CPSC 540: Machine Learning

Structure Learning, Structured SVMs

Mark Schmidt

University of British Columbia

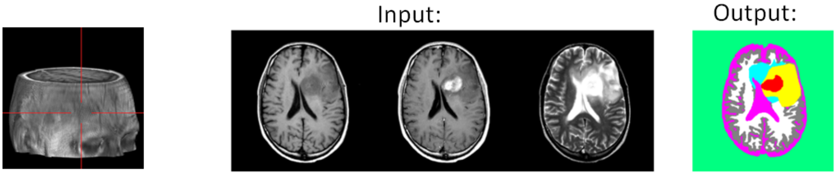
Winter 2017

Admin

- Assignment 4:
  - Due today, 1 late day for Wednesday, 2 for the following Monday.
- No office hours tomorrow.
- Project proposals: "no news is good news".
- Assignment 5 and project/final descriptions are coming soon.

Last Time: Structured Prediction

- We discussed structured prediction:
  - Supervised learning where output y is a general object.
- For example, automatic brain tumour segmentation:



- We want to label all pixels and model depeendencies between pixels.

- We could formulate this as a density estimation problem of modeling $p(x, y)$.
  - Here y is the labeling of the entire image.
  - But features x may be complicated.

- CRFs generalize logistic regression and directly model $p(y|x)$.

Outline

Ising Models

- The Ising model for binary x_i is defined by

$$p(x_1, x_2, \dots, x_d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} x_i w_i + \sum_{(i,j) \in E} x_i x_j w_{ij}\right).$$

- Consider using $x_i \in \{-1, 1\}$:
  - If $w_i > 0$ it encourages $x_i = 1$.
  - If $w_{ij} > 0$ it encourages neighbours i and j to have the same value.
    - E.g., neighbouring pixels in the image receive the same label ("attractive" model)

- This model is a special case of a pairwise UGM with

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

General Pairwise UGM

- For general discrete x_i a generalization is

$$p(x_1, x_2, \ldots, x_d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} w_{i,x_i} + \sum_{(i,j) \in E} w_{i,j,x_i,x_j}\right),$$

which can represent any "positive" pairwise UGM (meaning $p(x) > 0$ for all x).

- Interpretation of weights for this UGM:
  - If $w_{i,1} > w_{i,2}$ then we prefer $x_i = 1$ to $x_i = 2$.
  - If $w_{i,j,1,1} > w_{i,j,2,2}$ then we prefer $(x_i = 1, x_j = 1)$ to $(x_i = 2, x_j = 2)$.

- As before, we can use parameter tieing:
  - We could use the same w_{i,x_i} for all positions i.
  - Ising model corresponds to tieing of the w_{i,j,x_i,x_j}.

Log-Linear Models

- These models are special cases of log-linear models which have the form

$$p(x|w) = \frac{1}{Z}\exp\left(w^T F(x)\right),$$

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

- The log-linear NLL is convex and has the form

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

and the gradient can be written as

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

- So if the gradient is zero, the empirical features match the and expected features.

Training Log-Linear Models

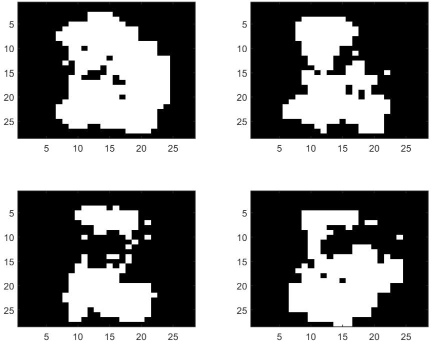
- The term $\mathbb{E}[F(x)]$ in the gradient may be hard to compute.
  - In a pairwise UGM, it depends on univariate and pairwise marginals.
- It's common to use variational or Monte Carlo estimates of these marginals.
  - In RBMs, we alternate between block Gibbs sampling and stochastic gradient.
- Or a crude approximation is pseudo-likelihood,

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

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

Pairwise UGM on MNIST Digits

- Samples from a lattice-structured UGM:



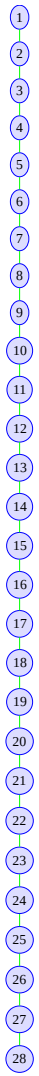
- Training: 100k stochastic gradient w/ Gibbs sampling steps with $\alpha_t = 0.01$.
- Samples are iteration 100k of Gibbs sampling with fixed w.

