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

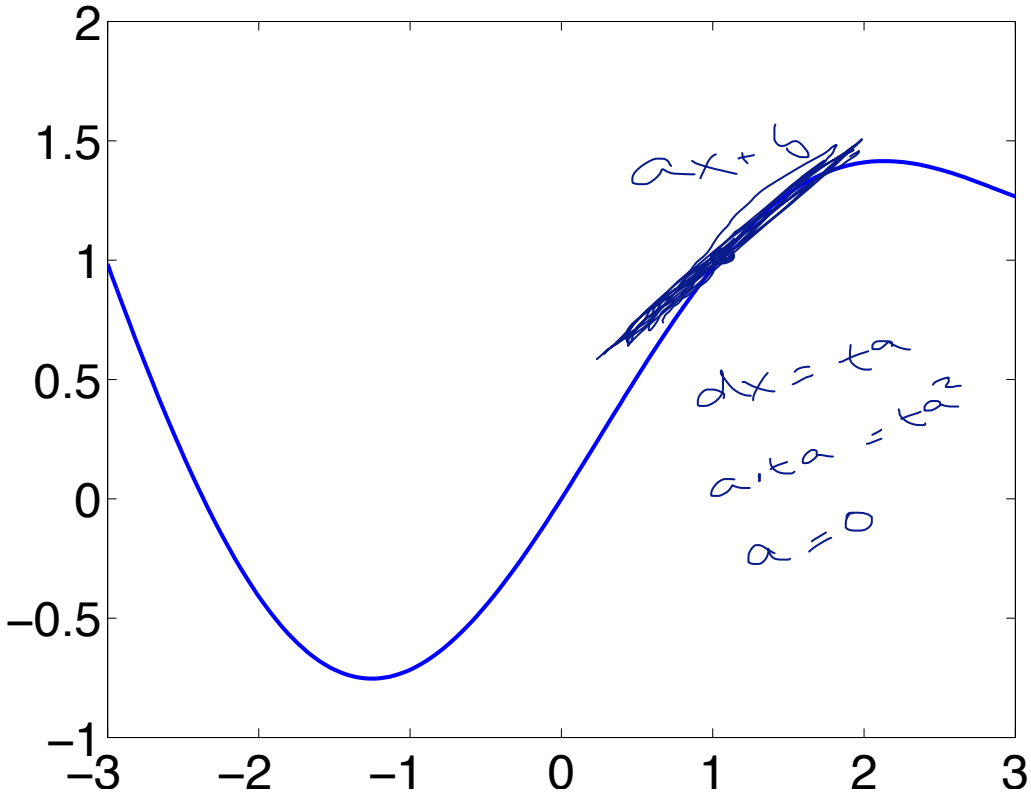
Review

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

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($A^T dX$)



