Optimization for the information and data sciences

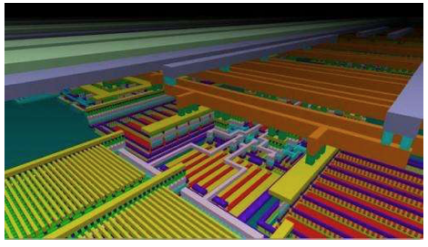
Mahdi Soltanolkotabi

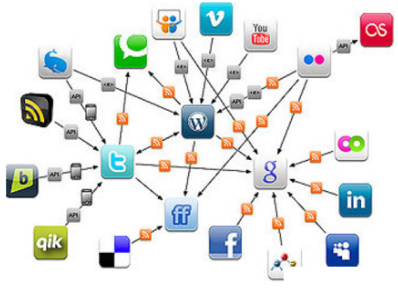
A look back

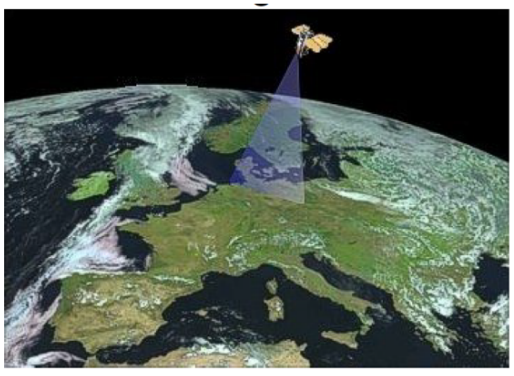
Ming Hsieh Department of Electrical Engineering

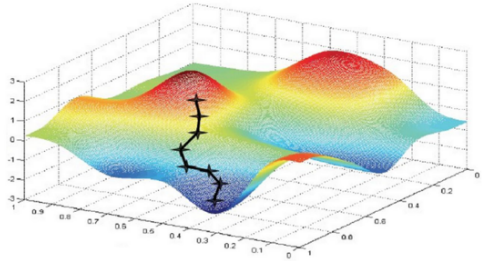
USC University of Southern California

Optimization at the heart of data science

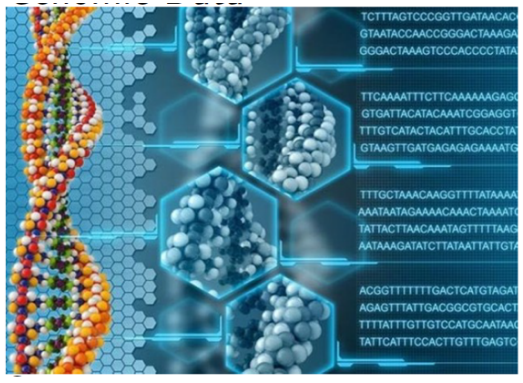


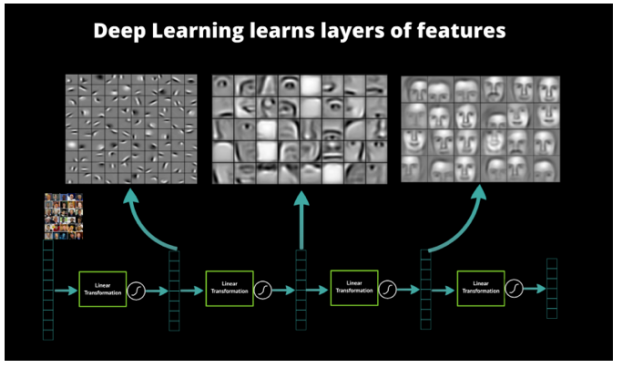


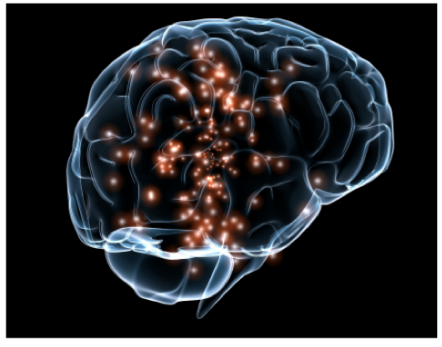


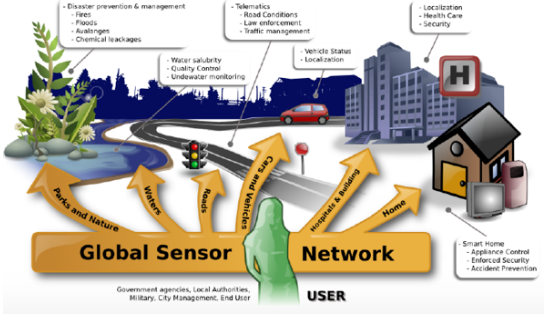


Optimization









Modeling

mathematical optimization

- problems in engineering design, data analysis and statistics, economics, management, . . . , can often be expressed as mathematical optimization problems
- techniques exist to take into account multiple objectives or uncertainty in the data

tractability

- roughly speaking, tractability in optimization requires convexity
- algorithms for nonconvex optimization find local (suboptimal) solutions, or are very expensive
- surprisingly many applications can be formulated as convex problems

Theoretical consequences of convexity

- local optima are global
- extensive duality theory
  - systematic way of deriving lower bounds on optimal value
  - necessary and sufficient optimality conditions
  - certificates of infeasibility
  - sensitivity analysis
- solution methods with polynomial worst-case complexity theory (with self-concordance)

Practical consequences of convexity

(most) **convex problems can be solved globally and efficiently**

- interior-point methods require 20 – 80 steps in practice
- basic algorithms (e.g., Newton, barrier method, . . .) are easy to implement and work well for small and medium size problems (larger problems if structure is exploited)
- more and more high-quality implementations of advanced algorithms and modeling tools are becoming available
- high level modeling tools like `cvx` ease modeling and problem specification

