Machine Learning

CSCI 5622 Fall 2020

Prof. Claire Monteleoni

Today

- Intro to Online Learning (continued)
- Intro to Reinforcement Learning

Online Learning

Canonical online learning

Additive updates (e.g. standard Perceptron)

Multiplicative updates (e.g. Winnow)

Some online learning algorithms:

- A modification of Perceptron in which the angle decreases monotonically → monotonic error decrease when data is Uniform

- Algorithms descended from Winnow, for learning from time-varying data streams:

Tracking the best expert

Tracking shifting experts

Fixed-share

Learn-α

Multiplicative updates

Canonical algorithms:

- Halving Algorithm of [Angluin, ‘88].
- Winnow Algorithm due to [Littlestone, ‘88].
- Weighted Majority Algorithm [Littlestone & Warmuth, ’89]

- Algorithms descended from these for learning from time-varying data streams:
  - Tracking the best expert
  - Tracking shifting experts
    - Fixed-share
    - Learn-α

Online learning with expert advice

Consider any prediction or forecasting problem with an ensemble of "experts." An expert is a time-series (*i.e.* a sequence of "predictions"), but need not be a good predictor.

- Predicting climate change
  - Intergovernmental Panel on Climate Change (IPCC) multi-model ensemble of climate models
- Weather prediction
  - Combine the predictions of an ensemble of weather models
- Portfolio management / volatility prediction
  - Experts can be analysts regularly making predictions about stock performance
  - Experts can be the stock prices themselves
- GDP Nowcasting
  - Experts can be monthly reports, *cf.* GDPNow (FT900, Monthly Retail Trade Report)
  - Experts can be GDPNow and other such real-time prediction methods

Online learning with expert advice

Problem set-up:

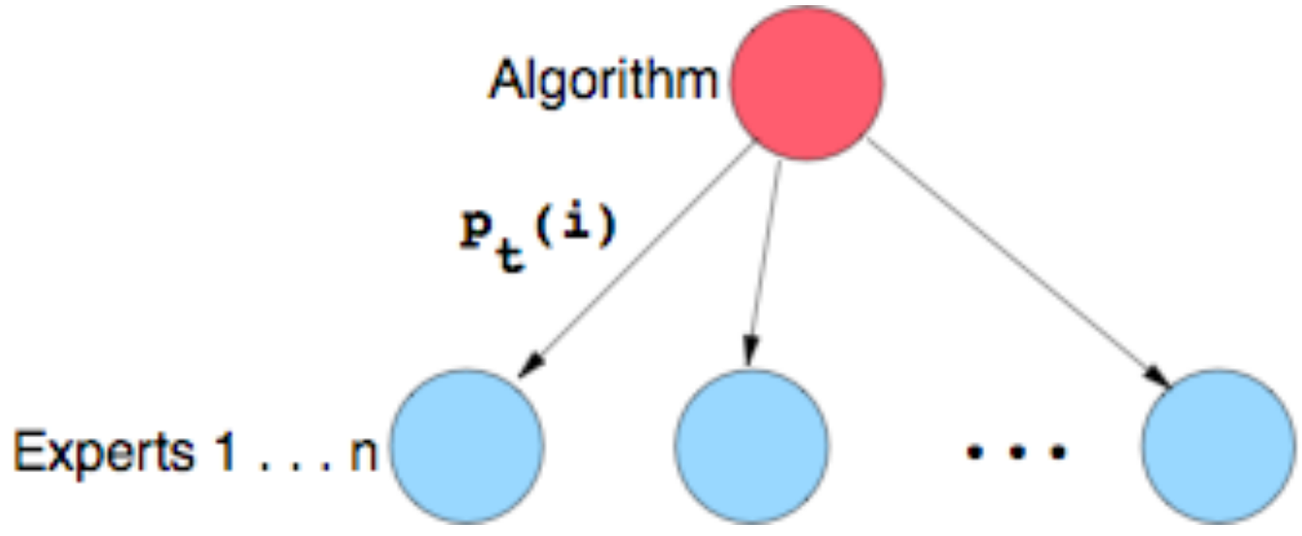
- Observations arrive one-at-a-time, in a stream
- A set of “experts” make predictions, at each time

Typical online learning algorithm template:

- First, the algorithm observes expert predictions (only) and must make a combined prediction
- Then, the true observation is revealed
- Finally, the Algorithm can update the “weight” of each expert
- Repeat

Online learning with expert advice

Learner maintains distribution over *n* "experts." [An ensemble method]



Experts are black boxes: need not be good predictors, can vary with time, and depend on one another.

Learner predicts based on a probability distribution $p_t(i)$ over experts, i, representing how well each expert has predicted recently.

L(i, t) is prediction loss of expert i at time t. Defined per problem.

Update $p_t(i)$ using Bayesian updates: $p_{t+1}(i) \propto p_t(i)e^{-L(i,t)}$

Regret model

No statistical assumptions (non-stochastic setting)

No assumptions on observation sequence.

E.g., observations can even be generated online by an adaptive adversary.

Framework models supervised learning:

Regression, estimation or classification.

Many prediction loss functions:

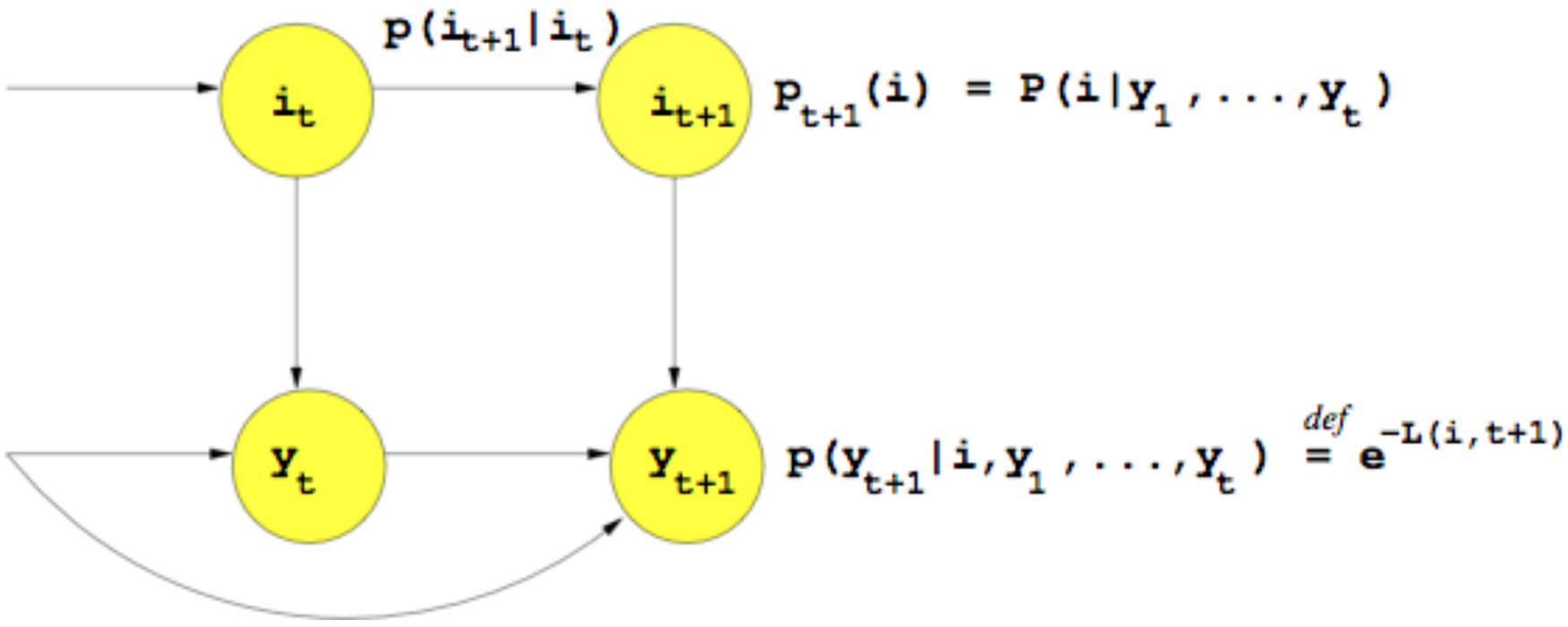
- many hypothesis classes
- problem need not be separable

Analyze regret: difference in cumulative prediction loss from that of the optimal (in hind-sight) comparator algorithm for the particular sequence observed.

Shifting algorithms

To handle changing observations, maintain $p_t(i)$ via (generalized) HMM.

Hidden state: identity of the current best expert.



Performing Bayesian updates on this HMM yields a family of algorithms.

$$p_{t+1}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j)$$

Static update, P(i | j) = δ(i,j) gives [Littlestone&Warmuth'89] algorithm:

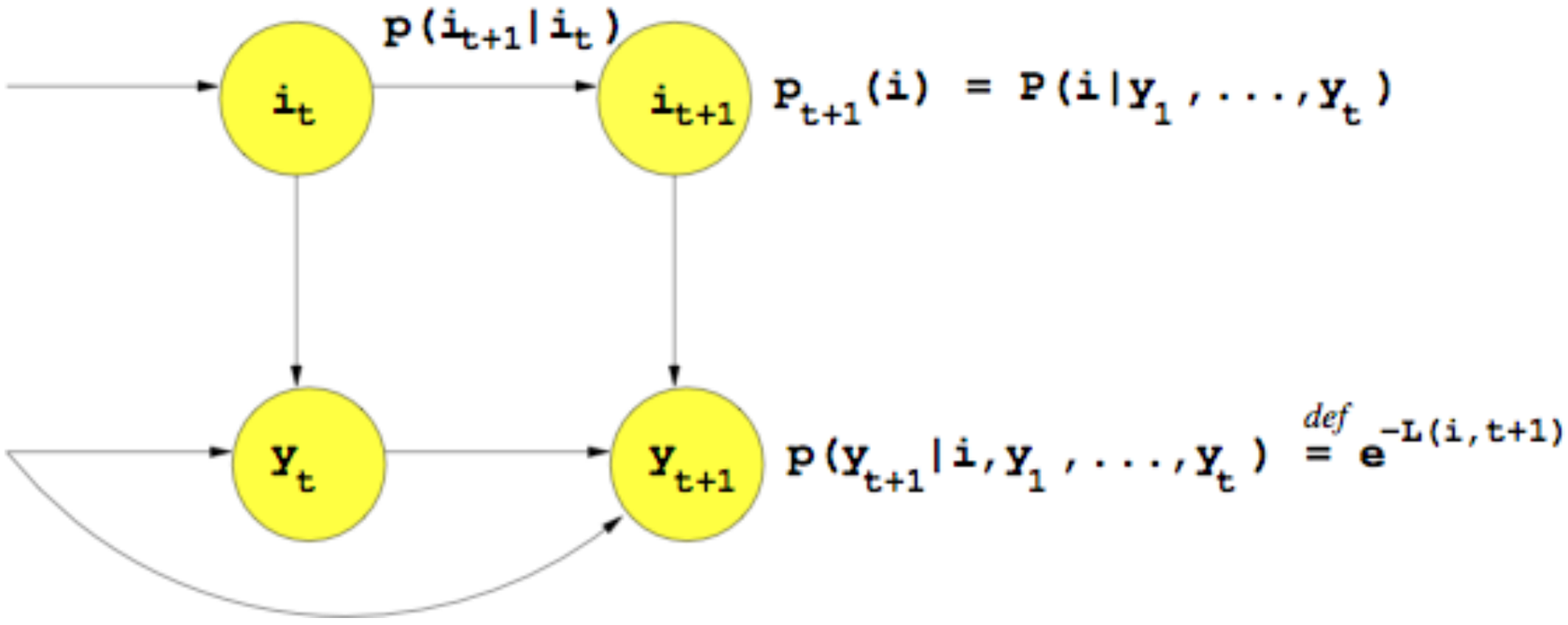
Weighted Majority Algorithm, a.k.a. Static-Expert. $p_{t+1}(i) \propto p_t(i) e^{-L(i,t)}$

Regret wrt i in E, is additive O(log |E|).

Shifting algorithms

To handle changing observations, maintain $p_t(i)$ via (generalized) HMM.

Hidden state: identity of the current best expert.



Performing Bayesian updates on this HMM yields existing OL algorithms.

$$p_{t+1}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j)$$

[Herbster&Warmuth'98]

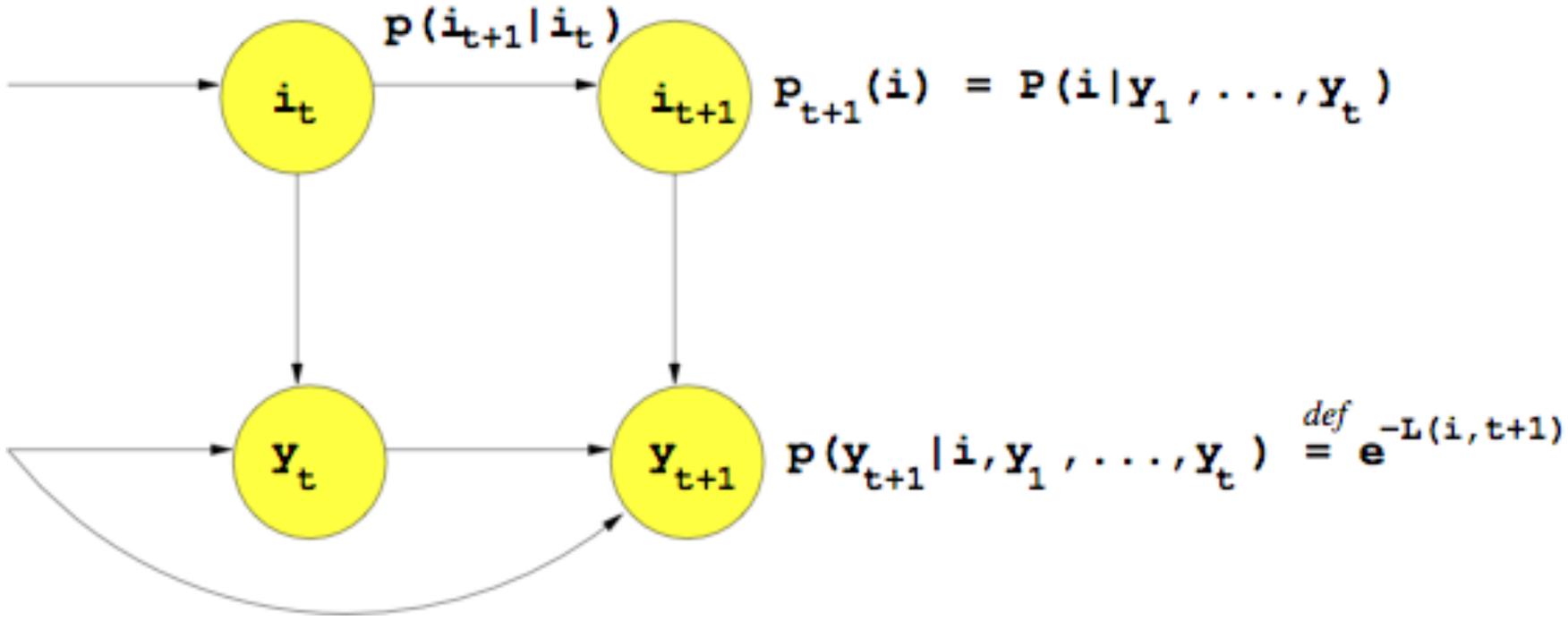
Model shifting concepts via:

$$P(i|j;\alpha) = \begin{cases} (1-\alpha) & i = j \\ \frac{\alpha}{n-1} & i \neq j \end{cases}$$

Regret wrt best k segmentation of sequence by E, is f(k, α, |E|).

Upper bound on regret

Theorem [Monteleoni & Jaakkola, NIPS 2003]: For OL algorithms performing weight updates on "experts" as Bayesian updates of:



i.e. $p_{t+1}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j)$ with transition matrix Θ: $\theta_{ij} = p(j \mid i)$.

$$L_T(\Theta) - L_T(\Theta^*) \leq (T-1) \max_i D(\Theta_i \| \Theta_i^*)$$

Corollary: If $P(i|j;\alpha) = \begin{cases} (1-\alpha) & i = j \\ \frac{\alpha}{n-1} & i \neq j \end{cases}$

$$L_T(\alpha) - L_T(\alpha^*) \leq (T-1) D(\alpha \| \alpha^*)$$

$O(1)$ upper bounds.

Lower bound on regret

Theorem [M & Jaakkola, NIPS 2003]:

A lower bound on regret for shifting algorithms.

Value of bound is sequence dependent.

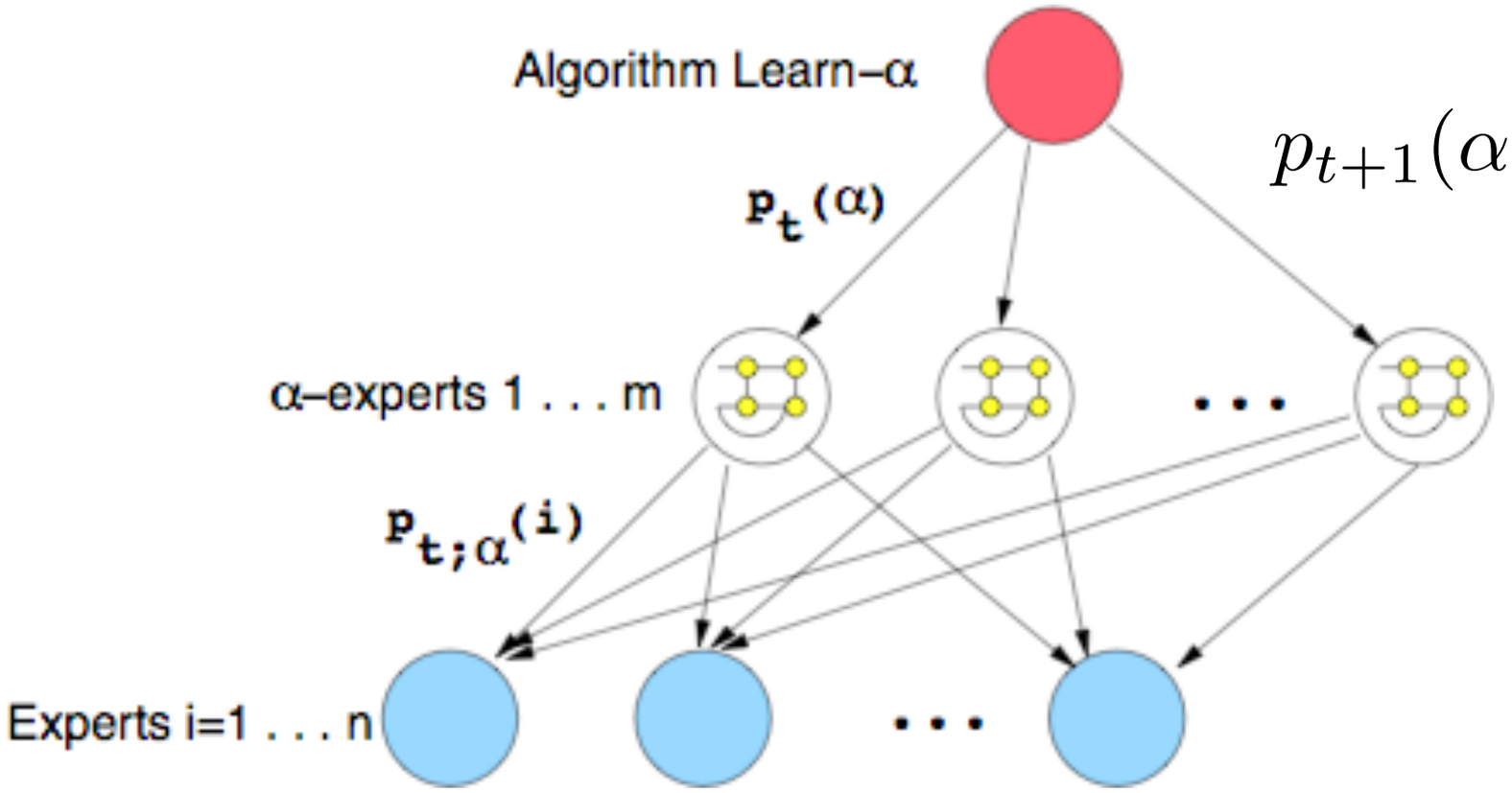
Depends both on α^* and second derivative of L_T near α^*.

Can be $\Omega(T)$, depending on the sequence of length *T*.

Learn-α algorithm

[M, 2003] [M & Jaakkola, NIPS 2003]

Learn-α algorithm: Track best α–expert, each updating with different value of α. Use Bayesian updates to track the best α.



$$p_{t+1}(\alpha) \propto p_t(\alpha) e^{-L(\alpha,t)}$$

$$p_{t+1;\alpha}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j;\alpha)$$

Upper bound for Learn-α

Upper bound on regret for Learn-α algorithm of O(log T).

Theorem [M & Jaakkola, NIPS 2003]: The Learn-α algorithm with m values of α, has cumulative loss on T observations:

$$L_T^{top} \leq L_T(\alpha^*) + \log m + (T-1)D(\alpha^* \| \alpha_{j^*})$$

Corollary: Given T, we can choose the m α values that optimize the bound, yielding:

$$L_T^{top} \leq L_T(\alpha^*) + \frac{1}{2}\log T + O(1)$$

- Discretize α such that

$$D(\alpha^* \| \alpha_{j^*}) \leq \frac{C}{T-1}$$

Application of Learn-α to wireless

[Monteleoni, Balakrishnan, Feamster & Jaakkola '07]

Energy/Latency tradeoff for 802.11 wireless nodes:

Awake state consumes too much energy.

Sleep state cannot receive packets.

IEEE 802.11 Power Saving Mode:

Base station buffers packets for sleeping node.

Node wakes at regular intervals (S = 100 *ms*) to process buffered packets, B. → Latency introduced due to buffering.

