An Informal Mathematical Tour of Feature Learning

Nando de Freitas
Ben Marlin, Kevin Swersky, Bo Chen, Marc'Aurelio Ranzato
Misha Denil, Hugo Larochelle, Loris Bazzani






+ + +

How we get supervisory signals in "unsupervised" feature learning?

1. **Reconstruction**
2. **Contrastive learning**
3. **Association/correlation (time, translation, etc.)**

- clustering ⟷ density estimation
- reconstruction
- margin / rank loss
- < NCA, Siamese, CD

Questions and choices we face

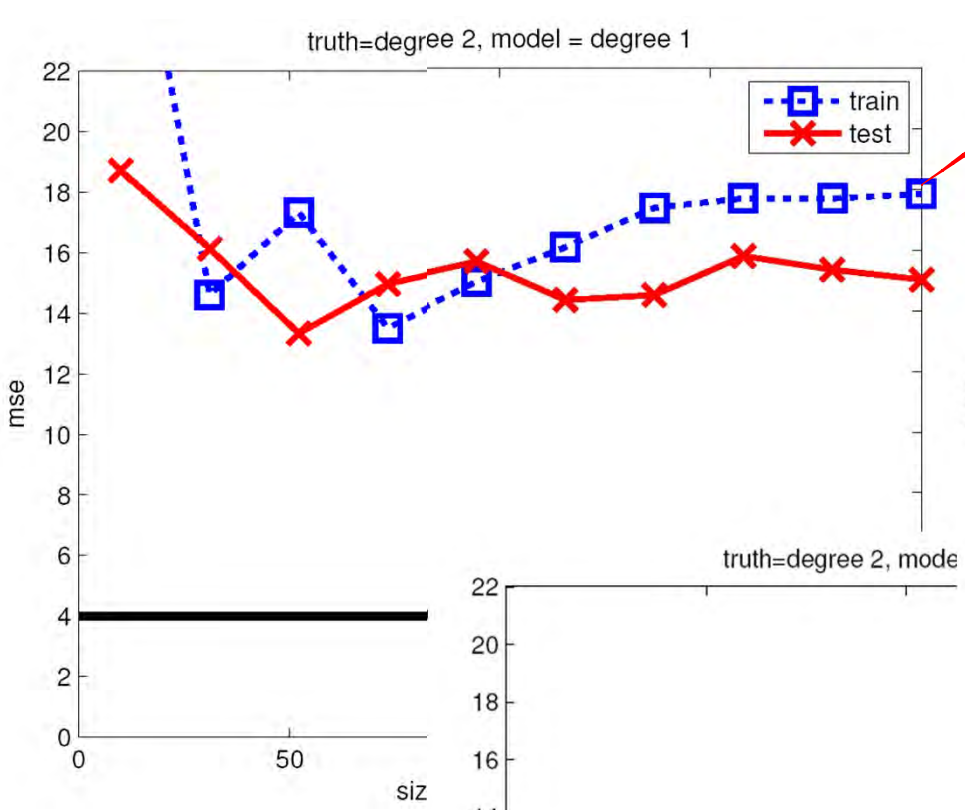
1. Deterministic or probabilistic models? e.g. energy based models and autoencoders Vs RBMs, sparse coding, hierarchical Dirichlet processes, …

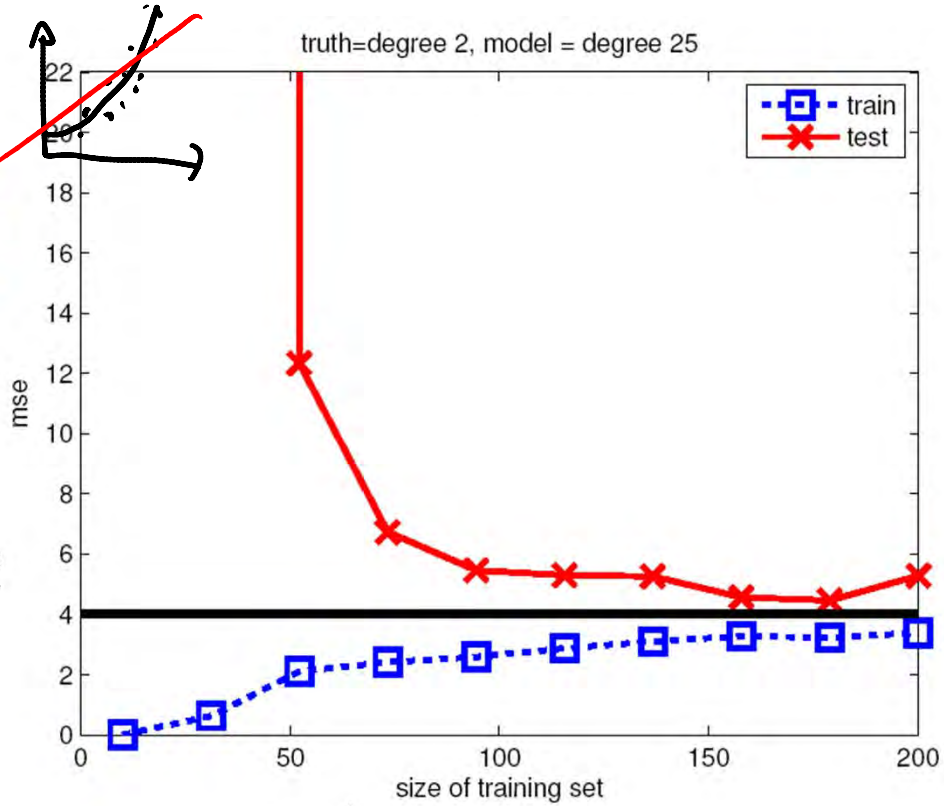
2. What inductive principles are appropriate and when? Maximum likelihood, Bayes, max-margin, score matching, ratio matching, pseudo-likelihood, contrastive divergence, method of moments, indirect inference, …

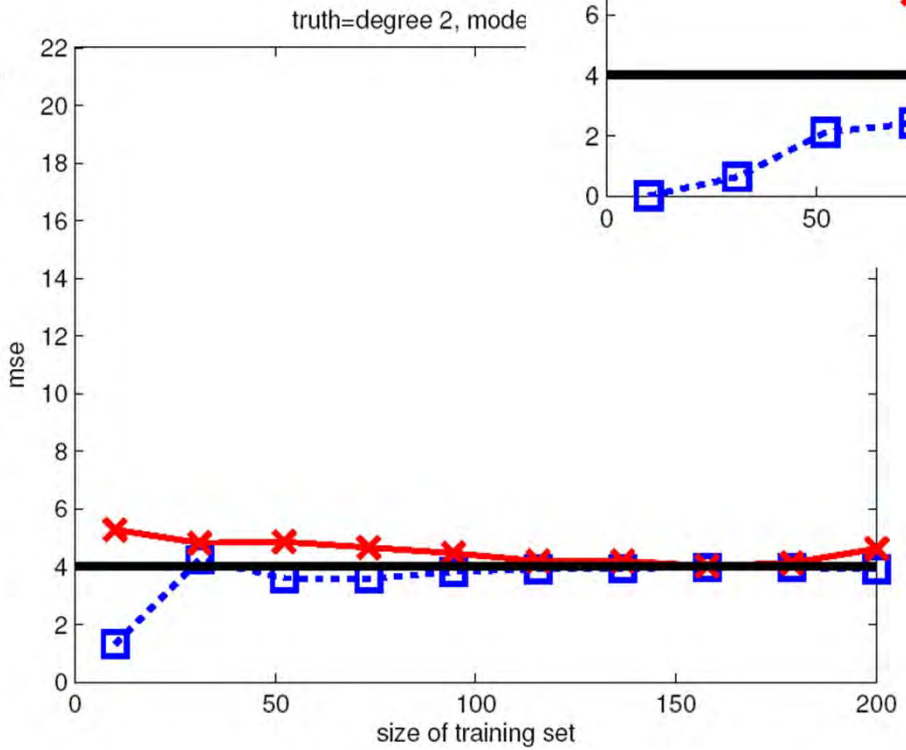
3. Static, discrete models or continuous-time dynamical systems?

4. Traditional AI (logic, search, constraints) or the ML approach?

How much data and model complexity?



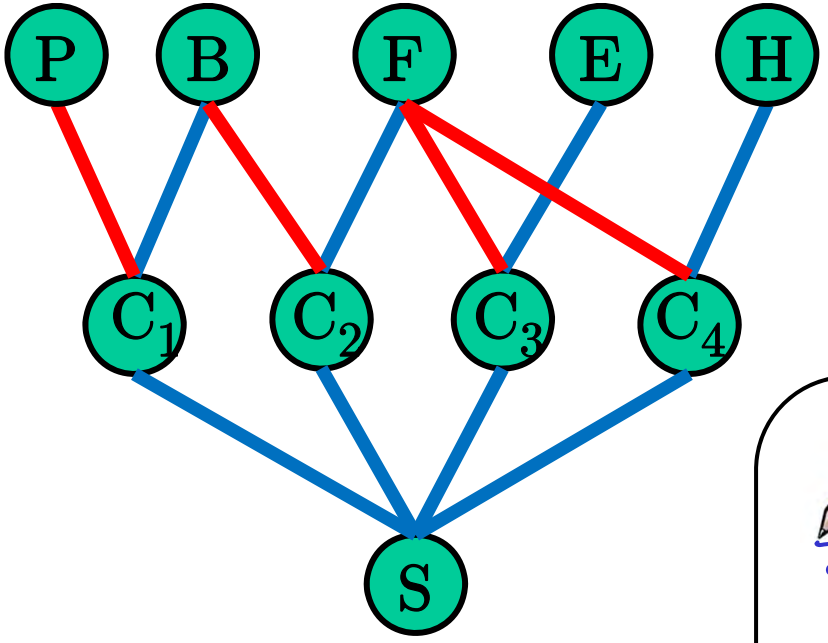




[Murphy, 2012]

Re-visiting logic, computational complexity and constraint satisfaction (2-SAT)

Consider the CNF expression $S = C_1 \wedge \ldots \wedge C_m$, where each clause C_i is a disjunction of literals $x_{i,1} \vee \ldots \vee x_{i,k_i}$ defined on propositional variables. When each clause has two parents at most, the problem is known as 2-SAT.



$S = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

implication

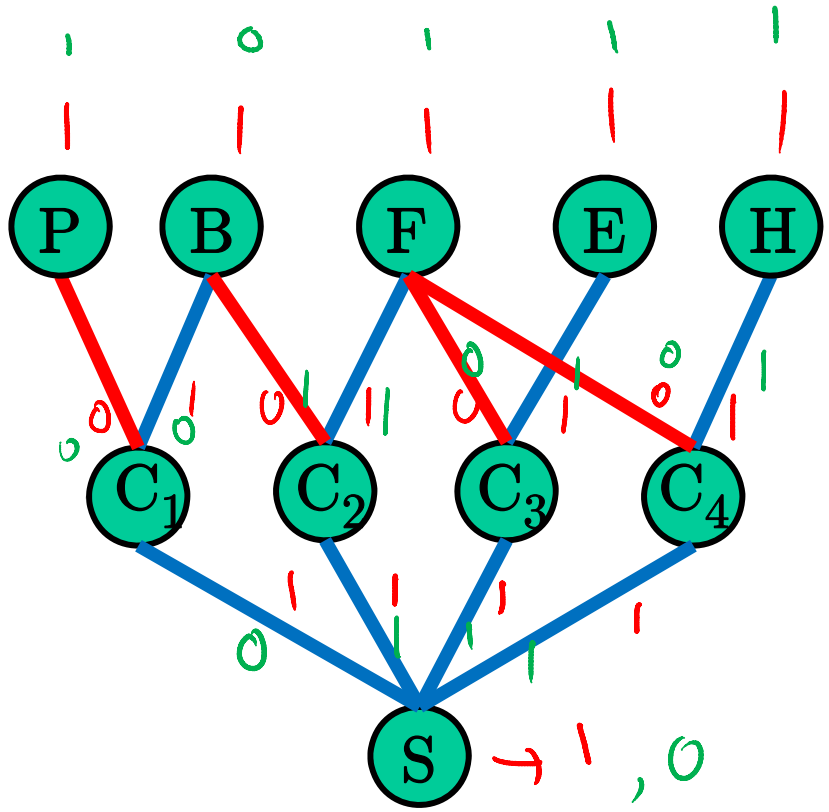
$Parrot \implies Bird \quad C_1 = \neg P \vee B$

$Bird \implies Flies \quad C_2 = \neg B \vee F$

$Flies \implies Escapes \quad C_3 = \neg F \vee E$

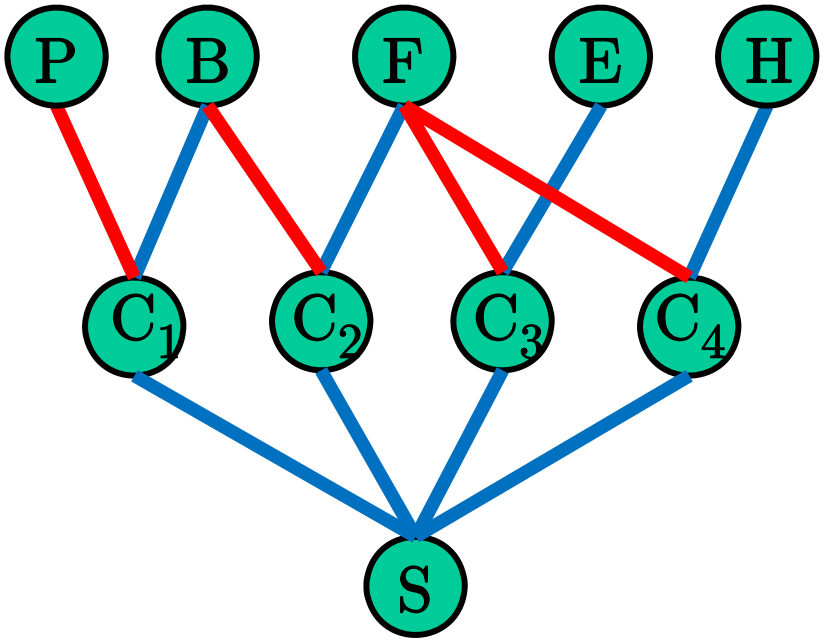
$Flies \implies HasWings \quad C_4 = \neg F \vee H$

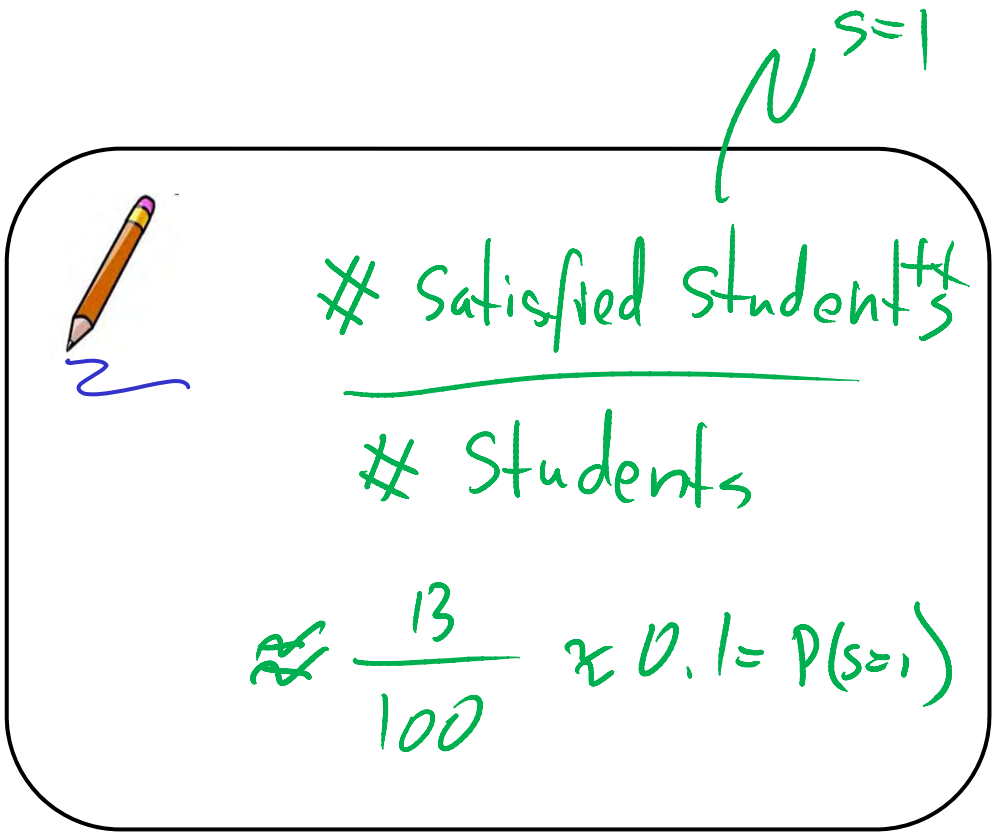
P	B	$\neg P \vee B$ (imply)
0	0	1
0	1	1
1	0	0
1	1	1



$$Parrot \implies Bird \qquad \neg P \vee B$$
$$Bird \implies Flies \qquad \neg B \vee F$$
$$Flies \implies Escapes \qquad \neg F \vee E$$
$$Flies \implies HasWings \qquad \neg F \vee H$$

1. **Verification**: Does (P=1,B=1,F=1,E=1,H=1), i.e. (11111), satisfy this 2-SAT problem?
2. **Verification**: Does (10111) satisfy it?
3. **Maximization**: What is the maximum number of clauses that can be satisfied?
4. What is the number of possible assignments to (PBFEH)?
5. **Counting**: How many assignments satisfy this 2-SAT example?

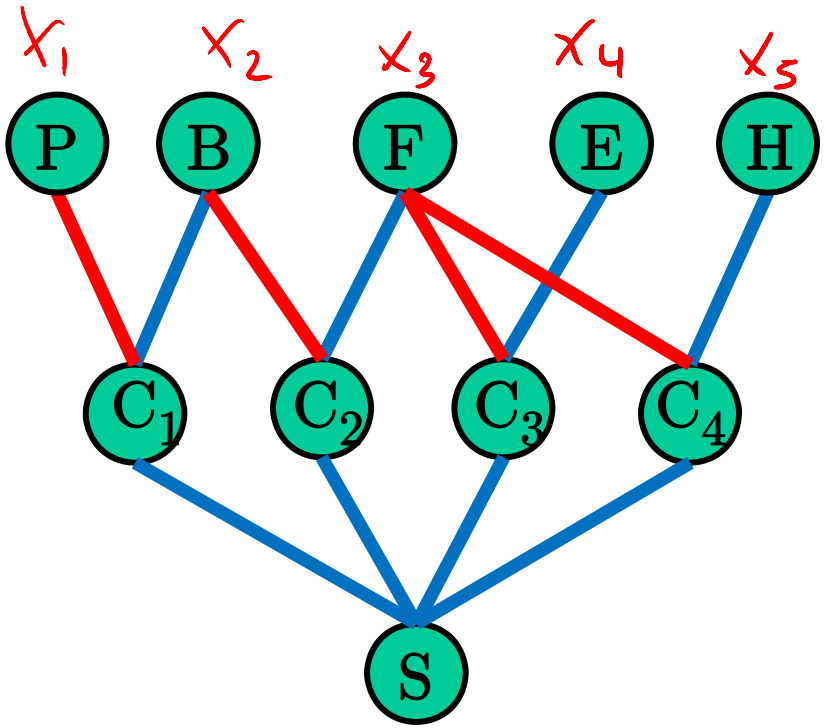




Counting: How many assignments satisfy this 2-SAT example
Approximate answer: Use the Monte Carlo Method.

i. **Sample P, B, F, E and H by flipping a coin for each variable N times.**
ii. **For each sample of (PBFEH), check for satisfiability.**
iii. **The probability of satisfiability, P(S=1), is approximated as the number of satisfying samples divided by N.**
iv. **The expected number number of satisfiable samples n = P(S=1) 2^5.**

From max-2-SAT to energy models



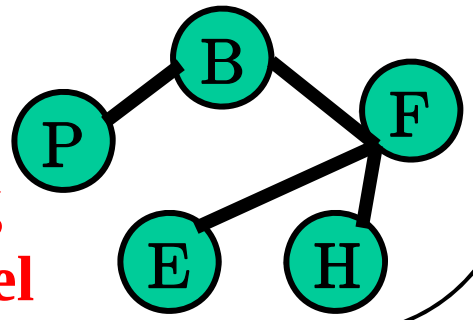
$$\neg P \vee B \longrightarrow \theta_1 P(1-B)$$
$$\neg B \vee F \longrightarrow \theta_2 B(1-F)$$
$$\neg F \vee E \longrightarrow \theta_3 F(1-E)$$
$$\neg F \vee H \longrightarrow \theta_4 F(1-H)$$

Introduce the notation
$P = X_1, B = X_2, F = X_3, E = X_4, H = X_5.$
The total energy of the system is:

$$E = \theta_1 P + \theta_2 B + (\theta_3 + \theta_4) F - \theta_1 PB - \theta_2 BF - \theta_3 FE - \theta_4 FH$$

Assume some clauses are harder to satisfy than others. Introduce a weight (θ) to measure this.

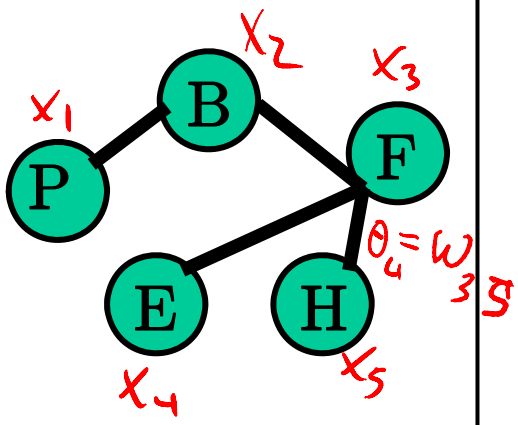
To obtain the Energy of the system of binary variables, use X for a negated propositional variable and 1-X otherwise. Then sum over all clauses.



Ising model

$$E = \theta_1 P + \theta_2 B + (\theta_3 + \theta_4) F - \theta_1 PB - \theta_2 BF - \theta_3 FE - \theta_4 FH$$

Let $P = x_1, B = x_2, F = x_3, E = x_4$ and $H = x_5$



The energy can be written as:

$$E = -\sum_{i=1}^{5} b_i x_i - \sum_{i=1}^{5} \sum_{j>i} x_i w_{ij} x_j$$

In our case:

$b_1 = -\theta_1$ | $w_{12} = \theta_1$

$b_2 = -\theta_2$ | $w_{23} = \theta_2$

$b_3 = -\theta_3 - \theta_4$ | $w_{34} = \theta_3$

$b_4 = 0$ | $w_{35} =$

$b_5 = 0$ | $w_{45} =$

From max-2-SAT to Energy to Probability

Let us look at the energy of a few configurations, assuming all the $\theta_i = 1$. In this case the energy is simply:

$$E(x_1, x_2, \ldots, x_5) = x_1 + x_2 + 2x_3 - x_1x_2 - x_2x_3 - x_3x_4 - x_3x_5$$

What is the lowest energy? When is it attained?
What is the maximum energy?
What should the most probable configuration be?

x_1	x_2	x_3	x_4	x_5	E
1	1	1	1	1	0
0	1	1	1	1	0
1	0	1	1	1	1
1	1	1	0	0	2
0	0	1	1	1	0

$$P(x_1 x_2 x_3 x_4 x_5) = \frac{e^{-E(x_1 \cdots x_5)}}{Z}$$

$$Z = \sum_{x_1} \sum_{x_2} \cdots \sum_{x_5} e^{-E(x_1, \cdots x_5)}$$

Boltzmann distribution

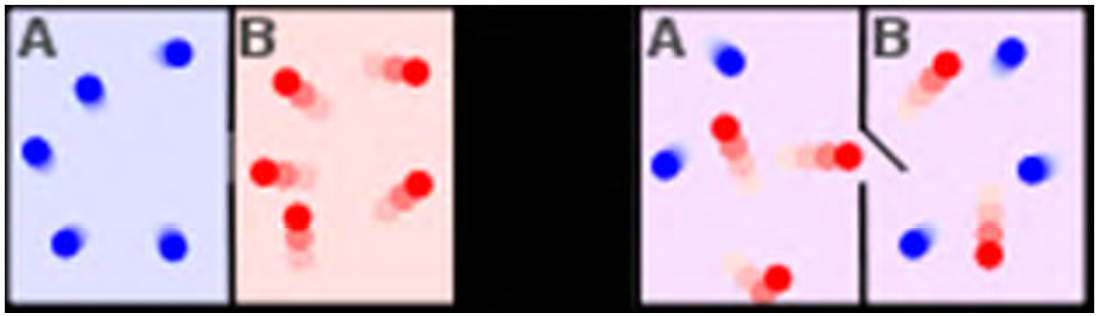
Ising models and the 2nd law of thermodynamics

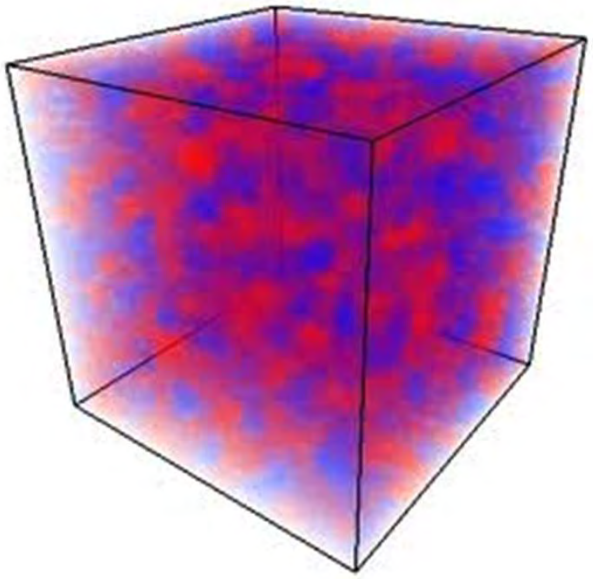
- **The Ising model describes many physical phenomena**

  - "*The Ising model can be reinterpreted as a statistical model for the motion of atoms. A coarse model is to make space-time a lattice and imagine that each position either contains an atom or it doesn't.*" **Wikipedia Ising Model page.**

  - "*The original motivation for the model was the phenomenon of magnetism.*"

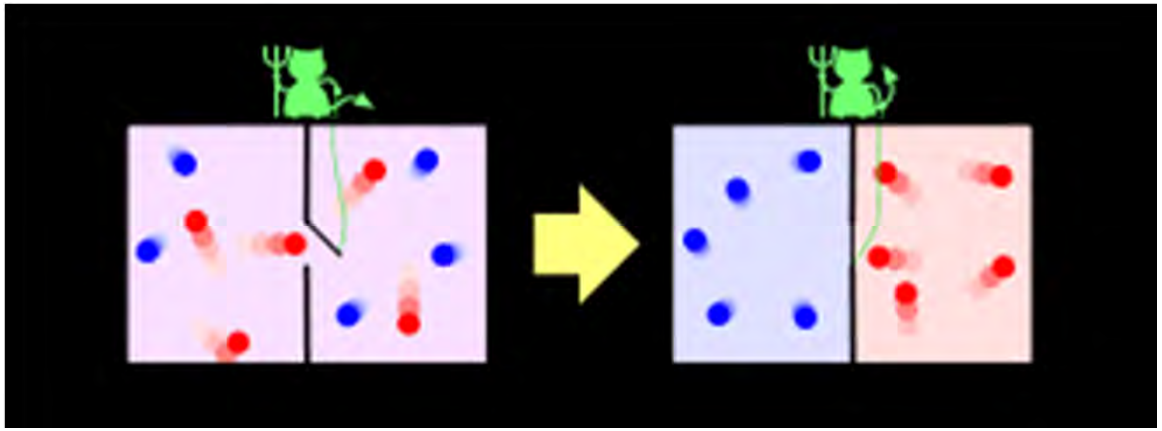
- **Second law of thermodynamics and stability**





On information and energy – Maxwell's Demon

In this thought experiment, "*an imaginary container is divided into two parts by an insulated wall, with a door that can be opened and closed by what came to be called "Maxwell's Demon". The hypothetical demon is only able to let the "hot" molecules of gas flow through to a favored side of the chamber, causing that side to appear to spontaneously heat up while the other side cools down.*"



- **Does this violate the 2nd law?**
- **What is the relation of information and energy?**

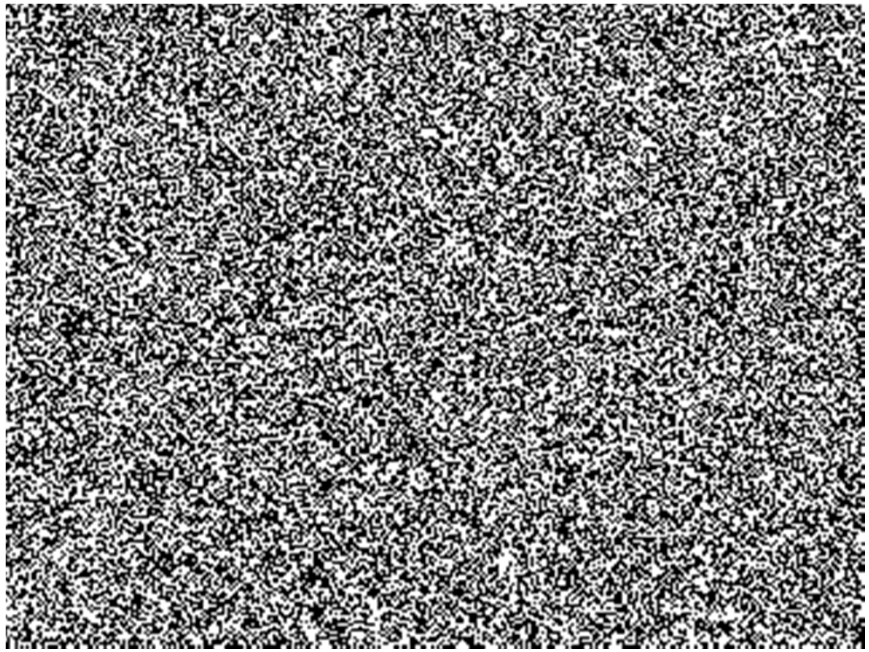
Nature seems to be very powerful at solving hard computational problems.

