Admin

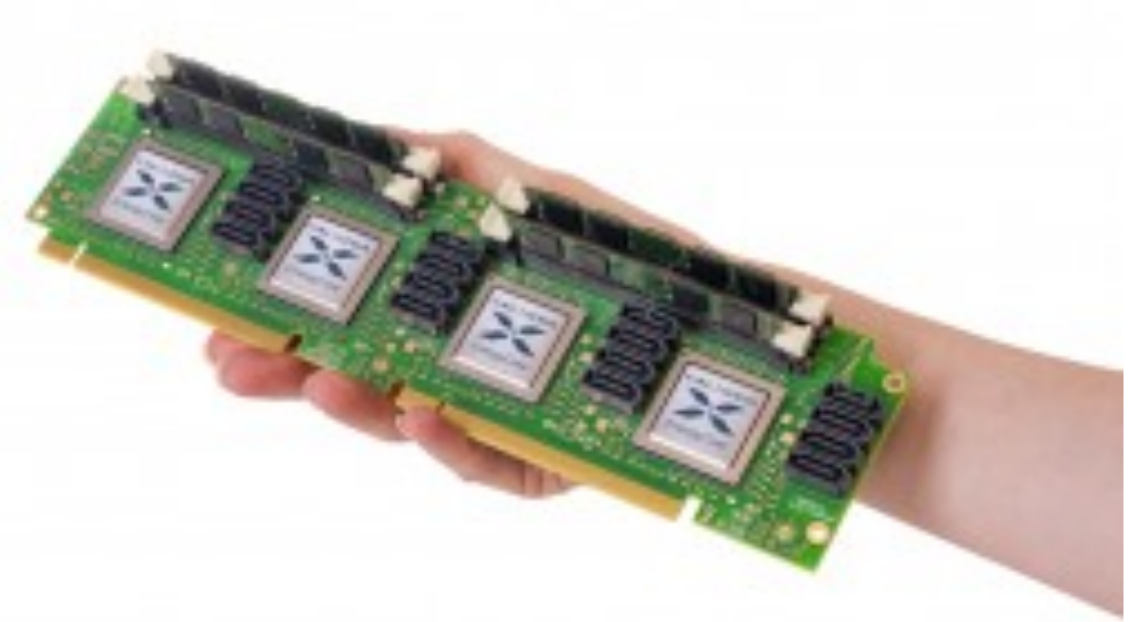
- Project proposal—this Friday 10/11
  - Title
  - Andrew email addresses of participants
  - description (~500–750 words, or equivalent in pics/eqns)
    - dataset—access, contents, what do you hope to learn?
    - what is the first step? possible milestones?
    - minimal and stretch success criteria
- HW2—2 weeks from today—Mon 10/21
- Midterm—10/28 in class

Projects

- Availability of an interesting data set
  - idea for what interesting things are in the data set
  - idea how to get at these things
- We are looking for interactivity
  - not just "run algorithms XYZ on data ABC," but ***interpret results*** and change course accordingly

Project ideas—ML on FAWN

- FAWN = Fast Array of Wimpy Nodes
  - handle highly multithreaded workload by throwing lots of low-energy processors at it, but great inter-node communication
- Calxeda: "Data Center Performance, Cell Phone Power"
  - one box = up to12 boards * 4 SOCs * 4 Cortex A9 cores
  - 192 high-end cell phones
  - Infiniband network
    - 100s of Gbit/s
    - ping time = 100ns (not ms!)



http://www.calxeda.com

MACHINE LEARNING ON FAWN

basically, these are 'high end' ARM processors with a reconfigurable infiniband-like network. so the trade-off between cpu and communication is quite different from what you usually find on your standard EC2 or cluster instance. and this offers new opportunities in terms of making this scale.

Project—wearable accelerometer

- Alex offers to buy hardware (disclaimer: may be different from picture)
- Goal: interpret data
  - segment and decompose observations into motion primitives
  - infer gait changes
  - monitor convalescing patients

