RANDOM TOPICS

stochastic gradient descent
&
Monte Carlo

MASSIVE MODEL FITTING

$$\text{minimize} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

Big!
(over 100K)

least squares

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|^2 = \sum_i \frac{1}{2}(a_i x - b_i)^2$$

SVM

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + h(LDw) = \frac{1}{2}\|w\|^2 + \sum_i h(l_i d_i w)$$

low-rank factorization

$$\text{minimize} \quad \frac{1}{2}\|D - XY\|^2 = \sum_{ij} \frac{1}{2}(d_{ij} - x_i y_j)^2$$

THE BIG IDEA

$$\text{minimize} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

True gradient

$$\nabla f = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x)$$

Idea: choose a subset of the data

$$\Omega \subset [1, N]$$

usually just one sample

$$\nabla f \approx g_\Omega = \frac{1}{|\Omega|}\sum_{i\in\Omega} \nabla f_i(x)$$

INFINITE SAMPLE VS FINITE SAMPLE

finite sample

$$\text{minimize} \quad f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

We can solve finite sample problem to high accuracy….

infinite sample

…but true accuracy is limited by sample size

$$\text{minimize} \quad f(x) = \mathbf{E_s}[f_s(x)] = \int_s f_s(x)p(s)ds$$

SGD

select data

compute gradient

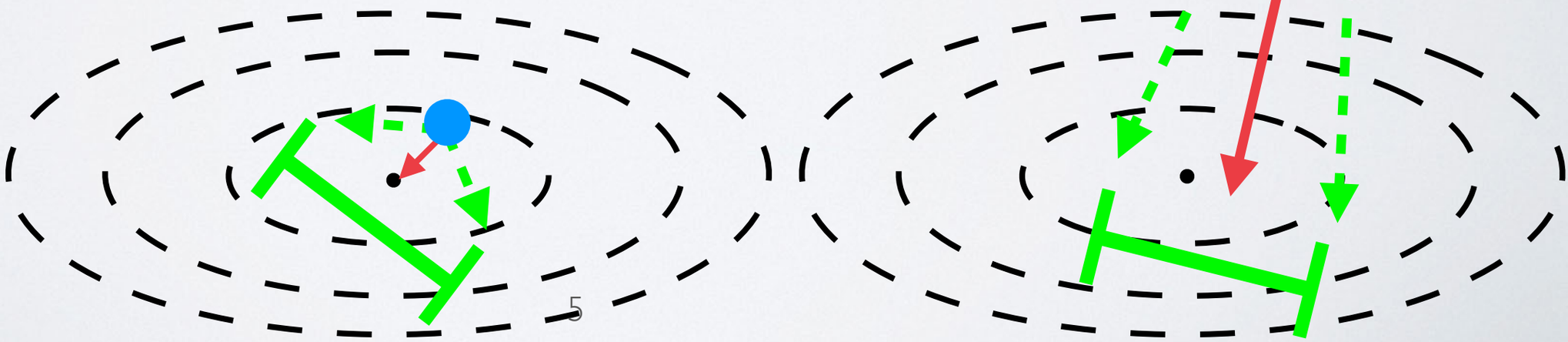
$$g^k \approx \frac{1}{M} \sum_{i=1}^{M} \nabla f(x^k, d_i)$$

update

$$x^{k+1} = x^k - \tau_k g^k$$

small error
solutions gets worse

big error
solution improves



SGD

select data

compute gradient

$$g^k \approx \nabla f(x, d_8)$$

update

$$x^{k+1} = x^k - \tau_k g^k$$

Error must decrease
as we approach solution

classical solution

shrink stepsize

$$\lim_{k \to \infty} \tau_k = 0$$

slow convergence

$$O(1/\sqrt{k})$$

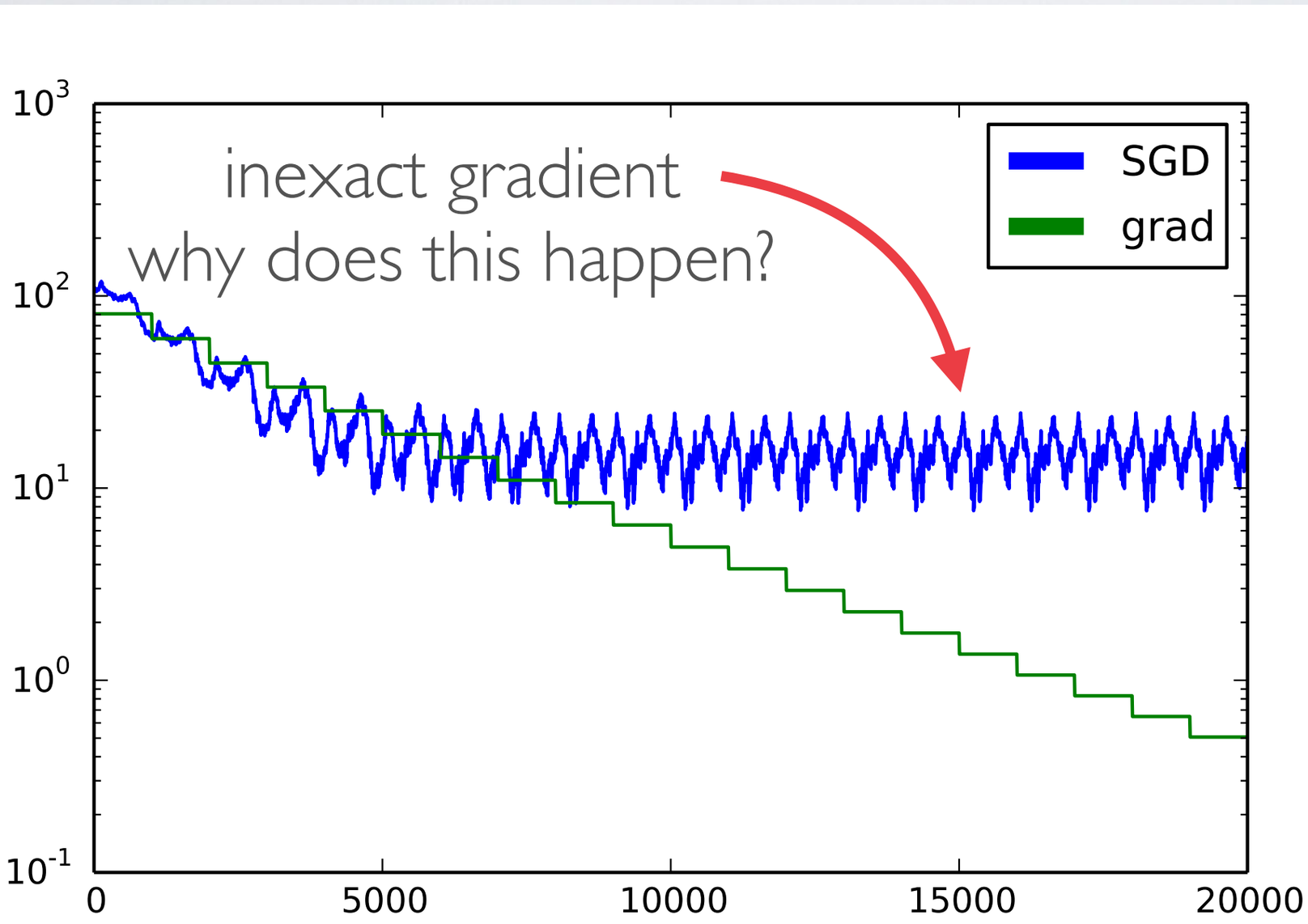
Variance reduction

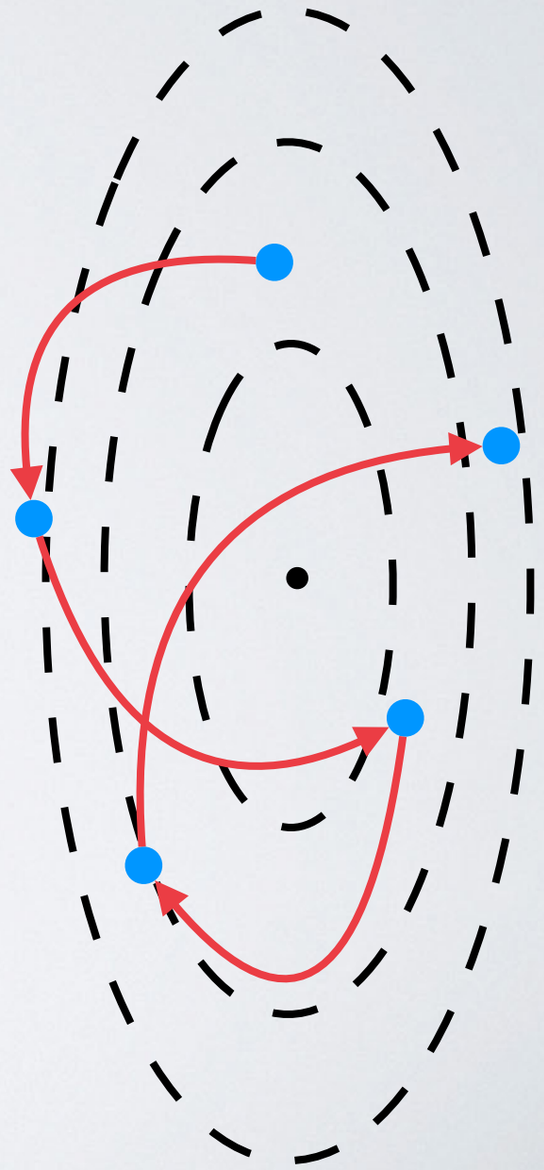
Correct error in gradient approximations

EXAMPLE: WITHOUT DECREASING STEPSIZE

$$\text{minimize} \quad \frac{1}{2}\|Ax - b\|^2$$

what's happening?





DECREASING STEP SIZE

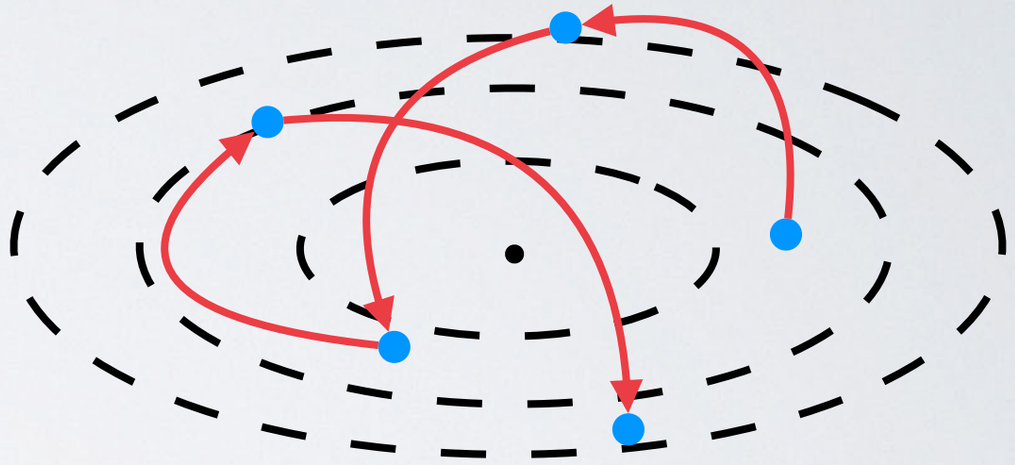
$$x^{k+1} = x^k - \tau_k \nabla f_k(x)$$

$$\tau_k = \frac{a}{b+k}$$ why?

