CPSC 440: Advanced Machine Learning

Log-Linear Models

Mark Schmidt

University of British Columbia

Winter 2022

Last Time: Undirected Graphical Models

- We discussed undirected graphical models

$$p(x_1, x_2, \ldots, x_d) \propto \prod_{c \in \mathcal{C}} \phi_c(x_c),$$

which write joint distribution as product of non-negative potentials over subsets c.

- The most common variant is pairwise UGMs,

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

which includes Markov chains and multivariate Gaussians as special cases.

- In tree-structured graphs (no loops), common inference operations are $O(dk^2)$.
  - By generalizing the methods used for Markov chains.
  - But runtime is exponential in "treewidth" for graphs with loops.

Vancouver Rain Data: DAG vs. UGM

- We previously considered the "Vancouver Rain" dataset:

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	...
Month 1	0	0	0	1	1	0	0	1	1	
Month 2	1	0	0	0	0	0	1	0	0	
Month 3	1	1	1	1	1	1	1	1	1	
Month 4	1	1	1	1	0	0	1	1	1	
Month 5	0	0	0	0	1	1	0	0	0	
Month 6	0	1	1	0	0	0	0	1	1	

- We previously fit this with a Markov chain under the DAG factorization:

$$p(x_1, x_2, \dots, x_d) = p(x_1) \prod_{j=2}^{d} p(x_j \mid x_{j-1}),$$

where we used tabular potentials (so learning was counting).

Vancouver Rain Data: DAG vs. UGM

- Consider fitting a Markov chain under the UGM factorization:

$$p(x_1, x_2, \ldots, x_d) \propto \left(\prod_{j=1}^{d} \phi_j(x_j)\right)\left(\prod_{j=2}^{d} \phi_{j,j-1}(x_j, x_{j-1})\right).$$

- Consider the following UGM parameterization (for $x_j \in \{-1, +1\}$):

$$\phi_j(x_j) = \exp(w_j x_j), \quad \phi_{ij}(x_i, x_j) = \exp(v_{ij} x_i x_j),$$

where w_j is a node weight, v_{ij} is an edge weight, and we have used Ising edges.

  - We use the exponential function to make the potentials non-negative.
    - We call this a log-linear model: logarithms of potentials are linear.
  - Ising potentials can reflect how strongly neighbours are attracted/repulsed.
  - For the rain data, we would expect $v_{ij} > 0$ (adjacent days likely to have same value).
  - For the rain data, it makes sense to tie w_j across j and v_{ij} across (i, j) values.

Vancouver Rain Data: DAG vs. UGM

- Our log-linear model of the rain data under the Ising parameterization:

$$\begin{aligned} p(x_1, x_2, \ldots, x_d \mid w, v) &\propto \left(\prod_{j=1}^{d} \exp(w x_j) \right) \left(\prod_{j=2}^{d} \exp(v x_j x_{j-1}) \right) \\ &= \exp\left(\sum_{j=1}^{d} w x_j + \sum_{j=2}^{d} v x_j x_{j-1} \right) \\ &= \exp\left(w \sum_{j=1}^{d} x_j + v \sum_{j=2}^{d} x_j x_{j-1} \right) \\ &= \exp\left(\begin{bmatrix} w \\ v \end{bmatrix}^T \begin{bmatrix} \sum_{j=1}^{d} x_j \\ \sum_{j=2}^{d} x_j x_{j-1} \end{bmatrix} \right), \end{aligned}$$

which is an exponential family in canonical form.

- NLL will be convex in terms of w and v, derivative of NLL has simple form.
- If we did not tie parameters, we would have a statistic for each time.

Log-Linear UGM NLL and Gradient (Rain Data)

- Our convex NLL over n training examples is

$$f(w,v) = -w\sum_{i=1}^{n}\sum_{j=1}^{d} x_j^i - v\sum_{i=1}^{n}\sum_{j=2}^{d} x_j^i x_{j-1}^i + n\log Z(w,v),$$

  and we typically train log-linear models using gradient descent.