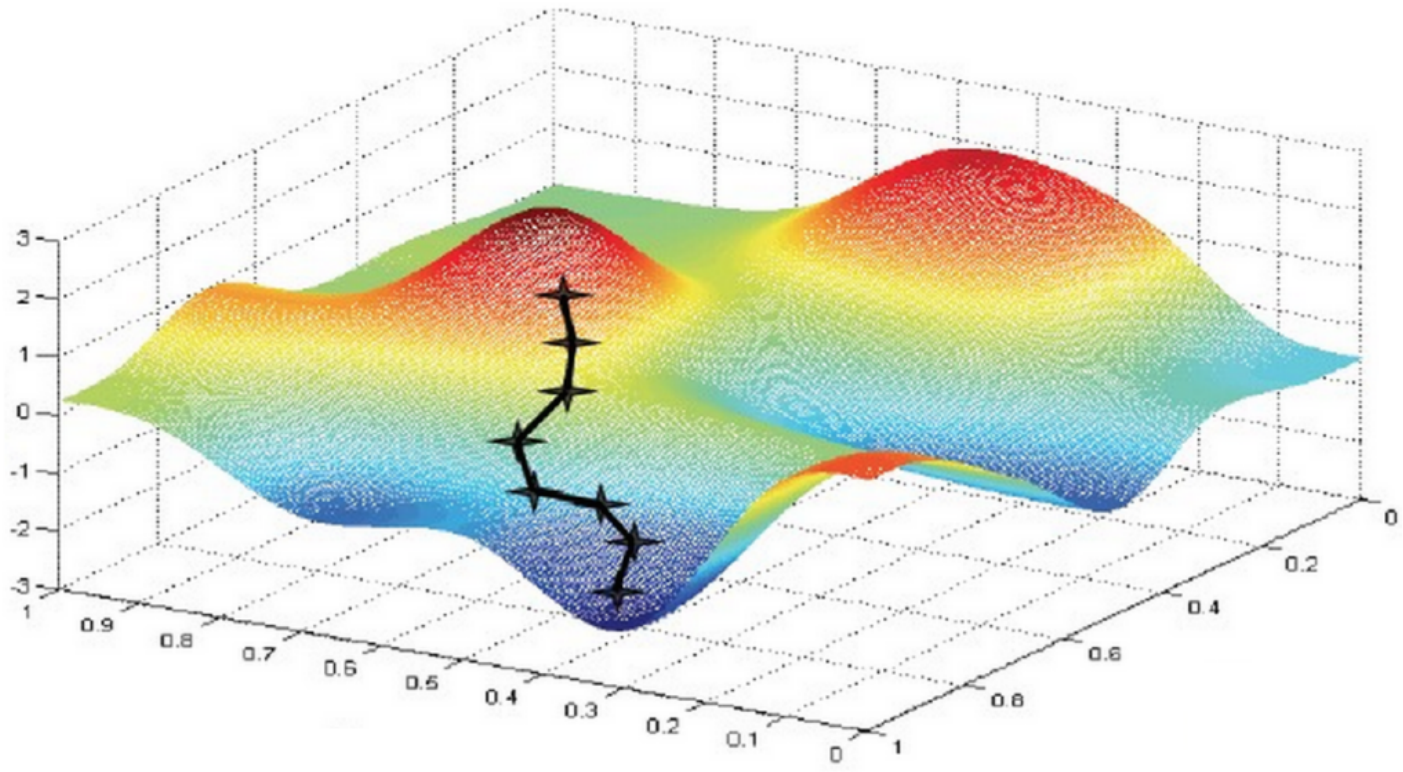
How to use convex optimization

to use convex optimization in some applied context

- use rapid prototyping, approximate modeling
  - start with simple models, small problem instances, inefficient solution methods
  - if you don't like the results, no need to expend further effort on more accurate models or efficient algorithms
- work out, simplify, and interpret optimality conditions and dual
- even if the problem is quite nonconvex, you can use convex optimization
  - in subproblems, e.g., to find search direction
  - by repeatedly forming and solving a convex approximation at the current point

Algorithms

First Order Methods



Gradient method

to minimize a convex differentiable function f: choose initial point $x^{(0)}$ and repeat

$$x^{(k)} = x^{(k-1)} - t_k \nabla f(x^{(k-1)}), \qquad k = 1, 2, \ldots$$

Step size rules

- fixed: t_k constant
- backtracking line search
- exact line search: minimize $f(x - t\nabla f(x))$ over t

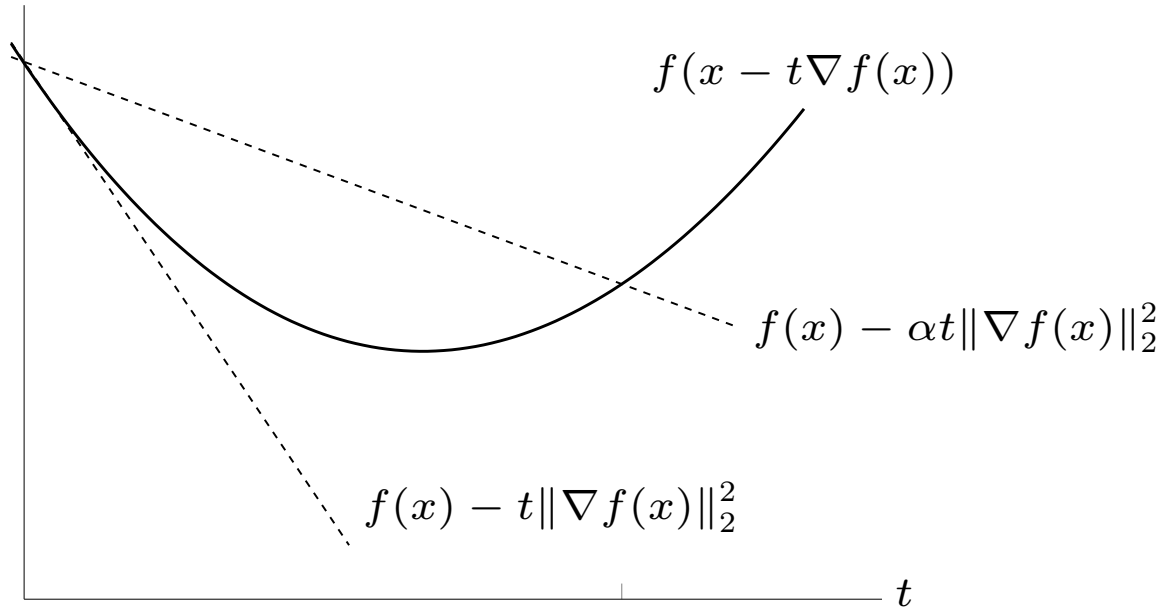
Advantages of gradient method

- every iteration is inexpensive
- does not require second derivatives

Backtracking line search

initialize t_k at $\hat{t} > 0$ (for example, $\hat{t} = 1$); take $t_k := \beta t_k$ until