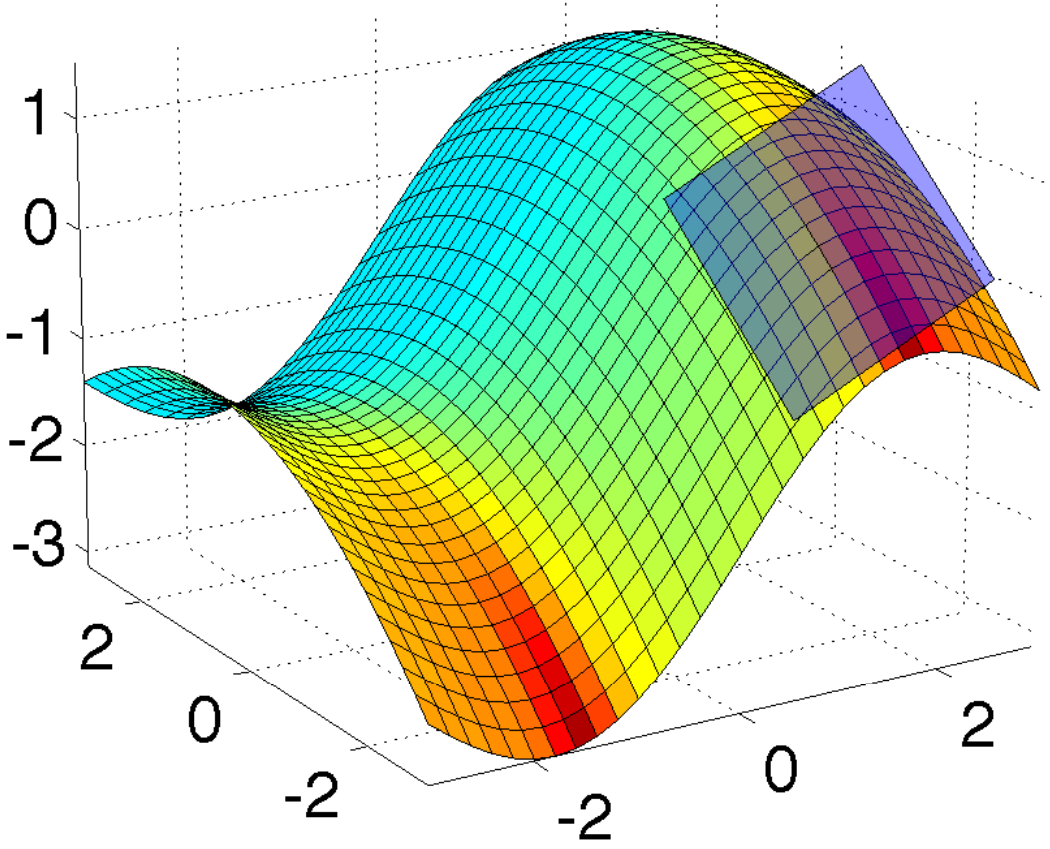
Finding a maximum

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($A^T dX$)

$dx = t\vec{a}$

$df \approx t\vec{a}^T\vec{a} > 0$ if $a \neq 0$



And so forth…

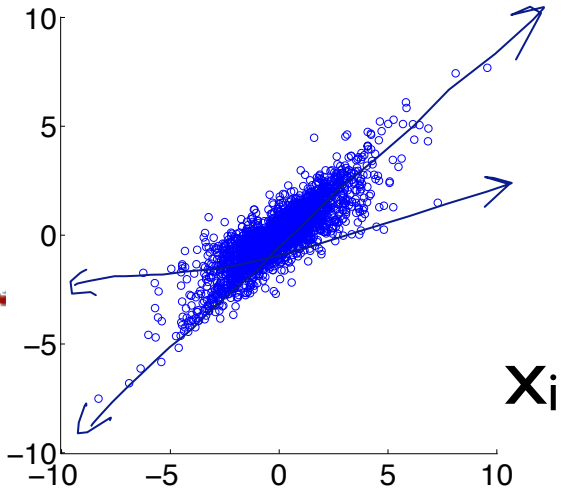
- Can't draw it for X a matrix, tensor, …
- But same principle holds: set coefficient of dX to 0 to find min, max, or saddle point:
  - if df = c(A; dX) [+ r(dX)] then

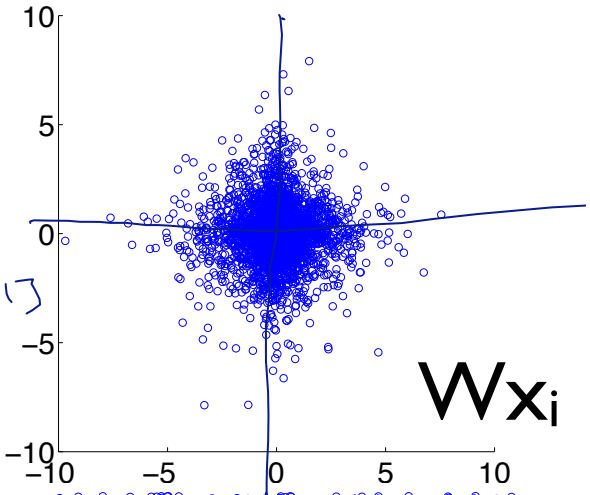
$dX = tA \quad df = c(A; tA) = t\,c(A; A)$

  - so: max/min/sp iff $c(A; A) = 0$
  - for c(.; .) any "product", $\Longleftarrow \quad A = 0$

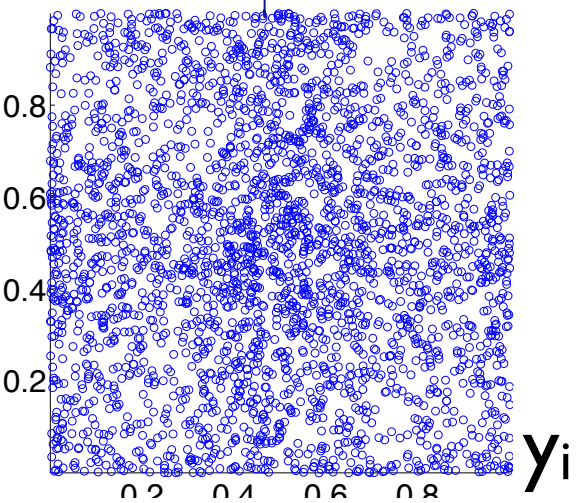
Ex: Infomax ICA

- 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 $g: \mathbb{R} \to [0,1]$
- Want: independent components