- The derivative with respect to w has simple form but requires inference,

$$\nabla_w f(w,v) = -\sum_{i=1}^{n}\sum_{j=1}^{d} x_j^i + n\sum_{j=1}^{d} p(x_j = 1 \mid w,v).$$

  where in Markov chains all marginals can be obtained from forward-backward.
  - The gradient with respect to v is similar, using pairwise marginals.
  - If you did the forward pass to compute $\log Z(w,v)$, autodif could do backward pass.

- At solution, we will that $p(x_j = 1)$ is frequency of this happening in data.
  - And $p(x_j = x_{j-1})$ matching frequency in data from the Ising edge parameter.

Log-Linear NLL and Gradient (General Case)

- We often write log-linear UGMs in an exponential family form

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

where the feature functions $F(x)$ count the number of times we use each w_j.
  - Examples of feature functions, and potentails for categoricals, in bonus.

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

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

- The gradient has a simple form but requires inference,

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

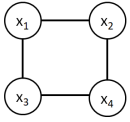
which is #P-hard in general graphs.
  - So we need to consider approximations when learning.

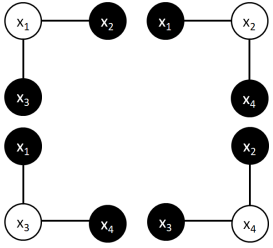
Approximate Learning: Pseudo-Likelihood

- A popular approximation to the NLL is pseudo-likelihood.
  - "Fast, convex, and crude."

- Pseudo-likelihood turns learning into d single-variables problem (similar to DAGs),

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





Approximate Learning: Marginal Approximations

- Another way to approximate the NLL is with approximate inference.
  1. Deterministic variational approximations of $\mathbb{E}[F(x)]$ (we will cover these later).
     - Approximate p by a simpler q, and compute expectation for q.
  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 $p(x \mid w)$.
     - Unfortunately, usually we also cannot generate IID samples efficiently.

Approximate Learning with MCMC Marginal Approximation

- An innefficient approach to using MCMC approximation of gradient:
  1. At iteration k, we want to sample from $p(x \mid w^k)$.
     - Start from some x^0, sample x^1, sample x^2, and so on.
     - Treat the last sample x^k from the Markov chain as an IID sample.
  2. Update the parameters using x^k to get an unbiased gradient approximation,

$$w^{k+1} = w^k + \alpha_k(F(x) - F(x^k)),$$

- If the Markov chain is run long enough, can show convergence using standard stochastic gradient descent arguments.
  - But have to run MCMC on each iteration of the SGD method.

Younes Algorithm ("Persistent Contrastive Divergence")

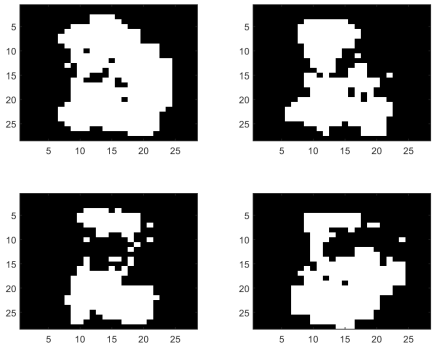
- Younes algorithm (also known as "persistent contrastive divergence"):
  1. At iteration k, we want to sample from $p(x \mid w^k)$.
     - Set $x^0 = x^{k-1}$, sample x^1, sample x^2, and so on.
     - Treat the last sample x^k from the Markov chain as an IID sample.
  2. Update the parameters using x^k to get an unbiased gradient approximation,

$$w^{k+1} = w^k + \alpha_k(F(x) - F(x^k)),$$

- In Younes algorithm, you do not need to run Markov chain to stationarity.
  - Usually you only run MCMC for 1 or a small number of iterations.
  - This gives a biased estimate, but is much faster than running MCMC to stationarity.
  - And with small-enough step-size, can show convergence.

Pairwise UGM on MNIST Digits

- Samples from a lattice-structured pairwise UGM trained on MNIST:



- Training: 100k stochastic gradient w/ Gibbs sampling steps with $\alpha_t = 0.01$.
- Samples are iteration 100k of Gibbs sampling with fixed w.