$$f(x - t_k \nabla f(x)) < f(x) - \alpha t_k \|\nabla f(x)\|_2^2$$



$0 < \beta < 1$; we will take $\alpha = 1/2$ (mostly to simplify proofs)

acceleration/multistep

gradient method akin to an ODE

$$x[k+1] = x[k] - \alpha \nabla f(x[k])$$

$$\dot{x} = -\nabla f(x)$$

to prevent oscillation, add a second order term

$$\ddot{x} = -b\dot{x} - \nabla f(x)$$

$$x[k+1] = x[k] - \alpha \nabla f(x[k]) + \beta(x[k] - x[k-1])$$

$$x[k+1] = x[k] + \alpha p[k]$$

$$p[k] = -\nabla f(x[k]) + \beta p[k-1]$$

heavy ball method (constant **α**,**β**)

when *f* is quadratic, this is *Chebyshev's iterative method*

optimal method

Nesterov's optimal method (1983,2004)

$$\alpha_k = \frac{1}{L}$$

$$x[k+1] = x[k] + \alpha_k p[k]$$

$$p[k] = -\nabla f(x[k] + \beta_k(x[k] - x[k-1])) + \beta_k p[k-1]$$

heavy ball with *extragradient step*

$$\lambda_{k+1}^2 = (1-\lambda_{k+1})\lambda_k^2 + \kappa^{-1}\lambda_{k+1}$$

$$\beta_k = \frac{\lambda_k(1-\lambda_k)}{\lambda_k^2 + \lambda_{k+1}}$$

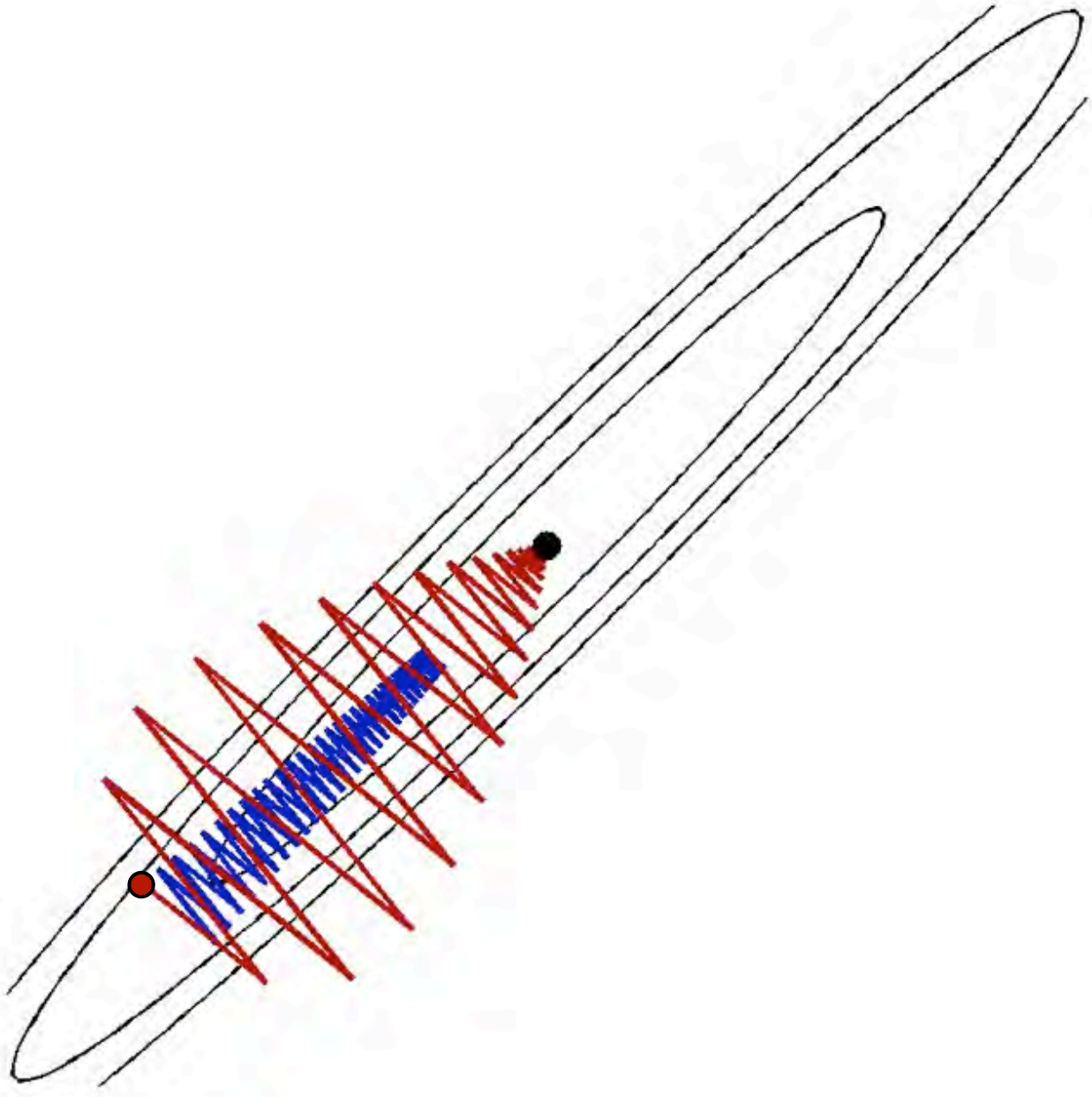
$$t_k = \tfrac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$$

$$\beta_k = \frac{t_k - 1}{t_{k+1}}$$

FISTA (Beck and Teboulle 2007)

$$\beta_k = \frac{k-1}{k+2}$$

- Recent fixes use line search to find parameters and still achieve optimal rate (modulo log factors)
- Analysis based on *estimate sequences*, using simple quadratic approximations to *f*



Fletcher-Reeves CG algorithm

CG algorithm of page 3-12 modified to minimize non-quadratic convex f

Initialize: choose $x^{(0)}$

For $k = 1, 2, \ldots$

1. if $k = 1$, take $p_1 = -\nabla f(x^{(0)})$; otherwise, take

$$p_k = -\nabla f(x^{(k-1)}) + \beta_k p_{k-1} \quad \text{where} \quad \beta_k = \frac{\|\nabla f(x^{(k-1)})\|_2^2}{\|\nabla f(x^{(k-2)})\|_2^2}$$