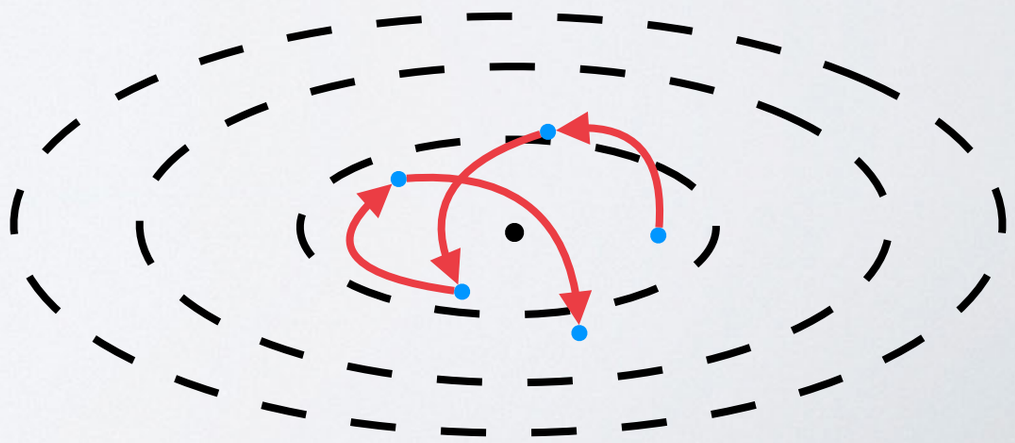
(almost) equivalent to
choose larger sample

used for
strongly convex problems

big stepsize



small stepsize



AVERAGING

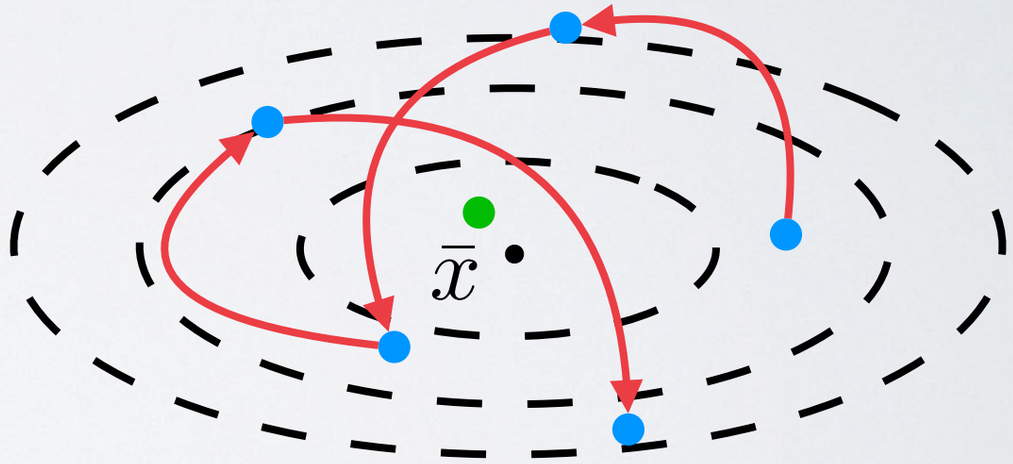
$$x^{k+1} = x^k - \tau_k \nabla f_k(x)$$

$$\tau_k = \frac{a}{\sqrt{k+b}}$$

averaging

"ergodic" averaging

$$\bar{x}^{k+1} = \frac{1}{k+1} \sum x_i$$



why is this bad?

used for
weakly convex problems

does this limit convergence rate?

AVERAGING

ergodic averaging

$$\bar{x}^{k+1} = \frac{1}{k+1}\sum x_i$$

compute without storage

$$\bar{x}^{k+1} = \frac{k}{k+1}\bar{x}^{k} + \frac{1}{k+1}x^{k+1}$$

short memory version

$$\bar{x}^{k+1} = \frac{k}{k+\eta}\bar{x}^{k} + \frac{\eta}{k+\eta}x^{k+1}$$

tradeoff variance for bias

averaging

$\eta \geq 1$

KNOWN RATES: WEAKLY CONVEX

See

Shamir and Zhang, ICML '13

Rakhlin, Shamir, Sridharan, CoRR '11

Theorem

Suppose f is convex, $\|\nabla f(x)\| \leq G$, and that the diameter of $\mathbf{dom}(f)$ is less than D. If we use the stepsize

$$\tau_k = \frac{c}{\sqrt{k}},$$

then

$$\mathtt{E}[f(\bar{x}_k) - f^\star] \leq \left(\frac{D^2}{c} + cG\right) \frac{2 + \log(k)}{\sqrt{k}}$$

KNOWN RATES: STRONGLY CONVEX

Shamir and Zhang, ICML '13

Theorem

Suppose f is strongly convex with parameter m, and that $\|\nabla f(x)\| \leq G$. If you use stepsize $\tau_k = 1/mk$, and the limited memory averaging

$$\bar{x}^{k+1} = \frac{k}{k+\eta}\bar{x}^k + \frac{\eta}{k+\eta}x^{k+1}$$

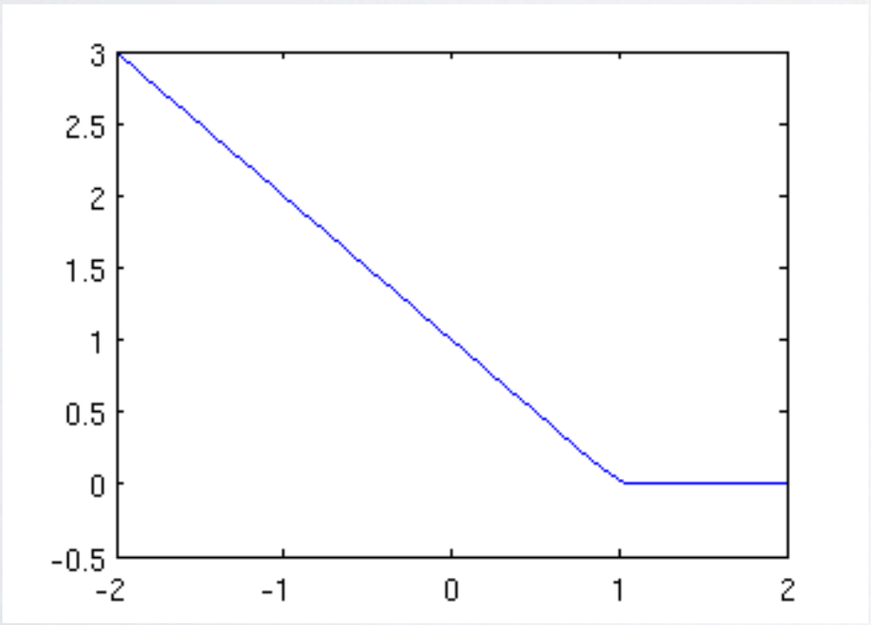
with $\eta \geq 1$, then

$$\mathtt{E}[f(\bar{x}_k) - f^\star] \leq 58(1+\eta/k)\left(\eta(\eta+1) + \frac{(\eta+.5)^3(1+\log k)}{k}\right)\frac{G^2}{mk}$$

EXAMPLE: SVM

PEGASOS: Primal Estimated sub-GrAdient SOlver for SVM

$$\text{minimize} \quad \frac{1}{2}\|w\|^2 + Ch(Aw)$$



$$\nabla h(x) = \begin{cases} -1, & x < 1 \\ 0, & \text{otherwise} \end{cases}$$

note: this is a "**subgradient**" descent method

PEGOSOS

PEGASOS: Primal Estimated sub-GrAdient SOlver for SVM

$$\text{minimize} \quad \sum_i \frac{\lambda}{2}\|w\|^2 + h(a_i w)$$

While "not converged":