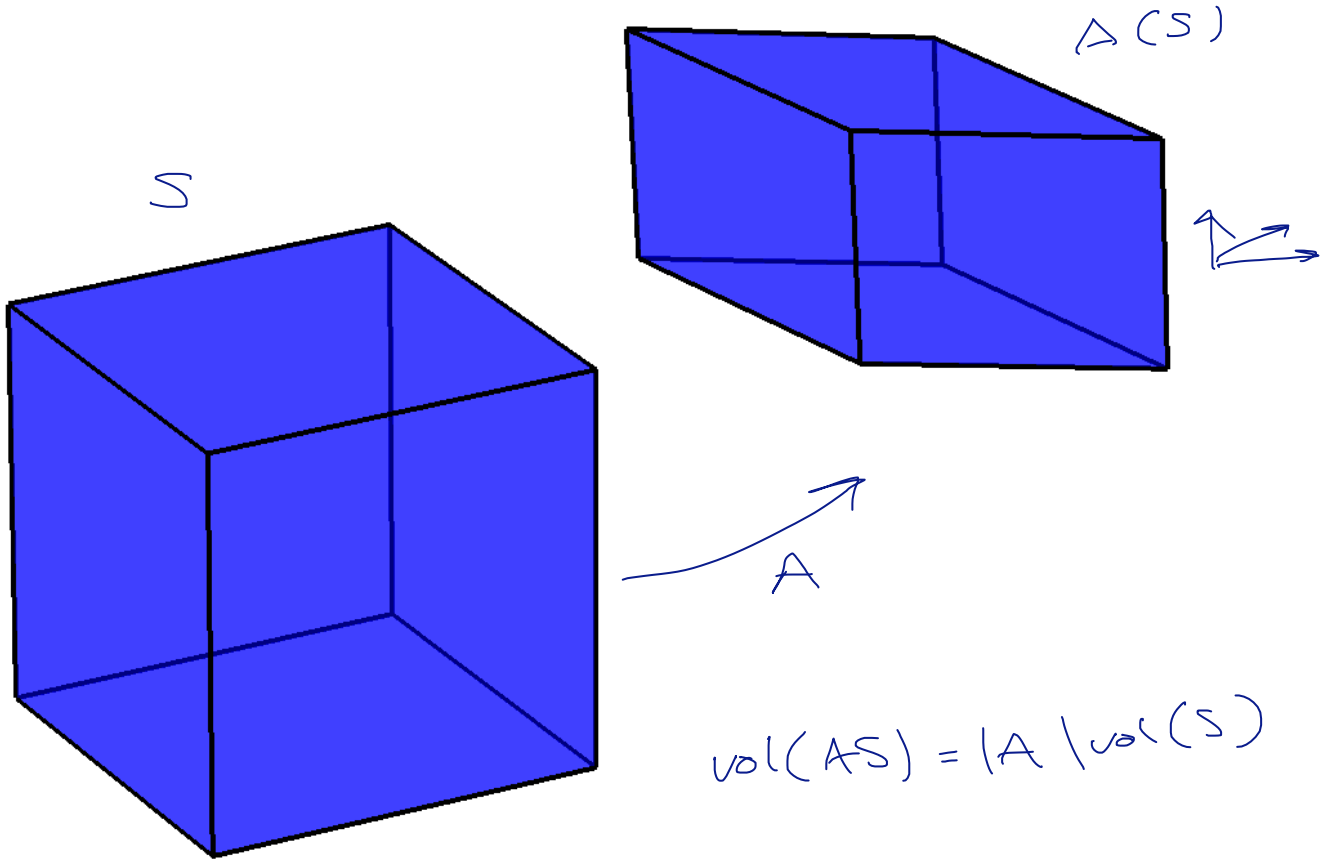




unif →



Volume rule



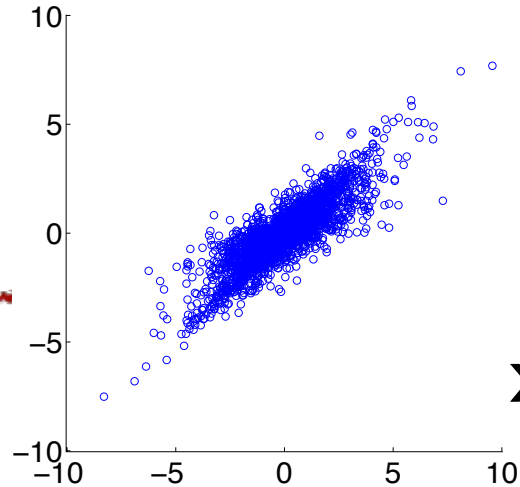
Ex: Infomax ICA

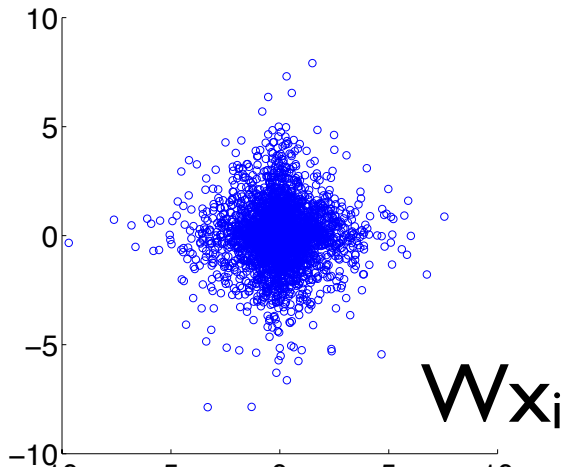
- $y_i = g(Wx_i)$
  - $dy_i = J(x_i, W)\,dx_i = J_i\,dx_i$

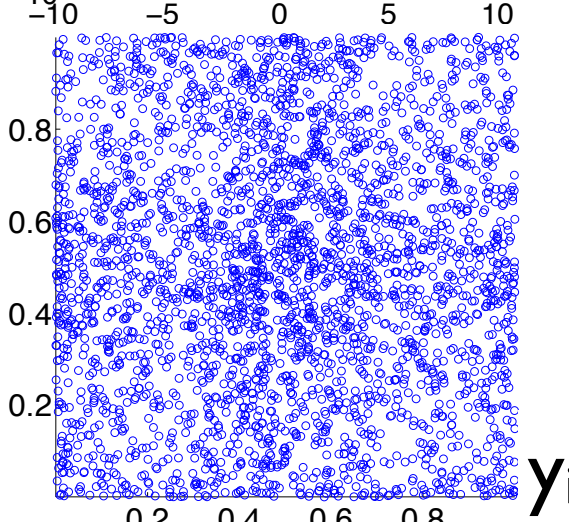
$\max H(P(y)) = -\int P(y) \ln P(y)\,dy = -E(\ln P(y))$

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

$\max_W \sum_i (\ln |\det J(x_i, W)| - \ln P(x_i))$







Gradient

$J_i = \mathrm{diag}(u_i) W$

- $L = \sum_i \ln |\det J_i|$ $\quad y_i = g(W x_i)$ $\quad dy_i = J_i\, dx_i$

$dJ_i = (d\,\mathrm{diag}(u_i)) W + \mathrm{diag}(u_i)\, dW$

$= \mathrm{diag}(v \circ dW x_i) W + \mathrm{diag}(u_i)\, dW$

$dy_i = g'(Wx_i) \circ d(Wx_i)$

$= u_i \circ (W dx_i)$

$= \mathrm{diag}(u_i) W dx_i$

$u_i = g'(Wx_i)$

$du_i = \underbrace{g''(Wx_i)}_{v_i} \circ (dW x_i)$

$d\,\mathrm{diag}(u_i) = \mathrm{diag}(v_i \circ dW x_i)$

Gradient

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

$$dL = \sum_i d \ln |\det J_i|$$

$$= \sum_i \mathrm{tr}(J_i^{-1} dJ_i)$$

$$= \sum_i \mathrm{tr}(W^{-1} dW + W^{-1} \mathrm{diag}(u_i)^{-1} \mathrm{diag}(\alpha_i)\, \mathrm{diag}(dW x_i)\, W)$$

$$= \sum_i \mathrm{tr}(W^{-1} dW) + \mathrm{tr}(\mathrm{diag}(\alpha_i)\, \mathrm{diag}(dW x_i))$$

$$= n \,\mathrm{tr}(W^{-1} dW) + \mathrm{tr}\left(\left(\sum_i x_i \alpha_i^T\right) dW\right)$$