http://www.bodymedia.com

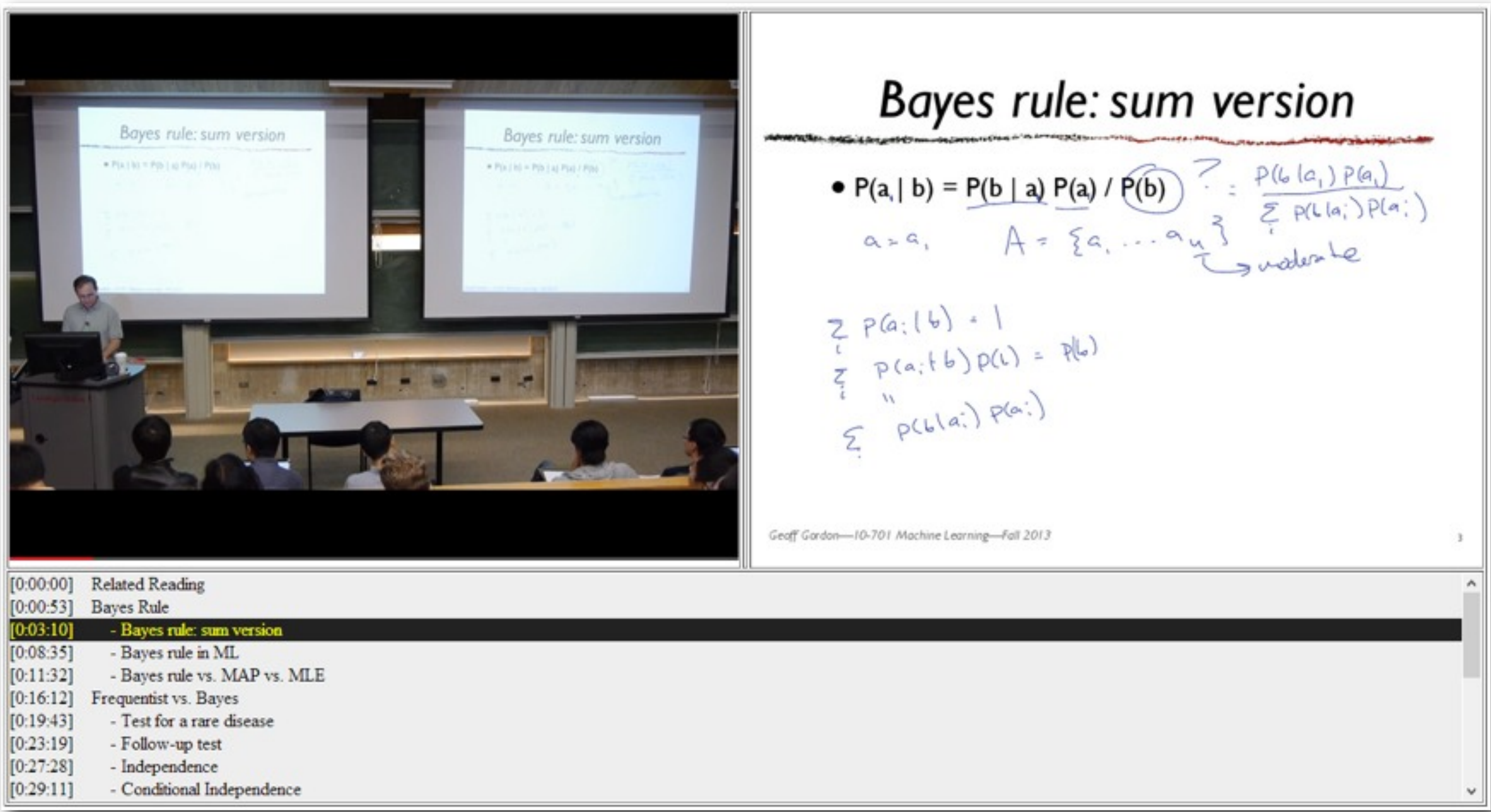
Sunday, October 6, 2013

ACCELEROMETER SENSOR PROJECTS

i'm happy to buy a wearable accelerometer for any team who wants to work on this type of data. basically, the idea is to segment and decompose observations into motion primitives. this can then be used to infer gait changes, e.g. to monitor reconvalescing patients.

just fyi - most commercial devices (fitbit, jawbone up, nike fuel) don't provide raw data.

