CPSC 540: Machine Learning

Conditional Random Fields

Mark Schmidt

University of British Columbia

Winter 2020

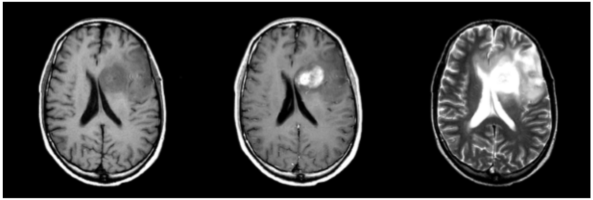
3 Classes of Structured Prediction Methods

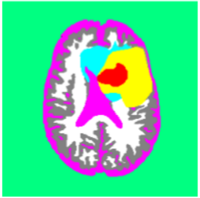
3 main approaches to structured prediction (predicting object y given features x):

1. Generative models use $p(y \mid x) \propto p(y, x)$ as in naive Bayes.
   - Turns structured prediction into density estimation.
     - But remember how hard it was just to model images of digits?
     - We have to model features and solve supervised learning problem.
2. Discriminative models directly fit $p(y \mid x)$ as in logistic regression (next topic).
   - View structured prediction as conditional density estimation.
     - Just focuses on modeling y given x, not trying to model features x.
     - Lets you use complicated features x that make the task easier.
3. Discriminant functions just try to map from x to y as in SVMs.
   - Now you don't even need to worry about calibrated probabilities.

Motivation: Automatic Brain Tumor Segmentation

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





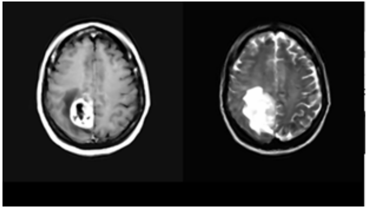
- Applications:
  - Radiation therapy target planning, quantifying treatment response.
  - Mining growth patterns, image-guided surgery.
- Challenges:
  - Variety of tumor appearances, similarity to normal tissue.
  - "You are never going to solve this problem".

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

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

  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

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

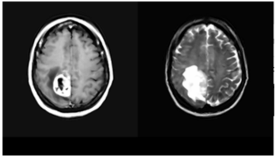
- Now we have the same "good" logistic regression model, but v controls how strongly we want neighbours to be the same.

- Note that we're going to jointly learn w and v.
  - We'll find the optimal joint logistic regression and Ising model.

- When we model conditional of y given x as a UGM, we call it a conditional random field (CRF).
  - Key advantadge of this (discriminative) approach:
    - Don't need to model features x as in "generative" models.
    - We saw with MNIST digits that modeling images is hard.

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. Added constraint $v \geq 0$ to use graph cut decoding.

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

- CRFs are like logistic regression (no modeling x) vs naive Bayes (modeling x).
  - $p(y \mid x)$ (discriminative) vs. $p(y, x)$ (generative).

Conditional Random Fields

- The brain CRF can be written as a conditional log-linear models,

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

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

- The NLL is convex and has the form

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

  and the gradient can be written as

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

- Unlike before, we now have a $Z(x)$ and set of expectations for each x.
  - Train using gradient methods like quasi-Newton, SG, or SAG.

Rain Data without Month Information

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

$$p(y_1, y_2, \dots, y_k) \propto \exp\left(\sum_{c=1}^{k} y_c \omega + \sum_{c=2}^{k} y_c y_{c-1} \nu\right).$$

- First term reflects that "not rain" is more likely.
- Second term reflects that consecutive days are more likely to be the same.
  - This model is equivalent to a Markov chain model.

- We could condition on month to model "some months are less rainy".

Rain Data with Month Information using CRFs

- Discriminative appraoch: fit a CRF model conditioned on month x,

$$p(y_1, y_2, \dots, y_k \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c\omega + \sum_{c=2}^{d} y_c y_{c-1}\nu + \sum_{c=1}^{k}\sum_{j=1}^{12} y_c x_j v_j\right).$$

- The conditional UGM given x has a chain-structure

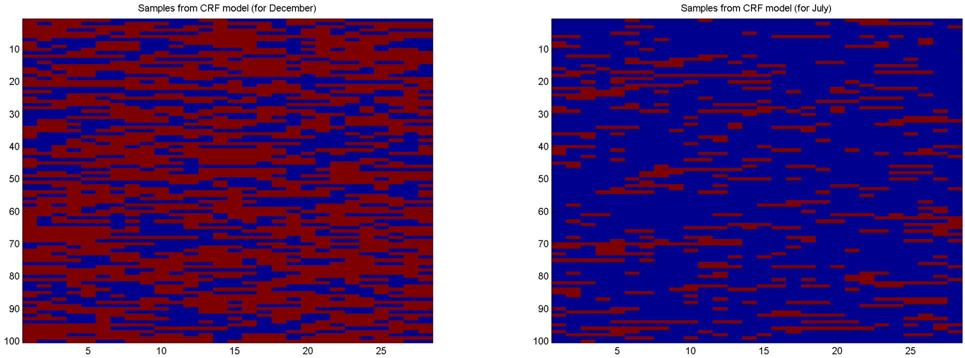
$$\phi_i(y_i) = \exp\left(y_i\omega + \sum_{j=1}^{12} y_i x_j v_j\right), \quad \phi_{ij}(y_i, y_j) = \exp(y_i y_j \nu),$$

so inference can be done using forward-backward.

  - And it's log-linear so the NLL will be convex.

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.
  - Better than mixture of 10 Markov chains (EM training), which gets 16.53.
    - Probably due to finding global minima when fitting CRF.

