MACHINE LEARNING DEPARTMENT

5.1 Linear Algebra

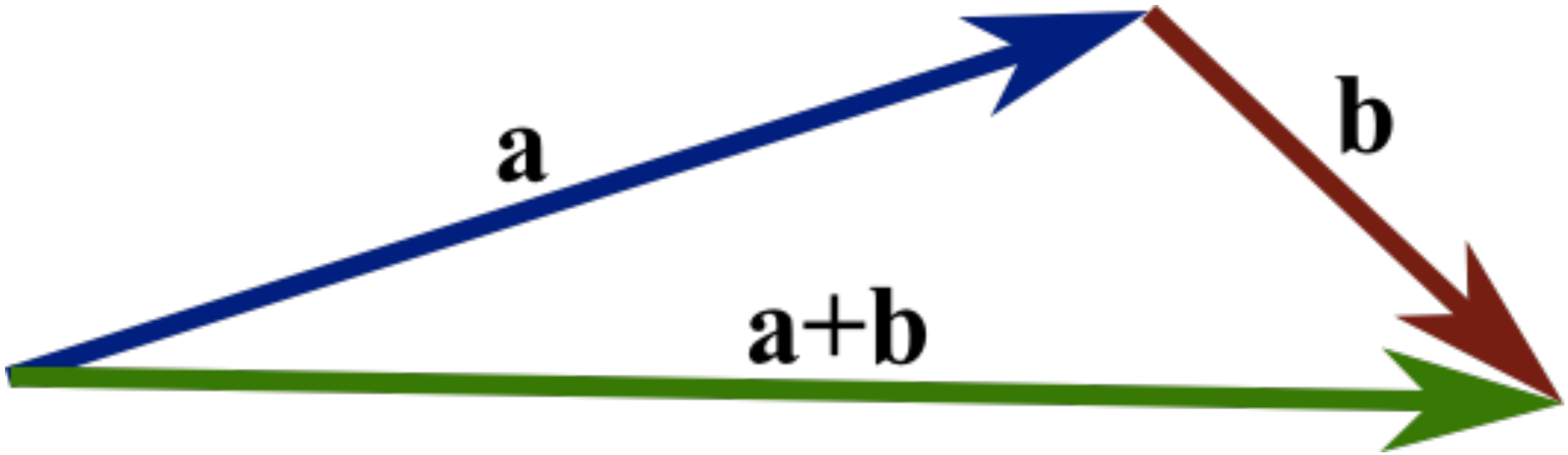
5 Math and Optimization

Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

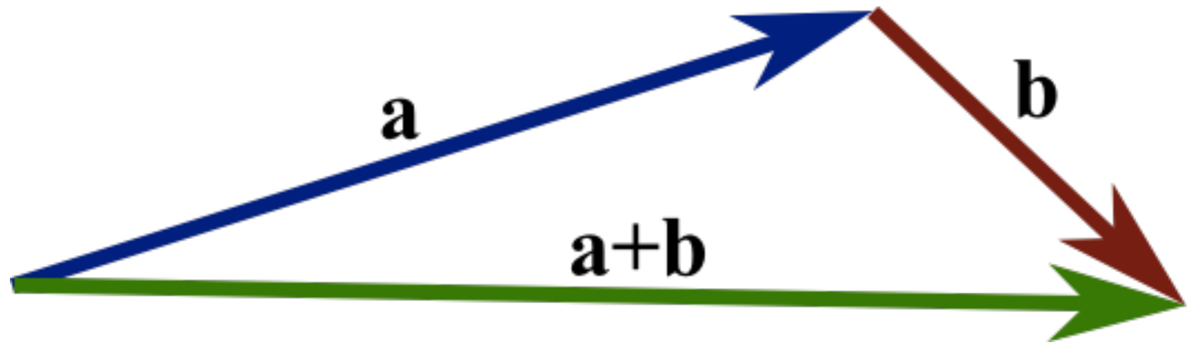
Carnegie Mellon University



Vectors

Vector space

- Space of objects V that look like vectors …



- Example - points in 3D space. We can add them up, scale them, subtract them, etc.

$$(3, 2, 9) + (1, 0.5, -3) = (4, 2.5, 6)$$

- Example - polynomials

$$(3x^2 + 2x^5 + 9x^6) + (x + 0.5x^5 - 3x^6) = (x + 3x^2 + 2.5x^5 + 6x^6)$$

Vector space

- Associativity and commutativity

$$a + (b + c) = (a + b) + c \text{ and } a + b = b + a$$

- Identity and inverse element

$$a + 0 = a \text{ for all } a \text{ and } a + (-a) = 0$$

- Scalar multiplication

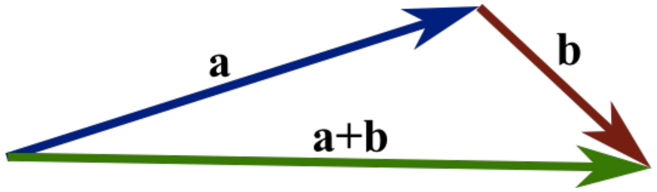
$$\alpha(\beta a) = (\alpha\beta)a \text{ and } 1a = a \text{ for all } a, \alpha, \beta$$

- Distributive law

$$(\alpha + \beta)a = \alpha a + \beta a \text{ and } \alpha(a + b) = \alpha a + \alpha b$$

Examples

- Coordinate spaces



- Polynomials

$$5x^2 + 9.1x^3$$

- Function spaces (generalizes polynomials)

$$\alpha f(x) + \beta g(x) \in V$$

- Linear systems of equations

```
 a  + 3b +  c = 0
4a  + 2b + 2c = 0
```

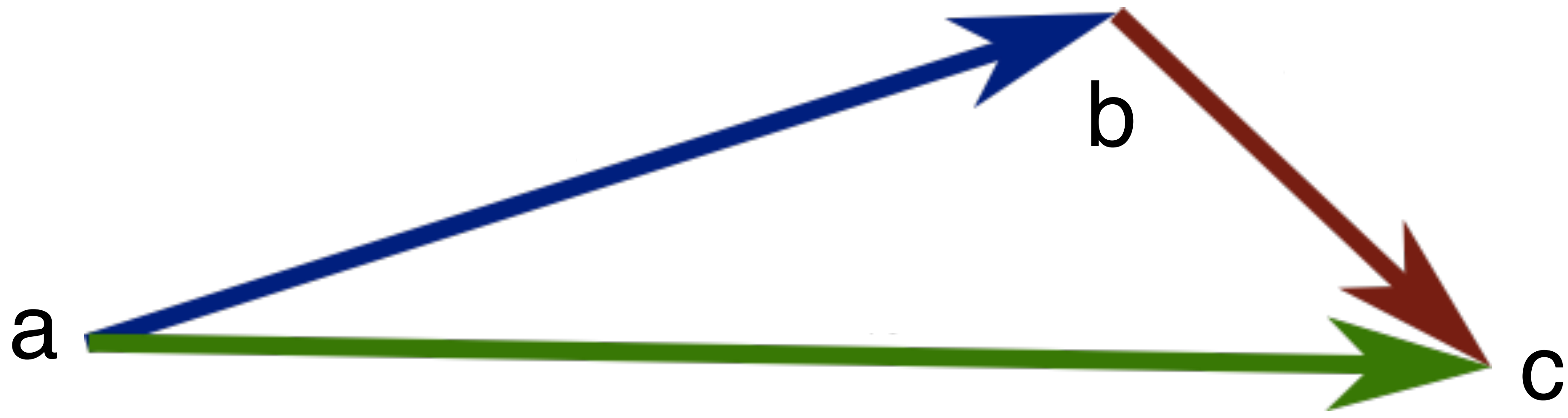
  any linear combination holds, too

- Complex numbers

& Inner Products

Metric

- Symmetric distance function d(a,b)



- Conditions

$$d(a, b) \geq 0 \text{ for all } a, b$$

$$d(a, b) = 0 \text{ implies } a = b$$

$$d(a, b) + d(b, c) \geq d(a, c) \text{ (triangle)}$$

Example

- Euclidean distance

$$d(a,b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

- Trivial distance

$$d(a,b) = \sum_{i=1}^{d} \{a_i \neq b_i\}$$

- Linear combination

$$d(a,b) = d'(a,b) + d''(a,b)$$

Norm

- Essentially a translation invariant metric

$$\|a\| \geq 0 \text{ for all } a$$
$$\|a\| = 0 \text{ implies } a = 0$$
$$\|\alpha a\| = \alpha \|a\|$$
$$\|a + b\| \leq \|a\| + \|b\|$$

triangle inequality

- From norms to metrics

$$d(a, b) := \|a - b\|$$

symmetry, nonnegativity, triangle inequality

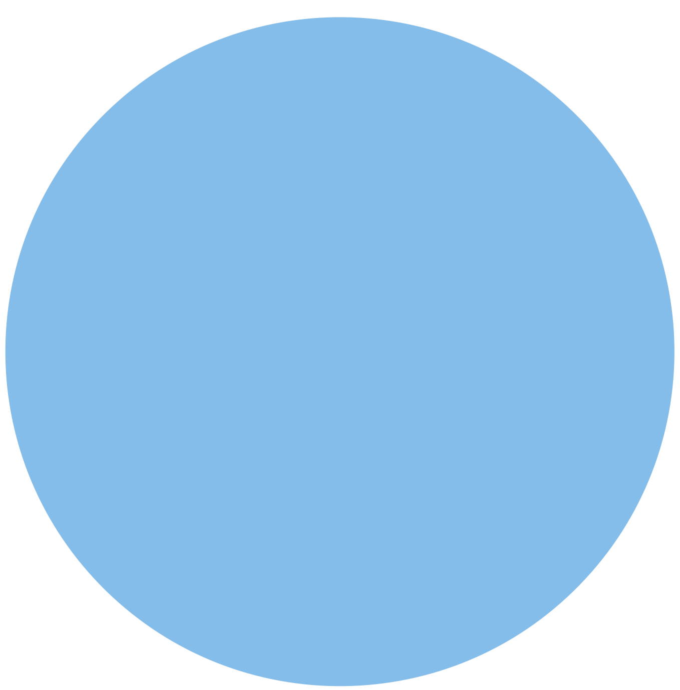
Examples

- Euclidean norm

$$\|a\|_2 := \left[\sum_{i=1}^{d} a_i^2\right]^{\frac{1}{2}}$$

- Unit ball

$$B_1 := \left\{ x \middle| \sum_{i=1}^{d} x_i^2 \leq 1 \right\}$$



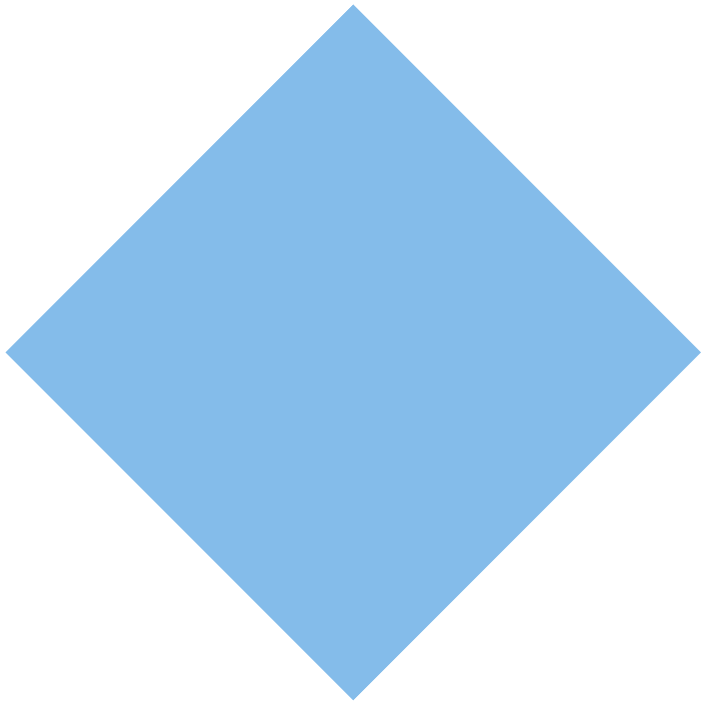
Examples

- l_1 norm

$$\|a\|_1 := \left[\sum_{i=1}^{d} |a_i|\right]$$

- Unit ball

$$B_1 := \left\{ x \,\middle|\, \sum_{i=1}^{d} |x_i| \leq 1 \right\}$$



Examples

- l_∞ norm

$$\|a\|_\infty := \max_{1 \le i \le d} |a_i|$$

- Unit ball

$$B_\infty := \left\{ x \middle| \max_{1 \le i \le d} |x_i| \le 1 \right\}$$

- l_p norm

$$\|a\|_p := \left[\sum_{i=1}^{d} |a_i|^p \right]^{\frac{1}{p}}$$

Carnegie Mellon University

Inner (dot) products

- Linear function on vector space

$$f(\alpha a + b) = \alpha f(a) + f(b)$$

- This is a vector space in its own right
  - Can we find a less awkward notation?
    Write in terms of coordinates

$$\langle a, b \rangle := \sum_{i=1}^{d} a_i b_i$$

  - How about norms?
    Dual norm

$$\|a\|_* := \sup_{\|b\| \leq 1} \langle a, b \rangle$$

Dual Norms

$$\|a\|_* := \sup_{\|b\| \leq 1} \langle a, b \rangle$$

- Scaling (trivial)
- Nonnegativity & symmetry (trivial)
- Triangle inequality

$$\begin{aligned} \|a + a'\|_* &= \sup_{\|b\| \leq 1} \langle a + a', b \rangle \\ &\leq \sup_{\|b\| \leq 1} \langle a, b \rangle + \sup_{\|b\| \leq 1} \langle a', b \rangle \\ &= \|a\|_* + \|a'\|_* \end{aligned}$$

- Holder inequality

$$\langle a, b \rangle \leq \|a\| \, \|b\|_*$$

Examples

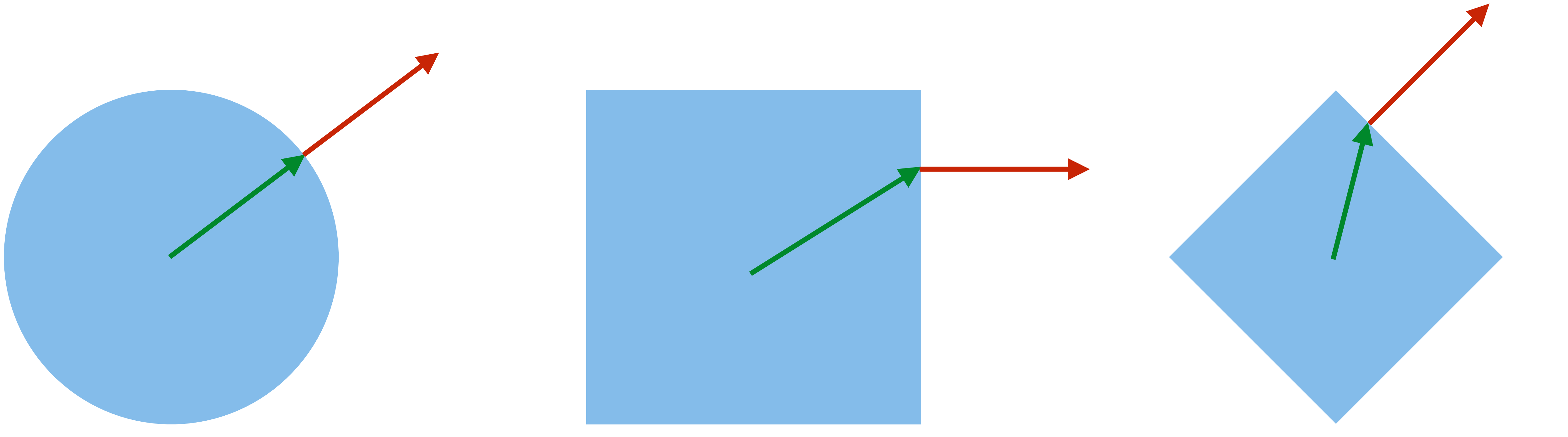
- Example for Euclidean norm

$$\|a\|_* := \sup_{\|b\| \leq 1} \langle a, b \rangle$$

$$\underset{b}{\text{maximize}} \sum_{i=1}^{d} a_i b_i \text{ subject to } \sum_{i=1}^{d} b_i^2 \leq 1$$

- This is solved for unit vector in the same direction as a, i.e. $b = \|a\|_2^{-1} a$
- Dual norm is also Euclidean norm
- Generally, 1/p + 1/q = 1 for norms