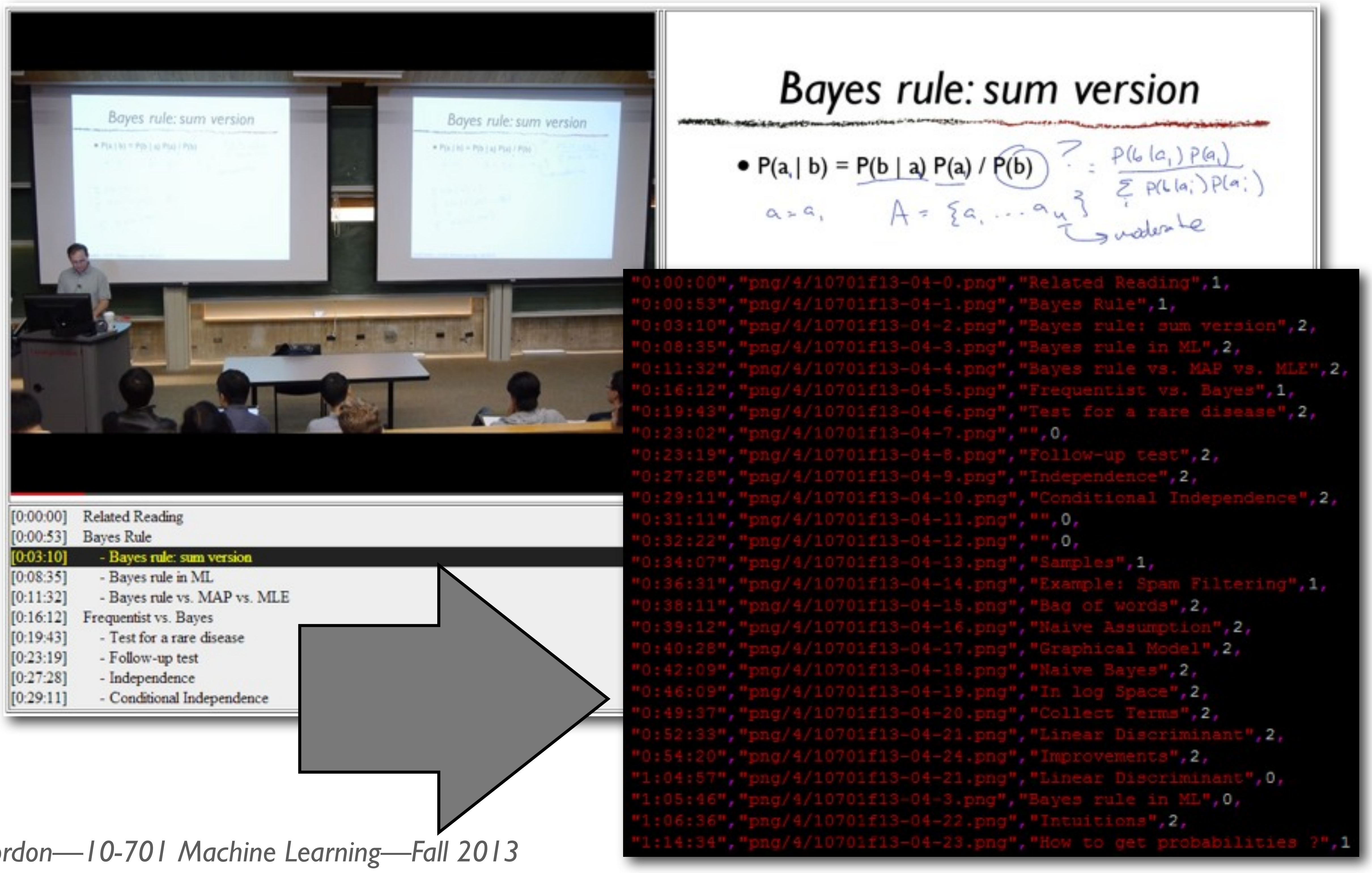
Project—video annotation



Sunday, October 6, 2013

MATCHING VIDEOS W/ SLIDES (Ahmed)
video data from our recorded lectures -- e.g., try to auto-match the video with PDFs of the slides -- also videolectures.net, techtalks.tv

Project—video annotation



Sunday, October 6, 2013

MATCHING VIDEOS W/ SLIDES (Ahmed)
video data from our recorded lectures -- e.g., try to auto-match the video with PDFs of the slides -- also videolectures.net, techtalks.tv

Project—video annotation

- An ML project
  - Can use 3rd party toolboxes to compute features (e.g. OpenCV)—we don't care how you get them
  - Must have a learning component: use annotated lectures for training
    - ours, or scrape videolectures.net, techtalks.tv
- This is a project to satisfy a practical need
  - Your work will be used
  - We will need working, understandable code to be published as open source

Project—educational data

- Watch students interact w/ online tutoring system
- Understand what it is that they are learning, how each student is doing
- Big data set:
  - http://pslcdatashop.web.cmu.edu/KDDCup/
  - I helped run this challenge, so I have ideas about what might work…
- Goals: cluster problems by skills used, cluster students by knowledge of skills

* the KDD cup 2010 data, which is recorded from millions of interactions between thousands of middle-school students and an intelligent tutoring system.

Ed data, revisited

- Or, much smaller data but deeper learning
  - watch a student solve a problem
    - capture pen strokes as they draw diagrams or solve equations—I can provide software/HW for this
  - learn to distinguish solutions from random marks on paper, or eventually good solutions from bad ones
  - what is ***latent structure*** of a solution ("diagram grammar")

Project ideas—Kaggle

- Runs many ML competitions
  - data from StackExchange, cell phone accelerometers, solar energy, household energy consumption, flight delays, molecular activity, …
- Similar idea to challenge problems on our HWs, but less structure, and competing against the whole world
  - CMU is the hardest part of the world to compete against, so you should have no trouble…

Project ideas—Twitter

- Get a huge pile of tweets
- Build a network
- Analyze the network
- Learn something
  - topics, social groups, hot news items, political disinformation (“astroturf”), …

Others

- Loan repayment probability
- Grape vine yield
- Neural data: MEG, EEG, fMRI, spike trains
- Music: audio or MIDI
- …

neural response data -- for anything from fMRI to MEG to spike trains, there are people around who we can get it from. A fascinating problem is to look at a natural stimulus (image, audio, text, movie, ...) and correlate it with what the brain does when a subject experiences that stimulus.