Outline

Motivation: Rain Data with Month Information

- Our Ising UGM model for the rain data with tied parameters,

$$p(y_1, y_2, \ldots, y_k \mid w, v) \propto \exp\left(\sum_{c=1}^{k} w y_c + \sum_{c=2}^{k} v y_c y_{c-1}\right),$$

  where I have switched the variable names from x_j to y_c (but model is same).

- First term will refelct that "not rain" is a more likely.
- Second term reflects that consecutive days are more likely to be the same.
  - This model is equivalent to a Markov chain model.
- But the model does not reflect that some months are less rainy.

- We can add features that reflect the month (or other information).
  - Multi-label supervised learning, but modeling dependence in labels y_c.
  - Adding fixed features to a UGM is also called a conditional random field (CRF).

Conditional Random Field (CRF) for Rain Data

- A CRF model of rain data, conditioned on 12 "one of k" month features x_j,

$$p(y_1, y_2, \ldots, y_k \mid x, w_0, w, v) \propto \exp\left(\sum_{c=1}^{k} w_0 y_c + \sum_{c=2}^{k} v y_c y_{c-1} + \sum_{c=1}^{k} y_c w^T x\right).$$

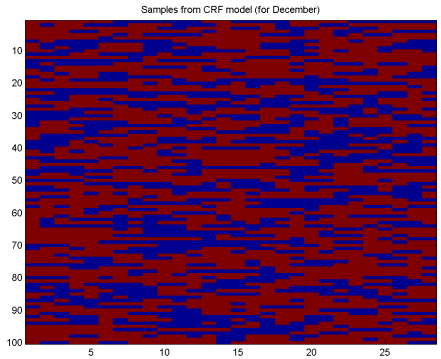
- The potentials in this model over the random variables y_c are:

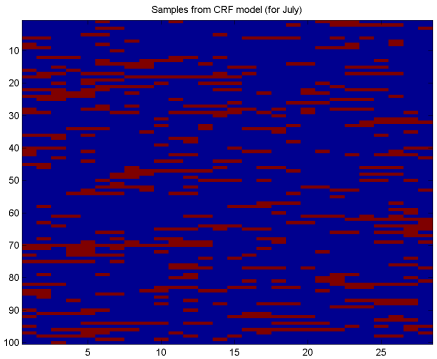
$$\phi_i(y_i) = \exp\left(w_0 y_i + y_i w^T x\right), \quad \phi_{ij}(y_i, y_j) = \exp(v y_i y_j).$$

- If we draw the UGM over y_c variables we get a chain structure.
  - So inference can be done using forward-backward.
  - And it's still log-linear so the NLL will be convex.
    - Gradient descent finds global optimum jointly with respect to w_0, w, and v.

Rain Data with Month Information

- Samples from CRF conditioned on x being December (left) and July (right):





- Conditional NLL is 16.21, compared to Markov chain which gets NLL 16.81.
  - Code for this and a variety of other UGM models is here:
    `https://www.cs.ubc.ca/~schmidtm/Software/UGM.html`

Conditional Random Fields (General Case)

- We often write the likelihood for general CRFs in the form

$$p(y \mid x, w) = \frac{1}{Z(x, w)} \exp(w^T F(x, y)),$$

for some parameters w and features $F(x, y)$.

- The NLL is convex and has the form

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

and the gradient can be written as

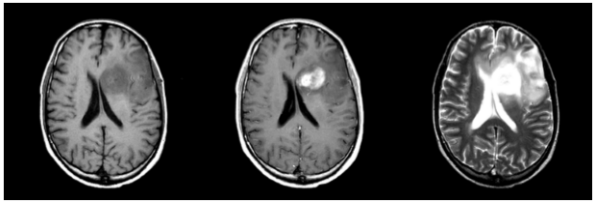
$$-\nabla \log p(y \mid x, w) = -F(x, y) + \mathbb{E}_{y \mid x, w}[F(x, y)],$$

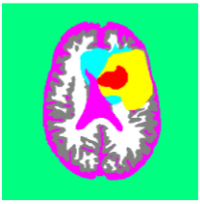
which requires inference for each value of x in training data.