Examples



Linear Independence

Basis

- Set of elements in vector space
  - Linear independence

$$\sum_i \alpha_i a_i = 0 \text{ if and only if all } \alpha_i = 0$$

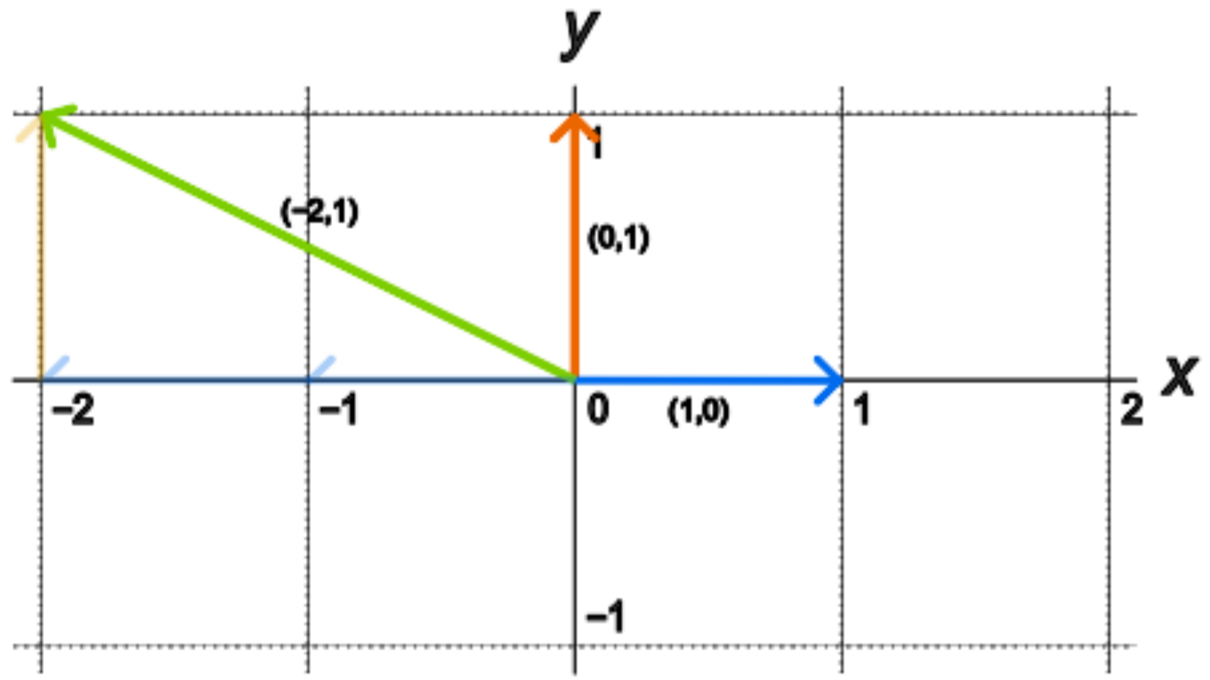
  - Spans the space

$$\text{for all } v \in V \text{ there exists } \sum_i \alpha_i a_i = v$$

- Useful for mapping objects to 'numbers'
  - Vector representation
  - Inner product
  - Linear maps in vector spaces - matrices

Examples

- Euclidean space with canonical basis



- Square integrable functions (L2)
  Fourier basis sin(ax), cos(ax)
- Basis need not be finite set

Not a basis

… but a tight frame (look up wavelet literature)

A basis

… but only a frame (look up wavelet literature)

Basis decomposition

- How do we solve this …

$$\text{for all } v \in V \text{ there exists } \sum_i \alpha_i a_i = v$$

… linear systems are a vector space

$$\underbrace{\langle v, a_j \rangle}_{=:\beta_j} = \left\langle \sum_i \alpha_i a_i, a_j \right\rangle = \sum_i \alpha_i \underbrace{\langle a_i, a_j \rangle}_{=:A_{ij}}$$

- Rewrite in matrix notation

$$\beta = \alpha^\top A$$

Now we are back to regular matrix/vector

Food for thought

- Basis for vector space of polynomials
  - Inner product
  - Restrict to homogeneous polynomials?
- Fourier transform
  - sin ax, cos ax - how many do we need?

Rotations

Orthogonality

- Solving $\beta = \alpha^\top A$ requires work.
  It would be much nicer without A.
- Orthonormal basis

$$\langle a_i, a_j \rangle = \delta_{ij}$$

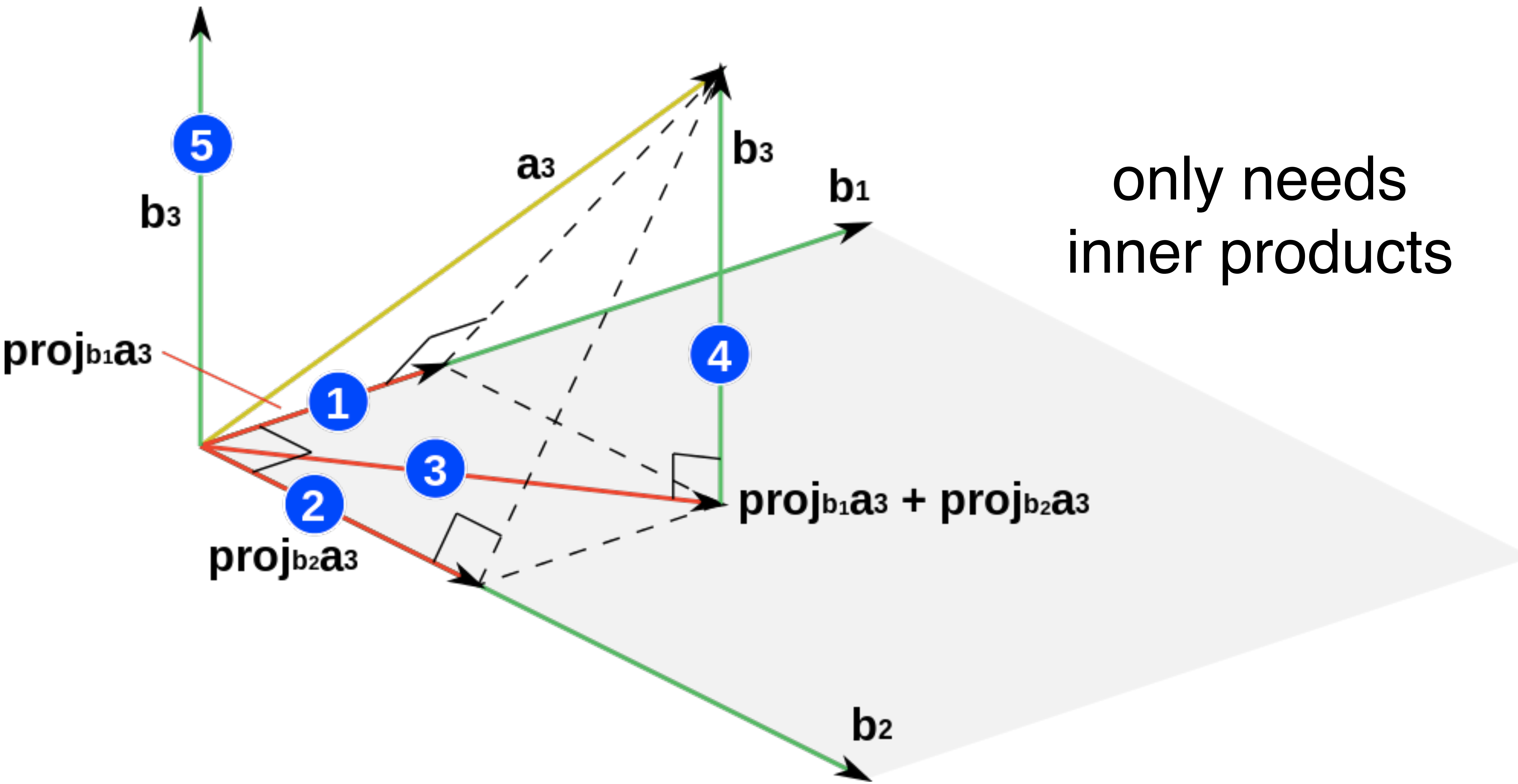
- Gram-Schmidt algorithm for construction

$$v_1 = \|a_1\|^{-1} a_1$$

$$t_i = a_i - \sum_{j=1}^{i-1} v_j \langle a_i, v_j \rangle \text{ and } v_i = \|t_i\|^{-1} t_i$$

Orthogonality follows by induction

Gram Schmidt algorithm



only needs inner products

Orthogonal matrix

- Matrix of orthonormal vectors V
- Product with its transpose

$$\left[V^\top V\right]_{ij} = \sum_l \left[v_i\right]_l \left[v_j\right]_l = \langle v_i, v_j \rangle = \delta_{ij}$$

- Transpose is also orthogonal
- Product of two is still orthogonal

$$(UV)^\top (UV) = V^\top U^\top U V = V^\top V = 1$$

Useful things

- Keeps inner products

$$\langle Va, Vb \rangle = a^\top V^\top V b = a^\top b$$

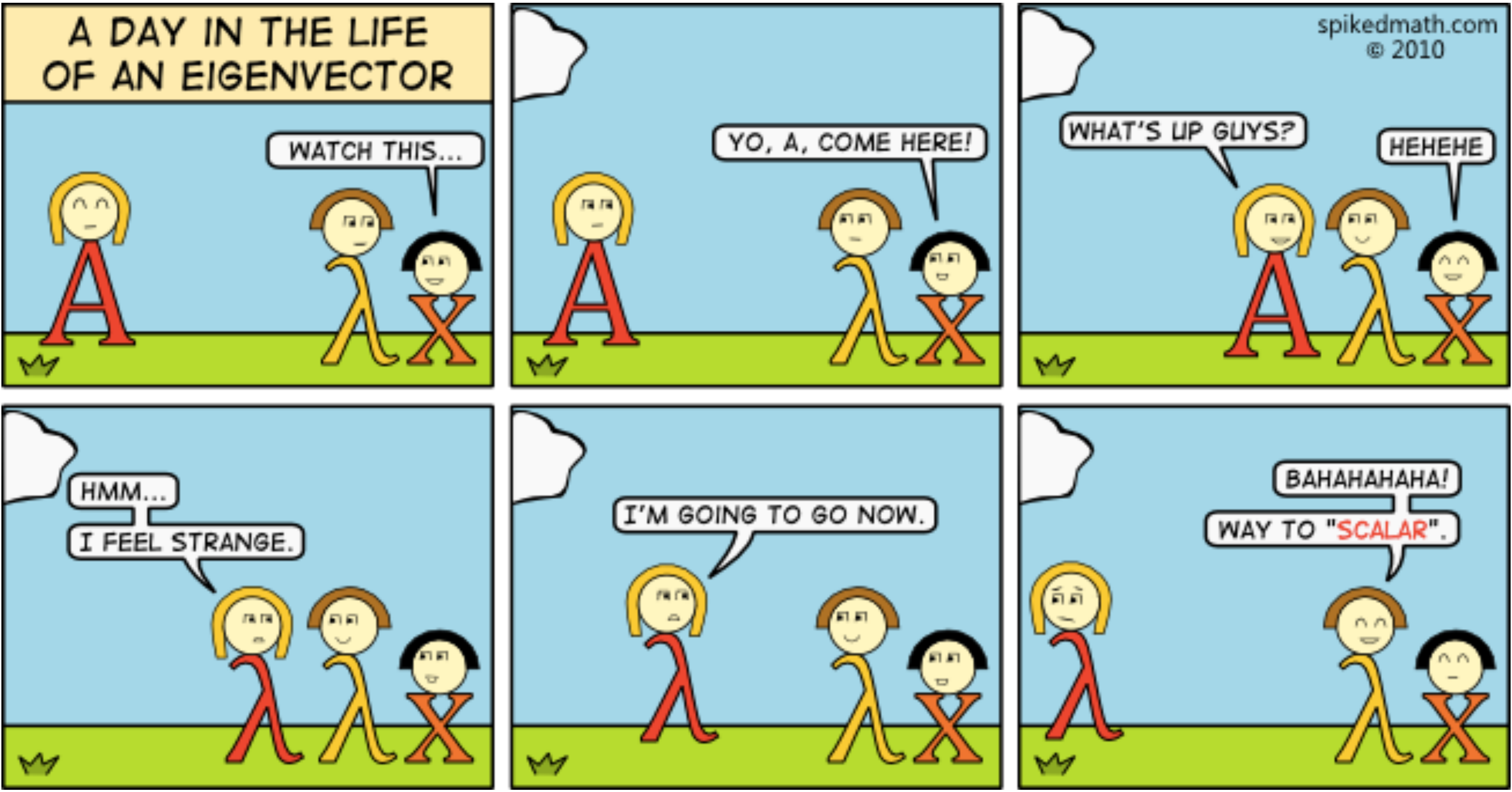
- Keeps lengths invariant, too …

$$\|Va\|^2 = \langle Va, Va \rangle = \langle a, a \rangle = \|a\|^2$$

- We can reconstruct the inner product just from the lengths … hint …

$$\|a + b\|^2 - \|a - b\|^2$$

- Fourier transform does this, too


A DAY IN THE LIFE OF AN EIGENVECTOR
WATCH THIS...
YO, A, COME HERE!
spikedmath.com © 2010
WHAT'S UP GUYS?
HEHEHE
HMM...
I FEEL STRANGE.
I'M GOING TO GO NOW.
BAHAHAHAHA!
WAY TO "SCALAR".

Matrices

Eigenvectors

- Eigenvector & eigenvalue

$$Mx = \lambda x$$

(sanity check) Why does M have to be square?

- For nonsymmetric matrix we can only get a Jordan normal form

$$M = P \begin{bmatrix} \boxed{\begin{matrix}\lambda_1 & 1 & \\ & \lambda_1 & 1 \\ & & \lambda_1\end{matrix}} & & & & \\ & \boxed{\begin{matrix}\lambda_2 & 1\\ & \lambda_2\end{matrix}} & & & \\ & & \boxed{\lambda_3} & & \\ & & & \ddots & \\ & & & & \boxed{\begin{matrix}\lambda_n & 1 \\ & \lambda_n\end{matrix}}\end{bmatrix} P^{-1}$$

Eigenvectors for Symmetric Matrices

- Eigenvector & eigenvalue

$$Mx = \lambda x$$

- Orthogonality

$$\lambda_i \langle x_i, x_j \rangle = \langle M x_i, x_j \rangle = \langle x_i, M x_j \rangle = \lambda_j \langle x_i, x_j \rangle$$

  orthogonality whenever eigenvalues don't match
- Within subspace of same eigenvalues trivial.

$$M = U \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} U^\top$$

Positive Semidefinite Matrices

- All eigenvalues are nonnegative
- Equivalent to $x^\top M x \geq 0$
- Proof

$$x^\top M x = \left[\sum_i \alpha_i x_i\right]^\top M \left[\sum_j \alpha_j x_j\right] = \sum_{i,j} \alpha_i \alpha_j x_i^\top M x_j = \sum_i \alpha_i^2 \lambda_i \geq 0$$

  - If all eigenvalues are nonnegative this holds
  - If one of them is negative, pick matching eigenvector for contradiction

Norms

- Recall - vector norm

$$\|a\|_* := \sup_{\|b\| \leq 1} \langle a, b \rangle$$

- For vectors we get to play with left and right side

$$\sup_{x,y} x^\top M y \text{ subject to } \|x\|, \|y\| \leq 1$$

- Special case - operator norm (l2 space)
  Picks largest eigenvalue / singular value

Norms

- Frobenius Norm

$$\|M\|_{\text{Frob}}^2 = \sum_{ij} M_{ij}^2 = \sum_i \lambda_i^2$$

- Trace Norm

$$\|M\|_{\text{KyFan}} = \sum_i |\lambda_i|$$

- Operator Norm

$$\|M\| = \max_i |\lambda_i|$$

- Simple inequality (follows from norms)

$$n^{-1} \|M\|_{\text{KyFan}} \leq n^{-\frac{1}{2}} \|M\|_{\text{Frob}} \leq \|M\|$$

Trace

- Trace

$$\operatorname{tr} M := \sum_i M_{ii}$$

- Commuting property

$$\operatorname{tr} AB = \sum_i \left[\sum_j A_{ij} B_{ji} \right] = \sum_j \left[\sum_i B_{ji} A_{ij} \right] = \operatorname{tr} BA$$

- Eigenvalue sum

$$\operatorname{tr} M = \operatorname{tr} U^\top \Lambda U = \operatorname{tr} \Lambda U U^\top = \operatorname{tr} \Lambda = \sum_i \lambda_i$$

Determinant

- Product of all eigenvalues

$$\det M = \prod_i \lambda_i = \sum_{\pi \in S_n} \operatorname{sgn}(\pi) \prod_{i=1}^n M_{i\pi(i)}$$

  - Linear in each column of M
  - Sign flips due to permutations
  - Hence det(a,a,….) = 0
  - Hence vanishes for linearly dependence
- Computes n-dimensional volume

5.2 Unconstrained Problems

5 Math and Optimization

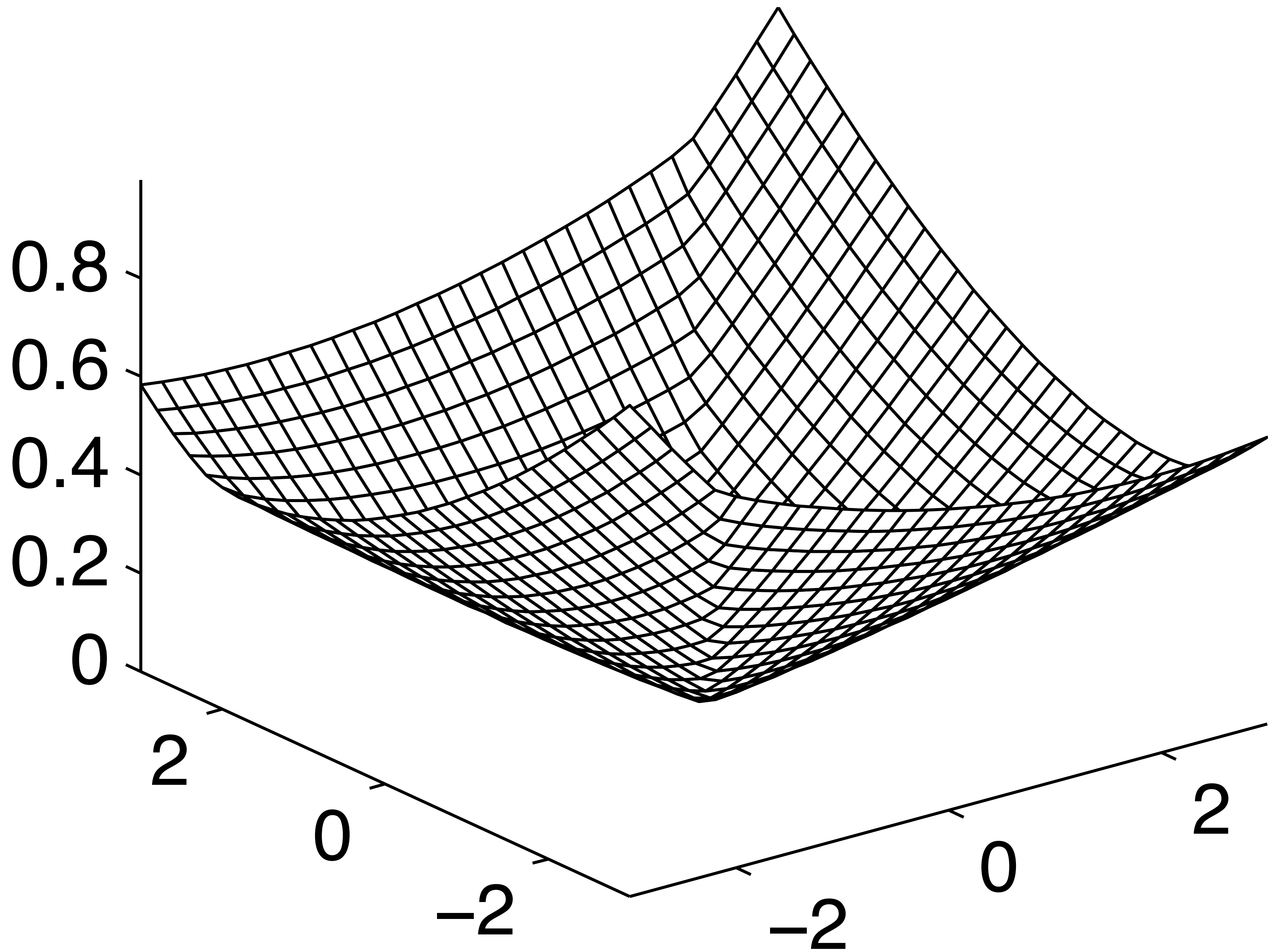
Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