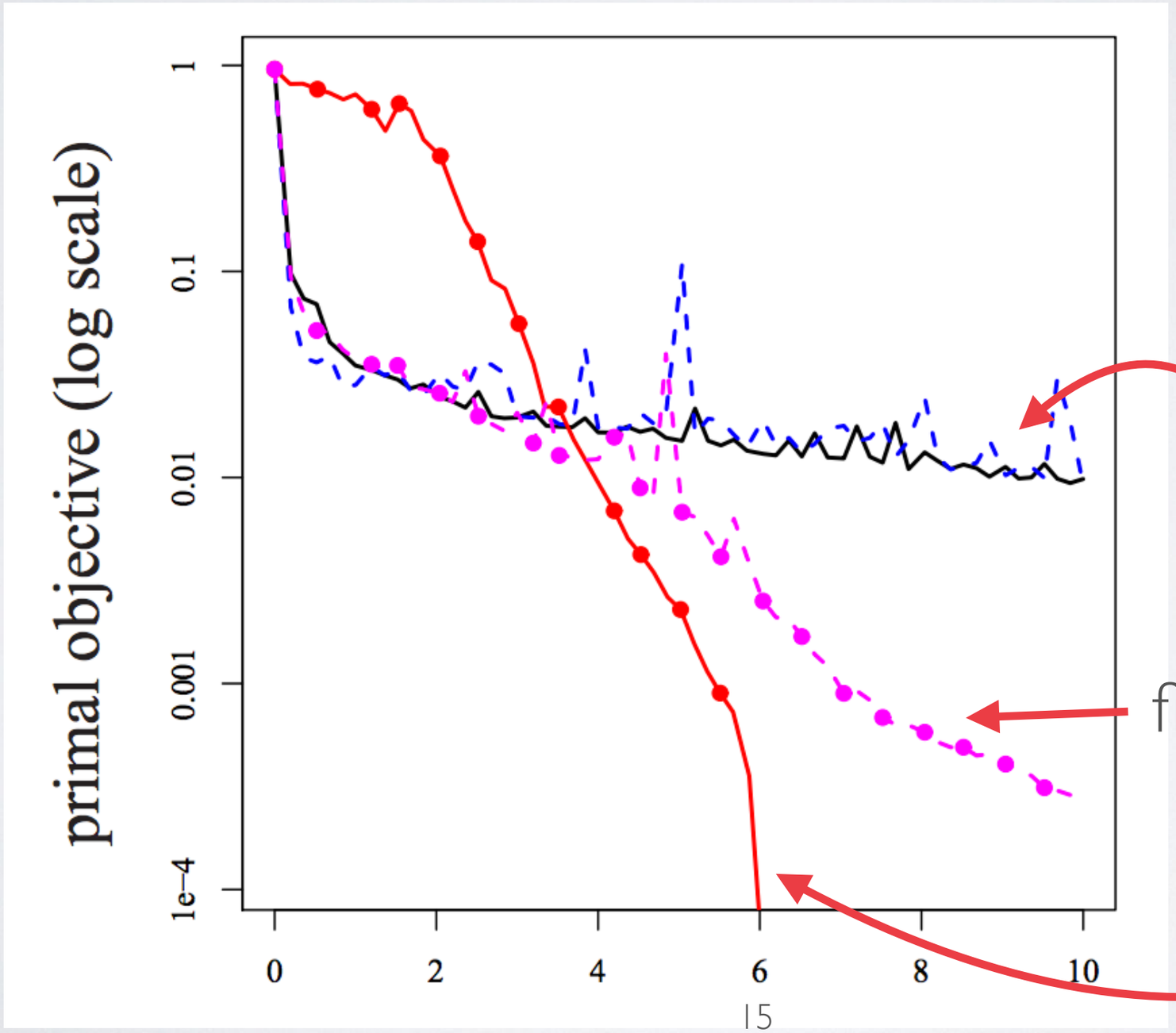
$$\tau^k = \frac{1}{\lambda k}$$

$$\text{If } a_k^T w^k \geq 1: \quad w^{k+1} = w^k - \tau^k \lambda w^k$$

$$\text{If } a_k^T w^k < 1: \quad w^{k+1} = w^k - \tau^k(\lambda w^k - a_i^T)$$

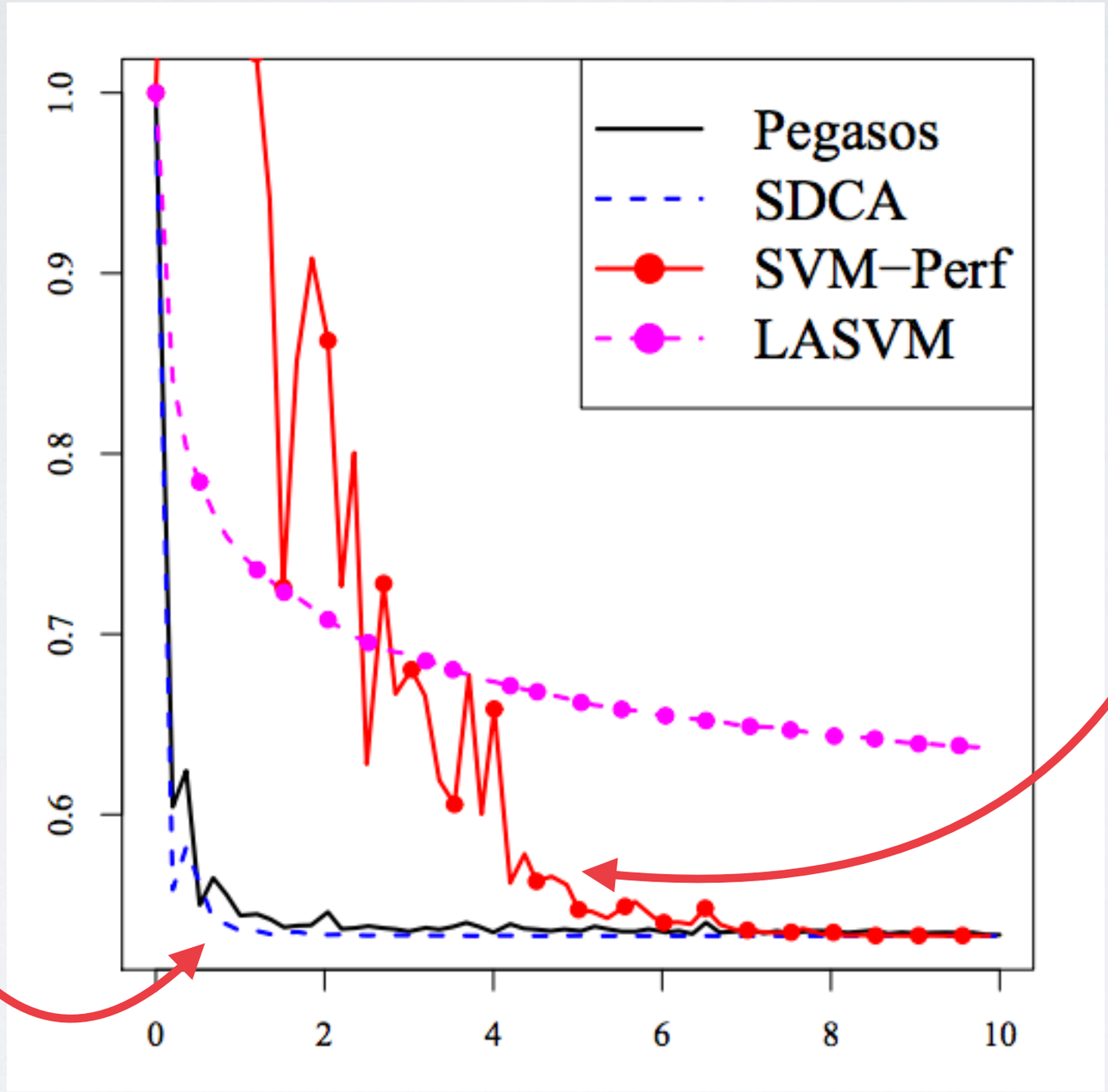
$$\hat{w}^{k+1} = \frac{1}{k+1}\sum_{i=1}^{k+1} w^k$$ ← not used in practice

PEGOSOS


primal objective (log scale)
1
0.1
0.01
0.001
1e−4
0
2
4
6
8
10
Pegasos
SDCA
SVM−Light
LASVM
stochastic methods
finite sample method
gradient method

PEGOSOS

classification error (CV)



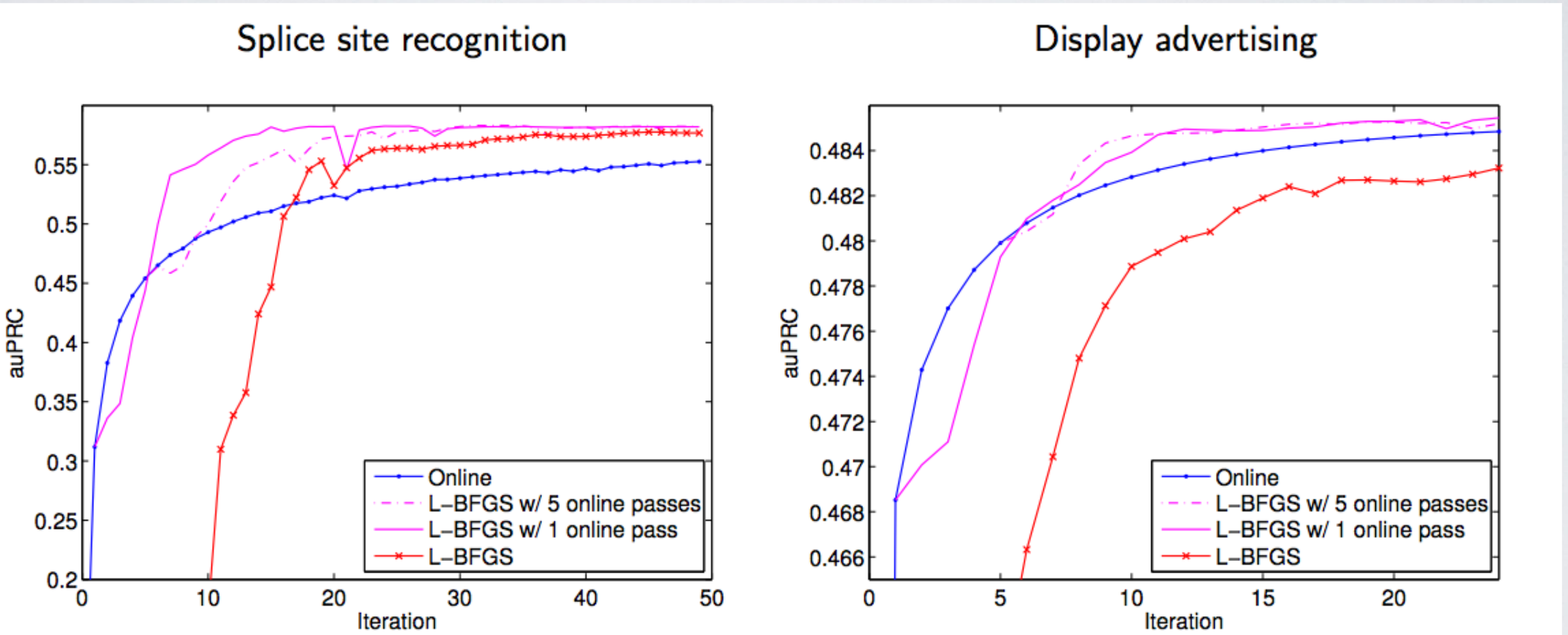
gradient method

stochastic methods

You **hope** to converge before SGD gets slow!

