Matrix differential calculus

10-725 Optimization
Geoff Gordon
Ryan Tibshirani

Review

- Matrix differentials: sol'n to matrix calculus pain
  - compact way of writing Taylor expansions, or …
  - definition:
    - df = a(x; dx) [+ r(dx)]
    - a(x; .) linear in 2nd arg
    - r(dx)/||dx|| → 0 as dx → 0
- d(…) is linear: passes thru +, scalar *
- Generalizes Jacobian, Hessian, gradient, velocity

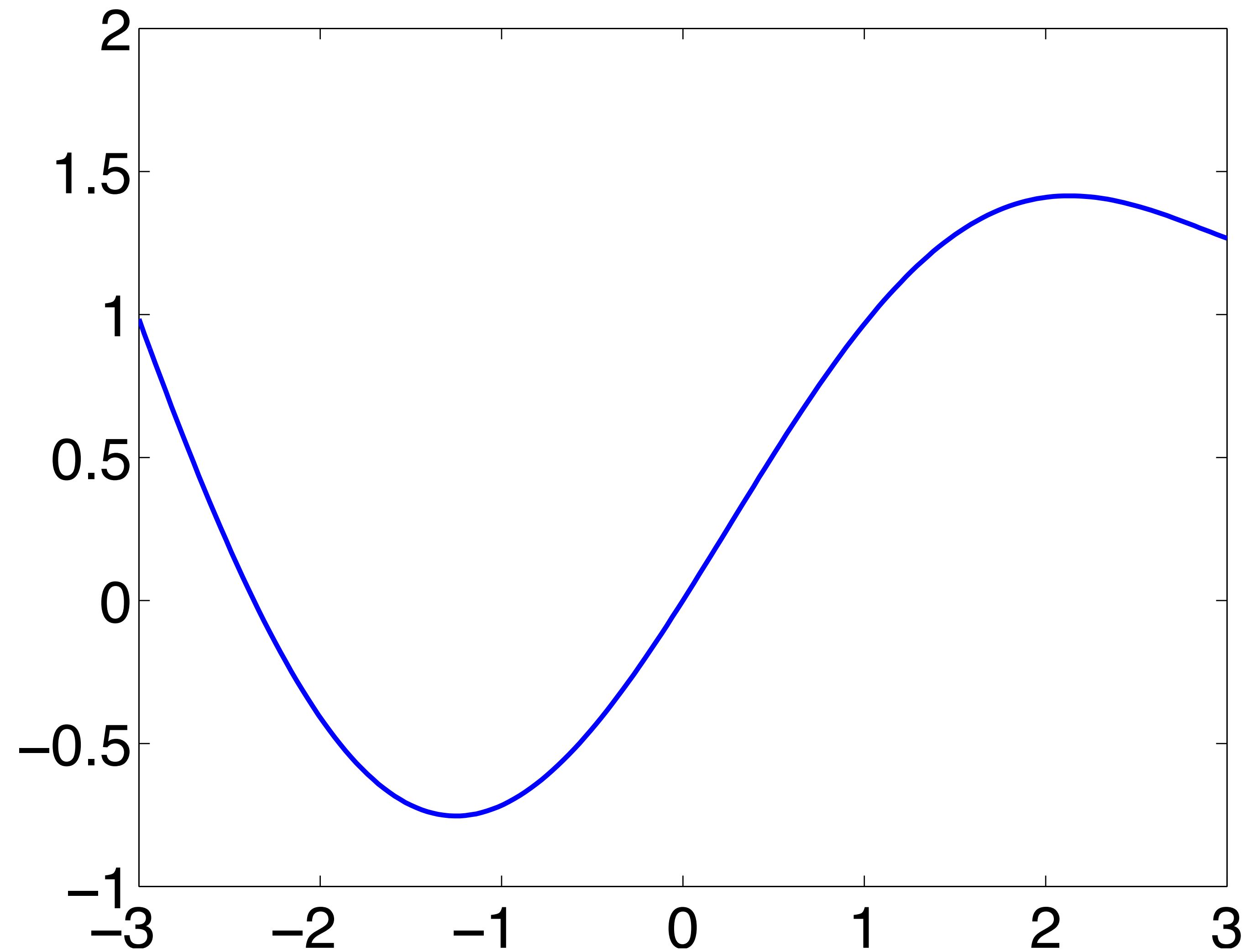
Review

- Chain rule
- Product rule
- Bilinear functions: cross product, Kronecker, Frobenius, Hadamard, Khatri-Rao, …
- Identities
  - rules for working with $\circ$, tr()
  - trace rotation
- Identification theorems

Finding a maximum

or minimum, or saddle point

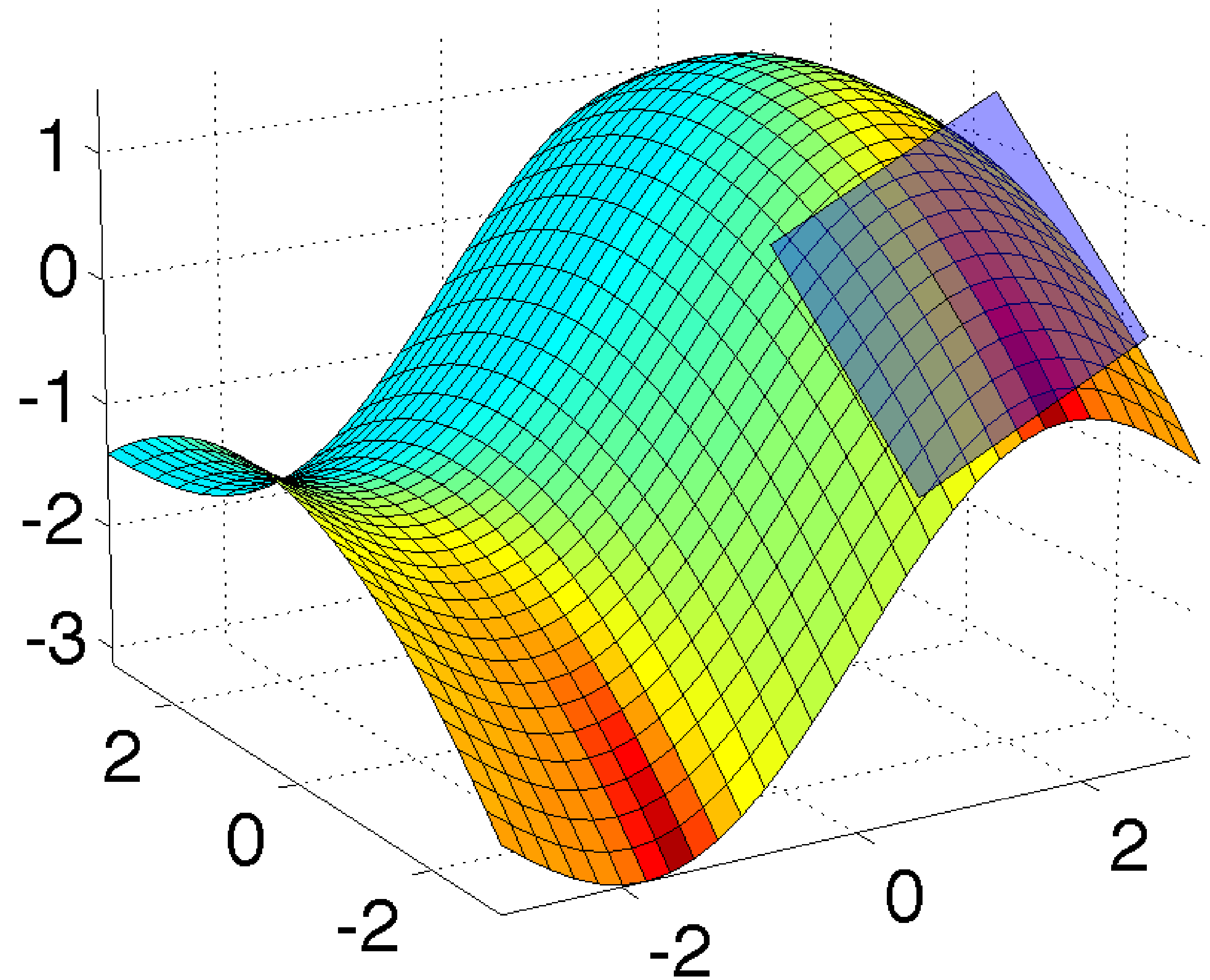
ID for df(x)	scalar x	vector ***x***	matrix X
scalar f	df = a dx	df = $\boldsymbol{a}^T d\boldsymbol{x}$	df = $tr(A^T dX)$