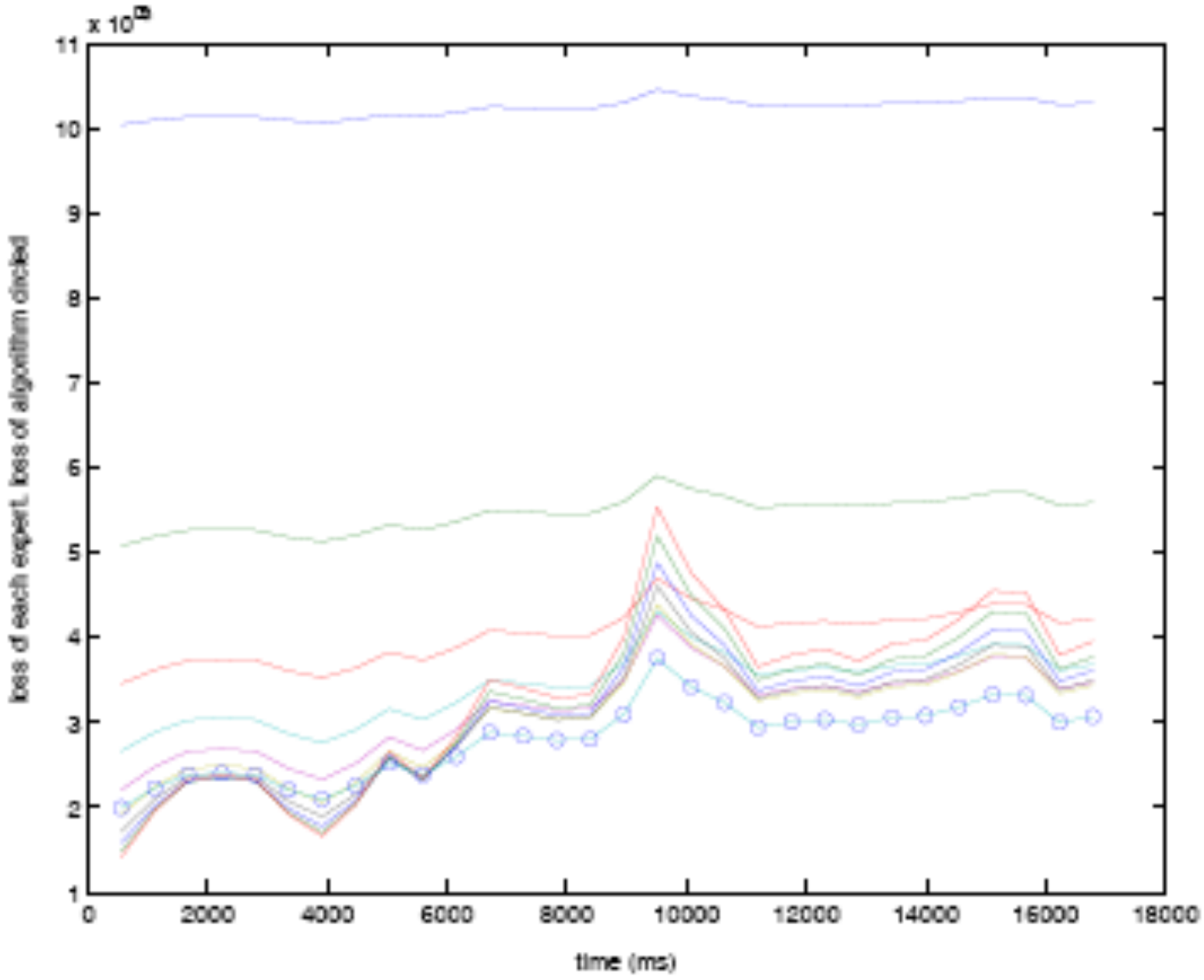
Apply Learn-α to adapt sleep duration to shifting network activity.

Simultaneously learn rate of shifting online.

Experts: discretization of possible sleeping times, e.g. 100 *ms*.

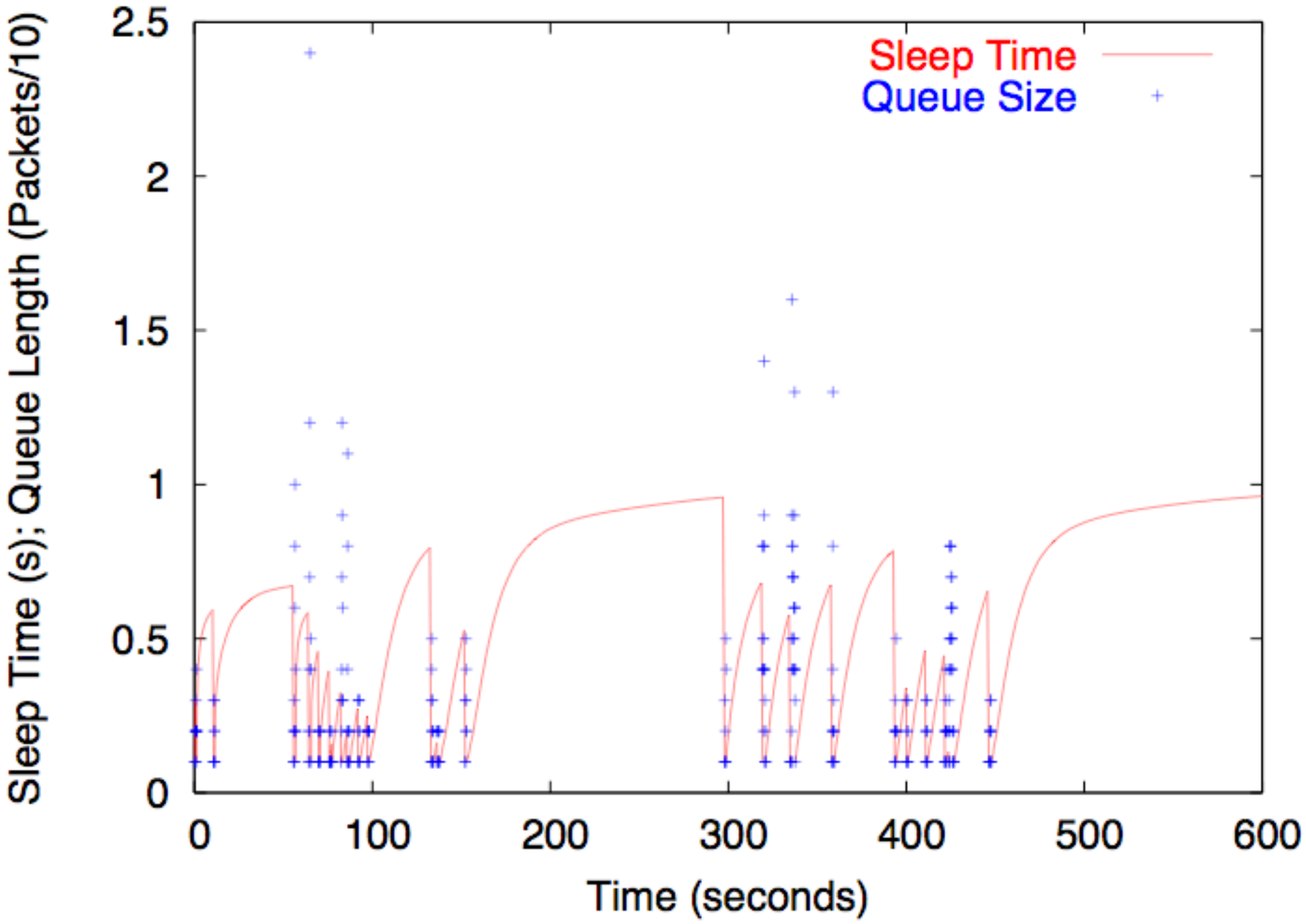
Minimize loss function convex in energy, latency: $\gamma \frac{B_t S_t}{2} + \frac{1}{\log S_t}$

Application of Learn-α to wireless



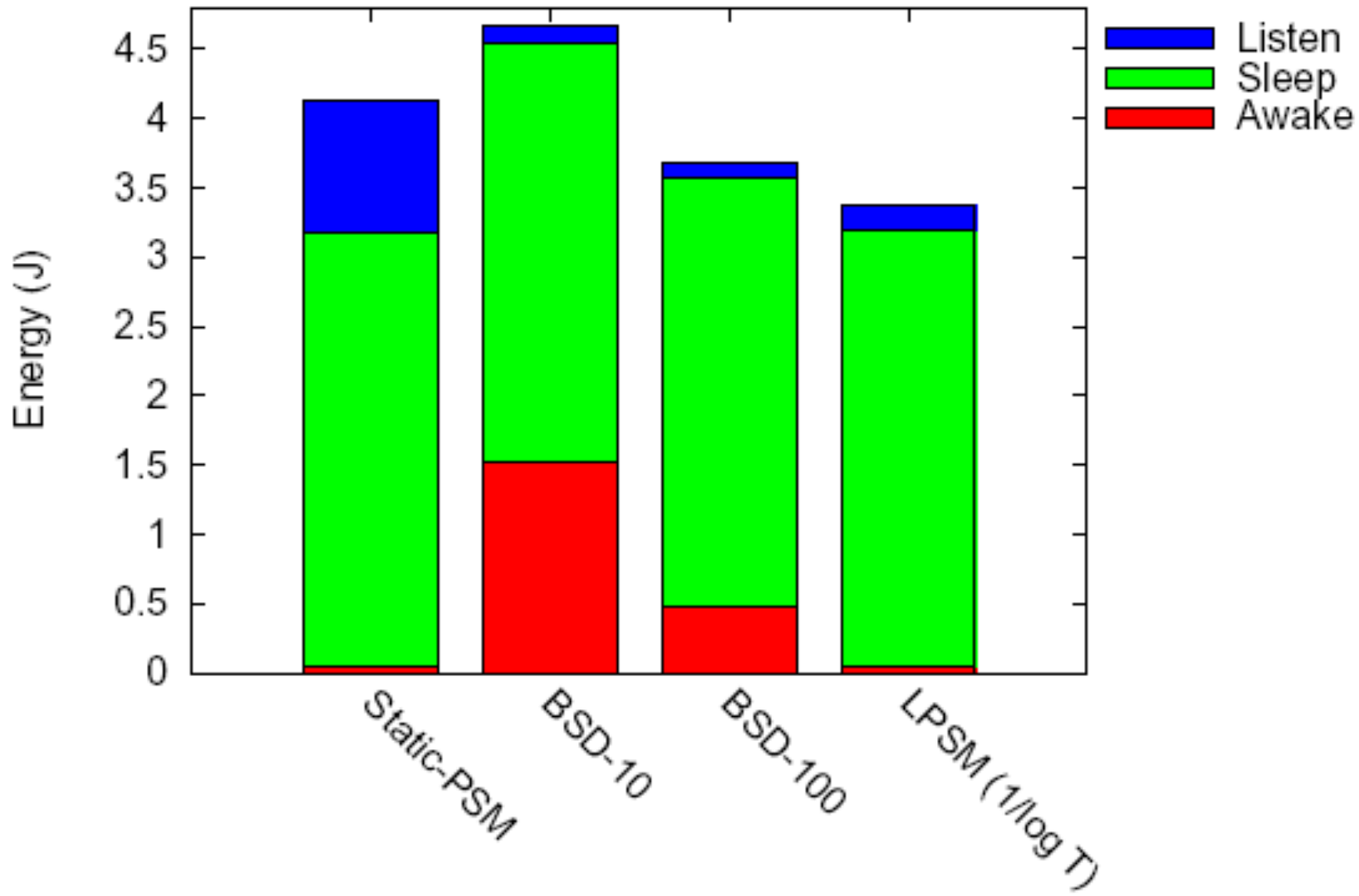
Loss of each expert, loss of algorithm circled.

Application of Learn-α to wireless



Evolution of sleep times

Application of Learn-α to wireless



Energy usage: reduced by 7-20% from 802.11 PSM

Average latency 1.02x that of 802.11 PSM

Reinforcement Learning

- Introduction to Reinforcement Learning
  - Formulation of Reinforcement Learning
  - Markov decision process (MDP)
  - Policy, and optimal policy
  - *Q*-learning

Content primarily adapted from: Richard S. Sutton Andrew G. Barto: Reinforcement Learning: An Introduction. 2nd Edition draft, 2017.

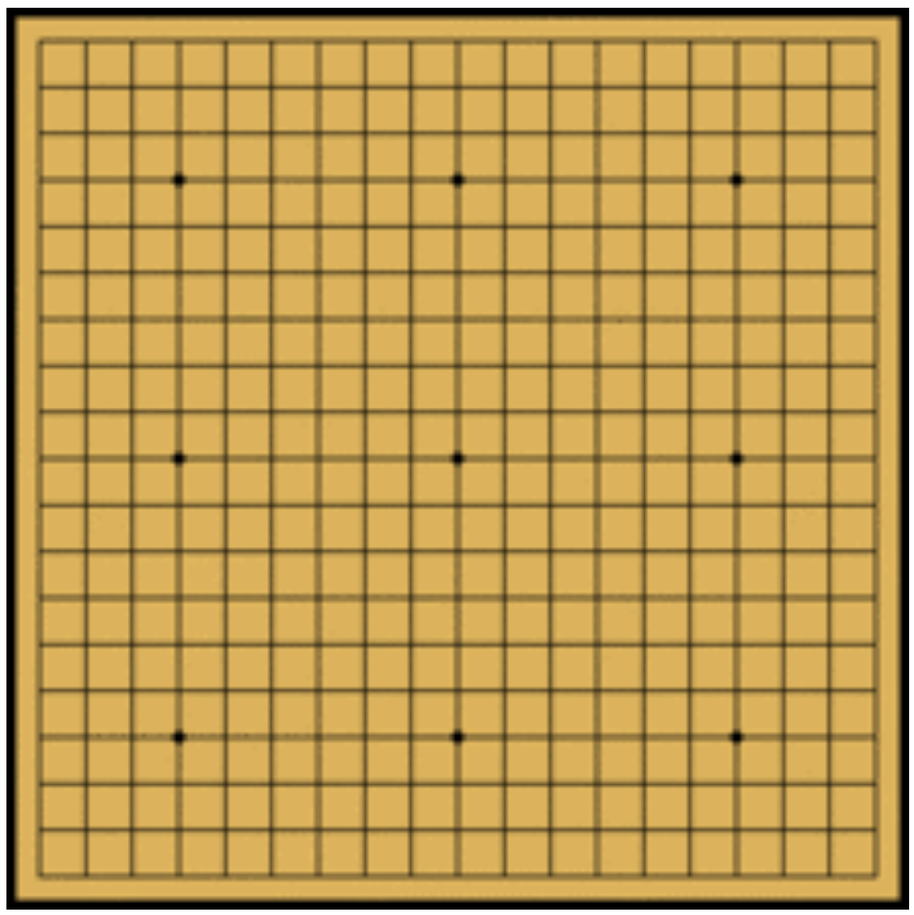
Slides primarily adapted from: J. Boyd-Graber, C. Ketelsen, C. Tan

Reinforcement learning examples

- Mnih et al. 2013
- https://www.youtube.com/watch?v=V1eYniJ0Rnk

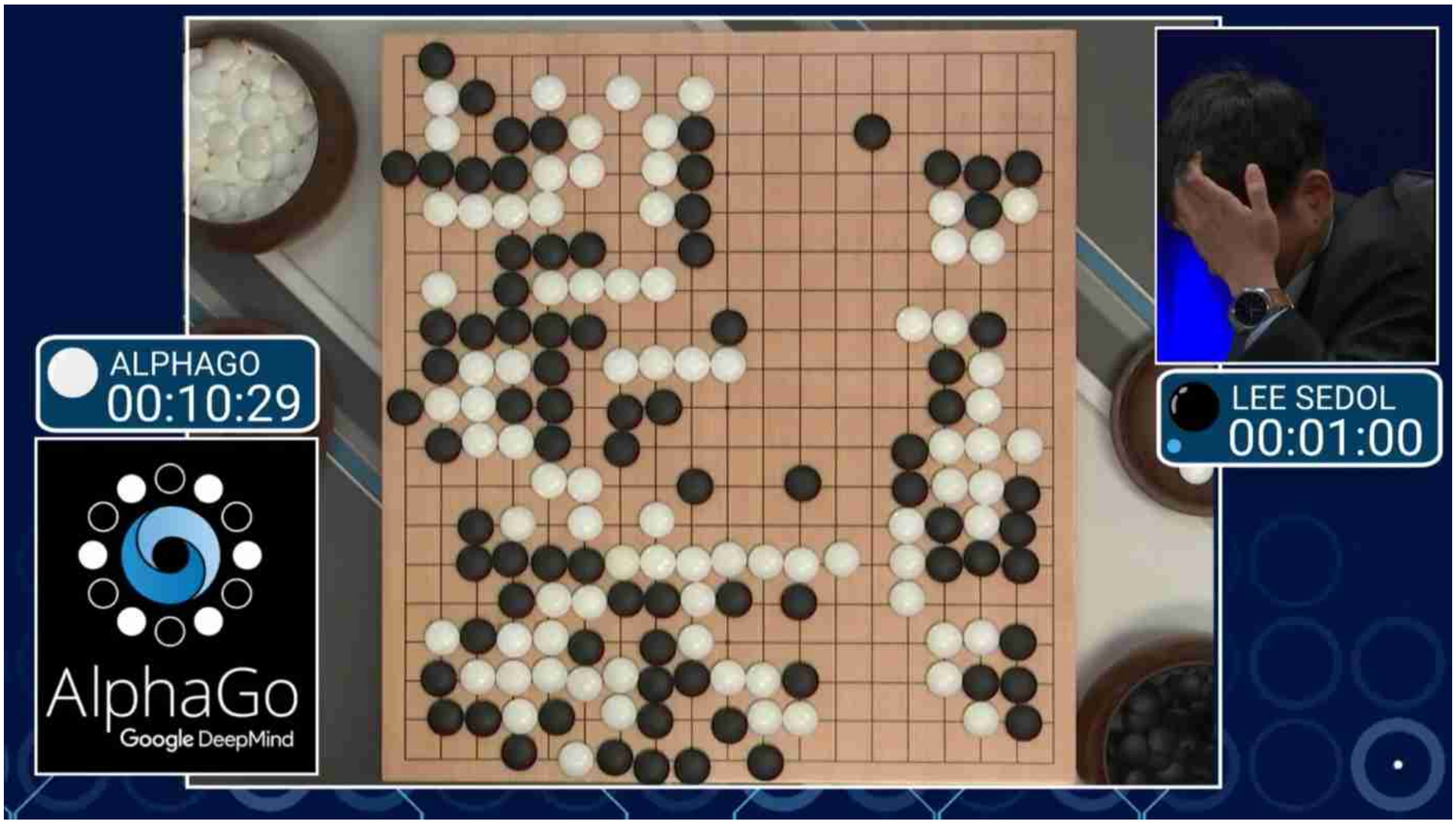
Reinforcement learning examples

- Go



- Wait for it:… https://www.youtube.com/watch?v=Jq5SObMdV3o

Reinforcement learning examples: 2016



An agent learns to behave in an environment

Supervised learning

Data: X

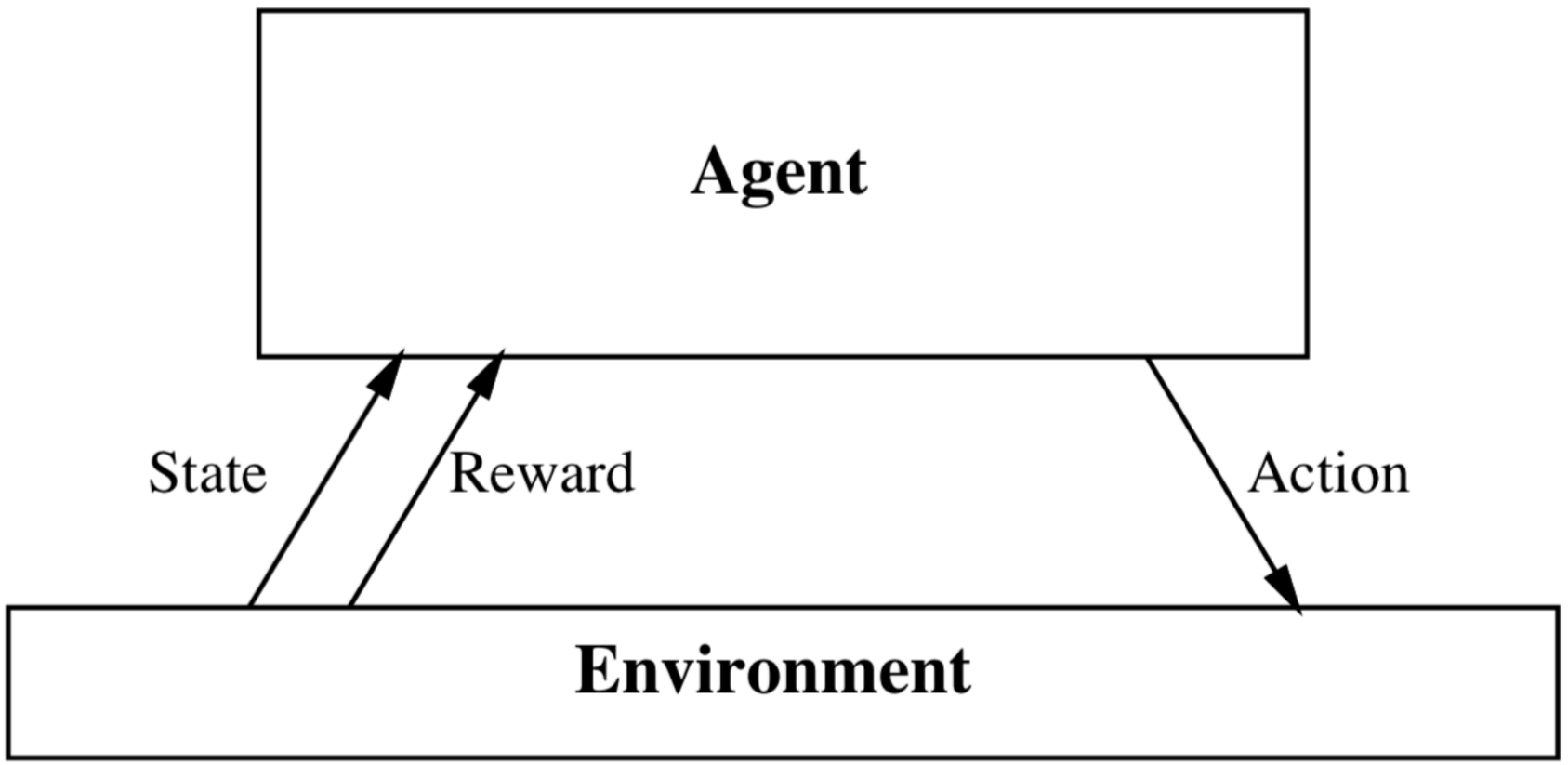
Labels: Y

Unsupervised learning

Data: X

Latent structure: Z

Reinforcement learning



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \ldots$$

Markov decision process (MDP)