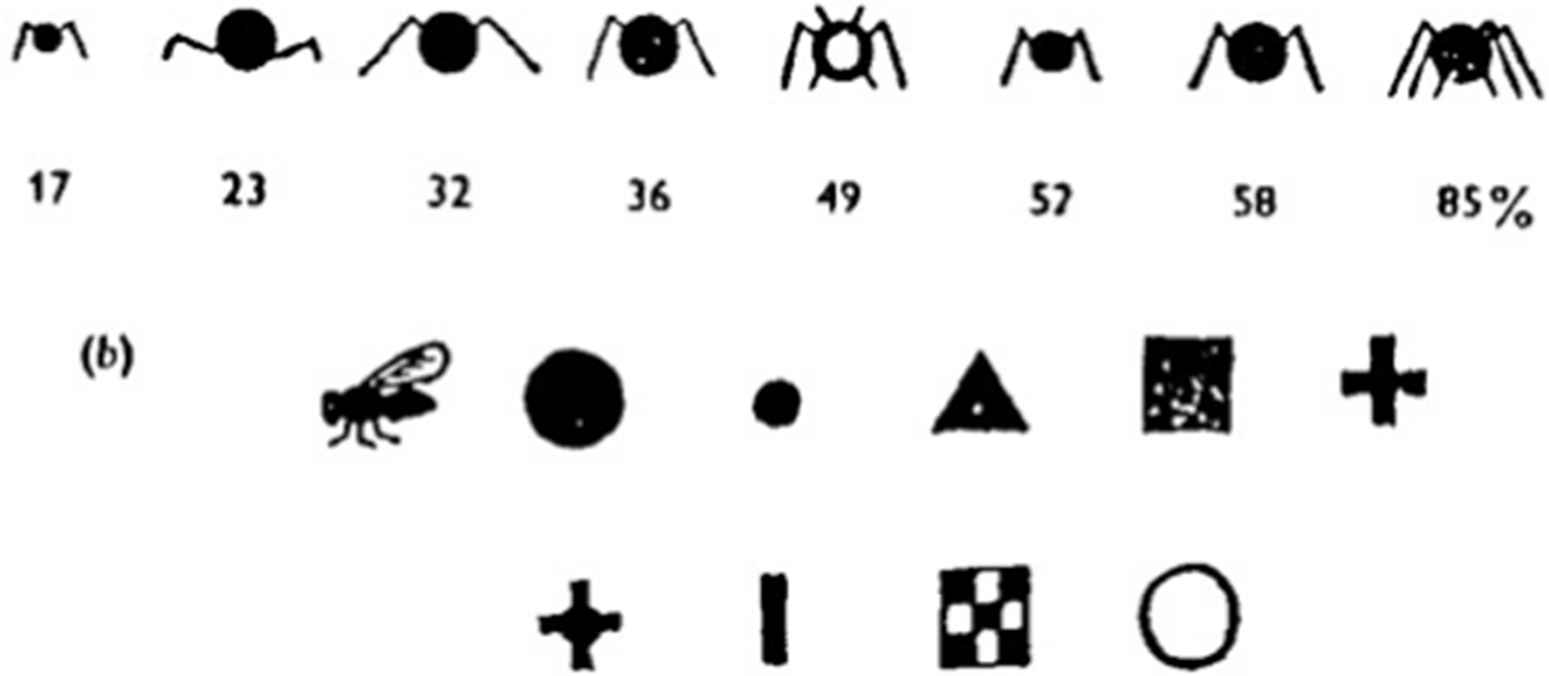
Can we recruit nature to do computing?

Nature has limits!

But it is important that we follow nature's guidelines for design!

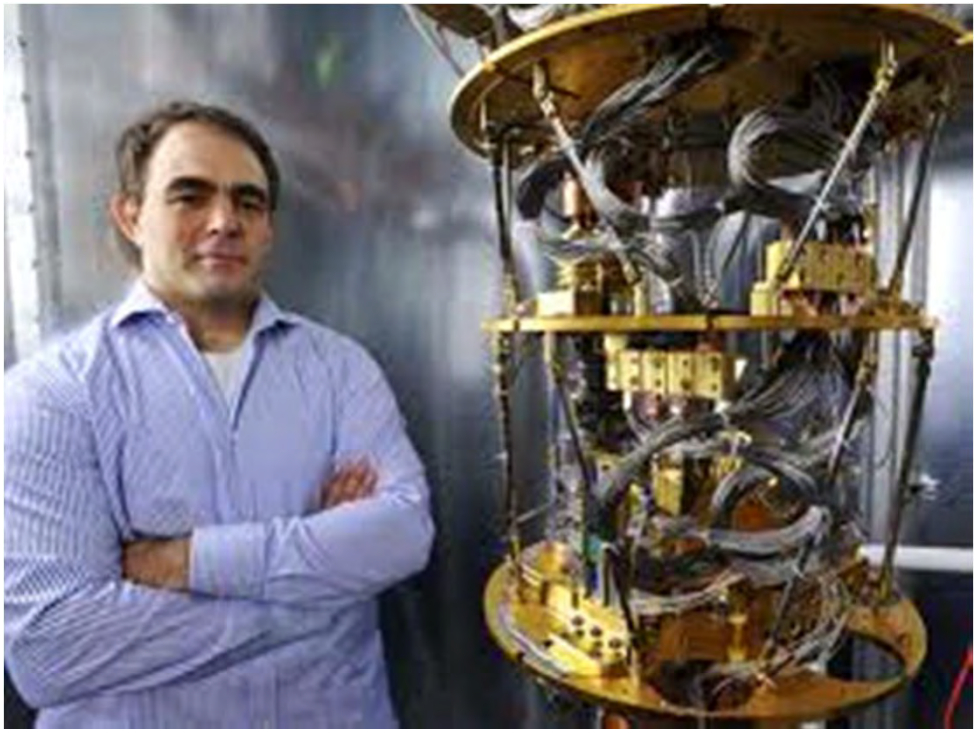
The space of 1000 by 1000 black and white images is $256^{1000000}$



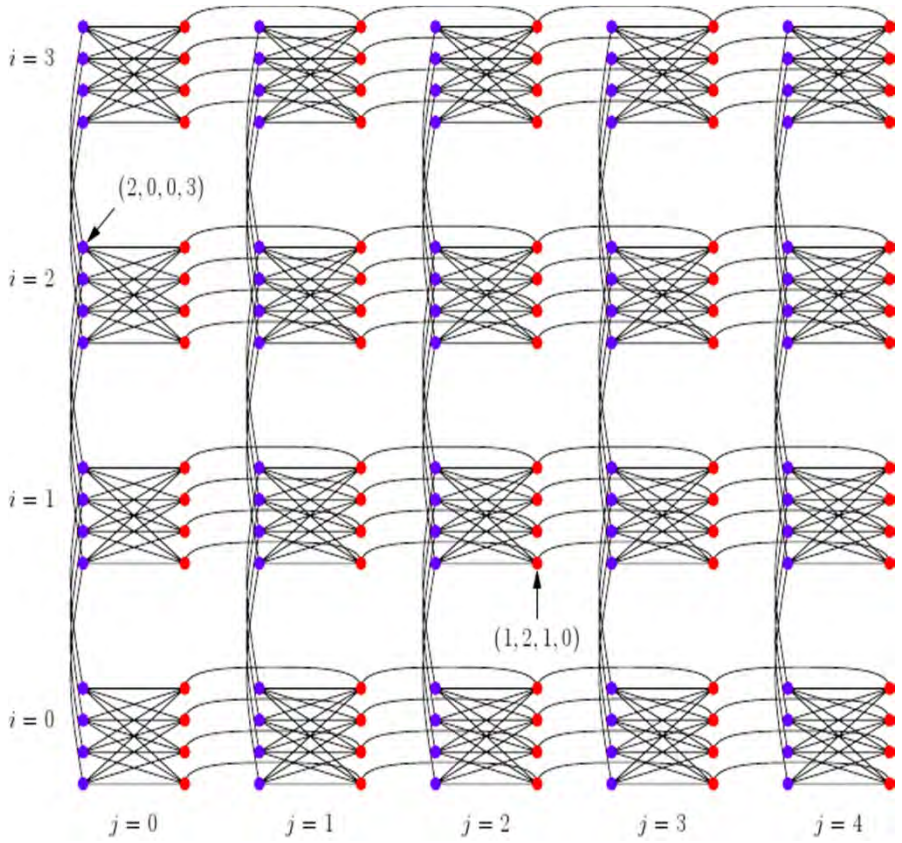


Text-fig. 12. Stimuli found by Drees to evoke courtship (*a*) and prey capture (*b*) in male jumping spiders (*Epiblemum scenicum*). The numbers beneath each figure in (*a*) are the percentage of trials on which courtship was evoked. After Drees (1952).

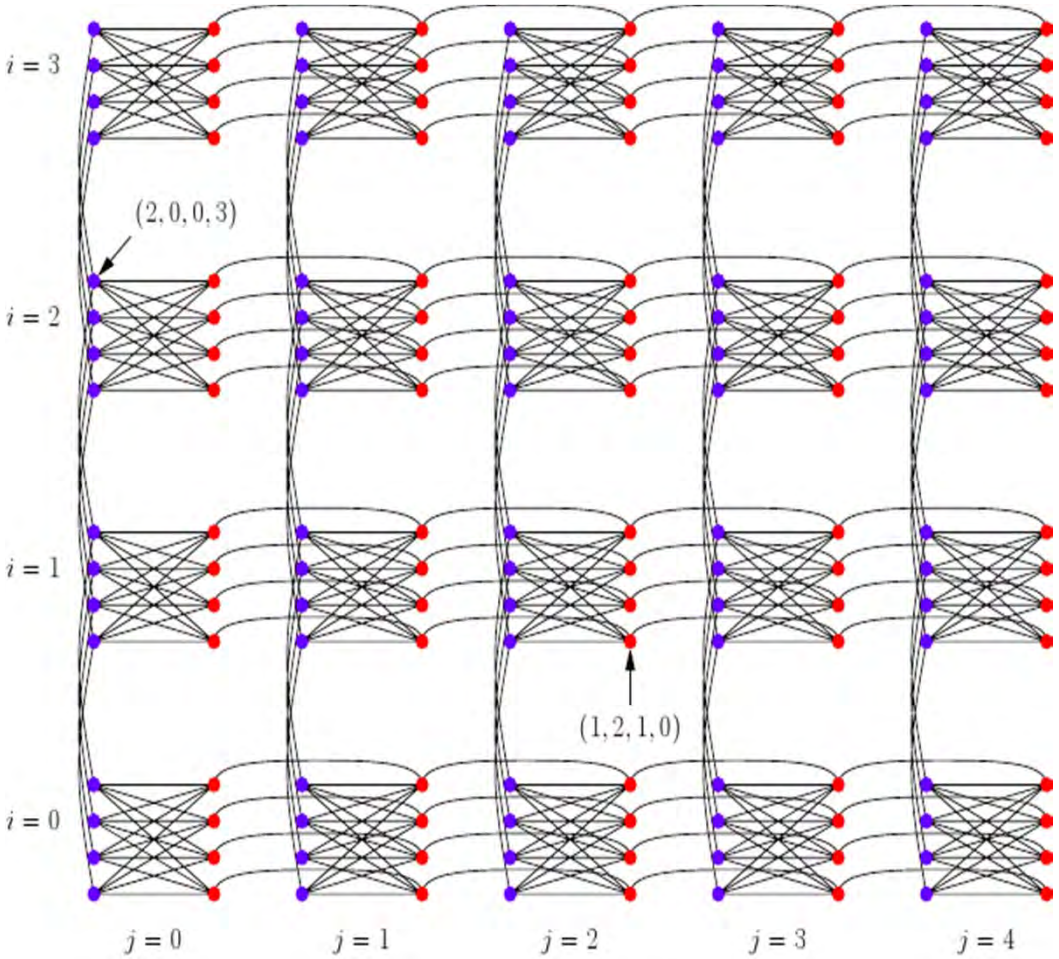
We might not be able to recruit nature to solve NP hard problems, but we can use it to speed-up problems that we currently struggle with: e.g. cryptography and neural computation



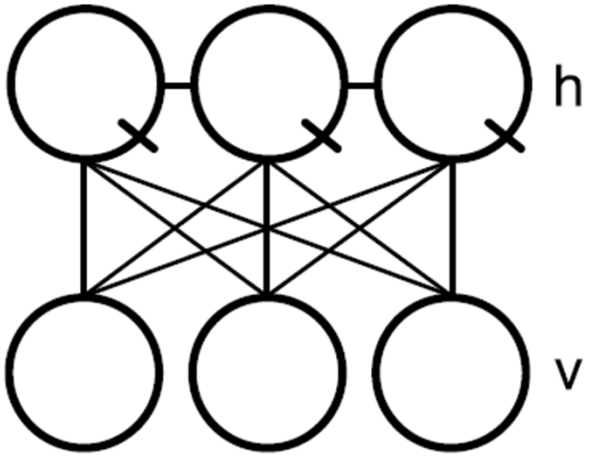
Geordie Rose, D-Wave Systems



Quantum annealing for deep learning



Lateral connections



[Denil & dF, 2011]

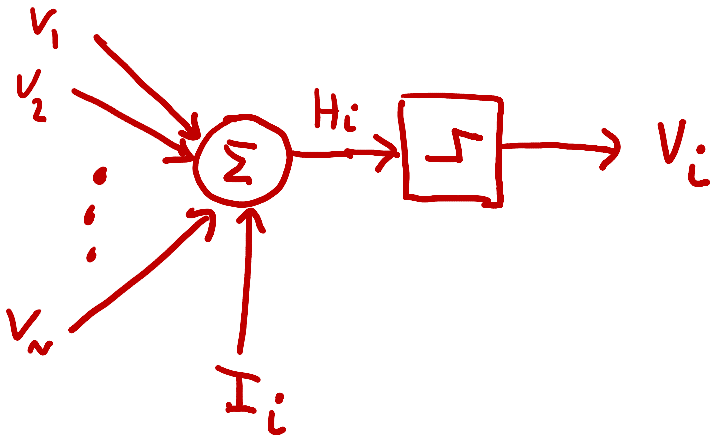
Hopfield networks and dynamics

$$H_i = \sum_{j \neq i} T_{ij} V_i + I_i$$

$\Theta = \{T, U\}$

known.

$$V_i = \begin{cases} 0 & \text{if } H_i < U_i \\ 1 & \text{if } H_i \geq U_i \end{cases}$$

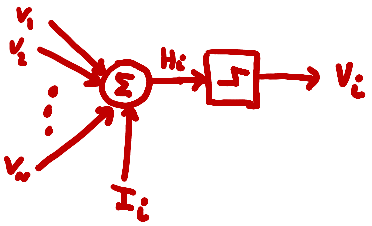


Let $E = -\sum_i \sum_{j \neq i} T_{ij} V_i V_j - \sum_i I_i V_i + \sum_j U_i V_i$

$$H_i = \sum_{j \neq i} T_{ij} V_i + I_i$$

$\Theta = \{T, U\}$ known.

$$V_i = \begin{cases} 0 & \text{if } H_i < U_i \\ 1 & \text{if } H_i \geq U_i \end{cases}$$



Let $E = -\sum_i \sum_{j \neq i} T_{ij} V_i V_j - \sum_i I_i V_i + \sum_j U_i V_i$

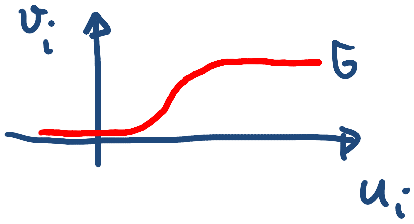
$$dE = -\left[\underbrace{\sum_{j \neq i} T_{ij} V_j + I_i - U_i}_{B_i}\right] dV_i$$

What can we say about dE? Can it be positive?

dE

E decreases

Hopfield networks and dynamics



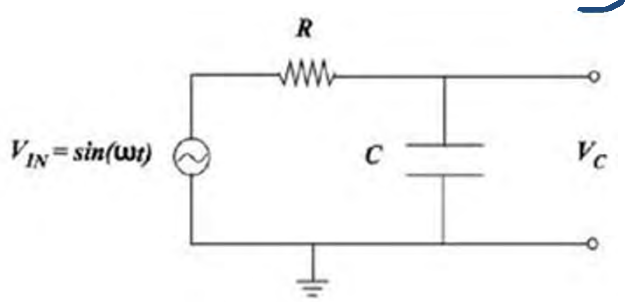
$$v_i = G(u_i)$$

$$\frac{du_i}{dt} + \frac{1}{R_i C_i} u_i = \frac{1}{C_i}\left[\sum_j T_{ij} V_j + I_i\right]$$

$C_i \equiv$ Capacitance

$R_i \equiv$ Resistance

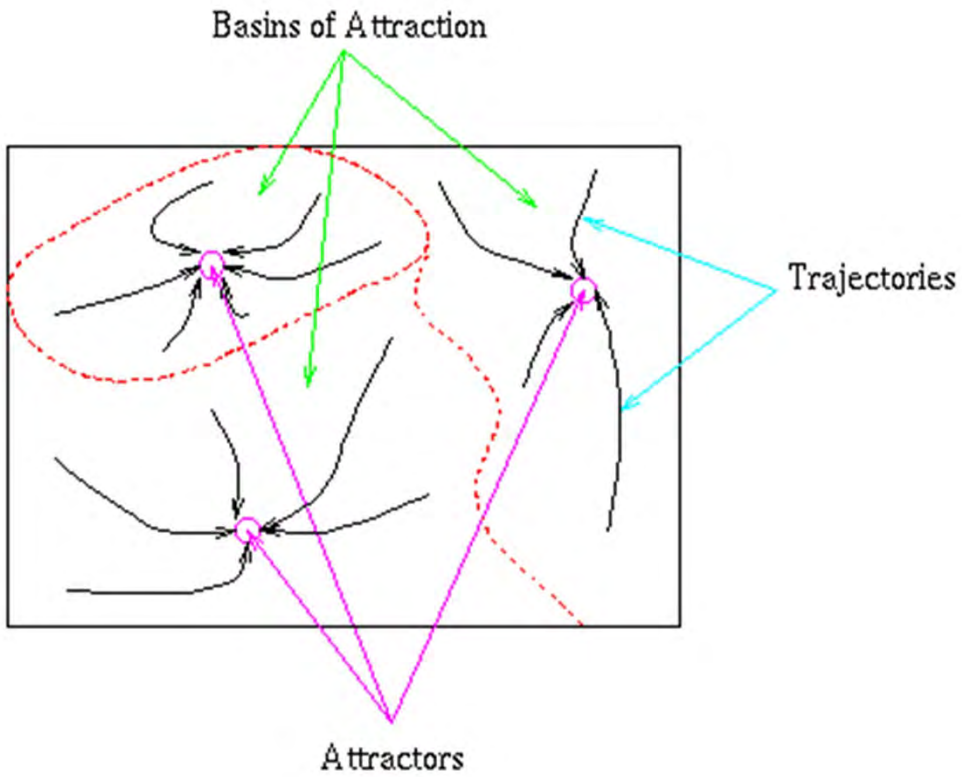
$T_{ij} \equiv$ Impedance

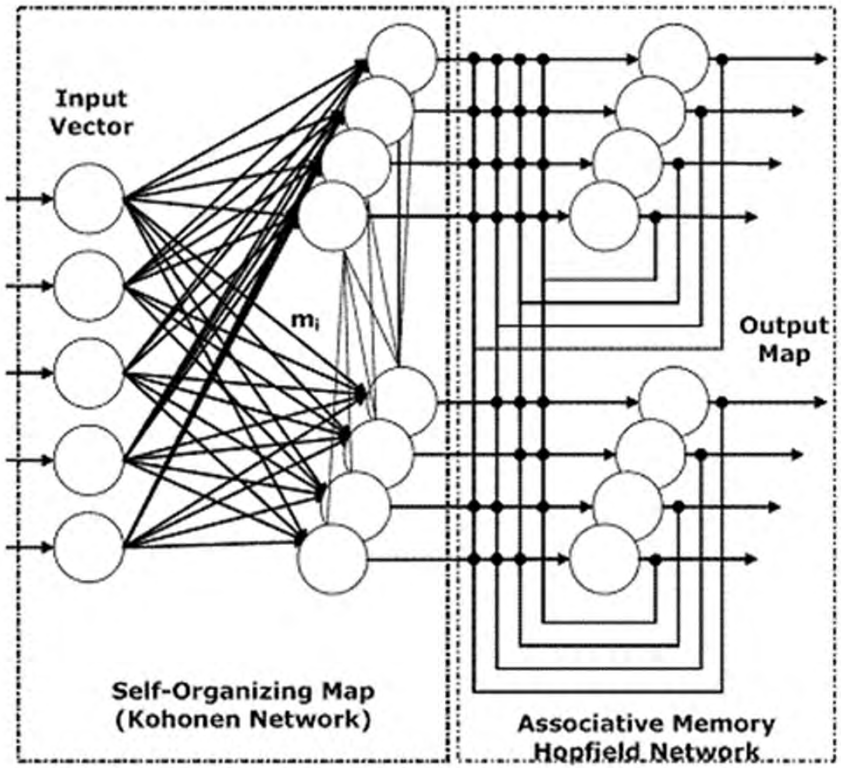


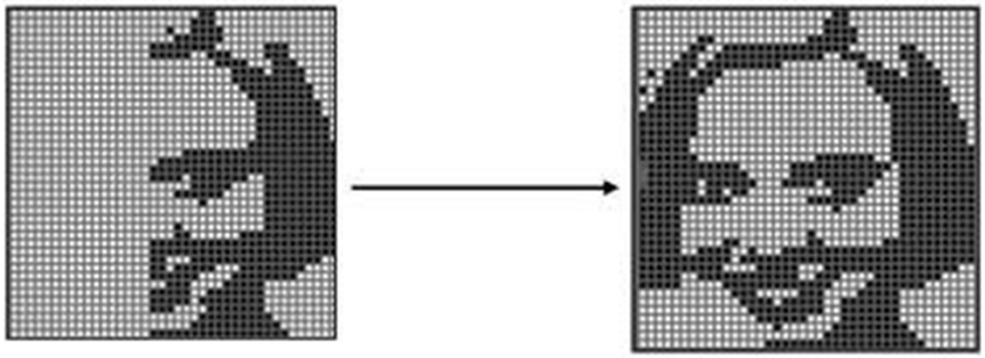
$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} T_{ij} V_i V_j - \sum_i I_i V_i + \sum_i \frac{1}{R_i} \int_0^{V_i} G_i^{-1}(v)\, dv$$

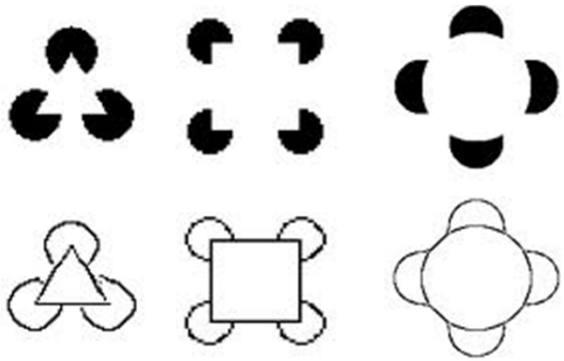
$$\frac{dE}{dt} < 0 \quad !$$

Hopfield networks and dynamics



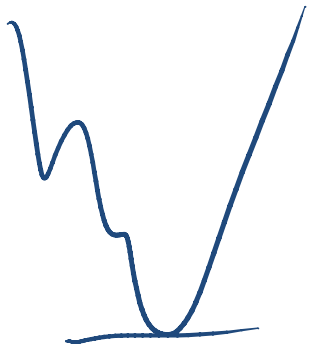






Stochastic approximation

$$\min_{\theta} \int H(\theta, x)\, p(x)\, dx$$



$$J(\theta) = \int H(\theta, x)\, p(x)\, dx$$

$$\nabla J(\theta) = \int \nabla H(\theta, x)\, p(x)\, dx = 0$$

$$\alpha \cdot \nabla J(\theta) = \alpha \cdot 0$$

$$\theta + \alpha \cdot \nabla J(\theta) = \theta$$

$$x^{(i)} \sim p(x)$$

$$\nabla J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla H(x^{i}; \theta)$$

Stochastic approximation

$$\theta = \theta - \alpha \nabla J(\theta)$$

$$\mu_T = \frac{1}{T}\sum_{t=1}^{T} x_t$$

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{N}\sum_{i=1}^{N} \nabla H(\theta_t, x^i)$$

$$\mu_T = \frac{1}{T} x_T + \frac{T-1}{T}\cdot\frac{1}{T-1}\sum_{t=1}^{T-1} x_t$$

$$\mu_T = \mu_{T-1}\left(1 - \frac{1}{T}\right) + \frac{1}{T} x_T$$

$$\theta_{t+1} = \theta_t - \alpha \nabla H(\theta_t, x^t)$$

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) + \alpha \underbrace{\left[\nabla J(\theta_t) - \nabla H(\theta_t, x^t)\right]}_{\text{error}}$$

$$\frac{d\theta}{dt} = \alpha \nabla J(\theta)$$

MLE - definition

- The idea of **Maximum Likelihood Estimation (MLE)** is to find the parameters $\boldsymbol{\theta}$ that maximize the probability of the data $\mathcal{D}$ given these parameters:

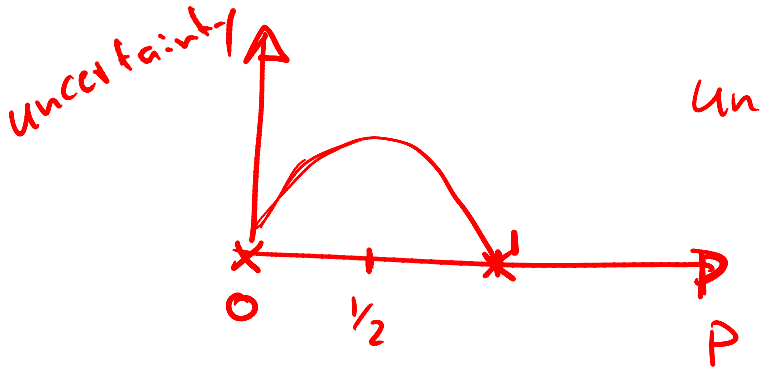
$$\hat{\boldsymbol{\theta}} := \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})$$

- MLE assumes that the data has been generated by a distribution $p(\mathcal{D}|\boldsymbol{\theta}_0)$ for some true parameter $\boldsymbol{\theta}_0$.

