# CPSC 540: Machine Learning

## Hidden Markov Models, Boltzmann Machines

Mark Schmidt

University of British Columbia

Winter 2017

# Admin

- Assignment 4:
  - Due Monday.
- Interested in TAing CPSC 340 in the summer?
  - Contact Mike Gelbart.
- Suggestions from unofficial course evals:
  - Split into 2 courses.
  - Post lecture slides without transitions.
  - Supplementary documents/notes/book.
  - Extra office hours on Tuesdays at 2:30.

# Last Time: Approximate Inference

- We've been discussing graphical models for density estimation,

$$p(x_1, x_2, \dots, x_d) = \prod_{j=1}^{d} p(x_j | x_{\mathsf{pa}(j)}), \quad p(x_1, x_2, \dots, x_d) \propto \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

  where are natural and widely-used models for many phenomena.
  - These will also be among ingredients of more advanced models we'll see later.

- But typical calculations involving graphical models are typically NP-hard.
  - We can convert to DAGs to UGMs, so we'll just study UGMs.

- We considered approximate inference in discrete UGMs:
  1. Iterated conditional mode (ICM) algorithm for approximate decoding.
  2. Gibbs sampling MCMC algorithm for approximate sampling.
  3. Mean-field variational method for approximate marginals.

# Pseudo-Code for ICM

- ICM is a coordinate-wise method for approximate decoding:
  - Choose a coordinate $i$ to update.
  - Maximize $x_i$ keeping other variables fixed.

- Consider a pairwise UGM:

$$p(x_1, x_2, \dots, x_d) \propto \left(\prod_{i=1}^{d} \phi_i(x_i)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right),$$

and consider updating a node $i$ that only has 2 neighbours ($j$ and $k$):

1. Compute $M_i(x_i) = \phi_i(x_i)\phi_{ij}(x_i, x_j)\phi_{ik}(x_i, x_k)$ for all $x_i$.
2. Set $x_i$ to the largest value of $M_i(x_i)$.

# Pseudo-Code for Gibbs Sampling

- Gibbs sampling is a coordinate-wise method for approximate sampling:
  - Choose a coordinate $i$ to update.
  - Sample $x_i$ keeping other variables fixed.

- Consider a pairwise UGM:

$$p(x_1, x_2, \ldots, x_d) \propto \left(\prod_{i=1}^{d} \phi_i(x_i)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right),$$

and consider updating a node $i$ that only has 2 neighbours ($j$ and $k$):

1. Compute $M_i(x_i) = \phi_i(x_i)\phi_{ij}(x_i, x_j)\phi_{ik}(x_i, x_k)$ for all $x_i$.
2. Sample $x_i$ proportional to $M_i(x_i)$.

# Pseudo-Code for Mean FIeld

- Mean field is a coordinate-wise method for approximate marginals:
  - Choose a coordinate $i$ to update.
  - Update $\underbrace{q_i(x_i)}_{\text{for all } x_i}$ keeping other variables fixed ($q_i(x_i)$ approximates $p_i(x_i)$).

- Consider a pairwise UGM:

$$p(x_1, x_2, \dots, x_d) \propto \left(\prod_{i=1}^{d} \phi_i(x_i)\right)\left(\prod_{(i,j)\in E} \phi_{ij}(x_i, x_j)\right),$$

and consider updating a node $i$ that only has 2 neighbours ($j$ and $k$):

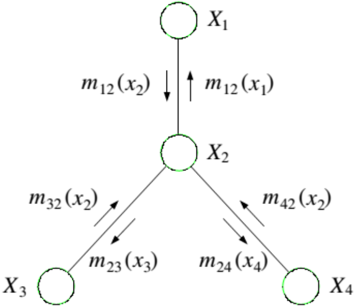
1. Compute $M_i(x_i) = \exp\left(\sum_{x_j} q_j(x_j)\log\phi_{ij}(x_i, x_j) + \sum_{x_k} q_k(x_k)\log\phi_{ik}(x_i, x_k)\right)$.
2. Set $q_i(x_i)$ proportional to $\phi_i(x_i)M_i(x_i)$.

# Last Time: Belief Propagation

- We discussed belief propagation for forest-structured UGMs.