ALLREDUCE

but if you don't…
use SGD as warm start for iterative method



Agarwal. "Effective Terascale Linear Learning." '12

SGD

select data

compute gradient

$$g^k \approx \nabla f(x, d_8)$$

update

$$x^{k+1} = x^k - \tau_k g^k$$

Error must decrease
as we approach solution

variance reduction solution

make gradient more accurate

preserve fast convergence

SGD+VARIANCE REDUCTION

select data **compute gradient** **update**

$$g^k \approx \nabla f(x, d_8) - \text{error}^8 \rightarrow x^{k+1} = x^k - \tau_k g^k$$

Error must decrease
as we approach solution

variance reduction solution

make gradient more accurate

preserve fast convergence

VR APPROACHES

SVRG

Johnson, Zhang, 2013

The original: requires full gradient computations

SAGA

Defazio, Bach, Lacoste-Julian, 2014

SAG

Le Roux, Schmidt, Bach, 2013

Avoid full gradient computations

many more…

shameless self-promotion

Central VR

A VR approach targeting **distributed** ML

"Efficient Distributed SGD with Variance Reduction," ICDM 2016

SVRG

gradient tableau

$\nabla f_1(x_m^1)$
$\nabla f_2(x_m^2)$
$\nabla f_3(x_m^3)$
$\vdots$
$\nabla f_{n-1}(x_m^{n-1})$
$\nabla f_n(x_m^n)$

First epoch

SVRG

gradient tableau

$\nabla f_1(x_m^1)$
$\nabla f_2(x_m^2)$
$\nabla f_3(x_m^3)$
$\vdots$
$\nabla f_{n-1}(x_m^{n-1})$
$\nabla f_n(x_m^n)$

Average gradient over last epoch

$$\overline{g}_m = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_m^i)$$

SVRG

gradient tableau

$\nabla f_1(x_m^1)$
$\nabla f_2(x_m^2)$
$\nabla f_3(x_m^3)$
$\vdots$
$\nabla f_{n-1}(x_m^{n-1})$
$\nabla f_n(x_m^n)$

Average gradient over last epoch

$$\bar{g}_m = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(x_m^i)$$

corrected gradient

$$\nabla f_3(x_{m+1}^3) \; - \; (\nabla f_3(x_m^3) - \bar{g}_m)$$

new gradient

error

SVRG GETS BACK DETERMINISTIC RATES

Johnson and Zhang, 2013

Theorem

Suppose of objective terms are m strongly convex with L Lipschitz gradients. If the learning rate is small enough, then

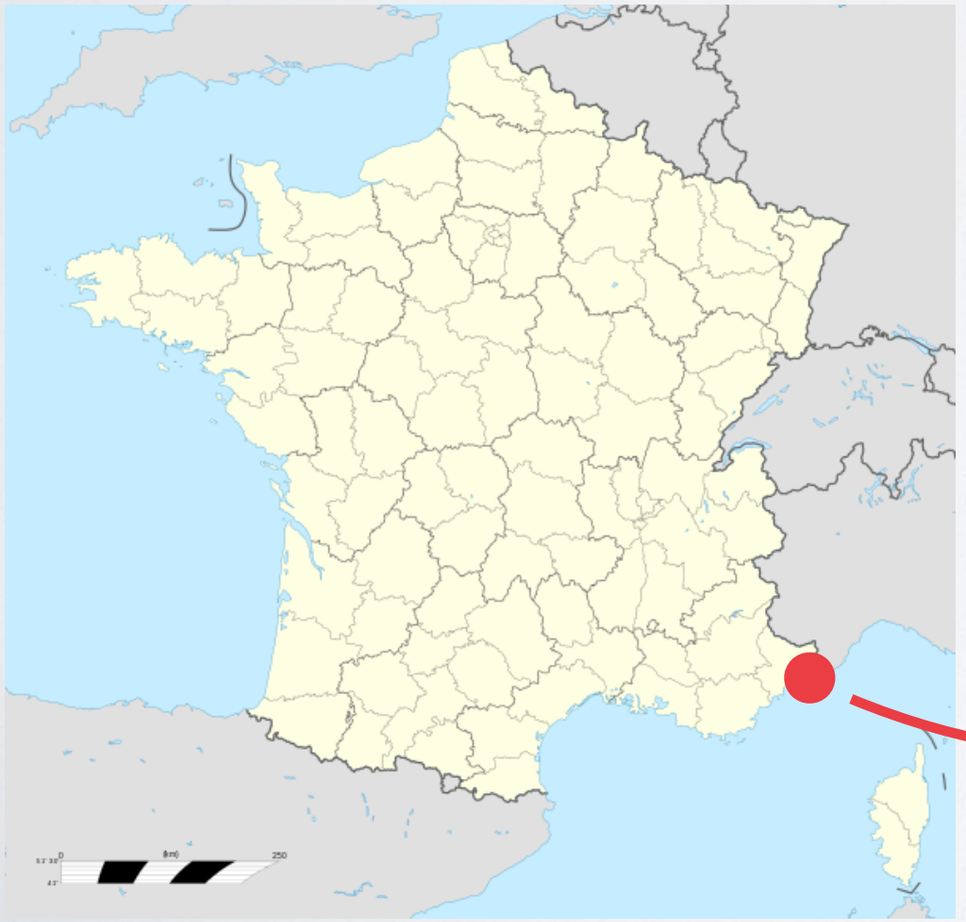
$$\mathbb{E}f(w^k) - f^\star \leq c^k[f(w^0) - f^\star]$$

for some $c < 1$.

MONTO CARLO METHODS

Methods that involve randomly sampling a distribution

Monte Carlo casino



Monaco?

BAYESIAN LEARNING

estimate parameters in a model

$$data = M(parameters)$$

measure these

estimate these

ingredients...

prior

probability distribution
of unknown parameters

likelihood

probability of data
given (unknown) parameters

Thomas Bayes

BAYES RULE

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

likelihood

prior

$$P(\text{parameters}|\text{data}) = \frac{P(\text{data}|\text{parameters})P(\text{parameters})}{P(\text{data})}$$

we really care about this

$$P(\text{parameters}|\text{data}) \propto P(\text{data}|\text{parameters})P(\text{parameters})$$

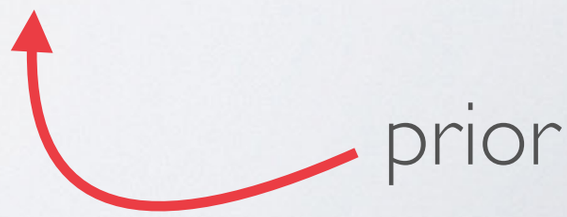
"posterior distribution"

EXAMPLE: LOGISTIC REGRESSION

$$P(y_i = 1|x_i) = \frac{\exp(x_i^T w)}{1 + \exp(x_i^T w)}$$

$$P(y, X|w) = \prod_i \frac{\exp(y_i \cdot x_i^T w)}{1 + \exp(y_i \cdot x_i^T w)}$$

$$P(\text{parameters}|\text{data}) \propto P(\text{data}|\text{parameters})P(\text{parameters})$$



EXAMPLE: LOGISTIC REGRESSION

$$P(\text{parameters}|\text{data}) \propto P(\text{data}|\text{parameters})P(\text{parameters})$$

$$P(w|y,X) = \left(\prod_i \frac{\exp(y_i \cdot x_i^T w)}{1+\exp(y_i \cdot x_i^T w)}\right) P(w)$$