- finite set of states $\mathcal{S}$
- set of actions for each state $\mathcal{A}(s)$
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in \mathcal{A}(s)$
- Markov assumption: S_{t+1}, R_{t+1} only depends on S_t, A_t.

$$p(s', r \mid s, a) \doteq P(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$$

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1$$

Markov decision process (MDP)

Transition probability:

$$p(s' \mid s, a) \doteq P(S_t = s' \mid S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

Markov decision process (MDP)

Transition probability:

$$p(s' \mid s, a) \doteq P(S_t = s' \mid S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

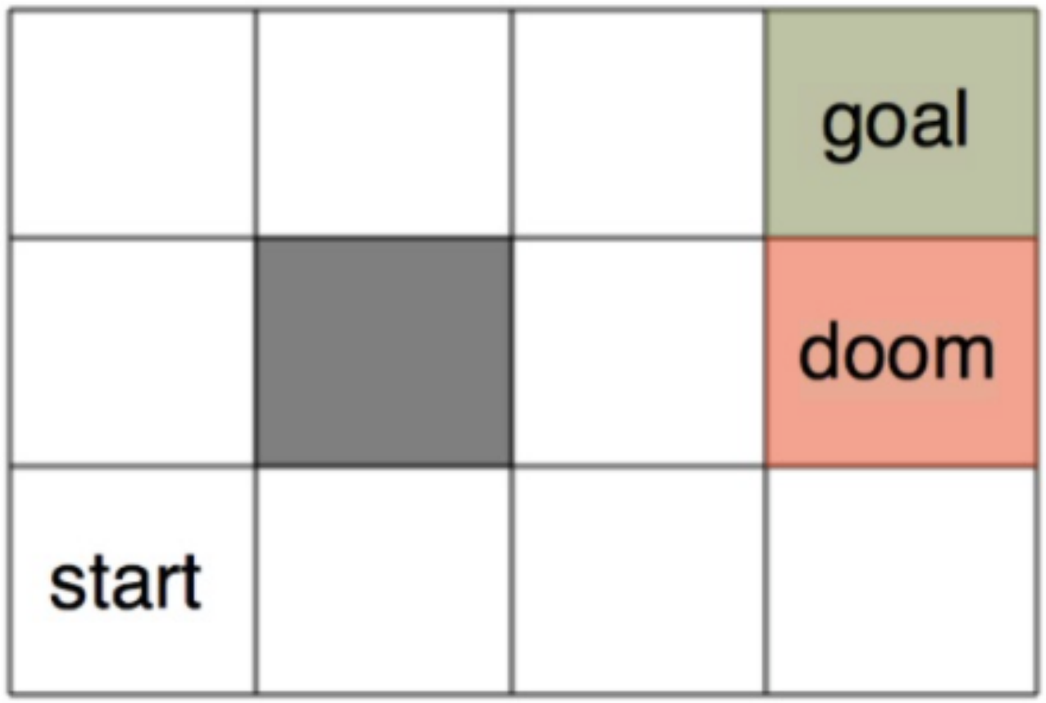
Expected rewards

$$r(s, a) \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a\right] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \int} p(s', r \mid s, a)$$

$$r(s, a, s') \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

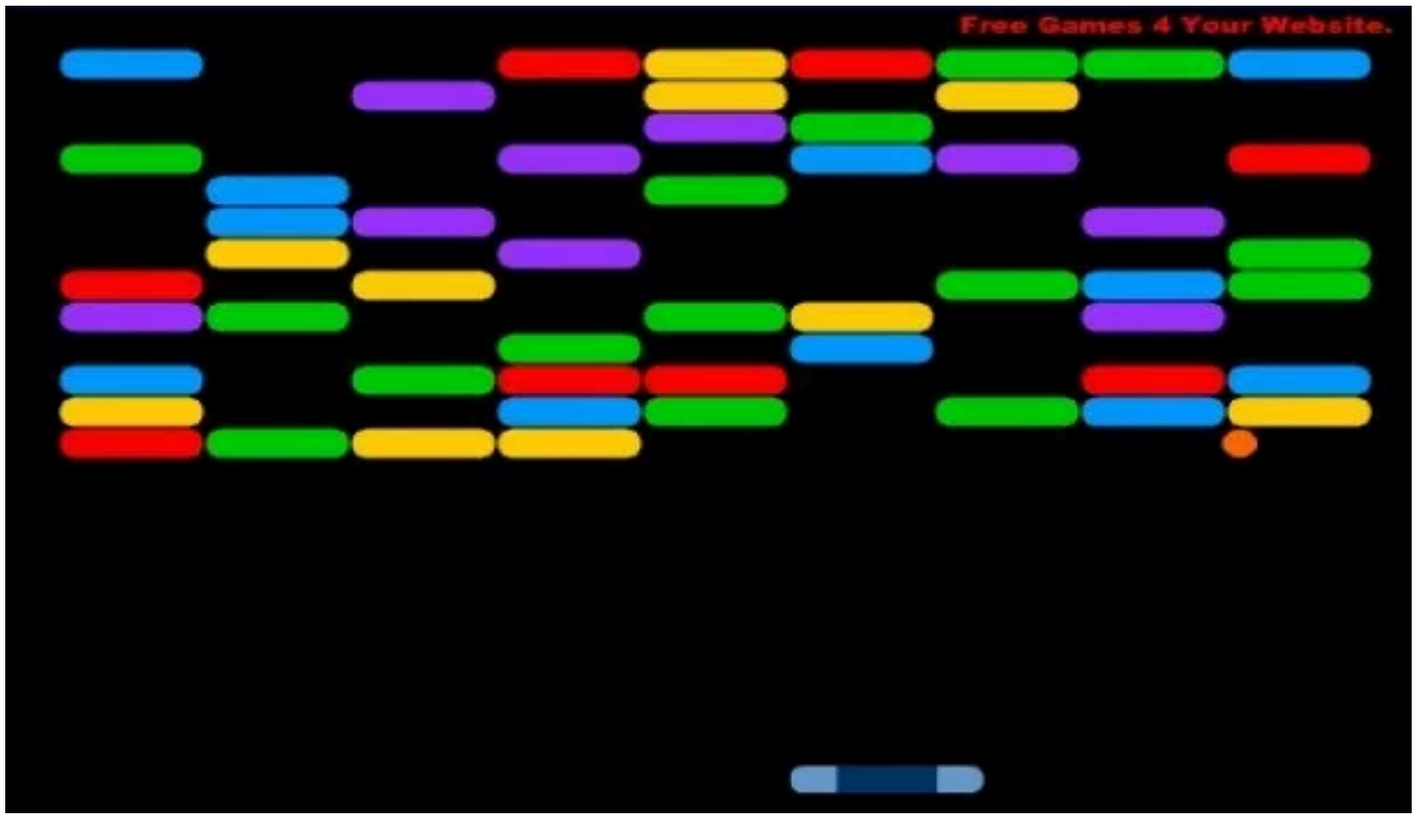
A few examples

- Grid world



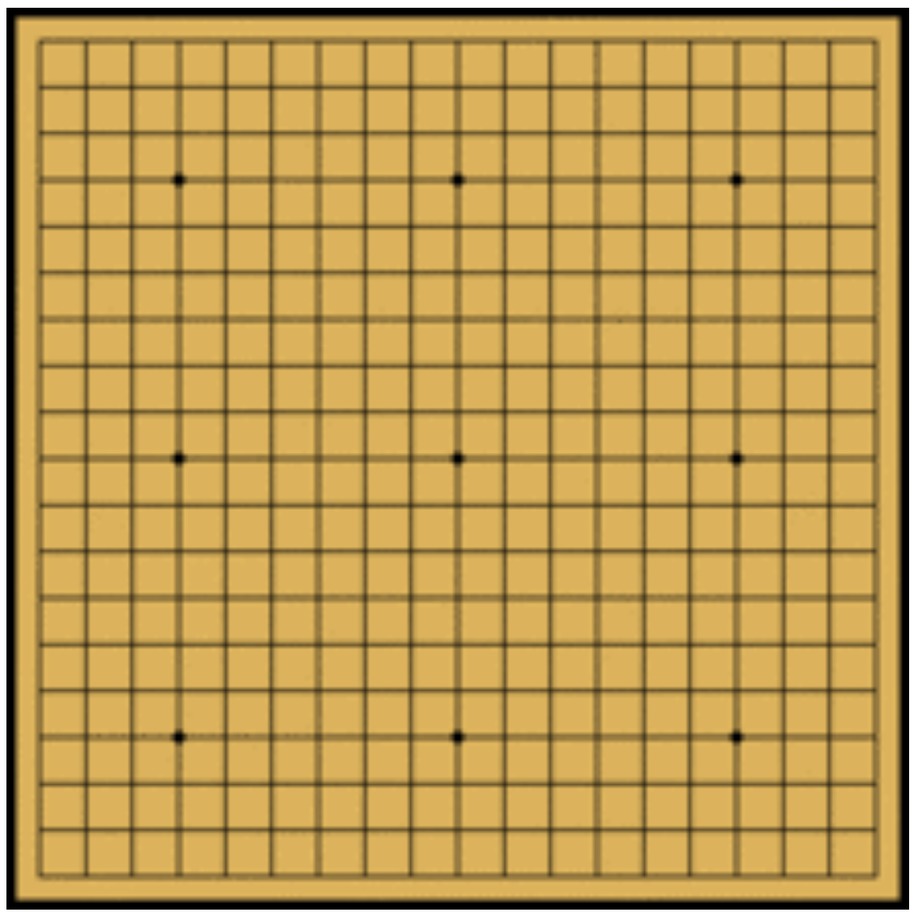
A few examples

- Atari game (Bonus: try Google image search “atari breakout”)



A few examples

- Go



- Wait for it:… https://www.youtube.com/watch?v=Jq5SObMdV3o

An RL agent's learning task:

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

$$\mathbb{E}\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\right]$$

  from any starting state in S
- here $0 \leq \gamma < 1$ is the discount factor for future rewards

Note something new:

- Target function is $\pi : S \rightarrow A$
- but we have no training examples of form $\langle s, a \rangle$
- training examples are of form $\langle \langle s, a \rangle, r \rangle$

An RL agent needs the following:

- Policy: agent's behavior function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

Policy

- Policy: Determines the agent's behavior, i.e. action selection.
  - Deterministic: Policy is a function of state: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

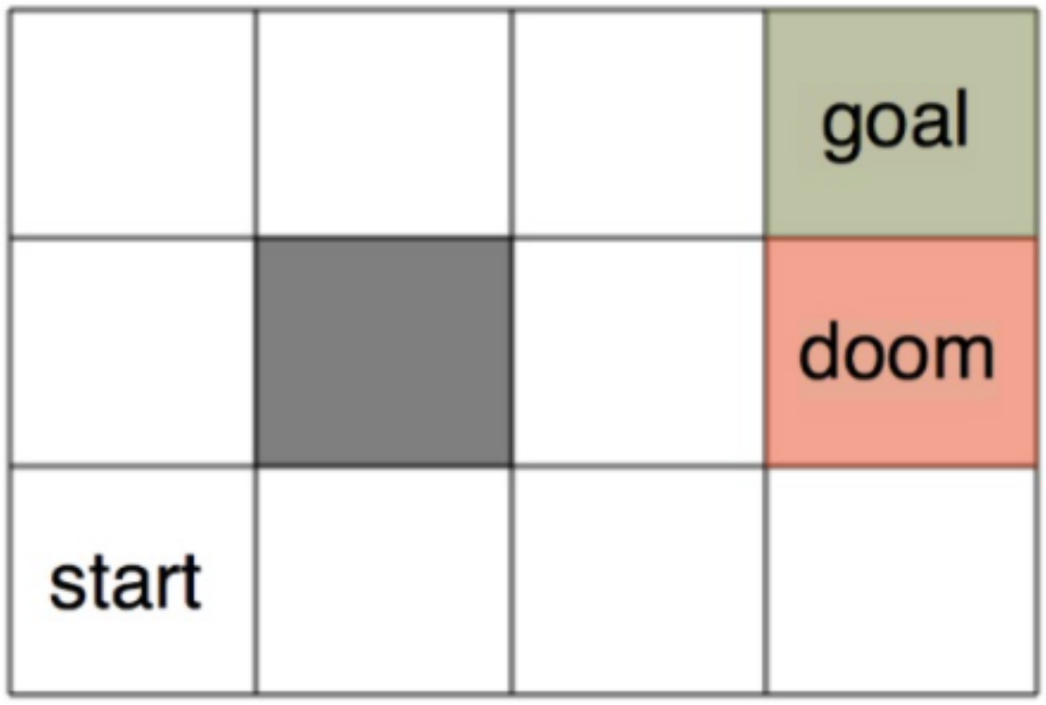
$$a = \pi(s)$$

  - Stochastic: Policy is a distribution over actions, conditioned on state:

$$\pi(a|s) = p(a|s)$$

Policy examples in grid world

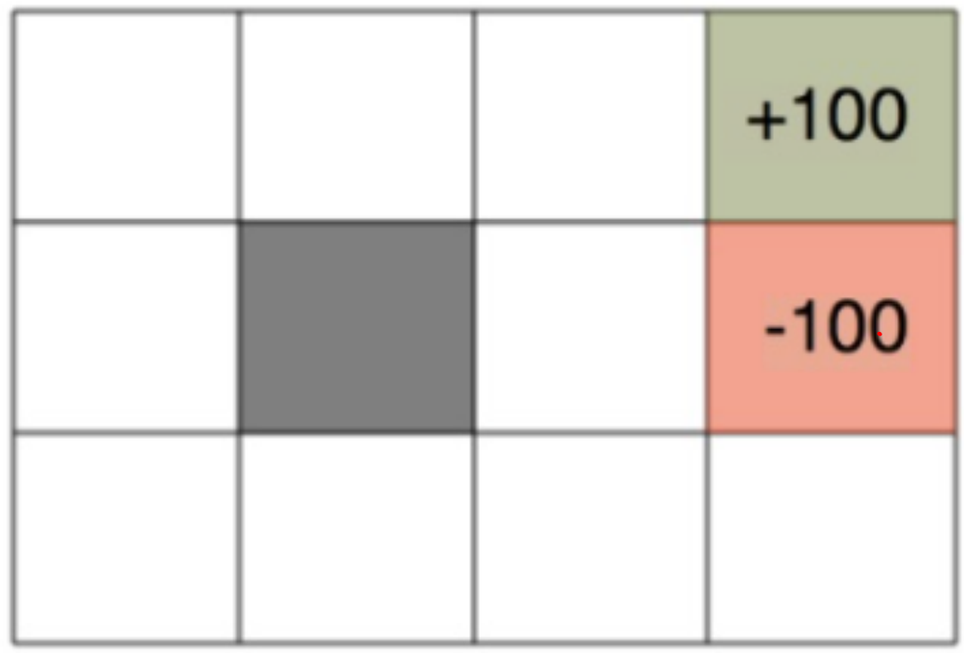
- Grid world



Actions: agent most move one step in a direction:

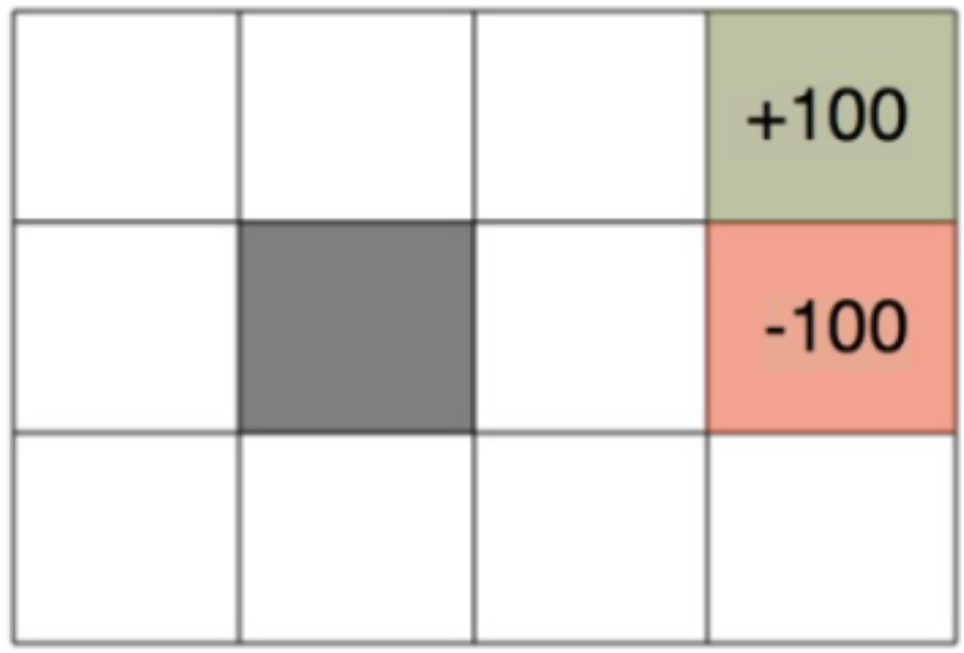
- {Up, Down, Right, Left}

Policy examples in grid world



Rewards can be positive or negative
Delayed reward: might not get reward until you reach goal
Might have **negative** reward until you reach goal

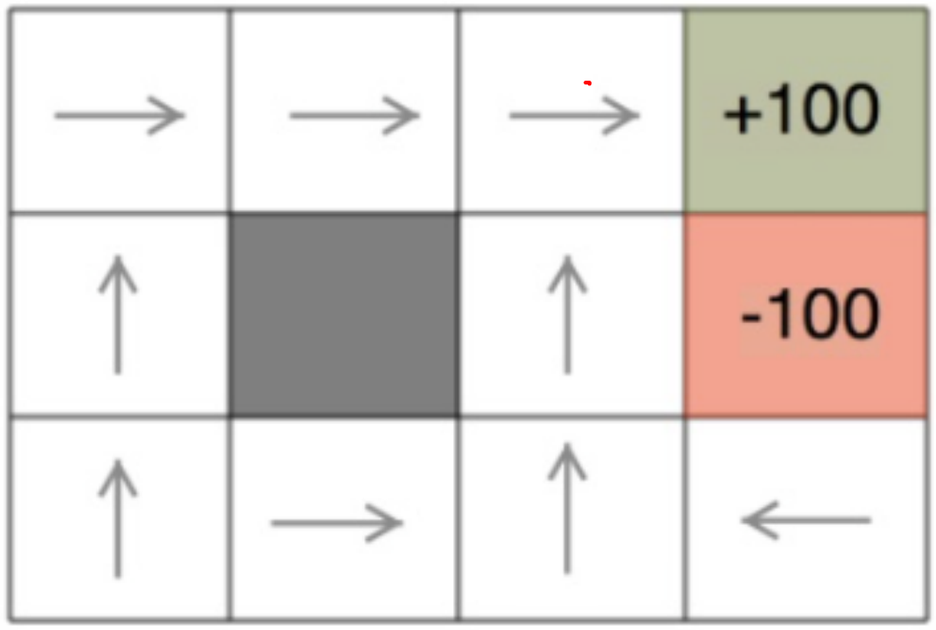
Policy examples in grid world



What is the optimal policy if $R(s) = -5$?