Structure Learning in UGMs

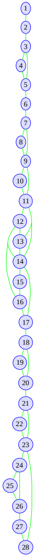
- The problem of choosing the graph is called structure learning.
  - Generalizes feature selection: we want to find all relationships between variables.
- Finding optimal tree is a minimum spanning tree problem.
  - "Chow-Liu algorithm": based on pairiwse mutual information
- NP-hard for non-tree DAG and UGMs.
  - For DAGs, we usually do a greedy search through space of acyclic graphs.
- For Ising UGMs, we can use L1-regularization of w_{ij} values.
  - If $w_{ij} = 0$, then we remove dependency.
- For discrete UGMs, we can use group L1-regularization of w_{i,j,x_i,x_j} values.
  - If $w_{i,j,x_i,x_j} = 0$ for all x_i and x_j, we remove dependency.

Structure Learning on Rain Data

Large λ (and optimal tree):

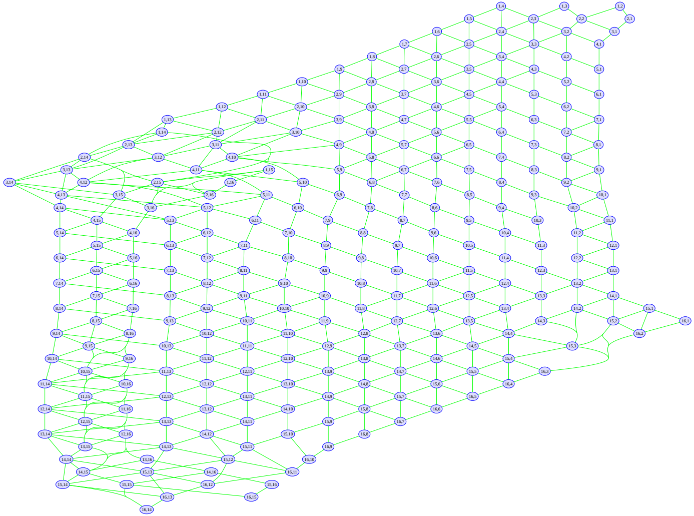


Small λ:



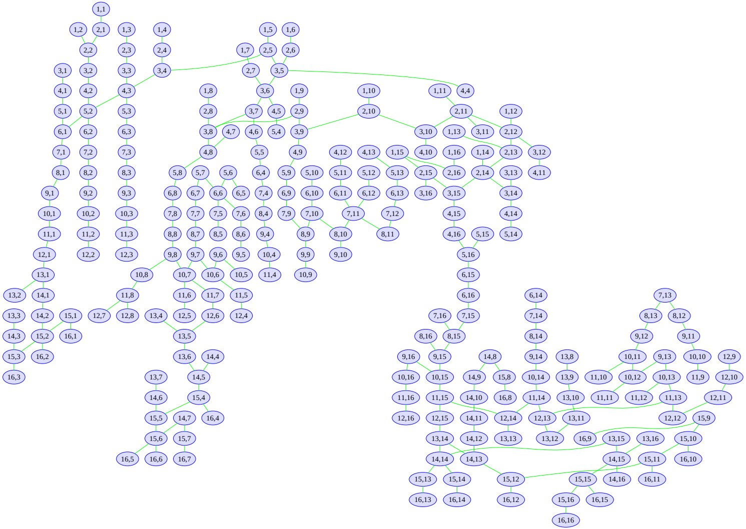
Structure Learning on USPS Digits

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



Structure Learning on USPS Digits

Optimal tree on USPS digits:



20 Newsgroups Data

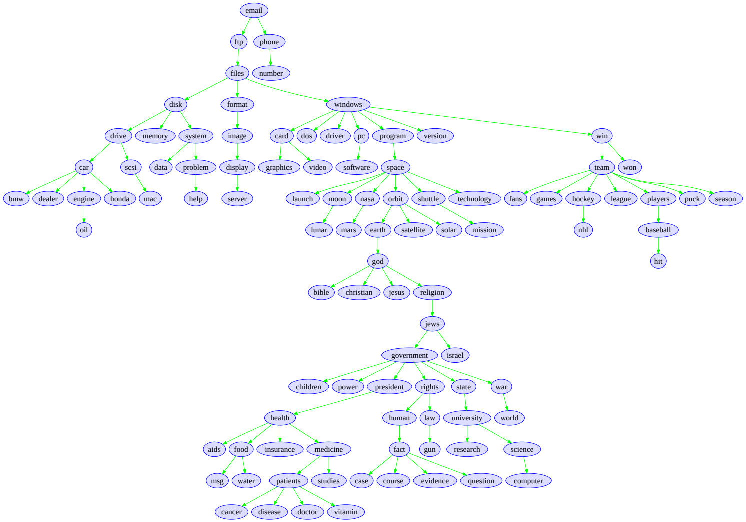
Data containing presence of 100 words from newsgroups posts:

car	drive	files	hockey	mac	league	pc	win
0	0	1	0	1	0	1	0
0	0	0	1	0	1	0	1
1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	0
0	0	1	0	0	0	1	1

Structure learning should give relationship between words.

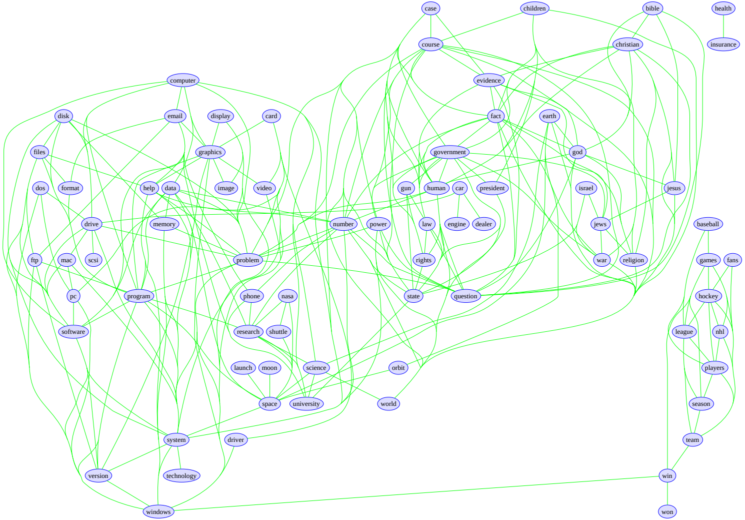
Structure Learning on News Words

Optimal tree on news Words:



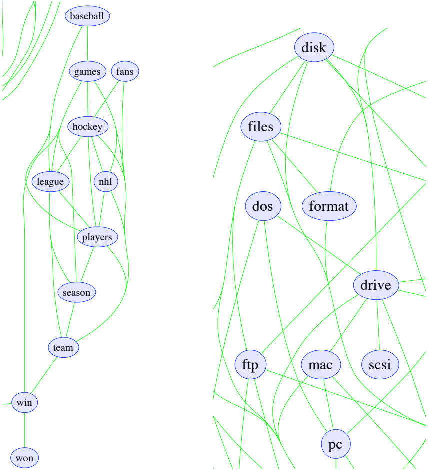
Structure Learning on News Words

Group-L1 on news words:



Structure Learning on News Words

Group-L1 on news words:



Outline

Rain Data without Month Information

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

$$p(y_1, y_2, \dots, y_d) = \frac{1}{Z} \exp\left(\sum_{i=1}^{d} y_i w + \sum_{j=2}^{d} y_j y_{j-1} v\right).$$

- First term reflects that "not rain" is more likely.
- Second term reflects that consecutive days are more likely to be the same.

- But how can we that "some months are less rainy"?

Rain Data with Month Information: Boltzmann Machine

- We could add 12 binary latent variable z_j,

$$p(y_1, y_2, \dots, y_d, z) = \frac{1}{Z}\exp\left(\sum_{i=1}^{d} y_i w + \sum_{i=2}^{d} y_i y_{i-1} v + \sum_{i=1}^{d}\sum_{j=1}^{12} y_i z_j v_2 + \sum_{j=1}^{12} z_j w_2\right),$$

  which is a variaton on a Boltzmann machine.
  - Modifies the probability of "rain" for each of the 12 values.

- Inference is more expensive due to the extra variables.

Rain Data with Month Information: MRF

- If we know the months we could add an explicit month feature x_j

$$p(y_1, y_2, \dots, y_d, x) = \frac{1}{Z}\exp\left(\sum_{i=1}^{d} y_i w + \sum_{i=2}^{d} y_i y_{i-1} v + \sum_{i=1}^{d}\sum_{j=1}^{12} y_i x_j v_2 + \sum_{j=1}^{12} x_j w_2\right),$$

- Learning might be easier: we're given known clusters.
- But still have to model distribution x.
  - It's easy in this case because months are uniform.
  - But in other cases we may want to use a complicated x.
  - And inference is more expensive than chain-structured models.

Rain Data with Month Information: CRF

- In conditional random fields we fit distribution conditioned on x,

$$p(y_1, y_2, \dots, y_d | x) = \frac{1}{Z}\exp\left(\sum_{i=1}^{d} y_i w + \sum_{i=2}^{d} y_i y_{i-1} v + \sum_{i=1}^{d}\sum_{j=1}^{12} y_i x_j v_2\right).$$

  - Now we don't need to model x.
    - Just need to figure out how x affects y.

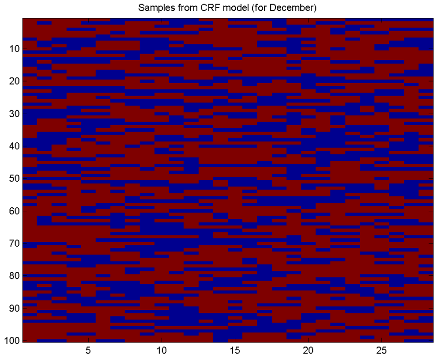
- The conditional UGM given x has a chain-structure

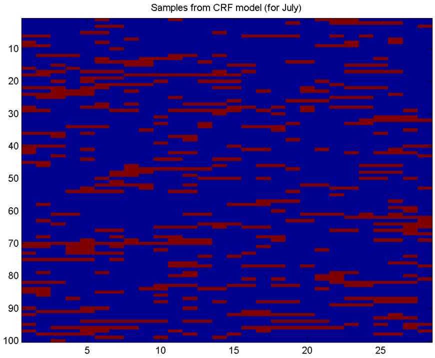
$$\phi_i(y_i) = \exp\left(y_i w + \sum_{j=1}^{12} y_i x_j v_2\right), \quad \phi_{ij}(y_i, y_j) = \exp(y_i y_j v),$$

so inference can be done using forward-backward.

Rain Data with Month Information

- Samples from CRF conditioned on x for December and July:

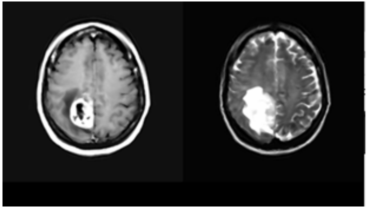




- Code available as part of UGM package.

Brain Tumour Segmentation with Label Dependencies

- We could label all voxels i as "tumour" or not using logistic regression,



$$p(y^i|x^i) = \frac{\exp(y^i w^T x^i)}{\exp(w^T x^i) + \exp(-w^T x^i)}$$

- But this misses dependence in labels y^i:
  - We prefer neighbouring voxels to have the same value.

Brain Tumour Segmentation with Label Dependencies

- With independent logistic, joint distribution over all voxels is

$$p(y^1, y^2, \ldots, y^d | x^1, x^2, \ldots, x^d) = \prod_{i=1}^{d} \frac{\exp(y^i w^T x^i)}{\exp(w^T x^i) + \exp(-w^T x^i)}$$
$$\propto \exp\left(\sum_{i=1}^{d} y^i w^T x^i\right),$$

which is a UGM with no edges,

$$\phi_i(y^i) = \exp(y^i w^T x^i),$$

so given the x^i there is no dependence between the y^i.

Brain Tumour Segmentation with Label Dependencies

- Adding an Ising-like term to model dependencies between y^i gives

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

- 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.

Brain Tumour Segmentation with Label Dependencies

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

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

- 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

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

Conditional Log-Linear Models

- All these CRFs can be written as conditional log-linear models,

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

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

- The NLL is convex and has the form

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

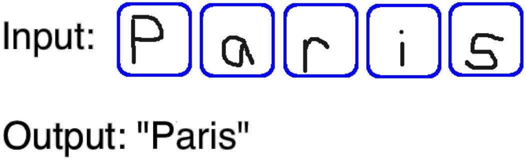
and the gradient can be written as

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

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

Modeling OCR Dependencies

- What dependencies should we model for this problem?



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

Tractability of Discriminative Models

- If the y^i graph is a tree, we can easily fit CRFs.

- But there are other cases where we can fit conditional log-linear models.
  - "Dictionary" feature is non-Markov, but exact computation still easy.
  - We can use pseudo-likelihood or approximate inference.

- 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).

Outline

Learning for Structured Prediction

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 $p(y\|x)$	Logistic regression	CRF
Discriminant function $y = f(x)$	SVM	Structured SVM

- Discriminaitve models don't need to model x.
- Discriminant functions don't worry about probabilities.
  - Based on decoding, which is different than inference in structured case.

SVMs and Likelihood Ratios

- Logistic regression optimizes a likelihood of the form

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

- But if we only want correct decisions it's sufficient

$$\frac{p(y^i|x^i, w)}{p(-y^i|x^i, w)} \geq \kappa,$$

  for any $\kappa > 1$.

- Taking logarithms and plugging in probabilities gives

$$y^i w^T x^i + \log Z - (-y^i w^T x^i) - \log Z \geq \log \kappa$$

- Since κ is arbitrary let's use $\log(\kappa) = 2$,

$$y^i w^T x^i \geq 1.$$

SVMs and Likelihood Ratios

- So to classify all i correctly it's sufficient that

$$y^i w^T x^i \geq 1,$$

  but this linear program may have no solutions.

- To give solution, allow non-negative "slack" r_i and penalize size of r_i,

$$\underset{w,r}{\text{argmin}} \sum_{i=1}^{n} r_i \quad \text{with} \quad y^i w^T x^i \geq 1 - r_i \quad \text{and} \quad r_i \geq 0.$$

- If we apply our Day 2 linear programming trick in reverse this minimizes

$$f(w) = \sum_{i=1}^{n} [1 - y^i w^T x^i]^+$$

  and adding an L2-regularizer gives the standard SVM objective.
  - The notation $[\alpha]^+$ means $\max\{0, \alpha\}$.

Multi-Class SVMs: nk-Slack Formulation

- With multi-class logistic regression we use

$$p(y^i = c | x^i, w) \propto \exp(w_c^T x^i).$$

- If want correct decisions it's sufficient for all $y' \neq y^i$ that

$$\frac{p(y^i | x^i, w)}{p(y' | x^i, w)} \geq \kappa.$$

- Following the same steps as before, this corresponds to

$$w_{y^i}^T x^i - w_{y'}^T x^i \geq 1.$$

- Adding slack variables our linear programming trick gives

$$f(W) = \sum_{i=1}^n \sum_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+,$$

which with L2-regularization we'll call the nk-slack multi-class SVM.

Multi-Class SVMs: n-Slack Formulation

- If want correct decisions it's also sufficent that

$$\frac{p(y^i|x^i,w)}{\max_{y'\neq y^i} p(y'|x^i,w)}.$$

- This leads to the constraints

$$\max_{y'\neq y^i}\{w_{y^i}^T x^i - w_{y'}^T x^i\} \geq 1.$$

- Following the same steps gives an alternate objective

$$f(W) = \sum_{i=1}^n \max_{y'\neq y^i}[1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+,$$

which with L2-regularization we'll call the n-slack multi-class SVM.

Multi-Class SVMs: nk-Slack vs. n-Slack

- Our two formulations of multi-class SVMs:

$$f(W) = \sum_{i=1}^{n} \sum_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+ + \frac{\lambda}{2}\|W\|_F^2,$$

$$f(W) = \sum_{i=1}^{n} \max_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+ + \frac{\lambda}{2}\|W\|_F^2.$$

- The nk-slack loss penalizes based on all y' that could be confused with y^i.
- The n-slack loss only penalizes based on the "most confusing" alternate example.

- While nk-slack often works better, n-slack can be used for structured prediction...

Hidden Markov Support Vector Machines

- For decoding in conditional random fields to entire labeling correct we need

$$\frac{p(y^i|x^i, w)}{p(y'|x^i, w)} \geq \gamma,$$

for all alternative configuraitons y'.

- Following the same steps are before we obtain

$$f(w) = \sum_{i=1}^{n} \max_{y' \neq y}[1 - \log p(y^i|x^i, w) + \log p(y'|x^i, w)]^+ + \frac{\lambda}{2}\|w\|^2,$$

the hidden Markov support vector machine (HMSVM).

- Tries to make log-probability of true y^i greater than for other y' by more than 1.

Hidden Markov Support Vector Machines

- Two problems with the HMSVM:
  1. It requires finding second-best decoding, which is harder than decoding.
  2. It views any alternative labeling y' as equally bad.

- Suppose that $y^i = \begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$, and predictions of two models are

$$y' = \begin{bmatrix}1 & 1 & 0 & 1\end{bmatrix}, \quad y' = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix},$$

  should both models receive the same loss on this example?

Adding a Loss Function

- We can fix both HMSVM issues by replacing the "correct decision" constraint,

$$\log p(y^i|x^i, w) - \log p(y'|x^i, w) \geq 1,$$

with a constraint containing a loss function g,

$$\log p(y^i|x^i, w) - \log p(y'|x^i, w) \geq g(y^i, y').$$

- Usually we take $g(y^i, y')$ to be the difference between y^i and y'.
- If $g(y^i, y^i) = 0$, you can maximize over all y' instead of $y' \neq y^i$.
  - Further, if g is written as sum of functions depending on the graph edges, finding "most violated" constraint is equivalent to decoding.

Structure SVMs

- These constraints lead to the max-margin Markov network objective,

$$f(w) = \sum_{i=1}^{n} \max_{y'} [g(y^i, y') - \log p(y^i | x^i, w) + \log p(y' | x^i, w)]^+ + \frac{\lambda}{2} \|w\|^2,$$

  which is also known as a structured SVM.

- Beyond learning principle, key differences between CRFs and SSVMs:
  - SSVMs require decoding, not inference, for learning:
    - Exact SSVMs in cases like graph cuts, matchings, rankings, etc.
  - SSVMs have loss function for complicated accuracy measures:
    - But loss needs to decompose over parts for tractability.
    - Could also formulate 'loss-augmented' CRFs.

- We can also train with approximate decoding methods.
  - State of the art training: block-coordinate Frank Wolfe (bonus slides).

Summary

- Log-linear models are the most common UGM when learning parameters.
- Structure learning is the problem of learning the graph structure.
  - Hard in general, but L1-regularization gives a fast heuristic.
- Conditional log-linear models are the most common CRF models.
  - But you can fit some non-Markov models too.
- Structured SVMs are a generalization of SVMs to structured prediction.
  - Only require decoding instead of inference.

- Next time: convolutional neural networks.

Bonus Slide: SVMs for Ranking with Pairwise Preference

- Suppose we want to rank examples.
- A common setting is with features x^i and pairwise preferences:
  - List of objects (i, j) where we want $y^i > y^j$.
- Assuming a log-linear model,

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

we can derive a loss function based on the pairwise preference decisiosn,

$$\frac{p(y^i|x^i, w)}{p(y^j|x^j, w)} \geq \gamma,$$

which gives a loss function of the form

$$f(w) = \sum_{(i,j) \in R} [1 - w^T x^i + w^T x^j]^+.$$

Bonus Slide: Fitting Structured SVMs

Overview of progress on training SSVMs:

- Cutting plane and bundle methods (e.g., svmStruct software):
  - Require $O(1/\epsilon)$ iterations.
  - Each iteration requires decoding on every training example.
- Stochastic sub-gradient methods:
  - Each iteration requires decoding on a single training example.
  - Still requires $O(1/\epsilon)$ iterations.
  - Need to choose step size.
- Dual Online exponentiated gradient (OEG):
  - Allows line-search for step size and has $O(1/\epsilon)$ rate.
  - Each iteration requires inference on a single training example.
- Dual block-coordinate Frank-Wolfe (BCFW):
  - Each iteration requires decoding on a single training example.
  - Requires $O(1/\epsilon)$ iterations.
  - Closed-form optimal step size.
  - Theory allows approximate decoding.

Bonus Slide: Block Coordinate Frank Wolfe

Key ideas behind BCFW for SSVMs:

- Dual problem has as the form

$$\min_{\alpha_i \in \mathcal{M}_i} F(\alpha) = f(A\alpha) - \sum_i f_i(\alpha_i).$$

  where f is smooth.
- Problem structure where we can use block coordinate descent:
  - Normal coordinate updates intractable because $\alpha_i \in |\mathcal{Y}|$.
  - But Frank-Wolfe block-coordinate update is equivalent to decoding

$$s = \underset{s' \in \mathcal{M}_i}{\text{argmin}}\, F(\alpha) + \langle \nabla_i F(\alpha), s' - \alpha_i \rangle.$$

$$\alpha_i = \alpha_i - \gamma(s - \alpha_i).$$

  - Can implement algorithm in terms of primal variables.
- Connections between Frank-Wolfe and other algorithms:
  - Frank-Wolfe on dual problem is subgradient step on primal.
  - 'Fully corrective' Frank-Wolfe is equivalent to cutting plane.