Step back and take stock

- Lots of ML methods:

linear regression
logistic regression
Parzen windows
Watson Nadaraya
k nearest neighbor
naive Bayes
perceptron
kernel perceptron

Step back and take stock

- Lots of ML methods:

Sunday, October 6, 2013

linear regression
logistic regression
Parzen windows
Watson Nadaraya
k nearest neighbor
naive Bayes
perceptron
kernel perceptron

Common threads

- Machine learning principles (MLE, Bayes, …)
- Optimization techniques (gradient, LP, …)
- Feature design (bag of words, polynomials, …)

Goal: you should be able to mix and match by turning these 3 knobs to get a good ML method for a new situation

Machine learning principles

- MLE: "a model that fits training set well (assigns it high probability) will be good on test set"
- regularized MLE: "even better if model is 'simple'"
- MAP: "want the most probable model given data"
- Bayes: "average over all models according to their probability"

all similar but not same

MLE: max_model P(data | model) [equivalently, min log P(data|model)]

reg MLE : min_model log P(data|model) + penalty(model)
"simple" = low penalty

MAP: min_model log P(data | model) + log P(model)
note: log P(data) is constant, so might as well as −log P(data)

Bayes rule P(model|data) = P(data|model) P(model)/P(data)

reg MLE vs. MAP: no need for penalty to be log(P(model))

MAP vs. Bayes: optimization vs integration

More principles

- Nonparametric: "future data will look like past data"

- Empirical risk minimization: "a simple model that fits our training set well (assigns it low E(loss)) will be good on our test set"

ERM: min sum_i loss(ex_i; model) + penalty(model)
or equivalently: min sum_i loss(ex_i; model) s.t. penalty(model) ≤ k
    didn't define ERM officially before now, but above eqns are definition

similar to reg. MLE and MAP, but no need for loss or penalty to be log probabilities

To get guarantees, need to limit size of parameter set optimized over
    we'll get to how later in course;
    e.g., regression: limit norm of weights based on size of training set

Examples

- linear regression (Gaussian errors)
- linear regression (no error assumption)
- ridge regression
- k-nearest neighbors
- Naive Bayes for text classification
- Watson Nadaraya
- Parzen windows

linear/Gaussian regression: MLE
linear regression, no err assumption: ERM
ridge regression: MAP or penalized MLE
k-nn: nonparametric
NB: Bayes
WN: chains nonparametric density est. w/ Bayes rule
Parzen: nonparametric

others:
perceptron (online ERM)
P(word|class) in naive Bayes (Laplace smoothing = Bayes)
logistic reg: MLE (or pen. MLE or MAP for L1/L2)
LASSO: pen. MLE or MAP

examples we haven't covered yet: graphical models, LDA, Bayes regression, kernel mean maps, SVMs

Selecting a principle

- Computational efficiency vs. data efficiency vs. what we're willing to assume
  - e.g., full Bayesian integration is often great for small data, but really expensive to compute
  - e.g., for huge # of examples and high-d parameter space, stochastic gradient may be the only viable option
  - e.g., if we're not willing to make strong assumptions about data distribution, suggests nonparametric or ERM
- Often wind up trying several routes
  - e.g., to see which one leads to a tractable optimization

ERM can even allow non-i.i.d. data

Common thread: optimization

- Use a principle to derive an objective fn
  - hopefully convex, often not
- Select algorithm to min or max it
  - or sometimes integrate it—like optimization, but harder

integration is kind of like optimization: need to find places where integrand is big
but harder: e.g., #P vs NP

Optimization techniques

- If we're lucky: set gradient to 0, solve analytically
- (Sub)gradient method
    - analyzed this one: –log(error) = O(# iters) (bad constant)
- Stochastic (sub)gradient method
- Newton's method
- Linear prog., quadratic prog., SOCPs, SDPs, …
- Other: EM, APG, ADMM, …

also mentioned: bundle methods, distributed (sub)gradient

Comparison

of techniques for minimizing a convex function

	Newton	APG	(sub)grad	stoch. (sub)grad.
convergence				
cost/iter				
assumptions				

conv:	*****	***	*/**/***	*
cost:	$$$$$	$$$	$$	$
assume:	++++	++	+/++/+++	+

Newton: fast convergence [ln 1/eps = O(k^2)]; expensive iterations (gradient, Hessian, linear solve); strongest smoothness requirements (2 derivatives, self-concordance or Hessian bounds)

accelerated gradient: cheaper iterations (gradient & prox); weaker smoothness (Lipschitz continuous gradient (LCG), but only for data-dependent part of objective); convergence 1/eps=O(k^2) (or exp(O(k)) for strongly convex)

(sub)gradient: cheaper iterations, slower convergence, weakest smoothness requirements

1/eps = O($\sqrt{k}$) w/o LCG

1/eps = O(k) if LCG

1/eps = exp(O(k)) if strongly convex (but with huge constant)

stochastic (sub)gradient: cheapest iterations, slowest converence [1/eps = O($\sqrt{k}$)], weakest smoothness requirements

Common thread: features

- Customer/collaborator/boss hands you SQL DB
- You need to turn it into valid input for one of these algorithms
  - discarding outliers, calculating features that encapsulate important ideas
- Options:
  - finite-length vector of real numbers
  - kernels: infinite feature spaces; strings, graphs, trees, etc.

kernels are cool, but need some effort to kernelize

only way to teach featurization is by example:
text -> bag of words,
real #s -> [logs, low-order polys, ...],
audio -> spectrogram,
image -> [pixels, SIFT, optical flow, ...],
social network -> graph
(note: haven't given any graph algos yet)