  - For rain data need to do run forward-backward 12 times.
  - If each example has its own features, need to run it n times.
  - So it can make sense to use stochastic gradient if n is large.

Motivation: Automatic Brain Tumor Segmentation

- Task: identification of tumours in multi-modal MRI.





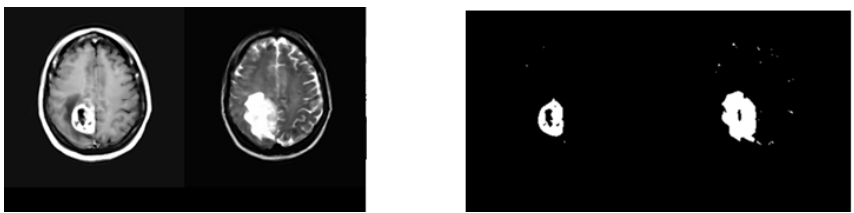
- Applications:
  - Radiation therapy target planning, quantifying treatment response.
  - Mining growth patterns, image-guided surgery.
- Challenges:
  - Variety of tumor appearances, similarity to normal tissue.
  - "You are never going to solve this problem".

Brain Tumour Segmentation with Label Dependencies

- After a lot pre-processing and feature engineering (convolutions, priors, etc.), final system used logistic regression to label each pixel as "tumour" or not.

$$p(y_c \mid x_c) = \frac{1}{1+\exp(-2y_c w^T x_c)} = \frac{\exp(y_c w^T x_c)}{\exp(w^T x_c)+\exp(-w^T x_c)}$$

- Gives a high "pixel-level" accuracy, but sometimes gives silly results:



- Classifying each pixel independently misses dependence in labels y^i:
  - We prefer neighbouring voxels to have the same value.

Brain Tumour Segmentation with Label Dependencies

- With independent logistic, conditional distribution over all labels in one image is

$$p(y_1, y_2, \ldots, y_k \mid x_1, x_2, \ldots, x_k) = \prod_{c=1}^{k} \frac{\exp(y_c w^T x_c)}{\exp(w^T x_c) + \exp(-w^T x_c)}$$
$$\propto \exp\left(\sum_{c=1}^{d} y_c w^T x_c\right),$$

  where here x_c is the feature vector for position c in the image.

- We can view this as a log-linear UGM with no edges,

$$\phi_c(y_c) = \exp(y_c w^T x_c),$$

  so given the x_c there is no dependence between the y_c.

Brain Tumour Segmentation with Label Dependencies

- Adding an Ising-like term to model dependencies between y_i gives

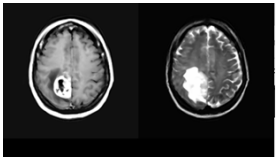
$$p(y_1, y_2, \ldots, y_k \mid x_1, x_2, \ldots, x_k) \propto \exp\left(\sum_{c=1}^{k} y_c w^T x_c + \sum_{(c,c') \in E} y_c y_{c'} v\right),$$

- Now we have the same "good" logistic regression model, but v controls how strongly we want neighbours to be the same.

- We can run gradient descent to jointly optimize w and v (convex NLL).
  - So we find the optimal joint logistic regression and Ising model.

Conditional Random Fields for Segmentation

- Recall the performance with the independent classifier:



- The pairwise CRF better modelled the "guilt by association":
  - Trained with pseudo-likelihood, constraining $v \geq 0$.
    - Decoding with "graph cuts" (see bonus lecture).

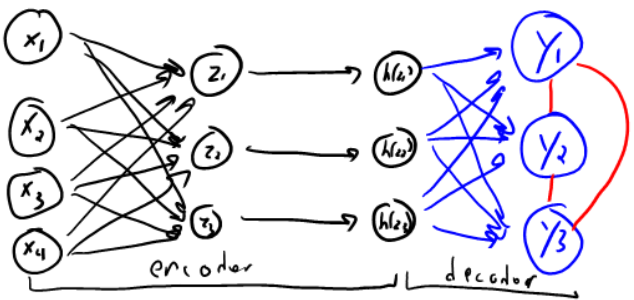
(We were using edge features $x_{cc'}$ too, see bonus (and different λ on edges).)