  (undirected graphs with no loops, which must be pairwise)

- Belief propagation is a message-passing algorithm with a specific message order.
  - "Forward pass" away from root, and "backward" pass from leaves to root.




https://www.quora.com/
Probabilistic-graphical-models-what-are-the-relationships-between-sum-product-algorithm-belief-propagation-and-junction-tree-

# Last Time: Belief Propagation

- Belief propagation "messages" have the form:

$$M_{ic}(x_c) \propto \sum_{x_i} \phi_i(x_i)\phi_{ic}(x_i, x_c)M_{ji}(x_i)M_{ki}(x_i),$$

  when we're sending to "child" $c$ after receiving messages from "parents" $j$ and $k$.

- We obtain the "forward" and "backward" Markov chain messages with 1 parent.

- Univariate marginals are proportional to $\phi_i(x_i)$ times all "incoming" messages.

- Replace $\sum_{x_i}$ with $\max_{x_i}$ for decoding.
  - "Sum-product" and "max-product" algorithms.

# Loopy Belief Propagation

- Belief propagation "messages" have the form:

$$M_{ic}(x_c) \propto \sum_{x_i} \phi_i(x_i)\phi_{ic}(x_i, x_c)M_{ji}(x_i)M_{ki}(x_i),$$

  when we're sending to "child" $c$ after receiving messages from "parents" $j$ and $k$.

- A "hacker" approach to approximate marginals (loopy belief propagation):
  - Choose an edge $ic$ to update.
  - Update messages $M_{ic}(x_c)$ keeping all other messages fixed.
  - Repeat until "convergence".

- Empirically much better than mean field, we've spent 20 years figuring out why.

# Discussion of Loopy Belief Propagation

- Loopy BP locally minimizes KL, but isn't optimizing an objective.
  - Convergence of loopy BP is hard to characterize: does not converge in general.

- Mean-field optimizes "Gibbs mean-field free energy": a lower bound on $Z$.
- If it converges loopy BP finds fixed point of "Bethe free energy":
  - Not a bound but a better approximation than mean-field.

- Recent works give convex variants that upper bound $Z$.
  - Tree-reweighted belief propagation.

- Only has closed-form update for Gaussian/discete UGMs.
  - Can approximate non-Gaussian/discrete models using expectation propagation.

- For details on the above, see `people.eecs.berkeley.edu/~wainwrig/Papers/WaiJor08_FTML.pdf`

# Outline

# Closure of UGMs under Conditioning

- UGMs are closed under conditioning:
  - If $p(x)$ is a UGM, then $p(x_A|x_B)$ can be written as a UGM (for partition $A$ and $B$).

- Consider a 4-node chain-structured UGM, $(x_1) - (x_2) - (x_3) - (x_4)$,

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{34}(x_3, x_4).$$

- Conditioning on $x_2$ and $x_3$ gives UGM over $x_1$ and $x_4$ (tedious: bonus slide)

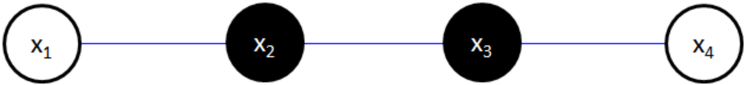
$$p(x_1, x_4|x_2, x_3) = \frac{1}{Z'}\phi_1'(x_1)\phi_4'(x_4),$$

where new potentials "absorb" the shared potentials with observed nodes:

$$\phi_1'(x_1) = \phi_1(x_1)\phi_{12}(x_1, x_2), \quad \phi_4'(x_4) = \phi_4(x_4)\phi_{34}(x_3, x_4).$$

# Closure of UGMs under Conditioning

- Conditioning on $x_2$ and $x_3$ in a chain,



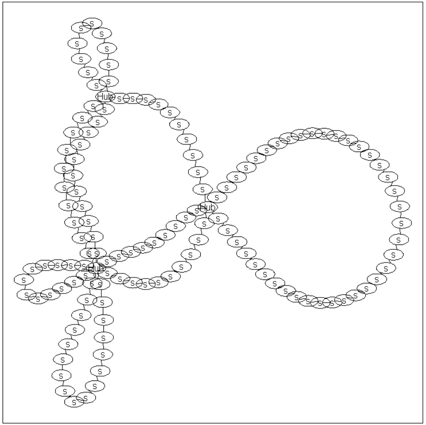

gives a UGM defined on $x_1$ and $x_4$ that is disconnected:


- Graphically, we "erase the black nodes and their edges".

- Notice that inference in the conditional UGM may be mucher easier.