2. update $x^{(k)} = x^{(k-1)} + \alpha_k p_k$ where

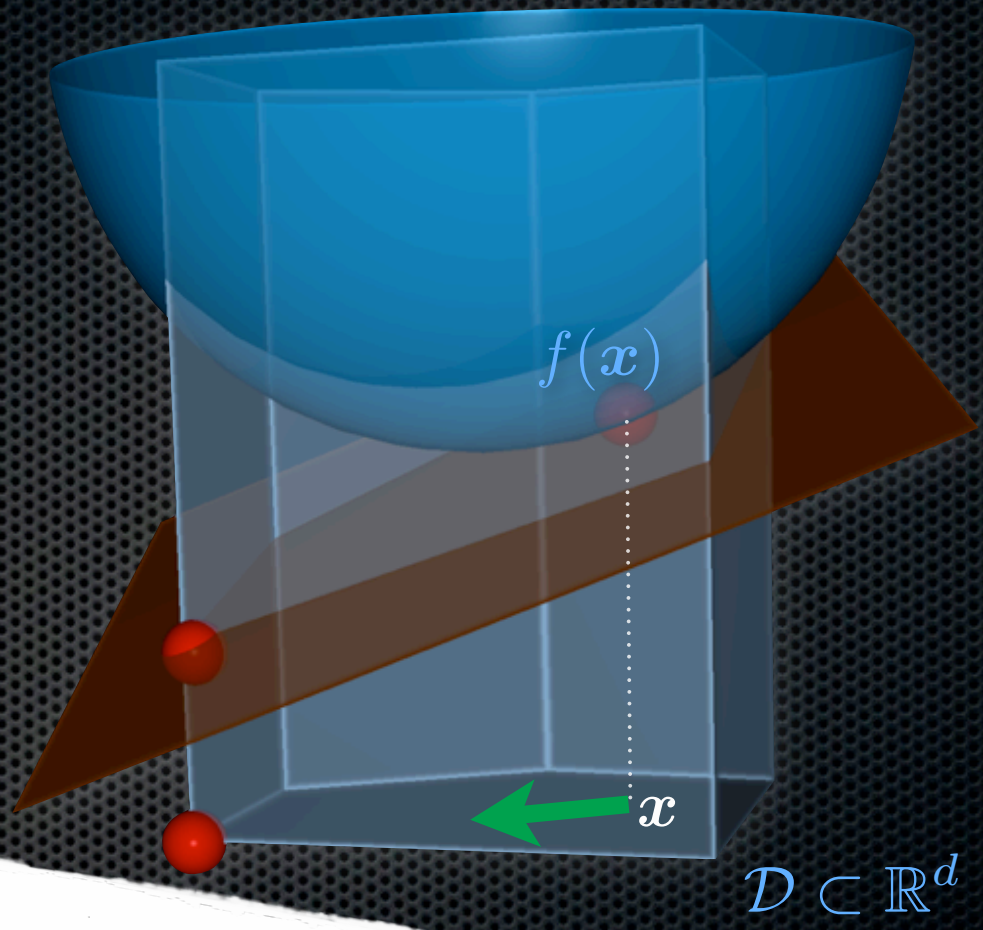
$$\alpha_k = \operatorname*{argmin}_{\alpha} f(x^{(k-1)} + \alpha p_k)$$

if $\nabla f(x^{(k)})$ is sufficiently small, return $x^{(k)}$

Frank-Wolfe Algorithm

"Conditional Gradient Method"

"Reduced Gradient Method"



AN ALGORITHM FOR QUADRATIC PROGRAMMING

Marguerite Frank and Philip Wolfe[1]
Princeton University

1956

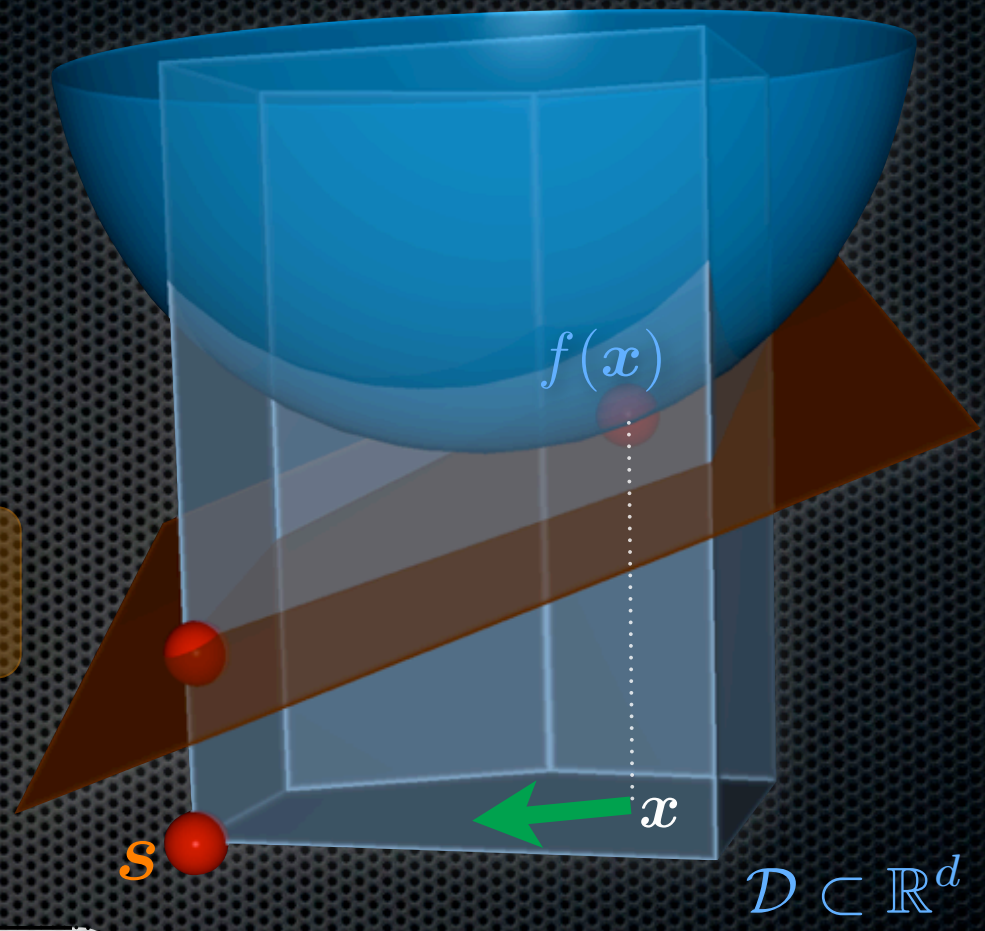
A finite iteration method for calculating the solution of quadratic programming problems is described. Extensions to more general non-linear problems are suggested.