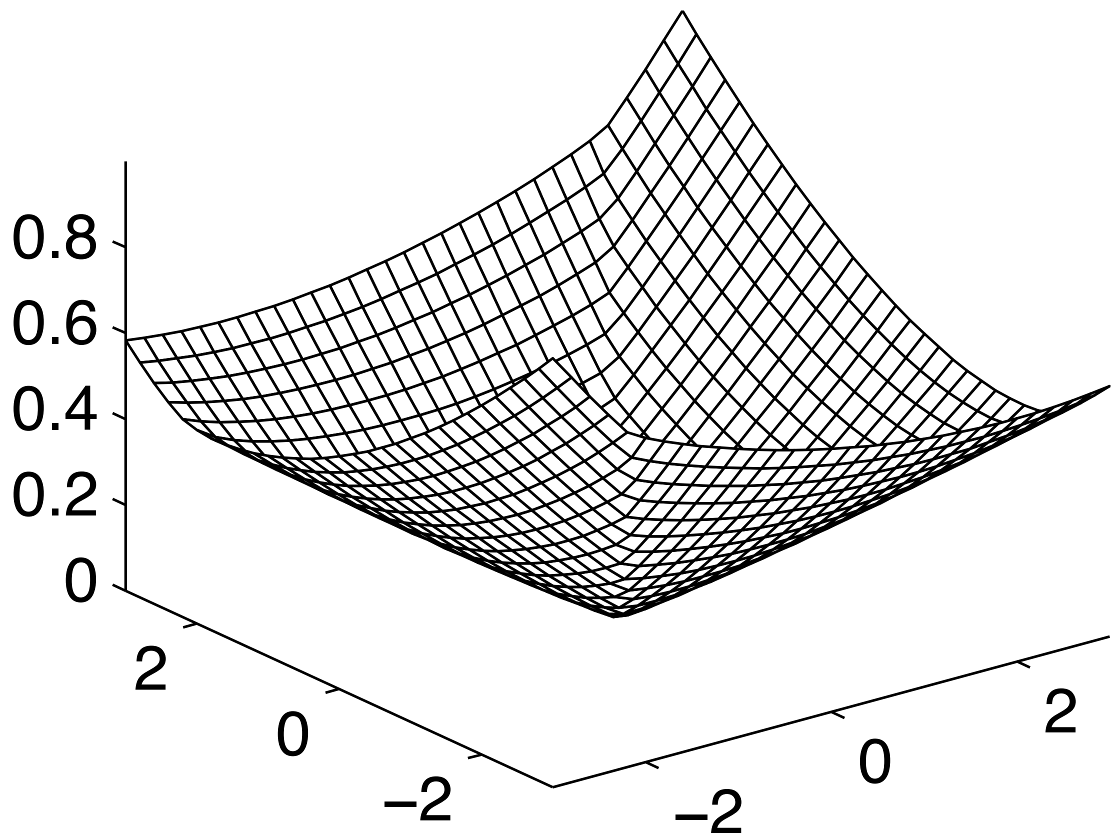
MACHINE LEARNING DEPARTMENT

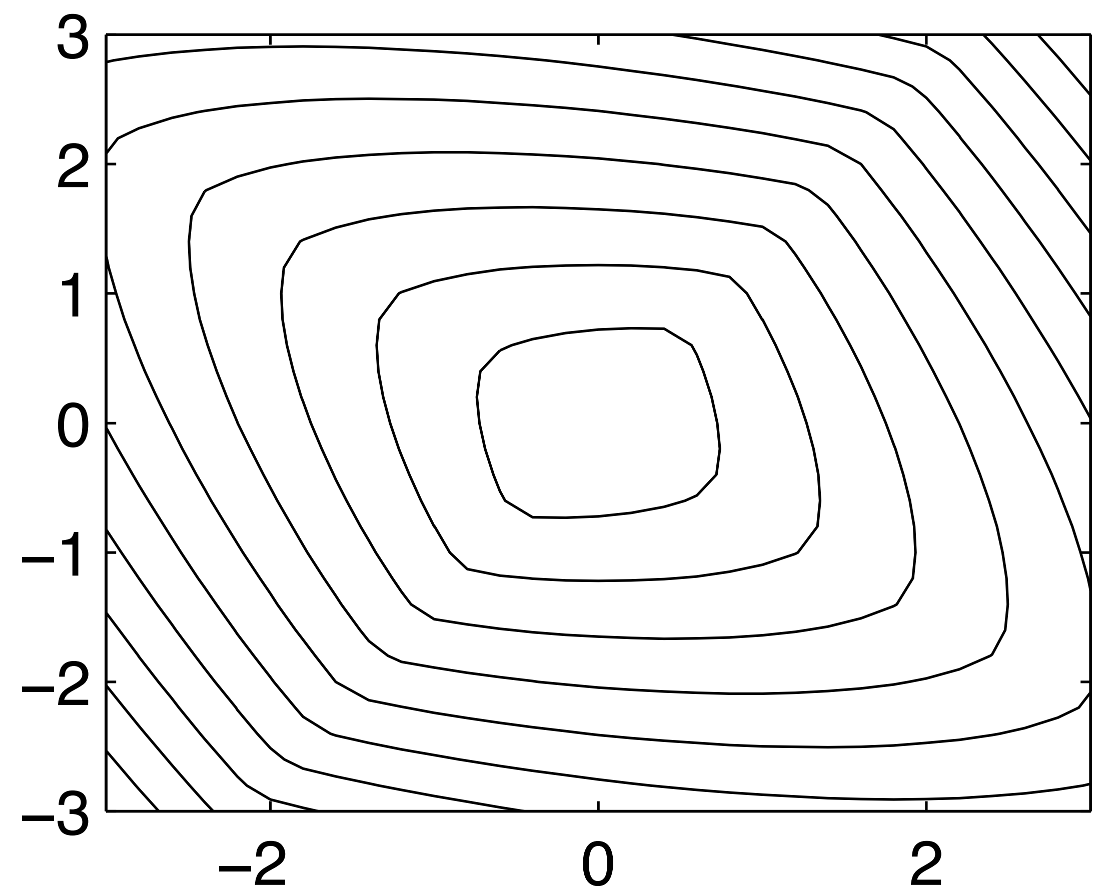
Carnegie Mellon University



Convexity 101

Convexity 101





- Convex set

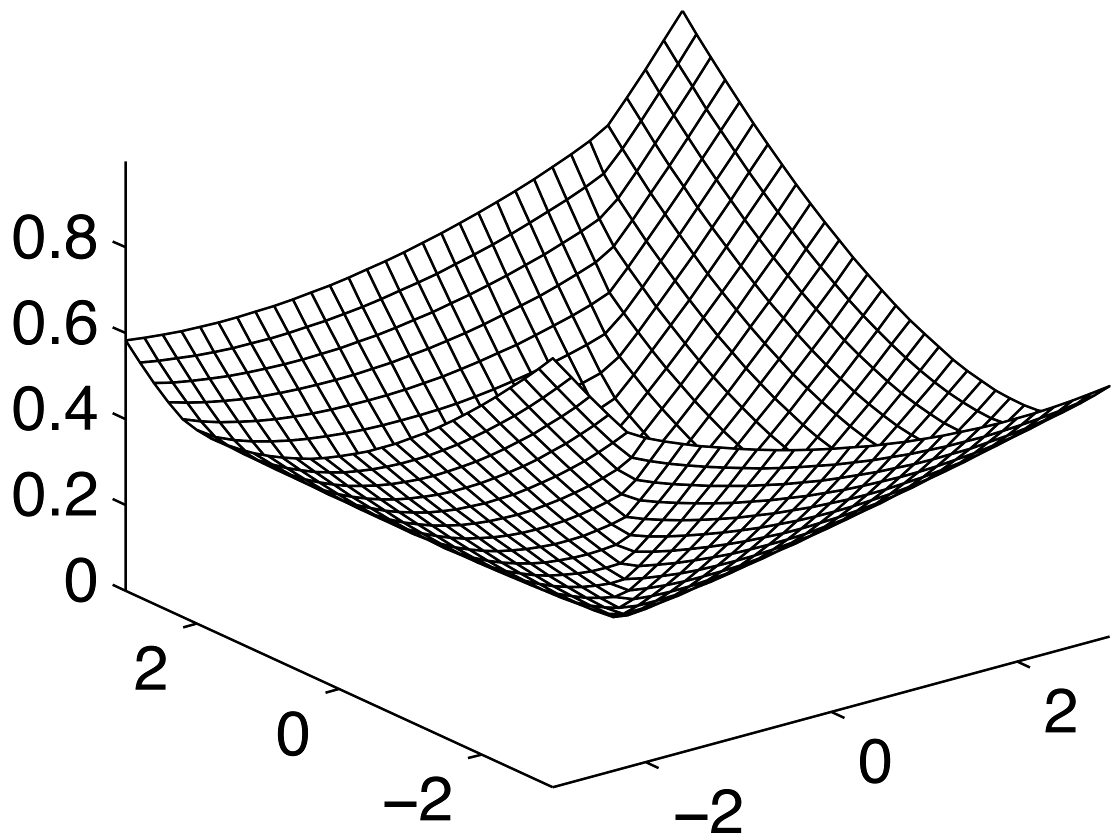
  For $x, x' \in X$ it follows that $\lambda x + (1-\lambda)x' \in X$ for $\lambda \in [0,1]$

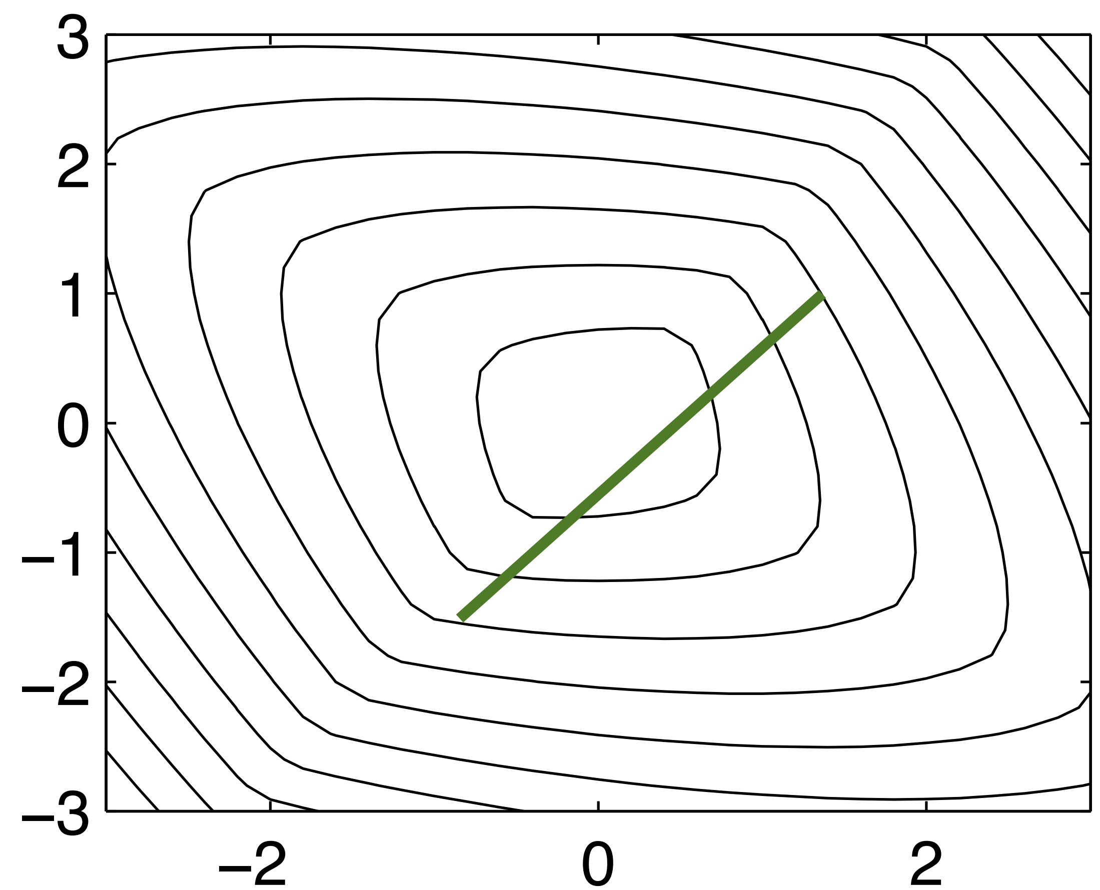
- Convex function

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

Carnegie Mellon University

Convexity 101





- Convex set

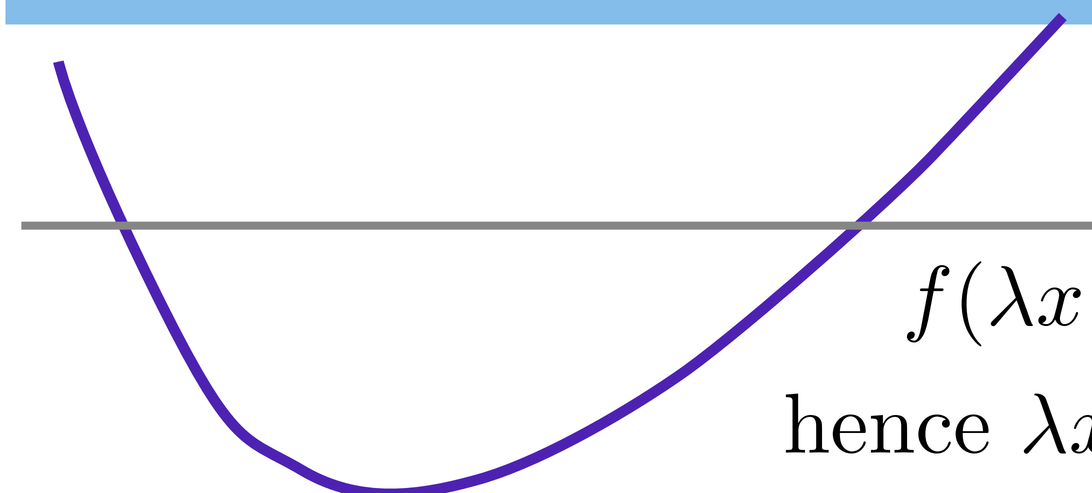
  For $x, x' \in X$ it follows that $\lambda x + (1 - \lambda)x' \in X$ for $\lambda \in [0, 1]$

- 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

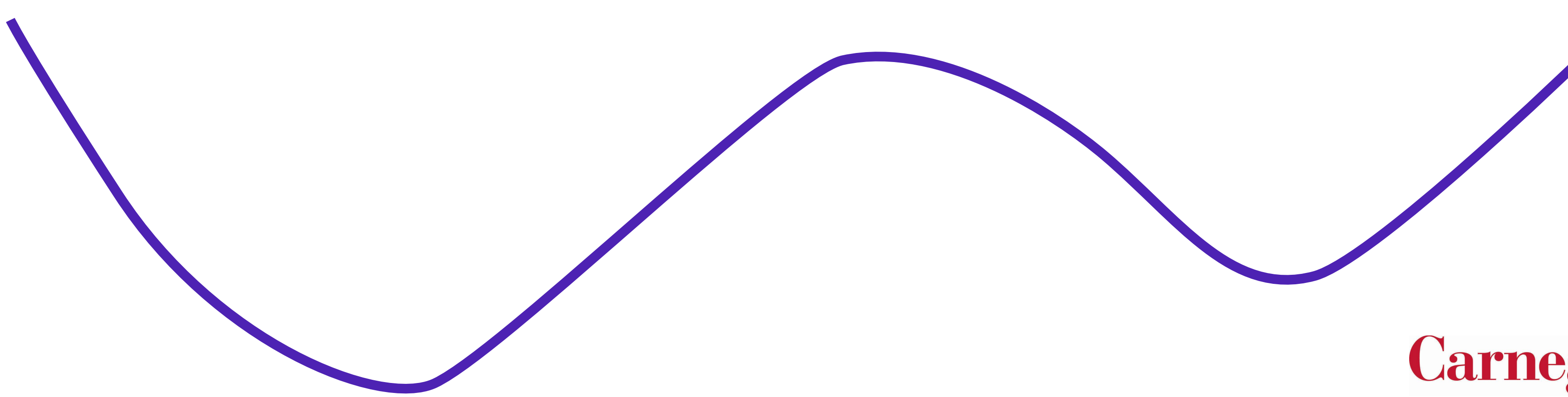


$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

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

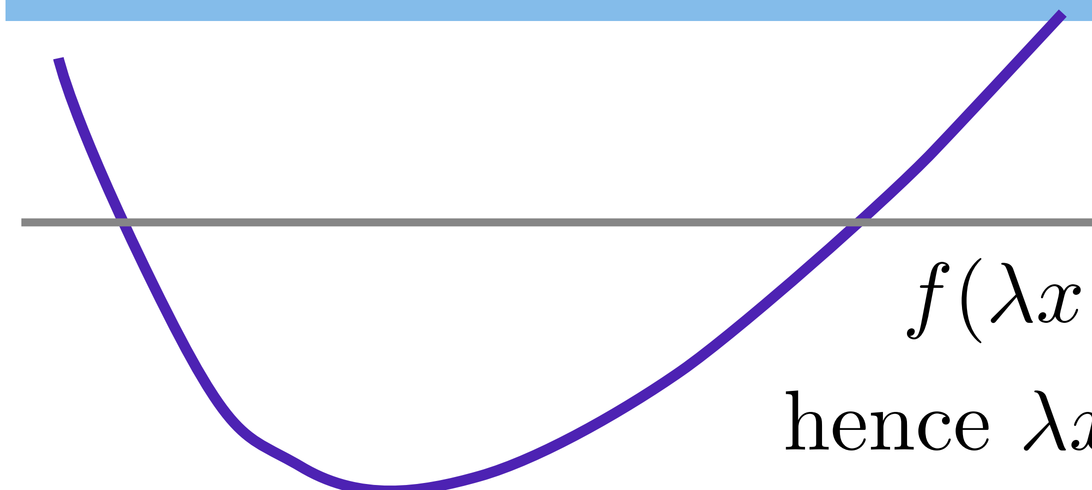
- Convex functions don't have local minima