Combining Neural Networks and UGMs

- Instead of fixed features, you could use a neural network:

$$p(y \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c v^T h(W^3 h(W^2(W^1 x_c))) + \sum_{(c,c')\in E} u y_c y_{c'}\right).$$

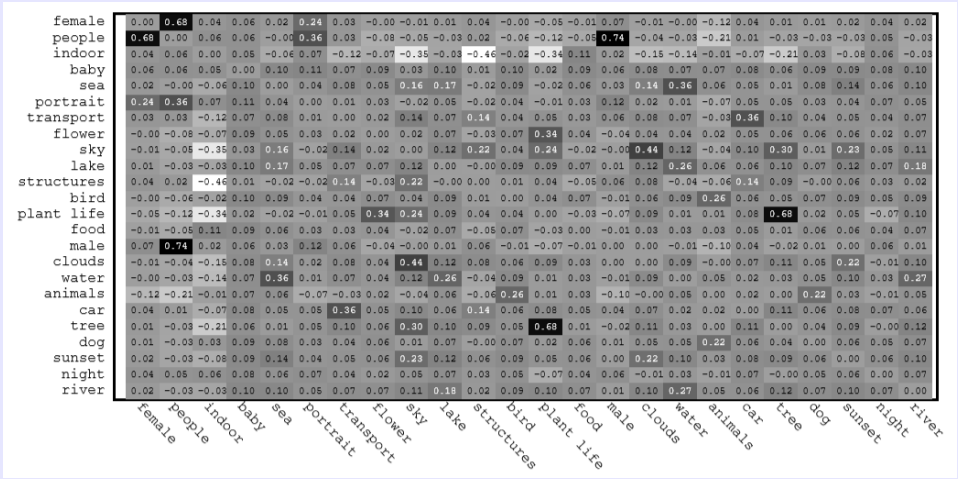
or you could have an encode-decode model spit out potentials of a UGM:



- These are sometimes called a conditional neural fields or deep structured model.

Multi-Label Classification

- Learned dependencies on a mult-label image classification dataset:



Automatic Differentiation (AD) vs. Inference

- Deep structured model gradient combines neural/Markov gradients:
  1. Forward pass through neural network to get $\hat{y}_c$ predictions.
  2. Forward message passing to compute normalizing constant.
  3. Backwards message passing to compute marginals.
  4. Backwards pass through neural network to get all gradients.
- You could skip the last two steps if you use automatic differentiation.

- But with approximate inference, AD may or may not work:
  - AD will work for iterative variational inference methods (which we'll cover later).
    - But it takes way more memory than needed (needs to store all iterations).
  - AD will not tend to work for Monte Carlo methods.
    - Cannot AD through sampling (need tricks like "common random numbers").

- Recent trend: run iterative variational method for a fixed number of iterations.
  - AD can give gradient of result after this fixed number of iterations.
  - "Train the inference you will use at test time".

Combining FCNs and CRFs

- DeepLab used a fully-connected pairwise UGM on top layer of FCN:

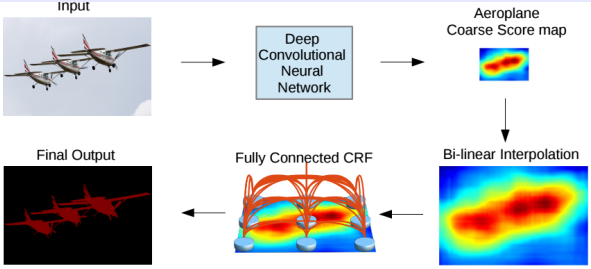


Fig. 1: Model Illustration. A Deep Convolutional Neural Network such as VGG-16 or ResNet-101 is employed in a fully convolutional fashion, using atrous convolution to reduce the degree of signal downsampling (from 32x down 8x). A bilinear interpolation stage enlarges the feature maps to the original image resolution. A fully connected CRF is then applied to refine the segmentation result and better capture the object boundaries.