MLE - properties

- For independent and identically distributed (i.i.d.) data from $p(x|\boldsymbol{\theta}_0)$, the MLE minimizes the **Kullback-Leibler divergence**:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(x_i|\boldsymbol{\theta}) \\
&= \arg\max_{\boldsymbol{\theta}} \tfrac{1}{N} \sum_{i=1}^{N} \log p(x_i|\boldsymbol{\theta}) - \tfrac{1}{N} \sum_{i=1}^{N} \log p(x_i|\boldsymbol{\theta}_0) \\
&= \arg\max_{\boldsymbol{\theta}} \tfrac{1}{N} \sum_{i=1}^{N} \log \frac{p(x_i|\boldsymbol{\theta})}{p(x_i|\boldsymbol{\theta}_0)} \\
\longrightarrow &\quad \arg\min_{\boldsymbol{\theta}} \int \log \frac{p(x_i|\boldsymbol{\theta}_0)}{p(x_i|\boldsymbol{\theta})} p(x|\boldsymbol{\theta}_0) dx
\end{aligned}
$$



$$\text{uncertainty} = -\sum p \log p$$

$$\int \log \frac{P(x|\theta_0)}{P(x|\theta)} P(x|\theta_0)\, dx$$

$$= \int \log P(x|\theta_0)\, P(x|\theta_0) dx - \int \log P(x|\theta)\, P(x|\theta_0) dx$$

MLE - properties

- Under smoothness and identifiability assumptions, the MLE is **consistent**:

$$\hat{\boldsymbol{\theta}} \xrightarrow{p} \boldsymbol{\theta}_0$$

or equivalently,

$$\text{plim}(\hat{\boldsymbol{\theta}}) = \theta_0$$

or equivalently,

$$\lim_{N \to \infty} P(|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0| > \alpha) \to 0$$

for every α.

MLE - properties

Math Stat. John Rice

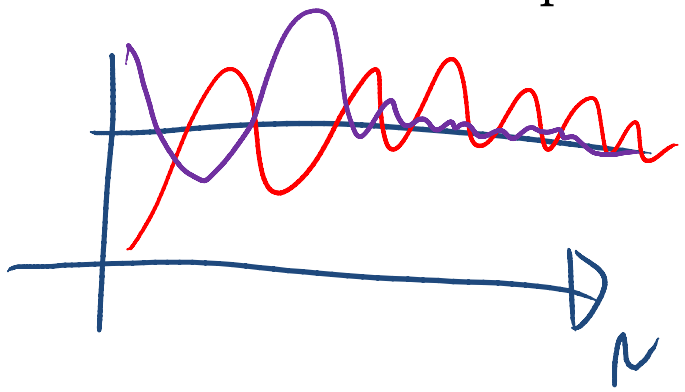
- The MLE is **asymptotically normal**. That is, as $N \to \infty$, we have:

$$\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0 \Longrightarrow N(0, I^{-1})$$

  where I is the **Fisher Information matrix**.

- It is asymptotically optimal or **efficient**. That is, asymptotically, it has the lowest variance among all well behaved estimators. In particular it attains a lower bound on the CLT variance known as the **Cramer-Rao lower bound**.

- But what about issues like robustness and computation? Is MLE always the right option?



Bias and variance

- Note that the estimator is a function of the data: $\hat{\boldsymbol{\theta}} = g(\mathcal{D})$.
- Its **bias** is:

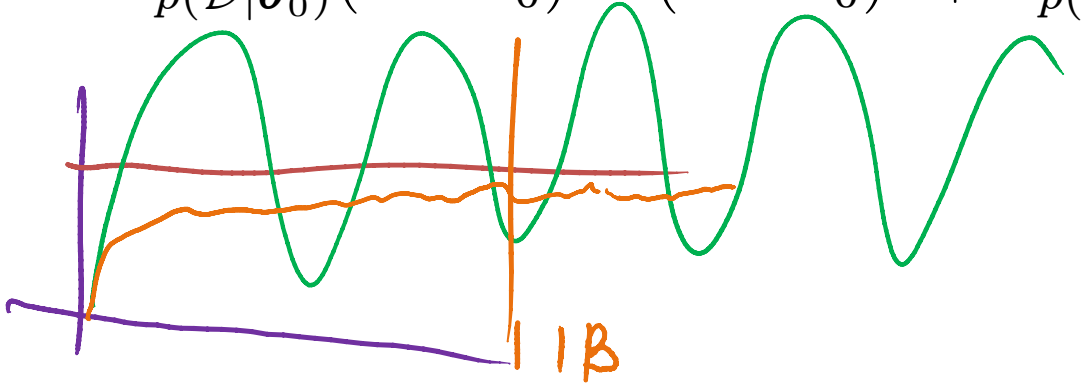
$$bias(\hat{\boldsymbol{\theta}}) = \mathbb{E}_{p(\mathcal{D}|\boldsymbol{\theta}_0)}(\hat{\boldsymbol{\theta}}) - \boldsymbol{\theta}_0 = \bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_0$$

- Its **variance** is:

$$\mathbb{V}(\hat{\boldsymbol{\theta}}) = \mathbb{E}_{p(\mathcal{D}|\boldsymbol{\theta}_0)}(\hat{\boldsymbol{\theta}} - \bar{\boldsymbol{\theta}})^2$$

- Its **mean squared error** is:

$$\text{MSE} = \mathbb{E}_{p(\mathcal{D}|\boldsymbol{\theta}_0)}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)^2 = (\bar{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)^2 + \mathbb{E}_{p(\mathcal{D}|\boldsymbol{\theta}_0)}(\hat{\boldsymbol{\theta}} - \bar{\boldsymbol{\theta}})^2$$



MLE for the univariate Gaussian distribution

- In the case of iid data sampled from a univariate Gaussian, the log-likelihood is given by

$$\ell(\mu, \sigma^2) = \sum_{i=1}^{N} \log \mathcal{N}(x_i|\mu, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - \mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

- To find the maximum of this function, we set the partial derivatives to 0 and solve. (We should check that the second derivative is positive.)

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i = \overline{x}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \overline{x})^2 = \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right) - (\overline{x})^2$$

- The quantities $\sum_i x_i$, $\sum_i x_i^2$ and N are called the **sufficient statistics** of the data, since they capture all the relevant information needed for estimating the parameter.

MLE for the Bernoulli distribution

- We toss a coin N times and record the sequence of heads and tails, $\mathcal{D} = (x_1, x_2, \ldots, x_N)$. How do we estimate the probability of heads from this?

- The log-likelihood is given by

$$\ell(\theta) = \sum_{i=1}^{N} \log \mathrm{Ber}(x_i|\theta) = \sum_{i=1}^{N} \log \left[\theta^{x_i}(1-\theta)^{1-x_i}\right] = N_1 \log\theta + N_2 \log(1-\theta)$$

  where $N_1 = \sum_i x_i$ is the number of heads and $N_2 = \sum_i (1 - x_i)$ is the number of tails (these are the sufficient statistics).

- To find the MLE, we find the maximum of this expression as follows:

$$\frac{d\ell}{d\theta} = \frac{N_1}{\theta} - \frac{N_2}{1-\theta} = 0$$

$$\hat{\theta} = \frac{N_1}{N_1 + N_2}$$

  where $N_1 + N_2 = N$.

MLE for binary logistic regression

- Recall that binary logistic regression has the form

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \text{Ber}(y_i|\text{sigm}(\mathbf{w}^T\mathbf{x}_i))$$

where $\text{sigm}(\eta) = 1/(1+e^{-\eta})$ and $y_i \in \{0, 1\}$. Let $\pi_i = \text{sigm}(\mathbf{w}^T\mathbf{x}_i)$.

- Then the negative log-likelihood of all the data is given by

$$\begin{aligned} J(\mathbf{w}) &= -\sum_{i=1}^{N} \log[\pi_i^{\mathbb{I}(y_i=1)} \times (1-\pi_i)^{\mathbb{I}(y_i=0)}] \\ &= -\sum_{i=1}^{N} [y_i \log \pi_i + (1-y_i)\log(1-\pi_i)] \end{aligned}$$

This is also called the **cross-entropy** error function.

MLE for binary logistic regression

- The gradient and Hessian of $J(\mathbf{w})$ are given by:

$$\mathbf{g}(\mathbf{w}) = \frac{d}{d\mathbf{w}} J(\mathbf{w}) = \sum_i (\pi_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\pi} - \mathbf{y})$$

$$\mathbf{H} = \frac{d}{d\mathbf{w}} \mathbf{g}(\mathbf{w})^T = \sum_i (\nabla_{\mathbf{w}} \pi_i)\mathbf{x}_i^T = \sum_i \pi_i(1-\pi_i)\mathbf{x}_i\mathbf{x}_i^T = \mathbf{X}^T \mathrm{diag}(\pi_i(1-\pi_i))\mathbf{X}$$

- One can show that $\mathbf{H}$ is positive definite; hence the NLL is **convex** and has a unique global minimum.
- To find this minimum, we will however have to learn a few things about optimization.

Revision: Gradient and Hessian

- Let $\mathbf{x}$ be an n-dimensional vector, and $f(\mathbf{x})$ a scalar-valued function. The gradient vector of f with respect to $\mathbf{x}$ is the following vector:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Derivative of a scalar product

$$\nabla_{\mathbf{x}} \ \mathbf{a}^T \mathbf{x} = \mathbf{a}$$

Derivative of a quadratic form

$$\nabla_{\mathbf{x}} \ \mathbf{x}^T \mathbf{A} \mathbf{x} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

If $\mathbf{A}$ is symmetric, this becomes $\nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{A} \mathbf{x} = 2\mathbf{A}\mathbf{x}$.

- The **Hessian** matrix of a scalar valued function with respect to $\mathbf{x}$, written $\nabla^2_{\mathbf{x}} f(\mathbf{x})$ or simply as $\mathbf{H}$, is the $n \times n$ matrix of partial derivatives,

$$\nabla^2_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_1} & \frac{\partial^2 f(\mathbf{x})}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_n^2} \end{bmatrix}$$

  Obviously this is a symmetric matrix, since

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i}.$$

- We can think of the Hessian as the gradient of the gradient, which can be written as

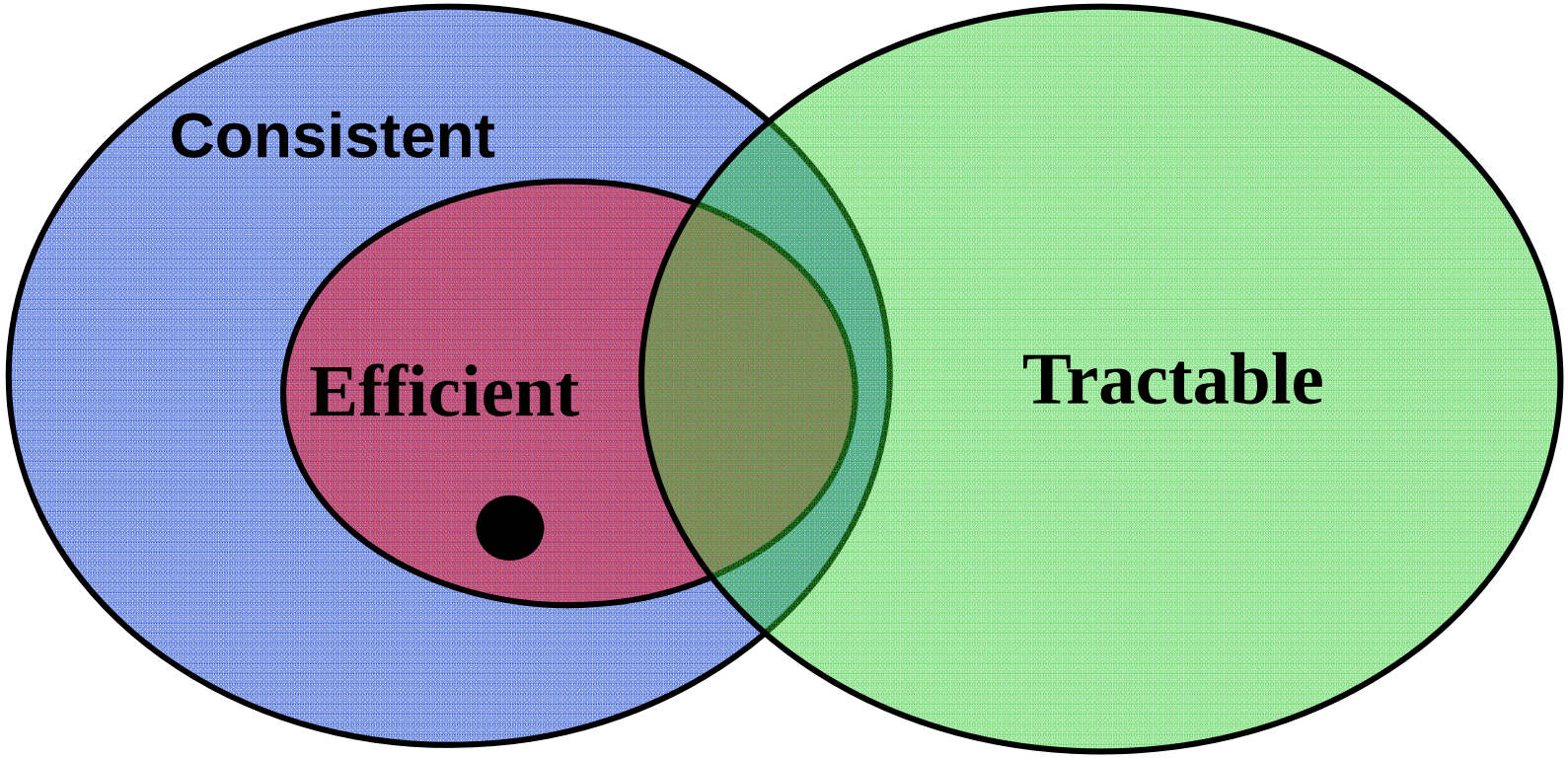
$$\mathbf{H} = \nabla_{\mathbf{x}} (\nabla^T_{\mathbf{x}} f(\mathbf{x}))$$

- For the quadratic form $f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}$, the gradient vector is $(\mathbf{A} + \mathbf{A}^T)\mathbf{x}$, so the Hessian is $\mathbf{A} + \mathbf{A}^T$. If $\mathbf{A}$ is symmetric, this is $2\mathbf{A}$.

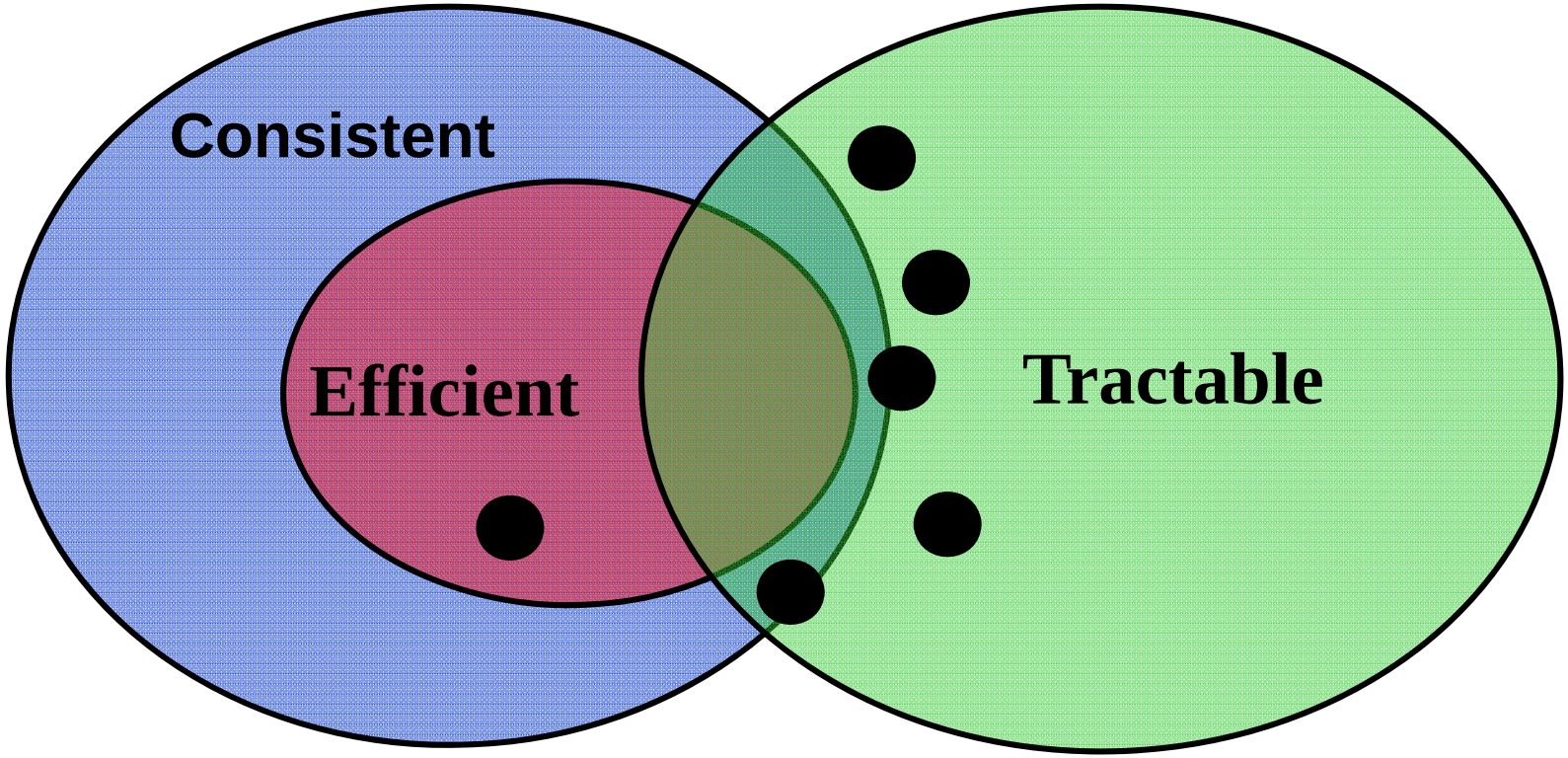
Problems with MLE

- Computation can be expensive.
- It suffers from overfitting.
- It provides a point-estimate (best guess) of the parameters. I does not give any measure of uncertainty in this guess.
- It migh not be the best option when the data generating mechanism is outside the model class. Even if it finds the closest solution in terms of KL divergence, other distance metrics might be more appropriate.

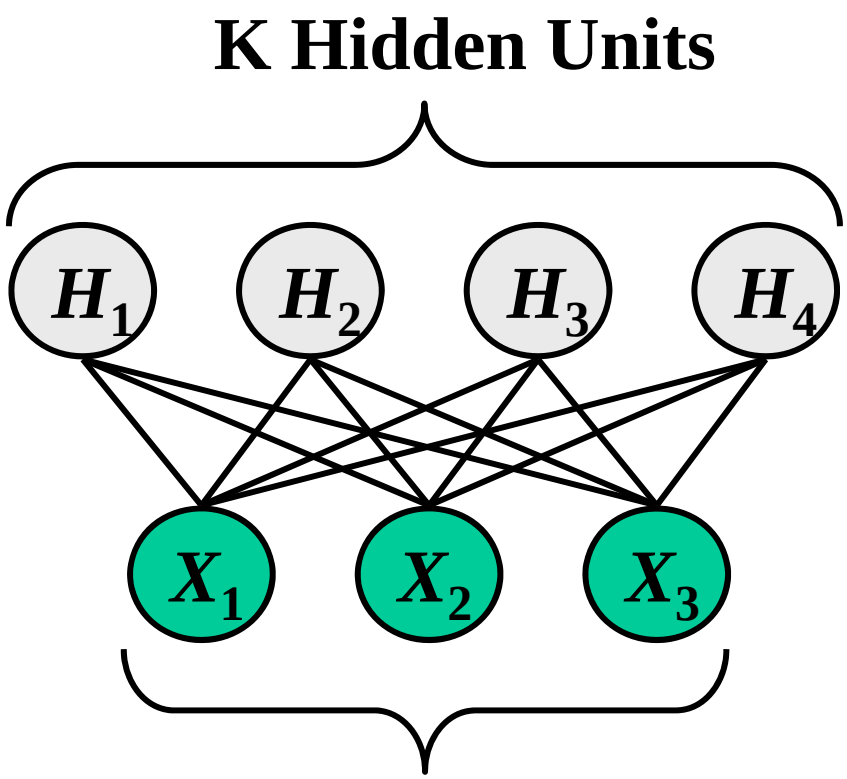
Maximum Likelihood Estimation is statistically consistent and efficient but is not computationally tractable for many models of interest like RBM's, MRF's, CRF's due to the partition function.



Many more tractable estimators have been proposed: Pseudo/Composite-likelihood [Besag, 1975], ratio matching [Hyvärinen, 2007], generalized score matching [Lyu, 2009], non-local contrastive objectives [Vickrey, 2010], minimum Probability Flow [Sohl-Dickstein, 2010], …



Example: **Restricted Boltzmann Machines**



A Restricted Boltzmann Machine (RBM) is a Boltzmann Machine with a bipartite graph structure.

Typically one layer of nodes are fully observed variables (the visible layer), while the other consists of latent variables (the hidden layer).

Example: **Restricted Boltzmann Machines**

The joint probability of the visible and hidden variables is defined through a bilinear energy function.

$$E_\theta(\boldsymbol{x}, \boldsymbol{h}) = -(\boldsymbol{x}^T W \boldsymbol{h} + \boldsymbol{x}^T b + \boldsymbol{h}^T c)$$

$$P_\theta(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{\mathcal{Z}} \exp\left(-E_\theta(\boldsymbol{x}, \boldsymbol{h})\right)$$

$$\mathcal{Z} = \sum_{\boldsymbol{x}' \in \mathcal{X}} \sum_{\boldsymbol{h}' \in \mathcal{H}} \exp\left(-E_\theta(\boldsymbol{x}', \boldsymbol{h}')\right)$$

Example: **Restricted Boltzmann Machines**

The bipartite graph structure gives the RBM a special property: the visible variables are conditionally independent given the hidden variables and vice versa.

$$P_\theta(x_d = 1|\boldsymbol{h}) = \frac{1}{1 + \exp(-(\sum_{k=1}^{K} W_{dk} h_k + x_d b_d))}$$

$$P_\theta(h_k = 1|\boldsymbol{x}) = \frac{1}{1 + \exp(-(\sum_{d=1}^{D} W_{dk} x_d + h_k c_k))}$$

Example: **Restricted Boltzmann Machines**

The marginal probability of the visible vector is obtained by summing out over all joint states of the hidden variables.

$$P_\theta(\boldsymbol{x}) = \frac{1}{\mathcal{Z}} \sum_{h \in \mathcal{H}} \exp\left(-E_\theta(\boldsymbol{x}, \boldsymbol{h})\right)$$

$$P_\theta(\boldsymbol{x}) = \frac{1}{\mathcal{Z}} \exp\left(-F_\theta(\boldsymbol{x})\right)$$

$$F_\theta(\boldsymbol{x}) = -\left(\boldsymbol{x}^T b + \sum_{k=1}^{K} \log\left(1 + \exp\left(\boldsymbol{x}^T W_k + c_k\right)\right)\right)$$

Example: **Restricted Boltzmann Machines**

This construction eliminates the latent, hidden variables, leaving a distribution defined in terms of the visible variables.

However, computing the normalizing constant (partition function) still has exponential complexity in D.

$$\mathcal{Z} = \sum_{\boldsymbol{x}' \in \mathcal{X}} \exp\left(-F_\theta(\boldsymbol{x}')\right)$$

Stochastic Maximum Likelihood

$x^i \sim P_e(x)$

Exact maximum likelihood learning is intractable in an RBM. Stochastic ML estimation can instead be applied, usually using a simple block Gibbs sampler.

$$f^{ML}(\theta) = \sum_{\boldsymbol{x} \in \mathcal{X}} P_e(\boldsymbol{x}) \log P_\theta(\boldsymbol{x}) \approx \frac{1}{N} \sum_{i=1}^{N} \log P_\theta(x_i)$$

$$\nabla f^{ML} \approx -\left(\frac{1}{N} \sum_{n=1}^{N} \nabla F_\theta(\boldsymbol{x}_n) - \frac{1}{S} \sum_{s=1}^{S} \nabla F_\theta(\tilde{\boldsymbol{x}}_s) \right)$$

Contrastive Divergence

The contrastive divergence principle results in a gradient that looks identical to stochastic maximum likelihood. The difference is that CD samples from the T-step Gibbs distribution.

$$f^{CD}(\theta) = \sum_{\boldsymbol{x} \in \mathcal{X}} P_e(\boldsymbol{x}) \log \left(\frac{P_e(\boldsymbol{x})}{P_\theta(\boldsymbol{x})} \right) - Q_\theta^t(\boldsymbol{x}) \log \left(\frac{Q_\theta^t(\boldsymbol{x})}{P_\theta(\boldsymbol{x})} \right)$$

$$\nabla f^{CD} \approx -\frac{1}{N} \left(\sum_{n=1}^{N} \nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\tilde{\boldsymbol{x}}_n) \right)$$

Pseudo-Likelihood

The principle of maximum pseudo-likelihood is based on optimizing a product of one-dimensional conditional densities under a log loss.

$$f^{PL}(\theta) = \sum_{\boldsymbol{x} \in \mathcal{X}} \sum_{d=1}^{D} P_e(\boldsymbol{x}) \log P_\theta(x_d | \boldsymbol{x}_{-d})$$

$$\nabla f^{PL} = \frac{-1}{N} \sum_{n,d} P_\theta(\bar{\boldsymbol{x}}_{dn}^d | \boldsymbol{x}_{-dn}) \left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d) \right)$$

Ratio Matching

The ratio matching principle is very similar to pseudo-likelihood, but is based on minimizing a squared difference between one dimensional conditional distributions.

$$f^{RM}(\theta) = \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{d=1}^{D} \sum_{\xi\in\{0,1\}} P_e(\boldsymbol{x}) \Big(P_\theta(X_d = \xi | \boldsymbol{x}_{-d}) - P_e(X_d = \xi | \boldsymbol{x}_{-d}) \Big)^2$$

$$\nabla f^{RM} = \frac{2}{N} \sum_{n=1}^{N} \sum_{d=1}^{D} g(u_{dn})^3 u_{dn} \left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d) \right)$$

$$g(u) = \frac{1}{1+u}, \quad u_{dn} = P_\theta(\boldsymbol{x}_n) / P_\theta(\bar{\boldsymbol{x}}_n^d)$$

Generalized Score Matching

The generalized score matching principle is similar to ratio matching, except that the difference between inverse one dimensional conditional distributions is minimized.

$$f^{GSM}(\theta) = \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{d=1}^{D} P_e(\boldsymbol{x}) \left(\frac{1}{P_\theta(x_d|\boldsymbol{x}_{-d})} - \frac{1}{P_e(x_d|\boldsymbol{x}_{-d})} \right)^2$$

$$\nabla f^{GSM} = \frac{2}{N} \sum_{n=1}^{N} \sum_{d=1}^{D} (u_{dn}^{-2} - u_{dn}) \left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d) \right)$$

$$g(u) = u^{-2} - 2u, \quad u_{dn} = P_\theta(\boldsymbol{x}_n)/P_\theta(\bar{\boldsymbol{x}}_n^d)$$

Gradient Comparison

$$\nabla f^{ML} \approx -\left(\frac{1}{N}\sum_{n=1}^{N}\nabla F_\theta(\boldsymbol{x}_n) - \frac{1}{S}\sum_{s=1}^{S}\nabla F_\theta(\tilde{\boldsymbol{x}}_s)\right)$$

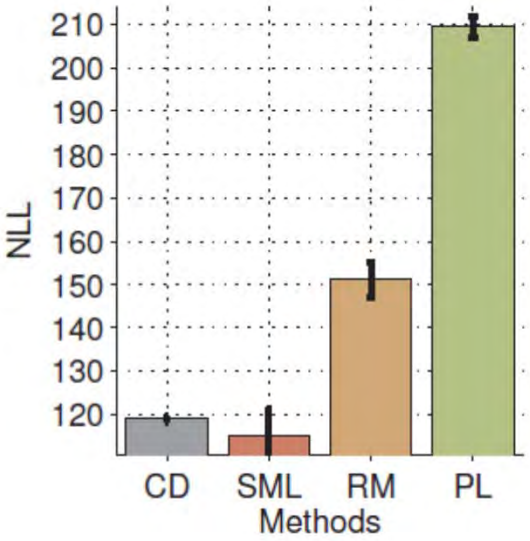
$$\nabla f^{CD} \approx -\frac{1}{N}\left(\sum_{n=1}^{N}\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\tilde{\boldsymbol{x}}_n)\right)$$

$$\nabla f^{PL} = \frac{-1}{N}\sum_{n,d} P_\theta(\bar{\boldsymbol{x}}_{dn}^d|\boldsymbol{x}_{-dn})\left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d)\right)$$