$$\log P(w|y,X) = \log P(w) + \sum_i -\log(1+\exp(-y_i \cdot x_i^T w))$$

prior

Example: $\log P(w) = -|w|$

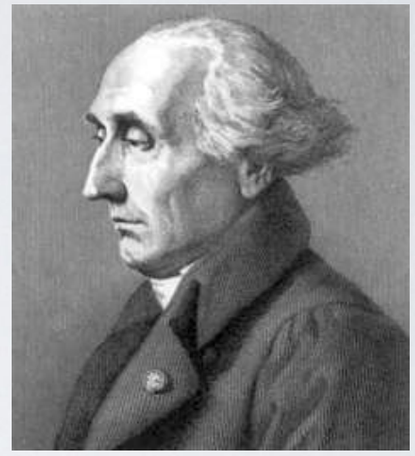
BAYESIANS OFFER NO SOLUTIONS

Bayes

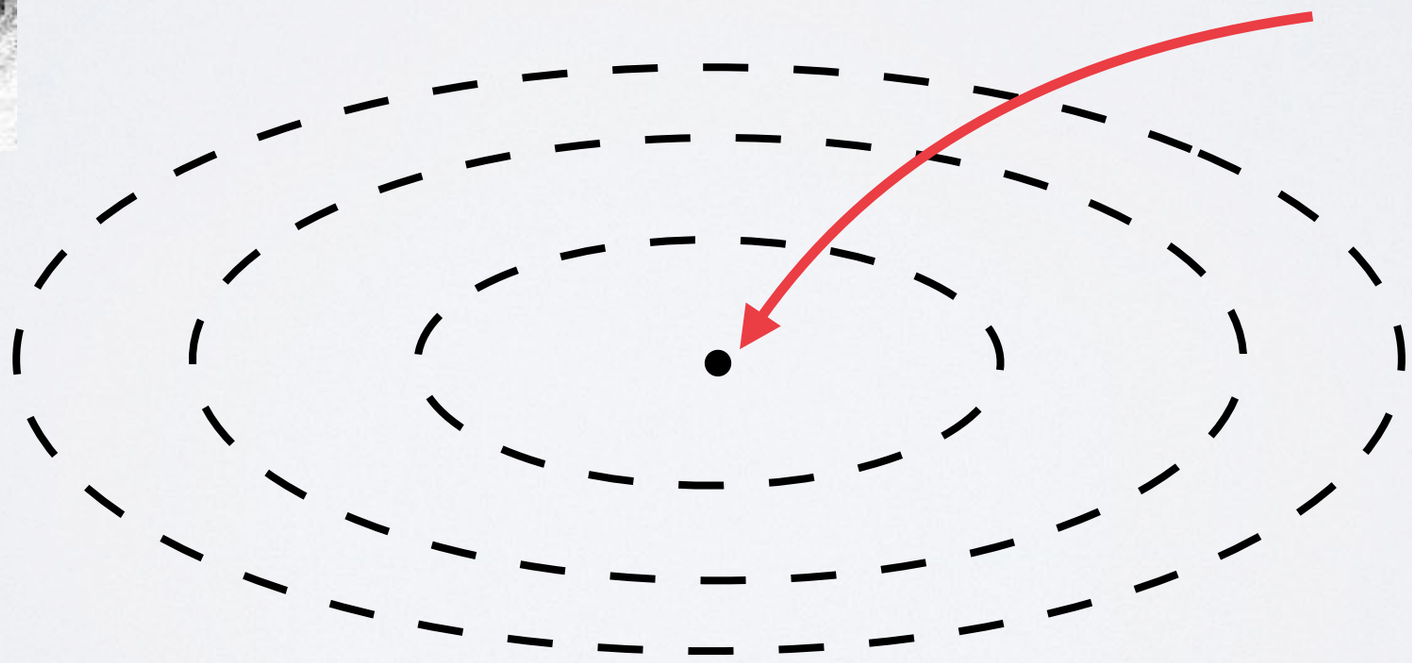
solution space

$$P(w|D,l)$$

Lagrange



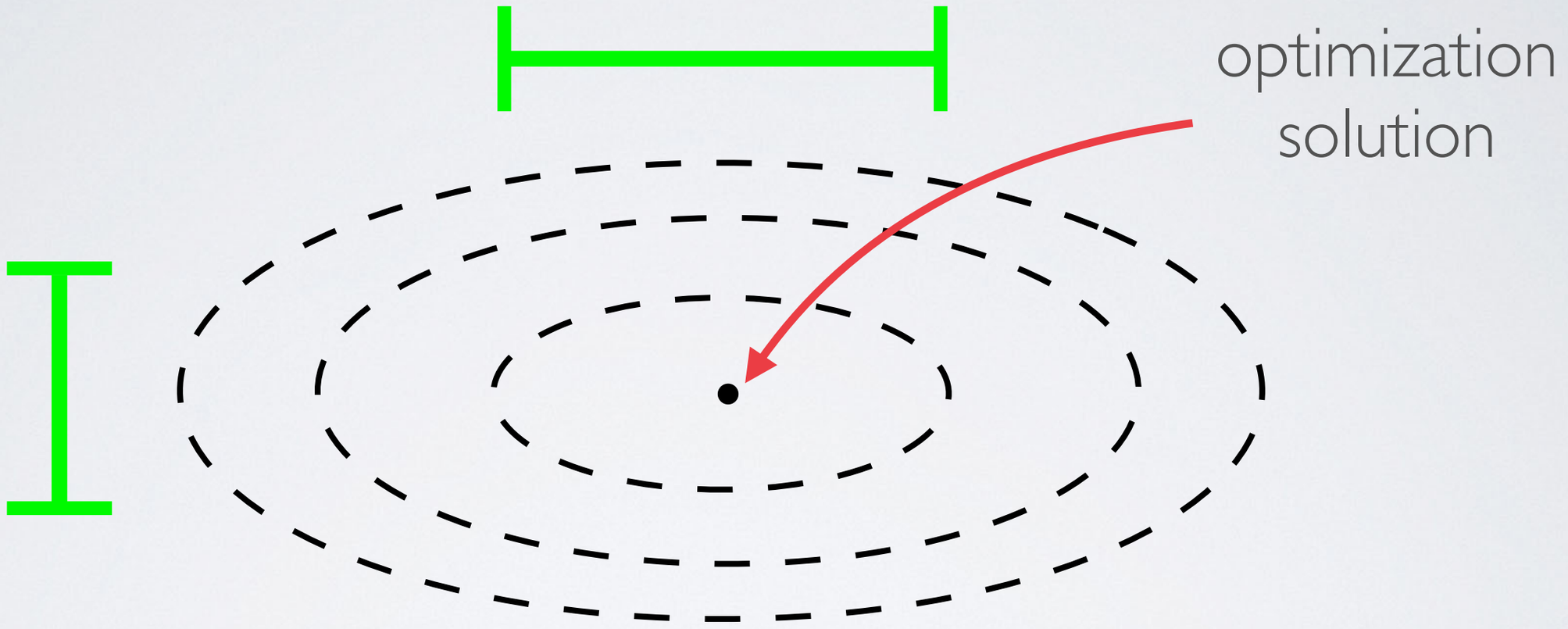
optimization solution



Monte Carlo methods randomly sample this solution space

WHY MONTE CARLO?

You need to know more than just a max/minimizer



$$\mathtt{E}(w) = \int wP(w)\,dw$$

$$\mathtt{Var}(w) = \mathtt{E}[(w-\mu)(w-\mu)^T] = \int ww^T P(w)\,dw - \mu\mu^T$$

EXAMPLE: POKER BOTS

You can't solve your problem by other (faster) methods

$$\text{maximize} \quad \log P(w)$$

we don't know derivative

poker bots

model parameters describe player behavior

2.5M different community cards

10 players = 59K raise/fold/call perms per round

WHY MONTE CARLO?

You're incredibly lazy

$$\text{maximize} \quad \log P(w)$$

I could differentiate this,
I just don't wanna

$$\arg\max \log P(w) \approx \max_k \{P(w^k)\}$$

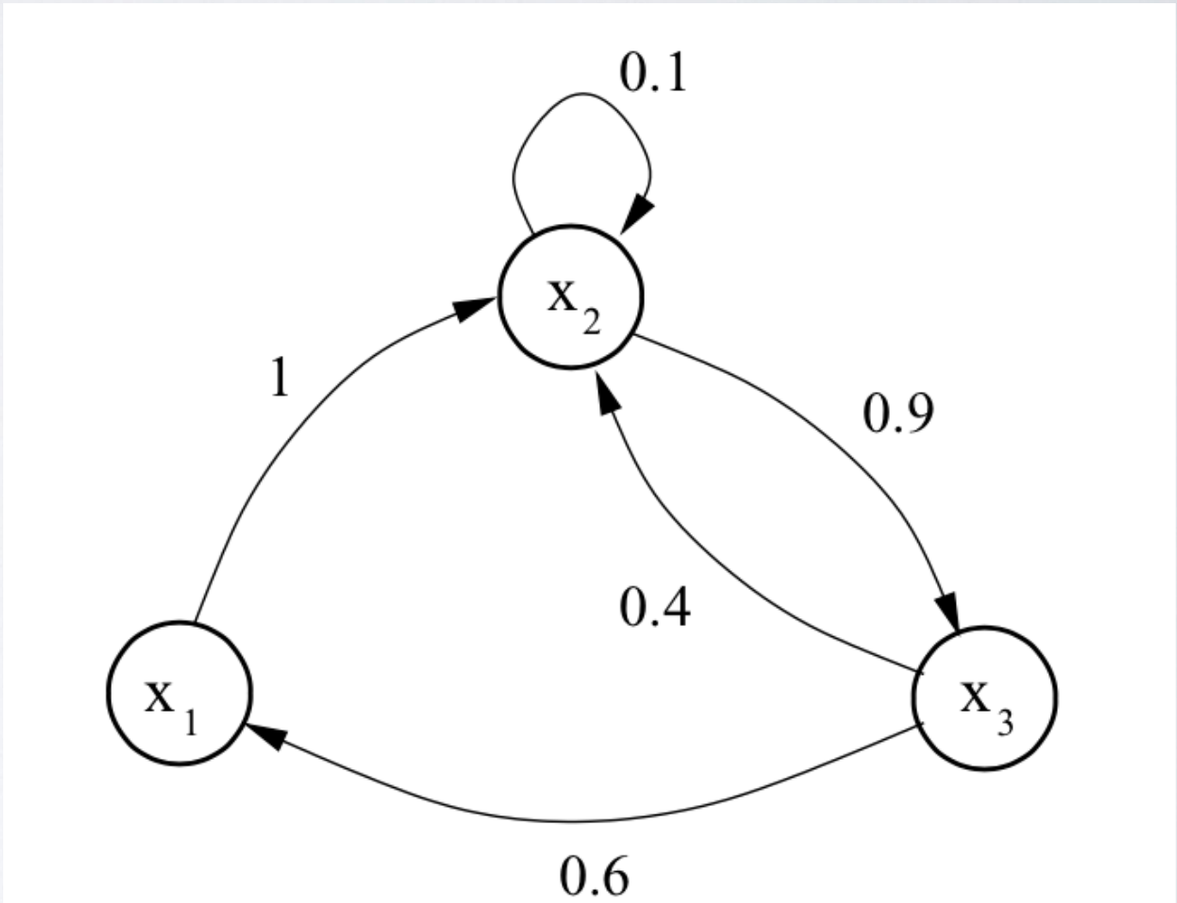
MARKOV CHAINS

What is an MC?

MC has steady state if

Irreducible: can visit any state starting at any state

Aperiodic: does not get trapped in deterministic cycles



METROPOLIS HASTINGS

ingredients

proposal distribution $q(y|x)$

posterior distribution $p(x)$

MH Algorithm

Start with x^0

For $k = 1, 2, 3, \ldots$

Choose candidate y from $q(y|x^k)$

Compute acceptance probability

$$\alpha = \min\left\{1, \frac{p(y)q(x^k|y)}{p(x^k)q(y|x^k)}\right\}$$

Set $x^{k+1} = y$ with probability α

otherwise $x^{k+1} = x^k$

CONVERGENCE

Irreducible: support of q must contain support of p

Aperiodic: there is positive probability of jumping anywhere

Theorem

Suppose the support of q contains the support of p. Then the MH sampler has a stationary distribution, and the distribution is equal to p.

METROPOLIS ALGORITHM

ingredients

proposal distribution $q(y|x)$

posterior distribution $p(x)$

Assume proposal is symmetric

$$q(x^k|y) = q(y|x^k)$$

Metropolis Hastings

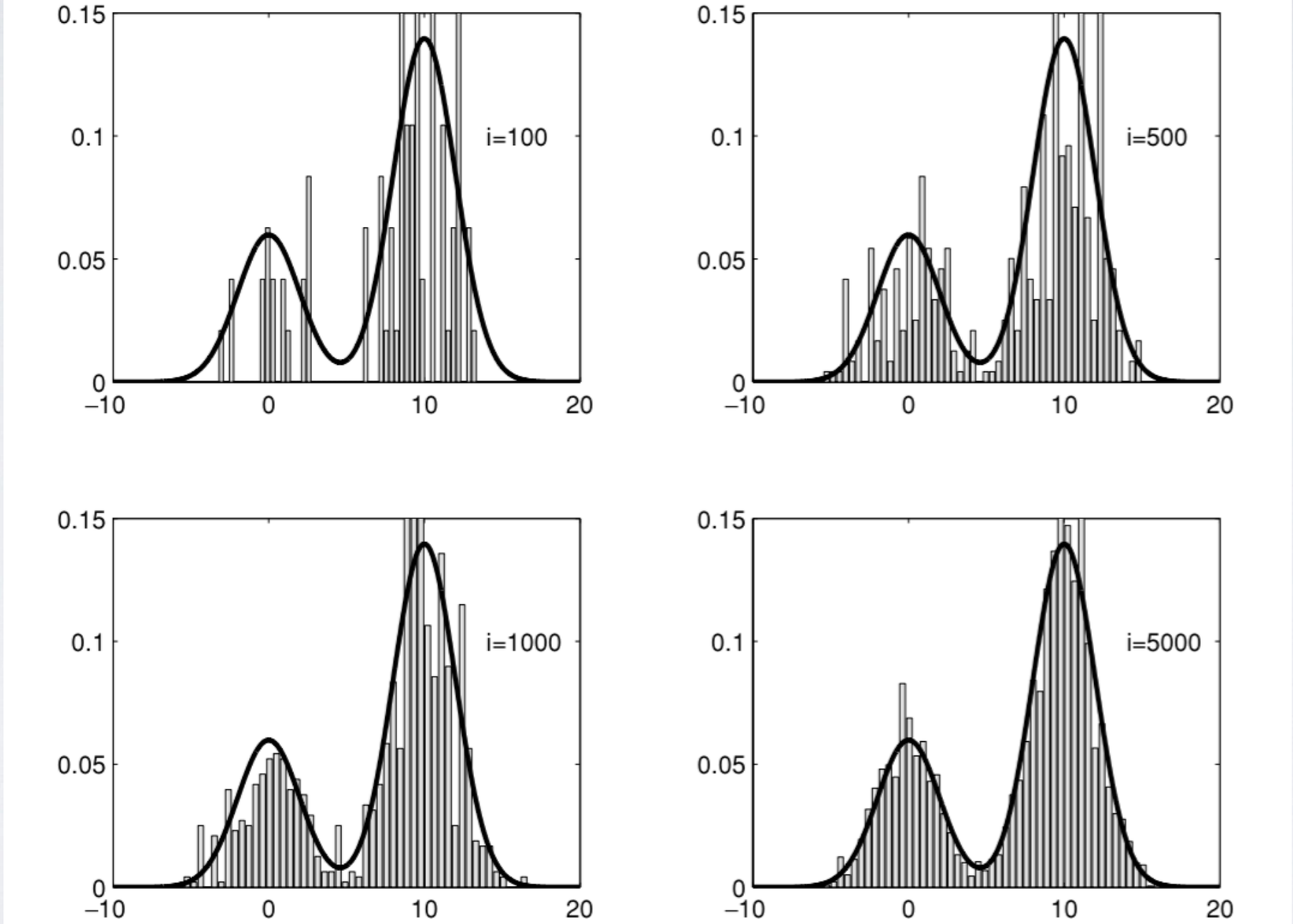
$$\alpha = \min\left\{1, \frac{p(y)q(x^k|y)}{p(x^k)q(y|x^k)}\right\}$$