Finding a maximum

or minimum, or saddle point

ID for df(x)	scalar x	vector $\boldsymbol{x}$	matrix X
scalar f	df = a dx	df = $\boldsymbol{a}^T d\boldsymbol{x}$	df = tr($A^T dX$)

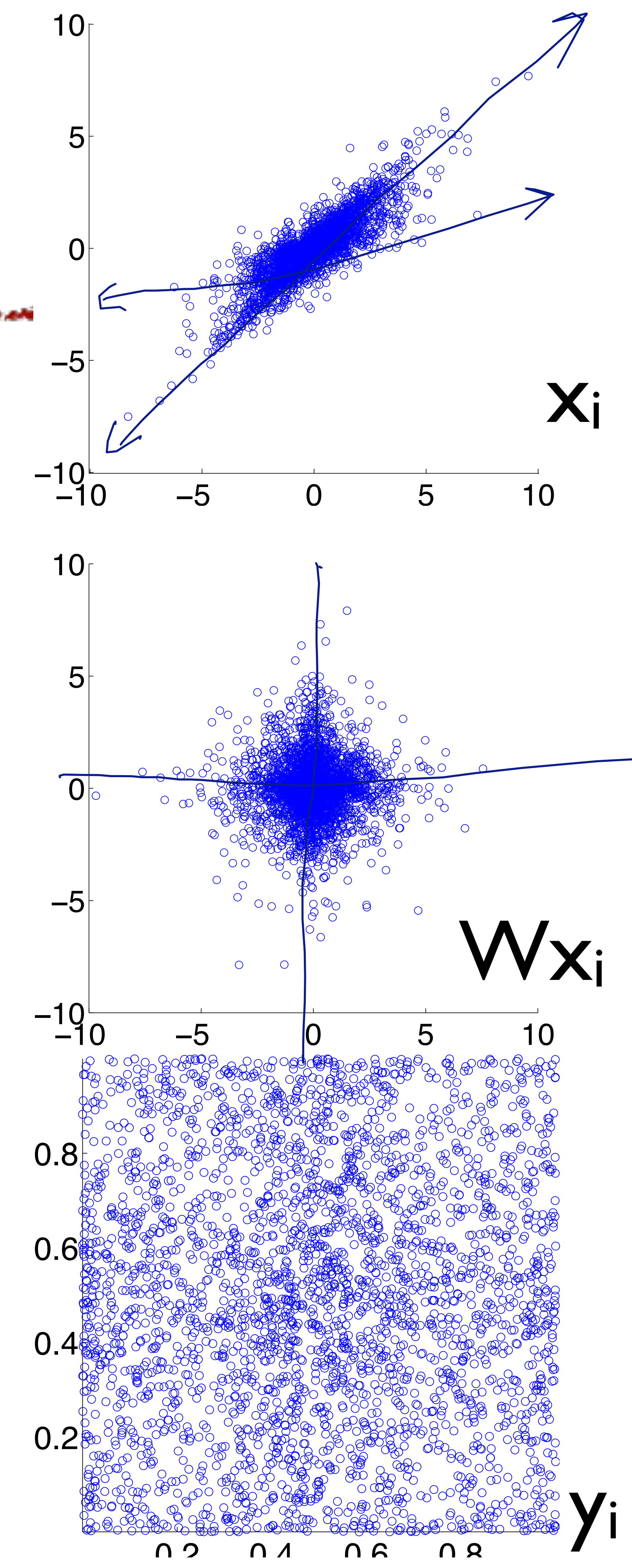


And so forth…

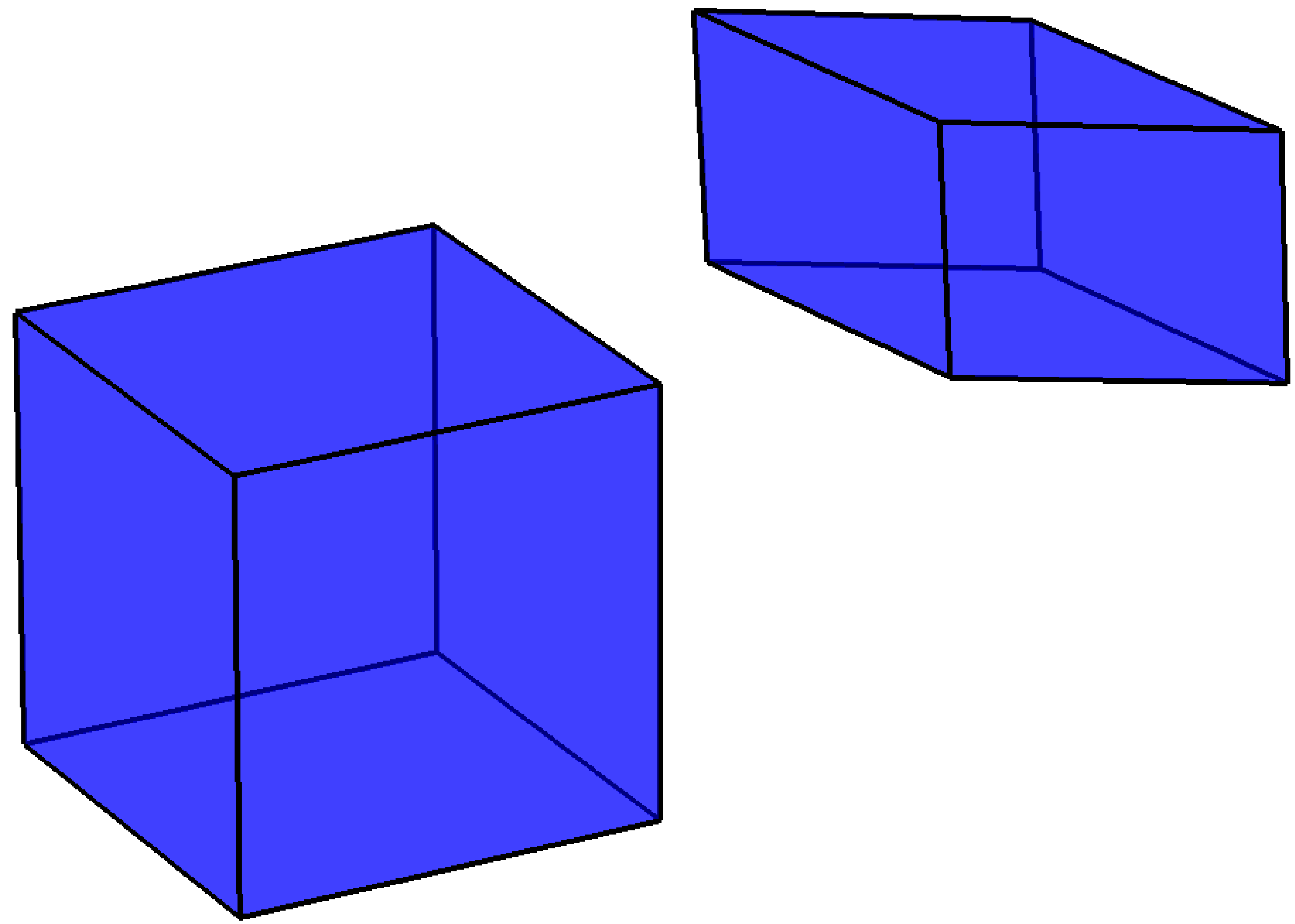
- Can't draw it for X a matrix, tensor, …
- But same principle holds: set coefficient of dX to 0 to find min, max, or saddle point:
  - if df = c(A; dX) [+ r(dX)] then
  - so: max/min/sp iff
  - for c(.; .) any "product",

