CPSC 440: Advanced Machine Learning

HMMs and RBMs

Mark Schmidt

University of British Columbia

Winter 2022

Last Time: Expectation Maximization

- EM considers learning with observed data X and hidden data Z.
- In this case the "marginal" log-likelihood has a nasty form,

$$\log p(X \mid \Theta) = \log\left(\sum_{Z} p(X, Z \mid \Theta)\right).$$

- EM applies when "complete" likelihood, $p(X, Z \mid \Theta)$, has a nice form.
- EM iterations take the form of a weighted "complete" NLL,

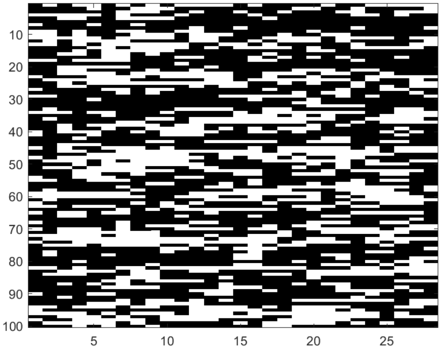
$$\Theta^{t+1} \in \operatorname*{argmax}_{\Theta}\left\{\sum_{Z} \alpha_Z^t \log p(X, Z \mid \Theta)\right\},$$

  where the weights are $\alpha_Z^t = p(Z \mid X, \Theta^t)$ based on previous Θ^t.
- We looked at the simple form of the EM update for mixture models,

$$\Theta^{t+1} \in \operatorname*{argmax}_{\Theta} \sum_{i=1}^{n} \sum_{z^i=1}^{k} \underbrace{p(z^i \mid x^i, \Theta^t)}_{\text{responsibility}} \underbrace{\log p(x^i, z^i \mid \Theta)}_{\text{complete-data log-lik}},$$

Back to the Rain Data

- We previously considered the “Vancouver Rain” data:



- We used homogeneous Markov chains to model between-day dependence.

Back to the Rain Data

- Previously we used a conditional random field to model the month information.
- We could alternately try to learn the clusters using a mixture model.
  - But mixture of independents would not capture dependencies within cluster.
- Mixture of Markov chains could capture direct dependence *and* clusters,

$$p(x_1, x_2, \dots, x_d) = \sum_{c=1}^{k} p(z = c) \underbrace{p(x_1 \mid z = c)p(x_2 \mid x_1, z = c) \cdots p(x_d \mid x_{d-1}, z = c)}_{\text{Markov chain for cluster } c}.$$

- Cluster z chooses which homogeneous Markov chain parameters to use.
  - We could learn that some months are more likely to have rain (like winter months).
  - Can do inference by running forward-bacwkard on each mixture, fit model with EM.

Comparison of Models on Rain Data

- Independent (homogeneous) Bernoulli:
  - Average NLL: 18.97 (1 parameter).
- Independent Bernoullis:
  - Average NLL: 18.95, (28 parmaeters).
- Mixture of Bernoullis ($k = 10$, five random restarts of EM):
  - Average NLL: 17.06 ($10 + 10 \times 28 = 290$ parameters)
- Homogeneous Markov chain:
  - Average NLL: 16.81 (3 parameters)
- Mixture of Markov chains ($k = 10$, five random restarts of EM):
  - Average NLL: 16.53 ($10 + 10 \times 3 = 40$ parameters).
  - Parameters of one of the clusters (possibly modeling summer months):

$$p(z = 5) = 0.14$$

$$p(x_1 = \text{“rain”} \mid z = 5) = 0.22 \qquad \text{(instead of usual 37\%)}$$

$$p(x_j = \text{“rain”} \mid x_{j-1} = \text{“rain”}, z = 5) = 0.49 \qquad \text{(instead of usual 65\%)}$$

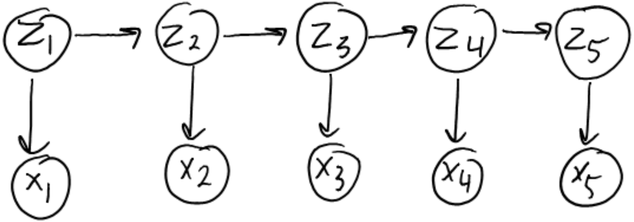
$$p(x_j = \text{“rain”} \mid x_{j-1} = \text{“not rain”}, z = 5) = 0.11 \qquad \text{(instead of usual 35\%)}$$

Back to the Rain Data

- The rain data is artificially divideded into months.
- We previously discussed viewing rain data as one very long sequence ($n = 1$).
- We could apply homogeneous Markov chains due to parameter tieing.
  - But a mixture doesn't make sense when $n = 1$.
- What we want: different "parts" of the sequence come from different clusters.
  - We transition from "summer" cluster to "fall" cluster at some time j.
- One way to address this is with a "hidden" Markov model (HMM):
  - Instead of examples being assigned to clusters, days are assigned to clusters.
  - Have a Markov dependency between cluster values of adjacent days.