Metropolis

$$\alpha = \min\left\{1, \frac{p(y)}{p(x^k)}\right\}$$

EXAMPLE: GMM

histogram of Metropolis iterates



Andrieu, Freitas, Doucet, Jordan '03

PROPERTIES OF MH

Pro's

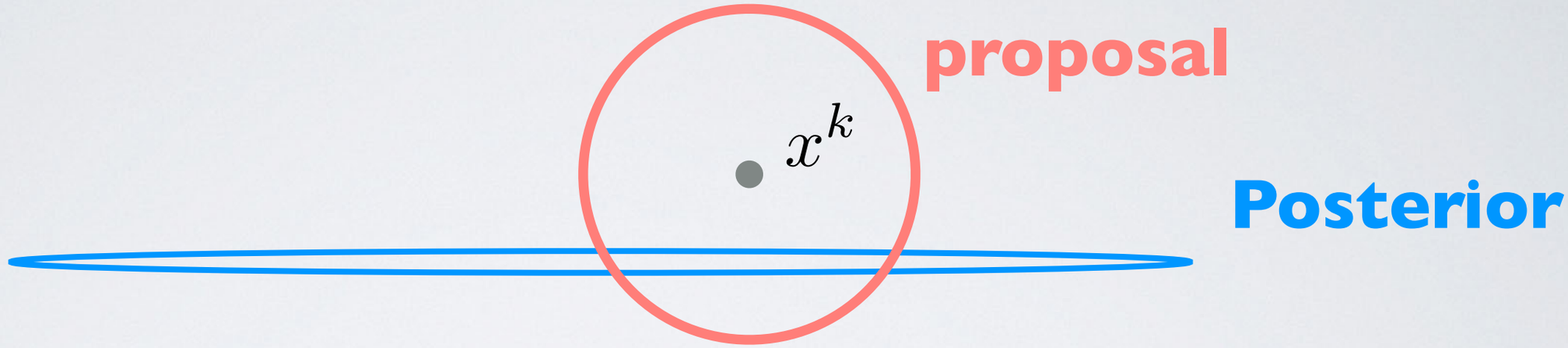
We don't need a normalization constant for posterior
We can run many chains in parallel (cluster/GPU)
We don't need any derivatives

$$P(\text{parameters}|\text{data}) = \frac{P(\text{data}|\text{parameters})P(\text{parameters})}{P(\text{data})}$$

$$P(\text{parameters}|\text{data}) \propto P(\text{data}|\text{parameters})P(\text{parameters})$$

PROPERTIES OF MH

This is bad…



Con's

"Mixing time" depends on proposal distribution

- too wide = constant rejections = slow mixing
- too narrow = short movements=slow mixing

Samples only meaningful at stationary distribution

- "Burn in" samples must be discarded
- Many samples needed because of correlations

SIMULATED ANNEALING

$$\text{maximize} \quad p(x)$$

Easy choice: run MCMC, then

$$\max_k p(x^k)$$

Better choice: sample the distribution

$$p^{\frac{1}{T_k}}(x^k)$$

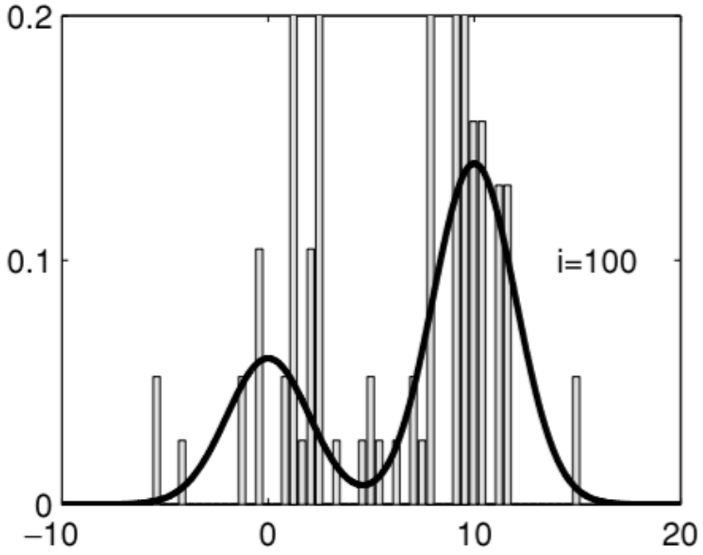
"temperature"

Why is this better?

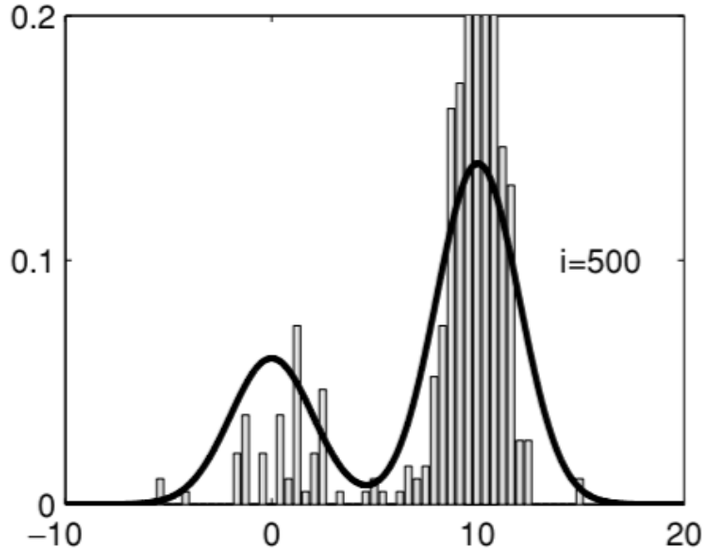
Where does the name come from?

COOLING SCHEDULE

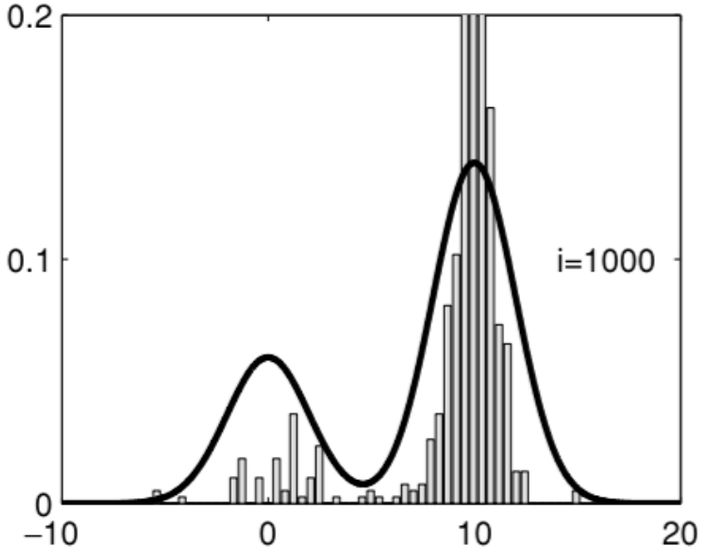
hot



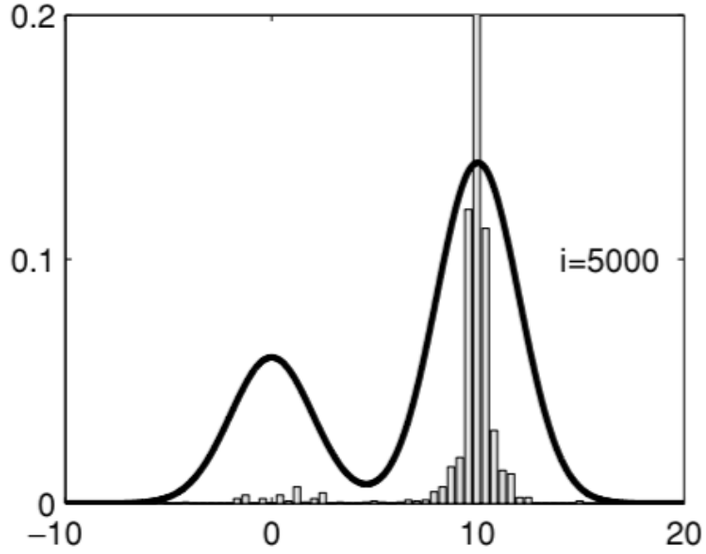
warm



cool



cold



Andrieu, Freitas, Doucet, Jordan '03

CONVERGENCE

SA solves non-convex problem, even NP-complete problems, as time goes to infinity.

Theorem

Suppose the MCMC mixes fast enough that epsilon-dense sampling occurs in finite time starting at every temperature. For an annealing schedule with temperature

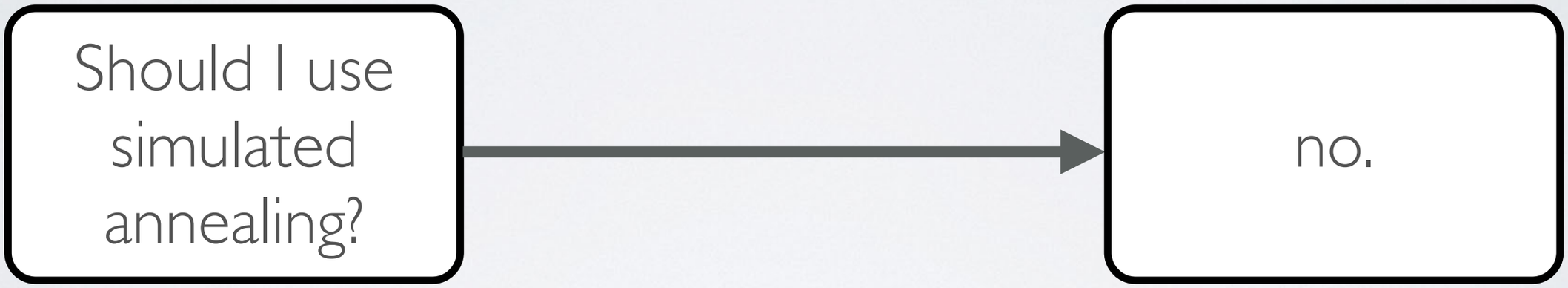
$$T_k = \frac{1}{C \log(k + T_0)}$$

simulated annealing converges to a global optima with probability 1.