1. INTRODUCTION

The problem of maximizing a concave quadratic
linear inequality constraints has
putational side and

The Linearized Problem

$$\min_{s' \in \mathcal{D}} f(x) + \langle s' - x, \nabla f(x) \rangle$$



Algorithm 1 Frank-Wolfe

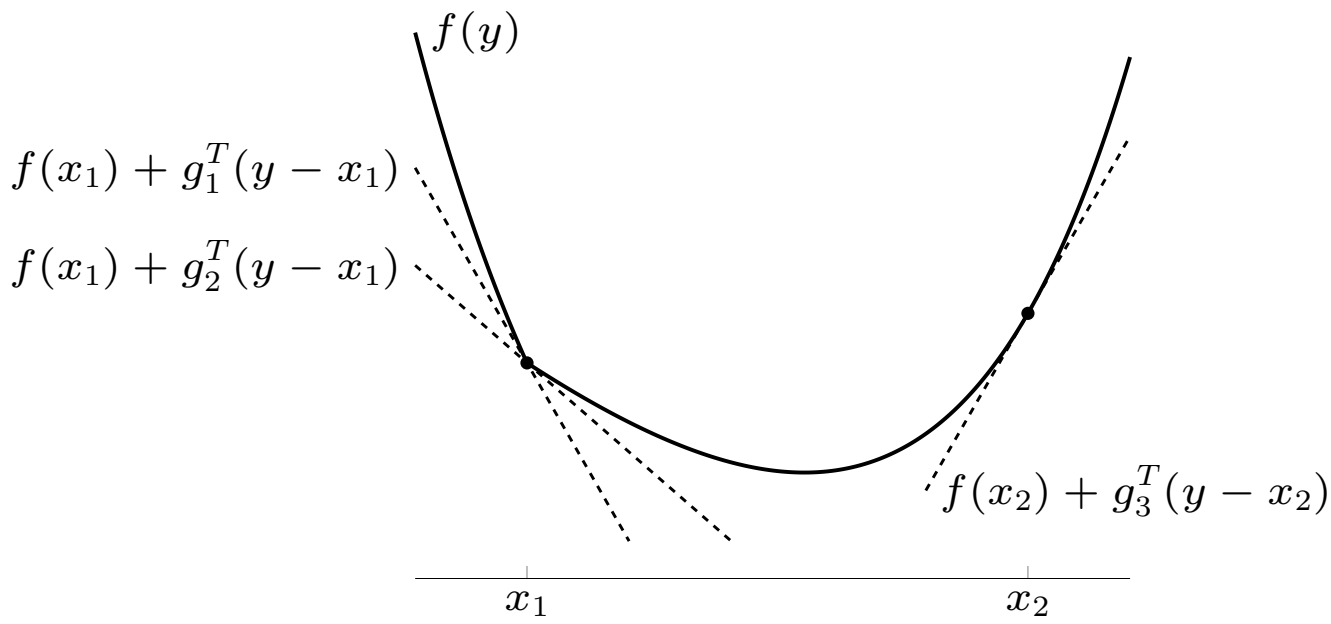
Let $x^{(0)} \in \mathcal{D}$
for $k = 0 \dots K$ **do**
  Compute $s := \arg\min_{s' \in \mathcal{D}} \langle s', \nabla f(x^{(k)}) \rangle$
  Let $\gamma := \frac{2}{k+2}$
  Update $x^{(k+1)} := (1 - \gamma) x^{(k)} + \gamma s$
end for

Non-differentiable stuff

Subgradient

g is a **subgradient** of a convex function f at $x \in \operatorname{dom} f$ if

$$f(y) \geq f(x) + g^T(y - x) \quad \forall y \in \operatorname{dom} f$$



g_1, g_2 are subgradients at x_1; g_3 is a subgradient at x_2

Subgradient method

to minimize a nondifferentiable convex function f: choose $x^{(0)}$ and repeat

$$x^{(k)} = x^{(k-1)} - t_k g^{(k-1)}, \quad k = 1, 2, \ldots$$

$g^{(k-1)}$ is any subgradient of f at $x^{(k-1)}$

Step size rules

- fixed step: t_k constant
- fixed length: $t_k \|g^{(k-1)}\|_2 = \|x^{(k)} - x^{(k-1)}\|_2$ is constant
- diminishing: $t_k \to 0$, $\sum_{k=1}^{\infty} t_k = \infty$

Proximal gradient method

unconstrained optimization with objective split in two components

$$\text{minimize} \quad f(x) = g(x) + h(x)$$

- g convex, differentiable, $\operatorname{dom} g = \mathbf{R}^n$
- h convex with inexpensive prox-operator (many examples in lecture 8)

Proximal gradient algorithm

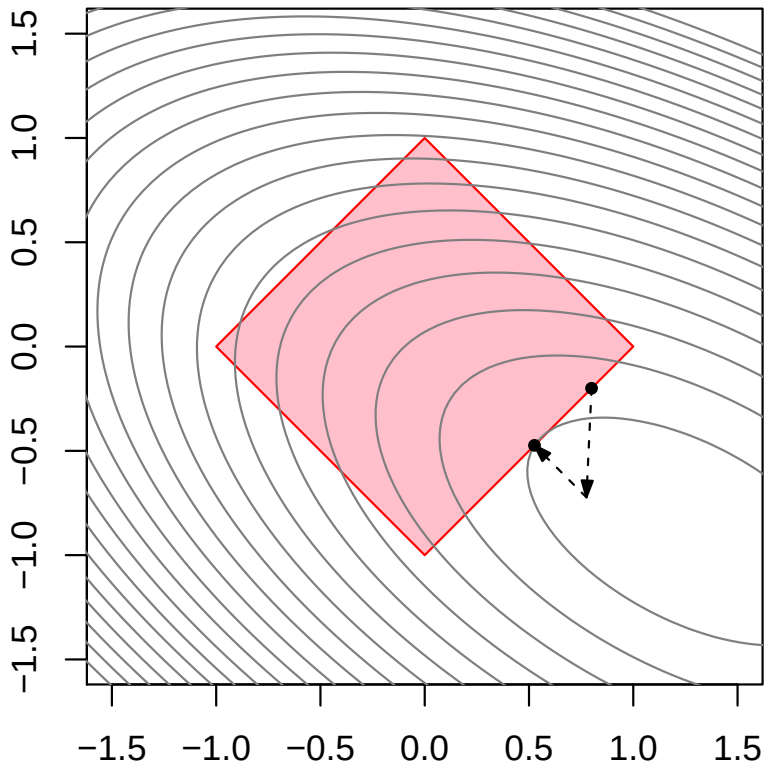
$$x^{(k)} = \operatorname{prox}_{t_k h}\left(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\right)$$

