Demystifying Depth:
Learning dynamics in deep linear neural networks

Andrew M. Saxe

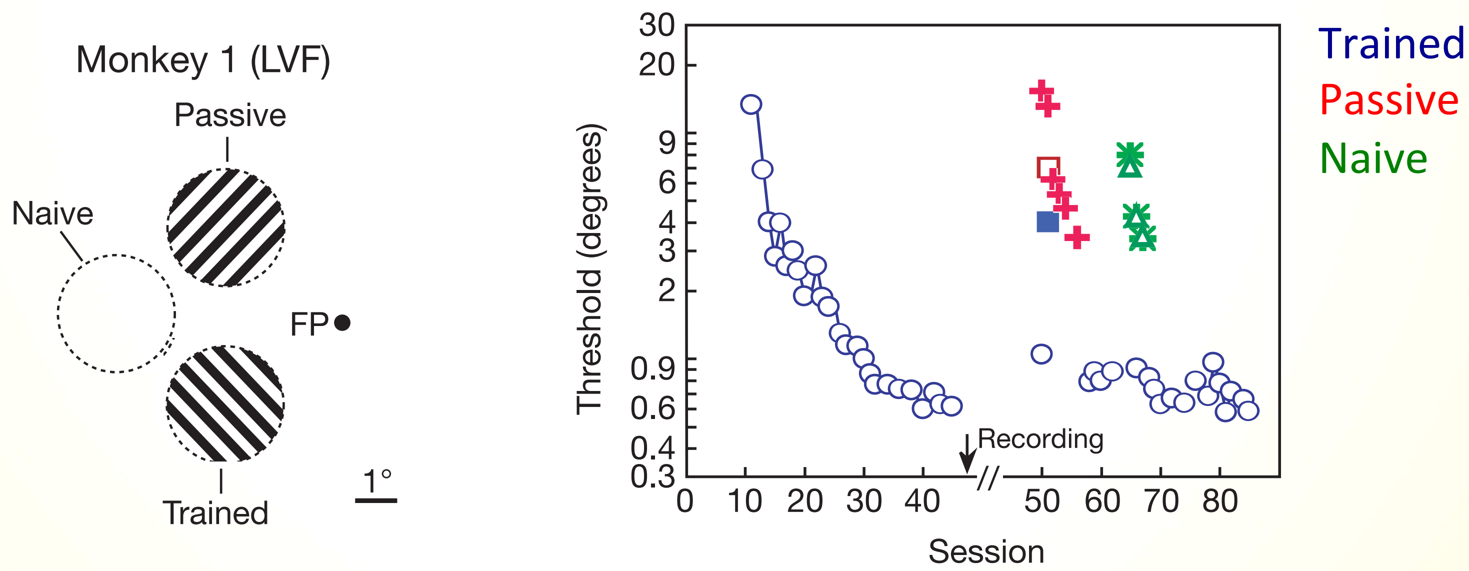
Stanford University

Linking learning and plasticity

- Humans and other organisms are incredibly sophisticated learners
- Across a variety of tasks, we get much better with practice
- How do changes in synaptic strength across the brain enable this learning?

Perceptual Learning

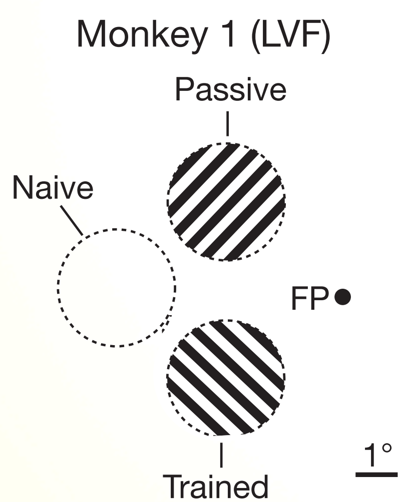
- Practicing orientation discrimination improves behavioral performance

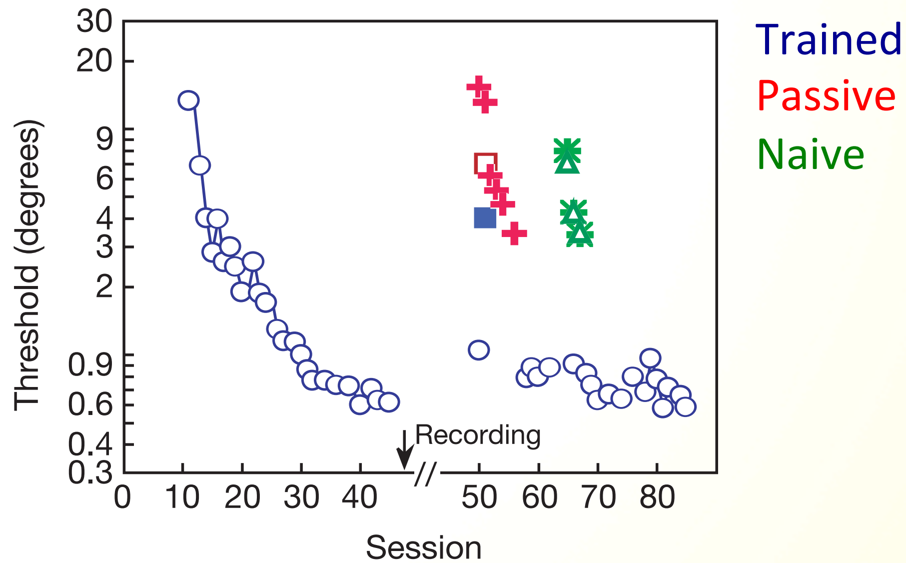


Schoups et al., 2001

Perceptual Learning

- Practicing orientation discrimination improves behavioral performance

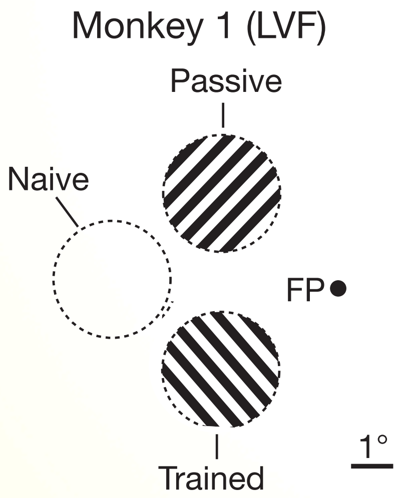


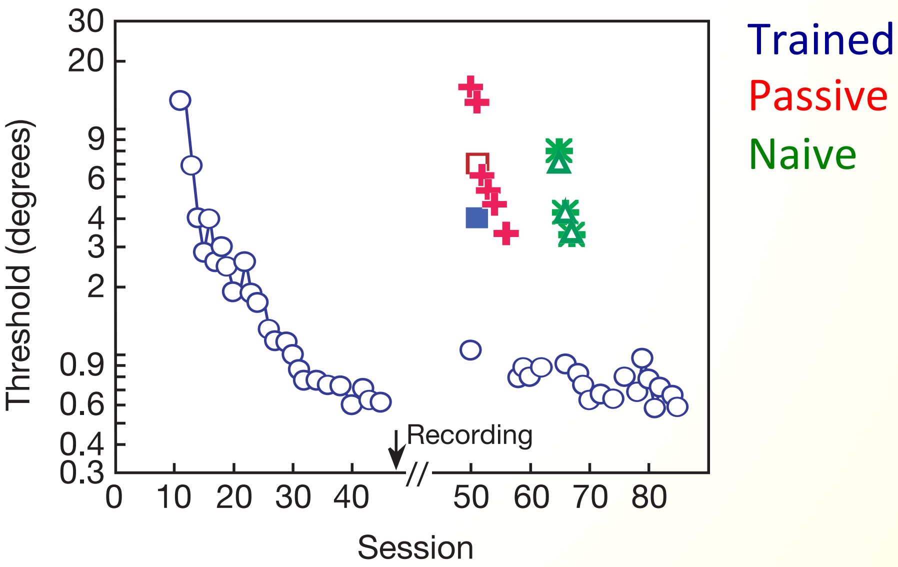


Schoups et al., 2001

Perceptual Learning

- Practicing orientation discrimination improves behavioral performance



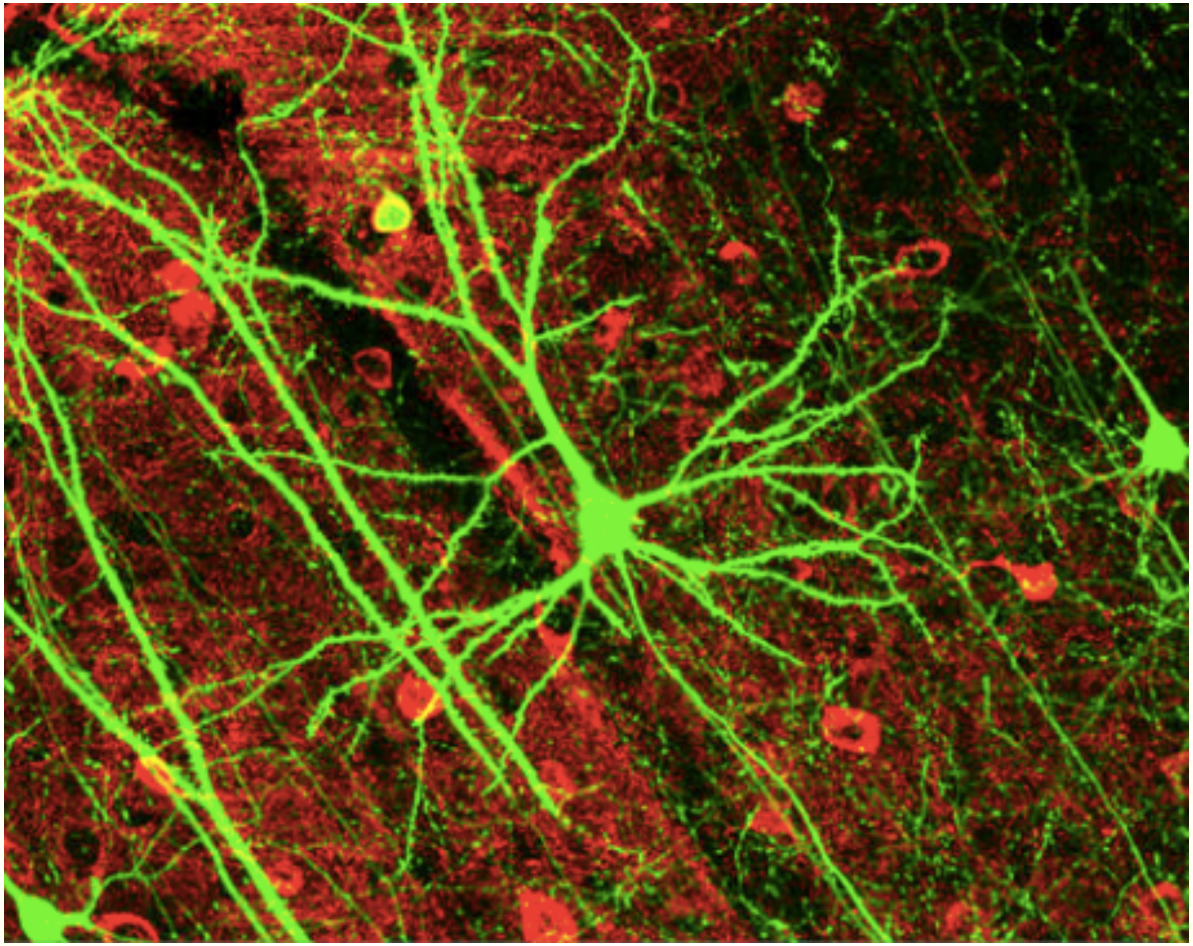


Schoups et al., 2001

The brain

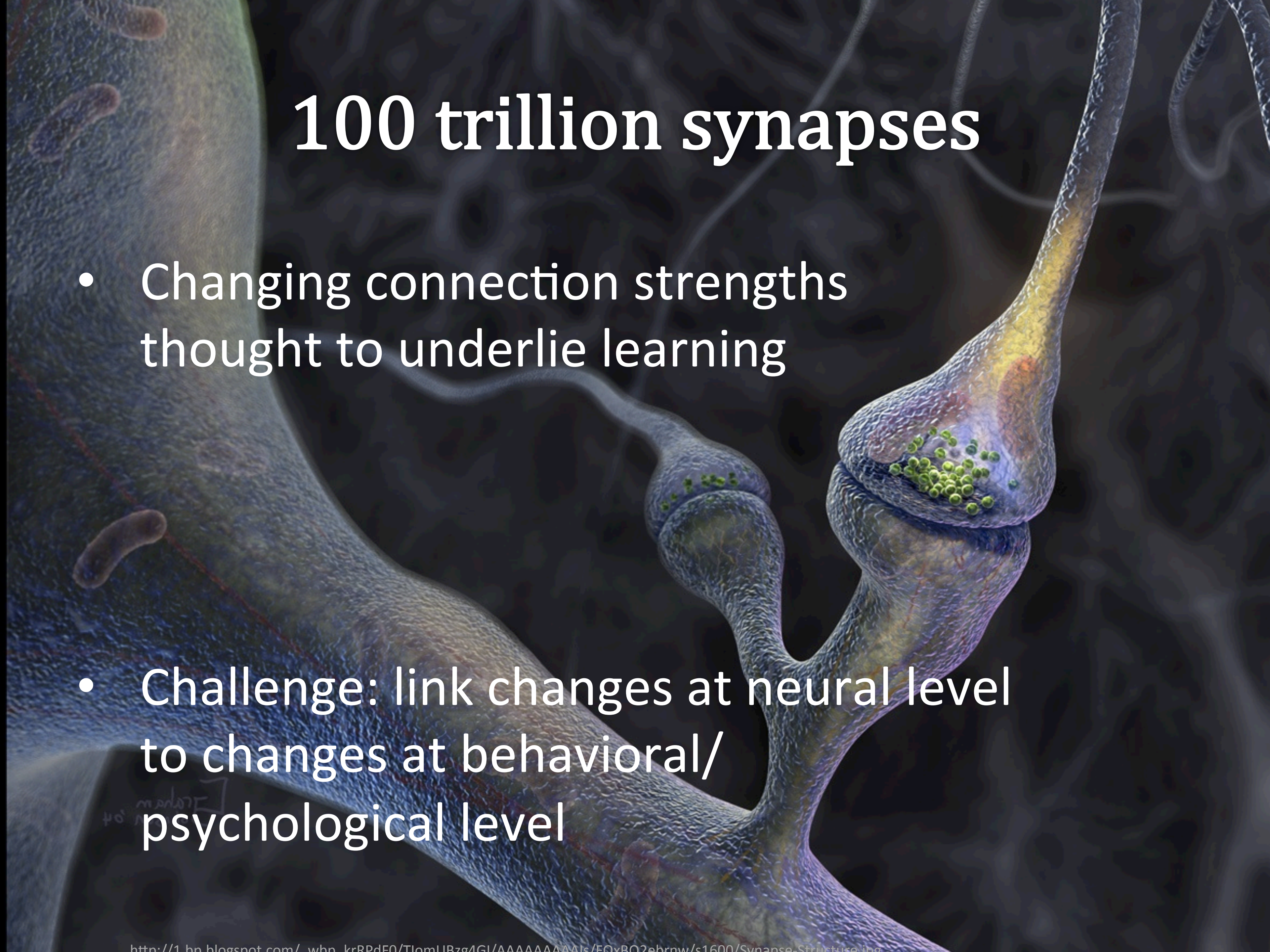
http://phenomena.nationalgeographic.com/files/2013/04/brain-990x622.png

50 billion neurons



http://cdn.medicalxpress.com/newman/gfx/news/2009/neuron.png

100 trillion synapses



- Changing connection strengths thought to underlie learning

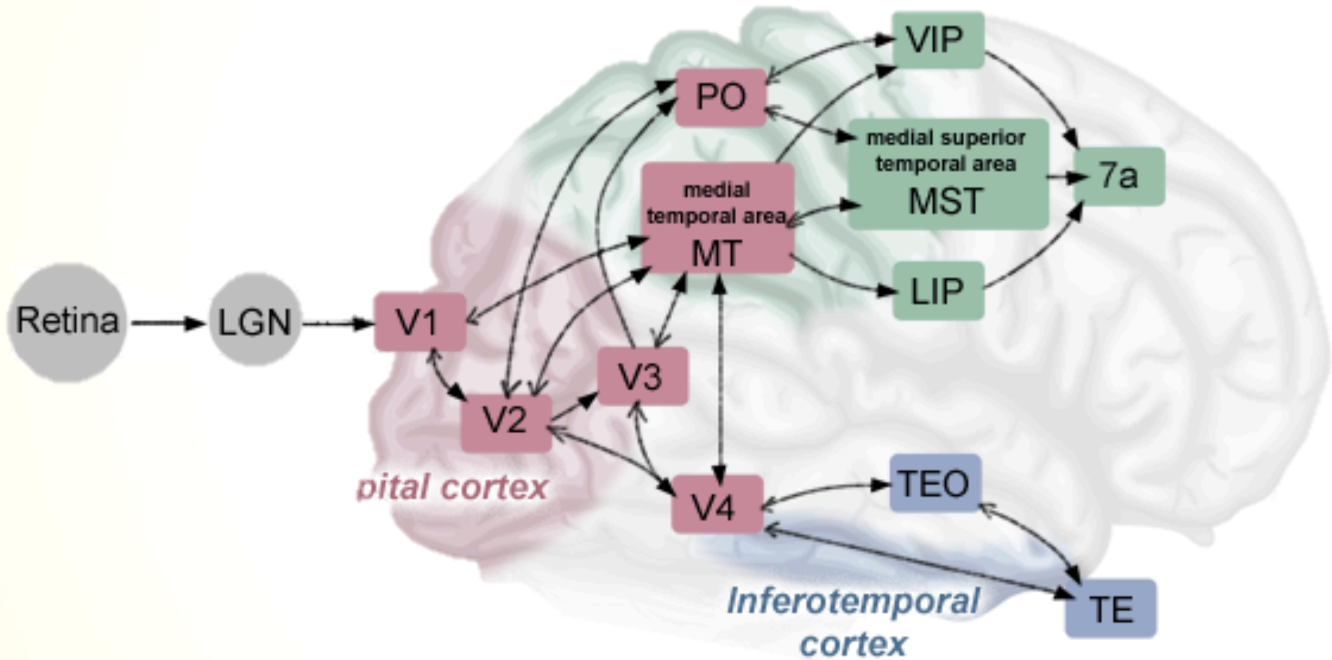
- Challenge: link changes at neural level to changes at behavioral/ psychological level

http://1.bp.blogspot.com/_wbp_krRPdF0/TIomUBzg4GI/AAAAAAAAAIs/FQxBQ2ebrnw/s1600/Synapse-Structure.jpg

Depth

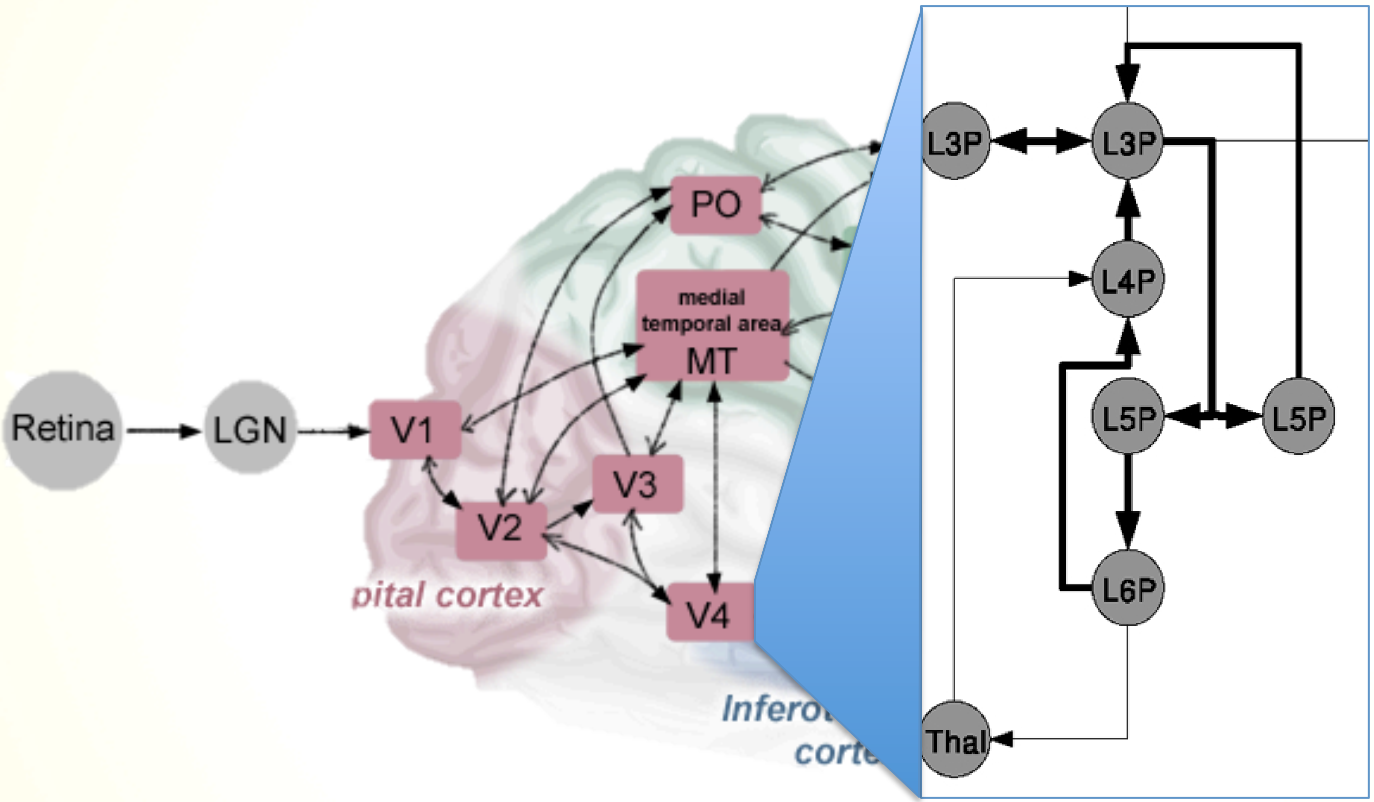
- The brain has a layered structure
  - Anatomically
  - Physiologically
- I will argue this strongly impacts learning dynamics in the brain

Depth: Layered anatomy



http://www.eyebrainpedia.com/En/Localisation/images/EncephaleAiresVisuellesPosterieures.png

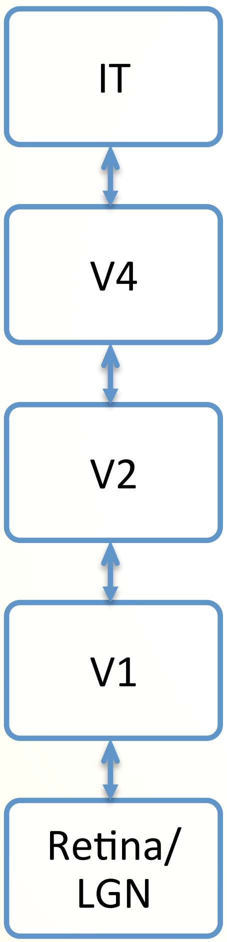
Depth: Layered anatomy

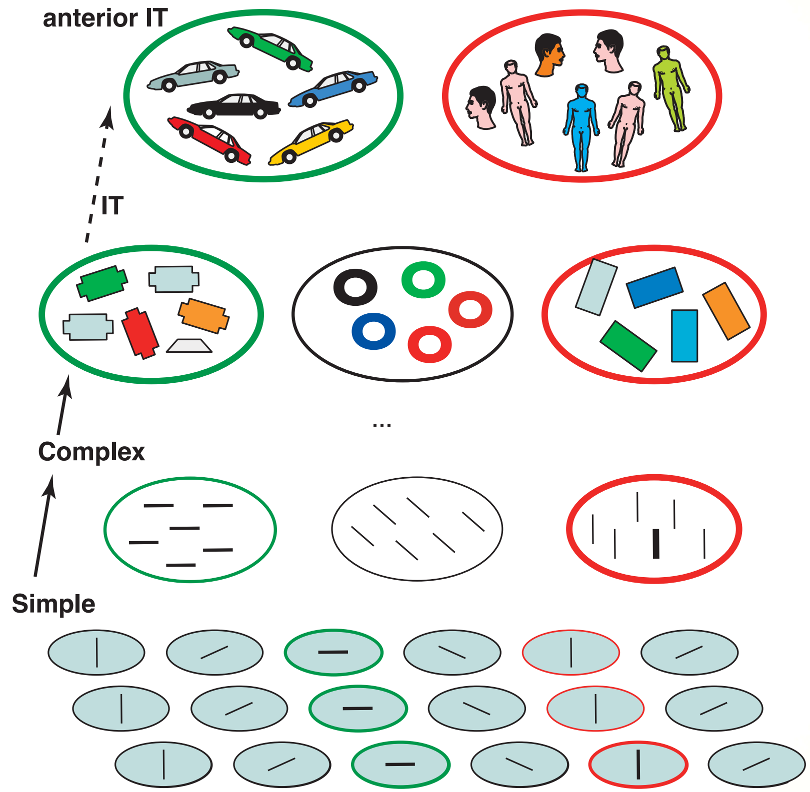


Douglas & Martin, 2004

http://www.eyebrainpedia.com/En/Localisation/images/EncephaleAiresVisuellesPosterieures.png

Depth: Layered physiology





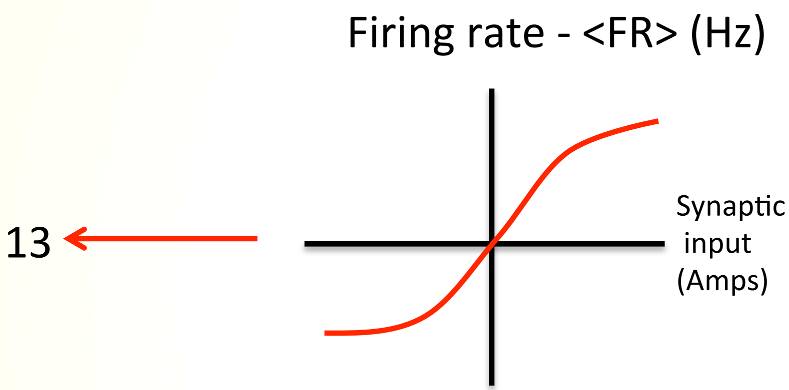
Object detectors

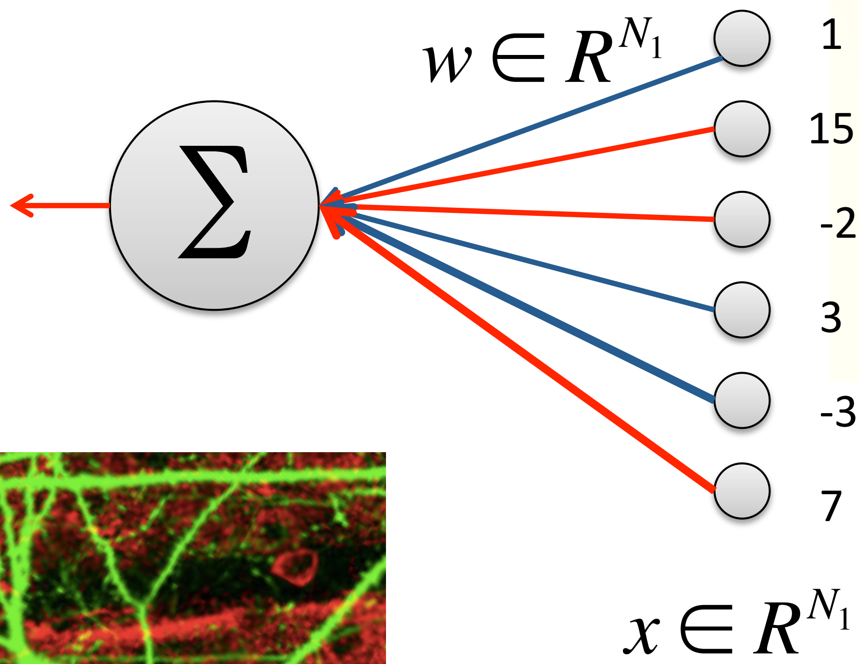
Shape detectors

Edge detectors

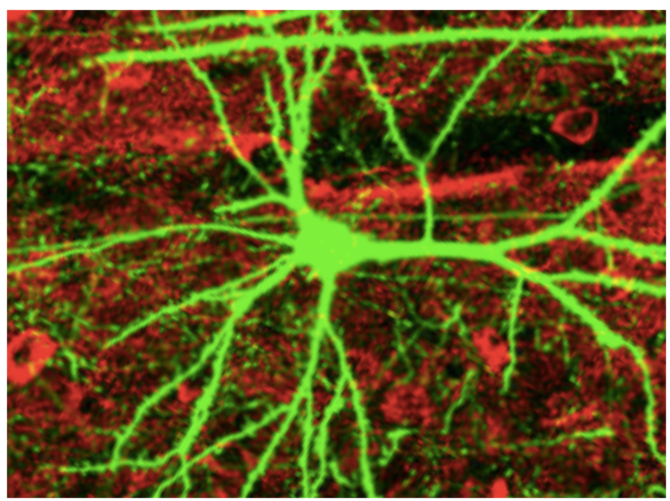
Ahissar & Hochstein, 2004

Artificial neurons

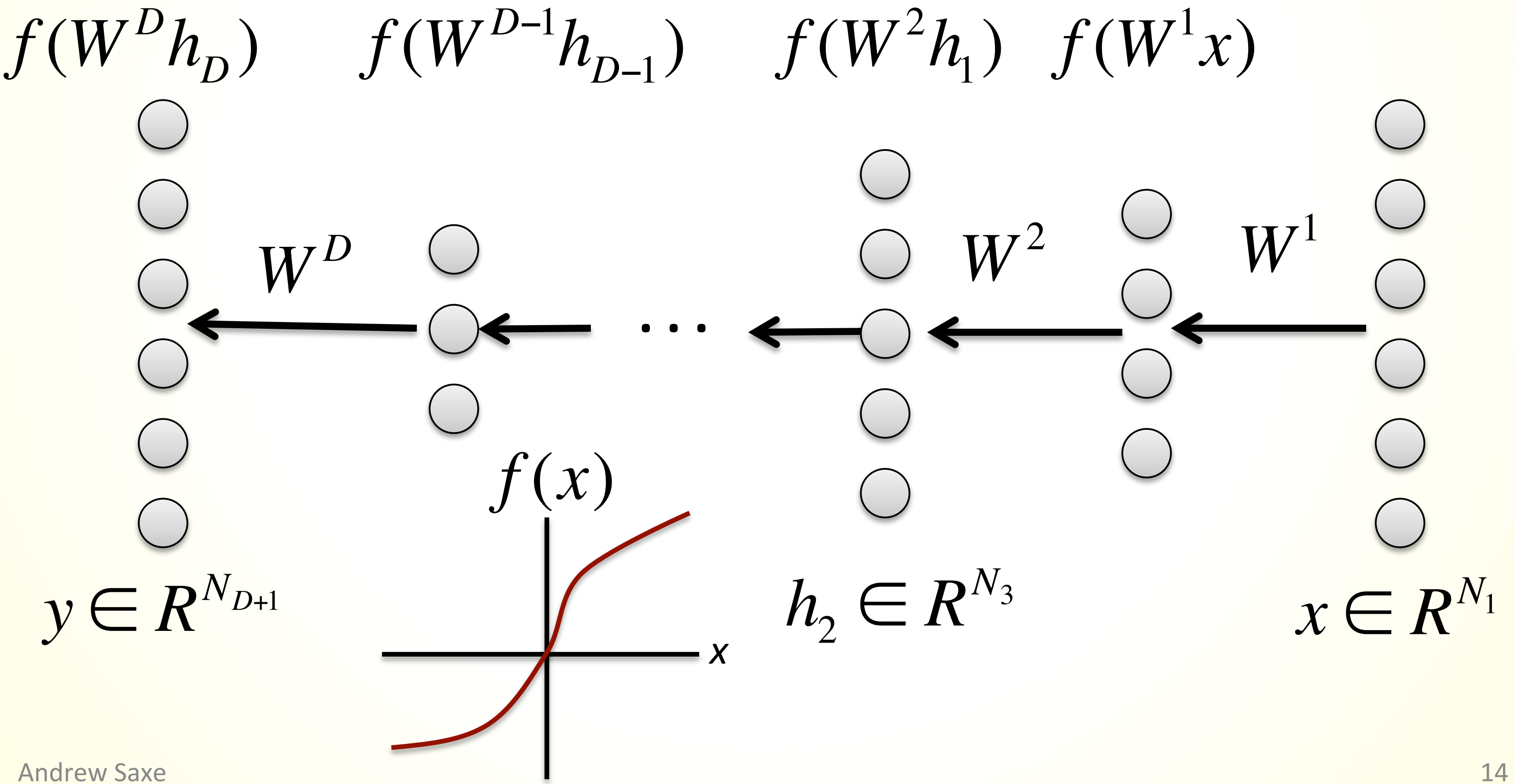




$$y = f(w^T x)$$

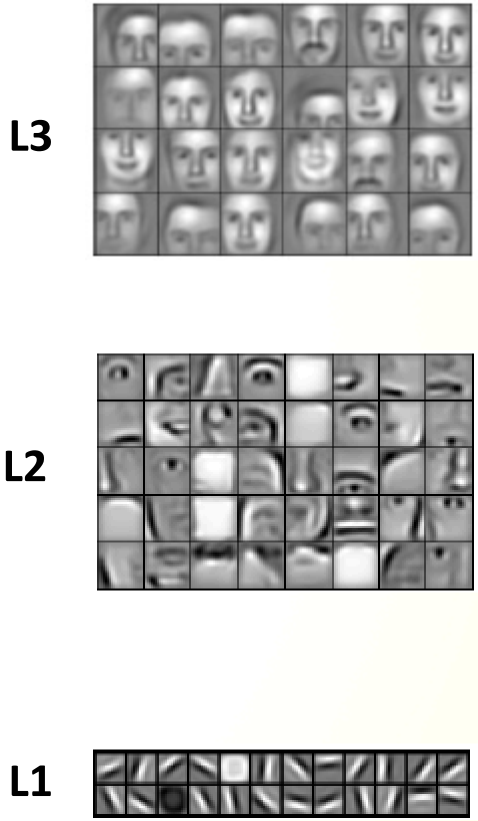


Deep neural networks



Deep learning in AI

- Many-layered artificial neural networks
- Currently state-of-the-art on many real world datasets
  - Object recognition
  - Speech recognition
  - Text processing
- Black boxes
- Nonlinearities resistant to theory