  Proof by contradiction - linear interpolation breaks local minimum condition



Convexity 101

- Below-set of convex function is convex

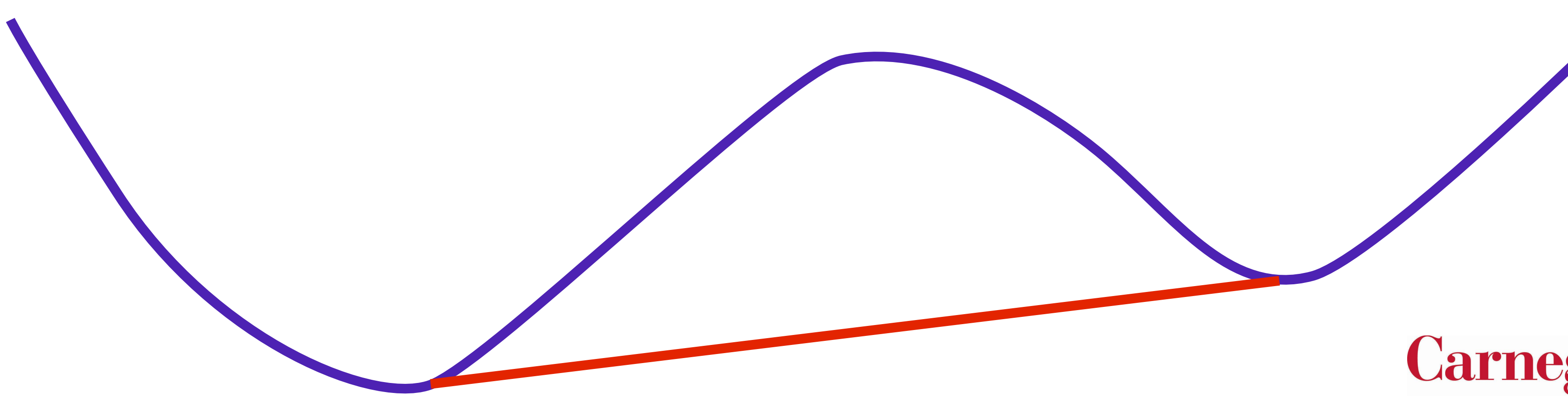


$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$$

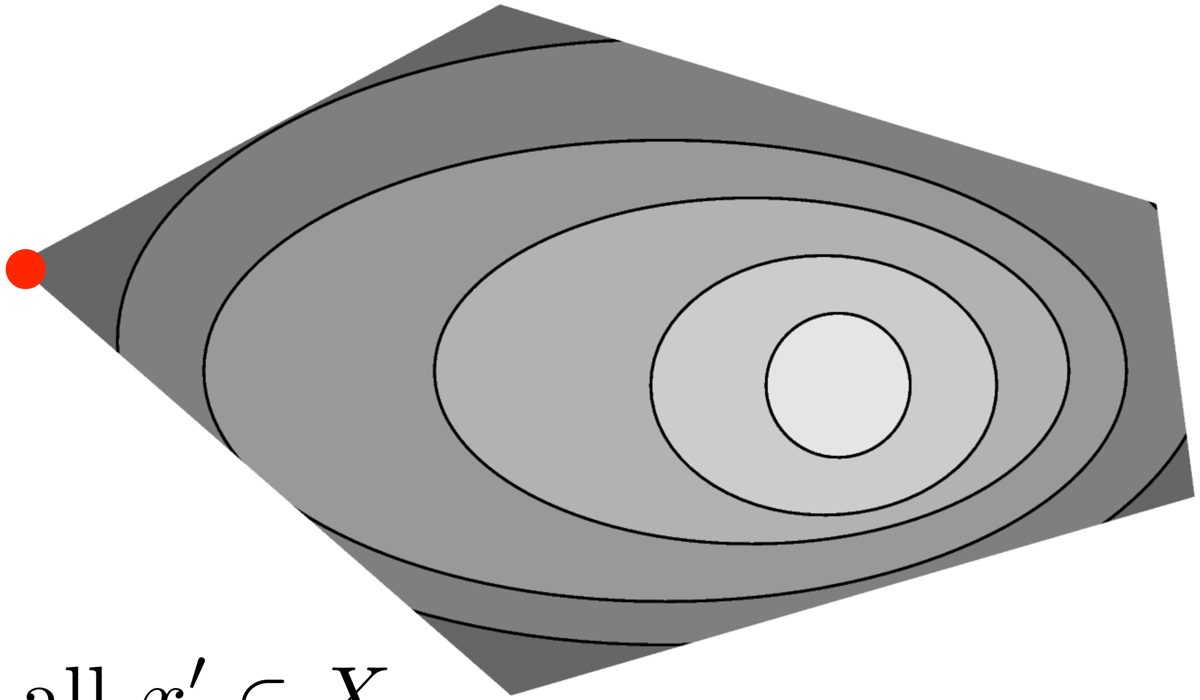
$$\text{hence } \lambda x + (1-\lambda)x' \in X \text{ for } x, x' \in X$$

- 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

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

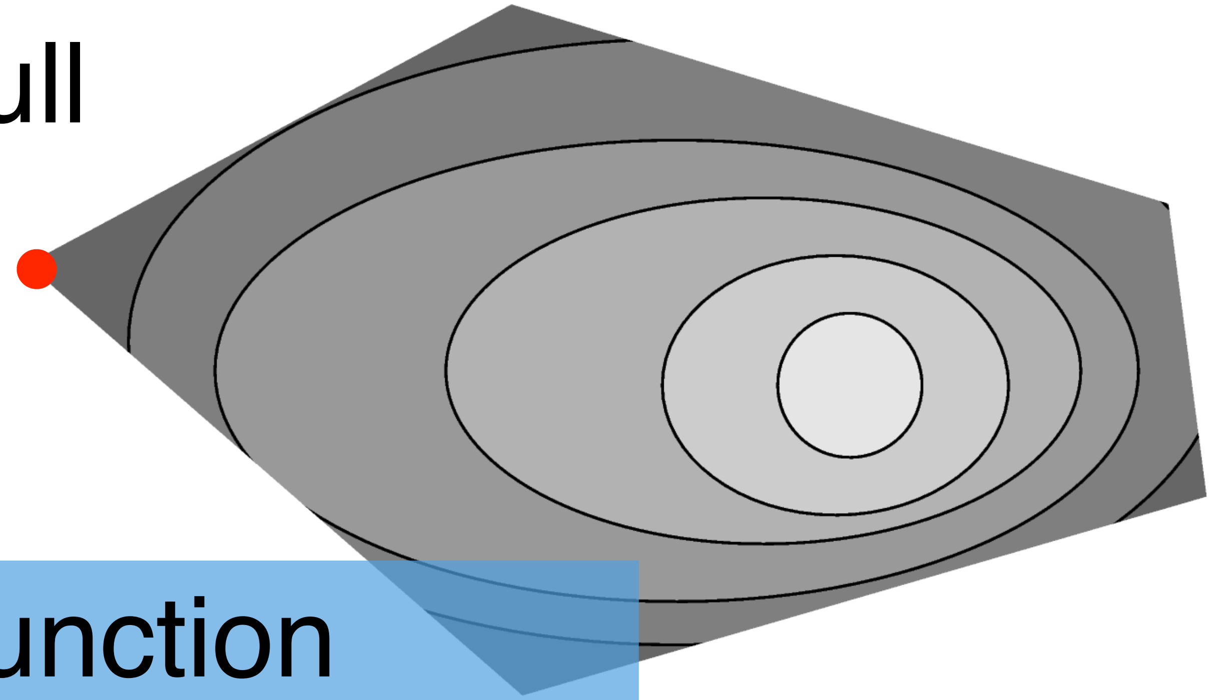
- Convex hull of set is a convex set

Convexity 101

- Supremum on convex hull

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

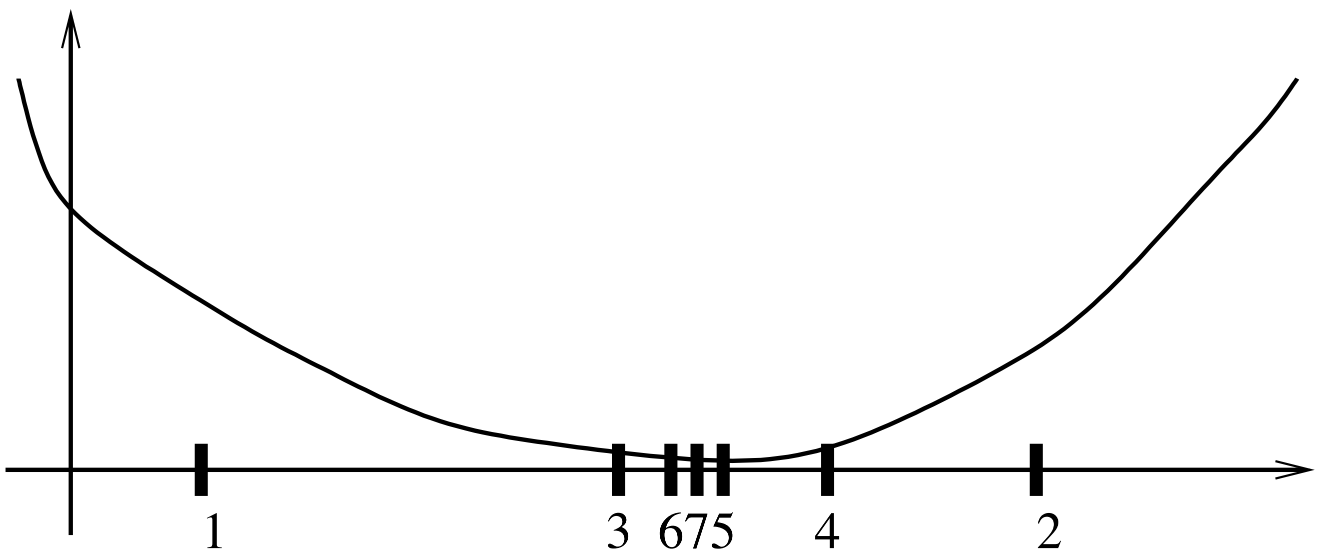
  Proof by contradiction



- 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

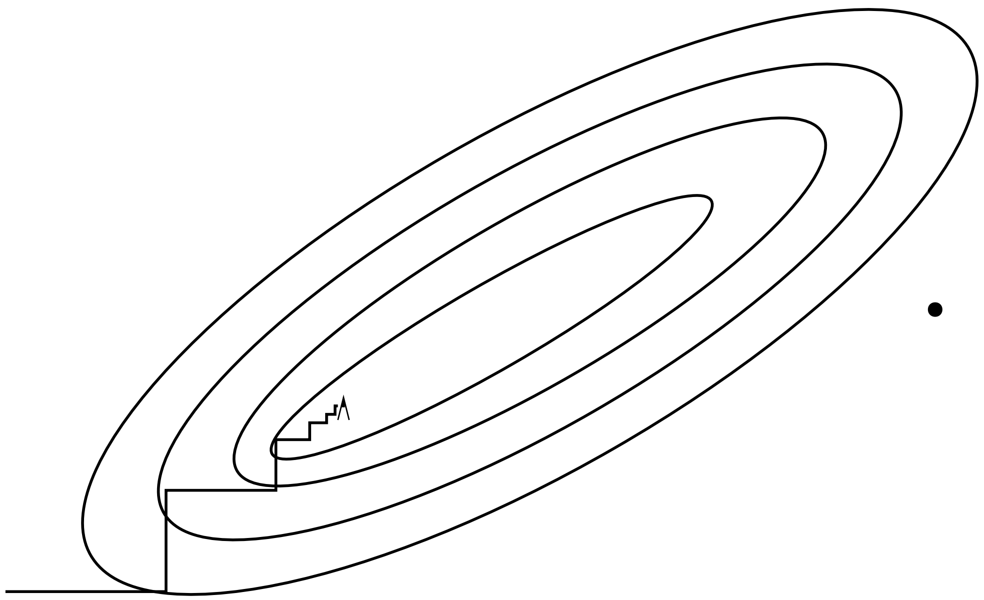
  until $(B - A)\min(|f'(A)|, |f'(B)|) \leq \epsilon$

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

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

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

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

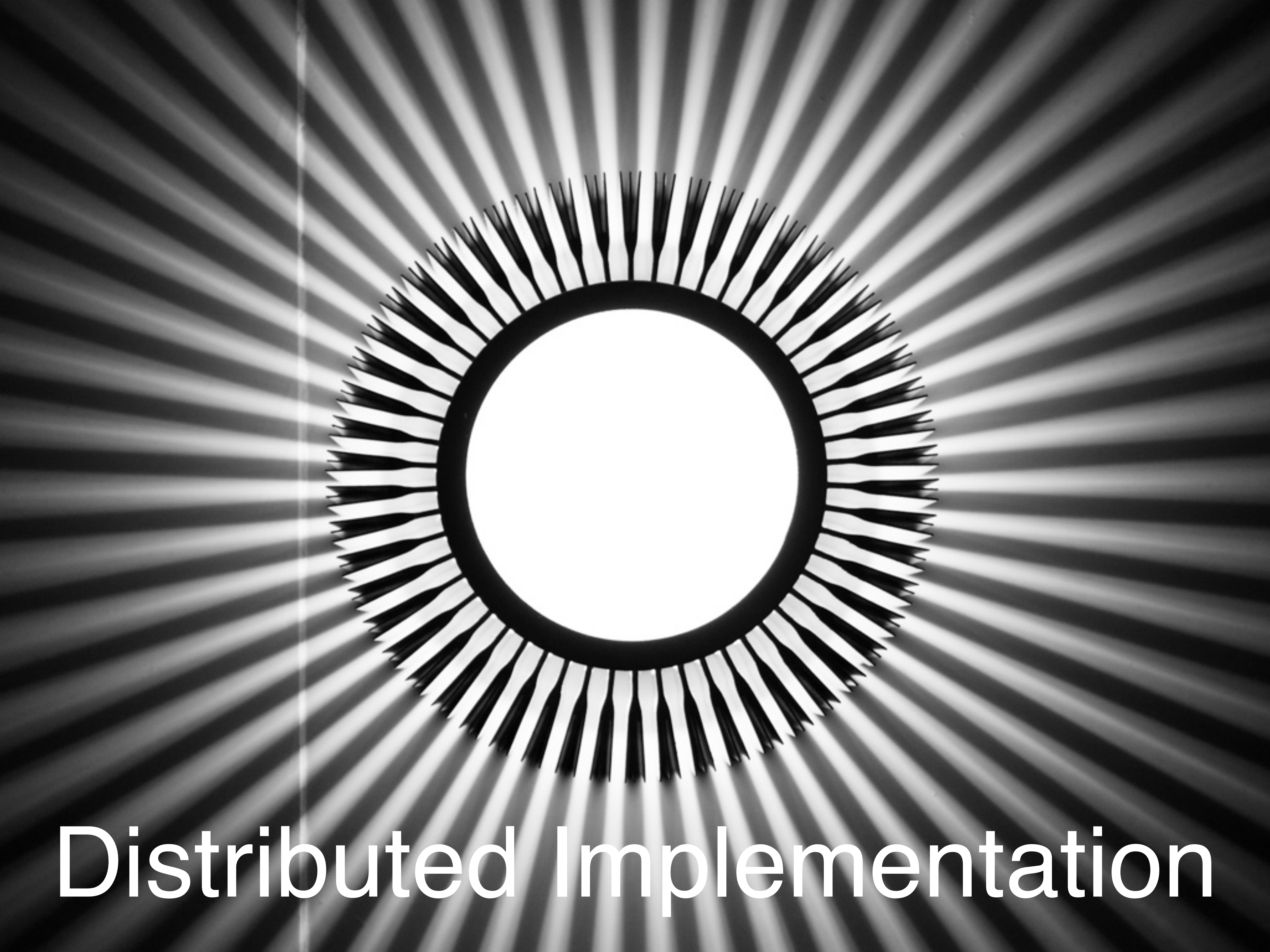
Using strong convexity

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

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

- Iteration bound

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


Distributed Implementation

Basic steps

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

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

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 $x \in \mathbf{dom}\, f$.