$$\frac{dL}{dW} = n W^{-T} + \sum_i \alpha_i x_i^T = n(W^{-T} + C)$$

Natural gradient

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

$G = S W^{-1} W^{-T}$

$G W^T W = S$

ICA natural gradient

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

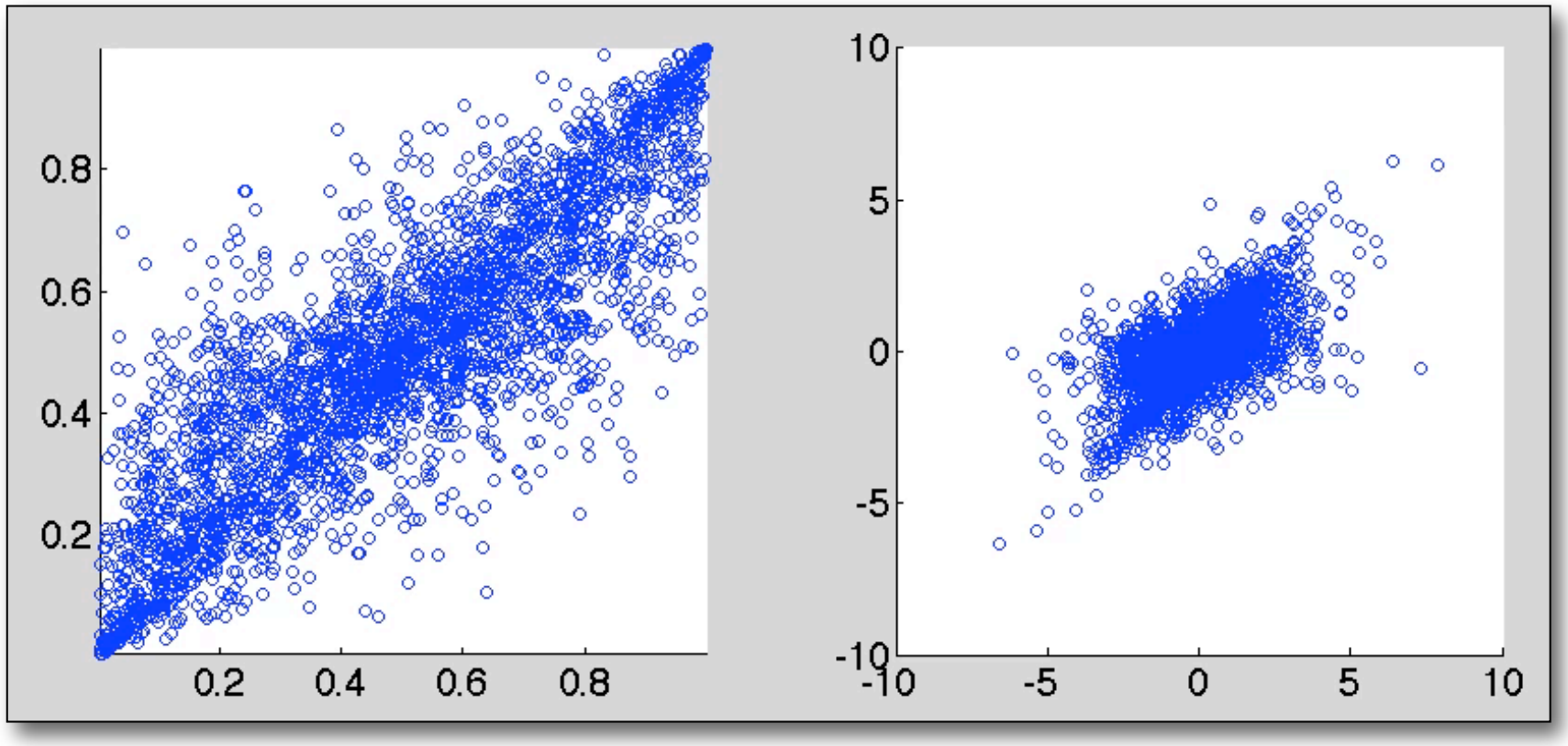
y_i

Wx_i

start with $W_0 = I$

ICA natural gradient

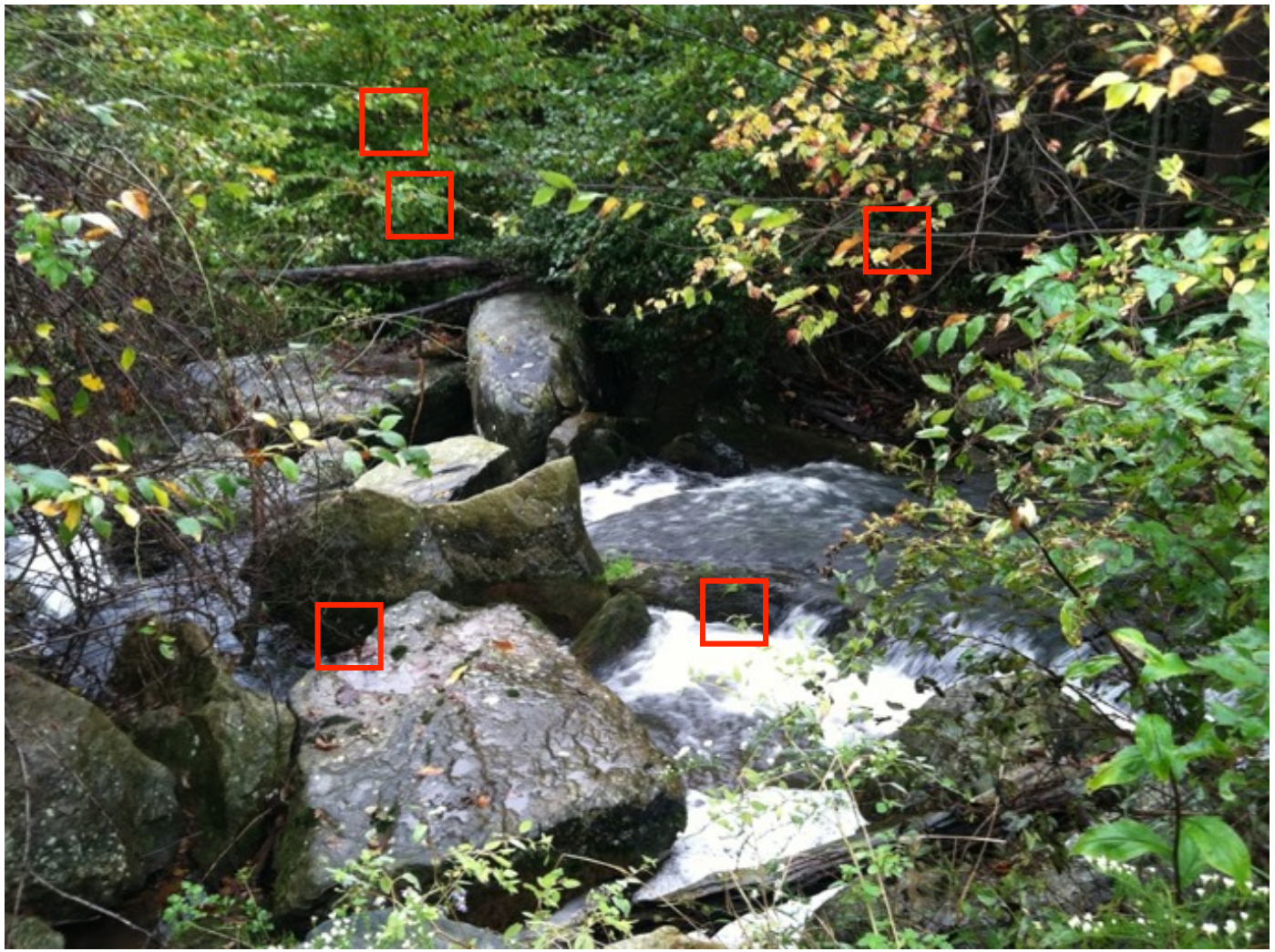
- $[W^{-T} + C] W^T W =$



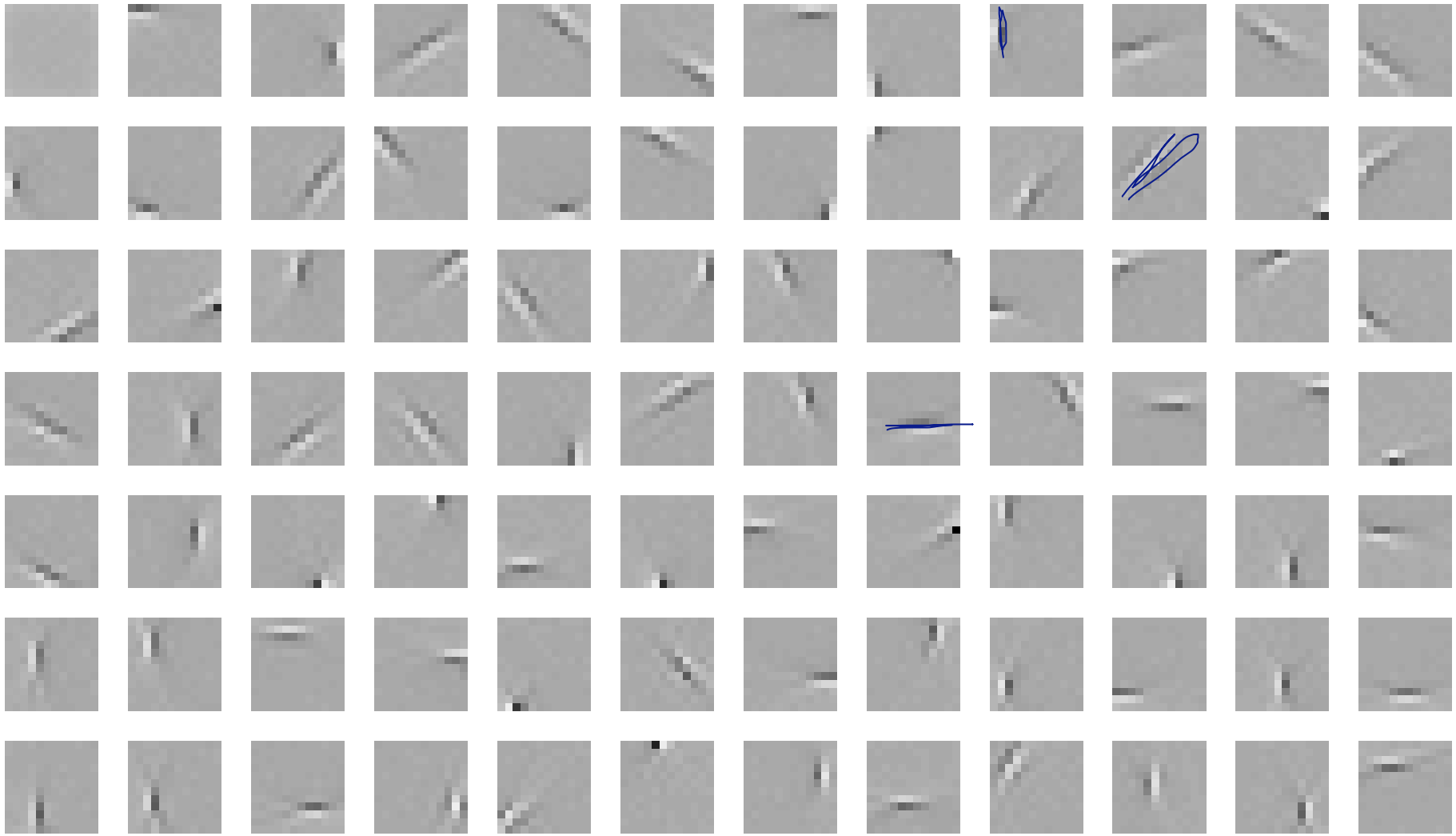
y_i Wx_i

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