- $t_k > 0$ is step size, constant or determined by line search
- can start at infeasible $x^{(0)}$ (however $x^{(k)} \in \operatorname{dom} f = \operatorname{dom} h$ for $k \geq 1$)

Therefore proximal gradient update step is:

$$x^+ = P_C\big(x - t\nabla g(x)\big)$$

i.e., perform usual gradient update and then project back onto C. Called projected gradient descent



Stochastic and Incremental Algorithms

stochastic gradient

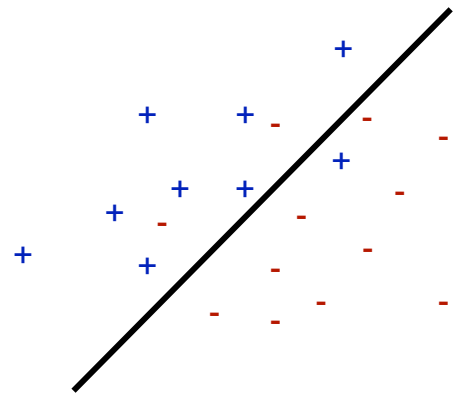
$$\text{minimize} \quad \mathbb{E}_{\xi}[f(x, \xi)]$$

Stochastic Gradient Descent:

For each k, sample ξ_k and compute $x[k+1] = x[k] - \alpha_k \nabla_x f(x[k], \xi_k)$

- Robbins and Monro (1950)
- Adaptive Filtering (1960s-1990s)
- Back Propagation in Neural Networks (1980s)
- Online Learning, Stochastic Approximation (2000s)

Support Vector Machines



cancer vs other illness
fraud vs normal purchase
up-going vs down-going muons

$$\text{minimize} \quad \sum_{i=1}^{n} \max(1 - y_i x^T z_i, 0) + \lambda \|x\|_2^2$$

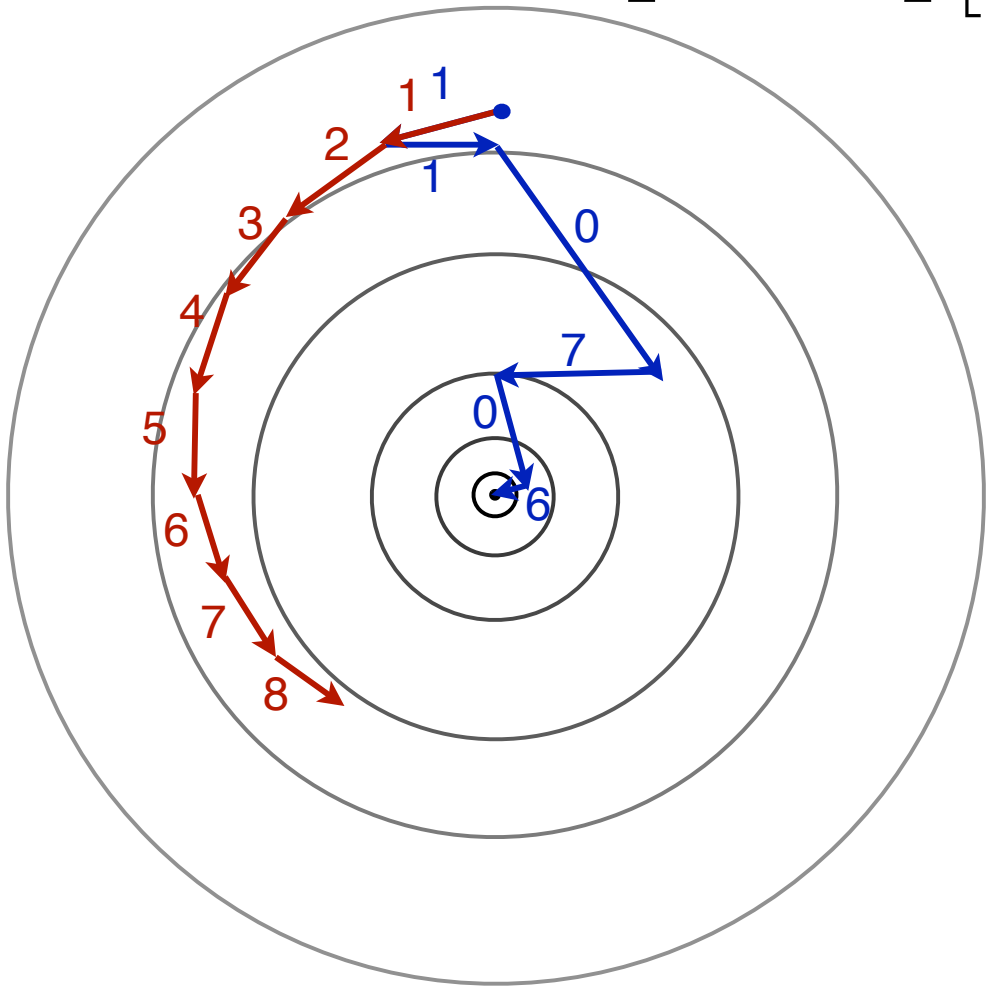
$$\text{minimize}_x \sum_{i=1}^{n} \left(\max(1 - y_i x^T z_i, 0) + \frac{\lambda}{n} \|x\|^2 \right)$$

- Step 1: Pick i and compute the sign of the assignment: $\hat{y}_i = \text{sign}(x^T z_i)$
- Step 2: If $\hat{y}_i \neq y_i$, $x \leftarrow (1 - \frac{\alpha\lambda}{n})x + \alpha y_i z_i$

$$\text{minimize} \quad \sum_{k=0}^{9} \left(\cos\left(\tfrac{\pi k}{10}\right) x_1 + \sin\left(\tfrac{\pi k}{10}\right) x_2\right)^2 = 5x_1^2 + 5x_2^2$$

Stepsize = 1/2

$$x - \frac{1}{2}\nabla f_j(x) = \frac{1}{2}\begin{bmatrix} 1 - c_j & -s_j \\ -s_j & 1 + c_j \end{bmatrix} x$$



Choose directions in order

Choose a direction uniformly with replacement

Second Order Methods

Newton's method

Consider the unconstrained, smooth convex optimization problem

$$\min_x \ f(x)$$

where f is convex, twice differentable, and $\mathrm{dom}(f) = \mathbb{R}^n$.