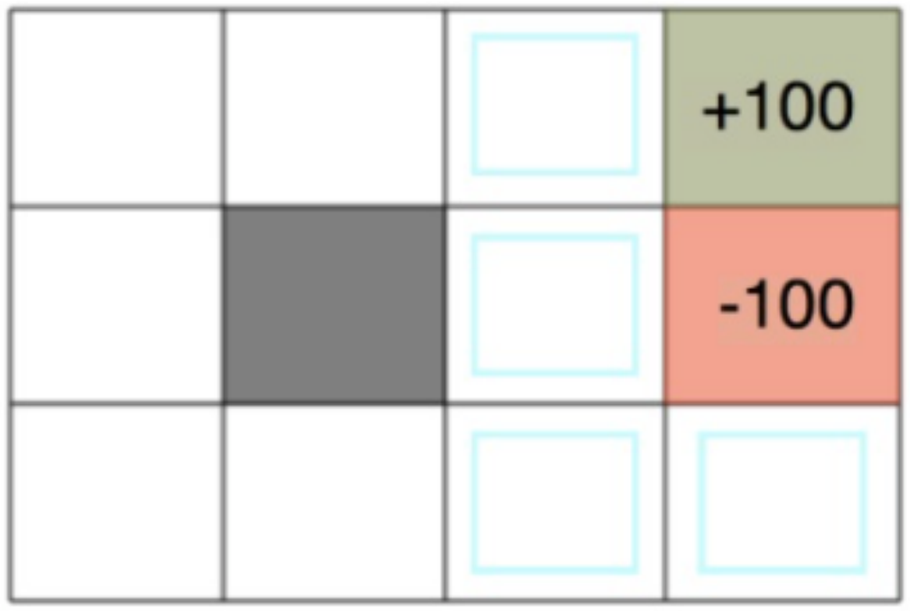
(for all states except absorbing states)

Policy examples in grid world



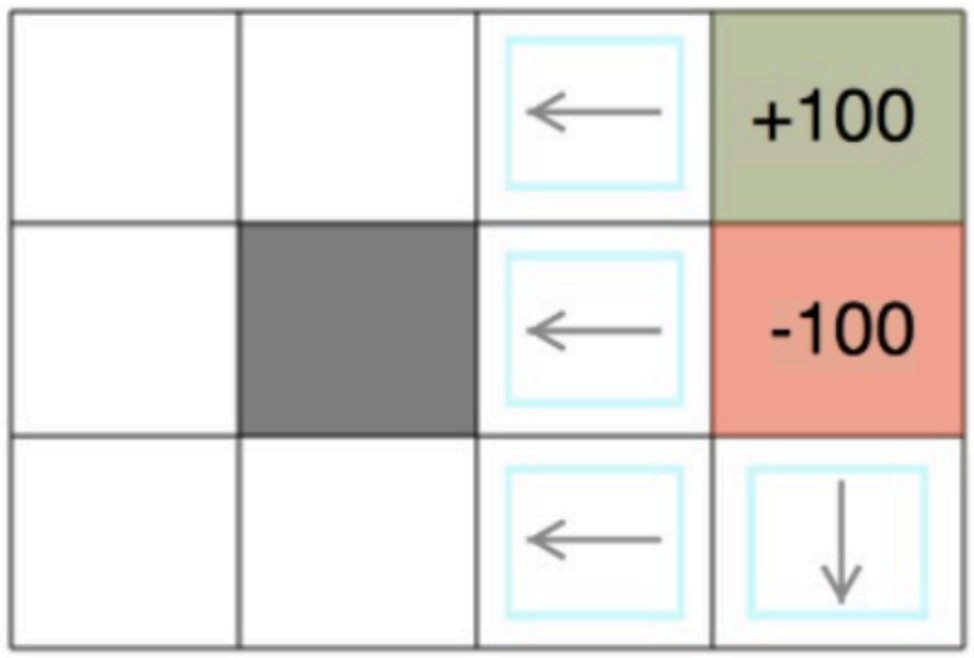
What is the optimal policy if $R(s) = -5$?

Policy examples in grid world



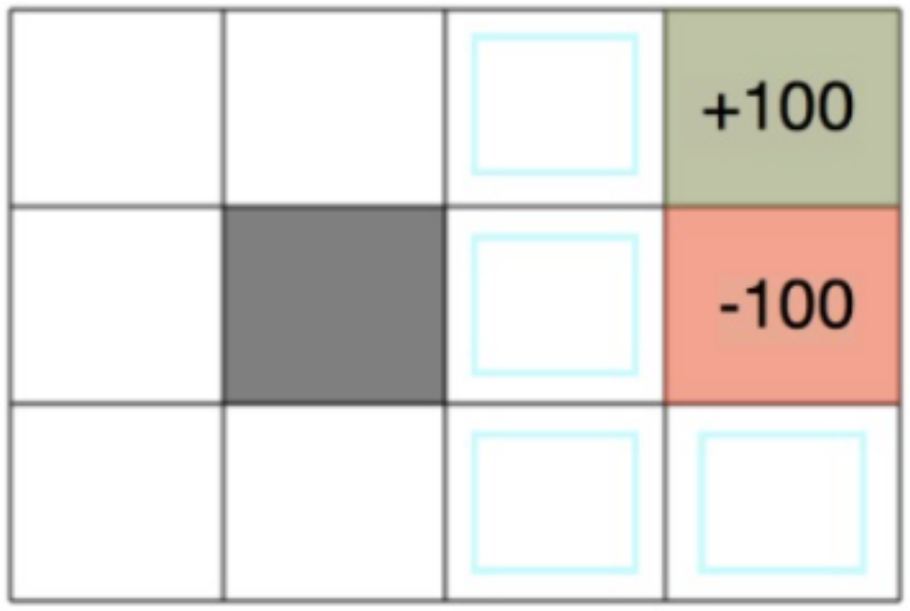
What is the optimal policy if $R(s) = 200$?

Policy examples in grid world



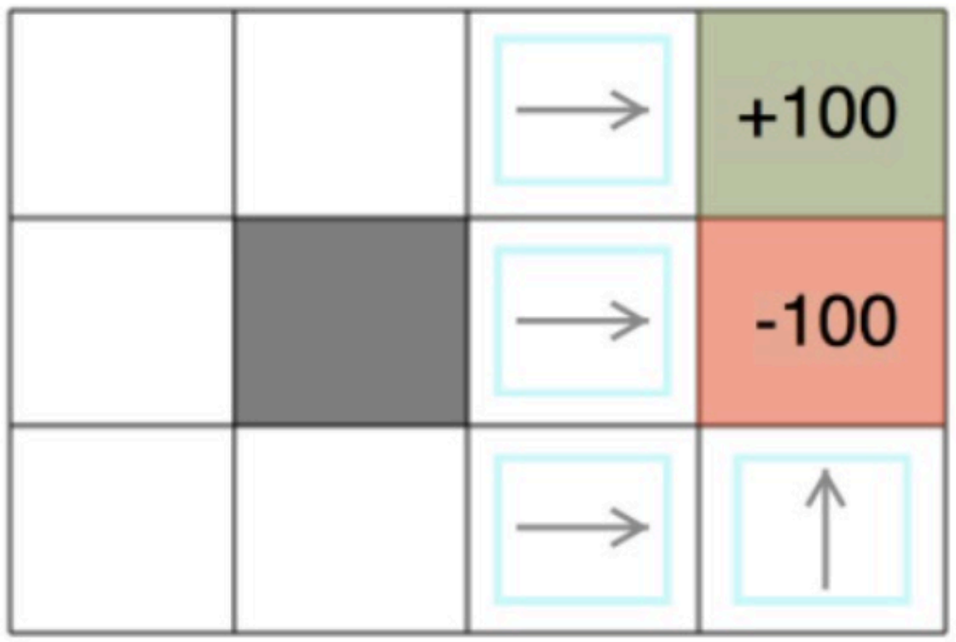
What is the optimal policy if $R(s) = 200$?

A concrete grid example



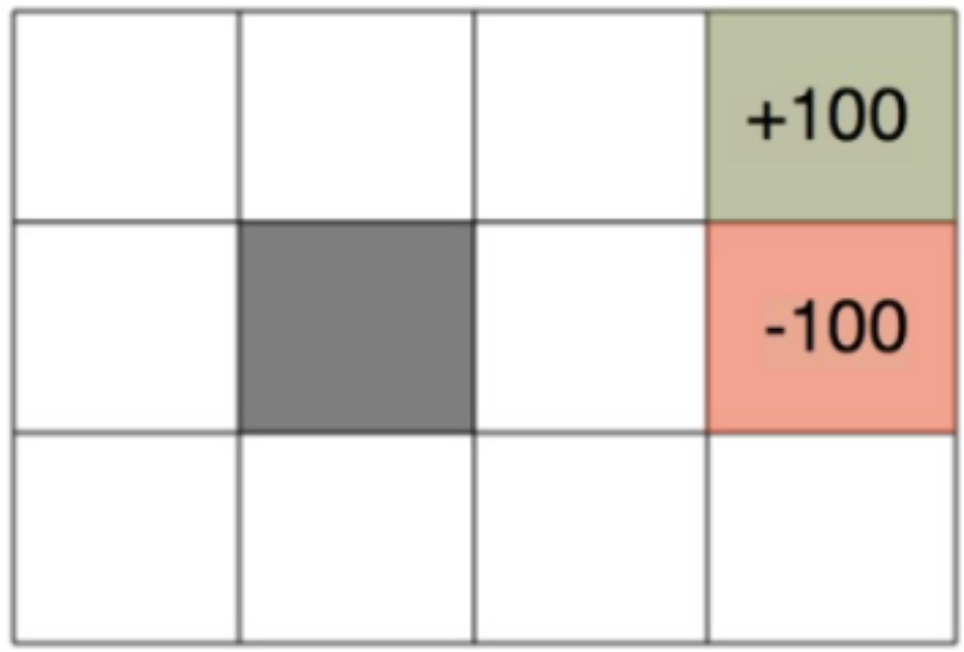
What is the optimal policy if $R(s) = -200$?

Policy examples in grid world



What is the optimal policy if $R(s) = -200$?

Policy examples in grid world



Take-Away: The optimal policy is highly dependent on the details of the rewards.

Value function

For each possible policy π the agent might adopt, we can define an evaluation function over states

Value function: $V^{\pi}(s) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$

$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

where $r_t, r_{t+1}, \ldots$ are from following policy π starting at state s

How to learn V?

We might try to have agent learn the evaluation function V^{π^*} (which we write as V^*)
It could then do a lookahead search to choose best action from any state s because

$$\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

A problem:

- This works well if agent knows $\delta : S \times A \rightarrow S$, and $r : S \times A \rightarrow \Re$
- But when it doesn't, it can't choose actions this way

Q-value function

Define new function very similar to V^*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

If agent learns Q, it can choose optimal action even without knowing δ!

$$\pi^*(s) = \arg\max_a [r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \arg\max_a Q(s,a)$$

Q is the evaluation function the agent will learn

Q-value function

Q-value function gives expected total reward

- from state s and action a
- under policy π
- with discount factor γ (future rewards mean less than immediate)

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s,a\right]$$

How to learn Q

$$V^*(s) = \max_{a'} Q(s, a')$$

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

Nice! Let $\hat{Q}$ denote learner's current approximation to Q. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

where s' is the state resulting from applying action a in state s

For each s, a initialize table entry $\hat{Q}(s,a) \leftarrow 0$

Observe current state s

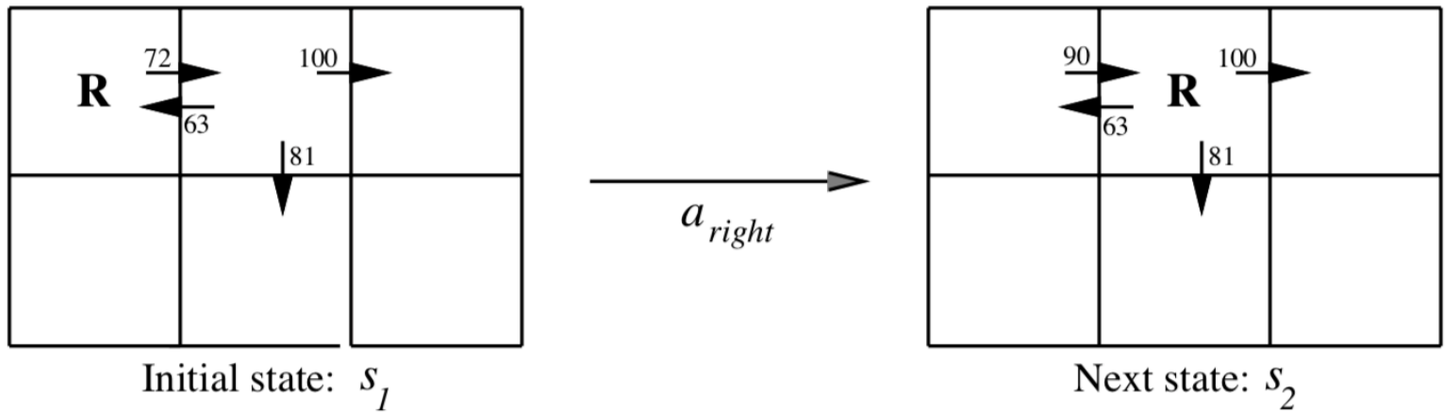
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s,a)$ as follows:

$$\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- $s \leftarrow s'$

Example:



$$\hat{Q}(s_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a')$$

$$\leftarrow 0 + 0.9 \max\{63, 81, 100\} = 90$$

if rewards non-negative, then

$$(\forall s, a, n) \;\; \hat{Q}_{n+1}(s, a) \geq \hat{Q}_n(s, a)$$

and

$$(\forall s, a, n) \;\; 0 \leq \hat{Q}_n(s, a) \leq Q(s, a)$$

$\hat{Q}$ converges to Q.

Learning Q allows one to learn the optimal policy

An optimal value function is the maximum achievable value

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a) = Q^{\pi^*}(s,a)$$

If you know the value function, you can derive policy

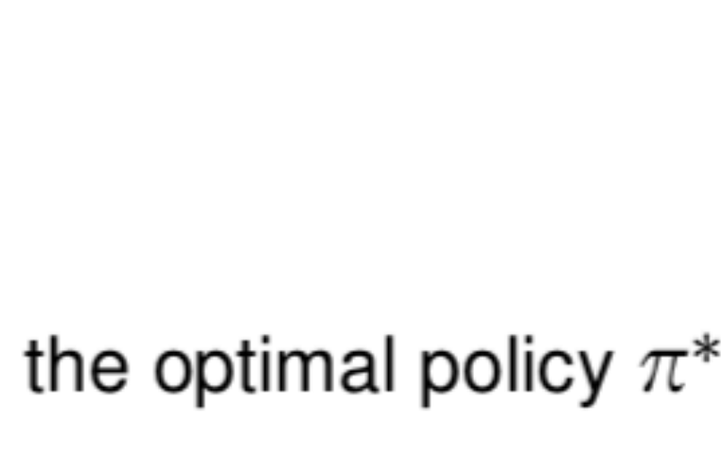
$$\pi^* = \arg\max_{a} Q(s,a)$$

Q-learning:

the task is to learn the optimal policy π^*

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

- $r(s,a)$ (immediate reward) values



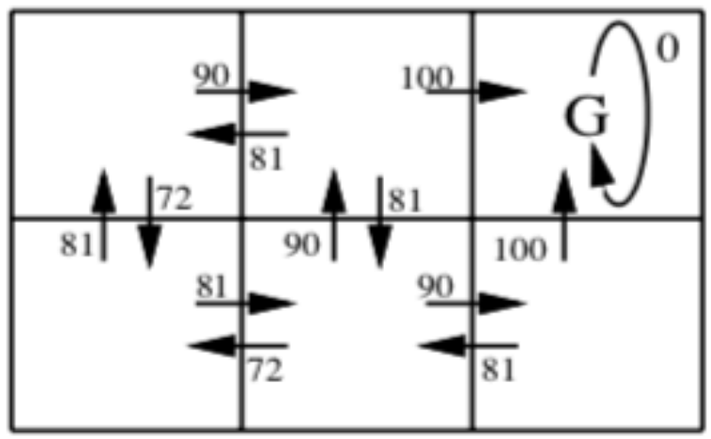
- $Q(s,a)$ values
- One optimal policy

Q-learning:

the task is to learn the optimal policy π^*

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

- $Q(s, a)$ values



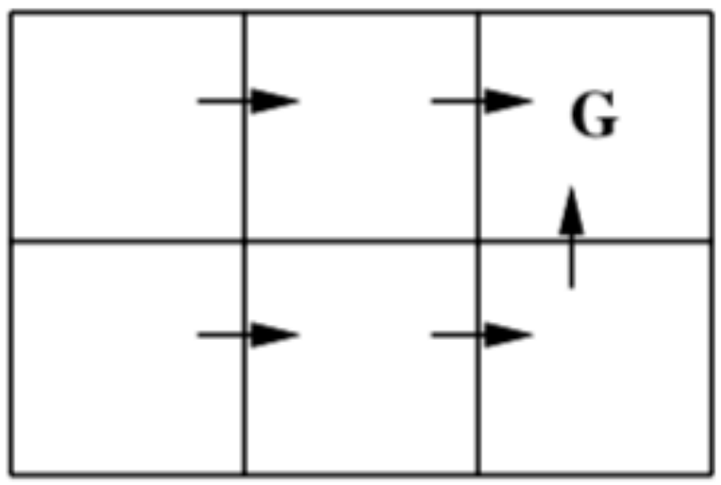
- One optimal policy

Q-learning:

the task is to learn the optimal policy π^*

$$\pi^* \equiv \arg\max_{\pi} V^{\pi}(s), (\forall s)$$

- $r(s,a)$ (immediate reward) values
- $Q(s,a)$ values
- One optimal policy



Q-learning: non-deterministic case

What if reward and next state are non-deterministic?

We redefine V, Q by taking expected values

$$V^{\pi}(s) \equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

$$\equiv E[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$$

$$Q(s,a) \equiv E[r(s,a) + \gamma V^*(\delta(s,a))]$$

Q-learning: non-deterministic case

Alter training rule to

$$\hat{Q}_n(s,a) \leftarrow (1-\alpha_n)\hat{Q}_{n-1}(s,a) + \alpha_n[r + \max_{a'} \hat{Q}_{n-1}(s',a')]$$

where

$$\alpha_n = \frac{1}{1+\text{visits}_n(s,a)}$$

Can still prove convergence of $\hat{Q}$ to Q [Watkins and Dayan, 1992]

Bonus slides

Value Iteration

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right] \\
&\leq \sum_{k=0}^{\infty} \gamma^k R_{max} = \frac{R_{max}}{1-\gamma}
\end{aligned}
$$

Take-away: Discounted reward makes an infinite-horizon value-function **finite**. This is great, because it allows us to compare the values of different sequences.

Value Iteration

Optimal value function can be used to define π_*

$$\pi_*(s) = \arg\max_a \sum_{s'} p(s' \mid s, a)\, v_*(s')$$

Value Iteration

The Bellman optimality equation

$$v_*(s) = \max_a \mathbb{E}_{\pi_*} \left[G_t \mid S_t = s, A_t = a \right]$$

Value Iteration

The Bellman optimality equation

$$
\begin{aligned}
v_*(s) &= \max_a \mathbb{E}_{\pi_*} \left[G_t \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi_*} \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E} \left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a \right] \\
&= \max_a \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right]
\end{aligned}
$$

Value Iteration

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

Suppose there are N total states in the system

Bellman's equation is really N equations in N unknowns

Should be easy!

Value Iteration

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

Suppose there are N total states in the system

Bellman's equation is really N equations in N unknowns

Should be easy!

Except the $\max$ makes the system nonlinear

Nonlinear + Recursive definition ...

Need an **iterative** solution

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$

[Sutton&Barto 2017]

Value Iteration

$$v_*(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')]$$

Iterative Idea:

1. Start with arbitrary values for $v(s)$
2. Update v's based on neighbors
3. Repeat until values don't change anymore

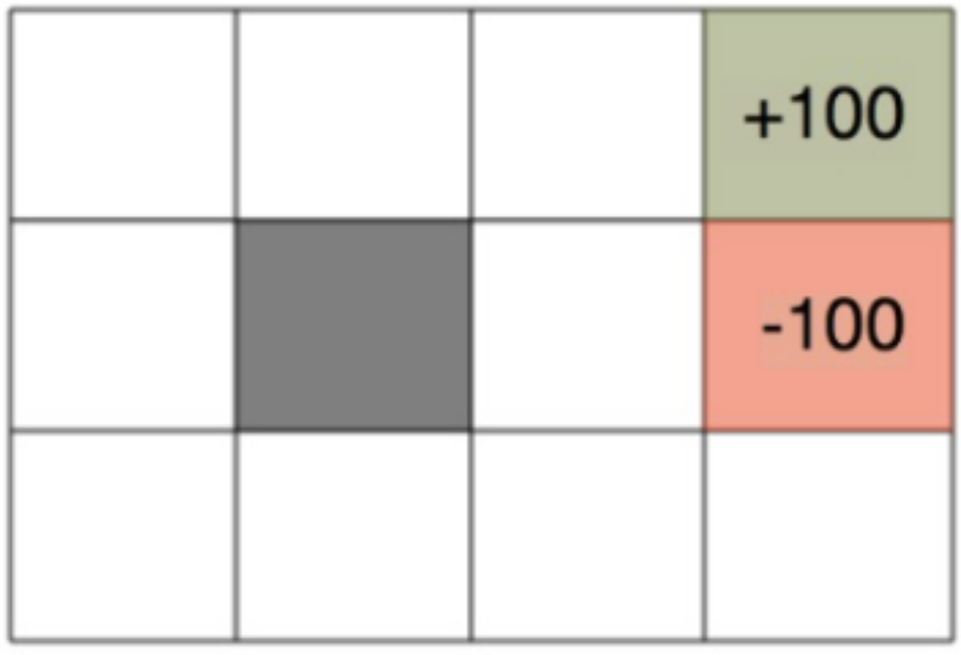
Define the *approximation* of the value at iteration k as $\hat{v}_k(s)$

Update Rule:

$$\hat{v}_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$$

Value Iteration

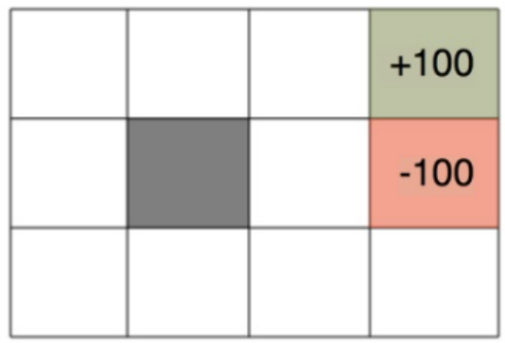
$$\hat{v}_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$$



Assume $\gamma = \frac{1}{2}$, $\hat{v}_0(s) = 0$, $R(s) = -5$ (except absorbing states),
$P(\text{action}) = 0.8, P(\text{left angle}) = 0.1, P(\text{right angle}) = 0.1$
Question: What is $\hat{v}_1(3,3)$?