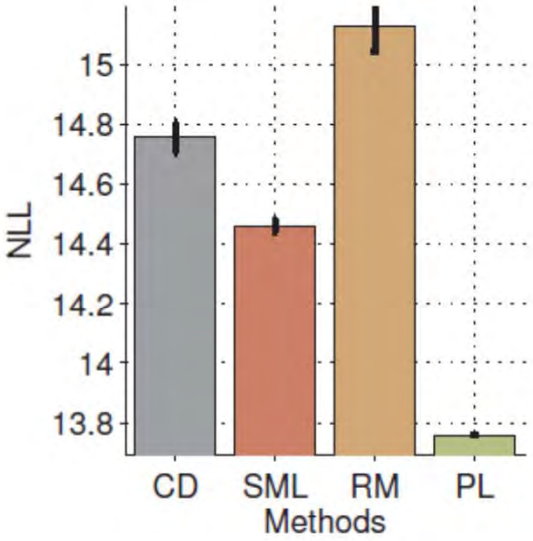
$$\nabla f^{RM} = \frac{2}{N}\sum_{n=1}^{N}\sum_{d=1}^{D} g(u_{dn})^3 u_{dn}\left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d)\right)$$

$$\nabla f^{GSM} = \frac{2}{N}\sum_{n=1}^{N}\sum_{d=1}^{D} (u_{dn}^{-2} - u_{dn})\left(\nabla F_\theta(\boldsymbol{x}_n) - \nabla F_\theta(\bar{\boldsymbol{x}}_n^d)\right)$$

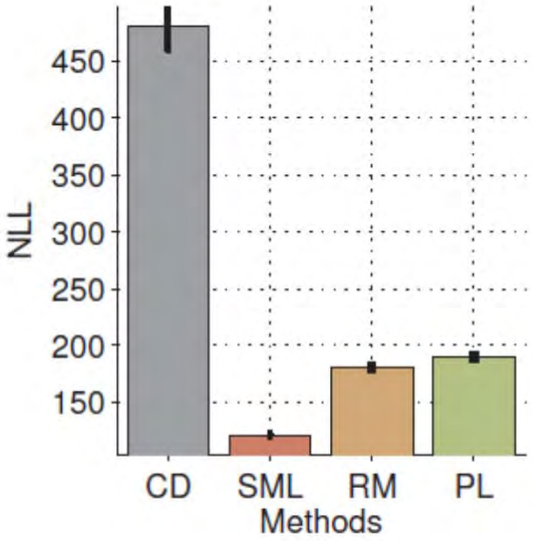
Experiments: **Log Likelihood**



(a) MNIST



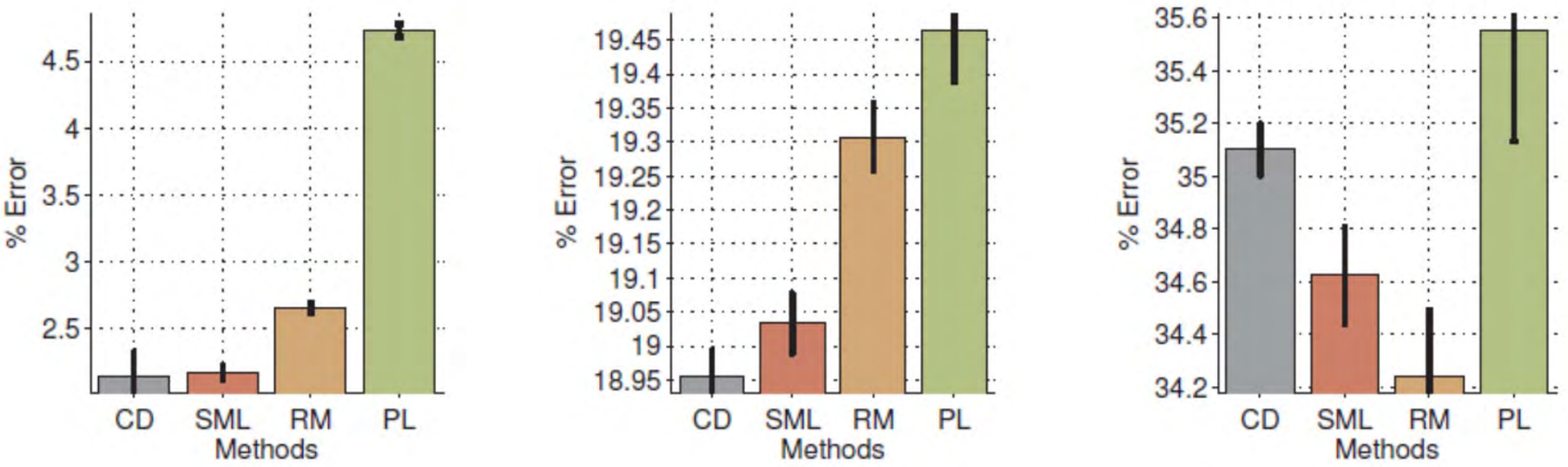
(b) 20News



(c) CalTech

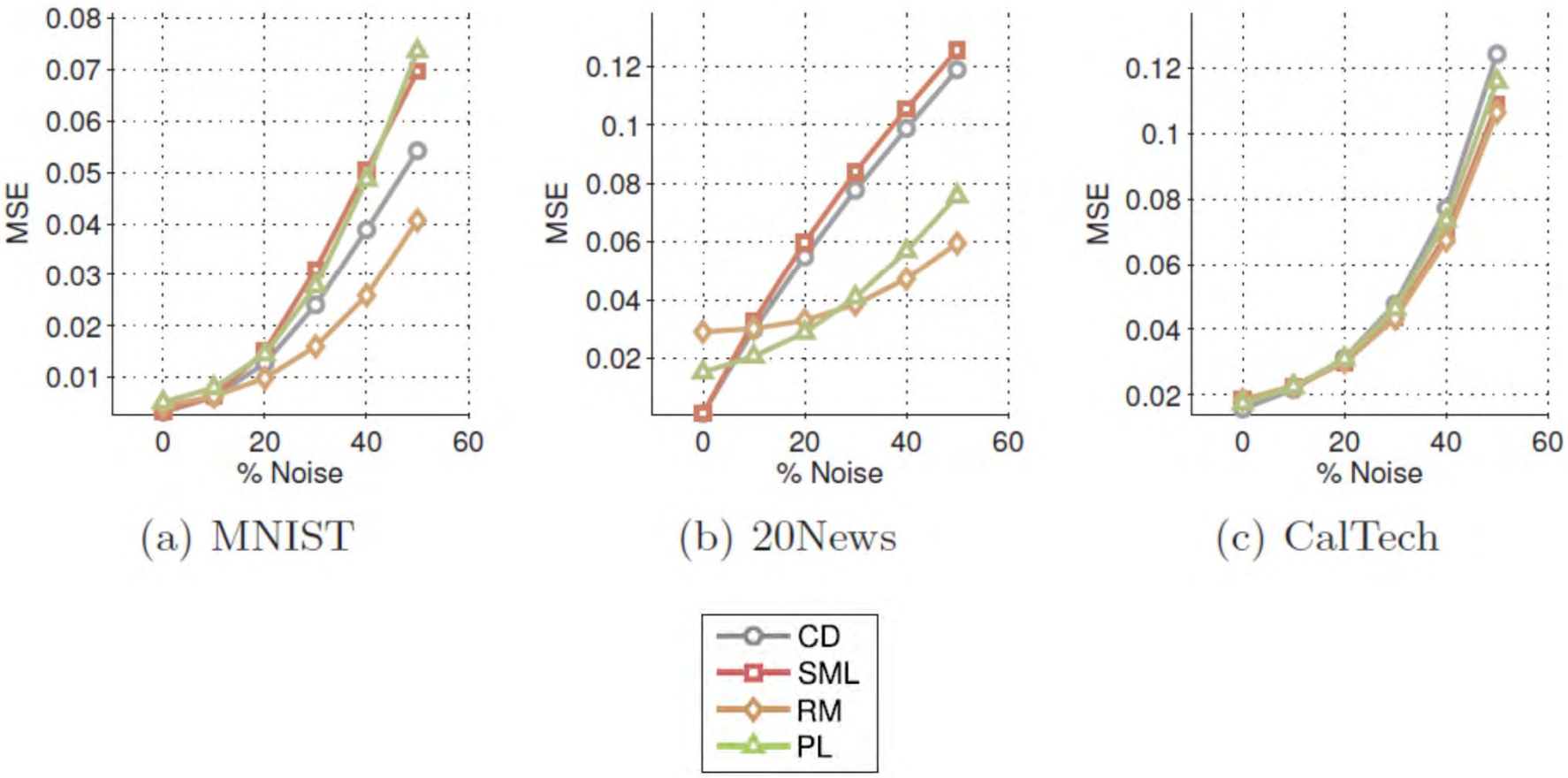
[Marlin, Swersky, Chen & dF, 2010]

Experiments: **Classification Error**



[Marlin, Swersky, Chen & dF, 2010]

Experiments: **De-noising**



[Marlin, Swersky, Chen & dF, 2010]

Theoretical results

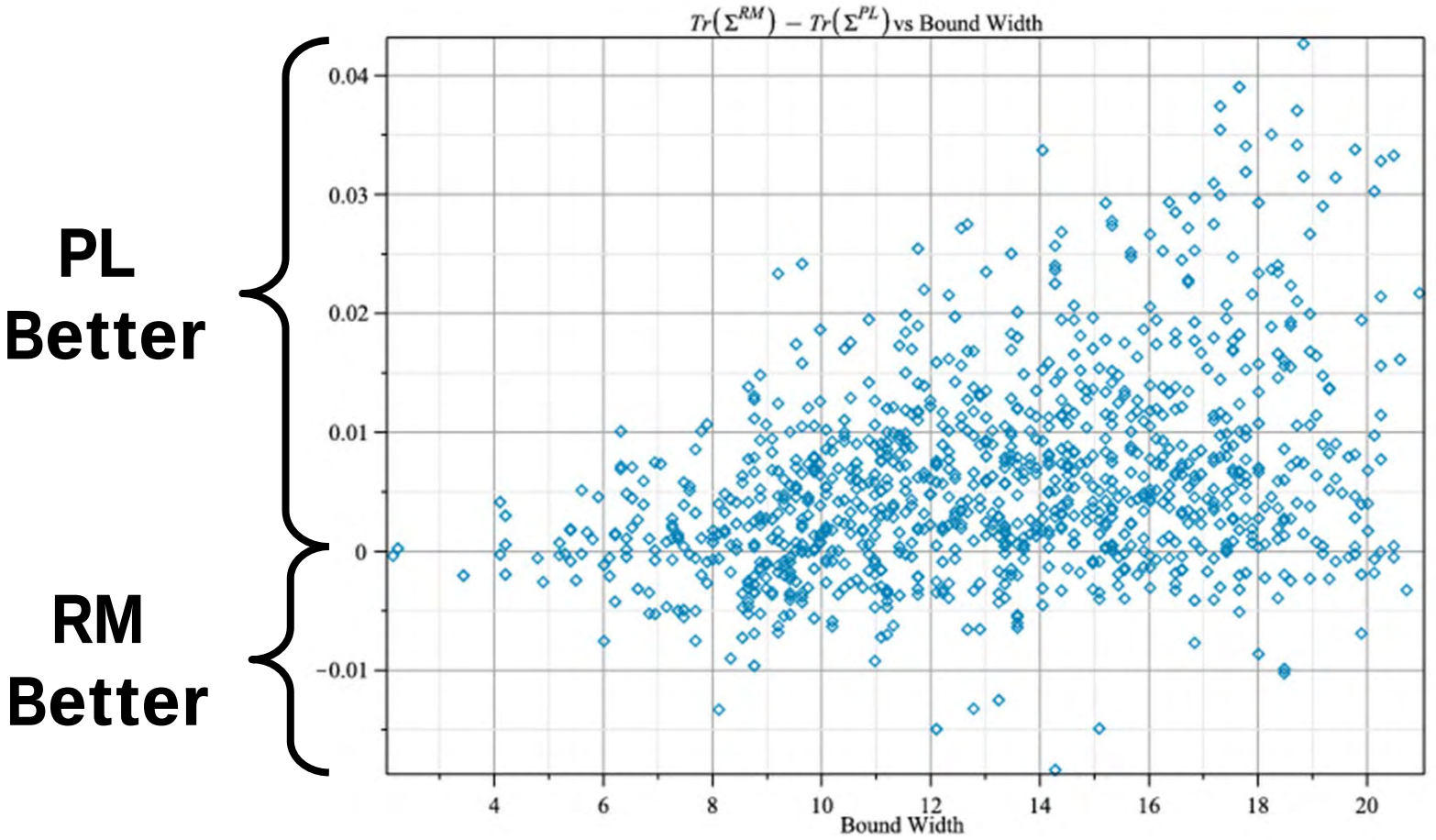
Consistency: PL and RM are both asymptotically consistent when the model is well-specified.

Efficiency: Neither PL nor RM is strictly more efficient than the other. We have a partial understanding of the relationship between their asymptotic covariance matrices.

$$\frac{v_{min}^2}{p_{max}^4}\Sigma^{PL} \leq \Sigma^{RM} \leq \frac{v_{max}^2}{p_{min}^4}\Sigma^{PL}$$

[Marlin & dF, 2011]

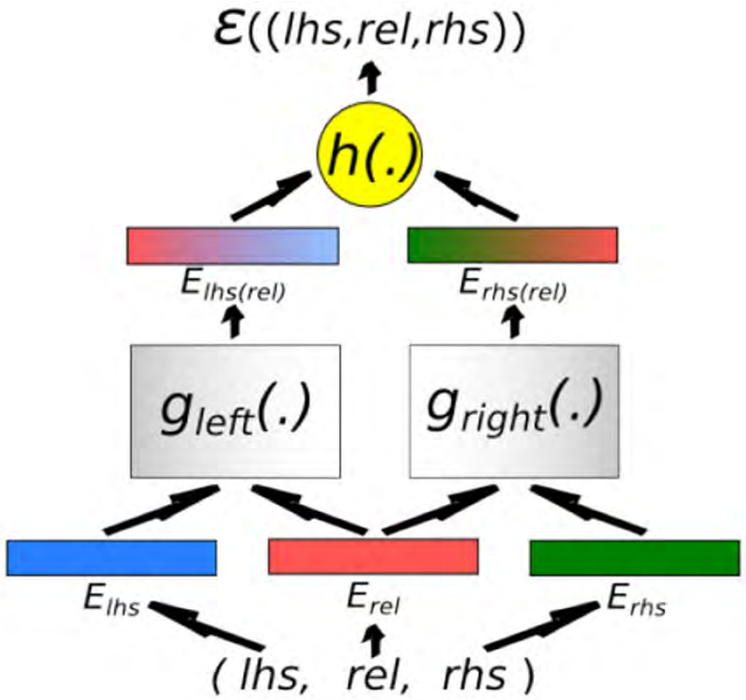
Discrete MRFs: Efficiency Simulations



Max-margin learning

We have seen several examples of the ranking loss in this summer school, e.g. semantic learning, NLP:

$$\sum_{x \in \mathcal{D}} \sum_{\tilde{x} \sim Q(\tilde{x}|x)} \max\left(\mathcal{E}(x) - \mathcal{E}(\tilde{x}) + 1, 0\right)$$



$Q(\tilde{x}|x)$ is a corruption process

[Bordes, Glorot, Weston & Bengio, **2012**]

Max-margin learning

The principle of contrasting is very useful for unsupervised feature learning.

But we must always choose a "negative set" with respect to which we contrast the "positive set" (the given data)

Can we optimize this process of choosing the negative set?

Let us again use an RBM as a running example

$$p_W(\boldsymbol{x}) = \sum_{\boldsymbol{h}} \frac{\exp\left(\sum_{d=1}^{D}\sum_{k=1}^{K} W_{dk} x_d h_k\right)}{\sum_{\boldsymbol{x}'}\sum_{\boldsymbol{h}'} \exp\left(\sum_{d=1}^{D}\sum_{k=1}^{K} W_{dk} x'_d h'_k\right)}$$

$$\begin{aligned}
p_W(\boldsymbol{x}) &\propto \sum_{\boldsymbol{h}} \exp\left(\sum_{d=1}^{D}\sum_{k=1}^{K} W_{dk} x_d h_k\right) \\
&= \sum_{\boldsymbol{h}} \prod_{k=1}^{K} \exp\left(\sum_{d=1}^{D} W_{dk} x_d h_k\right) \\
&= \prod_{k=1}^{K} \sum_{h=0}^{1} \exp\left(\sum_{d=1}^{D} W_{dk} x_d h\right) \\
&= \prod_{k=1}^{K} \left(1 + \exp\left(\sum_{d=1}^{D} W_{dk} x_d\right)\right)
\end{aligned}$$

Max-margin learning

Make the data more probable than anything else

$$W^* = \arg\min_W \; W^T W$$
$$\text{st } \forall n \; \forall \boldsymbol{x} \;\; p_W(\boldsymbol{x}_n) \geq p_W(\boldsymbol{x})$$

$$p_W(x) = \frac{\exp(-E_W(\boldsymbol{x}))}{\sum_{\boldsymbol{x}'} \exp(-E_W(\boldsymbol{x}'))}$$
$$E_W(\boldsymbol{x}) = -\sum_{k=1}^{K} \log\left(1 + \exp\left(\sum_{d=1}^{D} W_{dk} x_d\right)\right)$$

[Marlin 2010]

Max-margin learning

We write the constraint in terms of energy, introduce a margin α and slack variables η:

$$p_W(\boldsymbol{x}_n) \geq p_w(\boldsymbol{x})$$

$$\log(p_W(\boldsymbol{x}_n)) \geq \log(p_w(\boldsymbol{x}))$$

$$-E_W(\boldsymbol{x}_n) - \sum_{\boldsymbol{x}'} \exp(-E_W(\boldsymbol{x}')) \geq -E_W(\boldsymbol{x}) - \sum_{\boldsymbol{x}'} \exp(-E_W(\boldsymbol{x}'))$$

$$-E_W(\boldsymbol{x}_n) \geq -E_W(\boldsymbol{x})$$

$$W^* = \arg\min_W \; W^T W + \lambda \sum_{n=1}^{N} \sum_{\boldsymbol{x}} \eta_n \boldsymbol{x}$$

$$\text{st } \forall n \; \forall \boldsymbol{x} \;\; E_W(\boldsymbol{x}) - E_W(\boldsymbol{x}_n) \geq \alpha - \eta_n \boldsymbol{x}$$

Max-margin learning

$$W^* = \arg\min_W \; W^T W + \lambda \sum_{n=1}^{N} \sum_{\boldsymbol{x}} \eta_n \boldsymbol{x}$$

$$\text{st } \forall n \; \forall \boldsymbol{x} \quad E_W(\boldsymbol{x}) - E_W(\boldsymbol{x}_n) \geq \alpha - \eta_n \boldsymbol{x}$$

This objective has an exponential number of constraints. We can obtain a linear number of constraints by contrasting against the configuration of lowest energy

$$W^* = \arg\min_W \; W^T W + \lambda \sum_{n=1}^{N} \eta_n$$

$$\text{st } \forall n \quad \min_{\boldsymbol{x}} E_W(\boldsymbol{x}) - E_W(\boldsymbol{x}_n) \geq \alpha - \eta_n$$

Max-margin learning

A common trick is to rewrite the objective using a hinge loss: $h(y, \alpha) = \max(0, \alpha - y)$

$$W^* = \arg\min_W \sum_{n=1}^{N} h(\Delta(\boldsymbol{x}_n, \boldsymbol{x}_W^*), \alpha) + cW^T W$$

$$\Delta_W(\boldsymbol{x}, \boldsymbol{x}') = E_W(\boldsymbol{x}') - E_W(\boldsymbol{x})$$

$$\boldsymbol{x}_W^* = \arg\min_{\boldsymbol{x}} E_W(\boldsymbol{x})$$

So we get back to the ranking objective that Jason Weston and Yoshua Bengio introduced, but contrasting against a different "dataset".

Is this discussion of statistical estimators of any relevance in the continuous case?

Discriminative autoencoders (neural nets devoid of probability) seem to do a reasonable job already

Unification and design

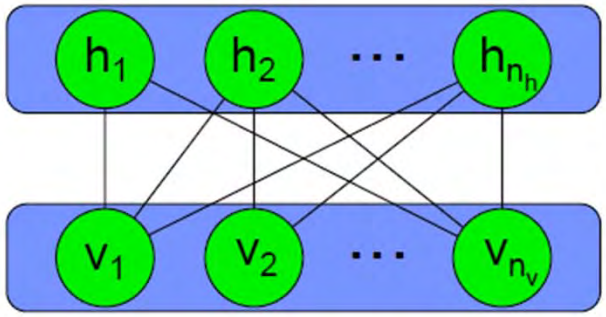
- Autoencoders and energy based models represent the state-of-the-art for feature learning in several domains
- We provide some unifying theory for these model families
- This will allow us to derive new autoencoder architectures from energy based models
- It will also give us an interesting way to regularize these autoencoders

Probabilistic EBMs

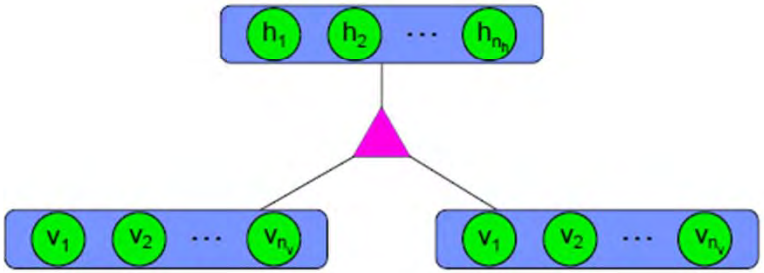
- A probability distribution over data vectors v, with latent variables h and parameters θ
- Defined using an *energy function* $E_\theta(v,h)$ as:

$$P_\theta(v) = \frac{\exp(-E_\theta(v,h))}{Z(\theta)}$$

- Normalized by the *partition function* $Z(\theta) = \int_v \int_h \exp(-E_\theta(v,h))dh\,dv$
- $Z(\theta)$ is usually not analytical
- Features given by h



Restricted Boltzmann Machine



Factored EBM (e.g. mPoT)

Example: **Gaussian-Bernoulli RBMs**

An Gaussian-Bernoulli restricted Boltzmann machine is an energy based model with visible (observed) variables $v \in \mathbb{R}^D$, hidden (latent) variables $h \in \{0,1\}^K$, and parameters $\theta = \{W\}$. The energy is given by:

$$E_\theta(v,h) = \frac{1}{2} v^T v - v^T W h$$

The free-energy is obtained by marginalizing over the hidden variables:

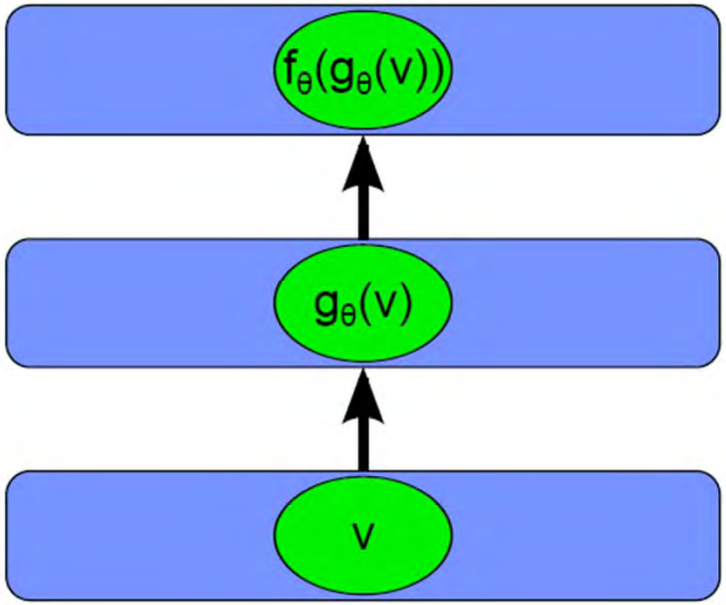
$$F_\theta(v) = \frac{1}{2} v^T v - \sum_{j=1}^{K} \log\left(1 + \exp\left(v^T W_j\right)\right)$$

Deterministic autoencoders

- A feed-forward neural network designed to reconstruct its input
- Encoding function: $g_\theta(v)$
- Reconstruction function: $f_\theta(g_\theta(v))$
- Minimizes reconstruction error:

$$\hat{\theta} = \arg\min_\theta ||f_\theta(g_\theta(v)) - v||^2$$

- Features given by $g_\theta(v)$



Vanilla 1-layer autoencoder

A commonly used autoencoder architecture maps inputs $v \in \mathbb{R}^D$ to outputs $\hat{v} \in \mathbb{R}^D$ through a deterministic hidden layer $\hat{h} \in [0,1]^K$ using parameters $\theta = \{W\}$ by the following system:

$$\hat{v} = f_\theta(g_\theta(v)) = W\hat{h}$$
$$\hat{h} = g_\theta(v) = \sigma(W^T v)$$

Where $\sigma(x) = \frac{1}{1+\exp(-x)}$.
We train this by minimizing the reconstruction error:

$$\ell(\theta) = \sum_{i=1}^{D} \frac{1}{2}(v_i - \hat{v}_i)^2$$

Energy based models	Autoencoders
• Undirected • Probabilistic • Slow inference • Intractable objective function	• Directed • Deterministic • Fast inference • Tractable objective function

- Unifying these model families lets us leverage the advantages of each

Score matching

- Minimize distance between the *score functions* $\nabla_v \log(\widetilde{P}(v))$ and $\nabla_v \log(P_\theta(v))$:

$$\hat{\theta} = \arg\min_\theta J(\theta) = \mathbb{E}_{\widetilde{p}(v)} \left[||\nabla_v \log(\widetilde{P}(v)) - \nabla_v \log(P_\theta(v))||^2 \right]$$

- Equivalent to (Hyvarinen, 2006):

$$J(\theta) = \mathbb{E}_{\widetilde{p}(v)} \left[\sum_{i=1}^{D} \frac{1}{2} \left(\frac{\partial \log(P_\theta(v))}{\partial v_i} \right)^2 + \frac{\partial^2 \log(P_\theta(v))}{\partial v_i^2} \right] + const$$

- Does not depend on $Z(\theta)$ since $\nabla_v Z(\theta) = 0$

Denoising autoencoders

$q_0(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\delta(\|\mathbf{x}-\mathbf{x}^{(i)}\|)$

$q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sigma^d}e^{-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}}-\mathbf{x}\|^2}$

Empirical pdf associated with D_n.

Smoothing kernel or noise model: isotropic Gaussian of variance σ^2.

$$\psi(\mathbf{x};\theta) = \frac{\partial \log p(\mathbf{x};\theta)}{\partial \mathbf{x}}$$

$$J_{ESMq}(\theta) = \mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{2}\left\|\psi(\mathbf{x};\theta) - \frac{\partial \log q(\mathbf{x})}{\partial \mathbf{x}}\right\|^2\right]$$

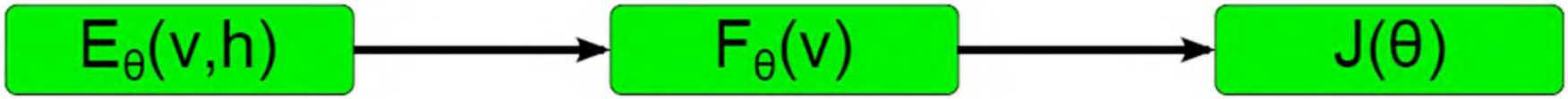
$$J_{DSMq_\sigma}(\theta) = \mathbb{E}_{q_\sigma(\mathbf{x},\tilde{\mathbf{x}})}\left[\frac{1}{2}\left\|\psi(\tilde{\mathbf{x}};\theta) - \frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}}\right\|^2\right]$$

$$\frac{\partial \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})}{\partial \tilde{\mathbf{x}}} = \frac{1}{\sigma^2}(\mathbf{x}-\tilde{\mathbf{x}})$$

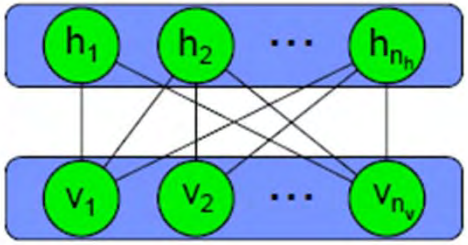
[Vincent, 2011]

Recipe

1. Write down energy function $E_\theta(v, h)$
2. Marginalize over latent variables to form $F_\theta(v)$
3. Derive score matching objective $J(\theta)$
   - In practice, use Theano to do this step!