Ex: Infomax ICA

- Training examples $x_i \in \mathbb{R}^d$, $i = 1{:}n$
- Transformation $y_i = g(Wx_i)$
  - $W \in \mathbb{R}^{d \times d}$ parameter
  - $g(z) =$ scalar fn, componentwise
- Want: independent components



Volume rule



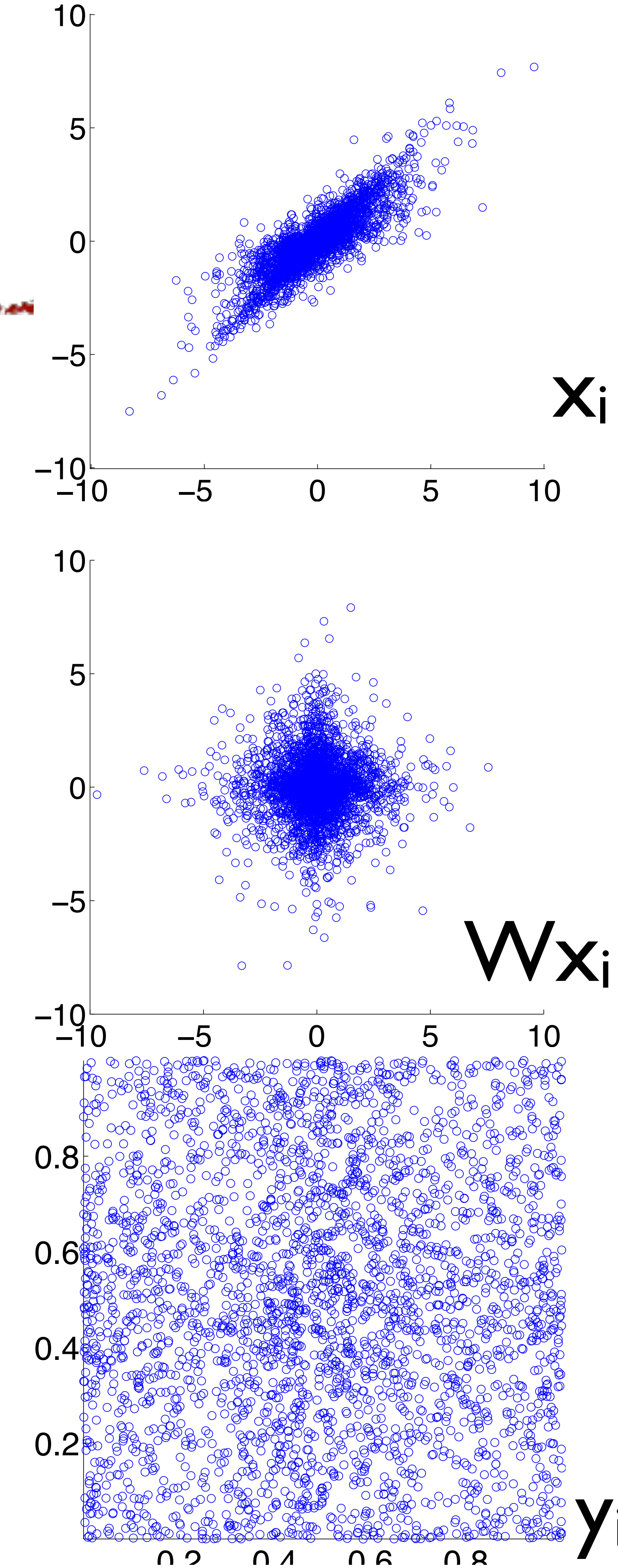
Ex: Infomax ICA

- $y_i = g(Wx_i)$
  - $dy_i =$ $J(x_i, W)\,dx_i = J_i\,dx_i$

est. of $-\int P(y_i) \ln P(y_i)$

- Method: $\max_W \sum_i -\ln(P(y_i))$
  - where $P(y_i) =$ $P(x_i) / |\det J(x_i, W)|$

$\max_W \sum_i \left(\ln|\det J(x_i, W)| - \ln P(x_i) \right)$ (the $\ln P(x_i)$ term crossed out)



Gradient

- $L = \sum_i \ln |\det J_i| \qquad y_i = g(Wx_i) \qquad dy_i = J_i \, dx_i$

Gradient

$J_i = \mathrm{diag}(u_i)\, W \quad dJ_i = \mathrm{diag}(u_i)\, dW + \mathrm{diag}(v_i)\, \mathrm{diag}(dW\, x_i)\, W$

$dL =$

Natural gradient

- $L(W): R^{d \times d} \to R \quad dL = \mathrm{tr}(G^T dW)$
- step $S = \arg\max_S M(S) = \mathrm{tr}(G^T S) - \|SW^{-1}\|_F^2 / 2$
  - scalar case: $M = gs - s^2 / 2w^2$
- M =
- dM =