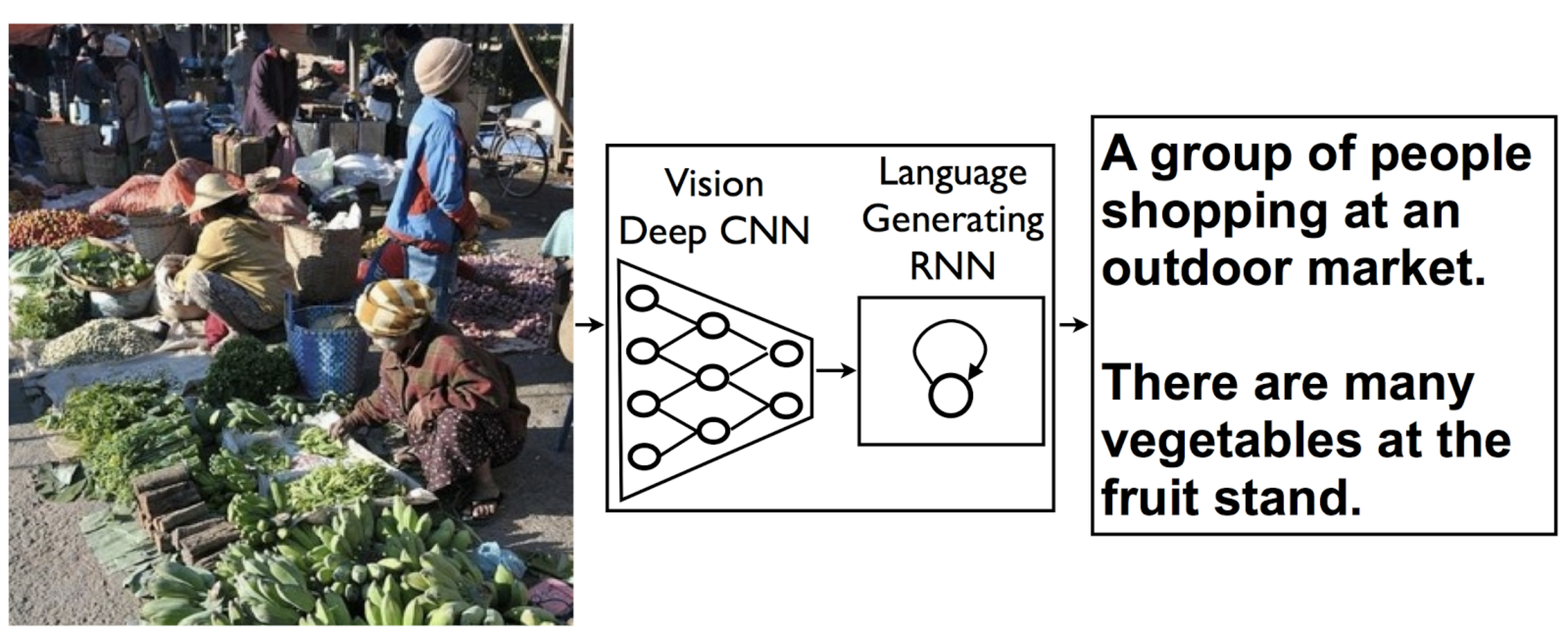
Lee et al., 2009

Object Recognition

- Decisively state of the art in visual object recognition from images

ImageNet large scale visual recognition challenge, Russakovsky et al., 2014

Image captioning



Google Brain, Vinyals et al., 2014

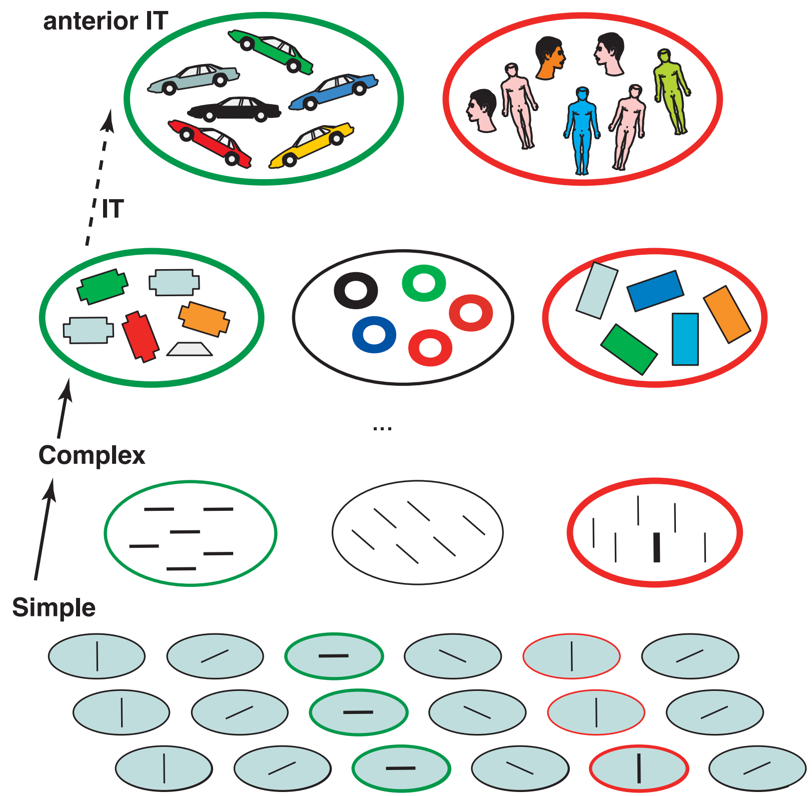
Why depth?

- Compactly represent complex input-output functions
- Divide and conquer: slowly build up complexity by composing simple elements
- Transform inputs/outputs into suitable internal representation
- High performance on benchmark tasks

Depth complicates learning

- Must choose distribution of changes across layers
- Introduces
  - Coupling
  - Symmetries
- Learning often much slower

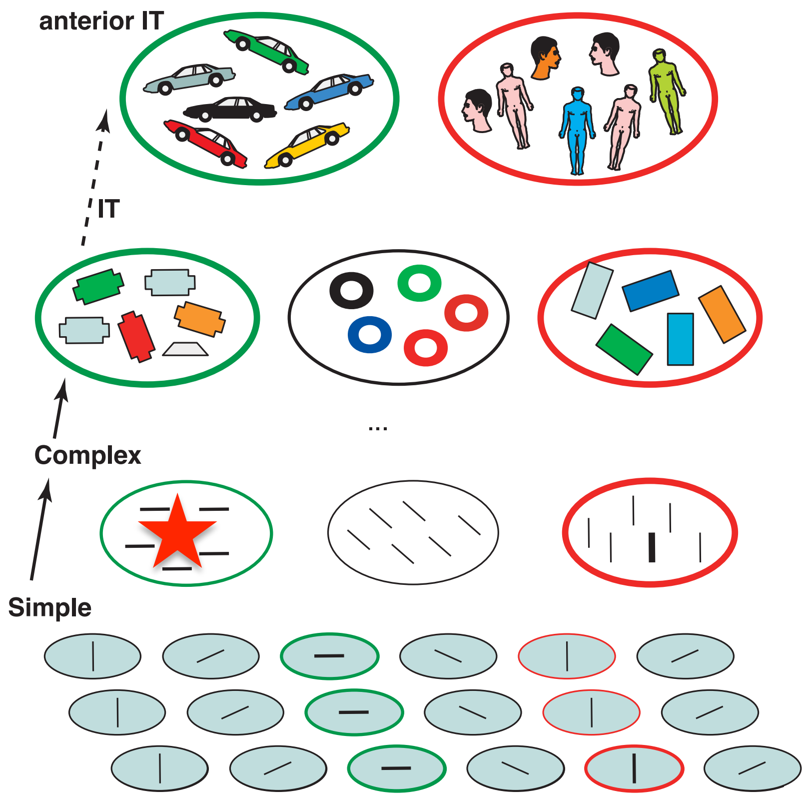
The coupling problem



Ahissar & Hochstein, 2004

The coupling problem

Change V1

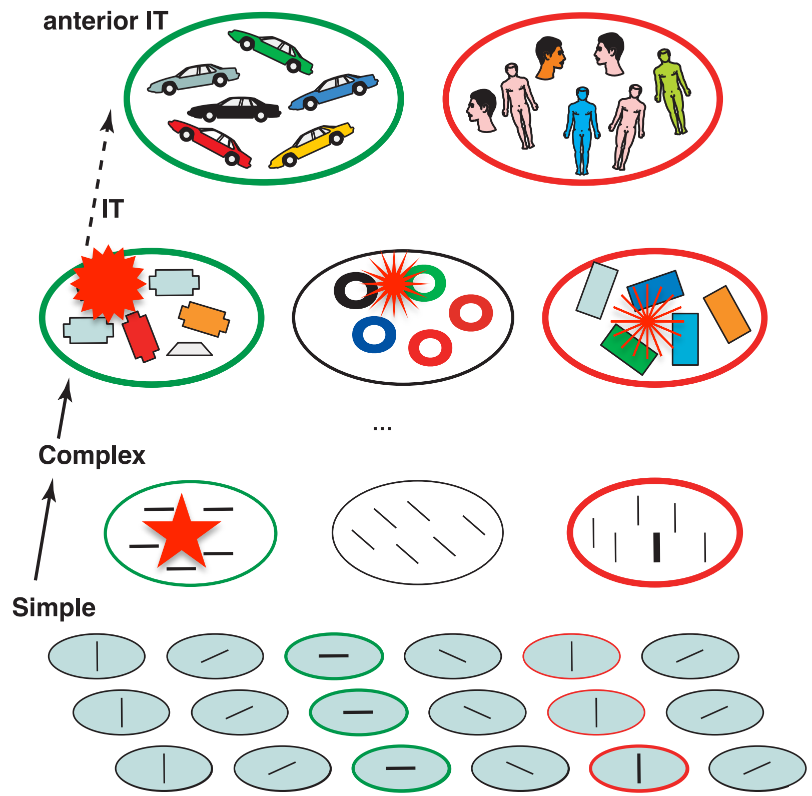


Ahissar & Hochstein, 2004

The coupling problem

?

Change V1



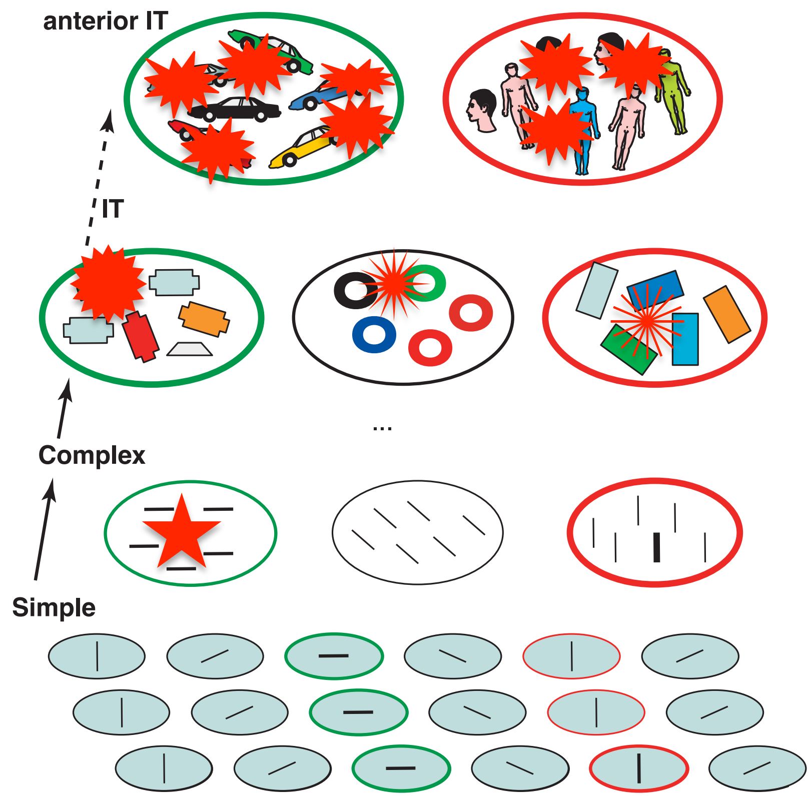
Ahissar & Hochstein, 2004

The coupling problem

?

?

Change V1



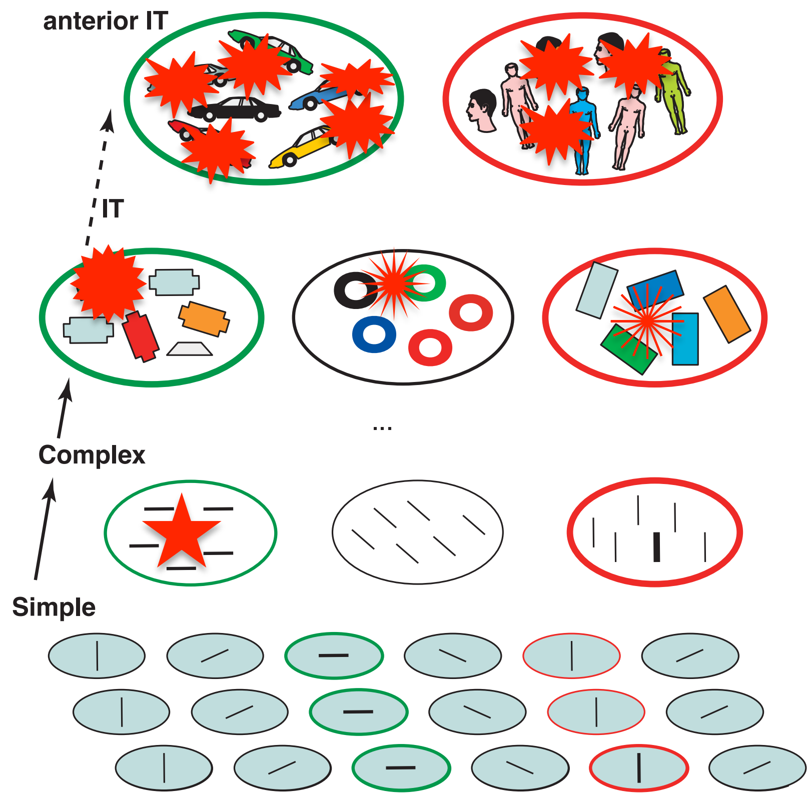
Ahissar & Hochstein, 2004

The coupling problem

?

?

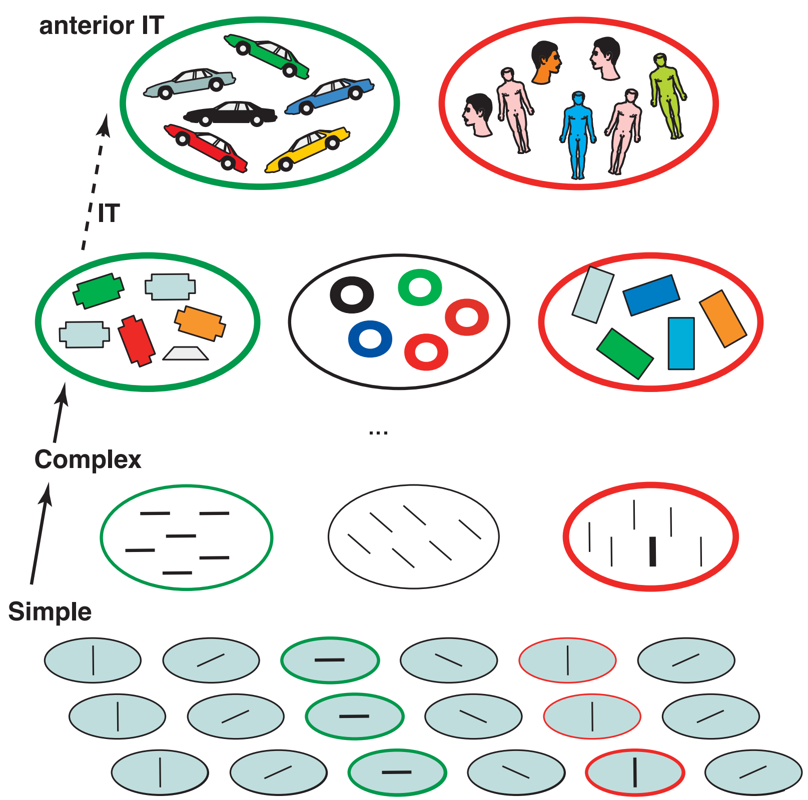
Change V1



Must consider how a change propagates to output

Ahissar & Hochstein, 2004

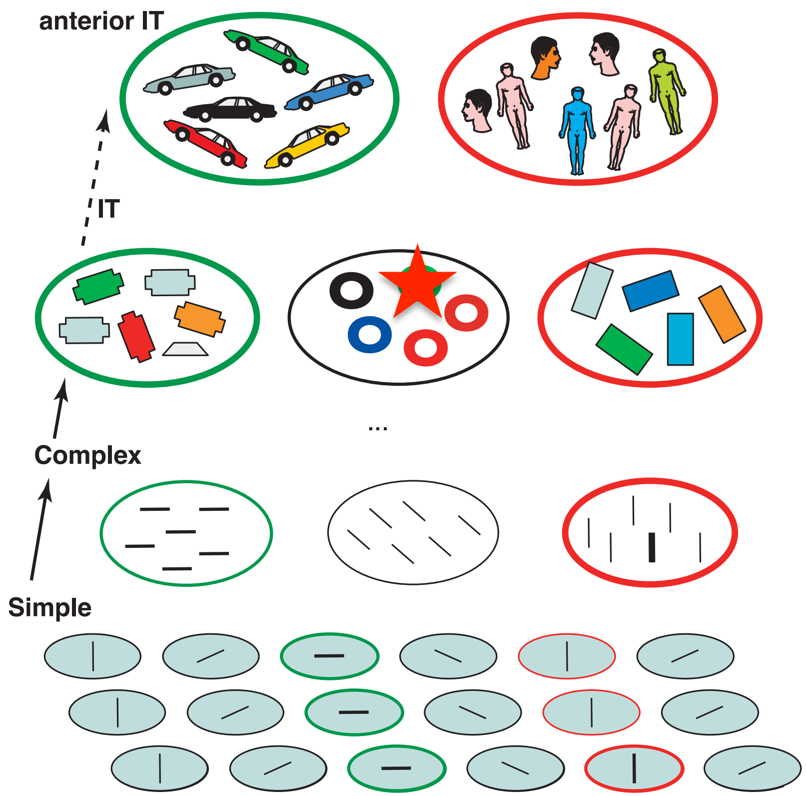
The symmetry problem



Ahissar & Hochstein, 2004

The symmetry problem

Change V4

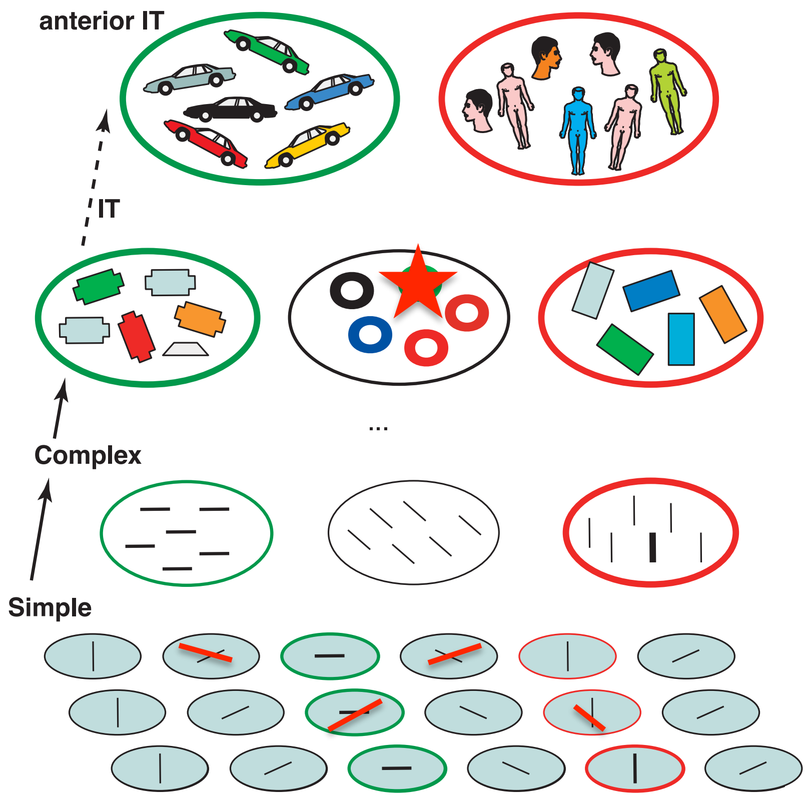


Ahissar & Hochstein, 2004

The symmetry problem

Change V4

Change V1

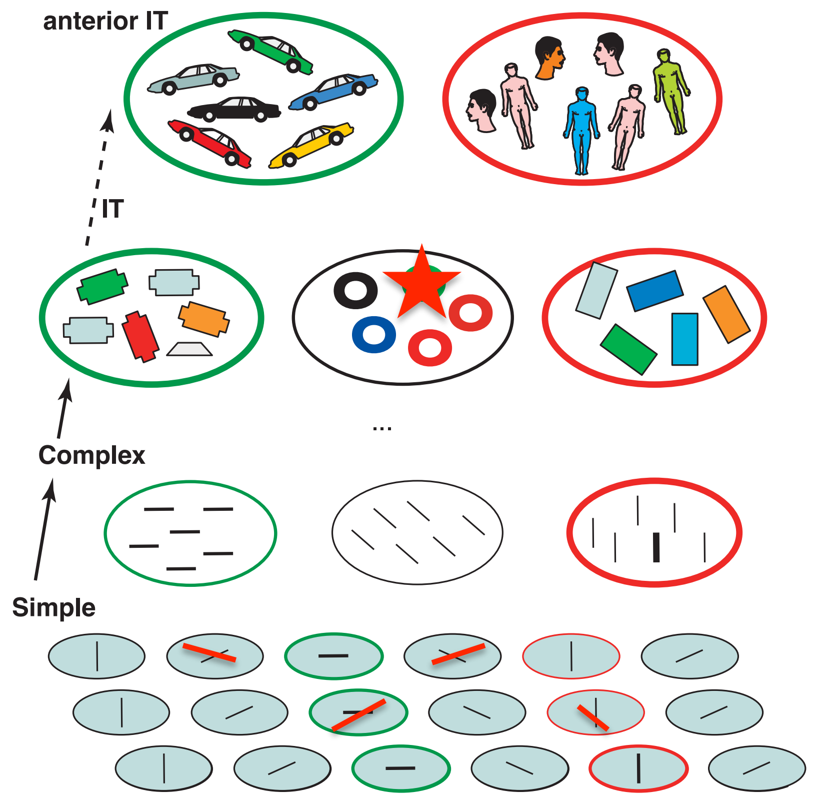


Ahissar & Hochstein, 2004

The symmetry problem

Change V4

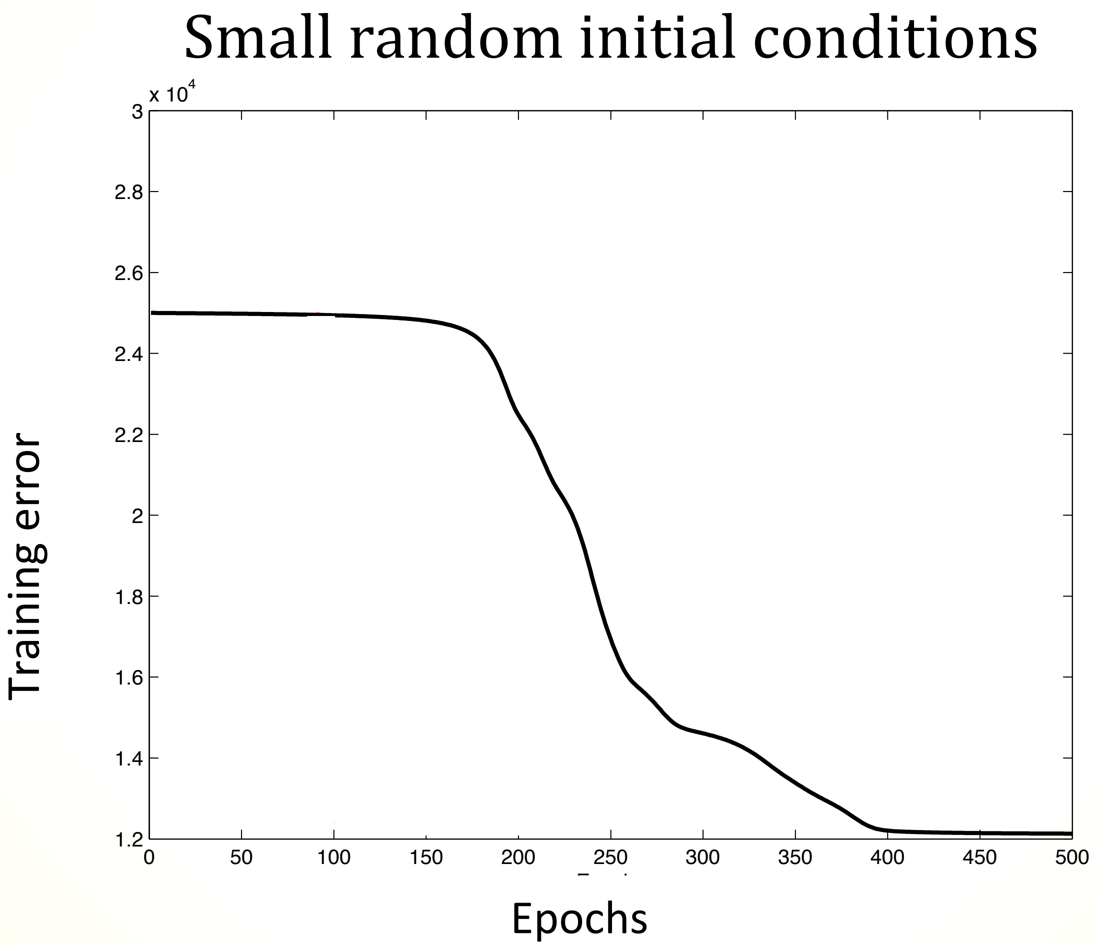
Change V1



Many equivalent changes—must choose one

Ahissar & Hochstein, 2004

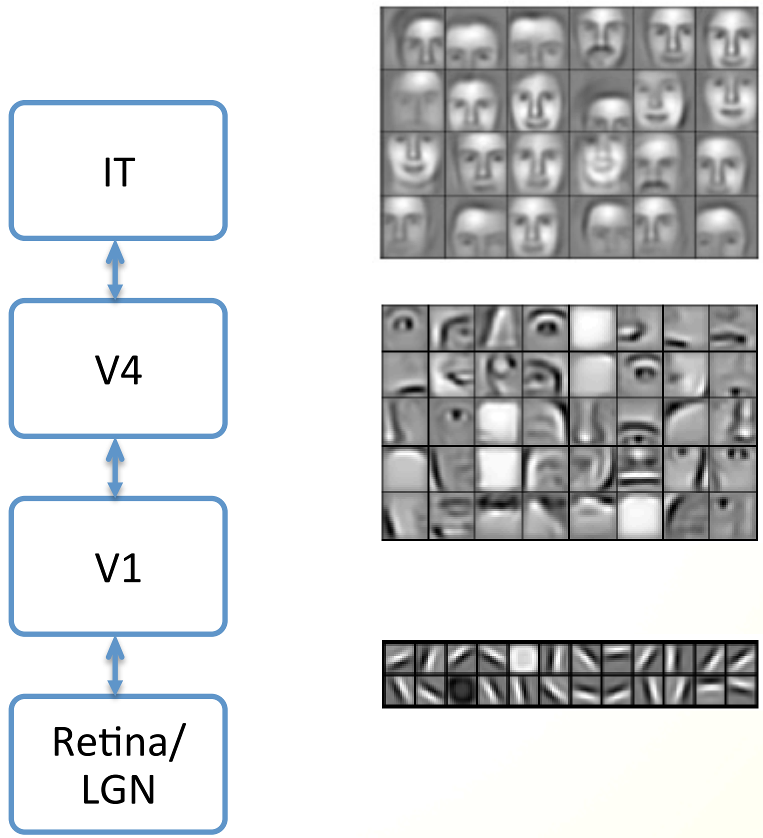
Slow learning



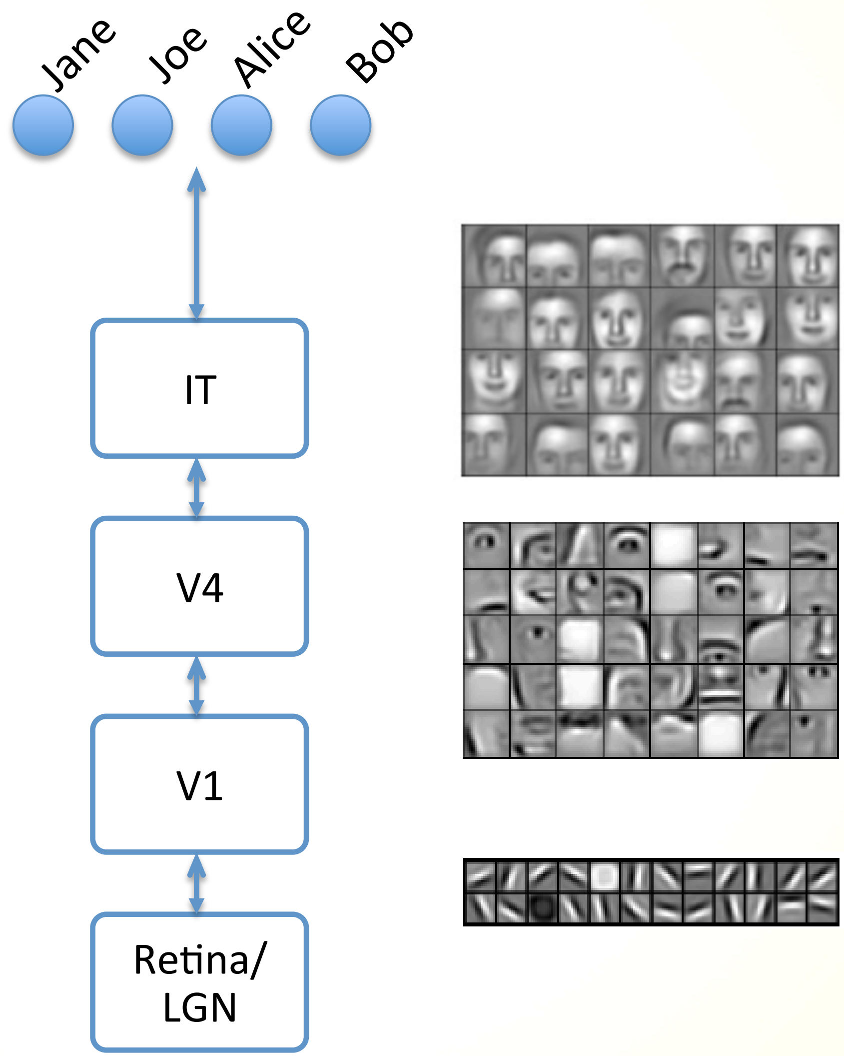
Breakthrough: Unsupervised layerwise pretraining

Suppose you want to recognize faces.

First learn a rich hierarchy of general purpose features for the visual world.



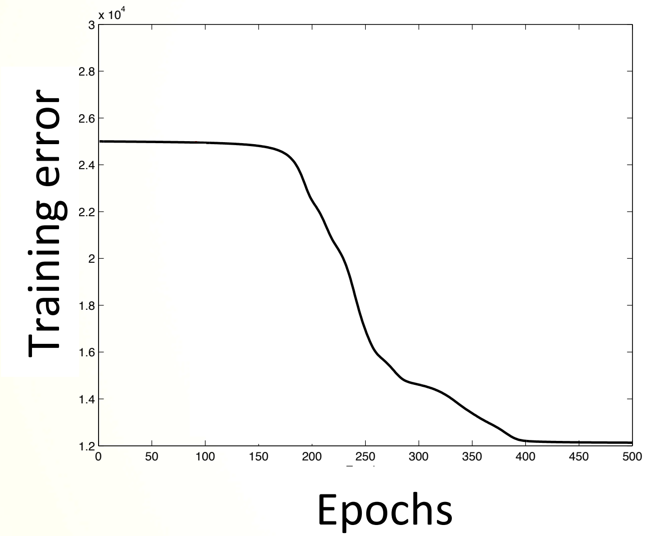
Supervised fine tuning



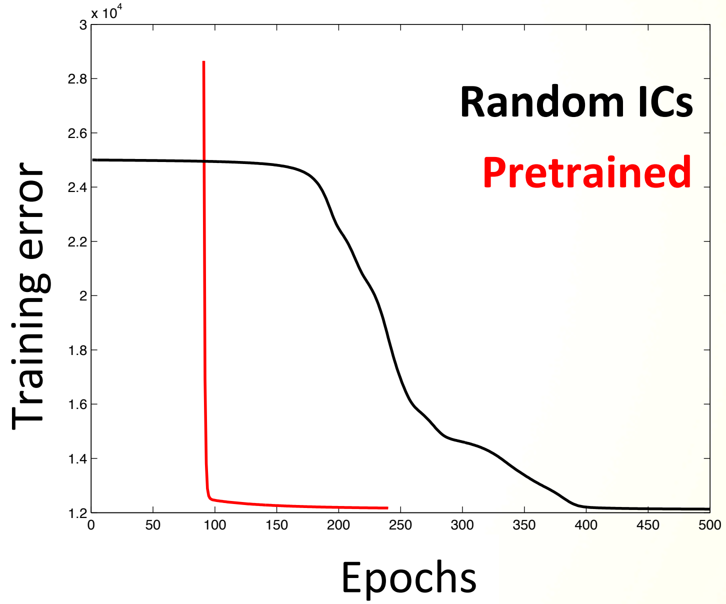
Then learn the actual task you care about.