https://arxiv.org/pdf/1606.00915.pdf

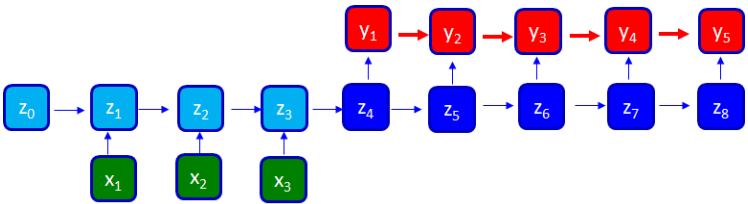
- But most recent version of the paper removed the UGM.

Do we need UGMs in Neural Networks?

- Recall that encode-decode hidden layers already capture label dependencies.
  - So do we need a UGM to explicitly model label dependencies in output layer?
- Factor 1: data size (big vs. small).
  - With a small dataset, it could be helpful to have direct dependencies in model.
  - With a large dataset, the hidden layers should reflect dependencies.
- Factor 2: how you evaluate the model (individual parts or full decoding).
  - If you measure "pixel level" or "word level" error, UGMs may not help.
  - If you measure "whole image" or "whole sentence" error, UGMs may help.
    - Because for example inference can discourage unlikely joint labelings.

Combining RNNs and Graphical Models

- An example where we use explicit label dependencies is language translation:

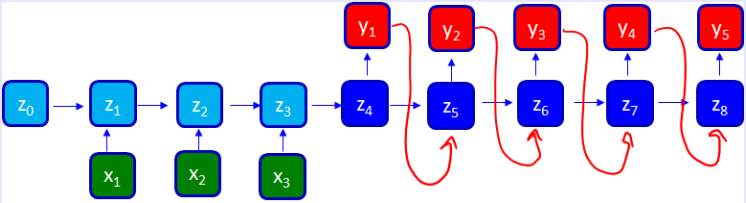


- Above model has usual deterministic edges, and DAG edges on labels.
- Can use Viterbi decoding to find best translation in this model.
  - Taking into account probability of seeing neighbouring words.

- But there is not much information in the DAG part of the model.
  - Only modeling dependencies between adjacent words.
- What we really want is to have the label we output affect the hidden state.
  - So that the encoding reflects previously-output words.

Combining RNNs and Graphical Models

- In order for the hidden states to depend on the output, we have this monstrosity:



- This can still be written as a Markov chain, but we cannot do Viterbi decoding.
  - Problem is that the hidden states in decoder become random variables.
  - So the state at each time has discrete and continuous parts (cannot be enumerated).

- To do decoding in this thing, we typically use beam search.
  - Heuristic algorithm that maintains "k best decodings up to time t".
    - Can be arbitrarily bad, but works if decoding is obvious as we go forward in time.
  - The type of edge and decoding strategy is also common with transformers.

Summary

- Log-linear parameterization can be used to learn UGMs:
  - Maximum likelihood is convex, but requires normalizing constant and inference.
- Approximate UGM learning:
  1. Change objective function: pseudolikelihood.
  2. Approximate marginals: Monte Carlo or variational methods.
     - Younes algorithm for using MCMC within SGD.
- Conditional random fields generalize logistic regression:
  - Multi-label model that explicitly models label dependencies.
- Combining CRFs with deep learning.
  - You can learn features and and the explicit label dependencies.

- Next time: a universal model for continuous density estimation.

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.

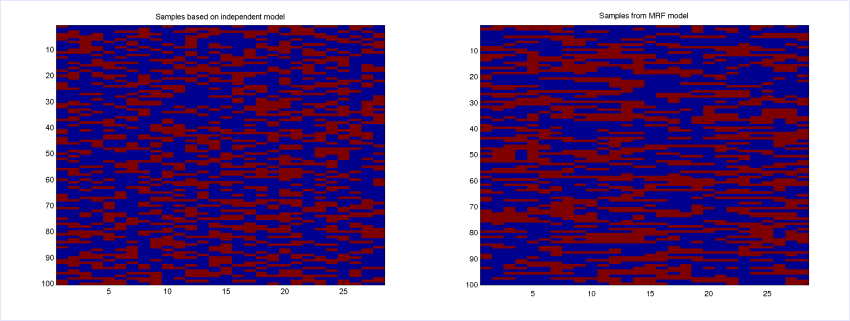
- We could fix one of these at 0 due to the normalization.
  - But we often don't do this when using regularization.