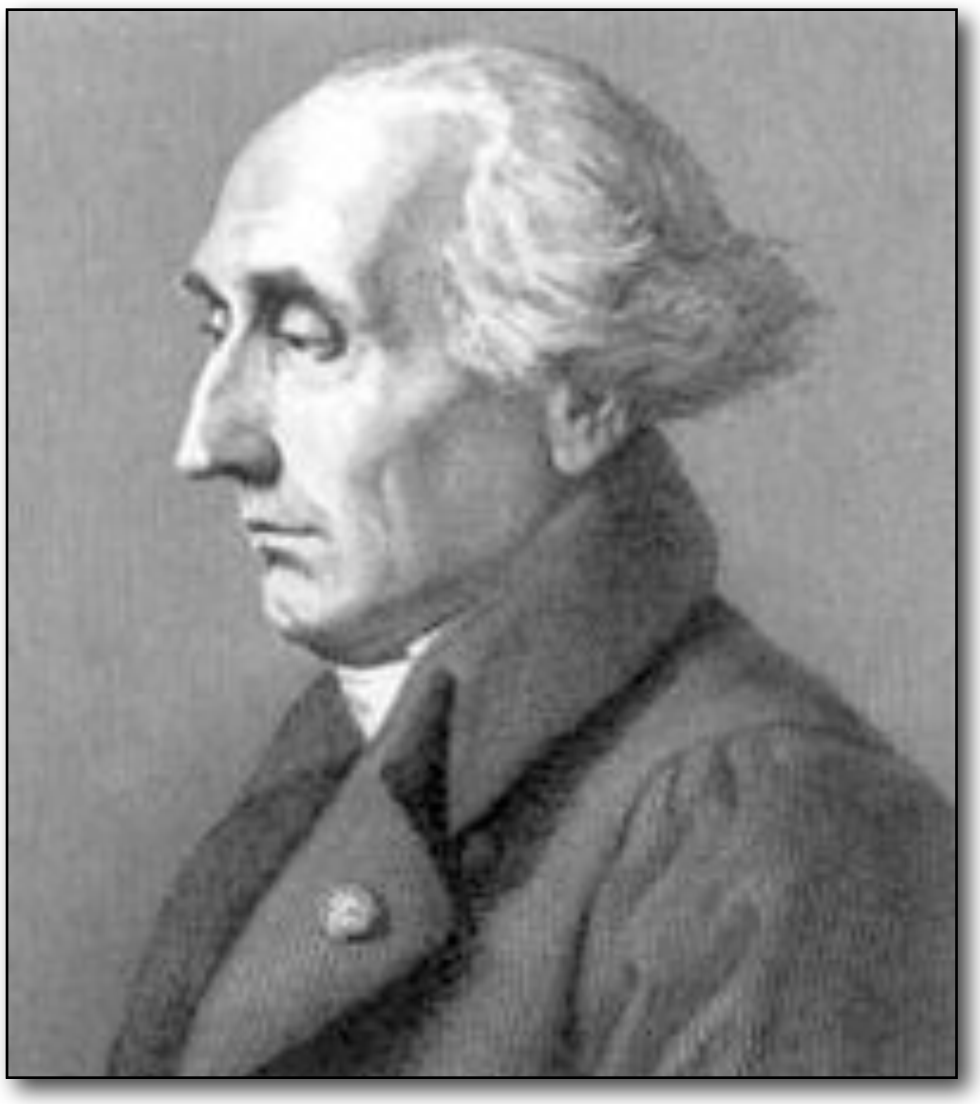
HW challenge probs: practice in finite-length-vector feature engineering, two very different input datasets

Where does it all lead?

- Different principles, assumptions, optimization techniques, feature generation methods lead to different algorithms for same qualitative problem (e.g., many algos for "regression")
- Different principles can give same/similar algos
  - linear regression as conditional Bayes under Gaussian errors, or as ERM under square loss
  - many different linear classifiers: perceptron, NB, logistic regression, SVM, …

Lagrange multipliers

- Technique for turning constrained optimization problems into unconstrained ones