# Inference in Conditional UGM

- Consider the following graph which could describe bus stops:



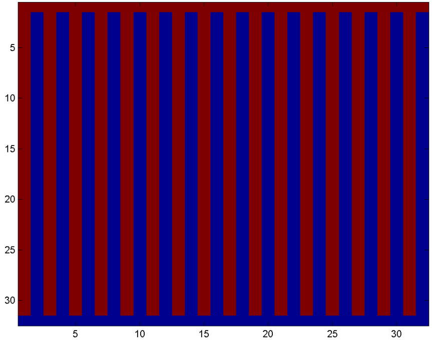

- If we condition on the “hubs”, the graph forms a forest (and inference is easy).

# Block-Structured Approximate Inference

- Basic approximate inference methods like ICM and Gibb sampling:
  - Update one $x_j$ at a time.
  - Efficient because conditional UGM is 1 node.
- Better approximate inference methods use block updates:
  - Update a set of $x_j$ at once.
  - Efficient if conditional UGM allows exact inference.
- A common choice is tree-structured blocks.

# Block-Structured Approximate Inference

- Dividing a lattice into two tree-structured blocks:

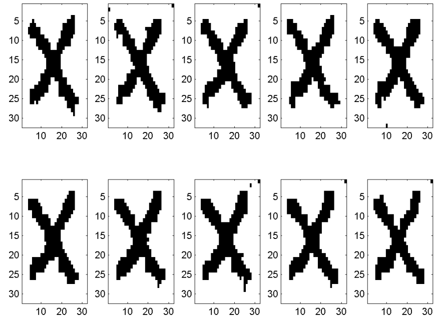



- We can maximize/sample blue pixels given red pixels, and vice versa.
  - We could also consider random tree-structured sub-graphs.

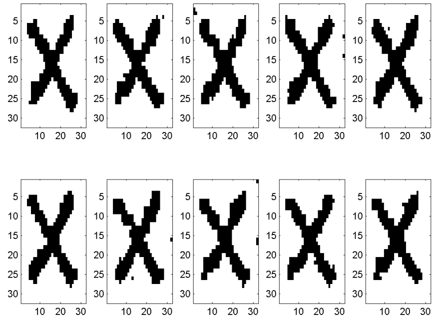
# Block Gibbs Sampling in Action

Gibbs vs. tree-structured block-Gibbs samples:

Samples from Gibbs sampler



Samples from Block Gibbs sampler



# Discussion of Advanced Inference Methods

- Block versions of basic methods:
  - Block ICM.
  - Block Gibbs sampling.
  - Structured mean field (disjoint blocks).
  - Generalized belief propagagtion (disjoint blocks).

- We can do exact decoding in binary pairwise UGMs with "attractive potentials",

$$\log \phi_{ij}(1,1) + \log \phi_{ij}(2,2) \geq \log \phi_{ij}(1,2) + \log \phi_{ij}(2,1),$$

  as a graph cut problem (very widely-used in computer vision).
  - Alpha-beta swaps and alpha-expansions do block updates with this operation.
  - Analogous sampling method is Swendson-Wang.

- The final class of approximate inference methods are convex relaxations:
  - Formulate decoding or inference as an integer linear program.
  - Approximate this with a linear program or sem-definite program.

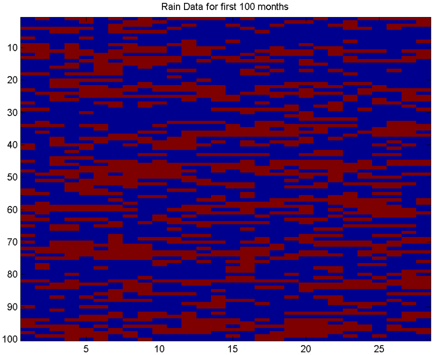
# Outline

# Where we are where we're going...

- Last $n$ lectures: four topics related to density estimation:
  1. Mixture models can model clusters in the data.
  2. Latent-factor models consider interacting hidden factors in the data.
  3. Graphical models can model direct dependencies between variables.
  4. Approximate inference is needed when probabilities are too complicated
- Each has many applications, but they're limited/boring on their own.
- But by combining them we get very powerful models.
  - Next time we'll start combining them with supervised learning tricks from 340.

# Back to the Rain Data

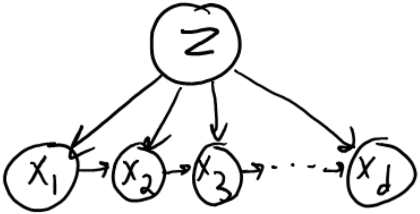
- We previously considered the "Vancouver Rain" data:




- We said that a homogeneous Markov chain is a good model:
  - Captures direct dependency between adjcaent days.

