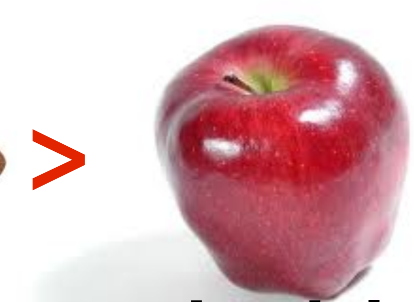
- Find good solution
  - Show that expected value is well behaved
  - Show that tails are bounded
  - Sufficiently large random draw must contain at least one good element (e.g. CM sketch)
- Find good majority
  - Show that majority satisfies condition
  - Bound probability of minority being overrepresented (e.g. Mean-Median theorem)
- Much more in these books
  - Raghavan & Motwani (Randomized Algorithms)
  - Alon & Spencer (Probabilistic Method)

Submodular maximization

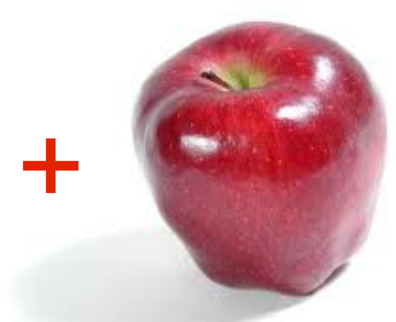
- Submodular function
  - Defined on sets
  - Diminishing returns property

$$f(A \cup C) - f(A) \geq f(B \cup C) - f(B) \text{ for } A \subseteq B$$

- Example
  For web search results we might have individually

 >  >  >  >  > 

But if we can show only 4 we should probably pick

 +  +  + 

Submodular maximization

- Optimization problem

$$\max_{X \in \mathcal{X}} f(X) \text{ subject to } |X| \leq k$$

  Often NP hard even to find tight approximation

- Greedy optimization procedure
  - Start with empty set X
  - Find x such that $f(X \cup \{x\})$ is maximized
  - Add x to the set and repeat until |X|=k

Applications

- Feature selection
- Active learning and experimental design
- Disease spread detection in networks
- Document summarization
- Learning graphical models
- Extensions to
  - Weighted item sets
  - Decision trees

YAHOO!

Optimization

Basic Techniques

- Gradient descent
- Newton's method
- Conjugate Gradient Descent
- Broden-Fletcher-Goldfarb-Shanno (BFGS)

- Constrained Convex Optimization
  - Properties
  - Lagrange function
  - Wolfe dual
- Batch methods
  - Distributed subgradient
  - Bundle methods
- Online methods
  - Unconstrained subgradient
  - Gradient projections
  - Parallel optimization

Further reading

- Nesterov and Vial (expected convergence)
  http://dl.acm.org/citation.cfm?id=1377347
- Bartlett, Hazan, Rakhlin (strong convexity SGD)
  http://books.nips.cc/papers/files/nips20/NIPS2007_0699.pdf
- TAO (toolkit for advanced optimization)
  http://www.mcs.anl.gov/research/projects/tao/
- Ratliff, Bagnell, Zinkevich
  http://martin.zinkevich.org/publications/ratliff_nathan_2007_3.pdf
- Shalev-Shwartz, Srebro, Singer (Pegasos paper)
  http://dl.acm.org/citation.cfm?id=1273598
- Langford, Smola, Zinkevich (slow learners are fast)
  http://arxiv.org/abs/0911.0491
- Hogwild (Recht, Wright, Re)
  http://pages.cs.wisc.edu/~brecht/papers/hogwildTR.pdf