Value Iteration

$$\hat{v}_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$$

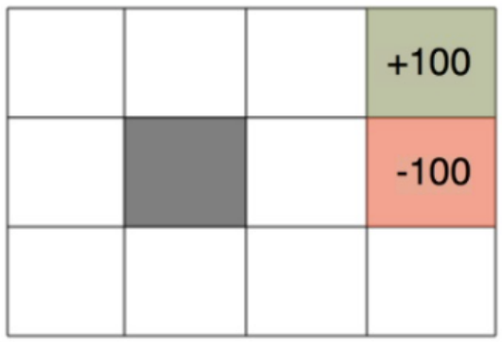


Assume $\gamma = \frac{1}{2}$, $V_0(s) = 0$, $R(s) = -5$ (except absorbing states),
$P(\text{action}) = 0.8, P(\text{left angle}) = 0.1, P(\text{right angle}) = 0.1$

$$\hat{v}_1(3,3) = \max \left\{ \begin{array}{l} U: \\ D: \\ L: \\ R: \end{array} \right\}$$

Value Iteration

$$\hat{v}_{k+1}(s) = \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')]$$



Assume $\gamma = \frac{1}{2}$, $V_0(s) = 0$, $R(s) = -5$ (except absorbing states),
$P(\text{action}) = 0.8, P(\text{left angle}) = 0.1, P(\text{right angle}) = 0.1$

$$\hat{v}_1(3,3) = \max \left\{ \begin{array}{l} U : 0.8 * (-5 + \frac{1}{2}\hat{v}_0(3,3)) + 0.1 * (100 + \frac{1}{2}\hat{v}_0(4,3)) \\ +0.1 * (-5 + \frac{1}{2}\hat{v}_0(2,3)) \\ D : 0.8 * (-5 + \frac{1}{2}\hat{v}_0(3,2)) + 0.1 * (100 + \frac{1}{2}\hat{v}_0(4,3)) \\ +0.1 * (-5 + \frac{1}{2}\hat{v}_0(2,3)) \\ L : 0.8 * (-5 + \frac{1}{2}\hat{v}_0(2,3)) + 0.1 * (100 + \frac{1}{2}\hat{v}_0(3,2)) \\ +0.1 * (-5 + \frac{1}{2}\hat{v}_0(3,3)) \\ R : 0.8 * (100 + \frac{1}{2}\hat{v}_0(4,3)) + 0.1 * (-5 + \frac{1}{2}\hat{v}_0(3,3)) \\ +0.1 * (-5 + \frac{1}{2}\hat{v}_0(3,2)) \end{array} \right\}$$

Value Iteration

- Guaranteed to converge to the optimal v_*
- $\Rightarrow$ optimal π^*
- $\hat{v}_k$ might converge very slowly to v_*
- $\Rightarrow$ but quickly to π^*

Value Iteration

- Guaranteed to converge to the optimal v_*
- $\Rightarrow$ optimal π^*
- $\hat{v}_k$ might converge very slowly to v_*
- $\Rightarrow$ but quickly to π^*

So far we've assumed complete knowledge of the MDP
Haven't really had to **learn** anything yet ...
Next time: Q-learning, more ideas about possible RF methods

Goal

- Episode: ending at a terminal state, e.g., a play/move/round of a game
- Continuous task: keep trying for an infinite num. steps
- G_t: the expected discounted return:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$T = \infty, \gamma < 1$$

When looking at episodes, we can also have finite T and $\gamma = 1$

Value function

The value function is defined as the expected value at a given state s and policy π.
The value function estimates how good it is to be in a given state.

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$
$$= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

Action-value function (Q-function)

The action-value function is defined as the expected value of taking action a at a given state s and policy π.

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right]$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right]$$

Optimal policy and optimal value function

A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states.
$\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$

Optimal policy and optimal value function

A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states.
$\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$

Optimal state-value function:

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

Optimal action-value function:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right]$$

Optimal policy and optimal value function

A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states.
$\pi \geq \pi'$ iff $v_\pi(s) \geq v_{\pi'}(s) \forall s \in \mathcal{S}$

Optimal state-value function:

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

Optimal action-value function:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a)$$

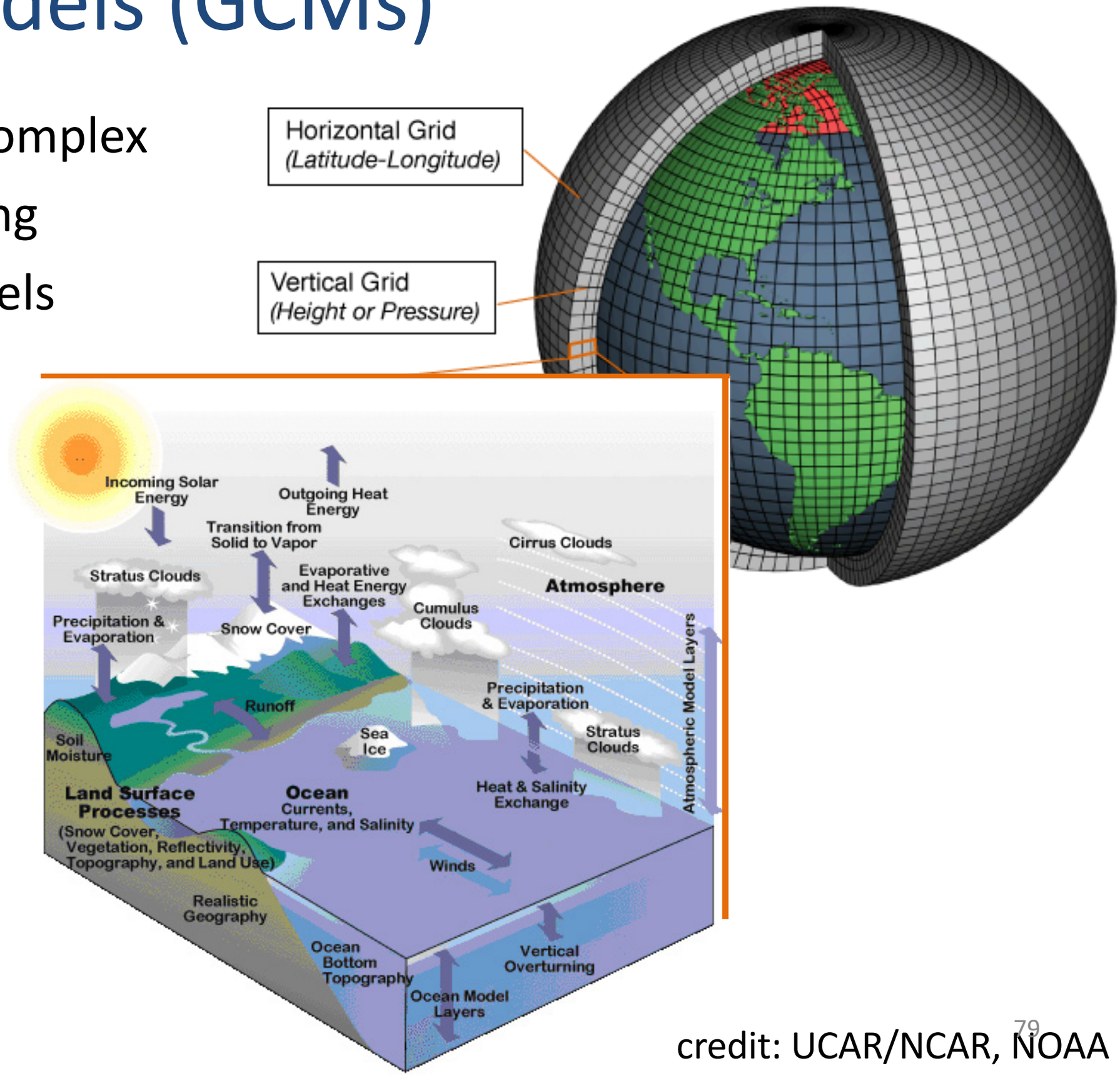
Bonus slides

Application to Climate Model Ensembles

Climate models (GCMs)

Climate model: a complex system of interacting mathematical models

- Not data-driven
- Based on scientific first principles
  - Meteorology
  - Oceanography
  - Geophysics
  - …
- Discretization into grid boxes
- Scale resolution differences



credit: UCAR/NCAR, NOAA

Intergovernmental Panel on Climate Change

- IPCC: Intergovernmental Panel on Climate Change
  - Nobel Peace Prize 2007 (shared with Al Gore).
  - Interdisciplinary scientific body, formed by UN in 1988.
  - Fourth Assessment Report, 2007, on global climate change
    450 lead authors from 130 countries, 800 contributing authors, over 2,500 reviewers.
  - Fifth Assessment Report, September 2013. Over 830 authors.

- Climate models contributing to IPCC reports include:

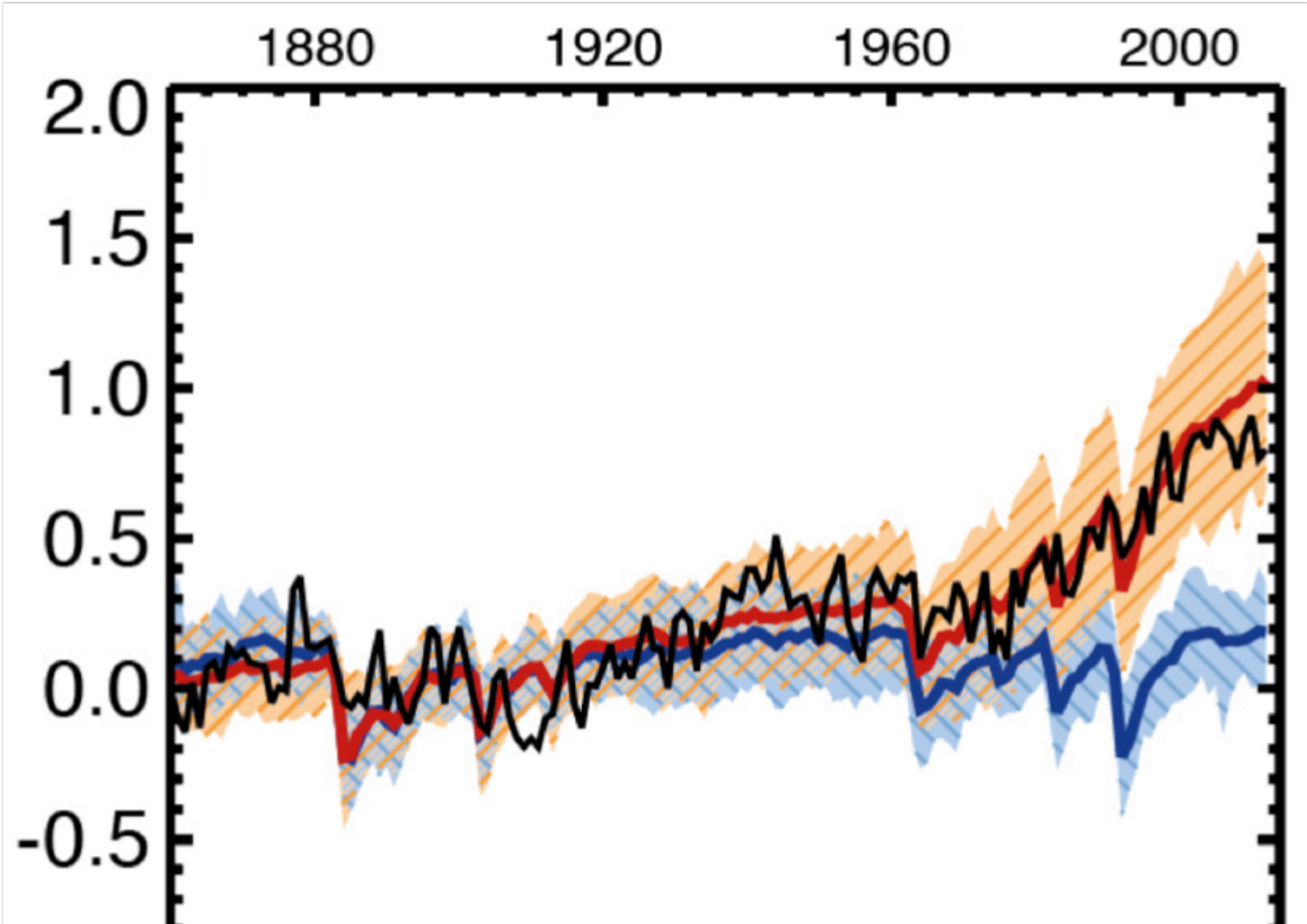
  Bjerknes Center for Climate Research (Norway), Canadian Centre for Climate Modelling and Analysis, Centre National de Recherches Météorologiques (France), Commonwealth Scientific and Industrial Research Organisation (Australia), Geophysical Fluid Dynamics Laboratory (Princeton University), Goddard Institute for Space Studies (NASA), Hadley Centre for Climate Change (United Kingdom Meteorology Office), Institute of Atmospheric Physics (Chinese Academy of Sciences), Institute of Numerical Mathematics Climate Model (Russian Academy of Sciences), Istituto Nazionale di Geofisica e Vulcanologia (Italy), Max Planck Institute (Germany), Meteorological Institute at the University of Bonn (Germany), Meteorological Research Institute (Japan), Model for Interdisciplinary Research on Climate (Japan), National Center for Atmospheric Research (Colorado), among others.

IPCC findings: human influence on climate

Black: true observations.

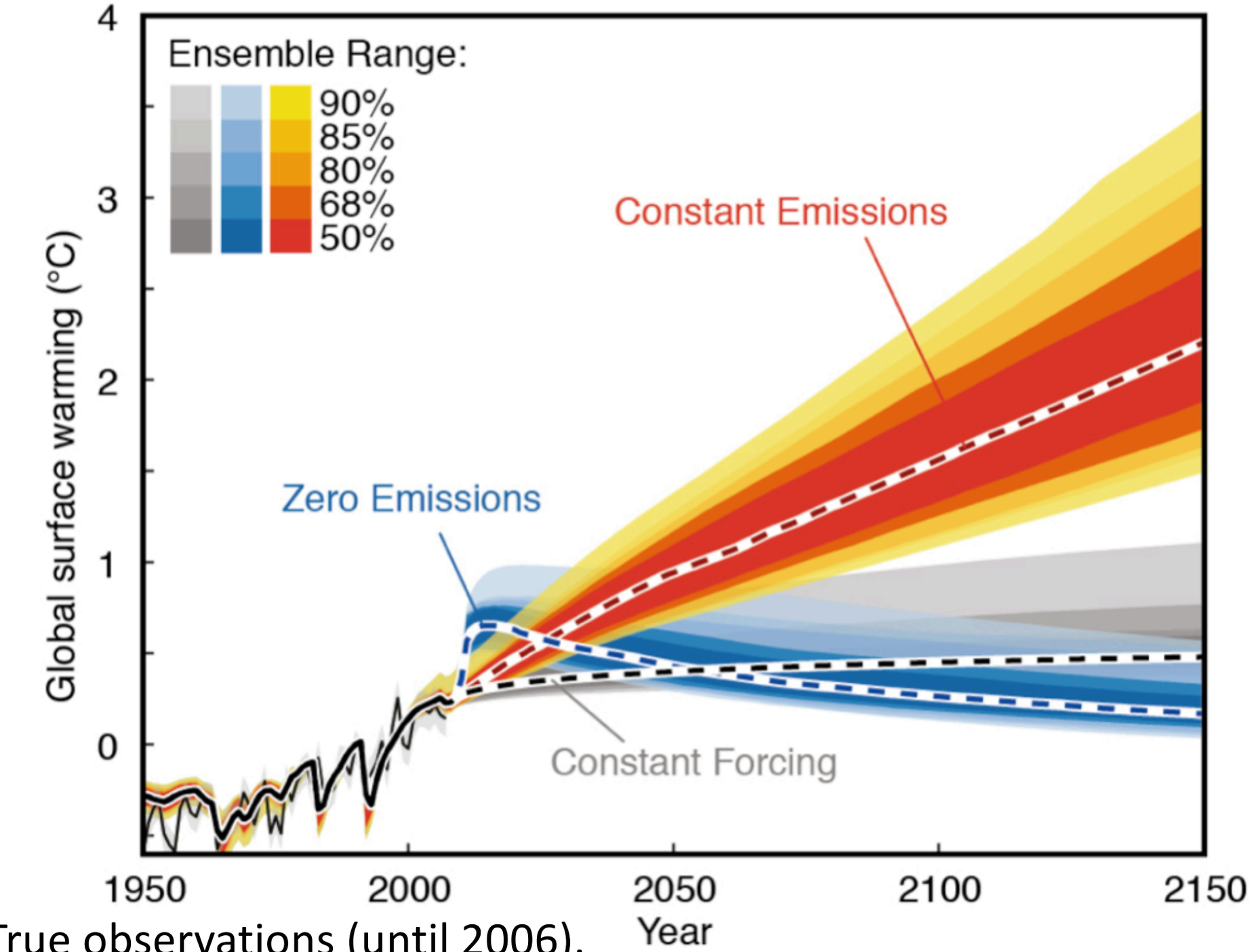
Orange/red: Climate model simulations with human-induced greenhouse gasses.

Blue: Climate model simulations *without* human-induced greenhouse gasses.



IPCC 2013

Modeling future scenarios



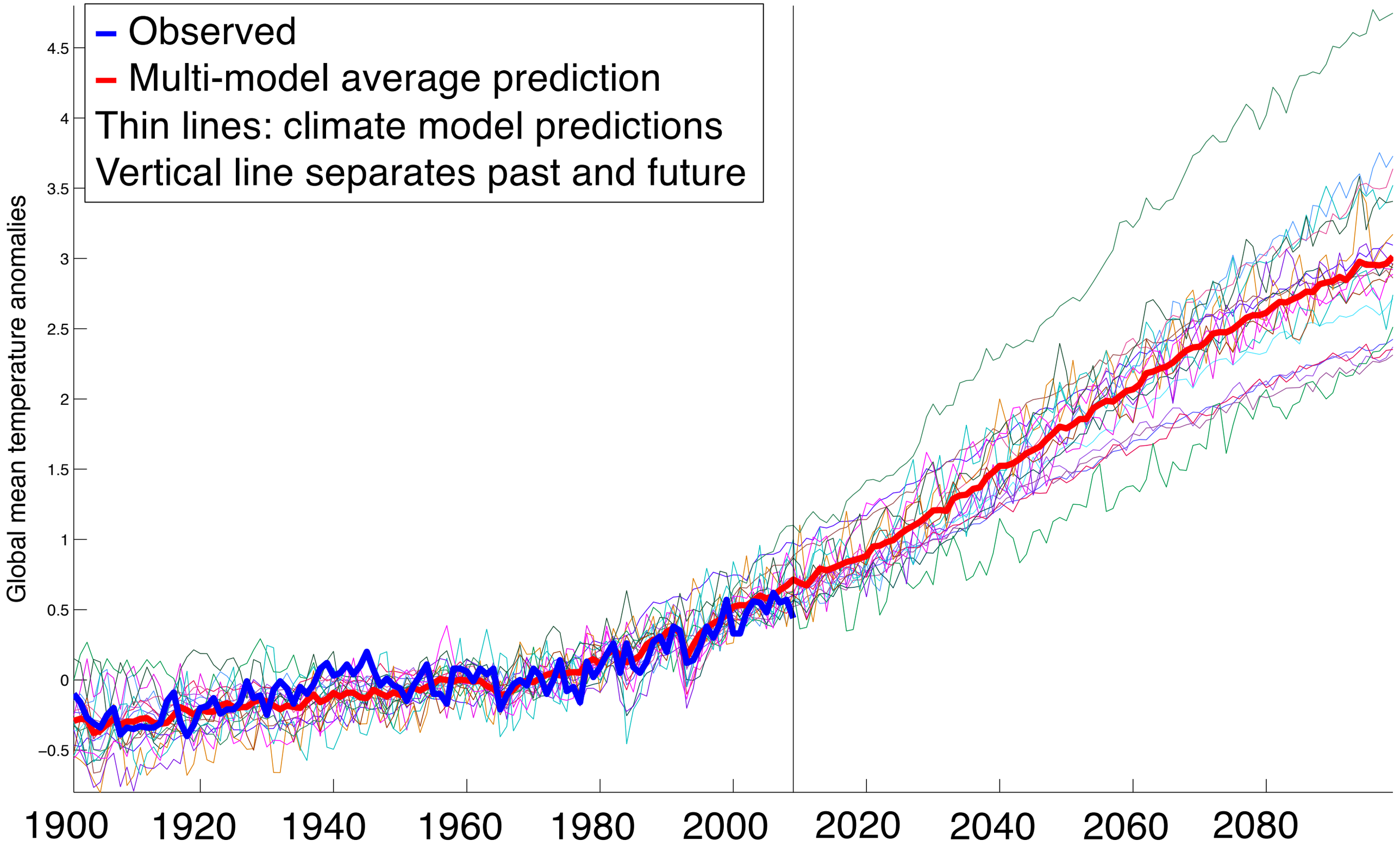
Black: True observations (until 2006).

Orange/red: Constant emissions.

Grey: Constant atmospheric composition (constant forcing).

Blue: Zero emissions starting 2010 (impossible).

credit: IPCC 2013



Future fan-out.

Improving predictions of the IPCC ensemble

- Coupled Model Intercomparison Project (CMIP)

  [Meehl et al., Bull. AMS, '00]
- No one model predicts best all the time, for all variables.
- Average prediction over all models is better predictor than any single model. [Reichler & Kim, Bull. AMS '08], [Reifen & Toumi, GRL '09]
- Bayesian approaches in climate science e.g. [Smith et al. JASA '08]
- IPCC held 2010 Expert Meeting on how to better combine model predictions.

Can we do better, using Machine Learning?

Challenge: How should we predict future climates?

  – While taking into account the multi-model ensemble predictions

Contributions

- Tracking Climate Models (TCM) [M, Schmidt, Saroha, & Asplund, SAM 2011; NASA CIDU 2010]: Online learning with expert advice.