Granville, "Simulated annealing: A proof of convergence"

WHEN TO USE SIMULATED ANNEALING

flow chart



No practical way to choose temperature schedule
Too fast = stuck in local minimum (risky)
Too slow = no different from MCMC
Act of desperation!

GIBBS SAMPLER

Want to sample

$$P(x_1, x_2, x_3)$$

on stage k, pick some coordinate j

$$q(y|x^k) = \begin{cases} p(y_j|x^k_{j^c}), \text{ if } y_{j^c} = x^k_{j^c} \\ 0, \text{ otherwise} \end{cases}$$

$$p(y) = p(y_j \text{ and } y_{j^c})$$

$$\alpha = \frac{p(y)q(x^k|y)}{p(x^k)q(y|x^k)} = \frac{p(y)p(x^k_j|x^k_{j^c})}{p(x^k)p(y_j|y_{j^c})}$$

$$= \frac{p(y_{j^c})}{p(x^k_{j^c})} = 1$$

$$P(B) = \frac{P(A \text{ and } B)}{P(A|B)}$$

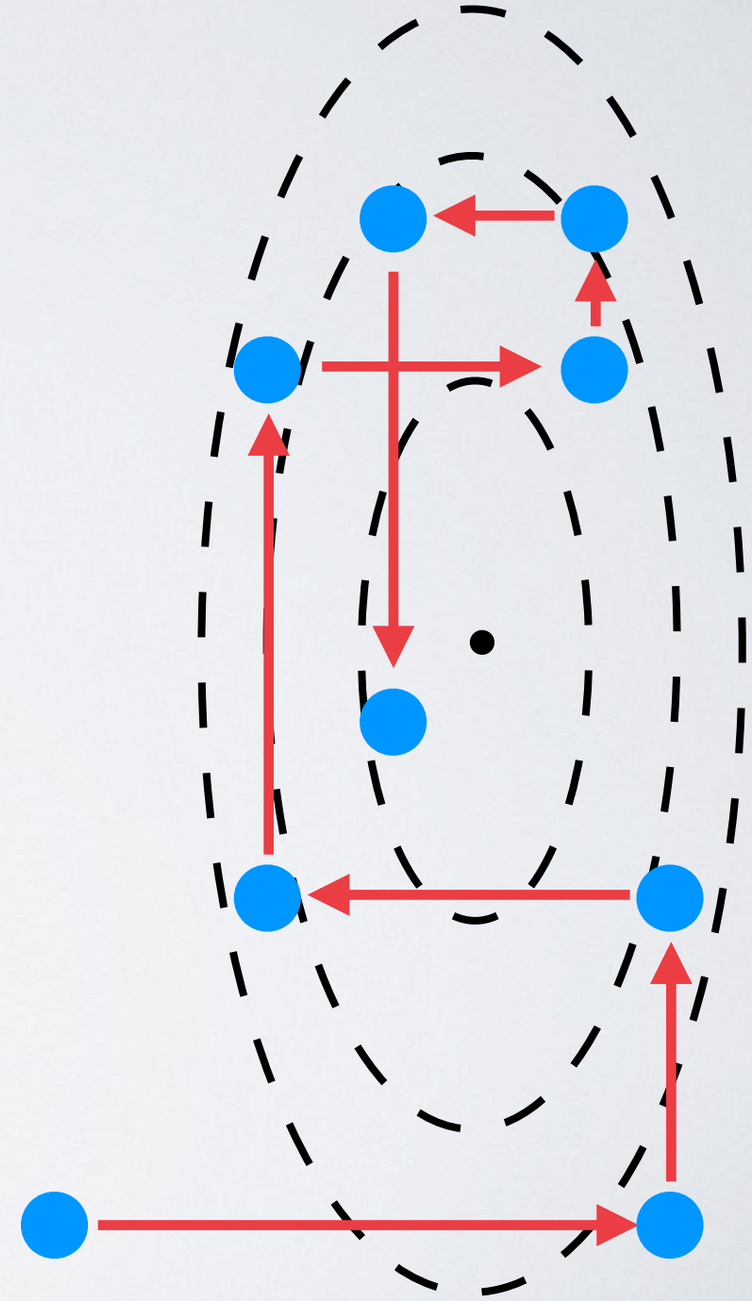
GIBBS SAMPLER

Want to sample

$$P(x_1, x_2, x_3)$$

iterates

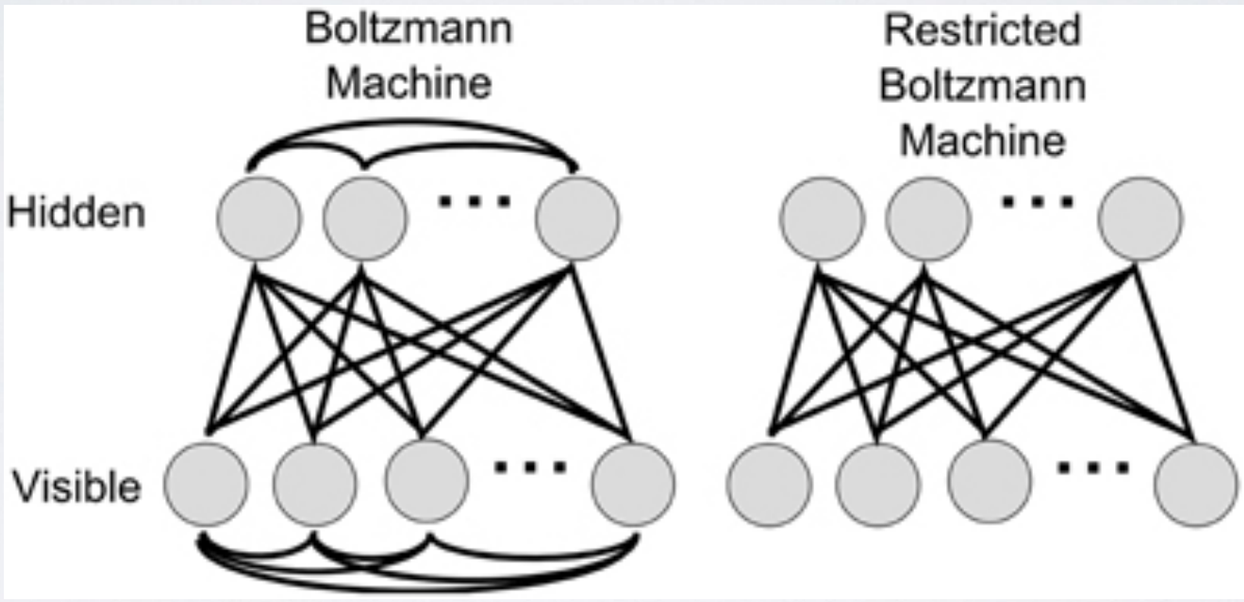
$$x^2 \sim P(x_1 | x_2^1, x_3^1, x_4^1, \ldots, x_n^1)$$
$$x^3 \sim P(x_2 | x_1^2, x_3^2, x_4^2, \ldots, x_n^2)$$
$$x^4 \sim P(x_3 | x_1^3, x_2^3, x_4^3, \ldots, x_n^3)$$
$$\ldots$$



APPLICATION: SAMPLING GRAPHICAL MODELS

Restricted Boltzmann Machine (RBM)

binary random variables 0/1



weights

$$E(v,h) = -a^T v - b^T h - v^T W h$$

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

"partition function"

APPLICATION: SAMPLING GRAPHICAL MODELS

Restricted Boltzmann Machine (RBM)

$$E(v,h) = -a^T v - b^T h - v^T W h \qquad P(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

$$P(v_i = 1|h) = \frac{P(v_i = 1|h)}{P(v_i = 0|h) + P(v_i = 1|h)}$$ remove normalization

$$= \frac{\exp(-a_i + \sum_j w_{ij} h_j + C)}{\exp(C) + \exp(-a_i + \sum_j w_{ij} h_j + C)}$$ aggregate constant

$$= \frac{\exp(-a_i + \sum_j w_{ij} h_j)}{1 + \exp(-a_i + \sum_j w_{ij} h_j)}$$ cancel constants

$$= \sigma(-a_i + \sum_i w_{ij} h_j)$$ sigmoid function

BLOCK GIBBS FOR RBM

stage 1

freeze hidden

randomly sample visible

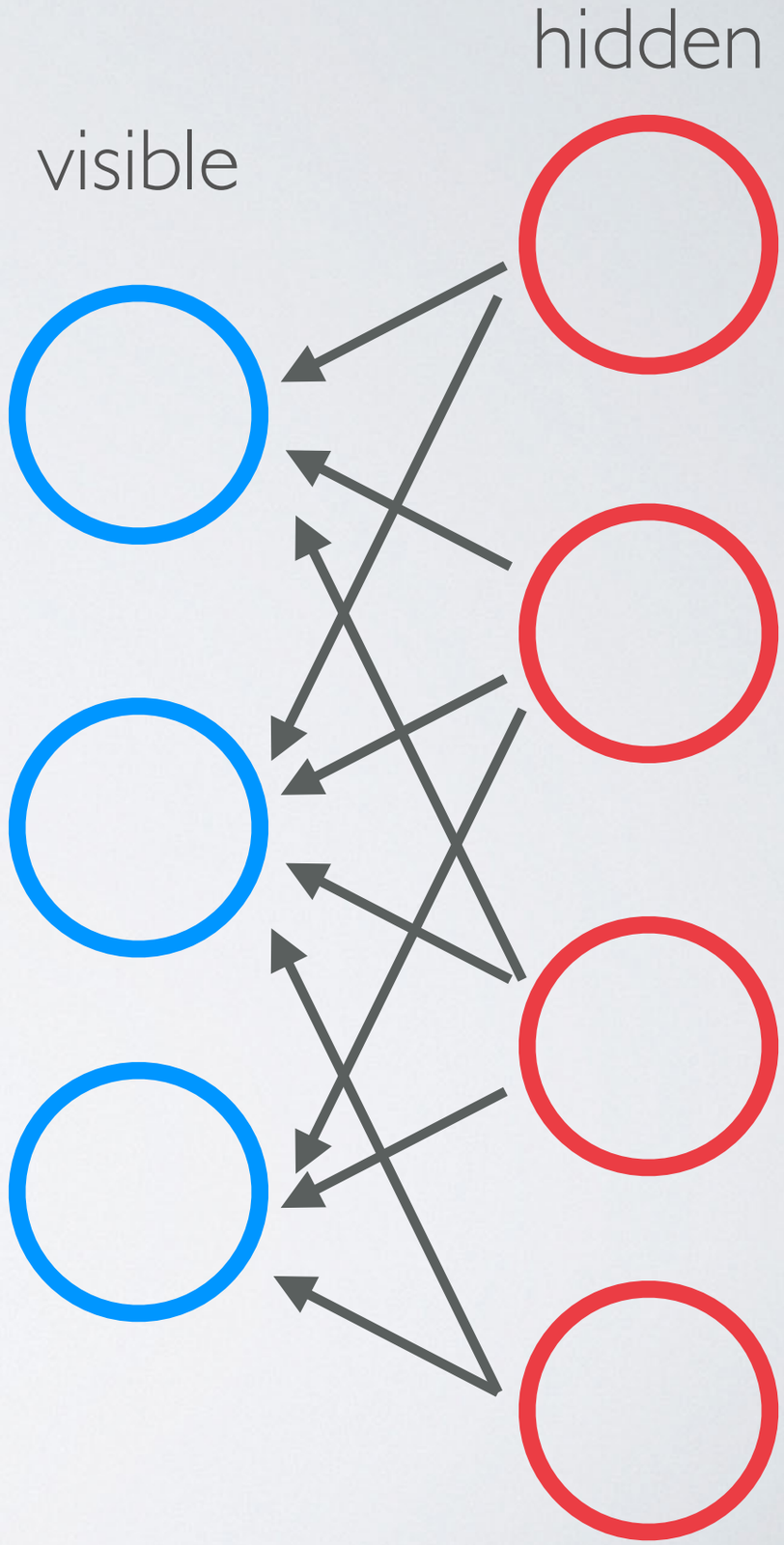
$$P(v_i = 1|h) = \sigma(-a_i + \sum_j w_{ij} h_j)$$