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.

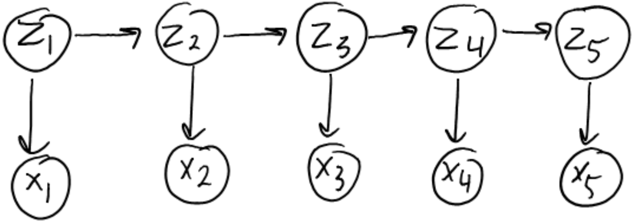


$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- We're going to learn clusters z_j and the hidden dynamics between days.
  - Hidden cluster z_j could be "summer" or "winter" (we're learning the clusters).
  - Transition probability $p(z_j \mid z_{j-1})$ is probability of staying in "summer".
    - Initial probability $p(z_1)$ is probability of starting chain in "summer".
  - Emission probability $p(x_j \mid z_j)$ is probability of "rain" during "summer".

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.

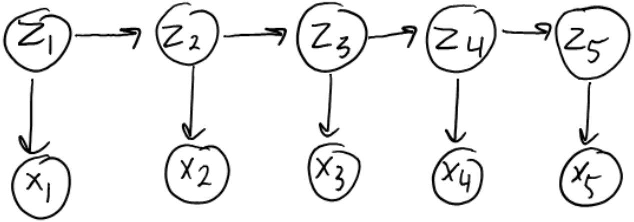


$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- You observe the x_j values but do not see the z_j values.
  - There is a "hidden" Markov chain, whose state determines the cluster at each time.
- HMMs generalize both Markov chains and mixture of categoricals.
  - Both models are obtained under appropriate parameters.

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.



$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- Note that the x_j can be continuous even with discrete clusters z_j.
  - Data could come from a mixture of Gaussians, with cluster changing in time.
- If the z_j are continuous it's often called a state-space model.
  - If everything is Gaussian, it leads to Kalman filtering.
  - Keywords for non-Gaussian: unscented Kalman filter and particle filter.

Applications of HMMs and Kalman Filters

- HMMs variants are probably the most-used time-series model.

Applications [edit]

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

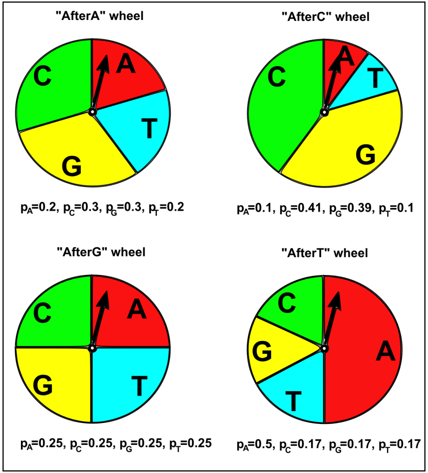
- Single Molecule Kinetic analysis[16]
- Cryptanalysis
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Document Separation in scanning solutions
- Machine translation
- Partial discharge
- Gene prediction
- Alignment of bio-sequences
- Time Series Analysis
- Activity recognition
- Protein folding[17]
- Metamorphic Virus Detection[18]
- DNA Motif Discovery[19]

Applications [edit]

- Attitude and Heading Reference Systems
- Autopilot
- Battery state of charge (SoC) estimation[39][40]
- Brain-computer interface
- Chaotic signals
- Tracking and Vertex Fitting of charged particles in Particle Detectors[41]
- Tracking of objects in computer vision
- Dynamic positioning
- Economics, in particular macroeconomics, time series analysis, and econometrics[42]
- Inertial guidance system
- Orbit Determination
- Power system state estimation
- Radar tracker
- Satellite navigation systems
- Seismology[43]
- Sensorless control of AC motor variable-frequency
- Simultaneous localization and mapping
- Speech enhancement
- Visual odometry
- Weather forecasting
- Navigation system
- 3D modeling
- Structural health monitoring
- Human sensorimotor processing[44]