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

communicate final value to each machine

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

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

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

compute partial gradients and aggregate

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

- 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)

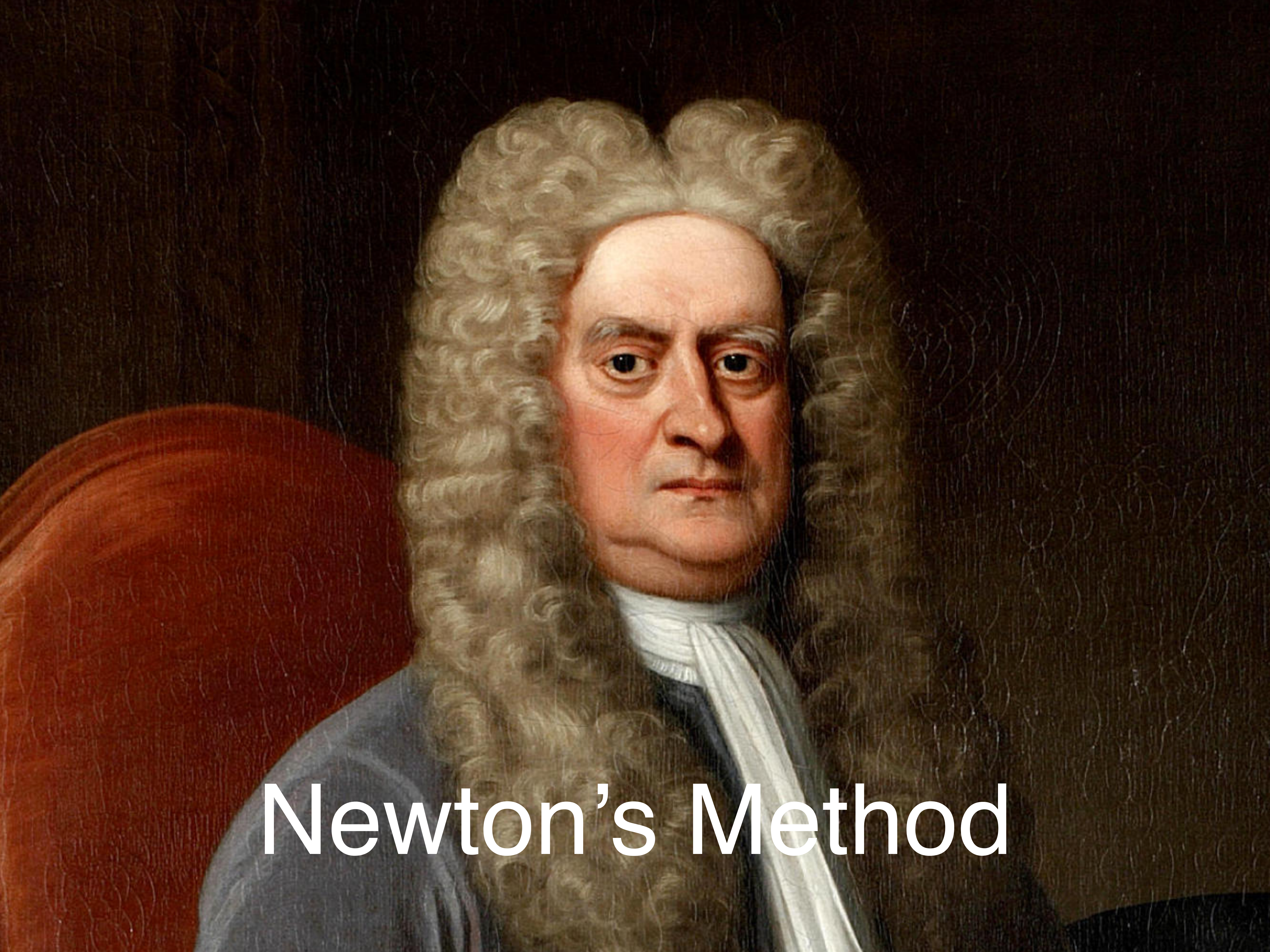
A Better Algorithm

- 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

Newton Method

- Convex objective function f
- Nonnegative second derivative

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

- Taylor expansion

Hessian

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

gradient

- 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

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

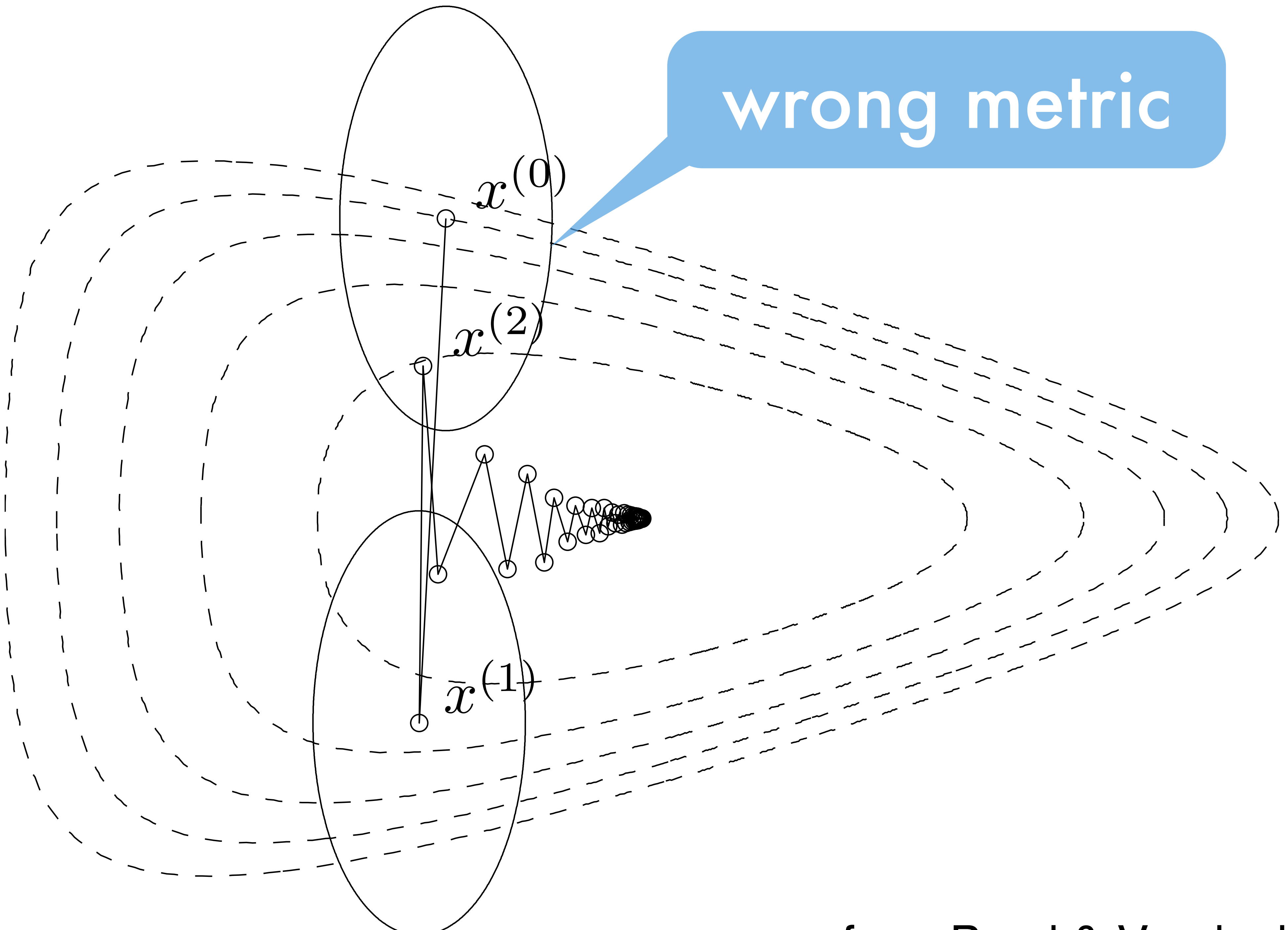
  - Quadratic convergence once the bound holds

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

  - Newton method is affine invariant (proof by chain rule)

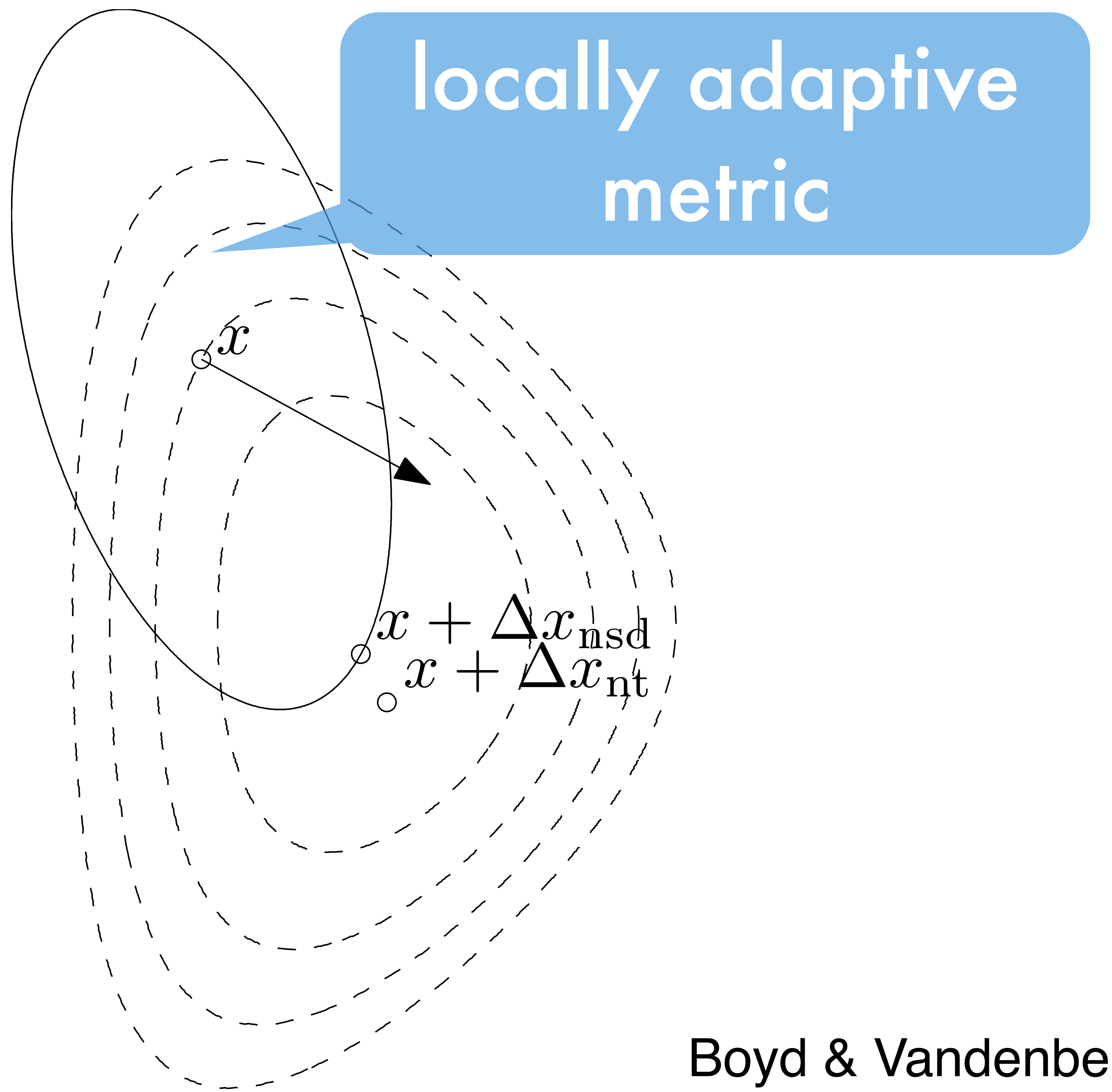
See Boyd and Vandenberghe, Chapter 9.5 for much more

Newton method rescales space



from Boyd & Vandenberghe

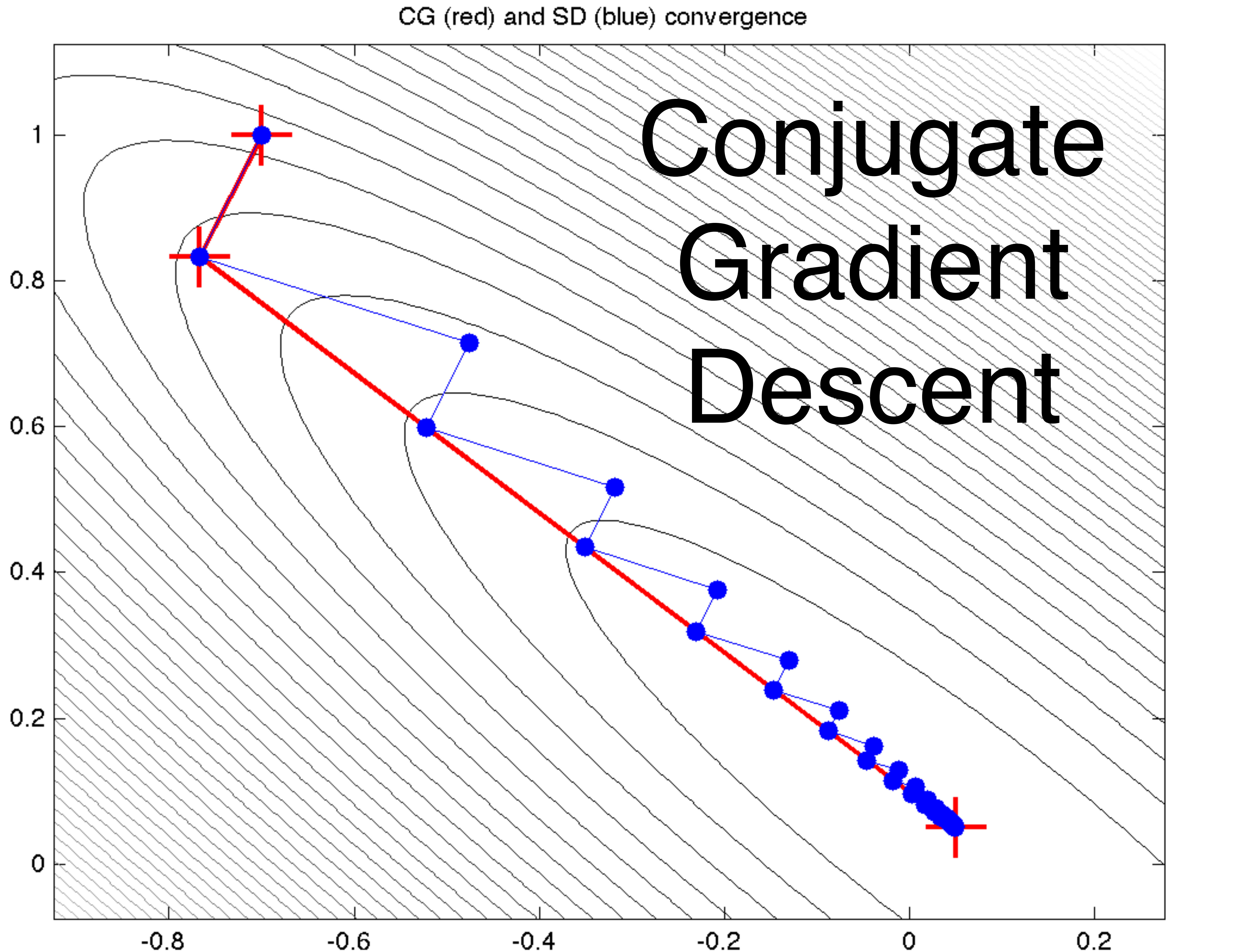
Newton method rescales space



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


CG (red) and SD (blue) convergence
Conjugate
Gradient
Descent
1
0.8
0.6
0.4
0.2
0
-0.8
-0.6
-0.4
-0.2
0
0.2

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

Proof

- m mutually K orthogonal vectors $x_i \in \mathbb{R}^m$
  - form a basis
  - allow expansion $$z = \sum_{i=1}^{m} x_i \frac{x_i^\top K z}{x_i^\top K x_i}$$
  - solve linear system $$z = \sum_{i=1}^{m} x_i \frac{x_i^\top y}{x_i^\top K x_i} \text{ for } y = Kz$$
- 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

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 vi are K orthogonal and show that xi+1 is optimal in span of {v1 .. vi}
- 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 vi+1 is K orthogonal to all vj (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

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)