- 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

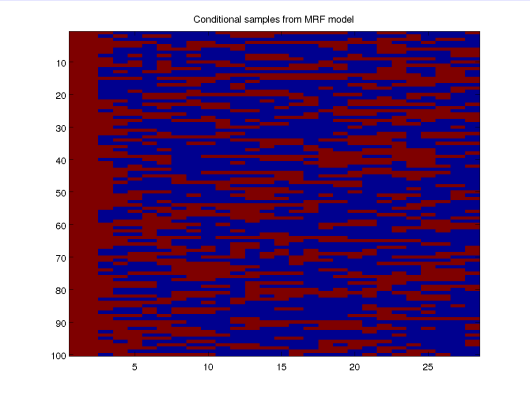
Independent model vs. chain-UGM model with tied nodes and Ising tied edges:

- 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:



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

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

UGM Training Objective Function

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

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

  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.

Brain Tumour Segmentation with Label Dependencies

- We got a bit more fancy and used edge features x^{ij},

$$p(y^1, y^2, \dots, y^d \mid x^1, x^2, \dots, x^d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} y^i w^T x^i + \sum_{(i,j) \in E} y^i y^j v^T x^{ij} \right).$$

- For example, we could use $x^{ij} = 1/(1 + |x^i - x^j|)$.
  - Encourages y_i and y_j to be more similar if x^i and x^j are more similar.

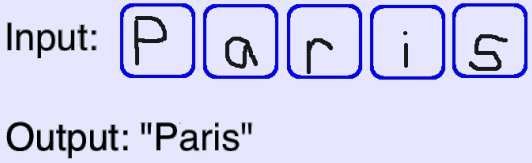
- This is a pairwise UGM with

$$\phi_i(y^i) = \exp(y^i w^T x^i), \quad \phi_{ij}(y^i, y^j) = \exp(y^i y^j v^T x^{ij}),$$

so it didn't make inference any more complicated.

Modeling OCR Dependencies

- What dependencies should we model for this problem?



  - $\phi(y_c, x_c)$: potential of individual letter given image.
  - $\phi(y_{c-1}, y_c)$: dependency between adjacent letters ('q-u').
  - $\phi(y_{c-1}, y_c, x_{c-1}, x_c)$: adjacent letters and image dependency.
  - $\phi_c(y_{c-1}, y_c)$: inhomogeneous dependency (French: 'e-r' ending).
  - $\phi_c(y_{c-2}, y_{c-1}, y_c)$: third-order and inhomogeneous (English: 'i-n-g' end).
  - $\phi(y \in \mathcal{D})$: is y in dictionary $\mathcal{D}$?

Tractability of Discriminative Models

- Features can be very complicated, since we just condition on the x_c, .
- Given the x_c, tractability depends on the conditional UGM on the y_c.
  - Inference tasks will be fast or slow, depending on the y_c graph.
- Besides "low treewidth", some other cases where exact computation is possible:
  - Semi-Markov chains (allow dependence on time you spend in a state).
    - For example, in rain data the seasons will be approximately 3 months.
  - Context-free grammars (allows potentials on recursively-nested parts of sequence).
  - Sum-product networks (restrict potentials to allow exact computation).
  - "Dictionary" feature is non-Markov, but exact computation still easy.
- We can alternately use our previous approximations:
  1. Pseudo-likelihood (what we used).
  2. Monte Carlo approximate inference (eventually better but probably much slower).
  3. Variational approximate inference (fast, quality varies).

Structure Learning in UGMs

- Recall that in Ising UGMs, our edge potentials have the form

$$\phi_{ij}(x_i, x_j) = \exp(w_{ij} x_i x_j).$$

- 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.