Example: Modeling DNA Sequences

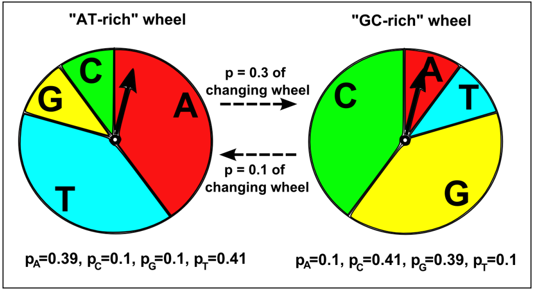
- Previously: Markov chain for DNA sequences:



https://www.tes.com/lessons/WE5E9RncBhieAQ/dna

Example: Modeling DNA Sequences

- Hidden Markov model (HMM) for DNA sequences (two hidden clusters):



- This is a (hidden) state transition diagram.
  - Can reflect that probabilities are different in different regions.
  - The actual regions are not given, but instead are nuissance variables handled by EM.

- A better model might use a hidden and visible Markov chain.
  - With 2 hidden clusters, you would have 8 "probability wheels" (4 per cluster).
  - Would have "treewidth 2", so inference would be tractable.

Inference and Learning in HMMs

- Given observed features x_j, likelihood of a joint z_j assignment is

$$p(z_1, z_2, \dots z_d \mid x_1, x_2, \dots, x_d) \propto p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- We can do inference with forward-backward by converting to potentials:

$$\begin{aligned} \phi_1(z_1) &= p(z_1)p(x_1 \mid z_1) \\ \phi_j(z_j) &= p(x_j \mid z_j) & (j > 1) \\ \phi_{j,j-1}(z_j, z_{j-1}) &= p(z_j \mid z_{j-1}). \end{aligned}$$

- Marginals from forward-backward are used to update parameters in EM.
  - In this setting EM is called the "Baum-Welch" algorithm.
  - As with other mixture models, learning with EM is sensitive to initialization.

Who is Guarding Who?

- There is a lot of data on scoring/offense of NBA basketball players.
  - Every point and assist is recorded, more scoring gives more wins and $$$.

- But how do we measure defense ("stopping people from scoring")?
  - We need to know who each player is guarding (which is not recorded)

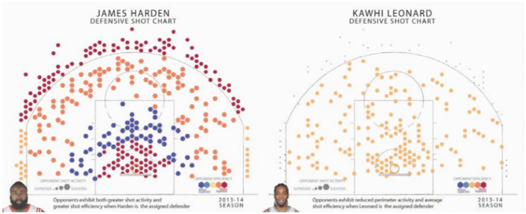


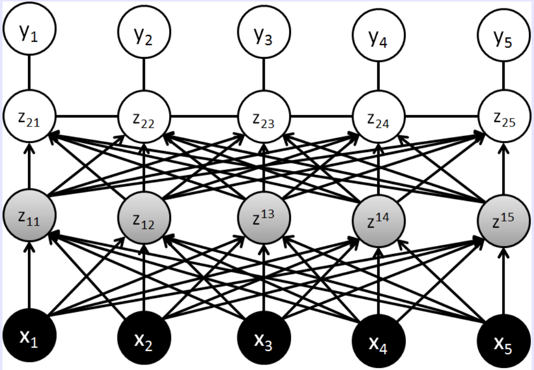
Figure 2a. Graphical depiction of a defender's volume (size) and disruption scores (color). Kawhi Leonard tends to suppress shots on the perimeter. More comparisons are provided in the Appendix.

`http://www.lukebornn.com/papers/franks_ssac_2015.pdf`

- HMMs can be used to model who is guarding who over time.
  - `https://www.youtube.com/watch?v=JvNkZdZJBt4`

Neural Networks with Latent-Dynamics

- Could have (undirected) HMM parameters come out of a neural network:
  - Tries to model hidden dynamics across time.



- Combines deep learning, mixture models, and graphical models.
  - "Latent-dynamics model".
  - Previously achieved among state of the art in several applications.