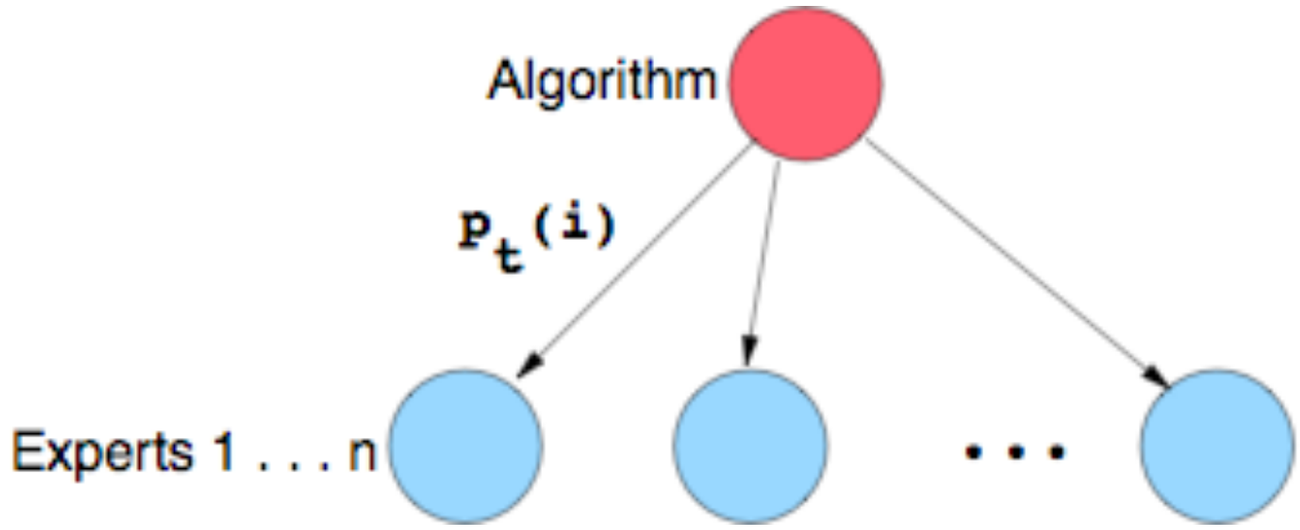
- Neighborhood-Augmented TCM (NTCM) [McQuade & M, AAAI 2012]: Extend TCM to model geospatial neighborhood influence.

- MRF-based approach [McQuade & M, to appear, 2017].

- Climate Prediction via Matrix Completion [Ghafarianzadeh & M, Late-Breaking Paper, AAAI 2013]: use sparse matrix completion.

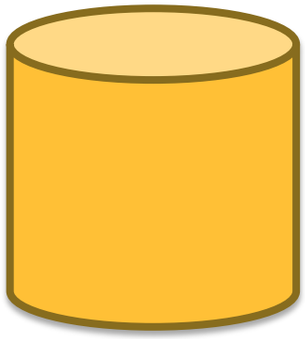
Online learning with expert advice

- Learner maintains distribution over *n* “experts.”

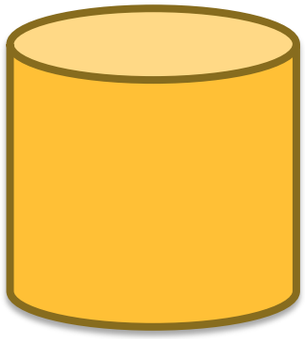


- Experts are black boxes: need not be good predictors, can vary with time, and depend on one another.
- Learner informs prediction using a probability distribution $p_t(i)$ over experts, *i*, depending on *L(i,t)*, loss of expert *i*’s prediction with respect to the observation at time *t*.
- Different algorithms to update $p_t(i)$ correspond to different modeling assumptions for time-varying nature of data stream.

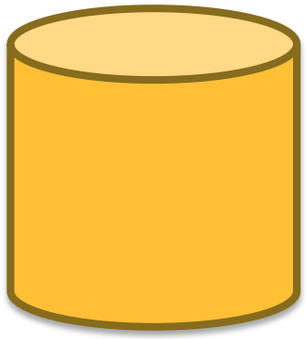
Average prediction



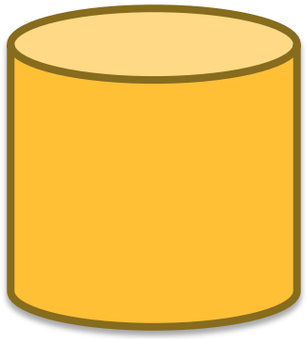
Model A



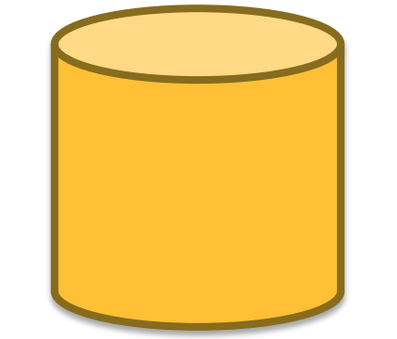
Model B



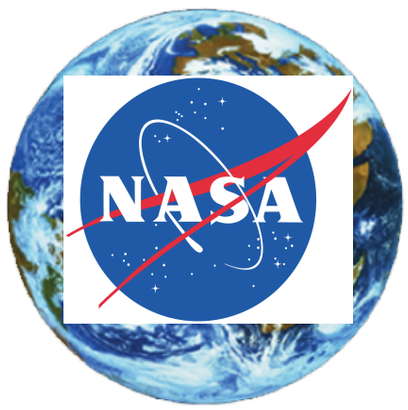
Model C



Model D

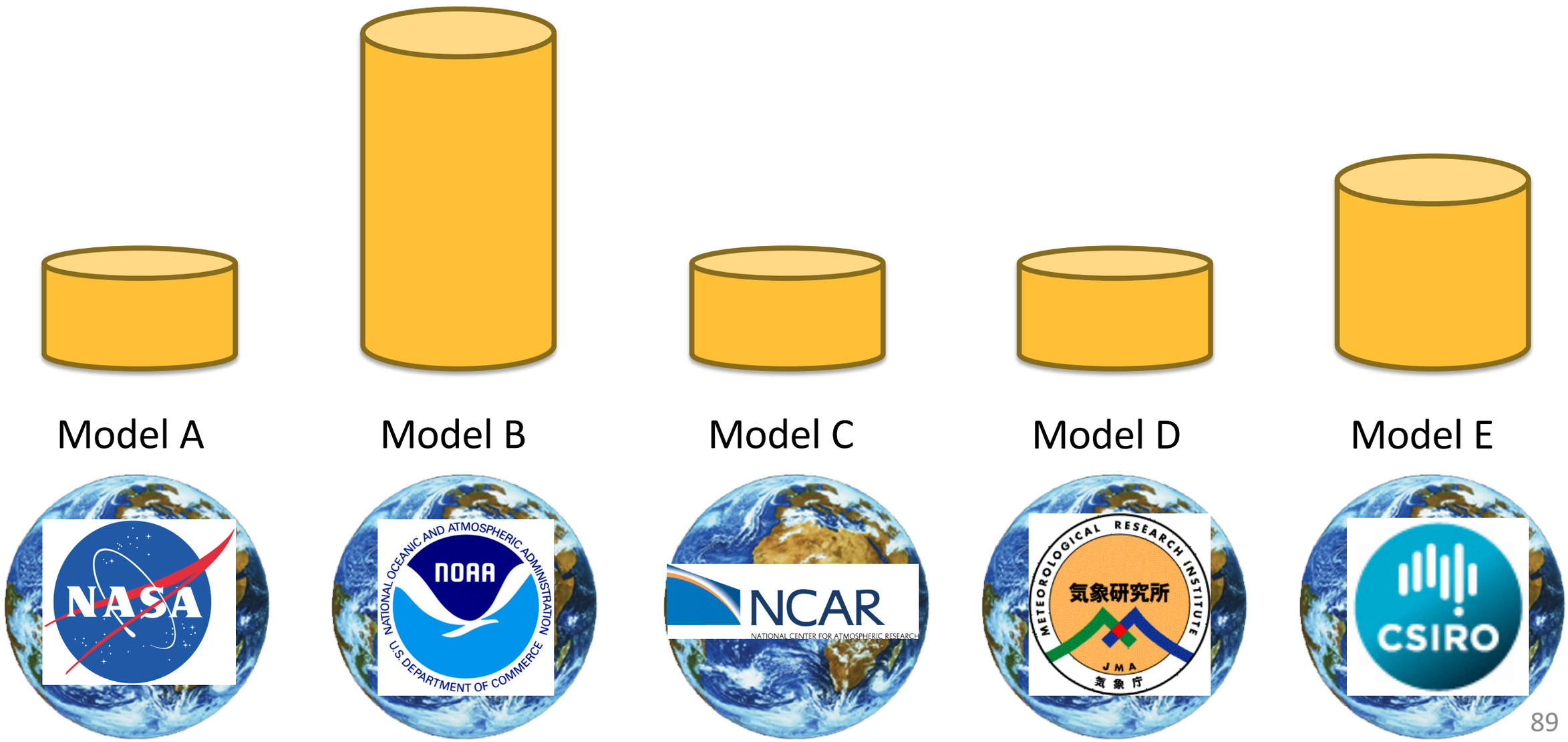


Model E

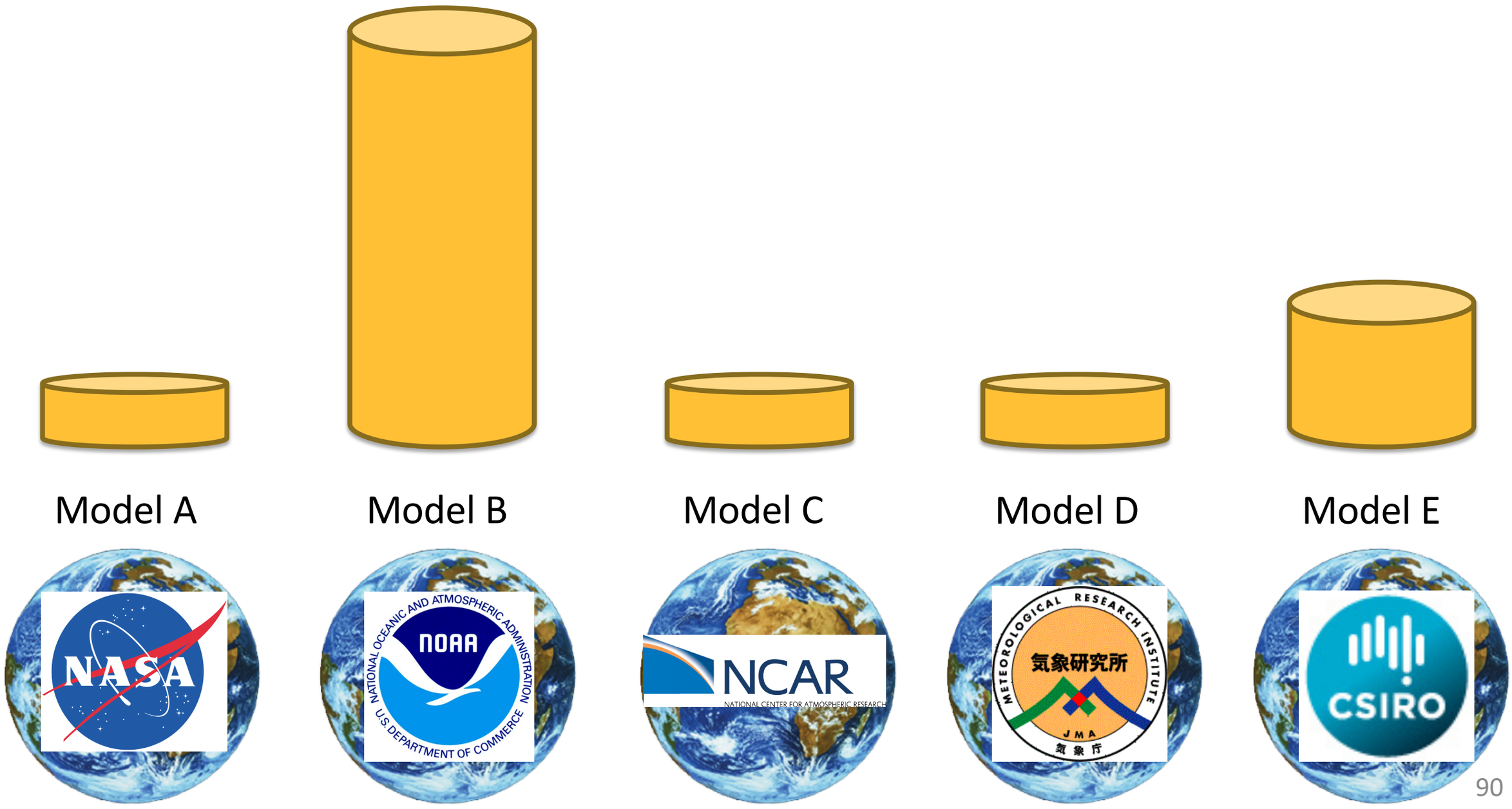


Adaptive, weighted average prediction

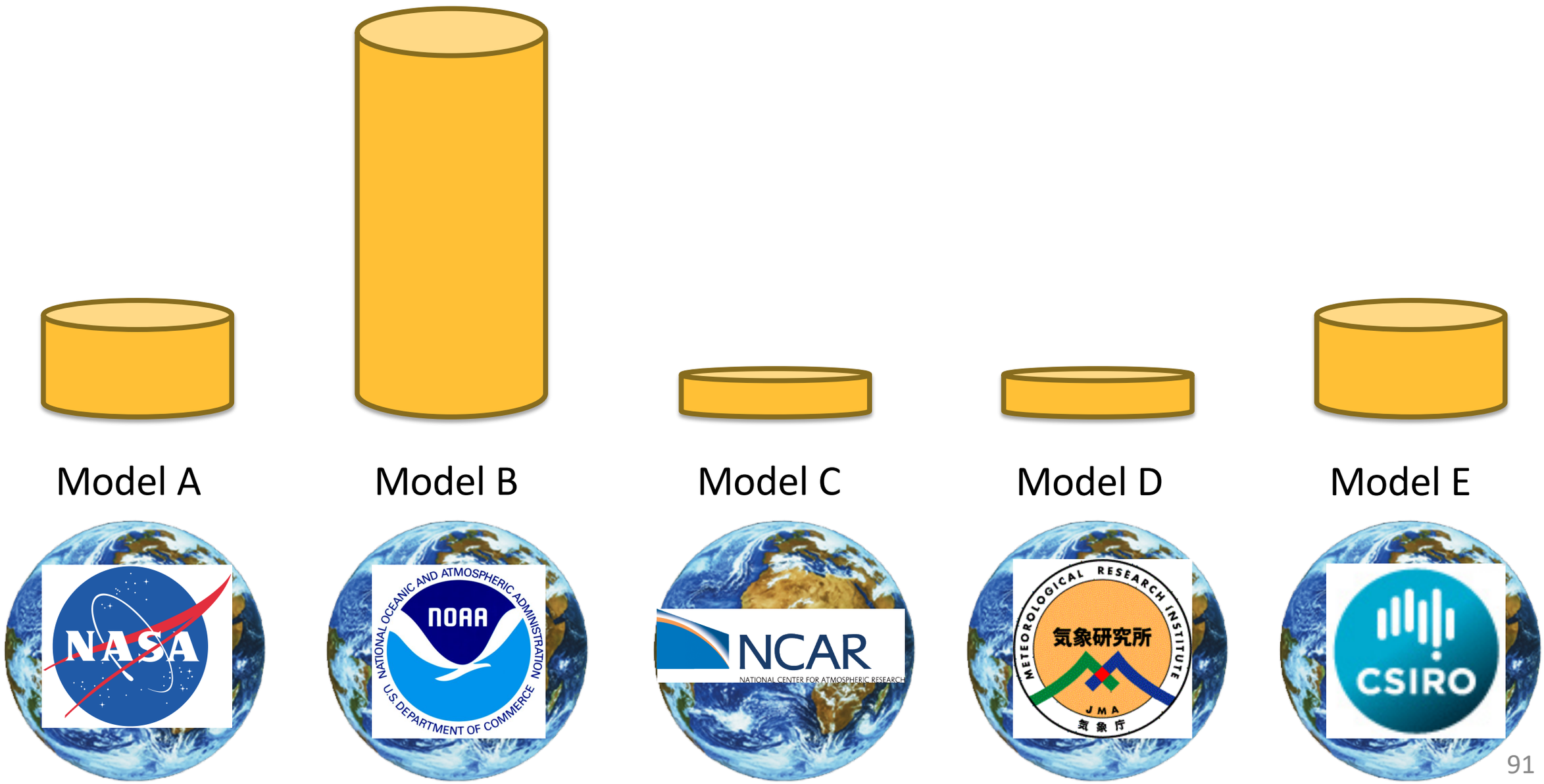
Adaptive, weighted average prediction



Adaptive, weighted average prediction



Adaptive, weighted average prediction

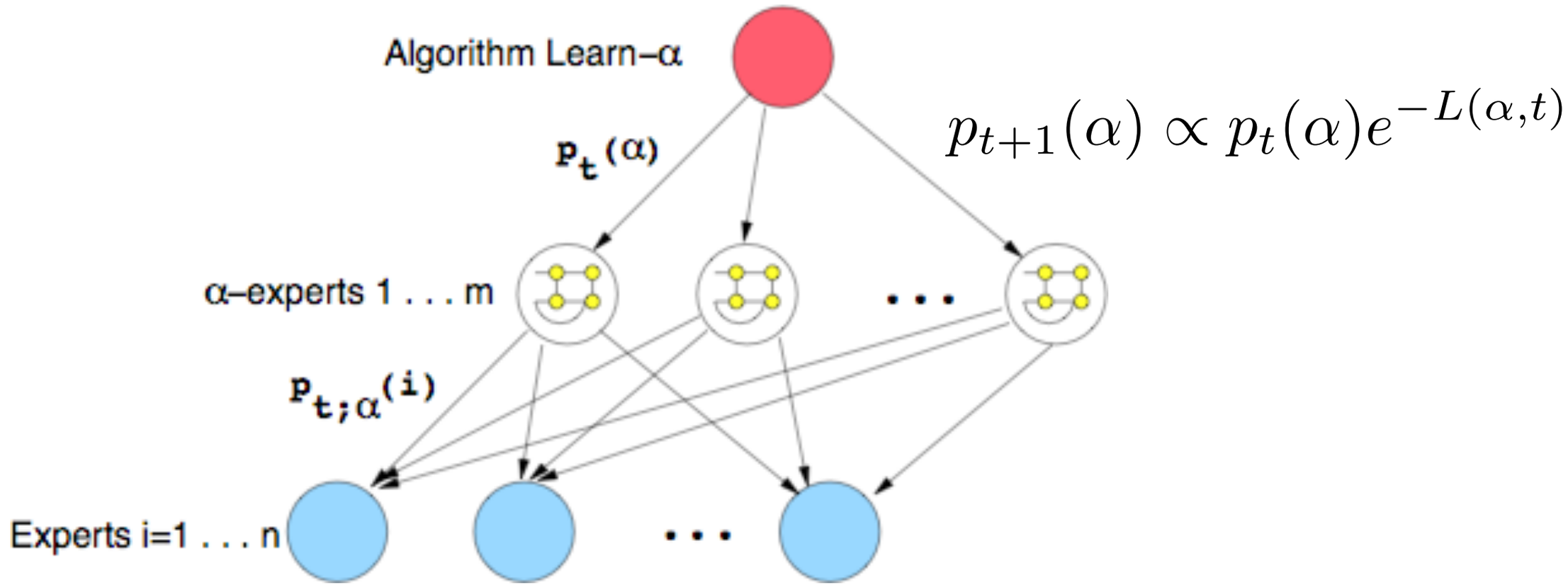


Tradeoff: explore vs. exploit

Tradeoff: Quickly finding current best predicting model vs. being ready to quickly switch to other models.

Tradeoff hinges on how often the identity of the best model switches.

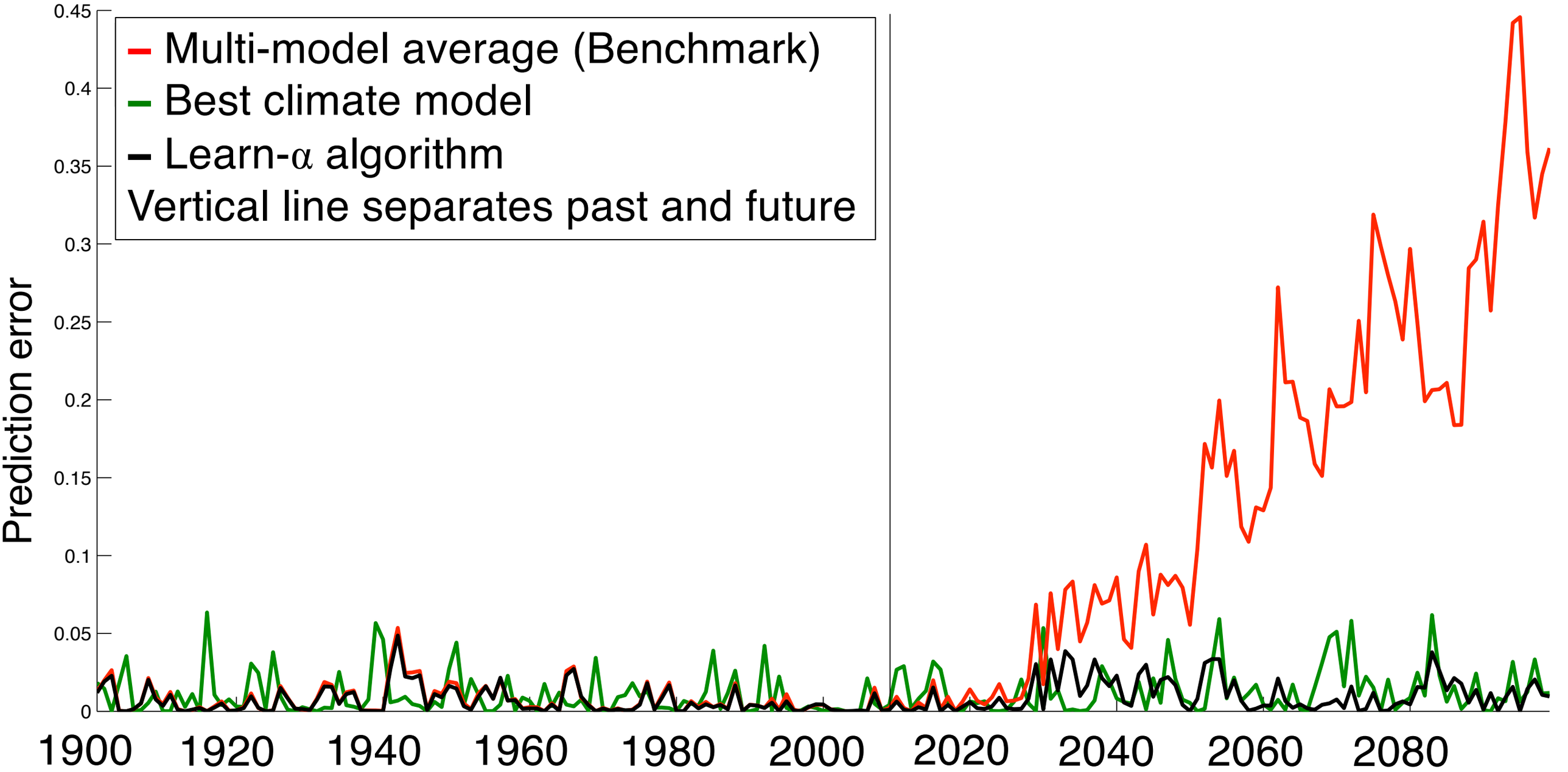
Online learning: non-stationary data



$$p_{t+1}(\alpha) \propto p_t(\alpha) e^{-L(\alpha,t)}$$

Learn-α Algorithm [M & Jaakkola, NIPS 2003]:

- Learns the switching rate: level of non-stationarity: α.
- Tracks a set of meta-experts, online learning algorithms, each with a different value of the α parameter.