# Back to the Rain Data

- But doesn't it rain less in the summer?
- There are hidden clusters in the data not captured by the Markov chain.
  - But mixture of independent models are innefficient at representing direct dependency.
- We can capture direct dependence *and* clusters with mixture of Markov chains:



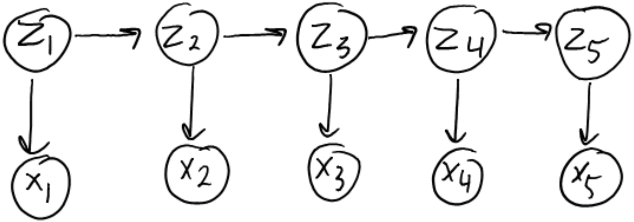

- Cluster $z$ chooses which homogeneous Markov chain parameters to use.
  - We could learn that we're more likely to have rain in wineter.
  - Graph has treewidth of 2: exact inference and EM will be cheap.

# Back to the Rain Data

- The rain data is artificially divideded into months.
- Consider viewing rain data as one very long sequence ($n = 1$).
- This doesn't affect homogeneous Markov chain because of parameter tieing.
- But a mixture doesn't make sense when $n = 1$.
- One way to address this:
  - Let each day have it's own cluster.
  - Have a Markov dependency between cluster values of adjacent days.

# Hidden Markov Models

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

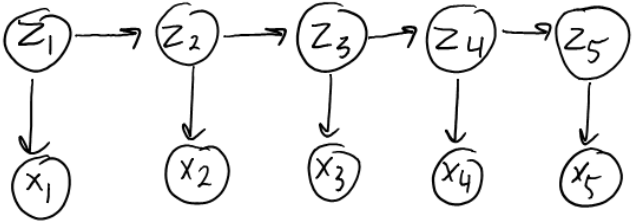



- For the rain data, cluster $z_j$ could be "rainy season" or "dry season".
  - Each $x_j$ is spit out based on $z_j$, the value of our cluster at time $j$.
  - We model probability staying in same $z_j$ or transitioning to another.

- Inference is easy in this model: it's a tree.
- Learning with EM is also easy due to chain-structured $z_j$ dependence:
  - Convert to UGM, conditioning on $x_j$ gives a chain, run forward-backward.

# Hidden Markov Models

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




- Note that the $x_j$ can be continuous even with discrete clusters $z_j$.

- 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.

- Variants of HMMs are probably the most-used time-series model...

# Applications of HMMs and Kalman Filters

## 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 drives
- Simultaneous localization and mapping
- Speech enhancement
- Visual odometry
- Weather forecasting
- Navigation system
- 3D modeling
- Structural health monitoring
- Human sensorimotor processing[44]

# Who is Guarding Who?

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

- But how do we measure defense?
  - We need to know who each player is guarding.

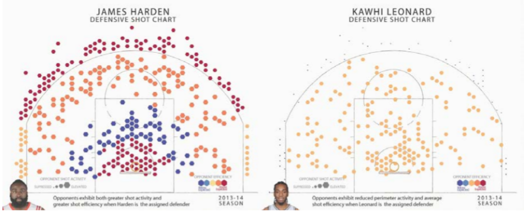



**Figure 2a.** Graphical depiction of a defender's volume (size) and disruption scores (color). Kawhi Leonard tends to suppress shots on the perimeter. More comparisons are provided in the Appendix.