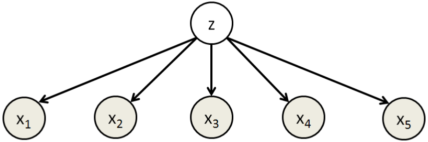
Outline

Mixture of Bernoullis Models

- Recall the mixture of Bernoullis models:

$$p(x) = \sum_{c=1}^{k} p(z = c) \prod_{j=1}^{d} p(x_j \mid z = c).$$

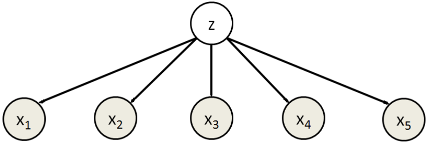
- Given z, each variable x_j comes from a product of Bernoullis



- This is enough to model *any* multivariate binary distribution.
  - But not an efficient representation: number of cluster might need to be huge.
    - Need to learn each cluster independently (no “shared” information across clusters).

Mixture of Independents as a UGM

- The mixture of independents assumptions can be represented as a UGM:



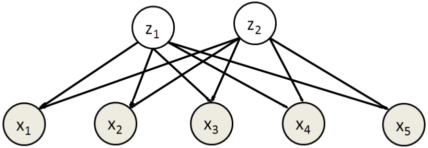
  - "The x_j are independent given the cluster z".
- A log-linear parameterization for $x_j \in \{-1, +1\}$ and $z \in \{-1, +1\}$ could be

$$\phi_j(x_j) = \exp(w_j x_j), \quad \phi_z(z) = \exp(vz), \quad \phi_{j,z}(x_j, z) = \exp(w_j x_j z).$$

- We have three types of parameters:
  - Weight w_j in ϕ_j affects probability of $x_j = 1$ (independent of cluster).
  - Weight v in ϕ_z affecst probability that $z_j = 1$ (prior for cluster).
  - Weight w_j in $\phi_{j,z}$ affects probability that x_j and z are same.
    - Can encourage each binary variable to be same or different than "cluster sign".

"Double Clustering" Model

- Now consider adding a second binary cluster variable:



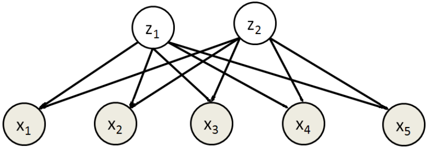
  - "The x_j are independent given both cluster variables z_1 and z_2".
- A log-linear parameterization for $x_j \in \{-1, +1\}$ and $z_c \in \{-1, +1\}$ could be

$$\phi_j(x_j) = \exp(w_j x_j), \quad \phi_c(z_c) = \exp(v_c z_c), \quad \phi_{j,c}(x_j, z_c) = \exp(w_{jc} x_j z)$$

- We have three types of parameters:
  - Weight w_j in ϕ_j affects probability of $x_j = 1$ (independent of cluster).
  - Weight v_c in ϕ_z affecst probability that $z_c = 1$ (prior for cluster variable).
  - Weight w_{jc} in $\phi_{j,z}$ affects probability that x_j and z_c are same.
    - Can encourage each binary variable to be same or different than "cluster variable".

"Double Clustering" Model

- Now consider adding a second binary cluster variable:

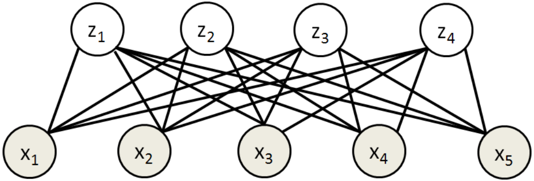


- Have we gained anything?
  - We have 4 clusters based on two hidden variables.
  - Each cluster shares parameters with 2 of the other clusters.

- Hope is to achieve some degree of composition
  - Don't need to re-learn basic things about the x_j in each cluster.
  - Maybe one hidden z_c models clusters, and another models correlations.
    - So that when you use both, you can capture both aspects.

Restricted Boltzmann Machines (RBMs)

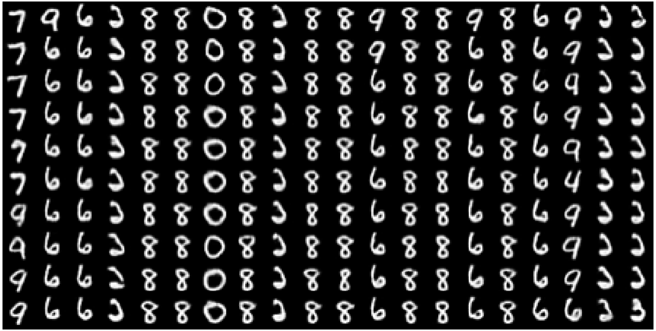
- Now consider adding two more binary latent variables:



- Now we have 16 clusters, in general we'll have 2^k with k hidden binary nodes.
  - This discrete latent-factors give combinatorial number of mixtures.
    - You can think of each z_c as a "part" that can be included or not ("binary PCA").
- This is called a restricted Boltzmann machine (RBM).
  - A Boltzmann machine is a UGM with binary hidden variables.
- It is restricted because all edges are between "visible" x_j and "hidden" z_c.
  - If we know the x_j, then the z_c are independent.
  - If we know the z_c, then the x_j are independent.
  - Inference on both x and z is hard.
    - But we could alternate between Gibbs sampling of all x and all z variables.

Generating Digits with RBMs

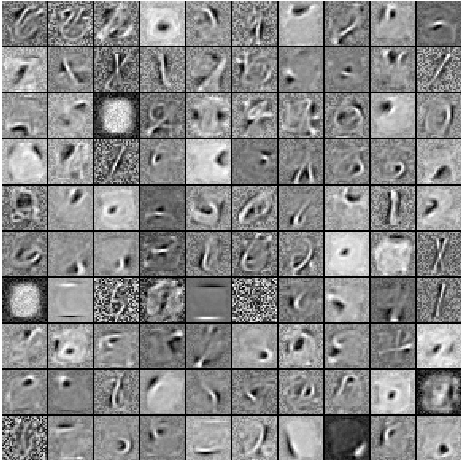
Here are the samples generated by the RBM after training. Each row represents a mini-batch of negative particles (samples from independent Gibbs chains). 1000 steps of Gibbs sampling were taken between each of those rows.



http://deeplearning.net/tutorial/rbm.html

Generating Digits with RBMs

Visualizing each z_c's interaction parameters (w_{jc} for all j) as images:

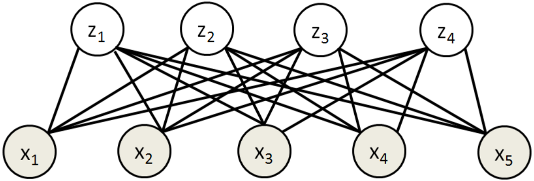


http://deeplearning.net/tutorial/rbm.html

Restricted Boltzmann Machines

- The RBM graph structure leads to a joint distribution of the form

$$p(x,z) = \frac{1}{Z}\left(\prod_{j=1}^{d}\phi_j(x_j)\right)\left(\prod_{c=1}^{k}\phi_c(z_c)\right)\left(\prod_{j=1}^{d}\prod_{c=1}^{k}\phi_{jc}(x_j,z_c)\right).$$



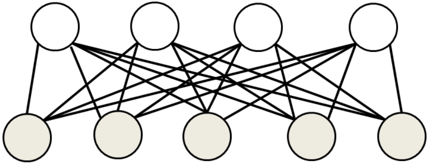
- RBMs usually use a log-linear parameterization like

$$p(x,z) \propto \exp\left(\sum_{j=1}^{d} w_j x_j + \sum_{c=1}^{k} v_c z_c + \sum_{j=1}^{d}\sum_{c=1}^{k} w_{jc} x_j z_c\right),$$

for parameters w_j, v_c, and w_{jc} (variants exist for non-binary x_j).

Learning UGMs with Hidden Variables

- For RBMs we have hidden variables:



- With hidden ("nuisance") variables z the observed likelihood has the form

$$p(x) = \sum_z p(x,z) = \sum_z \frac{\tilde{p}(x,z)}{Z}$$
$$= \frac{1}{Z} \underbrace{\sum_z \tilde{p}(x,z)}_{Z(x)} = \frac{Z(x)}{Z},$$

where $Z(x)$ is the partition function of the conditional UGM given x.
  - $Z(x)$ is cheap in RBMs because the z are independent given x.

Learning UGMs with Hidden Variables

- This gives an observed NLL of the form

$$-\log p(x) = -\log(Z(x)) + \log Z,$$

  where $Z(x)$ sums over hidden z values, and Z sums over z and x.

- The second term is convex but the first term is non-convex.
  - This is expected when we have hidden variables.