Rain Data with Month Information using CRFs

- A CRF model conditioned on month x,

$$p(y_1, y_2, \ldots, y_k \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{c=1}^{k} y_c \omega + \sum_{c=2}^{d} y_c y_{c-1} \nu + \sum_{c=1}^{k} \sum_{j=1}^{12} y_c x_j v_j\right).$$

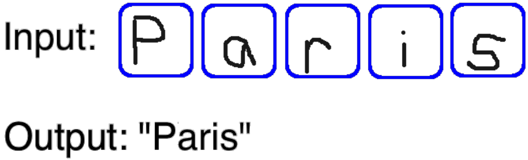
- Comparing this to other approaches:
  - Generative: model $p(y_1, y_2, \ldots, y_k, x)$.
    - Have to model distribution of x, and inference is more expensive (not a chain).
    - Also uses known clusters.
    - Learning is still convex.
  - Mixtre/Boltzmann: add latent variables z that might learn month information.
    - Have to model distribution of z, inference is more expensive (not a chain).
    - Doesn't use known clusters so needs more data.
    - But might learn a better clustering if months aren't great clusters.
    - Learning is non-convex due to sum over z values.

Outline

1. Conditional Random Fields
2. Beyond Basic CRFs

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.
- $\phi_c(y_{i-1}, y_c)$: inhomogeneous dependency (French: 'e-r' ending).
- $\phi_c(y_{c-2}, y_{c-1}, y^i)$: 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/decoding 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).
  - 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).

CRF "Product of Marginals" Objective

- In CRFs we typically optimize the likelihood, $p(y \mid x, w)$.
  - This focuses on getting the joint likelihood of the sequence y right.

- What if we are interested in getting the "parts" y_c right?
  - In sequence labeling, your error is "number of positions you got wrong" in sequence.
  - As opposed to "did you get the whole sequence right?"

- In this setting, it could make more sense to optimize the product of marginals:

$$\prod_{c=1}^{k} p(y_c \mid x, w) = \prod_{c=1}^{k} \sum_{\{y' \mid y'_c = y_c\}} p(y' \mid x, w).$$

- Non-convex, but probably a better objective.
- If you know how to do inference, this paper shows how to get gradients:
  - `https://people.cs.umass.edu/~domke/papers/2010nips.pdf`

Learning for Structured Prediction (Big Picture)

3 types of classifiers discussed in CPSC 340/540:

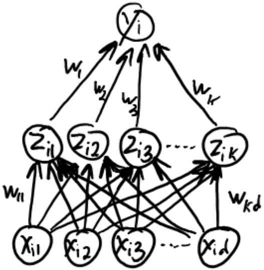
Model	"Classic ML"	Structured Prediction
Generative model $p(y, x)$	Naive Bayes, GDA	UGM (or "MRF")
Discriminative model $p(y \mid x)$	Logistic regression	CRF
Discriminant function $y = f(x)$	SVM	Structured SVM

- Discriminative models don't need to model x.
  - Don't need "naive Bayes" or Gaussian assumptions.

- Discriminant functions don't even worry about probabilities.
  - Based on decoding, which is different than inference in structured case.
  - Useful when inference is hard but decoding is easy.
  - Examples include "attractive" graphical models, matching problems, and ranking.
  - I put my material on structured SVMs here:
    - `https://www.cs.ubc.ca/~schmidtm/Courses/540-W19/L28.5.pdf`

Feedforward Neural Networks

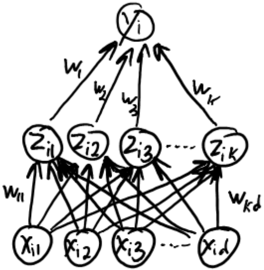
- In 340 we discussed feedforward neural networks for supervised learning.
- With 1 hidden layer the classic model has this structure:



- Motivation:
  - For some problems it's hard to find good features.
  - This learns features z that are good for particular supervised learning problem.

Neural Networks as DAG Models

- It's a DAG model but there is an important difference with our previous models:
  - The latent variables z_c are deterministic functions of the x_j.



- Makes inference given x trivial: if you observe all x_j you also observe all z_c.
  - In this case y is the only random variable.

Deep Learning for Structured Prediction (Big Picture)

- How is deep learning being used for structured prediction?
  - Discriminative approaches are most popular.

- Typically you will send x through a neural network to get representation z, then:
  1. Perform inference on $p(y \mid z)$ (backpropagate using exact/approximate marginals).
     - Neural network learns features, CRF "on top" models dependencies in y_c.
  2. Run m approximate inference steps on $p(y \mid z)$, backpropagate through these steps.
     - "Learn to use the inference you will be using" (usually with variational inference).
  3. Just model each $p(y_c \mid z)$ (treat labels as independent given representation).
     - Assume that structure is already captured in neural network goo (no inference).

- Current trend: less dependence on inference and more on learning representation.
  - "Just use an RNN rather than thinking about stochastic grammars."
  - We're improving a lot at learning features, less so for inference.
  - This trend may or may not reverse in the future...

Summary

- 3 types of structured prediction:
    - Generative models, discriminative models, discriminant functions.
- Conditional random fields generalize logistic regression:
    - Discriminative model allowing dependencies between labels.
    - Log-linear parameterization again leads to convexity.
    - But requires inference in graphical model.
- Reducing the reliance on inference is a current trend in the field.
    - Rely on neural network to learn clusters and dependencies.
- Next time: our (overdue) visit to the world of deep learning.

Brain Tumour Segmentation with Label Dependencies

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

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

- 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

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

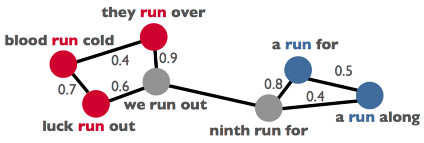
so it didn't make inference any more complicated.

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).