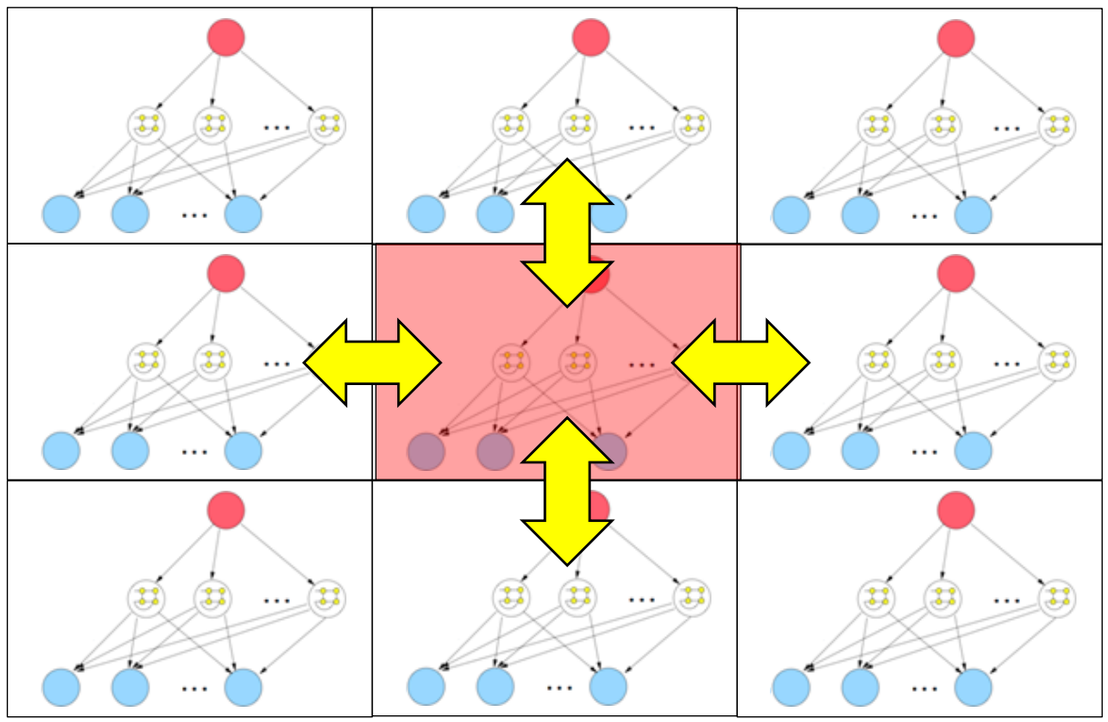
Learning curves

[M, Schmidt, Saroha, & Asplund, SAM 2011; NASA CIDU 2010]

Incorporating neighborhood influence

[McQuade & M, AAAI 2012]

- Climate predictions are made at higher geospatial resolutions.
- Run instances of Learn-α (variant) on multiple sub-regions that partition the globe.
- Model neighborhood influences among geospatial regions.



Incorporating neighborhood influence

Neighborhood-augmented Learn-α.

Non-homogenous HMM transition dynamics:

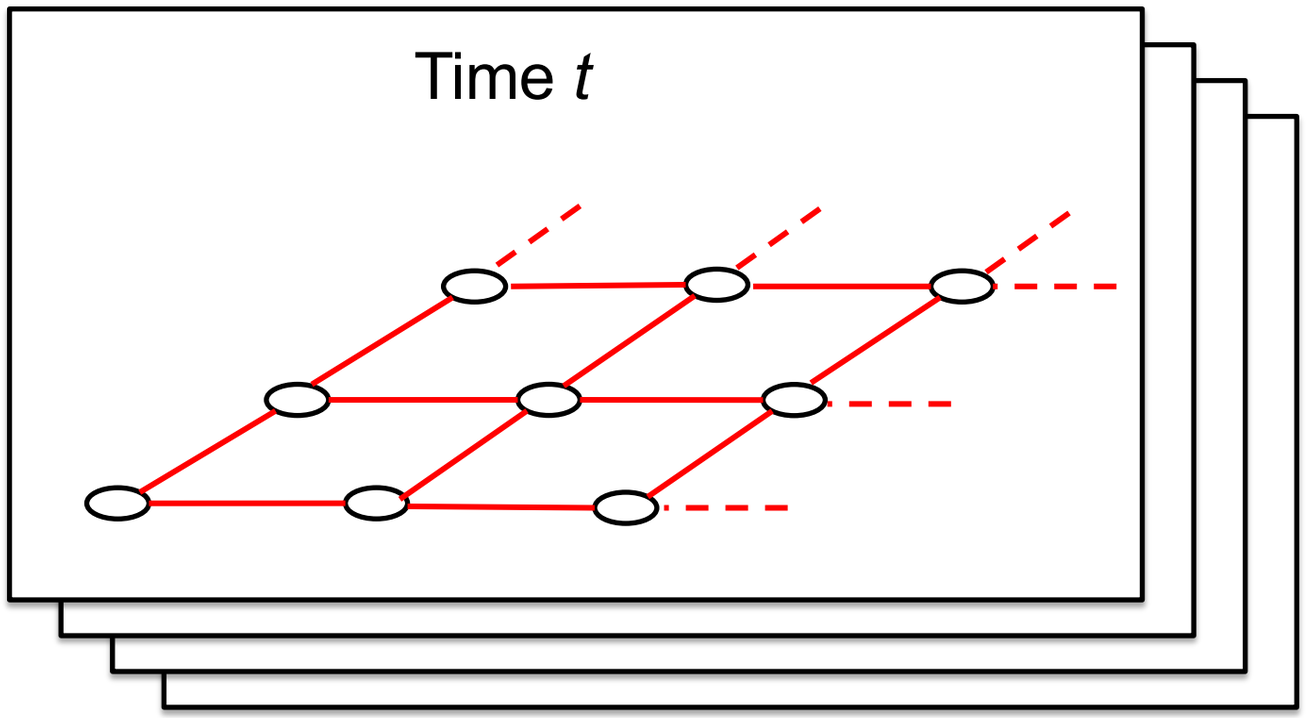
$$P(i \mid k;\alpha) = \begin{cases} (1-\alpha) & \text{if i=k} \\ \frac{\alpha}{Z}\left[(1-\beta) + \beta \frac{1}{|S(r)|} \sum_{s \in S(r)} P_{t,s}(i)\right] & \text{if i}\neq\text{k} \end{cases}$$

- *S(r)* - neighborhood scheme: set of "neighbors" of region *r*
- $P_{t,s}(i)$ - probability of expert (climate model) *i* in region *s*
- β - regulates geospatial influence
- *Z* - normalization factor

MRF-based approach

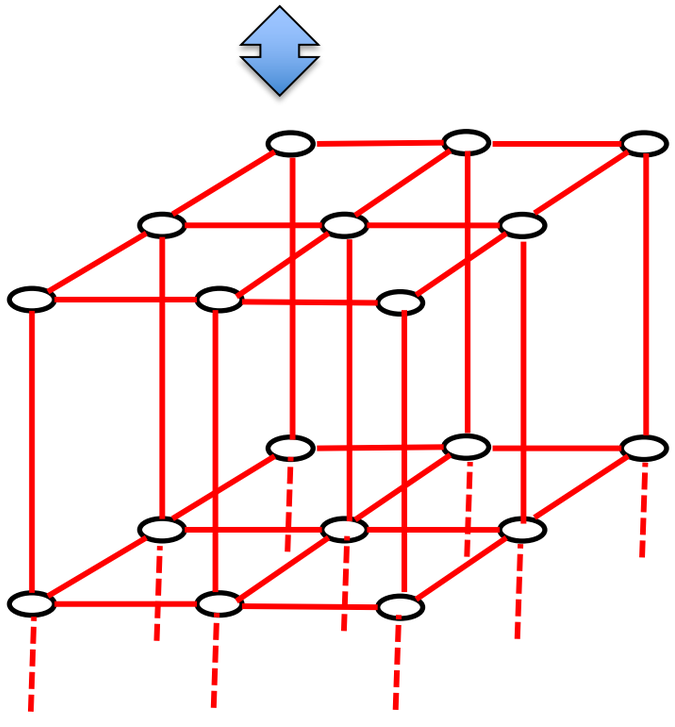
[McQuade & M, to appear, 2015]

Geospatial lattice

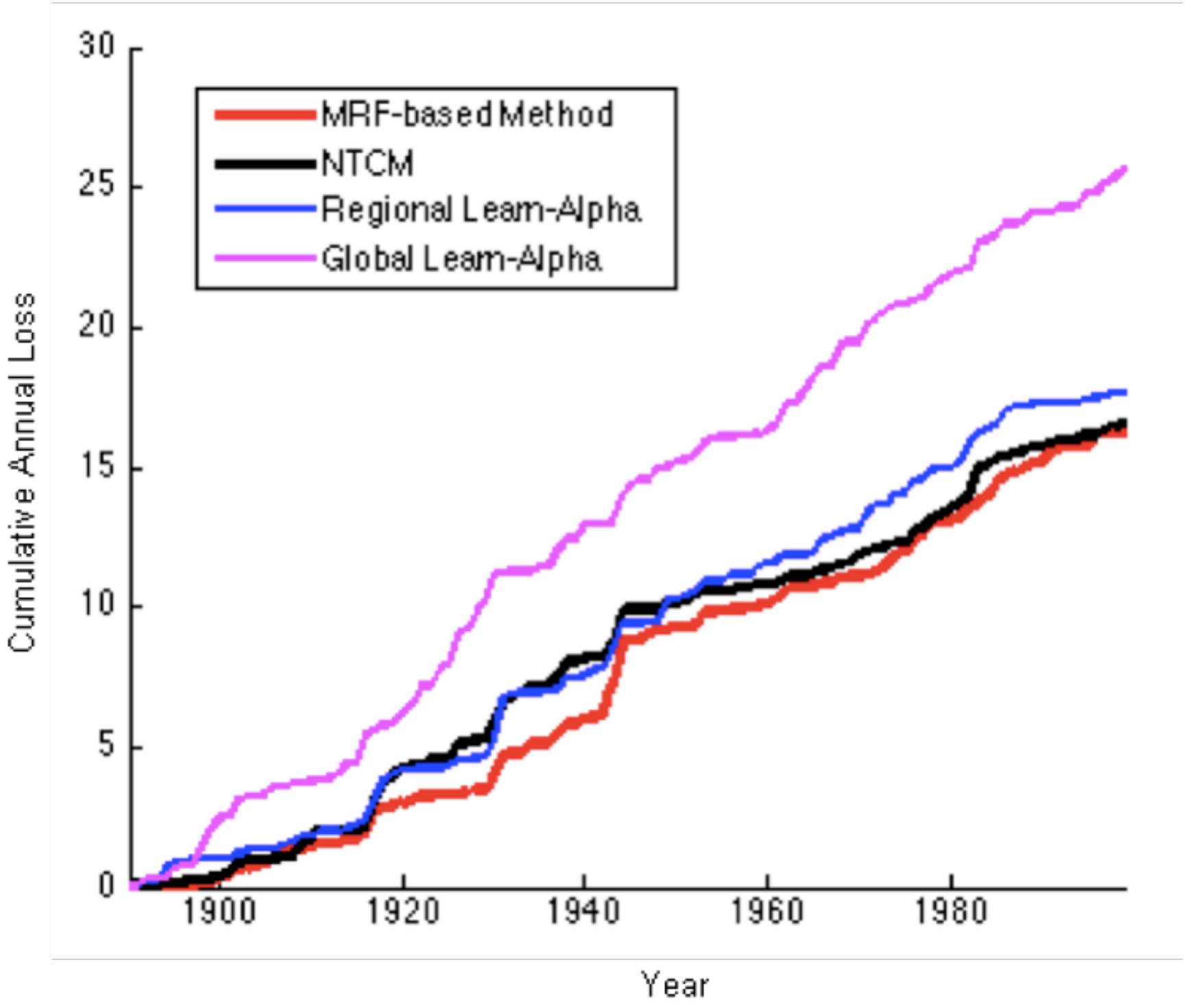


Time *t*

Time *t-1*



MRF-based approach



MRF-based approach

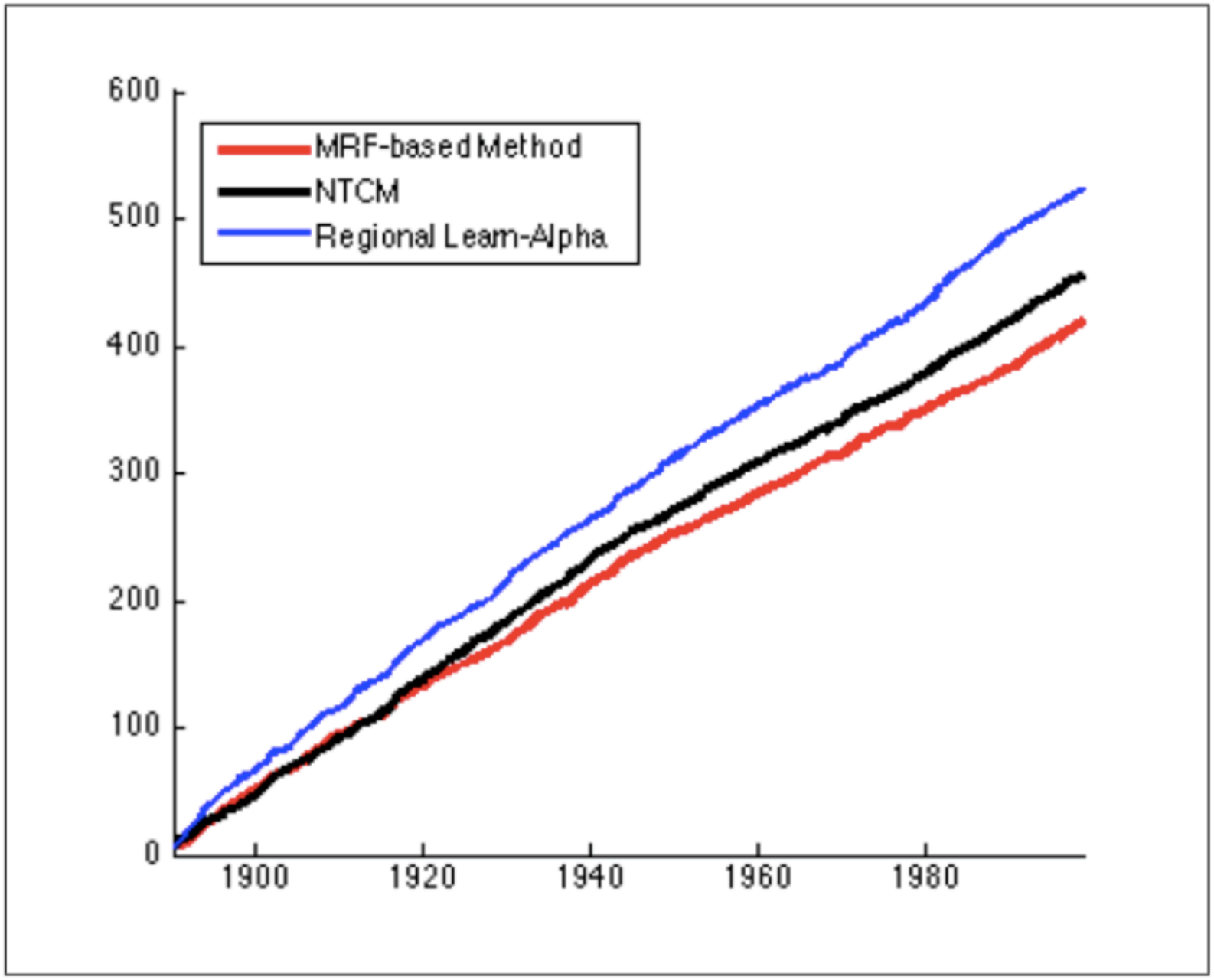
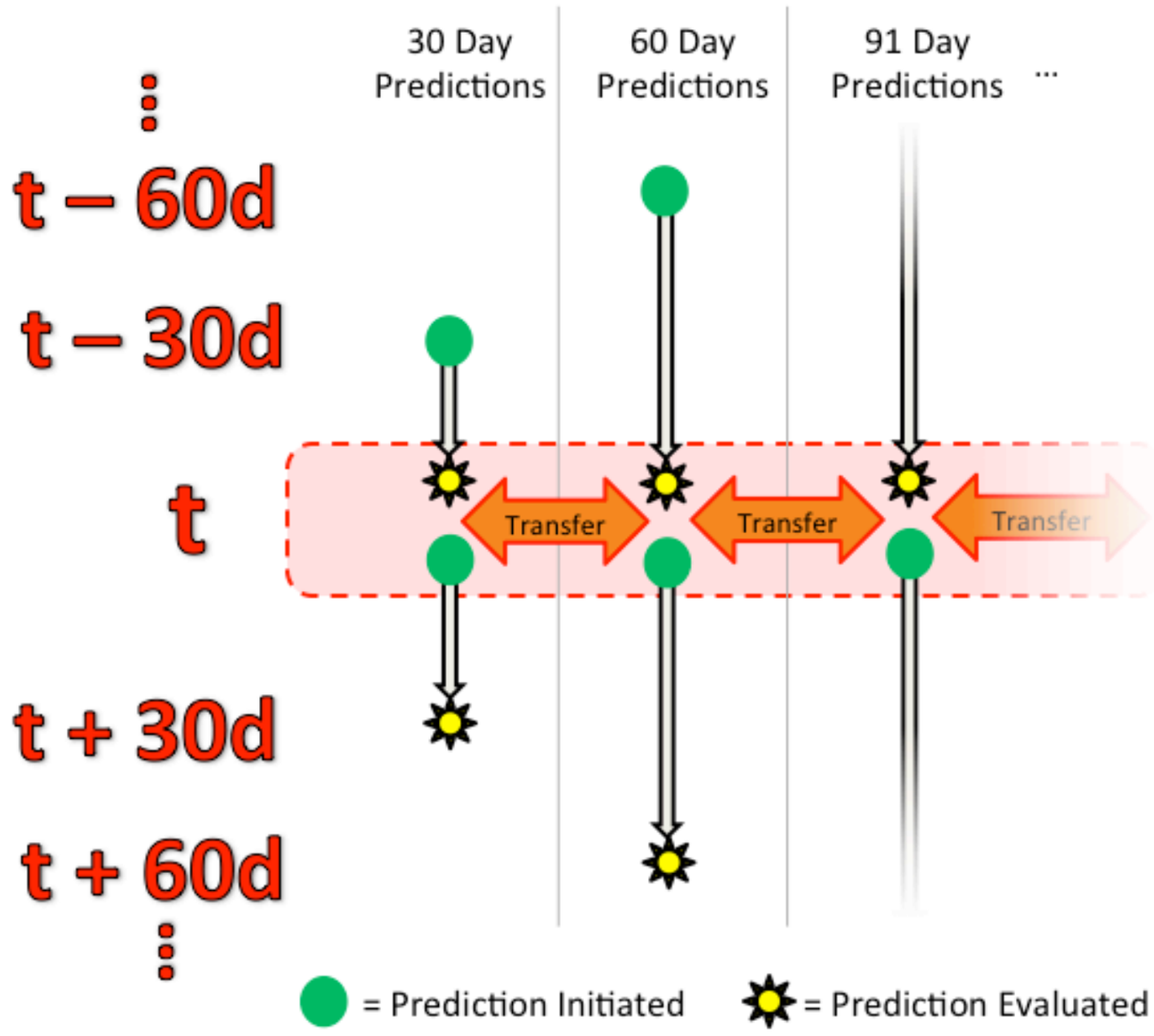


FIGURE 1.11: Cumulative mean regional loss of the hindcast.