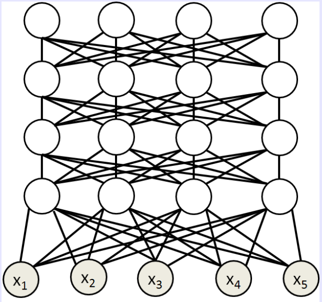
- With a log-linear parameterization, the gradient has the form

$$-\nabla \log p(x) = -\mathbb{E}_{z \mid x}[F(X, Z)] + \mathbb{E}_{z,x}[F(X, Z)].$$

- For RBMs, first term is cheap due to independence of z given x.
- We can approximate second term using block Gibbs sampling.
  - For other problems, you would also need to approximate first term.

Deep Boltzmann Machines

- 15 years ago, a hot topic was "stacking RBMs", as in deep Boltzmann Machine:



- Part of the motivation for people to re-consider "deep" models.
- Model above allows block Gibbs sampling "by layer".
    - Variables in layer are conditionally independent given layer above and below.

Deep Boltzmann Machines

- Performance of deep Boltzmann machine on NORB data:

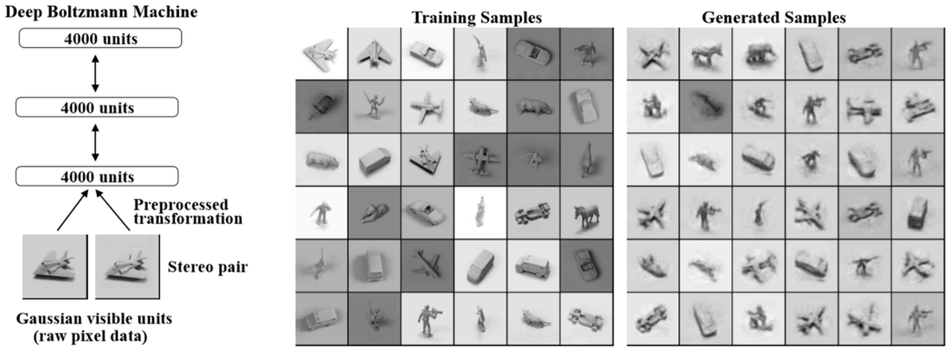
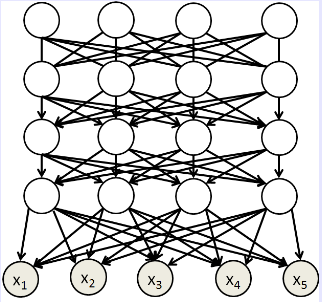


Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

http://www.cs.toronto.edu/~fritz/absps/dbm.pdf

Deep Belief Networks

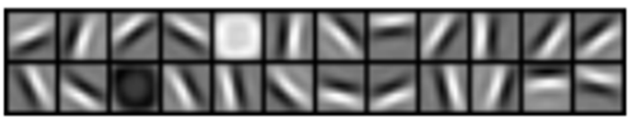
- There were also deep belief networks where RBM outputs DAG layers.



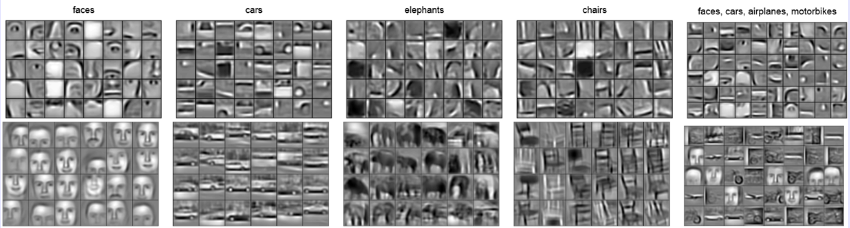
- More difficult to train and do inference due to explaining away.
- Though easier to sample using ancestral sampling.

Cool Pictures Motivation for Deep Learning

- First layer of z_i in a convolutional deep belief network:



- Visualization of second and third layers trained on specific objects:



http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

- Many classes use these particular images to motivate deep neural networks.
  - But they're not from a neural network: they're from a deep DAG model.

Summary

- Hidden Markov models model time-series with hidden per-time cluster.
  - Tons of applications, typically more realistic than Markov models.
- Restricted Boltzmann machines (RBMs):
  - UGMs with binary hidden variables.
  - Pairwise edges only between visible and hidden.
    - Allows efficient block Gibbs sampling for inference and learning.
  - Deep Boltzmann machines "stack" RBMs into a deep density estimation model.

- Next time: modeling cancer mutation signatures.