Recall: Newton's method

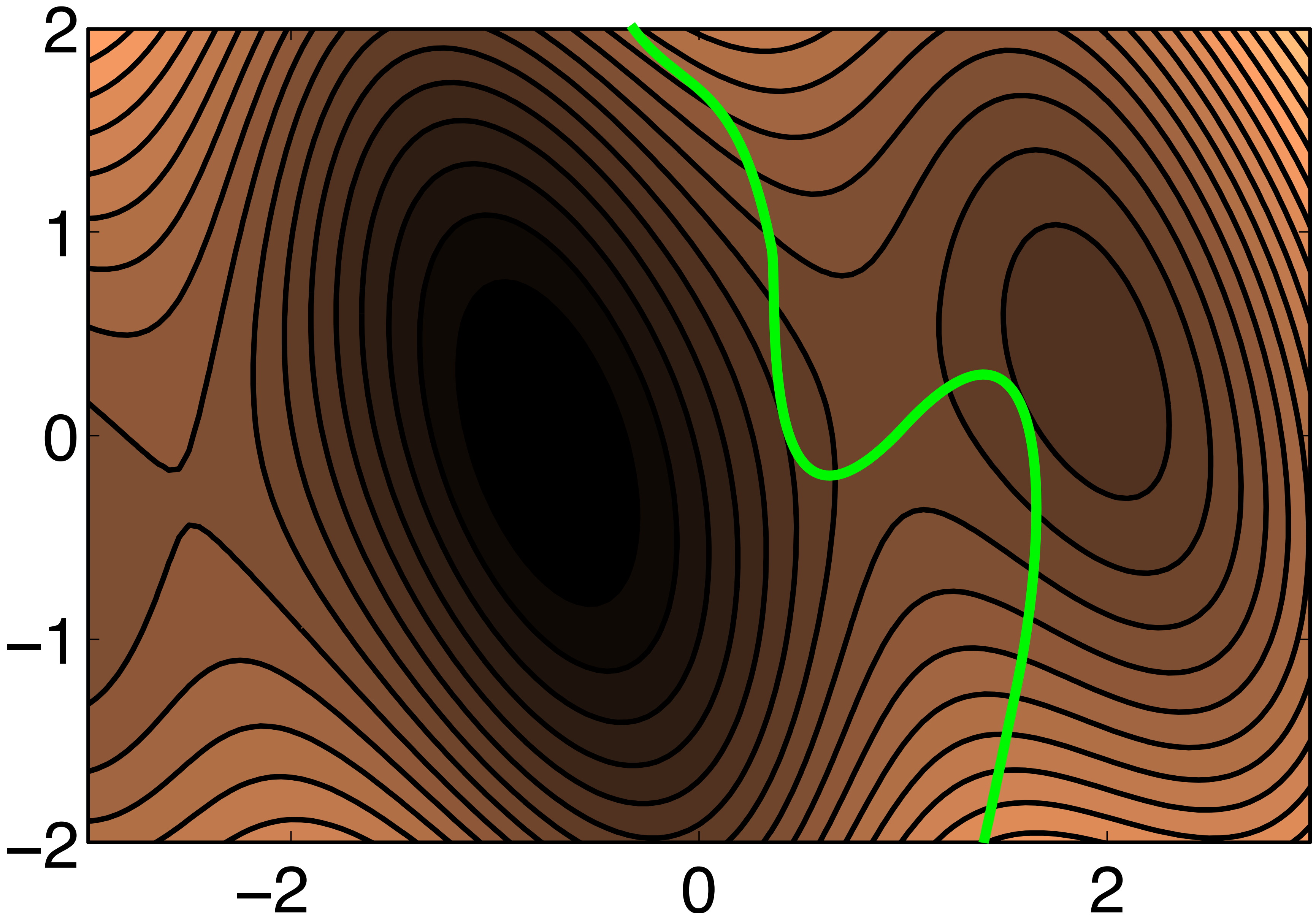
- $\min_x f(x) \rightarrow$
  - $f: R^d \rightarrow R$

w/o constr: H(x) Δx + g(x) = 0
    g(x) = gradient R^d->R^d
    H(x) = Hessian R^d->R^{d*d}

why: f(x+Δx) ~ f+g'Δx+Δx'HΔx/2
f = f(x), g = g(x), H = H(x)
set derivative wrt Δx to 0:
    0 = g + HΔx
HΔx is predicted change in gradient, use it to cancel g

Equality constraints

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



f(x): contours
p(x)=0: line

quiz: where are the local optima?
A: places where gradient f'(x) is normal to the curve (can't slide L or R to decrease fn)
i.e., f'(x) = lambda p'(x)
draw: they are places where contours of f are tangent to p=0

lambda = "Lagrange multiplier" -- multiplies constraint normal to scale it to match gradient

Optimality w/ equality

- min f(x) s.t. p(x) = 0
  - f: $R^d \to R$ p: $R^d \to R^k$ ($k \leq d$)
  - g: $R^d \to R^d$ H: $R^d \to R^{d \times d}$ (gradient, Hessian of f)
- Useful special case: min f(x) s.t. Ax = b

def: C = {x | Ax = b}

How do we express g(x) ⊥ C?
    z ⊥ C iff z'(x-y) = 0 forall x,y in C
    idea: z = A'lambda
    then z'(x-y) = lambda' A(x-y)
     = lambda'(b-b) = 0.
necessary & sufficient (count dimensions)

So, want g(x) = A' lambda.
    ie, gradient = linear combo of rows of A

===
How do we know A' lambda is a full basis? A' lambda is a space of rank(A) dimensions; Ax = 0 is a space of nullity(A) dimensions; rank + nullity is the full dimension of the space, so we've accounted for every dimension as either free to vary under the constraint or orthogonal to the constraint.