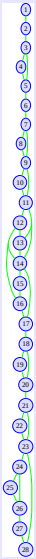
- L1-regularization of w_{ij} values performs structure learning in UGM.

- 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

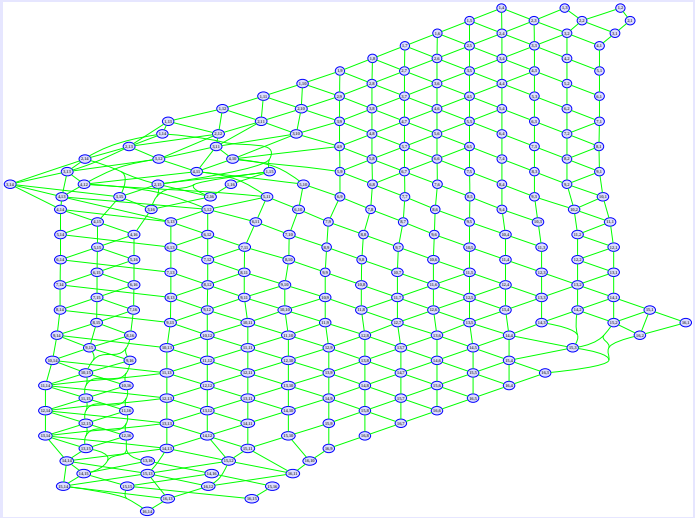
Large λ (and optimal tree):

Small λ:



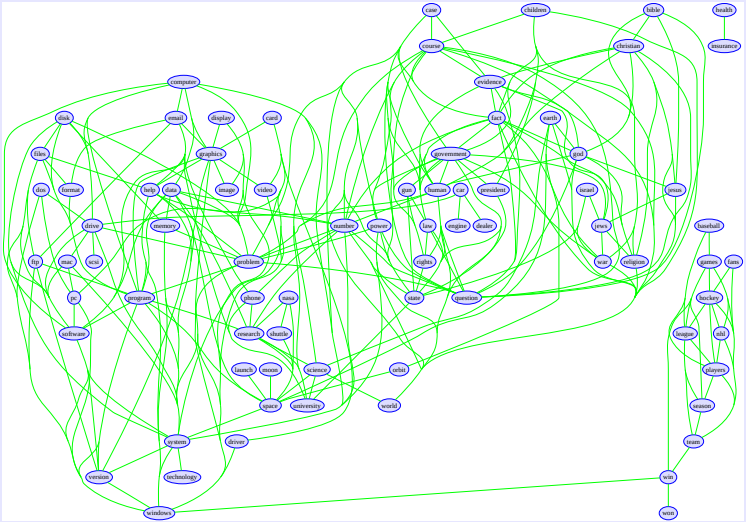
Structure Learning on USPS Digits

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



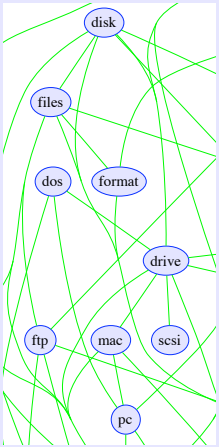
Structure Learning on News Words

Group-L1 on newsgroups data:



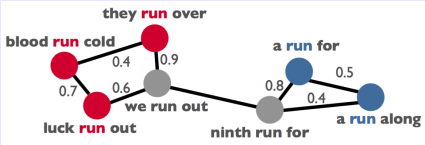
Structure Learning on News Words

Group-L1 on newsgroups data:



Posterior Regularization

- In some cases it might make sense to use posterior regularization:
  - Regularize the probabilities in the resulting model.
- Consider an NLP labeling task where
  - You have a small amount of labeled sentences.
  - You have a huge amount of unlabeled sentences.
- Maximize labeled likelihood, plus total-variation penalty on $p(y_c \mid x, w)$ values.
  - Give high regularization weights to words appearing in same trigrams:



http://jgillenw.com/conll2013-talk.pdf

- Useful for "out of vocabulary" words (words that don't appear in labeled data).
  - Has been replaced in recent by continuous word representations like word2vec.