Faster deep learning

Small random initial conditions



Pretrained initial conditions



Computational hypotheses

- **H1:** Depth enables compact representation of complex tasks
- **H2:** Naïve deep learning is slow
- **H3:** Unsupervised layerwise pretraining speeds deep learning
- **H4:** Unsupervised pretraining improves generalization
- **H5:** Supervised fine tuning follows gradient direction
- **H6:** Domain general approach

Understanding Depth

- What is the specific impact of depth on learning dynamics?

Understanding Depth

- What is the specific impact of depth on learning dynamics?

- Wanted: Theory that describes size & timing of changes across layers

Outline

- Part 1: Theory of deep linear learning

- Part 2: Applications
  - Critical period plasticity
  - Perceptual learning
  - Semantic cognition
  - Perceptual decisions
  - Reinforcement learning

Towards a theory of deep learning dynamics

– What is learned when?

– How does learning speed scale with depth?

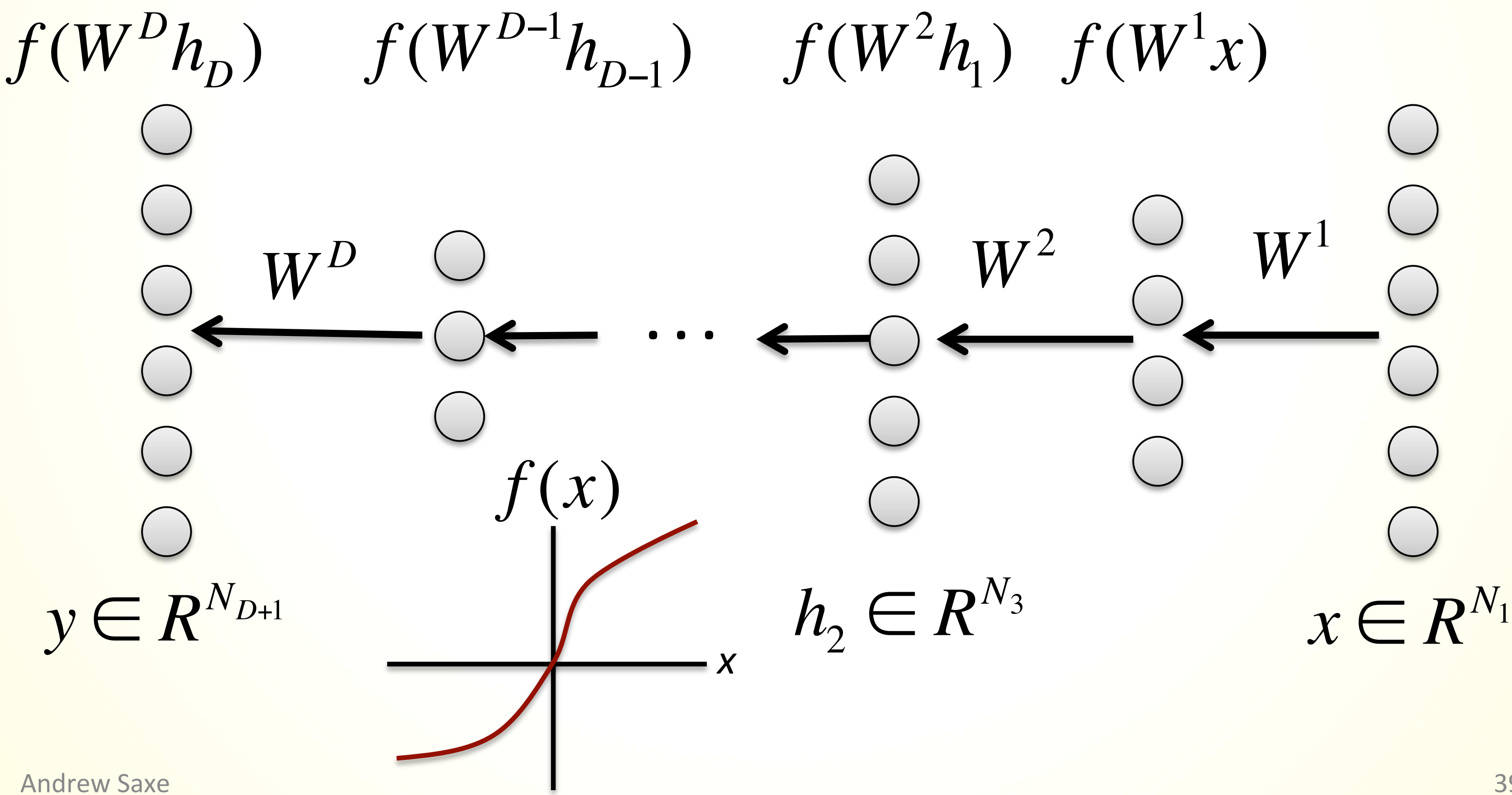
– How do different weight initializations impact learning speed?

Deep linear neural networks

- Develop theory using a simple model class
- Particularly for brain sciences, crucial to have a minimal, tractable model
  - Conceptual clarity
  - Unambiguous predictions
  - Isolate contribution of depth

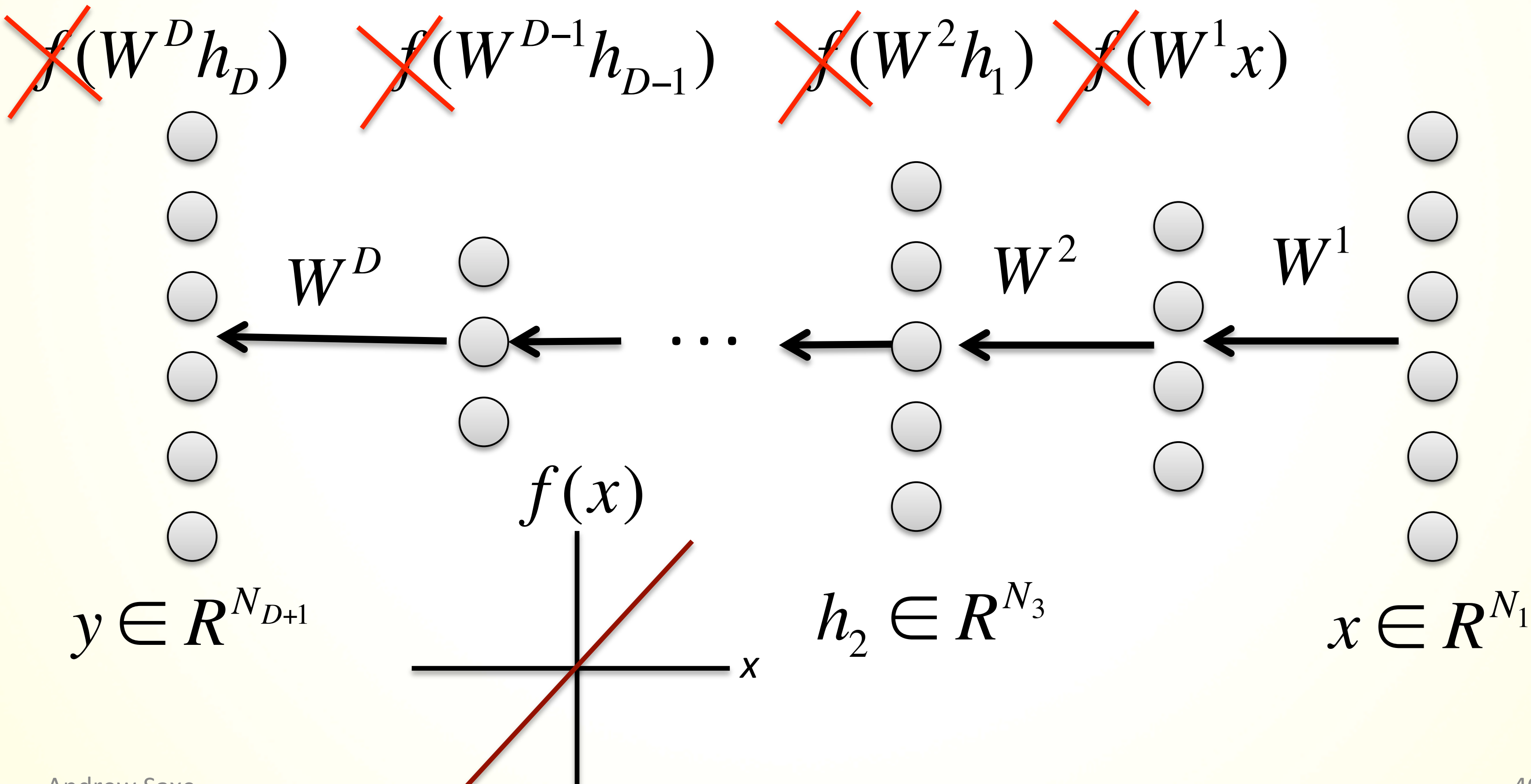
Deep network

- Little hope for a complete theory with arbitrary nonlinearities



Deep *linear* network

- Use a deep *linear* network as a starting point.



Deep *linear* network

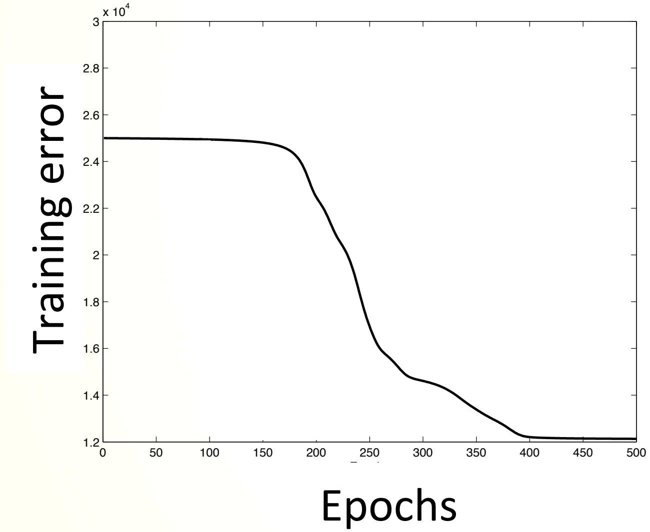
- Input-output map: Always linear

$$y = \left(\prod_{i=1}^{D} W^i \right) x \equiv W^{tot} x$$

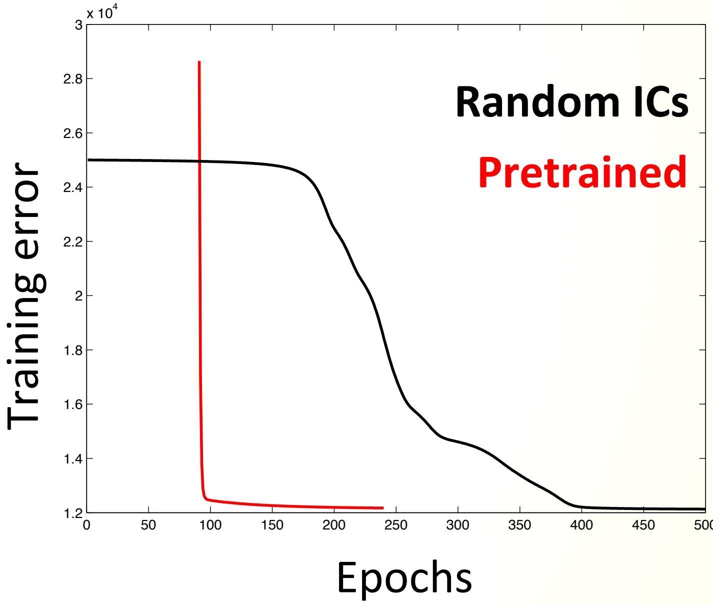
- Isolates impact of depth—little else going on

Trivial?

Plateaus and sudden transitions



Faster convergence from pretrained initial conditions



- Build intuitions for nonlinear case by analyzing linear case
- Will give exact analytic description of these error curves

Gradient descent learning

- Minimize squared error on data $\{x^\mu, y^\mu\}, \mu = 1, \ldots, P.$

$$\sum_\mu \left\| y^\mu - \left(\prod_{i=1}^{D} W^i \right) x^\mu \right\|^2$$

- Gradient descent dynamics: Nonlinear; coupled; nonconvex

$$\Delta W^l = \lambda \sum_{\mu=1}^{P} \left(\prod_{i=l+1}^{D} W^i \right)^T \left[y^\mu x^{\mu T} - \left(\prod_{i=1}^{D} W^i \right) x^\mu x^{\mu T} \right] \left(\prod_{i=1}^{l-1} W^i \right)^T$$

$$l = 1, \cdots, D$$

- Useful for studying *learning dynamics*, not representation power.

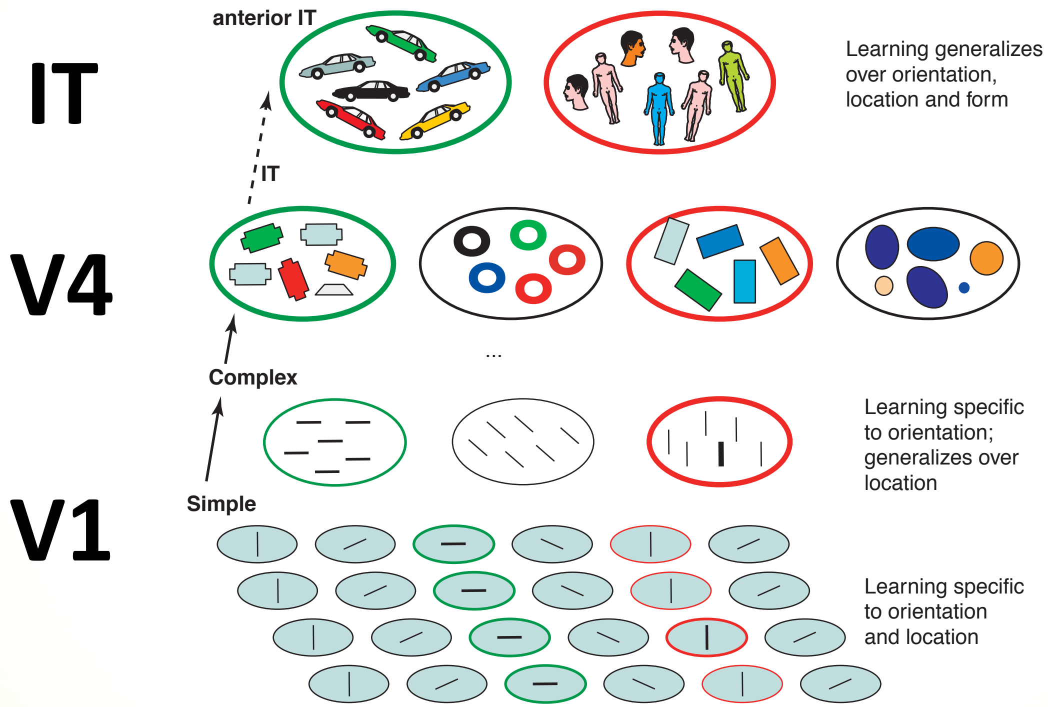
Gradient Descent Learning

- Make small change in weights that most rapidly improves task performance

- Change each weight in proportion to the gradient of the error $\Delta W = -\lambda \frac{\partial E}{\partial W}$

Resolving symmetries

- Could change IT; Could change V1

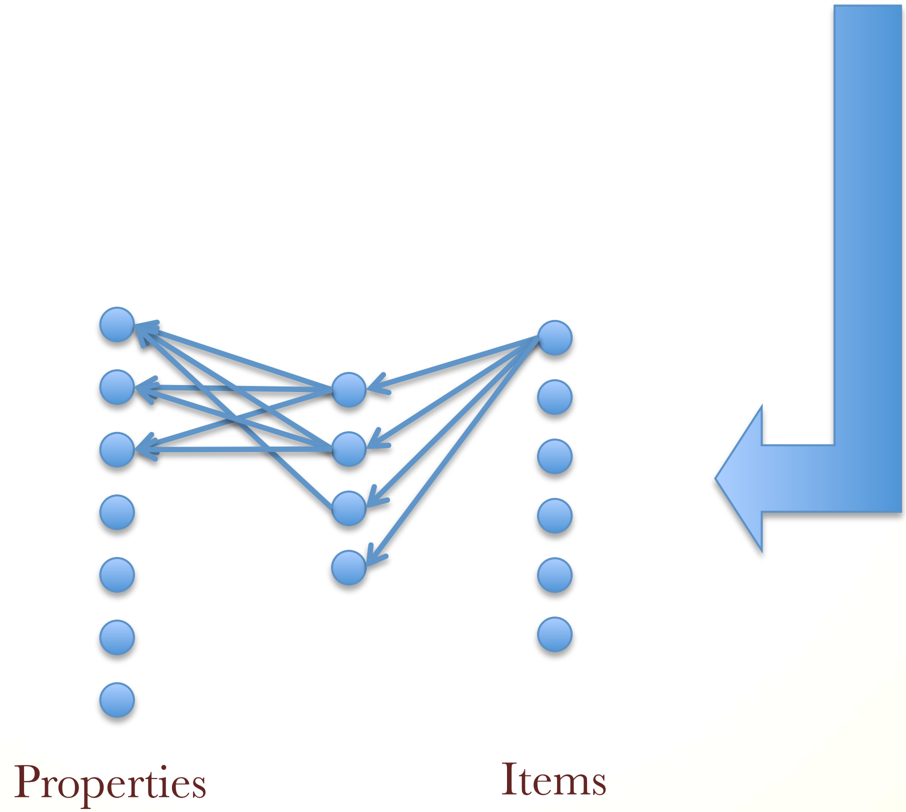


- What would most rapidly improve task performance?

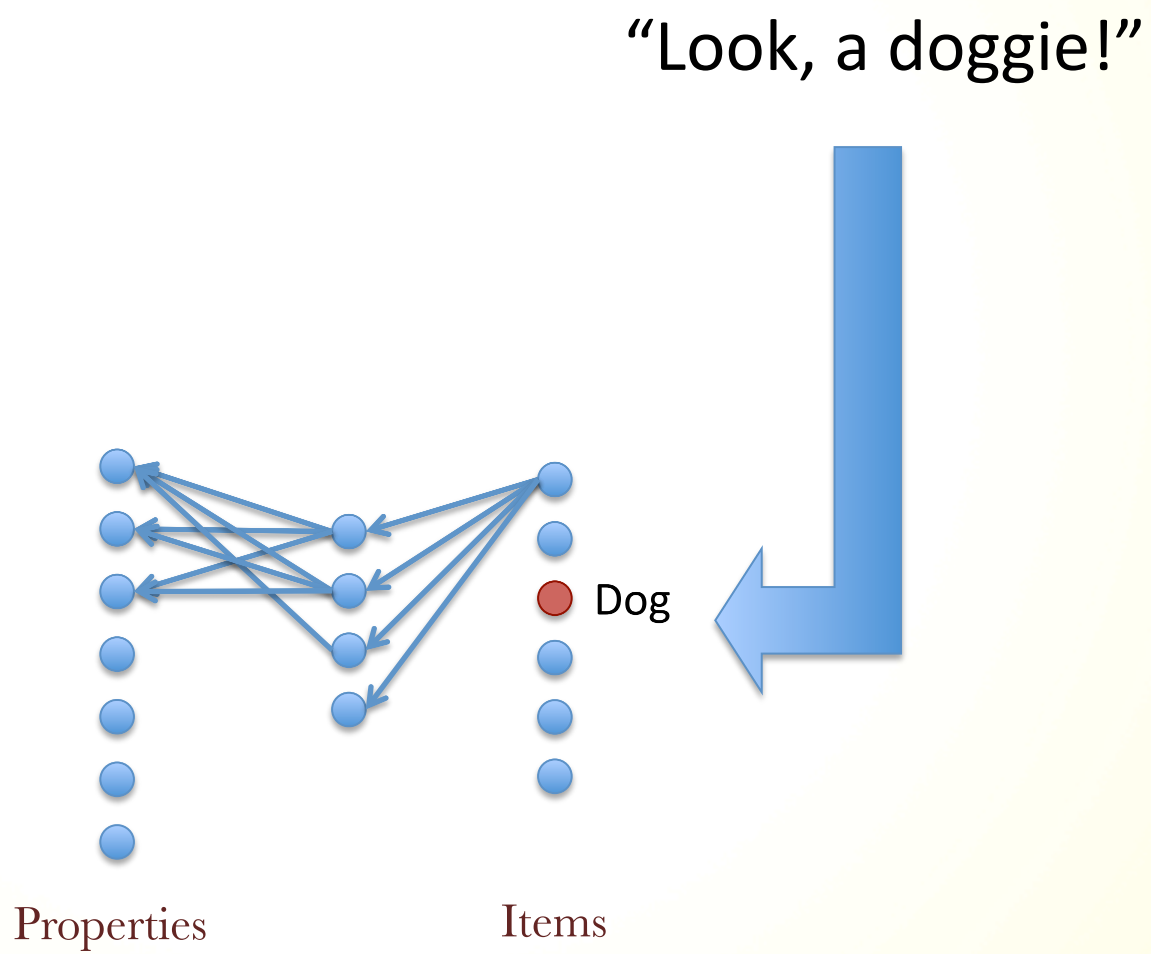
Ahissar & Hochstein, 2004

Error-corrective learning

“Look, a doggie!”

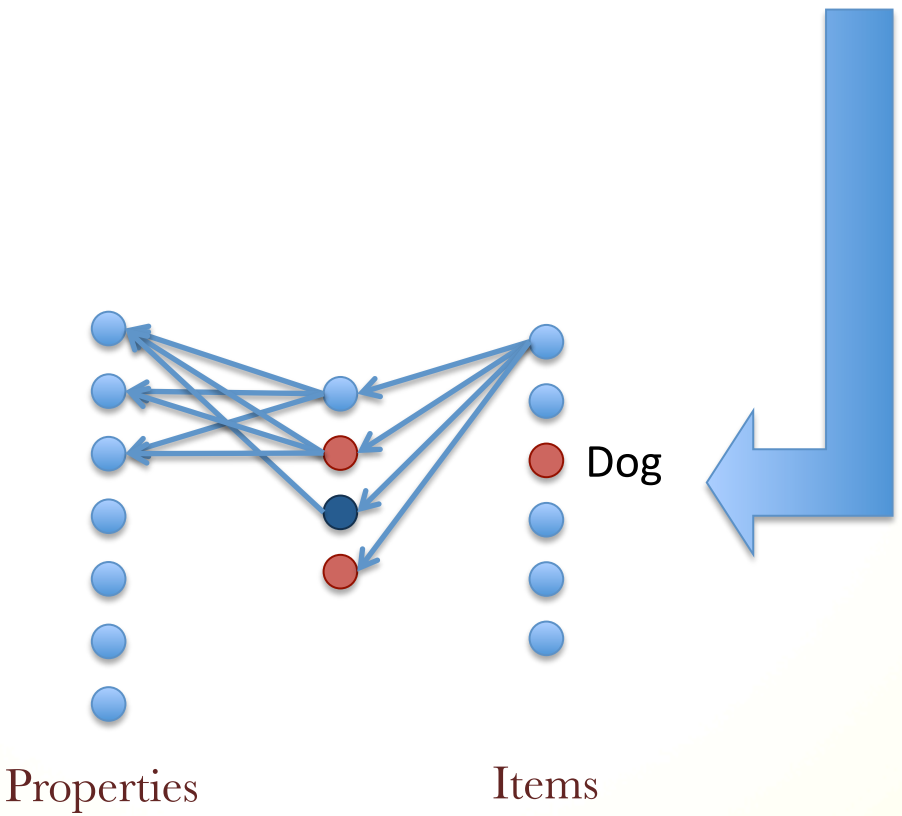


Error-corrective learning



Error-corrective learning

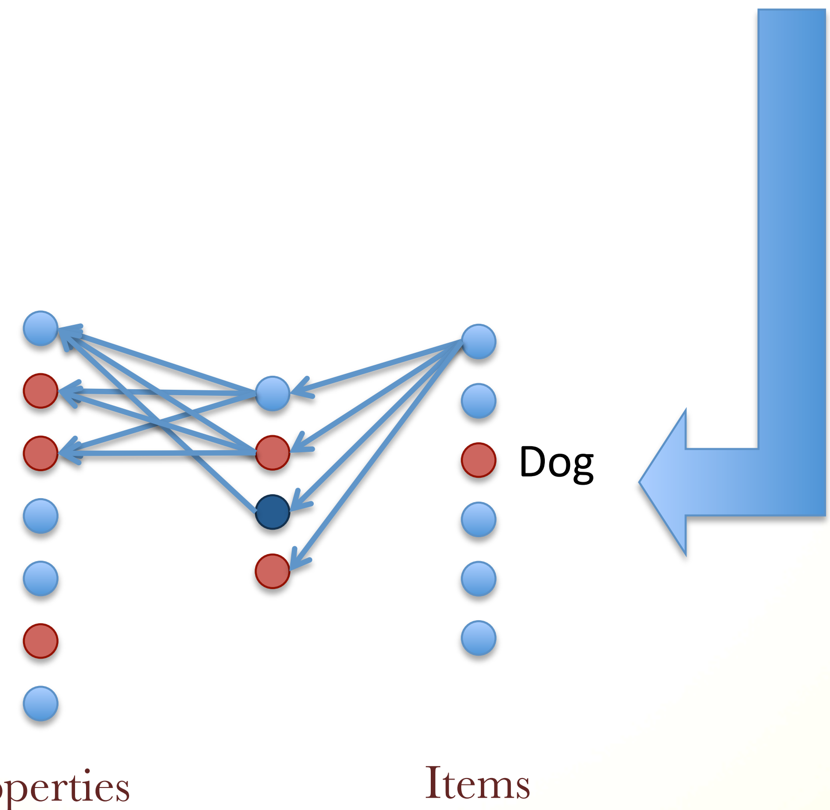
“Look, a doggie!”



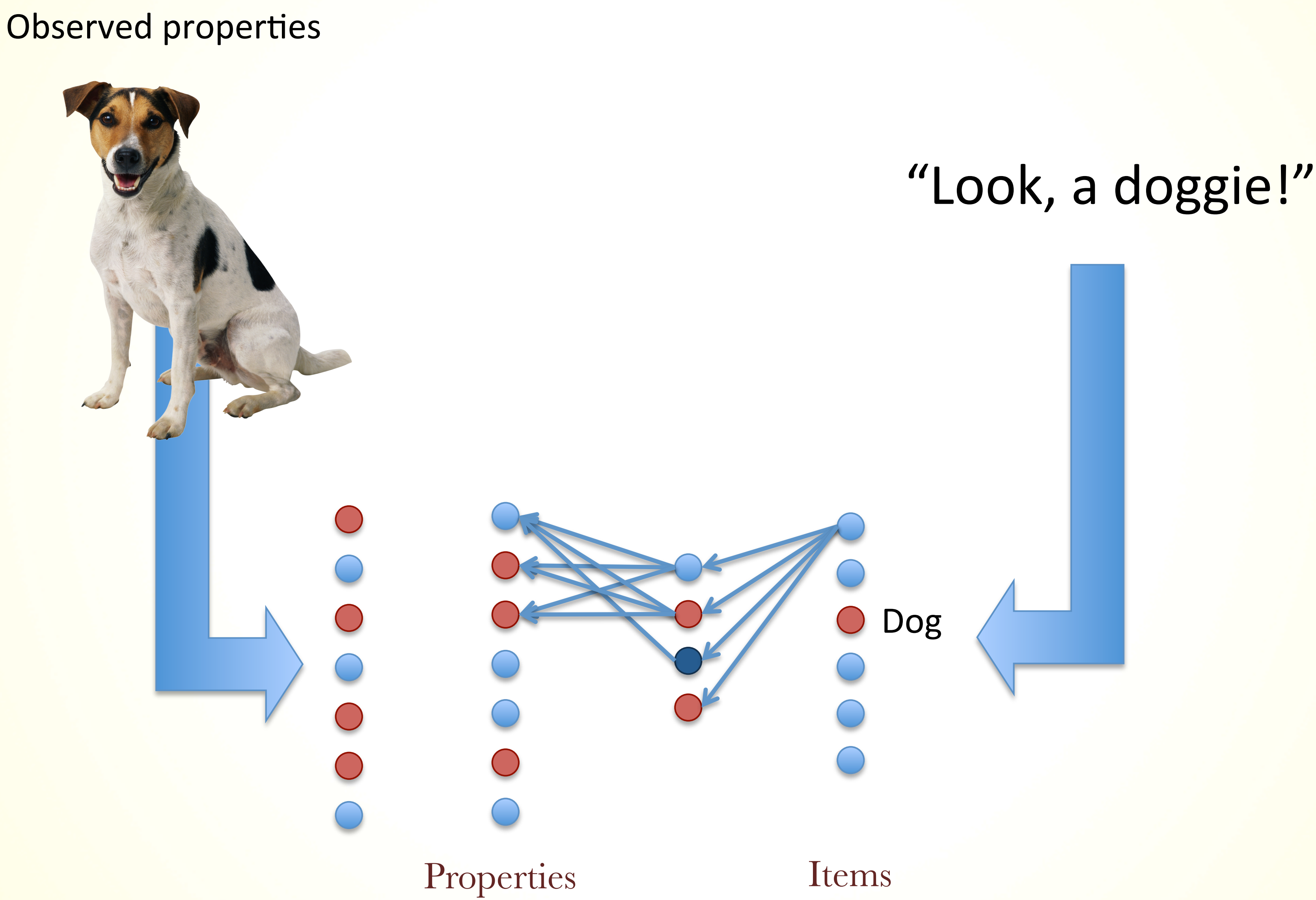
Error-corrective learning

“Look, a doggie!”