stage 2

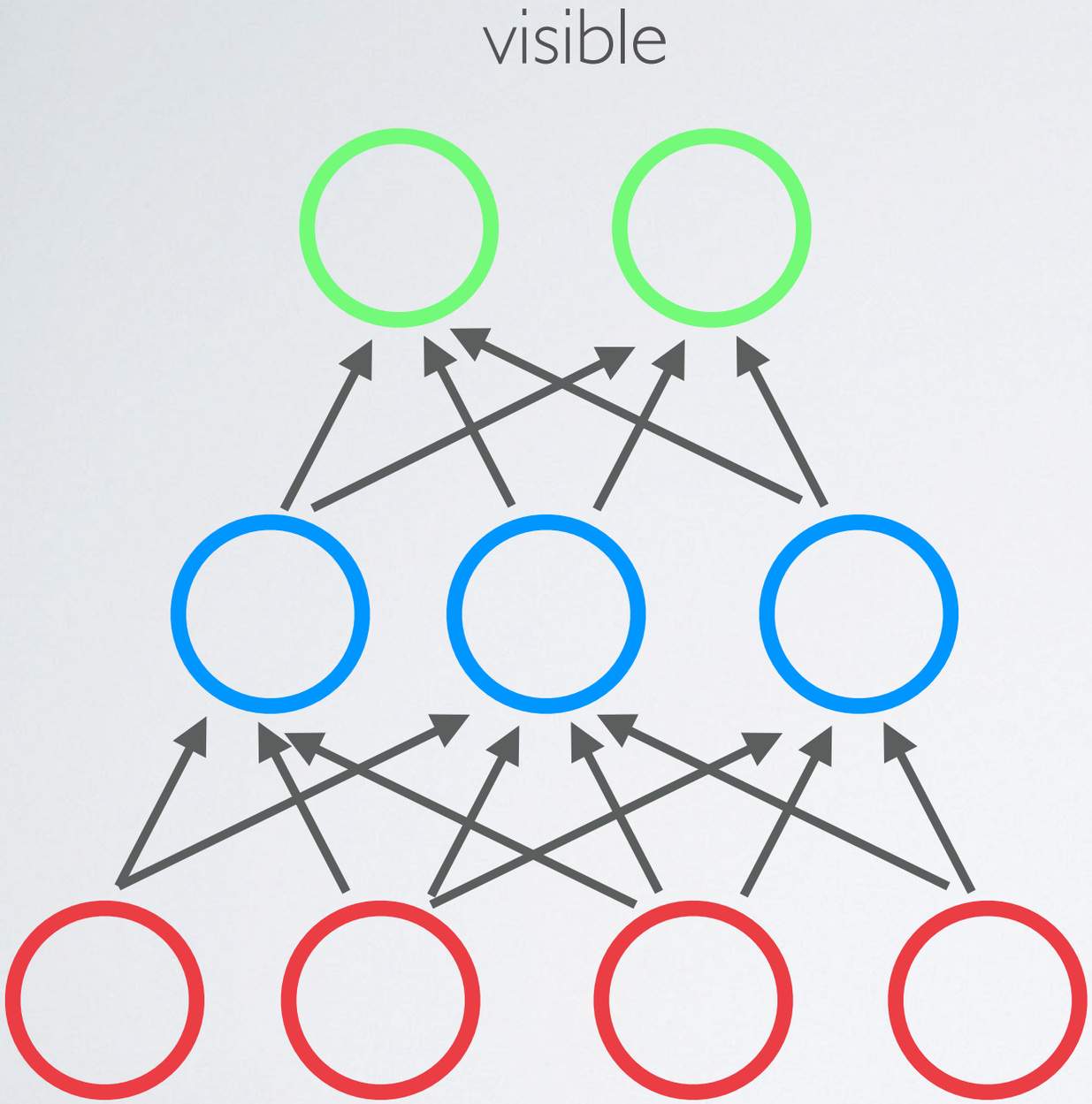
freeze visible

randomly sample hidden

$$P(h_j = 1|v) = \sigma(-b_j + \sum_i w_{ij} v_i)$$



DEEP BELIEF NETS



DBN = layered RBM

Each layer **only** depends on layer beneath it (feed forward)

Probability for each hidden node is sigmoid function

EXAMPLE: MNIST

Gan, Henao, Carlson, Carin '15

Train 3-layer DBN with 200 hidden units

trained on 60K MNIST digits

pre-training: layer-by-layer training

training: train all weights simultaneously

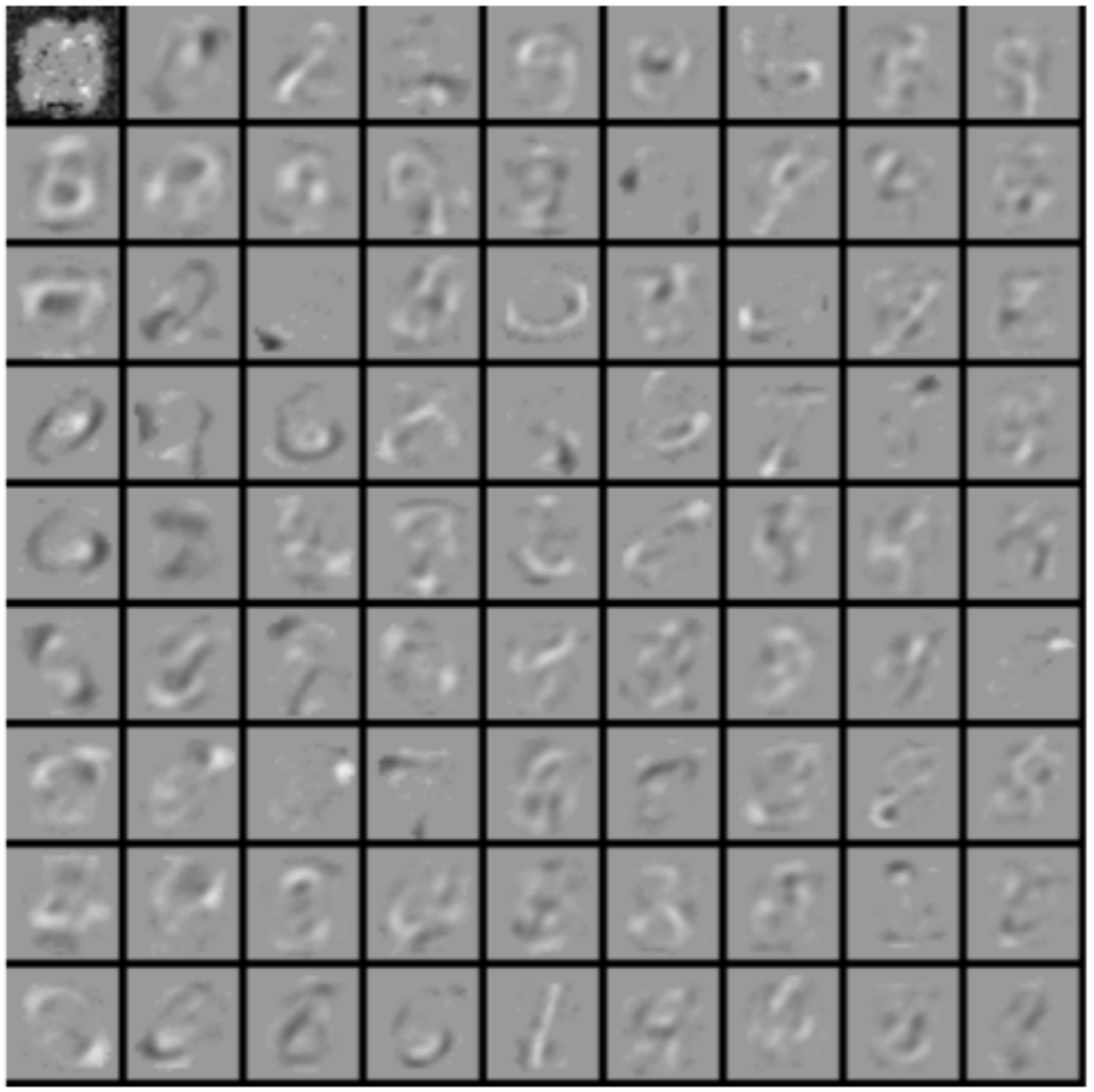
training done using Gibbs sampler

Gibbs sampler used to explore final solution

EXAMPLE: MNIST

training data

Learned features



observations sampled from deep belief network using Gibbs sampler

COMPARISON

	rate	**when to use**
Gradient/ Splitting	e^{-ck}	you value reliability and precision (moderate speed, high accuracy)
SGD	$1/k$	you value speed over accuracy (high speed, moderate accuracy)
MCMC	$1/\sqrt{k}$	you value simplicity (no gradient) or need statistical inference (slow and inaccurate)

DO MATLAB EXERCISE MCMC