5.3 Constrained Problems

5 Math and Optimization

Alexander Smola

Introduction to Machine Learning 10-701

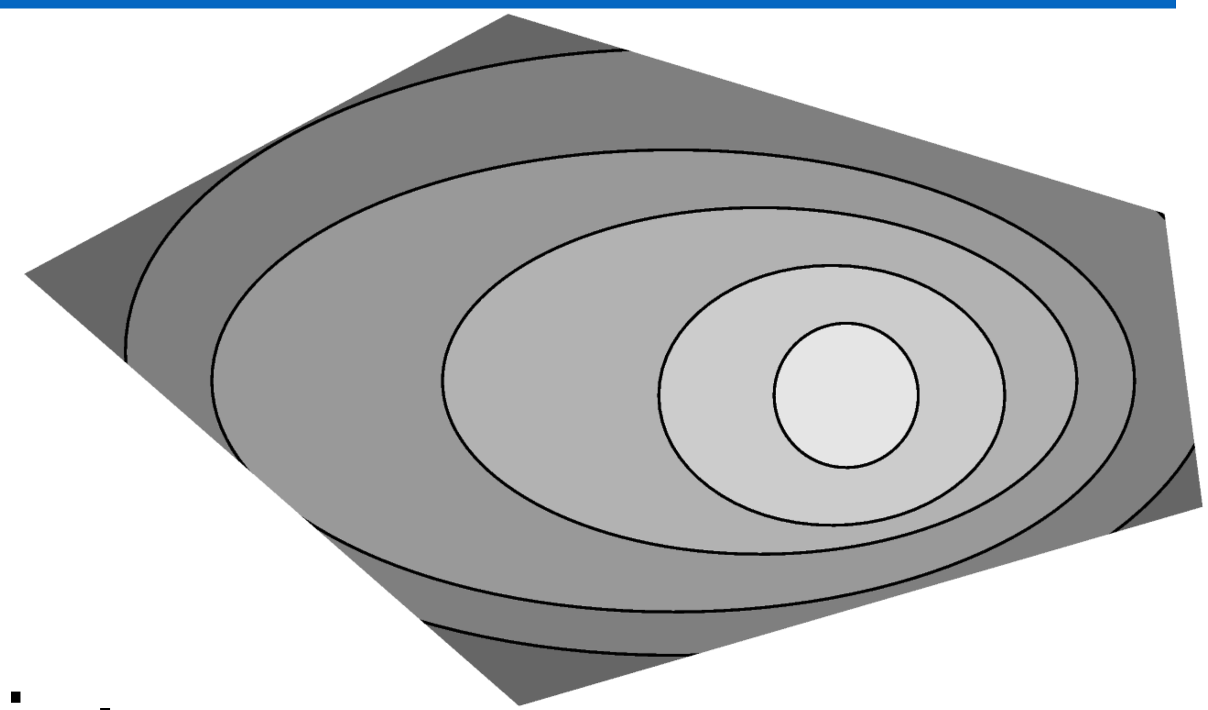
http://alex.smola.org/teaching/10-701-15

MACHINE LEARNING DEPARTMENT

Carnegie Mellon University

Basic Convexity

Constrained Convex Minimization

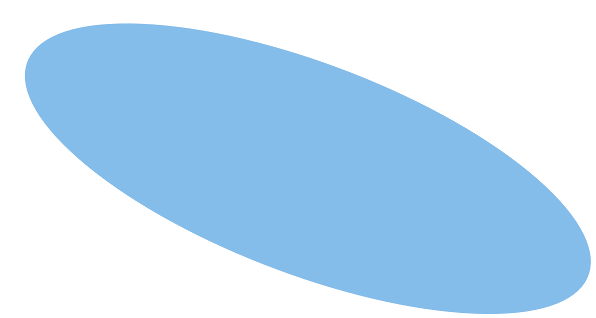


- Optimization problem

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

- 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

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

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

- Common constrain
  - linear inequality constraints

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

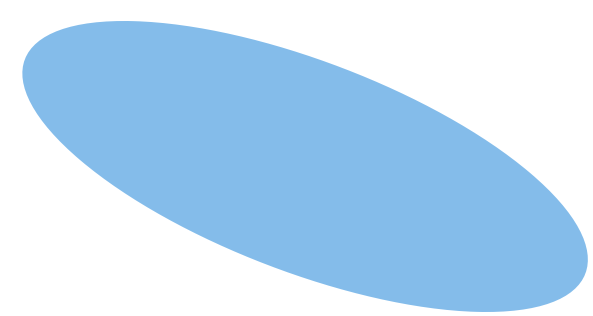
  - quadratic cone constraints

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

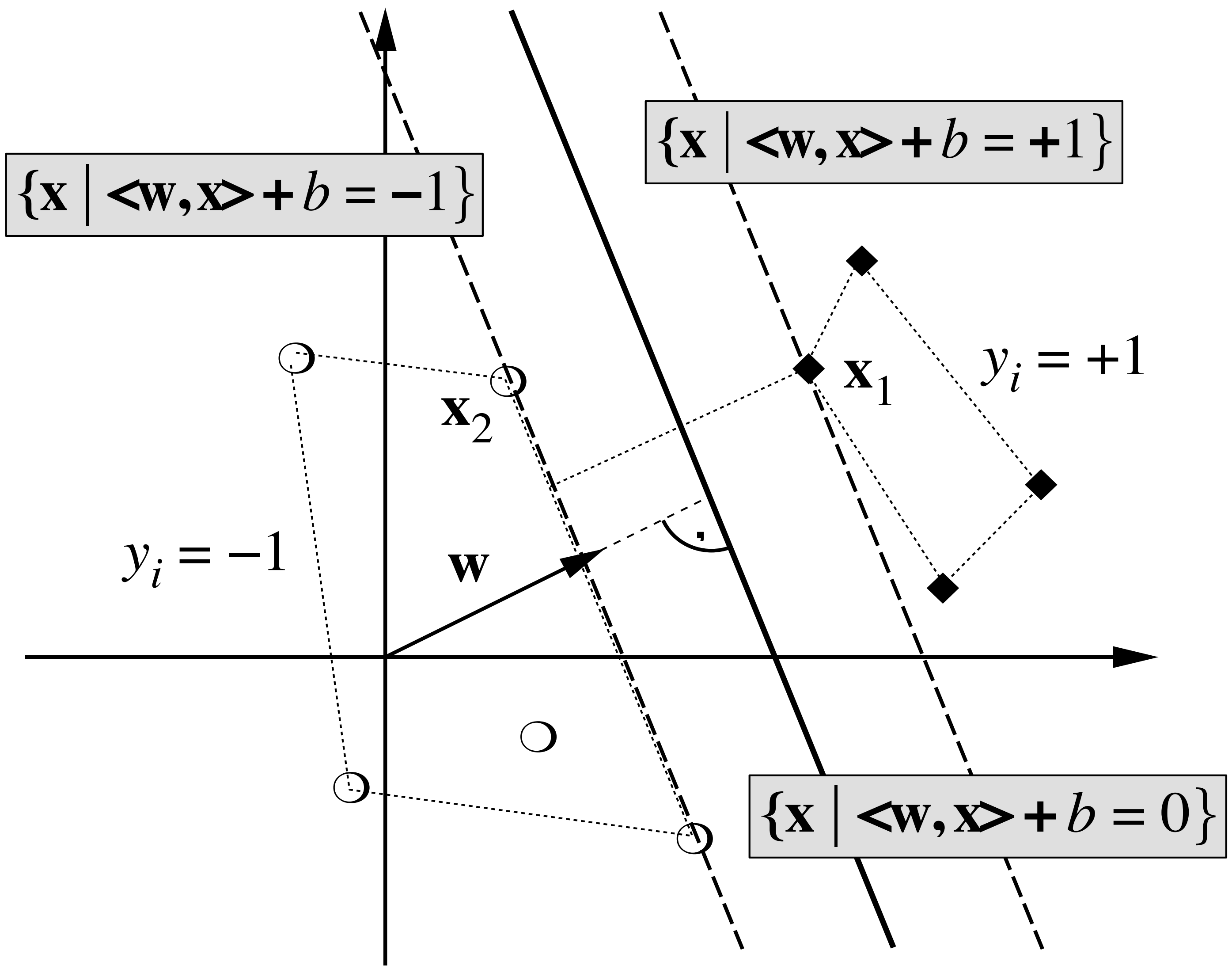
  - semidefinite constraints

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

Equality is special case
Why?



Example - Support Vectors



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

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

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

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

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 ci(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

$$
\begin{aligned}
\partial_x L(x^*, \alpha^*) = \partial_x f(x^*) + \sum_i \alpha_i^* \partial_x c_i(x^*) &= 0 \text{ (Saddlepoint in } x^*) \\
\partial_{\alpha_i} L(x^*, \alpha^*) = c_i(x^*) &\leq 0 \text{ (Saddlepoint in } \alpha^*) \\
\sum_i \alpha_i^* c_i(x^*) &= 0 \text{ (Vanishing KKT-gap)}
\end{aligned}
$$

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

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

- 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

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

Linear Programs

- Objective

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

- 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

- 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

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

plug into L

$$\partial_\alpha L(x, \alpha) = Ax + d \leq 0$$

$$0 = \alpha^\top (Ax + d)$$

$$0 \leq \alpha$$

- 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

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

- Lagrange function

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

- Optimality conditions

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

plug into L

$$\partial_\alpha L(x, \alpha) = Ax + d \leq 0$$

$$0 = \alpha^\top(Ax + d)$$

$$0 \leq \alpha$$

Quadratic Program

- Eliminating x from the Lagrangian via

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

- Lagrange function

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

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

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

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

- Dual

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

- 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

$$L(x,\alpha) = f(x) + \alpha^\top [Ax - b]$$
$$\partial_x L(x,\alpha) = \boxed{\partial_x f(x) + A^\top \alpha = 0}$$
$$\partial_\alpha L(x,\alpha) = \boxed{Ax - b = 0}$$

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

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

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

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

- 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

Some optimization problems

- Density estimation

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

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

- 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

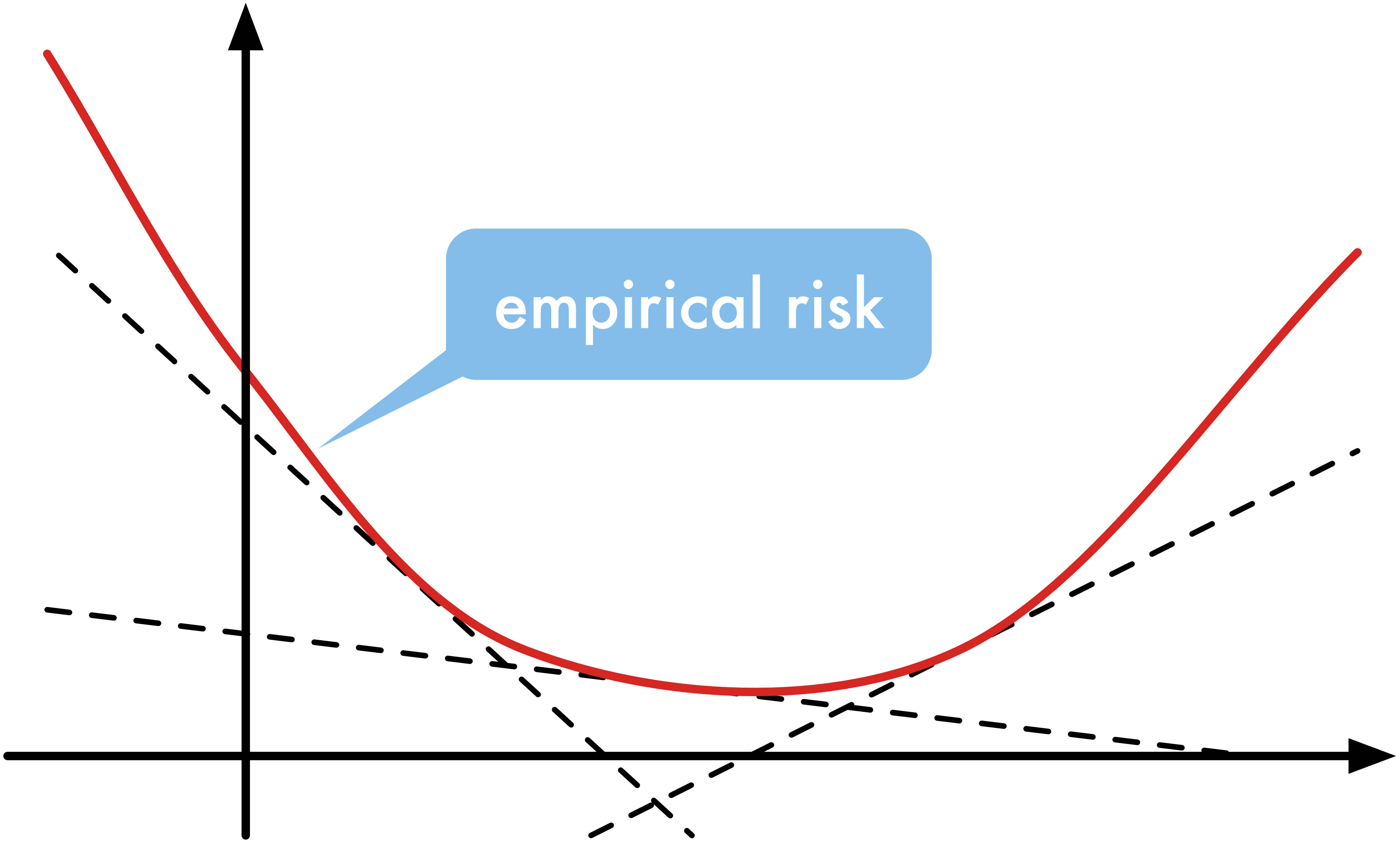
regularizer

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

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

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

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

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

Bundle Bound

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

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

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

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

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.

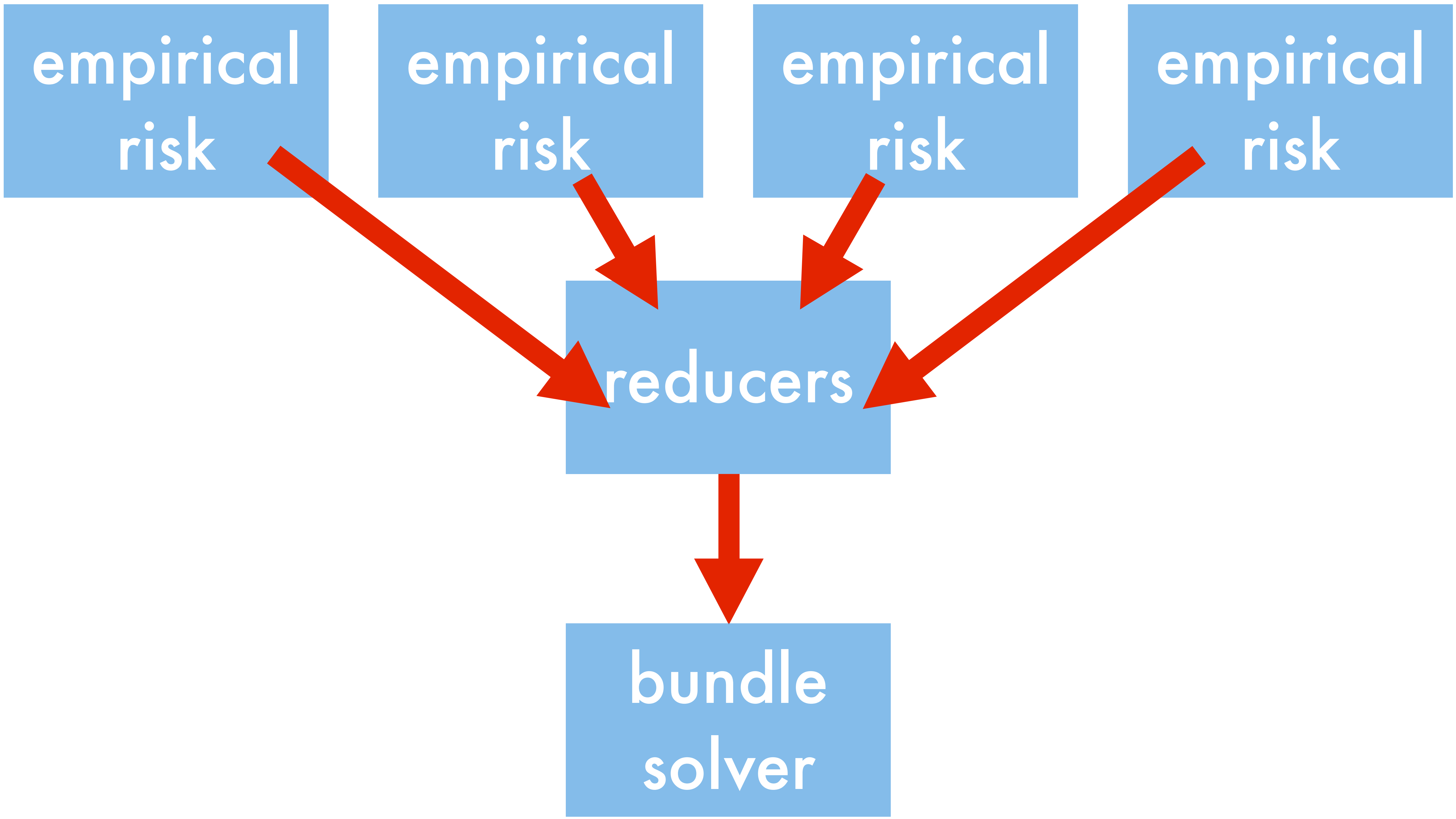
Loss independent of solver/regularizer

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.