`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`

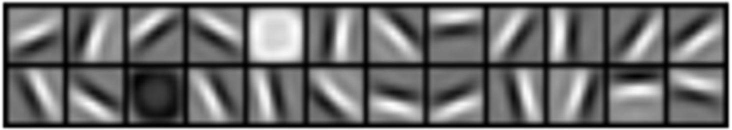
# Outline

# Deep Density Estimation

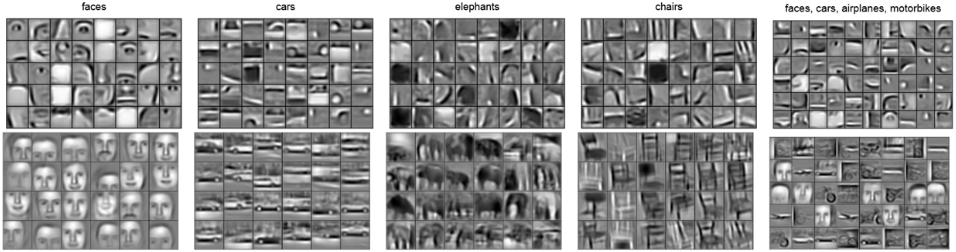
- In 340 we discussed supervised deep learning.
  - And autoencoders as a form of unsupervised learning.
- Does it make sense to talk about deep density estimation?
- Standard argument:
  - Human learning seems to be mostly unsuperivsed.
  - Could we learn unsupervised models with much less data?
- Deep belief networks started modern deep learning movement (2006).
  - One of first non-convolutional deep networks that people got working.

# Cool Pictures Motviation for Deep Learning

- First layer of $z_i$ trained on 10 by 10 image patches:



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



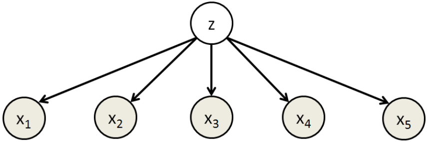

http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

# Mixture of Independent Models

- Recall the mixture of independent models:

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

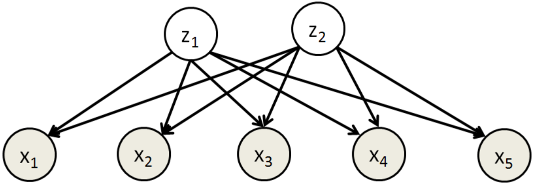
- Given $z$, each variable $x_j$ comes from some "nice" distribution.




- This is enough to model *any* distribution.
  - Just need to know cluster of $x$ and distribution of $x_j$ given $z$.
  - But not efficient representation: number of cluster might be be huge.

# Latent DAG Model

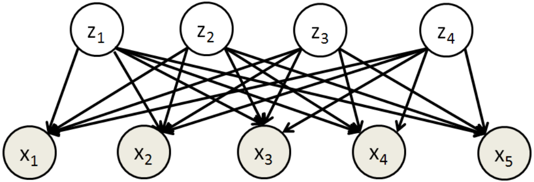
- Consider the following model with binary $z_1$ and $z_2$:




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

# Latent DAG Model

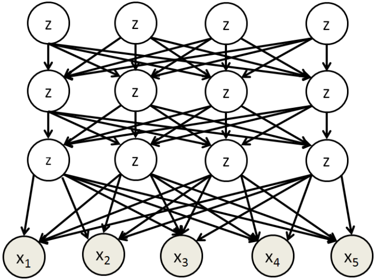
- Consider the following model:




- Now we have 16 clusters, in general we'll have $2^k$ with $k$ hidden nodes.
  - The discrete latent-factors give combinatorial number of mixtures.
  - We'll assume $p(x_j|z_1, z_2, z_3, z_4)$ is a linear model (Gaussian, logistic, etc.).
    - Distributed representation where $x$ is made of parts $z$.
    - With $d$ visible $x_j$ and $k$ hidden $z_j$, we only have $dk$ parameters.

# Deep Belief Networks

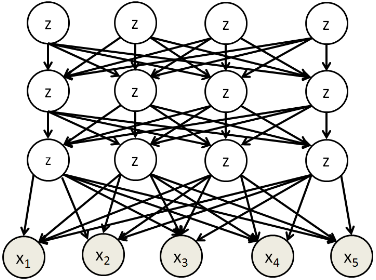
- Deep belief networks add more binary hidden layers:




- Data is at the bottom.
- First hidden layer could be "basic ingredients".
- Second hidden layer could be general "parts".
- Third hidden layer coul be "abstract concept".

# Deep Belief Networks

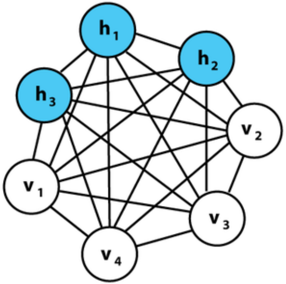
- Deep belief networks add more binary hidden layers:




- If we were conditioning on *top* layer:
  - Sampling would be easy.
- But we're conditioning on the *bottom* layer:
  - Everything is hard.
  - There is combinatorial "explaining away".
- Common training method:
  - Greedy "layerwise" training as a restricted Boltzmann machine.