More generally

g(x) = J(x)^T lambda
J_{ij} = dh_i/dx_j
ie, gradient = lin. comb. of constraint normals

J: R^d --> R^{k*d}

h(x) = Ax - b --> J(x) = A

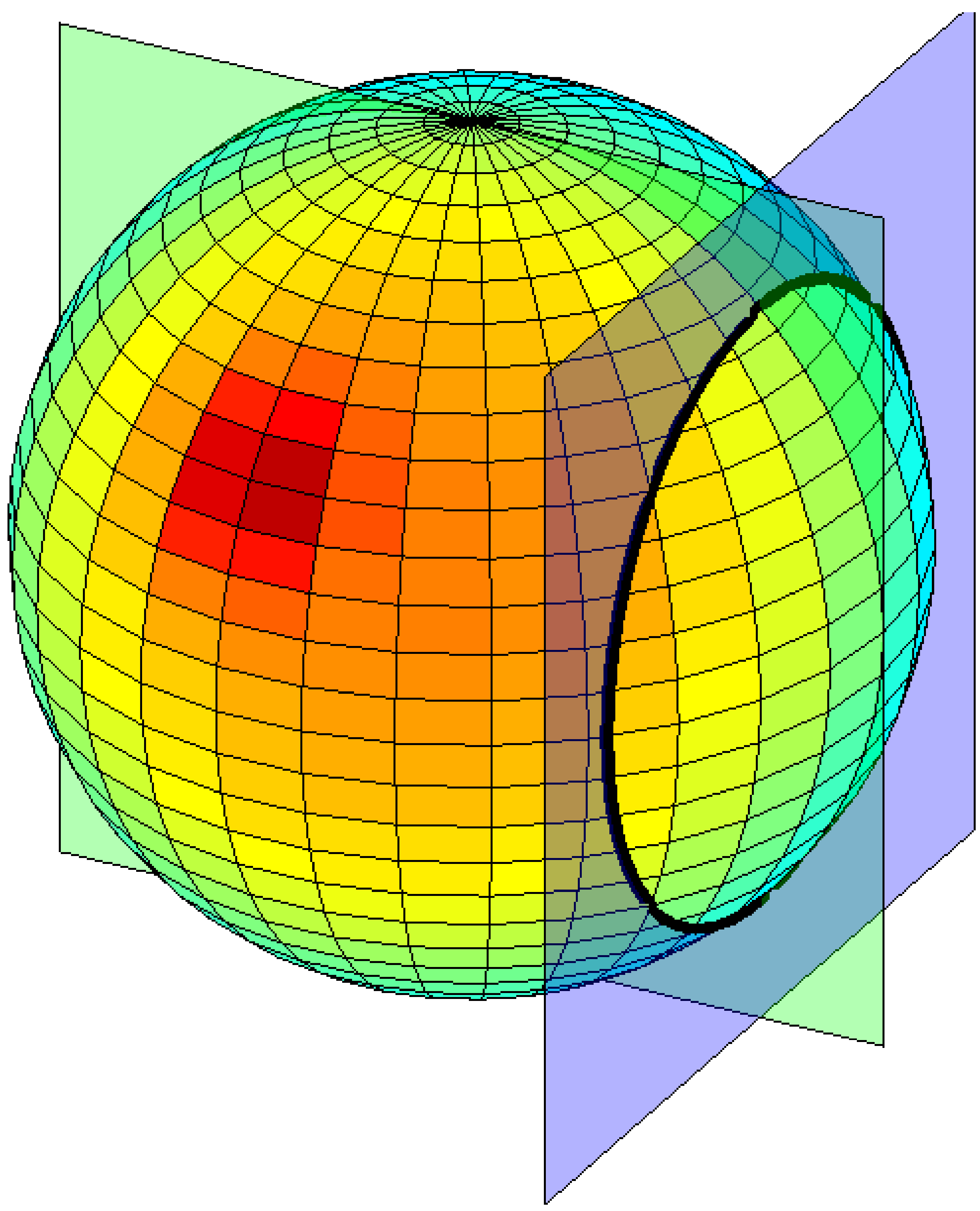
===
another way to think of it: cancel out the portion of gradient orthogonal to p(x)=0 using best lambda. Remainder is projection of gradient onto constraint.

Picture

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

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

$$a^\top x = b$$



Sunday, October 6, 2013

c: pointing up
constraints: sphere, blue plane
(intersection = dark circle)

Constraint normals: 2[x y z], a
So, at opt:

    c = 2 lam1 [x y z] + lam2 a
    x^2 + y^2 + z^2 = 1
    a'x = b

(green plane = span of normals @ optimum)

===

max z s.t.
x^2 + y^2 + z^2 = 1
x = .7

opt: x = .7, y = 0, z = √.51
constraint normals: 2*[.7 0 √.51], [1 0 0]
lam2 = –lam1
lam2 = 1/(2√.51)

===

```
>> [x, y, z] = sphere(30); h = surfl(x, y, z); axis equal off; set(gca, 'fontsize', 24); h = patch(.7*[1 1 1 1], [1 1 -1 -1], [1 -1 -1 1], 'b'); set(h, 'facealpha', .3)

>> [ex, ey] = ellipse([0;0], eye(2), 50); r = sqrt(1-.7^2); line(.7*ones(size(ex)), ex*r, ey*r,
```