Seasonal prediction: Online multi-task learning

[McQuade & M, Climate Informatics 2015; SIGMOD DSMM 2016]

- Given forecasts of multiple time periods
- Each forecast period treated as a different task
- Allow influence between tasks



Online multi-task learning

Task-similarity matrix [*cf.* Saha et al., AISTATS 2011]

- Allow influence between "neighboring" forecast lengths, parameterized by s

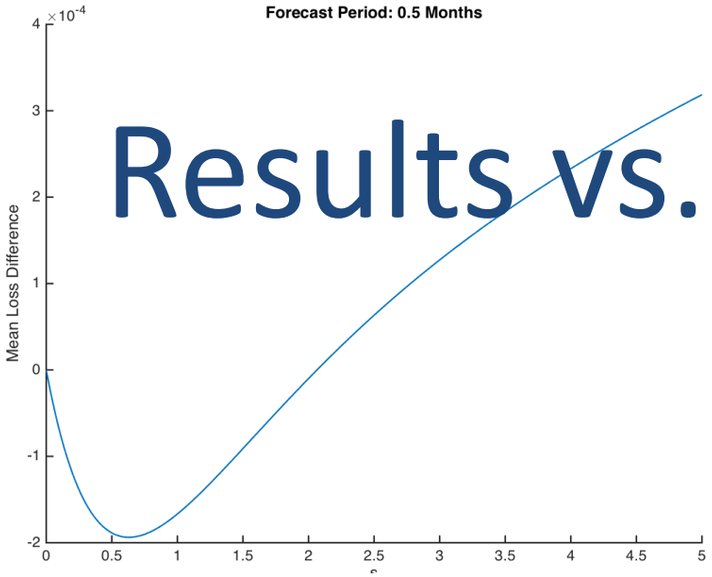
Months: 0.5 1.5 2.5 3.5 . . . 11.5

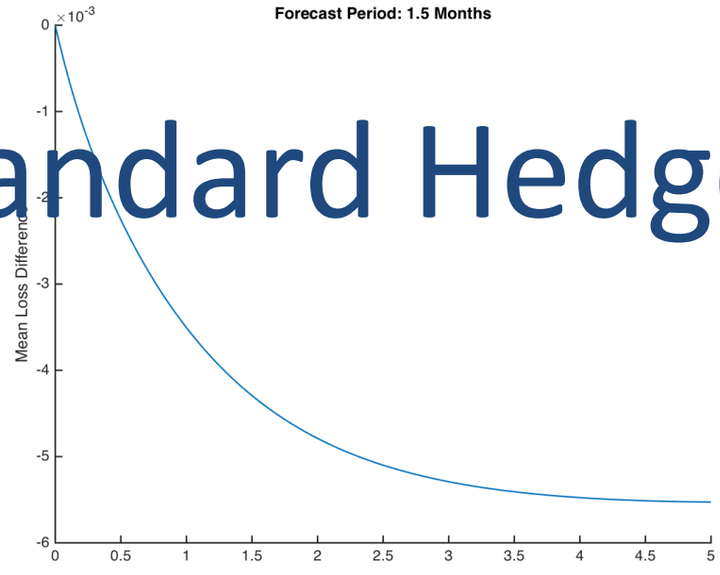
$$S = \begin{array}{c} 0.5 \\ 1.5 \\ \vdots \\ 11.5 \end{array} \begin{bmatrix} \frac{1}{1+s} & \frac{s}{1+s} & 0 & 0 & \cdots \\ 0 & \frac{s}{1+2s} & \frac{1}{1+2s} & \frac{s}{1+2s} & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \\ & & & & \frac{1}{1+s} \end{bmatrix}$$

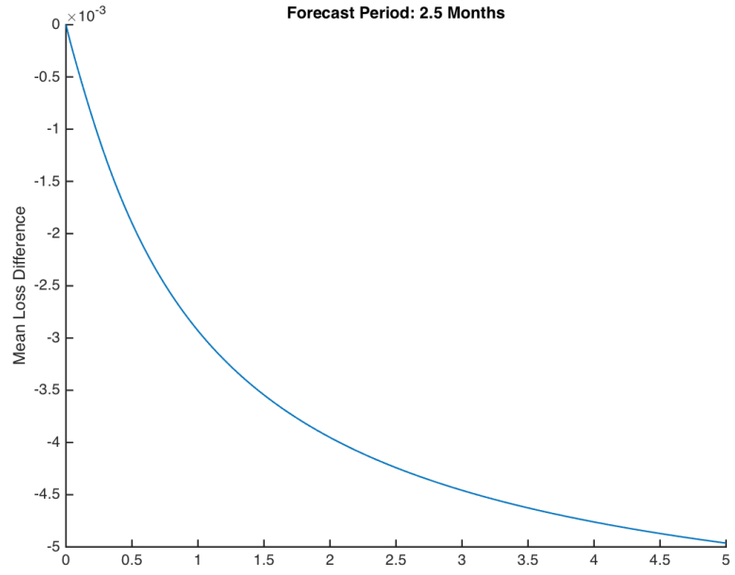
Multi-task update rule (extended from Hedge / Static-Expert algorithm)

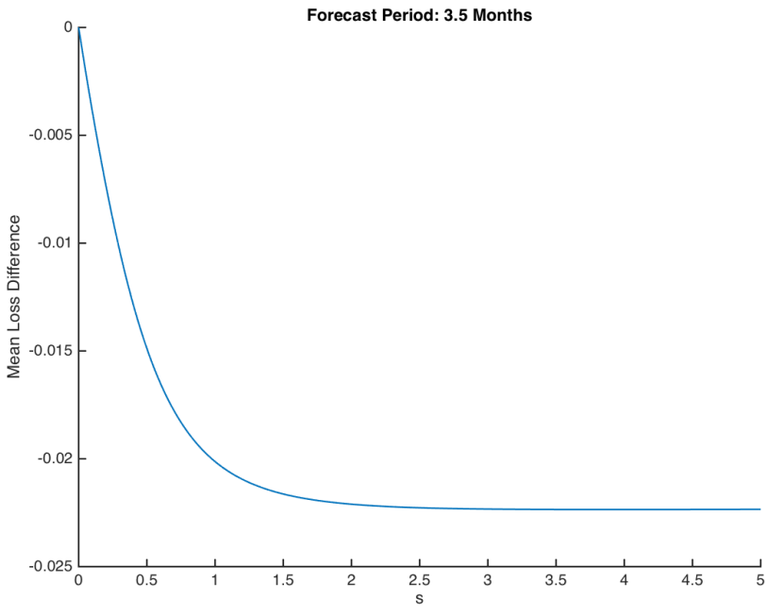
$$p_{t+1,j}(i) \propto p_{t,j}(i) e^{-\sum_k S_{j,k} L_{k,t}(i)}$$

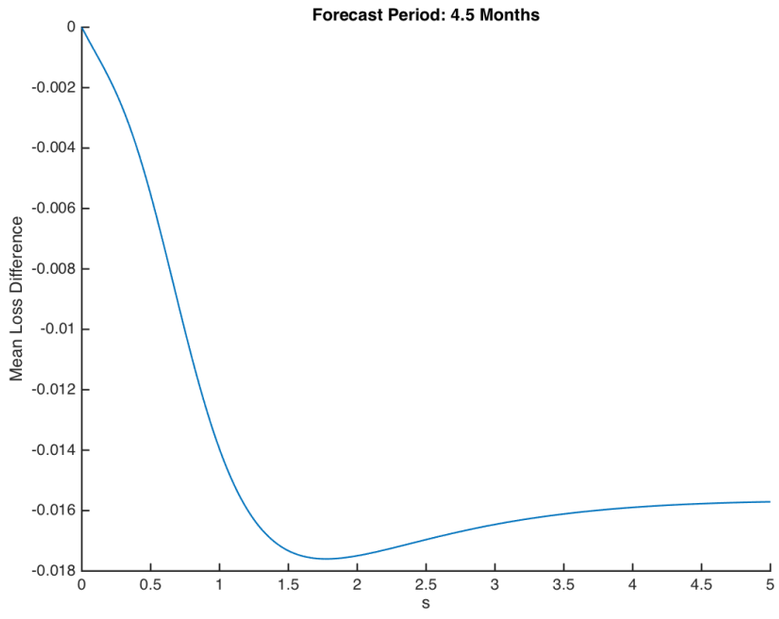
Results vs. standard Hedge

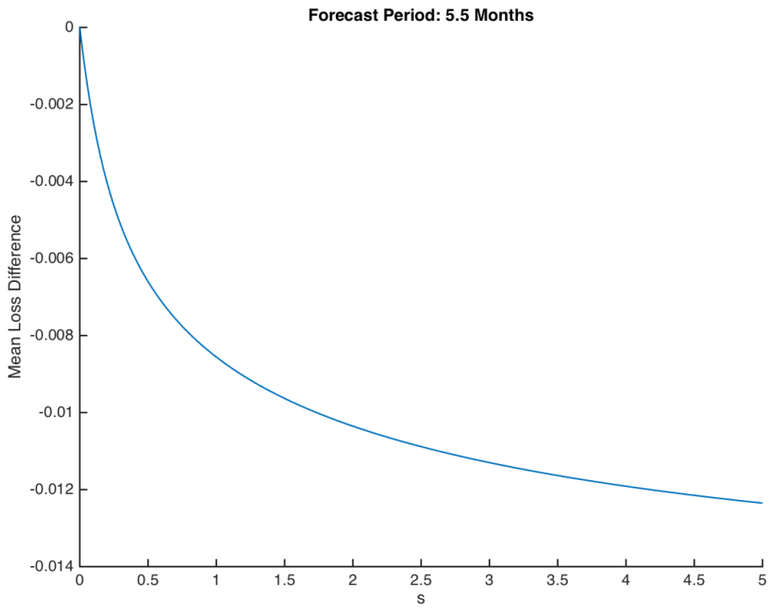


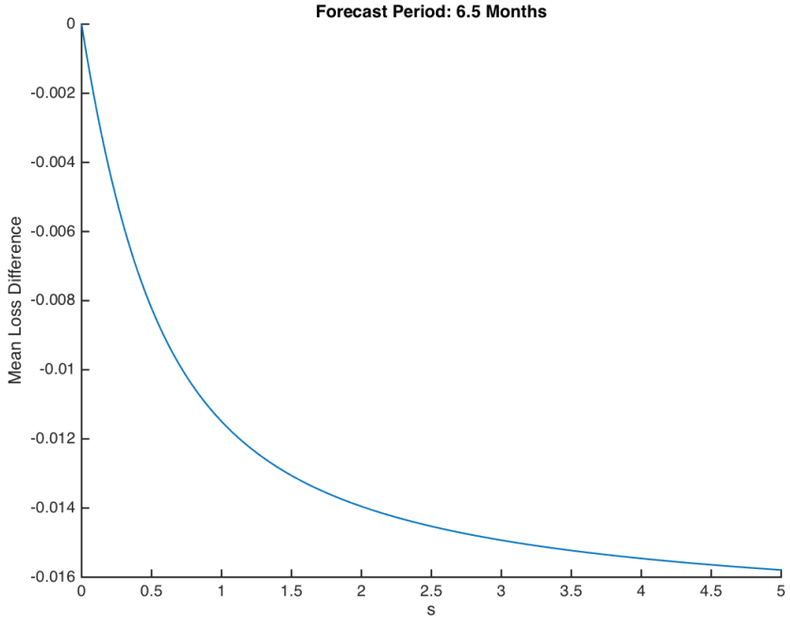


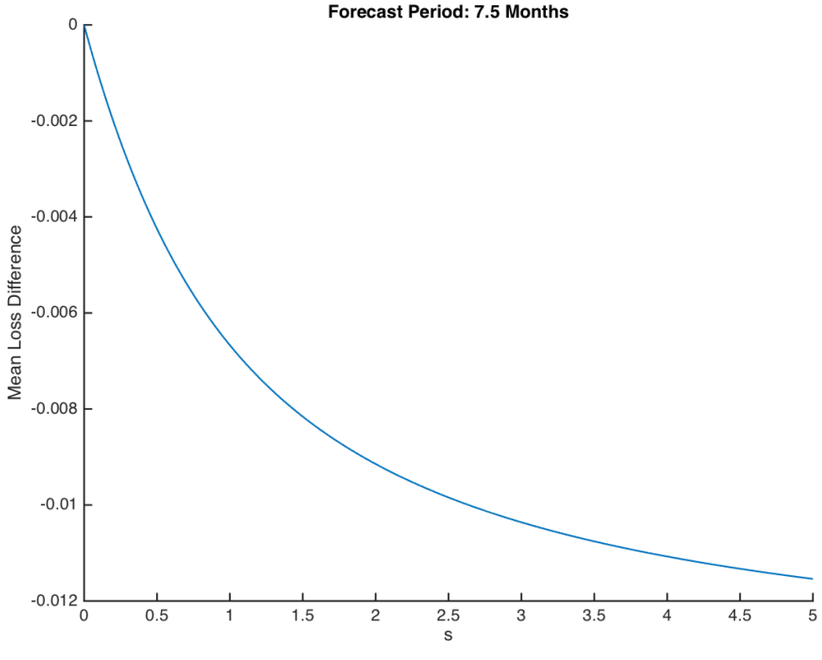


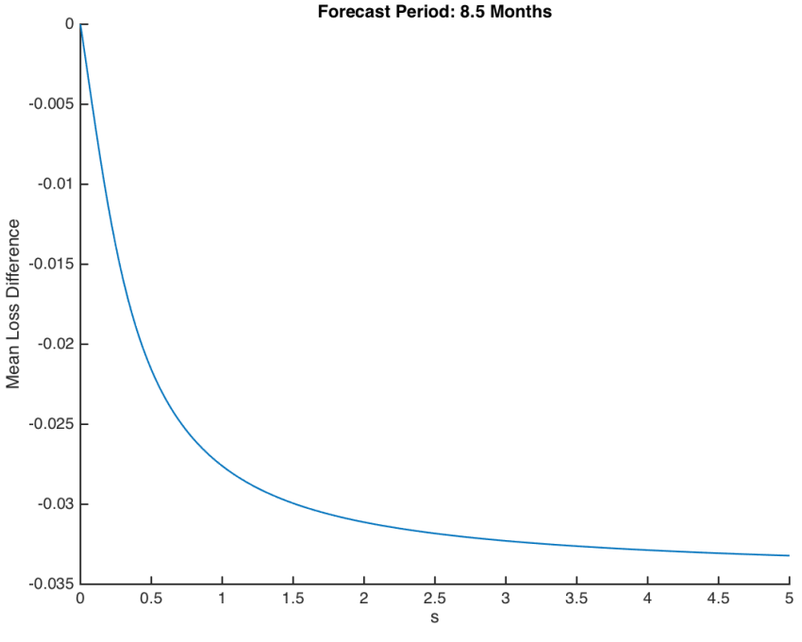




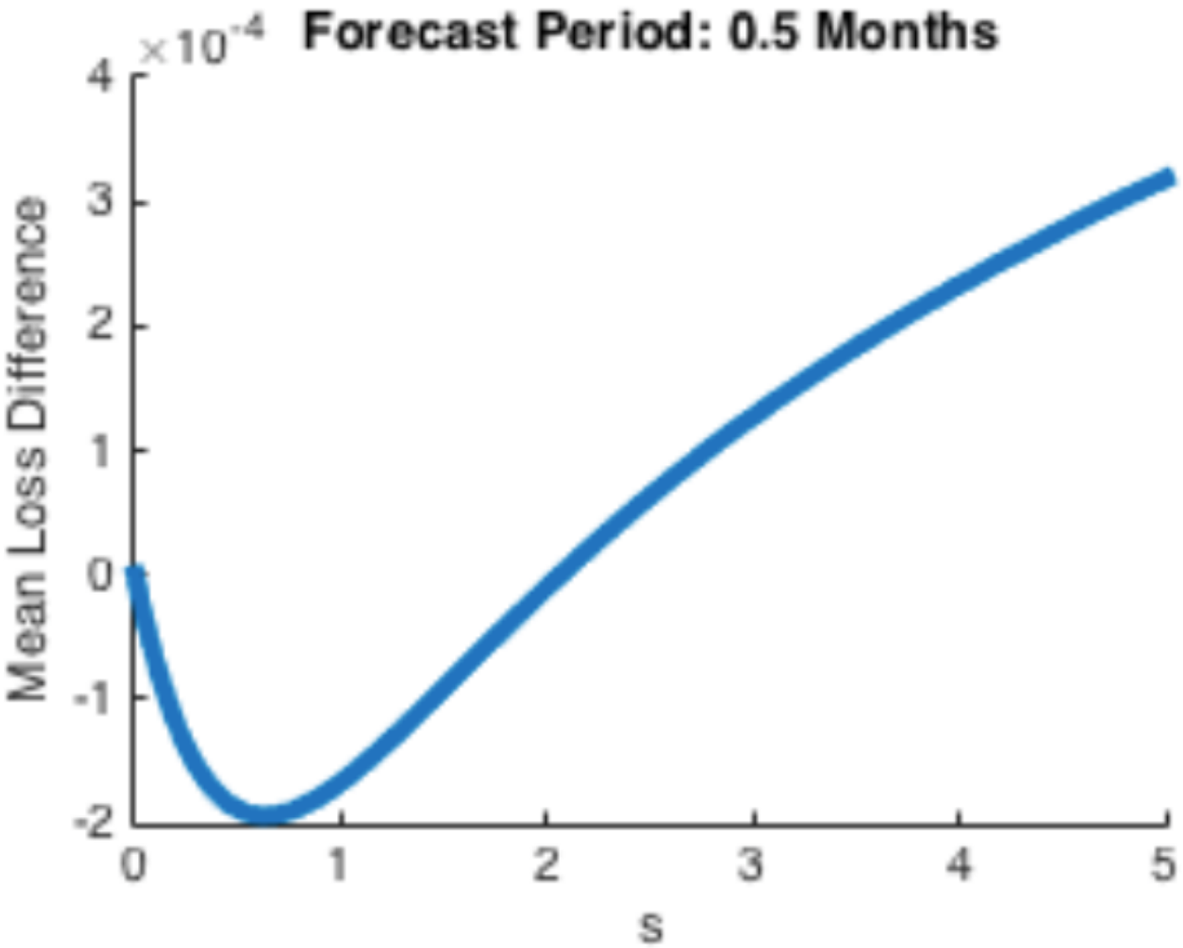


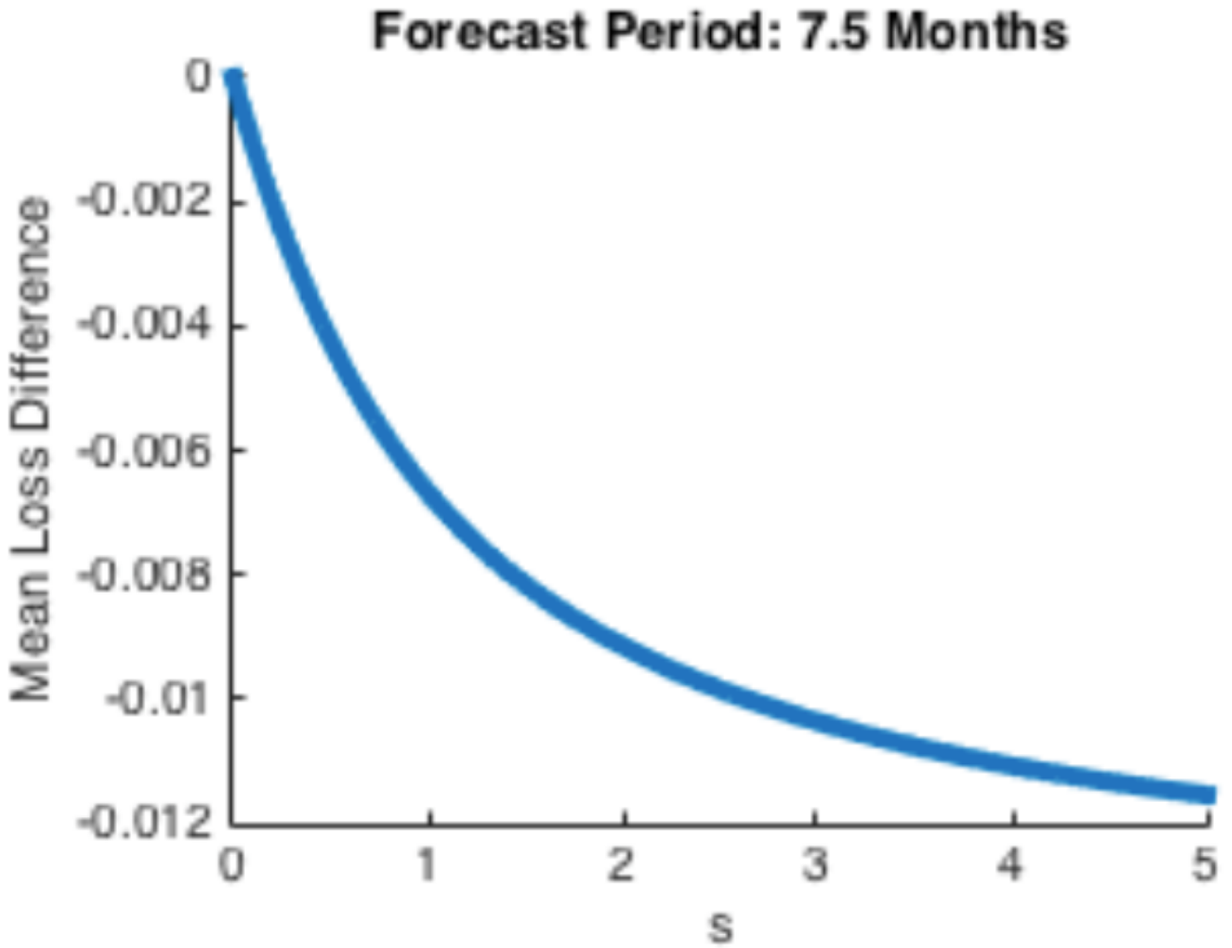






Results vs. standard Hedge





For each of the 12 forecast periods, sharing influence from other forecast periods improved predictions.

Only for the 2 forecast periods was loss increased for some s values.

Application to financial volatility prediction [McQuade & M, DSMM 2016]