Carnegie Mellon University

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

MACHINE LEARNING DEPARTMENT

5.4 Online Optimization

5 Math and Optimization

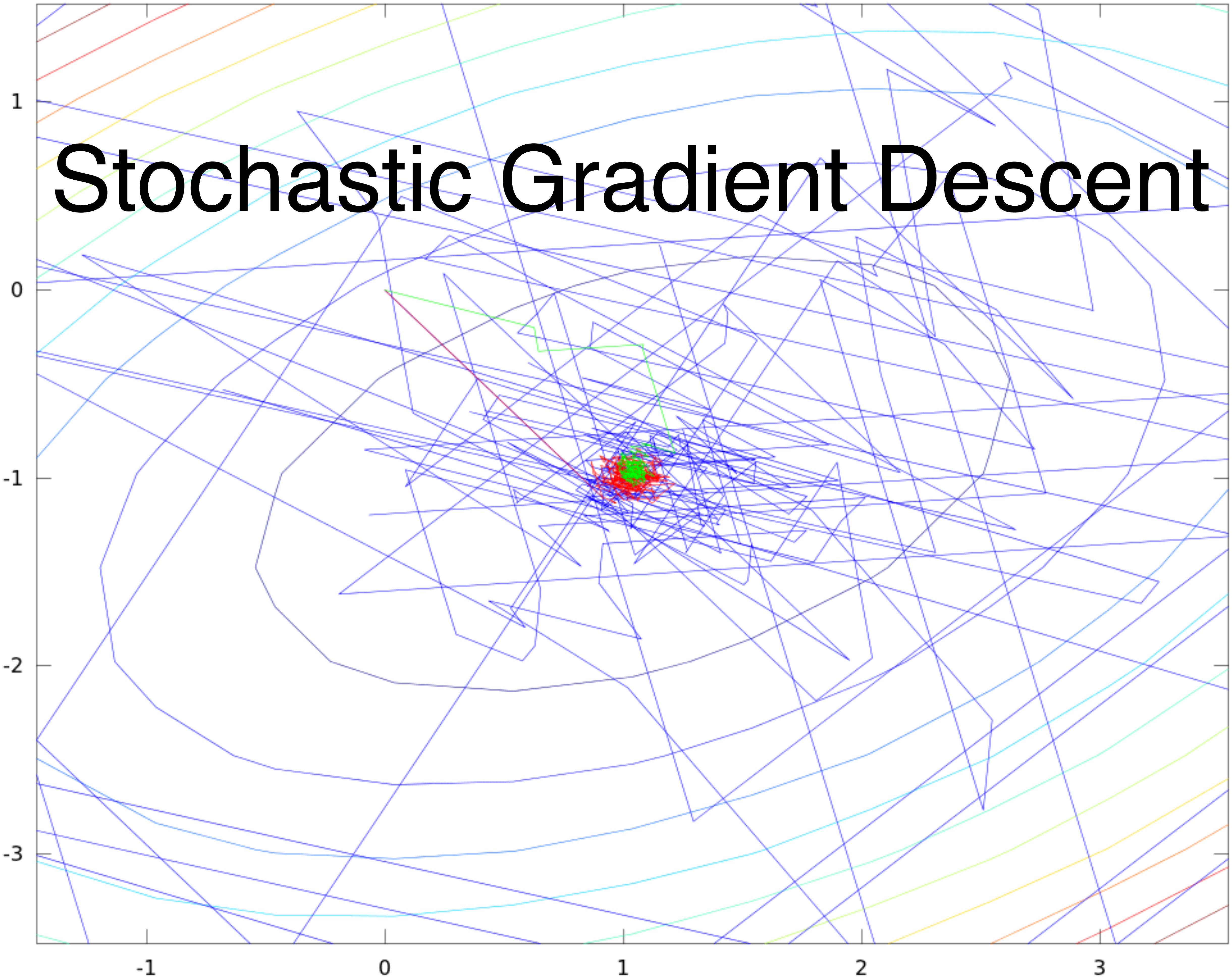
Alexander Smola

Introduction to Machine Learning 10-701

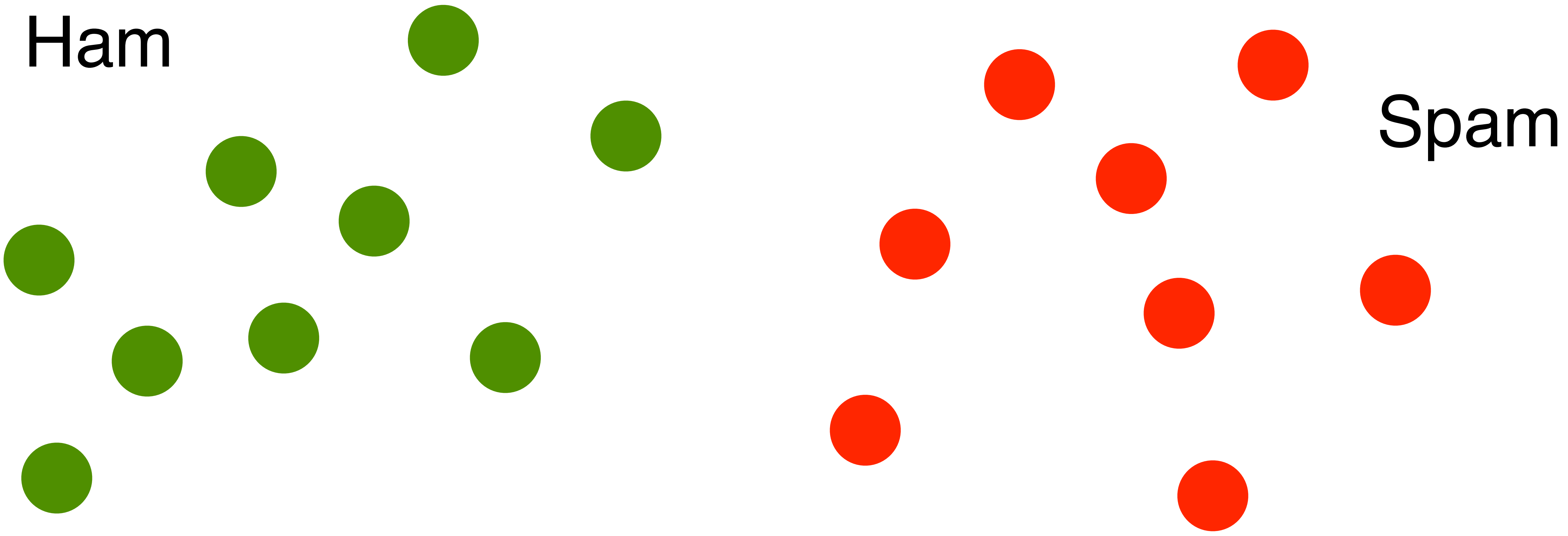
http://alex.smola.org/teaching/10-701-15

Carnegie Mellon University

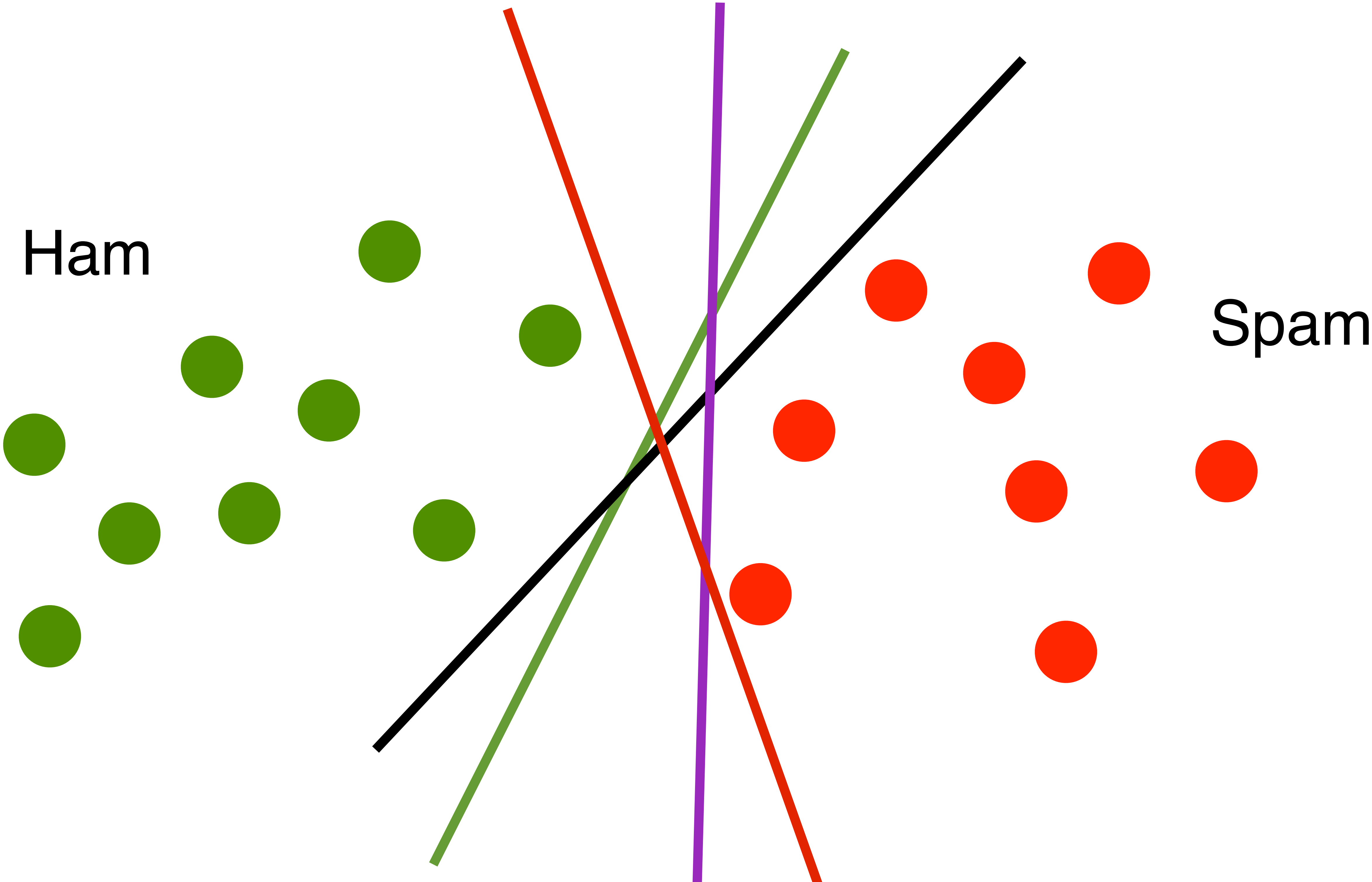
Stochastic Gradient Descent



Recall … Perceptron



Recall … Perceptron



The Perceptron

initialize $w = 0$ and $b = 0$
repeat
  if $y_i [\langle w, x_i \rangle + b] \leq 0$ **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 $$f(x) = \sum_{i \in I} \langle x_i, x \rangle + b$$
- This is SGD on the hinge loss

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

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

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

- Exponentially decaying averaging

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

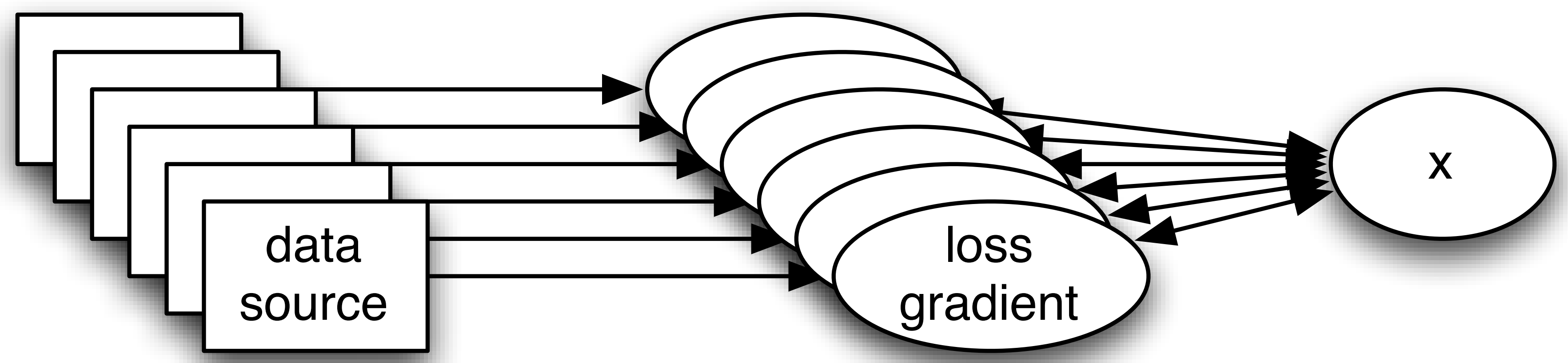
and plugging this into the discrepancy yields

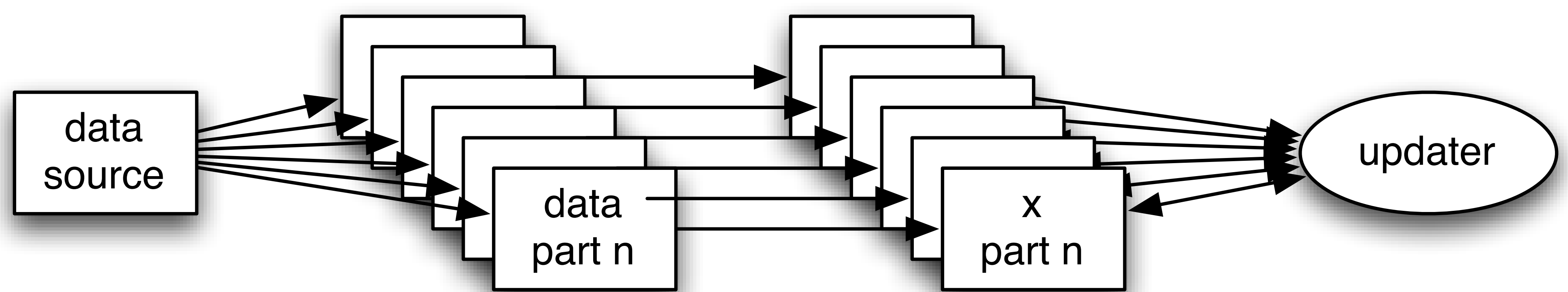
$$l(\bar{\theta}) - l^* \leq \frac{2L^2}{\lambda T} \log \left[1 + \frac{\lambda R T^{\frac{1}{2}}}{2L} \right] \text{ for } \eta = \frac{2}{\lambda T} \log \left[1 + \frac{\lambda R T^{\frac{1}{2}}}{2L} \right]$$

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



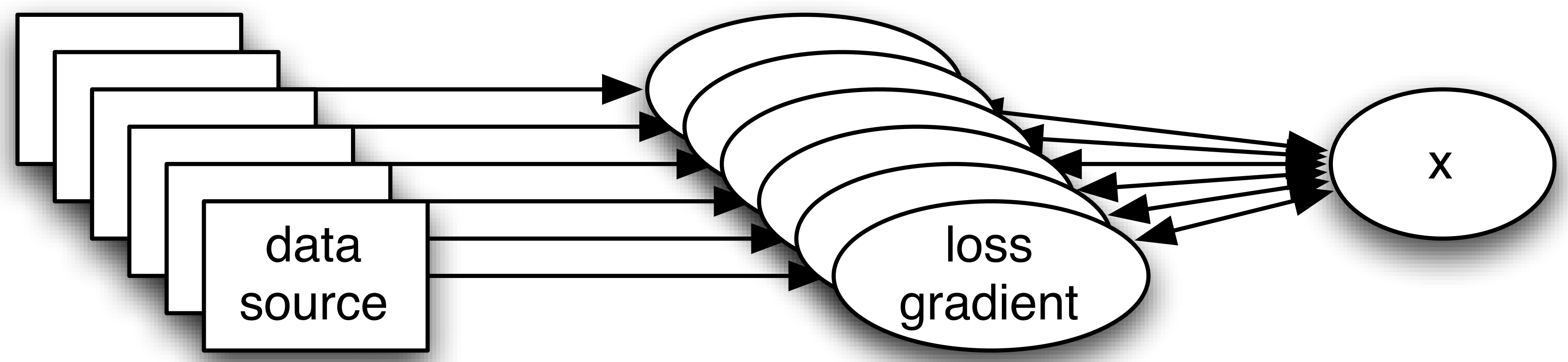


Online Learning

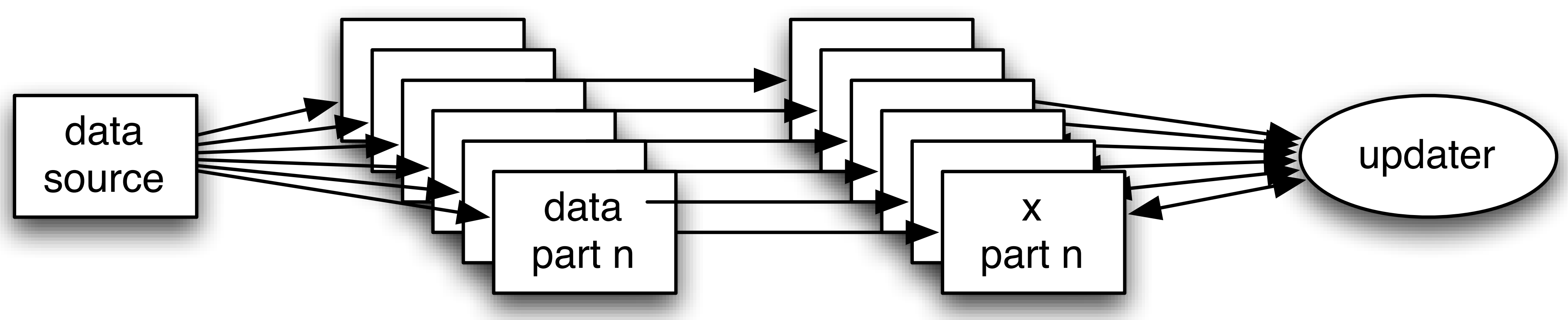
- **General Template**
  - Get instance
  - Compute instantaneous gradient
  - Update parameter vector
- **Problems**
  - **Sequential execution (single core)**
  - **CPU core speed is no longer increasing**
  - **Disk/network bandwidth: 300GB/h**
  - **Does not scale to TBs of data**
  - **Batch subgradient 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 l2 regularizer)