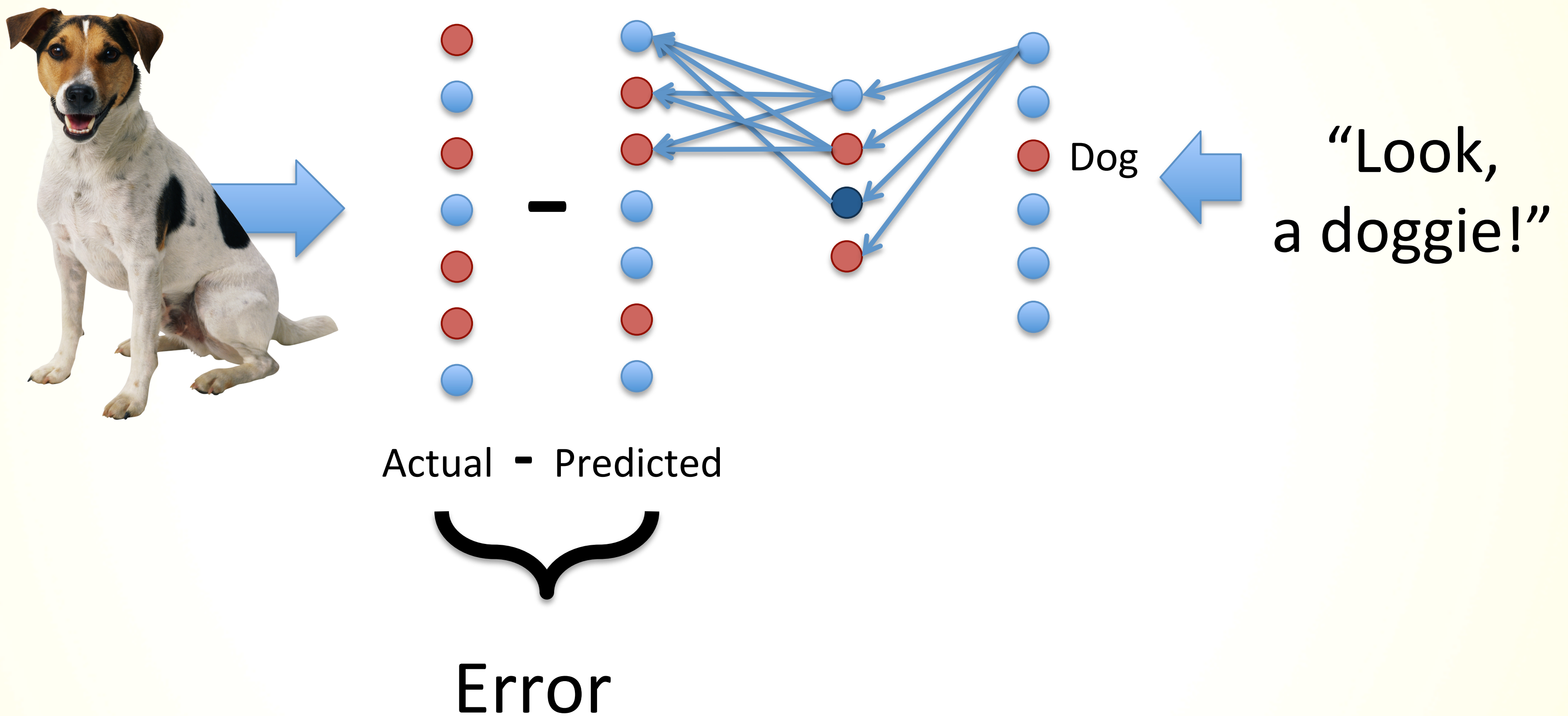
Guess properties given current connections



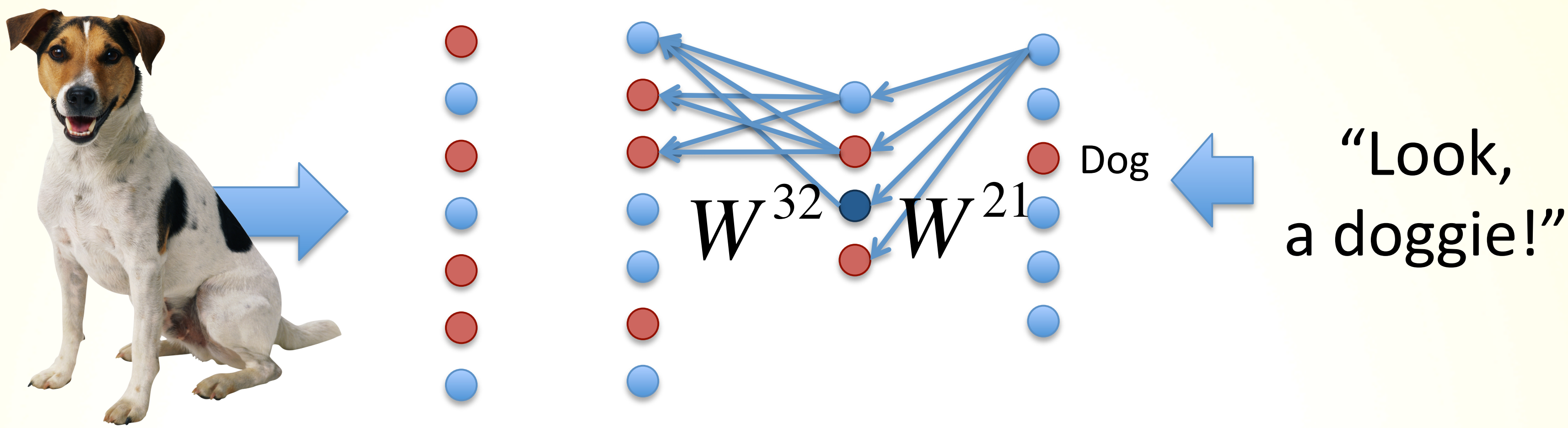
Error-corrective learning



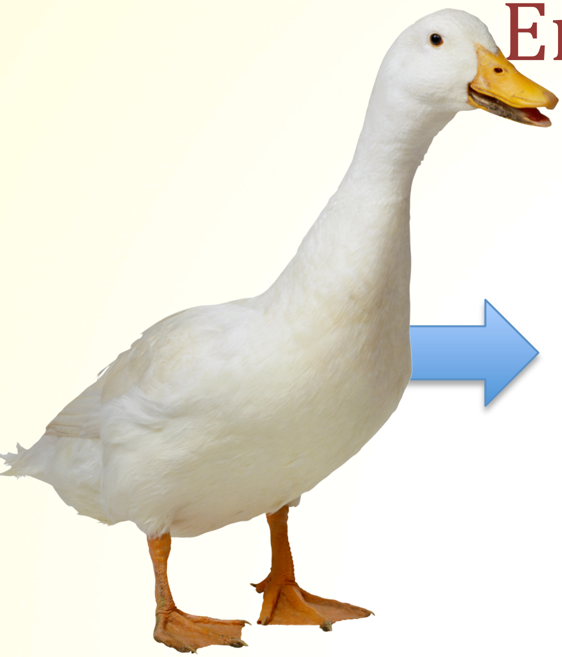
Error-corrective learning

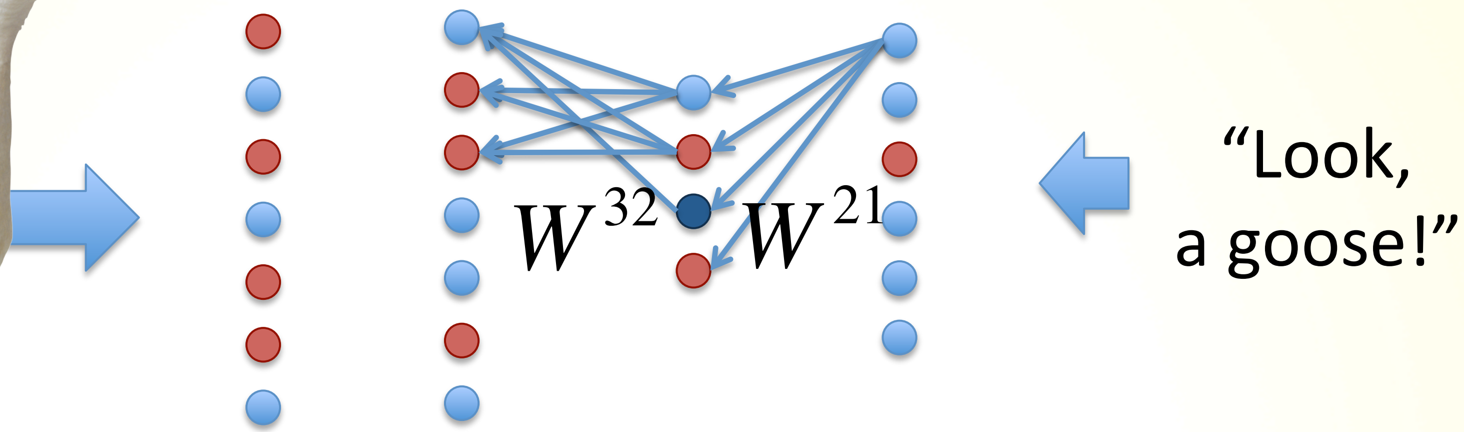


Error-corrective learning

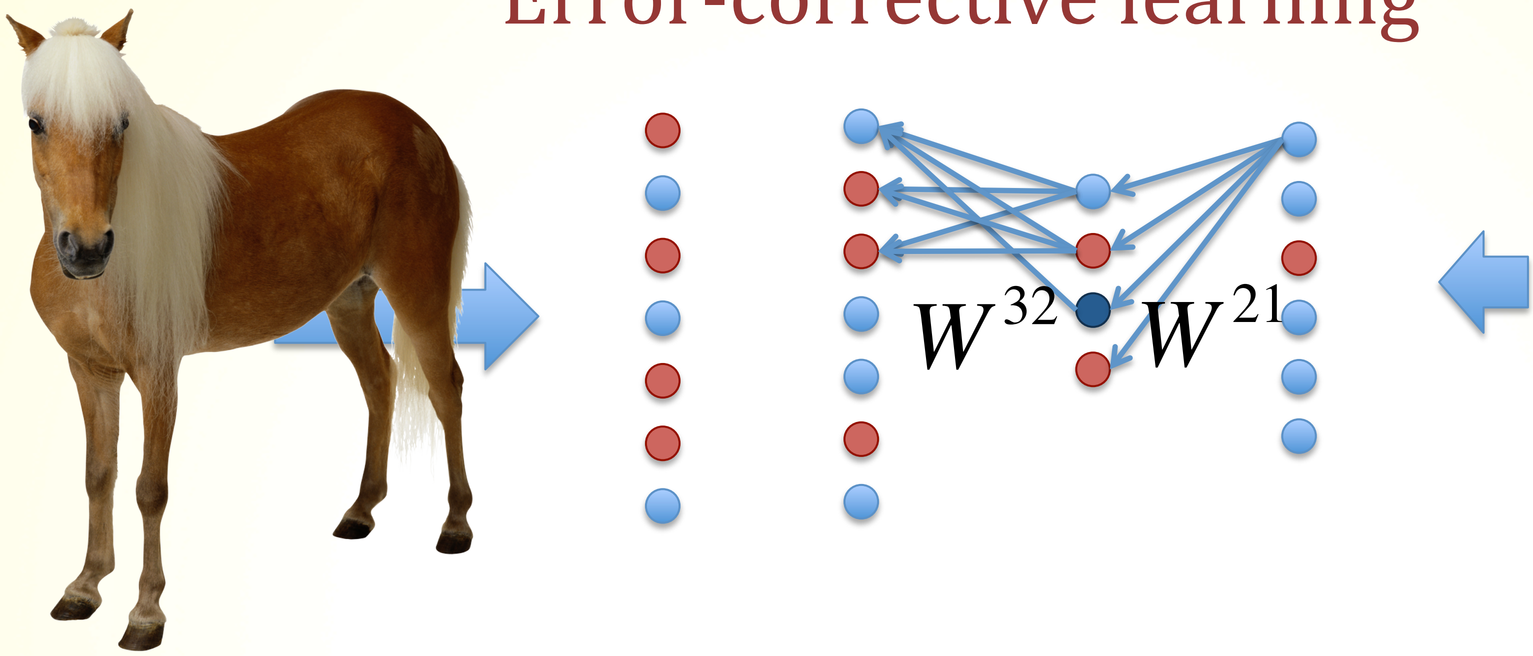


Error-corrective learning



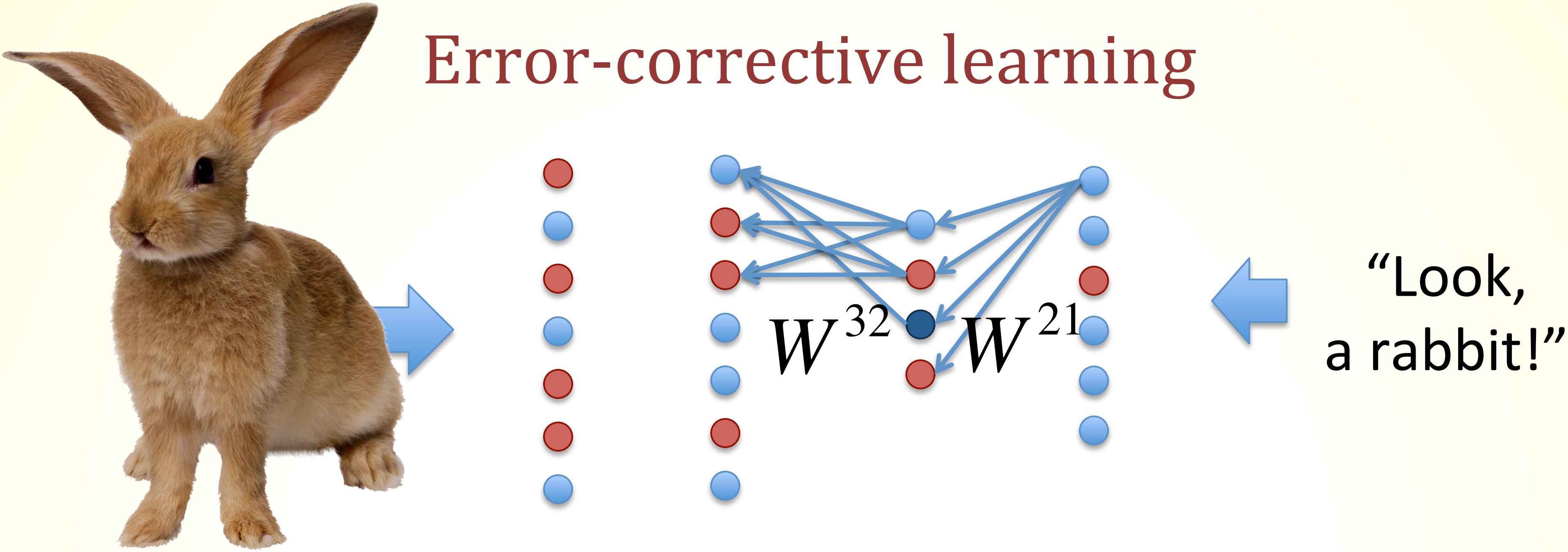


Error-corrective learning

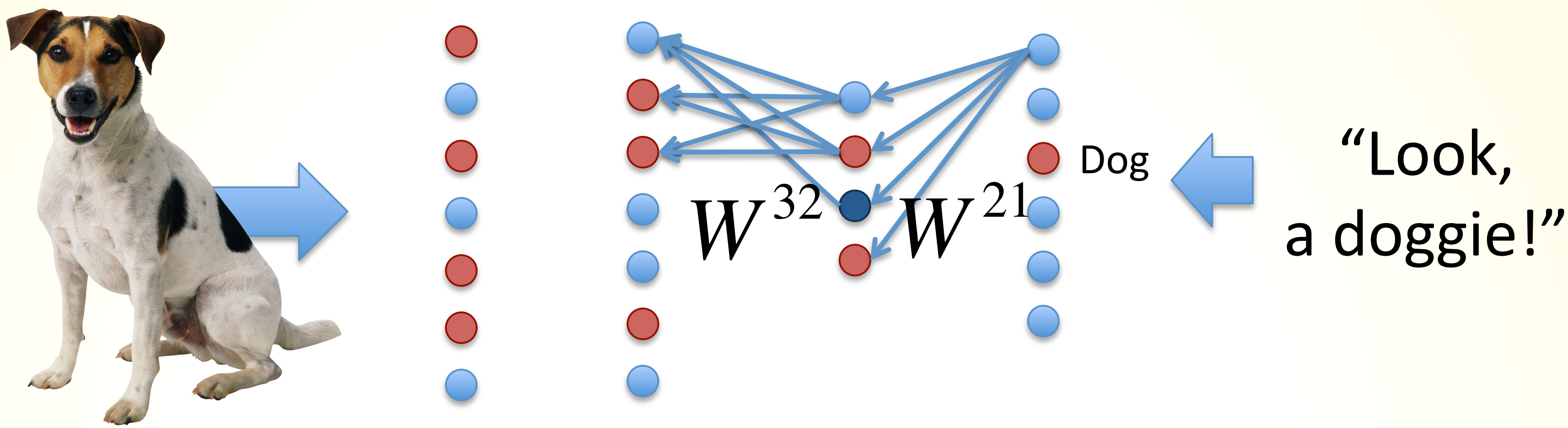


“Look,
a horse!”

Error-corrective learning

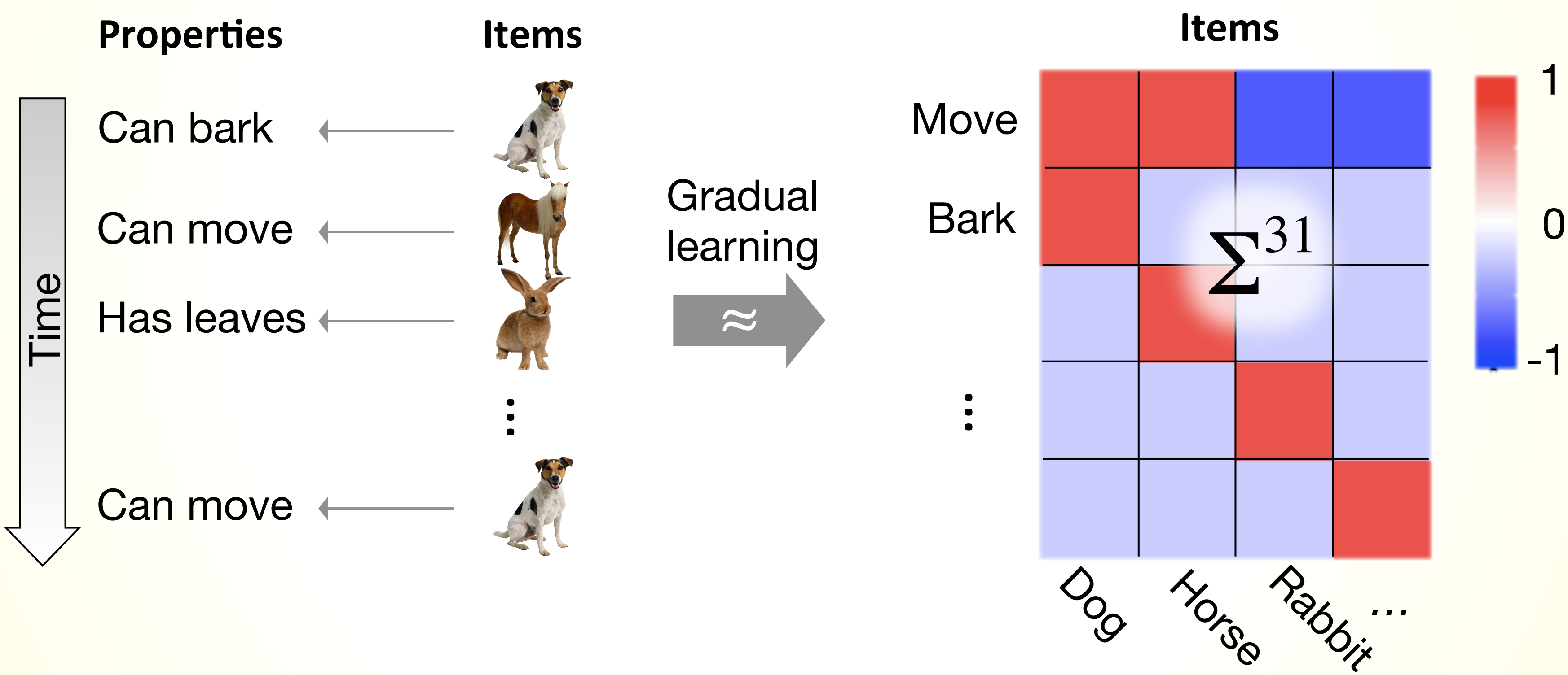


Error-corrective learning

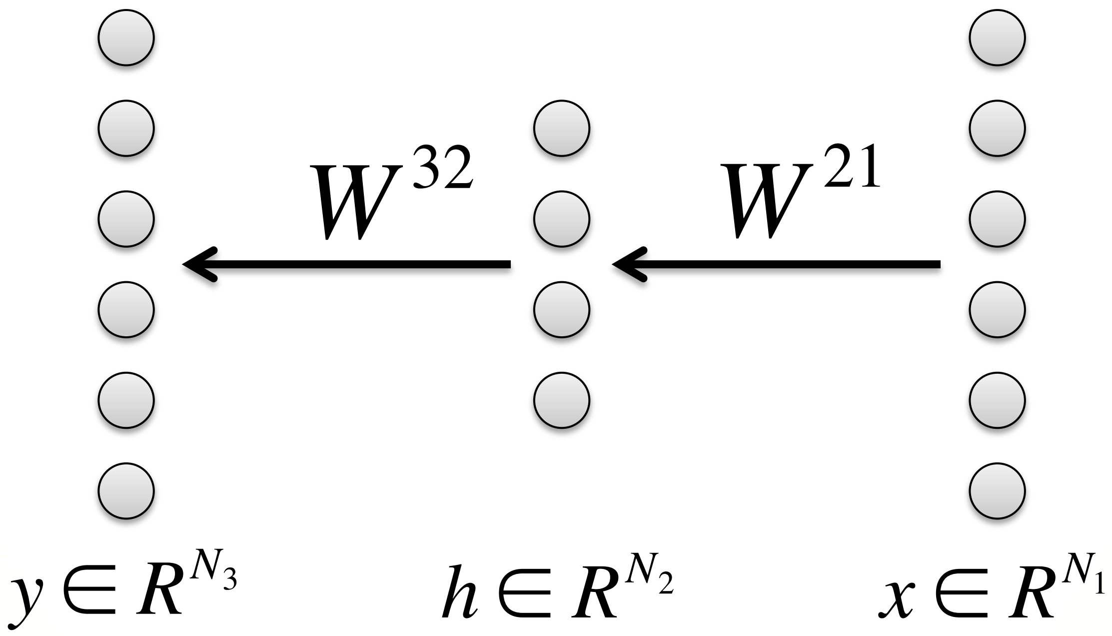


- Each experience changes weights a little
- Many small changes accumulate

From individual episodes to integrated correlations



Three layer dynamics



Problem formulation

- Network trained on patterns $\{x^\mu, y^\mu\}, \mu = 1, \ldots, P.$
- Batch gradient descent on squared error $\|Y - W^{32}W^{21}X\|_F^2$
- Dynamics

$$\tau \frac{d}{dt} W^{21} = W^{32^T} \left(\Sigma^{31} - W^{32}W^{21}\Sigma^{11}\right)$$

$$\tau \frac{d}{dt} W^{32} = \left(\Sigma^{31} - W^{32}W^{21}\Sigma^{11}\right) W^{21^T}$$

Input correlations: $\Sigma^{11} \equiv E\left[xx^T\right] = I$

Input-output correlations: $\Sigma^{31} \equiv E\left[yx^T\right]$

(see paper for more general input correlations)

Fixed points (Baldi & Hornik, 1989)

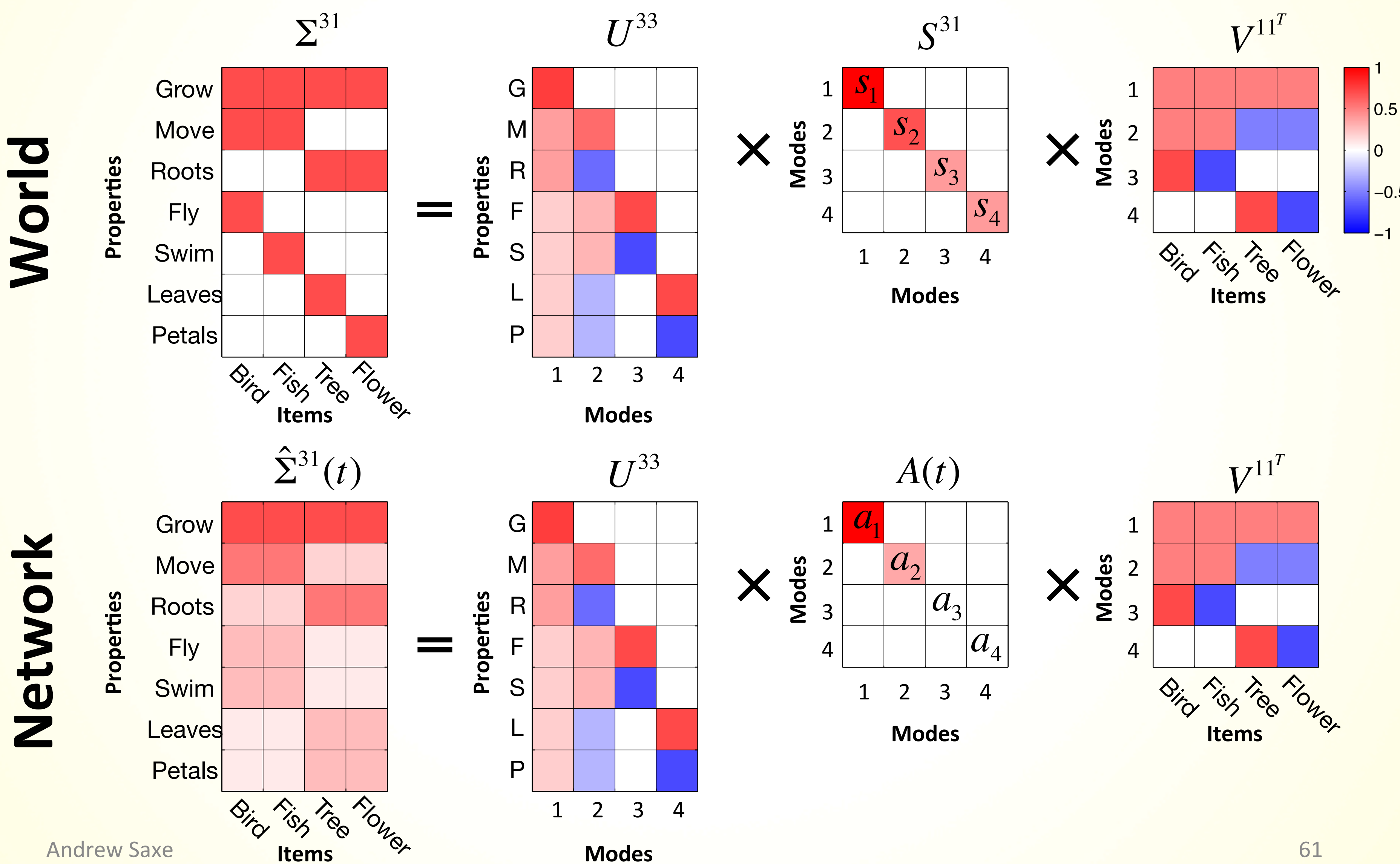
- All fixed points are global minima or saddle pts
- As $t \longrightarrow \infty$, weights approach

$$W^{32}(t)W^{21}(t) \quad \longrightarrow \quad \Sigma^{31} = U^{33}S^{31}V^{11^T} = \sum_{\alpha=1}^{N_1} s_\alpha u^\alpha v^{\alpha T}$$

- (Baldi & Hornik, 1989; Sanger, 1989)

- Well-known end point of learning
- But what *dynamics* occur along the way?

SVD change of variables



Analytic learning trajectory

SVD of input-output correlations:

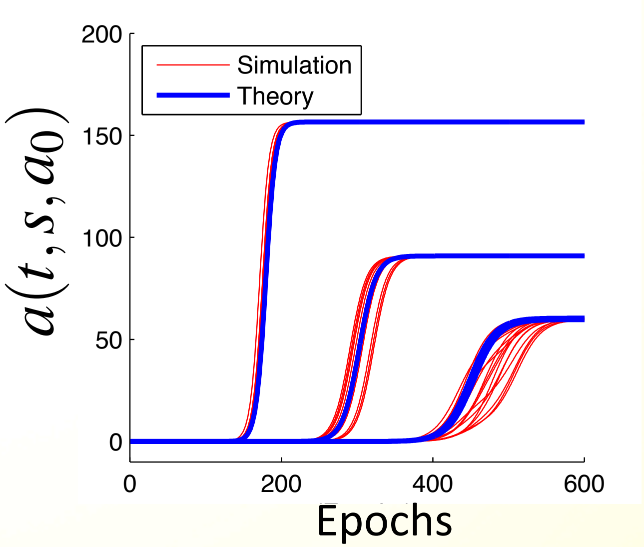
$$\Sigma^{31} = U^{33} S^{31} V^{11^T} = \sum_{\alpha=1}^{N_1} s_\alpha u^\alpha v^{\alpha T}$$

τ	1/Learning rate
s	Singular value
a_0	Initial mode strength

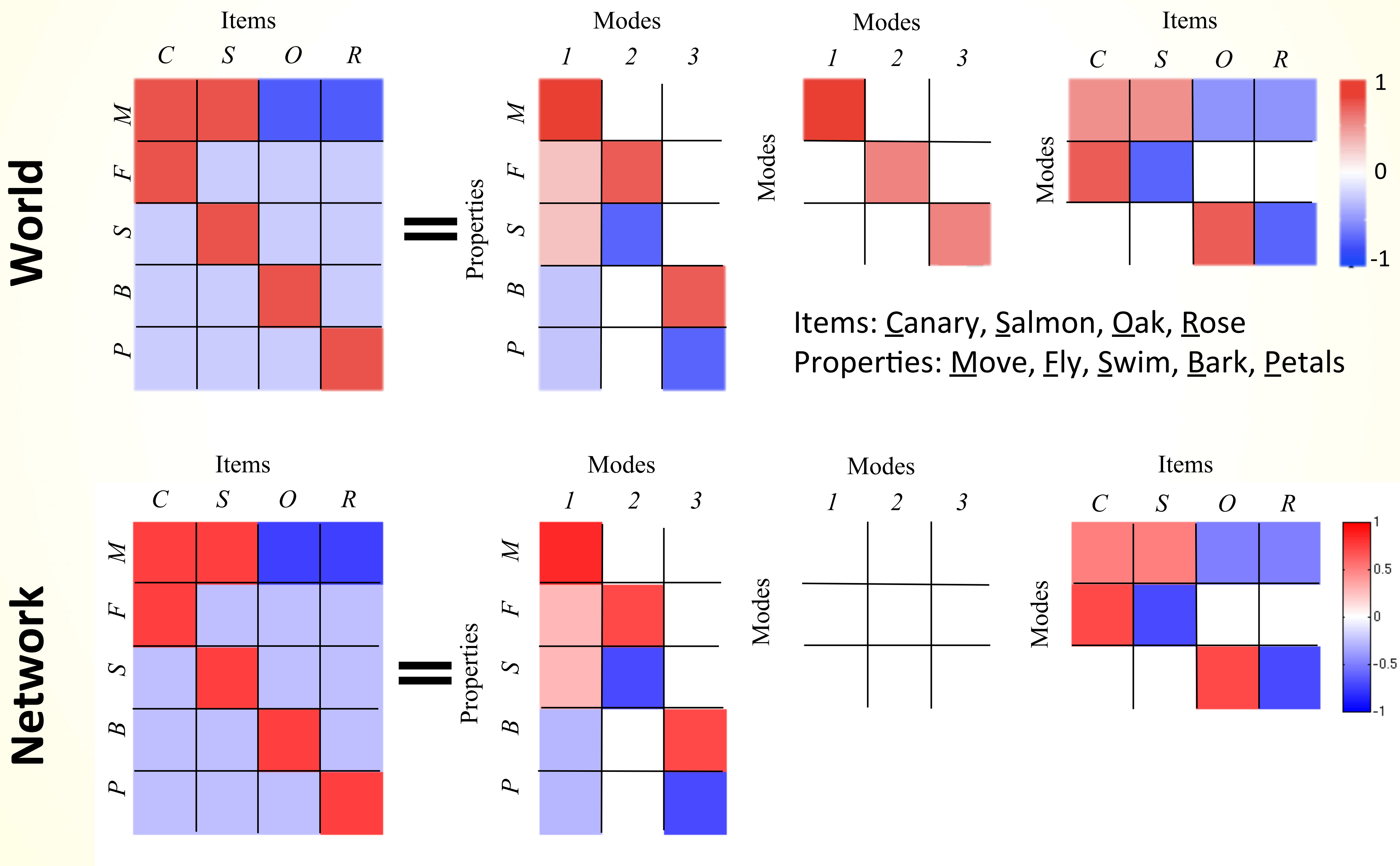
Network input-output map:

$$W^{32}(t) W^{21}(t) = \sum_{\alpha=1}^{N_2} a(t, s_\alpha, a_\alpha^0)\, u^\alpha v^{\alpha T} \quad \text{where} \quad a(t, s, a_0) = \frac{s e^{2st/\tau}}{e^{2st/\tau} - 1 + s/a_0}$$

- Starting from balanced, decoupled initial conditions
- Each mode evolves independently

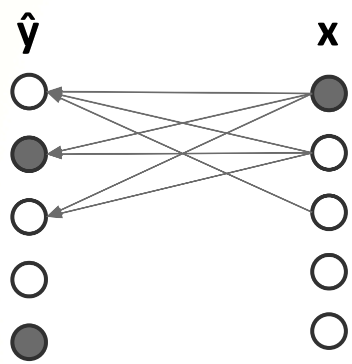


Learning dynamics

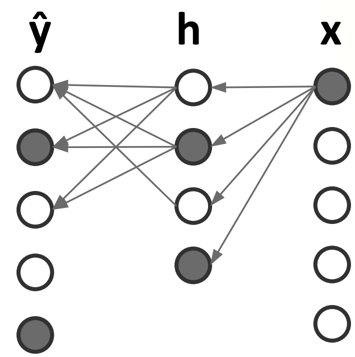


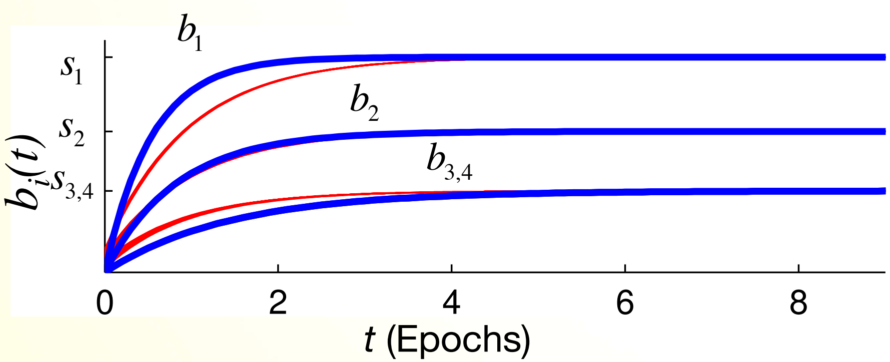
Learning dynamics

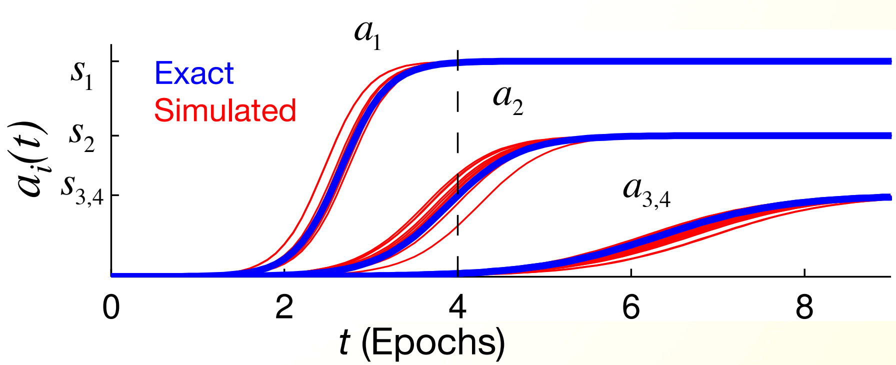
Shallow



Deep







Timescale of learning

- Each mode is **learned in time** $O(\tau/s)$

τ	1/Learning rate
s	Singular value

- Singular values of input-output correlations determine learning speed

Deeper networks

- Can generalize to arbitrary depth network

- Each effective singular value *a* evolves independently according to

$$\tau \frac{d}{dt} a = (N_l - 1) a^{2-2/(N_l-1)} (s - a)$$

τ	1/Learning rate
s	Singular value
N_l	# layers

- In deep networks, combined gradient is $O(N_l/\tau)$

Optimal learning rate scaling

- Deep net learning time depends on optimal (largest stable) learning rate
- Estimate by taking inverse of maximal eigenvalue of Hessian over relevant region
- Optimal learning rate scales as $O(1/N_l)$ (N_l = # layers)

Deep linear learning speed

- How does learning speed retard with depth?