Newton's method: choose initial $x^{(0)} \in \mathbb{R}^n$, and

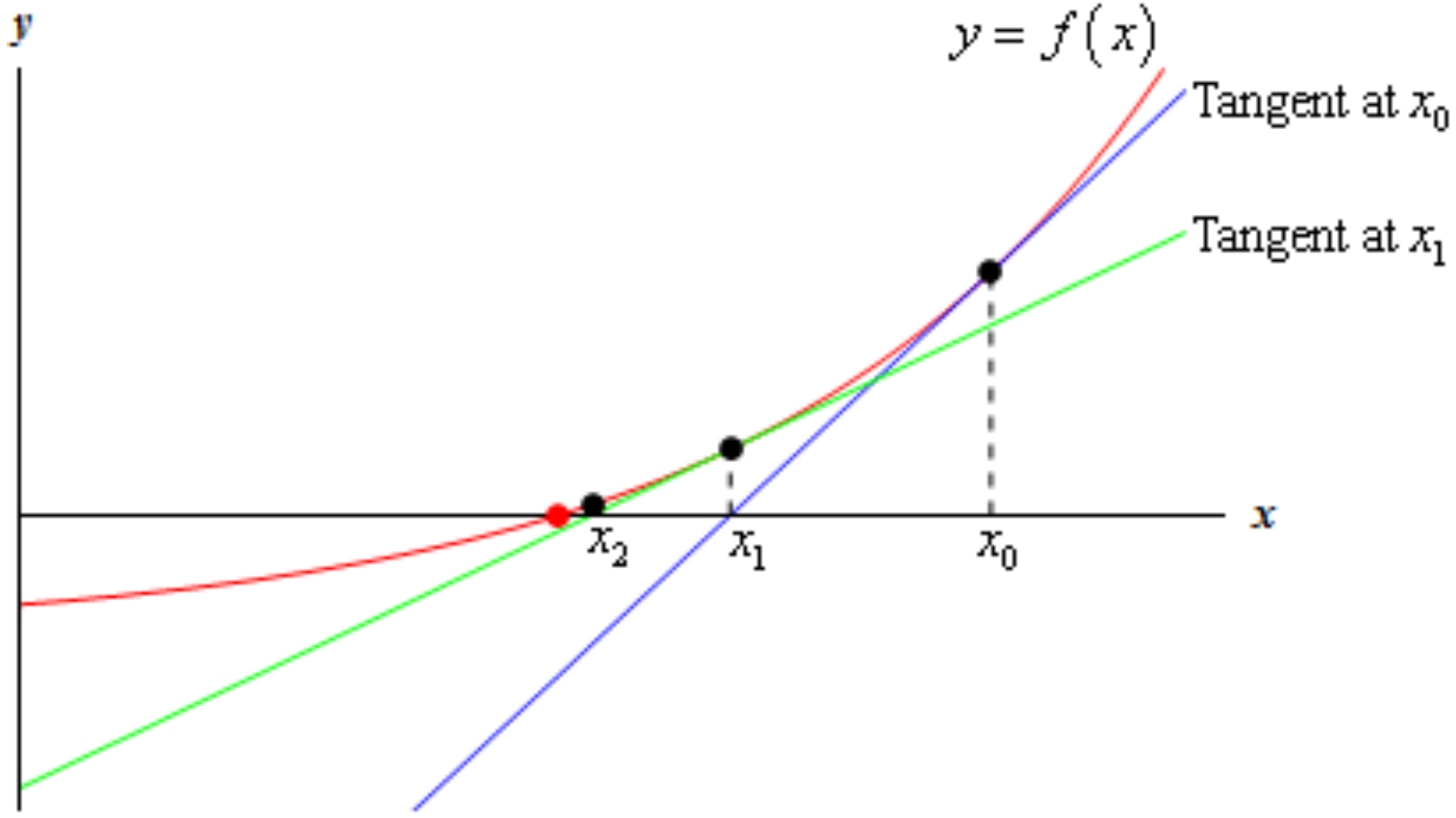
$$x^{(k)} = x^{(k-1)} - \left(\nabla^2 f(x^{(k-1)})\right)^{-1} \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Here $\nabla^2 f(x^{(k-1)})$ is the Hessian matrix of f at $x^{(k-1)}$

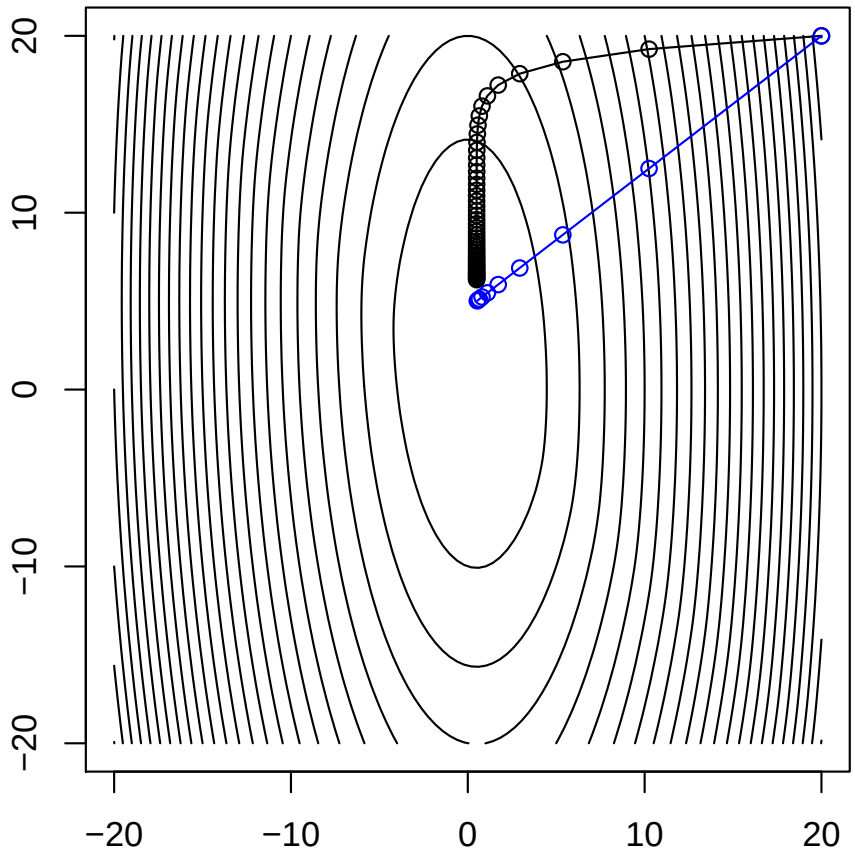
Compare to gradient descent: choose initial $x^{(0)} \in \mathbb{R}^n$, and

$$x^{(k)} = x^{(k-1)} - t_k \cdot \nabla f(x^{(k-1)}), \quad k = 1, 2, 3, \ldots$$

Newton's Method



For $f(x) = (10x_1^2 + x_2^2)/2 + 5\log(1 + e^{-x_1 - x_2})$, compare gradient descent (black) to Newton's method (blue), where both take steps of roughly same length



(For our example we needed to consider a nonquadratic ... why?)