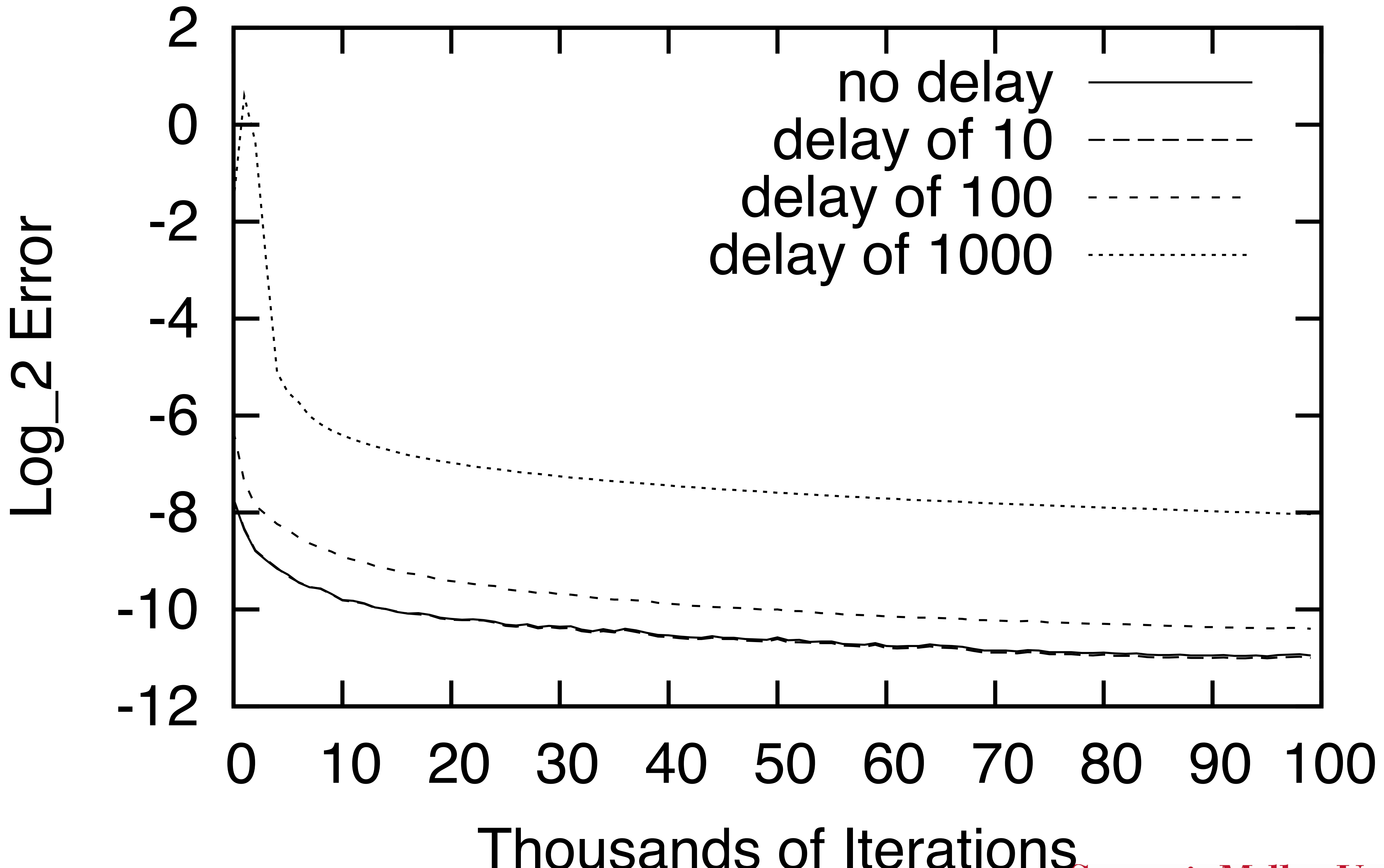
$$w \leftarrow w - \eta_t g(w, x_t) x_t$$

- General coordinate-based penalty

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

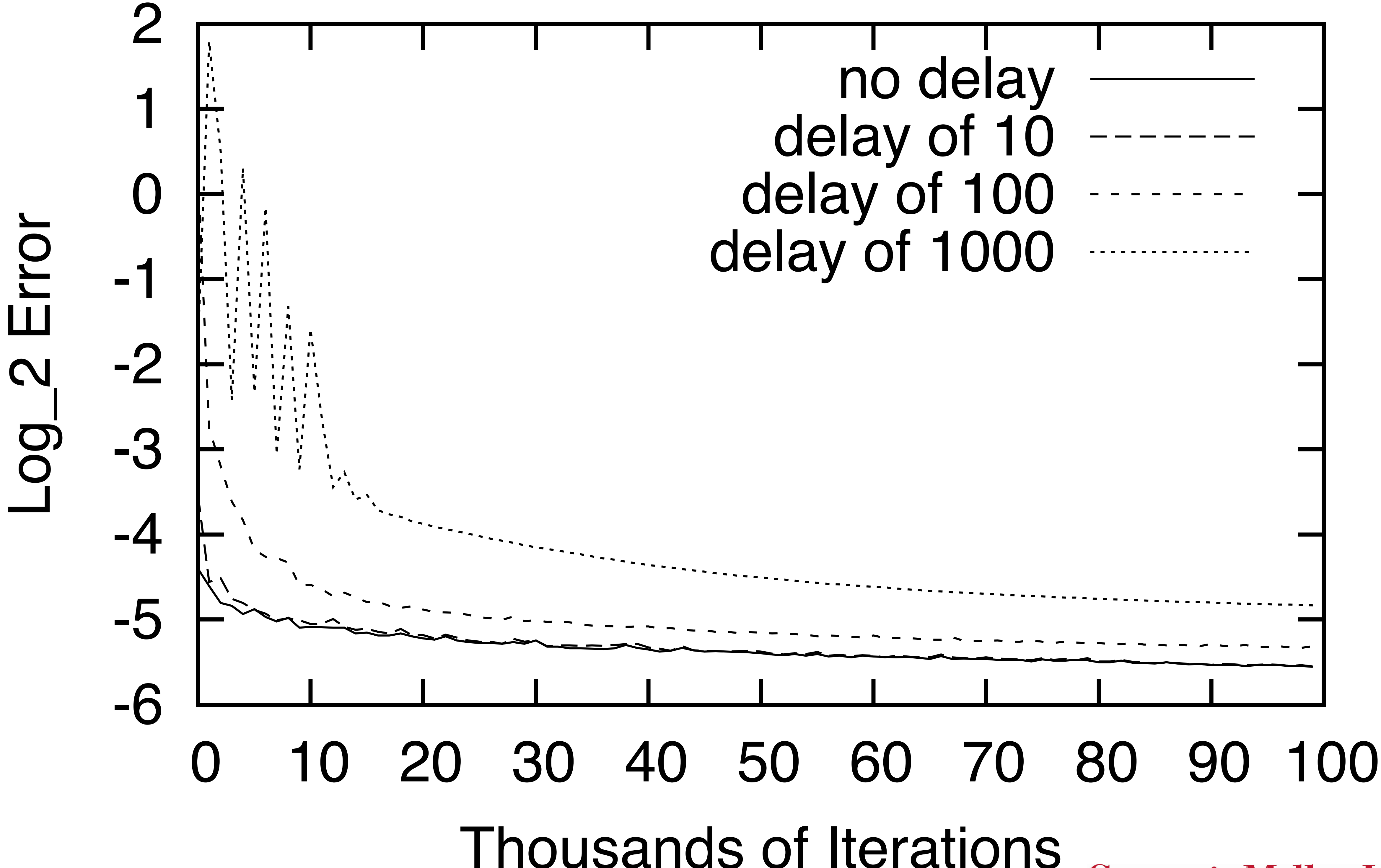
- Key insight - we only need to know the accurate value of wj whenever we use it
  - Store w_j with timestamp of last update
  - Before using w_j update using past stepsizes
  - Approximate sum over stepsizes by integral (Quadrianto et al, 2010; Li and Langford, 2009)

Convergence on TREC



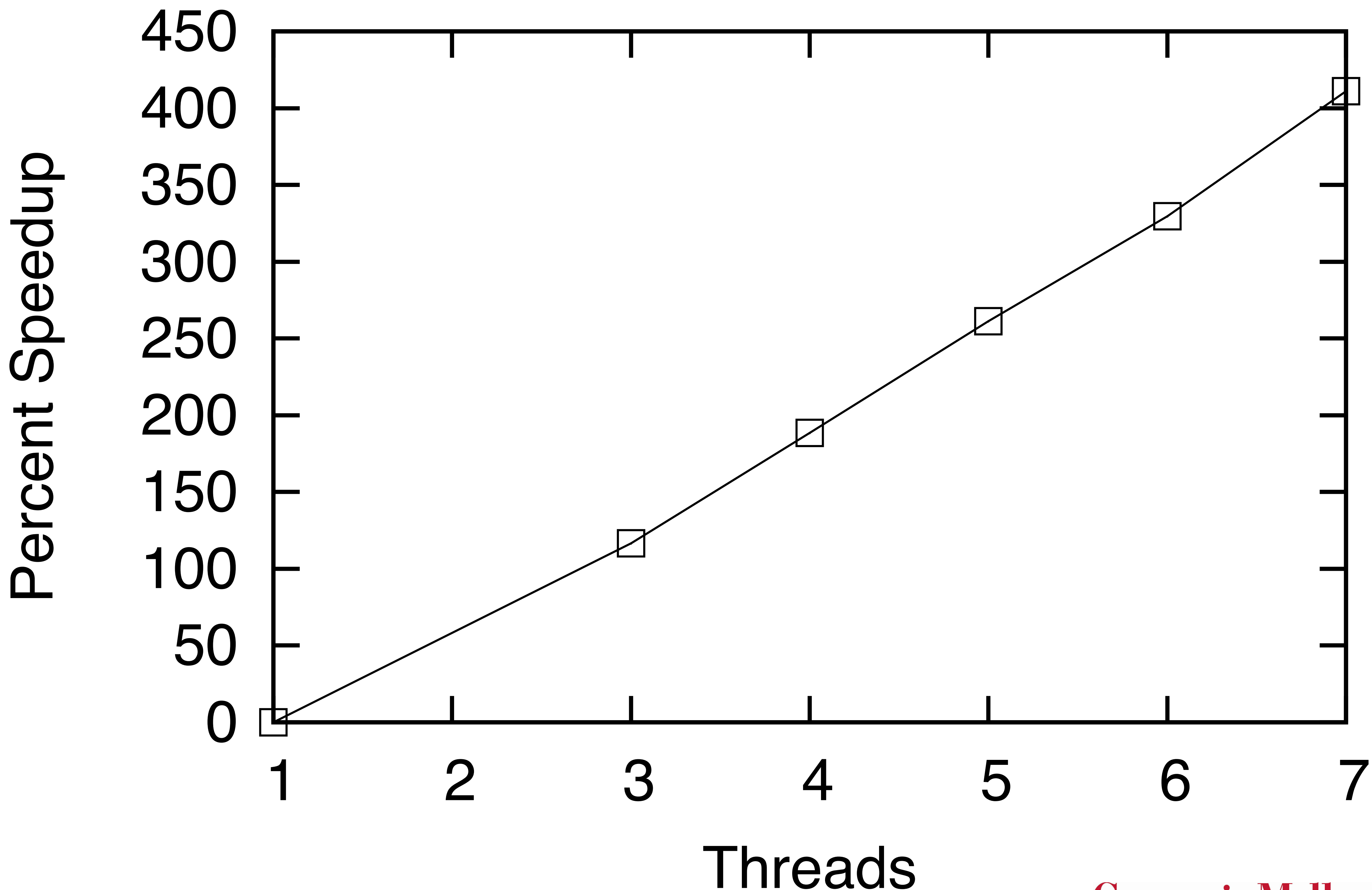
Carnegie Mellon University

Convergence on Y!Data



Carnegie Mellon University

Speedup on TREC



Carnegie Mellon University

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

$$T = \frac{\ln k - (\ln \eta + \ln \lambda)}{2\eta\lambda}$$

5.5 Discrete Problems

5 Math and Optimization

Alexander Smola

Introduction to Machine Learning 10-701

http://alex.smola.org/teaching/10-701-15

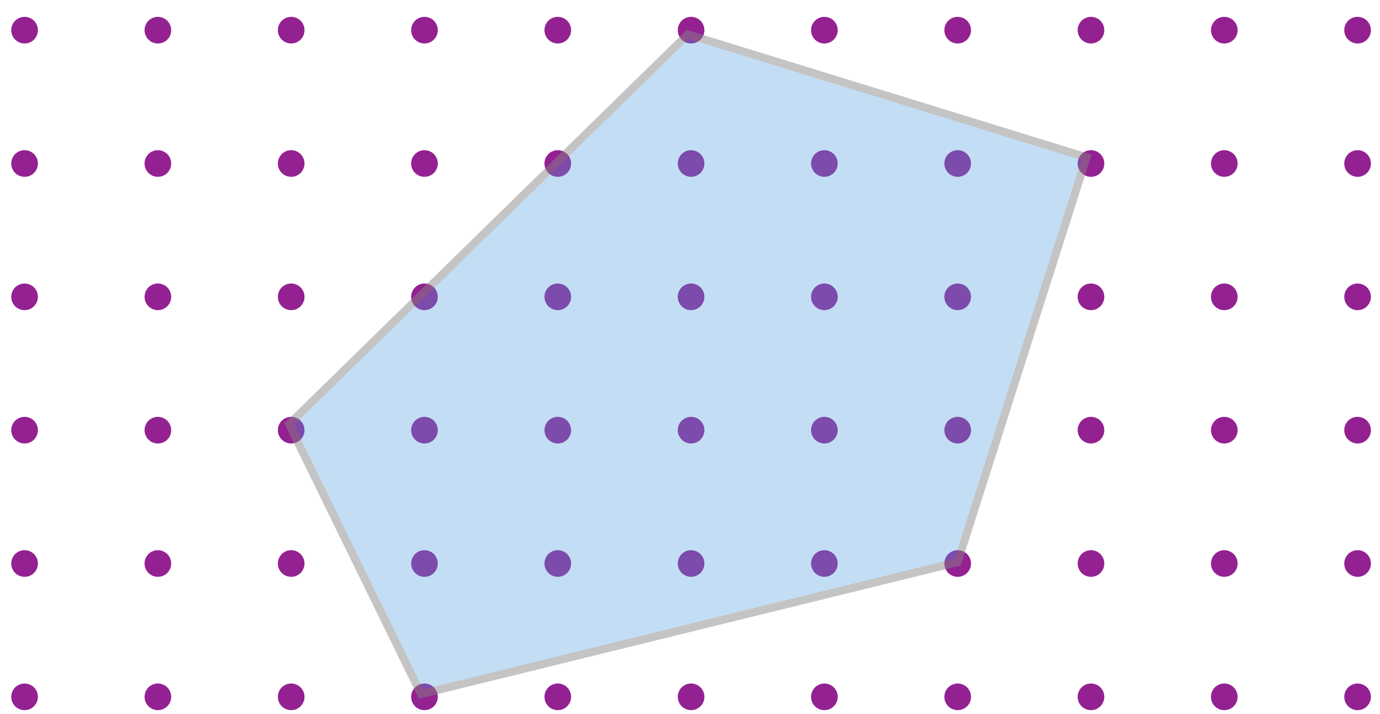
MACHINE LEARNING DEPARTMENT

Carnegie Mellon University

Integer programming relaxations

- Optimization problem

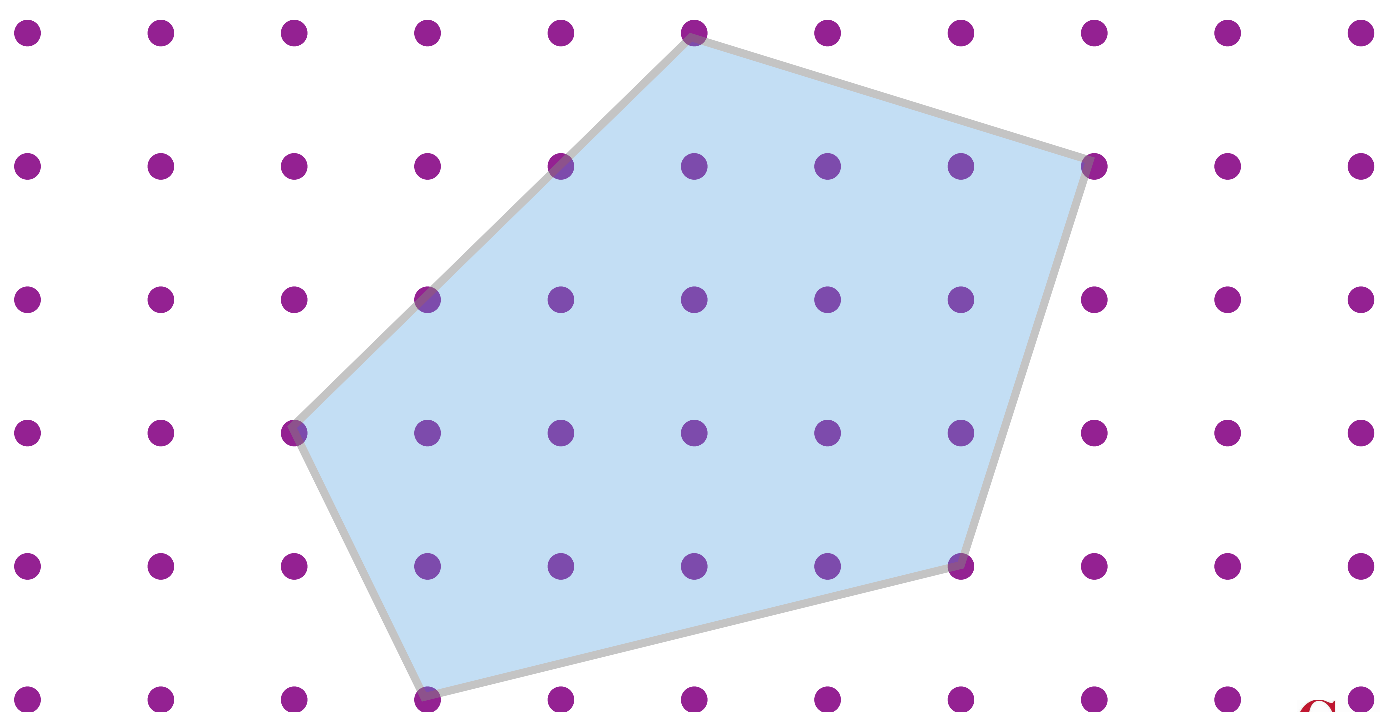
$$\underset{x}{\text{minimize}}\ c^\top x \text{ subject to } Ax \le b \text{ and } x \in \mathbb{Z}^n$$



- 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 cij 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

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