- Time difference for deep net vs 3 layer net is

$$t_\infty - t_3 \approx O\left(s / a(0)\right)$$

s	Singular value
a(0)	Initial mode strength

- Very deep linear network can be **only a finite time slower** than shallow one!

-For special initial conditions and *O(1)* initial mode strength

Deep linear learning speed

- Intuition:
  - Gradient norm $O(N_l)$
  - Learning rate $O(1/N_l)$ (N_l = # layers)
  - Learning time $O(1)$
- Deep learning *can be fast* with the right ICs.

MNIST learning speeds

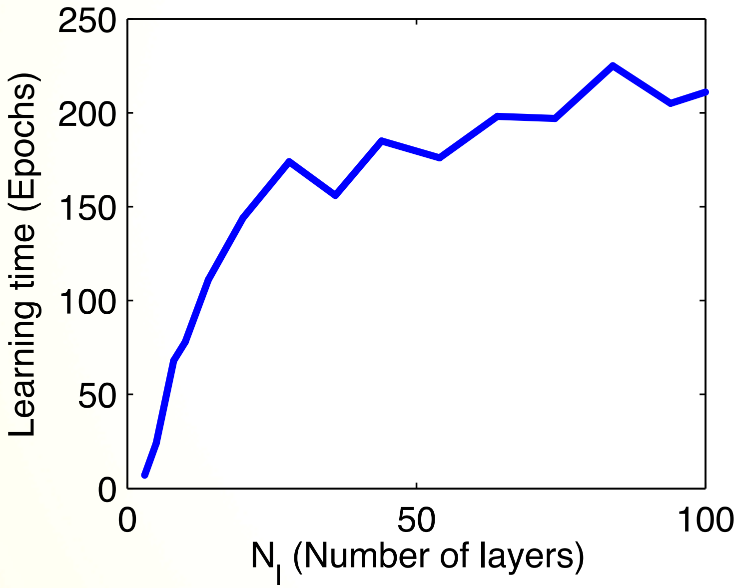
- Trained deep *linear* nets on MNIST digit classification

- Depths ranging from 3 to 100
- 1000 hidden units/layer (overcomplete)
- Decoupled initial conditions with fixed initial mode strength
- Batch gradient descent on squared error
- Optimized learning rates for each depth

- Calculated epoch at which error fell below fixed threshold

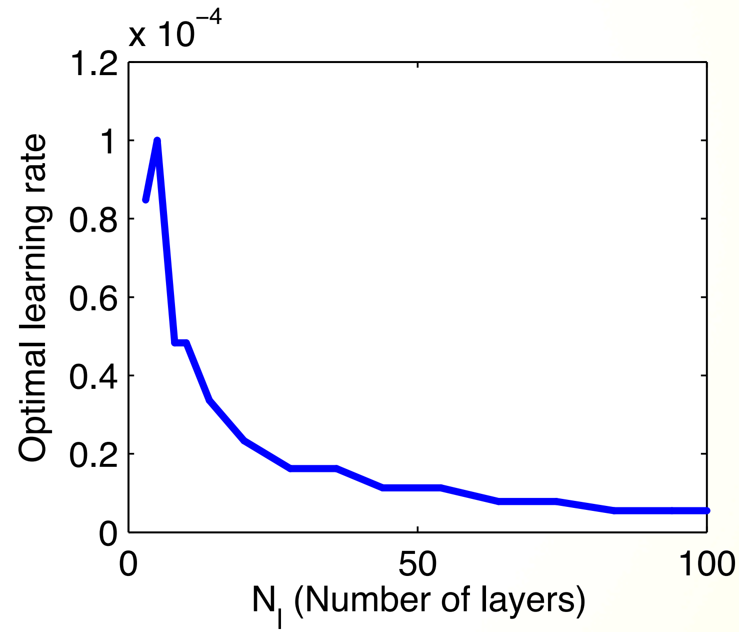
MNIST learning speeds

Time to criterion



Depth

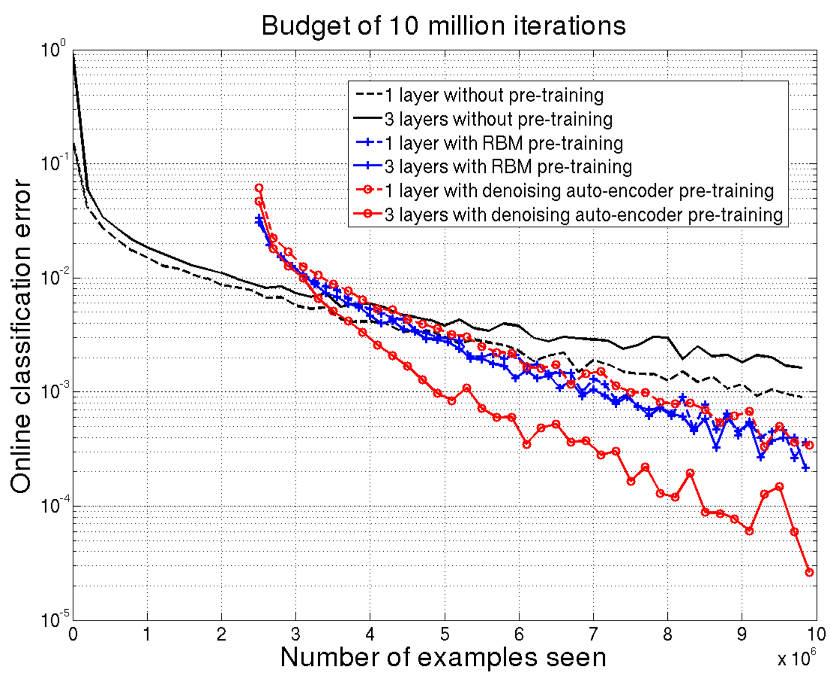
Optimal learning rate



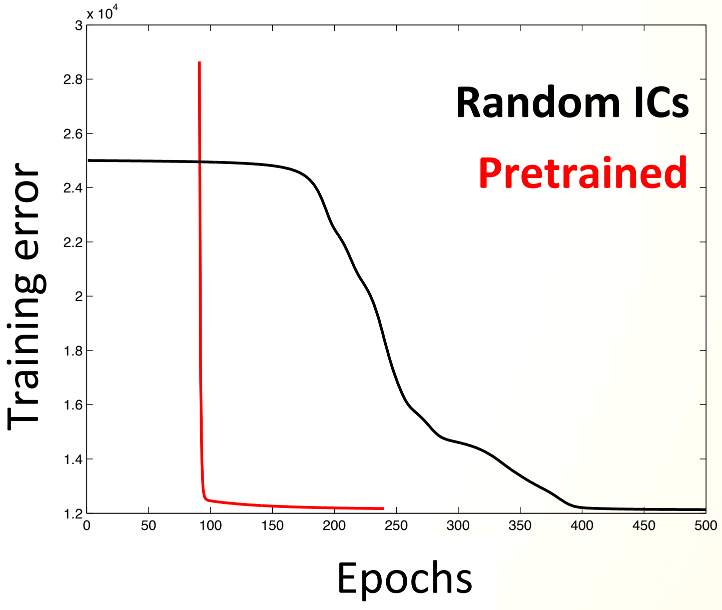
Depth

Why is unsupervised pretraining fast?

Erhan et al., 2010



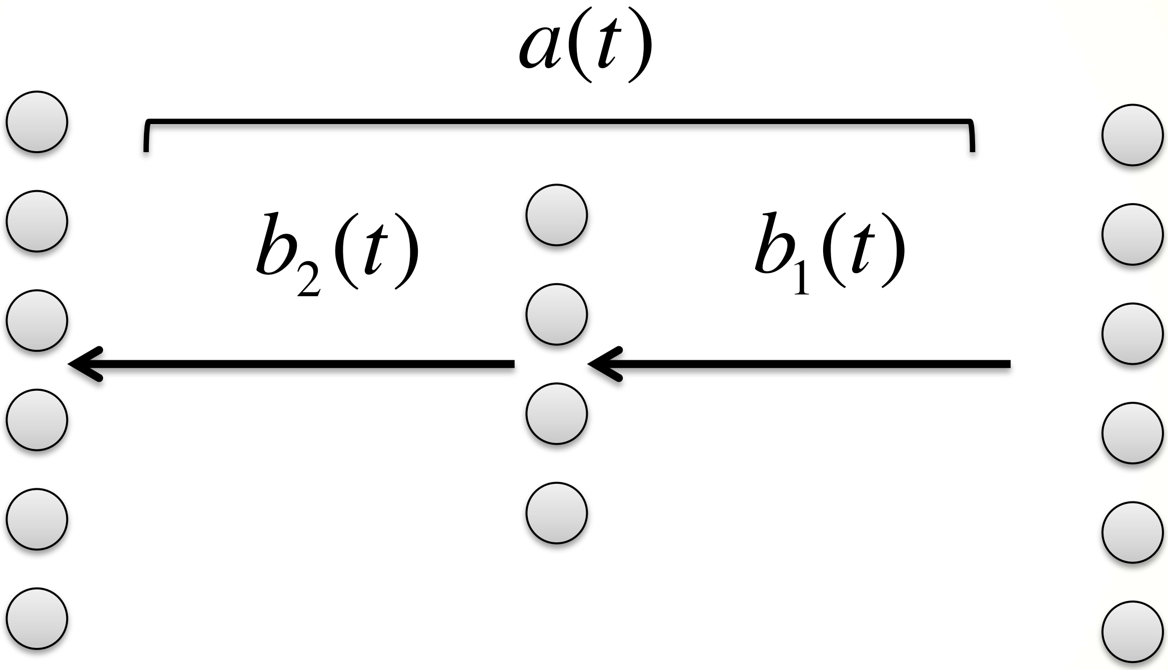
Deep linear network



The crucial question: Initial mode scaling

- Learning speed $t_\infty - t_3 \approx O\left(s / a(0)\right)$
- If *a(0)* gets smaller with more layers, deep learning is slow
- If *a(0)* stays constant with more layers, deep learning is fast

Why are small random weights slow?



$$a(t) = b_1(t) b_2(t) \cdots b_{N_l}(t)$$

Effective singular value

Layer strengths

Why are small random weights slow?

- Learning delay $t_\infty - t_3 \approx O\left(s / a(0)\right)$
- Initial scaling $a(0) = b_1(0) b_2(0) \cdots b_{N_l}(0) \approx O(c^{N_l})$
  $c < 1$
- Learning is slow due to very small initial conditions—stuck on plateau right by saddle pt
- Not due to saturating nonlinearities

Deep linear unsupervised pretraining

- Pretraining with autoencoders is simple

- Each weight matrix comes to be *orthogonal*

Why are pretrained weights fast?

- Learning delay $t_\infty - t_3 \approx O\left(s / a(0)\right)$
- Pretraining initializes all $b_i(0)$=1
- Initial scaling $a(0) = b_1(0) b_2(0) \cdots b_{N_l}(0) \approx O(1)$
- Learning is fast—have moved away from saddle pt

The effect of pretraining

- Direct training time scales exponentially with depth

$$t_{DT} \approx O\left(\frac{1}{b_0^{N_l}}\right)$$

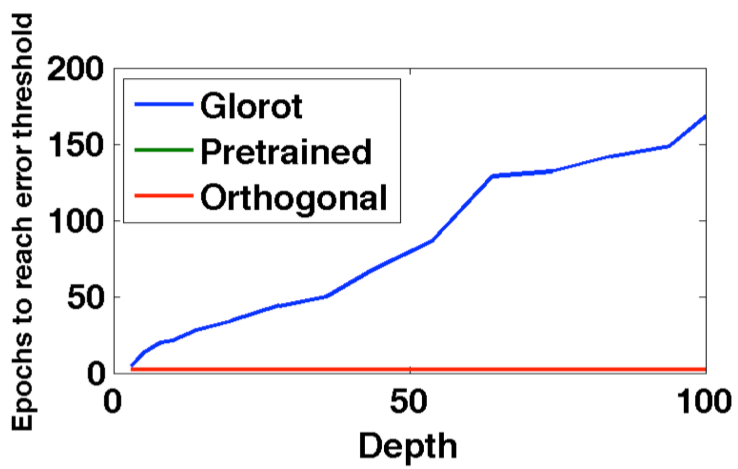
- Pretraining + fine-tuning time scales linearly with depth

$$t_{PT+FT} \approx O\left(N_l \log\left(\frac{1}{b_0^2 \varepsilon}\right)\right)$$

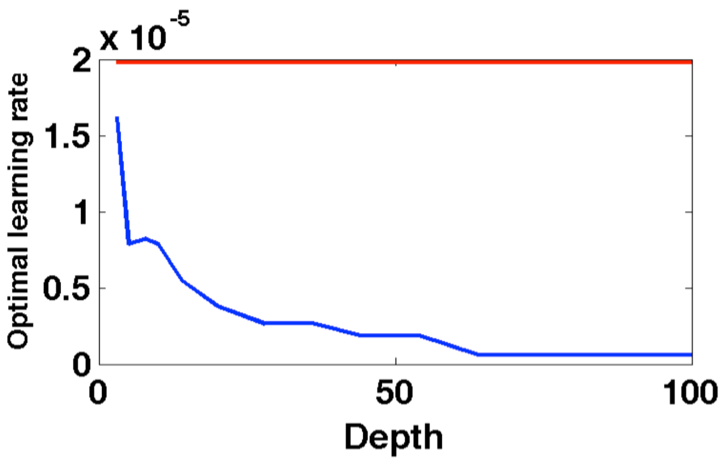
Depth-independent training time

- Deep *linear* networks on MNIST
- Glorot: Scaled random initialization (Glorot & Bengio, 2010)

Time to criterion



Optimal learning rate

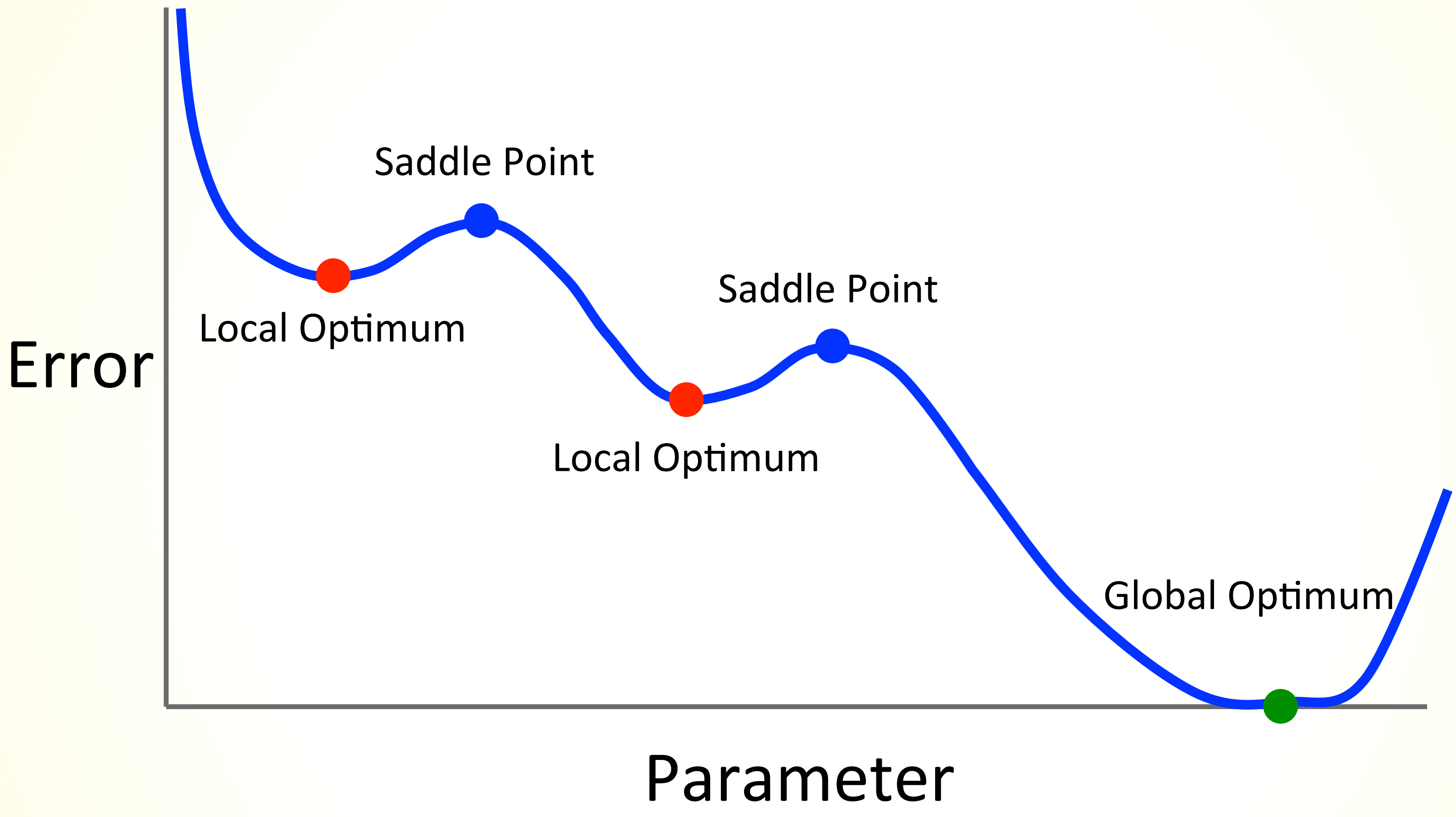


- Pretrained and orthogonal have fast **depth-independent** training times!

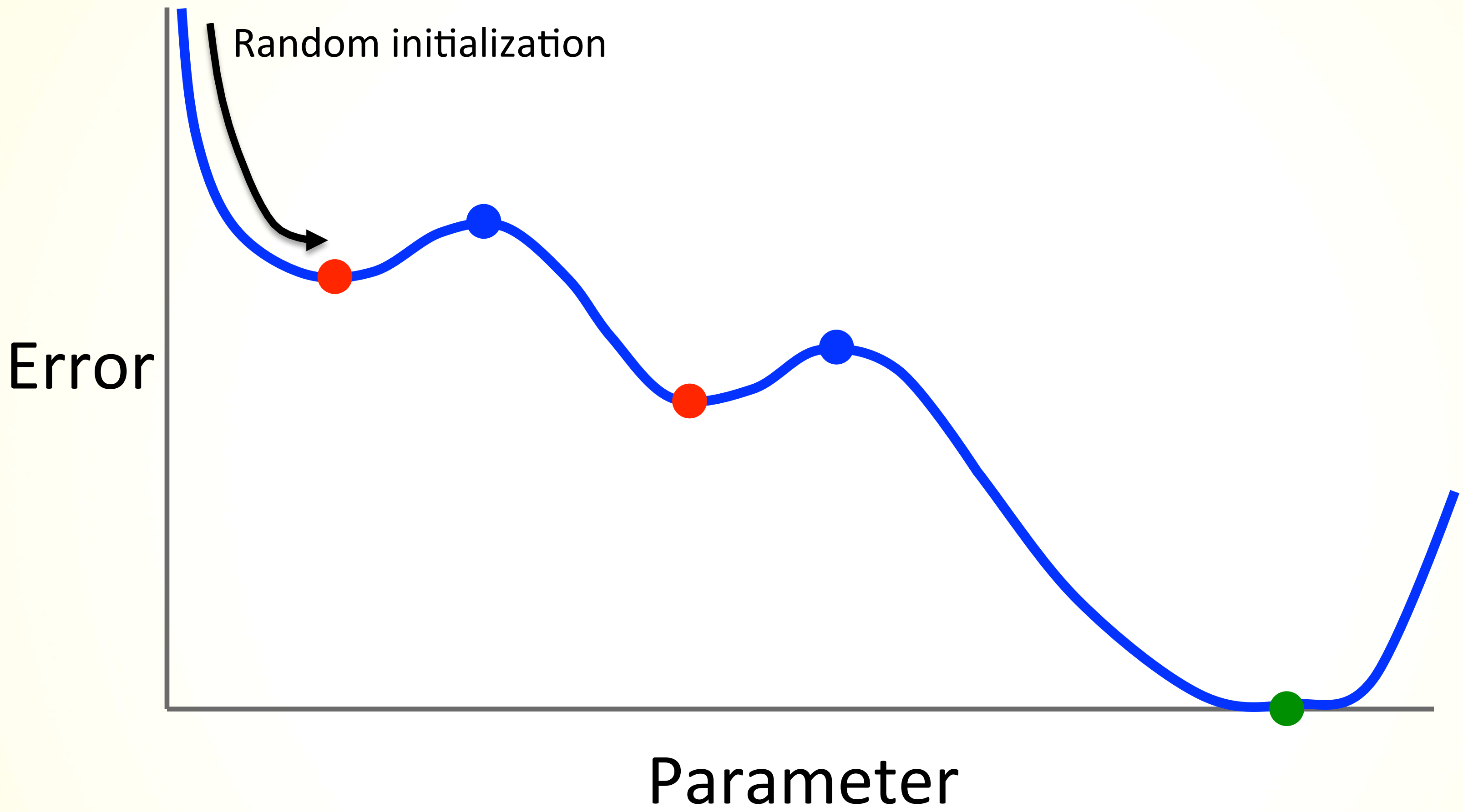
Revised conceptual picture

- Nonlinearities not the culprit
  - Naïve deep learning is slow even in the absence of
    - local minima
    - saturating nonlinearities
- Plateaus near saddle points are the culprit
  - Layer strengths close to zero, when multiplied together, are exponentially closer

Cartoon Error Surface



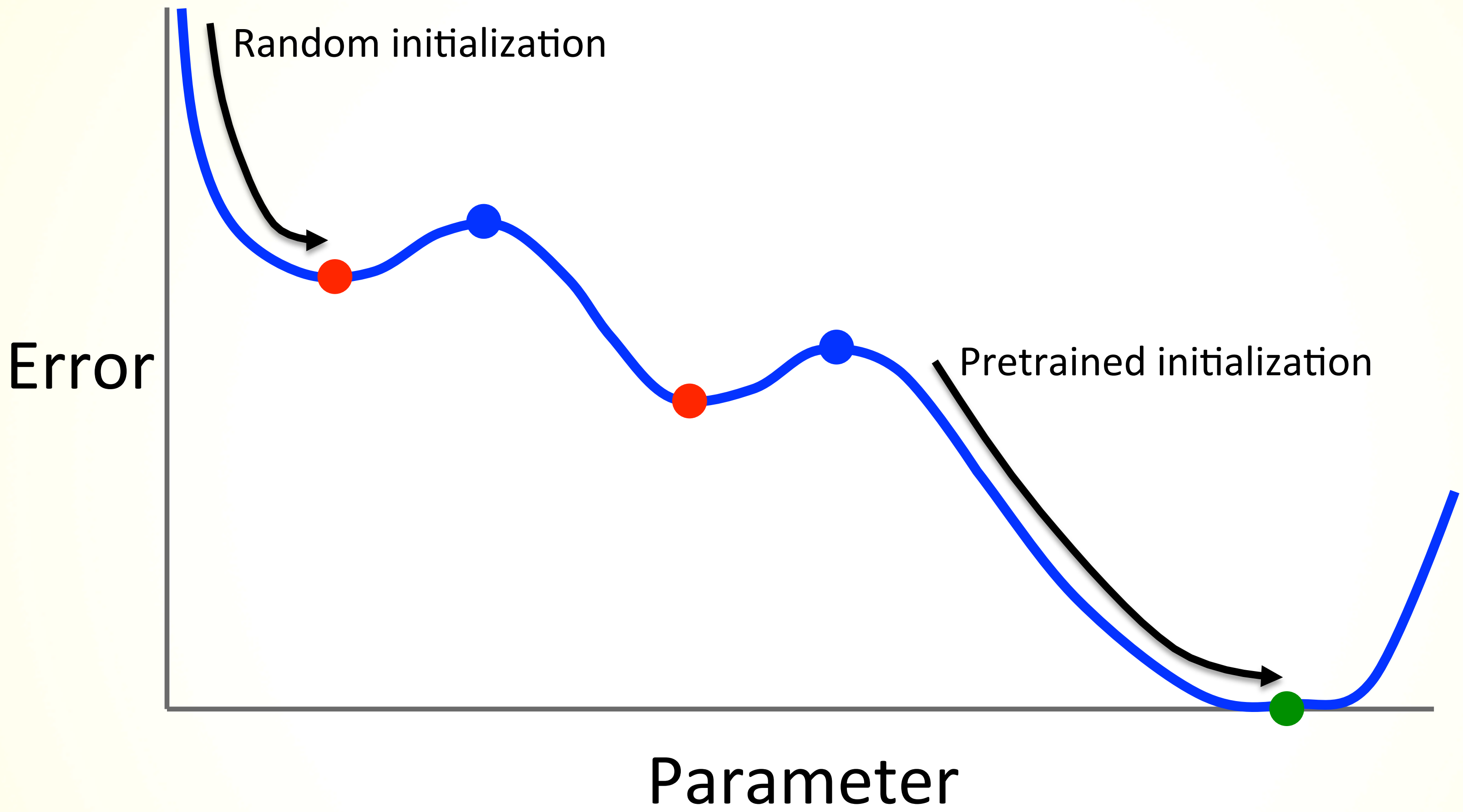
Cartoon Error Surface



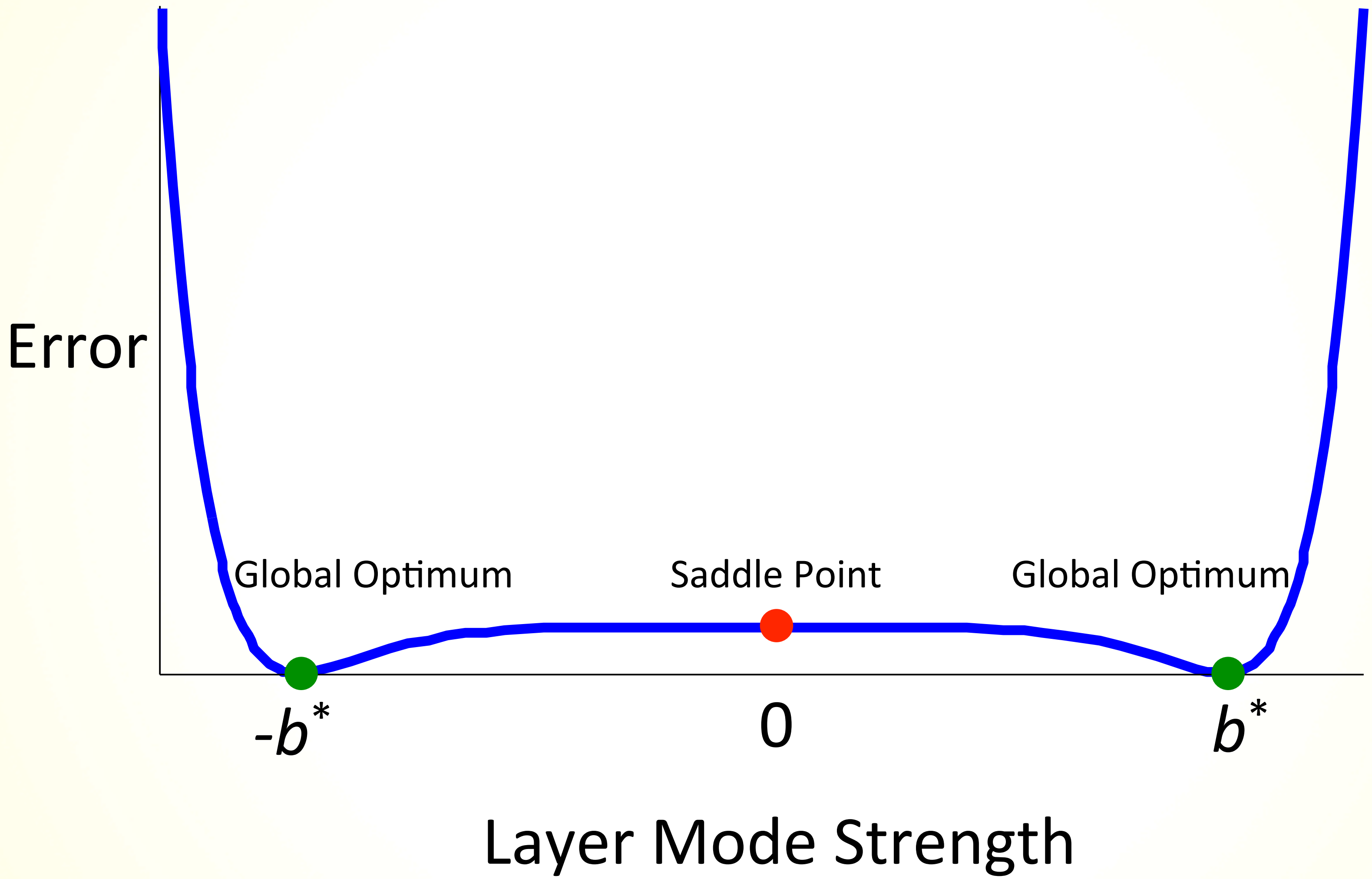
Previous Intuition

- Overwhelmingly likely to end in local minimum
- Unsupervised pretraining combats this by starting in good basin of attraction

Cartoon Error Surface



Actual Error Surface



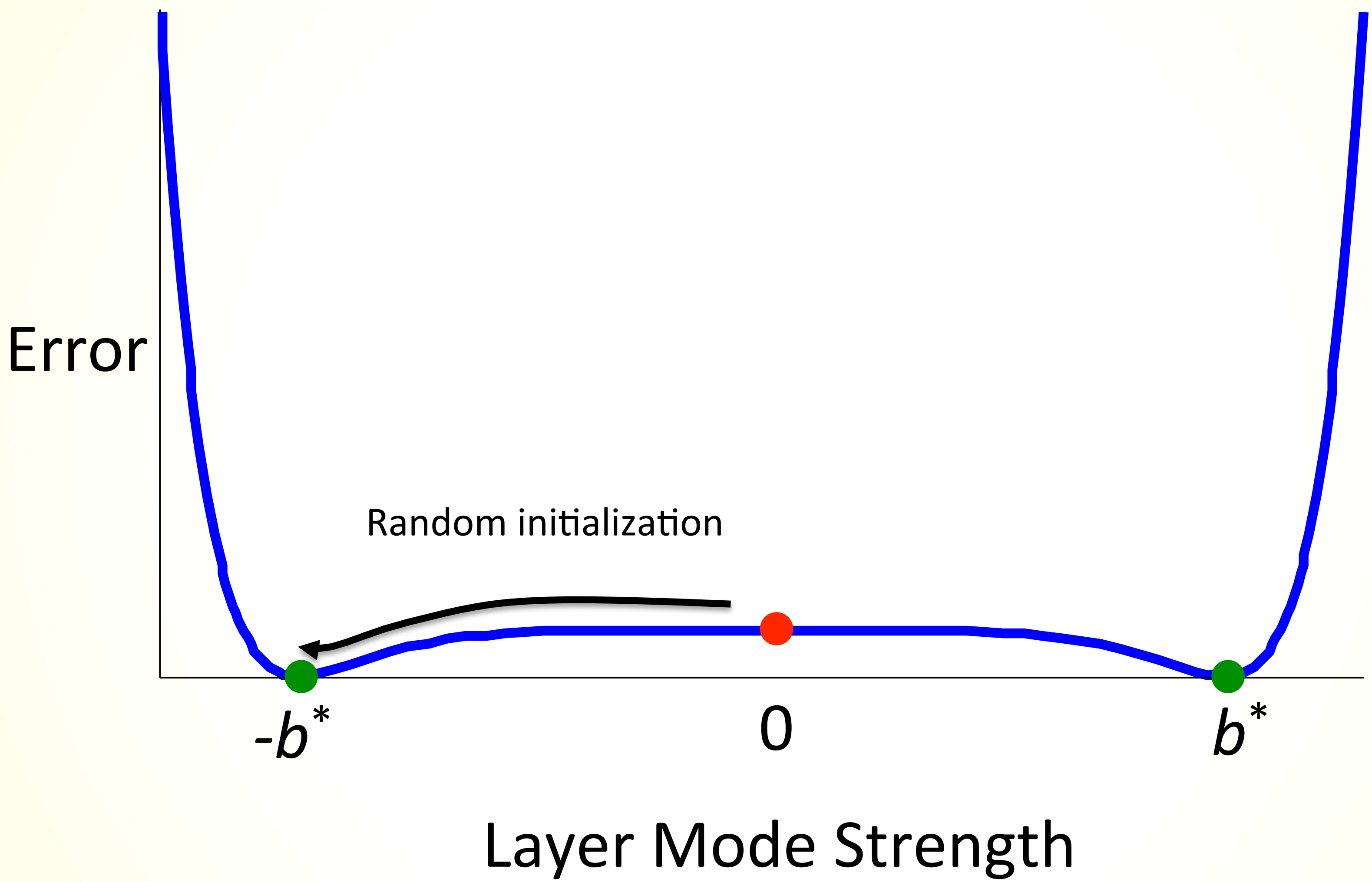
Actual Error Surface

- No local optima
- All minima are global minima
- (Baldi & Hornik, 1989)