Example 1: **Gaussian-Bernoulli RBMs**



- We apply score matching to the GBRBM model:

$$J(\theta) = const + \sum_{i=1}^{D} \left[\frac{1}{2} (v_i - \hat{v}_i)^2 + \sum_{j=1}^{K} W_{ij}^2 \hat{h}_j \left(1 - \hat{h}_j\right) \right]$$

- This is also a regularized autoencoder
- The regularizer is a sum of weighted variances of the Bernoulli random variables h conditioned on v

[Swersky, Ranzato, Buchman, Marlin & dF 2011]

Example 2: **mPoT model**

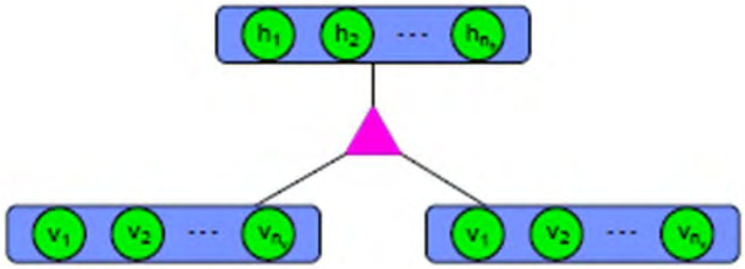
The mPoT is an energy based model with Gaussian-distributed visible units, Gamma-distributed hidden units $h^c \in \mathbb{R}^{+K}$, and Bernoulli-distributed hidden units $h^m \in \{0,1\}^M$ with parameters $\theta = \{C, W\}$.

$$E_\theta(v, h^m, h^c) = \sum_{k=1}^{K} \left[h_k^c (1 + \frac{1}{2}(C_k^T v)^2) + (1-\gamma)\log(h_k^c) \right] + \frac{1}{2} v^T v - v^T W h^m$$

$$F_\theta(v) = \sum_{k=1}^{K} \gamma \log(1 + \frac{1}{2}(\phi_k^c)^2) - \sum_{j=1}^{M} \log(1 + e^{\phi_j^m}) + \frac{1}{2} v^T v$$

$$\phi_k^c = C_k^T v$$

$$\phi_j^m = W_j^T v$$



Example 2: **mPoT model**

$$J(\theta) = \sum_{i=1}^{D} \frac{1}{2} \psi_i(p_\theta(v))^2 + \sum_{k=1}^{K} \left(\rho(\hat{h}_k^c) D_{ik}^2 + \hat{h}_k^c K_{ik} \right) + \sum_{j=1}^{M} \left(\hat{h}_j^m (1 - \hat{h}_j^m) W_{ij}^2 \right) - 1$$

$$\psi_i(p_\theta(v)) = \sum_{k=1}^{K} \hat{h}_k^c D_{ik} + \sum_{j=1}^{M} \hat{h}_j^m W_{ij} - v_i$$

$$K_{ik} = \sum_{k=1}^{K} -C_{ik}^2$$

$$D_{ik} = \sum_{k=1}^{K} \left(-\left(C_k^T v \right) C_{ik} \right)$$

$$\hat{h}_k^c = \frac{\gamma}{1 + \frac{1}{2}(\phi_k^c)^2}$$

$$\hat{h}_j^m = \text{sigm}(\phi_j^m)$$

$$\rho(x) = x^2$$

[Swersky, Ranzato, Buchman, Marlin & dF 2011]

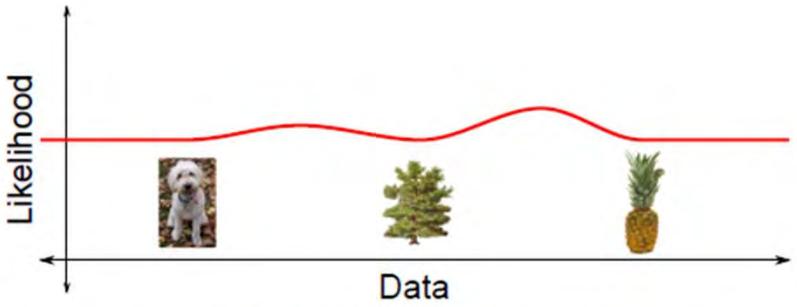
Theorem

The score matching objective for a latent EBM can be expressed succinctly in terms of expectations of the energy with respect to the conditional distribution $P_\theta(h|v)$.

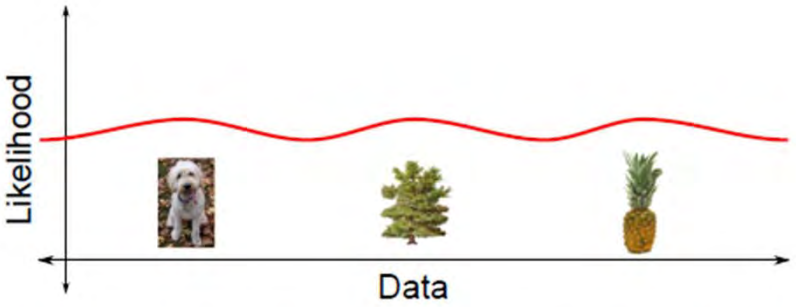
$$J(\theta) = \mathbb{E}_{\tilde{p}(v)} \left[\sum_{i=1}^{n_v} \frac{1}{2} \left(\mathbb{E}_{p_\theta(h|v)} \left[\frac{\partial E_\theta(v,h)}{\partial v_i} \right] \right)^2 + var_{p_\theta(h|v)} \left[\frac{\partial E_\theta(v,h)}{\partial v_i} \right] - \mathbb{E}_{p_\theta(h|v)} \left[\frac{\partial^2 E_\theta(v,h)}{\partial v_i^2} \right] \right].$$

- The score matching objective for a latent EBM always consists of 3 terms:
  1. Squared error
  2. Variance of error
  3. Curvature
- Variance and curvature can be seen as regularizers

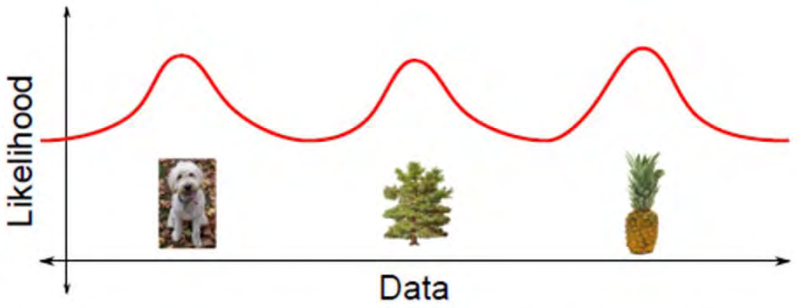
Intuition



Model probability before learning

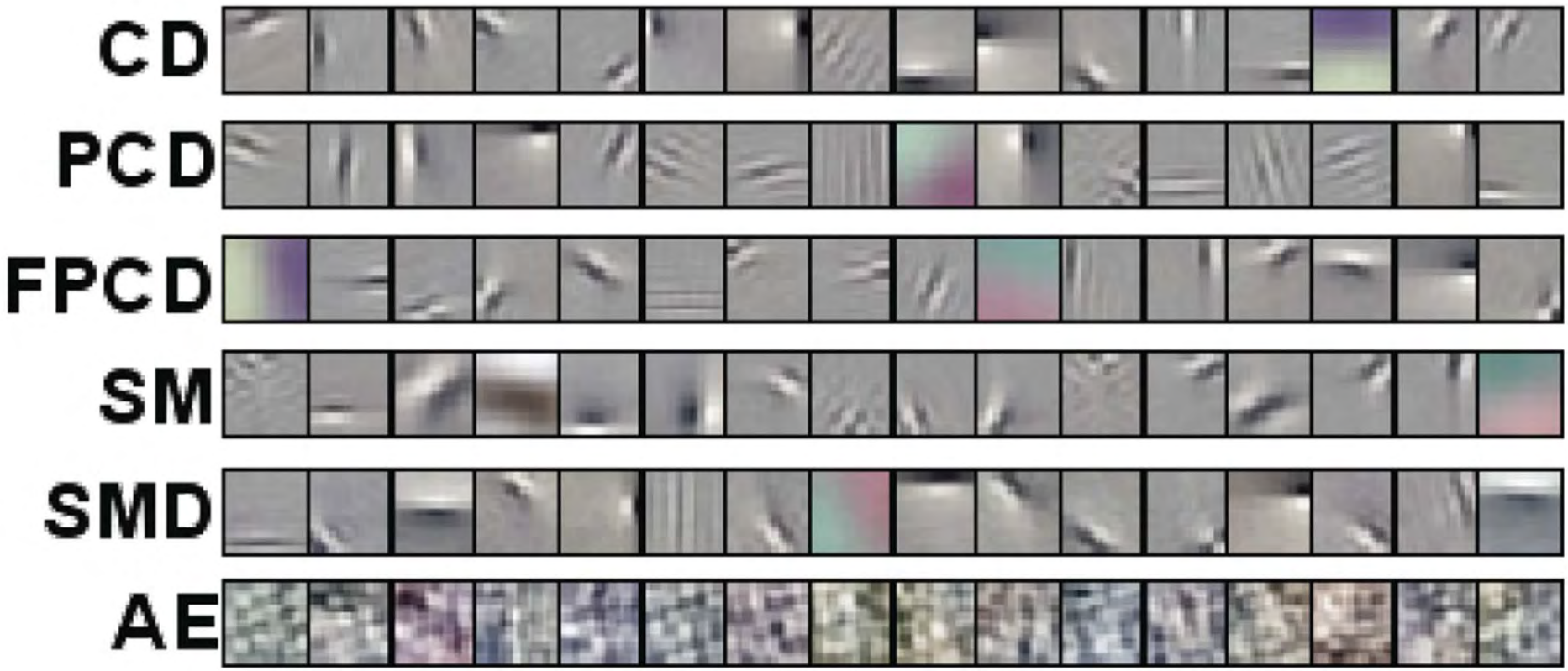


Squared error centers mass on data



Variance and Curvature shape the distribution

Learned covariance filters



K-means for feature learning

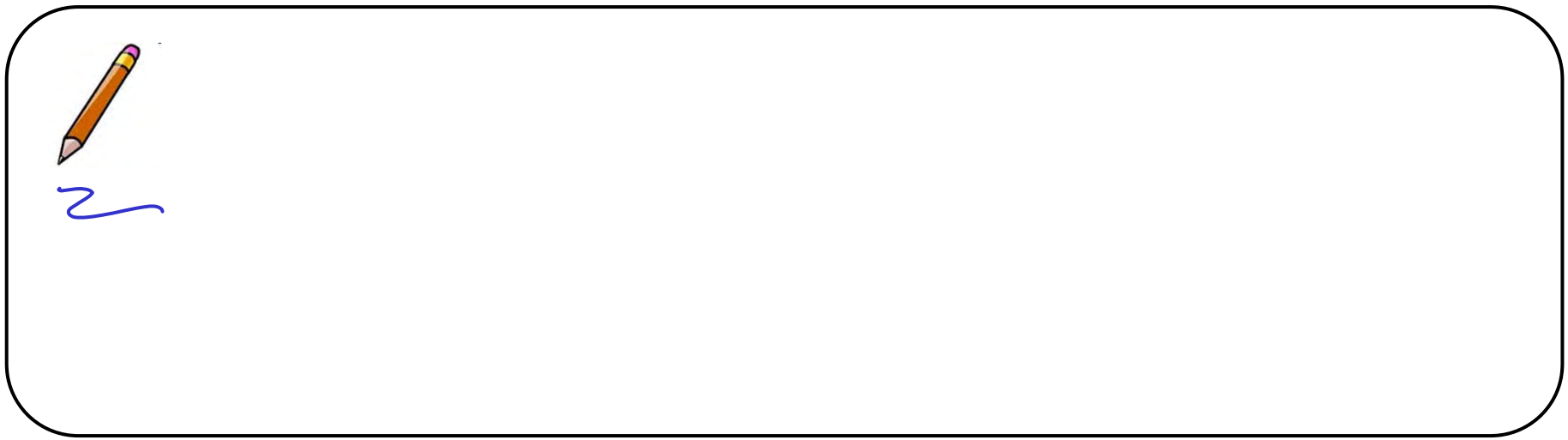
K-means clustering: We apply K-means clustering to learn K centroids $c^{(k)}$ from the input data. Given the learned centroids $c^{(k)}$, we consider two choices for the feature mapping f. The first is the standard 1-of-K, hard-assignment coding scheme:

$$f_k(x) = \begin{cases} 1 & \text{if } k = \arg\min_j ||c^{(j)} - x||_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The second is a non-linear mapping that attempts to be "softer" than the above encoding, but also yield sparse outputs through simple competition:

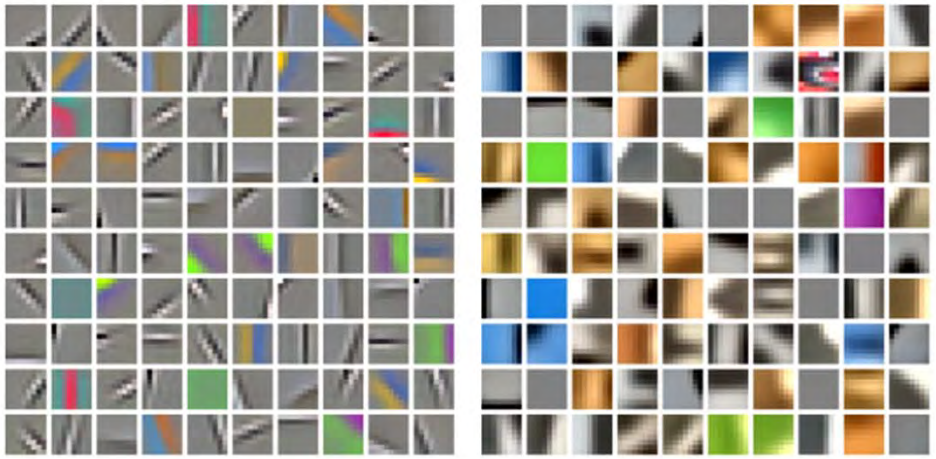
$$f_k(x) = \max\{0, \mu(z) - z_k\} \tag{3}$$

where $z_k = ||x - c^{(k)}||_2$ and $\mu(z)$ is the mean of the elements of z.

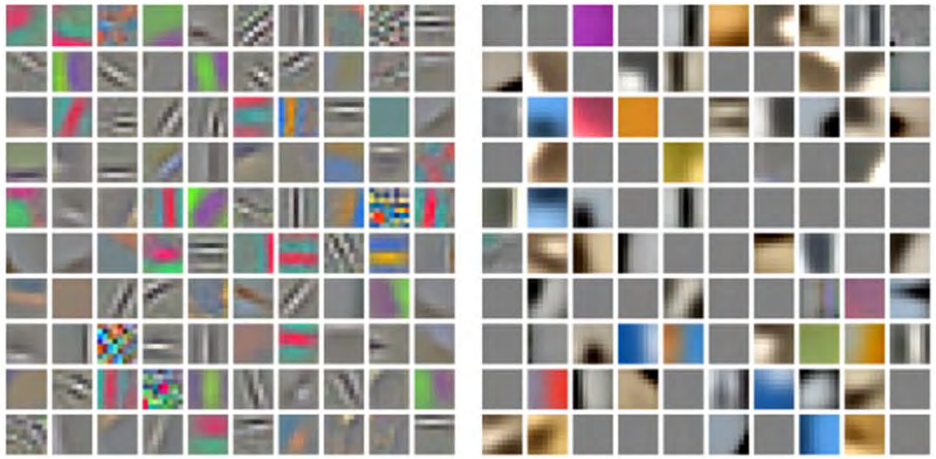


[Adam Coates, Honglak Lee & Andrew Ng 2009]

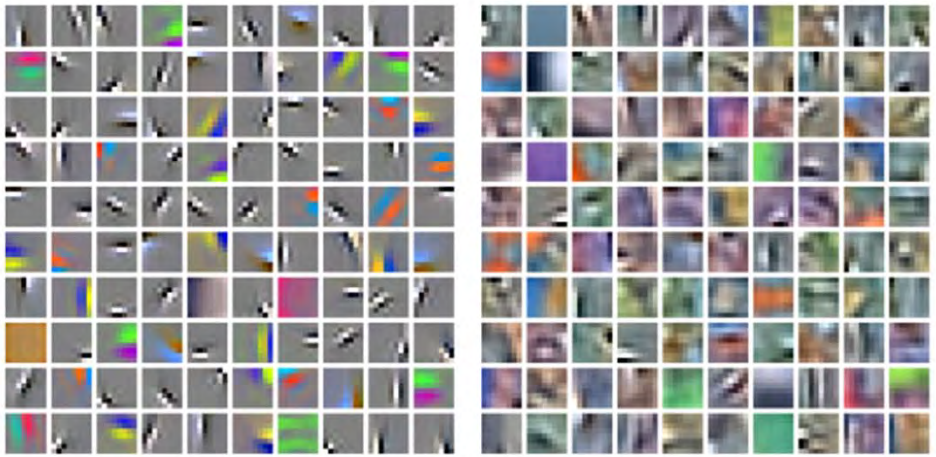
Learned bases (centroids)



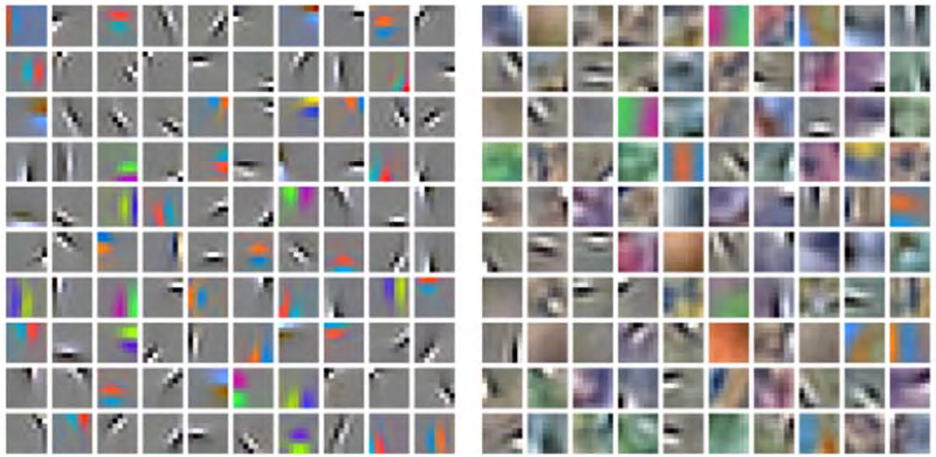
(a) K-means (with and without whitening)



(b) GMM (with and without whitening)



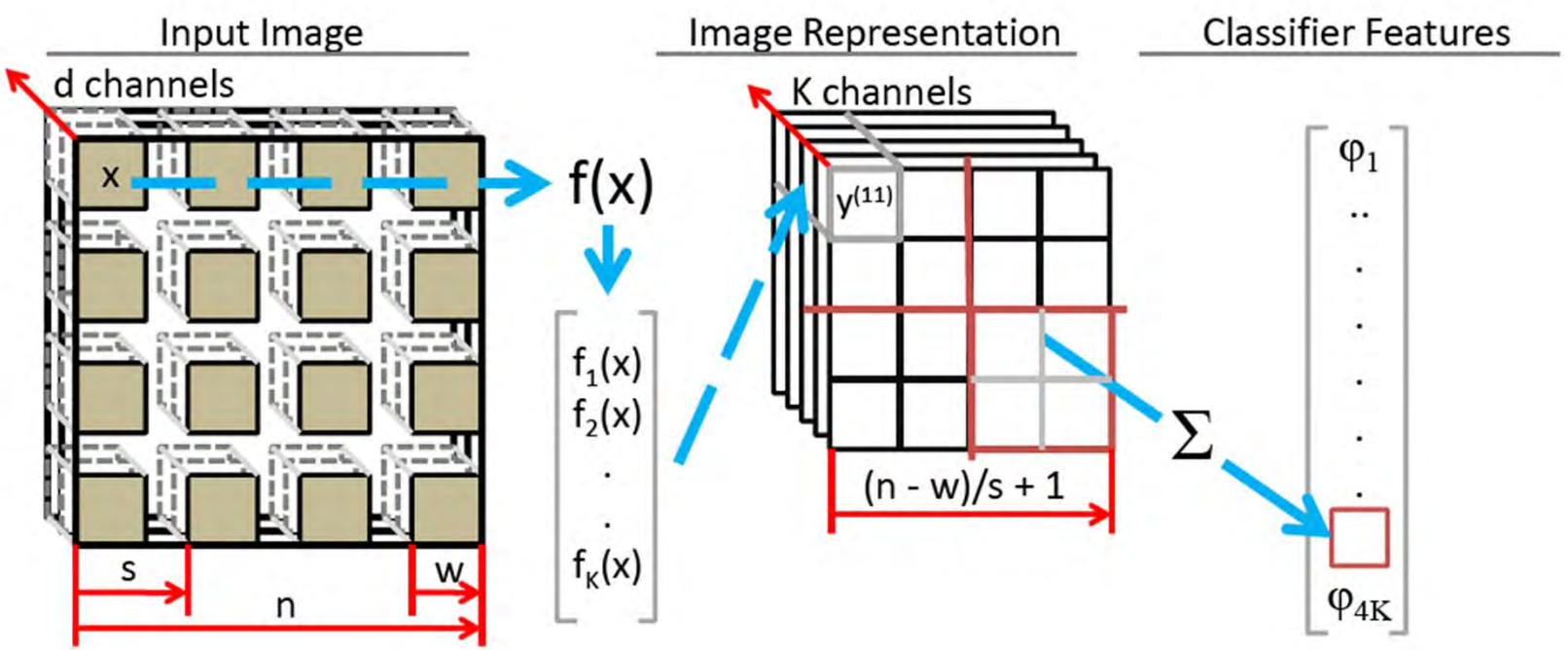
(c) Sparse Autoencoder (with and without whitening)



(d) Sparse RBM (with and without whitening)

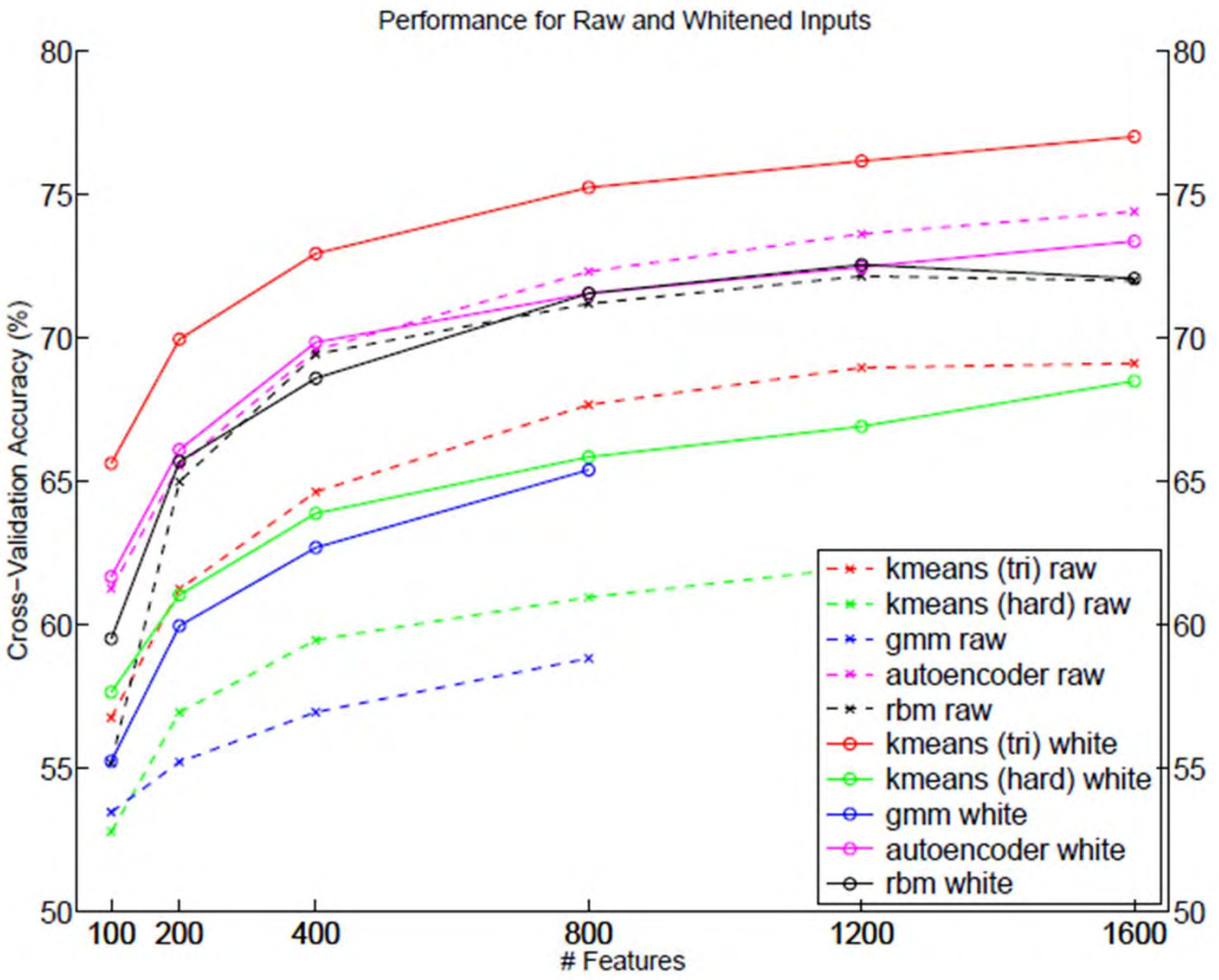
[Adam Coates, Honglak Lee & Andrew Ng 2009]

Mapping image to feature vector



[Adam Coates, Honglak Lee & Andrew Ng 2009]

K-means for feature learning



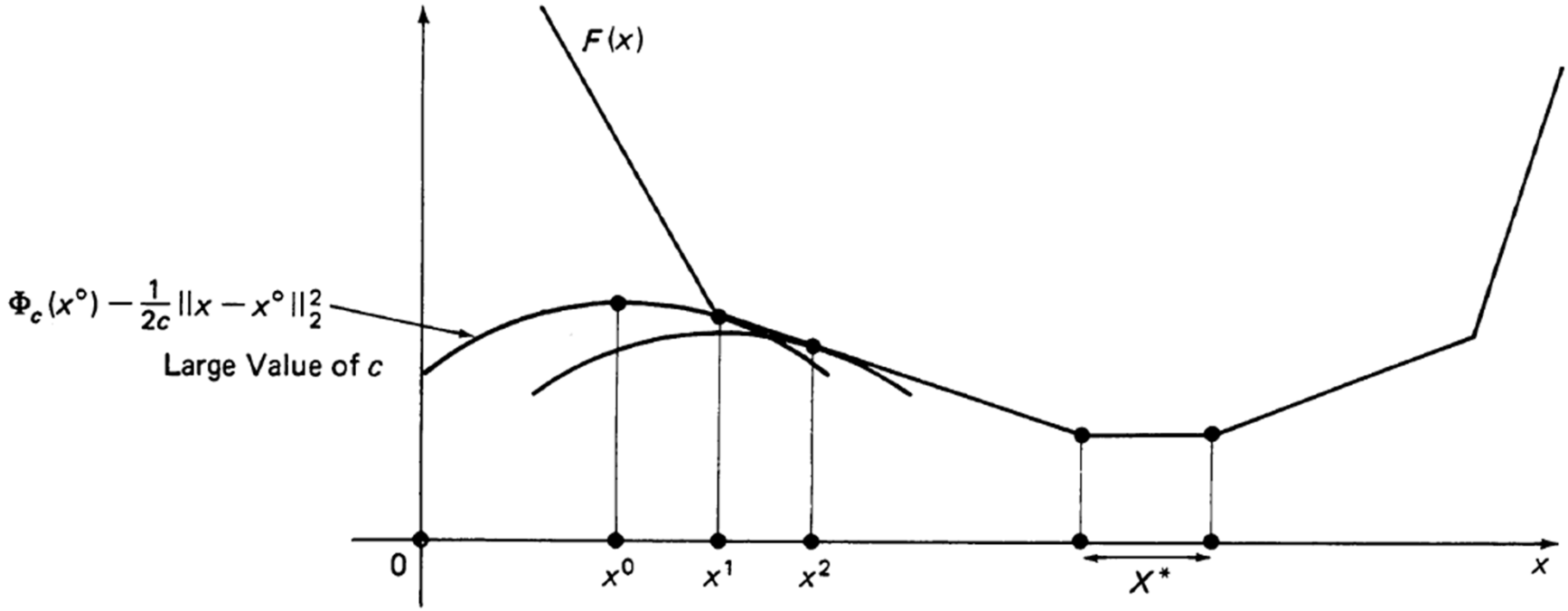
[Adam Coates, Honglak Lee & Andrew Ng 2009]

On threshold features and sparse coding

Proposition 1. *The soft threshold features*

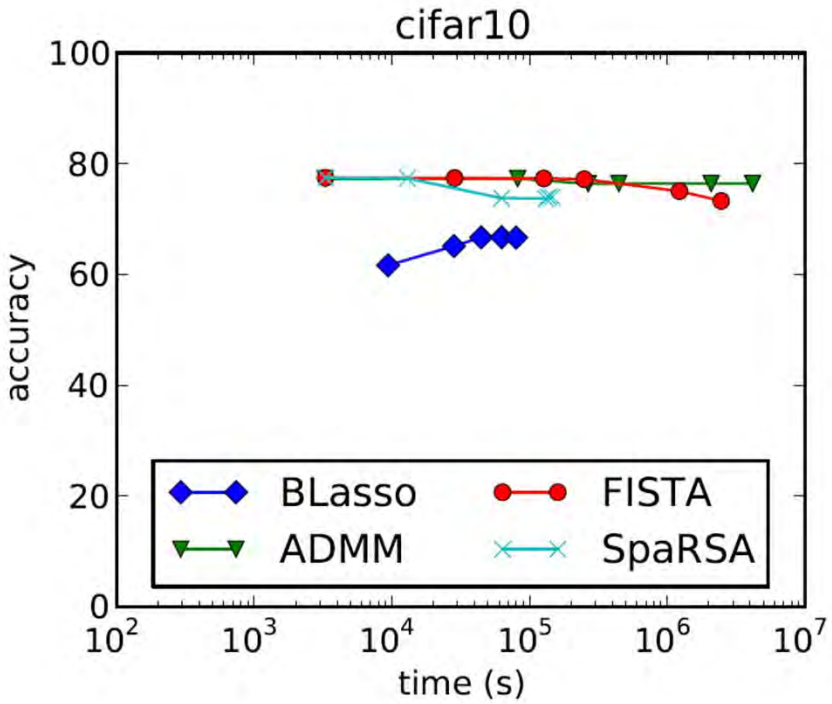
$$z_k(x) = \max\{0, w_k^{\mathrm{T}} x - \lambda\}$$

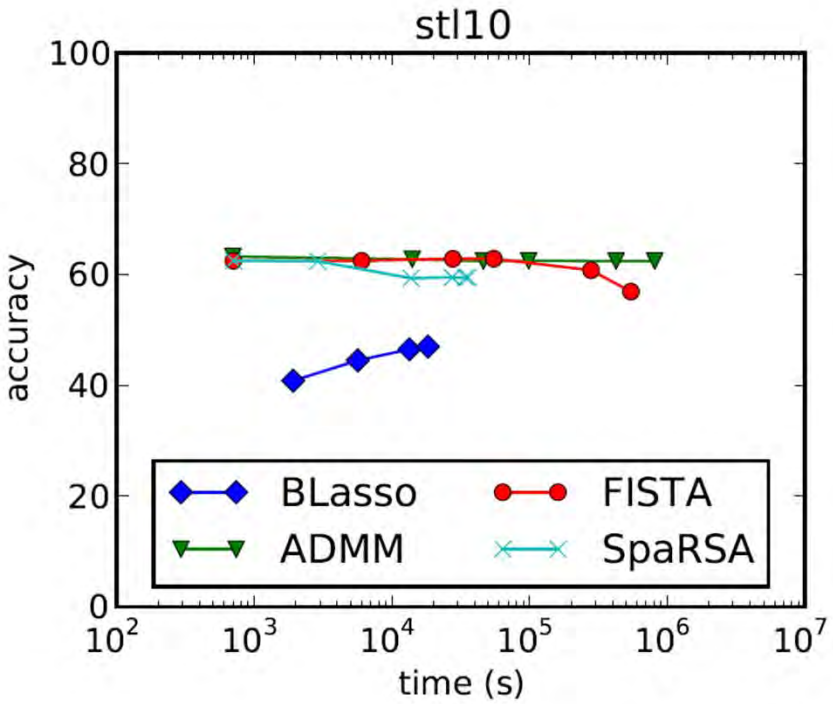
are given by a single step (of size 1) of proximal gradient descent on the non-negative sparse coding objective with regularization parameter λ and known dictionary W, starting from the fully sparse solution.