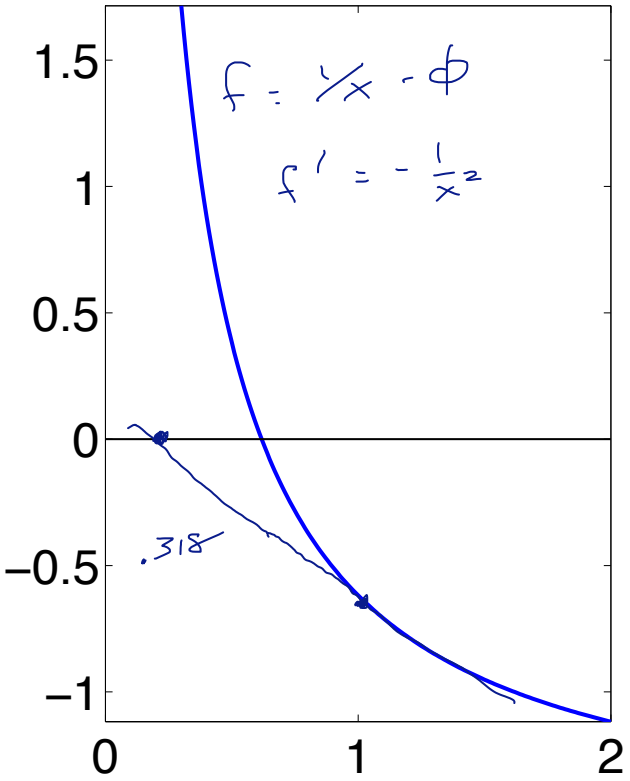
Geoff Gordon
Ryan Tibshirani

Nonlinear equations

- $x \in R^d$ $\quad f: R^d \to R^d$, diff'ble
  - solve: $f(x) = 0$
- Taylor: $f(y) \approx f(x) + J(x)(y-x) = \hat{f}(y)$
  - J: Jacobian
- Newton: $\hat{f}(y) = 0$

$$f(x) + J(x)(y-x) = 0$$

$$y - x = -J(x)^{-1} f(x)$$



$$1 - \phi + (-1)(y-x)$$

$$\hat{f} = f(x) - \frac{1}{x^2} dx = 0$$

$$\frac{f(x)}{1/x - \phi} - \frac{1}{x^2} dx = 0$$

$$1 - x\phi - \frac{1}{x} dx = 0$$

$$x - x^2\phi < dx$$

Error analysis

$\epsilon = x\phi - 1 \qquad x^+ = x + x(1 - x\phi)$

$\epsilon^+ = x^+\phi - 1$

$= [x + x\underbrace{(1 - x\phi)}_{-\epsilon}]\phi - 1$

$= x\phi - x\epsilon\phi - 1$

$= +\epsilon - x\epsilon\phi$

$= \epsilon(1 - x\phi) = -\epsilon^2$

dx = x*(1-x*phi)

```
0: 0.75000000000000000
1: 0.58985588132818410
2: 0.61674926047875970
3: 0.61803131814154530
4: 0.61803398873835470
5: 0.61803398874989480
6: 0.61803398874989490
7: 0.61803398874989480
8: 0.61803398874989490
-----------------------------------------------
*: 0.61803398874989480
```