# Boltzmann Machines

- Boltzmann machines are UGMs with binary latent variables:

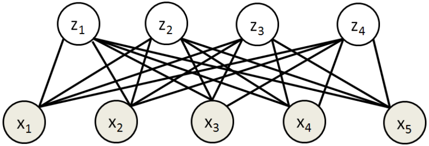



https://en.wikipedia.org/wiki/Boltzmann_machine

- Yet another latent-variable model for density estimation.
  - Hidden variables again give a combinatorial latent representation.
- Hard to do anything in this model, even if you know all the $z$.

# Restricted Boltzmann Machine

- By restricting graph structure, some things get easier:
  - Restricted Boltzmann machines (RBMs): edges only between the $x_j$ and $z_c$.

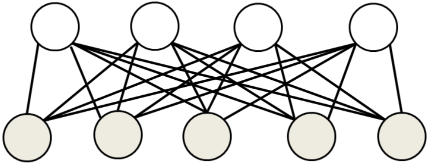



  - Given visible $x$, inference on $z$ is easy:
    - E.g., block Gibbs sampling is just sampling each $z_c$ independently.
  - Given hidden $z$, inference on $x$ is easy:
    - E.g., block Gibbs sampling is just sampling each $x_j$ independently.

  - Standard training method:
    - Use block Gibbs sampling to approximate gradient (next time).

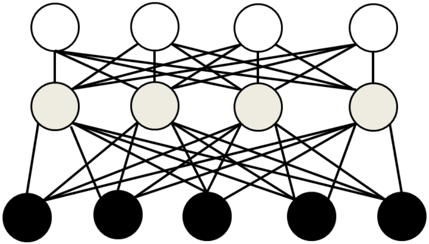
# Greedy Layerwise Training of Stacked RBMs

- Step 1: Train an RBM.



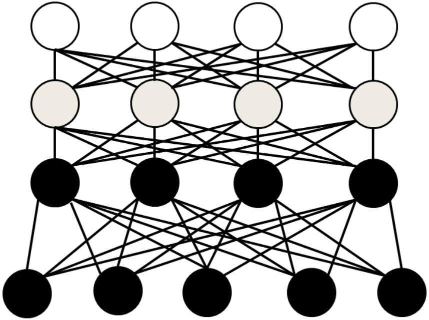
# Greedy Layerwise Training of Stacked RBMs

- Step 1: Train an RBM.
- Step 2:
  - Fix first hidden layer values.
  - Train an RBM.



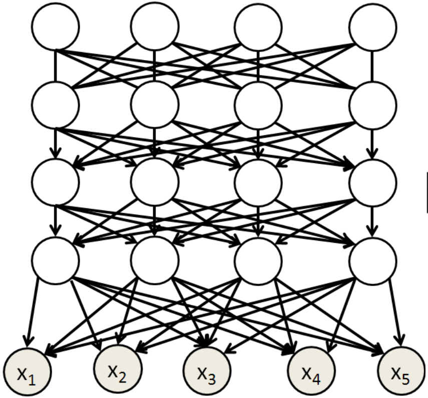
# Greedy Layerwise Training of Stacked RBMs

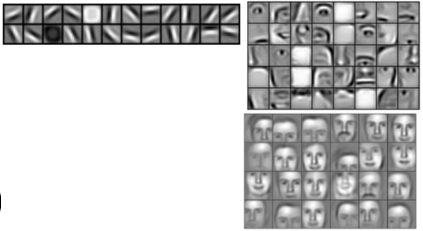
- Step 1: Train an RBM.
- Step 2:
  - Fix first hidden layer values.
  - Train an RBM.
- Step 3:
  - Fix second hidden layer values.
  - Train an RBM.



# Deep Belief Networks

- Keep top as an RBM.
- For the other layers, use DAG parameters that implement block sampling.
  - Can sample by runing block Gibbs on top layer for a while, then ancestral sampling.