[Denil & dF 2011]

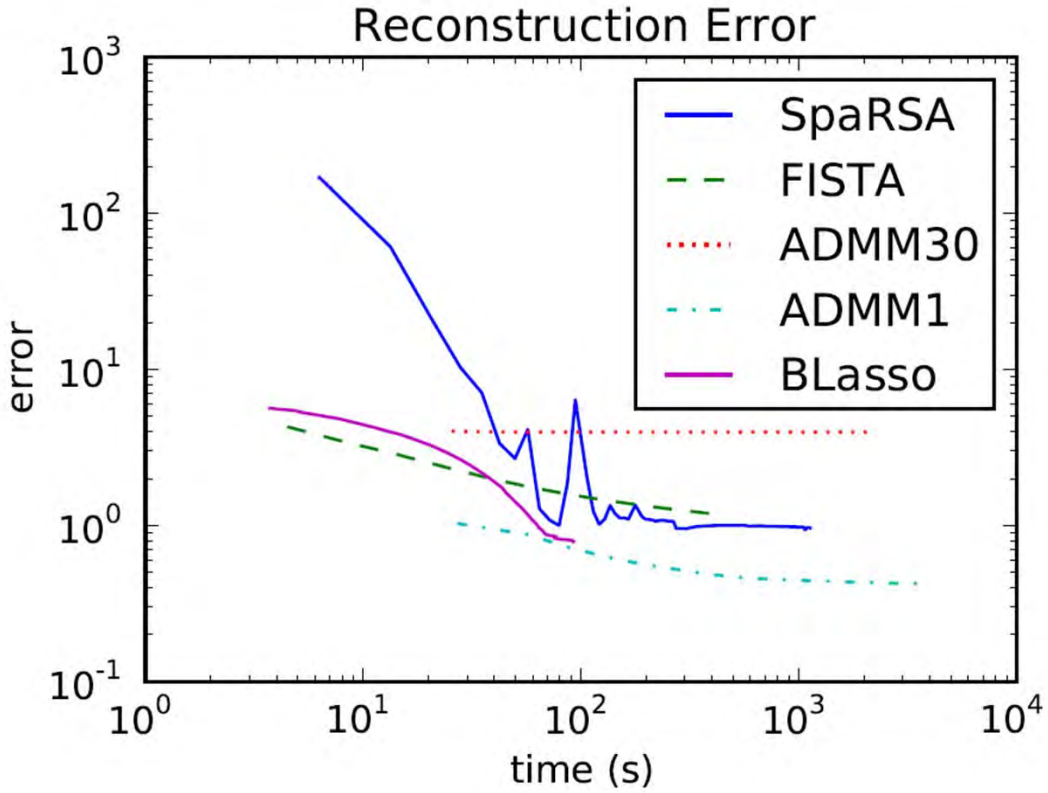
Fast scalable feature methods





[Denil & dF 2011]

Fast scalable feature methods

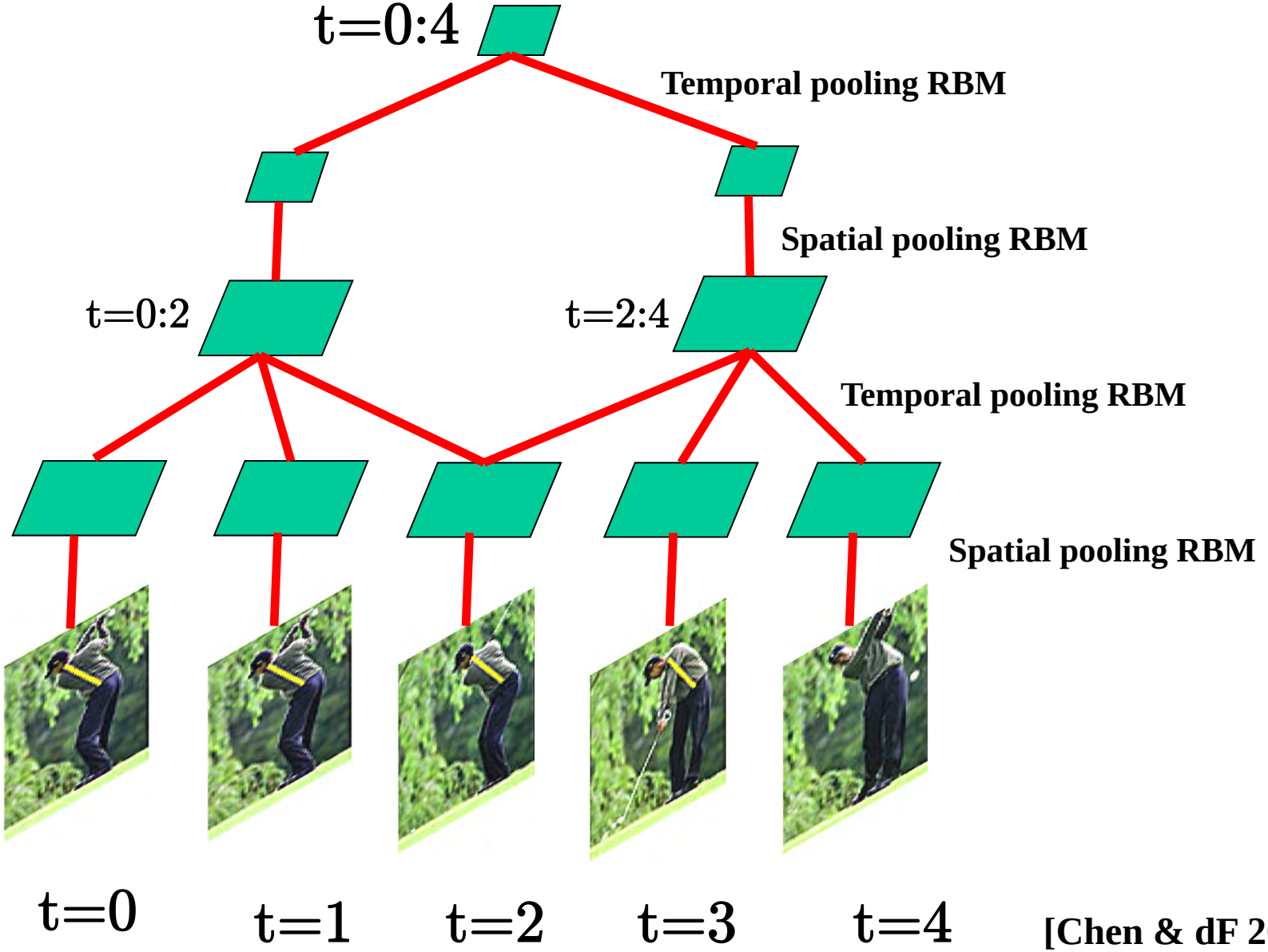


CIFAR10	Accuracy
BLasso	66.7
FISTA	**77.5**
ADMM	77.3
SpaRSA	**77.5**

STL10	Accuracy
BLasso	47.0
FISTA	62.8
ADMM	**63.2**
SpaRSA	62.5

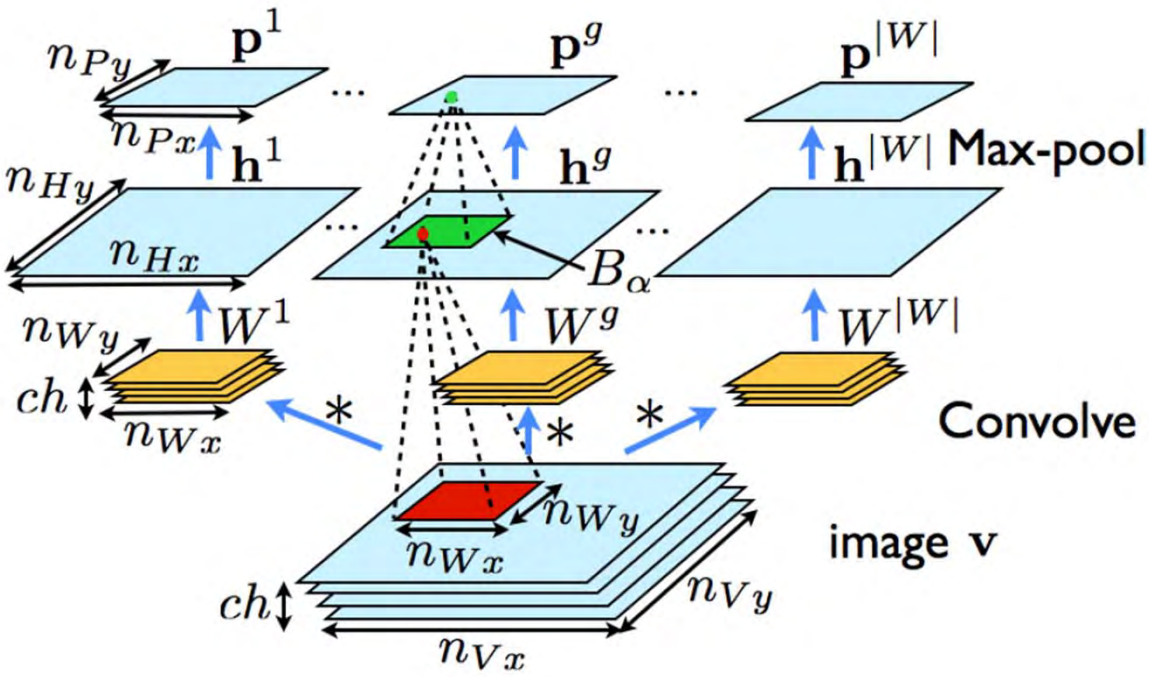
[Denil & dF 2011]

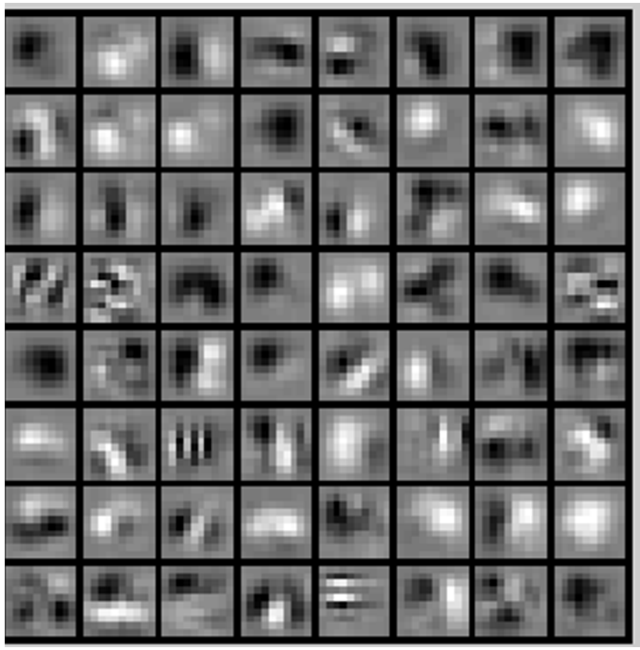
Spatio-temporal feature learning



[Chen & dF 2010]

Spatio-temporal feature learning





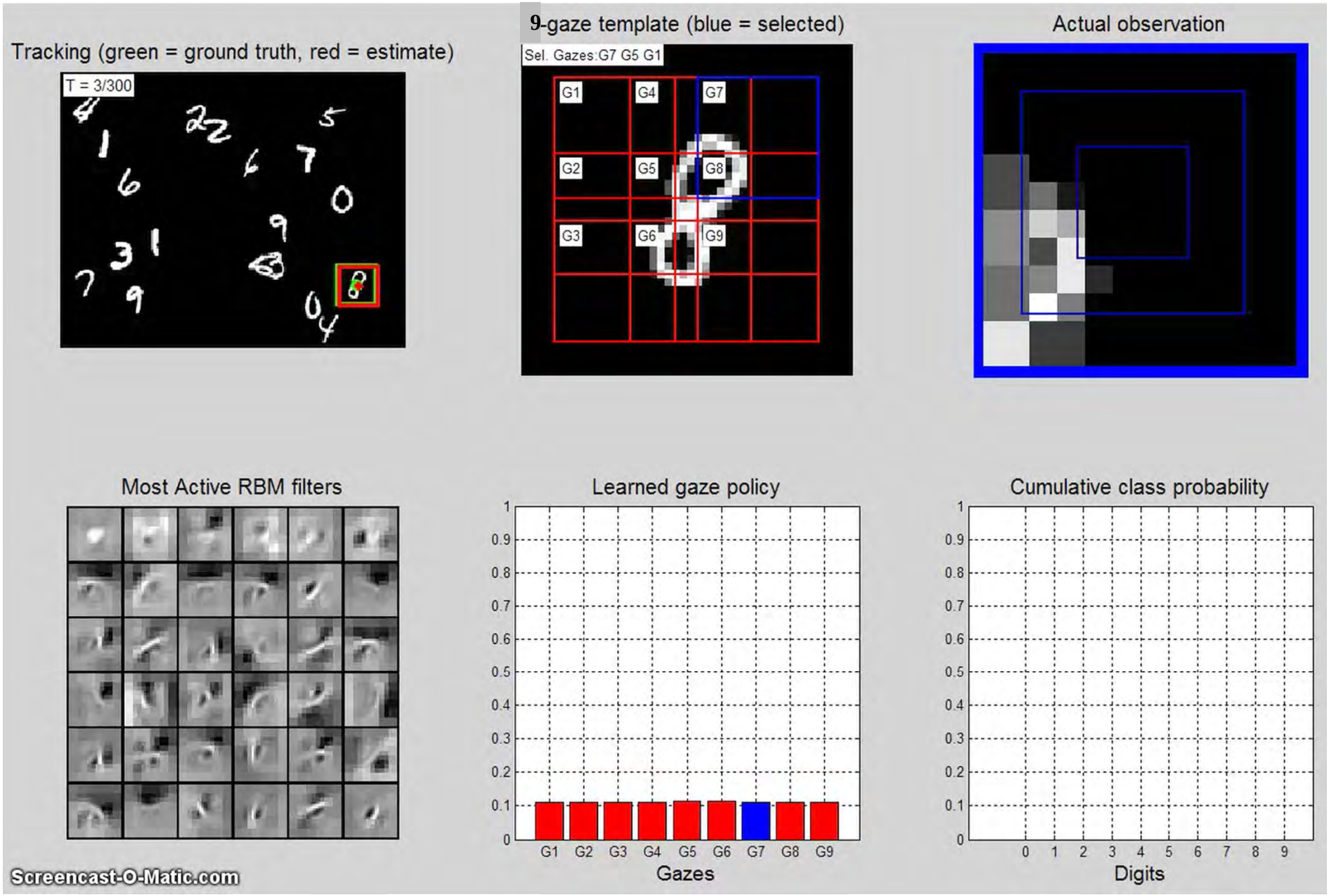
[Chen & dF 2010]

Imagining so as to plan where to look and search

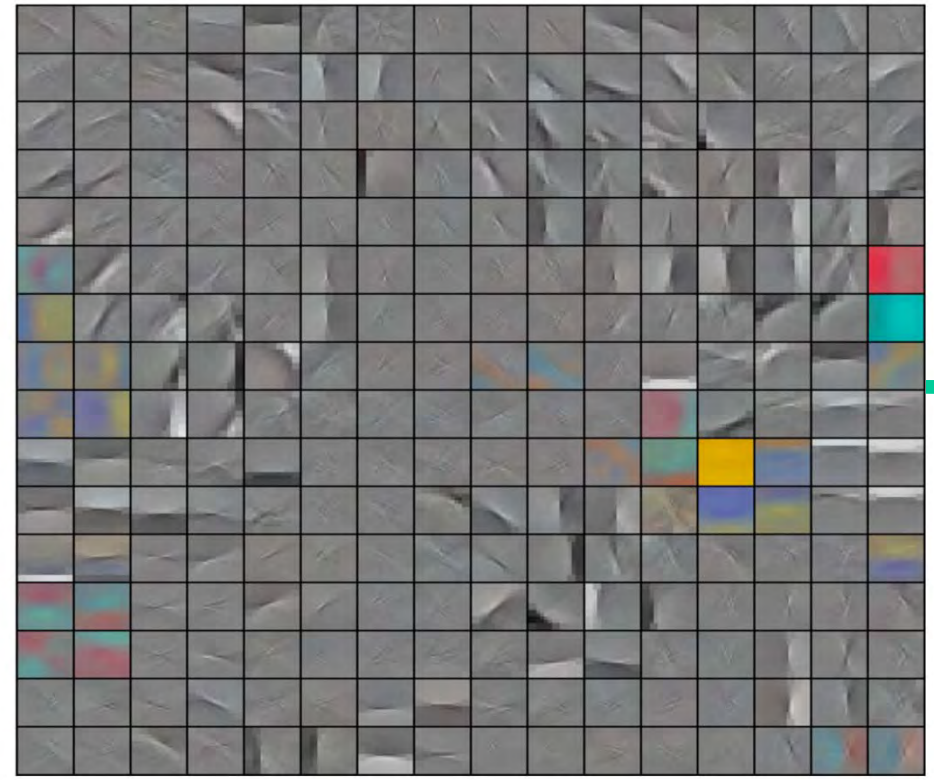
Observed gaze sequence

Model predictions

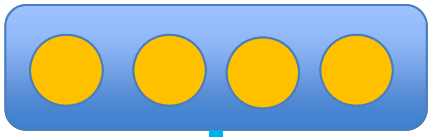
[Bo Chen & NdF 2010]



[Denil, Bazzani, Larochelle & dF, 2012]