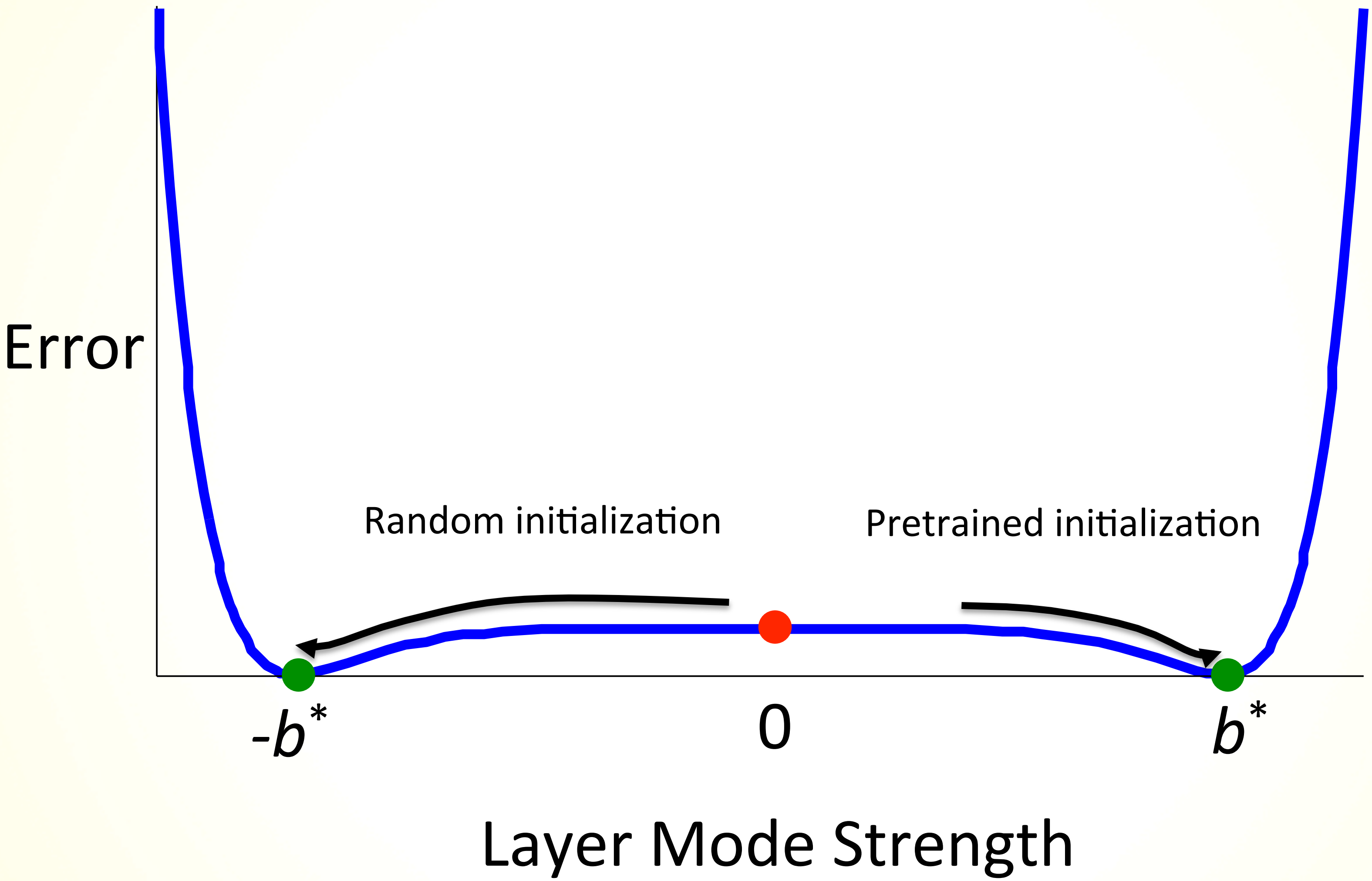
- Gets stuck on plateau near saddle point

- Unsupervised pretraining combats this by increasing initial scaling

Actual Error Surface



Actual Error Surface



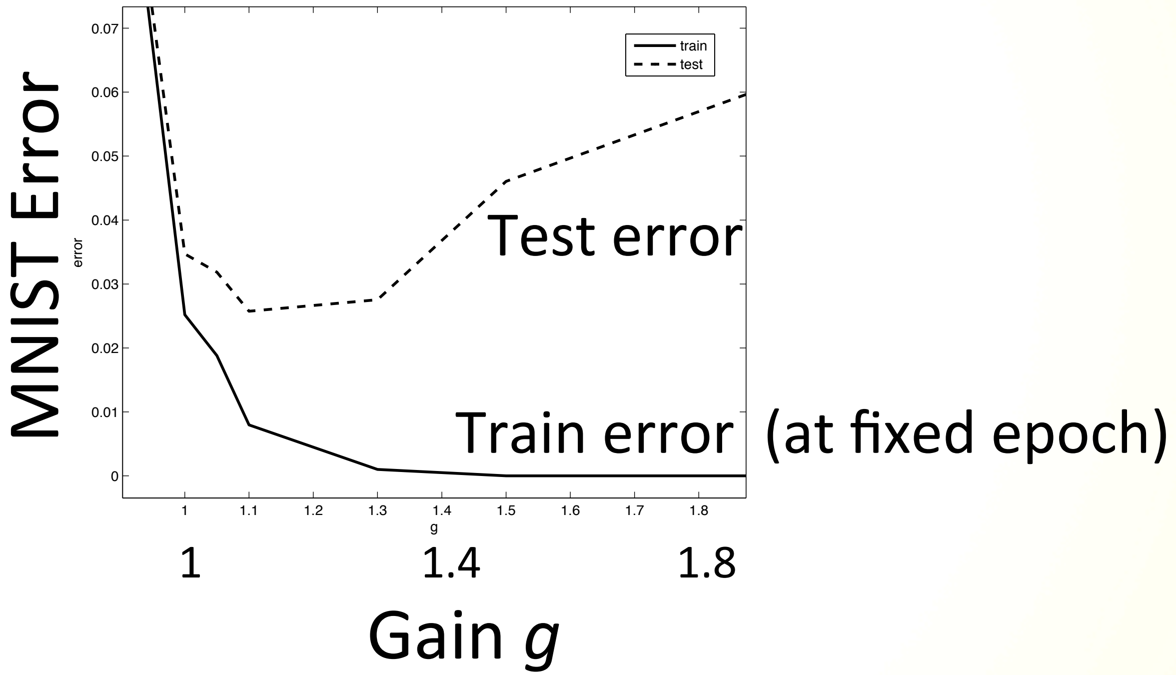
Nonlinear deep networks?

- Theory describes how deep linear networks behave

- Need to verify behavior in nonlinear nets

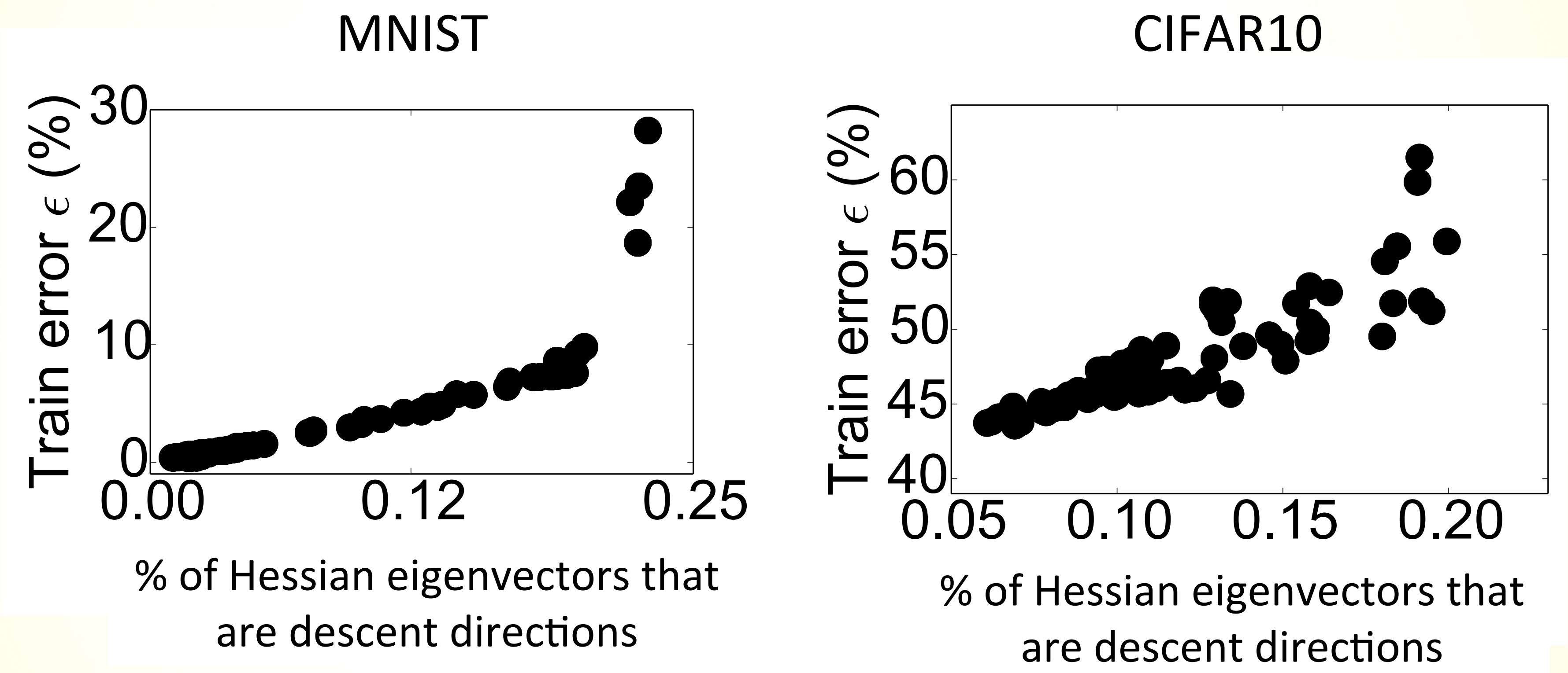
30 layer tanh networks

- Deep networks + large initializations train exceptionally quickly
- Can compute gain *g* necessary to overcome compressive nonlinearities



- These improved initializations have played a part in recent SOTA systems (He et al., 2015; van den Oord et al., 2015; Le et al., 2015).

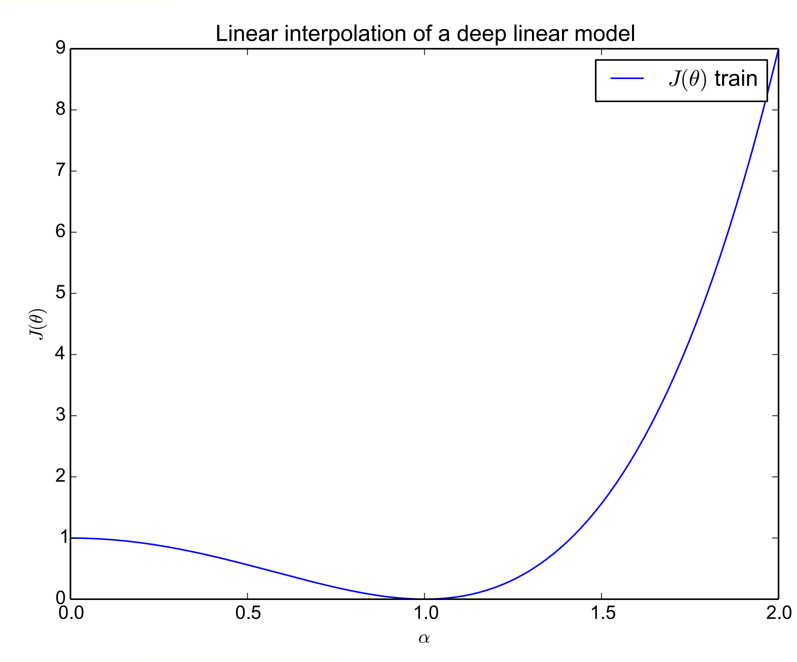
Few local minima, many saddle points



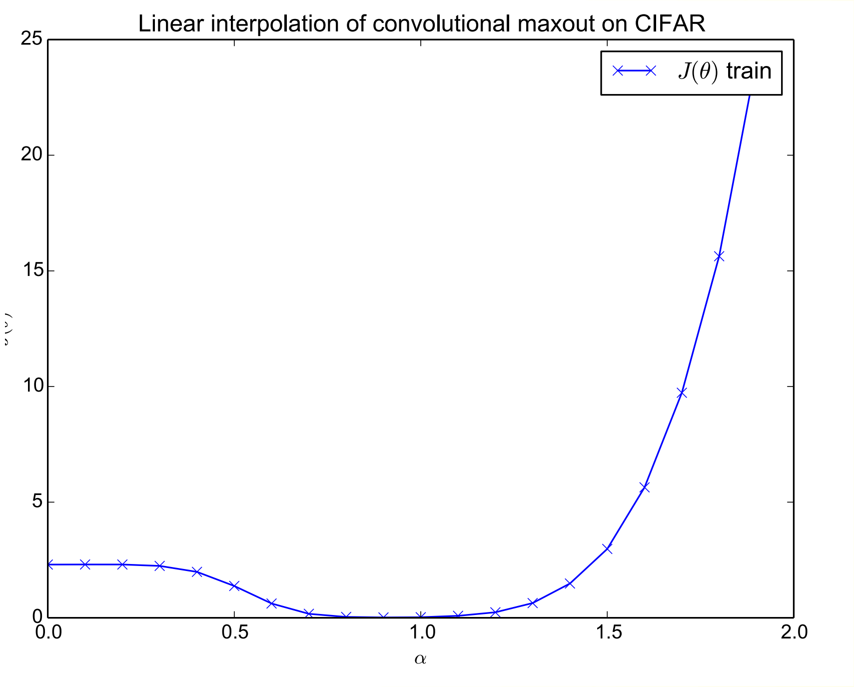
Dauphin et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization." Arxiv, 2014

Qualitatively similar error surface

Deep Linear Network



SOTA Conv. Maxout Network



Goodfellow, Vinyals, & Saxe, 2015

Summary of theory

– What is learned when?
  - Modes of the SVD learned in time 1/s

– How does learning speed scale with depth?
  - Direct training scales exponentially

$$t_{DT} \approx O\left(\frac{1}{b_0^{N_l}}\right)$$

  - Layerwise pretraining + fine-tuning scales linearly

$$t_{PT+FT} \approx O\left(N_l \log\left(\frac{1}{b_0^2 \varepsilon}\right)\right)$$

Outline

- Part 1: Theory of deep linear learning
- Part 2: Applications
  - Critical period plasticity
  - Perceptual learning
  - Semantic cognition
  - Perceptual decisions
  - Reinforcement learning

Intentional action

- “Every animal is, in some degree at least, a perceiver and a behaver.” JJ Gibson
- Deep learning models are largely perceptual
- What about action selection?

Deep learning for action selection?

- Key intuitions of deep learning approach don't hold in traditional control models
  - No compositionality
  - No layered, hierarchical structure
  - No model that supports distributed representations of tasks, goals, …
  - Discrete action spaces

Markov decision processes

A Markov decision process is one mathematical formulation of an optimal control problem. It is defined by four objects $(X, U, p(y|x,u), l(x,u))$

- X is the state space
- U is the action space
- $p(y|x,u)$ are the transition probabilities
- $l(x,u)$ is the immediate cost for being in state x and choosing action u

Our goal is to choose a policy $\pi(x)$ mapping states to actions that minimizes

$$v^{\pi}(x) = \mathop{\mathbf{E}}_{y_0 = x}\left[\sum_{\tau=0}^{t_f - 1} l(y_\tau, \pi(y_\tau))\right]$$

Optimal cost-to-go function

- The **optimal cost-to-go function** is the expected cumulative cost for starting at state x and acting optimally thereafter

- It encodes all relevant information about the future

- In particular, acting greedily with respect to the optimal cost-to-go function is perfectly optimal

Cost-to-go:

$$v^{\pi^*}(x) = \mathop{\mathbf{E}}_{y_0=x}\left[\sum_{\tau=0}^{t_f-1} l(y_\tau, \pi^*(y_\tau))\right]$$

Optimal action:

$$\pi^*(x) = \operatorname*{argmin}_{\pi} v^{\pi}(x)$$

Dynamic programming principle

- The dynamic programming principle is a statement about the cost-to-go function

- It says that the cost-to-go $v(x)$ for a state x is equal to the instantaneous cost for the optimal action plus the expected cost-to-go of the resulting next state

- This gives the famous Bellman equation

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y \sim p(\cdot|x,u)} \left[v(y) \right] \right\}$$

Problems?

- Discrete action space
- No compositionality
- No hierarchy
- Overly flexible cost function

Discrete action space

- Typically, at each time step choose one of M discrete actions

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y \sim p(\cdot|x,u)} \left[v(y) \right] \right\}$$

Very slow

- Curse of dimensionality

- (all possible joint angles for shoulder) X (all possible joint angles for elbow) X …

Discrete action space

- No notion of combining subactions to form a complete action

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y \sim p(\cdot|x,u)} \left[v(y) \right] \right\}$$

- E.g., muscle synergies

- Need distributed, combinatorial representation of actions

Compositional tasks

- No notion of combining subtasks to accomplish a new task

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y\sim p(\cdot|x,u)}\left[v(y)\right] \right\}$$

Hierarchy

- No notion of hierarchy

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y \sim p(\cdot|x,u)} \left[v(y) \right] \right\}$$

- Options are hierarchical, but only slightly

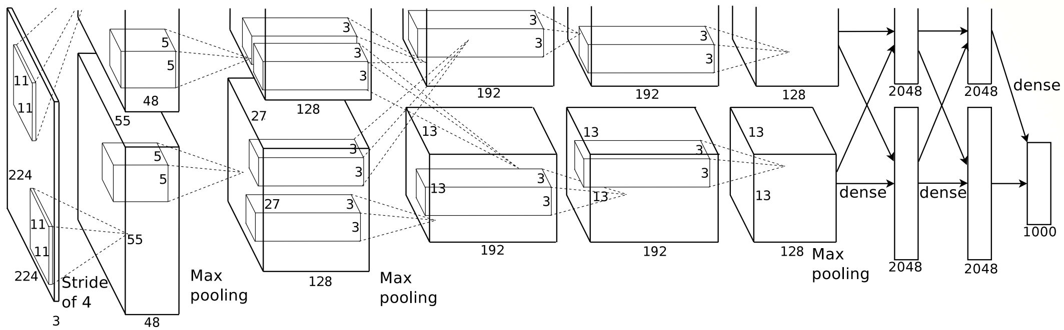
Overflexible cost functions

- Problem formulation might be *too* general

$$v(x) = \min_u \left\{ l(x,u) + \mathbf{E}_{y \sim p(\cdot|x,u)} \left[v(y) \right] \right\}$$

- We usually take energetically efficient action

SOTA Example: Atari player



Action A
Action B
…
Action F

Input

Deep network

Value function

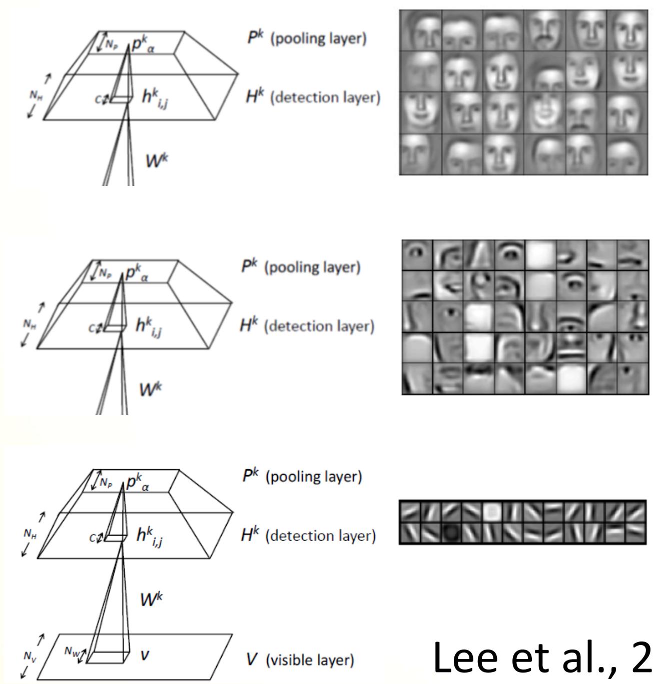
- Deep network predicts ultimate future reward accrued from taking each action
- 10,000,000 examples, 160,000,000 presentations
- Works extremely well (often better than human!)

- Change any detail of the task (shooting bad guy now worth 2 points not 1), have to substantially retrain

Wanted:
Composable action selection unit

- The RBM of action selection

"Forms within forms"



Lee et al., 2009

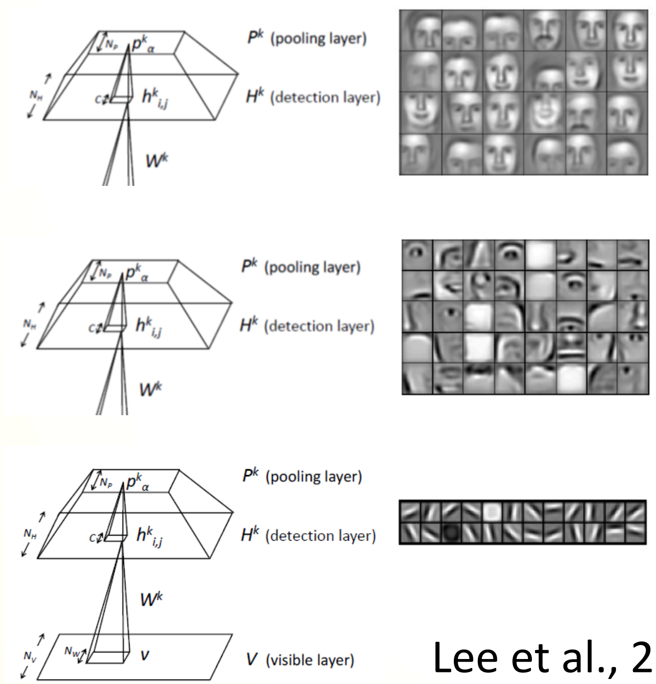
"Acts within acts"

?

Wanted: Composable action selection unit

- The RBM of action selection

"Forms within forms"



Lee et al., 2009

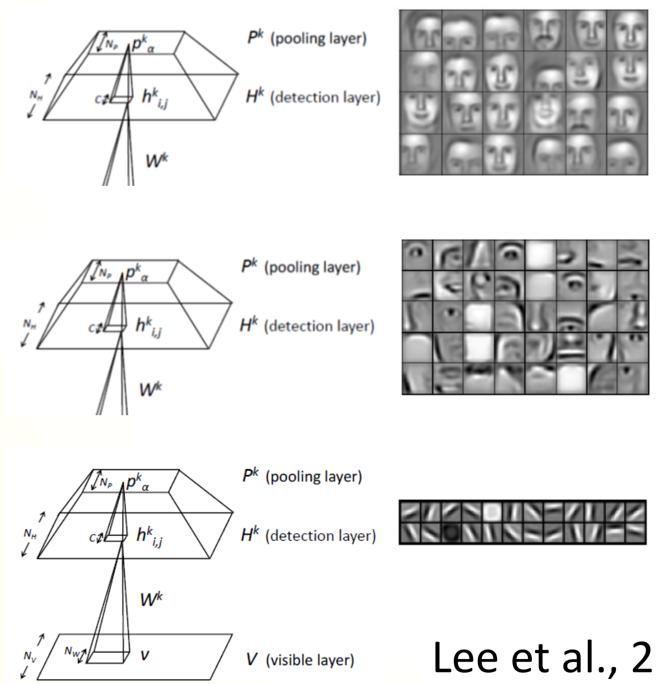
"Acts within acts"

- Graded, non-discrete action space
- Distributed representation of desires/wants
- Blend previously learned information to do novel tasks
- Do action selection, goal inference, and social causal learning
- Nested acts within acts

Wanted:
Composable action selection unit

- The RBM of action selection

"Forms within forms"



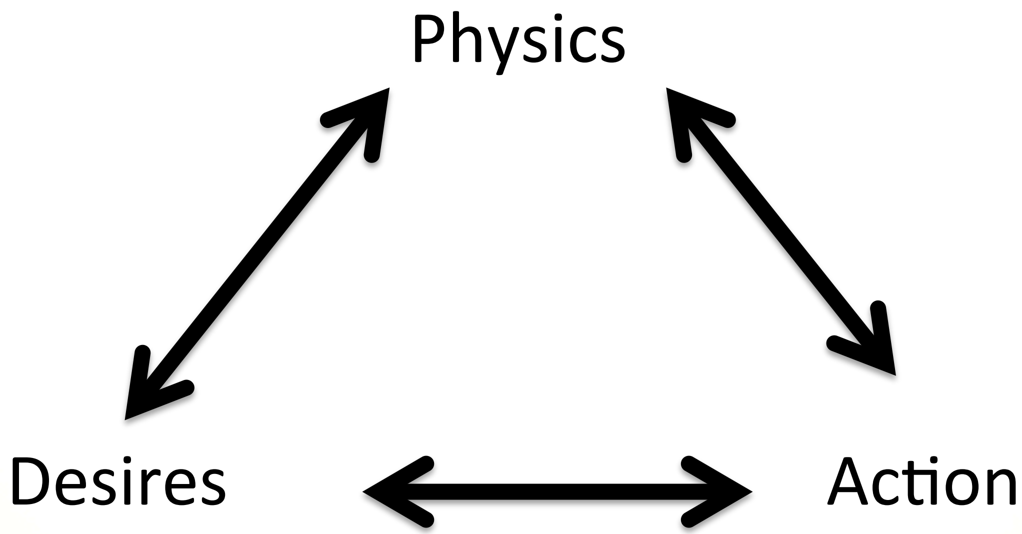
Lee et al., 2009

"Acts within acts"

- **Graded, non-discrete action space**
- **Distributed representation of desires/wants**
- **Blend previously learned information to do novel tasks**
- **Do action selection, goal inference, and social causal learning**
- Nested acts within acts

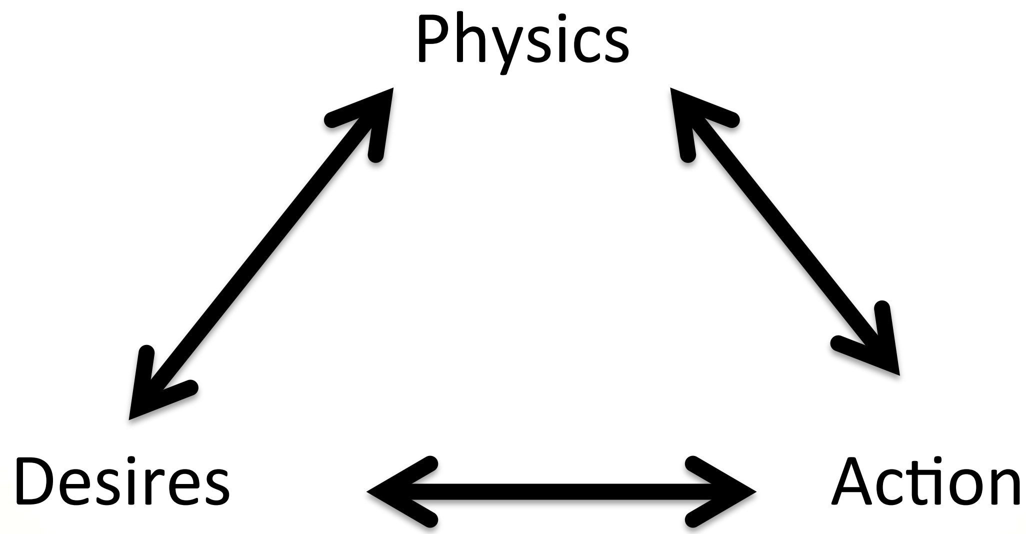
The basic model: Multitask z-learner

- Instantiates three elements



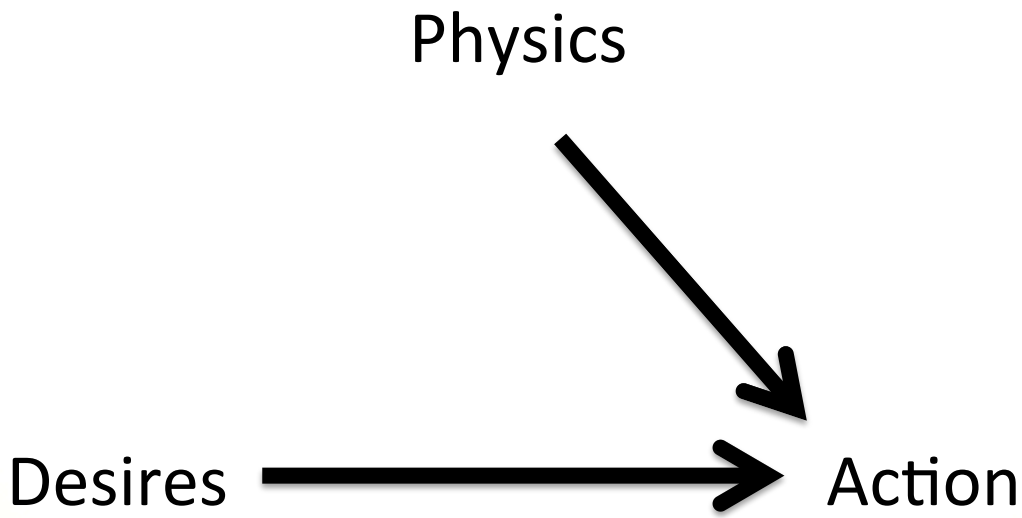
The basic model: Multitask z-learner

- Given any two, infer third



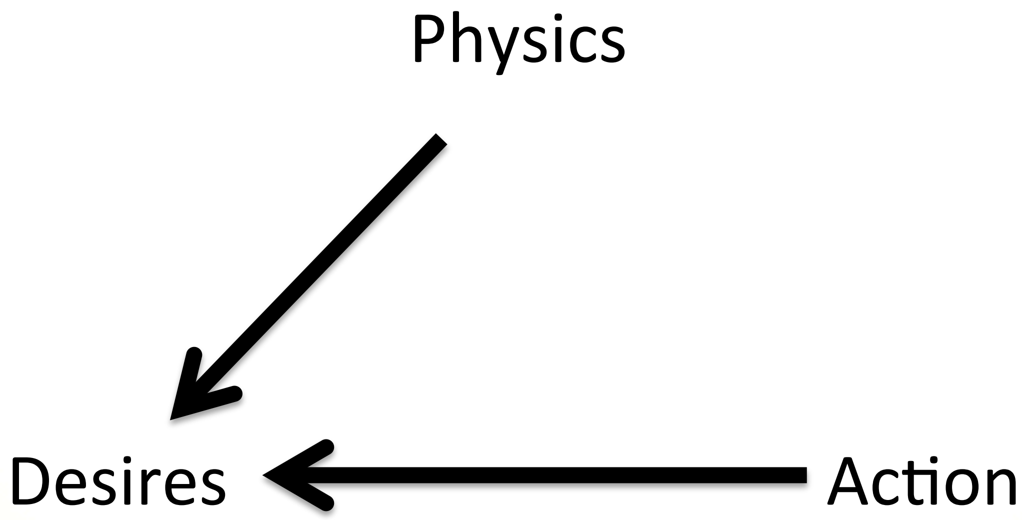
The basic model: Multitask z-learner

- Reinforcement learning



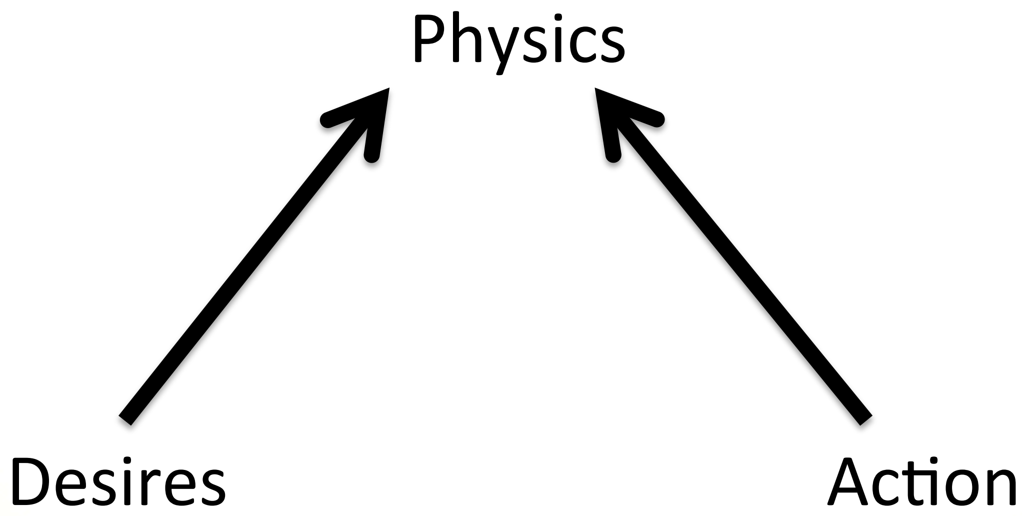
The basic model: Multitask z-learner

- Goal inference/inverse reinforcement learning



The basic model: Multitask z-learner

- Social causal learning



The basic model: Multitask z-learner

$$s_t \xrightarrow{P} s_{t+1}$$

Physics

r_t

Desires

u_t

Action

Physics (causal world structure)

P is *transition matrix*

$$s_t \xrightarrow{P} s_{t+1}$$

One-hot vector One-hot vector

Desires/goals/wants

$$r_t$$

Vector of:
instantaneous rewards expected for reaching each state

Action

$$u_t$$

Probability distribution over the next state, s_{t+1}

Action

$$u_t$$

Probability distribution over the next state, s_{t+1}

- Initially may seem odd:
  - if you specify transition probabilities directly, just jump to highest reward state!

- Totally graded notion of actions. Just bias yourself a little more toward the states you want, and away from those you don't.

Intentional actions:
Balancing reward seeking with effort

Main innovation (Todorov, 2009):

Achieve your desires

Minimize exertion

Intentional action

Intentional actions: Balancing reward seeking with effort

Main innovation (Todorov, 2009):

Maximize sum of rewards   minimize deviation from physics

Intentional action

Intentional actions: Balancing reward seeking with effort

Main innovation (Todorov, 2009):

Maximize sum of rewards minimize $KL(u_t \,||\, Ps_t)$

Intentional action

Optimal actions

$$u_t^* = \underset{u_t}{\operatorname{argmax}} \sum_t r_t^T s_t - KL(u_t \,\|\, P s_t)$$

Can analytically compute this

For LMDPs, optimal action directly computable from cost-to-go function *v(x)*. Define exponentiated cost-to-go (desireability) function: $z(x) = \exp(-v(x))$

Bellman equation *linear* in *z*:

$$z(x) = \exp(-q(x))\mathbf{E}_{y \sim p(\cdot|x)}\left[z(y)\right]$$

Or $z_i = Mz_i + n_b$ where z_i encodes desireability of interior states

Crucial property: Solutions for two different boundary reward structures linearly compose (Todorov, 2009)

$$\tilde{q}_b^{1+2} = a\tilde{q}_b^1 + b\tilde{q}_b^2 \Rightarrow z_i^{1+2} = az_i^1 + bz_i^2$$

Multitask Z-learning: Learn about a set of boundary reward structures $\tilde{q}_b^c, c = 1, \ldots, m$

- represent any new task as a linear combination of these
- optimal *z(x)* is linear combination of component tasks' $z^c(x)$

Compositionality restored!

