CPSC 540: Machine Learning

Log-Linear Models, Conditional Random Fields

Mark Schmidt

University of British Columbia

Winter 2017

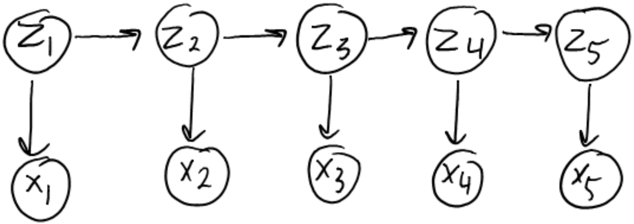
Admin

- Assignment 4:
    - Due Monday, 1 late day for Wednesday, 2 for the following Monday.
- Tuesday office hours from 2:30-3:30 (except March 21 and April 4).
- Interested in TAing CPSC 340 in the summer?
    - Contact Mike Gelbart.

Last Time: Hidden Markov Models

- We discussed hidden Markov models as more-flexible time-series model,

$$p(x, z) = p(z_1)\left(\prod_{j=2}^{d} p(z_j | z_{j-1})\right)\prod_{j=1}^{d} p(x_j | z_j).$$

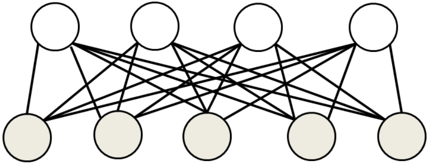


- Widely-used for sequence and time-series data.
  - Inference is easy because it's a tree, learning is normally done with EM.
  - Hidden latent dynamics can capture longer-term dependencies.

Last Time: Restricted Boltzmann Machines

- We discussed restricted Boltzmann machines as mix of clustering/latent-factors,

$$p(x,h) = \frac{1}{Z}\left(\prod_{i=1}^{d}\phi_i(x_i)\right)\left(\prod_{j=1}^{k}\phi_j(h_j)\right)\left(\prod_{i=1}^{d}\prod_{j=1}^{k}\phi_{ij}(x_i,h_j)\right).$$



- Bipartite structure allows block Gibbs sampling:
  - Conditional UGM removes observed nodes.
- Ingredient for training deep belief networks and deep Boltzmann machines.

Outline

Structured Prediction with Undirected Graphical Models

- Consider a pairwise UGM with no hidden variables,

$$p(x) = \frac{1}{Z}\left(\prod_{i=1}^{d}\phi_i(x_i)\right)\left(\prod_{(i,j)\in E}\phi_{ij}(x_i,x_j)\right).$$

- Previously we focused on inference in UGMs:
  - We've discussed decoding, inference, and sampling.
  - We've discussed [block-]coordinate approximate inference.

- We've also discussed a variety of UGM structures:
  - Lattice structures, hidden Markov models, Boltzmann machines.

- Today: learning the potential functions ϕ.

Maximum Likelihood Formulation

- With IID training x^i, MAP estimate for parmeters w solves

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

where we've assumed a Gaussian prior.

- But how should the non-negative ϕ be related to w?

- Naive parameterization:

$$\phi_i(x_i) = w_{i,x_i}, \quad \phi_{ij}(x_i, x_j) = w_{i,j,x_i,x_j}.$$

subject to $w \geq 0$.
  - Not convex, and assumes no parameter tieing.

Log-Linear Parameterization of UGMs

- To enforce non-negativity we'll exponentiate

$$\phi_i(x_i) = \exp(w_m),$$

  for some m.
- This is also called a log-linear parameterization,

$$\log \phi_i(x_i) = w_m.$$

- The NLL is convex under this parameterization.
  - Normally, exponentiating to get non-negativity introduces local minima.
- To allow parameter tieing, we'll make m map potentials to elements of w.

Log-Linear Parameterization of UGMs

- So our log-linear parameterization has the form

$$\log \phi_i(x_i) = w_{m(i,x_i)}, \quad \log \phi_{ij}(x_i, x_j) = w_{m(i,j,x_i,x_j)}.$$

  where m maps from potentials to parameters.

- Parameter tieing can be done with choice of m:
  - If $m(i, x_i) = x_i$ for all i, each node has same potentials.
    (parameters are tied)
  - Could make nodes have different potentials by mapping $\phi_i(x_i)$ to different parameters.
  - We could have groups: E.g., weekdays vs. weekends, or boundary.
  - We'll use the convention that $m(i, x_i) = 0$ means that $\phi_i(x_i) = 1$.
  - Similar logic holds for edge potentials.

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,

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

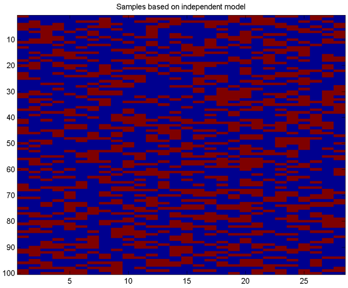
- Applying gradient descent gives MLE of

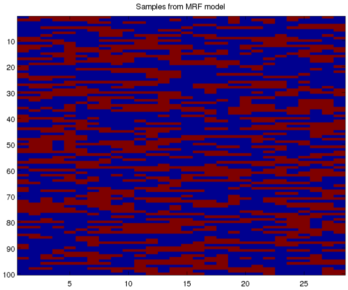
$$w = \begin{bmatrix} 0.16 \\ 0.85 \end{bmatrix}, \quad \phi_i = \begin{bmatrix} \exp(w_1) \\ \exp(0) \end{bmatrix} = \begin{bmatrix} 1.17 \\ 1 \end{bmatrix}, \quad \phi_{ij} = \begin{bmatrix} \exp(w_2) & \exp(0) \\ \exp(0) & \exp(w_2) \end{bmatrix} = \begin{bmatrix} 2.34 & 1 \\ 1 & 2.34 \end{bmatrix},$$

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

Example: Ising Model of Rain Data

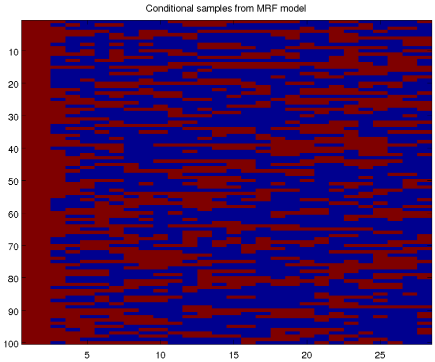
Independent model vs. Ising chain-UGM model:





Example: Ising Model of Rain Data

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



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.

Energy Function and Log-Linear Parameterization

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

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

and $E(x) = -\log \tilde{p}(x)$ is called the energy function.

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

$$\begin{aligned}
-E(X) &= \log\left(\left(\prod_i \exp(w_{m(i,x_i)})\right)\left(\prod_{(i,j)\in E} \exp(w_{m(i,j,x_i,x_j)})\right)\right) \\
&= \log\left(\exp\left(\sum_i w_{m(i,x_i)} + \sum_{(i,j)\in E} w_{m(i,j,x_i,x_j)}\right)\right) \\
&= \sum_i w_{m(i,x_i)} + \sum_{(i,j)\in E} w_{m(i,j,x_i,x_j)}.
\end{aligned}$$

Feature Vector Representation

- By appropriately indexing things (bonus slide) we can write

$$-E(x) = w^T F(x),$$

or

$$p(x) \propto p(w^T F(x)),$$

for a particular feature function $F(x)$:
  - Element j of $F(X)$ counts the number of times we use w_j.

- For the 2-parameter rain data model we have:

$$F(x) = \begin{bmatrix} \text{number of times it rained} \\ \text{number of times adjacent days were the same} \end{bmatrix}.$$

UGM Training Objective Function

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

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

  where $F(X) = \frac{1}{n}\sum_i F(x^i)$ are the sufficient statistics of the dataset.
  - Given sufficient statistics $F(X)$, can throw out examples x^i.

    (only go through data once)
- Function $f(w)$ is convex (it's linear plus a big log-sum-exp function).
  - But it requires $\log Z(w)$.

Optimization with UGMs

- We just showed that NLL with log-linear parameterization is

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

  and the gradient with respect to parameter j has a simple form

$$\begin{aligned}\nabla_j f(w) &= -F_j(X) + \sum_{x'} \frac{\exp(w^T F(x'))}{Z(w)} F_j(x') \\ &= -F_j(X) + \sum_{x'} p(x') F_j(x') \\ &= -F_j(X) + \mathbb{E}_{x'}[F_j(x')].\end{aligned}$$

- Derivative of $\log(Z)$ is expected value of feature.
- Optimality ($\nabla_j f(w) = 0$) means sufficient statistics match in model and data.
  - Frequency of w_j appearing is the same in the data and the model.
- But computing gradient requires inference.

Approximate Learning

Strategies when inference is not tractable:

1. Use approximate inference:
   - Variational methods.
   - Monte Carlo methods.
     - Younes: alternate between Gibbs sampling and stochastic gradient, "persistent contrastive divergence".

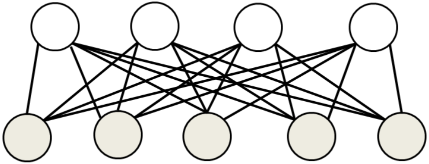
2. Change the objective function:
   - Pseudo-likelihood (fast, convex, and crude):

$$\log p(x_1, x_2, \ldots, x_d) \approx \sum_{j=1}^{d} \log p(x_j | x_{-j}),$$

     transforms learning into logistic regression on each part.
   - SSVMs: generalization of SVMs that only requires decoding (next time).

Learning UGMs with Hidden Variables

- For RBMs we have hidden variables:



- With hidden variables the observed likelihood has the form

$$
\begin{aligned}
p(x) &= \sum_z p(x,z) = \sum_z \frac{\tilde{p}(x,z)}{Z} \\
&= \frac{\sum_z \tilde{p}(x,z)}{Z} = \frac{Z(x)}{Z},
\end{aligned}
$$

where $Z(x)$ is the partition function of the conditional UGM.

Learning UGMs with Hidden Variables

- This gives an observed NLL of the form

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

- The second term is convex but the first term is non-convex.
- We typically use MCMC/variational on each term, rather than EM.
  - In RBMs, $Z(x)$ is cheap due to independent of z given x.

- Binary RBMs usually use a log-linear parameterization:

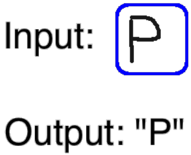
$$-E(x,h) = \sum_{i=1}^{d} x_i w_i + \sum_{j=1}^{k} h_j v_j + \sum_{i=1}^{d}\sum_{j=1}^{k} x_i w_{ij} h_j,$$

  for parameters w_i, v_j, and w_{ij}.
  - Recall that we have $p(x,h) \propto \exp(-E(x,h))$.

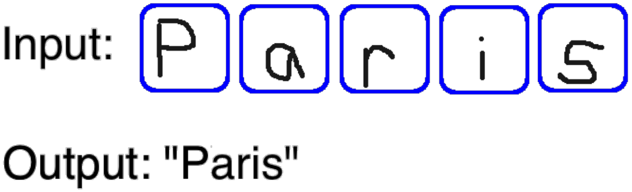
Outline

Motivation: Structured Prediction

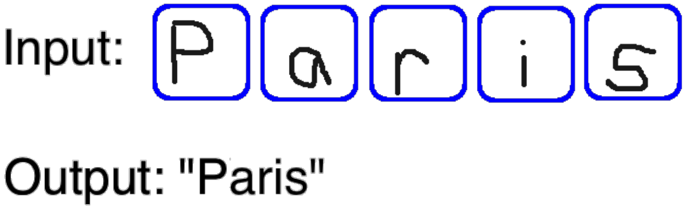
Classical supervised learning focuses on predicting single discrete/continuous label:



Structured prediction allows general objects as labels:



“Classic” ML for Structured Prediction



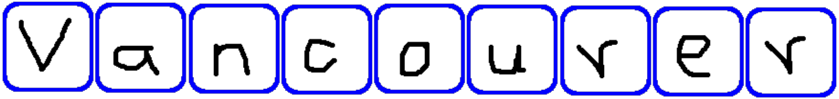
Two ways to formulate as “classic” machine learning:

1. Treat each word as a different class label.
   - Problem: there are too many possible words.
   - You will never recognize new words.
2. Predict each letter individually:
   - Works if you are really good at predicting individual letters.
   - But some tasks don’t have a natural decomposition.
   - Ignores dependencies between letters.

Motivation: Structured Prediction

- What letter is this?

- What are these letters?

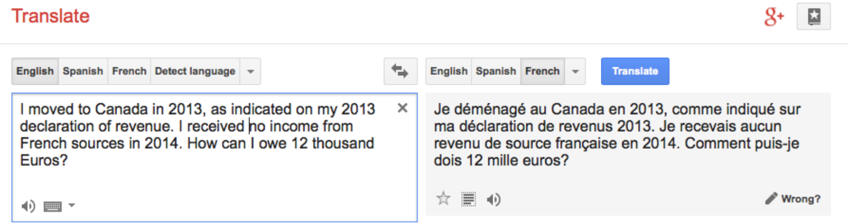


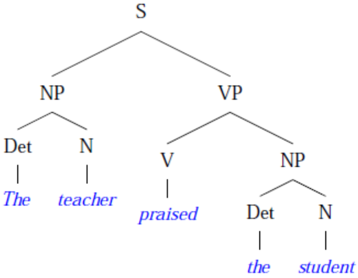
- Predict each letter using "classic" ML and neighbouring images?
  - Turn this into a standard supervised learning problem?
- Good or bad depending on goal:
  - Good if you want to predict individual letters.
  - Bad if goal is to predict entire word.

Supervised Learning vs. Structured Prediction

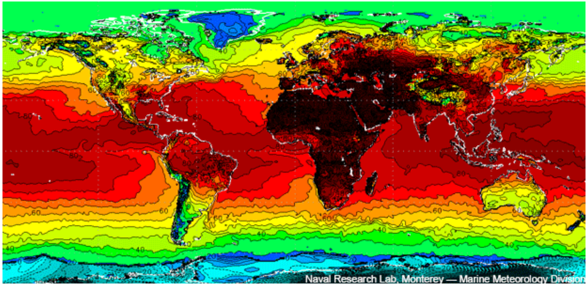
- In 340 we focused a lot on “classic” supervised learning:
  - Model $p(y|x)$ where y is a single discrete/continuous variable.
- In 540 we've focused a lot on density estimation:
  - Model $p(x)$ where x is a vector or general object.
- Structured prediction is the logical combination:
  - Model $p(y|x)$ where y is a vector or general object.

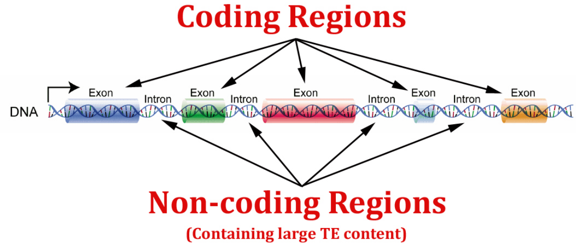
Examples of Structured Prediction



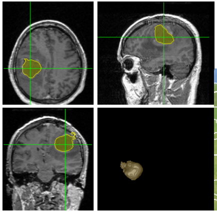


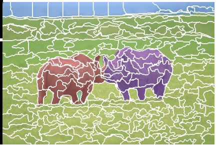
Examples of Structured Prediction

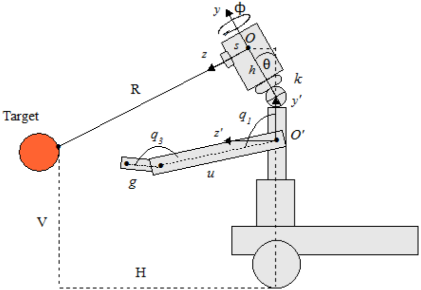




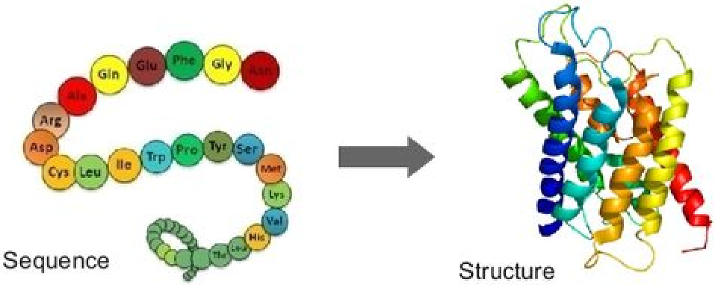
Examples of Structured Prediction







Examples of Structured Prediction

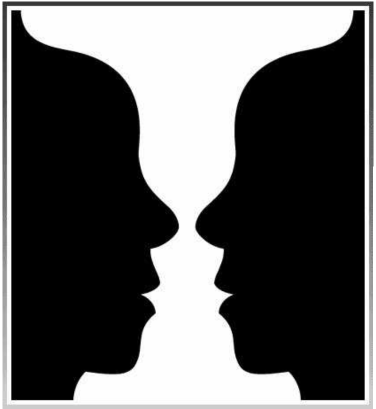


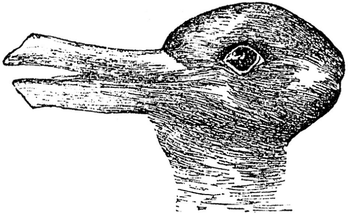
In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell-Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

Does the brain do structured prediction?

Gestalt effect: "whole is other than the sum of the parts".





What do you see?
By shifting perspective you might see an old woman or a young woman.

3 Classes of Structured Prediction Methods

3 main approaches to structured prediction:

1. Generative models use $p(y|x) \propto p(y, x)$ 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 $p(y|x)$ as in logistic regression.
   - View structured prediction as conditional density estimation.
     - Just focuses on modeling y given x, not trying to modle 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.

We'll jump to discriminative models, since we've covered density estimation.

Outline

Conditional Random Fields (CRFs)

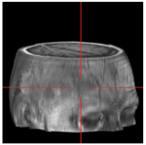
- We can do conditional density estimation with any density estimator:
  - Conditional mixture of Bernoulli, conditional Markov chains, conditional DAGs, etc.

- But the most common approach is conditional random fields (CRFs).
  - Generalization of logistic regression based on UGMs.
  - Extremely widely-used in natural language processing.
  - Now being combined with deep learning for vision (next week).

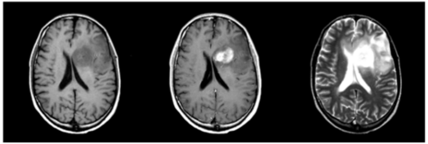
- I believe CRFs are second-most cited ML paper of the 2000s:
  1. Latent Dirichlet Allocation (last week of class).
  2. Conditional random fields.
  3. Deep learning.

Motivation: Automatic Brain Tumor Segmentation

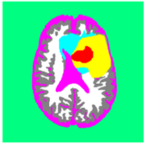
- Task: identification of tumours in multi-modal MRI.



Input:



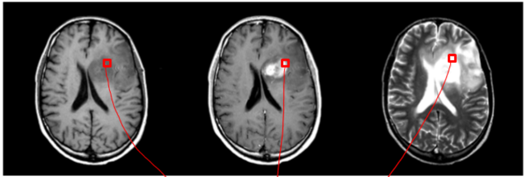
Output:



- 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".

Naive Approach: Voxel-Level Classifier

- We could treat classifying a voxel as supervised learning:



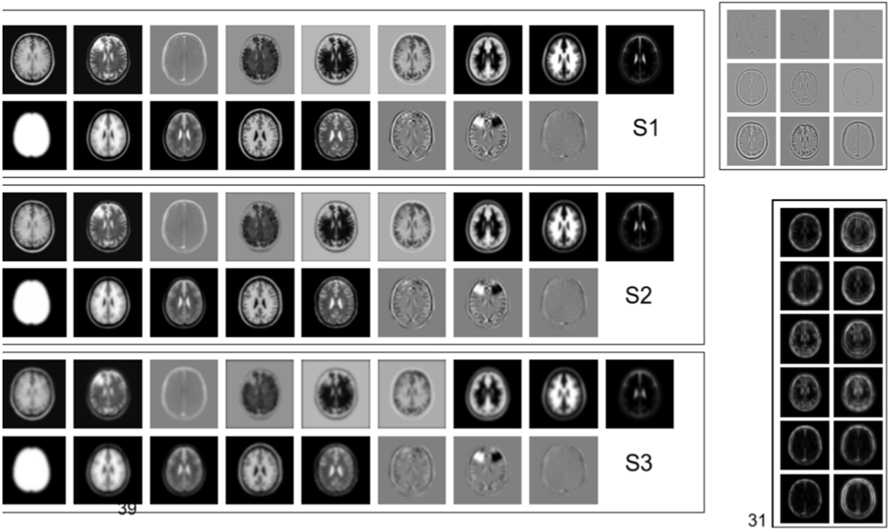
$x^i = \langle 98, 187, 246 \rangle$ $\quad$ $y^i =$ "tumor"

- "Learn" model that predicts y^i given x^i.
  - Given the model, we can classify new voxels.
- Advantage: we can appy machine learning, and ML is cool.
- Disadvantage: it doesn't work at all.

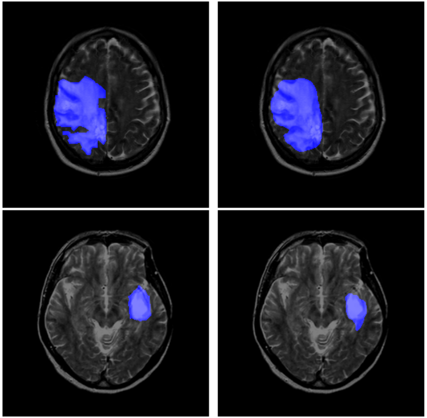
Fixed the Naive Approach

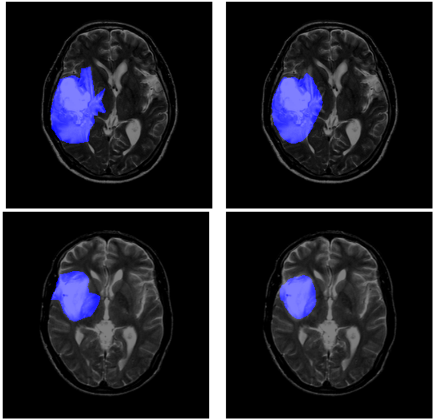
- Challenges:
  - Intensities are not standardized within or across images.
  - Location matters.
  - Context matters (significant intensity overlap between normal/abnormal).
- Partial solutions:
  - Pre-processing to to normalize intensities.
  - Alignment to standard coordinate system to model location.
  - Use convolutions to incorporate neighbourhood information.

Final Feature Set



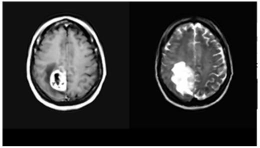
Performance of Final System





Challenges and Research Directions

- Final system used linear classifier, and typically worked well.
- But several ML challenges arose:
  1. Time: 14 hours to train logistic regression on 10 images.
     - Lead to quasi-Newton, stochastic gradient, and SAG work.
  2. Overfitting: using all features hurt, so we used manual feature selection.
     - Lead to regularization, L1-regularization, and structured sparsity work.
  3. Relaxation: post-processing by filtering and "hole-filling" of labels.
     - Lead to conditional random fields, shape priors, and structure learning work.



Multi-Class Logistic Regression: View 1

- Recall that multi-class logistic regression makes decisions using

$$\hat{y} = \underset{y \in \{1,2,\dots,k\}}{\text{argmax}} \; w_y^T F(x).$$

- Here $F(x)$ are features and we have a vector w_y for each class y.

- Normally we fit w_y using regularized maximum likelihood assuming

$$p(y|x, w_1, w_2, \dots, w_k) \propto \exp(w_y^T F(x)).$$

- This softmax probability yields a differentiable and convex NLL.

Multi-Class Logistic Regression: View 2

- Recall that multi-class logistic regression makes decisions using

$$\hat{y} = \operatorname*{argmax}_{y \in \{1,2,\dots,k\}} w_y^T F(x).$$

- Claim: can be written using a single w and features of x and y,

$$\hat{y} = \operatorname*{argmax}_{y \in \{1,2,\dots,k\}} w^T F(x,y).$$

- To do this, we can ues the construction

$$w = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_k \end{bmatrix}, \quad F(x,1) = \begin{bmatrix} F(x) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad F(x,2) = \begin{bmatrix} 0 \\ F(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

which gives $w^T F(x,y) = w_y^T F(x)$.

Multi-Class Logistic Regression: View 2

- So multi-class logistic regression with new notation uses

$$\hat{y} = \underset{y \in \{1,2,\ldots,k\}}{\text{argmax}} \; w^T F(x,y).$$

- And usual softmax probabilities give

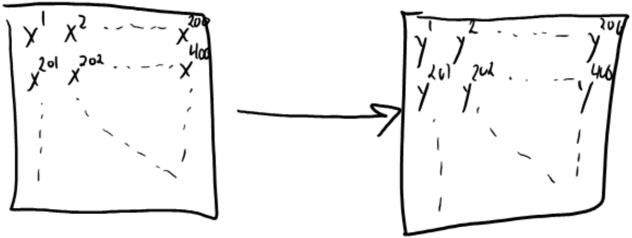
$$p(y|x,w) \propto \exp(w^T F(x,y)).$$

- View 2 gives extra flexibility in defining features:
  - For example, we might have different features for class 1 and 2:

$$F(x,1) = \begin{bmatrix} F(x) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad F(x,2) = \begin{bmatrix} 0 \\ G(x) \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Multi-Class Logistic Regression for Segmentation

- In brain tumour example, each x^i is the features for voxel i:
  - Softmax model gives $p(y^i|x^i, w)$ for any label y^i of voxel i.
- But we want to label the whole image:

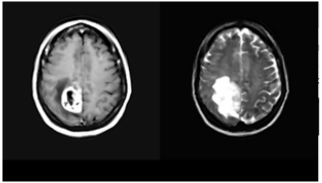


- Probability of full-image labeling Y given image X with independent model is

$$p(Y|X, w) = \prod_{i=1}^{n} p(y^i|x^i, w).$$

Conditional Random Fields

- Unfortunately, independent model gives silly results:



- This model of $p(Y|X,w)$ misses the guilt by association:
  - Neighbouring voxels are likely to receive the same values.

- The key ingredients of conditional random fields (CRFs):
  - Use softmax with features of entire image and labelling $F(X,Y)$:
  - We can model dependencies using features that depend on multiple y^i.

Conditional Random Fields

- Interpretation of independent model as a special case of CRF:

$$p(Y|X,w) = \prod_{i=1}^{n} p(y^i|x^i,w) \propto \prod_{i=1}^{n} \exp(w^T F(x^i,y^i))$$
$$= \exp\left(\sum_{i=1}^{n} w^T F(x^i,y^i)\right)$$
$$= \exp(W^T F(X,Y)),$$

where we're using

$$W = \begin{bmatrix} w \\ w \\ w \\ \vdots \\ w \end{bmatrix}, \quad F(X,Y) = \begin{bmatrix} F(x^1,y^1) \\ F(x^2,y^2) \\ F(x^3,y^3) \\ \vdots \\ F(x^n,y^n) \end{bmatrix}.$$

Conditional Random Fields

- Interpretation of independent model as a special case of CRF:

$$p(Y|X,w) = \prod_{i=1}^{n} p(y^i|x^i,w) \propto \prod_{i=1}^{n} \exp(w^T F(X,y^i))$$
$$= \exp\left(\sum_{i=1}^{n} w^T F(X,y^i)\right)$$
$$= \exp(W^T F(X,Y)),$$

where we're using

$$W = \begin{bmatrix} w \\ w \\ w \\ \vdots \\ w \end{bmatrix}, \quad F(X,Y) = \begin{bmatrix} F(X,y^1) \\ F(X,y^2) \\ F(X,y^3) \\ \vdots \\ F(X,y^n) \end{bmatrix}.$$

- Since we always condition on X, features F can depend on any part of X.

Conditional Random Fields

- Example of modeling dependencies between neighbours as a CRF:

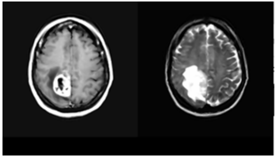
$$p(Y|X,w) = \exp(W^T F(X,Y)),$$

$$W = \begin{bmatrix} w \\ w \\ w \\ \vdots \\ w \\ v \\ v \\ \vdots \\ v \end{bmatrix}, \quad F(X,Y) = \begin{bmatrix} F(X,y^1) \\ F(X,y^2) \\ F(X,y^3) \\ \vdots \\ F(X,y^n) \\ F(X,y^1,y^2) \\ F(X,y^2,y^3) \\ \vdots \\ F(X,y^{n-1},y^n) \end{bmatrix}.$$

- Use features $F(X,y^i,y^j)$ of the dependency between y^i and y^j (with weights v).

Conditional Random Fields for Segmentation

- Recall the performance with the independent classifier:
  - Features of the form $F(X, y^i)$.



- Consider a CRF that also has pairwise features:
  - Features $F(X, y^i, y^j)$ for all (i, j) corresponding to adjacent voxels.
  - Models "guilt by association":

Conditional Random Fields as Graphical Models

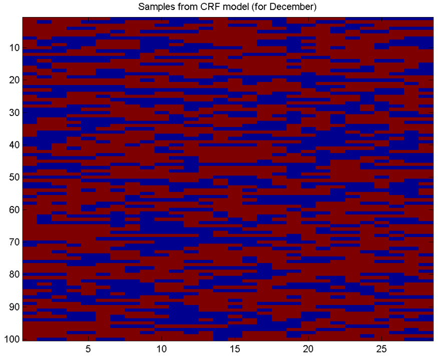
- Seems great: we can now model dependencies in the labels.
  - Why not model threeway interactions with $F(X, y^i, y^j, y^k)$?
  - How about adding things like shape priors $F(X, Y_r)$ for some region r?
- Challenge is that inference and decoding become hard.
- We can view CRFs as undirected graphical models,

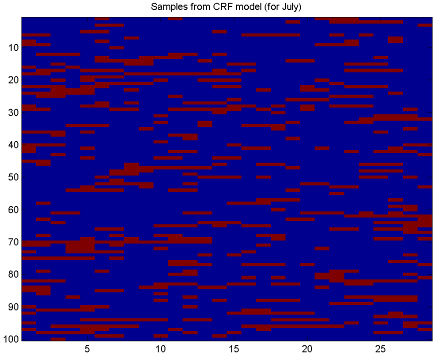
$$p(Y|X, w) \propto \prod_{c \in \mathcal{C}} \phi_c(Y_c),$$

- We have potential $\phi_c(Y_c)$ if Y_c appear together in one or more features $F(X, Y_c)$.
- For complicated graphs, we need approximate inference/training.
  - We used pseudo-likelihood for training and ICM for decoding.
  - ICM was later replaced by graph cuts, since we want adjacent pixels to be similar.

Rain Demo with Month Data

- Let's just add an explicit month variable to the rain data:
  - Fit a CRF of p(rain | month).
  - Use 12 binary indicator features giving month.
  - NLL goes from 16.8 to 16.2.
- Samples of rain data conditioned on December and July:





Summary

- Log-linear parameterization can be used to learn UGMs:
  - Maximum likelihood is convex, but requires normalizing constant Z.
- Structured prediction is supervised learning with a complicated y^i.
  - 3 flavours are generative models, discriminative models, and discriminant functions.
- Conditional random fields generalize logistic regression:
  - Discriminative model allowing dependencies between labels.
  - But requires inference in graphical model.

Next time: generalizing SVMs to structured prediction.

Bonus Slide: Feature Representation of Log-Linear UGMs

- Consider this identity

$$w_{m(i,x_i)} = \sum_f w_f \mathcal{I}[m(i,x_i) = f],$$

- Use this identity to write any log-linear energy in a simple form

$$\begin{aligned}
-E(X) &= \sum_i w_{m(i,x_i)} + \sum_{(i,j)\in E} w_{m(i,j,x_i,x_j)} \\
&= \sum_i \sum_f w_f \mathcal{I}[m(i,x_i) = f] + \sum_{(i,j)\in E} \sum_f w_f \mathcal{I}[m(i,j,x_i,x_j) = f] \\
&= \sum_f w_f \left(\sum_i \mathcal{I}[m(i,x_i) = f] + \sum_{(i,j)\in E} \mathcal{I}[m(i,j,x_i,x_j) = f] \right) \\
&= w^T F(X)
\end{aligned}$$