Quasi-Newton Methods

BFGS

From an initial guess $\mathbf{x}_0$ and an approximate Hessian matrix B_0 the following steps are repeated as $\mathbf{x}_k$ converges to the solution.

1. Obtain a direction $\mathbf{p}_k$ by solving: $B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.
2. Perform a line search to find an acceptable stepsize α_k in the direction found in the first step, then update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.
3. Set $\mathbf{s}_k = \alpha_k \mathbf{p}_k$.
4. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$.
5. $B_{k+1} = B_k + \dfrac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} - \dfrac{B_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} B_k}{\mathbf{s}_k^{\mathrm{T}} B_k \mathbf{s}_k}$.

BFGS

From an initial guess $\mathbf{x}_0$ and an approximate Hessian matrix B_0 the following steps are repeated as $\mathbf{x}_k$ converges to the solution.

1. Obtain a direction $\mathbf{p}_k$ by solving: $B_k \mathbf{p}_k = -\nabla f(\mathbf{x}_k)$.
2. Perform a line search to find an acceptable stepsize α_k in the direction found in the first step, then update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$.
3. Set $\mathbf{s}_k = \alpha_k \mathbf{p}_k$.
4. $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$.
5. $B_{k+1} = B_k + \dfrac{\mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} - \dfrac{B_k \mathbf{s}_k \mathbf{s}_k^{\mathrm{T}} B_k}{\mathbf{s}_k^{\mathrm{T}} B_k \mathbf{s}_k}$.

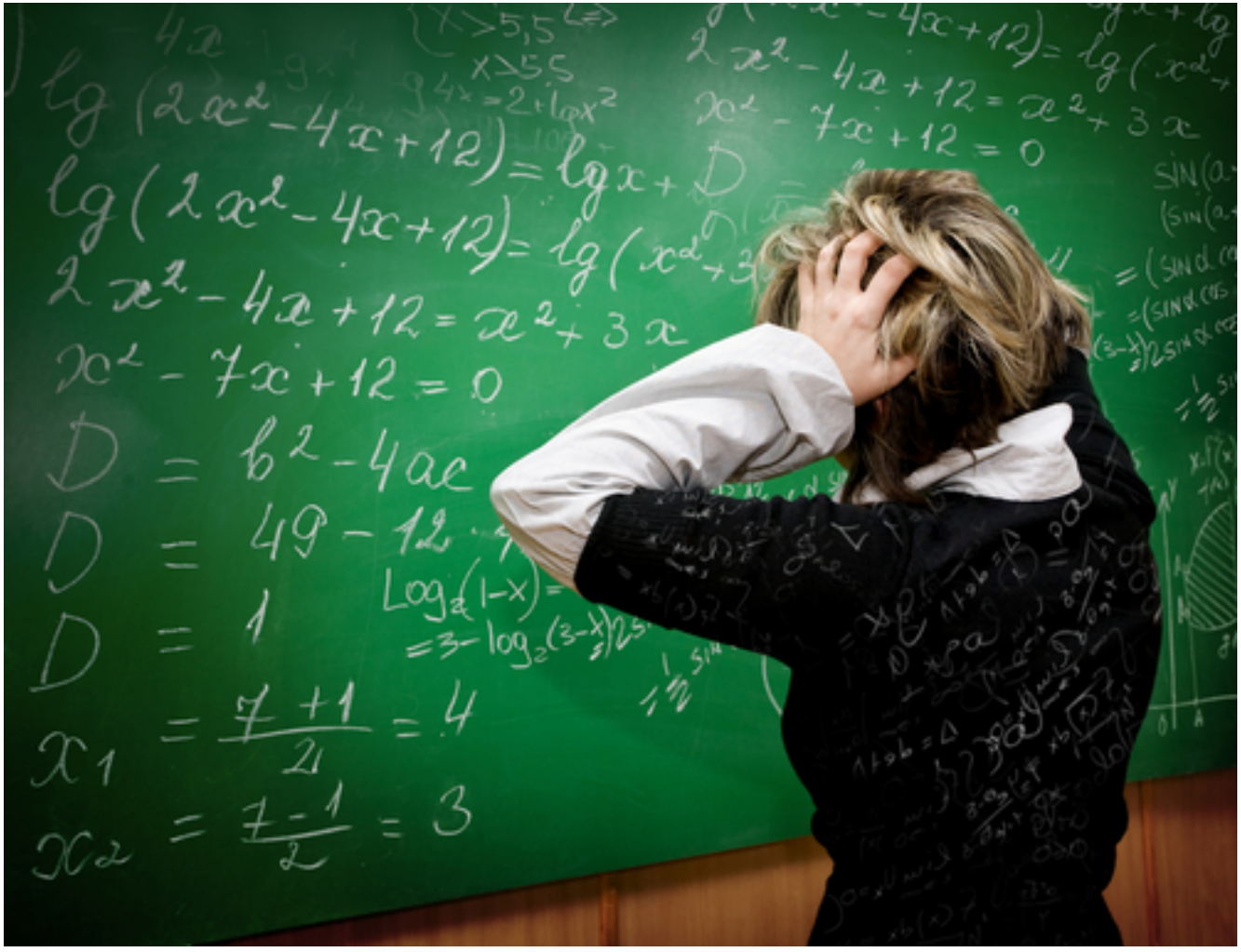
Broyden, Fletcher, Goldfarb, Shanno

Objections

- Too much math
- Too many application
- Too many homeworks

Objections

- Too much math
- Too many application
- Too many homeworks



Just the beginning...

Thanks!

Thanks!