ICA natural gradient

- $[W^{-T} + C]\, W^T W =$

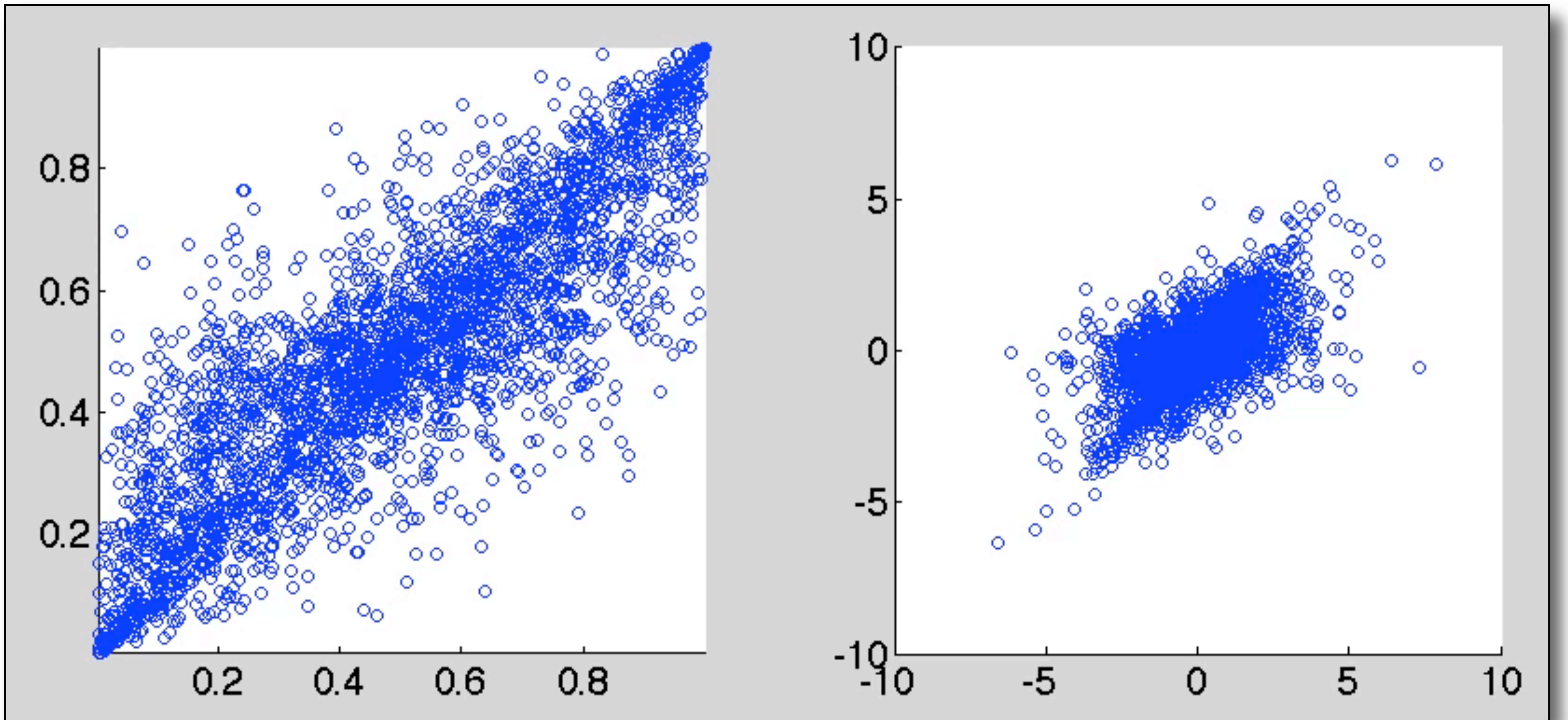
y_i Wx_i

start with $W_0 = I$

ICA natural gradient

- $[W^{-T} + C]\, W^T W =$

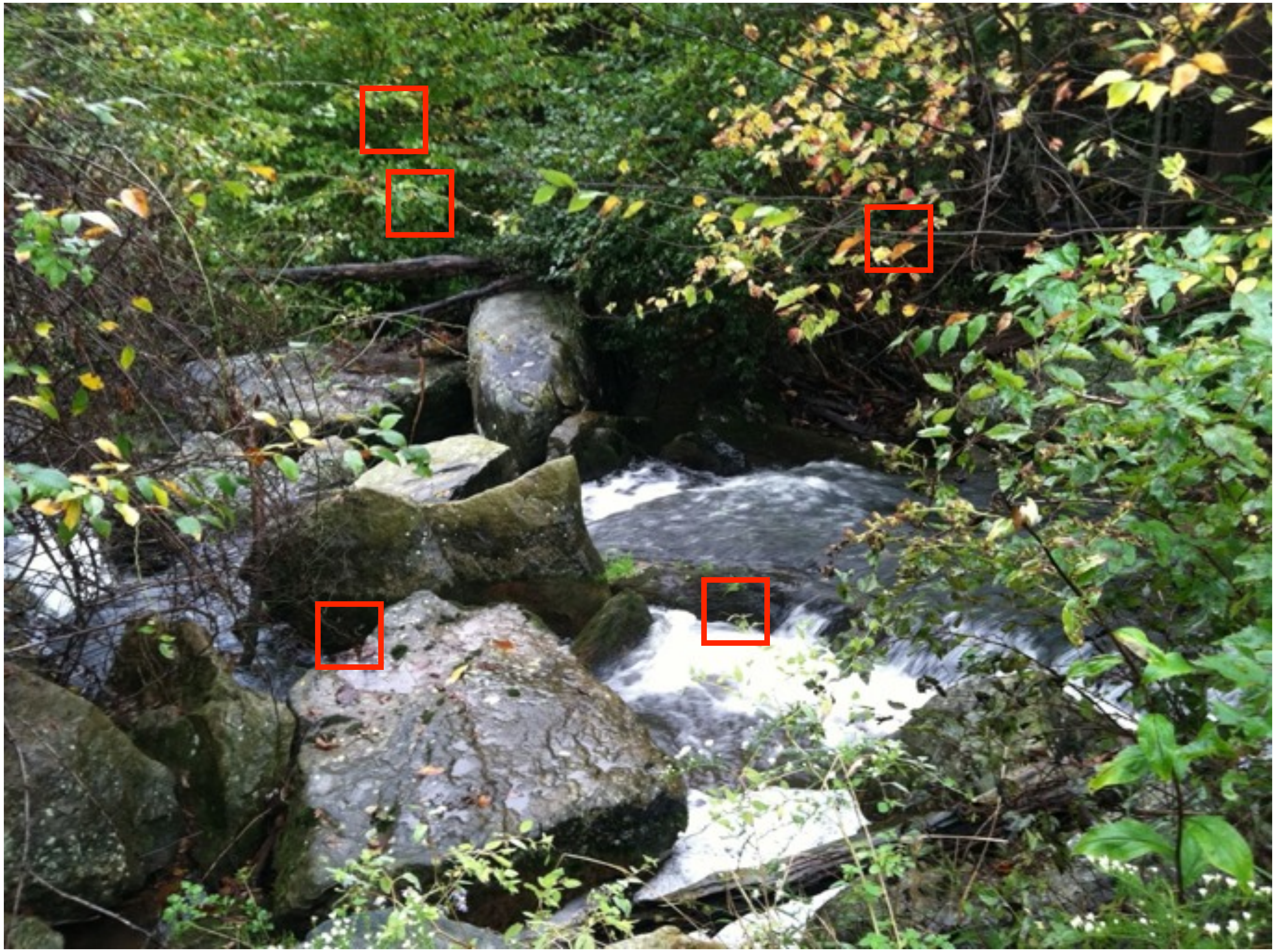
y_i



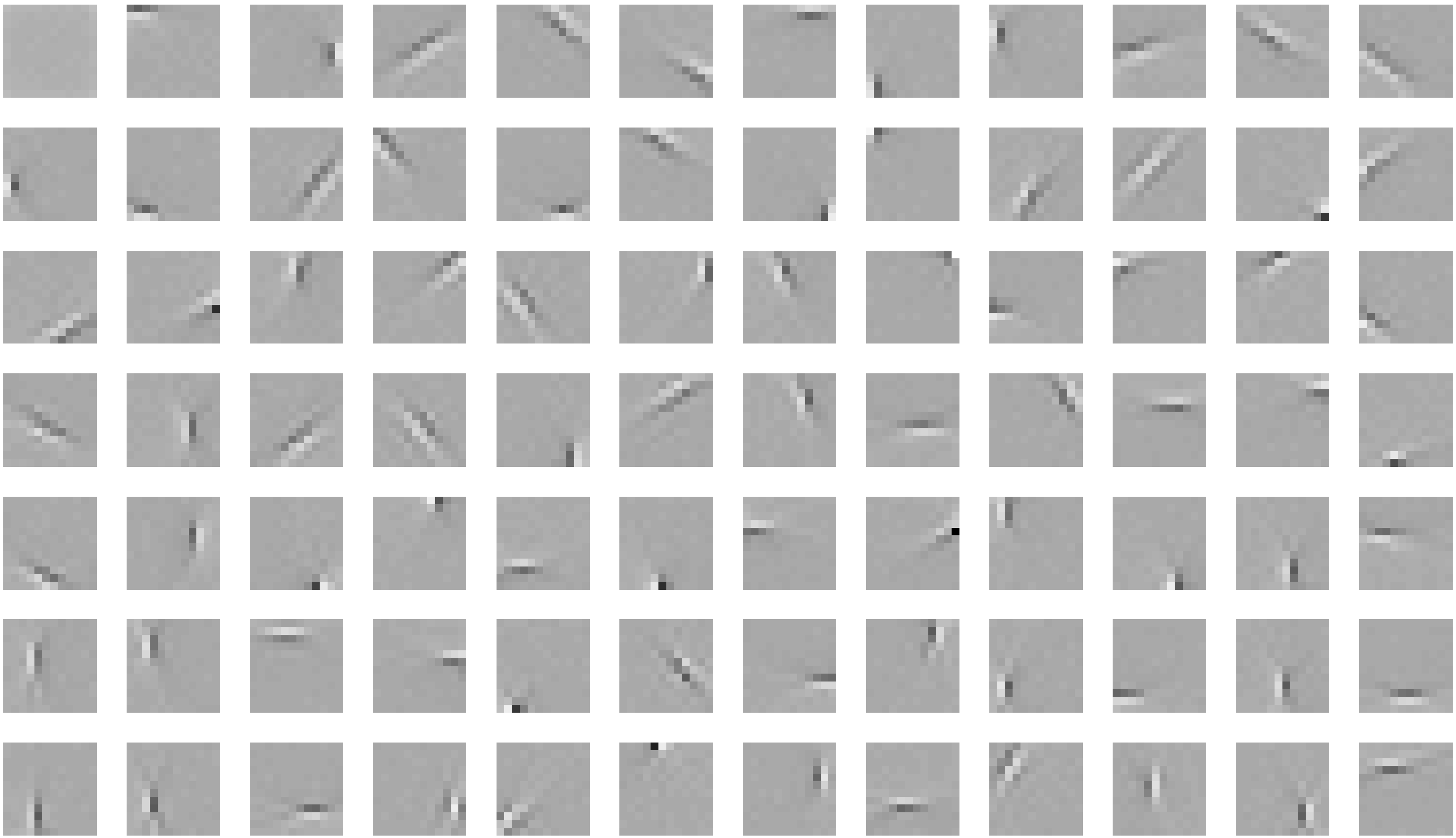
Wx_i

start with $W_0 = I$

ICA on natural image patches



ICA on natural image patches



More info

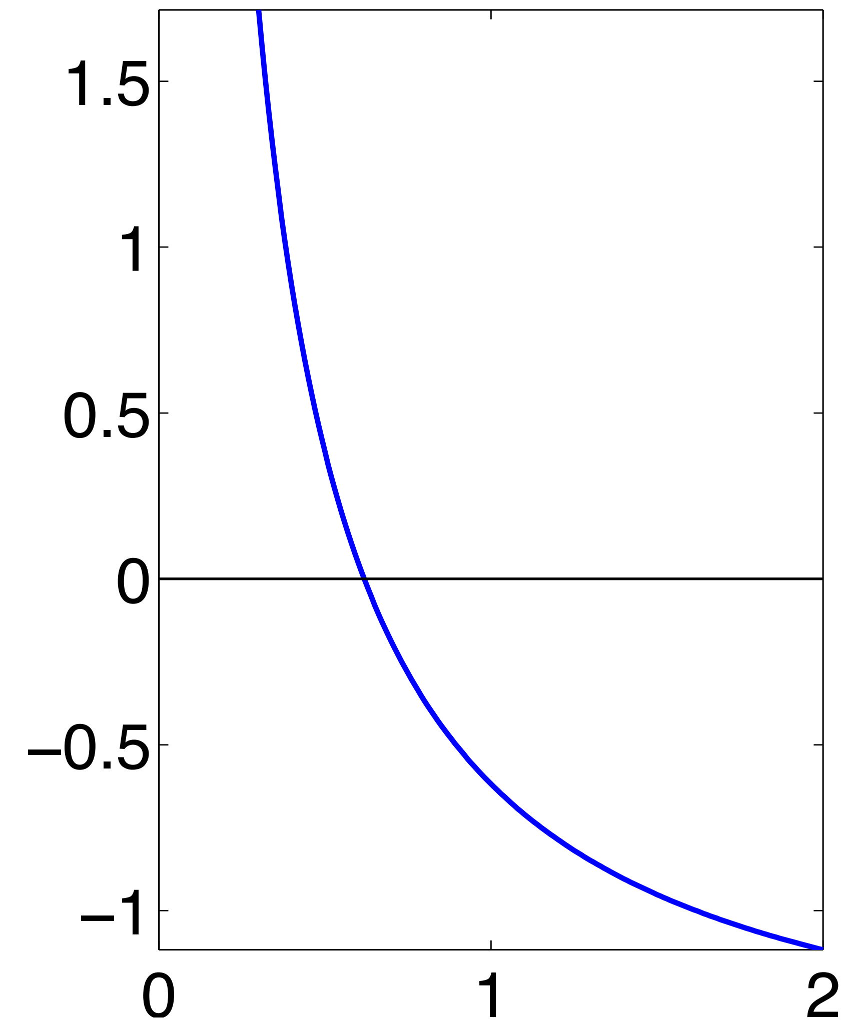
- Minka's cheat sheet:
  - http://research.microsoft.com/en-us/um/people/minka/papers/matrix/
- Magnus & Neudecker. *Matrix Differential Calculus*. Wiley, 1999. 2nd ed.
  - http://www.amazon.com/Differential-Calculus-Applications-Statistics-Econometrics/dp/047198633X
- Bell & Sejnowski. An information-maximization approach to blind separation and blind deconvolution. Neural Computation, v7, 1995.