Newton w/ equality

- min f(x) → $H(x)\Delta x = -g(x)$
- min f(x) s.t. p(x) = 0
  - f: $R^d \to R$,   p: $R^d \to R^k$
- Now suppose:
  - dg/dx =   dp/dx =
- Optimality:

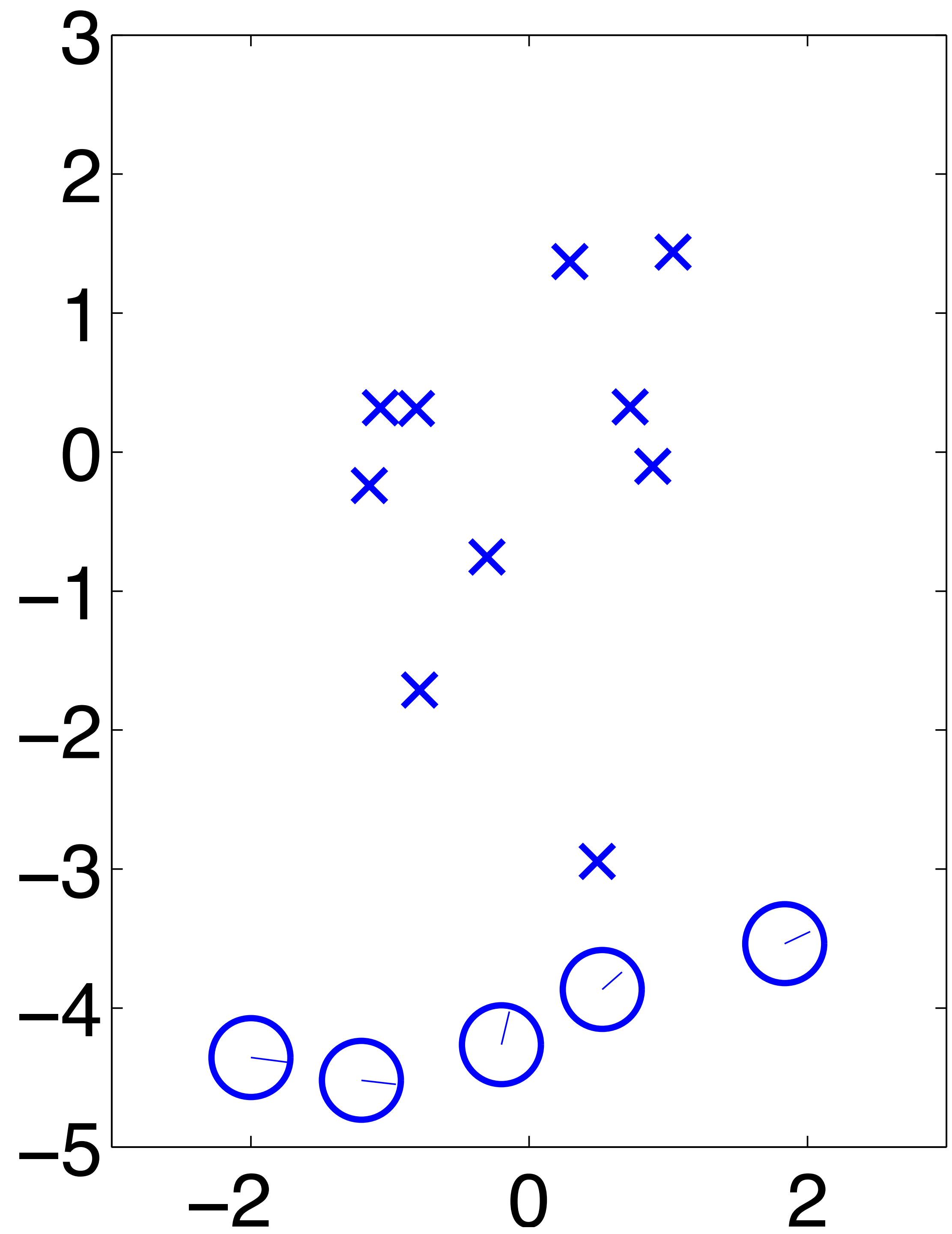
Sunday, October 6, 2013

Now suppose
    dg/dx = H(x) [Jacobian of g = Hessian of f]
    dp/dx = J(x) [Jacobian of p]
    [sizes: H is d*d, J is k*d]
First-order approx of constraint:
    $p(x) + J(x)\Delta x = 0$
First-order approx of optimality conditions:
    $H(x)\,\Delta x + g(x) = J(x)^T\lambda$
    LHS: predicted gradient after update Δx
    RHS: orthogonal to (approx) constraint

Newton step:
    [H –J'; J 0] [Δx; λ] = [–g; –p]
N = [H –J'; J 0] is (k+d)*(k+d), PSD if H is

Ex: bundle adjustment for SLAM

- Solve for:
  - Robot positions x_t, θ_t
  - Landmark positions y_k
- Given: odom., radar, vision, …
- Constraints:
  - observations consistent w/ map



Sunday, October 6, 2013

xt, yk in R^2
theta_t in [-pi,pi]

example: distance measurements d_{kt}
||x_t - y_k||^2 = d_{kt}^2 + noise
(min |noise| goes in objective)