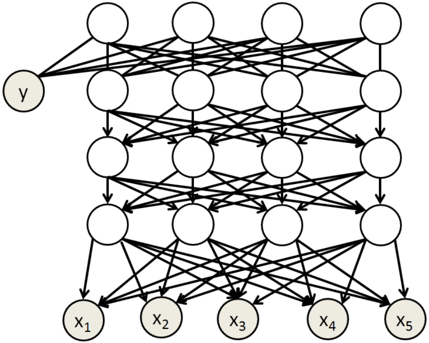


http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

# Deep Belief Networks

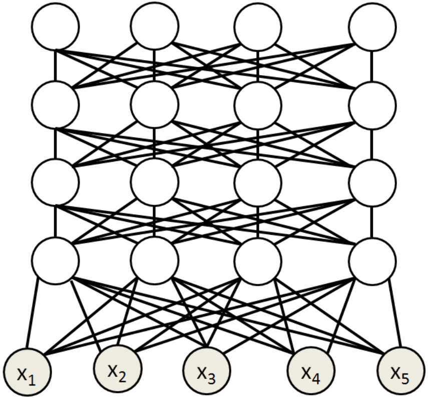
- Can add a class label to last layer.




- Can use "fine-tuning" as a feedforward neural network to refined weights.
  - `https://www.youtube.com/watch?v=KuPai0ogiHk`

# Deep Boltzmann Machines

- Deep Boltzmann machines just keep as an undirected model.
  - Sampling is nicer: no explaning away within layers.
  - Variables in layer are independent given variables in layer above and below.



# Deep Boltzmann Machines

- Performance of deep Boltzmann machine on NORB data:

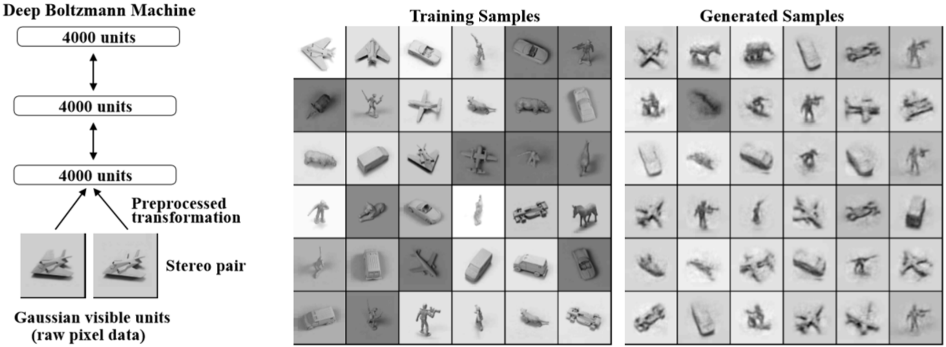



Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

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

# Summary

- Loopy belief propagation is a heuristic for estimating marginals.
- Conditioning in UGMs leads to a smaller/simpler UGM.
- Block approximate inference works better than single-variable methods.
- Hidden Markov models model time-series with latent factors.
- Boltzmann machines are UGMs with binary hidden variables.
    - Restricted Boltzmann machines only allow connections between hidden/visible.
- Deep belief networks and Boltzmann machines have layers of hidden variables.

- Next time: we'll use these tools for supervised learning.

# Bonus Slide: Conditioning in UGMs

- Conditioning on $x_2$ and $x_3$ in 4-node chain-UGM gives

$$
\begin{aligned}
p(x_1, x_4 | x_2, x_3) &= \frac{p(x_1, x_2, x_3, x_4)}{p(x_2, x_3)} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \frac{1}{Z}\phi_1(x_1')\phi_2(x_2)\phi_3(x_3)\phi_4(x_4')\phi_1(x_1', x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4')} \\
&= \frac{\frac{1}{Z}\phi_1(x_1)\phi_2(x_2)\phi_3(x_3)\phi_4(x_4)\phi_1(x_1, x_2)\phi_2(x_2, x_3)\phi_3(x_3, x_4)}{\frac{1}{Z}\phi_2(x_2)\phi_3(x_3)\phi_2(x_2, x_3)\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1(x_1)\phi_4(x_4)\phi_1(x_1, x_2)\phi_3(x_3, x_4)}{\sum_{x_1', x_4'} \phi_1(x_1')\phi_4(x_4')\phi_1(x_1', x_2)\phi_3(x_3, x_4')} \\
&= \frac{\phi_1'(x_1)\phi_4'(x_4)}{\sum_{x_1', x_4'} \phi_1'(x_1')\phi_4'(x_4')}
\end{aligned}
$$

# Bonus Slide: Other Graphical Models

- Factor graphs: we use a square between variables that appear in same factor.
  - Can distinguish between a 3-way factor and 3 pairwise factors.
- Chain-graphs: DAGs where each block can be a UGM.
- Ancestral-graph:
  - Generalization of DAGs that is closed under conditioning.
- Structurla equation models: generalization of DAGs that allows cycles.