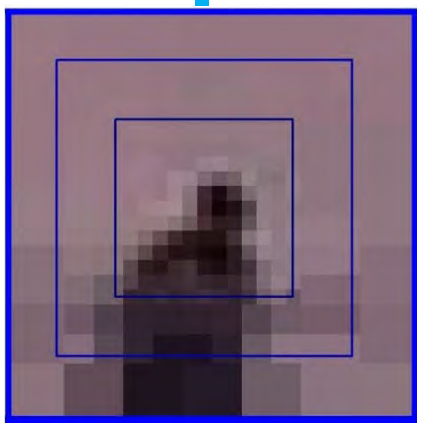


Hidden activations



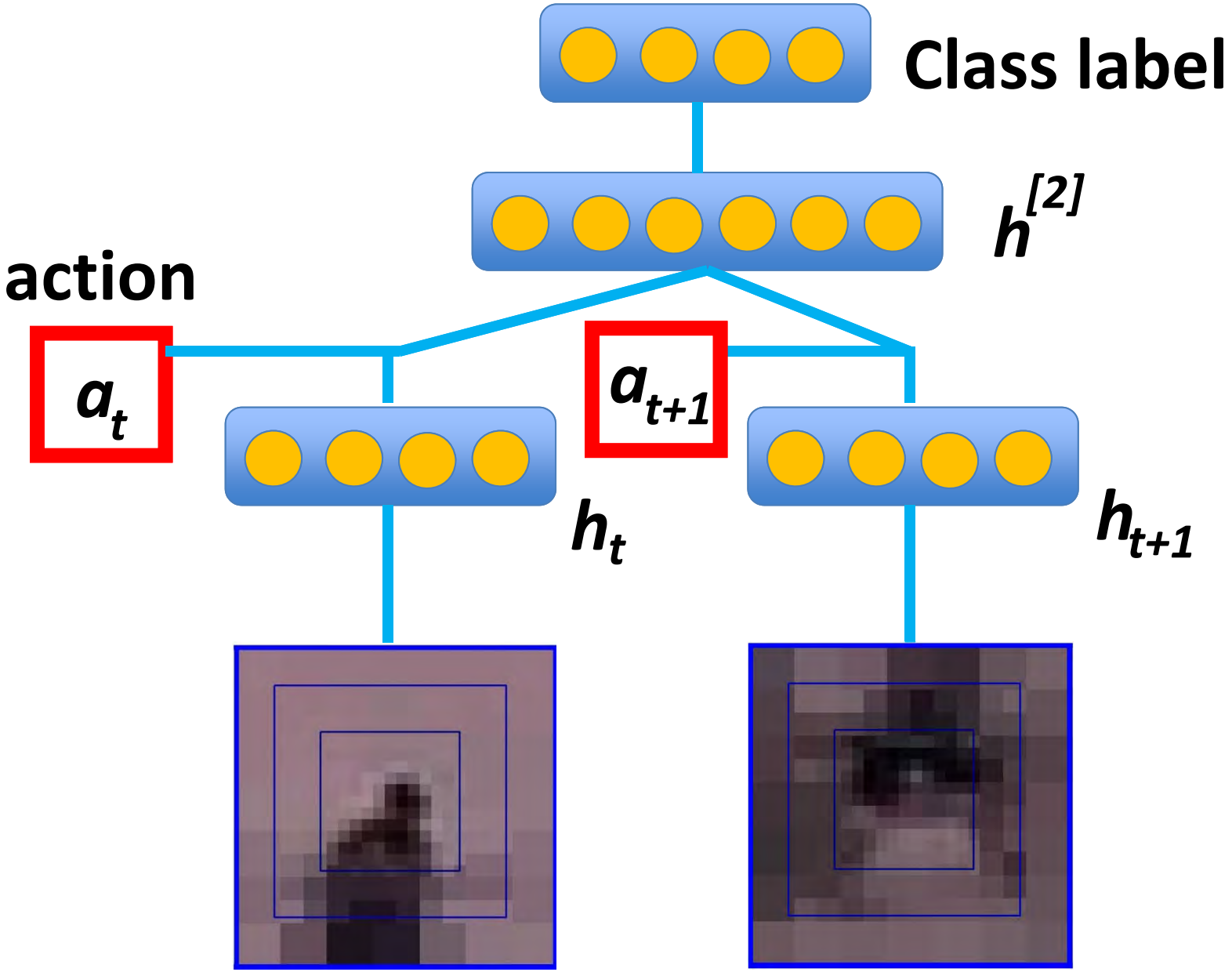
h

weights

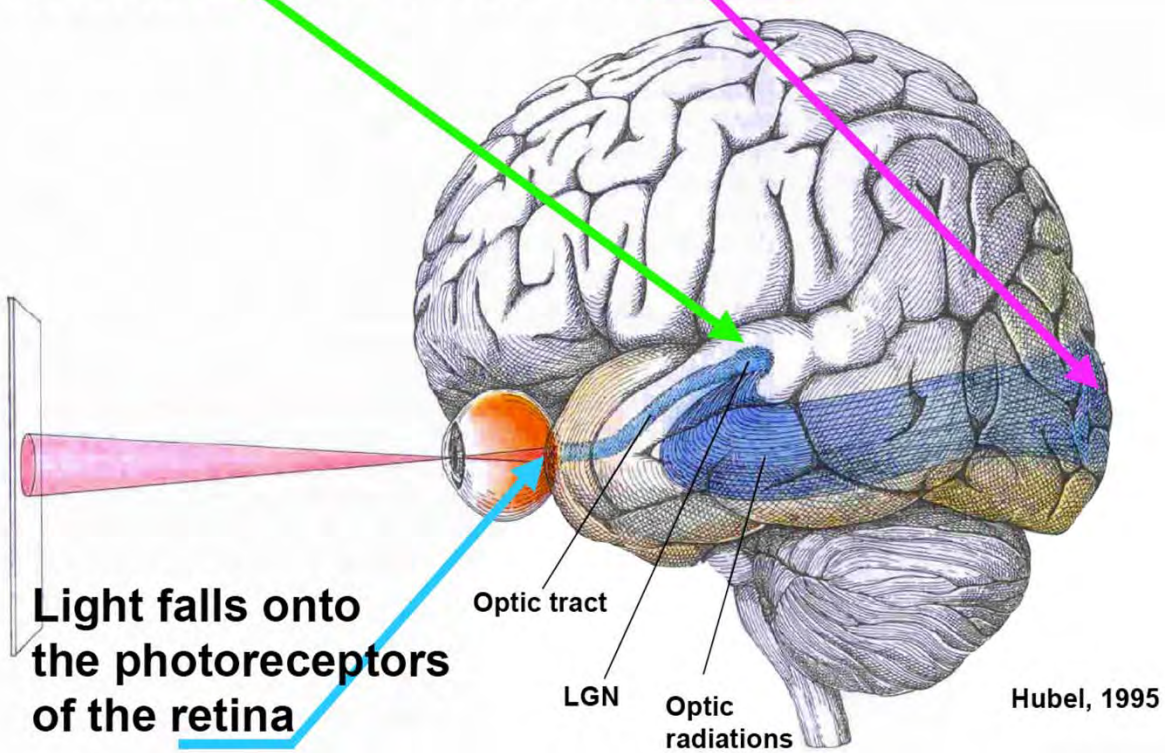


Foveated image

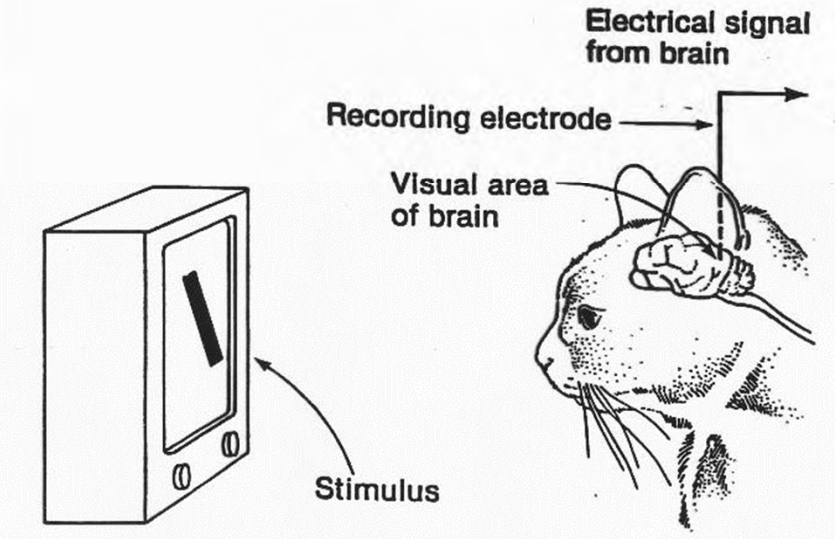
Multi-fixation RBM

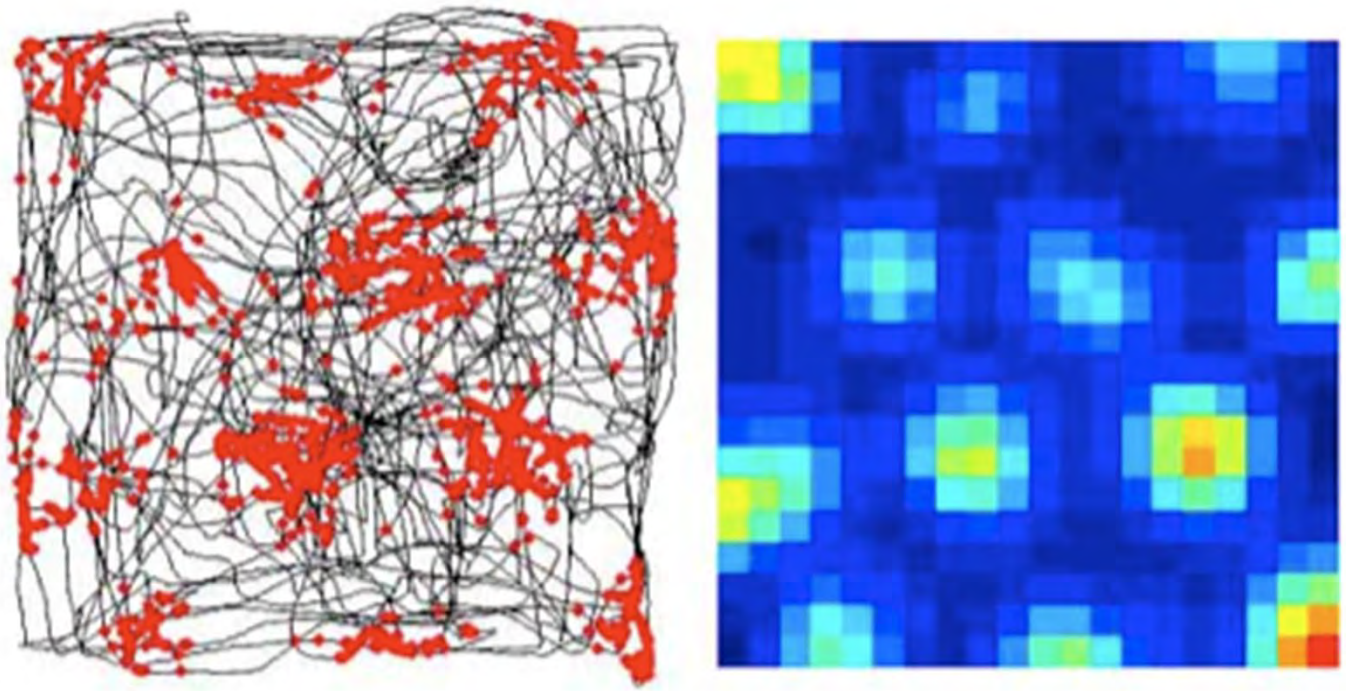


Thalamus (LGN) serves strategic role in gating of information flow to cortex



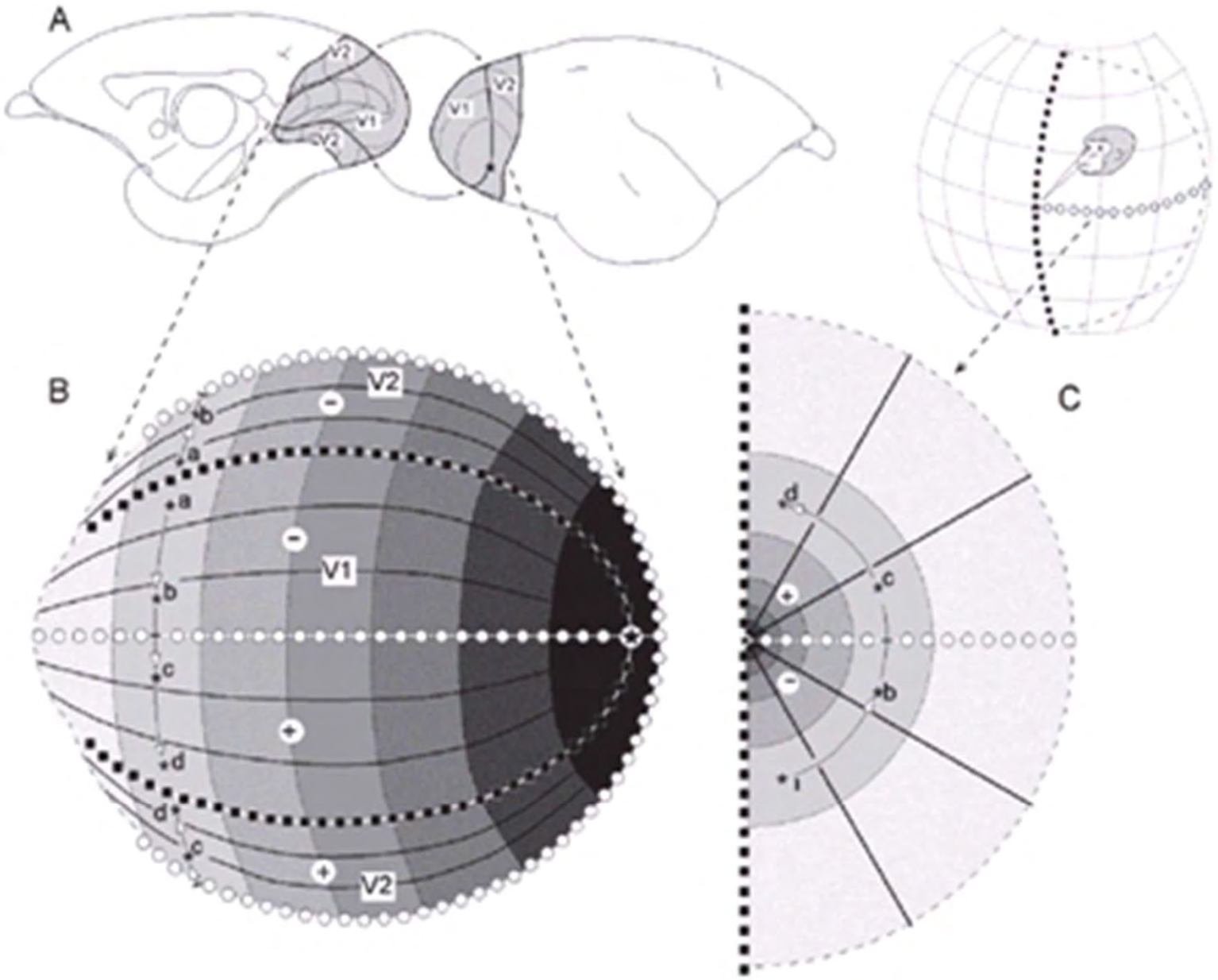
Hubel, 1995





GRID/PLACE CELLS: The black lines show how a rat explored a large box in a fairly haphazard manner. The red dots are the neuron response obtained with an electrode inserted in the rat's subcortex .

[Hafting et al 2005]



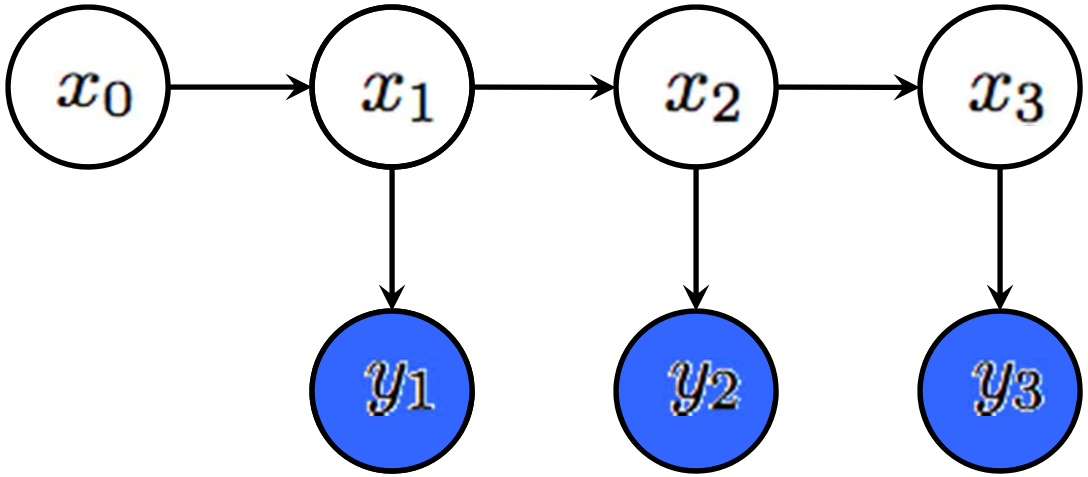
[Ro ca, 2002]

Multi-target tracking

$$\begin{pmatrix} s_{t,1} \\ s_{t,2} \\ v_{t,1} \\ v_{t,2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} s_{t-1,1} \\ s_{t-1,2} \\ v_{t-1,1} \\ v_{t-1,2} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \epsilon_{t,1} \\ \epsilon_{t,2} \end{pmatrix}$$

$$\begin{pmatrix} y_{t,1} \\ y_{t,2} \end{pmatrix} = \begin{pmatrix} s_{t,1} \\ s_{t,2} \end{pmatrix} + \begin{pmatrix} e_{t,1} \\ e_{t,2} \end{pmatrix}$$

Bayes net depiction



$$p(x_3|y_{1:3})$$

$$p(x_t|y_{1:t}) \propto p(y_t|x_t) \int p(x_t|x_{t-1})p(x_t|y_{1:t-1})dx_{t-1}$$

Nasty integral

Importance Sampling for Optimal Filtering / Tracking

- Input: $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$
- Prediction:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1}$$

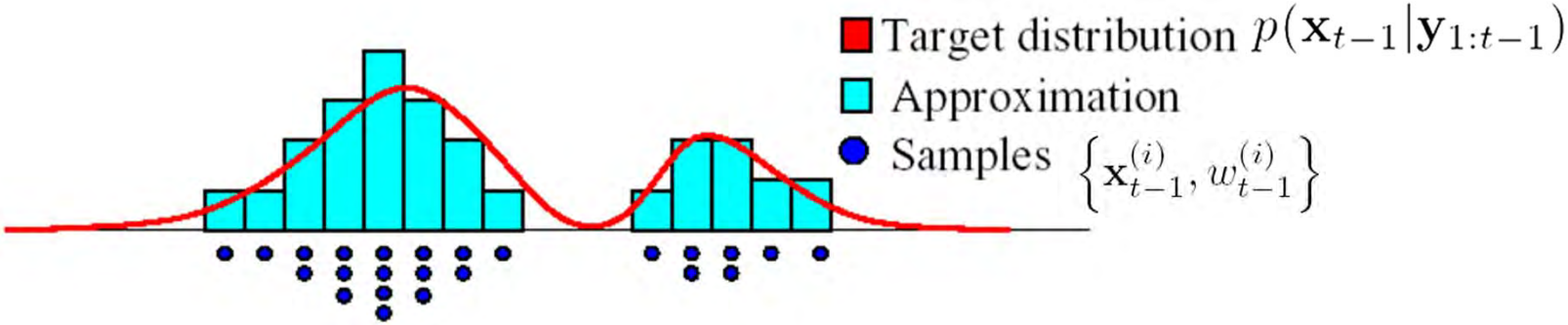
- Bayes update:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{\int p(\mathbf{y}_t|\mathbf{x}_t)\, p(\mathbf{x}_t|\mathbf{y}_{1:t-1})\, d\mathbf{x}_t}$$

- Output: $p(\mathbf{x}_t|\mathbf{y}_{1:t})$

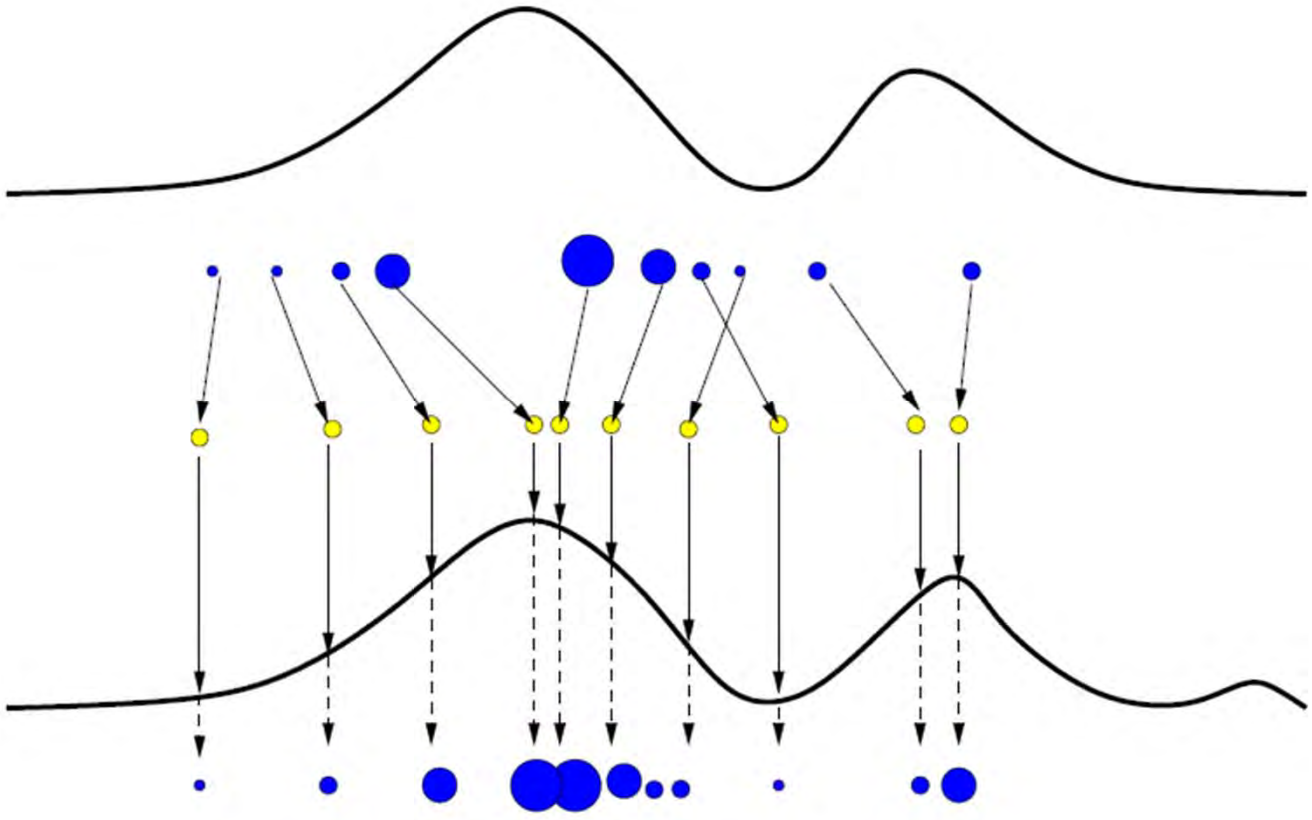
One can Compute the Integrals Recursively in Time

Given the samples $\left\{\mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}$ from $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$

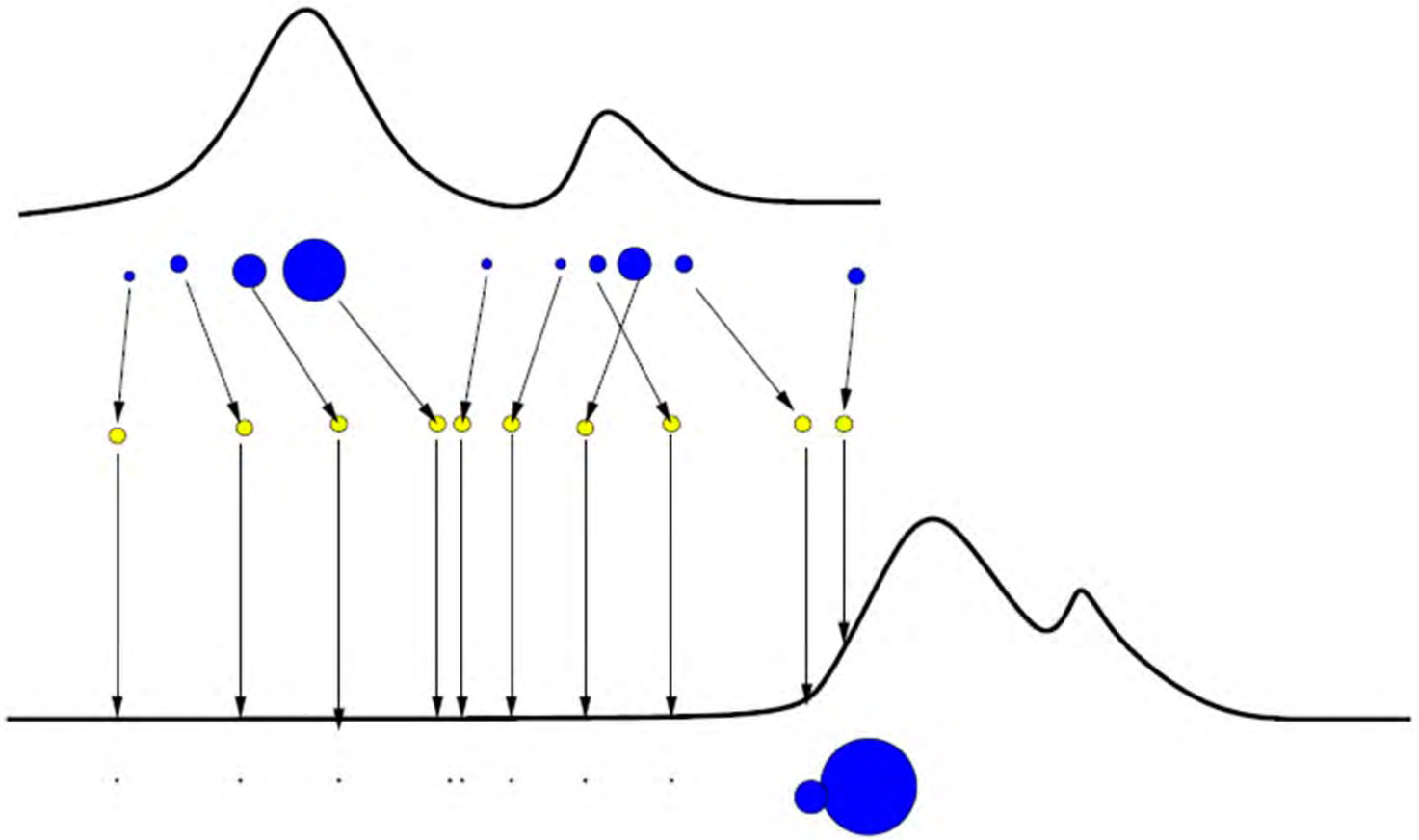


$$\int p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\, d\mathbf{x}_{t-1} \rightarrow \sum_{j=1}^{N} w_{t-1}^{(j)} p\left(\mathbf{x}_t \middle| \mathbf{x}_{t-1}^{(j)}\right)$$

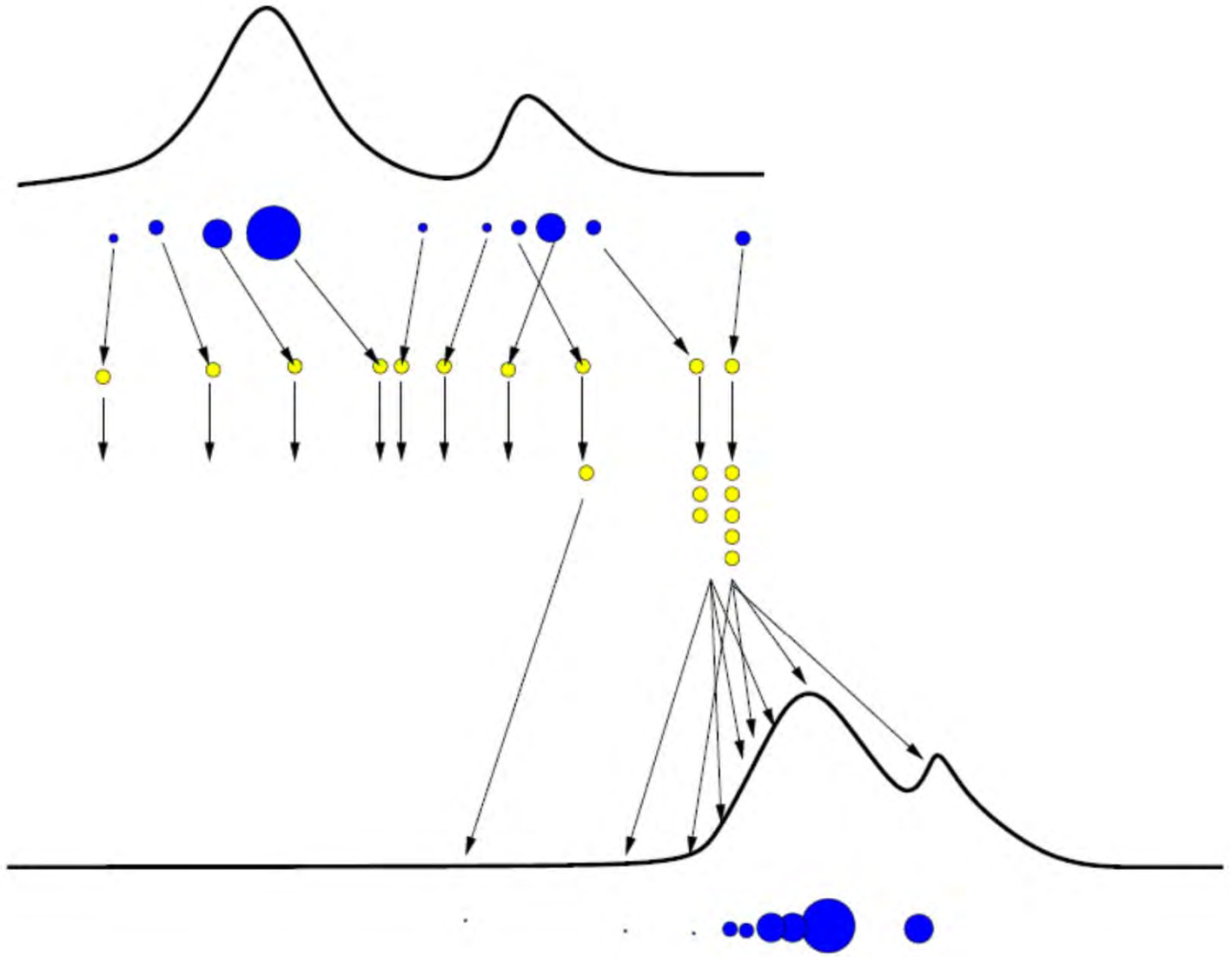
Particle Filtering (SIS)



Particle Filtering (SIS)



Particle Filtering (SIR)



Particle Filtering Code

- For $i = 1, \ldots, N$, sample $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$ and set $t = 1$.

- For $i = 1, \ldots, N$, sample $\widetilde{\mathbf{x}}_t^{(i)} \sim p\left(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)}\right)$.
- For $i = 1, \ldots, N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = p\left(\mathbf{y}_t | \widetilde{\mathbf{x}}_t^{(i)}\right)$$

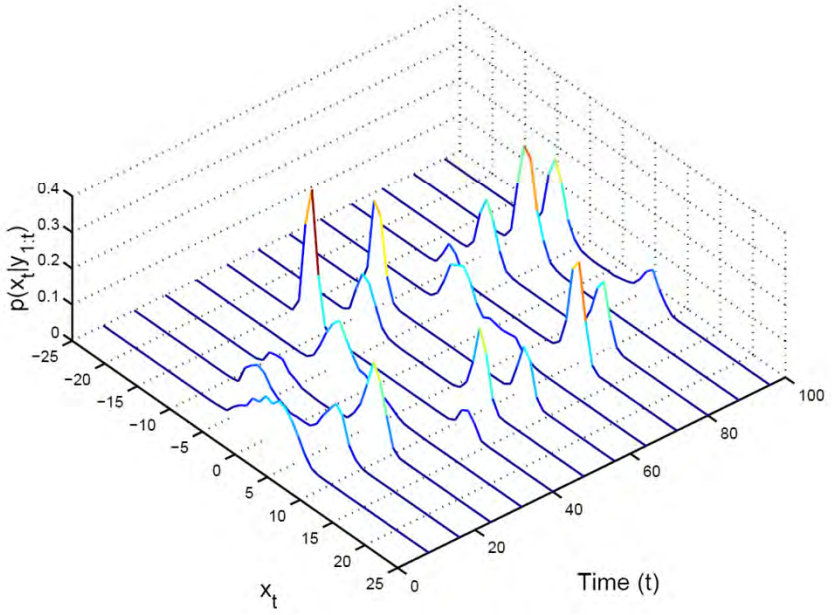
- Normalise the importance weights.
- Select • ttest samples (black-box).

Simple Example

$$x_t = \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t$$

$$y_t = \frac{x_t^2}{20} + w_t$$

where $x_0 \sim \mathcal{N}(0, \sigma_1^2)$, v_t and w_t are mutually independent white Gaussian noises, $v_t \sim \mathcal{N}(0, \sigma_v^2)$ and $w_t \sim \mathcal{N}(0, \sigma_w^2)$



- For $i = 1, ..., N$, sample $\mathbf{x}_0^{(i)} \sim \mathcal{N}\left(0, \sigma_1^2\right)$
- For $i = 1, ..., N$, sample

$$x_t^{(i)} = \frac{1}{2} x_{t-1}^{(i)} + 25 \frac{x_{t-1}^{(i)}}{1 + x_{t-1}^{2(i)}} + 8 \cos\left(1.2t\right) + \mathcal{N}\left(0, \sigma_v^2\right)$$

- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{1}{2\sigma_w^2}\left(y - \frac{x_t^{2(i)}}{20}\right)^2}$$

- Normalise the importance weights.
- Resample fittest samples (black-box).