Newton's method

10-725 Optimization
Geoff Gordon
Ryan Tibshirani

Nonlinear equations

- $x \in R^d$ f: $R^d \to R^d$, diff'ble
  - solve:
- Taylor:
  - J:
- Newton:



Error analysis

dx = x*(1-x*phi)

```
0: 0.7500000000000000
1: 0.5898558813281841
2: 0.6167492604787597
3: 0.6180313181415453
4: 0.6180339887383547
5: 0.6180339887498948
6: 0.6180339887498949
7: 0.6180339887498948
8: 0.6180339887498949
------------------------------
*: 0.6180339887498948
```

Bad initialization

```
                 1.30000000000000000
                -0.13447744098732260
                -0.29821570332700800
                -0.74032738540221900
                -2.36747434311485970
               -13.80392364122258190
              -335.92148595161961570
           -183256.04833606714964840
      -54338444778.11452484130859380
```

Minimization

- $x \in R^d$ f: $R^d \rightarrow R$, twice diff'ble
  - find:
- Newton:

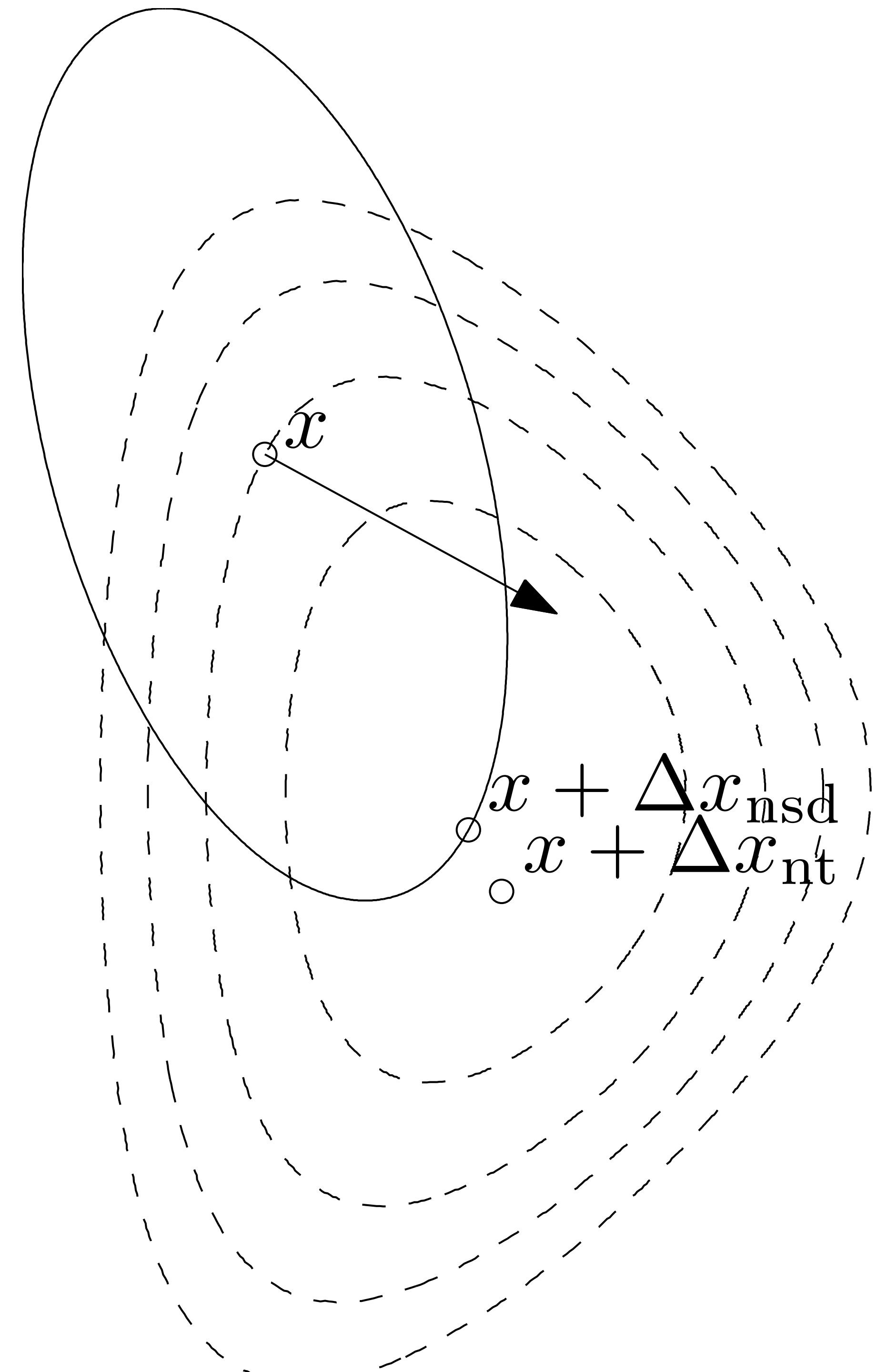
Descent

- Newton step: $d = -(f''(x))^{-1} f'(x)$
- Gradient step: $-g = -f'(x)$
- Taylor: df =
- Let $t > 0$, set dx =
  - df =
- So:

Steepest descent

$g = f'(x)$
$H = f''(x)$

$||d||_H =$



Newton w/ line search

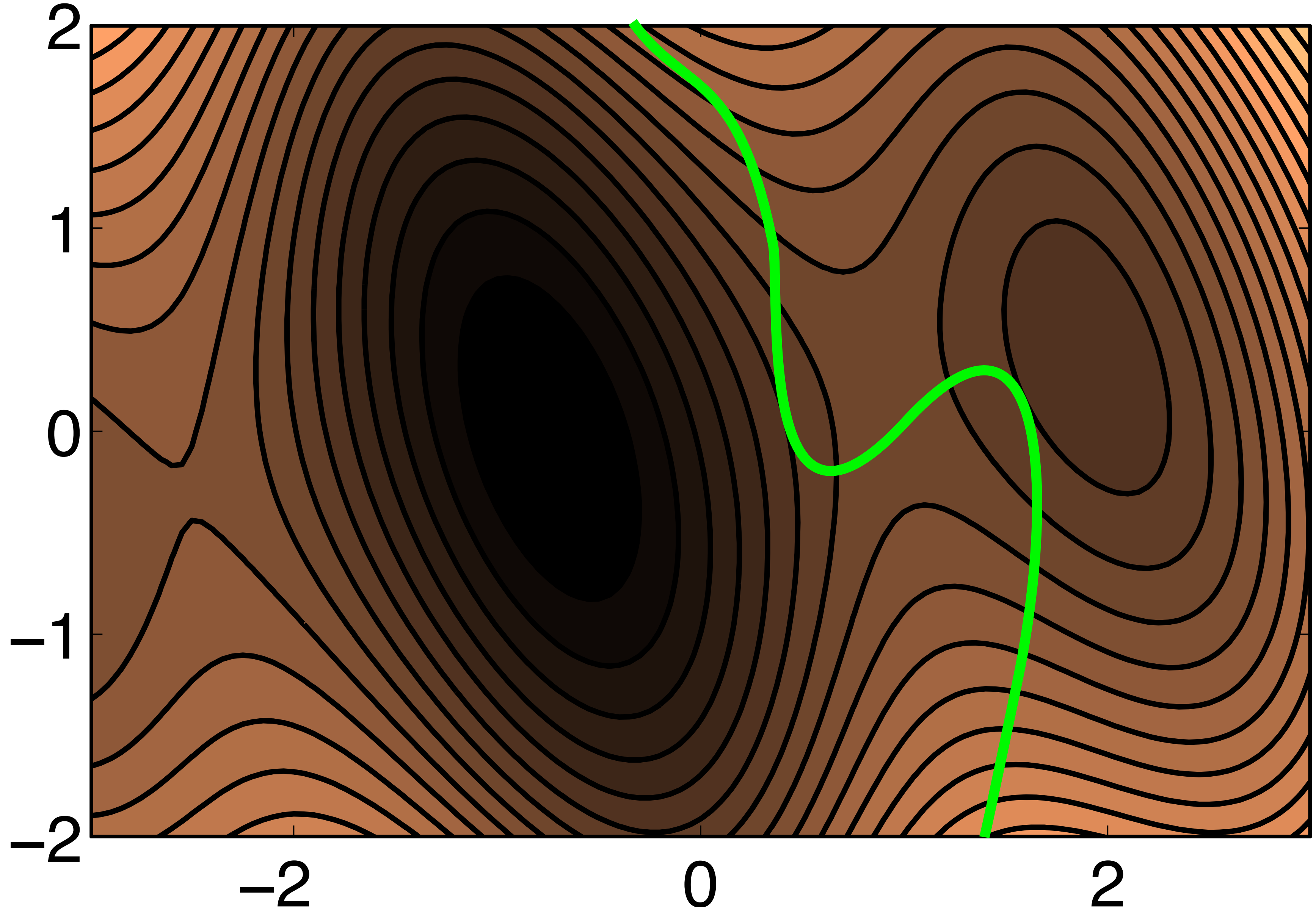
- Pick x_1
- For $k = 1, 2, \ldots$
  - $g_k = f'(x_k)$; $H_k = f''(x_k)$ — *gradient & Hessian*
  - $d_k = -H_k \backslash g_k$ — *Newton direction*
  - $t_k = 1$ — *backtracking line search*
  - while $f(x_k + t_k d_k) > f(x_k) + t\, g_k^T d_k / 2$
    - $t_k = \beta\, t_k$ — *$\beta<1$*
  - $x_{k+1} = x_k + t_k d_k$ — *step*

Properties of damped Newton

- Affine invariant: suppose $g(x) = f(Ax+b)$
  - $x_1, x_2, \ldots$ from Newton on $g()$
  - $y_1, y_2, \ldots$ from Newton on $f()$
  - If $y_1 = Ax_1 + b$, then:
- Convergent:
  - if f bounded below, $f(x_k)$ converges
  - if f strictly convex, bounded level sets, x_k converges
  - typically quadratic rate in neighborhood of x^*

Equality constraints

- min f(x) s.t. h(x) = 0



Optimality w/ equality

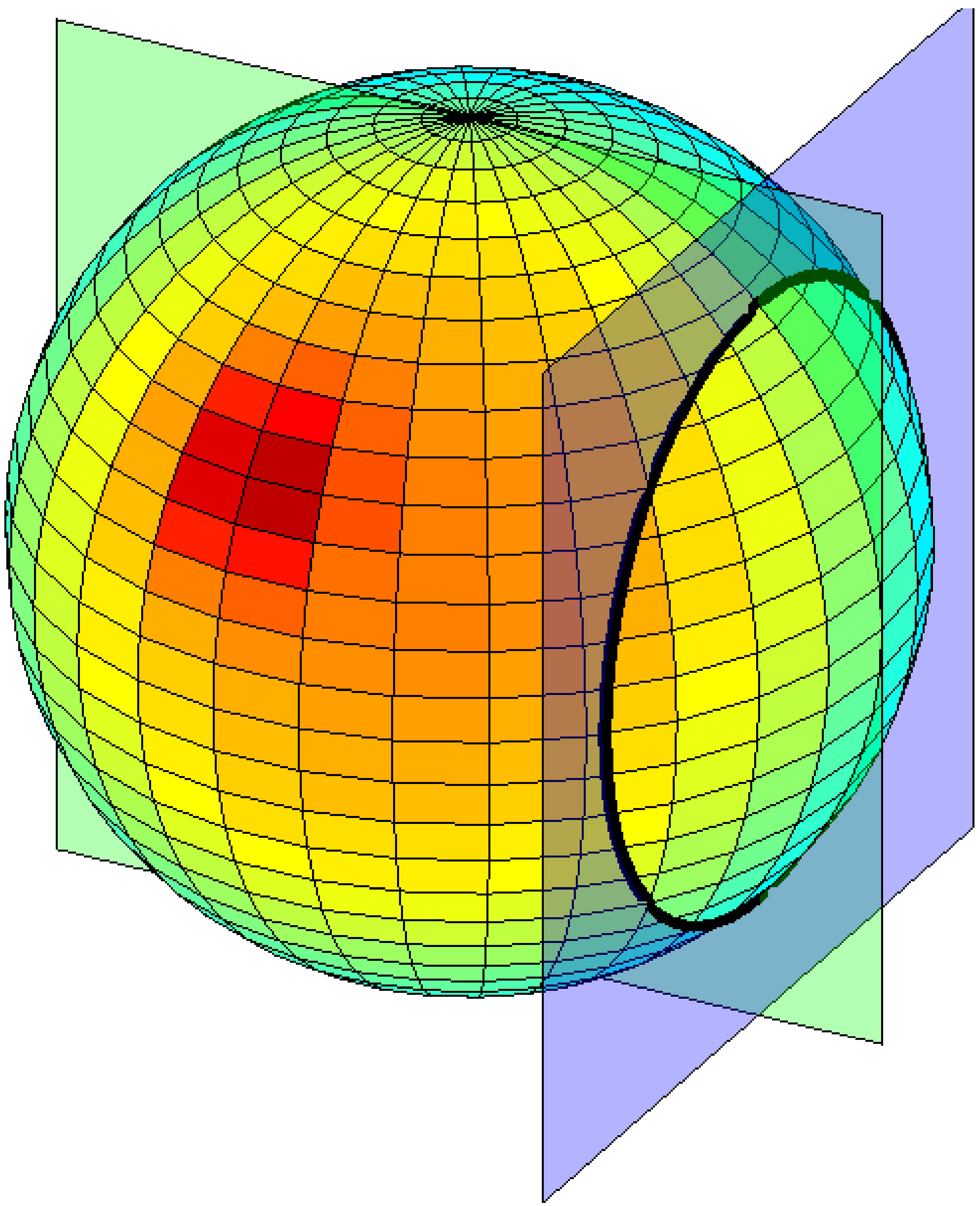
- min f(x) s.t. h(x) = 0
  - f: $R^d \to R$, h: $R^d \to R^k$ (k ≤ d)
  - g: $R^d \to R^d$ (gradient of f)
- Useful special case: min f(x) s.t. Ax = 0