- Learn about *N* tasks

- Can weight these *N* tasks together to perform *infinite* variety of composite tasks

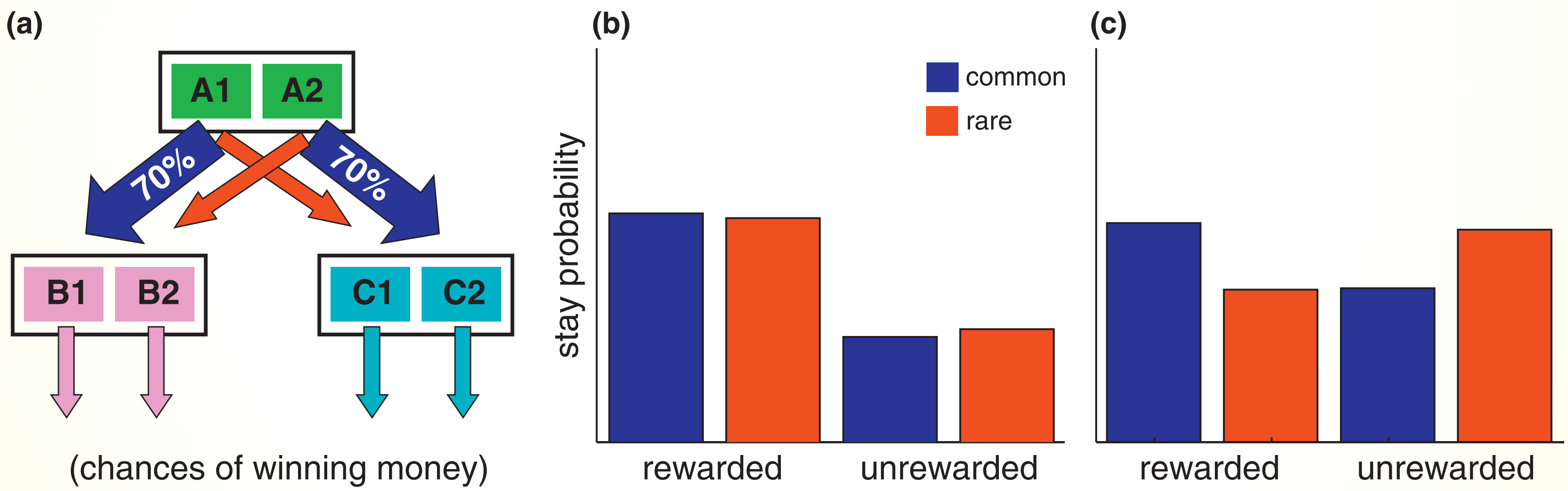
- Examples coming…

Boundary states

- Only get compositionality at boundary states

Outcome revaluation in sequential choice

- Humans and animals can rapidly adapt to changing rewards in sequential choices (Daw et al., 2011)

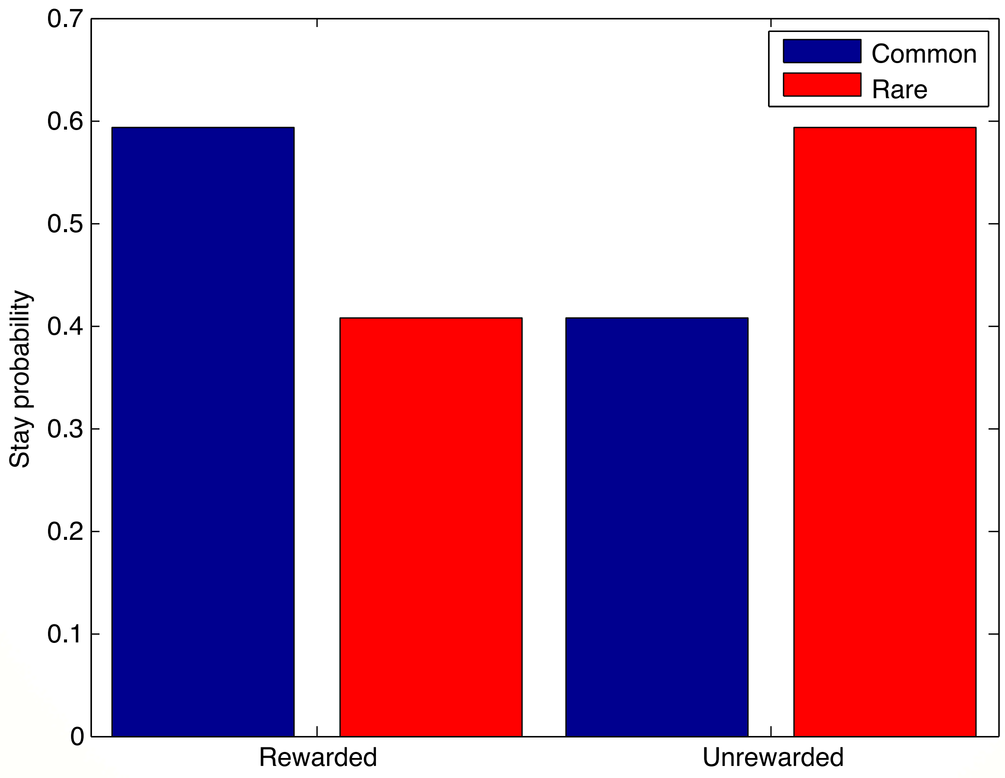


Model-free

"Model-based"

Fig 1, Doll et al., 2012

Multitask Z:
Instant outcome revaluation

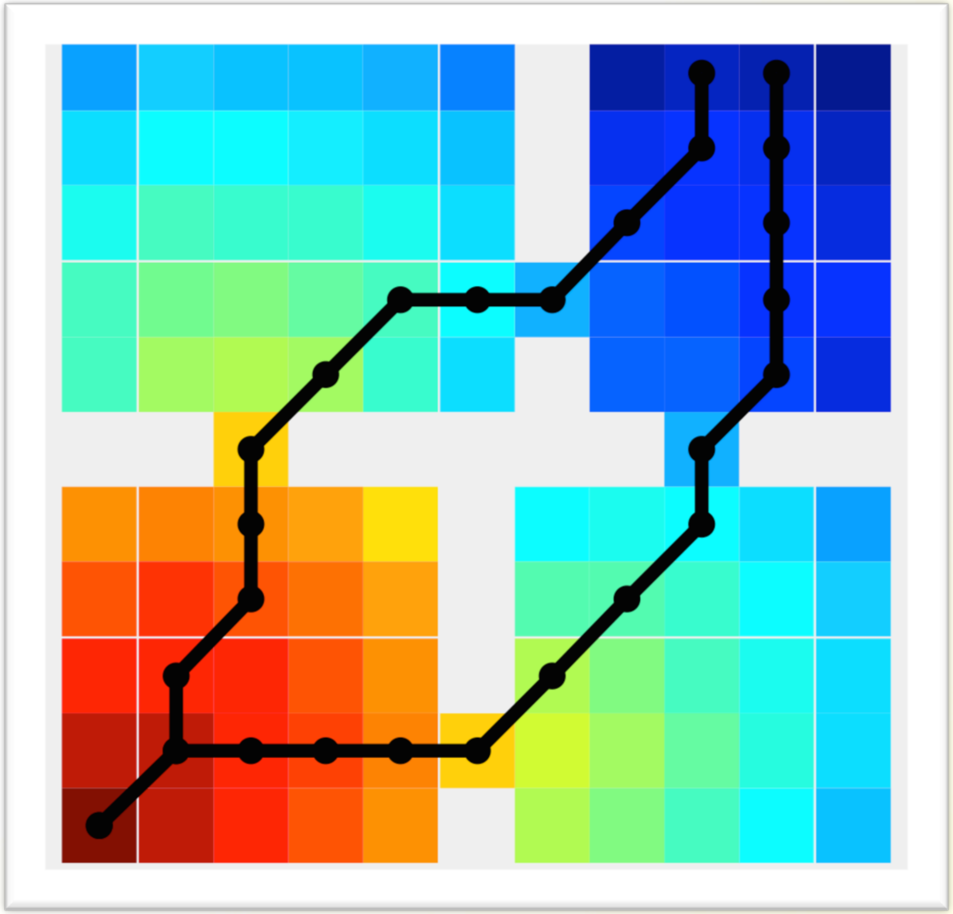


- Behaves like model-based methods but no forward search

Latent learning in spatial navigation

After random exploration of a maze environment, introduction of a reward at one location leads to instant goal-directed behavior towards that point (Tolman, 1948)

- Covert multitask z-learning during exploration enables immediate navigation to rewarded locations when reward structure becomes known

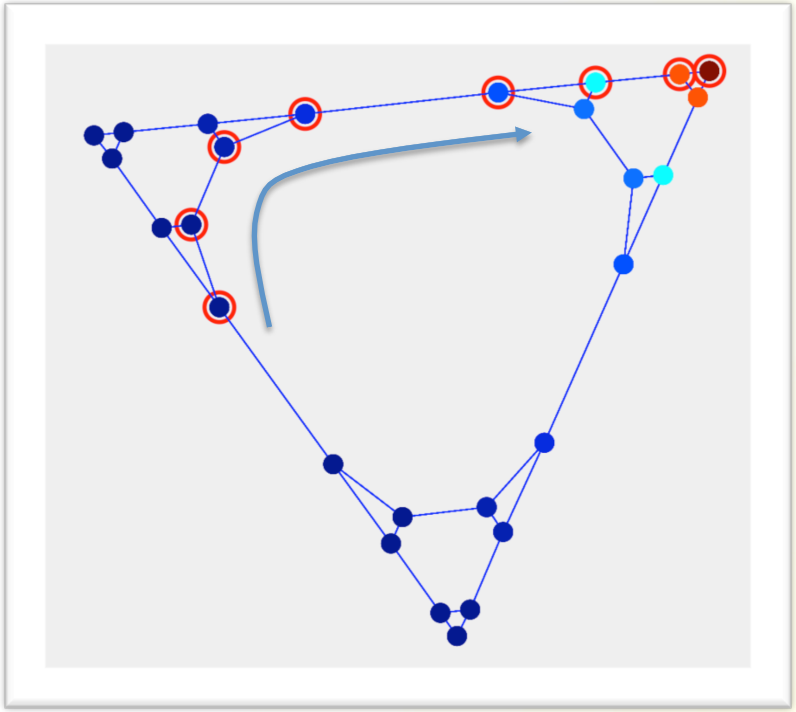


Tower of Hanoi

Applicable to goal-directed action in more complex domains (Diuk et al., 2013)

- Move blocks to peg 3; smaller blocks must always be stacked on larger blocks

- After exploration, multitask Z-learning is capable of navigating to arbitrary configurations

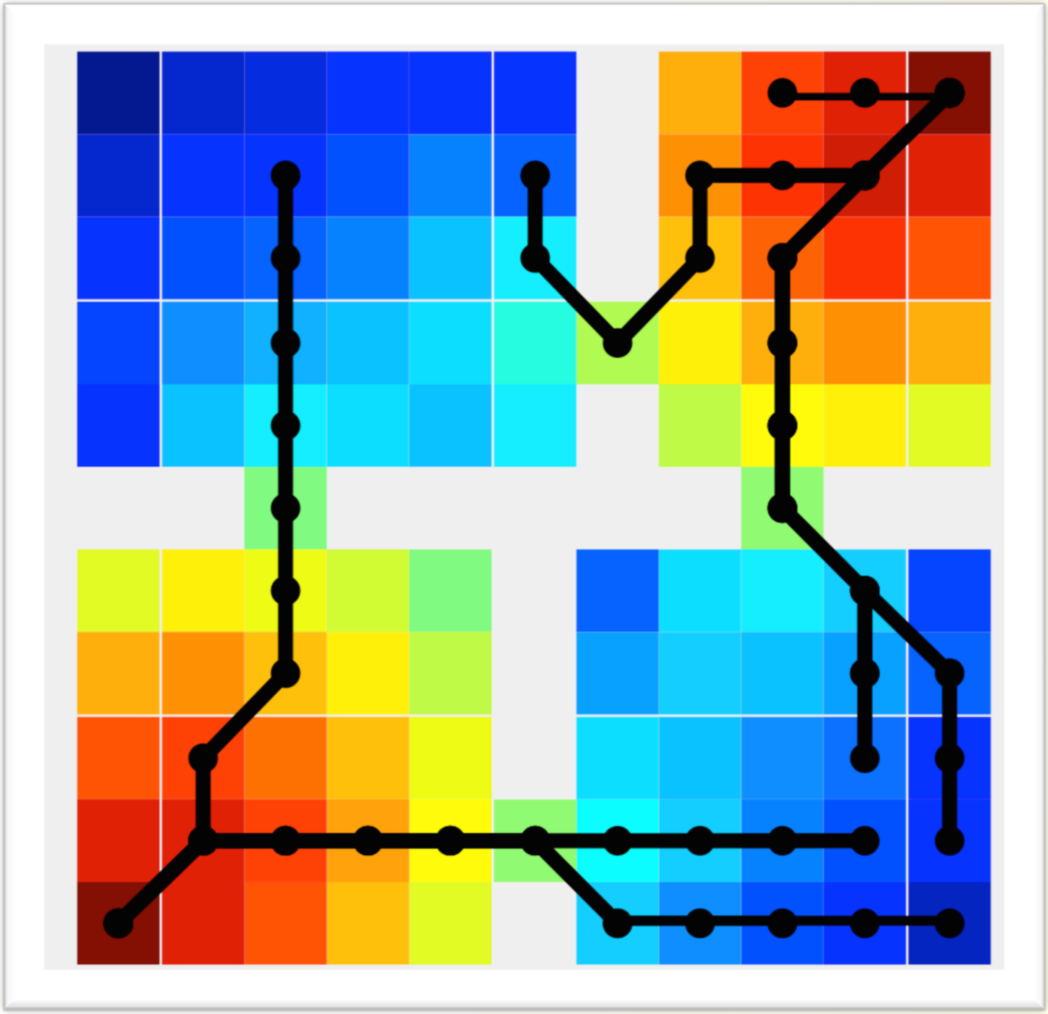


State graph with cost-to-go and optimal trajectory

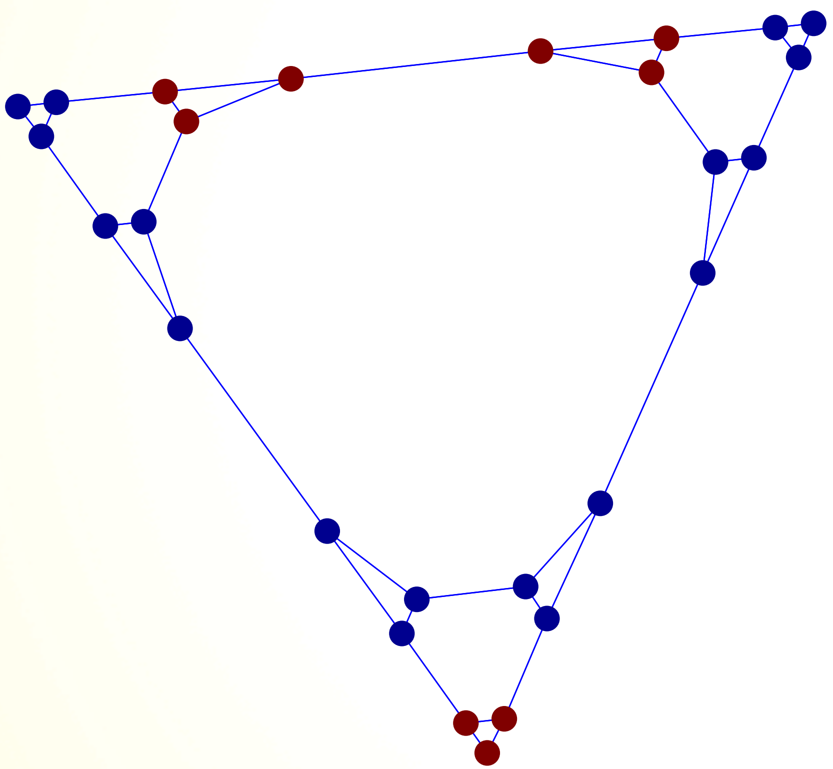
Exploiting compositionality: "Navigate to room A or B"

Can respond flexibly to a variety of navigation tasks

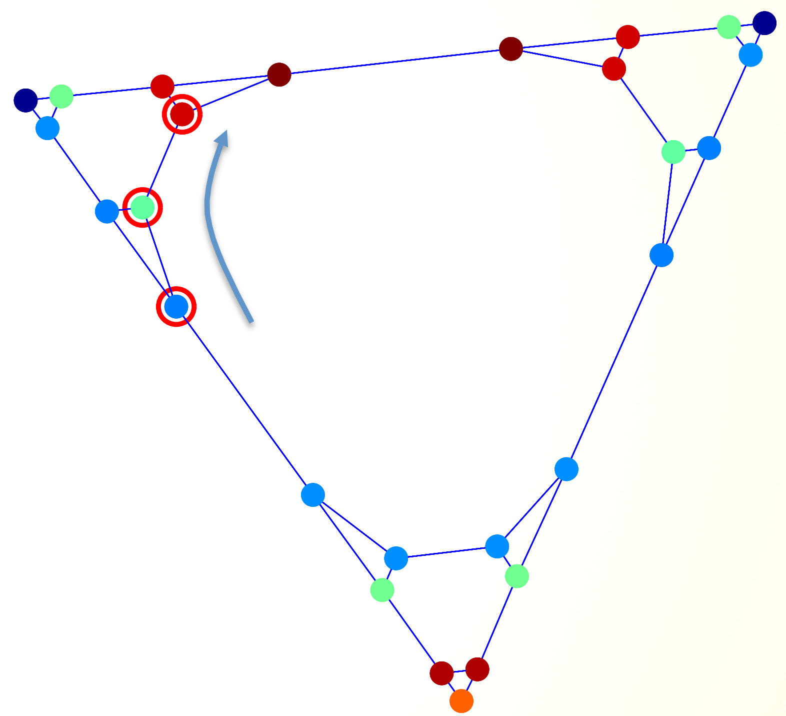
- Find food or water (specific satiety experiments)
- Go to a point, while avoiding door #2
- Important note: Not the same as planning through arbitrary cost map because of boundary state formulation.



“Place medium-size block on middle peg”



Instantaneous rewards



Cost-to-go/trajectory

Exploiting compositionality

Compositionality enables rapid response to *novel* complex queries

- Stack small block on large block
- Place medium block on peg 1, small block on peg 3

- Models highly practiced expert quite familiar with domain
- Can be combined with model-based search

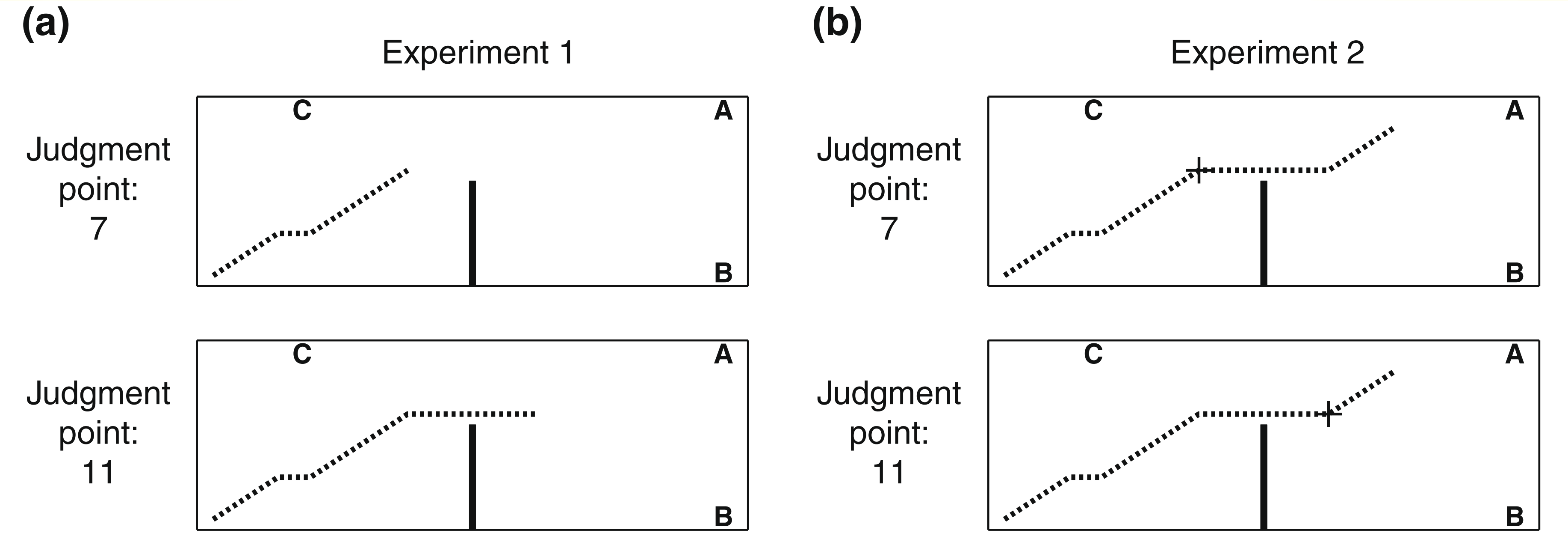
Multitask z-learning for action selection

- New algorithm with interesting properties:
  - **Instantaneous optimal** adaptation to new terminal state rewards
  - Relies on careful problem formulation to permit compositionality
  - Off-policy algorithm over states (not state/action pairs)
  - Compatible with function approximation

- Compatible with model-based & model-free accounts, which are tractable in the LMDP

Inferring goals/wants/desires

- “Dogs are the sort of agents that like bones” –Tenenbaum



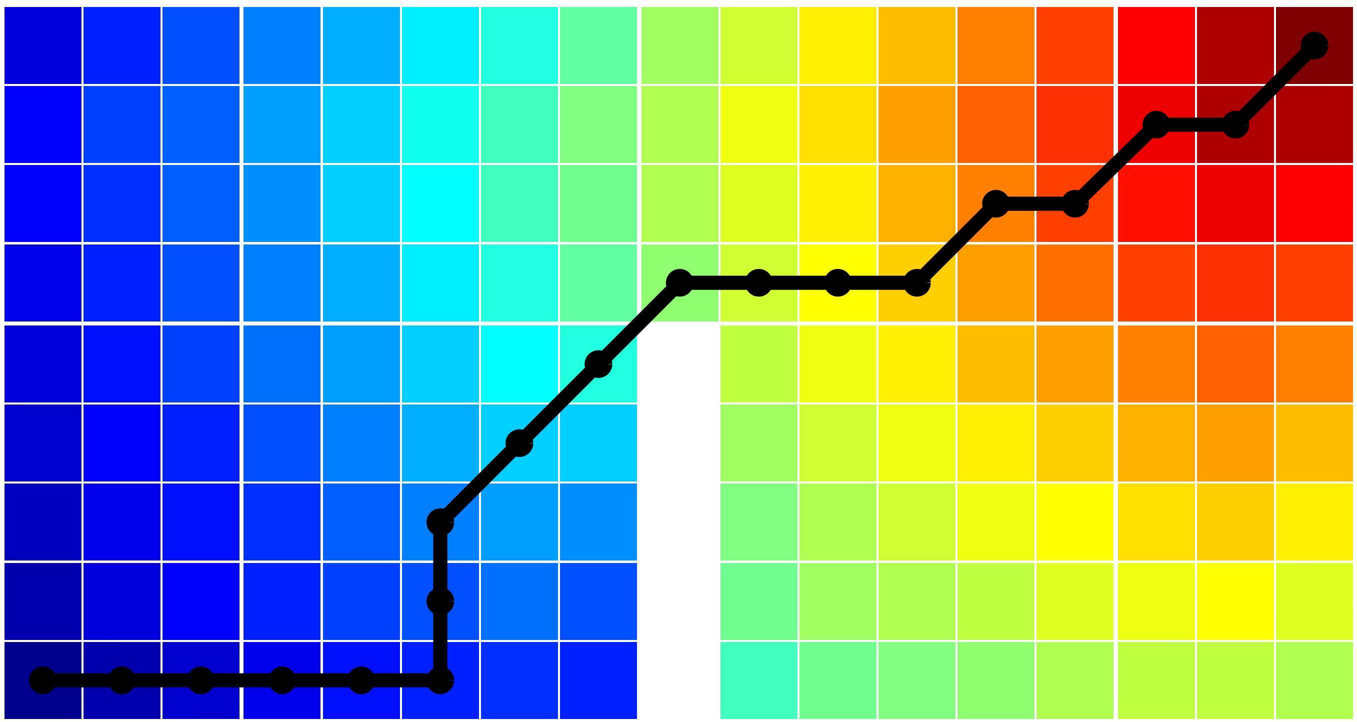
Baker, Saxe, & Tenenbaum, 2009

Inferring goals/wants/desires

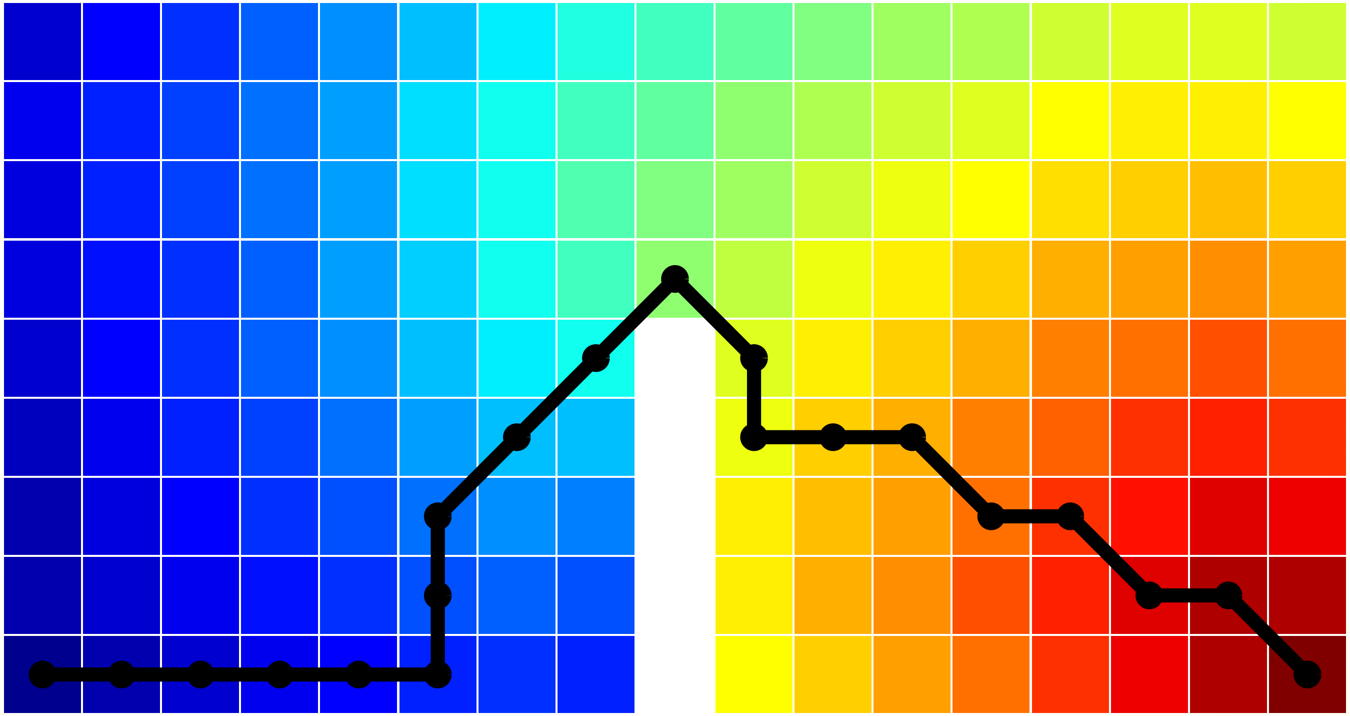
- Corresponds to *inverse* reinforcement learning (Ng & Russell, 2000; Dvijotham & Todorov, 2010)