Particle Methods More Generally

The goal is to approximate a target distribution over a sequence of states $\mathbf{x}_{1:n} \triangleq \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ that is growing with "time" as well as the partition function.

$$\pi_n(\mathbf{x}_{1:n}) = Z_n^{-1} f_n(\mathbf{x}_{1:n}) \qquad Z_n \triangleq \int f_n(\mathbf{x}_{1:n}) d\mathbf{x}_{1:n}$$

We do this using sequential importance sampling (M&U, 49)

$$w_n = \frac{f_n(\mathbf{x}_{1:n})}{q_n(\mathbf{x}_{1:n})} = \frac{f_n(\mathbf{x}_{1:n})}{f_{n-1}(\mathbf{x}_{1:n-1})} \frac{1}{q(\mathbf{x}_n|\mathbf{x}_{1:n-1})} w_{n-1}$$

e.g. For filtering, we use: $f_n(\mathbf{x}_{1:n}) = \prod_{t=1}^{n} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{y}_t|\mathbf{x}_t)$

Sequential importance sampling step

- For $i = 1, ..., N$, sample from the proposal

$$\mathbf{x}_t^{(i)} \sim q\left(\mathbf{x}_t \middle| \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}\right)$$

- For $i = 1, ..., N$, evaluate the importance weights

$$\widetilde{w}_t^{(i)} = \frac{p\left(\mathbf{y}_t \middle| \mathbf{x}_t^{(i)}\right) p\left(\mathbf{x}_t^{(i)} \middle| \mathbf{x}_{t-1}^{(i)}\right)}{q\left(\mathbf{x}_t^{(i)} \middle| \mathbf{y}_t, \mathbf{x}_{t-1}^{(i)}\right)} \widetilde{w}_{t-1}^{(i)}$$

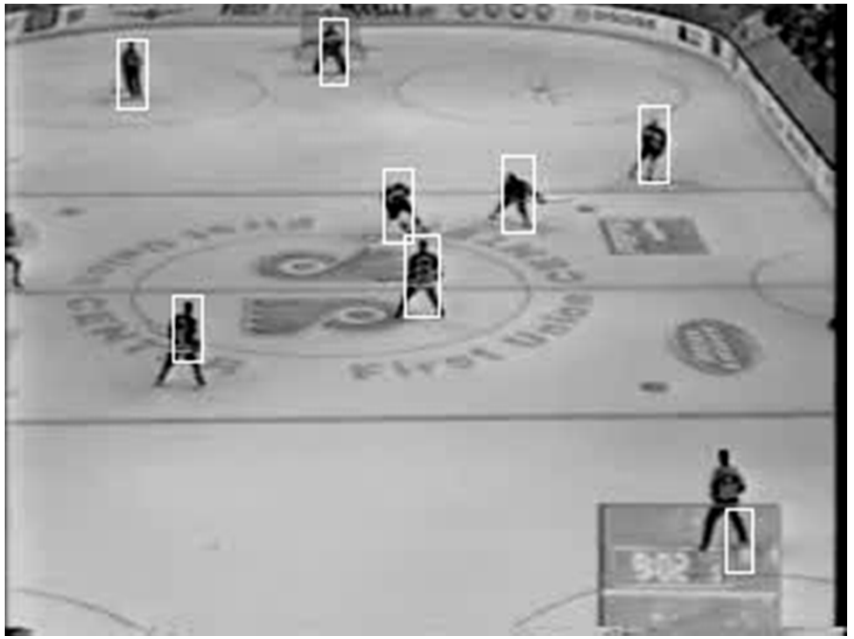
- Normalise the importance weights

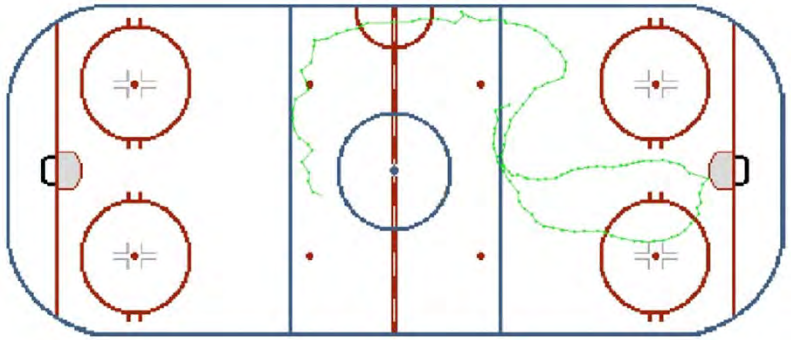
$$w_t^{(i)} = \frac{\widetilde{w}_t^{(i)}}{\sum_j^N \widetilde{w}_t^{(j)}}$$

Selection step

- Resample the discrete weighted measure $\left\{\mathbf{x}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^N$ to obtain an unweighted measure $\left\{\mathbf{x}_t^{(i)}, \frac{1}{N}\right\}_{i=1}^N$ of N new particles.

Using Clever Proposals: e.g. Boosting, RFs

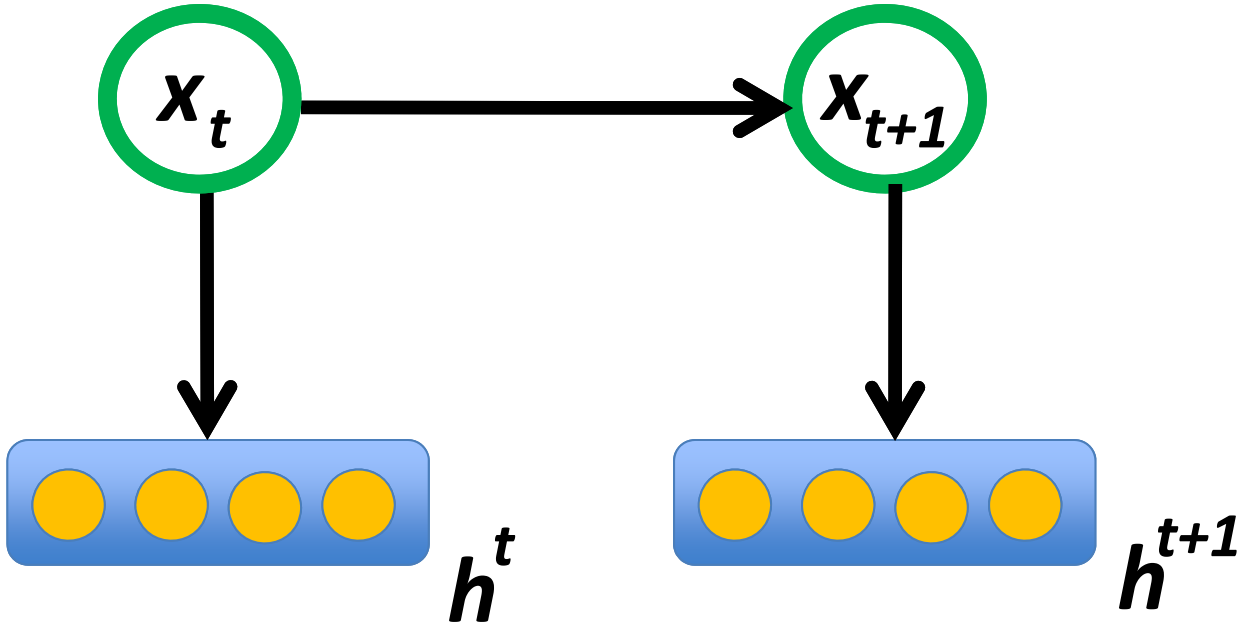




[Okuma, Taleghani, dF, Little & Lowe 2004]

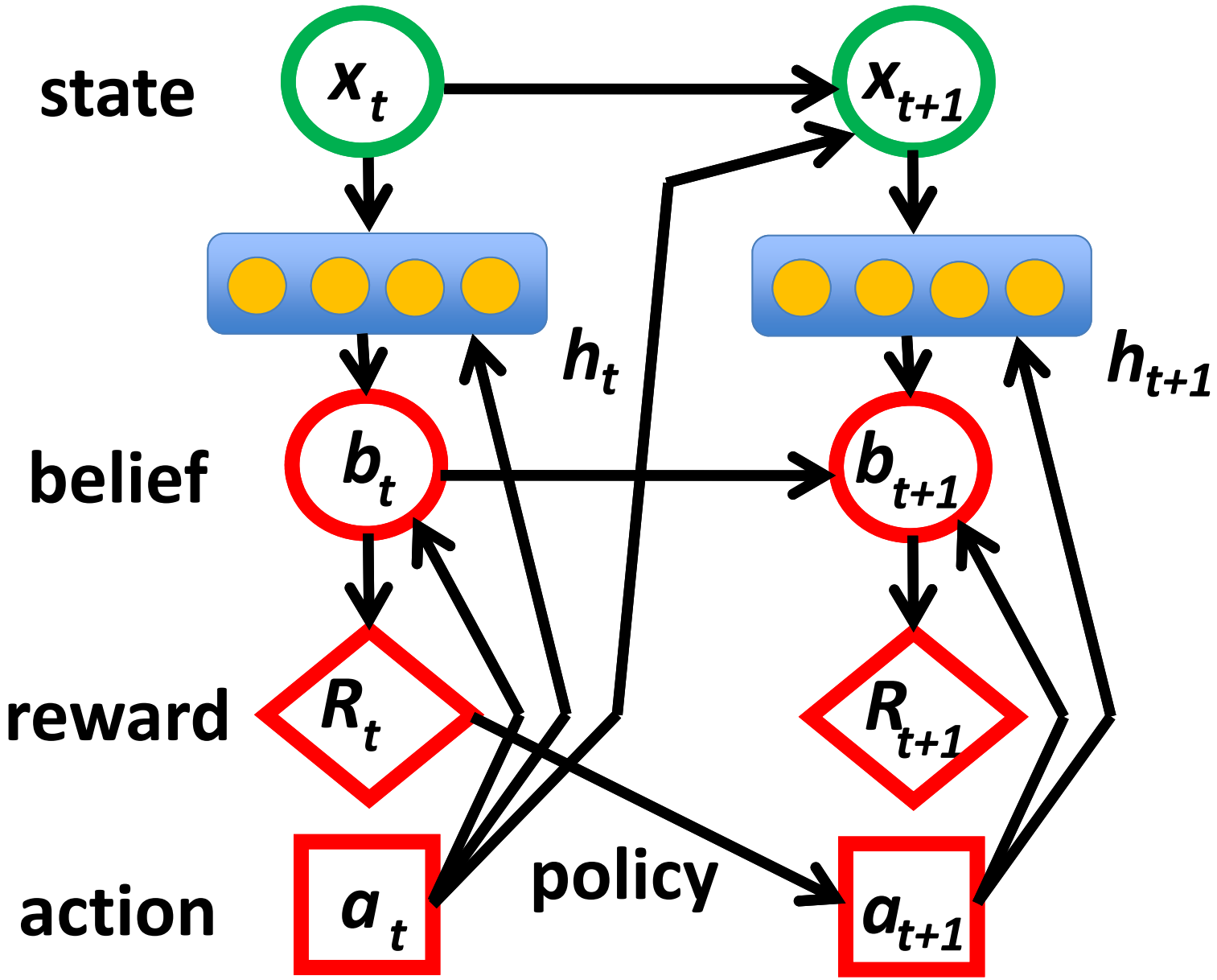
Use deep architectures as the observation models

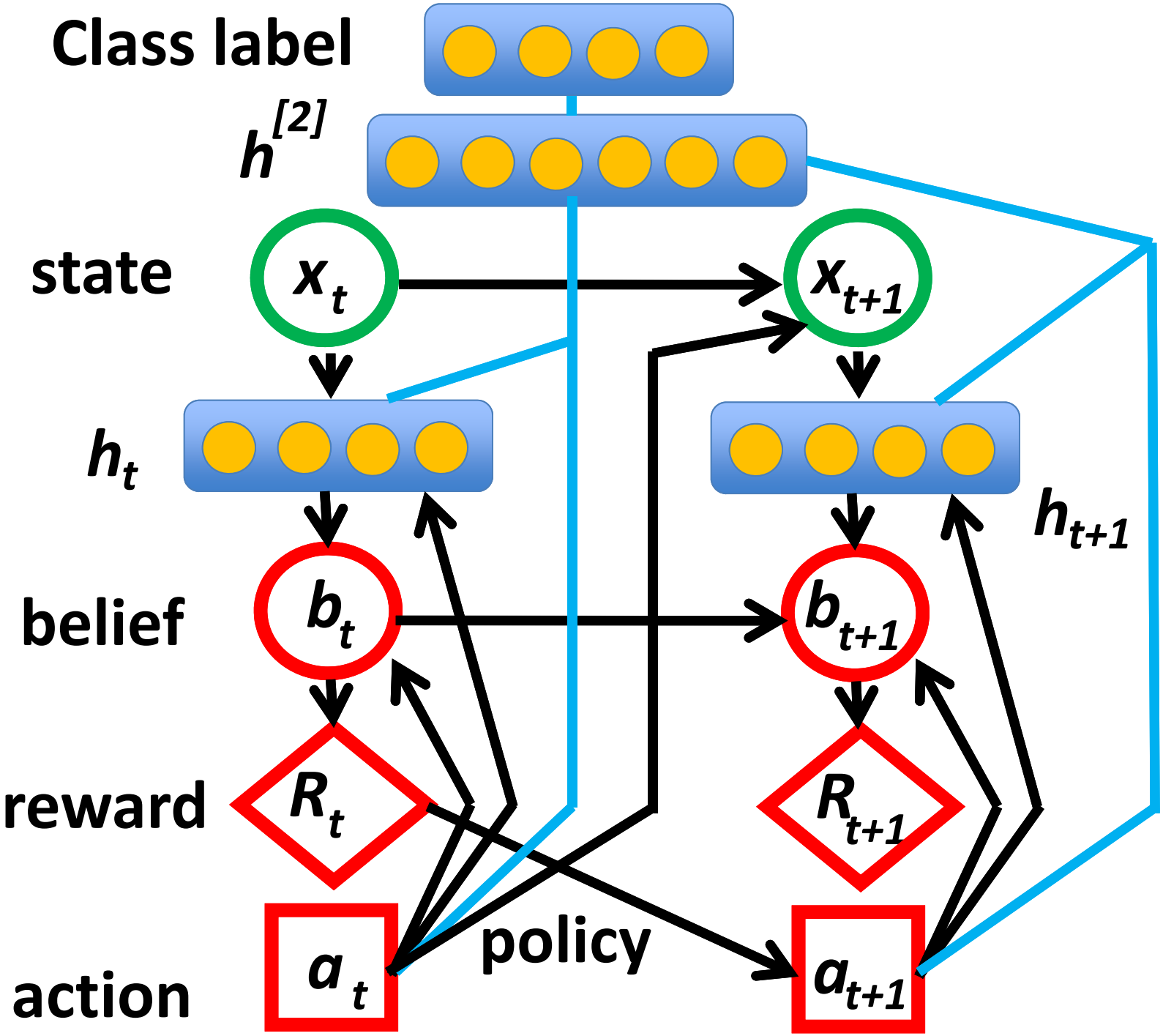
x encodes location, scale, speed, rotation, discrete states, …



Goal: Estimate belief state $b_{t+1} = p(x_{t+1} | h_{1:t+1})$

Sequential decision making and optimization




Class label
$h^{[2]}$
state
x_t
x_{t+1}
h_t
h_{t+1}
belief
b_t
b_{t+1}
reward
R_t
R_{t+1}
action
a_t
policy
a_{t+1}

Pseudo-code

1. *Initialization*, $t = 0$.
   - For $i = 1, \ldots, N$, $\mathbf{x}_0^{(i)} \sim p(\mathbf{x}_0)$, set $t = 1$, and initialize the template.
   - Set $R_0(\mathbf{a}_0 = k) = 0$ for $k = 1, \ldots, K$, where K is the number of gazes in the template

2. *Importance sampling step*
   - For $i = 1, \ldots, N$, $\tilde{\mathbf{x}}_t^{(i)} \sim q_t\left(d\mathbf{x}_t^{(i)} \middle| \tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{h}_{1:t}, \mathbf{a}_{1:t}\right)$ and set $\tilde{\mathbf{x}}_{0:t}^{(i)} = \left(\mathbf{x}_{0:t-1}^{(i)}, \tilde{\mathbf{x}}_t^{(i)}\right)$.
   - For $i = 1, \ldots, N$, $k = 1, \ldots, K$, , evaluate the importance weights

$$\tilde{w}_t^{(i),k} \propto \frac{p\left(\mathbf{h}_t | \tilde{\mathbf{x}}_t^{(i)}, \mathbf{a}_t = k\right) p\left(\tilde{\mathbf{x}}_t^{(i)} | \tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{a}_{t-1}\right)}{q_t\left(\tilde{\mathbf{x}}_t^{(i)} \middle| \tilde{\mathbf{x}}_{0:t-1}^{(i)}, \mathbf{h}_{1:t}, \mathbf{a}_{1:t}\right)}.$$

   - Normalise the importance weights $w_t^{(i),k} = \frac{\tilde{w}_t^{(i),k}}{\sum_{j=1}^N \tilde{w}_t^{(j),k}}$

3. *Gaze control step*
   - Update the policy

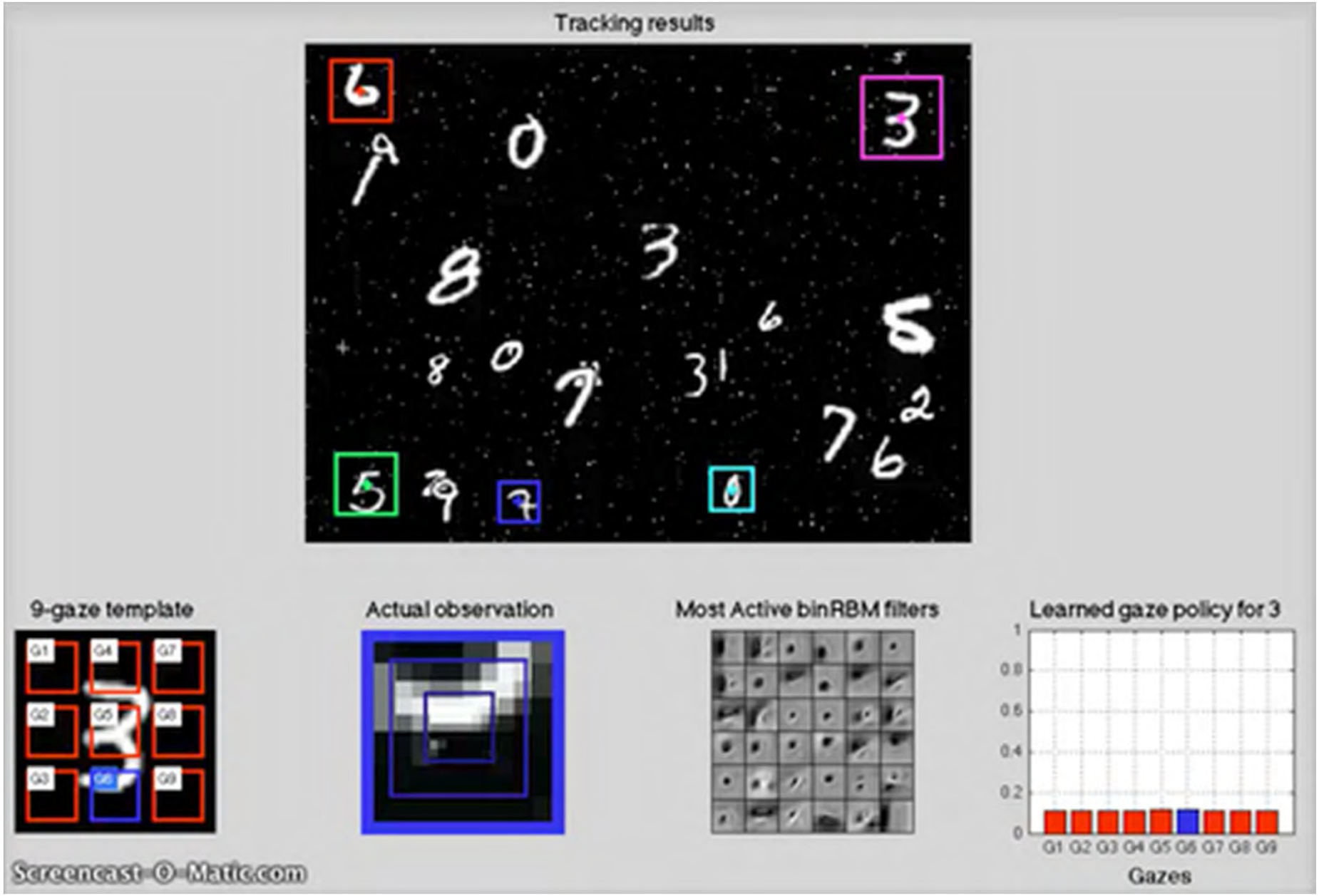
$$\pi_t(a_t = k | R_{t-1}) = \frac{\exp(\eta R_{t-1}(\mathbf{a}_t = k))}{\sum_{j=1}^K \exp(\eta R_{t-1}(\mathbf{a}_t = j))}$$

   - Receive rewards $r_{t,k} = \sum_{i=1}^N \left(w_t^{(i),k}\right)^2$ for $k = 1, \ldots, K$
   - Set $R_t(\mathbf{a}_t = k) = R_{t-1}(\mathbf{a}_t = k) + r_{t,k}$ for $k = 1, \ldots, K$
   - Sample action $k^\star$ according to the policy $\pi_t(\cdot)$.
   - Set $w_t^{(i)} = w_t^{(i),k^\star}$ for $i = 1, \ldots, N$

4. *Selection step*
   - Resample with replacement N particles $\left(\mathbf{x}_{0:t}^{(i)}; i = 1, \ldots, N\right)$ from the set $\left(\tilde{\mathbf{x}}_{0:t}^{(i)}; i = 1, \ldots, N\right)$ according to the normalized importance weights $w_t^{(i)}$.
   - Set $t \leftarrow t + 1$ and go to step 2.

[Denil, Bazzani, Larochelle & dF, 2012]

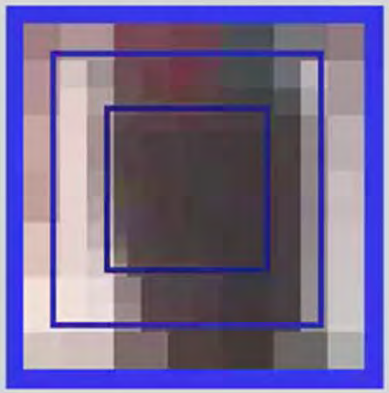
Multi-target: Localization & invariance



Tracking result

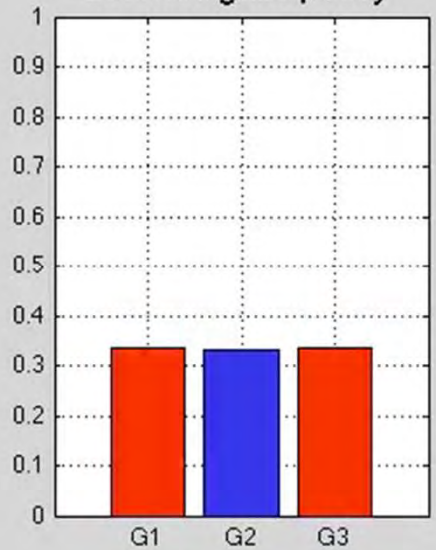
3-gaze template

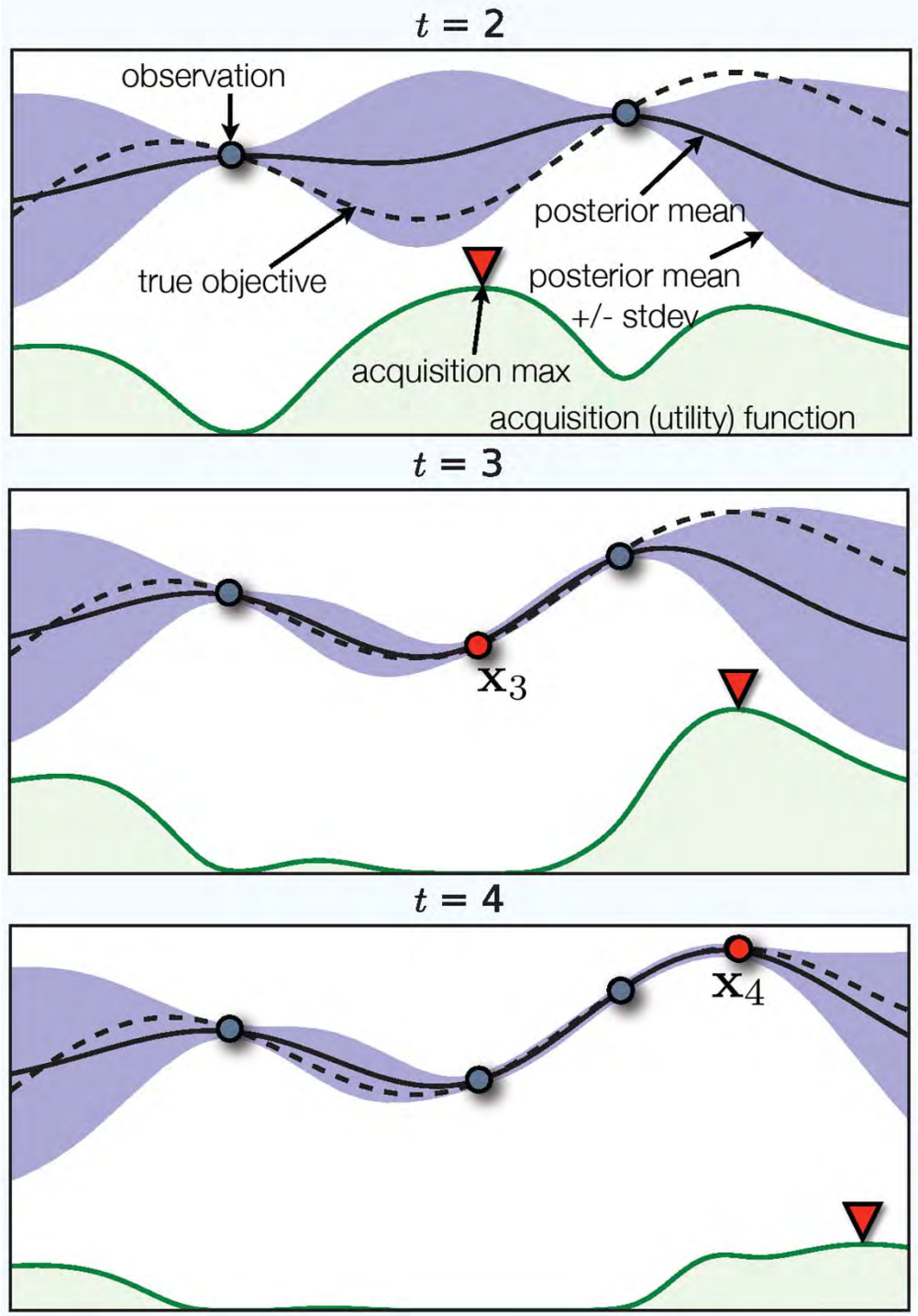
Actual observation



Most Active mcRBM filters

Learned gaze policy





Optimizing the reward function, which is only known point-wise. No need for derivatives.

Bayesian optimization

1: **for** $t = 1, 2, \ldots$ **do**

2: Find $\mathbf{x}_t$ by combining attributes of the posterior distribution in a utility function u and maximizing: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}}\, u(\mathbf{x}|\mathcal{D}_{1:t-1})$.

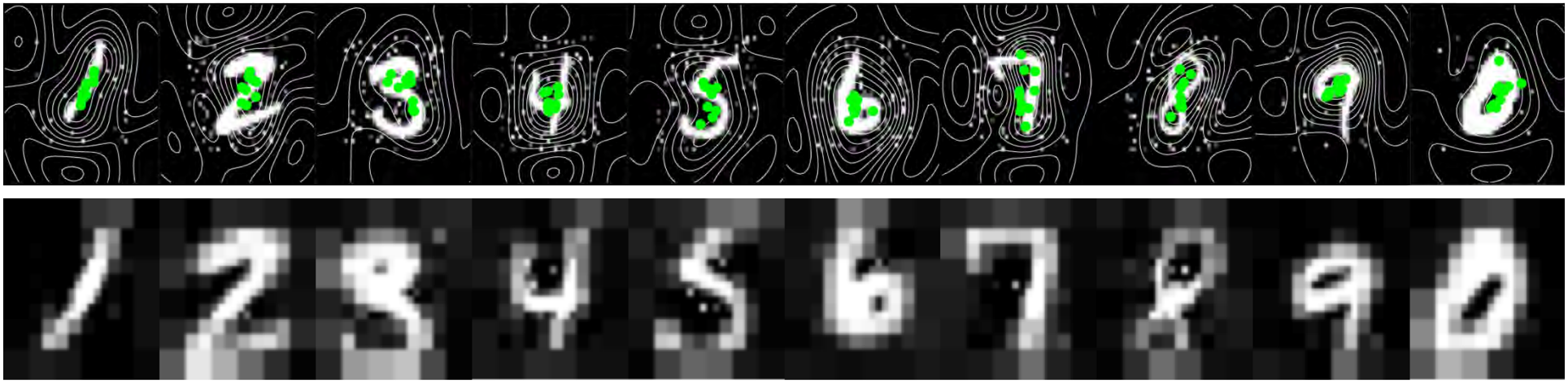
3: Sample the objective function: $y_t = f(\mathbf{x}_t) + \varepsilon_t$.

4: Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.

5: **end for**

Control with Bayesian optimization

Digits Experiment:



Face Experiment: