CPSC 440: Advanced Machine Learning

Log-Linear Models

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: Approximate Inference

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

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

  where are natural and widely-used models for many phenomena.
  - These will also be among ingredients of more advanced models we'll see later.
- For high-treewidth graphs, we considered approximate inference methods:
  1. Iterated conditional mode (ICM) applies coordinate-wise optimization.
  2. Gibbs sampling applies coorrdinate-wise sampling.
     - A special case of Markov chain Monte Carlo (MCMC).
     - ICM and Gibbs work better if you update blocks with low treewidth.
- For binary pairwise UGMs with "attractive" potentials,

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

  we can do exact decoding efficiently for any treewidth via "graph cuts".

Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- If we have more than 2 states, we can't use graph cuts.
- Alpha-beta swaps are an approximate decoding method for "pairwise attractive",

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta, \beta) \geq \log \phi_{ij}(\alpha, \beta) + \log \phi_{ij}(\beta, \alpha).$$

  - Each step choose an α and β, optimally "swaps" labels among these nodes.
- Alpha-expansions are another variation based on a slightly stronger assumption,

$$\log \phi_{ij}(\alpha, \alpha) + \log \phi_{ij}(\beta_1, \beta_2) \geq \log \phi_{ij}(\alpha, \beta_1) + \log \phi_{ij}(\beta_2, \alpha).$$

  - Steps choose label α, and consider replacing the label of any node not labeled α.

Alpha-Beta Swap and Alpha-Expansions: ICM with Graph Cuts

- These don't find global optima in general, but make huge moves:

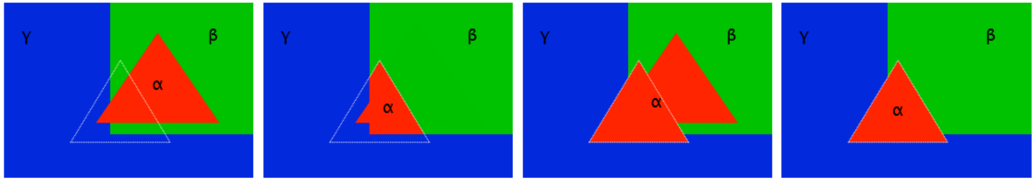
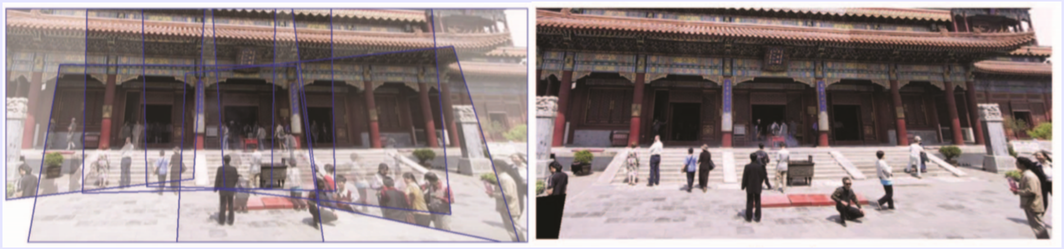


Figure 1: From left to right: Initial labeling, labeling after $\alpha\beta$-swap, labeling after α-expansion, labeling after α-expansion β-shrink. The optimal labeling of the α pixels is outlined by a white triangle, and is achieved from the initial labeling by one ~~α-expansion β-shrink move~~. $\alpha\beta$-swap move

- A somewhat-related MCMC method is the Swendson-Wang algorithm.

Example: Photomontage

- Photomontage: combining different photos into one photo:

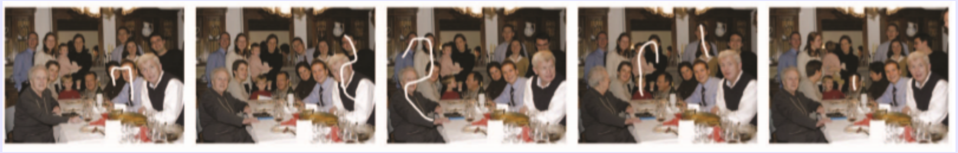


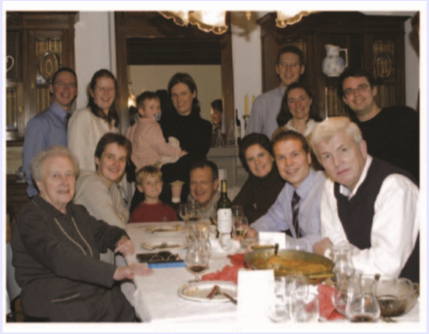
http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

- Here, x_i corresponds to identity of original image at position i.

Example: Photomontage

- Photomontage: combining different photos into one photo:





http://vision.middlebury.edu/MRF/pdf/MRF-PAMI.pdf

Outline

Structured Prediction with Undirected Graphical Models

- Consider a pairwise UGM,

$$p(x) = \frac{1}{Z}\left(\prod_{j=1}^{d}\phi_j(x_j)\right)\left(\prod_{(j,k)\in E}\phi_{jk}(x_j,x_k)\right).$$

- We've been focusing on the case where the potentials ϕ are known.
  - We've discussed exact inference, and approximate decoding and sampling.
  - We've discussed [block-]coordinate approximate inference.

- We're now going to discuss learning the potentials ϕ from data.

- Unfortunately, Z makes this complicated compared to DAGs.
  - You can't fit each potential independently.

Naive Parameterization of UGMs

- We'll want to make the ϕ depend on a set of parameters w.
- As before, with n IID training x^i we can do MAP estimation,

$$w = \operatorname*{argmin}_w - \sum_{i=1}^n \log p(x^i \mid w) + \frac{\lambda}{2}\|w\|^2,$$

  where I've assumed an independent Gaussian prior on w.
- A naive parameterization is to just directly treat potentials as parameters:

$$\phi_j(s) = w_{j,s}, \quad \phi_{jk}(s,s') = w_{j,k,s,s'},$$

  so $w_{j,s}$ is "potential of node j being in state s".
  - And optimize subject to all parameters being non-negative.
  - This unfortunately leads to a non-convex optimizaiton.

Log-Linear Parameterization of UGMs

- Instead of using non-negative w, we can instead exponentiate w,

$$\phi_j(s) = \exp(w_{j,s}), \quad \phi_{jk}(s,s') = \exp(w_{j,k,s,s'}).$$

- This gives a log-linear model,

$$\begin{aligned} p(x \mid w) &\propto \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{(j,k)\in E} \phi_{jk}(x_j,x_k)\right) \\ &= \exp\left(\sum_{j=1}^{d} w_{j,x_j} + \sum_{(j,k)\in E} w_{j,k,x_j,x_k}\right), \end{aligned}$$

  and leads to a convex NLL.
  - Normally, exponentiating to get non-negativity introduces local minima.

Parameter Tieing in UGMs

- So our log-linear parameterization has the form

$$\log \phi_j(s) = w_{j,s}, \quad \log \phi_{jk}(s, s') = w_{j,k,s,s'},$$

  which can represent any positive pairwise potentials.

- There exist many common variations on parameter tieing:
  - We might want w_{j,x_j} to be the same for all j (all nodes use same potentials).
    - You can similarly tie the edge parameters across all edges.
    - This is similar to homogenous Markov chains.
  - In the Ising model we tied across states: $w_{j,k,1,1} = w_{j,k,2,2}$ and $w_{j,k,1,2} = w_{j,k,2,1}$.
  - We could also have special potentials for the boundaries.
    - Many language models are homogeneous, except for start/end of sentences.

Energy Function and Feature Vector Representation

- Recall that we use $\tilde{p}(x)$ for the unnormalized probability,

$$p(x) = \frac{\tilde{p}(x)}{Z}.$$

- In physics, the value $E(x) = -\log \tilde{p}(x)$ is called the energy function.

- With the log-linear parameterization, the energy function is linear,

$$-E(X) = \sum_{j} w_{j,x_j} + \sum_{(j,k)\in E} w_{j,k,x_j,x_k}.$$

- To account for parameter tieing, we often write

$$-E(x) = w^T F(x), \quad \text{or equivalently} \quad p(x) \propto \exp(w^T F(x)),$$

  where feature function F counts number of times we use each parameter.
  - Includes usual softmax as a special case.

Example of Feature Function

- Consider the 2-node 1-edge UGM (1)–(2), where each state has 2 values.
  - So we have potentials $\phi_1(x_1)$, $\phi_2(x_2)$, and $\phi_{12}(x_1, x_2)$ and want to have

$$w^T F(x) = w_{1,x_1} + w_{2,x_2} + w_{1,2,x_1,x_2}.$$

- With no parameter tieing and $x = \begin{bmatrix} 2 & 1 \end{bmatrix}$, our parameter vector and features are

$$w = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{1,2,1,1} \\ w_{1,2,1,2} \\ w_{1,2,2,1} \\ w_{1,2,2,2} \end{bmatrix}, \quad F(x) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

Example of Feature Function

- If we instead had Ising potentials (just measuring whether $x_1 = x_2$) we would have

$$w^T F(x) = w_{1,x_1} + w_{2,x_2} + w_{1,2,\text{same}},$$

  where $w_{1,2,\text{same}}$ is the parameter specifying how much we want $x_1 = x_2$.

- With no parameter tieing and $x = \begin{bmatrix} 2 & 1 \end{bmatrix}$, our parameter vector and features are

$$w = \begin{bmatrix} w_{1,1} \\ w_{1,2} \\ w_{2,1} \\ w_{2,2} \\ w_{1,2.\text{same}} \end{bmatrix}, \quad F(x) = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

Log-Linear UGM NLL and Gradient

- With log-linear parameterization of UGMs,

  $$\phi_j(s) = \exp(w_{j,s}), \quad \phi_{jk}(s,s') = \exp(w_{j,k,s,s'}), \quad \phi_{jkl}(s,s',s'') = \exp(w_{j,k,l,s,s',s''}).$$

  the likelihood of an example x given parameter w is given by

  $$p(x \mid w) = \frac{\exp\left(w^T F(x)\right)}{Z},$$

  and the feature functions $F(x)$ count the number of times we use each w_j.

- This leads to a convex NLL (first term is linear, second is a big log-sum-exp),

  $$-\log p(x \mid w) = -w^T F(x) + \log(Z),$$

- The gradient has a simple form (derivation in bonus)

  $$\nabla_w - \log p(x \mid w) = -F(x) + \mathbb{E}[F(x)],$$

  where expectation is over x values (inference problem with current w).

Computing Log-Linear Gradient as Inference

- For 1 example, gradient in log-linear UGM with respect to parameter w_j is

$$\nabla_{w_j} f(w) = -F_j(x) + \mathbb{E}[F_j(x)].$$

- Example of $\phi_{10}(3) = \exp(w_{10,3})$ (potential that feature 10 is in state 3).
  - Averaging over n examples, the gradient with no parameter tieing is given by

$$\nabla_{w_{10,3}} f(w) = -\underbrace{\frac{1}{n}\left[\sum_{i=1}^{n} I[x_{10}^i = 3]\right]}_{\text{frequency in data}} + \underbrace{p(x_{10} = 3 \mid w)}_{\text{model ``frequency''}}.$$

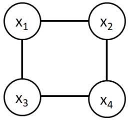
  - So if $\nabla_{w_{10,3}} f(w) = 0$, probabilities match frequencies in training data.
  - At MLE, you match the frequencies of all the potentials in the training data.
  - Typical training method: deterministic gradient descent methods (if have Z).
- But computing gradient requires inference (computing marginals like $p(x_{10} = 3)$).

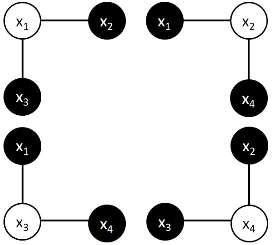
Approximate Learning: Pseudo-Likelihood

- Methods for approximate learning (when can't compute marginals efficiently):
    - Change the objective to an approximation that does not require marginals.
        - A popular approach is pseudo-likelihood (fast, convex, and crude):

$$p(x_1, x_2, \dots, x_d) \approx \prod_{j=1}^{d} p(x_j \mid x_{-j}) = \prod_{j=1}^{d} p(x_j \mid x_{\text{nei}(j)}),$$

which turns learning into d single-variable problems (similar to DAGs).





Approximate Learning: Marginal Approximations

- Methods for approximate learning (when can't compute marginals efficiently):
  - Approximate the marginals and use these within the gradient formula.
    1. Deterministic variational approximations of $\mathbb{E}[F(x)]$ (we will cover these later).
    2. Monte Carlo approximation of $\mathbb{E}[F_j(x)]$ given current parameters w:

$$\begin{aligned}\nabla f(w) &= -F(x) + \mathbb{E}[F(x)] \\ &\approx -F(x) + \underbrace{\frac{1}{t}\sum_{i=1}^{t} F(x^i)}_{\text{Monte Carlo approx}},\end{aligned}$$

based on samples from $p(x \mid w)$.

Younes Algorithm ("Persistent Contrastive Divergence")

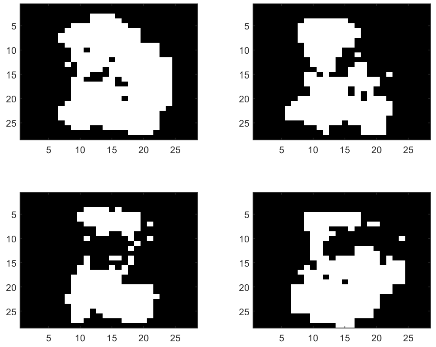
- Unfortunately, we typically cannot efficiently generate IID samples.
  - In cases where computing marginals is not efficient.

- Standard approach to use Monte Carlo approximation of gradient:
  1. Run Gibbs sampling for a long time to with current w^k.
     - To hopefully generate an IID sample x^k from $p(x \mid w^k)$.
  2. SGD Update based on this sample: $w^{k+1} = w^k + \alpha_k(F(x) + F(x^k))$.
- Younes algorithm (also known as "persistent contrastive divergence"):
  1. Run Gibbs sampling for a short time starting from x^{k-1} with current w^k.
     - Usually, you do 1 pass through the variables to generate new x^k.
  2. SGD Update based on this sample: $w^{k+1} = w^k + \alpha_k(F(x) + F(x^k))$.

- Younes algorithm works, even though gradient approximations are biased.
  - With much faster iterations than Monte Carlo with Gibbs sampling.

Pairwise UGM on MNIST Digits

- Samples from a lattice-structured pairwise UGM:



- Training: 100k stochastic gradient w/ Gibbs sampling steps with $\alpha_t = 0.01$.
- Samples are iteration 100k of Gibbs sampling with fixed w.
  - Bonus slides: structure learning in log-linear UGMs with L1-regularization.

Outline

“THE REVOLUTION WILL NOT BE SUPERVISED” PROMISES FACEBOOK’S YANN LECUN IN KICKOFF AI SEMINAR

POSTED MARCH 6TH, 2018

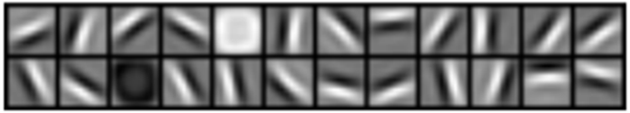
http://engineering.nyu.edu/news/2018/03/06/revolution-will-not-be-supervised-promises-facebooks-yann-lecun-kickoff-ai-seminar

Deep Density Estimation

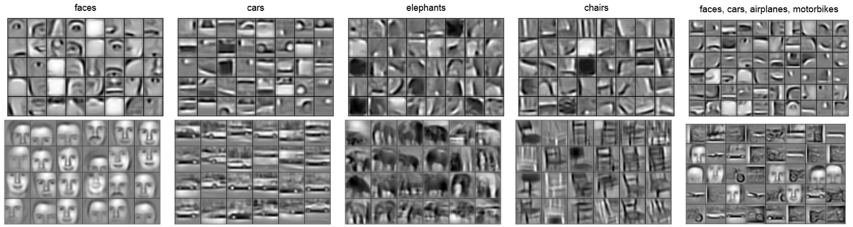
- In 340 we discussed deep learning methods for supervised learning.
- Does it make sense to talk about deep unsupervised learning?
- Standard argument:
    - Human learning seems to be mostly unsupervised.
    - Supervision gives limited feedback: bits in a class label vs. an image.
    - Could we learn unsupervised models with much less data?
- Deep belief networks started modern deep learning movement (2006).

Cool Pictures Motivation 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

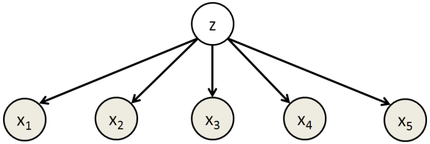
- Many classes use these particular images to motivate deep neural networks.
  - But they're not from a neural network: they're from a DAG model.

Mixture of Independent Models

- Recall the mixture of independent models:

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

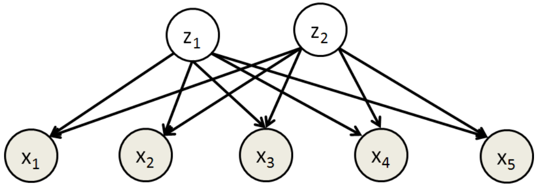
- Given z, each variable x_j comes from some "nice" distribution.



- This is enough to model *any* distribution.
  - Just need to know cluster of example x and distribution of x_j given z.
  - But not an efficient representation: number of cluster might need to be huge.
    - Need to learn each cluster independently (no "shared" information across clusters).

Latent DAG Model

- Consider the following model with binary z_1 and z_2:

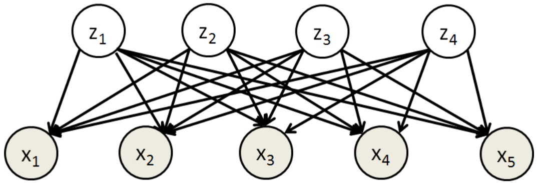


- Have we gained anything?
  - We have 4 clusters based on two hidden variables.
  - Each cluster shares a parent/part with 2 of the other clusters.

- 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.

Latent DAG Model

- Consider the following model:



- 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").
  - Usually assume $p(x_j \mid z_1, z_2, z_3, z_4)$ 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.
  - Unfortunately, somewhat hard to use:
    - Combinatorial "explaining away" between z_c value when conditioning on x.
    - Restricted Boltzmann Machines (RBMs) are a similar undirected model...

Summary

- Alpha-beta swaps and alpha exapnsions.
  - Powerful approximate decoding methods in "attractive" models.
- Log-linear parameterization can be used to learn UGMs:
  - Maximum likelihood is convex, but requires normalizing constant Z and inference.
- Approximate UGM learning:
  1. Change objective function: pseudolikelihood.
  2. Approximate marginals: Monte Carlo or variational methods.
- Multi-Cluster Mixture Model
  - Cluster is defined by values of a set of k binary variables.
  - Exponential number of clusters, but explaining away makes inference hard.
- Next time: the work that started the the modern deep learning movement.

Example: Ising Model of Rain Data

- E.g., for the rain data we could parameterize our node potentials using

$$\log(\phi_i(x_i)) = \begin{cases} w_1 & \text{no rain} \\ 0 & \text{rain} \end{cases}.$$

- Why do we only need 1 parameter?
  - Scaling $\phi_i(1)$ and $\phi(2)$ by constant doesn't change distribution.
- In general, we only need $(k-1)$ parameters for a k-state variable.
  - But if we're using regularization we may want to use k anyways (symmetry).

Example: Ising Model of Rain Data

- The Ising parameterization of edge potentials,

$$\log(\phi_{ij}(x_i, x_j)) = \begin{cases} w_2 & x_i = x_j \\ 0 & x_i \neq x_j \end{cases}.$$

- Applying gradient descent gives MLE of

$$w = \begin{bmatrix} 0.16 \\ 0.85 \end{bmatrix}, \quad \phi_i = \begin{bmatrix} \exp(w_1) \\ \exp(0) \end{bmatrix} = \begin{bmatrix} 1.17 \\ 1 \end{bmatrix}, \quad \phi_{ij} = \begin{bmatrix} \exp(w_2) & \exp(0) \\ \exp(0) & \exp(w_2) \end{bmatrix} = \begin{bmatrix} 2.34 & 1 \\ 1 & 2.34 \end{bmatrix},$$

  preference towards no rain, and adjacent days being the same.
- Average NLL of 16.8 vs. 19.0 for independent model.

Full Model of Rain Data

- We could alternately use fully expressive edge potentials

$$\log(\phi_{ij}(x_i, x_j)) = \begin{bmatrix} w_2 & w_3 \\ w_4 & w_5 \end{bmatrix},$$

  but these don't improve the likelihood much.

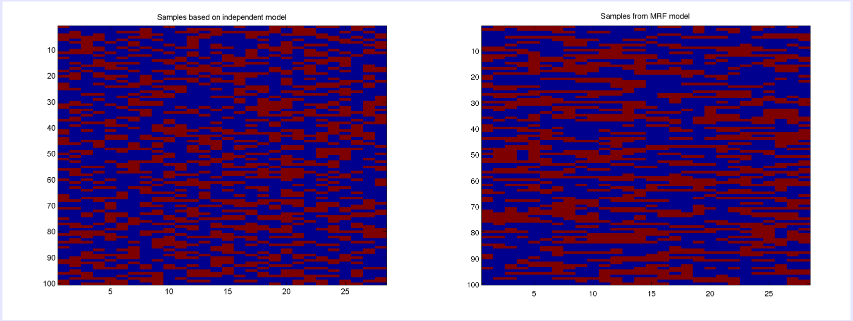
- We could fix one of these at 0 due to the normalization.
  - But we often don't do this when using regularization.

- We could also have special potentials for the boundaries.
  - Many language models are homogeneous, except for start/end of sentences.

Example: Ising Model of Rain Data

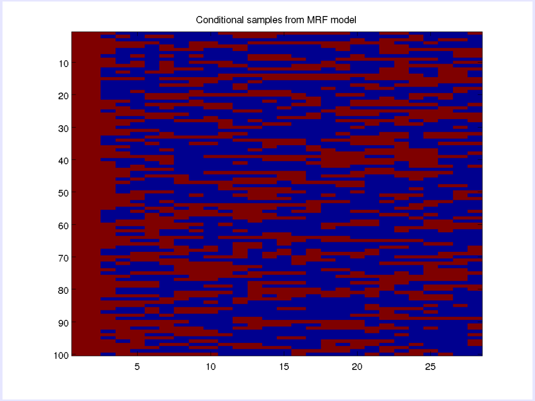
Independent model vs. chain-UGM model with tied nodes and Ising tied edges:

- For this dataset, using untied or general edges doesn't change likelihood much.



Example: Ising Model of Rain Data

Samples from Ising chain-UGM model if it rains on the first day:



UGM Training Objective Function

- With log-linear parameterization, NLL for IID training examples is

$$\begin{aligned} f(w) &= -\sum_{i=1}^{n} \log p(x^i \mid w) = -\sum_{i=1}^{n} \log\left(\frac{\exp(w^T F(x^i))}{Z(w)}\right) \\ &= -\sum_{i=1}^{n} w^T F(x^i) + \sum_{i=1}^{n} \log Z(w) \\ &= -w^T F(X) + n \log Z(w). \end{aligned}$$

  where the $F(X) = \sum_i F(x^i)$ are called the sufficient statistics of the dataset.
  - Given sufficient statistics $F(X)$, we can throw out the examples x^i.
    (only go through data once)
- Function $f(w)$ is convex (it's linear plus a big log-sum-exp function).
  - But notice that Z depends on w

.

Log-Linear UGM Gradient

- For 1 example x, we showed that NLL with log-linear parameterization is

$$f(w) = -w^T F(x) + \log Z(w).$$

- The partial derivative with respect to parameter w_j has a simple form

$$\begin{aligned}\nabla_{w_j} f(w) &= -F_j(x) + \sum_x \frac{\exp(w^T F(x))}{Z(w)} F_j(x) \\ &= -F_j(x) + \sum_x p(x \mid w) F_j(x) \\ &= -F_j(x) + \mathbb{E}[F_j(x)].\end{aligned}$$

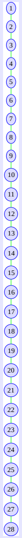
- Observe that derivative of $\log(Z)$ is expected value of feature.

Structure Learning in UGMs

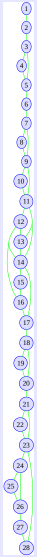
- Recall that in Ising UGMs, our edge potentials have the form
$$\phi_{ij}(x_i, x_j) = \exp(w_{ij}x_ix_j).$$
- If we set $w_{ij} = 0$, it sets $\phi_{ij}(x_i, x_j) = 1$ for all x_i and x_j.
    - Potential just "multiplies by 1", which is equivalent to removing the edge.
- L1-regularization of w_{ij} values performs structure learning in UGM.
- For general log-linear, each edge has multiple parameters $w_{i,j,s,s'}$.
    - In this case we can use "group L1-regularization" for structure learning.
        - Each group will be all parameters $w_{i,j,\cdot,\cdot}$ associated with an edge (i, j).

Structure Learning on Rain Data

Large λ (and optimal tree):

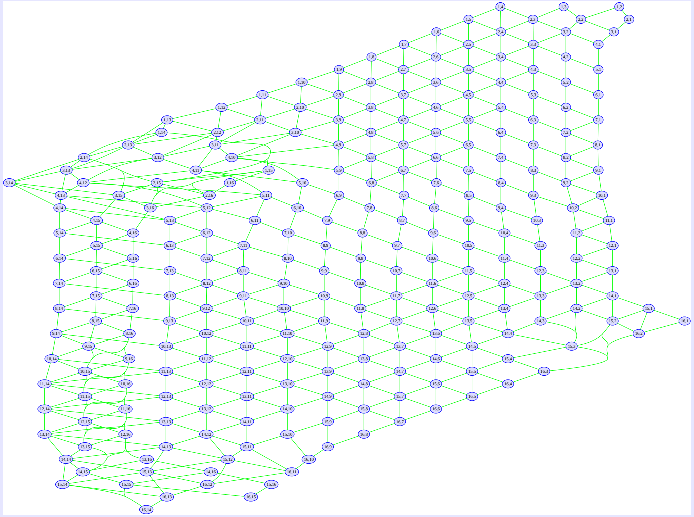


Small λ:



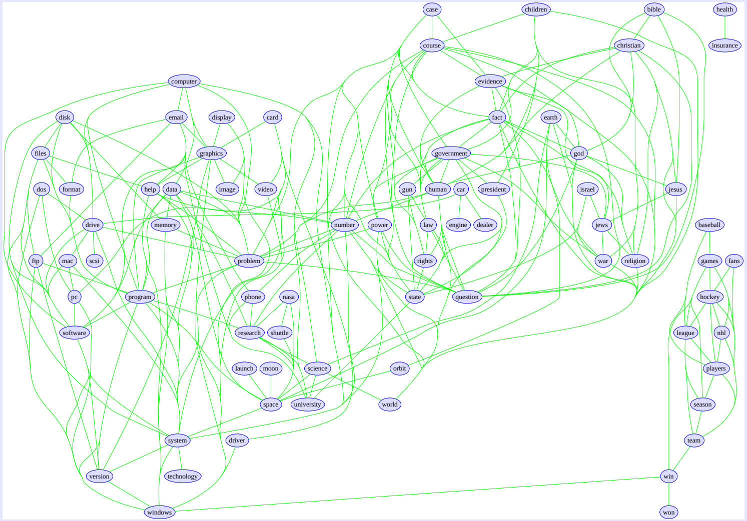
Structure Learning on USPS Digits

Structure learning of pairwise UGM with group-L1 on USPS digits:



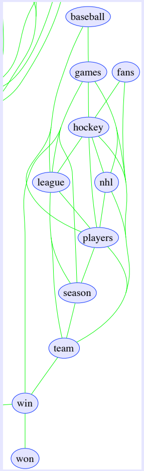
Structure Learning on News Words

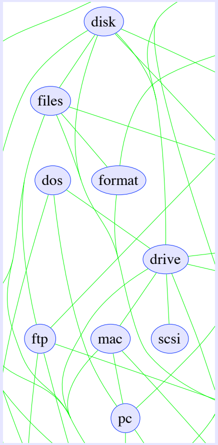
Group-L1 on newsgroups data:



Structure Learning on News Words

Group-L1 on newsgroups data:


baseball
games
fans
hockey
league
nhl
players
season
team
win
won


disk
files
dos
format
drive
ftp
mac
scsi
pc