Bad initialization

```
 1.3000000000000000
-0.1344774409873226
-0.2982157033270080
-0.7403273854022190
-2.3674743431148597
-13.8039236412225819
-335.9214859516196157
-183256.0483360671496484
-54338444778.1145248413085938
```

Minimization

- $x \in R^d$ $\quad f: R^d \to R$, twice diff'ble
  - find: $\min_x f(x)$ $\quad 0 = f'(x) \doteq g(x)$ $\quad g: R^d \to R^d$
- Newton: $dx = -(g'(x))^{-1} g(x)$

$= -(f''(x))^{-1} f'(x)$

$= -H^{-1} g$

Descent

- Newton step: d = –(f”(x))$^{-1}$ f’(x)
- Gradient step: –g = –f’(x)
- Taylor: df = $g^T dx$ [+ r(dx)]
- Let t > 0, set dx = $t\,d$
  - df = $-tf'(x)(f''(x))^{-1} f'(x) + r(td)$
    $-tf'(x)(f''(x))^{-1} f'(x)$: >0 iff $f''(x) \succ 0$
- So:

Steepest descent

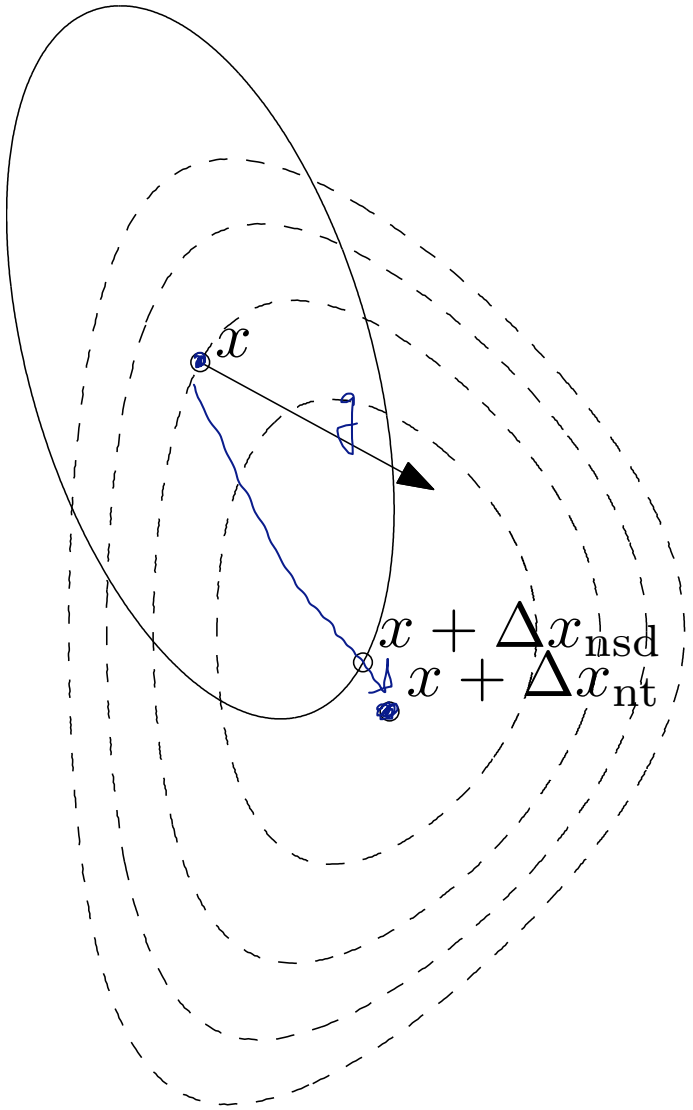
$g = f'(x)$

$H = f''(x)$

$\|d\|_H = \sqrt{d^T H d}$

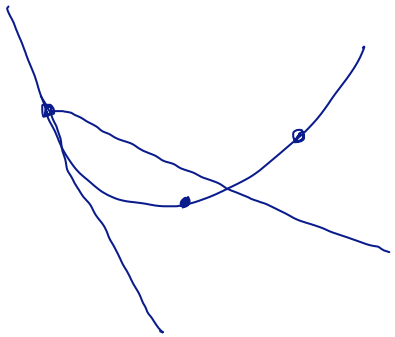
$\min_d \; g^T d + \underbrace{\|d\|_H^2}_{d^T H d}$

$d = -H^{-1} g$



Newton w/ line search

"damped"



- Pick x_1
- For k = 1, 2, …
  - $g_k = f'(x_k)$; $H_k = f''(x_k)$ — *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: $y_i = Ax_i + b \quad \forall i$
- 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*