Picture

$$\max c^\top \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ s.t.}$$

$$x^2 + y^2 + z^2 = 1$$

$$a^\top x = b$$



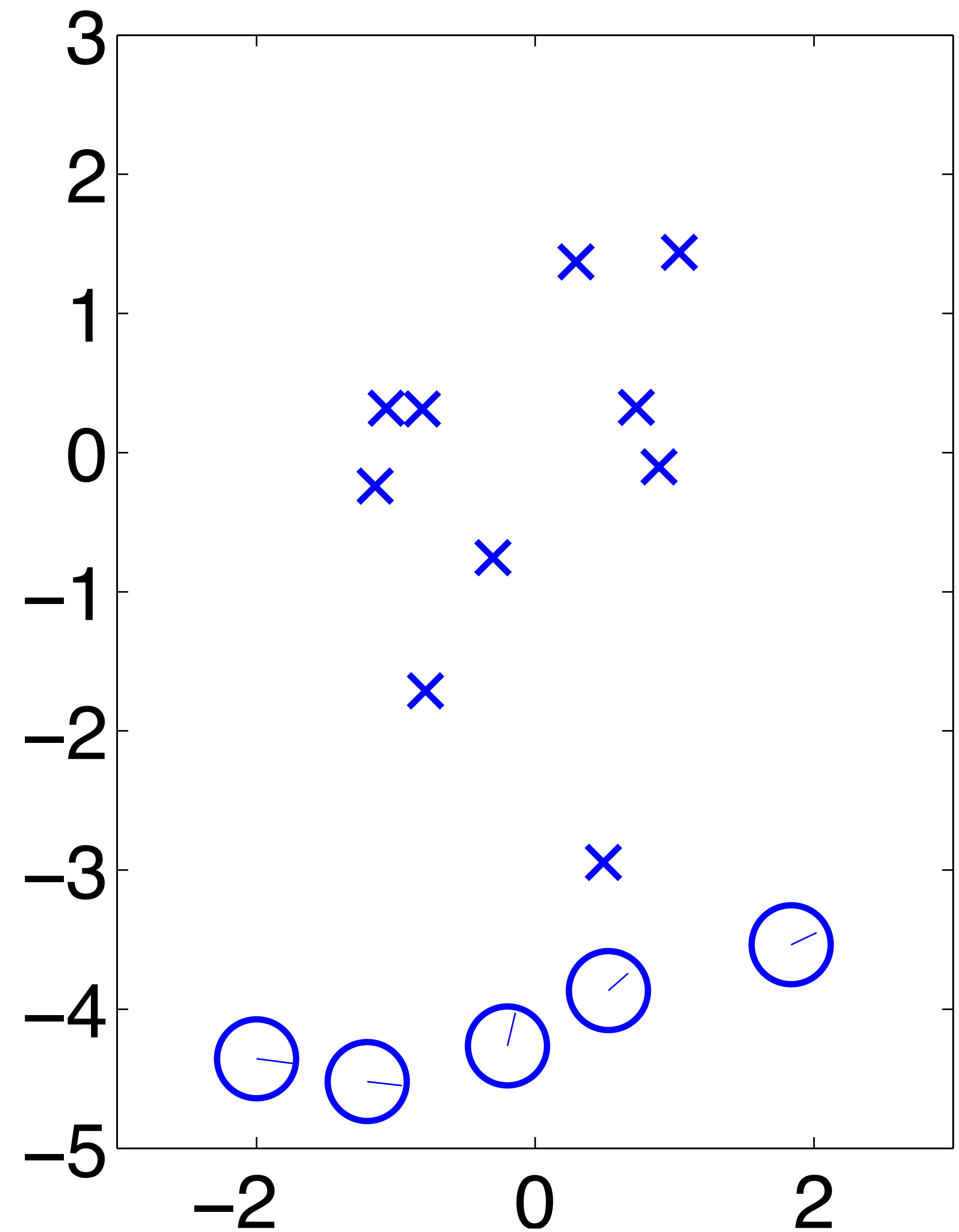
Optimality w/ equality

- min f(x) s.t. h(x) = 0
  - f: $R^d \to R$, h: $R^d \to R^k$ (k ≤ d)
  - g: $R^d \to R^d$ (gradient of f)
- Now suppose:
  - dg = dh =
- Optimality:

Example: bundle adjustment

- Latent:
  - Robot positions x_t, θ_t
  - Landmark positions y_k
- Observed: odometry, landmark vectors
  - $v_t = R_{\theta t}[x_{t+1} - x_t] + noise$
  - $w_t = [\theta_{t+1} - \theta_t + noise]_\pi$
  - $d_{kt} = R_{\theta t}[y_k - x_t] + noise$

O = {*observed kt pairs*}



Example: bundle adjustment

- Latent:
  - Robot positions x_t, θ_t
  - Landmark positions y_k
- Observed: odometry, landmark vectors
  - $v_t = R_{\theta t}[x_{t+1} - x_t] + \textit{noise}$
  - $w_t = [\theta_{t+1} - \theta_t + \textit{noise}]_\pi$
  - $d_{kt} = R_{\theta t}[y_k - x_t] + \textit{noise}$

Bundle adjustment

$$\min_{x_t, u_t, y_k} \sum_t \|v_t - R(u_t)[x_{t+1} - x_t]\|^2 + \sum_t \|R_{w_t} u_t - u_{t+1}\|^2 +$$

$$\sum_{(t,k)\in O} \|d_{k,t} - R(u_t)[y_k - x_t]\|^2$$

$$\text{s.t. } u_t^\top u_t = 1$$

Ex: MLE in exponential family

$$L = -\ln \prod_k P(x_k \mid \theta)$$

$$P(x_k \mid \theta) =$$

$$g(\theta) =$$

MLE Newton interpretation

Comparison

of methods for minimizing a convex function

	Newton	FISTA	(sub)grad	stoch. (sub)grad.
convergence				
cost/iter				
smoothness				

Variations

- Trust region
  - $[H(x) + tI]dx = -g(x)$
  - $[H(x) + tD]dx = -g(x)$
- Quasi-Newton
  - use only gradients, but build estimate of Hessian
  - in R^d, d gradient estimates at "nearby" points determine approx. Hessian (think finite differences)
  - can often get "good enough" estimate w/ fewer—can even forget old info to save memory (L-BFGS)

Variations: Gauss-Newton

$$L = \min_\theta \sum_k \frac{1}{2} \|y_k - f(x_k, \theta)\|^2$$

Variations: Fisher scoring

- Recall Newton in exponential family

$$E[xx^\top \mid \theta]d\theta = \bar{x} - E[x \mid \theta]$$

- Can use this formula in place of Newton, even if not an exponential family
    - descent direction, even w/ no regularization
    - "Hessian" is independent of data
    - often a wider radius of convergence than Newton
    - can be superlinearly convergent