- Observe P and a trajectory resulting from u_t
- Infer r_t

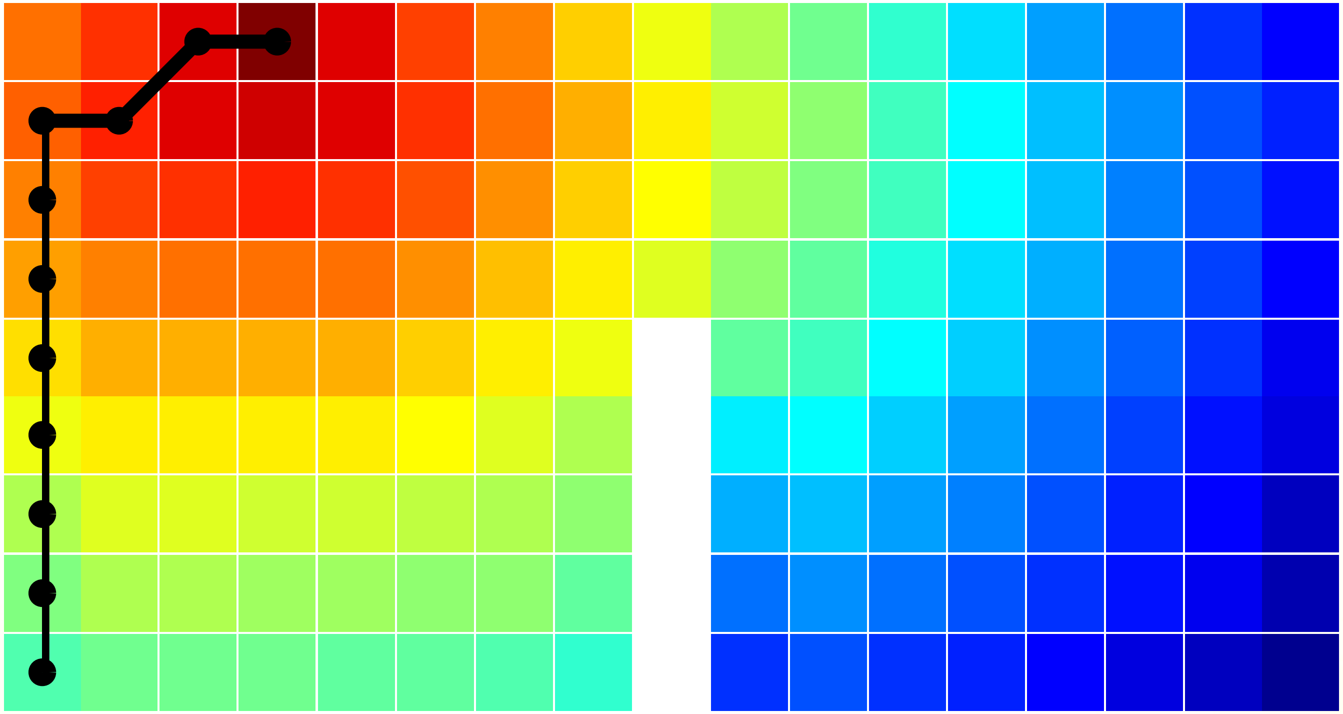
Goal A



Goal B

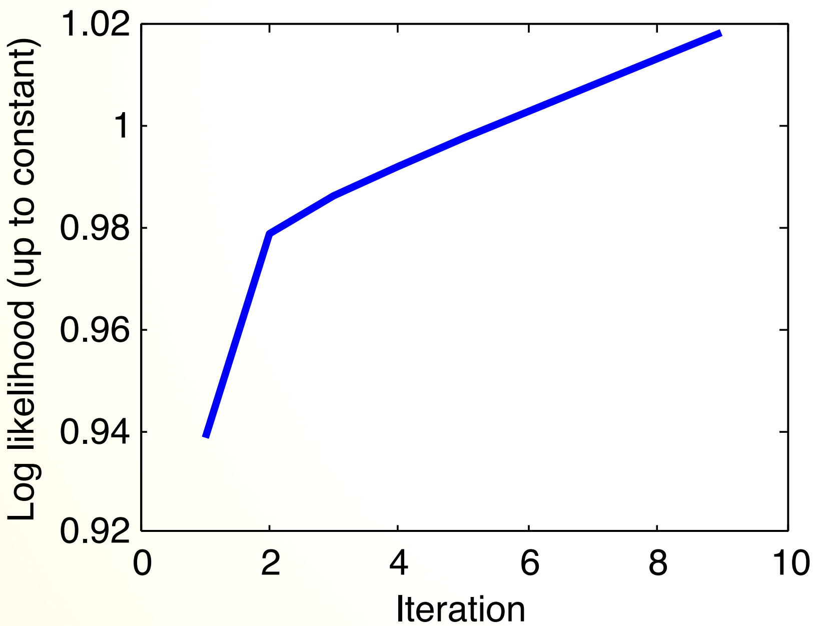


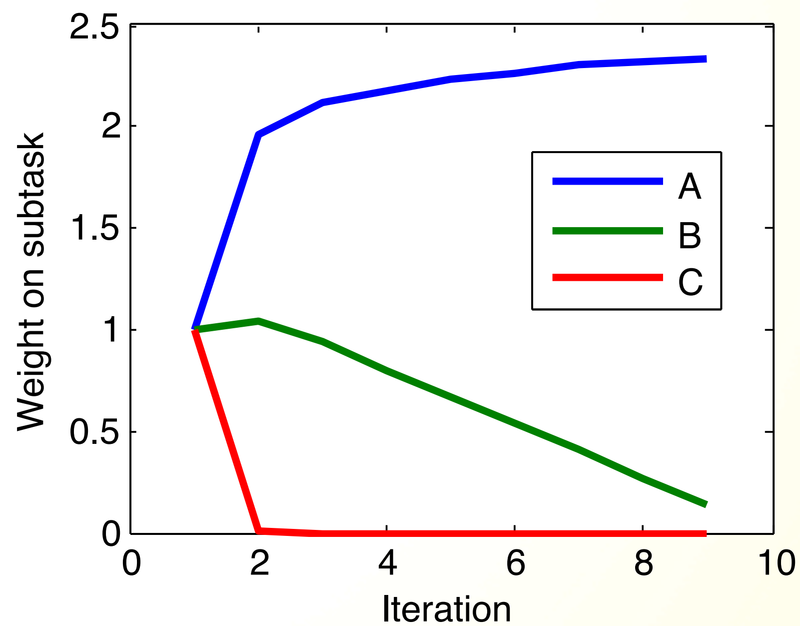
Goal C



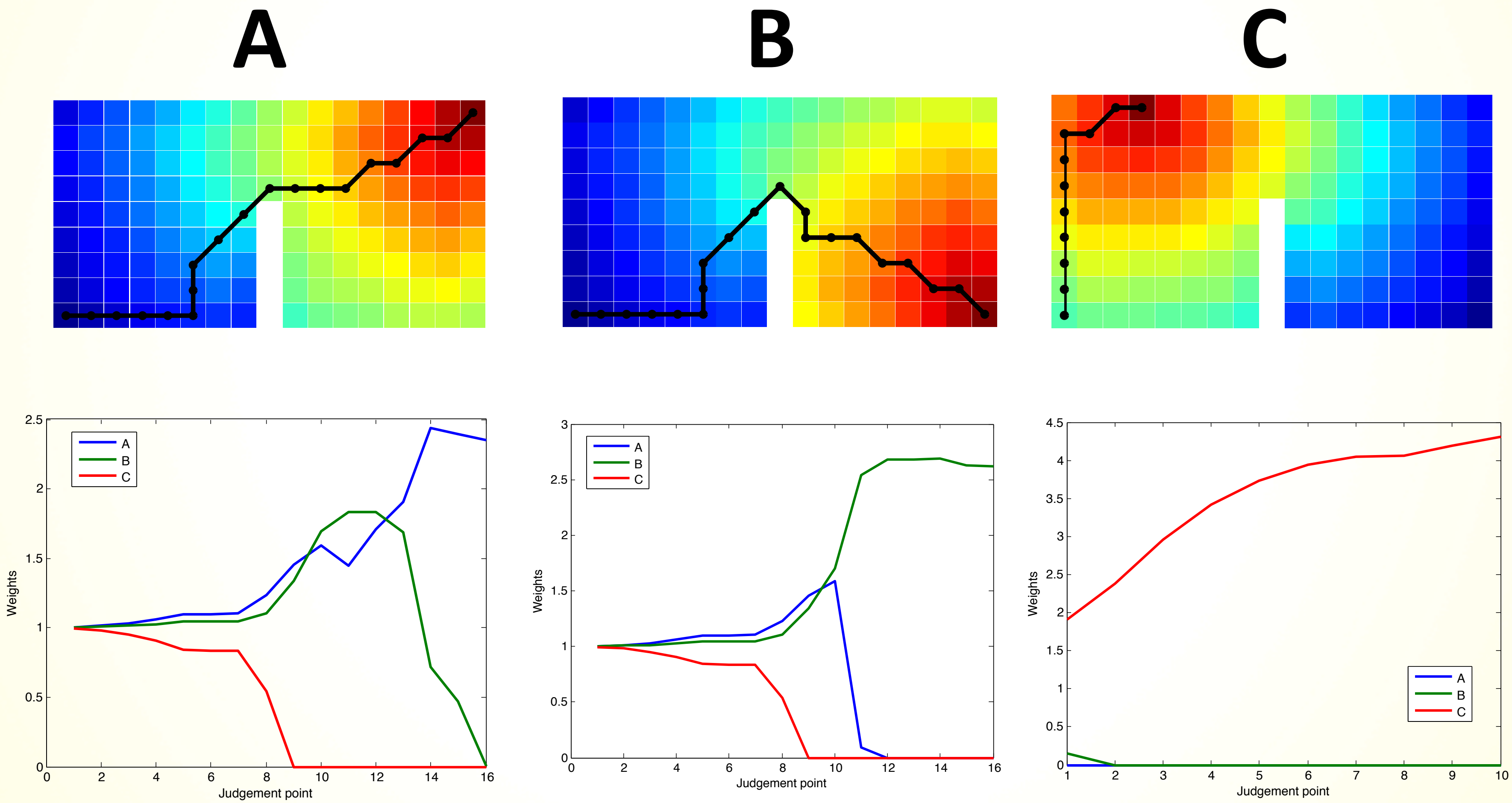
Inference process

- Maximize Log Likelihood of task combination weighting





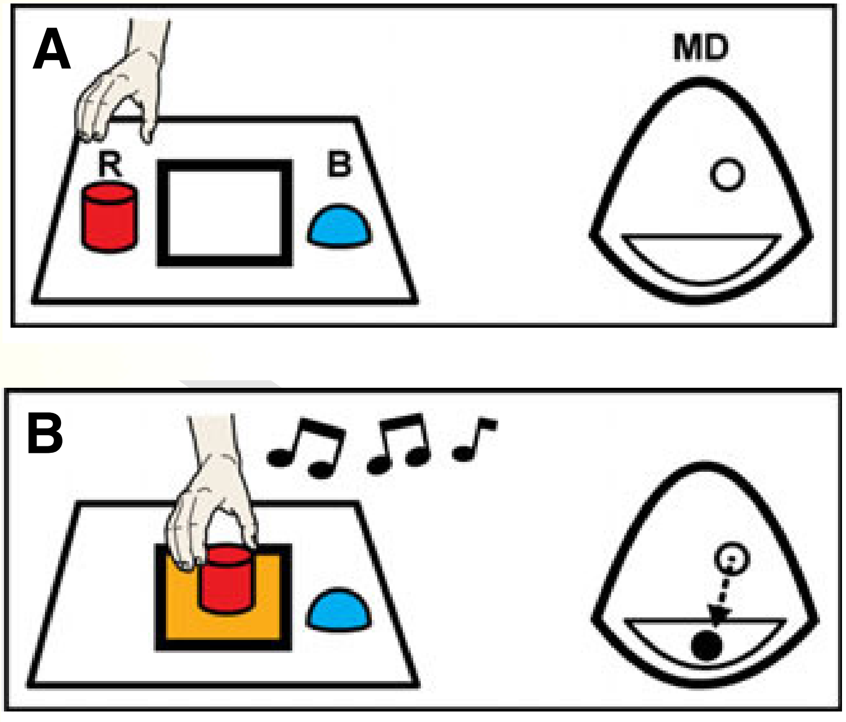
Goal inference

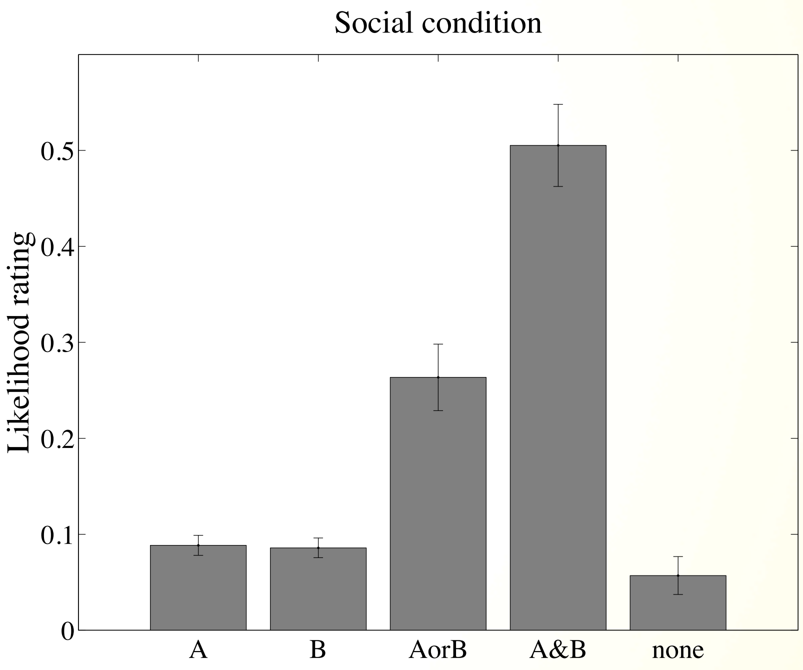


Goal inference

- From actions and physics, can infer goals
- Lots left to be done
  - Hierarchically structured actions
  - Changing goals

Social causal learning





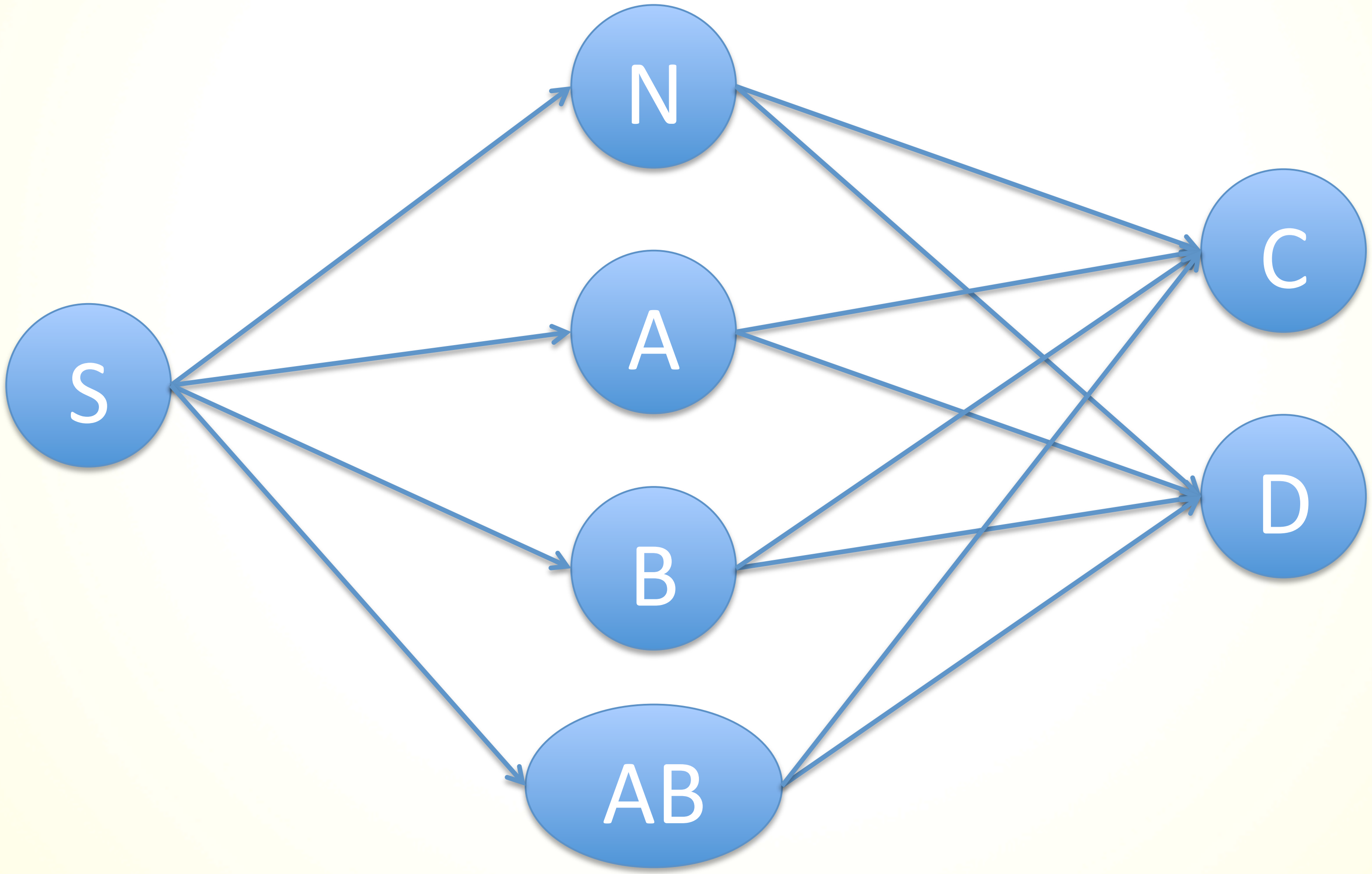
Waismeyer, Meltzoff, & Gopnik, 2014; Goodman, Baker, & Tenenbaum, 2009

Social causal learning

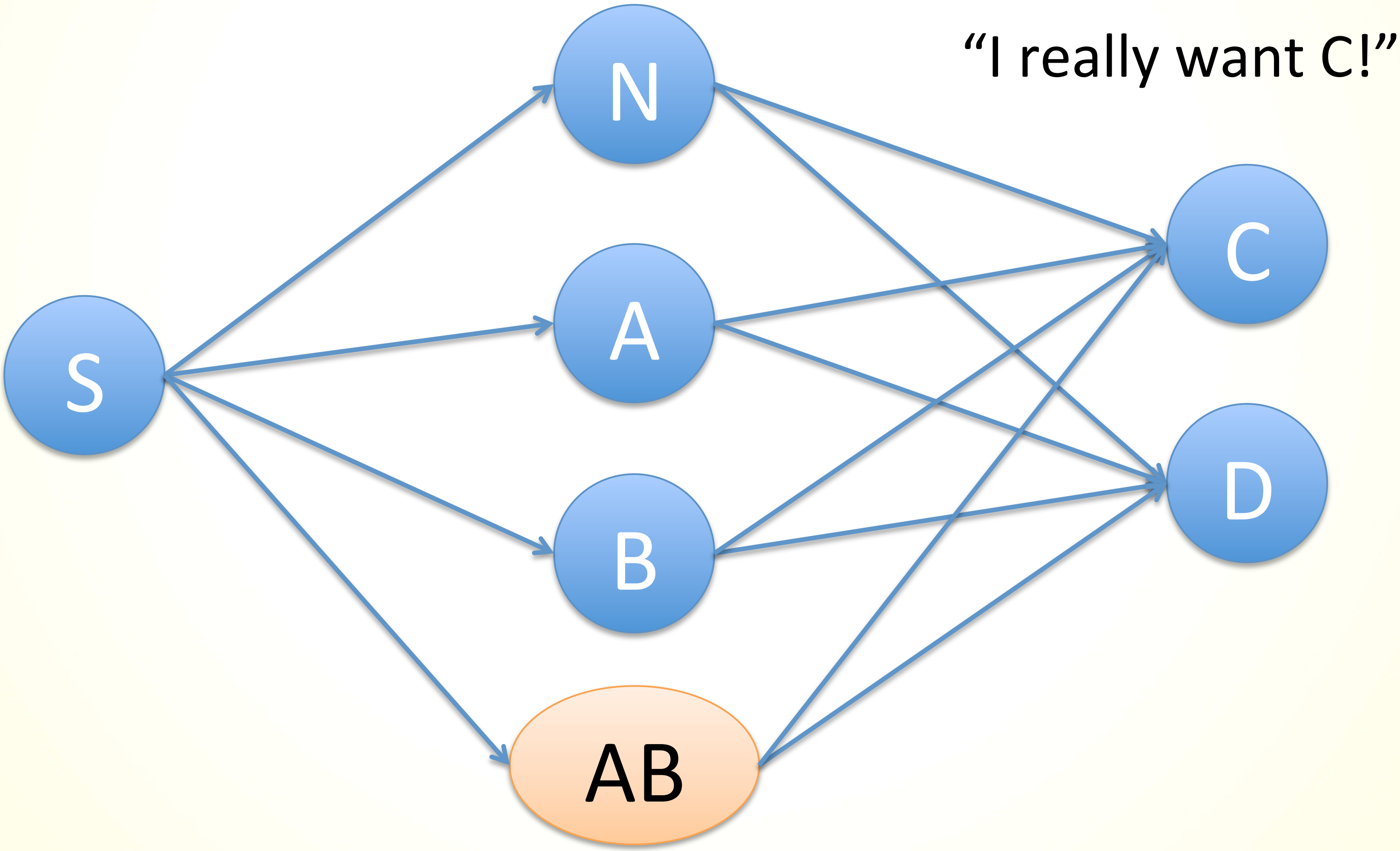
- Corresponds to a novel problem

- Observe desires r_t and a trajectory resulting from u_t
- Infer P

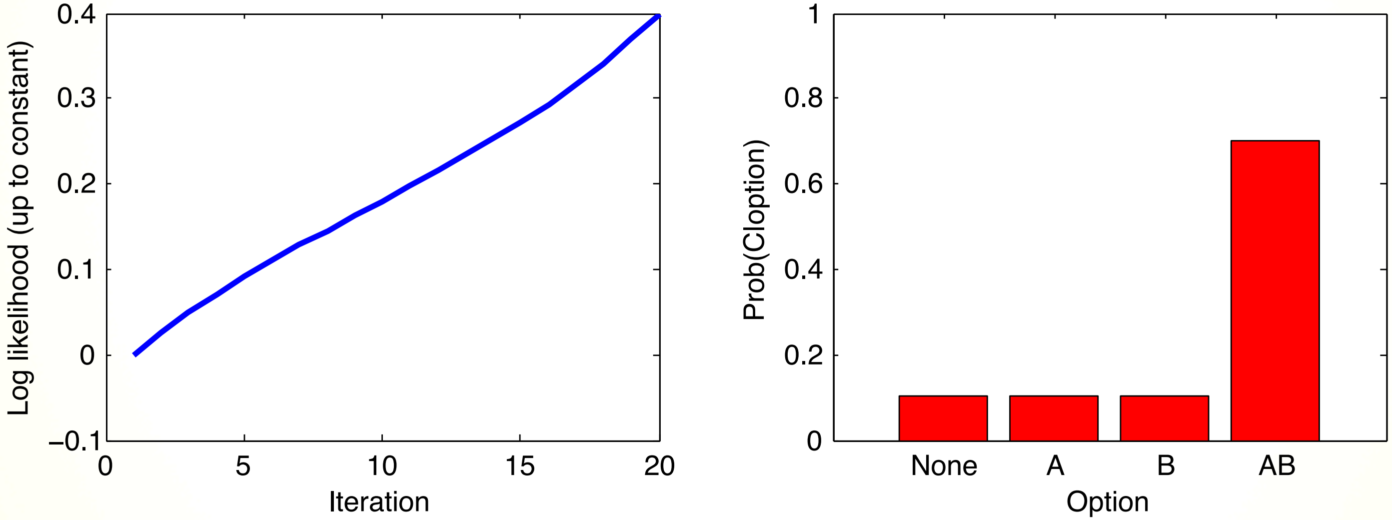
Problem formulation



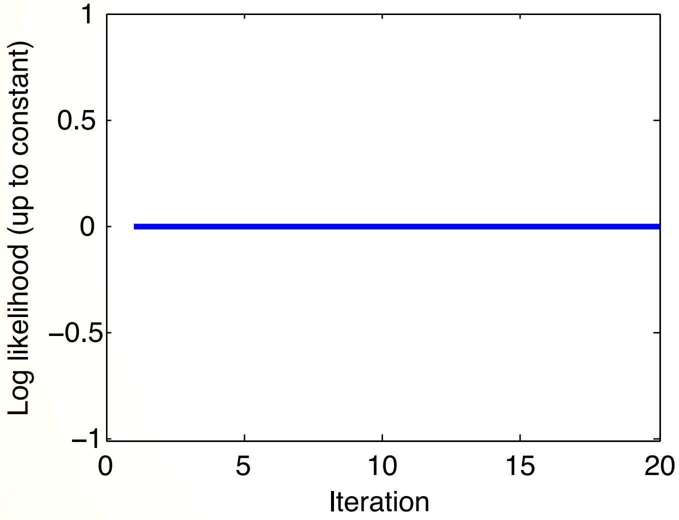
Problem formulation

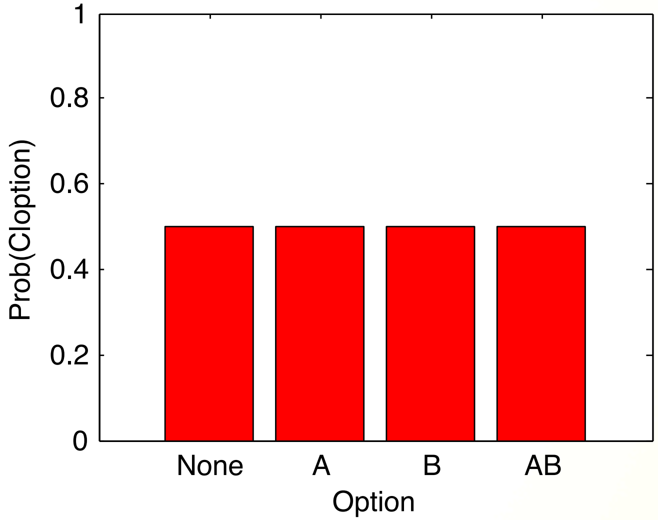


“I really want C”



“Don’t care whether I get C or D”



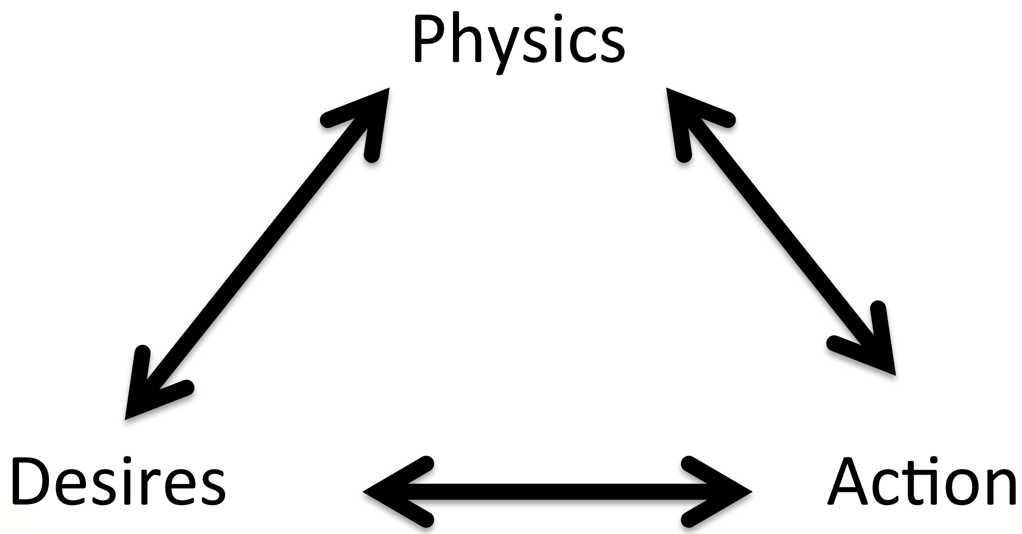


Social causal learning

- Novel learning setting not studied in engineering
- From desires and actions, infer physics/ causal structure

Conclusion

- Intentional action



Conclusion

- Get actions from physics and desires
- Get desires from actions and physics
- Get physics from actions and desires

Challenges

- Recursive reasoning
- Hierarchy
- Beliefs

The brain is not a deep linear network

- Simple models help hone intuitions and are an important precursor to treating more complex cases

- What are deep linear networks good for?
  - Learning dynamics
  - Specific consequences of depth
  - Conceptual underpinnings

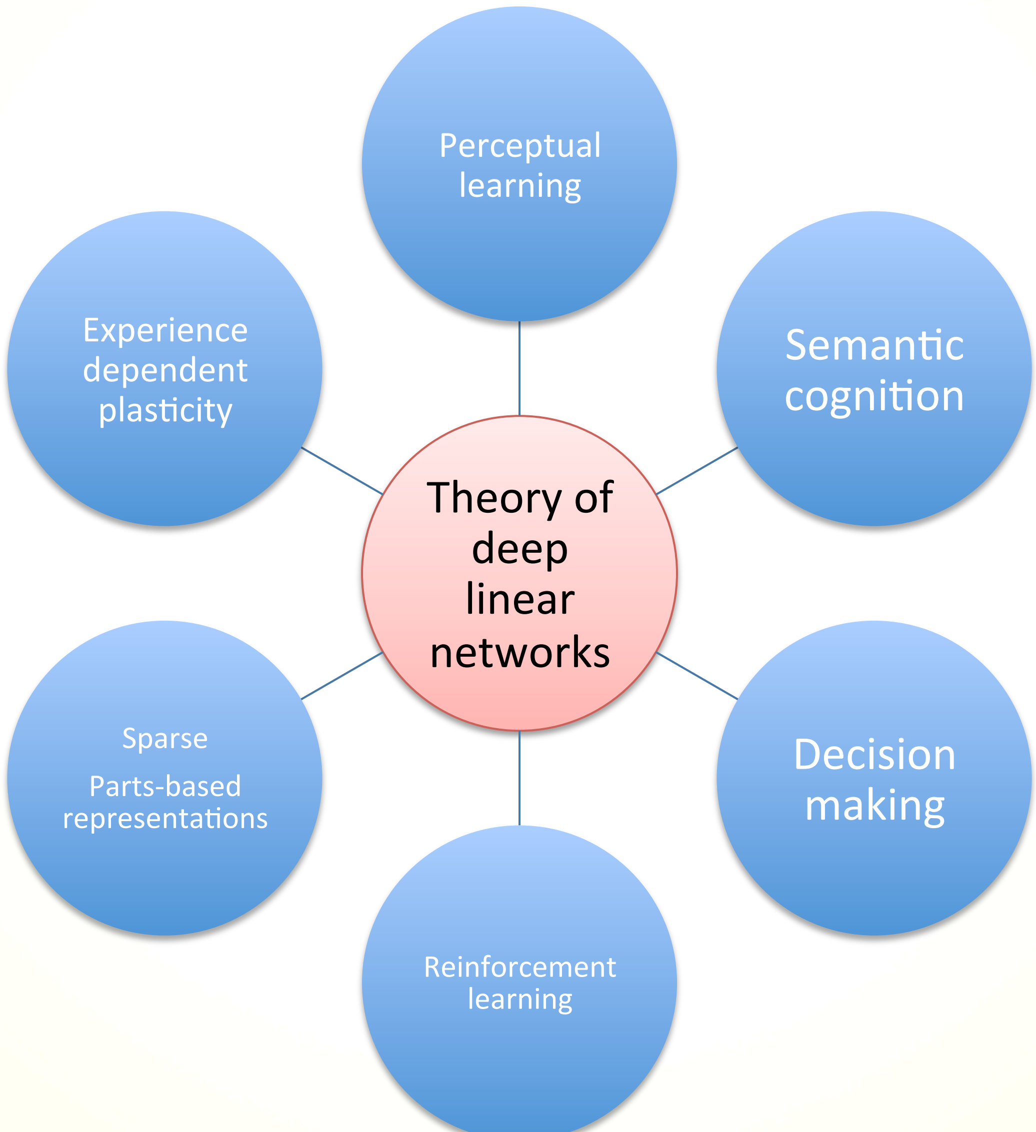
- What aren't they good for?
  - Understanding increased representational power due to nonlinearities

- Must check behavior in deep nonlinear nets, will not always coincide with linear case

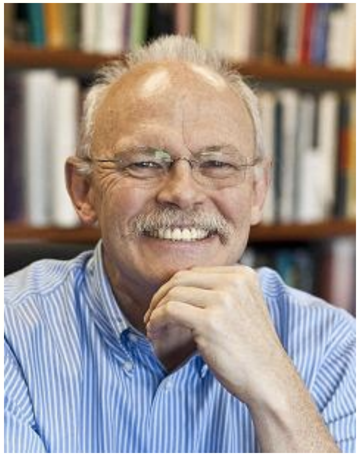
Conclusion

- Learning in a deep, chain-like structure is hard
- Overcoming this challenge may shape how the brain learns in a variety of contexts
- Explains progressive stage-like differentiation in semantic learning
- Spans levels of analysis: single neurons to aspects of semantic cognition

Extensions


Perceptual learning
Semantic cognition
Decision making
Reinforcement learning
Sparse
Parts-based representations
Experience dependent plasticity
Theory of deep linear networks

Thank you!



Jay McClelland

Christoph Schreiner

Andrew Ng

Surya Ganguli

Thank you!

Warm thanks to

- **Rachel Lee**
- Maneesh Bhand
- Ritvik Mudur
- Bipin Suresh
- Koh Pang Wei
- Zhenghao Chen
- Andrew Maas
- Quoc Le
- Ian Goodfellow
- Chris Baldassano
- Jeremy Glick
- Juan Gao
- Cynthia Henderson
- Daniel Hawthorne
- Dave Jackson
- Bryan Seybold
- Craig Atencio
- Nick Steinmetz
- Logan Grosenick

- Members of McClelland, Ng, Schreiner, & Ganguli labs

Questions?

Warm thanks to

- **Rachel Lee**
- Maneesh Bhand
- Ritvik Mudur
- Bipin Suresh
- Koh Pang Wei
- Zhenghao Chen

- Andrew Maas
- Quoc Le
- Ian Goodfellow
- Chris Baldassano
- Jeremy Glick
- Juan Gao

- Cynthia Henderson
- Daniel Hawthorne
- Dave Jackson
- Bryan Seybold
- Craig Atencio
- Nick Steinmetz
- Logan Grosenick

- Members of McClelland, Ng, Schreiner, & Ganguli labs

Biological plausibility

- Gradient descent in the brain?
- Computational level hypothesis $\Delta W = -\lambda \frac{\partial E}{\partial W}$

- Backpropagation: one *algorithm* among many to compute gradient

- Other candidate algorithms:
  - Generalized recirculation algorithm
  - Attention-gated reinforcement learning (AGREL) algorithm

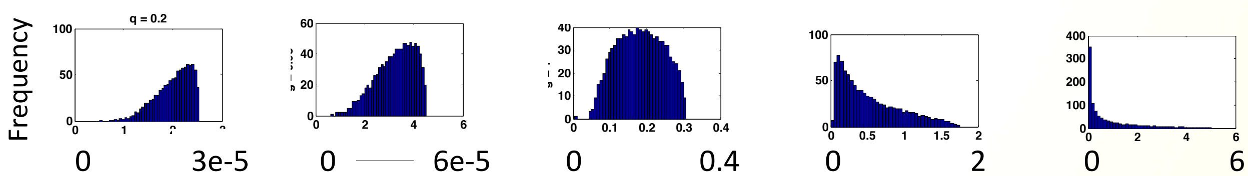
Dynamic Isometry in *nonlinear* nets

Suggests initialization for *nonlinear* nets

- near-isometry on subspace of large dimension
- Singular values of *end-to-end* Jacobian $J_{ij}^{N_l,1}(x^{N_l}) \equiv \left.\frac{\partial x_i^{N_l}}{\partial x_j^1}\right|_{x^{N_l}}$ concentrated around 1.

Scale orthogonal matrices by gain *g* to counteract contractive nonlinearity

Singular values of *J*



Gain *g=0.9* *g=0.95* *g=1* *g=1.05* *g=1.1*

Just beyond *edge of chaos (g>1)* may be good initialization

Dynamic Isometry Initialization

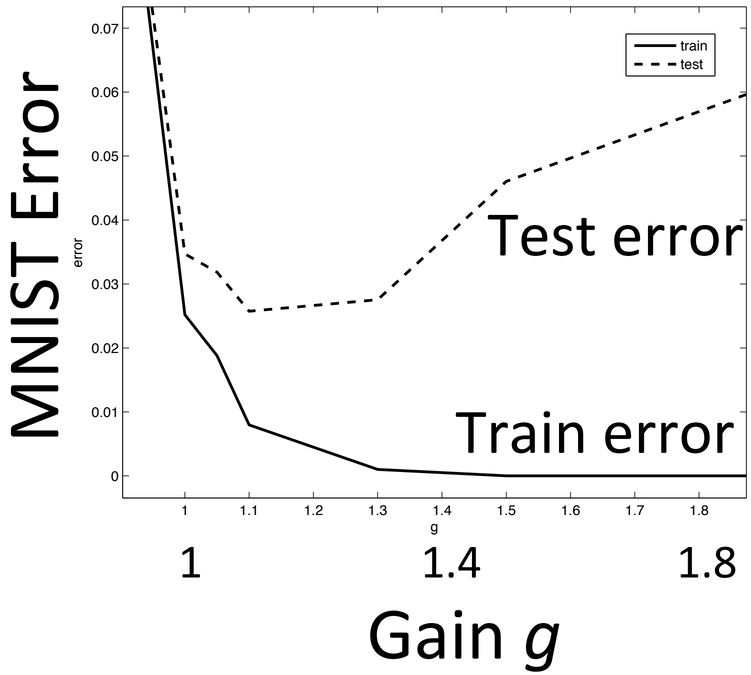
- g>1 speeds up **30 layer nonlinear** nets
  - Tanh network, softmax output, 500 units/layer
  - No regularization (weight decay, sparsity, dropout, etc)

MNIST Classification error, epoch 1500	**Train Error (%)**	**Test Error (%)**
Glorot (g=1, random)	2.3	3.4
g=1.1, random	1.5	3.0
g=1, orthogonal	2.8	3.5
Dynamic Isometry (g=1.1, orthogonal)	**0.095**	**2.1**

- Dynamic isometry reduces test error by 1.4% pts

Fast Training from Large Gain Initializations

- Deep networks + large gain factor *g* train exceptionally quickly
- But large *g* incurs heavy cost in generalization performance



- Suggests small initial weights regularize towards smoother functions
- Training difficulty arises from *saddle points*, not local minima